User's Manual

M16C シリーズ,R8C ファミリ用
C/C++コンパイラパッケージ V.6.00
アセンブラ、最適化リンケージエディタ
ユーザーズマニュアル

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00 2011.01

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。

はじめに

本マニュアルは、「M16C シリーズ、R8C ファミリ用アセンブラ、最適化リンケージエディタ」の使用方法を述べたものです。

本製品はアセンブリ言語で記述したソースプログラムを、M16C シリーズ、R8C ファミリ用オブジェクトプログラムおよびロードモジュールに変換します。

ご使用になる前に、本マニュアルを良く読んで理解してください。

■表記上の注意事項

本マニュアルの説明の中で用いられる記号は、次の意味を示しています。

〈 〉	この記号で囲まれた内容を指定することを示します。
[ ]	省略してもよい項目を示します。
. . .	直前の項目を 1 回以上指定することを示します。
△	1 個以上の空白を示します。
|	|で区切られた項目を選択できることを示します。

本マニュアルはIBM PC*[1]互換機およびその互換機上で動作するWindows® XP、Windows Vista®またはWindows® 7*[2]に対応するように書かれています。

【注】 *1 IBM PC は、米国 International Business Machines Corporation の登録商標です。

*2 Microsoft®, Windows®は、米国 Microsoft Corporation の米国及びその他の国における登録商標または商標です。

※ その他、本マニュアルの文中に使われている会社名および製品名、システム名などは各社の登録商標または商標です。

目次

1. 概要

1.1 アセンブラおよび最適化リンケージエディタの構成

アセンブラおよび最適化リンケージエディタの構成を示します。

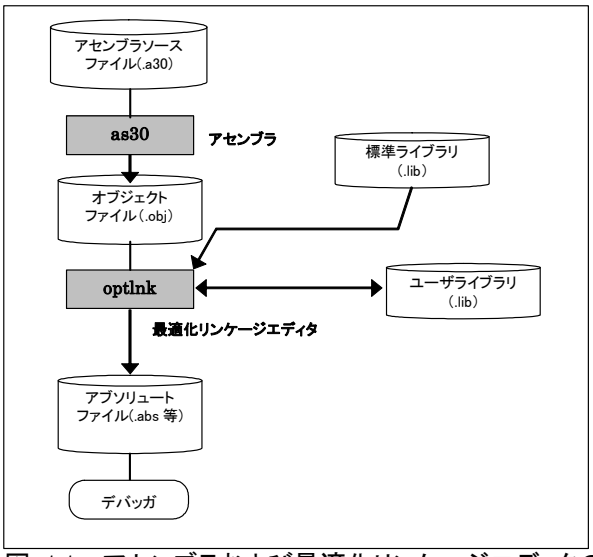


図 1.1 アセンブラおよび最適化リンケージエディタの構成

1.1.1 as30

as30 は、アセンブラの実行ファイルです。
アセンブラソースファイル(.a30)を、オブジェクトファイル(.obj)に変換します。

as30 は、以下のプログラムで構成しています。

- アセンブラドライバ(as30)

  マクロプロセッサ、構造化プリプロセッサ及びアセンブラプロセッサを連続して起動するプログラムです。

- マクロプロセッサ(mac30)

  ソースファイル中のマクロ指示命令を処理しアセンブリ言語ファイルを生成します。マクロプロセッサが生成したアセンブリ言語ファイルは、アセンブラプロセッサの処理終了後に消去されます。ユーザの記述したソースファイルが変更されることはありません。

- 構造化プリプロセッサ(pre30)

  ソースファイル中の構造化記述命令を処理しアセンブリ言語ファイルを生成します。構造化プリプロセッサが生成したアセンブリ言語ファイルは、アセンブラプロセッサの処理終了後に消去されます。ユーザの記述したソースファイルが変更されることはありません。構造化プリプロセッサを起動するには、as30のコマンドオプション(-P)を指定してください。

- アセンブラプロセッサ(asp30)

  マクロプロセッサ、構造化プリプロセッサが前処理を行ったアセンブリ言語ファイルをオブジェクトファイルに変換します。

1.1.2 optlnk

optlnk は、最適化リンケージエディタの実行ファイルです。
複数のオブジェクトファイル(.obj)およびライブラリファイル(.lib)をアブソリュートファイル(.abs 等)、またはライブラリファイル(.lib)に変換します。

1.2 オプション指定規則

以下にアセンブラおよび最適化リンケージエディタで利用できる起動コマンドを説明します。
なお、これらのコマンドを利用する前に、「7. 環境変数」を参照のうえ、必要な環境変数が設定されていることを確認してください。

1.2.1 アセンブラ(as30)

as30 は、アセンブラの起動コマンドです。

【コマンド記述形式】

as30 [△<オプション> …][△<ファイル名>[△<オプション> …] …]

1.2.2 最適化リンケージエディタ(optlnk)

optlnk は、最適化リンケージエディタの起動コマンドです。
リンク処理だけではなく、以下に挙げる機能も含んでいます。

- アブソリュートファイル(.abs等)作成時の最適化
- ライブラリファイルの作成や編集
- モトローラS形式ファイル、インテルHEX形式ファイル、およびバイナリファイルへのコンバート

【コマンド記述形式】

optlnk [△<オプション> …][△<ファイル名>[△<オプション> …] …] 　　<オプション> : －<オプション>[=<サブオプション>][,…]

1.3 バージョンアップ内容と移行方法

旧バージョンからのバージョンアップ内容と、ユーザーアプリケーションを移行する際の方法と注意事項については、「M16C シリーズ,R8C ファミリ用 C/C++ コンパイラパッケージ V.6.00 C/C++ コンパイラユーザーズマニュアル」の「付録 K バージョンアップ内容と移行方法」を参照してください。

2. アセンブラの仕様

2.1 アセンブラの翻訳限界

アセンブラの翻訳限界を以下に示します。

表 2.1 アセンブラの翻訳限界

	項目	翻訳限界
1	1 行文字数	8190 文字
2	シンボル長	1 行文字数*1
3	シンボル数	制限なし(メモリ容量に依存)
4	外部参照シンボル数	制限なし(メモリ容量に依存)
5	外部定義シンボル数	制限なし(メモリ容量に依存)
6	セクションの最大サイズ	0FFFFFH または 0FFFFH バイト
7	セクション数	65265 個(デバッグ情報あり)、65274 個(デバッグ情報なし)
8	ファイルインクルード	ネストは 9 レベル
9	文字列長	1 行文字数*1
10	ファイル名の文字数	1 行文字数*1
11	環境変数設定文字数	2048 バイト
12	マクロ定義数	65535 個

【注】 *1 同じ行に指定した文字列の長さにより、これよりも小さい値となります。

2.2 文字セット

as30 では次の文字セットをサポートしています。ソフトウェアを起動する際のコマンド入力やソースプログラムの記述の際にはこの文字を使用してください。

表 2.2 文字セット

	項目	文字
1	英大文字	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
2	英小文字	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
3	数字	0 1 2 3 4 5 6 7 8 9
4	特殊文字	″ # $ % & ’ ( ) * + , - . / : ; [ ] ¥ ^ _ | ~
5	空白文字	(スペース) (タブ)
6	改行文字	(リターン) (ラインフィード)

3. アセンブラの記述規則

3.1 プログラム記述上の注意事項

as30 を使用する場合には、次に示す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でラベル、シンボル、ビットシンボルなどに使用しないでください。予約語には、拡張用として"IF"、"ENDIF"などが含まれています。
- as30の指示命令からピリオドをとった文字列については、名前に使用してもエラーとなりません。しかし、as30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。
- システムラベル(..で始まる文字列)については、ユーザがソースプログラム内に記述した場合エラーとなりません。しかし、as30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。

3.2 プログラムの記述規則

3.2.1 文字セット

「2. アセンブラの仕様」で示した文字セットでソースプログラムを記述できます。

3.2.2 予約語

as30 は、アセンブル指示命令やニーモニックなどと同一の文字列を予約語として扱います。予約語は特別な機能を持っているため、予約語をソースプログラムの中でラベル名やシンボル名などには使用できません。また、予約語は大文字と小文字を区別しません。"ABS"と"abs"は同じ予約語となります。

予約語には以下のものがあります。

(1) アセンブラ制御命令

本マニュアルで説明している全てのアセンブル指示命令と、ピリオドで始まる文字列は全て予約語です。

(2) ニーモニック

M16C シリーズ、R8C ファミリのニーモニックは全て予約語です。

(3) レジスタ・フラグ名

M16C シリーズ、R8C ファミリのレジスタ名およびフラグ名は全て予約語です。

(4) 演算子

本マニュアルで説明している全ての演算子及び構造化演算子は全て予約語です。

(5) 構造化記述命令

本マニュアルで説明している構造化記述命令は全て予約語です。

(6) システムラベル

アセンブラが生成するラベルをシステムラベルといいます。ピリオド二つで始まる名前は、全てシステムラベルとして扱います。

3.2.3 名前

名前はアセンブリ言語ファイルの中で任意に定義し使用できます。
名前は、次の種類に分けられ、それぞれ記述できる範囲が異なります。

表 3.1 名前の種類

名前の種類	内容
ラベル	アドレスを値として持つ名前です。
シンボル	定数を値として持つ名前です。
ビットシンボル	定数(ビット位置)とアドレスを値として持つ名前です。 8 ビット長のメモリの各 1 ビット毎に名前を付けて区別できます。
セクション	.SECTION 指示命令で定義されるセクションの名前です。
マクロ	マクロの定義名です。
ロケーションシンボル	ロケーションシンボル'$'が記述されている行のオペレーション部の先頭アドレスを示します。

名前の記述規則

- 名前の文字数は「2.1 アセンブラの翻訳限界」に記載した1行文字数になります。
- 名前には英数字、アンダースコア(_)およびダラー($)が使用できます。
- 名前の先頭には、数字は使用できません。
- 名前は大文字と小文字を区別して扱います。"LAB"と"Lab"は異なる名前として扱われます。

【注】 予約語と同一の名前を使用することはできません。万一使用した場合のプログラムの動作については保証いたしません。

(1) ラベル

- ラベルの名前は「名前の記述規則」にしたがいます。
- 定義の際には、名前の最後に必ずコロン(:)を付けてください。
- セクション内で定義できます。
- 指示命令で、領域を確保する際にラベル名を指定できます。

例)
```
flags:    .BLKB     1
work:     .BLKD     1
```

- ソース行の任意の場所にラベル名を記述できます。

例)
```
name1:
     _name:
          sym_name:
```

- ラベルを参照する場合は、命令のオペランドに名前を記述します。

例)
```
JMP     sym_name
```

(2) シンボル

- シンボルの名前は「名前の記述規則」にしたがいます。
- 数値は、アセンブル実行時に確定しなければなりません。
- セクション内またはセクション外で定義できます。
- 数値定義の指示命令".EQU"を使用します。

例)
```
value1    .EQU   1
value2    .EQU   2
```

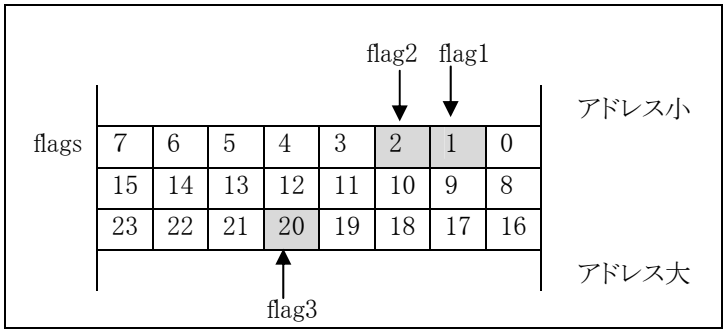
- シンボルを参照する場合は、命令のオペランドに名前を記述します。

例)
```
MOV.W     R0,value1
value3    .EQU    value2+1
```

(3) ビットシンボル

- ビットシンボルの名前は「名前の記述規則」にしたがいます。
- ビット位置を指定する数値は、アセンブル実行時に確定する値を指定します。
- セクション内またはセクション外で定義できます。
- ビットシンボル定義の指示命令".BTEQU"を使用します。

例)
```
flags     .EQU        400H
flag1     .BTEQU      1,flags
flag2     .BTEQU      2,flags
flag3     .BTEQU      20,flags
```



- ビット命令アドレッシング命令のオペランドに記述できます。

例)
```
BCLR      flag1
BCLR      flag2
BCLR      flag3
```

(4) セクション

- セクションの名前は「名前の記述規則」にしたがいます。
- セクションの詳細については、「9.3 リンク制御指示命令」の指示命令".SECTION"を参照してください。

(5) マクロ

- マクロの名前は「名前の記述規則」にしたがいます。
- マクロの詳細については、「9.6 マクロ指示命令」の指示命令".MACRO"を参照してください。

(6) ロケーションシンボルの記述規則

- ニーモニックのオペランドに記述してください。
- ロケーションシンボルを式の項に記述できます。
- 構造化記述文にロケーションシンボルを記述できます。

例)
```
JMP.B    $+5
[lab1]   =   $
[lab1]   =   $+2
```

【注】 ロケーションシンボルをオフセットとするアドレスを分岐命令のニーモニックに記述する場合は、分岐先アドレスまでの全てのニーモニックに対して、最適化が行われないように記述してください。

3.3 行の記述方法

3.3.1 行の種類

as30 は、ソースプログラムを 1 行単位で処理します。各行は記述されている内容によって次のように分類されます。

(1) 指示命令行

- as30 の指示命令を記述した行です。
- 指示命令は、1 行に1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】指示命令とニーモニックを同じ一行に記述できません。

(2) アセンブラソース行

- ニーモニックを記述した行です。
- アセンブラソース行には、コメントを記述できます。
- アセンブラソース行には、先頭にラベル名を記述できます。

【注】 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できません。

(3) ラベル定義行

- ラベル名だけを記述した行です。

(4) コメント行

- コメントだけを記述した行です。

(5) 空行

- スペース、タブ又は改行コードだけを含む行です。

3.3.2 行の記述規則

(1) 行の区切り

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。

(2) 行の長さ

「表 2.1 アセンブラの翻訳限界」を参照してください。
なお最大文字数を越えた文字については、処理しません。
【注】 各行の記述は、必ず一行の範囲に記述してください。

3.3.3 指示命令行の記述規則

- 指示命令とそのオペランドの間に、必ずスペース又はタブを記述してください。
- オペランドを複数個記述する場合は、オペランドとオペランドの間に、必ずカンマ(,)を記述してください。
- オペランドとカンマの間には、スペース又はタブを記述できます。
- 指示命令によっては、オペランドを記述しないものがあります。
- 指示命令は行の先頭から記述できます。
- 指示命令行の先頭には、スペース又はタブを記述できます。
- 指示命令行にコメントを記述する場合は、指示命令とオペランドのつぎに、セミコロン(;)を記述し、それ以降の桁にコメントを記述してください。コメントは、アセンブラリストファイルに出力されます。
- 指示命令のオペランドとコメントの間にはスペース又はタブを記述できます。

【注】 as30 は、セミコロン(;)以降の桁に記述した内容は全てコメントとして処理を行います。セミコロン以降の桁に記述したニーモニックや指示命令について、アセンブラはコード生成を行いません。セミコロン(;)の記述位置には注意してください。as30 は、ダブルクォーテーション(")又はシングルクォーテーション(')で囲まれたセミコロン(;)は、コメントの先頭文字と判断しません。

記述例)

```
                .SECTION   ram,DATA
                .ORG    00H
sym             .EQU    0
work:           .BLKB   1
```

3.3.4 アセンブラソース行の記述規則

ニーモニックの記述方法は「ソフトウエアマニュアル」を参照してください。
ここでは、as30 で処理可能なアセンブラソース行の記述規則を説明します。

- ニーモニックと、そのオペランドの間には必ずスペース又はタブを記述してください。
- オペランドを複数個記述する場合は、オペランドとオペランドの間には必ずカンマ(,)を記述してください。
- オペランドとカンマの間には、スペース又はタブを記述できます。
- ニーモニックによっては、オペランドを記述しないものがあります。
- ニーモニックは行の先頭から記述できます。
- アセンブラソース行の先頭には、スペース又はタブを記述できます。
- アセンブラソース行でラベルを定義する場合は、必ずニーモニックよりも前の桁にラベル名を記述してください。
- ラベル定義のラベル名の直後には必ずコロンを記述してください。
- ラベル名とニーモニックの間には、スペース又はタブを記述できます。

- アセンブラソース行にコメントを記述する場合は、ニーモニックとオペランドのつぎに、セミコロン(;)を記述し、それ以降の桁にコメントを記述してください。
- コメント行は、アセンブラリストファイルに出力されます。
- ニーモニックのオペランドとコメントの間にはスペース又はタブを記述できます。

3.3.5 ラベル定義行の記述規則

- ラベル名の直後には必ずコロン(:)を記述してください。
- ラベル名とコロン(:)の間には、何も記述しないでください。
- ラベル名は行の先頭から記述できます。
- 行の先頭にはスペース又はタブを記述できます。
- コメントは、アセンブラリストファイルに出力されます。
- ラベルとコメントの間にはスペース又はタブを記述できます。

記述例)
```
start:
label:    .BLKB    1
main:     nop
loop:
```

3.3.6 コメント行の記述規則

- 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。
- 指示命令行、アセンブラソース行およびラベル定義行以降にコメントを記述することができます。
- コメント行の先頭にはスペース又はタブを記述できます。
- コメントには、全ての文字を記述できます。

記述例)
```
 ; comment line
MOV.W     #0,A0
       ; comment
```

【注】 as30 は、セミコロン(;)以降の桁に記述したニーモニックや指示命令について、コードを生成しません。セミコロン(;)の記述位置には注意してください。as30 は、ダブルクォーテーション(")又はシングルクォーテーション(')で囲まれたセミコロン(;)は、コメントの先頭文字と判断しません。

3.3.7 空行の記述規則

- ソースプログラムの可読性を向上するなどの必要に応じて、文字を含まない行を記述できます。
- 空行には、スペース、タブ、リターン及びラインフィード文字以外は記述できません。

記述例)
```
loop :

  JMP   loop
```

3.4 行の連結

- '¥¥'を記述した場合、次の行を'¥¥'を記述した位置に連結します。
- '¥¥'を記述した行にコメントを記述できます。ただし、連結結果にはコメントは出力されません。
- '¥¥'を記述した行でエラーが発生した場合、連結される最終行に対して出力されます。
- 次に行連結の記述例と連結結果を示します。

【注】 連結された結果の行の最大文字数が「表 2.1 アセンブラの翻訳限界」に記載した文字数以下になるように記述してください。ただし、連結される行の先頭のスペース及びタブは文字数に含まれません。2 バイトコード文字の直後に'¥'を記述した場合、'¥¥'と認識される場合がありますのでご注意ください。

記述例1)
```
.BYTE    1,¥¥
         2, ¥¥
         3      ¥¥
         ,4
```

連結結果)
```
.BYTE    1,2, 3        ,4
```

記述例2)
```
.BYTE     1,¥¥ ;comment
2,;Comment ¥¥
3 ;COMMENT
```

連結結果)
```
.BYTE     1,2, ;Comment
3 ;COMMENT
```

記述例3)
```
.BYTE     1,¥¥
2,¥¥
3,  ¥¥
4
```

連結結果)
```
.BYTE     1,2,3,  4
```

3.5 オペランド

3.5.1 オペランドの種類

ニーモニック及び指示命令には、その命令の制御の対象を示すオペランドが記述できます。オペランドには、次に示す種類があります。

【注】 オペランドを持たない命令もあります。オペランドの有無については、各命令の記述規則を参照してください。

(1) 数値

整数と、浮動小数点数が記述できます。

(2) 名前

ラベル名及びシンボル名が記述できます。

(3) 式

数値及び名前を項に持つ式が記述できます。

(4) 文字列

文字又は文字列を ASCII コードとして扱えます。

3.5.2 オペランドの記述規則

オペランドの記述位置

オペランドと、オペランドをもつ命令との間に、必ずスペース又はタブを記述してください。

以降の章に各オペランドの記述規則を説明します。

3.5.3 数値

ソースファイルに記述できる数値として次の種類をサポートしています。

なお、整数値の内部表現は-2147483648～2147483647 です。

- 2 進数
- 8 進数
- 10 進数
- 16 進数
- 浮動小数点数

(1) 2 進数

0～1 のいずれかの数字で記述し、接尾辞として B または b を添付します。

記述例)
```
1011000B
1011000b
```

(2) 8 進数

0～7 までの数字で記述し、接尾辞として O または o を添付します。

記述例)
```
60702O
60702o
```

(3) 10 進数

0～9 までの数字で記述します。

記述例)
```
9243
```

(4) 16 進数

0～9、A～F、a～f で記述し、接尾辞として H または h を添付します。
アルファベットで始まる数値の場合は接頭辞として 0 を添付します。

記述例)
```
0A5FH
5FH
0a5fh
5fh
```

(5) 浮動小数点数

浮動小数点数は制御命令 ".FLOAT"と".DOUBLE" のオペランドにのみ記述できます。
浮動小数点数は式に記述できません。
浮動小数点数で表される次の範囲の値を記述できます。
FLOAT (32bits) 1.17549435×10^{-38} ～ 3.40282347×10^{38}
DOUBLE (64bits) $2.2250738585072014\times10^{-308}$ ～ $1.7976931348623157\times10^{308}$

記述方法) (仮数部)E(指数部)
(仮数部)e(指数部)

記述例)
```
3.4E35      ;3.4×10^35
3.4e-35     ;3.4×10^-35
-.5E20      ;-0.5×10^20
5e-20       ;5.0×10^-20
```

3.6 式

数値、名前及び演算子を組み合わせた式を記述できます。

- 演算子と数値の間には、スペース又はタブを記述できます。
- 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。
- 式の演算結果の値の範囲は、-2147483648～2147483647となります。
- 式の項に浮動小数点数は記述できません。

【注】 演算結果が、-2147483648～2147483647 を超えた場合、範囲外エラーにはなりません。

3.7 演算子

as30 のソースプログラムに記述できる演算子の一覧を示します。

- 単項演算子

表 3.2 単項演算子

演算子	機能
+	続く値を正の値として扱います。
-	続く値を負の値として扱います。
~	続く値の論理否定値を扱います。
SIZEOF	オペランドに指定したセクションのサイズ(バイト数)を値として扱います。
TOPOF	オペランドに指定したセクションの開始アドレスを値として扱います。

SIZEOF および TOPOF は、オペランドとの間に空白文字またはタブを記述します。
(例) SIZEOF program

- 二項演算子

表 3.3 二項演算子

演算子	機能
+	左辺値と右辺値を加算します。
-	左辺値から右辺値を減算します。
*	左辺値と右辺値を乗算します。
/	左辺値を右辺値で除算します。
%	左辺値を右辺値で割った余りを扱います。
>>	左辺値を右辺値回右へビットシフトします。
<<	左辺値を右辺値回左へビットシフトします。
&	左辺値と右辺値のビット毎の論理積値を扱います。
|	左辺値と右辺値のビット毎の論理和値を扱います。
^	左辺値と右辺値のビット毎の排他的論理和値を扱います。

- 条件演算子

  条件演算子は制御命令".IF", ".ELIF" のオペランドにだけ記述できます。

表 3.4 条件演算子

演算子	機能
>	左辺値が右辺値より大きいことを評価します。
<	左辺値が右辺値より小さいことを評価します。
>=	左辺値が右辺値以上であることを評価します。
<=	左辺値が右辺値以下であることを評価します。
==	左辺値が右辺値と等しいことを評価します。
!=	左辺値が右辺値と等しくないことを評価します。

- 演算優先順位変更演算子

表 3.5 演算優先順位変更演算子

演算子	機能
( )	( ) で囲った式を最優先で演算します。一つの式に複数の() が記述されている場合は、左側が優先されます。( ) はネストした記述ができます。

3.8 式の演算優先順位

オペランドに記述されている式は、次に示す優先順位に従い演算を行った結果の数値を値として扱います。

- 演算子がもつ優先順位の高いものから演算します。演算子の優先順位を以下表に示します。
- 同一の優先順位を持つ演算子は、左から順に演算します。
- ()で囲った式は、優先順位が一番高くなります。

表 3.6 式の演算優先順位

優先順位	演算子の種類	演算子
1	演算優先順位変更演算子	()
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	|, ^
8	条件演算子	>, <, >=, <=, ==, !=

3.9 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。文字列には、7ビット長 ASCII コードの文字が記述できます。オペランドに文字列を記述する際には、特に指定のある場合を除いて、シングル又はダブルクォーテーションで囲って記述してください。

記述例）
```
"string"
'string'
```

3.10 ニーモニック記述の概要

アセンブラニーモニックの記述規則について、詳しくは「M16C シリーズ、R8C ファミリの各ソフトウェアマニュアル」を参照してください。

(1) サイズ指定子

ニーモニックの処理対象となるデータのサイズ(B,W,L)を指定します。必ず指定してください。

(2) 分岐距離指定子

分岐及びサブルーチン呼び出し命令の分岐先への距離を指定します。通常は指定する必要はありません。
オペランドが間接アドレッシングの場合は分岐距離指定子を記述してください。省略されている場合はエラーとなります。

(3) 命令フォーマット指定子

オペコードの形式を指定します。命令フォーマット(G,Q,Z,S)が異なると、オペコード及びオペランドのコード長が異なります。通常は指定する必要はありません。

(4) アドレッシングモード指定子

オペランドデータのアドレッシングモードを指定します。as30 では、相対アドレッシングのアドレス範囲を指定する部分をアドレッシングモード指定子と呼びます。

記述例）
```
":16"及び":8"がアドレッシングモード指定子です
MOV.W    work1:16[SB],work2:8[SB]
```

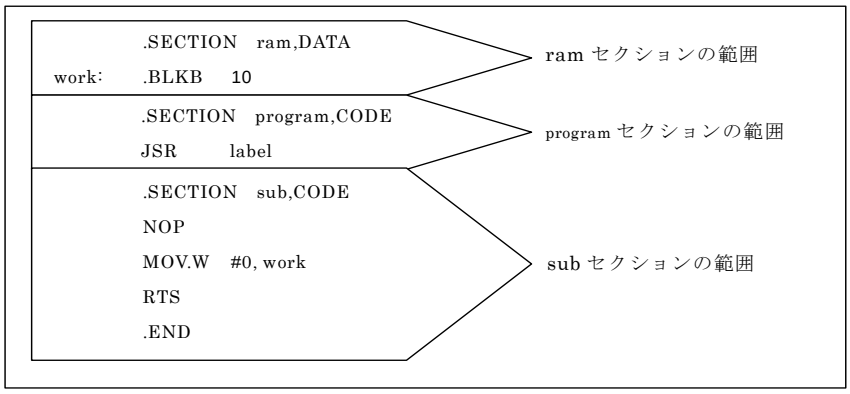
4. プログラミング

4.1 セクション

アセンブラが出力するオブジェクトファイルの実行命令、データの各領域は、セクションを構成します。
セクションの区切りは、次のように決められます。

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

なお、同一名称のセクションが複数存在する場合は、1 つのセクションに連結されます。



4.1.1 セクションの種類

as30 は、リロケータブル情報をセクション単位で出力します。セクションはセクション内に記述される命令とセクション宣言によって以下のように分類されます。分類は「絶対属性の DATA タイプセクション」のように属性とタイプの組み合わせによって決まります。

- 絶対属性セクション（属性）
- 相対属性セクション（属性）
- CODE タイプセクション（タイプ）
- DATA タイプセクション（タイプ）
- ROMDATA タイプセクション（タイプ）

1. 絶対属性セクション

- アセンブル時にコードの配置アドレスが決定するセクションです。
- 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。
- 絶対セクション内で指定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,CODE
      .ORG       1000H
```

2. 相対属性セクション

- アセンブル時にコードの配置アドレスが決定しないセクションです。
- 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。
- セクションを相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ないでください。

```
例）  .SECTION   program,CODE
```

3. CODEタイプセクション（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 タイプセクションは、ROM 領域に配置するセクションです。

```
例）  .SECTION   program,CODE
```

4. DATAタイプセクション（可変データ領域）

- 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 タイプセクションは、RAM 領域に配置するセクションです。

```
例）  .SECTION   mem,DATA
```

5. ROMDATAタイプセクション（固定データ領域）

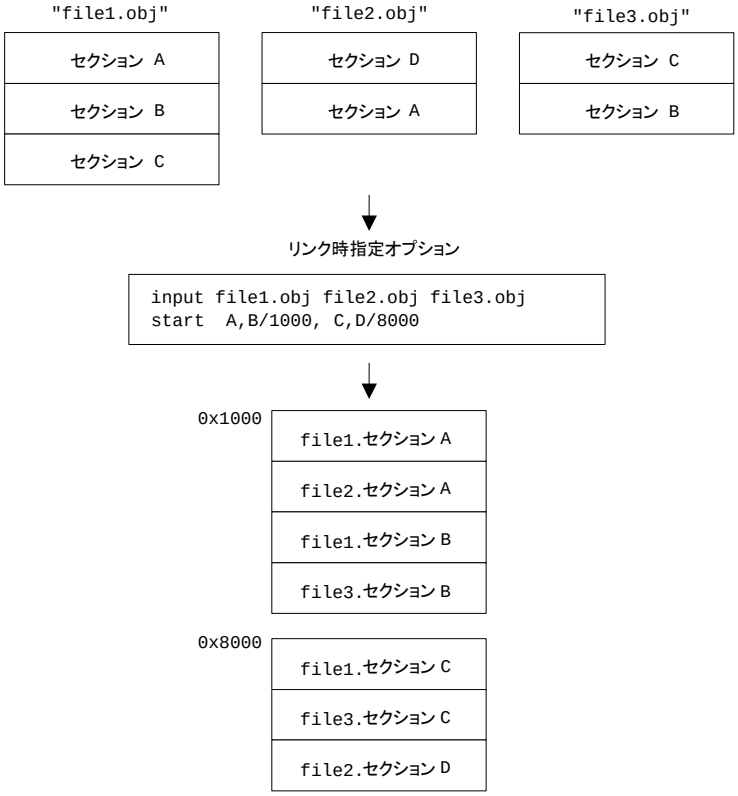
- プログラム以外の固定データを記述する領域です。
- データを設定する指示命令が記述できます。
- 領域を確保する指示命令を除く全ての命令が記述できます。
- ROMDATA タイプセクションは、ROM 領域に配置するセクションです。

```
例）        .SECTION        const,ROMDATA
```

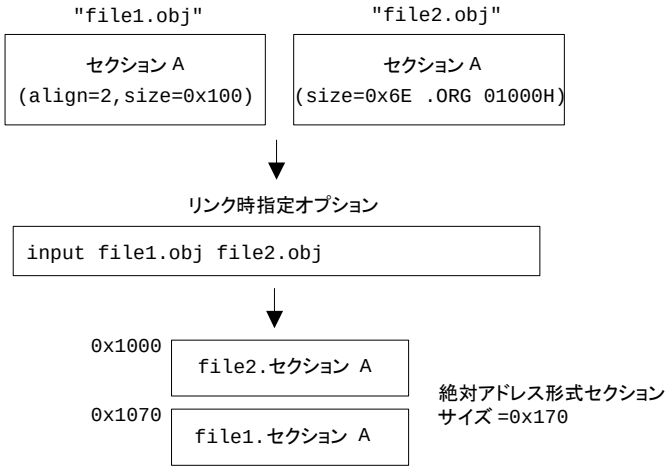
4.1.2 セクションの結合

最適化リンケージエディタでは、入力オブジェクトファイル内の同一セクションを結合し、start オプションによって指定されたアドレスに割り付けます。

1. 異なるファイルの同名セクションは、ファイルの入力順に連続して割り付けます。

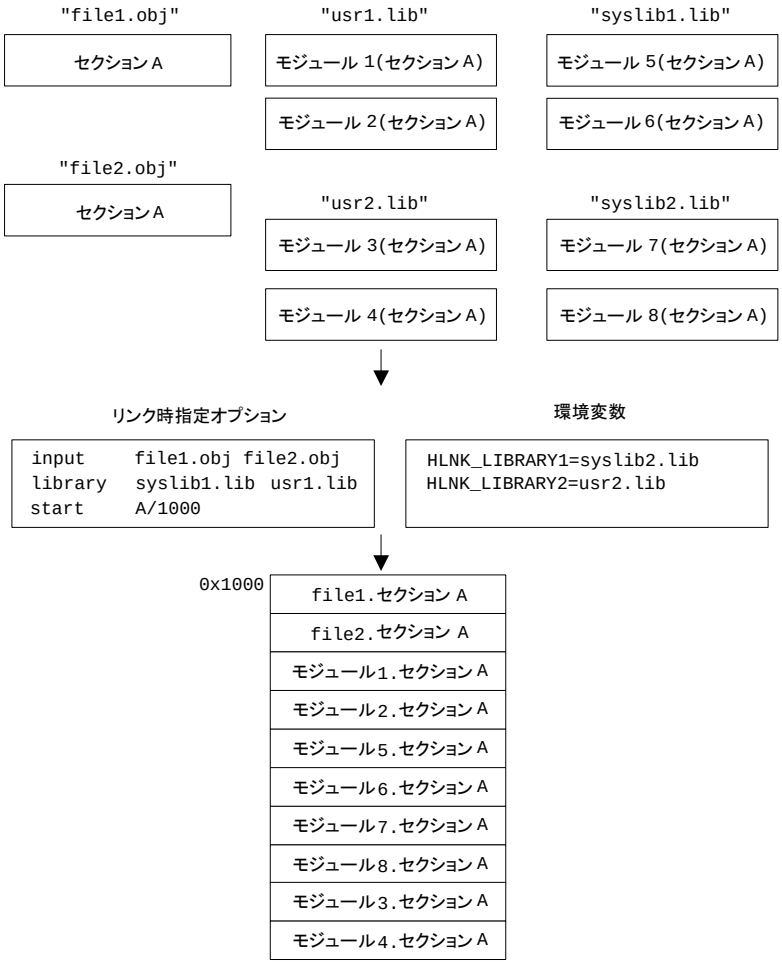


2. 同名セクションに絶対アドレス形式と相対アドレス形式が含まれている場合、絶対アドレス形式セクションの後に相対アドレス形式セクションを結合します。



3. 同名セクションの結合順序に関する規則は、優先度の高い順に以下の通りです。

- input オプションまたはコマンドライン上の入力ファイル指定順
- library オプションのユーザライブラリ指定順およびライブラリ内モジュール入力順
- library オプションのシステムライブラリ指定順およびライブラリ内モジュール入力順
- 環境変数(HLNK_LIBRARY1～3)のライブラリ指定順およびライブラリ内モジュール入力順



4.2 ラベルとシンボル

4.2.1 属性

ラベルとシンボル(ビットシンボルも含む)は、以下の 4 つの属性に分類されます。

- グローバル
- ローカル
- リロケータブル
- アブソリュート

属性はラベルとシンボルの定義および参照の組み合わせにより決定します。

	グローバル	ローカル	リロケータブル	アブソリュート
グローバル	---	×	○	○
ローカル	×	---	○	○
リロケータブル	○	○	---	×
アブソリュート	○	○	×	---

例)グローバル属性は、リロケータブル属性またはアブソリュート属性との組み合わせが可能です。

1. 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。
- グローバルな属性をもつ名前の情報はオブジェクトファイルに出力されます。

2. ローカル

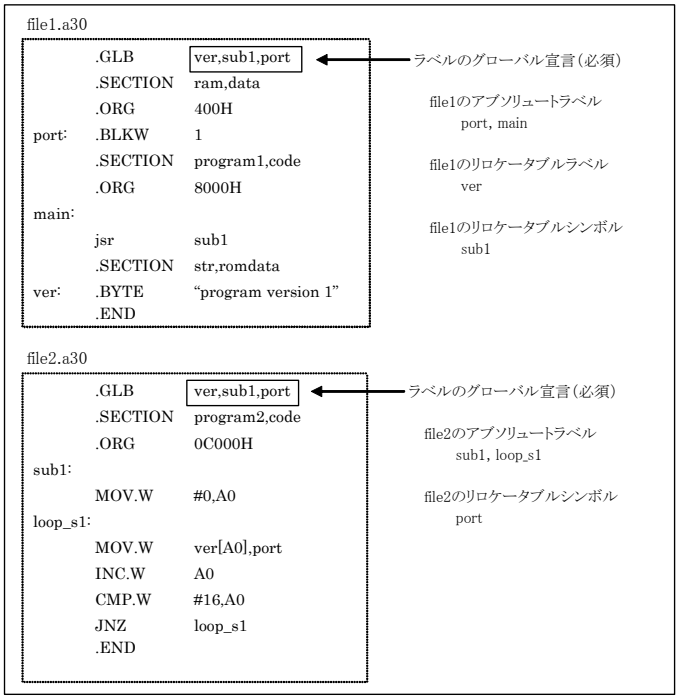
- 指示命令".GLB"または".BTGLB"で指定のない名前は、すべてローカルとなります。
- ローカルな名前は、定義した同一ファイル内でだけ参照できます。
- ローカルな名前は、別のファイルで同一のラベル名を使用できます。
- ローカルラベル及びローカルシンボルの情報は、相対属性セクション内で定義されているものだけがオブジェクトファイルに出力されます。ただし、コマンドオプション(-S/-SM)を指定してアセンブルした場合は、全てのローカルラベル及びローカルシンボルの情報がオブジェクトファイルに出力されます。

3. リロケータブル

- 相対属性セクション内に定義されたラベルの属性はリロケータブルです。
- 外部参照されたラベル、シンボル、ビットシンボルの属性はリロケータブルです。

4. アブソリュート

- 絶対属性セクション内で定義されているラベルの属性はアブソリュートです。
- 定数で定義されたシンボル、ビットシンボルの属性はアブソリュートです。



4.2.2 値の決定

アブソリュート属性およびリロケータブル属性の値は以下の通り決定されます。

1. アブソリュート属性

- アブソリュート属性をもつラベルとシンボルの値はアセンブル実行時に決定されます。

2. リロケータブル属性

- リロケータブル属性をもつラベルとシンボルの値はリンク時に決定されます。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を越えた場合はウォーニングが出力されます。

4.2.3 コマンドオプションによるシンボル定義

as30 は、プログラムの起動時にコマンドオプション(-D)によって、シンボルの定義ができます。シンボル定義の機能は、条件アセンブル機能などと組み合わせて使用できます。詳細は「条件アセンブル機能」の項を参照してください。

4.3 インクルードファイルの参照

as30 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。

1. インクルードファイルの記述規則

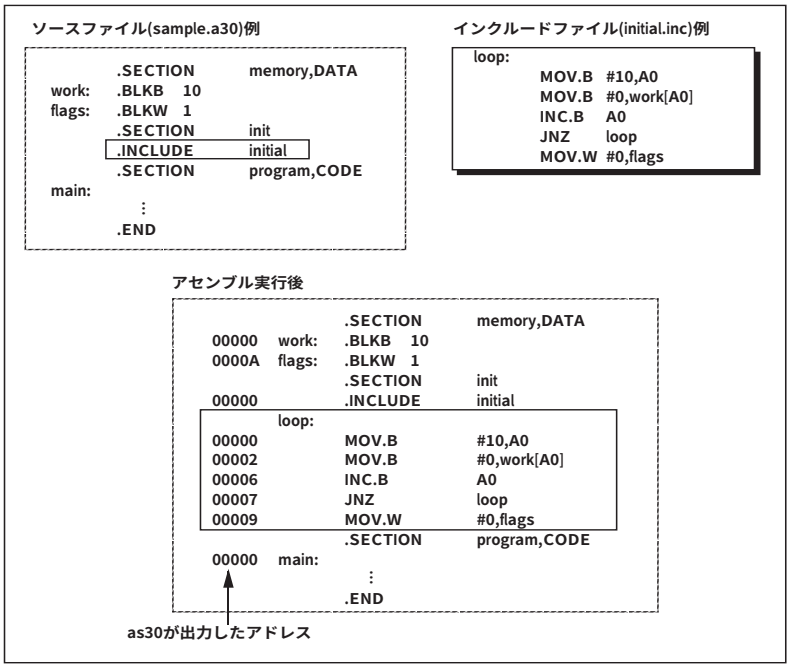
インクルードファイルの記述は、ソースプログラムの記述規則に従って記述してください。

注意事項
指示命令".END"は、インクルードファイル内に記述しないでください。

2. 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。

ソース記述例)



4.4 as30 のコード最適化選択

as30 は、M16C シリーズ、R8C ファミリのアドレッシングモードの中から、できるだけ最短のコードを選択します。
アセンブラは次の条件にあてはまる場合にコードの最適化を行います。

- 分岐距離指定子が省略されている場合（通常は省略します）
- 命令フォーマット指定子が省略されている場合（通常は省略します）
- アドレッシングモード指定子が省略されている場合
- 上記の組み合わせ

1. 分岐距離指定子が省略されている場合

as30 は、次の条件を全て満たす場合に最適選択を行います。

- オペランドが1つのラベルで記述されている場合、または1つのラベルを含む式で記述されている場合。
  ラベル
  ラベル＋アセンブル時確定値
  ラベルーアセンブル時確定値
  アセンブル時確定値＋ラベル
- オペランドのラベルが同一ファイルの同一セクション内で定義されている場合。
- 命令の記述されているセクションと、オペランドのラベルを定義しているセクションが共に絶対属性で、かつ同一ファイル内に記述されている場合。

最適化規則

- 無条件分岐命令
  分岐距離、'.A','.W','.B','.S'の中から、分岐可能な最短の命令を選択します。
  分岐命令と分岐先のラベルが同一セクション内にある場合だけ'.S'サイズを選択します。

- サブルーチン呼び出し命令
  分岐距離、'.A','.W'の中から、分岐可能な最短の命令を選択します。

- 条件分岐命令
  分岐距離、'.B'または代替え命令を生成します。

注意事項
リストファイルのソース行情報は、記述したソース行をそのままを出力します。コード情報部には、代替え命令のコードを出力します。
"ADJNZ"および"SBJNZ"命令に対して条件分岐命令と同等の分岐最適化を行います。

2. 命令フォーマット指定子が省略されている場合

- as30 は、命令フォーマット指定子が省略されたニーモニックについて最適化を行います。
- as30 は、命令フォーマット指定子が省略されている場合、アドレッシングモードを決定してから命令フォーマットを選択します。

3. アドレッシングモード指定子が省略されている場合（SB相対またはFB相対の選択）

アドレッシングモード指定子が省略されている場合、次の条件を満たす場合にコードの最適選択が行われます。

- ディスプレースメント付きアドレッシングで、ディスプレースメントの値がアセンブル実行時に決定する場合、アドレッシングモードの最適選択を行います。
- 指示命令".SB"または".FB"が定義されている場合は、条件によって 8 ビット SB 相対アドレッシングモード（以降 SB 相対と示します）または 8 ビット FB 相対アドレッシングモード（以降 FB 相対と示します）を選択します。

以下にそれぞれのアドレッシングモードを選択する場合毎に、条件を示します。

(a) SB 相対アドレッシングモードが選択される条件

次のいずれかの条件を満たす場合に SB 相対を選択します。
- アセンブル実行時にオペランドの値が確定し、その値が SB 相対を選択可能な範囲である場合。
  SB 相対を選択可能な範囲は、アドレス 0～0FFFFH の範囲内、かつ 16 ビットレジスタ(SB)をベースに＋0～＋255 の範囲です。
- 指示命令".SBSYM"で宣言されたシンボルがオペコードに記述されている場合。

SB 相対選択例)

```
.SB     100H
symbol  .EQU    108H
ABS.B   symbol
```

注意事項

SB 相対アドレッシングを使用する場合は、必ず指示命令".SB"で SB レジスタ値を設定してください。
SB レジスタ値がアセンブル時に確定しない式で定義されている場合は、最適化を行いません。

ビット命令アドレッシング命令の SB 相対アドレッシングモード選択
- ニーモニックが命令フォーマットにショート形式を持つ場合は、ショート形式の SB 相対を選択します。
- ニーモニックが命令フォーマットにショート形式を持たない場合は、16 ビット SB 相対アドレッシングモードを選択します。

(b) FB 相対アドレッシングモードが選択される条件

次のいずれかの条件を満たす場合に FB 相対を選択します。
- 指示命令".FBSYM"で宣言されたシンボルがオペコードに記述されている場合。
- 指示命令".FBSYM"で宣言されたシンボルを含む次の式がオペコードに記述されている場合。
  (シンボル)＋アセンブル時確定値
  (シンボル)－アセンブル時確定値
  アセンブル時確定値＋(シンボル)

注意事項

FB 相対アドレッシングを使用する場合、必ず指示命令".FB"で FB レジスタ値を設定してください。

4. アドレッシングモード指定子が省略されている場合(アドレスレジスタ間接の選択)

アドレッシングモード指定子が省略されている場合、次の条件を満たす場合にコードの最適選択が行われます。
- ディスプレースメント付きアドレッシングで、ディスプレースメントの値がアセンブル実行時に決定し、その値が 0 の場合、アドレッシングの最適選択を行います。

アドレスレジスタ間接選択例)

```
symbol  .EQU    0H
ABS.B   symbol[A0]    ; アドレスレジスタ間接を選択します
ABS.B   symbol:8[A0]  ; アドレスレジスタ相対を選択します。
```

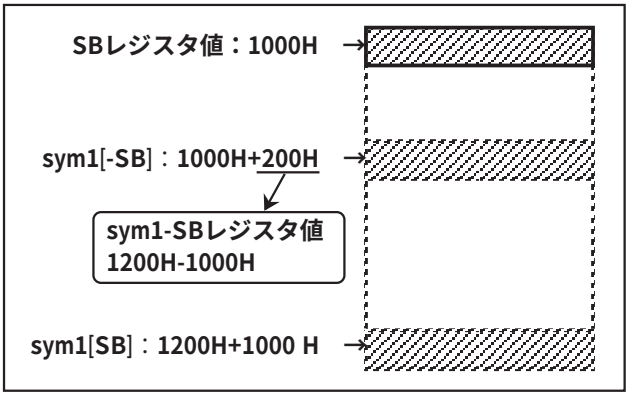
4.5 SBレジスタオフセットアドレス指定

as30 のプログラミングでは、SB レジスタ値からのオフセットアドレスを記述できます。

- 指示命令".SB"で指定されたアドレス値に指定したオフセット値を加えた値を処理対象とします。
- SB 相対アドレッシングモードでコード生成されます。
- SB 相対アドレッシングモードを記述できるオペランドに指定します。
- オフセットは、ラベル、シンボルまたは数値が記述できます。

記述例例）

```
sym1        .EQU    1200H
.SECTION    P
.SB         1000H
MOV.B       #0,sym1[SB]
MOV.B       #0,sym1[-SB]
.END
```



4.6 スペシャルページベクタテーブル

スペシャルページ分岐

M16C ファミリのアセンブリ言語では、"JMPS"ニーモニックを記述するとスペシャルページベクタテーブルを使用したスペシャルページ分岐ができます。

スペシャルページサブルーチン

M16C ファミリのアセンブリ言語では、"JSRS"ニーモニックを記述するとスペシャルページベクタテーブルを使用したスペシャルページサブルーチン呼び出しができます。

スペシャルページベクタテーブル

スペシャルページベクタテーブルの概要を示します。

- スペシャルページベクタテーブルは、アドレス 0FFE00H から 0FFFDBH 番地に割り当てられています。
- 1 ベクタテーブルは、2 バイトで構成されます。
- 1 ベクタテーブル毎にスペシャルページ番号が割り当てられています。
- スペシャルページ番号は、アドレス 0FFE00H 番地から 255、254 と 2 バイト毎に小さくなります。

スペシャルページベクタテーブルの詳細については、「ソフトウエアマニュアル」を参照してください。

本マニュアルでは、スペシャルページベクタテーブルの設定方法と参照方法を示します。

なお、スペシャルページベクタテーブルを自動で生成する場合は、アセンブラ指示命令".SVECTOR"を参照してください。

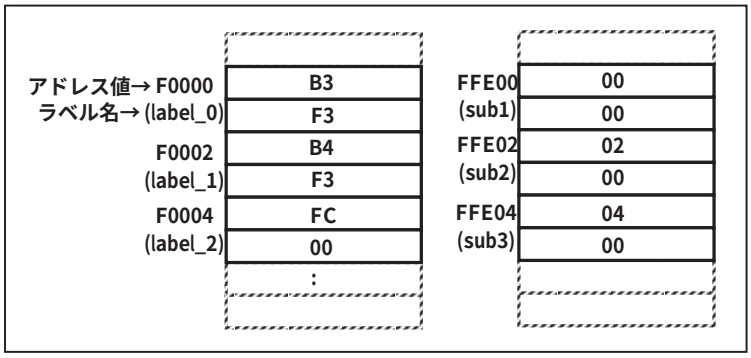
4.6.1 スペシャルページベクタテーブルの設定方法

スペシャルページベクタテーブルには、スペシャルページ内アドレスの下位 16 ビットを格納します。

- スペシャルページベクタテーブル専用のセクションを定義してください。
- 指示命令".ORG"で絶対アドレスを定義してください。
- アドレスは必ず 0FFE00H から 0FFFDBH 番地の範囲内で偶数番地を設定してください
- スペシャルページ内アドレスの下位 16 ビットを指示命令".WORD"で ROM に格納します。

記述例)

