User's Manual

M16C シリーズ、R8C ファミリ用
C コンパイラパッケージ V.5.45
アセンブラ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.2.00　2010.04

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。

AS30 ユーザーズマニュアル目次

マニュアルについて

マニュアルの記述規則

ユーザーズマニュアルの記述規則および記述用語の説明を示します。

英大文字(A～Z)

英大文字(A～Z)で記された文字列は、命令語などの固有の文字列であることを示します。ただし、実行プログラム名および、ファイルの拡張子は英小文字で記述しています。

英小文字(a～z)

英小文字(a～z)で記された文字列は、任意の文字列であることを示します。ユーザが任意に記述できるラベル名などを示します。ただし、実行プログラム名および、ファイルの拡張子は英小文字で記述しています。

¥

ディレクトリの区切りを示します。マニュアル上では、特に断らない限りコマンド入力例として MS-DOS 上の表記を用いています。

[ ]

括弧内の記述は省略できることを示します。また、括弧内に'|'で区切られた複数の記述がある場合は、括弧内の内容の 0 個以上を選択して記述することを示します。

...

直前に記述している内容を繰り返し指定できることを示します。

XX

プログラムが出力するメッセージのうち、該当するファイル名やコマンドオプションなどに置き換えて出力される文字列であることを示します。

>

コマンドプロンプトを示します。

マニュアルの記述用語

ユーザーズマニュアルの記述用語について次に示します。

AS30

M16C ファミリ用アセンブラ製品を示します。また、AS30 に含まれるプログラム全てを総称します。

as30, mac30, pre30, asp30, ln30, lmc30, lb30, xrf30, abs30

AS30 製品に含まれる実行プログラム名です。また、各プログラムを起動する起動コマンドを示します。

AS30.EXE, PRE30.EXE, ASP30.EXE, LN30.EXE, LMC30.EXE, LB30.EXE, XRF30.EXE, ABS30.EXE

MS-DOS 上の実行プログラム名を示します。

ニーモニック

M16C ファミリ用のアセンブリ言語命令を示します。

構造化記述命令

本アセンブラ製品で処理できる構造化記述命令を示します。アセンブラ製品によって記述方法が異なります。

指示命令

アセンブラを制御するための命令です。アセンブラ製品によって命令が若干異なります。

命令

ニーモニック、構造化記述命令及び指示命令の総称です。

アセンブリプログラム

本アセンブラ製品で処理可能なニーモニック、構造化記述命令及び指示命令で記述されたプログラムを示します。

ソースファイル

アセンブリプログラムを記述したファイルを示します。

AS30 の仕様

AS30 は以下に示す仕様に基づいて設計しています。この仕様の範囲内でご使用願います。

制限事項

項目	仕様
同時にオープンするファイル数	最大 9 ファイル
環境変数設定文字数	2048 バイト(文字)
ソースファイルでの 1 行の文字数	512 バイト(文字)
マクロ定義の数	65535

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。

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

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

数字

0 1 2 3 4 5 6 7 8 9

特殊文字

" # $ % & ' ( ) * + , - . / : ; [ ] ¥ ^ _ | ~

空白文字

（スペース）（タブ）

改行文字

（リターン）（ラインフィード）

注意事項

漢字などの多バイト文字は使用できません。

製品の概要

AS30 は、M16C ファミリシングルチップマイクロコンピュータ制御プログラムの開発を、アセンブリ言語レベルで支援するソフトウェアシステムです。

アセンブリ言語で記述したソースプログラムをソースレベルデバッグが可能なフォーマットのファイルに変換します。また、M16C ファミリの ROM に書き込み可能なフォーマットのファイルへ変換するプログラムも付属しています。さらに、C コンパイラと組み合わせることにより、C 言語によるプログラム開発が行えます。

製品の構成

AS30 は、次に示す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、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。アセンブラドライバは、複数のアセンブリソースファイルを処理することができます。

●マクロプロセッサ

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ソース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ソースファイルは、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。

●プリプロセッサ

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ソースファイルを生成します。プリプロセッサが生成したアセンブリソースファイルは、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。プリプロセッサを起動するには、as30 のコマンド行でコマンドオプションを指定してください。

●アセンブラプロセッサ

マクロプロセッサ、プリプロセッサが前処理を行ったアセンブリソースファイルをリロケータブルモジュールファイルに変換します。

●リンケージエディタ(ln30)

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

●ライブラリアン(lb30)

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

●ロードモジュールコンバータ(lmc30)

リンケージエディタの生成したアブソリュートモジュールファイルを ROM に書き込み可能な機械語ファイルに変換します。

●クロスリファレンサ(xrf30)

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

●アブソリュートリスタ(abs30)

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

機能概要

●リロケータブルプログラミング機能

プログラムを複数のファイルに分割して記述することができます。分割して記述したプログラムはファイル毎にアセンブルできます。単一のファイルに絶対アドレスを割り付ければ、単体でデバッグができます。また、複数のソースプログラムファイルを 1 つのデバッグファイルに結合できます。

●最適化コード生成機能

AS30 は、ソースプログラムに対して、最もコード生成効率のよい、アドレッシングモード及び分岐命令を選択する機能を持っています。

●マクロ機能

AS30 は、プログラムの可読性を向上するためのマクロ機能を持っています。

●高級言語のソースレベルデバッグ情報の出力

M16C ファミリ対応の高級言語で開発したプログラムに対して、ソースレベルでのデバッグを可能にする、高級言語デバッグ情報を出力します。

●ファイルの生成

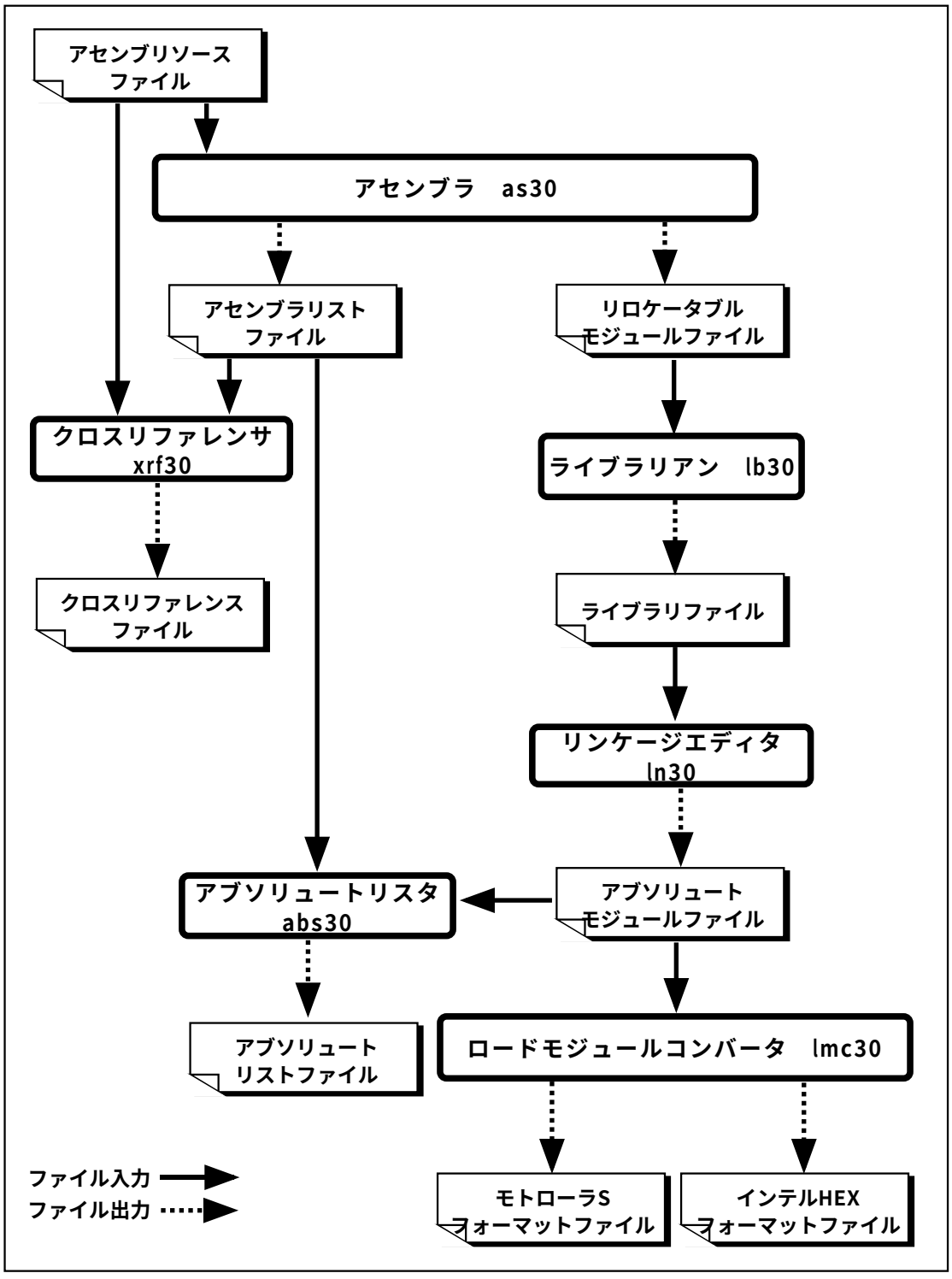
AS30 の各プログラムは、リロケータブルモジュールファイル、アブソリュートモジュールファイル、エラータグファイル及びリストファイルなどのファイルを生成します。

●IEEE-695 フォーマットのファイル生成機能

AS30 が生成するバイナリ形式のファイルは IEEE-695 フォーマットで出力します。IEEE-695 フォーマットに準拠したフォーマットを採用している、他の M16C ファミリ用開発ツールとの共用が可能です。

IEEE(Institute of Electrical and Electronics Engineers:アメリカ電気電子技術者協会)

AS30 の処理概要


アセンブリソース
ファイル
アセンブラ　as30
アセンブラリスト
ファイル
リロケータブル
モジュールファイル
クロスリファレンサ
xrf30
ライブラリアン　lb30
クロスリファレンス
ファイル
ライブラリファイル
リンケージエディタ
ln30
アブソリュートリスタ
abs30
アブソリュート
モジュールファイル
アブソリュート
リストファイル
ロードモジュールコンバータ　lmc30
ファイル入力
ファイル出力
モトローラS
フォーマットファイル
インテルHEX
フォーマットファイル

as30 の構成

as30 は、以下に説明する 3 つのプログラムを制御するプログラムです。

注意事項
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直接起動しないでください。

マクロプロセッサ機能概要

- ソースファイルに記述されているマクロ指示命令を処理します。
- 処理されたファイルは、構造化プリプロセッサ又はマクロプロセッサで処理可能なファイル形式です。

構造化プリプロセッサ機能概要

- ソースファイルに記述されている構造化指示命令を処理します。
- 処理されたファイルは、アセンブラプロセッサで処理可能なファイル形式です。

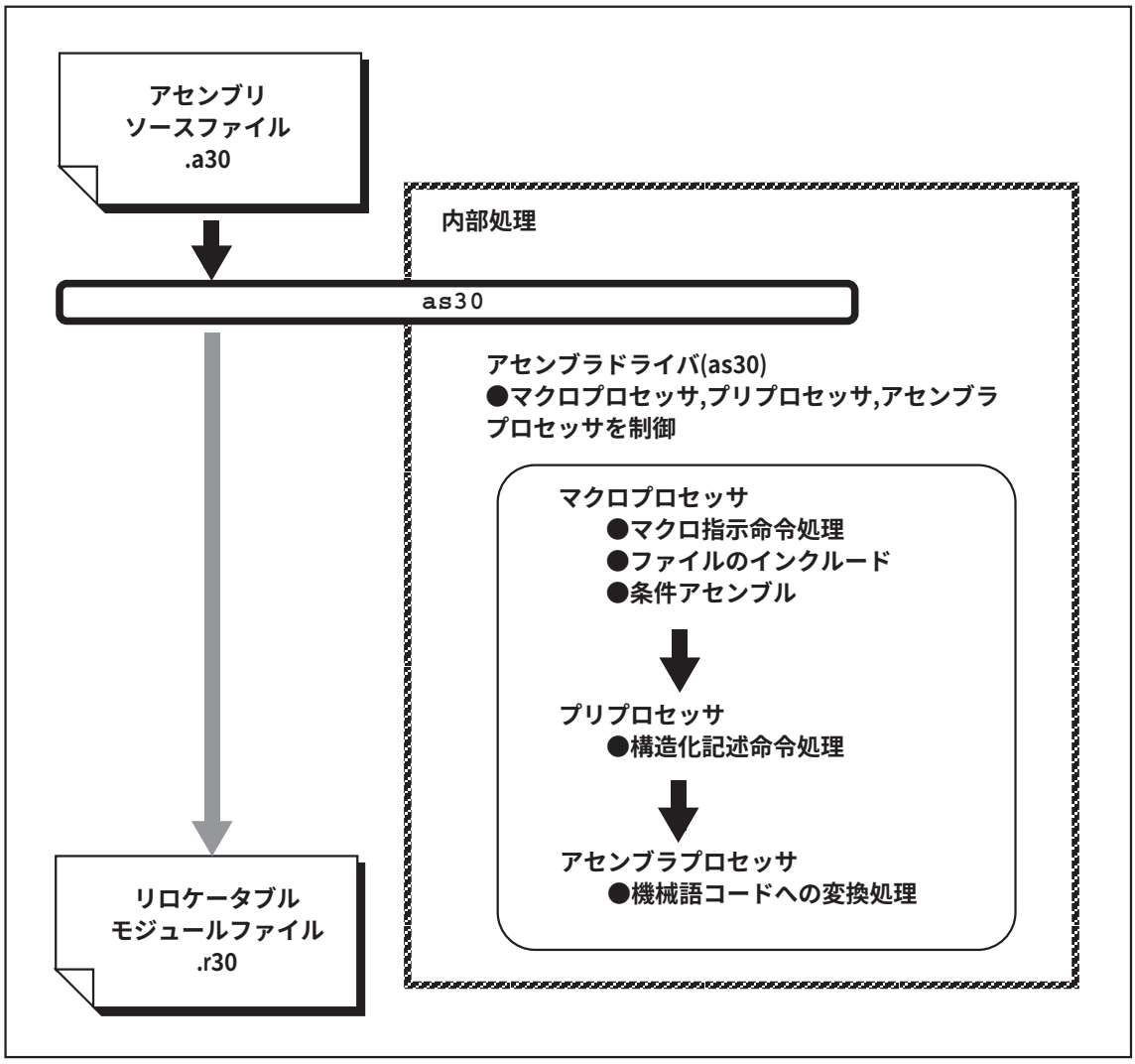
アセンブラプロセッサ機能概要

- ソースファイルに記述されているアセンブリ言語、マクロプロセッサ及び構造化プリプロセッサに処理された結果のアセンブリ言語をリロケータブルモジュールファイルに変換します。

as30 の処理概要

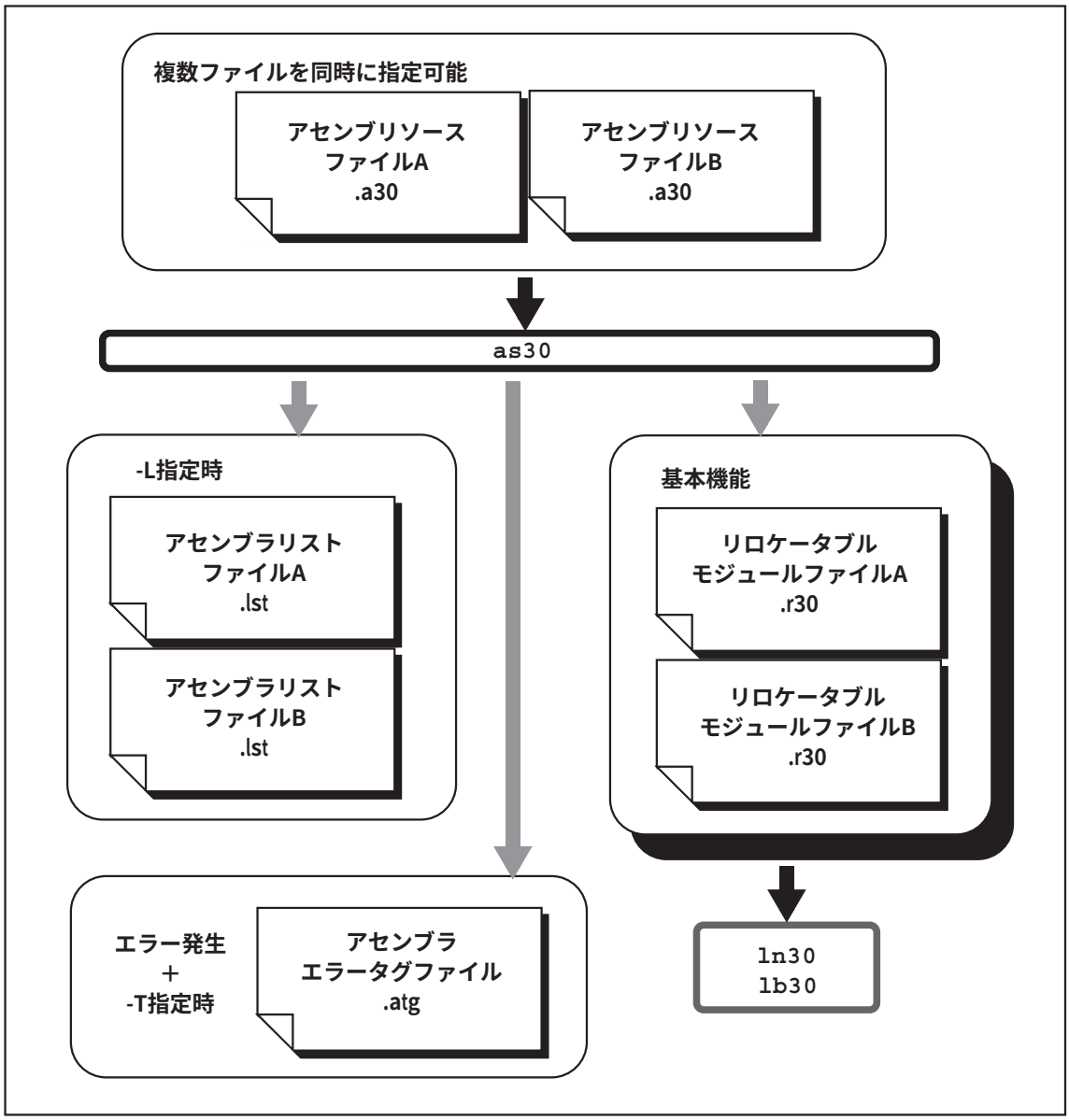
as30 は、次の順序で各プログラムを起動します。

1　マクロプロセッサ
2　構造化プリプロセッサ（コマンドオプション'-P'が指定されたとき）
3　アセンブラプロセッサ



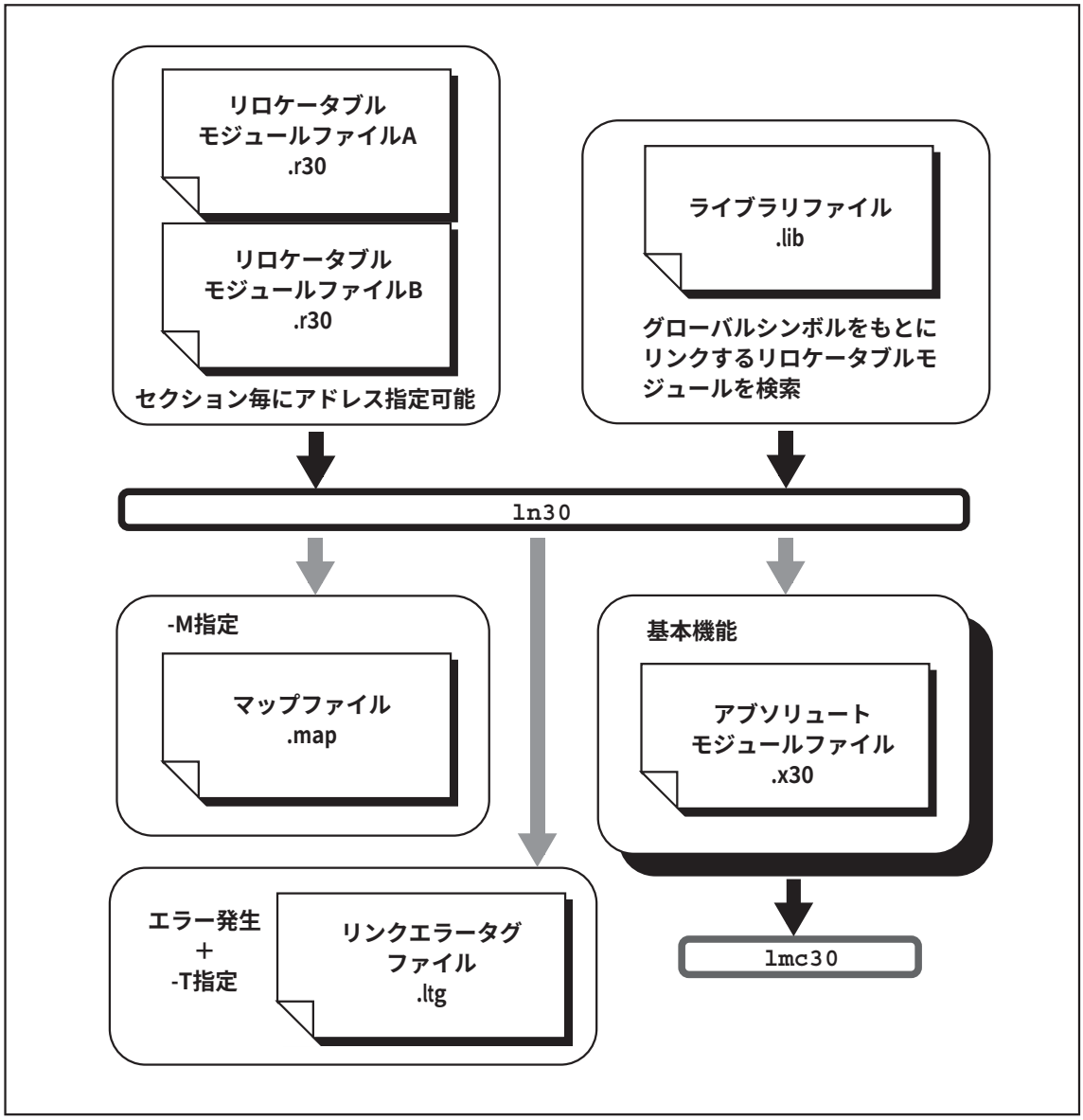
as30 の機能概要

- リロケータブルモジュールファイルを生成します。
- アセンブラリストファイルを生成します。



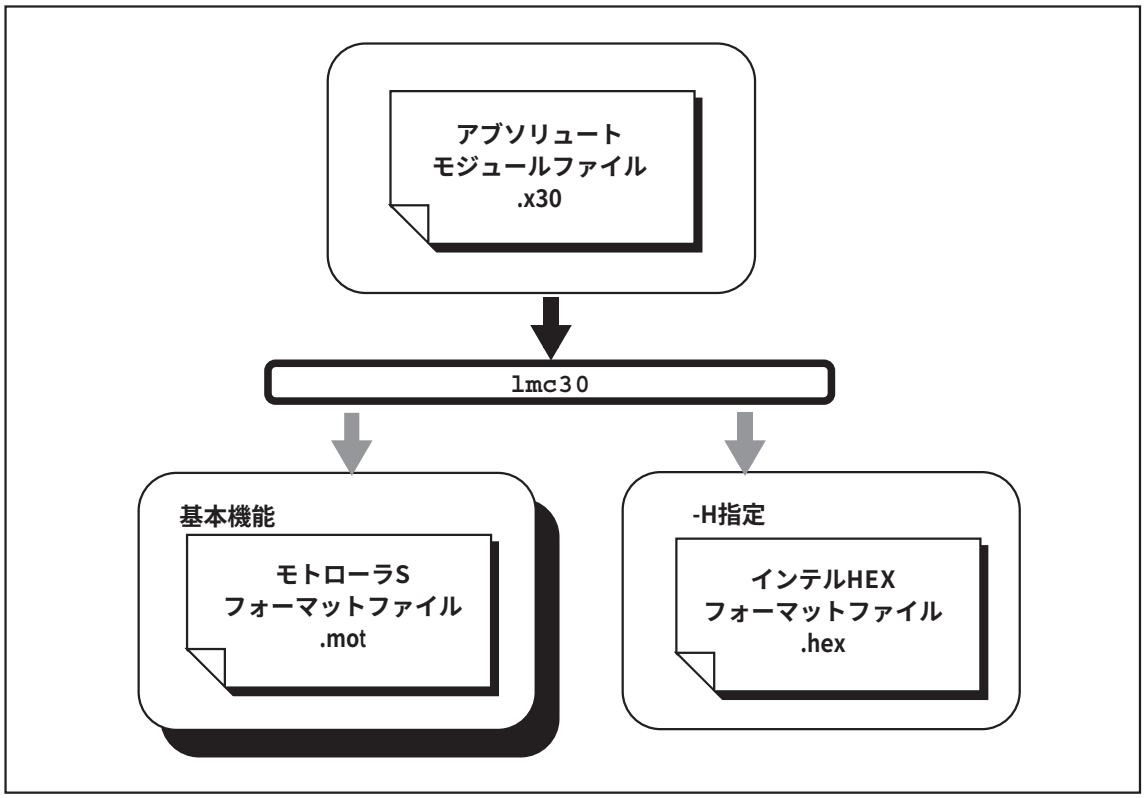
ln30 機能概要

- アブソリュートモジュールファイルを生成します。
- アドレス配置を示すマップファイルを生成します。
- セクションを任意のアドレスに配置します。
- ライブラリファイルに登録されている任意のリロケータブルモジュールを利用できます。



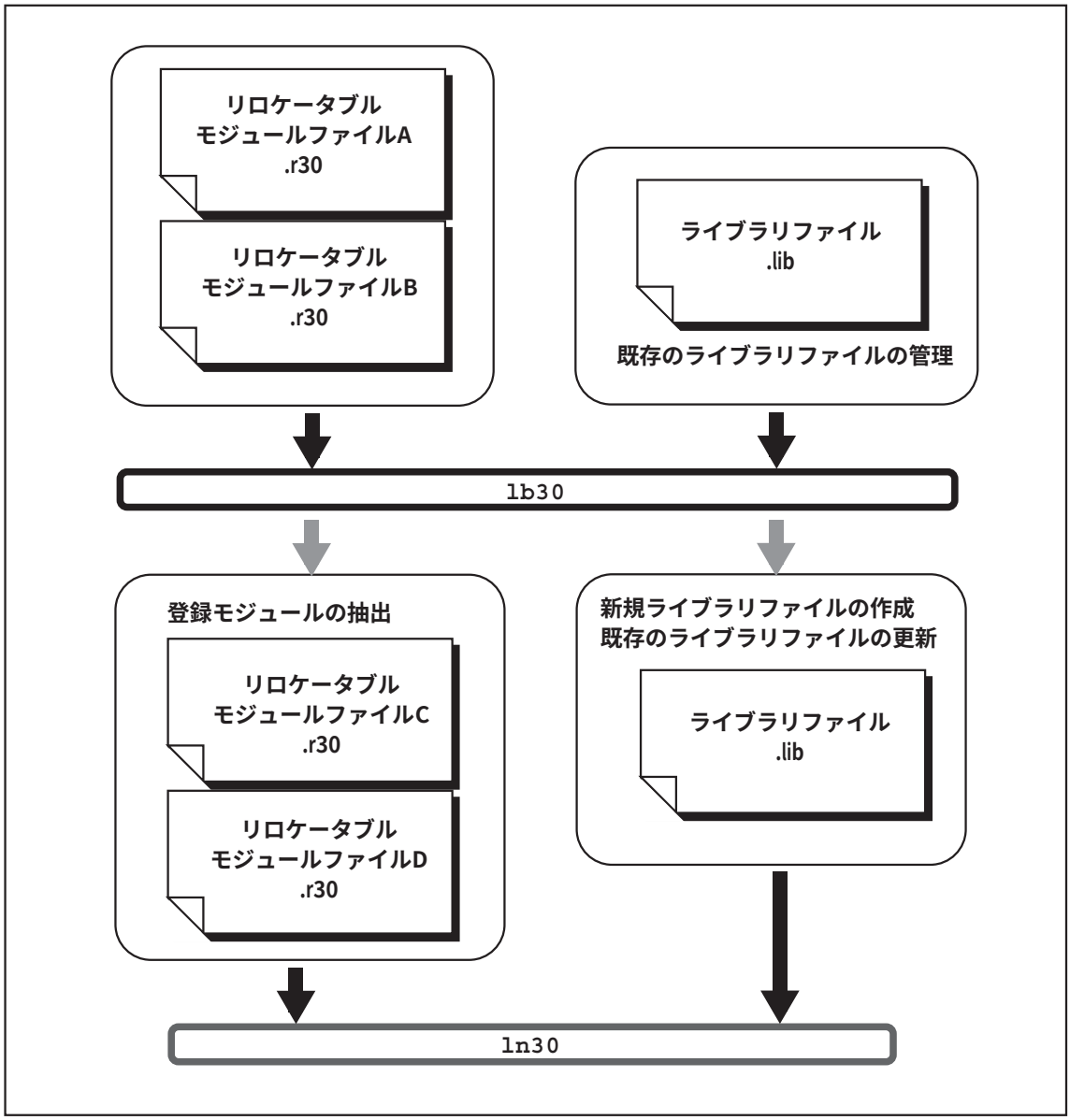
lmc30 の機能概要

- PROM に書き込み可能なモトローラ S フォーマットファイルを生成します。
- PROM に書き込み可能なインテル HEX フォーマットファイルを生成します。



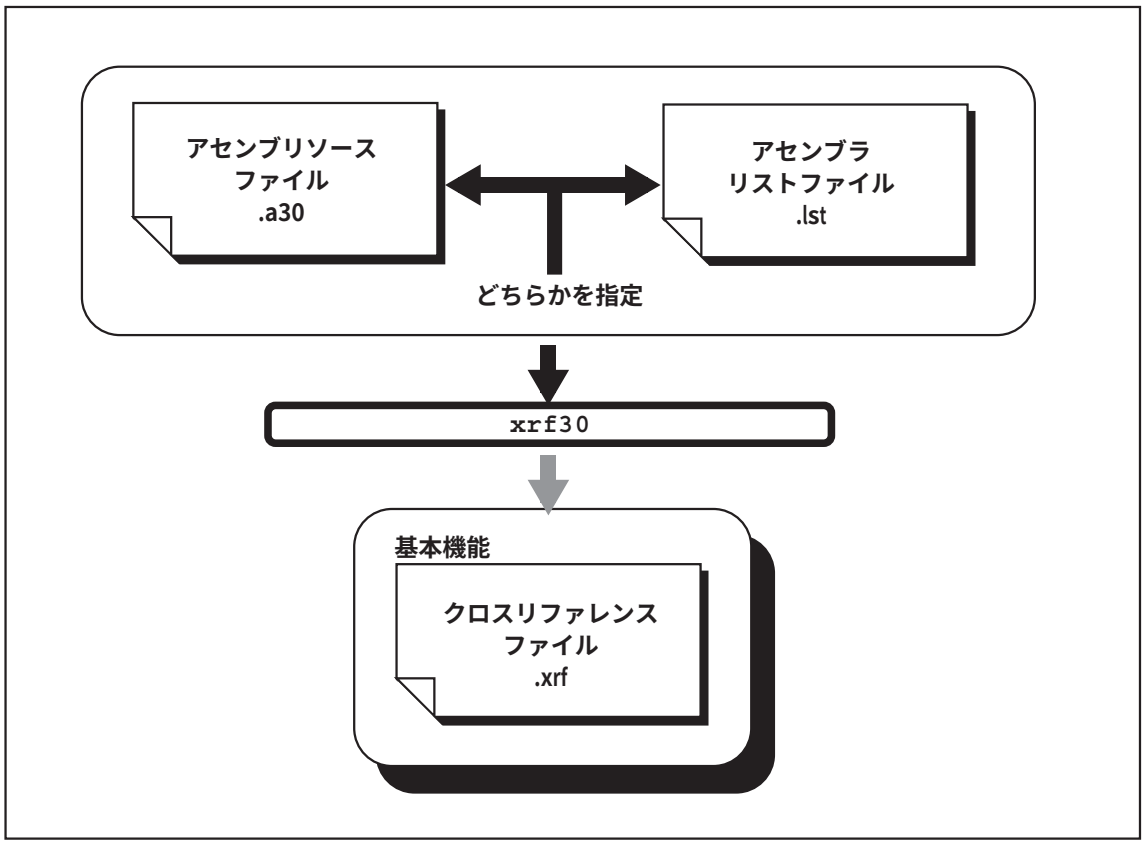
lb30 の機能概要

- ライブラリファイルを新しく生成します。
- ライブラリファイルに登録されているリロケータブルモジュールを更新したり削除したりします。
- ライブラリファイルに登録されているリロケータブルモジュール情報を一覧にしたリストファイルを生成します。



