

FR ファミリ SOFTUNE™ リンケージキット マニュアル V6 対応

FR ファミリ SOFTUNE™ リンケージキット マニュアル V6 対応

富士通マイクロエレクトロニクス株式会社

はじめに

■ 本書の目的と対象読者

本書は、富士通 SOFTUNE リンケージキットの機能および使用方法を説明したものです。

本書は、ファミリマイクロプロセッサを使用した応用プログラムを開発する技術者を対象にしています。

リンケージキットはリンカ、ライブラリアン、オブジェクト形式コンバータの3種類のプログラムで構成されています。

■ 商標

FR は、FUJITSU RISC Controller の略で富士通マイクロエレクトロニクス株式会社の製品です。

SOFTUNE は、富士通マイクロエレクトロニクス株式会社の商標です。

Microsoft, Windows は、米国 Microsoft Corporation の米国およびその他の国における登録商標です。

本書に記載されている社名および製品名などの固有名詞は、各社の商標または登録商標です。

■ 本書の全体構成

本書は、以下に示す 4 部構成となっています。

第 1 部 リンケージキット編

この部では、リンケージキットに含まれるツールの概要と、ツール全体に共通する項目について説明します。

第 2 部 リンカ編

この部では、リンカの仕様、オプション、および出力リストなどについて説明します。

第 3 部 ライブラリアン編

この部では、ライブラリアンの仕様、オプション、および出力リストなどについて説明します。

第 4 部 オブジェクト形式コンバータ編

この部では、オブジェクト形式コンバータの種類、オプション一覧、機能説明、およびオブジェクト形式の変換について説明します。

付録

付録では、リンケージキットのエラーメッセージ、HEX8/HEX16/HEX32 フォーマットのレコード形式、および S フォーマットのレコード形式などを記載しています。

- 本資料の記載内容は、予告なしに変更することがありますので、ご用命の際は営業部門にご確認ください。
- 本資料に記載された動作概要や応用回路例は、半導体デバイスの標準的な動作や使い方を示したもので、実際に使用する機器での動作を保証するものではありません。したがって、これらを使用するにあたってはお客様の責任において機器の設計を行ってください。これらの使用に起因する損害などについては、当社はその責任を負いません。
- 本資料に記載された動作概要・回路図を含む技術情報は、当社もしくは第三者の特許権、著作権等の知的財産権やその他の権利の使用権または実施権の許諾を意味するものではありません。また、これらの使用について、第三者の知的財産権やその他の権利の実施ができることの保証を行うものではありません。したがって、これらの使用に起因する第三者の知的財産権やその他の権利の侵害について、当社はその責任を負いません。
- 本資料に記載された製品は、通常の産業用、一般事務用、パーソナル用、家庭用などの一般的用途に使用されることを意図して設計・製造されています。極めて高度な安全性が要求され、仮に当該安全性が確保されない場合、社会的に重大な影響を与えかつ直接生命・身体に対する重大な危険性を伴う用途（原子力施設における核反応制御、航空機自動飛行制御、航空交通管制、大量輸送システムにおける運行制御、生命維持のための医療機器、兵器システムにおけるミサイル発射制御をいう）、ならびに極めて高い信頼性が要求される用途（海底中継器、宇宙衛星をいう）に使用されるよう設計・製造されたものではありません。したがって、これらの用途にご使用をお考えのお客様は、必ず事前に営業部門までご相談ください。ご相談なく使用されたことにより発生した損害などについては、責任を負いかねますのでご了承ください。
- 半導体デバイスはある確率で故障が発生します。当社半導体デバイスが故障しても、結果的に人身事故、火災事故、社会的な損害を生じさせないように、お客様は、装置の冗長設計、延焼対策設計、過電流防止対策設計、誤動作防止設計などの安全設計をお願いします。
- 本資料に記載された製品を輸出または提供する場合は、外国為替及び外国貿易法および米国輸出管理関連法規等の規制をご確認の上、必要な手続きをおとりください。
- 本書に記載されている社名および製品名などの固有名詞は、各社の商標または登録商標です。

目次

第 1 部	リンケージキット編	1
第 1 章	リンケージキットの仕様	3
1.1	リンケージキットの概要	4
1.2	起動方法	5
1.3	強制終了	6
1.4	終了コード	7
1.5	起動メッセージ	8
1.6	終了メッセージ	9
1.7	ヘルプメッセージ	10
1.8	識別子	11
1.9	ファイル名規則	12
1.10	環境変数	13
1.10.1	TMP (ワークディレクトリ)	14
1.10.2	FELANG (メッセージ言語)	15
1.10.3	FETOOL(インストールディレクトリ)	16
1.10.4	LIB911(ライブラリファイル検索ディレクトリ)	17
1.10.5	OPT911(デフォルトオプションファイル格納ディレクトリ)	18
1.10.6	OPT(デフォルトオプションファイル格納ディレクトリ)	19
第 2 章	オプション	21
2.1	オプション	22
2.2	オプションパラメータでの数値表現	23
2.3	オプション指定時の注意と評価	24
2.4	包含 / 相反関係にあるオプションの指定	25
2.5	コマンドラインの指定例	26
第 3 章	共通オプション	27
3.1	共通オプション一覧	28
3.2	共通オプション詳細	29
3.2.1	デフォルトオプションファイル抑止指定 (-Xdof)	30
3.2.2	オプションファイルからの読み込み指定 (-f)	31
3.2.3	ヘルプメッセージの表示 (-help)	33
3.2.4	版数 / メッセージ出力指定 (-V)	34
3.2.5	版数 / メッセージ出力抑止 (-XV)	35
3.2.6	終了メッセージ表示指定 (-cmsg)	36
3.2.7	終了メッセージ表示抑止指定 (-Xcmsg)	37
3.2.8	ワーニング発生時の終了コードを 1 にする指定 (-cwno)	38
3.2.9	ワーニング発生時の終了コードを 0 にする指定 (-Xcwno)	39
第 4 章	オプションファイル	41
4.1	オプションファイルの概要	42
4.2	オプションファイル中での継続指定	43
4.3	オプションファイル中のコメント指定	44

4.4	オプションファイルの記述例	45
4.5	デフォルトオプションファイル	46
第 2 部	リンカ編	49
第 5 章	リンカの仕様	51
5.1	リンカの概要	52
5.2	リンカの機能	53
5.2.1	入出力ファイル / メッセージに関する制御	55
5.2.2	セクションの結合 / 配置に関する制御	56
5.2.3	ライブラリの検索に関する制御	57
5.2.4	エントリアドレス / シンボル値の設定	58
5.3	セクションの種類	59
5.4	セクションの結合	60
5.5	セクションの配置	61
5.5.1	セクションの結合順序が指定されなかった場合の配置例	62
5.5.2	セクションの結合順序が指定された場合の配置例	63
5.5.3	セクショングループの指定がある場合の配置例	64
5.6	セクションの自動配置	65
5.6.1	-AL 1 が指定された場合のセクションの自動配置	66
5.6.2	-AL 2 が指定された場合のセクションの自動配置	68
5.7	ライブラリの検索	71
5.7.1	ライブラリファイルが 1 つの場合の検索例 1	72
5.7.2	ライブラリファイルが 1 つの場合の検索例 2	74
5.7.3	ライブラリファイルが 1 つの場合の検索例 3	75
5.7.4	ライブラリファイルが複数の場合の検索例 1	76
5.7.5	ライブラリファイルが複数の場合の検索例 2	78
5.7.6	ライブラリファイルが個別に指定された場合の処理	80
5.8	ROM/RAM 領域	82
5.9	ROM RAM 転送セクション	83
5.10	CPU 情報ファイル	86
5.11	SOFTUNE V3/V5 ツールで作成したオブジェクトの入力	87
5.12	FR 用オブジェクトと FR80 用オブジェクトの混在	88
第 6 章	リンカのオプション	91
6.1	リンカのオプション一覧	92
6.2	リンカのオプション詳細	94
6.2.1	出力ロードモジュールファイル名指定 (-o)	95
6.2.2	デバッグ情報の出力指定 (-g)	96
6.2.3	デバッグ情報の削除指定 (-Xg)	97
6.2.4	絶対形式ロードモジュールの出力指定 (-a)	98
6.2.5	相対形式ロードモジュールの出力指定 (-r)	99
6.2.6	パディングデータ指定 (-p)	100
6.2.7	ROM 領域のフィル指定 (-fill)	101
6.2.8	外部シンボル情報出力指定 (-symtab)	103
6.2.9	外部シンボル情報出力抑止指定 (-Xsymtab)	104
6.2.10	マップリストファイル名の指定 (-m)	105
6.2.11	マップリスト出力の抑止指定 (-Xm)	106
6.2.12	リスト表示の名前の省略解除 (-dt)	107

6.2.13	メモリ使用情報リストの出力指定 (-mmi)	108
6.2.14	リスト表示のデマングルシンボル名の出力抑止指定 (-Xdemangle)	109
6.2.15	リスト表示のデマングルシンボル名の出力指定 (-demangle)	110
6.2.16	リスト行の桁数指定 (-pw)	111
6.2.17	リスト 1 ページの行数指定 (-pl)	112
6.2.18	ROM 領域のチェックサム指定 (-cs)	113
6.2.19	警告メッセージ出力レベルの指定 (-w)	117
6.2.20	ROM 領域の指定 (-ro)	118
6.2.21	RAM 領域の指定 (-ra)	119
6.2.22	セクション配置順 / アドレスの指定 (-sc)	120
6.2.23	セクショングループの指定 (-gr)	122
6.2.24	パックリンク指定 (-pk)	123
6.2.25	自動配置指定 (-AL)	124
6.2.26	検索ライブラリファイルの指定 (-l)	126
6.2.27	ライブラリ検索パスの指定 (-L)	127
6.2.28	シンボル個別のライブラリの指定 (-el)	128
6.2.29	ライブラリ検索の抑止指定 (-nl)	129
6.2.30	デフォルトライブラリ検索の抑止指定 (-nd)	130
6.2.31	エントリアドレスの指定 (-e)	131
6.2.32	外部シンボル値の仮設定 (-df)	132
6.2.33	ターゲット CPU 指定 (-cpu)	133
6.2.34	CPU 情報ファイル指定 (-cif)	134
6.2.35	オブジェクト混在チェックレベル指定 (-omcl)	135
6.2.36	デバッグ情報存在チェック抑止指定 (-NCI0302LIB)	137
6.2.37	内蔵 ROM/RAM 領域の自動設定 (-set_rora)	138
6.2.38	内蔵 ROM/RAM 領域自動設定の抑止指定 (-Xset_rora)	139
6.2.39	ユーザ指定領域のチェック指定 (-check_rora)	140
6.2.40	ユーザ指定領域のチェック抑止指定 (-Xcheck_rora)	142
6.2.41	セクション配置領域チェック指定 (-check_locate)	143
6.2.42	セクション配置領域チェック抑止指定 (-Xcheck_locate)	146
6.2.43	サイズ 0 のセクション配置チェック指定 (-check_size0_sec)	147
6.2.44	サイズ 0 のセクション配置チェック抑止指定 (-Xcheck_size0_sec)	149
6.2.45	プレリンク処理の抑止指定 (-XPLNK)	150
6.2.46	相対アセンブルリスト入力ディレクトリ指定 (-alin)	151
6.2.47	絶対形式アセンブルリスト出力ディレクトリ指定 (-alout)	152
6.2.48	絶対形式アセンブルリスト出力指定 (-als)	153
6.2.49	絶対形式アセンブルリスト出力モジュール指定 (-alsf)	154
6.2.50	絶対形式アセンブルリスト出力抑止指定 (-Xals)	155
6.2.51	ROM/RAM, ARRAY リスト出力指定 (-alr)	156
6.2.52	ROM/RAM, ARRAY リスト出力モジュール指定 (-alrf)	157
6.2.53	ROM/RAM, ARRAY リスト出力抑止指定 (-Xalr)	158
6.2.54	ROM/RAM, ARRAY リストのシンボルとアドレスの表示位置指定 (-na, -an)	159
6.2.55	外部シンボル相互参照情報リスト出力指定 (-xl)	161
6.2.56	外部シンボル相互参照情報リストファイル名の指定 (-xlf)	162
6.2.57	外部シンボル相互参照情報リスト出力抑止指定 (-Xxl)	163
6.2.58	ローカルシンボル情報リスト出力指定 (-sl)	164
6.2.59	ローカルシンボル情報リストファイル名の指定 (-slf)	165
6.2.60	ローカルシンボル情報リスト出力抑止指定 (-Xsl)	166
6.2.61	セクション詳細マップリスト出力指定 (-ml)	167

6.2.62	セクション詳細マップリストファイル名の指定 (-mlf)	168
6.2.63	セクション詳細マップリスト出力抑止指定 (-Xml)	169
第 7 章	リンカの出力リストファイル	171
7.1	リンカの出力するリストファイルの種類	172
7.2	リンクリストファイル	173
7.2.1	コントロールリスト	174
7.2.2	マップリスト	176
7.2.3	メモリ使用情報リスト	178
7.2.4	シンボルリスト	181
7.3	絶対形式アセンブルリストファイル	183
7.3.1	ヘッダ, インフォメーションリスト	185
7.3.2	ROM/RAM, ARRAY リスト	186
7.3.3	アセンブルソースリスト	188
7.3.4	セクション情報リスト	190
7.3.5	クロスリファレンスリスト	192
7.4	外部シンボル相互参照情報リストファイル	194
7.5	ローカルシンボル情報リストファイル	196
7.6	セクション配置詳細情報リストファイル	198
第 8 章	リンカの制限事項および Q&A	201
8.1	リンカの制限事項	202
8.2	リンカの使用上の Q&A	203
第 3 部	ライブラリアン編	205
第 9 章	ライブラリアンの仕様	207
9.1	ライブラリアンの機能	208
9.2	ライブラリアンの機能分類	209
9.3	ライブラリファイルの作成編集	210
9.4	ライブラリファイル内のモジュール抽出	212
9.5	ライブラリのデバッグ情報の削除	213
9.6	ライブラリファイルの内容チェックと表示	214
9.7	SOFTUNE V3/V5 言語ツールで作成されたオブジェクト	215
9.8	SOFTUNE V3/V5 言語ツールで作成されたライブラリ	216
9.9	FR 用オブジェクトと FR80 用オブジェクトの混在	217
第 10 章	ライブラリアンのオプション	219
10.1	ライブラリアンのオプション一覧	220
10.2	ライブラリアンのオプション詳細	221
10.2.1	モジュールの追加 (登録) (-a)	222
10.2.2	モジュールの置換 (登録) (-r)	223
10.2.3	モジュールの削除 (-d)	224
10.2.4	モジュールの抽出 (-x)	225
10.2.5	リストファイルの出力指定 (-m)	226
10.2.6	リストファイルの出力抑止指定 (-Xm)	227
10.2.7	リストファイルの詳細情報の出力指定 (-dt)	228
10.2.8	リスト 1 ページの行数指定 (-pl)	229
10.2.9	リスト 1 行の桁数指定 (-pw)	230

10.2.10	バックアップファイルの作成 (-b)	231
10.2.11	バックアップファイルの作成抑止 (-Xb)	232
10.2.12	ライブラリファイルの内容検査 (-c)	233
10.2.13	ファイル内容の最適化 (-O)	234
10.2.14	デバッグ情報の未削除指定 (-g)	235
10.2.15	デバッグ情報の削除指定 (-Xg)	236
10.2.16	CPU 情報ファイル指定 (-cif)	237
10.2.17	ターゲット CPU 指定 (-cpu)	238
10.2.18	オブジェクト混在チェックレベル指定 (-omcl)	239
第 11 章	ライブラリアンのリストフォーマット	241
11.1	リストファイルの情報内容	242
11.2	モジュール名リスト	243
11.3	モジュールごとの詳細情報	244
11.4	ライブラリ内の外部定義 / 参照シンボル情報	245
第 12 章	ライブラリアンの制限事項および Q&A	247
12.1	ライブラリアンの制限事項	248
12.2	ライブラリアンの使用上の Q&A	249
第 4 部	オブジェクト形式コンバータ編	251
第 13 章	オブジェクト形式コンバータの仕様	253
13.1	オブジェクト形式コンバータの概要	254
13.2	オブジェクト形式コンバータの種類	256
13.3	オブジェクト形式コンバータの実行	258
第 14 章	オブジェクト形式コンバータの共通オプション	259
14.1	オブジェクト形式コンバータのオプション一覧	260
14.2	出力ファイル名の変更 (-o)	261
14.3	パディング (-p)	263
第 15 章	ロードモジュールコンバータ (f2ms, f2hs, f2is, f2es)	265
15.1	ロードモジュールコンバータの概要	266
15.2	ロードモジュールコンバータのオプション一覧	267
15.3	ロードモジュールコンバータのオプション詳細	268
15.3.1	出力 S フォーマット指定 (-S1/-S2/-S3)	269
15.3.2	出力 HEX フォーマット指定 (-I16/-I20/-I32)	270
15.3.3	スタートアドレスレコード出力指定 (-entry)	271
15.3.4	スタートアドレスレコード出力抑止指定 (-Xentry)	273
15.3.5	整形指定 (-adjust)	274
15.4	f2ms(絶対形式ロードモジュール S フォーマット変換)	275
15.5	f2hs(絶対形式ロードモジュール HEX フォーマット変換)	276
15.6	f2is(絶対形式ロードモジュール HEX8 フォーマット変換), f2es(絶対形式ロードモジュール HEX16 フォーマット変換)	277
第 16 章	フォーマットアジャスタ (m2ms, h2hs)	279
16.1	フォーマットアジャスタの概要	280

16.2	フォーマットアジャスタのオプション一覧	283
16.3	フォーマットアジャスタのオプション詳細	284
16.3.1	出力レコード内データ長指定 (-len)	285
16.3.2	出力範囲指定 (-ran)	286
16.3.3	出力 S フォーマット指定 (-S1/-S2/-S3)	287
16.3.4	出力 HEX フォーマット指定 (-I16/-I20/-I32)	289
16.3.5	開始アドレス変更指定 (-ST)	290
第 17 章	バイナリコンバータ (m2bs, h2bs)	291
17.1	バイナリコンバータの概要	292
17.2	バイナリコンバータのオプション一覧	294
17.3	バイナリコンバータのオプション詳細	295
17.3.1	出力範囲指定 (-ran)	296
17.3.2	スプリットモード指定 (-sp)	297
17.3.3	スプリットモード抑止指定 (-Xsp)	298
17.3.4	マップリストファイルの作成指定 (-m)	299
17.3.5	マップリストファイルの作成抑止指定 (-Xm)	301
第 18 章	その他のコンバータ	303
18.1	m2is(S フォーマット HEX8 フォーマット変換)	304
18.2	m2es(S フォーマット HEX16 フォーマット変換)	305
18.3	i2ms(HEX8 フォーマット S フォーマット変換)	306
18.4	e2ms(HEX16 フォーマット S フォーマット変換)	307
第 19 章	オブジェクト形式コンバータの制限事項および Q&A	309
19.1	オブジェクト形式コンバータの制限事項	310
19.2	オブジェクト形式コンバータの使用上の Q&A	311
付録		313
付録 A	リンケージキットのエラーメッセージ	314
付録 B	HEX フォーマット	343
B.1	一般形式	344
B.2	データレコード (HEX8/HEX16/HEX32) タイプ : 00	346
B.3	エンドレコード (HEX8/HEX16/HEX32) タイプ : 01	347
B.4	拡張セグメントアドレスレコード (HEX16/HEX32) タイプ : 02	348
B.5	スタートセグメントアドレスレコード (HEX16/HEX32) タイプ : 03	349
B.6	拡張リニアアドレスレコード (HEX32) タイプ : 04	350
B.7	スタートリニアアドレスレコード (HEX32) タイプ : 05	351
付録 C	S レコード形式	352
C.1	S0 タイプ (ヘッダレコード)	353
C.2	S1 タイプ (データレコード : 2 バイトアドレス)	355
C.3	S2 タイプ (データレコード : 3 バイトアドレス)	356
C.4	S3 タイプ (データレコード : 4 バイトアドレス)	357
C.5	S5 タイプ (レコード数管理レコード)	358
C.6	S7 タイプ (ターミネータレコード)	359
C.7	S8 タイプ (ターミネータレコード)	360
C.8	S9 タイプ (ターミネータレコード)	361
付録 D	リンカのオプション一覧表	362
付録 E	ライブラリアンのオプション一覧表	364

付録 F	オブジェクト形式コンバータのコマンドおよびオプション一覧表	365
付録 G	OS による仕様の相違点	367
索引	369

第 1 部 リンケージキット編

リンケージキットに含まれるツールの概要と，ツール全体に共通する項目について説明します。

第 1 章 リンケージキットの仕様

第 2 章 オプション

第 3 章 共通オプション

第 4 章 オプションファイル

第1章

リンケージキットの仕様

この章では、リンケージキットに含まれるツールの概要、起動方法、終了方法および識別子などについて説明します。

1.1 リンケージキットの概要

1.2 起動方法

1.3 強制終了

1.4 終了コード

1.5 起動メッセージ

1.6 終了メッセージ

1.7 ヘルプメッセージ

1.8 識別子

1.9 ファイル名規則

1.10 環境変数

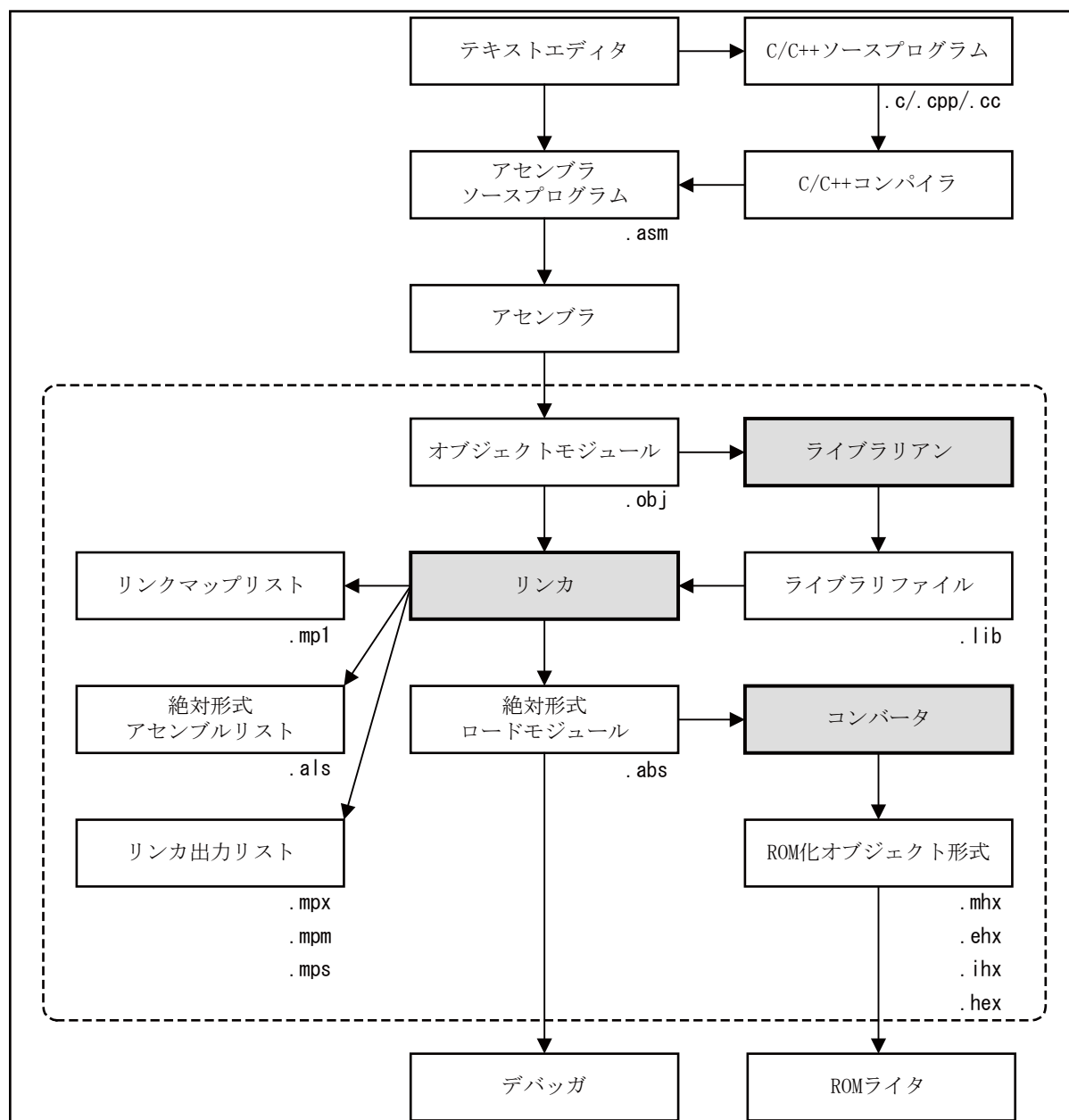
1.1 リンケージキットの概要

リンケージキットは、オブジェクトモジュールの結合に使用するリンカと、オブジェクトモジュールをまとめて管理するライブラリアン、その他、ROM書き込み用のオブジェクト形式へのコンバータで構成されています。

■ リンケージキットのサポート範囲

図 1.1-1 に、リンケージキットのサポート範囲を示します。

図 1.1-1 リンケージキットのサポート範囲



1.2 起動方法

リンケージキット (リンカ , ライブラリアン , オブジェクト形式コンバータ) を実行するための , コマンドラインの形式および指定方法について説明します。

■ コマンドラインの形式

SOFTUNE リンケージキットのコマンドラインの指定方法 (起動コマンドシンタックス) は , 次のとおりです。

- コマンドラインの指定方法 : コマンド名に続けて , ファイル名とオプションを必要
なだけ繰り返し指定する。

以下の説明では , オプションの指定をコマンド名の次にしてありますが , オプションの記述位置は , ファイル名の前後どちらでもかまいません。オプションについて , 詳しくは「第 2 章 オプション」を参照してください。

● リンカ

```
f1nk911s [ オプション ] ... <ファイル名> ...
```

<ファイル名> には , 入力するオブジェクトモジュールファイル名を指定します。複数のファイル名を指定するときは , スペースを入れます。

また , *.obj のようにワイルドカードの使用もできます。ファイル名のワイルドカードの展開は OS に依存しますので , 「付録 G OS による仕様の相違点」を参考にしてください。

リンカでは , -cpu オプションでターゲット CPU の指定が必要です。リンク処理実行の際は , 必ず -cpu オプションを指定してください。

● ライブラリアン

```
flibs [ オプション ] ... <ファイル名>
```

ライブラリアンは , SOFTUNE V6 で共通です。

<ファイル名> には , 編集対象のライブラリファイルを指定します。

ライブラリアンでは , -cpu オプションでターゲット CPU の指定が必要です。ライブラリ処理実行の際は , 必ず -cpu オプションを指定してください。

● オブジェクト形式コンバータ

```
コマンド名 [ オプション ] ... <ファイル名>
```

<ファイル名> には , それぞれのツールの機能に応じたオブジェクト形式のファイル名を指定します。以下の 3 形式のファイルが対象となります。

- リンカ出力の絶対形式ロードモジュール
- S フォーマット
- HEX フォーマット

1.3 強制終了

プログラムの実行を途中で止めたい場合は、CTRL キーを押しながら、C キーを押す（以降、"CTRL-C を押す" と記述します）ことにより行えます。CTRL-C が押されると、プログラムの実行が中断されます。

■ 強制終了

CTRL-C により、プログラム処理を中止した場合、出力結果のファイルは正しく作成されません。リンケージキットが実行時に使用する作業ファイルは消去されます。

1.4 終了コード

リンケージキットの各ツールは、その処理の終了状態を終了コードとして OS に返します。

■ 終了コードの値と終了状態

リンケージキットの各ツールは、その処理の終了状態（正常に終了したか、あるいはエラーが発生したかなどの状態）を終了コードとして OS に返します。終了コードと処理の終了状態の関係を、表 1.4-1 に示します。

表 1.4-1 終了コードと処理の終了状態

終了コード	処理の終了状態
0	正常に終了または警告レベルのエラーがあったとき
1	-cwno オプション指定時で、警告レベルのエラーがあったとき
2	正しい出力結果を作成できないようなエラーがあったとき
3	処理の続行が不可能となる致命的エラーがあったとき

1.5 起動メッセージ

リンケージキットは、-V オプションで起動メッセージの表示を行います。デフォルト処理では、起動メッセージの表示は行いません。

■ 起動メッセージと -V オプション

リンケージキットは、デフォルト処理では、処理中に検出されたエラーに関するメッセージを表示しますが、起動を示すメッセージは表示しません。起動時にメッセージ表示を行いたい場合は、-V オプションを用います。

また、-V オプションを無効にしたい場合は、-V オプションの後に -XV オプションを指定します。詳しくは「3.2.4 版数 / メッセージ出力指定 (-V)」、「3.2.5 版数 / メッセージ出力抑止 (-XV)」を参照してください。

■ 起動メッセージ

起動メッセージは、プログラム名と版数、著作権表示で構成されます。

以下に、起動メッセージを示します。

FR/FR80 Family SOFTUNE Linker V60Lxx ALL RIGHTS RESERVED, COPYRIGHT (C) FUJITSU MICROELECTRONICS LIMITED 1992-2008 LICENSED MATERIAL - PROGRAM PROPERTY OF FUJITSU MICROELECTRONICS LIMITED

1.6 終了メッセージ

リンケージキットは、`-cmsg` オプションで終了メッセージの表示を行います。デフォルト処理では、終了メッセージの表示は行いません。

■ 終了メッセージと `-cmsg` オプション

リンケージキットは、デフォルト処理では、処理中に検出されたエラーに関するメッセージを表示しますが、終了を示すメッセージは表示しません。終了時にメッセージ表示を行いたい場合は、`-cmsg` オプションを用います。

また、`-cmsg` オプションを無効にしたい場合は、`-cmsg` オプションより後に `-Xcmsg` オプションを指定します。詳しくは「3.2.6 終了メッセージ表示指定 (`-cmsg`)」、「3.2.7 終了メッセージ表示抑止指定 (`-Xcmsg`)」を参照してください。

■ 終了メッセージ

終了メッセージは、ツール名、エラーの有無を表示します。

以下に、終了メッセージ例を示します。

【エラーがない場合】

プログラム名 COMPLETED FOUND NO ERROR

【エラーがある場合】

プログラム名 COMPLETED FOUND ERROR

1.7 ヘルプメッセージ

ヘルプメッセージとして表示されるメッセージは、以下の 2 種類です。

- コマンドラインの記述形式
- 起動時オプションの一覧

■ ヘルプメッセージ

起動時にコマンド名以外何も指定しなかった場合、または `-help` オプションを指定したとき、コマンドラインの記述形式と起動時オプションの一覧を表示して終了します。詳しくは「3.2.3 ヘルプメッセージの表示 (`-help`)」を参照してください。

● ヘルプメッセージの例

図 1.7-1 に、リンカの場合 (英文) のヘルプメッセージの例を示します。

図 1.7-1 ヘルプメッセージの例

```
usage : flnk911s [-option ...] object[ object ...] *1
-----
options:

      ----- target CPU option -----
-cpu cpu-name      : Specify target CPU                (need)
      ----- linker mode options -----
-a                  : absolute linking mode              (absolute)
-r                  : relocatable linking mode
      ----- library options -----
-l filename[,...]  : specify library file name
                    :
                    :
```

*1: コマンドラインシンタックス (起動方法) の表示。

*2: オプションの一覧および、簡単な説明。

なお、メッセージは、環境変数 `FELANG` (「1.10.2 `FELANG` (メッセージ言語)」参照) 設定により、日本語で表示することも可能です。

1.8 識別子

リンケージキットでプログラム作成時などに扱う識別子には、以下の 7 種類があります。

- ファイル名
 - モジュール名
 - オプション名
 - セクション名
 - グループ名
 - ROM/RAM 領域名
 - シンボル名 (マングル名)
-

■ 識別子の構成文字種

識別子に使用可能な文字種は、以下のとおりです。

- 英字
- 数字
- アンダスコア (_)

先頭文字に数字の使用は許されません。

また、ファイル名に使用可能な文字の種類は、動作 OS に準拠します。したがって、ファイル名から作られるモジュール名も同等です。

■ 識別子の区別

英字の大文字小文字は区別されます。

■ 識別子文字数の制限

識別子の文字数制限は設けていません。

■ マングル名

C++ コンパイラが生成する関数名などの識別子は、関数の型情報などを表すための情報が付加されています。このことをマングルとよび、その識別子をマングル名とよびます。

リンカ内部で扱うラベルなどの識別子は、このマングル名を用いて指定を行います。

■ リスト出力時の識別名表示

リスト出力時にマングル名は、判り易いようにデコードを行って表示します。

リンケージキットが作成する各種リストファイル中では、識別子の名前全部を表示するようにはしていません。

長い識別子の名前は、先頭の 33 文字程度のみを出力し、残りの部分は表示していないものもあります。

1 行の中に表示される文字数は、リストのページ幅指定の変更で増減しますので、見やすいフォーマットを選択できます。

また、複数行にわたりますが、省略せずに表示するオプションも用意してあります。

1.9 ファイル名規則

入出力ファイルのファイル名は、OS での使用制限に準じています。
ファイル名は、オブジェクトモジュール中にも設定されますので、文字数やコード系に注意する必要がある場合があります。

■ ファイル名の文字数

入出力ファイルのファイル名は、OS での使用制限に準じています。

■ ファイル名の文字コード

C/C++ やアセンブラのソースファイル名は、出力されるオブジェクトモジュール内に、ソ - スファイル名情報として設定されるほか、モジュール名としても設定されます。

モジュール名には、「1.8 識別子」で示されるように英字、数字、アンダスコア (_) しか使用できません。したがって、漢字や空白を用いたファイル名では、アセンブル時に、モジュール名を指定するなどの工夫が必要です。

● ファイル名として使用できる文字 (Windows 版)

英字、数字、仮名文字、シフト JIS 漢字コード、および下記以外の記号が使用できます。

`\ / : ; , * ? " < > |`

空白を含むファイル名を指定する場合には、ファイル名を二重引用符 (") で括ってください。

環境変数に空白を含むディレクトリ名を指定する場合には、二重引用符 (") で括らないでください。

1.10 環境変数

リンケージキットでは、以下の 6 種類の環境変数をサポートしています。

- TMP
 - FELANG
 - FETOOL
 - LIB911
 - OPT911
 - OPT
-

■ TMP (ワークディレクトリ)

TMP は、作業用ディレクトリを指定します。詳しくは「1.10.1 TMP (ワークディレクトリ)」を参照してください。

■ FELANG (メッセージ言語)

FELANG は、メッセージの言語の選択、指定を行います。詳しくは「1.10.2 FELANG (メッセージ言語)」を参照してください。

■ FETOOL (インストールディレクトリ)

FETOOL は、開発ツールをインストールしたディレクトリを指定します。詳しくは「1.10.3 FETOOL (インストールディレクトリ)」を参照してください。

■ LIB911 (ライブラリファイル検索ディレクトリ)

LIB911 は、ライブラリを格納したディレクトリを指定します。詳しくは、「1.10.4 LIB911 (ライブラリファイル検索ディレクトリ)」を参照してください。

■ OPT911 (デフォルトオプションファイル格納ディレクトリ)

OPT911 は、リンカのデフォルトオプションファイルを格納したディレクトリを指定します。詳しくは、「1.10.5 OPT911 (デフォルトオプションファイル格納 ディレクトリ)」を参照してください。

■ OPT (デフォルトオプションファイル格納ディレクトリ)

OPT は、ライブラリアンおよびオブジェクトツールのデフォルトオプションファイルを格納したディレクトリを指定します。詳しくは、「1.10.6 OPT (デフォルトオプションファイル格納 ディレクトリ)」を参照してください。

1.10.1 TMP (ワークディレクトリ)

TMP は、リンケージキットが実行時に使用する作業用のディレクトリを指定します。
TMP の記述形式、説明、指定例を示します。

■ TMP (ワークディレクトリ)

【記述形式】

SET TMP= <パス名>

【説明】

リンケージキットが実行時に使用する作業用のディレクトリを指示します。

この環境変数 TMP は、他の開発ツール (C/C++ コンパイラやアセンブラなど) でも使用されます。

環境変数 TMP の設定がないときはカレントディレクトリを使用します。

【例】

```
SET TMP=G:\WORK
```

1.10.2 FELANG (メッセージ言語)

FELANG は、ヘルプメッセージ、およびエラーメッセージ表示のメッセージ言語の選択、指定を行います。

FELANG の記述形式、説明、指定例を示します。

■ FELANG (メッセージ言語)

【記述形式】

SET FELANG={ASCII EUC SJIS}

ASCII: 英語 ASCII コード (デフォルト)

EUC: 日本語 EUC コード

SJIS: 日本語 SJIS コード

【説明】

ヘルプメッセージおよびエラーメッセージ表示の日本語 / 英語 (メッセージ言語) の選択、指定を行います。

指定がない場合は、英語のメッセージ (ASCII 指定) が選択されます。日本語環境のないシステムで、EUC または SJIS 以外のコード系をご使用の場合には、FELANG 環境変数の設定を行わないか、ASCII を指定してください。

この環境変数 FELANG は、ほかの開発ツール (C/C++ コンパイラやアセンブラなど) でも使用されます。

【例】

```
SET FELANG=ASCII
```

1.10.3 FETOOL(インストールディレクトリ)

FETOOL は、リンケージキットをインストールした親ディレクトリを指定します。
FETOOL の記述形式、説明、指定例などを示します。

■ FETOOL (インストールディレクトリ)

【記述形式】

```
SET FETOOL= <パス名>
```

<パス名> には、ドライブ名を含めて指定してください。

【説明】

リンケージキットをインストールしたディレクトリを指定します。

リンケージキットは、ここで指定したディレクトリを起点として、メッセージファイルやライブラリファイルの格納ディレクトリを知り、実行に必要なファイルをアクセスします。

指定がない場合は、実行されたロードモジュールのあるディレクトリの親ディレクトリとなります。

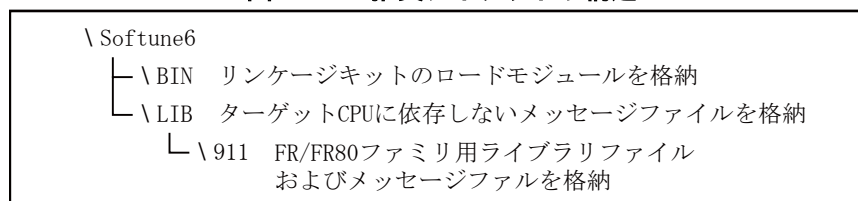
この環境変数 FETOOL は、ほかの開発ツール (C/C++ コンパイラやアセンブラなど) でも使用されます。

【例】

```
SET FETOOL=C:\Softtune6
```

【推奨ディレクトリ構造】

図 1.10-1 推奨ディレクトリ構造



【補足】

リンケージキットは、上記のようなディレクトリ構造に各ファイルを格納して実行されることを前提に作成されています。

FETOOL 環境変数は、"SOFTTUNE" ディレクトリのパスをリンケージキットに知らせるものです。

1.10.4 LIB911(ライブラリファイル検索ディレクトリ)

LIB911 (ライブラリファイル検索ディレクトリ) は、リンカが検索するライブラリファイルや CPU 情報ファイルを格納したディレクトリを指定します。

LIB911 の記述形式、説明、指定例などを示します。

■ LIB911 (ライブラリファイル検索ディレクトリ)

【記述形式】

```
SET LIB911= <パス名> [ ; <パス名> ... ]
```

<パス名> には、ドライブ名を含めて指定してください。

【説明】

リンカが検索するライブラリファイルや CPU 情報ファイルを格納したディレクトリを指定します。

通常は、C/C++ ライブラリを格納したディレクトリを指定しておきます。

複数の検索パスを指定する場合は、<パス名> を次の記号で区切ります。

- セミコロン (;)

複数指定時の検索順は、指定した順となります。

【例】

```
SET LIB911=C:\Softune6\LIB\11
```

【補足】

環境変数 FETOOL が指定されている場合、前項で説明したディレクトリ構造のライブラリ格納ディレクトリも検索されますので、環境変数 LIB911 の設定を行わなくても C/C++ ライブラリは検索されます。

ライブラリ検索パスは、リンカ実行時のオプション -L でも指定できます。

これらの複合指定によるライブラリ検索パスの優先順位は次のようになります。

- 1) リンカのオプション -L で指定されたディレクトリ
- 2) 環境変数 LIB911 で指定されたディレクトリ
- 3) 環境変数 FETOOL より導かれるディレクトリ (%FETOOL%\LIB\911)

ユーザがライブラリを作成した場合などは、C/C++ライブラリとの検索順に注意してパスの指定を行ってください。

1.10.5 OPT911(デフォルトオプションファイル格納ディレクトリ)

OPT911 (デフォルトオプションファイル格納ディレクトリ) は , リンカのデフォルトオプションファイルを格納したディレクトリを指定します。

OPT911 の記述形式 , 説明 , 指定例などを示します。

■ OPT911 (デフォルトオプションファイル格納ディレクトリ)

【記述形式】

`SET OPT911= <パス名>`

<パス名> には , ドライブ名を含めて指定してください。

【説明】

リンカで使用するデフォルトオプションファイルを格納したディレクトリを指定します。

この環境変数は省略できます。

省略した場合は , 開発環境ディレクトリ内のデフォルトオプションファイルを参照します。

開発環境ディレクトリ内のデフォルトオプションファイルは次のとおりです。

● リンカ

`%FETOOL%\LIB\911\FLNK911.OPT`

【例】

`SET OPT911=C:\Softune6\LIB\911`

1.10.6 OPT(デフォルトオプションファイル格納ディレクトリ)

OPT (デフォルトオプションファイル格納ディレクトリ) は , ライブラリアン , オブジェクトツールのデフォルトオプションファイルを格納したディレクトリを指定します。

OPT の記述形式 , 説明 , 指定例などを示します。

■ OPT (デフォルトオプションファイル格納ディレクトリ)

【記述形式】

```
SET  OPT= <パス名>
```

<パス名> には , ドライブ名を含めて指定してください。

【説明】

ライブラリアンや , オブジェクトツールで使用するデフォルトオプションファイルを格納したディレクトリを指定します。

この環境変数は省略できます。

省略した場合は , 開発環境ディレクトリ内のデフォルトオプションファイルを参照します。

開発環境ディレクトリ内のデフォルトオプションファイルは次のとおりです。

● ライブラリアン

- %FETOOL%\LIB\FLIB.OPT

● オブジェクトツール

- %FETOOL%\LIB\F2M.OPT
- %FETOOL%\LIB\F2H.OPT
- %FETOOL%\LIB\M2B.OPT
- %FETOOL%\LIB\M2M.OPT
- %FETOOL%\LIB\H2B.OPT
- %FETOOL%\LIB\H2H.OPT
- %FETOOL%\LIB\F2I.OPT
- %FETOOL%\LIB\F2E.OPT
- %FETOOL%\LIB\M2I.OPT
- %FETOOL%\LIB\M2E.OPT
- %FETOOL%\LIB\I2M.OPT
- %FETOOL%\LIB\E2M.OPT

【例】

```
SET  OPT=C:\Softune6\LIB
```

第 1 章 リンケージキットの仕様

第2章

オプション

この章では、リンケージキットのオプションについて説明します。

- 2.1 オプション
- 2.2 オプションパラメータでの数値表現
- 2.3 オプション指定時の注意と評価
- 2.4 包含 / 相反関係にあるオプションの指定
- 2.5 コマンドラインの指定例

2.1 オプション

オプションは、オプション名とパラメータで構成されます。オプションの形式や指定方法について説明します。

■ オプション形式

オプションの形式は以下のとおりです。

- オプション名 [パラメータ] ...

オプション名の先頭には、ハイフン (-) を付加します。

オプション名とパラメータの区切りにはスペースを置きます。

パラメータの有無および形式は、オプションごとに定義されていますので、個々のオプションの説明を参照してください。

また、オプション指定時には、以下の点に注意してください。

- オプション名の英字は、大文字と小文字の区別を行います。
- パラメータが必要なオプションは、パラメータを省略することはできません。
- 複数のオプションを指定する場合、例えば -a と -V を -aV のようにまとめて指定することはできません。
- ハイフンとオプション名の間にスペースを置くことはできません。

■ パラメータ

パラメータは、オプションの操作対象となるファイル名やモジュール名などを指定するものです。2 つ以上のパラメータを記述する場合には、通常カンマ (,) で区切りますが複雑なパラメータ指定では、カンマ以外の記号も使用しますので、詳細は個々のオプションの説明を参照してください。

【例】

```
-a  
gets.obj, puts.obj, getc.obj, putc.obj  
-sc CODE=0xC1000, DATA=0x1000
```

2.2 オプションパラメータでの数値表現

オプションパラメータでの数値表現には、10 進数と 16 進数が使えます。

■ オプションパラメータでの数値表現

オプションパラメータの数値の先頭が (0x) で始まるものは 16 進数、そうでないものは 10 進数とみなします。16 進表現の a ~ f は、大文字と小文字の区別なくどちらでも使用できます。

【例】

```
0x100      ... 16 進表現 (= 256)
100         ... 10 進表現 (= 0x64)
0xff と 0xFF は同じ。
```

2.3 オプション指定時の注意と評価

オプションを指定するとき、重複指定や指定順序に注意が必要なものがあります。リンケージキットでは、一定の規則に従ってオプションの評価をします。

■ オプション指定時の注意と評価

オプション指定時の注意と評価の規則を、以下に示します。

● パラメータの不要なオプション

1 回だけ指定があれば良く、何度指定しても同じです。

【例】

`-V` : メッセージ出力の指定

`-V -V` と何度指定してもエラーではありません。

● パラメータが必要なオプション

複数回のオプション指定があるとき、評価方法が異なります。

- 最後の指定 1 つだけが有効なもの
- 指定順に意味があり、すべての指定が有効なもの
- 指定順に意味がなく、すべての指定が有効なもの

【例 1 最後の指定 1 つだけが有効なもの】

`-o file.abs` : 出力ファイル名の指定

`-o file.abs -o file.rel` のように複数回指定した場合には、後指定が有効になります (この例の場合 `file.rel` が有効)。

【例 2 指定順に意味があり、すべての指定が有効なもの】

`-l lib1.lib -l lib2.lib` : 検索ライブラリ指定 (リンカ)

`-l lib2.lib -l lib1.lib` とすると、ライブラリ検索の順番が逆になります。

【例 3 指定順に意味がなく、すべての指定が有効なもの】

`-sc CODE=0x1000 -sc DATA=0x200` : セクション配置指定 (リンカ)

`-sc DATA=0x200 -sc CODE=0x1000` としても、セクション配置は個々に独立のため同等の指定となります。

2.4 包含 / 相反関係にあるオプションの指定

ほかのオプションと包含関係にあるものは、上位のオプション指定が有効になります。また、ほかのオプションと相反関係にあるものは、後指定が有効になります。

■ 包含関係にあるオプションの指定例

【例】

-Xm -pw 80 : リスト出力抑止とページ幅の指定

-pw オプションは、リスト出力を行うときのみ有効であるため、-Xm (リスト出力抑止) オプションにより、指定自体が意味のないものとなります。

-pw 80 -Xm と指定順が逆でも同じです。

■ 相反関係にあるオプションの指定例

ほかのオプションと相反関係にあるものは、後指定が有効になります。

【例 1】

-a -r

絶対形式出力指定と相対形式出力指定 (リンカ) -r が有効となります。

【例 2】

-m mapfile -Xm

リストファイル名指定 -m はリスト出力抑止 -Xm で取り消され、リスト出力は行われません。

2.5 コマンドラインの指定例

コマンドライン指定例を3種類示し、説明します。

■ コマンドラインの指定例

【例1】

```
flnk911s  
flnk911s file1.obj file2.obj -g -a -help
```

コマンド名だけの指定の場合、もしくは、途中で、オプションがわからなくなったときなど、-help オプション HELP を指定すると、簡単なヘルプメッセージを表示します。

【例2】

```
flibs sys.lib -m sys.mp2 ... *1  
flibs -m sys.mp2 sys.lib ... *2
```

オプションの指定位置は決まっていないので、コマンドラインで自由に記述できます。

*1 と *2 の指定はどちらも有効であり、同等の指定です。

【例3】

```
flnk911s *.obj -g -o sample.abs  
flnk911s '*.obj' -g -o sample.abs
```

ワイルドカードを使用した、複数入力ファイル名の指定。

< 注意事項 >

UNIX 系 OS では、コマンドラインに記述されたワイルドカードの展開がシェルで行われてしまい、思わぬ結果 (オプション指定) となる場合があります。

このようなことを避けるため、UNIX 系 OS でワイルドカードを含むオプションを指定する際はシングルクォート (') で括って指定を行ってください。

第3章

共通オプション

リンケージキットには、どのツールにも共通して使用できるオプションがあります。これらのオプションは、C/C++ コンパイラやアセンブラにも用意されています。

この章では、リンケージキットの共通オプションについて説明します。また、ツール固有のオプションについては、それぞれの部で説明します。

3.1 共通オプション一覧

3.2 共通オプション詳細

3.1 共通オプション一覧

ここでは、リンケージキットで共通して使用できるオプション一覧を示します。

■ 共通オプション一覧

表 3.1-1 に、リンケージキットで共通に指定できるオプション一覧を示します。

表 3.1-1 共通オプション一覧

機 能	オプション	備 考
デフォルトオプションファイル読み込み抑止指定	-Xdof	
オプションファイル名指定	-f	
ヘルプメッセージの表示指定	-help	
プログラムの版数、起動メッセージ表示の指定	-V	
プログラムの版数、起動メッセージの出力抑止	-XV	デフォルト
終了メッセージの表示指定	-cmsg	
終了メッセージの出力抑止	-Xcmsg	デフォルト
ワーニング発生時の終了コードを 1 にする指定	-cwno	
ワーニング発生時の終了コードを 0 にする指定	-Xcwno	デフォルト

3.2 共通オプション詳細

ここでは、リンケージキットで共通して使用できる個々のオプションについて説明します。

■ -Xdof オプション

-Xdof オプションは、デフォルトオプションファイルの読み込みを取り消します。詳しくは、「3.2.1 デフォルトオプションファイル抑止指定 (-Xdof)」を参照してください。

■ -f オプション

-f オプションでは、オプションを記述したファイルから、オプションを読み込むことを指示します。詳しくは、「3.2.2 オプションファイルからの読み込み指定 (-f)」を参照してください。

■ -help オプション

-help オプションでは、ヘルプメッセージを表示することを指示します。詳しくは、「3.2.3 ヘルプメッセージの表示 (-help)」を参照してください。

■ -V オプション

-V オプションでは、プログラムの起動時のメッセージを出力します。なお、このメッセージは、デフォルト処理実行では表示されません。詳しくは、「3.2.4 版数 / メッセージ出力指定 (-V)」を参照してください。

■ -XV オプション

-XV オプションでは、起動時のメッセージの出力を抑止することを指示します。詳しくは、「3.2.5 版数 / メッセージ出力抑止 (-XV)」を参照してください。

■ -cmsg オプション

プログラムの終了メッセージを表示します。詳しくは、「3.2.6 終了メッセージ表示指定 (-cmsg)」を参照してください。

■ -Xcmsg オプション

プログラムの終了メッセージの表示を抑止します。詳しくは、「3.2.7 終了メッセージ表示抑止指定 (-Xcmsg)」を参照してください。

■ -cwno オプション

プログラムでワーニングが発生した場合に、終了コードとして、OS に "1" を返します。詳しくは、「3.2.8 ワーニング発生時の終了コードを 1 にする指定 (-cwno)」を参照してください。

■ -Xcwno オプション

プログラムでワーニングが発生した場合に、終了コードとして、OS に "0" を返します。詳しくは、「3.2.9 ワーニング発生時の終了コードを 0 にする指定 (-Xcwno)」を参照してください。

3.2.1 デフォルトオプションファイル抑止指定 (-Xdof)

デフォルトオプションファイルの読み込みを取り消します。
本オプションの指定がない場合、デフォルトオプションファイルは常に読み込まれます。

■ デフォルトオプションファイル抑止指定 (-Xdof)

【記述形式】

-Xdof

【パラメータ】

なし

【説明】

デフォルトオプションの読み込みを取り消します。

本オプションの指定がない場合、デフォルトオプションファイルは常に読み込まれます。

デフォルトオプションファイルに関しては、「4.5 デフォルトオプションファイル」を参照してください。

【注意】

本オプションは、コマンドラインでの指定のみ有効です。

【例】

```
flnk911s test.obj -Xdof -cpu MB91101
```

3.2.2 オプションファイルからの読み込み指定 (-f)

-f オプションは、オプションを記述したファイルからオプションを読み込むことを指示します。また、このファイル内容を、コマンドラインに指定されたものと同等に扱います。

■ オプションファイルからの読み込み指定 (-f)

【記述形式】

```
-f <オプションファイル名>
```

【パラメータ】

<オプションファイル名>

オプションや入力ファイルを記述したファイル名

【説明】

<オプションファイル名> で指定したファイルに、オプションや入力ファイル名を記述しておきます。

オプションを記述したファイルから、オプションを読み込むことを指示します。

このファイル内容を、コマンドラインでの指定と同様に評価し、処理します。

ファイル名の拡張子は、デフォルトで決められたものではありません。

【注意】

オプションファイル内では、-f オプション自身の指定はできません。

【例 1】

```
f2ms -V -f optfile.f2m
optfile.f2m の内容
```

```
#
# from FJ-OMF to S format
#
cpp903.abs      # IN  ABS-LM
-o cpp903.mhx   # OUT S format
```

これは、以下のようにコマンドラインで指定したのと同等です。

```
f2ms -V ccp903.abs -o ccp903.mhx
```

【例 2】

```
flibs syslib.lib -f objfile.opt
```

syslib.lib に登録するモジュールを objfile.opt に記述し、ライブラリアンはこのファイル内容を参照してライブラリファイルを作成します。

objfile.opt の内容は、例えば以下ようになります。

```
-a  
putc.obj, getc.obj, puts.obj, gets.obj,  
memchr.obj, strcat.obj, strerr.obj, strpbrk.obj,  
strchr.obj, strcmp.obj, strcpy.obj, strlen.obj
```

ライブラリ名指定まで含め、次のように指定できます。

```
flibs -f libfile.opt
```

この場合、libfile.opt の内容は、以下ようになります。

```
syslib.lib  
-a  
putc.obj, getc.obj, puts.obj, gets.obj,  
memchr.obj, strcat.obj, strerr.obj, strpbrk.obj,  
strchr.obj, strcmp.obj, strcpy.obj, strlen.obj
```

オプションファイルを2回指定することも可能です。

```
flibs syslib.lib -f objgr1.opt -f objgr2.opt
```

objgr1.opt および objgr2.opt の内容は、例えば以下ようになります。

objgr1.opt の内容

```
-a putc.obj, getc.obj, puts.obj, gets.obj
```

objgr2.opt の内容

```
-a memchr.obj, strcat.obj, strerr.obj, strpbrk.obj,  
strchr.obj, strcmp.obj, strcpy.obj, strlen.obj
```


3.2.3 ヘルプメッセージの表示 (-help)

-help オプションは、プログラムを実行せずにヘルプメッセージを表示することを指示します。コマンドラインの指定形式と、オプションの概要を表示します。

■ ヘルプメッセージの表示 (-help)

【記述形式】

-help

【パラメータ】

なし

【説明】

-help オプションは、コマンドラインの指定形式とオプション一覧を簡単に表示します。

ヘルプメッセージは標準出力 (stdout) に出力されます。

コマンド名だけの指定時にも、同じヘルプメッセージが出力されます。

入力ファイル名や他のオプションが指定されている場合にも、本オプションが指定されていると、プログラムを実行せずにヘルプメッセージのみ表示します。

3.2.4 版数 / メッセージ出力指定 (-V)

-V オプションでは、プログラムの起動時のメッセージの出力を行います。

■ 版数 / メッセージ出力指定 (-V)

【記述形式】

-V

【パラメータ】

なし

【説明】

-V オプションは、起動メッセージの出力を行うようにする指定です。なお、リンケージキットの各ツールは、デフォルト実行では起動メッセージを出力しません。起動メッセージ出力は、必ずこの -V オプションを使って行います。

起動メッセージには、プログラム版数、著作権表示などが含まれます。

メッセージは標準出力 (stdout) に出力します。

【例 1】

```
flnk911s ccp903
```

本オプションを指定しない場合、実行開始時にメッセージを出力しません。

終了時は、OS のプロンプトが出て、次のコマンド入力待ちになるだけです。

【例 2】

```
f2ms ccp903 -V
```

```
SOFTUNE FJ-OMF to S-FORMAT Converter V60L02
```

```
ALL RIGHTS RESERVED, COPYRIGHT (C) FUJITSU LIMITED 1992
```

```
LICENSED MATERIAL - PROGRAM PROPERTY OF FUJITSU LIMITED
```

実行開始時に、起動メッセージ (プログラム名、版数、コピーライト) を表示します。

【例 3】

```
flibs -V
```

-V オプションだけが指定された場合、プログラム名、プログラム版数、コピーライトメッセージを表示したのち、直ちにプログラムを終了します。

3.2.5 版数 / メッセージ出力抑止 (-XV)

-XV オプションは、-V オプションを無効にします。これにより、プログラムの起動メッセージの出力を行わないようにします。

■ 版数 / メッセージ出力抑止 (-XV)

【記述形式】

-XV

【パラメータ】

なし

【説明】

リンケージキットの各ツールは、デフォルト実行では起動メッセージを出力しませんので、起動メッセージを出力させるには -V オプションを指定します。

-XV オプションは、-V 指定を無効にするときに指定します。

【例 1】

```
flnk911s ccp903
flnk911s ccp903 -XV
```

デフォルト処理では、実行開始時にメッセージを出力しません。

上記指定はどちらも同じです。

【例 2】

```
f2ms -f lkit.opt ccp903 -XV
```

オプションファイルを利用した実行時、オプションファイル内の指定を一時的に変更したい場合があります。lkit.opt 内に -V オプションがある場合、lkit.opt の内容を変更せずコマンドライン上で -XV 指定を行えば、-V オプションを取り消すことができます。

3.2.6 終了メッセージ表示指定 (-cmsg)

終了メッセージを表示します。

■ 終了メッセージ表示指定 (-cmsg)

【記述形式】

-cmsg

【パラメータ】

なし

【説明】

プログラムの終了メッセージを表示します。

リンケージキットはデフォルトでは、プログラムの終了メッセージの表示を行いません。

【例1】

```
flnk911s ccp903
```

本オプションを指定しない場合、終了時にメッセージを出力しません。

終了時は、OSのプロンプトが出て、次のコマンド入力待ちになるだけです。

【例2】

```
f2ms ccp903 -cmsg
```

```
F2MS COMPLETED FOUND NO ERROR
```

終了時に、終了メッセージ(プログラム名、エラーの有無)を表示します。

3.2.7 終了メッセージ表示抑止指定 (-Xcmsg)

終了メッセージの表示を抑止します。

■ 終了メッセージ表示抑止指定 (-Xcmsg)

【記述形式】

-Xcmsg

【パラメータ】

なし

【説明】

終了メッセージの表示を抑止します。

リンケージキットは、デフォルトでプログラムの終了メッセージの表示を行いません。本オプションはプログラムの終了メッセージの表示オプション(-cmsg)を取り消す場合に使用します。

【例1】

```
flnk911s ccp903
flnk911s ccp903 -Xcmsg
```

デフォルト処理では、終了時にメッセージを出力しません。

上記指定はどちらも同じです。

【例2】

```
f2ms -f lkit.opt ccp903 -Xcmsg
```

オプションファイルを利用した実行時、オプションファイル内の指定を一時的に変更したい場合があります。lkit.opt 内に -cmsg オプションがある場合、lkit.opt の内容を変更せずコマンドライン上で -Xcmsg 指定を行えば、-cmsg オプションを取り消すことができます。

3.2.8 ワーニング発生時の終了コードを 1 にする指定 (-cwno)

プログラム実行時にワーニングのみが発生した場合の終了コードを 1 に変更します。

■ ワーニング発生時の終了コードを 1 にする指定 (-cwno)

【記述形式】

-cwno

【パラメータ】

なし

【説明】

プログラム実行時にワーニングのみが発生した場合の終了コードを "1" に変更します。
SOFTUNE リンケージキットでは、通常ワーニングのみが発生した場合の終了コードは "0" です。

【例 1】

```
flnk911s ccp903 -cwno
```

プログラム実行時にワーニングのみ発生した場合、OS への終了コードが "1" になります。

【例 2】

```
flnk911s ccp903
```

プログラム実行時にワーニングのみの発生の場合、OS への終了コードはデフォルトである "0" のままです。

3.2.9 ワーニング発生時の終了コードを 0 にする指定 (-Xcwno)

プログラム実行時にワーニングのみが発生した場合の終了コードを 0 に変更します。

■ ワーニング発生時の終了コードを 0 にする指定 (-Xcwno)

【記述形式】

-Xcwno

【パラメータ】

なし

【説明】

プログラム実行時にワーニングのみが発生した場合の終了コードをデフォルトである "0" に戻します。

SOFTUNE リンケージキットでは、通常ワーニングのみが発生した場合の終了コードは "0" です。

本オプションはワーニング発生時の終了コードを "1" にするオプション (-cwno) を取り消す場合に使用します。

【例 1】

```
flnk911s ccp903
```

```
flnk911s ccp903 -Xcwno
```

デフォルト処理では、ワーニング発生時の終了コードは "0" です。

上記指定はどちらも同じです。

【例 2】

```
f2ms -f lkit.opt ccp903 -Xcwno
```

オプションファイルを利用した実行時、オプションファイル内の指定を一時的に変更したい場合があります。lkit.opt 内に -cwno オプションがある場合、lkit.opt の内容を変更せずコマンドライン上で -Xcwno 指定を行えば、-cwno オプションを取り消すことができます。

第4章

オプションファイル

この章では、リンケージキットのオプションファイルについて説明します。

- 4.1 オプションファイルの概要
- 4.2 オプションファイル中での継続指定
- 4.3 オプションファイル中のコメント指定
- 4.4 オプションファイルの記述例
- 4.5 デフォルトオプションファイル

4.1 オプションファイルの概要

オプションファイルは、コマンドライン入力を簡略化するために、あらかじめ処理に必要なファイル名やオプションを記述しておくファイルです。

■ オプションファイル

オプションファイルは、コマンドラインで指定する入力ファイル名やオプションを記述したファイルです。

記述シンタックスは、コマンドライン上での指定方法と同じです。

ただし、オプションファイル中では、次の2項目が加わります。

- コメント文が記述できる。
- 任意区切り位置で改行可能である。

コメント文は、コメント記号(#)で始まり、改行で終了します。

コメント文および改行記号は、コマンドライン上でのスペースと同等に扱われます。

■ オプションファイル指定による実行

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、指定するファイル名やオプションが多い場合には指定しきれなくなります。また、毎回そのような指定を行うことは、効率の低下と入力ミスによる誤動作の原因になります。

処理が定型化してきたり、指定するオプションやファイル名が多かったりしたとき、毎回コマンドライン入力を行う手間を省くために、ファイルに記述された内容を、コマンドラインでの指定と同等に扱い処理できます。その際、処理に必要なファイル名やオプションを、テキストエディタを使いオプションファイルに作成しておき、-fオプションを用いて実行します。

【例】

```
flibs -f optfile
```

オプションファイル "optfile" の内容

```
prg.lib
-a main.obj
-a send.obj, receive.obj, exchange.obj
-a account.obj
-m prg.mp2
```

オプションファイル内の記述形式は、上記の例のようにコマンドラインでの記述形式と同じです。上記の例ではオプションを各行に分けていますが、以下のように1行で記述してもかまいません。

```
prg.lib -a main.obj ... -a account.obj -m prg.mp2
```

この例では、オプションだけではなく、編集対象のライブラリファイル(prg.lib)の記述も行っています。

このように、オプションファイル中には、(-fオプション、-Xdofオプションの指定を除き)コマンドラインで記述することのできるすべての指定が同じ形式で記述できます。

4.2 オプションファイル中での継続指定

オプションファイル中では、オプションやパラメータの区切りの箇所で改行し、継続して指定を行うことができます。

■ オプションファイル中の継続指定

オプションファイルにオプションやファイル名を記述していると、オプションが1行に記述しきれなかったり、見やすさのため複数行に渡って記述したりしたいと思うことがあります。

このため、オプションファイル中では、オプションやパラメータの区切りの箇所で改行できるようにしています。以下に、オプションファイルの記述例を2つ示します。

【例1 オプションファイルの内容を1行で記述した場合の例】

```
      :  
-a mod01, mod02, obj03, obj04  
      :
```

【例2 オプションファイルの内容を2行で記述した場合の例】

```
      :  
-a mod01, mod02,  
  obj03, obj04   継続行  
      :
```

4.3 オプションファイル中のコメント指定

オプションファイル中には、コメントを入れることができます。

■ オプションファイル中のコメント指定

オプションファイル中にコメントを入れる場合、コメント開始記号には (#) を使用します。

【例 オプションファイルの内容にコメントを入れた場合の例】

下線部分がコメントです。

```
# Example of Library Options
syslib.lib      # INDICATES LIBRARY FILE
-a mod01, mod02, obj03, obj04 # Add Modules
```

4.4 オプションファイルの記述例

オプションファイルの記述例として、コマンドラインでの
-a mod01,mod02,obj03,obj04 と等価に扱われるオプションファイル中での指定方法を以下に示します。

■ オプションファイルの記述例

<code>-a mod01,mod02,obj03,obj04</code>	同じ
<code>-a mod01, mod02 ,obj03 , obj04</code>	, の前後にスペース挿入
<code>-a mod01,mod02,obj03,obj04 # comment</code>	文末にコメントを付加
<code># comment line -a mod01,mod02,obj03,obj04</code>	コメント行を追加
<code>-a mod01,mod02,obj03, obj04</code>	, の後ろで改行 パラメータを継続
<code>-a mod01,mod02,obj03 ,obj04</code>	, の前で改行 パラメータを継続
<code>-a mod01,mod02,obj03, # comment obj04</code>	コメント挿入 パラメータを継続
<code>-a mod01,mod02,obj03,obj04</code>	-a の後ろで改行 パラメータ全部を継続