```
                .SECTION sp_vect,ROMDATA
                .ORG     0FFE00H
    sub1:       .WORD    label_0 & 0FFFFH       ;Special Page No.255
    sub2:       .WORD    label_1 & 0FFFFH       ;Special Page No.254
    sub3:       .WORD    label_2 & 0FFFFH       ;Special Page No.253
                ;
                .ORG     0FFFDAH
    sub238:     .WORD    label_238 & 0FFFFH     ;Special Page No.18
```



4.6.2 スペシャルページベクタテーブルの参照方法

スペシャルページベクタテーブルを参照するには、次の 2 つの方法があります。

- スペシャルページベクタテーブルのアドレスを指定する。
- スペシャルページ番号を指定する。

記述規則

- スペシャルページベクタテーブルのアドレスを指定する場合は、必ず"¥"を先頭に記述してください。
- スペシャルページ番号を指定する場合は、必ず"#"を先頭に記述してください。

記述例 1: スペシャルページベクタテーブルのアドレス指定

```
        .SECTION  p
main:
        JSRS      ¥sub1
        JSRS      ¥sub2
        JMPS      ¥sub3
        .SECTION  special
        .ORG      0F0000H
Label_0:
        MOV.B     #0,R0H
        RTS
label_1:
        MOV.B     #0,R0L
        RTS
label_2:
        JMP       main
        .END
```

記述例 2: スペシャルページ番号を指定

```
        .SECTION  p
main:
        JSRS      #255
        JSRS      #254
        JMPS      #253
        ;
```

4.7 マクロ機能

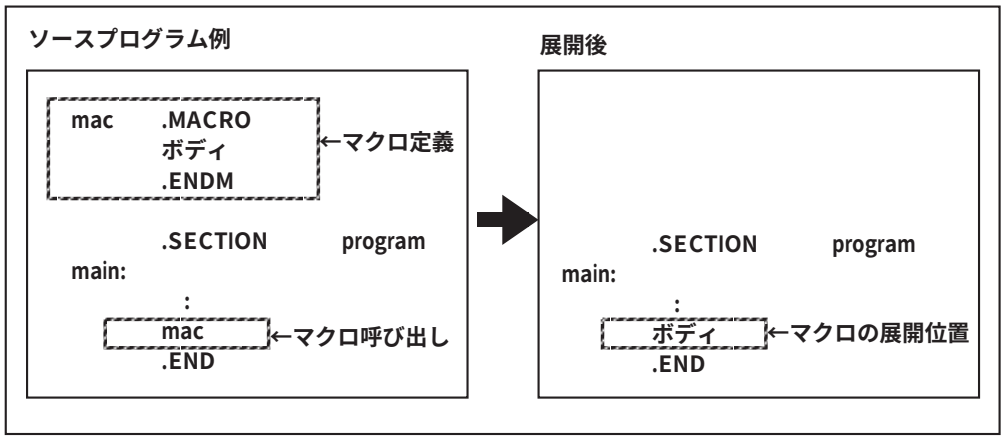
as30 で使用できるマクロ機能について説明します。as30 がサポートしているマクロ機能は 2 種類あります。

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

次に、各マクロ機能について説明します。

4.7.1 マクロ機能

- マクロ機能は、マクロ定義したマクロ名を、マクロ呼び出しすることで使用できます。
- マクロ定義だけでは、マクロ機能を使用することになりません。
- マクロ定義とマクロ呼び出しは、次のような関係になります。



マクロ定義

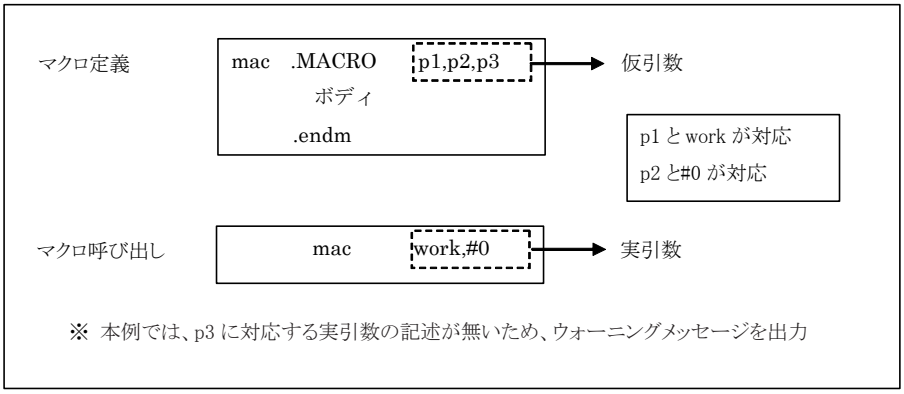
- マクロ定義は、指示命令".MACRO"を使用して、1行以上の命令の集まりを1つのマクロ名に定義します。
- マクロ名及びマクロ引数は、大文字と小文字を区別して扱います。
- マクロ定義の終了は、指示命令".ENDM"で示します。
- 指示命令".MACRO"と".ENDM"に囲まれた行を、マクロボディと呼びます。
- マクロ定義には、仮引数を定義できます。
- 再帰的なマクロ定義ができます。
- マクロのネストはマクロ定義及びマクロ呼び出しを含めて 65535 レベル以下です。
- 同一名のマクロの再定義ができます。
- マクロ定義は、セクションの範囲外に記述できます。
- マクロボディには、ソースプログラムに記述可能な全ての命令を記述できます。
- マクロ仮引数(80個以下)が記述できます。
- 1つのアセンブラソースファイル内の合計が 65535 個以下のマクロローカルラベルを記述できます。

マクロローカルラベル

- 指示命令".LOCAL"で宣言したラベルはマクロローカルラベルとなります。
- マクロローカルラベルは、マクロ定義内でのみ使用できます。
- マクロローカルラベル宣言をしたラベル名は、マクロの範囲外で同一名のラベルを記述できます。
- マクロローカルラベルとして使用するラベルは、そのラベルを定義する以前に、マクロローカルラベルであることを宣言してください。

マクロ呼び出し

- 指示命令".MACRO"で定義したマクロ名を記述することで、マクロ呼び出しが行えます。
- マクロ呼び出しによって、マクロボディのコードが生成されます。
- マクロ名の前方参照(マクロ呼び出し行よりも後の行で定義されているマクロ名を記述すること)はできません。必ずマクロ定義は、呼び出し行よりも前の行に記述してください。
- マクロ名の外部参照(別のファイルで定義されているマクロ名を記述すること)はできません。複数の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にマクロを定義し、そのファイルをインクルードしてください。
- マクロ定義されている仮引数に対応する実引数を記述できます。



4.7.2 繰り返しマクロ機能

- 指示命令".MREPEAT"と".ENDR"で囲まれたボディを指定した回数分繰り返し、指定した行以降に展開します。
- 繰り返しマクロは、定義した行に、展開されます。
- 繰り返しマクロの定義行にはラベルを記述できます。

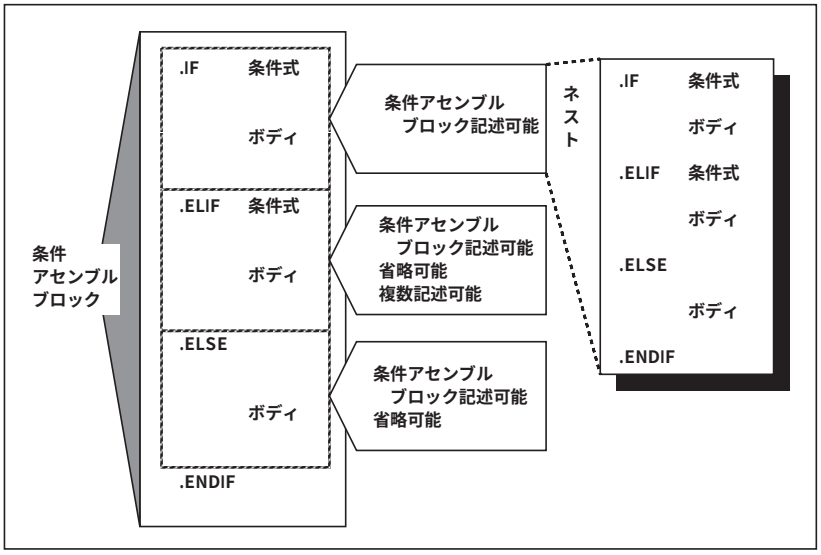
注意事項

このラベルは、マクロ名ではありません。繰り返しマクロのマクロ呼び出しはありません。

4.8 条件アセンブル機能

条件アセンブルとは、与えられた条件に従ってソース行を機械語に変換したりしなかったりの制御を行うことを言います。条件を判断した結果、アセンブルされなかった行のコードは生成されません。

条件アセンブルブロックの構成



条件アセンブルの実行例

3種類のメッセージを選択してアセンブルする場合の例を次に示します。アセンブラソースファイル名は、"sample.a30"とします。

ソース記述例)

```
        .SECTION    outdata,ROMDATA,ALIGN
.IF    TYPE==1
        .BYTE   "PROTO TYPE"
.ELIF  TYPE>1
        .BYTE   "MASS PRODUCTION  TYPE"
.ELSE
        .BYTE   "DEBUG MODE"
.ENDIF
        .END
```

コマンド入力例１）

```
>as30 sample.a30 -DTYPE=1
```

アセンブル結果１）

```
.SECTION   outdata,ROMDATA,ALIGN
.BYTE      "PROTO TYPE"
.END
```

コマンド入力例２）
```
>as30 sample.a30 -Dtype=2
```

アセンブル結果２）
```
.SECTION   outdata,ROMDATA,ALIGN
.BYTE      "MASS PRODUCTION TYPE"
.END
```

コマンド入力例３）
```
>as30 sample.a30 -Dtype=0
```

アセンブル結果３）
```
.SECTION   outdata,ROMDATA,ALIGN
.BYTE      " DEBUG MODE"
.END
```

ソースファイル内に条件を記述する場合

アセンブラソースファイル内で"type"に値を設定する方法を示します。

```
type    .EQU      1
         .SECTION    outdata,ROMDATA,ALIGN
.IF     type==1
         .BYTE    "PROTO TYPE"
.IF     type>1
         .BYTE    "MASS PRODUCTION TYPE"
.ELSE
         .BYTE    "DEBUG MODE"
.ENDIF
         .END
```

5. アセンブラ操作方法

5.1 コマンド入力時の注意事項

コマンドプロンプト上で操作を行ってください。
ファイル名に使用できるピリオドは一つ以下です。

5.2 コマンド行の構成

コマンド行では、次の情報を入力してください。

プログラム名

使用するプログラムの名前です。

コマンドパラメータ

コマンドパラメータには、起動するプログラムの処理対象となるファイル名や、プログラムの機能を記号で示したコマンドオプションを含みます。

コマンドパラメータは次の情報を含みます。

- ファイル名
  プログラムの処理対象となるファイルの名前です。
- コマンドオプション
  プログラム毎の基本的な機能を利用するために、プログラム起動時に付加します。

5.3 コマンド行の入力規則

as30 の各プログラムを起動する場合の、コマンド行への入力は次の規則に従って行ってください。

- コマンド行文字数
  コマンド行に指定できる文字数は、2048文字（バイト）以下です。
  as30の使用環境（OSの種類）によっては、上記以下の文字数に制限される場合もあります。
- 起動プログラム名とファイル名の間は、必ずスペースを記述してください。
  ファイル名と、各コマンドオプションの間には、必ずスペースを記述してください。
- ファイル名の記述規則は、上記の他に OS によって規定されます。詳しくは、それぞれの OS の説明書を参照してください。
- ファイル名に含めることのできるピリオド(.)は、一箇所のみです。
- ファイルの拡張子（ピリオド以降の文字）の規則は、それぞれのプログラムの起動方法を確認してください。

5.4 コマンドパラメータの指定規則

コマンドパラメータの指定順序

コマンドオプションとアセンブラソースファイル名は任意の順序で指定できます。

アセンブラソースファイル名(必須)

- 必ず、1つ以上のアセンブラソースファイル名を指定してください。
- アセンブラソースファイル名には、パスの指定ができます。
- アセンブラソースファイル名は、80 個まで指定できます。
- 拡張子が".a30"であるファイルは、拡張子を省略できます。

注意事項

複数指定されたアセンブラソースファイルのうち、エラーを含むアセンブラソースファイルの処理以降のファイルは処理しません。

コマンドオプション

- コマンドオプションは、省略できます。
- コマンドオプションは、複数指定できます。
- コマンドオプションには、文字列や、数値が指定できるものがあります。

注意事項

コマンドオプションと文字列又は数値の間には、スペース又はタブを記述しないでください。

コマンドオプションを無効にできるのは、as30 のコマンドオプションだけです。その他のプログラムを起動する場合には使用できません。

コマンド入力例)

```
>as30 sample -L            …(1)
>as30 sample -S            …(2)
>as30 sample -L -S -L      …(3)
```

(1) コマンドオプション"-L"を指定
(2) コマンドオプション"-S"を指定
(3) コマンドオプション"-S -L"を指定

5.5 アセンブラコマンドオプション

5.5.1 ソースオプション

表 5.1 ソースオプション

No.	オプション	内容	ダイアログメニュー
1	-I	インクルードファイルの検索ディレクトリを指定する	アセンブラ〈ソース〉 [オプション項目]: インクルード検索ディレクトリ
2	-D	シンボル定義を行う	アセンブラ〈ソース〉 [オプション項目]: 識別子定義

インクルードファイル検索ディレクトリ指定

-I

書 式　　-I＜パス名＞

説 明　　指定されたディレクトリからアセンブラ指示命令".INCLUDE"で指定されたインクルードファイルを検索します。
本オプションは複数指定することができません。複数指定した場合は、最後に指定したオプションの内容が有効になります。
インクルードファイルの検索順位については、指示命令".INCLUDE"を参照してください。

記述例

```
 >as30 -Iwork¥include sample.a30
```

work¥include ディレクトリからインクルードファイルを検索します。

備 考　　"-I"に続けてディレクトリパス名を指定してください。
オプションとディレクトリパス名の間にはスペース又はタブは記述できません。

シンボル定数設定

-D

書 式　　-D＜シンボル名＞＝＜数値＞[:＜シンボル名＞＝＜数値＞. . .]

説 明　　シンボルに値を設定します。
値は絶対値として扱います。
本オプションで定義したシンボルは、アセンブラソースファイル内に記述したシンボル定義と同様に扱われます。したがって、アセンブラソースファイル内に同一名のシンボル定義が記述されている場合は、その記述位置で、シンボルを再定義したことになります。
本オプションで定義したシンボルは、ソースプログラム中の先頭箇所にシンボル定義を行った場合と同様の処理になります。ただし、アセンブラリストファイルには出力されません。

記述例

```
>as30 -Dname=1 sample.a30                …(1)
>as30 -Dname=1:symbol=1 sample.a30       …(2)
>as30 -Dname=1 sample1.a30 sample2.a30   …(3)
```

(1)　name というシンボルに1を設定します。
(2)　name 及び symbol というシンボルに1を設定します。
(3)　sample1 と sample2 のファイルに対して、name というシンボルを定義します。

備 考　　コマンドオプションとシンボル名の間には、スペース又はタブを記述しないでください。
複数のシンボルに値を定義できます。複数のシンボル定義を行う場合は、
-D(シンボル名)＝(数値):(シンボル名)＝(数値)のように、コロンで区切って続けて記述してください。
コロンの前後に、スペース又はタブは記述できません。

5.5.2 オブジェクトオプション

表 5.2 オブジェクトオプション

No.	オプション	内容	ダイアログメニュー
1	-S[M]	ローカルシンボル情報を出力する	アセンブラ〈オブジェクト〉
2	-finfo	インスペクタ情報を生成する	アセンブラ〈オブジェクト〉
3	-goptimize	モジュール間最適化用付加情報出力	アセンブラ〈オブジェクト〉
4	-N	デバッグ情報を出力しない	アセンブラ〈オブジェクト〉
5	-P	構造化記述命令を変換する	アセンブラ〈オブジェクト〉
6	-M	構造化記述命令をバイト型でニーモニックに変換する	アセンブラ〈オブジェクト〉
7	-O	出力ファイルのディレクトリを指定する	アセンブラ〈オブジェクト〉

ローカルシンボル情報の出力指定

-S

書 式　　-S [M]

説 明　　ローカルシンボル情報をオブジェクトファイルに出力します。
本オプションに"M"を付加することで、システムラベル情報もオブジェクトファイルに出力します。
ローカルシンボルを使用してシンボリックデバッグを行う場合には、本オプションを指定してアセンブルを行ってください。

記述例

```
>as30 -S sample.a30    …(1)
>as30 -SM sample.a30   …(2)
```

(1) sample.a30 のローカルシンボル情報を sample.obj に出力します。
(2) sample.a30 のシステムラベル情報及びローカルシンボル情報を sample.obj に出力します。

注意事項　　シンボル及びラベルの情報は、リンケージリストのシンボル情報で確認できます。
指示命令".INSF"のオペランドに記述されたサブルーチン開始ラベル名および指示命令".CALL"のオペランドに記述されたサブルーチン名は、本オプションに関係なくオブジェクトファイルに出力します。

インスペクタ情報を生成

-finfo

書 式　　-finfo

説 明　　NC30 の"-finfo"オプションで生成された各情報、またはアセンブラ指示命令で記述されたインスペクタ情報をオブジェクトファイルに出力します

記述例

```
>as30 -L -S -finfo sample.a30
```

備 考　　文字の大文字と小文字を区別しますので、すべて小文字で指定してください。

注意事項　　統合化環境をご使用の場合、本オプションはデフォルトで選択されます。

モジュール間最適化

-goptimize

書 式　-goptimize

説 明　モジュール間最適化用付加情報を出力します。
本オプションを指定したファイルは、リンク時にモジュール間最適化の対象になります。

記述例

```
>as30 -goptimize sample.a30
```

注意事項　本オプション指定時、無条件分岐命令およびサブルーチン呼び出し命令に分岐距離指定子が記述されていても optlnk でモジュール間最適化の対象になります。
本オプション指定時、無条件分岐命令、サブルーチン呼び出し命令、条件分岐命令、加算&条件分岐命令および減算&条件分岐命令の分岐距離は、表 5.3 ファイルフォーマット指定子の内容に従って決定します。

表 5.3 ファイルフォーマット指定子

命令	分岐距離指定子	分岐距離
JMP	.S	PC*1+2 ≦ オペランド≦ PC*1+8
	.B	PC*1−126 ≦ オペランド≦ PC*1+127
	.W	PC*1−32766 ≦ オペランド≦ PC*1+32767
JSR	.W	PC*1−32766 ≦ オペランド≦ PC*1+32767
GEU/C, GTU, EQ/Z, N, LTU/NC, LEU, NE/NZ, PZ	---	PC*1−126 ≦ オペランド≦ PC*1+127
LE, O, GE, GT, NO, LT	---	PC*1−125 ≦ オペランド≦ PC*1+128
ADJNZ	---	PC*1−125 ≦ オペランド≦ PC*1+128
SBJNZ	---	PC*1−125 ≦ オペランド≦ PC*1+128

*1 PC は命令の先頭番地を示します。

デバッグ情報の出力抑止

-N

書 式　-N

説 明　デバッグ情報をオブジェクトファイルに出力しません。
オブジェクトファイルのサイズを縮小できます。

記述例

```
>as30 -N sample.a30
```

注意事項　本オプションを指定して生成したオブジェクトファイルから、作成したアブソリュートファイルでは、ソース行レベルでのデバッグはできません。

構造化記述命令を処理

-P

書 式　　-P

説 明　　アセンブラソースファイルに記述されている構造化記述命令を処理します。

記述例

```
 >as30 -P -LS sample.a30
```

アセンブラソースファイル内の構造化記述命令を処理し、展開部分をアセンブラリストファイルに出力します。

備 考　　構造化記述命令を使用している場合は、必ず本オプションを指定してください。

構造化記述命令の変数をバイト型で生成

-M

書 式　　-M

説 明　　構造化記述命令において、型の決定していない変数をバイト型として処理します。

記述例

```
 >as30 -P -M sample.a30
 >as30 -M -P sample.a30
```

備 考　　本オプションはコマンドオプション"-P"とともに指定してください。
本オプションが指定されていない場合、型の決定していない変数はワード型で処理します。

出力ファイルのディレクトリを指定する

-O

書 式　　-O＜ディレクトリ名＞

説 明　　アセンブラが生成するオブジェクトファイル、アセンブラリストファイル及びアセンブラエラータグファイルの出力先ディレクトリを指定します。
ディレクトリ名には、ドライブ名を含めて指定できます。また、相対パスによる指定も可能です。

記述例

```
 >as30 -Oc:¥work¥asmout sample.a30        …(1)
 >as30 sample.a30 -O..¥tmp                …(2)
 >as30 -Oc:¥work¥asmout sample.a30 -L -T  …(3)
```

(1) オブジェクトファイルを、Cドライブの"¥work¥asmout"ディレクトリに出力します。
(2) オブジェクトファイルを、カレントディレクトリの親ディレクトリに属すtmp ディレクトリに出力します。
(3) オブジェクトファイル、アセンブラエラータグファイル及びアセンブラリストファイルを、Cドライブの"¥work¥asmout"ディレクトリに出力します。

備 考　　本オプションとディレクトリ名の間に、スペース又はタブは記述できません。

5.5.3 リストオプション

表 5.4 リストオプション

No.	オプション	内容	ダイアログメニュー
1	-L	アセンブラリストファイルを生成する	アセンブラ〈リスト〉
2	-H	アセンブラリストファイルのヘッダ情報を出力しない	アセンブラ〈リスト〉

アセンブラリストファイル生成

-L

書 式　-L[C|D|I|M|S]

説 明　オブジェクトファイルの他にアセンブラリストファイルを生成します。
生成したリストファイルの拡張子は、".lst"になります。
コマンドオプション"-O"でディレクトリを指定している場合は、指定したディレクトリにアセンブラリストファイルを生成します。

記述例

```
>as30 -LM sample.a30
```

備 考　本オプションには、'C'、'D'、'I'、'M'、'S'のファイルフォーマット指定子を指定できます。
ファイルフォーマット指定子と-Lの間に、スペース又はタブは記述できません。
ファイルフォーマット指定子は、同時に複数の指定が可能です。
ファイルフォーマット指定子の指定順序は任意です。

表 5.5 ファイルフォーマット指定子

フォーマット指定子	機能
C	行連結をそのままリストファイルに出力する。
D	.DEFINE を置き換える以前の情報をリストファイルに出力する。
I	条件アセンブルで条件が偽となった行をアセンブラリストファイルに出力します。
M	マクロ記述の展開行をアセンブラリストファイルに出力します。
S	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。

アセンブラリストファイルのヘッダ出力停止

-H

書 式　-H

説 明　アセンブラリストファイルのヘッダ情報の出力を抑止します。

記述例

```
>as30 -L -H sample.a30
```

備 考　コマンドオプション"-L"と同時に指定してください。

5.5.4 チューニングオプション

表 5.6 チューニングオプション

No.	オプション	内容	ダイアログメニュー
1	-A	ニーモニックオペランドを評価する	アセンブラ〈チューニング〉
2	-PATCH [6N]_TA -PATCH [6N]_TAn	三相モータ制御用タイマ機能の注意事項を回避するコードを生成する	アセンブラ〈チューニング〉

ニーモニックオペランドの評価

-A

書 式　　-A

説 明　　即値とアドレス値の両方が記述できるニーモニックに対して、オペランドが即値であることを示す"#"が記述されていない場合にウォーニングを出力します。

記述例　　ソース記述例)

```
 .section    prg,code
MOV.W       0,400H
 .end
```

-A 指定時のリストファイル出力例)

```
1                           .section  prg,code
2  00000  73FF00000004     MOV.W     0,400H
sample.a30(2) : A1207 (W) Addressing is described by the numerical value
3                           .end
```

注意事項　　オペランドが、ラベルを除く数値及びアセンブル時に値が確定したシンボルの場合にウォーニングを出力します。

三相モータ制御用タイマ機能の注意事項回避

-PATCH[6N]_TA／-PATCH[6N]_TAn

書 式　-PATCH_TA、-PATCH_TAn、-PATCH6N_TA、-PATCH6N_TAn

説 明　三相モータ制御用タイマ機能の注意事項を回避するコードを生成します。
タイマ A1-1 レジスタ(TA11)、タイマ A2-1 レジスタ(TA21)およびタイマ A4-1 レジスタ(TA41)で示される番地に対して MOV 命令(ワード長)で値を書きこむ場合に限り、回避コードを生成します。(上記番地はアセンブル時、確定する値のみ対象となります)

記述例1　ソース記述例)

```
 .section  prg,code
MOV.W  #7E, TA11
 .end
```

-PATCH_TA 指定時のリストファイル出力例)

```
1                           .section  prg,code
2  00000  75CF42037E00      MOV.W  #7E, TA11
          75CF42037E00      ; This is a line which AS30 output.
3                           .end
```

→ 記述された同一 MOV 命令が回避コードとして生成されます。

記述例2　ソース記述例)

```
 .section  prg,code
MOV.W  #7E,TA11
 .end
```

-PATCH_TA2 指定時のリストファイル出力例)

```
1                           .section  prg,code
2  00000  75CF42037E00      MOV.W  #7E,TA11
          0404
          75CF42037E00      ; This is a line which AS30 output.
3                           .end
```

→ "n"で指定された個数の NOP 命令および記述された同一 MOV 命令が回避コードとして生成されます。

備 考　"-PATCH_TAn"の"n"には 0 から 99 までの 10 進数値が指定できます。
本オプションは必ず大文字で指定してください。

表 5.7 回避コード生成の対象番地

オプション指定	回避コード生成の対象番地
-PATCH_TA -PATCH_TAn	TA11 が 342H 番地、TA21 が 344H 番地、 TA41 が 346H 番地
-PATCH6N_TA -PATCH6N_TAn	TA11 が 1C2H 番地、TA21 が 1C4H 番地、 TA41 が 1C6H 番地

注意事項　"-R8C"オプションと同時に指定することはできません。
本注意事項の詳細につきましては“MAEC TECHNICAL NEWS No.M16C-95-0302”を参照してください。

5.5.5 その他オプション

表 5.8 その他オプション

No.	オプション	内容	ダイアログメニュー
1	-.	画面へのメッセージ出力を停止する	アセンブラ〈その他〉
2	-C	as30 が mac30、pre30 及び asp30 に渡したコマンド行を表示する	アセンブラ〈その他〉
3	-F	..FILE が示すファイル名をソースファイル名に固定する	アセンブラ〈その他〉
4	-V	全てのプログラムのバージョン番号を表示する	アセンブラ〈その他〉
5	-subcommand=〈ファイル名〉	コマンドラインをファイルから入力する	アセンブラ〈その他〉[ユーザー指定オプション]
6	-T	アセンブラエラータグファイルを生成する	アセンブラ〈その他〉[ユーザー指定オプション]
7	-X	タグファイルを引数として外部プログラムを起動する	アセンブラ〈その他〉[ユーザー指定オプション]

画面へのメッセージ出力停止

-.

書 式　　-.

説 明　　as30 が処理を行う際のメッセージを出力しません。
エラーメッセージ、ウォーニングメッセージ及び指示命令".ASSERT"によるメッセージは出力されます。

記述例

```
>as30 -. sample.a30
```

出力例

-. 指定時)

```
>as30 -. sample.a30
sample.a30(2): A2225 (E) Section type is not appropriate
```

-. 未指定時)

```
>as30 sample.a30
M16C Series and R8C Family Assembler system Version 6.00.00
Copyright (C) 1995 (1996 - 2010) Renesas Electronics Corporation and
Renesas Solutions Corp. All rights reserved.

(sample.asm )
macro processing now

assembler processing now
sample.a30(2): A2225 (E) Section type is not appropriate

TOTAL ERROR(S)    00001
TOTAL WARNING(S)  00000
TOTAL LINE(S)     00007   LINES
CODE      0000003(00003H) program
```

コマンド起動行表示

-C

書 式　　-C

説 明　　as30 が mac30、pre30 及び asp30 を起動する際に付加したコマンドオプションを画面上で確認できます。

記述例1　　"-C"が指定されている場合は次のように出力されます(コピーライトメッセージ、" All rights reserved."の次の行から示します)。

```
>as30 -C -L sample

( sample.a30 )
mac30.exe -L -rREV.F sample.a30
macro processing now

asp30.exe -finfo -no_utl -G -L sample.m30
assembler processing now
```

記述例2　　画面へのメッセージ出力停止オプションと組み合わせた場合は次のように出力されます。

```
>as30 -. -C -L sample

( sample.a30 )
mac30.exe -L -rREV.F sample.a30
asp30.exe -finfo -no_utl -G -L sample.m30
```

..FILE 展開制御

-F

書 式　　-F

説 明　　指示命令..FILE が展開するファイル名を、コマンド行から指定されたアセンブラソースファイル名に固定します。

記述例

```
>as30 -F sample.a30
```

sample.a30 アセンブラソースファイルがインクルードしているファイル"include.inc"内に記述されている指示命令"..FILE"が展開するファイル名が"sample"となります。本オプションが指定されていない場合は、"..FILE"が展開するファイル名は、"include"となります。

バージョン表示

-V

書 式　　-V

説 明　　as30 に含まれる全てのプログラムのバージョン番号を表示して、処理を終了します。
コマンド行の他のパラメータは全て無視されます。
オブジェクトファイルは出力されません。

記述例

```
>as30 -V
```

備 考　　本オプションのみを指定してください。

コマンドラインをファイルから入力

-subcommand

書 式　　-subcommand=<ファイル名>

説 明　　subcommand オプション指定時は、アセンブラ起動時のアセンブラオプションをサブコマンドファイルで指定します。
サブコマンドファイル中の書式は、コマンドラインの書式と同一です。

記述例

サブコマンドファイル opt.sub の内容）

```
-L -H
```

サブコマンドの指定）

```
>as30 -subcommand=opt.sub sample.a30
```

アセンブラでの解釈）

```
>as30 -L -H sample.a30
```

備 考　　サブコマンドファイル内に -subcommand を指定することはできません。

アセンブラエラータグファイル生成

-T

書 式　　-T

説 明　　アセンブラエラーまたはウォーニングが発生した場合に、アセンブラエラータグファイルを生成します。
エディタのタグジャンプ機能を利用可能なフォーマットでファイルを出力します。
本オプションを指定しても、エラーがゼロの場合はファイルを生成しません。
エラーが発生した場合は、オブジェクトファイルは生成しません。なお、ウォーニングのみの発生した場合は、オブジェクトファイルを生成します。
エラータグファイル名は、アセンブラソースファイル名の拡張子を".atg"に変更したものになります。

記述例

```
>as30 -T sample.a30
```

エラーが発生した場合、"sample.atg"ファイルを生成します

外部プログラムを起動

-X

書 式　　-X＜外部プログラム＞

説 明　　アセンブラエラータグファイルを生成し、"-X"に続けて指定した実行プログラムを起動します。
エラーが発生した場合、'-T'の指定の有無に関わらずアセンブラエラータグファイルを生成し指定されたプログラムでアセンブラエラータグファイルを開きます。

記述例

```
>as30 -Xedit sample.a30
```

備 考　　本オプションとプログラム名の間に、スペース又はタブは記述できません。

5.5.6 CPUオプション

表 5.9 CPU オプション

No.	オプション	内容	ダイアログメニュー
1	-R8C	R8C シリーズに対応したコードを生成する (メモリ空間 0H 番地 ～ 0FFFFH 番地)	CPU<CPU タイプ> [R8C ファミリ(ROM 64KB 未満)向けコード生成]
	-R8CE	R8C シリーズに対応したコードを生成する (メモリ空間 0H 番地 ～ 0FFFFFH 番地)	CPU<CPU タイプ> [R8C ファミリ(ROM 64KB 以上)向けコード生成]
2	-R8Cxx	チップセレクト付クロック同期シリアル(SSU)または I²C バスインタフェース(IIC)の注意事項を回避するコードを生成する	CPU<CPU タイプ> [R8C ファミリ(ROM 64KB 未満)向けコード生成] R8C/14,15,16,17 の制限事項回避設定

生成コード制御

-R8C/-R8CE

書 式　-R8C
-R8CE

説 明　R8C シリーズに対応したコードを生成します。

表 5.10 メモリ空間

オプション指定	メモリ空間
-R8C	0H 番地 ～ 0FFFFH 番地
-R8CE	0H 番地 ～ 0FFFFFH 番地

記述例

```
>as30 -R8C sample.a30
```

備 考　本オプションは大文字で指定してください。

注意事項　本オプション指定時、シンボル定数設定オプション"-D_R8C_=1"が付加されます。
"-PATCH[6N]_TA"または"-PATCH[6N]_TAn"オプションと同時に指定することはできません。
本オプション指定時、指示命令".SVECTOR"は使用できません。
本オプション指定時、スペシャルページ分岐命令(JMPS)およびスペシャルページサブルーチン呼び出し命令(JSRS)を使用することはできません。

生成コード制御

-R8Cxx

書 式　　-R8C＜グループ名＞

説 明　　チップセレクト付クロック同期シリアル(SSU)または I2C バスインタフェース(IIC)の注意事項を回避するコード生成を行います。(上記レジスタのアドレス値がアセンブル時確定する場合のみ対象となります。)
SSU または IIC 機能に該当するグループ名が指定された場合、メッセージ"R8C/xx group in information file 'r8ctiny.txt' is used."を出力します。
上記以外は、オプション"-R8C"指定時と同一機能になります。

記述例

```
 >as30 -R8C14 sample.a30
```

R8C/14 の SSU に関する注意事項を回避するコードを生成します。

ソースファイル記述例)

```
 .glb     P1,SSCRH
 .section   prg
mov.b    #10H,P1
mov.b    #03H,SSCRH
 .end
```

リストファイル出力例)