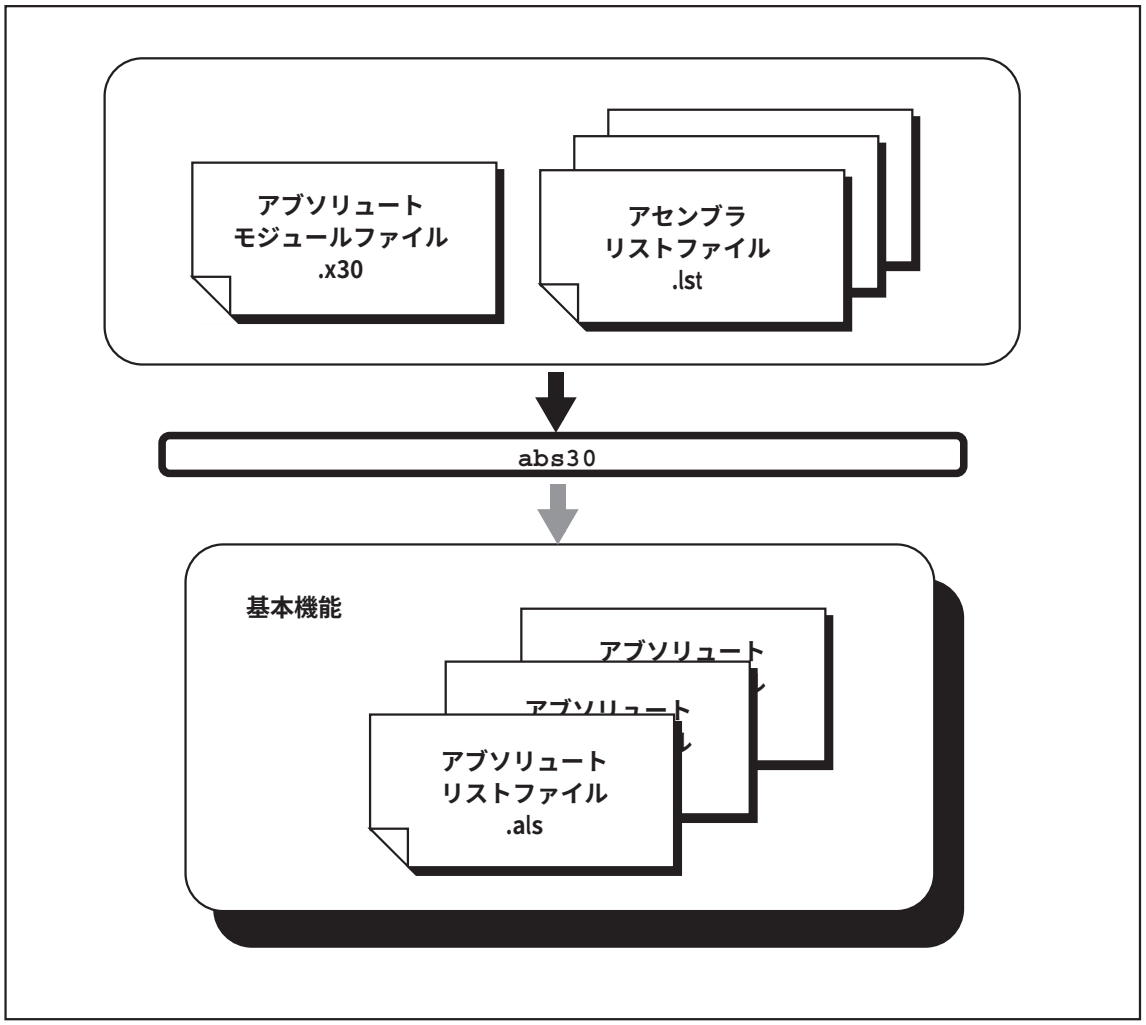
xrf30 の機能概要

- アプリケーションプログラムの中に記述されているラベルとシンボルを一覧できるクロスリファレンスファイルを生成します。
- システムラベル情報の出力を制御できます。



abs30 の機能概要

・　リンク結果のアブソリュートアドレスをアセンブラリストファイルに出力します。



AS30 の機能

AS30 は M16C ファミリマイコンの制御プログラム開発をアセンブリ言語レベルで支援するソフトウェアシステムです。

AS30 は次の機能をサポートしています。

- リロケータブルプログラミング
- ライブラリファイルの参照
- インクルードファイルの参照
- コードの最適選択
- SB レジスタオフセットアドレス指定
- スペシャルページベクタテーブル参照
- マクロ機能
- 条件アセンブル機能
- 構造化記述機能
- 環境変数の参照
- メッセージの出力

注意事項
ハードウエア条件については実際に使用するシステムを考慮して、ソース記述やリンク処理を行ってください。ハードウエアの条件とは、●RAM サイズとそのアドレス範囲及び●ROM サイズとそのアドレス範囲を示します。

リロケータブルプログラミング

as30 は、プログラムを複数のファイルに分割して開発するためのリロケータブルプログラミングをサポートしています。

複数のソースファイルを ln30 が一つの機械語ファイルに連結します。複数のファイルを一つにまとめる際に、「セクション」単位でプログラムの再配置が可能です。

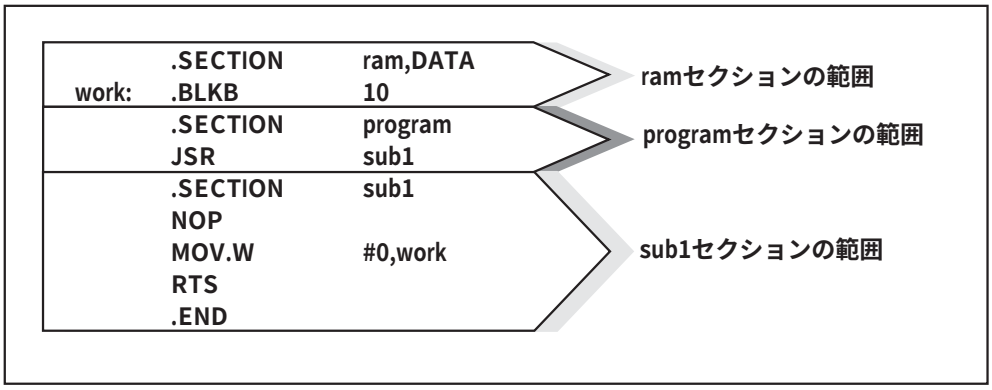
また、ファイル間での同一ラベルの参照が可能です。

セクション

AS30 は、セクション単位でアドレスを管理します。

セクションの区切りは、次のように決められます。

- 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



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を単位として出力します。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」のようにタイプと属性の組み合わせによって決まります。

1 絶対属性セクション（属性）
2 相対属性セクション（属性）
3 CODE タイプセクション（タイプ）
4 DATA タイプセクション（タイプ）
5 ROMDATA タイプセクション（タイプ）

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
```

相対属性セクション

- 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。

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 タイプのセクションは、アブソリュートモジュールにおいて、ROM 領域に配置されるように指定してください。

例）

```
.SECTION    program,CODE
```

DATA 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 タイプのセクションは、アブソリュートモジュールにおいて、RAM 領域に配置されるように指定してください。

例）

```
.SECTION    mem,DATA
```

ROMDATA 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 タイプのセクションは、アブソリュートモジュールにおいて、ROM 領域に配置されるように指定してください。

例）

```
.SECTION    const,ROMDATA
```

セクション配置

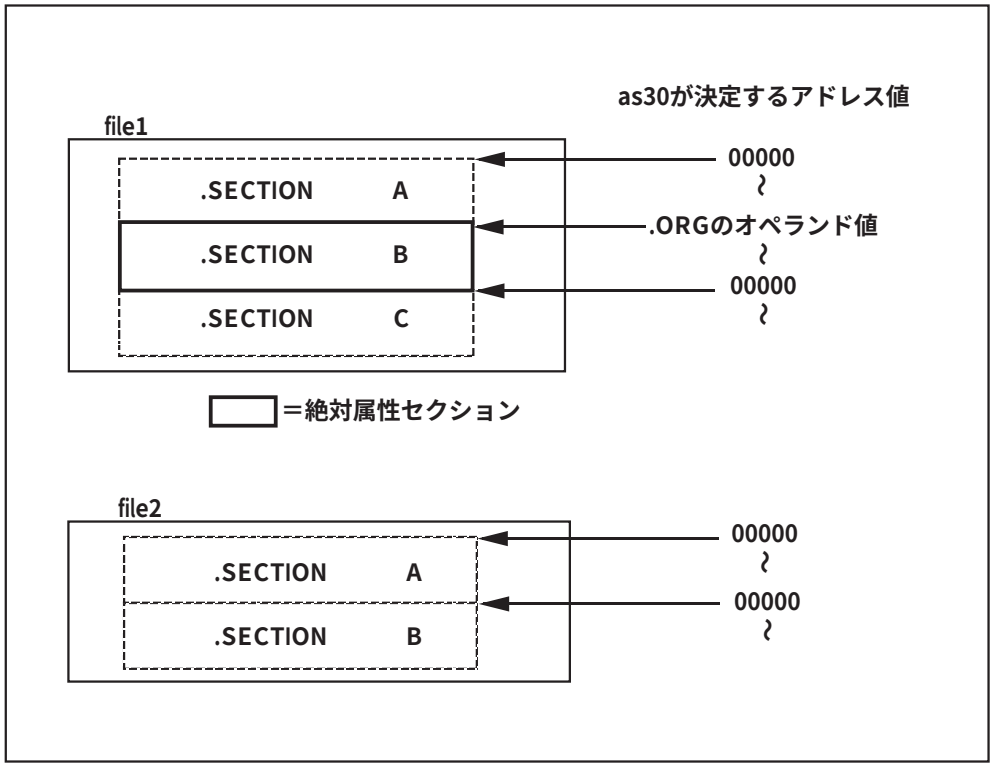
as30 アセンブラは、ソースプログラムに記述されているセクションにアドレス指定がされていなければ、常にアドレス 0 からセクション内のアドレスを決定します。ln30 リンケージエディタは、すべてのセクションのアドレスが重なることがないように配置します。

注意事項
as30 及び ln30 は、実際の M16C ファミリの機種毎の ROM 及び RAM の物理的なアドレス配置を関知しません。したがって、リンクの結果によって、DATA タイプのセクションがチップの ROM 領域に配置されてしまうこともあります。リンクを実行する場合は、実際のチップのアドレスを確認して、セクションを配置するようにしてください。

as30 のセクション管理

アセンブラは、セクションの定義にアドレス指定がされている場合を除いて、リロケータブルアドレス値（仮のアドレス値）を出力します。

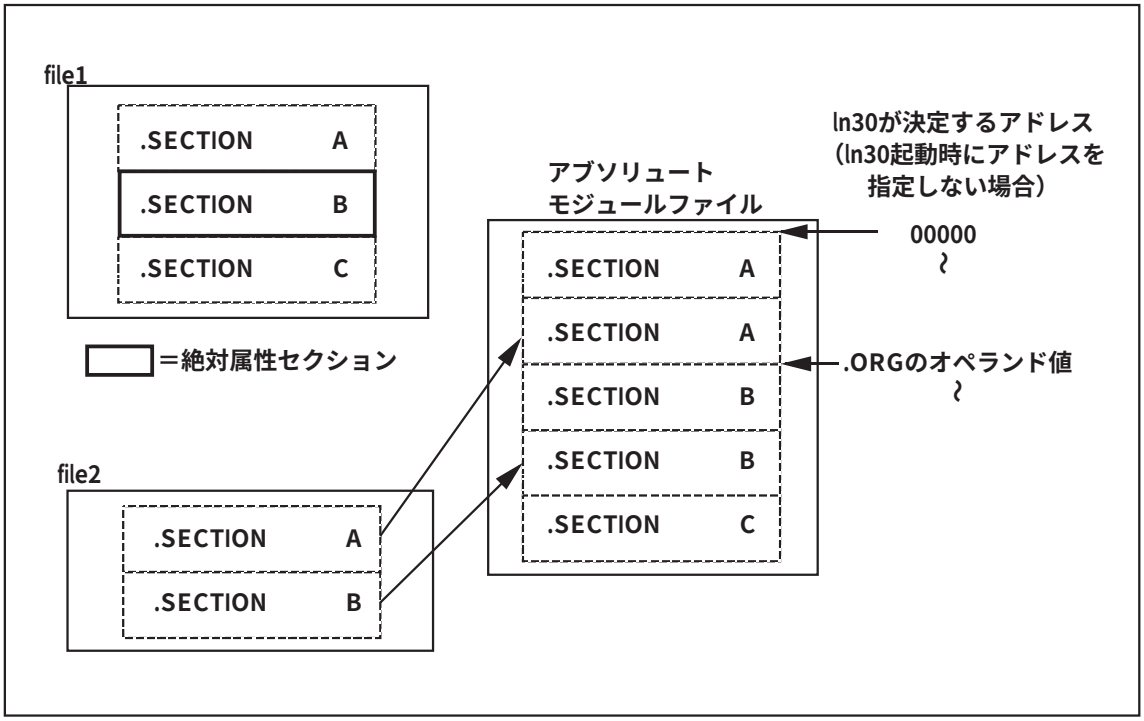
- 絶対属性セクションは、指定されているアドレスから順に絶対アドレスが決定されます。
- 相対属性セクションは、セクション毎に 0 から順にアドレス（リロケータブル）が決定されます。相対属性のセクションの開始アドレスはすべて 0 です。



ln30 のセクション管理

ln30 リンケージエディタは、次の規則に従ってセクションを配置し絶対アドレスを決定していきます。

- 同一名のセクションを一箇所にまとめます。複数ファイルに同一名のセクションがある場合もすべて一箇所にまとめられます。
- セクションの開始アドレスは、ln30 のコマンドオプション(-order)の指定に従って決定されます。
- セクションの開始アドレスは、指定がなければ 0 から順に決定します。
- 異なる名前のセクションの順序は ln30 のコマンドオプション(-order)の指定に従って決定されます。
- 異なる名前のセクションの順序は ln30 に読み込まれた順に配置されます。
- 絶対属性の後に、絶対属性を配置しようとした場合は、ln30 はワーニングを出力します。
- セクション名が同一で、セクションの属性又はタイプの情報に矛盾がある場合は、ln30 はワーニングを出力します。
- セクションタイプが"DATA"で、複数のセクションでアドレスがオーバーラップする場合は、ワーニングを出力します。セクションはオーバーラップして配置されます。
- セクションタイプが"CODE"又は"ROMDATA"で、複数のセクションでアドレスがオーバーラップする場合は、エラーとなります。
- 同一名のセクションで、相対属性の後に絶対属性を配置しようとした場合は、エラーとなります。



先頭アドレスの偶数番地指定

相対属性のセクションに限り、リンク時に決定するセクションのスタートアドレスが必ず偶数番地になるように設定することができます。

指示命令".SECTION"のオペランドに"ALIGN"を指定するとそのセクションは必ず偶数番地からスタートします。

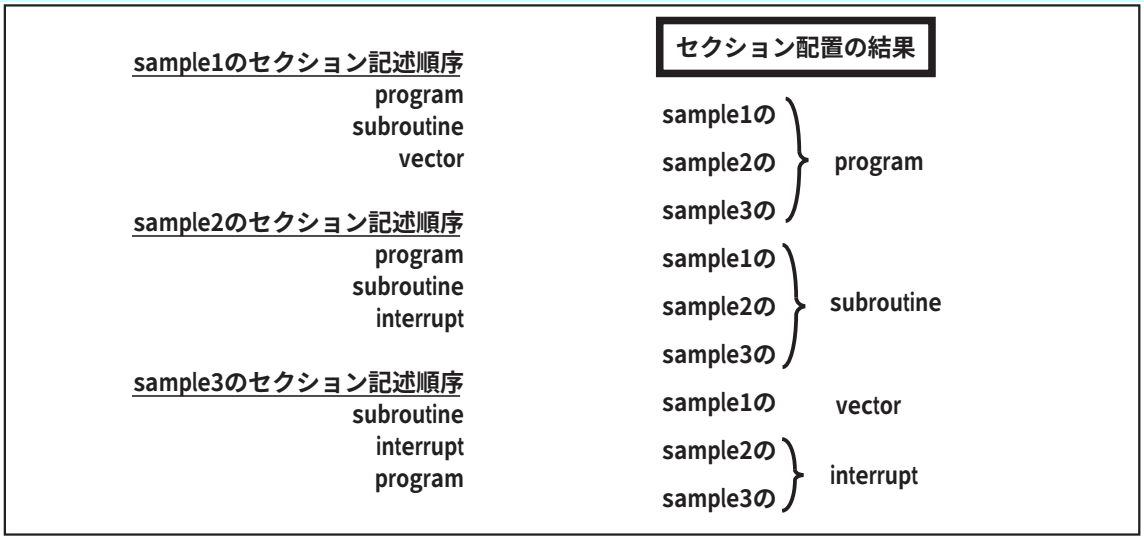
例）

```
.SECTIONprogram,CODE,ALIGN
```

ln30 のセクション配置例）

３つのリロケータブルモジュールファイルを次のコマンド入力でリンクする場合。この場合、生成されるアブソリュートモジュールファイル名は"sample1.x30"となります。

```
>ln30 sample1 sample2 sample3
```



外部参照

リロケータブルアセンブル機能の一部として、ラベル、シンボルおよびビットシンボルの外部参照が可能です。ソースプログラムを 2 つ以上のファイルに分割して記述する場合に、自分自身のファイル内では定義されていないラベルなどをプログラムに記述することを外部参照といいます。

- ラベル
  メモリのアドレスにたいして定義されている名前を示します。
- シンボル
  任意の値にたいして定義されている名前を示します。
- ビットシンボル
  メモリのビット位置を示す値に定義されている名前を示します。

ラベルとシンボルの属性

ラベルとシンボル（ビットシンボルも含む）は、定義のされかたによって 4 つの属性にわかれます。ローカルな属性のものは外部参照はできません。

「グローバルなリロケータブルラベル」のように組み合わせて区別されます。

1 グローバル
2 ローカル
3 リロケータブル
4 アブソリュート

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リロケータブルファイルに出力されます。

ローカル

- 指示命令".GLB"又は".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リロケータブル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リロケータブルファイルに出力されます。

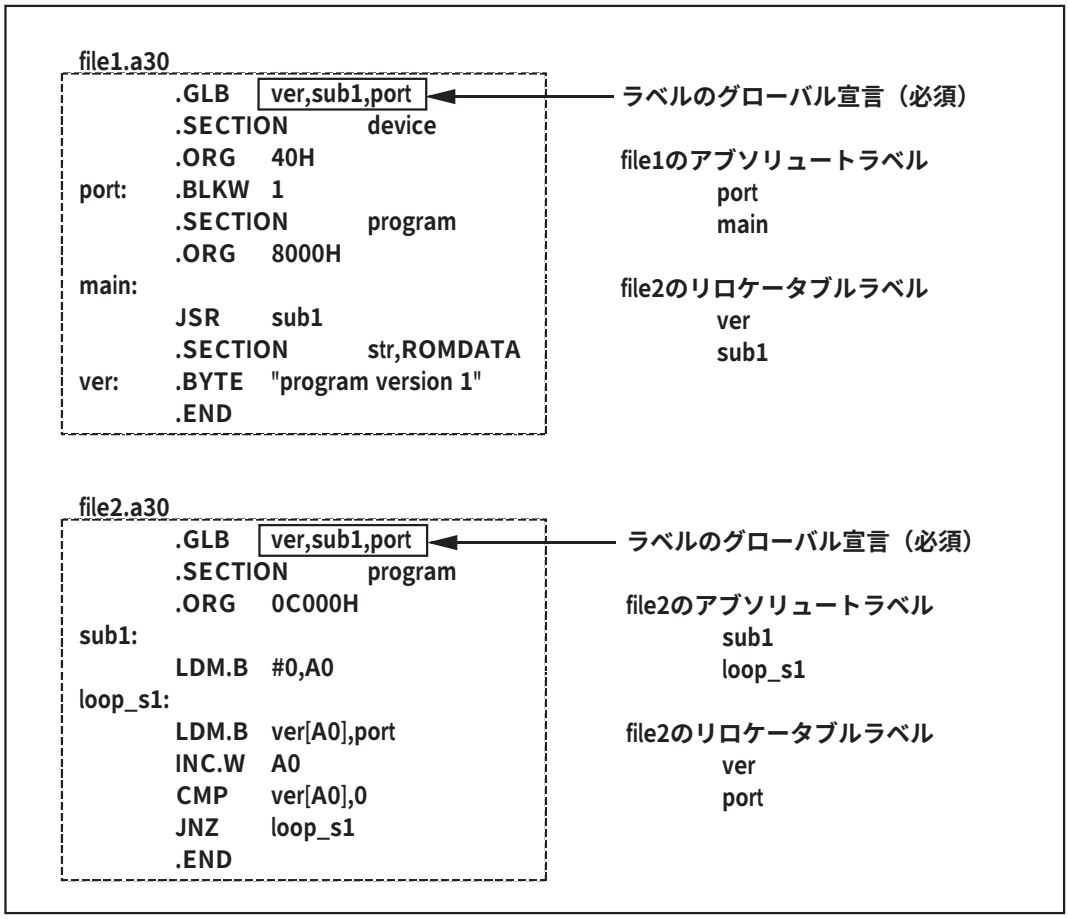
リロケータブル

- 相対属性セクション内のローカルラベル、シンボル、ビットシンボルの属性はリロケータブルです。
- セクションの属性に関係なくグローバルなラベル、シンボル、ビットシンボルの属性はリロケータブルです。

アブソリュート

- グローバル、ローカルの違いに関係なく、絶対属性セクション内で定義されているローカルラベル、シンボル、ビットシンボルの属性はアブソリュートです。

ラベル、シンボルおよびビットシンボルの値



アブソリュート属性

- アブソリュート属性をもつ名前の値はアセンブル実行時に決定されます。

リロケータブル属性

- リロケータブル属性をもつ名前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ワーニングが出力されます。

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。

ライブラリファイルの参照

次の条件の全てを満たした場合に、ln30 はライブラリファイルに登録されているリロケータブルモジュールをリンクします。

- コマンド行でライブラリファイル参照を指定した場合。
- 指定された全てのリロケータブルモジュールファイルを配置した結果、値の決定しなかったグローバルラベルが残っている場合。

注意事項

ln30 は、必要なグローバルラベルの定義を行っているリロケータブルモジュール全体をリンクします。

ライブラリモジュールの参照規則

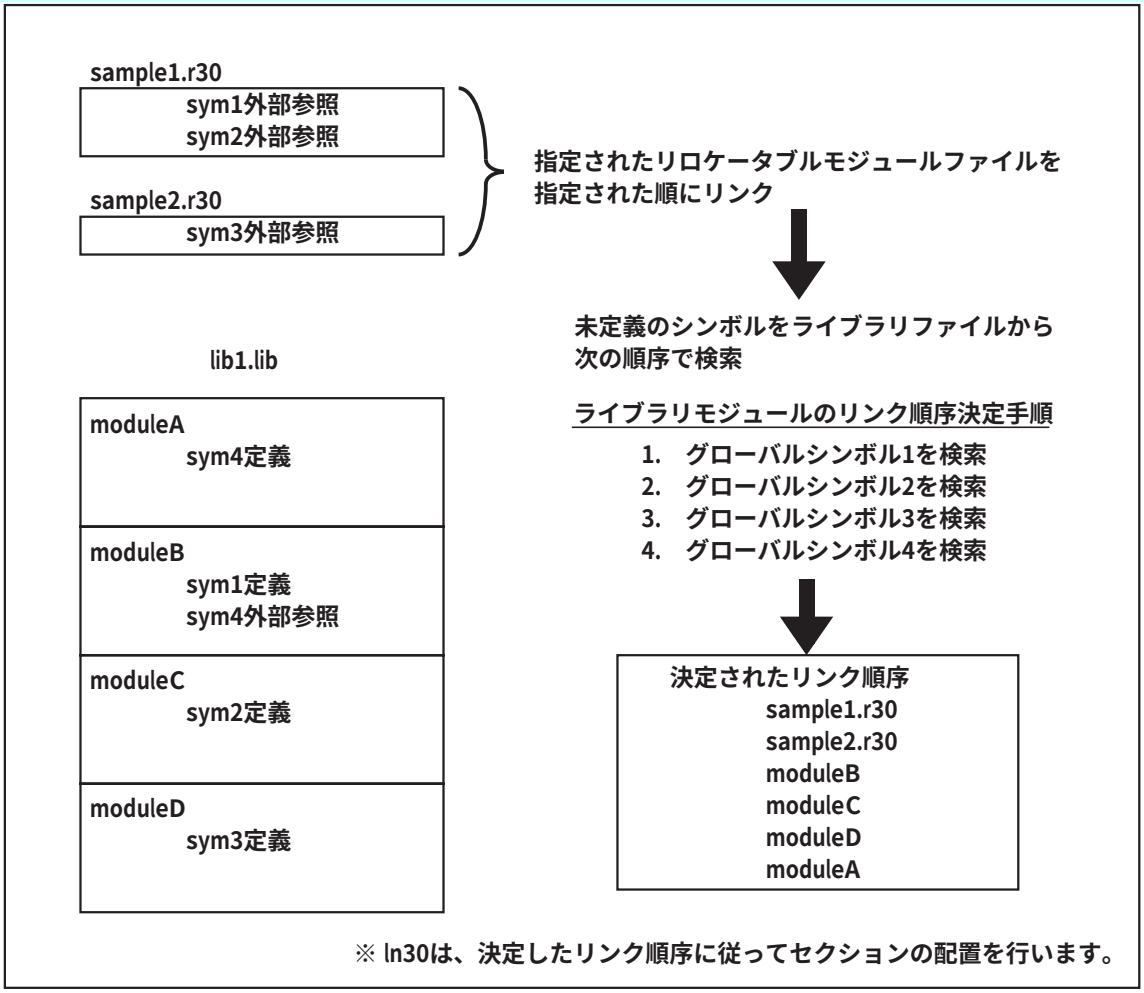
ln30 は、次の順序でリンクするリロケータブルモジュールを決定します。リンクすることが決定したリロケータブルモジュールは、セクション単位で再配置されます。セクションの再配置規則は、リロケータブルモジュールファイルのセクションの再配置規則と同様です。

1. ライブラリファイルに登録されている、リロケータブルモジュールのグローバルラベル情報を検索します。リロケータブルモジュールの参照は、ライブラリファイルに登録されている順に行います。
2. ライブラリファイルから検索したラベルと、値の決まっていないラベルとを比較し、一致するものがあれば、ライブラリファイル内のリロケータブルモジュールをリンクします。
3. ライブラリファイル内のリロケータブルモジュールを一巡した結果、値の決定していないグローバルラベルが残った場合（ライブラリファイルに登録されているリロケータブルモジュールに外部参照ラベルが存在した場合）、再度ライブラリファイルのモジュールを登録順に検索します。

ライブラリモジュールの参照例

コマンド入力例）

```
>ln30 sample1 sample2 -L lib1.lib
```



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。

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。

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。インクルードファイルの拡張子が'.inc'の場合は拡張子を省略できます。

ソース記述例）

ソースファイル(sample.a30)例

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

インクルードファイル(initial.inc)例

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

アセンブル実行後

```
                .SECTION    memory,DATA
00000   work:   .BLKB   10
0000A   flags:  .BLKW   1
                .SECTION    init
00000           .INCLUDE    initial
        loop:
00000           MOV.B       #10,A0
00002           MOV.B       #0,work[A0]
00006           INC.B       A0
00007           JNZ         loop
00009           MOV.W       #0,flags
                .SECTION    program,CODE
00000   main:
                  :
                .END
```

as30が出力したアドレス

コードの最適選択

as30 は、M16C 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

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。
- オペランドが、１つのラベルを含む式で記述されている場合。
  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。

注意事項

- 最適選択を行う条件を満たさない場合は、指示命令".OPTJ"の指定にしたがって、コードを生成します。
- グローバルラベルを参照している分岐命令を最適化する場合は、nc30 の”-OGJ(-Oglb_jmp)”、as30 の”-JOPT”、ln30 の”-JOPT”オプションを指定してください。ただし、これらオプションを指定した場合は、指示命令”.OPTJ”は無視されます。

最適選択規則

- 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'又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。

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選択を行います。
- 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。

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"又は".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）又は 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。

つぎ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。

SB 相対が選択される条件

- 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、64K バイトアドレス空間に対する 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。

注意事項
SB 相対アドレッシングを使用するためには、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実行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。

1 ビット操作命令の 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。

FB 相対が選択される条件

- 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ためには、必ず指示命令".FB"で FB レジスタ値を設定してください。

アドレッシングモードの最適選択例

次に、as30 が最適選択を行ったアドレッシングモードと、そのソース記述例を示します。

8 ビットディスプレースメント付きアドレスレジスタ相対
例）

```
sym1        .EQU    11H
            ABS.B   sym1+1[A0]
```

SB 相対
例 1）

```
sym2        .EQU    2
sym3        .EQU    3
    .SB     0
    .SBSYM  sym3
    ABS.B   sym3-sym2
```

例 2）

```
    .SB     100H
sym4        .EQU    108H
    ABS.B   sym4
```

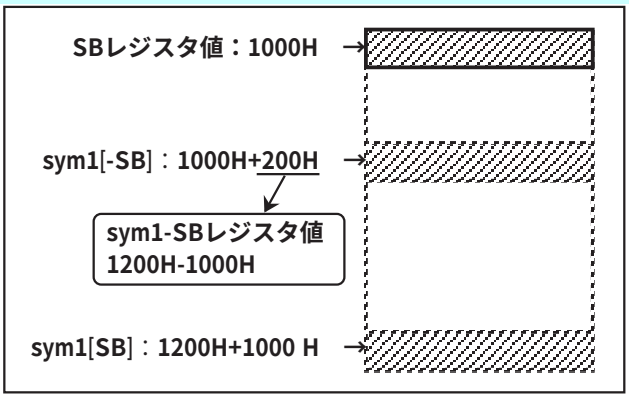
SBレジスタオフセットアドレス指定

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指定する記述ができます。

- 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、シンボル又は数値が記述できます。

記述例

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



スペシャルページベクタ参照

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 番地から 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、「M16C ファミリ 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。

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。

- セクションを必ず定義してください。
- 指示命令".ORG"で絶対アドレスを定義してください。

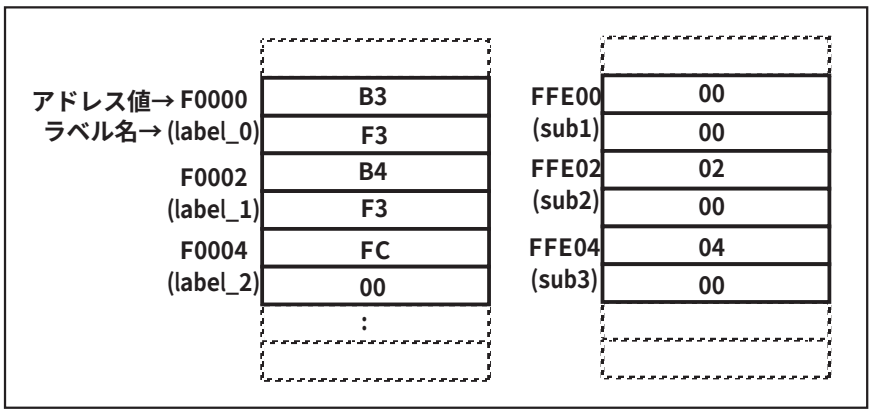
注意事項

アドレスは必ず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）

```
    .SECTIONsp_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
```



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。

1. 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。
2. スペシャルページ番号を指定する。

記述規則

- スペシャルページ番号を指定する場合は、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。

記述例 1: スペシャルページベクタテーブルのアドレスを指定

```
    .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
      :
```

マクロ機能

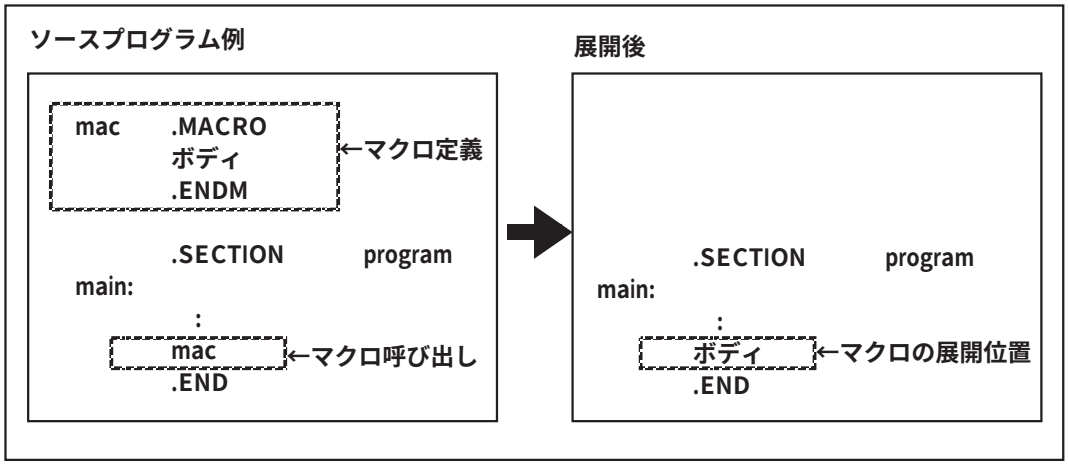
AS30 で使用できるマクロ機能について説明します。AS30 がサポートしているマクロ機能は 2 種類あります。

- 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。

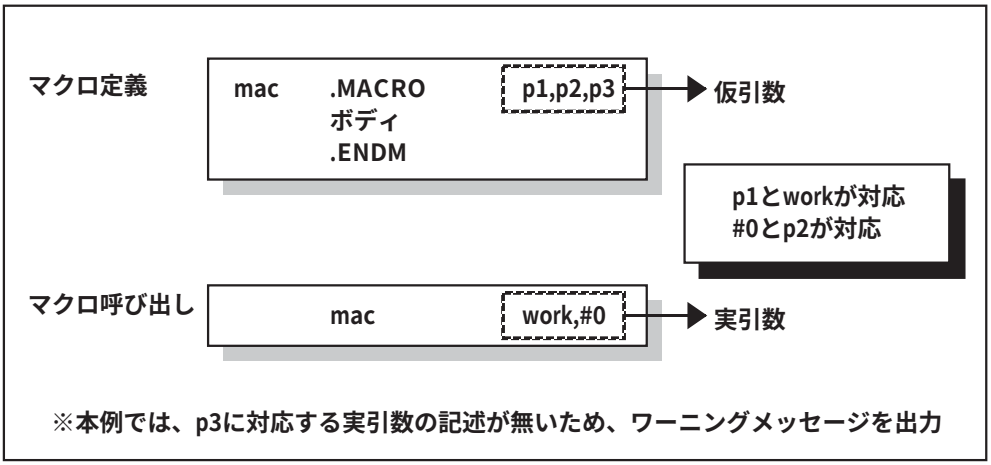
マクロ機能

- 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つのアセンブリ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。



マクローカルラベル

- 指示命令".LOCAL"で宣言したラベルはマクローカルラベルとなります。
- マクローカルラベルは、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、マクロボディの内容をマクロ呼び出しを行った行に展開します。
- 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。

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。

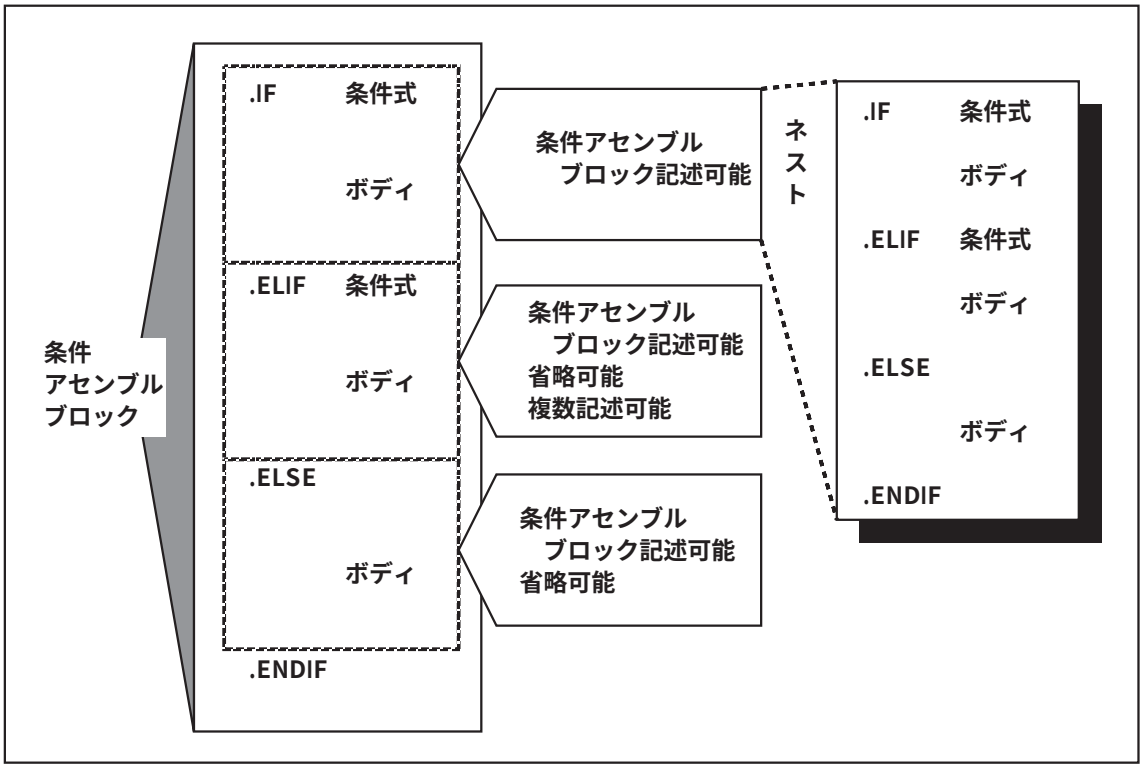
注意事項

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。

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ないので、ROM のサイズを節約できます。

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。アセンブリ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=2
.IF             TYPE==0
    .BYTE       "PROTO TYPE"
.ELIF           TYPE>0
    .BYTE       "MASS PRODUCTION    TYPE"
.ELSE
    .BYTE       "DEBUG MODE"
.ENDIF
    .END
```

