

日立マイクロコンピュータ開発環境システム
Hシリーズ
リンケージエディタ、ライブラリアン、
オブジェクトコンバータ
ユーザーズマニュアル

HS6400LECU6S

発行年月日	平成 8 年 6 月 第 1 版
	平成 10 年 9 月 第 2 版
発行	株式会社 日立製作所
	半導体事業本部 統括営業本部
編集	株式会社 超 L メディア
	技術ドキュメントグループ

©株式会社 日立製作所 1996

はじめに

本マニュアルは、MS-DOS*またはUNIX*上で稼働するHシリーズ リンケージエディタ、ライブラリアン、およびオブジェクトコンバータの使い方について説明したものです。

本マニュアルは、下記の3つの編から構成されています。

- ・「リンケージエディタ編」
- ・「ライブラリアン編」
- ・「オブジェクトコンバータ編」

【注】 Hシリーズリンケージエディタ Ver.6.0、ライブラリアン Ver.2.0、オブジェクトコンバータ Ver.2.0を使用している場合は「Ver.6.0の追加・変更内容」を参照してください。

本マニュアルの他に、リンケージエディタ、ライブラリアン、およびオブジェクトコンバータと関連するHシリーズ クロスソフトウェアとして、次のマニュアルがあります。

- ・ H8S, H8/300 シリーズ クロスアセンブラ ユーザーズマニュアル
- ・ H8S, H8/300 シリーズ C/C++コンパイラ ユーザーズマニュアル
- ・ H8S, H8/300 シリーズ シミュレータ・デバッガ ユーザーズマニュアル

- ・ H8/500 シリーズ クロスアセンブラ ユーザーズマニュアル
- ・ H8/500 シリーズ C コンパイラ ユーザーズマニュアル
- ・ H8/500 シリーズ シミュレータ・デバッガ ユーザーズマニュアル

- ・ SuperH RISC engine アセンブラ ユーザーズマニュアル
- ・ SuperH RISC engine C/C++コンパイラ ユーザーズマニュアル
- ・ SH シリーズ シミュレータ・デバッガ ユーザーズマニュアル

【注】 * MS-DOS は米国マイクロソフト社により管理されている、オペレーティングシステムの名称です。

* UNIX は X/Open 社により管理されている、オペレーティングシステムの名称です。

表記上の注意事項

本マニュアルの説明の中で使用する記号は、次の内容を示しています。

- < > 指定する情報の内容を意味します。
- { } いずれか1つを選択することを意味します。
- [] 省略できることを意味します。
- ・・・ 任意の回数だけ繰り返すことを意味します。
- 1つ以上のスペースまたはタブを意味します。
- (RET) リターンキーの入力を意味します。

MS-DOS 版では、ファイルの拡張子は大文字で設定されます。

本マニュアルでは、16進数を次のように表します。

数値の前に、H'を記述する。(例、H'1000)

特に断らない場合、前になにもつかない数値は10進数です。

目次

リンケージエディタ編

第1章 概要.....	1
1.1 機能概要	5
1.2 オブジェクトモジュールとロードモジュール.....	6
1.3 ユニットとセクション	7
第2章 リンケージエディタの機能	9
2.1 モジュールの結合機能	12
2.1.1 セクションの結合.....	12
2.1.2 ライブラリファイルからの入力.....	21
2.1.3 ライブラリファイル内のモジュールの結合抑止	23
2.2 アドレスの解決機能.....	24
2.2.1 外部参照の解決	24
2.2.2 セクション内のアドレス解決	26
2.2.3 アドレス未解決シンボルの表示抑止	27
2.3 ロードモジュールファイルの再入力機能.....	28
2.3.1 ユニットの自動置換	29
2.3.2 ユニットの強制置換	30
2.4 マルチリンケージ機能	31
2.5 デバッグ援助機能.....	32
2.6 アドレスチェック機能	33
2.7 ROM化支援機能.....	34
第3章 リンケージエディタの実行	37
3.1 コマンドラインのフォーマット	40
3.2 コマンドライン指定による実行.....	41
3.3 サブコマンド指定による実行	42
3.3.1 会話形式による実行	42
3.3.2 サブコマンドファイルによる実行	43
3.4 リンケージエディタの終了.....	45

第4章 リンケージエディタのオプション / サブコマンド	47
4.1 オプション / サブコマンドのフォーマット	50
4.2 オプション / サブコマンドの種類	52
4.3 ファイル制御	57
4.3.1 INPUT - 入力ファイルの指定	57
4.3.2 OUTPUT - 出力ファイルの指定	59
4.3.3 LIBRARY - ライブラリファイルの指定	61
4.3.4 PRINT - リストファイルの指定	63
4.3.5 EXCLUDE - ライブラリファイルモジュールの 結合抑止指定	64
4.3.6 DIRECTORY - ディレクトリ名置き換えの指定	66
4.3.7 FSYMBOL - 解決済み外部定義シンボルの出力指定	67
4.4 メモリ割り付け	69
4.4.1 START - セクションの先頭アドレス / 結合順序の指定	69
4.4.2 ENTRY - 実行開始アドレスの指定	72
4.4.3 ALIGN_SECTION - 境界調整が異なる セクションの結合指定	73
4.4.4 CHECK_SECTION - セクションのチェックの指定	74
4.4.5 AUTOPAGE - 自動ページングの指定	75
4.4.6 CPU - CPU 情報ファイルによるアドレスチェックの指定	76
4.4.7 CPUCHECK - CPU 情報ファイルによる アドレスチェック時のエラー出力指定	78
4.4.8 ROM - ROM 化支援機能の指定	79
4.5 実行制御	81
4.5.1 EXCHANGE - ユニットの強制置換	81
4.5.2 SUBCOMMAND - サブコマンドファイルの指定	83
4.5.3 FORM - 出力ロードモジュールファイル形式の指定	85
4.5.4 DEBUG - デバッグ情報出力の指定	86
4.5.5 SDEBUG - デバッグ情報ファイル出力の指定	87
4.5.6 END - サブコマンド入力の終了指定	88
4.5.7 EXIT - リンケージ処理の終了指定	89
4.5.8 ABORT - リンケージ処理の強制終了指定	90
4.5.9 ECHO - サブコマンドファイルのエコーバックの指定	91
4.5.10 UDF - 未定義シンボルの表示指定	92
4.5.11 UDFCHECK - 未定義シンボルが存在する場合の エラー出力指定	93
4.6 デバッグ援助	94
4.6.1 LIST - 中間リンケージ情報の表示	94

4.6.2	RENAME - ユニット名 / 外部定義シンボル名 / 外部参照シンボル名の変更	95
4.6.3	DELETE - ユニット / 外部定義シンボルの削除	97
4.6.4	DEFINE - 外部参照シンボルの強制定義	99
第 5 章	リンケージエディタの入力	101
5.1	オブジェクトモジュールファイル	103
5.2	リロケータブルロードモジュールファイル	104
5.3	ライブラリファイル	105
5.4	デフォルトライブラリファイル	106
第 6 章	リンケージエディタの出力	107
6.1	リンケージリスト	109
6.2	ロードモジュールファイル	116
6.3	中間ファイル	117
6.4	コンソールメッセージ	118
第 7 章	エラーメッセージ	121
第 8 章	制限事項一覧	135
付録		
付録 A.	リンケージエディタ使用例	141
付録 B.	ファイル名の構成	151

図表目次

図 1.1	プログラムの開発手順	3
図 1.2	モジュール、ユニットおよびセクションの関係	7
図 2.1	セクションの集合	12
図 2.2	単純結合	13
図 2.3	共有結合	13
図 2.4	ダミー結合	13
図 2.5	結合順序指定ありのセクション結合例	14
図 2.6	結合順序指定なしのセクション結合例	15
図 2.7	セクション属性の異なる同一名セクション結合例	16
図 2.8	ページタイプの結合例（自動ページング指定なし、先頭アドレス指定なし）	18
図 2.9	ページタイプの結合例（自動ページング指定あり、先頭アドレス指定なし）	19
図 2.10	ページタイプの結合例（自動ページング指定あり、先頭アドレス指定あり）	20
図 2.11	モジュール結合の例（入力オブジェクトモジュール）	22
図 2.12	モジュール結合の例（入力ライブラリファイル）	22
図 2.13	モジュール結合の例（出力ロードモジュール）	23
図 2.14	参照のない外部参照シンボルを含むモジュール例	23

図 2.15	外部参照の解決.....	25
図 2.16	セクション内のアドレス解決.....	27
図 2.17	ロードモジュールファイルの再入力機能.....	28
図 2.18	ユニットの自動置換.....	29
図 2.19	マルチリンケージ機能.....	31
図 2.20	ROM 化支援機能を使ったときのメモリマップ.....	34
図 2.21	ROM 化支援機能を使ったときのシンボルのアドレス.....	35
図 6.1	入力情報の出力例.....	109
図 6.2	PRINT で出力するマップリスト例.....	110
図 6.3	LIST で出力するマップリスト例.....	110
図 6.4	PRINT で出力する外部定義シンボルリスト例.....	112
図 6.5	LIST で出力する外部定義シンボルリスト例.....	112
図 6.6	PRINT で出力する未定義シンボルリスト例.....	113
図 6.7	LIST で出力する未定義シンボルリスト例.....	113
図 6.8	変更 / 削除シンボルリスト例.....	114
図 6.9	強制定義シンボルリスト例.....	115
図 A.1	サブコマンドファイル “ exlink.sub ”	142
図 A.2	リンケージリスト “ program1.map ” の内容.....	143
図 A.3	リンケージリスト “ example.map ” の内容.....	146
表 3.1	リンケージエディタ実行上の注意事項.....	39
表 4.1	オプション / サブコマンド一覧表.....	52
表 6.1	インフォメーションメッセージ一覧表.....	119
表 7.1	ウォーニングメッセージ一覧表.....	124
表 7.2	エラーメッセージ一覧表.....	129
表 7.3	フェイタルエラーメッセージ一覧表.....	131
表 8.1	制限事項一覧表.....	137
表 A.1	入力ファイル一覧表.....	141
表 A.2	ライブラリ内モジュール一覧表.....	141

ライブラリアン編

第1章 概要.....	153
第2章 ライブラリアンの機能.....	157
2.1 ライブラリファイルの作成.....	159
2.2 既存ライブラリファイルの編集.....	160
2.3 ライブラリファイル内のモジュール抽出.....	162
2.4 ライブラリファイルの内容表示.....	163
第3章 ライブラリアンの実行.....	165
3.1 コマンドラインのフォーマット.....	169
3.2 コマンドライン指定による実行.....	170
3.3 サブコマンド指定による実行.....	171
3.3.1 会話形式による実行.....	171
3.3.2 サブコマンドファイルによる実行.....	172
3.4 ライブラリアンの終了.....	173
第4章 ライブラリアンのオプション / サブコマンド.....	175
4.1 オプション / サブコマンドのフォーマット.....	178
4.2 オプション / サブコマンドの種類.....	181
4.3 ファイル制御.....	185
4.3.1 LIBRARY - 編集対象のライブラリファイルの指定.....	185
4.3.2 OUTPUT - 出力ライブラリファイルの指定.....	186
4.3.3 DIRECTORY - ディレクトリ名の置き換えの指定.....	188
4.4 実行制御.....	189
4.4.1 SUBCOMMAND - サブコマンドファイルの指定.....	189
4.4.2 CREATE - ライブラリファイルの生成.....	190
4.4.3 ADD - モジュールの登録 / 追加.....	192
4.4.4 REPLACE - モジュールの置換.....	195
4.4.5 DELETE - モジュールの削除.....	198
4.4.6 EXTRACT - モジュールの抽出.....	199
4.4.7 RENAME - セクション名の変更.....	200
4.4.8 END - サブコマンド入力の終了指定.....	201
4.4.9 EXIT - ライブラリアンの終了指定.....	202
4.4.10 ABORT - ライブラリアンの強制終了指定.....	203
4.5 内容表示.....	204
4.5.1 LIST - ライブラリファイルの内容表示.....	204
4.5.2 SLIST - ライブラリファイルのセクション内容の表示.....	206
第5章 ライブラリアンの入力.....	207
5.1 オブジェクトモジュールファイル.....	209

5.2	リロケータブルロードモジュールファイル	210
5.3	ライブラリファイル	211
第 6 章	ライブラリアンの出力	213
6.1	ライブラリファイル	215
6.2	ライブラリアンリスト	216
6.3	セクション名リスト	219
6.4	コンソールメッセージ	221
第 7 章	エラーメッセージ	223
第 8 章	制限事項一覧	231
付録		
付録 A.	ライブラリアン使用例	237
A.1	コマンドライン指定による実行例	237
A.2	サブコマンド指定による実行例	239
付録 B.	ライブラリアン使用上の注意 (MS-DOS で使用する場合)	241

図表目次

図 2.1	ライブラリファイルの作成例	159
図 2.2	モジュールの追加	160
図 2.3	モジュールの削除	160
図 2.4	モジュールの置換	161
図 2.5	モジュールの抽出	162
図 6.1	ライブラリアンリストの形式	216
図 6.2	ライブラリアンリスト例 (“ S ” 指定あり、UNIX の場合)	218
図 6.3	ライブラリアンリスト例 (“ S ” 指定なし、UNIX の場合)	218
図 6.4	セクション名リストの形式	219
図 6.5	セクション名リスト例	220
図 A.1	コマンドライン指定例の実行結果	238
図 A.2	サブコマンド指定例の実行結果	240
表 3.1	コマンドラインの指定方法	169
表 4.1	オプション / サブコマンド一覧表	181
表 4.2	オプション / サブコマンドの相互関係	182
表 7.1	ウォーニングメッセージ一覧表	226
表 7.2	エラーメッセージ一覧表	227
表 7.3	フェイタルエラーメッセージ一覧表	229
表 8.1	制限事項一覧表	233

オブジェクトコンバータ編

第1章 オブジェクトフォーマットの変換	243
1.1 オブジェクトコンバータの機能	246
1.2 オブジェクトコンバータの実行	247
1.3 コンバートファイルの分割出力	250
1.4 エラーメッセージ	251

図表目次

図 1.1 S タイプオブジェクト形式	248
表 1.1 オブジェクトコンバータのエラーメッセージ	252

Ver.6.0 の追加・変更内容

第1章 リンケージエディタ	255
1.1 オブジェクトフォーマット	258
1.2 START オプション / サブコマンドの機能拡張	261
1.3 シンボルアドレス出力機能	263
1.4 中間ファイルのディレクトリ指定	265
1.5 仕様変更	266
第2章 ライブラリアン	267
2.1 オブジェクトフォーマット	269
第3章 オブジェクトコンバータ	271
3.1 オブジェクトフォーマット	274
3.2 コンバートファイルの分割出力	275
3.3 オプションの追加	276

図表目次

図 1.1 同一アドレスへの複数セクション割り付け	261
図 1.2 シンボルアドレス出力機能の使用例	264
表 1.1 新規機能一覧	257
表 1.2 使用可能なデバッグとオプション / サブコマンドの関係	258
表 3.1 新規機能一覧	273

索引277

リンケージエディタ編

1. 概要

第1章 目次

1.1	機能概要.....	5
1.2	オブジェクトモジュールとロードモジュール.....	6
1.3	ユニットとセクション.....	7

マイクロコンピュータ用のプログラムにおいても大規模なプログラムや複雑なプログラムの必要性が高くなるにしたがって、高級言語を使用したり、プログラムを分割して開発する機会が多くなってきます。このようなプログラム開発を実現するために、まずコンパイラまたはアセンブラを使用してソースプログラムをオブジェクトモジュールに変換します。続いてリンケージエディタを使用して複数のモジュールを結合し、目的とする1つのロードモジュールに編集します。

H シリーズ リンケージエディタ（以降リンケージエディタと略します）はアセンブラまたはCコンパイラが出力した複数のオブジェクトモジュールファイルを入力し、それらを結合および編集して、1つのロードモジュールファイルを出力します。

リンケージエディタを用いたプログラム開発手順の流れを図 1.1 に示します。

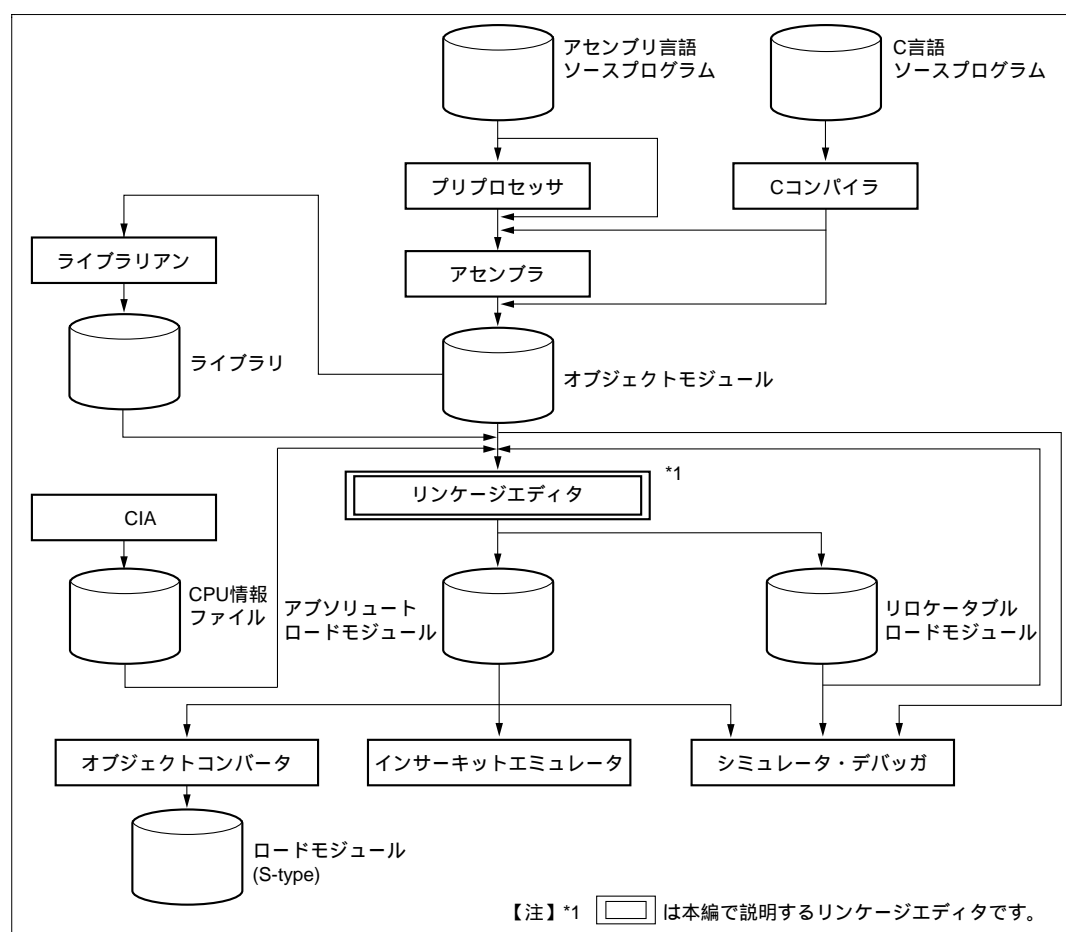


図 1.1 プログラムの開発手順

次にリンケージエディタの特長を示します。

- (1) コマンドライン指定による実行とサブコマンド指定による実行の2通りの実行形式を利用することにより、使用目的に合わせたリンケージエディタの制御が行えます。
- (2) リンケージエディタが出力したロードモジュールファイルを再び入力し、再編集したロードモジュールファイルを出力することができます。
- (3) シミュレータ・デバッガ、インサーキットエミュレータで使用するデバッグ情報を、オプションの指定によりロードモジュールファイル中に含ませることができます。

1.1 機能概要

リンケージエディタには次の5つの基本的な機能があります。

(1) モジュールの結合機能

コンパイラまたはアセンブラが出力したオブジェクトモジュールを結合、編集する機能です。

(2) アドレスの解決機能

各モジュール間の外部参照や相対アドレスで示されているアドレスを解決する機能です。

(3) ロードモジュールファイルの再入力機能

リンケージエディタが出力したロードモジュールファイルを再入力する機能です。

(4) マルチリンケージ機能

1度のリンケージエディタの起動中に複数のリンケージ処理を行う機能です。

(5) デバッグ援助機能

中間リンケージ結果の表示やエラーに対する暫定的な処置を行う機能です。

1.2 オブジェクトモジュールとロードモジュール

ソースプログラムをコンパイル(またはアセンブル)した結果として出力するものをオブジェクトモジュールと呼び、複数のオブジェクトモジュールをリンケージエディタによって結合したものをロードモジュールと呼びます。

ロードモジュールにはアブソリュートロードモジュールとリロケータブルロードモジュールの2種類の形式があります。アブソリュートロードモジュールは絶対番地が割り付けられた実行可能な形式で、再結合、再配置に必要なリロケーション情報を持ちません。リロケータブルロードモジュールは相対番地が割り付けられたものでリロケーション情報を持っており、再度リンケージエディタに入力し、再結合、再配置を行うことができます。ロードモジュール形式の選択は、リンケージエディタの FORM オプション / サブコマンドを使って指定します。FORM オプション / サブコマンドの詳細については、「4.5.3 FORM - 出力ロードモジュールファイル形式の指定」を参照してください。

これらのオブジェクトモジュール、アブソリュートロードモジュールおよびリロケータブルロードモジュールをまとめてモジュールと呼びます。

モジュールには、Hシリーズの品種により、結合時のアドレス割り付け方法の異なるページタイプと非ページタイプの2種類があります。H8/500シリーズがページタイプで、H8/300,H8SシリーズとSuperHシリーズが非ページタイプです。

1.3 ユニットとセクション

ユニットとはモジュール内のコンパイル（またはアセンブル）単位を意味します。したがってコンパイラ（またはアセンブラ）が出力したオブジェクトモジュールは、1 ユニット / 1 モジュールで構成されています。またリンケージエディタにより複数のオブジェクトモジュールを結合したロードモジュールは、1 つのモジュールの中に複数個のユニットが存在します。

セクションとはユニットを構成する最小単位で、リンケージエディタの処理の単位でもあります。

図 1.2 にモジュール、ユニットおよびセクションの関係を示します。

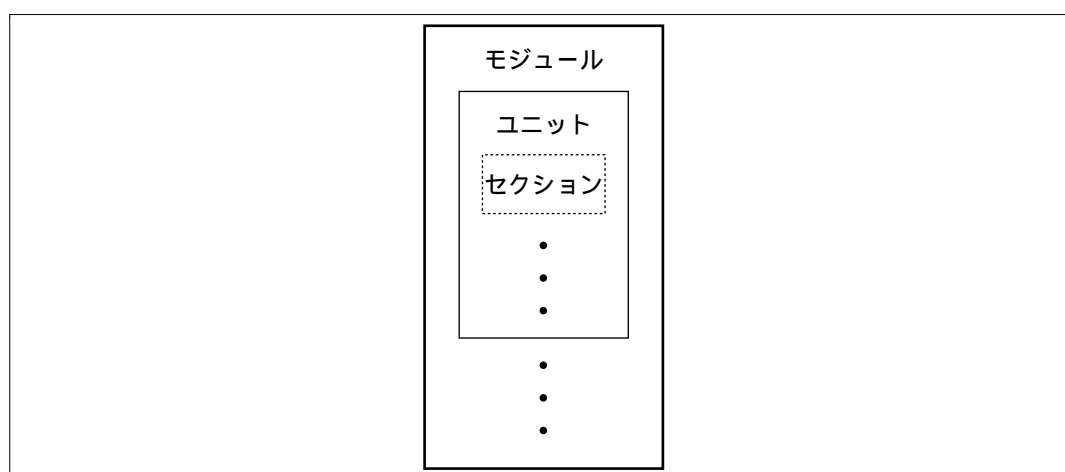


図 1.2 モジュール、ユニットおよびセクションの関係

セクションはセクションを識別するためのセクション名、セクションの内容、使用法を示すセクション属性、およびアブソリュートまたはリロケータブルの形式種別を持ちます。セクション名が同じでもセクション属性または形式種別が異なるセクションは、別セクションとして扱います。

セクション属性および形式種別は次のように分類されます。

(1) セクション属性

- ・コード …… 命令、定数エリア
- ・データ …… プログラムで値を変更する変数エリア
- ・スタック …… 初期化できないスタック、ワークエリア等
- ・コモン …… 複数のモジュールで共通に使用する変数エリア
- ・ダミー …… 実体を持たない変数構造定義部等

(2) 形式種別

- ・アブソリュート …… すでに絶対アドレスが割り付けられているセクション
- ・リロケータブル …… 絶対アドレスが決まっていないセクション

2. リンケージエディタの機能

第2章 目次

2.1	モジュールの結合機能.....	12
2.1.1	セクションの結合.....	12
2.1.2	ライブラリファイルからの入力.....	21
2.1.3	ライブラリファイル内のモジュールの結合抑止	23
2.2	アドレスの解決機能	24
2.2.1	外部参照の解決.....	24
2.2.2	セクション内のアドレス解決.....	26
2.2.3	アドレス未解決シンボルの表示抑止.....	27
2.3	ロードモジュールファイルの再入力機能	28
2.3.1	ユニットの自動置換	29
2.3.2	ユニットの強制置換	30
2.4	マルチリンケージ機能.....	31
2.5	デバッグ援助機能.....	32
2.6	アドレスチェック機能.....	33
2.7	ROM化支援機能	34

リンケージエディタの5つの基本的な機能については、「1.1 機能概要」で触れましたが、本章ではこれらの機能の詳細について述べます。本章の説明文や例の中で使用しているリンケージエディタを制御するためのオプションおよびサブコマンドの詳細については、「3. リンケージエディタの実行」および「4. リンケージエディタのオプション / サブコマンド」の各項を参照してください。

2.1 モジュールの結合機能

リンケージエディタは指定された入力ファイルからモジュールを読み出し、モジュール間の結合を行うことにより1つのロードモジュールを作成します。モジュールの結合は、モジュールを構成する要素として最小の単位であるセクションごとに処理を行います。

2.1.1 セクションの結合

形式種別がリロケータブルのセクションでのみセクションの結合を行います。形式種別がアブソリュートのセクションの場合には、すでに絶対アドレスが割り付けられているため、結合は行いません。セクションの結合は、次の手順で行います。

(1) 同一セクションの集合

ユニット間にまたがって同一名のセクションを集めます。

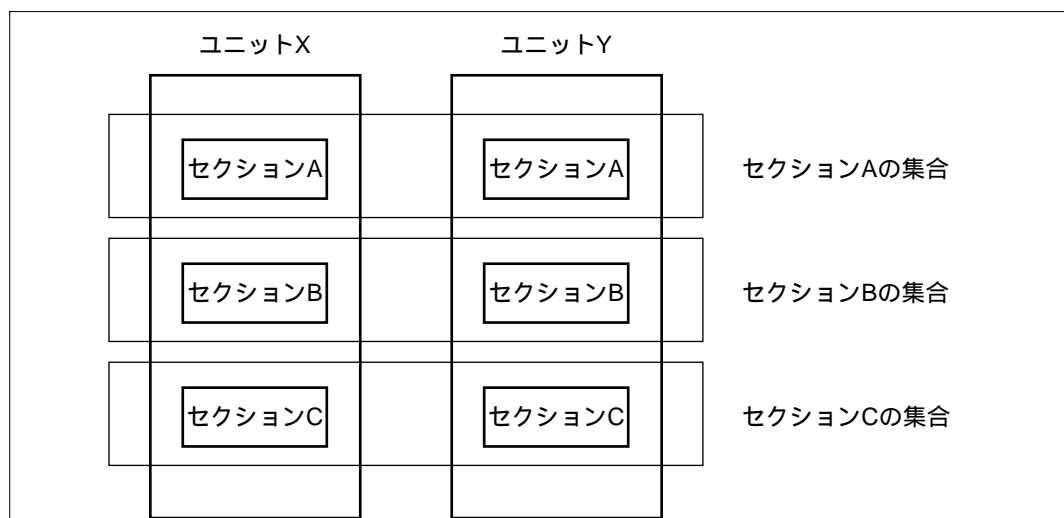


図 2.1 セクションの集合

もし、同じ名前を持つセクション同士でセクション属性が異なる場合には、ウォーニングメッセージを出力し、別セクションとして処理します。

(2) 同一セクションの結合

同一セクションを結合します。結合にはセクション属性により次の3種類の結合方法があります。

(a) 単純結合

セクション属性がコード、データ、およびスタックの場合、同一セクションを連続したアドレスに配置します。配置はモジュールの入力順になります。

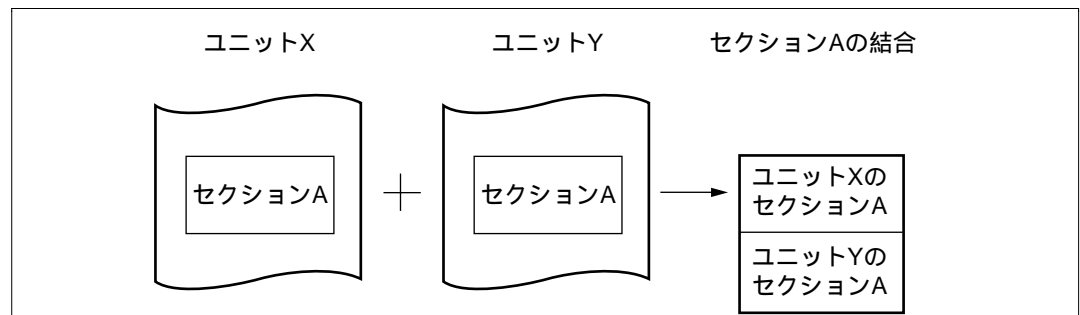


図 2.2 単純結合

(b) 共有結合

セクション属性がコモンの場合、同一セクションを同じアドレスに配置します。セクションの大きさは同一セクションの中の最大の大きさになります。

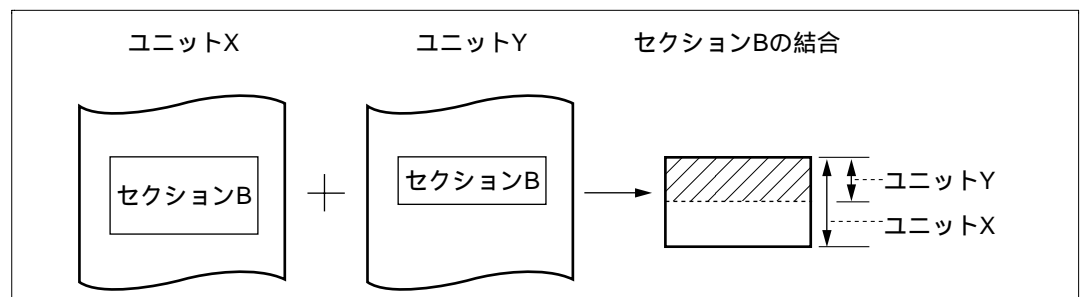


図 2.3 共有結合

(c) ダミー結合

セクション属性がダミーの場合、セクションは実体を持たないので結合を行いません。

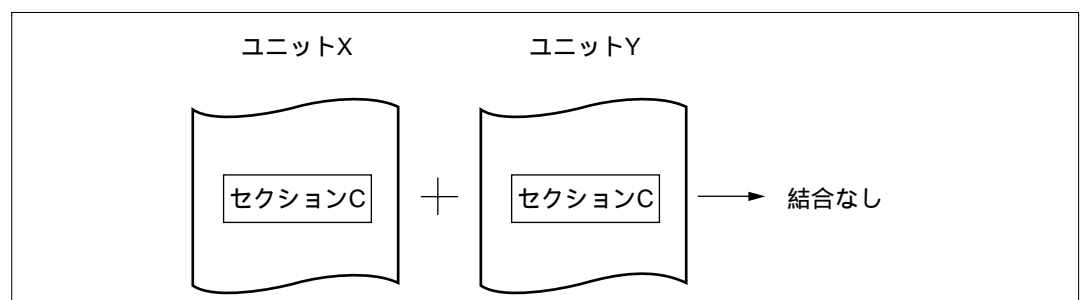


図 2.4 ダミー結合

(3) セクション同士の結合

リンケージエディタ実行時にセクションの結合順序が指定された場合は、その順序にしたがってセクションを結合します。セクションの結合順序の指定がない場合はセクションの出現順に結合します。

(a) 結合順序指定ありのとき

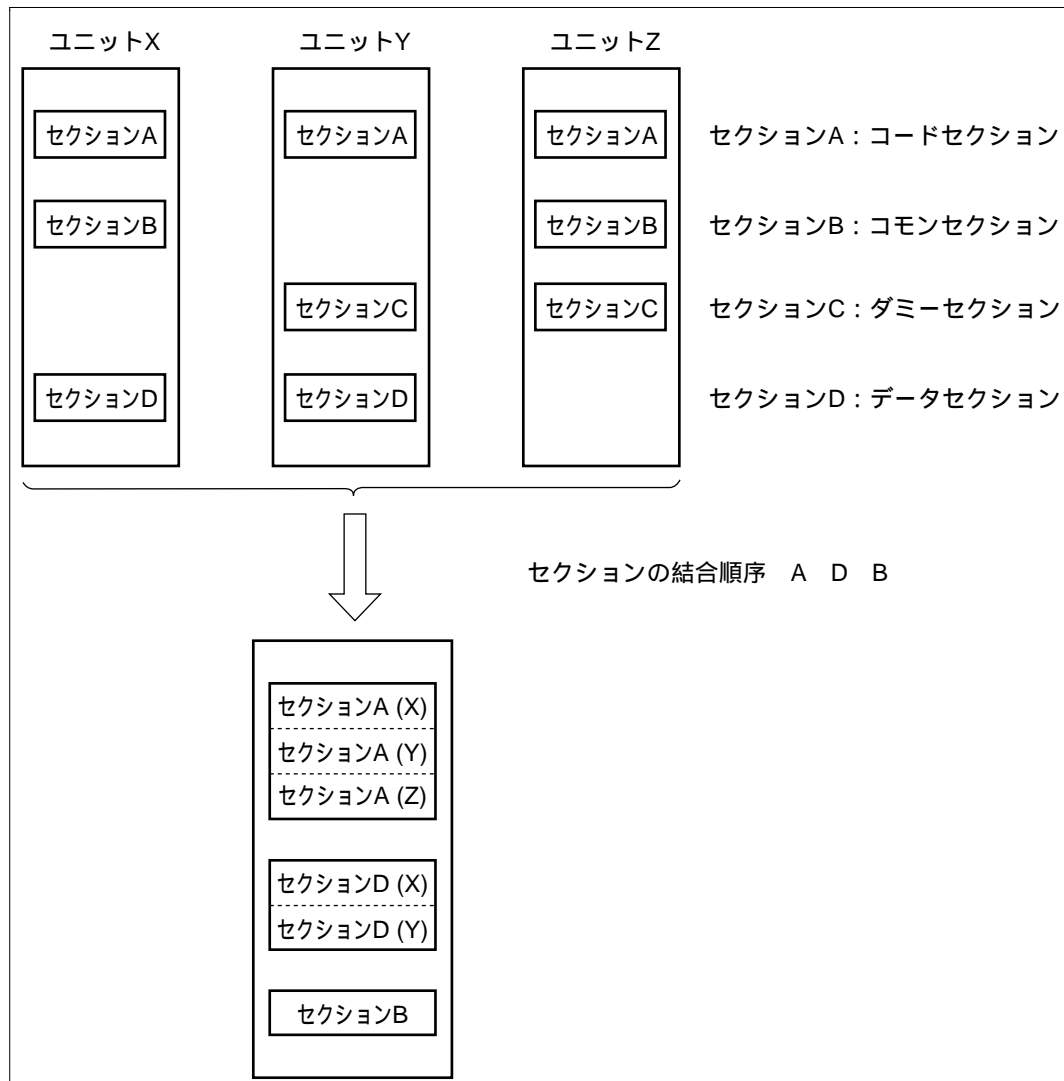


図 2.5 結合順序指定ありのセクション結合例

セクションの結合順序は、出力するロードモジュールがアブソリュート形式の時だけ指定できます。指定はSTART オプション / サブコマンドによって行います。

(b) 結合順序指定なしのとき

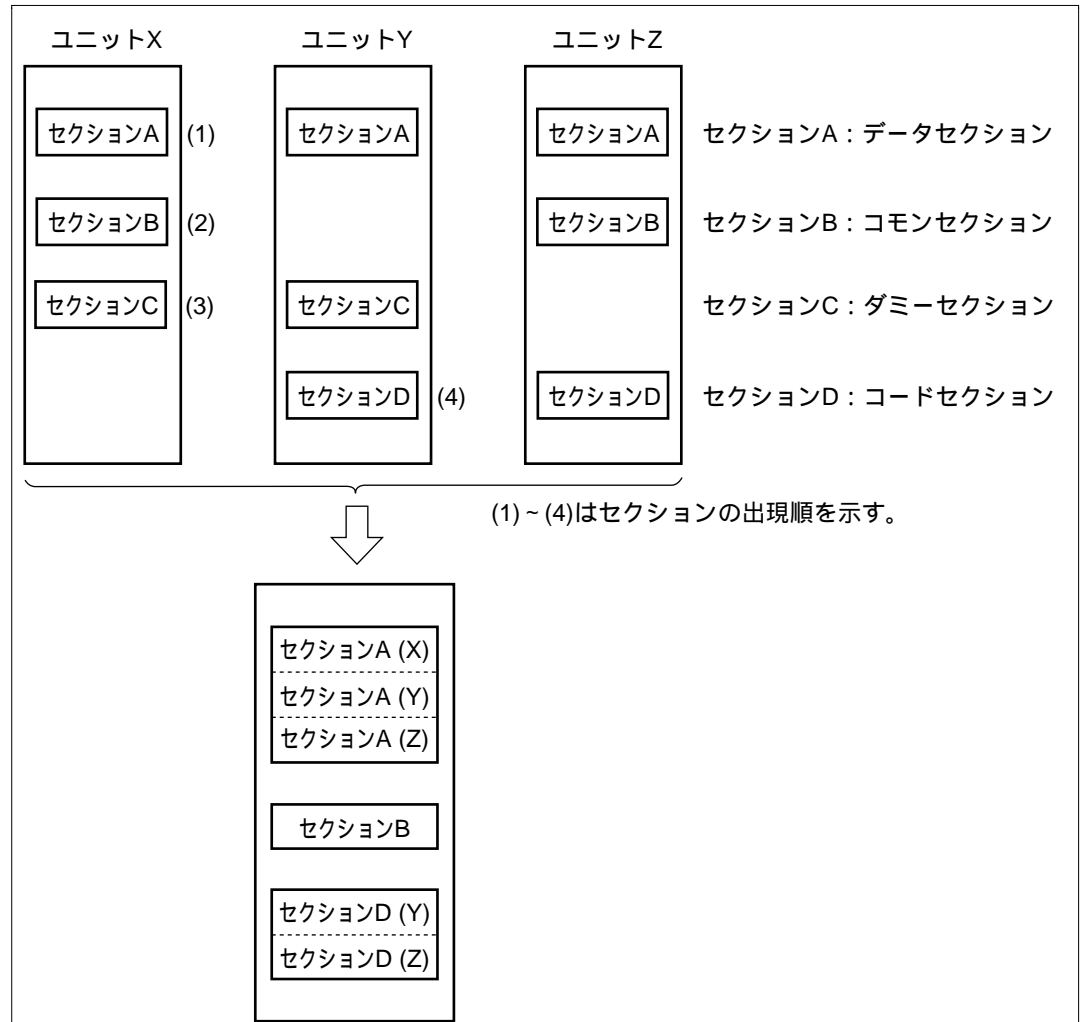


図 2.6 結合順序指定なしのセクション結合例

同一名を持つセクション同士でセクション属性が異なるものに関しては、別セクションとして出現順に結合します。

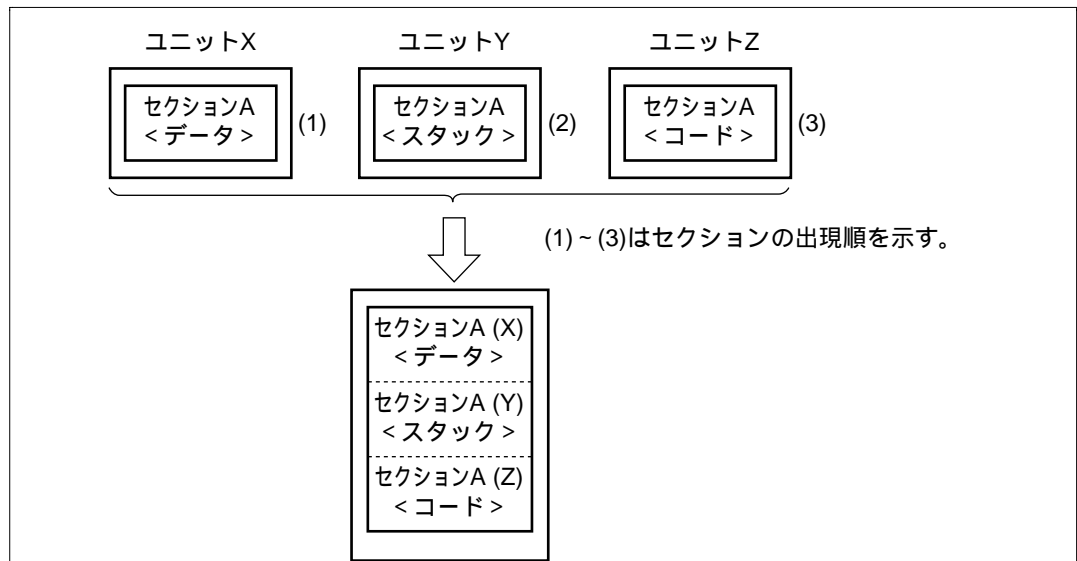


図 2.7 セクション属性の異なる同一名セクション結合例

(4) アドレスの割り付け

セクションごとにアドレスの割り付けを行います。出力するロードモジュールがアブソリュート形式の場合、絶対アドレスを割り付けます。セクションの結合順序と先頭アドレスが START オプション / サブコマンドによって指定されると、指定された先頭アドレスから順に各セクションに絶対アドレスを割り付けます。先頭アドレスの指定がない場合は、0 (ゼロ) 番地から絶対アドレスを割り付けます。

形式種別がアブソリュートのセクションと形式種別がリロケートブルのセクションを結合した場合、同じ絶対アドレスにセクションが重複して割り付けられることがあります。リンケージエディタでは重複した場合、ウォーニングメッセージを表示します。

ページタイプのモジュールの結合では、セクションごとにアドレスを割り付けていくと、1 つのセクションがページの境界をまたがる場合があります。リンケージエディタではページの境界にまたがってセクションを割り付けた場合、ウォーニングを出力します。しかし、このようにしてアドレスを割り付けたロードモジュールを実行するのは大変困難です。そのため、リンケージエディタではユニット単位でセクションがページの境界をまたがらないように自動的に次のページの先頭からセクションを割り付けることができます。これを自動ページングと呼び、AUTOPAGE オプション / サブコマンドで指定します。ページタイプのモジュールの結合での、アドレス割り付けの違いを図 2.8 (自動ページング指定なし、先頭アドレス指定なし)、図 2.9 (自動ページング指定あり、先頭アドレス指定なし)、および図 2.10 (自動ページング指定あり、先頭アドレス指定あり) に示します。