```
 1                  .glb    P1,SSCRH
 2                  .section test
 3 00000  C710E100  mov.b   #10H,P1
 4 00004  C703B800  mov.b   #03H,SSCRH
          FE01      ; Generates code to escape precautions on the SSU or IIC register
 5                  .end
```

備 考　　本オプションは大文字で指定してください。

注意事項　　注意事項については、RENESAS TECHNICAL UPDATE を参照してください。
統合開発環境 High-performance Embedded Workshop をご使用の場合、本オプションは"オプション"→"Renesas M16C Standard Toolchain"の"CPU"から設定してください。
統合化開発環境 TM をご使用の場合、"オプションブラウザ"の"CFLAGS → 一般"または"AFLAGS → コード生成ターゲットの選択"から設定してください。
"-PATCH(6N)_TA"または"-PATCH(6N)_TAn"オプションと同時に指定することはできません。
SSU または IIC 機能が搭載されていないグループについては、本オプションを指定する必要はありません。

6. 最適化リンケージエディタ操作方法

6.1 オプション指定規則

6.1.1 コマンドラインの形式

コマンドラインの形式は以下のとおりです。

optlnk [{△<ファイル名>|△<オプション列>}. . .]
<オプション列>:-<オプション>[= <サブオプション>[,. . .]

6.1.2 サブコマンドファイルの形式

サブコマンドファイルの形式は以下のとおりです。

<オプション> {=|△}[<サブオプション>[,. . .]] [△&] [;<コメント>]
&:継続行指定

サブコマンドファイル形式の詳細は、「6.2.8 サブコマンドファイルオプション」を参照してください。

6.2 オプション解説

オプション、サブオプションの英大文字は短縮形指定時の文字を、下線は省略時解釈を示します。

また、統合開発環境の対応するダイアログメニューを、タブ名<カテゴリ名>[項目]…で示します。オプションの順序は、統合開発環境のタブとその中のカテゴリに対応しています。

ファイル名、パス名には、括弧記号"("および")"を含まないようにしてください。

6.2.1 入力オプション

表 6.1 入力カテゴリオプション一覧

	項目	オプション	ダイアログメニュー	指定内容
1	入力 ファイル	Input = <sub> [[,|△]...] <sub>: <ファイル名> [(<モジュール名>[,...])]	リンカ<入力> [オプション項目 :] [リロケータブルファイル/オブジェクトファイル]	入力ファイルを指定 (コマンドラインでは input なしで指定します)
2	ライブラリ ファイル	LIBrary = <ファイル名>[,…]	リンカ<入力> [オプション項目 :] [ライブラリファイル]	入力ライブラリファイルを指定
3	バイナリ ファイル	Binary = <sub>[,...] <sub> : <ファイル名> (<セクション名> [:<アライメント数>] [,<シンボル名>])	リンカ<入力> [オプション項目 :] [バイナリファイル]	入力バイナリファイルを指定
4	シンボル 定義	DEFine = <sub>[,...] <sub>: <シンボル名> = {<シンボル名> | <数値> }	リンカ<入力> [オプション項目 :] [シンボル定義]	未定義シンボルの強制定義 シンボル名と同値として定義 数値で定義
5	実行開始 アドレス	ENTry = { <シンボル名> | <アドレス> }	リンカ<入力> [エントリポイント :]	エントリシンボルを指定 エントリアドレスを指定
6	プレリンカ	NOPRElink	リンカ<入力> [プレリンカ制御 :]	プレリンカの起動を抑止

入力ファイル

Input

リンカ<入力>[オプション項目 :][リロケータブルファイル/オブジェクトファイル]

書 式　Input = <サブオプション>[{,|△}…]
<サブオプション> : <ファイル名>[(<モジュール名>[,…])]

説 明　入力ファイルを指定します。複数ある場合にはカンマ(,)または空白文字で区切って指定します。
ワイルドカード(*,?)も指定できます。ワイルドカードで指定した文字列はアルファベット順に展開します。数字と英文字は数字が先、英大文字と英小文字は英大文字が先になります。
入力ファイルとして指定できるのは、コンパイラ、アセンブラ出力オブジェクトファイル、最適化リンケージエディタ出力のリロケータブルファイルおよびアブソリュートファイルです。またライブラリ名(<モジュール名>)の形式で、ライブラリ内モジュールを入力ファイルとして指定することもできます。モジュール名は拡張子なしで指定します。
入力ファイル名に拡張子の指定がない場合、モジュール名がない時は「obj」、モジュール名がある時は「lib」を仮定します。

例　input=a.obj lib1(e)　; a.obj と lib1.lib 内のモジュール e を入力します
input=c*.obj　; c で始まる拡張子 obj のファイルを全て入力します

備 考　form=object および extract 指定時、本オプションは無効です。
コマンドライン上で入力ファイルを指定する場合は、input 無しで指定します。

ライブラリファイル

LIBrary

リンカ<入力>[オプション項目 :][ライブラリファイル]

書 式　LIBrary = <ファイル名>[,…]

説 明　ライブラリファイルを指定します。複数ある場合にはカンマ(,)で区切って指定します。
ワイルドカード(*,?)も指定できます。ワイルドカードで指定した文字列はアルファベット順に展開します。数字と英文字は数字が先、英大文字と英小文字は英大文字が先になります。
入力ファイル名に拡張子の指定がない場合は、「lib」を仮定します。
form=library オプションまたは extract オプション指定時は、ライブラリファイルを編集対象ライブラリとして入力します。
それ以外の場合は、入力ファイルとして指定されたファイル間でのリンケージ処理後に、未定義シンボルをライブラリファイルから検索します。
ライブラリファイル内シンボルの検索は、ライブラリオプション指定ユーザライブラリファイル(指定順)、ライブラリオプション指定システムライブラリファイル(指定順)、デフォルトライブラリ(環境変数 HLNK_LIBRARY1,2,3)の順序で行います。

例　library=a.lib,b　; a.lib と b.lib を入力します。
library=c*.lib　; c で始まる拡張子 lib のファイルを全て入力します。

バイナリファイル

Binary

リンカ<入力>[オプション項目 :] [バイナリファイル]

書 式　Binary = <サブオプション>[,…]
<サブオプション> : <ファイル名>(<セクション名>[:<アライメント数>][/<セクション属性>][,<シンボル名>])
<セクション属性> : CODE | DATA
<アライメント数> : 1 | 2 | 4 | 8 | 16 | 32 (デフォルトは 1)

説 明　入力バイナリファイルを指定します。複数ある場合にはカンマ(,)で区切って指定します。
ファイル名に拡張子の指定がない場合は、「bin」を仮定します。
入力したバイナリデータは、指定したセクションのデータとして配置します。セクションのアドレスは start オプションで指定します。セクションは省略できません。
またシンボルを指定することにより、定義シンボルとしてリンクすることもできます。C/C++プログラムで参照している変数名の場合、プログラム中での参照名先頭に_を付加します。
本オプションで指定したセクションには、セクション属性、アライメント数の指定が可能です。
セクション属性は、CODE または DATA を指定できます。
セクション属性の指定が無い場合、デフォルトとして書き込み、読み取り、実行全ての属性が有効になります。
アライメント数に指定可能な値は 2 の累乗です。それ以外の値を指定することはできません。
アライメント数の指定がない場合、デフォルト値として 1 が有効になります。

例　input=a.obj
start=P,D*/200
binary=b.bin(D1bin),c.bin(D2bin:4,_datab)
form=absolute

b.bin を D1bin セクションとして、0x200 番地から配置します。
c.bin を D2bin セクション(アライメント数 4)として、D1bin の後に配置します。
c.bin データを定義シンボル_datab としてリンクします。

備 考　form={object | library}または strip 指定時、本オプションは無効です。
また入力オブジェクトファイル指定がない場合、本オプションは指定できません。

シンボル定義

DEFine

リンカ<入力>[オプション項目 :] [シンボル定義]

書 式　DEFine = <サブオプション>[,…]
<サブオプション> : <シンボル名> = {<シンボル名> | <数値>}

説 明　未定義シンボルを外部定義シンボルまたは数値で強制定義します。
数値は 16 進数で指定します。先頭が A～F の場合は先にシンボルを検索し、該当するシンボルがなければ数値として解釈します。先頭に 0 を付加した場合は常に数値と解釈します。シンボル名が C/C++変数名の場合、プログラム中での定義名先頭に_を付加します。C++関数名の場合は(main 関数は除く)引数列を含めたプログラム中の定義名をダブルクォーテーションで囲んで指定します。ただし引数が void の場合は、"関数名()"で指定します。

例　define=_sym1=data　;_sym1 を外部定義シンボル data と同値として定義します。
define=_sym2=4000　;_sym2 を 0x4000 として定義します。

備 考　form={object | relocate | library}指定時、本オプションは無効です。

実行開始アドレス

ENTry

リンカ<入力>[エントリポイント :]

書 式　ENTry = {<シンボル名> | <アドレス>}

説 明　実行開始アドレスを外部定義シンボルまたはアドレスで指定します。
アドレスは 16 進数で指定します。先頭が A～F の場合は先に定義シンボルを検索し、該当するシンボルがなければアドレスと判断します。先頭に 0 を付加した場合は常にアドレスと解釈します。
シンボル名は、C 関数名の場合プログラム中での定義名先頭に_を付加します。C++関数名の場合は(main 関数は除く)引数列を含めたプログラム中の定義名をダブルクォーテーションで囲んで指定します。ただし引数が void の場合は、"関数名()"で指定します。
コンパイル、アセンブル時に entry シンボルを指定している場合、本オプション指定を優先します。

例
```
entry=_main          ;C/C++の main 関数を実行開始アドレスとして設定します。
entry="init()"       ;C++の init 関数を実行開始アドレスとして設定します。
entry=100            ;0x100 を実行開始アドレスとして設定します。
```

備 考　form={object | relocate | library}または strip 指定時、本オプションは無効です。
未参照シンボル削除最適化(optimize=symbol_delete)指定時には、実行開始アドレスは必ず必要です。指定がない場合は、未参照シンボル削除最適化指定は無効です。本オプションでアドレスを指定している場合は、未参照シンボル削除最適化を無効にします。

プレリンカ

NOPRElink

リンカ<入力>[プレリンカ制御 :]

書 式　NOPRElink

説 明　プレリンカの起動を抑止します。
プレリンカは、C++テンプレートインスタンスの自動生成機能および実行時型検査機能をサポートします。C++テンプレート機能および実行時型検査機能を使用していない場合は、noprelink オプションを指定してください。リンク時間が短くなります。

備 考　extract または strip 指定時、本オプションは無効です。
C++テンプレート機能および実行時型検査機能を使用し、form=lib または、form=rel を指定する場合には、noprelink を指定しないでください。

6.2.2 出力オプション

表 6.2 出力カテゴリオプション一覧

	項目	オプション	ダイアログメニュー	指定内容
1	出力形式	FOrm = { Absolute | Relocate | Object | Library [= {S | U}] | Hexadecimal | Stype | Binary }	リンカ<出力> [出力形式 :]	アブソリュート形式 リロケータブル形式 オブジェクト形式 ライブラリ形式 インテル HEX 形式 モトローラ S 形式 バイナリ形式
2	デバッグ情報	DEBug SDebug NODEBug	リンカ<出力> [デバッグ情報 :]	出力あり(出力ファイル内) デバッグ情報ファイル出力 出力なし
3	レコードサイズ統一	REcord = { H16 | H20 | H32 | S1 | S2 | S3 }	リンカ<出力> [レコードサイズ統一 :]	インテル HEX レコード インテル拡張 HEX レコード インテル 32bitHEX レコード S1 レコード S2 レコード S3 レコード
4	ROM 化支援	ROm = <sub>[,…] <sub> : <ROM セクション名> =<RAM セクション名>	リンカ<出力> [オプション項目 :] [ROM から RAM へマップするセクション]	RAM に領域を確保し、シンボルを RAM 上のアドレスでリロケーション解決
5	出力ファイル	OUtput = <sub>[,…] <sub> : <ファイル名> [=<出力範囲>] <出力範囲>: { <先頭アドレス> - <終了アドレス> | <セクション名>[:…] }	リンカ<出力> [オプション項目 :] [出力ファイル/インフォメーション抑止] [出力ファイルの分割]	出力ファイルを指定 (範囲指定、分割出力可能)
6	外部シンボル割り付け情報ファイル	MAp [= <ファイル名>]	リンカ<出力> [外部シンボル割り付け情報ファイル出力]	外部シンボル割り付け情報ファイル出力を指定(SuperH ファミリ, RX ファミリ向け)
7	空きエリア出力指定	SPace [= {<数値> | Random}]	リンカ<出力> [オプション項目 :] [空きエリア出力指定] [空きエリア出力]	空きエリアへの出力値の指定
8	インフォメーションメッセージ	Message_ NOMessage [= <sub>[,…]] <sub> : <エラー番号> [- <エラー番号>]	リンカ<出力> [オプション項目 :] [出力ファイル/インフォメーション抑止] [インフォメーションレベルメッセージ抑止]	出力あり 出力なし (エラー番号、範囲指定可能)

	項目	オプション	ダイアログメニュー	指定内容
9	参照されない定義シンボルの通知	MSg_unused	リンカ<出力> [オプション項目 :] [メッセージ出力指定] [参照されない定義シンボルの通知]	1 回も参照されない定義シンボルをメッセージ出力により通知
10	セクション内データの詰め込み配置	DAta_stuff	リンカ<出力> [オプション項目 :] [セクション内データの詰め込み配置]	コンパイル単位間の空き領域を詰めてデータを配置(SuperHファミリ, H8,H8S,H8SXファミリ向け)
11	データレコードのバイト数指定	BYte_count=<数値>	リンカ<出力> [データレコード長 :]	データレコードの最大バイト数を指定
12	CRC 演算	CRc = <サブオプション> <サブオプション>: <出力位置>=<計算範囲>[/<多項式>][:<エンディアン>] <出力位置>:<アドレス> <計算範囲>: <先頭アドレス>-<終了アドレス>[,...] <多項式> :{ CCITT | 16 } <エンディアン>: {BIG|LITTLE}	リンカ<出力> [オプション項目 :] [CRC コード]	リンク時に計算範囲の CRC(Cyclic Redundancy Check)演算を行い、計算結果を出力位置に埋め込む
13	セクション終端にパディング	PADDING	リンカ<出力> [パディング]	アライメントにあわせてセクション終端にパディングを出力
14	特定ベクタ番号のアドレス設定	VECTN=<サブオプション>[,...] <サブオプション>: <ベクタ番号>={<シンボル>|<アドレス>}	リンカ<出力> [オプション項目 :] [ベクタ] [特定ベクタ :]	可変ベクタの特定ベクタ番号へのアドレスを設定(RX ファミリ、M16C ファミリ向け)
15	可変ベクタの空き領域のアドレス設定	VECT={<シンボル>|<アドレス>}	リンカ<出力> [オプション項目 :] [ベクタ] [空きベクタ :]	可変ベクタの空き領域へのアドレスを設定(RX ファミリ,M16C ファミリ向け)
16	utl30 情報出力	UTL	リンカ<出力> [UTL 情報]	UTL30 向け情報を出力(M16C ファミリ向け)
17	ジャンプテーブル出力	JUMP_ENTRIES_FOR_PIC=<セクション名>[,...]	リンカ<出力> [ジャンプテーブル出力]	ジャンプテーブルを出力(RX ファミリ向け)

出力形式

FOrm

リンカ<出力>[出力形式 :]

書 式　　FOrm = {Absolute | Relocate | Object | Library[={S|U}]
| Hexadecimal | Stype | Binary}

説 明　　出力形式を指定します。
本オプションの省略時解釈は、form=absolute です。サブオプションの一覧を表 6.3 に示します。

表 6.3 form オプションのサブオプション一覧

	サブオプション名	内 容
1	absolute	アブソリュートファイルを出力します。
2	relocate	リロケータブルファイルを出力します。
3	object	オブジェクトファイルを出力します。extract オプションでライブラリから 1 個のモジュールをオブジェクトファイルとして取り出すときに使用します。
4	library	ライブラリファイルを出力します。 library=s 指定時、出力ライブラリファイルをシステムライブラリとします。 library=u 指定時、出力ライブラリファイルをユーザライブラリとします。 省略時解釈は、library=u です。
5	hexadecimal	インテル HEX 形式ファイルを出力します。インテル HEX フォーマットは「13.1.2 インテル HEX 形式ファイル」を参照してください。
6	stype	モトローラ S 形式ファイルを出力します。モトローラ S フォーマットは「13.1.1 モトローラ S 形式ファイル」を参照してください。
7	binary	バイナリファイルを出力します。

備 考　　出力形式と入力ファイル、他オプションとの関係を表 6.4 に示します。

表 6.4 出力形式と入力ファイル、他オプションとの関係

	出力形式	指定オプション	入力可能なファイル形式	指定可能なオプション*1
1	Absolute	strip あり	アブソリュートファイル	input, output
		上記以外	オブジェクトファイル リロケータブルファイル バイナリファイル ライブラリファイル	input, library, binary, debug/nodebug, sdebug, cpu, ps_check, start, rom, entry, output, map, hide, optimize/nooptimize, samesize, symbol_forbid, samecode_forbid, variable_forbid, function_forbid, section_forbid, absolute_forbid, profile, cachesize, sbr, compress, rename, delete, define, fsymbol, stack, noprelink, memory, msg_unused, data_stuff, show=symbol, reference, xreference
2	Relocate	extract あり	ライブラリファイル	library, output
		上記以外	オブジェクトファイル リロケータブルファイル バイナリファイル ライブラリファイル	input, library, debug/nodebug, output, hide, rename, delete, noprelink, msg_unused, data_stuff, show=symbol, xreference
3	Object	extract あり	ライブラリファイル	library, output
4	Hexadecimal Stype Binary		オブジェクトファイル リロケータブルファイル バイナリファイル ライブラリファイル	input, library, binary, cpu, ps_check, start, rom, entry, output, map, space, optimize/nooptimize, samesize, symbol_forbid, samecode_forbid, variable_forbid, function_forbid, section_forbid, absolute_forbid, profile, cachesize, sbr, rename, delete, define, fsymbol, stack, noprelink, record, s9*2, byte_count*3, memory, msg_unused, data_stuff, show=symbol, reference, xreference
			アブソリュートファイル	input, output, record, s9*2, byte_count*3, show=symbol, reference, xreference
5	Library	strip あり	ライブラリファイル	library, output, memory*4, show=symbol, section
		extract あり	ライブラリファイル	library, output
		上記以外	オブジェクトファイル リロケータブルファイル	input, library, output, hide, rename, delete, replace, noprelink, memory*4, show=symbol, section

【注】 *1 message/nomessage, change_message, logo/nologo, form, list, subcommand は常に指定できます。
*2 s9 は出力形式が form=stype のときだけ指定できます。
*3 byte_count は出力形式が form= hexadecimal のときだけ指定できます。
*4 hide 指定する場合は使用できません。

デバッグ情報

DEBug
SDebug
NODEBug

リンカ<出力>[デバッグ情報 :]

書 式　DEBug
SDebug
NODEBug

説 明　debug 情報の出力有無を指定します。
debug オプションは、出力ファイル中にデバッグ情報を出力します。
sdebug オプションは、〈出力ファイル名〉.dbg ファイルにデバッグ情報を出力します。
nodebug オプションは、デバッグ情報を出力しません。
form=relocate 指定時に sdebug オプションを指定したときは、debug オプションと解釈します。
output オプションで複数ファイル出力を指定時に debug オプションを指定したときは、sdebug オプションと解釈して、〈先頭出力ファイル名〉.dbg に出力します。
本オプション省略時解釈は、debug です。

備 考　form={object | library | hexadecimal | stype | binary}、strip、または、extract 指定時、本オプションは無効です。

レコードサイズ統一

REcord

リンカ<出力>[レコードサイズ統一 :]

書 式　REcord = {H16 | H20 | H32 | S1 | S2 | S3}

説 明　アドレス範囲に関係なく、一定のデータレコードで出力します。
指定したデータレコードより大きいアドレスが存在した場合、アドレスに合わせてデータレコードを選択します。
本オプション省略時は、それぞれのアドレスに合わせて混在したデータレコードを出力します。

備 考　form=hexadecimal または stype 指定がないとき、本オプションは無効です。

ROM 化支援

ROm

リンカ<出力>[オプション項目 :][ROM から RAM へマップするセクション]

書 式　ROm = 〈サブオプション〉[,…]
〈サブオプション〉:〈ROM セクション名〉=〈RAM セクション名〉

説 明　初期化データ領域の ROM 用、RAM 用領域を確保し、ROM セクション内定義シンボルを RAM セクション内アドレスになるようリロケーションします。
ROM セクションには初期値のあるリロケータブルセクションを指定します。
RAM セクションには存在しないセクションまたはサイズ 0 のリロケータブルセクションを指定します。

例　rom=D=R
start=D/100,R/8000
D セクションと同サイズの R セクションを確保し、D セクション内定義シンボルを R セクション上のアドレスでリロケーションします。

備 考　form={object | relocate | library}または strip 指定時、本オプションは無効です。

出力ファイル

OUtput

リンカ<出力>[オプション項目 :][出力ファイル/インフォメーション抑止][出力ファイルの分割]

書 式　OUtput ＝ 〈サブオプション〉[,…]
〈サブオプション〉: 〈ファイル名〉[=〈出力範囲〉]
〈出力範囲〉: {〈先頭アドレス〉-〈終了アドレス〉| 〈セクション名〉[:…]}

説 明　出力ファイル名を指定します。form={absolute | hexadecimal | stype | binary}のときは、複数ファイルを指定できます。アドレスは 16 進数で指定します。先頭が A～F の場合は先にセクションを検索し、該当するセクションがなければアドレスと判断します。先頭に 0 を付加した場合は常にアドレスと解釈します。
本オプションの省略時解釈は、〈先頭入力ファイル名〉.〈デフォルト拡張子〉です。
デフォルト拡張子は、次のようになります。
form=absolute:「abs」、form=relocate :「rel」、form=object:「obj」
form=library :「lib」、form=hexadecimal:「hex」、form=stype :「mot」
form=binaruy :「bin」

例　output=file1.abs=0-ffff,file2.abs=10000-1ffff
0～0xffff 間を file1.abs に、0x10000～0x1ffff 間を file2.abs に出力します。

output=file1.abs=sec1:sec2,file2.abs=sec3
sec1,sec2 セクションを file1.abs に、sec3 セクションを file2.abs に出力します。

備 考　マイコン種別が RX ファミリでビッグエンディアンのときに、セクション単位で出力する場合は、セクションサイズを 4 の倍数にしてください。

外部シンボル割り付け情報ファイル出力

MAp

リンカ<出力>[外部シンボル割り付け情報ファイル出力]

書 式　MAp [= <ファイル名>]

説 明　コンパイラが外部変数アクセス最適化で使用する外部変数割り付け情報ファイルを出力します。
<ファイル名>を指定しなかった場合は、output オプションで指定したファイル名、もしくは先頭入力ファイル名で、拡張子が bls のファイルを出力します。
外部変数割り付け情報ファイル作成時の変数宣言順と、再コンパイル後のオブジェクトを読み込んだ時の変数宣言順が変わっている場合はエラーを出力します。

備 考　form={absolute | hexadecimal | stype | binary}を指定した場合のみ、本オプションは有効です。
マイコン種別が SuperH ファミリおよび RX ファミリで有効です。

空きエリア出力指定

SPace

リンカ<出力>[オプション項目 :] [空きエリア出力指定] [空きエリア出力]

書 式　SPace [= {<数値> | Random}]

説 明　出力範囲のメモリの空き領域を、ユーザが指定するデータで充填します。
充填するデータとしては、乱数、もしくは 16 進数の数値を指定することができます。
空きエリアを埋める方法は、output オプション指定時の出力範囲指定方法によって下記のように異なります。
・出力範囲:セクション指定
　指定されたセクション間に空きが存在した場合に指定データを出力
・出力範囲:アドレス範囲指定
　指定された範囲内に空きが存在した場合に指定データを出力
出力データサイズは、1,2,4 バイト単位で有効となります。出力データサイズは space オプションに指定する 16 進数の個数で決まります。3 バイトデータを指定した場合、上位桁を 0 拡張し 4 バイトのデータとして扱われます。
また、奇数桁データを指定した場合も、上位桁に 0 拡張して偶数桁入力として扱われます。
空きエリアのサイズが出力データサイズの倍数でない場合、出力できるだけ出力し、メッセージによる警告を行います。

備 考　本オプションにてサブオプションの指定がされなかった場合は、空きエリアへの出力は行いません。
本オプションは form={ binary |stype | hexadecimal}オプションを指定した場合にのみ有効となります。
output オプションによる出力範囲指定がされなかった場合は、本オプション指定は無効となります。

インフォメーションメッセージ

Message
NOMessage

リンカ<出力>[オプション項目 :][出力ファイル/インフォメーション抑止][インフォメーションレベルメッセージ抑止]

書 式　Message
NOMessage [= 〈サブオプション〉[,…]]
〈サブオプション〉:〈エラー番号〉[-〈エラー番号〉]

説 明　インフォメーションレベルメッセージの出力有無を指定します。
message オプション指定時は、インフォメーションレベルメッセージを出力します。
nomessage オプション指定時は、インフォメーションレベルメッセージの出力を抑止します。またエラー番号を指定すると、指定したエラー番号のメッセージ出力を抑止できます。ハイフン(-)を使用して抑止するエラー番号の範囲を指定することもできます。エラー番号としてウォーニング、エラーレベルメッセージ番号を指定した場合、change_message でインフォメーションレベルに変更したと仮定し、メッセージ出力を抑止します。
本オプションの省略時解釈は nomessage です。

例　nomessage=4,200-203,1300
L0004 および L0200～L0203 および L1300 のメッセージ出力を抑止します。

参照されない定義シンボルの通知

MSg_unused

リンカ<出力>[オプション項目 :][メッセージ出力指定][参照されない定義シンボルの通知]

書 式　MSg_unused

説 明　本オプションを指定した場合、リンク処理の中で一度も参照されることのなかった外部定義シンボルを、メッセージ出力によってユーザに知らせます。

例　optlnk -msg_unused a.obj

備 考　入力ファイルが absolute 形式の場合、本オプション指定は無効です。
メッセージ出力させるためには、同時に message オプションの指定が必要です。
コンパイル時にインライン展開された関数に対してメッセージ出力する場合があります。その場合、関数定義に static 宣言することで、メッセージ出力を抑えることができます。
以下のいずれかに該当する場合、参照関係の解析が正しく行うことができず、メッセージ出力により通知される情報が不正確となります。
- アセンブル時に goptimize オプションが指定されておらず、同一ファイル内、かつ同一セクションへの分岐がある場合（マイコンが H8,H8S,H8SX ファミリの場合のみ）
- 同一ファイル内の定数シンボルへの参照
- コンパイル時に最適化が有効で、直下の関数を呼び出す場合
- コンパイル時に外部変数アクセス最適化が有効な場合(マイコンが SuperH ファミリの場合のみ)
- ソースファイル上で#pragma tbr を記述した際にオフセット値を直接指定している場合(マイコンが SH-2A/SH2A-FPU の場合のみ)
- リンク時の最適化によって、定数やリテラルの統合が生じる場合

DAta_stuff

リンカ<出力>[オプション項目 :][セクション内データの詰め込み配置]

書 式　　DAta_stuff

説 明　　リンク時に、セクション内のデータを詰め込んで配置します。本オプション機能の対象となるセクションは、定数領域、初期化データ領域、未初期化データ領域です。
本オプションを指定した場合、コンパイル単位のセクションのアライメントにより生じる空き領域を詰めてリンクを行います。
ただし、データの配置順は変更しません。
本オプションを指定しない場合、コンパイル単位のセクションのアライメントに従いリンクを行います。本オプションの指定により、アライメントで生じる冗長な空き領域を詰めることができ、データセクション全体のサイズ低減が期待できます。

例

〈tp1.c〉
```
long a;
char b,c;
```

〈tp2.c〉
```
char   d;
long   e;
char   f;
```

〈コンパイル後のデータセクションサイズ（SuperH ファミリ用コンパイラの出力例)〉
tp1.obj : 4+1+1 = 6 バイト
tp2.obj : 1+3[*]+4+1 = 9 バイト

〈tp1.obj と tp2.obj、リンク後のデータセクションサイズ〉
1） data_stuff 指定なし
オブジェクトファイルを各セクションのアライメントに従ってリンクします(従来処理)。
6 バイト[tp1] + 2 バイト[*] + 9 バイト[tp2] = 17 バイト

2） data_stuff 指定あり
セクション内のデータを詰めて配置させ、アライメントによる 冗長な空き領域を埋めてリンクします。
(4+1+1)バイト + 1 バイト + 1 バイト[*] + 4 バイト + 1 バイト = 13 バイト

【注 1】 *: アライメントのために生じる空き領域
【注 2】 コンパイル後のデータセクションサイズは、コンパイル時のオプション指定などによって変化するので、上記例のようにならない場合があります。

備 考　　SuperH ファミリ用コンパイラの smap オプションを指定したオブジェクトファイルをリンクする際に本オプションを指定した場合、動作は保証しません。
アセンブラ出力のオブジェクトファイルに対しては、本オプション機能は適用されません。
下記のいずれかの条件の場合、本オプション指定は無効です。
- form=library,object,relocate 指定時
- アブソリュートファイル入力時
- memory=low 指定時
- nooptimize 指定がない場合

本オプションを指定して生成したリロケータブルファイルに対してはリンク時の最適化が適用されません。
マイコン種別が RX ファミリ、M16C シリーズ、R8C ファミリの場合は、本機能を使用できません。

データレコードのバイト数指定

BYte_count

リンカ<出力>[データレコード長 :]

書 式　BYte_count=〈数値〉

説 明　Intel-Hex 形式ファイルを生成する際に、データレコードのバイト数最大値を指定するためのオプションです。バイト数としては、1byte の 16 進数(01～FF)を指定することができます。本オプションを記述しない場合、バイト数最大値は FF として Intel-Hex ファイルを生成します。

例　byte_count=10

備 考　生成するファイル形式が Intel-Hex 形式(form=hex)ではない場合、本オプションは無効です。

CRC 演算

CRc

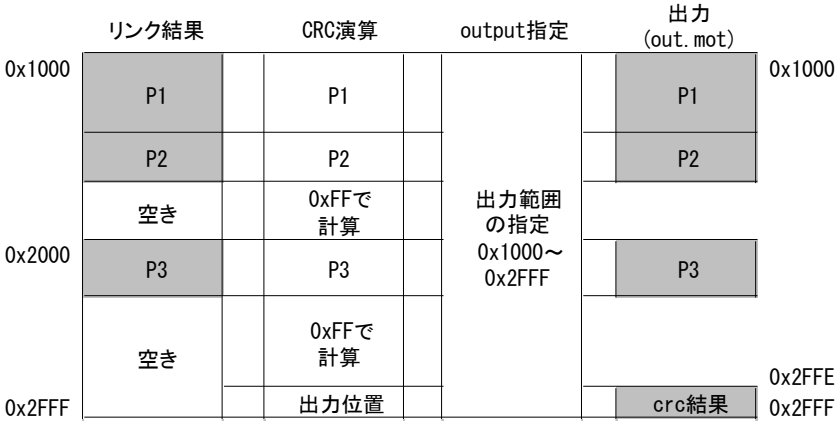
リンカ<出力>[オプション項目 :][CRC コード]

書 式　CRc = 〈サブオプション〉
〈サブオプション〉: 〈出力位置〉=〈計算範囲〉[/〈多項式〉][:〈エンディアン〉]
〈出力位置〉: 〈アドレス〉
〈計算範囲〉: 〈先頭アドレス〉-〈終了アドレス〉[,...]
〈多項式〉 : { CCITT | 16 }
〈エンディアン〉: {BIG|LITTLE}

説 明　計算範囲で指定された内容を下位アドレスから上位アドレスの順で CRC(Cyclic Redundancy Check)演算を行い、計算結果を出力位置のアドレスに出力します。
エンディアンは、RX ファミリの場合に指定可能なオプションです。エンディアンを指定した場合は、エンディアンにしたがって、計算結果を出力位置のアドレスに出力します。指定しない場合は、アブソリュートファイルのエンディアンで計算結果を出力位置のアドレスに出力します。
多項式は CRC-CCITT または CRC-16 を選択できます。(デフォルトは CRC-CCITT)

多項式
CRC-CCITT
X^16+X^12+X^5+1
ビット表現(10001000000100001)
CRC-16
X^16+X^15+X^2+1
ビット表現(11000000000000101)

例 1　　optlnk *.obj -form=stype -start=P1,P2/1000,P3/2000
　　-crc=2FFE=1000-2FFD -output=out.mot=1000-2FFF

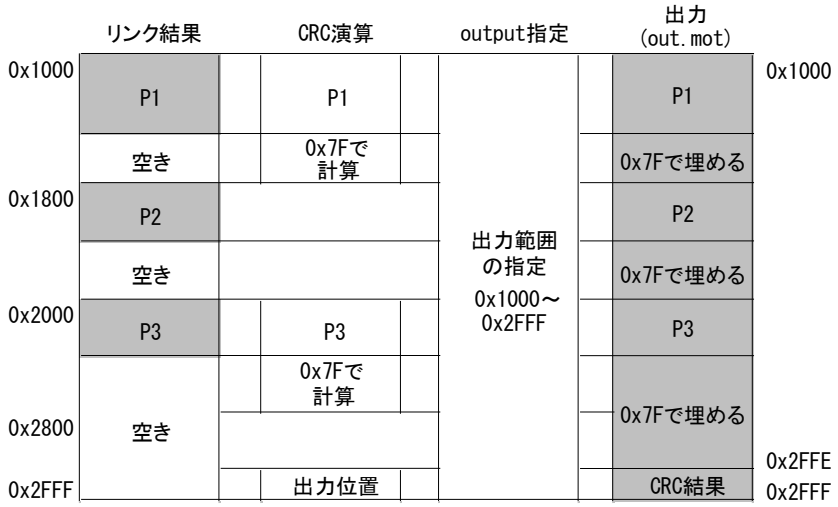


crc オプション：-crc=2FFE=1000-2FFD
　0x1000～0x2FFD の領域に対して CRC 演算を行い、その結果を 0x2FFE 番地に出力します。
　計算範囲にある空き領域は space オプションが指定されていない場合、space=0xFF が指定されていると仮定して、CRC 演算を行います。

output オプション：-output=out.mot=1000-2FFF
　space オプションが指定されていないため、空きの領域は「out.mot」ファイルに出力されません。CRC 演算は、空き領域では 0xFF で計算を行いますが、0xFF を埋めることはありません。

【注】 1.　CRC 出力位置は、計算範囲に含むことは出来ません。
　　2.　CRC 出力位置は output オプションの出力範囲に含まれている必要があります。

例 2　　optlnk *.obj -form=stype -start=P1/1000,P2/1800,P3/2000
-space=7F -crc=2FFE=1000-17FF,2000-27FF
-output=out.mot=1000-2FFF



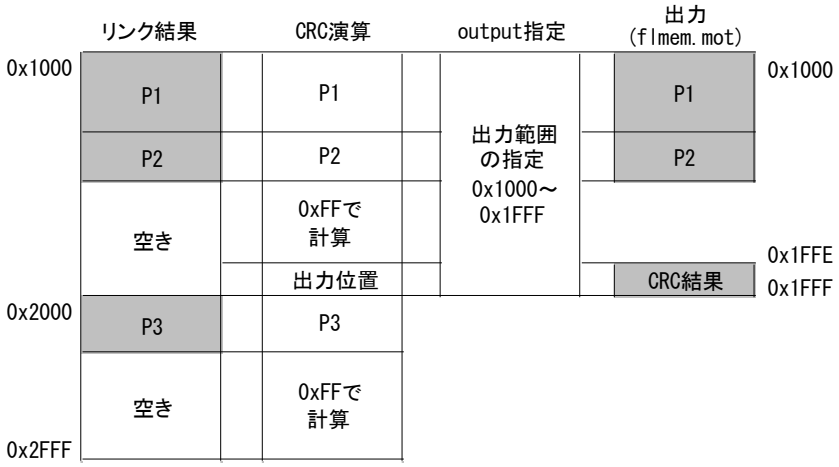
crc オプション：-crc=2FFE=1000-2FFD,2000-27FF
0x1000～0x17FF と 0x2000～0x27FF の2つの領域に対して CRC 演算を行い、その結果を 0x2FFE 番地に出力します。
CRC 演算は計算対象として、連続していない複数の計算範囲を指定できます。

space オプション：-space=7F
指定された計算範囲の空き領域は space オプションの値(0x7F)で計算されます。

output オプション：-output=out.mot=1000-2FFF
space オプションが指定されているため、空き領域は「out.mot」ファイルに出力されます。空き領域は 0x7F で充填されます。

【注】1. CRC 演算の計算順は計算範囲の指定順ではありません。下位アドレスから上位アドレスの順に計算されます。
2. crc オプションと space オプションを同時に指定する場合、space オプションに random または 2バイト以上の値を指定することは出来ません。1 バイトのデータを指定してください。

例 3　　optlnk *.obj -form=stype -start=P1,P2/1000,P3/2000
-crc=1FFE=1000-1FFD,2000-2FFF
-output=flmem.mot=1000-1FFF



crc オプション：-crc=1FFE=1000-1FFD,2000-2FFF
0x1000～0x1FFD と 0x2000～0x2FFF の領域に対して CRC 演算を行い、その結果を 0x1FFE 番地に出力します。
計算範囲にある空き領域は space オプションが指定されていない場合、space=0xFF が指定されていると仮定して、CRC 演算を行います。

output オプション：-output=flmem.mot=1000-1FFF
space オプションが指定されていないため、空きの領域は「flmem.mot」ファイルに出力されません。
CRC 演算は、空き領域では 0xFF で計算を行いますが、0xFF を埋めることはありません。

備　考　　複数のアブソリュートファイル入力時は、本オプションは無効です。
出力形式が form={hexadecimal | stype}の場合に有効です。
space オプションが指定されていない場合で、計算範囲に出力されない空き領域があるとき、空き領域には 0xFF が設定されているものとして CRC の計算が行われます。
CRC 演算の計算範囲にオーバーレイ指定されている領域が含まれる場合はエラーになります。

サンプルコード

crc オプションで計算された CRC 演算結果を比較するためのサンプルコードです。
サンプルコードのプログラムは、optlnk の CRC 演算結果と一致します。

多項式 CRC-CCITT の場合

```
typedef  unsigned  char    uint8_t;
typedef  unsigned  short   uint16_t;
typedef  unsigned  long    uint32_t;

uint16_t CRC_CCITT(uint8_t *pData, uint32_t iSize)
{
  uint32_t ui32_i;
  uint8_t      *pui8_Data;
  uint16_t ui16_CRC = 0xFFFFu;

  pui8_Data = (uint8_t *)pData;

  for(ui32_i = 0; ui32_i < iSize; ui32_i++)
  {
    ui16_CRC  = (uint16_t)((ui16_CRC >> 8u) |
                  ((uint16_t)((uint32_t)ui16_CRC << 8u)));
    ui16_CRC ^= pui8_Data[ui32_i];
    ui16_CRC ^= (uint16_t)((ui16_CRC & 0xFFu) >> 4u);
    ui16_CRC ^= (uint16_t) ((ui16_CRC << 8u) << 4u);
    ui16_CRC ^= (uint16_t)(((ui16_CRC & 0xFFu) << 4u) << 1u);
  }
  ui16_CRC = (uint16_t)( 0x0000FFFFul &
               ((uint32_t)~(uint32_t)ui16_CRC) );
  return ui16_CRC;
}
```

多項式 CRC-16 の場合

```
#define POLYNOMIAL  0xa001      // 生成多項式 CRC-16

typedef  unsigned  char    uint8_t;
typedef  unsigned  short   uint16_t;
typedef  unsigned  long    uint32_t;

uint16_t CRC16(uint8_t *pData, uint32_t iSize)
{
  uint16_t crcdData = (uint16_t)0;
  uint32_t data = 0;
  uint32_t i,cycLoop;

  for(i=0;i<iSize;i++){
    data = (uint32_t)pData[i];
    crcdData = crcdData ^ data;
    for (cycLoop = 0; cycLoop < 8; cycLoop++) {
        if (crcdData & 1) {
            crcdData = (crcdData >> 1) ^ POLYNOMIAL;
        } else {
            crcdData = crcdData >> 1;
        }
    }
  }
  return crcdData;
}
```

セクション終端にパディング埋め込み

PADDING

リンカ<出力>[パディング :]

書 式　PADDING

説 明　セクションサイズが、セクションのアライメントの倍数となるように、セクション終端にデータを埋め込みます。

例　−start=P,C/0 −padding
　P セクションのアライメント:4 バイト
　P セクションのサイズ:0x06 バイト
　C セクションのアライメント:1 バイト
　C セクションのサイズ:0x03 バイト
　の場合、
　P セクションに 2 バイトのパディングデータを埋め込んで、サイズを 0x08 バイトにしてリンクする。
−start=P/0,C/7 −padding
　P セクションのアライメント:4 バイト
　P セクションのサイズ:0x06 バイト
　C セクションのアライメント:1 バイト
　C セクションのサイズ:0x03 バイト
　の場合、
　P セクションに 2 バイトのパディングデータを埋め込んで、サイズを 0x08 バイトにしてリンクすると、
　C セクションと重複してしまうため、L2321 エラーを出力する。

備 考　生成するパディングデータの値は 0x00 です。
絶対アドレスセクションには、パディングを行いませんので、絶対アドレスセクションはユーザにてサイズを調整してください。
マイコン種別が SuperH ファミリおよび RX ファミリのときに有効です。

特定ベクタ番号のアドレス設定

VECTN

リンカ<出力>[オプション項目 :] [特定ベクタ]

書 式　VECTN = <サブオプション>[,...]
<サブオプション> : <ベクタ番号> = {<シンボル> | <アドレス>}

説 明　可変ベクタテーブル(vector セクション)の特定ベクタ番号に対して、オプションで指定されたアドレスを設定します。
本オプションを使用した場合、ソース上に割り込み関数記述がなくても、可変ベクタテーブルを vector セクションとして作成し、テーブルへアドレスを設定します。

<ベクタ番号>は、10 進数で 0～255 の範囲で指定してください。
<シンボル>は、対象関数の外部名で指定してください。
<アドレス>は、指定アドレスを 16 進数で指定してください。

例　−vectn=30=_f1,31=0000F100　; ベクタ番号 30 番に_f1 のアドレスを、
　; ベクタ番号 31 番に 0x0f100 を設定します

備 考　マイコン種別が RX ファミリ、M16C シリーズ、R8C ファミリの場合に有効です。
ユーザが vector セクションをソースプログラムで作成している場合、可変ベクタテーブルの自動生成は行なわないため、本オプションは無効になります。

空きベクタ領域のアドレス設定

VECT

リンカ<出力> [オプション項目：] [空きベクタ]

書 式　VECT={<シンボル>|<アドレス>}

説 明　可変ベクタテーブル(vector セクション)で、アドレス未設定のベクタ番号に対してオプション指定のアドレスを設定します。
本オプションを使用した場合、ソース上の割り込み関数記述がなくても、可変ベクタテーブルを vector セクションとしてリンカが作成し、テーブルへアドレスを設定します。
<シンボル>は、対象関数の外部名を記述してください。
<アドレス>は、設定するアドレスを 16 進数表記で記述してください。

備 考　マイコン種別が RX ファミリ、M16C シリーズ、R8C ファミリの場合に有効です。
ユーザが vector セクションをソースプログラムで作成している場合、可変ベクタテーブルの自動生成は行なわないため、本オプションは無効になります。
{<シンボル>|<アドレス>}の記述で、先頭を 0 と記述したものは全てアドレスとして判断します。

Utl30 向け情報ファイル出力

UTL

リンカ<その他>[その他のオプション]utl ファイル出力

書 式　UTL

説 明　コンパイラパッケージ付属のツール(utl30)に入力するための外部ファイル(utl ファイル)を生成します。
生成するファイルの名称は「<出力ファイル名>.utl」となります。

例　tp.obj
utl
output=test.abs
tp.obj 内のインスペクタ情報を test.utl に出力します。

備 考　本オプションは、M16C マイコン向けのコンパイラを使用した場合のみ有効です。
本オプションは、abs ファイル入力時の処理には使用できません。
form={object | library} 指定時、本オプションは無効です。

ジャンプテーブル出力

JUMP_ENTRIES_FOR_PIC

リンカ<出力>[ジャンプテーブル:]

書 式　JUMP_ENTRIES_FOR_PIC = <セクション名>[,...]

説 明　指定セクション内の外部定義シンボルへ分岐するジャンプテーブルのアセンブラソースを出力します。
ファイル名は、<出力ファイル>.jmp です。

例　jump_entries_for_pic=sct2,sct3
output=test.abs
セクション sct2,sct3 の外部定義シンボルへ分岐するジャンプテーブルを test.jmp に出力します。

[test.jmp の出力例]

```
;OPTIMIZING LINKAGE EDITOR GENERATED FILE 2009.07.19
                .glb _func01
                .glb _func02
                .SECTION      P,CODE
_func01:
                MOV.L         #1000H,R14
                JMP           R14
_func02:
                MOV.L         #2000H,R14
                JMP           R14
                .END
```

備 考　form={object | relocate | library}または strip 指定時、本オプションは無効です。
マイコン種別が RX 系以外の場合は、本オプションは無効です。
生成するジャンプテーブルは、P セクションへ出力します。
セクション名に指定できるセクション種別は、プログラムセクションのみです。

6.2.3 リストオプション

表 6.5 リストカテゴリオプション一覧

	項目	コマンドライン形式	ダイアログメニュー	指定内容
1	リストファイル	LISt [= <ファイル名>]	リンカ <リスト> [リンケージリスト出力]	リストファイル出力を指定
2	リスト内容	SHow [= <sub>[,...]] <sub> : {SYmbol| Reference| SEction| Xreference| Total_size| VECTOR| ALL }	リンカ <リスト> [リスト内容 :]	 シンボル情報 参照回数 セクション情報 クロスリファレンス情報 合計セクションサイズ ベクタ情報出力 全情報出力

リストファイル

LISt

リンカ <リスト> [リンケージリスト出力]

書 式　LISt [= <ファイル名>]

説 明　リストファイル出力およびリストファイル名を指定します。
リストファイル名を指定しない場合には、出力(または先頭出力)ファイルと同じファイル名で、拡張子が form=library または extract 指定時「lbp」、それ以外のとき「map」のリストファイルが作成されます。

リスト内容

SHow

リンカ <リスト> [リスト内容 :]

書 式　　SHow[= <sub>[,…]]
　　<sub> : { SYmbol | Reference | SEction | Xreference | Total_size
　　|VECTOR | ALL }

説 明　　リストの出力内容を指定します。
サブオプションの一覧を表 6.6 に示します。
各リストの具体例については「8.6 リンケージリスト」、「8.8 ライブラリリスト」を参照してください。

表 6.6 show オプションのサブオプション一覧

	出力形式	サブオプション名	意味
1	form=library または extract 指定時	symbol	モジュール内シンボル名一覧を出力します。(extract 指定時)
		reference	指定できません。
		section	モジュール内セクション一覧を出力します。(extract 指定時)
		xreference	指定できません。
		total_size	指定できません。
		vector	指定できません。
		all	指定できません(extract 指定時)。 モジュール内シンボル名、セクション一覧を出力します(form=library 指定時)。
2	form=library 以外 かつ extract 指定なし時	symbol	シンボルアドレス、サイズ、種別、最適化内容を出力します。
		reference	シンボルの参照回数を出力します。
		section	指定できません。
		xreference	クロスリファレンス情報を出力します。
		total_size	ROM 配置対象、RAM 配置対象ごとに、セクションの合計サイズを表示します。
		vector	ベクタ情報を出力します。
		all	show=symbol,xreferrence,total_size 指定時と同内容を出力します。(form=rel) show=symbol,total_sizel 指定時と同内容を出力します。(form=rel,data_stuff) show=symbol,reference,xreference ,total_size 指定時と同内容を出力します。(form=abs) show=symbol,reference,xreference,total_size 指定時と同内容を出力します。(form=hex/stype/bin) form=obj のときは指定できません。

備　考　　オプション form とオプション show および show=all で有効/無効になる組み合わせは以下のようになります。

		Symbol	Reference	Section	Xreference	Vector	Total_size
form=abs	show のみ	有効	有効	無効	無効	無効	無効
	show=all	有効	有効	無効	有効	有効	有効
form=lib	show のみ	有効	無効	有効	無効	無効	無効
	show=all	有効	無効	有効	無効	無効	無効
form=rel	show のみ	有効	無効	無効	無効	無効	無効
	show=all	有効	無効	無効	有効*1	無効	有効
form=obj	show のみ	有効	有効	無効	無効	無効	無効
	show=all	無効	無効	無効	無効	無効	無効
form=hex/bin/sty	show のみ	有効	有効	無効	無効	無効	無効
	show=all	有効	有効	無効	有効	有効*1	有効*1

*1　入力ファイルが absolute 形式の場合は無効です。

クロスリファレンス情報の出力に関しては、下記制限があります。

- 出力ファイルがrelocatable形式で、かつdata_stuffオプションを使用している場合、クロスリファレンス情報は出力できません。
- 入力ファイルがabsolute形式の場合、参照側アドレスの情報は出力されません。
- アセンブル時にgoptimizeオプションが指定されていない場合、同一ファイル内への分岐に関する情報は出力されません。(マイコンがH8,H8S,H8SXファミリの場合のみ)
- 同一ファイル内の、定数シンボルへの参照に関する情報は出力されません。
- コンパイル時に最適化が有効で、直下の関数を呼び出す場合についての情報は出力されません。
- 外部変数アクセス最適化が有効な場合、ベースとなるシンボルを除いて、変数の参照情報は出力されません。(マイコンがSuperHファミリおよび RXファミリの場合のみ)
- ソースファイル上で#pragma tbrを記述した際にオフセット値を直接指定している場合、当該関数についての情報は出力されません。(マイコンがSH-2A/SH2A-FPUの場合のみ)
- リンク時の最適化を指定した場合、定数やリテラルの統合が生じると、その定数やリテラルに関する参照情報は出力されません。
- show=total_sizeで表示する情報は、別オプションtotal_sizeでの表示内容と同じです。
- show=vectorは、マイコン種別がRXファミリ、M16Cシリーズ、R8Cファミリのとき使用できます。
- show=referenceが有効な場合に、#pragma addressで指定された変数の参照回数が0として出力されます。(マイコンがSuperHファミリおよび RXファミリの場合のみ)

6.2.4 最適化オプション

表 6.7 最適化カテゴリオプション一覧

	項目	コマンドライン形式	ダイアログメニュー	指定内容
1	最適化	OPtimize [= <sub>[,...]]	リンカ<最適化>	最適化あり
		<sub> : { STring_unify	[最適化方法 :]	定数/文字列の統合
		| SYmbol_delete	[最適化設定]	未参照シンボルの削除
		| Variable_access	[設定 :]	短絶対アドレッシングモード活用
		| Register		レジスタ退避/回復の最適化
		| SAMe_code		共通コードの統合
		| SHort_format		アドレッシングモードの短縮
		| Function_call		間接アドレッシングモード活用
		| Branch		分岐命令の最適化
		| SPeed		実行速度優先の最適化
		| SAFe }		安全な最適化
		NOOPtimize		最適化なし
2	共通コード サイズ	SAMESize = <サイズ> (省略時:sames=1e)	リンカ<最適化> [統合サイズ :]	共通コード統合の対象となる最小サイズの指定
3	プロファイル 情報	PROfile = <ファイル名>	リンカ<最適化> [プロファイル情報 :]	プロファイル情報ファイルの指定 (動的最適化を行います)
4	キャッシュ サイズ	CAchesize = <sub> <sub>: Size = <サイズ>| Align = <ラインサイズ> (省略時:ca=s=8,a=20)	リンカ<最適化> [キャッシュサイズ :]	 キャッシュサイズの指定 キャッシュラインサイズの指定 (SuperH ファミリ向け)
5	最適化 部分抑止	SYmbol_forbid = <シンボル名>[,...] SAMECode_forbid = <関数名>[,...] Variable_forbid = <シンボル名>[,...] FUnction_forbid = <関数名>[,...] SEction_forbid = <sub>[,...] <sub> : [<ファイル名>| <モジュール名>] (<セクション名>[,...]) Absolute_forbid = <アドレス> [+ <サイズ>] [,...]	リンカ<最適化> [最適化方法 :] [最適化部分抑止]	未参照シンボル削除抑止シンボル 共通コード統合抑止シンボル 短絶対アドレッシングモード活用抑止シンボル 間接アドレッシングモード活用 抑止シンボル 最適化抑止セクション 最適化抑止アドレス範囲

最適化

OPtimize
NOOPtimize

リンカ<最適化>[最適化方法 :][最適化設定][設定 :]

書 式　OPtimize[= 〈サブオプション〉[,…]]
NOOPtimize
〈サブオプション〉:{STring_unify | SYmbol_delete | Variable_access
| Register | SAMe_code | SHort_format
| Function_call | Branch | SPeed | SAFe}

説 明　モジュール間最適化実行有無を指定します。
optimize オプション指定時は、コンパイル、アセンブル時に goptimize オプションを指定したファイルに対して最適化を行います。
nooptimize オプション指定時は、モジュールの最適化を行いません。
本オプションの省略時解釈は、optimize です。サブオプションの一覧を表 6.8 に示します。

表 6.8　optimize オプションのサブオプション一覧

サブオプション	意　味	最適化対象プログラム*1							
		SHC	SHA	H8C	H8A	RXC	RXA	NCC	NCA
パラメータなし	全ての最適化を実行します。	○	×	○	○	○	×	○	×
string_unify	const 属性を持つ定数に対し、同一値定数を統合します。const 属性を持つ定数には次のものが含まれます。 ・C/C++プログラム中の const 修飾型変数 ・文字列データの初期値/リテラル定数	○	×	○	×	×	×	×	×
symbol_delete	1 度も参照のない変数/関数を削除します。 必ずコンパイル時に#pragma entry を指定するか、optlnk で entry オプションを指定してください。	○	×	○	×	○	×	○	×
variable_access	8/16 ビット絶対アドレッシングモードでアクセス可能な領域にアクセス回数の多い変数を割り当てます。 必ずコンパイル、アセンブル時に cpu オプションを指定してください。	×	×	○	○	×	×	×	×
register	関数の呼び出し関係を解析し、レジスタの再割付および冗長なレジスタ退避/回復コードを削除します。 必ずコンパイル時に#pragma entry を指定するか、optlnk で entry オプションを指定してください。	○	×	○	×	×	×	×	×
same_code	複数の同一命令列をサブルーチン化します。	○	×	○	×	○	×	×	×
short_format	ディスプレースメント/イミディエートのコードサイズを短縮可能な場合、コードサイズがより小さくなる命令に置き換えます。	×	×	○	○	○	×	×	×

サブオプション	意　味	最適化対象プログラム*1							
		SHC	SHA	H8C	H8A	RXC	RXA	NCC	NCA
function_call	0～0xFF の範囲に空きがあれば、アクセス回数の多い関数のアドレスを割り当てます。また、マイコン種別が H8SX ファミリの場合には、下記領域も使用されます。 H8SXN : 0x100～0x1FF H8SXM,H8SXA,H8SXX : 0x200～0x3FF 必ずコンパイル、アセンブル時に cpu オプションを指定してください。	×	×	○	○	×	×	×	×
branch	プログラムの配置情報に基づいて、分岐命令サイズを最適化します。他の最適化項目を実行すると、指定の有無に関わらず必ず実行します。	○	×	○	○	○	×	○	×
speed	オブジェクトスピード低下を招く可能性のある最適化以外を実行します。 optimize=string_unify, symbol_delete, varaible_access, register, short_format, branch と同じです。	○	×	○	○	○*2	×	○*2	×
safe	変数や関数の属性によって制限される可能性のある最適化以外を実行します。 optimize=string_unify, register, short_format, branch と同じです。	○	×	○	○	○*4	×	○*3	×

【注】*1 SHC: SuperH ファミリ用 C/C++プログラム、SHA: SuperH ファミリ用アセンブリプログラム
H8C: H8,H8S,H8SX ファミリ用 C/C++プログラム、H8A: H8,H8S,H8SX ファミリ用アセンブリプログラム
RXC: RX ファミリ用 C/C++プログラム、RXA: RX ファミリ用アセンブリプログラム
NCC: M16C シリーズ、R8C ファミリ用 C/C++プログラム、NCA: M16C シリーズ、R8C ファミリ用アセンブリプログラム
*2 symbol_delete, branch, short_format が有効になります。
*3 branch が有効になります。
*4 short_format, branch が有効になります。

備　考　form={object | relocate | library}または strip 指定時、本オプションは無効です。
コンパイル時に外部変数アクセス最適化を指定した場合、定数/リテラル統合最適化(optimize=string_unify)は無効になります。
optimize=short_format 指定は、マイコン種別が H8SX ファミリ, RX ファミリの場合にのみ有効です。
マイコン種別が SH-2A/SH2A-FPU の場合、optimize=register の機能によってコードサイズが増加する場合があります。
プログラム内に#pragma entry で実行開始関数を指定、または実行開始アドレス(entry)を指定してしない場合、optimize=symbol_delete は無効になります。

共通コードサイズ

SAMESize

リンカ<最適化>[統合サイズ :]

書 式　SAMESize = 〈サイズ〉

説 明　共通コード統合最適化(optimize=same_code)で、最適化対象となる最小コードサイズを指定します。8～7FFF までの 16 進数で指定してください。
本オプションの省略時解釈は、samesize=1E です。

備 考　optimize=same_code の指定がないとき、本オプションは無効です。

プロファイル情報

PROfile

リンカ<最適化>[プロファイル情報 :]

書 式　PROfile = 〈ファイル名〉

説 明　プロファイル情報ファイルを指定します。
プロファイル情報ファイルとして指定できるのは、ルネサス統合開発環境 Ver. 2.0 以降が出力するプロファイル情報ファイルだけです。
プロファイル情報ファイルを指定すると、モジュール間最適化で動的情報に基づいた最適化を実行できます。
プロファイル情報入力により影響がある最適化を表 6.9 に示します。

表 6.9 プロファイル情報と最適化の関係

サブオプション	意 味	最適化対象プログラム*1			
		SHC	SHA	H8C	H8A
variable_access	動的アクセス回数の多い変数を優先的に割り当てます。	×	×	○	○
function_call	動的アクセス回数の多い関数の最適化優先順位を下げます。	×	×	○	○
branch	動的に呼び出し回数の多い関数を呼び出し元の関数の近くに配置します。 SuperH ファミリ用プログラムの場合は、cachesize オプションで指定するキャッシュサイズを意識した配置最適化を行います。	○	△ *2	○	△

【注】*1 SHC: SuperH ファミリ用 C/C++プログラム、SHA: SuperH ファミリ用アセンブリプログラム、
H8C: H8,H8S,H8SX ファミリ用 C/C++プログラム、H8A: H8,H8S,H8SX ファミリ用アセンブリプログラム
*2 関数単位の移動は行いませんが、入力ファイル単位の移動は実行します。

備 考　optimize 指定がないとき、本オプションは無効です。

キャッシュサイズ

CAchesize

リンカ<最適化>[キャッシュサイズ :]

書 式　CAchesize = 〈sub〉
〈sub〉:Size = 〈サイズ〉 | Align = 〈ラインサイズ〉

説 明　キャッシュサイズおよびキャッシュラインサイズを指定します。
profile オプション指定時、分岐命令最適化(optimize=branch)で使用します。
サイズはキロバイト単位、ラインサイズはバイト単位の 16 進数で指定してください。
本オプションの省略時解釈は、cachesize=size=8,align=20 です。

備 考　profile 指定がないとき、本オプションは無効です。

最適化部分抑止

SYmbol_forbid
SAMECode_forbid
Variable_forbid
FUnction_forbid
SEction_forbid
Absolute_forbid

リンカ<最適化>[最適化方法 :][最適化部分抑止]

書 式　SYmbol_forbid = 〈シンボル名〉[,…]
SAMECode_forbid = 〈関数名〉[,…]
Variable_forbid = 〈シンボル名〉[,…]
FUnction_forbid = 〈関数名〉[,…]
SEction_forbid = 〈sub〉[,…]
〈sub〉: [〈ファイル名〉|〈モジュール名〉](〈セクション名〉[,…])
Absolute_forbid = 〈アドレス〉[+ 〈サイズ〉][,…]

説 明　特定のシンボル、セクション、アドレス範囲の最適化を抑止します。アドレス、サイズは 16 進数で指定してください。C/C++変数名、C 関数名はプログラム中での定義名先頭に_を付加します。C++関数の場合は、引数列を含めたプログラム中の定義名をダブルクォーテーションで囲んで指定します。但し引数が void の場合は、"関数名()"で指定します。各オプションの意味を表 6.10 に示します。

表 6.10 最適化部分抑止オプション一覧

オプション	パラメータ	意味
symbol_forbid	関数名|変数名	未参照シンボル削除最適化を抑止します。
samecode_forbid	関数名	共通コード統合最適化を抑止します。
variable_forbid	変数名	短絶対アドレッシングモード活用最適化を抑止します。
function_forbid	関数名	間接アドレッシングモード活用最適化を抑止します。
section_forbid	セクション名 ファイル名 モジュール名	特定セクションの最適化を抑止します。入力ファイル名、もしくは ライブラリモジュール名を同時に指定することで、最適化抑止対象を セクション全体だけでなく、特定ファイルに限定することも可能です。
absolute_forbid	アドレス[+サイズ]	アドレス＋サイズの範囲の最適化を抑止します。

例　　symbol_forbid="f(int)"　; C++関数 f(int) は参照回数 0 でも削除しません。
section_forbid=(P1)　　; P1 セクションへの全最適化を抑止します。
section_forbid=a.obj(P1,P2)
; a.obj 内の P1,P2 セクションへの全最適化を抑止します。

備　考　　最適化を使用しないリンク処理では、本オプションは無効です。
パスを記述した入力ファイルを最適化抑止する場合、section_forbid オプションではファイル名にパスを記述してください。

6.2.5 セクションオプション

表 6.11 セクションカテゴリオプション一覧

	項目	コマンドライン形式	ダイアログメニュー	指定内容
1	セクションアドレス	STARt= <sub>[,...] <sub> : [(]<セクション名> [[: | ,] <セクション名 >[,...]] [)][,...] [/<アドレス>]	リンカ <セクション> [設定項目 :] [セクション]	セクションの開始アドレス指定
2	シンボルアドレス ファイル	FSymbol = <セクション名>[,...]	リンカ <セクション> [設定項目 :] [シンボルアドレスファイル]	外部定義シンボルアドレスの定義ファイル出力

セクションアドレス

STARt

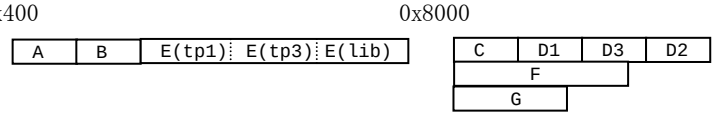
リンカ <セクション> [設定項目 :] [セクション]

書 式　STARt = <sub> [,…]
<sub>: [(]<セクション名>[[:|,] <セクション名>[,…]][)][,…][/<アドレス>]

説 明　セクションの開始アドレスを指定します。アドレスは 16 進数で指定してください。
セクション名はワイルドカード"*"も指定できます。ワイルドカードで指定したセクションは入力順に展開します。
セクションをコロン":"で区切ることにより、複数のセクションを同一アドレスに割り付ける(セクションオーバーレイ配置)ことが可能です。
同一アドレスに割り付け指定したセクション間は、指定順に割り付けます。
また、丸括弧"()"で囲むことにより、オーバーレイ配置する対象セクションを変更できます。
同一セクション内オブジェクトは、入力ファイルの指定順、入力ライブラリの指定順に割り付けます。
アドレスの指定がない場合は、0 番地から割り付けます。
start オプションで指定していないセクションは、最終割り付けアドレスに続いて割り付けます。

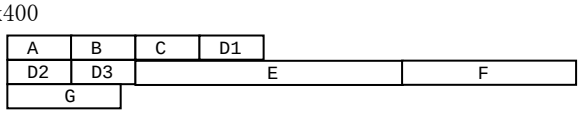
例　　下記順番でオブジェクトを入力する場合のセクション配置を例に示します。
(括弧内は各オブジェクトが持つセクション)
tp1.obj(A,D1,E) -> tp2.obj(B,D3,F) -> tp3.obj(C,D2,E,G) -> lib.lib(E)

(1) -start=A,B,E/400,C,D*:F:G/8000



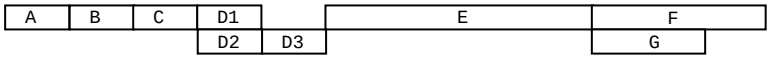
":"で区切った C,F,G セクションは、同一アドレスに割りつきます。
ワイルドカードで記述したセクション(ここでは D で始まる名前のセクション)は、
入力した順番で割りつきます。
同名セクション内(ここでは E セクション)は、入力したオブジェクトから順番に割りつきます。
ライブラリ入力による同名セクション(ここでは E セクション)は、入力オブジェクトの次に割りつきます。

(2) -start=A,B,C,D1:D2,D3,E,F:G/400



":"で区切った直後のセクション(この例の場合は A,D2,G)を先頭として、それぞれ先頭が同一アドレスに割りつきます。

(3) -start=A,B,C,(D1:D2,D3),E,(F:G)/400



"()"で同一アドレス配置を括った場合、"()"の直前のセクション(この例の場合は C,E)の直後を先頭として、"()"内の同一アドレス配置が行われます。
"()"の直後のセクション(この場合 E)は、"()"内の最後尾のセクションの直後に続けて配置されます。

備　考　　form={object | relocate | library}または strip 指定時、本オプションは無効です。
括弧"()"は、ネストして記述することはできません。
括弧"()"内では、少なくともひとつはコロン":"の記述が必要です。コロン":"を記述しない場合には、括弧"()"は記述できません。
括弧"()"を記述した場合、"()"外にコロン":"を記述することはできません。
括弧"()"を使用して本オプションを記述した場合、リンカの最適化機能は無効になります。

シンボルアドレスファイル

FSymbol

リンカ <セクション> [設定項目 :] [シンボルアドレスファイル]

書 式　FSymbol = 〈セクション名〉[,…]

説 明　指定したセクション内外部定義シンボルをアセンブラ制御命令形式でファイルに出力します。
ファイル名は、〈出力ファイル〉.fsy です。

例　fsymbol=sct2,sct3
output=test.abs
セクション sct2,sct3 の外部定義シンボルを test.fsy に出力します。

[test.fsy の出力例]