コマンド入力例 1）

```
>as30 sample -Dtype=0
```

アセンブル結果 1）

```
    .SECTION    outdata,ROMDATA,ALIGN=2
    .BYTE       "PROTO TYPE"
    .END
```

コマンド入力例 2）

```
>as30 sample -Dtype=1
```

アセンブル結果 2）

```
    .SECTION    outdata,ROMDATA,ALIGN=2
    .BYTE       "MASS PRODUCTION TYPE"
    .END
```

コマンド入力例 3)

```
>as30 sample -Dtype=-1
```

アセンブル結果 3)

```
    .SECTIONoutdata,ROMDATA,ALIGN=2
    .BYTE                   "DEBUG MODE"
    .END
```

ソースファイル内に条件を記述する場合

アセンブリ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        0
    .SECTIONoutdata,ROMDATA,ALIGN
    .IF         type==0
                .BYTE       "PROTO TYPE"
    .IF         type>0
                .BYTE       "MASS PRODUCTION TYPE"
    .ELSE
                .BYTE       "DEBUG MODE"
    .ENDIF
    .END
```

構造化記述機能

AS30 プログラミングでは、構造化記述命令を使用した構造化記述が可能です。

- アセンブラは、構造化記述命令に対応するアセンブリ言語のブランチ命令を生成します。
- アセンブラは、生成したブランチ命令の分岐先のラベルを生成します。
- アセンブラは、構造化記述命令から生成したアセンブリ言語を、アセンブラリストファイルに出力します（コマンドオプション指定時）。
- 構造化記述命令は、構造化記述文とその条件式によって分岐する制御ブロックを選択できます。制御ブロックとは、代入文を除くある構造化記述文から次の構造化記述文までの間を言います。

ソース行情報の出力

as30 は、「M16C ファミリ用 C コンパイラ NC30」及び「AS30 のマクロ記述」のソースデバッグを実現するために必要な情報をリロケータブルモジュールファイルに出力します。

環境変数の参照

AS30 が参照する環境変数は次のものがあります。

環境変数名	環境変数を参照するプログラム
AS30COM	as30
BIN30	as30
INC30	as30
LIB30	ln30
TMP30	as30,ln30,lb30

AS30COM

as30 は、環境変数に設定されているコマンドオプションを添付して、ファイルを処理します。

"--"を使用することにより、本環境変数に設定したコマンドオプションを無効にできます。

次に示すコマンドオプションが、環境変数に設定できます。

オプション名	機能
L	アセンブラリストファイルの生成
N	デバッグ情報の出力停止
P	構造化記述命令処理指定
S	ローカルシンボル情報の出力
SM	システムラベルを含むローカルシンボル情報を出力
T	タグファイルの生成

AS30COM の設定方法

```
>SET AS30COM=-L -N -S -T
```

AS30COM の設定解除方法

```
>SET AS30COM=
```

AS30COM の使用例

as30 は、AS30COM が設定されている場合、次の順序でコマンドオプションを設定します。

1　AS30COM に設定されているコマンドオプションを設定します。

2　コマンド行から入力されたコマンドオプションを設定します。

次に、AS30COM にオプションを設定した例、コマンド行からコマンドオプションを入力した例及び有効となるコマンドオプションの例を示します。

AS30COM の設定例

```
>SET OPT30=-L -N -S -T
```

コマンド入力例　－1

```
>as30 -Dsym=0 --N
```

as30 実行時に有効となるオプション　－1

```
>as30 -Dsym=0 -L -S -T
```

コマンド入力例　－2

```
>as30 -O¥tmp --T -SM -LM
```

as30 実行時に有効となるオプション　－2

```
>as30 -O¥tmp -N -SM -LM
```

BIN30

- アセンブラドライバ（as30）は、設定されているディレクトリにあるマクロプロセッサ、構造化プリプロセッサ及びアセンブラプロセッサを起動します。これらのプログラムが格納されているディレクトリを BIN30 に設定してください。
- 複数のディレクトリ名を指定できます。複数のディレクトリを指定した場合は、左から順にディレクトリを検索します。

INC30

- as30 は、アセンブリソースファイルに記述されているインクルードファイルを、INC30 に設定されているディレクトリから検索します。
- 複数のディレクトリ名を指定できます。複数のディレクトリを指定した場合は、左から順にディレクトリを検索します。

LIB30

- ln30 は、LIB30 に設定されているディレクトリから、リンク対象となるライブラリファイルを検索します。
- 複数のディレクトリ名を指定できます。複数のディレクトリを指定した場合は、左から順にディレクトリを検索します。

TMP30

- プログラムは、ファイルを処理するための作業ファイルを、環境変数に指定されているディレクトリに生成します。
- 作業ファイルは、プログラムの処理終了時に消去されます。

環境変数設定例

複数のディレクトリを設定する場合、ディレクトリ名をセミコロンで区切って記述してください。

```
>SET INC30=C:¥COMMON;C:¥PROJECT
```

メッセージ出力

AS30 アセンブラ製品に含まれる各ソフトウェアは、処理の経過を画面に出力します。また、ファイルを処理する過程で発生したエラーを画面やファイルに出力します。

エラーメッセージ

各プログラム毎にエラーメッセージ又はワーニングメッセージが出力される場合があります。エラーメッセージの内容にしたがって、エラーの原因と思われる問題を解決して処理を再度実行してください。

エラーメッセージの種類

- エラーメッセージ
  メッセージを出力しファイルの生成などを行いません。エラータグファイルを生成する場合があります。
- ワーニングメッセージ
  メッセージを出力し規定のファイルを生成します。エラータグファイルを生成する場合があります。

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	OS のファイルシステムまたは、メモリシステムのエラーが発生しプログラム終了

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

入出力ファイル

AS30 で扱うファイルの種類を入力ファイルと、出力ファイルに分けて次の表に示します。ファイル名は任意に付けることができます。ただし、ファイル名の拡張子を省略して指定した場合、AS30 がデフォルトで表の（）内に示す拡張子を付加します。

as30 の入出力ファイル

入力ファイル	出力ファイル
ソースファイル(.a30) インクルードファイル(.inc)	リロケータブルモジュールファイル(.r30) アセンブラリストファイル(.lst) アセンブラエラータグファイル(.atg) 分岐情報ファイル(.jin)

ln30 の入出力ファイル

入力ファイル	出力ファイル
リロケータブルモジュールファイル(.r30) ライブラリファイル(.lib) 分岐情報ファイル(.jin) コマンドファイル	アブソリュートモジュールファイル(.x30) マップファイル(.map) リンクエラータグファイル(.ltg)

lmc30 の入出力ファイル

入力ファイル	出力ファイル
アブソリュートモジュールファイル(.x30)	モトローラ S フォーマットファイル(.mot) インテル HEX フォーマットファイル(.hex) ID ファイル（.id）

lb30 入出力ファイル

入力ファイル	出力ファイル
リロケータブルモジュールファイル(.r30) コマンドファイル	ライブラリファイル(.lib) リロケータブルモジュールファイル(.r30) ライブラリリストファイル(.lls)

xrf30 の入出力ファイル

入力ファイル	出力ファイル
ソースファイル(.a30) アセンブラリストファイル(.lst) コマンドファイル	クロスリファレンスファイル(.xrf)

abs30 の入出力ファイル

入力ファイル	出力ファイル
アブソリュートモジュールファイル(.x30) アセンブラリストファイル(.lst)	アブソリュートリストファイル(.als)

ソースファイル

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「プログラムの記述規則」に従って記述してください。

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。

インクルードファイル

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。

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。

リロケータブルモジュールファイル

as30 は、リロケータブルモジュールファイルを生成します。このファイルを ln30 がリンクして、アブソリュートモジュールファイルを生成します。

リロケータブルモジュールファイルフォーマット

as30 は、IEEE-695 に準拠したフォーマットのリロケータブルモジュールファイルを生成します。
注意事項
このファイルは、バイナリ形式のため、画面やプリンタに出力したり、編集したりできません。
このファイルをエディタでオープンしたり、編集した場合の以降の処理は正常に行われませんのでご注意ください。

リロケータブルモジュールファイル名

アセンブリソースファイルの拡張子（デフォルトで".a30"）を".r30"に変更したものが、リロケータブルモジュールファイルのファイル名になります。(sample.a30→sample.r30)

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、アセンブリソースファイルのあるディレクトリに生成します。

アセンブラリストファイル

as30 は、アセンブラリストファイルの生成を指示するコマンドオプションを指定した場合のみ、ソース行情報及びリロケータブル情報を出力可能なテキスト形式のファイルとして生成します。

アセンブラリストファイルフォーマット

以下に示す情報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出力フォーマットを《アセンブラリストファイル例－１》に示します。

(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(XMSDA)
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。

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 ビット変位 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....)
アセンブリソースファイルの内容を表示します。

アセンブラリストファイルの情報

アセンブラリストファイルには、次の情報が出力されます。

- ヘッダ情報
  アセンブラリストの１ページ毎に、次に示す情報をページの先頭に出力出力します。

```
* R8C/Tiny,M16C FAMILY ASSEMBLER *     SOURCE LIST          Tue Nov 18 12:02:53 1997    PAGE
002

  SEQ.   LOC.     OBJ.                  0XMSDA ....*....SOURCE STATEMENT....7....*....8
```

- アドレス定義行
  指示命令".ORG"でロケーションアドレスを定義した行を示します。

```
   11                                        .SECTIONRAM,DATA
   12   00400                                      .ORG       000400h
```

- 領域定義行
  指示命令".BLKB",".BLKW",".BLKA",".BLKL",".BLKF","BLKD"で領域定義を行った行を示します。

```
 55                                     .SECTION        ram1,data
 56  00000(000001H)                work1:   .BLKB     1
 57  00001(000001H)                work2:   .BLKB     1
 58  00002(000002H)                work3:   .BLKW     1
 59  00004(000002H)                work4:   .BLKW     1
```

- コメント行
  コメントのみを記述した行です。

```
 12                                     ;--------------------------------
 13                                     ; Macro define
```

- シンボル定義行
  指示命令".EQU"でシンボル定義を行った行を示します。

```
 65  00000001h                      sym1     .EQU     1
 66  00000002h                      sym2     .EQU     2
 67  00000003h                      sym3     .EQU     sym1 + sym2
```

- ビットシンボル定義行
  指示命令".BTEQU"でビットシンボル定義を行った行を示します。

```
 62  1,00000000h                    flag1    .BTEQU   1,0
 63  2,00000000h                    flag2    .BTEQU   2,0
```

- 定数データ定義行
  指示命令".BYTE",".WORD",".ADDR",".DWORD",".FLOAT",".DOUBLE"で ROM 領域にデータを設定した行を示します。

```
175  0003E   41                M              .BYTE    "A"
176  0003F   42                M              .BYTE    "B"
177  00040   43                M              .BYTE    "C"
178  00041   44                M              .BYTE    "D"
```

- マクロ定義行
  マクロ定義を行っている行です。

```
 46                                     mac5     .MACRO   p1
 47                                D              .MREPEAT   .LEN{'p1'}
 48                                D              .BYTE    .SUBSTR{'p1', ..MACREP,1}
 49                                D              .ENDR
 50                                D              .ENDM
```

- ラベル定義行
  ラベルのみを記述した行です。

```
 70  00000                          samp_start:
```

- ニーモニック記述行
  M16C ファミリのニーモニックを記述した行です。

```
 71  00000   4100                 S*             BCLR     flag1
 72  00002   4200                 S*             BCLR     flag2
```

- 条件アセンブル情報行
  条件アセンブルを行った行を示します。コマンドオプション(-LI/LMI)を指定した場合のみ、出力されます。

```
  74                                        .IF       MODE == 1
  75                                  X     MOV.B     #sym1,R0L
  76                                        .ELIF     MODE == 2
  77                                  X     MOV.B     #sym2,R0L
  78                                        .ELSE
  79  00004   B4                         Z   MOV.B     #0,R0L
  80                                        .ENDIF
```

- マクロ呼び出し行
  マクロ呼び出しを行っている行です。コマンドオプション(-LM)を指定した場合は、マクロを展開した結果のアセンブリソース行を出力します。

```
 173                                        mac5      ABCD
```

- 構造化記述行
  構造化記述命令でプログラムを記述した行です。

```
  42                                        for A0 < A1
  43  F800A                           S       ..fr0000:
  44  F800A   C154                    S     CMP.W     A1,A0
  45  F800C   680B                    S     JC        ..fr0002
  46                                            [ WORK_W ] = [ A0 ]
  47  F800E   736F0104                S     MOV.W     [A0],WORK_W
  48                                            [ A0 ] = [ A1 ]
  49  F8012   7376                     S             MOV.W     [A1],[A0]
  50                                            A0 = ++A0
  51  F8014   B2                       S             INC.W     A0
  52                                            A1 = --A1
  53  F8015   FA                       S             DEC.W     A1
  54                                        next
  55  F8016   FEF3                     SB            JMP       ..fr0000
  56  F8018                            S       ..fr0002:
```

- インクルードファイル表示行
  読み込んだインクルードファイルを表示した行です。

```
  65                                            .INCLUDE      sample.inc
  66                                  1             .SECTION      ram,DATA
  67  00000(000008H)                  1       work8:    .BLKD         1
  68  00008(000004H)                  1       work_4: .BLKF         1
  69                                  1             .SECTION      constdata,ROMDATA
  70  00000   3031323334353637      1       num_val:.BYTE         "0123456789"
                  3839                        1
```

- アセンブル結果情報
  アセンブルを行った結果の全エラー数、全ワーニング数及び全リスト行数を出力します。

```
Information List

TOTAL ERROR(S)      00000
TOTAL WARNING(S)    00000
TOTAL LINE(S)       00181    LINES
```

- セクション情報
  セクションのタイプ、セクションサイズ及びセクション名をリストにして出力します。

```
Section List

Attr        Size                Name
DATA        0000006(00006H)     ram1
CODE        0000066(00042H)     prog1
```

アセンブラリストファイル名

アセンブリソースファイルの拡張子（デフォルトでは".a30"）を".lst"に変更したものが、アセンブラリストファイルのファイル名になります。(sample.a30→sample.lst)

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、アセンブリソースファイルのあるディレクトリに生成します。

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/-X)を指定した場合のみアセンブリ、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。

このファイルをタグジャンプ機能を持つエディタで処理することによりエラーの修正を容易に行えます。

アセンブラエラータグファイルフォーマット

エディタのタグジャンプ機能を使用できるフォーマットになっています。
以下に示すように、アセンブリソースファイル名、エラー行番号、エラーメッセージの順に出力します。

```
sample.err 21 Error (asp30): Operand value is not defined
sample.err 72 Error (asp30): Undefined symbol exist "work2"
```

アセンブラエラータグファイル名

アセンブリ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)

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、アセンブリソースファイルのあるディレクトリに生成します。

分岐情報ファイル

as30 は、-JOPT オプションを指定した場合、グローバルラベルを参照している分岐命令が存在する a30 ファイルに対して、外部分岐の最適化を行うための分岐情報ファイルを生成します。

分岐情報ファイルフォーマット

分岐情報ファイルは、as30 と ln30 の内部処理専用ファイルです。
注意事項
このファイルは、編集しないでください。編集した場合の以降の処理は正常に行われませんのでご注意ください。

分岐情報ファイル名

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".jin"に変更したものが分岐情報ファイルのファイル名になります。(sample.a30→sample.jin)

ファイル生成ディレクトリ

リロケータブルモジュールファイルと同じディレクトリに生成します。

コマンドファイル

コマンド行に入力するコマンドオプションの指定内容を記述したファイルです。コマンド行に入力可能な文字数が限られているため、指定するファイル数が多くコマンド行から入力できない場合に使用します。

コマンドファイルの指定方法

コマンドファイルをプログラム起動時に指定する方法を次に示します。コマンドファイルを利用できるプログラムは、ln30、lb30 および xrf30 のみです。

- コマンドファイル名の先頭には、@を付けて指定してください。
- コマンドファイル名には、ディレクトリパスを指定できます。
- 指定したディレクトリパスにファイルが存在しない場合は、エラーとなります。

コマンドファイル指定例）

```
>ln30 @cmdfile
```

コマンドファイルフォーマット

コマンドファイルの記述規則を説明します。

- コマンドファイル内に、コマンドファイル自身の名前は記述できません。
- コマンドファイルにはコマンドパラメータを複数行にわたって記述できます。
- コマンドファイルに記述する行の行頭及び行末には、","（カンマ）の記述はできません。
- セクション配置指定を複数行にわたって記述したい場合は、改行した後で "-ORDER" オプションを行の先頭に記述してください。
- ファイルの一行に記述可能な文字数は、2048 文字までです。2048 文字を越えた場合はエラーとなります。
- コマンドファイルにはコメントを記述することができます。コメントを記述する際には、コメントの先頭に"#"を記述してください。＃以降改行文字までをコメントとして処理します。

コマンドファイルの記述例）

```
#sample of command file
# Relocatable module file
sample1   sample2   sample3
# set to start address
-ORDER   ram=80
-ORDER   prog,sub,data
-M
```

コマンドファイル名

任意のファイル名で記述できます。

アブソリュートモジュールファイル

ln30 は、複数のリロケータブルモジュールファイルから、一つのアブソリュートモジュールファイルを生成します。

アブソリュートモジュールファイルフォーマット

このファイルは、IEEE-695 に準拠したフォーマットになっています。

注意事項

このファイルは、バイナリ形式のため、画面やプリンタに出力したり、編集したりできません。このファイルをエディタでオープンしたり、編集したりした場合の以降の処理は正常に行われませんのでご注意ください。

アブソリュートモジュールファイル名

通常は、コマンド行から入力された、リロケータブルモジュールファイル名のうちの１番目のファイル名の拡張子".r30"を".x30"に変更したものが、アブソリュートモジュールファイル名になります。

(sample.r30→sample.x30)
コマンドオプション(-O)で、ファイル名を指定した場合は、指定した名前でファイルが生成されます。

ファイル生成ディレクトリ

通常は、カレントディレクトリに生成します。
コマンドオプション(-O)のファイル名にパスが指定されている場合は、そのパスのディレクトリに生成します。

マップファイル

ln30 は、コマンドオプション(-M/MS)を指定した場合のみ、リンク情報、セクションの最終配置アドレス情報、シンボル情報などをマップファイルに出力します。シンボル情報は、コマンドオプション(-MS)を指定した場合だけマップファイルに出力します。

マップファイル生成のオプションを指定しているときは、セクションオーバーラップエラーが発生した場合でもマップファイルを生成します。

マップファイルフォーマット

次に、示す情報を順にリスト形式でマップファイルに出力します。マップファイルの出力フォーマットを《マップファイル例》に示します。

(1) リンク情報
コマンド行情報、リロケータブルモジュールファイル名及びリロケータブルモジュールファイルの生成日時情報、指示命令“.VER”、“.ID”、“.OFSREG”及び“.PROTECT”情報を出力します。

(2) セクション情報
再配置されたセクション名、属性、タイプ、スタートアドレス、セクションサイズ、セクション整列の有無及びモジュール名（リロケータブルモジュールファイル名）の情報を出力します。

\# VARIABLE VECTOR TABLE INFORMATION （可変ベクタテーブル情報）
可変ベクタテーブルの情報を出力します。

\# SPECIAL VECTOR TABLE INFORMATION （スペシャルページベクタテーブル情報）
スペシャルページベクタテーブルの情報を出力します。

(3) グローバルラベル情報
グローバルラベル名とアドレス情報を出力します。コマンドオプション"－MS"を指定した場合のみ出力されます。

(4) グローバルシンボル情報
グローバルシンボル名と数値情報を出力します。コマンドオプション"－MS"を指定した場合のみ出力されます。

(5) グローバルビットシンボル情報
グローバルビットシンボル名とビット位置及びメモリアドレス情報を出力します。コマンドオプション"－MS"を指定した場合のみ出力されます。

(6) ローカルラベル情報
モジュール名（リロケータブルモジュールファイル名）、ローカルラベル名及びアドレス情報を出力します。コマンドオプション"－MS"を指定した場合のみ出力されます。

(7) ローカルシンボル情報
モジュール名（リロケータブルモジュールファイル名）、ローカルシンボル名及び数値情報を出力します。コマンドオプション"－MS"を指定した場合のみ出力されます。

(8) ローカルビットシンボル情報
モジュール名（リロケータブルモジュールファイル名）、ローカルビットシンボル名、ビット位置及びメモリアドレス情報を出力します。コマンドオプション"－MS"を指定した場合のみ出力されます。

マップファイルサンプル

```
#############################################
# (1) LINK INFORMATION                      #
#############################################
ln30 -ms smp

# ID CODE DATA
ID   “Code”

# ROM CODE PROTECT DATA
PROTECT   12H

# LINK FILE INFORMATION
smp (smp.r30)
    Jun 27 14:58:58 1995


#############################################
# (2) SECTION INFORMATION                   #
#############################################
# SECTION                   ATR   TYPE      START   LENGTH ALIGN MODULENAME
ram                         REL   DATA      000000   000014              smp
program                     REL   CODE      000014   000003              smp
# Total ---------------------------------------------------------------
                                  DATA                 000014(0000020) Byte(s)
                                  ROMDATA              000000(0000000) Byte(s)
                                  CODE                 000003(0000003) Byte(s)

#############################################
# (3) GLOBAL LABEL INFORMATION              #
#############################################
work                    000000

#############################################
# (4) GLOBAL EQU SYMBOL INFORMATION         #
#############################################
sym2                    00000000

#############################################
# (5) GLOBAL EQU BIT-SYMBOL INFORMATION     #
#############################################
sym1                    1 000001

#############################################
# (6) LOCAL LABEL INFORMATION               #
#############################################
@ smp ( smp.r30 )
main                    000014    tmp                   00000a

#############################################
# (7) LOCAL EQU SYMBOL INFORMATION          #
#############################################
@ smp ( smp.r30 )
sym3                    00000003

#############################################
# (8) LOCAL EQU BIT-SYMBOL INFORMATION      #
#############################################
@ smp ( smp.r30 )
sym4                    1 000000
```

マップファイル名

アブソリュートモジュールファイルの拡張子(.x30)を".map"に変更したものが、マップファイルのファイル名になります。(sample.x30→sample.map)

ファイル生成ディレクトリ

アブソリュートモジュールファイルと同じディレクトリに生成します。

リンクエラータグファイル

ln30 は、コマンドオプション(-T)を指定した場合のみ、リンクエラー情報をファイルに出力します。このとき、エラーが発生した箇所をアセンブリソース行で出力します。

このファイルをタグジャンプ機能を持つエディタで処理することによりエラーの修正を容易に行えます。

リンクエラータグファイルフォーマット

フォーマットは、アセンブラエラータグファイルと同じです。エディタのタグジャンプ機能が使用できます。
次に示すように、アセンブリソースファイル名、エラー行番号、エラーメッセージの順に出力します。

```
smp.inc 2 Warning (ln30): smp2.r30 : Absolute-section is written after the absolute-section 'ppp'
smo.inc 2 Error (ln30): smp2.r30 : Address is overlapped in 'CODE' section 'ppp'
```

リンクエラータグファイル名

アブソリュートモジュールファイルの拡張子(.x30)を".ltg"に変更したものがリンクエラータグファイルのファイル名になります。(sample.x30→sample.ltg)

ファイル生成ディレクトリ

アブソリュートモジュールファイルと同じディレクトリに生成します。

モトローラＳフォーマットファイル

lmc30 は、X30 ファイルから一つのモトローラＳフォーマットファイル（以降'MOT ファイル'と記述します）を生成します。

モトローラＳフォーマットファイル名

X30 ファイルの拡張子(.x30)を".mot"に変更したものが MOT ファイルのファイル名になります。(sample.x30→sample.mot)

ファイル生成ディレクトリ

ファイルはカレントディレクトリに生成されます。

モトローラＳフォーマットファイルフォーマット

マイクロコンピュータの内蔵 ROM や EPROM などに書き込み可能なモトローラ S フォーマットです。
モトローラＳフォーマットファイルを生成する際に次のデータを指定できます。

- 0H～0FFFFFFH 番地までのデータ領域を設定できます。
- 1 データレコード長を 16 バイト又は 32 バイトのいずれかに設定できます。
- 実行開始アドレスを設定できます。
- lmc30 はアドレスが昇順になっているファイルを生成します。

インテルHEXフォーマットファイル

lmc30 は、X30 ファイルから一つのインテル HEX フォーマットファイル（以降 HEX ファイルと記述します）を生成します。

インテル HEX フォーマットファイル名

X30 ファイルの拡張子(.x30)を".hex"に変更したものが HEX ファイルのファイル名になります。
(sample.x30→sample.hex)

ファイル生成ディレクトリ

ファイルはカレントディレクトリに生成されます。

インテル HEX フォーマットファイルフォーマット

コマンドオプション(-H)を指定した場合のみ、マクロコンピュータの内蔵 ROM や EPROM などに書き込み可能なインテル HEX フォーマットです。
HEX ファイルを生成する際に次のデータを指定できます。

- 0H～0FFFFFFH 番地までのデータ領域を設定できます。
- lmc30 はアドレスが昇順になっているファイルを生成します。
- 1 データレコード長を 16 バイト又は 32 バイトのいずれかに設定できます。

注意事項
1Mbyte 空間(0H～0FFFFFH)を越えるアブソリュートモジュールファイルの場合は、専用 HEX フォーマットで出力します。

IDファイル

lmc30 は、アセンブラ指示命令 “.ID” またはコマンドオプション “-ID” が指定されている場合、ID コードチェック機能の ID コードファイルを生成します。

ID ファイルフォーマット

画面やプリンタに出力可能なテキスト形式です。このファイルを参照することで、ID コードチェック機能の ID コードを確認することができます。
コマンドオプション情報、ID コード格納番地情報の順に出力します。

```
－IDCodeNo1
FFFDF   : 43
FFFE3   : 6F
FFFEB   : 64
FFFEF   : 65
FFFF3   : 4E
FFFF7   : 6F
FFFFB   : 31
```

ID ファイル名

X30 ファイルの拡張子(.x30)を".id"に変更したものが ID ファイルのファイル名になります。
(sample.x30→sample.id)

ファイル生成ディレクトリ

ファイルはカレントディレクトリに生成されます。

ライブラリファイル

lb30 は、as30 が生成したリロケータブルモジュールファイルをモジュールとして、一つのファイルにまとめたライブラリファイルを生成します。

ライブラリファイルフォーマット

このファイルは、IEEE-695 フォーマットに準拠しています。

注意事項

このファイルは、バイナリ形式のため、画面やプリンタに出力したり、編集したりできません。このファイルをエディタでオープンしたり、編集したりした場合の以降の処理は正常に行われませんのでご注意ください。

ライブラリファイル名

コマンド行で指定した名前のライブラリファイルが生成されます。拡張子は".lib"です。コマンド行で、ライブラリファイル名を省略できません。

ファイル生成ディレクトリ

コマンド行でパスを指定した場合は、そのディレクトリに生成されます。パスの指定が無い場合はカレントディレクトリに生成されます。

ライブラリリストファイル

lb30 は、ライブラリファイル及びライブラリファイルに登録されている任意のリロケータブルモジュールのリストファイルを生成します。

ライブラリリストファイルフォーマット

画面やプリンタに出力可能なテキスト形式です。このファイルを参照することで、ライブラリファイルに登録されているリロケータブルモジュールの概要を確認することができます。
次に、ライブラリリストファイルに出力される情報を示します。

(1) ライブラリ情報
　１つのライブラリファイルに、１回出力されます。次に示す情報を出力します。

- ライブラリファイル名
  ライブラリファイル名を出力します。
- ファイル更新日時
  ライブラリファイルの最新の更新日時を出力します。
- モジュール数
  ライブラリファイルに登録されているモジュールの総数を出力します。
- グローバルシンボル数
  ライブラリファイルに登録されているグローバルラベル及びグローバルシンボルの総数を出力します。

(2) モジュール情報
　登録されているモジュール毎に、１回出力されます。次に示す情報を出力します。

- モジュール名
  ライブラリファイルに登録されているモジュール名を出力します。
- バージョン情報
  指示命令".VER"で指定された文字列情報を出力します。
- 登録日時
  モジュールをライブラリファイルに登録した日時を示します。
- モジュールサイズ
  登録されているモジュールのコード及びデータサイズを出力します。
  リロケータブルモジュールファイルのファイルサイズとは異なります。
- グローバルシンボル名
  モジュール内で定義されているグローバルシンボル及びグローバルラベル名を出力します。

・ 外部参照シンボル名
　モジュールが外部参照しているグローバルシンボル及びグローバルラベル名を出力します。

```
Librarian (lb30) for M16C Family Version 1.00.00
Library file name:                libsmp.lib
Last update time:                 1995-Jul-7 15:44
Number of module(s):              1
Number of global symbol(s):       12

Module name:            sample
.Ver:                        .VER          "sample program file"
Date:                        1995-Jul-7 15:43
Size:                        00894H
Global symbol(s):       btsym5   btsym6   btsym7
                             btsym8   btsym9   sub1
                             sub2     sym5     sym6
                             sym7     sym8     sym9
```

ライブラリリストファイル名

コマンド行で指定した名前のライブラリファイルが生成されます。拡張子は".lls"です。コマンド行で、ライブラリファイル名を省略できません。

ファイル生成ディレクトリ

コマンド行でパスを指定した場合は、そのディレクトリに生成されます。パスの指定が無い場合はカレントディレクトリに生成されます。

クロスリファレンスファイル

xrf30 は、アセンブリソースファイルを基に、シンボル及びラベルの定義と参照の情報をまとめたファイルを生成します。

クロスリファレンスファイルフォーマット

画面やプリンタに出力可能なテキスト形式です。従って、デバッグなどの際にプリントアウトして、シンボルを定義したアセンブリソースファイル内の位置を確認できます。

クロスリファレンスファイルの情報

次にクロスリファレンスファイルに出力される情報を示します。

(1) ラベル名
　ラベル名を出力します。

(2) ファイル名
　上記ラベルの記述されているファイル名を出力します。

(3) 参照及び宣言されている行番号とその区別を示す記号
　行番号に次に示す記号を添付して出力します。
　:d　定義行
　:j　分岐命令による参照行
　:s　サブルーチン呼び出し命令による参照行

```
btsym0
  sample.a30
    00023:d
btsym1
  sample.a30
    00024:d
btsym2
  sample.a30
    00025:d
btsym20
  sample.a30
    00033:d
```

クロスリファレンスファイル名

アセンブラリストファイル名又はアセンブリソースファイル名の拡張子(.lst)を".xrf"に変更したものがクロスリファレンスファイル名になります。(sample.lst → sample.xrf;sample.a30 → sample.xrf)
複数のファイル名を指定した場合は、先頭のファイル名の拡張子を".xrf"に変更したものがクロスリファレンスファイル名になります。

ファイル生成ディレクトリ

コマンド行で、パスを指定した場合は、そのディレクトリにファイルを生成します。
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ファイルを生成します。
上記のディレクトリ指定が無い場合は、カレントディレクトリにファイルを生成します。

アブソリュートリストファイル

abs30 は、画面やプリンタに出力可能なフォーマットのアブソリュートリストファイルを生成します。

アブソリュートリストファイルフォーマット

アブソリュートリストファイルのフォーマットは、ロケーション情報が絶対アドレス情報に変換される点を除いてはアセンブラリストファイルと同一です。

アブソリュートリストファイル名

アセンブラリストファイルの拡張子(.lst)を".als"に変更したものがアブソリュートリストファイル名になります。(sample.lst→sample.als)

ファイル生成ディレクトリ

コマンドオプション(-O)が指定されている場合は、そのディレクトリにファイルを生成します。
上記以外の場合は、カレントディレクトリにファイルを生成します。

プログラムの起動方法

AS30 に含まれる各プログラムの基本的な操作方法について説明します。

コマンド入力時の注意事項

・MS-DOS 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。

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

プログラムを正しく実行するために必要な情報全てをコマンドパラメータと呼びます。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が含まれます。
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。

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。
- ファイル名の長さは、ディレクトリ指定を含めて 512 文字（バイト）以下です。ただし、起動するプログラム名やコマンドオプションを含めてコマンド行の文字数が既定の範囲に収まるようにしてください。
- 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、一箇所だけです。
- 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。
- コマンドオプションを指定する際には、必ず"ー"を添付して記述してください

as30 の操作方法

as30 のコマンドパラメータの指定規則を説明します。

as30 コマンドパラメータ

オプション	機能
-.	画面へのメッセージ出力を停止する
-A	ニーモニックオペランドを評価する
-C	as30 が mac30、pre30 及び asp30 に渡したコマンド行を表示する
-D	シンボル定義を行う
-finfo	インスペクタ情報を生成する
-F	..FILE が示すファイル名をソースファイル名に固定する
-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
-M60	M16C/60 グループに対応したコードを生成する
-M61	M16C/61 グループに対応したコードを生成する
-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
-JOPT	グローバルラベルを参照している分岐命令を最適化する
-L	アセンブラリストファイルを生成する
-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
-N	高級言語のソース行情報を出力しない
-O	生成ファイルの生成ディレクトリを指定する
-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
-P	構造化記述命令を変換する
-R8C	R8C ファミリ（アドレス空間 0～0FFFFH）に対応したコードを生成する
-R8CE	R8C ファミリ（アドレス空間 0～0FFFFFH）に対応したコードを生成する
-R8Cxx	チップセレクト付クロック同期シリアル（SSU）または I^2C 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
-S	ローカルシンボル情報を出力する
-T	アセンブラエラータグファイルを生成する
-V	全てのプログラムのバージョン番号を表示する
-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

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リソースファイル名は任意の順序で指定できます。

アセンブリソースファイル名（必須）

- 必ず、1 つ以上のアセンブリソースファイル名を指定してください。
- アセンブリソースファイル名には、パスの指定ができます。
- アセンブリソースファイル名は、80 個まで指定できます。
  複数指定されたアセンブリソースファイルのうち、エラーを含むアセンブリソースファイルの処理以降のファイルは処理しません。
- 拡張子が".a30"であるファイルは、拡張子を省略できます。

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。

- 後に続くコマンドオプションを無効にするには、"－－"を添付してコマンドオプションを記述してください。

注意事項
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)
>as30 sample -S -L --S       …(4)
```

(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"を指定
(4)コマンドオプション"L"を指定

数値の指定方法

- 数値は、必ず 16 進数で指定してください。
- 数値の先頭がアルファベットになる場合は、必ず 0 を付加して指定してください。

数値の指定例）

```
55
5A
0A5
```

as30 コマンドオプション

以降に、コマンドオプションの指定方法を説明します。

-.

