評価ボード用Ｈ８Ｓ，Ｈ８／３００
クロスアセンブラシステム
ユーザーズマニュアル

クロスアセンブラ編

1. 起動方法

アセンブラを起動するコマンドラインの形式は、次のとおりです。

```
asm38△入力ファイル名[,入力ファイル名･･･][[△]－オプション名･･･]
```

（1） 起動コマンド

アセンブラの起動コマンドとして ”asm38” を入力します。

（2） 入力ファイル名

アセンブラの入力となるソースファイル名を指定します。複数のファイルを指定する場合は、カンマ(,)で区切って指定します。この場合、.END 制御命令は最後に指定するソースファイル内に指定してください。アセンブラは、それらのソースファイルに含まれるソースステートメントをアセンブルし、１つのオブジェクトファイルに出力します。

ソースファイル名にファイル形式の指定がない場合は、アセンブラが自動的に ”mar” または”src”をファイル形式と仮定してファイルを入力します。

（3） オプション名

オプション名は、必ず先頭にハイフン（-）をつけて指定してください。また、オプション名と入力ファイル名またはオプション名同士は、連続して指定しても、1つ以上のスペースまたはタブで区切っても構いません。

オプション名の一覧については、**「4．オプション一覧」**を参照してください。

2. コーディング規則

2.1. ソースステートメント

ソースプログラムを「文章」に例えると、１つの「文」にあたるのがソースステートメントです。そのソースステートメントを構成する「単語」として、予約語（レジスタ名や演算子）やシンボルなどがあります。

2.1.1. ソースステートメントの構成

ソースステートメント内の構成を、以下に示します。

［ラベル］［△オペレーション［△オペランド］］［コメント］

コーディング例

LABEL1:	MOV.L	@R0 , R1	; ｿｰｽｽﾃｰﾄﾒﾝﾄの例です。
ﾗﾍﾞﾙ	ｵﾍﾟﾚｰｼｮﾝ	ｵﾍﾟﾗﾝﾄﾞ	ｺﾒﾝﾄ

（１） ラベル

ソースステートメントにつける名札として、シンボルを書きます。シンボルとは、プログラマが定義する名前です。

（２） オペレーション

オペレーションの種類には、実行命令、制御命令があります。

（３） オペランド

オペレーションの実行対象となるものを書きます。

オペランドの個数と種類は、オペレーションによって決まります。オペランドを必要としないオペレーションもあります。

（４） コメント

プログラムを分かりやすくするための注釈を書きます。

2.1.2. ソースステートメントの書き方

ソースステートメントは、ASCII 文字で記述します。ただし、文字列またはコメントの中には、シフト JIS コードを記述できます。

１行の最大長は、255 バイトです。

（1） ラベルの書き方

ラベルは、次のように書きます。

・１カラム目から書き始める。

または

・ラベル名の直後にコロン（：）をつける。

（2） オペレーションの書き方

オペレーションは、次のように書きます。

・ラベルを記述していない場合、２カラム目以降から書き始める。

・ラベルを記述している場合、ラベルとの間に、１つ以上の空白またはタブを置いて書き始める。

（3） オペランドの書き方

オペランドは、オペレーションとの間に、１つ以上の空白またはタブを置いて書き始めます。

（4） コメントの書き方

コメントは、セミコロン（；）の後に書きます。

アセンブラは、セミコロンから行末までをコメントと見なします。

2.1.3. 複数行にわたるソースステートメントの書き方

次のようなときに、プログラムを見やすくするため、１つのソースステートメントを複数の行に分けて書くことができます。

・ソースステートメントが長くなる場合

・オペランドの１つ１つにコメントをつけたい場合

■コーディング例

```
        .DATA.L         H'FFFF0000  ,     ;初期値１
+                       H'FF00FF00  ,     ;初期値２
+                       H'00FF00FF        ;初期値３
```

・オペランドとオペランドを区切るカンマ（, ）を切れ目として改行します。

・すぐ次の行の１カラム目にプラス（＋）を書きます。

・そのプラスの後ろに、続きのソースステートメントを書きます。

2.2. シンボル

2.2.1. シンボルの役割

シンボルは、プログラマが定義する名前であり、次の役割を果たします。

・アドレスシンボル：データの格納場所、分岐先などのアドレスを表します。
・定数シンボル：定数を表します。
・レジスタ別名：汎用レジスタを表します。
・セクション名：セクション★の名前を表します。

【注】★ セクションとは、プログラムの１区切りであり、リンケージ処理を実行する単位です。

2.2.2. シンボルの名づけ方

（１）シンボルに使用できる文字

次の ASCII 文字を使用できます。

・英大文字、英小文字(A～Z，a～z)
・数字(0～9)
・アンダースコア（_)
・ドル($)

アセンブラは、シンボル中の英大文字と英小文字を区別します。

（２）先頭の文字

次のいずれかに限ります。

・英大文字、英小文字(A～Z，a～z)
・アンダースコア（_)
・ドル（$)

【注意】 ドル($)1 文字は、ロケーションカウンタを表す予約語です。

（３）最大文字数

32 文字です。

アセンブラは、32 文字を超える部分を無視します。

（４）シンボルとして使用できない名前

レジスタ名、演算子またはドル($)は、シンボルとして使用できません。また、以下に示すような名前は、アセンブラが内部シンボル★として使用します。プログラマが同じ名前を使うことはできません。