```
;OPTIMIZING LINKAGE EDITOR GENERATED FILE 1999.11.26
;fsymbol = sct2, sct3

;SECTION NAME = sct2
 .export _f
_f: .equ h'00000000
 .export _g
_g: .equ h'00000016
;SECTION NAME = sct3
 .export _main
_main: .equ h'00000020
 .end
```

備 考　form={object | relocate | library}または strip 指定時、本オプションは無効です。
マイコン種別が H8,H8S,H8SX ファミリ, SuperH ファミリ, RX ファミリのときに使用できます。

6.2.6 ベリファイオプション

表 6.12 ベリファイカテゴリオプション一覧

	項目	コマンドライン形式	ダイアログメニュー	指定内容
1	アドレス整合性のチェック	CPu = { <cpu 情報ファイル名> | <メモリ種別> = <アドレス範囲>[,...] | STRIDE } <メモリ種別> = { ROm | RAm | XROm | XRAm | YROm | YRAm | FIX} <アドレス範囲>: <先頭アドレス> - <終了アドレス>	リンカ <ベリファイ> [アドレス整合チェック :]	セクションアドレスの割り付け可能範囲を指定セクション名をセクション分割の対象に指定
2	物理空間上の重複チェック	PS_check=<sub>[:<sub>...] <sub>: <LS>,<LS>[,...] <LS>: <開始アドレス> -<終端アドレス>	リンカ <ベリファイ> [物理空間上の重複チェック :]	物理空間上で重なり合うアドレス範囲を指定
3	セクション分割対象外指定	CONTIGUOUS_SECTION = <セクション名>[,…]	リンカ <ベリファイ> [分割対象がセクション :]	セクション名をセクション分割の対象外セクションに指定

アドレス整合性のチェック

CPu

リンカ <ベリファイ> [アドレス整合チェック :]

書 式　CPu = { <cpu 情報ファイル名>
　　| <メモリ種別> = <アドレス範囲>[,…]
　　| STRIDE}
＜メモリ種別＞ = { ROm | RAm | XROm | XRAm | YROm | YRAm | FIX }
＜アドレス範囲＞ : <先頭アドレス> - <終了アドレス>

説 明　cpu=stride 未指定時は、セクションの割り付けアドレスに対して、アドレス範囲に入らない場合は、エラーを出力します。
cpu=stride 指定時は、セクションの割り付けアドレスに対して、アドレス範囲に入らない場合は、次の同メモリ種別に配置、または、分割して配置します。

[例]サブオプション stride を指定しない場合
start=D1,D2/100
cpu=ROM=100-1FF,RAM=200-2FF
D1 が 100-1FF、D2 が 200-2FF の範囲に収まるとき、正常終了します。収まらないときエラーを出力します。
[例]サブオプション stride を指定した場合
start=D1,D2/100
cpu=ROM=100-1FF,RAM=200-2FF,ROM=300-3FF
cpu=stride
D1,D2 が ROM 属性の領域に(セクションを分割して/分割しないで)収まるとき、正常終了します。セクションを分割しても収まらないときリンクエラーになります。

xrom/xram は DSP の X メモリ領域、yrom/yram は DSP の Y メモリ領域を指定します。
セクション割り付けが可能なアドレス範囲を 16 進数で指定してください。ROM/RAM の属性は、モジュール間最適化で使用します。
メモリ種別"FIX"には、アドレス固定の領域(I/O エリア等)を指定します。
メモリ種別"FIX"と、それ以外のメモリ種別のアドレス範囲が重複した場合は、メモリ種別"FIX"を有効とします。

サブオプション stride は、メモリ種別が、ROM または RAM で、アドレス範囲にセクションが収まらなかった場合に、セクションを分割して同じメモリ種別の領域に割り付けます。
サブオプション stride で、セクションを分割する単位は、モジュール単位になります。
[例]
cpu=ROM=0-FFFF,RAM=10000-1FFFF
セクションアドレスが、0-FFFF または 10000-1FFFF の間に入っているかチェックします。
モジュール間最適化では、異なる属性間でのオブジェクトの移動は行いません。

cpu=ROM=100-1FF,ROM=400-4FF,RAM=500-5FF
cpu=stride
セクションアドレスが、100-1FF の間に収まらなかった場合に、セクションをモジュール単位で分割して 400-4FF に割り付けます

備 考　form={object | relocate | library}または strip 指定時、本オプションは無効です。
cpu=stride および memory=low 指定時、無効になります。
マイコン種別が SH2DSP,SH3DSP,SH4ALDSP 以外の場合は、メモリ種別が xrom,xram,yrom,yram の指定は無効となります。
cpu=stride および optimize=register が有効な場合、L2320 エラーが出力されることがあります。その場合には、optimize=register を無効にしてください。
cpu=stride を指定し、B セクションが分割された場合、0 初期化するための情報として 8 バイト×分割数分だけ C$BSEC セクションのサイズが増加します。

物理空間上の重複チェック

PS_check

リンカ <ベリファイ> [物理空間上の重複チェック :]

書 式　PS_check=<sub>[:<sub>...]
<sub>: <LS>,<LS>[,...]
<LS>: <開始アドレス>-<終端アドレス>

説 明　アドレス値では重なっていないが、実際にメモリ上に配置すると重なってしまうオブジェクトを検出するためのオプションです。
本オプションを使用することにより、SH3 や SH4 などで、論理アドレス上では重ならないが実メモリ上に配置する際に重なってしまうオブジェクトを検出することが可能です。
本オプションによって重複を検出した場合、エラーとしてリンク処理を終了します。
メモリ上で重なり合うアドレス範囲(書式の中の<LS>)をオプションに記述してください。
複数の物理メモリに対してチェックしたい場合には、':'で区切って記述する
ことでチェック可能です。

例　SH4 は、MMU が無効状態の場合、4G バイトのアドレス空間は、512M バイト(29bit)の外部メモリ空間へマッピングします(4G バイトアドレスの上位 3bit を無視してマッピングします)。
たとえば、ユーザモードで使用可能な U0 領域(00000000～0x7fffffff)に対して、外部メモリ(512M)にマッピングする場合のオブジェクトの重なりは、下記記述で検出可能です。

-PS_check=00000000-1fffffff,20000000-3fffffff,40000000-5fffffff,60000000-7fffffff

本オプション記述により、00000000,20000000,40000000,60000000 番地はすべて、実メモリ上では同じ場所に配置されることを表します。

備 考　本オプションは、SuperH ファミリのマイコンに対してのみ有効です。
出力形式(form オプション)が object,relocate,library の場合、本オプションは無効です。
absolute ファイルを入力する場合の処理は、本オプションは無効です。
マイコンのアドレス空間の仕様については、各マイコンのハードウェアマニュアルを参照してください。

セクション分割対象外指定

CONTIGUOUS_SECTION

リンカ<ベリファイ>[分割対象外セクション :]

書 式　CONTIGUOUS_SECTION=<セクション名>[,...]

説 明　cpu=stride が有効なときに、セクションを分割せずに同じメモリ種別の割り付け可能なアドレス領域に割り付けるセクションを指定します。

[例]
start=P,PA,PB/100
cpu=ROM=100-1FF,ROM=300-3FF,ROM=500-5FF
cpu=stride
contiguous_section=PA

セクション P を 100 番地に割り付けます。
contiguous_section 指定したセクション PA が、1FF 番地までに割り付けることができない場合、セクション PA を分割せずに、300 番地から割り付けます。
contiguous_section 指定してないセクション PB が、3FF 番地までに割り付けることができない場合、セクション PB を分割して、500 番地から割り付けます。

備 考　cpu オプションのサブオプションの stride が無効なとき、本オプションは無効です。

6.2.7 その他オプション

表 6.13 その他カテゴリオプション一覧

	項目	コマンドライン形式	ダイアログメニュー	指定内容
1	終端コード	S9	リンカ <その他> [その他のオプション :] [S9 レコードを終端に出力]	S9 レコードを常に出力
2	スタック情報 ファイル	STACk	リンカ <その他> [その他のオプション :] [スタック情報ファイル(sni)出力]	スタック使用量情報ファイル出力
3	デバッグ 情報圧縮	COmpress NOCOmpress	リンカ <その他> [その他のオプション :] [デバッグ情報圧縮]	デバッグ情報を圧縮する デバッグ情報を圧縮しない
4	メモリ使用量 削減指定	MEMory = [High | Low]	リンカ <その他> [その他のオプション :] [入力ファイルロード時のメモリ 使用量削減]	入力ファイルをロードする際の メモリ使用量指定
5	シンボル名変更	RENname = <sub>[,...] <sub> : { [<ファイル名>] (<名前>=<名前>[,...]) | [<モジュール名>] (<名前>=<名前>[,...]) }	リンカ <その他> [ユーザ指定オプション :]	シンボル名、セクション名の変更
6	シンボル名削除	DELete = <sub>[,...] <sub> : { <モジュール名> | [<ファイル名>] (<名前>[,...]) }	リンカ <その他> [ユーザ指定オプション :]	シンボル名、モジュール名の削除
7	モジュールの 置き換え	REPlace = <sub>[,...] <sub> : <ファイル> [(<モジュール>[,...])]	リンカ <その他> [ユーザ指定オプション :]	ライブラリファイル内同名 モジュールの置き換え
8	モジュールの 抽出	EXTract = <モジュール>[,...]	リンカ <その他> [ユーザ指定オプション :]	ライブラリファイル内指定 モジュールの抽出
9	デバッグ 情報削除	STRip	リンカ <その他> [ユーザ指定オプション :]	アブソリュートファイル、 ライブラリファイルの デバッグ情報削除

	項目	コマンドライン形式	ダイアログメニュー	指定内容
10	メッセージレベル	CHange_message = <sub>[,...] <sub>: {Information | Warning | Error} [=<エラー番号> [-<エラー番号>] [,...]]	リンカ <その他> [ユーザ指定オプション :]	メッセージレベルの変更
11	ローカルシンボル名秘匿指定	Hide	リンカ <その他> [ユーザ指定オプション :]	ローカルシンボル名情報を削除
12	合計セクションサイズの表示	Total_size	リンカ <その他> [ユーザ指定オプション :]	標準出力へ、リンク後の合計セクションサイズを表示できます。
13	エミュレータ向けの情報ファイル	RTs_file	リンカ<その他> [その他のオプション :] [関数出口情報ファイル(rts)出力]	エミュレータ向けの情報ファイルを出力します。 (SuperH ファミリ向け)

終端コード

S9

リンカ <その他>[その他のオプション :][S9 レコードを終端に出力]

書 式　S9

説 明　エントリアドレスが 0x10000 を超える場合でも、S9 レコードを終端に出力します。

備 考　form=stype 指定がないとき、本オプションは無効です。

スタック情報ファイル

STACk

リンカ <その他>[その他のオプション :][スタック情報ファイル(sni)出力]

書 式　STACk

説 明　スタック使用量情報ファイルを出力します。
ファイル名は、<出力ファイル名>.sni になります。

備 考　form={object | relocate | library}および strip 指定時、本オプションは無効です。

デバッグ情報圧縮

COmpress
NOCOmpress

リンカ <その他> [その他のオプション :] [デバッグ情報圧縮]

書 式　COmpress
NOCOmpress

説 明　デバッグ情報の圧縮有無を指定します。
compress オプションを指定した場合、デバッグ情報を圧縮します。
nocompress オプションを指定した場合、デバッグ情報を圧縮しません。
デバッグ情報を圧縮すると、デバッガのロード速度が速くなります。また、nocompress オプションを指定すると、リンク時間が短くなります。
本オプションの省略時解釈は、nocompress です。

備 考　form={object | relocate | library | hexadecimal | stype | binary}
または strip オプションを指定した場合、compress オプションは無効です。

メモリ使用量削減指定

MEMory

リンカ <その他> [その他のオプション :] [入力ファイルロード時のメモリ使用量削減]

書 式　MEMory ＝ [High | Low]

説 明　リンク時に使用するメモリ量を指定します。
memory=high オプションを指定した場合、従来通りの処理を行います。
memory=low オプションを指定した場合、リンク時に必要な情報のロードを細かく行うことにより、使用するメモリ量の削減を行います。ファイルアクセスの頻度が増えるため、メモリ使用量が実装メモリを超えない状況では memory=high オプション指定より処理が遅くなります。

大規模なプロジェクトをリンクした際、最適化リンケージエディタのメモリ使用量が稼動マシンの実装メモリ量を越えてしまい、動作が遅くなっているような場合には memory=low オプション指定をお試しください。

備 考　下記オプションを指定した場合、memory=low オプション指定は無効となります。
optimize, compress, delete, rename, map, stack, replace,
list と show[={reference | xreference}]を同時指定,
また、入力ファイルや出力ファイル形式によっても無効となる組み合わせがあります。詳細は、「6.2.2 出力オプション」の表 6.4 を参照してください。

シンボル名変更

REName

リンカ <その他> [ユーザ指定オプション :]

書 式　　REName = <サブオプション>[,…]
<サブオプション> :　{ [<ファイル>](<名前> = <名前>[,…])
　　| [<モジュール>](<名前> = <名前>[,…])}

説 明　　外部シンボル名、セクション名を変更します。
特定のファイルまたは特定のライブラリ内モジュールに含まれるシンボル名、セクション名を変更することもできます。
C/C++変数名の場合、プログラム中での定義名先頭に_を付加します。
関数名を変更した場合の動作は保証できません。
指定した名前がセクション、シンボルの両方に存在した場合、シンボル名を優先します。
同一ファイル名、モジュール名が複数存在する場合は、先に入力した方を優先します。

例　　rename=(_sym1=data)　;_sym1 を data に変更します。
rename=lib1(P=P1)　　;ライブラリモジュール lib1 内の P セクションを
　　;P1 セクションに変更します。

備 考　　extract または strip 指定時、本オプションは無効です。
form=absolute 指定時、入力されたライブラリのセクション名を変更することができません。

シンボル名削除

DELete

リンカ <その他> [ユーザ指定オプション :]

書 式　　DELete = <サブオプション>[,…]
<サブオプション> :　{ [<ファイル>](<名前>[,…])
　　| <モジュール> }

説 明　　外部シンボル名またはライブラリモジュールを削除します。
特定のファイルに含まれるシンボル名、モジュールを削除することもできます。
C/C++変数名、C 関数名はプログラム中での定義名先頭に_を付加します。C++関数の場合は、引数列を含めたプログラム中の定義名をダブルクォーテーションで囲んで指定します。但し引数が void の場合は、"関数名()"で指定します。同一ファイル名が複数存在する場合は、先に入力した方を優先します。
本オプションで、シンボル名削除を指定した場合、オブジェクトは削除されず、属性が内部シンボルに変更されます。

例　　delete=(_sym1)　　; 全ファイル中のシンボル名_sym1 を削除します。
delete=file1.obj(_sym2)　　; f ile1.obj 内のシンボル名_sym2 を削除します。

備 考　　extract または strip 指定時、本オプションは無効です。

モジュールの置き換え

REPlace

リンカ <その他> [ユーザ指定オプション :]

書 式　REPlace = <サブオプション>[,…]
<サブオプション> :　<ファイル名>[(<モジュール名>[,…])]

説 明　ライブラリモジュールを置換します。
指定したファイルまたはライブラリモジュールと library オプションで指定したライブラリ内同名モジュールを置換します。

例　replace=file1.obj　;モジュール file1 と file1.obj を置換します。
replace=lib1.lib(mdl1)　;モジュール mdl1 とライブラリファイル lib1.lib 内モジュール mdl1 を置換します。

備 考　form={object | relocate | absolute | hexadecimal | stype | binary}
および extract、strip 指定時、本オプションは無効です。

モジュールの抽出

EXTract

リンカ <その他> [ユーザ指定オプション :]

書 式　EXTract = <モジュール名>[,…]

説 明　ライブラリモジュールを抽出します。
指定したライブラリモジュールを library オプションで指定したライブラリファイルから抽出します。

例　extract=file1　;モジュール file1 を抽出します。

備 考　form={absolute | hexadecimal | stype | binary}および strip 指定時、本オプションは無効です。
form=library 指定時、モジュールを削除できます。
form={absolute|relocate|hexadecimal|stype|binary}指定時、外部シンボルを削除できます。

デバッグ情報削除

STRip

リンカ <その他> [ユーザ指定オプション :]

書 式　STRip

説 明　アブソリュートファイル、ライブラリファイルのデバッグ情報を削除します。
strip オプション指定時は、入力ファイルと出力ファイルは 1 対 1 対応になります。

例　input=file1.abs file2.abs file3.abs
strip
file1.abs, file2.abs, file3.abs のデバッグ情報を削除し、それぞれ file1.abs, file2.abs, file3.abs に出力します。
デバッグ情報削除前のファイルは、file1.abk, file2.abk, file3.abk にバックアップします。

備 考　form={object | relocate | hexadecimal | stype | binary}指定時、本オプションは無効です。

メッセージレベル

CHange_message

リンカ <その他> [ユーザ指定オプション :]

書 式　CHange_message = <サブオプション>[,…]
<サブオプション> : <エラーレベル>[=<エラー番号>[-<エラー番号>][,…]]
<エラーレベル>　: {Information | Warning | Error}

説 明　インフォメーション、ウォーニング、エラーレベルのメッセージレベルを変更します。
メッセージ出力時の実行継続/中断を変更できます。

例　change_message=warning=2310
L2310 をウォーニングレベルに変更し、L2310 出力時も処理を継続します。

change_message=error
全てのインフォメーション、ウォーニングメッセージをエラーレベルに変更します。
メッセージを一つでも出力すると、処理を中断します。

ローカルシンボル名秘匿指定

Hide

リンカ <その他> [ユーザ指定オプション :]

書 式　Hide

説 明　本オプションを指定した場合、出力ファイル内のローカルシンボル名情報を消去します。
ローカルシンボルに関する名前の情報が消去されますので、バイナリエディタなどでファイルを開いてもローカルシンボル名は確認できなくなります。生成されるファイルの動作への影響は一切ありません。
ローカルシンボル名を機密扱いにしたい場合などに本オプションを指定してください。

秘匿対象となるシンボルの種類を以下に挙げます。
- ソースファイル:static 型修飾子を指定した変数名、関数名など
- ソースファイル:goto 文のラベル名
- アセンブリソース:外部定義(参照)シンボル宣言していないシンボル名

※ エントリ関数名は秘匿対象になりません。

例　ソースファイルで本オプションの機能が有効となる記述の例を以下に示します。

```
int g1;
int g2=1;
const int g3=3;
static int s1;                  //<--- static 変数名は秘匿対象
static int s2=1;                //<--- static 変数名は秘匿対象
static const int s3=2;      //<--- static 変数名は秘匿対象

static int sub1()               //<--- static 関数名は秘匿対象
{
      static int s1;            //<--- static 変数名は秘匿対象
      int l1;

      s1 = l1; l1 = s1;
      return(l1);
}

int main()
{
      sub1();
      if (g1==1)
            goto L1;
      g2=2;
L1:                                     //<--- goto 文のラベル名は秘匿対象
      return(0);
}
```

備 考　本オプションは出力ファイル形式が absolute,relocate,library の場合のみ有効です。
コンパイル、アセンブル時に goptimize オプションを指定したファイルを入力する場合、出力ファイル形式が relocate,library の場合は本オプションを指定できません。
外部変数アクセス最適化を行う状況で本オプションを指定する場合は、一度目のリンク時には指定せず、二度目のリンク時にのみ本オプションを指定してください。
デバッグ情報内のシンボル名は、本オプションを指定しても削除されません。

合計セクションサイズの表示

Total_size

リンカ<その他>[その他のオプション :] [合計セクションサイズ画面表示]

書 式　Total_size

説 明　リンク後のセクションの合計サイズを、標準出力に表示するためのオプションです。
下記の 3 種類のセクションに分けて、合計サイズを表示します。
- 実行可能なプログラムセクション
- プログラムセクション以外の ROM 領域配置セクション
- RAM 領域配置セクション

本オプションを使用することにより、ROM,RAM に配置する合計のセクションサイズを容易に認識することができます。

備 考　リンケージリストへ合計サイズを表示するには、別途 show=total_size オプションを使用する必要があります。
ROM 化支援機能(rom オプション)対象のセクションの場合、転送元(ROM)と転送先(RAM)の両方で領域を使用するため、双方の合計サイズに対してセクションサイズを加算します。

エミュレータ向けの情報ファイル

RTs_file

リンカ<その他>[その他のオプション :] [関数出口情報ファイル(rts)出力]

書 式　-RTs_file

説 明　エミュレータで使用するための情報、関数出口情報ファイル(.rts ファイル)を生成するオプションです。
お使いのエミュレータのマニュアルに従って、本オプションを使用してください。エミュレータの機種によって使用できない場合があります。
関数出口情報ファイルは、「<出力するロードモジュール名>.rts」というファイル名で生成されます。例えば、output オプションで指定する出力ファイル名を「test.abs」とした場合、関数出口情報ファイルは「test.rts」というファイル名で生成されます。
関数出口情報ファイルはロードモジュールと同じディレクトリに作成されます。

備 考　form={object | relocate | library}指定時、本オプションは無効です。
アブソリュートファイルを入力する場合、本オプションは無効です。
エミュレータのマニュアルに従って本オプションを使用してください。エミュレータの機種によって使用できない場合があります。
本オプションは、マイコン種別が SuperH ファミリのとき使用できます。

6.2.8 サブコマンドファイルオプション

表 6.14 サブコマンドファイルカテゴリオプション一覧

	項目	コマンドライン形式	ダイアログメニュー	指定内容
1	サブコマンドファイル	SUbcommand = <ファイル名>	リンカ <サブコマンドファイル> [サブコマンドファイルを指定]	サブコマンドファイルによるオプション指定

サブコマンドファイル

SUbcommand

リンカ<サブコマンドファイル> [サブコマンドファイルを指定]

書 式　SUbcommand = <ファイル名>

説 明　オプションをサブコマンドファイルで指定します。
サブコマンドファイルの書式は以下の通りです。

<オプション> {=|△}[<サブオプション>[,…]][△&] [;<コメント>]

オプションとサブオプションの区切りは、=の代わりに空白も指定できます。
input オプションの場合は、サブオプション区切りに空白を指定できます。
サブコマンドファイル内では 1 オプション/行で指定します。
サブオプションを 1 行に記述できない場合は、&を用いて継続指定できます。
サブコマンドファイル中に subcommand オプションは指定できません。

例　コマンドライン指定：optlnk file1.obj -sub=test.sub file4.obj
サブコマンド指定　:input　file2.obj file3.obj ;ここはコメントです。
　　library　lib1.lib, &　　; 継続行を指定します。
　　lib2.lib
サブコマンドファイルで指定したオプション内容を、コマンドライン上のサブコマンド指定位置に展開し、実行します。
ファイルの入力順序は、file1.obj, file2.obj, file3.obj, file4.obj になります。

6.2.9 マイコンオプション

表 6.15 CPU タブオプション一覧

	項目	コマンドライン形式	ダイアログメニュー	指定内容
1	SBR アドレス指定	SBr = { <SBR アドレス> | User}	CPU [SBR 値 :]	8bit 絶対領域の開始アドレスを指定 (H8SX ファミリ向け)

8bit 絶対領域アドレス値指定

SBr

CPU [SBR 値 :]

書 式　SBr ＝ { <アドレス> | User }

説 明　SBR のアドレス値を指定します。
本オプションでアドレス値を指定することにより、abs8 領域を用いた最適化が可能になります。本オプションで user を指定した場合は、abs8 領域への最適化は抑止されます。

備 考　本オプションはマイコン種別が H8SX ファミリの場合にのみ有効です。
ソース内、あるいはツールのオプション指定などで、複数の SBR アドレスが指定された場合には、本オプションは指定の如何に関わらず user が指定されたものとして扱われます。

6.2.10 残りのオプション

表 6.16 残りのオプション一覧

	項目	コマンドライン形式	ダイアログメニュー	指定内容
1	コピーライト	LOgo	–	出力あり
		NOLOgo		出力なし
2	継続指定	END	–	既入力オプション列を実行し、処理終了後は以降のオプション列を入力し、処理を継続
3	終了指定	EXIt	–	オプション入力の終了を指定

コピーライト

LOgo
NOLOgo

なし（常に nologo が有効）

書 式　LOgo
NOLOgo

説 明　コピーライトの出力有無を指定します。
logo オプション指定時はコピーライト表示を出力します。
nologo オプション指定時はコピーライト表示出力を抑止します。
本オプションの省略時解釈は、logo です。

継続処理

END

なし

書 式　END

説 明　END より前に指定したオプション列を実行します。リンケージ処理終了後、END 以降に指定したオプション列の入力、リンケージ処理を継続します。
本オプションは、コマンドライン上では指定できません。

例
```
input=a.obj,b.obj              ; 処理(1)
start=P,C,D/100,B/8000         ; 処理(2)
output=a.abs                   ; 処理(3)
end
input=a.abs                    ; 処理(4)
form=stype                     ; 処理(5)
output=a.mot                   ; 処理(6)
```

(1)～(3)の処理を実行し、a.abs を出力します。
その後、(4)～(6)の処理を実行し、a.mot を出力します。

終了処理

EXIt

なし

書 式　　EXIt

説 明　　オプション指定の終了を指定します。
本オプションは、コマンドライン上では指定できません。

例　　コマンドライン指定:　optlnk -sub=test.sub -nodebug

```
test.sub:       input=a.obj,b.obj                ; 処理(1)
                start=P,C,D/100,B/8000           ; 処理(2)
                output=a.abs                     ; 処理(3)
                exit
```

(1)～(3)の処理を実行し、a.abs を出力します。
Exit 実行後のコマンドライン指定の nodebug オプションは無効になります。

7. 環境変数

7.1 環境変数一覧

環境変数の一覧を表 7.1 に示します。

表 7.1 環境変数

	環境変数	説明	設定省略時の解釈
1	path	実行ファイルの格納フォルダを指定します。	省略不可
2	BIN30	コンパイラ、アセンブラおよび最適化リンケージエディタ等の実行ファイルの格納したフォルダを指定します。	省略不可
3	INC30	コンパイラとアセンブラの標準インクルードファイル格納フォルダを指定します。	省略不可
4	LIB30	コンパイラやアセンブラの標準ライブラリや内部ツールを格納したフォルダを指定します。	省略不可
5	TMP30	テンポラリファイルを作成するフォルダを指定します。	省略不可
6	HLNK_LIBRARY1 HLNK_LIBRARY2 HLNK_LIBRARY3	最適化リンケージエディタが使用するデフォルトライブラリ名を指定します。library オプションで指定したライブラリを優先してリンクします。その後未解決のシンボルがある場合、1、2、3 の順にデフォルトライブラリを検索します。	省略時、値は設定されません
7	HLNK_TMP	最適化リンケージエディタがテンポラリファイルを作成するフォルダを指定します。この環境変数の指定がない場合は、カレントフォルダにテンポラリファイルを作成します。	省略時、値は設定されません
8	HLNK_DIR	最適化リンケージエディタのライブラリ検索フォルダを指定します。	省略時、値は設定されません

- INC30、HLNK_LIBRARY1、HLNK_LIBRARY2、およびHLNK_LIBRARY3で複数ディレクトリを指定する場合は、";"(セミコロン)で区切ってください。
- フォルダの指定にはアクセス権のあるフォルダを指定してください。
- これらの環境変数は、インストール時に作成されるバッチファイルsetnc30.batを実行することで簡単に設定できます。（High-performance Embedded Workshopをご使用になる際は、上記バッチファイルを実行する必要はありません） setnc30.batは＜High-performance Embedded Workshop格納ディレクトリ＞¥Tools¥Renesas¥ nc30wa ¥<バージョン番号> に格納されています。

7.2 プリデファインドマクロ

オプション指定やバージョンに合わせて、以下のようなシンボル定数設定オプションが設定されます。

表 7.2 シンボル定数設定オプション

	オプション	シンボル定数設定オプション
1	-R8C、-R8CE、-R8Cxx	-D_R8C_=1

8. ファイル仕様

8.1 ファイル名の付け方

ファイル名指定時に拡張子を省略した場合、標準のファイル拡張子を付加したファイル名を使用します。統合開発環境で使用する標準のファイル拡張子を表 8.1 に示します。

表 8.1 統合開発環境で使用する標準のファイル拡張子

No.	拡張子	意味
1	a30	アセンブラソースファイル
2	inc	アセンブラインクルードファイル
3	lst	アセンブラリストファイル
4	atg	アセンブラエラータグファイル
5	obj	オブジェクトファイル
6	abs	アブソリュートファイル
7	map	リンケージリストファイル
8	id	ID ファイル
9	lib	ライブラリファイル
10	lbp	ライブラリリストファイル
11	mot	モトローラ S フォーマットファイル
12	hex	インテル(拡張)HEX フォーマットファイル
13	bin	バイナリファイル
14	sni	スタック情報ファイル
15	pro	プロファイル情報ファイル
16	dbg	デバッグ情報ファイル
17	rti	拡張子 td のファイルで指定された定義を含むオブジェクトファイル
18	cal	呼び出し情報ファイル
19	bls	外部シンボル割り付け情報ファイル
20	utl	utl30 向けファイル
21	rel	リロケータブルファイル

rti_ではじまるファイル名はシステム予約名ですので使用しないでください。

8.2 アセンブラソースファイル

8.2.1 ソースファイルフォーマット

テキスト形式のファイルです。テキストエディタなどで「3. アセンブラ記述規則」に従って記述してください。

8.2.2 ソースファイル名

任意のソースファイル名を指定してください。本アセンブラではソースファイルの拡張子はデフォルトで".a30"です。他の拡張子でファイル名を定義した場合は、アセンブラを起動する際にフルネームでファイルを指定してください。

8.3 インクルードファイル

8.3.1 インクルードファイルフォーマット

テキストエディタなどで「プログラムの記述規則」に従って記述してください。

8.3.2 インクルードファイル名

任意のインクルードファイル名を指定してください。本アセンブラではインクルードファイルの拡張子はデフォルトで".inc"です。他の拡張子でファイル名を定義した場合は、インクルードファイルを指定しているソース行にフルネームで指定してください。

8.4 アセンブラリストファイル

8.4.1 アセンブラリストファイルの構成

アセンブラリストファイルには、アセンブル結果の情報を表示します。
アセンブラリストファイルの構成と内容を表 8.2 に示します。

表 8.2 アセンブラリストファイルの構成と内容

No.	リストファイルへ表示する情報	内容	オプション"-H"指定時
1	リストヘッダ情報	アセンブラリストファイルの生成日時、ページおよびオブジェクト情報のヘッダ情報	出力しない
2	オブジェクト情報	オブジェクトコード、ソースコード	出力する
3	統計情報	エラーの総数、ソースプログラムの行数、セクションサイズ	出力する

【注】オプション"-H"は、オプション"-L" を指定した場合に有効です。

8.4.2 リストヘッダ情報

リストヘッダ情報は、アセンブラリストファイルにデフォルトで出力されます。
なお、オプション"-H"が指定された場合は、出力されません。
出力例は、「図 8.1 アセンブラリストファイル出力例」を参照ください。

8.4.3 オブジェクト情報

オブジェクト情報の出力例を「図 8.1 アセンブラリストファイル出力例」に示します。

```
* M16C Series and R8C Family Assmbler *   SOURCE LIST       Tue Jun 1 12:34:56 2010  PAGE 001

  SEQ.   LOC.    OBJ.              OXMSDA ....*....SOURCE STATEMENT....7....*....8....*....9....
  (1)    (2)     (3)               (4)    (5)

    1                                      ;                                                        ヘッダ情報
    2                                      ;          AS30 sample source file
    3                                      ;
    4                                      ;----------------------------------
    5
    6                                      ; Macro define
    7                                  D   mac1        .MACRO     p1,p2
    8                                  D               MOV.W      p1,p2
    9                                  D               MOV.W      p1,R2
   10                                  D               MOV.W      p2,R3
   11                                                  .ENDM
   12
   13                                                  .SECTION   ram1,data
   14  00000(000001H)                     work1:      .BLKB      1
   15  00001(000001H)                     work2:      .BLKB      1
   16
   17  00000001h                          sym1        .EQU       1
   18  00000002h                          sym2        .EQU       2
   19
   20                                                  .SECTION   prog1,code
   21  00000                              samp_start:
   22
   23                                     .IF  MODE == 1
   24                                 X               MOV.B      #sym1,ROL
   25                                     .ELI  MODE == 2
   26                                 X               MOV.B      #sym2,ROL
   27                                     .ELSE
   28  00000  B4                      Z               MOV.B      #0,ROL
   29                                     .ENDIF
   30
   31  00001                              ..tl0001:
   32  00001  E301                    S               CMP.B      #sym1,ROH
   33  00003  6A04                                    JEQ        ..tl0002
   34  00005  D802                    Q               MOV.B      #0,R1L
   35  00007  61                      S               JMP        lab1
   36  00008                              ..tl0002:
   37  00008  D803                    Q               MOV.B      #0,R1H
   38  0000A                              lab1:
   39  0000A  D91F0000r               Q               MOV.W      #sym1,work1
   40  0000E  D92F0000r               Q               MOV.W      #sym2,work2
   41                                                  mac1       R0,12h
   42  00012  730F1200                M               MOV.W      R0,12h
   43  00016  7302                    M               MOV.W      R0,R2
   44  00018  73F31200                M               MOV.W      12h,R3
   45                                 M               .ENDM
   46                                                  .END
```

図 8.1 アセンブラリストファイル出力例

(1) リスト行情報(SEQ.)
アセンブラリストの行番号を出力します。

(2) ロケーション情報(LOC.)
アセンブル時に決定できる範囲のオブジェクトコードのロケーションアドレスを出力します。

(3) オブジェクトコード情報(OBJ.)
ニーモニックに対応するオブジェクトコードを出力します。

(4) 行情報(0XMDA)
as30 がソース行を処理した結果の情報を出力します。次の情報が出力されます。

表 8.3 アセンブラリストファイルの行情報

0	X	M	S	D	A	内　　容
0-9						インクルードファイルのネストレベルを示します。
	X					条件アセンブルで条件が偽となった行を示します。
		M				マクロ命令の展開行であることを示します。
		D				マクロ命令の定義行であることを示します。
			S			構造化記述文の展開行であることを示します。
				S		分岐距離指定子 S を指定したことを示します。
				B		分岐距離指定子 B を指定したことを示します。
				W		分岐距離指定子 W を指定したことを示します。
				A		分岐距離指定子 A を指定したことを示します。
				Z		命令フォーマットのゼロ形式(:Z)を選択したことを示します。
				S		命令フォーマットのショート形式(:S)を選択したことを示します。
				Q		命令フォーマットのクイック形式(:Q)を選択したことを示します。
					*	8 ビット SB 相対アドレッシングモードを選択したことを示します。

(5) ソース情報(SOURCE STATEMENT)
アセンブラソースファイルの内容を表示します。

8.4.4 統計情報

統計情報の出力例を図 8.2 に示します。

```
Information List      (1)

TOTAL ERROR(S)      00000
TOTAL WARNING(S)    00000
TOTAL LINE(S)       00046    LINES

Section List          (2)

Attr                         SizeName
DATA      0000002(00002H)     ram1
CODE      0000028(0001CH)    prog1
```

図 8.2 統計情報の出力例

(1) エラー、警告それぞれのメッセージ数と、ソース行の総数

(2) セクション情報(セクション属性、サイズ、セクション名)

8.5 アセンブラエラータグファイル

as30 は、コマンドオプション(-T または -X)を指定した場合のみアセンブラソースファイルをアセンブルする際に発生したエラーをファイルに出力します。

(1) アセンブラエラータグファイル名
アセンブラソースファイルの拡張子(デフォルトでは".a30")を".atg"に変更したものが、アセンブラエラータグファイルのファイル名になります。(sample.a30→sample.atg)

(2) ファイル生成ディレクトリ
コマンドオプション(-O)でディレクトリを指定した場合は、そのディレクトリに生成します。
指定のない場合は、アセンブラソースファイルのあるディレクトリに生成します。

8.6 リンケージリスト

最適化リンケージエディタが出力するリンケージリストの内容と形式について説明します。

8.6.1 リンケージリストの構成

リンケージリストの構成と内容を表 8.4 に示します。

表 8.4 リンケージリストの構成と内容

No.	リストファイルへ表示する情報	内容	show オプション* 指定	show オプション省略時
1	オプション情報	コマンドライン、サブコマンドで指定したオプション列を表示	なし	出力する
2	エラー情報	エラーメッセージを表示	なし	出力する
3	リンケージマップ情報	セクション名、先頭／最終アドレス、サイズ、種別を表示	なし	出力する
4	シンボル情報	静的定義シンボル名、アドレス、サイズ、種別をアドレス順に表示 show=reference を指定した場合は、各シンボルの参照回数、最適化実行有無も表示	show=symbol show=reference	出力しない 出力しない
5	シンボル削除最適化情報	最適化で削除したシンボルを表示	show=symbol	出力しない
6	クロスリファレンス情報	シンボルの参照情報を表示	show=xreference	出力しない
7	合計セクションサイズ	RAM,ROM,およびプログラムセクションの合計サイズを表示	show=total_size	出力しない
8	ベクタ情報	ベクタ番号とアドレスの情報を表示	show=vector	出力しない
9	CRC 情報	CRC の演算結果および出力位置アドレスを表示	なし	CRC オプション指定時は常に出力

【注】*show オプションは list オプションを指定した場合に有効です。

8.6.2 オプション情報

コマンドライン、サブコマンドファイルで指定したオプション列を出力します。オプション情報の出力例を図 8.3 に示します(optlnk -sub=test.sub -list -show 指定時)。

```
(test.subの内容)
INPUT test.obj
```

```
*** Options ***

-sub=test.sub
INPUT test.obj (2)   } (1)
-list
-show
```

図 8.3 オプション情報の出力例(リンケージリスト)

(1) コマンドライン、サブコマンドで指定したオプション列を、指定順に出力します。

(2) サブコマンドファイルtest.sub内のサブコマンドです。

8.6.3 エラー情報

エラーメッセージを出力します。エラー情報の出力例を図 8.4 に示します。

```
*** Error Information ***

** L2310 (E) Undefined external symbol “strcmp” referred to in “test.obj”   } (1)
```

図 8.4 エラー情報の出力例(リンケージリスト)

(1) エラーメッセージを出力します。

8.6.4 リンケージマップ情報

各セクションの先頭／最終アドレス、サイズ、種別をアドレス順に出力します。リンケージマップ情報の出力例を図 8.5 に示します。

```
*** Mapping List ***

SECTION                          START      END        SIZE   ALIGN
  (1)                             (2)       (3)         (4)    (5)

P
                                 00001000  00001000       1   1
C
                                 00001004  00001007       4   4
D_2
                                 00001008  000014dd     4d6   2
B_2
                                 000014de  000050b3     3bd6  2
```

図 8.5 リンケージマップ情報の出力例(リンケージリスト)

(1) セクション名を表示します。

(2) 先頭アドレスを表示します。

(3) 最終アドレスを表示します。

(4) セクションサイズを表示します。

(5) セクションのアライメント数を表示します。

8.6.5 シンボル情報

show=symbol を指定した場合、外部定義シンボルまたは静的内部定義シンボルのアドレス、サイズ、種別をアドレス順に出力します。また、show=reference を指定した場合は、各シンボルの参照回数、最適化実行の有無も出力します。シンボル情報の出力例を図 8.6 に示します。

```
*** Symbol List ***

SECTION=(1)
FILE=(2)          START          END            SIZE
                   (3)           (4)             (5)
  SYMBOL          ADDR           SIZE           INFO           COUNTS  OPT
  (6)              (7)           (8)             (9)            (10)   (11)

SECTION=P
FILE=test.obj
                  00000000       00000428           428
  _main
                  00000000              2       func ,g             0
  _malloc
                  00000000             32       func ,l             0
FILE=mvn3
                  00000428       00000490            68
  $MVN#3
                  00000428              0       none ,g             0
```

図 8.6 シンボル情報の出力例(リンケージリスト)

(1) セクション名を表示します。

(2) ファイル名を表示します。

(3) (2)のファイルに含まれる該当セクションの先頭アドレスを表示します。

(4) (2)のファイルに含まれる該当セクションの最終アドレスを表示します。

(5) (2)のファイルに含まれる該当セクションのセクションサイズを表示します。

(6) シンボル名を表示します。

(7) シンボルアドレスを表示します。

(8) シンボルサイズを表示します。

(9) シンボル種別を次のように表示します。

データ種別:	func	・・・・・・・・	関数名
	data	・・・・・・・・	変数名
	entry	・・・・・・・・	エントリ関数名
	none	・・・・・・・・	未設定(ラベル、アセンブラシンボル)
宣言種別:	g	・・・・・・・・	外部定義
	l	・・・・・・・・	内部定義

(10) シンボル参照回数を表示します。show=referenceを指定した場合のみ表示します。参照回数を表示しないときは、*を表示します。

(11) 最適化有無を次のように表示します。

ch	・・・・・・・・	最適化によって変更されたシンボル
cr	・・・・・・・・	最適化によって生成されたシンボル
mv	・・・・・・・・	最適化によって移動されたシンボル

8.6.6 シンボル削除最適化情報

シンボル削除最適化(optimize=symbol_delete)によって削除されたシンボルのサイズ、種別を出力します。シンボル削除最適化情報の出力例を図 8.7 に示します。

```
*** Delete Symbols ***

SYMBOL                          SIZE    INFO
  (1)                           (2)     (3)
  _Version
                                   4    data ,g
```

図 8.7 シンボル削除情報の出力例(リンケージリスト)

(1) 削除シンボル名を表示します。

(2) 削除シンボルサイズを表示します。

(3) 削除シンボルの種別を以下のように表示します。

データ種別:	func	・・・・・・・・	関数名
	data	・・・・・・・・	変数名
宣言種別:	g	・・・・・・・・	外部定義
	l	・・・・・・・・	内部定義

8.6.7 クロスリファレンス情報

show=xrefernce を指定した場合、シンボルの参照情報(クロスリファレンス情報)を出力します。クロスリファレンス情報の出力例を図 8.8 に示します。

```
*** Cross Reference List ***

No    Unit Name  Global.Symbol   Location    External Information
(1)      (2)          (3)          (4)              (5)
0001  a
      SECTION=P  _func
                                 00000100
                 _func1
                                 00000116
                 _main
                                 0000012c
                 _g
                                 00000136
      SECTION=B
                 _a
                                 00000190    0001(00000140:P)
                                             0002(00000178:P)
                                             0003(0000018c:P)
0002  b
      SECTION=P
                 _func01
                                 00000154    0001(00000148:P)
                 _func02
                                 00000166    0001(00000150:P)
0003  c
      SECTION=P
                 _func03
                                 00000184
```

図 8.8 クロスリファレンス情報の出力例(リンケージリスト)

(1) Unit番号。オブジェクト単位の識別番号。

(2) オブジェクト名。リンク時の入力指定順になる。

(3) シンボル名。セクションごとに配置アドレスの昇順に出力される。

(4) シンボルの配置アドレス。
form=rel 指定時は、セクション先頭からの相対値となる。

(5) 参照している外部シンボルのアドレスを表す。
出力形式は以下のようになる。
〈Unit 番号〉(〈アドレス or セクション内オフセット〉:〈セクション名〉)

8.6.8 合計セクションサイズ

show=total_size を指定した場合、ROM セクション、RAM セクション、およびプログラムセクションの合計サイズを出力します。合計の出力例を図 8.9 に示します。

```
*** Total Section Size ***

RAMDATA SECTION:        00000660 Byte(s)
(1)
ROMDATA SECTION:        00000174 Byte(s)
(2)
PROGRAM SECTION:        000016d6 Byte(s)
(3)
```

図 8.9 合計セクションサイズの出力例(リンケージリスト)

(1) RAMデータセクションの合計サイズ。

(2) ROMデータセクションの合計サイズ。

(3) プログラムセクションの合計サイズ。

8.6.9 可変ベクタテーブル情報

show=vector を指定した場合、可変ベクタテーブルの内容を表示します。可変ベクタテーブルの出力例を図 8.10 に示します。

```
*** Variable Vector Table List ***


NO.     SYMBOL/ADDRESS
(1)     (2)
  0     __brk
  1     __dummy_int
  2     __dummy_int
  3     __dummy_int
  4     __int3
  5     __timer_b5
  :
  <省略>
```

図 8.10 可変ベクタテーブルの出力例(リンケージリスト)

(1) 可変ベクタ番号。

(2) シンボルを表示します。シンボルが定義されていない場合はアドレスで表示します。

8.6.10 スペシャルページベクタテーブル情報

show=vector を指定した場合、スペシャルページベクタテーブルの内容を表示します。スペシャルページベクタテーブルの出力例を図 8.11 に示します。

```
*** Special Vector Table List ***


NO.    SYMBOL/ADDRESS
(1)    (2)
 20     _sfunc20
 19     _sfunc19
 18     _sfunc18
```

図 8.11 スペシャルページベクタテーブルの出力例(リンケージリスト)

(1) スペシャルページベクタ番号。

(2) シンボルを表示します。シンボルが定義されていない場合はアドレスで表示します。

8.6.11 IDコード、プロテクトコードおよびOFSREGコード情報

アセンブラ指示命令".ID"、".PROTECT"または".OFSREG"設定時に各データの設定情報を出力します。各種情報の出力例を図 8.12 に示します。

```
*** ID code information ***  (1)

CHARACTOR STRING="sample"
NUMERICAL VALUE=
 0000ffdf: 73
 0000ffe3: 61
 0000ffeb: 6d
 0000ffef: 70
 0000fff3: 6c
 0000fff7: 65
 0000fffb: 00

*** Protect code or OFSREG code information ***  (2)

 0000ffff: ff
```

図 8.12 ID コード、プロテクトコードおよび OFSREG 情報の出力例(リンケージリスト)

(1) IDコード情報

(2) プロテクトコードまたはオプション機能選択レジスタ値

8.7 IDファイル

アセンブラ指示命令".ID"設定時、ID ファイルを出力します。ID ファイルの出力例を図 8.13.1 および図 8.13.2 に示します。

```
-IDsample -protectx FF
FFFDF : 73
FFFE3 : 61
FFFEB : 6D
FFFEF : 70
FFFF3 : 6C
FFFF7 : 65
FFFFB : 00
FFFFF : FF
```

図 8.13.1 ID ファイルの出力例

```
-IDsample -ofsregx FF
0FFDF : 73
0FFE3 : 61
0FFEB : 6D
0FFEF : 70
0FFF3 : 6C
0FFF7 : 65
0FFFB : 00
0FFFF : FF
```

図 8.13.2 ID ファイルの出力例(-R8C、-R8CE または-R8Cxx 設定時)

8.8 ライブラリリスト

本節では、最適化リンケージエディタが出力するライブラリリストの内容と形式について説明します。

8.8.1 ライブラリリストの構成

ライブラリリストの構成と内容を表 8.5 に示します。

表 8.5 ライブラリリストの構成と内容

No.	リストの作成	内容	サブオプション*	show オプション省略時
1	オプション情報	コマンドライン、サブコマンドで指定した オプション列を表示	—	出力する
2	エラー情報	エラーメッセージを表示	—	出力する
3	ライブラリ情報	ライブラリ情報を表示	—	出力する
4	ライブラリ内モジュール、 セクション、シンボル情報	ライブラリ内モジュールを表示	—	出力する
		show=symbol を指定した場合は、 モジュール内シンボル名一覧も表示	show=symbol	出力しない
		show=section を指定した場合は、 各モジュール内セクション名、 シンボル名一覧も表示	show=section	出力しない

【注】*すべてのオプションは、list オプションを指定した場合に有効です。

8.8.2 オプション情報

コマンドライン、サブコマンドファイルで指定したオプション列を出力します。オプション情報の出力例を図 8.14 に示します(optlnk -sub=test.sub -list -show を指定した場合)。

```
 (test.subの内容)
form    library
in      adhry.obj
output  test.lib
```

```
*** Options ***

-sub=test.sub
form    library
in      adhry.obj      } (2)
output  test.lib       }
-list                  }
-show                  } (1)
```

図 8.14 オプション情報の出力例(ライブラリリスト)

(1) コマンドライン、サブコマンドで指定したオプション列を、指定順に出力します。

(2) サブコマンドファイルtest.sub内のサブコマンドです。

8.8.3 エラー情報

エラー、ウォーニングなどのメッセージを出力します。エラー情報の出力例を図 8.15 に示します。

```
*** Error Information ***

** L1200 (W) Backed up file “main.lib” into “main.lbk”    (1)
```

図 8.15 エラー情報の出力例(ライブラリリスト)

(1) ウォーニングメッセージを出力します。

8.8.4 ライブラリ情報

ライブラリの種別を出力します。ライブラリ情報の出力例を図 8.16 に示します。