4.5 デフォルトオプションファイル

オプションファイル機能の1つですが、起動時オプション "-f" を指定しなくてもあらかじめ指定されているオプションファイルを読み込み実行します。
この機能をデフォルトオプションファイルといいます。

■ デフォルトオプションファイル

デフォルトオプションファイルは、オプションファイル機能の1つですが、起動時オプション "-f" を指定しなくてもあらかじめ指定されているオプションファイルを読み込み実行します。この機能をデフォルトオプションファイルといいます。

デフォルトオプションファイルは、常にプログラムの起動時に読み込まれますので、ユーザの環境に適した起動時オプションをあらかじめ指定しておくことができます。

デフォルトオプションファイル機能を抑止するには、起動時オプション "-Xdof" を指定します。

このオプションが指定されると、デフォルトオプションファイルは読み込まれません。
デフォルトオプションファイル名は、表 4.5-1 で示すように決まっています。

表 4.5-1 リンケージキットのデフォルトオプションファイル名

ツール名	プログラム名	オプションファイル名
リンカ	flnk911s	flnk911.opt
ライブラリアン	flibs	flib.opt
オブジェクト形式 コンバータ	f2ms	f2m.opt
	f2hs	f2h.opt
	m2bs	m2b.opt
	m2ms	m2m.opt
	h2bs	h2b.opt
	h2hs	h2h.opt
	f2is	f2i.opt
	f2es	f2e.opt
	m2is	m2i.opt
	m2es	m2e.opt
	i2ms	i2m.opt
	e2ms	e2m.opt

デフォルトオプションファイルの参照手順を次に示します。

● 環境変数 "OPT911" または "OPT" が設定されている場合

環境変数で設定されているディレクトリのファイルを参照します。

• リンカ

%OPT911%\ デフォルトオプションファイル

• ライブラリアン、オブジェクトツール

%OPT%\ デフォルトオプションファイル

● 環境変数 "OPT911" または "OPT" が設定されていない場合

開発環境ディレクトリ内のデフォルトオプションファイルを参照します。

- リンカ

`%FETOOL%\LIB\911\ デフォルトオプションファイル`

- ライブラリアン, オブジェクトツール

`%FETOOL%\LIB\ デフォルトオプションファイル`

< 注意事項 >

デフォルトオプションファイルが見つからない場合, リンケージキットはエラーメッセージを表示しません。

第 2 部 リンカ編

リンカの仕様 , オプション , および出力リストなどについて説明します。

第 5 章 リンカの仕様

第 6 章 リンカのオプション

第 7 章 リンカの出力リストファイル

第 8 章 リンカの制限事項および Q&A

第5章

リンカの仕様

この章では、リンカの概要とリンカの各機能について説明します。

- 5.1 リンカの概要
- 5.2 リンカの機能
- 5.3 セクションの種類
- 5.4 セクションの結合
- 5.5 セクションの配置
- 5.6 セクションの自動配置
- 5.7 ライブラリの検索
- 5.8 ROM/RAM 領域
- 5.9 ROM RAM 転送セクション
- 5.10 CPU 情報ファイル
- 5.11 SOFTUNE V3/V5 ツールで作成したオブジェクトの入力
- 5.12 FR 用オブジェクトと FR80 用オブジェクトの混在

5.1 リンカの概要

リンカは、アセンブラが出力した複数のオブジェクトモジュールを結合し、アドレス解決などを行って実行形式のロードモジュールを作成するツールです。

■ リンカの概要

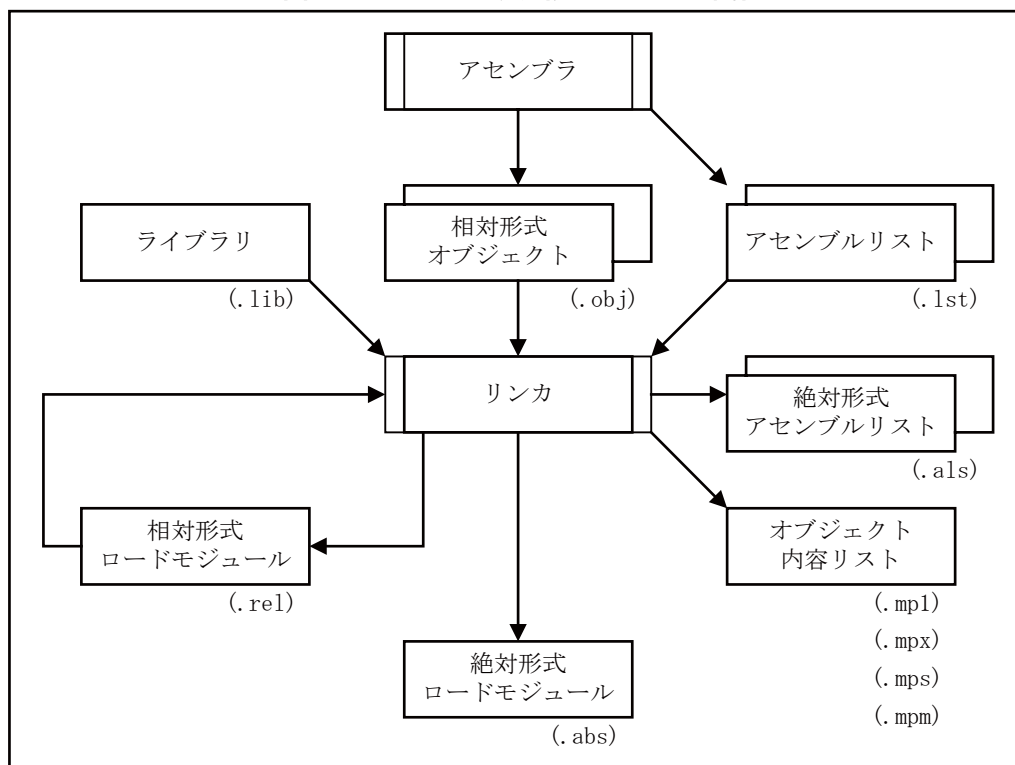
開発するプログラム規模が大きくなると、1本のソースプログラムだけですべてを記述するのには無理が生じてきます。

また、C/C++ コンパイラを使用した開発では、通常 C や C++ のライブラリファイルの取り込みが必要になります。

複数の関連しあったオブジェクトモジュールを結合して、メモリ配置アドレスを割当て、実行可能な形式のロードモジュールを作成するのがリンカの役割です。

図 5.1-1 にリンカと入出力ファイルの関係を示します。

図 5.1-1 リンカと入出力ファイルの関係



5.2 リンカの機能

リンカには多くの機能がありますが、大きく分けると次の 4 つに分類できます。

- 入出力ファイル/メッセージに関する制御
- セクションの結合/配置に関する制御
- ライブラリの検索に関する制御
- エントリアドレス/シンボル値の設定

■ 入出力ファイル/メッセージに関する制御

入出力ファイルとその制御の概要について説明します。詳しくは「5.2.1 入出力ファイル/メッセージに関する制御」を参照してください。

- 入力ファイルは、次の 4 つです。
 - 「アセンブラ出力オブジェクトモジュールファイル」
 - 「リストファイル」
 - 「リンカ出力の相対形式ロードモジュールファイル」
 - 「ライブラリファイル」
- 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は、コマンドラインまたはオプションファイル中に記述した順に処理されます。
- 出力ファイルは、次の 3 つです。
 - 最終目的となる「絶対形式ロードモジュールファイル」
 - 再入力可能な「相対形式ロードモジュールファイル」
 - 「リンクマップリストファイル」
- 出力ロードモジュールの形式（絶対/相対）指定や、出力ファイル名の変更ができます。
- マップリストは 1 ページの行数やページ幅の変更ができます。
- オプションを指定することにより、次の 4 つのファイルを出力できます。
 - アセンブラが出力したリストファイルを絶対形式にした「絶対形式アセンブルリストファイル」
 - 各モジュールで使用している外部シンボルについてモジュール間の定義 / 参照の相互参照を表示した「外部シンボル相互参照情報リスト」
 - モジュールごとに使用したローカルシンボルに関する情報を表示した「ローカルシンボル情報リスト」
 - 各モジュール内のセクション配置アドレスを表示した「セクション詳細マップリスト」

これらのファイルを出力するには、リンク時のロードモジュールの形式を絶対指定にする必要があります。

- メッセージにはプログラム版数などの情報を含む起動メッセージ、簡単な使用方法を示すヘルプメッセージ、およびエラーメッセージがあります。起動メッセージ出力の有無を選択したり、ワーニング検出レベルの選択ができます。

■ セクションの結合 / 配置に関する制御

セクションの結合 / 配置に関する制御の概要について説明します。詳しくは、「5.2.2 セクションの結合 / 配置に関する制御」を参照してください。

- ROM および RAM のアドレス範囲を指定することで、領域外への配置チェックができます。
- セクションの指定では複数セクションをグループ化し、まとめて処理したり、属性別のセクション選択などができます。
- セクションの指定には、ワイルドカードが使用できますので、セクションが多数ある場合のセクション結合や配置の指定が簡潔に行えます。
- ROM 化支援機能を備えています。
- 指定した ROM および RAM 領域内への自動配置ができます。

■ ライブラリの検索に関する制御

ライブラリの検索に関する制御について概要を説明します。詳しくは「5.2.3 ライブラリの検索に関する制御」を参照してください。

- C/C++ での開発時に、リンクに必要な C/C++ の実行時ライブラリを自動識別して結合できます (デフォルトライブラリファイルの検索)。
- ユーザが作成した複数のライブラリを検索できます。
- 個々のシンボルごとに、検索するライブラリファイルを指定できます。
- ライブラリ検索の抑止ができます。

■ エントリアドレス / シンボル値の設定

一時的に未定義外部シンボルに値を割り付けたり、エントリアドレスを設定したりできます。詳しくは「5.2.4 エントリアドレス / シンボル値の設定」を参照してください。

5.2.1 入出力ファイル/メッセージに関する制御

リンカの機能のうち、入出力ファイル/メッセージに関する制御を行うそれぞれの機能について説明します。

■ 入力オブジェクトファイルの指定

リンカの入力となるファイルは、アセンブラ出力のオブジェクトモジュールファイルと、リンカ出力の相対形式ロードモジュールファイルです。

入力するファイルすべてを指定します。ワイルドカードを使用することで簡単に指定が行えます。

■ 出力ロードモジュールファイル名の指定

リンク後に生成される出力ロードモジュールファイル名は、デフォルトではリンカが最初に入力したモジュールのファイル名をもとに作られます。

デフォルトファイル名は、リンク結果の全体を示す名称としては不適當である場合が多いので、このデフォルト出力ファイル名を変更するための機能です。

特に、ワイルドカードを使用してファイル名を指定した場合には、出力ファイル名がわかりにくくなりますので、出力ファイル名の指定を行うことを推奨します。

■ デバッグ情報の継承

デバッグのためには、シンボル情報やソースファイルに関する情報が必要です。

C/C++ およびアセンブラで、デバッグ情報作成の指定 (-g オプション) を行うと、オブジェクトモジュール中にデバッグ情報が作成されます。

リンカでは、このデバッグ情報を継承してロードモジュール中に出力するか、削除するかを選択できます。

■ 出力フォーマットの指定

リンクした結果として、絶対形式ロードモジュールを生成するのか、相対形式ロードモジュールを生成するのかを指定する機能です。

■ リストファイル名の指定

リストファイルは出力オブジェクトファイル名をもとに作成されますが、このデフォルトファイル名を変更するための機能です。

■ リストファイルの形式変更

リストファイル作成ではページ制御を行っていますが、1 ページの行数や 1 行の文字数を変更できます。

また、シンボル名の文字数が長いときは 1 行に表示できる範囲で打ち切っていますが、ユーザの定義どおりの名称で表示するように指示できます。

■ 警告チェックレベルの選択

警告は軽微なエラーであり、リンク処理が継続できる程度の不都合が発生した場合に通知されるメッセージですが、修正を要するものと、そのまま特に問題のないものがあります。これらのチェックレベルを選択できます。

■ 起動メッセージ表示の選択

起動時のツール名と著作権の表示を行うか否かを選択できます。

■ 終了メッセージ表示の選択

終了メッセージの表示を行うか否かを選択できます。

5.2.2 セクションの結合 / 配置に関する制御

リンカの機能のうち、セクションの結合 / 配置に関する制御を行うそれぞれの機能について説明します。

■ ROM および RAM 領域の指定

ROM 領域と RAM 領域のアドレス範囲を指定して領域名称を定義すると、セクションの配置指定時にアドレス指定の代わりにこの領域名称を用いることができ、範囲外への配置に対するチェックが可能になります。

セクションの自動配置を行う場合には、この領域指定範囲内に配置します。

■ セクションの配置順と配置アドレスの指定

すべてのセクションは任意の順序で、任意の領域に配置できます。セクション名の指定には、ワイルドカードが使用できます。

また、セクション内容種別を付加した指定が可能なので、ワイルドカードと組み合わせることで同じ内容種別 (code, data など) のセクションだけを集めることもできます。

■ セクションのグループ化

リンカは、セクション単位に結合と配置を行います。多くのセクション名を使ってプログラムを作成した場合、セクションの配置指定が煩雑になります。

複数セクションにグループ名を付け、連続領域に集めることで、そのまとまりをあたかも 1 つのセクションのように扱うことができます。

■ ROM 化支援

C/C++でのプログラム開発においては初期値付変数が生成され、その変数の書換えなどの処理を行うことが頻繁にあります。

組み込み用のアプリケーションにおいては、初期値データは ROM に置き、実行前に RAM ヘデータを転送しないと使用できないことになります。

このような使用を可能にするための機能です。詳しくは「5.9 ROM RAM 転送セクション」を参照してください。

5.2.3 ライブラリの検索に関する制御

リンカの機能のうち、ライブラリの検索に関する制御を行うそれぞれの機能について説明します。

■ ライブラリ検索パス指定

ライブラリの検索パスは、通常 C/C++ ライブラリを格納したディレクトリを環境変数で指定しておきますが、ユーザが自身で作成したライブラリを別のディレクトリに格納した場合などに指定します。

■ 検索ライブラリファイル指定

C/C++コンパイラが提供する実行時ライブラリ以外に、ユーザが独自に作ったライブラリファイルの名前を検索対象ライブラリとして指定します。

■ シンボルごとの検索ライブラリファイル指定

複数ライブラリファイルに同一の外部シンボル名が含まれている場合に、どのライブラリファイルのモジュールをリンクするかを指定するために使用します。

■ ライブラリ検索の抑止

デフォルトライブラリの検索を禁止したり、すべてのライブラリ検索を行わないようにすることができます。

5.2.4 エントリアドレス / シンボル値の設定

リンクの機能のうち、エントリアドレス / シンボル値の設定に関する機能について説明します。

■ エントリアドレス指定

プログラムの実行開始アドレスを、出力ロードモジュール中に設定する機能です。

■ 外部シンボル値の設定

プログラムが未完成であるか、外部シンボル名を間違えているかなどが原因で、リンク後に定義されていない外部シンボルがある場合はエラーとなります。

このエラーを一時的に取り除き、とりあえず実行可能なロードモジュールを作成したい場合に仮の値を設定する機能です。

5.3 セクションの種類

リンカが結合処理を行うときの最小単位はセクションです。

セクションはプログラムでの使用目的の違いにより，配置や結合の方法に特徴があります。

セクション名，内容種別，配置属性，結合属性などについて説明します。

■ セクション名

セクションを識別するために付けた名前です。

■ セクション内容種別

使用目的の違いによるセクション内容種別には以下の 5 つがあります。

実行，Read，Write の属性は，アセンブラが決定します。表 5.3-1 にセクションの種別を示します。

表 5.3-1 セクションの種別

種 別	説 明	属 性
CODE	プログラムコード領域	実行，Read
DATA	変数領域	Read，Write
CONST	初期値付変数領域	Read
STACK	スタック領域	Read，Write
IO	I/O 領域	Read，Write

■ セクションの配置属性

セクションの配置に関する属性は，再配置可能か否かの 2 種類あります。表 5.3-2 にセクションの配置属性を示します。

表 5.3-2 セクションの配置属性

属 性	説 明
ABS	絶対番地の指定されたセクションです。
REL	再配置可能なセクションです。

■ セクションの結合属性

セクションの結合に関する属性は，共有か連結かの 2 種類あります。表 5.3-3 にセクションの結合属性を示します。

表 5.3-3 セクションの結合属性

属 性	説 明
PUBLIC	次々に連続して結合されます。
COMMON	同じアドレスに重ねあわせて結合されます。

■ セクションの識別

リンカでは，セクション名と内容種別と結合属性が同一で，REL 属性のセクションを同一セクションとして扱います。

ABS 属性セクションは，リンカの配置処理の対象にはなりません。

リンカは，セクション名によってセクションの識別を行いますので，同じセクション名で内容種別や属性の異なるセクションは定義しないでください。

5.4 セクションの結合

リンカでは、「複数のオブジェクトを結合する」という表現を多く用いますが、正確には「オブジェクト中のセクションを結合する」という意味です。

セクションの結合方法には、単純連結結合 (PUBLIC) と共有結合 (COMMON) があります。

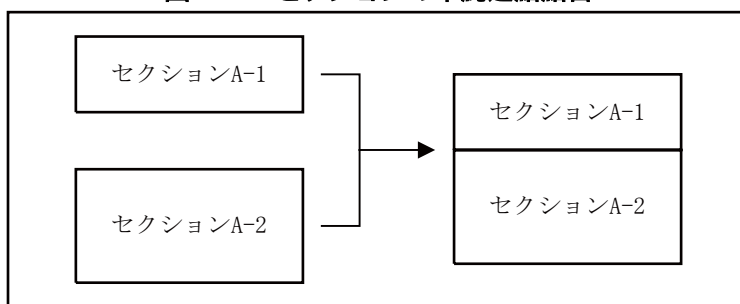
■ セクションの単純連結結合

同じセクション名と内容種別を持ち、結合属性が PUBLIC である REL セクションは単純連結結合を行います。

図 5.4-1 に、2 つのオブジェクトファイル中の同一セクションの単純連結結合の概要を示します。

結合後の全体のサイズは、A-1 と A-2 のサイズの合計に、バウンダリ調整により生じた、A-1 と A-2 の間の隙間分のサイズを加えたものとなります。

図 5.4-1 セクションの単純連結結合



■ セクションの共有結合

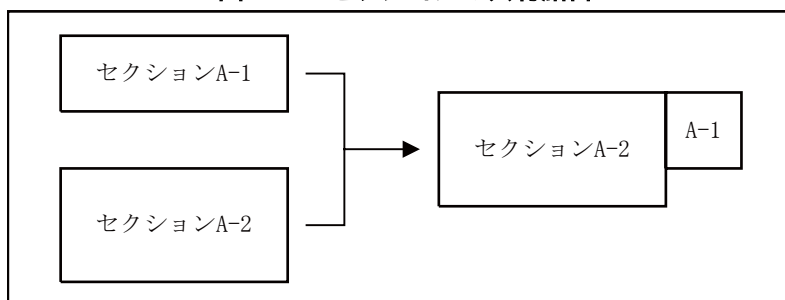
同じセクション名と内容種別を持ち、結合属性が COMMON である REL セクションは共有結合を行います。

初期値のない、データセクションでの使用が考えられます。

図 5.4-2 に、2 つのオブジェクトファイル中の同一セクションの共有結合の概要を示します。

結合後の全体のサイズは、A-1 と A-2 のサイズのうち大きい方になります。

図 5.4-2 セクションの共有結合



5.5 セクションの配置

リンカは、同一のセクションを結合したのちそれぞれのセクションの配置アドレスを決定します。ここでは、ユーザがアドレス指定をした場合を含めて、リンカのセクション配置の方法を説明します。

■ セクションのリンク

結合と配置処理は、相対セクションのみが対象となります。

絶対セクションは、結合・配置の対象にはなりません。

セクションの結合・配置は、以下のような手順で行います。

- 1) それぞれのオブジェクトモジュールから同一セクションを集める。
- 2) 集めた同一セクション同士を、結合属性に従って結合する。
- 3) 同一セクションを結合したものを配置する。

セクション配置順序に関するオプションが指定されている場合はそれに従って行われ、指定がない場合はオブジェクトファイル中での出現順がそのまま配置の順番になります。

詳しくは、「5.5.1 セクションの結合順序が指定されなかった場合の配置例」、「5.5.2 セクションの結合順序が指定された場合の配置例」、「5.5.3 セクショングループの指定がある場合の配置例」を参照してください。

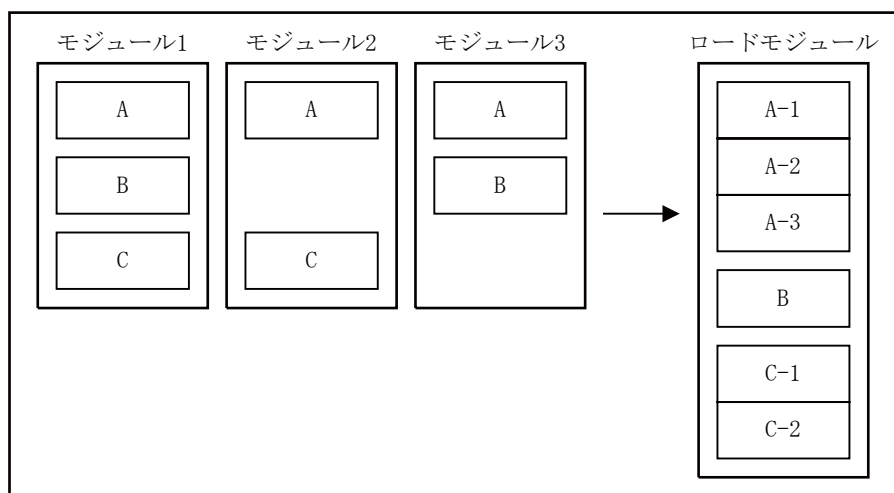
5.5.1 セクションの結合順序が指定されなかった場合の配置例

セクションの結合順序が指定されなかった場合の配置例を図 5.5-1 に示し，この図に従って説明します。

■ セクションの結合順序が指定されなかった場合の配置例

オブジェクトファイルの入力順が，モジュール 1 2 3 であるとき，セクションの出現順は A B C となるので，配置アドレスも低い方から順に A B C となります。

図 5.5-1 セクションの結合順序が指定されなかった場合の配置例



(注意事項) セクション A, C は PUBLIC 属性を持ち，セクション B は COMMON 属性を持つとした場合。

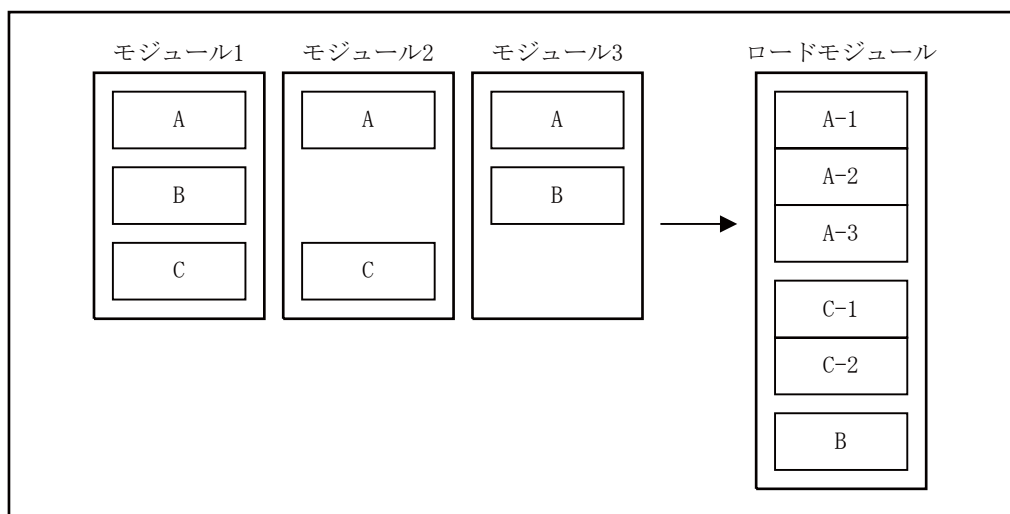
5.5.2 セクションの結合順序が指定された場合の配置例

セクションの結合順序が指定された場合の配置例を図 5.5-2 に示し，この図に従って説明します。

■ セクションの結合順序が指定された場合の配置例

オブジェクトファイルの入力順が，モジュール1 2 3であるとき，セクションの出現順はA B Cであるが，配置順としてA C Bが指定された場合。

図 5.5-2 セクションの結合順序が指定された場合の配置例



(注意事項) セクション A, C は PUBLIC 属性を持ち，セクション B は COMMON 属性を持つとした場合。

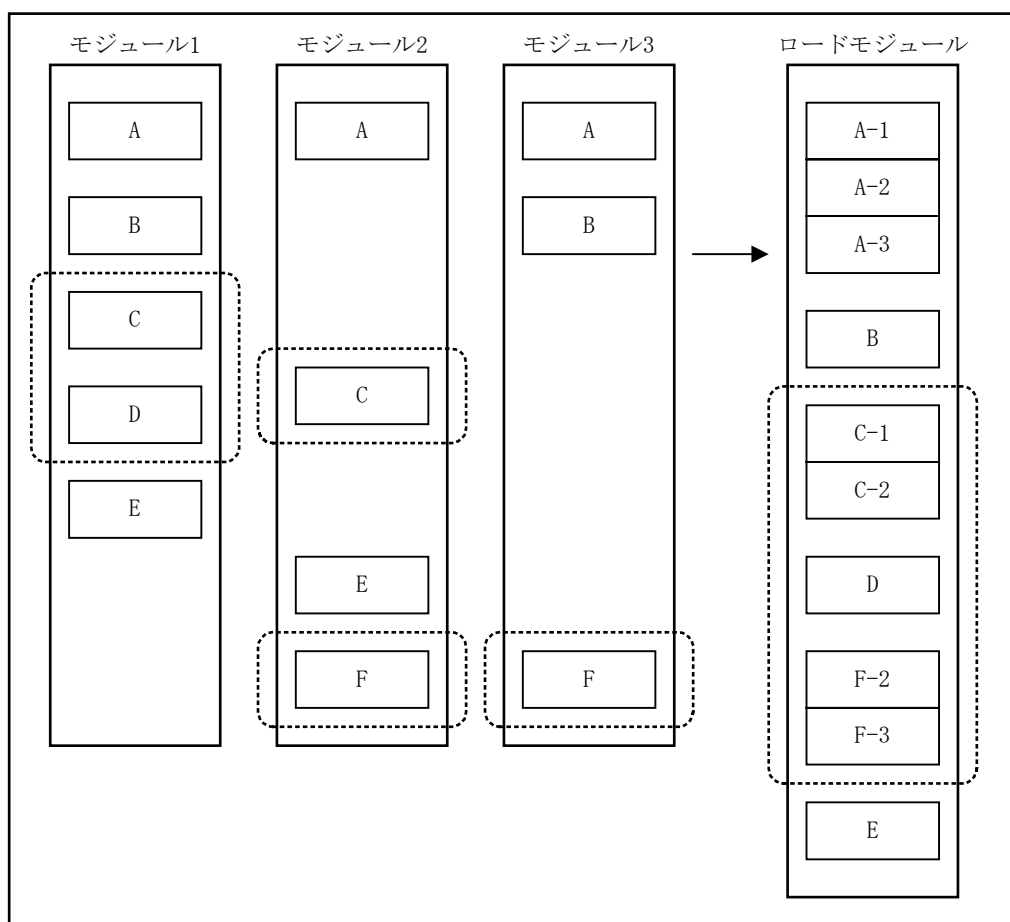
5.5.3 セクショングループの指定がある場合の配置例

セクショングループの指定がある場合の配置例を図 5.5-3 に示し，この図に従って説明します。

■ セクショングループの指定がある場合の配置例

グループ指定がある場合は，それぞれのグループに属するセクションは連続した領域に配置されます。セクションの出現順は，A B C D E F ですが，FはCの所属するグループ (C D F) に含まれているので，Eの前に配置されます。

図 5.5-3 セクショングループの指定がある場合の配置例



(注意事項) セクション B, E は COMMON 属性を持ち，ほかのセクションは PUBLIC 属性とする。配置順序指定はなく，セクション C, D, F が，同一グループにグループ化される場合。

5.6 セクションの自動配置

通常リンカは、ユーザの配置指定に従って、セクション配置アドレスの決定を行います。しかし、-AL オプションを指定することで、セクション配置アドレスの決定を自動的にリンカに任せることが可能です。

-ra, -ro オプションで指定した領域内へセクション配置を行うとき、アブソリュートセクションが存在していれば、配置アドレスが重ならないように、リロケータブルセクションの配置を行います。このとき、アライメント値およびサイズの大きいセクションから順に配置し、空き領域が最小となるような最適配置を行います。

■ セクションの自動配置

本リンカは、以下に示す 2 つのタイプのセクションの自動配置をサポートしています。

- -AL 1 が指定された場合のセクションの自動配置
- -AL 2 が指定された場合のセクションの自動配置

セクションの自動配置の詳細については、「5.6.1 -AL 1 が指定された場合のセクションの自動配置」、「5.6.2 -AL 2 が指定された場合のセクションの自動配置」をご参照ください。

5.6.1 -AL 1 が指定された場合のセクションの自動配置

-AL 1 が指定されたとき，リンカは領域内に存在するアブソリュートセクションと配置アドレスが重ならないように，リロケートブルセクションの配置を行います。

■ 配置アドレスの決定方法

自動配置の対象となるセクションは，-sc オプションで領域名指定が行われているセクションです。

各セクションの配置は，アライメント値の大きいセクション順に処理し，アライメント値が同じ場合にはサイズの大きいセクションから順に処理されます。

表 5.6-1 に，各セクションのアライメント値とサイズを示します。

表 5.6-1 各セクションのアライメント値 / サイズ

セクション名	アライメント値	サイズ
code1	2	0x0180
code2	2	0x0100
code3	2	0x0200
code4	4	0x0100
code5	4	0x0200
code6	2	0x0020

例えば，表 5.6-1 のようなセクションでは，次のように配置処理の順番が決定されます。

- 1) アライメント値が 4 であるセクション (code4, code5) が，アライメント値が 2 のセクション (code1, code2, code3) より先に配置処理が行われます。
- 2) code4 と code5 では，サイズが大きい code5 の方が先に配置処理されます。

したがって，表 5.6-1 に示されるセクションの配置処理の順番は，表 5.6-2 で示すようになります。

表 5.6-2 各セクションのアライメント値 / サイズ

配置処理順	セクション名	アライメント値	サイズ
1	code5	4	0x0200
2	code4	4	0x0100
3	code3	2	0x0200
4	code1	2	0x0180
5	code2	2	0x0100
6	code6	2	0x0020

配置先は，配置可能で最小の空きエリアが選択されます。

■ -AL 1 が指定された場合の配置例

リンカのオプション指定と各セクションの内容が、下記例（図 5.6-1、表 5.6-3）の場合の配置例を示します。

図 5.6-1 リンカのオプション指定

```
-ro ROM=0xC1000/0xC18FF
-sc code1+code2+code3+code4+code5+code6=ROM
-AL 1
:
```

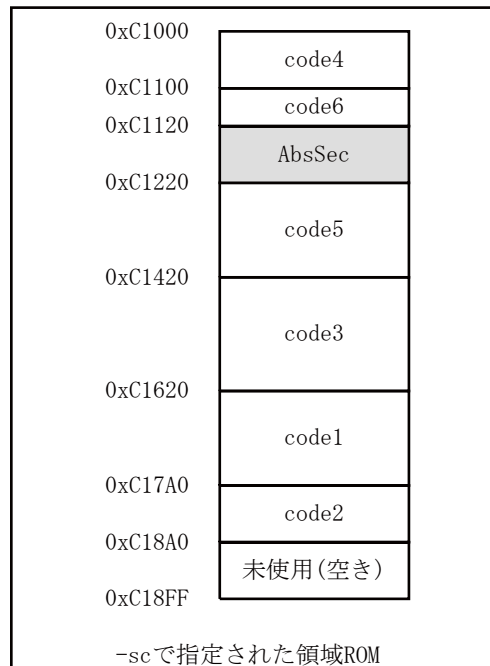
表 5.6-3 各セクションの内容

セクション名	配置属性	アドレス範囲	アライメント値	サイズ
code1	REL	-	2	0x0180
code2	REL	-	2	0x0100
code3	REL	-	2	0x0200
code4	REL	-	4	0x0100
code5	REL	-	4	0x0200
code6	REL	-	2	0x0020
AbsSec	ABS	0xC1120 ~ 0xC121F	0	0x0100

図 5.6-2 に、-AL 1 が指定された場合の配置例を示します。

このように、-AL 1 を使用すると、リンカが指定された領域内で、アブソリュートセクションと重ならないように、また空き領域が最小となるような最適配置を行います。

図 5.6-2 -AL になる AL 1 が指定された場合の配置例



5.6.2 -AL 2 が指定された場合のセクションの自動配置

-AL 2 が指定された場合，リンカは配置指定の行われていないセクションを空き領域に自動で配置します。

その際，リンカはセクションの種別をもとに，ROM 領域または RAM 領域のどちらに配置するかを判断します。

ここでは，セクションの種別ごとの配置先や配置アドレス決定の順序について説明します。

また，-AL 2 が指定された場合の配置例を図 5.6-4 に示します。

■ セクション種別と配置先

-AL 2 が指定された場合，リンカは配置指定の行われていないセクションを自動的に配置します。このとき，リンカはセクション種別により，表 5.6-4 に示すように配置先（領域）の決定を行います。

表 5.6-4 セクション種別と配置先

配置先	セクション種別
ROM 領域 (-ro で指定された領域)	CODE CONST
RAM 領域 (-ra で指定された領域)	IO DATA STACK

■ 配置アドレスの決定方法

-AL 2 が指定された場合，リンカは，表 5.6-5 に示す順にセクション配置アドレス決定を行います。

表 5.6-5 で示す順のように，自動配置よりもユーザ指定が優先されるようになっています。

なお，リンカは，配置可能な場所を検索する場合，常に低位アドレスから検索を行います。

表 5.6-5 セクション配置先

順番	処理対象のセクション	配置先と配置方法
1	ABS 属性を持つセクション	セクションが持つアドレスに配置する。
2	-sc オプションで，"-sc Section=0x0100" のように，配置アドレスの指定が行われているセクション	指定されたアドレスに配置する。
3	-sc オプションで，"-sc Section=ROM" のように，配置領域の指定が行われているセクション	指定された領域内で，ほかのセクションと重ならないように，配置可能な場所を検索して配置する。
4	配置に関する指定が行われていないセクション	表 5.6-4 に従って配置領域の決定を行い，その領域内で，ほかのセクションと重ならないように，配置可能な場所を検索して配置する。

■ -AL 2 が指定された場合の配置例

リンカのオプション指定と各モジュールに含まれるセクションが、下記例 (図 5.6-3, 表 5.6-6) の場合の配置例を示します。

図 5.6-3 リンカのオプション指定

```
file1.obj, file2.obj file3.obj
-ro ROM=0xFF8000/0xFFFFFFFF
-ra RAM=0x000000/0x0007FF
-sc ivect=0xFFFF00
-AL 2
```

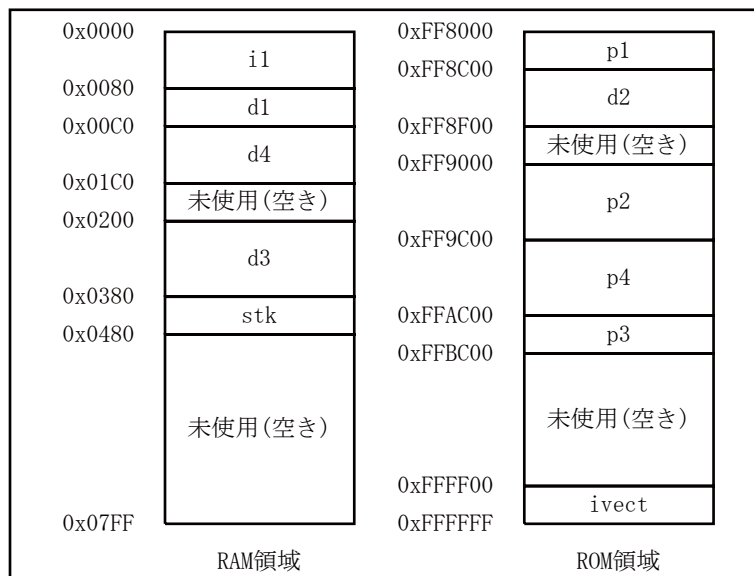
表 5.6-6 各モジュールに含まれるセクション

モジュール	セクション名	種別	配置属性	アドレス	サイズ
file1.obj	p1	CODE	REL	-	0x0C00
	p4	CODE	REL	-	0x1000
	stk	STACK	REL	-	0x0100
	ivect	CONST	REL	-	0x0100
file2.obj	i1	IO	ABS	0x0000	0x0080
	d1	DATA	REL	-	0x0040
	d3	DATA	ABS	0x0200	0x0180
file3.obj	p3	CODE	REL	-	0x1000
	d2	CONST	REL	-	0x0300
	p2	CODE	ABS	0x9000	0x0C00
	d4	DATA	REL	-	0x0100

図 5.6-4 に、-AL 2 が指定された場合の配置例を示します。

このように、-AL 2 を使用すると、リンカが自動的にセクション配置を行うため、ユーザはプログラム動作上必要な、最小限のオプション指定で済み、セクション配置指定の手間が軽減されます。

図 5.6-4 -AL 2 が指定された場合の配置例



5.7 ライブラリの検索

リンカでの検索を行いたいライブラリファイルは、以下の 3 とおりの方法で指定できます。

- デフォルトライブラリの設定
 - -l オプションによる指定
 - -el オプションによる指定
-

■ 検索ライブラリの指定

リンカは、指定された入力ファイルの結合がすべて終了した時点で、未定義のシンボルが残っていれば、その解決のためにライブラリファイルの検索を行います。

ライブラリファイルの検索は、検索漏れが発生しないように再帰的に行われます。

検索すべきライブラリファイルは、以下の 3 とおりの方法で指定できます。

● デフォルトライブラリの設定

プログラムを C/C++ 言語で記述した場合、リンク時には C/C++ ライブラリが必要となります。

リンク時にユーザが検索すべきライブラリファイルを指定することは面倒であり、指定を間違えれば意図しないモジュールを結合してしまうこともあります。

このようなことを防ぐために、C/C++ コンパイラは選択すべきライブラリファイル名の情報をアセンブラの疑似命令で指示し、アセンブラはその情報をオブジェクトモジュール内に設定します。

このようにして、リンクされるオブジェクトモジュール内に設定されたライブラリファイル名をデフォルトライブラリと呼びます。

● -l オプションによる指定

デフォルトライブラリとして設定されていないライブラリファイルを検索対象とした場合は、ユーザがリンク時に指定しなくてはなりません。

ユーザ自身がライブラリファイルを作成し、これをリンクしたい場合、上記のデフォルトライブラリとするために、アセンブラでライブラリ指定の疑似命令を記述するか、リンカ起動時に -l オプションで指定します。

なお、-l オプションについて詳しくは「6.2.26 検索ライブラリファイルの指定 (-l)」を参照してください。

● -el オプションによる指定

ライブラリファイルは複数個指定できます。

異なるライブラリファイルに、同一名の外部定義シンボルが含まれている場合もあり得ます（原因の判りにくい障害を招く恐れがありますので、極力避けてください）。

-el オプションは、シンボルごとにシンボル検索を行うライブラリファイル名を特定するための指定です。詳しくは「6.2.28 シンボル個別のライブラリの指定 (-el)」を参照してください。

■ ライブラリファイルの検索順序

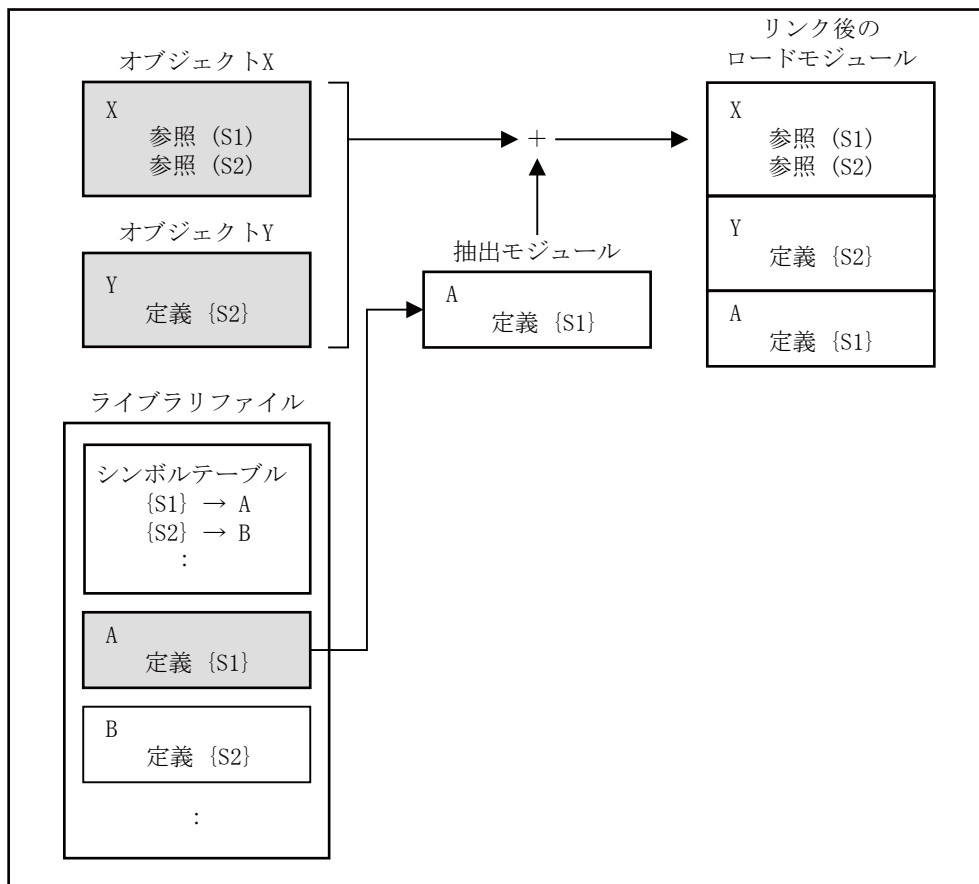
リンカは、まず -el オプションで指定のあるシンボルについてシンボル参照を解決し、次に -l オプションで指定された順にライブラリの検索を行い、最後にデフォルトライブラリを検索します。ライブラリファイルから、モジュールの取り込みがなくなるまで、この一連の検索が続けられます。

5.7.1 ライブラリファイルが1つの場合の検索例 1

ライブラリファイルが1つの場合の検索例は3つあります。このうち1つを図 5.7-1 に示し、この図に従って説明します。

■ ライブラリファイルが1つの場合の検索例 1

図 5.7-1 ライブラリファイルが1つの場合の検索例 1



オブジェクトモジュール X と Y を結合した結果、X 内の外部参照シンボル (S2) は、Y 内の外部定義シンボル {S2} によって解決されます。

X 内の外部参照シンボル (S1) が未解決なのでライブラリを検索します。

モジュール A に、外部定義シンボル {S1} が含まれているので、このモジュールを新たにリンクします。

これで、未解決の外部参照シンボルはなくなったので、モジュール X, Y, A から成るロードモジュールを作成してリンク処理を終了します。

ランブラリ中のモジュール B に外部定義シンボル {S2} がありますが、既にリンクされたモジュール Y に存在しているため、(S2) に関してはライブラリ検索の対象になりません。

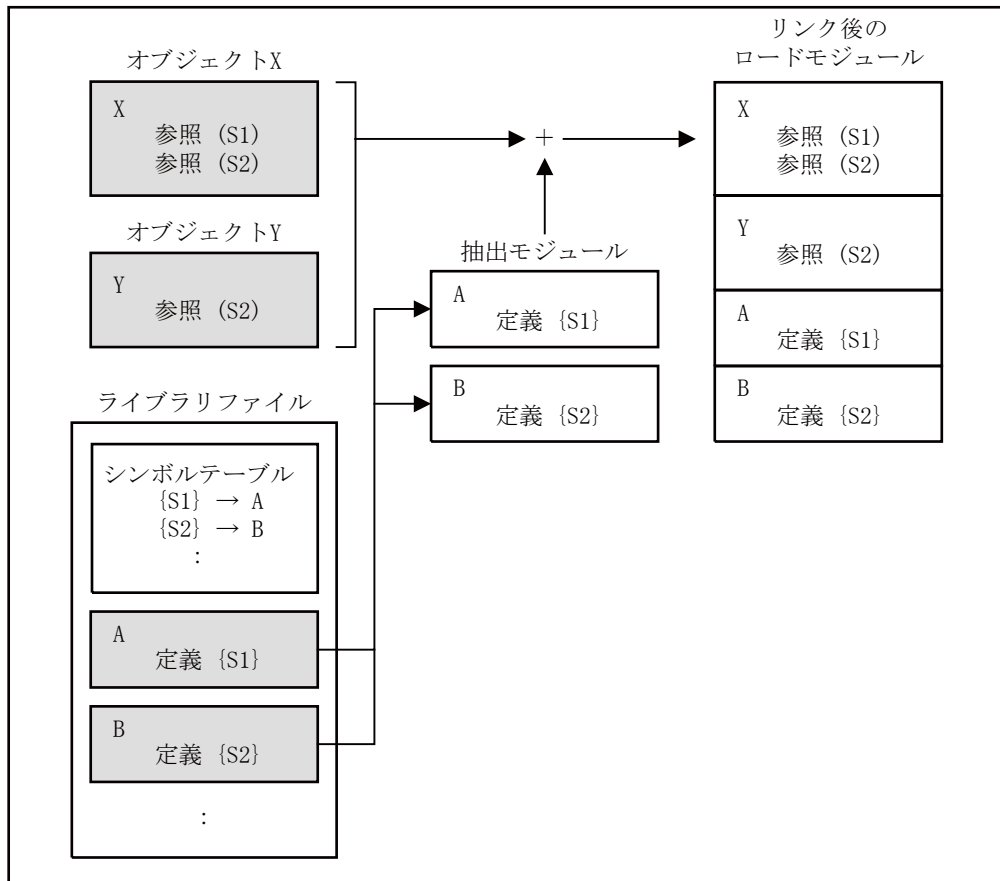
(注意事項) 図中の () で括ったシンボル名は " 参照 " を、{ } で括ったシンボル名は " 定義 " を表します。

5.7.2 ライブラリファイルが1つの場合の検索例 2

ライブラリファイルが1つの場合の検索例は3つあります。このうち1つを図 5.7-2 に示し、この図に従って説明します。

■ ライブラリファイルが1つの場合の検索例 2

図 5.7-2 ライブラリファイルが1つの場合の検索例 2



オブジェクトモジュール X と Y を結合した結果、X, Y 内の外部参照シンボル (S1) と (S2) は、共に未解決となります。

ライブラリを検索し、モジュール A に外部定義シンボル {S1} が、モジュール B に外部定義シンボル {S2} が含まれているので、この2つのモジュールを新たにリンクします。

これで、未解決の外部参照シンボルはなくなったので、モジュール X, Y, A, B から成るロードモジュールを作成してリンク処理を終了します。

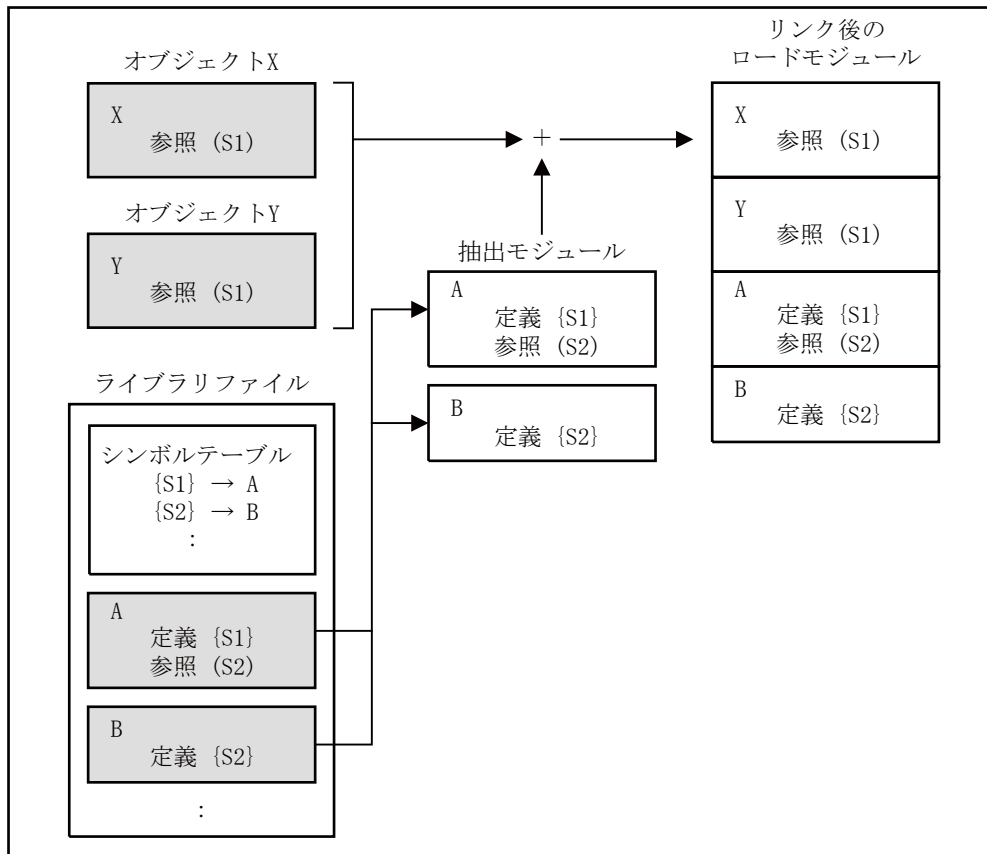
(注意事項) 図中の () で括ったシンボル名は "参照" を、{ } で括ったシンボル名は "定義" を表します。

5.7.3 ライブラリファイルが1つの場合の検索例 3

ライブラリファイルが1つの場合の検索例は3つあります。このうち1つを図 5.7-3 に示し、この図に従って説明します。

■ ライブラリファイルが1つの場合の検索例 3

図 5.7-3 ライブラリファイルが1つの場合の検索例 3



オブジェクトモジュール X と Y を結合した結果、X, Y 内の外部参照シンボル (S1) が未解決となります。

ライブラリを検索し、モジュール A に外部定義シンボル {S1} が含まれているので、このモジュールを新たにリンクします。

この結果、外部参照シンボル (S2) が新たな未解決シンボルとなったため、もう一度ライブラリを検索し、{S2} の定義されているモジュール B をリンクします。

これで、未解決の外部参照シンボルはなくなったので、モジュール X, Y, A, B から成るロードモジュールを作成してリンク処理を終了します。

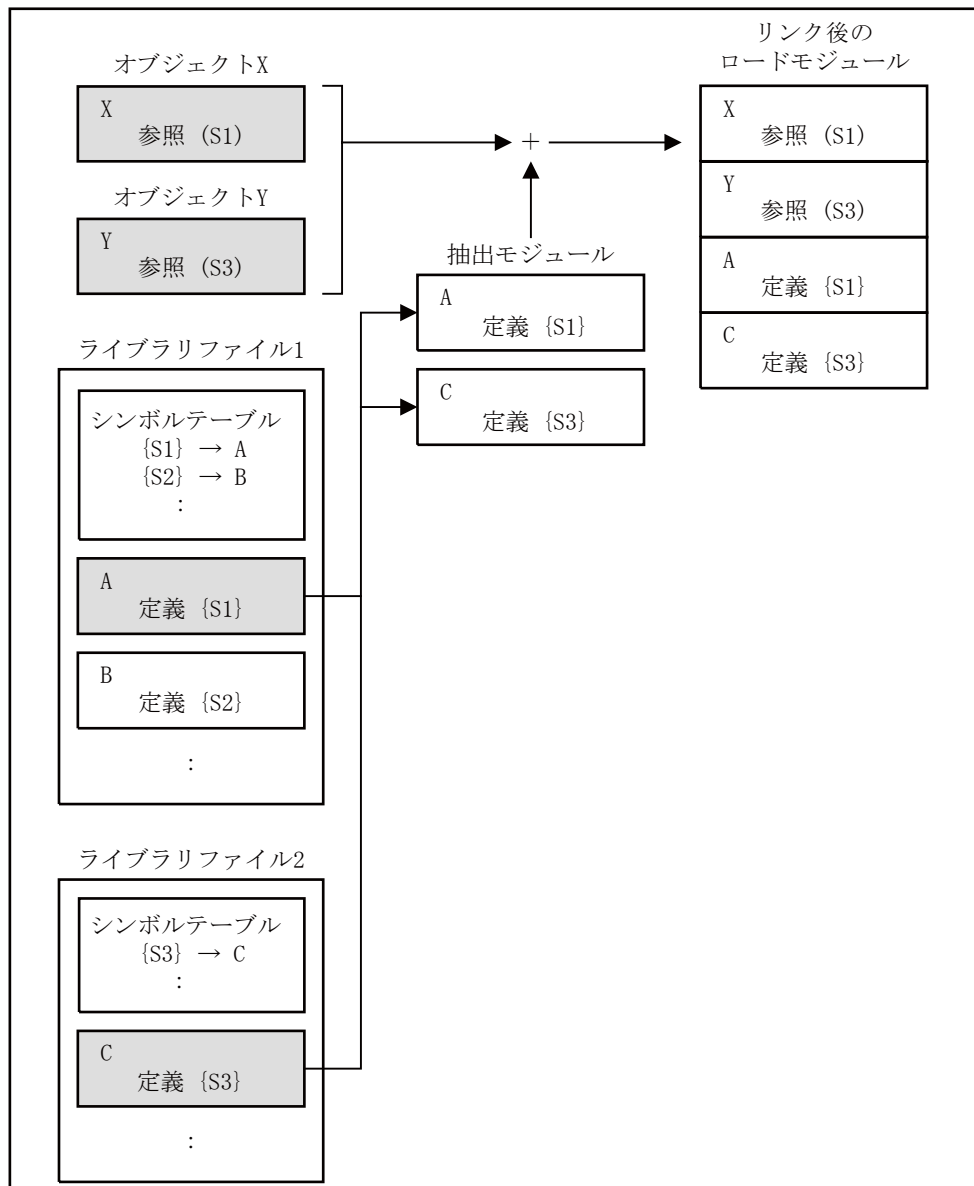
(注意事項) 図中の () で括ったシンボル名は " 参照 " を、{ } で括ったシンボル名は " 定義 " を表します。

5.7.4 ライブラリファイルが複数の場合の検索例 1

ライブラリファイルが複数の場合の検索例は2つあります。このうち1つを図5.7-4に示し、この図に従って説明します。

■ ライブラリファイルが複数の場合の検索例 1

図 5.7-4 ライブラリファイルが複数の場合の検索例 1



検索対象のライブラリファイルが複数ある場合、指定された順にライブラリ検索を行います。X, Y モジュールのリンク後、未解決シンボルの {S1} と {S3} が残ります。

ライブラリ 1、ライブラリ 2 の順で検索します。

ライブラリ 1 の検索で {S1} の定義のあるモジュール A をリンクします。

次に (S3) についてライブラリ 1 を検索しますが、見つからないのでライブラリ 1 の検索を終了し、ライブラリ 2 を検索します。

{S3} がモジュール C で定義されていますので、このモジュールをリンクします。

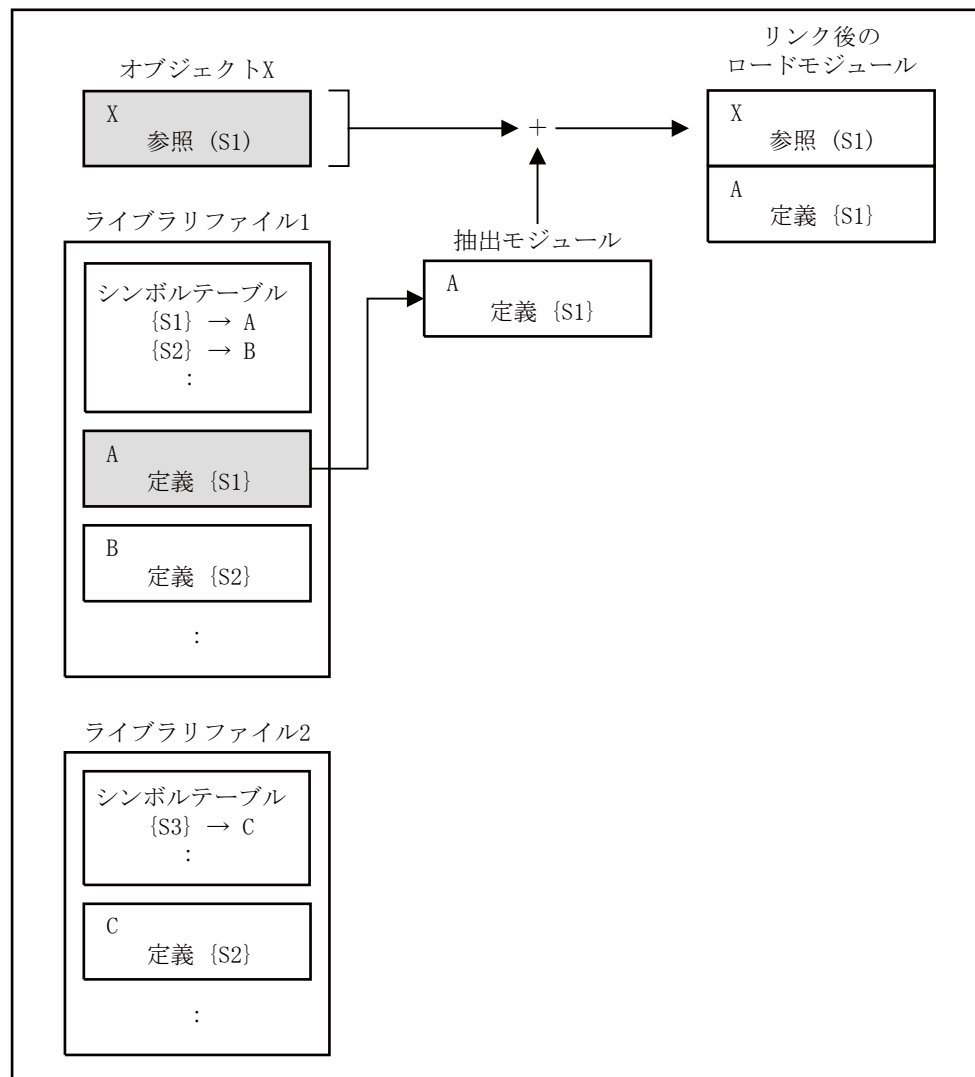
(注意事項) 図中の () で括ったシンボル名は " 参照 " を、{ } で括ったシンボル名は " 定義 " を表します。

5.7.5 ライブラリファイルが複数の場合の検索例 2

ライブラリファイルが複数の場合の検索例は2つあります。このうち1つを図5.7-5に示し、この図に従って説明します。

■ ライブラリファイルが複数の場合の検索例 2

図 5.7-5 ライブラリファイルが複数の場合の検索例 2



検索対象のライブラリファイルが複数ある場合、指定された順にライブラリ検索を行います。

複数のライブラリファイルに同一名の外部シンボル含まれているような場合、先に検索したライブラリ中のモジュールをリンクします。

X モジュールの未解決シンボル (S1) の解決のために、ライブラリ 1、ライブラリ 2 の順で検索します。

ライブラリ 1 の検索で {S1} の定義のあるモジュール A をリンクします。

この時点で、未解決シンボルはなくなりましたので、ライブラリの検索は終了します。
この場合、ライブラリ 2 の検索は行われません。

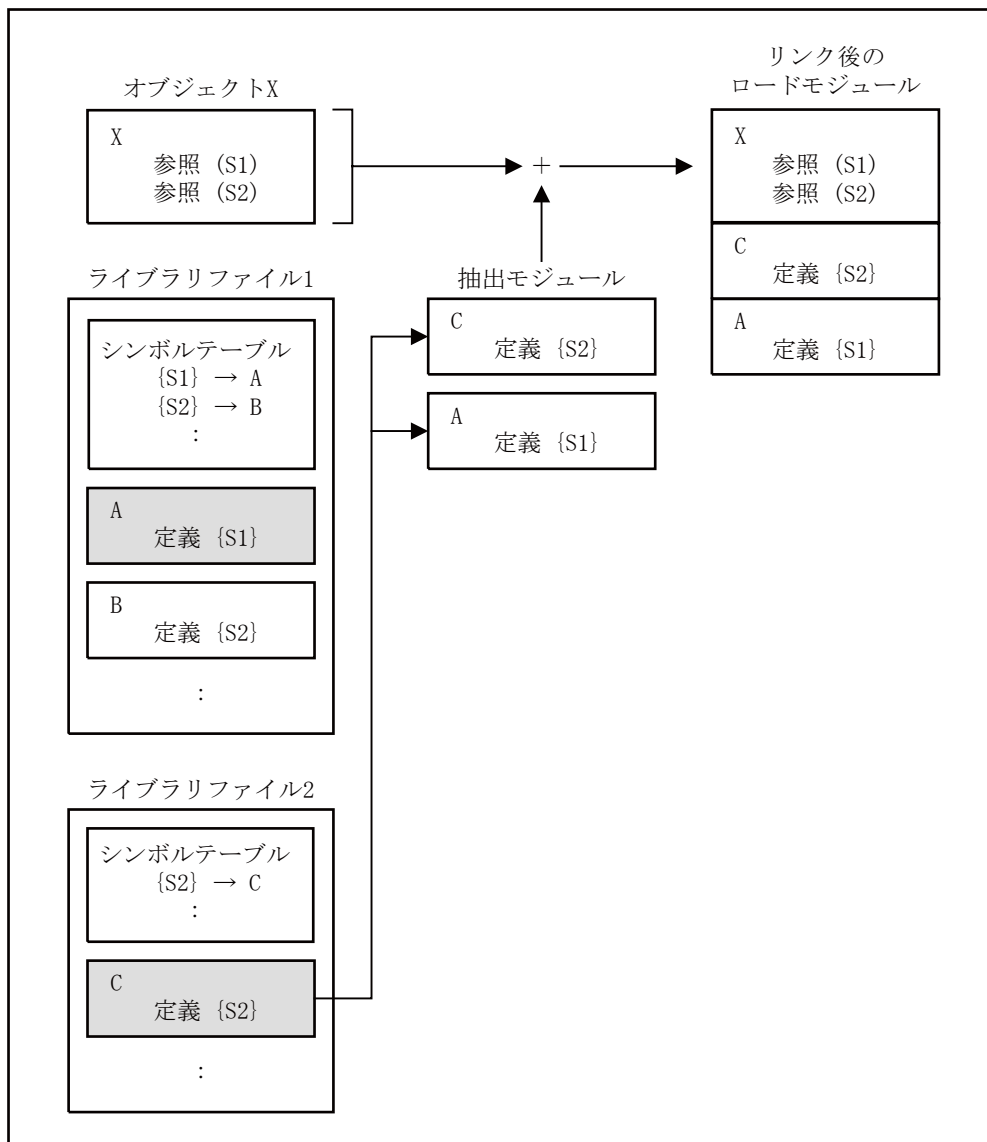
(注意事項) 図中の () で括ったシンボル名は " 参照 " を , { } で括ったシンボル名は
" 定義 " を表します。

5.7.6 ライブラリファイルが個別に指定された場合の処理

ライブラリファイルが個別に指定された場合の処理を図 5.7-6 に示し、この図に従って説明します。

■ ライブラリファイルが個別に指定された場合の処理

図 5.7-6 ライブラリファイルが個別に指定された場合の処理



ライブラリファイルが個別に指定された外部参照シンボルは、まず最初に検索されます。

ここでは、(S2) シンボルをライブラリ 2 から取り込むように指定されたものとします。

(S2) についてライブラリ 2 を検索し、モジュール C をリンクします。

(S1) の解決に、ライブラリファイル 1 を検索し、モジュール A をリンクします。

ライブラリファイル 1 中の {S2} は、検索対象になりません。

(注意事項)

- 図中の () で括ったシンボル名は " 参照 " を , { } で括ったシンボル名は " 定義 " を表します。
- 指定されたライブラリに , 検索対象の外部定義シンボルが見つからなかった場合 , この例のようにほかの検索対象ライブラリ中に該当するシンボルがあればシンボル解決が行われますので注意してください。

5.8 ROM/RAM 領域

組み込み用のアプリケーション開発では、使用することのできる ROM および RAM のサイズや、アドレス範囲に制約があるものです。

リンク時に、リンクにこの領域を通知しておくことで、サイズオーバや使用できないアドレスへのセクション配置についてのチェックができます。

セクションの自動配置を行う場合には、この領域指定範囲内に配置します。

■ ROM/RAM 領域の設定とセクション配置

リンクのセクション配置は、【例 1】に示すようにセクション名 (CODE) と、配置する先頭アドレス (0x1000) を指定するといった方法で行います。

【例 1】

```
-sc CODE=0x1000,DATA=0x0180
```

セクションの終了アドレスは、リンク結果の出力であるマップリストにより確認する必要があります。

本リンクでは、セクション配置の先頭アドレスと終了アドレスを指定することで、決められた範囲内に配置されたか否かをチェックする機能を備えています。

まず、【例 2】に示すように -ro および -ra オプションにより、配置アドレス範囲を決めて、領域名と対応づけます。

【例 2】

```
-ro CodeA=0x1000/0x3FFF
```

```
-ra DataA=0x0180/0x057F
```

領域名 CodeA は、0x1000 ~ 0x3FFF のアドレス範囲を示します。

領域名 DataA は、0x0180 ~ 0x057F のアドレス範囲を示します。

領域名を使用したセクション配置指定は、【例 3】のようになります。

【例 3】

```
-sc CODE=CodeA, DATA=DataA
```

領域名を使用したセクション配置では、指定したアドレス範囲内に配置されたか否かをチェックできます。

< 注意事項 >

ROM/RAM 領域は -cpu オプションで指定された MB 番号を元に CPU 情報ファイルから自動的に次の名前で設定されます。

- ROM 領域 _ROM_*_
- RAM 領域 _RAM_*_

これらを用いて -sc オプションの設定を行うことが可能です。CPU 情報ファイルについての詳細は、「5.10 CPU 情報ファイル」を参照してください。

5.9 ROM RAM 転送セクション

C/C++ コンパイラを使用したプログラム開発においては初期値付変数が生成され、その変数の書換えなどの処理を行うことが頻繁にあります。

これらの変数は実行時に書換えられるため、アプリケーション実行時は RAM になくてもなりません。したがって、組み込み用のプログラムにおいては、初期値データは ROM に置き、アプリケーション実行前に RAM へ初期値データを転送しないと使用できないことになります。