_$$nnnnn （nnnnn は数字(0～9)です）

【注】★ 内部シンボルとは、アセンブラの内部処理のため必要なシンボルです。内部シンボルは、アセンブルリストやオブジェクトモジュールには、出力されません。

2.3. ロケーションカウンタ

ロケーションカウンタは、オブジェクトコード(実行命令やデータをコンピュータが理解できる形式に変換したコード)を配置するアドレス(ロケーション)を指し示します。

ロケーションカウンタの値(以下、ロケーションカウンタ値と称します)は、オブジェクトコードの出力に応じて自動的に変化します。

また、.ORG, .ALIGN 制御命令により、意図的にロケーションカウンタ値を変えることもできます。

2.4. 演算子

2.4.1. 演算子の種類

表 2.1 に、演算子の一覧を示します。

表 2.1　演算子一覧

演算区分	演算子	演算内容	書き方
算術演算	+	単項プラス	＋項
	-	単項マイナス	－項
	+	加算	項 1 ＋ 項 2
	-	減算	項 1 － 項 2
	*	乗算	項 1 ＊ 項 2
	/	除算	項 1 ／ 項 2
論理演算	～	単項否定	～項
	&	論理積	項 1 & 項 2
	|	論理和	項 1 | 項 2
	～	排他的論理和	項 1 ～ 項 2
セクション集合演算	STARTOF	セクション集合の先頭アドレスを求める	STARTOF セクション名
	SIZEOF	セクション集合のサイズをバイト単位で求める	SIZEOF　セクション名
抽出演算	HIGH	上位バイト抽出	HIGH　項
	LOW	下位バイト抽出	LOW　項
	HWORD	上位ワード抽出	HWORD　項
	LWORD	下位ワード抽出	LWORD　項

2.4.2. 演算の順序

表 2.2 に、演算子の優先順位と結合規則を示します。

表 2.2 演算子の優先順位と結合規則

優先順位	演算子	結合規則
1 (高)	+ - ～ STARTOF SIZEOF HIGH LOW HWORD LWORD *	右から左の順に演算を処理する。
2	* /	左から右の順に演算を処理する。
3	+ -	左から右の順に演算を処理する。
4	&	左から右の順に演算を処理する。
5 (低)	| ～	左から右の順に演算を処理する。

【注】* 優先順位 1 の演算子は単項演算子です。

2.5. セクション

本クロスシステムでは、プログラムを「セクション」という単位で区切って作成します。

（1）セクションの意味と種類

ソースプログラムを「文章」に例えると、１つの「章」にあたるのがセクションです。セクションは、リンケージエディタがオブジェクトモジュールをリンケージするときの処理単位です。

セクションには次の種類があります。

・コードセクション
　おもに、実行命令、初期値を持つデータを記述します。

・データセクション
　おもに、データを記述します。初期値を持つデータの場合、リンケージ処理で ROM 化が必要です。

・コモンセクション
　データセクションと同じですが、ソースファイル間で共通に使用されるデータを記述します。リンケージ処理で同じアドレスに割り付けられます。

・スタックセクション
　スタック領域として、初期値を持たないデータを記述します。

・ダミーセクション
　データ構造を記述します。初期値を持たないデータを記述します。実行命令、初期値を持つデータを記述することはできません。ダミーセクションはオブジェクトモジュールに出力されません。

（2）絶対アドレスセクションと相対アドレスセクション

メモリへの配置方法の違いからセクションを考えると、絶対アドレスセクションと相対アドレスセクションに大別できます。

・絶対アドレスセクション

絶対アドレスセクションのメモリ上での配置は、ソースプログラムで指定します。絶対アドレスセクション内のロケーションは、メモリ上の配置そのものを示すアドレスです。

絶対アドレスセクションは、.SECTION 制御命令で LOCATE オペランドを指定することによって宣言します。

・相対アドレスセクション

相対アドレスセクションのメモリ上での配置は、ソースプログラムでは指定せず、リンケージのときに指定します。相対アドレスセクション内のロケーションは、セクション先頭からの相対距離を表すアドレスです。

相対アドレスセクションは、.SECTION 制御命令で LOCATE オペランドを指定しないことによって宣言します。

相対アドレスセクションをメモリ上に配置する先頭アドレスは、リンケージ処理の、START オプション／サブコマンドで指定します。また、リンケージ処理では、別々にアセンブルしたソースプログラムの同じ名前の相対アドレスセクションを連続した領域に結合します。

（3）セクション集合の演算

リンケージ処理をした後で結合されたセクション全体をセクション集合と呼びます。セクション集合には、SIZEOF と STARTOF の２つの演算子があり、それぞれセクション集合のバイト数と、セクション集合の先頭アドレスを参照します。