```
*** Library Information ***

LIBRARY NAME=test.lib   (1)
CPU=SuperH              (2)
ENDIAN=Big              (3)
ATTRIBUTE=system        (4)
NUMBER OF MODULE=1      (5)
```

図 8.16 ライブラリ情報の出力例(ライブラリリスト)

(1) ライブラリ名を表示します。

(2) マイコン名を表示します。

(3) エンディアン種別を表示します。

(4) ライブラリファイルの属性がシステムライブラリかユーザライブラリかを表示します。

(5) ライブラリ内モジュール数を表示します。

8.8.5 ライブラリ内モジュール、セクション、シンボル情報

ライブラリ内のモジュール一覧を出力します。

show=symbol を指定した場合はモジュール内シンボル名一覧を、show=section を指定した場合はモジュール内セクション名、シンボル名一覧を出力します。

ライブラリ内モジュール、セクション、シンボル情報の出力例を図 8.17 に示します。

```
*** Library List ***

MODULE           LAST UPDATE
 (1)                 (2)
  SECTION
    (3)
    SYMBOL
      (4)
adhry
                 29-Feb-2000 12:34:56
  P
   _main
   _Proc0
   _Proc1
  C
  D
   _Version
  B
   _IntGlob
   _CharGlob
```

図 8.17 ライブラリ内モジュール、セクション、シンボル情報の出力例(ライブラリリスト)

(1) モジュール名を表示します。

(2) モジュールを登録した日付を表示します。モジュールが更新された場合は、最新の更新日付を表示します。

(3) モジュール内セクション名を表示します。

(4) セクション内をシンボル表示します。

9. アセンブラ指示命令

9.1 アドレス制御指示命令

アドレス制御指示命令は、アセンブラがアドレス更新をする場合の指示を行う指示命令です。
絶対アドレス形式セクション内のアドレスを除いて、アセンブラが制御を行うアドレスはリロケータブル値になります。

表 9.1 アドレス制御指示命令

指示命令	機能内容
.ORG	開始アドレスを宣言します。 本指示命令を記述したセクションは、絶対アドレス形式セクションとなります。
.BLKB	1 バイト単位で RAM 領域を確保します。
.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。
.WORD(S)	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数データを ROM 領域に格納します。
.DOUBLE	8 バイトで表される浮動小数点数データを ROM 領域に格納します。
.ALIGN	奇数アドレスを偶数アドレスに変換することを指示します。

アドレス宣言

.ORG

書 式　　[△].ORG△＜オペランド＞

説 明　　本指示命令をセクション定義指示命令".SECTION"の直後に記述することで、当該セクションを絶対属性とします。
本指示命令は絶対属性セクション内では複数回記述できます。
相対属性セクション内には、本指示命令は記述できません。

記述例

```
.SECTION   value,ROMDATA
.ORG       0FF00H
.BYTE      "abcdefghijklmnopqrstuvwxyz"
.ORG       0FF80H
.BYTE      "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
.END
```

次の記述は、相対属性セクションでの本指示命令使用のため、エラーになります。

```
.SECTION   value,ROMDATA
.BYTE      "abcdefghijklmnopqrstuvwxyz"
.ORG       0FF80H
.BYTE      "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
.END
```

備 考　　オペランドに記述できる値は、0～0FFFFFH の範囲の数値です(-R8C オプション指定時の範囲は 0～0FFFFH)。
オペランドには式、シンボルを記述できます。ただし、オペランドの値はアセンブル実行時に確定する値でなければなりません。

注意事項　　絶対属性セクションは、リンク時にアドレスの再配置ができません。
セクション定義指示命令".SECTION"を記述した直後の行に".ORG"の記述が無い場合は、そのセクションは相対属性セクションになります。
セクションタイプが CODE または ROMDATA の同一名称セクション内に複数の".ORG"を記述した場合、コードが存在しない空き領域には NOP 命令(04H)が埋め込まれます。

1 バイト長の RAM 領域を確保

.BLKB

書 式　　[△][＜ラベル名：＞△].BLKB△＜オペランド＞

説 明　　1 バイト単位で、オペランドで指定したバイト数の RAM 領域を確保します。
オペランドには式、シンボルが記述できます。ただし、オペランドの値はアセンブル実行時に確定する値でなければなりません。

記述例

```
symbol    .EQU    1
.SECTION  area,DATA
work1:    .BLKB   1
work2:    .BLKB   symbol
          .BLKB   symbol+1
```

注意事項　　本指示命令は必ず、DATA タイプのセクション内に記述してください。
ラベル名には、必ずコロン(:)を記述してください。

2 バイト長の RAM 領域を確保

.BLKW

書 式　　[△][＜ラベル名：＞△].BLKW△＜オペランド＞

説 明　　2バイト単位で、オペランドで指定したバイト数の RAM 領域を確保します。
オペランドには式、シンボルが記述できます。ただし、オペランドの値はアセンブル実行時に確定する値でなければなりません。

記述例

```
symbol    .EQU    1
.SECTION  area,DATA
work1:    .BLKW   1
work2:    .BLKW   symbol
          .BLKW   symbol+1
```

注意事項　　本指示命令は必ず、DATA タイプのセクション内に記述してください。
ラベル名には、必ずコロン(:)を記述してください。

3 バイト長の RAM 領域を確保

.BLKA

書 式　　[△][＜ラベル名：＞△].BLKA△＜オペランド＞

説 明　　3バイト単位で、オペランドで指定したバイト数の RAM 領域を確保します。
オペランドには式、シンボルが記述できます。ただし、オペランドの値はアセンブル実行時に確定する値でなければなりません。

記述例

```
symbol    .EQU    1
.SECTION  area,DATA
work1:    .BLKA   1
work2:    .BLKA   symbol
          .BLKA   symbol+1
```

注意事項　　本指示命令は必ず、DATA タイプのセクション内に記述してください。
ラベル名には、必ずコロン(:)を記述してください。

4 バイト長の RAM 領域を確保

.BLKL

書 式　　[△][＜ラベル名：＞△].BLKL△＜オペランド＞

説 明　　4バイト単位で、オペランドで指定したバイト数の RAM 領域を確保します。
オペランドには式、シンボルが記述できます。ただし、オペランドの値はアセンブル実行時に確定する値でなければなりません。

記述例

```
symbol    .EQU    1
.SECTION  area,DATA
work1:    .BLKL   1
work2:    .BLKL   symbol
          .BLKL   symbol+1
```

注意事項　　本指示命令は必ず、DATA タイプのセクション内に記述してください。
ラベル名には、必ずコロン(:)を記述してください。

4 バイト長の RAM 領域を確保

.BLKF

書 式　　[△][＜ラベル名：＞△].BLKF△＜オペランド＞

説 明　　4バイト単位で、オペランドで指定したバイト数の RAM 領域を確保します。
オペランドには式、シンボルが記述できます。ただし、オペランドの値はアセンブル実行時に確定する値でなければなりません。

記述例

```
symbol    .EQU    1
.SECTION  area,DATA
work1:    .BLKF   1
work2:    .BLKF   symbol
          .BLKF   symbol+1
```

注意事項　　本指示命令は必ず、DATA タイプのセクション内に記述してください。
ラベル名には、必ずコロン(:)を記述してください。

8 バイト長の RAM 領域を確保

.BLKD

書 式　　[△][＜ラベル名：＞△].BLKD△＜オペランド＞

説 明　　8バイト単位で、オペランドで指定したバイト数の RAM 領域を確保します。
オペランドには式、シンボルが記述できます。ただし、オペランドの値はアセンブル実行時に確定する値でなければなりません。

記述例

```
symbol    .EQU    1
.SECTION  area,DATA
work1:    .BLKD   1
work2:    .BLKD   symbol
          .BLKD   symbol+1
```

注意事項　　本指示命令は必ず、DATA タイプのセクション内に記述してください。
ラベル名には、必ずコロン(:)を記述してください。

1 バイト長の ROM データを格納

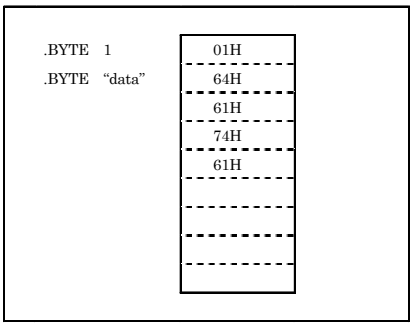
.BYTE

書 式　　[△][＜ラベル名:＞△].BYTE△＜オペランド＞

説 明　　オペランドで指定の 1 バイト長のデータを ROM に格納します。
オペランドには式、シンボルが記述できます。
複数のオペランドを記述するときは、カンマ(,)で区切って記述してください。
オペランドにはシングルクォーテーション(')または、ダブルクォーテーション(")で囲って、文字または、文字列を記述できます。このとき格納されるデータは、文字の ASCII コードになります。

記述例

```
.SECTION  value,ROMDATA
.BYTE     1
.BYTE     "data"
.BYTE     symbol
.BYTE     symbol+1
.BYTE     1,2,3,4,5
.END
```



注意事項　　本指示命令は必ず、DATA タイプ以外のセクション内に記述してください。
ラベル名には、必ずコロン(:)を記述してください。

(符号付) 2 バイト長の ROM データを格納

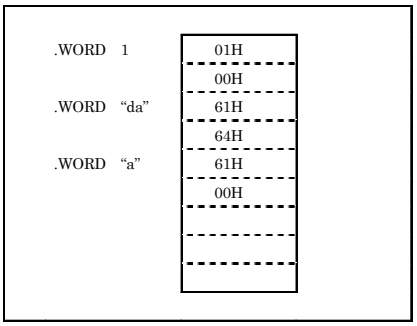
.WORD (S)

書 式　　[△][＜ラベル名:＞△].WORD△＜オペランド＞
[△][＜ラベル名:＞△].WORDS△＜オペランド＞

説 明　　オペランドで指定の2バイト長のデータを ROM に格納します。
オペランドには式、シンボルが記述できます。
複数のオペランドを記述するときは、カンマ(,)で区切って記述してください。
オペランドにはシングルクォーテーション(')または、ダブルクォーテーション(")で囲って、文字または、文字列を記述できます。このとき格納されるデータは、文字の ASCII コードになります。

記述例

```
.SECTION  value,ROMDATA
.WORD     1
.WORD     "da","a"
.WORD     symbol
.WORD     symbol+1
.WORD     1,2,3,4,5
.END
```



注意事項　　本指示命令は必ず、DATA タイプ以外のセクション内に記述してください。
ラベル名には、必ずコロン(:)を記述してください。
オペランドに記述できる文字列長は、2文字までです。
".WORD"のオペランドに記述できる値は-32768 から 65535、".WORDS"のオペランドに記述できる値は-32768 から 32767 になります。

3 バイト長の ROM データを格納

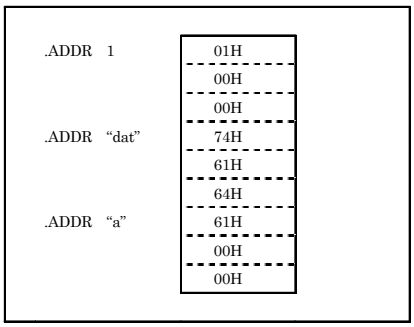
.ADDR

書 式　　[△][<ラベル名:>△].ADDR△<オペランド>

説 明　　オペランドで指定の3バイト長のデータを ROM に格納します。
オペランドには式、シンボルが記述できます。
複数のオペランドを記述するときは、カンマ(,)で区切って記述してください。
オペランドにはシングルクォーテーション(')または、ダブルクォーテーション(")で囲って、文字または、文字列を記述できます。このとき格納されるデータは、文字の ASCII コードになります。

記述例

```
.SECTION  value,ROMDATA
.ADDR     1
.ADDR     "dat","a"
.ADDR     symbol
.ADDR     symbol+1
.ADDR     1,2,3,4,5
.END
```



注意事項　　本指示命令は必ず、DATA タイプ以外のセクション内に記述してください。
ラベル名には、必ずコロン(:)を記述してください。
オペランドに記述できる文字列長は、3文字までです。

4 バイト長の ROM データを格納

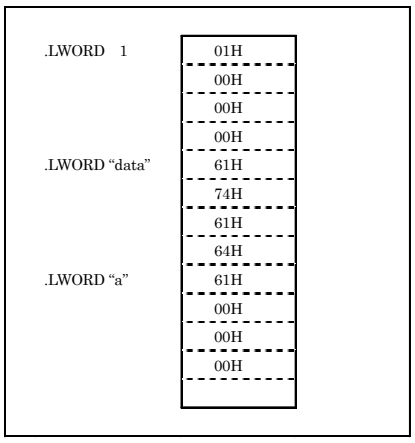
.LWORD

書 式　　[△][＜ラベル名:＞△].LWORD△＜オペランド＞

説 明　　オペランドで指定の4バイト長のデータを ROM に格納します。
オペランドには式、シンボルが記述できます。
複数のオペランドを記述するときは、カンマ(,)で区切って記述してください。
オペランドにはシングルクォーテーション(')または、ダブルクォーテーション(")で囲って、文字または、文字列を記述できます。このとき格納されるデータは、文字の ASCII コードになります。

記述例

```
.SECTION  value,ROMDATA
.LWORD    1
.LWORD    "data","a"
.LWORD    symbol
.LWORD    symbol+1
.LWORD    1,2,3,4,5
.END
```



注意事項　　本指示命令は必ず、DATA タイプ以外のセクション内に記述してください。
ラベル名には、必ずコロン(:)を記述してください。
オペランドに記述できる文字列長は、4文字までです。

4 バイト長の ROM データを格納

.FLOAT

書 式　　[△][＜ラベル名：＞△].FLOAT△＜オペランド＞

説 明　　オペランドで指定の 4 バイト長の浮動小数点データを ROM に格納します。

記述例

```
                .FLOAT    5E2
const:          .FLOAT    5e2
```

注意事項　　浮動小数点数の記述方法は、「3.5.2 オペランドの記述規則」を参照してください。
ラベル名には必ず、コロン(:)を記述してください。

8 バイト長の ROM データを格納

.DOUBLE

書 式　　[△][＜ラベル名:＞△].DOUBLE△＜オペランド＞

説 明　　オペランドで指定の8バイト長の浮動小数点データを ROM に格納します。

記述例

```
                .DOUBLE    5E2
    const:      .DOUBLE    5e2
```

備 考　　浮動小数点数の記述方法は、「3.5.2 オペランドの記述規則」を参照してください。
ラベル名には必ず、コロン(:)を記述してください。

アドレス補正

.ALIGN

書 式　　[△].ALIGN

説 明　　本指示命令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、アドレスを補正した結果、空になったところに NOP のコード(04H)を書き込みます。
セクションタイプが DATA の場合は、アドレス値を+1 します。
本指示命令を記述した箇所のアドレスが偶数の場合は、補正は行いません。

記述例

```
.SECTION   program,CODE,ALIGN
MOV.W      #0,R0
.ALIGN
MOV.W      #0,R1

.SECTION   program,CODE
.ORG       0f000H
MOV.W      #0,R0
.ALIGN
MOV.W      #0,R1
.END
```

注意事項　　相対セクションの場合、セクション定義指示命令".SECTION"に",ALIGN"を記述してください。

9.2 アセンブル制御指示命令

アセンブル制御指示命令自身はデータを生成しません。命令に対する機械語コードの生成を制御する指示命令です。アドレスの更新は行いません。

表 9.2 アセンブル制御指示命令

指示命令	機能内容
.EQU	シンボルを設定します。
.BTEQU	ビットシンボルを設定します。
.END	アセンブラソースファイルの終了を指定します。
.SB	SB レジスタ値を仮定します。以降の行のアドレッシングモードは仮定した値を基準に生成されます。
.SBSYM	本指示命令で指定したシンボル及びラベルに対して SB 相対アドレッシングモードでコードを生成します。
.SBBIT	本指示命令で指定したビットシンボルに対して SB 相対アドレッシングモードでコードを生成します。
.FB	FB レジスタ値を仮定します。以降の行のアドレッシングモードを、仮定した値を基準に生成します。
.FBSYM	本指示命令で指定したシンボル及びラベルに対して FB 相対アドレッシングモードでコードを生成します。
.INCLUDE	本指示命令を記述した位置に、指定したファイルの内容を読み込みます。
.SB_AUTO(_xxx)	SB 相対アドレッシングの自動生成を実施します。

数値シンボル定義

.EQU

書 式　　[△]＜シンボル名＞△.EQU△＜オペランド＞

説 明　　シンボルに 32 ビット符号付き整数値(−2147483648～2147483647)の範囲の値を定義します。
オペランドには式、シンボルが記述できます。ただし、オペランドの値はアセンブル実行時に確定する値でなければなりません。
シンボル名は、グローバル指定ができます。

記述例

```
symbol    .EQU   1
symbol1   .EQU   symbol+symbol
symbol2   .EQU   2
```

注意事項　　オペランドには前方参照となるシンボル名は記述できません。

ビットシンボル定義

.BTEQU

書 式　[△]＜ビットシンボル＞△.BTEQU△＜ビット位置＞,＜アドレス値＞
[△]＜ビットシンボル＞△.BTEQU△＜ビットシンボル＞

説 明　ビット位置とアドレス値を定義します。本指示命令で定義したシンボルをビットシンボルと呼びます。
本指示命令でビットシンボルを定義することで、ビット処理命令のオペランドにビットシンボルを記述できます。
定義されるビット位置は、指定したアドレス値のメモリの最下位ビットを起点として、ビット位置を示す値をオフセットとしたビットです。
ビット位置を示す数値は、0～65535 の範囲の整数値が記述できます。
ビット位置には式、シンボルが記述できます。ただし、オペランドの値はアセンブル実行時に確定する値でなければなりません。
アドレス値には式、シンボルが記述できます。
ビットシンボル名は、グローバル指定ができます。

記述例

```
.GLB    flag1
one     .EQU      1
bit0    .BTEQU    0,0
bit1    .BTEQU    1,flag
bit2    .BTEQU    2,flag+1
bit3    .BTEQU    one+one,flag
bit4    .BTEQU    one,flag1
bit5    .BTEQU    bit0
```

注意事項　アセンブル実行時に確定しないアドレス値で定義されたビットシンボルを外部参照指示命令".BTGLB"のオペランドに定義することはできません。
オペランドのビットシンボル名は前方参照できません。
オペランドのビットシンボルはアセンブル実行時に値が確定するシンボル名を記述してください。

アセンブラソースファイルの終了宣言

.END

書 式　　[△].END

説 明　　ソースプログラムの終了を宣言します。
本指示命令を記述した行以降の記述内容は、リストファイルに出力するのみで、コード生成などの処理は行いません。

記述例

```
.SECTION   tbl,romdata
.BYTE      1,2,3,4,5
.END
```

備 考　　本指示命令は、一つのアセンブラソースファイルに必ず一つ以上記述する必要があります。

注意事項　　as30 は、本指示命令以降の行についてはエラーの検出を行いません。

SB レジスタ値宣言

.SB

書 式　　[△].SB△＜オペランド＞

説 明　　SB レジスタ値を仮定します。
アセンブル実行時に SB レジスタの値を本指示命令で定義した値であると判断し、以降のコードを生成します。
本指示命令行以降で、指示命令".SBSYM"で指定されたラベル名を使用できます。
指示命令".SBSYM"で指定されたラベル名を用いた命令に対して、".SB"で仮定した値を基点とした SB 相対アドレッシングモードでコードを生成します。
オペランドには、0～0FFFFH の範囲の整数値が記述できます。
オペランドには式、シンボルが記述できます。ただし、オペランドの値はアセンブル実行時に確定する値でなければなりません。

記述例

```
.SB     80H
LDC     #80H,SB
```

注意事項　　本指示命令は、アセンブラに対して SB レジスタ値を仮定するように指示する命令であり、実際の SB レジスタ値に値を設定できるものではありません。実際に SB レジスタ値を設定するためには、本指示命令の直前または直後に次の命令を記述してください。

例）LDC　　#80H,SB

SB 相対アドレッシングモード指定

.SBSYM

書 式　　[△].SBSYM△＜名前＞[,＜名前＞. . .]

説 明　　本指示命令のオペランドに指定した名前に対して、SB 相対アドレッシングモードが選択されます。
本指示命令のオペランドに指定した名前を含む、絶対 16 ビットアドレッシングモードの式に対して、SB 相対アドレッシングモードが選択されます。
オペランドにリロケータブルな値をもつ名前を指定できます。

記述例 1

```
.SB         80H
LDC         #80H,SB
.SBSYM      sym1,sym2
```

注意事項　　本指示命令を記述する以前に、必ず指示命令".SB"で SB レジスタ値を設定してください。
本指示命令で指定されたラベル名を使って、指示命令".EQU"指示命令で定義されたシンボルについては、SB 相対アドレッシングモードは選択されません。
本指示命令で指定したシンボルが、指示命令".FBSYM"で指定したシンボルと二重定義にならないように記述してください。

以下の例では、sym2 に対して SB 相対アドレッシングモードは選択されません。

```
.SBSYM   sym1
sym2     .EQU     sym1+1
```

SB 相対アドレッシングモード宣言(ビット操作命令)

.SBBIT

書 式　[△].SBBIT△＜オペランド＞[,＜オペランド＞. . .]

説 明　本指示命令のオペランドに指定した名前に対して、SB 相対アドレッシングモードが選択されます。
ビット処理命令がショート形式を持つ場合は、11 ビット SB 相対アドレッシングまたは 16 ビット SB 相対アドレッシングモードが選択されます。
ビット処理命令がショート形式を持たない場合は、8 ビット SB 相対アドレッシングまたは 16 ビット SB 相対アドレッシングモードが選択されます。
オペランドのビットシンボルが外部参照の場合は、16 ビット SB 相対アドレッシングモードが選択されます。ただし、ショート形式を持つニーモニックに対して、ショート形式(:S)を指定して記述した場合は、11 ビット SB 相対アドレッシングモードが選択されます。
オペランドには、指示命令".BTEQU"及び指示命令".BTGLB"で定義されたビットシンボルを記述できます。

記述例

```
.BTGLB    extbit
.SB       80H
LDC       #80H,SB
.SBBIT    bsym,extbit
BCLR      bsym      ;Select 11 bits SB
BAND      bsym      ;Select 16 bits SB
BSET      extbit    ;16 bits SB
BSET:S    extbit    ;11 bits SB
```

注意事項　オペランドには前方参照となるビットシンボルが記述できます。
本指示命令を記述する以前に、必ず指示命令".SB"で SB レジスタ値を設定してください。

FB レジスタ値宣言

.FB

書 式　　[△].FB△＜オペランド＞

説 明　　FB レジスタ値を仮定します。
アセンブル実行時に FB レジスタの値を本指示命令で定義した値であると判断し、以降のコードを生成します。
以降の行で、指示命令".FBSYM"で指定されたラベル名を使用できます。
指示命令".FBSYM"で指定されたラベル名を用いた命令に対して、.FB で仮定した値を基点とした FB 相対アドレッシングモードでコードを生成します。
オペランドには、0～0FFFFH の範囲の整数値が記述できます。
オペランドには式、シンボルが記述できます。ただし、オペランドの値はアセンブル実行時に確定する値でなければなりません。

記述例

```
.FB     80H
LDC     #80H,FB
```

注意事項　　本指示命令は、アセンブラに対して FB レジスタ値を仮定するように指示する命令であり、実際の FB レジスタ値に値を設定できるものではありません。実際に FB レジスタ値を設定するためには、本指示命令の直前または直後に次の命令を記述してください。

例）LDC　　#80H,FB

FB 相対アドレッシングモード選択

.FBSYM

書 式　　[△].FBSYM△＜名前＞[,＜名前＞...]

説 明　　本指示命令のオペランドに指定した名前に対して、FB 相対アドレッシングモードが選択されます。
本指示命令のオペランドに指定した名前を含む、絶対 16 ビットアドレッシングモードのオペランドに対して、FB 相対アドレッシングモードが選択されます。
オペランドにリロケータブルな値をもつ名前を指定できます。

記述例

```
.FB     80H
LDC     #80H,FB
.FBSYM  sym1,sym2
```

備 考　　本指示命令を記述する以前に、必ず指示命令".FB"で FB レジスタ値を設定してください。
本指示命令で指定したシンボルが、指示命令".SBSYM"で指定したシンボルと二重定義にならないように記述してください。

インクルードファイル指定

.INCLUDE

書 式　　[△].INCLUDE△＜ファイル名＞

説 明　　ソースプログラムの行に、他のファイルの内容全てを読み込みます。
本指示命令で読み込まれたファイルの内容は、読み込んだファイル内に記述した場合と、同じ一つのファイルとして処理されます。
インクルードファイルは 9 レベルまでネスティングできます。
インクルードファイル名に絶対パスを記述した場合は、記述したディレクトリ内のファイルを検索します。ファイルが見つからない場合はエラーとなります。
インクルードファイル名に絶対パスを記述していない場合は、次に示す順序でファイルを検索します。

(1) as30起動時にコマンド行で指定したファイル名にディレクトリ指定がない場合は、インクルード指示命令で指定されたファイル名を検索します。as30起動時にコマンド行で指定したファイル名にディレクトリ指定がある場合は、インクルード指示命令で指定されたファイル名にコマンド行で指定されたディレクトリ名を付加して検索します。
(2) コマンドオプション-Iで指定されたディレクトリを検索します。
(3) 環境変数INC30に設定されているディレクトリを検索します。

記述例

```
.INCLUDE   initial.a30
.INCLUDE   ..FILE@.inc
```

備 考　　オペランドのファイル名には、必ずファイル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。

注意事項　　インクルードファイル内で、自分自身をインクルード指定しないでください。

SB 相対アドレッシングの自動生成

.SB_AUTO（_xxx）

書 式

```
[△].SB_AUTO
[△].SB_AUTO_S          ;C 言語表記関数名，アセンブラ表記関数名
[△].SB_AUTO_SBVAL      ;SB レジスタの設定値
[△].SB_AUTO_SBSYM      ;SB 相対アドレッシング対象シンボル名
[△].SB_AUTO_R
[△].SB_AUTO_E
```

説 明

SB 相対アドレッシングモードが選択されます。
SB レジスタの退避、復帰およびレジスタ値の設定命令を生成します。

表 9.3 SB 相対アドレッシングの自動生成指示命令

指示命令	内容
.SB_AUTO	SB 相対アドレッシングの自動生成開始を示します。
.SB_AUTO_S	関数の開始を示します。
.SB_AUTO_SBVAL	SB レジスタの退避命令(PUSHC)および SB レジスタ値の設定命令(LDC)を生成します。
.SB_AUTO_SBSYM	オペランドに指定した名前に対して、SB 相対アドレッシングモードが選択されます。
.SB_AUTO_R	SB レジスタ値の復帰命令(POPC)を生成します。
.SB_AUTO_E	関数の終了を示します。

記述例

```
        .glb   func1
    func1:
        .sb_auto_s        func1,_func1
        .sb_auto_sbval    i1
        .sb_auto_sbsym    i1,_i2,_i3
        ;
        .sb_auto_r
        rts
        .sb_auto_e
```

注意事項

本指示命令は、Cコンパイラ専用の指示命令となりますので記述することはできません。
条件によっては、".SB_AUTO_SBVAL"および".SB_AUTO_R"で命令は生成されません。

9.3 リンク制御指示命令

プログラムを複数のファイルに分割して記述するリロケータブルアセンブルを実行するための指示命令です。

表 9.4 リンク制御指示命令

指示命令	機能内容
.SECTION	アドレスを再配置するための最小の単位となるセクションを定義します。 セクション情報には、セクション名、セクションタイプ及びセクション属性があります。
.GLB	シンボルが外部シンボルであることを宣言します。宣言したシンボルの定義が同一ファイル内にあれば、外部シンボルとなります。 宣言したシンボルの定義が同一ファイル内にない場合は、外部参照シンボルとなります。
.BTGLB	ビットシンボルが外部シンボルであることを宣言します。
.RVECTOR	ソフトウエア割り込み番号とソフトウエア割り込み名を設定します。
.SVECTOR	スペシャルページ番号とスペシャルページ名を設定します。
.INITSCT	セクション名を仮定義します。 本指示命令は、C 言語スタートアップ(initsct.c)の初期化関数で生成される指示命令であり、コンパイラ専用の指示命令です。

セクション定義

.SECTION

書 式　[△].SECTION△＜セクション名＞[,＜セクションタイプ＞][,ALIGN]

説 明　セクションの始まりを宣言します。セクションの範囲は、次のセクション指示命令または指示命令".END"までです。
セクションタイプは、CODE,ROMDATA,DATA のいずれかを記述します。省略時は CODE として扱います。
",ALIGN"を指定した場合、相対セクション属性とし、ファイル内のセクションの先頭アドレスを偶数番地にリンケージエディタで調整します。
セクション指示命令の直後の行に".ORG"指示命令を記述した場合、絶対セクション属性として扱います。

記述例

```
.SECTION    program,CODE
    NOP
.SECTION    ram,DATA
    .BLKB   10
.SECTION    dname,ROMDATA
    .BYTE   "abcd"
.END
```

備 考　セクションタイプと",ALIGN"の記述順序は任意です。

注意事項　同一セクション名のセクション定義を複数宣言した場合、1つのセクションに連結されます。その際、2つめ以降のセクション定義に記述された",ALIGN"は無視されます。

グローバル宣言

.GLB

書 式　　[△].GLB△＜名前＞[,＜名前＞...]

説 明　　名前で指定のラベル及びシンボルをグローバル属性として宣言します。
指示した名前がファイル内で定義されている場合、外部のファイルから参照できます。
指定した名前がファイル内で定義されていない場合、外部のファイルに定義があるものとします。

記述例

```
.GLB        name1,name2,name3
.GLB        name4
.SECTION    program
MOV.W       #0,name1
```

ビットシンボルグローバル宣言

.BTGLB

書 式　　[△].BTGLB△＜ビットシンボル名＞[, ＜ビットシンボル名＞. . .]

説 明　　名前で指定のビットシンボルをグローバル属性として宣言します。
指示したビットシンボルがファイル内で定義されている場合、外部のファイルから参照できます。
指定したビットシンボルがファイル内で定義されていない場合、外部のファイルに定義があるものとします。

記述例

```
.BTGLB     flag1,flag2,flag3
.BTGLB     flag4
.SECTION   program
BCLR       flag1
```

注意事項　　アセンブル実行時に未確定となるシンボルで定義されたビットシンボルは外部参照指定できません。

ソフトウエア割り込みの設定

.RVECTOR

書 式　[△].RVECTOR△＜ソフトウエア割り込み番号＞,＜ソフトウエア割り込み名＞

説 明　本指示命令により、リンク時に可変ベクタテーブルを自動生成します。可変ベクタテーブルはセクション名"vector"で生成します。
本指示命令で全てソフトウエア割り込み番号を設定しなくても、可変ベクタテーブルの全領域(256 バイト)を生成します。
ソフトウエア割り込み番号は、0 から 63 の範囲でアセンブル時に確定する値を記述します。
ソフトウエア割り込み名は、シンボルまたはラベルを記述します。
本指示命令によって可変ベクタテーブルを自動生成した場合、リンカが生成するリンケージリストファイル(.map)に可変ベクタテーブル情報が出力されます。

記述例

```
.rvector   21,timerA0
           ;timerA0 をソフトウエア割り込み番号の 21 番に設定します。
```

備 考　可変ページベクタテーブルの中で、本指示命令で設定していないソフトウエア割り込み番号に対応する部分(空き領域)については、以下の優先順位で値を設定します。

(1) リンクオプション"-VECT"で設定した値
(2) グローバルラベル"_dummy_int"の値
(3) グローバルラベル"dummy_int"の値
(4) 上記に該当しない場合、空き領域には"00H"を設定

注意事項　本指示命令を記述し"vector"セクションにプログラムを記述した場合、本指示命令はエラーとなります。("vector"セクションにプログラムを記述しないでください)
本指示命令で指定したソフトウエア割り込み番号は、リンクオプション"-VECTN"で指定することはできません。

スペシャルページの設定

.SVECTOR

書 式　[△].SVECTOR△＜スペシャルページ番号＞,＜スペシャルページ名＞

説 明　本指示命令により、リンク時にスペシャルページベクタテーブルを自動生成します。スペシャルページベクタテーブルはセクション名"svector"で生成します。
スペシャルページ番号は、18 から 255 の範囲でアセンブル時に確定する値を記述します。
スペシャルページ名は、シンボルまたはラベルを記述します。
本指示命令によってスペシャルページベクタテーブルを自動生成した場合、リンカが生成するリンケージリストファイル(.map)にスペシャルページベクタテーブル情報が出力されます。

記述例

```
.svector   250,spFunc
           ; __spFunc をスペシャルページ番号の 250 番に設定します。
```

備 考　リンク時、セクション名"svector"は、スペシャルページ番号18から指定されたスペシャルページ番号の一番大きい番号までの領域を確保されます。
自動生成を行った結果、スペシャルページベクタテーブルに空き領域(本指示命令で指定されていないスペシャルページ番号)が存在する場合は FFH が埋め込まれます。

注意事項　"-R8C"オプションが指定されていない場合は、本指示命令は設定できません。
本指示命令を記述し"svector"セクションにプログラムを記述した場合、本指示命令はエラーとなります。
("svector"セクションにプログラムを記述しないでください)

スペシャルページ番号 18 から特定の領域に対して空き領域を設定する場合、指示命令".RESERVE_AREA"を設定します。
下記記述例では、スペシャルページ番号 18 および 19 の領域が空き領域になります。

記述例)

```
.reserve_area     0fffd8h,4
```

セクション名の仮定義

.INITSCT

書 式　[△].INITSCT△＜セクション名＞,＜セクションタイプ＞,align
[△].INITSCT△＜セクション名＞,＜セクションタイプ＞,noalign

説 明　セクション名を仮定義します。
C 言語スタートアップ専用の指示命令です。

記述例

```
.initsct   bss_NE,data,align     ;アライメント補正あり
.initsct   bss_NO,data,noalign   ;アライメント補正なし
```

注意事項　本指示命令は、C 言語スタートアップ(initsct.c)の初期化関数で生成される指示命令であり、コンパイラ専用の指示命令です。

9.4 リスト制御指示命令

リストファイルに出力する情報や、リストファイルのフォーマットの制御を行います。コード生成には影響しません。

表 9.5 リンク制御指示命令

指示命令	機能内容
.LIST	リストファイルを生成する際に、アセンブラソースファイルの行単位でリストファイルへの出力を制御します。
.PAGE	リストファイルを生成する際に、アセンブラソースファイルの任意の位置でリストを改ページします。同時に任意のメッセージをヘッダ部分に出力します。
.FORM	リストファイルの 1 ページに出力する行数及び桁数を設定します。

リスト出力

.LIST

書 式　　[△].LIST△ON|OFF

説 明　　アセンブラリストファイルへの行の出力を制御します。
行の出力を停止する場合は、".LIST OFF"を記述します。行の出力を再開する場合は、".LIST ON"を記述します。
リストへの行の出力を停止している範囲においても、エラー発生行についてはリストファイルに出力します。
本指示命令を指定しない場合は、全ての行をリストファイルに出力します。

記述例

```
.LIST   ON
.LIST   OFF
```

リストファイル改ページ出力

.PAGE

書 式　　[△].PAGE△[″＜文字列＞″]
[△].PAGE△['＜文字列＞']

説 明　　アセンブラリストファイルを改ページします。
オペランドを指定した場合、その文字列を改ページしたヘッダの″SOURCE LIST″部分に置き換えて出力します。
オペランドは、シングルクォーテーション(')又はダブルクォーテーション(″)で囲って記述してください。
オペランドを省略した場合は、デフォルト文字列(SOURCE LIST)または直前に.PAGE で指定した文字列を出力します。

記述例

```
.PAGE
.PAGE   ″strings″
.PAGE   'strings'
```

注意事項　　ヘッダに出力できる最大文字数は 135 文字です。ただし、指示命令″.FORM″でリストファイルの桁数を指定した場合は、″リストファイルの桁数-65 文字″です。

リストファイルの行数と桁数を指定

.FORM

書 式　　[△].FORM△[＜行数＞][,＜桁数＞]

説 明　　アセンブラリストファイルの1ページの行数を 20～255 行の範囲で指定します。
アセンブラリストファイルの1ページの桁数を 80～295 桁の範囲で指定します。
本指示命令を記述した次のページから記述内容が有効になります。ただし、本指示命令をアセンブラソースファイルの1行目に記述した場合は、1ページ目から、指定内容が有効になります。
本指示命令を指定しない場合は、66 行、200 桁で出力します。

記述例

```
.FORM   20,80
.FORM   60
.FORM   ,100
.FORM   line,culum
```

備 考　　1つのアセンブラ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。
行数及び桁数には式を記述できます。
オペランドに桁数のみを指定する場合は、数値の直前に必ずカンマ(,)を記述してください。

注意事項　　前方参照となるシンボルは記述できません。

9.5 条件アセンブル制御指示命令

条件アセンブル制御指示命令を使って、指定した範囲の行のアセンブルを行うか、行わないかを指定できます。

表 9.6 条件アセンブル制御指示命令

指示命令	機能内容
.IF	条件アセンブルブロックの始まりを示します。条件の判定を行います。
.ELIF	二つ以上の条件ブロックを記述する場合に、二つ目以降の条件を判定します。
.ELSE	全ての条件が偽である場合に、アセンブルを行うブロックの始まりを示します。
.ENDIF	条件アセンブルブロックの終了を示します。

条件アセンブル命令

.IF

書 式　　[△].IF△＜条件式＞
[△] [ボディ]
[△].ENDIF

説 明　　条件アセンブルブロックの始まりを示します。
オペランドに記述した条件を判定し、真であれば以降に続くボディをアセンブルします。
条件が真である場合にアセンブルされる行は、指示命令".ELIF",".ELSE"及び".ENDIF"行の前までです。
条件アセンブルブロック内には、as30 のソースプログラムに記述可能な全ての命令を記述できます。

条件式の記述例

```
sym<1
sym < 1
sym+2 < data1
sym+2 < data1+2
'smp1'==name
```

記述例

```
.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

条件式は、指示命令のオペランドに一つだけ記述できます。
条件式には、必ず条件演算子を記述してください。
次に示す演算子が記述できます。

表 9.7 .IF および.ELIF 指示命令の条件演算子

条件演算子	内容
＞	左辺値が右辺値より大きい場合に真となります
＜	左辺値が右辺値より小さい場合に真となります
＞＝	左辺値が右辺値より大きいか等しい場合に真となります
＜＝	左辺値が右辺値より小さいか等しい場合に真となります
＝＝	左辺値と右辺値が等しい場合に真となります
！＝	左辺値と右辺値が等しくない場合に真となります

条件式の演算は符号付き 32 ビットで演算します。
条件演算子の左辺及び右辺には、シンボルが記述できます。
条件演算子の左辺及び右辺には、式が記述できます。式は、「プログラムの記述規則」の「式の記述規則」に従って記述してください。

条件演算子の左辺及び右辺には、文字列が記述できます。文字列は、必ずシングルクォーテーション(')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。このとき、文字列の大小は、文字コードの値で判定されます。

"ABC" < "CBA" → 414243H < 434241H で真となります。
"C" < "A" → 43H < 41H で偽となります。

条件演算子の前後には、スペースまたはタブが記述できます。
条件式は、指示命令".IF"及び".ELIF"のオペランドに記述できます。

注意事項　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として式を判定します。その際、エラーメッセージは出力されません。
条件演算子の左辺及び右辺の式は、基本的にアセンブル実行時に確定する値を設定してください。

.ELIF

書 式　[△].IF△＜条件式＞
[△][ボディ]
[△].ELIF△＜条件式＞
[△][ボディ]
[△].ENDIF

説 明　複数の条件で条件アセンブルを行いたい場合に、".IF"と組み合わせて条件を記述します。
オペランドに記述した条件を判定し、真であれば以降に続くボディをアセンブルします。
条件が真である場合にアセンブルされる行は、指示命令".ELIF",".ELSE"及び".ENDIF"行の前までです。

記述例

```
.IF    TYPE==0
    .byte   "Proto Type Mode"
.ELIF  TYPE>0
    .byte   "Mass Production Mode"
.ELSE
    .byte   "Debug Mode"
.ENDIF
```

備 考　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。

条件アセンブル命令

.ELSE

書 式　　[△].IF△＜条件式＞
[△][ボディ]
[△].ELSE
[△][ボディ]
[△].ENDIF

[△].IF△＜条件式＞
[△][ボディ]
[△].ELIF△＜条件式＞
[△][ボディ]
[△].ELSE
[△][ボディ]
[△].ENDIF

説 明　　全ての条件が偽である場合に、アセンブルを実行する行の始まりを示します。
指示命令".ENDIF"の前の行までをアセンブルします。

記述例

```
.IF    TYPE==0
    .byte   "Proto Type Mode"
.ELIF  TYPE>0
    .byte   "Mass Production Mode"
.ELSE
    .byte   "Debug Mode"
.ENDIF
```

備 考　　本指示命令は、条件アセンブルブロック内に 1 回のみ記述できます。
本指示命令にオペランドはありません。

.ENDIF

書 式　[△].IF△＜条件式＞
[△][ボディ]
[△].ENDIF

説 明　条件アセンブルブロックの終了を示します。

記述例

```
.IF    TYPE==0
    .byte   "Proto Type Mode"
.ELIF  TYPE>0
    .byte   "Mass Production Mode"
.ELSE
    .byte   "Debug Mode"
.ENDIF
```

備 考　本指示命令は、条件アセンブルブロックに必ず一つ記述してください。
本指示命令にオペランドはありません。

9.6 マクロ指示命令

マクロ機能および繰り返しマクロ機能を定義するための指示命令です。

表 9.8 マクロ指示命令

指示命令	機能内容
.MACRO	マクロ名を定義します。マクロボディの始まりを定義します。
.ENDM	マクロボディの終了を示します。
.EXITM	マクロボディの展開を中止します。
.LOCAL	マクロ内ローカルラベルを宣言します。
.MREPEAT	繰り返しマクロボディの始まりを示します。
.ENDR	繰り返しマクロボディの終了を示します。
..MACPARA	マクロ呼び出しの実引数の個数を値として持ちます。
..MACREP	繰り返しマクロボディの展開回数を値として持ちます。
.LEN	指定した文字列の文字数を値として持ちます。
.INSTR	指定した文字列の中で指定した文字列の始まる位置を値として持ちます。
.SUBSTR	指定した文字列の中で指定した位置から指定した文字数分の文字を切り出します。

マクロ定義

.MACRO

書 式　　マクロ定義
[△]＜マクロ名＞△.MACRO△[＜仮引数＞[,＜仮引数＞...]]
[△]ボディ
[△].ENDM

マクロ呼び出し
[△]＜マクロ名＞△[＜実引数＞[,＜実引数＞...]]

説 明　　マクロ名を定義します。
マクロ名の記述は、「プログラムの記述規則」の「名前の記述規則」に従ってください。
マクロ定義の始まりを示します。

マクロ定義記述例

```
mac   .MACRO p1,p2,p3
      .IF    ..MACPARA == 3
        .IF 'p1' == 'byte'
            MOV.B   #p2,p3
        .ELSE
            MOV.W   #p2,p3
        .ENDIF
      .ELIF  ..MACPARA == 2
        .IF 'p1' =='byte'
            MOV.B   p2,R0L
        .ELSE
            MOV.W   p2,R0
        .ENDIF
      .ELSE
        MOV.W    R0,R1
      .ENDIF
.ENDM
```

マクロ呼び出し記述例

```
mac   word,10,R0
```

マクロ展開例

```
mac word,10,R0
      .IF 3 == 3
        .IF 'word' == 'byte'
        .ELSE
          MOV.W #10,R0
        .ENDIF
      .ELIF 3 == 2
      .ELSE
      .ENDIF
.ENDM
```

仮引数　　マクロ仮引数の名前の記述は、「プログラムの記述規則」の「名前の記述規則」に従ってください。
マクロ仮引数の名前は、ネストしているマクロ定義を含めて、異なる名前で定義してください。
指示命令".MACRO"のオペランドに記述した仮引数は、必ずマクロボディ内に記述してください。
仮引数は、1行に記述できる文字数の範囲内で80個まで記述できます。

実引数　　実引数は、マクロ呼び出しの際に仮引数に対応させて記述してください。
特殊文字を実引数に記述する場合は、ダブルクォーテーションで囲って記述してください。
実引数には、ラベル、グローバルラベル及びシンボルが記述できます。
実引数には式が記述できます。

実引数の展開

仮引数と実引数は、左から記述されている順に置き換えられます。
仮引数が定義されていて、マクロ呼び出しで実引数の記述が無い場合は、仮引数にあたる部分のコードは出力されません。
仮引数の数が、実引数の数より多い場合は、対応する実引数がない仮引数にあたる部分のコードは出力されません。
ボディに記述した仮引数をシングルクォーテーション(')で囲った場合は、対応する実引数をシングルクォーテーションで囲って出力されます。
1つの実引数がカンマ(,)を含む場合に、括弧(())で囲った場合は、括弧を含めて変換されます。
実引数の数が、仮引数の数より多い場合は、対応する仮引数が無い実引数については処理されません。

実引数の展開例

マクロ定義例)

```
name    .MACRO    string
.BYTE   'string'
.ENDM
```

マクロ呼び出し例 1)

```
name    "name,address"
.BYTE   'name,address'
```

マクロ呼び出し例 2)

```
name    (name,address)
.BYTE   '(name,address)'
```

注意事項　　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。
実引数と仮引数の数が合わない場合は、as30 はウォーニングメッセージを出力します。

マクロ定義の終了

.ENDM

書 式　　[△]＜マクロ名＞△.MACRO
[△]ボディ
[△].ENDM

説 明　　一つのマクロ定義のボディが終了する事を示します。

記述例

```
lda  .MACRO  value
     MOV.W   #value,A0
.ENDM

lda  0
     MOV.W   #0,A0
```

備 考　　必ず、指示命令".MACRO"に対応させて記述してください。

マクロ展開の中止

.EXITM

書 式　　[△]＜マクロ名＞△.MACRO
[△]ボディ
[△].EXITM
[△]ボディ
[△].ENDM

説 明　　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。

記述例

```
data1  .MACRO   value
     .IF  value == 0
          .EXITM
     .ELSE
          .BLKB  value
     .ENDIF
.ENDM

data1  0
       .IF   0 == 0
             EXITM
       .ENDIF
```

説 明　　マクロ定義のボディ内に記述してください。

マクロ内ローカルラベル宣言

.LOCAL

書 式　　[△].LOCAL△＜ラベル名＞[,＜ラベル名＞...]