ROM RAM 転送セクションは、このような使用を可能にするための機能です。

■ ROM RAM 転送セクション

C/C++ コンパイラを使用したプログラム開発においては初期値付変数が生成され、その変数の書換えなどの処理を行うことが頻繁にあります。

組み込み用プログラムでは、初期値付変数データは ROM に置かれますが、これらの変数は実行時に書換えられるため、アプリケーション実行時は RAM になくてもなりません。

したがって、アプリケーション実行前に RAM へ初期値データを転送して使用することになります。

本リンカでは、このような使用を簡単にするため、ROM RAM 転送セクションを指定することで、プログラムの参照アドレスは RAM 上で解決し、初期値データは ROM 上に配置するしくみをサポートしています。

■ ROM RAM 転送セクションの使用方法

ROM RAM 転送セクションの指定は、-sc オプションを用いて次のように使用します。初期値付変数がまとめられているセクションを INIT とします。

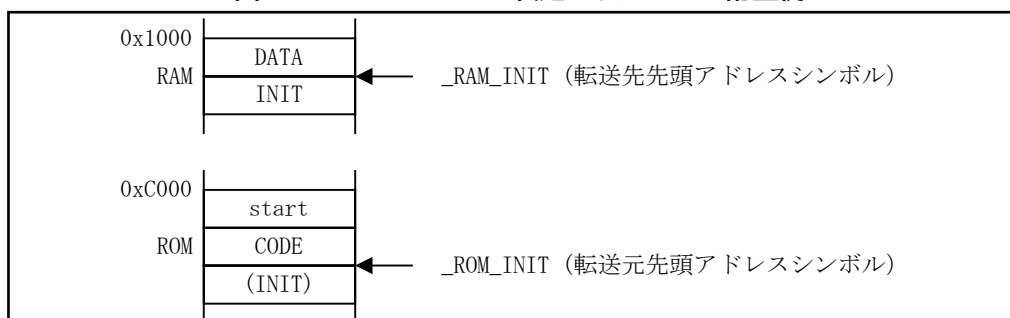
DATA は、初期値なし変数のセクションとし、start は、ROM 上の初期値付変数データを RAM へ転送するプログラム、CODE は、実行されるアプリケーションプログラムとします。

```
-sc DATA+INIT=0x1000,start+CODE+@INIT=0xC000
```

ここで、上のように、RAM 側 (0x1000) に INIT を配置するように指定し、ROM 側 (0xC000) にもセクション名の先頭に @ マークを付加して INIT を配置するように指定します。

このように配置指定を行うと INIT は ROM RAM 転送セクションとして処理され、図 5.9-1 のような配置になり、INIT のアドレス解決は RAM 側で行われ、初期値データは ROM 側に配置されます。

図 5.9-1 ROM RAM 転送セクションの配置例



このとき、_ROM_INIT や _RAM_INIT のように、ROM RAM 転送セクションには、それぞれセクションの先頭を示すシンボルが "_ROM_ セクション名", "_RAM_ セクション名" の規則で自動生成されます。

ユーザは、これらシンボルを、ROM 上の初期値付変数データを RAM へ転送するプログラム内で使用できます。

"_ROM_ セクション名", "_RAM_ セクション名" はリンカの予約シンボルです。プログラム中で定義を行わないでください。詳細は、「8.1 リンカの制限事項」を参照してください。また、初期値付変数データの転送プログラム例は、「8.2 リンカの使用上の Q&A」を参照してください。

■ ROM RAM 転送セクションの注意

ROM RAM 転送セクションを指定するとセクションの Write 属性が無条件に変更されます。

ROM 側へ配置されたセクションは、Write 属性が不可に変更され、RAM 側へ配置されたセクションは Write 属性が可に変更されます。

次のようなプログラムの ROM RAM 転送セクションを例に説明します。

```
.program    sample

            .section    init, data
val1: .word    0x1234
val2: .word    0x5678

            .section    Data, data
val3: .res.w    0x1

            .section    Prog1, code
ld        @val1, r2
ld        @val2, r3

            .section    Prog2, code
ldi        @val3, r13
cmp        #0, r13
```

上記プログラムでは、初期値付変数が記述されているセクション init、変数領域が記述されているセクション Data、プログラムコードが記述されているセクション Prog1、Prog2 があります。

それぞれのセクションのアセンブル後の実行、Read、Write の可 / 不可は、表 5.9-1 のようになります。

表 5.9-1 アセンブル後の各セクションの属性

セクション名	実行	Read	Write
init	×		
Data	×		
Prog1			×
Prog2			×

: 可

× : 不可

ここで、リンク時に、init と Prog1 の ROM → RAM 転送セクション指定を行うと、それぞれのセクションのリンク後の実行、Read、Write の可 / 不可は、表 5.9-2 のようになります。

表 5.9-2 リンク後の各セクションの属性

セクション名	実行	Read	Write	備考
init(RAM 側)	×			
init(ROM 側)	×		×	リンカにより用意された ROM 側のセクション Write が不可に変更された
Data	×			
Prog1(RAM 側)				Write が可に変更された
Prog1(ROM 側)			×	リンカにより用意された ROM 側のセクション
Prog2			×	

: 可

× : 不可

5.10 CPU 情報ファイル

リンカは、-cpu オプションから CPU を特定し、CPU 情報ファイルから ROM/RAM 領域を自動的に設定します。

■ CPU 情報ファイル

リンカは、-cpu オプションから CPU を特定し、CPU 情報ファイルから該当するチップの情報を抽出し、ROM/RAM 領域を自動的に設定します。

■ ROM/RAM 領域名

リンカは、ROM/RAM 領域を次の名前で設定します。

- ROM 領域の場合：_ROM_*_

上記 * には、低位アドレスの領域順に 1 から順番に番号が入ります。領域が 1 つしかない場合は、"_ROM_1_" となります。

- RAM 領域の場合：_RAM_*_

上記 * には、低位アドレスの領域順に 1 から順番に番号が入ります。領域が 1 つしかない場合は、"_RAM_1_" となります。

これら名前は、-sc オプションで使用可能です。

■ CPU 情報ファイル名

以下に、CPU 情報ファイル名と検索ディレクトリを示します。

- CPU 情報ファイル名

- 911.csv

- 検索ディレクトリ

- %FETOOL%\LIB\911

■ 内蔵 ROM/RAM 領域自動設定の抑止指定

リンカは、デフォルトで -cpu オプションから CPU を特定し、CPU 情報ファイルから該当するチップの情報を抽出し、ROM/RAM 領域を自動的に設定します。

本機能を抑止したい場合は、-Xset_rora オプションを指定します。

< 注意事項 >

CPU 情報ファイルが見つからない場合、または CPU 情報ファイル内に該当 MB 番号の情報が存在しない場合、リンカはエラーを出力します。

5.11 SOFTUNE V3/V5 ツールで作成したオブジェクトの入力

リンカ (flnk911s) では、SOFTUNE V3 や SOFTUNE V5 の言語ツールで作成されたオブジェクトを混在することが可能です。

■ SOFTUNE V3/V5 ツールで作成したオブジェクトの混在

リンカ (flnk911s) では、SOFTUNE V3 や SOFTUNE V5 の言語ツール (fasm911s や flib911s, flibs) で作成されたオブジェクトやライブラリの混在が可能です。

リンカは、-w オプションで 2 が指定されているときのみ、SOFTUNE V3/V5 のオブジェクトが入力された場合にインフォメーションを出力します。

< 注意事項 >

flnk911s が出力するロードモジュールファイルは、ダウンロード高速化に対応した新しいファイル形式で出力されるため、SOFTUNE V3/V5 デバッガや、SOFTUNE V3/V5 ロードモジュールコンバータに入力できません。

なお、新しいロードモジュールコンバータ (f2ms, f2is, f2es) は、SOFTUNE V3/V5 形式のロードモジュールも処理可能です。

5.12 FR 用オブジェクトと FR80 用オブジェクトの混在

リンカ (flink911s) は、-cpu オプションで設定したターゲット CPU が FR80 の場合、FR 用オブジェクトが混在すると警告を出力します。

-cpu オプションで指定したターゲット CPU が FR の場合、FR80 用オブジェクトが混在するとエラーを出力します。

■ FR 用オブジェクトと FR80 用オブジェクトの混在

リンカ (flink911s) は、-cpu オプションで設定したターゲット CPU が FR80 の場合、FR 用オブジェクトが混在すると警告を出力します。

ターゲット CPU が FR80 のときの FR 用オブジェクト混在警告は、オブジェクト混在チェックレベル指定オプション (-omcl) でエラー、または混在許可 (メッセージ出力なし) に変更できます。

リンカは、-cpu オプションで指定したターゲット CPU が FR の場合、FR80 用オブジェクトが混在すると必ずエラーを出力します。

< 注意事項 >

ターゲット CPU が FR80 の時に、FR 用オブジェクトを混在する場合、表 5.12-1 に示す FR と FR80 の命令非互換に十分注意してください。

■ FR と FR80 の命令非互換

FR と FR80 では「表 5.12-1 FR と FR80 の命令非互換」に示すように完全な命令互換ではありません。

このため、リンカでは -cpu オプションで指定したターゲット CPU が FR80 の、FR 用オブジェクトが混在すると警告を出力します。

-cpu オプションで指定したターゲット CPU が FR の場合、FR80 用オブジェクトが混在するとエラーを出力します。

表 5.12-1 FR と FR80 の命令非互換

非互換命令	FR	FR80
LDRES @Ri+,#u4		×
STRES #u4,@Ri		×
COPOP #u4,#CC,CRj,CRi		×
COPLD #u4,#CC,Rj,CRi		×
COPST #u4,#CC,CRj,Ri		×
COPSV #u4,#CC,CRj,Ri		×
SRCH0 Ri	×	
SRCH1 Ri	×	
SRCHC Ri	×	

: 命令あり × : 命令なし

■ FR/FR80 共通オブジェクト

FR/FR80 共通オブジェクトは、表 5.12-1 に示す命令がないオブジェクトです。

このため、FR/FR80 共通オブジェクトは、-cpu オプションで指定したターゲット CPU が FR と FR80 のどちらの場合もリンクすることができます。

FR/FR80 共通オブジェクトの作成方法の詳細は、『FR ファミリ SOFTUNE アセンブラ マニュアル V6 対応』を参照してください。

■ -cpu オプションと混在可能な CPU 一覧

表 5.12-2 に、-cpu オプションと混在可能なオブジェクト、ライブラリを示します。

表 5.12-2 -cpu オプションと混在可能な CPU 一覧

リンク時の -cpu オプション指定	オブジェクト、ライブラリ作成時の -cpu オプション		
	FR	FR80	FR/FR80 共通 オブジェクト
FR		×	
FR80			

: 同一ターゲットです。混在ではありません。

: FR/FR80 共通オブジェクトの為、混在可能です。

: リンク時に警告が出力されます。警告は、-omcl オプションでエラー、混在許可（メッセージ出力なし）に変更できます。

× : 混在できません。リンクはエラーを出力します。

第6章

リンカのオプション

リンカの各オプションについて詳しく説明します。

6.1 リンカのオプション一覧

6.2 リンカのオプション詳細

6.1 リンカのオプション一覧

リンカの動作を細かく指示するために、オプションがあります。

■ リンカのオプション一覧

リンカのオプション一覧を、表 6.1-1 に示します。

表 6.1-1 リンカのオプション一覧 (1 / 2)

		機 能	オプション	備 考
入出力制御オプション	出力ロードモジュールに関する指定	出力ロードモジュールファイル名指定	-o	デフォルト
		デバッグ情報出力指定	-g	
		デバッグ情報削除指定	-Xg	デフォルト
		絶対形式ロードモジュール出力指定	-a	デフォルト
		相対形式ロードモジュール出力指定	-r	
		パディングデータ指定	-p	デフォルト 0
		ROM 領域のフィル指定	-fill	
		外部シンボル情報出力指定	-symtab	
		外部シンボル情報出力抑止指定	-Xsymtab	デフォルト
	出力リストに関する指定	マップリストファイル名指定	-m	デフォルト
		マップリスト出力抑止指定	-Xm	
		リスト表示の名前省略解除	-dt	
		メモリ使用情報リストの出力指定	-mmi	
		デマングルシンボル名の出力抑止指定	-Xdemangle	
		デマングルシンボル名の出力指定	-demangle	デフォルト
		リスト行の桁数指定	-pw	デフォルト 80
		リスト 1 ページの行数指定	-pl	デフォルト 0
		ROM 領域のチェックサム指定	-cs	
	出力メッセージに関する指定	警告メッセージ出力レベル指定	-w	デフォルト 1
配置 / 結合オプション		ROM 領域指定	-ro	
		RAM 領域指定	-ra	
		セクション配置	-sc	
		セクショングループ指定	-gr	
		バックリンク指定	-pk	
		自動配置指定	-AL	デフォルト 0
ライブラリ制御オプション		検索ライブラリファイル指定	-l	
		ライブラリ検索パス指定	-L	
		シンボル個別のライブラリ指定	-el	
		ライブラリ検索抑止指定	-nl	
		デフォルトライブラリ検索抑止指定	-nd	

表 6.1-1 リンカのオプション一覧 (2 / 2)

	機 能	オプション	備 考
その他のリンク制御 オプション	エントリアドレス指定	-e	
	外部シンボル値の仮設定	-df	
	ターゲット CPU 指定	-cpu	必須
	CPU 情報ファイル指定	-cif	
	オブジェクト混在チェックレベル指定	-omcl	デフォルト 1
	デバッグ情報存在チェック抑止指定	-NCI0302LIB	
	内蔵 ROM/RAM 領域の自動設定	-set_rora	デフォルト
	内蔵 ROM/RAM 領域自動設定の抑止指定	-Xset_rora	
	ユーザ指定領域のチェック指定	-check_rora	
	ユーザ指定領域のチェック抑止指定	-Xcheck_rora	デフォルト
	セクション配置領域チェック指定	-check_locate	
	セクション配置領域チェック抑止指定	-Xcheck_locate	デフォルト
	サイズ 0 のセクション配置チェック指定	-check_size0_sec	
	サイズ 0 のセクション配置チェック抑止指定	-Xcheck_size0_sec	デフォルト
	プレリンク処理の抑止指定	-XPLNK	
絶対形式アセンブル リスト出力関連 オプション	相対形式アセンブルリスト入力ディレクトリ指定	-alin	
	絶対形式アセンブルリスト出力ディレクトリ指定	-alout	
	絶対形式アセンブルリスト出力指定	-als	
	絶対形式アセンブルリスト出力モジュール指定	-alsf	
	絶対形式アセンブルリスト出力抑止指定	-Xals	
	ROM/RAM, ARRAY リスト出力指定	-alr	
	ROM/RAM, ARRAY リスト出力モジュール指定	-alrf	
	ROM/RAM, ARRAY リスト出力抑止指定	-Xalr	
	ROM/RAM, ARRAY リストのシンボルとアドレス の表示位置指定	-na/-an	
オブジェクト内容リスト 出力関連オプション	外部シンボル相互参照情報リスト出力指定	-xl	
	外部シンボル相互参照情報リストファイル名指定	-xlf	
	外部シンボル相互参照情報リスト出力抑止指定	-Xxl	
	ローカルシンボルリスト出力指定	-sl	
	ローカルシンボルリストファイル名の指定	-slf	
	ローカルシンボルリスト出力抑止指定	-Xsl	
	セクション詳細マップリスト出力指定	-ml	
	セクション詳細マップリストファイル名の指定	-mlf	
	セクション詳細マップリスト出力抑止指定	-Xml	
共通オプション	デフォルトオプションファイル読み込み抑止指定	-Xdof	
	オプションファイル読み込み指定	-f	
	ヘルプメッセージ表示指定	-help	
	版数 / メッセージ出力指定	-V	
	版数 / メッセージ出力抑止	-XV	デフォルト
	終了メッセージ表示指定	-cmsg	
	終了メッセージ表示抑止指定	-Xcmsg	デフォルト
	ワーニング発生時の終了コードを 1 にする指定	-cwno	
	ワーニング発生時の終了コードを 0 にする指定	-Xcwno	デフォルト

6.2 リンカのオプション詳細

ここでは、リンカの各オプションについて説明します。

なお、リンテージキットで共通のオプションは、「第3章 共通オプション」で説明しています。

■ 出力モジュールに関するオプション

出力モジュールに関するオプションの詳細を「6.2.1 出力ロードモジュールファイル名指定 (-o)」～「6.2.9 外部シンボル情報出力抑止指定 (-Xsymtab)」で説明します。

■ 出力リストに関するオプション

出力リストに関するオプションの詳細を「6.2.10 マップリストファイル名の指定 (-m)」～「6.2.18 ROM領域のチェックサム指定 (-cs)」で説明します。

■ 出力メッセージに関する指定

出力メッセージに関するオプションの詳細を「6.2.19 警告メッセージ出力レベルの指定 (-w)」で説明します。

■ 配置 / 結合オプション

配置 / 結合に関するオプションの詳細を「6.2.20 ROM領域の指定 (-ro)」～「6.2.25 自動配置指定 (-AL)」で説明します。

■ ライブラリ制御オプション

ライブラリ制御オプションの詳細を「6.2.26 検索ライブラリファイルの指定 (-l)」～「6.2.30 デフォルトライブラリ検索の抑止指定 (-nd)」で説明します。

■ その他のリンク制御オプション

その他のリンク制御に関するオプションの詳細を「6.2.31 エントリアドレスの指定 (-e)」～「6.2.45 プレリンク処理の抑止指定 (-XPLNK)」で説明します。

■ 絶対形式アセンブルリスト出力関連オプション

絶対形式アセンブルリスト出力に関するオプションの詳細を「6.2.46 相対アセンブルリスト入力ディレクトリ指定 (-alin)」～「6.2.54 ROM/RAM, ARRAY リストのシンボルとアドレスの表示位置指定 (-na, -an)」で説明します。

■ オブジェクト内容リスト出力関連オプション

オブジェクト内容リスト出力に関するオプションの詳細を「6.2.55 外部シンボル相互参照情報リスト出力指定 (-xl)」～「6.2.63 セクション詳細マップリスト出力抑止指定 (-Xml)」で説明します。

6.2.1 出力ロードモジュールファイル名指定 (-o)

リンク結果のロードモジュールファイル名を指定します。このオプションが指定されないときは、最初に入力したファイル名から出力ファイル名を作ります。

■ 出力ロードモジュールファイル名指定 (-o)

【記述形式】

-o <ロードモジュールファイル名>	(デフォルト)
--------------------	---------

【パラメータ】

<ロードモジュールファイル名>

出力ロードモジュールファイル名です。

【説明】

リンク結果の出力ロードモジュールファイル名を指定します。

本オプション指定がない場合は、リンクが最初に入力したファイル名をもとに、リンクモードに応じた拡張子を付加した名称で出力ファイルを作ります。

リンクモード	デフォルト拡張子
絶対形式出力 (-a オプション)	.abs
相対形式出力 (-r オプション)	.rel

<ロードモジュールファイル名>の指定で拡張子を省略した場合も、リンクモードの違いにより同様の拡張子が付加されます。

【例1】

```
flnk911s putc.obj,getc.obj
```

ファイル名 putc.abs でロードモジュールファイルを作成します。

【例2】

```
flnk911s *.obj -o outfile
```

ファイル名 outfile.abs でロードモジュールファイルを作成します。

この例のように、入力オブジェクトファイル指定にワイルドカードを使用する場合には、本オプションを使用して出力ファイル名を明確にしておくことをお勧めします。

【例3】

```
flnk911s *.obj -o outfile.
```

ファイル名 outfile. でロードモジュールファイルを作成します。

ファイル名の最後にピリオドがある場合は、拡張子ありとみなします。

【例4】

```
flnk911s *.obj -r -o outfile.rel
```

ファイル名 outfile.rel で相対形式ロードモジュールファイルを作成します。

6.2.2 デバッグ情報の出力指定 (-g)

C/C++ コンパイラやアセンブラで、デバッグ情報の出力指定をして作られたオブジェクトモジュールには、デバッガで利用するためのデバッグ情報が含まれています。

デバッグ情報をリンク後も継承して使用するためには、-g オプションを指定します。

■ デバッグ情報の出力指定 (-g)

【記述形式】

```
-g
```

【パラメータ】

なし

【説明】

リンカは、入力するオブジェクトモジュールファイル、または相対形式ロードモジュールファイルにデバッグ情報が含まれている場合、デフォルト出力ではデバッグ情報を削除します。

出力ロードモジュールファイル中にデバッグ情報を残したい場合に、本オプションを使用します。

リンカは、デバッグ情報を新たに作り出すことはありませんので、入力ファイルにデバッグ情報が含まれていない場合は、指定することに意味はありません。

デバッグ時にシンボリックデバッグを行いたい場合には、C/C++ コンパイラ、アセンブラ、リンカのすべてのツールで -g オプションの指定をしてください。

【例】

図 6.2-1 デバッグ情報の出力指定例

```
flnk911s -f rllnk.opt b1 b2
```

rllnk.opt	
-r	# relocatable LM output
-g	# debug info.
-pw 100	# page width
-o rel1.rel	# output filename

6.2.3 デバッグ情報の削除指定 (-Xg)

C/C++ コンパイラやアセンブラで、デバッグ情報の出力指定をして作られたオブジェクトモジュールには、デバッガで利用するためのデバッグ情報が含まれています。

デバッグ情報をリンク後に取り去るためには、-Xg オプションを指定します。または、-g オプションの指定を行わないようにします。

■ デバッグ情報の削除指定 (-Xg)

【記述形式】

-Xg

(デフォルト)

【パラメータ】

なし

【説明】

リンカは、デフォルト出力ではデバッグ情報を削除しますので、本オプションを指定する必要はありません。

オプションファイルを利用したリンクにおいて、オプションファイル中に -g オプションがある場合など、-g オプションを無効にするときに使用します。

【例】

図 6.2-2 デバッグ情報の削除指定例

```
flnk911s -f rllnk.opt a1 a2 a3 -Xg -o a123.rel
```

rllnk.opt	
-r	# relocatable LM output
-g	# debug info.
-pw 100	# page width
-o rel1.rel	# output filename

6.2.4 絶対形式ロードモジュールの出力指定 (-a)

-a オプションは、リンカの最終目的ファイルである、絶対形式のロードモジュールを作成することを指示するオプションです。

■ 絶対形式ロードモジュールの出力指定 (-a)

【記述形式】

-a

(デフォルト)

【パラメータ】

なし

【説明】

絶対形式のロードモジュールファイル出力を指示します。

リンカのデフォルト出力は絶対形式ですので、通常使用することはありません。-r オプション指定を打ち消し -a 指定に変更する場合に指定します。

絶対形式の出力ファイルは、以下の名前で作成されます。

- -o オプション指定がないとき：最初に指定された入力ファイルの拡張子を ".abs" とした名称。
- -o オプション指定があるとき：指定された名称。拡張子指定がないときは、".abs" を付加した名称。

【例】

```
flnk911s a1 a2 a3 -r -o a123.abs -a
```

コマンドラインの途中の -r オプションを無効にします。

出力ロードモジュールファイルは、絶対形式で作成されます。

6.2.5 相対形式ロードモジュールの出力指定 (-r)

-r オプションは、再入力可能な相対形式ロードモジュールを作成することを指示するオプションです。相対形式のロードモジュールは、アドレス解決を行わないで複数のモジュールを 1 つのファイルにまとめた形式を持っています。

■ 相対形式ロードモジュールの出力指定 (-r)

【記述形式】

```
-r
```

【パラメータ】

なし

【説明】

相対形式のロードモジュールファイル出力を指示します。

リンカのデフォルト出力（絶対形式）を変更するときに指定します。

-a オプションのあとに -r オプションを指定すると、-a オプションを無効にすることができます。

相対形式のロードモジュールは、アドレス解決を行わないで複数のオブジェクトモジュールを 1 つのファイルにまとめた形式です。リンカへの再入力が可能です。以降のリンク処理で指定する入力ファイル数を少なくできます。リロケータブルロードモジュール中のモジュールを差し替えることはできません。

このオプションが指定された場合、絶対形式アセンブルリスト、オブジェクト内容リストに関するオプションはすべて無視され、それらのファイルは出力されません。

出力ファイルは、以下の名前で作成されます。

- -o オプション指定がないとき：最初に指定された入力ファイルの拡張子を ".rel" とした名称。
- -o オプション指定があるとき：指定された名称。拡張子指定がないときは、".rel" を付加した名称。

【例】

```
flnk911s a1 a2 a3 -r -o a123.rel
```

リンカの出力オブジェクトを相対形式にします。

< 注意事項 >

最初に指定された入力ファイルが、拡張子 ".rel" であるとき、出力ファイル名が同じになります。この場合、入力ファイルの内容は保存されませんので、不都合がある場合には、-o オプションで出力ファイル名の指定をしてください。

6.2.6 パディングデータ指定 (-p)

-p オプションは、境界整合などで発生したオブジェクトの隙間を指定した値で埋めるオプションです。

本オプションは、絶対形式のロードモジュールを作成する場合にのみ有効です。相対形式ロードモジュールを作成する場合には意味を持ちません。

■ パディングデータ指定 (-p)

【記述形式】

-p <値>	(デフォルト : 0)
--------	---------------

【パラメータ】

< 値 >

1 バイトのデータ

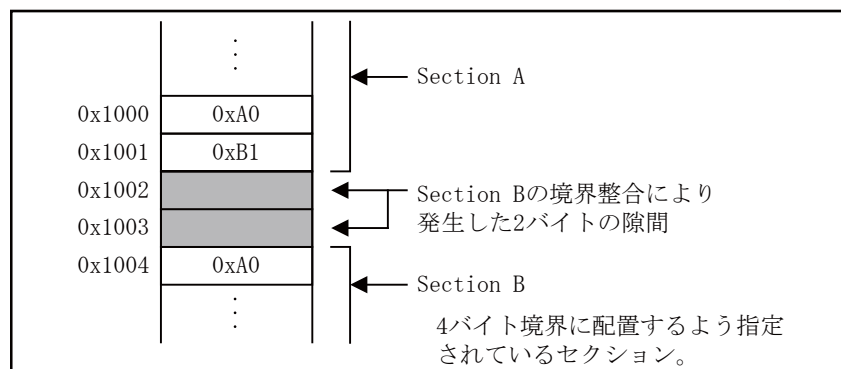
【説明】

絶対形式ロードモジュールファイルを作成時に、セクション配置により発生したオブジェクトの隙間を埋めるバイトの値を指示します。

0 ~ 255 の値が指定可能です。

リンカでセクション配置を行う際、セクションの境界整合などの条件により、図 6.2-3 のようにオブジェクトデータの存在しない数バイトの隙間が発生することがあります。

図 6.2-3 セクション配置時の境界整合により隙間の発生した例



本オプションはセクション配置により発生した隙間を特定のデータ値で埋める場合に使用します。

リンカは本オプションが指定されていない場合、オブジェクトデータの存在しない隙間を 0 で埋めます。

【例】

```
flnk911s a1 a2 a3 -p 255 ... 隙間を 255 で埋めます。
flnk911s a1 a2 a3 -p 0xff ... 隙間を 255 で埋めます。
flnk911s a1 a2 a3 -p 0xaa ... 隙間を 170 で埋めます。
```

6.2.7 ROM 領域のフィル指定 (-fill)

-fill オプションは指定した領域を指定した値で埋めるオプションです。

■ ROM 領域のフィル指定 (-fill)

【記述形式】

```
-fill <スタートアドレス>/<エンドアドレス>,<フィル値>/<[{8|16|32}]>/<[{B|L}]>
```

【パラメータ】

<スタートアドレス>

フィル対象領域の先頭アドレス：フィルを行う領域の開始アドレスを指定します。

<エンドアドレス>

フィル対象領域の終了アドレス：フィルを行う領域の終了アドレスを指定します。

<フィル値>

フィル値：指定した領域を埋める値を指定します。

<[{8|16|32}]>

ビット幅：フィル値のビット幅を指定します。ビット幅指定省略時のビット幅は 8 ビットです。

<[{B|L}]>

エンディアン：フィル値のエンディアンを指定します (B：ビッグエンディアン / L：リトルエンディアン)。エンディアン指定省略時のエンディアンはビッグエンディアンです。

【説明】

スタートアドレスとエンドアドレスで指定した領域を指定したフィル値で埋めます。

-fill オプションは、必要に応じて複数指定できます。

フィル値には領域を埋める値のほかに、フィル値のビット幅 (8/16/32 ビット) やフィル値のエンディアン (ビッグエンディアン / リトルエンディアン) を指定できます。

フィル値を指定した場合、リンクは指定された領域をオブジェクトデータとして絶対形式ロードモジュール (ABS) に出力します。

フィル対象領域が重なった場合、重なった領域は後に指定した -fill オプションのフィル値で埋められます。

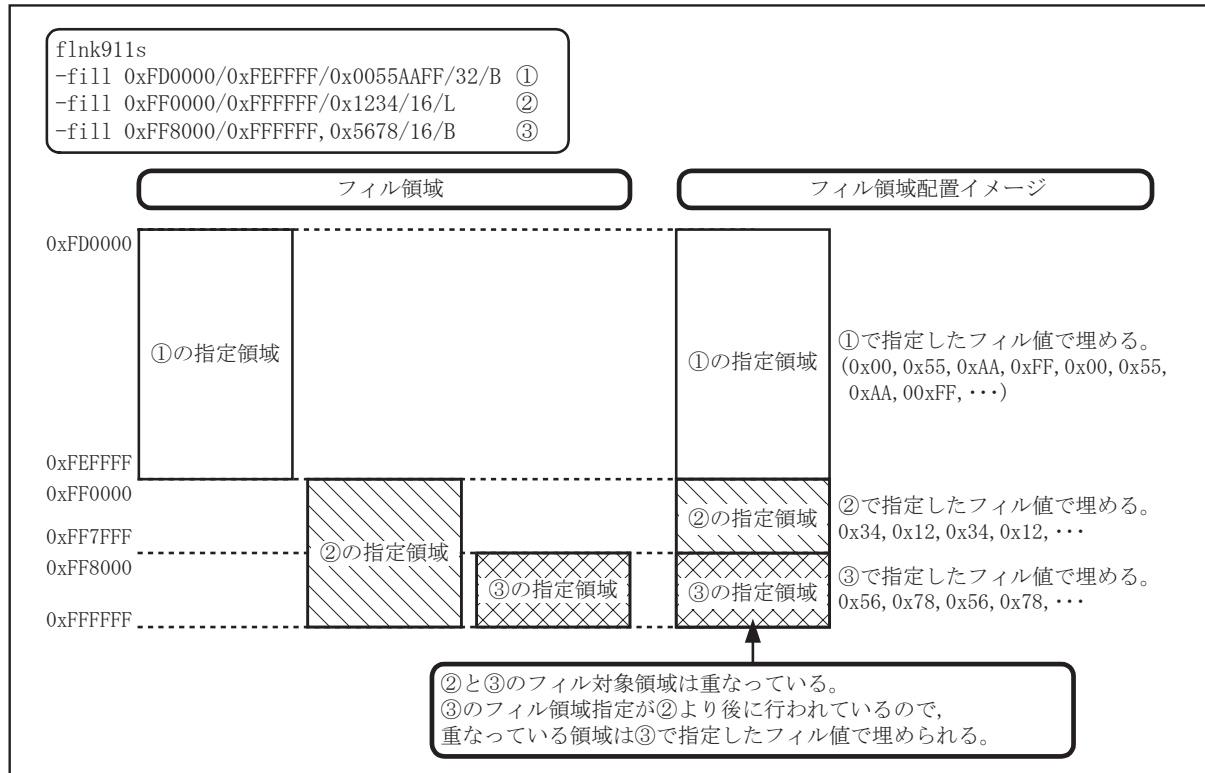
本オプションは、絶対形式ロードモジュール作成時のみ有効となります。

【例】

以下のような指定を行った場合、フィル領域は図 6.2-4 のように配置されます。

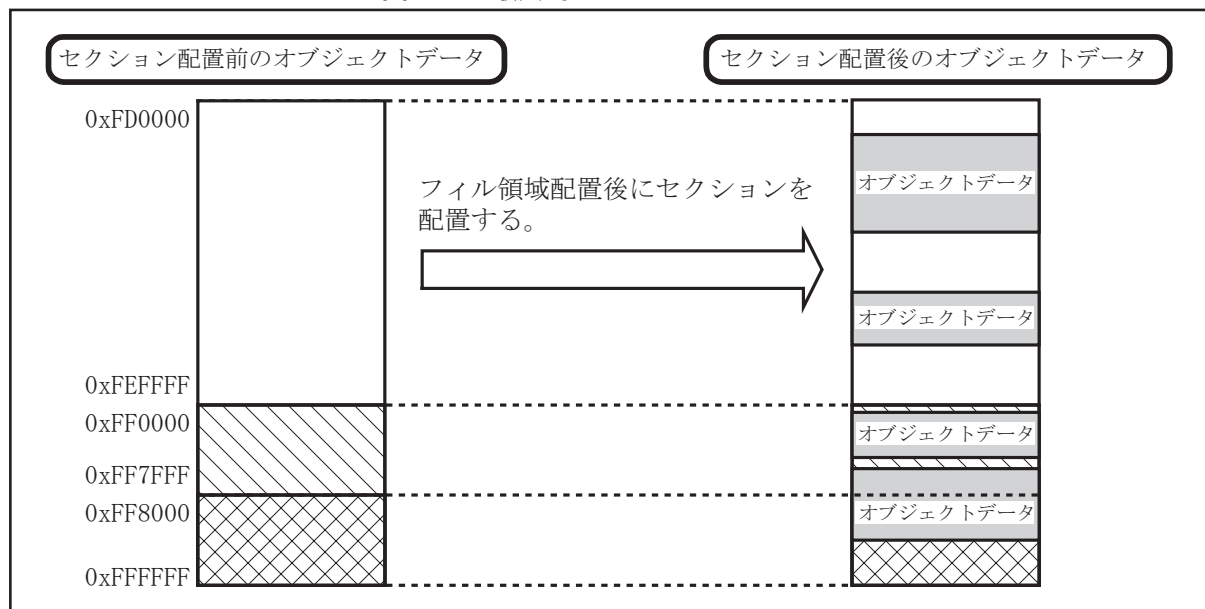
```
flnk911s -fill 0xFD0000/0xFEFFFFFF,0x0055AAFF/32/B -fill
0xFF0000/0xFFFFFFFF,0x1234/16/L
-fill 0xFF8000/0xFFFFFFFF,0x5678/16/B
```

図 6.2-4 フィル領域指定例



フィル指定領域が配置された後、セクションが配置されます。

図 6.2-5 最終的なオブジェクトデータ



6.2.8 外部シンボル情報出力指定 (-symtab)

-symtab オプションは、絶対形式ロードモジュールを作成する際に、外部シンボル情報をロードモジュールに出力することを指示するオプションです。

本オプションは、絶対形式のロードモジュールを作成する場合にのみ有効です。

相対形式ロードモジュールを作成する場合には意味を持ちません。

■ 外部シンボル情報出力指定 (-symtab)

【記述形式】

-symtab

【パラメータ】

なし

【説明】

絶対形式のロードモジュールファイル出力する際に、ロードモジュール内に外部シンボル情報を含めることを指示します。

外部シンボル情報とは、リンカが外部シンボル値の解決に使用する情報で、SOFTUNE Workbench でデバッグを行う際に必要となるデバッグ情報とは、別の情報です。

リンカのデフォルトでは、外部シンボル情報を絶対形式ロードモジュールに出力しません。

【例】

```
flnk911s a1 a2 a3 -symtab
```

6.2.9 外部シンボル情報出力抑止指定 (-Xsyntab)

-Xsyntab オプションは、絶対形式ロードモジュールを作成する際に、外部シンボル情報をロードモジュールへ出力することを抑止するオプションです。

本オプションは、絶対形式のロードモジュールを作成する場合にのみ有効です。

相対形式ロードモジュールを作成する場合には意味を持ちません。

■ 外部シンボル情報出力抑止指定 (-Xsyntab)

【記述形式】

-Xsyntab

【パラメータ】

なし

【説明】

絶対形式のロードモジュールファイル出力する際に、ロードモジュール内に外部シンボル情報を含めないことを指示します。

外部シンボル情報とは、リンカが外部シンボル値の解決に使用する情報で、SOFTUNE Workbench でデバッグを行う際に必要となるデバッグ情報とは、別の情報です。

リンカのデフォルトです。

-syntab オプションを無効にしたい場合に使用します。

【例】

```
flnk911s a1 a2 a3 -syntab -Xsyntab
```


6.2.10 マップリストファイル名の指定 (-m)

リンカが出力するマッピングリストファイルのファイル名を指定します。
このオプションが指定されないときは、出力ロードモジュールファイル名からファイル名を作ります。

■ マップリストファイル名の指定 (-m)

【記述形式】

-m	<マップリストファイル名>	(デフォルト)
----	---------------	---------

【パラメータ】

<マップリストファイル名>

出力マップリストファイル名パラメータは省略できません。

【説明】

リンクは、デフォルトでマップリストファイルを出力します。このとき、出力ロードモジュールファイルの拡張子を、".mp1" とした名称のファイルが作られます。

-m オプションは、デフォルトのマップリストファイル名を変更したいときに使用します。

-Xm オプションの後に -m オプションを指定すると、-Xm オプションを無効にすることができます。

【例】

```
flnk911s a1 a2 a3 -r -o a123.rel -m a123.map
```

リンカが出力するマップリストファイル名を `a123.map` にします。

6.2.11 マップリスト出力の抑止指定 (-Xm)

リンカにマップリストファイルを出力しないことを指示します。このオプションを指定しないときは、必ずマップリストファイルが作成されます。

■ マップリスト出力の抑止指定 (-Xm)

【記述形式】

-Xm

【パラメータ】

なし

【説明】

マップリストファイルの出力の抑止を行います。

-m オプションの後に -Xm オプションを指定すると、-m オプションを無効にすることができます。

-Xm オプションの指定により、-dt, -pw, -pl オプションは無効になります。

【例】

```
flnk911s a1 a2 a3 -r -o a123.rel -Xm
```

マップリストファイルの作成は行わないようにします。

6.2.12 リスト表示の名前の省略解除 (-dt)

リンカのマップリスト，オブジェクト内容リストには，セクション名やシンボル名など名前の表示がありますが，デフォルトでのリスト出力では見やすさを考慮して，長い名前は途中までを表示するようにしています。

このオプションは，名前を省略せずに出力することを指示します。

マップリストの表示フォーマットについては，「7.2 リンクリストファイル」を参照してください。

■ リスト表示の名前の省略解除 (-dt)

【記述形式】

-dt

【パラメータ】

なし

【説明】

マップリスト，オブジェクト内容リストで表示されるシンボル名およびセクション名を途中で省略せずに表示します。この場合，1 つのシンボル名，セクション名は数行に渡って表示されます。

デフォルト出力では 80 文字程度の表示ができますが，リスト 1 行の表示桁数を小さな値に設定したときに表示可能な文字数も減りますので，シンボル名およびセクション名の省略表示が起こりえます。

【例】

```
flnk911s a1 a2 a3 -o a123.abs -m a123.map -dt
```

使用されているシンボル名およびセクション名は，省略することなくリストに表示します。

6.2.13 メモリ使用情報リストの出力指定 (-mmi)

リンカが出力するマップリストファイルの中にメモリ使用情報を出力するように指示します。

■ メモリ使用情報リストの出力指定 (-mmi)

【記述形式】

-mmi

【パラメータ】

なし

【説明】

リンカが、デフォルトで出力するマップリストファイルの中に、オプションで指定された ROM 領域と RAM 領域の使用状況を示すメモリ使用情報リストを出力します。

使用可能領域、使用した領域や領域内ギャップの位置とサイズなどの情報を表示します。

【例】

```
flnk911s lomp00 im_lnk -mmi -ra RAM1=0x1000/0x1FFF, RAM2=0x2000/0x23FF  
-ro ROM1=0xBC000/0xBCFFF, ROM2=0xBD000/0xBFFF -AL 2
```

< 注意事項 >

-mmi が指定されていても、以下の場合は出力を行いません。

- メモリ領域の設定がない (-ra, -ro の指定がない)。
 - マップリストファイルの出力が有効になっていない。
-

6.2.14 リスト表示のデマングルシンボル名の出力抑止指定 (-Xdemangle)

リンカのマップリスト中シンボルリスト部に表示される外部シンボル名のデマングルシンボル名表示を抑止します。

■ リスト表示のデマングルシンボル名の出力抑止指定 (-Xdemangle)

【記述形式】

-Xdemangle

【パラメータ】

なし

【説明】

リンカのマップリスト中シンボルリスト部に表示される外部シンボル名のデマングル名の表示を抑止します。

● デマングルシンボル名とは

テンプレート関数名などでは、ほかのシンボル名と重ならないように C++ コンパイラ内部でマングル処理されたシンボル名が使用されます。

デマングルシンボル名は上記のマングル処理されたシンボル名を変換し、ユーザに識別可能にしたものです。

【例】

```
flnk911s a1 a2 a3 -o a123.abs -m a123.abs -Xdemangle
```

6.2.15 リスト表示のデマングルシンボル名の出力指定 (-demangle)

リンカのマップリスト中シンボルリスト部に表示される外部シンボル名のデマングルシンボル名表示します。

■ リスト表示のデマングルシンボル名の出力指定 (-demangle)

【記述形式】

-demangle	(デフォルト)
-----------	---------

【パラメータ】

なし

【説明】

リンカのマップリスト中シンボルリスト部に表示される外部シンボル名のデマングル名を表示します。

● デマングルシンボル名とは

テンプレート関数名などでは、ほかのシンボル名と重ならないように C++ コンパイラ内部でマングル処理されたシンボル名が使用されます。

デマングルシンボル名は上記のマングル処理されたシンボル名を変換し、ユーザに識別可能にしたものです。

【例】

```
flnk911s a1 a2 a3 -o a123.abs -m a123.abs -demangle
```

6.2.16 リスト行の桁数指定 (-pw)

リンカ出力のマッピングリスト, オブジェクト内容リストは, デフォルトでは 1 行に 80 桁まで表示します。このオプションは, 1 行の表示桁数を変更するときに指定します。

■ リスト行の桁数指定 (-pw)

【記述形式】

-pw <桁数>	(デフォルト : 80)
----------	--------------

【パラメータ】

<桁数>

1 行に表示する桁数。80 ~ 1023 の範囲で指定します。

【説明】

リンクリストファイル, オブジェクト内容リストの 1 行の長さの指定を行います。

この指定がない場合は, 80 桁になります。

70 ~ 79 の値を指定した場合, リンカは I0311L を出力後, 桁数を 80 桁にします。

【例】

```
flnk911s a1 a2 a3 -o a123.abs -m a123.map -dt -pw 100
```

リスト行の表示桁数を 100 にします。

< 注意事項 >

以下のリストは -pw オプションで 1 行の桁数を指定できます。

- リンクマッピングリスト (.mp1)
- セクション詳細マッピングリスト (.mpm)

以下のリストは -pw オプションで 1 行の桁数を指定できません。

- 絶対形式アセンブルリスト (.als)
- 外部シンボル相互参照情報リスト (.mpx)
- ローカルシンボル情報リスト (.mps)

6.2.17 リスト 1 ページの行数指定 (-pl)

リンカ出力のマッピングリスト, オブジェクト内容リストは, デフォルトではページ制御を行わないで表示します。

このオプションは, 1 ページの表示行数を変更するときに指定します。

■ リスト 1 ページの行数指定 (-pl)

【記述形式】

<code>-pl <行数></code>	(デフォルト : 0)
-----------------------------	-------------

【パラメータ】

<行数>

1 ページに表示する行数。0 または, 20 ~ 255 の範囲で指定します。

【説明】

リンクリストファイル, オブジェクト内容リストの 1 ページの行数指定を行います。

この指定がない場合はページ制御を行いません。

0 の指定は, ページ制御を行わないようにするものです。

【例】

```
flnk911s a1 a2 a3 -o a123.abs -m a123.map -dt -pl 64 -pw 100
```

リスト 1 ページの表示行数を 64 にします。

< 注意事項 >

以下のリストは -pl オプションで 1 ページの行数を指定できます。

- リンクリスト (.mp1)
- セクション詳細マッピングリスト (.mpm)

以下のリストは -pl オプションで 1 ページの行数を指定できません。

- 絶対形式アセンブルリスト (.als)
 - 外部シンボル相互参照情報リスト (.mpx)
 - ローカルシンボル情報リスト (.mps)
-

6.2.18 ROM 領域のチェックサム指定 (-cs)

-cs オプションは指定した領域のチェックサムを行うオプションです。
 チェックサムの演算方法は、単純加算 (SUM) と巡回冗長検査 (CRC) があります。
 チェックサム結果はマップリストに出力されます。

■ ROM 領域のチェックサム指定 (-cs)

【記述形式】

```
-cs <スタートアドレス>/<エンドアドレス>[, <スタートアドレス><エンドアドレス>, ...],  

  <{SUM16 [= 補数形式] | SUM32 [= 補数形式] | CRC16 [= 生成多項式値] | CRC32  

  [= 生成多項式値]}>,<フィル値>
```

【パラメータ】

<スタートアドレス>

チェックサム演算対象領域の先頭アドレス：チェックサム演算を行う領域の開始アドレスを指定します。

<エンドアドレス>

チェックサム演算対象領域の終了アドレス：チェックサム演算を行う領域の終了アドレスを指定します。

<{SUM16[= 補数形式] | SUM32[= 補数形式] | CRC16[= 生成多項式値] | CRC32[= 生成多項式値]}>

チェックサム演算方法を指定します。

16 ビット単純加算 (SUM16), 32 ビット単純加算 (SUM32), 16 ビット巡回冗長検査 (CRC16), 32 ビット巡回冗長検査 (CRC32) が指定できます。

単純加算 (SUM16, SUM32) を指定した場合、補数形式を指定できます。

補数形式：0：補数なし, 1：1 の補数, 2：2 の補数

補数形式省略時は、補数なしとしてチェックサム演算を行います。

巡回冗長検査 (CRC16, CRC32) を指定した場合、生成多項式を指定できます。

生成多項式を省略した場合、生成多項式は以下値としてチェックサム演算を行います。

CRC16 の場合：0x8005(CRC-ANSI)

CRC32 の場合：0x104C11DB7(CRC-32 ITU-T)

<フィル値>

チェックサム演算対象領域の中でオブジェクトデータが存在しない箇所がある場合、フィル値で埋められます。

チェックサム演算対象領域が重なった場合、後で指定した -cs オプションのフィル値が有効となります。

【説明】

スタートアドレスとエンドアドレスで指定した領域のチェックサム演算を行います。
複数の領域をまとめてチェックサム演算する場合、1 つの -cs オプションにまとめて指定してください。

各領域のチェックサム演算を別々に行う場合は、それぞれ別の -cs オプションで領域を指定してください。

```
-cs 0xFE0000/0xFE8FFF,0xFF8000/0xFFFFFFFF,SUM32=2,0xAA
```

0xFE0000/0xFE8FFF と 0xFF8000/0xFFFFFFFF の領域をまとめてチェックサム演算を行います。

```
-cs 0xFE0000/0xFE8FFF,SUM32=2,0xAA -cs 0xFF8000/0xFFFFFFFF,SUM32=2,0xAA
```

0xFE0000/0xFE8FFF の領域のチェックサム演算と 0xFF8000/0xFFFFFFFF の領域のチェックサム演算を別々に行います。

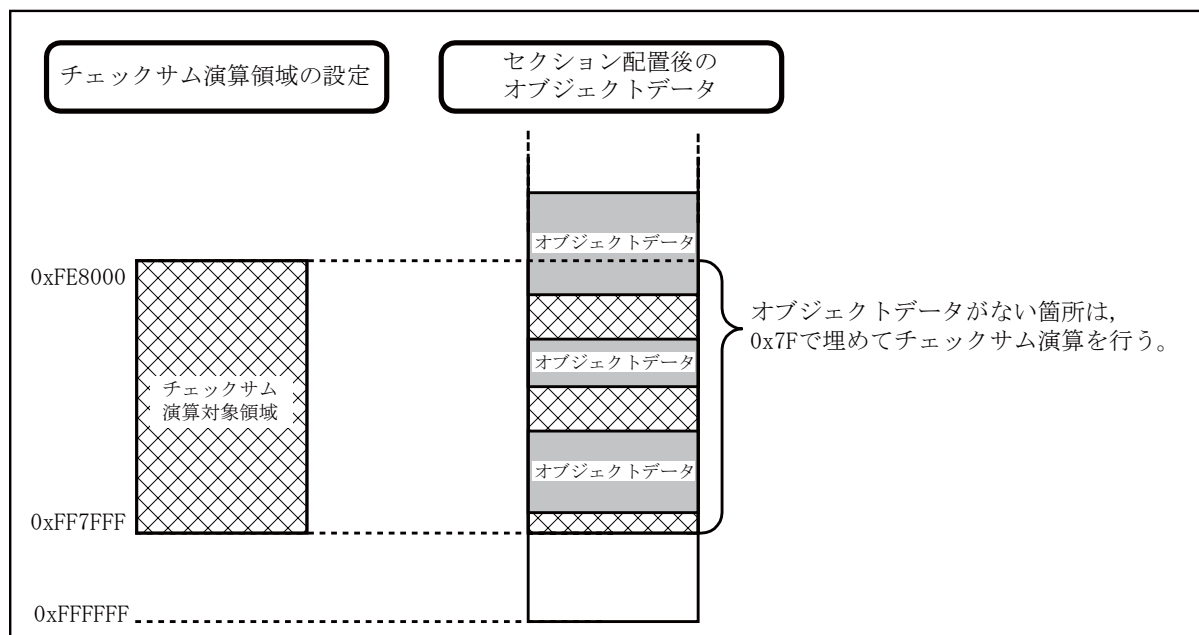
-cs オプションはチェックサム演算の方法、オブジェクトコードが存在しない部分を埋める値を設定できます。

【例 1】

```
flnk911s -cs 0xFE8000/0xFF7FFF,SUM32,0x7F
```

32 ビット単純加算で 0xFE8000/0xFF7FFF の領域のチェックサム演算を行います。

チェックサム演算対象領域でオブジェクトデータがない箇所は 0x7F で埋めます。

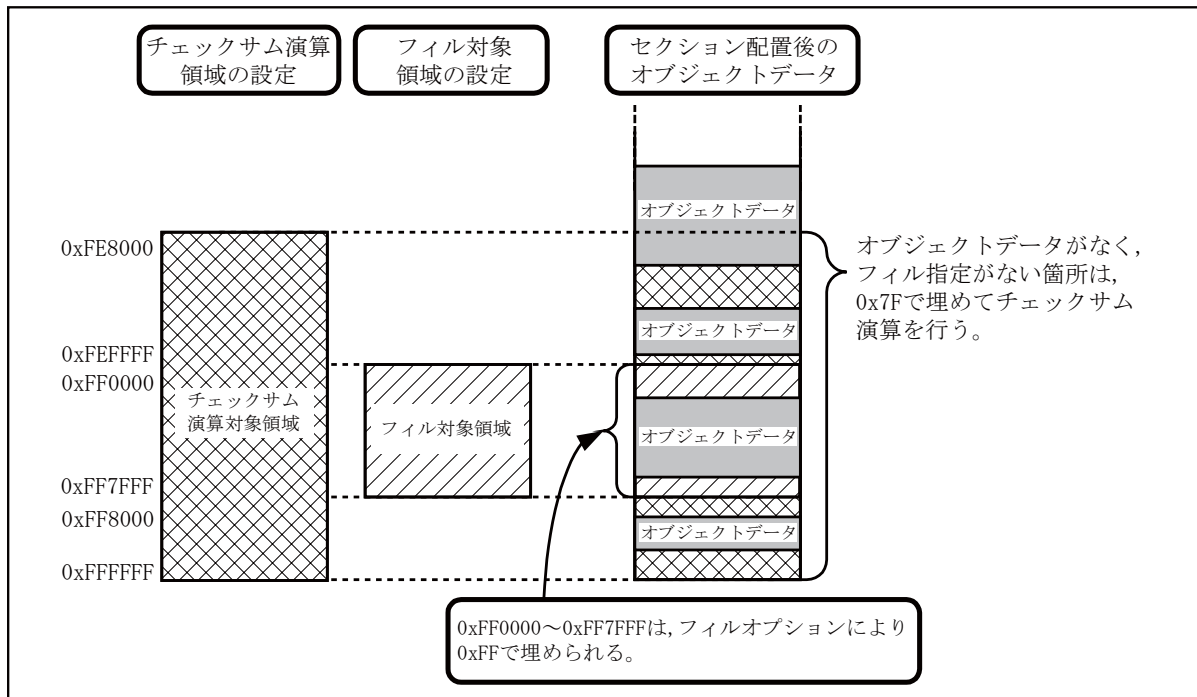


【例 2】

```
flnk911s -fill 0xFF0000/0xFF7FFF,0xFF/8 -cs 0xFE8000/0xFFFFFFFF,SUM16=2,0x7F
```

16 ビット単純加算 (補数 2) で 0xFE8000/0xFFFF の領域のチェックサム演算を行います。

-cs と -fill を同時に指定した場合、オブジェクトデータがなくフィル指定されていない箇所を 0x7F で埋めてチェックサム演算を行います。

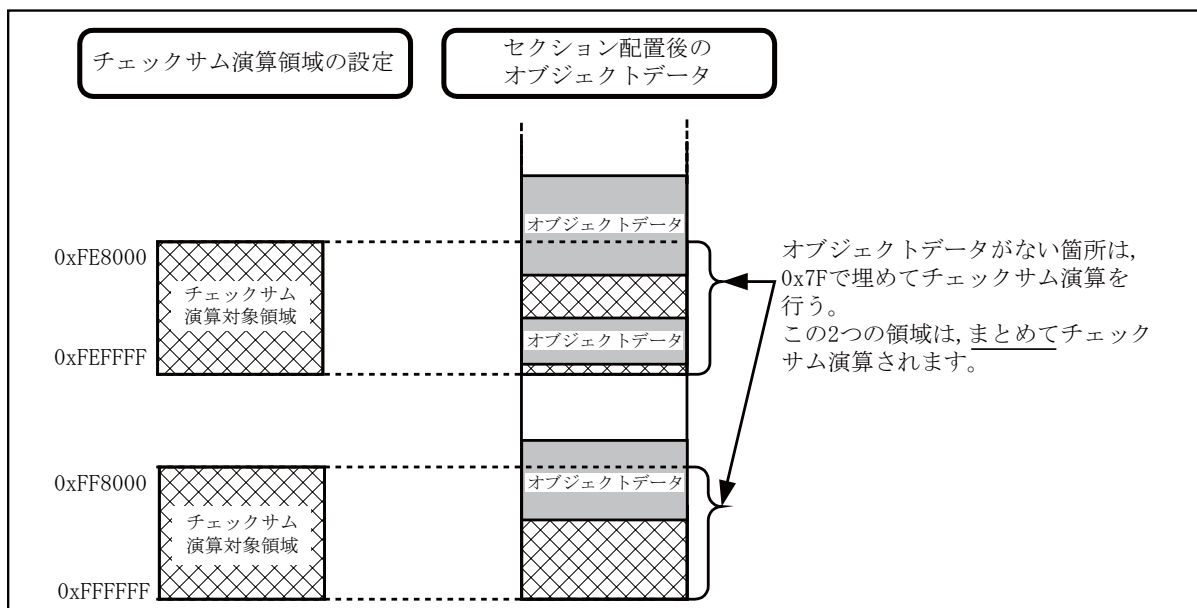


【例 3】

```
flnk911s -cs 0xFE8000/0xFEFFFF,0xFF8000/0xFFFFFFF,CRC16,0x7F
```

16 ビット巡回冗長検査 (CRC16) で 0xFE8000/0xFEFFFF, 0xFF8000/0xFFFFFFF の領域のチェックサム演算をします。

0xFE8000/0xFEFFFF, 0xFF8000/0xFFFFFFFの領域をまとめてチェックサム演算をします。

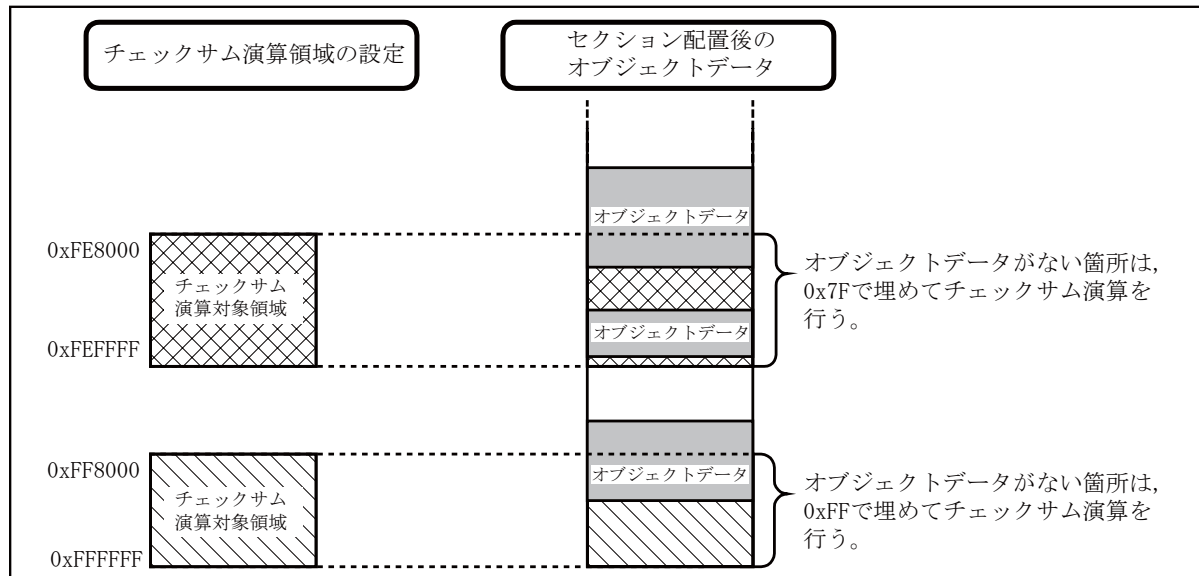


【例 4】

```
flnk911s -cs 0xFE8000/0xFEFFFF,CRC16,0x7F -cs 0xFF8000/0xFFFFFFF,CRC32,0xFF
```

16 ビット巡回冗長検査 (CRC16) で 0xFE8000/0xFEFFFF の領域のチェックサム演算を行います。

32 ビット巡回冗長検査 (CRC32) で 0xFF8000/0xFFFFFFF の領域のチェックサム演算を行います。



【例 5】

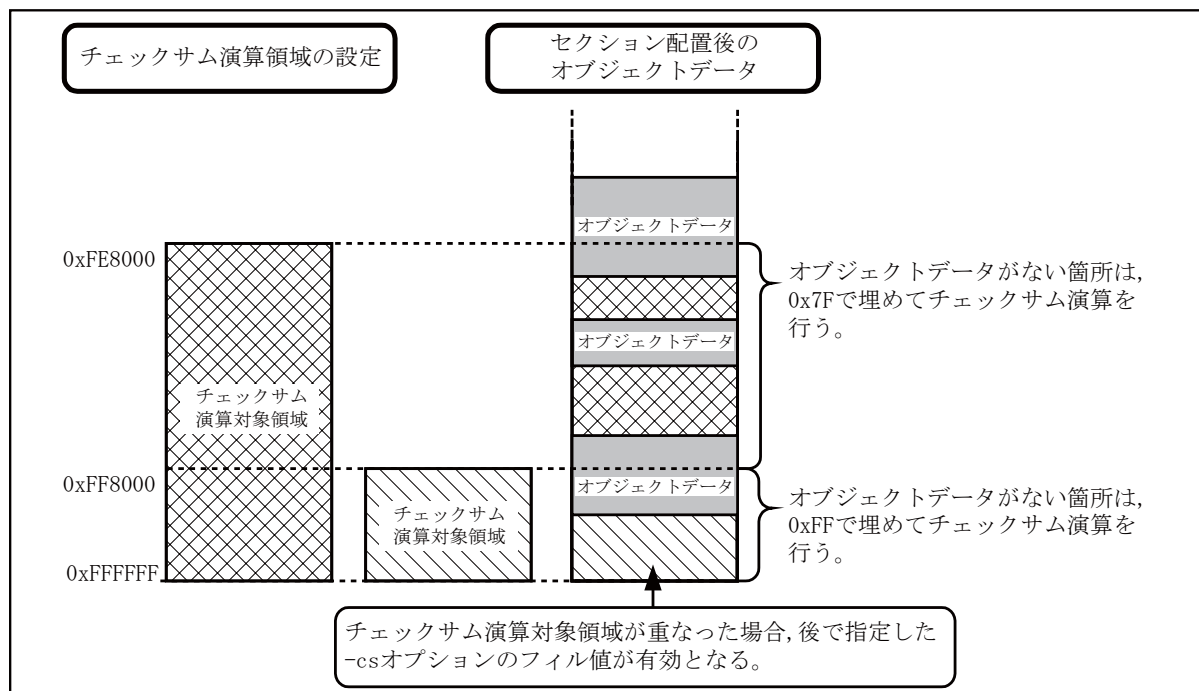
```
flnk911s -cs 0xFE8000/0xFFFFFFF,SUM16=1,0x7F -cs 0xFF8000/0xFFFFFFF,  
SUM32=2,0xFF
```

16 ビット単純加算で 0xFE8000/0xFFFFFFF の領域のチェックサム演算を行います。

32 ビット単純加算で 0xFF8000/0xFFFFFFF の領域のチェックサム演算を行います。

0xFE8000/0xFF7FFF の領域で、オブジェクトデータがない箇所は 0x7F で埋められます。

0xFF8000/0xFFFFFFF の領域で、オブジェクトデータがない箇所は 0xFF で埋められます。



6.2.19 警告メッセージ出力レベルの指定 (-w)

警告メッセージの出力レベルを設定します。リンカの警告メッセージを完全に抑止するときや、リンカの動作状態の確認をするときに使用します。

■ 警告メッセージ出力レベルの指定 (-w)

【記述形式】

<code>-w <数値></code>

【パラメータ】

<数値>

警告レベルとして 0, 1 または 2 を指定します。

【説明】

警告レベルのメッセージ出力を抑止したり、より詳細なメッセージ出力を行うように得たい情報をコントロールします。

0 : 警告レベルのメッセージを出力しません。

1 : 通常のチェックです。(デフォルト)

2 : 通常無視してよいレベルのものと、単なるリンカの動作内容を通知するメッセージまで出力します。

詳細は、「付録 A リンケージキットのエラーメッセージ」を参照してください。

【例】

```
flnk911s a1 a2 a3 -o a123.abs -w 2 -Xm
```

すべてのメッセージを出力します。

6.2.20 ROM 領域の指定 (-ro)

プログラムで使用する予定の ROM 領域を定義することにより、セクションの配置を行うときのアドレス指定を簡略化します。

また、プログラムサイズのチェックも可能にします。

■ ROM 領域の指定 (-ro)

【記述形式】

```
-ro <領域名> = <スタートアドレス> / <エンドアドレス>  
[, <領域名> = <スタートアドレス> / <エンドアドレス> ] ...
```

【パラメータ】

<領域名>

設定するアドレス領域を示す名前

<スタートアドレス>

設定するアドレス領域の先頭アドレス

<エンドアドレス>

設定するアドレス領域の最終アドレス

【説明】

ROM 領域の定義を行います。領域は、必要に応じていくつでも定義できます。

スタートアドレスとエンドアドレスを指定して、その領域に名前を付けます。

このオプションで定義した領域名は、セクション配置オプションで使います。

-ro オプションの定義だけではリンカの動作には何の効果も及ぼさないので、必ずセクション配置オプションでのアドレス指定パラメータに、ここで定義した領域名を使うことになります。

【例】

```
flnk911s *.obj -o ap.abs -ro RomA=0x1000/0x2FFF -sc code=RomA ...
```

セクション名 code のセクションを、0x1000 ~ 0x2FFF のアドレスに配置することを指定します。

0x1000 番地から順に配置し、0x2FFF 番地を超えた場合には警告メッセージを出力します。

6.2.21 RAM 領域の指定 (-ra)

プログラムで使用する予定の RAM 領域を定義することにより、セクションの配置を行うときのアドレス指定を簡略化します。

また、プログラムサイズのチェックも可能にします。

■ RAM 領域の指定 (-ra)

【記述形式】

```
-ra <領域名> = <スタートアドレス> / <エンドアドレス>
[, <領域名> = <スタートアドレス> / <エンドアドレス> ] ...
```

【パラメータ】

<領域名>

設定するアドレス領域を示す名前

<スタートアドレス>

設定するアドレス領域の先頭アドレス

<エンドアドレス>

設定するアドレス領域の最終アドレス

【説明】

RAM 領域の定義を行います。領域は、必要に応じていくつでも定義できます。

スタートアドレスとエンドアドレスを指定して、その領域に名前を付けます。

このオプションで定義した領域名は、セクション配置オプションで使います。

-ra オプションの定義だけではリンカの動作には何の効果も及ぼさないので、必ずセクション配置オプションでのアドレス指定パラメータに、ここで定義した領域名を使うことになります。