出力するロードモジュールがリロケートブル形式の場合、セクションごとにセクションの先頭からの相対アドレスを割り付けます。

ロードモジュールの出力形式は、FORM オプション / サブコマンドで指定します。

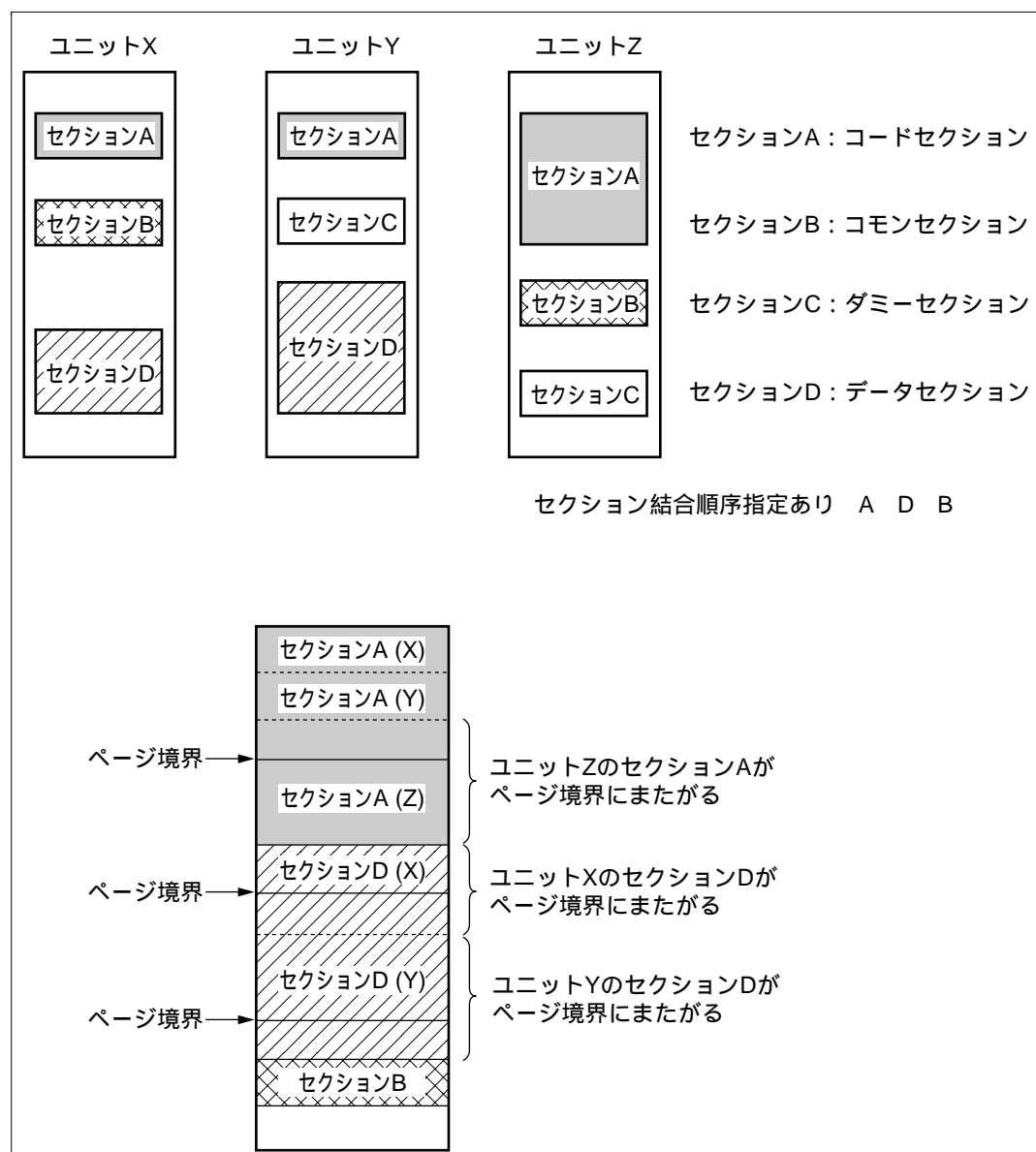


図 2.8 ページタイプの結合例 (自動ページング指定なし、先頭アドレス指定なし)

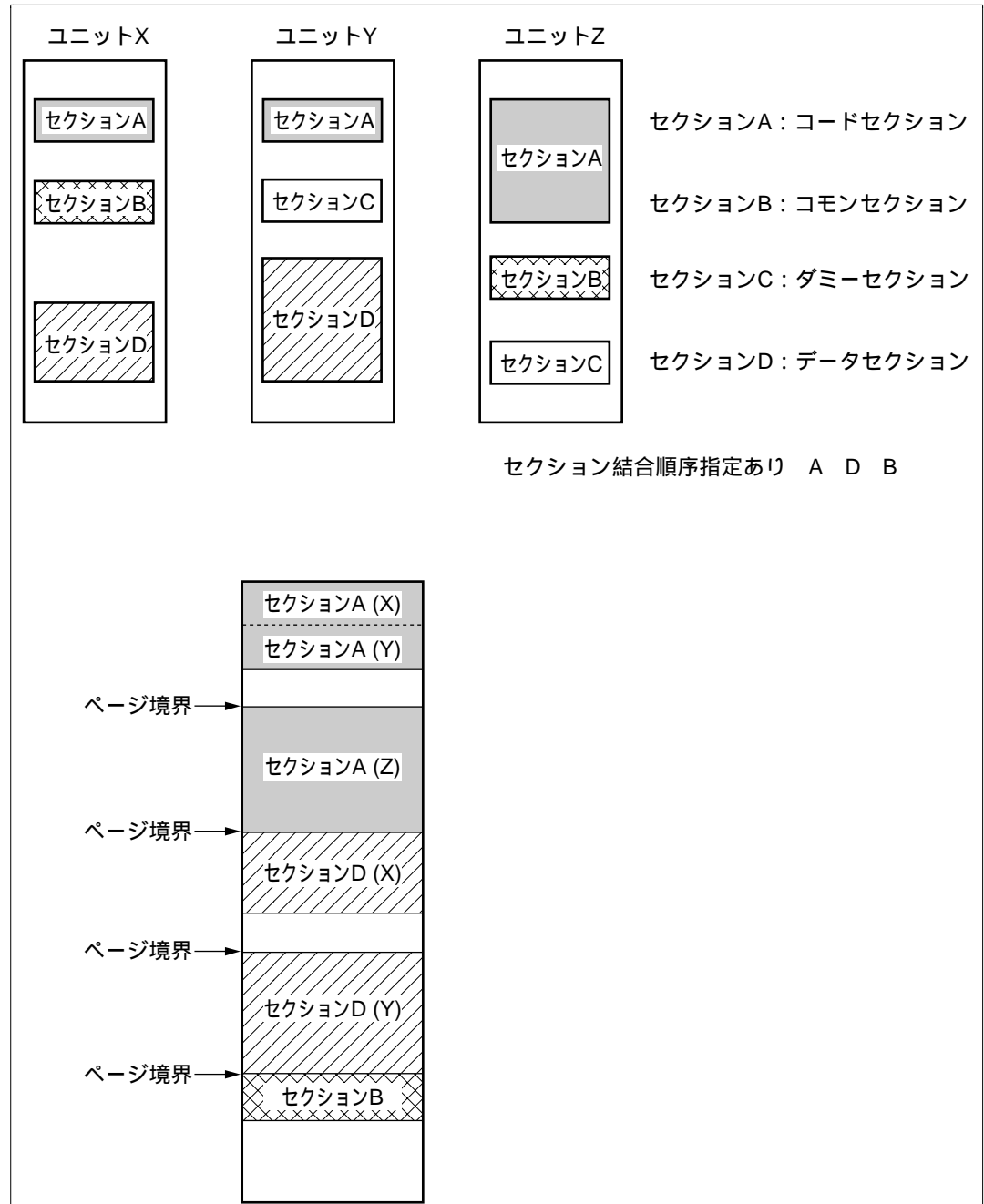


図 2.9 ページタイプの結合例 (自動ページング指定あり、先頭アドレス指定なし)

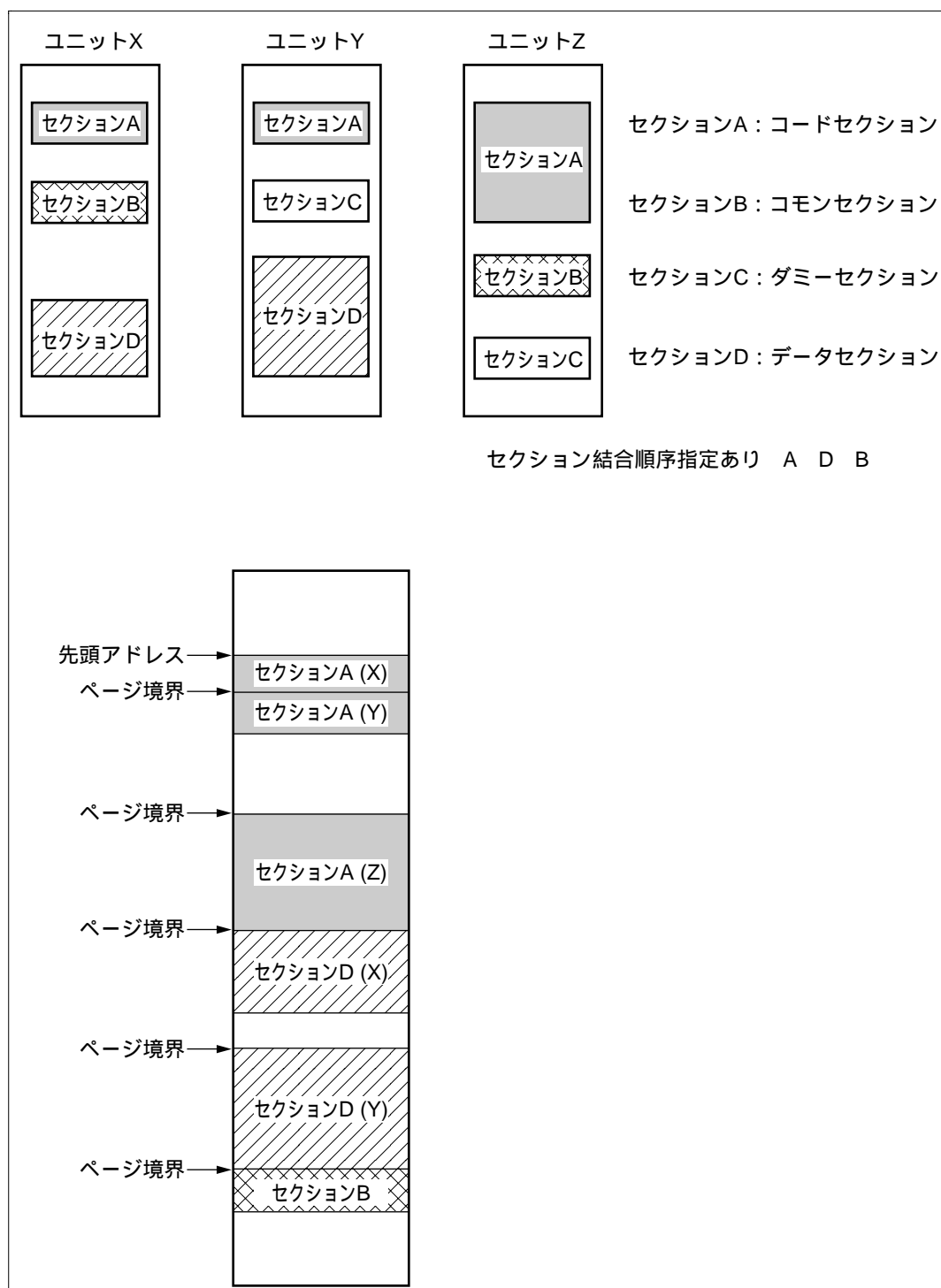


図 2.10 ページタイプの結合例（自動ページング指定あり、先頭アドレス指定あり）

2.1.2 ライブラリファイルからの入力

リンケージエディタではライブラリアンで作成したライブラリファイルからオブジェクトジュールまたはリロケータブルロードモジュールを入力し、モジュールの結合を行うことができます。ライブラリファイルからの入力には次に示す2つの方法があります。

(1) モジュール名指定による入力

入力ファイル名を指定するときに、ライブラリファイル名とモジュール名を指定することにより、ライブラリファイル中の特定モジュールを入力することができます。入力ファイルの指定は、コマンドラインあるいはINPUT サブコマンドによって行います。

(2) 自動入力

リンケージエディタは指定されたモジュールをすべて入力した後で外部参照シンボルの解決処理を行います。どのモジュールにおいても定義されていない外部参照シンボルがある場合、まず指定されたライブラリファイルをサーチし、ライブラリファイルの中から未解決外部参照シンボルを定義しているモジュールを自動的に入力し、結合処理を行います。もし、ライブラリファイルの中に未解決外部参照シンボルを定義しているモジュールが存在しない場合、ユーザによってあらかじめ定義されたデフォルトライブラリ内をサーチし、未解決外部参照シンボルを定義しているモジュールを自動的に入力し、結合処理を行います。

デフォルトライブラリの中に未解決外部参照シンボルを定義しているモジュールが存在しない場合、未定義エラーとなります。

デフォルトライブラリについては、「5.4 デフォルトライブラリファイル」を参照してください。

また、ライブラリファイルにはシステムライブラリファイルとユーザライブラリファイルの区別があり、リンケージエディタはユーザライブラリファイルから先にサーチします。そのため、同じ名前の外部定義シンボルを含むモジュールが、指定されたシステムライブラリファイルとユーザライブラリファイルの両方にある場合、ユーザライブラリ中のモジュールを結合します。なお、システムライブラリファイル同士またはユーザライブラリファイル同士の場合は指定順にサーチします。

ライブラリファイル中にページタイプのモジュールと非ページタイプのモジュールを混在して作成することができますが、リンケージエディタは2種類のタイプのモジュールを同時に入力した場合はエラーとしますので、ライブラリファイルの作成およびライブラリファイルの指定時には十分に注意してください。

ライブラリファイルの指定は、LIBRARY オプション / サブコマンドによって行います。

また、システムライブラリファイルとユーザライブラリファイルの指定方法は、本マニュアルのライブラリアン編を参照してください。

次にライブラリファイルを指定した場合のモジュールの結合順について例を用いて説明します。

(1) INPUT サブコマンドにより、オブジェクトモジュール a、b を入力します。

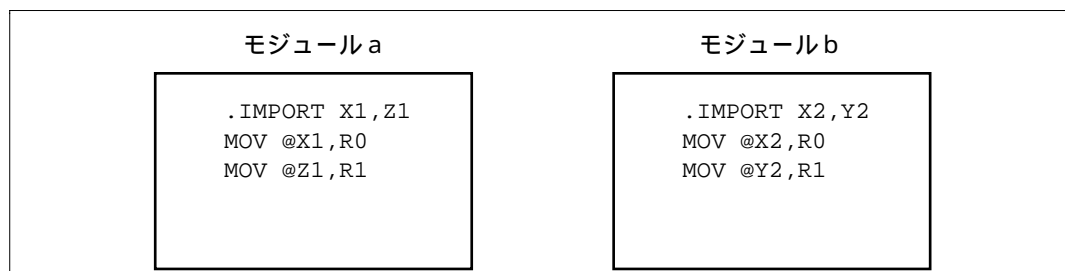


図 2.11 モジュール結合の例 (入力オブジェクトモジュール)

(2) LIBRARY サブコマンドにより、lib1、lib2、lib3 の順に入力します。



図 2.12 モジュール結合の例 (入力ライブラリファイル)

(3) 各入力ファイルで宣言した外部参照シンボルをすべて集めた後、最初に指定したライブラリに外部定義シンボルがないかサーチします。シンボルがある場合は、外部定義シンボルを含むモジュールを結合します。複数のシンボルが同一ライブラリのそれぞれ別モジュールに宣言されているときは、ライブラリ内の出現順にモジュールを結合します。

当該ライブラリにシンボルがない場合は、次に指定したライブラリ内をサーチします。

上記例では、以下の結合順になります。

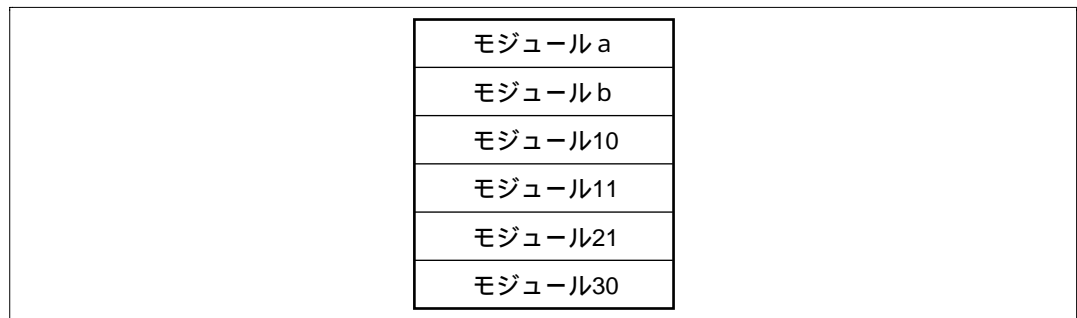


図 2.13 モジュール結合の例（出力ロードモジュール）

2.1.3 ライブラリファイル内のモジュールの結合抑止

参照のない外部参照シンボルがあるとき、それを定義しているモジュールを結合する可否をオプション / サブコマンドにより選択することができます。下記のコーディング例でシンボル abc は、外部参照シンボルの宣言はありますが、実行文での参照がありません。結合抑止を指定した場合、シンボル abc を定義しているライブラリファイル内のモジュールは結合しません。

```

      .IMPORT  xyz,abc
      MOV.W   @xyz,R0
      .
      .
      .END

```

図 2.14 参照のない外部参照シンボルを含むモジュール例

C 言語でコーディングするとき、extern により外部参照シンボルを記述しますが、そのシンボルを参照していないことがあります（例えば、stdio.h には参照のない外部参照シンボルが多数記述されています）。本機能を使用することにより不要なモジュールの結合を抑止できるため、プログラムサイズを小さくすることができます。モジュールの結合抑止の指定は、EXCLUDE オプション / サブコマンドで行います。

2.2 アドレスの解決機能

あるモジュールから別のモジュール内にあるシンボルを参照する場合や、同じモジュール内のシンボルでもリロケータブルセクション内のシンボルを参照する場合には、アセンブル時にそれらのシンボルの絶対アドレスは決定できません。リンケージエディタではそれらのシンボルに対して絶対アドレスを決定し、参照位置へその絶対アドレスを設定します。

2.2.1 外部参照の解決

モジュール内で別のモジュール内にあるシンボルを参照する場合、アセンブラはオブジェクトプログラム内に外部参照情報を出力します。一方、別のモジュールから参照されるシンボルについて外部定義の宣言をすることにより、オブジェクトプログラム内に外部定義情報が出力されます。リンケージエディタは、これらの外部参照情報と外部定義情報を結び付けます。さらに、オプションまたはサブコマンドによって指定されたアドレス情報により、定義シンボルの絶対アドレスを決定し、参照シンボルへその絶対アドレスを設定します。

図 2.15 に外部参照の解決の例を示します。図 2.15 中での各モジュール、セクションおよび指定するサブコマンドは次に示すとおりです。

(1) モジュール a

- ・ セクション X のみで構成され、そのサイズは H'5000 バイト
- ・ A1 の位置からモジュール b の S4 を外部参照
- ・ A2 の位置からモジュール b の S2 を外部参照

(2) モジュール b

- ・ セクション X とセクション Y で構成
- ・ セクション X のサイズは H'2000 バイト
- ・ セクション Y のサイズは H'3000 バイト
- ・ S1 はセクション Y の先頭で、S2 はセクション Y の先頭から H'1000 バイトの位置
- ・ S3 はセクション X の先頭で、S4 はセクション X の先頭から H'1200 バイトの位置

(3) モジュール c

- ・ セクション Z のみで構成され、そのサイズは H'4000 バイト
- ・ C1 の位置からモジュール b の S3 を外部参照
- ・ C2 の位置からモジュール b の S1 を外部参照

(4) サブコマンド

```

INPUT  a,b,c

START  X,Y,Z(10000)

EXIT

```

a、b、cの3つのモジュールを入力します。X、Y、Zの順にセクションを結合し、先頭アドレスをH'10000番地とします。

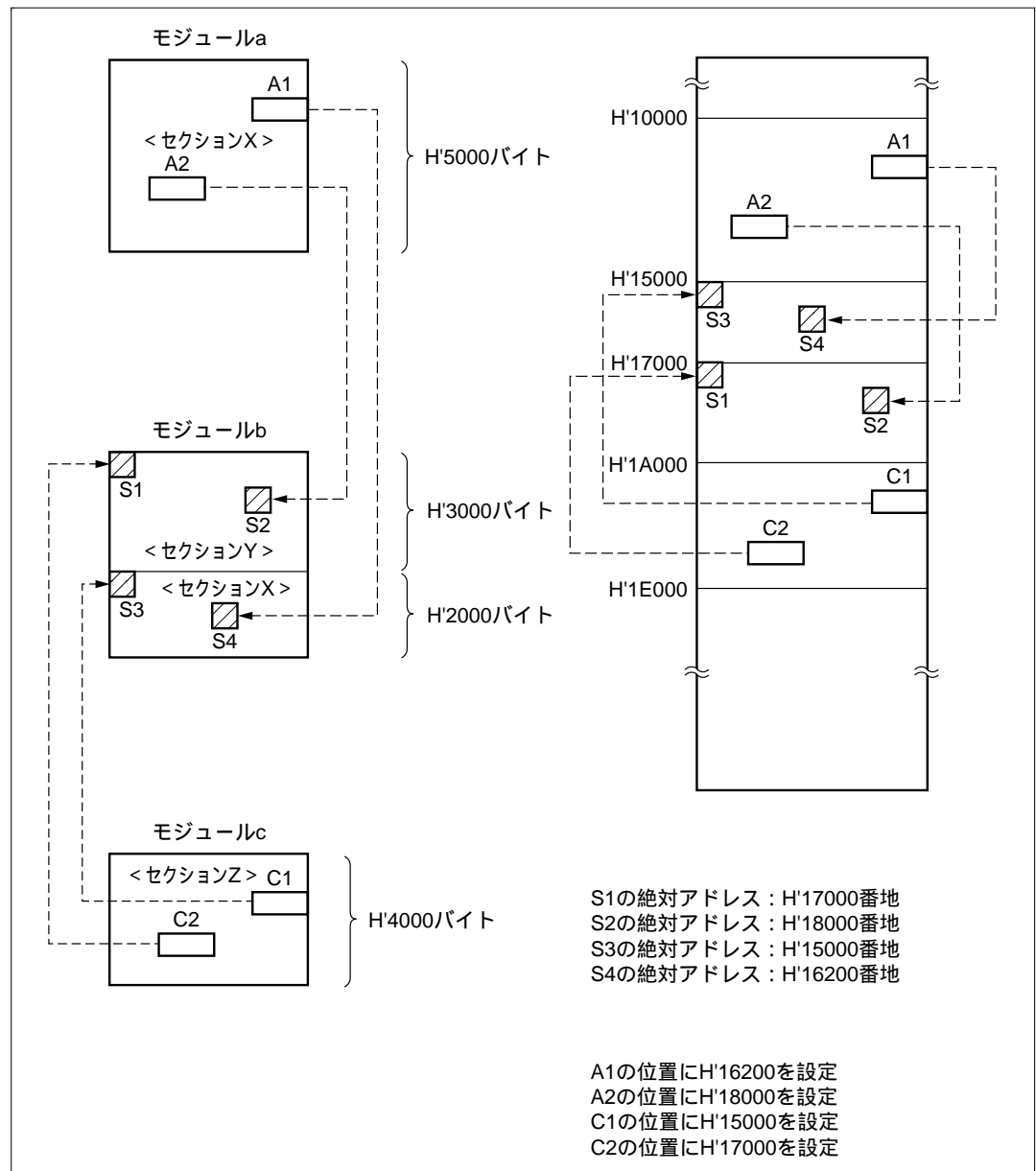


図 2.15 外部参照の解決

2.2.2 セクション内のアドレス解決

モジュール内のリロケータブルなセクション中で定義したシンボルを同じモジュール内で参照した場合、アセンブラはそのアドレスをセクションの先頭からの相対アドレスで表します。

リンケージエディタではこの相対アドレス値と、オプションまたはサブコマンドによって指定されたアドレス情報を使用して絶対アドレスを決定し、設定しなおします。

図 2.16 にセクション内のアドレス解決の例を示します。図 2.16 中での各モジュール、セクションおよび指定するサブコマンドは次に示すとおりです。

(1) モジュール a

- ・セクション X のみで構成され、そのサイズは H'5000 バイト

(2) モジュール b

- ・セクション X、セクション Y およびセクション Z で構成
- ・セクション X のサイズは H'6000 バイト
- ・セクション Y のサイズは H'1000 バイト
- ・セクション Z のサイズは H'2000 バイト
- ・B1 の位置から S1 を参照
- ・B2 の位置から S3 を参照
- ・B3 の位置から S2 を参照
- ・S1 はセクション X の先頭から H'3000 バイトの位置
- ・S2 はセクション X の先頭から H'4500 バイトの位置
- ・S3 はセクション X の先頭から H'5000 バイトの位置

(3) サブコマンド

```
INPUT  a,b

START  X,Y,Z(10000)

EXIT
```

a、b の 2 つのモジュールを入力します。X、Y、Z の順にセクションを結合し、先頭アドレスを H'10000 番地とします。

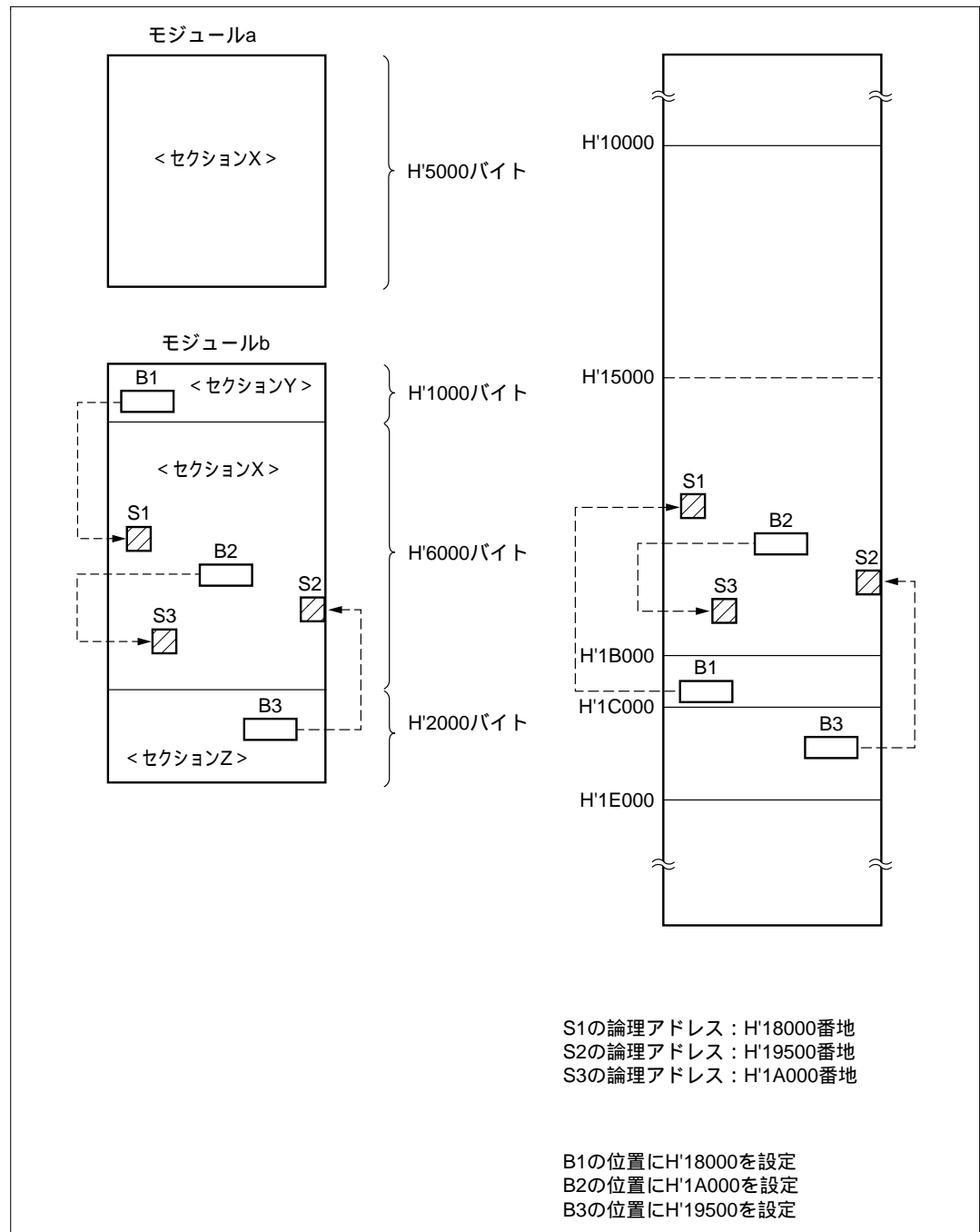


図 2.16 セクション内のアドレス解決

2.2.3 アドレス未解決シンボルの表示抑止

リロケータブルロードモジュール指定の場合に限り、未解決なシンボル名の表示を抑止することができます。これは UDF オプション / サブコマンドにより選択することができます。

2.3 ロードモジュールファイルの再入力機能

プログラムの変更を行った場合や外部参照シンボルが未解決のまま残ってしまった場合には、再びリンケージエディタを使用してロードモジュールファイルを作りなおさなければいけません。このとき、ひとつひとつのオブジェクトモジュールファイルを指定しなおすのではなく、すでに作られているロードモジュールファイルと変更のあったオブジェクトモジュールファイル（あるいは、外部定義シンボルを含んでいるオブジェクトモジュールファイル）だけを指定することで、ロードモジュールファイルを作成できる機能がロードモジュールファイルの再入力機能です。

再入力機能でモジュールの置き換えがある場合は、ユニット単位で置き換わります。ユニットの置換については、「2.3.1 ユニットの自動置換」を参照してください。

再入力するロードモジュールファイルの指定は、オブジェクトモジュールファイルの入力指定と同様にコマンドラインあるいはINPUTサブコマンドで行います。

再入力できるロードモジュールファイルはリロケータブル形式のものだけです。リロケータブル形式のロードモジュールファイルを作成する場合は、FORM オプション / サブコマンドでファイル形式の指定を行います。

図 2.17 にロードモジュールファイルの再入力の概略を示します。

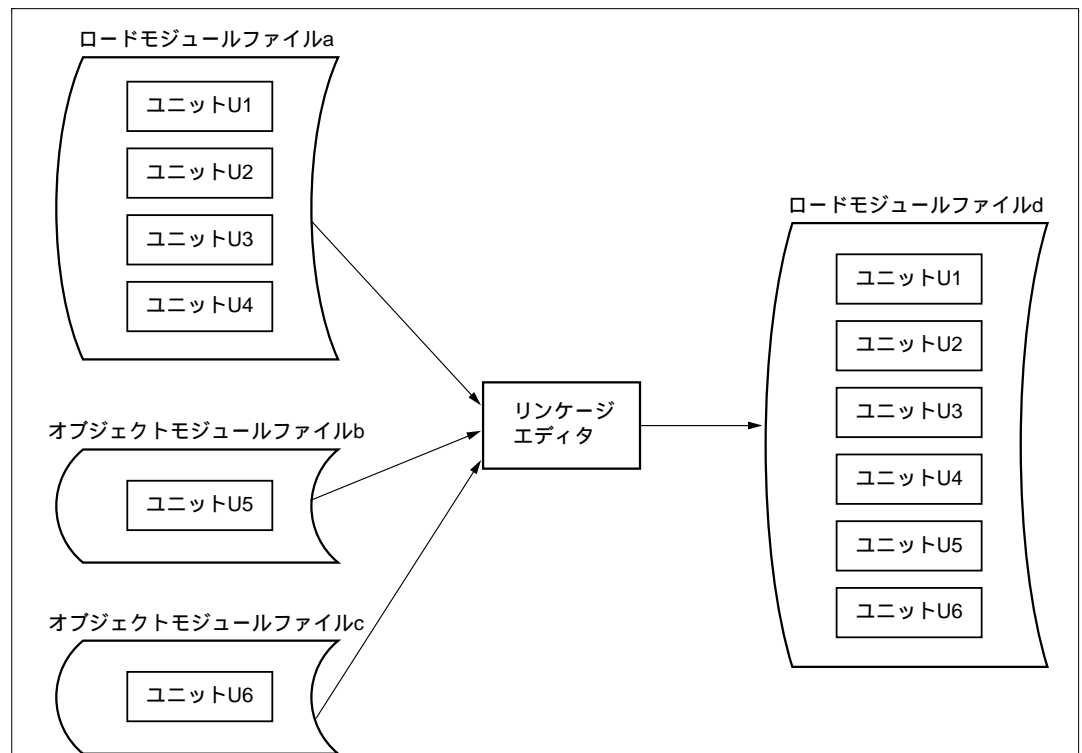


図 2.17 ロードモジュールファイルの再入力機能

リンケージエディタがロードモジュールファイル a とオブジェクトモジュールファイル b および c の 3 ファイルを入力し、新たなロードモジュールファイル d を出力します。ロードモジュールファイル d は、ユニット U1、U2、U3、U4、U5 および U6 で構成されます。

2.3.1 ユニットの自動置換

リンケージエディタは複数のモジュール中に同じ名前のユニットがあった場合には、先に指定されているモジュール中のユニットを優先して取り込みます。そこで、ロードモジュールファイル中のユニットを置き換えたい場合には、まず置き換えるユニットを含むファイルを指定し、続いて当該ロードモジュールファイルを指定すれば、ユニットが置き換わったのと同じことになります。これをユニットの自動置換と呼びます。

ユニットの自動置換を利用すれば、デバッグ時などプログラムの修正が頻繁にある場合に、入力するファイルの指定順序を変えるだけで容易に新しいロードモジュールファイルを作成することができます。

図 2.18 にユニットの自動置換の例を示します。

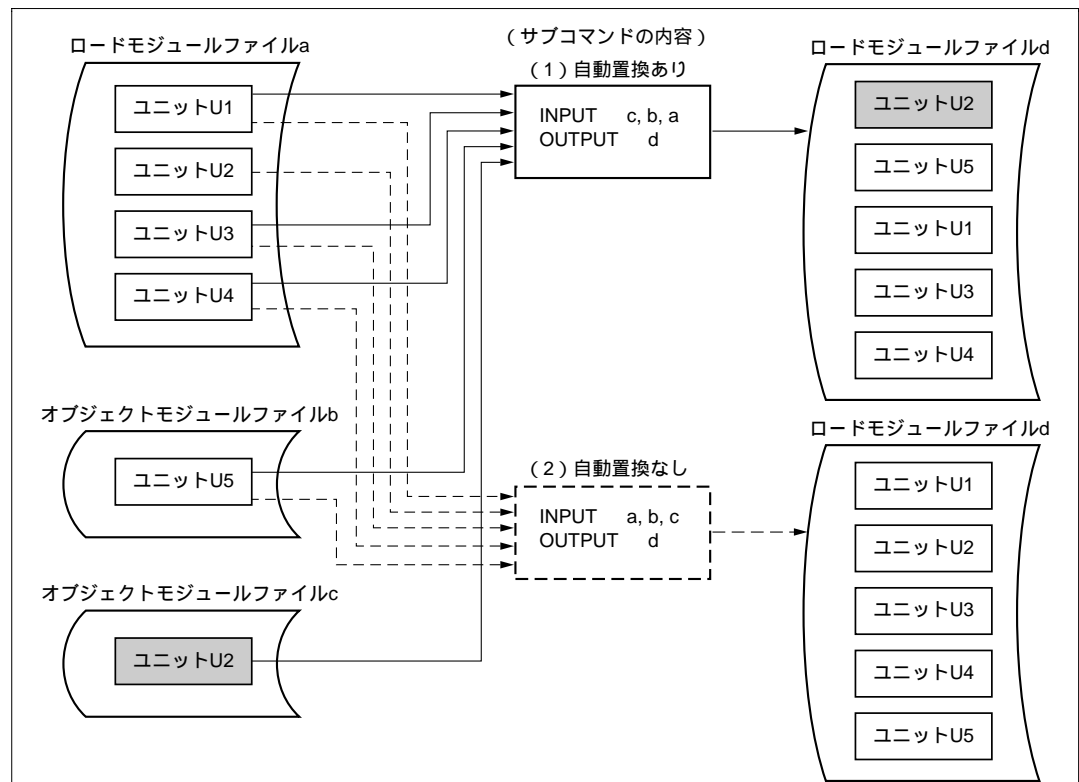


図 2.18 ユニットの自動置換

(1) 自動置換あり

オブジェクトモジュールファイルc、b、ロードモジュールファイルaの順で入力します。ロードモジュールファイルa中のユニットU2は、すでにオブジェクトモジュールファイルcより入力済みのため、ファイルa中のユニットU2は取り込みません。

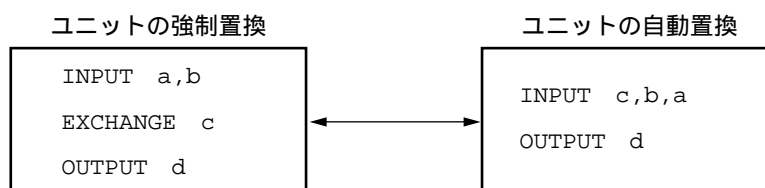
(2) 自動置換なし

ロードモジュールファイルa、オブジェクトモジュールファイルb、cの順で入力します。オブジェクトモジュールファイルc中のユニットU2は、すでにロードモジュールファイルaより入力済みのため、ファイルc中のユニットU2は取り込みません。

2.3.2 ユニットの強制置換

ユニットの置き換えをする場合、ユニットの自動置換を利用するのではなく置き換えるユニットをサブコマンドで指定することもできます。これをユニットの強制置換と呼び、EXCHANGE サブコマンドを使って指定します。

次のサブコマンドを指定すれば、前節の図 2.18 のユニットの自動置換と同じ内容をユニットの強制置換を使って行えます。



上記のユニットの強制置換の例では、ロードモジュールファイル a 中のユニット U1、U2、U3、U4 およびオブジェクトモジュールファイル b 中のユニット U5 を入力後、入力済みのユニット U2 をオブジェクトモジュールファイル c 中のユニット U2 に強制置換します。出力ロードモジュールファイル d にはファイル a 中のユニット U1、U3、U4、ファイル b 中のユニット U5、およびファイル c 中のユニット U2 が含まれます。

したがって、ロードモジュールファイル d は、図 2.18 の自動置換の例のロードモジュールファイル d と同じユニットの構成内容になります。

2.4 マルチリンケージ機能

リンケージエディタが1度のリンケージ処理で扱える入力ファイルの数は256個までです。複数の入力ファイルを指定する場合には、ロードモジュールファイルの再入力によるリンケージ処理を行うことができます。このときリンケージ処理ごとにリンケージエディタを起動させるのではなく、1度のリンケージエディタの起動で、複数のリンケージ処理が行える機能がマルチリンケージ機能です。

マルチリンケージ機能で1つのリンケージ処理の終了は、END サブコマンドで示します。ただし、最後のリンケージ処理の終了は、EXIT サブコマンドで示します。

図 2.19 にマルチリンケージ機能の例を示します。

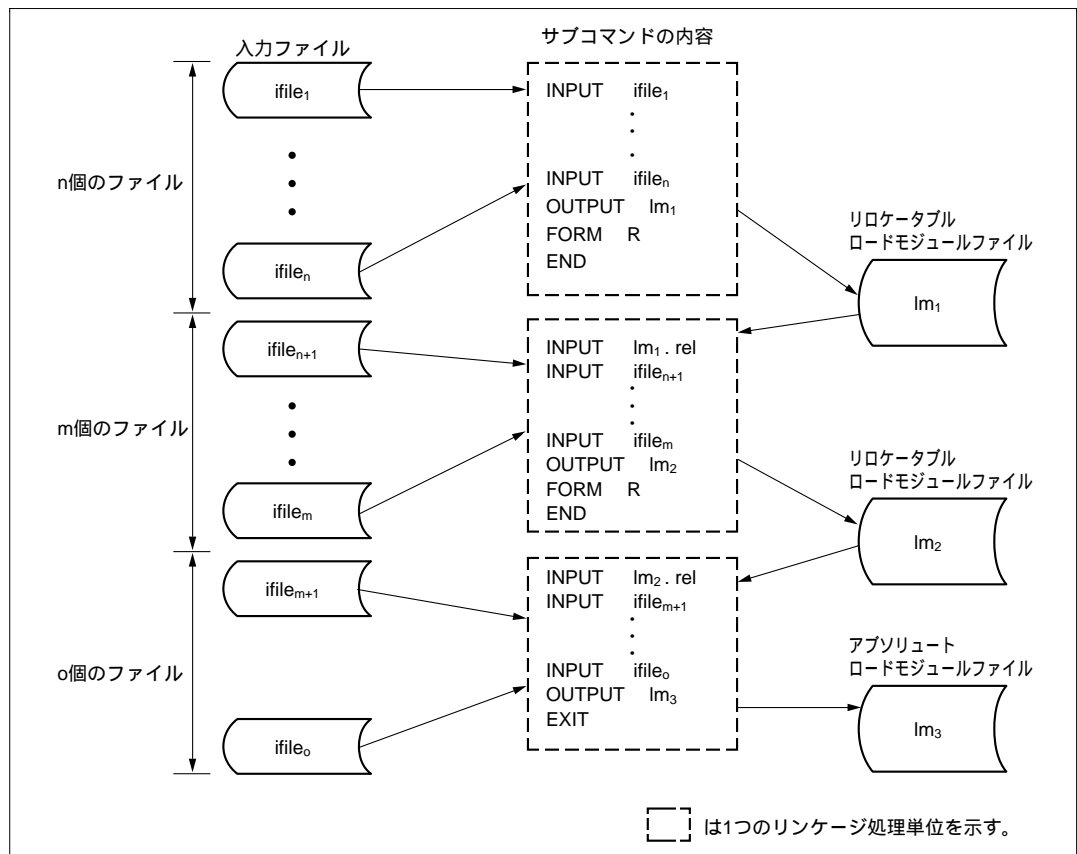


図 2.19 マルチリンケージ機能

【注】 マルチリンケージ中にデフォルトライブラリ機能を併用すると最初のリンケージ処理でデフォルトライブラリ内のモジュールを取り込みます。最後のリンケージ処理でデフォルトライブラリを取り込みを行いたいときは、途中のリンケージ処理ではNOLIBRARY サブコマンドを指定してください。

