お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

M16C/80 シリーズ
プログラム作成の手引き < アセンブリ言語編 >

はじめに

「M16C/80 シリーズ　アセンブリ言語プログラミングマニュアル」は、ルネサス CMOS16 ビットマイクロコンピュータ M16C/80 シリーズ用のアプリケーションプログラム開発において必要となる、基本的な知識を説明しています。本書のプログラミング言語はアセンブリ言語です。

M16C/80 シリーズを初めてお使いになる方は「第 1 章　M16C/80 シリーズの概要」を、CPU のアーキテクチャと命令については知りたい方は「第 2 章　CPU プログラミングモデル」を、アセンブラ指示命令について知りたい方は「第 3 章　アセンブラの機能」を、実践的なテクニックを知りたい方は「第 4 章　プログラミングスタイル」をご参照ください。M16C/80 シリーズの命令体系については、「M16C/80 シリーズ　ソフトウェアマニュアル」でさらに詳しく説明していますので、合わせてご利用ください。

なお、M16C/80 シリーズ各品種のハードウェアにつきましてはご使用品種のユーザーズマニュアルを、開発サポートツールにつきましては各ツールの操作説明書をご利用ください。

本書の使い方

本書は M16C/80 シリーズのアセンブリ言語プログラミングマニュアルです。M16C/80 シリーズの CPU コアをもつ全品種で共通に使用することができます。

本書を使用するにあたって、電気回路、論理回路、およびマイクロコンピュータの基本的な知識が必要です。

本書は 4 つの章から構成されています。以下に目的に応じた参照先（章、節）を示します。

○ M16C/80 シリーズの概要、特長を理解したい

→ 第 1 章　M16C/80 シリーズの概要

○アドレス空間、レジスタ構成、アドレッシングなどプログラミングに必要な事項を理解したい

→第 2 章　CPU プログラミングモデル

○ 命令の機能、書き方、使用できるアドレッシングを確認したい

→ 第 2 章　CPU プログラミングモデル　2.6　命令セット

○ 割り込みの使い方を知りたい

→ 第 2 章　CPU プログラミングモデル　2.7　割り込みの概要

→ 第 4 章　プログラミングスタイル　　4.3　割り込み使用時の設定方法

○ 指示命令の機能、使い方を確認したい

→ 第 3 章　アセンブラの機能　3.2　ソースプログラムの書き方

○ M16C/80 シリーズのプログラミングテクニックを知りたい

→ 第 4 章　プログラミングスタイル　　4.5　ちょっと小耳を ...（ﾌﾟﾛｸﾞﾗﾐﾝｸﾞﾃｸﾆｯｸ）

○ M16C/80 シリーズの開発手順を知りたい

→ 付録　AS308 システムのコマンド入力形式とコマンドパラメータ

目次

図目次

表目次

第　1　章

M16C/80 シリーズの概要

1.1 M16C/80 シリーズの特長

M16C/80 シリーズは組み込み機器を対象として開発されたシングルチップマイクロコンピュータです。
この節では M16C/80 シリーズの特長を紹介します。

M16C/80 シリーズの特長

M16C/80 シリーズは、使用頻度の高い命令を 1 バイトオペコードに配置しています。そのため、より小さなメモリ容量で効率の良いプログラムを書くことができます。

さらに、M16C/80 シリーズは 16 ビットマイコンでありながら 1 ビット /4 ビット /8 ビット /32 ビット処理を効率よく行います。特に 32 ビット処理は M16C/60 シリーズより高効率ととなってます。 1 クロックで実行できる命令も多数用意されています。そのため、高速な処理プログラムを書くことができます。

M16C/80 シリーズはリニアな 16M バイトのアドレス空間を持っています。したがって、プログラムサイズが大きな用途にも対応できます。

M16C/80 シリーズの特長を以下にまとめます。

(1)少ないメモリ容量で効率の良いプログラムを実現できる
(2)高速処理プログラムを実現できる
(3)16M バイトのアドレス空間を持っている

1.2 M16C/80 グループの概要

この節では M16C/80 シリーズの内部構成の例として M16C/80 グループを紹介します。M16C/80 グループは M16C/80 シリーズのベースとなる製品です。詳細については、各品種のデータシート、ユーザーズマニュアルをご参照ください。

内部ブロック図

図 1.2.1 に M16C/80 グループのブロック図を示します。

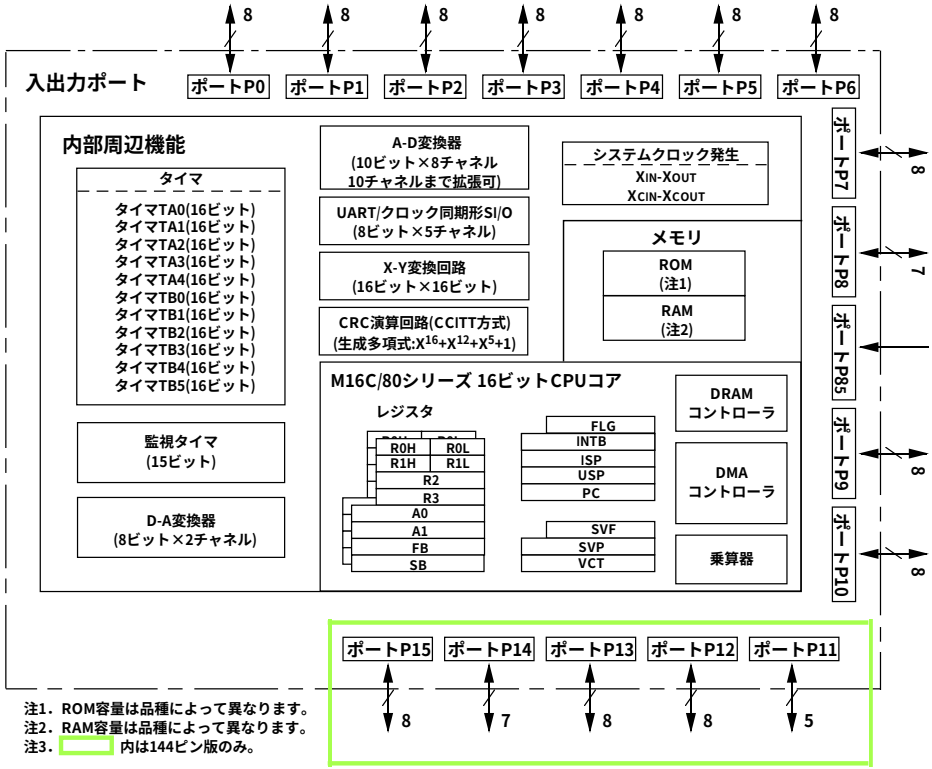


図 1.2.1 M16C/80 グループのブロック図

M16C/80 グループの概略仕様

表 1.2.1 に M16C/80 グループの概略仕様を示します。

表 1.2.1　M16C/80 グループの概略仕様

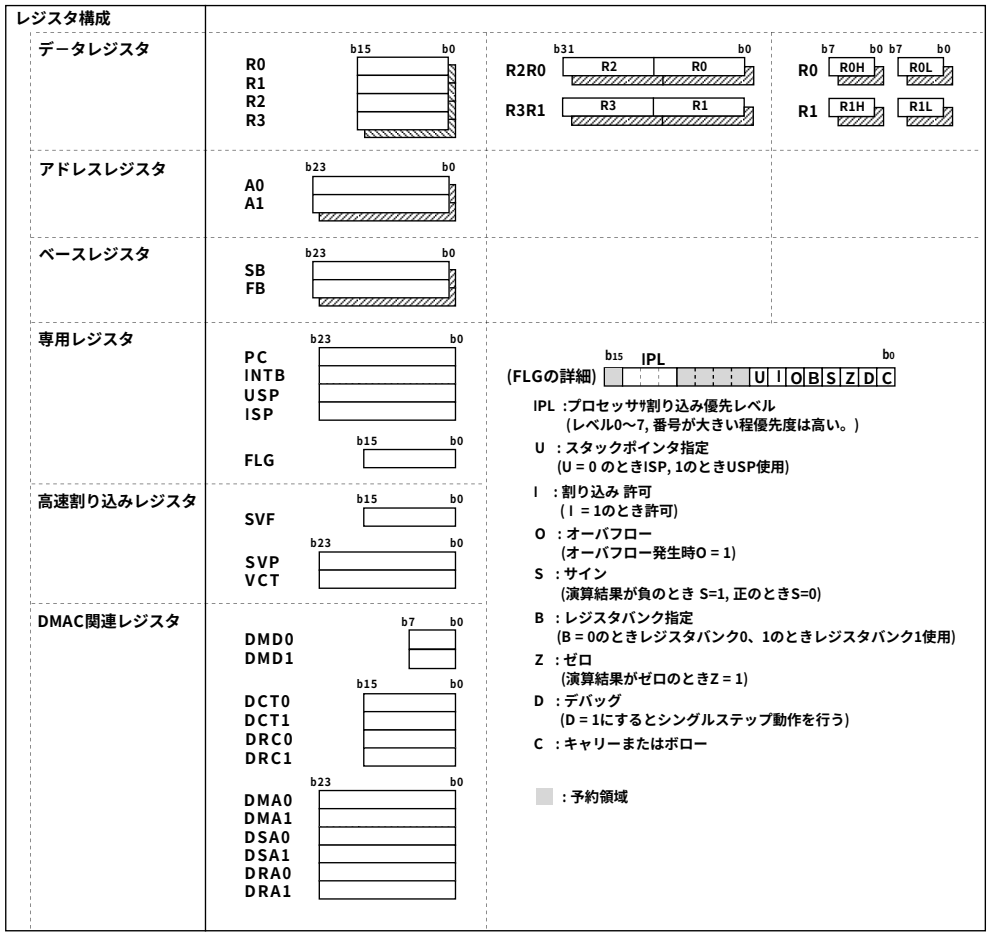
項　　目		性　　能
基本命令数		106命令
最短命令実行時間		50ns(f(XIN)=20MHz時)
メモリ容量	ROM	128Kバイト
	RAM	10Kバイト
入出力ポート	P0～P10	8ビット×10、7ビット×1(100ピン版) (ただしP85は除く)
	P0～P15	8ビット×13、7ビット×1、5ビット×1 (144ピン版)(ただしP85は除く)
入力ポート	P85	1ビット×1
多機能タイマ	TA0,TA1,TA2,TA3,TA4 TB0,TB1,TB2,TB3,TB4,TB5	16ビット×5 16ビット×6
シリアルI/O	UART0,UART1,UART2 UART3,UART4	(UARTまたはクロック同期形)×5
A-D変換器		10ビット×(8+2)チャネル
D-A変換器		8ビット×2
DMAC		4チャネル
DRAMコントローラ		CASビフォアRASリフレッシュ、 セルフリフレッシュ、EDO、FP対応
CRC演算回路		CRC-CCITT方式
X-Y変換回路		16ビット×16ビット
監視タイマ		15ビット×1(プリスケーラ付)
割り込み		内部29要因、外部8要因、ソフトウエア4要因、 7レベル
クロック発生回路		2回路内蔵(帰還抵抗内蔵、セラミック共振子、 または水晶発振子外付け)
メモリ拡張		可能（16Mバイト）

（注）M30800MC の場合です。メモリ容量の詳細はデータシートおよびユーザーズマニュアルをご参照ください。

レジスタ構成

レジスタ構成を表1.2.2に示します。レジスタR0、R1、R2、R3、A0、A1、SB、FBの8本は、2セット用意されています。これらはレジスタバンク指定フラグによって切り替わります。

表 1.2.2　M16C/80 シリーズのレジスタ構成

項目	内　容
レジスタ構成	

第 2 章

CPU プログラミングモデル

2.1　アドレス空間

M16C/80 シリーズのアドレス空間は 000000H 番地から FFFFFFH 番地までの 16M バイトです。ここでは M16C/80 グループのアドレス空間とメモリ配置、SFR 領域、固定ベクタ領域について説明します。

アドレス空間

M16C/80 グループのアドレス空間を図 2.1.1 に示します。

000000H 番地～0003FFH 番地はSFR（スペシャルファンクションレジスタ）領域です。 品種展開では0003FFH 番地から番地の小さい方向に SFR 領域を拡張します。

000400H 番地以降はメモリ領域です。品種展開では 000400H 番地から番地の大きい方向に RAM 領域を、FFFFFFH 番地から番地の小さい方向に ROM 領域を拡張します。ただし、FFFE00H 番地～ FFFFFFH 番地は固定ベクタ領域です。

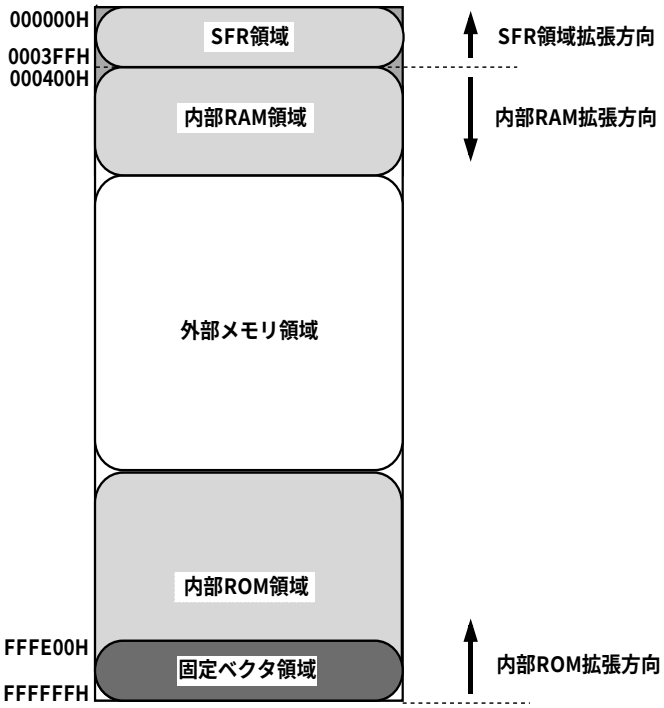


図 2.1.1　アドレス空間

2.1.1 動作モードとメモリ配置

M16C/80 グループは、シングルチップモード、メモリ拡張モード、マイクロプロセッサモードの 3 つのモードから 1 つの動作モードを選択します。動作モードによってアドレス空間と使用できる領域とメモリ配置が異なります。

動作モードとメモリ配置

●シングルチップモード

シングルチップモードは、内部領域（SFR、内部 RAM、内部 ROM）だけのアクセスが可能なモードです。

●メモリ拡張モード

メモリ拡張モードは、内部領域（SFR、内部 RAM、内部 ROM）および外部メモリ領域のアクセスが可能なモードです。

●マイクロプロセッサモード

マイクロプロセッサモードは、SFR および内部 RAM 領域と外部メモリ領域のアクセスが可能なモードです（内部 ROM 領域はアクセスできません）。

各動作モードのメモリ配置を図 2.1.2 に示します。

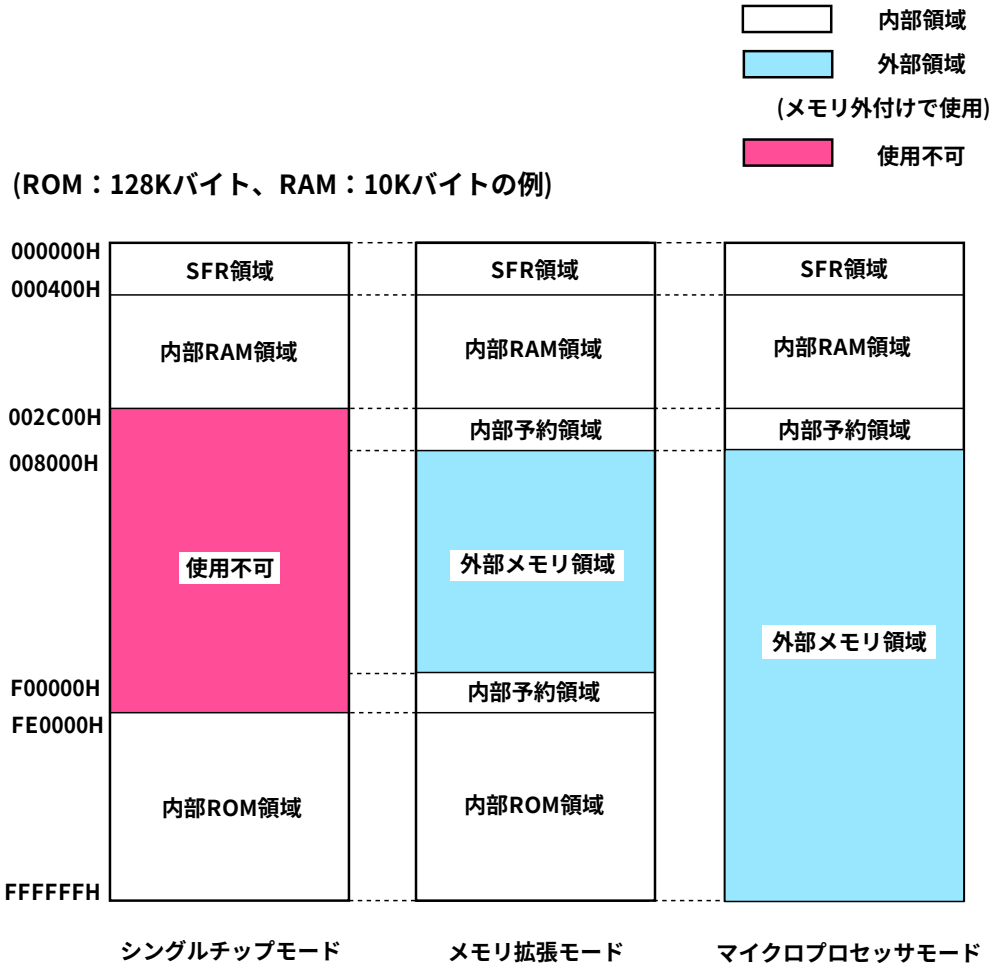


図 2.1.2　動作モードとメモリ配置

2.1.2 SFR 領域

SFR 領域には、動作モードを決定するプロセッサモードレジスタや、入出力ポート、A-D 変換器、UART、タイマなど周辺装置の制御レジスタが割り付けられています。制御レジスタのビット構成については M16C/80 グループのデータシートおよびユーザーズマニュアルをご参照ください。

なお、SFR 領域内の未使用領域は予約領域となっているため使用できません。

SFR 領域　制御レジスタの配置(100 ピン版)

SFR 領域　制御レジスタの配置を図 2.1.3、2.1.4、2.1.5、2.1.6 に示します。

000016	
000116	
000216	
000316	
000416	プロセッサモードレジスタ0(PM0)
000516	プロセッサモードレジスタ1(PM1)
000616	システムクロック制御レジスタ0(CM0)
000716	システムクロック制御レジスタ1(CM1)
000816	ウエイト制御レジスタ(WCR)
000916	アドレス一致割り込み許可レジスタ(AIER)
000A16	プロテクトレジスタ(PRCR)
000B16	外部データバス幅制御レジスタ(DS)
000C16	メインクロック分周レジスタ(MCD)
000D16	
000E16	監視タイマスタートレジスタ(WDTS)
000F16	監視タイマ制御レジスタ(WDC)
001016	アドレス一致割り込みレジスタ0(RMAD0)
001116	
001216	
001316	
001416	アドレス一致割り込みレジスタ1(RMAD1)
001516	
001616	
001716	
001816	アドレス一致割り込みレジスタ2(RMAD2)
001916	
001A16	
001B16	
001C16	アドレス一致割り込みレジスタ3(RMAD3)
001D16	
001E16	
001F16	
002016	エミュレータ専用割り込みベクタテーブル
002116	レジスタ(EIAD) ＊
002216	
002316	エミュレータ割り込み識別レジスタ(EITD) ＊
002416	エミュレータ用プロテクトレジスタ(EPRR) ＊
002516	
002616	
002716	
002816	
002916	
002A16	
002B16	
002C16	
002D16	
002E16	
002F16	
003016	ROM領域設定レジスタ(ROA) ＊
003116	デバッグモニタ領域設定レジスタ(DBA) ＊
003216	拡張領域設定レジスタ0(EXA0) ＊
003316	拡張領域設定レジスタ1(EXA1) ＊
003416	拡張領域設定レジスタ2(EXA2) ＊
003516	拡張領域設定レジスタ3(EXA3) ＊
003616	
003716	
003816	
003916	
003A16	
003B16	
003C16	
003D16	
003E16	
003F16	
004016	DRAM制御レジスタ(DRAMCONT)
004116	DRAMリフレッシュ間隔設定レジスタ(REFCNT)
004216	
004316	
004416	

006016	
006116	
006216	
006316	
006416	
006516	
006616	
006716	
006816	DMA0割り込み制御レジスタ (DM0IC)
006916	タイマB5割り込み制御レジスタ(TB5IC)
006A16	DMA2割り込み制御レジスタ (DM2IC)
006B16	UART2受信/ACK割り込み制御レジスタ(S2RIC)
006C16	タイマA0割り込み制御レジスタ(TA0IC)
006D16	UART3受信/ACK割り込み制御レジスタ(S3RIC)
006E16	タイマA2割り込み制御レジスタ(TA2IC)
006F16	UART4受信/ACK割り込み制御レジスタ(S4RIC)
007016	タイマA4割り込み制御レジスタ(TA4IC)
007116	バス衝突検出(UART3)割り込み制御レジスタ (BCN3IC)
007216	UART0受信割り込み制御レジスタ(S0RIC)
007316	A-D変換割り込み制御レジスタ(ADIC)
007416	UART1受信割り込み制御レジスタ(S1RIC)
007516	
007616	タイマB1割り込み制御レジスタ(TB1IC)
007716	
007816	タイマB3割り込み制御レジスタ(TB3IC)
007916	
007A16	INT5割り込み制御レジスタ(INT5IC)
007B16	
007C16	INT3割り込み制御レジスタ(INT3IC)
007D16	
007E16	INT1割り込み制御レジスタ(INT1IC)
007F16	
008016	
008116	
008216	
008316	
008416	
008516	
008616	
008716	
008816	DMA1割り込み制御レジスタ(DM1IC)
008916	UART2送信/NACK割り込み制御レジスタ(S2TIC)
008A16	DMA3割り込み制御レジスタ (DM3IC)
008B16	UART3送信/NACK割り込み制御レジスタ(S3TIC)
008C16	タイマA1割り込み制御レジスタ(TA1IC)
008D16	UART4送信/NACK割り込み制御レジスタ(S4TIC)
008E16	タイマA3割り込み制御レジスタ(TA3IC)
008F16	バス衝突検出(UART2)割り込み制御レジスタ (BCN2IC)
009016	UART0送信割り込み制御レジスタ(S0TIC)
009116	バス衝突検出(UART4)割り込み制御レジスタ (BCN4IC)
009216	UART1送信割り込み制御レジスタ(S1TIC)
009316	キー入力割り込み制御レジスタ(KUPIC)
009416	タイマB0割り込み制御レジスタ(TB0IC)
009516	
009616	タイマB2割り込み制御レジスタ(TB2IC)
009716	
009816	タイマB4割り込み制御レジスタ(TB4IC)
009916	
009A16	INT4割り込み制御レジスタ(INT4IC)
009B16	
009C16	INT2割り込み制御レジスタ(INT2IC)
009D16	
009E16	INT0割り込み制御レジスタ(INT0IC)
009F16	復帰用優先順位レジスタ (RLVL)
00A016	
00A116	
00A216	
00A316	
00A416	

＊　エミュレータ専用のレジスタですので、ユーザーは使用できません。アクセスしないでください。

図 2.1.3　制御レジスタの配置 1

02C0H 02C1H	X0レジスタ(X0R) Y0レジスタ(Y0R)
02C2H 02C3H	X1レジスタ(X1R) Y0レジスタ(Y1R)
02C4H 02C5H	X2レジスタ(X2R) Y2レジスタ(Y2R)
02C6H 02C7H	X3レジスタ(X3R) Y3レジスタ(Y3R)
02C8H 02C9H	X4レジスタ(X4R) Y4レジスタ(Y4R)
02CAH 02CBH	X5レジスタ(X5R) Y5レジスタ(Y5R)
02CCH 02CDH	X6レジスタ(X6R) Y6レジスタ(Y6R)
02CEH 02CFH	X7レジスタ(X7R) Y7レジスタ(Y7R)
02D0H 02D1H	X8レジスタ(X8R) Y8レジスタ(Y8R)
02D2H 02D3H	X9レジスタ(X9R) Y9レジスタ(Y9R)
02D4H 02D5H	X10レジスタ(X10R) Y10レジスタ(Y10R)
02D6H 02D7H	X11レジスタ(X11R) Y11レジスタ(Y11R)
02D8H 02D9H	X12レジスタ(X12R) Y12レジスタ(Y12R)
02DAH 02DBH	X13レジスタ(X13R) Y13レジスタ(Y13R)
02DCH 02DDH	X14レジスタ(X14R) Y14レジスタ(Y14R)
02DEH 02DFH	X15レジスタ(X15R) Y15レジスタ(Y15R)
02E0H	XY制御レジスタ(XYC)
02E1H	
02E2H	
02E3H	
02E4H	
02E5H	
02E6H	
02E7H	
02E8H	
02E9H	
02EAH	
02EBH	
02ECH	
02EDH	
02EEH	
02EFH	
02F0H	
02F1H	
02F2H	
02F3H	
02F4H	
02F5H	
02F6H	UART4特殊モードレジスタ2(U4SMR2)
02F7H	UART4特殊モードレジスタ(U4SMR)
02F8H	UART4送受信モードレジスタ(U4MR)
02F9H	UART4転送速度レジスタ(U4BRG)
02FAH 02FBH	UART4送信バッファレジスタ(U4TB)
02FCH	UART4送受信制御レジスタ0(U4C0)
02FDH	UART4送受信制御レジスタ1(U4C1)
02FEH 02FFH	UART4受信バッファレジスタ(U4RB)

0300H	タイマB3,4,5カウント開始フラグ(TBSR)
0301H	
0302H 0303H	タイマA1-1レジスタ(TA11)
0304H 0305H	タイマA2-1レジスタ(TA21)
0306H 0307H	タイマA4-1レジスタ(TA41)
0308H	三相PWM制御レジスタ0(INVC0)
0309H	三相PWM制御レジスタ1(INVC1)
030AH	三相出力バッファレジスタ0(IDB0)
030BH	三相出力バッファレジスタ1(IDB1)
030CH	短絡防止タイマ(DTT)
030DH	タイマB2割り込み発生頻度設定カウンタ(ICTB2)
030EH	
030FH	
0310H 0311H	タイマB3レジスタ(TB3)
0312H 0313H	タイマB4レジスタ(TB4)
0314H 0315H	タイマB5レジスタ(TB5)
0316H	
0317H	
0318H	
0319H	
031AH	
031BH	タイマB3モードレジスタ(TB3MR)
031CH	タイマB4モードレジスタ(TB4MR)
031DH	タイマB5モードレジスタ(TB5MR)
031EH	
031FH	割り込み要因選択レジスタ(IFSR)
0320H	
0321H	
0322H 0323H	
0324H	
0325H	
0326H	UART3特殊モードレジスタ2(U3SMR2)
0327H	UART3特殊モードレジスタ(U3SMR)
0328H	UART3送受信モードレジスタ(U3MR)
0329H	UART3転送速度レジスタ(U3BRG)
032AH 032BH	UART3送信バッファレジスタ(U3TB)
032CH	UART3送受信制御レジスタ0(U3C0)
032DH	UART3送受信制御レジスタ1(U3C1)
032EH 032FH	UART3受信バッファレジスタ(U3RB)
0330H	
0331H	
0332H	
0333H	
0334H	
0335H	
0336H	UART2特殊モードレジスタ2(U2SMR2)
0337H	UART2特殊モードレジスタ(U2SMR)
0338H	UART2送受信モードレジスタ(U2MR)
0339H	UART2転送速度レジスタ(U2BRG)
033AH 033BH	UART2送信バッファレジスタ(U2TB)
033CH	UART2送受信制御レジスタ0(U2C0)
033DH	UART2送受信制御レジスタ1(U2C1)
033EH 033FH	UART2受信バッファレジスタ(U2RB)

図 2.1.4　制御レジスタの配置 2

0340₁₆	カウント開始フラグ(TABSR)
0341₁₆	時計用プリスケーラリセットフラグ(CPSRF)
0342₁₆	ワンショット開始フラグ(ONSF)
0343₁₆	トリガ選択レジスタ(TRGSR)
0344₁₆	アップダウンフラグ(UDF)
0345₁₆	
0346₁₆	タイマA0(TA0)
0347₁₆	
0348₁₆	タイマA1(TA1)
0349₁₆	
034A₁₆	タイマA2(TA2)
034B₁₆	
034C₁₆	タイマA3(TA3)
034D₁₆	
034E₁₆	タイマA4(TA4)
034F₁₆	
0350₁₆	タイマB0(TB0)
0351₁₆	
0352₁₆	タイマB1(TB1)
0353₁₆	
0354₁₆	タイマB2(TB2)
0355₁₆	
0356₁₆	タイマA0モードレジスタ(TA0MR)
0357₁₆	タイマA1モードレジスタ(TA1MR)
0358₁₆	タイマA2モードレジスタ(TA2MR)
0359₁₆	タイマA3モードレジスタ(TA3MR)
035A₁₆	タイマA4モードレジスタ(TA4MR)
035B₁₆	タイマB0モードレジスタ(TB0MR)
035C₁₆	タイマB1モードレジスタ(TB1MR)
035D₁₆	タイマB2モードレジスタ(TB2MR)
035E₁₆	
035F₁₆	
0360₁₆	UART0送受信モードレジスタ(U0MR)
0361₁₆	UART0転送速度レジスタ(U0BRG)
0362₁₆	UART0送信バッファレジスタ(U0TB)
0363₁₆	
0364₁₆	UART0送受信制御レジスタ０(U0C0)
0365₁₆	UART0送受信制御レジスタ１(U0C1)
0366₁₆	UART0受信バッファレジスタ(U0RB)
0367₁₆	
0368₁₆	UART1送受信モードレジスタ(U1MR)
0369₁₆	UART1転送速度レジスタ(U1BRG)
036A₁₆	UART1送信バッファレジスタ(U1TB)
036B₁₆	
036C₁₆	UART1送受信制御レジスタ0(U1C0)
036D₁₆	UART1送受信制御レジスタ1(U1C1)
036E₁₆	UART1受信バッファレジスタ(U1RB)
036F₁₆	
0370₁₆	UART送受信制御レジスタ2(UCON)
0371₁₆	
0372₁₆	
0373₁₆	
0374₁₆	
0375₁₆	
0376₁₆	フラッシュメモリ選択レジスタ1(FMR1)(注1)
0377₁₆	フラッシュメモリ選択レジスタ0(FMR0)(注1)
0378₁₆	DMA0要因選択レジスタ(DM0SL)
0379₁₆	DMA1要因選択レジスタ(DM1SL)
037A₁₆	DMA2要因選択レジスタ(DM2SL)
037B₁₆	DMA3要因選択レジスタ(DM3SL)
037C₁₆	CRCデータレジスタ(CRCD)
037D₁₆	
037E₁₆	CRCインプットレジスタ(CRCIN)
037F₁₆	

0380₁₆	A-Dレジスタ0(AD0)
0381₁₆	
0382₁₆	A-Dレジスタ1(AD1)
0383₁₆	
0384₁₆	A-Dレジスタ2(AD2)
0385₁₆	
0386₁₆	A-Dレジスタ3(AD3)
0387₁₆	
0388₁₆	A-Dレジスタ4(AD4)
0389₁₆	
038A₁₆	A-Dレジスタ5(AD5)
038B₁₆	
038C₁₆	A-Dレジスタ6(AD6)
038D₁₆	
038E₁₆	A-Dレジスタ7(AD7)
038F₁₆	
0390₁₆	
0391₁₆	
0392₁₆	
0393₁₆	
0394₁₆	A-D制御レジスタ2(ADCON2)
0395₁₆	
0396₁₆	A-D制御レジスタ0(ADCON0)
0397₁₆	A-D制御レジスタ1(ADCON1)
0398₁₆	D-Aレジスタ0(DA0)
0399₁₆	
039A₁₆	D-Aレジスタ1(DA1)
039B₁₆	
039C₁₆	D-A制御レジスタ(DACON)
039D₁₆	
039E₁₆	
039F₁₆	
03A0₁₆	
03A1₁₆	
03A2₁₆	
03A3₁₆	
03A4₁₆	
03A5₁₆	
03A6₁₆	
03A7₁₆	
03A8₁₆	
03A9₁₆	
03AA₁₆	
03AB₁₆	
03AC₁₆	
03AD₁₆	
03AE₁₆	
03AF₁₆	機能選択レジスタC(PSC)
03B0₁₆	機能選択レジスタA0(PS0)
03B1₁₆	機能選択レジスタA1(PS1)
03B2₁₆	機能選択レジスタB0(PSL0)
03B3₁₆	機能選択レジスタB1(PSL1)
03B4₁₆	機能選択レジスタA2(PS2)
03B5₁₆	機能選択レジスタA3(PS3)
03B6₁₆	機能選択レジスタB2(PSL2)
03B7₁₆	機能選択レジスタB3(PSL3)
03B8₁₆	
03B9₁₆	
03BA₁₆	
03BB₁₆	
03BC₁₆	
03BD₁₆	
03BE₁₆	
03BF₁₆	

注1．このレジスタはフラッシュメモリ版にのみ存在します。

図 2.1.5　制御レジスタの配置 3

03C0H	ポートP6(P6)
03C1H	ポートP7(P7)
03C2H	ポートP6方向レジスタ(PD6)
03C3H	ポートP7方向レジスタ(PD7)
03C4H	ポートP8(P8)
03C5H	ポートP9(P9)
03C6H	ポートP8方向レジスタ(PD8)
03C7H	ポートP9方向レジスタ(PD9)
03C8H	ポートP10(P10)
03C9H	
03CAH	ポートP10方向レジスタ(PD10)
03CBH	
03CCH	
03CDH	
03CEH	
03CFH	
03D0H	
03D1H	
03D2H	
03D3H	
03D4H	
03D5H	
03D6H	
03D7H	
03D8H	
03D9H	
03DAH	プルアップ制御レジスタ２(PUR2)
03DBH	プルアップ制御レジスタ３(PUR3)
03DCH	
03DDH	
03DEH	
03DFH	
03E0H	ポートP0(P0)
03E1H	ポートP1(P1)
03E2H	ポートP0方向レジスタ(PD0)
03E3H	ポートP1方向レジスタ(PD1)
03E4H	ポートP2(P2)
03E5H	ポートP3(P3)
03E6H	ポートP2方向レジスタ(PD2)
03E7H	ポートP3方向レジスタ(PD3)
03E8H	ポートP4(P4)
03E9H	ポートP5(P5)
03EAH	ポートP4方向レジスタ(PD4)
03EBH	ポートP5方向レジスタ(PD5)
03ECH	
03EDH	
03EEH	
03EFH	
03F0H	プルアップ制御レジスタ0(PUR0)
03F1H	プルアップ制御レジスタ1(PUR1)
03F2H	
03F3H	
03F4H	
03F5H	
03F6H	
03F7H	
03F8H	
03F9H	
03FAH	
03FBH	
03FCH	
03FDH	
03FEH	
03FFH	ポート制御レジスタ(PCR)

注1. 03C9H, 03CBH～03D3H番地の領域は将来展開予定の製品のために用意しているものです。
03CBH, 03CEH, 03CFH, 03D2H, 03D3H番地には必ず “ FFH ” を初期設定してください。

図 2.1.6　制御レジスタの配置 4

動作モードの決定

M16C/80 グループの動作モードは、CNVss 端子、プロセッサモードレジスタ 0（000004H 番地）のビット 0 とビット 1 で決定します。

図 2.1.7 に M16C/80 グループのプロセッサモードレジスタ 0 を示します。

プロセッサモードレジスタ 0 (注1)

b7	b6	b5	b4	b3	b2	b1	b0
	0						

シンボル	アドレス	リセット時
PM0	0004$_{16}$番地	80$_{16}$(注2)

ビットシンボル	ビット名	機　能	R	W
PM00	プロセッサモードビット	b1 b0 0 0 : シングルチップモード 0 1 : メモリ拡張モード 1 0 : 使用禁止 1 1 : マイクロプロセッサモード	○	○
PM01			○	○
PM02	R/Wモード選択ビット(注7)	0 : $\overline{RD}$,$\overline{BHE}$,$\overline{WR}$ 1 : RD,WRH,WRL	○	○
PM03	ソフトウエアリセットビット	このビットに“1”を書き込むとマイクロコンピュータはリセットされる。 読み出し時の値は“0”。	○	○
PM04	マルチプレクスバス空間選択ビット(注3)	b5 b4 0 0 : マルチプレクスバスを使用しない 0 1 : CS2の空間に割り当てる 1 0 : CS1の空間に割り当てる 1 1 : CS全空間に割り当てる (注4)	○	○
PM05			○	○
予約ビット		必ず“0”を設定してください。	○	○
PM07	クロック出力機能選択ビット(注5)	0 : BCLK(注6) 1 : システムクロック制御レジスタ0のビット0、1で決定される機能	○	○

注1. このレジスタを書き替える場合、プロテクトレジスタ(000A$_{16}$番地)のビット1を“1”にしてください。
注2. CNVss端子にVccレベルを印加しているときは、リセット時03$_{16}$になります
(PM00およびPM01が“1”に、PM07が“0”になります)。
注3. マイクロプロセッサモード、メモリ拡張モードでモード1、2、3、選択時有効。
モード0選択時、マルチプレクスバスを使用しないでください。
モード2選択時、$\overline{CS2}$空間に割り当てる設定をしないでください。
注4. リセット解除後、セパレートバスで動作しますので、マイクロプロセッサモード時、$\overline{CS}$全空間マルチプレクスバスは選択できません。
メモリ拡張モード時、$\overline{CS}$全空間マルチプレクスバスを選択した場合、
アドレスバスはチップセレクトごとに64Kバイトの範囲です。
モード0選択時、マルチプレクスバスは使用できません。
モード1選択時、$\overline{CS}$全空間を選択すると、$\overline{CS0}$～$\overline{CS2}$が該当します。
モード2選択時、$\overline{CS}$全空間を選択すると、$\overline{CS0}$、$\overline{CS1}$が該当します。
モード3選択時、$\overline{CS}$全空間を選択すると、$\overline{CS0}$～$\overline{CS3}$が該当します。
注5. シングルチップモード時、PM07に“0”を設定してもBCLKは出力されません。
マイクロプロセッサ/メモリ拡張モード時にクロック出力を停止する場合は、PM07=“1”、システムクロック制御レジスタ0のビット0(CM00)およびビット1(CM01)を“0”に設定してください。このとき、P5$_3$からは“L”が出力されます。
注6. BCLKについては、データシートをご参照ください。
注7. DRAMコントローラでデータバスを16ビット幅で使用する場合は“1”を選択してください。

図 2.1.7　プロセッサモードレジスタ 0

2.1.3　固定ベクタ領域

M16C/80 グループの固定ベクタ領域は FFFE00H 番地から FFFFFFH 番地です。

固定ベクタ領域の FFFE00H 番地から FFFFDBH 番地はスペシャルページベクタテーブルです。ここにサブルーチンの先頭番地またはジャンプ先を格納すると、サブルーチンコール命令やジャンプ命令を2バイトで実行でき、プログラムステップ数の節減に役立ちます。

固定ベクタ領域の FFFFDCH 番地から FFFFFFH 番地は、リセットおよび NMI などの固定割り込みベクタテーブルです。ここに割り込みルーチンの先頭アドレスを格納します。なお、タイマ割り込みなどの割り込みベクタテーブルは、内部レジスタ(INTB)により任意の番地に設定することができます。詳しくは第4章　割り込みの節をご参照ください。

固定ベクタ領域のメモリ配置図

スペシャルページベクタテーブルと固定ベクタ領域のメモリ配置を図 2.1.6 に示します。

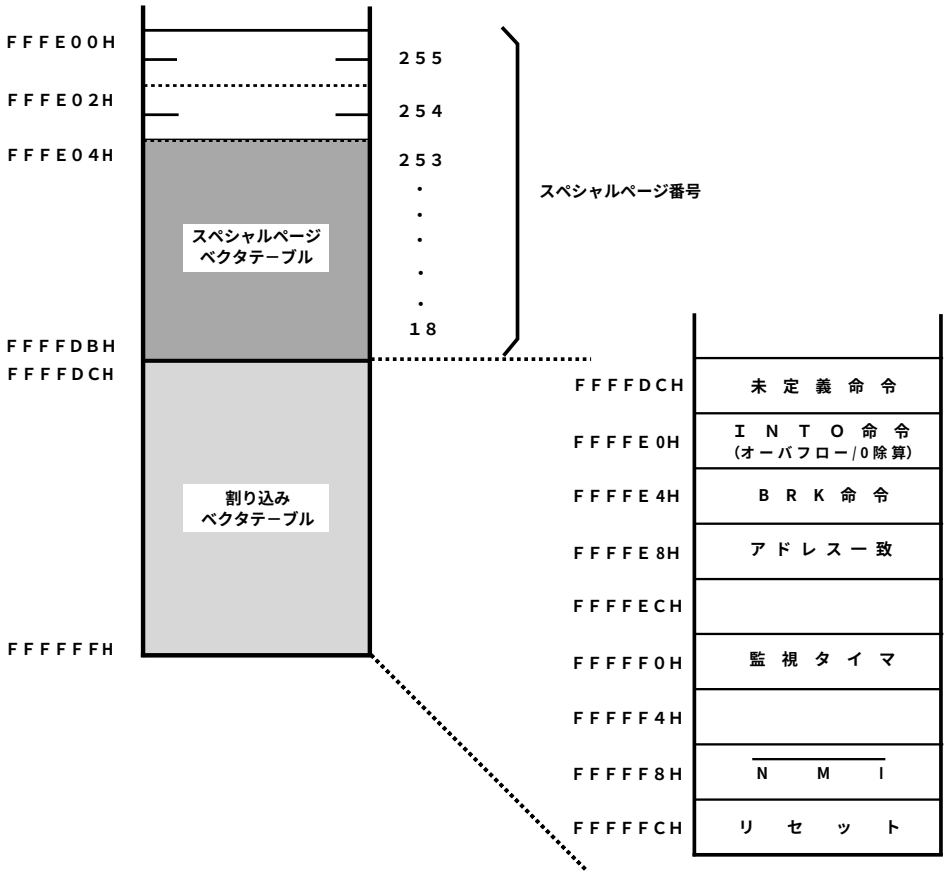


図 2.1.8　固定ベクタ領域のメモリ配置

2.2 レジスタセット

M16C/80 シリーズ CPU コアの通常レジスタ、高速割り込みレジスタおよび DMAC 関連レジスタについて説明します。

レジスタ構成

M16C/80 シリーズ CPU コアのレジスタ構成を図 2.2.1 に示します。レジスタ R0、R1、R2、R3、A0、A1、FB、SB の 8 本は 2 セット用意されています。各レジスタの機能を以下に示します。

通常レジスタ

(1)データレジスタ(R0/R1/R2/R3)

データレジスタ(R0/R1/R2/R3)は 16 ビットで構成されており、主に転送や算術、論理演算に使用します。

R0 と R1 は、上位(R0H/R1H)と下位(R0L/R1L)を別々に 8 ビットのデータレジスタとして使用することもできます。また、一部の命令では R2 と R0、R3 と R1 を組み合わせて 32 ビットのデータレジスタ(R2R0/R3R1)としても使用できます。

(2)アドレスレジスタ(A0/A1)

アドレスレジスタ(A0/A1)は24ビットで構成されており、データレジスタと同等の機能を持ちます。また、アドレスレジスタ間接アドレッシングおよび、アドレスレジスタ相対アドレッシングに使用します。

(3)フレームベースレジスタ(FB)

フレームベースレジスタ(FB)は 24 ビットで構成されており、FB 相対アドレッシングに使用します。

(4)スタティックベースレジスタ(SB)

スタティックベースレジスタ(SB)は 24 ビットで構成されており、SB 相対アドレッシングに使用します。

(5)プログラムカウンタ(PC)

プログラムカウンタ(PC)は 24 ビットで構成されており、実行する命令の番地を示します。

(6)割り込みテーブルレジスタ(INTB)

割り込みテーブルレジスタ(INTB)は 24 ビットで構成されており、割り込みベクタテーブルの先頭番地を示します。

(7)スタックポインタ(USP/ISP)

スタックポインタは、ユーザスタックポインタ(USP)と割り込みスタックポインタ(ISP)の2種類があり、共に 24 ビットで構成されています。

使用するスタックポインタ(USP/ISP)はスタックポインタ指定フラグ(Uフラグ)によって切り換えられます。U フラグは、フラグレジスタ(FLG)のビット 7 です。

USP、ISP には偶数を設定してください。偶数を設定した方が実行効率が良くなります。

(8)フラグレジスタ(FLG)

フラグレジスタ(FLG)は 11 ビットで構成されており、1 ビット単位でフラグとして使用します。

高速割り込みレジスタ

(9)フラグ待避レジスタ(SVF)

フラグ待避レジスタ(SVF)は16ビットで構成されており、高速割り込み発生時フラグレジスタを待避させるのに使用します。

(10)PC 待避レジスタ(SVP)

PC待避レジスタ(SVP)は24ビットで構成されており、高速割り込み発生時プログラムカウンタを待避させるのに使用します。

(11)ベクタレジスタ(VCT)

ベクタレジスタ(VCT)は 24 ビットで構成されており、高速割り込み発生時飛び先番地を示します。

DMAC 関連レジスタ

(12)DMA モードレジスタ(DMD0/DMD1)

DMA モードレジスタ(DMD0/DMD1)は 8 ビットで構成されており、DMA の転送モードなどを設定するレジスタです。

(13)DMA 転送カウントレジスタ(DCT0/DCT1)

DMA転送カウントレジスタ(DCT0/DCT1)は16ビットで構成されており、DMAの転送回数を設定するレジスタです。

(14)DMA 転送カウントリロードレジスタ(DRC0/DRC1)

DMA 転送カウントリロードレジスタ(DRC0/DRC1)は 16 ビットで構成されており、DMA 転送カウントレジスタのリロードレジスタです。

(15)DMA メモリアドレスレジスタ(DMA0/DMA1)

DMA メモリアドレスレジスタ(DMA0/DMA1)は 24 ビットで構成されており、DMA の転送元あるいは転送先のメモリアドレスを設定するレジスタです。

(16)DMA SFR アドレスレジスタ(DSA0/DSA1)

DMA SFR アドレスレジスタ(DSA0/DSA1)は 24 ビットで構成されており、DMA 転送の転送元あるいは転送先の固定アドレスを設定するレジスタです。

(17)DMA メモリアドレスリロードレジスタ(DRA0/DRA1)

DMA メモリアドレスリロードレジスタ(DRA0/DRA1)は 24 ビットで構成されており、DMA メモリアドレスレジスタへのリロードレジスタです。

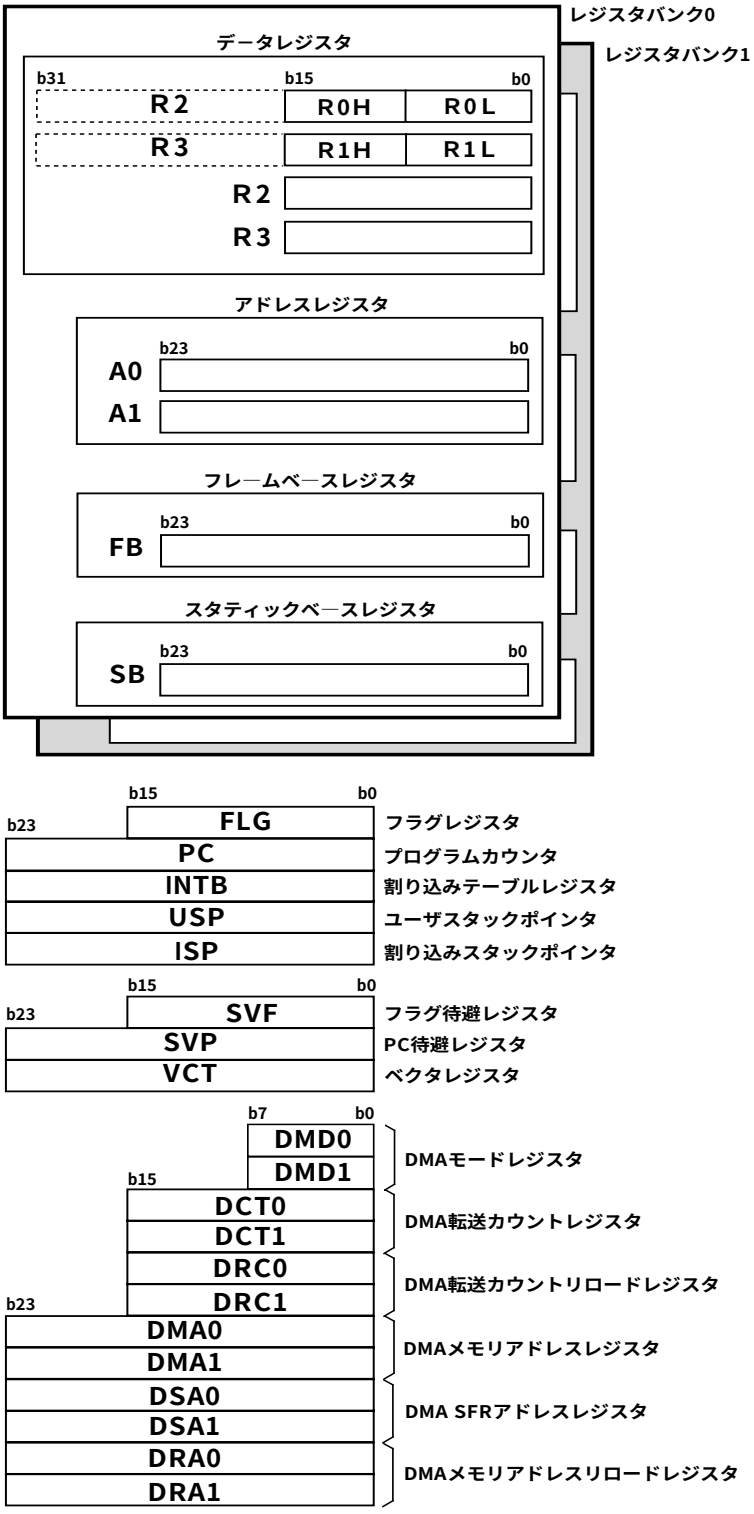

レジスタバンク0
レジスタバンク1
データレジスタ
b31
b15
b0
R2
R0H
R0L
R3
R1H
R1L
R2
R3
アドレスレジスタ
b23
b0
A0
A1
フレームベースレジスタ
FB
スタティックベースレジスタ
SB
b15
b0
b23
FLG
フラグレジスタ
PC
プログラムカウンタ
INTB
割り込みテーブルレジスタ
USP
ユーザスタックポインタ
ISP
割り込みスタックポインタ
SVF
フラグ待避レジスタ
SVP
PC待避レジスタ
VCT
ベクタレジスタ
b7
DMD0
DMD1
DMAモードレジスタ
DCT0
DCT1
DMA転送カウントレジスタ
DRC0
DRC1
DMA転送カウントリロードレジスタ
DMA0
DMA1
DMAメモリアドレスレジスタ
DSA0
DSA1
DMA SFRアドレスレジスタ
DRA0
DRA1
DMAメモリアドレスリロードレジスタ

図 2.2.1　レジスタ構成

フラグレジスタ(FLG)

フラグレジスタ(FLG)のビット構成を図 2.2.2 に示します。各フラグの機能を以下に示します。

●ビット 0:キャリーフラグ(C フラグ)
算術論理ユニットで発生したキャリー、ボロー、シフトアウトしたビットなどを保持します。

●ビット 1:デバッグフラグ(D フラグ)
シングルステップ割り込みを許可するフラグです。
このフラグが“1”のとき、命令実行後シングルステップ割り込みが発生します。割り込みを受け付けるとこのフラグは“0”になります。

●ビット 2:ゼロフラグ(Z フラグ)
演算の結果が 0 のとき“1”になり、それ以外のとき“0”になります。

●ビット 3:サインフラグ(S フラグ)
演算の結果が負のとき“1”になり、それ以外のとき“0”になります。

●ビット 4:レジスタバンク指定フラグ(B フラグ)
レジスタバンクの選択を行います。このフラグが“0”のときレジスタバンク 0 が指定され、“1”のときレジスタバンク 1 が指定されます。

●ビット 5:オーバフローフラグ(O フラグ)
演算の結果がオーバフローしたとき“1”になります。

●ビット 6:割り込み許可フラグ(I フラグ)
マスカブル割り込みを許可するフラグです。
このフラグが“1”のとき割り込みは許可され、“0”のとき割り込みは禁止されます。割り込みを受け付けるとこのフラグは“0”になります。

●ビット 7:スタックポインタ指定フラグ(U フラグ)
このフラグが“1”のときユーザスタックポインタ(USP)が指定され、“0”のとき割り込みスタックポインタ（ISP）が指定されます。
ハードウェア割り込みを受け付けたとき、またはソフトウェア割り込み番号 0～31 の INT 命令を実行したとき、このフラグは“0”になります。

●ビット 8～ビット 11:予約領域

●ビット 12～ビット 14:プロセッサ割り込み優先レベル(IPL)
プロセッサ割り込み優先レベル(IPL)は 3 ビットで構成されており、レベル 0～レベル 7 までの 8 段階のプロセッサ割り込み優先レベルを指定します。
要求があった割り込みの優先レベルが、プロセッサ割り込み優先レベル（IPL）より大きい場合、その割り込みは許可されます。

●ビット 15:予約領域

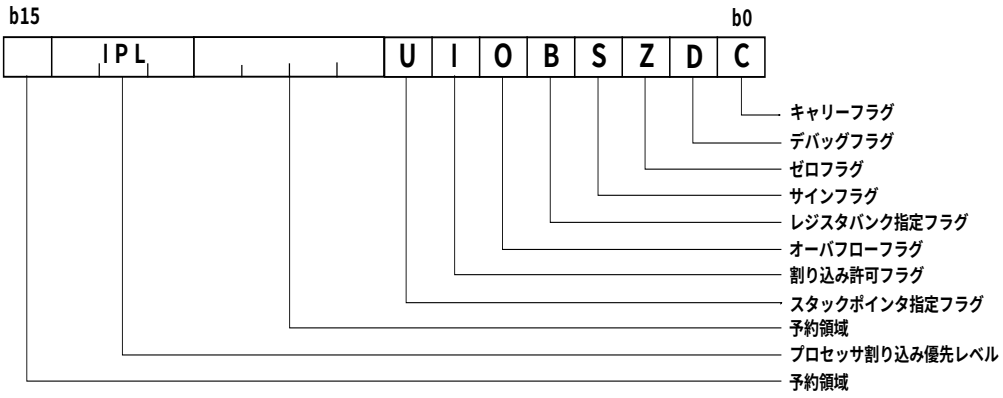


図 2.2.2　フラグレジスタ(FLG)のビット構成

リセット解除後のレジスタの状態

リセット解除後の各レジスタの表 2.2.1 に示します。（注）

表 2.2.1　リセット解除後のレジスタの状態

レジスタ名	リセット解除後の状態
データレジスタ(R0/R1/R2/R3)	0000H
アドレスレジスタ(A0/A1)	000000H
スタティックベースレジスタ(SB)	000000H
フレームベースレジスタ(FB)	000000H
割り込みテーブルレジスタ(INTB)	000000H
ユーザスタックポインタ(USP)	000000H
割り込みスタックポインタ(ISP)	000000H
フラグレジスタ(FLG)	0000H
DMAモードレジスタ(DMD0/DMD1)	00H
DMA転送カウントレジスタ(DCT0/DCT1)	不定
DMA転送カウントリロードレジスタ(DRC0/DRC1)	不定
DMAメモリアドレスレジスタ(DMA0/DMA1)	不定
DMA SFRアドレスレジスタ(DSA0/DSA1)	不定
DMAメモリアドレスリロードレジスタ(DRA0/DRA1)	不定

（注）リセット解除後の SFR 領域の制御レジスタの状態については M16C/80 グループのデータシート、およびユーザーズマニュアルをご参照ください。

2.3 データタイプ

M16C/80 シリーズで取り扱うデータタイプは整数、10 進(BCD)、ストリング、ビットの 4 種類です。ここではデータタイプについて説明します。

整数

整数は符号付きと符号なしがあります。符号付き整数の負の値は 2 の補数で表現します。

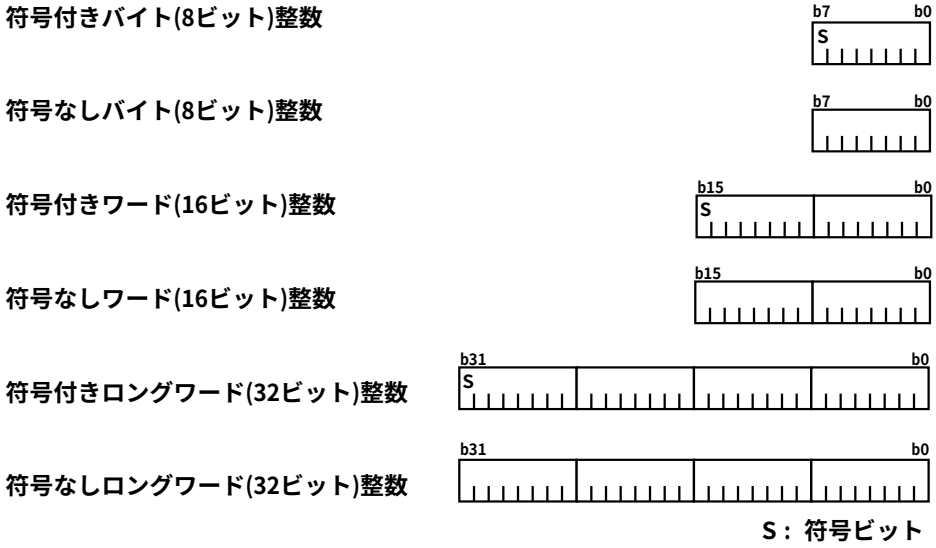


図 2.3.1 整数データ

10 進(BCD)

BCD コードはパック形式で取り扱います。
10 進演算命令 DADC、DADD、DSBB、DSUB の 4 種類の命令で使用できます。

図 2.3.2 10 進データ

ストリング

1バイトまたは1ワード(16ビット)のデータを任意の数だけ連続して並べたデータのブロックをストリングといいます。

ストリング命令 SMOVB、SMOVF、SSTR、SCMPU、SMOVU、SIN、SOUT の 7 種類の命令で使用できます。

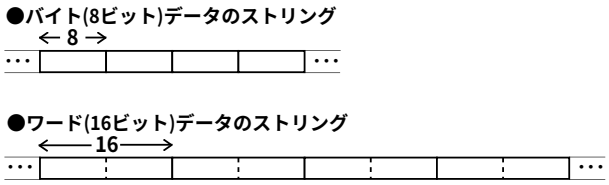


図 2.3.3　ストリングデータ

ビット

ビットはビット命令 BCLR、BSET、BTST、BNTST などの 14 種類の命令で使用できます。各レジスタのビットは、レジスタ名と 0～7 のビット番号で指定します。メモリのビットはアドレッシングモードにより、指定方法、指定範囲が異なります。　使い方については、「2.5.4　ビット命令アドレッシング」をご参照ください。