【例】

```
flnk911s *.obj -o ap.abs -ra RamD=0x0100/0x01FF -sc data=RamD ...
```

セクション名 data のセクションを、0x0100 ~ 0x01FF のアドレスに配置することを指定します。

0x0100 番地から順に配置し、0x01FF 番地を超えた場合には警告メッセージを出力します。

6.2.22 セクション配置順 / アドレスの指定 (-sc)

セクション配置の開始アドレスおよび配置順をリンカに指示します。

■ セクション配置順 / アドレスの指定 (-sc)

【記述形式】

```
-sc <セクション名リスト> [/ <内容種別> ]
      [ = { <アドレス> | <領域名> } ] [, ... ]
```

【パラメータ】

<セクション名リスト>

セクション名, またはセクショングループ名リストセクション名の指定には, ワイルドカードが使えます。複数指定するときは, + 記号で連結します。

<内容種別>

code, data, stack, const, IO

<アドレス>

配置先頭アドレス

<領域名>

ROM/RAM 指定オプションで指定した領域名

【説明】

セクション配置の順序指定, および配置アドレスの指定を行います。

セクション配置の順序は, パラメータに記述された順序に従います。

アドレスまたは領域名指定がない場合は, 0 番地から配置します。

セクション名の先頭に @ マークを付けると, 実行時にデータを ROM から RAM に転送して動作する ROM → RAM 転送セクションの ROM 側のアドレス指定になります。

ワイルドカードを使用する場合は, 以下の文字で囲んでください。ただし, オプションファイル中ではワイルドカードを " (ダブルクォート) で囲まないでください。

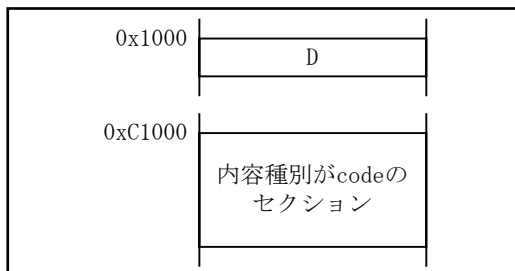
【例 1】

```
flnk911s *.obj -o ap.abs -sc */code=0xC1000,D=0x1000 ...
```

図 6.2-6 にこの場合の配置例を示します。

0xC1000 番地から内容種別が code のセクションを配置し, 0x1000 番地からセクション D を配置します。

図 6.2-6 セクションの配置例 1



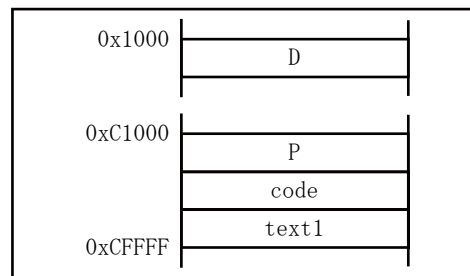
【例 2】

```
flink911s *.obj -o ap.abs -ro ROM=0xC1000/0xCFFFF -ra RAM=0x1000/0x13FF
-sc P+code+text1=ROM,D=RAM
```

図 6.2-7 にこの場合の配置例を示します。

ROM/RAM オプションを使用したセクション配置指定の場合、配置終了アドレスのチェックが行えます。

図 6.2-7 セクションの配置例 2



<セクション名リスト> に使用できるワイルドカード文字は "*" のみで、記述パターンは以下の 4 とおりです。

パターン	例	説明
*	-sc */code	内容種別が code であるすべてのセクションとマッチします。
マッチする	ab_1, code_1, XXsect など	
マッチしない	なし	
~ *	-sc ab_*/code	内容種別が code であり、セクション名の先頭 3 文字が "ab_" であるすべてのセクションとマッチします。
マッチする	ab_1, ab_XX, ab_ など	
マッチしない	aab_XX, ab など	
* ~	-sc *_1/code	内容種別が code であり、セクション名の最後の 2 文字が "_1" であるすべてのセクションとマッチします。
マッチする	ab_1, XX_1, _1 など	
マッチしない	ab_11, _ など	
~ * ~	-sc ab_*_1/code	内容種別が code であり、セクション名の先頭 3 文字が "ab_" で、かつ最後の 2 文字が "_1" であるすべてのセクションとマッチします。
マッチする	ab_XX_1, ab_ _1 など	
マッチしない	aab_XX_11, ab_1 など	

< 注意事項 >

ワイルドカード指定に一致するセクションには、ほかの sc オプションで指定されたセクションや絶対アドレス (セクションの配置属性が ABS) を持つセクションは含まれません。

```
-sc *=RAM -sc CODE=RAM
```

上記のような指定で、ワイルドカード指定は、"CODE" 以外の再配置可能なセクション (セクションの配置属性が REL) すべてと一致します。

6.2.23 セクショングループの指定 (-gr)

複数のセクションをユーザの目的に応じて1つに連結し、グループ名をつけます。セクションの配置指定時にこのグループ名を使用することで、複数セクションをまとめて扱えます。

■ セクショングループの指定 (-gr)

【記述形式】

```
-gr <グループ名> = <セクション名リスト> [/ <内容種別> ] [, ... ]
```

【パラメータ】

<グループ名>

グループ化する複数セクションの総称

<セクション名リスト>

グループ化するセクション名を記述します。

ワイルドカードが使用できます。

複数指定するときは、+ 記号で連結します。

<内容種別>

code, data, stack, const, IO

【説明】

グループ化するセクションの指定と、グループ内でのセクションの配置の順序指定を行います。

セクション配置の順序は、パラメータに記述された順序に従います。

グループ名は、セクション名や他のグループ名と重複しないユニークなものである必要があります。

1つのグループに属するセクションが、ほかのグループに属してはいけません。

ワイルドカードを使用する場合は、"(ダブルクォート)で囲んでください。ただし、オプションファイル中ではワイルドカードを"(ダブルクォート)で囲まないでください。

【例1】

```
flnk911s *.obj -o ap.abs -ro ROM=0xC1000/0xCFFFF -ra RAM=0x1000/0x13FF ...
-gr romG=P+CODE+text1 -sc romG=ROM,D=RAM ...
```

グループ化オプションを使用した場合は、セクション名を幾つも指定するかわりに、その全体をグループ名で代表することができます。

【例2】

```
flnk911s *.obj -o ap.abs -gr cdgrp=*/code -sc cdgrp=0xC1000
```

内容種別がcodeであるセクションをすべて結合してcdgrpというグループ名を付けます。それを-scオプションで0xC1000番地に配置します。

6.2.24 バックリンク指定 (-pk)

セクションの配置情報のアライン指定 (バウンダリ調整) を無効にし, 1 バイトバウンダリでセクションの結合配置を行います。

アプリケーションプログラムの実行が正しく行われなくなる危険性があります。使用にあたっては十分に注意をしてください。

■ バックリンク指定 (-pk)

【記述形式】

<code>-pk <セクション名></code>

【パラメータ】

<セクション名>

バックリンクしたいセクション名を記述する。

【説明】

セクションの配置情報のアライン指定 (バウンダリ調整) を無効にし, 続けて (バウンダリ調整数 1 で) リンクします。

-pk 指定を行う際には, 十分注意してください。本オプションの指定によっては, プログラムの動作が遅くなったり, 動作しなくなる場合もあります。

-pk オプションは構造体メンバのアライン調整を行うものではありません。

【例】

```
flnk911s *.obj -o ap.abs -pk code ...
```

6.2.25 自動配置指定 (-AL)

セクションの自動配置の指定を行います。

■ 自動配置指定 (-AL)

【記述形式】

-AL { 0 1 2 }

【パラメータ】

< 0 >

自動配置を行いません。(デフォルト)

< 1 >

領域内にアブソリュートセクションが存在するとき、そのセクションと重複しないように配置します。

< 2 >

セクションの属性から、ROM領域に配置すべきか、RAM領域に配置すべきかを判断し、それぞれの領域内の空き領域に配置します。

【説明】

セクションの自動配置の指定を行います。

- パラメータ 1 の場合 :-ra, -ro オプションで指定した領域内へセクション配置を行うとき、アブソリュートセクションが存在していれば、配置アドレスが重ならないように、リロケートブルセクションの配置を行います。このとき、アライメント値およびサイズの大いセクションから順に配置し、空き領域が最小となるような最適配置を行います。
- パラメータ 2 の場合 :-sc オプションで配置指定が行われなかったセクションに対して、セクションの属性から、ROM領域に配置すべきか、RAM領域に配置すべきかを判断し、それぞれの領域内の空き領域に配置します。

「5.6 セクションの自動配置」を合わせて参照してください。

【例】

```
flnk911s -AL 1 -ro ROM=0x1000/0x1FFF -sc code1+code2+code3=ROM ...
```

ただし、各セクションは以下のようなものとします。

- code1 : リロケートブル, サイズ = 0x18
- code2 : リロケートブル, サイズ = 0x10
- code3 : リロケートブル, サイズ = 0x30
- AbsSec : アブソリュート, アドレス範囲 = 0x1010 ~ 0x1017

この場合の、リンクマップ例を図 6.2-8 に示します。

図 6.2-8 リンクマップ例

S_Addr. -E_Addr	Size	SectionType	AL	Sec
00001000-0000100F	00000010	CODE P R-XI	02 REL	code2
00001010-00001017	00000008	CODE N R-XI	00 ABS	AbsSec
00001018-00001047	00000030	CODE P R-XI	02 REL	code3
00001048-0000105F	00000018	CODE P R-XI	02 REL	code1

< 注意事項 >

-AL 1 が指定されていても、以下の場合は自動配置を行いません。

- 領域の設定がない (-ra, -ro の指定がない)。
- -sc のアドレス指定で領域名を使用していない。
- 指定した領域内に配置されるアブソリュートセクションがない。

-AL 2 を指定した場合、以下の場合はエラーになりリンク処理を行いません。

- 領域の設定がない (-ra, -ro の指定がない)。
通常、リンカは CPU 情報ファイルから -ra, -ro を自動的に設定します。
- -w 2 を指定すると自動配置機能が働いた場合メッセージが出力されます。

6.2.26 検索ライブラリファイルの指定 (-l)

デフォルトライブラリ以外に検索すべきライブラリを指示します。複数のライブラリファイルがある場合、検索順に指定します。

■ 検索ライブラリファイルの指定 (-l)

【記述形式】

`-l <ライブラリファイル名> [, ...]`

【パラメータ】

<ライブラリファイル名>

検索するライブラリファイル名を記述します。パス名付きの指定も可能。ワイルドカード使用可能。

【説明】

指定された順に、ライブラリファイルを検索します。

デフォルトライブラリより先に、ここで指定したライブラリが検索されます。

パス名なしでライブラリファイルを指定した場合は以下の順に検索します。

1. -L オプションで指定したディレクトリ
2. 環境変数 LIB911 で指定のディレクトリ
3. 環境変数 FETOOL より導かれるシステムライブラリのディレクトリ

カレントディレクトリは検索しません。カレントディレクトリを検索対象にしたいときは、-L オプションまたは環境変数 LIB911 にピリオド (.) を指定してください。

-nl オプションより後で、-l オプションを指定することで -nl オプションを無効にすることができます。

ワイルドカードを使用する場合は、" (ダブルクォート) で囲んでください。ただし、オプションファイル中ではワイルドカードを" (ダブルクォート) で囲まないでください。

【例】

```
flnk911s *.obj -o ap.abs -l ../lib/com.lib,libu
```

```
flnk911s *.obj -o ap.abs -l p*.lib
```

ファイル名の先頭1文字が "p" であるすべてのライブラリファイルが検索対象になります。

6.2.27 ライブラリ検索パスの指定 (-L)

ライブラリファイルを検索するパス名の指定を行います。

■ ライブラリ検索パスの指定 (-L)

【記述形式】

<code>-L <ライブラリパス名> [, ...]</code>

【パラメータ】

<ライブラリパス名>

ライブラリファイルを格納したパス名

【説明】

-L オプションで指定したライブラリファイルが、どのディレクトリにあるのかをリンクに指示します。

通常は、本オプションを指定しなくてもよいように環境変数 LIB911 を設定しておきます。

環境変数 LIB911 で指定したパスには、C コンパイラに付属の C ライブラリを格納しますので、特別にユーザが作成したライブラリを別ディレクトリで管理したいような場合に、-L オプションを使用してください。

複数のパスが指定された場合は、指定順に検索します。

ライブラリファイルは、まずここで指定されたパスから探し、なければ環境変数 LIB911 で指定されたパス、環境変数 FETOOL より導かれるシステムライブラリパスの順で検索します。

ライブラリファイルの指定で、パス名まで含めて指定された場合は、そのパスだけを検索します。

-nl オプションがある場合は、ライブラリ検索を行わないので -L オプションは無効になります。

【例】

```
flnk911s *.obj -o ap.abs -L C:\usr\usrlib -l com.lib,libu
```

6.2.28 シンボル個別のライブラリの指定 (-el)

外部参照シンボルの値の解決に使用するライブラリファイルを特定できます。

■ シンボル個別のライブラリの指定 (-el)

【記述形式】

<code>-el <シンボル名リスト> = <ライブラリファイル名> [, ...]</code>

【パラメータ】

<シンボル名リスト>

外部参照シンボル名を記述します。

複数シンボルを指定する場合は, / で区切ります。

<ライブラリファイル名>

検索対象とするライブラリファイル名

パス名付きの指定も可能です。なお, ワイルドカードは使用できません。

【説明】

本オプションで指定した外部参照シンボルの値の解決に使用するライブラリファイルを特定します。

同じ外部定義シンボル名を含むモジュールが, 複数ライブラリに存在しているのが分かっており, リンカが標準で行うライブラリ検索順では希望しないモジュールが結合されてしまうときに使用します。

この機能を使用せざるを得ないのは, 複数のライブラリを使用するときにライブラリの作成方法に問題があるためです。思わぬ障害混入の原因を含んでいますので, できれば使用しないことが望ましいものです。

ライブラリファイルの作り直しを検討してください。

なお, パス名指定なしでライブラリファイル名を指定した場合の検索ディレクトリは, -I オプションと同じです。

【例】

```
flnk911s *.obj -o sp.abs -L C:\usr\usr\lib -l libu,sublib -el sym1=sublib
```


6.2.29 ライブラリ検索の抑止指定 (-nl)

ライブラリファイルの検索を行わないことを指示します。

■ ライブラリ検索の抑止指定 (-nl)

【記述形式】

-nl

【パラメータ】

なし

【説明】

ライブラリファイルの検索を行わないようにします。

【例 1】

```
flnk911s -L C:\usr\usr\lib -l libu,sublib *.obj -o ap.abs -nl
```

先に指定した -L, -l, -el オプションを無効にし、デフォルトライブラリも含めて、ライブラリの検索を行わないようにします。

【例 2】

```
flnk911s -l lib1 *.obj -o ap.abs -nl -l lib2
```

一度指定した lib1.lib をキャンセルして、lib2.lib を検索するように指定し直すことができます。

この例のように、複数の -l オプションの間に -nl オプションを指定すると、-nl の指定以前に指定していた -l オプションはすべてキャンセルされます。しかし、-L オプション指定、-el オプション指定、-nd オプション指定および、デフォルトライブラリの検索は -nl 指定以前の設定状態になります。

したがって、この例の場合は、lib2.lib とデフォルトライブラリを検索します。

6.2.30 デフォルトライブラリ検索の抑止指定 (-nd)

デフォルトライブラリとは、C/C++ コンパイラが使用することを仮定したライブラリファイルであり、オブジェクトファイル中にそのライブラリファイル名が設定されています。

このデフォルトライブラリファイルの検索を行わないことを指示します。

■ デフォルトライブラリ検索の抑止指定 (-nd)

【記述形式】

-nd

【パラメータ】

なし

【説明】

デフォルトライブラリファイルの指定を無効とし、検索しないようにします。

【例】

```
flnk911s -L C:\usr\usrlib -l libu,sublib *.obj -o ap.abs -nd
```

-l オプションで指定したライブラリのみを検索し、デフォルトライブラリの検索は行わないようにします。

6.2.31 エントリアドレスの指定 (-e)

ユーザプログラムの実行開始アドレスを、外部定義シンボルで指定します。

■ エントリアドレスの指定 (-e)

【記述形式】

<code>-e <シンボル名></code>

【パラメータ】

<シンボル名>

エントリポイントのシンボル名

外部定義シンボル以外は指定できません。

【説明】

ユーザプログラムの実行開始アドレスを、外部定義シンボルで指定したものに変更します。

実行開始アドレスは、アセンブラの .end 疑似命令で指定できます。

エントリポイントは、シミュレータデバッガで、実行開始時の PC (プログラムカウンタ) の初期値として設定されます。

【例】

```
flnk911s *.obj -o ap.abs -e ProgStart
```

6.2.32 外部シンボル値の仮設定 (-df)

ユーザプログラム未定義シンボルの値を強制的に定義します。

■ 外部シンボル値の仮設定 (-df)

【記述形式】

<code>-df <シンボル名> = { <数値> <外部定義シンボル名> }</code>

【パラメータ】

<シンボル名>

外部参照シンボルのシンボル名

<数値>

定義する値

<外部定義シンボル名>

値の定義されている外部シンボル名

【説明】

未定義の外部参照シンボルの値を強制的に定義します。

リンカはリロケーション解決にこの値を用いてオブジェクトデータを作成しますが、出力する絶対形式ロードモジュールファイル中のシンボル情報には影響を与えません。

デバッガなどでロードした場合、ここで指定したシンボル名は未定義のままです。

【例】

```
flnk911s -L \usr\usr\lib -l libu,sublib *.obj -o ap.abs -df Sym1=100
```

Sym1 が定義されていないとき、値を 100 とします。

6.2.33 ターゲット CPU 指定 (-cpu)

ターゲット CPU の指定を行います。

リンクを行うプログラムのターゲット CPU を MB 番号で指定します。

■ ターゲット CPU 指定 (-cpu)

【記述形式】

<code>-cpu < MB 番号 ></code>

【パラメータ】

< MB 番号 >

ターゲット CPU の MB 番号

【説明】

リンクを行うプログラムのターゲット CPU を MB 番号で指定します。

【例】

```
flnk911s *.obj -o ap.abs -cpu MB91110
```

< 注意事項 >

リンク処理を実行する際には、本オプションでターゲット CPU の指定が必要です。

6.2.34 CPU 情報ファイル指定 (-cif)

リンカで使用する CPU 情報ファイルを指定します。

■ CPU 情報ファイル指定 (-cif)

【記述形式】

<code>-cif < CPU 情報ファイル名 ></code>

【パラメータ】

< CPU 情報ファイル名 >

リンカで使用する CPU 情報ファイル

【説明】

リンカで使用する CPU 情報ファイルを指定します。

【例】

```
flnk911s *.obj -o ap.abs -cpu MB91110
-cif C:\Softune6\lib\911\cpu_info\MB91110.csv
```

< 注意事項 >

SOFTUNE Tools は、CPU 情報ファイルを参照して、CPU に関する情報を取得します。
関連するツール間で異なる CPU 情報ファイルを参照した場合、作成するプログラムに問題が発生する可能性があります。

SOFTUNE Tools に標準で添付されている CPU 情報ファイルは、以下の場所にあります。

インストール先ディレクトリ \lib\911\911.csv

コンパイラとアセンブラパックを異なるディレクトリにインストールしている場合には、
各ツールに対して同一の CPU 情報ファイルを参照するように、-cif で指定してください。

6.2.35 オブジェクト混在チェックレベル指定 (-omcl)

-cpu オプションで指定したターゲット CPU が FR80 の時に FR 用オブジェクトが混在した場合の動作を指定します。動作は混在許可 (メッセージ出力なし), 警告, エラーが指定できます。

■ オブジェクト混在チェックレベル指定 (-omcl)

【記述形式】

```
-omcl <数値>
```

【パラメータ】

<数値>

警告レベルとして 0, 1 または 2 を指定します。

【説明】

-cpu オプションで指定したターゲット CPU が FR80 の時に FR 用オブジェクトが混在した場合の動作を指定します。動作は混在許可 (メッセージ出力なし), 警告, エラーが指定できます。

0 : 混在が発生してもメッセージを出力しません。

(混在許可)

1 : 混在が発生した場合には, 警告メッセージを出力します。

(デフォルト)

2 : 混在が発生した場合には, エラーメッセージを出力します。

(混在不可)

-omcl オプションでは, -cpu オプションで指定したターゲット CPU が FR の時に FR80 用オブジェクトが混在した場合の動作は変更できません。

-cpu オプションと混在可能な CPU については, 「5.12 FR 用オブジェクトと FR80 用オブジェクトの混在」を参照してください。

【例】

- -omcl オプション指定がない場合 (デフォルト動作)

```
flnk911s -cpu MB91680 a1 a2 a3 module_fr.obj
```

```
*** W1312L: 互換性のない CPU タイプのモジュールがあります (module_fr.obj)
```

混在を警告として扱います。

- -omcl 0 を指定した場合

```
flnk911s -cpu MB91680 a1 a2 a3 module_fr.obj -omcl 0
```

混在を検出しません

- -omcl 1 を指定した場合

```
flnk911s -cpu MB91680 a1 a2 a3 module_fr.obj -omcl 1
```

```
*** W1312L: 互換性のない CPU タイプのモジュールがあります (module_fr.obj)
```

第6章 リンカのオプション

混在を警告として扱います。

- -omcl 2 を指定した場合

```
flnk911s -cpu MB91680 a1 a2 a3 module_fr.obj -omcl 2
```

```
*** E4312L: 互換性のないCPUタイプのモジュールがあります (module_fr.obj)
```

混在をエラーとして扱います。

6.2.36 デバッグ情報存在チェック抑止指定 (-NCI0302LIB)

ライブラリファイルのモジュールに対するデバッグ情報存在チェックを抑止します。

■ デバッグ情報存在チェック抑止指定 (-NCI0302LIB)

【記述形式】

```
-NCI0302LIB
```

【パラメータ】

なし

【説明】

デバッグ情報出力指定 (-g) と警告レベル 2 (-w 2) を指定してリンカを動作させると、リンカはデバッグ情報が存在しないモジュールに対して、次のインフォメーションを出力します。

I0302L: デバッグ情報が存在しません (モジュール名)

本オプションを指定すると、リンカはライブラリファイルから抽出したモジュールに対して、上記インフォメーションメッセージの出力を行いません。

【例】

```
flnk911s -cpu MB91F155 -g -w 2 test.obj -l lib911.lib
```

```
*** I0302L: デバッグ情報が存在しません (C:\Softune6\lib\911\lib911.lib)
```

```
*** I0302L: デバッグ情報が存在しません (C:\Softune6\lib\911\lib911.lib)
```

```
.  
.
.
```

インフォメーション I0302L が出力されます。

```
flnk911s -cpu MB91F155 -g -w 2 test.obj -l lib911.lib -NCI0302LIB
```

インフォメーション I0302L は出力されません。

6.2.37 内蔵 ROM/RAM 領域の自動設定 (-set_rora)

CPU 情報ファイルからターゲット CPU の内蔵 ROM/RAM 領域情報の設定を行います。

■ 内蔵 ROM/RAM 領域の自動設定 (-set_rora)

【記述形式】

```
-set_rora
```

【パラメータ】

なし

【説明】

CPU 情報ファイルからターゲット CPU の内蔵 ROM/RAM 領域情報の設定を行います。本オプションを指定した際リンカは、CPU 情報ファイルから該当するチップの内蔵 ROM/RAM 領域を自動的に設定します。

リンカは、ROM/RAM 領域を次の名前で設定します。

- ROM 領域の場合：_ROM_*_

上記 * には、低位アドレスの領域順に 1 から順番に番号が入ります。領域が 1 つしかない場合は、"_ROM_1_" となります。

- RAM 領域の場合：_RAM_*_

上記 * には、低位アドレスの領域順に 1 から順番に番号が入ります。領域が 1 つしかない場合は、"_RAM_1_" となります。

これら名前は、-sc オプションで使用可能です。

【例】

```
flnk911s *.obj -o ap.abs -cpu MB91110 -set_rora
```

6.2.38 内蔵 ROM/RAM 領域自動設定の抑止指定 (-Xset_rora)

CPU 情報ファイルからターゲット CPU の内蔵 ROM/RAM 領域情報の設定を抑止します。

■ 内蔵 ROM/RAM 領域自動設定の抑止指定 (-Xset_rora)

【記述形式】

-Xset_rora

【パラメータ】

なし

【説明】

CPU情報ファイルからターゲットCPUの内蔵ROM/RAM領域情報の設定を抑止します。

【例】

```
flnk911s *.obj -o ap.abs -cpu MB91110 -Xset_rora
```

6.2.39 ユーザ指定領域のチェック指定 (-check_rora)

メモリマップが変わっていないかを MB 番号指定を変更するだけでチェックできます。

指定した ROM 領域, RAM 領域 (-ro, -ra オプション) が実際の内蔵 ROM, 内蔵 RAM のアドレスと合っているかチェックを行います。

シングルチップモードの場合にご使用ください。

■ ユーザ指定領域のチェック指定 (-check_rora)

【記述形式】

```
-check_rora
```

【パラメータ】

なし

【説明】

指定の ROM 領域, RAM 領域 (-ro, -ra オプション) が, 内蔵 ROM, 内蔵 RAM を超えていないかチェックを行います。

-ro/-ra オプションで指定した領域が内蔵 ROM, 内蔵 RAM に収まっていない場合に, 次の警告を出力します。

W1368L: -ro オプションで内蔵 ROM 領域の範囲外が指定されています (領域名)

W1369L: -ra オプションで内蔵 RAM 領域の範囲外が指定されています (領域名)

シングルチップモードをご使用の場合, このオプションと MB 番号を指定しておくと, その品種の内蔵 ROM, 内蔵 RAM のアドレスと合っているかチェックを行います。別品種にプログラムを移植する場合など, MB 番号を変更するだけでメモリマップが変わっていないかチェックできます。

また, 併せて -check_locate オプションを指定しておけば内蔵 ROM, 内蔵 RAM 内にプログラムが収まっているかのチェックも行えます。

【例】

```
flnk911s -cpu MB91F155 -check_rora -ro ROM = 0x00080800/0x000fffff
      -ra RAM1=0x00001000/0x00008fff, RAM2=0x00080000/0x000807ff ...
```

正常範囲指定 警告はありません。

```
flnk911s -cpu MB91154 -check_rora -ro ROM = 0x00080800/0x000fffff
      -ra RAM1=0x00001000/0x00008fff, RAM2=0x00080000/0x000807ff ...
```

W1368L: -ro オプションで内蔵 ROM 領域の範囲外が指定されています (ROM)

W1369L: -ra オプションで内蔵 RAM 領域の範囲外が指定されています (RAM1)

下線部の指定が範囲外のため警告が出力されます。

```
flnk911s -cpu MB91154 -check_rora -ro ROM = 0x000A0000/0x000fffff
```

```
-ra RAM1=0x00001000/0x00005fff, RAM2=0x00080000/0x000807ff ...
```

正常範囲指定 警告はありません。

< 注意事項 >

- ワーニング出力抑止 (-w 0) が指定されている場合でも本オプションが指定されている場合は警告が出力されます。
 - 本オプションは、絶対形式ロードモジュール作成時のみ有効です。
リロケータブルロードモジュール作成時は、本オプションは無視されます。
 - 本オプションは CPU 情報ファイル内の内蔵 ROM 情報および内蔵 RAM 情報を用いています。したがって、CPU 情報ファイル内に該当情報が存在しない場合には警告は出力されません。
-cpu オプションで正確な MB 番号指定を行ってください。
-

6.2.40 ユーザ指定領域のチェック抑止指定 (-Xcheck_rora)

指定した ROM 領域, RAM 領域 (-ro, -ra オプション) と内蔵 ROM 領域または内蔵 RAM 領域のチェックを抑止します。

-check_rora オプションを取り消す際に用います。

■ ユーザ指定領域のチェック抑止指定 (-Xcheck_rora)

【記述形式】

`-Xcheck_rora`

【パラメータ】

なし

【説明】

指定の ROM 領域, RAM 領域 (-ro, -ra オプション) と内蔵 ROM, 内蔵 RAM のアドレスのチェックを抑止します。

-check_rora オプションを取り消す際に用います。

【例】

```
flnk911s -cpu MB91154 -check_rora -ro ROM = 0x00080800/0x000fffff  
-ra RAM1=0x00001000/0x00008fff, RAM2=0x00080000/0x000807ff ...
```

W1368L: -ro オプションで内蔵 ROM 領域の範囲外が指定されています (ROM)

W1369L: -ra オプションで内蔵 RAM 領域の範囲外が指定されています (RAM1)

下線部の指定が範囲外のため警告が出力されます。

```
flnk911s -cpu MB91154 -check_rora -ro ROM = 0x00080800/0x000fffff  
-ra RAM1=0x00001000/0x00008fff, RAM2=0x00080000/0x000807ff  
-Xcheck_rora ...
```

下線部の指定が範囲外ですが, Xcheck_rora でチェック抑止を行っているため警告は出力されません。

6.2.41 セクション配置領域チェック指定 (-check_locate)

メモリ領域外に配置されていないことをチェックします。

指定の ROM 領域, RAM 領域 (-ro, -ra オプション) または, cpu 情報ファイル内の内蔵 ROM 情報, 内蔵 RAM 情報を元にセクション配置アドレスをチェックし, 領域外の配置に対して警告を出力します。

■ セクション配置領域チェック指定 (-check_locate)

【記述形式】

-check_locate

【パラメータ】

なし

【説明】

指定の ROM 領域, RAM 領域 (-ro, -ra オプション) または, cpu 情報ファイル内の内蔵 ROM/RAM 情報を元にセクション配置アドレスをチェックし, 領域外の配置に対して次の警告を出力します。

W1370L: ROM 領域外への配置です (セクション名)

W1371L: RAM 領域外への配置です (セクション名)

W1372L: RAM 領域または IO 領域外への配置です (セクション名)

W1373L: IO 領域外への配置です (セクション名)

以下にセクションタイプとチェック領域を示します。

セクション タイプ	チェック対象領域	
	-check_rora オプション指定あり	-check_rora オプション指定なし
ROM 領域に配置されるべきセクション		
CODE	-ro 指定領域外かつ内蔵 ROM 領域外に配置されているセクションに対して警告出力。	-ro 指定領域外に配置されているセクションに対して警告出力。
CONST		
ROM RAM 転送元セクション		
RAM 領域に配置されるべきセクション		
STACK	-ra 指定領域外かつ内蔵 RAM 領域外に配置されているセクションに対して警告出力。	-ra 指定領域外に配置されているセクションに対して警告出力。
ROM RAM 転送先セクション		
RAM 領域または IO 領域に配置されるべきセクション		
DATA	-ra 指定領域外かつ内蔵 RAM 領域、内蔵 IO 領域外に配置されているセクションに対して警告出力。	-ra 指定領域外に配置されているセクションに対して警告出力。
IO 領域に配置されるべきセクション		
IO	内蔵 IO 領域外に配置されているセクションに対して警告出力。	内蔵 IO 領域外に配置されているセクションに対して警告出力。

シングルチップモードでご使用の場合は、-check_rora と併せて指定することで、内蔵メモリ領域外にプログラムが配置されていないことをチェックできます。

また、その他の場合でも ROM 領域指定、RAM 領域指定 (-ro, -ra オプション) と併せて指定することでメモリ領域外にプログラムが配置されていないことをチェックできます。

【例】

下図のようなメモリマップで DATA タイプの DATA_A, DATA_B, DATA_C と CODE タイプの CODE_D, CODE_E, CODE_F と STACK タイプの STACK_G が配置されている場合のチェックは次のようになります。

```
flnk911s -cpu MB91154 -check_locate -ro ROM = 0x000A0000/0x000fffff
        -ra RAM1=0x00001000/0x00005fff, RAM2=0x00080000/0x000807ff ...
```

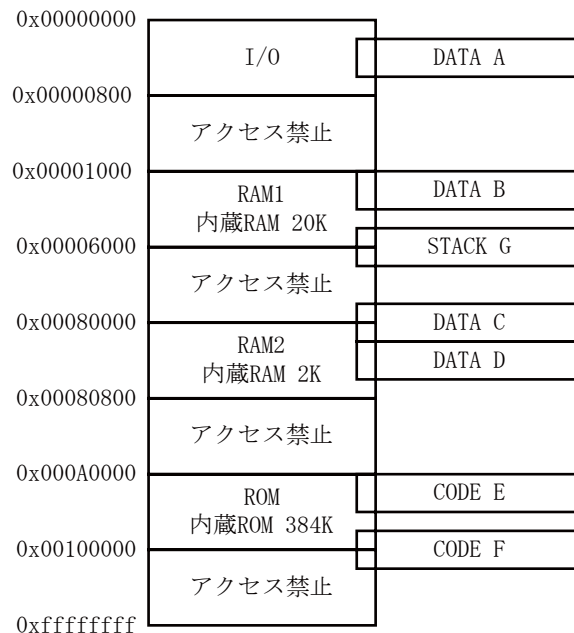
W1372L: RAM 領域または IO 領域外への配置です (DATA_A)

W1371L: RAM 領域外への配置です (STACK_G)

W1372L: RAM 領域または IO 領域外への配置です (DATA_C)

W1370L: ROM 領域外への配置です (CODE_D)

W1370L: ROM 領域外への配置です (CODE_F)



DATA_A は内蔵 IO 領域内に配置されているので警告は出力されません。

DATA_B は -ra で指定した領域内に配置されているので警告は出力されません。

DATA_C は -ra で指定した領域外から配置されているので警告が出力されます。

CODE_D は -ro で指定した領域外に配置されているので警告が出力されます。

CODE_E は -ro で指定した領域内に配置されているので警告は出力されません。

CODE_F は -ro で指定した領域外に配置されているので警告が出力されます。

STACK_G は -ra で指定した領域外に配置されているので警告が出力されます。

< 注意事項 >

- ワーニング出力抑止 (-w 0) が指定されている場合でも本オプションが指定されている場合は警告が出力されます。
- 本オプションは、絶対形式ロードモジュール作成時のみ有効です。
リロケータブルロードモジュール作成時は、本オプションは無視されます。
- 本オプションの使用にあたっては、-ro, -ra オプションを用いた領域指定を行う必要があります。
- リンカ処理対象外である ABS 属性のセクションに対してもチェックを行います。

6.2.42 セクション配置領域チェック抑止指定 (-Xcheck_locate)

セクション配置領域チェックを抑止します。
-check_locate オプションを取り消す際に使用します。

■ セクション配置領域チェック抑止指定 (-Xcheck_locate)

【記述形式】

-Xcheck_locate

【パラメータ】

なし

【説明】

セクション配置領域チェックを抑止します。
-check_locate オプションを取り消す際に用います。

6.2.43 サイズ 0 のセクション配置チェック指定 (-check_size0_sec)

サイズ 0 のセクションに対して配置チェックを行います。

■ サイズ 0 のセクション配置チェック指定 (-check_size0_sec)

【記述形式】

```
-check_size0_sec
```

【パラメータ】

なし

【説明】

サイズが 0 のセクションは、プログラムとして全く意味を持たないため、本来配置チェックは不要です。しかし、サイズが 0 のセクションを含めたすべてのセクションの配置をチェックしたい場合は、-check_size0_sec を指定してください。

-check_size0_sec を指定すると、サイズが 0 のセクションに対して配置が正しいかをチェックします。

チェックする項目は以下のとおりです。

- ROM 領域に書き込み可能なセクションが配置されていないかをチェック
- RAM 領域に初期値のあるセクションを配置していないかをチェック
- ROM 配置セクションが ROM 領域以外に配置されていないかをチェック
- RAM 配置セクションが RAM 領域以外に配置されていないかをチェック
- IO 配置セクションが IO 領域以外に配置されていないかをチェック
- 配置可能なアドレスが存在するかをチェック

-check_size0_sec を指定すると、サイズ 0 のセクションに対しても以下のメッセージが出力されます。

W1301L:ROM 領域に書き込み可能セクションが配置されました (セクション名)

W1303L:RAM エリアに初期値のあるセクションが配置されました (セクション名)

W1370L:ROM 領域外への配置です (セクション名)

W1371L:RAM 領域外への配置です (セクション名)

W1372L:RAM 領域または IO 領域外への配置です (セクション名)

W1373L:IO 領域外への配置です (セクション名)

E4365L: 配置可能なアドレスが領域 " 領域名 " に見つかりません (セクション名)

E4366L: 配置可能なアドレスが見つかりません (セクション名)

【例】

```
flnk911s -cpu mb91101 -check_size0_sec module1.obj module2.obj
```

< 注意事項 >

サイズ 0 セクションの配置チェックはデフォルトでは行われません。

サイズ 0 セクションの配置チェックを行いたい場合は、`-check_size0_sec` オプションを指定してください。

6.2.44 サイズ0のセクション配置チェック抑止指定 (-Xcheck_size0_sec)

サイズ0のセクションの配置チェックを抑止します。

-Xcheck_size0_sec を指定してもサイズが0ではないセクションは、通常どおり配置チェックを行います。

■ サイズ0のセクション配置チェック抑止指定 (-Xcheck_size0_sec)

【記述形式】

-Xcheck_size0_sec

(デフォルト)

【パラメータ】

なし

【説明】

サイズが0のセクションは、プログラムとして全く意味を持たないため、本来配置チェックは不要です。サイズ0のセクションの配置チェックを行いたくない場合は、-Xcheck_size0_sec を指定してください。

-Xcheck_size0_sec を指定してもサイズが0ではないセクションの配置チェックは通常どおり行います。

抑止されるチェック項目は以下のとおりです。

- ROM 領域に書き込み可能なセクションが配置されていないかをチェック
- RAM 領域に初期値のあるセクションを配置していないかをチェック
- ROM 配置セクションが ROM 領域以外に配置されていないかをチェック
- RAM 配置セクションが RAM 領域以外に配置されていないかをチェック
- IO 配置セクションが IO 領域以外に配置されていないかをチェック
- 配置可能なアドレスが存在するかをチェック

-Xcheck_size0_sec を指定することによって、サイズ0のセクションに対して以下のメッセージの出力が抑止されます。

W1301L:ROM 領域に書き込み可能セクションが配置されました (セクション名)

W1303L:RAM エリアに初期値のあるセクションが配置されました (セクション名)

W1370L:ROM 領域外への配置です (セクション名)

W1371L:RAM 領域外への配置です (セクション名)

W1372L:RAM 領域または IO 領域外への配置です (セクション名)

W1373L:IO 領域外への配置です (セクション名)

E4365L: 配置可能なアドレスが領域 " 領域名 " に見つかりません (セクション名)

E4366L: 配置可能なアドレスが見つかりません (セクション名)

【例】

```
flnk911s -cpu mb91101 -Xcheck_size0_sec module1.obj module2.obj
```

6.2.45 プレリンク処理の抑止指定 (-XPLNK)

テンプレート関数処理に用いるプレリンク処理を抑止します。

■ プレリンク処理の抑止指定 (-XPLNK)

【記述形式】

-XPLNK

【パラメータ】

なし

【説明】

C++ プログラムのテンプレート関数によるコードサイズの増加を抑止するため、リンクは、リンク処理を行う前にプレリンクを起動しプレリンク処理を行います。

本オプションはプレリンク処理を抑止します。

【例】

```
flnk911s a1 a2 a3 -o a123.abs -m a123.abs -XPLNK
```

6.2.46 相対アセンブルリスト入力ディレクトリ指定 (-alin)

相対アセンブルリストファイルが格納されているディレクトリを指定します。
このオプションがない場合は、オブジェクトモジュールと同じ場所となります。

■ 相対アセンブルリスト入力ディレクトリ指定 (-alin)

【記述形式】

-alin <パス名>

【パラメータ】

<パス名>

相対アセンブルリストファイルが格納されているディレクトリ

【説明】

絶対形式アセンブルリストファイルを出力するときに、使用するオプションです。

相対アセンブルリストファイルが格納されているディレクトリを指定します。

本オプションの指定がない場合は、オブジェクトモジュールと同じ場所になります。

-alf オプションで、相対アセンブルリストファイルをパス名付きで指定している場合には、-alf オプション指定のパスを優先します。

【例】

```
flnk911s *.obj -o ap.abs -alin d:\fr20 -alf swctr1.lst,mstdef.lst
flnk911s *.obj -o ap.abs -alsf d:\fr20\swctr1.lst,d:\fr20\mstdef.lst
```

上記の 2 つの例は同じ意味です。

6.2.47 絶対形式アセンブルリスト出力ディレクトリ指定 (-alout)

絶対形式アセンブルリストファイルを出力するディレクトリを指定します。

■ 絶対形式アセンブルリスト出力ディレクトリ指定 (-alout)

【記述形式】

<code>-alout <パス名></code>

【パラメータ】

<パス名>

絶対形式アセンブルリストファイルを出力するディレクトリ

【説明】

絶対形式アセンブルリストファイルを出力するディレクトリを指定します。

本オプションの指定がない場合は、カレントディレクトリとなります。

【例】

```
flnk911s *.obj -o ap.abs -alin d:\fr20 -alf swctrl.lst,mstdef.lst  
-alout d:\fr20\als
```


6.2.48 絶対形式アセンブルリスト出力指定 (-als)

絶対形式アセンブルリストファイルの出力を指定します。
すべてのオブジェクトモジュールに対する出力指示です。

■ 絶対形式アセンブルリスト出力指定 (-als)

【記述形式】

```
-als
```

【パラメータ】

なし

【説明】

すべてのモジュールに対して絶対形式アセンブルリストを作成することを指示します。

本オプションの指定がないときは、絶対形式アセンブルリストの作成を行いません。

前に指定した、-alsf, -Xals を無効にします。

【例】

```
flnk911s *.obj -o ap.abs -als
```

6.2.49 絶対形式アセンブルリスト出力モジュール指定 (-alsf)

絶対形式アセンブルリストファイルを出力するモジュールを指定します。
オブジェクトモジュールに対する選択出力指示を行います。

■ 絶対形式アセンブルリスト出力モジュール指定 (-alsf)

【記述形式】

<code>-alsf <相対アセンブルリストファイル名> [, ...]</code>

【パラメータ】

<相対アセンブルリストファイル名>

絶対形式アセンブルリスト作成の元になる相対アセンブルリストファイル名
ファイル名の指定にはワイルドカードが使えます。

【説明】

絶対形式アセンブルリストを作成するモジュールを選択します。

モジュールの指定は、相対アセンブルリストファイル名で行います。拡張子を省略した場合には、".lst" を仮定します。

指定されなかったモジュールは、絶対形式アセンブルリストの作成を行いません。

本オプションは、複数に分けて指定できます。

前に指定した、-als、-Xals を無効にします。

-alin オプションより相対アセンブルリストの格納パスを指定できますので、本オプション指定時にはパス指定を省略することが可能です。

【例】

```
flnk911s *.obj -o ap.abs -alsf swctrl.lst,mstdef.lst
flnk911s *.obj -o ap.abs -alsf swctrl.lst -alsf mstdef.lst
flnk911s *.obj -o ap.abs -alsf swctrl -alsf mstdef
```

上記の 3 つの例は同じ意味です。

6.2.50 絶対形式アセンブルリスト出力抑止指定 (-Xals)

すべてのモジュールに対して絶対形式アセンブルリストを、作成しないことを指示します。

■ 絶対形式アセンブルリスト出力抑止指定 (-Xals)

【記述形式】

-Xals	(デフォルト)
-------	---------

【パラメータ】

なし

【説明】

すべてのモジュールに対して絶対形式アセンブルリストを作成しないことを指示します。

本オプションはデフォルトです。

前に指定した、-als, -alsf を無効にするときに使用します。

【例】

```
flnk911s *.obj -o ap.abs -alf sectrl.lst,mstdef.lst -nl
```

6.2.51 ROM/RAM, ARRAY リスト出力指定 (-alr)

ROM/RAM, ARRAY リストを出力指定します。
コンパイル, アセンブル時にデバッグ情報が出力されていることが必要です。

■ ROM/RAM, ARRAY リスト出力指定 (-alr)

【記述形式】

-alr

【パラメータ】

なし

【説明】

すべての絶対形式アセンブルリストに ROM/RAM, ARRAY リストを付加します。

本オプションを使用する場合, -als オプションは省略できます。

前に指定した, -alrf, -Xalr を無効にします。

ROM/RAM, ARRAY リストを出力するためには, コンパイル, アセンブル, リンク時に
デバッグ情報出力オプション (-g) を指定してください。

【例】

```
flnk911s *.obj -o ap.abs -als -alr -g
```

```
flnk911s *.obj -o ap.abs -alr -g
```

上記の 2 つの例は同じ意味です。

6.2.52 ROM/RAM, ARRAY リスト出力モジュール指定 (-alrf)

ROM/RAM, ARRAY リストを出力するモジュールを指定します。
コンパイル, アセンブル時にデバッグ情報が出力されていることが必要です。

■ ROM/RAM, ARRAY リスト出力モジュール指定 (-alrf)

【記述形式】

```
-alrf <相対アセンブルリストファイル名> [, ... ]
```

【パラメータ】

<相対アセンブルリストファイル名>

ROM/RAM, ARRAY リストを出力するモジュールを相対アセンブルリストファイル名で指定します。

ファイル名の指定にはワイルドカードが使えます。

【説明】

絶対形式アセンブルリストに ROM/RAM, ARRAY リストを付加して出力したいモジュールを選択します。

モジュールの指定は, 相対アセンブルリストファイル名で行います。拡張子を省略した場合には, ".lst" を仮定します。

指定されなかったモジュールは, ROM/RAM, ARRAY リストが作成されません。

本オプションを使用する場合, -alsf オプションは省略できます。

本オプションは, 複数に分けて指定できます。

前に指定した, -alr, -Xalr を無効にします。

【例】

```
flnk911s *.obj -o ap.abs -alsf swctrl.lst,mstdef.lst
                                -alrf swctrl.lst,mstdef.lst

flnk911s *.obj -o ap.abs -alsf swctrl.lst
                                -alrf swctrl.lst,mstdef.lst

flnk911s *.obj -o ap.abs -alrf swctrl -alrf mstdef
```

上記の3つの例は同じ意味です。

6.2.53 ROM/RAM, ARRAY リスト出力抑止指定 (-Xalr)

すべての絶対形式アセンブルリストに ROM/RAM, ARRAY リストを付加しないことを指定します。

■ ROM/RAM, ARRAY リスト出力抑止指定 (-Xalr)

【記述形式】

-Xalr	(デフォルト)
-------	---------

【パラメータ】

なし

【説明】

すべての絶対形式アセンブルリストに ROM/RAM, ARRAY リストを付加しないことを指定します。

本オプションは、デフォルトですので特に指定する必要はありません。

前に指定した、-alr, -alrf を無効にします。

【例】

```
flnk911s *.obj -o ap.abs -als -alr -Xalr
```

```
flnk911s *.obj -o ap.abs -als -Xalr
```

上記の 2 つの例は同じ意味です。

6.2.54 ROM/RAM, ARRAY リストのシンボルとアドレスの表示位置指定 (-na, -an)

ROM/RAM, ARRAY リストのシンボルとアドレスの表示位置の指定を行います。
 -na が指定された場合, シンボル (NAME), アドレス (ADDRESS) の順に出力します。
 -an が指定された場合, アドレス (ADDRESS), シンボル (NAME) の順に出力します。

■ ROM/RAM, ARRAY リストのシンボルとアドレスの表示位置指定 (-na, -an)

【記述形式】

-na	(デフォルト)
-----	---------

【パラメータ】

なし

【説明】

ROM/RAM, ARRAY リストのシンボルとアドレスを NAME, ADDRESS の順に出力します。

シンボルはアルファベット順に出力します。

ROM/RAM, ARRAY リストが出力指定されたモジュールに対してのみ有効です。

このオプションの機能はデフォルトですので, -an オプションを取り消すときに指定します。

【例】

```
flnk911s *.obj -o ap.abs -alr -na
```

```
flnk911s *.obj -o ap.abs -alr
```

上記の 2 つの例は同じ意味です。

【記述形式】

-an

【パラメータ】

なし

【説明】

ROM/RAM, ARRAY リストのシンボルとアドレスを ADDRESS, NAME の順に出力します。

シンボルはアドレス順に出力します。

-na オプション (デフォルト) を変更するときに使用します。

ROM/RAM, ARRAY リストが出力指定されたモジュールに対してのみ有効です。

第6章 リンカのオプション

【例】

```
flnk911s *.obj -o ap.abs -alr -na -an
```

```
flnk911s *.obj -o ap.abs -alr -an
```

上記の2つの例は同じ意味です。

6.2.55 外部シンボル相互参照情報リスト出力指定 (-xl)

外部シンボル相互参照情報リストファイルを出力指定します。

■ 外部シンボル相互参照情報リスト出力指定 (-xl)

【記述形式】

-xl

【パラメータ】

なし

【説明】

外部シンボル相互参照情報リストファイルを作成することを指示します。

本オプションの指定がないときは、外部シンボル相互参照情報リストファイルの作成を行いません。

【例】

```
flnk911s *.obj -o ap.abs -xl
```

6.2.56 外部シンボル相互参照情報リストファイル名の指定 (-xlf)

外部シンボル相互参照情報リストファイルの出力先ディレクトリやファイル名を変更したいときに使用します。

■ 外部シンボル相互参照情報リストファイル名の指定 (-xlf)

【記述形式】

<code>-xlf</code> <出力ファイル名>

【パラメータ】

<出力ファイル名>

出力ファイル名を指定します。出力先のディレクトリを変更するときは前にパス名を付けます。

【説明】

指定した名前で外部シンボル相互参照情報リストファイルを作成します。

本オプションを使用する場合、`-xl` オプションは省略できます。

<出力ファイル名> 指定で拡張子を省略した場合は、デフォルト拡張子 `".mpx"` が付加されます。

本オプションの指定がない場合は、絶対形式ロードモジュールファイル名の拡張子を `".mpx"` に直したものを、出力ファイル名とします。

【例】

```
flnk911s *.obj -o ap.abs -xl -xlf ccp903.mpx
```

```
flnk911s *.obj -o ap.abs -xlf ccp903
```

上記の 2 つの例は同じ意味です。

6.2.57 外部シンボル相互参照情報リスト出力抑止指定 (-Xxl)

外部シンボル相互参照情報リストファイルを出力抑止指定します。

■ 外部シンボル相互参照情報リスト出力抑止指定 (-Xxl)

【記述形式】

-Xxl

(デフォルト)

【パラメータ】

なし

【説明】

外部シンボル相互参照情報リストファイルの出力抑止を指示します。

本オプションは、デフォルトですので特に指定する必要はありません。

前に指定した、-xl, -xlf を無効にします。

【例】

```
flnk911s *.obj -o ap.abs -xl -Xxl
```

```
flnk911s *.obj -o ap.abs -Xxl
```

```
flnk911s *.obj -o ap.abs
```

上記の 3 つの例は同じ意味です。

6.2.58 ローカルシンボル情報リスト出力指定 (-sl)

ローカルシンボル情報リストファイルを出力指定します。この場合、コンパイル、アセンブル、リンク時にデバッグ情報が出力されていることが必要です。

■ ローカルシンボル情報リスト出力指定 (-sl)

【記述形式】

-sl

【パラメータ】

なし

【説明】

ローカルシンボル情報リストファイルを作成することを指示します。

本オプションの指定がないときは、ローカルシンボル情報リストファイルの作成を行いません。

ローカルシンボル情報リストファイルを出力するためには、コンパイル、アセンブル、リンク時にデバッグ情報出力オプション (-g) を指定してください。

【例】

```
flnk911s *.obj -o ap.abs -sl -g
```

6.2.59 ローカルシンボル情報リストファイル名の指定 (-slf)

ローカルシンボル情報リストファイルの出力先ディレクトリやファイル名を変更したいときに使用します。

■ ローカルシンボル情報リストファイル名の指定 (-slf)

【記述形式】

-slf <出力ファイル名>

【パラメータ】

<出力ファイル名>

出力ファイル名を指定します。出力先のディレクトリを変更するときは前にパス名を付けます。

【説明】

指定した名前でローカルシンボル情報リストファイルを作成します。

本オプションを使用する場合、-sl オプションは省略できます。

<出力ファイル名> 指定で拡張子を省略した場合は、デフォルト拡張子 ".mps" が付加されます。

本オプションの指定がない場合は、絶対形式ロードモジュールファイル名の拡張子を ".mps" に直したものを、出力ファイル名とします。

【例】

```
flnk911s *.obj -o ap.abs -sl -slf ccp903.mps -g
```

```
flnk911s *.obj -o ap.abs -slf ccp903 -g
```

上記の 2 つの例は同じ意味です。

6.2.60 ローカルシンボル情報リスト出力抑止指定 (-Xsl)

ローカルシンボル情報リストファイルを出力抑止指定します。

■ ローカルシンボル情報リスト出力抑止指定 (-Xsl)

【記述形式】

-Xsl	(デフォルト)
------	-----------

【パラメータ】

なし

【説明】

ローカルシンボル情報リストファイルの出力抑止を指示します。

本オプションは、デフォルトですので特に指定する必要はありません。

前に指定した、-sl, -slf を無効にします。

【例】

```
flnk911s *.obj -o ap.abs -sl -Xsl
```

```
flnk911s *.obj -o ap.abs -Xsl
```

```
flnk911s *.obj -o ap.abs
```

上記の 3 つの例は同じ意味です。

6.2.61 セクション詳細マップリスト出力指定 (-ml)

セクション詳細マップリストファイルを出力指定します。

■ セクション詳細マップリスト出力指定 (-ml)

【記述形式】

-ml

【パラメータ】

なし

【説明】

セクション詳細マップリストファイルを作成することを指示します。

本オプションの指定がないときは、セクション詳細マップリストファイルの作成を行いません。

【例】

```
flnk911s *.obj -o ap.abs -ml
```

6.2.62 セクション詳細マップリストファイル名の指定 (-mlf)

セクション詳細マップリストファイルの出力先ディレクトリやファイル名を変更したいときに使用します。

■ セクション詳細マップリストファイル名の指定 (-mlf)

【記述形式】

<code>-mlf</code> <出力ファイル名>

【パラメータ】

<出力ファイル名>

出力ファイル名を指定します。出力先のディレクトリを変更するときは前にパス名を付けます。

【説明】

指定した名前でセクション詳細マップリストファイルを作成します。

本オプションを使用する場合、`-ml` オプションは省略できます。

<出力ファイル名> 指定で拡張子を省略した場合は、デフォルト拡張子 `".mpm"` が付加されます。

本オプションの指定がない場合は、絶対形式ロードモジュールファイル名の拡張子を `".mpm"` に直したものを、出力ファイル名とします。

【例】

```
flnk911s *.obj -o ap.abs -ml -mlf ccp903.mpm
```

```
flnk911s *.obj -o ap.abs -mlf ccp903
```

上記の2つの例は同じ意味です。

6.2.63 セクション詳細マップリスト出力抑止指定 (-Xml)

セクション詳細マップリストファイルを出力抑止指定します。

■ セクション詳細マップリスト出力抑止指定 (-Xml)

【記述形式】

-Xml

(デフォルト)

【パラメータ】

なし

【説明】

セクション詳細マップリストファイルの出力をすることを抑止指示します。

本オプションは、デフォルトですので特に指定する必要はありません。

前に指定した、-ml, -mlf を無効にします。

【例】

```
flnk911s *.obj -o ap.abs -ml -Xml
```

```
flnk911s *.obj -o ap.abs -Xml
```

```
flnk911s *.obj -o ap.abs
```

上記の 3 つの例は同じ意味です。

第7章

リンクカの実出力リストファイル

この章では、リンクカの実出力する各リストファイルのフォーマット、情報の見方について説明します。

- 7.1 リンカの実出力するリストファイルの種類
- 7.2 リンクリストファイル
- 7.3 絶対形式アセンブルリストファイル
- 7.4 外部シンボル相互参照情報リストファイル
- 7.5 ローカルシンボル情報リストファイル
- 7.6 セクション配置詳細情報リストファイル

7.1 リンカの実行するリストファイルの種類

リンカが実行するリストファイルは以下の 5 つです。

- リンクリストファイル
- 絶対形式アセンブルリスト
- 外部シンボル相互参照情報リスト
- ローカルシンボル情報リスト
- セクション詳細マップリスト

これらのファイルは、リンカ起動時のオプションで、実行するかどうか選択できます。

■ リンクリストファイル

リンクリストファイルは、リンカ起動時のオプションや入力モジュール名、モジュールリンク後のセクションとメモリ使用情報、外部シンボルの情報が出力されます。

■ 絶対形式アセンブルリスト

絶対形式アセンブルリストは、モジュールリンク後の情報をもとに、アセンブラが出力した相対形式のアセンブルリストを絶対形式で表示したリストです。

アセンブラ言語レベルでのデバッグを行う場合に参照でき、リンクリストではわからない機械語の 1 ステップ単位のアドレスを知ることができます。

■ 外部シンボル相互参照情報リスト

外部シンボル相互参照情報リストは、リンク後の各モジュールの外部定義シンボルと外部参照シンボルのモジュール間相互参照の情報が出力されます。

■ ローカルシンボル情報リスト

ローカルシンボル情報リストは、リンク後のモジュールごとのローカルシンボルを含んだ変数や関数などの情報を出力します。

■ セクション詳細マップリスト

セクション詳細マップリストは、リンク後のモジュールごとのセクション配置情報を出力します。

7.2 リンクリストファイル

リンクリストファイルは、情報内容により4つの部分に分かれています。

- コントロールリスト
- マップリスト
- メモリ使用情報リスト
- 外部シンボルリスト

ここでは、それぞれに出力される項目を説明します。

■ リンクリストファイルの構成

リンカの実行リストファイルは、4つの部分に分かれています。

- コントロールリスト
 - 指定されたオプション
 - 入力オプション
 - エラーメッセージ
- マップリスト
 - セクション名
 - セクション属性
 - リンク後のセクション配置アドレス
- メモリ使用情報リスト
 - ROM/RAM 使用情報
 - 領域内情報
 - 総合評価値情報
- 外部シンボルリスト
 - 外部シンボル名
 - 定義、参照の種別
 - シンボルの値

各リストの境界には、改ページの制御文字が出力されます。

7.2.1 コントロールリスト

コントロールリストには、リンクを実行したときに指定したオプションと、入力モジュール名を表示します。また、リンク処理中に発見したエラーも表示します。

■ コントロールリスト部のリスト出力フォーマット

コントロールリスト部のリスト出力フォーマットを図 7.2-1 に示します。

図 7.2-1 コントロールリスト部のリスト出力フォーマット

FR/FR80 Family SOFTUNE Linker	Control List	YYYY-MM-DD hh:mm:ss	Page:	1
Option File(s)				
① オプションファイル名表示エリア				
Control(s)				
② リンカ制御オプション表示エリア				
Input Module(s)				
③ 入力モジュール表示エリア				
Error(s)				
④ エラーメッセージ表示エリア				

● ページヘッダ

先頭行にリンク名、リスト名、日時、ページ番号を表示します。

オプションファイル名表示エリア

オプションファイルが使用された場合に、そのファイル名を表示します。

使用されなかった場合は、「** no use **」を表示します。

リンク制御オプション表示エリア

指定されたオプションおよびデフォルトで有効としたオプションを表示します。

オプションファイル内で指定されたオプションの場合、オプションの前に@を付けます。

入力モジュール表示エリア

1 から始まる通し番号を付けて、ファイル名とモジュール名を表示します。

エラーメッセージ表示エリア

処理中に検出したエラーメッセージを表示します。

エラーがない場合は、「** Nothing **」を表示します。

■ コントロールリスト部のリスト表示例

コントロールリスト部のリスト表示例を図 7.2-2 に示します。

図 7.2-2 コントロールリスト部のリスト表示例

```
FR/FR80 Family SOFTUNE Linker Control List    2003-08-26 15:18:11 Page: 1

Option File(s)

    ** no use **

Control(s)

    -a
    -g
    -l usrlb.lib
    -ro prog=0x8000/0xffff
    -ra data=0x0000/0x03ff
    -sc P+code=prog,D+data+S=data

Input Module(s)

    1 pca02.obj(pca01)
    2 pcasb.obj(pcasb)
    3 xccdef.obj(xccdef)

Error(s)

    ** Nothing **
```

7.2.2 マップリスト

マップリストには、セクションの名前、内容種別、属性と結合後のセクション配置アドレスを表示します。

■ マップリスト部のリスト出力フォーマット

マップリスト部のリスト出力フォーマットを図 7.2-3 に示します。

図 7.2-3 マップリスト部のリスト出力フォーマット

FR/FR80 Family SOFTUNE Linker Mapping List					
YYYY-MM-DD hh:mm:ss Page:					2
S_Addr.	-E_Addr.	Size	Section Type Al	Sec. (Top **)	C
① マップ情報表示エリア					

● ページヘッダ

先頭行にリンカ名、リスト名、日時、ページ番号を表示します。

マップ情報表示エリア

マップ情報は、開始アドレス順、ただし、開始アドレスが同じ場合はセクション出現順に表示されます。

S-Addr : セクションの開始アドレス (16 進)
 E-Addr : セクションの終了アドレス (16 進)
 Size : セクションのサイズ (16 進)
 Section : セクション内容種別

セクションの内容種別を表示します。

CODE プログラムセクション
 DATA データセクション
 CONST 初期値付データセクション
 STACK スタックセクション
 IO IO セクション

セクション種別の後に、結合属性を表示します。

P 単純連結結合
 C 共有結合
 N 結合なし

type : セクションの属性

左から順に、以下の属性を表示します。

R/- Read 可 / 不可
 W/- Write 可 / 不可

X/- 実行可 / 不可

I/- 初期値あり / なし

Al : セクション配置時の境界調整数 (16 進)

境界調整数が 0x100 以上の場合は, "***" と表示されます。

Sec.(Top **) : セクション名

ROM RAM 転送セクションで生成されたセクションには先頭に "#" が表示されます。

*** は, 指定されたページ幅でセクション名が何桁表示できるかを示します。

(注意事項) セクションサイズが "0" のセクションの終了アドレスは, 「.....」で表示します。

■ マップリスト部のリスト表示例

マップリスト部のリスト表示例を図 7.2-4 に示します。

図 7.2-4 マップリスト部のリスト表示例

```
FR/FR80 Family SOFTUNE Linker Mapping List 2003-08-09 20:41:12 Page: 2

S_Addr. -E_Addr.  Size      Section Type Al      Sec.(Top 29)      C
00000000-0000001F 00000020 DATA P RW-- 08 REL D
00000020-00000039 0000001A DATA P RW-- 02 REL data
0000003A-00000053 0000001A STAK P RW-- 02 REL S
00000054-0000006D 0000001A DATA P RW-I 02 REL init
00008000-00008039 0000003A CODE P R-XI 02 REL P
0000803A-00008053 0000001A CODE P R-XI 02 REL code
00008054-0000806D 0000001A DATA P R--I 02 REL #init
```

7.2.3 メモリ使用情報リスト

メモリ使用情報リストには、RAM 指定領域と ROM 指定領域の、領域名、空きまたはオーバ領域サイズ、指定領域の先頭・終端アドレスなどを表示します。

■ メモリ使用情報リスト部のリスト出力フォーマット

メモリ使用情報リスト部のリスト出力フォーマットを図 7.2-5 に示します。

図 7.2-5 メモリ使用情報リスト部のリスト出力フォーマット

FR/FR80 Family SOFTUNE Linker ROM/RAM Used Info YYYY-MM-DD hh:mm:ss						Page:	3
S_Addr.	-E_Addr.	Size	Remain	Name/State	C		
① ROM/RAM使用情報							
.....							
② 領域内情報							
.....							
③ 総合評価値情報							
.....							