2.5 デバッグ援助機能

デバッグ援助機能はプログラムデバッグ段階で中間リンケージ結果を確認したり、エラーのあるロードモジュールに対して暫定的な回復措置を施す場合に利用します。デバッグ援助機能には、中間リンケージ情報の表示と変更および削除、外部シンボル名等の定義があります。

以下に各機能の内容を示します。

(1) 中間リンケージ情報の表示

サブコマンド入力途中でリンケージ処理中のロードモジュールの情報を見たい場合に利用します。LIST サブコマンドを指定すると、中間リンケージ情報を標準出力へ表示します。

表示するリンケージ情報には、次の3種類の情報があります。

- (a) リンケージマップ
- (b) 未解決の外部参照シンボル
- (c) 外部定義シンボル

(2) ユニット名 / 外部定義シンボル名 / 外部参照シンボル名の変更および削除

ユニット名、外部定義シンボル名または外部参照シンボル名が重複してしまった場合にその名前を変更したり、削除することができる機能です。ただし、外部参照シンボル名の削除はできません。

変更は RENAME サブコマンドで指定し、削除は DELETE サブコマンドで指定します。

(3) 外部参照シンボルの強制定義

外部参照シンボルに対して暫定的に値を定義する場合に利用します。この機能を利用して定義した値は当該リンケージ処理でのみ有効です。

外部参照シンボルの強制定義は DEFINE オプション / サブコマンドで指定します。

2.6 アドレスチェック機能

リンケージエディタでオブジェクト形式のロードモジュールを作成する場合、セクションのアドレスの割り付けが、対象となる CPU のメモリマップに合わないと、このロードモジュールはメモリにロードできません。

リンケージエディタには、CPU のメモリマップ情報（以降 CPU 情報とします）をファイルから読み出し、この情報に基づいて、セクションに割り付けられたアドレスが有効か否かをチェックする機能があります。

アドレスチェックを実行する場合は、CPU オプション / サブコマンドで CPU 情報ファイルを指定します。CPU 情報ファイルは、シミュレータ・デバッガに含まれる CPU 情報解析プログラム（CIA）を使用して作成します。

H8S,H8/300 シリーズ、SuperH シリーズ以外の CPU については、CPU 情報解析プログラムをサポートしていないため、アドレスチェック機能は使用できませんのでご注意ください。

なお、CPU 情報ファイルの作成方法については、「H8S,H8/300 シリーズ シミュレータ・デバッガ ユーザーズマニュアル」または「SH シリーズ シミュレータ・デバッガ ユーザーズマニュアル」を参照してください。

2.7 ROM 化支援機能

C 言語でコーディングしたユーザが、ロードモジュールを ROM に書き込みたい場合、初期値を持ったデータ領域（以下 D セクションとします）も ROM に書き込まれてしまいます。このため、リンケージエディタでは、ROM 化を支援する機能として以下の処理を行い、ユーザの負担を軽減しています。

- (1) D セクションと同じ大きさの領域（D' セクションとします）を RAM 用として出力ロードモジュールに確保します。ロードモジュールのメモリマップは、以下のようになります。

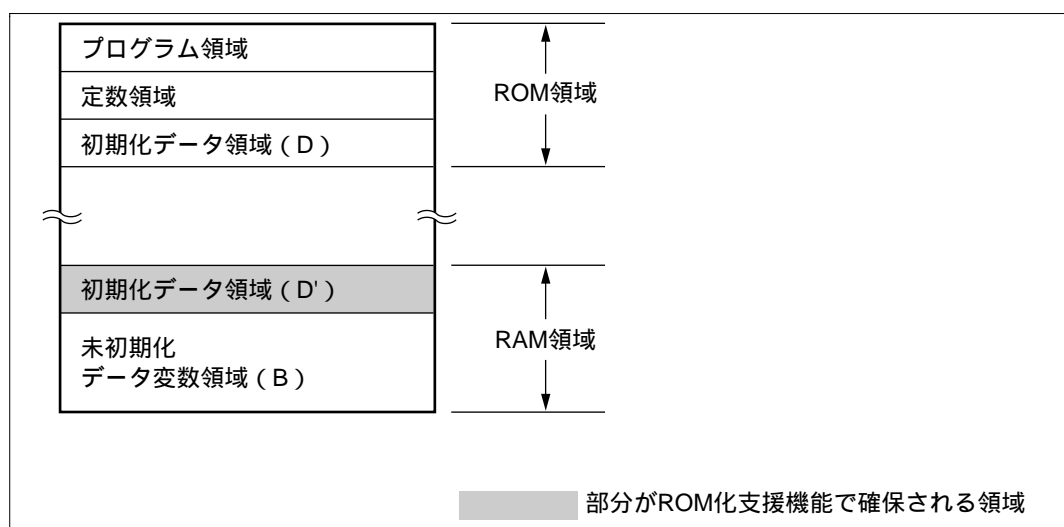


図 2.20 ROM 化支援機能を使ったときのメモリマップ

- (2) D セクション内に宣言した変数を参照する場合、RAM 領域をポイントするようにアドレスを変更します。変数のアドレスは、

D セクションの先頭アドレス + セクション内相対アドレス

でしたが、ROM 化支援機能より

D' セクションの先頭アドレス + セクション内相対アドレス

になります。

(例) MOV @a, R0

D セクションに宣言したシンボル a のアドレスは、図 2.21 のように (x) + (y) となります。

オブジェクトコード上にもこのアドレスが格納されます。

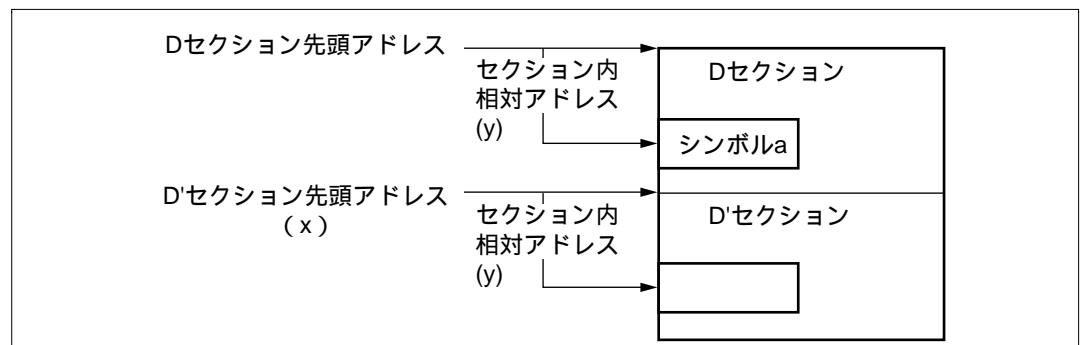


図 2.21 ROM 化支援機能を使ったときのシンボルのアドレス

(3) スタートアップルーチンにより、ROM 上のデータを RAM 上にコピーします。

スタートアップルーチンにコピー処理を組み込むことで実現します。組み込み方法は、C コンパイラのユーザーズマニュアルを参照してください。

3. リンケージエディタの 実行

第3章 目次

3.1	コマンドラインのフォーマット	40
3.2	コマンドライン指定による実行	41
3.3	サブコマンド指定による実行.....	42
	3.3.1 会話形式による実行.....	42
	3.3.2 サブコマンドファイルによる実行.....	43
3.4	リンケージエディタの終了	45

リンケージエディタを実行するために、まずリンケージエディタを起動するコマンドラインを指定します。コマンドラインには、入力するファイル名と、必要に応じてリンケージエディタに様々な指示を与えるオプションを指定します。このコマンドラインの指定だけでも十分リンケージエディタは実行できますが、入力するファイルの数が多かったり、リンケージエディタに対して詳細な指示を与えたい場合のために、さらにサブコマンドによる実行もできます。

(1) コマンドライン指定による実行

コマンドラインでの入力ファイルとオプションの指定だけでリンケージエディタを実行する方法です。入力ファイルの数が少なく、リンケージ処理の手順も比較的単純な場合に利用します。

(2) サブコマンド指定による実行

コマンドラインだけでなく、サブコマンドでの入出力ファイルやリンケージエディタに対する実行制御の指定によってリンケージエディタを実行する方法です。入力するファイルやモジュールの数が多かったり、詳細なセクションの結合順序指定やマルチリンケージ等の機能を使う場合に利用します。サブコマンドによる実行には、会話形式とサブコマンドファイルによる2つの方法があります。

ファイル名の構成については、「付録B. ファイル名の構成」を参照してください。
表3.1に、リンケージエディタ実行上の注意事項を示します。

表 3.1 リンケージエディタ実行上の注意事項

OS	注意事項
MS-DOS	<p>本リンケージエディタをご使用になる前に、MS-DOS のシステム構築ファイル (config.sys) の内容をエディタで以下のように指定してください。</p> <div style="border: 1px solid black; padding: 5px; display: inline-block;"> <pre>FILES=20 SHELL=a:¥command.com a:¥ /p</pre> </div> <div style="display: inline-block; vertical-align: middle; margin-left: 10px;"> <p>..... (1)</p> <p>..... (2)</p> </div> <p>(1) リンケージエディタ動作時に、同時にオープンできるファイル数の指定 (2) COMMAND.COM を再ロードする際に必要なパス情報の指定</p>
UNIX	<p>コマンドラインについては、OS のシェル (コマンドインタプリタ) により、本リンケージエディタに制御が渡る前にチェックが行われます。したがって、OS がコマンドラインで指定可能としている文字についてご確認ください。</p>

3.1 コマンドラインのフォーマット

リンケージエディタのコマンドラインのフォーマットは、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>...]] (RET)
```

(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、またはリロケータブルロード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"のファイル形式を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(-)を付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。オプション名の詳細については、「4. リンケージエディタのオプション / 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サブコマンド指定による実行になります。

3.2 コマンドライン指定による実行

コマンドラインに指定された情報だけでリンケージエディタを実行する方法で、コマンドライン中に入力ファイルを指定するところの実行方法になります。出力ファイル等のリンケージエディタへの指示はオプションで指定します。入力ファイル数も少なく、セクションの結合順序等のリンケージエディタへの詳細な指示が必要でない場合は、このコマンドラインの指示だけで十分リンケージ処理が行えます。以下に、コマンドラインによる実行の例を示します。例中のオプションの詳細機能については、「4. リンケージエディタのオプション/サブコマンド」を参照してください。

(例1)

```
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、アブソリュート形式の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。

(例2)

```
lnk main,key,display,print-OUTPUT=calc-PRINT=calc-FORM=
R-DEBUG (RET)
```

リンケージエディタは"main.obj"、"key.obj"、"display.obj"および"print.obj"の4つのファイルを入力し、リロケータブル形式のロードモジュールファイル"calc.rel"を出力します。出力するロードモジュールファイルには、デバッグ情報を組み込みます。リンケージリストは"calc.map"へ出力します。

3.3 サブコマンド指定による実行

入力するファイルやモジュールが多かったり、複雑なセクションの結合をしたい場合には、コマンドラインだけでは指定しきれないことがあります。このような場合に、サブコマンド指定による実行を利用します。サブコマンド指定による実行には、サブコマンドを1つずつ標準入力から入力していく会話形式による方法と、あらかじめサブコマンド群をサブコマンドファイルとして作成しておき、この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による方法があります。

(1) 会話形式

会話形式による実行は、サブコマンドの数が比較的小さい場合に利用します。また、プログラムのデバッグ段階でリンケージエディタを使用する場合のように、中間リンケージ結果を確認したり、エラーを暫定的に回復したりする場合にも利用します。

(2) サブコマンドファイル

サブコマンドファイルによる実行は、サブコマンドの数が多い場合や、ほぼ固定した内容、手順でリンケージエディタを使用する場合に利用します。コマンドラインでSUBCOMMAND オプションを指定すると、サブコマンドファイルによる実行になります。入力するサブコマンドファイル名は、SUBCOMMAND オプションのパラメータとして指定します。

リンケージエディタでは会話形式によるサブコマンド入力でもサブコマンドファイルを実行することができます。SUBCOMMAND サブコマンドを指定し、サブコマンドファイル名をパラメータとして指定します。

3.3.1 会話形式による実行

リンケージ処理に必要なサブコマンドを標準入力から直接入力する方法で、コマンドラインで入力ファイルの指定がなく、SUBCOMMAND オプションの指定もない場合に、この実行方法になります。入力するサブコマンドの数が比較的小さい場合や、プログラムデバッグの初期段階のようにリンケージ結果を確認しながらサブコマンドを入力する場合に利用します。デバッグ援助機能を利用する場合は、会話形式による実行が適しています。以下に、会話形式による実行の例を示します。例中のサブコマンドの詳細機能については、「4. リンケージエディタのオプション / サブコマンド」を参照してください。

(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)
:FORM R (RET) ... (7)
:EXIT (RET) ... (8)

```

- (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) リロケートブル形式のロードモジュールを作成します。
- (8) ロードモジュールファイル"main.rel"を出力し、リンケージ処理を終了します。

3.3.2 サブコマンドファイルによる実行

あらかじめリンケージ処理に必要なサブコマンドをサブコマンドファイルに作成しておき、SUBCOMMAND オプション / サブコマンドのパラメータとしてこのサブコマンドファイルを指定することにより実行する方法です。この方法を利用することで、入力するサブコマンドの数が多い場合や、リンケージエディタに対する指定順序や内容がほぼ決まっているような場合に、いちいち標準入力から入力する手間を省くことができます。サブコマンドファイルは、エディタを利用して作成します。

以下に、サブコマンドファイルによる実行の例を示します。例中のサブコマンドの詳細機能は、「4. リンケージエディタのオプション / サブコマンド」を参照してください。

(例1)

```
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)
```

```
FORM A ... (7)
```

```
EXIT ... (8)
```

(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。ファイル形式は省略されているので、".rel"または".abs"のどちらかを仮定します。

(3) 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) 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) 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) アブソリュート形式のロードモジュールを作成します。出力ファイル名のファイル形式は、".abs"となります。

(8) 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。

(例2)

```
lnk (RET) ... (1)
```

```
:SUBCOMMAND pgmlnk.sub (RET) ... (2)
```

(1) リンケージエディタを実行します。パラメータを指定していないので、会話形式による実行となります。

(2) "pgmlnk.sub"からサブコマンドを入力します。

サブコマンドファイル内にEXIT サブコマンドがない場合、サブコマンド入力待ちとなります。

3.4 リンケージエディタの終了

リンケージエディタが終了するとき、エラーのレベルをリターンコードとしてシステムに返します。リターンコードを返すことにより、コマンドファイルの実行を制御することができます。

リターンコードはエラーのレベルにより以下の値となります。

システム	MS-DOS	UNIX
正常終了	0	0
ウォーニング	0	0
エラー	1	1
フェイタルエラー	1	1

4. リンケージエディタのオプション / サブコマンド

第4章 目次

4.1	オプション / サブコマンドのフォーマット	50
4.2	オプション / サブコマンドの種類	52
4.3	ファイル制御	57
4.3.1	INPUT - 入力ファイルの指定	57
4.3.2	OUTPUT - 出力ファイルの指定	59
4.3.3	LIBRARY - ライブラリファイルの指定	61
4.3.4	PRINT - リストファイルの指定	63
4.3.5	EXCLUDE - ライブラリファイルモジュールの結合抑止指定	64
4.3.6	DIRECTORY - ディレクトリ名置き換えの指定	66
4.3.7	FSYMBOL - 解決済み外部定義シンボルの出力指定	67
4.4	メモリ割り付け	69
4.4.1	START - セクションの先頭アドレス / 結合順序の指定	69
4.4.2	ENTRY - 実行開始アドレスの指定	72
4.4.3	ALIGN_SECTION - 境界調整が異なるセクションの結合指定	73
4.4.4	CHECK_SECTION - セクションのチェックの指定	74
4.4.5	AUTOPAGE - 自動ページングの指定	75
4.4.6	CPU - CPU 情報ファイルによるアドレスチェックの指定	76
4.4.7	CPUCHECK - CPU 情報ファイルによるアドレスチェック時の エラー出力指定	78
4.4.8	ROM - ROM 化支援機能の指定	79
4.5	実行制御	81
4.5.1	EXCHANGE - ユニットの強制置換	81
4.5.2	SUBCOMMAND - サブコマンドファイルの指定	83
4.5.3	FORM - 出力ロードモジュールファイル形式の指定	85
4.5.4	DEBUG - デバッグ情報出力の指定	86

4. リンケージエディタのオプション / サブコマンド

4.5.5	SDEBUG	- デバッグ情報ファイル出力の指定.....	87
4.5.6	END	- サブコマンド入力の終了指定.....	88
4.5.7	EXIT	- リンケージ処理の終了指定	89
4.5.8	ABORT	- リンケージ処理の強制終了指定.....	90
4.5.9	ECHO	- サブコマンドファイルのエコーバックの指定.....	91
4.5.10	UDF	- 未定義シンボルの表示指定.....	92
4.5.11	UDFCHECK	- 未定義シンボルが存在する場合のエラー出力指定	93
4.6	デバッグ援助.....		94
4.6.1	LIST	- 中間リンケージ情報の表示.....	94
4.6.2	RENAME	- ユニット名 / 外部定義シンボル名 / 外部参照シンボル名の変更.....	95
4.6.3	DELETE	- ユニット / 外部定義シンボルの削除.....	97
4.6.4	DEFINE	- 外部参照シンボルの強制定義.....	99

オプションおよびサブコマンドは、リンケージエディタに対してファイル名の指定やセクションの結合順序といった様々な指示を与えるためのものです。オプションおよびサブコマンドの機能は、ファイル制御、メモリ割り付け、実行制御、デバッグ援助の4種類に大別されますが、これらの機能を単独あるいは組み合わせて利用することによって、リンク時に多様なロードモジュールの編集が実現できます。

(1) ファイル制御機能

ファイル制御機能は、リンケージエディタに対して入力ファイルおよび出力ファイルを指示する機能です。入力ファイルとしてはオブジェクトモジュールファイル、リロケータブルロードモジュールファイルおよびライブラリファイルがあり、出力ファイルとしてはロードモジュールファイルとリストファイルがあり、それらを指定する場合に利用します。

(2) メモリ割り付け機能

メモリ割り付け機能は、リンケージエディタに対してセクションの結合順序や先頭アドレスを指示したり、出力するロードモジュールに与える実行開始アドレスを指示する機能です。セクションの結合順序を変更したり、特定のアドレスから実行するロードモジュールを作成する場合に利用します。

(3) 実行制御機能

実行制御機能は、リンケージエディタに対して入出力情報の形式やリンケージ処理の終了を指示する機能です。サブコマンドをサブコマンドファイルから入力したり、ロードモジュール中にデバッグ情報を含ませる場合に利用します。

(4) デバッグ援助機能

デバッグ援助機能は、リンケージエディタに対してリンケージ処理中のロードモジュールの内容表示や、外部シンボル名の変更等を指示する機能です。プログラムデバッグの段階で中間リンケージ結果を確認したり、エラーのあるロードモジュールに対して暫定的な回復措置を施す場合に利用します。

オプションとサブコマンドは同一名称でかつ同等の機能を持っていますが、指定時のフォーマットは異なります。また、オプションでしか指定できないものやサブコマンドでしか指定できないものがありますので、「4.1 オプション / サブコマンドのフォーマット」および「4.2 オプション / サブコマンドの種類」を参照してください。

なお、各オプションおよびサブコマンドの詳細な機能や指定方法については、「4.3 ファイル制御」から「4.6 デバッグ援助」までの各項を参照してください。

4.1 オプション / サブコマンドのフォーマット

(1) オプション / サブコマンドの構成

(a) 名称部

名称部には、オプション名またはサブコマンド名を指定します。各名称については、「4.2 オプション / サブコマンドの種類」を参照してください。

(b) パラメータ部

パラメータ部には、オプションまたはサブコマンドの操作対象となるファイル名、アドレス値等を指定します。オプションまたはサブコマンドの種類によって必要なものと不要なもの、指定方法が異なるものがありますので、「4.3 ファイル制御」から「4.6 デバッグ援助」の各項を参照してください。

名称部とパラメータ部の区切り記号は、オプションとサブコマンドで異なります。オプションの場合はイコール (=) で区切り、サブコマンドの場合は1つ以上のスペースまたはタブで区切ります。

オプションのフォーマット

< 名称部 > = < パラメータ部 >

サブコマンドのフォーマット

< 名称部 > < パラメータ部 >

(例)

-OUTPUT=loadf..... オプションの場合

OUTPUT loadf サブコマンドの場合

この例では、"OUTPUT"が名称部で、"loadf"がパラメータ部です。

(2) サブコマンドの継続指定

サブコマンドを1行に指定しきれないような場合(500文字/行までですが、OSに依存する場合があります)は、継続指定を利用します。継続指定は行の最後にアンパサンド(&)を指定することにより示しますが、必ず各パラメータの区切りで指定しなければいけません。パラメータの途中で指定した場合は、そのパラメータの一部分と判断します。また、アンパサンドの後ろに文字(スペース、タブは除く)を指定した場合は、エラーとなり、継続指定にはなりません。なお、継続指定に対する会話形式での入力待ちのプロンプトは"-"になります。

(例)

```

:INPUT  obj00,lib(mod0,mod1),&(RET)
-obj01,obj02  (RET)

```

→ 継続指定を示します。

```

:INPUT  obj00,lib(mod0,mod1),ob&  (RET)
:

```

→ パラメータの途中での指定のため、ob&というファイル名で処理されます。

→ 継続行とはなりません。

(3) サブコマンドのコメント指定

サブコマンドファイル中に注釈等を入れたい場合は、コメント指定を利用します。サブコマンド行中にセミコロン(;)を指定することにより、それ以降をコメントとして扱うことができます。ただし、サブコマンドの名称部またはパラメータ部とセミコロンとの間には、必ず1つ以上のスペースまたはタブが必要です。

また、サブコマンド行の先頭にセミコロンを指定することにより、その行全体をコメントとして扱うことができます。

(例)

```

; EXAMPLE OF LINKAGE SUBCOMMAND
... 行全体がコメントです。

LIBRARY  syslib  ; INDICATES LIBRARY FILE
...  "INDICATES LIBRARY FILE"がコメントです。

INPUT  object.rel;abc
...  "object.rel;abc"が1つのパラメータになります。

```

4.2 オプション / サブコマンドの種類

20 種類のオプションと 29 種類のサブコマンドがあります。表 4.1 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。

表 4.1 オプション / サブコマンド一覧表

項番	分類	オプション/サブコマンド名 (否定形)	機能	オプション	サブコマンド	節番号
1	ファイル制御	<u>I</u> NPUT	入力ファイルの指定	×		4.3.1
		<u>O</u> UTPUT* (<u>N</u> OOUTPUT)	出力ファイルの指定			4.3.2
		<u>L</u> IBRARY (<u>N</u> OLIBRARY*)	ライブラリファイルの指定			4.3.3
		<u>P</u> RINT (<u>N</u> OPRINT*)	リストファイルの指定			4.3.4
		<u>E</u> XCLUDE (<u>N</u> OEXCLUDE*)	ライブラリファイルの結合抑止 指定			4.3.5
		<u>D</u> IRECTORY	ディレクトリ名置き換えの指定	×		4.3.6
		<u>F</u> SYMBOL	解決済み外部定義シンボルの 出力指定			4.3.7
2	メモリ割り付け	<u>S</u> TART	セクションの先頭アドレス / 結合順序の指定			4.4.1
		<u>E</u> NTRY	実行開始アドレスの指定			4.4.2
		<u>A</u> LIGN_SECTION	境界調整が異なるセクションの 結合指定			4.4.3
		<u>C</u> HECK_SECTION	セクションのチェックの指定			4.4.4
		<u>A</u> UTOPAGE (<u>N</u> OAUTOPAGE*)	自動ページングの指定			4.4.5
		<u>C</u> PU	CPU 情報ファイルによる アドレスチェックの指定			4.4.6
		<u>C</u> PUCHECK	CPU 情報ファイルによるアドレス チェック時のエラー出力指定			4.4.7
		<u>R</u> OM	ROM 化支援機能の指定			4.4.8
3	実行制御	<u>E</u> XCHANGE	エントリの強制置換	×	○	4.5.1
		<u>S</u> UBCOMMAND	サブコマンドファイルの指定	○	○	4.5.2

項番	分類	オプション/サブコマンド名 (否定形)	機能	オプション	サブコマンド	節番号
		<u>FORM</u>	出力ロードモジュールファイル形式の指定	○	○	4.5.3
		<u>DEBUG</u> (<u>NODEBUG</u> *)	デバッグ情報出力の指定	○	○	4.5.4
		<u>SDEBUG</u>	デバッグ情報ファイル出力の指定	○	○	4.5.5
		<u>END</u>	サブコマンド入力の終了指定	×	○	4.5.6
		<u>EXIT</u>	リカージ処理の終了指定	×	○	4.5.7
		<u>ABORT</u>	リカージ処理の強制終了指定	×	○	4.5.8
		<u>ECHO</u> * (<u>NOECHO</u>)	サブコマンドファイルのエコーバックの指定	○	○	4.5.9
		<u>UDF</u> * (<u>NOUDF</u>)	未定義シンボルの表示指定	○	○	4.5.10
		<u>UDFCHECK</u>	未定義シンボルが存在する場合のエラー出力指定	○	○	4.5.11
4	デバッグ援助	<u>LIST</u>	中間リカージ情報の表示	×	○	4.6.1
		<u>RENAME</u>	エント名/外部定義シンボル名 /外部参照シンボル名の変更	×	○	4.6.2
		<u>DELETE</u>	エント/外部定義シンボルの削除	×	○	4.6.3
		<u>DEFINE</u>	外部参照シンボルの強制定義	○	○	4.6.4

- 【注】 1. アンダーラインの部分は最も短い短縮形を示します。
2. ○印は使用できることを、×印は使用できないことを示します。
3. アスタリスクはデフォルトで選択するオプション / サブコマンドを示します。

(1) オプション / サブコマンドの否定形

オプション / サブコマンドには否定形を指定できるものがあり、先頭に"NO"を付けて指定します。否定形にはパラメータ部は指定できません。オプション / サブコマンドの否定形には次の8種類があります。

- (a) NOOUTPUT ロードモジュールファイルの出力抑止指定
(b) NOLIBRARY ライブラリファイルの未使用指定
(c) NOPRINT リストファイルの出力抑止指定
(d) NOEXCLUDE モジュールの結合指定
(e) NOAUTOPAGE 自動ページングの抑止指定
(f) NODEBUG デバッグ情報の出力抑止指定
(g) NOECHO サブコマンドファイルのエコーバック抑止指定
(h) NOUDF 未定義シンボルの表示抑止指定

(2) オプションのデフォルト

デフォルト (オプションを指定しない場合) は次のオプションが選ばれます。

- (a) OUTPUT (パラメータ部なし)
- (b) NOLIBRARY
- (c) NOPRINT
- (d) NOEXCLUDE
- (e) NOAUTOPAGE
- (f) FORM=A
- (g) NODEBUG
- (h) ECHO
- (i) UDF

(3) 名称部の短縮指定

オプションおよびサブコマンドの名称部は、その名称であることが識別できる部分まで短縮して指定ができます。例として"DEBUG"の短縮指定を示します。

D DELETE および DEFINE と区別できないため、エラーとなります。

DE DELETE および DEFINE と区別できないため、エラーとなります。

DEB DEBUG と識別します。

DEBU DEBUG と識別します。

DEBUG DEBUG と識別します。

DEBUGS DEBUGS はないため、エラーとなります。

(4) オプションの有効範囲

コマンドライン指定による実行の場合、オプション指定の内容だけで実行します。サブコマンド指定による実行の場合、オプション指定内容の有効範囲は、最初に指定された END サブコマンド (END サブコマンドの指定がない場合は EXIT サブコマンド) までです。ただし、オプションと相反する機能を持つサブコマンドが指定された場合、エラーメッセージを表示しオプション指定は無効となり、サブコマンド指定の内容が有効となります。最初の END サブコマンド以降は、サブコマンドの指定内容が有効になります。

(例)

<code>lnk -NOOUTPUT (RET)</code>	}	NOOUTPUTオプションが有効のため、 出力ファイルを作成しません。
<code>.</code>		
<code>.</code>		
<code>.</code>		
<code>:END (RET)</code>	}	OUTPUTサブコマンドが有効のため、 出力ファイル"loadfile.abs"を作成します。
<code>.</code>		
<code>.</code>		
<code>:OUTPUT loadfile (RET)</code>		
<code>.</code>		
<code>.</code>		

4. リンケージエディタのオプション / サブコマンド

オプション / サブコマンドの説明に使用する表の内容を以下に示します。

(1) フォーマット

名 称	オプション	サブコマンド	否定形
パラメータ			

オプション / サブコマンドの名称
およびパラメータの指定形式です。
名称の下線部は最も短い短縮指定
を示します。

(2) 機 能

オプション / サブコマンドの機能
概要です。

(3) 説 明

オプション / サブコマンドの詳細
な機能や制限事項です。

(4) 指定例

オプション / サブコマンドの指定
例です。

4.3 ファイル制御

4.3.1 INPUT - 入力ファイルの指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	なし	INPUT	なし
パ ラメータ	<入力ファイル名> [(<モジュール名> [, <モジュール名> ...])] [{, } <入力ファイル名> [(<モジュール名> [, <モジュール名> ...])] ...]		

(2) 機 能

入力ファイルおよび入力モジュールを指定します。

(3) 説 明

(1) 機能説明

- ・パラメータに指定されたファイルまたは指定されたファイル中のモジュールを入力します。
- ・パラメータに指定できるファイルの種類は、オブジェクトモジュールファイル、ロードモジュールファイルおよびライブラリファイルの3種類です。
- ・モジュール名はライブラリファイルの場合にのみ指定でき、指定されたライブラリファイル中の当該モジュールだけを入力します。
- ・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次のファイル形式を仮定してファイルを入力します。

モジュール名指定なし ... "obj"

モジュール名指定あり ... "lib"

(2) 使用上の制限

- ・ロードモジュールファイルとしては、リロケータブルロードモジュールのみ指定できます。アブソリュートロードモジュールを指定した場合は、エラーとなり、そのファイルは入力しません。
- ・ライブラリファイル以外のファイルの場合にモジュール名を指定すると、エラーとなり、そのファイルは入力しません。
- ・1度のリンケージ処理で扱える入力ファイルの数は、ライブラリファイルを含めて最大 65,535 個です。65,535 個を超えて指定した場合は、エラーとなり、65,535 個目を超えたファイルは入力しません。このような場合は、マルチリンケージ機能を利用してください。
- ・ページタイプと非ページタイプの異なるモジュールを同時に入力することはできません。タイプの異なるモジュールを入力すると、エラーとなり、リンケージエディタは実行を終了します。

(4) 指定例

INPUT main

オブジェクトモジュールファイル"main.obj"を入力します。

INPUT funclib(sin,cos),tan.o

ライブラリファイル"funclib.lib"中のモジュール"sin"および"cos"とオブジェクトモジュールファイル"tan.o"を入力します。

4.3.2 OUTPUT - 出力ファイルの指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	<u>O</u> UTPUT	<u>O</u> UTPUT	<u>NO</u> OUTPUT
パ ラメータ	[<出力ファイル名>]		

(2) 機 能

ロードモジュールの出力ファイル名を指定します。

(3) 説 明

(1) 機能説明

- ・ リンケージエディタが生成したロードモジュールを、指定されたファイルに出力します。
- ・ 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ファイル形式を付けます。そのファイル形式はロードモジュールファイル形式により次のようになります。

アブソリュート形式 ... "abs"

リロケータブル形式 ... "rel"

ロードモジュールファイル形式は、FORM オプション / サブコマンドの指定内容で決まります。指定がない場合は、アブソリュート形式です。

- ・ OUTPUT オプション / サブコマンドによって出力ファイル名が指定されなかった場合は、先頭に指定された入力ファイルのファイル名に上記のファイル形式を付けたファイルが出力ファイルになります。
- ・ NOOUTPUT オプション / サブコマンドが指定された場合は、ロードモジュールファイルの出力はありません。

(2) 使用上の制限

- ・ NOOUTPUT オプション / サブコマンドにパラメータを指定することはできません。
- ・ 出力ファイル名を指定する場合は、入力ファイル名と重複しないようにご注意ください。

(4) 指定例

-OUTPUT=prgload

ロードモジュールファイル"prgload.abs"(または"prgload.rel")を出力します。

-OUTPUT

先頭に指定されたオブジェクトモジュールファイルのファイル名に"abs" (または"rel") を付けたロードモジュールファイルを出力します。

OUTPUT main.10

ロードモジュールファイル"main.10"を出力します。

4.3.3 LIBRARY - ライブラリファイルの指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	<u>LIBRARY</u>	<u>LIBRARY</u>	<u>NOLIBRARY</u>
パ ラメータ	<ライブラリファイル名> [, <ライブラリファイル名>...]		

(2) 機 能

入力ライブラリファイルを指定します。

(3) 説 明

(1) 機能説明

- ・ 入力ファイルとして指定されたファイル間でのリンケージ処理後に未解決の外部参照シンボルが残った場合に、リンケージエディタがサーチするライブラリファイルを指定します。
- ・ ユーザライブラリファイルとシステムライブラリファイルを指定した場合、リンケージエディタはユーザライブラリファイルを先にサーチします。
- ・ ライブラリ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"lib"をファイル形式として仮定してライブラリファイルを入力します。
- ・ NOLIBRARY オプション / サブコマンドの指定がある場合は、ライブラリファイル（デフォルトライブラリを含む）からの入力はありません。ただし、サブコマンド指定による実行の場合は、オプションの有効範囲に制限がありますので、「4.2(4) オプションの有効範囲」を参照してください。

(2) 使用上の制限

- ・リンケージエディタが入力できるライブラリファイルは、Hシリーズ ライブラリアンを使用して作成したライブラリファイルだけです。
- ・1度のリンケージ処理で扱える入力ファイルの数は、ライブラリファイルを含めて最大 65,535 個です。65,535 個を越えて指定した場合は、エラーとなり、65,535 個目を越えたファイルは入力しません。このような場合は、マルチリンケージ機能を利用してください。
- ・ページタイプと非ページタイプの異なるモジュールを同時に入力することはできません。タイプの異なるモジュールを入力すると、エラーとなり、リンケージエディタは実行を終了します。
- ・NOLIBRARY オプション / サブコマンドにパラメータを指定することはできません。

(4) 指定例

`-LIBRARY=syslib`

ライブラリファイルとして"syslib.lib"を指定します。

`LIBRARY system,debug`

ライブラリファイルとして"system.lib"と"debug.lib"を指定します。

4.3.4 PRINT - リストファイルの指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	<u>P</u> R I N T	<u>P</u> R I N T	<u>N</u> O P R I N T
パ ラメータ	$\left\{ \begin{array}{l} \text{<リストファイル名>} \\ \# \end{array} \right\}$		

(2) 機 能

リンケージリストを出力するリストファイルを指定します。

(3) 説 明

(1) 機能説明

- ・ 指定されたリストファイルにリンケージリストを出力します。
- ・ パラメータに"#"が指定された場合は、リストファイルを標準出力に出力します。
- ・ PRINT オプション / サブコマンドの指定がない場合、または NOPRINT オプション / サブコマンドが指定された場合は、リンケージリストは出力しません。
- ・ リスト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ファイル形式として自動的に"map"を仮定してリストファイルを出力します。
- ・ リンケージリストの内容については、「6.1 リンケージリスト」を参照してください。

(2) 使用上の制限

- ・ NOPRINT オプション / サブコマンドにパラメータを指定することはできません。

(4) 指定例

```
-PRINT=linkage
```

リンケージリストをリストファイル"linkage.map"へ出力します。

```
PRINT earth.prn
```

リンケージリストをリストファイル"earth.prn"へ出力します。

4.3.5 EXCLUDE - ライブラリファイルモジュールの結合抑止指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	EXCLUDE	EXCLUDE	NOEXCLUDE
パ ラメータ	なし		

(2) 機 能

参照のない外部参照シンボルが存在する場合、それを定義しているライブラリ内のモジュールの結合を抑止することを指定します。

(3) 説 明

(1) 機能説明

- ・参照のない外部参照シンボルがある場合、それを定義しているモジュールを結合しません。
- ・NOEXCLUDE オプション / サブコマンドを指定した場合、参照のない外部参照シンボルを定義しているモジュールを結合します。EXCLUDE オプション / サブコマンドを省略した場合も、定義モジュールの結合を行います。

(2) 使用上の制限

- ・INPUT、EXCHANGE サブコマンドにより入力ファイル指定をした後は、EXCLUDE サブコマンドは使用できません。
- ・EXCLUDE オプション / サブコマンドは、出力ロードモジュール形式がアブソリュート時だけ指定可能です。このため、マルチリンケージ機能を利用して最後のリンケージ処理でアブソリュートロードモジュールを作成する場合、デフォルトライブラリ機能を使用していると、最初のリンケージ処理でライブラリ内のモジュールを取り込んでしまいます。最後のリンケージ処理でデフォルトライブラリの取り込みを行いたい場合は、途中のリンケージ処理で NOLIBRARY サブコマンドを指定してください。

(4) 指定例

-EXCLUDE

参照のない外部参照シンボルがある場合、それを定義しているモジュールの結合を抑止します。

4.3.6 DIRECTORY - ディレクトリ名置き換えの指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	なし	<u>D</u> IRECTORY	なし
パラメータ	<シンボル名> (<ディレクトリ名>)		

(2) 機 能

ディレクトリ名をシンボル名に置き換える。

本機能により、冗長なディレクトリ名を単純なシンボル名で指定できます。

(3) 説 明

(1) ディレクトリ名の登録

DIRECTORY サブコマンドでディレクトリ名をシンボル名に登録します。

DIRECTORY <シンボル名> (<ディレクトリ名>)

(2) ディレクトリ名の参照

登録したディレクトリ名の参照は、シンボル名を \$ と / で囲みます (MS-DOS 版は \$ と ¥)。シンボル名が未登録の場合は、置き換えを行いません。

\$シンボル名 / - - - > ディレクトリ名 / に置き換えます

(3) ディレクトリ名は、16 個まで登録できます。

(4) 指定例

```
DIRECTORY     symbol(dir1/dir2)
```

```
INPUT     $symbol/file1.obj
```

ディレクトリ名 dir1/dir2 をシンボル名 symbol として登録します。

\$symbol/を dir1/dir2/に置き換え、ファイル名を dir1/dir2/file1.obj とします。

4.3.7 FSYMBOL - 解決済み外部定義シンボルの出力指定

(1) フォーマット

名 称	オプション		サブコマンド	否定形
	FSYMBOL		FSYMBOL	なし
パラメータ	オプション	<セクション名> [, <セクション名> ...]		
	サブコマンド	<セクション名> [, <セクション名> ...]		

(2) 機能

シンボルアドレス出力機能は、リンケージエディタで解決した外部定義シンボルをアセンブラ制御命令の形式でファイルに出力します。出力ファイルをアセンブル、リンクすることにより、当該シンボル定義を含むオブジェクトプログラムをリンクすることなく、外部参照シンボルのアドレスを解決することができます。

(3) 説明

(1) 機能説明

- ・本オプションは、リンケージエディタで解決した外部定義シンボルをアセンブラ制御命令の形式でファイルに出力します。出力するファイル名は、ロードモジュール名にファイル拡張子".fsy"を付加したファイルとなります。

(2) 使用上の制限事項

- ・出力するロードモジュールファイルがリロケータブル形式の場合、本オプション / サブコマンドは使用できません。
- ・外部定義シンボル名の長さが 238 文字を超えるシンボルは、ウォーニングメッセージ 126 を出力し、238 文字までが有効になります。
- ・指定したセクション名が存在しない場合、ウォーニングメッセージ 127 を表示し、処理を継続します。指定したセクションが全て存在しない場合、ファイルは出力されません。

(4) 指定例

図 4.1 は、製品 A の機能 A を機能 B に変更し、製品 B を開発する例です。本機能を用いて、共通 ROM 内シンボルのアドレスを解決することにより、共通 ROM が流用できます。

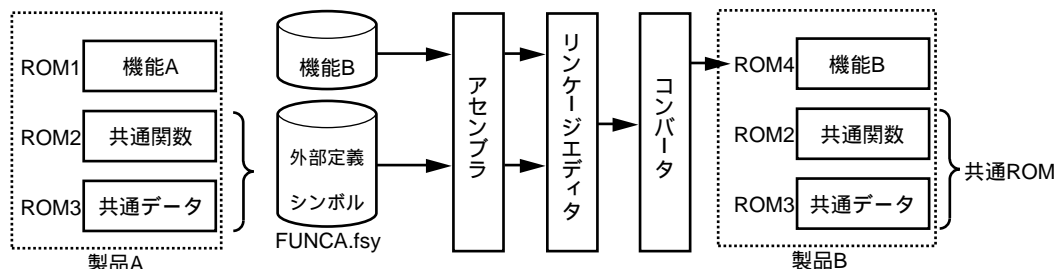


図 4.1 シンボルアドレス出力機能の使用例

【外部定義シンボルファイル出力の指定例】

```
lnk ROM1,ROM2,ROM3-output=FUNCA -fsymbol=sct2,sct3
```

sct2 と sct3 の外部定義シンボルをファイルに出力します。

【ファイル (FUNCA.fsy) の出力例】

```
;H SERIES LINKAGE EDITOR GENERATED FILE    1997.10.10
;fsymbol = sct2, sct3

;SECTION NAME = sct1
.export sym1
sym1: .equ    h'00FF0080
.export sym2
sym2: .equ    h'00FF0100
;SECTION NAME = sct2
.export sym3
sym3: .equ    h'00FF0180