2.6. コーディング例

以下、本章では、コーディング例を示します。

例 1）

```
        .CPU        2600A
;
        .IMPORT   SUBRTN
        .IMPORT   DATA1
;
P1_ADR     .EQU      H'FFFFFF84
PORT4      .BEQU     4,P1_ADR
;
        .SECTION   A,CODE,ALIGN=2
MAIN:
        MOV.L     @DATA1,ER0
        BTST      PORT4          ;BTST         #4,@H'FFFFFF84
        BEQ       SKIP
        BCLR      PORT4          ;BCLR         #4,@H'FFFFFF84
        JSR       @SUBRTN
        MOV.L     @DATA1,ER1
        MOV.L     ER0,@ER1
SKIP:
        BRA      MAIN
;
        .END       MAIN
```

例２）

```
        .CPU       2600A
;
        .EXPORT   SUBRTN
        .EXPORT   DATA1
;
        .SECTION   A,CODE,ALIGN=2
SUBRTN:
        ADD.L     #1,ER0
        RTS
;
        .SECTION   C,DATA,ALIGN=4
DATA1:
        .DATA.L   H'00
        .END
```

3. 制御命令一覧

制御命令は、アセンブラが解釈し、実行する命令です。
表 3.1 に、制御命令の一覧を示します。
文字は、大文字・小文字のどちらでも使用できます。

表 3.1　制御命令一覧（1）

項番	分類	ニーモニック	機能と書き方
1	CPUに関するもの	.CPU	機能　：ＣＰＵを指定する 書き方：△.CPU　CPU 種別[:ｱﾄﾞﾚｽ空間のﾋﾞｯﾄ幅] ・CPU 種別は、ｱｾﾝﾌﾞﾙするｿｰｽﾌﾟﾛｸﾞﾗﾑの対象とする CPU を指定 H8S/2600ｱﾄﾞﾊﾞﾝｽﾄﾓｰﾄﾞ = 2600A[:32],2600A:28, 2600A:24,2600A:20 H8S/2600ﾉｰﾏﾙﾓｰﾄﾞ = 2600N H8S/2000ｱﾄﾞﾊﾞﾝｽﾄﾓｰﾄﾞ = 2000A[:32],2000A:28, 2000A:24,2000A:20 H8S/2000ﾉｰﾏﾙﾓｰﾄﾞ = 2000N H8/300Hｱﾄﾞﾊﾞﾝｽﾄﾓｰﾄﾞ = 300HA[:24],300HA:20 H8/300Hﾉｰﾏﾙﾓｰﾄﾞ = 300HN H8/300 = 300 H8/300L = 300L ・ｱｸｾｽ可能な範囲は、ｱﾄﾞﾚｽ空間のﾋﾞｯﾄ幅により異なる ・本制御命令は、ﾌﾟﾛｸﾞﾗﾑの最初に記述し、1 回限り有効 ・本制御命令は、CPUｵﾌﾟｼｮﾝの指定がない場合に有効
2	ロケーションカウンタに関するものセクションまたは	.SECTION	機能　：セクションを宣言する 書き方：△.SECTION　ｾｸｼｮﾝ名 [,ｾｸｼｮﾝ属性 [{ ,LOCATE = 先頭ｱﾄﾞﾚｽ ,ALIGN = 境界調整数 }]] ・ｾｸｼｮﾝ属性は、 CODE，DATA，STACK，COMMON，DUMMY ・ｾｸｼｮﾝ属性を省略すると CODE ・境界調整数は相対ｱﾄﾞﾚｽｾｸｼｮﾝの境界調整数を 2 のべき乗の整数で指定 ・境界調整数を省略すると 2 ・先頭ｱﾄﾞﾚｽは、絶対ｱﾄﾞﾚｽｾｸｼｮﾝの先頭ｱﾄﾞﾚｽを指定 ・先頭ｱﾄﾞﾚｽの値は、0 ～ ｱﾄﾞﾚｽ空間の最大値 （2600A の場合、H'FFFFFFFF）