● ページヘッダ

先頭行にリンカ名、リスト名、日時、ページ番号を表示します。

ROM/RAM 使用情報表示エリア

- ROM/RAM 使用情報表示エリア内で、出力された行の先頭に "#" の文字が付いた行は、-ro, -ra オプションにより指定された情報を表示しています。
- 出力された行の先頭に "#" の文字が付いていない行は、有効エリア内のセクション群の先頭アドレス、終端アドレス、ギャップ領域を含む使用領域サイズと使用領域サイズに対する過不足値を表示しています。
- -ro, -ra オプションにより指定された領域に、配置できなかったセクションのセクション名とセクションサイズの表示も行います。

S_Addr : 開始アドレス (16 進)

E_Addr : 終了アドレス (16 進)

Size : 領域サイズ (16 進)

Remain : メモリ内の状況領域サイズ (16 進)

先頭の記号は以下のことを示します。

+ : 空き領域サイズ

- : オーバ領域サイズ

スペース : 空き / オーバ領域がゼロの場合

Name/State : メモリ領域名、セクション名

領域内情報エリア

領域内情報エリアは、有効エリア内の空き領域、使用済領域、ギャップ領域をマップイメージで表示します。

この情報エリアに表示される情報は、指定領域に対して配置されたセクションのみの情報です。

S-Addr	:	領域の開始アドレス	(16 進)
E-Addr	:	領域の終了アドレス	(16 進)
Size	:	領域サイズ	(16 進)
Name/State	:	領域タイプ	
		FREE	: 空き領域
		USED	: 使用済領域
		GAP	: ギャップ領域

総合評価値情報

総合評価値情報は、ROM エリアの総合と RAM エリアの総合の 2 つに対して、以下の 3 つの情報を表示します。

- 指定領域の合計値 (Total)
- 使用した領域の合計値 (Used) (注意事項) ギャップ領域を含む
- 空きまたはオーバ領域の合計値 (Remainder)

■ メモリ使用情報リスト部のリスト表示例

メモリ使用情報リスト部のリスト表示例を図 7.2-6 に示します。

図 7.2-6 メモリ使用情報リスト部のリスト表示例

FR/FR80 Family SOFTUNE Linker ROM/RAM Used Info List 2003-09-02 20:45:25 Page: 3						
S-Addr.	- E-Addr.	Size	Remain	Name/State	C	
<div># 00001000 - 00001FFF 00001000 ----- RAM1</div> <div>00001000 - 00001774 00000775 +0000088B</div> <div>00001000 - 00001136 00000137 ----- USED</div> <div>00001137 - 00001137 00000001 ----- GAP</div> <div>00001138 - 0000152B 000003F4 ----- USED</div> <div>0000152C - 0000152F 00000004 ----- GAP</div> <div>00001530 - 00001774 00000245 ----- USED</div> <div>00001775 - 00001FFF 0000088B ----- FREE</div> <div>←オプションで指定したメモリ領域</div> <div>←配置したセクションに対するメモリ状況</div> <div>RAM1領域の情報</div>						
<div># 00002000 - 000023FF 00000400 ----- RAM2</div> <div>00002000 - 00003A1E 00001A1F -0000161F</div> <div>** Not Locate ** 00000B28 ----- data01</div> <div>** Not Locate ** 00000EF6 ----- data02</div> <div>00002000 - 000023FF 00000400 ----- FREE</div> <div>←*1</div> <div>←配置できなかったセクション情報 *2</div> <div>RAM2領域の情報</div>						
RAM -- Total(00001400) Used(00002194) Remainder(-00000D94) ←全RAM領域の情報						
<div># 000BC000 - 000BCFFF 00001000 ----- ROM1</div> <div>000BBF00 - 000BDFFF 00002100 -00001100</div> <div>000BC000 - 000BCFFF 00001000 ----- USED</div> <div>←*3</div> <div>←ROM1領域の情報</div>						
<div># 000BD000 - 000BFFFF 00003000 ----- ROM2</div> <div>000BD000 - 00000000 +00003000</div> <div>000BD000 - 000BFFFF 00003000 ----- FREE</div> <div>ROM2領域の情報</div>						
ROM -- Total(00004000) Used(00002100) Remainder(+00001F00) ←全ROM領域の情報						

説明

- *1:自動配置オプションによりメモリ領域にセクションを配置しようとして配置できなかったセクションがあるメモリ領域の情報は、そのセクションのサイズを加えた状態を示しています。(数値表現として 0xFFFFFFFF を超えた場合、その数値の下位 32bit 分が表示されます。)
- *2:自動配置オプションのモード 2 が指定された場合の配置できなかったセクションは、(ROM 領域、RAM 領域のどちらかの)最後に指定されたメモリ領域に対して表示を行います。
- *3:ユーザ配置指定が行われたセクションは、そのセクションの先頭アドレスが含まれるメモリ領域に含みます。

7.2.4 シンボルリスト

シンボルリストには、外部シンボル名、定義、参照の種別、シンボルの値を表示します。

■ シンボルリスト部のリスト出力フォーマット

シンボルリスト部のリスト出力フォーマットを図 7.2-7 に示します。

図 7.2-7 シンボルリスト部のリスト出力フォーマット

FR/FR80 Family SOFTUNE Linker				Symbol List	YYYY-MM-DD hh:mm:ss	Page:	1
Symbol Value	Type	Def.	Symbol Name(Top **)				C
① シンボルリストエリア							

● ページヘッダ

先頭行にリンカ名、リスト名、日時、ページ番号を表示します。

シンボルリスト表示エリア

Symbol Value : シンボルアドレス or シンボル値 (16 進)

Type : シンボル種別

次のいずれかが表示されます。

Addr. : シンボルはアドレスレベルです。

EQU : シンボルは EQU 定義シンボルです。

bit : シンボルはビット属性です。

Def. : シンボルの定義

次のいずれかが表示されます。

OM/LM : 入力オブジェクトモジュールまたは相対形式ロードモジュール中で定義されています。

LIB : 結合されたライブラリ中で定義されています。

user : -df オプションにより値を仮設定されたシンボルです。

Symbol Name : シンボル名

"**" は、指定されたページ幅でシンボル名が何桁表示できるかを示します。

(注意事項) シンボルが、参照されている場合は、そのままシンボル名を表示し、参照されていない場合は、シンボル名の先頭に "@" を付けて表示します。

また、C/C++ コンパイラで生成されたシンボルの場合、マングル名とシンボル名が表示されます。

■ シンボルリスト部のリスト表示例

シンボルリスト部のリスト表示例を図 7.2-8 に示します。

図 7.2-8 シンボルリスト部のリスト表示例

FR/FR80 Family SOFTUNE Linker Symbol List				2003-08-15 15:26:32	Page:	3
Symbol Value	Type	Def.	Symbol Name (Top **)			C
000000CA (ABS)	Addr.	OM/LM	____ct__8USDollarFUiT1			
			USDollar::USDollar(unsigne dint,unsigned int)			
00001234 (ABS)	Addr.	user	____nw__FUi			
			operator new(unsigned int)			
00000000 (ABS)	Addr.	OM/LM	____pl__FR8USDollarT1			
			operator +(USDollar &,USDollar &)			
0000004A (ABS)	Addr.	OM/LM	_main			
			main			

7.3 絶対形式アセンブルリストファイル

リンカが出力する絶対形式アセンブルリストは、以下により構成されています。

- ヘッダ
- インフォメーションリスト
- ROM/RAM, ARRAY リスト
- アセンブルソースリスト
- セクション情報リスト
- クロスリファレンスリスト

ここでは、それぞれに出力される項目を説明します。

■ 絶対形式アセンブルリストの形式

● ヘッダ

各ページの先頭に出力されます。

● インフォメーションリスト

アセンブラが出力したインフォメーションリストをそのまま出力します。

● ROM/RAM, ARRAY リスト

- ROM/RAM リスト

ROM/RAM 領域に配置されているグローバルシンボルの名前、絶対アドレス情報などが出力されます。

- ARRAY リスト

配列要素の名前、構造体のメンバ名、絶対アドレス情報などが出力されます。

オプション `-alr` を指定すると、ROM/RAM, ARRAY リストを表示します。`-Xalr` を指定すると ROM/RAM, ARRAY リストは表示されません。

● アセンブルソースリスト

アセンブルソースリストは、ソースプログラムをアセンブルしたときの様々な情報を行単位に表示したものです。エラー情報、ロケーション、オブジェクトコードなどが表示されます。

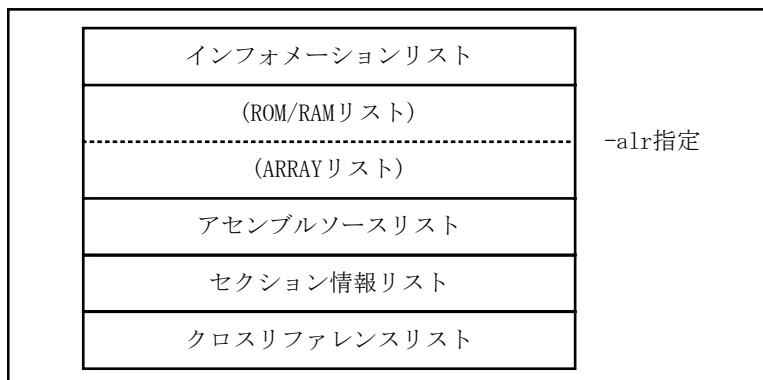
● セクション情報リスト

ソースプログラム内で定義されたセクションの名前、属性などが表示されます。

● クロスリファレンスリスト

ソースプログラム内で使用しているシンボル名の定義、参照関係を行番号で示します。
リストの構成は、図 7.3-1 のようになっています。

図 7.3-1 アセンブルリストの構成



■ アセンブルリスト内のエラーメッセージ

アセンブル時に発生したエラーが、もし、アセンブルリスト上にあれば、エラーメッセージがそのまま表示されます。

7.3.1 ヘッダ, インフォメーションリスト

ヘッダは4行で構成されています。

また, リストの1ページ目には, ヘッダに続いてインフォメーションリストが表示されます。

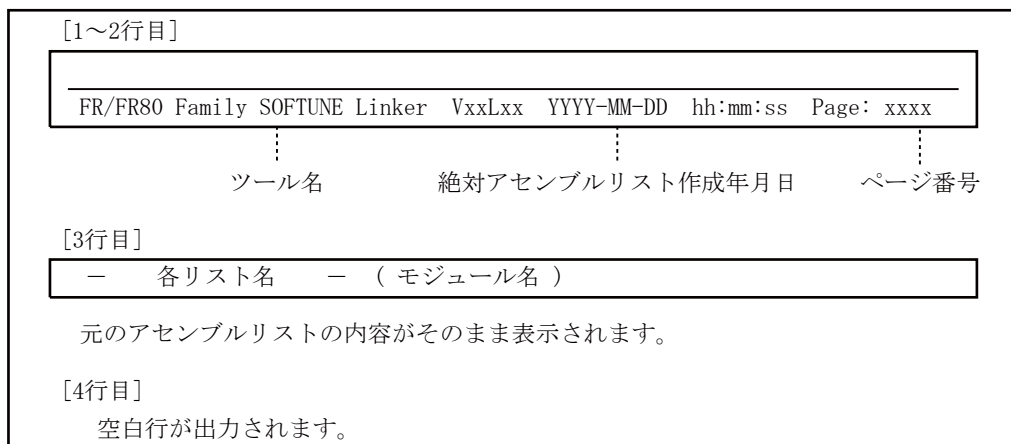
インフォメーションリストには次の情報が表示されます。

- アセンブル時のオプション指定内容
- エラー数とワーニング数
- ソースファイル名
- インクルードファイル名
- オプションファイル名
- 他

■ ヘッダ形式

ヘッダは4行より構成されています。各ページの先頭に表示されます。1～2行目はアセンブルリスト全体を通じて共通の形式をもち, 3行目は各ソースプログラムで異なります。ヘッダの形式を図7.3-2に示します。

図 7.3-2 ヘッダの形式



7.3.2 ROM/RAM, ARRAY リスト

ROM/RAM, ARRAY リストは、C/C++ ソースプログラム内で記述されるシンボルについての情報を示すものです。

ROM/RAM, ARRAY リストを出力するには、コンパイル、アセンブル時にデバッグ情報が出力指定されていることが必要です。

■ ROM/RAM, ARRAY リスト

ROM/RAM, ARRAY リストの形式を図 7.3-3 に示します。

図 7.3-3 ROM/RAM, ARRAY リストの形式

・ ROM/RAM リストの形式				
- ROM/RAM LISTING - (モジュール名)				
NAME	ADDRESS	VALUE	KIND	MEMORY
--- 名前 ---	xxxxxxx	xxxxxxx	x	xxx
*1	*2	*3	*4	*5
・ ARRAY リストの形式				
- ARRAY LISTING - (モジュール名)				
NAME	ADDRESS	VALUE		
--- 名前 ---	xxxxxxx	xxxxxxx		
*1	*2	*3		

*1 : シンボル名

プログラム内で記述されているシンボルの名前を表示します。

ARRAY リストの場合は配列、構造体の要素すべてを表示します。

*2 : アドレス

絶対アドレスを 16 進で表示します。

シンボル名とアドレスの表示位置はオプションで変更できます。-na を指定するとシンボル名、アドレスの順で、-an を指定するとアドレス、シンボル名の順に表示します。

詳細は、「6.2.54 ROM/RAM, ARRAY リストのシンボルとアドレスの表示位置指定 (-na, -an)」を参照してください。

*3 : 値

シンボルに初期値が代入されている場合はその値を 10 進で表示します。

*4: シンボルの種別

以下があります。

L: 変数値

F: 関数名

T: タグ名参照

*5: メモリ配置

シンボルが配置されているメモリ領域を表示します。

ROM: ROM 領域

RAM: RAM 領域

【例 1】

図 7.3-4 ROM/RAM リスト例

- ROM/RAM LISTING - (sample1)				
NAME	ADDRESS	VALUE	KIND	MEMORY
_Line	000002B8	00000100	L	ROM
_Nameleng	0000004C	00000016	L	RAM
_symref	00005592		T	ROM
_Tflg	00000763	00000002	L	RAM

【例 2】

図 7.3-5 ARRAY リスト例

- ARRAY LISTING - (sample1)		
NAME	ADDRESS	VALUE
_symref[0].val	00005592	00000100
_symref[0].nam	00005596	00000002
_symref[0].atr	00005598	00000016
_symref[1].val	0000559A	00012000
_symref[1].nam	0000559E	00000002
_symref[1].atr	000055A0	00000016
_Xpcr[0][0]	00001066	00000400
_Xpcr[0][1]	000010A0	00000120

7.3.3 アセンブルソースリスト

アセンブルソースリストはロケーション部分が絶対アドレス、オブジェクトコード部分がリンク後の確定したコードで表示されます。

■ アセンブルソースリストの形式

アセンブルソースリストの形式を図 7.3-6 に示します。

図 7.3-6 アセンブルソースリストの形式

- SOURCE LISTING - (モジュール名)						
SN	LOC	OBJ	LLINE	SOURCE		
XX	XXXXXXXX	XXXXXXXXXXXXXXXXXXXXXXXXX	X	XXXXXXXXXXXXXXXX	X	-----
*1	*2	*3		*4 *5		*6 *7

上に示す形式の 1 行目の部分をソースリストヘッダとよびます。ソースリストヘッダはページごとに表示されます。

*1: セクション頭文字

セクション名の先頭 2 文字までを表示します。

*2: ロケーション

32 ビットのロケーション値を 16 進で表示します。

*3: オブジェクトコード

リンク後の決定したオブジェクトコードを 16 進で表示します。1 行にすべて表示できないときは複数行で表示します。

*4: オブジェクトコードの種類

オブジェクトコードに含まれる値の属性を次の優先順位で表示します。

I : 外部参照値

S : セクション値

空白 : 絶対値

相対アセンブルリストに表示されている "R" は絶対形式アセンブルリストでは絶対値に変換されるため表示されません。

*5: 行番号

行番号を 10 進 10 桁で表示します。

*6: プリプロセッサおよび最適コードチェックの処理状況表示

プリプロセッサ

X : アセンブル対象外となった行
 & : マクロ展開行

最適コードチェック

x : 最適化で削除された命令
 C : 最適化で別の命令に置き換えたことを示す
 O : 最適化で新たに生成した命令
 V : 最適化で下方の命令と入れ替えたことを示す (A と対)
 A : 最適化で情報の命令と入れ替えたことを示す (V と対)

*7: ソース行

ソースプログラムの 1 行を表示します。リスト 1 行におさまらない場合、複数行で表示します。

【例】

図 7.3-7 アセンブルソースリスト例

- SOURCE LISTING - (sample1)			
SN	LOC	OBJ	LLINE SOURCE
IN	002CE724	-----<INIT>-----	1025
IN	002CE724	[2] 02	1026 . DATAB. B 2, 2
IN	002CE726	0020 0010	1027 label . DATA. H 32, 16
CO	0000A280	-----<CODE>-----	1028 . SECTION CODE, CODE
CO	0000A280	9F840000043C	1029 LDI:32 #LS1, R4
			----- 行番号 ----- ソースプログラム
			----- オブジェクトコード：リンク後の確定した値で表示します。
			----- ロケーションカウンタ：32ビットの絶対アドレスで表示します。
			----- セクション頭文字：セクション名の先頭2文字までを表示します。

7.3.4 セクション情報リスト

セクション情報リストは、プログラム内で定義されるセクションについての情報を示しているものです。

セクション情報リストは、改ページを行い、新しいページから表示します。

■ セクション情報リスト

セクション情報リストの形式を図 7.3-8 に示します。

図 7.3-8 セクション情報リストの形式

FR/FR80 Family SOFTUNE Linker VxxLxx

YYYY-MM-DD

hh:mm:ss

Page: xxxx

- SECTION LISTING - (モジュール名)

NO	SECTION-NAME	SIZE	ATTRIBUTES		
xx	----- セクション名 -----	xxxxxxxxxx	xxx	xxx	xxxx=xxxxx
*1	*2	*3	*4	*5	

*1：セクション出現番号

0 から始まります。オブジェクトファイル中のセクション番号に相当します。

*2：セクション名

定義されたセクション名を出現順に表示します。

*3：セクションサイズ

セクションのサイズを 16 進 32 ビット長で表示します。

*4：セクションタイプ

セクションの種別を表示します。種別には次があります。

CODE : コードセクション

DATA : データセクション

CONST : 初期値付きデータセクション

COMMON : 共有セクション

STACK : スタックセクション

IO : IO セクション

*5：セクション配置形式

相対アセンブルリストの表示をそのまま出力します。

アセンブル時に相対セクションならば ALIGN 値を表示します。絶対セクションならば LOCATE 値を表示します。

【例】

図 7.3-9 セクション情報リスト例

FR/FR80 Family SOFTUNE Linker V60L04 2003-08-23 10:04:51 Page: 78

- SECTION LISTING - (sample1)

NO	SECTION-NAME	SIZE	ATTRIBUTES
0	DATA	00000004	DATA REL ALIGN=4
1	INIT	0000001C	DATA REL ALIGN=4
2	CONST	00000020	CONST REL ALIGN=4
3	CODE	00000038	CODE REL ALIGN=2

7.3.5 クロスリファレンスリスト

クロスリファレンスリストは、プログラム内で記述される名前についての情報とその定義と参照の関係を示すものです。

クロスリファレンスリストは、改ページを行い、新しいページから表示します。

■ クロスリファレンスリスト

クロスリファレンスリストの形式を図 7.3-10 に示します。

図 7.3-10 クロスリファレンスリストの形式

FR/FR80 Family SOFTUNE Linker VxxLxx YYYY-MM-DD hh:mm:ss Page: XXXX			
- CROSSREFERENCE LISTING - (モジュール名)			
NAME	ATTRIB.	VALUE	DEFINITION/REFERENCE
----- 名前 -----	xxxx/xxxx	xxxxxxxxxx	xxx xxx xxx
*1	*2	*3	*4

*1：名前

シンボル名、セクション名を、大文字、小文字、アルファベット順に表示します。

*2：シンボル種別

シンボルの種別を以下の形式で表示します。

ABS ：絶対シンボル
REL ：相対シンボル
ABS/EXP ：絶対シンボル（外部定義指定）
REL/EXP ：相対シンボル（外部定義指定）
IMP ：外部参照シンボル
SECT/ABS ：絶対セクション
SECT/REL ：相対セクション
UNDEFINED: 未定義シンボル
REGLIST ：レジスタシンボル

*3：値

シンボルが値を持つ場合、その値を 32 ビットの絶対アドレスで表示します。

*4：定義、参照行番号

シンボルを定義している行と参照している行を示します。

シンボルが定義されている行番号の後ろにシャープ記号 "#" が付けられます。

【例】

図 7.3-11 クロスリファレンスリスト例

FR/FR80 Family SOFTUNE Linker V60L04				2003-08-23	13:58:00	Page:	86
- CROSSREFERENCE LISTING - (sample)							
NAME	ATTRIB.	VALUE	DEFINITION/REFERENCE				
ARRSY	SECT/REL	000021C2	266	#			
BINCOL	REL/EXP	00000308	40	#	92	13	
DATA	REL	00006D58	437	#			
IROAS	ABS	00000101	79	#	10	4	

7.4 外部シンボル相互参照情報リストファイル

外部シンボル相互参照情報リストファイルは、リンク後の各オブジェクトモジュールの外部定義シンボルと外部参照シンボルのモジュール間の相互参照情報を表示します。

■ 外部シンボル相互参照情報リストファイル

外部シンボル相互参照情報リストの出力形式を図 7.4-1 に示します。

図 7.4-1 外部シンボル相互参照情報リストの出力形式

External Symbol Cross Reference List				YYYY-MM-DD	hh:mm:ss	Page:	1
Module(s)							
1.	Module01						
2.	Module02						
	.						
	.						
	.						
15.	Module15						
External Symbol Cross Reference List				YYYY-MM-DD	hh:mm:ss	Page:	2
--- symbol ---		--- type/value ---		--- module (No.) ---			
extsym1		Addr.	0x000012E8	1#	2 3 4 5 6 8 11 12 14		
extsym2		Addr.	0x000C3F34	2 3 4#	5 6 8 11 12 14		
extsym3		Addr.	0x000012E6	1#	2 3 4 5 6 8 11 12 13		
					14 15		
operator new(unsigned int)		Addr.	0x00000AAA	1 2 3 4 7#	8		
operator new[] (unsigned int)		Addr.	0x00000B1E	4 8#			
extsymunresolved		????	0x00000000	10 15			
nlp		EQU	0x00000001	4 5 6 11 12 14#			
main		Addr.	0x00000314	3#	5		

● Module(s)

1 から始まる通し番号を付けて、モジュール名を示します。

● symbol

シンボル名または関数名を表示します。(最大 50 文字)

● type/value

type には以下があります。

- Addr. : アドレス
- EQU : EQU シンボル
- Bit : ビットシンボル
- ??? : 未定義

value は、値を示します。ビットシンボルの場合、括弧内にビットポジションを示します。

● module (No.)

定義 / 参照のあったモジュールを番号で示します。# 記号は定義モジュールです。

7.5 ローカルシンボル情報リストファイル

ローカルシンボル情報リストファイルは、絶対形式ロードモジュールを構成する個々のモジュールごとのローカルシンボルを含んだ変数や関数などの情報を表示します。

このリストはデバッグ情報をもとに作成されますので、コンパイル、アセンブル時に、-g オプションを指定してください。

■ ローカルシンボル情報リストファイル

ローカルシンボル情報リストの出力形式を図 7.5-1 に示します。

図 7.5-1 ローカルシンボル情報リストの出力形式

Local Symbol List

YYYY-MM-DD

hh:mm:ss

Page: 1

Module(s)

1. Module01

2. Module02

.

.

15. Module15

Local Symbol List

YYYY-MM-DD

hh:mm:ss

Page: 2

=== Module No.1(module01) ===

--- symbol ---

--- Kind ---

--- val ---

func1(int)

Func.

g

0x000C3F34

localstatic1

Var.

s

0x000012EE

localstatic2

Var.

s

0x000012F0

.

.

=== Module No.15(module015) ===

--- symbol ---

--- Kind ---

--- val ---

operator+(USDollar &,USDollar &)

Func.

g

0x00000000

main

Func.

g

0x0000004A

USDollar::USDollar(unsigned int)

Func.

g

0x000000CA

Atable

loc.

s

0xFFFFFFFFC

Extsym

Var.

g

0x00001342

● Module(s)

1 から始まる通し番号を付けて、モジュール名を示します。

● symbol

シンボル名または関数名を表示します。

関数内で使用しているシンボルは、3 カラム目から表示します。

シンボル名の先頭から最大 50 文字迄がリストの 1 行に表示されます。

● Kind

以下のシンボル種別を表示します。

Var. : 変数 (C/C++)

Func. : 関数 (C/C++)

loc. : ローカル (C/C++)

Addr. : アドレス (ASM)

EQU : EQU シンボル (ASM)

bit : bit シンボル (ASM)

??? : 未定義

s : static (C/C++)

g : global (C/C++)

● val

シンボルの値を示します。

(注意事項) 構造体の詳細情報 (メンバ名) や, typedef 定義などは表示しません。

7.6 セクション配置詳細情報リストファイル

セクション配置詳細リストファイルは、絶対形式ロードモジュールを構成する個々のモジュールごとのセクション配置の情報を作成します。

マップリストファイルに1つにまとめたセクション全体のマップリストが表示されますが、さらに詳しいセクション配置情報を知ることができます。

■ セクション配置詳細情報リストファイル

セクション配置詳細情報リストの出力形式を図 7.6-1 に示します。

図 7.6-1 セクション配置詳細情報リストの出力形式

Section Mapping List

Module(s)

1. Module01

2. Module02

.

.

15. Module15

Section Mapping List

S.Addr. -E.Addr.

Size

Section

Type

Al

M.No.

Sec.(Top 18)

00000000-00000048

00000049

DATA

P

RW--

04

1

data

0000004A-000003E1

00000348

DATA

P

RW--

04

2

data

000003E4-.....

00000000

DATA

P

RW--

04

3

data

000003E4-00000643

00000260

DATA

P

RW-I

04

4

init

00000644-00000743

00000100

STACK

P

RW--

04

4

sectionnametoolong+
toolong

000C0000-000C0441

00000442

CODE

P

R-XI

02

1

code

000C0442-000C148B

0000104A

CODE

P

R-XI

02

3

code

*000C148C-000C201D

00000B92

CODE

P

R-XI

02

4

code

*000C2000-000C2203

00000204

CODE

P

R-XI

02

5

subprogl

000C3000-000C325F

00000260

DATA

P

R--I

04

4

#init

YYYY-MM-DD

hh:mm:ss

Page: 1

YYYY-MM-DD

hh:mm:ss

Page: 2

● Module(s)

1 から始まる通し番号を付けて、モジュール名を示します。

● S.Addr.-E.Addr.

セクションの開始と終了アドレスです。

アドレスが他と重なっているセクションは、開始アドレスの前に(*)が付加されます。

サイズ0のセクションの場合、終了アドレスの表示は「.....」になります。

● Size

許されるアドレス空間をオーバーフローしているセクションは、サイズの最大値+1を表示します。

● Section

セクションの内容種別を表示します。

CODE : プログラムセクション
DATA : データセクション
CONST : 初期値付きデータセクション
STACK : スタックセクション
IO : IO セクション

セクション種別の後に、結合属性を表示します。

P : 単純連結結合
C : 共有結合
N : 結合なし

● type

左から順に、以下の属性を表示します。

R/- : Read 可 / 不可
W/- : Write 可 / 不可
X/- : 実行可 / 不可
I/- : 初期値あり / なし

● AI

セクション配置の境界調整数を 16 進で表示します。

● M.No.

モジュール番号を表示します。Module(s) で表示のモジュール番号を示します。

● Sec.(Top xx)

セクション名を表示します。(Top xx) の xx は、セクション名を行の折り返しなしで表示できる文字数です。

セクション名の前に # がついているセクションは、実行前に RAM へ転送される初期値付きデータの配置されたセクションであることを示します。

第 8 章

リンクの制限事項および Q&A

この章では、リンクの制限事項や使用上の Q & A について述べます。

8.1 リンクの制限事項

8.2 リンクの使用上の Q&A

8.1 リンカの制限事項

リンカを使用する上で、処理可能な入力ファイル数やセクション数、シンボル数などに表 8.1-1 に示す制限があります。

■ リンカの制限事項

リンカは、入力ファイル数やセクション数、シンボル数などに表 8.1-1 に示す制限があります。

ただし、最大限界値まで処理が可能なわけではありません。

リンカは動的にメモリ獲得を行いながら処理を行います。

処理に必要なメモリ獲得ができなくなった場合には、メモリ不足のエラーメッセージを通知し、処理を中断します。

表 8.1-1 リンカの制限一覧

項 目	制限値	備 考
オプションファイル数	無制限	メモリ依存
オプションファイル内の行数	無制限	メモリ依存
オプションファイル内の 1 行の文字数	無制限	メモリ依存
オプションファイルのネスト	不可	
入力ファイル数	4,294,967,295 個	メモリ依存
入力モジュール数	4,294,967,295 個	メモリ依存
入出力ファイルサイズ	無制限	OS 依存
モジュール名 / セクション名 / シンボル名文字数	無制限	メモリ依存
ファイル名文字数	無制限	OS 依存
セクション数	4,294,967,295 個	メモリ依存
最大セクションサイズ	4G バイト	
外部定義シンボル数	4,294,967,295 個	メモリ依存
外部参照シンボル数	4,294,967,295 個	メモリ依存
外部定義シンボル参照数	無制限	メモリ依存
最大ソース数	4,294,967,295 個	メモリ依存
最大ソース行数	4,294,967,295 行	メモリ依存

■ リンカの予約名

リンカは、ROM RAM 転送機能を用いたセクションごとに "_ROM_ セクション名", "_RAM_ セクション名" でシンボルの自動生成を行います。

したがって、同名のシンボルがユーザプログラム内で定義されていると、"W1327L: このシンボルは既に定義されています (シンボル名)" が発生します。

ユーザは "_ROM_ セクション名", "_RAM_ セクション名" でシンボル定義を行わないでください。

また、同様に ROM/RAM 領域名の "_ROM_ 数値_", "_RAM_ 数値_" は CPU 情報ファイルから自動的に設定されます。

ユーザは "_ROM_ 数値_", "_RAM_ 数値_" で領域名定義 (-ro, -ra 指定) を行わないでください。

8.2 リンカの使用上の Q&A

リンカの使用に関する Question と Answer を示します。

■ リンカの使用上の Q&A

● ワイルドカードの使用

Q.	入力するオブジェクトモジュールファイル数が非常に多いのですが、ワイルドカードは使用できますか？
A.	コマンドライン上では、入力ファイルの指定にワイルドカードを使用すると、リンカが展開して実行します。オプションファイル中にも入力ファイル名の指定ができますが、ここでもワイルドカードが利用できます。下記の例を参考にして、使用してください。
例	<pre>flnk911s *.obj -o outfile.abs flnk911s mactrl.obj xz???.obj</pre>

Q.	セクション配置指定時、セクション名にワイルドカードが使用できますが、どのように使用したらよいですか？
A.	内容種別が同じセクションをまとめたいとき、または多数のセクション名を用いてプログラムを作成したときに使用すると便利な場合があります。セクション名を命名するときに、リンカでワイルドカードが使用できることを考慮して、キーワードとなる文字を決めておくことが必要になるかもしれません。
例	<p>内容種別が data のセクションとして、DTdata1, DTdata2, DTdata3, DTdata4 ...</p> <p>内容種別が code のセクションとして、CDprog1, CDprog2, CDprog3, CDprog4, ...</p> <p>のような名前でセクション名定義をしているとします。この場合、下記のような指定方法が選択できます (-sc オプションの部分のみ示します)。</p> <pre>-sc DTdata1+DTdata2+DTdata3+DTdata4=0x1000, CDprog1+CDprog2+CDprog3+CDprog4=0x3000 -sc DT*=0x1000, CD*=0x3000 -sc */data=0x1000, */code=0x3000</pre>

● 初期値付変数の扱い

Q.	<p>C/C++ コンパイラを使用した組み込み用のプログラム開発においては、初期値付変数が生成されますが、このデータはプログラム実行時に書換えられるため、実行時は RAM 上にはなりません。</p> <p>プログラム作成時の手順と注意事項を教えてください。</p>
A.	<p>組み込み用のプログラムでは、初期値付変数は最初 ROM にあり、参照するときには RAM にはなりません。したがって、プログラムでの参照アドレスは RAM にし、アプリケーション実行前に ROM から RAM への初期値データを転送するしくみを実現しておかないとプログラムの実行ができないことになります。</p> <p>リンカがサポートする ROM RAM 転送セクション機能を用いることでこのしくみが実現されています。</p> <p>C/C++ コンパイラが生成する初期値付変数は、名前が INIT のセクションにまとめられます。</p> <p>プログラム作成中は、初期値付変数の総バイト数と RAM のサイズに注意する以外は特に気を付けることはありません。</p> <p>ROM RAM 転送セクション機能については「5.9 ROM RAM 転送セクション」を参照してください。</p> <p>ユーザは、初期値データの転送プログラムをアセンブラなどで記述する必要があります。</p> <p>例に「初期値データ転送のプログラム例」を示します。</p>
例	<p>[初期値データ転送のプログラム例: (2バイトずつデータを転送しています。)]</p> <pre> .import _ROM_INIT, _RAM_INIT..... .section INIT,data,align=4..... .section start,code,align=2 LDI #(sizeof INIT+1) & ~0x1,R13 CMP #0, R13 BEQ NOT_INIT LDI #_ROM_INIT, R2 LDI #_RAM_INIT, R3 LOOP: ADD #-2, R13 LDUH @(R13, R2), R0 BGT:D LOOP STH R0, @(R13, R3) NOT_INIT: </pre> <p>_ROM_INIT は、ROM 上の INIT セクション (転送元) の先頭アドレスを示すシンボルです。_RAM_INIT は、RAM 上の INIT セクション (転送先) の先頭アドレスを示すシンボルです。これらのシンボルは、リンカの ROM RAM 転送セクション機能で自動的に生成されます。import 命令を使用して宣言しておいてください。</p> <p>セクションサイズ (転送サイズ) の抽出のため、内容のない INIT セクションを定義します。</p>

第 3 部 ライブラリアン編

ライブラリアンの仕様 , オプション , 出力リストなどについて説明します。

第 9 章 ライブラリアンの仕様

第 10 章 ライブラリアンのオプション

第 11 章 ライブラリアンのリストフォーマット

第 12 章 ライブラリアンの制限事項および Q&A

第9章

ライブラリアンの仕様

ライブラリアンの機能と各機能概要を説明します。
ライブラリファイルを作成するツールです。

- 9.1 ライブラリアンの機能
- 9.2 ライブラリアンの機能分類
- 9.3 ライブラリファイルの作成編集
- 9.4 ライブラリファイル内のモジュール抽出
- 9.5 ライブラリのデバッグ情報の削除
- 9.6 ライブラリファイルの内容チェックと表示
- 9.7 SOFTUNE V3/V5 言語ツールで作成されたオブジェクト
- 9.8 SOFTUNE V3/V5 言語ツールで作成されたライブラリ
- 9.9 FR 用オブジェクトと FR80 用オブジェクトの混在

9.1 ライブラリアンの機能

ライブラリアンは、アセンブラが出力した複数のオブジェクトモジュールをまとめて、1つのライブラリファイルを作成するツールです。

■ ライブラリアンの役割

プログラムを開発する場合は、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。

コンパイルおよびアセンブルした結果は、リンカによって1つに結合し目的とするプログラムを作成します。

ライブラリアンは、アセンブラが出力した複数のオブジェクトモジュールをまとめてライブラリファイルを作成するツールです。

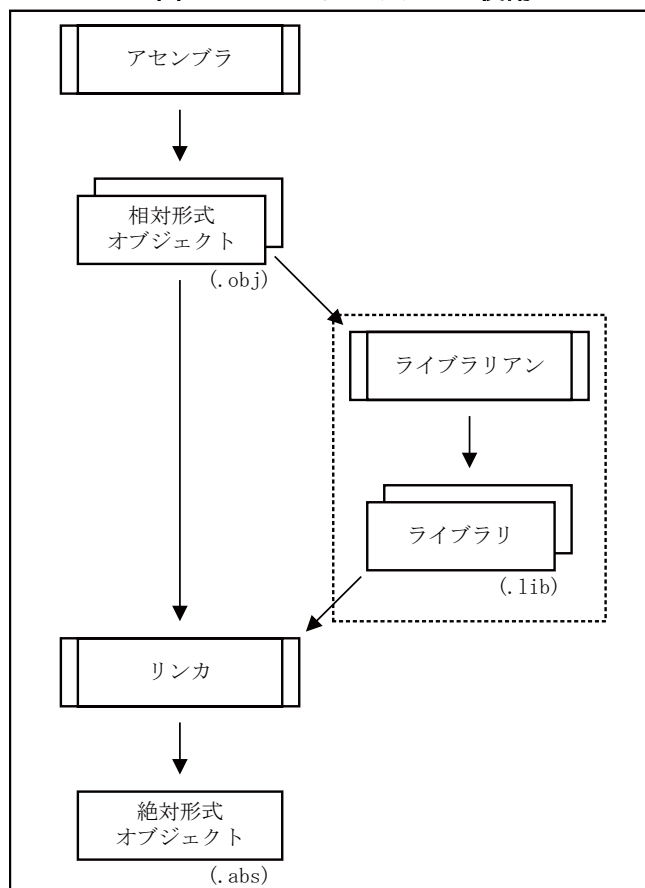
プログラムを構成する複数のモジュールを、そのプログラム専用のライブラリファイルに登録することにより、一括して管理できます。

よく利用するモジュールをまとめて登録し、汎用ライブラリファイルを作成しておけば、ほかのプログラムへの流用が容易に行えます。Cのライブラリは、このようにして利用されている良い例です。

ライブラリファイルは、ライブラリアンによってモジュール単位に追加、削除、置換などの編集が行えるため、各モジュールを最新の状態で保管できます。

図 9.1-1 に、ライブラリアンの役割を示します。

図 9.1-1 ライブラリアンの役割



9.2 ライブラリアンの機能分類

ライブラリアンには、次の6つの機能があります。

- ライブラリファイルの新規作成
 - ライブラリファイルの編集
 - ライブラリファイル内のモジュール抽出
 - デバッグ情報の削除
 - ライブラリファイルの内容チェック
 - ライブラリファイルの内容表示
-

■ ライブラリファイルの新規作成

オブジェクトモジュールファイルを入力ファイルとして、新規にライブラリファイルを作成する機能です。

■ ライブラリファイルの編集

既に作成されているライブラリファイルに、新たにオブジェクトモジュールを追加したり、不要になったオブジェクトモジュールを削除したりする機能です。

また、ライブラリファイルに登録したモジュールに障害があることが判ったり、機能の変更を行う場合、修正して差し替える必要があります。この作業は削除と追加でもできますが、置換を行う機能もあります。

■ ライブラリファイル内のモジュール抽出

ライブラリファイルに登録されているオブジェクトモジュールを抽出し、オブジェクトモジュールファイルの形式に戻す機能です。

■ デバッグ情報の削除

デバッグ情報付のオブジェクトモジュールが登録されているとき、デバッグ情報のみを取り除いて登録しなおします。

■ ライブラリファイルの内容チェック

ライブラリファイルを作っているオブジェクトモジュールの集まりの中で、外部シンボルの定義 / 参照関係がきちんと解決されているかを調べます。

デバッグ情報付のオブジェクトモジュールが登録されているか否かを調べます。

■ ライブラリファイルの内容表示

ライブラリファイルに登録されているモジュール名や、外部シンボルなどの情報をリストファイルまたは標準出力に出力します。

9.3 ライブラリファイルの作成編集

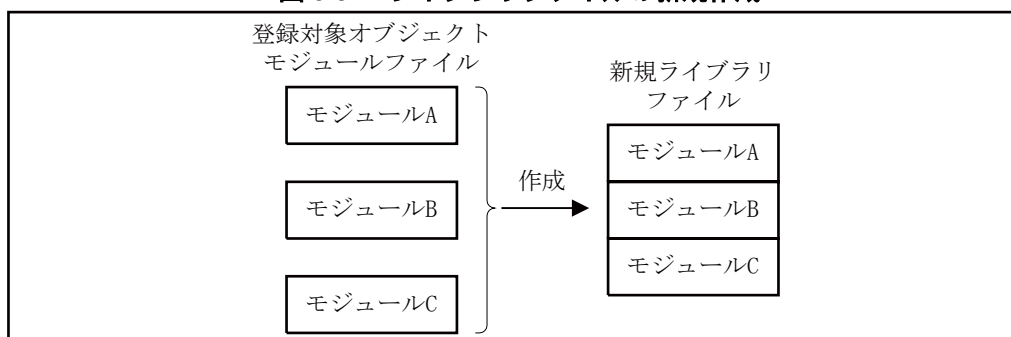
アセンブラが出力した（複数の）オブジェクトモジュールを1つにまとめ、ライブラリファイルとして登録できます。

また、既に作成済のライブラリファイルに対して、モジュールの追加、削除、置換ができます。

■ ライブラリファイルの新規作成

アセンブラが出力した（複数の）オブジェクトモジュールを1つにまとめライブラリファイルとして登録できます（図9.3-1を参照）。

図 9.3-1 ライブラリファイルの新規作成



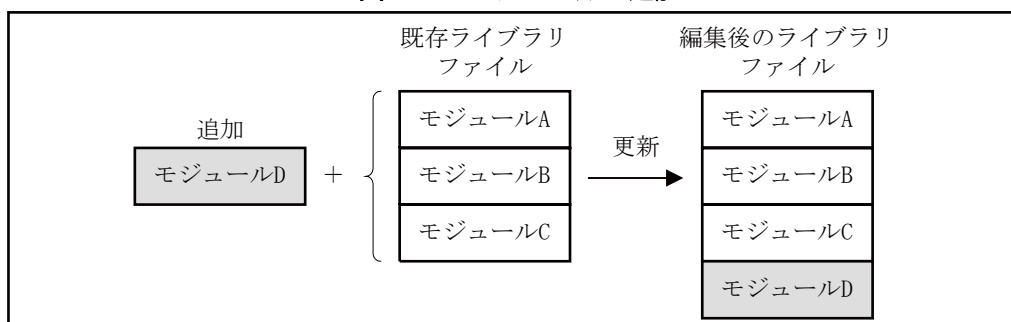
■ ライブラリファイルの編集

すでに作成済のライブラリファイルに対して、モジュールの追加、削除、置換ができます。

● モジュールの追加

既存のライブラリファイルにモジュールを追加します（図9.3-2を参照）。

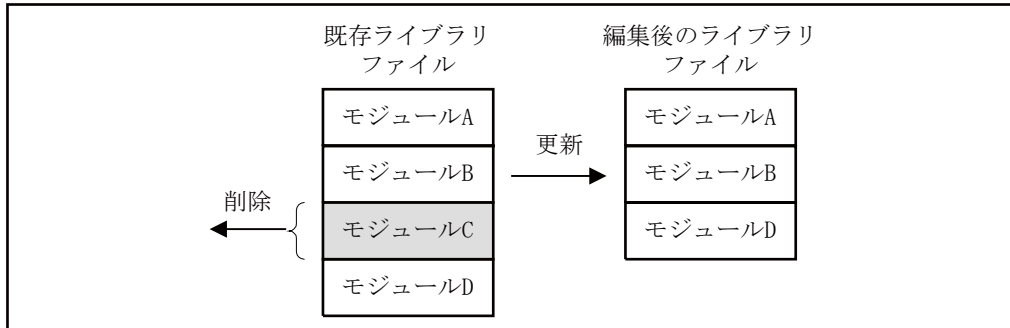
図 9.3-2 モジュールの追加



● モジュールの削除

既存のライブラリファイルから不要になったモジュールを削除します（図 9.3-3 を参照）。

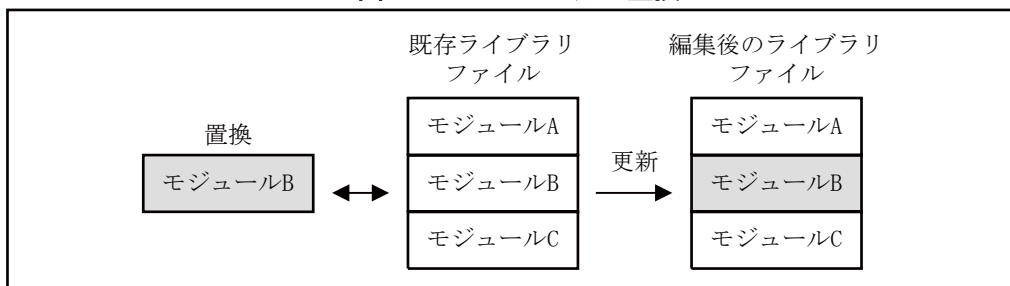
図 9.3-3 モジュールの削除



● モジュールの置換

既存の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ます（図 9.3-4 を参照）。

図 9.3-4 モジュールの置換



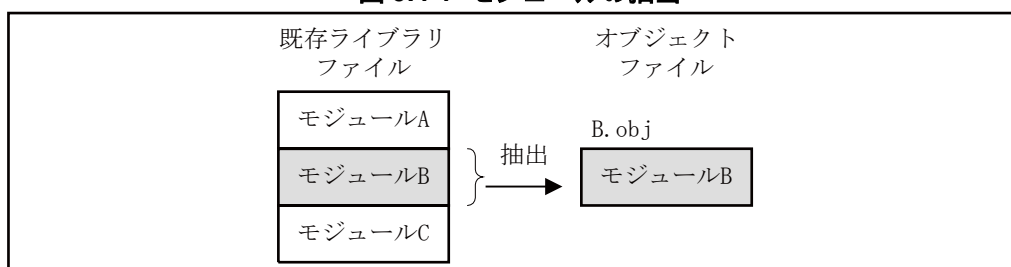
9.4 ライブラリファイル内のモジュール抽出

ライブラリファイルからモジュールを抽出し、オブジェクトモジュール形式ファイルに戻します。

■ ライブラリファイル内のモジュール抽出

ライブラリファイルからモジュールを抽出し、オブジェクトモジュール形式ファイルに戻します(図 9.4-1 を参照)。

図 9.4-1 モジュールの抽出



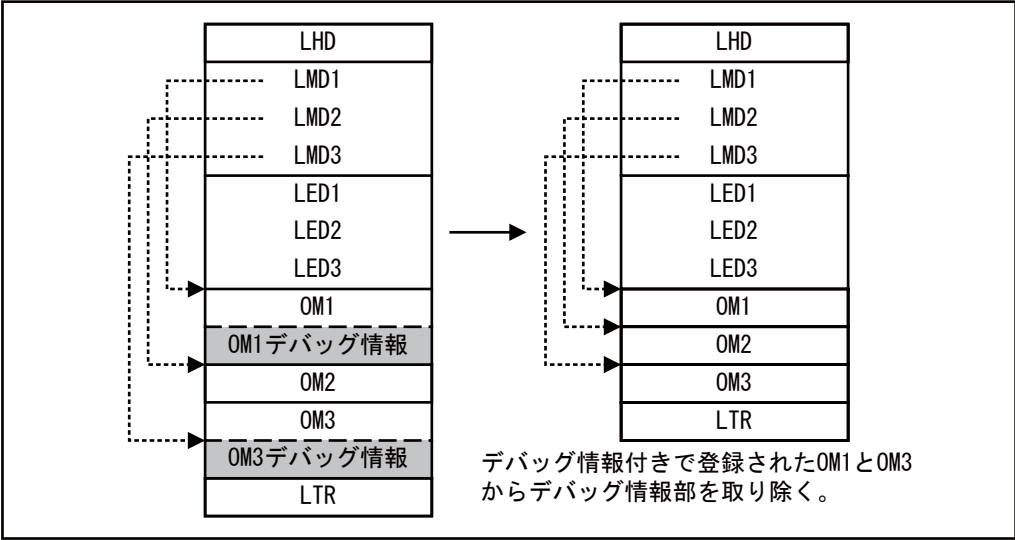
9.5 ライブラリのデバッグ情報の削除

ライブラリに登録されているオブジェクトモジュールにデバッグ情報ブロック付きのものがある場合、デバッグ情報部を取り除いて登録しなおします。

■ デバッグ情報の削除

ライブラリに登録されているオブジェクトモジュールにデバッグ情報ブロック付きのものがある場合、デバッグ情報部を取り除いて登録しなおします（図 9.5-1 を参照）。

図 9.5-1 デバッグ情報の削除



9.6 ライブラリファイルの内容チェックと表示

次の 2 項目のチェックをします。

- ライブラリ内未解決外部参照シンボルの有無
- デバッグ情報付きモジュールの有無

また、ライブラリファイルの作成 / 更新日時、モジュールの登録日時、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の名前などの情報を知ることができます。

■ ライブラリファイルの内容チェック

次の 2 項目のチェックをします。

● ライブラリ内未解決外部参照シンボルの有無

リンカでは、ライブラリから取り込んだモジュールに外部参照シンボルが含まれていた場合、まず始めに同じライブラリファイル中で定義シンボルを探します。

したがって、1 つのライブラリファイル中では、あるモジュール中に外部参照シンボルがある場合、該当する外部定義シンボルを含むモジュールが同じライブラリファイル内に必ず登録されていることが推奨されます。

1 つのライブラリファイル中での、外部参照・定義シンボルの対応関係を調べ、定義のない参照シンボルが残っている場合に、診断メッセージを出力します。

● デバッグ情報付モジュールの有無

デバッグ情報付のオブジェクトモジュールが含まれている場合に、診断メッセージを出力します。

-g オプション指定でのライブラリファイルへのモジュール登録処理では、オブジェクトモジュール中にデバッグ情報が含まれていても、取り除くことなくそのままライブラリに登録します。

これは、ライブラリに登録したモジュールのデバッグを行うことを考慮したためですが、動作確認が済めばデバッグ情報は必要がなくなります。

■ 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、リストファイルまたは標準出力へ編集出力します。

このリストにより、ライブラリファイルの作成 / 更新日時、モジュールの登録日時、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の名前などの情報を知ることができます。

表示内容については、「第 11 章 ライブラリアンのリストフォーマット」で説明します。

9.7 SOFTUNE V3/V5 言語ツールで作成されたオブジェクト

ライブラリアンは、SOFTUNE V3 または V5 言語ツールで作成されたオブジェクトも入力可能です。

■ SOFTUNE V3/V5 言語ツールで作成されたオブジェクト

ライブラリアン (flibs) は、SOFTUNE V3 または V5 用言語ツールで作成されたオブジェクトもライブラリ内に混在させることが可能です。

SOFTUNE V3 または V5 用言語ツールで作成されたオブジェクトの処理を行った場合に、ライブラリアンは、インフォメーションを出力します。

9.8 SOFTUNE V3/V5 言語ツールで作成されたライブラリ

ライブラリアンは、SOFTUNE V3 または V5 言語ツールで作成されたライブラリの編集を行うことができます。

■ SOFTUNE V3/V5 言語ツールで作成されたライブラリ

ライブラリアンは、SOFTUNE V3 または V5 用言語ツールで作成されたライブラリの編集を行うことができます。

SOFTUNE V3 または V5 用言語ツールで作成されたライブラリの編集を行った場合に、ライブラリアンは、インフォメーションを出力します。

さらに、自動的に編集前のライブラリファイルの拡張子を bak に変更したバックアップファイルを作成します。

9.9 FR 用オブジェクトと FR80 用オブジェクトの混在

ライブラリアン (flibs) は、-cpu オプションで設定したターゲット CPU が FR80 の場合、FR 用オブジェクトが混在すると警告を出力します。

-cpu オプションで指定したターゲット CPU が FR の場合、FR80 用オブジェクトが混在するとエラーを出力します。

■ FR 用オブジェクトと FR80 用オブジェクトの混在

ライブラリアン (flibs) は、-cpu オプションで設定したターゲット CPU が FR80 の場合、FR 用オブジェクトが混在すると警告を出力します。

ターゲット CPU が FR80 のときの FR 用オブジェクト混在警告は、オブジェクト混在チェックレベル指定オプション (-omcl) でエラー、または混在許可 (メッセージ出力なし) に変更できます。

ライブラリアンは、-cpu オプションで指定したターゲット CPU が FR の場合、FR80 用オブジェクトが混在すると必ずエラーを出力します。

< 注意事項 >

ターゲット CPU が FR80 の時に、FR 用オブジェクトを混在する場合、表 9.9-1 に示す FR と FR80 の命令非互換に十分注意してください。

■ FR と FR80 の命令非互換

FR と FR80 では「表 9.9-1 FR と FR80 の命令非互換」に示すように完全な命令互換ではありません。

このため、ライブラリアンでは -cpu オプションで指定したターゲット CPU が FR80 の場合、FR 用オブジェクトが混在すると警告を出力します。

-cpu オプションで指定したターゲット CPU が FR の場合、FR80 用オブジェクトが混在するとエラーを出力します。

表 9.9-1 FR と FR80 の命令非互換

非互換命令	FR	FR80
LDRES @Ri+,#u4		×
STRES #u4,@Ri		×
COPOP #u4,#CC,CRj,CRi		×
COPLD #u4,#CC,Rj,CRi		×
COPST #u4,#CC,CRj,Ri		×
COPSV #u4,#CC,CRj,Ri		×
SRCH0 Ri	×	
SRCH1 Ri	×	
SRCHC Ri	×	

: 命令あり × : 命令なし

■ FR と FR80 の共通オブジェクト

FR/FR80 共通オブジェクトは、表 9.9-1 に示す命令がないオブジェクトです。

このため、FR/FR80 共通オブジェクトは、-cpu オプションで指定したターゲット CPU が FR と FR80 のどちらの場合もリンクすることができます。

FR/FR80 共通オブジェクトの作成方法の詳細は、『FR ファミリ SOFTUNE アセンブラ マニュアル V6 対応』を参照してください。

■ -cpu オプションと混在可能な CPU 一覧

表 9.9-2 に、-cpu オプションと混在可能なオブジェクトを示します。

表 9.9-2 -cpu オプションと混在可能な CPU 一覧

ライブラリ編集時の -cpu オプション指定	オブジェクト作成時の -cpu オプション		
	FR	FR80	FR/FR80 共通 オブジェクト
FR		×	
FR80			

: 同一ターゲットです。混在ではありません。

: FR/FR80 共通オブジェクトのため、混在可能です。

: ライブラリ編集時に警告が出力されます。警告は、-omcl オプションでエラー、混在許可 (メッセージ出力なし) に変更できます。

× : 混在できません。ライブラリアンはエラーを出力します。

第10章

ライブラリアンのオプション

この章では、ライブラリアンの各オプションの形式、パラメータ、注意事項などについて説明します。

10.1 ライブラリアンのオプション一覧

10.2 ライブラリアンのオプション詳細

10.1 ライブラリアンのオプション一覧

ライブラリアンの動作を細かく指示するために、オプションがあります。

■ オプション一覧

オプション名と機能概要を、表 10.1-1 に示します。

オプションに必要なパラメータや機能の詳細は、各オプションの説明を参照してください。

表 10.1-1 ライブラリアンのオプション一覧

	機 能	オプション	備 考
ライブラリの作成、編集に関するオプション	モジュールの追加 (登録)	-a	
	モジュールの置換 (登録)	-r	
	モジュールの削除	-d	
	モジュールの抽出	-x	
出力リストに関するオプション	リストファイルの出力指定	-m	
	リストファイルの出力抑止指定	-Xm	デフォルト
	リストファイルの詳細情報の出力指定	-dt	s, d, r, a
	リスト 1 ページの行数指定	-pl	デフォルト 60
	リスト 1 行の桁数指定	-pw	デフォルト 80
ファイル検索、保護に関するオプション	バックアップファイルの作成	-b	
	バックアップファイルの作成抑止	-Xb	デフォルト
	ライブラリファイルの内容検査	-c	
	ファイル内容の最適化	-O	
その他のオプション	デバッグ情報の未削除指定	-g	
	デバッグ情報の削除指定	-Xg	
	ターゲット CPU 指定	-cpu	必須
	CPU 情報ファイル指定	-cif	
	オブジェクト混在チェックレベル指定	-omcl	デフォルト 1
共通オプション	デフォルトオプションファイル読み込み抑止指定	-Xdof	
	オプションファイル読み込み指定	-f	
	ヘルプメッセージ表示指定	-help	
	版数 / メッセージ出力指定	-V	
	版数 / メッセージ出力抑止指定	-XV	デフォルト
	終了メッセージ出力指定	-cmsg	
	終了メッセージ出力抑止指定	-Xcmsg	デフォルト
	ワーニング発生時の終了コードを 1 にする指定	-cwno	
	ワーニング発生時の終了コードを 0 にする指定	-Xcwno	デフォルト

10.2 ライブラリアンのオプション詳細

ここでは、ライブラリアンの各オプションについて説明します。

なお、リンケージキットで共通のオプションは、「第 3 章 共通オプション」で説明しています。

■ ライブラリの作成，編集に関するオプション

ライブラリの作成，編集に関するオプションの詳細を「10.2.1 モジュールの追加（登録）(-a)」～「10.2.4 モジュールの抽出(-x)」で説明します。

■ 出力リストに関するオプション

出力リストに関するオプションの詳細を「10.2.5 リストファイルの出力指定(-m)」～「10.2.9 リスト 1 行の桁数指定(-pw)」で説明します。

■ ファイル検索，保護に関するオプション

ファイル検索，保護に関するオプションの詳細を「10.2.10 バックアップファイルの作成(-b)」～「10.2.13 ファイル内容の最適化(-O)」で説明します。

■ その他のオプション

その他のオプションの詳細を「10.2.14 デバッグ情報の未削除指定(-g)」～「10.2.17 ターゲット CPU 指定(-cpu)」で説明します。

10.2.1 モジュールの追加 (登録) (-a)

ライブラリファイルを新規に作成する場合、既にあるライブラリファイルにモジュールの追加を行う場合には -a オプションを使用します。

■ モジュールの追加 (登録) (-a)

【記述形式】

-a <オブジェクトモジュールファイル名> [, ...]

【パラメータ】

<オブジェクトモジュールファイル名>

アセンブラが出力したオブジェクトモジュールファイル名

【説明】

ライブラリファイルへ登録するモジュールを内容とするファイル名を指定します。

ファイル名に拡張子の指定がないときは、".obj" を拡張子とします。

登録しようとするモジュールと同じ名前のモジュールが既に登録されている場合、エラーメッセージを出力しモジュールの登録処理は行いません。

同名の外部定義シンボルがある場合も同様に登録処理は行いません。

<オブジェクトモジュールファイル名> の指定には、ワイルドカードが使用できます。

【例 1】

```
flibs syslib.lib -a mod1.obj,mod2.obj,modx.obj
```

オブジェクトモジュールファイル mod1.obj, mod2.obj, modx.obj をライブラリファイル syslib.lib に登録します。

- syslib.lib がいないとき： 新規作成
- syslib.lib があるとき： 追加登録

【例 2】

```
flibs syslib -a mod*.obj -a chksw
```

カレントディレクトリ中の拡張子 .obj のオブジェクトモジュールファイルの内先頭 3 文字が mod であるものと、chksw.obj を登録します。

<注意事項>

ワイルドカードを使用する場合は、<オブジェクトモジュールファイル名> をカンマで区切って指定できません。上述の例のように複数の -a オプションに分けて指定してください。

ワイルドカード指定時は拡張子の補完は行いませんので、必ず拡張子まで指定してください。

ファイル名のワイルドカードの展開は、OS に依存しますので、「付録 G OS による仕様の相違点」を参考にしてください。

10.2.2 モジュールの置換 (登録) (-r)

既に作成しているライブラリファイル中のモジュールを新しい同名のモジュールと置き換えます。

■ モジュールの置換 (登録) (-r)

【記述形式】

```
-r <オブジェクトモジュールファイル名> [, ... ]
```

【パラメータ】

<オブジェクトモジュールファイル名>

アセンブラが出力したオブジェクトモジュールファイル名

【説明】

指定したファイル内の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はモジュールの置換を行い、存在しない場合には指定モジュールを登録します。

ファイル名に拡張子の指定がないときは ".obj" が拡張子となります。

<オブジェクトモジュールファイル名> の指定には、ワイルドカードが使用できます。

【例 1】

```
flibs syslib.lib -r loadx.obj,loady.obj
```

loadx.obj および loady.obj 内の 2 つのモジュールを、編集集中のライブラリファイル内の同名モジュールと置換えます。

ライブラリファイル中に同名モジュールがない場合には、追加登録されます。

【例 2】

```
flibs syslib.lib -r load?.obj
```

カレントディレクトリ中の拡張子 .obj のオブジェクトモジュールファイルのうち先頭 4 文字が load であり、その後ろに任意の 1 文字が付くファイル内のモジュールを置換えの対象とします。

< 注意事項 >

ワイルドカードを使用する場合は、<オブジェクトモジュールファイル名> をカンマで区切って指定できません。複数の -r オプションに分けて指定してください。

ワイルドカード指定時は拡張子の補完は行いませんので、必ず拡張子まで指定してください。

ファイル名のワイルドカードの展開は、OS に依存しますので、「付録 G OS による仕様の相違点」を参考にしてください。

10.2.3 モジュールの削除 (-d)

ライブラリファイルから不要になったモジュールを取り除きます。

■ モジュールの削除 (-d)

【記述形式】

`-d <モジュール名> [, ...]`

【パラメータ】

<モジュール名>

削除するモジュール名

【説明】

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。

ここでの指定は、ファイル名ではなくモジュール名ですので注意してください。

【例】

```
flibs syslib.lib -d inchar,outchar
```

syslib.lib から、2 つのモジュール inchar と outchar を削除します。

< 注意事項 >

アセンブラの疑似命令でモジュール名の指定ができますが、特別な理由がない限りオブジェクトモジュールファイル名とモジュール名は同じにしてください。異なる名前にすると、ライブラリの編集作業においては誤りの原因になります。

ファイル名とモジュール名を同じにするためには、英数字とアンダバーだけを使用したファイル名を用います。

モジュール名を確認したい場合は、リスト出力オプション (-m) を使用してリストファイルの先頭部分に出力されるモジュール名を参照してください。

10.2.4 モジュールの抽出 (-x)

ライブラリファイルからモジュールを抽出し，登録前のオブジェクトモジュールファイルに戻します。

■ モジュールの抽出 (-x)

【記述形式】

<code>-x <モジュール名> [, <オブジェクトモジュールファイル名>]</code>
--

【パラメータ】

<モジュール名>

抽出するモジュール名

<オブジェクトモジュールファイル名>

抽出モジュールの出力ファイル名

【説明】

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。

抽出したモジュールは，登録前と同じ形式のオブジェクトモジュールファイルになります。

<オブジェクトモジュールファイル名> の指定を省略すると，<モジュール名> に拡張子 ".obj" を付けた名前でファイルを作ります。

【例】

```
flibs syslib -x add
```

```
flibs syslib.lib -x add,add.obj
```

既存ライブラリファイルから，モジュール add を抽出し add.obj ファイルを作成します。

```
flibs syslib -x add,add.o
```

既存ライブラリファイルから，モジュール add を抽出し add.o ファイルを作成します。

<注意事項>

-x オプションは，抽出したいモジュールの数だけ指定できます。

同じモジュール名が指定された場合は，後に指定した方が有効になります。以下の例では，add.obj は作成されず，addfunc.obj のみ作成されます。

```
flibs syslib -x add -x add,addfunc.obj
```

SOFTUNE V3 用言語ツールで作成されたモジュールは，従来のオブジェクト形式で取り出されます。

10.2.5 リストファイルの出力指定 (-m)

ライブラリファイルに登録されているモジュール名や外部定義シンボル名などの情報リスト出力を行います。

■ リストファイルの出力指定 (-m)

【記述形式】

```
-m { <リストファイル名> | - }
```

【パラメータ】

<リストファイル名>

出力するライブラリアンリストのファイル名を指定。リストを標準出力に出力する場合にはハイフン (-) を指定。

【説明】

ライブラリファイルに登録されているモジュール名や外部定義シンボル名などの情報リスト出力を行います。

<リストファイル名> に拡張子指定がない場合, ".mp2" を付加します。

このオプションにより出力される情報は登録されているモジュール名ですが, 後述の -dt オプションで, より詳しい情報を表示させることができます。

リスト内容は, ライブラリアンの処理終了時の状態を示します。

ほかに編集作業に関するオプションがない場合には, 指定したライブラリファイルの内容がリストされます。

ライブラリファイルの内容をリストに残さずに画面上で確認したい場合には, パラメータにハイフンを指定します。

【例 1】

```
flibs syslib.lib -m libx.mp2
```

syslib.lib に登録されているモジュール名リストを libx.mp2 に出力します。

【例 2】

```
flibs syslib -a obj1,obj2 -m libx.lis
```

obj1.obj と obj2.obj のモジュールを追加した結果作成された syslib.lib の内容を libx.lis に出力します。

【例 3】

```
flibs syslib -m -
```

syslib.lib に登録されているモジュール名リストを標準出力に出力します。

10.2.6 リストファイルの出力抑止指定 (-Xm)

ライブラリアンにリストファイルを出力しないことを指示します。

■ リストファイルの出力抑止指定 (-Xm)

【記述形式】

-Xm

【パラメータ】

なし

【説明】

リストファイルの出力の抑止を行います。

-m オプションの後に -Xm オプションを指定すると、-m オプションを無効にすることができます。

【例】

```
flibs syslib.lib -m libx.mp2 -Xm
```

リストファイルの作成は行わないようにします。

10.2.7 リストファイルの詳細情報の出力指定 (-dt)

-m オプションでリスト出力を行いますが、登録モジュール名の一覧を表示するだけです。

ライブラリに登録されている個々のモジュールごとにセクションや外部シンボルの情報を得たいとき、あるいはライブラリ全体の外部定義 / 外部参照シンボル情報を得たいときは、-dt オプションを使用します。

■ リストファイルの詳細情報の出力指定 (-dt)

【記述形式】

```
-dt <情報種別> [, <情報種別> ] ...
```

【パラメータ】

<情報種別>

s: モジュールごとにセクション名とそのサイズを出力します。

d: モジュールごとに外部定義シンボルを出力します。

r: モジュールごとに外部参照シンボルを出力します。

a: ライブラリ全体の外部定義シンボルと、ライブラリ内で未解決の外部参照シンボルを出力します。

【説明】

このオプション指定がないとき、リストファイルには登録されているモジュール名だけが出力されます。このオプションは、さらに詳しい情報を得たいときに使用します。

<情報種別> の指定は省略できません。

<情報種別> 指定は、複数のキーワードをカンマで区切って指定できます。

-m オプションが指定されていない場合、本オプション指定は無効になります。

【例 1】

```
flibs syslib.lib -m libx.mp2 -dt r,s
```

外部参照シンボルおよびセクション名を含むリストを libx.mp2 に出力します。

【例 2】

```
flibs syslib -m libx.lis -dt s,d,r,a
```

ライブラリアンが出力可能なすべての情報を libx.lis に出力します。

10.2.8 リスト 1 ページの行数指定 (-pl)

リスト 1 ページの出力行数をデフォルト (60 行) から変更するときに使用します。

■ リスト 1 ページの行数指定 (-pl)

【記述形式】

<code>-pl <行数></code>	(デフォルト : 60)
-----------------------------	----------------

【パラメータ】

<行数>

0, 20 ~ 255 の範囲で指定する。

【説明】

リストファイルの 1 ページに印字する行数を指定します。

0 指定は、リストファイル出力時、ページ制御を行わないようにします。

-m オプションが指定されていない場合、本オプション指定は無効になります。

【例 1】

```
flibs syslib.lib -m libx.mp2 -pl 40
```

リスト 1 ページの行数を、40 にします。

【例 2】

```
flibs syslib.lib -m - -dt s -pl 0
```

改ページなしで、セクション情報を付加したリストを標準出力に出力します。

10.2.9 リスト 1 行の桁数指定 (-pw)

リスト 1 行の出力桁数をデフォルト (80 文字) から変更するときに使用します。

■ リスト 1 行の桁数指定 (-pw)

【記述形式】

<code>-pw <桁数></code>	(デフォルト : 80)
-----------------------------	----------------

【パラメータ】

< 桁数 >

80 ~ 1023 の範囲で指定する。

【説明】

リストファイルの 1 行に印字する桁数を指定します。

長いシンボル名, セクション名, モジュール名がデフォルト桁数では 2 行以上になって見にくい場合に使用してください。

デフォルト表示桁数 (80) のとき, 1 行内で表示できる文字数は以下のとおりです。

- モジュール名: 21 文字
- セクション名: 19 文字
- シンボル名 : 34 文字

-m オプションが指定されていない場合, 本オプション指定は無効になります。

【例】

```
flibs syslib.lib -m libx.mp2 -pw 90
```

リスト 1 行の桁数を, 90 に変更します。

この場合, 1 行内で表示できる各名称の文字数は以下のようになります。

- モジュール名: 31 文字
- セクション名: 29 文字
- シンボル名 : 39 文字

< 注意事項 >

ライブラリアンのリストは, -pw オプションでの桁数指定により, 1 行内で表示するモジュール名, セクション名, シンボル名の文字数が変わります。

シンボル名は, 1 行を左右 2 つのフィールドに分けて表示しますので, 最も長いシンボル名の 2 倍 + 12 が見やすいリストになります。

10.2.10 バックアップファイルの作成 (-b)