.end
```

【アセンブル、再リンクの指定例】

```
asmsh ROM4
asmsh FUNCA.fsy
lnk ROM4,FUNCA
```

ROM2, ROM3 のオブジェクトファイルをリンクすることなく、ROM4 の外部参照シンボルを解決します。

【注意】

本機能を使用する場合、共通関数から機能 A 内シンボルは参照できません。

4.4 メモリ割り付け

4.4.1 START - セクションの先頭アドレス / 結合順序の指定

(1) フォーマット

名 称	オプション		サブコマンド	否定形
	START		START	なし
パ ラメ タ	オ プ シ ョ ン	MS-DOS版	<セクション名> [, <セクション名>...] [[(<パ° -ジ° アド° レス>:] <先頭アド° レス>)] [, <セクション名> [, <セクション名>...] [[(<パ° -ジ° アド° レス>:] <先頭アド° レス>)] ...]	
		UNIX版	<セクション名> [, <セクション名>...] [(<パ° -ジ° アド° レス>:] <先頭アド° レス>] [, <セクション名> [, <セクション名>...] [(<パ° -ジ° アド° レス>:] <先頭アド° レス>] ...]	
	サ ブ コ マ ン ド	<セクション名>[, <セクション名>...][(<パ° -ジ° アド° レス>:] <先頭アド° レス>)] [, <セクション名>[, <セクション名>...][(<パ° -ジ° アド° レス>:] <先頭アド° レス>)]...		

複数セクションの同一アドレスへの割り付け指定

オ プ シ ョ ン	MS-DOS 版	-START=<セクション名> [, <セクション名>...] [:<セクション名> [, <セクション名>...]] (<先頭アド° レス>)
	UNIX 版	-START=<セクション名> [, <セクション名>...] [:<セクション名> [, <セクション名>...]] /<先頭アド° レス>
サ ブ コ マ ン ド	START <セクション名> [, <セクション名>...] [:<セクション名> [, <セクション名>...]] (<先頭アド° レス>)	

(2) 機 能

セクションの結合順序およびその先頭アドレスを指定します。

(3) 説 明

(1) 機能説明

- ・ 指定されたアドレスからセクションを指定された順序に割り付けます。
- ・ 先頭アドレスの指定を省略した場合は、セクションの結合順序だけを指定することになり、セクションを0番地から割り付けます。
- ・ ページアドレスはページタイプのモジュールの場合にのみ指定できます。ページアドレスの指定を省略した場合は、ページアドレスとして0(ゼロ)を仮定します。

- ・ ページアドレスおよび先頭アドレスは 16 進数で指定します。
- ・ パラメータ中に指定のないセクションを入力した場合、指定された先頭アドレスのうち最も高位のアドレスのセクションの並びの最後にそのセクションを割り付けます。
- ・ START オプション / サブコマンドが指定されていない場合は、0 番地からセクションを出現順に割り付けます。
- ・ START オプション / サブコマンドは複数回指定することができます。

(2) 使用上の制限

- ・ 出力するロードモジュールがリロケータブル形式の場合、START オプション / サブコマンドは使用できません。
- ・ 非ページタイプのモジュールの場合にページアドレスを指定すると、エラーとなり、リンケージエディタは終了します。
- ・ 16 進数は必ず先頭を 0~9 の数字で指定します。

(例) 0ABCD ... 正しい指定

ABCD ... 誤った指定

- ・ 指定できるページアドレスの範囲は、0~0FF(16 進数)です。
- ・ 指定できる先頭アドレスの範囲は、H シリーズの品種により異なります。

H8/500 シリーズ : 0~0FFFF (16 進数)

H8/300 シリーズ : 0~0FFFF (16 進数)

(300, 300L, 300HN)

H8/300 シリーズ : 0~0FFFFFFF (16 進数)

(300HA)

H8S シリーズ : 0~0FFFF (16 進数)

(2600N, 2000N)

H8S シリーズ : 0~0FFFFFFFF (16 進数)

(2600A, 2000A)

SuperH シリーズ : 0~0FFFFFFFF (16 進数)

(4) 指定例

-START=CODE, DATA, BSS, STACK

セクションを"CODE"、"DATA"、"BSS"、"STACK"の順序で結合し、0 番地から割り付けます。

-START=CONTROL, BANK0, BANK1 (0F00) (MS-DOS 版)

-START=CONTROL, BANK0, BANK1 / 0F00 (UNIX 版)

セクションを"CONTROL"、"BANK0"、"BANK1"の順序で結合し、F00(16 進数)番地から割り付けます。

START CONTROL, BANK0, BANK1 (0:0F00)

セクションを"CONTROL"、"BANK0"、"BANK1"の順序で結合し、0 ページの F00 (16 進数) 番地から割り付けます。

START RAM0, RAM1 (8000) , ROM1, ROM2 (1000) , ROM0

セクションを"RAM0"、"RAM1"の順序で結合し、8000(16 進数)番地から割り付けます。

セクション"ROM1"、"ROM2"の順序で結合し、1000(16 進数)番地から割り付けます。

セクション"ROM0"を 0 番地から割り付けます。

4.4.2 ENTRY - 実行開始アドレスの指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	<u>ENTRY</u>	<u>ENTRY</u>	なし
パ ラメータ	<外部定義シンボル名>		

(2) 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します。

(3) 説 明

(1) 機能説明

- ・ 出力するロードモジュールの実行開始アドレスを、指定された外部定義シンボルの値（アドレス）に設定します。
- ・ ENTRY オプション / サブコマンドの指定がなく、出力ロードモジュール形式がアブソリュートの場合は、出力するロードモジュール中のコードセクションのうち、最も先頭に現れたセクションの先頭アドレスが実行開始アドレスになります。

(2) 使用上の制限

- ・ ENTRY オプション / サブコマンドを 2 度以上指定した場合は、後から指定したアドレスが有効となります。

(4) 指定例

```
-ENTRY=PRG_ENT
```

外部定義シンボル"PRG_ENT"のアドレスを実行開始アドレスとして設定します。

```
ENTRY MAIN
```

外部定義シンボル"MAIN"のアドレスを実行開始アドレスとして設定します。

4.4.3 ALIGN_SECTION - 境界調整が異なるセクションの結合指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	ALIGN_SECTION	ALIGN_SECTION	なし
パ ラメータ	なし		

(2) 機 能

同一名で境界調整数（アセンブラの .SECTION 制御命令の ALIGN オペランドで指定する）が異なるセクションでも、同一のセクションとしてアドレスを割り付けます。

(3) 説 明

機能説明

- ・アセンブラの .SECTION 制御命令の ALIGN オペランドで異なる境界調整数を指定して同一名のセクションを生成することができます。この場合、同一名のセクションであっても境界調整数が異なるため、同一のセクションとしてアドレスを割り付けることができません。本オプションを指定することにより、同一のセクションとして割り付けることができます。

(4) 指定例

-ALIGN_SECTION

境界調整数が異なる場合でも、同一セクションとしてアドレスを割り付けます。

4.4.4 CHECK_SECTION - セクションのチェックの指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	CHECK_SECTION	CHECK_SECTION	なし
パ ラメータ	なし		

(2) 機 能

START オプション / サブコマンドで指定しなかったセクションが入力ファイル中に存在する時、ウォーニングを出力し、処理を継続します。

(3) 説 明

- (1) 入力ファイル中に START オプション / サブコマンドでセクションの先頭アドレスを指定しなかったセクションの有無をチェックして、ウォーニングメッセージ 120 を出力します。
- (2) ウォーニングメッセージの出力後も処理を継続します。

(4) 指定例

-CHECK_SECTION

セクションの先頭アドレスが割り付けられていないセクションの有無をチェックして、ウォーニングを出力します。

4.4.5 AUTOPAGE - 自動ページングの指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	<u>A</u> UTOPAGE	<u>A</u> UTOPAGE	<u>N</u> OAUTOPAGE
パ ラメータ	なし		

(2) 機 能

ページタイプのモジュールに対して、自動ページングによるアドレス割り付けを指定します。

(3) 説 明

(1) 機能説明

- ・ ページタイプのモジュールの結合処理に対して、自動ページングによるアドレス割り付けを指定します。
- ・ AUTOPAGE オプション / サブコマンドの指定がない場合、または NOAUTOPAGE オプション / サブコマンドの指定がある場合は、自動ページングによるアドレス割り付けは行いません。

(2) 使用上の制限

- ・ 非ページタイプのモジュールの結合に対して、AUTOPAGE オプション / サブコマンドは指定できません。指定した場合、エラーとなり、リンケージ処理は終了します。
- ・ ページタイプのモジュールの結合に対して、NOAUTOPAGE オプション / サブコマンドを指定した場合、セクションがページの境界にまたがる可能性がありますので注意してください。セクションがページの境界にまたがった場合、リンケージエディタはウォーニングを出力します。

(4) 指定例

AUTOPAGE

自動ページングによるアドレス割り付けをします。

-NOAUTOPAGE

ページに関係なくアドレス割り付けをします。

4.4.6 CPU - CPU 情報ファイルによるアドレスチェックの指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	<u>C</u> PU	<u>C</u> PU	なし
パ ラメータ	<CPU情報ファイル名>		

(2) 機 能

CPU 情報ファイルによるアドレスチェックの実行を指定します。

(3) 説 明

(1) 機能説明

- ・ CPU 情報に基づき、各セクションに割り付けられたアドレスの有効性をチェックします。次の場合、セクションのアドレス割り付けが不適当と見なし、ウォーニングメッセージを出力します。ただし、ロードモジュールファイルにはそのままのアドレスで出力します。

(a) メモリが割り付けられていない領域にセクションを割り付けた場合

(b) 1 つのセクションを異なるメモリ種別および属性を持つ複数のメモリ領域にわたって割り付けた場合

- ・ CPU 情報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ファイル形式として自動的に"cpu"を仮定して、CPU 情報ファイルを入力します。

(2) 使用上の制限

- ・ 次のような場合、リンケージエディタはウォーニングメッセージを出力し、CPU オプション / サブコマンドの指定を無効とします。

(a) FORM オプション / サブコマンドで、ロードモジュールの出力形式としてリロケータブル形式を指定した場合

(b) CPU 情報ファイルの情報形式が正しくない場合

(c) H8S,H8/300 シリーズまたは SH シリーズ以外のオブジェクトモジュールのリンケージ処理に CPU 情報ファイルを指定した場合

- ・ CPU オプション / サブコマンドを複数回指定した場合は、ウォーニングメッセージを出力し、最後に指定したファイルを有効とします。

(4) 指定例

`-CPU=cinf`

CPU 情報ファイル"cinfcpu"を入力します。

`CPU c300.inf`

CPU 情報ファイル"c300.inf"を入力します。

4.4.7 CPUCHECK - CPU 情報ファイルによるアドレスチェック時のエラー出力指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	<u>CPUCHECK</u>	<u>CPUCHECK</u>	なし
パ ラメータ	なし		

(2) 機 能

CPU オプション / サブコマンドで CPU 情報ファイルによるアドレスチェックをした際、出力されるウォーニングメッセージをエラーメッセージに変更します。

(3) 説 明

(1) 機能説明

- ・ CPU 情報ファイルで指定したメモリレイアウトと矛盾したメモリ割り付けを行った場合、エラー329 を出力して異常終了します。エラーとなる条件は、CPU オプション / サブコマンドのみ指定した場合のウォーニングとなる条件と同一です。

(2) 使用上の制限

- ・ CPU オプション / サブコマンドの指定がない場合、CPUCHECK オプション / サブコマンドを指定しても意味を持ちません。

(4) 指定例

-CPUCHECK

CPU オプション / サブコマンドのウォーニングケースをエラーケースとし、リンケージエディタを異常終了するよう指定します。

4.4.8 ROM - ROM 化支援機能の指定

(1) フォーマット

名 称	オプション		サブコマンド	否定形
	ROM		ROM	なし
パ ラメータ	MS-DOS版	(<セクション1>,<セクション2>) [, (<セクション1>,<セクション2>) ...]		
	UNIX版	<セクション1>/<セクション2> [<セクション1>/<セクション2>...]		
	<セクション1>: ROM用初期化データ領域 (転送元) のセクション名を指定します。 <セクション2>: RAM用初期化データ領域 (転送先) のセクション名を指定します。			

(2) 機 能

初期化データ領域を ROM に書き込んだ場合でも値を更新できるように、RAM 用の領域を確保することを指定します。

(3) 説 明

(1) 機能説明

- ・セクション 1 で指定したセクションサイズと同じ大きさのセクションをセクション 2 として出力ロードモジュールに確保します。セクション 2 のセクション属性は、セクション 1 の属性を引き継ぎます。
- ・セクション 1 内で宣言したシンボルを参照する場合、セクション 2 内のアドレスになるようにリロケーションします。セクション 1 は、リロケータブルセクションを指定します。
- ・セクション 1、セクション 2 は 64 組まで指定できます。
- ・ROM 化支援機能の詳細については、「2.7 ROM 化支援機能」を参照してください。

(2) 使用上の制限

- ・出力するロードモジュールがリロケータブル形式の場合、ROM オプション / サブコマンドは使用できません。
- ・同一名セクションが存在し、それをセクション 1 に指定した場合、最初に入力されたセクションを選択します。
- ・セクション 1 が存在しない場合はエラーとなります。
- ・セクション 1 としてダミーセクションを指定することはできません。

・ セクション 2 に存在するセクションを指定する場合は、下記の制限があります。

- (a) 各ユニット内のセクション 2 のサイズは 0
- (b) リロケートブルセクション
- (c) セクション 1 とセクション 2 の属性は同一

(4) 指定例

-ROM= (D, RAM_SCT) (MS-DOS 版)

-ROM=D/RAM_SCT (UNIX 版)

セクション名 D と同じ大きさのセクション RAM_SCT を出力ロードモジュールに確保します。またセクション D に割り付けたシンボルを参照している場合、RAM_SCT 上のアドレスとなるようにリロケーションします。

4.5 実行制御

4.5.1 EXCHANGE - ユニットの強制置換

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	なし	EXCHANGE	なし
パラメータ	<入力ファイル名> [(<ユニット名> [, <ユニット名> ...])]		

(2) 機 能

入力ファイル中のユニットと処理対象のロードモジュール中の同名ユニットとの置換を指定します。

(3) 説 明

(1) 機能説明

- ・ 指定した入力ファイル中のユニットとリンケージ処理中のロードモジュール中にある同名のユニットを置換します。
- ・ 入力ファイルとしては、オブジェクトモジュールファイルとロードモジュールファイルが指定できます。
- ・ 入力ファイルにロードモジュールファイルを指定し、ユニット名の指定がない場合は、そのロードモジュールファイル中の全ユニットが置換の対象となります。
- ・ 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"をファイル形式として仮定してファイルを入力します。
- ・ ユニットの置換は、全入力ファイルの取り込み後に行います。また、複数の EXCHANGE サブコマンドの指定がある場合は、その指定された順序にしたがってユニットの置換を行います。

(2) 使用上の制限

- ・ ロードモジュールファイルとしては、リロケータブルロードモジュールのみ指定できます。アブソリュートロードモジュールを指定した場合は、エラーとなり、そのファイルを入力しません。
- ・ 入力ファイルとして、ライブラリファイルを指定することはできません。指定しても、エラーとなり、そのファイルを入力しません。

(4) 指定例

EXCHANGE datain

オブジェクトモジュールファイル"datain.obj"中のユニットに同名のユニットを置換します。

EXCHANGE function.rel(tan,atan)

ロードモジュールファイル"function.rel"中のユニット"tan"および"atan"に同名のユニットを置換します。

4.5.2 SUBCOMMAND - サブコマンドファイルの指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	<u>SUBCOMMAND</u>	<u>SUBCOMMAND</u>	なし
パ ラメータ	<サブコマンドファイル名>		

(2) 機 能

入力するサブコマンドファイルを指定します。

(3) 説 明

(1) 機能説明

- ・指定されたサブコマンドファイルからサブコマンドを入力します。
- ・コマンドラインで SUBCOMMAND オプションの指定がなく、入力ファイルの指定もない場合は、会話形式のサブコマンド指定によってリンケージエディタを実行します。
- ・コマンドラインで SUBCOMMAND オプションの指定がなく、入力ファイルの指定がある場合は、コマンドラインの指定によってリンケージエディタを実行します。

(2) 使用上の制限

- ・コマンドラインでサブコマンドファイルといっしょに入力ファイルや他のオプションを指定したときは、サブコマンドファイルの実行は、指定位置に関係なく、オプションの最後となります。例えば

```
lnk in1,in2 -SUB=linkage.sub -FORM=R
```

(1) (2) (3)

とあると、(3)、(1)、(2)の順に解釈、実行されます。linkage.sub 内に FORM=A が指定されていると、FORM=A が有効（後に解釈されたものが有効となる）となります。

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

(4) 指定例

`-SUBCOMMAND=linkage.sub`

サブコマンドファイル"linkage.sub"を入力し、その内容にしたがってリンケージエディタを実行します。

4.5.3 FORM - 出力ロードモジュールファイル形式の指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	FORM	FORM	なし
パ ラメータ	{ A R }		

(2) 機 能

出力ロードモジュールファイル形式（アブソリュート形式またはリロケータブル形式）を指定します。

(3) 説 明

(1) 機能説明

- ・パラメータに"A"を指定した場合は、アブソリュート形式のロードモジュールファイルを出力します。
- ・パラメータに"R"を指定した場合は、リロケータブル形式のロードモジュールファイルを出力します。
- ・FORM オプション / サブコマンドの指定がない場合は、アブソリュート形式のロードモジュールファイルを出力します。

(2) 使用上の制限

- ・ROM または START オプション / サブコマンド指定があるときパラメータ"R"の指定はできません。

(4) 指定例

-FORM=R

リロケータブル形式のロードモジュールファイルを出力します。

FORM A

アブソリュート形式のロードモジュールファイルを出力します。

4.5.4 DEBUG - デバッグ情報出力の指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	<u>DEBUG</u>	<u>DEBUG</u>	<u>NODEBUG</u>
パ ラメータ	なし		

(2) 機 能

出力ロードモジュールファイルにデバッグ情報を組み込むことを指定します。

(3) 説 明

(1) 機能説明

- ・デバッガでシンボリックデバッグを行うために必要なデバッグ情報を、出力ロードモジュールファイルに組み込みます。
- ・DEBUG オプション / サブコマンドの指定がない場合、または NODEBUG オプション / サブコマンドの指定がある場合は、デバッグ情報の組み込みは行いません。

(2) 使用上の制限

- ・NOOUTPUT オプション / サブコマンドの指定がある場合は、DEBUG オプション / サブコマンドを指定しても意味を持ちません。

(4) 指定例

DEBUG

出力ロードモジュールファイルにデバッグ情報を組み込みます。

-NODEBUG

出力ロードモジュールファイルにデバッグ情報を組み込みません。

4.5.5 SDEBUG - デバッグ情報ファイル出力の指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	<u>S</u> DEBUG	<u>S</u> DEBUG	なし
パ ラメータ	なし		

(2) 機 能

ロードモジュールのデバッグ情報を別ファイルに分けて出力します。

一部のデバッガは、オブジェクトとデバッグ情報を分けて入力する仕様となっています。
このようなデバッガを使用する場合に限り、本オプション / サブコマンドを指定します。

(3) 説 明

(1) 機能説明

- ・ロードモジュールのデバッグ情報を別ファイルに分けて出力します。
オブジェクト用ファイル・・・ファイル拡張子".abs"
デバッグ用ファイル・・・・・・・ファイル拡張子".dbg"
- ・デバッグ情報を別ファイルにすることにより、デバッグ時にロードモジュールのダウンロード時間を短縮することができます。

(2) 使用上の制限

- ・出力するロードモジュールがリロケータブル形式の場合、SDEBUG オプション / サブコマンドは使用できません。
- ・NOOUTPUT オプション / サブコマンドの指定がある場合は、SDEBUG オプション / サブコマンドを指定しても意味を持ちません。

(4) 指定例

-SDEBUG

デバッグ用ファイルとオブジェクト用ファイルを出力します。

4.5.6 END - サブコマンド入力の終了指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	なし	<u>END</u>	なし
パ ラメータ	なし		

(2) 機 能

サブコマンドの入力をいったん終了し、リンケージ処理を開始することを指定します（サブコマンド入力に戻ります）。

(3) 説 明

(1) 機能説明

- ・サブコマンドの入力をいったん終了し、リンケージ処理を開始します。リンケージ処理終了後は、再びサブコマンドの入力に戻ります。このとき、リンケージエディタは初期状態に戻ります。
- ・END サブコマンドは、1 度のリンケージエディタの起動で複数のリンケージ処理を行うマルチリンケージ機能を利用する場合に、1 つのリンケージ処理の終了を示すものです。
- ・マルチリンケージ機能を利用しない場合およびマルチリンケージ機能での最後のリンケージ処理に対しては、END サブコマンドではなく、EXIT サブコマンドを指定してください。

(2) 使用上の制限

- ・1 つのリンケージ処理で入力ファイルの指定なしに END サブコマンドを指定するとエラーとなります。

(4) 指定例

END

サブコマンドの入力をいったん終了し、リンケージ処理を開始します。

4.5.7 EXIT - リンケージ処理の終了指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	なし	<u>EXIT</u>	なし
パ ラメータ	なし		

(2) 機 能

サブコマンド入力を終了し、リンケージ処理を開始することを指定します (サブコマンド入力には戻りません)。

(3) 説 明

機能説明

- ・サブコマンドの入力を終了し、リンケージ処理を開始します。リンケージ処理終了後は、リンケージエディタの実行を終了します。
- ・サブコマンドファイルの実行の場合で、EXIT サブコマンドの指定がない場合は、サブコマンド入力待ちとなります。
- ・1 つのリンケージ処理で入力ファイルの指定なしに EXIT サブコマンドを指定するとエラーとなります。

(4) 指定例

EXIT

サブコマンドの入力を終了し、リンケージ処理を開始します。

4.5.8 ABORT - リンケージ処理の強制終了指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	なし	<u>A</u> BORT	なし
パ ラメータ	なし		

(2) 機 能

リンケージ処理を強制終了させることを指定します。

(3) 説 明

機能説明

- ・ リンケージエディタの実行を強制的に終了します。
- ・ サブコマンドの入力ミス等のため、リンケージエディタの実行を終了したい場合に使用します。

(4) 指定例

ABORT

リンケージエディタの実行を強制的に終了します。

4.5.9 ECHO - サブコマンドファイルのエコーバックの指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	<u>E</u> CHO	<u>E</u> CHO	<u>NOE</u> CHO
パ ラメータ	なし		

(2) 機 能

サブコマンドファイル実行時、サブコマンドのエコーバックを抑止するかしないかを指定します。

(3) 説 明

機能説明

- ・サブコマンドファイル実行時、サブコマンドをコンソール表示します。
ECHO オプション / サブコマンドを指定しない場合もサブコマンドを表示します。
- ・サブコマンドファイル実行時に NOECHO オプション / サブコマンドを指定した場合は、サブコマンドをコンソールに表示しません。

(4) 指定例

-ECHO

サブコマンドファイル入力時、実行するサブコマンドをコンソール表示します。

4.5.10 UDF - 未定義シンボルの表示指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	<u>UDF</u>	<u>UDF</u>	<u>NOUDF</u>
パ ラメータ	なし		

(2) 機 能

未定義シンボルがある場合ウォーニングメッセージを表示するかしないかを指定します。

(3) 説 明

(1) 機能説明

- ・リロケータブルロードモジュール作成時、未定義シンボルがある場合にウォーニングメッセージ 105 を表示します。UDF オプション / サブコマンド指定がない場合も未定義シンボルがある場合は表示します。

リロケータブルロードモジュール作成時に NOUDF オプション / サブコマンドを指定した場合は、未定義シンボルがあっても、ウォーニングメッセージを表示しません。

(2) 使用上の制限

- ・アブソリュートロードモジュール作成時は、NOUDF オプション / サブコマンドは無視します。

(4) 指定例

-FORM=R-NOUDF

リロケータブルロードモジュール作成時、未定義シンボルがあってもウォーニングメッセージを表示しません。

4.5.11 UDFCHECK - 未定義シンボルが存在する場合のエラー出力指定

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	UDFCHECK	UDFCHECK	なし
パ ラメータ	なし		

(2) 機 能

未定義シンボルがある場合エラーメッセージを表示し、アブソリュートロードモジュールの生成を抑止します。

(3) 説 明

(1) 機能説明

- ・未定義の外部参照シンボルが存在する場合、エラーメッセージ 221 を出力してアブソリュートロードモジュールを生成しません (UDFCHECK オプション / サブコマンドの指定がない場合は、ウォーニングメッセージ 105 を表示し、アブソリュートロードモジュールは生成されます)。

(2) 使用上の制限

- ・リロケータブルロードモジュール作成時は、UDFCHECK オプション / サブコマンドは無視します。

(4) 指定例

-UDFCHECK

未定義シンボルがあればエラーメッセージを表示し、アブソリュートロードモジュールの生成を抑止します。

4.6 デバッグ援助

4.6.1 LIST - 中間リンケージ情報の表示

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	なし	<u>LIST</u>	なし
パ ラメータ	$\left\{ \begin{array}{c} M \\ U \\ X \end{array} \right\}$		

(2) 機 能

入力ファイルのリンケージ情報を表示します。

(3) 説 明

(1) 機能説明

- ・現在入力中のファイルを基にリンケージ情報を標準出力へ表示します。
- ・表示内容は、パラメータで指定します。パラメータの内容は次のとおりです。
 - M ... リンケージマップを表示します
 - U ... 未解決の外部参照シンボルを表示します
 - X ... 外部定義シンボルを表示します

(2) 使用上の制限

入力ファイルを基にしたのリンケージ情報表示を行うため、表示内容には以下の制限があります。

- ・パラメータ M が指定された場合
 - リロケータブルセクションの開始アドレスは常に 0 です。
- ・パラメータ U が指定された場合
 - LIST サブコマンド直前までの INPUT サブコマンドで指定した入力ファイル中に、対応する外部定義シンボルが存在しない外部参照シンボルを表示します。

(4) 指定例

LIST M

リンケージ処理中のロードモジュールのリンケージマップを表示します。

LIST U

リンケージ処理中のロードモジュールの未解決外部参照シンボルを表示します。

4.6.2 RENAME - ユニット名 / 外部定義シンボル名 / 外部参照シンボル名の変更

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	なし	RENAME	なし
パラメータ	$\left\{ \begin{array}{l} \text{UN}=\langle \text{ユニット名1} \rangle (\langle \text{ユニット名2} \rangle) \\ \text{ER}=\langle \text{ユニット名} \rangle. \langle \text{外部参照シンボル名1} \rangle (\langle \text{外部参照シンボル名2} \rangle) \\ \text{ED}=\langle \text{ユニット名} \rangle. \langle \text{外部定義シンボル名1} \rangle (\langle \text{外部定義シンボル名2} \rangle) \end{array} \right\}$ $\left[\begin{array}{l} \text{UN}=\langle \text{ユニット名1} \rangle (\langle \text{ユニット名2} \rangle) \\ \text{ER}=\langle \text{ユニット名} \rangle. \langle \text{外部参照シンボル名1} \rangle (\langle \text{外部参照シンボル名2} \rangle) \\ \text{ED}=\langle \text{ユニット名} \rangle. \langle \text{外部定義シンボル名1} \rangle (\langle \text{外部定義シンボル名2} \rangle) \end{array} \right] \dots]$		

(2) 機 能

入力ファイル中のユニット名、外部参照シンボル名および外部定義シンボル名の変更を指定します。

(3) 説 明

(1) 機能説明

- ・入力ファイル中の指定されたユニット名、外部参照シンボル名および外部定義シンボル名をそれぞれカッコ内に指定された名前に変更します。
- ・ユニットの場合は、"UN="に続いて指定されたユニット名をカッコ内に指定された名前に変更します。
- ・外部参照シンボルの場合は、"ER="に続いて指定された外部参照シンボル名をカッコ内に指定された名前に変更します。外部参照シンボル名は、その外部参照シンボルを含むユニット名に続いてピリオド(.)で区切って指定します。
- ・外部定義シンボルの場合は、"ED="に続いて指定された外部定義シンボル名をカッコ内に指定された名前に変更します。外部定義シンボル名は、その外部定義シンボルを含むユニット名に続いてピリオド(.)で区切って指定します。

(2) 使用上の制限

- ・ RENAME サブコマンドが作用するのは、本コマンド以降に指定した最初の INPUT サブコマンドの入力ファイルのみです。
- ・ RENAME サブコマンドの直後に指定できるサブコマンドは、次の5種類だけです。
 - (a) INPUT サブコマンド
 - (b) EXCHANGE サブコマンド
 - (c) RENAME サブコマンド
 - (d) DELETE サブコマンド
 - (e) ABORT サブコマンド

複数の RENAME サブコマンドまたは DELETE サブコマンドと一緒に指定された場合は、その指定順序にしたがって作用します。

(4) 指定例

```
RENAME UN=datalist(datalst1)
```

ユニット"dalist"を"datalst1"に変更します。

```
RENAME ED=cntl.TRUNK(P_TRUNK),ER=cntl1.REC_DATA(RECV_DATA)
```

ユニット"cntl"中の外部定義シンボル"TRUNK"を"P_TRUNK"に変更します。また、ユニット"cntl1"中の外部参照シンボル"REC_DATA"も"RECV_DATA"に変更します。

4.6.3 DELETE - ユニット / 外部定義シンボルの削除

(1) フォーマット

名 称	オプション	サブコマンド	否定形
	なし	<u>DELETE</u>	なし
パラメータ	$\left\{ \begin{array}{l} \text{UN}=\langle \text{ユニット名} \rangle \\ \text{ED}=\langle \text{ユニット名} \rangle. \langle \text{外部定義シンボル名} \rangle \end{array} \right\}$ $[, \left\{ \begin{array}{l} \text{UN}=\langle \text{ユニット名} \rangle \\ \text{ED}=\langle \text{ユニット名} \rangle. \langle \text{外部定義シンボル名} \rangle \end{array} \right\} \dots]$		

(2) 機 能

入力ファイル中のユニットおよび外部定義シンボルの削除を指定します。

(3) 説 明

(1) 機能説明

- ・入力ファイル中の指定されたユニットおよび外部定義シンボルを削除します。
- ・ユニットの場合は、"UN="に続いて指定されたユニットを削除します。
- ・外部定義シンボルの場合は、"ED="に続いて指定された外部定義シンボルを削除します。外部定義シンボル名は、外部定義シンボルを含むユニット名に続いてピリオド(.)で区切って指定します。

(2) 使用上の制限

- ・DELETE サブコマンドが作用するのは、本コマンド以降に指定した最初の INPUT サブコマンドの入力ファイルのみです。
- ・DELETE サブコマンドの直後に指定できるサブコマンドは、次の5種類だけです。
 - (a) INPUT サブコマンド
 - (b) EXCHANGE サブコマンド
 - (c) DELETE サブコマンド
 - (d) RENAME サブコマンド
 - (e) ABORT サブコマンド
- ・RENAME サブコマンドと一緒に指定された場合は、その指定順序にしたがって変更または削除を行います。

(4) 指定例

```
DELETE UN=snap_unit
```

ユニット"snap_unit"を削除します。

4. リンケージエディタのオプション / サブコマンド

DELETE UN=dummy,ED=main.DUMMY_ENTER

ユニット"dummy"を削除します。また、ユニット"main"中の外部定義シンボル
"DUMMY_ENTER"も削除します。

4.6.4 DEFINE - 外部参照シンボルの強制定義

(1) フォーマット

名 称	オプション		サブコマンド	否定形
	DEFINE		DEFINE	なし
パ ラメータ	オ プ シ ョ ン	MS-DOS版	<外部参照シボ [*] ル名> <数値> ({ [<ペー [*] ジ アド [*] レス>:] <アド [*] レス> }) <外部定義シボ [*] ル名> [,<外部参照シボ [*] ル名> <数値> ({ [<ペー [*] ジ アド [*] レス>:] <アド [*] レス> }) ...] <外部定義シボ [*] ル名>	
		UNIX版	<外部参照シボ [*] ル名> / { <数値> [<ペー [*] ジ アド [*] レス>:] <アド [*] レス> <外部定義シボ [*] ル名> } [,<外部参照シボ [*] ル名> / { <数値> [<ペー [*] ジ アド [*] レス>:] <アド [*] レス> <外部定義シボ [*] ル名> } ...]	
	サ ブ コ マ ン ド	<外部参照シボ [*] ル名> ({ <数値> [<ペー [*] ジ アド [*] レス>:] <アド [*] レス> <外部定義シボ [*] ル名> }) [,<外部参照シボ [*] ル名> ({ <数値> [<ペー [*] ジ アド [*] レス>:] <アド [*] レス> <外部定義シボ [*] ル名> }) ...]		

(2) 機 能

外部参照シンボルの強制定義を指定します。

(3) 説 明

(1) 機能説明

- ・ 外部参照シンボルを、指定された値、アドレスまたは外部定義シンボルの値で強制的に定義します。
- ・ ページアドレスはページタイプのモジュールの場合にのみ指定できます。ページアドレスの指定を省略した場合は、ページアドレスとして0(ゼロ)を仮定します。
- ・ 数値、ページアドレスおよびアドレスは16進数で指定します。

(2) 使用上の制限

- ・ 定義する値として外部定義シンボルを指定する場合は、定義済みのシンボルを指定してください。
- ・ 非ページタイプのもジュールの場合にページアドレスを指定すると、エラーとなり、リンケージエディタは終了します。
- ・ 16進数は必ず先頭を数字で指定してください。
- ・ 指定できるページアドレスの範囲は、0~0FF (16進数) です。
- ・ 指定できる数値およびアドレスの範囲は、Hシリーズの品種により異なります。

H8/500 シリーズ : 0~0FFFF (16進数)

H8/300 シリーズ : 0~0FFFF (16進数)

H8/300 シリーズ : 0~0FFFFFFF (16進数)

(H8/300HA)

H8S シリーズ : 0~0FFFF (16進数)

H8S シリーズ : 0~0FFFFFFFF (16進数)

(2600A, 2000A)

SH シリーズ : 0~0FFFFFFFF (16進数)

- ・ DEFINE サブコマンドで強制定義した値は、リロケータブルロードモジュールには反映されません。
- ・ EXCLUDE オプション / サブコマンド指定がある場合、参照のない外部参照シンボルを DEFINE サブコマンドで指定しても無視します。

(4) 指定例

-DEFINE=PORT10(0E8) (MS-DOS 版)

-DEFINE=PORT10/0E8 (UNIX 版)

未定義外部参照シンボル"PORT10"を E8(16進数)という値を持つシンボルとして定義します。

DEFINE MAIN_RTN(PRG_EXIT)

未定義外部参照シンボル"MAIN_RTN"を外部定義シンボル"PRG_EXIT"と同じ値を持つシンボルとして定義します。

5. リンケージエディタの入力

第5章 目次

5.1	オブジェクトモジュールファイル	103
5.2	リロケータブルロードモジュールファイル	104
5.3	ライブラリファイル	105
5.4	デフォルトライブラリファイル	106

5.1 オブジェクトモジュールファイル

リンケージエディタは、HシリーズのCコンパイラおよびアセンブラの出力したオブジェクトモジュールファイルを入力することができます。

5.2 リロケータブルロードモジュールファイル

リンケージエディタが出力したリロケータブルロードモジュールファイルを再入力することができます。アブソリュートロードモジュールファイルは再入力できません。

5.3 ライブラリファイル

H シリーズ ライブラリアンで作成したライブラリファイルを入力することができます。ライブラリファイル内のモジュールは、モジュール名指定によって入力するか、LIBRARY オプション / サブコマンドを使用して自動的に入力します。LIBRARY をオプション / サブコマンドについては、「4.3.3 LIBRARY - ライブラリファイルの指定」を参照してください。

5.4 デフォルトライブラリファイル

Hシリーズ ライブラリアンで作成したライブラリファイルを LIBRARY オプション / サブコマンドを使用せずに暗黙のうちに入力することができます。これがデフォルトライブラリ機能です。

デフォルトライブラリファイルは以下の3つの条件が成立したとき入力されます。

- ・ リンケージエディタ入力前にデフォルトライブラリ名として予約している論理名にライブラリファイルを割り付けている場合。
- ・ NOLIBRARY オプション / サブコマンドを指定していない場合。
- ・ LIBRARY オプション / サブコマンドで指定したライブラリ内をサーチ後、未解決外部参照シンボルが存在する場合。

リンケージエディタは、論理名

1. HLNK_LIBRARY1
2. HLNK_LIBRARY2
3. HLNK_LIBRARY3

に割り付けているライブラリファイルを 1、2、3 の順番に入力し、未解決外部参照シンボルを定義しているモジュールをサーチします。ユーザが論理名に対応するライブラリファイルを指定するには、OS の set (MS-DOS 版) または、setenv (UNIX 版) コマンドを使用します。

(例)

```
set HLNK_LIBRARY1=user.lib ( MS-DOS 版 )
```

論理名 HLNK_LIBRARY1 に、ライブラリファイル"user.lib"を割り付けます。

6. リンケージエディタの出力

第6章 目次

6.1	リンケージリスト.....	109
6.2	ロードモジュールファイル	116
6.3	中間ファイル	117
6.4	コンソールメッセージ	118

6.1 リンケージリスト

PRINT オプション / サブコマンドまたは LIST サブコマンドを指定した場合、次のロードモジュールファイルの内容を標準出力またはファイルに出力します。

- (1) 入力情報 (PRINT のみ)
- (2) リンケージマップリスト (PRINT/LIST M)
- (3) 外部定義シンボルリスト (PRINT/LIST X)
- (4) 未定義シンボルリスト (PRINT/LIST U)
- (5) 変更 / 削除シンボルリスト (PRINT のみ)
- (6) 強制定義シンボルリスト (PRINT のみ)

次に、それぞれの出力形式を示します。

(1) 入力情報

コマンドパラメータ、会話形式サブコマンドまたはサブコマンドファイルで入力した内容を図 6.1 の形式で出力します。

```

LINK COMMAND LINE
LNK -SUB=FUNC.SUB
-----
(i)
LINK SUBCOMMANDS

INP main
RENAME ED=sin.sin0(sin1)
DELETE ED=sin.sin3
INP sin
DEFINE undef1(100),undef2(sin1)
PRINT fmap
INP cos
INP tan
INP calc.lib(division)
FORM a
ROM (SECT1,SECT1N)
OUT func
EXIT
** sin0 IS RENAMED TO sin1
** sin3 IS DELETED
** 105 UNDEFINED EXTERNAL SYMBOL(division.undef3)

```

(ii)

図 6.1 入力情報の出力例

(i) コマンドラインで入力した文字列を出力します。

(ii) 会話形式で入力した文字列またはサブコマンドファイルから入力した文字列を出力します。また、それに対するエラーメッセージおよびインフォメーションメッセージも出力します。

(2) リンケージマップリスト

(a) PRINT オプション / サブコマンドを指定した場合、セクション別に図 6.2 の形式で出力します。

*** LINKAGE EDITOR LINK MAP LIST ***									
SECTION	NAME	START -		END	LENGTH		UNIT NAME		
ATTRIBUTE :	CODE (2)	NOSHR (3)	ROM (4)						
<u>SECT1</u> (1)			H'00000000	-	H'00000004	H'00000005			
				(5)	(7)	(6)	<u>main</u>		
			H'00000006	-	H'00000017	H'00000012	sin		
							(8)		
			H'00000018	-	H'00000019	H'00000002	cos		
							tan		
			H'0000001A	-	H'0000002D	H'00000014	division		
							(9)		
			H'0000002E	-	H'00000043	H'00000016	division		
							(10)		
* TOTAL ADDRESS *			H'00000000	-	H'00000043	H'00000044			
				(9)		(10)			

図 6.2 PRINT で出力するマップリスト例

(b) LIST サブコマンドのパラメータで "M" を指定した場合、ファイル別に図 6.3 の形式で出力します。

*** LINKAGE EDITOR LINK MAP LIST ***									
FILE NAME :	<u>main.OBJ</u>								
	(11)								
MODULE NAME :	<u>main</u> (8)								
UNIT NAME :	<u>main</u> (7)								
SECTION NAME		ATTRIBUTE							
		START	-	END		LENGTH			
		CODE NOSHR							
<u>SECT1</u> (1)		H'00000000	-	H'00000004		H'00000005			
			(5)			(6)			

図 6.3 LIST で出力するマップリスト例

(1) セクション名を結合順に表示します。

(2) 内容種別を次のように表示します。

DATA …… データまたはコモン

CODE …… コード

DUMMY …… ダミー

STACK …… スタック

RESV …… 特殊

UNDEF …… 未定義

***** …… 未設定

(3) 結合属性を次のように表示します。

SHR.....共有結合

NOSHR...単純結合

DUMMY..ダミー結合

UNDEF...結合属性を定義していない

*****.....結合属性が設定されていない

(4) ROM 化支援機能に該当するセクションの場合に表示します。

ROM.....ROM 用セクション (ROM オプション / サブコマンドのセクション 1 に
相当するもの)

RAM.....RAM 用セクション (ROM オプション / サブコマンドのセクション 2 に
相当するもの)

(5) オブジェクトの先頭アドレスと最終アドレスを 16 進数で表示します。

ページタイプの場合は、次のようにページアドレスとアドレスをコロン (:) で区
切って表示します。

H' XXXXX : XXXXX
 ↑ ↑
 ページアドレス アドレス

(6) オブジェクトのアドレス長を 16 進数で表示します。

(7) ユニット名を表示します。

(8) モジュール名を表示します。

(9) 当該セクションの先頭アドレスと最終アドレスを表示します。

ページタイプの場合は、次のようにページアドレスとアドレスをコロン (:) で区
切って表示します。

H' XXXXX : XXXXX
 ↑ ↑
 ページアドレス アドレス

(10) 当該セクションの総アドレス長を表示します。

(11) ファイル名を表示します (LIST のみ)。

(3) 外部定義シンボルリスト

外部定義シンボルがあった場合に出力します。

(a) PRINT オプション / サブコマンドを指定した場合、図 6.4 の形式で出力します。

*** LINKAGE EDITOR EXTERNALLY DEFINED SYMBOLS LIST ***			
SYMBOL	NAME	ADDR	TYPE
	cos1	H'0000000A	EQU
	sin1	H'0000004A	DAT
	sin2	H'0000005B	DAT
(1)		(2)	(3)

図 6.4 PRINT で出力する外部定義シンボルリスト例

(b) LIST サブコマンドのパラメータで"X"を指定した場合、図 6.5 の形式で出力します。

*** LINKAGE EDITOR EXTERNALLY DEFINED SYMBOLS LIST ***			
SYMBOL	NAME	ADDR	TYPE
	cos1	H'0000000A	EQU
	sin1	H'00000000	DAT
	sin2	H'00000011	DAT
(1)		(2)	(3)

図 6.5 LIST で出力する外部定義シンボルリスト例

(1) 外部シンボル名をアルファベット順に表示します。

(2) 外部シンボルの値を 16 進数で表示します。

ページタイプの場合は、次のようにページアドレスとアドレスをコロン (:) で区切って表示します。

H' XXXX : XXXX

↑ ↑

ページアドレス アドレス

(3) シンボル種別を次のように表示します。

DAT.....データ / 変数名

EQU.....定数値として定義されたシンボル名

ENT.....入口名

*****未定義 / 未設定

(4) 未定義シンボルリスト

未定義シンボルがあった場合に出力します。

(a) PRINT オプション / サブコマンドを指定した場合、図 6.6 の形式で出力します。

```

*** LINKAGE EDITOR UNRESOLVED EXTERNAL REFERENCE LIST ***

FILE NAME      : calc.lib
                (1)
MODULE NAME    : division
UNIT NAME      : division (2)

SYMBOL  NAME    (3)      TYPE
undef3
                (4)      (5)

```

図 6.6 PRINT で出力する未定義シンボルリスト例

(b) LIST サブコマンドのパラメータで"U"を指定した場合、図 6.7 の形式で出力します。

```

*** LINKAGE EDITOR UNRESOLVED EXTERNAL REFERENCE LIST ***

FILE NAME      : calc.lib
                (1)
MODULE NAME    : division
UNIT NAME      : division (2)