表 3.1　制御命令一覧（２）

項番	分類	ニーモニック	機能と書き方
2	ロケーションカウンタに関するものセクションまたは	.ORG	機能　：ロケーションカウンタ値を設定する 書き方：△.ORG　ﾛｹｰｼｮﾝｶｳﾝﾀ値 ・ﾛｹｰｼｮﾝｶｳﾝﾀ値は、0～ｱﾄﾞﾚｽ空間の最大値 （2600A の場合、H'FFFFFFFF） ・絶対ｱﾄﾞﾚｽｾｸｼｮﾝの場合、ｾｸｼｮﾝの先頭ｱﾄﾞﾚｽより大きい値を指定 ・絶対ｱﾄﾞﾚｽｾｸｼｮﾝの場合、ﾛｹｰｼｮﾝｶｳﾝﾀ値は絶対番地となる ・相対ｱﾄﾞﾚｽｾｸｼｮﾝの場合、ﾛｹｰｼｮﾝｶｳﾝﾀ値はｾｸｼｮﾝ先頭からの相対番地となる
		.ALIGN	機能　：ロケーションカウンタ値を補正する 書き方：△.ALIGN　境界調整数 ・境界調整数は２のべき乗の整数 ・相対ｱﾄﾞﾚｽｾｸｼｮﾝで境界調整数を指定している場合、それ以上の値でなければならない
3	シンボルに関するもの	.EQU	機能　：シンボルに値を設定する(再設定が不可能) 書き方：ｼﾝﾎﾞﾙ　.EQU　ｼﾝﾎﾞﾙ値 ・ｼﾝﾎﾞﾙ値は、0～H'FFFFFFFF ・ｼﾝﾎﾞﾙ値には、前方参照ｼﾝﾎﾞﾙ、外部参照ｼﾝﾎﾞﾙは指定できない
		.BEQU	機能　：ビットデータ名を設定する 書き方：ｼﾝﾎﾞﾙ　.BEQU　ﾋﾞｯﾄ番号，置換ｼﾝﾎﾞﾙ名 ・ﾋﾞｯﾄ操作命令の対象となるﾒﾓﾘ上の 1ﾋﾞｯﾄﾃﾞｰﾀに名称をつける ・ﾋﾞｯﾄﾃﾞｰﾀ名は、ﾋﾞｯﾄ操作命令のｵﾍﾟﾗﾝﾄﾞに指定できる ・指定されたﾋﾞｯﾄﾃﾞｰﾀ名は、#xx,@aa 形式に置換 ・ﾋﾞｯﾄ番号は、後方参照の絶対値で、0～7 の値を指定 ・置換ｼﾝﾎﾞﾙ名は、ﾋﾞｯﾄ操作命令の絶対ｱﾄﾞﾚｽ形式に指定できるｱﾄﾞﾚｽｼﾝﾎﾞﾙを指定
		.ASSIGN	機能　：シンボルに値を設定する(再設定が可能) 書き方：ｼﾝﾎﾞﾙ　.ASSIGN　ｼﾝﾎﾞﾙ値 ・ｼﾝﾎﾞﾙ値は、0～H'FFFFFFFF ・ｼﾝﾎﾞﾙ値には、前方参照ｼﾝﾎﾞﾙ、外部参照ｼﾝﾎﾞﾙは指定できない

表 3.1　制御命令一覧（3）

項番	分類	ニーモニック	機能と書き方
3	シンボルに関するもの	.REG	機能　：レジスタ別名を定義する 書き方：ｼﾝﾎﾞﾙ　.REG　(ﾚｼﾞｽﾀ名) ・ﾚｼﾞｽﾀ名には、単一ﾚｼﾞｽﾀと複数ﾚｼﾞｽﾀがある 単一ﾚｼﾞｽﾀ ...1 つのﾚｼﾞｽﾀにﾚｼﾞｽﾀ別名をつける ﾚｼﾞｽﾀ名には R0L～R7L,R0H～R7H,R0～R7, E0～E7,ER0～ER7 複数ﾚｼﾞｽﾀ ...2 つ以上のﾚｼﾞｽﾀにﾚｼﾞｽﾀ別名をつける CPU 種別が H8S/2600、H8S/2000 の LDM 命令， STM 命令に指定 ﾚｼﾞｽﾀ名には(ER0-ER1),(ER2-ER3),(ER4-ER5), (ER6-ER7),(ER0-ER2),(ER4-ER6), (ER0-ER3),(ER4-ER7) ・本制御命令より後のｿｰｽｽﾃｰﾄﾒﾝﾄで参照できる ・ｼﾝﾎﾞﾙは外部定義できない
4	データまたはデータ領域を確保するもの	.DATA	機能　：整数データを確保する 書き方：[ｼﾝﾎﾞﾙ]　.DATA[.sz]　整数ﾃﾞｰﾀ [,整数ﾃﾞｰﾀ･･･] ・sz は、B，W，L のいずれか。省略すると B（ﾊﾞｲﾄ） ・整数ﾃﾞｰﾀには、前方参照ｼﾝﾎﾞﾙ、外部参照ｼﾝﾎﾞﾙも指定できる ・整数ﾃﾞｰﾀの値は以下のとおり B...H'0～H'FF、H'FFFFFF80～H'FFFFFFFF W...H'0000～H'FFFF、H'FFFF8000～H'FFFFFFFF L...H'00000000～H'FFFFFFFF、H'80000000～H'FFFFFFFF
		.DATAB	機能　：整数データブロックを確保する 書き方：[ｼﾝﾎﾞﾙ]　.DATAB[.sz]　ﾌﾞﾛｯｸ数，整数ﾃﾞｰﾀ ・sz には、B，W，L のいずれか。省略すると B（ﾊﾞｲﾄ） ・ﾌﾞﾛｯｸ数には、前方参照ｼﾝﾎﾞﾙを含まない絶対値を指定 ・ﾌﾞﾛｯｸ数は、ｱﾄﾞﾚｽ空間のﾋﾞｯﾄ幅により異なる ・整数ﾃﾞｰﾀの値は以下のとおり B...H'0～H'FF、H'FFFFFF80～H'FFFFFFFF W...H'0000～H'FFFF、H'FFFF8000～H'FFFFFFFF L...H'00000000～H'FFFFFFFF、H'80000000～H'FFFFFFFF
		.SDATA	機能　：文字列データを確保する 書き方：[ｼﾝﾎﾞﾙ]　.SDATA　"文字列"[,"文字列"･･･] ・文字列の末尾に、制御ｺｰﾄﾞを指定できる ・制御ｺｰﾄﾞの書き方は、"文字列"<制御ｺｰﾄﾞ>