画面へのメッセージ出力停止

機能

- 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、ワ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。

記述規則

- コマンド行の任意の位置に指定できます。

記述例

```
>as30 -. sample
```

エラー出力例）

```
>as30 -. sample

sample.a30 2 Error (as30): Section type is not appropriate
```

-A

ニーモニックオペランドの評価

機能

- 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ワーニングを出力します。

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ワーニングを出力します。

記述規則

- コマンド行の任意の位置に指定できます。

記述例

```
>as30 -A sample
```

-C

コマンド起動行表示

機能

- 環境変数(AS30COM)にコマンドオプションを指定している場合に、本オプションを指定すると、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。

記述規則

- コマンド行の任意の位置に指定できます。

記述例 1

```
>as30 -C -N sample
```

- AS30COM に'-L　-T'が設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 -N sample

( sample.a30 )
mac30 -L -T sample.a30
macro prossesing now

asp30 -L -T sample.m30
assembler prossesing now
TOTAL ERROR(S)          00000
          :
```

記述例 2

```
>as30 -. -C -N sample
```

- 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 sample
mac30 -L -T sample.a30

asp30 -L -T sample.m30
```

-D

シンボル定数設定

機能

- シンボルに値を設定します。
- 値は絶対値として扱います。

注意事項

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。

- 本オプションで定義したシンボルは、アセンブリソースファイル内に記述したシンボル定義と同じに扱われます。つまり、アセンブリ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。
- コマンド行で複数のファイルを指定した場合、本オプションで定義したシンボルは、全てのファイル内で定義されます。

記述規則

- －D（シンボル名）＝（数値）のように指定してください。
- コマンド行の任意の位置に記述できます。
- 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。

記述例

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

-finfo

インスペクタ情報を生成

機能

- 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リロケータブルモジュールファイルに出力します。

注意事項

TM をご使用の場合、本オプションはデフォルトで指定されます。

記述規則

- コマンド行の任意の位置に指定できます。
- 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。

記述例

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

-F

..FILE 展開制御

機能

- 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リソースファイル名に固定します。

記述規則

- コマンド行の任意の位置に記述できます。

記述例

```
as30 -F sample
```

sample.a30 アセンブリ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。

-H

アセンブラリストファイルのヘッダ出力停止

機能

・ アセンブラリストファイルのヘッダ情報が出力されません。

注意事項
abs30 が処理するアセンブラリストファイルを生成する場合は、本オプションを指定しないでください。

記述規則

・ コマンド行の任意の位置に記述できます。
・ コマンドオプション'-L'と同時に指定してください。

記述例

sample.lst ファイルにヘッダ情報を出力しません。

```
>as30 -L -H sample
```

-I

インクルードファイル検索ディレクトリ指定

機能

・ 指定されたディレクトリからソースファイルに記述されている".INCLUDE"で指定されたインクルードファイルを検索します。

記述規則

・ コマンド行の任意の位置に指定できます。
・ "-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。

記述例

¥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。

```
>as30 –I¥work¥include
```

-JOPT

外部分岐の最適化

機能

・ グローバルラベルを参照している分岐命令(JMP,JSR)を最適化します。

注意事項
・ 本オプションを指定する場合、nc30 の”-OGJ(-Oglb_jmp)”および ln30 の”-JOPT”オプションを指定してください。
・ 本オプションを指定した場合、分岐情報ファイル（拡張子.jin）が生成されます。分岐情報ファイルは編集しないでください。また、拡張子.jin は使用しないでください。
・ 本オプションと同時に、指示命令”.OPTJ”が使用されている場合は、本オプションが有効になります。

記述規則

・ コマンド行の任意の位置に指定できます。

記述例

```
>as30 -JOPT sample
```

-L

アセンブラリストファイル生成

機能

- リロケータブルモジュール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"となります。
- 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。

記述規則

- コマンド行の任意の位置に指定できます。
- 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。
- 本オプションは、環境変数"AS30COM"に設定できます。

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。

記述例

```
>as30 -LMI sample
>as30 -LMS sample
>as30 -LCDIMS sample
```

-M

構造化記述命令の変数をバイト型で生成

機能

- 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。

記述規則

- コマンド行の任意の位置に指定できます。
- 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。

記述例

```
>as30 -P -M sample
>as30 -M -P sample
```

-M60/-M61

生成コード制御

機能

・ 各オプションを付加した場合のアセンブラの処理は次のようになります。

オプション名	処理内容
-M60	'SHL','SHA','ROT'命令の直後に常に'NOP'命令を挿入します。 'JMP.A'及び'JSR.A'の命令の直前に常に'JMP.B'を挿入します。
- M61	アセンブラは処理を行いません。ln30 の機能を参照してください。

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。

記述規則

・ コマンド行の任意の位置に指定できます。

記述例

```
>as30 -M60 sample
```

-N

行情報の出力停止

機能

・ C 言語のソース行情報をリロケータブルモジュールファイルに出力しません。
・ リロケータブルモジュールファイルのサイズを縮小できます。

注意事項
本オプションを指定して生成したリロケータブルモジュールファイルから、作成したアブソリュートモジュールファイルでは、ソース行レベルでのデバッグはできません。

記述規則

・ コマンド行の任意の位置に指定できます。
・ 本オプションは、環境変数"AS30COM"に設定できます。設定方法は、「AS30COM の使用例」を参照してください。

記述例

```
as30 -N sample
```

-O

生成ファイルの出力先ディレクトリ指定

機能

- アセンブラが生成するリロケータブルモジュールファイル、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。

記述規則

- -O（ディレクトリ名）のように記述してください。
- 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。

記述例

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

(1) リロケータブルモジュールファイルを、C ドライブの¥work¥asmout ディレクトリに生成します。
(2) リロケータブルモジュールファイルを、カレントディレクトリの親ディレクトリに属す、tmp ディレクトリに生成します。
(3) リロケータブルモジュールファイル、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 ディレクトリに生成します。

-P

構造化記述命令を処理

機能

- アセンブリ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。

記述規則

- コマンド行の任意の位置に指定できます。
- 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。

記述例

アセンブリ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。

```
>as30 -P -LS sample
```

-PATCH(6N)_TA／-PATCH(6N)_TAn

三相モータ制御用タイマ機能の注意事項回避

機能

- 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。
  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。
- 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、回避コードが生成されます。（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）

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。

記述規則

- コマンド行の任意の位置に指定できます。
- “－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。

記述例 1

ソース記述例)

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

-PATCH_TA 指定、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、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。

-R8C/-R8CE

生成コード制御

機能

・ R8C ファミリに対応したコードを生成します。

オプション	アドレス空間
-R8C	0H 番地 ～ 0FFFFH 番地
-R8CE	0H 番地 ～ 0FFFFFH 番地

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。
・“-M60”、“-M61”、“-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。

記述規則

・ コマンド行の任意の位置に指定できます。
・ 本オプションは大文字で指定してください。

記述例

```
>as30 –R8C sample
```

-R8Cxx

生成コード制御

機能

- 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。）
- 本オプション指定時、環境変数“LIB30” に登録されている“r8ctiny.txt”を参照します。
- 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。
- 本オプション指定時、オプション“-R8C”が付加されます。

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。
- “-M60”、“-M61”、“-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。

記述規則

- コマンド行の任意の位置に指定できます。
- 本オプションは大文字で指定してください。
- -R8C(グループ名)　のように指定してください。

注意事項

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。

記述例

```
>as30 –R8C14 sample
```

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)

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

-S

ローカルシンボル情報の出力指定

機能

- ローカルシンボル情報をリロケータブルモジュールファイルに出力します。
- 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リロケータブルモジュールファイルに出力します。
- ローカルシンボル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。

注意事項
シンボリックデバッグ可能なシンボル及びラベルの情報については、ln30 が出力するマップファイル(.map)で確認できます。

記述規則

- システムラベル情報とローカルラベル情報を同時に出力するには、"-SM"と入力してください。
- コマンド行の任意の位置に指定できます。
- 本オプションは、環境変数"AS30COM"に設定できます。設定方法は、「AS30COM の使用例」を参照してください。

記述例

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

(1) 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.r30 に出力します。
(2) sample.a30 のローカルシンボル情報を sample.r30 に出力します。

-T

アセンブラエラータグファイル生成

機能

- アセンブラエラーが発生した場合に、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。
- エラーを発生した場合は、リロケータブルモジュールファイルは生成しません。ワーニングのみの発生の場合は、リロケータブルモジュールファイルを生成します。
- エラータグファイル名は、アセンブリ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。

記述規則

- コマンド行の任意の位置に指定できます。
- 本オプションは、環境変数"AS30COM"に設定できます。設定方法は、「AS30COM の使用例」を参照してください。

記述例

エラーが発生した場合、"sample.atg"ファイルを生成します。

```
>as30 -T sample
```

-V

バージョン表示

機能

- 本オプションは、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。

注意事項
本オプションを指定した場合は、コマンド行の他のパラメータは全て無視されます。

記述規則

- 本オプションのみを指定してください。

記述例

```
>as30 -V
```

-X

外部プログラムを起動

機能

- 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。
- 本オプションを指定した場合、'-T'の指定の有無に関わらずエラーが発生したときは、アセンブラエラータグファイルを生成します。

記述規則

- -X（プログラム名）のように入力してください。
- 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。
- コマンド行の任意の位置に指定できます。

記述例

```
>as30 -Xedit samp
```

as30 エラーメッセージ一覧

'#' 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。
')' 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。
',' 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。
'.B' or '.W' is not specified
？　.B 又は.W の指定がありません。
！　.B 又は.W は省略できません。ニーモニックに.B 又は.W を記述してください。
'.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。
'.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。
'.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。
'.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。
'.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。
'.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。
'.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。
'.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。
'.MACRO' or '.MREPEAT' is missing for '.EXITM'
？　.EXITM に対する .MACRO 又は .MREPEAT がありません。
！　.EXITM の記述位置を確認してください。
'.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。
'.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。
'.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。

'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。
'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 又は SWITCH 文内に記述してください。
'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。
'CONTINUE' is missing for 'FOR' or 'DO'
？　CONTINUE の使用箇所が不適当です。
！　CONTINUE は FOR 又は DO 文内に記述してください。
'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。
'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。
']' 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。
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。
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。
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'TMP30'にディレクトリを指定してください。
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。
Can't open '.ASSERT' message file 'xxxx'
？　.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。
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。
Can't open include file 'xxxx'
？　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。
Can't read file 'filename'
？　'filename'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。
Can't write '.ASSERT' message file 'xxxx'
？　.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。
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。

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。

Characters exist in expression
？ 命令又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。

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。

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。

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。

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。

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。

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。

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。

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。

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。

Error occurred in executing 'xxx'
？ xxx の実行でエラーが発生しました。
！ xxx を実行し直してください。

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。

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。

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。

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。

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。

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。
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。。
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。
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。
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。
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。
Invalid directive command 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。
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。
Invalid mnemonic which isn’t supported in ‘ - R8C’
？ R8C ファミリでサポートされていない命令が記述されています。
！ 命令を記述し直してください。
Invalid operand(s) exist in instruction
？ 命令に無効なオペランドがあります。
！ 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。
Invalid option 'xx' is in environment data
？ 無効なコマンドオプション xx 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。
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。
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。
Invalid option 'xx' is used
？ 無効なコマンドオプション xx を使用しています。
！ 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。
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。
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。
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。
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。

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。
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。
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。
No '.ENDIF' statement
？　　.ENDIF 記述がありません。
！　　.ENDIF の記述位置を確認してください。.ENDIF を記述してください。
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。
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。
No '.FB' statement
？　　.FB の記述がありません。
！　　8 ビット変位 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。
No '.SB' statement
？　　.SB の記述がありません。
！　　8 ビット変位 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。
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。
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。ニーモニック又はオペランドの記述に誤りがないか確認してください。
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。
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。
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。
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。又はメモリを増設してください。
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。
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。
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。

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。
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。
Option 'xx' is not appropriate
？ コマンドオプション xx の記述が正しくありません。
！ コマンドオプションを指定し直してください。
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。
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。
Reserved word is missing
？ 予約語の記述がありません。
！ [SB],[FB],[A1],[A0],[SP]又は[A1A0]を記述してください。
Reserved word is used as label or symbol
？ 予約語をラベル又はシンボルに用いています。
！ ラベル又はシンボル名を記述し直してください。
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。
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。
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。
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又は 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。
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。
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。
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。
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。
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。
Size or format specifier is not appropriate
？ サイズ指定子又はフォーマット指定子の記述に間違いがあります。
！ サイズ指定子又はフォーマット指定子を記述し直してください。

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。
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。
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。
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。
Specifies option that can't use with 'xx'
？　'xx' と同時に指定することができないオプションが指定されています。
！　オプションを指定し直してください。
Statement not preceded by 'CASE' or 'DEFAULT'
？　SWITCH 文において CASE 又は DEFAULT より先に命令行があります。
！　命令行は必ず CASE 及び DEFAULT 文の後に記述してください。
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。
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。
！　シンボル定義及びシンボル名を確認してください。
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。
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"で同一のシ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。
Symbol has 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。
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。
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。
Symbol is undefined
？　シンボルが未定義です。
！　未定義のシンボル名は使用できません。前方参照となるシンボル名は記述できません。シンボル名を確認してください。
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。
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。
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、シンボル名又はラベル名を記述してください。

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。
Too many nesting lebel 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。
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。
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。
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。
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。
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。
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。
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。
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。

as30 ワーニングメッセージ一覧

'-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。
'.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。
'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。
'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。
'.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 を無視して処理します。

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。
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。
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。
Fixed data in 'CODE' section
？ CODE セクション内に指示命令、.BYTE, .WORD, .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, .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。
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。
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。
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。
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。
Mnemonic is '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。
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。
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。
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。
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。
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。

ln30 の操作方法

コマンドパラメータ

ln30 のコマンドパラメータ一覧を次に示します。

オプション名	機能
-.	画面へのメッセージ出力を停止する
-E	アブソリュートモジュールの開始アドレスを指定する
-G	ソースデバッグ情報をアブソリュートモジュールファイルに出力する
-JOPT	グローバルラベルを参照している分岐命令を最適化する
-L	参照するライブラリファイル名を指定する
-LD	ライブラリファイルの検索ディレクトリを指定する
-LOC	セクションデータを指定アドレスから配置する
-M	マップファイルを生成する
-M60	M16/60 グループに対応したコードを生成する
-M61	M16/61 グループに対応したコードを生成する
-MS	シンボル情報を含むマップファイルを生成する
-MSL	16 文字を越えるシンボルをそのままマップファイルに出力する
-NOSTOP	発生したエラー全てを画面に出力する
-O	アブソリュートモジュールファイル名を指定する
-ORDER	セクションの配置順序及び配置アドレスを指定する
-R8C	R8C ファミリ（アドレス空間 0～0FFFFH）に対応したコードを生成する
-R8CE	R8C ファミリ（アドレス空間 0～0FFFFFH）に対応したコードを生成する
-T	リンクエラータグファイルを生成する
-U	未使用関数名の検出を実施する
-V	リンケージエディタのバージョン番号を表示する
-VECT	可変ベクタテーブルの空き領域に値を設定する
-VECTN	ソフトウエア割り込み番号のアドレス値を設定する
-W	ワーニング発生時、アブソリュートモジュールファイルを生成しない
@file	コマンドファイルの記述内容に従ってリンケージエディタを実行する

コマンドパラメータの指定規則

ln30 のコマンドパラメータは次の規則に従って指定してください。

- リロケータブルモジュールファイル名とコマンドオプションの指定順序は任意です。
  >ln30　（コマンドオプション）　（リロケータブルモジュールファイル名）
  >ln30　（リロケータブルモジュールファイル名）　（コマンドオプション）

リロケータブルモジュールファイル名（必須）

- 必ず、一つ以上のリロケータブルモジュールファイルを指定してください。
- ファイル名にはパスが指定できます。
- 複数のリロケータブルファイルを指定する場合は、ファイル名の間に、必ずスペースかタブを挿入してください。

アブソリュートモジュールファイル名

- 通常 ln30 は、リロケータブルモジュールファイル名のうち一番目に指定されたファイル名をアブソリュートモジュールファイル名として生成します。
- アブソリュートモジュールファイル名を指定する場合は、コマンドオプション(-O)で指定してください。

ライブラリファイル名

- 参照するライブラリファイルを指定する場合は、コマンドオプション(-L)で指定してください。ファイル名には、パスが指定できます。
- ライブラリファイルは、環境変数(LIB30)が設定されていれば、そのディレクトリから検索し、該当するファイルが無い場合は、カレントディレクトリを検索します。または、コマンドオプション(-LD)で指定されたディレクトリから検索し、該当するファイルが無い場合は、カレントディレクトリを検索します。

コマンドオプション

- コマンドオプションを指定する場合は、コマンドオプションとその他の指定の間には、必ずスペースかタブを挿入してください。

アドレス指定

- ln30 は、セクション単位で絶対アドレスを決定し、アブソリュートモジュールファイルを生成します。
- ln30 を起動する際に、コマンド行からセクションの開始アドレスを指定することができます。
- アドレス値を指定する際には、16 進数で指定してください。なお、数値の先頭がアルファベット文字になる場合は、先頭に 0 を付けて指定してください。

例）

```
7fff
64
0a57
```

コマンドファイル

ln30 は、コマンドパラメータをファイルに記述し、そのファイルを読み込んでプログラムを実行できます。

コマンドファイル名の指定方法

- コマンドファイル名の先頭には、@を付けて指定してください。
- コマンドファイル名には、ディレクトリパスを指定できます。
- 指定したディレクトリパスにファイルが存在しない場合は、エラーとなります。

例）

```
>ln30 @cmdfile
```

コマンドファイルの記述方法

コマンドファイルの記述方法は、「入出力ファイル」の「コマンドファイルフォーマット」を参照してください。

ln30 コマンドオプション

以降に、コマンドオプションの指定方法を説明します。

－.

画面へのメッセージ出力を停止

機能

- ln30 が処理を行う際のメッセージを出力しません。
- エラーメッセージ及びワーニングメッセージは出力されます。

記述規則

- コマンド行の任意の位置に指定できます。

記述例

```
>ln30 -. smaple1 sample2
```

－E

プログラム開始アドレス指定

機能

- エントリーアドレスを設定します。エントリーアドレスは、デバッガにスタートアドレスを示すためのアドレスです。
- アドレス値の指定には数値又はラベル名が記述できます。ただし、ローカルラベル名は指定できません。

記述規則

- -E　（数値又はラベル名）のように入力してください。
- 本オプションと数値又はラベル名の間には、必ずスペースを入力してください。
- 数値は必ず 16 進数で指定してください。
- 数値の先頭が英文字（a,b,c,d,e,f）になる場合は、必ず先頭に 0 を記述してください。
- コマンド行の任意の位置に記述できます。

記述例

```
>ln30 sample1 sample2 -E num          …(1)
>ln30 sample1 sample2 -E 0f0000     …(2)
```

(1) "sample1.x30"のエントリーアドレスに、グローバルラベル"num"が持っているアドレス値を指定

(2) "sample1.x30"のエントリーアドレスに、f0000 を指定

-G

ソースデバッグ情報出力

機能

- C 言語やマクロ記述のソース行情報をアブソリュートモジュールファイルに出力します。
- 本オプションを指定しないで生成したアブソリュートモジュールファイルでは、ソース行レベルでのデバッグはできません。

注意事項

as30 実行時に行情報の出力停止オプション（-N）を指定して生成したリロケータブルモジュールファイルをリンクした場合は、本オプション(-G)を指定してもソース行レベルでのデバッグはできません。

- ソースデバッグ情報をアブソリュートモジュールファイルに出力します。

記述規則

- コマンド行の任意の位置に指定できます。

記述例

```
>ln30 -G sample1 sample2
```

-JOPT

外部分岐の最適化

機能

- グローバルラベルを参照している分岐命令(JMP,JSR)を最適化します。

注意事項

- 本オプションを指定する場合、nc30 の”-OGJ(-Oglb_jmp)”および as30 の”-JOPT”オプションを指定してください。
- 本オプションを指定した場合、分岐情報ファイル（拡張子.jin）が生成されます。分岐情報ファイルは編集しないでください。また、拡張子.jin は使用しないでください。
- 本オプションと同時に、指示命令”.OPTJ”が使用されている場合は、本オプションが有効になります。
- 本オプションを指定する場合は、nc30、as30 および ln30 を同一ディレクトリで起動してください。

記述規則

- コマンド行の任意の位置に指定できます。

記述例

```
>ln30 -JOPT sample1 sample2
```

-L

ライブラリファイル名指定

機能

- リンク実行時に参照するライブラリファイル名を指定します。
- ln30 は、指定したライブラリファイル内から、グローバルシンボル情報を読み込んで、必要なリロケータブルモジュールをリンクします。

記述規則

- -L　（ライブラリファイル名）のように入力してください。
- 本オプションとファイル名の間には、必ずスペースを入力してください。
- コマンド行の任意の位置に指定できます。
- ライブラリファイル名にはパスが指定できます。
- ライブラリファイルは複数個指定できます。ライブラリファイルを複数指定する場合は、ファイル名をカンマで区切って指定してください。このとき、カンマの前後にスペース又はタブは記述できません。

記述例

```
>ln30 sample1 sample2 -L lib1              …(1)
>ln30 sample1 sample2 -L work¥lib1     …(2)
>ln30 sample1 sample2 -L lib1,lib2     …(3)
```

(1) カレントディレクトリ又は環境変数(LIB30)で指定されているディレクトリ内の lib1.lib ファイルを指定
(2) カレントディレクトリの下の work ディレクトリ内の lib1.lib ファイルを指定
(3) カレントディレクトリ又は環境変数(LIB30)で指定されているディレクトリの lib1.lib 及び lib2.lib ファイルを指定

-LD

ライブラリファイルのディレクトリ指定

機能

- ライブラリファイルを参照するディレクトリ名を指定します。
- 本オプションを指定した場合も、ライブラリファイル名は指定してください。
- 本オプションで指定したディレクトリ名は、次に本オプションで指定し直すまで有効です。
- ライブラリファイル名にパスを指定した場合は、本オプションで指定したディレクトリに、ライブラリファイルパスを連結したディレクトリのライブラリファイルが処理対象となります。

記述規則

- -LD　（ディレクトリ名）のように入力してください。
- 本オプションとディレクトリ名の間には、必ずスペースを入力してください。
- コマンド行の任意の位置に指定できます。

記述例

```
>ln30 sample1 sample2 -LD ¥work¥lib -L lib1　…(1)
>ln30 sample1 sample2 -LD ¥work¥lib -L lib1 -LD ¥work¥tmp -L lib2 (2)
>ln30 sample1 -LD ¥work -L lib¥lib1　…(3)
```

(1) ¥work¥lib¥lib1 ファイルを参照
(2) ¥work¥lib¥lib1,¥work¥tmp¥lib2 ファイルを参照
(3) ¥work¥lib¥lib1 ファイルを参照

-LOC

セクションデータ配置指定

機能

- 指定されたセクション内のデータを格納するアドレスの指定を行います。
- 指定されたセクション内のシンボル値（アドレス）はソースファイルに記述された、.ORG によるアドレス、またはリンカの -ORDER オプションで指定されたアドレスを基準に生成されます。
- プログラムを RAM 上で動作させるようなアプリケーションに利用できます。

注意事項

1. ALIGN の指定により再配置されたプログラムが正常に動作しない場合があります。そのため、"-LOC"オプションを指定する場合、セクションの先頭アドレスが偶数番地のセクションは偶数番地に、先頭アドレスが奇数番地のセクションは奇数番地に再配置先を指定してください。
2. “-LOC”オプションは指定されたセクションの登録アドレスを指定するものであり、実行時のアドレス領域に転送する機能はありません。

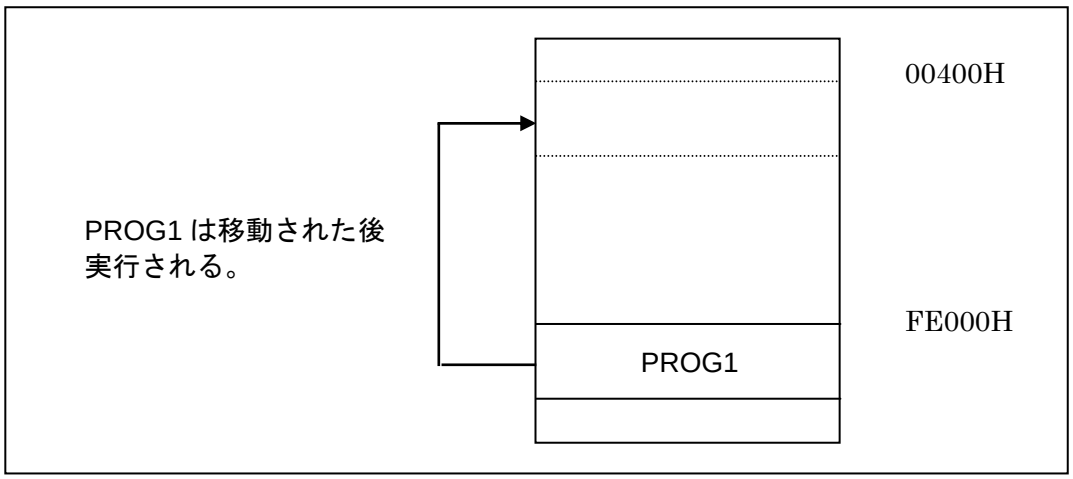
記述規則

- オプションとセクション名はスペースで区切ってください。
- ＝ の前後にはスペースやタブを記述できません。
- アドレスは必ず指定してください。
- 複数セクション配置を指定する場合はカンマ（,）で区切って記述してください。

記述例

本記述例では、下図のように FE000h 番地の ROM に格納されているセクション名 PROG1 を 00400h 番地の RAM へ転送して RAM 上(00400h 番地)で実行する場合を示します。
この場合の ln30 への指定の記述は以下のようになります。

```
>ln30 -ORDER PROG1=00400 -LOC PROG1=0FE000
```



-M

マップファイル生成

機能

- アドレスマッピング情報を格納したマップファイルを生成します。
- 生成するマップファイルは、アブソリュートモジュールファイルの拡張子を".map"に変更したファイル名になります。
- 本オプションを指定している場合は、セクションオーバーラップエラーが発生してもマップファイルを生成します。

記述規則

- コマンド行の任意の位置に指定できます。

記述例

sample1.x30 と sample1.map ファイルを生成

```
>ln30 -M sample1 sample2
```

-M60/-M61

生成コード制御

機能

- 各オプションを付加した場合のアセンブラの処理は次のようになります。

オプション名	処理内容
-M60	リンク時に 64Kbytes のバンクをまたがる分岐命令がある場合にワーニングを出力します。ワーニングが出力された場合は指示命令'.SJMP'で対応してください。
-M61	リンク時に 64Kbytes のバンクをまたがる分岐命令がある場合にワーニングを出力します。ワーニングが出力された場合は指示命令'.SJMP'で対応してください。

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。

記述規則

- コマンド行の任意の位置に指定できます。

記述例

```
>as30 -M60 sample
```

-MS/-MSL

マップファイルにシンボル情報出力

機能

- アドレスマッピング情報及びシンボル情報を格納したマップファイルを生成します。
- -MS を指定した場合、16 文字を越えるシンボルは 16 文字までしかマップファイルに出力されません。
- -MSL を指定した場合、16 文字を越えるシンボルをそのまま出力します。
- 生成するマップファイルは、アブソリュートモジュールファイルの拡張子を".map"に変更したファイル名になります。
- 本オプションを指定している場合は、セクションオーバーラップワーニングが発生してもマップファイルを生成します。

記述規則

- コマンド行の任意の位置に指定できます。

記述例

sample1.x30 と sample1.map ファイルを生成

```
>ln30 sample1 sample2 -MS
```

-NOSTOP

全エラー出力指定

機能

- 発生したリンクエラーを全て画面に出力します。
- 本指示命令を指定しない場合は、最大 20 個までのエラーを画面に出力します。

記述規則

- コマンド行の任意の位置に指定できます。

記述例

```
>ln30 sample1 sample2 –NOSTOP
```

-O

アブソリュートモジュールファイル名指定

機能

- ln30 が生成するアブソリュートモジュールファイル名を任意の名前に設定できます。
- 本オプションで、アブソリュートモジュールファイル名を指定しない場合は、コマンド行で一番目に指定されている、リロケータブルモジュールファイル名の拡張子を".x30"としたものをアブソリュートモジュールファイル名とします。

記述規則

- -O　（ファイル名）のように入力してください。
- オプションとファイル名の間には、必ずスペースを入力してください。
- ファイル名の拡張子は省略できます。省略した場合の拡張子は、".x30"となります。
- ファイル名には、パスを指定できます。

記述例

```
>ln30 sample1 sample2 -O abssmp                    …(1)
>ln30 -O ¥work¥absfile¥abssmp sample1 sample2 …(2)
```

(1)　"abssmp.x30"ファイル名を指定
(2)　ディレクトリ"¥work¥absfile"に"abssmp.x30"を生成することを指定

-ORDER

セクションアドレス及び再配置順序指定

機能

・ セクションの配置順序と、セクションの開始アドレスを指定します。

注意事項

絶対セクションに対して、開始アドレスを指定した場合はエラーとなります。

・ 開始アドレスを指定しない場合は、０からアドレスを配置します。
・ 同一名のセクションが指定したリロケータブルファイルに存在するときは、指定したファイル順にセクションを配置します。このとき、相対属性を持つセクションの後に絶対属性を持つものが配置されるとエラーとなります。
・ 存在しないセクション名を指定した場合は、そのセクション名を無視します。

記述規則

・ -ORDER （セクション名）,（セクション名）又は-ORDER （セクション名）＝（スタートアドレス）のように入力してください。
・ オプションとセクション名の間には、必ずスペースを入力してください。
・ セクション名とセクション名又はアドレス値とセクション名は、カンマで区切って指定してください。このとき、カンマの前後にスペース又はタブは入力できません。
・ コマンド行の任意の位置に指定できます。

記述例

```
>ln30 sample1 sample2 -ORDER main,sub,dat              …(1)
>ln30 sample1 sample2 -ORDER main=0f000,sub,dat …(2)
```

(1) main,sub,dat の順にアドレス 0H からセクションを配置
(2) main,sub,dat の順にアドレス 0f000H からセクションを配置

存在しないセクション名(noprog)を指定した場合

```
>ln30 sample1 sample2 -ORDER main=0f000,noprog,dat             …(3)
>ln30 sample1 sample2 -ORDER main=0f000,noprog=0f2000,dat …(4)
>ln30 sample1 sample2 -ORDER main=0f000,noprog=0f2000,dat=0f3000 …(5)
```

(3) main,dat の順にアドレス 0f000H からセクションを配置
(4) main は 0f000H から、dat は 0f2000H から配置
(5) main は 0f000H から、dat は 0f3000H から配置

-R8C/-R8CE

生成コード制御

機能

・ R8C ファミリシリーズに対応したコードを生成します。

オプション	アドレス空間
-R8C	0H 番地 ～ 0FFFFH 番地
-R8CE	0H 番地 ～ 0FFFFFH 番地

注意事項
“-M60”または“-M61”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、オプション機能選択レジスタへ設定される値がメッセージ“The value of option function select register is xxH”として出力されます。

記述規則

・ コマンド行の任意の位置に指定できます。
・ 本オプションは大文字で指定してください。

記述例

```
>ln30 –R8C sample
```

-T

リンクエラータグファイル生成

機能

・ リンクエラーが発生した場合、リンク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発生しなければ、ファイルは生成しません。
・ エラータグファイル名は、先頭に指定したリロケータブルモジュールファイル名の拡張子を".ltg"に変更したものになります。コマンドオプション"-O"でアブソリュートモジュールファイル名を指定している場合は、指定されたファイル名の拡張子を".ltg"に変更したものがエラータグファイル名になります。
・ リンクエラーの発生場所は、アセンブリソース行番号で出力されます。