ライブラリファイルの編集処理を行うと、ライブラリファイルの内容が変更されます。

編集前のライブラリファイルのバックアップを残したい場合に -b オプションを指定します。

■ バックアップファイルの作成 (-b)

【記述形式】

-b

【パラメータ】

なし

【説明】

ライブラリアンは、モジュールの追加や削除など編集作業を行うと、ファイル内容が変更されますので元の内容は失われます。

このオプションにより、元のファイルのバックアップファイルを作成します。

バックアップファイルの拡張子は、".bak" になります。

バックアップは1世代分しか行いませんので、大事なライブラリファイルを編集する場合は、ユーザ自身でバックアップをとってからライブラリアンを使用してください。

【例】

```
flibs syslib -a putc.obj -b
```

編集前の syslib.lib を編集後 syslib.bak として残します。

編集後の syslib.lib は、putc.obj と getc.obj が追加されます。

< 注意事項 >

入力ライブラリが SOFTUNE V3 ツールで作成されたファイルの場合、バックアップされるファイルも SOFTUNE V3 用ファイルのまま、".bak" ファイルとして保存されます。

また、入力ライブラリが SOFTUNE V5 ツールで作成されたファイルの場合、バックアップされるファイルも SOFTUNE V5 用ファイルのまま、".bak" ファイルとして保存されます。

10.2.11 バックアップファイルの作成抑止 (-Xb)

バックアップ取得オプションである `-b` を取り消すときに `-Xb` オプションを使用します。

■ バックアップファイルの作成抑止 (-Xb)

【記述形式】

<code>-Xb</code>	(デフォルト)
------------------	-----------

【パラメータ】

なし

【説明】

ライブラリアンは、デフォルト設定で編集対象のライブラリファイルのバックアップは作成しません。これは、`-Xb` オプションを指定したのと同じです。

このオプションは、`-b` オプションの指定を無効にする場合に指定します。

【例】

次の 3 つの指定は、すべて同じ処理を行います。

```
flibs syslib -a putc.obj,getc.obj
```

```
flibs syslib -a putc.obj,getc.obj -Xb
```

```
flibs syslib -b -a putc.obj,getc.obj -Xb
```

編集前の `syslib.lib` は、編集後削除されます。

編集後の `syslib.lib` には、`putc.obj` と `getc.obj` が付加されています。

10.2.12 ライブラリファイルの内容検査 (-c)

ライブラリファイルの内容の簡単な検査を行うオプションです。

■ ライブラリファイルの内容検査 (-c)

【記述形式】

-c

【パラメータ】

なし

【説明】

次の 2 項目の検査を行います。

- ライブラリ内未解決外部参照シンボルの有無

リンクは、ライブラリ内に未解決の外部参照シンボルが含まれる場合、同じライブラリ内に含まれる定義シンボルモジュールのリンクを優先します。

1 つのライブラリファイル中での、外部参照 / 定義シンボルの対応関係を調べ、該当する外部定義シンボルのない外部参照シンボルが含まれる場合に、メッセージを出力します。

- デバッグ情報付モジュールの有無

-g オプションを指定したライブラリファイルへのモジュール登録処理では、オブジェクトモジュール中のデバッグ情報を取り除くことなくそのままライブラリに登録します。

これは、ライブラリに登録したモジュールのデバッグを行うことを考慮したためです。動作確認が済めばデバッグ情報は必要がなくなります。

ライブラリ中にデバッグ情報付のモジュールが登録されているか否かを調べ、もしあればメッセージを出力します。

【例】

```
flibs syslib.lib -c
syslib.lib の内容をチェックします。
```

< 注意事項 >

ライブラリファイルの内容検査 (-c) オプションは、ほかのオプションと共に指定することはできません。

上記の指定例のように、単独で指定してください。

10.2.13 ファイル内容の最適化 (-O)

ライブラリに登録されているオブジェクトモジュールにデバッグ情報があれば取り除きます。

■ ファイル内容の最適化 (-O)

【記述形式】

-O

【パラメータ】

なし

【説明】

ライブラリに登録されているオブジェクトモジュールにデバッグ情報ブロック付のものがある場合、デバッグ情報部を取り除いて登録しなおします。

オブジェクトモジュールファイル中では、デバッグ情報がかなり大きな部分を占めていますので、デバッグ情報を削除することでライブラリファイルのサイズをかなり小さくすることができます。

【例】

```
flibs syslib -O
```

syslib.lib ファイルから、デバッグ情報を削除します。

< 注意事項 >

ファイル内容の最適化 (-O) オプションは、他のオプションと共に指定できません。
上記の指定例のように、単独で指定してください。

10.2.14 デバッグ情報の未削除指定 (-g)

ライブラリファイルにオブジェクトモジュールを登録するときに、デバッグ情報を削除しないようにするときに指定します。

■ デバッグ情報の未削除指定 (-g)

【記述形式】

-g

【パラメータ】

なし

【説明】

ライブラリアンは、通常、デバッグ情報を取り去ってライブラリファイルに登録します。

本オプションが指定された場合には、デバッグ情報の削除は行わず、そのままライブラリファイルに登録します。

ライブラリ作成後にデバッグ情報を削除したい場合には、最適化オプション -O を使用してライブラリファイルを作り直すことができます。

【例】

```
flibs syslib.lib -a inchar,outchar -g
```

inchar.obj,outchar.obj に、デバッグ情報が含まれていても削除せずにライブラリファイルに登録します。

10.2.15 デバッグ情報の削除指定 (-Xg)

デバッグ情報の未削除指定オプションである -g を取り消すときに -Xg オプションを使用します。

■ デバッグ情報の削除指定 (-Xg)

【記述形式】

-Xg

(デフォルト)

【パラメータ】

なし

【説明】

ライブラリアンは、通常オブジェクトモジュールにデバッグ情報が含まれている場合、デバッグ情報を取り去ってライブラリファイルに登録します。これは、この -Xg オプションを指定したのと同じです。

オプションは、-g オプションの指定を打ち消すために指定します。

最適化オプション -O を使用すれば、ライブラリ作成後にデバッグ情報を一括して削除できます。

【例】

次の 3 つの指定は、すべて同じ処理を行います。

```
flibs syslib.lib -a inchar,outchar
```

```
flibs syslib.lib -a inchar,outchar -Xg
```

```
flibs syslib.lib -g -a inchar,outchar -Xg
```

inchar.obj,outchar.obj に含まれているデバッグ情報は、ライブラリファイルに登録しません。

10.2.16 CPU 情報ファイル指定 (-cif)

ライブラリで使用する CPU 情報ファイルを指定します。

■ CPU 情報ファイル指定 (-cif)

【記述形式】

`-cif < CPU 情報ファイル名 >`

【パラメータ】

< CPU 情報ファイル名 >

ライブラリで使用する CPU 情報ファイル

【説明】

ライブラリで使用する CPU 情報ファイルを指定します。

【例】

```
flibs syslib.lib -a inchar,outchar -cpu MB91100A
-cif C:\Softune6\lib\911\cpu_info\MB91100A.csv
flibs syslib.lib -a inchar,outchar -cpu MB91130
-cif C:\Softune6\lib\911\911.csv
```

< 注意事項 >

SOFTUNE Tools は、CPU 情報ファイルを参照して、CPU に関する情報を取得します。
関連するツール間で異なる CPU 情報ファイルを参照した場合、作成するプログラムに問題が発生する可能性があります。

SOFTUNE Tools に標準で添付されている CPU 情報ファイルは、以下の場所にあります。

インストール先ディレクトリ \lib\911\911.csv

コンパイラとアセンブラパックを異なるディレクトリにインストールしている場合には、各ツールに対して同一の CPU 情報ファイルを参照するように、-cif で指定してください。

10.2.17 ターゲット CPU 指定 (-cpu)

ターゲット CPU の指定を行います。
ライブラリファイル化を行うプログラムのターゲット CPU を MB 番号で指定します。

■ ターゲット CPU 指定 (-cpu)

【記述形式】

<code>-cpu < MB 番号 ></code>

【パラメータ】

< MB 番号 >

ターゲット CPU の MB 番号

【説明】

ライブラリファイル化を行うプログラムのターゲット CPU を MB 番号で指定します。

【例】

```
flibs syslib.lib -a inchar,outchar -cpu MB91100A
flibs syslib.lib -a inchar,outchar -cpu MB91130
```

< 注意事項 >

ライブラリ化処理を実行する際には、本オプションでターゲット CPU の指定が必要です。
本オプションの省略はできません。

10.2.18 オブジェクト混在チェックレベル指定 (-omcl)

-cpu オプションで指定したターゲット CPU が FR80 の時に FR 用オブジェクトが混在した場合の動作を指定します。動作は混在許可 (メッセージ出力なし), 警告, エラーが指定できます。

■ オブジェクト混在チェックレベル指定 (-omcl)

【記述形式】

```
-omcl <数値>
```

【パラメータ】

<数値>

警告レベルとして 0, 1 または 2 を指定します。

【説明】

-cpu オプションで指定したターゲット CPU が FR80 の時に FR 用オブジェクトが混在した場合の動作を指定します。動作は混在許可 (メッセージ出力なし), 警告, エラーが指定できます。

0 : 混在が発生してもメッセージを出力しません。

(混在許可)

1 : 混在が発生した場合には, 警告メッセージを出力します。

(デフォルト)

2 : 混在が発生した場合には, エラーメッセージを出力します。

(混在不可)

-omcl オプションでは, -cpu オプションで指定したターゲット CPU が FR の時に FR80 用オブジェクトが混在した場合の動作は変更できません。

-cpu オプションと混在可能な CPU については, 「表 9.9-2 -cpu オプションと混在可能な CPU 一覧」を参照してください。

【例】

- -omcl オプション指定がない場合 (デフォルト動作)

```
flibs -cpu MB91680 -a module_fr.obj fr80.lib
```

```
*** W1412U: 互換性のない CPU タイプのモジュールがあります (module_fr.obj)
```

混在を警告として扱います。

- -omcl 0 を指定した場合

```
flibs -cpu MB91680 -a module_fr.obj fr80.lib -omcl 0
```

混在を検出しません

- -omcl 1 を指定した場合

```
flibs -cpu MB91680 -a module_fr.obj fr80.lib -omcl 1
```

```
*** W1412U: 互換性のない CPU タイプのモジュールがあります (module_fr.obj)
```

混在を警告として扱います。

- -omcl 2 を指定した場合

```
flibs -cpu MB91680 -a module_fr.obj fr80.lib -omcl 2
```

```
*** E4412U: 互換性のない CPU タイプのモジュールがあります (module_fr.obj)
```

混在をエラーとして扱います。

第11章

ライブラリアンのリスト フォーマット

この章では、ライブラリアンのリストファイルの構成について説明します。

11.1 リストファイルの情報内容

11.2 モジュール名リスト

11.3 モジュールごとの詳細情報

11.4 ライブラリ内の外部定義 / 参照シンボル情報

11.1 リストファイルの情報内容

ライブラリアンのリストファイルは、ライブラリファイルの内容を以下の 5 つの分類で出力します。

- モジュール名
- モジュールごとのセクション情報
- モジュールごとの外部定義シンボル情報
- モジュールごとの外部参照シンボル情報
- モジュール全体の外部定義 / 参照シンボル情報

リスト出力を行うためには、`-m` オプションおよび `-dt` オプションの指定が必要です。

■ リストファイルの構成

図 11.1-1 にリストファイルの構成を示します。

図 11.1-1 リストファイルの構成

<リストヘッダ> <ul style="list-style-type: none"> ・ライブラリファイル名 ・登録モジュール数/外部定義シンボル数 ・CPU情報など 	-mのみ
<モジュール数> <ul style="list-style-type: none"> ・登録モジュール名 	-mのみ
<div><モジュール毎のセクション情報></div>	-dt s指定
<div><モジュール毎の外部参照シンボル名情報></div>	-dt r指定
<div><モジュール毎の外部定義シンボル名情報></div>	-dt d指定
. . . 	
<div><モジュール全体の外部定義/参照シンボル名情報></div>	-dt a指定

11.2 モジュール名リスト

ライブラリアンのデフォルトリスト出力 (-dt オプション指定なしのとき) では, ライブラリファイル中の登録モジュール名だけを表示します。

■ リスト出力概要

-m オプションが指定された場合, ライブラリファイルの内容を表示します。
ライブラリアンリストの形式は, 図 11.2-1 のとおりです。

図 11.2-1 ライブラリアンリストの形式 (デフォルト)

*1	Library File Name	:	sample.lib	
*2	Number of Modules	:	3	
*3	Number of Symbols	:	9	
*4	CPU information	:	MB91101	
*5	Library Creation Date		1999-03-01 14:23:50	
*6	Library Revision Date		1999-04-17 09:41:15	
	[Module Name]	[Entry Date]	[Creation Date]	[OMF]
*7	ModuleA	1999-03-01 14:23:50	1999-03-19 10:03:21	
	ModuleB	1997-04-17 09:41:15	1996-10-07 20:18:58	*
	ModuleC	1999-03-01 14:23:50	1999-02-23 15:15:00	

- *1 : ライブラリファイル名
- *2 : ライブラリファイルに登録されているモジュールの数 (10 進数)
- *3 : 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数 (10 進数)
- *4 : CPU の情報 (MB 番号)
- *5 : ライブラリファイルを最初に作成した日時
- *6 : ライブラリファイルの最新更新日時.....新規作成時は *5 と同じ
- *7 : [Module Name] 登録モジュール名 (アルファベット順)
モジュール名は (ページ幅 - 59) 文字を 1 行に表示します。デフォルト (-pw 80) の場合は, 21 文字です。

[Entry Date] モジュールがライブラリファイルに登録された日時
[Creation Date] モジュールが作成された日時
[OMF] OMF 種別

Softune V3 用言語ツールで作成されたモジュールに * が表示されます。

11.3 モジュールごとの詳細情報

モジュールごとの詳細情報には以下の 3 つがあり，出力指示は `-dt` オプションで行います。

- セクション情報 (`-dt s`)
- 外部定義シンボル情報 (`-dt d`)
- 外部参照シンボル情報 (`-dt r`)

■ リスト出力概要

図 11.3-1 にライブラリアンリストの形式 (詳細指定時) を示します。

図 11.3-1 ライブラリアンリストの形式 (詳細指定時)

Library File Name : sample.lib			
Number of Modules : 3			
Number of Symbols : 3			
CPU information : MB91101			
Library Creation Date 1999-03-01 14:23:50			
Library Revision Date 1999-04-17 09:41:15			
[Module Name] [Entry Date] [Creation Date] [OMF]			
ModuleA 1999-03-01 14:23:50 1999-03-19 10:03:21 *			
*1	-- Section -	-- Type --	-- Size --
	code	code	0x000002E8
	data	data	0x0000006A
*2	-- Ext_Ref Symbol(s) --		
	p_text	tx_len	
*3	-- Ext_Def Symbol(s) --		
	prtext		
[Module Name] [Entry Date] [Creation Date] [OMF]			
.			
.			
.			

*1 : `-dt` オプションの `s` パラメータにより出力されます。

モジュール内のセクションの情報です。

セクション名，セクション属性，サイズを表示します。

セクション名は，(ページ幅 - 61) 文字を 1 行に表示します。

*2 : `-dt` オプションの `r` パラメータにより出力されます。

モジュール内の外部参照シンボル名を 1 行に 2 シンボルずつ表示します。

*3 : `-dt` オプションの `d` パラメータにより出力されます。

モジュール内の外部定義シンボル名を 1 行に 2 シンボルずつ表示します。

外部シンボル名は ((ページ幅 - 12) / 2) 文字を 1 行に表示します。

11.4 ライブラリ内の外部定義 / 参照シンボル情報

ライブラリファイルに登録されているすべてのモジュールに関する、外部定義シンボルおよび外部参照シンボルの情報を表示できます。出力指示は `-dt` オプションで行います。(`-dt a`)

■ リスト出力概要

図 11.4-1 にライブラリアンリストの形式 (詳細指定時) を示します。

図 11.4-1 ライブラリアンリストの形式 (詳細指定時)

Library File Name : sample.lib			
Number of Modules : 3			
Number of Symbols : 3			
CPU information : MB91101			
Library Creation Date 1999-03-01 14:23:50			
Library Revision Date 1999-04-17 09:41:15			
[Module Name]	[Entry Date]	[Creation Date]	[OMF]
ModuleA	1999-03-01 14:23:50	1999-03-19 10:03:21	*
	.		
	.		
	.		
*1 [ALL Ext_Def Symbol(s)]			
chriget		p_text	
prtext			
*2 [ALL Ext_Ref Symbol(s)]			
chr_get		tx_len	

*1 : `-dt` オプションの `a` パラメータにより出力されます。

ライブラリファイル内全体の外部定義シンボル名の一覧を、1 行に 2 シンボルずつ表示します。

*2 : ライブラリファイル内で対応する外部定義シンボルのない外部参照シンボルの一覧を、1 行に 2 シンボルずつ表示します。

外部シンボル名は $((\text{ページ幅} - 12) / 2)$ 文字を 1 行に表示します。

第12章

ライブラリアンの制限事項 およびQ&A

この章では、ライブラリアンの制限事項や使用上のQ&Aについて述べます。

12.1 ライブラリアンの制限事項

12.2 ライブラリアンの使用上のQ&A

12.1 ライブラリアンの制限事項

ライブラリアンを使用する上で、1 つのライブラリファイルに登録可能なモジュール数や外部シンボル数の制限や、注意しなくてはならない事柄を説明します。

■ ライブラリアンの制限事項

ライブラリアンには、表 12.1-1 に示す制限があります。

ただし、最大限界値まで処理が可能なわけではありません。

ライブラリアンは動的にメモリ獲得を行いながら処理を行います。

処理に必要なメモリ獲得ができなくなった場合には、メモリ不足のエラーメッセージを通知し、処理を中断します。

表 12.1-1 ライブラリアンの制限一覧

項 目	制限値	備 考
オプションファイル数	無制限	メモリ依存
オプションファイル内の行数	無制限	メモリ依存
オプションファイル内の 1 行の文字数	4,095 文字	
オプションファイルのネスト	不可	
入力ファイル数	4,294,967,295 個	メモリ依存
入力モジュール数	4,294,967,295 個	メモリ依存
入出力ファイルサイズ	無制限	OS 依存
モジュール名 / セクション名 / シンボル名文字数	無制限	メモリ依存
ファイル名文字数	無制限	OS 依存
セクション数	4,294,967,295 個	メモリ依存
最大セクションサイズ	4G バイト	
外部定義シンボル数	4,294,967,295 個	メモリ依存
外部参照シンボル数	4,294,967,295 個	メモリ依存
外部定義シンボル参照数	無制限	メモリ依存

■ 必要なディスク容量についての注意

既存のライブラリファイルの編集を行い、バックアップファイルを作成する際は、以下の点にご注意ください。

- 新しく作成するライブラリファイルと既存のバックアップファイルが格納できる空き容量がある。

■ オプションの指定に関する注意

ライブラリファイルの内容検査 (-c) およびファイル内容の最適化 (-O) の両オプションは、それぞれ単独で指定してください。

ほかのオプションとの併用はできません。

12.2 ライブラリアンの使用上の Q&A

ライブラリアンの使用上に関する Question と Answer を示します。

■ ライブラリファイルの作成に関する Q&A

Q.	ライブラリファイルに登録できるファイル形式は何ですか？
A.	アセンブラが出力する、オブジェクトモジュールです。 これは、デフォルト拡張子が、(.obj) で作成されるファイルです。
例	<pre> fasm911s file1 file1.obj 出力 fasm911s file2 file2.obj 出力 flibs libfile -a file1.obj,file2.obj </pre>

Q.	自作したライブラリファイルからの取り込みモジュールに障害があるようなので、デバッグしたいのですがシンボル情報が使えません。
A.	<p>デバッグ情報付きのオブジェクトモジュールでないと、デバッグ時にシンボル情報の使用はできません。 デバッグ情報付きで作成したオブジェクトモジュールに差し替える必要があります。 デバッグの必要がありそうなオブジェクトモジュールをライブラリ化して使用する場合には、あらかじめデバッグ情報付で登録 (-g オプション) しておくとう良いでしょう。 デバッグが終了した時点で、デバッグ情報を削除 (-O オプション) することができます。</p>
例	<pre> fasm911s file1 -g デバッグ情報付 file1.obj 出力 flibs libfile -r file1 -g デバッグ情報付 libfile.lib 出力 flibs libfile -O デバッグ情報無 libfile.lib 出力 </pre>

Q.	汎用に作成したサブルーチンをライブラリにまとめたいと思っていますがオブジェクトモジュールの数が非常に多く、全部のファイル名を指定するのが面倒です。
A.	オブジェクトモジュールのライブラリへの追加 (-a オプション) と置換 (-r オプション) には、ワイルドカード指定ができます。
例	<pre> flibs libfile -a "*.obj" 拡張子 ".obj" のファイルをすべて登録 </pre>

Q.	以前に作成したライブラリファイルの内容がわからなくなっていました。どんなモジュールが登録されているか調べる方法がありますか？
A.	<p>ライブラリファイルの内容は、-m オプションで見ることができます。"-m ファイル名" を指定すれば、デフォルト拡張子が (.mp2) のリストファイルが作成されます。 -m オプションで出力される内容では情報不足の場合、-dt オプションを併用してより詳細な情報を得ることができます。</p>
例	<pre> flibs libfile -m libdoc リストファイルlibdoc.mp2 出力 flibs libfile -m libdoc -dt a,s 詳細情報を含んだリストファイル libdoc.mp2 出力 </pre>

Q.	ライブラリファイルの内容を確認したいのですが、ファイルにとるほどではないので、画面上に出力できますか？
A.	-m オプションのファイル名指定のかわりにハイフン (-) を指定すると標準出力への表示になります。
例	<pre>flibs libfile -m - flibs libfile -m - -dt a,s flibs libfile -a file3.obj -m -</pre>

Q.	ライブラリファイルの内容をリスト出力したところ、文字数の多いシンボル名の表示が 2 行に渡ってしまい見にくいのですが？
A.	-pw オプションで、1 行に表示する桁数を増やしてください。デフォルトは 80 桁なので、2 行目に表示された部分が 4 文字ならその倍の 8 を加えた 88 桁以上を指定すれば 1 行に収まります。
例	<pre>flibs libfile -m libdoc -pw 90</pre>

Q.	ライブラリファイル中のモジュールを新しいモジュールと入替えたところ幾つか間違えて登録してしまったことに気づきました。置換前のライブラリファイルもバックアップをとっておらず、復旧するのに苦労しました。
A.	ライブラリアンでは、-b オプションの指定により 1 世代分のバックアップファイル (拡張子 ".bak") を作成できます。 ライブラリファイルの編集を行う場合は、予め元のライブラリのバックアップをとっておくことが望ましいですが、必要に応じて -b オプションを指定してください。
例	<pre>flibs libfile -r file1,file2 -d mod4 -b</pre>

第 4 部 オブジェクト形式コンバータ編

この部では、オブジェクト形式コンバータの種類、オプション一覧、機能説明、オブジェクト形式の変換について説明します。

- 第 13 章 オブジェクト形式 コンバータの仕様
- 第 14 章 オブジェクト形式 コンバータの共通オプション
- 第 15 章 ロードモジュールコンバータ (f2ms, f2hs, f2is, f2es)
- 第 16 章 フォーマットアジャスタ (m2ms, h2hs)
- 第 17 章 バイナリコンバータ (m2bs, h2bs)
- 第 18 章 その他のコンバータ
- 第 19 章 オブジェクト形式コンバータの制限事項および Q&A

第13章

オブジェクト形式 コンバータの仕様

この章では、オブジェクト形式コンバータの概要と種類を説明します。

オブジェクト形式コンバータはオブジェクト形式を変換するツールです。

- 13.1 オブジェクト形式コンバータの概要
- 13.2 オブジェクト形式コンバータの種類
- 13.3 オブジェクト形式コンバータの実行

13.1 オブジェクト形式コンバータの概要

オブジェクト形式コンバータは以下のファイル形式を処理の対象としています。

- リンカ出力の絶対形式ロードモジュール
- S フォーマット
- HEX フォーマット
- バイナリデータファイル

■ オブジェクト形式コンバータの概要

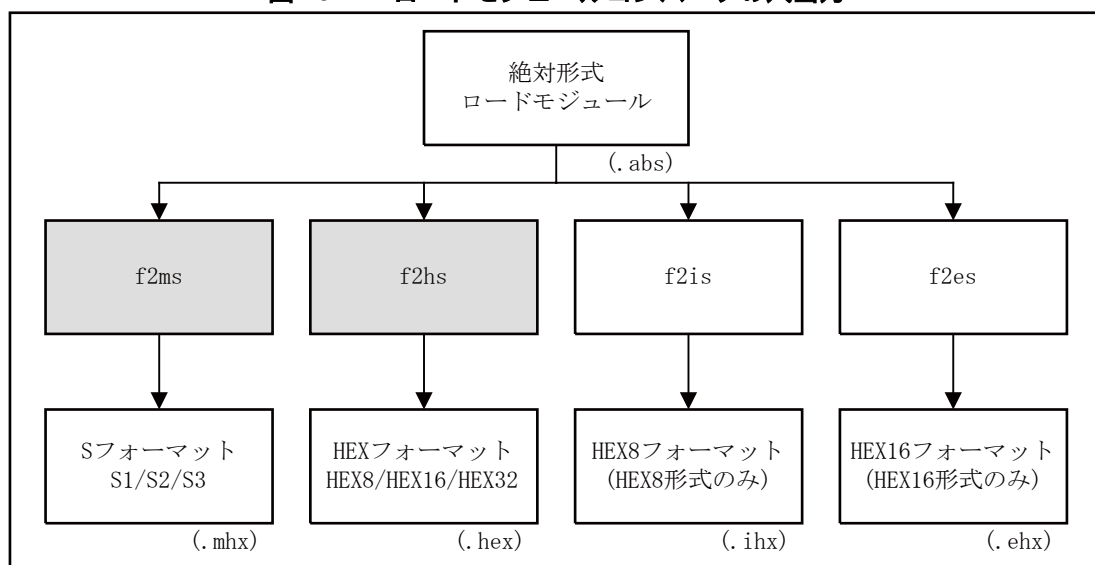
オブジェクト形式コンバータには、ロードモジュールコンバータ、アジャスタ（整形ツール）、バイナリコンバータ、コンバータの 4 種類があります。

● ロードモジュールコンバータ

ロードモジュールコンバータはリンカ出力の絶対形式ロードモジュールを汎用フォーマットに変換する際に用います。

図 13.1-1 に、ロードモジュールコンバータの入出力を示します。

図 13.1-1 ロードモジュールコンバータの入出力

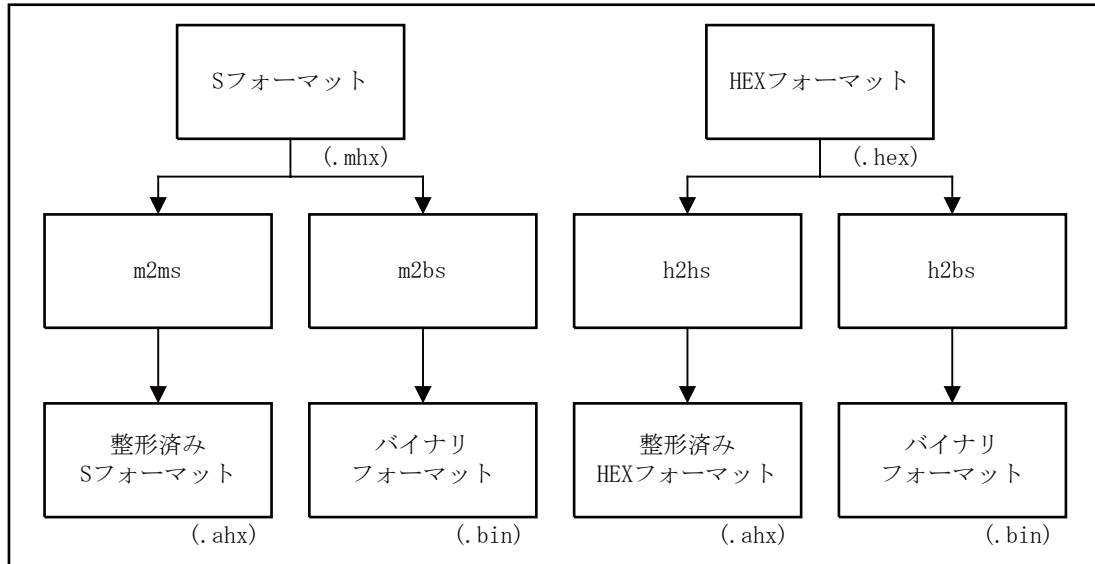


● アジャスタ, バイナリコンバータ

アジャスタは S フォーマットや HEX フォーマットの整形をする際に, バイナリコンバータは S フォーマットや HEX フォーマットをバイナリに変換する際に用います。

図 13.1-2 に, アジャスタ, バイナリコンバータの入出力を示します。

図 13.1-2 アジャスタ, バイナリコンバータの入出力

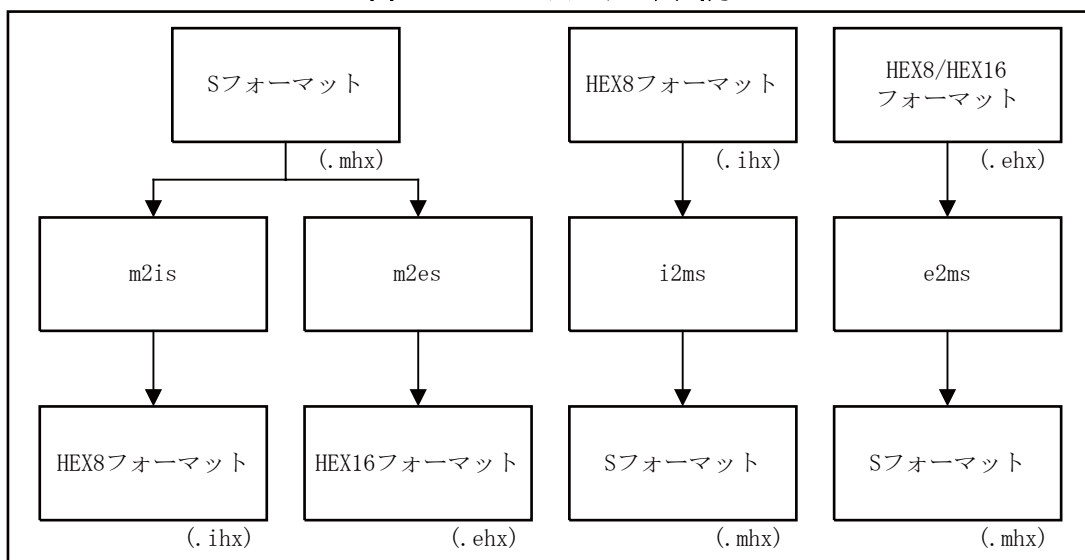


● コンバータ

コンバータは S フォーマットと HEX8/HEX16 フォーマットの相互変換する際に用います。

図 13.1-3 に, コンバータの入出力を示します。

図 13.1-3 コンバータの入出力



13.2 オブジェクト形式コンバータの種類

オブジェクト形式コンバータの各コマンド名は、x2ys のように命名されており、2 の前の x が入力ファイルのオブジェクト形式を、後の y が出力ファイルのオブジェクト形式を示しています。

x と y の部分のアルファベット 1 文字は、次の形式を示しています。

- f: リンカ出力の絶対形式ロードモジュール
- m: S フォーマット
- h: HEX フォーマット (HEX8/HEX16/HEX32)
- b: バイナリデータ形式
- i: HEX8 フォーマット (HEX8)
- e: HEX16 フォーマット (HEX16)

■ ロードモジュールコンバータの種類

オブジェクト形式の変換は、表 13.2-1 のコマンドで行います。

S フォーマットへの変換には f2ms を用います。

また、HEX フォーマットへの変換には f2hs を用います。

f2is を用いれば HEX8 フォーマットへ、f2es を用いれば HEX16 フォーマットへ変換可能ですが、HEX8/HEX16/HEX32 に対応した f2hs を用いることを推奨します。

表 13.2-1 ロードモジュールコンバータのコンバート内容

コマンド名	コンバート内容	
f2ms	絶対形式ロードモジュール	S フォーマット
f2hs	絶対形式ロードモジュール	HEX8/HEX16/HEX32 フォーマット
f2is	絶対形式ロードモジュール	HEX8 フォーマット
f2es	絶対形式ロードモジュール	HEX16 フォーマット

■ フォーマットアジャスタ

フォーマットアジャスタは、S フォーマットのオブジェクトファイルや HEX フォーマットのオブジェクトファイルを整形します。詳しくは、「第 16 章 フォーマットアジャスタ (m2ms, h2hs)」を参照してください。

■ バイナリコンバータ

S フォーマットや HEX フォーマットのオブジェクトファイルをバイナリデータ (メモリーイメージ) へ変換しファイルへ出力します。詳しくは「第 17 章 バイナリコンバータ (m2bs, h2bs)」を参照してください。

■ その他のコンバータの種類

オブジェクト形式の変換は、表 13.2-2 のコマンドで行います。

表 13.2-2 その他のコンバータのコンバート内容

コマンド名	コンバート内容	
m2is	S フォーマット	HEX8 フォーマット
m2es	S フォーマット	HEX16 フォーマット
i2ms	HEX8 フォーマット	S フォーマット
e2ms	HEX16 フォーマット	S フォーマット

13.3 オブジェクト形式コンバータの実行

オブジェクト形式コンバータの実行は、コマンド名の後に、入力ファイル名を指定するだけで実行できます。

■ オブジェクト形式コンバータのコマンド実行

各コマンドとも、コマンド名の後に、入力ファイル名を指定するだけで実行できます。

<code>x2ys <入力ファイル名> [オプション]</code>

指定された<入力ファイル名>を x 形式として処理し、y 形式ファイルを作成します。各オブジェクト形式をファイル名から識別できるように、オブジェクト形式コンバータでは次に示すデフォルト拡張子を使用します。

絶対形式ロードモジュール	: .abs
S フォーマット	: .mhx, .ahx
HEX8/HEX16/HEX32	: .hex, .aix
バイナリデータ形式	: .bin
HEX8	: .ihx
HEX16	: .ehx

バイナリコンバータおよびアジャスタの実行の際は、必ず `-ran` オプションが必要です。詳しくは「17.3.1 出力範囲指定 (-ran)」を参照してください。

【例】

```
f2ms sample
```

リンカ出力の絶対形式ロードモジュール `sample.abs` を入力して、S フォーマットの `sample.mhx` ファイルを出力します。

第14章

オブジェクト形式 コンバータの共通オプション

この章では、オブジェクト形式コンバータの各共通オプションについて詳しく説明します。

14.1 オブジェクト形式コンバータのオプション一覧

14.2 出力ファイル名の変更 (-o)

14.3 パディング (-p)

14.1 オブジェクト形式コンバータのオプション一覧

オブジェクト形式コンバータは、コマンド名の後に入力ファイル名を指定するだけで実行できますが、それに加えて幾つかのオプションが使用できます。

■ オブジェクト形式コンバータの共通オプションの種類

オブジェクト形式コンバータの各コマンドでは、以下のオプションが共通で使用できます。

表 14.1-1 にオブジェクト形式コンバータの共通オプション一覧を示します。

表 14.1-1 オブジェクト形式コンバータの共通オプション一覧

機 能	オプション	備 考
出力ファイル名の変更	-o	
パディングデータ指定	-p	
デフォルトオプションファイル読み込み抑止指定	-Xdof	* 共通オプション
オプションファイル読み込み指定	-f	* 共通オプション
ヘルプメッセージ表示指定	-help	* 共通オプション
版数 / メッセージ出力指定	-V	* 共通オプション
版数 / メッセージ出力抑止	-XV	* 共通オプション
終了メッセージ表示指定	-cmsg	* 共通オプション
終了メッセージ表示抑止指定	-Xcmsg	* 共通オプション
ワーニング発生時の終了コードを 1 にする指定	-cwno	* 共通オプション
ワーニング発生時の終了コードを 0 にする指定	-Xcwno	* 共通オプション

オプションに関する簡単な説明は、コマンド名のみの入力、もしくは -help オプションで表示できます。

```
x2ys
x2ys -help
```

14.2 出力ファイル名の変更 (-o)

コンバート後の出力ファイルの作成ディレクトリおよびファイル名をデフォルトから変更します。

■ 出力ファイル名の変更 (-o)

【記述形式】

-o <オブジェクトファイル名>

【パラメータ】

<オブジェクトファイル名>

出力ファイルの名前

【説明】

コンバート後の出力ファイル名を変更したいときに指定します。

パス名付きで指定することで、出力先ディレクトリも変更できます。

本オプションを省略した場合は、入力ファイル名の拡張子を変換後のフォーマットのデフォルト拡張子に直したものを出力ファイル名とします。

<オブジェクトファイル名> 指定で拡張子を省略した場合も、デフォルト拡張子を付加します。

各フォーマットのデフォルト拡張子は、以下の 6 つです。

絶対形式ロードモジュール	: .abs
S フォーマット	: .mhx .ahx
HEX8/HEX16/HEX32	: .hex .aix
バイナリデータファイル	: .bin
HEX8	: .ihx
HEX16	: .ehx

【例 1】

f2ms ccp903 (-o オプションを使用しない例)

絶対形式ロードモジュール ccp903.abs を入力し、S フォーマットの ccp903.mhx を出力します。上記と同等の指定例を 4 つ示します。

```
f2ms ccp903.abs -o ccp903.mhx
f2ms ccp903.abs -o ccp903
f2ms ccp903      -o ccp903.mhx
f2ms ccp903      -o ccp903
```

【例 2】

f2ms ccp903 -o ccp903.hex

出力ファイル名を、ccp903.hex に変更します。

【例 3】

```
f2ms ccp903 -o ..\hex\ccp903m.hex
```

出力先ディレクトリを、カレントから ..\hex に変更し、出力ファイル名を、ccp903m.hex に変更します。

< 注意事項 >

バイナリコンバータで、-sp オプションを指定した場合は、<オブジェクトファイル名> の評価が異なります。<オブジェクトファイル名> の指定には、拡張子は含まれないものとし、<オブジェクトファイル名> で指定したファイル名に無条件に拡張子を付加します。

例えば、オブジェクトファイル名に "binary.bin" と指定されていれば出力ファイル名は "binary.bin.b01", "binary.bin.b02", ..., "binary.bin.bxx" となります。

14.3 パディング (-p)

任意のアドレス範囲を、指定した値のデータで埋めます。

ロードモジュールコンバータ (f2ms, f2hs) の整形オプション指定時、バイナリコンバータ、アジャスタでは、ファイルにデータの存在しない箇所が指定した値のデータで埋まります。

■ パディング (-p)

【記述形式】

```
-p <値> , [ <開始アドレス> , <終了アドレス> ]
```

【パラメータ】

<値>

1 バイトのデータ

<開始アドレス>

<値> を設定する開始アドレス。

* f2ms, f2hs の整形指定 (-adjust) 時、バイナリコンバータ、アジャスタでは指定しません。

<終了アドレス>

<値> を設定する終了アドレス。

* f2ms, f2hs の整形指定 (-adjust) 時、バイナリコンバータ、アジャスタでは指定しません。

【説明】

指定したアドレス範囲を、指定した値のデータで埋めます。

ロードモジュールコンバータ (f2ms, f2hs) で整形指定 (-adjust) 時やバイナリコンバータおよびアジャスタでは <値> の設定のみを行います。

ロードモジュールコンバータ (f2ms, f2hs) で整形指定 (-adjust) 時やバイナリコンバータおよびアジャスタでは、ファイルにデータの存在しない箇所が指定した値のデータで埋まります。

【例 1】

```
f2ms ccp903 -p 0xEF,0x1FE4,0x1FFF
```

絶対形式ロードモジュールから、S フォーマットにコンバートします。

その際、0x1FE4 番地から 0x1FFF 番地までを、0xEF というデータで S フォーマットファイルの最後尾に追加作成します。

【例 2】

```
f2ms ccp903 -p 0xEF,0x1FE4,0x1FFF -adjust
```

整形指定 (-adjust) 時に、パディングオプション (-p) で開始 / 終了アドレスが指定されているのでエラーになります。

```
f2ms ccp903 -p 0xEF -adjust
```

絶対形式ロードモジュールから、S フォーマットにコンバートします。

その際、データが存在しない箇所を、0xEF というデータで埋めます。

第 14 章 オブジェクト形式 コンバータの共通オプション

【例 3】

```
m2bs ccp903 -ran 0x0,0x1FFF -p 0xEF
```

S フォーマットから , 0x0 番地から 0x1FFF 番地までの範囲をバイナリイメージにコンバートします。

その際 , S フォーマットにデータが存在しない箇所を , 0xEF というデータで埋めます。

第 15 章

ロードモジュールコンバータ (*f2ms*, *f2hs*, *f2is*, *f2es*)

この章では、ロードモジュールコンバータについて説明します。

- 15.1 ロードモジュールコンバータの概要
- 15.2 ロードモジュールコンバータのオプション一覧
- 15.3 ロードモジュールコンバータのオプション詳細
- 15.4 *f2ms*(絶対形式ロードモジュール S フォーマット変換)
- 15.5 *f2hs*(絶対形式ロードモジュール HEX フォーマット変換)
- 15.6 *f2is*(絶対形式ロードモジュール HEX8 フォーマット変換), *f2es*(絶対形式ロードモジュール HEX16 フォーマット変換)

15.1 ロードモジュールコンバータの概要

ロードモジュールコンバータは、絶対形式ロードモジュールを汎用フォーマットである S フォーマットや HEX フォーマットに変換します。

■ ロードモジュールコンバータの概要

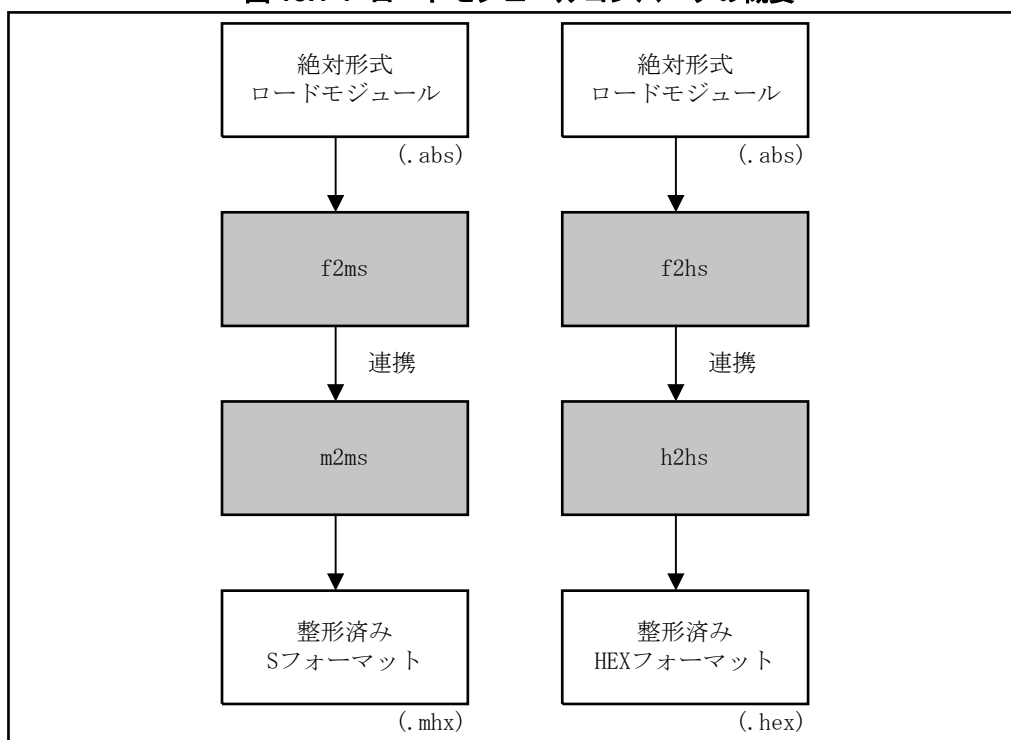
ロードモジュールコンバータは、絶対形式ロードモジュールを汎用フォーマットである S フォーマットや HEX フォーマットに変換します。

S フォーマットに変換するには f2ms を用います。また、HEX フォーマットに変換するには f2hs を用います。

f2is は HEX8 フォーマットに、f2es は HEX16 フォーマットに変換しますが、f2hs を用いれば HEX8/HEX16/HEX32 のすべての HEX フォーマットに変換可能です。

図 15.1-1 に示すように、f2ms, f2hs では整形指定オプション (-adjust) を指定することで、アジャスタと連携動作し出力ファイルの整形を行うことが可能です。

図 15.1-1 ロードモジュールコンバータの概要



15.2 ロードモジュールコンバータのオプション一覧

ここでは、ロードモジュールコンバータのオプション名と、機能概要を一覧で示します。

■ ロードモジュールコンバータのオプション一覧

表 15.2-1 に、ロードモジュールコンバータのオプション一覧を示します。

表 15.2-1 ロードモジュールコンバータのオプション一覧

機 能	オプション	備 考
出力ファイル名の変更	-o	*コンバータ共通オプション
パディングデータ指定	-p	*コンバータ共通オプション
S1 フォーマット出力指定	-S1	f2ms のみ
S2 フォーマット出力指定	-S2	f2ms のみ
S3 フォーマット出力指定	-S3	f2ms のみ
HEX8 フォーマット出力指定	-I16	f2hs のみ
HEX16 フォーマット出力指定	-I20	f2hs のみ
HEX32 フォーマット出力指定	-I32	f2hs のみ
スタートアドレスレコード出力指定	-entry	f2hs のみ
スタートアドレスレコード出力抑止指定	-Xentry	f2hs のみ
整形指定	-adjust	f2ms, f2hs のみ
デフォルトオプションファイル読み込み抑止指定	-Xdof	* 共通オプション
オプションファイル読み込み指定	-f	* 共通オプション
ヘルプメッセージ表示指定	-help	* 共通オプション
版数 / メッセージ出力指定	-V	* 共通オプション
版数 / メッセージ出力抑止	-XV	* 共通オプション
終了メッセージ表示指定	-cmsg	* 共通オプション
終了メッセージ表示抑止指定	-Xcmsg	* 共通オプション
ワーニング発生時の終了コードを 1 にする指定	-cwno	* 共通オプション
ワーニング発生時の終了コードを 0 にする指定	-Xcwno	* 共通オプション

15.3 ロードモジュールコンバータのオプション詳細

ここでは、ロードモジュールコンバータの各オプションについて説明します。
なお、リンケージキットで共通のオプションは「第 3 章 共通オプション」で、コンバータで共通のオプションは「第 14 章 オブジェクト形式 コンバータの共通オプション」で、説明しています。

■ 出力 S フォーマット指定 (-S1/-S2/-S3)

S フォーマットで出力する際に使用するレコードを指定します。詳細は「15.3.1 出力 S フォーマット指定 (-S1/-S2/-S3)」の説明を参照してください。

■ 出力 HEX フォーマット指定 (-I16/-I20/-I32)

HEX フォーマットで出力する際に使用するレコードを指定します。詳細は「15.3.2 出力 HEX フォーマット指定 (-I16/-I20/-I32)」の説明を参照してください。

■ スタートアドレス出力指定 (-entry)

HEX フォーマットで出力する際にスタートセグメントアドレスレコードやスタートリニアアドレスレコードを出力します。詳細は「15.3.3 スタートアドレスレコード出力指定 (-entry)」の説明を参照してください。

■ スタートアドレス出力抑止指定 (-Xentry)

スタートアドレスレコード出力指定を取り消す際に指定します。詳細は「15.3.4 スタートアドレスレコード出力抑止指定 (-Xentry)」の説明を参照してください。

■ 整形指定 (-adjust)

S フォーマットや HEX フォーマットを出力した後にアジャスタの起動を指定します。詳細は「15.3.5 整形指定 (-adjust)」の説明を参照してください。

15.3.1 出力 S フォーマット指定 (-S1/-S2/-S3)

データを出力する際に使用するフォーマットを指定します。

■ 出力 S フォーマット指定 (-S1/-S2/-S3)

【記述形式】

-S1

-S2

-S3

【パラメータ】

なし

【説明】

データ内容を出力する際に使用するフォーマットを指定します。

f2ms は、データ内容を S1 レコード、S2 レコード、S3 レコードのどれか 1 つを用いて出力します。

S1 レコードと S2 レコードの両方を用いて出力を行うようなことはありません。

-S1、-S2、-S3 の指定は後指定が有効になります。また、-S1、-S2、-S3 のオプションが指定されない場合、f2ms はデータのアドレスに合わせ、S1/S2/S3 の混在で出力します。

【注意】

本オプションによる指定と、出力範囲が矛盾する場合、エラーを出力し処理を行います。

本オプションの指定により、出力に使用するターミネータレコード (S9 レコード、S8 レコード、S7 レコード) も変化します (表 15.3-1 参照)。

表 15.3-1 出力 S フォーマット指定一覧

指定	出力可能範囲	ターミネータレコード	備考
-S1	0x00000000 ~ 0x0000FFFF	S9 レコード	16bit address
-S2	0x00000000 ~ 0x00FFFFFF	S8 レコード	24bit address
-S3	0x00000000 ~ 0xFFFFFFFF	S7 レコード	32bit address

15.3.2 出力 HEX フォーマット指定 (-I16/-I20/-I32)

データを出力する際に使用する HEX フォーマットを指定します。

■ 出力 HEX フォーマット指定 (-I16/-I20/-I32)

【記述形式】

-I16

-I20

-I32

【パラメータ】

なし

【説明】

データ内容を出力する際に使用する HEX フォーマットを指定します。

f2hs は、データ内容を HEX8, HEX16, HEX32 のどれか 1 つの形式を用いて出力します。

-I16, -I20, -I32 の指定は後指定が有効になります。また、-I16, -I20, -I32 のオプションが指定されない場合、f2hs はデータのアドレスに合わせ、HEX8/HEX16/HEX32 の混在で出力します。

【注意】

本オプションによる指定と、出力範囲が矛盾する場合、エラーを出力し処理を行いません。

表 15.3-2 出力 HEX フォーマット指定一覧

指定	出力可能範囲	フォーマット	備考
-I16	0x00000000 ~ 0x0000FFFF	HEX8 フォーマット	16bit address
-I20	0x00000000 ~ 0x000FFFFF	HEX16 フォーマット	20bit address
-I32	0x00000000 ~ 0xFFFFFFFF	HEX32 フォーマット	32bit address

15.3.3 スタートアドレスレコード出力指定 (-entry)

データを出力する際にスタートセグメントアドレスレコードまたはスタートリニアアドレスレコードを出力します。

本オプションは f2hs でのみ指定できます。

■ スタートアドレスレコード出力指定 (-entry)

【記述形式】

```
-entry
```

【パラメータ】

なし

【説明】

スタートセグメントアドレスレコードまたは、スタートリニアアドレスレコードの出力を行います。

入力ファイルにスタートアドレス情報がない場合は、警告 (W1504U: 入力ファイルにスタートアドレス情報がありません) を出力します。

スタートアドレスレコードは、出力 HEX フォーマット指定オプション (-I16/-I20/-I32) の指定と、入力データ範囲によって以下の表 15.3-3 のように出力されます。

表 15.3-3 HEX フォーマットスタートアドレスレコードの出力

出力 HEX フォーマット指定	入力データ範囲	出力するスタートアドレスレコード
-I16		警告 (W1503U:-I16 指定時に -entry オプションが指定されました) が出力され、スタートアドレスレコードは出力しません。
-I20	0x0 ~ 0xFFFFF	スタートセグメントアドレスレコード
-I32	0x0 ~ 0xFFFFF	スタートセグメントアドレスレコード
	0x100000 ~ 0xFFFFFFFF	スタートリニアアドレスレコード

出力 HEX フォーマット指定オプションを省略した場合は、HEX32 フォーマット出力指定オプション (-I32) 指定時と同じ処理になります。

【例】

```
f2hs ccp903.abs -entry -I16
```

HEX8 フォーマット出力オプション (-I16) が指定されているため、警告 (W1503U:-I16 指定時に -entry オプションが指定されました) が出力され、スタートアドレスレコードは出力されません。

```
f2hs ccp903.abs -entry -I20
```

スタートセグメントアドレスレコードを出力します。

```
f2hs ccp903.abs -entry -I32
```

入力ファイルのデータ範囲が 0x0 ~ 0xFFFFF のとき、スタートセグメントアドレスレコードを出力します。

入力ファイルのデータ範囲が 0x100000 ~ 0xFFFFFFFF のとき、スタートリニアアドレスレコードを出力します。

15.3.4 スタートアドレスレコード出力抑止指定 (-Xentry)

スタートセグメントアドレスレコードまたはスタートリニアアドレスレコードの出力を抑止します。

本オプションは f2hs でのみ指定できます。

■ スタートアドレスレコード出力抑止指定 (-Xentry)

【記述形式】

-Xentry

【パラメータ】

なし

【説明】

スタートセグメントアドレスレコードまたは、スタートリニアアドレスレコードの出力を抑止します。

本オプションは、スタートアドレスレコード出力指定 (-entry) を取り消すときに使用します。

【例】

```
f2hs -entry ccp903.abs -I20 -Xentry
```

スタートアドレスレコード出力指定 (-entry) を取り消し、スタートアドレスレコードは出力されません。

15.3.5 整形指定 (-adjust)

フォーマット変換後にフォーマットアジャスタを自動的に呼び出し、データ出力形式を整形します。

■ 整形指定 (-adjust)

【記述形式】

-adjust

【パラメータ】

なし

【説明】

ロードモジュールをフォーマット変換後にフォーマットアジャスタを自動で呼び出し整形を行います。

整形対象となる開始 / 終了アドレスは、自動的に設定されます。

本オプションを指定した場合には、パディング (-p) オプションで開始 / 終了アドレスのパラメータ指定を行うとエラーになります。

本オプションを指定した場合には、フォーマットアジャスタのオプションも指定可能となります。

【例】

```
f2ms ccp903 -p 0xEF,0x1FE4,0x1FFF -adjust
```

パディングオプションに開始 / 終了アドレスが指定されているためエラーになります。

```
f2ms ccp903 -p 0xEF -adjust
```

絶対形式ロードモジュールから、整形済み S フォーマットにコンバートします。

その際、データが存在しない箇所を 0xEF というデータで埋めます。

15.4 f2ms(絶対形式ロードモジュール S フォーマット変換)

リンカ出力の絶対形式ロードモジュールを S フォーマットに形式変換します。

0 ~ 0xFFFFFFFF 番地のデータが変換対象になります。

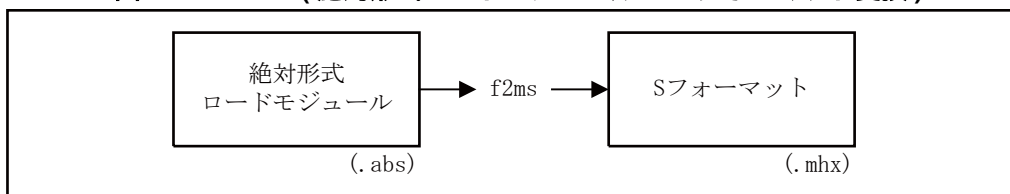
f2ms コマンドは, SOFTUNE V3/V5 系の絶対形式ロードモジュールも処理できます。

■ f2ms(絶対形式ロードモジュール S フォーマット変換)

【機能】

f2ms コマンドは, リンカ出力の絶対形式ロードモジュールからオブジェクトデータ部を読み込み, S フォーマットファイルに形式変換します。

図 15.4-1 f2ms(絶対形式ロードモジュール S フォーマット変換)



【アドレス】

S フォーマットで表現できるアドレスの最大値を示します。

S1 タイプ 0xFFFF

S2 タイプ 0xFFFFFFFF

S3 タイプ 0xFFFFFFFF

リンカ出力の絶対形式ロードモジュールは, 0 ~ 0xFFFFFFFF の範囲のアドレスを表せます。

S フォーマットも同様のアドレス範囲をサポートしていますので, データの欠落なくコンバートを行うことができます。

f2ms は, 入力データの割り付けアドレスに応じて, 以下のレコードを出力します。

0x00000000 ~ 0x0000FFFF : S1

0x00010000 ~ 0x00FFFFFF : S2

0x01000000 ~ 0xFFFFFFFF : S3

15.5 f2hs(絶対形式ロードモジュール HEX フォーマット 変換)

リンカ出力の絶対形式ロードモジュールを HEX フォーマットに形式変換します。

0 ~ 0xFFFFFFFF 番地のデータが変換対象になります。

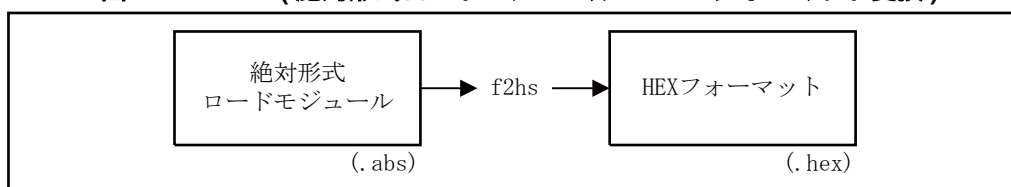
f2hs コマンドは , SOFTUNE V3/V5 系の絶対形式ロードモジュールも処理できます。

■ f2hs(絶対形式ロードモジュール HEX フォーマット変換)

【機能】

f2hs コマンドは、リンカ出力の絶対形式ロードモジュールからオブジェクトデータ部を読み込み、HEX フォーマットファイルに形式変換します。

図 15.5-1 f2hs(絶対形式ロードモジュール HEX フォーマット変換)



【アドレス】

HEX フォーマットで表現できるアドレスの最大値を示します。

HEX8 0xFFFF

HEX16 0xFFFFFFFF

HEX32 0xFFFFFFFF

リンカ出力の絶対形式ロードモジュールは、0 ~ 0xFFFFFFFF の範囲のアドレスを表せます。

HEX フォーマットも同様のアドレス範囲をサポートしていますので、データの欠落なくコンバートを行うことができます。

f2hs は、入力データの割り付けアドレスに応じて、以下の形式で出力します。

0x00000000 ~ 0x0000FFFF : HEX8

0x00010000 ~ 0x000FFFFF : HEX16

0x00100000 ~ 0xFFFFFFFF : HEX32

15.6 f2is(絶対形式ロードモジュール HEX8 フォーマット変換), f2es(絶対形式ロードモジュール HEX16 フォーマット変換)

f2is はリンカ出力の絶対形式ロードモジュールを HEX8 フォーマットに, f2es はリンカ出力の絶対形式ロードモジュールを HEX16 フォーマットに変換します。

f2is では 0 ~ 0xFFFF 番地のデータが変換対象に, f2es では 0 ~ 0xFFFFF 番地のデータが変換対象になります。

f2is, f2es コマンドは, SOFTUNE V3/V5 系の絶対形式ロードモジュールも処理できます。

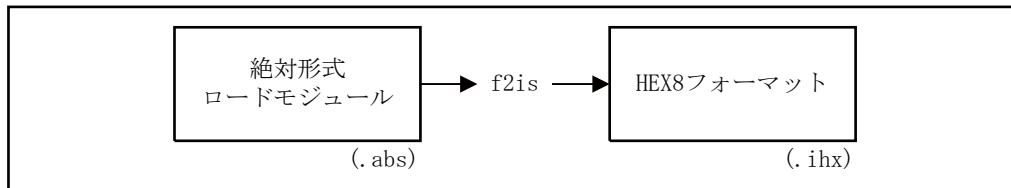
f2is, f2es は, 旧版との互換性維持のためリンケージキットに含まれています。HEX フォーマットへの変換には f2hs を使用されることを推奨いたします。

■ f2is(絶対形式ロードモジュール HEX8 フォーマット変換)

【機能】

f2is コマンドは, リンカ出力の絶対形式ロードモジュールからオブジェクトデータ部を読み込み, HEX8 フォーマットファイルに形式変換します。

図 15.6-1 f2is(絶対形式ロードモジュール HEX8 フォーマット変換)



【説明】

f2is では「第 14 章 オブジェクト形式 コンバータの共通オプション」で示す共通オプションが使用可能です。

【アドレス】

HEX8 フォーマットで表現できるアドレスの最大値は, 0xFFFF です。

< 注意事項 >

リンカ出力の絶対形式ロードモジュールは, 0 ~ 0xFFFFFFFF のアドレスを表せますが, HEX8 フォーマットへコンバートすると 0x10000 以上のアドレスに割り付けられたデータは切捨てられます。

コンバート元のアドレス範囲に気をつけて使用してください。

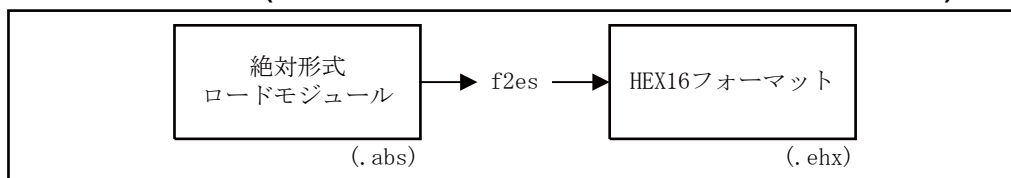
HEX8 フォーマットファイルは, データレコードとトレーラレコードで構成されます。

■ f2es(絶対形式ロードモジュール HEX16 フォーマット変換)

【機能】

f2es コマンドは、リンカ出力の絶対形式ロードモジュールからオブジェクトデータ部を読み込み、HEX16 フォーマットファイルに形式変換します。

図 15.6-2 f2es(絶対形式ロードモジュール HEX16 フォーマット変換)



【説明】

f2es では「第 14 章 オブジェクト形式 コンバータの共通オプション」で示す共通オプションが使用可能です。

【アドレス】

HEX16 フォーマットで表現できるアドレスの最大値は、0xFFFFF です。

< 注意事項 >

リンカ出力の絶対形式ロードモジュールは、0 ~ 0xFFFFFFFFF のアドレスを表せますが、HEX16フォーマットへコンバートすると0x100000以上のアドレスに割り付けられたデータは切捨てられます。

コンバート元のアドレス範囲に気をつけて使用してください。

HEX16 フォーマットでは、0x10000 以上のアドレスを表現するために拡張セグメントアドレスレコードを使用します。

ファイル中の拡張セグメントアドレスレコードは、次の拡張セグメントアドレスレコードが現れるまで有効です。拡張セグメントアドレスレコードなしにデータが現れたときには、拡張セグメントアドレス指定が0であったとしてアドレス計算を行います。

HEX16 フォーマットのファイルの先頭には、スタートアドレスレコードが作成されます。

第16章

フォーマットアジャスタ (*m2ms*, *h2hs*)

この章では、フォーマットアジャスタについて説明します。

- 16.1 フォーマットアジャスタの概要
- 16.2 フォーマットアジャスタのオプション一覧
- 16.3 フォーマットアジャスタのオプション詳細

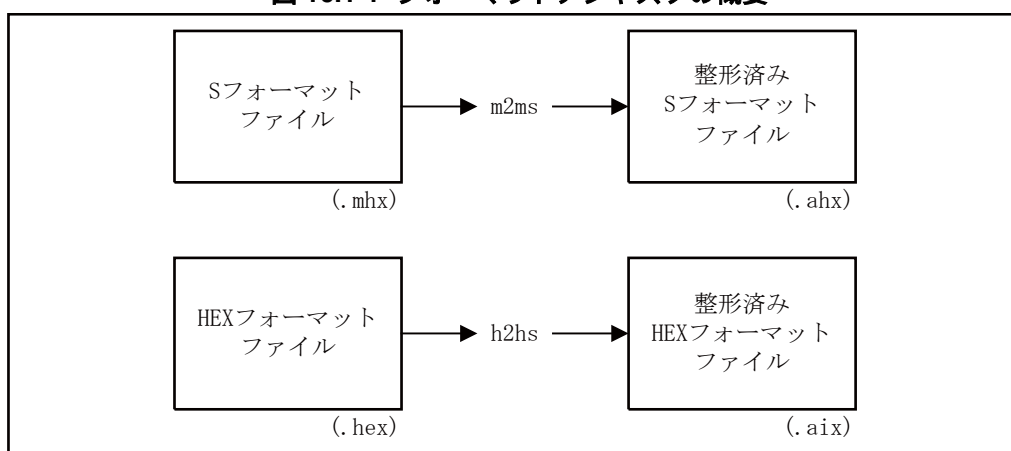
16.1 フォーマットアジャスタの概要

フォーマットアジャスタは、S フォーマットや HEX フォーマットで作成したデータをアドレス昇順に並び替え、1 レコードに含むデータの数で指定した値に揃える働きをします。

■ フォーマットアジャスタの概要

フォーマットアジャスタは、フォーマットのファイル1レコードに含むデータの数で指定の値に揃え、形式を整えます。図 16.1-1 にフォーマットアジャスタの概念図を示します。

図 16.1-1 フォーマットアジャスタの概要



入力するフォーマット中にデータが存在しない箇所は、0xFF (デフォルト値) で埋められます。

出力するファイルはメモリ内容をそのまま S フォーマットや HEX フォーマットに変換したものと同等です。

入力ファイル中にデータが存在しない箇所を特定の値で埋めるには、パディングオプション (-p オプション) を用います。オプションの使用方法については、「14.3 パディング (-p)」をご参照ください。

■ 動作例

本ツールは、以下のように作成したフォーマットファイルの1レコードあたりのデータ長が一定でないような場合に、1レコードに含むデータ長を統一できます。

図 16.1-2 フォーマットアジャスタの動作例



■ フォーマットアジャスタの機能

フォーマットアジャスタには以下の機能があります。

- データをアドレス昇順に並び替えます。
- 指定したアドレス範囲内にデータの欠落がある場合、その部分は起動時に指定したデータを埋め込みます (デフォルト値は 0xff)。
- レコードの開始アドレスは、起動時に指定した出力データ長で整合をとった値を使用します。