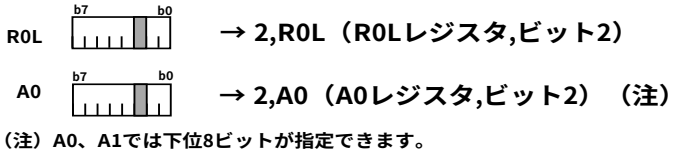


図 2.3.4　レジスタのビット指定

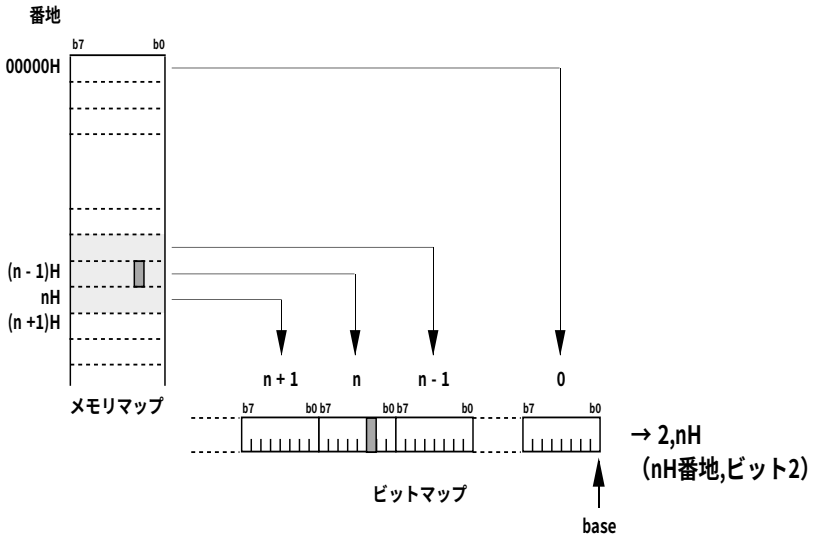


図 2.3.5　メモリのビット指定

2.4 データ配置

M16C/80シリーズでは、ニブル(4ビット)やバイト(8ビット)のデータも効率良く取り扱うことができます。ここでは取り扱うことのできるデータの配置について説明します。

レジスタのデータ配置

レジスタのデータサイズとビット番号の関係を図 2.4.1 に示します。

図のように、最下位ビット(LSB)のビット番号は 0 になります。最上位ビット(MSB)のビット番号は取り扱うデータサイズによって異なります。

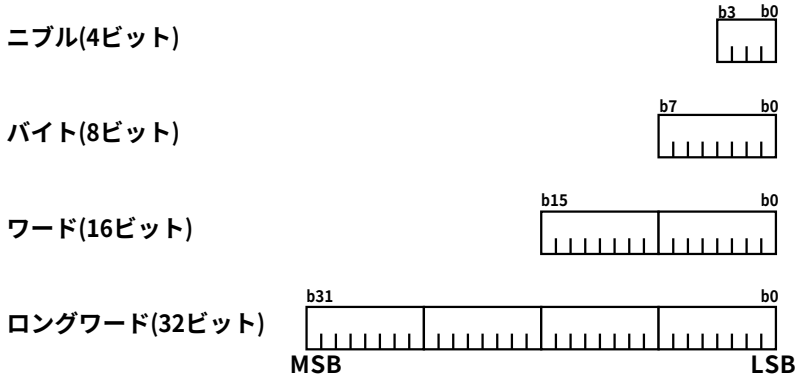


図 2.4.1 レジスタのデータ配置

メモリ上のデータ配置

M16C/80 シリーズメモリ上のデータ配置を図 2.4.2 に示します。

メモリでは図のように 8 ビット単位でデータを配置します。ワード(16 ビット)は下位バイト、上位バイトに分けて、下位バイト:DATA(L)を小さい番地に配置します。アドレス(24 ビット)、ロングワード(32 ビット)も同様に下位バイト:DATA(LL)からメモリ上に配置します。

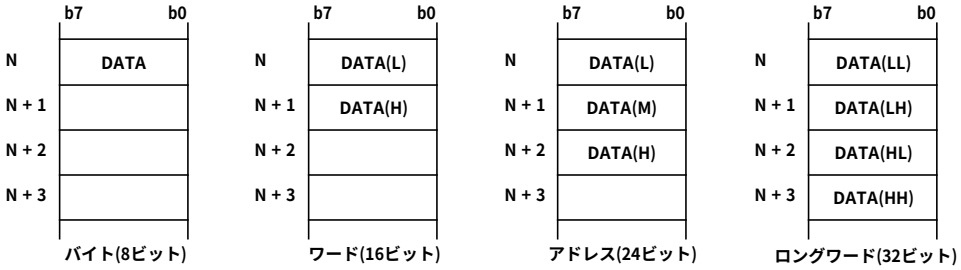


図 2.4.2 メモリ上のデータ配置

2.5 アドレッシングモード

M16C/80 シリーズのアドレッシングについて説明します。
アドレッシングモードは以下に示す 3 つのタイプに分けられます。
(1)一般命令アドレッシング
000000H 番地から FFFFFFH 番地までのアドレス空間全体をアクセスします。
(2)間接命令アドレッシング
000000H 番地から FFFFFFH 番地までのアドレス空間全体をアクセスします。
(3)特定命令アドレッシング
000000H 番地から FFFFFFH 番地までのアドレス空間全体、および専用レジスタをアクセスします。
(4)ビット命令アドレッシング
000000H 番地から FFFFFFH 番地までのアドレス空間全体をビット単位でアクセスします。

アドレッシングモード一覧

全アドレッシングを表 2.5.1 と表 2.5.2 にまとめます。

表 2.5.1　M16C/80 シリーズのアドレッシングモード 1

項目	内　容	
アドレッシングモード	一般命令	間接命令
即値	imm:8/16/32ビット	×
レジスタ直接	データレジスタ, アドレスレジスタのみ	×
専用レジスタ直接	×	×
絶対	abs:16ビット(0～FFFFH) 24ビット(0～FFFFFFH)	×
絶対間接	×	[abs : 16/24ビット]　(0～FFFFFFH)
アドレスレジスタ間接	[A0] or [A1]　dspなし	×
アドレスレジスタ二段間接	×	[[A0]] or[[A1]] dspなし　(0～FFFFFFH)
アドレスレジスタ相対	[A0] or [A1]　dsp : 8/16/24ビット	×
アドレスレジスタ相対間接	×	[dsp:8/16/24[A0]] or [dsp:8/16/24[A1]]　(0～FFFFFFH)
SB相対とFB相対	dsp:8[SB] dsp:16[SB] dsp : 8/16ビット (0～255/0～65534) dsp:8[FB] dsp:16[FB] dsp:8/16ビット (-128～+127/-32768～+32767)	× ×
SB相対間接と FB相対間接	× ×	[dsp:8/16[SB]]　(0～FFFFFFH) [dsp:8/16[FB]]　(0～FFFFFFH)
スタックポインタ相対	dsp:8[SP] dsp : 8ビット(-128～+127)　※MOV命令のみ	×
プログラムカウンタ相対	×	×
ＦＬＧ直接	×	×

表 2.5.2　M16C/80 シリーズのアドレッシングモード 2

項目	内　容	
アドレッシングモード	特定命令	ビット命令
即値	×	×
レジスタ直接	×	R0L/R0H/R1L/R1H/A0/A1のみ
専用レジスタ直接	INTB、ISP、SP、DMD0など専用レジスタのみ	×
絶対	×	base:19/27ビット　(0〜FFFFH / 0〜0FFFFFFH)
絶対間接	×	×
アドレスレジスタ間接	×	bit,[A0] or bit,[A1](0H〜0FFFFFFH) bit：0〜7
アドレスレジスタ二段間接	×	×
アドレスレジスタ相対	×	bit,base[A0] or bit,base[A1] base：11/19/27
アドレスレジスタ相対間接	×	×
SB相対とFB相対	× ×	bit,base:11[SB]　(0H〜FFH) bit,base:19[SB]　(0H〜FFFFH) bit,base:11[FB]　(-128〜+127) bit,base:19[FB]　(-32768〜+32767)
SB相対間接と FB相対間接	× ×	× ×
スタックポインタ相対	×	×
プログラムカウンタ相対	label : .S : +0 〜 +7（JMP命令のみ） .B : -128 〜 +127（JMP, JSR命令のみ） .W : -32768 〜 +32767 (JMP, JSR命令のみ) .lengthなし: -127〜+128（Jcnd命令のみ）	×
ＦＬＧ直接	×	U, I, O, B, S, Z, D, Cフラグ　※FCLR, FSET命令のみ

2.5.1　一般命令アドレッシング

この項では一般命令アドレッシングの各アドレッシングについて説明します。

即値

#IMM で示した即値が演算対象になります。即値の前に # を付けてください。
(記号)#IMM、#IMM8、#IMM16、#IMM32
(例) #123(10 進数)
#7DH(16 進数)
#01111011B(2 進数)

絶対

abs16/24 で指定した値が演算対象の実効アドレスになります。実効アドレスの範囲は abs16 では 000000H ～ 000FFFFH 番地、abs24 では 000000H ～ FFFFFFH 番地です。
(記号)abs16、abs24
(例) MOV.B　#12H,DATA

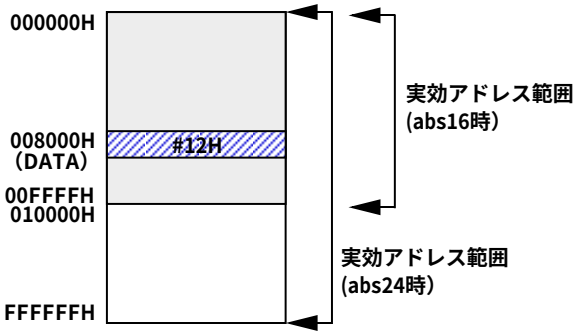


図 2.5.1　絶対アドレッシング

レジスタ直接

指定したレジスタが演算対象になります。
ただし、データレジスタ、アドレスレジスタのみ指定できます。
(記号) 8 ビット　R0L、R0H、R1L、R1H
16 ビット　R0、R1、R2、R3、A0、A1
32 ビット　R2R0、R3R1

アドレスレジスタ間接

アドレスレジスタの値が実効アドレスとなります。実効アドレスの範囲は 000000H ～ FFFFFFH 番地です。

(記号)[A0]、[A1]
(例)MOV.B　#12H,[A0]

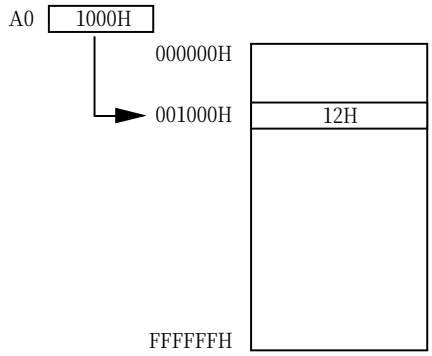


図 2.5.2　アドレスレジスタ間接アドレッシング

アドレスレジスタ相対

アドレスレジスタとディスプレースメント(dsp)(注)を符号なしで加算した結果が演算対象の実効アドレスとなります。実効アドレスの範囲は 000000H ～ FFFFFFH 番地です。加算結果が FFFFFFH を越えた場合、25 ビット以上は無視され 000000H 番地側に戻ります。

(記号)dsp:8[A0]、dsp:16[A0]、dsp:24[A0]、dsp:8[A1]、dsp:16[A1]、dsp:24[A1]

(1)dsp を変位として扱った場合

(例)MOV.B　#34H,5[A0]

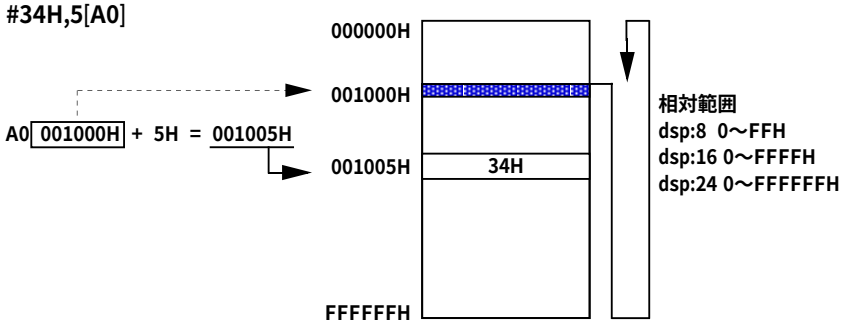


図 2.5.3　アドレスレジスタ相対アドレッシング 1

(2)アドレスレジスタ(A0)を変位として扱った場合

(例)MOV.B　#56H,1234H[A0]

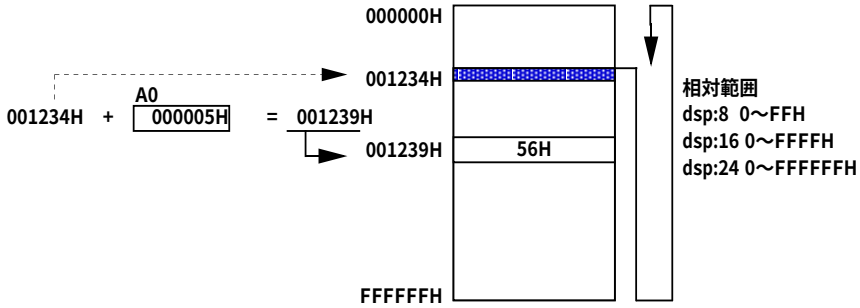


図 2.5.4　アドレスレジスタ相対アドレッシング 2

(3)加算結果が FFFFFFH を越えた場合

（例）MOV. B　#56H,123456H[A0]

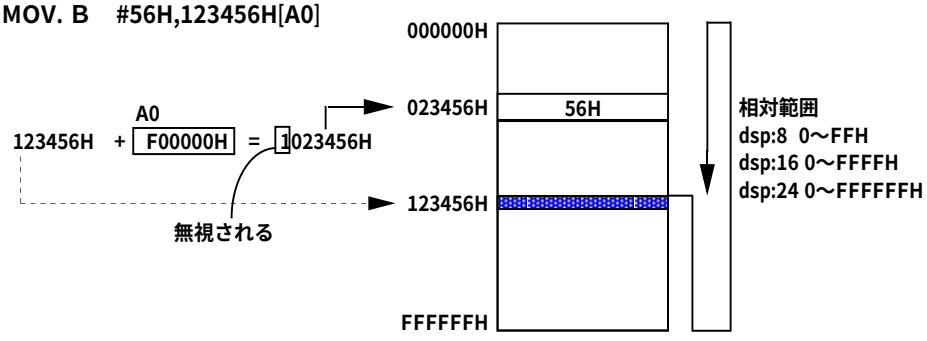


図 2.5.5　アドレスレジスタ相対アドレッシング 3

（注）ディスプレースメント(dsp)とは基準アドレスからの変位です。本書では 8 ビットの dsp を dsp:8、16 ビットの dsp を dsp:16、24 ビットの dsp を dsp:24 と表現します。

SB 相対

SBレジスタの内容で示したアドレスにdspで示した値を符号なし加算した結果が演算対象の実効アドレスとなります。実効アドレスの範囲は 000000H ～ FFFFFFH 番地です。加算結果が FFFFFFH を越えた場合、25 ビット以上は無視され、000000H 番地側に戻ります。

(記号)dsp:8[SB]、dsp:16[SB]
(例)MOV.B　#12H,5[SB]

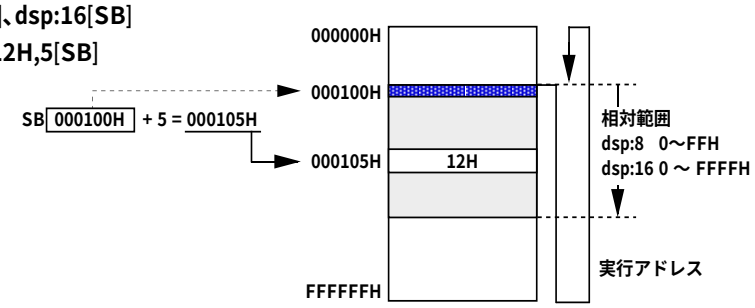


図 2.5.6　SB 相対アドレッシング

FB 相対

FBレジスタの内容で示したアドレスにdspで示した値を符号付き加算した結果が演算対象の実効アドレスとなります。実効アドレスの範囲は 000000H ～ FFFFFFH 番地です。加算結果が 000000H ～ FFFFFFH を越えた場合、25 ビット以上は無視され、000000H 番地側または FFFFFFH 番地側に戻ります。

(記号)dsp:8[FB]、dsp:16[FB]
(1)dsp の値が正のとき
(例)MOV.B　#12H,5[FB]

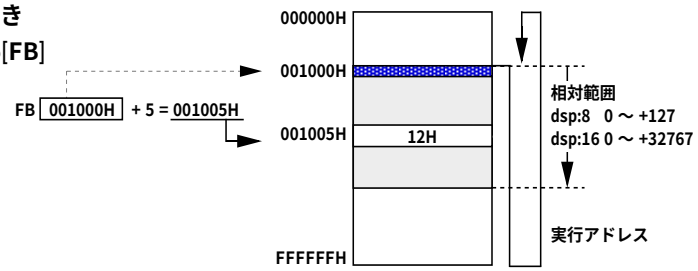


図 2.5.7　FB 相対アドレッシング 1

(2)dsp の値が負のとき
(例)MOV.B　#12H,-5[FB]

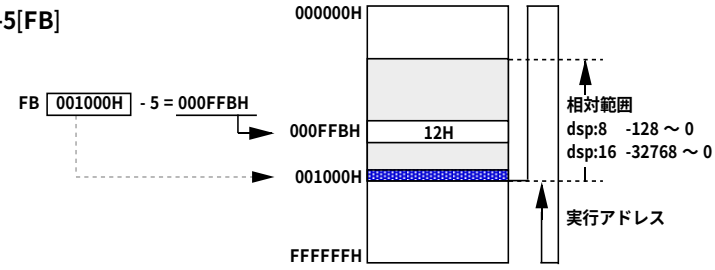


図 2.5.8　FB 相対アドレッシング 2

コラム　SB 相対と FB 相対の違い

SB 相対アドレッシングは SB レジスタの内容で示したアドレスに dsp で示した値を符号なし加算した結果を演算対象の実効アドレスとするアドレッシングです。相対範囲は dsp:8[SB]の場合は 0 ～ +255(FFH)で、dsp:16[SB]の場合は 0 ～ +65535(FFFFH)です。

FB 相対アドレッシングは FB レジスタの内容で示したアドレスに dsp を加算した値、または dsp を減算した結果を演算対象の実効アドレスとするアドレッシングです。相対範囲は dsp:8[FB]の場合は -128 ～ +127(80H ～ 7FH)で、dsp:16[FB]の場合は -32768 ～ +32767(8000H ～ 7FFFH)です。FB 相対は負方向にアクセスすることができます。dsp は 8 ビットと 16 ビットが使えます。

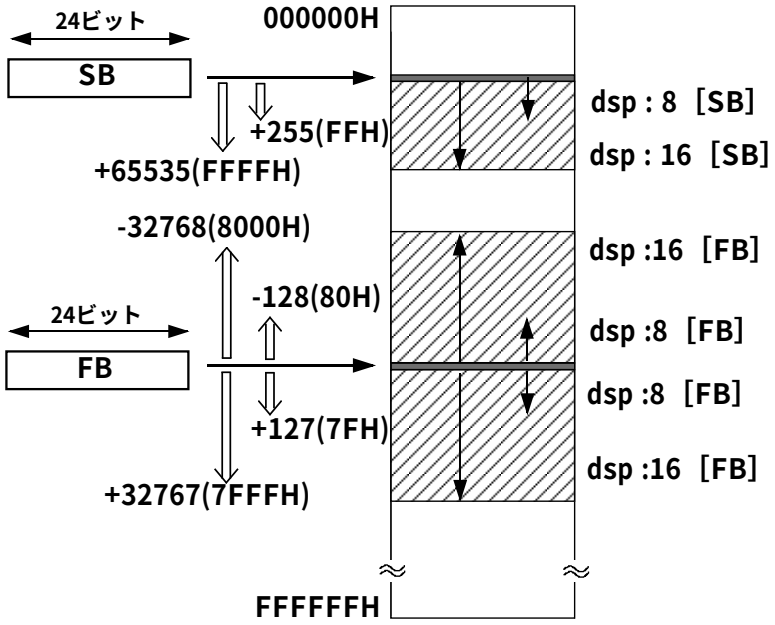


図 2.5.9　SB 相対アドレッシングと FB 相対アドレッシング

コラム　SB 相対の応用例

SB相対アドレッシングは図2.5.10のようにサブルーチンの固有データテーブルへ応用できます。各サブルーチンを動作させるために必要なデータはサブルーチンの呼び出し時に切り替えますが、SB相対アドレッシングを使うと SB レジスタを書き替えるだけで簡単に切り替えることができます。

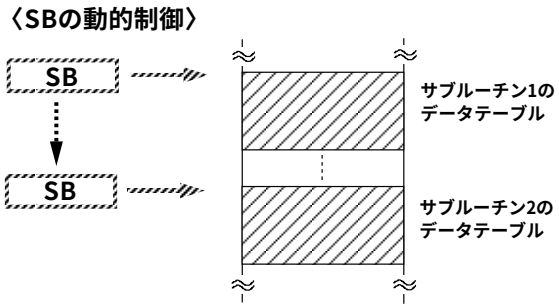


図 2.5.10　SB 相対の応用例

コラム　FB 相対の応用例

FB相対アドレッシングは図2.5.11のように関数を呼び出したときのスタックフレームへ応用できます。スタックフレームでは、マイナス方向のアドレスにローカル変数領域が配置されるので、ベースからプラス方向もマイナス方向もアクセスできる FB 相対アドレッシングが必要になります。

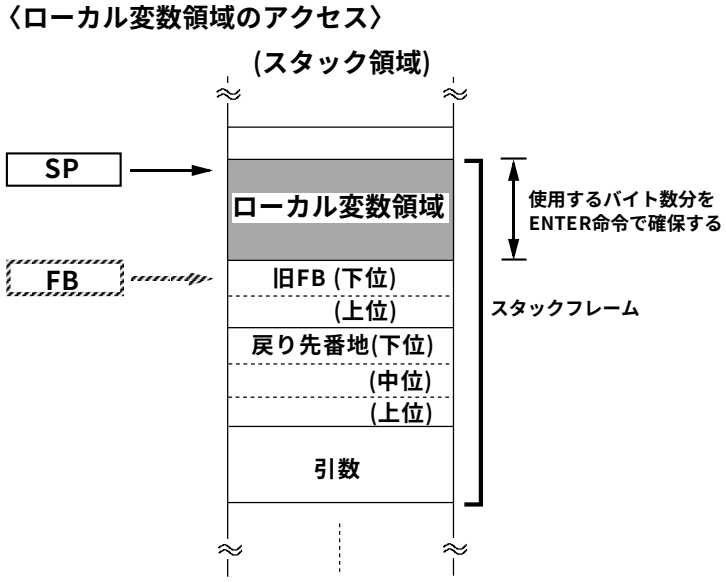


図 2.5.11　FB 相対の応用例

スタックポインタ相対(SP 相対)

SP 相対アドレッシングは、SB レジスタの内容で示したアドレスに dsp で示した値を符号付き加算した結果が演算対象の実効アドレスとなります。SP 相対アドレッシングは MOV 命令でのみ使用できます。実効アドレスの範囲は 000000H～FFFFFFH 番地です。加算結果が 000000H～FFFFFFH を越えた場合、25 ビット以上は無視され、000000H 番地側または FFFFFFH 番地側に戻ります。

(記号)dsp:8[SP]
(1)dsp の値が正のとき
(例)MOV.B　R0L,5[SP]

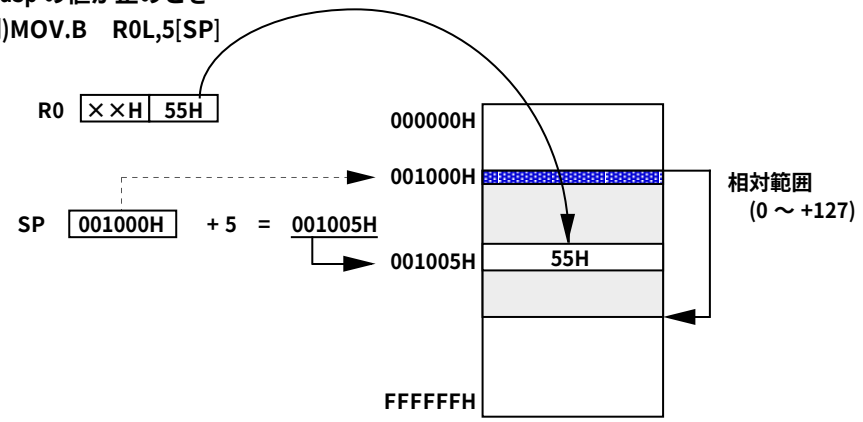


図 2.5.12　SP 相対アドレッシング 1

(2)dsp の値が負のとき
(例)MOV.B　R0L,-5[SP]

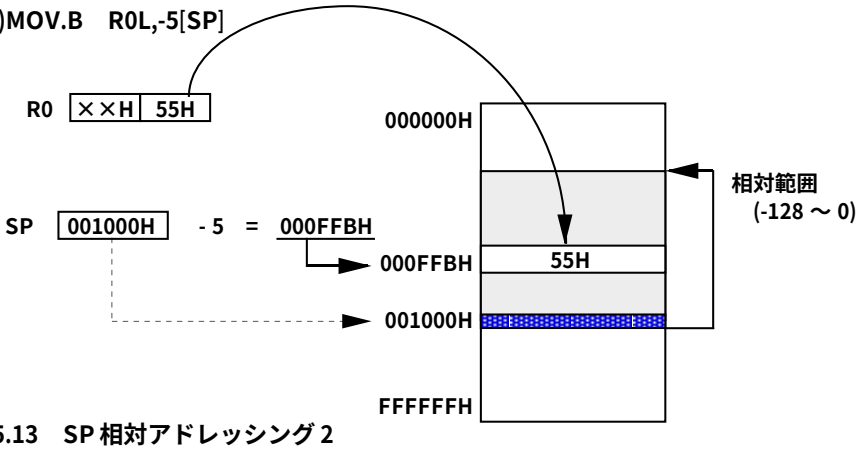


図 2.5.13　SP 相対アドレッシング 2

コラム　相対アドレッシングの相対範囲

相対アドレッシングの相対範囲を表 2.5.2 にまとめます。

表 2.5.3　相対アドレッシングの相対範囲

アドレッシングモード	記述形式	相対範囲
アドレスレジスタ相対	dsp:8[An] dsp:16[An] dsp:24[An]	0～255(FFH) 0～65535(FFFFH) 0～16777215(FFFFFFH)
SB相対とFB相対	dsp:8[SB] dsp:16[SB] dsp:8[FB] dsp:16[FB]	0～255(0FFH) 0～65535(0FFFFH) -128(80H)～+127(7FH) -32768(8000H)～+32767(7FFFH)
スタックポインタ相対	dsp:8[SP]	-128(80H)～+127(7FH)

2.5.2 間接命令アドレッシング

間接命令アドレッシングは 000000H ～ FFFFFFH までのアドレス空間をアクセスできます。この項では、間接命令アドレッシングの各アドレッシングについて説明します。

絶対間接

絶対アドレッシングで示したメモリからの連続した 4 バイトの値が演算対象の実効アドレスとなります。実効アドレスの範囲は 000000H ～ FFFFFFH 番地です。

(記号)[abs16]、[abs24]
(例)MOV.B　[001000H],R0L

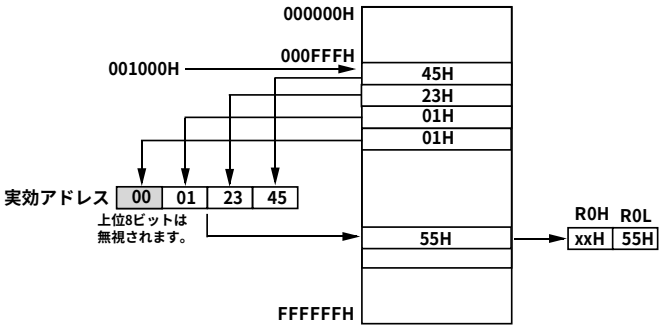


図 2.5.14　絶対間接アドレッシング

アドレスレジスタ二段間接

アドレスレジスタ間接アドレッシングで示したメモリからの連続した 4 バイトの値が演算対象の実効アドレスとなります。実効アドレスの範囲は 000000H ～ FFFFFFH 番地です。

(記号)[[A0]]、[[A1]]
(例)MOV.B　[[A0]],R0L

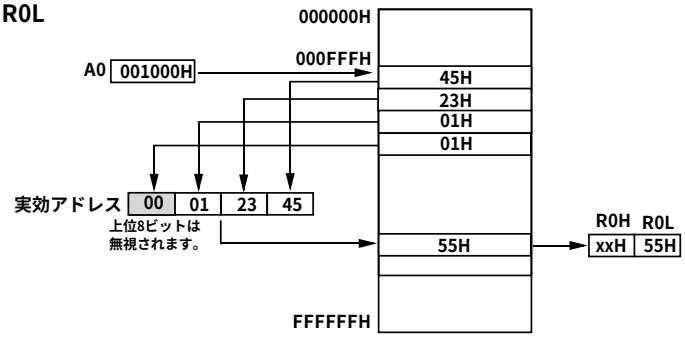


図 2.5.15　アドレスレジスタ二段間接アドレッシング

アドレスレジスタ相対間接

アドレスレジスタ相対アドレッシングで示したメモリからの連続した4バイトの値が実効アドレスとなります。実効アドレスの範囲は 000000H ～ FFFFFFH 番地です。

(記号)[dsp:8[A0]]、[dsp:8[A1]]、[dsp:16[A0]]、[dsp:16[A1]]、[dsp:24[A0]]、[dsp:24[A1]]
(例)MOV.B　[5[A0]],R0L

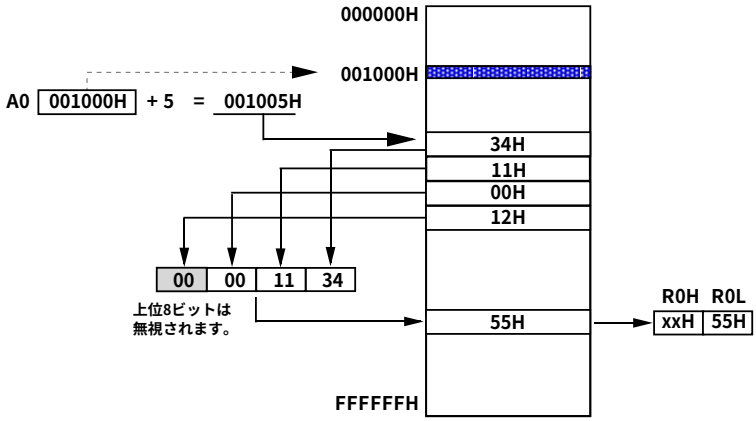


図 2.5.16　アドレスレジスタ相対間接アドレッシング

SB 相対間接

SB 相対アドレッシングで示したメモリからの連続した 4 バイトの値が演算対象の実効アドレスとなります。実効アドレスの範囲は 000000H ～ FFFFFFH 番地です。

(記号)[dsp:8[SB]]、[dsp:16[SB]]
(例)MOV.B　[2[SB]],R0L

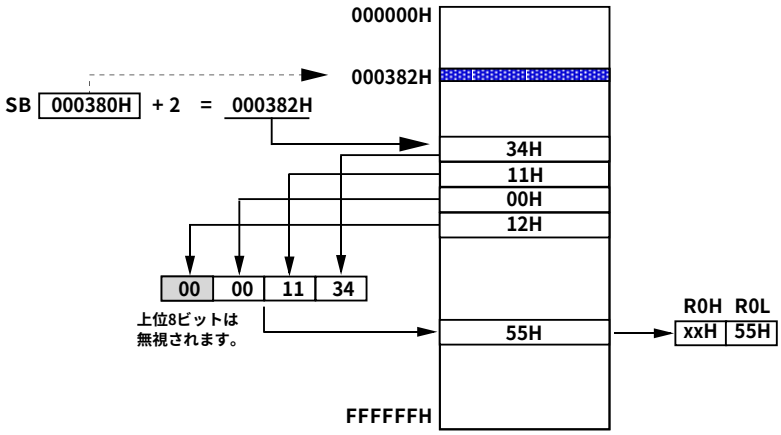


図 2.5.17　SB 相対間接アドレッシング

FB 相対間接

FB相対アドレッシングで示したメモリからの連続した4バイトの値が実効アドレスとなります。実効アドレスの範囲は 000000H ～ FFFFFFH 番地です。

(記号)[dsp:8[FB]]、[dsp:16[FB]]
(例)MOV.B　[2[FB]],R0L

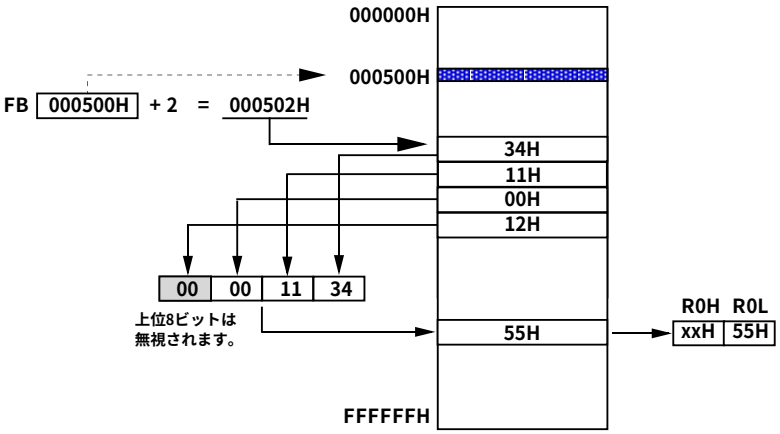


図 2.5.18　FB 相対間接アドレッシング 1

(例)MOV.B　[-25[FB]],R0L

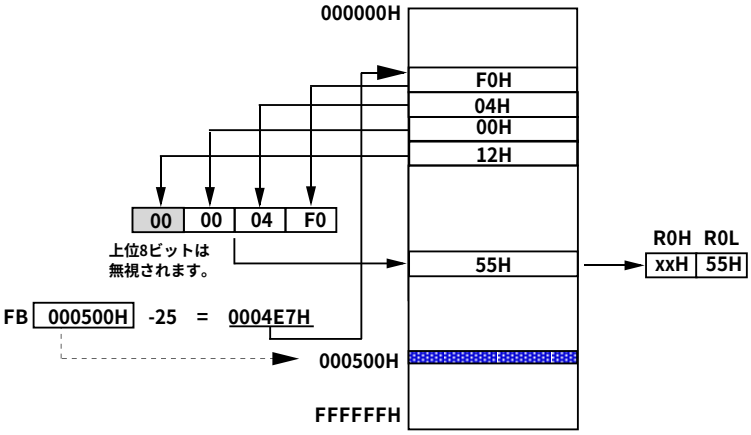


図 2.5.19　FB 相対間接アドレッシング 2

2.5.3　特定命令アドレッシング

特定命令アドレッシングは 000000H ～ FFFFFFH までのアドレス空間をアクセスできるアドレッシング、および専用レジスタをアクセスするアドレッシングです。この項では、特定命令アドレッシングの各アドレッシングについて説明します。

専用レジスタ直接

指定した専用レジスタが演算の対象となります。このアドレッシングは LDC、STC、POPC、PUSHC 命令で使用できます。

SP を指定した場合、U フラグで示すスタックポインタが対象となります。

(記号)INTB、ISP、SP、SB、FB、FLG、SVP、VCT、SVF、DMD0、DMD1、DCT0、
DCT1、DRC0、DRC1、DMA0、DMA1、DSA0、DSA1、DRA0、DRA1
(例)LDC　#001000H,ISP

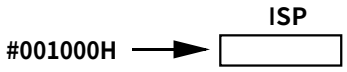


図 2.5.20　専用レジスタ直接アドレッシング

プログラムカウンタ相対

分岐距離指定子（.length）が（.S）の場合、基準の番地にディスプレースメント（dsp）で示した値を符号なしで加算した結果が実行アドレスになります。

このアドレッシングは JMP 命令で使用できます。

(1)分岐距離指定子(.length)が .S の場合
(記号)label　(PC+2 ≦ label ≦ PC+9)

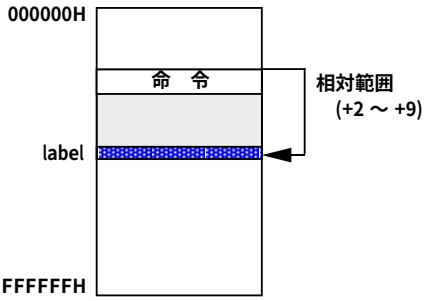


図 2.5.21　PC 相対アドレッシング 1

分岐距離指定子（.length）が（.B）または（.W）の場合、基準の番地にディスプレースメント（dsp）で示した値を符号付きで加算した結果が実行アドレスになります。
このアドレッシングは JMP、JSR 命令で使用できます。

(2)分岐距離指定子(.length)が .B の場合
(記号)label　(PC-128 ≦ label ≦ PC+127)

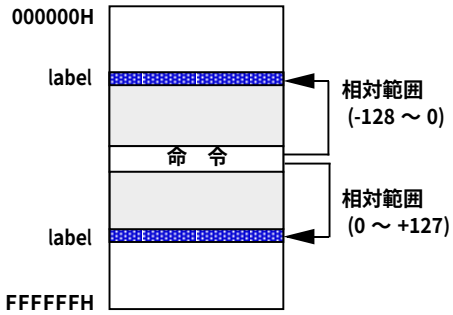


図 2.5.22　PC 相対アドレッシング 2

(3)分岐距離指定子(.length)が .W の場合
(記号)label　(PC-32768 ≦ label ≦ PC+32767)

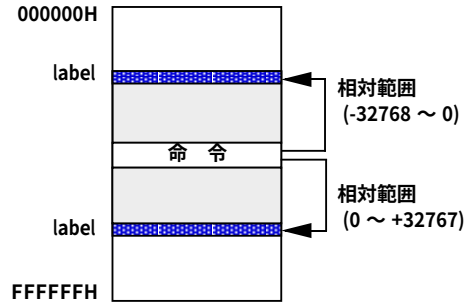


図 2.5.23　PC 相対アドレッシング 3

分岐距離指定子（.length）の記述を省略した場合はアセンブラが最適な指定子を選択します。また、加算結果が 000000H ～ FFFFFFH を越える場合、25 ビット以上は無視され、000000H 番地または FFFFFFH 番地側に戻ります。

2.5.4　ビット命令アドレッシング

000000H ～ FFFFFFH までのアドレス空間をビット単位でアクセスします。
このアドレッシングはビット命令に使用します。この項ではビット命令アドレッシングの各アドレッシングについて説明します。

絶対

base で示したアドレスのビット 0 から bit で示したビット数だけ離れたビットが演算対象となります。
bit,base19、bit,base27 の指定可能領域は 000000H ～ 00FFFFH 番地、000000H ～ FFFFFFH 番地です。

(記号)bit,base19、bit,base27

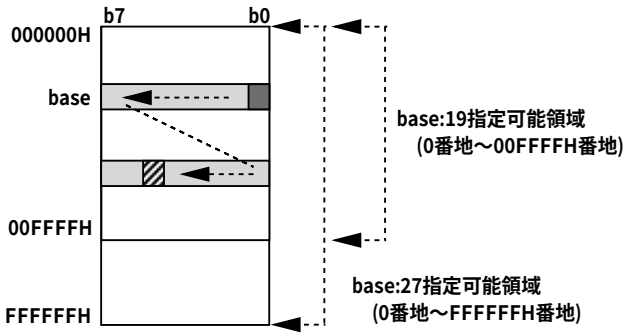


図 2.5.24　ビット命令絶対アドレッシング 1

(例 1)BCLR　18,base_addr
(例 2)BCLR　4,base_addr2

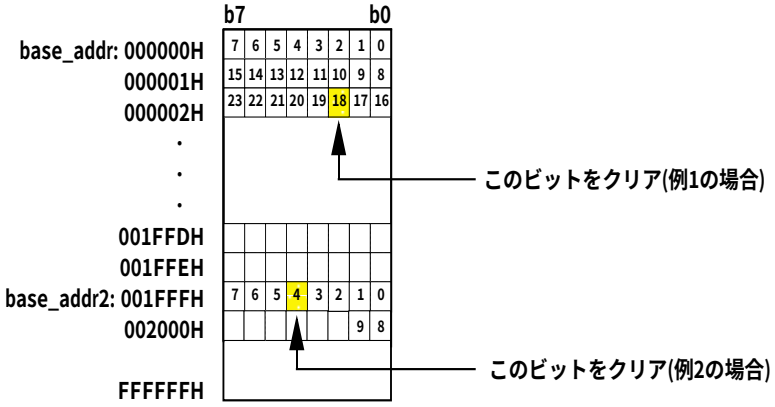


図 2.5.25　ビット命令絶対アドレッシング 2

レジスタ直接

指定したレジスタのビットが演算対象になります。ビット位置（bit）は 0 ～ 7 が指定できます。A0、A1 では下位 8 ビットが指定できます。

(記号)bit,R0L、bit,R0H、 bit,R1L、bit,R1H、bit,A0、bit,A1

(例)BCLR　6,R0L

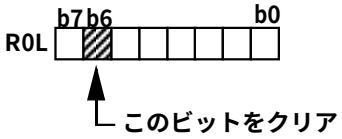


図 2.5.26　ビット命令レジスタ直接アドレッシング

FLG 直接

指定したフラグが演算対象になります。このアドレッシングは FCLR、FSET 命令で使用できます。ビット位置は FLG レジスタの下位 8 ビットのみ指定できます。

(記号)U、I、O、B、S、Z、D、C

(例)FSET　U

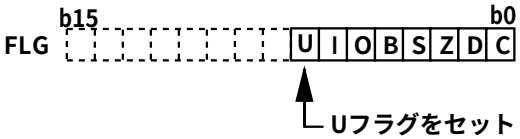


図 2.5.27　ビット命令 FLG 直接アドレッシング

アドレスレジスタ間接

アドレスレジスタ（A0/A1）の内容で示したアドレスのビット 0 からビット数だけ離れたビットが演算対象となります。

指定可能領域は 000000H ～ FFFFFFH 番地です。ビット位置は 0 ～ 7 が指定できます。

(記号)bit,[A0]、bit,[A1]

(例)BCLR　5,[A0]

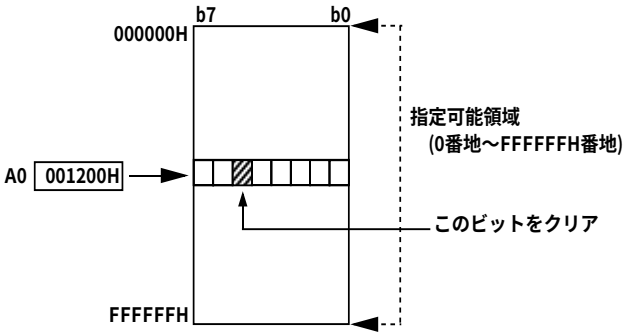


図 2.5.28　ビット命令アドレスレジスタ間接アドレッシング

アドレスレジスタ相対

アドレスレジスタ（A0/A1）の内容で示したアドレスに base で示した値を符号なしで加算したアドレスのビット 0 から、ビットで示したビット数だけ離れたビットが演算対象となります。実効アドレスの範囲は 000000H ～ FFFFFFH 番地です。対象となるビットのアドレスが FFFFFFH を越えた場合、25 ビット以上は無視され、000000H 番地側に戻ります。

bit,base:11、bit,base:19、bit,base:27で指定できるアドレス範囲はアドレスレジスタの値からそれぞれ256バイト、65536 バイト、16777216 バイトです。

(記号)bit,base:11[A0]、bit,base:11[A1]、bit,base:19[A0]、bit,base:19[A1]、bit,base:27[A0]、bit,base:27[A1]

(例)BCLR　5,26H[A0]

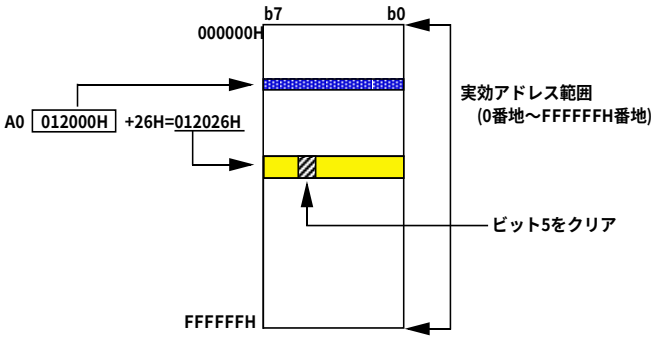


図 2.5.29　ビット命令アドレスレジスタ相対アドレッシング

SB 相対

基準はSBレジスタの値が示すアドレスです。SBレジスタの値にbaseを符号なしで加算します。その値が示すアドレスのビット0から、bitで示したビット数だけ離れたビットが演算対象となります。

bit,base:11、bit,base:19で指定できる指定可能領域はSBレジスタが示すアドレスからそれぞれ256バイト、65536バイトです。ただし、対象となるビットのアドレスがFFFFFFHを越えた場合、25ビット以上は無視され000000H番地側に戻ります。

(記号)bit,base:11[SB]、bit,base:19[SB]

注)bit,base:11[SB]: 最大 256 バイトの領域の 1 ビットを指定可能です。
bit,base:19[SB : 最大 64K バイトの領域の 1 ビットを指定可能です。

（例）BCLR 13,8[SB]

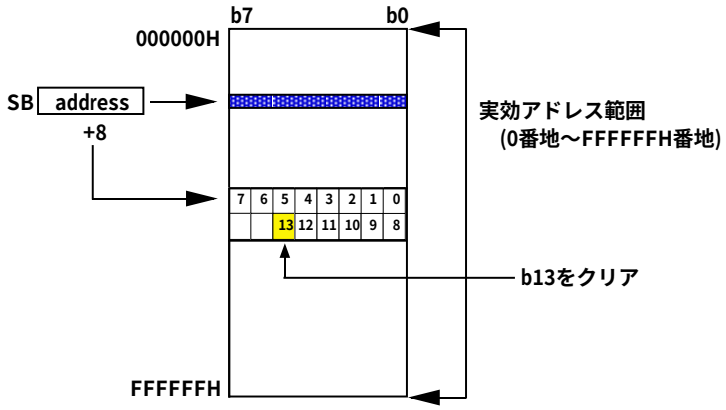


図 2.5.30　ビット命令 SB 相対アドレッシング

FB 相対

基準はFBレジスタの値が示すアドレスです。FBレジスタの値にbaseを符号付きで加算します。その値が示すアドレスのビット0から、bitで示したビット数だけ離れたビットが演算対象となります。

bit,base:11、bit,base:19で指定できるアドレスの範囲はFBレジスタ値を中心に256バイト、65536バイトです。

ただし、対象となるビットのアドレスが000000H～FFFFFFHを越えた場合、25ビット以上は無視され、000000H番地側、またはFFFFFFH番地側に戻ります。

(記号)bit,base:11[FB]、bit,base:19[FB]
(例)BCLR　5,-8[FB]

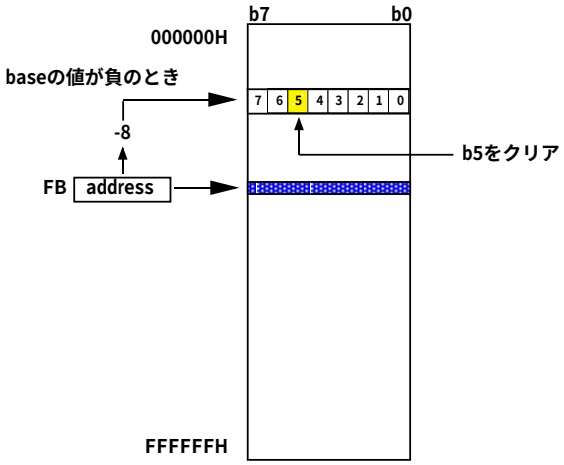


図 2.5.31　ビット命令 FB 相対アドレッシング

2.6 命令セット

M16C/80シリーズの命令セットについて説明します。命令セットは機能別に一覧表としてまとめました。さらに命令セットの中から特長のある命令について詳しく説明します。

以下に一覧表の中で使用する記号と意味を示します。

記号	意味
src	処理結果を格納しないオペランド
dest	処理結果を格納するオペランド
label	アドレスを意味するオペランド
abs16/24	絶対値（16 ビット、24 ビット）
dsp:8/16/24	変位（8 ビット、16 ビット、24 ビット）
#IMM4/8/16/24/32	即値（4 ビット、8 ビット、16 ビット、24 ビット、32 ビット）
.size	サイズ指定子(.B、.W、L)
.length	分岐距離指定子(.S、.B、.W、.A)
←	矢印の向きに転送
＋	加算
－	減算
×	乗算
÷	除算
∧	論理積
∨	論理輪
∀	排他的論理輪
¬	否定
| |	絶対値
EXT( )	( ) 内の符号拡張
U、I、O、B、S、Z、D、C	フラグ名
R0L、R0H、R1L、R1H	8 ビットのレジスタ名
R0、R1、R2、R3、A0、A1	16 ビットのレジスタ名
R2R0、R3R1、A1A0	32 ビットのレジスタ名
SB、FB、SP、PC、...	レジスタ名
MOV*Dir*、BM*Cnd*、J*Cnd*、...	*Dir*（方向）、*Cnd*（条件）のニーモニックは斜体で示す。
J*GEU/C*、J*EQ/Z*	J*GEU/C* は J*GEU* または J*C* と書き J*EQ/Z* は J*EQ* または J*Z* と書くことを示す。
INDEX*type*	type（モデファイタイプ）のニーモニックは斜体で示す。
"○"	（アドレッシング） 使用できる。
	（フラグ変化） 実行結果にしたがってフラグが変化する。
"－"	（フラグ変化） フラグは変化しない。

2.6.1 命令の記述

ここでは M16C/80 の命令記述形式について説明します。

命令の記述形式

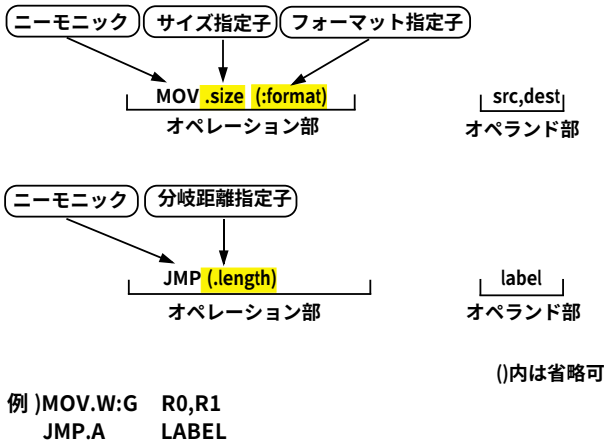


図 2.6.1 命令の記述形式

ニーモニック：命令の動作を示します。
サイズ指定子：ニーモニックの処理対象となるデータサイズを指定します。
分岐距離指定子：分岐命令及びサブルーチンコール命令の分岐先への距離を指定します（通常は省略します）。
フォーマット指定子：オペコードの形式を指定します。オペコードの形式によってオペコードおよびオペランドのコード長が異なります（通常は省略します）。

― 指定子 ―

サイズ指定子	内容
.B	バイト（8ビット）サイズを示します
.W	ワード（16ビット）サイズを示します
.L	ロングワード（32ビット）を示します

分岐距離指定子	内容
.S	分岐距離：+2～+9（3ビット前方相対）
.B	分岐距離：-128～+127（8ビット相対）
.W	分岐距離：-32768～+32767（16ビット相対）
.A	分岐距離：0～FFFFFFH（24ビット絶対）

分岐距離指定子	内容
.Z	ゼロ形式
.S	ショート形式
.Q	クイック形式
.G	ジェネリック形式

選択優先順位
高
↕
低

（注）命令フォーマット指定子を持たない命令もあります。

図 2.6.2 指定子

ー 命令フォーマット ー

1. ジェネリック形式(:G)

オペコードには動作および src と dest のアドレッシング情報も含まれます。

表 2.6.1　ジェネリック形式

オペコード	srcコード	destコード
2～3バイト	0～4バイト	0～3バイト

2. クイック形式(:Q)

オペコードには動作および即値データと dest のアドレッシング情報も含まれます。ただし、オペコードに含まれる即値データは -7 ～ +8 または -8 ～ +7(命令によって異なります)で表現できる数値です。

表 2.6.2　クイック形式

オペコード	destコード
2バイト	0～3バイト

3. ショート形式(:S)

オペコードには動作および src と dest のアドレッシング情報も含まれます。ただし、使用できるアドレッシングモードは制限されます。S 形式は一部のアドレッシングで使用します。

表 2.6.3　ショート形式

オペコード	srcコード	destコード
1バイト	0～2バイト	0～2バイト

4. ゼロ形式(:Z)

オペコードには動作および即値データと dest のアドレッシング情報も含まれます。　　ただし、即値データは 0 固定です。Z 形式は一部のアドレッシングで使用します。

表 2.6.4　ゼロ形式

オペコード	destコード
1バイト	0～2バイト

2.6.2 命令一覧

機能別に各ニーモニックの内容、アドレッシング、フラグの変化について示します。

転送

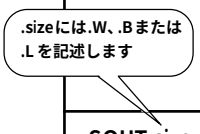
ニーモニック	説　　明
MOV.size src,dest	srcをdestに転送、または即値をdestにセット
MOVA src,dest	srcのアドレスをdestに転送
MOV*HH* src,dest	srcの上位4ビットをdestの上位4ビットに転送
MOV*HL* src,dest	srcの上位4ビットをdestの下位4ビットに転送
MOV*LH* src,dest	srcの下位4ビットをdestの上位4ビットに転送
MOV*LL* src,dest	srcの下位4ビットをdestの下位4ビットに転送
MOVX src,dest	8ビットの即値を32ビットに符号拡張後destに転送
POP.size dest	スタック領域から値を復帰
POPC dest	スタック領域からdestで示す専用レジスタに復帰
POPM dest	複数のレジスタの値を一括してスタック領域から復帰
PUSH.size src	レジスタ/メモリ/即値をスタック領域へ待避
PUSHA src	srcのアドレスをスタック領域へ待避
PUSHC src	srcの専用レジスタをスタック領域へ待避
PUSHM src	複数のレジスタをスタック領域へ待避
SIN.size	A0を固定の転送元番地、A1を転送先番地、R3を転送回数として、アドレスの加算方向へストリング転送
SMOVB.size	A0を転送元番地、A1を転送先番地、R3を転送回数として、アドレスの減算方向へストリング転送
SMOVF.size	A0を転送元番地、A1を転送先番地、R3を転送回数として、アドレスの加算方向へストリング転送
SMOVU.size	A0を転送元番地、A1を転送先番地、R3を転送回数として、アドレスの加算方向へ0が検出されるまでストリング転送

.size には .W、.B または .L を記述します

＊a　src または dest に R0L レジスタを選択します。
＊b　R0L、R0H、R1L、R1H から選択できます。
＊c　即値は 8 ビットで、とりうる範囲は
-128 ≦ #IMM8 ≦ +127 です。
＊d　SP 指定の時は U フラグで示すスタックポインタが対象です。
＊e　dest は FLG 指定の時のみ変化します。

アドレッシング											フラグ変化							
オペランド	一般命令								特定命令									
	即値	絶対	レジスタ直接	アドレスレジスタ間接	アドレスレジスタ相対	SB相対	FB相対	SP相対	専用レジスタ直接	プログラムカウンタ相対	U	I	O	B	S	Z	D	C
src*f	○	○	○	○	○	○	○	○			－	－	－	－	○	○	－	－
dest*f		○	○	○	○	○	○	○										
src		○			○	○	○				－	－	－	－	－	－	－	－
dest			○															
src			ROL*a								－	－	－	－	－	－	－	－
dest		○	○*b	○	○	○	○											
src		○	○*b	○	○	○	○											
dest			ROL*a															
src	○*c										－	－	－	－	○	○	－	－
dest*f		○	○	○	○	○	○											
dest*f		○	○	○	○	○	○				－	－	－	－	－	－	－	－
dest									○*d		*e	*e	*e	*e	*e	*e	*e	*e
dest									○		－	－	－	－	－	－	－	－
src*f	○	○	○	○	○	○	○				－	－	－	－	－	－	－	－
src		○			○	○	○				－	－	－	－	－	－	－	－
src									○*d		－	－	－	－	－	－	－	－
src			○						○		－	－	－	－	－	－	－	－
											－	－	－	－	－	－	－	－
											－	－	－	－	－	－	－	－
											－	－	－	－	－	－	－	－
											－	－	－	－	－	－	－	－

*f　間接アドレッシング[src]、[dest]は、R0L/R0/R2R0、R0H/R2/-、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM 以外で使用できます。



ニーモニック	説　　明
SOUT.size	A0を転送元番地、A1を固定の転送先番地、R3を転送回数として、アドレスの加算方向へストリング転送
SSTR.size	R0L/R0をストアデータ、A1を転送先番地、R3を転送回数として、アドレスの加算方向へストリング転送
STNZ.size src,dest	Zフラグが"0"のときsrcをdestへ転送
STZ.size src,dest	Zフラグが"1"のときsrcをdestへ転送
STZX.size src1,src2,dest	Zフラグが"1"のときsrc1をdestへ転送、"0"のときsrc2をdestへ転送
XCHG.size src,dest	srcとdestの内容を交換

＊g　即値は 8/16 ビットを選択します。

アドレッシング											フラグ変化							
オペランド	一般命令								特定命令		U	I	O	B	S	Z	D	C
	即値	絶対	レジスタ直接	アドレスレジスタ間接	アドレスレジスタ相対	SB相対	FB相対	SP相対	専用レジスタ直接	プログラムカウンタ相対								
											−	−	−	−	−	−	−	−
											−	−	−	−	−	−	−	−
src	○*g										−	−	−	−	−	−	−	−
dest*h		○	○	○	○	○	○											
src	○*g										−	−	−	−	−	−	−	−
dest*h		○	○	○	○	○	○											
src	○										−	−	−	−	−	−	−	−
dest*h		○	○	○	○	○	○											
src			○								−	−	−	−	−	−	−	−
dest*h		○	○	○	○	○	○											