記述規則

・ コマンド行の任意の位置に指定できます。

記述例

エラーが発生した場合、"sample1.ltg"ファイルを生成

```
>ln30 sample1 sample2 -T
```

-U

未使用関数名の検出

機能

・ C 言語ソースファイルで記述された未使用関数名に対してワーニングを出力します。

注意事項

1. 未使用関数の削除は、本当に必要でない関数であるか確認した上で行ってください。
2. 本オプションは NC30 および AS30 の"-finfo"オプションが指定された場合に有効となります。
3. C 言語プログラムからアセンブラ関数を呼び出す場合、アセンブラ関数に対してアセンブラ指示命令".INSF"および".EINSF"を必ず記述してください。
4. 割り込み関数をアセンブラ関数で記述した場合、未使用関数としてワーニングが出力されます。(アセンブラ指示命令".INSF"および".EINSF"が記述されている場合)
5. 以下に示す名称は未使用関数の検出対象外となります。

アセンブラ関数名：start
C 言語の関数名　：main、標準ライブラリ関数(ランタイムライブラリ関数)

記述規則

・ コマンド行の任意の位置に指定できます。

記述例

```
>ln30 -U sample
```

-V

バージョン表示

機能

・ ln30 のバージョン番号を表示します。

注意事項

本オプションを指定した場合は、コマンド行の他のパラメータは全て無視されます。

記述規則

・ 本オプションのみを記述してください。

記述例

```
>ln30 -V
```

-VECT

可変ベクタテーブルの空き領域に値を設定

機能

・ 可変ベクタテーブルの自動生成を行った際の空き領域（指示命令“.RVECTOR”で指定されていないソフトウエア割り込み）に対して値を設定します。
空き領域に設定する値には数値またはグローバルラベル名が指定できます。なお、設定する値は 4 バイトのデータとして扱われます。

注意事項

・ 本オプションを指定した場合、可変ベクタテーブルが自動生成されます。
可変ベクタテーブルの自動設定については、アセンブラ指示命令“.RVECTOR”を参照してください。
・ “-VECTN”オプションが指定されている場合は、“-VECTN”オプションが優先されます。

記述規則

・ -VECT　（数値又はラベル名）のように入力してください。
・ 本オプションと数値又はラベル名の間には、必ずスペースを入力してください。
・ 数値は必ず 16 進数で指定してください。
・ 数値の先頭が英文字（a,b,c,d,e,f）になる場合は、必ず先頭に 0 を記述してください。
・ コマンド行の任意の位置に記述できます。

記述例

```
>ln30 –VECT stop sample1 sample2　…(1)
>ln30 sample1 sample2　　…(2)
```

(1) 可変ベクタテーブルの空き領域に対して、“stop”が持つアドレス値を設定します。
(2) 可変ベクタテーブルの空き領域に対して、“__dummy_int”が持つアドレス値を設定します。

-VECTN

ソフトウエア割り込み番号のアドレス値の設定

機能

・　指定されたソフトウエア割り込み番号のアドレスを設定します。

注意事項

・ 本オプションを指定した場合、可変ベクタテーブルが自動生成されます。
可変ベクタテーブルの自動設定については、アセンブラ指示命令“.RVECTOR”を参照してください。
・ 指示命令“.RVECTOR”で指定したソフトウエア割り込み番号を本オプションで指定するとエラーになります。
・ “-VECT”オプションより本オプションが優先されます。

記述規則

・ -VECTN　（数値又はシンボル名），ソフトウエア割り込み番号 のように入力してください。
・ ソフトウエア割り込み番号を複数指定する場合は、
（数値又はシンボル名）,ソフトウエア割り込み番号,（数値又はシンボル名）,ソフトウエア割り込み番号
のように指定してください。
・ 本オプションと数値又はラベル名の間には、必ずスペースを入力してください。
・ 数値は必ず 16 進数で指定してください。
・ 数値の先頭が英文字（a,b,c,d,e,f）になる場合は、必ず先頭に 0 を記述してください。
・ 数値またはソフトウエア割り込み番号は、必ずカンマで区切ってください。
・ カンマの前後にスペースまたはタブは入力できません。
・ ソフトウエア割り込み番号は、必ず 10 進数で入力してください。
・ コマンド行の任意の位置に記述できます。

記述例

```
>ln30 –VECTN stop,20 sample1 sample2　…(1)
>ln30  –VECTN stop0,20,stop1,21 sample1 sample2　　…(2)
```

(1)　ソフトウエア割り込み番号の 20 番に対して、“stop”が持つアドレス値を設定します。
(2)　ソフトウエア割り込み番号の 20 番に対して“stop0”、21 番に対して“stop1”が持つアドレス値を設定します。

-W

ワーニング発生時、アブソリュートモジュールファイルを生成しない

機能

- ワーニング発生時、アブソリュートモジュールファイルを生成しません。
- ワーニング発生時、OS の戻り値に 10 を返します。

注意事項

・本オプションの指定がない場合は、OS の戻り値に 0 を返します。

記述規則

- コマンド行の任意の位置に指定できます。

記述例

```
>ln30 –W sample1 sample2
```

@

コマンドファイル参照

機能

- 指定したファイルの内容をコマンドパラメータとして ln30 を起動します。

記述規則

- ＠（ファイル名）のように入力してください。
- 本オプションとファイル名の間にはスペースは記述できません。
- 他のパラメータはコマンド行に記述できません。

記述例

```
>ln30 @cmdfile
```

ln30 エラーメッセージ一覧

'-loc' section 'section' is multiple defined
？　-loc オプションで指定されたセクション名が二重定義です。
！　セクション名を確認してください。
'-loc' section 'section' is not found
？　-loc オプションで指定されたセクションが見つかりません。
！　セクション名を確認してください。
'-order' section 'section' is multiple defined
？　-order で指定されたセクション名が二重定義です。
！　セクションは、一度だけ定義してください。
'-order' section 'section' is not found
？　-order で指定されたセクションが見つかりません。
！　セクション名を確認して実行し直してください。
'-VECT' option parameter 'symbol' is undefined
？　-VECT で指定されたシンボル'symbol'が未定義です。
！　シンボル名を確認してください。
'-VECTN' option parameter 'symbol' is undefined
？　-VECTN で指定されたシンボル'symbol'が未定義です。
！　シンボル名を確認してください。
'CODE' section 'section-1' is overlapped on the 'section-2'
？　CODE のセクション'section-1'と'section-2'がオーバーラップしています。
！　セクションがオーバーラップしないように配置し直してください。
'ROMDATA' section 'section-1' is overlapped on the 'section-2'
？　ROMDATA タイプのセクション'section-1'と'section-2'がオーバーラップしています。
！　セクションがーオーバーラップしないように配置し直してください。
'section' is written after the same name of relocatable section
？　相対属性セクションの後に同名の絶対属性セクション'section'を連結しています。
！　絶対属性の後に相対属性を配置してください。
'symbol' is multiple defined
？　シンボル'symbol'が二重定義されています。
！　外部シンボル名を確認してください。
'symbol' value is undefined
？　シンボル'symbol'の値が未定義です。
！　値を 0 として処理します。シンボル値を確認してください。
Absolute section 'section' is relocated
？　絶対セクション'section'を再配置しています。
！　セクションの配置指定をし直してください。
Address is overlapped in 'CODE' section 'section'
？　CODE タイプのセクション'section'内でアドレスがオーバーラップしています。
！　セクションがオーバーラップしないように配置し直してください。
Address is overlapped in 'ROMDATA' section 'section'
？　ROMDATA タイプのセクション'section'内でアドレスがオーバーラップしています。
！　アドレスがオーバーラップしないように配置し直してください。
Can't close file 'file'
？　'file'ファイルがクローズできません。
！　ディレクトリ情報を確認してください。
Can't close temporary file
？　テンポラリファイルがクローズできません。
！　ディスクの残り容量を確認してください。
Can't create file 'file'
？　'file'ファイルが生成できません。
！　ディレクトリ情報を確認してください。

Can't create temporary file
？ テンポラリファイルが生成できません。
！ ディレクトリが書き込み禁止になっていないか確認してください。

Can't generate automatically the variable interrupt vector table
？ 可変ベクタテーブルを自動生成することができません。
！ 可変ベクタテーブルを自動生成する場合、”vector”セクションにプログラムを記述しないでください。

Can't generate automatically the special page vector table
？ スペシャルページベクタテーブルを自動生成することができません。
！ スペシャルページベクタテーブルを自動生成する場合、”svector”セクションにプログラムを記述しないでください。

Can't open file 'file'
？ '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。

Can't open temporary file
？ テンポラリファイルがオープンできません。
！ ディレクトリ情報を確認してください。

Can't remove file 'file'
？ 'file'ファイルが削除できません。
！ ファイルのパーミッションを確認してください。

Can't remove temporary file
？ テンポラリファイルが削除できません。
！ ファイルのパーミッションを確認してください。

Can't registered symbol in the list
？ シンボルをリストに登録できません。
！ 本エラーが発生した場合は、お手数ですが弊社までご連絡いただきますようお願いします。

Command-file line characters exceed
？ コマンドファイルの 1 行の文字数が制限を越えています。
！ コマンドファイルの内容を確認してください。

Command line is too long
？ コマンド行の文字数が多すぎます。
！ コマンドファイルを作成してください。

DEBUG Information mismatch in file
？ リロケータブルモジュールファイルのフォーマットバージョンが異なるファイルが混在しています。
！ 最新のアセンブラでアセンブルしなおしてください。

Error occured in executing 'xxx'
？ 'xxx'起動中にエラーが発生しました。
！ 'xxx'のエラーメッセージを確認してください。

Illegal file extension '.xxx' is used
？ ファイルの拡張子'.xxx'に間違いがあります。
！ ファイルの拡張子を指定し直してください。

Illegal format 'file'
？ 'file'ファイルのフォーマットに間違いがあります。
！ リロケータブルファイルが as30 で生成されたものであることを確認してください。

Illegal format 'file' :expression error occurred
？ 'file'ファイルのフォーマットに間違いがあります。
！ リロケータブルファイルが as30 で生成されたものであることを確認してください。

Illegal format 'file' :it's not jin file
？ 'file'ファイルのフォーマットに間違いがあります。分岐情報ファイルではありません。
！ 分岐情報ファイルが as30 で生成されたものであることを確認してください。

Illegal format 'file' :it's not library file
? 'file'ファイルのフォーマットに間違いがあります。ライブラリファイルではありません。
! ライブラリファイルが lb30 で生成されたものであることを確認してください。
Illegal format 'file' :it's not relocatable file
? 'file'ファイルのフォーマットに間違いがあります。リロケータブルファイルではありません。
! リロケータブルファイルが as30 で生成されたものであることを確認してください。
Interrupt number 'X' is multiple defined
? ソフトウエア割り込み番号'X'が二重定義されています。
! ソフトウエア割り込み番号を確認してください。
Interrupt number 'X' is multiple defined by '-VECTN' and '.rvector'
? ソフトウエア割り込み番号'X'が-VECTN オプションと.rvector 指示命令で二重定義されています。
! ソフトウエア割り込み番号を確認してください。
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。
MCU information mismatch in file 'file'
? 'file'ファイルの MCU 情報が一致していません。
! リロケータブルファイルが as30 で生成されたものであることを確認してください。
No input files specified
? 入力ファイルが指定されていません。
! ファイル名を指定してください。
Not enough memory
? メモリが足りません。
! メモリを増設してください。
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。
Option parameter address exceed xxx
? オプションで指定したアドレスが xxx を越えています。
! コマンドを入力し直してください。
Special page number 'X' is multiple defined
? スペシャルページ番号'X'が二重定義されています。
! スペシャルページ番号を確認してください。
Symbol type of floating point is not supported
? シンボルタイプの浮動小数点はサポートしていません。
! 本エラーが発生した場合は、お手数ですが弊社までご連絡いただきますようお願いします。
Wrong value is specified by option "-loc".
? "-loc"で指定されたアドレスに誤りがあります。
! "-loc" の注意事項を確認して記述し直してください。
Zero division exists in the expression
? リロケーションデータの演算に 0 除算があります。
! 式を記述し直してください。

ln30 ワーニングメッセージ一覧

'-e' option parameter 'symbol' is undefined
? -e で指定されたシンボル'symbol'が未定義です。
! ソースプログラム内で、'symbol'を定義してください。値を 0 として処理します。
'CODE' section 'section-1' is overlapped on the 'section-2'.
? CODE セクションの'section-1'が'section-2'にオーバーラップしています。オーバーラップして配置しました。
! セクションがオーバーラップ可能か確認してください。

'DATA' section 'section-1' is overlapped on the 'section-2'
？　DATA セクションの'section-1'が 'section-2'にオーバーラップしています。オーバーラップして配置しました。
！　セクションがオーバーラップ可能か確認してください。
'ROMDATA' section 'section-1' is overlapped on the 'section-2'.
？　ROMDATA セクションの'section-1'が 'section-2'にオーバーラップしています。オーバーラップして配置しました。
！　セクションがオーバーラップ可能か確認してください。
'label' value exceed xxx
？　ラベル'label'の値が xxx を越えています。
！　セクションの配置アドレスを確認してください。ラベルの定義を確認してください。
'section' data exceed xxx
？　セクションのデータが xxx 番地を越えています。
！　セクションの配置アドレスを確認してください。
16-bits signed value is out of range -32768 -- 32767 address='address'
？　リロケーションデータの演算結果が -32768 から ＋32767 の範囲を越えています。
！　オーバーフロー分は、無視した結果を扱います。
16-bits unsigned value is out of range 0 -- 65535 address='address'
？　リロケーションデータの演算結果が 0 から 65535 の範囲を越えています。
！　オーバーフロー分は、無視した結果を扱います。
16-bits value is out of range -32768 -- 65535 address='address'
？　リロケーションデータの演算結果が -32768 から 65535 の範囲を越えています。
！　オーバーフロー分は、無視した結果を扱います。
24-bits signed value is out of range -8388608 -- 8388607 address='address'
？　リロケーションデータの演算結果が -8388608 から 8388607 の範囲を越えています。
！　オーバーフロー分は、無視した結果を扱います。
24-bits unsigned value is out of range 0 -- 16777215 address='address'
？　リロケーションデータの演算結果が 0 から 16777215 の範囲を越えています。
！　オーバーフロー分は、無視した結果を扱います。
24-bits value is out of range -8388608 -- 16777215 address='address'
？　リロケーションデータの演算結果が -8388608 から 16777215 の範囲を越えています。
！　オーバーフロー分は、無視した結果を扱います。
4-bits signed value is out of range -8 -- 7 address='address'
？　リロケーションデータの演算結果が -8 から 7 の範囲を越えています。
！　オーバーフロー分は、無視した結果を扱います。
8-bits signed value is out of range -128 -- 127 address='address'
？　リロケーションデータの演算結果が -128 から 127 の範囲を越えています。
！　オーバーフロー分は、無視した結果を扱います。
8-bits unsigned value is out of range 0 -- 255 address='address'
？　リロケーションデータの演算結果が 0 から 255 の範囲を越えています。
！　オーバーフロー分は、無視した結果を扱います。
8-bits value is out of range -128 -- 255 address='address'
？　リロケーションデータの演算結果が -128 から 255 の範囲を越えています。
！　オーバーフロー分は、無視した結果を扱います。
Absolute-section is written after the absolute-section 'section'
？　絶対属性セクション'section'の後に同名の絶対属性を連結しています。アドレスが不連続に配置される可能性があります。
！　ソース記述を確認してください。
Absolute-section is written before the absolute-section 'section'
？　絶対属性セクション'section'の前に絶対属性を連結しています。
！　連結を実行します。ソースプログラムのアドレス指定を確認してください。
Address information mismatch in file 'file'
？　リロケータブルファイル'file'ファイルのアドレス情報が一致していません。
！　リロケータブルファイルが as30 で生成されたものであることを確認してください。

Address is overlapped in the same 'DATA' section 'section'
？ 同一名の DATA セクション'section'内でアドレスがオーバーラップしています。オーバーラップして配置しました。
！ アドレスがオーバーラップ可能か確認してください。

Directive command '.ID' is duplicated
？ 指示命令 ‘.ID’が重複指定されています。
！ ‘.ID’は 1 つのアブソリュートモジュールファイルに 1 回だけ記述できます。
余分な’.ID’を削除してください。

Directive command '.PROTECT' or '.OFSREG' is duplicated
？ 指示命令 ‘.PROTECT’ または‘.OFSREG’が重複指定されています。
！ ‘.PROTECT’ または‘.OFSREG’は 1 つのアブソリュートモジュールファイルに 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。

Global function ‘xxx’ is never used
？ グローバル関数’xxx’は一度も使用されていません。
！ 必要な関数か確認してください。

JMP.S instruction exist at end of bank (address xx)
？ ショートジャンプ命令の分岐先がバンクをまたがっています。
！ 該当するアドレスでショートジャンプ命令を生成しないように、指示命令'.SJMP'で制御してください。

Local function ‘xxx’ is never used
？ ローカル関数’xxx’は一度も使用されていません。
！ 必要な関数か確認してください。

Object format version mismatch in file 'file'
？ 'リロケータブルファイル又はライブラリファイル file'のバージョン情報が一致していません。
！ リロケータブルフィル又はライブラリファイルが AS30 プログラムによって生成されたものであることを確認してください。ファイルを生成し直してください。本エラーが発生した場合は、お手数ですが弊社までご連絡いただきますようお願いします。

Section type mismatch 'section'
？ 同一名セクション'section'のセクションタイプが異なっています。
！ ソースのセクションタイプを確認してください。

The free area's address in vector table isn't specified.
？ 可変ベクタテーブルの空き領域にアドレスが設定されていません。
！ 可変ベクタテーブルの空き領域を確認してください。

lmc30 の操作方法

lmc30 の操作方法を説明します。lmc30 の主な機能は、アブソリュートモジュールファイルから、モトローラ S フォーマットの機械語ファイルを生成します。

コマンドパラメータ

lmc30 のコマンドパラメータ一覧を次に示します。

オプション名	機能
-.	画面へのメッセージ出力を停止する
-A	出力データのアドレス範囲を指定する
-E	実行開始アドレスを設定する
-F	空き領域データを設定する
-H	インテル HEX フォーマットの機械語ファイルを生成する
-ID	ID コードチェック機能の ID コードを設定する
-L	データレコード長を選択する
-O	出力ファイル名を指定する
-ofsregx	オプション機能選択レジスタに値を設定する
-R8C	R8C ファミリ(アドレス空間 0～0FFFFH)に対応したコードを生成する
-R8CE	R8C ファミリ(アドレス空間 0～0FFFFFH)に対応したコードを生成する
-V	ロードモジュールコンバータのバージョン番号を表示する
-protect1	ROM コードプロテクト機能のレベル 1 を設定する
-protectx	ROM コードプロテクト制御番地に値を設定する

コマンドパラメータの指定規則

lmc30 のコマンドパラメータは次の規則に従って指定してください。

コマンドパラメータの指定順序

1　コマンドオプション
2　アブソリュートモジュールファイル名（必須）
　>lmc30　（コマンドオプション）　（アブソリュートモジュールファイル名）

アブソリュートモジュールファイル名（必須）

- ln30 が生成したアブソリュートモジュールファイルを指定してください。
- アブソリュートモジュールファイル名は一つだけ指定してください。
- ファイルの拡張子(.x30)は省略できます。
- 拡張子が".x30"以外のファイル名は指定できません。

コマンドオプション

- コマンドオプションは必要に応じて指定してください。
- 複数のコマンドオプションを指定することができます。
- 複数のコマンドオプションを指定する場合のコマンドオプションの指定順序は任意です。

lmc30 コマンドオプション

lmc30 のコマンドオプションの指定方法について詳しく説明します。

-.

画面へのメッセージ出力を停止

機能

- lmc30 が処理を行う際のメッセージを出力しません。
- エラーメッセージ及びワーニングメッセージは出力されます。

記述規則

- 必ず、ファイル名の前に本オプションを指定してください。

記述例

```
>lmc30 -. debug
```

-A

出力データのアドレス範囲指定

機能

- 生成するファイルに出力する機械語データのアドレス範囲を指定します。
- 本オプションは以下の２種類の指定が可能です。
  1. 出力の開始アドレスと終了アドレスを指定
  2. 出力の開始アドレスのみ指定

記述規則

- “-A　（開始アドレス値：終了アドレス値）”または“-A （開始アドレス値）”のように指定してください。
- 本オプションと開始アドレス値の間には、必ずスペースを指定してください。
- アドレス値は、必ず 16 進数値で指定してください。
- ファイル名を指定する前に本オプションを指定してください。
- 出力の開始アドレス値のみ指定した場合、アブソリュートモジュールファイルに登録されているデータの最大アドレスが終了アドレス値となります。
- 開始アドレス値が終了アドレス値よりも大きい場合はエラーとなります。
- 指定されたアドレス範囲内でデータが存在しないアドレスに対しては、空き領域データ設定オプション'－F'が指定されている場合は指定されたデータを出力し、指定されていない場合は何も出力されません。
- 開始アドレス値がアブソリュートモジュールファイルに登録されているデータの最大アドレスよりも大きい場合はエラーとなります。
- 終了アドレス値がアブソリュートモジュールファイルに登録されているデータの最小アドレスよりも小さい場合はエラーとなります。
- 開始アドレス値と終了アドレス値が同一の場合はエラーとなります。

記述例

```
> lmc30 -A 1000:11FF　sample
```

アドレス範囲指定の開始アドレス値を 1000H、終了アドレス値を 11FFH とします。

```
> lmc30 -A 1000　sample
```

アドレス範囲指定の開始アドレス値を 1000H、終了アドレスアブソリュートモジュールファイル sample に登録されているデータの最大アドレス値となります。

-E

実行開始アドレスの設定

機能

- 実行開始アドレスを設定します。
- 設定したアドレスをモトローラ S フォーマットファイルに出力します。

記述規則

- -E　（アドレス値）のように入力してください。
- 本オプションと数値の間には、必ずスペースを入力してください。
- アドレス値は、必ず 16 進数で設定してください。
- アドレス値の先頭の値が、'a'～'f'の場合は先頭に必ず'0'を付けてください。

注意事項

本オプションは、"-H"と同時に指定できません。

記述例

```
>lmc30 -E 0f0000 debug    …(1)
>lmc30 -E 8000 debug      …(2)
```

(1)　実行開始アドレスを 0f000H に指定

(2)　実行開始アドレスを 8000H に指定

-F

空き領域データの設定

機能

- 指定されたアブソリュートモジュールファイル内のデータ登録されていないアドレスに対して任意データを出力します。
- 本オプションは以下の 3 種類の指定が可能です。
  1. 空き領域に設定するデータ値のみ指定
  2. 空き領域に設定するデータ値とその設定開始アドレス値を設定
  3. 空き領域に設定するデータ値とその設定開始アドレス値および終了アドレス値を設定

記述規則

- “－F　(空き領域設定データ値)”、“－F　(空き領域設定データ値：開始アドレス値)”または“－F　(空き領域設定データ値：開始アドレス値：終了アドレス値)”のように指定してください。
- 本オプションとデータの間には、必ずスペースを指定してください。
- アドレス値は、必ず１６進数値で指定してください。
- ファイル名を指定する前に本オプションを指定してください。
- 開始アドレス値が終了アドレス値よりも大きい場合はエラーとなります。
- 本オプションと”－A”オプションと併用する場合､空き領域設定データの出力範囲は”－A”オプションで指定したアドレス範囲内になければエラーとなります。
- 開始アドレス値がアブソリュートモジュールファイルに登録されているデータの最大アドレス値よりも大きい場合、開始および終了アドレス値ともに指定されている場合に限り指定アドレス範囲に空き領域設定データが追加出力されます。開始アドレス値のみ指定されている場合はエラーとなります。
- 終了アドレス値がアブソリュートモジュールファイルに登録されているデータの最小アドレス値よりも小さい場合、指定アドレス範囲に空き領域設定データが追加出力されます。
- 開始アドレス値と終了アドレス値が同一の場合はエラーとなります。

記述例

```
>lmc30 –A 1000:11FF -F 00 sample
```

アドレス範囲指定の開始アドレス値を 1000H、終了アドレス値を 11FFH として、その範囲内の空き領域に 00H を出力します。

```
>lmc30 –A 1000:11FF -F 00:1000:10FF sample
```

アドレス範囲指定の開始アドレス値を 1000H、終了アドレスを 11FFH として、その範囲内のうち 1000H から 10FFH 内の空き領域に 00H を出力します。

```
>lmc30 -F 00:1000:11FF sample
```

1000H から 11FFH 内の空き領域に 00H を出力します。アブソリュートモジュールファイル sample に登録されているデータ領域がこの 1000H から 11FFH 内に存在しないな場合、sample に登録されているデータおよび 00H を 1000H から 11FFH に出力します。

```
>lmc30 -F 00:1000 sample
```

1000H からアブソリュートモジュールファイル sample に登録されているデータ最終アドレス値内の空き領域に 00H を出力します。アブソリュートモジュールファイル sample に登録されているデータ最終アドレス値がこの 1000H よりも小さい場合はエラーとなります。

-H

インテル HEX 形式に変換

機能

- インテル HEX フォーマットの機械語ファイルを生成します。
- プログラムのアドレスが 100000H を超える場合は、専用 HEX フォーマットファイルを生成します。

記述規則

- ファイル名を指定する前に本オプションをしてしてください。
- 本オプションは、"-E"オプションと同時に設定できません。

記述例

```
>lmc30 -H debug
```

-ID

ID コードチェック機能の ID コードを設定

- 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。
- 指定した ID コードは、ID コード格納番地に各 8 ビットデータとして格納されます。また、ROM コードプロテクト制御番地またはオプション機能選択レジスタに FF を書き込みます。

オプション	ID コード格納番地
-R8C 指定なし	FFFDF、FFFE3、FFFEB、FFFEF、FFFF3、FFFF7、FFFFB
-R8C 指定あり	FFDF、FFE3、FFEB、FFEF、FFF3、FFF7、FFFB

- ROM コードプロテクト機能またはオプション機能選択レジスタに関するオプション（-protect1、-protectx、-ofsregx）を指定した場合、プロテクト制御番地またはレジスタには以下の値が書き込まれます。

-ID	-protect1, -protect, -ofsregx	値
指定	指定	オプション指定値
指定	指定なし	FF
指定なし	指定	オプション指定値
指定なし	指定なし	ソースプログラムに記述されたデータ

- ソースプログラムで ID コード格納番地にデータを書き込んでいる場合でも、このオプションを指定すると ID コード格納番地の値は必ず書き換えられます。オプションを指定しないとソースプログラムで記述した値が出力されます。
- オプションのみを指定した場合は、ID コードを FFFFFFFFFFFFFF として処理します。（例 6）
- 本オプションで設定された ID コードを示した ID ファイル（拡張子.id）を生成します。
- 指定した ID コードはアスキーコードで格納されます。

注意事項
アセンブラ指示命令“.ID”、“.OFSREG”または“.PROTECT”が記述されている場合、本オプションは無効となります。

記述規則

- 本コマンドオプションは必ず大文字で指定してください。
- -ID（コードプロテクト値 ）のように入力してください。
- ID コードは“-ID”に続けて指定してください。
- ID コードを直接指定する場合は"-ID#"に続けて数値を指定してください。

記述例

例 1）-IDCodeNo1
　　ID コード：436F64654E6F31

番地	FFFDF	FFFE3	FFFEB	FFFEF	FFFF3	FFFF7	FFFFB
データ	43	6F	64	65	4E	6F	31

例 2）-IDCode
　　ID コード：436F6465000000
例 3）-ID1234567
　　ID コード：31323334353637
例 4）-ID#49562137856132
　　ID コード：49562137856132
例 5）-ID#1234567
　　ID コード：12345670000000
例 6）-ID
　　ID コード：FFFFFFFFFFFFFF

-L

データレコード長の選択

機能

- モトローラ S フォーマットのデータレコード長を 32 バイトに設定します。
- インテル HEX フォーマットのデータレコード長を 32 バイトに設定します。

記述規則

- ファイル名を指定する前に本オプションを指定してください。

記述例

```
>lmc30 -L debug
```

-O

出力ファイル名指定

機能

- lmc30 が生成する機械語ファイルのファイル名を指定します。
- ファイル名にはパスの指定ができます。
- ファイル名の拡張子を指定することができます。拡張子の指定を省略した場合は、モトローラ S フォーマットは".mot"、インテル HEX フォーマットは".hex"で生成されます。
- 出力ファイルは、指定されたアブソリュートモジュールファイルと同じディレクトリに出力されます。

記述規則

- -O　（ファイル名）のように指定してください。
- 本オプションとファイル名の間には、必ずスペースを入力してください。
- ファイル名を指定する前に本オプションを指定してください。

記述例

```
>lmc30 -O test debug          …(1)
>lmc30 -O tmp¥test debug     …(2)
```

(1) "test.mot"ファイルファイル名を指定
(2) "tmp"ディレクトリに"test.mot"ファイルを生成

-ofsregx

オプション機能選択レジスタに値を設定

機能

- 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。
- オプション機能選択レジスタ（FFFFH）に任意の値を格納します。
- ソースプログラムでオプション機能選択レジスタにデータを書き込んでいる場合も、本オプションを指定した場合は必ずデータを書き換えます。オプションを指定しないとソースプログラムで記述した値が出力されます。

注意事項
アセンブラ指示命令“.ID”、“.OFSREG”または“.PROTECT”が記述されている場合、本オプションは無効となります。

記述規則

- 本コマンドオプションは必ず小文字で指定してください。
- -ofsregx （設定値）のように入力してください。
- 本オプションとコードプロテクト値の間には、必ずスペースを指定してください。
- コードプロテクト値には、0 から 0FFH の範囲の値が記述できます。
- コードプロテクト値は、必ず 16 進数で設定してください。
- 本オプションは必ず、“-R8C”オプションと組み合わせて指定してください。
- -protect1、-protect1 または“-protectx”オプションと同時に指定することはできません。

記述例

```
>lmc30 –R8C –ofsregx FF sample
```

-protect1

ROM コードプロテクト機能のレベル 1 を設定

機能

- 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。
- ROM コードプロテクト制御番地（FFFFFH）に 3F を格納します。
- ソースプログラムで ROM コードプロテクト制御番地にデータを書き込んでいる場合も、本オプションを指定した場合は必ずデータを書き換えます。オプションを指定しないとソースプログラムで記述した値が出力されます。

注意事項
アセンブラ指示命令“.ID”、“.OFSREG”または“.PROTECT”が記述されている場合、本オプションは無効となります。

記述規則

- 本コマンドオプションは必ず小文字で入力してください。
- -protectx オプションと同時に指定することはできません。
- -R8C または -ofsregx オプションと同時に指定することはできません。

記述例

```
>lmc30 -protect1 sample
```

-protectx

ROM コードプロテクト制御番地に値を設定

機能

- 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。
- ROM コードプロテクト制御番地（FFFFFH）に任意の値を格納します。
- ソースプログラムで ROM コードプロテクト制御番地にデータを書き込んでいる場合も、本オプションを指定した場合は必ずデータを書き換えます。オプションを指定しないとソースプログラムで記述した値が出力されます。

注意事項
アセンブラ指示命令“.ID”、“.OFSREG”または“.PROTECT”が記述されている場合、本オプションは無効となります。

記述規則

- 本コマンドオプションは必ず小文字で指定してください。
- -protectx （コードプロテクト値）のように入力してください。
- 本オプションとコードプロテクト値の間には、必ずスペースを指定してください。
- コードプロテクト値には、0 から 0FFH の範囲の値が記述できます。
- コードプロテクト値は、必ず 16 進数で設定してください。
- -protect1 オプションと同時に指定することはできません。
- -ofsregx オプションと同時に指定することはできません。

注意事項
“-R8C”オプションと組み合わせて指定した場合、“-ofsregx”として処理されます。

記述例

```
>lmc30 –protectx FF sample
```

-R8C/-R8CE

生成コード制御

機能

- R8C ファミリに対応したコードを生成します。

オプション	アドレス空間
-R8C	0H 番地 ～ 0FFFFH 番地
-R8CE	0H 番地 ～ 0FFFFFH 番地

記述規則

- ファイル名を指定する前に本オプションを指定してください。

記述例

```
>lmc30 –R8C sample
```

-V

バージョン表示

機能

・ lmc30 のバージョン番号を表示します。

注意事項

本オプションを指定した場合は、コマンド行の他のパラメータは全て無視されます。

記述規則

・ 本オプションのみを指定してください。

記述例

```
>lmc30 -V
```

lmc30 エラーメッセージ一覧

'-A' Option Illegal format '-A StartAddr:EndAddr
？　開始および終了アドレスが正しく設定されていません。
！　開始および終了アドレスを確認してください
'-E' option is too long
？　-e オプションの引数の並びが長すぎます。
！　オプションの指定方法を確認して指定し直してください。
'-F' Option Illegal format '-F Data:StartAddr:EndAddr
？　開始および終了アドレスが正しく設定されていません。
！　開始および終了アドレスを確認してください。
'-protect' or '-ofsreg' option multiple specified
？　‘-protect’または‘-ofsreg’オプションが重複指定されています。
！　余分な’-protect’ または‘-ofsreg’オプションを削除してください。
'xxx' option multiple specified
？　オプション 'xxx' を複数指定しています。
！　オプションの指定方法を確認して指定し直してください。
Address specified by '-A' option exceed output address
？　指定されたアドレスがアブソリュートモジュールファイルに登録されているデータの範囲外です。
！　アブソリュートモジュールファイルに登録されているデータの範囲内で指定ください。
Address specified by '-E' option exceed xxx
？　-e オプションで指定したアドレスが xxx を越えました。
！　アドレス値を指定し直してください。
Address specified by '-F' option exceed output address
？　指定されたアドレスがアブソリュートモジュールファイルに登録されているデータの範囲外です。
！　アブソリュートモジュールファイルに登録されているデータの範囲内で指定ください。
Can't close file 'filename'
？　'filename'ファイルをクローズできません。
！　ディレクトリの情報を確認してください。
Can't create file 'filename'
？　'filename'ファイルが作成できません。
！　ディレクトリの情報を確認してください。
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。
Command line is too long
？　コマンド行の文字列が長すぎます。
！　コマンドを入力し直してください。
Illegal file format 'filename' is used
？　ファイルのフォーマットが間違っています。
！　ファイル名を確認してください。ファイルを生成し直してください。
Invalid option 'option' is used
？　無効なオプション'option'を指定しています。
！　オプションを指定し直してください。
Not enough memory
？　メモリが足りません。
！　メモリを増設してください。
Option 'option' is not appropriate
？　オプションの使用方法が間違っています。
！　オプションの指定方法を確認して指定し直してください。

Specifies option that can't use in M16C
？ M16C ファミリを使用する際、指定できないオプションが指定されています。
！ オプションを指定し直してください。
Specifies option that can't use with ‘-R8C’
？ ‘-R8C’オプションと同時に指定できないオプションが指定されています。
！ オプションを指定し直してください。
Unknown file extension '.xxx' is specified
？ 指定したファイルの拡張子'.xxx'に間違いがあります。
！ ファイル名を確認してください。
Value is out of range ‘xx’
？ 指定された値が範囲外です。
！ 指定範囲内での値を指定してください。

lmc30 ワーニングメッセージ一覧

‘-ID’ option isn’t processed
？ ‘-ID’オプションは処理されません。
！ アセンブラ指示命令”.ID”または”.PROTECT”の記述があります。
‘-ofsreg’ option isn’t processed
？ ‘-ofsregx’オプションは処理されません。
！ アセンブラ指示命令”.ID”または”.OFSREG”の記述があります。
‘-protect’ or ‘-ofsreg’ option isn’t processed
？ ‘-protect’ または‘-ofsregx’オプションは処理されません。
！ アセンブラ指示命令”.ID”、”.PROTECT”または”.OFSREG”の記述があります。
'filename' does not contain object data
？ 指定したファイルにオブジェクトデータがありません。
！ ファイル名を確認してください。
Address exceed xxx
？ アドレスが xxx を越えました。
！ ソースの記述内容を確認してください。セクションの配置指定を確認してください。
Original HEX format for microcomputers is generated
？ 専用 HEX ファイルが生成されました。
！ 専用 HEX ファイルで問題ないか確認下さい。

lb30 の操作方法

lb30 の機能を使用するための操作方法を説明します。lb30 の機能は、複数のリロケータブルモジュールファイルを一つのライブラリファイルとして管理します。

コマンドパラメータ

lb30 のコマンドパラメータ一覧を次に示します。

オプション名	機能
-.	画面へのメッセージ出力を停止する
-A	ライブラリファイルにモジュールを追加する
-C	ライブラリファイルを新しく作成する
-D	ライブラリファイル内のモジュールを削除する
-L	ライブラリリストファイルを生成する
-R	ライブラリファイル内のモジュールを置き換える
-U	ライブラリファイル内のモジュールを更新する
-V	ライブラリアンのバージョン番号を表示する
-X	ライブラリファイルに登録されているモジュールをリロケータブルモジュールファイルとして抽出する
@file	コマンドファイルの内容に従ってライブラリアンを実行する

コマンドパラメータの指定規則

lb30 のコマンドパラメータは次の規則に従って指定してください。

コマンドパラメータの指定順序

lb30 のコマンドパラメータは必ず次の順序で指定してください。指定順序が間違っている場合には、正しく lb30 の処理が行われません。

1　コマンドオプション
2　ライブラリファイル名
3　リロケータブルモジュール（ファイル）名

>lb30　（コマンドオプション）　（ライブラリファイル名）　（リロケータブルモジュールファイル名）

ライブラリファイル名

- ライブラリファイル名は必ず指定してください。
- ファイル名には、ディレクトリパスを指定できます。
- コマンド行では、拡張子(lib)は省略できます。

リロケータブルモジュールファイル名（リロケータブルモジュール名）

- リロケータブルモジュールファイル名は必ず指定してください。
- リロケータブルモジュールファイル名の拡張子は'.r30'です。コマンド行では、拡張子を省略できます。
- リロケータブルモジュールファイルは、複数指定できます。このとき、ファイル名同士の間には、必ずスペースを入力してください。
- ファイル名にはディレクトリパスを指定できます。ディレクトリの指定がない場合は、カレントディレクトリにあるファイルを処理します。

コマンドオプション

- コマンドオプションの大文字小文字は区別しません。
- コマンドオプションのうち、'-A','-C','-D','-L','-R','-U','-X'は、ライブラリアン実行時に必ず一つを指定してください。コマンド行の指定に、次に示すコマンドオプションが一つも指定していなかったり、二つ以上同時に指定がある場合は、エラーとなります。

コマンドファイル

lb30 は、コマンドパラメータをファイルに記述し、そのファイルを読み込んでプログラムを実行できます。

コマンドファイル名の指定方法

- コマンドファイル名の先頭には、@を付けて指定してください。
- コマンドファイル名には、ディレクトリパスを指定できます。
- 指定したディレクトリパスにファイルが存在しない場合は、エラーとなります。

例）

```
>lb30 @cmdfile
```

コマンドファイルの記述方法

コマンドファイルの記述方法は、「入出力ファイル」の「コマンドファイルフォーマット」を参照してください。

lb30 コマンドオプション

以降に、コマンドオプションの指定規則を説明します。

-.