出力指定の開始アドレスが、起動時のデータ長の指定値の倍数でない場合

(レコード内データ長が 16 バイトで開始アドレスが 16 の倍数と異なるような場合)

- 出力情報の最初に表れるレコードは、指定開始アドレスからデータ長の指定値で整合を取ったアドレスまでのデータを格納します。
- 2 番目以降のレコード開始アドレスが、指定長で整合を取ったアドレスとなります。
- 入力するフォーマット情報中に、複数のエントリアドレスが存在する場合、最後に表れたエントリアドレスを変換して出力します。それ以外のエントリアドレスは無視されます。
- 変換元データに設定してあるエントリアドレスの値が、変換後のデータのアドレス範囲に含まれない場合、変換後のターミネータレコードには 0 を設定します。

図 16.1-3 にアドレス範囲 0xff0008 ~ 0xff004a を , レコード長 16 で変換した例を示します。

図 16.1-3 フォーマットアジャスタの変換例



16.2 フォーマットアジャスタのオプション一覧

ここでは、フォーマットアジャスタのオプション名と、機能概要を一覧で示します。

■ フォーマットアジャスタのオプション一覧

表 16.2-1 に、フォーマットアジャスタのオプション一覧を示します。

表 16.2-1 フォーマットアジャスタのオプション一覧

機 能	オプション	備 考
出力ファイル名の変更	-o	* コンバータ共通オプション
パディングデータ指定	-p	* コンバータ共通オプション
出力データ長指定	-len	デフォルト 16
出力範囲指定	-ran	必須
S1 フォーマット出力指定	-S1	m2ms のみ
S2 フォーマット出力指定	-S2	m2ms のみ
S3 フォーマット出力指定	-S3	m2ms のみ
HEX8 フォーマット出力指定	-I16	h2hs のみ
HEX16 フォーマット出力指定	-I20	h2hs のみ
HEX32 フォーマット出力指定	-I32	h2hs のみ
開始アドレス変更指定	-ST	
デフォルトオプションファイル読み込み抑止指定	-Xdof	* 共通オプション
オプションファイル読み込み指定	-f	* 共通オプション
ヘルプメッセージ表示指定	-help	* 共通オプション
版数 / メッセージ出力指定	-V	* 共通オプション
版数 / メッセージ出力抑止	-XV	* 共通オプション
終了メッセージ表示指定	-cmsg	* 共通オプション
終了メッセージ表示抑止指定	-Xcmsg	* 共通オプション
ワーニング発生時の終了コードを 1 にする指定	-cwno	* 共通オプション
ワーニング発生時の終了コードを 0 にする指定	-Xcwno	* 共通オプション

16.3 フォーマットアジャスタのオプション詳細

ここでは、フォーマットアジャスタの各オプションについて説明します。

なお、リンケージキットで共通のオプションは「第 3 章 共通オプション」で、コンバータで共通のオプションは「第 14 章 オブジェクト形式 コンバータの共通オプション」で、説明しています。

■ 出力レコード内データ長指定 (-len)

データ長の指定です。詳細は「16.3.1 出力レコード内データ長指定 (-len)」の説明を参照してください。

■ 出力範囲指定 (-ran)

フォーマットの整形を行う範囲の指定です。詳細は「16.3.2 出力範囲指定 (-ran)」の説明を参照してください。

■ 出力 S フォーマット指定 (-S1/-S2/-S3)

m2ms で S フォーマットで出力する際に使用するレコードを指定します。詳細は「16.3.3 出力 S フォーマット指定 (-S1/-S2/-S3)」の説明を参照してください。

■ 出力 HEX フォーマット指定 (-I16/-I20/-I32)

h2hs で HEX フォーマットで出力する際に使用するレコードを指定します。詳細は「16.3.4 出力 HEX フォーマット指定 (-I16/-I20/-I32)」の説明を参照してください。

■ レコード開始アドレス変更指定 (-ST)

S フォーマットで出力する際に使用する S レコードの開始アドレスを変更します。詳細は「16.3.5 開始アドレス変更指定 (-ST)」の説明を参照してください。

16.3.1 出力レコード内データ長指定 (-len)

出力するフォーマットの 1 レコードに出力するデータの数を指定します。

■ 出力レコード内データ長指定 (-len)

【記述形式】

-len <データ長>

【パラメータ】

<データ長>

16/32/64/128 の 4 種類から選択します。

【説明】

フォーマットを整形する際の、1 レコードに出力するデータのバイト数を指定します。

データ長は、16/32/64/128 の 4 種類が指定できます。

当オプションの指定を省略した場合、16 が指定されたものと仮定して処理します。

【注意】

本指定は 1 レコード内に含むデータのバイト数を指定するものであり、レコード長自体を指定するものではありません。

【例】

```
m2ms sfmtfile.mhx -len 32
```

sfmtfile.mhx を整形し、1 レコードあたり 32 バイトのデータを出力します。

```
m2ms sfmtfile.mhx          (-len 指定を省略した例)
```

sfmtfile.mhx を整形し、1 レコードあたり 16 バイトのデータを出力します。

```
m2ms sfmtfile.mhx -len 96
```

指定可能範囲外のデータ長が指示されたため、エラーとなります。

```
m2ms sfmtfile.mhx -len      (パラメータをすべて省略した例)
```

データ長の指定が省略されているのでエラーになります。

16.3.2 出力範囲指定 (-ran)

整形する範囲をアドレスで指定します。

■ 出力範囲指定 (-ran)

【記述形式】

`-ran <開始アドレス> [, <終了アドレス>]`

【パラメータ】

<開始アドレス>

開始アドレス

<終了アドレス>

終了アドレス

【説明】

整形する範囲をアドレスで指定します。

コンバート実行時は、当オプションの指定が必ず必要です。

開始アドレス、終了アドレスは、0x0 ~ 0xffffffff の範囲で指定します。

終了アドレスの指定は省略できます。終了アドレスを省略した場合、開始アドレスから 64K バイト分を整形します。

変換サイズが 2G バイト以上になるような値は指定できません。

【例】

`m2ms sfmtfile.mhx` (-ran オプションを使用しない例)

出力範囲が指定されていないのでエラーになります。

`m2ms sfmtfile.mhx -ran 0xD000,0xFFFF`

`sfmtfile.mhx` 中の 0xD000 ~ 0xFFFF 番地までのデータを整形します。

`m2ms sfmtfile.mhx -ran 0xD000` (終了アドレスを省略した例)

`sfmfile.mhx` 中の 0xD000 番地から 64K バイト分 (0x0D000 ~ 0x1CFFF) のデータを整形します。

`m2ms sfmtfile.mhx -ran 0xFFFF,0xD000`

開始アドレスより低位なアドレスが終了アドレスとして指定されているためエラーになります。

`m2ms sfmtfile.mhx -ran` (パラメータをすべて省略した例)

開始アドレスが省略されているのでエラーになります。

16.3.3 出力 S フォーマット指定 (-S1/-S2/-S3)

データを出力する際に使用するフォーマットを指定します。

S フォーマットアジャスタ (m2ms) のオプションです。

■ 出力 S フォーマット指定 (-S1/-S2/-S3)

【記述形式】

-S1

-S2

-S3

【パラメータ】

なし

【説明】

データ内容を出力する際に使用するレコードを指定します。

S フォーマットアジャスタは、データ内容を S1 レコード、S2 レコード、S3 レコードのどれか 1 つを用いて出力します。

S1 レコードと S2 レコードの両方を用いて出力を行うようなことはありません。

-S1、-S2、-S3 の指定は後指定が有効になります。また、-S1、-S2、-S3 のオプションが指定されない場合、S フォーマットアジャスタはデータ内容を S3 レコードで出力します。

【注意】

本オプションによる指定と、出力範囲が矛盾する場合、S フォーマットアジャスタはエラーを出力し処理を行いません。

本オプションの指定により、出力に使用するターミネータレコード (S9 レコード、S8 レコード、S7 レコード) も変化します (表 16.3-1 参照)。

表 16.3-1 出力レコード指定一覧

指定	出力可能範囲	ターミネータレコード	備考
-S1	0x0000 ~ 0xFFFF	S9 レコード	
-S2	0x000000 ~ 0xFFFFFFFF	S8 レコード	
-S3	0x00000000 ~ 0xFFFFFFFF	S7 レコード	(デフォルト)

【例】

```
m2ms sfmtfile.mhx -ran 0xD000,0x10000 -S1
```

出力範囲が, S1 レコードで表現できない 0x10000 までのためエラーになります。

```
m2ms sfmtfile.mhx -ran 0xE000,0xFFFF -S1
```

0xE000 ~ 0xFFFF の内容が S1 レコードを用いて整形出力されます。

```
m2ms sfmtfile.mhx -ran 0xE000,0xFFFF -S2
```

0xE000 ~ 0xFFFF の内容が S2 レコードを用いて整形出力されます。

```
m2ms sfmtfile.mhx -ran 0xE000,0xFFFF -S3
```

0xE000 ~ 0xFFFF の内容が S3 レコードを用いて整形出力されます。

16.3.4 出力 HEX フォーマット指定 (-I16/-I20/-I32)

データを出力する際に使用する HEX フォーマットを指定します。
 HEX フォーマットアジャスタ (h2hs) のオプションです。

■ 出力 HEX フォーマット指定 (-I16/-I20/-I32)

【記述形式】

-I16

-I20

-I32

【パラメータ】

なし

【説明】

データ内容を出力する際に使用するレコードを指定します。

HEX フォーマットアジャスタは、データ内容を HEX8 フォーマット、HEX16 フォーマット、HEX32 フォーマットのどれか 1 つを用いて出力します。

-I16、-I20、-I32 の指定は後指定が有効になります。

また、-I16、-I20、-I32 のオプションが指定されない場合、HEX フォーマットアジャスタはデータ内容を HEX32 フォーマットで出力します。

【注意】

本オプションによる指定と、出力範囲が矛盾する場合、HEX フォーマットアジャスタはエラーを出力し処理を行いません。

【例】

```
h2hs hfmtfile.hex -ran 0xD000,0x10000 -I16
```

出力範囲が、HEX8 で表現できない 0x10000 までのためエラーになります。

```
h2hs hfmtfile.hex -ran 0xE000,0xFFFF -I16
```

0xE000 ~ 0xFFFF の内容が HEX8 フォーマットを用いて整形出力されます。

```
h2hs hfmtfile.hex -ran 0xE000,0xFFFF -I20
```

0xE000 ~ 0xFFFF の内容が HEX16 フォーマットを用いて整形出力されます。

```
h2hs hfmtfile.hex -ran 0xE000,0xFFFF -I32
```

0xE000 ~ 0xFFFF の内容が HEX32 フォーマットを用いて整形出力されます。

16.3.5 開始アドレス変更指定 (-ST)

データ内容を出力する際に使用する開始アドレスを指定します。
データのアドレスを変更する場合に使用します。

■ 開始アドレス変更指定 (-ST)

【記述形式】

-ST <開始アドレス>

【パラメータ】

<開始アドレス>

開始アドレス

【説明】

データ内容を出力する際に使用する開始アドレスを指定します。

フォーマットアジャスタは、通常出力範囲指定 (-ran) で指定された開始アドレスを用いて開始アドレスの決定を行います。

本オプションが指定されると、出力する際に開始アドレスの変更を行います。

【例】

```
m2ms sfmtfile.mhx -ran 0xD000,0xFFFF -ST 0x0000
```

sfmtfile.mhx 中の 0xD000 ~ 0xFFFF 番地までのデータを整形し、0x0000 番地からのデータとして出力します。

```
m2ms sfmtfile.mhx -ran 0xD000,0xFFFF -ST 0x10000
```

sfmtfile.mhx 中の 0xD000 ~ 0xFFFF 番地までのデータを整形し、0x10000 番地からのデータとして出力します。

```
m2ms sfmtfile.mhx -ran 0xD000,0xFFFF -ST (パラメータを省略した例)
```

開始アドレスが省略されているのでエラーになります。

第17章

バイナリコンバータ (*m2bs*, *h2bs*)

この章では、バイナリコンバータの変換形式を説明します。

17.1 バイナリコンバータの概要

17.2 バイナリコンバータのオプション一覧

17.3 バイナリコンバータのオプション詳細

17.1 バイナリコンバータの概要

バイナリコンバータは、S フォーマットまたは HEX フォーマットで出力されたファイルをバイナリデータファイルへ変換するコンバータです。

単にバイナリデータへ変換するだけでなく複数のファイルへ分割して出力するスプリットモードなどをサポートしています。

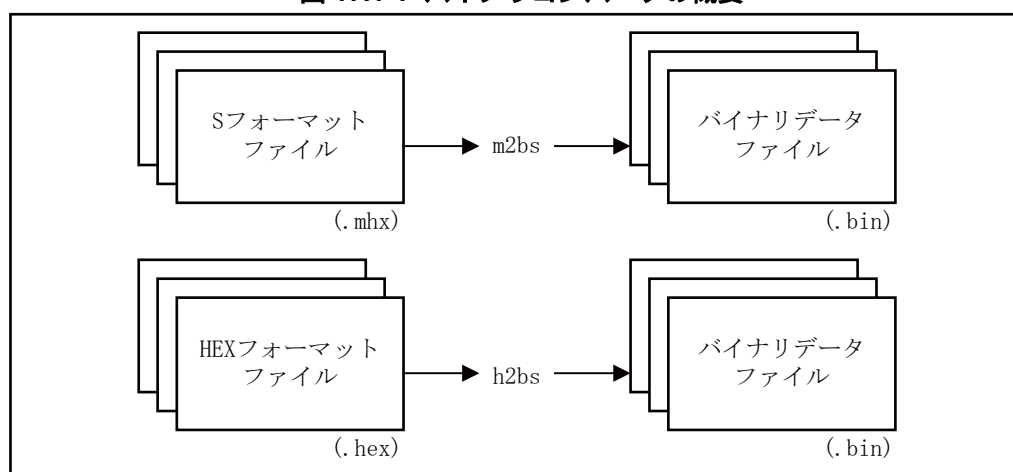
■ バイナリコンバータの概要

バイナリコンバータは、リンケージキットで作成された S フォーマットや HEX フォーマットのオブジェクトファイルをバイナリデータ (メモリイメージ) へ変換しファイルへ出力します。

S フォーマットをバイナリデータに変換するには m2bs を用い、HEX フォーマットをバイナリデータに変換するには h2bs を用います。

入力ファイル (S フォーマットファイルまたは HEX フォーマット) は複数個指定することが可能です。また、変換後のバイナリデータを、指定バイトごとに複数のファイルに分割して出力すること (以降、スプリットモードとよびます) も可能です。

図 17.1-1 バイナリコンバータの概要



< 注意事項 >

入力ファイル中にデータが存在しない箇所は、0xFF (デフォルト値) で埋められ、出力されるバイナリデータファイルはメモリ内容をそのままファイルへ出力したものと同等です。

入力ファイル中にデータが存在しない箇所を特定の値で埋めるには、パディングオプション (-p) を用います。オプションの使用方法については、「14.3 パディング (-p)」を参照してください。

出力バイナリファイルのデフォルト拡張子は、.bin です。なお、スプリットモード使用時は無条件に拡張子 .bxx (xx は 2 桁の番号 (01 ~ 16)) が付加されます。

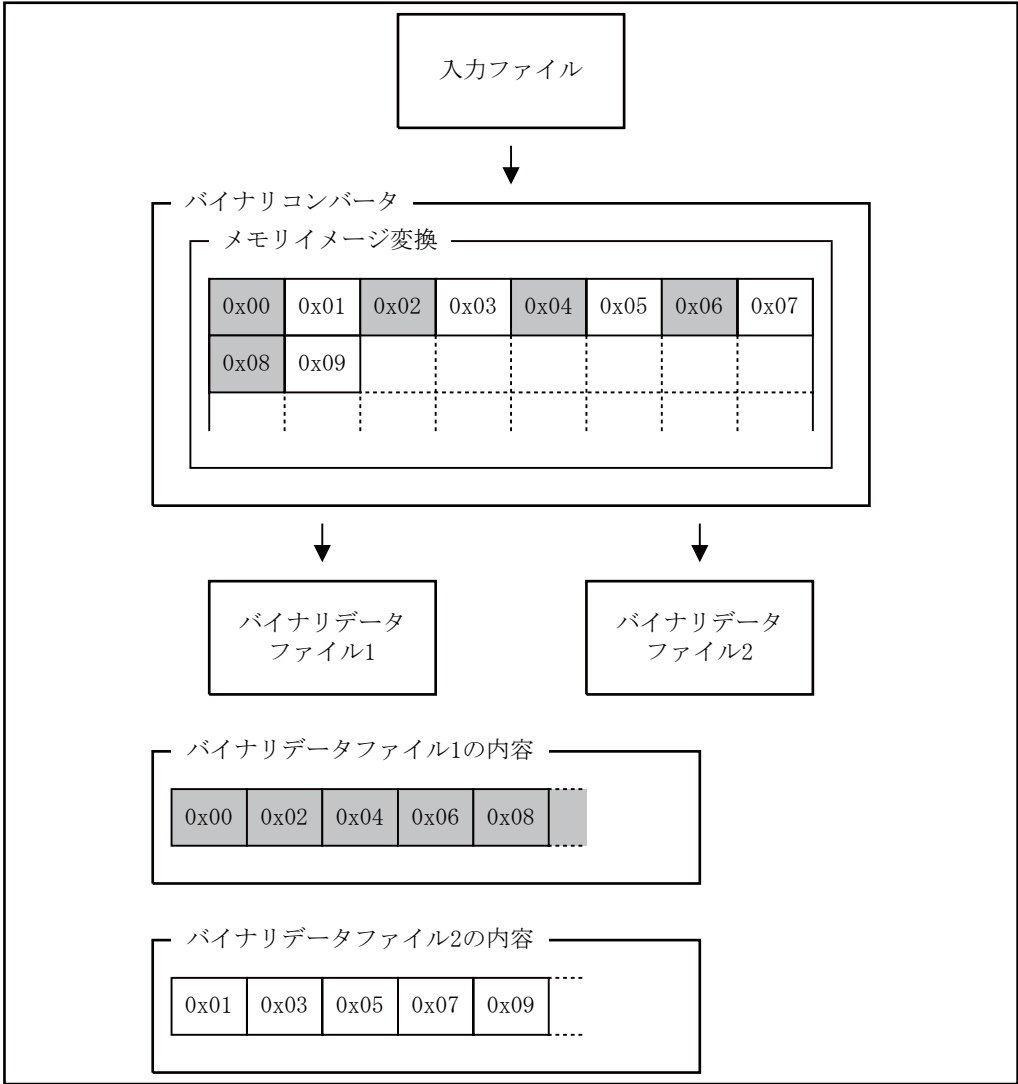
■ スプリットモードの概要

スプリットモードとは、バイナリコンバータにより変換されたメモリイメージを指定バイトごとに複数のバイナリデータファイルへ出力することを示します。

図 17.1-2 にスプリットモードの概略図を示します。図 17.1-2 では 2 つのファイルに 1 バイトごと交互に出力していますが、スプリットモードでは、最大 16 個のファイルに指定バイトごとに交互に出力することが可能です。

スプリットモードを使用するには、-sp オプションを用いて行います。-sp オプションの使用方法については、「17.3.2 スプリットモード指定 (-sp)」を参照してください。

図 17.1-2 スプリットモードの概要



17.2 バイナリコンバータのオプション一覧

ここでは、バイナリコンバータのオプション名と、機能概要を一覧で示します。

■ バイナリコンバータのオプション一覧

表 17.2-1 に、バイナリコンバータのオプション一覧を示します。

表 17.2-1 バイナリコンバータのオプション一覧

機 能	オプション	備 考
出力ファイル名の変更	-o	* コンバータ共通オプション
パディングデータ指定	-p	* コンバータ共通オプション
出力範囲指定	-ran	必須
スプリットモード指定	-sp	
スプリットモード抑止指定	-Xsp	
マップリストファイル作成指定	-m	
マップリストファイル作成抑止指定	-Xm	
デフォルトオプションファイル読み込み抑止指定	-Xdof	* 共通オプション
オプションファイル読み込み指定	-f	* 共通オプション
ヘルプメッセージ表示指定	-help	* 共通オプション
版数 / メッセージ出力指定	-V	* 共通オプション
版数 / メッセージ出力抑止	-XV	* 共通オプション
終了メッセージ表示指定	-cmsg	* 共通オプション
終了メッセージ表示抑止指定	-Xcmsg	* 共通オプション
ワーニング発生時の終了コードを 1 にする指定	-cwno	* 共通オプション
ワーニング発生時の終了コードを 0 にする指定	-Xcwno	* 共通オプション

17.3 バイナリコンバータのオプション詳細

ここでは、バイナリコンバータの各オプションについて説明します。

なお、リンケージキットで共通のオプションは「第 3 章 共通オプション」で、コンバータで共通のオプションは「第 14 章 オブジェクト形式 コンバータの共通オプション」で、説明しています。

■ 出力範囲指定 (-ran)

SフォーマットやHEXフォーマットからバイナリイメージへ変換する範囲の指定です。
詳細は「17.3.1 出力範囲指定 (-ran)」の説明を参照してください。

■ スプリットモード指定 (-sp)

バイナリイメージをスプリットモードで出力する場合に指定します。詳細は「17.3.2 スプリットモード指定 (-sp)」の説明を参照してください。

■ スプリットモード抑止指定 (-Xsp)

スプリットモード指定 (-sp) を無効にするための指定です。詳細は「17.3.3 スプリットモード抑止指定 (-Xsp)」の説明を参照してください。

■ マップリストファイルの作成指定 (-m)

コンバート時の情報をマップリストファイルに出力する場合に指定します。詳細は「17.3.4 マップリストファイルの作成指定 (-m)」の説明を参照してください。

■ マップリストファイルの作成抑止指定 (-Xm)

マップリストファイルの作成指定 (-m) を無効にするための指定です。詳細は「17.3.5 マップリストファイルの作成抑止指定 (-Xm)」の説明を参照してください。

17.3.1 出力範囲指定 (-ran)

バイナリイメージへ変換する範囲をアドレスで指定します。
バイナリコンバータは本オプションの指定が必ず必要です。

■ 出力範囲指定 (-ran)

【記述形式】

```
-ran <開始アドレス> [, <終了アドレス> ]
```

【パラメータ】

<開始アドレス>

開始アドレス

<終了アドレス>

終了アドレス

【説明】

バイナリイメージへ変換する範囲をアドレスで指定します。

コンバート実行時は、本オプションの指定が必ず必要です。

開始アドレス、終了アドレスは、0x0 ~ 0xFFFFFFFF の範囲で指定します。

終了アドレスの指定は省略できます。終了アドレスを省略した場合、開始アドレスから 64K バイト分がバイナリに変換されます。

変換サイズが 2G バイト以上になるような値は指定できません。

【例 1】

```
m2bs sfmtfile.mhx (-ran オプションを使用しない例)
```

出力範囲が指定されていないのでエラーになります。

【例 2】

```
m2bs sfmtfile.mhx -ran 0xD000,0xFFFF
```

sfmtfile.mhx 中の 0xD000 ~ 0xFFFF 番地までのデータを抜き出してバイナリイメージファイルにします。

【例 3】

```
m2bs sfmtfile.mhx -ran 0xD000 (終了アドレスを省略した例)
```

sfmtfile.mhx 中の 0xD000 番地から 64K バイト分 (0xD000 ~ 0xD0FFF) のデータを抜き出してバイナリイメージファイルにします。

【例 4】

```
m2bs sfmtfile.mhx -ran 0xFFFF,0xD000
```

開始アドレスより低位なアドレスが終了アドレスとして指定されているためエラーになります。

【例 5】

```
m2bs sfmtfile.mhx -ran (パラメータをすべて省略した例)
```

開始アドレスが省略されているのでエラーになります。

17.3.2 スプリットモード指定 (-sp)

バイナリイメージをスプリットモードで出力する場合に指定します。

■ スプリットモード指定 (-sp)

【記述形式】

```
-sp <出力ファイル数> [, <バイト数> ]
```

【パラメータ】

<出力ファイル数>

出力先ファイルを幾つに分割するかを指定します。2 ~ 16 の間で指定が可能です。

<バイト数>

スプリットの単位をバイト数で指定します。0x01 ~ 0xFFFFFFFF の間で指定が可能です。

【説明】

本オプションは、複数のファイルに交互にデータを出力するオプションです。例えば、32 ビット単位のデータをデータ幅が 16 ビットの ROM2 個を用いて構成するので、2 つのファイルに 2 バイトごと交互にデータを出力したい場合などに用います。

<バイト数> には、出力ファイルのうち 1 つでも 0 バイト出力になるような値は指定できません。

<バイト数> は省略が可能です。省略された場合、スプリット単位はデフォルトで 1 バイトに設定されます。

本オプションを使用すれば、64K バイトの範囲を 32K バイトごとの 2 つのバイナリイメージファイルにすることも可能です。ただし、本オプションは、あくまで複数のファイルに、データを交互に出力するためのものです。64K バイトの範囲を 32K バイトごとの 2 つに分割するようにパラメータ指定を行うと、最後の 1K バイトは前半のファイルに出力されます。

本オプションを指定すると出力ファイルに拡張子として ".bxx" (xx は 2 桁の番号 01 ~ 16) が無条件に付加されます。

【例 1】

```
m2bs sfmtfile.mhx (-sp オプションを使用しない例)
```

sfmtfile.bin にバイナリイメージが出力されます。

【例 2】

```
m2bs sfmtfile.mhx -sp 2
```

sfmtfile.b01 と sfmtfile.b02 に 1 バイトごと交互にバイナリイメージが出力されます。

【例 3】

```
m2bs sfmtfile.mhx -sp 2,2
```

sfmtfile.b01 と sfmtfile.b02 に 2 バイトごと交互にバイナリイメージが出力されます。

17.3.3 スプリットモード抑止指定 (-Xsp)

-Xsp オプションは、スプリットモード指定 (-sp) を無効にします。

■ スプリットモード抑止指定 (-Xsp)

【記述形式】

-Xsp

【パラメータ】

なし

【説明】

本オプションは、-sp 指定を無効にする場合に指定します。

本オプションはデフォルトですので特に指定する必要はありません。

【例】

```
m2bs ccp903.mhx -ran 0xE000,0xFFFF
```

```
m2bs cpp903.mhx -Xsp -ran 0xE000,0xFFFF
```

デフォルト処理では、スプリットモードでは動作しません。

上記指定はどちらも同じです。

```
m2bs -f option.file ccp903 -Xsp
```

オプションファイルを利用した実行時、オプションファイル内の指定を一時的に変更したい場合があります。

option.file 内に -sp オプションがある場合、option.file の内容を変更せずコマンドライン上で -Xsp オプション指定を行えば -sp オプションを取り消すことができます。

17.3.4 マップリストファイルの作成指定 (-m)

コンバート時の情報をマップリストファイルに出力します。

■ マップリストファイルの作成指定 (-m)

【記述形式】

-m <マップリストファイル名>

【パラメータ】

<マップリストファイル名>
出力マップリストファイル名

【説明】

コンバート時の情報をマップリストファイルに出力します。
マップリストファイルには、コンバート時の情報が出力されます。出力される情報には以下があります。項目 4) は -sp オプションが指定された場合にのみ出力されます。

- 1) 入力ファイル名情報
- 2) 出力ファイル名情報
- 3) 出力範囲情報
- 4) スプリット単位情報
- 5) パディングデータ値情報

<マップリストファイル> の拡張子を省略した場合、デフォルト拡張子 ".mp3" が付加されます。

【例 1】

m2bs sfmtfile.mhx -ran 0x10000,0x1FFFF (-m オプションの指定がない例)
-m オプションの指定がないのでマップリストファイルの作成は行われません。

【例 2】

m2bs sfmtfile.mhx -ran 0x10000,0x1FFFF -m logfile
logfile.mp3 にコンバート時の情報が出力されます。

図 17.3-1 logfile.mp3 の内容例 1

Input file	:sfmtfile.mhx
Output file	:sfmtfile.bin
Convert range	:0x00010000 - 0x0001FFFF
Padding data	:0xFF

【例 3】

```
m2bs sfmtfile.mhx -ran 0x10000,0x1FFFF -m logfile -sp 2,2
```

logfile.mp3 にコンバート時の情報が出力されます。

図 17.3-2 logfile.mp3 の内容例 2

Input file	:sfmtfile.mhx
Output file	:sfmtfile.b01
	:sfmtfile.b02
Convert range	:0x00010000 - 0x0001FFFF
Split byte	:2
Padding data	:0xFF

17.3.5 マップリストファイルの作成抑止指定 (-Xm)

-Xm オプションは、マップリストファイルの作成を抑止します。

■ マップリストファイルの作成抑止指定 (-Xm)

【記述形式】

-Xm

【パラメータ】

なし

【説明】

本オプションは、-m 指定を無効にする場合に指定します。

本オプションはデフォルトですので特に指定する必要はありません。

【例 1】

```
m2bs ccp903.mhx -ran 0xE000,0xFFFF
```

```
m2bs ccp903.mhx -Xm -ran 0xE000,0xFFFF
```

デフォルト処理では、マップリストファイルを出力しません。

上記指定はどちらも同じです。

【例 2】

```
m2bs -f option.file ccp903 -Xm
```

オプションファイルを利用した実行時、オプションファイル内の指定を一時的に変更したい場合があります。

option.file 内に -m オプションがある場合、option.file の内容を変更せずコマンドライン上で -Xm オプション指定を行えば -m オプションを取り消すことができます。

第18章

その他のコンバータ

この章では、その他のコンバータの各コマンドを詳しく説明します。

18.1 m2is(S フォーマット → HEX8 フォーマット変換)

18.2 m2es(S フォーマット → HEX16 フォーマット変換)

18.3 i2ms(HEX8 フォーマット → S フォーマット変換)

18.4 e2ms(HEX16 フォーマット → S フォーマット変換)

18.1 m2is(S フォーマット HEX8 フォーマット変換)

S フォーマットを HEX8 フォーマットに形式変換します。

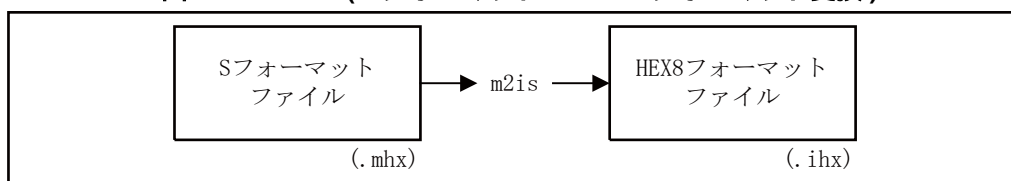
0 ~ 0xFFFF 番地のデータが変換対象になります。

■ m2is(S フォーマット HEX8 フォーマット変換)

【機能】

m2is コマンドは, S フォーマットファイルから HEX8 フォーマットへの変換を行います。

図 18.1-1 m2is(S フォーマット HEX8 フォーマット変換)



< 注意事項 >

S フォーマットは, 0 ~ 0xFFFFFFFF のアドレスを表せますが, HEX8 フォーマットへコンバートすると 0x10000 以上のアドレスに割り付けられたデータは切捨てられます。

コンバート元のアドレス範囲に気をつけて使用してください。

HEX8 フォーマットのファイルは, データレコードとトレーラレコードで構成されますので, S フォーマットのスタートアドレス情報は失われます。

18.2 m2es(S フォーマット HEX16 フォーマット変換)

S フォーマットを HEX16 フォーマットに形式変換します。

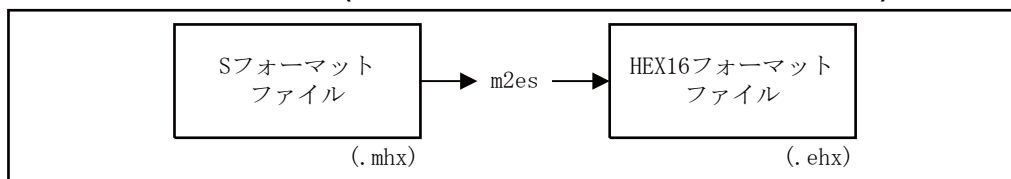
0 ~ 0xFFFFF 番地のデータが変換対象になります。

■ m2es(S フォーマット HEX16 フォーマット変換)

【機能】

m2es コマンドは、S フォーマットファイルから HEX16 フォーマットへの変換を行います。

図 18.2-1 m2es(S フォーマット HEX16 フォーマット変換)



< 注意事項 >

S フォーマットは、0 ~ 0xFFFFFFFF のアドレスを表せますが、HEX16 フォーマットへコンバートすると 0x100000 以上のアドレスに割り付けられたデータは切捨てられます。コンバート元のアドレス範囲に気をつけて使用してください。

18.3 i2ms(HEX8 フォーマット S フォーマット変換)

HEX8 フォーマットを S フォーマットに形式変換します。

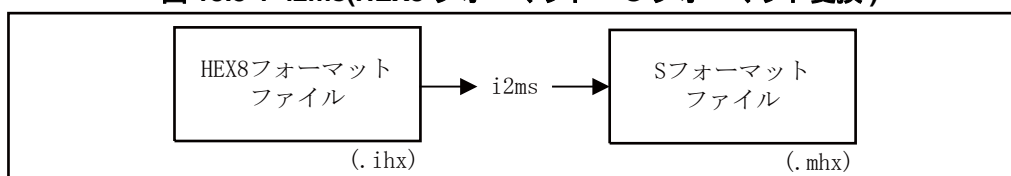
0 ~ 0xFFFF 番地のデータが変換対象になります。

■ i2ms(HEX8 フォーマット S フォーマット変換)

【機能】

i2ms コマンドは, HEX8 フォーマットから S フォーマットファイルへの変換を行います。

図 18.3-1 i2ms(HEX8 フォーマット S フォーマット変換)



< 注意事項 >

S フォーマットは, 0 ~ 0xFFFFFFFF のアドレスを表せますが, HEX8 フォーマットは 0x10000 以上のアドレスを表現できません。コンバート後の S フォーマットファイルは, S2, S3, S7, S8 タイプなしで作成されます。

HEX8 フォーマットはスタートアドレスの情報がありません。変換後スタートアドレスは 0 とします。

18.4 e2ms(HEX16 フォーマット S フォーマット変換)

HEX16 フォーマットを S フォーマットに形式変換します。

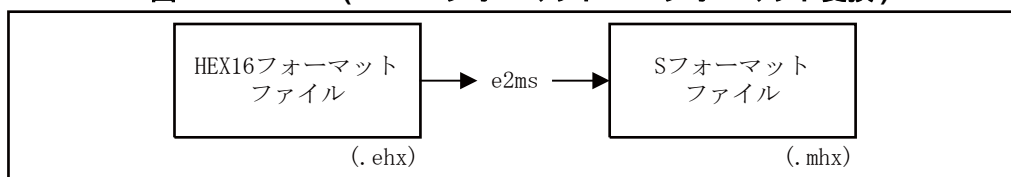
0 ~ 0xFFFFF 番地のデータが変換対象になります。

■ e2ms(HEX16 フォーマット S フォーマット変換)

【機能】

e2ms コマンドは、HEX16 フォーマットから S フォーマットファイルへの変換を行います。

図 18.4-1 e2ms(HEX16 フォーマット S フォーマット変換)



< 注意事項 >

S フォーマットは、0 ~ 0xFFFFFFFF のアドレスを表せますが、HEX16 フォーマットは 0x100000 以上のアドレスを表現できません。コンバート後の S フォーマットファイルは、S7, S3 タイプなしで作成されます。

HEX16 フォーマットのスタートアドレス情報は、S9 または S8 タイプに設定されます。

第19章

オブジェクト形式コンバータ の制限事項およびQ&A

この章では、オブジェクト形式コンバータの制限事項や使用上のQ &A について述べます。

19.1 オブジェクト形式コンバータの制限事項

19.2 オブジェクト形式コンバータの使用上のQ&A

19.1 オブジェクト形式コンバータの制限事項

バイナリコンバータおよびアジャスタには、いくつかの制限があります。その他のオブジェクト形式コンバータは使用する上で、処理上の制限事項は設けておりません。オブジェクト形式コンバータが実行に使用できるメモリを使いきってしまうまで処理可能です。

■ オブジェクト形式コンバータの制限事項

オブジェクト形式コンバータは、表 19.1-1 に示す制限があります。

ただし、最大限界値まで処理が可能なわけではありません。

オブジェクト形式コンバータは動的にメモリ獲得を行いながら処理を行います。

処理に必要なメモリ獲得ができなくなった場合には、メモリ不足のエラーメッセージを通知し、処理を中断します。

表 19.1-1 オブジェクト形式コンバータの制限一覧

項 目	制限値	備 考
オプションファイル数	無制限	メモリ依存
オプションファイル内の行数	無制限	メモリ依存
オプションファイル内の 1 行の文字数	無制限	メモリ依存
オプションファイルのネスト	不可	
入力ファイル数 (m2bs, m2ms, h2bs, h2hs)	64 個	
入力ファイル数 (上記以外のコンバータ)	1 個	
入出力ファイルサイズ	無制限	OS 依存
入出力ファイル行数	無制限	OS 依存
ファイル名文字数	無制限	OS 依存
最大メモリアドレス	0xffffffff	
最大変換サイズ (m2bs, m2ms, h2bs, h2hs)	2G バイト -1 バイト	メモリ依存

■ バイナリコンバータ、アジャスタの制限事項

入力ファイルは合計で 64 個まで指定できます。

入力ファイルが複数指定された場合、指定された順に処理します。

入力ファイル中で、同じアドレスに対するデータが存在する場合は、後から出てきたデータで上書きされます。

1 度に変換できるサイズは、最大 2G バイト -1 バイトまでです。

19.2 オブジェクト形式コンバータの使用上の Q&A

オブジェクト形式コンバータの使用に関する Question と Answer を示します。

■ オブジェクト形式コンバータの使用上の Q&A

Q.	多数のコンバータがありますが、どれを使用すれば良いのですか？
A.	コンバータの使用目的は、リンカ出力の絶対形式ロードモジュールファイルを ROM ライタで読み込み可能なオブジェクト形式に変換することです。 32bit のアドレス空間に完全に対応しているのは f2ms と f2hs です。S フォーマットへ変換の際は f2ms を、HEX フォーマットへ変換の際は f2hs の使用を推奨します。 その他のコンバートツールは、必要に応じて御使用ください。
例	f2ms absfile.abs -> S フォーマット absfile.mhx 出力

Q.	バイナリコンバータやフォーマットアジャスタ使用時に、"F9001U: メモリが足りません" とエラー出力され、変換できません。どうすれば良いのですか？
A.	バイナリコンバータやフォーマットアジャスタは、動作時に変換する領域と同じサイズのメモリを確保します。このため、1 度に広大な範囲をメモリアイメージに変換しようとする "F9001U: メモリが足りません" とエラー出力され処理が中断される可能性があります。 このような場合には、変換したい領域を連続する複数の領域に分割し、それぞれのメモリアイメージを作成後、ファイルを連結して 1 つのバイナリイメージにしてください。
例	0xC00000 ~ 0xFFFFFFFF までのバイナリイメージ領域を変換する場合 ・十分に使用メモリが確保できる場合 (通常) m2bs absfile.mhx -ran 0xC00000,0xFFFFFFFF ・"F9001U: メモリが足りません" とエラー出力された場合 m2bs absfile.mhx -ran 0xC00000,0xDFFFFFF -o absfile1.bin m2bs absfile.mhx -ran 0xE00000,0xFFFFFFFF -o absfile2.bin copy /b absfile1.bin + absfile2.bin absfile.bin

付録では、リンケージキットのエラーメッセージ、
HEX フォーマット、および S フォーマット形式な
どを記載しています。

付録 A リンケージキットのエラーメッセージ

付録 B HEX フォーマット

付録 C S レコード形式

付録 D リンカのオプション一覧表

付録 E ライブラリアンのオプション一覧表

付録 F オブジェクト形式コンバータのコマンドおよびオプ
ション一覧表

付録 G OS による仕様の相違点

付録 A リンケージキットのエラーメッセージ

リンケージキットの各ツールが出力するエラーメッセージの分類と表示形式について詳しく説明します。

■ リンケージキットのエラーメッセージ分類

エラーメッセージは、重要度に応じて次の4つのレベルに分類されています。

- インフォメーション

処理を確認する目的で、ユーザにその内容を通知するものです。エラーではありませんので、正しい処理結果が得られています。

- 警告

エラーよりは軽微であり、出力結果はほとんど問題なく使用できます。場合により、ユーザの意図と異なる処理が行われている可能性もあります。メッセージ内容を確認した上で、出力結果が使用可能か否かを判断してください。

- エラー

処理は実行しますが、正しい結果は得られないレベルの問題が発生しています。エラーの原因を取り除いて、再実行する必要があります。

- 致命的エラー

処理の実行が不可能なエラーです。ユーザの誤指定が原因のほか、実行環境の問題で発生します。

■ リンケージキットのエラーメッセージの表示形式

エラーメッセージは、各ツールとも図 A-1 に示す形式で出力します。

図 A-1 リンケージキットのエラーメッセージの表示形式

```
*** ファイル名(行番号) XnnnnT: メッセージ文(補助メッセージ)
```

各部	説明
ファイル名 (行番号)	エラーの発生したソースファイル名と、ソース行番号 この情報は、リンカの一部のメッセージでのみ出力されます。
X	エラーのレベルを次の 4 つの英字 1 文字で表します。 I ... インフォメーション E ... エラー W ... 警告メッセージ F ... 致命的なエラー
nnnn	エラー番号 エラー番号と、エラーレベルは以下の対応関係があります。 0000 ~ 0999 ... I 1000 ~ 1999 ... W 4000 ~ 4999 ... E 9000 ~ 9999 ... F
T	ツール識別を次の英字 1 文字で表します。 L ... リンカ U ... ライブラリアン、オブジェクト形式コンバータ
メッセージ文	エラーメッセージ本文 (日本語 / 英語の選択が可能)
補助メッセージ	エラーについてのより詳細な情報 エラー発生原因となったシンボル名などを表示します。 エラーメッセージ本文中に出力されることもあります。

【例】

```
*** sample.c(234) E4329L: Value out of range (0xFFFFE37D4)
```

ソースファイル名と行番号の表示も行われている例です。

```
*** E4402U: Duplicated module name (date.obj setdate)
```

ソースファイル名と行番号の表示が行われない例です。

```
*** F9001U: Insufficient memory
```

ソースファイル名と行番号および補助メッセージの表示は行われない例です。

■ リンカのエラーメッセージ

I0301L	このライブラリは使用されませんでした (ファイル名)
	Unused library (ファイル名)

リンク処理で使用されなかったライブラリがあります。

-w オプションで 2 を指定したときに通知されるメッセージです。

I0302L	デバッグ情報が存在しません (ファイル名)
	Debug information not exist (ファイル名)

入力ファイルにデバッグ情報が含まれていません。

-w オプションで 2 を指定したときに通知されるメッセージです。

W1351 より軽微です。単にデバッグ情報が存在しないことを通知するだけです。

I0303L	デバッグ情報を取り除きました
	Removed debug information

デバッグ情報を取り去って出力ファイルを作成しました。

-w オプションで 2 を指定したときに通知されるメッセージです。

I0304L	このファイルは警告レベルエラーを含んでいます (ファイル名)
	File include WARNING level error (ファイル名)

ここで示されたファイルは、以前のリンク時に警告メッセージがあったものです。

-w オプションで 2 を指定したときに通知されるメッセージです。

I0305L	境界調整を無視しました
	Ignore address alignment

-pk オプションの指定により、リンク時にバウンダリ調整を無視して配置を行いました。

-w オプションで 2 を指定したときに通知されるメッセージです。

I0306L	領域 "領域名" で、セクション最適化配置を行います
	Section allocated automatically in "領域名" area

ここで示された領域でセクション最適化配置を行いました。

-w オプションで 2 を指定したときに通知されるメッセージです。

I0307L	下位互換 CPU タイプのオブジェクトです (ファイル名)
	Lower compatible cpu type object (ファイル名)

ここで示されたファイルは、下位互換性のある異なる CPU タイプのモジュールです。

I0309L	Softune V3 のオブジェクトです (ファイル名)
	Softune V3 type object (ファイル名)

ここで示されたファイルは、SOFTUNE V3 言語ツールで作成されたモジュールです。

-w オプションで 2 を指定したときに通知されるメッセージです。

I0310L	Softune V5 のオブジェクトです (ファイル名)
	Softune V5 type object (ファイル名)

ここで示されたファイルは、SOFTUNE V5 言語ツールで作成されたモジュールです。

-w オプションで 2 を指定したときに通知されるメッセージです。

I0311L	値が範囲外です (-pw)
	Specified value out of range (-pw)

pw オプションで、70 ~ 79 の値が指定されました。リンクは桁数を 80 桁にします。

W1301L	ROM 領域に書き込み可能セクションが配置されました (セクション名)
	Writable section located in ROM area (セクション名)

ROM 領域として設定したアドレス範囲に書き込み可能セクションを配置しました。

セクション配置指定を見直してください。

W1303L	RAM エリアに初期値のあるセクションが配置されました (セクション名)
	Section with initial data located in RAM area (セクション名)

RAM 領域として設定したアドレス範囲に初期値を含むセクションを配置しました。

セクション配置指定を見直してください。

W1305L	ROM/RAM 領域 (領域名) の範囲外への配置がありました (セクション名)
	Section (セクション名) located on out of ROM/RAM area (領域名)

-ro または -ra オプションで指定したアドレス範囲を超えて配置されたセクションがあります。セクションマップで確認してください。

リンクのマップリストでは十分な情報が得られない場合は、セクション詳細マップリストでより詳細な情報を知ることができます。

W1306L	最大アドレスを超えました (セクション名)
	Exceeded maximum address (セクション名)

最大アドレスを超えて配置されたセクションがあります。

W1307L	セクション名が重複しています (セクション名)
	Duplicate section name exist (セクション名)

複数のモジュールに、名前が同じで属性や型の異なるセクションが存在します。

W1308L	セクション配置が重複しています (セクション 1, セクション 2)
	Overlap located section (セクション 1, セクション 2)

セクションが重複して配置されました。場合によってはプログラムの作動に支障を来しますので、注意が必要です。

オプションで回避することを推奨します。

W1312L	互換性のない CPU タイプのモジュールがあります (ファイル名)
	Uncompatible cpu type module (ファイル名)

ここで示されるファイルは互換性のないモジュールです。

入力されるモジュールは、ターゲット CPU が同じか互換 CPU でなくてはなりません。

本メッセージは、以下の場合に出力されます。

- FR80 用ロードモジュール作成時に FR 用のモジュールを入力した場合

警告の出力されたファイルについては、非互換の命令が使われていないことをご確認ください。

FR と FR80 の非互換命令など、詳細は「5.12 FR 用オブジェクトと FR80 用オブジェクトの混在」をご参照ください。

W1314L	絶対セクションに対して配置アドレスが指定されました (セクション名)
	Specified address to absolute section (セクション名)

ここで示されるセクションは、絶対番地があらかじめ決められていますので、セクションの再配置指定は無効です。

W1320L	モード識別 モード時にはライブラリ検索を行いません
	Not search library at モード識別 mode

ライブラリ検索処理は、絶対形式ロードモジュールを作成するとき以外行いません。

-r オプション指定時にはライブラリの検索指定は無効です。

補助メッセージのモード識別には、次が表示されます。

相対形式ロードモジュール出力指定 (-r) 時 : REL

W1321L	モード識別 モード時にはこの (オプション) オプションは無視されます
	Ignore (オプション) Option at モード識別 mode

指定されたリンクモードでは無効となる指定がなされたときに通知されます。

たとえば, 相対形式ロードモジュール出力指定時には, セクション配置処理は行いませんので -sc オプションを指定しても無効になります。

W1325L	エントリポイントが複数設定されています
	Entry point already set

複数の入力モジュールにエントリポイントが設定されています。

最初に設定されたエントリポイントが有効になります。

W1326L	エントリポイントが変更されました
	Entry point was changed

-e オプションの指定により, 既に設定されているエントリポイントが変更されました。

W1327L	このシンボルは既に定義されています (マングル名 シンボル名)
	Duplicate symbol definition (マングル名 シンボル名)

複数の入力モジュールに同じ外部定義シンボルが存在します。

最初に定義されたシンボル値が有効になります。

W1328L	外部シンボルの型が異なっています (マングル名 シンボル名)
	Mismatch symbol type (マングル名 シンボル名)

-e オプションによるエントリポイントの設定時に指定された外部定義シンボルが, 関数名, 変数名, アドレスラベル名のいずれでもない場合に通知されます。

E4326 より軽微な場合に検出されます。

W1332L	該当するファイルが見つかりませんでした (ファイル名)
	No match (ファイル名) argument

ワイルドカードによるファイル指定が行われましたが, 該当するファイルがありませんでした。

この指定を無視します。

W1351L	デバッグ情報が不足しています
	Debug information not exist

デバッグ情報が不足しているために、リストの一部が作成できません。

コンパイル、アセンブルおよびリンク時にデバッグ情報付加オプション `-g` を指定してください。

W1367L	モジュール名の重複がありました (ファイル名 モジュール名)
	Duplicated module name (ファイル名 モジュール名)

既にリンク対象となっているモジュール名と同じ名前のモジュール名が存在します。

モジュール名の重複は、絶対アセンブルリスト作成時やデバッグ時に正しくモジュールを特定できないなどの問題発生原因になります。

モジュール名の重複が起こらないように変更をお勧めします。

C/C++ コンパイラは、ファイル名から自動的にモジュール名の生成を行います。

ファイル名を変更してください。

W1368L	<code>-ro</code> オプションで内蔵 ROM 領域の範囲外が指定されています (領域名)
	The area specified for the <code>-ro</code> option is outside the internal-ROM area (領域名)

`-ro` オプションで指定した領域が、内蔵 ROM 領域の範囲外です。

指定範囲が正しいか確認をしてください。

ワーニング出力抑止 (`-w 0`) が指定されている場合でも本警告は出力されます。

W1369L	<code>-ra</code> オプションで内蔵 RAM 領域の範囲外が指定されています (領域名)
	The area specified for the <code>-ra</code> option is outside the internal-RAM area (領域名)

`-ra` オプションで指定した領域が、内蔵 RAM 領域の範囲外です。

指定範囲が正しいか確認をしてください。

ワーニング出力抑止 (`-w 0`) が指定されている場合でも本警告は出力されます。

W1370L	ROM 領域外への配置です (セクション名)
	The section is placed outside the ROM area (セクション名)

本来ROM領域内に配置されるべきセクションがROM領域の範囲外に配置されています。

配置が正しいかマップファイルで確認をしてください。

ワーニング出力抑止 (`-w 0`) が指定されている場合でも本警告は出力されます。

W1371L	RAM 領域外への配置です (セクション名)
	The section is placed outside the RAM area (セクション名)

本来RAM領域内に配置されるべきセクションがRAM領域の範囲外に配置されています。
配置が正しいかマップファイルで確認をしてください。

ワーニング出力抑止 (-w 0) が指定されている場合でも本警告は出力されます。

W1372L	RAM 領域または IO 領域外への配置です (セクション名)
	The section is placed outside the RAM area or the I/O area (セクション名)

本来 RAM 領域か IO 領域内に配置されるべきセクションが RAM 領域と IO 領域の範囲外に配置されています。

配置が正しいかマップファイルで確認をしてください。

ワーニング出力抑止 (-w 0) が指定されている場合でも本警告は出力されます。

W1373L	IO 領域外への配置です (セクション名)
	The section is placed outside the I/O area (セクション名)

本来 IO 領域内に配置されるべきセクションが I/O 領域の範囲外に配置されています。
配置が正しいかマップファイルで確認をしてください。

ワーニング出力抑止 (-w 0) が指定されている場合でも本警告は出力されます。

E4302L	セクションまたはセクショングループ名が見つかりません (セクション名 or グループ名)
	Not found section or section group name (セクション名 or グループ名)

-sc オプションで指定したセクション名あるいはグループ名が見つかりません。

E4303L	ROM/RAM 領域名が定義されていません (領域名)
	Undefined ROM/RAM area name (領域名)

-sc オプションで指定した ROM/RAM 領域名が定義されていません。

E4304L	シンボル名が見つかりません (マングル名 シンボル名)
	Symbol name is not found (マングル名 シンボル名)

-df オプションで指定した外部参照シンボルが見つからないとき、または -e オプションで指定した外部定義シンボルが見つからないときに通知されます。

E4305L	外部シンボルが定義されていません (マングル名 シンボル名)
	Unresolved external symbol (マングル名 シンボル名)

外部定義シンボルの定義がなかったため、リロケーション処理ができませんでした。
 -df オプションで指定した外部定義シンボル名が見つからないときにも通知されます。
 外部定義シンボルを含むモジュールの結合が必要です。

E4312L	互換性のない CPU タイプのモジュールがあります (ファイル名)
	Uncompatible cpu type module (ファイル名)

ここで示されるファイルは互換性のないモジュールです。
 入力されるモジュールは、ターゲット CPU が同じか互換 CPU でなくてはなりません。
 本メッセージは、以下の場合に出力されます。

- FR 用ロードモジュール作成時に FR80 用のモジュールを入力した場合
- FR80 用ロードモジュール作成時に FR 用のモジュールを入力した場合

FR と FR80 の非互換命令の詳細は「5.12 FR 用オブジェクトと FR80 用オブジェクトの混在」を参照してください。

E4319L	このセクションは存在しません (セクション名)
	Section not exists (セクション名)

グループ化オプション (-gr) で指定されたセクションが見つかりません。
 ここで示されるセクションが絶対セクションであった場合にも、セクション再配置の対象にならないため、当メッセージが通知されます。

E4326L	外部シンボルの型が異なります (マングル名 シンボル名)
	Different symbol type (マングル名 シンボル名)

-e オプションによるエントリポイントの設定時に指定された外部定義シンボルが、関数名、変数名、アドレスラベル名のいずれでもない場合に通知されます。

E4327L	リロケーション情報に誤りがあります
	Illegal RL information

リロケーション情報に誤りがある場合に検出されます。

E4329L	値が範囲外です (値)
	Value out of range (値)

リロケーション演算でオーバフローがありました。

このメッセージには、リロケーション対象となったデータ部の記述を含むソースプログラム名と行番号が表示されますので、プログラムの記述を確認してください。

エディタのタグジャンプが利用できる場合、該当するソース行がすぐにわかります。

E4330L	0 による除算が行われました
	Divide by 0

リロケーション演算の除算で除数が 0 になっています。

E4331L	配置アドレスを指定してください (セクション名)
	Indispensable to locate address (セクション名)

-sc オプションによるアドレス指定がないか、指定のないセクションが見つかった場合に通知されます。

E4332L	グループ名は使えません (グループ名)
	Not handling group name (グループ名)

-sc オプションでの ROM RAM 転送セクションの指定時に、グループ名は指定できません。

E4333L	ROM アドレスが指定されていません (セクション名)
	Not specified ROM address (セクション名)

ROM 領域に配置すべきリロケーション情報に対し、ROM アドレスの指定がありません。

-sc オプションで ROM アドレスの指定を行ってください。

E4351L	相対アセンブルリストファイルに対するオブジェクトがありません (ファイル名)
	Relocatable assemble list not correspond to object file (ファイル名)

補助メッセージに示す相対アセンブルリストファイルに対するオブジェクトが見つかりません。

リンカは補助メッセージに示すリストファイルの処理を中止し、次のリストファイルの処理を行います。

E4352L	相対アセンブルリストファイルがありません (ファイル名)
	Relocatable assemble list file not found (ファイル名)

補助メッセージに示す相対アセンブルリストファイルがありません。

リンカは補助メッセージに示すファイル名に対する絶対形式アセンブルリストファイルを作成せず、処理を続行します。

E4354L	相対アセンブルリストファイルの形式が正しくありません (ファイル名)
	Illegal relocatable assemble list file format (ファイル名)

補助メッセージに示す相対アセンブルリストファイルの形式は、リンカで正しく処理できるものではありません。

リンカは補助メッセージに示すリストファイルの処理を中止し、次のリストファイルの処理を行います。

E4355L	オブジェクトデータが一致していません (ファイル名)
	Object data not correspond (ファイル名)

絶対形式ロードモジュールファイルのオブジェクトデータと、補助メッセージに示す相対形式アセンブルリストのソースデータが一致していません。

アセンブルし直してから、再度実行してください。

E4357L	配列の次元数、または構造体のネストが多すぎます
	Too many array dimension or structure nested : exceeded 8

ARRAY リストに出力される配列の次元数または、構造体のネストが8を超えています。処理は続行しますが、これを超えた部分はリストに出力されません。

E4362L	ダミーセクションの指定があります (ファイル名)
	DUMMY section specified (ファイル名)

補助メッセージに示すファイルでダミーセクションが指定されています。

絶対形式アセンブルリストを作成する場合にはダミーセクションの記述をしないでください。

E4363L	セクションの最大サイズを超えました (セクション名)
	Exceeded maximum section size (セクション名)

セクションの最大サイズを超えたセクションがあります。セクションには、そのセクション種別により最大サイズが制限されるものがあります。それを超えた場合に検出されます。

セクションの見直しを行ってください。

E4365L	配置可能なアドレスが領域 " 領域名 " に見つかりません (セクション名)
	Not found locatable address in 領域名 (セクション名)

配置指定領域の該当セクションの配置可能な場所が見つかりません。

領域内のセクション構成を見直してください。

E4366L	配置可能なアドレスが見つかりません。(セクション名)
	Not found locatable address (セクション名)

全アドレス空間にセクションの配置可能な場所が見つかりません。

プログラム構成を見直してください。

E4367L	モジュール名の重複がありました (ファイル名 モジュール名)
	Duplicated module name (ファイル名 モジュール名)

既にリンク対象となっているモジュール名と同じ名前のモジュール名が存在します。

モジュール名の重複はできません。モジュール名を変更してください。

E4369L	コンパイルモデルの異なるモジュールがあります (ファイル名)
	Invalid module : conflict compile model (ファイル名)

コンパイルモデルの異なるモジュールはリンクできません。

E4370L	CPU 情報ファイルが見つかりません (ファイル名)
	CPU information file not found (ファイル名)

-cpu オプションによる指定ターゲットの CPU 情報ファイルが見つかりません。

下記ファイルが見つからない場合に検出されます。

- %FETOOL%\LIB\911\911.CSV
- %FETOOL%\LIB\911\cpu_info*.CSV

E4371L	CPU 情報が見つかりません (ファイル名)
	CPU information not found (ファイル名)

-cpu オプションによる指定ターゲットの CPU 情報が CPU 情報ファイル内に見つかりません。

下記ファイル内に指定ターゲットの CPU 情報が見つからない場合に検出されます。

- %FETOOL%\LIB\911\911.CSV
- %FETOOL%\LIB\911\cpu_info*.CSV

E4701L	実体生成されたテンプレート関数と同一の関数が定義されています (シンボル名)
	Symbol is referenced as an explicit specialization and a generated instantiation (シンボル名)

実体生成されたテンプレート関数と同一の関数が定義されています。

E4702L	シンボル名のデコーディング中にエラーが発生しました (シンボル名)
	Error occurred during symbol name decoding (シンボル名)

シンボル名のデコーディング中にエラーが発生しました。シンボル名はデコーディング以前のまま使用されます。

F9001L	メモリが足りません
	Insufficient memory

プログラム実行のためにはメモリが不足しています。バッチファイル中でリンクを起動している場合は、直接コマンドラインから起動してみてください。

F9011L	ファイルが見つかりません (ファイル名)
	Input file not found (ファイル名)

指定された入力ファイルが見つかりません。

F9012L	ライブラリファイルが見つかりません (ファイル名)
	Library file not found (ファイル名)

ここで示されるファイル名がデフォルトライブラリファイルである場合は、以下のことが原因です。

- 環境変数 LIB911 が設定されている場合。

環境変数 LIB911 が示すディレクトリにデフォルトライブラリファイルが格納されていない。

- 環境変数 LIB911 が設定されていない場合。

環境変数 FETOOL より導かれるディレクトリにデフォルトライブラリファイルが格納されていない。

-l, -el オプションで指定したライブラリファイルであるときは、ファイル名指定もしくは -L オプションで指定の検索パスが間違っていないか確かめてください。

F9015L	ファイルがオープンできません (ファイル名)
	File open error (ファイル名)

ここで示されたファイルが出力ファイルである場合、1つのディレクトリが管理できるファイルの制限数を超過している可能性があります。不要なファイルを削除するか移動してください。

F9016L	ファイルからの読み込みができません (ファイル名)
	File read error (ファイル名)

リード許可のないファイルである可能性があります。

F9017L	ファイルへの書き込みができません (ファイル名)
	File write error (ファイル名)

書き込み許可のない同名のファイルが存在している可能性があります。

または、ここで示されているファイルを書き込んでいるディスクの容量が不足している可能性があります。ディスクに空き容量を作ってもう一度リンクを実行してください。

F9021L	コマンドラインの指定が多すぎます
	Too many options

コマンドラインに指定した (オプションファイル中の指定も含む) , 入力ファイル名やオプションが多すぎます。

リンクの起動を複数回に分けてください。

F9022L	オプション名の指定に誤りがあります (オプション)
	Illegal option name (オプション)

リンクで利用できるオプションではありません。-help オプションまたは当マニュアルで確認してください。

F9023L	パラメータの指定に誤りがあります (オプション)
	Illegal option parameter (オプション)

このオプションで指定すべきパラメータに誤りがあります。

パラメータが不足している、区切り記号が誤っている、などシンタックスに起因するエラーであると思われる場合に通知します。

F9024L	パラメータの指定に不正な文字があります (オプション)
	Illegal character (オプション)

このオプションで指定すべきパラメータに誤りがあります。

数値の指定方法の誤りなど不正文字の使用によるエラーであると思われる場合に通知します。

F9026L	値が範囲外です (値)
	Specified value out of range (値)

-pl, -pw, -w オプションで許される範囲外の値が指定されています。

F9027L	オプションファイル中にオプションファイルの指定があります
	Option file nested

オプションファイルはネストできません。オプションファイル中に記述されている -f オプションを削除してください。

F9030L	入力ファイル名の指定がありません
	Missing input file name

入力ファイルを指定してください。

F9032L	入出力ファイル名が同じです (ファイル名)
	Output file name same as input one (ファイル名)

ここに示された出力ファイル名は、入力ファイル名と同じ名称ですので処理できません。

F9033L	ファイルの形式が正しくありません (ファイル名)
	Illegal file format (ファイル名)

以下のいずれかの場合に通知されるメッセージです。

- ライブラリファイルの形式が正しくない。
- ライブラリファイル中のオブジェクトモジュールの形式が正しくない。
- 入力ファイルが絶対形式ロードモジュールである。
- 入力モジュールの内容が正しくない。
- 相対形式アセンブルリストファイルの形式が正しくない。
- CPU 情報ファイルの形式が正しくない。

F9040L	ファイル名またはパス名が重複しています (ファイル名)
	Duplicated file or path name (ファイル名)

以下の 3 つのいずれかの場合に通知されるメッセージです。

- 同名の入力モジュールファイル名が指定されている。
- -l, -el オプションで同名のライブラリファイル名が指定されている。
- -L オプションで同じライブラリパス名が指定されている。

F9042L	セクション名の重複指定があります (セクション名)
	Duplicated section name (セクション名)

-sc または -gr オプションで、同一のセクション名が複数指定されました。

F9043L	シンボル名の重複指定があります (マングル名 シンボル名)
	Duplicated symbol name (マングル名 シンボル名)

-el オプションで、同じ外部参照シンボルが複数指定されています。

F9044L	セクショングループ名が重複しています (グループ名)
	Duplicated section group name (グループ名)

-gr オプションによるグループの設定時に、同じグループ名が複数使用されました。

F9047L	該当するファイルが見つかりませんでした (ファイル名)
	No match ファイル名 argument

ワイルドカードによるファイル指定が行われましたが、該当するファイルがありませんでした。

F9052L	'-cpu' オプションの指定がありません
	Missing '-cpu' option

-cpu オプションによるターゲット CPU の指定がありません。

-cpu オプションでターゲット CPU を指定する必要があります。

F9053L	'-ro','-ra' オプションの指定がありません
	Missing '-ro' or '-ra' option

自動配置を行うのに必要な '-ro','-ra' オプションが指定されていません。

F9054L	セクション名が重複しています (セクション名)
	Duplicate section name exist (セクション名)

複数のモジュールに、名前が同じで属性や型の異なるセクションが存在します。

絶対アセンブルリスト作成時に出力されます。

F9055L	プレリンクプロセスでエラーが発生しました
	Prelink command process returns error

プレリンクプロセスでエラーが発生しました。

F9056L	CPU 情報ファイルのバージョンが適合しません
	Mismatch CPU information file version

CPU 情報ファイルが古く適合しません。

最新の CPU 情報ファイルを入手してください。

F9070L	ディレクトリを変更できません (ディレクトリ名)
	Cannot change directory (ディレクトリ名)

実体生成情報ファイルのディレクトリ記述が正しくない可能性があります。各オブジェクトをコンパイルし直して実体生成情報ファイルを更新し、リンカを再実行してみてください。

F9072L	コンパイルプロセスでエラーが発生しました
	Compile command process returns error

リンカは、C++ で作成されたオブジェクトファイルをリンクする際、テンプレート関数を生成するために、C/C++ コンパイラを使用します。

このメッセージは以下の場合に出力されます。

- 入力ファイル (.obj) の形式が正しくない。
- C/C++ コンパイラを起動するためのメモリが不足している。
- C/C++ コンパイラがエラーを出力している。

F9073L	同一の実体生成が複数ファイルで指定されています (ファイル名)
	Instantiation assigned to more than one file (ファイル名)

別の実体生成情報ファイルで同一シンボルの実体生成が指定されています。各入力ファイルをコンパイルし直して実体生成情報ファイルを更新し、リンカを再実行してください。

F9074L	実体生成情報ファイルが見つかりません (ファイル名)
	Instantiation information file is missing (ファイル名)