SYMBOL  NAME    (3)      TYPE
undef1
undef2
undef3
                (4)      (5)

```

図 6.7 LIST で出力する未定義シンボルリスト例

- (1) 未定義シンボルを含むファイル名を表示します。
- (2) 未定義シンボルを含むモジュール名を表示します。
- (3) 未定義シンボルを含むユニット名を表示します。
- (4) 未定義シンボル名をアルファベット順に表示します。
- (5) 未定義シンボルの属性を次のように表示します。
 - DAT.....データ / 変数名
 - ENT.....入口名
 - ****.....未定義 / 未設定

(5) 変更 / 削除シンボルリスト

RENAME または DELETE サブコマンド指定により、ユニット名またはシンボル名の変更および削除処理を行った場合、PRINT オプション / サブコマンドを指定すると、図 6.8 に示す形式でリストを出力します。

*** LINKAGE EDITOR RENAME/DELETE LIST ***					
FILE NAME	:	sin.OBJ			
		(1)			
UNIT NAME	:	sin			
FROM NAME		(2)	TO NAME		TYPE RENAME/DELETE
sin0			sin1	ED	RENAME
sin3			(4)	ED	DELETE
(3)			(5)	(6)	

図 6.8 変更 / 削除シンボルリスト例

(1) 変更 / 削除の対象となるユニット名またはシンボル名を含むファイル名を入力順に表示します。

(2) ユニット名を表示します。ユニット名を変更または削除した場合は、旧ユニット名を表示します。

(3) 変更前の名前を表示します。

(4) 変更後の名前を表示します。削除の場合は表示しません。

(5) サブコマンドで指定した種別を表示します。

UN ユニット名

ED 外部定義シンボル名

ER 外部参照シンボル名

(6) 変更 / 削除の別を示します。

RENAME ・変更

DELETE ・削除

(6) 強制定義シンボルリスト

DEFINE オプション / サブコマンド指定により、外部参照シンボルの強制定義処理を行った場合、PRINT オプション / サブコマンドを指定すると、図 6.9 に示す形式でリストを出力します。

*** LINKAGE EDITOR DEFINE LIST ***		
UNDEFINE SYMBOL	DEFINED SYMBOL	DEFINED VALUE
undef1		H'00000100
undef2	sin1	H'0000004A
(1)	(2)	(3)

図 6.9 強制定義シンボルリスト例

- (1) 強制的に定義したシンボル名を表示します。
- (2) 外部定義シンボルを指定した場合は、そのシンボル名を表示します。
- (3) 定義したシンボルの値を 16 進数で表示します。

ページタイプの場合は、次のようにページアドレスとアドレスをコロン (:) で区切って表示します。

H' xxxx : xxxx

↑ ↑

アドレス

↑

ページアドレス

6.2 ロードモジュールファイル

リンケージエディタは、複数のオブジェクトモジュールファイルまたはリロケータブルロードモジュールファイルを1つに編集結合してロードモジュールファイルとして出力します。

FORM オプション / サブコマンドの指定により、アブソリュート形式とリロケータブル形式の2種類のロードモジュールファイルを出力します。FORM オプション / サブコマンドについては、「4.5.3 FORM - 出力ロードモジュールファイル形式の指定」を参照してください。

6.3 中間ファイル

環境変数により、中間ファイルの出力先のディレクトリを指定できるようになりました。
中間ファイルのディレクトリ指定は、HLNK_TMP 環境変数で指定します。

MS-DOS 版	set HLNK_TMP=<ディレクトリ名>
UNIX 版	setenv HLNK_TMP <ディレクトリ名>

6.4 コンソールメッセージ

リンケージエディタは次の6種類のメッセージを標準出力に表示します。

(1) 起動メッセージ

リンケージエディタのコマンドが入力されると表示します。

```
H SERIES LINKAGE EDITOR Ver. 5.3  
Copyright (C) Hitachi, Ltd.1989  
Copyright (C) HITACHI MICROCOMPUTER SYSTEM LTD. 1990  
Licensed Material of Hitachi, Ltd.
```

(2) 正常終了メッセージ

ロードモジュールファイルの編集処理が正常に終了した場合に表示します。

```
LINKAGE EDITOR COMPLETED
```

(3) 異常終了メッセージ

ロードモジュールファイル編集中にエラーが発生し処理を打ち切った場合、または、
ABORT サブコマンド指定により処理を打ち切った場合に表示します。

```
LINKAGE EDITOR ABORT
```

(4) サブコマンド要求記号

会話形式の実行においてサブコマンドの入力要求を示すために、コロンの(:)を表示します。

```
:
```

(5) サブコマンド継続記号

会話形式の実行においてサブコマンド行の継続を指定した場合、次行に継続行入力要求を示すためにマイナス(-)を表示します。

```
-
```

(6) インフォメーションメッセージ

ユニットの置換や外部定義シンボル名の変更などがあった場合、リンケージエディタの処理結果を示すインフォメーションメッセージを表示します。

インフォメーションメッセージの出力形式は、次のとおりです。

** <インフォメーションメッセージ>

1 カラム目

表 6.1 にインフォメーションメッセージ一覧を示します。表中の外部名には、ユニット名も含まれます。

表 6.1 インフォメーションメッセージ一覧表

項 番	インフォメーションメッセージ
	インフォメーションの内容
1	ユニット名 ₁ IS REPLACED WITH ユニット名 ₂ (ファイル名)
	ユニット名 ₁ を ファイル名 中の ユニット名 ₂ に置き換えた。
2	外部名 ₁ IS RENAMED TO 外部名 ₂
	外部名 ₁ を 外部名 ₂ に変更した。
3	外部名 IS DELETED
	外部名 を削除した。
4	DUPLICATE UNIT - (ユニット名) IN (ファイル名) IS DELETED
	ユニット名 のユニットを複数見つけたため、 ファイル名 中のユニットを削除した。
5	外部参照シンボル名 CANNOT DEFINED
	外部参照シンボル名 が見つからないため、強制定義できなかった。
6	外部名 CANNOT RENAMED
	外部名 が見つからないため、変更できなかった。
7	外部名 CANNOT DELETED
	外部名 が見つからないため、削除できなかった。
8	ユニット名 CANNOT REPLACED
	ユニット名 が見つからないため、置き換えができなかった。

7. エラーメッセージ

コマンドやサブコマンドの指定内容に誤りがあったり、リンケージ処理中にエラーを検出した場合、エラーメッセージを出力します。

エラーメッセージの出力形式は、次のとおりです。

** <エラー番号> <エラーメッセージ> [(<付加情報>)]
1 カラム目

- エラー番号：上1桁でエラーのレベルを示します。
- 1 x x：ウォーニング………当該モジュールの処理をスキップします。
 - 2 x x：エラー……… コマンドまたはサブコマンドファイルからの入力時には、処理を中断します。会話形式の場合には、エラーを検出した時点で当該サブコマンドの処理を打ち切り、次のサブコマンド要求になります。
 - 3 x x：フェイタルエラー… 処理を中断します。

表 7.1、7.2 および 7.3 にエラーメッセージ一覧を示します。各表の記述形式は次のとおりです。

エラー番号	エラーメッセージ	付加情報
エラー内容		
対応策その他		

(表中の記号説明) —：付加情報なしを示します。

表 7.1 ウォーニングメッセージ一覧表 (1)

101	DUPLICATE OPTION/SUBCOMMAND	オプション/サブコマンド名
同じオプションまたはサブコマンドを重複して指定している。		
後に指定したオプションまたはサブコマンドが有効になります。		
102	IDENTIFIER CHARACTER EXCEEDS 251	名前
251 文字を越える名前 (ユニット名、セクション名、シンボル名) を指定した。		
251 文字までが有効になります。		
104	DUPLICATE SYMBOL	シンボル名
外部定義シンボルが重複している。		
先に現われた外部定義シンボルが有効になります。		
105	UNDEFINED EXTERNAL SYMBOL	エント名、シンボル名
未定義の外部シンボルを参照している。		
外部参照は無効になり、0 (ゼロ) を仮定します。		
106	REDEFINED SYMBOL	シンボル名
定義済みのシンボルを DEFINE オプション / サブコマンドで定義した。		
DEFINE オプション / サブコマンドの指定は無効になります。		
107	SECTION ATTRIBUTE MISMATCH	セクション名
属性または境界調整数の異なる同名セクションを入力した。		
別セクションとして扱います。		
108 (注)	RELOCATION SIZE OVERFLOW	エント名、セクション名 - reloc 値
リロケーションの結果がリロケーションサイズを超えた。		
リロケーションサイズの範囲内にまるめ込みます。		
109	ENTRY POINT MULTIPLY DEFINED	---
実行開始アドレスの指定があるオブジェクトモジュールを複数指定した。		
先に現われた実行開始アドレスの指定が有効になります。		
110	SECTION ADDRESS EXCEED PAGE BOUNDARY	セクション名
セクションがページの境界にまたがっている。		
AUTOPAGE オプション / サブコマンドを指定してください。		
111	DUPLICATE SECTION NAME	セクション名
オプション / サブコマンドで同一セクション名を指定した。		
最初に指定したセクションを有効にします。		
112	ILLEGAL CPU INFORMATION FILE FORMAT	---
CPU 情報ファイルのファイル形式が正しくない。		
CPU オプション / サブコマンドの指定を無効にします。		
113	CONFLICTING DEVICE TYPE	---
入力オブジェクトモジュールの対象 CPU と異なる CPU 情報ファイルを指定している。		
CPU 情報ファイルの指定を無効にします。		

(次ページへ続く)

表 7.1 ウォーニングメッセージ一覧表 (2)

114	SECTION IS NOT IN SAME MEMORY AREA	セクション名:xxxx - yyyy
セクションが1つのメモリ領域内に入りきらず、xxxx 番地から yyyy 番地が異なるメモリ領域に割り付けられている。		
ロードモジュールにはそのまま出力します。		
115	INACCESSIBLE ADDRESS RANGE	セクション名
セクションが使用できない領域に割り付けられている。		
ロードモジュールにはそのまま出力します。		
116	INVALID CPU OPTION/SUBCOMMAND	---
ロードモジュールファイルをリロケータブル形式に指定して、CPU オプション / サブコマンドを指定している。		
CPU オプション / サブコマンドの指定を無効にします。		
117	ADDRESS SPACE DUPLICATE	---
セクションが重複している。		
ロードモジュールにはそのまま出力します。		
118	INVALID UDF OPTION/SUBCOMMAND	---
出力ロードモジュール形式がアブソリュート指定に対し、NOUDF オプション / サブコマンドを指定している。		
NOUDF オプション / サブコマンド指定を無効にします。		
119	RELOCATION VALUE IS ODD	エント名.セクション名 - reloc 値
ディスプレイメントに対するリロケーション結果が奇数である。		
リロケーションサイズの範囲内に入るよう、最下位ビットを切り落とします。		
120	START ADDRESS NOT SPECIFIED FOR SECTION	セクション名
START オプション / サブコマンドで指定しなかったセクションが存在した。		
セクション名を確認してください。		
121	CANNOT FIND SECTION	セクション名
指定したセクションが見つからない。		
セクションの指定を無効とします。		
122	TOO LONG SUBCOMMAND LINE	---
ディレクトリ名の置き換えで文字数が 511 文字を超えた。		
511 文字までを有効とします。		
123	TOO MANY DIRECTORY COMMANDS	---
DIRECTORY サブコマンドで 16 個を超えたディレクトリ名を指定した。		
16 個までを有効とします。		
124	NO DEBUG INFORMATION	---
デバッグ情報の全くないファイルに対して DEBUG、SDEBUG オプション / サブコマンドを指定した。		
コンパイル、アセンブル時にデバッグオプションを指定してください。		

(次ページへ続く)

表 7.1 ウォーニングメッセージ一覧表 (3)

125	CANNOT SET ENTRY POINT	---
出力ロードモジュールがリロケータブル形式に対して、実行開始アドレスに定数の外部参照シンボルを指定した。		
出力ロードモジュールをアブソリュート形式にするか、実行開始アドレスの指定を削除してください。		
126	TOO LONG CHARACTER NUMBER (FSYMBOL)	---
FSYMBOL オプション / サブコマンドで指定したセクション内のシンボルの文字数が 238 文字を超えた。		
該当するシンボルの文字数を 238 文字以下に変更してください。		
127	EXTERNAL SYMBOL 0	セクション名
指定したセクション内に外部定義シンボルが存在しない。		
セクション名を確認してください。		
128	ILLEGAL SYMBOL REFERENCE	シンボル名
同一アドレスに割り付けたセクション間でシンボルの参照があった。		
同一アドレスに割り付けたセクション間でシンボルを参照しないようプログラムを変更してください。		

【注】 ウォーニング 108 (RELOCATION SIZE OVERFLOW) についての発生条件、発生するプログラムの例、対策を下記に示します。

(1) 発生条件

リンケージエディタは、プログラムのアドレスを決定しますが、その際、アセンブル時または C コンパイル時に決定しているデータエリアのサイズに収まらない値が存在する場合に発生します。

(2) 発生するプログラムの例について

(a) H8S,H8/300 シリーズの場合

(例 1)

プログラム1		プログラム2	
	.EXPORT SYM1		.IMPORT SYM1
SYM1	.EQU H'1000	⋮	
	⋮	MOV.B	#SYM1,R1L ⋯ (1)
	⋮		

上記の 2 つのプログラムをアセンブルし、リンクした場合、(1) の命令で SYM1 をバイトで参照しており、参照の値の範囲が -128 ~ +255 である。しかし、プログラム 1 で SYM1 を H'1000 (4096) と定義しており値の範囲を超えるため、ウォーニング 108 を出力します。

(例2)

プログラム3		プログラム4	
	.EXPORT SYM2		.IMPORT SYM2
SYM2	.EQU H'C0 ... (2)		⋮
	⋮		⋮
		MOV	@SYM2:8,R0L ... (3)

上記の2つのプログラムをアセンブルし、リンクした場合、(3)でSYM2を8ビット絶対アドレスで参照している。8ビット絶対アドレスの場合のアクセス範囲は、65280～65535 (H'FF00～H'FFFF)です。しかし、(2)でSYM2をH'C0に外部定義しており、65280～65535 (H'FF00～H'FFFF)の範囲にないため、ウォーニング108を出力します。しかし、@SYM2:8によってH'FFC0番地をアクセスするので、@H'FFC0番地をアクセスしようとしている場合には、本メッセージは無視できます。

(b) H8/500シリーズの場合

(例3)

プログラム5		プログラム6	
	.EXPORT SYM3		.IMPORT SYM3
SYM3	.EQU H'FF		⋮
	⋮		⋮
		MOV	@(SYM3:8,R2),R3 ... (4)

上記の2つのプログラムをアセンブルし、リンクした場合、(4)の命令でSYM3を8ビットで参照しており、参照の値の範囲が-128～+127である。しかし、プログラム5でSYM3をH'FF(255)と定義しており値の範囲を超えるため、ウォーニング108を出力します。

(例4)

	.SECTION SEC1, CODE	
SYM4	.EQU \$;シンボルにロケーション値をセット
	⋮	
	MOV @SYM4:8,R0	;ロケーションが示す番地のデータ2バイトを転送 ... (5)

上記のプログラムをアセンブルし、セクション(SEC1)の先頭アドレスに1000番地(16進)を指定してリンクした場合、SYM4の値がH'1000になり、1バイトのデータサイズを超えるためウォーニング108を出力します。しかし、(5)の命令実行前にベースレジスタ(BR)にH'10が設定されている場合には、本メッセージは無視できます。

(例5)

プログラム7			プログラム8		
	.EXPORT	SYM5	.IMPORT	SYM5	
SYM5	.EQU	H'2000	⋮		
	⋮		MOV	@SYM5:8,R0	⋯ (6)

上記の2つのプログラムをアセンブルし、リンクした場合、(6)のSYM5の値をプログラム9でH'2000に外部定義しており、1バイトのデータサイズを超えるため、ウォーニング108を出力します。この場合も例4と同様、(6)の実行前にベースレジスタ(BR)にH'20が設定されている場合には、本メッセージは無視できます。

(3) 対策方法

本メッセージを無視できない場合には、以下に示すような対策方法があります。

(a) H8S,H8/300シリーズの場合

(i) 例1では、命令のオペレーションサイズをワードサイズに修正する方法と、ラベル(SYM1)の値の上位1バイト、または下位1バイトを取り出すように修正する方法がある。オペレーションサイズを変更する場合は、(1)の"MOV.B"を"MOV.W"とし、"R1L"を"R1"に修正する。

ラベルの値から上位1バイトを取り出す場合は、(1)の"#SYM1"を"#HIGH SYM1"に修正する。下位1バイトを取り出す場合は、"HIGH"演算子を"LOW"演算子にする。

(ii) 例2では、(2)のH'C0をH'FFC0に修正する。

(b) H8/500シリーズの場合

(i) 例3では、ラベル(SYM3)の値が1バイトのデータサイズを超える場合には、(4)の"SYM3:8"を"SYM3:16"に修正する。

(ii) 例4では、(5)の"@SYM4:8"を"@SYM4:16"に修正する。

(iii) 例5では、(6)の"@SYM5:8"を"@SYM5:16"に修正する。

(4) ウォーニング108のメッセージ出力形式について

ウォーニング108のメッセージは、以下の形式で出力されます。

** 108 RELOCATION SIZE OVERFLOW (ユニット名.セクション名 - オフセット値)

ここで、オフセット値とは、データオーバーフローの発生箇所が、"ユニット名.セクション名"で示すセクションの先頭から数えて"オフセット値"番地目に存在することを示します。

ここで、ユニット名とは、ファイル名を表しています。

表 7.2 エラーメッセージ一覧表 (1)

201	ILLEGAL SUBCOMMAND/OPTION	---
不正なサブコマンド名 (またはオプション名) を指定した。		
正しいサブコマンド名 (またはオプション名) を指定してください。		
202	SYNTAX ERROR	---
指定されたサブコマンド (またはオプション) に構文上の不正がある。		
サブコマンド (またはオプション) の構文を確認の上、再入力してください。		
203	TOO LONG SUBCOMMAND LINE	---
サブコマンドの長さが 512 文字を越えている。		
サブコマンドの長さが 512 文字に納まるように再入力してください。		
204	ILLEGAL SUBCOMMAND SEQUENCE	---
サブコマンドの指定順序が不正である。		
サブコマンドの指定順序を確認の上、再入力してください。		
207	ILLEGAL SECTION NAME	セクション名
不正なセクション名を指定した。		
正しいセクション名を指定してください。		
208	ILLEGAL SYMBOL NAME	シンボル名
不正なシンボル名を指定した。		
正しいシンボル名を指定してください。		
210	TOO MANY INPUT FILES	---
65,535 個を越えて入力ファイルを入力しようとした。		
いったんリロケータブル形式のロードモジュールファイルを作成し、ロードモジュールファイルの再入力により残りの入力ファイル名を指定してください。		
211	CANNOT FIND FILE	ファイル名
指定したファイルが見つからない。		
指定したファイル名を確認の上、再指定してください。		
212	CANNOT FIND UNIT	ユニット名
指定したユニットが見つからない。		
指定したユニット名を確認の上、再指定してください。		
213	CANNOT FIND MODULE	モジュール名
指定したモジュールが見つからない。		
指定したモジュールを確認の上、再指定してください。		

(次ページへ続く)

表 7.2 エラーメッセージ一覧表 (2)

214	DUPLICATE START ADDRESS SPECIFIED	---
同じ先頭アドレスを重複して指定した。		
先頭アドレスを変更して、再入力してください。		
216	PAGE ADDRESS EXCEEDED	---
ページアドレスが許容範囲を越えた。		
ページアドレスを見直し、再度指定してください。		
217	SUBCOMMAND COMMAND IN SUBCOMMAND FILE	---
サブコマンドファイル中に SUBCOMMAND サブコマンドが現われた。		
サブコマンドファイル中の SUBCOMMAND サブコマンドを削除してください。		
219	INVALID ADDRESS	アドレス
指定したアドレスが許容範囲を越えた。		
指定したアドレスが指定デバイスのアドレス範囲を越えています。指定アドレスの内容を確認の上、再実行してください。		
220	TOO MANY ROM COMMANDS	---
ROM サブコマンドで 64 組を越えたセクションを指定した。		
64 組以内になるように指定してください。		
221	CANNOT CREATE ABSOLUTE MODULE	モジュール名
未定義の外部参照シンボルが存在した。		
未定義の外部参照シンボルのアドレスを解決してください。		
222	DIVISION BY ZERO IN RELOCATION VALUE	エントリ名.セクション名-リセット値
0 除算を含むオブジェクトファイルを入力した。		
リロケーション演算を確認の上、0 除算が発生しないようにしてください。		

表 7.3 フェイタルエラーメッセージ一覧表 (1)

301	ILLEGAL COMMAND PARAMETER	---
不正なコマンドパラメータを指定した。		
コマンドパラメータの内容を確認の上、再実行してください。		
302	CANNOT OPEN FILE	ファイル名
ファイルをオープンできなかった。		
指定したファイル名を確認してください。ファイル名が正しい場合はディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。ライブラリファイルと同一ディレクトリに付加情報ファイル (*.lct) が存在しているか確認してください。		
303	CANNOT READ INPUT FILE	ファイル名
ファイルから入力できなかった。		
指定したファイル名を確認してください。ファイル名が正しい場合はディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
304	CANNOT WRITE OUTPUT FILE	ファイル名
ファイルへ出力できなかった。		
指定したファイル名を確認してください。ファイル名が正しい場合はディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
305	CANNOT CLOSE FILE	ファイル名
ファイルをクローズできなかった。		
指定したファイル名を確認してください。ファイル名が正しい場合はディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
306	ILLEGAL FILE FORMAT	ファイル名
指定したファイルのフォーマットが不正である。		
ファイルの内容および指定したファイル名を確認の上、再実行してください。1つのユニットに同一名称の外部参照シンボルが複数ある場合など、オブジェクト不正時に出力します。		
RENAME サブコマンドで外部シンボル名が同一になったときにも出力します。		
307	ILLEGAL RECORD FORMAT	ファイル名
指定したファイル中に不正なレコードがある。または、除数が0 (ゼロ) の除算が起きた。		
ソースプログラムの内容を確認の上、再アセンブル (または再コンパイル) 後、再実行してください。		
308	SECTION ADDRESS OVERFLOW	セクション名
セクションの割り付けアドレスが許容範囲を越えた。		
セクションの割り付けアドレスが指定デバイスのアドレス範囲を越えています。セクション先頭アドレスまたはユーザプログラムの構成の変更を行い、再実行してください。		

(次ページへ続く)

表 7.3 フェイタルエラーメッセージ一覧表 (2)

309	ADDRESS OVERFLOW	---
指定したアドレスが許容範囲を越えた。		
指定したアドレスが各 CPU で許されるアドレス範囲を越えています。指定アドレスの内容を確認の上、再実行してください。		
310	MEMORY OVERFLOW	---
リンケージエディタの使用可能なメモリにスペースがない。		
メモリの拡張またはユーザプログラムの変更を行い、再実行してください。		
311	PROGRAM ERROR	nnn
リンケージエディタの内部処理で何らかの障害が生じました。		
エラー番号 654 が発生した場合は、リンケージエディタが生成する付加情報ファイル(*.rct)またはライブラリアンが生成する付加情報ファイル(*.lct)とオブジェクトモジュール、ライブラリファイルの作成日付が同一であるか確認してください。		
それ以外についてはエラー番号(nnn)を確認の上、当社営業担当までご連絡ください。		
312	ILLEGAL START ADDRESS ALIGNMENT	アドレス
オブジェクトモジュールの境界調整数と矛盾するアドレスを指定した。		
オブジェクトモジュールの境界調整数を確認の上、再指定してください。		
314	CANNOT FIND SECTION	セクション名
指定したセクションが見つからない。		
指定したセクション名を確認の上、再指定してください。		
319	AUTOPAGE SPECIFIED AT NON-PAGE TYPE	---
非ページタイプの入力ファイルに対して AUTOPAGE オプション / サブコマンドを指定した。		
指定した入力ファイルの内容を確認の上、再指定してください。		
321	PAGE ADDRESS OVERFLOW	---
ページアドレスが許容範囲を越えた。		
ページアドレスが 0-0FF (16 進数) を越えています。セクション先頭アドレスまたはユーザプログラムの構成の変更を行い、再実行してください。		
322	PAGE ADDRESS SPECIFIED AT NON-PAGE TYPE	---
非ページタイプの入力ファイルに対してページアドレスを指定した。		
非ページタイプの入力ファイルに対して、START オプション / サブコマンドまたは DEFINE オプション / サブコマンドでページアドレスを指定しています。指定したファイル名、オプションまたはサブコマンドの内容を確認の上、再実行してください。		

(次ページへ続く)

表 7.3 フェイタルエラーメッセージ一覧表 (3)

323	SECTION SPECIFIED AT ROM OPTION /SUBCOMMAND DOES NOT EXIST	セクション名
ROM コマンドで指定したセクションは存在しない。		
セクション名を確認の上、再指定してください。		
325	ILLEGAL START SECTION	セクション名
START コマンドで指定したセクションの属性が不正である。		
セクションの属性を確認の上、再指定してください。		
326	CANNOT READ	---
ファイル (標準入力を含む) からの入力ができなかった。		
指定したファイル名を確認してください。ファイル名が正しい場合はディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
327	SYMBOL ADDRESS OVERFLOW	シンボル名
シンボルの割り付けアドレスが許容範囲を越えた。		
シンボルの割り付けアドレスが指定デバイスのアドレス範囲を越えています。セクション先頭アドレスまたはユーザプログラムの構成の変更を行い、再実行してください。		
328	ILLEGAL ROM SECTION	セクション名
ROM オプション / サブコマンドの指定でセクション 2 が不正である。		
セクション 2 のサイズが 0 以外、セクション 2 が絶対番地セクション、またはセクション 1 とセクション 2 の属性が違います。セクション 2 のサイズ、属性を確認の上、再指定してください。		
329	INVALID MEMORY MAP	---
CPU 情報ファイルと矛盾したメモリに割り付けられている。または、異なるメモリ種別にまたがって割り付けられている。		
CPU 情報ファイルと入力ファイルの内容を確認してください。		
330	ILLEGAL FILE FORMAT (INPUT ABSOLUTE FILE)	---
アブソリュートロードモジュールを入力した。		
指定した入力ファイルの内容を確認の上、再指定してください。		
331	ILLEGAL FILE FORMAT (MISMATCH OBJECT FORMAT VERSION)	---
オブジェクト形式の異なるファイルを入力した。		
指定した入力ファイルの内容を確認の上、再指定してください。		
332	ILLEGAL FILE FORMAT (INPUT MISMATCH CPU TYPE)	---
H シリーズ、SH シリーズ以外のオブジェクトファイルを入力した。		
指定した入力ファイルの内容を確認の上、再指定してください。		

(次ページへ続く)

表 7.3 フェイタルエラーメッセージ一覧表 (4)

333	CANNOT OPEN INTERNAL FILE	---
中間ファイルをオープンできなかった。		
環境変数 HLNK_TMP で指定したディレクトリ名を確認してください。ディレクトリ名が正しい場合はディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
334	CANNOT CLOSE INTERNAL FILE	---
中間ファイルがクローズできなかった。		
環境変数 HLNK_TMP で指定したディレクトリ名を確認してください。ディレクトリ名が正しい場合はディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
335	CANNOT DELETE INTERNAL FILE	---
中間ファイルが削除できなかった。		
環境変数 HLNK_TMP で指定したディレクトリ名を確認してください。ディレクトリ名が正しい場合はディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
336	CANNOT OUTPUT INTERNAL FILE	---
中間ファイルに出力できなかった。		
環境変数 HLNK_TMP で指定したディレクトリ名を確認してください。ディレクトリ名が正しい場合はディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
337	CANNOT READ INTERNAL FILE	---
中間ファイルから入力できなかった。		
環境変数 HLNK_TMP で指定したディレクトリ名を確認してください。ディレクトリ名が正しい場合はディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
338	DUPLICATE START ADDRESS SPECIFIED IN PAGE TYPE	---
ページタイプの入力ファイルに対して、複数セクションの同一アドレス割り付けを指定した。		
ページタイプは、複数セクションの同一アドレス割り付けができません。		
339	TOO MANY UNITS	---
指定したユニット数が 65,535 を超えた。		
ユニットを統一するなどしてユニット数を 65,535 以内にしてください。		
340	TOO MANY SECTIONS	---
指定したセクション数が 65,535 を超えた。		
セクションを統一するなどしてセクション数を 65,535 以内にしてください。		

8. 制限事項一覽

表 8.1 にリンケージエディタの制限事項一覧を示します。リンケージエディタでは、これらの制限事項を超える処理はできません。

表 8.1 制限事項一覧表

項番	項目	制限値	備考
1	入力ファイル数	最大 65,535 個	
2	入力ファイル形式	<ul style="list-style-type: none"> ・アセンブラ、コンパイラが出力したオブジェクトモジュールファイル ・リロケートブル形式のロードモジュールファイル ・ライブラリアン作成のライブラリファイル 	
3	アドレス / 数値指定	16 進数による指定のみ。 指定範囲は H シリーズの品種により異なる。	H8/500 シリーズ : 0 ~ 0FFFF H8/300 シリーズ : (300, 300L, 300HN) 0 ~ 0FFFF H8/300 シリーズ : (300HA) 0 ~ 0FFFFFFF H8S シリーズ : (2600N, 2000N) 0 ~ 0FFFF H8S シリーズ : (2600A, 2000A) 0 ~ 0FFFFFFF SuperH シリーズ : 0 ~ 0FFFFFFF
4	モジュール名 ユニット名 セクション名 シンボル名	251 文字まで	
5	モジュール数 ユニット数 セクション数 外部定義シンボル数 外部参照シンボル数	最大 65,535 個	ただし、リンケージエディタが稼動するシステムのメモリ容量によって制限を受けない場合

付録

付録 目次

付録 A. リンケージエディタ使用例	141
付録 B. ファイル名の構成	151

付録 A. リンケージエディタ使用例

リンケージエディタの使用例を表 A.1 に示す 11 個のオブジェクトモジュールファイルと 1 個のライブラリファイルを入力する例を用いて説明します。

表 A.1 入力ファイル一覧表

No.	ファイル名	ファイル種別
1	main.obj	オブジェクトモジュールファイル
2	init.obj	
3	cmndanl.obj	
4	cmndprc.obj	
5	table.obj	
6	term.obj	
7	keyin.obj	
8	file.obj	
9	printer.obj	
10	display.obj	
11	commu.obj	
12	function.lib	ライブラリファイル

また、ライブラリファイル"function.lib"は表 A.2 に示す 14 個のモジュールで構成されています。

表 A.2 ライブラリ内モジュール一覧表

No.	モジュール名	No.	モジュール名
1	mvdata	8	number
2	upshft	9	zerosprs
3	comp	10	ascbn
4	expr	11	binasc
5	rmargin	12	cnvbcd
6	lmargin	13	portio
7	sum	14	dos

(1) リンケージエディタの実行

次のコマンドラインを入力し、リンケージエディタを実行します。ここではサブコマンドファイル"exlink.sub"からサブコマンドを入力し、サブコマンド指定による実行例を示します。

```
lnk -SUBCOMMAND=exlink.sub (RET)
```

図 A.1 にサブコマンドファイル"exlink.sub"の内容を示します。

```
;
; First Linkage Process
;
form      r          ; Relocatable Load Module
input     MAIN       ; Input "MAIN.OBJ"
input     INIT       ; Input "INIT.OBJ"
input     CMNDANL    ; Input "CMNDANL.OBJ"
input     CMNDPRC    ; Input "CMNDPRC.OBJ"
input     TABLE     ; Input "TABLE.OBJ"
input     TERM       ; Input "TERM.OBJ"
library   FUNCTION   ; Library "FUNCTION.LIB"
output    PROGRAM1   ; Output "PROGRAM1.REL"
print     PROGRAM1   ; Print "PROGRAM1.MAP"
end
;
; Second Linkage Process
;
input     PROGRAM1.REL ; Input "PROGRAM1.REL"
input     KEYIN       ; Input "KEYIN.OBJ"
input     FILE        ; Input "FILE.OBJ"
input     PRINTER     ; Input "PRINTER.OBJ"
input     DISPLAY     ; Input "DISPLAY.OBJ"
input     COMMU       ; Input "COMMU.OBJ"
library   FUNCTION    ; Library "FUNCTION.REL"
; Sequence of Sections
start     program1,program2,function,global,local,f_local,stack_area
output    EXAMPLE     ; Output "EXAMPLE.ABS"
print     EXAMPLE     ; Print "EXAMPLE.MAP"
exit
```

図 A.1 サブコマンドファイル"exlink.sub"

図 A.1 に示すように、マルチリンケージ機能を利用して2度のリンケージ処理を行っています。最初のリンケージ処理では、6個のオブジェクトモジュールファイルとライブラリファイルを入力し、リロケートブル形式のロードモジュールファイル"program1.rel"とリンケージリスト"program1.map"を出力します。2度目のリンケージ処理では、ロードモジュールファイル"program1.rel"の再入力と残りのオブジェクトモジュールファイルを入力し、アブソリュート形式のロードモジュールファイル"example.abs"とリンケージリスト"example.map"を出力します。

図 A.2 に 1 度目のリンケージ処理で出力したリンケージリスト"program1.map"の内容を示します。また、図 A.3 に 2 度目のリンケージ処理で出力したリンケージリスト"example.map"の内容を示します。