画面へのメッセージ出力を停止

機能

- lb30 が処理を行う際のメッセージを出力しません。

記述規則

- 本オプションのみ他のオプションとの組み合わせが可能です。
- 他のオプションとの指定順序は任意です。

記述例

```
>lb30 -. -A new sample2
```

-A

モジュール追加

機能

- 既にあるライブラリファイルにリロケータブルモジュールを追加登録します。
- 指定したライブラリファイルが存在しない場合は、新しく作成します。
- 追加しようとしたリロケータブルモジュール名と同一のモジュールが登録されている場合は、エラーとなります。
- 追加しようとしたリロケータブルモジュールファイル内に、既にライブラリファイルに登録されているモジュール内で同じグローバルシンボル名の定義が存在する場合はエラーとなります。
- ライブラリファイルへのリロケータブルモジュールの登録は、コマンド行で指定された順に登録します。
- ライブラリファイルに登録されているリロケータブルモジュールを、そのリロケータブルモジュールファイルが生成された日時を基準に管理します。

記述規則

- -A　（ライブラリファイル名）　（リロケータブルモジュールファイル名）のように入力してください。
- 本オプションとライブラリファイル名の間及びライブラリファイル名とリロケータブルモジュールファイル名との間には、必ずスペースを入力してください。

記述例

new.lib ファイルに、sample3 を追加

```
>lb30 -A new.lib sample3.r30
```

-C

ライブラリファイル新規作成

機能

・ 新しくライブラリファイルを生成します。

注意事項
本コマンドオプションで指定したライブラリファイル名と同一のライブラリファイルが既に存在している場合は、古いライブラリファイルの内容は、新しく作成したライブラリファイルの内容に書き変わります。

記述規則

・ -C （ライブラリファイル名） （リロケータブルモジュールファイル名）のように入力してください。
・ 本オプションとライブラリファイル名の間及びライブラリファイル名とリロケータブルモジュールファイル名との間には、必ずスペースを入力してください。

記述例

sample1 と sample2 を登録した、new.lib ファイルを新しく作成

```
>lb30 -C new sample1 sample2
```

-D

モジュール削除

機能

・ ライブラリファイルに登録されているリロケータブルモジュールを削除します。
・ 削除したモジュールはどこにも残りません。

記述規則

・ -D （ライブラリファイル名） （リロケータブルモジュール名）
・ 本オプションとライブラリファイル名の間及びライブラリファイル名とリロケータブルモジュール名との間には、必ずスペースを入力してください。
・ 削除するリロケータブルモジュールを複数指定できます。このとき。モジュール名の間には、必ずスペースを入力してください。

記述例

・ new.lib というライブラリファイルに登録されている、sample2 というリロケータブルモジュールを削除します。

```
>lb30 -D new sample2
```

-L

ライブラリリストファイル生成

機能

- 指定したライブラリファイルの情報を格納したライブラリリストファイルを生成します。生成するライブラリリストファイルの拡張子は、'.lls'です。
- ライブラリファイル内の必要なモジュールの情報だけを格納したライブラリリストファイルを生成できます。
- 同じ名前のライブラリリストファイルが既に存在している場合は、新しいライブラリリストファイルに置き換えられます。
- ライブラリファイルに登録されているリロケータブルモジュールの情報を、ライブラリリストファイルに出力します。
- ライブラリリストファイルに出力するリロケータブルモジュールの作成日時は、すべてリロケータブルモジュールファイルの生成日時となります。

記述規則

- -L　（ライブラリファイル名）　[（リロケータブルモジュール名）]のように入力してください。
- 本オプションとライブラリファイル名の間及びライブラリファイル名とリロケータブルモジュールファイル名との間には、必ずスペースを入力してください。
- リロケータブルモジュール名は複数指定できます。このとき、リロケータブルモジュール名の間には、必ずスペースを入力してください。

記述例

```
>lb30 -L new                              …(1)
>lb30 -L new sample1                      …(2)
>lb30 -L new.lib sample1 sample3    …(3)
```

(1) new.lib というライブラリファイルに登録されている全てのモジュールの情報を new.lls というライブラリリストファイルに出力
(2) new.lib に登録されている sample1 というモジュールの情報を new.lls に出力
(3) new.lib に登録されている sample1,sample3 というモジュールの情報を new.lls に出力

-R

モジュール置き換え

機能

- ライブラリファイルに登録されているリロケータブルモジュールを指定したリロケータブルモジュールファイル内容に更新します。更新されるモジュールは、指定したリロケータブルモジュールファイル名と同一の名前のモジュールです。

記述規則

- -R　（ライブラリファイル名）　（リロケータブルモジュールファイル名）のように指定してください。
- 本オプションとライブラリファイル名の間及びライブラリファイル名とリロケータブルモジュールファイル名との間には、必ずスペースを入力してください。
- リロケータブルモジュールファイル名は複数指定できます。このとき、リロケータブルモジュールファイル名の間には、必ずスペースを入力してください。

記述例

new.lib ファイルに登録されている sample1 の内容を、同じ名前の sample1.r30 ファイルの内容に置き換え指定

```
>lb30 -R new sample1
```

-U

モジュール更新

機能

- ライブラリファイルに登録されているリロケータブルモジュールの作成日付と、更新しようとするリロケータブルモジュールファイルの作成日付を比較し、リロケータブルモジュールファイルの日付が新しい場合にのみ、モジュールを更新します。
- ライブラリファイルに登録されているリロケータブルモジュールの内容を更新します。
- ライブラリファイルに登録されているリロケータブルモジュールを、そのリロケータブルモジュールファイルが生成された日時を基準に管理します。

記述規則

- -U　（ライブラリファイル名）　（リロケータブルモジュールファイル名）のように入力してください。
- 本オプションとライブラリファイル名の間及びライブラリファイル名とリロケータブルモジュールファイル名との間には、必ずスペースを入力してください。
- リロケータブルモジュール名は複数指定できます。このとき、リロケータブルモジュール名の間には、必ずスペースを入力してください。

記述例

- new.lib ファイルに登録されている sample1 の作成日が、同じ名前の sample1.r30 ファイルの作成日より古い場合だけ、登録されている sample1 の内容を、sample1.r30 ファイルの内容に更新

```
>lb30 -U new sample1
```

-V

バージョン表示

機能

- lb30 のバージョン番号を表示します。

注意事項
本オプションを指定した場合は、コマンド行の他のパラメータは全て無視されます。

記述規則

- 本オプションのみを入力してください。

記述例

```
>lb30 -V
```

-X

モジュール抽出

機能

- ライブラリファイルに登録されているリロケータブルモジュールをリロケータブルモジュールファイルとして抽出します。
- ライブラリファイルは変更されません。
- 抽出したリロケータブルモジュールファイルの作成日時は、抽出した日時となります。ただし、リロケータブルモジュールファイル内のファイル生成情報は、as30 がそのリロケータブルモジュールファイルを生成した日時となります。
- 抽出したリロケータブルモジュールファイル名と同一名のファイルが存在する場合、上書きします。
- 抽出されたリロケータブルモジュールファイルは、as30 が出力したリロケータブルモジュールファイルと、同一の内容になります。

記述規則

- 本オプションとライブラリファイル名の間には、必ずスペースを入力してください。

記述例

- new.lib ファイルに登録されている、sample3 から、sample3.r30 というリロケータブルモジュールファイルを生成

```
>lb30 -X new sample3
```

@

コマンドファイル参照

機能

- 指定したファイルの内容をコマンドパラメータとして lb30 を起動します。

記述規則

- @（ファイル名）のように入力してください。
- 本オプションとファイル名の間に、スペース又はタブは記述できません。
- 他のパラメータはコマンド行に記述できません。

記述例

```
>lb30 @cmdfile
```

lb30 エラーメッセージ一覧

'filename' is not library file
? 'filename'ファイルはライブラリファイルではありません。
! ファイル名を確認してください。ファイルが lb30 で生成されたものであることを確認してください。

'filename' is not relocatable file
? 'filename'ファイルがリロケータブルファイルではありません。
! ファイル名を確認してください。ファイルが as30 で生成されたものであることを確認してください。

'module' already registered in 'filename'
? モジュール'module'はライブラリ'filename'に登録済みです。
! ライブラリファイル名及びリロケータブルファイル名を確認してください。

'module' does not match with 'filename'
? モジュール名'module'とリロケータブルファイル名'filename'が異なります。モジュール名が変更されています。
! リロケータブルファイル名を確認してください。

'module' is multiple specified
? 同一のモジュール名'module'を複数指定しています。
! モジュール名を指定し直してください。

'module' is not registered in 'filename'
? モジュール'module'がライブラリファイル'filename'に登録されていません。指定された処理（モジュールの削除又は更新）はできません。
! モジュール名を確認してください。

'symbol' is multiple defined at 'module1' and 'module2' in 'filename'
? 同名外部定義シンボル'symbol'がライブラリ'filename'中の'module1'と'module2'に二重定義されています。
! リロケータブルファイル名を確認してください。

'symbol' is multiple defined in 'filename'
? シンボル'symbol'は'filename'ファイルに二重定義されています。
! 本エラーが発生した場合は、お手数ですが弊社までご連絡いただきますようお願いします。

'symbol' is multiple defined in 'module1' and 'module2'
? 同名外部定義シンボル'symbol が'module1'と'module2'に二重定義されています。
! リロケータブルファイル名を確認してください。

'xxx' and 'xxx' are used
? 'xxx'オプションと'xxx'オプションを同時に使用しています。
! オプションは、同時に指定できません。コマンドを入力し直してください。

Can't close file 'filename'
? 'filename'ファイルをクローズできません。
! ディレクトリ情報を確認してください。

Can't close temporary file
? テンポラリファイルがクローズできません。
! ディレクトリ情報を確認してください。

Can't create file 'filename'
? 'filename'ファイルが作成できません。
! ディレクトリ情報を確認してください。

Can't create temporary file
? テンポラリファイルが生成できません。
! ディレクトリ情報を確認してください。

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。

Can't open temporary file
？　テンポラリファイルがオープンできません。
！　ディレクトリ情報を確認してください。
Can't write in file 'filename'
？　filename'ファイルの書き込みができません。メモリが不足しています。
！　メモリを増設してください。
Command-file is included in itself
？　コマンドファイル自身をインクルードしています。
！　コマンドファイルの記述内容を確認してください。
Command-file line characters exceed
？　コマンドファイルの 1 行の文字数が制限を越えています。
！　コマンドファイルの内容を確認してください。
Command line is too long
？　コマンド行の文字列が長すぎます。
！　コマンドファイルを作成してください。
Illegal file format 'filename'
？　'filename'ファイルのフォーマットが間違っています。
！　ファイル名を確認してください。
Invalid option 'option' is used
？　無効なオプション'option'を指定しています。
！　オプションを指定し直してください。
No public symbol is in 'filename'
？　ファイル'filename'に外部定義シンボルがありません。
！　リロケータブルファイルの内容を確認してください。
Not enough memory
？　メモリが足りません。
！　メモリを増設してください。
Symbol-name characters exceed 500
？　シンボル名が 500 文字をこえました。
！　ライブラリファイルを複数に分割してください。
Too many modules
？　登録モジュール数が多すぎます。
！　ライブラリファイルを複数に分割してください。
Unknown file extension '.xxx' is used
？　ファイル拡張子'.xxx'が間違っています。
！　ファイル名を確認してください。

lb30 ワーニングメッセージ一覧

'module' is not registered in library
？　モジュール'module'がライブラリに登録されていません。該当するモジュールは抽出しませんでした。
！　モジュール名を確認してください。
'module' is not registered in library, can't output list-file
？　モジュール'module'がライブラリに登録されていません。リストファイルに情報を出力しませんでした。
！　モジュール名を確認してください。
'module' was created in the current directory
？　モジュール'module'をカレントディレクトリに生成しました。
！　指定したディレクトリ名を確認してください。
Can't replace, 'module' is older than module in library
？　モジュール'module'の作成日時がライブラリ中のモジュールより古いため置換しませんでした。
！　リロケータブルファイルの生成日時を確認してください。

xrf30 の操作方法

xrf30 の機能を使用するための操作方法を説明します。xrf30 の機能は、指定したアセンブリソースファイルとアセンブラリストファイルから、分岐命令及びサブルーチン呼び出し命令の参照リスト（クロスリファレンス）ファイルを生成します。

コマンドパラメータ

xrf30 のコマンドパラメータ一覧を次に示します。

オプション名	機能
-.	画面へのメッセージ出力を停止する
-N	システムラベル情報をクロスリファレンスファイルに出力する
-O	クロスリファレンスファイルを出力するディレクトリを指定する
-V	クロスリファレンサのバージョン番号を表示する
@file	コマンドファイルの内容に従ってクロスリファレンサを実行する

コマンドパラメータの指定規則

xrf30 のコマンドパラメータは次の規則に従って指定してください。

コマンドパラメータの指定順序

- xrf30 のコマンドパラメータは任意の順序で指定できます。
  >xrf30　（ファイル名）　（コマンドオプション）
  >xrf30　（コマンドオプション）　（ファイル名）

アセンブリソースファイル名又はアセンブラリストファイル名

- 一つ以上のファイル名を必ず指定してください。
- ファイル名には、パスが指定できます。
- 最大 600 個までのファイルを指定できます。
- ファイル拡張子は必ず記述してください。
- 必ず、ファイル拡張子が".lst"のアセンブラリストファイルを指定してください。
- 複数のファイルを指定する場合は、ファイル名をスペース又はタブで区切って指定してください。

コマンドオプション

- 複数のコマンドオプションを指定できます。

コマンドファイル

- 入力パラメータを記述したコマンドファイル名を指定できます。
- コマンドファイルの記述方法は、ln30 の操作方法を参照してください。

xrf30 コマンドオプション

以降に、コマンドオプションの指定規則を説明します。

-.

画面へのメッセージ出力停止

機能

- xrf30 が処理を行う際のメッセージを出力しません。

記述規則

- 本オプションは、コマンド行の任意の位置に指定できます。

記述例

```
>xrf30 -. sample.a30
```

-N

システムラベル情報出力指定

機能

- as30 が出力するシステムラベルについての情報もクロスリファレンスファイルに出力します。
- システムラベルは、ピリオド２つ（..）で始まるラベルです。

記述規則

- 本オプションは、コマンド行の任意の位置に指定できます。

記述例

```
>xrf30 -N sample.lst   …(1)
>xrf30 -N sample.a30   …(2)
```

(1) sample.lst ファイルから、sample.xrf ファイルを生成
(2) sample.a30 ファイルから、sample.xrf ファイルを生成

-O

ファイル出力ディレクトリ指定

機能

- クロスリファレンスファイルを出力するディレクトリを指定します。

記述規則

- -O（ディレクトリ名）のように入力してください。
- 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。
- 本オプションは、コマンド行の任意の位置に指定できます。

記述例

```
>xrf30 –O¥work¥list sample.a30   …(1)
>xrf30 –O¥work¥list sample.lst   …(2)
```

(1) ¥work¥list ディレクトリに sample.xrf ファイルを生成
(2) ¥work¥list ディレクトリに sample.xrf ファイルを生成

-V

バージョン表示

機能

- xrf30 のバージョン番号を表示します。

注意事項
本オプションを指定した場合は、コマンド行の他のパラメータは全て無視されます。

記述規則

- 本オプションのみを指定してください。

記述例

```
>xrf30 -V
```

@

コマンドファイル参照

機能

- 指定したファイルの内容をコマンドパラメータとして、xrf30 を起動します。

記述規則

- 本オプションとファイル名の間に、スペース又はタブは記述できません。
- 他のパラメータはコマンド行に入力できません。

記述例

```
>xrf30 @cmdfile
```

xrf30 エラーメッセージ一覧

Can't create temporary file
？　テンポラリファイルの生成ができません。
！　ディレクトリ情報を確認してください。
Can't open file 'xxxx'
？　xxxx 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。
Command-file is included in itself
？　コマンドファイル自身をインクルードしています。
！　コマンドファイルの記述内容を確認してください。
Command-file line characters exceed
？　コマンドファイルの 1 行の文字数が制限を越えています。
！　コマンドファイルの内容を確認してください。
Command line is too long
？　コマンド行の文字列が長すぎます。
！　コマンドファイルを作成してください。
Input files exceed 80
？　入力ファイルの数が 80 を越えました。
！　コマンドを入力し直してください。コマンドファイルの内容を分割してください。
Invalid option 'xxx' is used
？　無効なコマンドオプション'xxx'の指定があります。
！　コマンドオプションを指定し直してください。
No input files specified
？　入力ファイルの指定がありません。
！　ファイル名を指定してください。
Not enough memory
？　メモリが足りません。
！　メモリを増設してください。
Option 'xxx' is not appropriate
？　コマンドオプションの指定が間違っています。
！　コマンドオプションの指定方法を確認して指定し直してください。

abs30 の操作方法

abs30 使用上のお願い

- １つのアセンブリソースファイル内に、同一名のセクションが複数個定義されており、そのセクションが指示命令'.LIST'で、アセンブラリストファイルに出力されていない場合には、正しい実アドレスが出力されない場合があります。また、「AS30」アセンブラプログラム以外のプログラム記述には対応していません。
- ソースファイルに構造化記述命令及びマクロ命令が記述されている場合は、必ずアセンブラリストファイルに構造化記述命令及びマクロ命令の展開行を出力するようにしてください（コマンドオプション"-LMS"を指定してアセンブルを行う）。また、アセンブラリストファイルにヘッダが出力されている必要があります（コマンドオプション-H は指定しないで as30 を起動してください）。

コマンドパラメータ

abs30 のコマンドパラメータ一覧を次に示します。

オプション名	機能
-.	画面へのメッセージ出力を停止する
-D	アセンブラリストファイルを検索するディレクトリを指定する
-O	アブソリュートリストファイルを出力するディレクトリを指定する
-V	アブソリュートリスタのバージョン番号を表示する

コマンドパラメータの指定規則

abs30 のコマンドパラメータは次の規則に従って指定してください。

コマンドパラメータの指定順序

- コマンドパラメータは、必ず次の順序に従って指定してください。

1　コマンドオプション
2　アブソリュートモジュールファイル名
3　アセンブラリストファイル名

>abs30　(コマンドオプション)　(アブソリュートモジュールファイル名)　(アセンブラリストファイル名)

アブソリュートモジュールファイル名（必須）

- アブソリュートモジュールファイル名は必ず指定してください。
- アブソリュートモジュールファイル名にはパスが指定できます。
- 拡張子(.x30)は省略できます。

アセンブラリストファイル名

- アセンブラリストファイルは、スペース又はタブで区切って複数指定することができます。
- アセンブラリストファイル名にはパスが指定できます。
- ファイル属性は省略できます。
- アセンブラリストファイル名は省略できます。

コマンドオプション

- コマンドオプションの大文字小文字は区別しません。
- コマンドオプションとその引数の間には、スペース又はタブを必ず入力してください。

abs30 コマンドオプション

以降に、コマンドオプションの指定規則を示します。

-.

画面へのメッセージ出力停止

機能

- abs30 が処理を行う際のメッセージを出力しません。

記述規則

- 本オプションは、コマンド行の任意の位置に指定できます。

記述例

```
>abs30 -. sample.x30
```

-D

ファイル検索ディレクトリ指定

機能

- アセンブラリストファイルの参照先ディレクトリを指定します。
- 本指定がない場合は、カレントディレクトリからアセンブラリストファイルを検索します。

記述規則

- -D（ディレクトリ名）のように入力してください。
- 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。

記述例

```
>abs30 -Ddir sample              …(1)
>abs30 -Ddir sample list1    …(2)
```

(1) カレントディレクトリの下の"dir"内のアセンブラリストファイルを検索
(2) カレントディレクトリの下の"dir"内の"list1.lst"を検索

-O

ファイル出力ディレクトリ指定

機能

- アブソリュートリストファイルの生成ディレクトリを指定します。
- 本指定がない場合は、カレントディレクトリにファイルを生成します。

記述規則

- -O（ディレクトリ名）のように入力してください。
- 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。

記述例

- カレントディレクトリの下の"abslist"ディレクトリにアブソリュートリストファイルを生成

```
>abs30 -Oabslist sample
```

-V

バージョン表示

機能

- abs30 のバージョン番号を表示します。
- 本オプションを指定した場合は、コマンド行の他のパラメータは全て無視されます。

記述規則

- 本オプションのみを指定してください。

記述例

```
>abs30 -V
```

abs30 エラーメッセージ一覧

Can't create file 'filename'
？　'filename'が生成できません。
！　ディレクトリ情報を確認してください。
Can't open file 'filename'
？　'filename'がオープンできません。
！　ファイル名を確認してください。
Can't write in file 'filename'
？　'filename'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。
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。
Error information is in 'filename'
？　'filename'はエラー情報を含んでいます。
！　アセンブラリストファイルを生成し直してください。
Illegal file format 'filename'
？　'filename'のフォーマットが正しくありません。
！　ファイル名を確認してください。
Input files number exceed 80
？　入力ファイル数が 80 を越えています。
！　コマンドを入力し直してください。
Not enough disk space
？　ディスク容量が不足です。
！　ディスク情報を確認してください。
Not enough memory
？　メモリ容量が不足です。
！　メモリを増設してください。
Section information is not appropriate in 'filename'
？　'filename'内のセクション情報が正しくありません。
！　ファイル名を確認してください。

abs30 ワーニングメッセージ一覧

Address area exceed 0FFFFFH
？　アドレス範囲 0FFFFFH を越えています。
！　アブソリュートモジュールファイル名を確認してください。
File 'l-filename' is missing corresponding to module in 'a-filename'
？　'a-filename'が持つモジュールに相当する'l-filename'がありません。該当するアブソリュートリストファイルは生成しませんでした。
！　アセンブラリストファイルを生成し直してください。アセンブラリストファイルのあるディレクトリを確認してください。
Lines 'num-num' are relocatable address in 'filename'
？　'filename'の'num-num'行はリロケータブルアドレスのままです。
！　アセンブリソースファイルに指示命令".LIST OFF"が記述されていることを確認してください。
No information of 'l-filenam' in 'a-filename'
？　'a-filename'は、'l-filename'の情報を持っていません。
！　ファイル名を確認してください。
No section information of l-name in x-name
？　x-name は、l-name のセクション情報を持っていません。
！　l-name からアブソリュートリストファイルは生成できません。
Overwrite in 'filename'
？　'filename'に上書きします。
！　古いファイルの内容は保存されません。

プログラムの記述規則

AS30 対応のソースプログラムを記述するための基本規則を示します。

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、ユーザーがソースプログラム内に記述した場合でもエラーは出力しませんが、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。

プログラムの記述規則

文字セット

「AS30 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。

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。

予約語は特別の機能をもっており、したがって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。

予約語は、大文字と小文字を区別しません。したがって、"ABS"と"abs"は同じ予約語となります。

以下のものが予約語になります。

アセンブル指示命令

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の全ては予約語です。

ニーモニック

M16C ファミリのニーモニック全てが予約語です。

レジスタ／フラグ名

M16C ファミリのレジスタ名及びフラグ名は全て予約語です。

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。

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。

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。

名前

名前は、ソースプログラムの中で、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。

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。
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コードの 1 バイト目のアドレスを示します。
マクロ名	マクロの定義名です。

名前の記述規則

- 名前の長さ
  名前として記述できる文字列の長さは、255 文字までです。
- 名前の判別
  名前は、大文字と小文字を区別して扱います。"LAB","Lab"は異なる名前として扱います。

注意事項
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。

ラベルの記述規則

- 名前には英数字とアンダーラインが使用できます。
- 名前の先頭には、数字は使用できません。
- 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。
- 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:
```

ラベルの参照方法

- ニーモニックのオペランドに名前を記述します。

例）

```
JMP         sym_name
```

シンボルの記述規則

- 数値は、アセンブル実行時に確定しなければなりません。
- 名前には英数字とアンダーラインが使用できます。
- 名前の先頭には、数字は使用できません。
- セクションの範囲外で定義できます。
- 数値定義の指示命令'.EQU'を使用します。

例）

```
value1      .EQU     1
value2      .EQU     2
```

シンボルの参照方法

・ 命令のオペランドにシンボルを記述します。

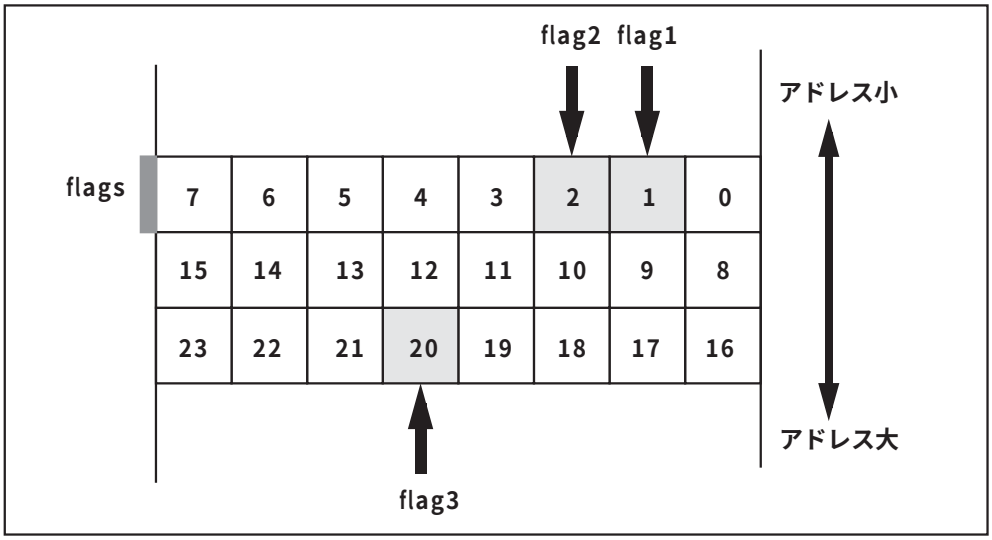
```
MOV.W       R0,value1
value3      .EQU        value2+1
```

ビットシンボルの記述規則

・ 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。
・ 名前には英数字とアンダーラインが使用できます。
・ 名前の先頭には、数字は使用できません。
・ セクションの範囲外で定義できます。
・ ビットシンボル定義の指示命令'.BTEQU'を使用します。

例）

```
flag1       .BTEQU   1,flags
flag2       .BTEQU   2,flags
flag3       .BTEQU   20,flags
```



ビットシンボルの参照方法

・ 1 ビット操作命令のオペランドに記述できます。

```
BCLR    flag1
BCLR    flag2
BCLR    flag3
```

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。
・ 名前や、予約語の先頭に'$'は、記述できません。
・ 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。

記述例）

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

注意事項
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。

行の記述方法

行の種類

as30 アセンブラは、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。

指示命令行

- as30 の指示命令を記述した行です。
- 指示命令は、一行に一つのみ記述できます。
- 指示命令行には、コメントを記述できます。

注意事項
指示命令とニーモニックを同じ一行に記述できません。

アセンブリソース行

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

注意事項
１行には、２つ以上のニーモニックは記述できません。
指示命令とニーモニックを同じ一行に記述できません。

ラベル定義行

- ラベル名だけを記述した行です。

コメント行

- コメントだけを記述した行です。

空行

- スペース、タブ又は改行コードだけを含む行です。

行の記述規則

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。

行の長さ

一行に記述可能な最大文字数は 512 文字です。512 文字を越えた文字については、as30 は処理しません。

注意事項
各行の記述は、必ず一行の範囲に記述してください。

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。

注意事項

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。

- 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。

記述例

```
    .SECTION   program,DATA
    .ORG       00H
sym             .EQU    0
work:           .BLKB   1
    .ALIGN
    .PAGE      "newpage"          ;newpage
    .ALIGN
```

アセンブリソース行の記述規則

ニーモニックの記述方法は『M16C ファミリ　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リ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。
- アセンブリソース行の先頭には、スペース又はタブを記述できます。
- アセンブリ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。
- アセンブリ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。

注意事項

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。

- 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。

記述例

```
    MOV.W   #0,R0
    RTS
main:       MOV.W   #0,A0
    RTS             ; End of subroutine
```

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。
- ラベル定義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。

注意事項

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。

- 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:
rabel:      .BLKB   1
main:       nop
loop:
```

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。
- 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
```

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

    JSR     sub1
```

行の連結

- 行末に'¥¥'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。

注意事項
連結された結果の行の最大文字数が 512 文字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。

- 次に行連結の記述例と連結結果を示します。

例 1）

```
        .BYTE   1,¥¥
                        2, ¥¥
                        3       ¥¥
                        ,4
```

連結結果

```
        .BYTE   1,2, 3       ,4
```

例 2）

```
        .BYTE   1,¥¥ ;commennt
                        2,;Comment ¥¥
                        3 ;COMMENT
```

連結結果

```
        .BYTE   1,2, ;Comment
                        3 ;COMMENT
```

例 3）

```
        .BYTE   1,¥¥
                        2,¥¥
                        3,   ¥¥
                        4
```

連結結果

```
        .BYTE   1,2,3,   4
```

オペランド

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。

注意事項
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。

数値

数値には、整数と、浮動小数点数が含まれます。

名前

ラベル名及びシンボル名が記述できます。

式

数値及び名前を項に持つ式が記述できます。

文字列

文字又は文字列を ASCII コードとして扱えます。

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。
各オペランドの記述規則を説明します。

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。

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

2 進数

- 0～1 のいずれかの数字で記述し末尾に B 又は b を添付する

例）

```
1010001B
1010001b
```

8 進数

- 0～7 までの数字で記述し末尾に O 又は o を添付する

例）

```
60702O
60702o
```

10 進数

- 0～9 までの数字で記する

例）

```
9423
1024
```

16 進数

- 0～9、a～f 又は A～F で記述し末尾に H 又は h を添付する
- アルファベットで始まる数値の場合は先頭に 0 を添付する

例）

```
0A5FH
5FH
0a5fh
5fh
```

浮動小数点数

- 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)　　$1.17549435 \times 10^{-38} \sim 3.40282347 \times 10^{38}$