説 明　　オペランドに記述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。
マクロローカルラベルは、異なるマクロ定義及びマクロ定義外であれば、同一の名前を複数個記述できます。

記述例

```
name  .MACRO
      .LOCAL  m1  ; 'm1' is macro local label
m1:
      nop
      jmp     m1
.ENDM
```

備 考　　本指示命令は、必ずマクロボディ内に記述してください。
本指示命令によるマクロローカルラベル宣言は、ラベル名を定義するより前に記述してください。
マクロローカルラベル名の記述は、「プログラムの記述規則」の「名前の記述規則」に従ってください。
本指示命令のオペランドは、カンマで区切って複数のラベルを記述できます。このときの最大ラベル数は100個までです。

注意事項　　マクロ定義がネストしている場合は、マクロ定義内で定義を行っているマクロ内のマクロローカルラベルは、同一名を使用できません。
インクルードファイルの内容を含む、一つのアセンブラソースファイルに記述できるマクロローカルラベルは65535 個までです。

繰り返しマクロの開始

.MREPEAT

書 式　　[△][＜ラベル＞:△].MREPEAT△＜数値＞
[△]ボディ
[△].ENDR

説 明　　繰り返しマクロの始まりを示します。
ボディを指定した数値回、繰り返して展開します。
繰り返し回数は、最大 65535 回まで指定できます。
65535 レベルまでのネストができます。
本指示命令を記述した場所に、マクロボディを展開します。

記述例

```
.MREPEAT   3
    nop
.ENDR
```

展開例

```
.MREPEAT   3
    nop
    nop
    nop
.ENDR
```

マクロ定義との組み合わせ例

```
rep  .MACRO num
     .MREPEAT  num
     .IF num > 49
         .EXITM
     .ENDIF
     nop
     .ENDR
.ENDM

rep  3
     nop
     nop
     nop
```

備 考　　オペランドには、シンボルを記述できます。
オペランドには、式が記述できます。
ボディ内に指示命令".EXITM"を記述できます。

注意事項　　前方参照となるシンボルは記述できません。

繰り返しマクロ終了

.ENDR

書 式　　[△][＜ラベル＞:△].MREPEAT△＜数値＞
[△]ボディ
[△].ENDR

説 明　　繰り返しマクロの終了を示します。

記述例

```
.MREPEAT   3
    nop
.ENDR
```

実引数の数に置き換え

..MACPARA

書 式　　[△]..MACPARA

説 明　　マクロ呼び出しの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記述できます。
本指示命令は式の項として記述できます。

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。

```
          .GLB mem
    name  .MACRO  f1,f2
          .IF       ..MACPARA == 2
            ADD.W  f1,f2
          .ELSE
            ADD.W  R0,f1
          .ENDIF
    .ENDM

    name  mem
          .ELSE
            ADD.W  R0,mem
          .ENDIF
```

注意事項　　″.MACRO″によるマクロボディの外に記述した場合は、値は0になります。その際、エラーメッセージは出力されません。

現在のマクロの繰り返し回数に置き換え

..MACREP

書 式　　[△]..MACREP

説 明　　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記述できます。
条件アセンブルのオペランドに記述できます。
本指示命令は式の項として記述できます。

記述例

```
.MREPEAT   3
   MOV.W   R0,..MACREP
.ENDR

   MOV.W   R0,1
   MOV.W   R0,2
   MOV.W   R0,3

.GLB   mem
mclr   .MACRO   value,name
.MREPEAT   value
   MOV.W   #0,name+..MACREP
.ENDR
.ENDM

mclr       3,mem
.MREPEAT   3
   MOV.W   #0,mem+1
   MOV.W   #0,mem+2
   MOV.W   #0,mem+3
.ENDR
```

注意事項　　″.MACRO″によるマクロボディの外に記述した場合は、値は0になります。その際、エラーメッセージは出力されません。

指定文字列の長さに置き換え

.LEN

書 式　　[△].LEN△{"＜文字列＞"}
[△].LEN△{'＜文字列＞'}

説 明　　オペランドに記述した文字列の文字列長を示します。
オペランドは、必ず{}で囲ってください。
文字列には、スペース及びタブを含む、7ビットアスキーコードの文字が記述できます。
文字列は、必ずクォーテーションで囲って記述してください。
本指示命令を式の項に記述できます。

記述例

```
        .byte    .LEN{"string"}
```

展開例

```
        .byte    6
```

マクロ定義との組み合わせ例 1

```
        bufset   .MACRO    f1,f2
            buffer@f1:    .BLKB .LEN{'f2'}
        .ENDM

        bufset   1,Printout_data
            buffer1    .BLKB   13

        bufset   2,Sample
            buffer2    .BLKB   6
```

マクロ定義との組み合わせ例 2

```
        buf      .MACRO    f1
            buffer: .BLKB    .LEN{"f1"}
        .ENDM

        buf    data              ; data は展開されません
            buffer:   .BLKB    2
```

注意事項　　漢字などの8ビットコードについては、正しく処理されませんが、as30 はエラーの検出を行いません。
マクロの引数を文字列として展開したい場合は、引数名をシングルクォーテーションで囲って記述してください。
ダブルクォーテーションで囲った場合はマクロ定義で記述した仮引数の文字列の長さになります。

文字列の開始位置に置き換え

.INSTR

書 式　[△].INSTR△{″＜文字列＞″,″＜検出文字列＞″,＜検出開始位置＞}
[△].INSTR△{'＜文字列＞','＜検出文字列＞',＜検出開始位置＞}

説 明　オペランドで指定した文字列のなかで、検出文字列が始まる位置を示します。
文字列の検索を開始する位置を指定できます。
オペランドは、必ず{ }で囲ってください。
検索開始位置は、シンボルを記述できます。
検索開始位置を1とした場合は、文字列の先頭を示します。
文字列には、スペース及びタブを含む、7ビットアスキーコードの文字が記述できます。
文字列は、必ずクォーテーションで囲って記述してください。
本指示命令は、式の項に記述できます。

記述例

指定した文字列(japanese)の先頭(top)からの、″se″文字列の位置(7)を取り出します。

```
point_set  .MACRO  source,dest,top
     point  .EQU    .INSTR{'source','dest',top}
.ENDM

point_set  japanese,se,1
     point  .EQU    7
```

注意事項　文字列よりも、検索文字列が長い場合の値は0となります。文字列のなかに、検索文字列が含まれていなかった場合の値は0となります。文字列の長さよりも、検索開始位置の値が大きかった場合の値は0となります。
漢字などの8ビットコードについては、正しく処理されませんが、as30 はエラーの検出を行いません。
マクロの引数を文字列として展開したい場合は、引数名をシングルクォーテーションで囲って記述してください。
ダブルクォーテーションで囲って記述した文字列は文字列そのものが展開されます。

文字列の切り出し

.SUBSTR

書 式　[△].SUBSTR△{"＜文字列＞","＜切り出し開始位置＞",＜切り出し文字数＞}
[△].SUBSTR△{'＜文字列＞','＜切り出し開始位置＞',＜切り出し文字数＞}

説 明　文字列の指定した位置から、指定した文字数を取り出します。
オペランドは、必ず{}で囲ってください。
切り出し開始位置及び切り出し文字数は、シンボルが記述できます。
切り出し開始位置を1とした場合は、文字列の先頭を示します。
文字列には、スペース及びタブを含む、7ビットアスキーコードの文字が記述できます。
文字列は、必ずクォーテーションで囲って記述してください。

記述例

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ます。
"..MACREP"は、". BYTE"の行を実行する毎に、1→2→3→4 と増加します。したがって、マクロの実引数として与えられた文字列の先頭の文字から順に 1 文字ずつ、".BYTE"のオペランドに与えることになります。

```
    name    .MACRO    data
            .MREPEAT  .LEN{'data'}
            .BYTE     .SUBSTR{'data',..MACREP,1}
            .ENDR
    .ENDM

    name    ABCD
            .BYTE     "A"
            .BYTE     "B"
            .BYTE     "C"
            .BYTE     "D"
```

注意事項　文字列の長さよりも切り出し開始位置の値が大きい場合の値は0となります。文字列の長さよりも切り出し文字数の値が大きい場合の値は0となります。切り出し文字数を0とした場合の値は0となります。
漢字などの8ビットコードについては、正しく処理されませんが、as30 はエラーの検出を行いません。
マクロの引数を文字列として展開したい場合は、引数名をシングルクォーテーションで囲って記述してください。
ダブルクォーテーションで囲って記述した文字列は文字列そのものが展開されます。

9.7 インスペクタ情報出力制御指示命令

インスペクタ情報の出力を制御する指示命令です。

表 9.9 インスペクタ情報出力制御指指示命令

指示命令	機能内容
.INSF	インスペクタ情報の関数(サブルーチン)開始情報を定義します。
.EINSF	インスペクタ情報の関数(サブルーチン)終了情報を定義します。
.CALL .CALLIND	インスペクタ情報の関数(サブルーチン)呼び出し先情報を定義します。
.STK	インスペクタ情報のスタック情報を定義します。

インスペクタ情報の関数開始を定義

.INSF

書 式　　[△].INSF△＜関数(サブルーチン)開始ラベル名＞,＜記憶クラス＞,＜フレームサイズ＞

説 明　　インスペクタ情報の関数(サブルーチン)開始情報を定義します。
関数(サブルーチン)開始情報から、指示命令".EINSF"までを1つの関数(サブルーチン)情報として定義します。

記述例

```
.INSF  glbfunc,G,0
glbfunc:
        ;
.EINSF

.INSF  locfunc,S,0
locfunc:
        ;
.EINSF
```

備 考　　記憶クラスは"G(グルーバルラベル)"、"S(ローカルラベル)"のいずれかを記述してください。
フレームサイズは整数値を記述してください。

注意事項　　本指示命令を記述した場合、必ず指示命令 ".EINSF"を記述してください。
本指示命令はアセンブリ言語記述専用の指示命令であり、NC30 の asm 関数にて本指示命令を記述した場合、エラーになります。
本指示命令は、コマンドオプション"-finfo" が指定された場合に有効となります。
関数(サブルーチン)開始ラベル名には、必ずアセンブラファイルに定義されているラベルを記述してください。

インスペクタ情報の関数終了を定義

.EINSF

書 式　　[△].EINSF

説 明　　インスペクタ情報の関数(サブルーチン)終了情報を定義します。
″.INSF″から、関数(サブルーチン)終了情報までを1つの関数(サブルーチン)情報として定義します。

記述例

```
    .INSF   glbfunc,G,0
    glbfunc:
            ;
    .EINSF
```

備 考　　本指示命令を記述した場合、必ず指示命令″.INSF″を記述してください。
本指示命令はアセンブリ言語記述専用の指示命令であり、NC30 の asm 関数にて本指示命令を記述した場合、エラーとなります。
本指示命令は、コマンドオプション″-finfo″が指定された場合に有効となります。

インスペクタ情報の関数呼び出し先情報を定義

.CALL、.CALLIND

書 式　[△].CALL△＜呼び出し先関数(サブルーチン)名＞,＜記憶クラス＞
[△].CALLIND

説 明　インスペクタ情報の関数(サブルーチン)呼び出し先情報を定義します。

記述例

```
.INSF  glbfunc,G,0
       ;
       .CALL   glbsub,G
       jsr     glbsub
       ;
       .CALL   locsub,S
       jsr     locsub
               ;
       .CALLIND
       jsri.w  extFunc
       ;
.EINSF
```

備 考　本指示命令で設定された関数(サブルーチン)呼び出し先情報は、最適化リンケージエディタが出力するスタック使用量情報ファイルに出力され、CallWalker で参照可能になります。
記憶クラスは"G(グローバルラベル)"、"S(ローカルラベル)"のいずれかを記述してください。
".CALL"または".CALLIND"の定義は、直後記述する分岐命令またはサブルーチン呼び出し命令により決定します。

指示命令	関数の呼び出し	CallWalker でのシンボル区分
.CALL	jmp、jsr、jmps、jsrs	C/C++の関数として表示
.CALLIND	jmpi、jsri	アドレス参照未解決関数

注意事項　本指示命令は、インスペクタ情報の関数開始情報と関数終了情報の範囲内で記述してください。
本指示命令は、コマンドオプション"-finfo"が指定された場合に有効となります。
本指示命令は、必ず分岐命令またはサブルーチン呼び出し命令の直前に記述してください。
".CALL"の呼び出し先関数(サブルーチン)名には、必ず定義または参照しているシンボル(ラベル)を記述してください。
本指示命令は、関数(サブルーチン)呼び出し先情報のみ設定します。したがいまして、関数(サブルーチン)呼び出しに必要なスタックサイズは、指示命令".INSF"のフレームサイズまたは指示命令".STK"のスタックサイズに設定してください。

インスペクタ情報のスタック情報定義

.STK

書 式　　[△].STK△＜スタックサイズ＞

説 明　　インスペクタ情報のスタック情報を定義します。

記述例

```
.INSF glbfunc, G, 0
      ;
      .STK    2     ;2 byte push
      PUSH.W  R0    ;
      ;
      .STK    -2    ;2 byte pop
      POP.W   R0    ;
      ;
.EINSF
```

備 考　　スタックサイズは整数値を記述してください。

注意事項　　本指示命令は、インスペクタ情報の関数開始情報と関数終了情報の範囲内で記述してください。
本指示命令は、コマンドオプション"-finfo"が指定された場合に有効となります。

9.8 拡張機能指示命令

拡張機能指示命令には、コード生成に影響を与える指示命令と与えない指示命令があります。

表 9.10 コード生成に影響を与える拡張機能指示命令

指示命令	機能内容
.ID	ID コードを設定します。
.OFSREG	オプション機能選択レジスタ値を設定します。(R8C ファミリ向け)
.PROTECT	ROM コードプロテクト値を設定します。(M16C シリーズ向け)
.RESERVE_AREA	設定された領域を予約扱いとします。

表 9.11 コード生成に影響を与えない拡張機能指示命令

指示命令	機能内容
.ASSERT	オペランドに記述した文字列を標準エラー出力またはファイルに出力します。
?	テンポラリラベルの定義と参照を指定します。
..FILE	as30 が処理を行っているアセンブラソースファイル名を示します。
@	@の前後の文字列を連結し、一つの文字列として扱います。
.DEFINE	シンボルに文字列を定義します。

ID コードチェック機能の ID コードを設定

.ID

書 式　　[△].ID△″ ＜ID コード文字列＞″
[△].ID△″ ＃＜ID コード値＞″

説 明　　指定した ID コードは、下表に示す ID コード格納番地に各 8 ビットデータで格納します。

オプション	IDコード格納番地
設定なし	FFFDFH、FFFE3H、FFFEBH、FFFEFH、FFFF3H、FFFF7H、FFFFBH
-R8C/-R8CE	FFDFH、 FFE3H、 FFEBH、 FFEFH、 FFF3H、 FFF7H、 FFFBH

ID コードは文字列または値で指定します。文字列で指定の場合は、アスキーコードに変換して格納しますので、7 文字以内で指定します。
値で指定の場合は、先頭に″#″をつけ、16 進数で 14 桁以内の値を指定します。
指定した ID コードは、アブソリュートファイル(.abs)、モトローラ S 形式ファイル(.mot)、インテル HEX 形式ファイル(.hex)およびバイナリ形式ファイル(.bin)へ出力します。また、リンケージリストファイル(.map)や ID ファイル(.id)へも出力するため、これらのファイルで設定値の確認ができます。
指示命令″.ID″を記述し、指示命令″.OFSREG″(または″.PROTECT″)を記述していない場合、
″.OFSREG　0FFH″(または″.PROTECT　0FFH″)を指定したものとします。このため、オプション機能選択レジスタ(または ROM コードプロテクト制御番地)の設定値は下表のようになります。

指示命令".ID"	指示命令".OFSREG"（または".PROTECT"）	設定値
指定	指定	指示命令".OFSREG"（または".PROTECT"）の設定値
指定	指定なし	0FFH
指定なし	指定	指示命令".OFSREG"（または".PROTECT"）の設定値
指定なし	指定なし	ソースプログラムに記述された値

数値指定による記述例)

a) ID コードに 14 桁の値を設定した場合

```
.id   "#11223344556677"
```

IDコード格納番地	0FFFDFH	0FFFE3H	0FFFEBH	0FFFEFH
IDコード	11H	22H	33H	44H
IDコード格納番地	0FFFF3H	0FFFF7H	0FFFFBH	
IDコード	55H	66H	77H	

b) ID コードに 14 桁に満たない値を設定した場合

```
.id   "#1234567"
      ; ID コード 12H、34H、56H、70H、00H、00H、00H を設定
```

c) ID コードの値を省略した場合

```
.id   "#"
      ; ID コード 00H、00H、00H、00H、00H、00H、00H を設定
```

文字列指定による記述例）

a） ID コードに 7 文字の文字列を設定した場合

```
.id  "smpcode"
```

IDコード格納番地	0FFFDFH	0FFFE3H	0FFFEBH	0FFFEFH
IDコード	73H	6DH	70H	63H
IDコード格納番地	0FFFF3H	0FFFF7H	0FFFFBH	
IDコード	6FH	64H	65H	

b） ID コードに 7 文字に満たない文字列を設定した場合

```
.id  "smp"
      ; ID コード 73H、6DH、70H、00H、00H、00H、00H を設定
```

c） ID コードの文字列を省略した場合

```
.id  ""
      ; ID コード FFH、FFH、FFH、FFH、FFH、FFH、FFH を設定
```

注意事項　ID コードチェック機能の詳細については、該当するマイコンのハードウェアマニュアルを参照してください。
複数のアセンブラソースファイルに本指示命令を記述した場合、optlnk でウォーニングになります。（最初にリンクされたオブジェクトファイルの値になります。）
本指示命令と指示命令".OFSREG"（または".PROTECT"）を設定する場合は、必ず同じソースファイル内で設定してください。

オプション機能選択レジスタに値を設定

.OFSREG

書 式　[△].OFSREG△＜数値＞

説 明　R8C ファミリのオプション機能選択レジスタ(0FFFFH 番地)に指定した値を格納します。値は 0～0FFH の範囲で指定します。本指示命令使用時は、-R8C または-R8CE オプションを指定してください。
指定した値は、アブソリュートファイル(.abs)、モトローラS 形式ファイル(.mot)、インテル HEX 形式ファイル(.hex)およびバイナリ形式ファイル(.bin)へ出力します。また、リンケージリストファイル(.map)や ID ファイル(.id)へも出力するため、これらのファイルで設定値の確認ができます。

記述例

```
; fixed vector section
;------------------------------------------------------------------
     .org      0FFFCh
RESET:
     .lword    start
     .ofsreg   0FFH        ;オプション機能選択レジスタに 0FFH を設定します。
```

注意事項　オプション機能選択レジスタの詳細については、該当するマイコンのハードウェアマニュアルを参照してください。
複数のアセンブラソースファイルに本指示命令を記述した場合、optlnk でウォーニングになります。(最初にリンクされたオブジェクトファイルの値になります。)
"-R8C"および"-R8CE"オプションが指定されていない場合は、指示命令".PROTECT"として処理します。
本指示命令と指示命令".ID"を設定する場合は、必ず同じソースファイル内で設定してください。
指示命令".ID"を記述し、本指示命令を記述しなかった場合、オプション機能選択レジスタの設定値は 0FFH になります。

ROM コードプロテクト制御番地に値を設定

.PROTECT

書 式　[△].PROTECT△＜数値＞

説 明　M16C シリーズの ROM コードプロテクト制御番地（0FFFFFH 番地）に指定した値を格納します。値は 0～0FFH の範囲で指定します。本指示命令使用時は、-R8C や-R8CE オプションは指定しないでください。
指定した値は、アブソリュートファイル(.abs)、モトローラS 形式ファイル(.mot)、インテル HEX 形式ファイル(.hex)およびバイナリ形式ファイル(.bin)へ出力します。また、リンケージリストファイル(.map)や ID ファイル(.id)へも出力するため、これらのファイルで設定値の確認ができます。

記述例

```
; fixed vector section
;------------------------------------------------------------------
     .org      0FFFFCh
RESET:
     .lword    start
     .protect  0FFH        ; ROM コードプロテクト制御番地に 0FFH を設定します。
```

注意事項　ROM コードプロテクト機能の詳細については、該当するマイコンのハードウェアマニュアルを参照してください。
複数のアセンブラソースファイルに本指示命令を記述した場合、optlnk でウォーニングになります。（最初にリンクされたオブジェクトファイルの値になります。）
″-R8C″または″-R8CE″オプションが指定されている場合は、指示命令″.OFSREG″として処理します。
本指示命令と指示命令″.ID″を設定する場合は、必ず同じソースファイル内で設定してください。
指示命令″.ID″を記述し、本指示命令を記述しなかった場合、ROMコードプロテクト制御番地の設定値は0FFHになります。

予約領域を設定

.RESERVE_AREA

書 式　　[△].RESERVE_AREA△＜予約領域開始アドレス＞, ＜予約領域サイズ＞

説 明　　設定された予約領域開始アドレスから予約領域サイズを予約領域とします。
予約領域には、プログラムを配置することができません。
オペランドに定義できる値は、アセンブル実行時に確定しなければなりません。
オペランドに設定できる値の範囲は、使用する CPU の最大アドレスです。

記述例

```
.RESERVE_AREA     0fffd8h,4
;0fffd8h 番地から 4 バイトが予約領域になります。
```

指定文字列を出力

.ASSERT

書 式　[△].ASSERT△"＜文字列＞"
[△].ASSERT△"＜文字列＞"△＞△＜ファイル名＞
[△].ASSERT△"＜文字列＞"△＞＞△＜ファイル名＞

説 明　オペランドに記述した文字列をアセンブル実行時に、標準エラー出力に出力します。
ファイル名を指定した場合は、オペランドに記述した文字列をファイルに出力します。
ファイル名に絶対パスを記述した場合は、記述したディレクトリにファイルを生成します。
ファイル名に絶対パスを記述していない場合
(1) as30起動時にコマンド行で指定したファイルにディレクトリ指定がない場合は、本指示命令で指定されたファイルをカレントディレクトリに生成します。
(2) as30起動時にコマンド行で指定したファイルにディレクトリ指定がある場合は、本指示命令で指定されたファイル名にコマンド行で指定されたファイルのディレクトリを付加したファイルを生成します。
ファイル名に指示命令"..FILE"を記述した場合は、as30 起動時にコマンド行で指定したファイルと同じディレクトリにファイルを生成します。

記述例

sample.dat ファイルにメッセージを出力します。

```
.ASSERT    "string" > sample.dat
```

sample.dat ファイルにメッセージを追加します。

```
.ASSERT    "string" >> sample.dat
```

現在処理中のファイルと同じ名前で拡張子を除くファイル名のファイルにメッセージを出力します。

```
.ASSERT    "string" > ..FILE
```

備 考　オペランドと指示命令の間には、スペースまたはタブを記述してください。
オペランドの文字列は必ずダブルクォーテーションで囲ってください。
文字列をファイルに出力するときは、"＞"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ある場合は、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。
"＞"または"＞＞"の前後には、スペースまたはタブを記述できます。
ファイル名に指示命令"..FILE"を記述できます。

テンポラリラベル

?

書 式　　[△]?:
＜ニーモニック＞△？＋
＜ニーモニック＞△？－

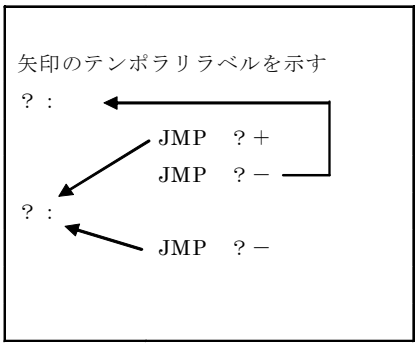
説 明　　テンポラリラベルを定義します。
直前または直後に定義されたテンポラリラベルを参照します。
同一ファイル内で定義及び参照が可能です。
ファイル内に 65535 個までのテンポラリラベルが定義できます。このとき、ファイル内に".INCLUDE"が記述されている場合は、インクルードファイル内のテンポラリラベルを含み 65535 個までの記述ができます(tl0001～tlFFFF)。
リストファイルには、テンポラリラベルとして変換された結果が出力されます。

記述例

```
?:
    JMP     ?+
    JMP     ?-
?:
    JMP     ?-
```

備 考　　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"?-"を記述してください。
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。

注意事項　　参照できるラベルは、直前または直後のラベルのみです。



ソースファイル名情報に置き換え

..FILE

書 式　[△]..FILE

説 明　as30 が処理中のファイル名に展開されます(アセンブラソースファイルまたはインクルードファイル)。

記述例

〈sample.a30 ファイル〉

```
    .INCLUDE   incfile.inc
    .INCLUDE   ..FILE@.inc              ・・・（1）
    .ASSERT    "comment" > ..FILE       ・・・（2）
```

〈incfile.inc ファイル〉

```
    .INCLUDE   ..FILE                   ・・・（3）
    .ASSERT    "comment" > ..FILE@.mes  ・・・（4）
```

上記例では、次のように展開します。
(1).INCLUDE sample.inc
(2).ASSERT "comment" > sample
(3).INCLUDE incfile
(4).ASSERT "comment" > incfile.mes

上記例で"-F"オプションを指定した場合、(3)、(4)の ..FILE の展開が incfile から sample に変わります。
(1).INCLUDE sample.inc
(2).ASSERT "comment" > sample
(3).INCLUDE sample
(4).ASSERT "comment" > sample.mes

備 考　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。

注意事項　本指示命令で読み込まれるファイル名は、ファイルの拡張子及びパスを除いた部分です。
コマンドオプション"-F"指定すると、"..FILE"は、コマンド行で指定したアセンブラ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を示します。

文字列の連結

@

書 式　　＜文字列＞@＜文字列＞
＜文字列＞@＜文字列＞[@＜文字列＞...]

説 明　　マクロ引数、マクロ変数、予約シンボル、指示命令"..FILE"の展開ファイル名及び指定文字列を連結します。

記述例

ファイル名の連結例)
現在処理中のファイル名が sample1.a30 の場合、sample.dat ファイルにメッセージを出力します。

```
    .ASEERT    "sample"  >  ..FILE@.dat
```

文字列の連結例)

```
    mov_nibble .MACRO   p1,src,p2,dest
         MOV@p1@p2      src,dest
    .ENDM

    mov_nibble   L,R0L,H,[A0]
         MOVLH    R0L,[A0]
```

備 考　　本指示命令の前後に記述したスペース及びタブは、文字列として連結します。
本指示命令の前後には、文字列が記述できます。
@を文字データ(40H)として記述する場合は。"(ダブルクォーテーション)で囲んでください。
@を含む文字列をシングルクォーテーションで囲った場合は、@の前後の文字列を連結します。
一行に複数回記述できます。

注意事項　　連結した文字列を名前とする場合は、本指示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。

文字列シンボル定義

.DEFINE

書 式　　[△]＜シンボル名＞△.DEFINE△＜文字列＞
[△]＜シンボル名＞△.DEFINE△'＜文字列＞'
[△]＜シンボル名＞△.DEFINE△"＜文字列＞"

説 明　　シンボルに文字列を定義します。
シンボルは再定義が可能です。

記述例

```
        .SECTION    ram,DATA
data1:  .BLKB       1
flag    .DEFINE     "01H,data1"

        .SECTION    program
        CLB         flag
```

備 考　　スペースまたはタブを含む文字列を定義する場合は、必ずシングルクォーテーション(')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。

注意事項　　本指示命令で定義されたシンボルは、外部参照指定ができません。

10. 構造化記述文

as30 は、次に示す9種類の構造化記述文をサポートしています。

(1) 代入文
右辺を左辺に代入します。

(2) IF ELIF ELSE ENDIF 文(以降 IF 文と記す)
IF文は制御の流れを2方向に変える命令で、分岐する方向は条件式によって決定されます。

(3) FOR NEXT 文(以降 FOR-NEXT 文と記す)
FOR-NEXT文は繰り返しを制御する命令で、指定した条件式が真である間、文を繰り返し実行します。

(4) FOR TO STEP NEXT 文(以降 FOR-STEP 文と記す)
FOR-STEP文は初期値、増分と最終値を指定することで繰り返し回数を制御する命令です。

(5) DO WHILE 文(以降 DO 文と記す)
DO文は条件式が満たされている(真である)間、文を繰り返し実行します。

(6) SWITCH CASE DEFAULT ENDS 文(以降 SWITCH 文と記す)
SWITCH文は条件式の値によっていずれかのCASEブロックに分岐します。

(7) BREAK 文
BREAK文は該当するFOR文、DO文またはSWITCH文の実行を中止して、その次に実行する文に分岐します。

(8) CONTINUE 文
CONTINUE文はそれを含む最小の繰り返しのFOR文、DO文中の繰り返しの判断を行う文に分岐します。

(9) FOREVER 文
FOREVER文は該当するFOR文およびDO文の条件式を常に真であると仮定して制御ブロックを繰り返し実行します。

10.1 変数

as30 の構造化記述では、マイクロコンピュータのレジスタやメモリを変数と言います。変数には次の種類があります。

(1) レジスタ変数

M16Cシリーズ、R8Cファミリが持っているレジスタを示します。

(2) フラグ変数

M16Cシリーズ、R8Cファミリのマイクロコンピュータが持っているフラグレジスタの中の各種フラグを示します。

(3) レジスタビット変数

レジスタ変数の各ビット位置を示します。

(4) メモリ変数

任意のラベルまたはシンボルを示します。

(5) メモリビット変数

任意のビットシンボルを示します。

(6) 予約変数

as30の構造化記述では、レジスタ変数、フラグ変数およびレジスタビット変数を予約変数名として処理します。したがって、これらの変数に使用されている名前は、メモリ変数名やシンボル名などに使用できません。レジスタおよびフラグの機能の詳細については、M16Cシリーズ、R8Cファミリの各ソフトウェアマニュアルを参照してください。

10.2 レジスタ変数

以下にレジスタ変数の一覧を示します。as30 はレジスタ変数名の大文字と小文字を区別しません。したがって、"R0L"と"r0l"は同じレジスタ変数を示します。

表 10.1 レジスタ変数

変数名	レジスタ名	変数型名
R0L、R0H、R1L、R1H	データレジスタ	バイト型
R0、R1、R2、R3	データレジスタ	ワード型
A0.B、A1.B	アドレスレジスタ	バイト型
A0、A0.W、A1、A1.W	アドレスレジスタ	ワード型
[A0.B]、[A1.B]	アドレスレジスタ間接	バイト型
[A0]、[A0.W]、[A1]、[A1.W]	アドレスレジスタ間接	ワード型
[A0.A]、[A1.A]	アドレスレジスタ間接	アドレス型
[A0.L]、[A1.L]	アドレスレジスタ間接	ロングワード型
FB	フレームベースレジスタ	ワード型
PC	プログラムカウンタ	アドレス型
INTBH、INTBL	割り込みテーブルレジスタ	ワード型
INTB	割り込みテーブルレジスタ	アドレス型
SP、ISP	スタックポインタ	ワード型
SB	スタティックベースレジスタ	ワード型
FLG	フラグレジスタ	ワード型
R2R0、R3R1	32 ビットデータレジスタ	ロングワード型
A1A0	32 ビットアドレスレジスタ	ロングワード型
[A1A0.B]	32 ビットアドレスレジスタ間接	バイト型
[A1A0]、[A1A0.W]	32 ビットアドレスレジスタ間接	ワード型
IPL	プロセッサ割り込み優先レベル	---

注意事項　SPは、Uフラグで示すスタックポインタ(ユーザスタックポインタまたは割り込みスタックポインタ)が対象となります。スタックポインタおよびUフラグの機能についての詳細は、M16Cシリーズ、R8Cファミリの各ソフトウェアマニュアルを参照してください。

10.3 スタック変数

以下にスタック変数の一覧を示します。as30 は、変数の大文字と小文字を区別しません。したがって、"STK"と"stk"は同じ変数です。

スタック領域への退避またはスタック領域からの復帰の場合に、スタック変数を記述できます。

表 10.2 スタック変数

スタック変数名	内容
[STK]	スタックポインタで示されるメモリ

注意事項　スタック領域は、Uフラグが"0"のとき割り込みスタックポインタ、Uフラグが"1"のときユーザスタックポインタを示します。

10.4 フラグ変数

以下にフラグ変数の一覧を示します。as30 はフラグ変数名の大文字と小文字を区別しません。したがって、"C"と"c"は同じフラグ変数を示します。フラグ変数の機能については、M16C シリーズ、R8C ファミリの各ソフトウェアマニュアルを参照してください。

表 10.3 フラグ変数

フラグ変数名	フラグ名
C	キャリーフラグ
D	デバッグフラグ
Z	ゼロフラグ
S	サインフラグ
B	レジスタバンク指定フラグ
O	オーバーフローフラグ
I	割り込み許可フラグ
U	スタックポインタ指定フラグ

10.5 レジスタビット変数

以下にレジスタビット変数の一覧を示します。as30 はレジスタビット変数名の大文字と小文字を区別しません。したがって、"BITR0_1"と"bitr0_1"は同じレジスタビット変数を示します。

表 10.4 レジスタビット変数

レジスタビット変数名	内容
BITR0_n	データレジスタ R0 のビット n、n は 0～15 のビット位置を記述する
BITR1_n	データレジスタ R1 のビット n、n は 0～15 のビット位置を記述する
BITR2_n	データレジスタ R2 のビット n、n は 0～15 のビット位置を記述する
BITR3_n	データレジスタ R3 のビット n、n は 0～15 のビット位置を記述する
BITA0_n	データレジスタ A0 のビット n、n は 0～15 のビット位置を記述する
BITA1_n	データレジスタ A1 のビット n、n は 0～15 のビット位置を記述する

レジスタビット変数の記述例:代入文)

```
BITR0_0 = 0
BITR1_1 = 0
BITR2_2 = 0
BITR3_3 = 0
BITA0_4 = 0
BITA1_5 = 0
```

レジスタビット変数の記述例:条件文)

```
if BITR0_1   ; レジスタ R0 の 1 ビット目を評価します。
     :
else
     :
endif

if BITR0_2   ; レジスタ R0 の 2 ビット目を評価します。
     :
else
     :
endif
```

10.6 メモリ変数

as30 の構造化記述では、ラベルおよびシンボルをメモリ変数として処理します。また、メモリ変数名は大文字と小文字を区別します。

10.6.1 メモリ変数の型

次の表に示す指示命令で定義されたラベルおよびシンボル名をメモリ変数として構造化記述文に使用できます。また、指示命令によって、変数には"変数型"が定義されます。

アセンブラは、変数型に従って、オブジェクトコードを生成します。

表 10.5 メモリ変数の型

アセンブラ指示命令	変数の型
.BTEQU、.BTGLB	ビット型
.BLKB、.BYTE	バイト型
.BLKW、.WORD	ワード型
.BLKA、.ADDR	アドレス型
.BLKL、.LWORD	ロングワード型
.GLB	外部参照ラベルおよびシンボルに対しては、1 行毎にサイズの記述を行うか、コマンドオプションによってサイズを決定します。

コマンドオプション"-M"の機能

as30のコマンドオプション"-M"が指定されている場合に、型が明示されていない変数は、バイト型でオブジェクトコードが生成されます。コマンドオプションの指定が無い場合は、ワード型でオブジェクトコードが生成されます。

10.6.2 メモリ変数のアドレッシングモード

構造化記述文では、メモリ変数に指定できるアドレッシングモードを示します。

アドレッシングモード指定子(:8、:16、:20)は、省略できます。

表 10.6 メモリ変数のアドレッシングモード

アドレッシングモード	アドレッシングモードの記述形式
絶対	[label:16]、 [label:20]
アドレスレジスタ相対	[label:8[A0]]、 [label:16[A0]] [label:8[A1]]、 [label:16[A1]] [label:20[A0]]
SB 相対	[label:8[SB]]、 [label:16[SB]]
FB 相対	[label:8[FB]]

10.6.3 メモリ変数の記述規則

- メモリ変数名を構造化記述文に記述する場合は、必ず[ ]または{ }で囲って記述してください。
- メモリ変数名と括弧の間にはスペースまたはタブが記述できます。
- アドレッシングモードを指定する場合は、アドレッシングモードも含めて必ず[ ]または{ }で囲って記述してください。

記述例 1)

```
          .GLB       work
          .SECTION   memory,DATA
    mem:  .BLKB      1
          .SECTION   program,CODE
          [mem] = 0
          [work].B = 0
          .END
```

記述例 2)

```
    if [label[SB]]
        :
    else
        :
    endif
```

10.6.4 サイズ指定子

サイズ指定子は、メモリ変数およびアドレスレジスタ間接([A0]、[A1])に対して設定できます。アセンブラは、サイズ指定子を記述したメモリ変数に対して、メモリ変数を定義したときに決定する変数の型に関係なく、一時的に指定したサイズでコードを生成します。

以下に記述できるサイズ指定子の一覧を示します。

表 10.7 サイズ指定子

サイズ指定子	変数型
.B	バイト型
.W	ワード型
.A	アドレス型
.L	ロングワード型

注意事項 サイズ指定子を設定した行のメモリ変数の型は、指示命令で決定された型よりもサイズ指定子の型が優先になります。

10.6.5 サイズ指定子の記述規則

- サイズ指定子は、括弧で囲ったメモリ変数の直後に記述してください。
- サイズ指定子と括弧の間にはスペースまたはタブが記述できます。

サイズ指定子の記述例)

```
            .SECTION   RAM,DATA
    LAB_B:  .BLKB      1
    LAB_W:  .BLKW      1

                .SECTION    ROM,CODE
    [LAB_B]     = R0L   ;   MOV.B  R0L,LAB_B
    [LAB_B].W   = R0    ;   MOV.W  R0,LAB_B
    [LAB_W]     = R0    ;   MOV.W  R0,LAB_W
    [LAB_W].B   = R0L   ;   MOV.B  R0L,LAB_W
    .END
```

10.7 メモリビット変数

次の表に示す指示命令で定義されたビットシンボル名をメモリビット変数として、構造化記述文に使用できます。

表 10.8 メモリビット変数

アセンブラ指示命令	変数の型
.BTEQU、.BTGLB	ビット型

10.7.1 メモリビット変数のアドレッシングモード

以下にメモリビット変数に指定できるアドレッシングモードを示します。

- アドレッシングモード指定子(:8、:11、:16) は、省略できます。
- 表の中の"bitnum"はビット番号、"addr"はメモリアドレスを示します。

表 10.9 メモリビット変数のアドレッシングモード

アドレッシングモード	アドレッシングモードの記述形式	
絶対	[bitsym:16]、	[bitnum、addr:16]
SB 相対	[bitsym:8[SB]]、	[bitnum、addr:8[SB]]
	[bitsym:11[SB]]、	[bitnum、addr:11[SB]]
	[bitsym:16[SB]]、	[bitnum、addr:16[SB]]
FB 相対	[bitsym:8[FB]]、	[bitnum、addr:8[FB]]

注意事項　アドレスレジスタ間接および相対アドレッシングは記述できません。

10.7.2 メモリビット変数の記述規則

- メモリビット変数名を構造化記述文に記述する場合は、必ず[ ]または{ }で囲って記述してください。
- メモリビット変数名と括弧の間にはスペースまたはタブが記述できます。
- アドレッシングモードを指定する場合は、アドレッシングモードも含めて必ず[ ]または{ }で囲って記述してください。

記述例 1　内部定義メモリビット変数の場合)

```
    BITSYM   .BTEQU  1,10h
    if [ BITSYM ]
            :
    else
    endif
```

記述例 2　外部参照メモリビット変数の場合)

```
    .BTGLB   BITSYM
    if [ BITSYM ]
            :
    else
            :
    endif
```

10.8 構造化演算子

構造化記述文に記述できる演算子を次に示します。

(1) 単項演算子

表 10.10 単項演算子

演算子	内容
+	正の数であることを示す
-	負の数であることを示す
~	ビット毎の否定(NOT)演算を行う
++	単項のインクリメント演算を行う
--	単項のデクリメント演算を行う

(2) 二項演算子

表 10.11 二項演算子

演算子	内容
+、+.C、+.D、+.CD	二項の加算演算を行う
-、-.C、-.D、-.CD	二項の減算演算を行う
、.S	二項の乗算演算を行う
/、/.S	二項の除算演算を行う
%、%.S、%.SE	二項の剰余演算を行う
&	ビット毎の論理積演算(AND)を行う
|	ビット毎の論理和演算(OR)を行う
^	ビット毎の排他的論理和演算(EOR)を行う
>>.C	左辺値を右辺の値ぶんだけキャリー付きで右にビット回転を行う
<<.C	左辺値を右辺の値ぶんだけキャリー付きで左にビット回転を行う
<>.R	左辺値を右辺の値ぶんだけキャリーなしでビット回転を行う 右辺値が正の場合左回転、負の場合右回転を行う
<>.A	左辺値を右辺の値ぶんだけキャリーなしで算術シフトを行う 右辺値が正の場合左シフト、負の場合右シフトを行う
<>.L	左辺値を右辺の値ぶんだけキャリーなしで論理シフトを行う 右辺値が正の場合左シフト、負の場合右シフトを行う
&&	論理積演算(AND)を行う
\|\| | 論理和演算(OR)を行う |

(3) 比較演算子

構造化記述文の条件式に記述できる比較演算子を次に示します。

表 10.12 比較演算子

演算子	内容
<、<.S	左辺が右辺より小さい場合に演算結果が真となる
>、>.S	左辺が右辺より大きい場合に演算結果が真となる
==	左辺と右辺が等しい場合に演算結果が真となる
!=	左辺と右辺が等しくない場合に演算結果が真となる
<=、<=.S	左辺が右辺より小さいか等しい場合に演算結果が真となる
>=、>=.S	左辺が右辺より大きいか等しい場合に演算結果が真となる

(4) 演算子の属性

二項演算子の加減算および比較演算子の一部に指定する、演算子の属性の意味を次に示します。

演算子と属性の間にスペースまたはタブは記述できません。

表 10.13 演算子の属性

演算子	内容
.C	キャリー、ボロー付き演算を行う
.D	10 進数で演算を行う
.CD	キャリー、ボロー付きで 10 進数演算を行う
.S(剰余を除く)	符号付きで演算を行う
.S(剰余)	演算結果の符号を被除数と同一にする
.SE	演算結果の符号を除数と同一にする

10.9 式

式には次に示す種類があります。

(1) 単項式

一つの項から成るものおよび一つの項に単項演算子を組み合わせた式

(2) 二項式

二つの項と演算子から成る式

(3) 複合式

単項式または式を論理演算子で組み合わせた式

10.9.1 式の項

項には次に示すものが記述できます。

(1) 変数

レジスタ変数、フラグ変数、レジスタビット変数、メモリ変数およびメモリビット変数が記述できます。

(2) 定数

「3.5 オペランド」で説明している数値が記述できます。

注意事項 二項の除算および剰余算を除いて、異なった型を持つ変数を用いた式は記述できません。

10.9.2 複合式

複合式の記述規則を次に示します。

- 論理演算子は、一つの式に二つまで記述できます。
- 複合式の演算は左から順に行われます。
- 構造化記述命令と複合式は、255 文字以内の一行に記述してください。
- 複合式を複数行にわたって記述することはできません。

```
IF [ WORK1 ] || [ WORK2 ] && [ WORK3 ]
      :
      :
ENDIF
```

10.9.3 式の記述例

式の記述例を、式の種類ごとに示します。"mem"および"work"はメモリ変数名を示します。

単項式の記述例)

```
[mem]
-[mem]
++[mem]
```

二項式の記述例)

```
[mem] + 1
-[mem] + 1
```

複合式の記述例)

```
[mem] || [work]
--[mem] && [work]
```

10.10 構造化記述文の構成

構造化記述文は、構造化記述命令とそのオペランドに記述する条件式とから成ります。構造化記述命令によっては、条件式を記述しないものもあります。

10.10.1 条件式

条件式の機能

- 構造化命令文に与える条件を示します。
- 条件式の演算結果が真であるか偽であるかによって、異なる制御ブロックへ分岐するようなオブジェクトコードを生成します。

条件式の記述規則

- 条件式は、構造化記述命令 IF、ELIF、FOR(FOR-NEXT)および WHILE のオペランドに記述できます。
- 条件式のオペランドには、「構造化記述の構文」で示した式が記述できます。
- 条件式と構造化記述命令との間には必ずスペースまたはタブを記述してください。
- 構造化記述命令と式を記述する場合、一行に記述してください。
- 複数行にわたる条件式を記述することはできません。

条件式の記述形式

- オペランド
- オペランド 比較演算子 オペランド
- ビット変数
- ビット変数 比較演算子 [1|0]

条件式の記述例)

条件式の記述例を次に示します。"mem"および"work"はメモリ変数名を示します。
"bit"はメモリビット変数名を示します。

```
IF [mem]
        :
ENDIF

FOR --[mem]
        :
NEXT

IF [mem] >= 0
        :
ENDIF

FOR [work] - [mem] <= 0
        :
NEXT
```

```
IF [bit]
        :
ENDIF

IF [bit] == 1
        :
ENDIF

IF [bit] != 0
        :
ENDIF
```

10.10.2 構造化記述文のネスティング

各構造化記述文は合計で 65535 レベルまでのネスティングが可能です。ただし、次の例のような文の交差はできません。また、マクロ指示命令、アセンブラ指示命令 ’.if’、’.elif’、’.else’、’.endif’を含めて文の交差はできません。

正しくない(交差している)ネスティング例)

```
FOR R0 = 1 TO 10 STEP 1
        :
   IF R1 == 3    ; FOR 構文の中で IF 構文を開始
        :
NEXT
   ENDIF         ; FOR 構文の外で IF 構文を終了
```

10.11 構造化記述命令

以降に構造化記述命令の記述規則を示します。

条件分岐文

IF 文

書 式(IF-ENDIF)

```
IF△＜条件式＞
    制御ブロック
ENDIF
```

説 明　IF 文の基本構成は、構造化記述命令 IF、ENDIF およびこれらの命令で囲まれた制御ブロックから成ります。
IF の条件が偽のとき、ENDIF に分岐します。

備 考　条件式には、「10.10.1 条件式」の項で示した式が記述できます。

書 式(ELSE)

```
IF△＜条件式＞
    制御ブロック
ELSE
    制御ブロック
ENDIF
```

説 明　IF 文には、構造化記述命令 ELSE を記述できます。
IF の条件式が偽のときに、ELSE に続く制御ブロックに分岐します。
制御ブロックが複数ある場合は、各制御ブロックの最後から ENDIF に分岐します。

備 考　ELSE は、IF と ENDIF の間に 1 つだけ記述できます。

書 式(ELIF)

```
IF△＜条件式＞
    制御ブロック
ELIF△＜条件式＞
    制御ブロック
ENDIF
```

説 明　IF 文には、構造化記述命令 ELIF を記述できます。
IF の条件式が偽のときに、ELIF の条件式を判断します。
ELIF の条件式が真のとき、直後の制御ブロックの先頭に分岐します。
ELIF の条件式が偽のとき、直後の構造化記述命令(ELIF、ELSE または ENDIF)に分岐します。
制御ブロックが複数ある場合は、各制御ブロックの最後から ENDIF に分岐します。

備 考　条件式には、「10.10.1 条件式」の項で示した式が記述できます。
ELIF は、IF と ELSE の間または IF と ENDIF の間に、1 つ以上記述できます。

記述例

```
IF [sym1] == 10
     :
ELIF [sym2] != 10
     :
ELSE
     :
ENDIF
```

展開例

```
CMP.B      #10,sym1
JNE        ..IF0002
     :
JMP        ..IF0003
..IF0002:
CMP.B      #10,sym2
JEQ        ..IF0004
     :
JMP        ..IF0003
..IF0004:
     :
..IF0003:
```

繰り返し文（条件指定）

FOR-NEXT 文

書 式　　FOR△＜条件式＞
　　制御ブロック
NEXT

説 明　　FOR 文の基本構成は、構造化記述命令 FOR、NEXT およびこれらの命令で囲まれた制御ブロックから成ります。
条件式が真のとき、直後の制御ブロックに分岐します。
条件式が偽のとき、構造化命令 NEXT の直後の行に分岐します。
制御ブロック内に、BREAK 文が記述できます。BREAK 文は、繰り返しの制御を強制的に終了します。
制御ブロック内に、CONTINUE 文が記述できます。CONTINUE 文は、NEXT 文に分岐します。
条件式に FOREVER 文を記述できます。FOREVER 文は、制御ブロックを繰り返し実行します。

記述例

```
    FOR R0 <.S 10
        :
    NEXT
```

展開例

R0 が 10 よりも小さい間繰り返し

```
    ..fr0000:
    CMP.W   #10,R0
    JGE     ..fr0002
        :
    JMP     ..fr0000
    ..fr0002
```

備 考　　条件式には、「10.10.1 条件式」の項で示した式が記述できます。

繰り返し文（回数指定）

FOR-STEP 文

書 式　FOR△＜ループカウンタ＞＝＜初期値＞△TO△＜最終値＞△[STEP△＜増分＞]
　　制御ブロック
NEXT

説 明　FOR 文の基本構成は、構造化記述命令 FOR、NEXT およびこれらの命令で囲まれた制御ブロックから成ります。
構造化記述命令 FOR のオペランドに指定されているループカウンタの値を増分だけ更新し、最終値と等しくなければ制御ブロックを実行します。
ループカウンタの値が最終値と等しい場合は、構造化記述命令 NEXT の直後の行に分岐します。
増分の値が負の数の場合は、ループカウンタをダウンカウントします。
増分が省略されている場合は、"+1"としてオブジェクトコードを生成します。
制御ブロック内に、BREAK 文が記述できます。BREAK 文は、繰り返しの制御を強制的に終了します。
制御ブロック内に、CONTINUE 文が記述できます。CONTINUE 文は、NEXT 文に分岐します。

記述例

```
FOR [LAB].W = 0 TO 10 STEP 1
        :
NEXT
```

展開例

```
      MOV.W   #0,LAB
..fr0000:
      CMP.W   #10,LAB
      JEQ     ..fr0002
        :
..fr0001:
      ADD.W   #1,LAB
      JMP     ..fr0000
      ..fr0002
```

備 考　ループカウンタには、レジスタ変数とメモリ変数が記述できます。
初期値および最終値には、変数または定数値が記述できます。
増分には定数値が記述できます。
定数値としてローカルシンボル名が記述できます。

注意事項　ループカウンタの値が必ず最終値に等しくなるまで制御ブロックは繰り返されます。
ループカウンタに使用しているレジスタ変数およびメモリ変数の内容を制御ブロック内で変更している場合は、FOR 文が正しく実行されません。

多条件分岐

SWITCH

書 式(基本構成)

```
SWITCH△＜式＞
CASE△＜データ＞
        制御ブロック
CASE△＜データ＞
        制御ブロック
ENDS
```

説 明　SWITCH 文の基本構成は、構造化記述命令 SWITCH、ENDS、CASE および CASE で区切られる制御ブロックから成ります。
SWITCH 文のオペランドに記述されている式の内容と一致するデータを持つ CASE 命令の直後の制御ブロックに分岐します。
全ての CASE 命令のデータについて評価します。

備 考　SWITCH のオペランドの式には、「10.9 式」で示した単項式および二項式が記述できます。
CASE とデータは必ず一つ以上記述してください。SWITCH と ENDS の間に CASE が存在しない場合は、ウォーニングが出力されます。
CASE のオペランドのデータには定数が記述できます。
CASE のオペランドのデータにはアセンブル実行時に確定する値以外は記述できません。
CASE のオペランドのデータに同一の値は記述できません。

書 式(BREAK)

```
SWITCH△＜式＞
CASE△＜データ＞
        制御ブロック
        BREAK
CASE△＜データ＞
        制御ブロック
ENDS
```

説 明　BREAK 文は無条件で ENDS に分岐します。

備 考　BREAK 命令は、制御ブロックの最後に記述してください。
制御ブロックの途中に記述した場合はウォーニングが出力されます。このとき、BREAK 命令と次の構造化記述命令の間の行のコードは生成されますが、その部分に分岐するコードは生成されません。

書 式(DEFAULT)

```
SWITCH△＜式＞
CASE△＜比較データ 1＞
        制御ブロック
CASE△＜比較データ 2＞
        制御ブロック
DEFAULT
        制御ブロック
ENDS
```

説 明　　式に一致するデータが一つもなかった場合に、DEFAULT の直後の制御ブロックに分岐します。
構造化記述命令 DEFAULT と ENDS の間に記述された CASE に対してウォーニングが出力されます。このとき、該当する CASE の直後の制御ブロックのオブジェクトコードは生成されますが、そのブロックに分岐するコードは生成されません。