表 3.1　制御命令一覧（4）

項番	分類	ニーモニック	機能と書き方
4	データ領域を確保するものまたはデータ	.SDATAB	機能　：文字列データブロックを確保する 書き方：[ｼﾝﾎﾞﾙ]　.SDATAB　ﾌﾞﾛｯｸ数，"文字列" ・ﾌﾞﾛｯｸ数には、前方参照ｼﾝﾎﾞﾙを含まない絶対値を指定 ・文字列の末尾に制御ｺｰﾄﾞを指定できる ・制御ｺｰﾄﾞの書き方は、"文字列"<制御ｺｰﾄﾞ> ・ﾌﾞﾛｯｸ数は、ｱﾄﾞﾚｽ空間のﾋﾞｯﾄ幅により異なる
		.RES	機能　：データ領域を確保する 書き方：[ｼﾝﾎﾞﾙ]　.RES[.sz]　領域数 ・sz には、B，W，L のいずれか。省略すると B（ﾊﾞｲﾄ） ・領域数には、前方参照ｼﾝﾎﾞﾙを含まない絶対値を指定 ・領域数は、ｱﾄﾞﾚｽ空間のﾋﾞｯﾄ幅により異なる
5	外部定義または外部参照に関するもの	.EXPORT	機能　：外部定義シンボルを宣言する 書き方：△.EXPORT　ｼﾝﾎﾞﾙ[,ｼﾝﾎﾞﾙ･･･] ・ｼﾝﾎﾞﾙに指定できるのは、以下のとおり .ASSIGN 制御命令で定義した以外の定数ｼﾝﾎﾞﾙ ﾀﾞﾐｰｾｸｼｮﾝ以外の絶対ｱﾄﾞﾚｽｼﾝﾎﾞﾙ 相対ｱﾄﾞﾚｽｼﾝﾎﾞﾙ
		.IMPORT	機能　：外部参照シンボルを宣言する 書き方：△.IMPORT　ｼﾝﾎﾞﾙ[,ｼﾝﾎﾞﾙ･･･] ・ﾌｧｲﾙ内で定義しているｼﾝﾎﾞﾙは指定できない
		.GLOBAL	機能　：外部参照シンボルまたは外部定義シンボルを宣言する 書き方：△.GLOBAL　ｼﾝﾎﾞﾙ[,ｼﾝﾎﾞﾙ･･･] ・ﾌｧｲﾙ内で定義しているｼﾝﾎﾞﾙは外部定義に、定義していないｼﾝﾎﾞﾙは外部参照になる ・外部定義ｼﾝﾎﾞﾙの制限については.EXPORT を参照
6	その他	.END	機能　：ソースプログラムの終わりを宣言する 書き方：△.END ・.END 以降にｿｰｽｽﾃｰﾄﾒﾝﾄがあっても無視される

4. オプション一覧

オプションは、アセンブル方法を細かく指示するための指定です。
表 4.1 に、オプションの一覧を示します。
オプション名の下線部は、省略形です。
文字は、大文字・小文字のどちらでも使用できます。

表 4.1　オプション一覧（1）

項番	分類	オプション名 (否定形)	機　能
1	CPUに関するもの	CPU	機能　：CPU の指定 書き方：　-CPU=CPU 種別[:ｱﾄﾞﾚｽ空間のﾋﾞｯﾄ幅] ・CPU 種別は、ｱｾﾝﾌﾞﾙするｿｰｽﾌﾟﾛｸﾞﾗﾑの対象とする CPU を指定 H8S/2600ｱﾄﾞﾊﾞﾝｽﾄﾓｰﾄﾞ = 2600A[:32],2600A:28, 2600A:24,2600A:20 H8S/2600ﾉｰﾏﾙﾓｰﾄﾞ = 2600N H8S/2000ｱﾄﾞﾊﾞﾝｽﾄﾓｰﾄﾞ = 2000A[:32],2000A:28, 2000A:24,2000A:20 H8S/2000ﾉｰﾏﾙﾓｰﾄﾞ = 2000N H8/300Hｱﾄﾞﾊﾞﾝｽﾄﾓｰﾄﾞ = 300HA[:24],300HA:20 H8/300Hﾉｰﾏﾙﾓｰﾄﾞ = 300HN H8/300 = 300 H8/300L = 300L ・ｱｸｾｽ可能な範囲は、ｱﾄﾞﾚｽ空間のﾋﾞｯﾄ幅により異なる
2	オブジェクトモジュールに関するもの	OBJECT (NOOBJECT)	機能　：オブジェクトモジュールの出力を制御する 否定形は、オブジェクト出力抑止 書き方：　-OBJECT[=出力ｵﾌﾞｼﾞｪｸﾄﾌｧｲﾙ名] -NOOBJECT ・出力ｵﾌﾞｼﾞｪｸﾄﾌｧｲﾙ名を省略すると、1 つめに指定した入力ﾌｧｲﾙ名のﾌｧｲﾙ形式を"obj"にしたものを出力 ・出力ｵﾌﾞｼﾞｪｸﾄﾌｧｲﾙ名のﾌｧｲﾙ形式を省略すると、"obj"を仮定
		DEBUG (NODEBUG)	機能　：デバッグ情報の出力を制御する 否定形は、デバッグ情報の出力抑止 書き方：-DEBUG -NODEBUG ・ｵﾌﾞｼﾞｪｸﾄﾓｼﾞｭｰﾙを出力する場合にのみ有効 ・DEBUGｵﾌﾟｼｮﾝを指定しないとﾃﾞﾊﾞｯｸﾞ情報を出力しない