DOUBLE(64bits)　　$2.2250738585072014 \times 10^{-308} \sim 1.7976931348623157 \times 10^{308}$

注意事項

浮動小数点数は、指示命令".DOUBLE",".FLOAT"のオペランドだけに記述できます。浮動小数点数は、0～9 までの数字と E 又は e で記述してください。

例）

```
3.4E35          ;3.4x1035
3.4e-35         ;3.4x10-35
-.5E20          ;-0.5x1020
5e-20           ;5.0x10-20
```

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。
- 式の項に文字定数は使用できません。
- 式の演算結果の値の範囲は、－2147483648～2147483648 となります。

注意
演算結果が、－2147483648～2147483648 を超えた場合もオーバフロー又はアンダーフローは判断しません。

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。

注意
演算子"SIZEOF"及び"TOPOF"は、オペランドとの間に、必ずスペース又はタブを記述してください。
条件演算子は、指示命令".IF"及び".ELIF"のオペランドだけに記述できます。

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。

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

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（指示命令.IF,.ELIF のオペランドだけに記述できます）。
＜	右辺値が左辺値より大きいことを評価します（指示命令.IF,.ELIF のオペランドだけに記述できます）。
＞＝	左辺値が右辺値より大きいか等しいことを評価します（指示命令.IF,.ELIF のオペランドだけに記述できます）。
＜＝	右辺値が左辺値より大きいか等しいことを評価します（指示命令.IF,.ELIF のオペランドだけに記述できます）。
＝＝	左辺値と右辺値が等しいことを評価します（指示命令.IF,.ELIF のオペランドだけに記述できます）。
！＝	左辺値と右辺値が等しくないことを評価します（指示命令.IF,.ELIF のオペランドだけに記述できます）。

演算優先順位変更演算子

演算子	機能
（）	（）で囲った演算を最優先で行います。一つの式に複数の（）が記述されている場合は、左が優先になります。（）はネストした記述ができます。

式の演算優先順位

as30 は、オペランドに記述されている式について、次に示す優先順位に従って、演算を行った結果の数値をオペランドの値として扱います。

1　演算子が持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次の表に示します。表の優先順位の欄の値の小さいものほど優先順位は高くなります。
2　同一の優先順位を持つ演算子は、左から順に演算を行います。
3　（）で囲うことで演算の優先順位を変更できます。

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

式とその値

式の記述例と、as30 が演算を行った結果の値について、次に例を示します。

式	演算結果
2+6/2	5
(2+6)/2	4
1<<3+1	16
(1<<3)+1	9
3*2%4/2	1
(3*2)%(4/2)	0
8|4/2	10
(8|4)/2	6
8&8/2	0
(8&8)/2	4
6*-3	-18
-(6*-3)	18
-6*-3	18

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。

例）

```
"string"
'string'
```

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 ファミリソフトウェアマニュアル」を参照してください。

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（8,16,32)を指定します。必ず指定してください。

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。

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。

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。
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]
```

ディスプレースメントつきアドレッシングモードの場合はディスプレースメントを指定します。

指示命令

AS30 対応のソースプログラムには、M16C ファミリのニーモニック以外に指示命令が記述できます。指示命令には、次の種類があります。

アドレス制御指示命令

アセンブル実行時にアドレス決定の指示ができます。

アセンブル制御指示命令

as30 の実行について指示できます。

リンク制御指示命令

アドレス再配置制御のための情報を定義できます。

リスト制御指示命令

as30 が生成するリストファイルのフォーマットを制御できます。

分岐最適化制御指示命令

分岐命令の最適選択を as30 に指示できます。

条件アセンブル制御指示命令

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

拡張機能指示命令

上記以外の制御を行う指示命令です。

インスペクタ情報出力制御命令

インスペクタ情報の出力を制御する指示命令です。

クロスツールが出力する指示命令

M16C ファミリ用 ツールソフトウェアが出力する指示命令は、ユーザーはソースプログラムに記述できません。記述した場合の動作については保証しません。

アドレス制御指示命令

as30 がアドレスの更新を行う場合の指示をします。

絶対属性セクション内のアドレスをのぞいて、as30 が制御を行うアドレスはリロケータブル値です。

アドレスを制御する指示命令には次のものがあります。

.ORG

本指示命令を記述した行以降の行の生成コードのアドレス値を指定します。本指示命令を記述したセクションは、絶対属性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

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。

アセンブル制御指示命令

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命令です。アドレスの更新は行いません。

アセンブル制御を指示する命令には次のものがあります。

.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

ソースプログラム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 相対変位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 相対変位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 相対変位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。

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。

.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。as30 及び ln30 はセクション情報を基に、コード生成及びリロケータブルファイルの統合とセクションの再配置を行います。

.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。
外部参照シンボルについては、as30 はアドレスを保留します。このとき、その値は ln30 が決定します。

.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。as30 及び ln30 の処理は.GLB と同様です。

.VER

オペランドに記述した文字列を ln30 が生成するマップファイルに、バージョン情報として出力します。この情報を利用して、リンクの制御ができます。

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。

.LIST

リストファイルを生成する際に、アセンブリソースファイルの行単位でリストファイルへの出力を行うか行わないかを制御します。

.PAGE

リストファイルを生成する際に、アセンブリ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。

分岐命令最適化制御指示命令

as30 は、無条件分岐命令とサブルーチン呼び出し命令を可能な、もっとも短いコードで生成できます。この機能の制御を行う指示命令の説明をします。

.OPTJ

無条件分岐命令とサブルーチン呼び出し命令の最適化を制御します。

拡張機能指示命令

これらの指示命令は、コードを生成しません。

.ASSERT

オペランドに記述した文字列を標準エラー出力又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リ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。

.ID

オペランドに記述した ID コードを ln30 が生成するマップファイルに、ID コード情報として出力します。また lmc30 が生成する ID ファイルにも出力します。

.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。

.OFSREG

オペランドに記述したオプション機能選択レジスタ値を ln30 が生成するマップファイルに、オプション機能選択レジスタデータ情報として出力します。

.PROTECT

オペランドに記述した ROM コードプロテクト値を ln30 が生成するマップファイルに、ROM コードプロテクトデータ情報として出力します。

.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。

.SB_AUTO(_xxx)

SB 相対アドレッシングの自動生成を実施します。

.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。

インスペクタ情報出力制御命令

インスペクタ情報の出力を制御する指示命令です。

.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

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。

条件アセンブル制御指示命令

as30 は、条件アセンブル指示命令を使って、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。

as30 で記述できる、条件アセンブル指示命令を次に示します。

.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。

マクロ指示命令

.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。

.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。

.ENDM

マクロボディの終了を示します。

.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っています。

..MACREP

繰り返しマクロボディの展開回数を値として持っています。

.LEN

指定した文字列の文字数を値として持っています。

.INSTR

指定した文字列の中での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。

指示命令の記述方法

それぞれの指示命令の機能と記述方法を詳しく説明します。以降の説明は、指示命令をアルファベット順に説明しています。

..FILE

ソースファイル名情報に置き換え

機能

・ as30 が処理中のファイル名に展開されます（アセンブリソースファイル又はインクルードファイル）。

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"を指定すると、"..FILE"は、コマンド行で指定したアセンブリ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。

記述形式

..FILE

記述規則

・ 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。

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

コマンド入力例）

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

..MACPARA

実引数の数に置き換え

機能

- 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。

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 となります。

記述形式

```
..MACPARA
```

記述規則

- 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         f1,f2
    .ELSE
    ADD         R0,f1
    .ENDIF
    .ENDM

    name        mem

    .ELSE
    ADD         R0,mem
    .ENDIF
    .ENDM
```

..MACREP

現在のマクロの繰り返し回数に置き換え

機能

- 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。

注意事項

マクロボディの外に記述した場合は、値は０となります。

- 条件アセンブルのオペランドに記述できます。

記述形式

..MACREP

記述規則

- 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
    .ENDM
```

.ADDR

3 バイト長のデータを格納

機能

- 3 バイト長の固定データを ROM に格納します。
- データを格納したアドレスにラベルを定義することができます。

記述形式

```
          .ADDR   （数値）
（名前:）   .ADDR   （数値）
```

記述規則

- オペランドに整数値を記述してください。
- 指示命令と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。
- オペランドにはシンボルを記述できます。
- オペランドには式を記述できます。
- 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、文字又は、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。

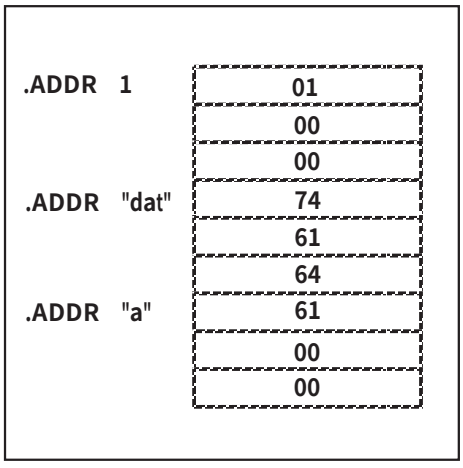
注意事項

オペランドに記述できる文字列長は、３文字までです。

- ラベルを定義する場合には、指示命令の前にラベル名を記述してください。
- ラベル名には必ず、コロン(:)を記述してください。

記述例

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



.ALIGN

アドレス補正

機能

- 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 又は、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。

記述形式

.ALIGN

記述規則

- 本指示命令は、次の条件に当てはまるセクション内に記述できます。

1　セクション定義の際にアドレス補正を指示している相対属性セクション

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

2　絶対属性セクション

```
.SECTION    program,CODE
            .ORG        0e000H
```

記述例

```
            .SECTIONprogram,CODE,ALIGN
            MOV.W    #0,R0
            .ALIGN
            .END

            .SECTIONprogram,CODE
            .ORG        0f000H
            MOV.W    #0,R0
            .ALIGN
            .END
```

```
ソース                                  アドレス,コード

.SECTION        count,ROMDATA,ALIGN
.ADDR   1                               00000   010000
.ALIGN                                  00003   04      NOPコードを挿入
.SECTION        ram,DATA,ALIGN
.BLKA   1                               00000
.ALIGN                                  00003           アドレスを＋1
.BLKB   1                               00004
.END
```

.ASSERT

指定文字列を出力

機能

- 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。

記述形式

```
.ASSERT     "（文字列）"
.ASSERT     "（文字列）" > （ファイル名）
.ASSERT     "（文字列）" >> （ファイル名）
```

記述規則

- オペランドと指示命令の間には、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、">"又は">>"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。
- ">"又は">>"の前後には、スペース又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。

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
```

.BLKA

3 バイト長の領域を確保

機能

- 3 バイト単位で、指定したバイト数の RAM 領域を確保します。
- 確保した RAM のアドレスに、ラベル名を定義することもできます。

記述形式

```
                .BLKA    （数値）
（名前:）       .BLKA    （数値）
```

記述規則

- 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。
  セクション定義の際に、セクション名に続けて",DATA"を記述することでセクションタイプが DATA タイプとなります。
- 指示命令と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。
- オペランドに整数値を記述してください。
- オペランドにはシンボルを記述できます。
- オペランドには式を記述できます。
- オペランドの式の値は、アセンブル実行時に確定しなければなりません。
- 領域にラベル名を定義する場合は、指示命令の前にラベル名を記述してください。ラベル名には、必ずコロン(:)を記述してください。

記述例

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

.BLKB

1 バイト長の領域を確保

機能

- 1 バイト単位で、指定したバイト数の RAM 領域を確保します。
- 確保した RAM のアドレスに、ラベル名を定義することもできます。

記述形式

```
                .BLKB    （数値）
（名前:）       .BLKB    （数値）
```

記述規則

- 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。
  セクション定義の際に、セクション名に続けて",DATA"を記述することでセクションタイプが DATA タイプとなります。
- 指示命令と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。
- オペランドに整数値を記述してください。
- オペランドにはシンボルを記述できます。
- オペランドには式を記述できます。
- オペランドの式の値は、アセンブル実行時に確定しなければなりません。
- 領域にラベル名を定義する場合は、指示命令の前にラベル名を記述してください。ラベル名には、必ずコロン(:)を記述してください。

記述例

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

.BLKD

8 バイト長の領域を確保

機能

- 8 バイト単位で、指定したバイト数の RAM 領域を確保します。
- 確保した RAM のアドレスに、ラベル名を定義することもできます。

記述形式

```
            .BLKD    （数値）
（名前:）    .BLKD    （数値）
```

記述規則

- 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。
  セクション定義の際に、セクション名に続けて",DATA"を記述することでセクションタイプが DATA タイプとなります。
- 指示命令と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。
- オペランドに整数値を記述してください。
- オペランドにはシンボルを記述できます。
- オペランドには式を記述できます。
- オペランドの式の値は、アセンブル実行時に確定しなければなりません。
- 領域にラベル名を定義する場合は、指示命令の前にラベル名を記述してください。ラベル名には、必ずコロン(:)を記述してください。

記述例

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

.BLKF

4 バイト長の領域確保

機能

- 4 バイト単位で、指定したバイト数の RAM 領域を確保します。
- 確保した RAM のアドレスに、ラベル名を定義することもできます。

記述形式

```
                .BLKF    （数値）
（名前:）       .BLKF    （数値）
```

記述規則

- 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。
  セクション定義の際に、セクション名に続けて",DATA"を記述することでセクションタイプが DATA タイプとなります。
- 指示命令と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。
- オペランドに整数値を記述してください。
- オペランドにはシンボルを記述できます。
- オペランドには式を記述できます。
- オペランドの式の値は、アセンブル実行時に確定しなければなりません。
- 領域にラベル名を定義する場合は、指示命令の前にラベル名を記述してください。ラベル名には、必ずコロン(:)を記述してください。

記述例

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

.BLKL

4 バイト長の領域確保

機能

- 4 バイト単位で、指定したバイト数の RAM 領域を確保します。
- 確保した RAM のアドレスに、ラベル名を定義することもできます。

記述形式

```
                .BLKL    （数値）
（名前:）       .BLKL    （数値）
```

記述規則

- 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。
  セクション定義の際に、セクション名に続けて",DATA"を記述することでセクションタイプが DATA タイプとなります。
- 指示命令と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。
- オペランドに整数値を記述してください。
- オペランドにはシンボルを記述できます。
- オペランドには式を記述できます。
- オペランドの式の値は、アセンブル実行時に確定しなければなりません。
- 領域にラベル名を定義する場合は、指示命令の前にラベル名を記述してください。ラベル名には、必ずコロン(:)を記述してください。

記述例

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

.BLKW

2 バイト長の領域確保

機能

- 2 バイト単位で、指定したバイト数の RAM 領域を確保します。
- 確保した RAM のアドレスに、ラベル名を定義することもできます。

記述形式

```
                .BLKW   （数値）
（名前:）       .BLKW   （数値）
```

記述規則

- 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。
  セクション定義の際に、セクション名に続けて",DATA"を記述することでセクションタイプが DATA タイプとなります。
- 指示命令と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。
- オペランドに整数値を記述してください。
- オペランドにはシンボルを記述できます。
- オペランドには式を記述できます。
- オペランドの式の値は、アセンブル実行時に確定しなければなりません。
- 領域にラベル名を定義する場合は、指示命令の前にラベル名を記述してください。ラベル名には、必ずコロン(:)を記述してください。

記述例

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

.BTEQU

ビットシンボル定義

機能

- ビット位置とメモリアドレスを定義します。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、1 ビット操作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。
- シンボリックデバッグでビットシンボルを使用できます。
- ビットシンボルは、グローバル指定ができます。

記述形式

（名前）　.BTEQU （ビット位置）,（アドレス値）
（名前）　.BTEQU （ビットシンボル）

記述規則

- 指示命令と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。
- ビット位置と、そのビットのメモリアドレスをカンマで区切って記述してください。
- 必ず、ビット位置を左に記述してください。
- 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。
- ビット位置は、必ずアセンブル実行時に確定する値を指定してください。
- オペランドのアドレス値の指定には、シンボルを記述できます。
- オペランドのアドレス値の指定には、アセンブル時未確定のラベル又はシンボルが記述できます。

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を外部参照（指示命令 '.BTGLB'のオペランドに記述）できません。

- オペランドには、式を記述できます。
- オペランドには、ビットシンボルを記述できます。

注意事項
ただし、オペランドのビットシンボル名は前方参照はできません。また、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。

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
```

.BTGLB

ビットシンボルグローバル宣言

機能

- 本指示命令で指定したビットシンボルが、グローバルシンボルであることを宣言します。
- 本指示命令で指定したビットシンボルで、ファイル内で定義されていないものは、外部のファイルで定義されているものとして処理します。
- 本指示命令で指定したビットシンボルで、ファイル内で定義されているものは、外部から参照できるように処理します。

記述形式

.BTGLB　（ビットシンボル名）
.BTGLB　（ビットシンボル名）[,（ビットシンボル名）...]

記述規則

- 指示命令と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。
- オペランドにグローバルシンボルとするビットシンボル名を記述します。

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。

- オペランドに複数のビットシンボル名を記述する場合は、カンマ(,)で区切って記述してください。

記述例

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

.BYTE

1 バイト長データを格納

機能

- 1 バイト長の固定データを ROM に格納します。
- データを格納したアドレスにラベルを定義することができます。

記述形式

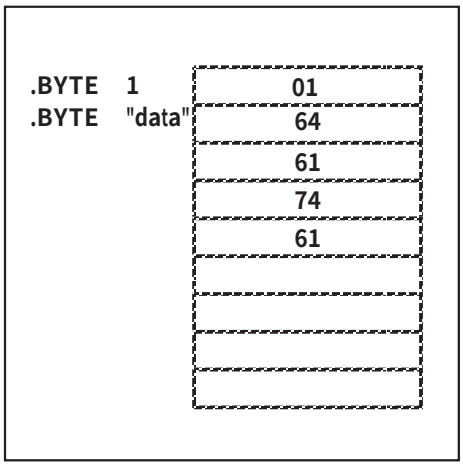
```
        .BYTE   （数値）
（名前:）   .BYTE   （数値）
```

記述規則

- オペランドに整数値を記述してください。
- 指示命令と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。
- オペランドにはシンボルを記述できます。
- オペランドには式を記述できます。
- 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、文字又は、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。
- ラベルを定義する場合には、指示命令の前にラベル名を記述してください。
- ラベル名には必ず、コロン(:)を記述してください。

記述例

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



.CALL

インスペクタ情報の関数呼び出し先情報を定義

機能

- 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。

記述形式

.CALL （呼び出し先関数（サブールーチン）名），（記憶クラス）

記述規則

- 本指示命令とオペランドの間には、必ずスペースまたはタブを記述してください。
- 呼び出し先関数（サブールーチン）名および記憶クラスは必ず記述してください。
- 記憶クラスを記述する場合は、カンマで区切って記述してください。
- 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。

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。

記述例

```
.INSF   glbfunc, G, 0
    :
jsr   glbsub
    .CALL   glbsub, G
    :
jsr   locsub
    .CALL   locsub, S
    :
.EINSF
```

.DEFINE

文字列シンボル定義

機能

- 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。

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。

記述形式

（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）"

記述規則

- 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。

記述例

```
                .SECTIONram,DATA
data1:          .BLKB       1
flag            .DEFINE     "#01H,data1"
                .SECTIONprogram
                CLB                     flag
```

.DOUBLE

8 バイト長データを格納

機能

- 8 バイト長の固定データを ROM に格納します。
- データを格納したアドレスにラベルを定義することができます。

記述形式

```
                .DOUBLE     （数値）
（名前:）       .DOUBLE     （数値）
```

記述規則

- オペランドに浮動小数点数を記述してください。
- 浮動小数点数の記述方法は、「オペランドの記述規則」を参照してください。
- 指示命令と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。
- ラベルを定義する場合には、指示命令の前にラベル名を記述してください。
- ラベル名には必ず、コロン(:)を記述してください。

記述例

```
                .DOUBLE   5E2
constant:       .DOUBLE   5e2
```

.EINSF

インスペクタ情報の関数終了を定義

機能

- 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。

記述形式

.EINSF

記述規則

- 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ラ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。

記述例

```
.INSF    glbfunc, G, 0
   :
.EINSF
```

.ELIF

条件アセンブル命令

機能

- 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。

記述形式

```
.IF　　条件式
ボディ
.ELIF　条件式
ボディ
.ENDIF
```

記述規則

- 本指示命令のオペランドには、必ず条件式を記述してください。
- 本指示命令と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。
- 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。

記述例

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

.ELSE

条件アセンブル命令

機能

- 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。

記述形式

```
.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
```

記述規則

- 本指示命令は、条件アセンブルブロック内に一つ以下記述できます。
- 本指示命令にオペランドはありません。

記述例

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

.END

アセンブリソースの終了宣言

機能

- 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。

記述形式

```
.END
```

記述規則

- 本指示命令は、一つのアセンブリ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、エラーの検出もしません。

記述例

```
.END
```

.ENDIF

条件アセンブル命令

機能

- 条件アセンブルブロックの終了を示します。

記述形式

```
.IF     条件式
        ボディ
.ENDIF
```

記述規則

- 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。

記述例

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

.ENDM

マクロ定義終了

機能

- 一つのマクロ定義のボディが終了する事を示します。

記述規則

- 必ず、指示命令".MACRO"に対応させて記述してください。

記述形式

（マクロ名）.MACRO
ボディ
.ENDM

記述例

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

    lda     0

    MOV.W   #0,A0
```

.ENDR

繰り返しマクロ終了

機能

・ 繰り返しマクロの終了を示します。

記述形式

[（ラベル）:] .MREPEAT　　（数値）
　　ボディ
　　.ENDR

記述規則

・ 必ず指示命令".MREPEAT"に対応させて記述してください。

記述例

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

    rep     3

    nop
    nop
    nop
```

.EQU

数値シンボル定義

機能

- 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。
- 本指示命令でシンボルを定義することにより、シンボリックデバッグ機能が使用できます。

記述形式

（名前）　.EQU　（数値）

記述規則

- シンボルに定義できる値は、アセンブル実行時に確定しなければなりません。
- 指示命令と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。
- シンボル定義のオペランドには、シンボルを記述できます。

注意事項

ただし、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。

- シンボル定義のオペランドには式を記述できます。
- シンボル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
```

.EXITM

マクロ展開の中止

機能

- 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。

記述形式

（マクロ名）.MACRO
ボディ
.EXITM
ボディ
.ENDM

記述規則

- マクロ定義のボディ内に記述してください。

記述例

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

    data1   0

    .IF     0 == 0
            .EXITM
    .ENDIF
    .ENDM
```

.FB

FB レジスタ値宣言

機能

- 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、FB 相対アドレッシングモードを指定できます。
- 指示命令".FBSYM"で指定されたラベル名を用いたニーモニックに対して、FB 相対アドレッシングモードでコードを生成します。

記述形式

.FB　　（数値）

記述規則

- 指示命令と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。
- アセンブリソースファイル内に必ず記述してください。
- FB 相対アドレッシングモードを記述する前に、必ず本指示命令を記述してください。
- オペランドには、0～0FFFFH の範囲の整数値が記述できます。

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、本指示命令の直前又は直後に次の命令を記述してください。
例）LDC　　#80H,FB

- オペランドにはシンボルが記述できます。

記述例

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

.FBSYM

FB 相対変位アドレッシングモード選択

機能

- 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。

記述形式

.FBSYM （名前）
.FBSYM （名前）[,（名前）...]

記述規則

- 指示命令と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。
- 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。
- 複数の名前を指定する場合は、名前をカンマで区切って記述してください。
- 本指示命令で指定したシンボルが、".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。

記述例

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

.FLOAT

4 バイト長データ格納

機能

- 4 バイト長の固定データを ROM に格納します。
- データを格納したアドレスにラベルを定義することができます。

記述形式

```
                .FLOAT  （数値）
（名前:）       .FLOAT  （数値）
```

記述規則

- オペランドに浮動小数点数を記述してください。
- 浮動小数点数の記述方法は、「オペランドの記述規則」を参照してください。
- 指示命令と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。
- ラベルを定義する場合には、指示命令の前にラベル名を記述してください。
- ラベル名には必ず、コロン(:)を記述してください。

記述例

```
                .FLOAT  5E2
constant:       .FLOAT  5e2
```

.FORM

リストファイルの行数と桁数を指定

機能

- 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リソースファイルの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、140 桁で出力します。

記述形式

.FORM　（行数）,（桁数）
.FORM　（行数）
.FORM　,（桁数）

記述規則

- １つのアセンブリソースファイルに複数回記述できます。
- 行数及び桁数にはシンボルを記述できます。

注意事項
前方参照となるシンボルは記述できません。

- 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。

記述例

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

.GLB

グローバル宣言

機能

- 本指示命令で指定したラベル及びシンボルがグローバルであることを宣言します。
- 本指示命令で指定したラベル及びシンボルで、ファイル内で定義されていないものは、外部のファイルで定義されているものとして処理します。
- 本指示命令で指定したラベル及び、シンボルで、ファイル内で定義されているものは、外部から参照できるように処理します。

記述形式

.GLB　（名前）
.GLB　（名前）[,（名前）...]

記述規則

- 指示命令と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。
- オペランドにグローバルラベルとするラベル名を記述します。
- オペランドにグローバルシンボルとするシンボル名を記述します。
- オペランドに複数のシンボル名を記述する場合は、カンマ(,)で区切って記述してください。

記述例

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

.ID

ID コードチェック機能の ID コードを設定

機能

- 指定した ID コードは、ID コード格納番地に各 8 ビットデータとして格納されます。また ROM コードプロテクト制御番地またはオプション機能選択レジスタに FFH を設定します。
- 設定した値はマップファイルおよび ID ファイルに出力します。
- 指定した ID コードは、アブソリュートモジュールファイル（.x30）へも出力されます。

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。

記述形式

.ID “（ ID コード文字列 ）”
.ID “#（ ID コード値 ）”

記述規則

- 指示命令とオペランドの間には、必ずスペースまたはタブを記述してください。
- ID コード文字列は、アスキーコードに変換して格納されます。
- ID コード文字列は、７文字以内で指定してください。
- ID コード値は先頭に'#'を付加してから指定してください。
- ID コード値は、数値のまま格納されます。
- ID コード値は、14 桁以内の整数値で指定してください。
- １つのアセンブリソースファイルに１度しか記述できません。

注意事項
複数のアセンブリソースファイルに本指示命令を記述した場合、LN30 でワーニングとなります。

文字列指定による記述例

```
; fixed vector section
;------------------------------------------------------------------
    .org  0FFFCh
RESET:
    .lword   start
    .id   “Code”      ; IDコード “Code”を設定します。
```

数値指定による記述例

```
; fixed vector section
;------------------------------------------------------------------
    .org  0FFFCh
RESET:
    .lword   start
    .id   “#20030401”      ; IDコード 20030401を設定します。
```

.IF

条件アセンブル命令

機能

- 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。

記述形式

```
.IF　　条件式
ボディ
.ENDIF
```

記述規則

- 本指示命令のオペランドには、必ず条件式を記述してください。
- 本指示命令と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。

条件式の機能

- 条件式の結果によって、条件アセンブルが行われます。

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。

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。

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。

- 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。

注意事項
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。

- 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(')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。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"　→　414243 < 434241 で真となります。
  "C" < "A"　→　43 < 41 で偽となります。
- 条件演算子の前後には、スペース又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。

条件式の記述例

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

条件アセンブル記述例

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

.INCLUDE

インクルードファイル指定

機能

- 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
  3. 環境変数 INC30 に設定されているディレクトリを検索

記述形式

.INCLUDE　　（ファイル名）

記述規則

- 指示命令と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。
- インクルードファイルのネスティングレベルは 9 レベルまでです。

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。

記述例

```
.INCLUDEinitial.a30
.INCLUDE..FILE@.inc
```

.INITSCT

セクション名の仮定義

機能

- 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。

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。

記述形式

.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

記述規則

- 指示命令とオペランドの間には、必ずスペースまたはタブを記述してください。
- セクション名とセクションタイプの間には、必ずカンマを記述してください。
- セクションタイプと align または noalign の間には、必ずカンマを記述してください。
- セクションタイプには、’CODE’、’ROMDATA’、’DATA’のいずれかを記述してください。
- align の指定がある場合、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２、４または８バイトアライメントに補正します。

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

        :
```

.INSF

インスペクタ情報の関数開始を定義

機能

- 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。

記述形式

.INSF　（関数（サブールーチン）開始ラベル名），（記憶クラス），（フレームサイズ）

記述規則

- 本指示命令とオペランドの間には、必ずスペースまたはタブを記述してください。
- 関数（サブールーチン）開始ラベル名、記憶クラスおよびフレームサイズは必ず記述してください。
- 記憶クラスおよびフレームサイズを記述する場合は、カンマで区切って記述してください。
- 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ラ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。

記述例

```
glbfunc:
.INSF    glbfunc, G, 0
    :
.EINSF

locfunc:
.INSF    locfunc, S, 0
    :
.EINSF
```

.INSTR

文字列の開始位置に置き換え

機能

- 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。

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。

記述形式

```
.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）}
```

記述規則

- オペランドは、必ず{}で囲ってください。
- 文字列、検出文字列及び検索開始位置は、必ず記述してください。
- 文字列、検出文字列及び検索開始位置は、カンマで区切って記述してください。
- カンマの前後には、スペース及びタブは記述できません。
- 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。

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。

- 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。

注意事項
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。

- 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。

```
top         .EQU    1

point_set   .MACRO  source,dest,top
point       .EQU    .INSTR{'source','dest',top}
    .ENDM
            :
    point_set japanese,se,1
            :
point       .EQU    7
```

.LEN

指定文字列の長さに置き換え

機能

・ オペランドに記述した文字列の文字列長を示します。

記述形式

.LEN {"（文字列）"}
.LEN {'（文字列）'}

記述規則

・ オペランドは、必ず{}で囲ってください。
・ 本指示命令とオペランドの間にスペース又はタブが記述できます。
・ 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。

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。

・ 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。

注意事項
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。

・ 本指示命令を式の項に記述できます。

記述例

```
bufset      .MACRO  f1,f2
buffer@f1:  .BLKB   .LEN{'f2'}
    .ENDM
            :
    bufset  1,Printout_data
    bufset  2,Sample
            :
    buffer1 .BLKB   13
    buffer2 .BLKB   6

buf         .MACRO  f1
buffer:     .BLKB   .LEN{"f1"}
    .ENDM

    buf     1,data  ; 'data' is not expanded.

    buffer  .BLKB   2
```

.LIST

リスト出力制御命令

機能

- アセンブラリストファイルへの行の出力を停止(OFF)することができます。
- 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。
- アセンブラリストファイルへの行の出力を開始(ON)することができます。
- 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。

記述形式

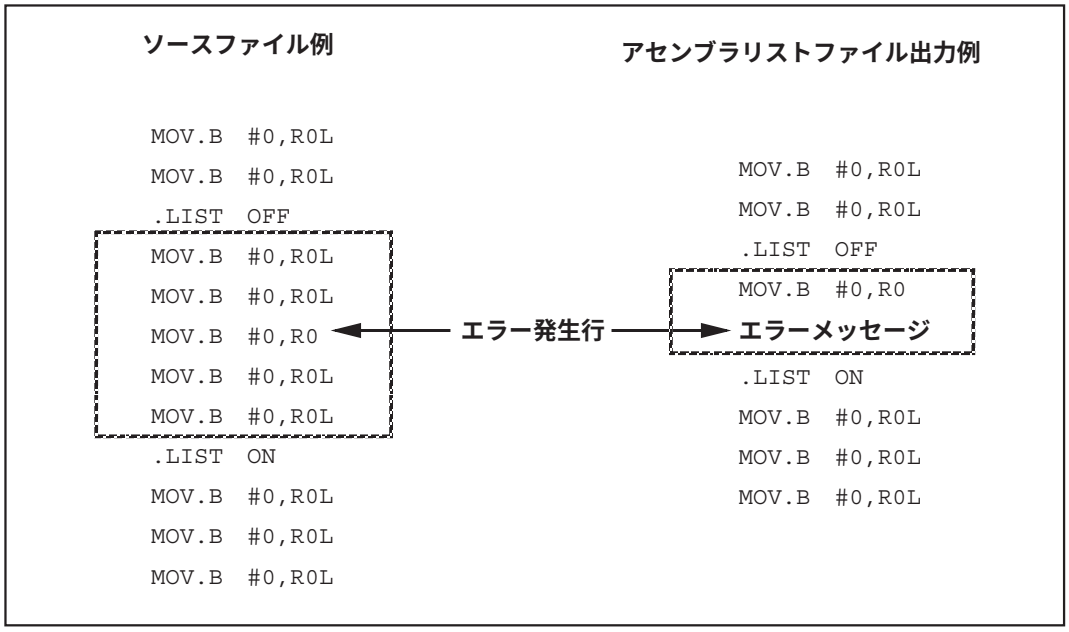
```
.LIST　　[ON|OFF]
```

記述規則

- 指示命令と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。
- 行の出力を停止する場合は、オペランドに'OFF'を記述してください。
- 行の出力を開始する場合は、オペランドに'ON'を記述してください。

記述例

```
.LIST　　ON
.LIST　　OFF
```



.LOCAL

マクロ内ローカルラベル宣言

機能

- 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。

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。

記述規則

.LOCAL　　（ラベル名）[,（ラベル名）...]

記述規則

- 本指示命令は、必ずマクロボディ内に記述してください。
- 本指示命令と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。
- 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
インクルードファイルの内容を含む、一つのアセンブリ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。

記述例

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

.LWORD

4 バイト長データ格納

機能

・4 バイト長の固定データを ROM に格納します。
・データを格納したアドレスにラベルを定義することができます。

記述形式

```
        .LWORD （数値）
（名前:）   .LWORD （数値）
```

記述規則

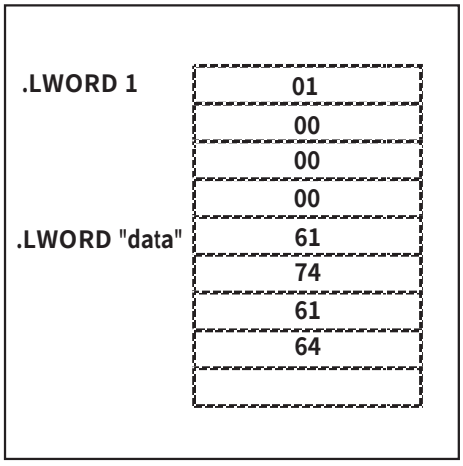
- オペランドに整数値を記述してください。
- 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- オペランドにはシンボルを記述できます。
- オペランドには式を記述できます。
- 複数のオペランドを記述するときは、カンマ(,)で区切って記述してください。
- オペランドにはクォーテーション(')又は、ダブルクォーテーション(")で囲って、文字又は、文字列を記述できます。このとき格納されるデータは、文字の ASCII コードになります。

注意事項
オペランドに記述できる文字列長は、４文字までです。

- ラベルを定義する場合には、指示命令の前にラベル名を記述してください。
- ラベル名には必ず、コロン(:)を記述してください。

記述例

```
.SECTIONvalue,ROMDATA
.LWORD          1
.LWORD          "data"
.LWORD          symbol
.LWORD          symbol+1
.LWORD          1,2,3,4,5
.END
```



.MACRO

マクロ定義

機能

- マクロ名を定義します。
- マクロ定義の始まりを示します。

記述形式

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）...]]