備 考　　SWITCH 文の ENDS の直前に構造化記述命令 DEFAULT と制御ブロックを記述できます。
構造化記述命令 DEFAULT は一つの SWITCH 文に、一つしか記述できません。

記述例

```
SWITCH [work]
CASE 1
      :
BREAK
CASE 2
      :
DEFAULT
      :
ENDS
```

展開例

```
      CMP.B   #1,WORK
      JNE     ..sw0004
            :
      JMP     ..sw0000
..sw0004:
      CMP.B   #2,WORK
      JNE     ..sw0006
            :
..sw0006:
            :
..sw0000:
```

繰り返し文（条件指定）

DO-WHILE

書 式　　DO
　　制御ブロック
WHILE△＜条件式＞

説 明　　DO 文の基本構成は、構造化記述命令'DO'、'WHILE'及びこれらの命令に囲まれた制御ブロックから成ります。
制御ブロックを実行した後で、WHILE のオペランドに記述されている条件式を判断します。
条件式が真のとき、DO に分岐します。
条件式が偽のとき、次の行に分岐します。
制御ブロック内に BREAK を記述できます。BREAK は、WHILE の次の行に分岐します。
制御ブロック内に CONTINUE を記述できます。CONTINUE は、WHILE 文に分岐します。
条件式に FOREVER を記述できます。FOREVER を記述した場合、無条件で DO 文に分岐します。

記述例

```
DO
     :
WHILE [lab].b == 1
```

展開例

```
..DO0000:
      :
    CMP.B  #1,lab
    JEQ    ..DO0000
..DO0002:
```

備 考　　条件式には、「10.10.1 条件式」の項で示した式が記述できます。

無条件分岐

BREAK

書 式　　BREAK

説 明　　無条件分岐命令を生成します。

記述例

```
FOR [lab]=1 TO 10 STEP 1
            :
BREAK
            :
NEXT
```

展開例

```
    MOV.W   #1,lab
..fr0000:
    CMP.W   #10,lab
    JEQ     ..fr0002
            :
    JMP     ..fr0002
            :
..fr0001:
    ADD.W   #1,lab
    JMP     ..fr0000
..fr0002:
    ; for NEXT
```

備 考　　BREAK 文は、 FOR、DO、SWITCH の制御ブロック内に記述できます。
FOR、DO、SWITCH の制御ブロック内でのみ、IF 文の制御ブロック内に記述できます。
通常の IF 文の制御ブロック内には記述できません。

無条件分岐（条件判断文へ）

CONTINUE

書 式　　CONTINUE

説 明　　条件判定式への無条件分岐命令を生成します。

記述例

```
FOR [lab]=1 TO 10 STEP 1
          :
CONTINUE
          :
NEXT
```

展開例

```
    MOV.W   #1,lab      ; FOR による生成
..fr0000:               ; FOR による生成
    CMP.W   #10,lab     ; FOR による生成
    JEQ     ..fr0002    ; FOR による生成
          :
    JMP     ..fr0001    ; CONTINUE による生成
          :
..fr0001:               ; STEP にる生成
    ADD.W   #1,lab      ; STEP にる生成
    JMP     ..fr0000    ; STEP にる生成
..fr0002:               ; NEXT による生成
```

備 考　　CONTINUE 文は、FOR、DO 文の制御ブロック内に記述できます。
FOR、DO の制御ブロック内でのみ、IF 文および SWITCH 文の制御ブロック内に記述できます。
通常の IF 文および SWITCH 文の制御ブロック内には記述できません。

繰り返しの条件を常に真とする

FOREVER

書 式(FOR△FOREVER
　制御ブロック
NEXT

DO
　制御ブロック
WHILE△FOREVER

説 明 制御ブロックを繰り返し実行しつづけます。
制御ブロック内に、BREAK 文が記述できます。BREAK 文は、繰り返しの制御を強制的に終了します。
制御ブロック内に、CONTINUE 文が記述できます。CONTINUE 文は、繰り返しの判断を行なう行に分岐します。

記述例

```
FOR  FOREVER
          :
NEXT
```

展開例

```
..fr0000:
          :
    JMP   ..fr0000
..fr0002:
```

右辺を左辺に代入

代入文

書 式　　＜左辺＞=＜右辺＞

説 明　　代入文は右辺の式を演算した結果を左辺の変数に代入します。
代入文には次に示す種類があります。

表 10.14 代入文

演算子	内容
=	符号なしの値を左辺に代入します。
=.S	右辺の符号拡張した値を左辺に代入します。
=.Z	右辺のゼロ拡張した値を左辺に代入します。
=.EL	LDE 命令を生成します。
=.ES	STE 命令を生成します。

備 考　　代入文"=.S"、"=.Z"、"=.EL"および"=.ES"の右辺には単項演算子および二項演算子を含む式は記述できません。
代入文"=.S"、"=.Z"の左辺および右辺には次に示す変数が記述できます。
　　メモリ変数([SP]相対は除く)
　　レジスタ変数のうちデータレジスタとアドレスレジスタ間接
代入文"=.EL"の左辺および右辺に記述できる変数は、ニーモニック"LDE"のオペランド dest および src に記述できる内容です。
代入文"=.ES"の左辺および右辺に記述できる変数は、ニーモニック"STE"のオペランド dest および src に記述できる内容です。
代入文の左辺と右辺にまったく同一の変数を記述した場合は、ウォーニングが出力されます。
異なった型を持つ変数を代入する場合、代入文の右辺には単項演算子および二項演算子を含む式は記述できません。

注意事項　　ニーモニックの詳細については、M16C シリーズ、R8C ファミリの各ソフトウェアマニュアルを参照してください。

代入文(=)に記述できる変数型の組み合わせ

表 10.15 代入文

代入文	右辺			
	バイト型	ワード型	アドレス型	ロングワード型
バイト型	○	×	×	×
ワード型	×	○	×	×
アドレス型	×	×	○	×
ロングワード型	×	×	×	○

符号拡張代入文(=.S)に記述できる変数型の組み合わせ

表 10.16 符号拡張代入文

符号拡張代入文	右辺			
	バイト型	ワード型	アドレス型	ロングワード型
バイト型	×	×	×	×
ワード型	○	×	×	×
アドレス型	×	×	×	×
ロングワード型	×	○	×	×

注意事項
ワード型 =.S バイト型の代入式で、左辺に"R2"または"R3"を指定した場合、"R0"レジスタを使用します。
ロングワード型 =.S ワード型の代入式で、左辺に"メモリ変数"または"R3R1"を指定した場合、"R2R0"を使用します。

ゼロ拡張代入文(=.Z)に記述できる変数型の組み合わせ

表 10.17 ゼロ拡張代入文

ゼロ拡張代入文	右辺			
	バイト型	ワード型	アドレス型	ロングワード型
バイト型	×	×	×	×
ワード型	○	×	×	×
アドレス型	○	○	×	×
ロングワード型	○	○	○	×

注意事項
ワード型=.Z バイト型の代入式で、右辺に"R2、 R3"を指定した場合、"R0"を使用します。

特殊命令代入文(=.EL、 =.ES)に記述できる変数型の組み合わせ

表 10.18 特殊命令代入文

特殊命令代入文	右辺			
	バイト型	ワード型	アドレス型	ロングワード型
バイト型	○	×	×	×
ワード型	×	○	×	×
アドレス型	×	×	×	×
ロングワード型	×	×	×	×

代入文の記述例とその展開例

表 10.19 代入文の例

ソース記述例	展開例	
R1 = R0	MOV.W	R0,R1
R0 = R0 + 2	ADD.W	#2,R0
R0 =.S R0L	EXTS.B	R0L
R0 =.Z R0L	MOV.B	#0,R0H
R0L =.EL [lab].B	LDE.B	lab,R0L
[lab].W =.ES R0	STE.W	R0,lab
R0 =.S R0H	MOV.B	R0H,R0L
	EXTS.B	R0L
[lab_w].W =.S R0L	MOV.B	R0L,lab_w
	EXTS.B	lab_w
R2R0 =.S R0	EXTS.W	R0
R2R0 =.S R1	MOV.W	R1,R0
	EXTS.W	R0
[lab_l].L =.S R0	EXTS.W	R0
	MOV.W	R0,lab_l
	MOV.W	R2,lab_l+2
R0 =.Z R0H	MOV.B	R0H.R0L
	MOV.B	#0,R0H
[lab_w].W =.Z R0L	MOV.B	R0H,lab_w
	MOV.B	#0,lab_w+1
[lab_a].A =.Z R0	MOV.W	R0L,lab_a
	MOV.B	#0,lab_a+2

10.12 構造化記述命令の構文

プログラミングで記述できる構造化記述文を示します。構造化記述をする場合は、以降に示す構文に従って記述してください。

用語定義
本項で使用する記述用語について説明します。用語の記述されている位置に、各用語の示す変数名および演算子が記述できます。

(1)レジスタ変数

表 10.20 レジスタ変数

レジスタ変数群	レジスタ変数名
regb	R0L、R0H、R1L、R1H、A0.B、A1.B、[A0.B]、[A1.B]
regw	R0、R1、R2、R3、A0、A1、[A0]、[A1]
regc	FB、SB、SP、ISP、FLG、INTBH、INTBL
reglw	R2R0、R3R1
regad	A1A0

注意事項
SPは、Uフラグで示すスタックポインタ(ユーザスタックポインタまたは割り込みスタックポインタ)が対象となります。スタックポインタおよびUフラグの機能についての詳細は、M16C シリーズ、R8C ファミリの各ソフトウェアマニュアルを参照してください。

(2)メモリ変数

表 10.21 メモリ変数

メモリ変数群	メモリ変数
memb	バイト型メモリ変数(″SP″ の記述を除く)
memw	ワード型メモリ変数(″SP″ の記述を除く)
mema	アドレス型メモリ変数
meml	ロングワード型メモリ変数
regmembit	レジスタビット変数、メモリビット変数
flgbit	フラグ変数

(3)演算子

表 10.22 演算子

演算子群	演算子
単項演算子	~、-、++、--
二項演算子 1	+(.C)、-(.C)
二項演算子 2	+.C(D) 、-.C(D)
二項演算子 3	*(.S)
二項演算子 4	/(.S)、%(.S、.SE)
二項演算子 5	&、|、^
二項演算子 6	>>.C、<<.C
二項演算子 7	<>.R
二項演算子 8	<>.A、<>.L
比較演算子	==、!= 、>(.S) 、<(.S) 、=>(.S) 、<=(.S)
一致比較演算子	==、!=
論理演算子	&&、||
定数	数値またはアセンブル時に確定する式の値

単純代入文および単項演算子を含んだ代入文の構文

注意事項 ″=.S″および″=.Z″における″regb″および″rebw″はデータレジスタ変数のみ記述できます。

左辺がメモリ変数の式)

```
memb  =  定数
memb  =  単項演算子  memb
memb  =  単項演算子  regb
memw  =  定数
memw  =.S  単項演算子  memw
memw  =.S  単項演算子  regw
memw  =.S  memb
memw  =.S  regb
memw  =.Z  memb
memw  =.Z  reg
mema  =  定数
mema  =  mema
mema  =.Z  memb
mema  =.Z  memw
mema  =.Z  regb
mema  =.Z  regw
memlw  =  定数
memlw  =  meml
memlw  =  R2R0
memlw  =  R3R1
memlw  =  A1A0
memlw  =.S  memw
memlw  =.S  regw
memlw  =.Z  memb
memlw  =.Z  memw
memlw  =.Z  mema
memlw  =.Z  regb
memlw  =.Z  regw
memb  =.ES  memb、regb
memw  =.ES  memw、regw
memb  =  [STK].B
memw  =  [STK].W
dsp:8[SP]  =  memb、regb
dsp:8[SP]  =  memw、regw
```

左辺がレジスタの代入文)

```
regb  =  定数
```

```
regb  =  [単項演算子]memb
regb  =  [単項演算子]regb
regw  =  定数
regw  =  [単項演算子]memw
regw  =  [単項演算子]regw
regw  =.S  memb
regw  =.S  regb
regw  =.Z  memb
regw  =.Z  regb
regl  =  定数
regl  =  meml
regl  =  R2R0
regl  =  R3R1
regl  =  A1A0
regl  =.S  memw
regl  =.S  regw
regl  =.Z  memb
regl  =.Z  memw
regl  =.Z  mema
regl  =.Z  regb
regl  =.Z  regw
regc  =  定数
regc  =  memw
regc  =  regw
regb（A0.B、A1.B は除く）  =  [STK].B
regw  =  [STK].W
regc  =  [STK].W
R0、R1、R2、R3、A0、A1、SB、FB（複数指定可）  =  [STK].W
INTB  =  定数
IPL  =  定数
```

左辺がメモリ変数またはレジスタ変数の式）

```
memb、regb  =.EL  memb
memw、regw  =.EL  memw
memw、regw  =  regc
memb、regb  =  dsp:8[SP]
memw、regw  =  dsp:8[SP]
mema、[A0.A]、[A1.A]、R2R0、R3R1、A1A0  =  regpc
```

左辺がスタック変数の代入文）

```
[STK].B  =  定数
[STK].B  =  memb
[STK].B  =  regb（A0.B および A1.B は除く）
[STK].W  =  定数
[STK].W  =  memw
[STK].W  =  regw
[STK].W  =  regc
[STK].W  =  R0、R1、R2、R3、A0、A1、SB、FB（複数記述可能）
[STK].A  =  mema
```

左辺がビット変数の代入文）

```
regmembit  =  1、0、~regmembit(左辺と同じビット名)
flgbit  =  1、0
```

単項演算子を含む代入文）

```
memb/regb  =  [単項演算子]memb/regb
memw/regw  =  [単項演算子]memw/regw
```

二項演算子 1 を含む代入文)
```
memb/regb  =[単項演算子]memb/regb  二項演算子 1  定数/memb/regb
memw/regw  =[単項演算子]memw/regw  二項演算子 1  定数/memw/regw
```

二項演算子 2 を含む代入文)
```
memb/regb=[単項演算子]memb/regb  二項演算子 2  定数/memb/regb
memw/regw=[単項演算子]memw/regw  二項演算子 2  定数/memw/regw
```

二項演算子 3 を含む代入文)
```
memw/regw  =  [単項演算子]memb/regb  二項演算子 3  定数
/memb/regb
meml/regl  =  [単項演算子]memw/regw  二項演算子 3  定数
/memw/regw
```

二項演算子 4 を含む代入文)
```
memb/regb  =  [単項演算子]memw/regw  二項演算子 4  定数
/memb/regb
memw/regw  =  meml/reglw/regad  二項演算子 4  定数/memw/regw
```

二項演算子 5 を含む代入文)
```
memb/regb  =  [単項演算子]memb/regb  二項演算子 5  定数
/memb/regb
memw/regw  =  [単項演算子]memw/regw  二項演算子 5  定数
/memw/regw
```

二項演算子 6 を含む代入文
```
memb/regb  =  [単項演算子]memb/regb  二項演算子 6  定数
memw/regw  =  [単項演算子]memw/regw  二項演算子 6  定数
```

二項演算子 7 を含む代入文)
```
memb/regb  =[単項演算子]memb/regb  二項演算子 7  定数/R1H
memw/regw  =[単項演算子]memw/regw  二項演算子 7  定数/R1H
```

二項演算子 8 を含む代入文)
```
memb/regb  =  [単項演算子]memb/regb  二項演算子 8  定数/R1H
memw/regw  =  [単項演算子]memw/regw  二項演算子 8  定数/R1H
meml/reglw/regad  =  meml/reglw/regad  二項演算子 8  定数/R1H
```

式 1 の構文
```
[単項演算子] [memb|regb]
[単項演算子] [memw|regw]
式 2
式 2  比較演算子  [即値|memb|regb]
式 2  比較演算子  [即値|memw|regw]
式 2  論理演算子  式 2
式 3  論理演算子  式 3
式 3
[regmembit|flgbit]
```

式2の構文

代入式の右辺に示した構文のうち、下記の内容を除くすべての構文

- 次に示すレジスタおよびスタック

  FB、SB、SP、ISP、FLG、INTBH、INTBL、INTB、IPL および [STK]
- 乗算の結果が 32 ビットになる式
- レジスタビット変数およびメモリビット変数の反転式

  ~regmembit

式3の構文

```
二項演算式.b  比較演算子  [定数|memb|regb]
二項演算式.w  比較演算子  [定数|memw|regw]
[regmembit|flgbit]  一致比較演算子  [1|0]
```

条件式の構文

```
IF 文の構文
   IF  式 1
       :
   ENDIF

FOR-NEXT 文の構文
   FOR  式 1
       :
   NEXT

FOR-STEP 文の構文
   FOR 変数  =  [変数|定数]  TO  [変数|定数]  STEP  定数
       :
   NEXT

WHILE 文の構文
   DO
       :
   WHILE  式 1

SWITCH 文の構文
   SWITCH  式 2
   CASE  比較データ
       :
   ENDS
```

11. アセンブラのエラーメッセージ

11.1 エラー形式

本章では、以下の形式で出力するエラーメッセージとエラー内容を説明します。

エラー番号 (エラーレベル) エラーメッセージ
エラー内容

エラーレベルは、エラーの重要度に従い、4種類に分類されます。

エラー番号	エラーレベル	エラータイプ	動作
A1000 - A1999	(W)	ウォーニング	処理を継続します。
A2000 - A2999	(E)	エラー	処理を中断します。
A3000 - A3999	(F)	フェータル	処理を中断します。
A4000 - A4999	(-)	インターナル	処理を中断します。

11.2 エラーの返値

各プログラムは処理を終了した際、処理結果によって次の値を OS に返します。

返値	内容
0	プログラムはエラーもなく正常に終了
1	コントロールCの入力によって強制的に終了
2	処理対象のファイルに対してフェータルが発生し終了
3	処理対象のファイルに対してエラーが発生し終了
4	コマンド行の入力に対してエラーが発生し終了

11.3 メッセージ一覧

A1001 (W) Non support command option 'xxx' is used
サポートしていないコマンドオプションを設定しています。
コマンドオプションを入力し直してください。

A1101 (W) Too many actual macro parameters
マクロ実引数の数が多すぎます。
余分な実引数は無視されます。

A1102 (W) Actual macro parameters are not enough
マクロ実引数の数がマクロ仮引数の数より少なくなっています。
該当する実引数のない仮引数は無効となります。

A1103 (W) String 'xxx' is too long
文字列が長すぎます。
文字列を短くしてください。

A1104 (W) Symbol 'xxx' is not defined (regarded as 0)
シンボルが定義されていません。(シンボルの値は 0 と見なされます)
シンボルを定義してください。

A1105 (W) Unnecessary ':' is found
マクロ名の後ろにコロンが付けられています。
マクロ名の後ろのコロンを削除してください。

A1106 (W) Source line exceeds 8192 characters
1 行に記述できる文字数が 8192 文字を超えています。
1 行に記述できる文字数が 8192 文字を超えないようにしてください。

A1107 (W) .END statement is in include file
インクルードファイルに.END 記述があります。
インクルードファイル内には、.END は記述できません。記述を削除してください。
.END を無視します。

A1200 (W) '.ALIGN' with not 'ALIGN' specified relocatable section
ALIGN 指定がないセクション内に指示命令.ALIGN が記述されています。
指示命令.ALIGN の記述位置を確認してください。
指示命令.ALIGN を記述するセクションのセクション定義行に.ALIGN 指定を記述してください。

A1201 (W) Destination address may be changed
分岐先が期待するものと異なる位置になる可能性があります。
アドレッシングモードが最適選択されないように分岐命令のオペランドを記述してください。

A1202 (W) Floating point value is out of range
浮動小数点数が範囲外です。
浮動小数点数の記述を確認してください。範囲を超えた分は考慮しません。

A1203 (W) Location counter exceed xxx

ロケーションカウンタが xxx を超えました。

.ORG のオペランド値を確認してください。ソースを記述し直してください。

A1204 (W) Moved between address registers as byte size

アドレスレジスタ同志の転送がバイトサイズで行なわれています。

ニーモニックを記述し直してください。

A1205 (W) Invalid '.SBSYM' declaration, it's declared by '.FBSYM'

シンボルは既に.FBSYM で宣言されています。.SBSYM 宣言は無視されます。

宣言を記述し直してください。

A1206 (W) Invalid '.FBSYM' declaration, it's declared by '.SBSYM'

シンボルは既に.SBSYM で宣言されています。.FBSYM 宣言は無視されます。

宣言を記述し直してください。

A1207 (W) Addressing is described by the numerical value

アドレスを指定するべきオペランドに数値が記述されています。

数値は'#'を付加して記述してください。

A1208 (W) The shift instruction which uses R1H is described

シフト命令のシフト回数が R1H で設定されています。

デバイスの注意事項に該当しないか確認してください。

A1209 (W) Mnemonic in 'ROMDATA' section

ROMDATA タイプのセクションにニーモニックが記述されています。

ニーモニックを記述するセクションは CODE タイプを指定してください。

A1210 (W) Fixed data in 'CODE' section

CODE セクション内に指示命令、.BYTE、.WORD(S)、.ADDR、.LWORD が記述されています。

ROM 領域にデータを格納する指示命令(.BYTE、.WORD(S)、.ADDR、.LWORD)を記述する

セクションは ROMDATA タイプを指定してください。

A1211 (W) Control register differ size

コントロールレジスタのサイズが M16C/80 では他の M16C シリーズでは異なります。

オペランドのデータサイズを M16C/80 シリーズのコントロールレジスタのサイズにあわせてください。

A1212 (W) Calculation result is different

演算結果が異なります。

演算結果を確認してください。

A1213 (W) Invalid '.FBSYM' declaration, it's declared by '.SBSYM'

シンボルは既に.SBSYM で宣言されています。.FBSYM 宣言は無視されます。

宣言を記述し直してください。

A1214 (W) Invalid '.SBSYM16' declaration, it's declared by '.SBSYM'
シンボルは既に.SBSYM で宣言されています。.SBSYM16 宣言は無視されます。
宣言を記述し直してください。

A1215 (W) Invalid '.SBSYM' declaration, it's declared by '.FBSYM'
シンボルは既に.FBSYM で宣言されています。.SBSYM 宣言は無視されます。
宣言を記述し直してください。

A1216 (W) Invalid '.SBSYM16' declaration, it's declared by '.FBSYM'
シンボルは既に.FBSYM で宣言されています。.SBSYM16 宣言は無視されます。
宣言を記述し直してください。

A1217 (W) Invalid '.SBSYM' declaration, it's declared by '.SBSYM16'
シンボルは既に.SBSYM16 で宣言されています。.SBSYM 宣言は無視されます。
宣言を記述し直してください。

A1218 (W) Invalid '.FBSYM' declaration, it's declared by '.SBSYM16'
シンボルは既に.SBSYM16 で宣言されています。.FBSYM 宣言は無視されます。
宣言を記述し直してください。

A1219 (W) '-JOPT' and '.OPTJ' are specified
-JOPT オプションと指示命令.OPTJ が共に指定されています。
指示命令.OPTJ は無視されます。

A1220 (W) '.ALIGN' size is different
アライメント補正値が異なります。
アライメント補正値を確認してください。

A1221 (W) Fixed point value is out of range
固定小数点が範囲外です。
固定小数点の記述を確認してください。範囲を超えた分は考慮しません。

A1222 (W) The register used by the operation is different
MCU の変更により記述された命令の機能が異なっています。
命令の機能を確認してください。

A1223 (W) Use string instructions
ストリングス命令が使用されています。
デバイスの注意事項に該当しないか確認してください。

A1224 (W) Use product sum operation instruction
積和演算命令が使用されています。
デバイスの注意事項に該当しないか確認してください。

A1225 (W) Invalid '.SB_AUTO_SBSYM' declaration, it's declared by '.FBSYM'
シンボルは既に.FBSYM で宣言されています。.SB_AUTOSBSYM 宣言は無視されます。
宣言を記述し直してください。

A1226 (W) Invalid '.FBSYM' declaration, it's declared by '.SB_AUTO_SBSYM'

シンボルは既に.SB_AUTOSBSYM で宣言されています。.FBSYM 宣言は無視されます。
宣言を記述し直してください。

A1227 (W) Section attribute mismatch

セクションの属性が異なっています。
セクションタイプおよび ALIGN 設定を同じにしてください。

A1228 (W) Non support directive command is used

サポートしていない指示命令を設定しています。
宣言を記述し直してください。

A1229 (W) Invalid '.SECTION' declaration

セクションを宣言することはできません。定義は無視されます。
.SECTION の記述位置を確認してください。

A1230 (W) Function information is not defined

インスペクタ情報の関数情報が定義されていません。
関数情報を定義してください。

A1300 (W) Statement has not effect

命令行として意味を持ちません。
命令の記述方法を確認してください。

A1301 (W) 'CASE' not exist in 'SWITCH' statement

SWITCH 文の中に CASE 記述がありません。
SWITCH 文には必ず一つ以上の CASE 文を記述してください。

A1303 (W) 'CASE' definition is after 'DEFAULT'

DEFAULT 記述以降に CASE の記述があります。
DEFAULT 命令はすべての CASE 文の後に記述してください。

A1304 (W) Bit number is ignored

ビット番号は指定できません。ビット番号を無視します
記述内容を確認してください。

A1305 (W) Too many structured label definition

生成するラベルが多すぎます。
ファイルを分割してアセンブルしてください。

A1306 (W) Unnecessary BREAK is found

一つの SWITCH ブロック内に複数の BREAK 文が記述されています。
BREAK 文を一つにしてください。

A2001 (E) No input files specified

入力ファイルの指定がありません。
入力ファイルを指定してください。

A2002 (E) Invalid option 'option' is used

無効なオプション'option'を使用しています。

オプションを指定し直してください。

A2003 (E) Option 'option' is not appropriate

オプションの使用方法が間違っています。

オプションの使用方法を確認して、指定し直してください。

A2004 (E) Source files number exceed 80

ファイルの数が 80 を超えています。

複数回にわけてアセンブルを実行してください。

A2005 (E) Command line is too long

コマンド行の文字数が多すぎます。

コマンドを入力し直してください。

A2006 (E) Specified an option that can't be used with '-R8C'

-R8C オプションと同時に指定することのできないオプションが設定されています。

記述内容を確認してください。

A2007 (E) Specified an option that can't be used with '-R8CE'

-R8CE オプションと同時に指定することのできないオプションが設定されています。

記述内容を確認してください。

A2101 (E) No .END statement

.END の記述がありません。

ソースプログラムの最後の行に.END を記述してください。

A2102 (E) Value is out of range

値が範囲外です。

レジスタなどのビット長に合った値を記述してください。

A2103 (E) Illegal operand is used

オペランドが間違っています。

オペランドの記述方法を確認して、記述し直してください。

A2104 (E) Illegal directive command is used

不正な指示命令を記述しています。

正しい指示命令に記述し直してください。

A2105 (E) Invalid label definition

無効なラベル記述をしています。

ラベル定義を記述し直してください。

A2106 (E) No ';' at the top of comment

コメント先頭に ';' が記述されていません。
コメントの先頭には、セミコロンを記述してください。ニーモニックまたはオペランドの記述に誤りがないか確認してください。

A2107 (E) Invalid symbol definition

無効なシンボル記述をしています。
シンボルの定義を記述し直してください。

A2108 (E) Include nesting over

インクルードのネスティングが多すぎます。
インクルードレベルが 9 以下になるように記述し直してください。

A2109 (E) Can't open include file 'filename'

インクルードファイルをオープンできません。
インクルードファイル名を確認してください。インクルードファイルの格納ディレクトリを確認してください。

A2110 (E) Can't open '.ASSERT' message file 'filename'

.ASSERT の出力ファイルをオープンできません。
ファイル名を確認してください。

A2111 (E) Can't write '.ASSERT' message file 'filename'

.ASSERT の出力ファイルに書き込みできません。
ファイルのパーミッションを確認してください。

A2112 (E) Including the include file in itself

インクルードファイル内で、自身をインクルードしています。
インクルードファイル名を確認して、記述し直してください。

A2113 (E) Too many macro nesting

マクロのネスティングが多すぎます。
マクロのネスティングレベルを 65535 レベル以下にしてください。ソース記述を確認してください。

A2114 (E) Too many macro local label definition

マクロローカルラベルの定義が多すぎます。
マクロローカルラベル数を 1 ファイル内に 65535 個以下にしてください。

A2115 (E) Operand number is not enough

オペランドが不足しています。
オペランドの記述方法を確認して、記述し直してください。

A2116 (E) Reserved word is used as label or symbol

予約語をラベルまたはシンボルに用いています。
ラベルまたはシンボル名を記述し直してください。

A2117 (E) ')' is missing

')' の記述がありません。
'(' に対応する')'を記述してください。

A2118 (E) '.IF' is missing for '.ELSE'

.ELSE に対する.IF がありません。

.ELSE の記述位置を確認してください。

A2119 (E) '.IF' is missing for '.ELIF'

.ELIF に対する.IF がありません。

.ELIF の記述位置を確認してください。

A2120 (E) '.IF' is missing for '.ENDIF'

.ENDIF に対する.IF がありません。

.ENDIF の記述位置を確認してください。

A2121 (E) '.MACRO' is missing for '.ENDM'

.ENDM に対する.MACRO がありません。

.ENDM の記述位置を確認してください。

A2122 (E) '.MREPEAT' is missing for '.ENDR'

.ENDR に対する.MREPEAT がありません。

.ENDR の記述位置を確認してください。

A2123 (E) '.MACRO' or '.MREPEAT' is missing for '.EXITM'

.EXITM に対する.MACRO または.MREPEAT がありません。

.EXITM の記述位置を確認してください。

A2124 (E) No macro name

マクロ名がありません。

マクロ定義には、マクロ名を記述してください。

A2125 (E) Symbol is multiple defined

シンボルが二重定義です。マクロ名と他の名前が重複しています。

名前を変更してください。

A2126 (E) Too many formal parameter

マクロの仮引数の定義数が多すぎます。

マクロの仮引数の数を 80 以下にしてください。

A2127 (E) Illegal macro parameter

マクロ引数に不正な記述があります。

マクロ引数の記述内容を確認してください。

A2128 (E) Source line is too long

ソース行が長すぎます。

ソース行の記述内容を確認してください。

A2129 (E) '.MACRO' is missing for '.LOCAL'

.LOCAL に対する.MACRO がありません。

.LOCAL の記述位置を確認してください。.LOCAL は、マクロブロック内にしか記述できません。

A2130 (E) Too many nesting level of condition assemble

条件アセンブルのネスティングが多すぎます。

条件アセンブルの記述を確認してください。

A2131 (E) No '.ENDM' statement

.ENDM 記述がありません。

.ENDM の記述位置を確認してください。.ENDM を記述してください。

A2132 (E) No '.ENDR' statement

.ENDR 記述がありません。

.ENDR の記述位置を確認してください。.ENDR を記述してください。

A2133 (E) Symbol is undefined

シンボルが未定義です。

未定義または前方参照となるシンボル名は使用できません。シンボル名を確認してください。

A2134 (E) No .ENDIF statement

ソースファイル内に IF 文に対応した ENDIF がありません。

ソースの記述を確認してください。

A2135 (E) Division by zero

0除算が行われています。

式を記述し直してください。

A2136 (E) Quote is missing

文字列に対する引用符の記述がありません。

文字列は引用符で囲って記述してください。

A2137 (E) Right quote is missing

右側の引用符がありません。

引用符を記述してください。

A2138 (E) '{' is missing

'{' の記述がありません。

'{' を記述してください。

A2139 (E) The value is not constant

値がアセンブル時確定値ではありません。

アセンブル時に確定するような、式、シンボル名またはラベル名を記述してください。

A2140 (E) Too many temporary label

テンポラリラベルの個数が多すぎます。

テンポラリラベルをラベル名に置き換えて記述してください。

A2141 (E) Temporary label is undefined

テンポラリラベルが未定義です。

テンポラリラベルの定義を行ってください。

A2142 (E) Syntax error in expression

式の記述に間違いがあります。

式の記述方法を確認して、記述し直してください。

A2143 (E) Symbol is expected

シンボルが不足しています。

シンボルの数を確認してください。

A2144 (E) Illegal macro statements

.IF とネストが交差しています。

.IF とネストが交差しないようにしてください。

A2145 (E) Invalid reserved word exist in operand

オペランド中に予約語が記述されています。

予約語は、オペランドに記述できません。オペランドを記述し直してください。

A2146 (E) Symbol has already defined as another type

シンボルは既に同一名で異なる指示命令で定義されています。指示命令.EQU と.BTEQU で同一のンボル名を定義できません。

シンボル名を変更してください。

A2147 (E) Symbol is missing

シンボルの記述がありません。

シンボル名を記述してください。

A2148 (E) Invalid bit-symbol exist

無効なビットシンボルの記述があります。

ビットシンボルの定義を記述し直してください。

A2149 (E) Operand expression is not completed

オペランド記述に不足があります。

オペランドの記述方法を確認して、記述し直してください。

A2200 (E) No '.END' statement

.END の記述がありません。

ソースプログラムの最後の行に.END を記述してください。

A2201 (E) Addressing mode specifier is not appropriate

アドレッシングモード指定子の記述に間違いがあります。

アドレッシングモード指定子の記述方法を確認してください。

A2202 (E) 'ALIGN' is multiple specified in '.SECTION'

.SECTION 定義行に複数の ALIGN 指定があります。

余分な ALIGN 指定を削除してください。

A2203 (E) Operand value is not defined

オペランドの値が未定義です。

オペランドには確定値を記述してください。

A2204 (E) Bit-symbol is in expression

式中にビットシンボルがあります。

ビットシンボルは式に記述できません。シンボル名を確認してください。

A2205 (E) Invalid bit-symbol exist

無効なビットシンボルの記述があります。

ビットシンボルの定義を記述し直してください。

A2206 (E) The value is not constant

値がアセンブル時確定値ではありません。

アセンブル時に確定するような、式、シンボル名またはラベル名を記述してください。

A2207 (E) Same items are multiple specified

オペランドの同一項目を複数指定しています。

オペランドの記述方法を確認して記述し直してください

A2208 (E) Same kind items are multiple specified

オペランドの同種の項目を複数指定しています。

オペランドの記述方法を確認して記述し直してください。

A2209 (E) Characters exist in expression

命令または式中に余分な文字があります。

式の記述規則を確認してください。

A2210 (E) Format specifier is not appropriate

フォーマット指定子の記述に間違いがあります。

フォーマット指定子の記述方法を確認してください。

A2211 (E) Symbol definition is not appropriate

シンボルの定義に間違いがあります。

シンボル定義方法を確認して記述し直してください。

A2212 (E) Invalid reserved word exist in operand

オペランド中に予約語が記述されています。

予約語は、オペランドに記述できません。オペランドを記述し直してください。

A2213 (E) 'JMP.S' operand label is not in the same section

JMP.S の分岐先が同一セクション内にありません。

JMP.S は、同一セクション内の分岐先にしか分岐できません。ニーモニックを記述し直してください。

A2214 (E) Reserved word is missing

予約語の記述がありません。

[SB],[FB],[A1],[A0],[SP]または[A1A0]を記述してください。

A2215 (E) No space after mnemonic or directive

ニーモニック、アセンブル指示命令の直後に空白文字がありません。

命令とオペランドの間に、空白文字を記述してください。

A2216 (E) No '.FB' statement

.FB の記述がありません。

8 ビット FB 相対アドレッシングモードを使用する場合は、必ず.FB でレジスタ値を仮定してください。

A2217 (E) No '.SB' statement

.SB の記述がありません。

8 ビット SB 相対アドレッシングモードを使用する場合は、必ず.SB でレジスタ値を仮定してください。

A2218 (E) No '.SECTION' statement

.SECTION の記述がありません。

ソースプログラムには、必ず1つ以上の.SECTION を記述してください。

A2219 (E) Operand value is not defined

オペランドの値が未定義です。

オペランドには確定値を記述してください。

A2220 (E) Operand size is not appropriate

オペランドのサイズが間違っています。

オペランドの記述方法を確認して、記述し直してください。

A2221 (E) Operand type is not appropriate

オペランドの種類が間違っています。

オペランドの記述方法を確認して、記述し直してください。

A2222 (E) Section attribute is not defined

セクションの属性が未定義です。このセクション内では指示命令.ALIGN は記述できません。

指示命令.ALIGN は、絶対属性セクションまたは ALIGN 指定のある相対属性セクション内に記述してください。

A2223 (E) Section has already determined as attribute

セクションは既に相対属性に確定しています。指示命令.ORG は記述できません。

セクションの属性を確認してください。

A2224 (E) Section name is missing

セクション名がありません。

オペランドにセクション名を記述してください。

A2225 (E) Section type is not appropriate

セクションタイプの記述が間違っています。

セクションタイプを記述し直してください。

A2226 (E) Section type is multiple specified

セクション定義行でセクションタイプの指定が重複しています。

セクション定義行には CODE、DATA、ROMDATA の指定は1つだけ記述してください。

A2227 (E) Size or format specifier is not appropriate

サイズ指定子またはフォーマット指定子の記述に間違いがあります。

サイズ指定子またはフォーマット指定子を記述し直してください。

A2228 (E) Size specifier is missing

サイズ指定子がありません。

サイズ指定子を記述してください。

A2229 (E) String value exist in expression

式中に文字列式が記述されています。

式を記述し直してください。

A2230 (E) Symbol is missing

オペランドにシンボルの記述がありません。

オペランドにシンボル名を記述してください。

A2231 (E) Symbol has already defined as another type

シンボルは、すでにビットシンボルとして定義されています。ビットシンボルは再定義できません。

シンボル名を変更してください。

A2232 (E) Symbol name is missing

定義するシンボルの名前がありません。

定義するシンボル名を記述してください。

A2233 (E) Symbol was already defined as the same type

シンボルは、すでにビットシンボルとして定義されています。ビットシンボルは再定義できません。

シンボル名を変更してください。

A2234 (E) Invalid operand(s) exist in instruction

一般命令に無効なオペランドがあります。

一般命令のオペランドの記述方法を確認して、記述し直してください。

A2235 (E) Syntax error in expression

式の記述に間違いがあります。

式の記述方法を確認して、記述し直してください。

A2236 (E) Invalid operand(s) exist in instruction

ビット命令に無効なオペランドがあります。

ビット命令のオペランドの記述方法を確認して、記述し直してください。

A2237 (E) Operand expression is not completed

オペランド記述に不足があります。

オペランドの記述方法を確認して、記述し直してください。

A2238 (E) Too many operand

オペランドが余分にあります。

オペランドの記述内容を確認してください。

A2239 (E) Too many operand data

オペランドのデータが多すぎます。

オペランドに記述されているデータ数が、一行に記述できる範囲を超えています。命令を複数に分けて記述してください。

A2240 (E) Undefined symbol exist

未定義のシンボルがあります。

シンボルを定義してください。

A2241 (E) Value is out of range

値が範囲外です。

レジスタなどのビット長に合った値を記述してください。

A2242 (E) Division by zero

0除算が行われています。

式を記述し直してください。

A2243 (E) '.VER' is duplicated

.VER が重複指定されています。

.VER は、1つのファイルに1回だけ記述できます。余分な.VER を削除してください。

A2244 (E) '#' is missing

'#' の記述がありません。

本オペランドには、イミディエイト値を記述してください。

A2245 (E) ',' is missing"

',' の記述がありません。

オペランドの区切りには、カンマを記述してください。

A2246 (E) ']' is missing

']' の記述がありません。

'[' に対応する ']' を記述してください。

A2247 (E) ')' is missing

')' の記述がありません。

'(' に対応する ')' を記述してください。

A2248 (E) Symbol defined by external reference data is defined as global symbol

グローバルシンボルに外部参照値により定義されたシンボルを用いています。

シンボル定義およびシンボル名を確認してください。

A2250 (E) Quote is missing

文字列に対する引用符の記述がありません。

文字列は引用符で囲って記述してください。

A2251 (E) Right quote is missing

右側の引用符がありません。

引用符を記述してください。

A2252 (E) Revision information mismatch in file

オブジェクトファイルのリビジョン情報が異なります。

オブジェクトファイルを作成したアセンブラのバージョンまたはオプションを確認してください。

A2253 (E) Invalid indirect operand(s) exist in operand

オペランドに無効な間接アドレッシング指定があります。

命令のオペランドの記述方法を確認して、記述し直してください。

A2254 (E) Illegal directive command is used

不正な指示命令を記述しています。

正しい指示命令に記述し直してください。

A2255 (E) '.EINSF' is missing for '.INSF'

.INSF に対する.EINSF がありません。

.INSF の記述位置を確認してください。

A2256 (E) '.INSF' is missing for '.EINSF'

.EINSF に対する.INSF がありません。

.EINSF の記述位置を確認してください。

A2258 (E) Invalid operand(s) exist in debug information

デバッグ情報に無効なオペランドがあります。

デバッグ情報を確認して、記述し直してください。

A2259 (E) Invalid mnemonic which isn't supported in '-R8C'

-R8C オプション指定時に使用できない命令が記述されています。

記述内容を確認してください。

A2260 (E) '.PROTECT' or '.OFSREG' is duplicated

.PROTECT または.OFSREG が重複指定されています。

.PROTECT または.OFSREG は1つのファイルに1回だけ記述できます。余分な.PROTECT または.OFSREG を削除してください。

A2261 (E) '.ID' is duplicated

.ID が重複指定されています。

.ID は1つのファイルに1回だけ記述できます。余分な.ID を削除してください。

A2262 (E) Section name is not appropriate

セクション名が間違っています。

セクション名を変更してください。

A2263 (E) Interrupt number was already defined

ソフトウエア割り込み番号はすでに定義されています。

ソフトウエア割り込み番号を変更してください。

A2264 (E) Special page number was already defined
スペシャルページ番号はすでに定義されています。
スペシャルページ番号を変更してください。

A2265 (E) Comm symbol has already defined as another type
コモンシンボルは既に他の定義で確定しています。
シンボルの定義を確認してください。

A2266 (E) Comm symbol has already defined as differ size
コモンシンボルは既に他のサイズで確定しています。
シンボルの定義を確認してください。

A2267 (E) Different register of the bank exist
異なるバンクのレジスタが記述されています。
バンクを確認してください。

A2268 (E) The addressing which can't be modified is specified
モディファイすることのできないアドレッシングが指定されています。
指示命令.INXxx の記述を確認してください。

A2269 (E) Can't use directive commands '.INXxx'
指示命令.INXxx は記述できません。
指示命令の直後に記述した命令を確認してください。

A2270 (E) Can't use directive commands '.INXLx' or '.INXBx'
指示命令.INXLx または.INXBx は記述できません。
指示命令の直後に記述した命令を確認してください。

A2271 (E) Can't use directive commands '.INXRx' or '.INXBx'
指示命令.INXRx または.INXBx は記述できません。
指示命令の直後に記述した命令を確認してください。

A2272 (E) Can't use directive commands '.INXBx'
指示命令.INXBx は記述できません。
指示命令の直後に記述した命令を確認してください。

A2273 (E) No '.LBBA' statement
.LBBA の記述がありません。
相対番地指定を行う場合は、必ず.LBBA でレジスタ値を仮定してください。

A2274 (E) Directive command '.RVECTOR' can't be described
指示命令.RVECTOR は記述できません。
可変ベクタテーブルの自動生成を行う場合、vector セクションにプログラムを記述しないでください。

A2275 (E) Directive command '.SVECTOR' can't be described
指示命令.SVECTOR は記述できません。
スペシャルページベクタテーブルの自動生成を行う場合、svector セクションにプログラムを記述しないでください。

A2276 (E) Invalid directive commnad which isn't supported in '-R8C'

-R8C オプションと同時に指定することのできない指示命令が記述されています。
記述内容を確認してください。

A2278 (E) Initialization function definition of the section is not appropriate

C 言語スタートアップ使用時のセクション初期化関数定義に間違いがあります。
セクション初期化関数定義の内容を確認してください。

A2279 (E) Invalid directive commnad '.SB_AUTO'

指示命令.SB_AUTO の定義に間違いがあります。
セクション初期化関数定義の内容を確認してください。

A2281 (E) Symbol has already defined as static type

シンボルが static で宣言されています。
指示命令.GLB の記述を削除してください。

A2300 (E) Operand size is not appropriate

オペランドのサイズが間違っています。
オペランドの記述方法を確認して、記述し直してください。

A2301 (E) Value is out of range

値が範囲外です。
レジスタなどのビット長に合った値を記述してください。

A2302 (E) Illegal operand is used

オペランドが間違っています。
オペランドの記述方法を確認して、記述し直してください。

A2303 (E) Addressing mode specifier is not appropriate

アドレッシングモード指定子の記述に間違いがあります。
アドレッシングモード指定子の記述方法を確認してください。

A2304 (E) Illegal directive command

不正な指示命令を記述しています。
正しい指示命令に記述し直してください。

A2305 (E) Invalid label definition

無効なラベル記述をしています。
ラベル定義を記述し直してください。

A2306 (E) Invalid symbol definition

無効なシンボル記述をしています。
シンボルの定義を記述し直してください。

A2308 (E) Questionable syntax

構造化記述命令の記述が間違っています。
記述方法を確認して記述し直してください。

A2311 (E) ELSE not associates with IF
ELSE に対する IF がありません。
ソースの記述を確認してください。

A2312 (E) ELIF not associates with IF
ELIF に対する IF がありません。
ソースの記述を確認してください。

A2313 (E) ENDIF not associates with IF
ENDIF に対する IF がありません。
ソースの記述を確認してください。

A2314 (E) NEXT not associates with FOR
NEXT に対する FOR がありません。
ソースの記述を確認してください。

A2315 (E) WHILE not associates with DO
WHILE に対する DO がありません。
ソースの記述を確認してください。

A2316 (E) ENDS not associates with SWITCH
ENDS に対する SWITCH がありません。
ソースの記述を確認してください。

A2317 (E) BREAK' is missing for 'FOR', 'DO' or 'SWITCH'
BREAK の使用箇所が不適当です。
BREAK 命令は、FOR、DO または SWITCH 文内に記述してください。

A2318 (E) 'CONTINUE' is missing for 'FOR' or 'DO'
CONTINUE の使用箇所が不適当です。
CONTINUE は FOR または DO 文内に記述してください。

A2320 (E) CASE not inside SWITCH
SWITCH 文以外で CASE 記述があります。
CASE 文は SWITCH 文内に記述してください。

A2321 (E) DEFAULT not inside SWITCH
SWITCH 文以外で DEFAULT 記述があります。
DEFAULT 文は SWITCH 文内に記述してください。

A2322 (E) Symbol is multiple defined
シンボルが二重定義です。マクロ名と他の名前が重複しています。
名前を変更してください。

A2324 (E) Undefined symbol exist
未定義のシンボルがあります。
シンボルを定義してください。

A2325 (E) Division by zero

0除算が行われています。

式を記述し直してください。

A2326 (E) DEFAULT' has already defined

SWITCH 中に DEFAULT が複数あります。

余分な DEFAULT 文を削除してください。

A2327 (E) Section type is not appropriate

セクションタイプの記述が間違っています。

セクションタイプを記述し直してください。

A2328 (E) Operand value is not defined

オペランドの値が未定義です。

オペランドには確定値を記述してください。

A2329 (E) Symbol has already defined as another type

シンボルは、すでにビットシンボルとして定義されています。ビットシンボルは再定義できません。

シンボル名を変更してください。

A2331 (E) No 'ENDIF' statement

ソースファイル内に IF 文に対応した ENDIF がありません。

ソースの記述を確認してください。

A2332 (E) No 'ENDS' statement

ソースファイル内に SWITCH 文に対応した ENDS がありません。

ソースの記述を確認してください。

A2333 (E) No 'NEXT' statement

ソースファイル内に FOR 文に対応した NEXT がありません。

ソースの記述を確認してください。

A2334 (E) No 'WHILE' statement

ソースファイル内に DO 文に対応した WHILE がありません。

ソースの記述を確認してください。

A2335 (E) 'CASE' has already defined as same value

同一値が複数の CASE 文のオペランドに記述されています。

CASE のオペランドに記述する値は重複しないように記述してください。

A2336 (E) Statement not preceded by 'CASE' or 'DEFAULT'

SWITCH 文において CASE または DEFAULT より先に命令行があります。

命令行は必ず CASE および DEFAULT 文の後に記述してください。

A2337 (E) Symbol is missing

シンボルの記述がありません。

シンボル名を記述してください。

A2338 (E) Size or Format specifier is not appropriate

サイズ指定子またはフォーマット指定子の記述に間違いがあります。

サイズ指定子またはフォーマット指定子を記述し直してください。

A3001 (F) Not enough memory

メモリが足りません。

メモリを増設してください。

A3002 (F) Invalid option 'option' is in environment data

無効なコマンドオプション'option'が環境変数にあります。

環境変数を設定し直してください。環境変数に設定可能なオプションは、"L,N,P,S,T"です。

A3003 (F) Can't open file 'filename'

'filename'ファイルがオープンできません。

ファイル名を確認してください。

A3004 (F) Error occurred in executing 'xxx'

xxx の実行でエラーが発生しました。

環境変数 LIB30 が設定されているか確認してください。

A3005 (F) Can't create Temporary file

テンポラリファイルが生成できません。

カレントディレクトリ以外にテンポラリファイルを作成するように、環境変数にディレクトリを指定してください。

A3006 (F) Illegal file name 'filename'

ファイル名が不正です。

ファイル名の記述規則に従ったファイル名を指定してください。

A3007 (F) Can't find work dir

カレントディレクトリ情報が取得できません。

再度アセンブラを実行してください。

A3101 (F) No input files specified

入力ファイルの指定がありません。

入力ファイルを指定してください。

A3102 (F) Invalid option 'option' is used

無効なオプション'option'を使用しています。

オプションを指定し直してください。

A3103 (F) Ignore option 'option'

無効なオプション'option'を指定しています。

指定したオプションは存在しません。コマンドを入力し直してください。

A3104 (F) Not enough memory

メモリが足りません。

ファイルを分割して実行し直してください。またはメモリを増設してください。

A3105 (F) Too many source files

指定したファイルの数が多すぎます。

ファイル数を 80 以下にしてください。複数回にわけてアセンブルを実行してください。

A3106 (F) Can't open file 'filename'

'filename'ファイルがオープンできません。

ファイル名を確認してください。

A3107 (F) Can't create Temporary file

テンポラリファイルが生成できません。

カレントディレクトリ以外にテンポラリファイルを作成するように、環境変数にディレクトリを指定してください。

A3108 (F) Can't write file 'filename'

'filename'ファイルに書き込むことができません。

ファイル名等を確認してください。

A3109 (F) Can't create file 'filename'

'filename'ファイルが生成できません。

ディレクトリ容量を確認してください。

A3110 (F) Command line is too long

コマンド行の文字数が多すぎます。

コマンドを入力し直してください。

A3201 (F) Can't open file

ファイルがオープンできません。

ファイル名を確認してください。

A3202 (F) Can't create file

ファイルが生成できません。

ディレクトリ容量を確認してください。

A3203 (F) Can't read file

ファイルを読み込むことができません。

ファイルのパーミッションを確認してください。

A3204 (F) Can't write file

ファイルに書き込むことができません。

ファイルのパーミッションを確認してください。

A3205 (F) Illegal file name

ファイル名が不正です。

ファイル名の記述規則に従ったファイル名を指定してください。

A3206 (F) Not enough memory

メモリが足りません。

ファイルを分割して実行し直してください。またはメモリを増設してください。

A3207 (F) Can't open Temporary file

テンポラリファイルが生成できません。

カレントディレクトリ以外にテンポラリファイルを作成するように、環境変数にディレクトリを指定してください。

A3208 (F) Can't create Temporary file

テンポラリファイルが生成できません。

カレントディレクトリ以外にテンポラリファイルを作成するように、環境変数にディレクトリを指定してください。

A3209 (F) Can't read Temporary file

ファイルを読み込むことができません。

ファイルのパーミッションを確認してください。

A3210 (F) Can't write Temporary file

テンポラリファイルに書き込むことができません。

ファイルのパーミッションを確認してください。

A3212 (F) Can't open file 'VERSION.txt' in environment variable LIB30

環境変数 LIB30 に VERSION.txt がありません。

環境変数 LIB30 を確認してください。

A3213 (F) Invalid format of 'VERSION.txt' in environment variable LIB30

環境変数 LIB30 に存在する VERSION.txt の記載内容が正しくありません。

VERSION.txt を削除し、上書きインストールしてください。

A3304 (F) Not enough memory

メモリが足りません。

ファイルを分割して実行し直してください。またはメモリを増設してください。

A3306 (F) Can't open file 'filename'

'filename'ファイルがオープンできません。

ファイル名を確認してください。

A3307 (F) Can't create Temporary file 'filename'

'filename'ファイルが生成できません。

ディレクトリ容量を確認してください。

A3308 (F) Can't write in file 'filename'

'filename'ファイルに書き込むことができません。

ファイルのパーミッションを確認してください。

A3309 (F) Can't create file 'filename'

'filename'ファイルが生成できません。

ディレクトリ容量を確認してください。

A4200 (-) Internal error

アセンブラの処理中に内部的な問題が発生しました。

メッセージ内の内部エラー番号、ファイル、行番号、コメントを添えて、販売元のサポートセンターまでご連絡ください。

12. 最適化リンケージエディタのエラーメッセージ

12.1 エラー形式とエラーレベル

本章では、以下の形式で出力するエラーメッセージとエラー内容を説明します。

エラー番号 (エラーレベル) エラーメッセージ
エラー内容

エラーレベルは、エラーの重要度に従い、5 種類に分類されます。

エラー番号	エラーレベル	エラータイプ	動作
L0000 – L0999 P0000 – P0999	(I)	インフォメーション	処理を継続します。
L1000 – L1999 P1000 – P1999	(W)	ウォーニング	処理を継続します。
L2000 – L2999 P2000 – P2999	(E)	エラー	オプション解析処理を継続し、処理を中断します。
L3000 – L3999 P3000 – P3999	(F)	フェータル	処理を中断します。
L4000 – P4000 –	(–)	インターナル	処理を中断します。

L で始まるエラー番号は、最適化リンケージエディタ出力メッセージです。
P で始まるエラー番号は、プレリンカ出力メッセージです。P で始まるエラー番号は、nomessage オプションや change_message オプションで指定できません。

12.2 エラーの返値

最適化リンケージエディタは処理を終了した際、処理結果によって次の値を OS に返します。

返値	内容
0	正常終了、インフォメーションメッセージ出力後終了、およびウォーニング出力後終了します。
1	エラー、フェータル、インターナルおよび強制終了します。

12.3 メッセージ一覧

L0001 (I) Section "セクション" created by optimization "最適化"

"最適化"の最適化によって、"セクション"を作成しました。

L0002 (I) Symbol "シンボル" created by optimization "最適化"

"最適化"の最適化によって、"シンボル"を作成しました。

L0003 (I) "ファイル"-"シンボル" moved to "セクション" by optimization

`variable_access` の最適化によって、"ファイル"内の"シンボル"を移動しました。

L0004 (I) "ファイル"-"シンボル" deleted by optimization

`symbol_delete` の最適化によって、"ファイル"内の"シンボル"を削除しました。

L0005 (I) The offset value from the symbol location has been changed by optimization :

"ファイル"-"セクション"-"シンボル±offset"

"シンボル±`offset`"の範囲で最適化によるサイズ変更があったため `offset` 値を変更しました。問題ないか確認してください。`offset` 値の変更を抑止したい場合は、"ファイル"のアセンブル時に `goptimize` オプション指定を外してください。

L0100 (I) No inter-module optimization information in "ファイル"

"ファイル"内にモジュール間最適化情報がありません。"ファイル"をモジュール間最適化の対象外にします。モジュール間最適化の対象にする場合は、コンパイル、アセンブル時に `goptimize` オプションを指定してください。ただし、`asmsh` には `goptimize` オプションはありません。

L0101 (I) No stack information in "ファイル"

"ファイル"内にスタック情報がありません。"ファイル"はアセンブラ出力ファイルまたは SYSROF->ELF コンバートファイルの可能性があります。最適化リンケージエディタが出力するスタック情報ファイルに当該ファイルの内容は含まれません。

L0102 (I) Stack size "サイズ" specified to the undefined symbol "シンボル" in "ファイル"

"ファイル"内の未定義シンボル"シンボル"に、スタックサイズ "サイズ" が指定されています。

L0103 (I) Multiple stack sizes specified to the symbol "シンボル"

シンボル"シンボル"は、複数のスタックサイズが指定されています。

L0300 (I) Mode type "モード種別１" in "ファイル" differ from "モード種別２"

異なるモード種別のファイルを入力しました。

L0400 (I) Unused symbol "ファイル"-"シンボル"

"ファイル"内の"シンボル"は使用されていません。

L0500 (I) Generated CRC code at "アドレス"

"アドレス"に CRC コードを出力しました.