*h　間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM 以外で使用できます。

ビット処理

ニーモニック	説　明
BAND src	Cフラグ ← src ∧ Cフラグ ;ビット論理積
BCLR dest	dest ← 0 ;ビットクリア
BITINDEX.size src	srcで指定したオペランドが次のビット命令のsrcまたはdestのインデックス値となる
BM*GEU/C* dest	C=1ならば dest ← 1、 それ以外は dest ← 0 ;条件ビット転送
BM*LTU/NC* dest	C=0ならば dest ← 1、 それ以外は dest ← 0
BM*EQ/Z* dest	Z=1ならば dest ← 1、 それ以外は dest ← 0
BM*NE/NZ* dest	Z=0ならば dest ← 1、 それ以外は dest ← 0
BM*GTU* dest	C∧Z=1ならば dest ← 1、 それ以外は dest ← 0
BM*LEU* dest	C∧Z=0ならば dest ← 1、 それ以外は dest ← 0
BM*PZ* dest	S=0ならば dest ← 1、 それ以外は dest ← 0
BM*N* dest	S=1ならば dest ← 1、 それ以外は dest ← 0
BM*GE* dest	S∀O=0ならば dest ← 1、 それ以外は dest ← 0
BM*LE* dest	(S∀O)∨Z=1ならば dest ← 1、 それ以外は dest ← 0
BM*GT* dest	(S∀O)∨Z=0ならば dest ← 1、 それ以外は dest ← 0
BM*LT* dest	S∀O=1ならば dest ← 1、 それ以外は dest ← 0
BM*O* dest	O=1ならば dest ← 1、 それ以外は dest ← 0
BM*NO* dest	O=0ならば dest ← 1、 それ以外は dest ← 0
BNAND src	Cフラグ ← $\overline{\text{src}}$ ∧ Cフラグ ;反転ビット論理積
BNOR src	Cフラグ ← $\overline{\text{src}}$ ∨ Cフラグ ;反転ビット論理和
BNOT dest	destを反転し、destに格納 ;ビット反転
BNTST src	Zフラグ ← $\overline{\text{src}}$、 Cフラグ ← $\overline{\text{src}}$;反転ビットテスト
BNXOR src	Cフラグ ← $\overline{\text{src}}$ ∀ Cフラグ ;反転ビット排他的論理和
BOR src	Cフラグ ← src ∨ Cフラグ ;ビット論理和
BSET dest	dest ← 1 ;ビットセット
BTST src	Zフラグ ← $\overline{\text{src}}$、 Cフラグ ← src ;ビットテスト
BTSTC dest	Zフラグ ← $\overline{\text{dest}}$、Cフラグ ← dest、dest ← 0 ;ビットテスト&クリア
BTSTS dest	Zフラグ ← $\overline{\text{dest}}$、Cフラグ ← dest、dest ← 1 ;ビットテスト&セット
BXOR src	Cフラグ ← src ∀ Cフラグ ;ビット排他的論理輪

アドレッシング								フラグ変化							
オペランド	ビット命令							U	I	O	B	S	Z	D	C
	絶対	レジスタ直接	アドレスレジスタ間接	アドレスレジスタ相対	SB相対	FB相対	FLG直接								
src	○	○	○	○	○	○		−	−	−	−	−	−	−	○
dest	○	○	○	○	○	○		−	−	−	−	−	−	−	−
src	○	○	○	○	○	○		−	−	−	−	−	−	−	−
dest	○	○	○	○	○	○	○	−	−	−	−	−	−	−	○*i
src	○	○	○	○	○	○		−	−	−	−	−	−	−	○
src	○	○	○	○	○	○		−	−	−	−	−	−	−	○
dest	○	○	○	○	○	○		−	−	−	−	−	−	−	−
src	○	○	○	○	○	○		−	−	−	−	−	○	−	○
src	○	○	○	○	○	○		−	−	−	−	−	−	−	○
src	○	○	○	○	○	○		−	−	−	−	−	−	−	○
dest	○	○	○	○	○	○		−	−	−	−	−	−	−	−
src	○	○	○	○	○	○		−	−	−	−	−	○	−	○
dest	○	○	○	○	○	○		−	−	−	−	−	○	−	○
dest	○	○	○	○	○	○		−	−	−	−	−	○	−	○
src	○	○	○	○	○	○		−	−	−	−	−	−	−	○

＊i　dest に C フラグを指定したとき、変化します。

演算

ニーモニック		説　明	
ABS.size	dest	dest ← ¦dest¦	;destの絶対値
ADC.size	src,dest	dest ← src + dest + Cフラグ	;キャリー付き16進加算
ADCF.size	dest	dest ← dest + Cフラグ	;キャリーフラグの加算
ADD.size	src,dest	dest ← src + dest	;キャリーなし16進加算
ADDX	src,dest	dest ← dest - 32ビット符号拡張(src)	;符号拡張キャリーなし16進加算
AND.size	src,dest	dest ← src ∧ dest	;論理積
CLIP.size	src1,src2,dest	if src1＞dest then dest←src1、if src2＜dest then dest←src2	;クリップ命令
CMP.size	src,dest	dest - src	;比較、結果はフラグで判断
CMPX	src,dest	dest - 32ビット符号拡張(src)	;比較、結果はフラグで判断
DADC.size	src,dest	dest ← src + dest + Cフラグ	;キャリー付き10進加算
DADD.size	src,dest	dest ← src + dest	;キャリーなし10進加算
DEC.size	dest	dest ← dest - 1	;デクリメント
DIV.size	src	R0(商),R2(剰余) ← R2R0 ÷ src	;符号付き除算
DIVU.size	src	R0(商),R2(剰余) ← R2R0 ÷ src	;符号なし除算
DIVX.size	src	R0(商),R2(剰余) ← R2R0 ÷ src	;符号付き除算
DSBB.size	src,dest	dest ← dest - src - Cフラグ	;ボロー付き10進減算

.size には .B、.W または .L を記述します

アドレッシング											フラグ変化							
オペランド	一般命令								特定命令									
	即値	絶対	レジスタ直接	アドレスレジスタ間接	アドレスレジスタ相対	SB相対	FB相対	SP相対	専用レジスタ直接	プログラムカウンタ相対	U	I	O	B	S	Z	D	C
dest*j		○	○	○	○	○	○				−	−	○	−	○	○	−	○
src	○	○	○	○	○	○	○				−	−	○	−	○	○	−	○
dest		○	○	○	○	○	○											
dest*j		○	○	○	○	○	○				−	−	○	−	○	○	−	○
src*j	○	○	○	○	○	○	○				−	−	○	−	○	○	−	○
dest*j		○	○	○	○	○	○		SP									
src*j	○	○	○	○	○	○	○				−	−	○	−	○	○	−	○
dest*j		○	○	○	○	○	○											
src*j	○	○	○	○	○	○	○				−	−	−	−	○	○	−	−
dest*j		○	○	○	○	○	○											
src	○										−	−	−	−	−	−	−	−
dest		○	○	○	○	○	○											
src*j	○	○	○	○	○	○	○				−	−	○	−	○	○	−	○
dest*j		○	○	○	○	○	○											
src	○										−	−	○	−	○	○	−	○
dest*j		○	○	○	○	○	○											
src	○	○	○	○	○	○	○				−	−	−	−	○	○	−	○
dest		○	○	○	○	○	○											
src	○	○	○	○	○	○	○				−	−	−	−	○	○	−	○
dest		○	○	○	○	○	○											
dest*j		○	○	○	○	○	○				−	−	−	−	○	○	−	−
src*j	○	○	○	○	○	○	○				−	−	○	−	−	−	−	−
src*j	○	○	○	○	○	○	○				−	−	○	−	−	−	−	−
src*j	○	○	○	○	○	○	○				−	−	○	−	−	−	−	−
src	○	○	○	○	○	○	○				−	−	−	−	○	○	−	○
dest		○	○	○	○	○	○											

*j　間接アドレッシング[src]、[dest]は、R0L/R0/R2R0、R0H/R2/-、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

ニーモニック		説　明	
DSUB.size	src,dest	dest ← dest - src	;ボローなし10進減算
INC.size	dest	dest ← dest + 1	;インクリメント
MAX.size	src,dest	if (src＞dest) then dest ← src	;最大値選択
MIN.size	src,dest	if (src＜dest) then dest ← src	;最小値選択
MUL.size	src,dest	dest ← dest ×src	;符号付き乗算
MULEX	src	R1R2R0t ← R2R0 ×src	;拡張符号付き乗算
MULU.size	src,dest	dest ← dest ×src	;符号なし乗算
NEG.size	dest	dest ← 0 - dest	;2の補数
NOT.size	dest	dest ← dest	;全ビット反転
OR.size	src,dest	dest ← dest ∨ src	;論理和
RMPA.size		A0を被乗数番地、A1を乗数番地、R3を回数とする積和演算	
ROLC.size	dest	destをCフラグを含めて1ビット左へ回転	;キャリー付き左回転
RORC.size	dest	destをCフラグを含めて1ビット右へ回転	;キャリー付き右回転
ROT.size	src,dest	destを符号付きのsrcで示すビット数分数分回転	;回転
SBB.size	src,dest	dest ← dest - src - C	;ボロー付き減算

.sizeには.Wか.Bを記述します

アドレッシング											フラグ変化							
オペランド	一般命令								特定命令									
	即値	絶対	レジスタ直接	アドレスレジスタ間接	アドレスレジスタ相対	SB相対	FB相対	SP相対	専用レジスタ直接	プログラムカウンタ相対	U	I	O	B	S	Z	D	C
src	○	○	○	○	○	○	○				－	－	－	－	○	○	－	○
dest		○	○	○	○	○	○											
dest*k		○	○	○	○	○	○				－	－	－	－	○	○	－	○
src	○	○	○	○	○	○	○				－	－	－	－	－	－	－	－
dest		○	○	○	○	○	○											
src	○	○	○	○	○	○	○				－	－	－	－	－	－	－	－
dest		○	○	○	○	○	○											
src*k	○	○	○	○	○	○	○				－	－	－	－	－	－	－	－
dest*k		○	○	○	○	○	○											
src*k		○	○	○	○	○	○				－	－	－	－	－	－	－	－
src*k	○	○	○	○	○	○	○				－	－	－	－	－	－	－	－
dest*k		○	○	○	○	○	○											
dest*k		○	○	○	○	○	○				－	－	○	－	○	○	－	○
dest*k		○	○	○	○	○	○				－	－	－	－	○	○	－	○
src*k	○	○	○	○	○	○	○				－	－	－	－	○	○	－	－
dest*k		○	○	○	○	○	○											
											－	－	○	－	－	－	－	－
dest*k		○	○	○	○	○	○				－	－	－	－	○	○	－	○
dest*k		○	○	○	○	○	○				－	－	－	－	○	○	－	○
src*k	○*l	○*m									－	－	－	－	○	○	－	○
dest*k		○	○	○	○	○	○											
src	○	○	○	○	○	○	○				－	－	○	－	○	○	－	○
dest		○	○	○	○	○	○											

*j　間接アドレッシング[src]、[dest]は、R0L/R0/R2R0、R0H/R2/-、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM 以外で使用できます。

*l　とりうる値の範囲は -8 ≦ #IMM4 ≦ +8 です。

*m　R1H を選択します。

ニーモニック	説　明
SC*GEU/C* dest	C=1ならば dest ← 1、 それ以外は dest ← 0 ;条件設定
SC*LTU/NC* dest	C=0ならば dest ← 1、 それ以外は dest ← 0
SC*EQ/Z* dest	Z=1ならば dest ← 1、 それ以外は dest ← 0
SC*NE/NZ* dest	Z=0ならば dest ← 1、 それ以外は dest ← 0
SC*GTU* dest	C∧Z=1ならば dest ← 1、 それ以外は dest ← 0
SC*LEU* dest	C∧Z=0ならば dest ← 1、 それ以外は dest ← 0
SC*PZ* dest	S=0ならば dest ← 1、 それ以外は dest ← 0
SC*N* dest	S=1ならば dest ← 1、 それ以外は dest ← 0
SC*GE* dest	S∀O=0ならば dest ← 1、 それ以外は dest ← 0
SC*LE* dest	(S∀O)∨Z=1ならば dest ← 1、 それ以外は dest ← 0
SC*GT* dest	(S∀O)∨Z=0ならば dest ← 1、 それ以外は dest ← 0
SC*LT* dest	S∀O=1ならば dest ← 1、 それ以外は dest ← 0
SC*O* dest	O=1ならば dest ← 1、 それ以外は dest ← 0
SC*NO* dest	O=0ならば dest ← 1、 それ以外は dest ← 0
SCMPU.size	A0を比較元番地、A1を比較先番地としてアドレスの加算方向へ比較していき、不一致になるか比較元番地が0になるまでストリング比較を行う。
SHA.size src,dest	destをｓｒｃで示すビット数分算術シフトする。LSB(MSB)からあふれたビットはCフラグに転送する。
SHL.size src,dest	destをｓｒｃで示すビット数分論理シフトする。LSB(MSB)からあふれたビットはCフラグに転送する。
SUB.size src,dest	dest ← dest - src ;ボローなし減算
SUBX src,dest	dest ← dest - 32ビット符号拡張(src) ;ボローなし拡張減算
TST.size src,dest	dest ∧ src ;テスト
XOR.size src,dest	dest ← dest ∀ src ;排他的論理和

.sizeには.B、.Wまたは.Lを記述します

アドレッシング											フラグ変化							
オペランド	一般命令								特定命令									
	即値	絶対	レジスタ直接	アドレスレジスタ間接	アドレスレジスタ相対	SB相対	FB相対	SP相対	専用レジスタ直接	プログラムカウンタ相対	U	I	O	B	S	Z	D	C
dest*n		○	○	○	○	○	○				—	—	—	—	—	—	—	—
src											—	—	○	—	○	○	—	○
src*n	○*o		R1H								—	—	○	—	○	○	—	○
dest*n		○	○	○	○	○	○											
src*n	○*o		R1H								—	—	○	—	○	○	—	○
dest*n		○	○	○	○	○	○											
src*n	○	○	○	○	○	○	○				—	—	○	—	○	○	—	○
dest*n		○	○	○	○	○	○											
src*n	○	○	○	○	○	○	○				—	—	○	—	○	○	—	○
dest*n		○	○	○	○	○	○											
src	○	○	○	○	○	○	○				—	—	—	—	○	○	—	—
dest		○	○	○	○	○	○											
src*n	○	○	○	○	○	○	○				—	—	—	—	○	○	—	—
dest*n		○	○	○	○	○	○											

*n　間接アドレッシング[src]、[dest]は、R0L/R0/R2R0、R0H/R2/-、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM 以外で使用できます。

*o　(.size)が(.B)、(.W)のとき、とりうる値の範囲は -8 ≦ #IMM4 ≦ +8(≠ 0)、(.L)のとき -16 ≦ #IMM8 ≦ +16(≠ 0)です。

ジャンプ

ニーモニック		説　明
.sizeには.Wか.Bを記述します		
ADJNZ.size	src,dest,label	dest ← dest + src　の結果が0以外であればlabelへ分岐　;加算&条件分岐
J*GEU/C*	label	C=1ならばlabelへ分岐、　それ以外は次の命令を実行　;条件分岐
J*LTU/NC*	label	C=0ならばlabelへ分岐、　それ以外は次の命令を実行
J*EQ/Z*	label	Z=1ならばlabelへ分岐、　それ以外は次の命令を実行
J*NE/NZ*	label	Z=0ならばlabelへ分岐、　それ以外は次の命令を実行
J*GTU*	label	C∧Z=1ならばlabelへ分岐、　それ以外は次の命令を実行
J*LEU*	label	C∧Z=0ならばlabelへ分岐、　それ以外は次の命令を実行
J*PZ*	label	S=0ならばlabelへ分岐、　それ以外は次の命令を実行
J*N*	label	S=1ならばlabelへ分岐、　それ以外は次の命令を実行
J*GE*	label	S∀O=0ならばlabelへ分岐、　それ以外は次の命令を実行
J*LE*	label	(S∀O)∨Z=1ならばlabelへ分岐、　それ以外は次の命令を実行
J*GT*	label	(S∀O)∨Z=0ならばlabelへ分岐、　それ以外は次の命令を実行
J*LT*	label	S∀O=1ならばlabelへ分岐、　それ以外は次の命令を実行
J*O*	label	O=1ならばlabelへ分岐、　それ以外は次の命令を実行
J*NO*	label	O=0ならばlabelへ分岐、　それ以外は次の命令を実行
JMP	label	labelへ分岐　;無条件分岐
JMPI	src	srcが示す番地へ分岐　;間接分岐
JMPS	src	スペシャルページベクタテーブルによる分岐
JSR	label	サブルーチン呼び出し
JSRI	src	間接サブルーチン呼び出し
JSRS	src	スペシャルページベクタテーブルによるサブルーチン呼び出し
RTS		サブルーチンからの復帰
SBJNZ.size	src,dest,label	dest ← dest - src　の結果が0以外であればlabelへ分岐　;減算&条件分岐

アドレッシング											フラグ変化							
オペランド	一般命令								特定命令		U	I	O	B	S	Z	D	C
	即値	絶対	レジスタ直接	アドレスレジスタ間接	アドレスレジスタ相対	SB相対	FB相対	SP相対	専用レジスタ直接	プログラムカウンタ相対								
src	○*p										—	—	—	—	—	—	—	—
dest		○	○	○	○	○	○											
label										label*q								
label										label*r	—	—	—	—	—	—	—	—
label		○*s								label*t	—	—	—	—	—	—	—	—
src		○	○	○	○	○	○				—	—	—	—	—	—	—	—
src		○	○	○	○	○	○				—	—	—	—	—	—	—	—
src	○*u										—	—	—	—	—	—	—	—
src		○	○	○	○	○	○				—	—	—	—	—	—	—	—
src	○*u										—	—	—	—	—	—	—	—
											—	—	—	—	—	—	—	—
src	○*v										—	—	—	—	—	—	—	—
dest		○	○	○	○	○	○											
label										label*q								

*p　即値の範囲は -8 ≦ #IMM4 ≦ +7 です。
*q　label の範囲は PC-126 ≦ label ≦ PC+129 です。PC は命令の先頭番地です。
*r　label の範囲は PC-127 ≦ label ≦ PC+128 です。PC は命令の先頭番地です。
*s　24 ビット絶対アドレスです。
*t　label の範囲は PC-32767 ≦ label ≦ PC+32768 です。PC は命令の先頭番地です。
*u　#IMM8 はスペシャルページ番号です。
*v　即値の範囲は -7 ≦ #IMM4 ≦ +8 です。

符号拡張

ニーモニック (.sizeには.Wか.Bを記述します)		説　明
EXTS.size	dest	dest ← .sizeに従った符号拡張(dest)
EXTS.size	src,dest	dest ← sizeに従った符号拡張(src)
EXTZ	src,dest	dest ← 16ビットの0拡張(src)

インデックス

ニーモニック (.sizeには.Wか.Bを記述します)		説　明
INDEX*B*.size	src	次の命令のアドレッシングをバイト単位でモディファイする
INDEX*BD*.size	src	
INDEX*BS*.size	src	
INDEX*W*.size	src	次の命令のアドレッシングをワード単位でモディファイする
INDEX*WD*.size	src	
INDEX*WS*.size	src	
INDEX*L*.size	src	次の命令のアドレッシングをロングワード単位でモディファイする
INDEX*LD*.size	src	
INDEX*LS*.size	src	

オペランド	アドレッシング										フラグ変化							
	一般命令								特定命令									
	即値	絶対	レジスタ直接	アドレスレジスタ間接	アドレスレジスタ相対	SB相対	FB相対	SP相対	専用レジスタ直接	プログラムカウンタ相対	U	I	O	B	S	Z	D	C
dest		○	○	○	○	○	○				—	—	—	—	○	○	—	—
src		○	○		○	○	○				—	—	—	—	○	○	—	—
dest		○	○	○	○	○	○											
src		○	○		○	○	○				—	—	—	—	○	○	—	—
dest		○	○	○	○	○	○											

オペランド	アドレッシング										フラグ変化							
	一般命令								特定命令									
	即値	絶対	レジスタ直接	アドレスレジスタ間接	アドレスレジスタ相対	SB相対	FB相対	SP相対	専用レジスタ直接	プログラムカウンタ相対	U	I	O	B	S	Z	D	C
src		○	○	○	○	○	○				—	—	—	—	—	—	—	—
src		○	○	○	○	○	○				—	—	—	—	—	—	—	—
src		○	○	○	○	○	○				—	—	—	—	—	—	—	—

高級言語、OS サポート

ニーモニック		説　　明
ENTER	src	スタックフレームの生成
EXITD	src	スタックフレームの解放
LDCTX	abs16,abs24	コンテキストの復帰
STCTX	abs16,abs24	コンテキストの待避

アドレッシング											フラグ変化							
オペランド	一般命令								特定命令									
	即値	絶対	レジスタ直接	アドレスレジスタ間接	アドレスレジスタ相対	SB相対	FB相対	SP相対	専用レジスタ直接	プログラムカウンタ相対	U	I	O	B	S	Z	D	C
src	○										—	—	—	—	—	—	—	—
		○	○	○	○	○	○				—	—	—	—	—	—	—	—
abs16*w											—	—	—	—	—	—	—	—
abs24*w																		
abs16*w											—	—	—	—	—	—	—	—
abs24*w																		

*w　abs16 にはタスク番号が格納されている RAM の番地、abs24 にはテーブルデータの先頭番地を設定します。

その他

ニーモニック		説　　明
BRK		BRK命令割り込みの発生
BRK2		デバッガ専用割り込みなのでユーザプログラムでの使用は禁止
FCLR	dest	フラグレジスタのフラグを “ 0 ” にクリア
FREIT		高速割り込み要求後の割り込みルーチンからの復帰
FSET		フラグレジスタのフラグを “ 1 ” にセット
INT		ソフトウェア割り込みの発生
INTO		O（オーバーフロー）フラグが “ 1 ” のとき、オーバーフロー割り込みの発生
LDC	src,dest	srcをdestが示す専用レジスタへ転送
LDIPL	src	srcをIPLへ転送
NOP		ノーオペレーション
REIT		割り込みルーチンからの復帰
STC	src,dest	srcで示す専用レジスタからdestへ転送
UND		未定義命令割り込みの発生
WAIT		プログラムの実行停止

アドレッシング											フラグ変化							
オペランド	一般命令								特定命令		U	I	O	B	S	Z	D	C
	即値	絶対	レジスタ直接	アドレスレジスタ間接	アドレスレジスタ相対	SB相対	FB相対	SP相対	専用レジスタ直接	プログラムカウンタ相対								
											○	○	—	—	—	—	○	—
											○	○	—	—	—	—	○	—
dest									○		選択したフラグが “0” になります							
											SVF の内容になります							
dest											選択したフラグが “1” になります							
src	○*x										○	○	—	—	—	—	○	—
											○	○	—	—	—	—	○	—
src	○	○	○	○	○	○	○				dest が FLG のときのみ変化します							
dest									○*y									
src	○*z										—	—	—	—	—	—	—	—
											—	—	—	—	—	—	—	—
											割り込み要求が受け付けられる前の FLG の状態に戻ります							
src	○*y										—	—	—	—	—	—	—	—
dest		○	○	○	○	○	○											
											○	○	—	—	—	—	○	—
											—	—	—	—	—	—	—	—

*x　#IMM6 はソフトウェア割り込み番号を指定します。
*y　PC レジスタを除く専用レジスタを選択できます。
*z　取りうる範囲は 0 ≦ #IMM3 ≦ 7 です。

2.6.3　転送命令

転送は通常バイトまたはワード単位で行います。転送命令は21命令用意されています。その中には、4ビットだけを転送する4ビット転送命令や、条件分岐を組み合わせた条件ストア命令、データをまとめて転送するストリング命令があります。

この項ではデータ転送に関する命令の中から上記の特長のある命令3種類について説明します。

4ビット転送命令

4ビット転送命令は、8ビットのレジスタ / メモリの上位4ビットまたは下位4ビットを転送する命令です。4ビット転送はアンパックド BCD コードの作成、4ビット単位の I/O ポート入出力に利用できます。

転送する4ビットが、上位か下位かによって *Dir* に入るニーモニックが変わります。

この4ビット転送命令では src または dest のどちらか一方を R0L にしてください。

表 2.6.5　4ビット転送命令

ニーモニック	記述形式	説明

MOVDir	MOVHH src,dest
	MOVHL src,dest
	MOVLH src,dest
	MOVLL src,dest
		転送
		上位4ビット:src → 上位4ビット:dest
		上位4ビット:src → 下位4ビット:dest
		下位4ビット:src → 上位4ビット:dest
		下位4ビット:src → 下位4ビット:dest

(注)srcかdestのどちらか一方は必ずR0L

条件ストア命令

条件ストア命令は、Z フラグの状態が条件となる条件付き転送命令です。この命令は、条件判断と転送を 1 命令で行うことができます。

条件ストア命令は、STZ、STNZ、STZX の 3 種類です。動作例を図 2.6.3 に示します。

表 2.6.6　条件ストア命令

ニーモニック	記述形式	説明
STZ	STZ　src,dest	Zフラグ=1のとき srcをdestに転送する
STNZ	STNZ　src,dest	Zフラグ=0のとき srcをdestに転送する
STZX	STZX　src1,src2,dest	Zフラグ=1のとき src1をdestに転送する Zフラグ=0のとき src2をdestに転送する

(注)src,src1,src2は、#IMM8/16が選択できます。

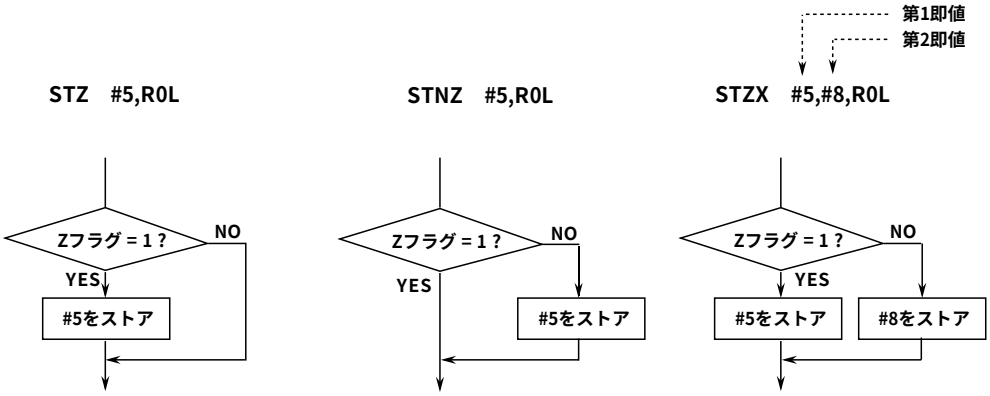


図 2.6.3　条件ストア命令の動作例

ストリング命令

ストリング命令はデータをまとめて転送する命令です。ブロック転送や RAM クリアに利用します。

図2.6.4に示すように、各レジスタに転送元アドレス、転送先アドレス、転送回数をセットして命令を実行します。転送はバイトまたはワード単位で行います。ストリング命令の動作例を図 2.6.5 に示します。

図 2.6.4　ストリング命令のレジスタ設定

表 2.6.7　ストリング命令

ニーモニック	記述形式	説明
SIN	SIN.B SIN.W	アドレスの加算方向へストリング転送する（転送元は固定）
SOUT	SOUT.B SOUT.W	アドレスの加算方向へストリング転送する（転送先は固定）
SMOVB	SMOVB.B SMOVB.W	アドレスの減算方向へストリング転送する
SMOVF	SMOVF.B SMOVF.W	アドレスの加算方向へストリング転送する
SMOVU	SMOVU.B SMOVU.W	アドレスの加算方向へ0が検出されるまでストリング転送する
SSTR	SSTR.B SSTR.W	アドレスの加算方向へストリングストアする

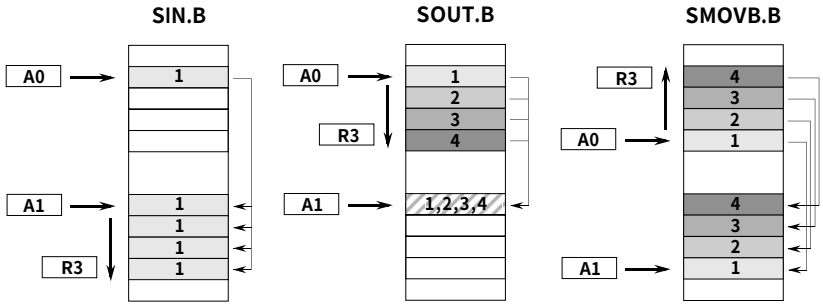


図 2.6.5　ストリング命令の動作例 1

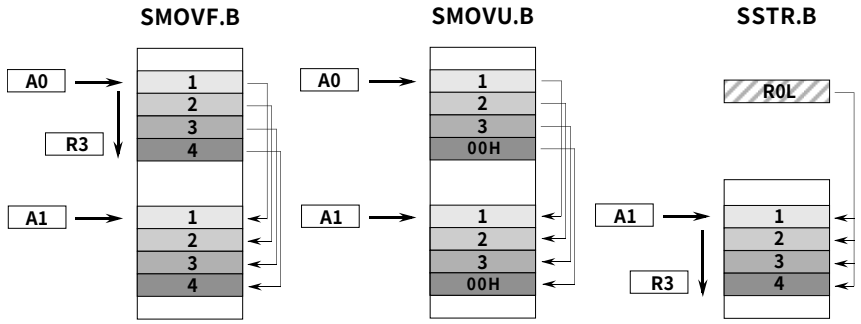


図 2.6.6　ストリング命令の動作例 2

2.6.4 演算命令

演算命令は 39 命令用意されています。この項では特長のある演算命令を説明します。

乗算命令

乗算命令には符号付き乗算命令と拡張符号付き乗算命令および符号なし乗算命令があります。符号付き乗算命令と符号なし乗算命令はサイズ指定ができます。.B を指定すると(8 ビット)×(8 ビット)=(16 ビット)で演算し、.W を指定すると(16 ビット)×(16 ビット)=(32 ビット)で演算します。

.B を指定した場合には、src と dest の両方にアドレスレジスタを使うことはできません。また、乗算命令ではフラグは変化しません。乗算命令の動作例を図 2.6.7 に示します。

表 2.6.8 乗算命令

ニーモニック	記述形式	説明
MUL	MUL.B src,dest MUL.W src,dest	符号付き乗算命令 dest ← desc ×src
MULEX	MULEX src,dest MULEX src,dest	拡張符号付き乗算命令 R1R2R0 ← R2R0 ×src
MULU	MULU.B src,dest MULU.W src,dest	符号なし乗算命令 dest ← dest ×src

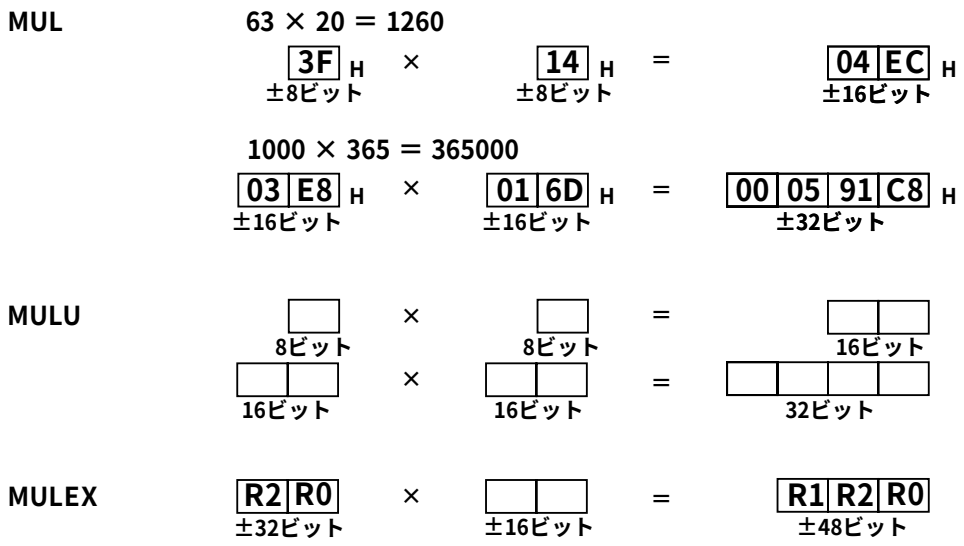


図 2.6.7 乗算命令の動作例

除算命令

除算命令には符号付き除算命令が 2 つと符号なし除算命令が 1 つがあります。3 つの命令はサイズ指定ができます。.B を指定すると(16 ビット)÷(8 ビット)=(8 ビット)...(余り 8 ビット)で演算し、.W を指定すると(32 ビット)÷(16 ビット)=(16 ビット)...(余り 16 ビット)で演算します。

また、除算命令では結果がオーバーフローするか“ 0 ”除算した場合、O フラグが変化します。除算命令の動作例を図 2.6.8 に示します。

表 2.6.9　除算命令

ニーモニック	記述形式		説明
DIV	DIV.B DIV.W	src src	符号付き除算命令 余りの符号は被除数の符号に一致
DIVX	DIVX.B DIVX.W	src src	符号付き除算命令 余りの符号は除数の符号に一致
DIVU	DIVU.B DIVU.W	src src	符号なし除算命令

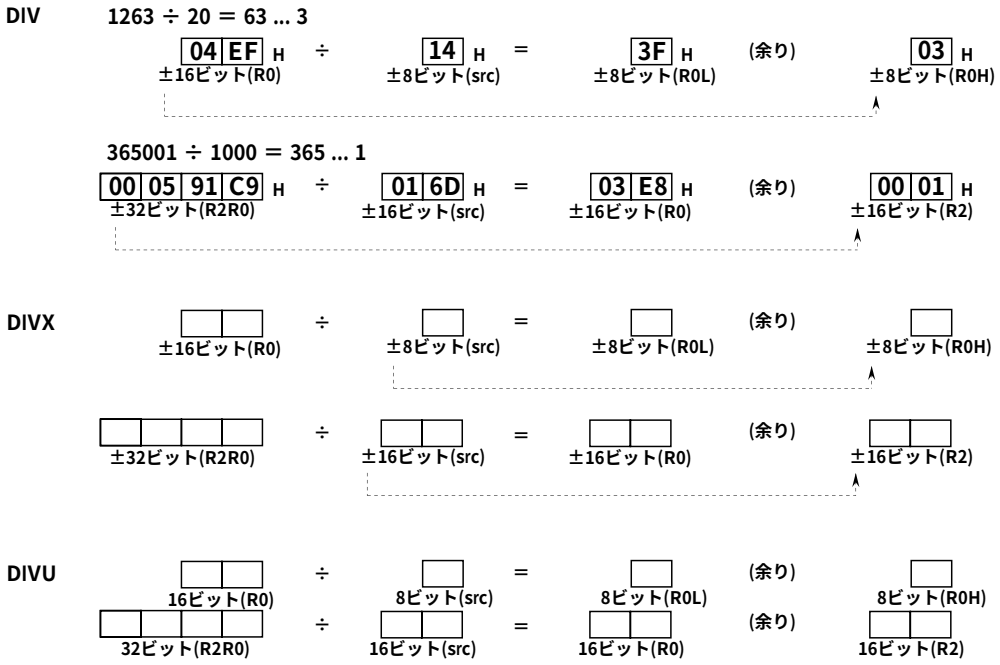


図 2.6.8　除算命令の動作例

DIV 命令と DIVX 命令の違いについて

DIV と DIVX は共に符号付きの除算命令ですが、2 つの命令の違いは余りの符号です。
表 2.6.10 に示すように、DIV 命令では余りの符号は被除数と同じで、DIVX 命令では除数と同じになります。

表 2.6.10　DIV 命令と DIVX 命令の違い

DIV	33 ÷ 4 = 8 ... 1	余りの符号は被除数と同じ
	33 ÷ (-4) = -8 ... 1	
	-33 ÷ 4 = -8 ... (-1)	
DIVX	33 ÷ 4 = 8 ... 1	余りの符号は除数と同じ
	33 ÷ (-4) = -9 ... (-3)	
	-33 ÷ 4 = -9 ... 3	

10 進加算命令

10 進加算命令にはキャリーなし 10 進加算命令とキャリー付き 10 進加算命令の 2 つがあります。
10 進加算命令を実行すると S フラグ、Z フラグ、C フラグが変化します。動作例を図 2.6.9 に示します。

表 2.6.11　10 進加算命令

ニーモニック	記述形式	説明
DADD	DADD .B src,dest DADD .W src,dest	10進でキャリーを含まずに加算
DADC	DADC .B src,dest DADC .W src,dest	10進でキャリーを含んで加算

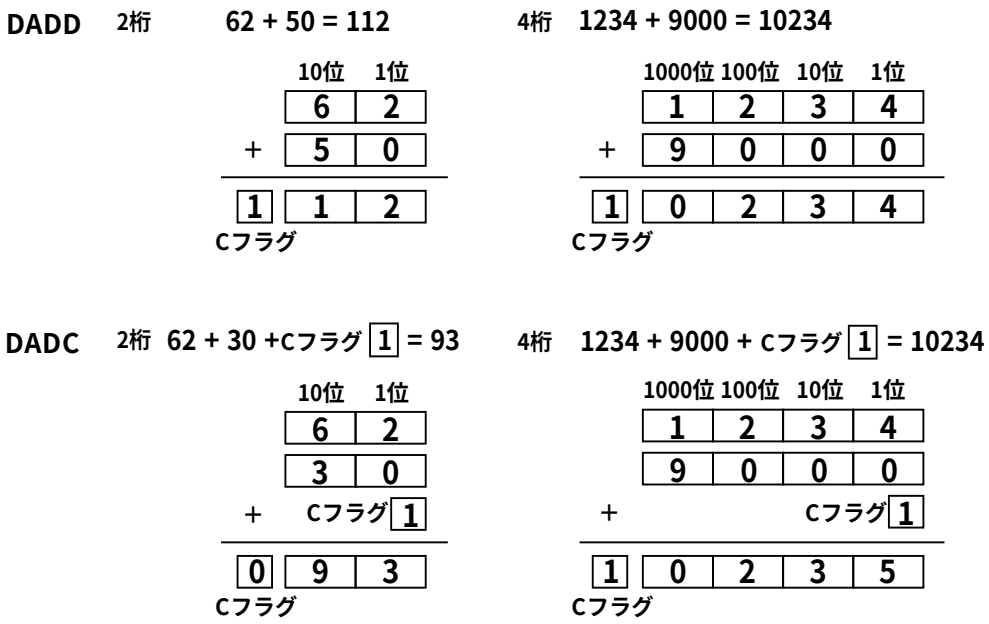


図 2.6.9　10 進加算命令の動作例

10 進減算命令

10 進減算命令にはボローなし 10 進減算命令とボロー付き 10 進減算命令の 2 つがあります。
10 進減算命令を実行すると S フラグ、Z フラグ、C フラグが変化します。動作例を図 2.6.10 に示します。

表 2.6.12　10 進減算命令

ニーモニック	記述形式	説明
DSUB	DSUB .B src,dest DSUB .W src,dest	ボロー含まずに10進で減算
DSBB	DSBB .B src,dest DSBB .W src,dest	ボローを含んで10進で減算

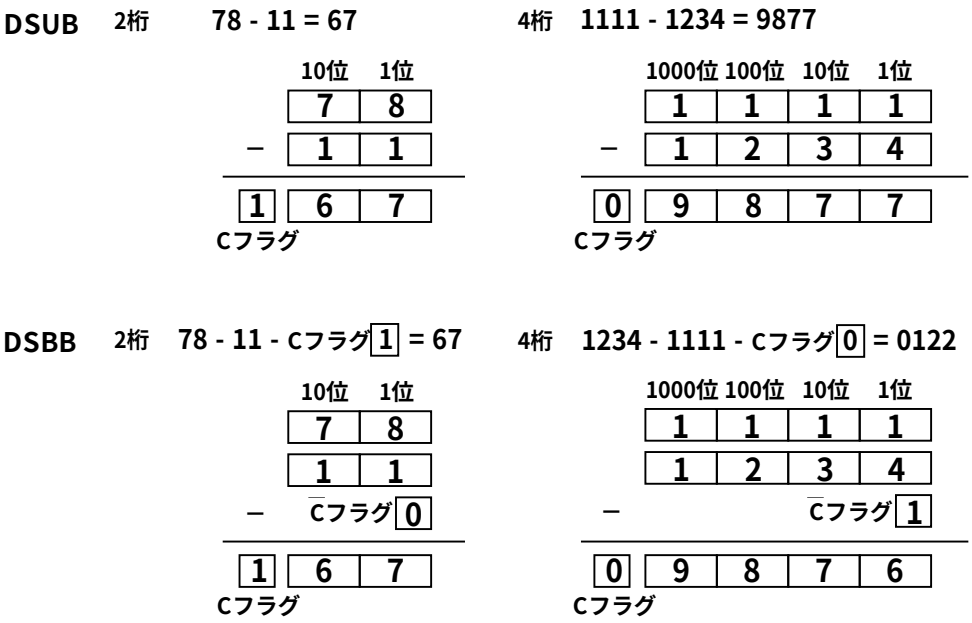


図 2.6.10　10 進減算命令の動作例

積和演算命令

積和演算命令は、積和演算を行い、演算中にオーバフローが発生したらオーバフローフラグが “ 1 ” にセットされます。図 2.6.9 に示すように、被乗数アドレス、乗数アドレス、積和回数は、各レジスタに設定します。積和演算命令の動作例を図 2.6.10 に示します。

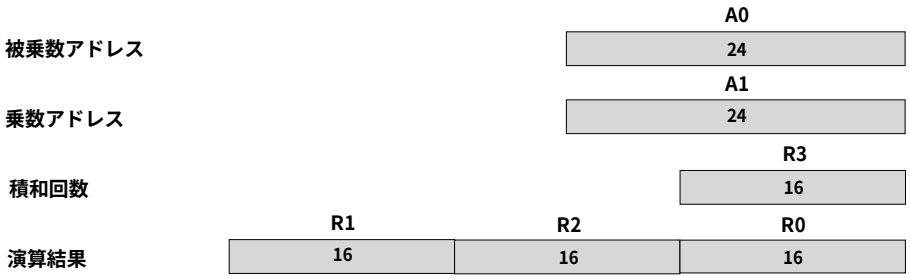


図 2.6.11　積和演算命令のレジスタ設定

表 2.6.13　積和演算命令

ニーモニック	記述形式	説明
RMPA	RMPA.B RPMA.W	A0を被乗数アドレス、A1を乗数アドレス、 R3を回数、とする積和演算

(注1)演算中にオーバフローが発生した場合、オーバフローフラグ(Oフラグ)を1にセットし演算を終了する。
(注2)演算中に割り込み要求があった場合は、演算途中の加算終了後積和回数を減算し、割り込みを受け付ける。

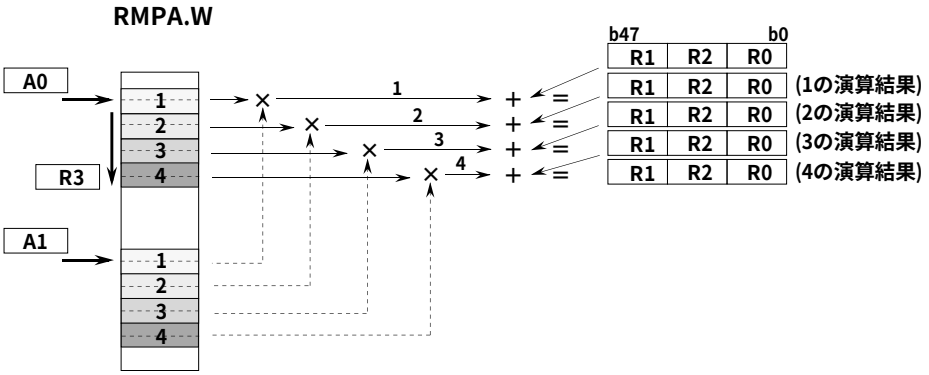


図 2.6.12　積和演算命令の動作例

MAX 命令、MIN 命令、CLIP 命令

M16C/80では、データの下限値、上限値の設定を1命令で行うことができる命令を3命令用意しています。CLIP命令はMAX 命令と MIN 命令を組み合わせた命令です。

これらの命令ではフラグは変化しません。動作例を図 2.6.13 に示します。

表 2.6.14　MAX 命令、MIN 命令、CLIP 命令

ニーモニック	記述形式	説明
MAX	MAX.B src,dest MAX.W src,dest	符号付きでsrcとdestを比較し、srcがdestより大きければsrcをdestに転送する
MIN	MIN.B src,dest MIN.W src,dest	符号付きでsrcとdestを比較し、srcがdestより小さければsrcをdestに転送する
CLIP	CLIP.B src1,,src2,dest CLIP.W src1,,src2,dest	符号付きでsrcとdestを比較し、src1がdestより大きければsrc1をdestに転送し、次にsrc2がdestより小さければsrc2をdestに転送するよって、src1≦dest≦src2ならば何もストアしない

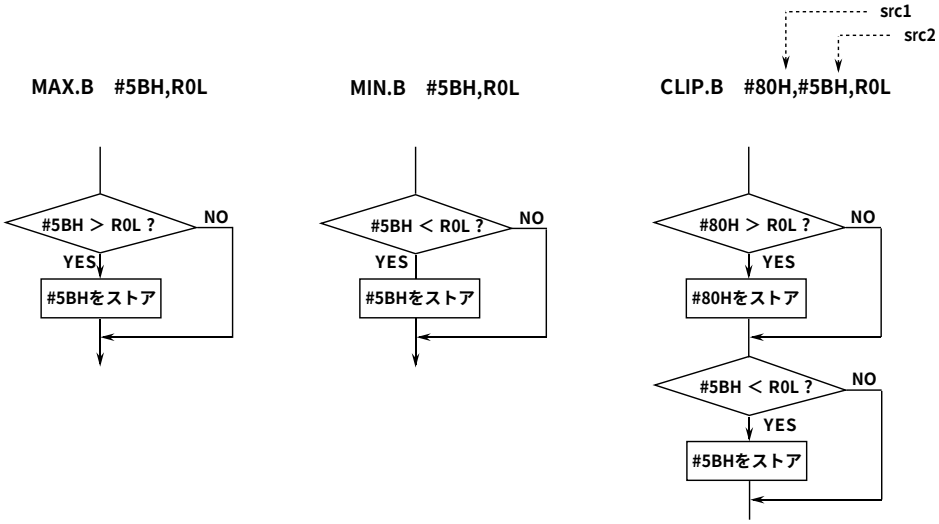


図 2.6.13　MAX 命令、MIN 命令、CLIP 命令の動作例

SC*cnd* 命令

M16C/80 では、フラグレジスタのフラグ内容により dest（1 ワード）に “ 1 ” または “ 0 ” を格納する命令を用意しています。動作例を図 2.6.14 に示します。

表 2.6.15　SC*cnd* 命令

ニーモニック	記述形式	説明
SC*cnd* dest	SC*GEU/C* dest	C=1ならば dest ← 1、それ以外は dest ← 0
	SC*LTU/NC* dest	C=0ならば dest ← 1、それ以外は dest ← 0
	SC*EQ/Z* dest	Z=1ならば dest ← 1、それ以外は dest ← 0
	SC*NE/NZ* dest	Z=0ならば dest ← 1、それ以外は dest ← 0
	SC*GTU* dest	C∧Z=1ならば dest ← 1、それ以外は dest ← 0
	SC*LEU* dest	C∧Z=0ならば dest ← 1、それ以外は dest ← 0
	SC*PZ* dest	S=0ならば dest ← 1、それ以外は dest ← 0
	SC*N* dest	S=1ならば dest ← 1、それ以外は dest ← 0
	SC*GE* dest	S∀O=0ならば dest ← 1、それ以外は dest ← 0
	SC*LE* dest	(S∀O)∨Z=1ならば dest ← 1、 それ以外は dest ← 0
	SC*GT* dest	(S∀O)∨Z=0ならば dest ← 1、 それ以外は dest ← 0
	SC*LT* dest	S∀O=1ならば dest ← 1、それ以外は dest ← 0
	SC*O* dest	O=1ならば dest ← 1、それ以外は dest ← 0
	SC*NO* dest	O=0ならば dest ← 1、それ以外は dest ← 0

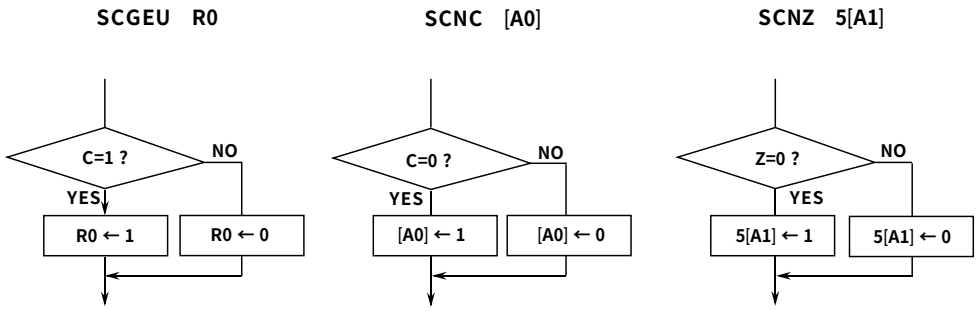


図 2.6.14　SC*cnd* 命令の動作例

2.6.5　分岐命令

分岐命令は 10 命令用意されています。この項では特長のある分岐命令を説明します。

無条件分岐命令

無条件で、label へ分岐する命令です。

分岐距離指定子は通常省略します。省略するとアセンブル時にアセンブラが分岐距離を判別し最適化します。無条件分岐命令の動作例を図 2.6.14 に示します。

表 2.6.16　無条件分岐命令

ニーモニック	記述形式	説明
JMP	JMP.S label JMP.B label JMP.W label JMP.A label	labelへ分岐する

分岐範囲: .S　+2～+9のPC相対アドレッシングによる分岐(オペランド:0バイト)
.B　-127～+128のPC相対アドレッシングによる分岐(オペランド:1バイト)
.W　-32767～+32768のPC相対アドレッシングによる分岐(オペランド:2バイト)
.A　24ビット絶対アドレッシングによる分岐(オペランド:3バイト)

```
JMP   LABEL1
```

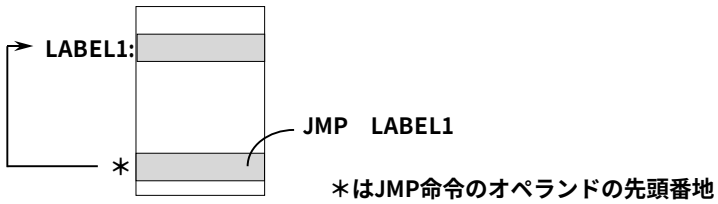


図 2.6.15　無条件分岐命令の動作例

間接分岐命令

src が示す番地に間接分岐する命令です。

分岐距離指定子に .W を指定した場合、JMPI 命令の先頭番地と src を符号付き加算した番地に分岐します。src がメモリのとき、必要なメモリ容量は 2 バイトです。分岐距離指定子に .A を指定した場合は、src が示す番地に分岐します。src がメモリのとき、必要なメモリ容量は 3 バイトです。この命令の分岐距離指定子は必ず指定してください。間接分岐命令の動作例を図 2.6.15 に示します。

表 2.6.17　間接分岐命令

ニーモニック	記述形式		説明
JMPI	JMPI.W JMPI.A	src src	srcが示す番地に間接分岐する

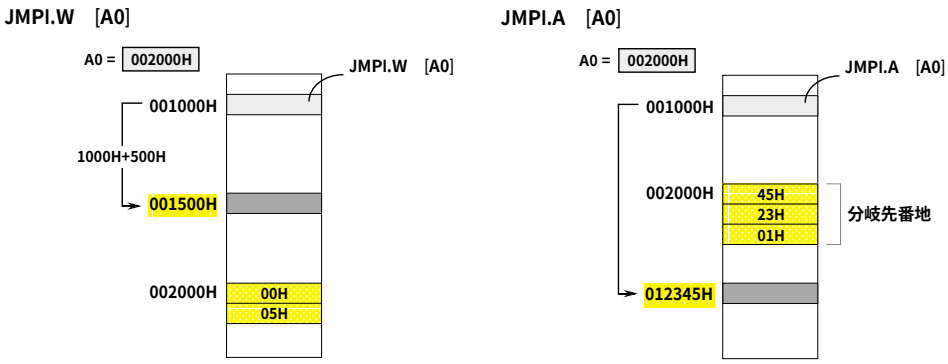


図 2.6.16　間接分岐命令の動作例

スペシャルページ分岐命令

スペシャルページベクタテーブルの各テーブルに設定している番地にFF0000Hを加算した番地へ分岐する命令です。分岐範囲はFF0000Hから FFFFFFH 番地です。スペシャルページ分岐命令はバイト数が2バイトで済む命令で、ROM の効率を上げることができます。

分岐先はスペシャルページ番号または label で指定します。スペシャルページ番号の前には“ # ”を、label の前には“ ¥ ”を付けてください。なお、label で指定した場合はアセンブラがスペシャルページ番号を算出します。スペシャルページ分岐命令の動作例を図 2.6.16 に示します。

表 2.6.18　スペシャルページ分岐命令

ニーモニック	記述形式
JMPS	JMPS　#スペシャルページ番号 JMPS　\スペシャルページベクタアドレス 18 ≦ スペシャルページ番号 ≦ 255

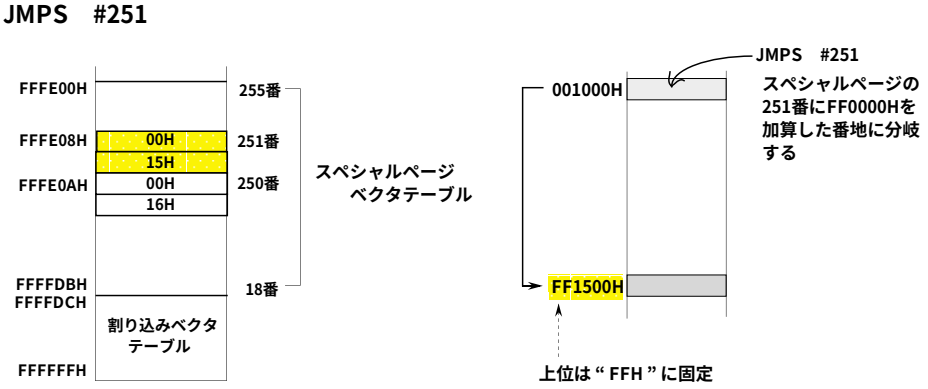


図 2.6.17　スペシャルページ分岐命令の動作例

条件分岐命令

条件分岐命令はフラグの状態を下記の条件で判定し、真ならば分岐し、偽ならば次の命令を実行します。条件分岐命令の動作例を図 2.6.18 に示します。

表 2.6.19　条件分岐命令

ニーモニック	記述形式	説明
JCnd	JCnd　label	条件が真ならばlabelへ分岐し、 条件が偽ならば次命令を実行する。

Cnd	真偽判定条件(14種類)	
GEU/C	C = 1	等しいまたは大きい/キャリーフラグが “ 1 ”
GTU	C = 1 & Z = 0	符号なしで大きい
EQ/Z	Z = 1	等しい/ゼロフラグが “ 1 ”
N	S = 1	負
LE	(Z = 1) ¦ (S = 1 & O = 0) ¦ (S = 0 & O = 1)	等しいまたは符号付きで小さい
O	O = 1	オーバフローフラグが “ 1 ”
GE	(S = 1 & O = 1) ¦ (S = 0 & O = 0)	等しいまたは符号付きで大きい
LTU/NC	C = 0	小さい/キャリーフラグが “ 0 ”
LEU	C = 0 ¦ Z = 1	等しいまたは小さい
NE/NZ	Z = 0	等しくない/ゼロフラフが “ 0 ”
PZ	S = 0	正またはゼロ
GT	(S = 1 & O = 1 & Z = 0) ¦ (S = 0 & O = 0 & Z = 0)	符号付きで大きい
NO	O = 0	オーバフローフラグが “ 0 ”
LT	(S = 1 & O = 0) ¦ (S = 0 & O = 1)	符号付きで小さい

分岐範囲:-127 ～ +128

```
JEQ   LABEL1
```

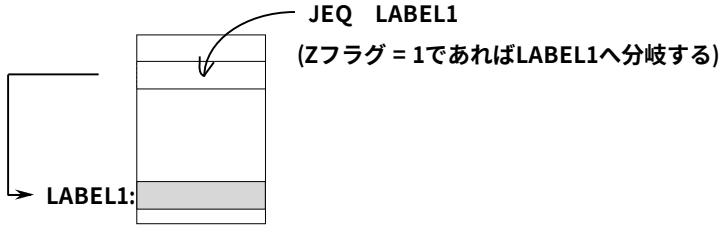


図 2.6.18　条件分岐命令の動作例

加算(減算)& 条件分岐命令

加算(減算)&条件分岐命令は、繰り返し処理の終了判断に便利な命令です。この命令で加算/減算する値は4ビット即値に限られます。ADJNZ 命令の場合は -8 ～ +7、SBJNZ 命令の場合は -7 ～ +8 になります。分岐できる範囲は、ADJNZ/SBJNZ 命令の先頭番地から -126 ～ +129 の領域です。加算(減算)& 条件分岐命令の動作例を図 2.6.19 に示します。

表 2.6.20　加算(減算)& 条件分岐命令

ニーモニック	記述形式	説明
ADJNZ	ADJNZ.B　#IMM4,dest,label ADJNZ.W　#IMM4,dest,label	destに即値を加算。 加算結果が0以外のときlabelへ分岐
SBJNZ	SBJNZ.B　#IMM4,dest,label SBJNZ.W　#IMM4,dest,label	destから即値を減算。 減算結果が0以外のときlabelへ分岐

(注1)#IMM4: 4ビット即値だけ (ADJNZの場合:-8～+7,SBJNZの場合:-7～+8)
(注2)分岐範囲 : PC相対アドレスで、ADJNZ/SBJNZ命令の先頭番地から-126～+129

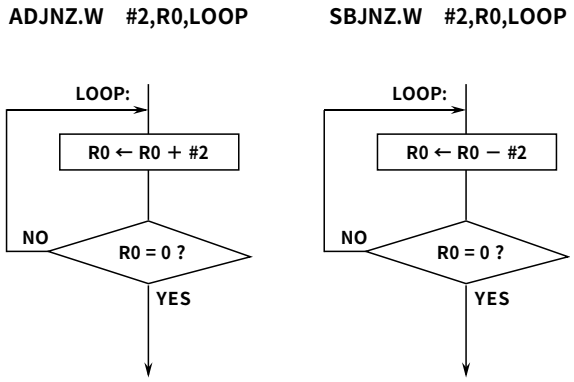


図 2.6.19　加算(減算)& 条件分岐命令の動作例

2.6.6 ビット命令

この項ではビット命令について説明します。

ビット論理演算命令

ビット論理演算命令は、レジスタ / メモリの指定ビットと、C フラグを論理演算し、結果を C フラグに格納する命令です。ビット論理演算命令の動作例を図 2.6.20 に示します。