表4.1　オプション一覧（2）

項番	分類	オプション名(否定形)	機　能
2	オブジェクトモジュールに関するもの	OPTIMIZE (NOOPTIMIZE)	機能　：最適化を指定する 　　否定形は、最適化を抑止 書き方：　-OPTIMIZE 　　-NOOPTIMIZE ・PC 相対形式、ﾃﾞｨｽﾌﾟﾚｰｽﾒﾝﾄ付きﾚｼﾞｽﾀ間接形式のﾃﾞｨｽﾌﾟﾚｰｽﾒﾝﾄｻｲｽﾞと絶対ｱﾄﾞﾚｽ形式の絶対ｱﾄﾞﾚｽｻｲｽﾞの最適化、最適化抑止を指定 ・本ｵﾌﾟｼｮﾝの対象となるのは、ﾃﾞｨｽﾌﾟﾚｰﾄﾒﾝﾄｻｲｽﾞ(:8，:16)、絶対ｱﾄﾞﾚｽの確保ｻｲｽﾞ(:8，:16，:24，:32)の指定がない実行命令 ・OPTIMIZEｵﾌﾟｼｮﾝを指定しないと最適化しない
3	ファイルインクルード機能に関するもの	INCLUDE	機能　：インクルードファイルの取り込み先を指定する 書き方：-INCLUDE=ﾃﾞｨﾚｸﾄﾘ名[,ﾃﾞｨﾚｸﾄﾘ名･･･] ・ｿｰｽﾌﾟﾛｸﾞﾗﾑ内で.INCLUDE 制御命令でﾃﾞｨﾚｸﾄﾘが指定されていると、そのﾃﾞｨﾚｸﾄﾘの前に付加 ・ｲﾝｸﾙｰﾄﾞﾌｧｲﾙの検索順は、ｶﾚﾝﾄﾃﾞｨﾚｸﾄﾘ → -INCLUDE 指定ﾃﾞｨﾚｸﾄﾘ ・複数のﾃﾞｨﾚｸﾄﾘを指定すると、指定された順に検索

リンケージエディタ編

1. 起動方法

リンケージエディタを起動するコマンドラインの形式は、次のとおりです。

```
lnk△[入力ファイル名[ { △ } 入力ファイル名]…]
                    { , }
    [[△]-オプション名[[△]-オプション名･･･]]
```

（1） 起動コマンド

リンケージエディタの起動コマンドとして ”lnk” を入力します。

（2） 入力ファイル名

リンケージエディタの入力となるオブジェクトモジュールファイル名を指定します。複数のファイルを指定する場合は、空白またはカンマ(,)で区切って指定します。

入力ファイル名にファイル形式の指定がない場合は、リンケージエディタが自動的に ”obj” をファイル形式と仮定してファイルを入力します。

（3） オプション名

オプション名は、必ず先頭にハイフン（-）をつけて指定してください。また、オプション名と入力ファイル名またはオプション名同士は、連続して指定しても、1つ以上のスペースまたはタブで区切っても構いません。

オプション名の一覧については、**「2. オプション/サブコマンド」**を参照してください。

（4） 実行形式の指定

リンケージエディタをコマンドライン指定のみで実行するか、サブコマンド指定によって実行するかは、コマンドラインにより指定できます。

(a)コマンドラインによる実行の指定

コマンドラインで入力ファイル名の指定があり、かつサブコマンドファイル指定がない場合は、コマンドライン指定だけの実行になります。

(b)サブコマンドによる実行の指定

コマンドラインで入力ファイル名の指定がないか、またはサブコマンドファイル指定がある場合は、サブコマンド指定による実行になります。

1.1.オプションのフォーマット

（1）指定の形式

オプションは、次の形式で指定します。

＜名称部＞＝＜パラメータ＞

パラメータがないものについては、＜名称部＞だけを指定します。

（2）オプションの指定例

lnk△add,sub,mul,div△-OUTPUT=arith△-ENTRY=main　(RET)

リンケージエディタは"add.obj"、"sub.obj"、"mul.obj"および"div.obj"の 4 つのファイルを入力し、ロードモジュールファイル arith.abs を出力します。出力するロードモジュールファイルの実行開始アドレスは、外部定義シンボル"main"のアドレスに設定します。また、リンケージリストは出力しません。

1.2. サブコマンドのフォーマット

サブコマンド指定の実行には、サブコマンドを標準入力から入力していく会話形式と、サブコマンド群を記述したサブコマンドファイルから入力していくサブコマンドファイルの方法があります。

1.2.1. 会話形式による実行

（1）指定の形式

サブコマンドは、次の形式で指定します。

＜名称部＞ △ ＜パラメータ＞

パラメータがないものについては、＜名称部＞だけを指定します。

（2）会話形式の指定例

```
 lnk   (RET)                                          …①
:INPUT△main   (RET)                                  …②
:INPUT△send,receive,exchange  (RET)                   …③
:INPUT△account   (RET)                               …④
:LIBRARY△syslib   (RET)                               …⑤
:PRINT△#   (RET)                                     …⑥
:EXIT   (RET)                                        …⑦
```