```
H SERIES LINKAGE EDITOR Ver. 5.3

LINK COMMAND LINE

LNK -SUBCOMMAND=EXLINK.SUB

LINK SUBCOMMANDS

;
; First Linkage Process
;
form      r          ; Relocatable Load Module
input     MAIN       ; Input "MAIN.OBJ"
input     INIT       ; Input "INIT.OBJ"
input     CMNDANL    ; Input "CMNDANL.OBJ"
input     CMNDPRC    ; Input "CMNDPRC.OBJ"
input     TABLE     ; Input "TABLE.OBJ"
input     TERM       ; Input "TERM.OBJ"
library   FUNCTION   ; Library "FUNCTION.LIB"
output    PROGRAM1   ; Output "PROGRAM1.REL"
print     PROGRAM1   ; Print "PROGRAM1.MAP"
end
** 105 UNDEFINED EXTERNAL SYMBOL(main.keyin)
** 105 UNDEFINED EXTERNAL SYMBOL(cmndprc.printer)
** 105 UNDEFINED EXTERNAL SYMBOL(cmndprc.file)
** 105 UNDEFINED EXTERNAL SYMBOL(cmndprc.keyin)
** 105 UNDEFINED EXTERNAL SYMBOL(cmndprc.commu)
** 105 UNDEFINED EXTERNAL SYMBOL(cmndprc.display)
** 105 UNDEFINED EXTERNAL SYMBOL(term.file)
```

図 A.2 リンケージリスト"program1.map"の内容 (入力情報)

H SERIES LINKAGE EDITOR Ver. 5.3						PAGE :	1
*** LINKAGE EDITOR LINK MAP LIST ***							
SECTION	NAME	START	-	END	LENGTH		
				UNIT NAME	MODULE NAME		
ATTRIBUTE :	CODE	NOSHR					
program1		H'00000000	-	H'00000349	H'0000034A		
				main	main		
		H'0000034A	-	H'00000467	H'0000011E		
				init	initialize		
		H'00000468	-	H'0000055D	H'000000F6		
				cmdanl	command_analyze		
		H'0000055E	-	H'000007E7	H'0000028A		
				cmdprc	command_process		
		H'000007E8	-	H'0000091F	H'00000138		
				term	terminate		
* TOTAL ADDRESS *		H'00000000	-	H'0000091F	H'00000920		
ATTRIBUTE :	DATA	NOSHR					
local		H'00000000	-	H'00001E1F	H'00001E20		
				main	main		
		H'00001E20	-	H'00001E3F	H'00000020		
				init	initialize		
		H'00001E40	-	H'00003C7F	H'00001E40		
				cmdanl	command_analyze		
		H'00003C80	-	H'000222BF	H'0001E640		
				cmdprc	command_process		
		H'000222C0	-	H'000222DF	H'00000020		
				term	terminate		
* TOTAL ADDRESS *		H'00000000	-	H'000222DF	H'000222E0		
ATTRIBUTE :	DATA	NOSHR					
global		H'00000000	-	H'000015CF	H'000015D0		
				table	global_table		
* TOTAL ADDRESS *		H'00000000	-	H'000015CF	H'000015D0		
ATTRIBUTE :	STACK	NOSHR					
stack_area		H'00000000	-	H'001E1FFF	H'001E2000		
				table	global_table		
* TOTAL ADDRESS *		H'00000000	-	H'001E1FFF	H'001E2000		

図 A.2 リンケージリスト"program1.map"の内容 (リンケージマップリスト)

H SERIES LINKAGE EDITOR Ver. 5.3					PAGE : 2	
*** LINKAGE EDITOR LINK MAP LIST ***						
SECTION	NAME	START	-	END	LENGTH	
				UNIT NAME	MODULE NAME	
ATTRIBUTE :	CODE	NOSHR				
function	H'00000000		-	H'0000001B	H'0000001C	
				comp	compare_string	
	H'0000001C		-	H'0000010F	H'000000F4	
				expr	expression	
	H'00000110		-	H'00000163	H'00000054	
				mvdata	move_data_string	
	H'00000164		-	H'00000193	H'00000030	
				upshft	upshift_character	
* TOTAL ADDRESS *	H'00000000		-	H'00000193	H'00000194	
ATTRIBUTE :	DATA	NOSHR				
f_local	H'00000000		-	H'0000000B	H'0000000C	
				comp	compare_string	
	H'0000000C		-	H'0000011B	H'00000110	
				expr	expression	
	H'0000011C		-	H'0000011F	H'00000004	
				upshft	upshift_character	
* TOTAL ADDRESS *	H'00000000		-	H'0000011F	H'00000120	

図 A.2 リンケージリスト"program1.map"の内容 (リンケージマップリスト)

H SERIES LINKAGE EDITOR Ver. 5.3					PAGE : 1	
*** LINKAGE EDITOR EXTERNALLY DEFINED SYMBOLS LIST ***						
SYMBOL	NAME		ADDR		TYPE	
cmndanl			H'00000000		DAT	
cmndprc			H'00000000		DAT	
cmndtbl			H'000000C8		DAT	
comp			H'00000000		DAT	
expr			H'00000000		DAT	
fltbl			H'000003C8		DAT	
header			H'00000000		DAT	
init			H'00000000		DAT	
keybuf			H'000001C8		DAT	
main			H'00000000		DAT	
mvdata			H'00000000		DAT	
prbuf			H'000014C8		DAT	
recbuf			H'000013C8		DAT	
stackarea			H'00000000		DAT	
term			H'00000000		DAT	
upshft			H'00000000		DAT	

図 A.2 リンケージリスト"program1.map"の内容 (外部定義シンボルリスト)

H SERIES LINKAGE EDITOR Ver. 5.3		PAGE :	1
*** LINKAGE EDITOR UNRESOLVED EXTERNAL REFERENCE LIST ***			
FILE NAME	: MAIN.OBJ		
MODULE NAME	: main		
UNIT NAME	: main		
	SYMBOL	NAME	TYPE
	keyin		***
FILE NAME	: CMNDPRC.OBJ		
MODULE NAME	: command_process		
UNIT NAME	: cmndprc		
	SYMBOL	NAME	TYPE
	commu		***
	display		***
	file		***
	keyin		***
	printer		***
FILE NAME	: TERM.OBJ		
MODULE NAME	: terminate		
UNIT NAME	: term		
	SYMBOL	NAME	TYPE
	file		***

図 A.2 リンケージリスト"program1.map"の内容 (未定義シンボルリスト)

H SERIES LINKAGE EDITOR Ver. 5.3	
LINK COMMAND LINE	
LINK SUBCOMMANDS	
;	
; Second Linkage Process	
;	
input	PROGRAM1.REL ; Input "PROGRAM1.REL"
input	KEYIN ; Input "KEYIN.OBJ"
input	FILE ; Input "FILE.OBJ"
input	PRINTER ; Input "PRINTER.OBJ"
input	DISPLAY ; Input "DISPLAY.OBJ"
input	COMMU ; Input "COMMU.OBJ"
library	FUNCTION ; Library "FUNCTION.REL"
; Sequence of Sections	
start	program1,program2,function,global,local,f_local,stack_area
output	EXAMPLE ; Output "EXAMPLE.ABS"
print	EXAMPLE ; Print "EXAMPLE.MAP"
exit	

図 A.3 リンケージリスト"example.map"の内容 (入力情報)

H SERIES LINKAGE EDITOR Ver. 5.3						PAGE : 1
*** LINKAGE EDITOR LINK MAP LIST ***						
SECTION	NAME	START	-	END	LENGTH UNIT NAME	MODULE NAME
ATTRIBUTE : CODE NOSHR						
program1		H'00000000	-	H'00000349	H'0000034A main	PROGRAM1
		H'0000034A	-	H'00000467	H'0000011E init	PROGRAM1
		H'00000468	-	H'0000055D	H'000000F6 cmdanl	PROGRAM1
		H'0000055E	-	H'000007E7	H'0000028A cmdprc	PROGRAM1
		H'000007E8	-	H'0000091F	H'00000138 term	PROGRAM1
* TOTAL ADDRESS *		H'00000000	-	H'0000091F	H'00000920	
ATTRIBUTE : CODE NOSHR						
program2		H'00000920	-	H'00000B1F	H'00000200 keyin	input_keyboard
		H'00000B20	-	H'00000C47	H'00000128 file	file_io
		H'00000C48	-	H'00000D49	H'00000102 printer	output_printer
		H'00000D4A	-	H'00000E61	H'00000118 display	display_console
		H'00000E62	-	H'00001127	H'000002C6 commu	communication
* TOTAL ADDRESS *		H'00000920	-	H'00001127	H'00000808	
ATTRIBUTE : CODE NOSHR						
function		H'00001128	-	H'00001143	H'0000001C comp	PROGRAM1
		H'00001144	-	H'00001237	H'000000F4 expr	PROGRAM1
		H'00001238	-	H'0000128B	H'00000054 mvdata	PROGRAM1
		H'0000128C	-	H'000012BB	H'00000030 upshft	PROGRAM1
		H'000012BC	-	H'00001343	H'00000088 lmargin	left_margin
		H'00001344	-	H'00001373	H'00000030 number	numbering_items
		H'00001374	-	H'000013F3	H'00000080 rmargin	right_margin

図 A.3 リンケージリスト"example.map"の内容 (リンケージマップリスト)

H SERIES LINKAGE EDITOR Ver. 5.3						PAGE : 2
*** LINKAGE EDITOR LINK MAP LIST ***						
SECTION	NAME	START	-	END	LENGTH UNIT NAME	MODULE NAME
ATTRIBUTE : CODE NOSHR						
function		H'000013F4	-	H'0000140B	H'00000018 sum	sum_items
		H'0000140C	-	H'000014C7	H'000000BC zerosprs	zero_suppres
		H'000014C8	-	H'00001533	H'0000006C ascbin	ascii_to_binary
		H'00001534	-	H'00001573	H'00000040 binasc	binary_to_ascii
		H'00001574	-	H'0000163F	H'000000CC cnvbcd	convert_to_bcd
		H'00001640	-	H'00001647	H'00000008 dos	interface_of_dos
		H'00001648	-	H'00001657	H'00000010 portio	interface_of_port
* TOTAL ADDRESS *		H'00001128	-	H'00001657	H'00000530	
ATTRIBUTE : DATA NOSHR						
global		H'00001658	-	H'00002C27	H'000015D0 table	PROGRAM1
* TOTAL ADDRESS *		H'00001658	-	H'00002C27	H'000015D0	
ATTRIBUTE : DATA NOSHR						
local		H'00002C28	-	H'00004A47	H'00001E20 main	PROGRAM1
		H'00004A48	-	H'00004A67	H'00000020 init	PROGRAM1
		H'00004A68	-	H'000068A7	H'00001E40 cmdanl	PROGRAM1
		H'000068A8	-	H'00024EE7	H'0001E640 cmdndprc	PROGRAM1
		H'00024EE8	-	H'00024F07	H'00000020 term	PROGRAM1
		H'00024F08	-	H'00025127	H'00000220 keyin	input_keyboard
		H'00025128	-	H'00025307	H'000001E0 file	file_io
		H'00025308	-	H'0002544B	H'00000144 printer	output_printer
		H'0002544C	-	H'0002554F	H'00000104 display	display_console

図 A.3 リンケージリスト"example.map"の内容 (リンケージマップリスト)

H SERIES LINKAGE EDITOR Ver. 5.3						PAGE :	3
*** LINKAGE EDITOR LINK MAP LIST ***							
SECTION	NAME	START	-	END	LENGTH UNIT NAME	MODULE NAME	
ATTRIBUTE : DATA NOSHR							
local		H'00025550	-	H'00025713	H'000001C4 commu	communication	
* TOTAL ADDRESS *		H'00002C28	-	H'00025713	H'00022AEC		
ATTRIBUTE : DATA NOSHR							
f_local		H'00025714	-	H'0002571F	H'0000000C comp	PROGRAM1	
		H'00025720	-	H'0002582F	H'00000110 expr	PROGRAM1	
		H'00025830	-	H'00025833	H'00000004 upshft	PROGRAM1	
		H'00025834	-	H'00025843	H'00000010 lmargin	left_margin	
		H'00025844	-	H'00025847	H'00000004 number	numbering_items	
		H'00025848	-	H'00025857	H'00000010 rmargin	right_margin	
		H'00025858	-	H'0002587B	H'00000024 zerosprs	zero_suppress	
		H'0002587C	-	H'00025883	H'00000008 ascbin	ascii_to_binary	
		H'00025884	-	H'00025887	H'00000004 binasc	binary_to_ascii	
		H'00025888	-	H'000258CF	H'00000048 cnvbcd	convert_to_bcd	
* TOTAL ADDRESS *		H'00025714	-	H'000258CF	H'000001BC		
ATTRIBUTE : STACK NOSHR							
stack_area		H'000258D0	-	H'002078CF	H'001E2000 table	PROGRAM1	
* TOTAL ADDRESS *		H'000258D0	-	H'002078CF	H'001E2000		

図 A.3 リンケージリスト"example.map"の内容 (リンケージマップリスト)

H SERIES LINKAGE EDITOR Ver. 5.3		PAGE : 1	
*** LINKAGE EDITOR EXTERNALLY DEFINED SYMBOLS LIST ***			
SYMBOL	NAME	ADDR	TYPE
ascbin		H'000014C8	DAT
binasc		H'00001534	DAT
cmdanl		H'00000468	DAT
cmdprc		H'0000055E	DAT
cmdtbl		H'00001720	DAT
cnvbcd		H'00001574	DAT
commu		H'00000E62	DAT
comp		H'00001128	DAT
display		H'00000D4A	DAT
dos		H'00001640	DAT
expr		H'00001144	DAT
file		H'00000B20	DAT
fltbl		H'00001A20	DAT
header		H'00001658	DAT
init		H'0000034A	DAT
keybuf		H'00001820	DAT
keyin		H'00000920	DAT
lmargin		H'000012BC	DAT
main		H'00000000	DAT
mvdata		H'00001238	DAT
number		H'00001344	DAT
portio		H'00001648	DAT
prbuf		H'00002B20	DAT
printer		H'00000C48	DAT
recbuf		H'00002A20	DAT
rmargin		H'00001374	DAT
stack_area		H'000258D0	DAT
sum		H'000013F4	DAT
term		H'000007E8	DAT
upshft		H'0000128C	DAT
zerosprs		H'0000140C	DAT

図 A.3 リンケージリスト"example.map"の内容 (外部定義シンボルリスト)

付録 B. ファイル名の構成

ファイル名は以下に示す構成となります。

<u>パス名</u>	<u>主ファイル名</u>	<u>ファイル形式</u>
------------	---------------	---------------

(1)
(2)
(3)

(1) パス名

ファイルが存在するディレクトリのパス名を MS-DOS 版では円記号 (¥) で、UNIX 版ではスラッシュ (/) で区切って指定します。省略された場合は、現在使用しているディレクトリとみなされます。

(2) 主ファイル名

ファイルの名前を指定します。

(3) ファイル形式

ファイルの種類を表す名前です。主ファイル名との間はピリオド(.)で区切ります。

(例1) ¥user ¥tool¥ prog .typ (MS-DOS版)

↑
↑
↑
ファイル形式
主ファイル名
パス名

(例2) /user /tool/ prog .typ (UNIX版)

↑ ↑ ↑
ファイル形式
主ファイル名
パス名

本リンクージエディタでは、ファイル名に関する一般規則は OS（オペレーティングシステム）に準じています。

【注】 入力ファイル名と出力ファイル名に同じファイル名を指定すると入力ファイル名の内容が壊れます。入出力ファイル名が重複しないようご注意ください。

ライブラリアン編

1. 概要

通常プログラムを開発する場合、ソースプログラムを機能単位などによりモジュール分割し、モジュールごとにコンパイル、アセンブルを行います。そして、コンパイル、アセンブルした結果をリンケージエディタによって1つに結合し、目的とするプログラムを作成します。

Hシリーズ ライブラリアン（以降ライブラリアンと略します）は、Cコンパイラ、アセンブラが出力した複数のオブジェクトモジュール、またはリンケージエディタが出力した複数のリロケータブルロードモジュールをまとめて、ライブラリファイルを作成します。ライブラリアンを使用すると次のような利点があります。

（1）モジュールの管理が容易

プログラムを構成する複数のオブジェクトモジュール、またはリロケータブルロードモジュール（以降モジュールと略します）を、そのプログラム専用のライブラリファイルに登録することにより、一括して管理することができます。また、汎用ライブラリファイルを作成し、汎用性のあるモジュールをまとめて登録しておけば、他のプログラムへの流用が容易に行えます。

ライブラリファイルは、モジュール単位で追加、削除、置換などの編集を行うことができ、各モジュールを最新の状態で保管することができます。

（2）リンケージ処理の効率向上

リンケージエディタには、ライブラリファイルを検出し、未解決の外部参照シンボルを定義しているモジュールを取り出し、結合する機能があります。このため、ライブラリファイルを使用することにより、リンケージ処理を効率良く行うことができます。

2. ライブラリアンの機能

第2章 目次

2.1	ライブラリファイルの作成.....	159
2.2	既存ライブラリファイルの編集.....	160
2.3	ライブラリファイル内のモジュール抽出.....	162
2.4	ライブラリファイルの内容表示.....	163

2.1 ライブラリファイルの作成

ライブラリファイルを新しく作成し、C コンパイラ、アセンブラが出力したオブジェクトモジュールまたはリンカージエディタが出力したリロケートブルロードモジュールを登録することができます。

ライブラリファイルの作成例を図 2.1 に示します。

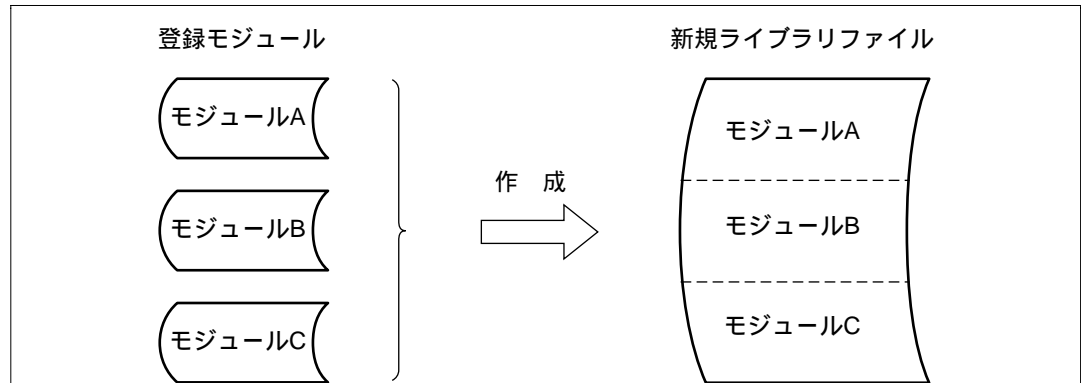


図 2.1 ライブラリファイルの作成例

2.2 既存ライブラリファイルの編集

すでに作成されている既存ライブラリファイルに対して、モジュールの追加、削除、および置換を行うことができます。

(1) モジュールの追加

既存ライブラリファイルにモジュールを追加することができます。

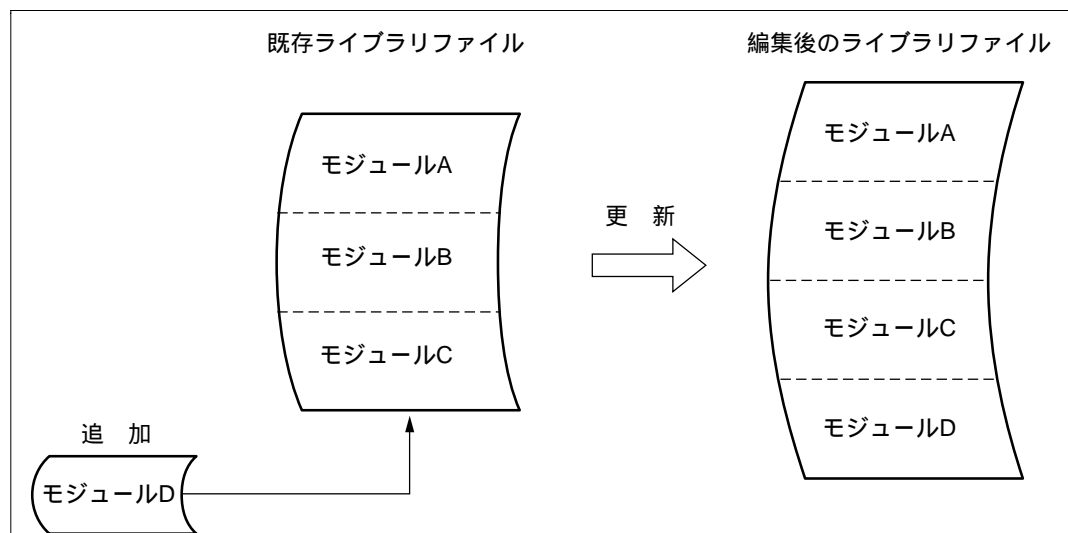


図 2.2 モジュールの追加

(2) モジュールの削除

既存ライブラリファイルから不要なモジュールを削除することができます。

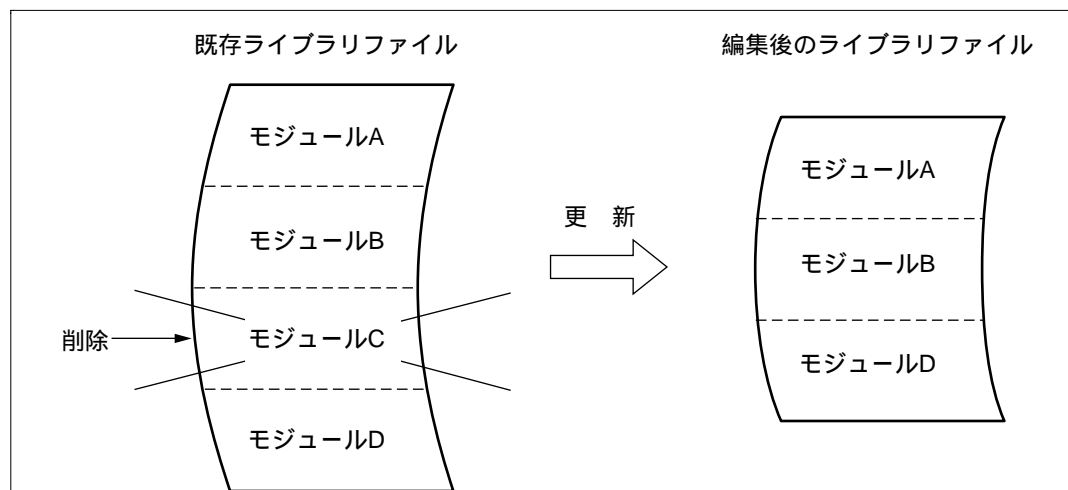


図 2.3 モジュールの削除

(3) モジュールの置換

既存ライブラリファイル内のモジュールを新しいモジュールと置き換えることができます。

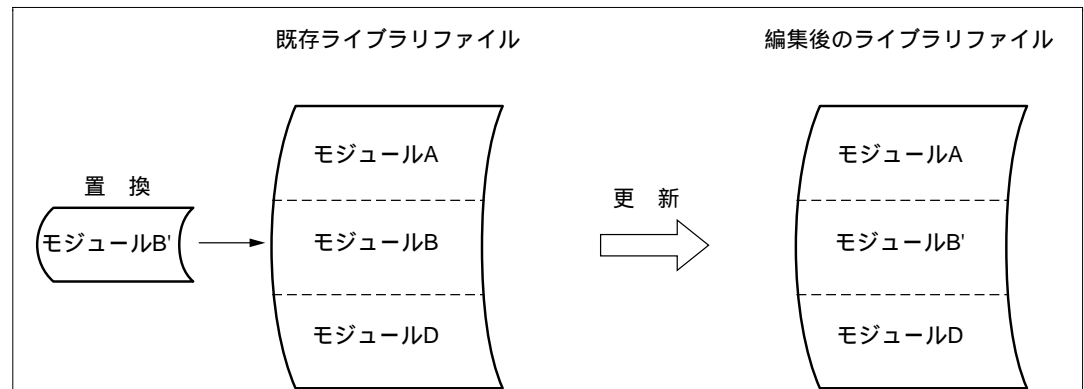


図 2.4 モジュールの置換

2.3 ライブラリファイル内のモジュール抽出

既存ライブラリファイルからモジュールを抽出し、別に新しいライブラリファイルを作ることができます。

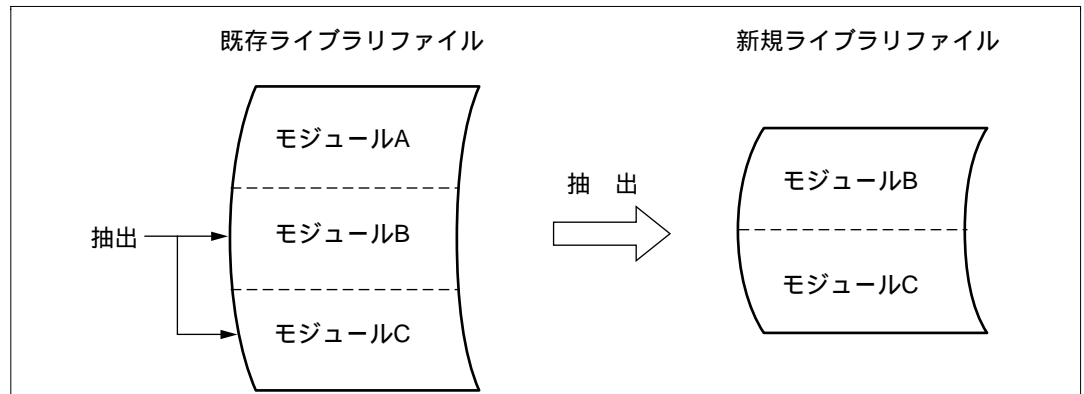


図 2.5 モジュールの抽出

2.4 ライブラリファイルの内容表示

ライブラリファイルのモジュール情報および外部定義シンボル情報などをライブラリアンリストとして、標準出力またはリストファイルへ出力します。ライブラリアンリストにより、ライブラリファイルの作成 / 更新日付、モジュールの登録日付、および各モジュールで定義されている外部定義シンボル名などを知ることができます。

表示内容の詳細については、「6.2 ライブラリアンリスト」を参照してください。

3. ライブラリアンの実行

第3章 目次

3.1	コマンドラインのフォーマット	169
3.2	コマンドライン指定による実行	170
3.3	サブコマンド指定による実行	171
	3.3.1 会話形式による実行	171
	3.3.2 サブコマンドファイルによる実行	172
3.4	ライブラリアンの終了	173

ライブラリアンを実行するために、まずライブラリアンを起動するコマンドラインを指定します。コマンドラインには、編集するライブラリファイル名やライブラリアンに様々な指示を与えるオプションを指定します。このコマンドラインの指定だけでも十分ライブラリアンは実行できますが、ライブラリアンに対して多様な指示を与えたい場合のために、さらにサブコマンドによる実行もできます。

(1) コマンドライン指定による実行

コマンドラインでのライブラリファイルとオプションの指定だけでライブラリアンを実行する方法です。ライブラリ編集処理の手順が比較的単純な場合に利用します。

(2) サブコマンド指定による実行

コマンドラインだけでなく、サブコマンドで入出力ファイルやライブラリアンに対する実行制御を指定することにより、ライブラリアンを実行する方法です。指定するファイルやモジュールの数が多かったり、複数のライブラリファイルの編集をまとめて行う場合に利用します。サブコマンドによる実行には、会話形式とサブコマンドファイルによる2つの方法があります。詳細は「3.3 サブコマンド指定による実行」を参照してください。

なお、コマンドラインおよびサブコマンドで指定するファイル名は以下に示す構成となります。

パス名	主ファイル名	ファイル形式
(a)	(b)	(c)

(a) パス名

ファイルが存在するディレクトリのパス名を、オペレーティングシステムがUNIXの場合はスラッシュ(/)、MS-DOSの場合は円マーク(¥)で区切って指定します。省略された場合は、現在使用しているディレクトリとします。

(b) 主ファイル名

ファイルの名前を指定します。

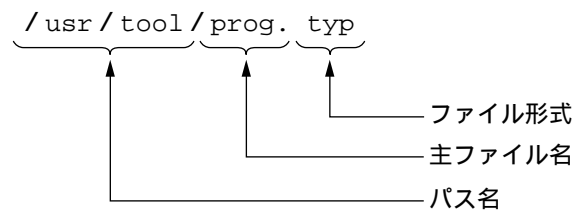
(c) ファイル形式

ファイルの種類を表す名前です。主ファイル名との間はピリオド(.)で区切ります。省略された場合、暗黙のファイル形式を仮定します。

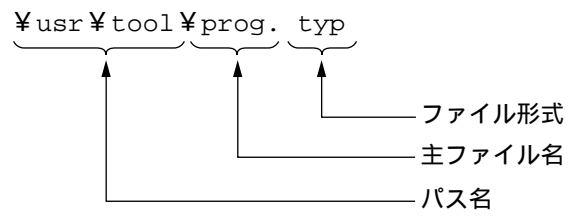
本ライブラリアンでは、ファイル名に関する一般規則はオペレーティングシステムに準じています。

【注】 オペレーティングシステムがUNIXの場合コマンドラインについては、オペレーティングシステムのシェル(コマンドインタプリタ)により、本ライブラリアンに制御が渡る前にチェックが行われます。したがって、オペレーティングシステムがコマンドラインで指定可能としている文字についてご確認ください。

(例) UNIX の場合



(例) MS-DOS の場合



3.1 コマンドラインのフォーマット

ライブラリアンのコマンドラインのフォーマットは、次のとおりです。

```
lbr[  [<ライブラリファイル名>]
      [[  ]-<オプション名>[[  ]-<オプション名>...]]] (RET)
```

(1) 起動コマンド

ライブラリアンの起動コマンドとして"lbr"を入力します。

(2) ライブラリファイル名

コマンドライン指定で既存ライブラリファイルの編集またはモジュールの抽出を行う場合、対象となるライブラリファイル名を指定します。

(3) オプション名

オプション名は、先頭にハイフン（－）を付けて指定してください。また、オプション名とライブラリ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。オプション名の詳細については、「4. ライブラリアンのオプション / サブコマンド」を参照してください。ライブラリアンはオプションの指定順にライブラリファイルの編集を行います。

(4) 実行形式の指定

コマンドラインの指定内容により、ライブラリファイルをコマンドライン指定のみ実行するか、サブコマンド指定によって実行するかを決定します。表 3.1 にコマンドラインの指定方法を示します。

表 3.1 コマンドラインの指定方法

オプションの指定 ライブラリファイル 名の指定	オプション指定なし	SUBCOMMAND ^{*1} オプションを指定	CREATE ^{*1} オプションを指定	SUBCOMMAND, CREATEオプション 以外を指定
ライブラリファイル名 指 定 あ り	^{*2}			コマンドライン 指 定 に よ る 実 行
ライブラリファイル名 指 定 な し	サブコマンド 指 定 に よ る 実 行	サブコマンド 指 定 に よ る 実 行	コマンドライン 指 定 に よ る 実 行	

【注】 *1 SUBCOMMANDおよびCREATE オプションについては、「4.ライブラリアンのオプション / サブコマンド」を参照してください。

*2 斜線部のオプションとライブラリファイル名指定の組み合わせは、ライブラリアンでは認められない指定形式です。この場合エラーとなり、ライブラリアンは実行されません。

3.2 コマンドライン指定による実行

コマンドラインに指定された情報だけでライブラリアンを実行する方法です。ライブラリアンへの編集処理手順等の指示はオプションで指定します。編集処理が単純な場合は、このコマンドラインの指定だけでライブラリの作成、更新が行えます。以下に、コマンドラインによる実行の例を示します。

(例1)

```
lbr      -CREATE=syslib.lib-ADD=obj00.obj,prg.lib (RET)
          (1)                (2)
```

(1) ライブラリファイル"syslib.lib"を新規作成します。

(2) "syslib.lib"に、オブジェクトモジュールファイル"obj00.obj"内のモジュールとライブラリファイル"prg.lib"に含まれるすべてのモジュールを登録します。

CREATE オプションを指定しても、モジュールの登録(ADD オプションの指定)を行わないとライブラリファイルは作成しません。

(例2)

```
lbr      syslib.lib-ADD=obj00.obj-DELETE=mod1 (RET)
          (1)                (2)                (3)
```

(1) ライブラリファイル"syslib.lib"を編集対象とします。

(2) "syslib.lib"にオブジェクトモジュールファイル"obj00.obj"内のモジュールを追加します。

(3) "syslib.lib"にすでに登録されているモジュール"mod1"を削除します。

3.3 サブコマンド指定による実行

コマンドラインの指定だけでは、入力できる文字数に制限があるため、指定するファイルやモジュールが多い場合に、指定しきれないことがあります。このような場合に、サブコマンド指定による実行を利用します。サブコマンド指定による実行には、サブコマンドを1つずつ標準入力から入力していく会話形式による方法と、あらかじめサブコマンド群をサブコマンドファイルとして作成しておき、この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による方法があります。

3.3.1 会話形式による実行

コマンドラインでライブラリファイルの指定がなく、オプションの指定もない場合に会話形式の実行になります。会話形式で実行すると、サブコマンド入力待ちのプロンプト": "を表示するので、ライブラリアンの処理に必要なサブコマンドを入力します。入力するサブコマンドの数が比較的少ない場合や、ライブラリアンリストを参照しながらサブコマンドを入力する場合に利用します。

以下に、会話形式による実行の例を示します。例中のサブコマンドの詳細機能については、「4. ライブラリアンのオプション / サブコマンド」を参照してください。

(例)

```

    lbr (RET)                                     ... (1)
: CREATE    prg.lib (RET)                         ... (2)
: ADD      main.obj (RET)                         ... (3)
: ADD      send.obj, receive.obj, exchange.obj (RET) ... (4)
: ADD      account.obj (RET)                     ... (5)
: LIST     (S) (RET)                             ... (6)
: EXIT (RET)                                     ... (7)

```

(1) ライブラリアンを会話形式で起動します。

(2) ライブラリファイル"prg.lib"を新規作成します。

(3) "main.obj"内のモジュールを"prg.lib"に登録します。

(4) "send.obj"、"receive.obj"、および"exchange.obj"内のモジュールを"prg.lib"に登録します。

(5) "account.obj"内のモジュールを"prg.lib"に登録します。

(6) シンボル情報を含むライブラリアンリストを標準出力へ出力します。

(7) ライブラリアン処理を終了します。

3.3.2 サブコマンドファイルによる実行

あらかじめライブラリアンの処理に必要なサブコマンドをサブコマンドファイルに作成しておきコマンドラインで SUBCOMMAND オプションのパラメータとしてこのサブコマンドファイルを指定することにより実行する方法です。この方法を利用することで、入力するサブコマンドの数が多い場合や、同じ編集処理を繰り返す場合に、そのたびに標準入力から入力する手間を省くことができます。サブコマンドファイルは、エディタを利用して作成します。以下に、サブコマンドファイルによる実行の例を示します。例中のサブコマンドの詳細機能については、「4. ライブラリアンのオプション / サブコマンド」を参照してください。

(例)

```
lbr -SUBCOMMAND=prglib.sub (RET) ... (1)
```

サブコマンドファイル"prglib.sub"の内容

CREATE	function.lib	... (2)
ADD	sin.obj,cos.obj,tan.obj	... (3)
ADD	asin.obj,acos.obj,atan.obj	... (4)
ADD	hsin.obj,hcos.obj,htan.obj	... (5)
ADD	log.obj,log10.obj	... (6)
EXIT		... (7)

- (1) ライブラリアンを起動し、サブコマンドファイル"prglib.sub"からサブコマンドを入力します。
- (2) ライブラリファイル"function.lib"を新規作成します。
- (3) オブジェクトモジュールファイル"sin.obj"、"cos.obj"、および"tan.obj"内のモジュールを"function.lib"に登録します。
- (4) オブジェクトモジュールファイル"asin.obj"、"acos.obj"、および"atan.obj"内のモジュールを"function.lib"に登録します。
- (5) オブジェクトモジュールファイル"hsin.obj"、"hcos.obj"、および"htan.obj"内のモジュールを"function.lib"に登録します。
- (6) オブジェクトモジュールファイル"log.obj"および"log10.obj"内のモジュールを"function.lib"に登録します。
- (7) ライブラリアン処理を終了します。

3.4 ライブラリアンの終了

ライブラリアンが終了するとき、エラーのレベルをリターンコードとしてシステムに返します。リターンコードを返すことにより、コマンドファイルの実行を制御することができます。

リターンコードはエラーのレベルにより以下の値となります。

オペレーティングシステム	UNIX	MS-DOS
正常終了	0	0
ウォーニング発生	0	0
エラー発生	1	2
フェイタルエラー（致命的なエラー）発生	1	4

4. ライブラリアンのオプション / サブコマンド

第4章 目次

4.1	オプション / サブコマンドのフォーマット	178
4.2	オプション / サブコマンドの種類	181
4.3	ファイル制御	185
4.3.1	LIBRARY - 編集対象のライブラリファイルの指定	185
4.3.2	OUTPUT - 出力ライブラリファイルの指定	186
4.3.3	DIRECTORY - ディレクトリ名の置き換えの指定	188
4.4	実行制御	189
4.4.1	SUBCOMMAND - サブコマンドファイルの指定	189
4.4.2	CREATE - ライブラリファイルの生成	190
4.4.3	ADD - モジュールの登録 / 追加	192
4.4.4	REPLACE - モジュールの置換	195
4.4.5	DELETE - モジュールの削除	198
4.4.6	EXTRACT - モジュールの抽出	199
4.4.7	RENAME - セクション名の変更	200
4.4.8	END - サブコマンド入力の終了指定	201
4.4.9	EXIT - ライブラリアンの終了指定	202
4.4.10	ABORT - ライブラリアンの強制終了指定	203
4.5	内容表示	204
4.5.1	LIST - ライブラリファイルの内容表示	204
4.5.2	SLIST - ライブラリファイルのセクション内容の表示	206

オプションおよびサブコマンドは、ライブラリアンに対して編集操作の指示を与えるためのものです。オプションおよびサブコマンドの機能は、ファイル制御、実行命令、内容表示の3種類に大別されますが、これらの機能を単独あるいは組み合わせで利用することによって、ライブラリファイルの作成、編集ができます。

オプションとサブコマンドは同一名称でかつ同等の機能を持っていますが、指定時のフォーマットは異なります。また、オプションでしか指定できないものやサブコマンドでしか指定できないものがありますので、「4.1 オプション / サブコマンドのフォーマット」および「4.2 オプション / サブコマンドの種類」をお読みください。以下に、オプション / サブコマンドの機能概要を説明します。

(1) ファイル制御機能

ファイル制御機能は、編集対象となるライブラリファイル名、および抽出したモジュールを出力するライブラリファイル名を指示する機能です。

(2) 実行制御機能

実行制御は、ライブラリアンに対して編集操作または処理の終了を指示する機能です。サブコマンドをサブコマンドファイルから入力したり、新規ライブラリファイルの作成、ライブラリファイルの更新などを行う場合に利用します。

(3) 内容表示

内容表示機能は、ライブラリファイルに登録されているモジュール名、外部定義シンボル名等の情報を表示する場合に利用します。

4.1 オプション / サブコマンドのフォーマット

(1) オプション / サブコマンドの構成

(a) 名称部

名称部には、オプション名またはサブコマンド名を指定します。各名称については、「4.2 オプション / サブコマンドの種類」を参照してください。

(b) パラメータ部

パラメータ部には、オプションまたはサブコマンドの操作対象となるファイル名^{*1}、モジュール名^{*2}等を指定します。オプションまたはサブコマンドの種類によって必要なものと不要なもの、指定方法が異なるものがありますので、「4.3 ファイル制御」から「4.5 内容表示」の各節を参照してください。

名称部とパラメータ部の区切り記号は、オプションとサブコマンドで異なります。オプションの場合はイコール (=) で区切り、サブコマンドの場合は1つ以上のスペースまたはタブで区切ります。

オプションのフォーマット

名称部	=	パラメータ部
-----	---	--------

サブコマンドのフォーマット

名称部	パラメータ部
-----	--------

(例)

-OUTPUT=lbfg オプションの場合

OUTPUT lbfg サブコマンドの場合

この例では、"OUTPUT"が名称部で"lbfg"がパラメータ部です。

【注】^{*1} ファイル名はパス名、主ファイル名およびファイル形式の3つの部分からなります。ファイル形式を省略した場合、各ファイルに対して次のようなファイル形式を仮定して処理します。

ライブラリファイル	" .lib "
オブジェクトモジュールファイル	}
リロケータブルロードモジュールファイル	
サブコマンドファイル	" .sub "
リストファイル	" .lst "

^{*2} モジュール名は、オブジェクトモジュールまたはリロケータブルロードモジュールの中で定義されているモジュールの名前です。

モジュール名は大文字と小文字を区別して扱います。例えば、次のようなモジュール名は異なる名前として処理します。

(例)	modul1	MODUL1
	abcde	Abcde

(2) サブコマンドの継続指定

サブコマンドを1行に指定しきれないような場合(500文字/行までですが、OSに依存する場合があります)は、継続指定を利用します。継続指定は行の最後にアンパサンド(&)を指定することにより示しますが、必ず各パラメータの区切りで指定しなければいけません。パラメータの途中で指定した場合は、継続指定とみなしません。また、アンパサンドの後ろに文字(スペース、タブも含まれます)を指定した場合は、エラーとなり、継続指定にはなりません。なお、継続指定に対する会話形式での入力待ちのプロンプトは"- "になります。

(例)

```
:ADD    obj00.lib(mod0,mod1),&(RET)
-obj01,obj02 (RET)
:ADD    obj00.lib(mod0,mod1),ob&(RET)
```

継続指定を示します。

パラメータの途中で継続指定しているためエラーとなります。

サブコマンドファイルの中でサブコマンド行を継続する場合も同様です。継続指定をした行の次の行が継続行となります。

(例)

サブコマンドファイル

<pre> : : DELETE SUB1,SUB2,&(RET) sub3 (RET) : :</pre>	<p>← 継続指定</p> <p>← 継続行</p>
---	----------------------------

(3) サブコマンドのコメント指定

サブコマンドファイル中に注釈等をいれたい場合は、コメント指定を利用します。サブコマンド行中にセミコロン(;)を指定することにより、それ以降をコメントとして扱うことができます。

ただし、サブコマンドの名称部またはパラメータ部とセミコロンとの間には、1つ以上のスペースまたはタブが必要です。

また、サブコマンド行の先頭にセミコロンを指定することにより、その行全体をコメントとして扱うことができます。

(例)

```
;EXAMPLE OF LIBRARIAN SUBCOMMAND
```

```
... 行全体がコメントです。
```

```
LIBRARY syslib ;INDICATES LIBRARY FILE
```

```
... "INDICATES LIBRARY FILE"がコメントです。
```

```
ADD module.obj;abc
```

```
... "module.obj;abc"が1つのパラメータになり、"abc"はコメントとして扱いません。
```

4.2 オプション / サブコマンドの種類

10 種類のオプションと 15 種類のサブコマンドがあります。表 4.1 にオプション / サブコマンド一覧を示します。

表 4.1 オプション / サブコマンド一覧表

項番	分類	オプション/サブコマンド名 ^{*1}	機能	オプション ^{*2}	サブコマンド	節番号
1	ファイル 制御	<u>L</u> IBRARY	編集対象のライブラリファイル の指定	×		4.3.1
		<u>O</u> UTPUT	出力ライブラリファイルの指定			4.3.2
		<u>D</u> IRECTORY	ディレクトリ名の置き換えの指 定	×		4.3.3
2	実行 制御	<u>S</u> UBCOMMAND	サブコマンドファイルの指定			4.4.1
		<u>C</u> REATE	ライブラリファイルの生成			4.4.2
		<u>A</u> DD	モジュールの登録 / 追加			4.4.3
		<u>R</u> EPLACE	モジュールの置換			4.4.4
		<u>D</u> ELETE	モジュールの削除			4.4.5
		<u>E</u> XTRACT	モジュールの抽出			4.4.6
		<u>R</u> ENAME	セクション名の変更			4.4.7
		<u>E</u> ND	サブコマンドの入力の終了指定	×		4.4.8
		<u>E</u> XIT	ライブラリアンの終了指定	×		4.4.9
		<u>A</u> BORT	ライブラリアンの強制終了指定	×		4.4.10
3	内容 表示	<u>L</u> IST	ライブラリファイルの内容表示			4.5.1
		<u>S</u> LIST	ライブラリファイルのセクショ ン内容表示			4.5.2

【注】 *1 アンダーラインの部分は最も短い短縮形を示します。

*2 印は使用できることを、×印は使用できないことを示します。

(1) 名称部の短縮指定

オプションおよびサブコマンドの名称部は、その名称であることが識別できる部分まで短縮して指定することができます。例として"EXTRACT"の短縮指定を示します。

EEXIT および END と区別できないため、エラーとなります。

EXEXIT と区別できないため、エラーとなります。

EXTEXTRACT と識別します。

EXTRAEXTRACT と識別します。

EXTRACTEXTRACT と識別します。

EXTRACTSEXTRACTS はないため、エラーとなります。

(2) オプション / サブコマンドの相互関係

すでに指定されているオプション / サブコマンドに対し、競合する機能をもつオプション / サブコマンドを後から指定することはできません。

表 4.2 に各オプション / サブコマンド間の相互関係を示します。

表 4.2 オプション / サブコマンドの相互関係

後続指定の オプション / サブコマンド 指定 済みの オプション / サブコマンド	SUB COM MAND	LIB RARY	CRE ATE	ADD	RE PL ACE	DE LETE	EX TRACT	RE NAME	OUT PUT	DIRECTORY	LIST	SLIST	END	EXIT	ABORT
SUBCOMMAND	x														
LIBRARY		x	x												
CREATE		x	x				x	x	x						
ADD		x	x				x		x						
REPLACE		x	x				x	x	x						
DELETE		x	x				x	x	x						
EXTRACT		x	x	x	x	x		x							
RENAME		x	x				x		x						
OUTPUT		x	x	x	x	x		x	x						
DIRECTORY															
LIST		x	x												
SLIST		x	x												
END				x	x	x	x	x	x		x	x	x	x	
EXIT	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
ABORT	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

: 後続指定可

x : 指定済みのオプション / サブコマンドと競合するため指定不可

(例)

```
lbr (RET)
: LIBRARY      funclib.lib (RET)
: CREATE      newlib.lib (RET) ← LIBRARYサブコマンドの後にCREATEサブコマンドは指定できません。エラーとなり、CREATEサブコマンドを無視します。
```



```
: END (RET)
: LIST (RET) ← ENDサブコマンドの後にLISTサブコマンドは指定できません。エラーとなります。
: EXIT (RET)
```

4. ライブラリアンのオプション / サブコマンド

オプション / サブコマンドの説明に使用する表の内容を以下に示します。

(1) フォーマット

名 称		オプション	サブコマンド
パラメータ			

オプション / サブコマンドの名称およびパラメータの指定形式です。名称の下線部は最も短い短縮指定を示します。

(2) 機 能

オプション / サブコマンドの機能概要です。

(3) 説 明

オプション / サブコマンドの詳細な機能や制限事項です。

(4) 指定例

オプション / サブコマンドの指定例です。

4.3 ファイル制御

4.3.1 LIBRARY - 編集対象のライブラリファイルの指定

(1) フォーマット

名 称	LIBRARY	オプション	サブコマンド
		な し	あ り
パラメータ	<ライブラリファイル名>		

(2) 機 能

編集対象の既存ライブラリファイルを指定します。

(3) 説 明

- (1) 既存ライブラリファイルを更新する場合、または既存ライブラリファイルからモジュールを抽出する場合に編集操作の先頭で指定します。
- (2) 本ライブラリアンで作成したライブラリファイル以外のファイルを指定することはできません。
- (3) ライブラリファイル名にファイル形式の指定がない場合、".lib"というファイル形式を仮定して処理します。
- (4) 新規ライブラリファイルの生成を指定する CREATE サブコマンドと一緒に使用することはできません。
- (5) 既存ライブラリファイルを編集した結果、モジュール登録数がゼロになった場合にはライブラリファイルの更新は行いません。
- (6) 変更後のライブラリファイルのアクセス権は、更新前の状態を保持せずに新規にファイルを作成したときと同じ状態になっていますのでご注意ください。

(4) 指定例

```
LIBRARY    syslib
```

ライブラリファイル"syslib.lib"を編集対象とします。

4.3.2 OUTPUT - 出力ライブラリファイルの指定

(1) フォーマット

名 称	OUTPUT		オプション	サブコマンド
			あ り	あ り
パラメータ	オプション	UNIX	<ライブラリファイル名>	
		MS-DOS	<ライブラリファイル名>	$\left\{ \begin{array}{l} (S) \\ (U) \end{array} \right\}$
	サブコマンド	<ライブラリファイル名> $\left\{ \begin{array}{l} (S) \\ (U) \end{array} \right\}$		

(2) 機 能

抽出したモジュールを出力するライブラリファイルを指定します。

(3) 説 明

(1) 既存ライブラリファイルからモジュールを抽出する場合、必ず OUTPUT オプション / サブコマンドを指定してください。

(2) ライブラリファイルは新規のファイルを指定してください。

ライブラリファイル名にファイル形式の指定がない場合は、".lib"というファイル形式を仮定します。

(3) 出力するライブラリファイルの属性を"(S)"または"(U)"で指定します。
省略した場合は"(U)"を仮定します。

(S)・・・システムライブラリ

(U)・・・ユーザライブラリ

この属性は、リンケージエディタによるライブラリファイルのサーチの優先順位を決めるものです。ユーザライブラリの方が優先順位が高くなります。ただし、オペレーティングシステムが UNIX の場合、オプションでの"(S)"と"(U)"の指定はできません。

(4) OUTPUT は、モジュール抽出を指定する EXTRACT オプション / サブコマンド指定の前後どちらで指定してもかまいません。

(5) CREATE、ADD、DELETE および REPLACE オプション / サブコマンドと一緒に使用することはできません。

(6) 抽出モジュール数がゼロの場合、OUTPUT オプション / サブコマンドで指定したライブラリファイルは生成しません。

(4) 指定例

`-OUTPUT=prog86`

EXTRACT オプションで抽出したモジュールをユーザライブラリとして "prog86.lib" というファイルに出力します。

`OUTPUT clib.o(S)`

EXTRACT サブコマンドで抽出したモジュールをシステムライブラリとして "clib.o" というファイルに出力します。

4.3.3 DIRECTORY - ディレクトリ名の置き換えの指定

(1) フォーマット

名 称	DIRECTORY	オプション	サブコマンド
		な し	あ り
パラメータ	<シンボル名> (<ディレクトリ名>)		

(2) 機 能

ディレクトリ名をシンボル名に登録します。

本機能により、冗長なディレクトリ名を単純なシンボル名で指定することができます。

(3) 説 明

(1) ディレクトリ名の登録

DIRECTORY サブコマンドでディレクトリ名をシンボル名に登録します。

DIRECTORY <シンボル名> (<ディレクトリ名>)

(2) ディレクトリ名の参照

登録したディレクトリ名の参照は、シンボル名を\$と/で囲みます (MS-DOS 版は\$と¥)。シンボル名が未登録の場合は、置き換えを行いません。

\$シンボル名/ - - - > ディレクトリ名/に置き換えます

(3) ディレクトリ名は、16個まで登録できます。

(4) 指定例