L0510 (I) Section "セクション" was moved other area specified in option "cpu=<メモリ属性>"
セクションを分割せずに cpu=<メモリ属性>にしたがって"セクション"を配置しました。

L0511 (I) Sections "セクション名","分割後のセクション名" are Non-contiguous
"セクション名"のセクションを分割し、"分割後のセクション名"のセクションを生成しました。

L1000 (W) Option "オプション" ignored
"オプション"は無効です。"オプション"を無視します。

L1001 (W) Option "オプション 1" is ineffective without option "オプション 2"
"オプション 1"は"オプション 2"が必要です。"オプション 1"を無視します。

L1002 (W) Option "オプション 1" cannot be combined with option "オプション 2"
"オプション 1"と"オプション 2"は同時に指定できません。"オプション 1"を無視します。

L1003 (W) Divided output file cannot be combined with option "オプション"
"オプション"指定時、出力ファイルの分割指定はできません。オプションの指定を無視します。先頭入力ファイル名を出力ファイル名として使用します。

L1004 (W) Fatal level message cannot be changed to other level : "番号"
Fatal レベルメッセージはレベル変更できません。"番号"の指定を無視します。change_message オプションで変更できるエラーは、Information/Warning/Error レベルです。

L1005 (W) Subcommand file terminated with end option instead of exit option
end オプションの後に処理指定がありません。exit オプションを仮定して処理します。

L1006 (W) Options following exit option ignored
exit オプションの後のオプションを無視しました。

L1007 (W) Duplicate option : "オプション"
"オプション"が重複しています。最後に指定したオプションを有効にします。

L1008 (W) Option "オプション" is effective only in cpu type "マイコン種別"
"オプション"は"マイコン種別"以外では無効です。"オプション"を無視します。

L1010 (W) Duplicate file specified in option "オプション" : "ファイル名"
"オプション"で同じファイルを 2 度指定しました。2 度目の指定を無視します。

L1011 (W) Duplicate module specified in option "オプション" : "モジュール"
"オプション"で同じモジュールを 2 度指定しました。2 度目の指定を無視します。

L1012 (W) Duplicate symbol/section specified in option "オプション" : "名前"
"オプション"で同じシンボル名またはセクション名を 2 度指定しました。2 度目の指定を無視します。

L1013 (W) Duplicate number specified in option "オプション" : "番号"

"オプション"で同じエラー番号を指定しました。最後に指定した方を有効にします。

L1100 (W) Cannot find "名前" specified in option "オプション"

"オプション"で指定したシンボル名またはセクション名が見つかりません。"名前"の指定を無視します。

L1101 (W) "名前" in rename option conflicts between symbol and section

`rename` オプションで指定した"名前"がセクション名とシンボル名の両方に存在します。
シンボル名を変更の対象にします。

L1102 (W) Symbol "シンボル" redefined in option "オプション"

"オプション"で指定したシンボルはすでに定義されています。そのまま処理を続けます。

L1103 (W) Invalid address value specified in option "オプション" : "アドレス"

"オプション"で指定した"アドレス"は無効な値です。"アドレス"の指定を無視します。

L1104 (W) Invalid section specified in option "オプション" : "セクション"

"オプション"で無効なセクションを指定しています。以下を確認してください。

(1) -outputオプションは、初期値のないセクションを指定できません。

(2) -jump_entries_for_picオプションは、コードセクション以外を指定できません。

L1110 (W) Entry symbol "シンボル" in entry option conflicts

`entry` オプションで指定した"シンボル"以外のシンボルがコンパイル、アセンブル時にエントリシンボルとして指定されています。オプション指定を優先します。

L1120 (W) Section address is not assigned to "セクション"

"セクション"のアドレス指定がありません。"セクション"を最後尾に配置します。
`optlnk` オプション`-start` を使用して、セクションのアドレスを設定してください。

L1121 (W) Address cannot be assigned to absolute section "セクション" in start option

"セクション"は絶対アドレスセクションです。絶対アドレスセクションに対するアドレス指定を無視します。

L1122 (W) Section address in start option is incompatible with alignment : "セクション"

`start` オプションで指定した"セクション"のアドレスはアライメント数と矛盾しています。アライメント数に合わせてセクションアドレスを補正します。

L1130 (W) Section attribute mismatch in rom option : "セクション 1,セクション 2"

`rom` オプションで指定した"セクション 1"と"セクション 2"の属性、アライメント数が異なります。"セクション 2"のアライメント数はどちらか大きい方を有効とします。

L1140 (W) Load address overflowed out of record-type in option "オプション"

アドレス値よりも小さい `record` 形式を指定しました。指定した `record` 形式を超える範囲は、別の `record` 形式で出力します。

L1141 (W) Cannot fill unused area from "アドレス" with the specified value

空きエリアのサイズが space オプションで指定された値の倍数となっていないため、"アドレス"以降に指定データを出力できませんでした。

1150 (W) Sections in "オプション" option have no symbol

"オプション" で指定したセクションは外部定義シンボルがありません。

L1160 (W) Undefined external symbol "シンボル"

未定義の"シンボル"を参照しています。

L1170 (W) Specified SBR addresses conflict

異なる複数の SBR アドレスが指定されました。SBR=USER として処理します。

L1171 (W) Least significant byte in SBR="定数" ignored

SBR オプションで指定されたアドレス"定数"の下位 8bit は無効です。

L1180 (W) Directive command "制御命令" is duplicated in "ファイル"

複数のソースファイルに、"制御命令"を記述しています。
"制御命令"は、複数記述することはできません。

L1181 (W) Fail to write "出力コード種別"

出力ファイルへの、"出力コード種別"の書き込みが失敗しました。
出力ファイルに、"出力コード種別"の書き込み先アドレスが含まれていない可能性があります。
出力コード種別:

ID コード書き込み失敗時・・・「"ID Code"」
→ L1181 Fail to write "ID Code"

PROTECT/OFSREG コード書き込み失敗時・・・「"Protect Code" or "OFSREG Code"」
→ L1181 Fail to write "Protect Code" or "OFSREG Code"

CRC コード書き込み失敗時・・・「"CRC Code"」
→ L1181 Fail to write "CRC Code"

L1182 (W) Cannot generate vector table section "セクション"

入力ファイル内に、ベクタテーブル"セクション"があります。リンカは、"セクション"を自動生成しません。

L1183 (W) Interrupt number "ベクタ番号" of "セクション" is defined in input file

VECTN オプションで記述したベクタ番号は、入力ファイル内で定義済みです。入力ファイルの内容を優先して、処理を継続します。

L1190 (W) Section "セクション" was moved other area specified in option "cpu=<メモ属性>"

外部変数アクセス最適化によりオブジェクトサイズが変更されたため、次の cpu 指定範囲の"セクション"を移動しました。

L1191 (W) Area of "FIX" is within the range of the area specified by “cpu=<メモリ属性>” :"<start>-<end>"

cpu オプションで、メモリ属性 FIX と FIX 以外の<start>-<end>範囲が重複していたため、FIX を有効にしました。

L1192 (W) Bss Section "セクション名" is not initialized

初期値なしのデータセクション"セクション名"は、初期設定プログラムで初期化できません。-cpu 指定範囲、ポインタ変数のサイズ指定を見直してください。

L1193 (W) Section "セクション名" specified in option "オプション" is ignored

-cpu=stride の機能で分割したセクションの、後半部への"オプション"指定は無効となります。後半部のセクションは"オプション"で指定しないでください。

L1194 (W) Section "セクション" in relocation "ファイル"-"セクション"-"オフセット" is changed.

"セクション""ファイル""オフセット" の位置にある"セクション"を参照していたリロケーションが、分割した後半セクションを参照するよう変更しました。分割しないようにするには、"セクション"を contiguous_section オプションで指定してください。

L1200 (W) Backed up file "ファイル 1" into "ファイル 2"

入力ファイル"ファイル 1"は書き換えられました。書き換える前の"ファイル 1"の内容は"ファイル 2"にバックアップされています。

L1300 (W) No debug information in input files

入力ファイル内にデバッグ情報がありません。debug,sdebug,compress オプション指定を無視します。コンパイル、アセンブル時に該当するオプションを指定しているか確認してください。

L1301 (W) No inter-module optimization information in input files

入力ファイル内にモジュール間最適化情報がありません。optimize オプションを無視します。コンパイル、アセンブル時に goptimize オプションを指定してください。

L1302 (W) No stack information in input files

入力ファイル内にスタック情報がありません。stack オプションを無視します。入力ファイルがアセンブラ出力ファイルまたは SYSROF->ELF コンバートファイルの場合は、stack オプションは無効です。

L1303 (W) No rts information in input files

.rts ファイルを生成可能な入力ファイルがありません。
.rts ファイルを生成せずに処理を終了します。

L1304 (W) No utl information in input files

utl ファイルを生成するための情報が、全く入力されませんでした。

L1305 (W) Entry address in "ファイル" conflicts : "アドレス"

異なるエントリーアドレスのファイルが複数入力されています。

L1310 (W) "セクション" in "ファイル" is not supported in this tool

"ファイル"内に非サポートセクションがありました。"セクション"を無視します。

L1311 (W) Invalid debug information format in "ファイル"

"ファイル"内のデバッグ情報は dwarf2 ではありません。debug 情報を削除します。

L1320 (W) Duplicate symbol "シンボル" in "ファイル"

"シンボル"は重複しています。先に入力したファイル内シンボルを優先します。

L1321 (W) Entry symbol "シンボル" in "ファイル" conflicts

エントリシンボル定義のあるオブジェクトファイルを複数入力しました。先に入力したファイル内のエントリシンボルを有効にします。

L1322 (W) Section alignment mismatch : "セクション"

アライメント数の異なる同名セクションを入力しました。アライメント数は最大の指定を有効にします。

L1323 (W) Section attribute mismatch : "セクション"

属性の異なる同名セクションを入力しました。絶対セクションと相対セクションの場合は、絶対セクションとして扱います。`read/write` 属性が異なる場合は、どちらも許可します。

L1324 (W) Symbol size mismatch : "シンボル" in "ファイル"

サイズの異なるコモンシンボルまたは定義シンボルが入力されました。定義シンボルを優先します。コモンシンボル同士の場合は、先に入力したファイル内シンボルを優先します。

L1325 (W) Symbol attribute mismatch : "シンボル" : "ファイル"

"ファイル"内の"シンボル"が、他のファイルの同名シンボルと属性が一致していません。シンボルを確認してください。

L1326 (W) Reserved symbol "シンボル" is defined in "ファイル"

予約された名称のシンボル"シンボル"が"ファイル"内で定義されています。

L1327(W) Section alignment in option "aligned_section" is small : "セクション"

`aligned_section` オプション指定時のアライメント数 16 の方が、"セクション"のアライメント数より小さいため、指定セクションに対するオプション指定を無視します。

L1330 (W) Cpu type "マイコン種別１" in "ファイル" differ from "マイコン種別２"

異なるマイコン種別のファイルを入力しました。マイコン種別を H8SX として処理を継続します。

L1400 (W) Stack size overflow in register optimization

レジスタ最適化で、スタックアクセスコードがコンパイラのスタック量制限値を超えました。レジスタ最適化指定を無視します。

L1401 (W) Function call nest too deep

関数の呼び出しネストが深すぎるため、レジスタ最適化を実施できません。

L1402 (W) Parentheses specified in option "start" with optimization

`start` オプションで括弧`"()"`を記述した場合、最適化機能は使用できません。
最適化機能を無効にします。

L1410 (W) Cannot optimize "ファイル"-"セクション" due to multi label relocation operation

複数ラベルのリロケーション演算を持つセクションは最適化できません。"ファイル"内の"セクション"を最適化対象外にします。

L1420 (W) "ファイル" is newer than "プロファイル"

"ファイル"は"プロファイル"より後に更新されました。プロファイル情報を無視します。

L1430 (W) Cannot generate effective bls file for compiler optimization

無効な bls ファイルが生成されました。コンパイル時に、外部変数アクセス最適化(map オプション)を指定しても、この最適化は実施できません。

コンパイラの外部変数アクセス最適化(map オプション)には、以下の制限があります。該当する内容がないかを確認し、セクション配置を見直してください。

コンパイル時に base オプションを使用している場合、コードセクションの直後にデータセクションを配置すると、外部変数アクセス最適化が実施できない場合があります。

※bls ファイルは"外部シンボル割り付け情報ファイル"を指します。コンパイラの map オプションに使用するための情報ファイルです。

L1500 (W) Cannot check stack size

スタックセクションがないため、コンパイル時の stack オプションで指定したスタックサイズの整合性をチェックできません。コンパイル時の stack オプションの整合性をチェックするためにはコンパイル時、アセンブル時に goptimize オプション指定が必要です。

L1501 (W) Stack size overflow : "スタックサイズ"

スタックセクションサイズが、コンパイル時に stack オプションで指定した"スタックサイズ"を超えました。コンパイル時のオプションを変更するか、スタック量を削減できるようにプログラムを変更してください。

L1502 (W) Stack size in "ファイル" conflicts with that in another file

複数のファイルで異なるスタックサイズを指定されています。コンパイル時のオプションを確認してください。

L1510 (W) Input file was compiled with option "smap" and option "map" is specified at linkage

"smap を指定してコンパイルしたファイルがあります。smap を指定したファイルは、2 回目のビルドで map オプションを指定してコンパイルしないでください。

P1600 (W) An error occurred during name decoding of "インスタンス"

"インスタンス"はデコードできませんでした。エンコード名でメッセージ出力します。

L2000 (E) Invalid option : "オプション"

P2000 (E) Invalid option : "オプション"

"オプション"はサポートしていません。

L2001 (E) Option "オプション" cannot be specified on command line

"オプション"はコマンドライン上では指定できません。サブコマンドファイル内で指定してください。

L2002 (E) Input option cannot be specified on command line

コマンドライン上で input オプションを指定しました。コマンドライン上での入力ファイル指定は input オプション無しで指定してください。

L2003 (E) Subcommand option cannot be specified in subcommand file

サブコマンドファイル内に subcommand オプションを指定しました。subcommand オプションはネストできません。

L2004 (E) Option "オプション 1" cannot be combined with option "オプション 2"

"オプション 1"と"オプション 2"は同時に指定できません。

L2005 (E) Option "オプション" cannot be specified while processing "プロセス"

"プロセス"処理に対して"オプション"は指定できません。

L2006 (E) Option "オプション 1" is ineffective without option "オプション 2"

"オプション 1"は"オプション 2"が必要です。

L2010 (E) Option "オプション" requires parameter

"オプション"はパラメータ指定が必要です。

L2011 (E) Invalid parameter specified in option "オプション" : "パラメータ"

"オプション"で無効なパラメータを指定しました。

L2012 (E) Invalid number specified in option "オプション" : "値"

"オプション"指定で無効な値を指定しました。値の範囲を確認してください。

L2013 (E) Invalid address value specified in option "オプション" : "アドレス"

"オプション"で指定した"アドレス"は無効な値です。0～FFFFFFFF の間の 16 進数で指定してください。

L2014 (E) Illegal symbol/section name specified in "オプション" : "名前"

"オプション"で指定したセクションまたはシンボル名に不正文字が使用されています。セクション/シンボル名で使用できるのは数字、英字、_、$(先頭は数字以外)です。

L2016 (E) Invalid alignment value specified in option "オプション" : "アライメント数"

"オプション"で指定した"アライメント数"は無効な値です。
1,2,4,8,16 または 32 を指定してください。

L2017 (E) Cannot output “セクション” specified in option “オプション”

"オプション"で指定した"セクション"のコードの一部を出力できません。命令コードのエンディアンを変換したことにより、"セクション"内命令コードの一部が非連続となりました。非連続部分の命令コードが属しているセクションは、リンケージリストからセクションアドレスを 4 バイト境界で確認の上、出力するセクションがどのセクションとエンディアン変換を行っているか確認してください。

L2020 (E) Duplicate file specified in option "オプション" : "ファイル"
"オプション"指定で同じファイルを 2 度指定しました。

L2021 (E) Duplicate symbol/section specified in option "オプション" : "名前"
"オプション"指定で同じシンボル名またはセクション名を 2 度指定しました。

L2022 (E) Address ranges overlap in option "オプション" : "アドレス範囲"
"オプション"で指定した"アドレス範囲"が重複しています。

L2100 (E) Invalid address specified in cpu option : "アドレス"
cpu オプションで cpu では指定できないアドレスを指定しました。

L2101 (E) Invalid address specified in option "オプション" : "アドレス"
"オプション"で指定した"アドレス"は cpu で指定できるアドレス範囲、または cpu オプションで指定した範囲を超えました。

L2110 (E) Section size of second parameter in rom option is not 0 : "セクション"
rom オプションの第 2 パラメータにサイズが 0 でない"セクション"を指定しました。

L2111 (E) Absolute section cannot be specified in rom option : "セクション"
rom オプションで絶対アドレスセクションを指定しました。

L2112 (E) "セクション 1" and "セクション 2" cannot mapped as ROM/RAM in "ファイル"
“ファイル名”で指定された"セクション 1"と"セクション 2"は ROM/RAM 連結となりません。

L2113 (E) Option "rom" and internal information in the file are conflicted
rom オプションの指定と内部情報が矛盾しています。

L2120 (E) Library "ファイル" without module name specified as input file
入力ファイルとしてモジュール名なしのライブラリファイルを指定しました。

L2121 (E) Input file is not library file : "ファイル(モジュール)"
入力ファイルで指定した"ファイル(モジュール)"はライブラリファイルではありません。

L2130 (E) Cannot find file specified in option "オプション" : "ファイル"
"オプション"で指定したファイルが見つかりません。

L2131 (E) Cannot find module specified in option "オプション" : "モジュール"
"オプション"で指定したモジュールがありません。

L2132 (E) Cannot find "名前" specified in option "オプション"
"オプション"で指定したシンボルまたはセクションが存在しません。

L2133 (E) Cannot find defined symbol "名前" in option "オプション"
"オプション"で指定した外部定義シンボルが存在しません。

L2140 (E) Symbol/section "名前" redefined in option "オプション"
"オプション"で指定したシンボル、セクションはすでに定義されています。

L2141 (E) Module "モジュール" redefined in option "オプション"
"オプション"で指定したモジュールはすでに登録されています。

L2142 (E) Interrupt number "ベクタ番号" of "セクション" has multiple definition
ベクタテーブル"セクション"の、ベクタ番号定義が複数入力されました。ベクタ番号には、ひとつのアドレスしか設定できません。ソースファイルの記述を見直してください。

L2143 (E) Invalid vector number specified : "number"
`number` で示すベクタ番号は指定できません。
`#pragma special` で指定したベクタ番号を見直してください。

L2200* (E) Illegal object file : "ファイル"
`ELF` フォーマット以外を入力しました。
`* P2200` と表示される場合があります。

L2201 (E) Illegal library file : "ファイル"
"ファイル"はライブラリファイルではありません。

L2202 (E) Illegal cpu information file : "ファイル"
"ファイル"はマイコン情報ファイルではありません。

L2203 (E) Illegal profile information file : "ファイル"
"ファイル"はプロファイル情報ファイルではありません。

L2210 (E) Invalid input file type specified for option "オプション" : "ファイル(種別)"
"オプション"指定時に処理できない"ファイル(種別)"を入力しました。

L2211 (E) Invalid input file type specified while processing "プロセス" : "ファイル(種別)"
"プロセス"処理に対して処理できない"ファイル(種別)"を入力しました。

L2212 (E) "オプション" cannot be specified for inter-module optimization information in "ファイル"
"ファイル"内にモジュール間最適化情報があるため、"オプション"オプションは使用できません。コンパイル、アセンブル時に `goptimize` オプションを使用しないでください。

L2220 (E) Illegal mode type "モード種別" in "ファイル"
異なる"モード種別"のファイルを入力しました。

L2221 (E) Section type mismatch : "セクション"
属性(初期値有無)の異なる同名セクションを入力しました。

L2223 (E) Cpu type "CPU 種別 1" in "ファイル" is incompatible with "CPU 種別 2"
異なる CPU 種別を入力しました。
一部の仕様に互換性がないため、リンクしても動作が保証できません。

L2300 (E) Duplicate symbol "シンボル" in "ファイル"
"シンボル"は重複しています。

L2301 (E) Duplicate module "モジュール" in "ファイル"
"モジュール"は重複しています。

L2310 (E) Undefined external symbol "シンボル" referenced in "ファイル"
"ファイル"内で未定義の"シンボル"を参照しています。

L2311 (E) Section "セクション 1" cannot refer to overlaid section : "セクション 2"-"シンボル"
同一アドレスを指定したオーバーレイセクション間でシンボル参照がありました。
"セクション 1"と"セクション 2"を同じアドレスに割り付けないでください。

L2320 (E) Section address overflowed out of range : "セクション"
"セクション"のアドレスが使用可能なアドレス範囲を超えました。

L2321 (E) Section "セクション１" overlaps section "セクション２"
"セクション 1"と"セクション 2"のアドレスが重複しました。start オプションのアドレス指定を変更してください。

L2322 (E) Section size too large: "セクション"
セクション"セクション"のサイズが大きすぎます。
$TBR セクションのサイズは 1024 バイト以内でなければなりません。

L2323 (E) Section "セクション 1(アドレス範囲)" overlaps with section "セクション 2(アドレス範囲)" in physical space
物理メモリの配置上で、"セクション 1"と"セクション 2"が重複しています。
各セクションの配置アドレスを見直してください。
<アドレス範囲> ： <セクションの開始アドレス>-<セクションの終端アドレス>

L2330 (E) Relocation size overflow : "ファイル"-"セクション"-"オフセット"
リロケーション演算結果がリロケーションサイズを超えました。分岐先が届かない、特定のアドレスに配置しなければならないシンボルを参照しているなどが考えられます。コンパイル、アセンブルリストで、"セクション"の"オフセット"位置の参照シンボルが正しい位置に配置されているか確認してください。

L2331 (E) Division by zero in relocation value calculation : "ファイル"-"セクション"-"オフセット"
リロケーション演算に 0 除算が発生しました。コンパイル、アセンブルリストで、"セクション"の"オフセット"位置の演算に問題がないか確認してください。

L2332 (E) Relocation value is odd number : "ファイル"-"セクション"-"オフセット"
リロケーション演算結果が奇数になりました。コンパイル、アセンブルリストで、"セクション"の"オフセット"位置の演算に問題がないか確認してください。

L2340 (E) Symbol name "ファイル"-"セクション"-"シンボル..." is too long

"セクション"内の"シンボル"の文字数がアセンブラの翻訳限界を超えました。
シンボルアドレスファイルを出力する場合は、アセンブラの翻訳限界文字数以下になるようなシンボル名としてください。

L2400 (E) Global register in "ファイル" conflicts : "シンボル","レジスタ"

"ファイル"内で指定したグローバルレジスタにはすでに別のシンボルが割り付いています。

L2401 (E) near8,near16 symbol "シンボル" is outside near memory area

"シンボル"は near8、near16 の範囲に割りついていません。start 指定を変更するか、コンパイル時の near 指定を外して、正しいアドレス計算ができるようにしてください。

L2402 (E) Number of register parameter conflicts with that in another file : "関数"

"関数"は複数のファイルで異なるレジスタパラメータ数を指定されています。

L2403 (E) Fast interrupt register in "ファイル" conflicts with that in another file

"ファイル"内で指定した高速割り込み用汎用レジスタ番号が、他ファイルと統一されていません。高速割り込み用汎用レジスタ番号を他ファイルに合わせて、再度コンパイルして下さい。

L2404 (E) Base register "ベースレジスタ種別" in "ファイル" conflicts with that in another file

"ファイル"内で指定した"ベースレジスタ種別"用のレジスタ番号が、他ファイルと統一されていません。
ベースレジスタ番号を他ファイルに合わせて、再度コンパイルして下さい。

L2405 (E) Option "コンパイルオプション" conflicts with that in other files

"コンパイルオプション"の指定が入力ファイル間で統一されていません。
コンパイルオプションを見直してください。

L2410 (E) Address value specified by map file differs from one after linkage as to "シンボル"

"シンボル"のアドレス値がコンパイル時に使用した外部シンボル割り付け情報ファイル内のアドレスとリンク後のアドレスで異なっています。下記の(1)～(3)を確認してください。

(1) コンパイル時のmap オプション指定前後でプログラムを変更している場合は、プログラムの変更をやめてください。

(2) optlnk の最適化によって、コンパイル時のmap オプション指定前後のシンボル並び順が変わることがあります。コンパイル時map オプションを無効にするか、optlnkの最適化オプションを無効にしてください。

(3) tbrオプションまたは#pragma tbr使用時、コンパイラの最適化によって、コンパイル時のmap オプション指定後のシンボルが削除されることがあります。コンパイル時mapオプションを無効にするか、tbrオプションまたは#pragma tbrを無効にしてください。

L2411 (E) Map file in "ファイル" conflicts with that in another file

入力ファイル間でコンパイル時に異なる外部シンボル割り付け情報ファイルを使用しています。

L2412 (E) Cannot open file : "ファイル"

"ファイル"(外部シンボル割り付け情報ファイル)がオープンできません。ファイル名およびアクセス権が正しいか確認してください。

L2413 (E) Cannot close file : "ファイル"

"ファイル"(外部シンボル割り付け情報ファイル)がクローズできません。ディスク容量に空きがない可能性があります。

L2414 (E) Cannot read file : "ファイル"

"ファイル"(外部シンボル割り付け情報ファイル)が読みこめません。ディスク容量に空きがない可能性があります。

L2415 (E) Illegal map file : "ファイル"

"ファイル"(外部シンボル割り付け情報ファイル)のフォーマットが不正です。ファイル名が正しいか確認してください。

L2416 (E) Order of functions specified by map file differs from one after linkage as to "関数名"

関数"関数名"は、コンパイル時に使用した外部シンボル割り付け情報ファイル内の情報とリンク後の配置とで、他の関数との並び順が異なっています。関数内`static`変数のアドレスが、外部シンボル割り付け情報ファイルとリンク後の結果とで異なっている可能性があります。

L2417 (E) Map file is not the newest version: "ファイル名"

`.bls` ファイルが最新バージョンではありません。

L2420 (E) "ファイル 1" overlap address "ファイル 2" : "アドレス"

ファイル 1 とファイル 2 のアドレスが重複しています。

P2500 (E) Cannot find library file : "ファイル"

ライブラリとして指定した"ファイル"がありません。

P2501 (E) "インスタンス" has been referenced as both an explicit specialization and a generated instantiation

すでに定義が存在しているインスタンスに対して、インスタンス生成を要求しています。
"インスタンス"を使用しているファイルに対して、`form=relocate` でリロケータブルファイルを作成していないか確認してください。

P2502 (E) "インスタンス" assigned to "ファイル 1" and "ファイル 2"

"ファイル 1"と"ファイル 2"に"インスタンス"定義が重複しています。
"インスタンス"を使用しているファイルに対して、`form=relocate` でリロケータブルファイルを作成していないか確認してください。

L3000 (F) No input file

入力ファイルがありません。

L3001 (F) No module in library

ライブラリ内のモジュール数が 0 になりました。

L3002 (F) Option "オプション 1" is ineffective without option "オプション 2"

"オプション 1"は"オプション 2"が必要です。

L3004 (F) Unsupported inter-module optimization information type “タイプ” in “ファイル”

ファイル内にサポートしていないモジュール間最適化情報”タイプ”がありました。コンパイラ、アセンブラのバージョンが正しいか確認してください。

P3005 (F) Instantiation loop

インスタンス生成処理がループしています。
入力ファイル名が別ファイルのファイルと一致している可能性があります。拡張子を除いたファイル名が一致しないようにファイル名を変更してください。

P3007 (F) Cannot create instantiation request file "ファイル"

インスタンス生成処理用の中間ファイルを作成できません。
オブジェクト作成フォルダ以下のアクセス権が正しいか確認してください。

P3008 (F) Cannot change to directory "フォルダ"

"フォルダ"に移動できません。"フォルダ"が存在するか確認してください。

P3009 (F) File "ファイル" is read-only

"ファイル"は読み取り専用です。アクセス権を変更してください。

L3100 (F) Section address overflow out of range : "セクション"

"セクション"のアドレスが使用可能な上限の領域を超えました。
`start` オプションのアドレス指定を変更してください。
アドレス空間の詳細については各マイコンのハードウェアマニュアルを参照してください。

L3102 (F) Section contents overlap in absolute section "セクション"

絶対アドレスセクションのセクション内データアドレスが重複しています。ソースプログラムを修正してください。

L3110 (F) Illegal cpu type "マイコン種別" in "ファイル"

異なるマイコン種別のファイルを入力しました。

L3111 (F) Illegal encode type "エンディアン種別" in "ファイル"

異なるエンディアン種別のファイルを入力しました。

L3112 (F) Invalid relocation type in "ファイル"

"ファイル"内にサポートしていないリロケーションタイプがありました。コンパイラ、アセンブラのバージョンが正しいか確認してください。

L3120 (F) Illegal size of the absolute code section : "セクション" in "ファイル"

"ファイル"に存在する絶対アドレスコードセクション"セクション" のサイズが不正です。CPU 種別が RX ファミリでビッグエンディアンの場合は、絶対アドレスコードセクションのサイズが 4 の倍数になるように変更してください。

L3200 (F) Too many sections
セクション数が翻訳限界を超えました。複数ファイル出力を指定すると解決できる可能性があります。

L3201 (F) Too many symbols
シンボル数が翻訳限界を超えました。複数ファイル出力を指定すると解決できる可能性があります。

L3202 (F) Too many modules
モジュール数が翻訳限界を超えました。ライブラリを分けて作成してください。

L3203 (F) Reserved module name “optlnk_generates”
`optlnk_generates_** (**`は、`01`～`99` までの数値`)`は、最適化リンケージエディタで使用する予約名称です。`.obj/.rel` ファイル名およびライブラリ内モジュール名として使用しています。ファイル名およびライブラリ内モジュール名で使用している場合は、変更してください。

L3300* (F) Cannot open file : "ファイル"
"ファイル"をオープンできません。ファイル名およびアクセス権が正しいか、確認してください。
`*  P3300` と表示される場合があります。

L3301 (F) Cannot close file : "ファイル"
"ファイル"をクローズできません。ディスク容量に空きがない可能性があります。

L3302 (F) Cannot write file : "ファイル"
"ファイル"に書き込めません。ディスク容量に空きがない可能性があります。

L3303* (F) Cannot read file : "ファイル"
"ファイル"を読めません。空ファイルを入力したか、ディスク容量に空きがない可能性があります。
`*  P3303` と表示される場合があります。

L3310* (F) Cannot open temporary file
中間ファイルをオープンできません。`HLNK_TMP` 指定が正しいか確認してください。
またはディスク容量に空きがない可能性があります。
`*  P3310` と表示される場合があります。

L3311 (F) Cannot close temporary file
中間ファイルをクローズできません。ディスク容量に空きがない可能性があります。

L3312 (F) Cannot write temporary file
中間ファイルに書き込めません。ディスク容量に空きがない可能性があります。

L3313 (F) Cannot read temporary file
中間ファイルを読めません。ディスク容量に空きがない可能性があります。

L3314 (F) Cannot delete temporary file
中間ファイルを削除できません。ディスク容量に空きがない可能性があります。

L3320* (F) Memory overflow

最適化リンケージエディタが内部で使用するメモリが不足しています。メモリを増やしてください。

* P3320 と表示される場合があります。

L3400 (F) Cannot execute "ロードモジュール"

"ロードモジュール"を起動できません。"ロードモジュール"のパスが設定されているか確認してください。

L3410 (F) Interrupt by user

標準入力端末から「(Ctrl)+C」キーによる割り込みを検出しました。

L3420 (F) Error occurred in "ロードモジュール"

"ロードモジュール"実行中にエラーが発生しました。

P3500 (F) Bad instantiation request file -- instantiation assigned to more than one file

インスタンス生成処理用の中間ファイルに誤りがあります。

リンク対象ファイルを再コンパイルしてください。

P3505 (F) corrupted template information file or instantiation request file

テンプレート処理用中間ファイル、またはインスタンス生成処理用の中間ファイルのデータが誤っています。

これらのファイルの編集はしないでください。

L4000* (-) Internal error : ("内部エラー番号") "ファイル 行番号" / "コメント"

最適化リンケージエディタの処理中に内部的な問題が発生しました。

メッセージ内の内部エラー番号、ファイル、行番号、コメントを添えて、販売元のサポートセンターまでご連絡ください。

* P4000 と表示される場合があります。

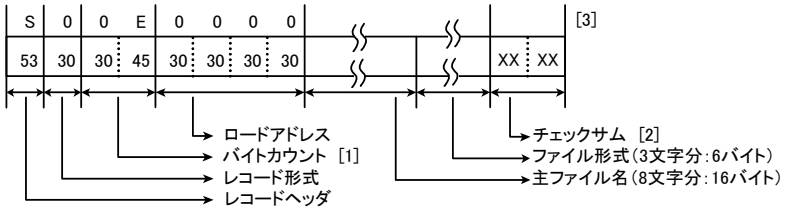
13. 付録

13.1 モトローラS形式、インテルHEX形式ファイル

本節では、最適化リンケージエディタによって出力されるモトローラ S 形式ファイルおよび、インテル HEX 形式ファイルについて説明します。

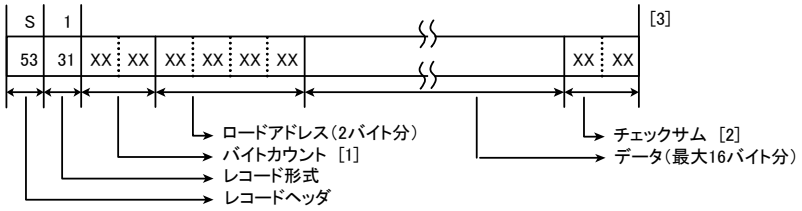
13.1.1 モトローラS形式ファイル

(a) ヘッダレコード(S0レコード)

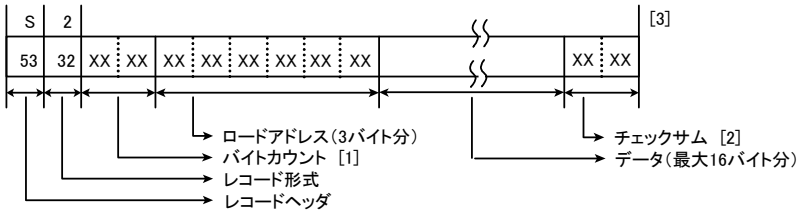


(b) データレコード(S1, S2, S3レコード)

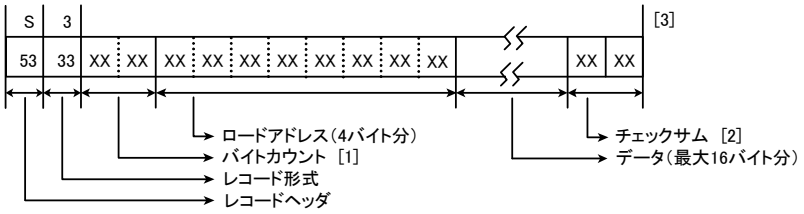
(i) ロードアドレスが0～FFFFの場合



(ii) ロードアドレスが10000～FFFFFFの場合

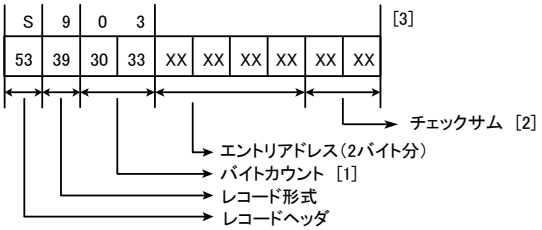


(iii) ロードアドレスが1000000～FFFFFFFFの場合

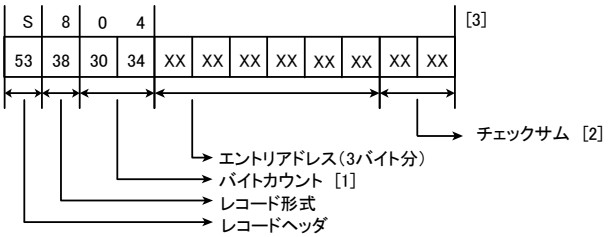


(c) エンドレコード(S9, S8, S7レコード)

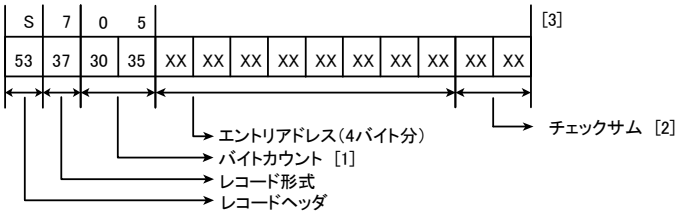
(i) エントリアドレスが0～FFFFの場合



(ii) エントリアドレスが10000～FFFFFFの場合



(iii) エントリアドレスが1000000～FFFFFFFFの場合



【注】 [1] ロードアドレス(またはエントリアドレス)からチェックサムまでのバイト数
[2] バイトカウンタからチェックサムの前までのデータ値をバイト単位に加算した結果の1の補数
[3] チェックサムの直後に改行コードが付加される

13.1.2 インテルHEX形式ファイル

各データレコードの実行アドレスは以下のように求めます。

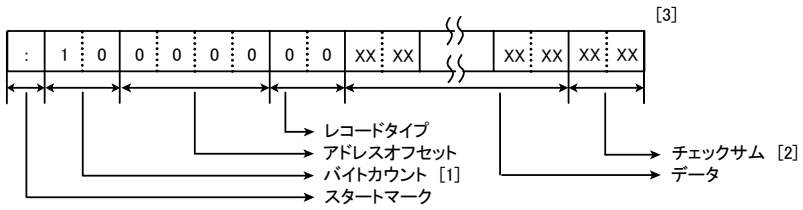
(1) セグメントアドレスの場合

(セグメントベースアドレス << 4) + (データレコードのアドレスオフセット)

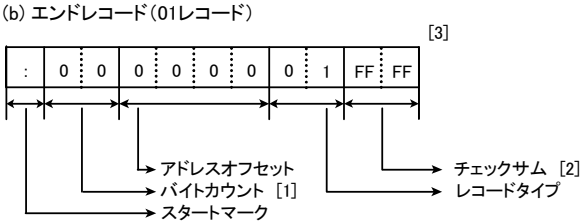
(2) リニアアドレスの場合

(リニアベースアドレス << 16) + (データレコードのアドレスオフセット)

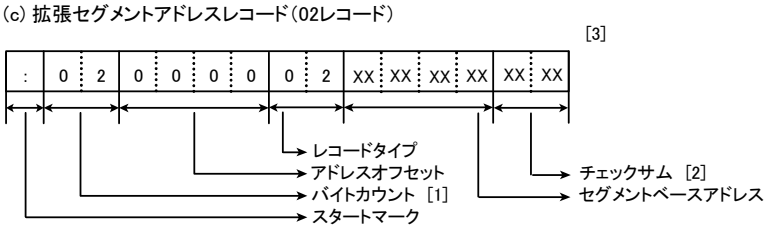
(a) データレコード(00レコード)



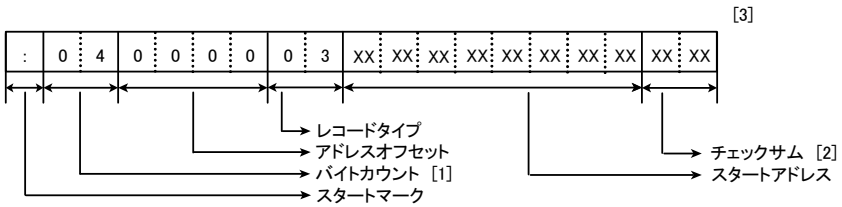
(b) エンドレコード(01レコード)



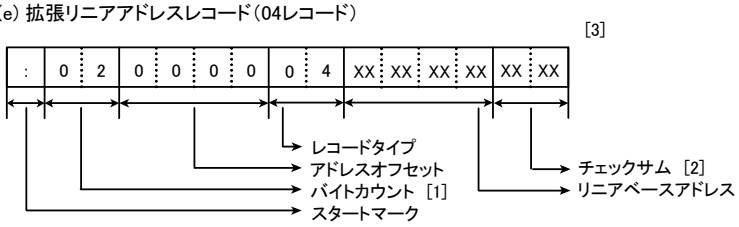
(c) 拡張セグメントアドレスレコード(02レコード)



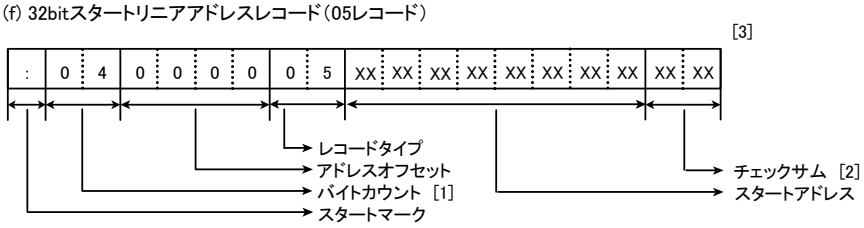
(d) スタートアドレスレコード(03レコード)



(e) 拡張リニアアドレスレコード(04レコード)



(f) 32bitスタートリニアアドレスレコード(05レコード)



【注】 [1] レコードタイプの次のデータから、チェックサムまでのバイト数
[2] バイトカウンタからチェックサムの前までのデータを、16進数で加算した結果の2の補数(下位8bitが有効)
[3] チェックサムの直後に改行コードが付加される

13.2 ASCIIコードー覧表

表 13.1 ASCII コードー覧表

下位 4 ビット	上位 4 ビット							
	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

ルネサスマイクロコンピュータ開発環境システム
ユーザーズマニュアル
M16Cシリーズ,R8Cファミリ用C/C++コンパイラパッケージ V.6.00
アセンブラ、最適化リンケージエディタ ユーザーズマニュアル

発行年月日　2011年01月16日　Rev.1.00

発行　ルネサス エレクトロニクス株式会社
〒211-8668 神奈川県川崎市中原区下沼部1753

編集　株式会社ルネサス ソリューションズ

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル）

(03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

Colophon 1.0

M16Cシリーズ、R8Cファミリ用
C/C++コンパイラパッケージ V.6.00
アセンブラ、最適化リンケージエディタ
ユーザーズマニュアル

ルネサス エレクトロニクス株式会社

RJJ10J2764-0100