表 2.6.21 ビット論理演算命令

ニーモニック	記述形式	説　明
BAND	BAND　src	Cフラグ ← src ∧ Cフラグ ; ビット論理積
BNAND	BNAND　src	Cフラグ ← src ∧ Cフラグ ; 反転ビット論理積
BNOR	BNOR　src	Cフラグ ← src ∨ Cフラグ ; 反転ビット論理和
BNXOR	BNXOR　src	Cフラグ ← src ∀ Cフラグ ; 反転ビット排他的論理和
BOR	BOR　src	Cフラグ ← src ∨ Cフラグ ; ビット論理和
BXOR	BXOR　src	Cフラグ ← src ∀ Cフラグ ; ビット排他的論理輪

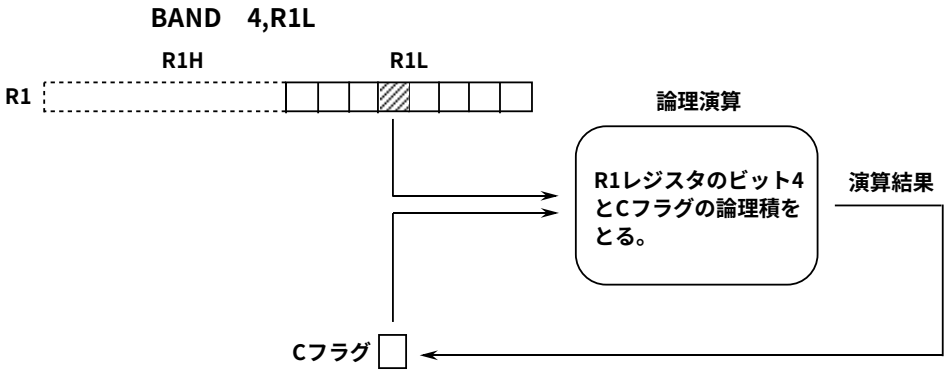


図 2.6.20 ビット論理演算命令の動作例

条件ビット転送命令

条件ビット転送命令は、条件付きのビット転送命令です。条件が真ならば“ 1 ”を転送し、偽ならば“ 0 ”を転送します。真偽判定はすべてフラグで行いますので、この命令の前にはフラグが変化する演算命令が必要です。条件ビット転送命令の動作例を図 2.6.21 に示します。

表 2.6.22　条件ビット転送命令

ニーモニック	記述形式	説明
BMCnd	BMCnd　dest BMCnd　C	条件が真ならば“ 1 ”を転送し、 条件が偽ならば“ 0 ”を転送する。

Cnd	真偽判定条件(14種類)	
GEU/C	C = 1	等しいまたは大きい/キャリーフラグが“ 1 ”
GTU	C = 1 & Z = 0	符号なしで大きい
EQ/Z	Z = 1	等しい/ゼロフラグが“ 1 ”
N	S = 1	負
LE	(Z = 1) ¦ (S = 1 & O = 0) ¦ (S = 0 & O = 1)	等しいまたは符号付きで小さい
O	O = 1	オーバフローフラグが“ 1 ”
GE	(S = 1 & O = 1) ¦ (S = 0 & O = 0)	等しいまたは符号付きで大きい
LTU/NC	C = 0	小さい/キャリーフラグが“ 0 ”
LEU	C = 0 ¦ Z = 1	等しいまたは小さい
NE/NZ	Z = 0	等しくない/ゼロフラフが“ 0 ”
PZ	S = 0	正またはゼロ
GT	(S = 1 & O = 1 & Z = 0) ¦ (S = 0 & O = 0 & Z = 0)	符号付きで大きい
NO	O = 0	オーバフローフラグが“ 0 ”
LT	(S = 1 & O = 0) ¦ (S = 0 & O = 1)	符号付きで小さい

BMGEU　3,1000H[SB]

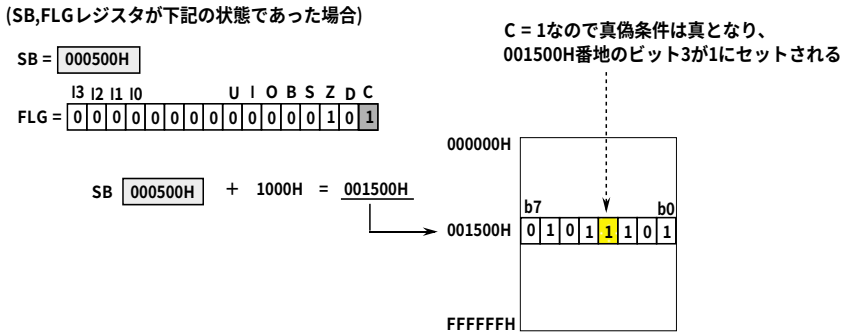


図 2.6.21　条件ビット転送命令の動作例

2.6.7　符号拡張命令

符号拡張命令は、符号ビットを拡張するビットに代入しビット長を長くする命令です。
この項では符号拡張命令について説明します。

符号拡張命令

符号拡張命令は、MSD（最上位ビット）に従った符号拡張命令と強制的に最上位ビットを“0”で拡張するゼロ拡張命令があります。符号拡張命令はサイズ指定子に.Bを指定すると16ビットに、.Wを指定すると32ビットに符号拡張を行います。

ゼロ拡張命令は16ビットにゼロ拡張します。符号拡張命令の動作例を図2.6.22に示します。

表2.6.23　符号拡張命令

ニーモニック	記述形式	説　　明
EXTS	EXTS.B　dest	destを16ビットに符号拡張します
	EXTS.W　dest	destを32ビットに符号拡張します
	EXTS.B　src,dest	srcを符号拡張しdestへ転送します
EXTZ	EXTZ　　src,dest	srcを16ビットにゼロ拡張しdestへ転送します

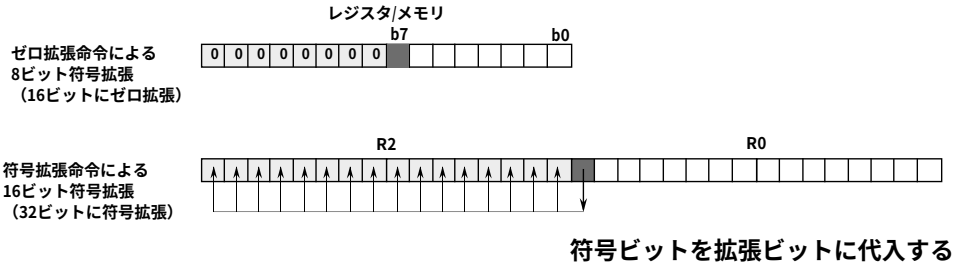


図2.6.22　符号拡張命令の動作例

2.6.8　インデックス命令

M16C/80にはC言語でプログラミングを行う場合、効率よく配列を参照できるようにインデックス命令を用意しています。インデックス命令によりアドレス演算なしに配列の要素を指定することができます。

この項ではインデックス命令について説明します。

インデックス命令

インデックス命令は、次の命令のアドレッシングをモデファイ（修飾）します。

インデックス命令のTypeとタイプごとの命令を表2.6.24に、またインデックス命令の動作例を図2.6.23に示します。

表 2.6.24　インデックス命令

Type	機能
B BD BS	次の命令のアドレッシングをバイト単位でモデファイします
W WD WS	次の命令のアドレッシングをワード単位でモデファイします
L LD LS	次の命令のアドレッシングをロングワード単位でモデファイします

ニーモニック	記述形式		説明
INDEXB	INDEXB.B INDEXB.W	src src	次に実行する命令のsrc,destが示すアドレスに、INDEXB命令のsrcの内容を符号なしで加算し実効アドレスとする
INDEXBD	INDEXBD.B INDEXBD.W	src src	次に実行する命令のdest（一部の命令ではsrc）が示すアドレスに、INDEXB命令のsrcの内容を符号なしで加算し実効アドレスとする
INDEXBS	INDEXBS.B INDEXBS.W	src src	次に実行する命令のsrcが示すアドレスに、INDEXB命令のsrcの内容を符号なしで加算し実効アドレスとする
INDEXW*a	INDEXB.B INDEXB.W	src src	次に実行する命令のsrc,destが示すアドレスに、INDEXW命令のsrcの内容を2倍した値を符号なしで加算し実効アドレスとする
INDEXWD*a	INDEXWD.B INDEXWD.W	src src	次に実行する命令のdest（一部の命令ではsrc）が示すアドレスに、INDEXB命令のsrcの内容を2倍した値を符号なしで加算し実効アドレスとする
INDEXWS*a	INDEXWS.B INDEXWS.W	src src	次に実行する命令のsrcが示すアドレスに、INDEXB命令のsrcの内容を2倍した値を符号なしで加算し実効アドレスとする
INDEXL*b	INDEXL.B INDEXL.W	src src	次に実行する命令のsrc,destが示すアドレスに、INDEXW命令のsrcの内容を4倍した値を符号なしで加算し実効アドレスとする
INDEXLD*b	INDEXLD.B INDEXLD.W	src src	次に実行する命令のdest（一部の命令ではsrc）が示すアドレスに、INDEXB命令のsrcの内容を4倍した値を符号なしで加算し実効アドレスとする
INDEXLS*b	INDEXLS.B INDEXLS.W	src src	次に実行する命令のsrcが示すアドレスに、INDEXB命令のsrcの内容を4倍した値を符号なしで加算し実効アドレスとする
BITINDEX	BITINDEX.B BITINDEX.W	src src	次に実行する命令のsrcまたはdestが示すアドレスのビット0から、BITINDEX命令のsrcで示したビット数だけ離れたビットを対象とします

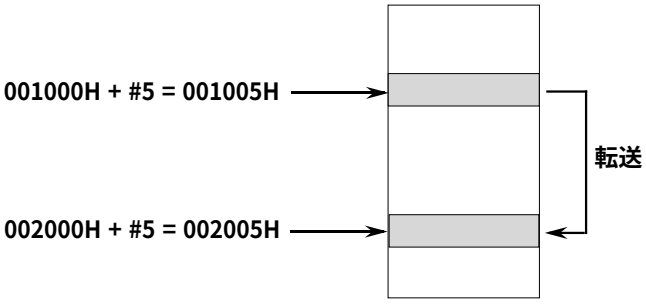
＊a　ワード単位の配列に対応します。

＊b　ロングワード単位の配列に対応します。

```
INDEXB.B        R1L
MOV.B           R0,002000H
```

R0 001000H

R1L #5




```
INDEXB.W        R1L
MOV.B           mem1,mem2
```

メモリ

R1L #5

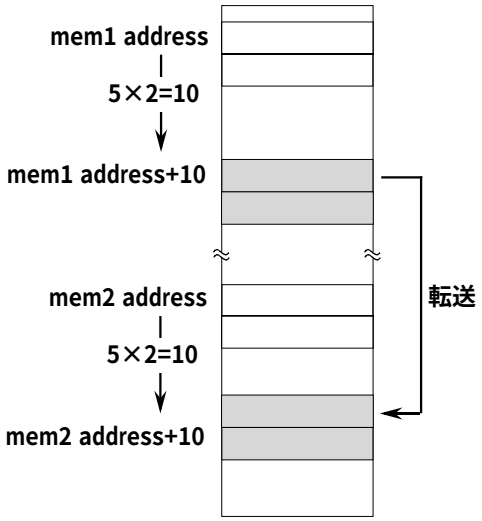


図 2.6.23　インデックス命令の動作例

2.6.9　高級言語、OS サポート命令

高級言語サポート命令はスタックフレームを構築/解放する命令です。また、OS サポート命令はタスクのコンテキストを待避/復帰する命令です。これらの命令は1命令で、高級言語に対応した複雑な処理やタスクのコンテキストの切り替えを実行します。

スタックフレームの構築

ENTER 命令はスタックフレームを構築する命令です。“ #IMM8 ”で自動変数領域のバイト数を設定します。スタックフレーム構築命令の動作例を図 2.624 に示します。

表 2.6.25　スタックフレーム構築命令

ニーモニック	記述形式	説明
ENTER	ENTER　#IMM8	スタックフレームを構築する

(注) #IMM8は、符号なし8ビット即値だけで 自動変数領域のサイズ
(バイト数)を示します。

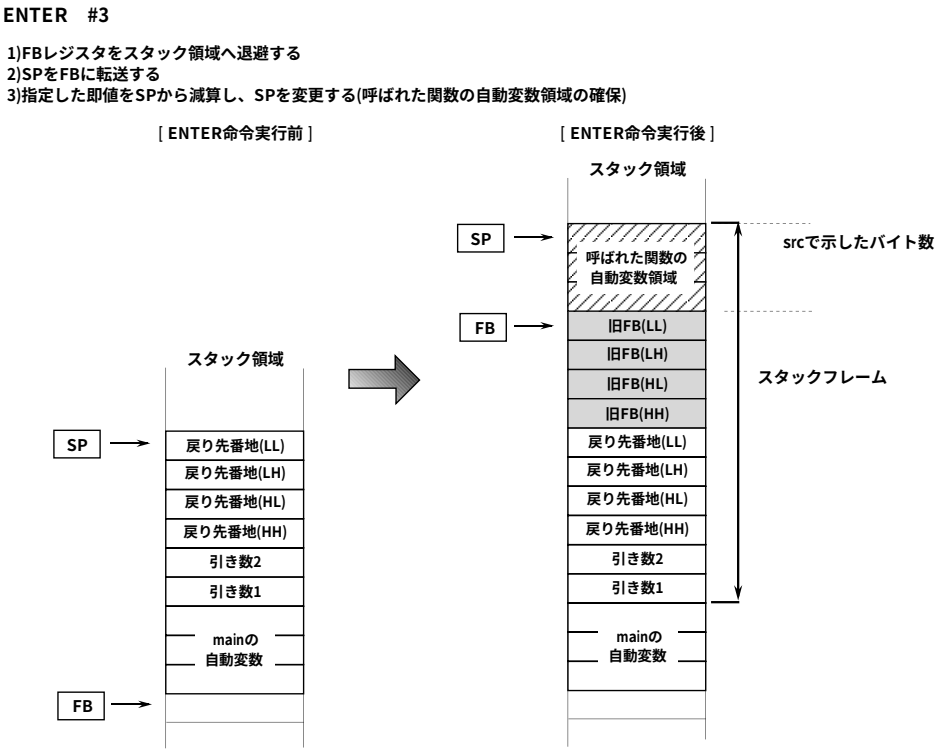


図 2.6.24　スタックフレーム構築命令の動作例

スタックフレームの解放

EXITD命令はスタックフレームの解放とサブルーチンからの復帰を同時に行います。スタックフレーム解放命令の動作例を図 2.6.25 に示します。

表 2.6.26　スタックフレーム解放命令

ニーモニック	記述形式	説明
EXITD	EXITD	スタックフレームを解放する

EXITD

1)FBレジスタをSPへ転送する
2)旧FBをスタック領域から復帰させる
3)サブルーチン（関数）からリターンする（RTS命令と同じ動作をする）

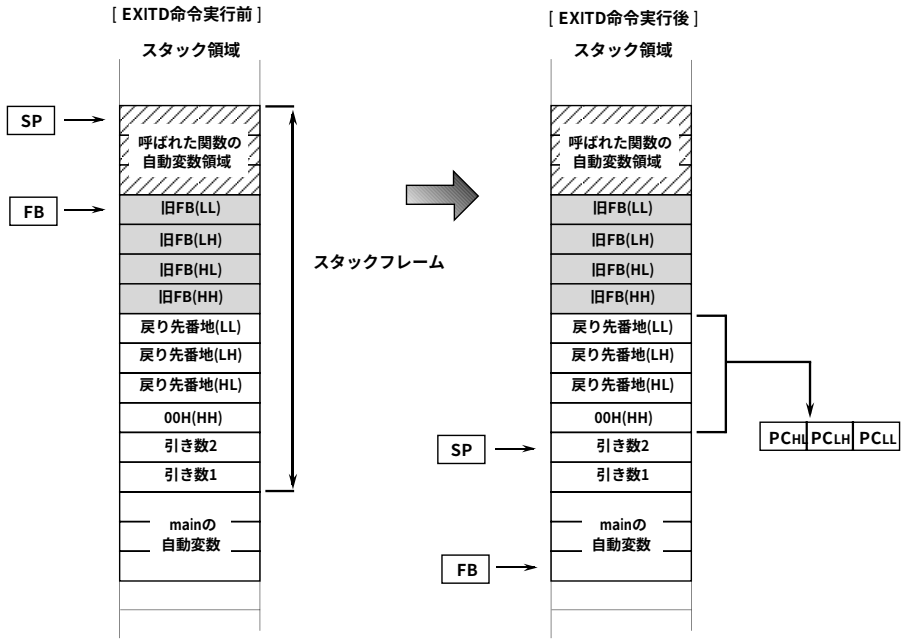


図 2.6.25　スタックフレーム解放命令の動作例

OS サポート命令

STCTX 命令はタスクのコンテキストを退避する命令です。LDCTX 命令はタスクのコンテキストを復帰する命令です。タスクのコンテキストテーブルを、図 2.6.26 に示します。コンテキストテーブルのレジスタ情報にはスタック領域にコンテキストとして退避するレジスタの種類を設定します。SP補正値には転送するレジスタのバイト数を設定します。OSサポート命令は 2 つの情報を使ってタスクのコンテキストをスタック領域に退避、復帰します。

表 2.6.27　OS サポート命令

ニーモニック	記述形式	説明
STCTX	STCTX　abs16,abs24	タスクのコンテキストを退避する
LDCTX	LDCTX　abs16,abs24	タスクのコンテキストを復帰する

(注1) abs16 : タスク番号(8ビット)を格納したメモリ番地
(注2) abs24 : コンテキストテーブルの先頭番地

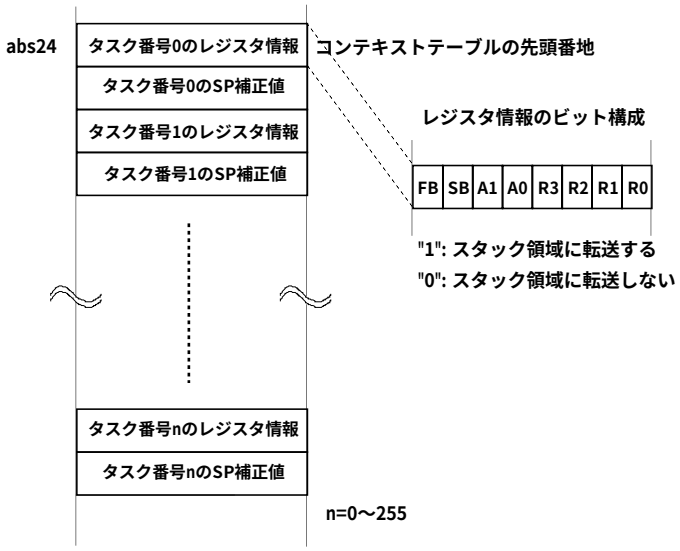


図 2.6.26　コンテキストテーブル

コンテキスト退避の動作（STCTX 命令）

動作1
abs16で示されたアドレスのメモリ内容を
タスク番号(8ビットデータ)として読み出す

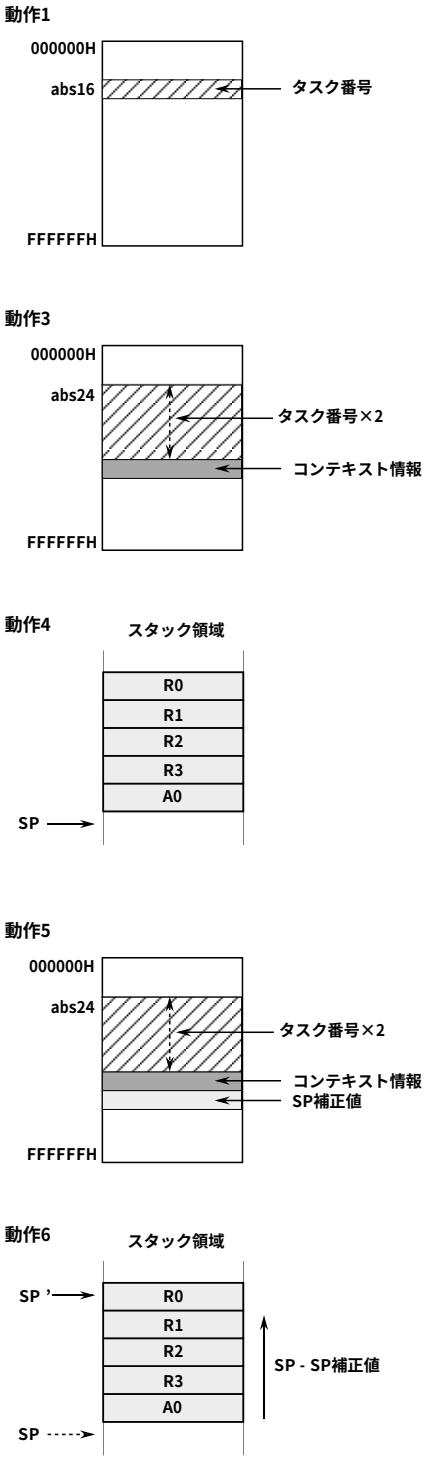
動作2
動作1で得られた値(タスク番号)を2倍し、
abs24(コンテキストテーブルの先頭番地)を加算する
(タスク番号)×2 + abs24

動作3
動作2の結果で示されたアドレスのメモリ内容を
コンテキスト情報(8ビットデータ)として読み出す

動作4
コンテキスト情報により指定されたレジスタを
スタック領域へ退避する

動作5
コンテキスト情報の次の番地(+1した番地)のアドレス
の内容をSP補正値(8ビットデータ)として読み出す

動作6
SPからSP補正値を減算し、SPを変更する

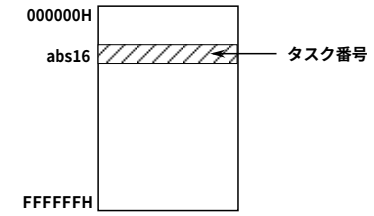


コンテキスト復帰の動作（LDCTX 命令）

動作1
abs16で示されたアドレスのメモリ内容を
タスク番号(8ビットデータ)として読み出す

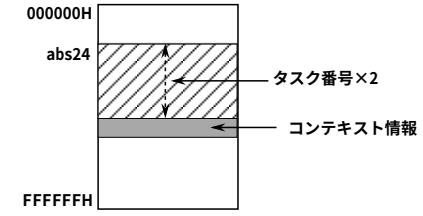
動作2
動作1で得られた値(タスク番号)を2倍し、
abs24(コンテキストテーブルの先頭番地)を加算する
(タスク番号)×2 + abs24

動作1



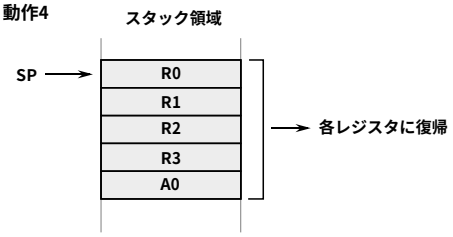
動作3
動作2の結果で示されたアドレスのメモリ内容を
コンテキスト情報(8ビットデータ)として読み出す

動作3



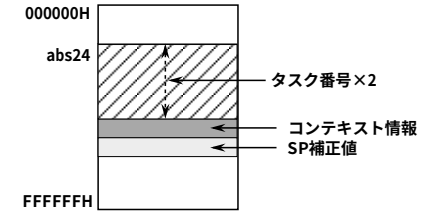
動作4
コンテキスト情報により指定されたレジスタを
スタック領域から復帰する
(この時点ではSPレジスタの値は変更しない)

動作4



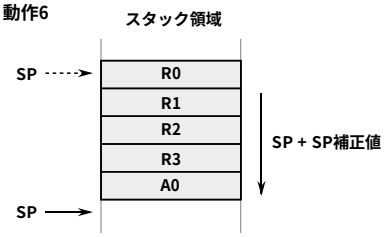
動作5
コンテキスト情報の次の番地(+1した番地)の内容を
SP補正値(8ビットデータ)として読み出す

動作5



動作6
SPにSP補正値を加算し、SPを変更する

動作6



2.7　割り込みの概要

割り込み要因の種類と割り込み要求を受け付けてから割り込みルーチンを実行するまでの内部処理（割り込みシーケンス）について説明します。各割り込みの使い方、設定方法については第 4 章をご参照ください。

2.7.1　割り込み要因とベクトル番地

この項では M16C/80 グループの割り込み要因について説明します。

M16C/80 グループの割り込み要因

M16C/80 グループの割り込み要因を図 2.7.1 に示します。

ハードウェア割り込みにはリセット、$\overline{\text{NMI}}$ など 7 種類の特殊割り込みと、タイマ、外部端子など内蔵する周辺機能に依存する周辺 I/O 割り込み(注)があります。特殊割り込みはノンマスカブル割り込みです。周辺 I/O 割り込みはマスカブル割り込みです。マスカブル割り込みの許可および禁止は、割り込み許可フラグ(I フラグ)、割り込み優先レベル選択ビット、およびプロセッサ割り込み優先レベル(IPL)によって行います。

ソフトウェア割り込みはソフトウェア割り込み命令の実行によって割り込み要求を発生します。ソフトウェア割り込みには INT 命令割り込み、BRK 命令割り込み、BRK2 命令割り込み、オーバフロー割り込み、未定義命令割り込みの 5 種類があります。ソフトウェア割り込みはノンマスカブル割り込みです。

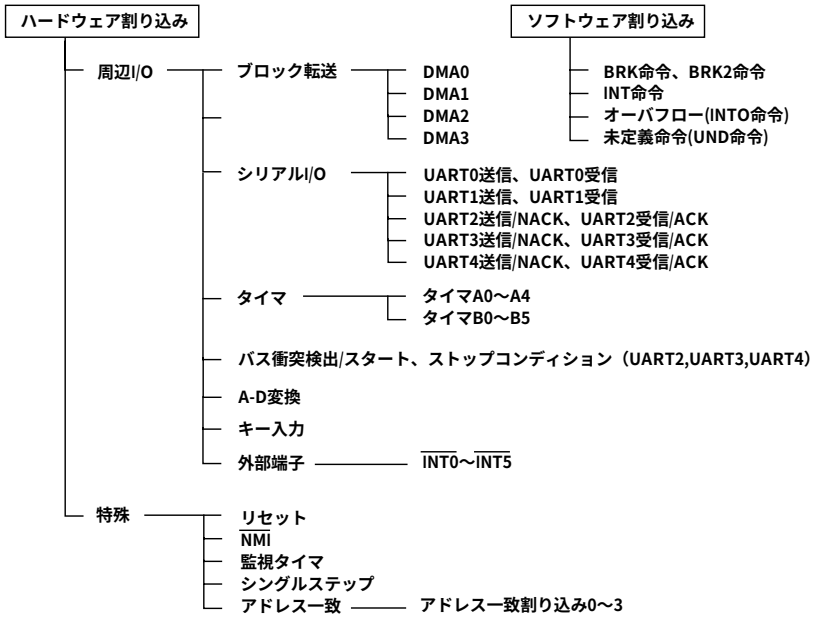


図 2.7.1　M16C/80 グループの割り込み要因

(注)周辺機能は使用する品種によって異なります。周辺割り込みについての詳細はデータシートおよびユーザーズマニュアルを参照してください。

ベクトル番地

ソフトウェア割り込みと特殊割り込みのベクトル番地を図 2.7.2、ハードウェア割り込みのベクトル番地を図 2.7.3 に示します。各割り込みを使用するときには、これらのベクトル番地に割り込みルーチンの先頭アドレスを設定してください。

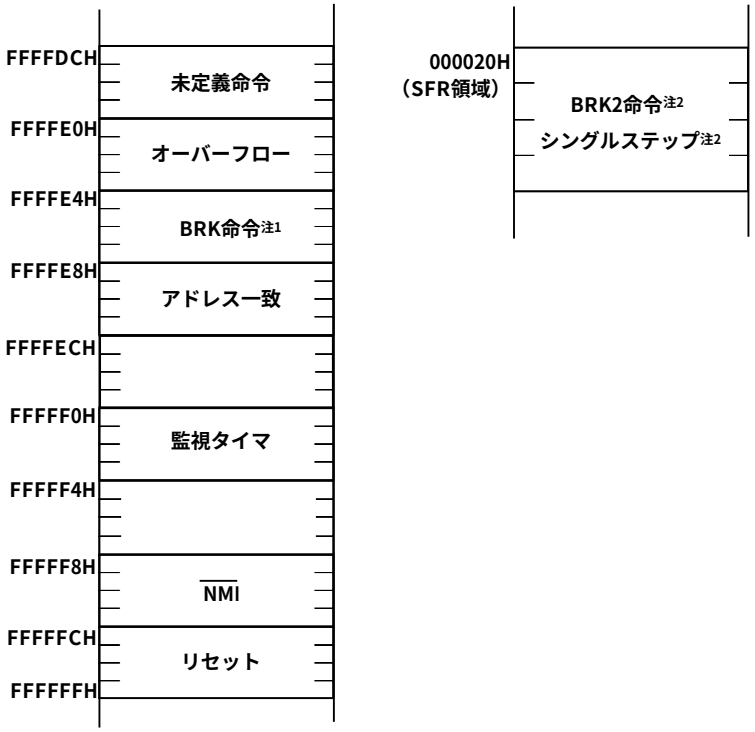


図 2.7.2　ソフトウェア割り込みと特殊割り込みのベクトル番地

注 1　ベクタの内容が全て“ FFH ”の場合は、可変ベクタテーブルのソフトウェア割り込み番号 0 のベクタが示す番地に分岐します。
注 2　ユーザ使用禁止。

ソフトウエア割り込み番号	ベクタテーブル番地 アドレス(L)～アドレス(H)	割り込み要因	備　考
ソフトウエア割り込み番号0	+0～+3(注1)	BRK命令	Iフラグによるマスク不可
ソフトウエア割り込み番号8	+32～+35(注1)	DMA0	
ソフトウエア割り込み番号9	+36～+39(注1)	DMA1	
ソフトウエア割り込み番号10	+40～+43(注1)	DMA2	
ソフトウエア割り込み番号11	+44～+47(注1)	DMA3	
ソフトウエア割り込み番号12	+48～+51(注1)	タイマA0	
ソフトウエア割り込み番号13	+52～+55(注1)	タイマA1	
ソフトウエア割り込み番号14	+56～+59(注1)	タイマA2	
ソフトウエア割り込み番号15	+60～+63(注1)	タイマA3	
ソフトウエア割り込み番号16	+64～+67(注1)	タイマA4	
ソフトウエア割り込み番号17	+68～+71(注1)	UART0送信	
ソフトウエア割り込み番号18	+72～+75(注1)	UART0受信	
ソフトウエア割り込み番号19	+76～+79(注1)	UART1送信	
ソフトウエア割り込み番号20	+80～+83(注1)	UART1受信	
ソフトウエア割り込み番号21	+84～+87(注1)	タイマB0	
ソフトウエア割り込み番号22	+88～+91(注1)	タイマB1	
ソフトウエア割り込み番号23	+92～+95(注1)	タイマB2	
ソフトウエア割り込み番号24	+96～+99(注1)	タイマB3	
ソフトウエア割り込み番号25	+100～+103(注1)	タイマB4	
ソフトウエア割り込み番号26	+104～+107(注1)	$\overline{\text{INT5}}$	
ソフトウエア割り込み番号27	+108～+111(注1)	$\overline{\text{INT4}}$	
ソフトウエア割り込み番号28	+112～+115(注1)	$\overline{\text{INT3}}$	
ソフトウエア割り込み番号29	+116～+119(注1)	$\overline{\text{INT2}}$	
ソフトウエア割り込み番号30	+120～+123(注1)	$\overline{\text{INT1}}$	
ソフトウエア割り込み番号31	+124～+127(注1)	$\overline{\text{INT0}}$	
ソフトウエア割り込み番号32	+128～+131(注1)	タイマB5	
ソフトウエア割り込み番号33	+132～+135(注1)	UART2送信/NACK　(注2)	
ソフトウエア割り込み番号34	+136～+139(注1)	UART2受信/ACK　(注2)	
ソフトウエア割り込み番号35	+140～+143(注1)	UART3送信/NACK　(注2)	
ソフトウエア割り込み番号36	+144～+147(注1)	UART3受信/ACK　(注2)	
ソフトウエア割り込み番号37	+148～+151(注1)	UART4送信/NACK　(注2)	
ソフトウエア割り込み番号38	+152～+155(注1)	UART4受信/ACK　(注2)	
ソフトウエア割り込み番号39	+156～+159(注1)	バス衝突検出、スタート/ストップコンディション検出(UART2)　(注2)	
ソフトウエア割り込み番号40	+160～+163(注1)	バス衝突検出、スタート/ストップコンディション検出(UART3)　(注2)	
ソフトウエア割り込み番号41	+164～+167(注1)	バス衝突検出、スタート/ストップコンディション検出(UART4)　(注2)	
ソフトウエア割り込み番号42	+168～+171(注1)	A-D	
ソフトウエア割り込み番号43	+172～+175(注1)	キー入力割り込み	
ソフトウエア割り込み番号44 ～ ソフトウエア割り込み番号63	+176～+179(注1) ～ +252～+255(注1)	ソフトウエア割り込み	Iフラグによるマスク不可

注1.　割り込みテーブルレジスタ(INTB)が示すアドレスからの相対アドレスです。
注2.　IICモード選択時にNACK/ACK、スタート/ストップコンディション検出割り込みが選択されます。
　　SS端子選択時障害エラー割り込みが選択されます。

図 2.7.3　ハードウェア割り込みのベクトル番地

2.7.2　可変ベクタテーブル

可変ベクタテーブルは、割り込みテーブルレジスタ(INTB)が指すアドレスを先頭アドレスとする、256 バイトのベクタテーブルです（図 2.7.3）。ベクタテーブルは SFR 領域、拡張予約領域以外ならどの領域でも配置が可能です。

可変ベクタテーブル

1 ベクタは 4 バイトで構成されており、各ベクタごとに 0～63 までのソフトウェア割り込み番号が割り付けられています。INT 命令とソフトウェア割り込み番号を使用することにより、擬似的に周辺 I/O の割り込みルーチンを実行することができます。図 2.7.4 に可変ベクタテーブルの配置状態を示します。

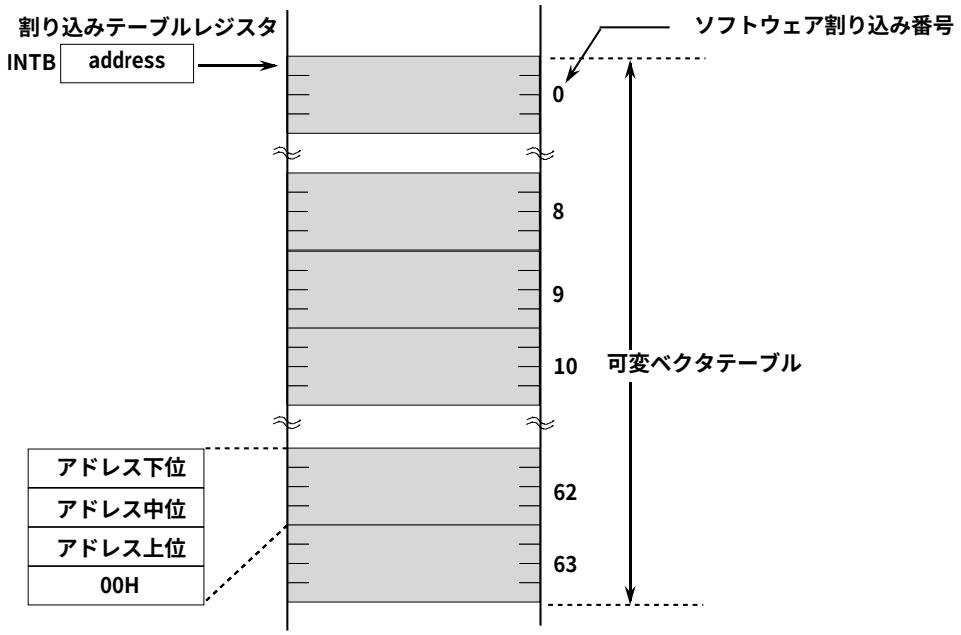


図 2.7.4　可変ベクタテーブルの配置状態

2.7.3 割り込み発生条件と割り込み制御レジスタのビット構成

この項では、割り込みが受け付けられる条件と割り込み制御レジスタのビット構成について説明します。

割り込み発生条件

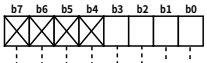
割り込みは以下の 3 つの条件を全て満たすときに要求を受け付けます。

- 割り込み許可フラグ（I フラグ）= 1 （割り込み許可）
- プロセッサ割り込みレベル（IPL）＜ 各割り込みの割り込み優先レベル
- 割り込み要求ビット（割り込み制御レジスタのビット 3）= 1

割り込み制御レジスタのビット構成

各割り込みが持つ割り込み制御レジスタを図 2.7.5 に示します。

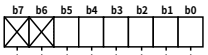
割り込み制御レジスタ



シンボル	アドレス	リセット時
ADIC	0073₁₆番地	XXXXX000₂
BCNiIC(i=2～4)	008F₁₆, 0071₁₆, 0091₁₆番地	XXXXX000₂
DMiIC(i=0～3)	0068₁₆, 0088₁₆, 006A₁₆, 008A₁₆番地	XXXXX000₂
KUPIC	0093₁₆番地	XXXXX000₂
TAiIC(i=0～4)	006C₁₆, 008C₁₆, 006E₁₆, 008E₁₆, 0070₁₆番地	XXXXX000₂
TBiIC(i=0～5)	0094₁₆, 0076₁₆, 0096₁₆, 0078₁₆, 0098₁₆, 0069₁₆番地	XXXXX000₂
SiTIC(i=0～4)	0090₁₆, 0092₁₆, 0089₁₆, 008B₁₆, 008D₁₆番地	XXXXX000₂
SiRIC(i=0～4)	0072₁₆, 0074₁₆, 006B₁₆, 006D₁₆, 006F₁₆番地	XXXXX000₂

ビットシンボル	ビット名	機　能	R	W
ILVL0	割り込み優先レベル選択ビット	b2b1b0 0 0 0：レベル0（割り込み禁止） 0 0 1：レベル1 0 1 0：レベル2 0 1 1：レベル3 1 0 0：レベル4 1 0 1：レベル5 1 1 0：レベル6 1 1 1：レベル7	○	○
ILVL1			○	○
ILVL2			○	○
IR	割り込み要求ビット	0：割り込み要求なし 1：割り込み要求あり	○	○ (注1)
何も配置されていない。 書き込む場合、“0”を書き込んでください。読み出した場合、その値は不定。			—	—

注1.　“0”だけ書き込み可(“1”を書き込まないでください)。



シンボル	アドレス	リセット時
INTiIC(i=0～5)	009E₁₆, 007E₁₆, 009C₁₆, 007C₁₆, 009A₁₆, 007A₁₆番地	XX00X000₂

ビットシンボル	ビット名	機　能	R	W
ILVL0	割り込み優先レベル選択ビット	b2b1b0 0 0 0：レベル0（割り込み禁止）(注2) 0 0 1：レベル1 0 1 0：レベル2 0 1 1：レベル3 1 0 0：レベル4 1 0 1：レベル5 1 1 0：レベル6 1 1 1：レベル7	○	○
ILVL1			○	○
ILVL2			○	○
IR	割り込み要求ビット	0：割り込み要求なし 1：割り込み要求あり	○	○ (注1)
POL	極性切り替えビット	0：立ち下がりエッジまたは“L”レベルを選択 1：立ち上がりエッジまたは“H”レベルを選択	○	○
LVS	レベルセンス/エッジセンス切り替えビット	0：エッジセンス 1：レベルセンス (注3)	○	○
何も配置されていない。 書き込む場合、“0”を書き込んでください。読み出した場合、その値は不定。			—	—

注1.　“0”だけ書き込み可(“1”を書き込まないでください)。
注2.　マイクロプロセッサモード、メモリ拡張モードで、INT3～INT5がデータバスとなる場合、INT3IC、INT4IC、INT5ICは割り込み禁止に設定してください。
注3.　レベルセンスを選択した場合、割り込み要因選択レジスタ(031F₁₆番地)の対応するビットを片エッジ(“0”)に設定してください。

図 2.7.5　割り込み制御レジスタのビット構成

注1　シンボルは M16C/80 グループのものなので、使用する品種により異なります。

2.7.4 割り込みの受け付けタイミングとシーケンス

この項では、割り込みの受け付けタイミングと割り込みシーケンスについて説明します。

割り込みの受け付けタイミング

命令実行中に割り込み要求が発生すると、その命令の実行終了後に優先順位が判定され、次のサイクルから割り込みシーケンスに移ります。この場合の割り込み受け付けタイミングを図2.7.6に示します。ただし、ストリング命令(SCMPU、SIN、SMOVB、SMOVF、SMOVU、SSTR、SOUT)と積和演算命令(RMPA)の実行中に割り込み要求が発生すると、命令の動作を一時中断し割り込みシーケンスに移ります。この場合の割り込み受け付けタイミングを図 2.7.7 に示します。

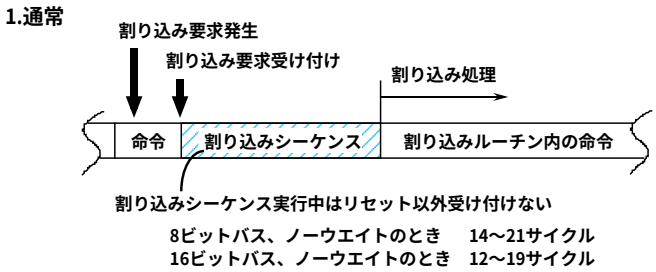


図 2.7.6　割り込み受け付けタイミング 1

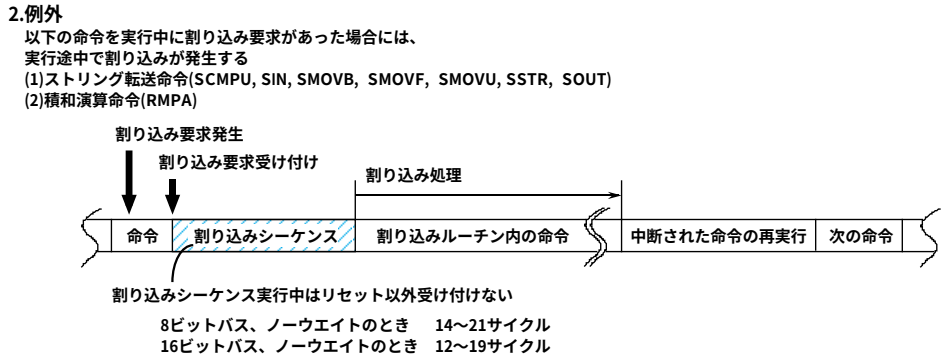


図 2.7.7　割り込み受け付けタイミング 2

割り込みシーケンス

割り込み要求が受け付けられてから割り込みルーチンが実行されるまでの、割り込みシーケンスについて説明します。

(1) 000000H番地(高速割り込みの場合、000002H番地)を読むことで、CPUは割り込み情報(割り込み番号、割り込み要求レベル)を獲得する。その後、該当する割り込みの要求ビットが“0”になる。
(2) 割り込みシーケンス直前のフラグレジスタ(FLG)の内容をCPU内部の一時レジスタ*1に退避する。
(3) 割り込み許可フラグ(Iフラグ)、デバッグフラグ(Dフラグ)、およびスタックポインタ指定フラグ(Uフラグ)を“0”にする(ただしUフラグは、ソフトウエア割り込み番号32～63のINT命令を実行した場合は変化しません)。
これらの動作により、
- スタックポインタは強制的に割り込みスタックポインタ(ISP)になる
（ただし、ソフトウェア割り込み番号32～63のINT命令を実行した場合は、割り込み発生時のスタックポインタ（ISPまたはUSP）を使用する）
- 多重割り込みが禁止となる
- シングルステップ割り込みが禁止となる

(4) CPU内部の一時レジスタ*1の内容、プログラムカウンタ(PC)の内容をスタック領域に退避する。高速割り込みの場合は、フラグ退避レジスタ(SVF)、PC退避レジスタ(SVP)に退避する。
(6) プロセッサ割り込み優先レベル(IPL)に、受け付けた割り込みの割り込み優先レベルを設定する。

割り込みシーケンス終了後は、割り込みルーチンの先頭番地から命令を実行します。
*1　ユーザは使用できません。

2.7.5　割り込み優先順位

この項では、割り込み優先順位について説明します。

割り込み優先順位

同一サンプリング時点（割り込み要求があるかどうかを調べるタイミング）で2つ以上の割り込み要求が存在した場合には、優先順位の高い割り込みが受け付けられます。マスカブル割り込み（周辺I/O割り込み）の優先順位は、割り込み優先レベル選択ビットによって任意の優先順位を設定することができます。ただし、割り込み優先レベルが同じ設定値の場合は、一番最初に要求がきた割り込みを最初に受け付け、それ以降はハードウェアで設定されている優先度[*1]の高い割り込みが受け付けられます。

リセット（リセットは優先順位の一番高い割り込みとして扱われます）、監視タイマ割り込みなど、ノンマスカブル割り込みの優先順位はハードウェアで設定されています。図2.7.8にハードウェアで設定されている割り込み優先順位を示します。

ソフトウェア割り込みは割り込み優先順位の影響を受けません。命令を実行すると必ず割り込みルーチンへ分岐します。

*1　品種により異なりますので、データシート、ユーザーズマニュアルなどを参照してください。

図 2.7.8　ハードウェアで設定されている割り込み優先順位

第 3 章
アセンブラの機能

3.1 AS308 システムの概要

AS308 システムは、M16C/80 シリーズシングルチップマイクロコンピュータ制御プログラムの開発を、アセンブリ言語レベルで支援するソフトウェアシステムです。AS308システムにはアセンブラの他に、リンケージエディタ、ロードモジュールコンバータが付属しています。

この節では AS308 の概要を説明します。

機能

●リロケータブルアセンブル機能
●最適化コード生成機能
●マクロ機能
●高級言語のソースレベルデバッグ機能
●各種ファイル生成機能
● IEEE-695 フォーマット (注1) のファイル生成機能

構成

AS308 システムは下記のプログラムで構成しています。

●アセンブラドライバ(as308)
マクロプロセッサ、アセンブラプロセッサを起動する実行ファイルです。アセンブラドライバは、複数のアセンブリソースファイルを処理することができます。

●マクロプロセッサ(mac308)
アセンブリソース中のマクロ指示命令およびアセンブラプロセッサのための前処理を行い、中間ファイルを生成します。中間ファイルはアセンブラプロセッサの処理終了後に消去されます。

●アセンブラプロセッサ(asp308)
マクロプロセッサが生成した中間ファイルをリロケータブルモジュールファイルに変換します。

●リンケージエディタ(ln308)
アセンブラプロセッサの生成したリロケータブルモジュールファイルをリンクし、アブソリュートモジュールファイルを生成します。

●ロードモジュールコンバータ(lmc308) (注2)
リンケージエディタの生成したアブソリュートモジュールファイルを ROM 化可能な機械語ファイルに変換します。

●ライブラリアン(lb308)
リロケータブルモジュールファイルを読み込み、ライブラリファイルを生成、管理します。

●クロスリファレンサ(xrf308)
ユーザーの作成したアセンブリソースファイル中の各種シンボルおよびラベルの定義情報を格納したクロスリファレンスファイルを生成します。

●アブソリュートリスタ(abs308)
アブソリュートモジュールファイルのアドレス情報を基に、プリントアウト可能なアブソリュートリストファイルを生成します。

(注 1)IEEE(Institute of Electrical and Electronics Engineers:アメリカ電気電子技術者協会)
(注 2)ロードモジュールコンバータは M16C/80 シリーズの ROM に書き込み可能なフォーマットへ変換するプログラムです。

AS308 システムの処理概要

AS308 システムのアセンブル処理の概要を図 3.1.1 に示します。

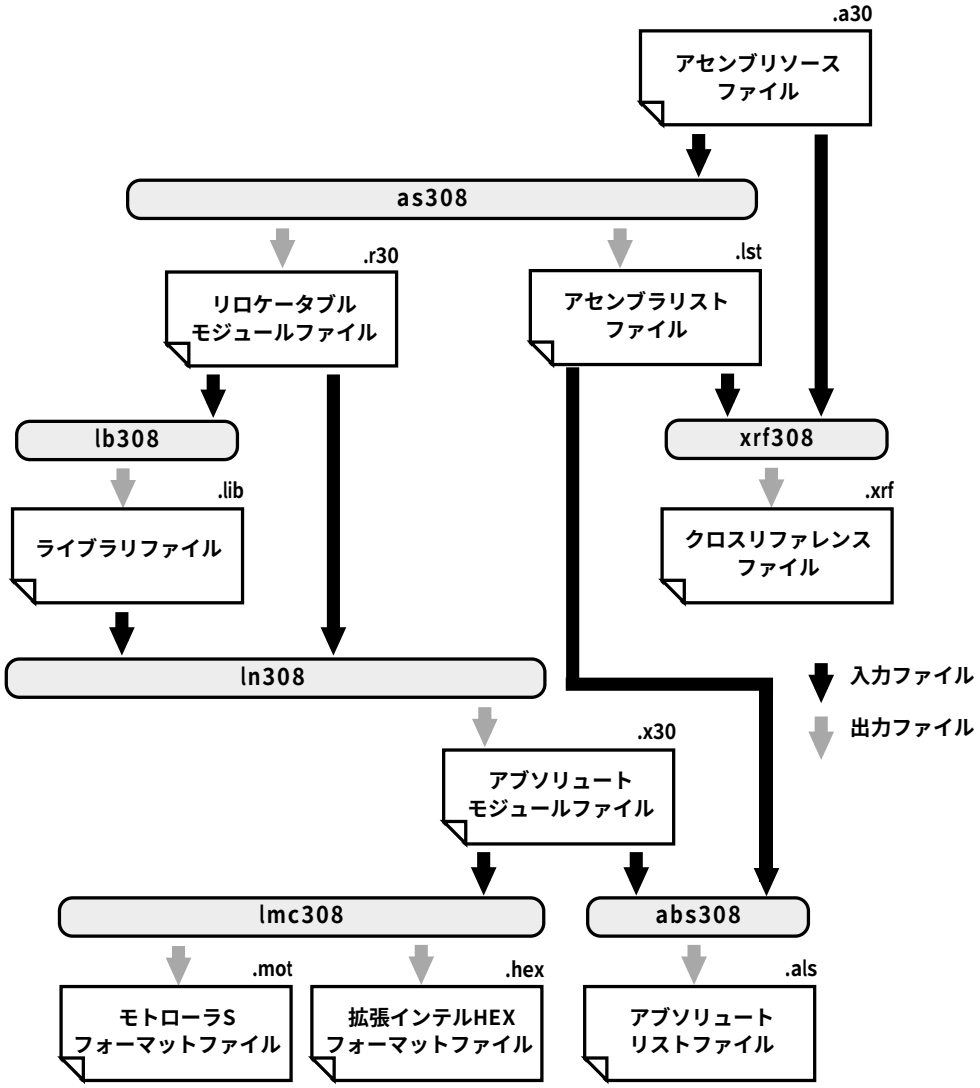


図 3.1.1　AS308 処理概要

AS308 システムの入出力ファイル

AS308 システムで扱うファイルを入力ファイルと、出力ファイルに分けて次の表に示します。ファイル名は任意に付けることができます。ただし、ファイル名の拡張子部分については、拡張子を省略すると、AS308 システムがデフォルトのファイル拡張子を付加します。この拡張子を表の( )内に記述します。

表 3.1.1 入出力ファイル一覧

プログラム名	入力ファイル名(拡張子)	出力ファイル名(拡張子)
アセンブラ as308	ソースファイル (.a30) インクルードファイル (.inc)	リロケータブルモジュールファイル (.r30) アセンブラリストファイル (.lst) アセンブラエラータグファイル (.atg)
リンケージエディタ ln308	リロケータブルモジュールファイル (.r30) ライブラリファイル (.lib)	アブソリュートモジュールファイル (.x30) マップファイル (.map) リンクエラータグファイル (.ltg)
ロードモジュールコンバータ lmc308	アブソリュートモジュールファイル (.x30)	モトローラSフォーマットファイル (.mot) 拡張インテルHEXフォーマットファイル (.hex)
ライブラリアン lb308	リロケータブルモジュールファイル (.r30) ライブラリファイル (.lib)	ライブラリファイル (.lib) リロケータブルモジュールファイル (.r30) ライブラリストファイル (.lls)
クロスリファレンサ xrf308	アセンブルソースファイル (.a30) アセンブラリストファイル (.lst)	クロスリファレンスファイル (.xrf)
アブソリュートリスタ abs308	アブソリュートモジュールファイル (.x30) アセンブラリストファイル (.lst)	アブソリュートリストファイル (.als)

3.2 ソースプログラムの書き方

AS308 システム対応のソースプログラムを記述する上で必要な事項として基本規則、アドレス管理、指示命令について説明します。AS308 システムについての詳細は AS308 ユーザーズマニュアルをご参照ください。

3.2.1 基本規則

AS308 システム対応のソースプログラムを記述するための基本規則を説明します。

プログラム記述上の注意事項

AS308 システムを使用する場合には、下記の内容に注意して記述してください。

●予約語は、ソースプログラム中の名前に使用しないでください。
●AS308 システムの指示命令からピリオドをとった文字列は、AS308 の処理に影響する文字列もありますので使用しないでください。、名前に使用してもエラーとなりません。
●システムラベル(.. で始まる文字列)については AS308 システムの拡張用に用いられる可能性がありますので使用しないでください。ユーザーがソースプログラム内に記述した場合エラーは出力しません。

文字セット

次に示す文字を使って、AS308 システム対応のアセンブリプログラムを記述できます。なお、命令およびオペランドはすべて半角文字で記述してください。コメント以外には多バイト文字(漢字など)は使用できません。

英文字
A B C D E F G H I J K L M N O P Q R
S T U V W X Y Z
小文字
a b c d e f g h i j k l m n o p q r s t u
v w x y z
数字
0 1 2 3 4 5 6 7 8 9
特殊文字
" # % & ' ( ) * + , - . / : ; [\ ] ^ _ ¦ ~
空白文字
(スペース) (タブ)
改行文字
(リターン) (ラインフィード)

予約語

ＡＳ３０８ システムの予約語を下記に示します。予約語は大文字と小文字を区別しません。"abs","ABS","Abs","ABs","AbS","abS","aBs","aBS" はすべて予約語 "ABS" となります。

アセンブル指示命令

3.2.3 項　指示命令を参照してください。

ニーモニック

ABS	ADC	ADCF	ADD	ADDX	ADJNZ	AND
BAND	BCLR	BITINDEX	BMC	BMEQ	BMGE	BMGEU
BMGT	BMGTU	BMLE	BMLEU	BMLT	BMLTU	BMN
BMNC	BMNE	BMNO	BMNZ	BMO	BMPZ	BMZ
BNAND	BNOR	BNOT	BNTST	BNXOR	BOR	BRK
BRK2	BSET	BTST	BTSTC	BTSTS	BXOR	CLIP
CMP	CMPX	DADC	DADD	DEC	DIV	DIVU
DIVX	DSBB	DSUB	ENTER	EXITD	EXTS	EXTZ
FCLR	FREIT	FSET	INC	INDEXB	INDEXBD	INDEXBS
INDEXW	INDEXWD	INDEXWS	INDEXL	INDEXLD	INDEXLS	INT
INTO	JC	JEQ	JGE	JGEU	JGT	JGTU
JLE	JLEU	JLT	JLTU	JMP	JMPI	JMPS
JN	JNC	JNE	JNO	JNZ	JO	JPZ
JSR	JSRI	JSRS	JZ	LDC	LDCTX	LDIPL
MAX	MIN	MOV	MOVA	MOVHH	MOVHL	MOVLH
MOVLL	MUL	MULEX	MULU	NEG	NOP	NOT
OR	POP	POPC	POPM	PUSH	PUSHA	PUSHC
PUSHM	REIT	RMPA	ROLC	RORC	ROT	RTS
SBB	SBJNZ	SCC	SCEQ	SCGE	SCGEU	SCGT
SCGTU	SCLE	SCLEU	SCLT	SCLTU	SCMPU	SCN
SCNC	SCNE	SCNO	SCNZ	SCO	SCPZ	SCZ
SHA	SHL	SIN	SMOVB	SMOVF	SMOVU	SOUT
SSTR	STC	STCTX	STNZ	STZ	STZX	SUB
SUBX	TST	UND	WAIT	XCHG	XOR	

レジスタ／フラグ名

A0	A1	B	C	D	DCT0	DCT1
DMA0	DMA1	DMD0	DMD1	DRA0	DRA1	DRC0
DRC1	DRA0	DSA1	FB	FLG	I	INTB
IPL	ISP	O	PC	R0	R0H	R0L
R1	R1H	R1L	R2	R2R0	R3	R3R1
S	SB	SP	SVF	SVP	U	USP
VCT	Z					

演算子

SIZEOF　TOPOF

システムラベル(ピリオド二つ ".." で始まる名前)

名前の記述

名前はソースプログラムの中で、任意に定義して使用できます。
名前は次の 5 種類に分けられます。なお、名前には予約語を使用することはできません。(注)

- **ラベル**
- **シンボル**
- **ビットシンボル**
- **ロケーションシンボル**
- **マクロ名**

名前の記述規則

①使用文字は英数字と "_"(アンダースコア)で、名前の長さは 255 文字までです。
②大文字と小文字を区別します。
③先頭文字に数字は使用できません。

(注)万一使用した場合のプログラムの動作については保証できません。

名前の種類

名前の定義方法を表 3.2.1 に示します。

表 3.2.1　ユーザーが定義する名前の種類

ラベル

機能
　特定のメモリアドレスを示します。

定義方法
　名前の最後に必ず":"(コロン)を付けます。
　定義方法は二つの方法があります。
　　1.指示命令で、領域を確保する。
　　例)
```
flag:    .BLKB   1
work:    .BLKB   1
```
　　2.ソース行の先頭に名前を記述する。
　　例)
```
name1:
_name:
sym_name:
```

参照方法
　命令のオペランドに名前を記述します。
　　例)
```
JMP     sym_name
```

シンボル

機能
　定数値を示します。

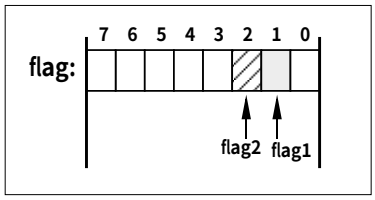
定義方法
　数値定義の指示命令を使用します。
　　例)
```
value1  .EQU    1
value2  .EQU    2
```

参照方法
　命令のオペランドにシンボルを記述します。
　　例)
```
MOV.W   R0, value2+1
value3  .EQU    value2+1
```

ビットシンボル

機能
　特定のメモリの特定のビット位置を示します。

定義方法
　ビットシンボル定義の指示命令を使用します。
　　例)
```
flag1   .BTEQU   1, flag
flag2   .BTEQU   2, flag
flag3   .BTEQU   20, flag
```



参照方法
　1ビット操作命令のオペランドに記述できます。
　　例)
```
BCLR    flag1
BCLR    flag2
BCLR    flag3
```