```
DIRECTORY     symbol(dir1/dir2)
```

```
ADD     $symbol/file1.obj
```

ディレクトリ名 dir1/dir2 をシンボル名 symbol として登録します。

\$symbol/を dir1/dir2/に置き換え、ファイル名を dir1/dir2/file1.obj とします。

4.4 実行制御

4.4.1 SUBCOMMAND - サブコマンドファイルの指定

(1) フォーマット

名 称	SUBCOMMAND	オプション	サブコマンド
		あ り	あ り
パラメータ	<サブコマンドファイル名>		

(2) 機 能

指定したファイルからサブコマンドを入力します。

(3) 説 明

(1) 指定したサブコマンドファイルからサブコマンドを1つずつ取り込み処理を行います。

(2) EXIT サブコマンドの指定がない場合は、入力コマンド待ちとなります。

(3) サブコマンドファイル名にファイル形式の指定がない場合、".sub"というファイル形式を仮定します。

(4) SUBCOMMAND と他のオプションと一緒に使用すると、SUBCOMMAND は指定位置にかかわらず最後に実行されます。

(4) 指定例

-SUBCOMMAND=makelib

サブコマンドファイル"makelib.sub"からサブコマンドを入力し、ライブラリアンを実行します。

4.4.2 CREATE - ライブラリファイルの生成

(1) フォーマット

名 称	CREATE		オプション	サブコマンド
			あ り	あ り
パラメータ	オプション	UNIX	<ライブラリファイル名>	
		MS-DOS	<ライブラリファイル名>	$\left\{ \begin{array}{l} (S) \\ (U) \end{array} \right\}$
	サブコマンド	<ライブラリファイル名> $\left\{ \begin{array}{l} (S) \\ (U) \end{array} \right\}$		

(2) 機 能

ライブラリファイルを新規に生成します。

(3) 説 明

(1) END または EXIT までのオプション / サブコマンド群の先頭で指定します。

(2) ライブラリファイルは新規ファイルを指定してください。

ライブラリファイル名にファイル形式の指定がない場合は、".lib" というファイル形式を仮定します。

(3) 生成するライブラリファイルの属性を "(S)" または "(U)" で指定します。

省略した場合は "(U)" を仮定します。

(S) ... システムライブラリ

(U) ... ユーザライブラリ

この属性は、リンケージエディタによるライブラリファイルのサーチの優先順位を決めるものです。ユーザライブラリの方が優先順位が高くなります。ただし、オペレーティングシステムが UNIX の場合、オプションでの "(S)" と "(U)" の指定はできません。

(4) LIBRARY サブコマンドと一緒に使用することはできません。

(5) 登録モジュール数がゼロの場合、ライブラリファイルは生成しません。

(4) 指定例

```
-CREATE=userlib.lib
```

ユーザライブラリとして"userlib.lib"を新規作成します。

```
CREATE syslib(S)
```

システムライブラリとして"syslib.lib"を新規作成します。

```
CREATE datax
```

ユーザライブラリとして"datax.lib"を新規作成します。

4.4.3 ADD - モジュールの登録 / 追加

(1) フォーマット

名 称	ADD		オプション	サブコマンド
			あ り	あ り
パラメータ	オプ ション	UNIX	$\left\{ \begin{array}{l} \langle \text{オブジェクトモジュールファイル名} \rangle \\ \langle \text{リロケートブルードモジュールファイル名} \rangle \\ \langle \text{ライブラリファイル名} \rangle \end{array} \right\} \{ [\quad ,], \dots \}$	
		MS-DOS	$\left\{ \begin{array}{l} \langle \text{オブジェクトモジュールファイル名} \rangle \\ \langle \text{リロケートブルードモジュールファイル名} \rangle \\ \langle \text{ライブラリファイル名} \rangle [(\langle \text{モジュール名} \rangle [[\quad ,], \dots])] \end{array} \right\} \{ [\quad ,], \dots \}$	
	サブ コマンド	$\left\{ \begin{array}{l} \langle \text{オブジェクトモジュールファイル名} \rangle \\ \langle \text{リロケートブルードモジュールファイル名} \rangle \\ \langle \text{ライブラリファイル名} \rangle [(\langle \text{モジュール名} \rangle [[\quad ,], \dots])] \end{array} \right\} \{ [\quad ,], \dots \}$		

(2) 機 能

指定したファイル内のモジュールをライブラリファイルに登録または追加します。

(3) 説 明

(1) 新規ライブラリファイルにモジュールに登録する場合、または、既存ライブラリファイルにモジュールを追加する場合に指定します。

(2) ファイル名指定のみの場合にファイル形式の指定がないと、".obj"というファイル形式を仮定します。また、ファイル名のあとにモジュールを指定した場合、ライブラリファイルとみなし、ファイル形式の指定がないと、".lib"というファイル形式を仮定します。

(3) ライブラリファイル内の特定のモジュールを追加するときは、ライブラリファイル名の後にモジュール名を指定します。

モジュール名は最大 10 個まで指定できます。ただし、オペレーティングシステムが UNIX の場合、オプションでの指定はできません。

(例)

ADD lbf(m1,m2,m3)

モジュール名
ライブラリファイル名

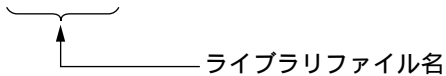
- (4) ライブラリファイル内のモジュール名を指定した場合、モジュールの追加は指定順序で処理するのではなく、指定モジュール名を昇順にソートした順序で追加処理を行います。

(例) `ADD lbf(e,a,d,c,b)`

(5)(1)(4)(3)(2) ... 追加処理順序

- (5) ライブラリファイル内のモジュール名指定を省略すると、ライブラリファイル内の全モジュールを追加します。

(例)

`ADD lbf.lib`
ライブラリファイル名

- (6) 追加するモジュールと同名のモジュールが編集集中のライブラリファイルに存在する場合、または追加するモジュール内で定義している外部定義シンボルと同名の外部定義シンボルが編集集中のライブラリファイルに存在する場合、ウォーニングメッセージを表示し、このモジュールの追加処理を行いません。

- (7) モジュール名はオブジェクトモジュールまたはリロケータブルロードモジュールの中で定義されている名前です。

ライブラリファイルにどのようなモジュールが登録されているか確認するには、LIST オプション / サブコマンドを利用すると便利です。

- (8) EXTRACT, OUTPUT オプション / サブコマンドと一緒に使用することはできません。

- (9) 下記のような場合にはエラーとなり、エラー発生以降の継続パラメータに対する処理は行いません。

- (a) 指定ファイルが存在しない。
- (b) ライブラリファイル中の指定モジュールが存在しない。
- (c) 指定ファイルの内容が正しくない。
- (d) 登録モジュール数が 32,767 を越えた。
- (e) メモリ容量不足で追加処理を行えない。
- (f) 入力ファイルの数が 256 個を越えている。

(4) 指定例

`-ADD=mod1,mod2,modx.o`

オブジェクトモジュールファイル"mod1.obj"、"mod2.obj"および"modx.o"内のモジュールを追加します。

`ADD iofnc(keyin,crtout)`

ライブラリファイル"iofnc.lib"内の2つのモジュール"keyin"と"crtout"を追加します。

`ADD syslib.lib`

ライブラリファイル"syslib.lib"内の全モジュールを追加します。

- (4) ライブラリファイル内のモジュール名を指定した場合、モジュールの置換は指定順序で処理するのではなく、指定モジュール名を昇順にソートした順序で置換処理を行います。

(例) REPLACE lbf(e,a,d,c,b)

(5)(1)(4)(3)(2) ... 置換処理順序

- (5) ライブラリファイル内のモジュール名指定を省略すると、ライブラリファイル内の全モジュールと、編集集中のライブラリファイル内のモジュールとの置換を行います。

(例)

REPLACE lbf.lib
 └─┬─┘
 ↑
 ライブラリファイル名

- (6) モジュール名はオブジェクトモジュールまたはリロケータブルロードモジュールの中で定義されている名前です。ライブラリファイルにどのようなモジュールが登録されているか確認するには、LIST オプション / サブコマンドを利用すると便利です。

- (7) EXTRACT, OUTPUT オプション / サブコマンドと一緒に使用することはできません。

- (8) 下記のような場合にはエラー発生以降の継続パラメータに対する処理は行いません。

- (a) 指定ファイルが存在しない。
- (b) ライブラリファイル中の指定モジュールが存在しない。
- (c) 指定ファイルの内容が正しくない。
- (d) 登録モジュール数が 32,767 を越えた。
- (e) メモリ容量不足で置換処理を行えない。
- (f) 入力ファイルの数が 256 個を越えた。

- (9) モジュールの置換処理は、ライブラリファイル内の同名のモジュールを削除後、REPLACE で指定されたファイルからモジュールを入力し、ライブラリファイルに登録します。このため新しく登録しようとしたモジュールの中にライブラリファイル内の他のモジュールで定義済みの外部定義シンボルを含む場合、旧モジュールの削除のみ行い、新しいモジュールの登録は行いませんのでご注意ください。

(4) 指定例

`-REPLACE=userlib.lib`

ライブラリファイル"userlib.lib"内の全モジュールと、編集中のライブラリファイル内のモジュールとの置換を行います。

`REPLACE loadx.rel,loady.rel`

リロケータブルロードモジュールファイル"loadx.rel"および"loady.rel"内のモジュールと、編集中のライブラリファイル内の同名モジュールとの置換を行います。

`REPLACE datax(member),omf`

ライブラリファイル"datax.lib"内のモジュール"member"およびオブジェクトモジュールファイル"omf.obj"内のモジュールと、編集中のライブラリファイル内の同名モジュールとの置換を行います。

4.4.5 DELETE - モジュールの削除

(1) フォーマット

名 称	DELETE	オプション	サブコマンド
		あ り	あ り
パラメータ	<モジュール名> [{ ！, }...]		

(2) 機 能

指定したモジュールをライブラリファイルから削除します。

(3) 説 明

(1) 指定したモジュールがライブラリファイル中に存在しない場合にはエラーとなり、エラー発生以降の継続パラメータに対する処理は行いません。

(2) モジュール名はオブジェクトモジュールまたはリロケータブルロードモジュールの中で定義されている名前です。

ライブラリファイルにどのようなモジュールが登録されているか確認するには、LIST オプション / サブコマンドを利用すると便利です。

(3) EXTRACT, OUTPUT オプション / サブコマンドと一緒に使用することはできません。

(4) 指定例

`-DELETE=inchar,outchar`

2つのモジュール"inchar"と"outchar"を削除します。

`DELETE datatbl,sort`

2つのモジュール"datatbl"と"sort"を削除します。

4.4.6 EXTRACT - モジュールの抽出

(1) フォーマット

名 称	EXTRACT	オプション	サブコマンド
		あ り	あ り
パラメータ	<モジュール名> [{ !, } …]		

(2) 機 能

指定したモジュールをライブラリファイルから抽出します。

(3) 説 明

- (1) 抽出したモジュールは OUTPUT オプション / サブコマンドで指定されたファイルにライブラリファイル形式で出力します。
- (2) モジュール名は、オブジェクトモジュールまたはリロケータブルロードモジュールの中で定義されている名前です。
ライブラリファイルにどのようなモジュールが登録されているか確認するには、LIST オプション / サブコマンドを使用すると便利です。
- (3) 指定したモジュールがライブラリファイル中に存在しない場合にはエラーとなり、エラー発生以降の継続パラメータに対する処理は行いません。
- (4) CREATE, ADD, DELETE または REPLACE オプション / サブコマンドと一緒に使用することはできません。

(4) 指定例

```
-EXTRACT=add,sub,mul,div
```

既存ライブラリファイルから4つのモジュール "add", "sub", "mul" および "div" を抽出します。

```
EXTRACT alpha,upper,lower,digit,cntrl
```

既存ライブラリファイルから5つのモジュール "alpha", "upper", "lower", "digit" および "cntrl" を抽出します。

4.4.7 RENAME - セクション名の変更

(1) フォーマット

名 称	RENAME	オプション	サブコマンド
		あ り	あ り
パラメータ	<モジュール名>[,...] (<セクション名1>=<セクション名2>[,...])		

(2) 機 能

ライブラリファイル内のセクション名をモジュール単位に変更します。

(3) 説 明

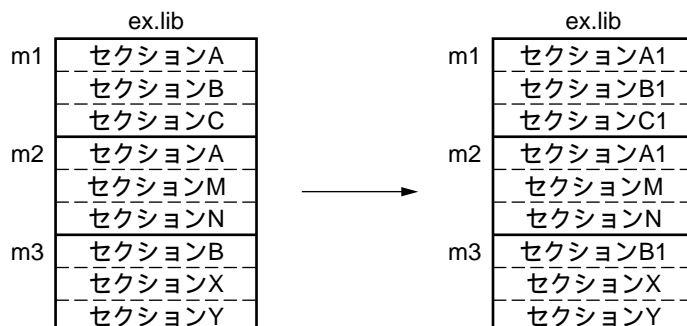
- (1) ライブラリファイル内のセクション名を変更することによって、リンク時のセクション単位によるメモリ割り付けに自由度をもたらすことができます。
- (2) リロケータブルロードモジュールを含むライブラリファイルのセクション名は変更できません。
- (3) デバッグ情報を含むモジュールのセクション名を変更した場合、デバッグ時のシンボル参照が正しく行えない場合があります。

(4) 指定例

```
RENAME      m1,m2,m3 (A=A1,B=B1,C=C1)
```

モジュール名 m1 の内のセクション A,B,C を A1,B1,C1 に変更します。

また、モジュール名 m2 内のセクション A を A1 に、モジュール名 m3 内の B を B1 に変更します。



4.4.8 END - サブコマンド入力の終了指定

(1) フォーマット

名 称	END	オプション	サブコマンド
		な し	あ り
パラメータ	な し		

(2) 機 能

新規作成または更新したライブラリファイルの出力を行います。

(3) 説 明

(1) 1回のライブラリアンの実行で複数ライブラリファイルの編集を行う場合、各ライブラリファイルの編集操作を END サブコマンドによって終了させます。

(2) END サブコマンドが指定されると、ライブラリアンはライブラリファイルの出力を行います。ただし、ライブラリファイルの登録モジュール数がゼロの場合、ライブラリファイルの作成または更新は行いません。

(4) 指定例

END

ライブラリファイルの出力を行います。

4.4.9 EXIT - ライブラリアンの終了指定

(1) フォーマット

名 称	EXIT	オプション	サブコマンド
		な し	あ り
パラメータ	な し		

(2) 機 能

ライブラリアンの処理を終了します。

(3) 説 明

- (1) サブコマンド指定による実行の場合、EXIT サブコマンドによって、ライブラリアンの実行を終了させます。
- (2) サブコマンドファイルの場合、EXIT サブコマンドのあとに続くサブコマンドは無効になります。
- (3) EXIT サブコマンドを使用する場合、直前の END サブコマンドの指定を省略することができます。省略した場合、EXIT サブコマンドが指定された時点でライブラリファイルを出力し、ライブラリアンの実行を終了します。

(4) 指定例

EXIT

ライブラリアンの処理を終了します。

4.4.10 ABORT - ライブラリアンの強制終了指定

(1) フォーマット

名 称	<u>ABORT</u>	オプション	サブコマンド
		な し	あ り
パラメータ	な し		

(2) 機 能

ライブラリアンの処理を強制終了させます。

(3) 説 明

- (1) サブコマンド指定による実行の途中で、ライブラリファイルの編集操作を打ち切りたい場合、ABORT サブコマンドを使用します。
- (2) ABORT サブコマンドを指定すると、編集中のライブラリファイルの作成または更新は行いません。ただし、ABORT サブコマンド指定以前で LIST サブコマンドによりリストをファイルに出力した場合、リストファイルはそのまま残ります。

(4) 指定例

ABORT

ライブラリアンの処理を中止します。

4.5 内容表示

4.5.1 LIST - ライブラリファイルの内容表示

(1) フォーマット

名 称	LIST			オプション	サブコマンド
				あ り	あ り
パラメータ	オプ ション	UNIX	[<リストファイル名>]		
		MS-DOS	[[<リストファイル名>] [(S)]]		
	サブ コマンド	[[<リストファイル名>] [(S)]]			

(2) 機 能

編集中のライブラリファイルの内容を標準出力装置またはファイルに出力します。

(3) 説 明

- (1) ライブラリファイルの登録モジュール名および外部定義シンボル名等の情報をリストとして出力します。リストの出力形式の詳細については「6.2 ライブラリアンリスト」を参照してください。
- (2) リストファイル名の指定がない場合はリストを標準出力装置に出力します。
- (3) リストファイル名の指定がある場合はリストをファイルに出力します。リストファイルは新規ファイルを指定してください。既存ファイルへの追加はできません。既存ファイルを指定した場合、既存ファイルの内容を書き換えます。
- (4) リストファイル名にファイル形式の指定がないと".lst"というファイル形式を仮定します。
- (5) モジュール内で定義されている外部定義シンボル名を表示したい場合には、"(S)" を指定します。"(S)" の指定がないときはモジュール名のみ表示します。ただし、オペレーティングシステムがUNIXの場合、オプションでの"(S)"の指定はできません。
- (6) LIST オプション / サブコマンドは編集操作中、何度指定してもかまいません。指定された時点のライブラリファイルの内容を表示します。

(4) 指定例

`-LIST`

リストを標準出力装置に出力します。外部定義シンボル名の表示は行いません。

`LIST`

リストを標準出力装置に出力します。外部定義シンボル名の表示は行いません。

`LIST libx(S)`

外部定義シンボルを含むリストを"libx.lst"というファイルに出力します。

4.5.2 SLIST - ライブラリファイルのセクション内容の表示

(1) フォーマット

名 称	SLIST	オプション	サブコマンド
		あ り	あ り
パラメータ	[<リストファイル名>]		

(2) 機 能

編集中のライブラリファイルの内容を標準出力装置またはファイルに出力します。

(3) 説 明

(1) ライブラリファイルの登録モジュール名、外部定義シンボル名および外部定義シンボル名の属するセクション名等の情報をリストとして出力します。リストの出力形式の詳細については「6.3 セクション名リスト」を参照してください。

(2) リストファイル名の指定がない場合はリストを標準出力装置に出力します。

(3) リストファイル名の指定がある場合はリストをファイルに出力します。

リストファイルは新規ファイルを指定してください。既存ファイルへの追加はできません。既存ファイルを指定した場合、既存ファイルの内容を書き換えます。

(4) リストファイル名にファイル形式の指定がないと".set"というファイル形式を仮定します。

(5) SLIST オプション / サブコマンドは編集操作中、何度指定してもかまいません。指定された時点のライブラリファイルの内容を表示します。

(4) 指定例

-SLIST

セクション名リストを標準出力装置に出力します。

SLIST libx

セクション名リストを"libx.set"というファイルに出力します。

5. ライブラリアンの入力

第5章 目次

5.1	オブジェクトモジュールファイル.....	209
5.2	リロケータブルロードモジュールファイル	210
5.3	ライブラリファイル.....	211

5.1 オブジェクトモジュールファイル

ライブラリアンは、C コンパイラ、アセンブラの出力したオブジェクトモジュールファイルを入力し、ライブラリファイルにモジュールとして登録することができます。

5.2 リロケータブルロードモジュールファイル

リンケージエディタが出力するリロケータブルロードモジュールファイルを入力し、1つのモジュールとしてライブラリファイルに登録することができます。

5.3 ライブラリファイル

ライブラリアンは編集対象としてライブラリファイルを入力します。

また、ライブラリファイルに登録するモジュールをライブラリファイルから入力することもできます。この場合、ライブラリファイルの特定のモジュールを入力することも、ライブラリファイル内のすべてのモジュールを1度に入力することも可能です。

ライブラリファイルは本ライブラリアンによって作成したものに限ります。

6. ライブラリアンの出力

第6章 目次

6.1	ライブラリファイル.....	215
6.2	ライブラリアンリスト	216
6.3	セクション名リスト.....	219
6.4	コンソールメッセージ	221

6.1 ライブラリファイル

ライブラリアンは、複数モジュールを1つにまとめて、ライブラリファイルとして出力します。また、既存のライブラリファイルを更新した結果、および既存ライブラリファイルから抽出したモジュールをライブラリファイルの形式で出力します。

6.2 ライブラリアンリスト

LIST オプション / サブコマンドが指定された場合、ライブラリファイルの内容表示を標準出力装置またはファイルに出力します。ライブラリアンリストの形式を図 6.1 に示します。

Library file name:	(1)	
(1)		
Attribute:	(2)	
Number of modules:	(3)	Creation date: (5)
Number of symbols:	(4)	Revision date: (6)
(7)	(8)	Entry date: (9)
(10)		(10)
:		:
:		:
(7)	(8)	Entry date: (9)

図 6.1 ライブラリアンリストの形式

(1) ライブラリファイル名を表示します。1 行に収まらない場合は、次行に継続して表示します。既存ライブラリファイルからのモジュールの抽出の場合、リストには既存ライブラリファイルの内容を表示します。

(2) ライブラリファイルの属性を表示します。

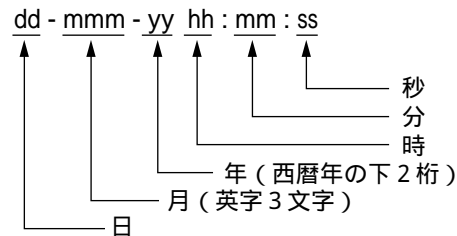
SYSTEM ... システムライブラリ

USER ... ユーザライブラリ

(3) 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 10 進数で表示します。

(4) ライブラリファイルに登録されている外部定義シンボルの総数を 10 進数で表示します。

(5) ライブラリファイルの作成年月日と時刻を表示します。表示形式は次のとおりです。



(6) ライブラリファイルの最新の更新年月日と時刻を表示します。CREATE オプション / サブコマンドによって新規作成されたライブラリファイルは作成年月日、時刻と同じ更新年月日、時刻を表示します。表示形式は作成年月日、時刻と同様です。

(7) ライブラリファイルに登録されているモジュールの名前をアルファベット順に表示します。

(8) モジュールに対する編集操作状態を表示します。

空 白 ... 既存ライブラリファイルに登録されているモジュール

(A) ... 追加モジュール

(R) ... 置換モジュール

(E) ... 抽出モジュール

なお、DELETE オプション / サブコマンドによって削除したモジュールはリストに表示しません。

(9) モジュールがライブラリファイルに登録された年月日、時刻を表示します。表示形式はライブラリファイルの作成年月日、時刻と同様です。

(10) LIST サブコマンドで "(S)" が指定された場合、各モジュールで定義している外部定義シンボル名を表示します。シンボル名は 1 行に 2 個ずつアルファベット順に表示します。

図 6.2 に LIST サブコマンドで "(S)" を指定した場合、図 6.3 に "(S)" を指定しなかった場合のリスト出力例を示します。

```

Library file name: clib.lib
Attribute: USER
Number of modules: 6          Creation date: 08-Jan-90 14:18:47
Number of symbols: 6         Revision date: 01-Mar-90 19:56:33

ABS.C                        Entry date: 08-Jan-90 14:18:47
    _abs

ATOF.C                       Entry date: 08-Jan-90 14:18:47
    _atof

ATOI.C                       Entry date: 08-Jan-90 14:18:47
    _atoi

ATOL.C                       Entry date: 08-Jan-90 14:18:47
    _atol

_ALOCBUF                     (A) Entry date: 01-Mar-90 19:56:33
    __alocbuf

_DIVI                        (A) Entry date: 01-Mar-90 19:56:33
    __divi

```

図 6.2 ライブラリアンリスト例 ("S"指定あり、UNIX の場合)

```

Library file name: clib.lib
Attribute: USER
Number of modules: 6          Creation date: 08-Jan-90 14:18:47
Number of symbols: 6         Revision date: 01-Mar-90 19:56:33

ABS.C                        Entry date: 08-Jan-90 14:18:47
ATOF.C                       Entry date: 08-Jan-90 14:18:47
ATOI.C                       Entry date: 08-Jan-90 14:18:47
ATOL.C                       Entry date: 08-Jan-90 14:18:47
_ALOCBUF                     (A) Entry date: 01-Mar-90 19:56:33
_DIVI                        (A) Entry date: 01-Mar-90 19:56:33

```

図 6.3 ライブラリアンリスト例 ("S"指定なし、UNIX の場合)

6.3 セクション名リスト

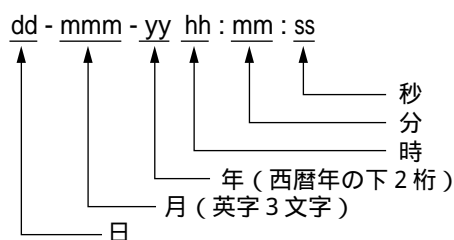
SLIST オプション / サブコマンドが指定された場合、ライブラリファイルのセクション内容表示を標準出力装置またはファイルに出力します。セクション名リストの形式を図 6.4 に示します。

Library file name:	(1)	
	(1)	
Attribute:	(2)	
Number of modules:	(3)	
	Creation date:	(4)
	Revision date:	(5)
(6)		
(7)		(8)
:		:
(6)		
(7)		(8)

図 6.4 セクション名リストの形式

- (1) ライブラリファイル名を表示します。1 行に納まらない場合は、次行に継続して表示します。既存ライブラリファイルからのモジュールの抽出の場合、リストには既存ライブラリファイルの内容を表示します。
- (2) ライブラリファイルの属性を表示します。
SYSTEM...システムライブラリ
USER ...ユーザライブラリ
- (3) 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 10 進数で表示します。

(4) ライブラリファイルの作成年月日と時刻を表示します。表示形式は次のとおりです。



(5) ライブラリファイルの最新の更新年月日と時刻を表示します。CREATE オプション / サブコマンドによって新規作成されたライブラリファイルは作成年月日、時刻と同じ更新年月日、時刻を表示します。表示形式は作成年月日、時刻と同様です。

(6) ライブラリファイルに登録されているモジュールの名前をアルファベット順に表示します。

(7) 各モジュールで定義している外部定義シンボル名を表示します。

(8) 外部定義シンボル名のセクション名を表示します。

図 6.5 に SLIST サブコマンドでのリスト出力例を示します。

```
Library file name: clib.lib
Attribute: USER          Creation date: 08-Jan-90 14:18:47
Number of modules: 6     Revision date: 01-Mar-90 19:56:33

ABS.C
    _abs                                P

ATOF.C
    _atof                              P

ATOI.C
    _atoi                            P1

ATOL.C
    _atol                             CODE

_ALOCBUF
    _alocbuf                          P

_DIVI
    _divi                             P2
```

図 6.5 セクション名リスト例

6.4 コンソールメッセージ

ライブラリアンは次のようなメッセージを標準出力装置に表示します。

(1) 起動メッセージ

ライブラリアンのコマンドが入力されると表示します。

```
H SERIES OBJECT LIBRARIAN Ver. 1.4  
Copyright (C) Hitachi, Ltd. 1988  
Copyright (C) HITACHI MICROCOMPUTER SYSTEM LTD. 1988  
Licensed Material of Hitachi, Ltd.
```

(2) 正常終了メッセージ

ライブラリファイルの編集処理が正常に終了した場合に表示します。

```
OBJECT LIBRARIAN COMPLETED
```

(3) 異常終了メッセージ

ライブラリファイル編集中にエラーが発生し処理を打ち切った場合、または ABORT サブコマンド指定により処理を打ち切った場合に表示します。

```
OBJECT LIBRARIAN ABORT
```

(4) サブコマンド要求記号

会話形式の実行においてサブコマンドの入力状態を表します。

```
:
```

(5) サブコマンド継続記号

会話形式の実行においてサブコマンドの継続が指定された場合、次行に継続行要求を示すために表示します。

```
-
```

7. エラーメッセージ

ライブラリアンのエラーメッセージは次のような形式で出力します。

** <エラー番号><エラーメッセージ> [(<付加情報>)]

エラー番号 : 上1桁でエラーのレベルを示します。(××はエラー番号の下2桁を示します。)

1×× : ウォーニング……………当該モジュールの処理をスキップします。

2×× : エラー……………コマンドまたはサブコマンドファイルからの入力時は、処理を中断します。会話形式の場合には、エラーを検出した時点で当該サブコマンドの処理を打ち切り、次のサブコマンド要求状態となります。

3×× : フェイタルエラー……………処理を中断します。

エラーメッセージの一覧を以下に示します。記述形式は次のとおりです。

エラー番号	エラーメッセージ	付加情報 ^{*1}
エラー内容		
対応策その他		

【注】^{*1} 付加情報にはエラーとなったファイル名、モジュール名またはシンボル名を表示します。エラーメッセージ一覧表の付加情報の欄が"---"の場合は、付加情報のないことを示します。

表 7.1 ウォーニングメッセージ一覧表

101	DUPLICATE MODULE	モジュール名
ライブラリファイルにすでに登録済みのモジュールを再登録しようとした。		
当該モジュールに対する処理をスキップします。		
102	DUPLICATE SYMBOL	モジュール名 ** シンボル名
ライブラリファイルにすでに登録済みの外部定義シンボルを再登録しようとした。		
当該モジュールに対する処理をスキップします。		
103	IDENTIFIER CHARACTER EXCEEDS 251	モジュール名
251 文字を超えるモジュール名が指定された。		
251 文字までを有効とします。		
104	EXIT SUBCOMMAND NOT FOUND - ASSUMED	---
EXIT サブコマンドが指定されていない。		
EXIT サブコマンドを仮定して処理を続行します。		
105	SUBCOMMAND LINE LENGTH TOO LONG	---
ディレクトリ名の置き換えで文字数が 511 文字を超えた。		
511 文字までを有効とします。		
106	TOO MANY DIRECTORY COMMANDS	---
DIRECTORY サブコマンドで 16 個を超えたディレクトリ名を指定した。		
16 個までを有効とします。		
107	MODULE COUNT 0	---
モジュールの総和が 0 となった。		
何もせずに処理を終了します。編集方法を確認してください。		
108	SECTION NOT FOUND	モジュール名 ** セクション名
指定したセクションが 1 つも見つからない。		
セクション名を確認の上、再度指定してください。		
109	CANNOT PRINT SECTION LIST	モジュール名
リロケータブルロードモジュールが登録されているファイルに SLIST オプション / サブコマンドを指定した。		
SLIST オプション / サブコマンドは、オブジェクトモジュールにのみ有効です。		
110	CANNOT RENAME SECTION NAME	モジュール名
リロケータブルロードモジュールが登録されているファイルに RENAME オプション / サブコマンドを指定した。		
RENAME オプション / サブコマンドは、オブジェクトモジュールにのみ有効です。		

表 7.2 エラーメッセージ一覧表

201	INVALID SUBCOMMAND/OPTION	---
不正なオプション / サブコマンド名が指定された。		
正しいオプション / サブコマンド名を指定してください。		
202	SYNTAX ERROR	---
指定されたオプション / サブコマンドに構文上の不正がある。		
オプション / サブコマンドの構文を確認の上、再入力してください。		
203	SUBCOMMAND LINE LENGTH TOO LONG	---
サブコマンドの長さが 512 文字を超えている。		
サブコマンドの長さが 512 文字に収まるように再入力してください。		
204	CONFLICTING SUBCOMMAND	---
サブコマンドの指定順序がおかしい、または指定できない組み合わせである。		
サブコマンドの指定順序を確認の上、再入力してください。		
205	ILLEGAL FILE NAME	---
不正なファイル名が指定された。		
正しいファイル名を指定してください。		
206	ILLEGAL MODULE NAME	---
不正なモジュール名が指定された。		
正しいモジュール名を指定してください。		
207	MODULE NOT FOUND	モジュール名
指定されたモジュールが見つからない。		
指定したモジュール名を確認の上、再指定してください。		
208	MISSING OUTPUT FILE NAME	---
EXTRACT オプション / サブコマンドに対する出力ファイルの指定がない。		
OUTPUT オプション / サブコマンドで、出力ファイルを指定してください。		
209	TOO MANY INPUT FILES	---
256 個を超えて入力ファイルを同時に入力しようとした。		
いったんライブラリファイルに出力し、そのライブラリファイルを再入力して残りのファイルを入力してください。		
210	TOO MANY MODULES	---
モジュール数が許容範囲を超えた。		
現在、作成または編集中のライブラリファイルにこれ以上モジュールを登録することはできません。別のライブラリファイルに登録してください。		

(次ページへ続く)

7. エラーメッセージ

211	TOO MANY SYMBOLS	---
シンボル数が許容範囲を超えた。		
現在、作成または編集中のライブラリファイルにこれ以上シンボルを登録することはできません。 別のライブラリファイルに登録してください。		
212	ILLEGAL FILE FORMAT	---
指定されたファイルのフォーマットが不正である。		
ファイルの内容を確認の上、再実行してください。		
213	MEMORY OVERFLOW	---
ライブラリアンの使用可能なメモリにスペースがない。		
メモリの拡張を行い再実行してください。		
214	FILE NOT FOUND	ファイル名
指定されたファイルが見つからない。		
ディレクトリの内容および指定したファイル名を確認の上、再指定してください。		
215	DUPLICATE SECTION	モジュール名 ** セクション名
指定したセクションがモジュール内に存在している。		
セクション名を確認の上、再指定してください。		
216	ILLEGAL SECTION NAME	---
指定したセクション名が不正である。		
セクション名を確認の上、再指定してください。		
217	CANNOT RENAME SECTION NAME	モジュール名
デバッグ情報が指定されているファイルに RENAME オプション/サブコマンドを指定した。		
RENAME オプション/サブコマンドは、デバッグ情報が指定されていないオブジェクトモジュール のみに有効です。		

表 7.3 フェイタルエラーメッセージ一覧表

301	INVALID COMMAND PARAMETER	---
不正なコマンドパラメータが指定された。		
コマンドパラメータの内容を確認の上、再実行してください。		
302	CONFLICTING OPTION	---
オプションの指定に矛盾がある。		
オプションの指定順序を確認の上、再入力してください。		
303	CANNOT OPEN FILE	ファイル名
ファイルをオープンできなかった。または CREATE, OUTPUT オプション / サブコマンドで既存ファイルを指定した。		
指定したファイル名を確認してください。ファイル名が正しい場合はディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。ライブラリファイルと同一ディレクトリに付加情報ファイル (*.lct) が存在しているか確認してください。再実行の際は、CREATE、OUTPUT オプション / サブコマンドで指定した既存ファイルを削除してください。		
304	CANNOT INPUT FILE	ファイル名
ファイルから入力できなかった。		
指定したファイル名を確認してください。		
ファイル名が正しい場合は、ディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
305	CANNOT OUTPUT FILE	ファイル名
ファイルへ出力できなかった。		
指定したファイル名を確認してください。		
ファイル名が正しい場合はディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
306	CANNOT CLOSE FILE	ファイル名
ファイルをクローズできなかった。		
指定したファイル名を確認してください。		
ファイル名が正しい場合はディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
307	CANNOT READ	---
強制終了が入力されたため処理を終了しました。		
再実行してください。		

(次ページへ続く)

308	MEMORY OVERFLOW	---
ライブラリアンの使用可能なメモリスペースがない。		
動作環境を確認の上、再実行してください。		
309	PROGRAM ERROR	nnn
ライブラリアンの内部処理で何らかの障害が生じました。		
エラー番号 100 が発生した場合は、リンケージエディタが生成する付加情報ファイル(*.rct)またはライブラリアンが生成する付加情報ファイル(*.lct)とオブジェクトモジュール、ライブラリファイルの作成日付が同一であるか確認してください。		
それ以外についてはエラー番号(nnn)を確認の上、当社営業担当までご連絡ください。		

【注】 オペレーティングシステムが UNIX の場合、下記の形式の中間ファイルを使用しているため、エラーメッセージの付加情報として中間ファイル名を表示する場合があります。

Annnnn. TEMP
└───┘
10進5桁の数字

8. 制限事項一覽

表 8.1 にライブラリアンの制限事項一覧を示します。ライブラリアンではこれらの制限値を超える処理はできません。

表 8.1 制限事項一覧表

項目	項目	制限値	備考
1	ライブラリファイル内の登録モジュール数	最大 32,767 個	ただし、ライブラリアンが稼働するシステムのメモリ容量によって制限を受けない場合
2	ライブラリファイル内の登録シンボル数	最大 65,535 個	
3	入力ファイル数	最大 256 個	サブコマンドファイルを除く LIBRARY/ADD/REPLACE で指定されたファイルの総数
4	ライブラリファイル内のモジュール名指定数	最大 10 個	ADD/REPLACE でライブラリファイルを指定した場合
5	ファイル名の長さ	最大 255 文字	デフォルトのファイル形式も含む ファイル名の形式は OS に依存する
6	モジュール名の長さ	最大 251 文字	
7	シンボル名の長さ	最大 251 文字	
8	入力ファイル形式	<ul style="list-style-type: none"> ・アセンブラ、C コンパイラが出力したオブジェクトモジュールファイル ・リロケータブル形式のロードモジュールファイル ・本ライブラリアンで作成したライブラリファイル 	

付録

付録 目次

付録 A. ライブラリアン使用例.....	237
A.1 コマンドライン指定による実行例	237
A.2 サブコマンド指定による実行例.....	239
付録 B. ライブラリアン使用上の注意.....	241

付録 A. ライブラリアン使用例

A.1 コマンドライン指定による実行例

lbr	-CREATE=func-ADD=abs, mod, sqrt, exp, log (RET) (1) 作成
	(a) (b)	
lbr	func-ADD=sin, cos-DELETE=abs, mod-LIST (RET) (2) 編集
	(c) (d) (e) (f)	
lbr	func-EXTRACT=sqrt, exp-OUTPUT=newfnc (RET) (3) 抽出
	(g) (h) (i)	

- (a) 新規にライブラリファイルを作成するため、オプション群の先頭で CREATE オプションを指定します。
- (b) 登録するモジュールのファイル名を ADD オプションで指定します。
- (c) 編集を行うライブラリファイル名を指定します。
- (d) 既存ライブラリファイルに追加するモジュールのファイル名を ADD オプションで指定します。
- (e) 既存ライブラリファイルから削除するモジュール名を DELETE オプションで指定します。
- (f) 編集した結果を確認するために、LIST オプションを指定します。
- (g) 抽出元となる既存ライブラリファイルを指定します。
- (h) 抽出するモジュール名を EXTRACT オプションで指定します。
- (i) 抽出したモジュールを出力する新しいライブラリファイルの名前を OUTPUT オプションで指定します。

図 A.1 に上記の実行結果を示します。

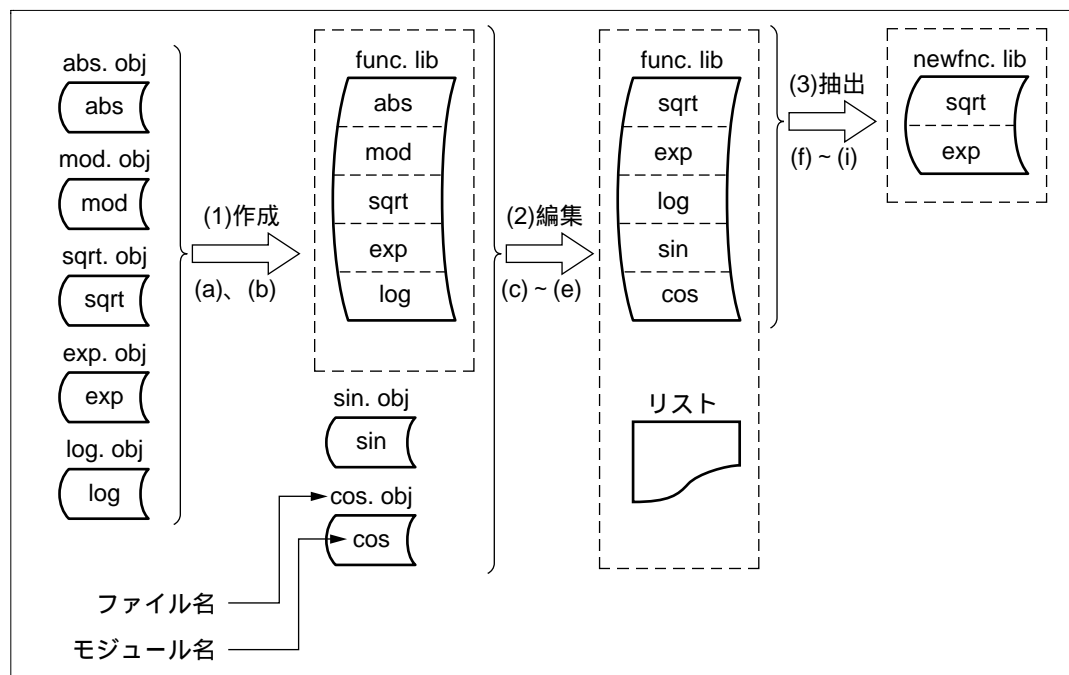


図 A.1 コマンドライン指定例の実行結果

A.2 サブコマンド指定による実行例

<u>lbr (RET)</u> (a)	
: <u>CREATE func (RET)</u> (b)	(1) 作成
: <u>ADD sqrt, exp, log, sin, cos (RET)</u> (c)	
: <u>END (RET)</u> (d)	
: <u>LIBRARY func (RET)</u> (e)	(2) 編集
: <u>REPLACE sin. new, cos. new, tan. new (RET)</u> (f)	
: <u>END (RET)</u> (g)	
: <u>LIBRARY func (RET)</u> (h)	(3) 抽出
: <u>LIST (RET)</u> (i)	
: <u>EXTRACT sqrt. exp (RET)</u> (j)	
: <u>OUTPUT newfnc (RET)</u> (k)	
: <u>END (RET)</u> (l)	
: <u>EXIT (RET)</u> (m)	

- (a) ライブラリアンを起動します。
- (b) 新規にライブラリファイルを作成するため、サブコマンド群の先頭で CREATE サブコマンドを指定します。
- (c) 登録するモジュールのファイル名を ADD サブコマンドで指定します。
- (d) 作成処理を終えるために、END サブコマンドを指定します。
- (e) 編集を行うライブラリファイル名を指定します。
- (f) 既存ライブラリファイル内のモジュールを REPLACE サブコマンドを用いて置換します。REPLACE では置き換えるモジュールのファイル名を指定します。
- (g) 編集処理を終えるために、END サブコマンドを指定します。
- (h) 抽出元となる既存ライブラリファイルを指定します。
- (i) 既存ライブラリファイルの内容を確認するため、LIST サブコマンドを指定します。
- (j) 抽出するモジュール名を EXTRACT サブコマンドで指定します。
- (k) 抽出したモジュールを出力する新しいライブラリファイルの名前を OUTPUT サブコマンドで指定します。

(l) 抽出処理を終えるために、END サブコマンドを指定します。

(m) ライブラリアンを終了させるために、EXIT サブコマンドを指定します。

図 A.2 に上記の実行結果を示します。

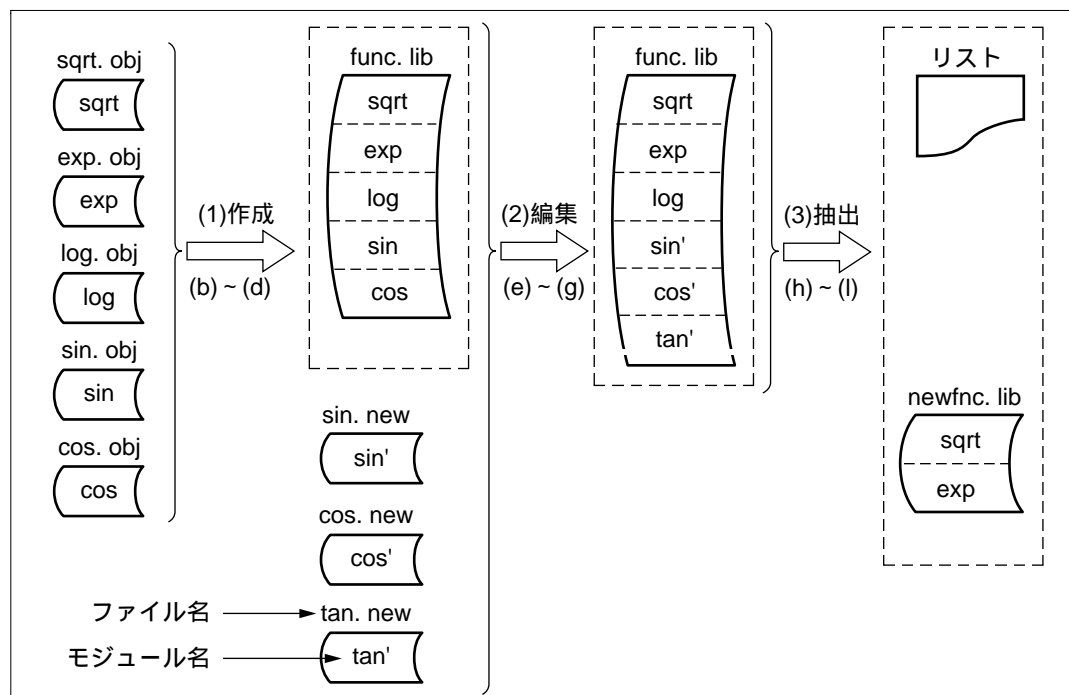


図 A.2 サブコマンド指定例の実行結果

付録 B. ライブラリアン使用上の注意 (MS-DOS で使用する場合)

本ライブラリアンを MS-DOS でご使用になる前に、MS-DOS のシステム構築ファイル (CONFIG.SYS) の内容をエディタで以下のように指定してください。

FILES=20 (1)
SHELL=a:¥command.com a:¥ /p (2)

(1) ライブラリアン動作時に、同時にオープンできるファイル数の指定

(2) COMMAND.COM を再ロードする際に必要なパス情報の指定

オブジェクトコンバータ編

1. オブジェクトフォーマットの 変換

第1章 目次

1.1	オブジェクトコンバータの機能.....	246
1.2	オブジェクトコンバータの実行.....	247
1.3	コンバートファイルの分割出力.....	250
1.4	エラーメッセージ.....	251

リンケージエディタが出力したロードモジュールをエミュレータあるいは PROM ライタへ入力するためには、ロードモジュールを S タイプオブジェクトフォーマットに変換しなければいけません。S タイプオブジェクトフォーマットへの変換は、オブジェクトコンバータを使用して行います。

1.1 オブジェクトコンバータの機能

Hシリーズ オブジェクトコンバータ Ver.2.0 では、以下オプションが追加になりました。

- ・ RECORD オプション
- ・ S9 オプション

(1) コマンドフォーマット

オプションを指定する場合、コマンドフォーマットは、次のとおりです。