記述規則

- マクロ名は必ず記述してください。
- 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。
- オペランドには、仮引数が定義できます。
- 本指示命令とマクロ仮引数の間には、必ずスペース又はタブを記述してください。
- 本指示命令とマクロ名の間には、スペース又はタブを記述できます。

仮引数の記述規則

- 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。
- 仮引数を複数定義する場合は、仮引数をカンマ(,)で区切って記述してください。
- 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。

注意事項
ダブルクォーテーションで囲った文字列は、全てその文字列そのものを示します。仮引数をダブルクォーテーションで囲わないでください。

- 仮引数は８０個まで記述できます。

注意事項
１行に記述できる文字数の範囲内で最大８０個まで記述できます。

実引数の記述規則

- マクロ名と実引数の間には、必ずスペース又はタブを記述してください。
- 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。

注意事項

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

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)'
```

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

            mac     word,10,R0

  .IF       3=3
            .ELSE
                        MOV.W   #10,R0
            .ENDIF
  .ENDIF
            .ENDM
```

.MREPEAT

繰り返しマクロの開始

機能

- 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。

記述形式

```
[（ラベル）:] .MREPEAT　　（数値）
　　ボディ
　.ENDR
```

記述規則

- オペランドは必ず記述してください。
- 本指示命令と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。
- 本指示命令行の先頭にラベルを記述できます。
- オペランドには、シンボルを記述できます。

注意事項

前方参照となるシンボルは記述できません。

- オペランドには、式が記述できます。
- ボディには、マクロ定義及びマクロ呼び出しが記述できます。
- ボディ内に指示命令".EXITM"を記述できます。

記述例

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

    rep     3

    nop
    nop
    nop
```

.OFSREG

オプション機能選択レジスタに値を設定

機能

- 指定した値は、オプション機能選択レジスタに設定されます。
- 設定した値は、マップファイルに出力されます。
- オプション機能選択レジスタに設定された値は、アブソリュートモジュールファイル（.x30）へも出力されます。

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。

記述形式

.OFSREG　（ 数値 ）

記述規則

- 本指示命令は、“-R8C”オプションと組み合わせて使用してください。
- 指示命令とオペランドの間には、必ずスペースまたはタブを記述してください。
- オペランドには、0～0FFH の範囲の整数値が記述できます。
- オペランドにはシンボルが記述できます。
- １つのアセンブリソースファイルに１度しか記述できません。

注意事項
- 複数のアセンブリソースファイルに本指示命令を記述した場合、LN30 でワーニングとなります。
- “-R8C”オプションが指定されていない場合は、指示命令“.PROTECT”として処理されます。

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。
```

.OPTJ

最適化制御命令

機能

- 分岐命令の最適化を制御します。
- 無条件分岐命令及びサブルーチン呼び出し命令で、分岐距離指定子が省略され、かつオペランドが最適化対象外の命令について、分岐距離を指定できます。
- 本指示命令を記述した以降の行について、指定した内容が有効になります。
- 本指示命令による最適化の指定は、一つのアセンブリソースファイルに複数回記述できます。

注意事項
nc30 の"-OGJ(-Oglb_jmp)"、または as30 の"-JOPT"オプションを指定した場合、本指示命令は無視されます。

記述形式

.OPTJ　[OFF|ON],[JMPW|JMPA],[JSRW|JSRA]

記述規則

- 本指示命令のオペランドには、次の３つの項目が記述できます。
  1　分岐命令の最適化制御
  2　最適化対象外の無条件分岐命令選択
  3　最適化対象外のサブルーチン呼び出し命令選択

分岐命令の最適化制御	
OFF	分岐命令を最適化しません
ON（初期値）	分岐命令を最適化します
最適化対象外の無条件分岐命令選択	
JMPW	最適化対象外の無条件分岐命令を"JMP.W"で生成し
JMPA（初期値）	最適化対象外の無条件分岐命令を"JMP.A"で生成します
最適化対象外のサブルーチン呼び出し命令選択	
JSRW	最適化対象外のサブルーチン呼び出し命令を"JSR.W"で生成します
JSRA（初期値）	最適化対象外のサブルーチン呼び出し命令を"JSR.A"で生成します

- 各項目の指定順序は、任意です。
- 各項目は省略できます。省略した場合、分岐距離は初期値又は以前に指定した内容から変化しません。

記述例

```
.OPTJ    OFF
.OPTJ    ON
.OPTJ    ON,JMPW
.OPTJ    ON,JMPW,JSRW
.OPTJ    ON,JMPW,JSRA
.OPTJ    ON,JMPA
.OPTJ    ON,JMPA,JSRW
.OPTJ    ON,JMPA,JSRA
.OPTJ    ON,JSRW
.OPTJ    ON,JSRA
```

.ORG

アドレス宣言

機能

・　本指示命令を記述したセクションを絶対属性とします。

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。

・　本指示命令を記述したセクションのアドレスはアブソリュート値になります。
・　本指示命令を記述した直後の行から記述したニーモニックのコードが格納されるアドレスを決定します。
・　本指示命令の直後の行から記述した領域確保指示命令で、確保されるメモリのアドレスを決定します。

記述形式

.ORG　　（数値）

記述規則

・　本指示命令は、必ず、セクション指示命令の直後に記述してください。

注意事項

".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となります。

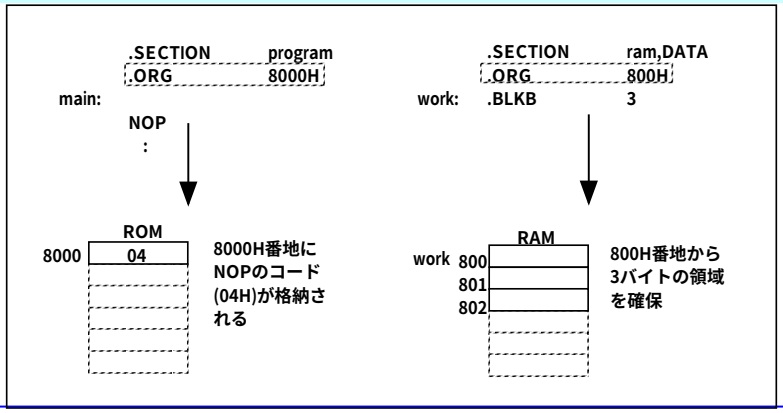
・　相対属性セクション内には、本指示命令は記述できません。
・　指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
・　オペランドに記述できる値は、0～0FFFFFH の範囲の数値です。
・　オペランドには式を記述できます。ただし、式の値がアセンブル実行時に確定する値でなければなりません。
・　オペランドにはシンボルを記述できます。ただし、式の値がアセンブル実行時に確定する値でなければなりません。
・　本指示命令は、相対属性指定を行ったセクション内には記述できません。
・　絶対属性セクション内であれば複数回記述できます。

記述例

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

次のような記述はエラーとなります。

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



.PAGE

リストファイル改ページ出力

機能

- アセンブラリストファイルを改ページします。
- オペランドに記述した文字列を改ページした際のヘッダ部分に出力します。

注意事項
ヘッダに出力できる最大文字数は（リストファイルの桁数）－65 文字です。リストファイルの桁数は、指示命令".FORM"で設定できます。

記述形式

.PAGE "（文字列）"
.PAGE '（文字列）'

記述規則

- 指示命令と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。
- オペランドは、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。
- オペランドは省略できます。

記述例

```
.PAGE
.PAGE       "strings"
.PAGE       'strings'
```

.PROTECT

ROM コードプロテクト制御番地に値を設定

機能

- 指定したプロテクトコードは、ROM コードプロテクト制御番地に設定されます。
- 設定した値はマップファイルに出力されます。
- ROM コードプロテクト制御番地に設定された値は、アブソリュートモジュールファイル(.x30)へも出力されます。

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。

記述形式

.PROTECT　（ 数値 ）

記述規則

- 指示命令とオペランドの間には、必ずスペースまたはタブを記述してください。
- オペランドには、0～0FFH の範囲の整数値が記述できます。
- オペランドにはシンボルが記述できます。
- １つのアセンブリソースファイルに１度しか記述できません。

注意事項
複数のアセンブリソースファイルに本指示命令を記述した場合、LN30 でワーニングとなります。

記述例

```
; fixed vector section
;--------------------------------------------------------------
    .org  0FFFCh
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。
```

.RVECTOR

ソフトウエア割り込みの設定

機能

- 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。
- 本指示命令に設定した内容は、可変ベクタテーブルに割り当てられます。
- 本指示命令に設定した内容は、リンカが生成するマップファイルに可変ベクタテーブル情報として出力されます。

注意事項

- 本指示命令を記述した場合、リンカで可変ベクタテーブルが自動生成されます。
  自動生成を行った結果、可変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ソフトウエア割り込み）が存在する場合、以下の優先順位で空き領域の値を設定します。
  (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
  なお、(1)(2)(3)のいずれも設定されていない場合は、空き領域には何も設定されません。
- 本指示命令を記述した場合、セクション名“vector”として 256 バイトの領域が確保されます。
- 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。

記述形式

.RVECTOR　ソフトウエア割り込み番号, ソフトウエア割り込み名

記述規則

- 指示命令とオペランドの間には、必ずスペースまたはタブを記述してください。
- ソフトウエア割り込み番号とソフトウエア割り込み名の間には、必ずカンマを記述してください。
- ソフトウエア割り込み番号は、アセンブル時に確定する値のみ記述できます。
- ソフトウエア割り込み番号は、0 から 63 の範囲で記述できます。
- ソフトウエア割り込み名には、シンボルまたはラベルが記述できます。

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。
```

.SB

SB レジスタ値宣言

機能

- 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。
- 以降の行で、SB 相対アドレッシングモードを指定できます。
- 指示命令".SBSYM"で指定されたラベル名を用いたニーモニックに対して、SB 相対アドレッシングモードでコードを生成します。

記述形式

.SB　　（数値）

記述規則

- 指示命令と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。
- アセンブリソースファイル内に必ず記述してください。
- SB 相対アドレッシングモードを使用する前に、必ず本指示命令を記述してください。
- オペランドには、0～0FFFFH の範囲の整数値が記述できます。

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、本指示命令の直前又は直後に次の命令を記述してください。

例）LDC　　#80H,SB

- オペランドにはシンボルが記述できます。

記述例

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

.SBBIT

SB 相対変位アドレッシングモード宣言

機能

- 本指示命令のオペランドに指定した名前に対して、SB 相対変位アドレッシングモードが選択されます。
- 1 ビット操作命令がショート形式を持つ場合は、11 ビット SB 相対変位アドレッシング又は 16 ビット SB 相対変位アドレッシングモードが選択されます。
- 1 ビット操作命令がショート形式を持たない場合は、8 ビット SB 相対変位アドレッシング又は 16 ビット SB 相対変位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 相対変位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 相対変位アドレッシングモードが選択されます。

記述形式

.SBBIT　（名前）
.SBBIT　（名前）[,（名前）...]

記述規則

- 指示命令と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。
- オペランドには'.BTEQU'及び'.BTGLB'で定義されたビットシンボルを記述できます。
- 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。
- 複数の名前を指定する場合は、名前をカンマで区切って記述してください。

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
```

.SBSYM

SB 相対変位アドレッシングモード指定

機能

- 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。
- リロケータブルな値をもつオペランドに対して、SB 相対アドレッシングモードを選択できます。

注意事項

本指示命令で指定されたラベル名を使って、.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。（記述例 2 の場合）

記述形式

.SBSYM （名前）
.SBSYM （名前）[,（名前）...]

記述規則

- 指示命令と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。
- オペランドには、ラベル及び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。
- 複数の名前を指定する場合は、名前をカンマで区切って記述してください。

記述例

例 1）

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

例 2）

次の例では、sym2 には、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_AUTO

.SB_AUTO / .SB_AUTO_S /.SB_AUTO_SBVAL / .SB_AUTO_SBSYM /.SB_AUTO_R / .SB_AUTO_E

SB 相対アドレッシングの自動生成

機能

- 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。

.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。

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。

記述形式

```
.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
```

コンパイラ出力例

```
        .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
```

.SECTION

セクション定義

機能

- セクション名を定義します。
- セクションの始まりを定義します。一つのセクション指示命令から、次のセクション指示命令又は指示命令".END"までを一つのセクションとして定義します。
- セクションタイプを定義します。
- 'ALIGN'指定がある場合、ln30 がセクションの始まりを偶数番地に割り当てます。
- ALIGN 指定をしたセクション又は絶対属性セクション内では、指示命令".ALIGN"が記述できます。

記述形式

```
.SECTION　　（セクション名）
.SECTION　　（セクション名）,（セクションタイプ）
.SECTION　　（セクション名）,（セクションタイプ）,ALIGN
.SECTION　　（セクション名）,ALIGN
```

記述規則

- セクション名は必ず記述してください。
- メモリ領域を確保したり、メモリにデータを格納するアセンブリ指示命令を記述する場合と、ニーモニックを記述する場合は必ず、本指示命令でセクションを定義してください。
- セクションタイプと ALIGN は、セクション名の後に記述してください。
- セクションタイプ及び、ALIGN 指定をする場合は、カンマで区切って記述してください。
- セクションタイプと ALIGN の記述順序は任意です。
- セクションタイプは、'CODE','ROMDATA','DATA'のいずれかを記述できます。
- セクションタイプは省略できます。このとき、as30 はセクションタイプを CODE として処理します。

記述例

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

.SJMP

ショートジャンプ命令生成制御

機能

- ショートジャンプ命令の生成を制御します。
- ".SJMP OFF"を記述した行以降でショートジャンプ命令を生成しません。
- ".SJMP ON"を記述した行以降でショートジャンプ命令を生成します。

記述形式

```
.SJMP    ON
.SJMP    OFF
```

記述規則

- 本指示命令と'ON'又は'OFF'の間には必ずスペース又はタブを記述してください。

記述例

```
                :
    .SJMP       ON
    JMP                 lab         ; SJMP Enable
    NOP
    .SJMP       OFF
    JMP                 lab         ; SJMP Disable
    NOP
lab:
                :
```

.STK

インスペクタ情報のスタック情報定義

機能

・ インスペクタ情報のスタック情報を定義します。

記述形式

.STK　スタックサイズ

記述規則

・ 本指示命令とオペランドの間には、必ずスペースまたはタブを記述してください。
・ スタックサイズは必ず記述してください。
・ 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。

記述例

```
.INSF    glbfunc, G, 0
    :
    .STK     2          ;2 byte push
    jsr    glbsub
    .STK     -2         ;2 byte pop
    :
.EINSF
```

.SUBSTR

文字列の切り出し

機能

・ 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。

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。

記述形式

.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）}

記述規則

・ オペランドは、必ず{}で囲ってください。
・ 文字列、切り出し開始位置及び切り出し文字数は、必ず記述してください。
・ 文字列、切り出し開始位置及び切り出し文字数は、カンマで区切って記述してください。
・ 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。

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。

・ 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。

注意事項
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。

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 文字づつ、".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"
```

.SVECTOR

スペシャルページの設定

機能

- 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。
- 本指示命令に設定した内容は、スペシャルページベクタテーブルに割り当てられます。
- 本指示命令に設定した内容は、リンカが生成するマップファイルにスペシャルページベクタテーブル情報として出力されます。

注意事項

- 本指示命令を記述した場合、リンカでスペシャルページベクタテーブルが自動生成されます。
- 本指示命令を記述した場合、セクション名“svector”が定義されます。
- 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。
- 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）

記述形式

.SVECTOR　スペシャルページ番号, スペシャルページ名

記述規則

- 指示命令とオペランドの間には、必ずスペースまたはタブを記述してください。
- スペシャルページ番号とスペシャルページ名の間には、必ずカンマを記述してください。
- スペシャルページ番号は、アセンブル時に確定する値のみ記述できます。
- スペシャルページ番号は、18 から 255 の範囲で記述できます。
- スペシャルページ名には、シンボルまたはラベルが記述できます。

記述例

```
.svector  250, __SPECIAL_250      ; __SPECIAL_250をスペシャルページ番号の250番に
                                    設定します。
```

.VER

指定文字列をマップファイルへ出力

機能

- 指定した文字列を ln30 が生成するマップファイルへ出力するようにリロケータブルモジュールファイルに出力します。
- マップファイルには指定した全ての文字列が出力されます。
- リロケータブルモジュールファイル毎に、ユーザーの指定する情報をマップファイルに出力できます。

記述形式

.VER　　"（文字列）"
.VER　　'（文字列）'

記述規則

- 指示命令と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。
- オペランドに、出力させたい文字列をクォーテーション(')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。
- オペランドは、一行の範囲内で記述してください。
- １つのアセンブリソースファイルに１度しか記述できません。。
- 指示命令".END"以前であれば任意の行に記述できます。

記述例

```
.VER        'strings'
.VER        "strings"
```

.WORD

2 バイト長データを格納

機能

- 2 バイト長の固定データを ROM に格納します。
- データを格納したアドレスにラベルを定義することができます。

記述形式

```
        .WORD   （数値）
（名前:）   .WORD   （数値）
```

記述規則

- オペランドに整数値を記述してください。
- 指示命令と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。
- オペランドにはシンボルを記述できます。
- オペランドには式を記述できます。
- 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、文字又は、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。

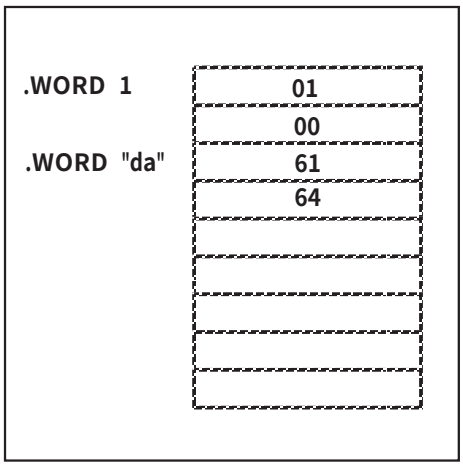
注意事項

オペランドに記述できる文字列長は、２文字までです。

- ラベルを定義する場合には、指示命令の前にラベル名を記述してください。
- ラベル名には必ず、コロン(:)を記述してください。

記述例

```
.SECTIONvalue,ROMDATA
.WORD   1
.WORD   "da","ta"
.WORD   symbol
.WORD   symbol+1
.WORD   1,2,3,4,5
.END
```



?

テンポラリラベル

機能

- テンポラリラベルを定義します。
- 直前又は直後に定義されたテンポラリラベルを参照します。

注意事項

参照できるラベルは、直前又は直後のラベルだけです。

- 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。

記述形式

?:

（ニーモニック） ?+

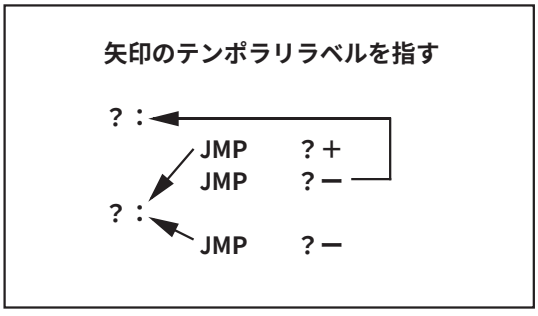
（ニーモニック） ?-

記述規則

- 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。

記述例

```
?:
    JMP     ?+
    JMP     ?-
?:
    JMP     ?-
```



@

文字列の連結

機能

- 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。

記述形式

（文字列）@（文字列）
（文字列）@（文字列）[@（文字列）...]

記述規則

- 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。

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]
```

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。

代入文

右辺を左辺に代入します。

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。

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。

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。

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。

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。

BREAK 文

BREAK 文は該当する FOR 文、DO 文又は 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。

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。

FOREVER 文

FOREVER 文は該当する FOR 文及び 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。

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。

レジスタ変数

M16C ファミリが持っているレジスタを示します。

フラグ変数

M16C ファミリのファンクションフラグを示します。

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。

メモリ変数

任意のラベル又はシンボルを示します。

メモリビット変数

任意のビットシンボルを示します。

予約変数

AS30 の構造化記述では、レジスタ変数、フラグ変数及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。レジスタ及びフラグの機能の詳細については、M16C ソフトウェアマニュアルを参照してください。

レジスタ変数

次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。

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	バイト型
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又は割り込みスタックポインタ）が対象となります。スタックポインタ及び U フラグの機能についての詳細は、「M16C ファミリ ソフトウエアマニュアル」を参照してください。

スタック変数

次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。

スタック変数名	内容
[STK]	スタックポインタで示されるメモリ

スタック領域への退避又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。

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によって、U フラグが 1 のときユーザスタックポインタによって示されます。

フラグ変数

次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 ファミリ ソフトウエアマニュアル」を参照してください。

フラグ変数	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

レジスタビット変数

次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。

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
```

構造化記述文のオペランド

```
if BITR0_1    ;Test bit 1 of register0
   :
else
   :
endif

if BITR0_2    ;Test bit 2 of register0
   :
else
   :
endif
```

メモリ変数

AS30 の構造化記述では、ラベル及びシンボルをメモリ変数として処理します。

as30 は、メモリ変数名の大文字と小文字を区別します。

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及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。

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	外部参照ラベル及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。

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。

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]]

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[ ]又は{ }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又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[ ]又は{ }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)

```
    [ label ] = 10
```

記述例 3)

```
    if [ label[SB] ]
                :
    else
                :
    endif
```

サイズ指定子

サイズ指定子は、メモリ変数及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。

次に記述できるサイズ指定子の一覧を示します。

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。

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又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   ROL,LAB_W
                :
    .END
```

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。

アセンブラ指示命令	変数の型
.BTEQU, .BTGLB	ビット型

メモリビット変数のアドレッシングモード

次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'はビット番号を、'addr'はメモリアドレスを示します。

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及び相対アドレッシングは記述できません。

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[ ]又は{ }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又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[ ]又は{ }で囲って記述してください。

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
```

例 2)　外部参照メモリビット変数の場合

```
   .BTGLB    BITSYM
   if [ BITSYM ]
                :
   else
                :
   endif
```

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。

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

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

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。

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

演算子の属性

二項演算子の加減算及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又はタブは記述できません。

属性	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

式

式には次に示す種類があります。

単項式

一つの項から成るもの及び一つの項に単項演算子を組み合わせた式

二項式

二つの項と演算子から成る式

複合式

単項式又は式を論理演算子で組み合わせた式

式の項

項には次に示すものが記述できます。

変数

レジスタ変数、フラグ変数、レジスタビット変数、メモリ変数及びメモリビット変数が記述できます。

定数

オペランドに記述可能な数値が記述できます。
乗算、剰余算では、次に示す範囲の定数の演算ができます。

注意事項

二項の除算及び剰余算を除いて、異なった型を持つ変数を用いた式は記述できません。

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
```

式の記述例

式の記述例を、式の種類ごとに示します。"mem"及び"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]
```

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。

条件式

- 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)"及び"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又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（255 文字以内）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"及び"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
```

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              ; Nested for state and if state
        :
next
endif                   ; Nested for state and if state
```

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。

IF 文

条件分岐文

IF - ENDIF

- IF 文の基本構成は、構造化記述命令'IF'、'ENDIF'及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 条件式
    制御ブロック
ENDIF
```

機能

- IF の条件が偽のとき、ENDIF に分岐します。
- ENDIF に対してシステムラベルが生成されます。

記述規則

- 'IF'と条件式の間には必ずスペース又はタブを記述してください。
- 条件式には、「条件式」の項で示した式が記述できます。

ELSE

機能

- 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。
- ELSE に対してシステムラベルが生成されます。

記述規則

- 'ELSE'は、'IF'と'ENDIF'の間に 1 つだけ記述できます。

```
IF 条件式
    制御ブロック
ELSE
    制御ブロック
ENDIF
```

ELIF

機能

- 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 に対してシステムラベルが生成されます。
- 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 又は 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。

記述規則

- ELIF と条件式の間には必ずスペース又はタブを記述してください。
- 'ELIF'は、'IF'と'ELSE'の間又は'IF'と'ENDIF'の間に、1 つ以上記述できます。

```
IF 条件式
    制御ブロック
ELIF 条件式
    制御ブロック
ELSE
    制御ブロック
ENDIF
```

ソース記述例

```
    IF [ sym1 ] == 10       ; value of sym1 equal 10?
                :               ; Yes
ELIF [ sym2 ] != 10   ; value of sym2 equal 10?
                :               ; No
ELSE
                :               ; Other
ENDIF
```

展開例

```
            CMP.B     #10,sym1
            JNE       ..IF0002
                :
            JMP       ..IF0003
    ..IF0002:
            CMP.B     #10,sym2
            JEQ       ..IF0004
                :
            JMP       ..IF0003
    ..IF0004:
                :
    ..IF0003:
```

FOR-STEP 文

繰り返し文（回数指定）

基本構成

・ FOR 文の基本構成は、構造化記述命令'FOR'、'NEXT'及びこれらの命令で囲まれた制御ブロックから成ります。

```
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
```

機能

・ 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。

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。

・ 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。
・ FOR 及び NEXT に対してシステムラベルが生成されます。
・ 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。

記述規則

・ 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。

注意事項

ループカウンタに使用しているレジスタ変数及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。

・ 制御ブロック内の任意の行に、'BREAK'文が記述できます。
・ 制御ブロック内の任意の行に、'CONTINUE'文が記述できます。
・ 初期値及び最終値には、変数又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。

FOREVER

FOREVER 命令は、繰り返しの終了条件がないことを示します。

```
FOR FOREVER
    制御ブロック
NEXT
```

機能

・ 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。

記述規則

・ 制御ブロック内の任意の行に、'BREAK'文が記述できます。
・ 制御ブロック内の任意の行に、'CONTINUE'文が記述できます。
・ 初期値及び最終値には、変数又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。

ソース記述例

- LAB を 0 に初期化し 10 まで繰り返す

```
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:
```

FOR-NEXT 文

繰り返し文（条件指定）

基本構成

・ FOR 文の基本構成は、構造化記述命令'FOR'、'NEXT'及びこれらの命令で囲まれた制御ブロックから成ります。

```
FOR  条件式
     制御ブロック
NEXT
```

機能

・ 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。
・ FOR 及び NEXT に対してシステムラベルが生成されます。
・ 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。このときの機能と記述規則は、FOR-STEP 文と同様です。

記述規則

・ FOR と条件式の間には必ずスペース又はタブを記述してください。
・ 制御ブロック内の任意の行に、'BREAK'文が記述できます。
・ 制御ブロック内の任意の行に、'CONTINUE'文が記述できます。

ソース記述例

・ R0 が 10 よりも小さい間繰り返し

```
FOR R0 <.S 10
        :
NEXT
```

展開例

```
..fr0000:
        CMP.W    #10,R0
    JGE      ..fr0002
        :
    JMP      ..fr0000
..fr0002
```

SWITCH 文

多条件分岐文

基本構成

- SWITCH 文の基本構成は、構造化記述命令'SWITCH'、'ENDS'、'CASE'及び'CASE'で区切られる制御ブロックから成ります。

```
SWITCH　　式
CASE　　データ
　制御ブロック
CASE　　データ
　制御ブロック
ENDS
```

機能

- CASE および ENDS に対応するシステムラベルが生成されます。
- 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。

記述規則

- SWITCH のオペランドの式には、「6.4　式」の項で示した単項式及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、ワ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。

BREAK

制御ブロックの最後に'BREAK'文を記述できます。

```
SWITCH 式
CASE データ
　制御ブロック
　BREAK
CASE データ
　制御ブロック
ENDS
```

機能

- 'BREAK'文は無条件で'ENDS'に分岐します。

記述規則

- 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ワ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。

DEFAULT

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。

```
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
```

機能

- 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 に対してワ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。

記述規則

- 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。

ソース記述例

```
SWITCH [ work ]
    CASE 1
            :
        BREAK
    CASE 2
            :
    DEFAULT
            :
ENDS
```

展開例

```
        CMP.B    #1,WORK        ; for "CASE 1"
        JNE      ..sw0004       ; for "CASE 1"
                 :
        JMP      ..sw0000       ; for BREAK
..sw0004:                       ; for "CASE 2"
        CMP.B    #2,WORK        ; for "CASE 2"
        JNE      ..sw0006       ; for "CASE 2"
                 :
..sw0006:                               ; for DEFAULT
                 :
..sw0000:                       ; for ENDS
```

DO 文

繰り返し文（条件指定）

基本構成

- 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。

```
DO
    制御ブロック
WHILE  条件式
```

機能

- 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。
- DO 及び WHILE に対してラベルが生成されます。

記述規則

- WHILE と条件式の間には必ずスペース又はタブを記述して下さい。
- 条件式には、「条件式」で示した式が記述できます。

ソース記述例

```
DO
      :
WHILE [lab].b ==1
```

展開例

```
..DO0000:
        :
      CMP.B   #1,lab
      JEQ     ..DO0000
..DO0002:
```

BREAK 文

無条件分岐

機能

- 無条件分岐命令を生成します。

記述規則

- 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。

ソース記述例

```
    FOR [lab]=1 TO 10 STEP 1
            :
          BREAK
            :
NEXT
```

展開例

```
        MOV.W     #1,lab                ; for FOR
..fr0000:                       ; for FOR
        CMP.W     #10,lab               ;for   FOR
        JEQ       ..fr0002              ; for FOR
            :
        JMP       ..fr0002      ; for BREAK
            :
..fr0001:                       ; for STEP
        ADD.W     #1,lab    ; for STEP
        JMP       ..fr0000      ; for STEP
..fr0002:                       ; for NEXT
```

CONTINUE 文

無条件分岐（条件判断文へ）

機能

- 無条件分岐命令を生成します。

記述規則

- 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 文及び SWITCH 文の制御ブロック内に記述できます。
- 通常の IF 文及び SWITCH 文の制御ブロック内には記述できません。

ソース記述例

```
        FOR [lab]=1 TO 10 STEP 1
            :
          CONTINUE
            :
NEXT
```

展開例

```
        MOV.W     #1,lab                ; for FOR
..fr0000:                               ; for FOR
        CMP.W     #10,lab   ; for FOR
        JEQ       ..fr0002      ; for FOR
            :
        JMP       ..fr0001      ; for CONTINUE
            :
..fr0001:                     ; for STEP
        ADD.W     #1,lab    ; for STEP
        JMP       ..fr0000              ; for STEP
..fr0002:                     ; for NEXT
```

FOREVER 文

繰り返しの条件を常に真とする

機能

- 無条件分岐命令を生成します。

記述規則

- FOREVER 文は、FOR 及び DO の条件式に記述できます。
- FOREVER を記述した条件式は、常に真となります。

FOR　　FOREVER
WHILE　　FOREVER

ソース記述例

```
FOR     FOREVER
            :
NEXT
```

展開例

```
        ..fr0000:
    :
                    JMP     ..sfr0000
     ..fr0002:
```

代入文

右辺を左辺に代入する

基本構成

- 代入命令（=）と左辺及び右辺から成ります。

機能

- 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。

演算子	機能
=	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。

代入文の記述規則

- 代入文'=.S','=.Z','=.EL'及び'=.ES'の右辺には単項演算子及び二項演算子を含む式は記述できません。
- 代入文'=.S','=.Z'の左辺及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'の左辺及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'及び'src'に記述できる内容です。
- 代入文'=.ES'の左辺及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'及び'src'に記述できる内容です。

注意事項
ニーモニックの詳細については、「M16C ファミリ　ソフトウエアマニュアル」を参照してください。

- 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、ワ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及び二項演算子を含む式は記述できません。

代入文（=）に記述できる変数型の組み合わせ

左辺	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

左辺	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"又は"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"又は"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

左辺	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

左辺	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

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 R0L	EXTS.B R0L
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 R0L	MOV.B #0,R0H
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
R0L =.EL [lab_b]	LDE.B lab_b,R0L
[lab_w].W =.ES R0	STE.W R0.lab_w

構造化記述命令の構文

AS30 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及び演算子が記述できます。

レジスタ変数

レジスタ変数は、次に示す３つに分類されます。

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又は割り込みスタックポインタ）が対象となります。スタックポインタ及び U フラグの機能についての詳細は、「M16C ファミリ ソフトウエアマニュアル」を参照してください。

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

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

数値又はアセンブル時に確定する式の値

単純代入文及び単項演算子を含んだ代入文の構文

注意事項
"=.S"及び"=.Z"における"regb"及び"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, regb =.EL memb
memw, regw =.EL memw
memb =.ES memb,regb
memw =.ES memw,regw
memw, regw = regc
memb,regb = dsp:8[SP]
memw,regw = dsp:8[SP]
mema, [A0.A], [A1.A], R2R0, R3R1, A1A0 = regpc
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

左辺がスタック変数の代入文

[STK].B = 定数
[STK].B = memb
[STK].B = regb（A0.B 及び 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

- 次に示すレジスタ及びスタック
  FB,SB,SP,ISP,FLG,INTBH,INTBL,INTB,IPL 及び[STK]
- 乗算の結果が 32 ビットになる式
- レジスタビット変数及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]

IF 文の構文

```
IF 式1
   :
ENDIF
```

FOR-STEP 文の構文

```
FOR変数= [... ][変数|定数]  TO [変数|定数] STEP 定数
   :
NEXT
```

FOR-NEXT 文の構文

```
FOR 式1
   :
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
```

M16C シリーズ、R8C ファミリ用　C コンパイラパッケージ V.5.45
アセンブラユーザーズマニュアル

発行年月日:　2010 年 4 月 1 日　Rev.2.00

発行:　ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部 1753 〒211-8668

編集:　株式会社ルネサス ソリューションズ

M16C シリーズ、R8C ファミリ用
C コンパイラパッケージ V.5.45
アセンブラ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
神奈川県川崎市中原区下沼部1753 〒211-8668

RJJ10J2501-0200