ロケーションシンボル

機能
　記述した行のアドレス示します。

定義方法
　必要ありません。

参照方法
　ドルマーク($)をオペランドに記述することで、
　記述した行のオペコードの1バイト目のアドレス
　を示します。

　　例)
```
JMP     $+5
```

オペランドの記述

ニーモニックおよび指示命令には、その命令の制御の対象を示す、オペランドを記述します。オペランドは記述方法によって5種類に分けられます。命令によってオペランドを持たない場合もあります。オペランドの有無や使用するオペランドの種類については、各命令の記述方法を参照してください。

●数値

数値は10進数、16進数、2進数、および8進数で記述できます。それぞれの記述方法と記述例を表3.2.2に示します。

表 3.2.2　オペランドの記述

種類	記述例	内容
2進数	10010001B 10010001b	末尾に'B'または'b'を記述します。
8進数	60702o 60702O	末尾に'O'または'o'を記述します。
10進数	9423	末尾には何も記述しません。
16進数	0A5FH 5FH 0a5fh 5fh	0～9、a～fまたはA～Fのいずれかで記述し、末尾に'H'、 または'h'を記述します。ただし、アルファベットではじまる数値の場合は、先頭に'0'を付けます。
浮動小数点数	3.4E35 3.4E-35 -.5e20 5e20	指数部には`E`または`e`の後に符号付きで指数を記述します。3.4×10^{35}は3.4E35と記述します。
名前	loop	ラベル名、シンボル名をそのまま記述します。
式	256/2 label/3	数値、名前および演算子を組み合わせて記述します。
文字列	"string" 'string'	シングルまたはダブルクォーテーションで囲って記述してください。

●浮動小数点数

浮動小数点数で表される次の範囲の値を記述できます。浮動小数点数の記述方法と記述例を表3.2.2に示します。浮動小数点数は、指示命令 ".DOUBLE"、".FLOAT" のオペランドだけに使用できます。それぞれの指示命令で記述できる範囲を表 3.2.3 に示します。

表 3.2.3　浮動小数点数の記述範囲

指示命令	記述範囲
FLOAT(32 ビット長)	**$1.17549435 \times 10^{-38} \sim 3.40282347 \times 10^{38}$**
DOUBLE(64ビット長)	**$2.2250738585072014 \times 10^{-308} \sim 1.7976931348623157 \times 10^{308}$**

●名前

ラベル名、シンボル名が記述できます。名前の記述方法と記述例を表 3.2.2 に示します。

●式

数値、名前および演算子を組み合わせた式を記述できます。演算子は複数組み合わせて記述できます。シンボル値として式を記述する場合は、式の値がアセンブ ル時に確 定するように式を記述してください。式の演算結果の値は -2147483648 ～ 2147483648 となります。浮動小数点数は式に記述できません。なお、式の記述方法と記述例を表 3.2.2 に示します。

●文字列

一部の指示命令のオペランドに文字列が記述できます。文字列では、7 ビット長 ASCII コードを使用します。シングルまたはダブルクォーテーションで囲って記述してください。文字列の記述方法と記述例を表 3.2.2 に示します。

演算子

AS308 のソースプログラムに記述できる演算子を表 3.2.4 に示します。

表 3.2.4　演算子一覧

単項演算子	
+	正の値
−	負の値
~	論理否定値
SIZEOF	セクションのサイズ(バイト数)
TOPOF	セクションの開始アドレス

二項演算子	
+	加算
−	減算
＊	乗算
／	除算
%	割った余り
>>	右へビットシフト
<<	左へビットシフト
&	論理積
¦	論理和
^	排他的論理和

条件演算子	
>	左辺値が右辺値より大きい
<	右辺値が左辺値より大きい
>=	左辺値が右辺値より大きいか等しい
<=	右辺値が左辺値より大きいか等しい
==	左辺値と右辺値が等しい
!=	左辺値と右辺値が等しくない

演算優先順位変更演算子	
()	()で囲った演算を最優先でおこないます。一つの式に複数の()が記述されている場合は、左が優先になります。()はネスティングができます。

(注 1)演算子 "SIZEOF" 及び "TOPOF" はオペランドとの間に、スペースまたはタブを記述してください。

(注 2)条件演算子は、指示命令 ".IF" 及び ".ELIF" のオペランドだけに記述できます。

演算優先順位

演算は演算子の優先順位の高いものから演算します。優先順位を表3.2.5に示します。優先順位が等しい場合には、式の左から順に演算します。演算の順番は(　)で囲うことで変更できます。

表 3.2.5　演算行の記述

	優先順位	演算子の種類	内容
高 ↑	1	演算順位変更演算子	(,)
	2	単項演算子	+ , −, ~ , SIZEOF , TOPOF
	3	二項演算子1	＊,／, %
	4	二項演算子2	+, −
	5	二項演算子3	>> , <<
	6	二項演算子4	&
	7	二項演算子5	¦ , ^
↓ 低	8	条件演算子	> , < , >= , <= , == , !=

行の記述

AS308 はソースプログラムを行単位で処理します。改行文字で区切られ、改行文字の直後の文字から、次の改行文字までを 1 行とします。1 行に記述可能な最大文字数は 255 文字です。行は記述されている内容によって 5 種類に分けられます。それぞれの記述方法を表 3.2.6 に示します。

- ●指示命令行
- ●アセンブリソース行
- ●ラベル定義行
- ●コメント行
- ●空行

表 3.2.6　行の種類

指示命令行

機能
as308の指示命令を記述した行です。

記述方法
指示命令は、一行に一つのみ記述できます。
指示命令行には、コメントを記述できます。
注意事項
指示命令とニーモニックを同じ一行に記述することはできません。

例)

```
        .SECTION    program,DATA
        .ORG        400H
sym     .EQU        0
work:   .BLKB       1
        .ALIGN
        .PAGE       "newpage"
        .ALIGN
```

アセンブリソース行

機能
ニーモニックを記述した行です。

記述方法
アセンブリソース行には、ラベル名（先頭）、コメントを記述できます。
注意事項
１行には、２つ以上のニーモニックは記述できません。
指示命令とニーモニックを同じ一行に記述することはできません。

例)

```
        MOV.W       #0,R0
        RTS
main:   MOV.W       #0,A0
        RTS
```

ラベル定義行

機能
ラベル名だけを記述した行です。

記述方法
ラベル名の直後には必ずコロン(:)を記述してください。

例)

```
start:
label:  .BLKB   1
main:   nop
loop:
```

コメント行

機能
コメントだけを記述した行です。

記述方法
コメントの先頭には、必ずセミコロン(;)を記述してください。

例)

```
        ;コメント行
        MOV.W       #0,A0
```

空行

機能
見かけ上何も記述していない行です。

記述方法
スペース、タブまたは、改行コードだけを記述してください。

3.2.2　アドレス管理

この項では AS308 システムのアドレス制御方法について説明します。

AS308 システムはアドレス制御を行なう際に RAM サイズと ROM サイズは考慮しません。したがって実際に使用するシステムにおけるアドレス範囲を考えて、ソースプログラムの記述やリンク処理を行なってください。

アドレスの管理方法

AS308 システムは、セクション単位でアドレスを管理します。セクションの区切りは、次のように決められます。なお、セクションのネスティングは定義できません。

セクションの区切り

①指示命令 ".SECTION" を記述した行から、次の指示命令 ".SECTION" を記述した前の行までの間

②指示命令 ".SECTION" を記述した行から、指示命令 ".END" を記述した前の行までの間

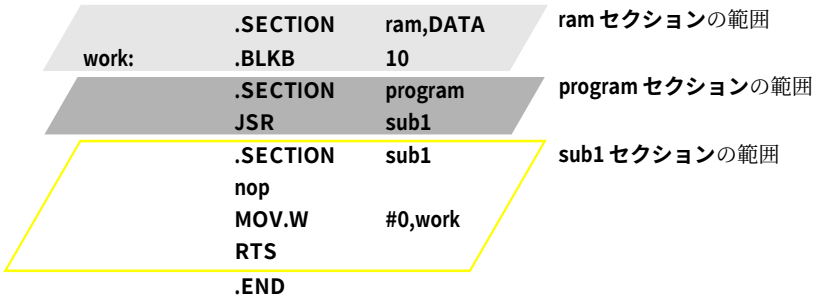


図 3.2.1　AS308 システムにおけるセクションの範囲

セクションのタイプ

アドレス制御の単位となるセクションには、タイプを設定できます。タイプによって、セクション内に記述できる命令が異なります。

表 3.2.7　セクションのタイプ

タイプ	内容と記述例
ＣＯＤＥ (プログラム領域)	・プログラムを記述する領域です。 ・領域を確保する指示命令を除く全ての命令が記述できます。 ・CODEタイプのセクションは、アブソリュートモジュールにおいて、ROM領域に配置されるように指定してください。 例) .SECTION　program,CODE
ＤＡＴＡ (データ領域)	・内容が変更可能なメモリを配置する領域です。 ・領域を確保する指示命令が記述できます。 ・DATAタイプのセクションは、アブソリュートモジュールにおいて、RAM領域に配置されるように指定してください。 例) .SECTION　mem,DATA
ＲＯＭＤＡＴＡ (固定データ領域)	・プログラム以外の固定データを記述する領域です。 ・データを設定する指示命令が記述できます。 ・領域を確保する指示命令を除く全ての命令が記述できます。 ・ROMDATAタイプのセクションは、アブソリュートモジュールにおいて、ROM領域に配置されるように指定してください。 例) .SECTION　const,ROMDATA

セクションの属性

アドレス制御の単位となるセクションは、アセンブル時に属性が付けられます。

表 3.2.8　セクションの属性

属性	内容と記述例
相対	・アセンブル時にセクション内のアドレスがリロケータブル値となります。 ・相対属性セクション内で定義されたラベルの値はリロケータブルです。
絶対	・アセンブル時にセクション内のアドレスがアブソリュート値となります。 ・絶対属性セクション内で定義されたラベルの値はアブソリュートです。 ・セクションを絶対属性にするためには、指示命令".SECTION "を記述した次の行で、指示命令".ORG"でアドレスを指定してください。 例) .SECTION program,DATA .ORG 1000H

先頭アドレスの偶数番地指定

相対属性のセクションでは、リンク時に決定するセクションのスタートアドレスが必ず偶数番地になるように設定することができます。調整を行ないたいときは、指示命令 ".SECTION" のオペランドに "ALIGN" を指定してください。

例)

```
.SECTION    program, CODE, ALIGN
```

AS308 システムのアドレス管理

AS308 システムが、複数のファイルにわたって記述されたアセンブリソースプログラムを、1 つの実行形式のファイルに変換する方法を次に示します。

as308 のアドレス管理

(1) 絶対属性となるセクションは、指定されているアドレスから順に絶対アドレスを決　　定します。

(2) 相対属性となるセクションは、セクション毎に0から順にアドレスを決定します。相　　対属性のセクションの開始アドレスはすべて 0 です。

ln308 のアドレス管理

(1) 全てのファイルの同一名のセクションを、指定されたファイルの順に並べます。

(2) まとめたセクションの開始アドレスは、ln308 のコマンドオプション（-order）の指　　定に従って決定されます。

(3) 最初のセクションの開始アドレスは、指定がなければ 0 から順に決定します。

(4) 異なる名前のセクションの順序は、指定がなければ ln308 に読み込まれた順に連続　　して配置されます。

(5) 同一名のセクションで相対属性の後に絶対属性を配置しようとした場合はエラー　　となります。

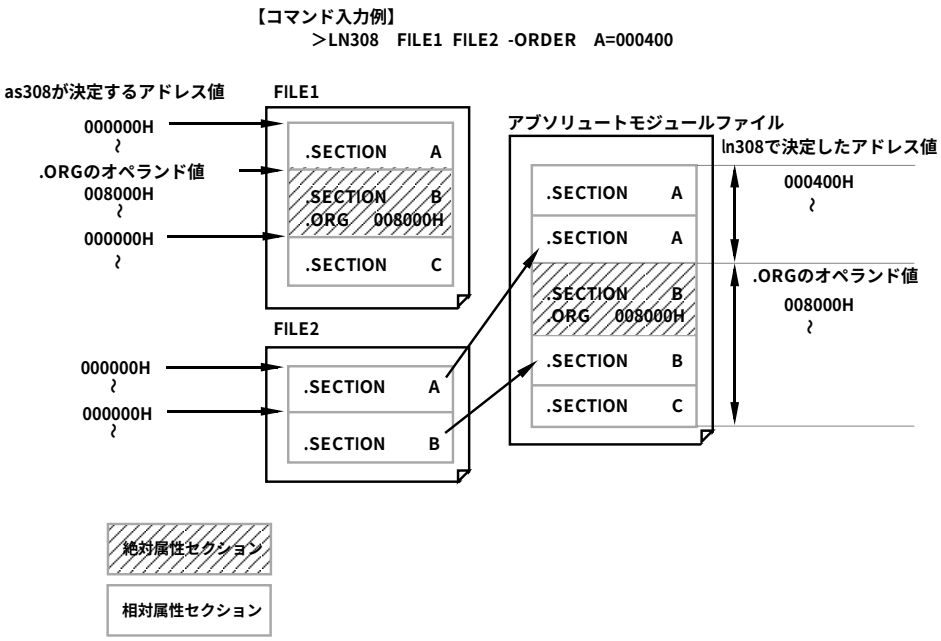


図 3.2.2　アドレス管理の例

インクルードファイルの読み込み

AS308 システムは、ソースプログラムの任意の行で、インクルードファイルを読み込むことができます。プログラムの可読性の向上などに利用できます。

インクルードファイルの読み込み

指示命令".INCLUDE" のオペランドに読み込みたいファイル名を記述します。この行　の位置にインクルードファイルの内容が全て読み込まれます。

例)

```
.INCLUDE        initial.inc
```

ソースファイル(sample.a30)

```
        .SECTION    memory,DATA
work:   .BLKB   10
flags:  .BLKW   1
        .SECTION    init
        .INCLUDE    initial.inc
        .SECTION    program,CODE
main:
            ⋮
        .END
```

インクルードファイル(initial.inc)

```
loop:
        MOV.W       #10,A0
        MOV.B       #0,work[A0]
        INC.W       A0
        JNZ         loop
        MOV.W       #0,flags
```

展開イメージ

アセンブル実行後

```
                            .SECTION    memory,DATA
000000          work:       .BLKB   10
00000A          flags:      .BLKW   1
                            .SECTION    init
000000                      .INCLUDE    initial
000000          loop:   MOV.W       #10,A0
000002                  MOV.B       #0,work[A0]
000005                  INC.W       A0
000007                  JNZ         loop
000009                  MOV.W       #0,flags

000000                      .SECTION    program,CODE
                main:
                            ⋮
                            .END
```

as308が出力したアドレス

図 3.2.3　インクルードファイルの読み込み

グローバルとローカルのアドレス管理

AS308 システムにおけるラベル、シンボル及びビットシンボルの値の管理について説明します。
AS308 システムでは、ラベル、シンボル及びビットシンボルの値をグローバルとローカル、リロケータブルとアブソリュートに分けて扱います。以下にそれぞれの定義を示します。

●グローバル

指示命令".GLB"で指定したラベル及びシンボルは、それぞれグローバルラベル及びグローバルシンボルとなります。
指示命令 ".BTGLB" で指定したビットシンボルはグローバルビットシンボルとなります。
ファイル内に定義がある名前をグローバル指定したものは、外部のファイルからの参照が可能になります。
ファイル内に定義のない名前をグローバル指定したものは、外部のファイルで定義されている名前を参照する外部参照ラベル、シンボル、ビットシンボルとなります。

●ローカル

指示命令 ".GLB" または ".BTGLB" で指定のない名前は、すべてローカルとなります。
ローカルな名前は、定義した同一ファイル内でだけ参照できます。
ローカルな名前は、別のファイルで同一のラベル名を使用できます。

●リロケータブル

相対セクション内のローカルラベル、シンボル、ビットシンボルの値は、リロケータブルになります。
外部参照となるグローバルラベル、シンボル、ビットシンボルの値は、リロケータブルとなります。

●アブソリュート

絶対属性セクション内に定義のあるローカルラベル、シンボル、ビットシンボルの値は、アブソリュートとなります。

アブソリュートとなるラベル、シンボル、ビットシンボルは、アセンブル実行時に as308 が値を決定します。その他のラベル、シンボル、ビットシンボルの値は、すべてリンク時に ln308 が決定します。(注)

ラベルの関係を図 3.2.4 に示します。

(注) リンクした結果、アセンブラが決定した分岐命令やアドレッシングモードが指定可能な範囲外であった場合、ワーニングが出力されます。

file1.a30

```
        .GLB    ver,sub1,port
        .SECTION        device,DATA
        .ORG            400H
port:   .BLKW   1

        .SECTION        program,CODE
        .ORG            8000H
main:
        JSR             sub1

        .SECTION        str,ROMDATA
ver:    .BYTE   "program version 1"
        .END
```

ラベルのグローバル宣言(必須)

file1のアブソリュートラベル
- port　：グローバル、外部ファイルから参照可能
- main　：ローカル

file1のリロケータブルラベル
- ver　：グローバル、外部ファイルから参照可能
- sub1　：グローバル、外部ファイルを参照

file2.a30

```
        .GLB    ver,sub1,port

        .SECTION        program
        .ORG            0C000H
sub1:
        MOV.W           #0, A0
loop_s1:
        MOV.B           ver[A0], port
        INC.W           A0
        CMP.B           ver[A0], 0
        JNZ             loop_s1
        RTS
        .END
```

ラベルのグローバル宣言(必須)

file2のアブソリュートラベル
- sub1　：グローバル、外部ファイルから参照可能
- loop_s1　：ローカル

file2のリロケータブルラベル
- ver　：グローバル、外部ファイルを参照
- port　：グローバル、外部ファイルを参照

図 3.2.4　ラベルの関係

3.2.3 指示命令

ソースプログラムには、M16C/80 シリーズの機械語命令の他に AS308 システムの指示命令を使用できます。指示命令には下記の種類があります。指示命令の使い方について、種類別に説明します。

●アドレス制御命令

アセンブル実行時にアドレス決定の指示ができます。

●アセンブル制御指示命令

AS308 の実行について指示できます。

●リンク制御指示命令

アドレス再配置制御のための情報を定義できます。

●リスト制御指示命令

AS308 が生成するリストファイルのフォーマットを制御できます。

●分岐最適化制御指示命令

分岐命令の最適選択を AS308 に指示できます。

●条件アセンブル制御指示命令

アセンブル実行時に設定した条件によって、コード生成するブロックを選択できます。

●拡張機能指示命令

上記以外の制御を行なう指示命令です。

●クロスツールが出力する指示命令

M16C/80 シリーズ用ツールソフトウエアが出力する指示命令は、ユーザーはソースプログラムに記述できません。記述した場合の動作については保証できません。

アドレス制御

命令	機能	使い方･記述例
.ORG	アドレスを宣言します。	セクション指示命令".SECTION"の直後に記述してください。セクション指示命令の直後にない場合、そのセクションは相対属性セクションとなります。相対属性のセクション内には記述できません。 .ORG 0F0000H .ORG offset .ORG 0F0000H + offset
.BLKB	1バイト単位でRAM領域を確保します。	DATAセクション内に確保する領域数を記述してください。ラベル名を定義する場合は必ずコロン(:)を付けてください。 例) .BLKB 1 .BLKW number .BLKA number+1 label: .BLKL 1 label: .BLKF number label: .BLKD number+1
.BLKW	2バイト単位でRAM領域を確保します。	
.BLKA	3バイト単位でRAM領域を確保します。	
.BLKL	4バイト単位でRAM領域を確保します。	
.BLKF	4バイト単位でRAM領域を確保します。	
.BLKD	8バイト単位でRAM領域を確保します。	
.BYTE	1バイト長のデータをROM領域に格納します。	複数のオペランドを記述するときは、カンマ(,)で区切ってください。ラベル名を定義する場合は必ずコロン(:)を付けてください。 .FLOAT と.DOUBLEについてはオペランドに浮動小数点数を記述してください。 例) .SECTION value, ROMDATA .BYTE 1 .BYTE 1,2,3,4,5 .WORD "da","ta" .ADDR symbol .LWORD symbol+1 .FLOAT 5E2 constant: .DOUBLE 5e2
.WORD	2バイト長のデータをROM領域に格納します。	
.ADDR	3バイト長のデータをROM領域に格納します。	
.LWORD	4バイト長のデータをROM領域に格納します。	
.FLOAT	4バイトで表される浮動小数点データをROM領域に格納します。	
.DOUBLE	8バイトで表される浮動小数点データをROM領域に格納します。	
.ALIGN	奇数アドレスを偶数アドレスに補正します。	セクション定義の際にアドレス補正を指示している相対属性セクションか絶対属性セクション内に記述できます。 例) .SECTION work,DATA,ALIGN ram1: .BLKB 1 .ALIGN ram2: .BLKW 2 .SECTION const,ROMDATA .ORG 0F000H data1: .BYTE 12H .ALIGN data 2: .WORD symbol .END

アセンブル制御

命令	機能	使い方･記述例
.EQU	シンボルを定義します。	前方参照となるシンボル名は記述できません。オペランドにはシンボル、式を記述できます。シンボル、ビットシンボルはグローバル指定ができます。 例) symbol .EQU 1 symbol1 .EQU symbol+1 bit0 .BTEQU 0,400H bit1 .BTEQU 1,symbol1
.BTEQU	ビットシンボルを定義します。	
.END	アセンブルソースの終了を宣言します。	一つのアセンブリソースファイルに必ず一つ以上記述してください。as308は本指示命令以降の行については、コード生成などの処理は行いません。 例) .END
.SB	SBレジスタ値を仮定します。	".SBSYM", ".FBSYM"によりアドレッシングモードを選択する前に、必ず".SB",".FB"でSB、FBレジスタ値を設定してください。 またレジスタ値の設定は実際のレジスタ(SB,FB)には設定されませんので、本指示命令の直前または直後にレジスタ値を設定する命令を記述してください。 例) .SB 400H LDC #400H,SB .SBSYM sym1,sym2 .FB 500H LCD #580H,FB .FBSYM sym3,sym4
.SBSYM	SB相対アドレッシングを選択します。	
.SBBIT	ビット命令のSB相対アドレッシングを選択します。	
.FB	FBレジスタ値を仮定します。	
.FBSYM	FB相対アドレッシングを選択します。	
.INCLUDE	ファイルを指定位置に読み込みます。	オペランドのファイル名には必ず拡張子を記述してください。オペランドには、指示命令"..FILE"や"@"を含む文字列が記述できます。 例) .INCLUDE initial.a30 .INCLUDE ..FILE@.inc

リンク制御

命令	機能	使い方･記述例
.SECTION	セクション名を定義します。	セクションタイプ及びALIGNを同時に指定する場合はカンマで区切ってください。セクションタイプは、'CODE','ROMDATA','DATA'のいずれかを記述できます。セクションタイプを省略した場合は、"CODE"として処理します。 例) .SECTION program,CODE NOP .SECTION ram,DATA .BLKB 10 .SECTION dname,ROMDATA .BYTE "abcd" .END
.GLB	グローバルラベルを指定します。	オペランドに複数のシンボル名を記述する場合はカンマ(,)で区切ってください。 例) .GLB name1,name2,name3 .BTGLB flag4 .SECTION program MOV.W #0,name1 BCLR flag4
.BTGLB	グローバルビットシンボルを指定します。	
.VER	指定した文字列をマップファイルにバージョン情報として出力します。	オペランドは、1行の範囲内で記述してください。1つのアセンブリソースファイルに1度しか記述できません。 例) .VER 'strings' .VER "strings"

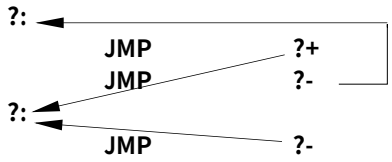
リスト制御

命令	機能	使い方･記述例
.LIST	リストファイルへの行データの出力を制御します。	行出力を停止する場合は'OFF'、開始する場合は'ON'をオペランドに'記述してください。指定しない場合はすべての行をリストファイルに出力します。 例) .LIST OFF MOV.B #0,R0L MOV.B #0,R0L MOV.B #0,R0L .LIST ON
.PAGE	リストファイルの指定位置で改ページを行ないます。	オペランドは、クォーテーション(')またはダブルクォーテーション(")で囲って記述してください。オペランドは省略できます。 例) .PAGE .PAGE "strings" .PAGE 'strings'
.FORM	リストファイルの1ページの桁及び行数を指定します。	1つのアセンブリソースファイルに複数回記述できます。行数及び桁数にはシンボルを記述できます。前方参照となるシンボルは記述できません。出力しない場合は66行､140桁で出力します。 例) .FORM 20,80 .FORM 60 .FORM ,100 .FORM line,culmn

分岐命令最適化制御

命令	機能	使い方･記述例
.OPTJ	分岐命令とサブルーチン呼び出しの最適化を制御します。	オペランドには、分岐命令の最適制御、最適化対象外の無条件分岐命令選択、最適化対象外のサブルーチン呼び出し命令選択の項目が記述できます。各項目の指定順序は任意で、省略もできます。省略した場合は分岐距離は初期値または以前に指定した内容から変化しません。 例) 下記に示す組み合わせのオペランドが記述できます。 .OPTJ OFF .OPTJ ON .OPTJ ON,JMPW .OPTJ ON,JMPW,JSRW .OPTJ ON,JMPW,JSRA .OPTJ ON,JMPA .OPTJ ON,JMPA,JSRW .OPTJ ON,JMPA,JSRA .OPTJ ON,JSRW .OPTJ ON,JSRA

拡張機能指示命令

命令	機能	使い方･記述例
.ASSERT	ファイルまたは標準エラー出力に指定した文字列を出力します。	ダブルクォーテーションで囲った文字列をファイルに出力する場合は、">"または">>"に続けてファイル名を指定してください。 >は新規にファイルを生成して、そのファイルにメッセージを出力します。以前に同一名のファイルが或る場合は、そのファイルに上書きされます。 >>はファイルの内容に追加して、メッセージを出力します。指定したファイルが存在しない場合は、新しくファイルを生成します。 ファイル名に指示命令"..FILE"を記述できます。 例) .ASSERT "string" > sample.dat .ASSERT "string" >> sample.dat .ASSERT "string" > ..FILE
?	テンポラリラベルの指定及び参照を行ないます。	テンポラリラベルとして定義したい行に"?:"を記述してください。直前に定義したテンポラリラベルを参照したい場合は、命令のオペランドに"?-"、直後に定義したテンポラリラベルを参照したい場合は、命令のオペランドに"?+"を記述してください。 例) 
..FILE	ソースファイル名情報を示します。	指示命令".ASSERT"及び指示命令".INCLUDE"のオペランドに記述できます。コマンドオプション"-F"を指定すると"..FILE"はコマンド行で指定したソースファイル名に固定されます。オプションを指定しない場合は"..FILE"が記述されているファイル名になります。 例) .ASSERT "sample" > ..FILE .INCLUDE ..FILE@.inc .ASSERT "sample" > ..FILE@.mes
@	@の前後の文字列を連結します。	1行に複数回記述できます。 連結した文字列を名前とする場合は、本命令の前後にスペース及びタブを記述しないでください。 例) .ASSERT "sample" > ..FILE@.dat 次のようなマクロ定義も可能です。 mov_nibble .MACRO p1,src,p2,dest MOV@p1@p2 src,dest .ENDM

条件アセンブル指示命令

命令	機能	使い方･記述例
.IF	条件アセンブルの開始を示します。	オペランドには必ず条件式を記述してください。 例) .IF TYPE==0 .BYTE "Proto Type Mode" .ELIF TYPE>0 .BYTE "Mass Production Mode" .ELSE .BYTE "Debug Mode" .ENDIF 条件式の記述規則) 演算結果のオーバフロー及びアンダーフローは判断しません。シンボルは前方参照(本指示命令行より後に定義されているシンボルを参照)はできません。前方参照のシンボルや、未定義のシンボルを記述した場合は、値を0として式を判断します。 条件式の記述例) sym<1 sym < 1 sym+2 < data1 sym+2 < data1+2 'smp1' ==name
.ELIF	条件アセンブルの条件を示します。	必ずオペランドには条件式を記述してください。本指示命令は1つの条件アセンブルブロック内に複数記述できます。 例) 同上
.ELSE	条件偽の際にアセンブルを行なうブロックの開始を示します。	本指示命令は、条件アセンブルブロック内に1つ以下記述できます。オペランドはありません。 例) 同上
.ENDIF	条件アセンブルの終了を示します。	本指示命令は、条件アセンブルブロック内に必ず1つ記述してください。オペランドはありません。 例) 同上

3.2.4 マクロ機能

AS308 で使用できるマクロ機能について説明します。AS308 には以下に示すマクロ機能があります。

●マクロ機能

マクロ指示命令".MACRO"～".ENDM"で定義し、定義したマクロを呼び出すことでマクロ機能を使用できます。

●繰返しマクロ機能

マクロ指示命令 ".MREPEAT" ～ ".ENDR" を記述することで、繰返しマクロ機能を使用できます。

マクロ定義とマクロ呼び出しの関係を図 3.2.5 に示します。

マクロ定義

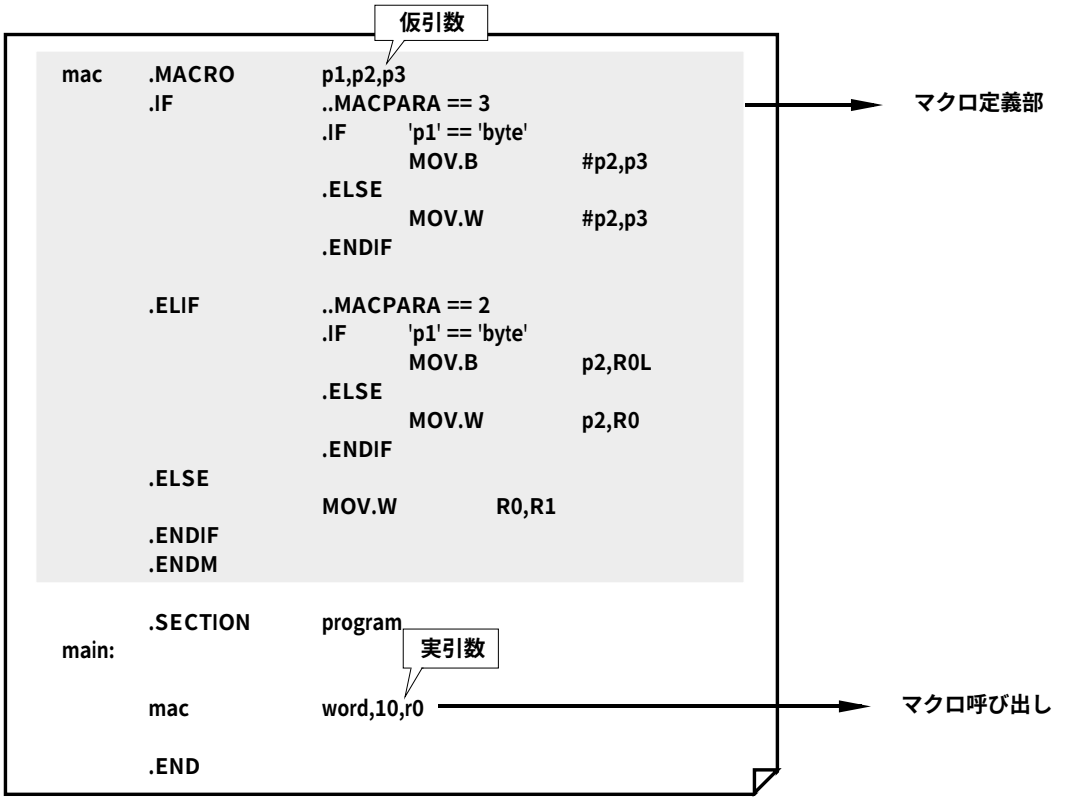
マクロ定義はマクロ指示命令".MACRO"を使用して１行以上の命令の集まりを１つのマクロ名に定義します。定義終了は ".ENDM" で示し、".MACRO" と ".ENDM" に囲まれた行をマクロボディと呼びます。

マクロボディにはソースプログラムに記述可能なすべての命令を記述できますが、ビットシンボルは記述できません。
マクロのネストはマクロ定義及びマクロ呼び出しを含めて 65535 レベル以下です。
マクロ名及びマクロ引数は大文字、小文字を区別して扱います。

マクロ呼び出し

マクロ定義されているマクロボディの内容をマクロ指示命令".MACRO"で定義したマクロ名を記述することで、その行に呼び出しを行います。マクロ名の外部参照はできません。複数ファイルから、同一のマクロを呼び出す場合はインクルードファイル内マクロを定義し、そのファイルをインクルードしてください。

マクロ記述例　（マクロ実引数の数を判断して、条件アセンブルを行う。）



展開後

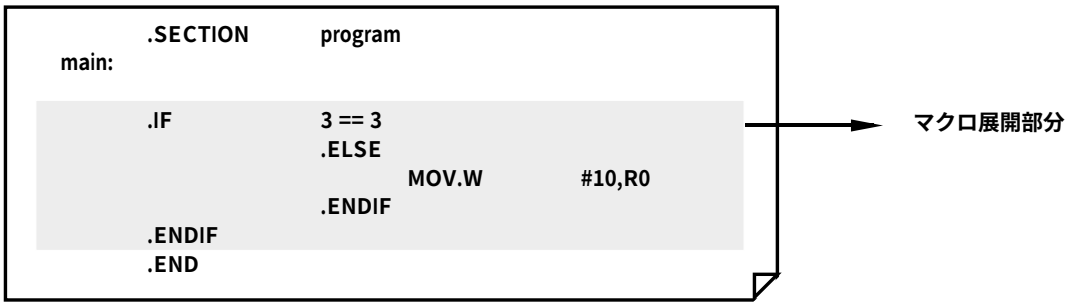


図 3.2.5　マクロ定義と呼び出し例 1

マクロローカル

指示命令".LOCAL"で宣言したマクロローカルラベルは、マクロ定義内のみ使用できます。マクロローカル宣言をしたラベルは、マクロの範囲外で同一のラベルを記述できます。図3.2.6に記述例を示します。この場合、m1がマクロローカルラベルになります。

```
name    .MACRO      source,dest,top
        .LOCAL      m1
m1:
        nop
        jmp         m1

        .ENDM
```

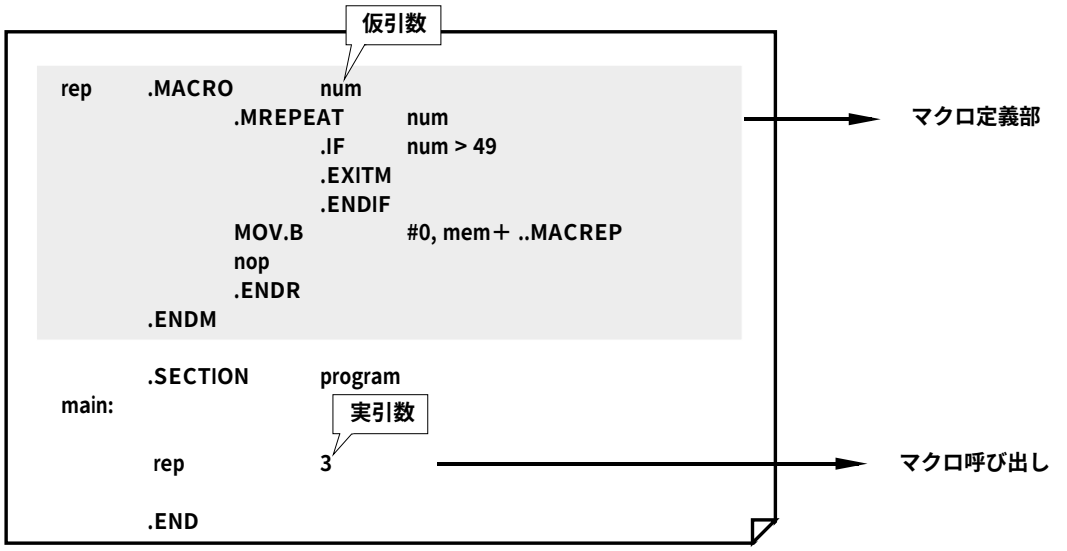
図 3.2.6　マクロ定義と呼び出し例 2

繰り返しマクロ機能

マクロ指示命令 ".MREPEAT" ～ ".ENDM" で囲まれたボディを指定した回数分繰り返し、指定した行に展開します。繰返しマクロのマクロ呼び出しはありません。

繰り返しマクロのマクロ定義とマクロ呼び出しの関係を図 3.2.7 に示します。

繰り返しマクロ記述例 （マクロ実引数の値で示された分だけ"MOV","nop"命令を展開する。）



展開後

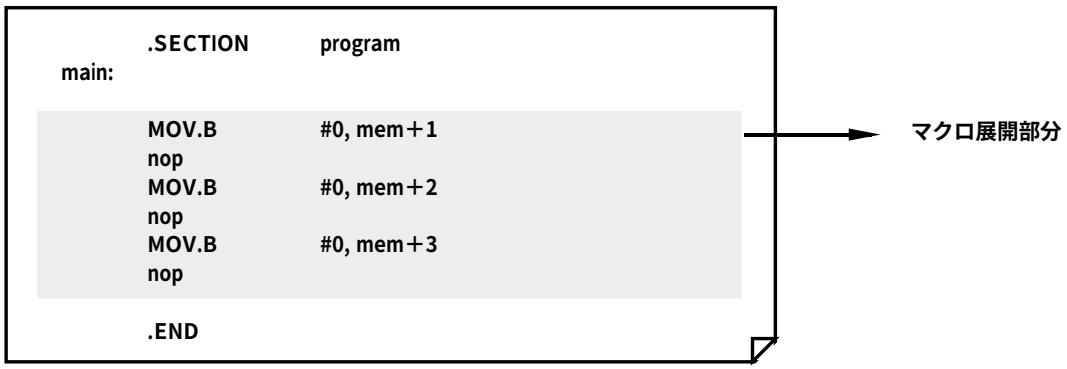


図 3.2.7　マクロ定義と呼び出し例 3

マクロ指示命令

AS308 のマクロ命令には、以下の種類があります。

●マクロ指示命令
マクロボディの開始や終了または中止、マクロ内のローカルなどの宣言をします。
●マクロシンボル
マクロ記述の式の項目として記述します。
●文字列関数
文字列の情報を表します。

マクロ指示命令

命令	機能	使い方･記述例
.MACRO	マクロ名の定義及びマクロ定義の開始を示します。	オペランドには必ず条件式を記述してください。 仮引数は１行に80個まで記述できます。仮引数をダブルクォーテーションで囲わないでください。 <記述形式> マクロ定義 (マクロ名)　.MACRO　[(仮引数)[,(仮引数)...]] マクロ呼び出し (マクロ名)　[(実引数)[,(実引数)...]] <記述例> 図3.2.5を参照してください。
.ENDM	マクロ定義の終了を示します。	必ず".MACRO"に対応させて記述してください。 <記述例> 図3.2.5を参照してください。
.LOCAL	オペランドに示されたラベルがマクロローカルラベルであることを宣言します。	マクロボディ内に記述してください。 オペランドはカンマで区切って複数のラベルを記述できます。このとき最大ラベル数は100個です。 <記述例> 図3.2.6を参照してください。
.EXITM	マクロボディの展開を中止し、最も近い".ENDM"に制御を渡します。	マクロ定義のボディ内に記述してください。 <記述例> 図3.2.7を参照してください。
.MREPEAT	繰返しマクロ定義の開始を示します。	ボディを指定した数値回繰り返して展開します。 繰返し回数は最大65535回です。 <記述例> 図3.2.7を参照してください。
.ENDR	繰返しマクロ定義の終了を示します。	必ず".MREPEAT"に対応させて記述してください。 <記述例> 図3.2.7を参照してください。

マクロシンボル

命令	機能	使い方･記述例
..MACPARA	マクロ呼び出しの際に与えられた実引数の個数を示します。	".MACRO"によるマクロ定義のボディ内に式の項として記述できます。マクロボディの外に記述した場合、値は０となります。 <記述例> 図3.2.5を参照してください。
..MACREP	繰り返しマクロが展開されている回数を示します。	".MREPEAT"によるマクロ定義のボディ内に式の項として記述できます。条件アセンブルのオペランドとしても記述できます。繰り返す度に値は１→２→３...と増加します。マクロボディの外に記述した場合、値は０となります。 <記述例> 図3.2.7を参照してください。

文字列関数

命令	機能	使い方･記述例
.LEN	オペランドに記述した文字列長を示します。	オペランドは必ず"{}"で囲い、文字列はクォーテーションで囲ってください。文字列には７ビットアスキーコードの文字が記述できます。 式の項として記述できます。 <記述形式> .LEN {"(文字列)"} .LEN {'(文字列)'} <記述例> 図3.2.8を参照してください。
.INSTR	オペランドで指定した文字列のなかで、検索文字列の開始位置を示します。	オペランドは必ず"{}"で囲い、文字列はクォーテーションで囲ってください。文字列には７ビットアスキーコードの文字が記述できます。 検索開始位置を１にした場合は、文字列の先頭を示します。 <記述形式> .INSTR {"(文字列)","(検索文字列)",(検索開始位置)} .INSTR {'(文字列)','(検索文字列)',(検索開始位置)} <記述例> 図3.2.9を参照してください。
.SUBSTR	オペランドで指定した文字列の位置から、指定した文字数を切り出します。	オペランドは必ず"{}"で囲い、文字列はクォーテーションで囲ってください。文字列には７ビットアスキーコードの文字が記述できます。 切り出し開始位置を１にした場合は、文字列の先頭を示します。 <記述形式> .SUBSTR {"(文字列)",(開始位置),(文字数)} .SUBSTR {'(文字列)',(開始位置),(文字数)} <記述例> 図3.2.10を参照してください。

.LEN 記述例

図 3.2.8 の例では、指定した文字列の文字列長 "Printout_data" は "13"、"Sample" は "6" を表しています。

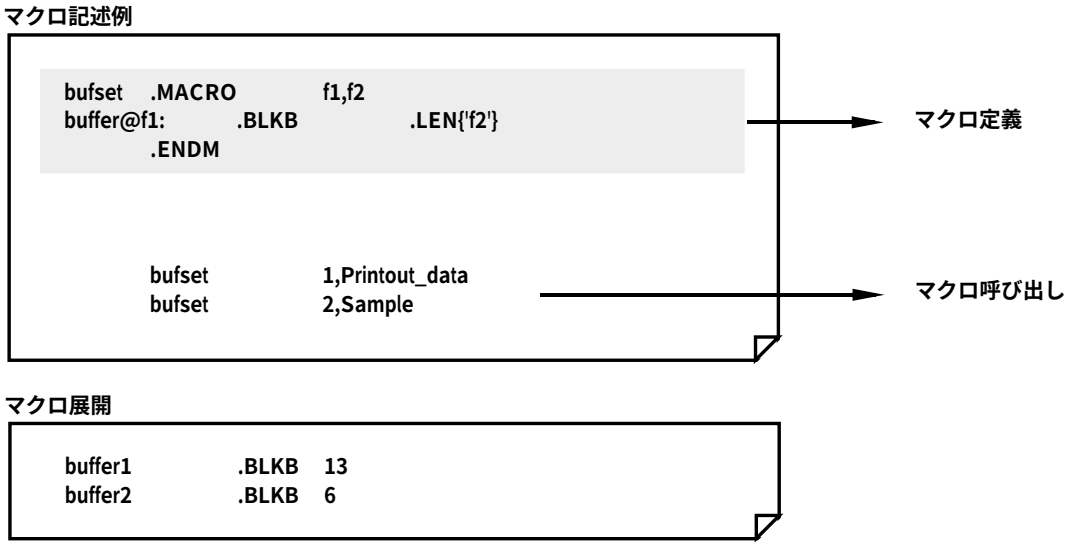


図 3.2.8　.LEN 記述例

.INSTR 記述例

図 3.2.9 の例では、指定した文字列(japanese)の先頭ｘ(top)からの、"se" 文字列の位置(7)を切り出しています。

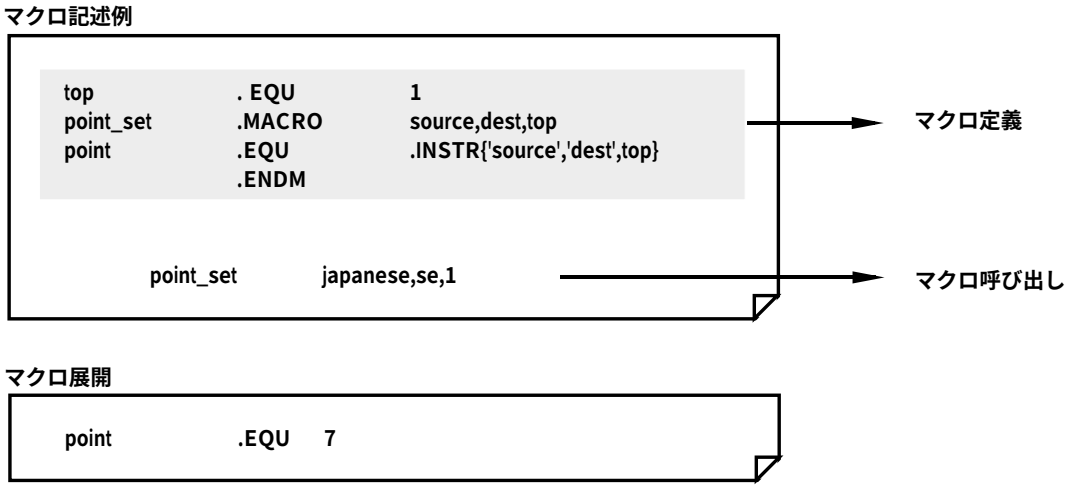


図 3.2.9　.INST 記述例

.SUBSTR 記述例

図 3.2.10 の例では、マクロの実引数として与えられた文字列の長さを、".MREPEAT" のオペランドに与えています。"..MACREP" は ".BYTE" の行を実行するごとに、１→２→３→４ ... と増加します。したがって、マクロ実引数として与えた文字列の先頭文字から順に１文字づつ ".BYTE' のオペランドに与えることになります。

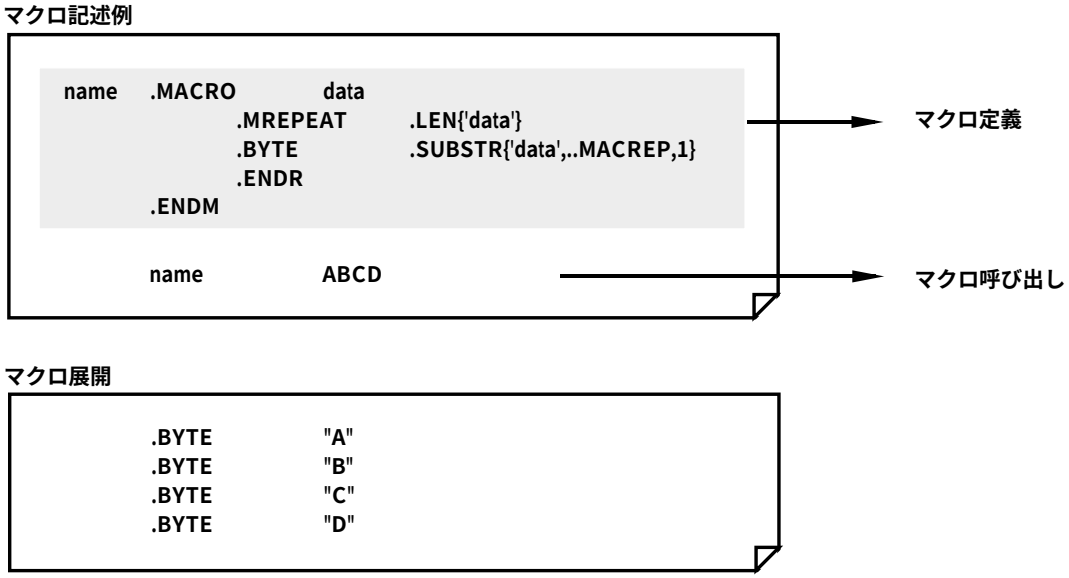


図 3.2.10　.SUBSTR 記述例

3.2.5　M16C/60 との相違点

AS308（M16C/80）では AS30（M16C/60）にはなかったアドレッシングモードや命令セット、及びアセンブルオプションなどが追加／変更されていますが、削除されたアドレッシングモードや命令セット、アセンブルオプションも幾つか存在します。本項では、その削除／変更された部分について示します。

変更されたアドレッシングモード

M16C/80 ではメモリ空間が 1M から 16M に拡張されたのに伴い、一般命令アドレッシングやビット命令アドレッシングなど全てのアドレッシングモードのアクセス可能範囲が拡大され、100000H 番地以降のアクセスが可能となっています。

したがって、M16C/80 では使用できなくなったアドレッシングモードについて以下に示します。

■特定命令アドレッシングの以下に示す各アドレッシングの削除

- 20 ビット絶対
- 20 ビットディスプレースメント付きアドレスレジスタ相対
- 32 ビットレジスタ直接
- 32 ビットアドレスレジスタ間接

＊ M16C/80 では全て一般命令アドレッシングでアクセスが可能となります。

■ビット命令アドレッシングの以下に示すアドレッシングの変更

- レジスタ直接

レジスタのビット操作については、ビット０～７までの指定となります。

M16C/60 でのレジスタ直接 （bit は 0 ～ 15）		M16C/80 でのレジスタ直接 （bit は 0 ～ 7）
bit, R0		bit, R0L
bit, R1		bit, R0H
bit, R2	→	bit, R1L
bit, R3		bit, R1H
bit, A0		bit, A0（下位 8bit のみ指定可能）
bit, A1		bit, A1（下位 8bit のみ指定可能）

削除された命令セット

M16C/80 では以下に示す命令が削除されています。

- **LDE 命令**
- **STE 命令**
- **LDINTB(LDC マクロ)命令**

＊M16C/80ではメモリ空間が16Mに拡張されており、10000H番地以降のアドレスについても一般命令アドレッシングでアクセスが可能となっているため、**LDE, STE** 命令が削除されています。

M16C/60 命令との互換性（１）

M16C/80 では、各命令で選択可能な src, dest（使用可能なオペランド）が M16C/60 と　異なっている命令がいくつか存在します。

AS308ではAS30(M16C/60シリーズ)で開発されたプログラムをアセンブルするためのコマンドオプションとして、"-mode60" をサポートしています。

以下に本オプションを付加した場合の AS308 の処理を示します。

- **MOV, CMP, ADD, SUB, AND, OR, NOT, PUSH, POP命令において、フォーマット指定子が記述されていてもそのフォーマット指定子を無視します。**
- **JMPI, JSRI 命令のアドレッシングモード指定子を無視します。**
- **ADD 命令でスタックポインタ(SP)へ加算する場合、サイズ指定子を ".L" として処理します。**
- **LDINTB 命令を LDC 命令に置き換えて処理します。**
  表 3.2.9 を参照してください。
- **STZ, STNZ, STZX 命令のオペランドをバイトサイズで処理します。**
- **LDE, STE 命令を MOV 命令に置き換えて処理します。**
  表ＸＸＸを参照してください。
- **1 ビット操作命令を AS308(M16C/80)対応の命令に置き換えて処理します。**
  表ＸＸＸを参照してください。
- **ビット操作命令の BCLR, BAND, BOR, BXOR, BNOT, BNAND, BNOR, BNXOR, BTST, BNTST, BTSTC, BTSTS, BMcnd についても、置き換え命令一覧で示している BSET 命令と同様に置き換えます。**

"-mode60" オプションによる置き換え命令一覧

表 3.2.9　置き換え命令一覧

AS30ソース記述形式	AS308置き換え結果
LDINTB　#imm20	LDC　#imm24, INTB
LDE.B/W　dbs:20, dest	MOV.B/W　abs, dest
LDE.B/W　dsp:20[A0], dest	MOV.B/W　dsp[A0], dest
STE.B/.W　src, abs:20	MOV.B/.W　src, abs
STE.B/.W　src, dsp:20[A0]	MOV.B/.W　src, dsp[A0]
BSET:G　bit, R0	BSET　bit, R0L BSET　bit, R0H
BSET:G　bit, R1	BSET　bit, R1L BSET　bit, R1H
BSET:G　bit, A0	ビット位置０～７に対してアセンブル可能
BSET:G　bit, A1	ビット位置０～７に対してアセンブル可能
BSET:G　[A0]	BITINDEX.B　[A0] BSET　0, 0
BSET:G　[A1]	BITINDEX.B　[A1] BSET　0, 0
BSET:G　base:8[A0]	BITINDEX.B　[A0] BSET　0, base
BSET:G　base:16[A0]	BITINDEX.B　[A0] BSET　0, base
BSET:G　base:8[A1]	BITINDEX.B　[A1] BSET　0, base
BSET:G　base:16[A1]	BITINDEX.B　[A1] BSET　0, base
BSET:G　bit, base:8[SB] BSET:G　bit, base:11[SB] BSET:G　bit, base:16[SB]	BSET　bit, base[SB]
BSET:G　bit, base:8[FB]	BSET　bit, base[FB]
BSET:G　bit, base:16	BSET　bit, base

M16C/60 命令との互換性（２）

以下に示す命令については、コマンドオプション"-mode60"を使用しても命令の置き換えができませんので、直接ソースプログラムを変更する必要があります。

- MOVA　　src, R0
- MOVA　　src, R1
- MOVA　　src, R2
- MOVA　　src, R3

  src ＝ dsp[A0], dsp[A1], dsp[SB], dsp[FB], abs16
- JMPI.A　　A1A0
- JSRI.A　　A1A0
- PUSHC　　INTBL
- PUSHC　　INTBH
- POPC　　INTBL
- POPC　　INTBH
- MUL.W　　generic, A0
- MULU.W　　generic, A0

  generic ＝ R0, R1, R2, R3, A0, A1, [A0], [A1],
  dsp[SB], dsp[FB], dsp[A0], dsp[A1], abs16
- LDC
- STC
- LDE.B/W　　[A1A0], generic
- STE.B/W　　generic, [A1A0]

  generic ＝ R0L/R0, R0H/R1, R1L/R2, R1H/R3, A0, A1, [A0], [A1],
  dsp[SB], dsp[FB], dsp[A0], dsp[A1], abs16
- BSET:G　　bit, R2
- BSET:G　　bit, R3
- ビット操作命令のBCLR, BAND, BOR, BXOR, BNOT, BNAND, BNOR, BNXOR, BTST, BNTST, BTSTC, BTSTS, BMcndについて、"BSET:G　bit,R2 / R3" 命令と同様の記述のもの

削除されたアセンブルオプション

AS308 では以下のオプションが削除されています。

- **-M60, -M61, -M62, -M62E（グループ別対応に関するオプション）**
- **-A（ニーモニックのオペランド評価に関するオプション）**
- **-P（構造化記述命令に関するオプション）**

構造化記述に関するオプション

AS308 では構造化記述機能をサポートしていませんが、AS30(M16C/60)で構造化記述で開発されたプログラムをアセンブルするためのオプション "-mode60p" をサポートしています。

- -mode60p
  AS30 の構造化記述に付属している構造化記述プリプロセッサ（pre30）を起動した後に、コマンドオプション "-mode60" の処理を行います。(注)

（注）AS308 は構造化記述には対応していませんので、全ての構造化ソースを処理できるものではありません。したがって、M16C/80 に存在しないニーモニックに展開している構造化や動作が異なるニーモニックについては本オプションでは対応できませんので、ユーザー側でアセンブリソースを変換する必要があります。

第 4 章
プログラミングスタイル

4.1 ハードウェアの定義

この節では SFR 領域の定義とインクルードファイルの作成方法、RAM データ領域の確保、ROM データ領域の確保、セクションの定義について説明します。

4.1.1 ＳＦＲ領域の定義

SFR 領域の定義部分はインクルードファイルで作成すると便利です。SFR 領域の定義方法には以下の 2 通りがあります。

.EQU による定義

指示命令 ".EQU" を使った SFR 領域の定義例を図 4.1.1 に示します。

```
;-----------------------------------------------------------------------
;       M30800 SFR 定義ファイル
;-----------------------------------------------------------------------
PM0     .EQU    0004H       ;プロセッサモードレジスタ 0
PM1     .EQU    0005H       ;プロセッサモードレジスタ 1
CM0     .EQU    0006H       ;システムクロック制御レジスタ 0
CM1     .EQU    0007H       ;システムクロック制御レジスタ 1
WCR     .EQU    0008H       ;ウエイト制御レジスタ
AIER    .EQU    0009H       ;アドレス一致割り込み許可レジスタ
PRCR    .EQU    000AH       ;プロテクトレジスタ
DS      .EQU    000BH       ;外部データバス幅制御レジスタ
MCD     .EQU    000CH       ;メインクロック分周
;
WDTS    .EQU    000EH       ;監視タイマスタートレジスタ
WDC     .EQU    000FH       ;監視タイマ制御レジスタ
RMAD0   .EQU    0010H       ;アドレス一致割り込みレジスタ 0
RMAD1   .EQU    0014H       ;アドレス一致割り込みレジスタ 1
RMAD2   .EQU    0018H       ;アドレス一致割り込みレジスタ 2
RMAD3   .EQU    001CH       ;アドレス一致割り込みレジスタ 3
;
EIAD    .EQU    0020H       ;エミュレータ専用割り込みベクタテーブル
EITD    .EQU    0023H       ;エミュレータ割り込み識別レジスタ
EPRR    .EQU    0024H       ;エミュレータ用プロテクトレジスタ
;
ROA     .EQU    0030H       ;ROM 領域設定レジスタ
```

プロセッサモードレジスタ 0 の配置されているアドレスを定義する
以後各レジスタのアドレスを定義する

2 バイト以上のレジスタについてはレジスタの先頭アドレスを定義する

図 4.1.1 ".EQU" による SFR 領域定義例

.BLKB による定義

指示命令 ".BLKB" を使った SFR 領域の定義例を図 4.1.2 に示します。

```
;-----------------------------------------------------------------
;           M30800 SFR 定義ファイル
;-----------------------------------------------------------------
  .SECTION      SFR,DATA
  .ORG   000004H
PM0:            .BLKB       1       :プロセッサモードレジスタ 0
PM1:            .BLKB       1       ;プロセッサモードレジスタ 1
CM0:            .BLKB       1       ;システムクロック制御レジスタ 0
CM1:            .BLKB       1       ;システムクロック制御レジスタ 1
WCR:            .BLKB       1       ;ウエイト制御レジスタ
AIER:           .BLKB       1       ;アドレス一致割り込み許可レジスタ
PRCR:           .BLKB       1       ;プロテクトレジスタ
DS:             .BLKB       1       ;外部データバス幅制御レジスタ
MCD:            .BLKB       1       ;メインクロック分周レ
;
  .ORG   00000EH
WDTS:           .BLKB       1       ;監視タイマスタートレジスタ
WDC:            .BLKB       1       ;監視タイマ制御レジスタ
RMAD0:          .BLKA       1       ;アドレス一致割り込みレジスタ 0
                .BLKB       1       ;
RMAD1:          .BLKA       1       ;アドレス一致割り込みレジスタ 1
                .BLKB       1       ;
RMAD2:          .BLKA       1       ;アドレス一致割り込みレジスタ 2
                .BLKB       1       ;
RMAD3:          .BLKA       1       ;アドレス一致割り込みレジスタ 3
                .BLKB       1       ;
;
  .ORG   000020H
EIAD:           .BLKA       1       ;エミュレータ専用割り込みベクタテーブルレジスタ
EITD:           .BLKB       1       ;エミュレータ割り込み識別レジスタ
EPRR:           .BLKB       1       ;エミュレータ用プロテクトレジスタ
;
  .ORG   000030H
ROA:            .BLKB       1       ;ROM 領域設定レジスタ
DBA:            .BLKB       1       ;デバッグモニタ領域設定レジスタ
EXA0:           .BLKB       1       ;拡張領域設定レジスタ 0
EXA1:           .BLKB       1       ;拡張領域設定レジスタ 1
```

セクション名の宣言

プロセッサモードレジスタ 0 の配置されているアドレスに合わせて絶対アドレスを指定

プロセッサモードレジスタ 0 の配置されている領域分確保

ここで絶対アドレスを 00000EH 番地に指定しないと監視タイマスタートレジスタの領域がプロテクトレジスタの次、00000DH に設定されてしまうので注意!!