① リンケージエディタを会話形式で実行します。
② オブジェクトモジュールファイル"main.obj"を入力します。
③ ３つのオブジェクトモジュールファイル"send.obj"、"receive.obj"、および"exchange.obj"を入力します。
④ オブジェクトモジュールファイル"account.obj"を入力します。
⑤ ライブラリファイル"syslib.lib"を入力します。
⑥ リンケージリストを標準出力へ出力します。

⑦ ロードモジュールファイル"main.abs"を出力し、リンケージ処理を終了します。

1.2.2. サブコマンドファイルによる実行

```
lnk△-SUBCOMMAND=prglnk.sub   (RET)                    …①
サブコマンドファイル"prglnk.sub"の内容
 OUTPUT△function                                      …②
 INPUT△sin,cos,tan                                    …③
 INPUT△asin,acos,atan                                 …④
 INPUT△hsin,hcos,htan                                 …⑤
 INPUT△log,log10                                      …⑥
 EXIT                                                 …⑦
```

① リンケージエディタを実行し、サブコマンドファイル"prglnk.sub"からサブコマンドを入力します。
② 出力ファイル名を"function"とします。
③ ３つのオブジェクトモジュールファイル"sin.obj"、"cos.obj"、および"tan.obj"を入力します。
④ ３つのオブジェクトモジュールファイル"asin.obj"、"acos.obj"、および"atan.obj"を入力します。
⑤ ３つのオブジェクトモジュールファイル"hsin.obj"、"hcos.obj"、および"htan.obj"を入力します。
⑥ オブジェクトモジュールファイル"log.obj"および"log10.obj"を入力します。
⑦ ロードモジュールファイル"function.abs"を出力し、リンケージ処理を終了します。

2. オプション/サブコマンド一覧

表 2.1 にオプション/サブコマンド一覧を示します。
文字は、大文字・小文字のどちらでも使用できます。
オプション／サブコマンド名の下線部は、省略形です。

表 2.1 オプション/サブコマンド一覧表（1）

項番	分類	ｵﾌﾟｼｮﾝ/ｻﾌﾞｺﾏﾝﾄﾞ名 (否定形)	機能、パラメータ	ｵﾌﾟｼｮﾝ	ｻﾌﾞｺﾏﾝﾄﾞ
1	ファイル制御	INPUT	機能　：入力ファイル指定 ﾊﾟﾗﾒｰﾀ：入力ﾌｧｲﾙ [{ △ / , }入力ﾌｧｲﾙ･･･] ・指定できるﾌｧｲﾙの種類は、ｵﾌﾞｼﾞｪｸﾄﾓｼﾞｭｰﾙ ・ﾃﾞﾌｫﾙﾄ拡張子は、".obj"を仮定 ・指定できるﾌｧｲﾙの数は、256ﾌｧｲﾙまで	×	○
		OUTPUT (NOOUTPUT)	機能　：出力ファイルの指定 　　否定形は、ロードモジュールの出力抑止 ﾊﾟﾗﾒｰﾀ：[出力ﾌｧｲﾙ名] ・OUTPUT で出力ﾌｧｲﾙ名の指定を省略すると、先頭の入力ﾌｧｲﾙ名を採用 ・ﾃﾞﾌｫﾙﾄ拡張子は、".abs"を仮定 ・NOOUTPUT では、ﾊﾟﾗﾒｰﾀの指定はない	○	○
		PRINT (NOPRINT)	機能　：リストファイルの指定 　　否定形は、リスト出力の抑止 ﾊﾟﾗﾒｰﾀ：{ ﾘｽﾄﾌｧｲﾙ名 / # } ・#を指定すると、標準出力へ出力 ・ﾃﾞﾌｫﾙﾄの拡張子は、".map"を仮定 ・NOPRINT では、ﾊﾟﾗﾒｰﾀの指定はない	○	○