```
cnvs <入力ファイル名>[ <出力ファイル名>[ <出力ファイル名>... ]][[ ]-<オプション名>...]
```

(2) RECORD オプション

本オプションは、ロードアドレスに関係なく、一定のデータレコード(S1,S2,S3)で出力します。指定したデータレコードより大きいロードアドレスが存在した場合、エラーメッセージ 310 を表示し、処理を終了します。

(例)

```
cnvs test.abs test.mot -record=s2
```

全てのデータレコードを S2 で出力します。

(3) S9 オプション

ROM ライタには、エンドレコードが S9 でなければ正常に書込みのできない製品があります。本オプションは、エントリアドレスが H'10000 を超える場合でも S9 レコードを終端に出力します。この場合、S9 レコードのエントリアドレスは、H'0 となります。

(例)

```
cnvs test.abs test.mot -s9
```

終端に S9 レコードを出力します。

補足) S タイプオブジェクト形式は、「Hシリーズ リンケージエディタ ライブラリアン オブジェクトコンバータ ユーザーズマニュアル」のオブジェクトコンバータ編を参照してください。

1.2 オブジェクトコンバータの実行

オブジェクトコンバータを実行するためのコマンドフォーマットは、次のとおりです。

```
cnvs <入力ファイル名>[ <出力ファイル名>] (RET)
```

ファイル名の構成については、「付録B. ファイル名の構成」を参照してください。

(1) 起動コマンド

オブジェクトコンバータの起動コマンドとして"cnvs"を入力します。

(2) 入力ファイル名

オブジェクトコンバータの入力となるアブソリュート形式のロードモジュールファイル名を指定します。リロケータブル形式のロードモジュールファイルを指定することはできません。入力ファイル名にファイル形式の指定がない場合は、オブジェクトコンバータが、自動的に"abs"をファイル形式と仮定してファイルを入力します。

(3) 出力ファイル名

オブジェクトコンバータの出力となるSタイプオブジェクトファイル名を指定します。出力ファイル名にファイル形式の指定がない場合は、オブジェクトコンバータが自動的に"mot"をファイル形式と仮定してファイルを出力します。

以下に使用例を示します。

(例)

```
cnvs prog1.lmd prog1.sty (RET)    ... (1)
```

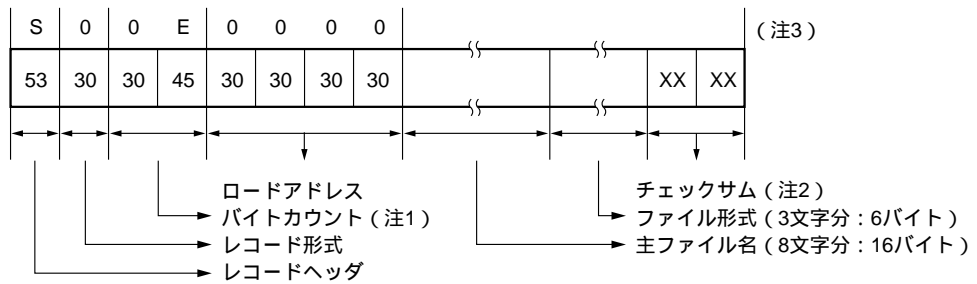
```
cnvs prog1 prog1 (RET)           ... (2)
```

(1) ファイル"prog1.lmd"を入力し、ファイル"prog1.sty"を出力します。

(2) ファイル"prog1.abs"を入力し、ファイル"prog1.mot"を出力します。

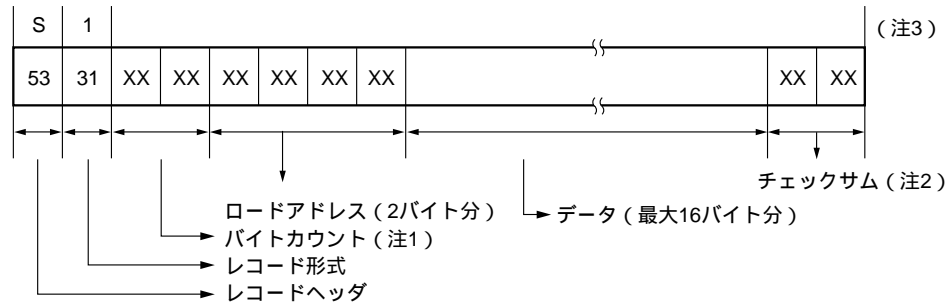
図 1.1 に S タイプオブジェクトの形式を示します。

(a) ヘッダレコード (S0 レコード)

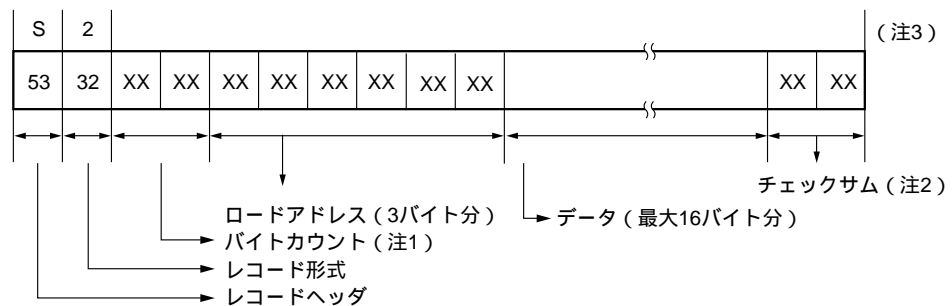


(b) データレコード (S1, S2, S3 レコード)

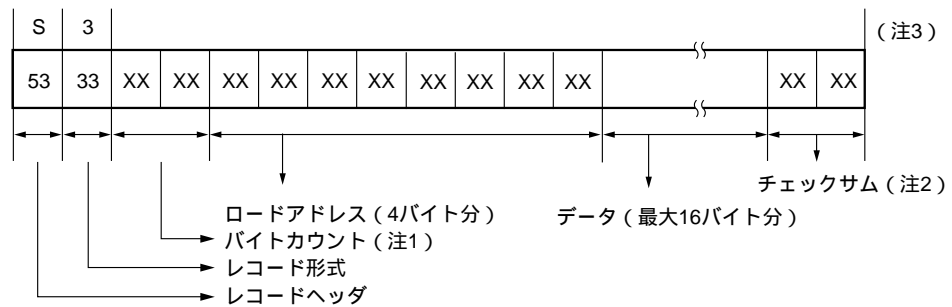
(i) ロードアドレスが 0 - 0FFFF (16 進数) の場合



(ii) ロードアドレスが 10000 - 0FFFFFFF (16 進数) の場合



(iii) ロードアドレスが 1000000 - 0FFFFFFFFF (16 進数) の場合

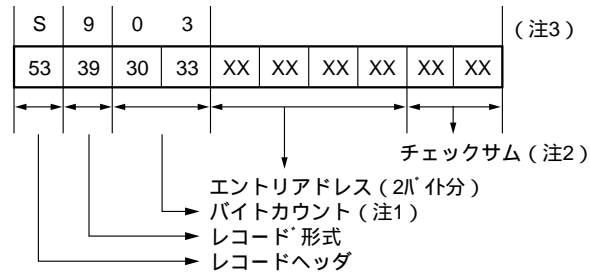


(次ページへ続く)

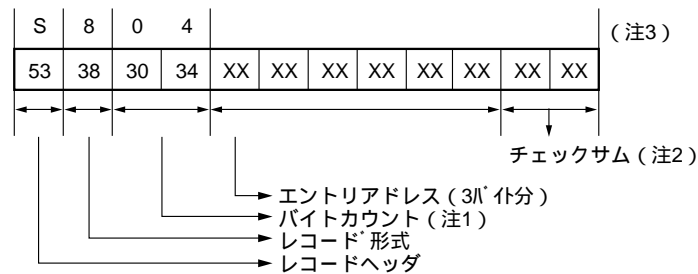
図 1.1 S タイプオブジェクト形式 (1)

(c) エンドレコード (S9,S8,S7 レコード)

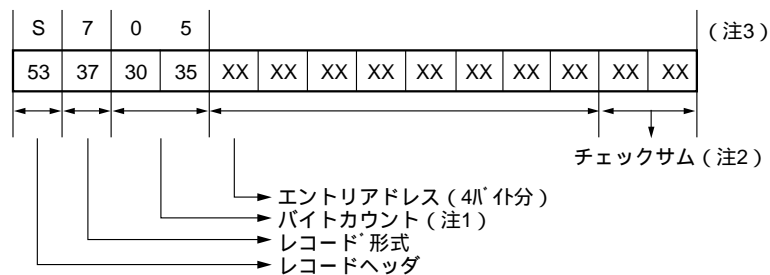
(i) エントリアドレスが 0-0FFFF (16 進数) の場合



(ii) エントリアドレスが 10000-0FFFFFFF (16 進数) の場合



(iii) エントリアドレスが 10000000-0FFFFFFFFF (16 進数) の場合



【注】(1) バイトカウントはロードアドレス (またはエントリアドレス) からチェックサムまでのバイト数を示す。

(2) チェックサムは、バイトカウントからチェックサムの前までのデータ値をバイト単位に加算した結果の1の補数とする。

(3) チェックサムの直後に改行コードが付加されます。

図 1.1 S タイプオブジェクト形式 (2)

1.3 コンバートファイルの分割出力

Sタイプに変換したオブジェクトプログラムを任意のアドレス範囲に分割してファイル出力できるようになりました。

(1) 指定方法

コンバートファイルの分割出力は、出力ファイル名とアドレス範囲を空白で区切って指定します。

分割しない場合	cnvs 入力ファイル名 [出力ファイル]
分割する場合	cnvs 入力ファイル名 出力ファイル=開始アドレス,終了アドレス [出力ファイル=開始アドレス,終了アドレス...]

(2) 機能説明

- ・出力ファイル名の後に開始アドレスと終了アドレスを指定すると、指定範囲内のオブジェクトをSタイプオブジェクトにコンバートします。
- ・開始アドレスおよび終了アドレスは、16進数定数で指定します。
- ・ページタイプの場合、開始アドレスと終了アドレスの前にページアドレスをコロン(：)で区切って指定します。

(3) 使用上の制限

- ・開始アドレスおよび終了アドレスに負の値を指定したとき、開始アドレスより終了アドレスが小さい場合、エラーメッセージ311を出力します。
- ・指定したアドレス範囲にオブジェクトが存在しない場合、ウォーニングメッセージ101を出力します。

(4) 指定例

- ・非ページタイプの場合(H8Sシリーズ、H8/300シリーズ、SuperH RISC engineファミリ)

```
cnvs test.abs test1.mot=0,FFFF test2.mot=10000,1FFFF
```

- ・ページタイプの場合(H8/500シリーズ)

```
cnvs test.abs test1.mot=0:0,0:FFFF test2.mot=1:0,1:FFFF
```

1.4 エラーメッセージ

コマンドの指定内容に誤りがあったり、コンバート処理中にエラーを検出した場合、エラーメッセージを出力します。

エラーメッセージの出力形式は、次のとおりです。

```
* *   <エラー番号>   <エラーメッセージ>[( <付加情報> )]
```

1 カラム目

表 1.1 にエラーメッセージの一覧を示します。表 1.1 の記述形式は、次のとおりです。

エラー番号	エラーメッセージ	付加情報
エラー内容		
対応策その他		

(表中の記号説明) -- : 付加情報なしを示します。

表 1.1 オブジェクトコンバータのエラーメッセージ一覧表

101	NO OBJECT IN SPECIFIED ADDRESS RANGE	---
指定したアドレスの範囲内にオブジェクトが存在しない。		
開始アドレスおよび終了アドレスを確認してください。		
301	INVALID COMMAND PARAMETER	---
不正なコマンドパラメータを指定した。		
コマンドパラメータの内容を確認の上、再実行してください。		
302	FILE NOT FOUND	ファイル名
指定したファイルが見つからない。		
ディレクトリの内容および指定したファイル名を確認の上、再実行してください。		
303	CANNOT OPEN FILE	ファイル名
ファイルをオープンできなかった。		
指定したファイル名を確認してください。ファイル名が正しい場合は、ディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
304	CANNOT READ FILE	ファイル名
ファイルから入力できなかった。		
指定したファイル名を確認してください。ファイル名が正しい場合は、ディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
305	CANNOT WRITE FILE	ファイル名
ファイルへ出力できなかった。		
指定したファイル名を確認してください。ファイル名が正しい場合は、ディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
306	CANNOT CLOSE FILE	ファイル名
ファイルをクローズできなかった。		
指定したファイル名を確認してください。ファイル名が正しい場合は、ディスク容量に空きがないか、またはディスクにハード的なエラーがある場合があります。確認の上、再実行してください。		
307	ILLEGAL FILE FORMAT	ファイル名
指定したファイルのフォーマットが不正である。		
ファイルの内容を確認の上、再実行してください。		
308	ILLEGAL FILE NAME	ファイル名
不正なファイル名を指定した。		
正しいファイル名を指定してください。		

(次ページへ続く)

309	MEMORY OVERFLOW	---
オブジェクトコンバータの使用可能なメモリにスペースがない。		
メモリの拡張またはユーザプログラムの変更を行い、再実行してください。		
310	LOAD ADDRESS OVERFLOW	---
ロードアドレスが RECORD オプションで指定したデータレコードを超えた。		
RECORD オプションのデータレコードを変更するか、RECORD オプションを削除してください。		
311	ILLEGAL ADDRESS RANGE SPECIFIED	---
開始アドレスまたは終了アドレスの指定が正しくない。		
開始アドレスまたは終了アドレスを訂正してください。		

Ver.6.0の追加・変更内容

本編は、以下のソフトウェアで追加、変更した機能の使用方法を述べたものです。

- ・Hシリーズ リンケージエディタ Ver.6.0
- ・Hシリーズ ライブラリアン Ver.2.0
- ・Hシリーズ オブジェクトコンバータ Ver.2.0

1. リンケージエディタ

第1章 目次

1.1	オブジェクトフォーマット	258
1.2	START オプション / サブコマンドの機能拡張	261
1.3	シンボルアドレス出力機能	263
1.4	中間ファイルのディレクトリ指定	265
1.5	仕様変更	266

本章では、Hシリーズ リンケージエディタ Ver.6.0 の追加機能を説明します。
新規に追加した機能の一覧を表 1.1 に示します。

表 1.1 新規機能一覧

No.	項 目	オプション/サポ ポート 環境変数	機 能
1	オブジェクトフォーマット (ELF/DWARF のサポ ポート)	ELF	ELF/DWARFオブジェクトフォーマットを出力
		SYSROF	SYSROFオブジェクトフォーマットを出力(従来オブジェクトフォーマットと互換)
		SYSROFPLUS	SYSROF/DWARFオブジェクトフォーマットを出力
2	STARTオプション/サポ ポート の機能拡張	START	複数セクションの同一アドレス割り付け機能
3	シンボルアドレス出力機能	FSYMBOL	解決済み外部定義シンボルをファイルに出力
4	中間ファイルのディレクトリ指定	HLNK_TMP	中間ファイル出力先のディレクトリ指定機能
5	その他	ENTRY	無効時にウォーニングメッセージ 125 出力

1.1 オブジェクトフォーマット

オブジェクト/デバッグ情報フォーマットとして、ELF/DWARFを指定できるようになりました。

(1) 指定方法

オブジェクトフォーマット出力の選択は、オプション/サブコマンドで指定します。
各オブジェクトフォーマットに対するオプション/サブコマンドは以下のとおりです。

オプション/サブコマンド名	オブジェクトフォーマット
ELF	ELF / DWARF
SYSROF	SYSROF (従来フォーマットと互換)
SYSROFPLUS	SYSROF / DWARF

オブジェクトフォーマットに関するオプションの指定を省略した場合、SYSROFのオブジェクトファイルを出力します。

デバッグ情報出力オプション/サブコマンド (DEBUG/SDEBUG) との組み合わせで、5種類のオブジェクトフォーマットを選択できます。使用するデバッガに合わせて選択してください。

表 1.2 使用可能なデバッガとオプション/サブコマンドの関係

使用可能なデバッガ	オプション/サブコマンド	
	オブジェクトフォーマット	デバッグ情報出力
ELF/DWARF対応のデバッガ	ELF *1	DEBUG
日立統合化ツール (Ver.4) + E8000	ELF *1	SDEBUG
日立統合化ツール (Ver.4) + E7000	SYSROFPLUS *1	SDEBUG
日立統合化ツール (Ver.3) + E7000	SYSROF	SDEBUG
日立デバッグインテグレーション (Ver.2) + E6000	SYSROF	DEBUG

*1: ELF および SYSROFPLUS オプションをデバッグ情報出力指定 (debug/sdebug) と同時に指定する場合、対応可能なコンパイラおよびアセンブラは次のとおりです。

製品名	対応バージョン
SuperH RISC engine C/C++コンパイラ	5.0 以降
SuperH RISC engine アセンブラ	4.0 以降
H8S,H8/300 シリーズ C/C++コンパイラ	3.0 以降
H8S,H8/300 シリーズ クロスアセンブラ	3.0 以降

【注意】

- ・ELF または SYSROFPLUS 対応のコンパイラ、アセンブラで生成したオブジェクトプログラム、ライブラリと旧バージョンで生成したオブジェクトプログラム、ライブラリは、混在してリンクできます。ただし、リンク時に ELF または SYSROFPLUS オプションを指定した場合、旧バージョンで生成したオブジェクトプログラム、ライブラリのデバッグ情報は削除します。
- ・リロケータブルロードモジュールを別ディレクトリに移動する場合は、リンケージエディタが生成する付加情報ファイル(*.rct)も一緒に移動してください。付加情報ファイルが同一ディレクトリに存在しない場合、デバッグ情報が出力されません。

(2) ELF オプション

本オプションは、ELF/DWARF のオブジェクトフォーマットを出力する場合に指定します。デバッグ環境が日立統合化マネージャの場合、リンク時に SDEBUG オプションを指定してください。

【注意】

- ・.ORG 制御命令で指定したロケーションカウンタ値が変更前のセクションと重複している場合、ELF/DWARF フォーマットの指定は行えません。
- ・DEBUG または SDEBUG オプションを指定した場合はモジュール間最適化ツール (OPTLNKSH, OPTLNK38) を起動してください。

【コマンド指定例】

```
shc test1.c -debug
shc test2.c -debug
asmsh test3.src -debug
optlnksh -subcommand=test1.sub
```

【test1.sub の内容】

```
INPUT test1,test2,test3
ELF
SDEBUG
EXIT
```

(3) SYSROF オプション

本オプションは、SYSROF のオブジェクトフォーマットを出力する場合に指定します。

【コマンド指定例】

```
shc test1.c -debug
shc test2.c -debug
asmsh test3.src -debug
lnk test1,test2,test3 -sysrof -debug
```

(4) SYSROFPLUS オプション

本オプションは、SYSROF/DWARF のオブジェクトフォーマットを出力する場合に指定します。デバッグ情報出力指定では、SDEBUG オプションを指定してください。

【注意】 SDEBUG オプションを指定した場合はモジュール間最適化ツール (OPTLNKSH, OPTLNK38) を起動してください。

【コマンド指定例】

```
shc test1.c -debug
shc test2.c -debug
asmsh test3.src -debug
optlnksh -subcommand=test2.sub
```

【test2.sub の内容】

```
INPUT test1,test2,test3
SYSROFPLUS
SDEBUG
EXIT
```

1.2 START オプション / サブコマンドの機能拡張

複数のセクションを同一アドレスに割り付けることができるようになりました。

(1) 指定方法

同一アドレスへの複数のセクションの割り付けは、START オプション / サブコマンドで同一アドレスに割り付けるセクション群をコロン (:) で区切って指定します。

オプション	MS-DOS 版	-START=<セクション名>[,<セクション名>...] [:<セクション名>[,<セクション名>...]...](<先頭アドレス>)
	UNIX 版	-START=<セクション名>[,<セクション名>...] [:<セクション名>[,<セクション名>...]...]/<先頭アドレス>
サブコマンド	START	<セクション名>[,<セクション名>...] [:<セクション名>[,<セクション名>...]...](<先頭アドレス>)

(2) 使用上の制限事項

- ・ページタイプのもジュール (H8/500 シリーズ) では、複数のセクションを同一アドレスに割り付ける機能は使用できません。
- ・複数セクションを同一アドレスに割り付けた場合、セクション間でのシンボル参照はできません。(図 1.1 の RAM_sct1 から RAM_sct2 のシンボルを参照するケース)

(3) 指定例

本機能を用いて、同時に存在しない複数のプログラム / データを外部 ROM から高速な内部 RAM に転送して実行する例を示します。

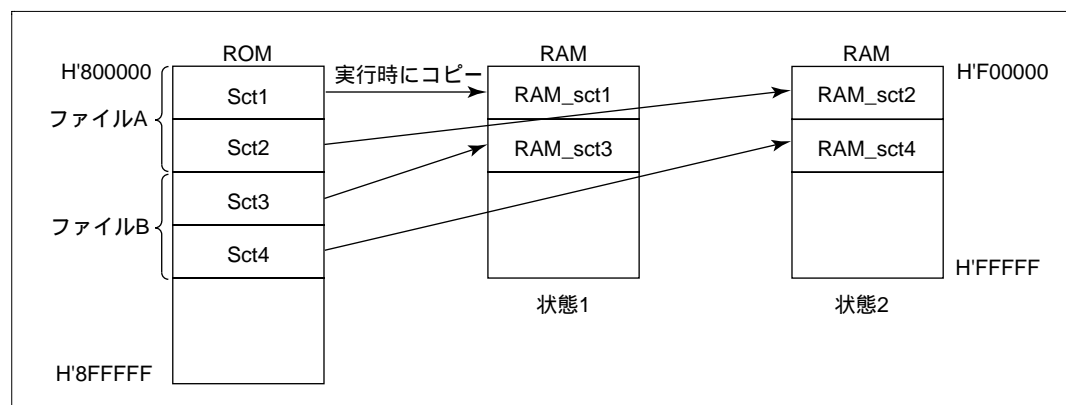


図 1.1 同一アドレスへの複数セクション割り付け

【コマンド指定例】

```
lnk -subcommand=test.sub
```

【test.sub の内容】

```
INPUT  A,B  
ROM    (Sct1, RAM_sct1), (Sct3, RAM_sct3)  
ROM    (Sct2, RAM_sct2), (Sct4, RAM_sct4)  
START  Sct1, Sct2, Sct3, Sct4(800000)  
START  RAM_sct1, RAM_sct3:RAM_sct2, RAM_sct4(0F00000)
```

【説明】

RAM_sct1 と RAM_sct2 を同一アドレスから割り付けます。RAM_sct3 は RAM_sct1 に、RAM_sct4 は RAM_sct2 に各々連結して割り付けます。

1.3 シンボルアドレス出力機能

(1) フォーマット

名 称	オプション		サブコマンド	否定形
	FSYMBOL		FSYMBOL	なし
パ ラメータ	オ プ ショ ン	<セクション名> [,<セクション名>...]		
	サ ブ コマ ンド	<セクション名> [,<セクション名>...]		

(2) 機能

シンボルアドレス出力機能は、リンケージエディタで解決した外部定義シンボルをアセンブラ制御命令の形式でファイルに出力します。出力ファイルをアセンブル、リンクすることにより、当該シンボル定義を含むオブジェクトプログラムをリンクすることなく、外部参照シンボルのアドレスを解決することができます。

(3) 説明

(1) 機能説明

- ・本オプションは、リンケージエディタで解決した外部定義シンボルをアセンブラ制御命令の形式でファイルに出力します。出力するファイル名は、ロードモジュール名にファイル拡張子".fsy"を付加したファイルとなります。

(2) 使用上の制限事項

- ・出力するロードモジュールファイルがリロケータブル形式の場合、本オプション / サブコマンドは使用できません。
- ・外部定義シンボル名の長さが 238 文字を超えるシンボルは、ウォーニングメッセージ 126 を出力し、238 文字までが有効になります。
- ・指定したセクション名が存在しない場合、ウォーニングメッセージ 127 を表示し、処理を継続します。指定したセクションが全て存在しない場合、ファイルは出力されません。

(4) 指定例

図 1.2 は、製品 A の機能 A を機能 B に変更し、製品 B を開発する例です。本機能を用いて、共通 ROM 内シンボルのアドレスを解決することにより、共通 ROM が流用できます。

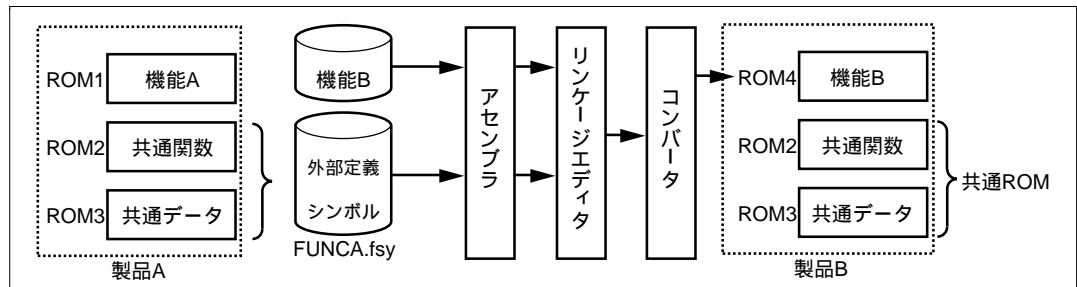


図 1.2 シンボルアドレス出力機能の使用例

【外部定義シンボルファイル出力の指定例】

```
lnk ROM1,ROM2,ROM3-output=FUNCA -fsymbol=sct2,sct3
```

sct2 と sct3 の外部定義シンボルをファイルに出力します。

【ファイル (FUNCA.fsy) の出力例】

```
;H SERIES LINKAGE EDITOR GENERATED FILE 1997.10.10
;fsymbol = sct2, sct3

;SECTION NAME = sct1
.export sym1
sym1: .equ h'00FF0080
.export sym2
sym2: .equ h'00FF0100
;SECTION NAME = sct2
.export sym3
sym3: .equ h'00FF0180

.end
```

【アセンブル、再リンクの指定例】

```
asmsh ROM4
asmsh FUNCA.fsy
lnk ROM4,FUNCA
```

ROM2,ROM3 のオブジェクトファイルをリンクすることなく、ROM4 の外部参照シンボルを解決します。

【注意】

本機能を使用する場合、共通関数から機能 A 内シンボルは参照できません。

1.4 中間ファイルのディレクトリ指定

環境変数により、中間ファイルの出力先のディレクトリを指定できるようになりました。

中間ファイルのディレクトリ指定は、HLNK_TMP 環境変数で指定します。

MS-DOS 版	set HLNK_TMP=<ディレクトリ名>
UNIX 版	setenv HLNK_TMP <ディレクトリ名>

1.5 仕様変更

1.5.1 ENTRY オプションの仕様変更

出力するロードモジュールがリロケータブル形式かつ実行開始アドレス(エントリポイント)が定数値の場合、ウォーニングメッセージ 125 を出力し、指定を無効とします。

1.5.2 リンケージリストの仕様変更

リンケージリストの出力フォーマットを 132 カラム / 行から 80 カラム / 行に変更しました。

1.5.3 入力ファイル数の制限拡張

リンケージエディタが 1 度のリンケージ処理で扱える入力ファイル数を 256 個から 65,535 個に拡張しました。

2. ライブラリアン

第2章 目次

2.1	オブジェクトフォーマット	269
-----	--------------------	-----

2.1 オブジェクトフォーマット

オブジェクトフォーマットとして、ELF/DWARF または SYSROF/DWARF を使用する場合は、H シリーズ ライブラリアン Ver.2.0 以降を使用してください。

オブジェクトフォーマットおよび対応するコンパイラ、アセンブラについては、1.1 オブジェクトフォーマットを参照してください。

【注意】

ライブラリファイルを別ディレクトリに移動する場合は、ライブラリアンが生成した付加情報ファイル(*.lct)も一緒に移動してください。付加情報ファイルが同一ディレクトリに存在しない場合、ライブラリアンではエラーメッセージ 303(CANNOT OPEN FILE(*.lct))を出力します。

3. オブジェクトコンバータ

第3章 目次

3.1	オブジェクトフォーマット	274
3.2	コンバートファイルの分割出力.....	275
3.3	オプションの追加.....	276

本章では、Hシリーズ オブジェクトコンバータ Ver.2.0 の追加機能を説明します。
新規に追加した機能の一覧を表 3.1 に示します。

表 3.1 新規機能一覧

No.	項 目	オプション	機 能
1	オブジェクトフォーマット		ELF/DWARFオブジェクトフォーマットをサポート
2	オブジェクトファイルの分割出力		任意のアドレス範囲毎にファイルを分割出力
3	出力レコードの統一	RECORD	一定のデータレコード (S1,S2,S3)で出力
4	S9レコードの出力	S9	終端に S9レコードを出力

3.1 オブジェクトフォーマット

リンケージエディタで ELF オプション / サブコマンドを指定した場合は、H シリーズ
オブジェクトコンバータ Ver.2.0 以降を使用してください。

3.2 コンバートファイルの分割出力

Sタイプに変換したオブジェクトプログラムを任意のアドレス範囲に分割してファイル出力できるようになりました。

(1) 指定方法

コンバートファイルの分割出力は、出力ファイル名とアドレス範囲を空白で区切って指定します。

分割しない場合	cnvs 入力ファイル名 [出力ファイル]
分割する場合	cnvs 入力ファイル名 出力ファイル=開始アドレス,終了アドレス [出力ファイル=開始アドレス,終了アドレス...]

(2) 機能説明

- ・出力ファイル名の後に開始アドレスと終了アドレスを指定すると、指定範囲内のオブジェクトをSタイプオブジェクトにコンバートします。
- ・開始アドレスおよび終了アドレスは、16進数定数で指定します。
- ・ページタイプの場合、開始アドレスと終了アドレスの前にページアドレスをコロン(：)で区切って指定します。

(3) 使用上の制限

- ・開始アドレスおよび終了アドレスに負の値を指定したとき、開始アドレスより終了アドレスが小さい場合、エラーメッセージ311を出力します。
- ・指定したアドレス範囲にオブジェクトが存在しない場合、ウォーニングメッセージ101を出力します。

(4) 指定例

- ・非ページタイプの場合(H8Sシリーズ、H8/300シリーズ、SuperH RISC engineファミリ)

```
cnvs test.abs test1.mot=0,FFFF test2.mot=10000,1FFFF
```

- ・ページタイプの場合(H8/500シリーズ)

```
cnvs test.abs test1.mot=0:0,0:FFFF test2.mot=1:0,1:FFFF
```

3.3 オプションの追加

Hシリーズ オブジェクトコンバータ Ver.2.0 では、以下オプションが追加になりました。

- ・ RECORD オプション
- ・ S9 オプション

(1) コマンドフォーマット

オプションを指定する場合、コマンドフォーマットは、次のとおりです。

```
cnvs <入力ファイル名>[ <出力ファイル名>[ <出力ファイル名>... ]][[ ]-<オプション名>...]
```

(2) RECORD オプション

本オプションは、ロードアドレスに関係なく、一定のデータレコード(S1,S2,S3)で出力します。指定したデータレコードより大きいロードアドレスが存在した場合、エラーメッセージ 310 を表示し、処理を終了します。

(例)

```
cnvs test.abs test.mot -record=s2
```

全てのデータレコードを S2 で出力します。

(3) S9 オプション

ROM ライタには、エンドレコードが S9 でなければ正常に書込みのできない製品があります。本オプションは、エントリアドレスが H'10000 を超える場合でも S9 レコードを終端に出力します。この場合、S9 レコードのエントリアドレスは、H'0 となります。

(例)

```
cnvs test.abs test.mot -s9
```

終端に S9 レコードを出力します。

補足) S タイプオブジェクト形式は、「Hシリーズ リンケージエディタ ライブラリアン オブジェクトコンバータ ユーザーズマニュアル」のオブジェクトコンバータ編を参照してください。

索引

ア行

アセンブラ	3 , 5 , 103 , 155
アドレス	
- の解決機能.....	5 , 24
- の割り付け.....	17
- 未解決シンボルの表示抑止	27
アドレスチェック	
- 機能.....	33
- の指定	76
アブソリュート.....	7 , 12
- 形式.....	15 , 17 , 33 , 59 , 85
- ロードモジュール.....	6 , 58 , 92
異常終了メッセージ.....	118 , 221
インフォメーションメッセージ	119
ウォーニング.....	123 , 225
ウォーニングメッセージ	12 , 17 , 92 , 124 , 125 , 226
ウォーニング 108 について	126
エコバックの指定.....	91
エラー	123 , 225
エラーメッセージ	123 , 129 , 130 , 225 , 227 , 251
オブジェクト	
- フォーマットの変換	245
- モジュール.....	3 , 5 , 6 , 21 , 22 , 155
- モジュールファイル	3 , 28 , 103 , 209
オブジェクトコンバータ	
- のエラーメッセージ	251 , 252
- の起動コマンド	246
- の実行	246

- の出力ファイル名.....	246
- の入力ファイル名.....	246
オプション	49 , 52 , 177 , 181
- の構成	50 , 178
- のデフォルト	54
- の否定形.....	53
- のフォーマット.....	50 , 178
- の有効範囲.....	54
- 名.....	40 , 169

力行

外部参照.....	24
- シンボル.....	21 , 22 , 23
- シンボル数.....	137
- シンボルの強制定義	32 , 99
- シンボル名.....	95 , 99
- シンボル名の変更.....	32 , 95
- の解決	24 , 25
外部定義	
- シンボル.....	21 , 28
- シンボル数.....	137
- シンボルの削除	32 , 97
- シンボル名.....	95 , 97
- シンボル名の変更.....	32 , 95
- シンボルリスト.....	109 , 112
会話形式.....	42 , 171
- による実行.....	42 , 43
起動コマンド	40 , 169
起動メッセージ.....	118 , 221
強制定義シンボルリスト	109 , 115
共有結合.....	13 , 111
形式種別.....	7 , 12 , 17
継続指定.....	179
結合属性.....	111
更新年月日	217
コマンドライン.....	39 , 40 , 41
- 指定による実行.....	4 , 39 , 41

- のフォーマット	40
コメント	179
コンソールメッセージ	118 , 221

サ行

再入力機能	5 , 28
削除	160 , 198
作成	159 , 191
作成年月日	217
サブコマンド	29 , 41 , 49 , 52 , 167 , 171 , 181
- 継続記号	118
- 指定による実行	4 , 39 , 42
- 入力の終了指定	88 , 202
- の継続指定	50 , 179 , 221
- の構成	50
- のコメント指定	51
- の否定形	53
- のフォーマット	50 , 178
- ファイル	42 , 43 , 83 , 142 , 172 , 189
- ファイルによる実行	43
- ファイルの指定	83
- 要求記号	118 , 221
参照のない外部参照シンボル	64
- を含むモジュール	23
C コンパイラ	155
システムライブラリファイル	21 , 61 , 186 , 190
実行開始アドレスの指定	72
実行形式の指定	41 , 169
実行制御	49 , 81
- 機能	49 , 177
自動入力	21
自動ページング	17 , 75
- の指定	75
シミュレータ・デバッガ	4 , 33
出力ファイルの指定	59
出力ロードモジュールファイル形式の指定	85
シンボル名	137

- の長さ	233
制限事項一覧	137 , 233
正常終了メッセージ	118 , 221
セクション	7
- 数	137
- 属性	7 , 12 , 13
- 内のアドレス解決	26 , 27
- の結合	12 , 13 , 14 , 15 , 16 , 17
- の結合順序	14 , 69
- の集合	12
- の先頭アドレスの指定	69
- 名	7 , 137
- 名リスト	219
絶対アドレス	17 , 24 , 26
相対アドレス	17 , 26
属性	186 , 190

タ行

ダミー結合	13 , 111
短縮指定	182
単純結合	13 , 111
置換	161 , 195
中間リンケージ情報の表示	32 , 94
抽出	162 , 199
追加	160 , 192
デバッグ	
- 援助	49 , 94
- 援助機能	5 , 32 , 49
- 情報	4 , 86
- 情報出力の指定	86
デフォルトライブラリ	21 , 106
- の論理名	106
- ファイル	106
登録	159 , 192
登録シンボル数	233
登録モジュール数	233

ナ行

内容種別.....	110
内容表示.....	163 , 177 , 204
入力情報(リスト).....	109
入力ファイル	
- 形式.....	137 , 233
- 数.....	137 , 233
- の指定	57
- 名.....	40 , 246

ハ行

パラメータ部.....	50 , 174
非ページタイプ.....	6 , 21 , 58 , 62 , 70 , 75 , 100
ファイル	
- 形式.....	40 , 57 , 59 , 61 , 63 , 76
- 制御.....	49 , 57 , 178
- 制御機能.....	49
ファイル名	167 , 178
- の構成	151
- の長さ	233
フェイタルエラー	123 , 173
- メッセージ.....	131 , 132 , 133 , 229
付加情報.....	123 , 225 , 251
ページタイプ.....	6 , 17 , 21 , 58 , 62 , 69 , 75 , 99
- の結合	18 , 19 , 20
変更 / 削除シンボルリスト	109 , 114

マ行

マルチリンケージ機能.....	5 , 31
未解決外部参照シンボル	21
未定義シンボル	
- の表示指定.....	89
- リスト	109 , 113
名称部.....	50 , 178
- の短縮指定.....	54
メモリ割り付け.....	49

- 機能.....	49
モジュール	5 , 6 , 7 , 12 , 21 , 155
- 数	137
- の結合	5 , 12 , 17 , 21 , 22 , 23
- の結合抑止(指定).....	23 , 64
- 名	57 , 137 , 178
- 名指定	21 , 57 , 105
- 名の長さ	137 , 233

ヤ行

ユーザライブラリファイル.....	21 , 61 , 186 , 190
ユニット.....	7 , 12 , 95
- 数	137
- の強制置換.....	30 , 81
- の削除	97
- の自動置換.....	29
- 名	95 , 137
- 名の削除	32
- 名の変更.....	32 , 95

ラ行

ライブラリアン.....	21 , 105 , 155
- の強制終了.....	203
- の終了	173 , 202
- リスト	216
ライブラリファイル.....	21 , 22 , 57 , 61 , 105 , 106 , 155 , 211
- からの入力.....	21
- の指定	21 , 61
- の属性	216
- 名	169
リストファイルの指定.....	63
リターンコード.....	45 , 173
リロケーション情報.....	6
リロケータブル.....	7 , 12
- 形式.....	17 , 59 , 85
- ロードモジュール.....	6 , 21 , 58

- ロードモジュールファイル	104 , 210
リンケージエディタ	155
- 使用例	141
- の機能	11
- の実行	39 , 142
- の終了	45
- の出力	109 , 116 , 118
- の入力	103 , 104 , 105 , 106
リンケージ処理	
- の強制終了指定	90
- の終了指定	89
リンケージマップリスト	109 , 110
リンケージリスト	63 , 109 , 143 , 144 , 145 , 146 , 147 , 148 , 149 , 150
ロードモジュール	3 , 6
- ファイル	3 , 4 , 5 , 28 , 29 , 116 , 118
- ファイルの再入力機能	5 , 28

英文キーワード

ABORT	90 , 203
ADD	192
ALIGN_SECTION	73
AUTOPAGE	75
CREATE	190
C コンパイラ	3 , 103 , 155
CHECK_SECTION	74
CPU	33 , 76
CPUCHECK	78
CPU 情報ファイル	33 , 76
DEBUG	86
DEFINE	32 , 99
DELETE	32 , 97 , 198
DIRECTORY	66 , 188
ECHO	91
ELF	258 , 259
END	31 , 88 , 201
ENTRY	72
EXCHANGE	30 , 81

EXCLUDE.....	23 , 64
EXIT.....	31 , 89 , 202
EXTRACT.....	199
FORM	6 , 17 , 28 , 85
FSYMBOL	67 , 263 , 264
HLNK_LIBRARY1 ~ 3	106
HLNK_TMP	117 , 265
INPUT	21 , 28 , 57
LIBRARY.....	22 , 61 , 185
LIST.....	32 , 94 , 204
NOAUTOPAGE.....	75
NODEBUG	86
NOECHO	91
NOEXCLUDE.....	64
NOLIBRARY	61
NOOUTPUT	59
NOPRINT.....	63
NOUDF.....	92
OUTPUT.....	59 , 186
PRINT	63 , 109
RENAME.....	32 , 95 , 200
REPLACE.....	195
ROM	79
ROM 化支援機能.....	34 , 79
S タイプオブジェクトフォーマット	245
SDEBUG.....	87
SLIST.....	206
START	15 , 69
SUBCOMMAND.....	42 , 43 , 83 , 189
SYSROF.....	258 , 259
SYSROFPLUS.....	258 , 260
UDF	27 , 92
UDFCHECK.....	93