何も配置されていない部分の領域も確保

図 4.1.2　".BLKB" による SFR 領域定義例

インクルードファイルの作成

ソースプログラムを分割して作成する場合、SFR定義など複数のファイルで使用する部分はインクルードファイルにします。インクルードファイルには通常 ".INC" という拡張子を付けます。

作成上の注意

(1)インクルードファイル中に ".EQU" を使用する場合

".EQU"はシンボルに値を定義する命令です。SFR定義のようにアドレスを定義することもできます。ただし、領域確保命令ではないのでアドレスを定義する場合は、アドレスが重ならないように注意してください。".EQU" を使用したインクルードファイルは複数のファイルで読み込むことができます。

(2)インクルードファイル中に ".ORG" を使用する場合

".ORG"を使用したインクルードファイルを複数のファイルで読み込むとリンクエラーになります。これは ".ORG" によって絶対アドレスが指定されているため、アドレスが 2 重定義となります。

(3)インクルードファイル中に ".BLKB"、".BLKW"、".BLKA" を使用している場合

".BLKB"、".BLKW"、".BLKA" は領域確保命令です。".BLKB"、".BLKW"、".BLKA"を使用したインクルードファイルを複数のファイルで読み込むと領域を別々に確保することになります。インクルードファイルのシンボルをローカルで使用している場合はエラーではありませんが、グローバルで使用している場合は 2 重定義になります。

複数のファイルで共通領域として利用したい場合は、領域確保している部分は共有定義ファイルとし、ソースファイルの 1 つとしてリンクします。そして、シンボルのグローバル指定部分をインクルードファイルにします。

インクルードファイルの読み込み

インクルードファイルを読み込むときは指示命令".INCLUDE" を使います。読み込むファイル名はフルネームで指定してください。

例)

SFR 領域の定義を行ったインクルードファイル "M30800.INC ” を読み込む場合

```
.INCLUDE     M30800.INC
```

4.1.2　RAM データ領域の確保

領域確保は以下の指示命令を使用します。

.BLKB	・・・	1 バイトの領域確保（整数値）
.BLKW	・・・	2 バイトの領域確保（整数値）
.BLKA	・・・	3 バイトの領域確保（整数値）
.BLKL	・・・	4 バイトの領域確保（整数値）
.BLKF	・・・	4 バイトの領域確保（浮動小数点数）
.BLKD	・・・	8 バイトの領域確保（浮動小数点数）

ワーク領域の設定例

ワーク領域の設定例を図 4.1.3 に示します。

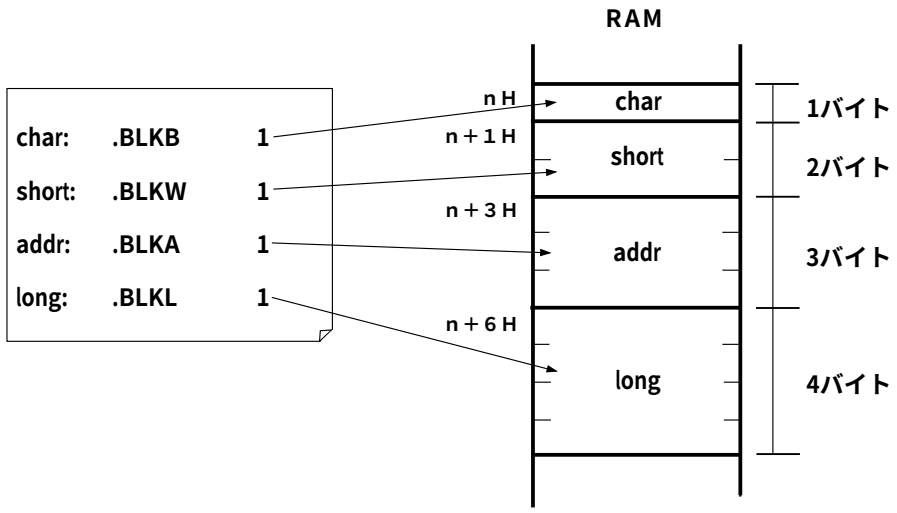


図 4.1.3　ワーク領域の設定例

4.1.3　ROM データ領域の確保

固定データの設定は以下の指示命令を使用します。記述例は「4.1.5　サンプルリスト 1(初期設定 1)」をご参照ください。

.BYTE　・・・　1 バイトデータの設定（整数値）
.WORD　・・・　2 バイトデータの設定（整数値）
.ADDR　・・・　3 バイトデータの設定（整数値）
.LWORD　・・・　4 バイトデータの設定（整数値）
.FLOAT　・・・　4 バイトデータの設定（浮動小数点数）
.DOUBLE　・・・　8 バイトデータの設定（浮動小数点数）

データテーブルの検索

データテーブルの例を図4.1.4に示します。このテーブルをアドレスレジスタ相対アドレッシングを用いてアクセスする方法を図 4.1.5 に示します。

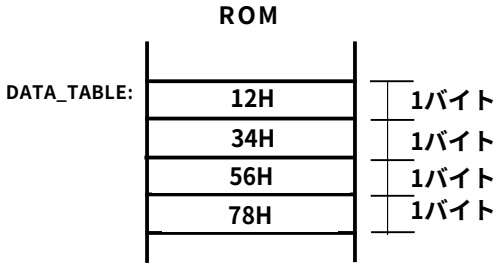


図 4.1.4　データテーブルの設定例

```
        :
        :
    MOV.W     #1, A0
    LDE.B     DATA_TABLE[A0], R0L    ;データテーブルの2バイト目(34H)を
                                     ;R0Lに格納
        :
        :
        :
        :
DATA_TABLE:
        .BYTE     12H,34H,56H,78H    ;1バイトデータの設定
        :
        :
        :
```

図 4.1.5　データテーブルの検索例

4.1.4　セクション定義

指示命令 ".SECTION" は、この指示命令を記述した行から新しいセクションを割り付ける宣言をします。

セクション定義の記述形式

.SECTION　セクション名［,（セクションタイプ）, ALIGN ］
［ ］内は省略可能

１つのセクション指示命令から次のセクション指示命令､又は".END"を記述した前の行までを１つのセクションとして定義します。セクション名は自由に設定できます。また、各セクションにはセクションタイプ（DATA,CODE,ROMDATA の３タイプ）が設定できます。このタイプによってセクション内に記述できる命令が異なりますのご注意ください。詳しくは AS308 ユーザーズマニュアルをご参照ください。
なお "ALIGN" 指定がある場合、リンカ(ln308)がセクションの始まりを偶数番地に割り当てます。

各セクションの設定例

セクションの設定例を図 4.1.6 に示します。

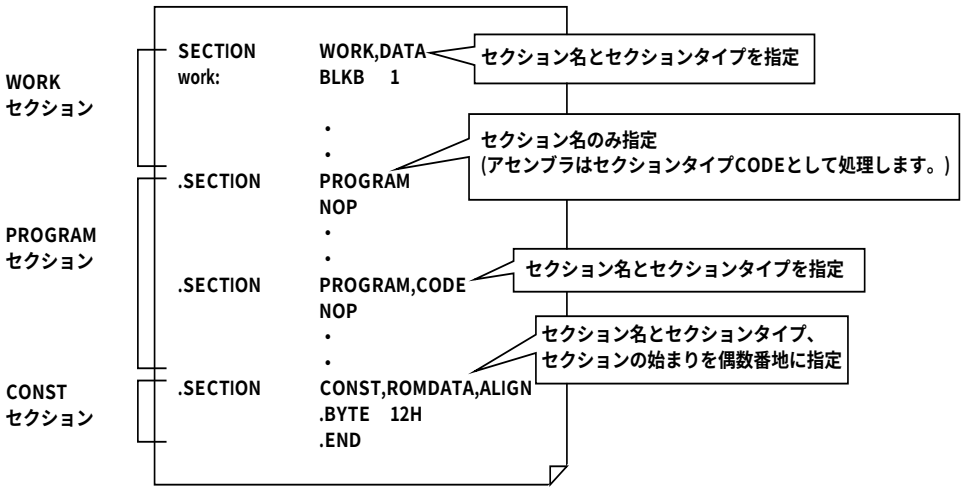


図 4.1.6　セクションの設定例

セクションの属性

各セクションはアセンブル時に属性が割り付けられます。属性は相対と絶対の２つです。

(1)相対属性

・リンク時に各セクションの配置を指定できます。（再配置可能）
・アセンブル時にセクション内のアドレスがリロケータブル値になります。
・相対属性セクション内で定義されたラベルの値はリロケータブルとなります。

(2)絶対属性

・SECTION" 指示命令の直後に ".ORG" によりアドレスを指定することで、そのセクションは絶対属性として扱われます。
・アセンブル時にセクション内のアドレスがアブソリュート値になります。
・絶対属性セクション内で定義されたラベルの値はアブソリュートとなります。

4.1.5 サンプルリスト 1（初期設定 1）

```
;********************** インクルード ************************************************
;
  .INCLUDE      M30800.INC
;
;********************** シンボル定義 ************************************************
;
RAM_TOP             .EQU    000400H             ;RAM の先頭アドレス
RAM_END             .EQU    002BFFH             ;RAM の終了アドレス
ROM_TOP             .EQU    0FE0000H            ;ROM の先頭アドレス
FIXED_VECT_TOP      .EQU    0FFFFDCH            ;固定ﾍﾞｸﾀの先頭アドレス
SB_BASE             .EQU    000400H             ;SB 相対のベースアドレス
FB_BASE             .EQU    000580H             ;FB 相対のベースアドレス
ISTACK_SIZE         .EQU    300H                ;割り込みスタック領域のサイズ
;
;********************** ワーク RAM 領域確保 *****************************************
;
  .SECTION      WORK,DATA
  .ORG          RAM_TOP
;
WORKRAM_TOP:
char:               .BLKB       1               ;1ﾊﾞｲﾄ領域確保
short:              .BLKW       1               ;2ﾊﾞｲﾄ領域確保
addr:               .BLKA       1               ;3ﾊﾞｲﾄ領域確保
long:               .BLKL       1               ;4ﾊﾞｲﾄ領域確保
WORKRAM_END:
;
;********************** ビットシンボル定義 ******************************************
;
char_b0             .BTEQU              0,char          ;char のビット 0
short_b1            .BTEQU              1,short         ;short のビット 1
addr_b2             .BTEQU              2,addr          ;addr のビット 2
long_b3             .BTEQU              3,long          ;long のビット 3
;
;********************** プログラム領域 **********************************************
;============== スタートアップ ======================================
;
  .SECTION      PROGRAM,CODE
  .ORG          ROM_TOP
START:
        LDC     #RAM_END+1,ISP          ;スタックポインタ(ISP)の初期値設定
;
        MOV.B       #03H,PRCR           ;プロテクトの解除
        MOV.W       #0183H,PM0          ;プロセッサモードレジスタ 0,1 の設定
        MOV.W       #2008H,CM0          ;システムクロック制御レジスタ 0,1 の設定
        MOV.B       #12H,MCD            ;メインクロック分周レジスタの設定
        MOV.B       #0,PRCR             ;全レジスタをプロテクト
;
```

インクルードファイルの読み込み

シンボルには ":"(コロン)を付けない

ラベルは名前の最後に ":"(コロン)を付ける

ハードウェアの RAM 領域に合わせる

ビットシンボルも ":"(コロン)を付けない

ハードウェア及びプログラミング上選択した内容に合わせる

ハードウェア及びプログラミング上選択した内容に合わせる

```
        MOV.W   #0,PS0              ;機能選択レジスタ A0,A1 の設定
        MOV.B   #0,PS2              ;機能選択レジスタ A2 の設定
        MOV.W   #0,PSL0             ;機能選択レジスタ B0,B1 の設定
        MOV.B   #0,PSL2             ;機能選択レジスタ B2 の設定
        MOV.B   #0,PSC              ;機能選択レジスタ C の設定
        BSET    2,PRCR              ;プロテクト解除
                                    ;(端子機能選択レジスタ A3 への書き込み許可)
        MOV.B   #0,PS3              ;機能選択レジスタ A3 の設定
;
        MOV.B   #03H,DS             ;外部データバス幅制御レジスタの設定
        MOV.B   #85H,WCR            ;ウエイト制御レジスタの設定
;
        LDC     #80H,FLG            ;フラグレジスタの初期値設定(U=1 に設定)
        LDC     #(RAM_END-ISTACK_SIZE)+1,SP     ;スタックポインタ(USP)の初期値設定
;
        MOV.W   #0FFF0H,PUR2        ;ポートP6～P10に内蔵プルアップ抵抗を接続
;
        MOV.B   #0FFH,03CBH         ;SFR 予約領域の初期設定
        MOV.W   #0FFFFH,03CEH
        MOV.W   #0FFFFH,03D2H
;
        .SB     SB_BASE             ;SB レジスタの値をアセンブラに対して宣言
        .FB     FB_BASE             ;FB レジスタの値をアセンブラに対して宣言
        LDC     #SB_BASE,SB         ;SB レジスタ初期値設定
        LDC     #FB_BASE,FB         ;FB レジスタ初期値設定
;
        MOV.W   #0,R0               ;WORK_RAM 0 クリア↓
        MOV.W   #(RAM_END+1 - RAM_TOP)/2,R3
        MOV.W   #WORKRAM_TOP,A1
        SSTR.W
;
;======================== メインプログラム ==================================
MAIN:
        MOV.B   DATA_TABLE[A0],R0L
        MOV.W   #1234H,R1
        BSET    char_b0
;               •
;               •
;               •
        JMP     MAIN
;
;====================== ダミー割り込みプログラム ============================
dummy:
        REIT
;
```

アセンブラに対する宣言

アセンブラに対して宣言した値と合わせる

```
;""Sample List""
;********** ｲﾝｸﾙｰﾄﾞ **********
;
   .INCLUDE     M30800.INC
;
********** ｼﾝﾎﾞﾙ定義 **********
;
RAM_TOP             .EQU   000400H              ;RAM の先頭ｱﾄﾞﾚｽ
RAM_END             .EQU   002BFFH              ;RAM の終了ｱﾄﾞﾚｽ
ROM_TOP             .EQU   0FE0000H         ;ROM の先頭ｱﾄﾞﾚｽ
FIXED_VECT_TOP      .EQU   0FFFFDCH         ;固定ﾍﾞｸﾀの先頭ｱﾄﾞﾚｽ
SB_BASE             .EQU   000400H              ;SB 相対のﾍﾞｰｽｱﾄﾞﾚｽ
FB_BASE             .EQU   000580H              ;FB 相対のﾍﾞｰｽｱﾄﾞﾚｽ
IPL_SIZE            .EQU   300H                 ;割り込みベクタ領域のサイズ
;
;********** ﾜｰｸRAM 領域確保 **********
;
   .SECTION     WORK,DATA
   .ORG         RAM_TOP
;
WORKRAM_TOP:
char:               .BLKB       1           ;1ﾊﾞｲﾄ領域確保
short:              .BLKW       1           ;2ﾊﾞｲﾄ領域確保
addr:               .BLKW       1           ;2ﾊﾞｲﾄ領域確保
long:               .BLKW       2           ;4ﾊﾞｲﾄ領域確保
WORKRAM_END:
;
;********** ﾋﾞｯﾄｼﾝﾎﾞﾙ定義 **********
;
char_b0             .BTEQU      0,char      ;char のﾋﾞｯﾄ0
short_b1            .BTEQU      1,short     ;short のﾋﾞｯﾄ1
addr_b2             .BTEQU      2,addr      ;addr のﾋﾞｯﾄ2
long_b3             .BTEQU      3,long      ;long のﾋﾞｯﾄ3
;
;**********      ﾌﾟﾛｸﾞﾗﾑ領域 **********
```

ハードウェアの ROM 領域に合わせる

固定ベクタの最下位番地から順に飛び先番地を設定する

使用しない割り込みの飛び先番地は、未使用の割り込み要求が発生した場合の暴走を防ぐために dummy 処理 (REIT 命令のみ)に飛び先番地を設定する

リセットベクトルにはプログラムの先頭番地を設定する
電源投入直後、またはリセット解除後にこのベクタに書き込まれた番地からプログラムがスタートする

図 4.1.7　初期設定記述例

4.2 CPU の初期設定

電源投入直後、またはリセット直後に各レジスタ、RAM 等の初期設定が必要です。CPU 内部のレジスタの未設定、及びプログラム実行前のメモリ等に意図しないデータが残っていた場合、プログラム暴走の原因になります。したがって初期設定はプログラムの最初におこないます。初期設定では以下の事項について設定します。

- **アセンブラに対する宣言**
- **CPU 内部のレジスタ、フラグ、RAM 領域の初期化**
- **ワーク領域の初期化**
- **ポート、タイマ、割り込み等の内蔵周辺機能の初期化**

4.2.1 CPU 内部レジスタの設定

通常はリセット解除後プロセッサの動作モードやシステムのクロック、及びポート機能に関するレジスタの設定を行なう必要があります。

プロセッサモードとシステムクロックの設定

プロセッサモードレジスタ 0,1 とシステムクロック制御レジスタ 0,1、及びメインクロック分周レジスタはプロテクト対象のレジスタであるため、プロテクトを解除した後に設定を行い、設定終了後再びプロテクトをかけるようにしてください。図 4.2.1 に設定例を示します。

```
;---------- プロセッサモードとシステムクロックの設定 ------------------------------
;
        MOV.B     #03H, PRCR        ;プロテクトの解除
        MOV.W     #0183H, PM0       ;プロセッサモードレジスタ 0,1 の設定
        MOV.W     #2008H, CM0       ;システムクロック制御レジスタ 0,1 の設定
        MOV.B     #12H, MCD         ;メインクロック分周レジスタの設定
        MOV.B     #0, PRCR          ;全レジスタをプロテクト
;
```

図 4.2.1　プロセッサモードとシステムクロックの設定例

ポート機能の設定

M16C/80 では、端子出力機能がポート出力と周辺機能出力との多重になっている場合、または 1 端子に複数の周辺機能出力が割り付けられている場合に、どの端子出力機能として使用するかを機能選択レジスタにより選択する必要があります。　図 4.2.2 に機能選択レジスタの設定例を示します。

```
;---------- 機能選択レジスタの設定 ----------------------------------------
;
        MOV.W     #0, PS0          ;機能選択レジスタ A0,A1 の設定
        MOV.B     #0, PS2          ;機能選択レジスタ A2 の設定
        MOV.W     #0, PSL0         ;機能選択レジスタ B0,B1 の設定
        MOV.B     #0, PSL2         ;機能選択レジスタ B2 の設定
        MOV.B     #0, PSC          ;機能選択レジスタ C の設定
        BSET      2, PRCR          ;プロテクト解除
                                   ;(端子機能選択レジスタ A3 への書き込み許可)
        MOV.B     #0, PS3          ;機能選択レジスタ A3 の設定
;
```

図 4.2.2　機能選択レジスタの設定例

4.2.2　スタックポインタの設定

サブルーチンや割り込みを使用する場合、戻り先番地などがスタックに退避されます。よってサブルーチンコール前、あるいは割り込み許可前にスタックポインタの設定をおこなっておく必要があります。設定例は「4.2.6 サンプルリスト 2(初期設定 2)」をご参照ください。

4.2.3　ベースレジスタ（SB,FB）の設定

M16C/80シリーズでは効率のよいデータのアクセスを行なうためにベースレジスタ相対アドレッシングというアドレッシングモードがあります。これはベースとなるアドレスからの相対番地でアクセスしますので、このアドレッシングモードを使用するにはベースアドレスをあらかじめ設定しておく必要があります。設定例は「4.2.6 サンプルリスト 2(初期設定 2)」をご参照ください。

4.2.4　固定割り込みベクタ（リセットベクタ）の設定

M16C/80シリーズには可変ベクタと固定ベクタの２種類のベクタが存在します。リセットベクタを含めた固定割り込みベクタの設定方法については「4.2.6　サンプルリスト 2(初期設定 2)をご参照ください。

4.2.5　内蔵周辺機能の設定

M16C/80グループが内蔵しているRAM、ポート、タイマ、DMAコントローラの初期設定について説明します。詳しくはご使用品種のユーザーズマニュアルの機能説明をご参照ください。

ワーク領域の初期設定

通常ワーク領域は初期設定で 0 クリアします。初期値が 0 でない場合、各ワーク領域に初期値を設定してください。図 4.2.3 にワーク領域の 0 クリアと初期値の設定例を示します。

```
;------------ ストリング命令によるワーク RAM の 0 クリア -------------------------
RAM_TOP         .EQU            0400H
RAM_END         .EQU            2BFFH
;
                MOV.W           #0, R0
                MOV.W           #(RAM_END+1 - RAM_TOP) / 2, R3
                MOV.W           #WORKRAM_TOP, A1
                SSTR.W                                  ;WARKRAM_TOP から
                                                        ;(RAM_END+1 - RAM_TOP) / 2回分 '0' を転送
;
;------------- ワーク RAM への初期値設定 ------------------------------------------------
                MOV.B           #0FFH, char             ;１バイトのデータ設定
;
                MOV.W           #0FFFFH, short          ;１ワードのデータ設定
;
                MOV.W           #0FFFFH, addr           ;３バイトのデータ設定
                MOV.B           #0FFH, addr+2
;
                MOV.L           #0FFFFFFFFH, long       ;１ロングワードのデータ設定
```

図 4.2.3　ワーク領域の初期設定例

ポートの初期設定

各ポートは、ポート方向レジスタを出力に設定した時点で、ポートからデータが出力されます。不定なデータを出力させないためには、方向レジスタを出力に設定する前に出力ポートに初期値を設定してください。図4.2.4にポートの初期設定例を示します。

```
;---------- ポートの初期値設定 ----------
;
        MOV.W     #0FFFFH,P6        ;ポート P6,P7 に初期値設定
        MOV.W     #0FFFFH,PD6       ;ポート P6,P7 を出力に設定
        MOV.B     #3CH,P9           ;ポート P9 に初期値を設定
;
        MOV.B     #04H,PRCR         ;プロテクト解除 (注1)
        MOV.W     #0FFH,PD9         ;ポート P9 を出力に設定 (注2)
;
```

図 4.2.4　ポートの初期設定例

タイマの設定

タイマ等の内蔵周辺機能を使用する場合、関連レジスタ(SFR 領域)に対する初期設定　を行います。図4.2.5にタイマ A0 の設定例を示します。

```
;---------- タイマ A0 の設定 ----------
;
TA0R    .BTEQU    3, TA0IC
TA0S    .BTEQU    0, TABSR
;
        MOV.B     #01000000B, TA0MR ;タイマ A0 モードレジスタ設定
                                    ;(モード:タイマモード , 分周比:1/8)
        MOV.W     #2500-1, TA0      ;タイマ A0 のカウント値設定
        BCLR      TA0R              ;タイマ A0 割り込み要求ビットクリア
;
        BSET      TA0S              ;タイマ A0 カウントスタート
```

図 4.2.5　タイマの設定例

注 1)ポート P9 方向レジスタはプロテクト対象のレジスタですので、プロテクトレジスタのビット 2 を '1' にして、プロテクトを解除してから値を設定します。
注 2)ポート P9 方向レジスタへの書き込み許可ビット(プロテクトレジスタのビット 2)は、許可状態にした次の " 書き込み命令 " で "0" になりますので、入出力の変更は必ずポート P9 方向レジスタへの書き込み許可ビットを '1' にセットした命令の直後に行ってください。また、その際割り込みの発生及び DMA の転送が行われないようにしてください。

DMA コントローラの設定

DMAC を使用する場合、関連レジスタ(CPU 内部レジスタ及び SFR 領域)に対する初期設定を行います。 図 4.2.6 に DMAC 関連レジスタを示します。

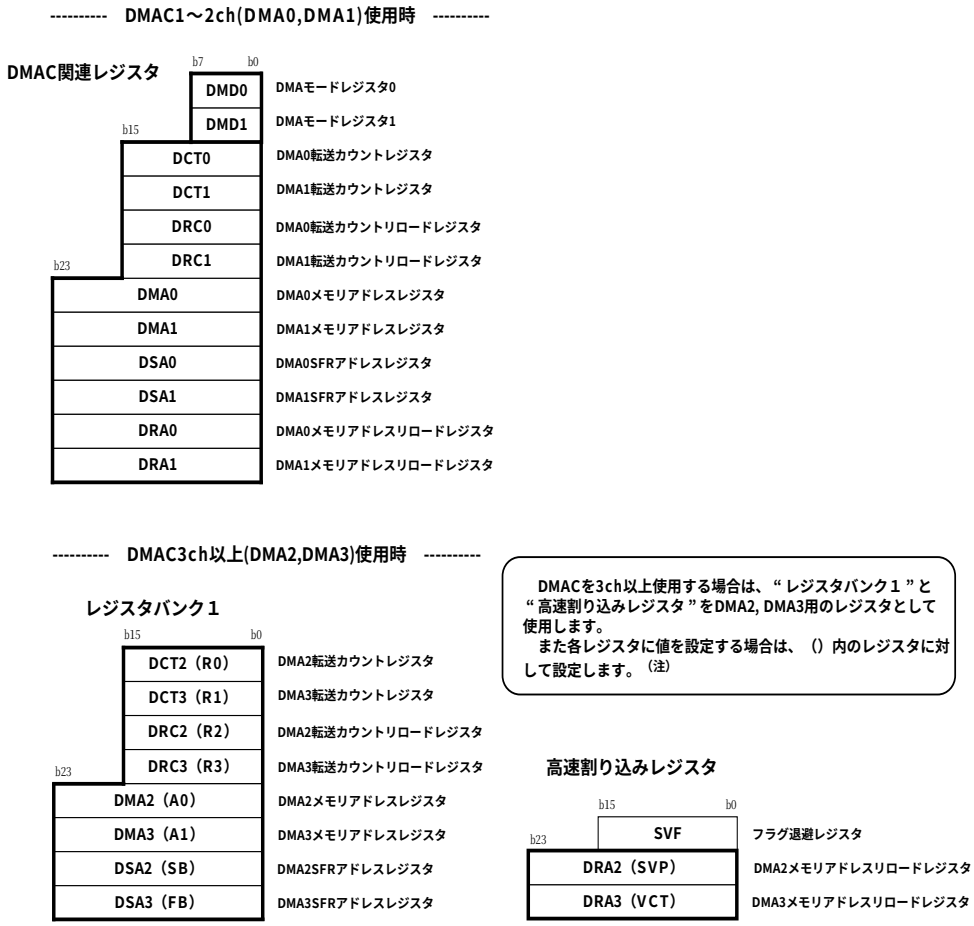


図 4.2.6DMAC 関連レジスタ

(注)DMA2,DMA3 関連レジスタを設定する場合、必ずフラグレジスタ(FLG)のレジスタバンク指定フラグ(B)を "1" にしてから設定してください。

(注)DMA2,DMA3 使用時は高速割り込み、及び割り込み処理ルーチン内でのレジスタバンク切り替えによるレジスタの退避／復帰は使用できません。

DMA コントローラ 1 ～ 2ch(DMA0,1)使用時の設定

DMAC を 1 ～ 2ch(DMA0,1)使用する場合の関連レジスタ(CPU 内部レジスタ及び SFR 領域)に対する設定例を以下に示します。

```
;---------------------- DMA0 の設定 ---------------------------------------
;
        MOV.B       #00000011B, DM0SL   ;DMA0 要求要因の設定
        LDC     #32, DRC0               ;DMA0 転送カウントリロードレジスタに
                                        ;転送回数を設定
        LDC     #32, DCT0               ;DMA0 転送カウントレジスタに転送回数
                                        ;を設定
        LDC     #0FF0000H, DRA0         ;DMA0 メモリアドレスリロードレジスタ
                                        ;に転送元アドレス(メモリ)を設定
        LDC     #0FF0000H, DMA0         ;DMA0 メモリアドレスレジスタに転送元
                                        ;アドレス(メモリ)を設定
        LDC     #P6, DSA0               ;DMA0 SFR アドレスレジスタに転送先
                                        ;アドレス(SFR)を設定
        LDC     #00001111B, DMD0        ;DMA モードレジスタ 0 の設定、及び DMA
                                        ;転送の許可
                                        ;転送単位：16 ビット、
                                        ;転送方向：順方向(メモリ)→固定(SFR)
                                        ;転送モード：リピート転送（DMA0 の許可）
```

図 4.2.7　DMA コントローラの設定例 1

DMA コントローラ 3ch 以上(DMA2,3)使用時の設定

DMACを3ch以上(DMA2.3)使用する場合の"DMA2"関連レジスタ(CPU内部レジスタ及びSFR領域)に対する設定例を以下に示します。

```
;---------------- 3ch 目以降(DMA2 または DMA3)の DMAC の設定 -------------------------
;
        FSET        B                   ; レジスタバンクを "1" に設定
;
        MOV.B       #00001111B, DM2SL   ;DMA2 要求要因の設定
        MOV.W       #16, R2             ;DMA2 転送カウントリロードレジスタ(R2)
                                        ;に転送回数を設定
        MOV.W       #16, R0             ;DMA2 転送カウントレジスタ(R0)に転送
                                        ;回数を設定
        LDC         #U0RB, SB           ;DMA2 SFR アドレスレジスタ(SB)に転送
                                        ;元アドレス(SFR)を設定
        LDC         #0500H, SVP         ;DMA2 メモリアドレスリロードレジスタ
                                        ;(SVP)に転送先アドレス(メモリ)を設定
        MOV.L       #0500H, A0          ;DMA2 メモリアドレスレジスタ(A0)に転送
                                        ;先アドレス(メモリ)を設定
        FCLR        B                   ; レジスタバンクを "0" に戻す
;
        LDC     #00000111B, DMD1        ;DMA モードレジスタ 1 の設定、及び DMA
                                        ;転送の許可
                                        ;転送単位：16 ビット、
                                        ;転送方向：固定(SFR)→順方向(メモリ)
                                        ;転送モード：リピート転送（DMA2 の許可）
```

図 4.2.8　DMA コントローラの設定例 2

(注)DMACを2ch以下で使用する場合は、極力DMA0、DMA1を使用するようにしてください。DMA2、DMA3を使用するとレジスタバンク１及び高速割り込みが使用できなくなります。

4.2.6　サンプルリスト 2(初期設定 2)

```
;********************** インクルード ***********************************************
;
  .INCLUDE          M30800.INC
;
;********************** シンボル定義 ***********************************************
;
RAM_TOP             .EQU   000400H      ; RAM の先頭アドレス
RAM_END             .EQU   002BFFH      ; RAM の終了アドレス
ROM_TOP             .EQU   0FE0000H     ; ROM の先頭アドレス
FIXED_VECT_TOP      .EQU   0FFFFDCH     ;固定ベクタの先頭アドレス
SB_BASE             .EQU   00400H       ; SB 相対のベースアドレス
FB_BASE             .EQU   00580H       ; FB 相対のベースアドレス
ISTACK_SIZE         .EQU   300H         ;割り込みスタック領域のサイズ
;
;********************** ワーク RAM 領域確保 ***************************************
;
  .SECTION      WORK,DATA
  .ORG          RAM_TOP
WORKRAM_TOP:
WORK_1:         .BLKB         1
WORK_2:         .BLKB         1
WORKRAM_END:
;
;********************** プログラム領域 *********************************************
;================== スタートアップ ======================
;
  .SECTION      PROGRAM,CODE              ;セクション名、セ
  .ORG          ROM_TOP                   ;開始アドレスの宣言
START:
        LDC     #RAM_END+1,ISP            ;スタックポインタ(ISP)の初期値設定
;
        MOV.B       #03H,PRCR             ;プロテクトの解除
        MOV.W       #0183H,PM0            ;プロセッサモードレジスタ 0,1 の設定
        MOV.W       #2008H,CM0            ;システムクロック制御レジスタ 0,1 の設定
        MOV.B       #12H,MCD              ;メインクロック分周レジスタの設定
        MOV.B       #0,PRCR               ;全レジスタをプロテクト
;
        MOV.W       #0,PS0          ;機能選択レジスタ A0,A
        MOV.B       #0,PS2          ;機能選択レジスタ A2 の
        MOV.W       #0,PSL0         ;機能選択レジスタ B0,B
        MOV.B       #0,PSL2         ;機能選択レジスタ B2 の設定
        MOV.B       #0,PSC          ;機能選択レジスタ C の設定
        BSET        2,PRCR          ;プロテクト解除
                                    ;(端子機能選択レジスタ A3 への書き込み許可)
        MOV.B       #0,PS3          ;機能選択レジスタ A3 の設定
;
```

リセット解除後に誤って NMI 割り込み(ノンマスカブル)が発生した場合に、プログラムの暴走を防ぐため、スタートアップの先頭で割り込みスタックポインタ（ISP）を設定する

プロセッサモードレジスタなどプロテクト対象になっているレジスタについては、プロテクトを解除した後に設定を行い、設定終了後再びプロテクトをかける

“端子出力機能選択レジスタ３ ”はプロテクト対象のレジスタのため、プロテクト解除後に設定する。
また、端子出力機能選択レジスタ３への書き込み許可ビット(プロテクトレジスタのビット3)については、許可状態にした次の書き込み令命で"0"になるため、初期値の設定は、必ず端子出力機能選択レジスタ３への書き込み許可ビットを "1" にセットする命令の直後に行うこと。

```
        MOV.B   #03H,DS             ;外部データバス幅制御レジスタの設定
        MOV.B   #85H,WCR            ;ウエイト制御レジスタの設定

        LDC     #80H,FLG            ;フラグレジスタの初期値設定(U=1 に設定)
        LDC     #(RAM_END-ISTACK_SIZE)+1,SP     ;スタックポインタ(USP)の初期値設定

        MOV.W   #003FFH,PUR2        ;ポートP6～P10に内蔵プルアップ抵抗を接続

        MOV.B   #0FFH,03CBH         ;SFR 予約領域の初期設定
        MOV.W   #0FFFFH,03CEH
        MOV.W   #0FFFFH,03D2H
;
.SB     SB_BASE                     ;SB レジスタの値をアセンブラに対して宣言
.FB     FB_BASE                     ;FB レジスタの値をアセンブラに対して宣言
        LDC     #SB_BASE,SB         ;SB レジスタ初期値設定
        LDC     #FB_BASE,FB         ;FB レジスタ初期値設定
;
        MOV.W   #0,R0               ;WORK_RAM 0 クリア↓
        MOV.W   #(RAM_END+1 - RAM_TOP)/2,R3
        MOV.W   #WORKRAM_TOP,A1
        SSTR.W
;================ メインプログラム ====================================
MAIN:
        JSR     INIT                ;ワーク RAM の初期値設定
MAIN_10:
        BTST    TA0R                ;TA0 割り込み要求フラグの判定
        JNC     MAIN_10
        BCLR    TA0R
        JSR     SUB_TA0             ;タイマ A0 処理
;               ・
;               ・
        JMP     MAIN_10
;
;================ INIT ルーチン ======================================
INIT:
;-------------------- ワーク RAM とポートの初期値設定 --------------------------
;
        MOV.B   #0FFH,WORK_1
        MOV.B   #0FFH,WORK_2
        MOV.B   #0FFH,P6
        MOV.B   #0FFH,PD6
;
;-------------------- タイマ A0 の設定 -----------------------------------
;
        MOV.B   #01000000B,TA0MR    ;タイマ A0 モードレジスタ設定
        MOV.W   #2500-1,TA0         ;タイマ A0 カウント値設定
        BCLR    TA0R                ;タイマ A0 割り込み要求ビットのクリア
        BSET    TA0S                ;タイマ A0 カウントスタート
```

SFR 領域の 03C9H、03CBH～03D3H 番地については、将来展開予定の製品のために用意している領域です。
03CBH、03CEH、03CFH、03D2H、03D3H 番地には必ず"FFH"を初期設定してください。

必ず整合性をとること

RAM アクセス（SB,FB 相対アドレッシング使用）前に、SB,FB レジスタを設定する

タイマに関するレジスタの初期値を設定した後に、カウント開始フラグをセットする

```
;---------- DMA0 の設定 ----------
        MOV.B       #00000011B, DM0SL   ;DMA0 要求要因の設定
        LDC     #32, DRC0               ;DMA0 転送カウントリロードレジスタに
                                        ;転送回数を設定
        LDC     #32, DCT0               ;DMA0 転送カウントレジスタに転送回数
                                        ;を設定
        LDC     #0FF0000H, DRA0         ;DMA0 メモリアドレスリロードレジスタ
                                        ;に転送元アドレス(メモリ)を設定
        LDC     #0FF0000H, DMA0         ;DMA0 メモリアドレスレジスタに転送元
                                        ;アドレス(メモリ)を設定
        LDC     #P6, DSA0               ;DMA0 SFR アドレスレジスタに転送先
                                        ;アドレス(SFR)を設定
        LDC     #00001111B, DMD0        ;DMA モードレジスタ 0 の設定、及び DMA
                                        ;転送の許可
                                        ;転送単位：16 ビット、
                                        ;転送方向：順方向(メモリ)→固定(SFR)
                                        ;転送モード：リピート転送（DMA0 の許可）
INIT_END:
        RTS
;
;================ SUB_TA0 ルーチン ==================================
SUB_TA0:
        MOV.B       WORK_1,R0L
        INC.B       R0L
;               •
;               •
SUB_TA0_END:
        RTS
;
;=============== ダミー割り込みプログラム ============================
dummy:
        REIT
;
;********************** 固定ベクタの設定 ***************************************
  .SECTION      F_VECT,ROMDATA
  .ORG          FIXED_VECT_TOP
;
  .LWORD        dummy               ;未定義命令割り込みベクタ
  .LWORD        dummy               ;オーバーフロー(INTO 命令)割り込みベクタ
  .LWORD        dummy               ;BRK 命令割り込みベクタ
  .LWORD        dummy               ;アドレス一致割り込みベクタ
  .LWORD        dummy               ;未使用
  .LWORD        dummy               ;監視タイマ割り込みベクタ
  .LWORD        dummy               ;未使用
  .LWORD        dummy               ;NMI 割り込みベクタ
  .LWORD        START               ;リセットベクタの設定
  .END
```

DMA に関するレジスタを全て設定した後に、DMAを許可（チャネル転送モード選択ビットを設定）する

図 4.2.9　初期設定の記述例 2

4.3 割り込み使用時の設定

ここでは割り込み処理プログラムを実行する場合に必要な処理、記述方法、及び多重割り込みの実行方法に関する説明をします。

M16C/80 シリーズで割り込みを発生させる条件として、以下の 3 条件が全て揃っている必要があります。

・割り込み許可フラグ（I）＝ 1（許可状態）
・IPL ＜ 各割り込みのソフトウエア割り込み優先レベル
・使用する割り込みの割り込み要求ビット＝ 1（割り込み要求あり）

上記の 3 条件を含め、割り込み処理プログラムを実行する場合、以下の処理が必要となります。

(1) 割り込みテーブルレジスタ（INTB）の設定
(2) 可変／固定ベクタの設定
(3) 割り込み制御レジスタの設定
(4) 割り込み許可フラグ（I）の許可
(5) 割り込み処理ルーチン内でのレジスタの退避と復帰

4.3.1 割り込みテーブルレジスタ（INTB）の設定

M16C/80 シリーズでは、内蔵周辺機能による割り込みのベクタテーブルが可変となっているため、割り込みを使用する前に割り込みテーブルレジスタ(INTB)によってベクタの先頭アドレスを設定する必要があります。

割り込みテーブルレジスタで指定された番地から 256 バイトが可変ベクタ領域になり、1 ベクタ 4 バイトとなります。各ベクタにはソフトウェア割り込み番号が割り付けられており、0 ～ 63 までの 64 ベクタとなります。

設定例は「4.3.6 サンプルリスト 3（割り込みの使用）」をご参照ください。

4.3.2　可変／固定ベクタの設定

割り込みが発生すると割り込み要因ごとに設定された番地にジャンプします。この飛び先番地を設定する部分を割り込みベクタと呼びます。

割り込みベクタの設定は、各割り込み処理プログラムの先頭番地を可変／固定ベクタテーブルにそれぞれ登録します。実際の登録例として「4.3.6　サンプルリスト 3(割り込みの使用)」をご参照ください。

可変ベクタテーブル

可変ベクタテーブルは、割り込みテーブルレジスタ(INTB)によって指し示された値を先頭アドレスとする 256 バイトの割り込みベクタテーブルで、ベクタテーブルはSFR領域を除いたメモリ空間のどこにでも配置が可能です。また、1 ベクタは 4 バイトで構成され、ベクタごとに 0 ～ 63 までのソフトウェア割り込み番号が割り付けられます。

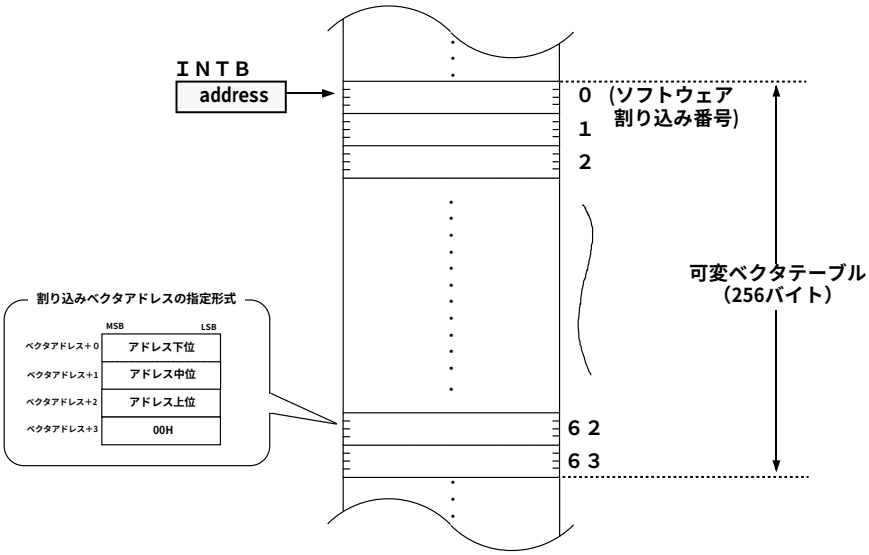


図 4.3.1　可変ベクタテーブル

4.3.3　割り込み制御レジスタの設定

各割り込み制御レジスタのビット 0 ～ビット 2 で、各割り込みの割り込み優先レベルを設定することができます。レベル =0 の時は割り込み禁止と同じ状態になりますので、"1" 以上に設定してください。　また、割り込み制御レジスタのビット 3 が割り込み要求フラグになります。リセット解除後は "0" になりますが、外部端子割り込みなどについては "1" にセットされている可能性もありますので、誤動作を防ぐためにも割り込み許可フラグ (I フラグ) を許可する前に、割り込み要求ビットを "0" にクリアしてください。　設定例は「4.3.6　サンプルリスト 3（割り込みの使用）」をご参照ください。

各割り込み制御レジスタのビット構成や優先レベル等についてはご使用品種のユーザーズマニュアルをご参照ください。

4.3.4　割り込み許可フラグ（I）の許可

電源投入直後、及びリセット解除後は割り込みは禁止状態になっていますので、プログラム中で許可にする必要があります。フラグレジスタ（FLG）中の I フラグを "1" にセットすることで割り込み許可状態にすることができます。

なお、I フラグを "1" にセットすると同時に割り込みが許可されるため、プログラムの先頭で許可すると暴走の原因となりますので、必ず初期設定を行った後に I フラグを許可してください。

4.3.5　割り込み処理ルーチン内でのレジスタの退避と復帰

割り込みが受け付けられると、以下のものが自動的にスタックに退避されます。退避順序、スタックの動作などの詳細については「4.5.3　スタックポインタの設定」をご参照ください。

- **PC(プログラムカウンタ)の内容**
- **FLG(フラグレジスタ)の内容**

割り込み処理プログラムからの復帰は、必ず"REIT命令"で行ってください。この命令によって割り込み処理終了後、スタックに退避されていたレジスタ、戻り先番地などが復帰され、割り込み発生前の処理を継続します。

自動的に退避されるレジスタ以外で、割り込み処理ルーチン内で変更する可能性のあるレジスタ（割り込み処理内で使用するレジスタ）については、ソフトウェアによってスタックに退避します。割り込み処理ルーチン内でのレジスタの退避／復帰方法の例として「図 4.3.2 または図 4.3.3」をご参照ください。

レジスタの退避と復帰の種類

レジスタの退避／復帰方法には、以下に示すように２通りの方法があります。

(1) プッシュ／ポップ命令による退避と復帰

(1a)個別退避

```
PUSH.B    R0L
PUSH.W    R1
```

(1b)個別復帰

```
POP.B     R0L
POP.W     R1
```

(2a)一括退避

```
PUSHM     R0,R1,R2,R3,A0,A1
```

(2b)一括復帰

```
POPM      R0,R1,R2,R3,A0,A1
```

(2) レジスタバンク切り替えによる退避と復帰

割り込み処理のオーバーヘッドタイムを短くしたい場合に有効です。

(a)レジスタバンク 1 を使用

```
FSET      B
```

(b)レジスタバンク 0 を使用

```
FCLR      B
```

割り込み処理ルーチンの記述（１）

図 4.3.2 に割り込み処理内での通常のレジスタの退避／復帰方法の例を示します。

```
****************** レジスタの個別退避と個別復帰 ************************************
INT_A0:
        PUSH.B    R0L                ;R0L を退避
        PUSH.B    R1L                ;R1L を退避
        PUSH.W    R2                 ;R2 を退避
        ・
        ・
        割り込み処理
        ・
        ・
        POP.W     R2                 ;R2 を復帰
        POP.B     R1L                ;R1L を復帰
        POP.B     R0L                ;R0L を復帰
;
        REIT                         ;割り込みからの復帰
```

個別で退避した場合は、復帰するときに退避した時と逆の順番で復帰させること

```
****************** レジスタの一括退避と一括復帰 ************************************
INT_A1:
        PUSHM     R0,R2,R3,A1        ;R0,R2,R3,A1 レジスタを一括して退避
        ・
        ・
        割り込み処理
        ・
        ・
        POPM      R0,R2,R3,A1        ;R0,R2,R3,A1 レジスタを一括して復帰
;
        REIT                         ;割り込みからの復帰
```

図 4.3.2　プッシュ／ポップ命令によるレジスタの退避と復帰

割り込み処理ルーチンの記述（２）

高速な割り込み応答をさせたい場合は、以下に示す " レジスタバンク切り替え " により行ってください。
レジスタ（R0,R1,R2,R3,A0,A1,SB,FB）を "FSET　B" または "FCLR　B" １命令（実行サイクル数：1サイクル）で退避／復帰できます。

```
****************** レジスタバンク切り替えによるレジスタの退避と復帰 ******************
INT_A2:
        FSET   B                        ;レジスタバンクを 0 → 1 に切り替え
        ・
        ・
割り込み処理
        ・
        ・
;
        REIT                            ;割り込みからの復帰
```

REIT命令により退避されていたFLGの内容が復帰されるため、レジスタバンクも1から0に復帰する

割り込みプログラム内では、バンク１の各レジスタ(R0,R1,R2,R3,A0,A1,SB,FB)を使用することになる

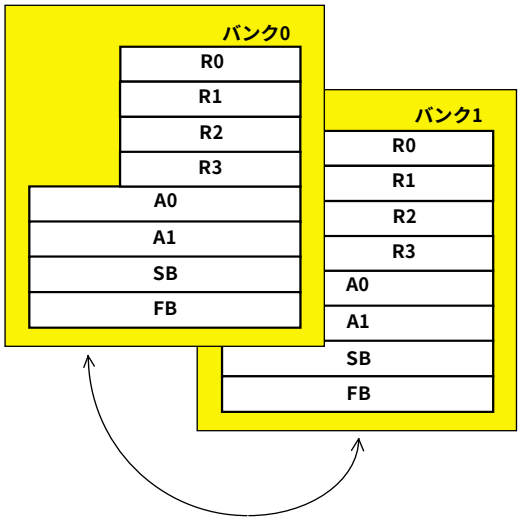


図 4.3.3　レジスタバンク切り替えによるレジスタの退避と復帰

(注)メインプログラム内でレジスタバンク0、1両方を使用している場合はレジスタバンク切り替えによるレジスタの退避と復帰はできません。

4.3.6 サンプルリスト 3（割り込みの使用）

```
;********************** インクルード ************************************************
;
  .INCLUDE      M30800.INC
;
;********************** シンボル定義 ************************************************
;
RAM_TOP             .EQU   000400H      ;RAM の先頭アドレス
RAM_END             .EQU   002BFFH      ;RAM の終了アドレス
ROM_TOP             .EQU   0FE0000H     ;ROM の先頭アドレス
VECT_TOP            .EQU   0FFFD00H     ;可変ベクタの先頭アドレス
FIXED_VECT_TOP      .EQU   0FFFFDCH     ;固定ベクタの先頭アドレス
SB_BASE             .EQU   000400H      ;SB 相対のベースアドレス
FB_BASE             .EQU   000580H      ;FB 相対のベースアドレス
ISTACK_SIZE         .EQU   300H         ;割り込みスタック領域のサイズ
;
;********************** ワーク RAM 領域確保 ***************************************
;
  .SECTION      WORK,DATA
  .ORG          RAM_TOP
WORKRAM_TOP:
WORK_1:         .BLKW         1
ANS_L:          .BLKW         1
ANS_H:          .BLKW         1
WORKRAM_END:
;
;********************** プログラム領域 ********************************************
;
  .SECTION      PROGRAM,CODE            ;セクション名、セクションタイプの宣言
  .ORG          ROM_TOP                 ;開始アドレスの宣言
START:
        LDC     #RAM_END+1,ISP          ;スタックポインタ(ISP)の初期値設定
;
        MOV.B       #03H,PRCR           ;プロテクトの解除
        MOV.W       #0183H,PM0          ;プロセッサモードレジスタ 0,1 の設定
        MOV.W       #2008H,CM0          ;システムクロック制御レジスタ 0,1 の設定
        MOV.B       #12H,MCD            ;メインクロック分周レジスタの設定
        MOV.B       #0,PRCR             ;全レジスタをプロテクト
;
        MOV.W       #0,PS0              ;機能選択レジスタ A0,A1 の設定
        MOV.B       #0,PS2              ;機能選択レジスタ A2 の設定
        MOV.W       #0,PSL0             ;機能選択レジスタ B0,B1 の設定
        MOV.B       #0,PSL2             ;機能選択レジスタ B2 の設定
        MOV.B       #0,PSC              ;機能選択レジスタ C の設定
        BSET        2,PRCR              ;プロテクト解除
                                        ;(端子機能選択レジスタ A3 への書き込み許可)
        MOV.B       #0,PS3              ;機能選択レジスタ A3 の設定
```

```
;
        MOV.B       #03H,DS             ;外部データバス幅制御レジスタの設定
        MOV.B       #85H,WCR            ;ウエイト制御レジスタの設定
;
        LDC     #80H,FLG                ;フラグレジスタの初期値設定(U=1 に設定)
        LDC     #(RAM_END-ISTACK_SIZE)+1,SP     ;スタックポインタ(USP)の初期値設定
        LDC     #VECT_TOP,INTB          ;割り込みテーブルレジスタの初期値設定
;
        MOV.W       #003FFH,PUR2        ;ポート P6 ～ P10 に内蔵プルアップ抵抗を接続
;
        MOV.B       #0FFH,03CBH         ;SFR 予約領域の初期設定
        MOV.W       #0FFFFH,03CEH
        MOV.W       #0FFFFH,03D2H
;
  .SB   SB_BASE                         ;SB レジスタの値をアセンブラに対して宣言
  .FB   FB_BASE                         ;FB レジスタの値をアセンブラに対して宣言
        LDC     #SB_BASE,SB             ;SB レジスタ初期値設定
        LDC     #FB_BASE,FB             ;FB レジスタ初期値設定
;
        MOV.W       #0,R0               ;WORK_RAM 0 クリア↓
        MOV.W       #(RAM_END+1 - RAM_TOP)/2,R3
        MOV.W       #WORKRAM_TOP,A1
        SSTR.W
;
;================== メインプログラム ==============================
MAIN:
        JSR         INIT                ;ワーク RAM の初期値設定
        FSET        I                   ;割り込み許可
MAIN_10:                                                    初期設定を行った後割り込みを許可する
        MOV.W       #5,WORK_1
        MULU.W      WORK_1,ANS_L
;               •
;               •
;               •
        JMP         MAIN_10
;
;================ INIT ルーチン ============================================
INIT:
        MOV.W       #0,WORK_1
        MOV.W       #0,ANS_L
        MOV.W       #0,ANS_H
                                                割り込みを発生させるには優先レベルを"1"以上に設定する
        MOV.B       #01000000B,TA0MR    ;タイマ A0 モードレジスタ設定
        MOV.W       #2500-1,TA0         ;タイマ A0 カウント値設定
        MOV.B       #00000111B,TA0IC    ;タイマ A0 割り込み優先レベルの設定
                                        ;(レベル：7）と割り込み要求ビットのクリア
        BSET        TA0S                ;タイマ A0 カウントスタート
INIT_END:
        RTS
```

```
;
;================= TA0 割り込み処理ルーチン ================================
INT_TA0:
        PUSHM   R0,R1,R2,R3,A0,A1    ;レジスタ退避
;               •
;               •
;               プログラム
;               •
;               •
        POPM    R0,R1,R2,R3,A0,A1    ;レジスタ復帰
INT_TA0_END:
        REIT                         ;割り込みからの復帰
;
;================= ダミー割り込みプログラム ================================
dummy:
        REIT
;
;********************** 可変ベクタテーブルの設定 ***************************************
;
  .SECTION  VECT,ROMDATA
  .ORG      VECT_TOP
  .LWORD            dummy       ;BRK 命令割り込
;
  .ORG      VECT_TOP + (8 * 4)
  .LWORD            dummy       ;DMA0 割り込みベクタ
  .LWORD            dummy       ;DMA1 割り込みベクタ
  .LWORD            dummy       ;DMA2 割り込みベクタ
  .LWORD            dummy       ;DMA3 割り込みベクタ
  .LWORD            INT_TA0     ;タイマ A0 割り込みベクタに割り込み処理の
                                ;先番アドレスを設定
  .LWORD            dummy       ;タイマ A1 割り込みベクタ
  .LWORD            dummy       ;タイマ A2 割り込みベクタ
  .LWORD            dummy       ;タイマ A3 割り込みベクタ
  .LWORD            dummy       ;タイマ A4 割り込みベクタ
  .LWORD            dummy       ;UART0 送信割り込みベクタ
  .LWORD            dummy       ;UART0 受信割り込みベクタ
  .LWORD            dummy       ;UART1 送信割り込みベクタ
  .LWORD            dummy       ;UART1 受信割り込みベクタ
  .LWORD            dummy       ;タイマ B0 割り込みベクタ
  .LWORD            dummy       ;タイマ B1 割り込みベクタ
  .LWORD            dummy       ;タイマ B2 割り込みベクタ
  .LWORD            dummy
  .LWORD            dummy
  .LWORD            dummy
  .LWORD            dummy       ;INT4 割り込みベクタ
  .LWORD            dummy       ;INT3 割り込みベクタ
  .LWORD            dummy       ;INT2 割り込みベクタ
  .LWORD            dummy       ;INT1 割り込みベクタ
  .LWORD            dummy       ;INT0 割り込みベクタ
```

割り込みからの復帰は "RTS" 命令ではなく、"REIT" 命令を使用する

ベクタ番号 1～7 番までは内蔵周辺機能の割り込みが配置されていないため、可変割り込みベクタの先頭から 32 バイト（4byte × 8）分先のアドレス（ベクタ番号 8 のアドレス）を設定するための記述

使用しない割り込みの飛び先番地は、未使用の割り込み要求が発生した場合の暴走を防ぐために dummy 処理 (REIT 命令のみ)に飛び先番地を設定する

```
  .LWORD        dummy     ;タイマ B5 割り込みベクタ
  .LWORD        dummy     ;UART2 送信／ NACK 割り込みベクタ
  .LWORD        dummy     ;UART2 受信／ ACK 割り込みベクタ
  .LWORD        dummy     ;UART3 送信／ NACK 割り込みベクタ
  .LWORD        dummy     ;UART3 受信／ ACK 割り込みベクタ
  .LWORD        dummy     ;UART4 送信／ NACK 割り込みベクタ
  .LWORD        dummy     ;UART4 受信／ ACK 割り込みベクタ
  .LWORD        dummy     ;バス衝突検出／スタート , ストップコンディション
                          ;（UART2）割り込みベクタ
  .LWORD        dummy     ;バス衝突検出／スタート , ストップコンディション
                          ;（UART3）割り込みベクタ
  .LWORD        dummy     ;バス衝突検出／スタート , ストップコンディション
                          ;（UART4）割り込みベクタ
  .LWORD        dummy     ;A-D 割り込みベクタ
  .LWORD        dummy     ;キー入力割り込みベクタ
;
;********** 固定ベクタの設定 ***************************************
;
  .SECTION    F_VECT,ROMDATA
  .ORG        FIXED_VECT_TOP
;
  .LWORD        dummy           ;未定義命令割り込みベクタ
  .LWORD        dummy           ;オーバフロー（INTO）割り込みベクタ
  .LWORD        dummy           ;BRK 命令割り込みベクタ
  .LWORD        dummy           ;アドレス一致割り込みベクタ
  .LWORD        dummy           ;未使用
  .LWORD        dummy           ;監視タイマ割り込みベクタ
  .LWORD        dummy           ;未使用
  .LWORD        dummy           ;NMI 割り込みベクタ
  .LWORD        START           ;リセットベクタ設定
;
  .END
```

図 4.3.4　サンプルプログラム 3（割り込みの使用）

4.3.7 ISP と USP について

M16C/80シリーズではスタックポインタが２つ(割り込みスタックポインタ(ISP)とユーザースタックポインタ(USP)）あります。使用するスタックポインタはU フラグによって切り替えることができます。