表 2.1　オプション/サブコマンド一覧表（2）

項番	分類	ｵﾌﾟｼｮﾝ/ｻﾌﾞｺﾏﾝﾄﾞ名 (否定形)	機能、パラメータ	ｵﾌﾟｼｮﾝ	ｻﾌﾞｺﾏﾝﾄﾞ
1	ファイル制御	LIBRARY (NOLIBRARY)	機能　：入力ライブラリファイルの指定 　　否定形は、ライブラリファイルからの入力はしない ﾊﾟﾗﾒｰﾀ：ﾗｲﾌﾞﾗﾘﾌｧｲﾙ名[,ﾗｲﾌﾞﾗﾘﾌｧｲﾙ名･･･] ・入力ﾌｧｲﾙとして指定されたﾌｧｲﾙ間でのﾘﾝｹｰｼﾞ処理後に未解決の外部参照ｼﾝﾎﾞﾙが残った場合に、ﾘﾝｹｰｼﾞｴﾃﾞｨﾀがｻｰﾁするﾗｲﾌﾞﾗﾘﾌｧｲﾙを指定 ・ﾗｲﾌﾞﾗﾘﾌｧｲﾙの詳細は、Cｺﾝﾊﾟｲﾗのﾏﾆｭｱﾙを参照	○	○
2	メモリ割り付け	START	機能　：セクション先頭アドレス/結合順序の指定 ﾊﾟﾗﾒｰﾀ：ｾｸｼｮﾝ名[,ｾｸｼｮﾝ名･･･][(先頭ｱﾄﾞﾚｽ)] 　　[,ｾｸｼｮﾝ名[,ｾｸｼｮﾝ名･･･][(先頭ｱﾄﾞﾚｽ)]･･･] ・先頭ｱﾄﾞﾚｽを省略すると、0 番地から割り付ける ・先頭ｱﾄﾞﾚｽは 16 進数で指定 ・先頭が A～F の場合、最初に 0 をつける ・先頭ｱﾄﾞﾚｽの値は、0～ｱﾄﾞﾚｽ空間の最大値 　　(2600A の場合、0FFFFFFFF) ・先頭ｱﾄﾞﾚｽ、結合順序を指定しなかったｾｸｼｮﾝがあった場合、並び順の末尾に結合	○	○
		ENTRY	機能　：出力するロードモジュールに対して、実行開始アドレスを指定 ﾊﾟﾗﾒｰﾀ：外部定義ｼﾝﾎﾞﾙ名 ・出力するﾛｰﾄﾞﾓｼﾞｭｰﾙ対して、実行開始ｱﾄﾞﾚｽを指定 ・ENTRYｵﾌﾟｼｮﾝ/ｻﾌﾞｺﾏﾝﾄﾞの指定がない場合、出力するﾛｰﾄﾞﾓｼﾞｭｰﾙ中のｺｰﾄﾞｾｸｼｮﾝのうち、最も先頭に現れたｾｸｼｮﾝの先頭ｱﾄﾞﾚｽが実行開始ｱﾄﾞﾚｽとなる ・ENTRYｵﾌﾟｼｮﾝ/ｻﾌﾞｺﾏﾝﾄﾞを２度以上指定した場合、後から指定したｱﾄﾞﾚｽが有効	○	○
		ROM	機能　：ROM 領域の初期化ﾃﾞｰﾀを変更できるように同一ｻｲｽﾞの RAM 領域を確保 ﾊﾟﾗﾒｰﾀ：(ROMｾｸｼｮﾝ,RAMｾｸｼｮﾝ) 　　[,(ROMｾｸｼｮﾝ,RAMｾｸｼｮﾝ)･･･] ・RAMｾｸｼｮﾝが実在する場合、ｻｲｽﾞは 0 でなければならない	○	○

表 2.1　オプション/サブコマンド一覧表（3）

項番	分類	ｵﾌﾟｼｮﾝ/ｻﾌﾞｺﾏﾝﾄﾞ名 (否定形)	機能、パラメータ	ｵﾌﾟｼｮﾝ	ｻﾌﾞｺﾏﾝﾄﾞ
3	実行制御	SUBCOMMAND	機能　：サブコマンドファイル指定 ﾊﾟﾗﾒｰﾀ：ｻﾌﾞｺﾏﾝﾄﾞﾌｧｲﾙ名 ・ｺﾏﾝﾄﾞﾗｲﾝで入力ﾌｧｲﾙと SUBCOMMANDｵﾌﾟｼｮﾝ/ｻﾌﾞｺﾏﾝﾄﾞの指定がない場合､標準入力からｻﾌﾞｺﾏﾝﾄﾞを会話形式で入力 ・ｻﾌﾞｺﾏﾝﾄﾞﾌｧｲﾙ内の内容は、ｵﾌﾟｼｮﾝより後に解析	○	○
		DEBUG (NODEBUG)	機能　：デバッグ情報出力の指定 ﾊﾟﾗﾒｰﾀ：なし ・ﾃﾞﾊﾞｯｸﾞ情報を出力ﾛｰﾄﾞﾓｼﾞｭｰﾙに出力 ・NOOUTPUTｵﾌﾟｼｮﾝ/ｻﾌﾞｺﾏﾝﾄﾞを指定されていると無効 ・DEBUGｵﾌﾟｼｮﾝ/ｻﾌﾞｺﾏﾝﾄﾞを指定しないとﾃﾞﾊﾞｯｸﾞ情報を出力しない	○	○
		EXIT	機能　：リンケージ処理の終了指定 ﾊﾟﾗﾒｰﾀ：なし ・ｻﾌﾞｺﾏﾝﾄﾞの並びの末尾に EXIT がないと、ｻﾌﾞｺﾏﾝﾄﾞ入力待ちとなる	×	○
4	デバッグ援助	DEFINE	機能　：外部参照シンボルの強制定義 ﾊﾟﾗﾒｰﾀ：外部参照ｼﾝﾎﾞﾙ名({ 数値 / ｱﾄﾞﾚｽ / 外部定義ｼﾝﾎﾞﾙ名 }) [,外部参照ｼﾝﾎﾞﾙ名 ({ 数値 / ｱﾄﾞﾚｽ / 外部定義ｼﾝﾎﾞﾙ名 }) ･･･] ・数値、ｱﾄﾞﾚｽは 16 進数で指定。先頭が A～F の場合、0 をつける ・値の範囲は、0～ｱﾄﾞﾚｽ空間の最大値 （2600A の場合、0FFFFFFFF）	○	○