(1)**U=0 の場合　　ISP を使用**

ISP で示されるアドレスにレジスタ等の退避／復帰が行われます。

(2)**U=1 の場合　　USP を使用**

USP で示されるアドレスにレジスタ等の退避／復帰が行われます。

アセンブリ言語でプログラミングする場合(高級言語やOS を使用しない場合)はISP を使用してください。USP を使用することもできますが、周辺I/O 割り込みの使用において注意が必要です。詳しくは同項の"ソフトウェア割り込み番号とスタックポインタの関係 "をご参照ください。

ソフトウェア割り込み番号の割り当てについて

M16C/80 シリーズではソフトウェア割り込み番号が 0 ～ 63 まであります。8 ～ 43 番は周辺 I/O 割り込みで予約されています。(注1) したがって、残りの 0 ～ 7, 44 ～ 63 番にソフトウェア割り込み（INT 命令 (注2)）を割り当ててください。

ただし、応用上ソフトウェア割り込み番号 32 ～ 63 番の割り込みは、OS 等（M16C/80 用リアルタイムモニタ (MR308)では 48 ～ 63 番を使用）で使用するソフトウェア割り込みとして割り当てられますので、OS 使用時はソフトウェア割り込み番号 0 ～ 7 番を使用してください。

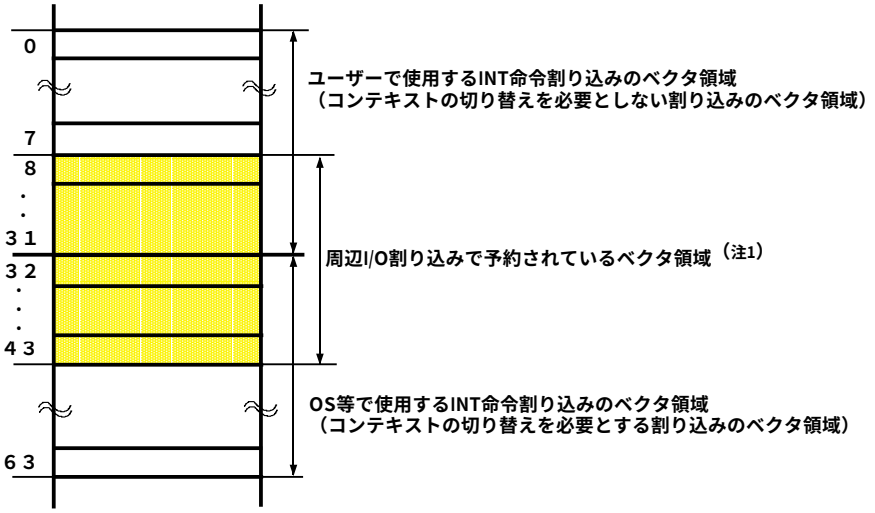


図 4.3.5　割り込み番号の割り当て

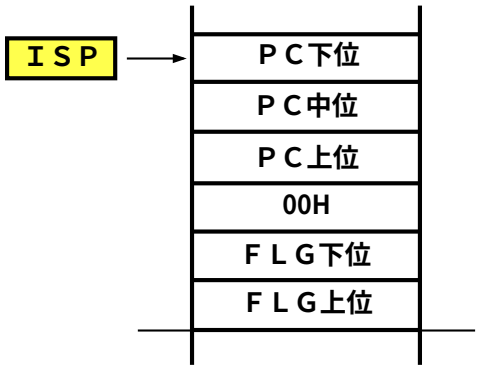
(注 1) 使用する品種によって異なりますので、使用品種のユーザーズマニュアルを参照してください。
(注 2) INT 命令のオペランドで指定された割り込み番号に格納されているアドレスに分岐します。

ソフトウェア割り込み番号とスタックポインタの関係

(1)周辺 I/O の割り込み、またはソフトウェア割り込み番号 0 ～ 31 番を使用した INT 命令の割り込み発生時

①CPU は 000000H 番地を読むことで、割り込み情報(割り込み番号、割り込み要求レベル)を獲得し、その後受け付けた割り込みの割り込み要求ビットを "0" にする。
② FLG レジスタの内容を CPU 内部の一時レジスタに退避する。
③ FLG レジスタの U、I、D フラグをクリアする。
③の動作により、
・スタックポインタは強制的に割り込みスタックポインタ(ISP)になる。
・多重割り込みを禁止する。
・デバッグモードをクリアする。(シングルステップを行わない。)
④ CPU 内部の一時レジスタ(FLG を退避しておいたレジスタ)、及び PC レジスタの内容をスタック領域へ退避する。
⑤プロセッサ割り込み優先レベル(IPL)に、受け付けた割り込みの割り込み優先レベルを設定する。
⑥割り込みベクタに書かれているアドレスを PC レジスタに転送する。

＜割り込み要求受け付け後のスタックの状態＞



＜割り込み要求受け付け後のＦＬＧの状態＞

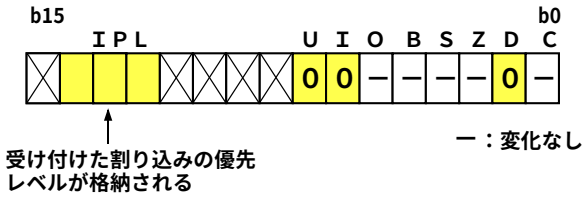


図 4.3.6　周辺 I/O 割り込み、またはソフトウェア割り込み番号 0 ～ 31 番を使用した INT 命令の割り込み発生時

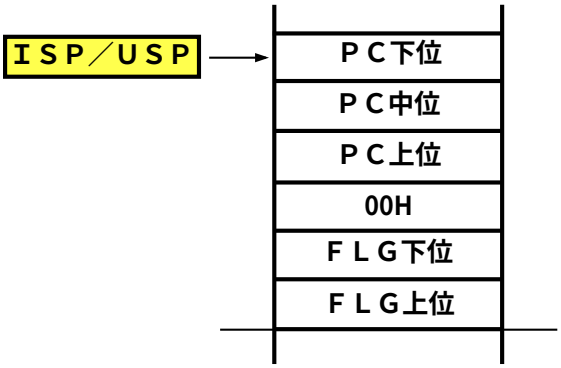
(2) ソフトウェア割り込み番号 32 ～ 63 番を使用した INT 命令の割り込み発生時

①CPU は 000000H 番地を読むことで、割り込み情報(割り込み番号、割り込み要求レベル)を獲得し、その後受け付けた割り込みの割り込み要求ビットを "0" にする。
②FLG レジスタの内容を CPU 内部の一時レジスタに退避する。
③FLG レジスタの I、D フラグをクリアする。

③の動作により、
・スタックポインタは割り込み発生時のスタックポインタを使用する。
・多重割り込みを禁止する。
・デバッグモードをクリアする。(シングルステップを行わない。)

④CPU 内部の一時レジスタ(FLG を退避しておいたレジスタ)、及び PC レジスタの内容をスタック領域へ退避する。
⑤プロセッサ割り込み優先レベル(IPL)に、受け付けた割り込みの割り込み優先レベルを設定する。
⑥割り込みベクタに書かれているアドレスを PC レジスタに転送する。

＜割り込み要求受け付け後のスタックの状態＞



＜割り込み要求受け付け後のＦＬＧの状態＞

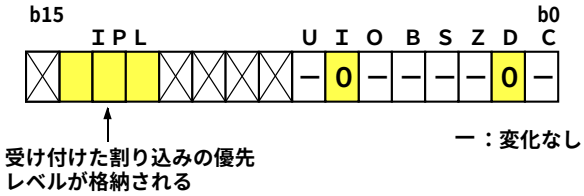


図 4.3.7　ソフトウェア割り込み番号 32 ～ 63 番を使用した INT 命令の割り込み発生時

4.3.8　多重割り込み

M16C/80 シリーズでは、1 つの割り込み要求が受け付けられると、割り込み許可フラグ(I)が自動的に '0'（割り込み禁止）となり、受け付けられた割り込み処理が終了するまで他の割り込みを受け付けなくなります。従って、多重で割り込みをかける場合は、割り込み処理ルーチン内で、割り込み許可フラグ(I)を '1' にセット（割り込み許可）することで可能となります。

多重割り込み実行例

多重割り込みの実行例として、①②③の多重割り込みが発生した場合の実行フローを図 4.3.8 に示します。

①メインルーチン実行中、"割り込み１"が発生
②"割り込み１"実行中、"割り込み２"が発生
③"割り込み２"実行中、"割り込み３"が発生

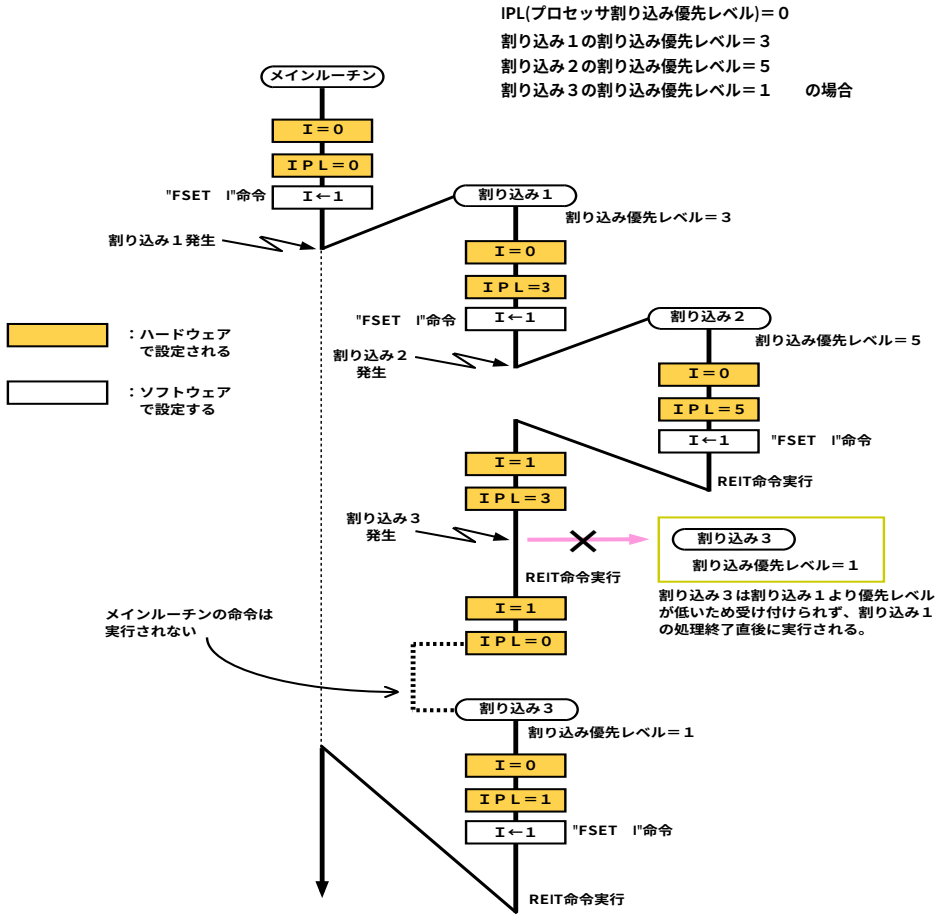


図 4.3.8　多重割り込みの実行例

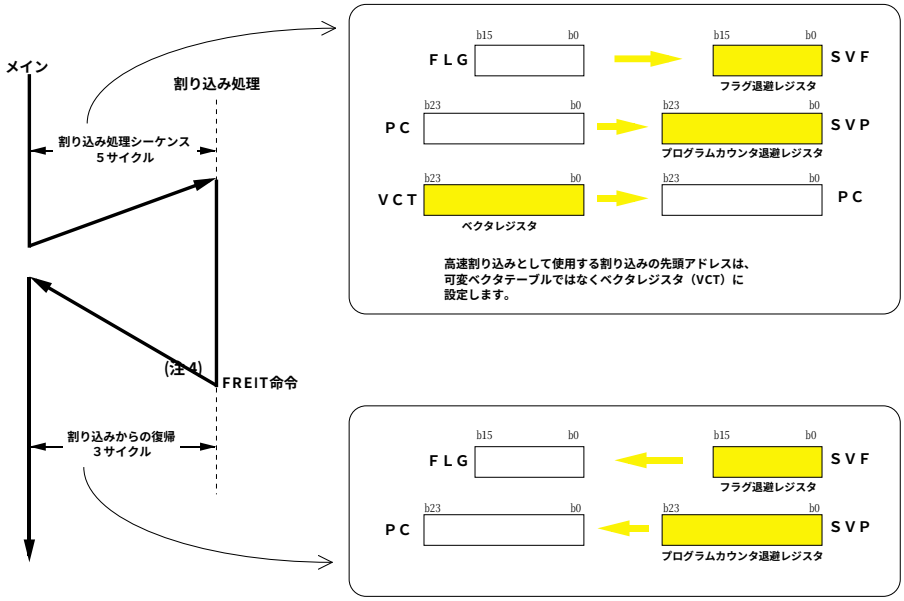
4.3.9　高速割り込み

高速割り込みは、割り込み応答（割り込み処理シーケンス）を５サイクル、復帰を３サイクルで実行できる割り込みです。

高速割り込みでは割り込みを受け付けると、フラグレジスタ(FLG)とプログラムカウンタ(PC)の退避を、それぞれ CPU 内部レジスタのフラグ退避レジスタ(SVF)と PC 退避レジスタ(SVP)に行い、ベクタレジスタ(VCT)で示される番地からプログラムを実行します。

高速割り込みは、高速割り込み指定ビット(注1)に"1"を設定することにより使用可能となり、ソフトウエア割り込み優先レベルが"7"に設定されている割り込み(注2)が高速割り込みとなります。　高速割り込みの動作を以下に示します。

高速割り込みの応答／復帰動作



高速割り込みは"割り込みシーケンス"を短くした割り込みですので、レジスタの退避／復帰については極力"バンク切り替え"により行い、高速割り込み処理ルーチン内では"レジスタバンク１"を専用のレジスタとして使用するようにしてください。(注3)

図 4.3.9　高速割り込みの動作

(注 1)「復帰用優先順位レジスタ」のビット 3 に割り付けられているビットです。
(注 2)高速割り込みとして設定できる割り込みは１つだけですので、割り込み優先レベルが"7"の割り込みを複数使用しないでください。
(注 3)この場合、メインルーチンで"レジスタバンク１"を使用することはできません。
(注 4)高速割り込みルーチンからの復帰は FREIT 命令にて実行してください。

高速割り込み使用時のプログラム記述例

```
;********************** インクルード ***********************************************
  .INCLUDE      M30800.INC
;********************** プログラム領域 *********************************************
;
  .SECTION      PROGRAM,CODE            ;セクション名、セクションタイプの宣言
  .ORG          ROM_TOP                 ;開始アドレスの宣言
START:
;               •
;               •
;               CPU の初期設定
;               •
;               •
;=================== メインプログラム ======================================
MAIN:
        JSR         INIT                ;ワーク RAM の初期値設定
        FSET        I                   ;割り込み許可
MAIN_10:
        MOV.W       WORK_1,R0
;               •
;               •
        JMP         MAIN_10
;
;================= INIT ルーチン ============================================
INIT:
        MOV.W       #0,WORK_1
        MOV.W       #0,WORK_2
;               •
        MOV.B       #00000111B,INT0IC   ;INT0 端子割り込みの優先レベル（レベル：7）、
                                        ;エッジ極性（立ち下がり）の設定、及び割り
                                        ;込み要求ビットのクリア
;
        LDC         #INT_INT0, VCT      ;ベクタレジスタの設定
        BSET        3, RLVL             ;復帰用優先順位レジスタの設定
INIT_END:
        RTS
;
;================= INT0 割り込み処理ルーチン ================================
INT_INT0:
        FSET    B                       ;レジスタバンク切り替えによるレジスタの退避
;               •
;               プログラム
;               •
;
INT_INT0_END:
        FREIT                           ;高速割り込みからの復帰
;
```

ベクタレジスタ(高速割り込み専用割り込みベクタ)に割り込み処理ルーチンの先頭アドレスを設定する

高速割り込みを使用する場合、割り込み優先レベルを "7" に設定できる割り込みは１つとなる

高速割り込み指定ビットを "1" にセットし、割り込み優先レベル "7" の割り込みを高速割り込みとして設定する

高速割り込みからの復帰は "FREIT 命令 " を使用する。

" レジスタバンク切り替え " によりレジスタの退避を行うことで、割り込み応答時間をさらに短くできます

```
;================= ダミー割り込みプログラム ================================
dummy:
        REIT
;
;********************** 可変ベクタテーブルの設定 ***************************************
;
  .SECTION      VECT,ROMDATA
  .ORG          VECT_TOP
  .LWORD            dummy       ;BRK 命令割り込みベクタ
;
  .ORG          VECT_TOP + (8*4)
  .LWORD            dummy       ;DMA0 割り込みベクタ
  .LWORD            dummy       ;DMA1 割り込みベクタ
  .LWORD            dummy       ;DMA2 割り込みベクタ
  .LWORD            dummy       ;DMA3 割り込みベクタ
  .LWORD            dummy       ;タイマ A0 割り込みベクタ
  .LWORD            dummy       ;タイマ A1 割り込みベクタ
  .LWORD            dummy       ;タイマ A2 割り込みベクタ
  .LWORD            dummy       ;タイマ A3 割り込みベクタ
  .LWORD            dummy       ;タイマ A4 割り込みベクタ
  .LWORD            dummy       ;UART0 送信割り込みベクタ
  .LWORD            dummy       ;UART0 受信割り込みベクタ
  .LWORD            dummy       ;UART1 送信割り込みベクタ
  .LWORD            dummy       ;UART1 受信割り込みベクタ
  .LWORD            dummy       ;タイマ B0 割り込みベクタ
  .LWORD            dummy       ;タイマ B1 割り込みベクタ
  .LWORD            dummy       ;タイマ B2 割り込みベクタ
  .LWORD            dummy       ;タイマ B3 割り込みベクタ
  .LWORD            dummy       ;タイマ B4 割り込みベクタ
  .LWORD            dummy       ;INT5 割り込みベクタ
  .LWORD            dummy       ;INT4 割り込みベクタ
  .LWORD            dummy       ;INT3 割り込みベクタ
  .LWORD            dummy       ;INT2 割り込みベクタ
  .LWORD            dummy       ;INT1 割り込みベクタ
  .LWORD            dummy       ;INT0 割り込みベクタ
  .LWORD            dummy       ;タイマ B5 割り込みベクタ
  .LWORD            dummy       ;UART2 送信／ NACK 割り込みベクタ
  .LWORD            dummy       ;UART2 受信／ ACK 割り込みベクタ
  .LWORD            dummy       ;UART3 送信／ NACK 割り込みベクタ
  .LWORD            dummy       ;UART3 受信／ ACK 割り込みベクタ
  .LWORD            dummy       ;UART4 送信／ NACK 割り込みベクタ
  .LWORD            dummy       ;UART4 受信／ ACK 割り込みベクタ
  .LWORD            dummy       ;バス衝突検出／スタート , ストップコンディション
                                ;(UART2) 割り込みベクタ
  .LWORD            dummy       ;バス衝突検出／スタート , ストップコンディション
                                ;(UART3) 割り込みベクタ
  .LWORD            dummy       ;バス衝突検出／スタート , ストップコンディション
                                ;(UART4) 割り込みベクタ
  .LWORD            dummy       ;A-D 割り込みベクタ
  .LWORD            dummy       ;キー入力割り込みベクタ
```

高速割り込みとして設定した割り込みについては、可変割り込みベクタに割り込み処理ルーチンの先頭番地は設定しない

```
;
;********************** 固定ベクタの設定 ***************************************
;
  .SECTION     F_VECT,ROMDATA
  .ORG         FIXED_VECT_TOP
;
  .LWORD             dummy             ;未定義命令割り込みベクタ
  .LWORD             dummy             ;オーバフロー（INTO）割り込みベクタ
  .LWORD             dummy             ;BRK 命令割り込みベクタ
  .LWORD             dummy             ;アドレス一致割り込みベクタ
  .LWORD             dummy             ;未使用
  .LWORD             dummy             ;監視タイマ割り込みベクタ
  .LWORD             dummy             ;未使用
  .LWORD             dummy             ;NMI 割り込みベクタ
  .LWORD             START             ;リセットベクタ設定
;
  .END
```

図 4.3.10　高速割り込み使用時のプログラム例

4.4 ソースファイルの分割

プログラムはいくつかのソースファイルに分割して記述します。分割することによって、プログラムがまとまり読みやすくなります。さらに、ファイルごとにアセンブルできるため、修正時のアセンブル時間を短縮することができます。この節ではソースファイルの分割について説明します。

4.4.1 セクションの概念

*アセンブリ言語で書かれたプログラムは一般的にワーク領域、プログラム領域、定数データ領域から構成されます。アセンブラ(as308)でソースファイル(***.A30)をアセンブルするとリロケータブルモジュールファイル(***.R30)が生成されます。リロケータブルモジュールファイルはこのような領域を１つ以上含んでいます。セクションはこの各領域ごとに付けられた名前です。つまりセクションとはプログラムの構成要素別に付けられた名前ということになります。*

なお、アセンブラ(as308)ではアブソリュートファイルの場合でも、１つのファイルにつき必ず１つ以上のセクションを指定しなければなりません。

セクションの機能

リンク時に、同一セクション名の領域を指定されたファイル順に連続して配置します。また各セクションの開始アドレスをリンク時に指定することができます。これはつまり、各セクションの再配置がソースプログラムを変更することなく何回でも可能ということです。図 4.4.1 に実際のセクションの配置例を示します。

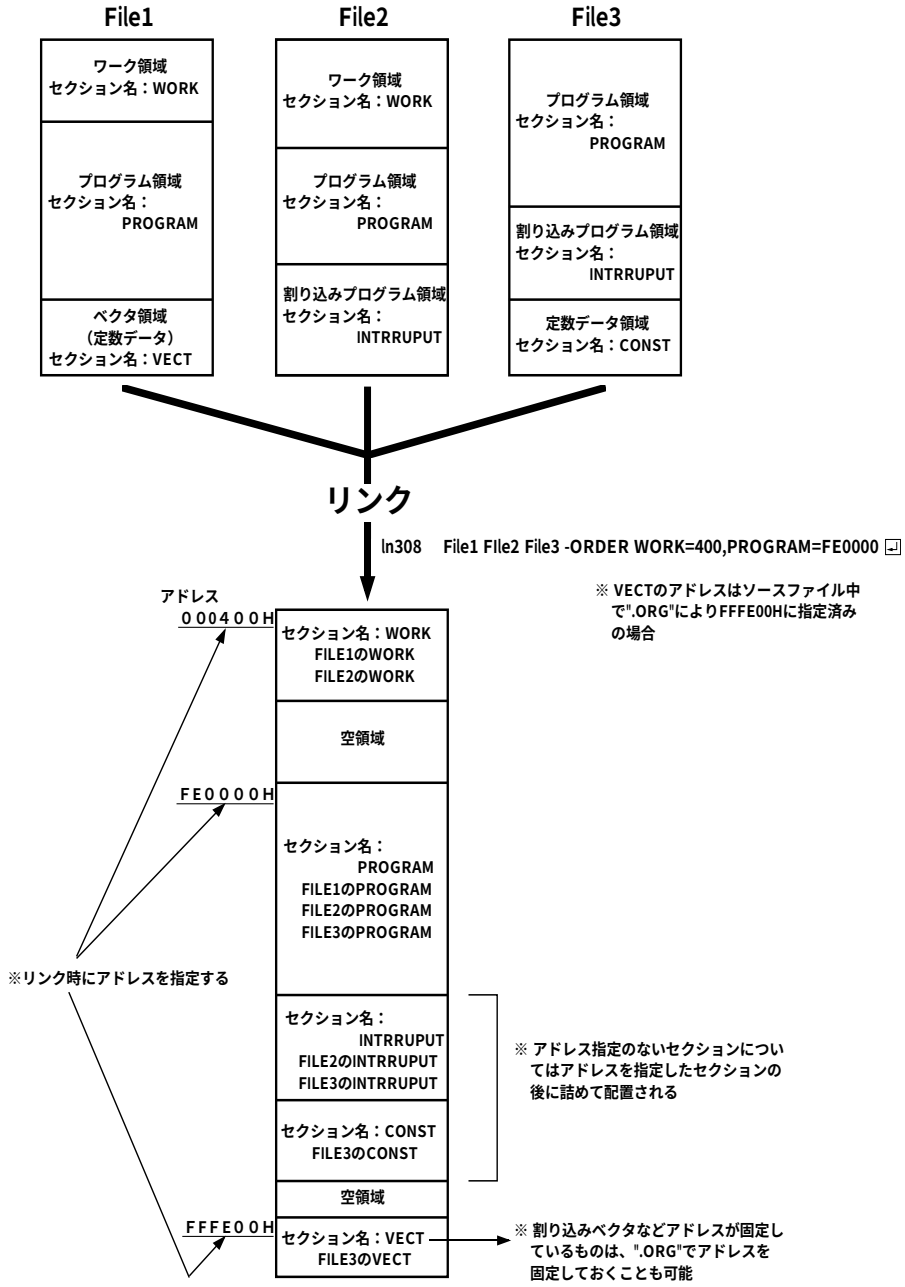


図 4.4.1　セクション配置例

4.4.2 ファイル分割の記述例

本書で使用する as308 はリロケータブルアセンブラです。リロケータブルアセンブラでは通常プログラムソースをいくつかのファイルに分割して記述します。ファイルを分割することによるメリットを以下に示します。

(1)プログラム、データの共有化

開発プロジェクト間のデータの交換が容易になり、既存ソフトウェアから必要な部分のみ再利用することができます。

(2)アセンブル時間の短縮

変更（修正）したファイルのみ再アセンブルするだけですみ、アセンブル時間を短縮することができます。

この項ではファイルを３つに分割(定義、メインプログラム、サブルーチン処理)して記述した場合のソースプログラムの記述例について説明します。

分割例 1　定義(WORK.A30)

ワーク RAM 領域確保とデータテーブル設定のセクションをファイル 1 に記述します。

```
;***************************************************************************
;                    ファイル 1（WORK.A30）
;***************************************************************************
;=================== ワーク RAM　領域確保 ================================
;
  .SECTION      WORK,DATA
  .ORG          000400H
  .GLB          WORK_1,WORK_2,WORK_3,WORK_4 ;グローバルラベルとして処理する
  .GLB          DATA_TABLE                  ;グローバルラベルとして処理する
  .BTGLB        W1_b0,W2_b1                 ;グローバルビットシンボルとして
                                            ;処理する
;
GLOBAL_WORK_TOP:
WORK_1:         .BLKB  1                    ;ワーク RAM 領域の確保↓
WORK_2:         .BLKB  1
WORK_3:         .BLKB  1
WORK_4:         .BLKB  1
GLOBAL_WORK_END:
W1_b0           .BTEQU       0,WORK_1       ;ビットシンボルの定義↓
W2_b1           .BTEQU       1,WORK_2       ;
;
;
;=================== 固定データ領域 ======================================
;
  .SECTION      CONSTANT,ROMDATA
;
DATA_TABLE:
          .BYTE  12H                        ;1 バイトデータの設定
          .BYTE  34H
          .BYTE  56H
          .BYTE  78H
DATA_TABLE_END:
;
  .END
```

ワーク RAM、及びラベルを別ファイルから参照可能にするため、".GLB" によりグローバルラベルの宣言を行う

".BTEQU" で定義されたビットシンボルを別ファイルから参照可能にするため、".BTGLB" によりグローバルビットシンボルの宣言を行う

図 4.4.2　分割ファイル 1(WORK.A30)

分割例 2　メインプログラム(MAIN.A30)

メインプロプログラム("PROGRAM" セクション)をファイル 2 に記述します。

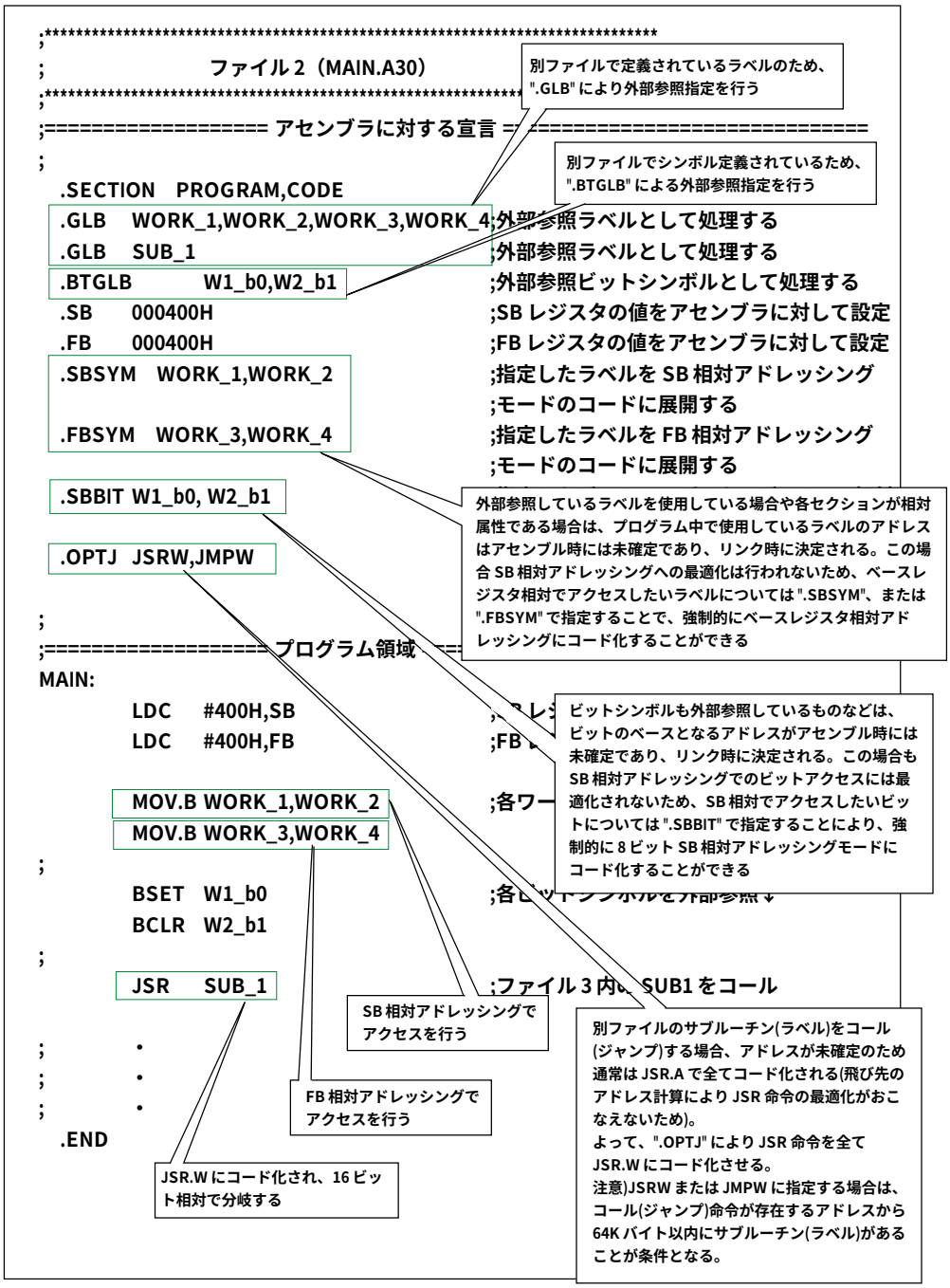


図 4.4.3　分割ファイル 2(MAIN.A30)

分割例 3　サブルーチン処理(SUB_1.A30)

サブルーチン処理("PROGRAM" セクション)とワーク RAM 領域の確保("WORK" セクション)をファイル 3 に記述します。

```
;***************************************************************************
;                    ファイル 3（SUB_1.A30）
;***************************************************************************
;********************** ワーク RAM 領域確保 ********************
;
  .SECTION      WORK,DATA
;
LOCAL_WORK_TOP:
LOCAL_1:              .BLKB       1       ;ローカルデータの領域確保
LOCAL_2:              .BLKB       1
LOCAL_WORK_END:
;
;********************** アセンブラに対する宣言 ********************
;
  .SECTION      PROGRAM,CODE
  .GLB          SUB_1                     ;グローバルラベルとして処理する
  .GLB          DATA_TABLE                ;外部参照ラベルとして処理する
;
  .SB           000400H                   ;SB レジスタの値をアセンブラに対して設定
  .FB           000580H                   ;FB レジスタの値をアセンブラに対して設定
  .SBSYM        LOCAL_1,LOCAL_2           ;指定したラベルを SB 相対アドレッシング
                ;モードでコード化する
;=================== プログラム領域 ==============
SUB_1:
        LDC     #400H,SB                  ;SB レジスタ初期値設定
        LDC     #580H,FB                  ;FB レジスタ初期値設定
;
        MOV.B #05H,LOCAL_1                ;ローカルデータ(LOCAL_1)を SB 相対
                                          ;アドレッシングでアクセス
;
        MOV.W#0,A0
        MOV.B DATA_TABLE[A0],LOCAL_2      ;外部参照による固定データテーブルの検索
        ADD.B LOCAL_1,LOCAL_2             ;ローカルデータ(LOCAL_1,LOCAL_2)の加算
;       ・
;       ・
;       ・
        RTS

  .END
```

グローバル宣言をしなければ、ファイル 3 (SUB_1.A30)のローカルラベルとして扱われる

ファイル 2(MAIN.A30)からサブルーチン(SUB_1)をコールしているため、.GLB で SUB_1 をグローバルラベルに指定しておく(ファイル内に存在するラベルのためグローバル宣言となる)

別ファイル(ファイル 1)に存在するラベルをプログラム中で使用しているため、外部参照指定を行う

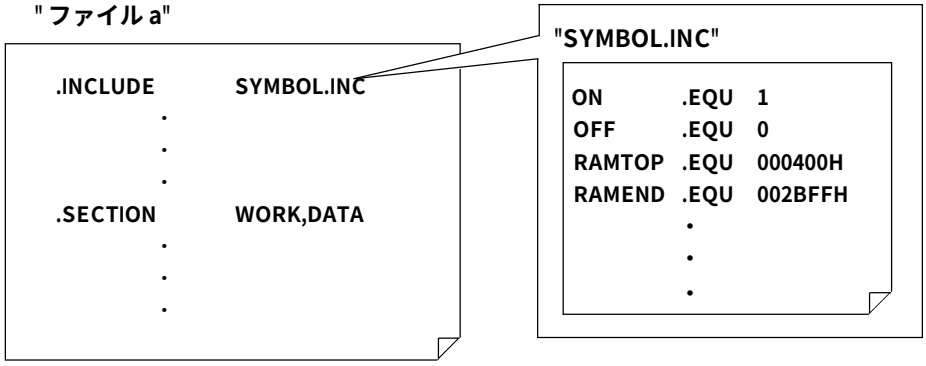
相対属性のセクションのため、リンクするまでラベルのアドレスが未確定となり、SB 相対アドレッシングモードへの最適化が行われないため、".SBSYM" により強制的に SB レジスタ相対アドレッシングにコード化する。
注意) ".SBSYM"(".FBSYM")により指定するデータについては、SB/FB 相対アドレッシングのアクセス範囲内にあることを確認した上で指定する。

図 4.4.4　分割ファイル 3(SUB_1.A30)

インクルードファイルの利用

通常シンボルやビットシンボル(.EQU, .BTEQUで定義したもの)、及びラベル(アドレス情報を持つもの)の外部参照指定の部分を１つのインクルードファイルにします。シンボルやラベルを外部参照する場合、各ファイルごとに外部参照を指定しなくても、インクルードファイルを読み込むことにより外部参照することができます。

(1)シンボルの参照例



(2)グローバルラベルの参照例

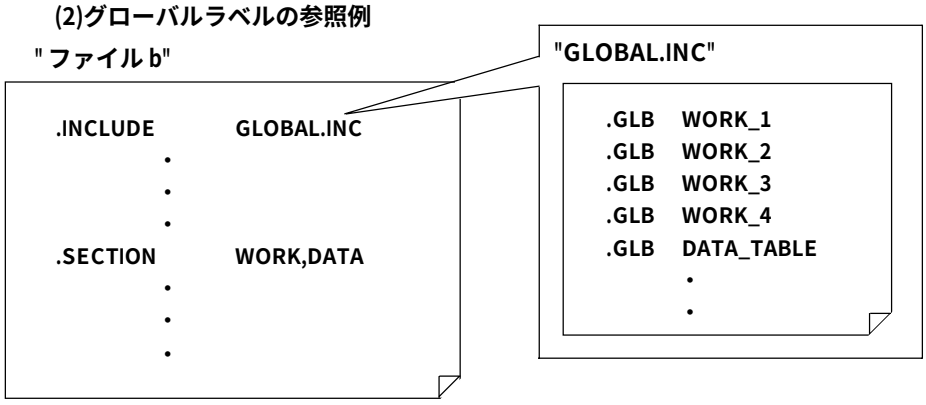


図 4.4.5　インクルードファイル例

指示命令 .LIST の利用

指示命令 ".LIST　ON, .LIST　OFF" をファイルの先頭と最後に記述することで、アセンブラリストファイルへのインクルードファイルの出力を停止することができます。図 4.4.6 にこの指示命令を使用しない場合(展開 1)と使用する場合(展開 2)のアセンブラリストファイルを示します。

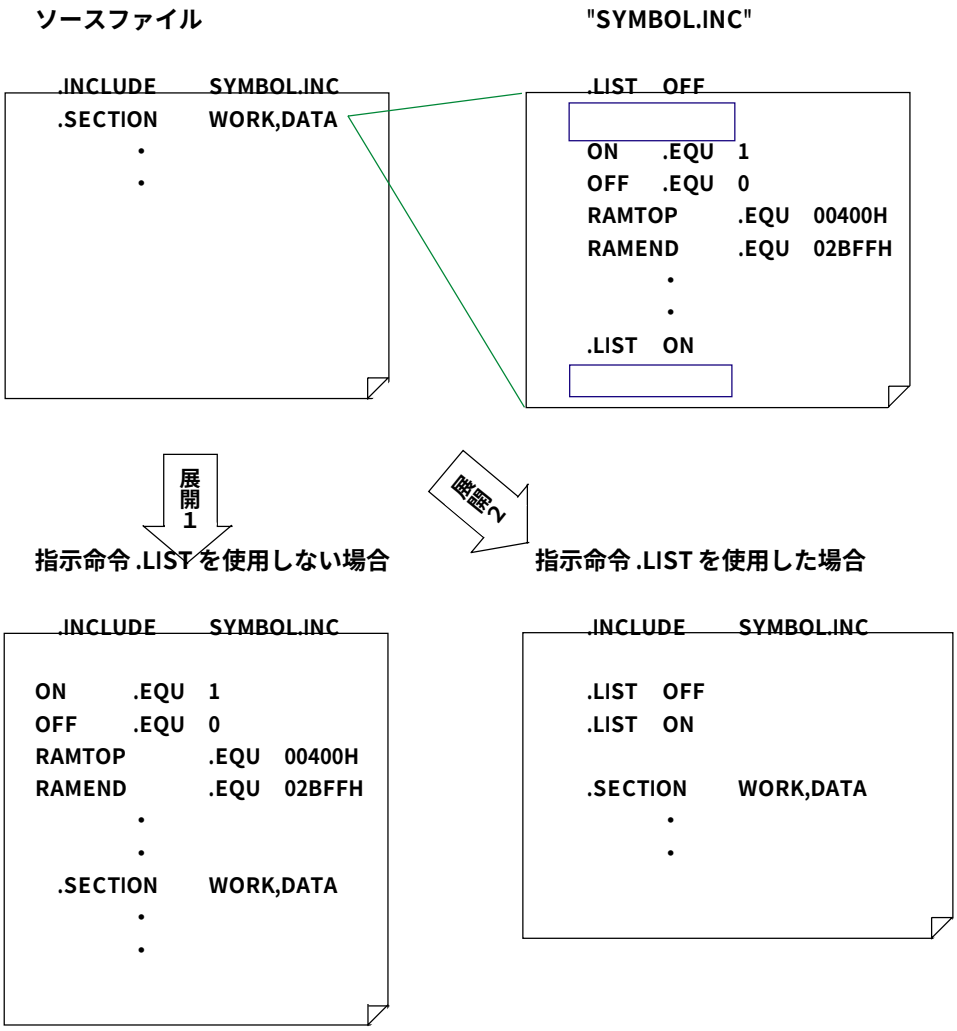


図 4.4.6　指示命令 .LIST の利用

4.4.3　ライブラリファイルの利用

*ライブラリファイルとはいくつかのリロケータブルモジュールファイルを集めたものです。頻繁に使用するモジュールは、AS308システム付属のライブラリアン(lib308)によって１つのライブラリファイルとして構成することができます。そのファイル(***.LIB)をリンク時に指定することで、必要なモジュールのみ（ファイル中で外部参照指定したモジュールのみ）抽出してリンクすることができます。これによりアセンブル時間の短縮、プログラムの再利用などが可能になります。以下にライブラリファイルの生成とライブラリファイルをリンクする場合の例を示します。*

ライブラリファイルの生成

ライブラリファイルの生成例を図 4.4.7 に示します。

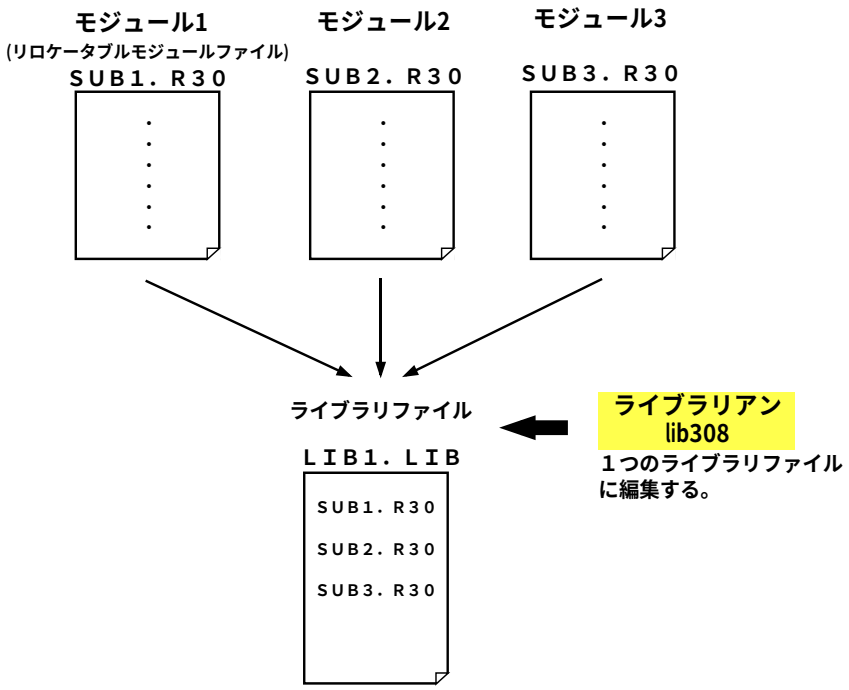


図 4.4.7　ライブラリファイルの生成

ライブラリファイルをリンクする場合の例

ライブラリファイルをリンクする場合の例を図 4.4.8 に示します。

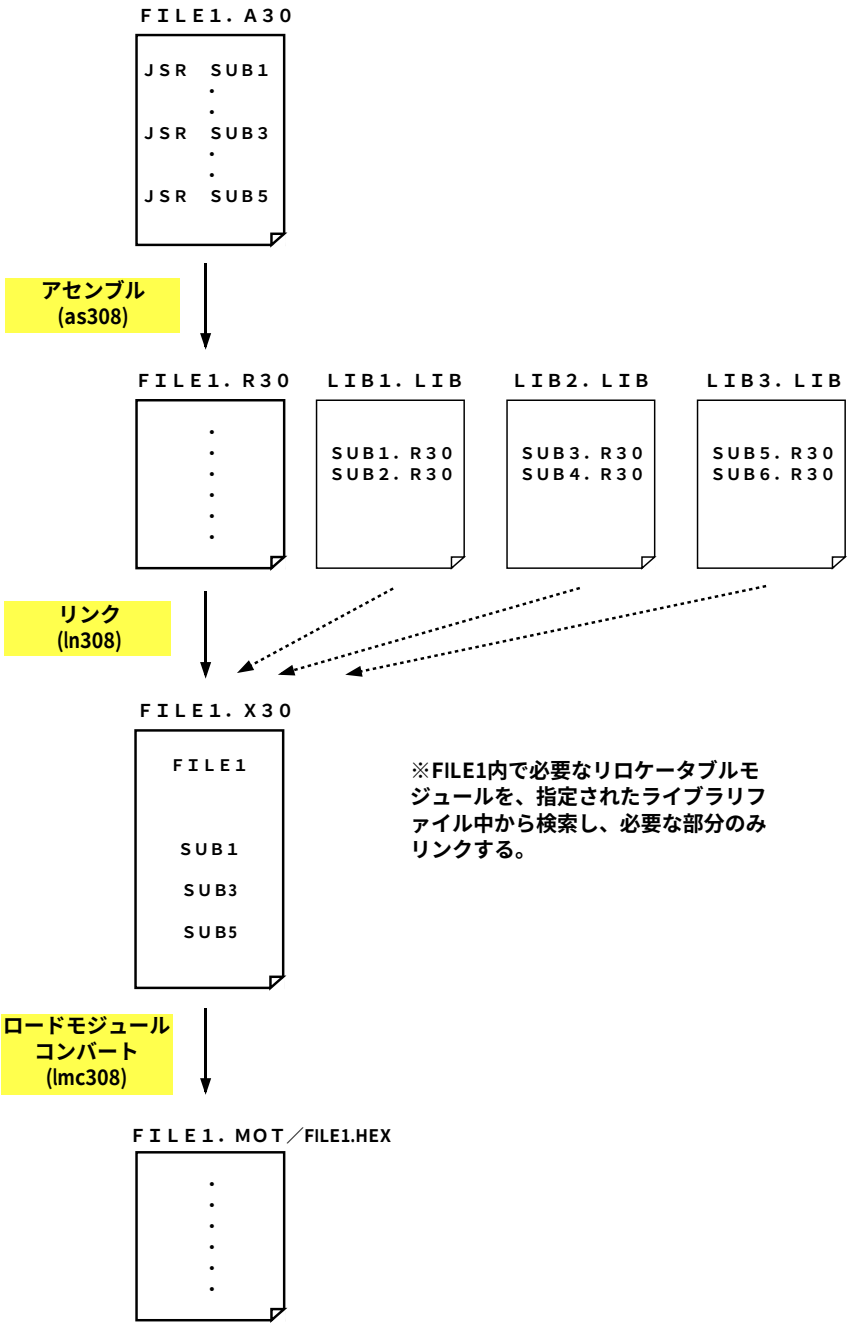


図 4.4.8　ライブラリファイルとリロケータブルモジュールファイルのリンク例

4.5　ちょっと小耳を...（プログラミングテクニック）

この節ではM16C/80シリーズを使用する上で、知っておくと便利な事項をまとめています。項目ごとにご参照ください。

4.5.1　SB、FB レジスタの使用方法

この項では SB レジスタと FB レジスタの設定値について説明します。

基本的な SB、FB レジスタの使用方法

頻繁にアクセスするようなデータを持つ領域の先頭アドレスを SB,FB レジスタに設定します。よって、頻繁に使用する SFR 領域やワーク RAM 領域のアドレスを設定すると効果的です。

図 4.5.1 に SB,FB レジスタ値を固定して使用する場合の設定例を示します。

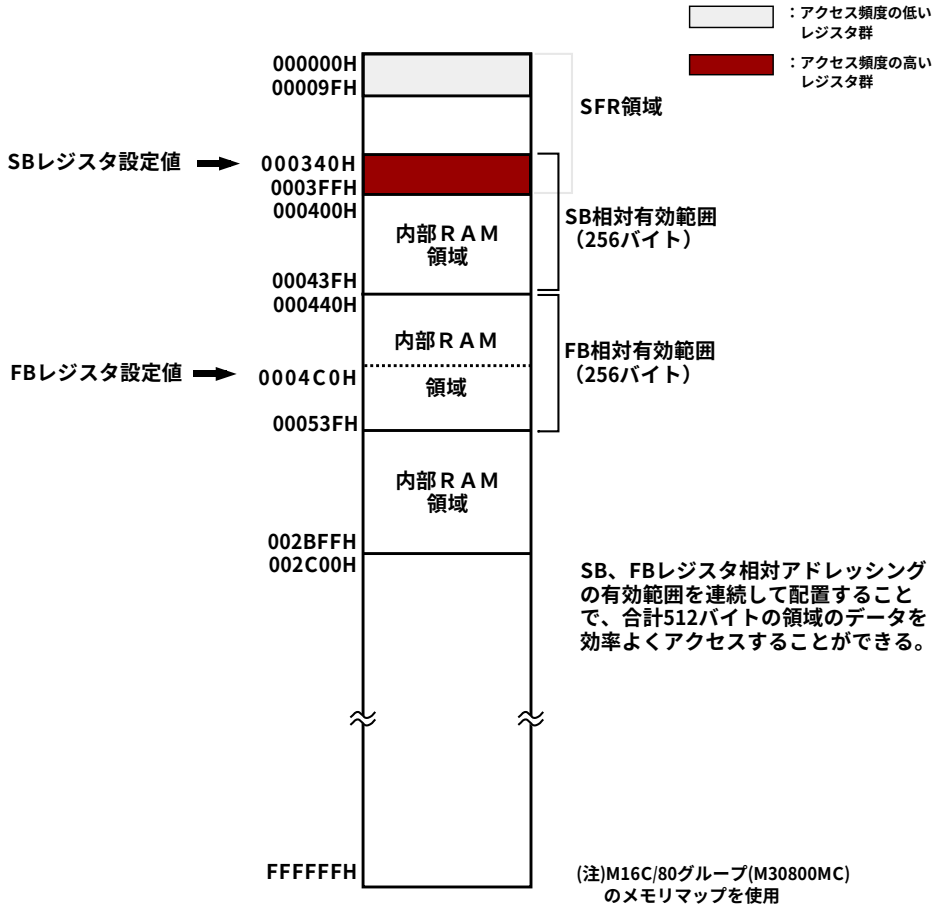


図 4.5.1　SB､FB レジスタ値を固定して使用する場合の設定例

SB、FB レジスタ使用についての応用

SB,FBレジスタの値をプログラム中で固定して使用した場合、効率の良いアクセスができる範囲はそれぞれ256バイトで、合計 512 バイトの領域が最大値となります。

SB/FB相対アドレッシングをもっと広範囲で使用し、ワークデータなどへのアクセス効率の向上、及びROM効率の向上を図りたい場合は、"サブルーチン毎"に設定する値を変更する、つまりSB,FBレジスタ値を動的に使用することで可能となります。

使用例については、"図4.5.3　SB,FBレジスタ値を動的に使用する場合のプログラム例"を参照してください。

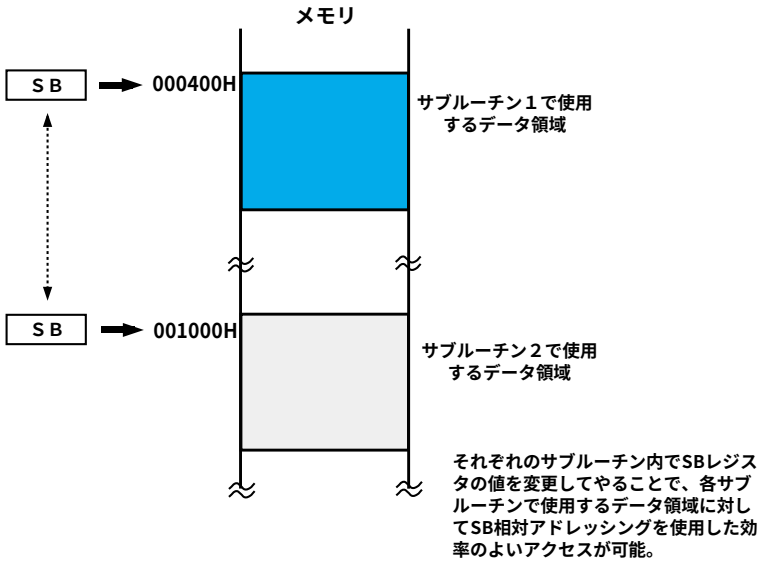


図 4.5.2　SB、FB レジスタ値を動的に使用する場合

SB、FB レジスタを動的に使用する場合のプログラミング例

以下に SB,FB レジスタ値を動的に使用した場合のプログラム例を示します。

```
;********************* プログラム領域 ************************************************
;================== スタートアップ ======================================
;
  .SECTION      PROGRAM,CODE            ;セクション名、セクションタイプの宣言
  .ORG          ROM_TOP                 ;開始アドレスの宣言
START:
        LDC     #RAM_END+1,ISP          ;スタックポインタ(ISP)の初期値設定
;       •
;       •
;       •
  .SB   340H                            ;SB レジスタの値をアセンブラに対して宣言
  .FB   4C0H                            ;FB レジスタの値をアセンブラに対して宣言
        LDC     #340H,SB                ;SB レジスタ初期値設定
        LDC     #4C0H,FB                ;FB レジスタ初期値設定
;       •
;       •
;
;================= メインプログラム ==========
MAIN:
        JSR             INIT            ;初期設定ルーチン
        FSET            I               ;割り込み許可
MAIN_10:
        JSR             SUB_1           ;サブルーチン "SUB_1" 呼び出し
;               •
;               •
        JSR             SUB_2           ;サブルーチン "SUB_2" 呼び出し
;               •
        JMP     MAIN_10
;
;================ INIT ルーチン ========================================
INIT:
        MOV.B           #0FFH,WORK_1
        MOV.B           #0FFH,WORK_2
;
        MOV.B           #01000000B,TA0MR    ;タイマ A0 モードレジスタ設定
        MOV.W           #2500-1,TA0         ;タイマ A0 カウント値設定
        MOV.B           #00000111B,TA0IC    ;タイマ A0 割り込み優先レベルの設定
        BSET            TA0S                ;タイマ A0 カウントスタート
INIT_END:
        RTS
;
```

必ず SB,FB レジスタに設定する値と同じ値をアセンブラに対しても設定すること

SB,FB レジスタの初期値を設定。
サンプルプログラムでは、メイン、及び "INIT" ルーチンで SB 相対を 340H ～ 43FH、FB相対を 440H～53FHの範囲で使用可能となる。

```
;================= SUB_1 ルーチン ====================================
SUB_1:
        .SB     400H                    ;変更する SB レジスタの値をアセンブラに対して宣言
        .FB     580H                    ;変更する FB レジスタの値をアセンブラに対して宣言
        LDC     #400H,SB                ;SB レジスタ値を変更
        LDC     #580H,FB                ;FB レジスタ値を変更
;
        MOV.B   WORK_1,R0L
        INC.B   R0L
;               •
;               •
SUB_1_END:
        RTS
;
;================= SUB_2 ルーチン ====================================
SUB_2:
        .SB     600H                    ;変更する SB レジスタの値をアセンブラに対して宣言
        .FB     780H                    ;変更する FB レジスタの値をアセンブラに対して宣言
        LDC     #600H,SB                ;SB レジスタ値を変更
        LDC     #780H,FB                ;FB レジスタ値を変更
;
        MOV.B   WORK_2,R1L
        DEC.B   R1L
;               •
;               •
SUB_2_END:
        RTS
;
;================= 割り込み ==========================================
INT_TA0:
        .SB     1000H                   ;変更する SB レジスタの値をアセンブラに対して宣言
;
        FSET    B                       ;レジスタの退避（SB,FB レジスタ含む）
        LDC     #1000H,SB               ;SB レジスタ値を変更
;
        MOV.B   #0, COUNT
        DADD.B  #2, DATA
;               •
;               •
INT_TA0_END:
        REIT
;               •
;               •
        .END
```

必ずアセンブラに対しても ".SB"、".FB"で、変更する値を指定すること！

サブルーチン"SUB_1"で使用したいSB,FB相対の範囲に従って、SB,FB レジスタの値を設定する。
注意）SB,FBレジスタ値の設定は、必ずワークRAMなどをアクセスする前に設定すること！（通常は各ルーチンの先頭で設定）

サブルーチン"SUB_2"で使用したいSB,FB相対の範囲に従って、SB,FB レジスタの値を設定する。
注意）SB,FBレジスタ値の設定は、必ずワークRAMなどをアクセスする前に設定すること！（通常は各ルーチンの先頭で設定）

割り込みは非同期で入ってくるため、必ずメインルーチンで使用しているSB、FB レジスタ値を退避してから値を再設定すること！

割り込み処理"INT_TA0"で使用したいSB,FB相対の範囲に従って、SB,FB レジスタの値を設定する。

図 4.5.3　SB,FB レジスタ値を動的に使用する場合のプログラム例

4.5.2 ROM ／ RAM データのアライメント指定

この項ではアライメントの指定について説明します。

アライメントの指定とは

指示命令 ".ALIGN" を記述した直後の行のコードを格納するアドレスを偶数に補正します。セクションタイプが "CODE" または "ROMDATA" の場合は、アドレス補正した結果空になったところに NOP 命令が書き込まれます。セクションタイプが "DATA" の場合は、アドレスのみ補正します。なお、本指示命令を記述した箇所のアドレスが偶数の場合は、補正は行われません。

本指示命令は、以下の条件に当てはまるセクション内に記述できます。

(1)セクション定義でアドレス補正を指示している相対属性のセクション

```
.SECTION    WORK, DATA, ALIGN
```

(2)絶対属性のセクション（特に制限なし）

```
.SECTION    WORK, DATA
.ORG        400H
```

アライメント指定(偶数アドレスへの補正)のメリット

データテーブルなどサイズの異なるデータが連続して配置されている場合、奇数バイトデータの次に配置されているデータは、奇数番地に割り付けられます。M16C/80 シリーズでは偶数番地から始まるワードデータ(2バイトのデータ)は 1 回のアクセスでリード／ライトされますが、奇数番地から始まるワードデータをアクセスした場合は、2 回のアクセスを必要とします。従ってワードやロングワードデータなど 2 バイト以上のデータについては、偶数番地に配置する方がアクセス効率が良く、命令の実行速度を上げることができます。ただしアライメント指定した場合、ROM(または RAM)効率は下がります。

図 4.5.4 にアライメント指定したプログラムの記述例を示します。

(1)セクションが相対属性の場合

```
                                        アドレス    コード
 .SECTION     WORK, DATA, ALIGN
WORK_1 .BLKW      1                     000000H
WORK_2 .BLKL      1                     000002H
WORK_3 .BLKB      1                     000006H
 .ALIGN                                 000007H          アドレスを +1
        .
        .                                  データテーブルなどのセクションは
                                           できるだけ偶数番地に設定する!!
 .SECTION     CONST, ROMDATA, ALIGN
  .BYTE       12H                       00000H    12H
 .ALIGN                                 00001H    04H    NOP コードを挿入
  .WORD       3456H                     00002H    5634H
              .
              .
```

(2)セクションが絶対属性の場合

```
                                        アドレス    コード
 .SECTION     WORK, DATA                    データテーブルなどのセクションは
 .ORG         400H                          できるだけ偶数番地に設定する!!
WORK_1        .BLKB       1           000400H
 .ALIGN                                 000401H          アドレスを +1
WORK_2        .BLKW       1           000402H
WORK_3        .BLKA       1           000404H
 .ALIGN                                 000407H          アドレスを +1
WORK_4        .BLKL       1           000408H
;
  .SECTION    PROGRAM,CODE
  .ORG        0FE0000H
        MOV.W       #0, WORK_1        FE0000H   130004H
              .
              .
```

図 4.5.4　アライメント指定例

4.5.3 スタックポインタの設定

スタックポインタの設定方法と、割り込みとサブルーチンを使用したときのスタック領域への退避と復帰について説明します。

スタックポインタ(ISP/USP)の設定方法

①使用するスタックポインタ(ISP,USP)の選択

通常アセンブリ言語でのみ開発する場合は、"ISP"を使用してください。

ISP,USPともに使用する場合は、Uフラグの初期値を"1"(USPを使用)に設定することで、メインルーチン側では"USP"で示されるスタック領域が使われ、周辺I/Oの割り込み処理ルーチン側では"ISP"で示されるスタック領域が使われることになります。(注1)

これにより、メイン処理と割り込み処理で別々にスタックの使用量を見積もることができますので、ファイルを分割して複数人数でプログラム開発をするような場合に有効です。 詳しくは「4.3.7 ISPとUSP」をご参照ください。

②選択したスタックポインタレジスタに初期値を設定する。

M16C/80グループのスタックはFILO方式(注2)ですので、スタックポインタの初期値はRAM領域の最終アドレスに設定することを推奨します。

また、スタックにレジスタなどが退避／復帰される場合、スタックポインタは＋方向、－方向ともに必ず2つずつ動きますので、初期値は必ず偶数番地に設定するようにしてください。 詳しくは、次頁の「スタック領域への退避と復帰」をご参照ください。

設定例）割り込みスタックポインタ(ISP)に"2C00H"、ユーザースタックポインタ(USP)に"2900H"を設定する場合(注3)

```
;----------------------- スタックポインタの初期化 -----------------------
;
        LDC   #002C00H, ISP             ;ISP に "2C00H" を設定
        FSET  U                         ;
        LDC   #002900H, SP              ;USP に "2900H" を設定
        FCLR  U                         ;メイン側では "USP" を使用し、割り込み
              ;処理ルーチン側で "ISP" を使用する
  ;
```

(注1) ISP,USPともに使用する場合は、それぞれのスタック領域が重ならないように領域を確保してください。また、必ず両方のスタックポインタに値を設定してください。

(注2) FILO(ファースト・イン・ラスト・アウト)方式：レジスタ等の退避はアドレスの大きい方から小さい方に向かって順に積まれ、復帰する場合は最後に退避したものから順にアドレスの大きい方に向かって取り出されていく方式。

(注3) ISP,USP,FLGは専用レジスタですので、LDC、FSET／FCLR命令等を使用して設定してください。

スタック領域への退避と復帰

以下の場合にスタック領域にレジスタ等が退避／復帰されます。

(1)割り込み受け付け時

割り込みが受け付けられた場合、以下に示すレジスタがスタック領域に退避されます。

プログラムカウンタ(PC) → ４バイト（最上位１バイトは 00H 固定）
フラグレジスタ(FLG) → ２バイト 合計６バイト

ただし高速割り込みの場合は、フラグレジスタ(FLG)はフラグ退避レジスタ(SVF)に、プログラムカウンタ(PC)は PC 退避レジスタ(SVP)にそれぞれ退避されるため、スタックには何も積まれません。

割り込み処理終了後、REIT 命令によりスタック領域に退避されていた上記のレジスタが復帰されます。

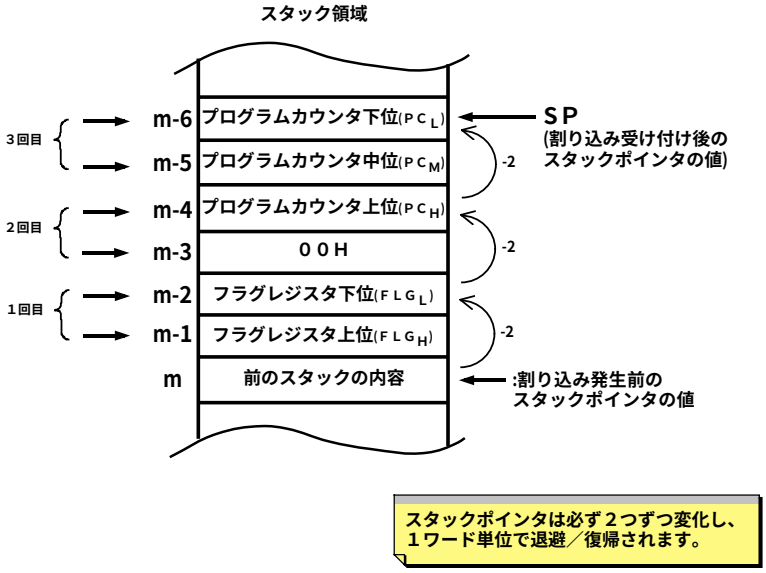


図 4.5.5 割り込み受け付け時のスタック動作と状態

(注) プッシュ、ポップ命令(PUSH,POP,PUSHM,POPM など)でスタックに対して１バイトデータを退避／復帰する場合でも、スタックポインタは必ず２つずつ動きます。

(2)サブルーチン呼び出し時(JSR、JSRI、JSRS 命令実行時)

JSR、JSRI、及び JSRS 命令が実行された場合、以下に示すレジスタがスタック領域に退避されます。

プログラムカウンタ(PC) → 4 バイト（最上位 1 バイトは 00H 固定）

サブルーチン終了後、RTS 命令によりスタック領域に退避されていた上記のレジスタが復帰されます。

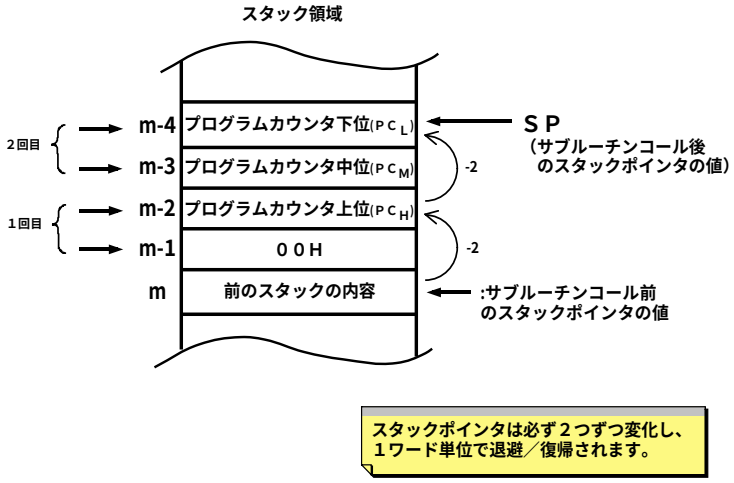


図 4.5.6 サブルーチン呼び出し時のスタック動作と状態

4.5.4 スペシャルページの使用

M16C/80 シリーズでは、固定ベクタの中に"スペシャルページベクタテーブル"と呼ばれる予約領域があり、各ベクタには"スペシャルページ番号"がついています。(2.1.3 固定ベクタ領域 参照)

このベクタにサブルーチン、あるいはジャンプ先のアドレスを格納しておき、スペシャルページサブルーチンコール命令(JSRS)、またはスペシャルページジャンプ命令(JMPS)でスペシャルページ番号を指定することにより、通常のサブルーチンコール命令(JSR)や、ジャンプ命令(JMP)(注)に比べ、少ないバイト数で分岐させることが可能となります。

これによりプログラムステップ数、及び ROM サイズを削減することができます。

スペシャルページの使用例

以下にスペシャルページを使用したサブルーチンコールのプログラム例を示します。

```
;********************* プログラム領域 ****************************************
;================== スタートアップ ==================================
;
    .SECTION     PROGRAM,CODE               ;プログラムセクション
    .ORG         0FF0000H                   ;プログラムの配置アドレスを指定
START:
            LDC     #RAM_END+1,ISP          ;スタックポインタ(ISP)の初期値設定
 ;          ・
 ;          ・
 ;
;================= メインプログラム =====================================
MAIN:
            JSR             INIT
MAIN_10:
            JSRS            #255            ;スペシャルページサブルーチンコールによる
;                  ・                         ;サブルーチン"SUB_1"の呼び出し
;                  ・
            JSRS            #251            ;スペシャルページサブルーチンコールによる
;                  ・                         ;サブルーチン"SUB_2"の呼び出し
            JMP     MAIN_10
;
;================= INIT ルーチン ==========================================
INIT:
            MOV.B           #0FFH,WORK_1
            MOV.B           #0FFH,WORK_2
;           ・
;           ・
INIT_END:
            RTS
;
```

スペシャルページによるサブルーチンコールでは飛び先の最上位アドレスが"FFH"固定となるため、プログラム(サブルーチン)は"FF0000H"～"FFFFFFH"に配置すること

通常のサブルーチンコール命令による呼び出し

サブルーチン"SUB_1"の先頭アドレスを格納したスペシャルページベクタ番号(255番)を指定して呼び出す

サブルーチン"SUB_2"の先頭アドレスを格納したスペシャルページベクタ番号(251番)を指定して呼び出す

注）分岐距離指定子が".S", ".B"の場合は、通常のジャンプ命令の方がコードサイズは小さくなります。

```
;================= SUB_1 ルーチン ====================================
SUB_1:
        MOV.B       WORK_1,R0L
        INC.B       R0L
;               •
;               •
SUB_1_END:
        RTS
;
;================= SUB_2 ルーチン ====================================
SUB_2:
        MOV.B       WORK_2,R1L
        DEC.B       R1L
;               •
;               •
SUB_2_END:
        RTS
;
;********** 固定ベクタ（スペシャルページベクタと固定割
;
  .SECTION      F_VECT,ROMDATA
  .ORG          0FFFE00H                ;スペシャルページベクタ 255 番のアドレス
                                        ;を指定（固定ベクタ領域の先頭番地）
  .WORD         SUB_1 & 00FFFFH         ;サブルーチン "SUB_1" の先頭番地を設定
;
  .ORG          0FFFE08H                ;スペシャルページベクタ 251 番のアドレス
                                        ;を指定
  .WORD         SUB_2 & 00FFFFH         ;サブルーチン "SUB_2" の先頭番地を設定
;
  .ORG          0FFFFDCH                ;固定割り込みベクタの先頭番地
  .LWORD            dummy               ;
  .LWORD            dummy               ;
  .LWORD            dummy               ;
  .LWORD            dummy               ;
  .LWORD            dummy               ;未使用
  .LWORD            dummy               ;監視タイマ割り込みベクタ
  .LWORD            dummy               ;未使用
  .LWORD            dummy               ;NMI 割り込みベクタ
  .LWORD            START               ;リセットベクタ設定
;
  .END
```

スペシャルページによる呼び出しをする場合、アドレスの最上位１バイト(bit16～bit23)は "FFH" 固定となるため、サブルーチンは FF0000H～FFFFFFH 番地内に配置すること！

ラベルはアセンブラにより "３バイト" に展開されるため、アセンブラの演算子"&"(論理積)により最上位1バイトをマスクし、アドレスの下位２バイトをスペシャルページベクタに設定する

スペシャルページベクタは、１ベクタ "２バイト" で構成されるため、スペシャルページベクタ２５１番は、"FFFE08H" 番地となる

図 4.5.7　スペシャルページサブルーチンコールの使用例

4.5.5 ソフトウエア割り込み（INTO 命令）の使用例

*INTO命令（オーバーフロー割り込み)は、フラグレジスタ(FLG)内の**オーバーフローフラグ(O)が"1"にセットされている状態で実行した場合**に割り込みが発生するソフトウエア割り込み命令です。*

よってINTO命令は、除算命令(DIV,DIVUなど)や積和演算命令(RMPA)で、演算結果がオーバーフローした場合の処理ルーチン呼び出し命令として使用することができます。

図4.5.8にINTO命令の使用例を示します。

INTO 命令の使用例

```
;********************* プログラム領域 ************************************************
;================= メインプログラム ====================================
MAIN:
        JSR     INIT                ;初期設定ルーチン
MAIN_10:
        MOV.L   DATA1, R2R0
        DIV.W   #5                  ;符号付き除算
        INTO                        ;オーバーフロー割り込み
        MOV.W   R0, ANS_DAT1
;           •
;           •
        MOV.L   #0, R2R0
        MOV.W   #0, R1
;
        MOV.L   #10000H, A0         ;被乗数格納アドレスの設定
        MOV.L   #20000H, A1         ;乗数格納アドレスの設定
        MOV.W   #0FFH, R3           ;積和回数の設定
        RMPA.W                      ;積和演算
        INTO                        ;オーバーフロー割り込み
;
        MOV.L   R2R0, ANS_DAT2
        MOV.W   R1, ANS_DAT2+4
;           •
;           •
        JMP     MAIN_10
;
;================= INIT ルーチン ========================================
INIT:
        MOV.W   #0FFFFH, DATA1
        MOV.W   #0, ANS_DAT1
;       •
;       •
INIT_END:
        RTS
;
```

除算結果がオーバーフロー(O フラグ＝1)した場合にのみ、INTO命令により割り込みが発生し、それ以外は割り込みは発生せず、次の命令を実行する。

積和演算中にオーバーフロー(O フラグ＝1)が発生した場合、命令の実行を中断して次の命令(INTO 命令)を実行するため、INTO命令により割り込みが発生する。

```
;================== オーバーフロー割り込み処理ルーチン =====================
INT_OVER_FLOW:
        FSET        B
;               •
;               演算結果オーバーフロー時の処理
;               •
INT_OVER_FLOW_END:
        REIT
;
;================== ダミー割り込みプログラム ================================
dummy:
        REIT
;
;********** 固定ベクタの設定 **************
;
  .SECTION    F_VECT,ROMDATA
  .ORG        0FFFFDCH                    ;固定割り込みベクタの先頭番地
;
  .LWORD          dummy           ;未定義命令割り込みベクタ
  .LWORD          INT_OVER_FLOW   ;オーバフロー（INTO）割り込みベクタに
                                  ;割り込み処理の先頭アドレスを設定
  .LWORD          dummy           ;BRK 命令割り込みベクタ
  .LWORD          dummy           ;アドレス一致割り込みベクタ
  .LWORD          dummy           ;未使用
  .LWORD          dummy           ;監視タイマ割り込みベクタ
  .LWORD          dummy           ;未使用
  .LWORD          dummy           ;NMI 割り込みベクタ
  .LWORD          START           ;リセットベクタ設定
;
  .END
```

オーバーフロー割り込み(INTO命令)のベクタは、固定ベクタになるので注意すること

図 4.5.8　INTO（ソフトウエア割り込み）命令の使用例

4.5.6　S/W 暴走対策

この項では、監視タイマやソフトウエア割り込み命令等を使用したソフトウエアによるプログラムの暴走対策について説明します。

監視タイマの使用

監視タイマとは15ビットで構成されたタイマで、プログラムの暴走検出用のタイマとして使用します。プログラムが暴走した場合、監視タイマがアンダーフローし、割り込みが発生します。この監視タイマ割り込み処理の中で、ソフトウェアリセット等によりプログラムを再スタートさせることができます。

監視タイマ割り込みはノンマスカブル割り込みです。リセット解除後は監視タイマは停止していますが、監視タイマスタートレジスタへの書き込みによりカウントを開始します。　なお監視タイマは、リセット時、監視タイマスタートレジスタへの書き込み動作時、及び監視タイマ割り込み要求発生時に初期化されます。

暴走検出方法

暴走検出時の動作フロー、および検出方法を以下に示します。

(1)動作フロー

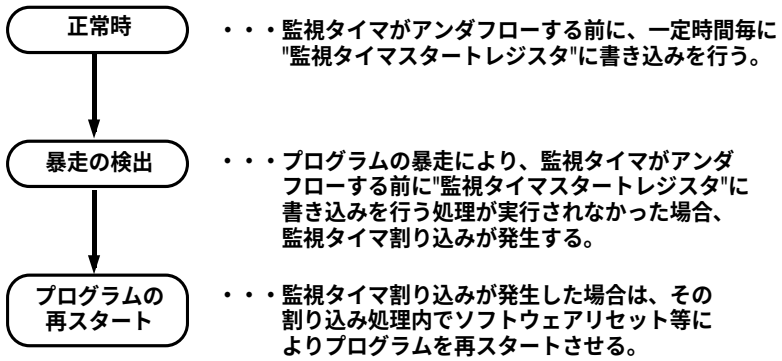


図 4.5.9　暴走検出時の動作フロー

(2)暴走の検出方法

監視タイマがアンダフローする前に監視タイマスタートレジスタに書き込みを行うようにプログラムします。監視タイマスタートレジスタに任意データを書き込むことによって、監視タイマに自動的に"7FFFH"がセットされます。(任意の値は設定できません)

書き込み動作を何ヶ所にも入れた場合、プログラムが暴走したにも関わらず、暴走した先で書き込み動作が行われ暴走が検出されない場合が考えられます。したがって、書き込みを行う箇所はメインルーチンなど必ず実行される箇所１ヶ所に入れるようにしてください。ただしメインルーチンの長さ、または割り込み処理の長さなどを考え、監視タイマ割り込みが発生する前に監視タイマスタートレジスタに書き込みを行うようにしてください。

(3)暴走したプログラムの再スタート

割り込み処理ルーチン内で、プロセッサモードレジスタ0のビット3(ソフトウェアリセットビット)に"1"を書き込むようにプログラムします。これによりソフトウェアリセットが発生し、プログラムはリセット状態から再スタートします。（この時内部 RAM の内容はリセット直前の状態で保持されます。）

また、あらかじめ監視タイマ割り込みの割り込みベクタにこの割り込み処理プログラムの先頭番地を設定しておいてください。

なお、プログラムをリセット状態から再スタートさせる時は、必ず " ソフトウェアリセットビット " によりリセットを行うようにしてください。監視タイマ割り込みの割り込みベクタにリセットベクタと同じアドレス値を設定すると、IPL(プロセッサ割り込み優先レベル)が "7" のままクリアされません。したがって、プログラムが再スタートしても他のすべての割り込みが禁止されてしまいますので、ご注意ください。

暴走検出プログラム例

図 4.5.10 と図 4.5.11 に監視タイマをプログラム暴走の検出に使用した場合のプログラム例を示します。

例 1）監視タイマスタートレジスタへの書き込み動作(サブルーチンを)、一定時間ごとに実行する

```
WDT_SET:
        MOV.B     R0L, WDTS          ;監視タイマスタートレジスタへの書き込み
        RTS
```

".EQU" でアドレスを定義しておく

監視タイマスタートレジスタに任意の値は設定できないので、R0L の値は不定でよい

図 4.5.10　暴走検出プログラム例 1

例 2）監視タイマ割り込みが発生した場合、システムを再スタートする割り込み処理プログラムを実行する

```
WDT_INT:
        LDC     #000380H,SB   注 1)   ;SB,FB レジスタの再設定
        LDC     #000500H,FB
;
        BSET    1, PRCR               ;プロセッサモードレジスタ 0、1 への書き込み許可
                                      ;(プロテクト解除)
        BSET    3, PM0                ;ソフトウェアリセット
;
        REIT    注 2)
        •
        •
        •
  .SECTION      VECT,ROMDATA
  .ORG          0FFFFF0H
  .LWORD        WDT_INT               ;監視タイマ割り込みベクタに割り込み処理の先頭番
                                      ;地を設定しておく
        •
        •
```

".EQU" でアドレスを定義しておく

プロテクトを解除してからソフトウェアリセットビットを "1" にセットし、ソフトウェアリセットをかける

注 1) 暴走時ベースレジスタ(SB,FB)の内容は保証されない。従って、SFR へ書き込む前に値を再設定する必要がある。
注 2)ソフトウェアリセットビットを "1" にセットした直後にリセットシーケンスに入る。よってそれ以降の命令は実行されない。

図 4.5.11　暴走検出プログラム例 2

ソフトウエア割り込み（UND ／ BRK 命令）の使用

"BRK 命令"、"UND 命令 " は共に命令を実行すると割り込みが発生するソフトウエア割り込み命令です。これらの命令を使用して、プログラムの暴走を検出することもできます。　以下に検出方法を示します。

暴走検出方法

BRK または UND 命令を、ROM のプログラム領域として使用している以外の領域に、あらかじめ埋め込んでおきます。これにより、プログラムが暴走し ROM の未使用領域をアクセスした場合、格納されている "UND" または "BRK" 命令をフェッチするため、その時点で割り込みが発生し、暴走を検出することができます。

また、使用していない割り込みベクタに " ダミー割り込み処理 " の先頭番地を格納しておくことにより、未使用の割り込みが万一発生してしまった場合の暴走を防ぐことも可能です。記述例については、4.3.6　サンプルプログラム 3 (割り込みの使用)を参照ください。

なお、暴走したプログラムの再スタート方法、及び発生した割り込みのプログラム例については、「監視タイマ」を使用した場合と同様に行ってください。

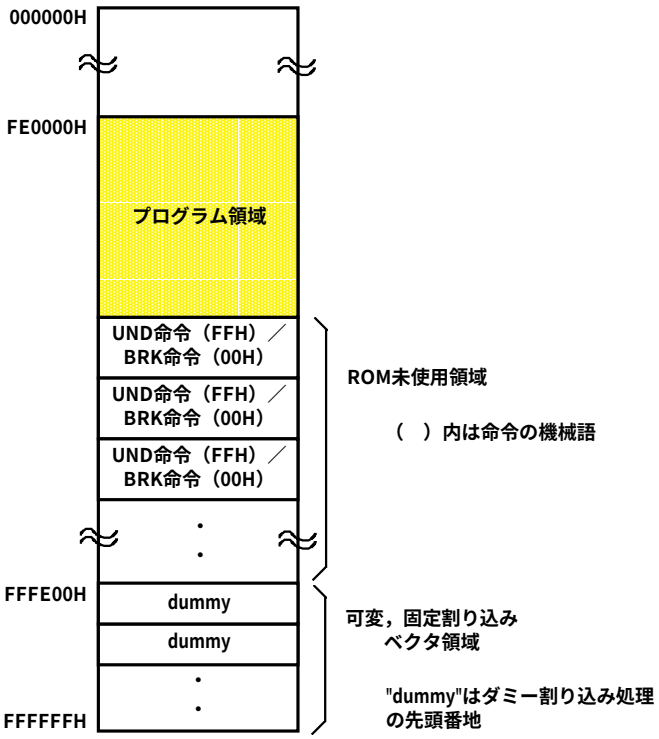


図 4.5.12　ソフトウエア割り込み命令による暴走検出

4.5.7 "-LOC" オプションの使用方法について

この項では、M16C/80シリーズ用アセンブラシステムに含まれるリンカ(LN308)のオプション"-LOC"(セクションデータの配置指定）の使用方法について説明します。

"-LOC" オプションについて

"-LOC"オプションは、指定したセクション内データの格納先アドレスを指定するオプションで、モジュールを実行時とは別の領域に格納する場合に使用します。(注)

したがって、指定されたセクション内のラベルの値（アドレス値）は、ソースファイル内で ".ORG" により指定されたアドレス、またはリンク時にリンカオプション "-ORDER" で指定されたアドレスを基準に生成します。

使用例

"FE0000H" 番地に格納されているセクション名 "PROGRAM" を、実行前に "1000H" 番地に転送した後、"1000H" 番地からプログラムを実行する場合の例を以下に示します。

記述例）>LN308 -ORDER PROGRAM=1000 **-LOC PROGRAM=0FE0000** SAMPLE･･･

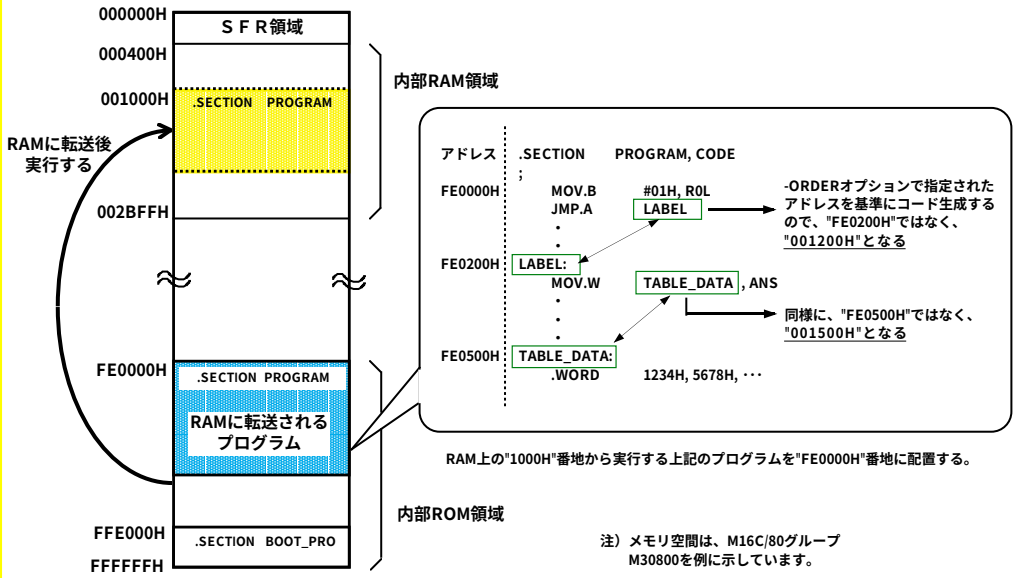


図 4.5.13 -LOC オプションによるセクションデータの配置指定例

(注)M16C/80 シリーズの FlashROM 内蔵版で、CPU 書き換えモードにより内蔵の FlashROM にプログラムを書き込む際、RAM 上でプログラム(FlasahROM への書き込みプログラムなど)を動作させるような場合に使用します。

4.6　定石処理プログラム

この節では M16C/80 シリーズのプログラミングにおける定石処理の例を示します。　合わせて、アプリケーションノート「M16C/80 シリーズ参考プログラム集」をご利用ください。

1．指定したビットの状態による条件分岐

```
        BTST        0, WORK_1
        JC ( /JGEU )    LABEL1      ;指定したビットが "1" の場合 LABEL1 に分岐
            •
            •
LABEL1:
        BTST        1, WORK_1
        JNC ( /JLTU )   LABEL2      ;指定したビットが "0" の場合 LABEL2 に分岐
            •
            •
LABEL2:
;
```

2 命令で条件分岐される

図 4.6.1　指定したビットの状態による条件分岐プログラム例

2．データテーブルの検索

```
        MOV.W   #1,A0
        LDE.B   DATA_TABLE[A0],R0L  ;データテーブルの 2 バイト目(34H)を R0L に格納
            •
            •
            •
            •
DATA_TABLE:
  .BYTE  12H,34H,56H,78H            ;1 バイトデータの設定
;
```

アドレスレジスタ相対アドレッシングを用いて行う。
テーブルの先頭番地をベースアドレスとし、そこからの相対アドレス値をアドレスレジスタ(A0,A1)に代入する。これにより、アドレスレジスタをパラメータとしてテーブル検索を行うことができる。

図 4.6.2　テーブル検索プログラム例

3．テーブルジャンプによるサブルーチン呼び出し

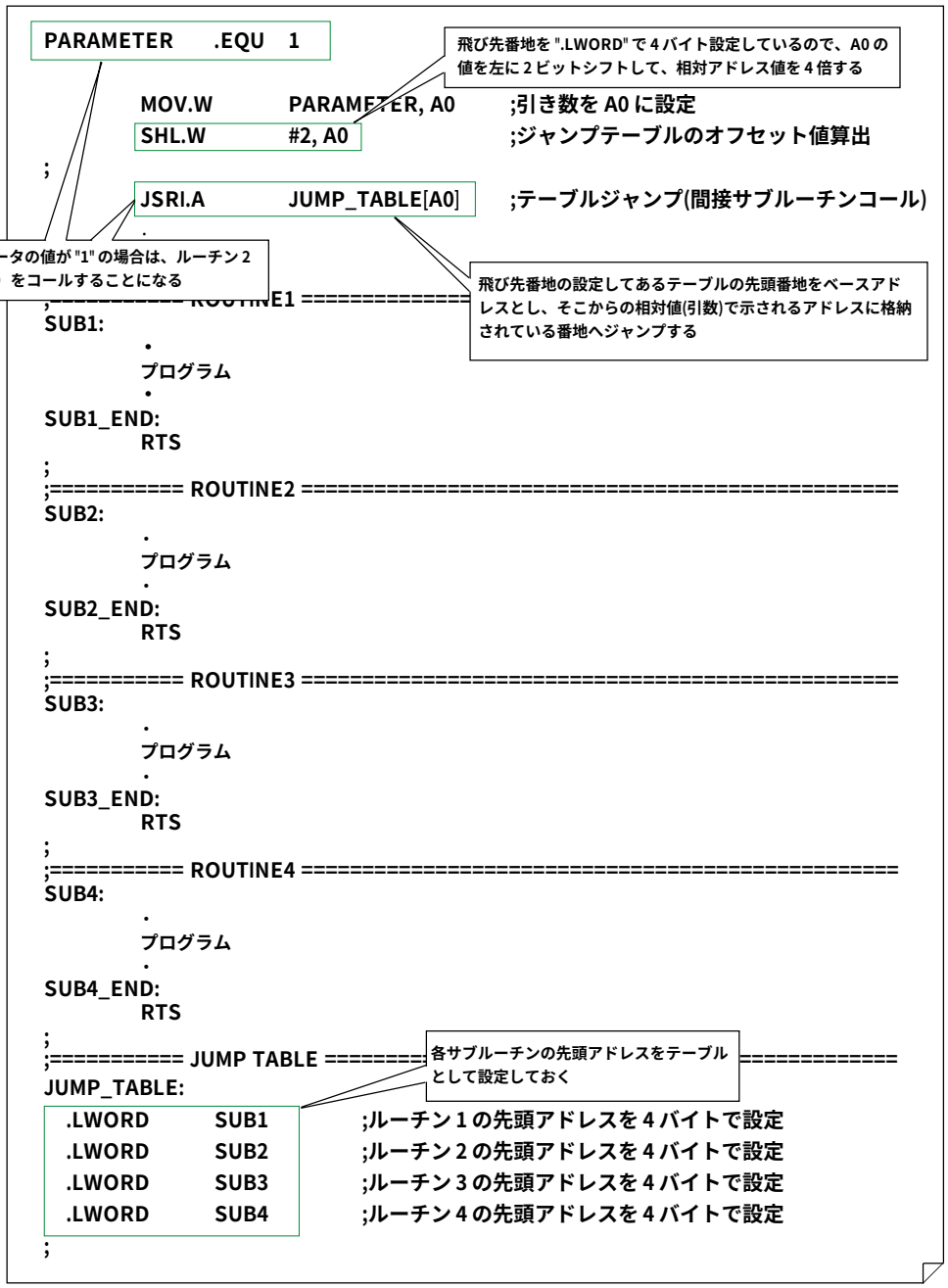


図 4.6.3　テーブルジャンプによるサブルーチン呼び出し例

付録

AS308 システムのコマンド入力形式とコマンドパラメータ

オブジェクトファイルの生成

AS308 システムはアセンブラ(as308)、リンケージエディタ(ln308)、ロードモジュールコンバータ(lmc308)、その他のツール(lb308、abs308、xrf308)から構成される開発支援ツールです。AS308 システムを使用して、オブジェクトファイルを生成する方法を説明します。

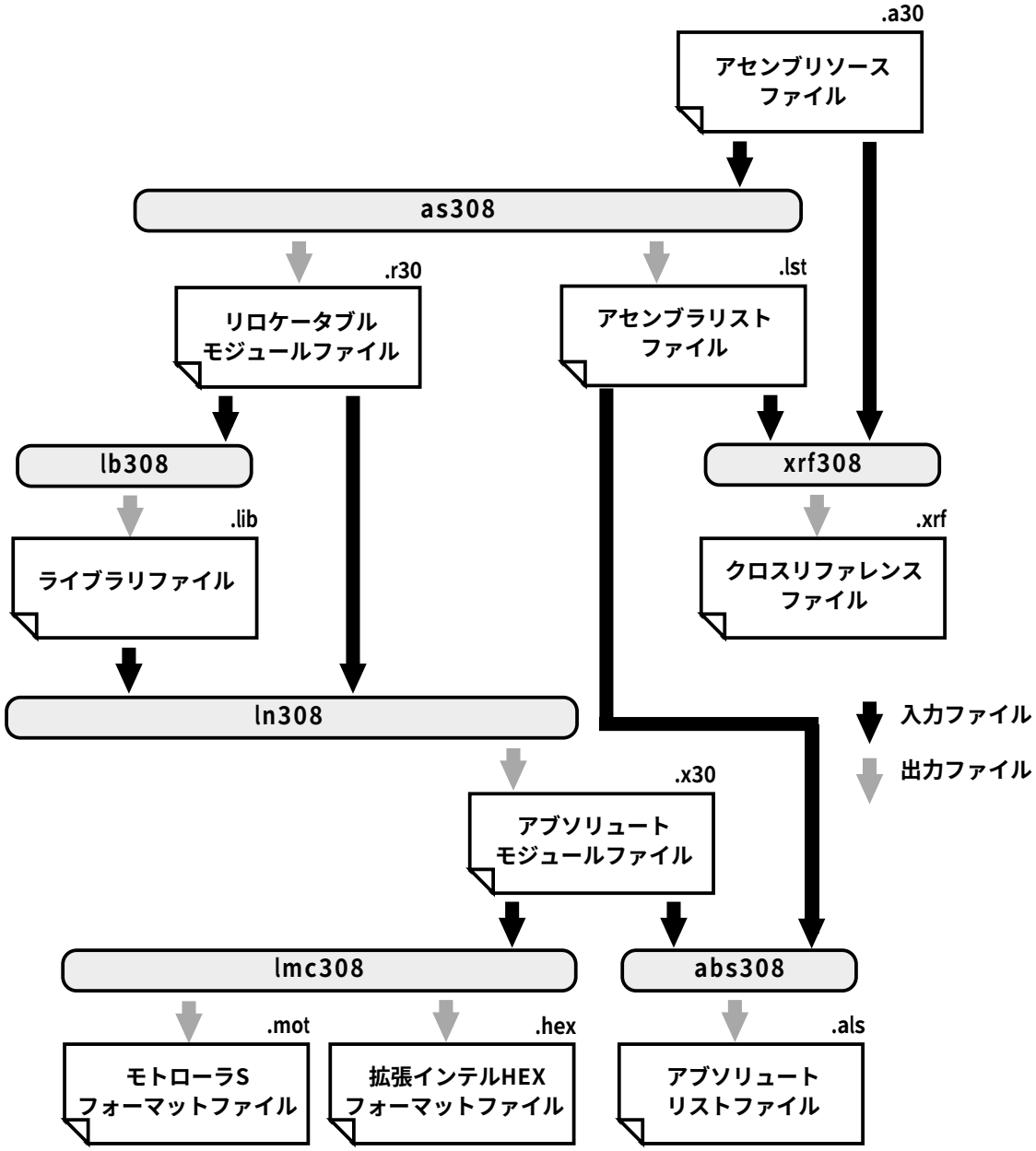


図 A.1　AS308 処理概要

(注)本書ではAS308システム全体を意味する場合は"AS308システム"(大文字)と記述します。アセンブラ(as308)のみを意味する場合は"as308"(小文字)と記述します。

A-1　アセンブル（as308）

リロケータブルアセンブラ as308 の生成ファイルと起動方法について説明します。

as308 の生成ファイル

(1)リロケータブルモジュールファイル(*.R30)　・・・ 随時生成**

IEEE-695 に準拠したファイルで、機械語データとその再配置情報を格納したファイルです。

(2)アセンブラリストファイル(*.LST)　・・・'-L' オプション指定時**

リスト行、ロケーション情報、オブジェクトコード、行情報を含んだプリントアウト用ファイルです。

(3)アセンブラエラータグファイル(*.ATG) ・・・'-T' オプション指定時**

ソースファイルをアセンブルする際に発生したエラーメッセージを格納したファイルです。エラーが発生しなかった場合は生成されません。このファイルはタグジャンプ機能を持つエディタで利用すると、エラー修正を容易に行えます。

as308 起動方法

>as308　ファイル名 . 拡張子　〔ファイル名 . 拡張子・・・〕〔オプション〕

ファイル名は必ず 1 個以上記述してください。拡張子(.A30)は省略できます。

表 A.1　as308 のコマンドオプション

コマンドオプション	機能
-.	アセンブル処理のメッセージを出力しません。
-A	ニーモニックオペランドの評価をします。
-abs16	16ビット絶対アドレッシングモードを指定する。 ※ 本オプションは必ず小文字で入力します。
-C	as308がmac308とasp308を各々起動したときのコマンドオプションを画面に表示します。
-Dシンボル名=定数	シンボル定数を設定します。
-F 展開ファイル名	指示命令..FILEの展開ファイルを固定します。
-H	アセンブラリストファイルのヘッダ情報を出力しません。
-I	インクルードファイルの検索ディレクトリを指定します。
-L	-L　アセンブラリストファイルを生成します。 -LC　行連結をそのままリストファイルに出力します。 -LD　.DEFINEを置き換える以前の情報をリストファイルに出力します。 -LI　条件アセンブルで偽となった部分もリストに出力します。 -LM　マクロ記述の展開部分もリストに出力します。 -LS　AS30用構造化記述命令の展開行をアセンブラリストファイルに出力します。
-mod60	AS30用に記述されたプログラムの命令の一部をAS308用に置き換えます。　※ 本オプションは必ず小文字で入力します。
-mod60p	AS30用構造化記述命令を処理します。 ※ 本オプションは必ず小文字で入力します。
-N	高級言語のソース行情報をリロケータブルモジュールファイルに出力しません。
-Oディレクトリパス名	アセンブラが生成するファイルのディレクトリを指定します。 O(オウ)とディレクトリ名の間にはスペースを記述しないでくだい。 (デフォルトはカレントディレクトリ)
-S	ローカルシンボル情報をリロケータブルモジュールファイルに出力します。
-T	アセンブルエラータグファイルを生成します。
-V	アセンブラシステムの各プログラムのバージョンを表示します。
-Xプログラム名	エラータグファイルを生成し、'-X'に続けて指定したプログラムを起動します。

as308 コマンド使用例

例)

各オプションはスペースで区切る

拡張子は省略すると ".A30"

```
>as308 -L -Oc:\work SAMPLE
```

SAMPLE.A30 から SAMPLE.LST と SAMPLE.R30 を生成し、C ドライブの \work ディレクトリに出力する。

コマンドオプションは大文字小文字どちらでもよい

```
>as308 -s -t  sample
```

SAMPLE.A30のローカルシンボル情報をSAMPLE.R30に出力し、エラータグファイル(SAMPLE.ATG)を生成する。

アセンブラリストファイル

アセンブラリストファイルの例を図 A.2 に示します。

リストの行番号
ロケーションアドレス
オブジェクトコード
as308 のアセンブル処理情報
アセンブリソース行

```
* M16C/80 SERIES ASSEMBLER *  SOURCE LIST    Wed Mar 24 16:04:39 1999  PAGE 001

SEQ.  LOC.          OBJ.       0XMSDA  .*....*....SOURCE STATEMENT....8....*....9....*....0....*....1....*....2....*....3....*....4
  1                                     ;""Sample List""
  2                                     ;********************* ｲﾝｸﾙｰﾄﾞ ***********************************************
  3                                     ;
  4                                             .INCLUDEM30800.INC
  5       1                             ;---------------------------------------------------------------
  6       1                              ;                    M30800 SFR定義ﾌｧｲﾙ
  7       1                              ;---------------------------------------------------------------
  8       1                                     .LIST     OFF
  9       1                                     .LIST     ON
 10                                     ;
 11                                     ;**********                                                  ******
 12                                     ;
 13  00000400h                          RAM_TOP           .EQU    000400H             ;RAMの先頭ｱﾄﾞﾚｽ
 14  00002BFFh                          RAM_END           .EQU    002BFFH             ;RAMの終了ｱﾄﾞﾚｽ
 15  00FE0000h                          ROM_TOP           .EQU    0FE0000H            ;ROMの先頭ｱﾄﾞﾚｽ
 16  00FFFFDCh                          FIXED_VECT_TOP    .EQU    0FFFFDCH            ;固定ﾍﾞｸﾀの先頭ｱﾄﾞﾚｽ
 17  00000400h                          SB_BASE           .EQU    000400H             ;SB相対のﾍﾞｰｽｱﾄﾞﾚｽ
 18  00000580h                          FB_BASE           .EQU    000580H             ;FB相対のﾍﾞｰｽｱﾄﾞﾚｽ
 19  00000300h                          ISP_SIZE          .EQU    300H                ;割り込みｽﾀｯｸ領域のｻｲｽﾞ
 20                                     ;
 21                                     ;********************* ﾜｰｸRAM 領域確保 ***************************************
 22                                     ;
 23                                             .SECTION          WORK,DATA
 24  000400                                     .ORG              RAM_TOP
 25                                     ;
 26  000400                             WORKRAM_TOP:
 27  000400(000001H)                    char:             .BLKB   1                   ;1ﾊﾞｲﾄ領域確保
 28  000401(000002H)                    short:            .BLKW   1                   ;2ﾊﾞｲﾄ領域確保
 29  000403(000003H)                    addr:             .BLKA   1                   ;3ﾊﾞｲﾄ領域確保
 30  000406(000004H)                    long:             .BLKL   1                   ;4ﾊﾞｲﾄ領域確保
 31  00040A                             WORKRAM_END:
 32                                     ;
```

インクルードファイルのネストレベルを示す

```
 33                                 ;********************* ﾋﾞｯﾄｼﾝﾎﾞﾙ定義 ***************************************
 34                                 ;
 35 0,00000400h                     char_b0         .BTEQU  0,char          ;charのﾋﾞｯﾄ0
 36 1,00000401h                     short_b1        .BTEQU  1,short         ;shortのﾋﾞｯﾄ1
 37 2,00000403h                     addr_b2         .BTEQU  2,addr          ;addrのﾋﾞｯﾄ2
 38 3,00000406h                     long_b3         .BTEQU  3,long          ;longのﾋﾞｯﾄ3
 39                                 ;
 * M16C/80 SERIES ASSEMBLER *  SOURCE LIST    Wed Mar 24 16:04:39 1999  PAGE 002
SEQ.  LOC.   OBJ.          0XMSDA  .*....*....SOURCE STATEMENT....8....*....9....*....0....*....1....*....2....*....3....*....4
 40                                         .PAGE
 41                                 ;********************* ﾌﾟﾛｸﾞﾗﾑ領域 ***************************************
 42                                 ;================== ｽﾀｰﾄｱｯﾌﾟ ================================
 43                                 ;
 44                                         .SECTION        PROGRAM,CODE
 45 FE0000                          ;       .ORG            ROM_TOP
 46 FE0000                          START:
 47 FE0000 D52F002C00                       LDC     #RAM_END+1,ISP          ;ｽﾀｯｸﾎﾟｲﾝﾀ(ISP)の初期値設定
 48                                 ;
 49 FE0005 F6E30A00         Q               MOV.B   #03H,PRCR               ;ﾌﾟﾛﾃｸﾄの解除
 50 FE0009 1504008301       S               MOV.W   #0183H,PM0              ;ﾌﾟﾛｾｯｻﾓｰﾄﾞﾚｼﾞｽﾀ0,1の設定
 51 FE000E 1506000820       S               MOV.W   #2008H,CM0              ;ｼｽﾃﾑｸﾛｯｸ制御ﾚｼﾞｽﾀ0,1の設定
 52 FE0013 140B0012         S               MOV.B   #12H,MCD                ;ﾒｲﾝｸﾛｯｸ分周ﾚｼﾞｽﾀの設定
 53 FE0017 120A00           Z               MOV.B   #0,PRCR                 ;全ﾚｼﾞｽﾀをﾌﾟﾛﾃｸﾄ
 54
  * M16C/80 SERIES ASSEMBLER *
SEQ.  LOC.   OBJ.          0XMSI
 85
 86                                                                                 ==============
 87 FE0066                          MAIN:
 88 FE0066 B88B00E0FF                       MOV.B   DATA_TABLE[A0],R0L
 89 FE006B 99EF3412                         MOV.W   #1234H,R1
 90 FE006F D2B800           *               BSET    char_b0
 91                                 ;                       ・
 92                                 ;                       ・
 93                                 ;                       ・
 94 FE0072 B8AB00E0FF                       MOV.B   DATA_TABLE,R0L
 95 FE0077 BBEE             B               JMP     MAIN
 96                                 ;
 101                                ;=====                                          =====
 102                                ;
 103
 104 FFE000
 105                                ;
 106 FFE000                         DATA_TABLE:
 107 FFE000 12345678                        .BYTE   12H,34H,56H,78H         ;1ﾊﾞｲﾄﾃﾞｰﾀの設定
 108 FFE004 34127856                        .WORD   1234H,5678H             ;2ﾊﾞｲﾄﾃﾞｰﾀの設定
 109 FFE008 563412BC9A78                    .ADDR   123456H,789ABCH         ;3ﾊﾞｲﾄﾃﾞｰﾀの設定
 110 FFE00E 78563412                        .LWORD  12345678H,9ABCDEF0H     ;4ﾊﾞｲﾄﾃﾞｰﾀの設定
             F0DEBC9A
 111 FFE016                         DATA_TABLE_END:
 112                                ;
 126                                ;
 127                                        .END

Information List
TOTAL ERROR(S)    00000
TOTAL WARNING(S)  00000
TOTAL LINE(S)     00127   LINES

Section List
Attr          Size                  Name
DATA          00000010(00000AH)     WORK
CODE          00000122(00007AH)     PROGRAM
ROMDATA       00000022(000016H)     CONSTANT
ROMDATA       00000036(000024H)     F_VECT
```

Z: 命令フォーマットのゼロ形式を選択したことを示す
S: 命令フォーマットのショート形式を選択したことを示す
Q: 命令フォーマットのクイック形式を選択したことを示す
*: 8ビット変位SB相対アドレッシングモードを選択したことを示す。

S: 3ビット相対ジャンプ(分岐距離指定子'S')を選択したことを示す
B: 8ビット相対ジャンプ(分岐距離指定子'B')を選択したことを示す
W: 16ビット相対ジャンプ(分岐距離指定子'W')を選択したことを示す
A: 絶対ジャンプ(分岐距離指定子'A')を選択したことを示す

アセンブルを行った結果の全エラー数、全ワーニング数、全リスト行数を出力する

セクションのタイプ、セクションサイズ、セクション名を出力する

図 A.2　アセンブラリストファイルの例

アセンブルエラータグファイル

アセンブラエラータグファイルの例を図 A.3 に示します。

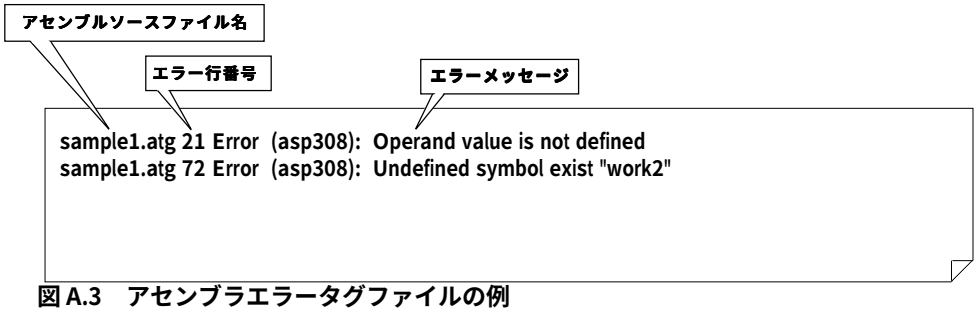


図 A.3　アセンブラエラータグファイルの例

A-2　リンク（ln308）

リンケージエディタ ln308 の生成ファイルと起動方法について説明します。

ln308 の生成ファイル

(1)アブソリュートモジュールファイル(*.X30)　・・・随時生成**

IEEE-695 に準拠したファイルで、as308 が出力したリロケータブルモジュールファイルを１つにまとめたファイルです。

(2)マップファイル(*.MAP)　・・・'-M ／ -MS' オプション指定時**

リンク情報、セクションの最終配置アドレス情報、シンボル情報などを含んだファイルです。シンボル情報はオプション '-MS' を指定したときのみマップファイルに出力します。

(3)リンクエラータグファイル(*.LTG)　・・・'-T' オプション指定時**

リロケータブルモジュールファイルをリンクする際に発生したエラーメッセージを格納したファイルです。エラーが発生しなかった場合は生成されません。このファイルはタグジャンプ機能を持つエディタで利用すると、エラー修正を容易に行えます。

ln308 起動方法

>ln308 リロケータブルファイル名 〔リロケータブルファイル名･･･〕〔オプション〕
ファイル名は必ず１個以上記述してください。拡張子(.R30)は省略できます。

表 A.2　ln308 のコマンドオプション

コマンドオプション	機能
-.	画面へのリンク処理のメッセージを出力しません。
-E　アドレス値	アブソリュートモジュールファイルの開始アドレスを設定します。オプション記号とアドレス値の間には必ずスペースを入れ、アドレス値はラベル名か16進数で記述します。
-G	ソースデバッグ情報をアブソリュートモジュールファイルに出力します。
-L　ライブラリファイル	リンク時に参照するライブラリファイルを指定します。
-LD　パス名	ライブラリファイルのディレクトリを指定します。
-LOC	指定されたセクションデータを指定したアドレスから配置します。
-M	マップファイルを生成します。マップファイル名はアブソリュートモジュールファイルの拡張子を .map に変更した名前になります。
-MS	シンボル情報を含むマップファイルを生成します。
-MSL	16文字を越えるシンボルをそのままマップファイルに出力します。
-NOSTOP	発生したエラーを全て画面に出力します。指定しない場合はエラーを最大20個画面に出力します。
-O　アブソリュートファイル名	アブソリュートモジュールファイル名を指定します。ファイルの拡張子は省略できます。省略した場合は .x30 になります。
-ORDER	セクションの配置順序及び開始アドレスを指定します。開始アドレスを指定しない場合は、0からアドレスを配置します。
-T	リンクエラータグファイルを出力します。
-V	バージョンを画面に表示し、そのまま終了します。
@コマンドファイル名	指定したファイルの内容をコマンドパラメータとして起動します。@とコマンドファイル名の間にはスペースを記述しません。他のオプションとは同時に使用できません。

ln308 コマンド使用例

例)
>ln308 SAMPLE1 SAMPLE2 -O ABSSMP
ABSSMP.X30 を生成する。

>ln308 @cmdfile
cmdfile の内容をコマンドパラメータとして ln308 を起動する。

```
#cmdfile の記述例
SAMPLE1 SAMPLE2          # リロケータブルファイル名
SAMPLE3                  # リロケータブルファイル名
-ORDER RAM=400           #RAM セクションの開始アドレスを 400H に指定
-ORDER PROG=0FE0000,SUB,DATA   # セクションを配置する順序を指定
-M                       # マップファイル生成コマンドオプション
```

拡張子 ".R30" は省略できる

コマンドオプションは大文字小文字どちらでもよい

アドレス値は16進数で記述する。先頭の数値がアルファベットの場合は先頭に `0` を付ける。16 進数を示す `H` は付けない。

セクション名は大文字小文字を区別する。複数指定する場合はカンマで区切り、カンマの前後にはスペース又はタブは入力しない。

コメントの先頭には '#' を付ける

リンクエラータグファイル

リンクエラータグファイルの例を図 A.4 に示します。

アセンブルソースファイル名

エラー / ワーニング行番号

エラーメッセージ

```
sample1.a30  87 Warning (ln308):  sample1.r30: Section type mismatch 'PRO'
sample1.a30  92 Warning (ln308):  sample1.r30: Absolute-section is written after the
 absolute-section 'PRO'
sample1.a30  92 Error (ln308): sample1.r30: Address is overlapped in 'CODE' section 'PRO'
```

図 A.4　リンクエラータグファイルの例

(注)アブソリュートモジュールファイルは、IEEE-695 に準拠したフォーマットのファイルです。バイナリ形式のため、画面やプリンタに出力したり、編集したりできません。

マップファイル

マップファイルの例を図 A.5 に示します。

```
########################################
# (1) LINK INFORMATION                 #
########################################
C:\MTOOL\BIN\LN308.EXE -MS SMP
# LINK FILE INFORMATION
SMP (SMP.r30)
            Mar 24 16:04:39 1999

########################################
# (2) SECTION INFORMATION              #
########################################
# SECTION           ATR       TYPE        START    LENGTH     ALIGN MODULENAME
WORK                ABS       DATA        000400   00000A     SMP
PROGRAM             ABS       CODE        FE0000   00007A     SMP
CONSTANT            ABS       ROMDATA     FFE000   000016     SMP
F_VECT              ABS       ROMDATA     FFFFDC   000024     SMP
# Total ----------------------------------------------------------------
            DATA            000000A(00000010) Byte(s)
            ROMDATA         000003A(00000058) Byte(s)
            CODE            000007A(00000122) Byte(s)
########################################
# (3) GLOBAL LABEL INFORMATION         #
########################################
work        000000
########################################
# (4) GLOBAL EQU SYMBOL INFORMATION    #
########################################
sym2        000000
########################################
# (5) GLOBAL EQU BIT-SYMBOL INFORMATION #
########################################
sym1        1 000001
########################################
# (6) LOCAL LABEL INFORMATION          #
########################################
@ SMP ( SMP.r30 )
DATA_TABLE        ffe000  DATA_TABLE_END    ffe016  MAIN          fe0066
START     fe0000  WORKRAM_END       00040a  WORKRAM_TOP       000400
addr        000403  char          000400  dummy       fe0079
long        000406  short         000401
########################################
# (7) LOCAL EQU SYMBOL INFORMATION     #
########################################
@ SMP ( SMP.r30 )
AD0         00000380  AD1         00000382  AD2         00000384
AD3         00000386  AD4         00000388  AD5         0000038a
TA2IC       0000006e  TA2MR       00000358  TA3         0000034c
TA3IC       0000008e  TA3MR       00000359  TA4         0000034e
U0MR        00000360  U0RB        00000366  U0TB        00000362
########################################
# (8) LOCAL EQU BIT-SYMBOL INFORMATION  #
########################################
@ SMP ( SMP.r30 )
addr_b2   2 000403  char_b0   0 000400  long_b3   3 000406
short_b1  1 000401
```

リンク情報

セクション情報

グローバルラベル情報
コマンドオプション "-MS" を指定した場合のみ出力される

グローバルシンボル情報
コマンドオプション "-MS" を指定した場合のみ出力される

グローバルビットシンボル情報
コマンドオプション "-MS" を指定した場合のみ出力される

ローカルラベル情報
コマンドオプション "-MS" を指定した場合のみ出力される

ローカルシンボル情報
コマンドオプション "-MS" を指定した場合のみ出力される

ローカルビットシンボル情報
コマンドオプション "-MS" を指定した場合のみ出力される

図 A.5　マップファイルの例

A-3　機械語ファイルの生成（lmc308）

ロードモジュールコンバータ lmc308 の生成ファイルと起動方法について説明します。

lmc308 の生成ファイル

(1)モトローラ S フォーマットファイル(*.MOT)　・・・通常時**
通常生成される機械語ファイルです。

(2)インテル HEX フォーマットファイル(*.HEX)　・・・`-H` オプション指定時**
'-H' オプション指定時に生成される機械語ファイルです。

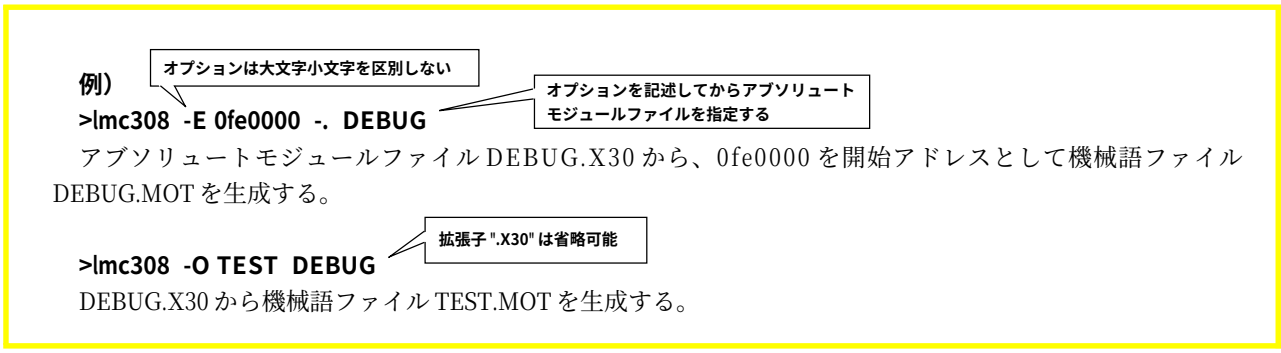
lmc308 起動方法

>lmc308　[オプション]　アブソリュートモジュールファイル名

表 A.3　lmc308 のコマンドオプション

コマンドオプション	機能
-.	エラー及びワーニングメッセージを除くすべてのメッセージを出力しない。
-E　開始アドレス	プログラムの開始アドレスを設定し、モトローラSフォーマットの機械語ファイルを生成する。-Hオプションとは同時に設定できません。
-H	インテルHEXフォーマットの機械語ファイルを生成する。100000Hを越える場合は、三菱専用HEXフォーマットファイルを生成します。 -Eオプションとは同時に設定できません。
-ID	IDコードチェック機能のIDコードを設定します。本オプションで指定されたIDコードを示したIDファイル(拡張子 .id)を生成します。
-L	モトローラSフォーマットのデータレコード長を32バイトに設定する。 インテルHEXフォーマットのデータレコード長を32バイトに設定する。
-O	lmc308が生成する機械語ファイルのファイル名を指定する。ファイル名にはパスの指定ができます。また、本オプションと機械語ファイル名の間には必ずスペースを入れてください。 なお、機械語ファイル名の拡張子は指定できません。(モトローラSフォーマット.mot / インテルHEXフォーマット.hex)
-V	lmc308のバージョンを画面に表示しそのまま終了する。
-PROTECT1	ROMコードプロテクト機能のレベル１を設定する。
-PROTECT2	ROMコードプロテクト機能のレベル２を設定する。

lmc308 コマンド使用例



アブソリュートモジュールファイル DEBUG.X30 から、0fe0000 を開始アドレスとして機械語ファイル DEBUG.MOT を生成する。

DEBUG.X30 から機械語ファイル TEST.MOT を生成する。

改訂記録	M16C/80 シリーズプログラム作成の手引き ＜アセンブリ言語編＞アプリケーションノート

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2003.09.25	－	初版発行

安全設計に関するお願い

- 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

- 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
- 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
- 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ（http://www.renesas.com）などを通じて公開される情報に常にご注意ください。
- 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
- 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
- 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
- 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
- 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。