入力ファイル (*.obj) に対して実体生成が必要ですが、実体生成情報ファイルが見つかりません。

入力ファイルをコンパイルし直して実体生成情報ファイルを作成し、リンカを再実行してください。

F9998L	
	File open failed (ファイル名)

リンカが使用するメッセージファイルがオープンできませんでした。

エラーメッセージファイルは、決められたディレクトリに格納してください。

- lkt911_a.msg または lkt911_e.msg

F9999L	プログラム内部エラーが発生しました (識別情報)
	Internal error (識別情報)

このエラーが出た場合は、ただちに営業部門へ連絡してください。

■ ライブラリアンのエラーメッセージ

I0401U	外部定義のない外部参照シンボルがあります
	Reference to undefined symbol

-c オプションにより報告されるメッセージです。ライブラリファイルの外部シンボルをチェックした結果、当ライブラリファイル中で解決されない外部参照シンボルが含まれていることを示しています。当メッセージの出力されるライブラリファイルをリンカで使用する場合は、どのモジュールの外部定義シンボルが使われるか注意が必要です。

I0402U	デバッグ情報が含まれています
	Debug information exists

-c オプションにより報告されるメッセージです。当ライブラリファイルにはデバッグ情報を含むモジュールが含まれています。ライブラリファイル中のデバッグ情報は、-O オプションで取り除くことができます。

I0407U	下位互換 CPU タイプのオブジェクトがあります (ファイル名)
	Lower compatible cpu type object (ファイル名)

ここで示されたファイルは、ライブラリファイルと下位互換性のある異なる CPU タイプのモジュールを含むオブジェクトファイル、あるいはライブラリファイルです。

I0409U	Softune V3 のオブジェクトです (ファイル名)
	Softune V3 type object (ファイル名)

ここで示されたファイルは、SOFTUNE V3 言語ツールで作成されたモジュールです。

I0410U	Softune V5 のオブジェクトです (ファイル名)
	Softune V5 type object (ファイル名)

ここで示されたファイルは、SOFTUNE V5 言語ツールで作成されたモジュールです。

I0411U	旧形式のライブラリファイルです (ファイル名)
	This is an old-format library file (ファイル名)

ここで示されたファイルは、SOFTUNE V3 または V5 言語ツールで作成されたライブラリです。自動的に編集前のライブラリファイルのバックアップをとります。拡張子は .bak です。

W1401U	'-pl' 指定を無視します
	Ignore '-pl' option

リスト行数指定 (-pl オプション) がありますが、対象となるマップリストが出力指定されていません。この指定を無視します。

W1402U	'-pw' 指定を無視します
	Ignore '-pw' option

リスト桁数指定 (-pw オプション) がありますが、対象となるマップリストが出力指定されていません。この指定を無視します。

W1403U	'-g' 指定を無視します
	Ignore '-g' option

デバッグ情報付ライブラリ作成指定 (-g オプション) は、モジュールの追加指定 (-a オプション) あるいは置換指定 (-r オプション) を行うときのみ意味を持ちます。この指定を無視します。

W1404U	行うべき処理がありませんでした
	Nothing to operate

ライブラリファイルの変更または、モジュールの抽出処理がありませんでした。

W1405U	削除するモジュールがありませんでした (モジュール名)
	Module not exists to delete (モジュール名)

-d オプションで指定のモジュールが、ライブラリファイルに含まれていません。

-m オプションで登録されているモジュール名を確かめてください。

W1406U	抽出するモジュールがありませんでした (モジュール名)
	Module not exists to extract (モジュール名)

-x オプションで指定のモジュールが、ライブラリファイルに含まれていません。

-m オプションで登録されているモジュール名を確かめてください。

W1412U	互換性のない CPU タイプのモジュールがあります (ファイル名)
	Uncompatible cpu type module (ファイル名)

ここで示されるファイルは互換性のないモジュールです。

入力されるモジュールは、ターゲット CPU が同じか互換 CPU でなくてはなりません。

本メッセージは、以下の場合に出力されます。

- FR80 用ライブラリファイル作成時に FR 用のモジュールを入力した場合

警告の出力されたファイルについては、非互換の命令が使われていないことをご確認してください。

FR と FR80 の非互換命令の詳細は「9.9 FR 用オブジェクトと FR80 用オブジェクトの混在」を参照してください。

E4402U	モジュール名の重複がありました (ファイル名 モジュール名)
	Duplicated module name (ファイル名 モジュール名)

既に登録されているモジュールと同じ名前のモジュールを追加登録しようとした。

1つのライブラリ中にモジュールの重複はできませんので、ここで示されるモジュールは登録しません。置き換えるならば、-r オプションを使用してください。

E4403U	外部定義シンボルの重複がありました (ファイル名 シンボル名)
	Duplicated external definition symbol name (ファイル名 シンボル名)

登録されているモジュールには、外部定義シンボルが含まれていますが、ここで示されたシンボルは既にライブラリに登録されています。

1つのライブラリ中に外部定義シンボルの重複はできませんので、ここで示されるシンボルを含むモジュールは登録しません。

E4404U	登録できないモジュール形式です (ファイル名)
	Invalid module : type (ファイル名)

ライブラリファイルに登録できるのは、アセンブラが出力するオブジェクトモジュール形式だけです。リンカ出力の絶対形式および相対形式ロードモジュールは登録できません。

E4405U	作成ツールの異なるモジュールがあります (ファイル名)
	Invalid module : conflict tool name (ファイル名)

このライブラリアンが処理できるファミリ用アセンブラ出力のオブジェクトファイルではありません。

E4406U	コンパイルモデルの異なるモジュールがあります (ファイル名)
	Invalid module : conflict compile model (ファイル名)

コンパイルモデル (メモリモデルなど) が異なるモジュールを同じライブラリに登録できません。

E4407U	ターゲット CPU の異なるモジュールがあります (ファイル名)
	Invalid module : conflict CPU type (ファイル名)

ターゲット CPU の異なるモジュールを同じライブラリに登録できません。または、ターゲット CPU と異なるライブラリファイルが指定されています。

E4412U	互換性のない CPU タイプのモジュールがあります (ファイル名)
	Uncompatible cpu type module (ファイル名)

ここで示されるファイルは互換性のないモジュールです。

入力されるモジュールは、ターゲット CPU が同じか互換 CPU でなくてはなりません。

本メッセージは、以下の場合に出力されます。

- FR 用ライブラリファイル作成時に FR80 用のモジュールを入力した場合
- FR80 用ライブラリファイル作成時に FR 用のモジュールを入力した場合

FR と FR80 の非互換命令の詳細は「9.9 FR 用オブジェクトと FR80 用オブジェクトの混在」を参照してください。

E4470U	CPU 情報ファイルが見つかりません (ファイル名)
	CPU information file not found (ファイル名)

-cpu オプションによる指定ターゲットの CPU 情報ファイルが見つかりません。

下記ファイルが見つからない場合に検出されます。

- %FETOOL%\LIB\911\911.CSV
- %FETOOL%\LIB\911\cpu_info*.CSV

E4471U	CPU 情報が見つかりません (ファイル名)
	CPU information not found (ファイル名)

-cpu オプションによる指定ターゲットの CPU 情報が CPU 情報ファイル内に見つかりません。

下記ファイル内に指定ターゲットの CPU 情報が見つからない場合に検出されます。

- %FETOOL%\LIB\911\911.CSV
- %FETOOL%\LIB\911\cpu_info*.CSV

F9001U	メモリが足りません
	Insufficient memory

プログラム実行のためのメモリが不足しています。

F9015U	ファイルがオープンできません (ファイル名)
	File open error (ファイル名)

ここで示されたファイルが出力ファイルである場合、1つのディレクトリが管理できるファイルの制限数を超過している可能性があります。不要なファイルを削除するか移動してください。

F9016U	ファイルからの読み込みができません (ファイル名)
	File read error (ファイル名)

リード許可のないファイルである可能性があります。

F9017U	ファイルへの書き込みができません (ファイル名)
	File write error (ファイル名)

書き込み許可のない同名のファイルが存在している可能性があります。

または、ここで示されているファイルを書き込んでいるディスクの容量が不足している可能性があります。ディスクに空き容量を作ってもう一度ライブラリアンを実行してください。

F9021U	コマンドラインの指定が多すぎます
	Too many options

コマンドラインに指定した (オプションファイル中の指定も含む)、入力ファイル名やオプションが多すぎます。

ライブラリアンの起動を複数回に分けてください。

F9022U	オプション名の指定に誤りがあります (オプション)
	Illegal option name (オプション)

オプション名の指定に誤りがあります。コマンドラインを訂正して再起動してください。

F9023U	パラメータの指定に誤りがあります (オプション)
	Illegal option parameter (オプション)

このオプションで指定すべきパラメータに誤りがあります。

F9026U	値が範囲外です (値)
	Specified value out of range (値)

-pl または -pw オプションのパラメータで指定可能な値ではありません。

-help オプションまたは本マニュアルで確認してください。

F9027U	オプションファイル中にオプションファイルの指定があります
	Option file nested

オプションファイルはネストできません。オプションファイル中に記述されている -f オプションを削除してください。

F9033U	ファイルの形式が正しくありません (ファイル名)
	Illegal file format (ファイル名)

このメッセージは次の場合に出力されます。

- ライブラリファイルの形式が正しくない。
- ライブラリファイル中のオブジェクトモジュールの形式が正しくない。
- 入力ファイルが絶対形式ロードモジュールである。
- 入力モジュールの内容が正しくない。
- 相対形式のアセンブルリストファイルが正しくない。
- CPU 情報ファイルの形式が正しくない。

ライブラリファイル名の指定がありません。

F9035U	ライブラリファイル名の指定がありません
	Missing library file name

ライブラリファイル名を指定してください。

F9036U	ライブラリファイル名が複数指定されています (ファイル名)
	Multiple library file name specified (ファイル名)

ライブラリファイルは 1 つだけ指定できます。ここに示されたファイル名またはこれにより前に指定したライブラリファイル名のどちらか 1 つを選択してください。

F9045U	'-O' 指定時は他のオプションの指定はできません
	'-O' option conflict with another option

-O オプション指定時は、ほかのオプションは指定できません。

F9046U	'-c' 指定時は他のオプションの指定はできません
	'-c' option conflict with another option

-c オプションは、他のオプションと組み合わせないでください。

F9047U	該当するファイルが見つかりませんでした (ファイル名)
	No match ファイル名 argument

ワイルドカードによるファイル指定が行われましたが、該当するファイルがありませんでした。

F9052U	'-cpu' オプションの指定がありません
	Missing '-cpu' option

-cpu オプションによるターゲット CPU の指定がありません。
-cpu オプションでターゲット CPU を指定する必要があります。

F9056U	CPU 情報ファイルのバージョンが適合しません
	Mismatch CPU information file version

CPU 情報ファイルが古く適合しません。
最新の CPU 情報ファイルを入手してください。

F9998U	
	File open failed (ファイル名)

ライブラリアンが使用するメッセージファイルがオープンできませんでした。
エラーメッセージファイル (lkt_a.msg, lkt_e.msg) は、決められたディレクトリに格納してください。

F9999U	プログラム内部エラーが発生しました (識別情報)
	Internal error (識別情報)

このエラーが出た場合は、ただちに営業部門へ連絡してください。

■ コンバータのエラーメッセージ

I0501U	スタートアドレスレコードを読み飛ばしました
	Skip start address record

HEX フォーマットにスタートアドレスレコードが含まれていましたが、不要なので読み飛ばしました。コンバータの処理には影響なく正しくコンバートは行われます。

W1501U	このファイルは警告レベルエラーを含んでいます (ファイル名)
	File include WARNING level error (ファイル名)

入力に指定したファイルは、リンク時に警告レベルのエラーがあったものです。問題ないかどうかを確認のうえ使用してください。

W1502U	出力指定フォーマットで表せないアドレスです (アドレス)
	Unable to convert address (アドレス)

コンバート先のフォーマットでは、表現できないアドレスのデータがコンバート元のファイルに含まれています。ここで示されたアドレスを超えた部分のデータはすべて捨てられます。

コンバート先のフォーマットを変更してください。

W1503U	-I16 指定時に -entry オプションが指定されました
	-entry option was specified at the time of -I16 specification

-I16 指定時にスタートアドレス出力指定オプションが指定されました。

W1504U	入力ファイルにスタートアドレス情報がありません
	Start address information is not in an input file

入力ロードモジュールファイルにスタートアドレス情報がありません。f2hs は、スタートアドレスレコードを出力せずに HEX フォーマットを出力します。

F9001U	メモリが足りません
	Insufficient memory

プログラム実行のためのメモリが不足しています。

F9011U	ファイルが見つかりません (ファイル名)
	Input file not found (ファイル名)

入力ファイルに指定したファイルが見つかりません。

F9015U	ファイルがオープンできません (ファイル名)
	File open error (ファイル名)

ここで示されたファイルが出力ファイルである場合、1つのディレクトリが管理できるファイルの制限数を超過している可能性があります。不要なファイルを削除するか移動してください。

F9016U	ファイルから読み込みができません (ファイル名)
	File read error (ファイル名)

リード許可のないファイルであるか、もしくはハードウェアに問題があることが考えられます。

F9017U	ファイルへの書き込みができません (ファイル名)
	File write error (ファイル名)

ここで示されているファイルを書き込んでいるディスクの容量が不足しています。ディスクに空き容量を作ってもう一度コンバータを実行してください。

F9021U	コマンドラインの指定が多すぎます
	Too many options

コマンドラインに指定 (オプションファイル中の指定も含む) した、入力ファイル名やオプションが多すぎます。

F9022U	オプション名の指定に誤りがあります (オプション)
	Illegal option name (オプション)

コンバータで使えるオプションではありません。-help オプションまたは本マニュアルで確認してください。

F9023U	パラメータの指定に誤りがあります (オプション)
	Illegal option parameter (オプション)

このオプションで指定すべきパラメータに誤りがあります。

F9026U	値が範囲外です (値)
	Specified value out of range (値)

オプションで許される範囲外の値が指定されています。

F9027U	オプションファイル中にオプションファイルの指定があります
	Option file nested

オプションファイルはネストできません。オプションファイル中に記述されている -f オプションを削除してください。

F9028U	指定したアドレスは大きすぎます (オプション : s= アドレス 1e= アドレス 2)
	Specified address too large (オプション : s= アドレス 1e= アドレス 2)

オプションのパラメータで指定したアドレスは , コンバート先のファイルフォーマットでは表せません。指定し直してください。

F9029U	指定アドレスの大小関係が逆です (オプション : s= アドレス 1e= アドレス 2)
	Start address opposite to end one (オプション : s= アドレス 1e= アドレス 2)

オプションのパラメータで指定したアドレスは , スタートアドレスとエンドアドレスの大小関係が逆です。

F9030U	入力ファイル名の指定がありません
	Missing input file name

コンバート元の入力ファイル名を指定してください。

F9031U	入力ファイル名は既に指定されています (ファイル名)
	Multiple input file name (ファイル名)

入力ファイルは 1 つだけ指定できます。ここに示されたファイル名またはこれより前に指定したファイル名のどちらか 1 つを入力ファイルとしてください。

F9032U	入出力ファイル名が同じです (ファイル名)
	Output file name same as input one (ファイル名)

ここに示された出力ファイル名は , 入力ファイル名と同じ名称ですので処理できません。

F9033U	ファイルの形式が正しくありません (ファイル名)
	Illegal file format (ファイル名)

入力ファイルは、処理しようとしているオブジェクト形式ではありません。
異なる形式のファイルを入力したか、もしくはファイルが壊れています。

F9034U	絶対形式でないファイルは扱えません (ファイル名)
	Not absolute load module file (ファイル名)

リンカ出力の絶対形式ロードモジュールではないファイルを入力しました。リンカで絶対形式にしてからコンバータで使用してください。

F9048U	出力範囲が指定されていません
	Missing output range

出力範囲の指定が行われていません。
出力範囲指定 (-ran) を指定してください。

F9049U	出力範囲のサイズが限界値を超えています
	Output range exceeded

出力範囲が限界値を超えています。

F9050U	出力ファイルが他の出力ファイルと同じです (ファイル名)
	Output file name same as other output one (ファイル名)

出力ファイル名に、ほかの出力ファイル名と同じ名前が指定されています。
出力ファイル名を変更してください。

F9051U	ファイル名が長すぎます (ファイル名)
	File name too long (ファイル名)

ここに示された出力ファイル名の指定が長すぎるため処理ができません。
ファイル名の指定を短くしてください。

F9998U	
	File open failed (ファイル名)

コンバータが使用するメッセージファイルがオープンできませんでした。
エラーメッセージファイル (lkt_a.msg, lkt_e.msg) は、決められたディレクトリに格納してください。

F9999U	プログラム内部エラーが発生しました (識別情報)
	Internal error (識別情報)

このエラーが出た場合は、ただちに営業部門へ連絡してください。

付録 B HEX フォーマット

HEX フォーマットについて説明します。

- HEX8 フォーマット : 8 ビット用に設定されたフォーマット
 - HEX16 フォーマット : 16 ビット用に拡張されたフォーマット
 - HEX32 フォーマット : 32 ビット用に拡張されたフォーマット
-

■ HEX フォーマット

- 一般形式 (「付録 B.1 一般形式」参照)
- データレコード (HEX8/HEX16/HEX32) タイプ : 00
(「付録 B.2 データレコード (HEX8/HEX16/HEX32) タイプ : 00」参照)
- エンドレコード (HEX8/HEX16/HEX32) タイプ : 01
(「付録 B.3 エンドレコード (HEX8/HEX16/HEX32) タイプ : 01」参照)
- 拡張セグメントアドレスレコード (HEX16/HEX32) タイプ : 02
(「付録 B.4 拡張セグメントアドレスレコード (HEX16/HEX32) タイプ : 02」参照)
- スタートセグメントアドレスレコード (HEX16/HEX32) タイプ : 03
(「付録 B.5 スタートセグメントアドレスレコード (HEX16/HEX32) タイプ : 03」参照)
- 拡張リニアアドレスレコード (HEX32) タイプ : 04
(「付録 B.6 拡張リニアアドレスレコード (HEX32) タイプ : 04」参照)
- スタートリニアアドレスレコード (HEX32) タイプ : 05
(「付録 B.7 スタートリニアアドレスレコード (HEX32) タイプ : 05」参照)

B.1 一般形式

HEX フォーマットは, (a) ~ (f) で示す 6 つのフィールドで構成されます。
各フィールドは, ASCII コードで設定されます。(g) については, 後述します。

■ 一般形式

図 B-1 一般形式

:	l1	l2	a1	a2	a3	a4	t1	t2	d1	d2	d3	d4	d*	d*	d*	d*	s1	s2	
(a)	(b)		(c)				(d)	(e)									(f)	(g)	

(a) :

レコードの開始を示し, ":" (0x3A) の文字です。

(b) :

(e) のデータ部のバイト数を示します。

実際の 1 バイトデータを当フォーマットでは 2 バイトの ASCII コードで示しますので, 上記の図では, d1, d2 を 1 とカウントします。

l1 が上位桁, l2 が下位桁で, 0 ~ 255 の値が設定できます。

ASCII で "00" ~ "FF", 16 進で "0x3030" ~ "0x4646" になります。

(c) :

(e) の内容がオブジェクトデータであるとき, 最初のデータに割り付けられたアドレスを示します。

a1 が上位桁, a4 が下位桁で, 0 ~ 65535 の値が設定できます。

ASCII で "0000" ~ "FFFF", 16 進で "0x30303030" ~ "0x46464646" になります。

(d) :

レコードのタイプを示します。

00 : データレコード (HEX8/HEX16/HEX32 形式)

01 : エンドレコード (HEX8/HEX16/HEX32 形式)

02 : 拡張セグメントアドレスレコード (HEX16/HEX32 形式)

03 : スタートセグメントアドレスレコード (HEX16/HEX32 形式)

04 : 拡張リニアアドレスレコード (HEX32 形式)

05 : スタートリニアアドレスレコード (HEX32 形式)

(e) :

(d) のレコードタイプごとに異なりますので, 各レコードの説明で示します。

(f) :

チェックサムです。(b) (c) (d) (e) の ASCII で表された 2 バイトのデータを 1 バイトの 16 進で表し, 各バイトを符号なしでオーバーフローを無視して加算します。

その結果の 2 の補数を求め, 2 バイトの ASCII にして設定します。

s1 が上位桁になります。

2 の補数：各ビットの 0 を 1 に, 1 を 0 にした値に 1 を加えた値

(g) :

一般に制御コード (CR, LF 等) が付加されます。

このフィールドのデータは, (a) の開始文字 ":" が来るまで読み飛ばします。

(a)(b)(c)(d)(f) のフィールドは必ず存在しますので, 1 レコードの長さは最低 11 バイト, 最大 521 バイトになります。

【例】

: 0200000020036C6	拡張セグメントアドレスレコード
: 0600100090D9226BB4FD43	データレコード
: 0400000035162000541	スタートアドレスレコード
: 000000001FF	エンドレコード

B.2 データレコード (HEX8/HEX16/HEX32) タイプ : 00

d1, d2 が (c) が示すアドレスのバイトデータになり , d3, d4 は , 次のアドレスのバイトデータです。

■ データレコード (HEX8/HEX16/HEX32)

図 B.2-1 データレコード (HEX8/HEX16/HEX32)

:	11	12	a1	a2	a3	a4	0	0	d1	d2	d3	d4	d*	d*	d*	d*	s1	s2	
(a)	(b)		(c)				(d)		(e)								(f)		(g)

(a) (b) (c) (d) (f) (g) については , 一般形式の説明を参照してください。

(e) は , オブジェクトデータで , 実際の 1 バイトデータを 2 バイトの ASCII で表現します。

上記の図では , (c) が示すアドレスのバイトデータは , d1, d2 になります。

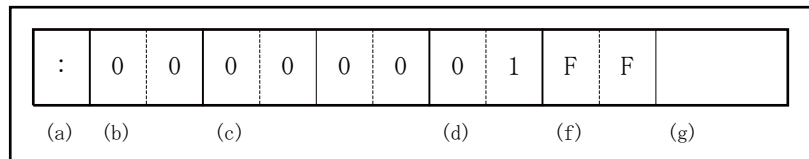
同様に d3, d4 は , 次のアドレスのバイトデータです。

B.3 エンドレコード (HEX8/HEX16/HEX32) タイプ : 01

エンドレコードは , 00000001FF で固定です。
最後のレコードとして 1 つだけ存在します。

■ エンドレコード (HEX8/HEX16/HEX32)

図 B.3-1 エンドレコード (HEX8/HEX16/HEX32)



(e) のフィールドは , 存在しません。したがって , (b) は 0 を設定します。

(c) は未使用ですが値としては 0 を設定します。

B.4 拡張セグメントアドレスレコード (HEX16/HEX32)

タイプ : 02

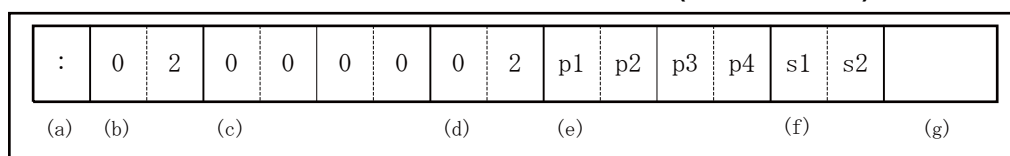
拡張セグメントアドレスレコードが現れると、次の拡張セグメントアドレスレコードが現れるまで、以降のデータレコードの各バイトデータは、次の式に従ってアドレスが計算されます。

$$((PA \times 0x10) + ((DA + DP) \bmod 0x10000)) \bmod 0x100000$$

- PA : 当レコードの (e) フィールドの値です。
- DA : データレコードの (c) フィールドの値で、ここでは相対アドレス扱いになります。
- DP : データレコードの (e) フィールド中でのデータ位置の先頭データを 0 として表した値です。

■ 拡張セグメントアドレスレコード (HEX16/HEX32)

図 B.4-1 拡張セグメントアドレスレコード (HEX16/HEX32)



(e) :

パラグラフアドレスで、実際の 2 バイトデータを 4 バイトの ASCII で表現します。
上記の図では、p1 が上位桁になります。

(c) :

未使用ですが値としては 0 を設定します。

拡張セグメントアドレスレコードが現れると、次の拡張セグメントアドレスレコードが現れるまで、以降のデータレコードの各バイトデータは、次の式に従ってアドレスが計算されます。

$$[(PA \times 0x10) + [(DA + DP) \bmod 0x10000]] \bmod 0x100000$$

- PA : 当レコードの (e) フィールドの値です。
- DA : データレコードの (c) フィールドの値で、ここでは相対アドレス扱いになります。
- DP : データレコードの (e) フィールド中でのデータ位置の先頭データを 0 として表した値です。

これは、i8086 での物理アドレスを求めるのと同じ方法であり、拡張セグメントアドレスレコードの追加により、20 ビットのアドレス値まで表現できます。

拡張セグメントアドレスレコードよりも前に現れた、データレコードについては、上記 PA を 0x0000 としてアドレス計算が行われます。

B.5 スタートセグメントアドレスレコード (HEX16/HEX32)

タイプ : 03

プログラムの実行開始番地を指定するためのレコードです。

スタートアドレスは、次の式に従って計算されます。

$$((PA \times 0x10) + IP) \text{ MOD } 0x100000$$

- PA : 当レコードの (e1) フィールドの値です。
- IP : 当レコードの (e2) フィールドの値です。

■ スタートセグメントアドレスレコード (HEX16/HEX32)

図 B.5-1 スタートセグメントアドレスレコード (HEX16/HEX32)

:	0	4	0	0	0	0	0	3	p1	p2	p3	p4	i1	i2	i3	i4	s1	s2	
(a)	(b)		(c)				(d)		(e)								(f)		(g)

(e) は、上記の図のように 2 つのフィールドに分かれ、(e1) 部にパラグラフアドレス、(e2) 部にオフセット値が設定されます。

p1, i1 がそれぞれ上位桁になります。

(c) は未使用ですが 0 を設定します。

スタートアドレスは、次の式に従って計算されます。

$$[(PA \times 0x10) + IP] \text{ MOD } 0x100000$$

- PA : 当レコードの (e1) フィールドの値です。
- IP : 当レコードの (e2) フィールドの値です。

当レコードの出現箇所は、エンドレコード以前ならどこでもかまいません。

出現回数は、0 または 1 になります。

B.6 拡張リニアアドレスレコード (HEX32)

タイプ : 04

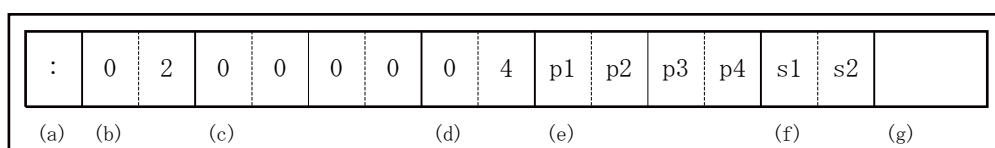
拡張リニアアドレスレコードが現れると、次の拡張リニアアドレスレコードが現れるまで、以降のデータレコードの各バイトデータは、次の式に従ってアドレスが計算されます。

$$((PA \times 0x10000) + ((DA + DP) \text{ MOD } 0x10000)) \text{ MOD } 0x100000000$$

- PA : 当レコードの (e) フィールドの値です。
- DA : データレコードの (c) フィールドの値で、ここでは相対アドレス扱いになります。
- DP : データレコードの (e) フィールド中でのデータ位置の先頭データを 0 として表した値です。

■ 拡張リニアアドレスレコード (HEX32)

図 B.6-1 拡張リニアアドレスレコード (HEX32)



(e):

パラグラフアドレスで、実際の 2 バイトデータを 4 バイトの ASCII で表現します。
上記の図では、p1 が上位桁になります。

(c):

未使用ですが値としては 0 を設定します。

リニアアドレスレコードが現れると、次の拡張リニアアドレスレコードが現れるまで、以降のデータレコードの各バイトデータは、次の式に従ってアドレスが計算されます。

$$[(PA \times 0x10000) + [(DA + DP) \text{ MOD } 0x10000]] \text{ MOD } 0x100000000$$

- PA : 当レコードの (e) フィールドの値です。
- DA : データレコードの (c) フィールドの値で、ここでは相対アドレス扱いになります。
- DP : データレコードの (e) フィールド中でのデータ位置の先頭データを 0 として表した値です。

拡張リニアアドレスレコードの追加により、32 ビットのアドレス値まで表現できます。
拡張リニアアドレスレコードよりも前に現れた、データレコードについては、上記 PA を 0x0000 としてアドレス計算が行われます。

B.7 スタートリニアアドレスレコード (HEX32) タイプ : 05

プログラムの実行開始番地を指定するためのレコードです。

■ スタートリニアアドレスレコード (HEX32)

図 B.7-1 スタートリニアアドレスレコード (HEX32)

:	0	4	0	0	0	0	0	5	e1	e2	e3	e4	e5	e6	e7	e8	s1	s2	
(a)	(b)	(c)		(d)				(e)	(f)								(g)		

(e) は , 32bit の実行開始アドレスが設定されます。
e1 が上位桁になります。
(c) は未使用ですが 0 を設定します。
当レコードの出現箇所は , エンドレコード以前ならどこでもかまいません。
出現回数は , 0 または 1 になります。

付録 C S レコード形式

S レコードフォーマットは、すべて "S" (0x53) の文字で始まり、S0 から S9 までの 8 タイプがあります。(S4 と S6 は使用しません)

■ S レコード形式

- S0 タイプ (ヘッダレコード) (「付録 C.1 S0 タイプ (ヘッダレコード)」参照)
- S1 タイプ (データレコード :2 バイトアドレス)
(「付録 C.2 S1 タイプ (データレコード :2 バイトアドレス)」参照)
- S2 タイプ (データレコード :3 バイトアドレス)
(「付録 C.3 S2 タイプ (データレコード :3 バイトアドレス)」参照)
- S3 タイプ (データレコード :4 バイトアドレス)
(「付録 C.4 S3 タイプ (データレコード :4 バイトアドレス)」参照)
- S5 タイプ (レコード数管理レコード)
(「付録 C.5 S5 タイプ (レコード数管理レコード)」参照)
- S7 タイプ (ターミネータレコード)
(「付録 C.6 S7 タイプ (ターミネータレコード)」参照)
- S8 タイプ (ターミネータレコード)
(「付録 C.7 S8 タイプ (ターミネータレコード)」参照)
- S9 タイプ (ターミネータレコード)
(「付録 C.8 S9 タイプ (ターミネータレコード)」参照)

C.1 S0 タイプ (ヘッダレコード)

当レコードは、コメント記述のために使用します。

■ S0 タイプ (ヘッダレコード)

図 C.1-1 S0 タイプ (ヘッダレコード)

S	0	11	12	0	0	0	0	c1	c2	c3	c4		c*	c*	s1	s2	
(a)		(b)		(c)				(d)							(e)		(f)
2バイト		2バイト		4バイト				nバイト							2バイト		

当レコードは、上記の (a) ~ (e) で示す 5 つのフィールドで構成されます。

S0 タイプはヘッダレコードと呼ばれ、S1 ~ S9 の各レコードに先立ち、ファイルの先頭に置かれます。各フィールドは、ASCII コードで設定されます。

(a) :

タイプフィールドで、ASCII コードで "S0"(0x5330) の文字です。

(b) :

(c) (d) (e) のバイト数を示します。

実際の 1 バイトデータを当フォーマットでは 2 バイトの ASCII コードで示しますので、(この部分の文字数 / 2) が設定されます。

11 が上位桁、12 が下位桁で、0 ~ 255 の値が設定できます。

ASCII で "00" ~ "FF", 16 進で "0x3030" ~ "0x4646" になります。

(c) :

使用しませんが ASCII で "0000" を設定します。

(d) :

バージョン管理情報などのメッセージを設定します。

設定方法は、下記の例を参照してください。

(e) :

チェックサムです。

(b) (c) (d) の ASCII で表された 2 バイトのデータを、1 バイトの 16 進で表し、各バイトを符号なしでオーバーフローを無視して加算します。

その結果の 1 の補数を求め、2 バイトの ASCII にして設定します。

s1 が上位桁になります。

- 1 の補数 : 各ビットの 0 を 1 に、1 を 0 にした値

(f) :

一般に制御コード (CR, LF 等) が付加されます。

このフィールドのデータは、(a) の開始文字 "S" がくるまで読み飛ばします。

【例】

S00600004844521B



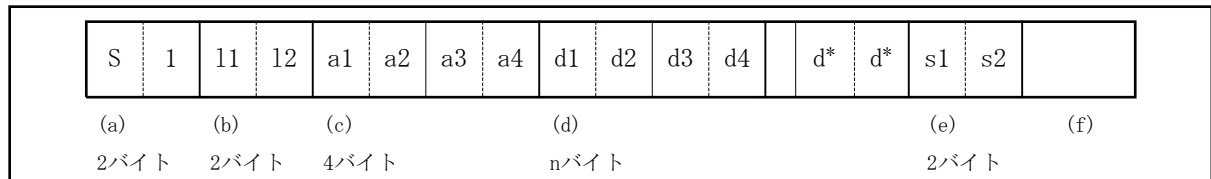
ASCII文字列の“HDR”を示す

C.2 S1 タイプ (データレコード : 2 バイトアドレス)

アドレスが、2 バイトで表せる (0x0000 ~ 0xFFFF) オブジェクトデータを格納するレコードです。

■ S1 タイプ (データレコード : 2 バイトアドレス)

図 C.2-1 S1 タイプ (データレコード : 2 バイトアドレス)



S1 タイプは、上記の (a) ~ (e) で示す 5 つのフィールドで構成されます。

(a) :

タイプフィールドで、ASCII コードで "S1"(0x5331) の文字です。

(b) :

(c) (d) (e) のバイト数を示します。

(「C.1 S0 タイプ (ヘッダレコード)」の説明を参照してください。)

(c) :

(d) の最初のデータに割り付けられたアドレスを示します。

a1 が上位桁、a4 が下位桁で、0 ~ 65535 の値が設定できます。

ASCII で "0000" ~ "FFFF", 16 進で "0x30303030" ~ "0x46464646" になります。

(d) :

オブジェクトデータで、実際の 1 バイトデータを 2 バイトの ASCII で表現します。

上記の図では、d1, d2 が (c) が示すアドレスのバイトデータになります。

同様に d3, d4 は、次のアドレスのバイトデータです。

(e) :

チェックサムです。

(「C.1 S0 タイプ (ヘッダレコード)」の説明を参照してください。)

(f) :

一般に制御コード (CR, LF 等) が付加されます。

(「C.1 S0 タイプ (ヘッダレコード)」の説明を参照してください。)

C.3 S2 タイプ (データレコード : 3 バイトアドレス)

S2 タイプは, (c) フィールドの大きさが S1 タイプと異なるもので, 3 バイトアドレスが必要なオブジェクトデータを格納するためのレコードです。

■ S2 タイプ (データレコード : 3 バイトアドレス)

図 C.3-1 S2 タイプ (データレコード : 3 バイトアドレス)

S	2	11	12	a1	a2	a3	a4	a5	a6	d1	d2	d3	d4		d*	d*	s1	s2	
(a)	(b)	(c)								(d)							(e)		(f)
2バイト	2バイト	6バイト								nバイト							2バイト		

S2 タイプは, 上記の (c) フィールドの大きさが S1 タイプと異なるもので, 3 バイトアドレスが必要なオブジェクトデータを格納するためのレコードです。

C.4 S3 タイプ (データレコード : 4 バイトアドレス)

S3 タイプは , (c) フィールドの大きさが S1 タイプと異なるもので , 4 バイトアドレスが必要なオブジェクトデータを格納するためのレコードです。

■ S3 タイプ (データレコード : 4 バイトアドレス)

図 C.4-1 S3 タイプ (データレコード : 4 バイトアドレス)

S	3	11	12	a1	a2	a3	a4	a5	a6	a7	a8	d1	d2	d3	d4		d*	d*	s1	s2			
(a)		(b)		(c)								(d)								(e)		(f)	
2バイト		2バイト		8バイト								nバイト								2バイト			

S3 タイプは , 上記の (c) フィールドの大きさが S1 タイプと異なるもので , 4 バイトアドレスが必要なオブジェクトデータを格納するためのレコードです。

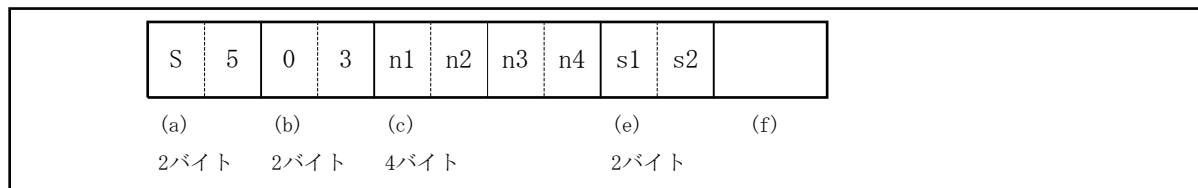
C.5 S5 タイプ (レコード数管理レコード)

ファイル中に含まれるレコードの数を設定します。

当レコードは、なくてもかまいません。出現場所は S0 と S9 の間で任意です。

■ S5 タイプ (レコード数管理レコード)

図 C.5-1 S5 タイプ (レコード数管理レコード)



S5 タイプは、上記の (a) ~ (e) で示す 4 つのフィールドで構成されます。

(a) :

タイプフィールドで、ASCII コードで "S5"(0x5335) の文字です。

(b) :

(c) (e) のバイト数を示します。

(「C.1 S0 タイプ (ヘッダレコード)」の説明を参照してください。)

(c) :

ファイル中のデータレコード (S1, S2, S3) の数を示します。

n1 が上位桁, n4 が下位桁で、0 ~ 65535 の値が設定できます。

ASCII で "0000" ~ "FFFF", 16 進で "0x30303030" ~ "0x46464646" になります。

(d) :

フィールドはありません。

(e) :

チェックサムです。

(「C.1 S0 タイプ (ヘッダレコード)」の説明を参照してください。)

(f) :

一般に制御コード (CR, LF 等) が付加されます。

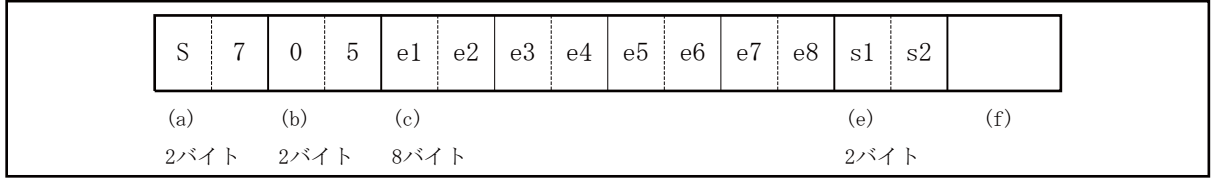
(「C.1 S0 タイプ (ヘッダレコード)」の説明を参照してください。)

C.6 S7 タイプ (ターミネータレコード)

ファイルの終了を表すレコードであり，実行開始アドレス情報も含んでいます。
当レコードは，ファイルの最後に置かれます。
実行開始アドレスの表現に 4 バイト必要な場合のターミネータレコードです。

■ S7 タイプ (ターミネータレコード)

図 C.6-1 S7 タイプ (ターミネータレコード)



S7 タイプは，上記の (a) ~ (e) で示す 4 つのフィールドで構成されます。

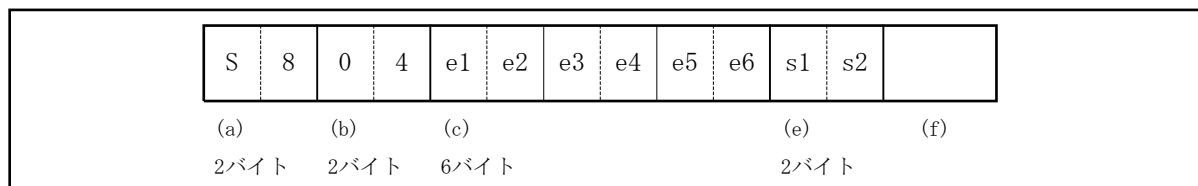
- (a) :
タイプフィールドで，ASCII コードで "S7"(0x5337) の文字です。
- (b) :
(c) (e) のバイト数を示します。"05" 固定となります。
- (c) :
実行開始アドレスを示します。
e1 が上位桁，e8 が下位桁になります。
- (d) :
フィールドはありません。
- (e) :
チェックサムです。
- (f) :
一般に制御コード (CR, LF 等) が付加されます。

C.7 S8 タイプ (ターミネータレコード)

S8 タイプは、(c) フィールドの大きさが S7 タイプと異なるもので、実行開始アドレスの表現に 3 バイトが必要な場合のターミネータレコードです。

■ S8 タイプ (ターミネータレコード)

図 C.7-1 S8 タイプ (ターミネータレコード)



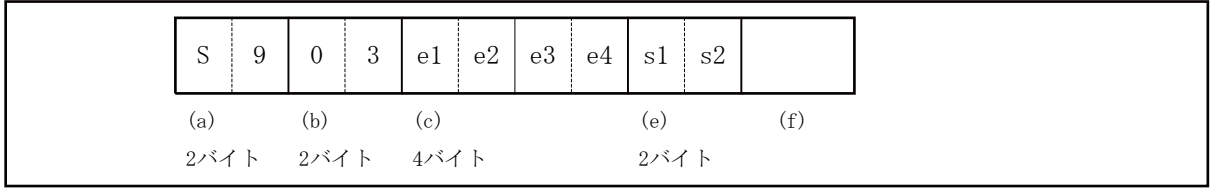
S8 タイプは、上記の (c) フィールドの大きさが S7 タイプと異なるもので、実行開始アドレスの表現に 3 バイトが必要な場合のターミネータレコードです。

C.8 S9 タイプ (ターミネータレコード)

S9 タイプは、(c) フィールドの大きさが S7 タイプと異なるもので、実行開始アドレスの表現に 2 バイトが必要な場合のターミネータレコードです。

■ S9 タイプ (ターミネータレコード)

図 C.8-1 S9 タイプ (ターミネータレコード)



S9 タイプは、上記の (c) フィールドの大きさが S7 タイプと異なるもので、実行開始アドレスの表現に 2 バイトが必要な場合のターミネータレコードです。

付録 D リンカのオプション一覧表

付表 D-1 にリンカのオプション一覧を示します。

■ リンカのオプション一覧

付表 D-1 リンカのオプション一覧表 (1 / 2)

機 能	オプション	備 考
出力ロードモジュールファイル名指定	-o	デフォルト
デバッグ情報出力指定	-g	
デバッグ情報削除指定	-Xg	デフォルト
絶対形式ロードモジュール出力指定	-a	デフォルト
相対形式ロードモジュール出力指定	-r	
バディングデータ指定	-p	デフォルト 0
ROM 領域のフィル指定	-fill	
外部シンボル出力指定	-symtab	
外部シンボル出力抑止指定	-Xsymtab	デフォルト
マップリストファイル名指定	-m	デフォルト
マップリスト出力抑止指定	-Xm	
リスト表示の名前省略解除	-dt	
メモリ使用情報リスト出力指定	-mmi	
リスト表示のデマングルシンボル名の出力抑止指定	-Xdemangle	
リスト表示のデマングルシンボル名の出力指定	-demangle	デフォルト
リスト行の桁数指定	-pw	デフォルト 80
リスト 1 ページの行数指定	-pl	デフォルト 0
ROM 領域のチェックサム指定	-cs	
警告メッセージ出力レベル指定	-w	
ROM 領域指定	-ro	
RAM 領域指定	-ra	
セクション配置	-sc	
セクショングループ指定	-gr	
バックリンク指定	-pk	
自動配置指定	-AL	デフォルト 0
検索ライブラリファイル指定	-l	
ライブラリ検索パス指定	-L	
シンボル個別のライブラリ指定	-el	
ライブラリ検索抑止指定	-nl	
デフォルトライブラリ検索抑止指定	-nd	
エントリアドレス指定	-e	
外部シンボル値の仮設定	-df	
ターゲット CPU 指定	-cpu	必須

付表 D-1 リンカのオプション一覧表 (2 / 2)

機 能	オプション	備 考
CPU 情報ファイル指定	-cif	
オブジェクト混在チェックレベル指定	-omcl	デフォルト 1
デバッグ情報存在チェック抑止指定	-NCI0302LIB	
内蔵 ROM/RAM 領域の自動設定	-set_rora	デフォルト
内蔵 ROM/RAM 領域自動設定の抑止指定	-Xset_rora	
ユーザ指定領域のチェック指定	-check_rora	
ユーザ指定領域のチェック抑止指定	-Xcheck_rora	デフォルト
セクション配置領域チェック指定	-check_locate	
セクション配置領域チェック抑止指定	-Xcheck_locate	デフォルト
サイズ 0 のセクション配置チェック指定	-check_size0_sec	
サイズ 0 のセクション配置チェック抑止指定	-Xcheck_size0_sec	デフォルト
プレリンク処理の抑止指定	-XPLNK	
相対アセンブルリスト入力ディレクトリ指定	-alin	
絶対形式アセンブルリスト出力ディレクトリ指定	-alout	
絶対形式アセンブルリスト出力指定	-als	
絶対形式アセンブルリスト出力モジュール指定	-alsf	
絶対形式アセンブルリスト出力抑止指定	-Xals	
ROM/RAM, ARRAY リスト出力指定	-alr	
ROM/RAM, ARRAY リスト出力モジュール指定	-alrf	
ROM/RAM, ARRAY リスト出力抑止指定	-Xalr	
ROM/RAM, ARRAY リストのシンボルとアドレスの表示位置指定	-na/-an	
外部シンボル相互参照情報リスト出力指定	-xl	
外部シンボル相互参照情報リストファイル名指定	-xlf	
外部シンボル相互参照情報リスト出力抑止指定	-Xxl	
ローカルシンボルリスト出力指定	-sl	
ローカルシンボルリストファイル名の指定	-slf	
ローカルシンボルリスト出力抑止指定	-Xsl	
セクション詳細マップリスト出力指定	-ml	
セクション詳細マップリストファイル名の指定	-mlf	
セクション詳細マップリスト出力抑止指定	-Xml	
デフォルトオプションファイル読み込み抑止指定	-Xdof	* 共通オプション
オプションファイル読み込み指定	-f	* 共通オプション
ヘルプメッセージ表示指定	-help	* 共通オプション
版数 / メッセージ出力指定	-V	* 共通オプション
版数 / メッセージ出力抑止	-XV	* 共通オプション
終了メッセージ表示指定	-cmsg	* 共通オプション
終了メッセージ表示抑止指定	-Xcmsg	* 共通オプション
ワーニング発生時の終了コードを 1 にする指定	-cwno	* 共通オプション
ワーニング発生時の終了コードを 0 にする指定	-Xcwno	* 共通オプション

付録 E ライブラリアンのオプション一覧表

付表 E-1 にライブラリアンのオプション一覧を示します。

■ ライブラリアンのオプション一覧

付表 E-1 ライブラリアンのオプション一覧表

機 能	オプション	備 考
モジュールの追加 (登録)	-a	
モジュールの置換 (登録)	-r	
モジュールの削除	-d	
モジュールの抽出	-x	
リストファイルの出力指定	-m	
リストファイルの出力抑止指定	-Xm	デフォルト
リストファイルの詳細情報の出力指定	-dt	s, d, r, a
リスト 1 ページの行数指定	-pl	デフォルト 60
リスト 1 行の桁数指定	-pw	デフォルト 80
バックアップファイルの作成	-b	
バックアップファイルの作成抑止	-Xb	デフォルト
ライブラリファイルの内容検査	-c	
ファイル内容の最適化	-O	
デバッグ情報の出力指定	-g	
デバッグ情報の出力抑止指定	-Xg	
CPU 情報ファイル指定	-cif	
ターゲット CPU 指定	-cpu	必須
オブジェクト混在チェックレベル指定	-omcl	デフォルト 1
デフォルトオプションファイル読み込み抑止指定	-Xdof	* 共通オプション
オプションファイル読み込み指定	-f	* 共通オプション
ヘルプメッセージ表示指定	-help	* 共通オプション
版数 / メッセージ出力指定	-V	* 共通オプション
版数 / メッセージ出力抑止指定	-XV	* 共通オプション
終了メッセージ出力指定	-cmsg	* 共通オプション
終了メッセージ出力抑止指定	-Xcmsg	* 共通オプション
ワーニング発生時の終了コードを 1 にする指定	-cwno	* 共通オプション
ワーニング発生時の終了コードを 0 にする指定	-Xcwno	* 共通オプション

付録 F オブジェクト形式コンバータのコマンドおよびオプション一覧表

付表 F-1 にオブジェクト形式コンバータのコマンド一覧を、付表 F-2 にオブジェクト形式コンバータのオプション一覧を示します。

■ オブジェクト形式コンバータのコマンド一覧

付表 F-1 オブジェクト形式コンバータのコマンド一覧

コマンド名	機 能	
f2ms	絶対形式ロードモジュール	S フォーマット
f2hs	絶対形式ロードモジュール	HEX フォーマット (HEX8/HEX16/HEX32)
f2is	絶対形式ロードモジュール	HEX8 フォーマット
f2es	絶対形式ロードモジュール	HEX16 フォーマット
m2ms	S フォーマット	S フォーマット (整形)
h2hs	HEX フォーマット	HEX フォーマット (整形)
m2bs	S フォーマット	バイナリデータ (メモリモージ)
h2bs	HEX フォーマット	バイナリデータ (メモリモージ)
m2is	S フォーマット	HEX8 フォーマット
m2es	S フォーマット	HEX16 フォーマット
i2ms	HEX8 フォーマット	S フォーマット
e2ms	HEX16 フォーマット	S フォーマット

■ オブジェクト形式コンバータのオプション一覧

付表 F-2 オブジェクト形式コンバータのオプション一覧

機 能	オプション	備 考
出力ファイル名の指定	-o	
パディングデータ指定	-p	
出力範囲指定	-ran	m2ms, h2hs, m2bs, h2bs のみ 必須
スプリットモード指定	-sp	m2bs, h2bs のみ
スプリットモード抑止指定	-Xsp	m2bs, h2bs のみ
マップリストファイル作成指定	-m	m2bs, h2bs のみ
マップリストファイル作成抑止指定	-Xm	m2bs, h2bs のみ
出力ファイル S フォーマット指定	-S1, -S2, -S3	f2ms, m2ms のみ
出力ファイル HEX フォーマット指定	-I16, -I20, -I32	f2hs, h2hs のみ
スタートアドレスレコード出力指定	-entry	f2hs のみ
スタートアドレスレコード出力抑止指定	-Xentry	f2hs のみ
整形指定	-adjust	f2ms, f2hs のみ
開始アドレス変更指定	-ST	m2ms, h2hs のみ
デフォルトオプションファイル読み込み抑止指定	-Xdof	* 共通オプション
オプションファイル読み込み指定	-f	* 共通オプション
ヘルプメッセージ表示指定	-help	* 共通オプション
版数 / メッセージ出力指定	-V	* 共通オプション
版数 / メッセージ出力抑止	-XV	* 共通オプション
終了メッセージ表示指定	-cmsg	* 共通オプション
終了メッセージ表示抑止指定	-Xcmsg	* 共通オプション
ワーニング発生時の終了コードを 1 にする指定	-cwno	* 共通オプション
ワーニング発生時の終了コードを 0 にする指定	-Xcwno	* 共通オプション

付録 G OS による仕様の相違点

付表 G-1, 付表 G-2, 付表 G-3 に, OS による仕様の相違点を示します。

■ OS による仕様の相違点

付表 G-1 OS による仕様の相違点

OS 依存項目	OS 種別	
	UNIX 系 OS 版	Windows 版
ファイル名の大文字 / 小文字の区別	する	しない
ファイル名のデフォルト拡張子	小文字	大文字小文字区別なし
環境変数 TMP 未指定時の作業ディレクトリ	/tmp	カレントディレクトリ
コマンドラインのワイルドカードによるファイル指定	シェルで展開されツールに渡される。	ツール内部で展開する。

付表 G-2 ワイルドカード展開の違い

ワイルドカードパターン	OS 種別	
	UNIX 系 OS 版	Windows 版
?	任意の 1 文字とマッチする。	ヌル文字または任意の 1 文字とマッチする。
*	任意の文字列とマッチする。	任意の文字列とマッチする。

付表 G-3 ワイルドカード展開の具体例

ワイルドカードパターン	OS 種別	
	UNIX 系 OS 版	Windows 版
a?.obj	a1.obj などがマッチする。 a.obj はマッチしない。	a1.obj などがマッチする。 a.obj もマッチする。
a*	a1.obj, a.obj, a.abs...などがマッチする。	a1.obj, a.obj, a.abs...などがマッチする。
*	abcz, abc.z などがマッチする。	abcz, abc.z などがマッチする。

索引

A

-a	絶対形式ロードモジュールの出力指定 (-a)98
	モジュールの追加 (登録) (-a)222
-adjust	整形指定 (-adjust)274
-AL	自動配置指定 (-AL)124
-AL 1	-AL 1 が指定された場合の配置例67
-AL 2	-AL 2 が指定された場合の配置例69
-alin	相対アセンブルリスト入力ディレクトリ指定 (-alin)151
-alout	絶対形式アセンブルリスト出力ディレクトリ指定 (-alout)152
-alr	ROM/RAM, ARRAY リスト出力指定 (-alr)156
-alrf	ROM/RAM, ARRAY リスト出力モジュール指定 (-alrf)157
-als	絶対形式アセンブルリスト出力指定 (-als)153
-alsf	絶対形式アセンブルリスト出力モジュール指定 (-alsf)154
-an	ROM/RAM, ARRAY リストのシンボルとアドレス の表示位置指定 (-na, -an)159

B

-b	バックアップファイルの作成 (-b)231
----	-----------------------------

C

-c	ライブラリファイルの内容検査 (-c)233
-check_locate	セクション配置領域チェック指定 (-check_locate)143
-check_rora	ユーザ指定領域のチェック指定 (-check_rora)140
-check_size0_sec	サイズ 0 のセクション配置チェック指定 (-check_size0_sec)147
-cif	CPU 情報ファイル指定 (-cif)134, 237
-cmsg	終了メッセージ表示指定 (-cmsg)36

-cmsg オプション	終了メッセージと -cmsg オプション9
-cpu	ターゲット CPU 指定 (-cpu)133, 238
CPU 情報ファイル	CPU 情報ファイル86
	CPU 情報ファイル指定 (-cif)134, 237
	CPU 情報ファイル名86
-cwno	ワーニング発生時の終了コードを 1 にする指定 (-cwno)38

D

-d	モジュールの削除 (-d)224
-demangle	リスト表示のデマングルシンボル名の出力指定 (-demangle)110
-df	外部シンボル値の仮設定 (-df)132
-dt	リスト表示の名前の省略解除 (-dt)107
	リストファイルの詳細情報の出力指定 (-dt)228

E

-e	エントリアドレスの指定 (-e)131
e2ms	e2ms(HEX16 フォーマット S フォーマット変換)307
-el	シンボル個別のライブラリの指定 (-el)128
-entry	スタートアドレスレコード出力指定 (-entry)271

F

-f	オプションファイルからの読み込み指定 (-f)31
f2es	f2es(絶対形式ロードモジュール HEX16 フォーマット変換)278
f2hs	f2hs(絶対形式ロードモジュール HEX フォーマット変換)276
f2is	f2is(絶対形式ロードモジュール HEX8 フォーマット変換)277
f2ms	f2ms(絶対形式ロードモジュール S フォーマット変換)275

FELANG	
FELANG (メッセージ言語)	15
FETOOL	
FETOOL (インストールディレクトリ)	16

G

-g	デバッグ情報の出力指定 (-g)	96
	デバッグ情報の未削除指定 (-g)	235
-gr	セクショングループの指定 (-gr)	122

H

-help	ヘルプメッセージの表示 (-help)	33
HEX16 フォーマット		
e2ms(HEX16 フォーマット		
S フォーマット変換)	307	
f2es(絶対形式ロードモジュール		
HEX16 フォーマット変換)	278	
m2es(S フォーマット		
HEX16 フォーマット変換)	305	
HEX8 フォーマット		
f2is(絶対形式ロードモジュール		
HEX8 フォーマット変換)	277	
i2ms(HEX8 フォーマット		
S フォーマット変換)	306	
m2is(S フォーマット		
HEX8 フォーマット変換)	304	
HEX フォーマット		
f2hs(絶対形式ロードモジュール		
HEX フォーマット変換)	276	

I

-I16	出力 HEX フォーマット指定	
	(-I16/-I20/-I32)	270, 289
-I20	出力 HEX フォーマット指定	
	(-I16/-I20/-I32)	270, 289
i2ms		
i2ms(HEX8 フォーマット		
S フォーマット変換)	306	
-I32	出力 HEX フォーマット指定	
	(-I16/-I20/-I32)	270, 289

L

-L	ライブラリ検索パスの指定 (-L)	127
-I	検索ライブラリファイルの指定 (-I)	126
-len	出力レコード内データ長指定 (-len)	285
LIB911		
LIB911 (ライブラリファイル検索		
ディレクトリ)	17	

M

-m	マッピングリストファイルの作成指定	
	(-m)	299
	マッピングリストファイル名の指定 (-m)	105
	リストファイルの出力指定 (-m)	226
m2es		
m2es(S フォーマット		
HEX16 フォーマット変換)	305	
m2is		
m2is(S フォーマット		
HEX8 フォーマット変換)	304	
-ml	セクション詳細マッピングリスト出力指定	
	(-ml)	167
-mlf	セクション詳細マッピングリストファイル名の指定	
	(-mlf)	168
-mmi	メモリ使用情報リストの出力指定 (-mmi)	108

N

-na	ROM/RAM, ARRAY リストのシンボルとアドレス	
	の表示位置指定 (-na, -an)	159
-NCI0302LIB	デバッグ情報存在チェック抑止指定	
	(-NCI0302LIB)	137
-nd	デフォルトライブラリ検索の抑止指定	
	(-nd)	130
-nl	ライブラリ検索の抑止指定 (-nl)	129

O

-O	ファイル内容の最適化 (-O)	234
-o	出力ファイル名の変更 (-o)	261
	出力ロードモジュールファイル名指定 (-o)	95
-omcl	オブジェクト混在チェックレベル指定	
	(-omcl)	135, 239
OPT		
OPT(デフォルトオプションファイル格納		
ディレクトリ)	19	
OPT911		
OPT911 (デフォルトオプションファイル格納		
ディレクトリ)	18	
OS		
OS による仕様の相違点	367	

P

-p	パディングデータ指定 (-p)	100
	パディング (-p)	263

-pk	パックリンク指定 (-pk)	123
-pl	リスト 1 ページの行数指定 (-pl)	112, 229
-pw	リスト 1 行の桁数指定 (-pw)	230
	リスト行の桁数指定 (-pw)	111

Q

Q&A

オブジェクト形式コンバータの使用上の Q&A	311
ライブラリファイルの作成に関する Q&A	249
リンカの使用上の Q&A	203

R

-r	相対形式ロードモジュールの出力指定 (-r)	99
	モジュールの置換 (登録) (-r)	223
-ra	RAM 領域の指定 (-ra)	119
RAM	RAM 領域の指定 (-ra)	119
	ROM/RAM 領域の設定とセクション配置	82
	ROM および RAM 領域の指定	56
-ran	出力範囲指定 (-ran)	286, 296
-ro	ROM 領域の指定 (-ro)	118
ROM	ROM/RAM 領域の設定とセクション配置	82
	ROM および RAM 領域の指定	56
	ROM 化支援	56
	ROM 領域の指定 (-ro)	118
ROM/RAM	内蔵 ROM/RAM 領域自動設定の抑止指定 (-Xset_rora)	139
	内蔵 ROM/RAM 領域の自動設定 (-set_rora)	138
ROM/RAM, ARRAY リスト	ROM/RAM, ARRAY リスト	186
	ROM/RAM, ARRAY リスト出力指定 (-alr)	156
	ROM/RAM, ARRAY リスト出力モジュール指定 (-alrf)	157
	ROM/RAM, ARRAY リスト出力抑止指定 (-Xalr)	158
	ROM/RAM, ARRAY リストのシンボルとアドレス の表示位置指定 (-na, -an)	159
ROM/RAM 領域名	ROM/RAM 領域名	86
ROM RAM 転送	ROM RAM 転送セクション	83
	ROM RAM 転送セクションの使用方法	83
	ROM RAM 転送セクションの注意	84

S

S0 タイプ	S0 タイプ (ヘッダレコード)	353
-S1	出力 S フォーマット指定 (-S1/-S2/-S3)	269, 287
S1 タイプ	S1 タイプ (データレコード: 2 バイトアドレス)	355
-S2	出力 S フォーマット指定 (-S1/-S2/-S3)	269, 287
S2 タイプ	S2 タイプ (データレコード: 3 バイトアドレス)	356
-S3	出力 S フォーマット指定 (-S1/-S2/-S3)	269, 287
S3 タイプ	S3 タイプ (データレコード: 4 バイトアドレス)	357
S5 タイプ	S5 タイプ (レコード数管理レコード)	358
S7 タイプ	S7 タイプ (ターミネータレコード)	359
S8 タイプ	S8 タイプ (ターミネータレコード)	360
S9 タイプ	S9 タイプ (ターミネータレコード)	361
-sc	セクション配置順 / アドレスの指定 (-sc)	120
-set_rora	内蔵 ROM/RAM 領域の自動設定 (-set_rora)	138
-sl	ローカルシンボル情報リスト出力指定 (-sl)	164
-slf	ローカルシンボル情報リストファイル名の指定 (-slf)	165
-sp	スプリットモード指定 (-sp)	297
-ST	開始アドレス変更指定 (-ST)	290
-symtab	外部シンボル情報出力指定 (-symtab)	103
S フォーマット	e2ms(HEX16 フォーマット S フォーマット変換)	307
	f2ms(絶対形式ロードモジュール S フォーマット変換)	275
	i2ms(HEX8 フォーマット S フォーマット変換)	306
	m2es(S フォーマット HEX16 フォーマット変換)	305
	m2is(S フォーマット HEX8 フォーマット変換)	304

T

TMP

TMP(ワークディレクトリ)14

V

-V

版数 / メッセージ出力指定 (-V)34

-V オプション

起動メッセージと -V オプション8

W

-w

警告メッセージ出力レベルの指定 (-w)117

X

-x

モジュールの抽出 (-x)225

-Xalr

ROM/RAM , ARRAY リスト出力抑止指定
(-Xalr)158

-Xals

絶対形式アセンブルリスト出力抑止指定
(-Xals)155

-Xb

バックアップファイルの作成抑止 (-Xb)232

-Xcheck_locate

セクション配置領域チェック抑止指定
(-Xcheck_locate)146

-Xcheck_rora

ユーザ指定領域のチェック抑止指定
(-Xcheck_rora)142

-Xcheck_size0_sec

サイズ 0 のセクション配置チェック抑止指定
(-Xcheck_size0_sec)149

-Xcmsg

終了メッセージ表示抑止指定 (-Xcmsg)37

-Xcwno

ワーニング発生時の終了コードを 0 にする指定
(-Xcwno)39

-Xdemangle

リスト表示のデマングルシンボル名の出力
抑止指定 (-Xdemangle)109

-Xdof

デフォルトオプションファイル抑止指定
(-Xdof)30

-Xentry

スタートアドレスレコード出力抑止指定
(-Xentry)273

-Xg

デバッグ情報の削除指定 (-Xg)97, 236

-xl

外部シンボル相互参照情報リスト出力指定
(-xl)161

-xlf

外部シンボル相互参照情報リストファイル名の
指定 (-xlf)162

-Xm

マップリスト出力の抑止指定 (-Xm)106
マップリストファイルの作成抑止指定
(-Xm)301
リストファイルの出力抑止指定 (-Xm)227

-Xml

セクション詳細マップリスト出力抑止指定
(-Xml)169

-XPLNK

ブレイク処理の抑止指定 (-XPLNK)150

-Xset_rora

内蔵 ROM/RAM 領域自動設定の抑止指定
(-Xset_rora)139

-Xsl

ローカルシンボル情報リスト出力抑止指定
(-Xsl)166

-Xsp

スプリットモード抑止指定 (-Xsp)298

-Xsymtab

外部シンボル情報出力抑止指定
(-Xsymtab)104

-XV

版数 / メッセージ出力抑止 (-XV)35

-Xxl

外部シンボル相互参照情報リスト出力抑止指定
(-Xxl)163

あ

アセンブルソースリスト	
アセンブルソースリストの形式	188

い

一般形式	
一般形式	344

え

エラーメッセージ	
アセンブルリスト内のエラーメッセージ	184
コンバータのエラーメッセージ	338
ライブラリアンのエラーメッセージ	331
リンカのエラーメッセージ	316
リンケージキットのエラーメッセージの 表示形式	315
リンケージキットのエラーメッセージ 分類	314
エントリアドレス	
エントリアドレスの指定 (-e)	131
エントリアドレス / シンボル値	
エントリアドレス / シンボル値の設定	54
エントリアドレス指定	
エントリアドレス指定	58
エンドレコード	
エンドレコード (HEX8/HEX16/HEX32)	347

お

オブジェクト	
SOFTUNE V3/V5 言語ツールで作成された オブジェクト	215
SOFTUNE V3/V5 ツールで作成したオブジェクト の混在	87
オブジェクト形式コンバータ	
オブジェクト形式コンバータのオプション 一覧	366
オブジェクト形式コンバータの概要	254
オブジェクト形式コンバータの共通オプションの 種類	260
オブジェクト形式コンバータの コマンド一覧	365
オブジェクト形式コンバータの コマンド実行	258
オブジェクト形式コンバータの使用上の Q&A	311
オブジェクト形式コンバータの制限事項	310
オブジェクト混在チェック	
オブジェクト混在チェックレベル指定 (-omcl)	135, 239
オプション	
オブジェクト形式コンバータのオプション 一覧	366
オブジェクト形式コンバータの共通オプションの 種類	260
オプション一覧	220
オプション形式	22
オプション指定時の注意と評価	24

オプションの指定に関する注意	248
オプションファイルからの読み込み指定 (-f)	31
相反関係にあるオプションの指定例	25
デフォルトオプションファイル	46
デフォルトオプションファイル抑止指定 (-Xdof)	30
バイナリコンバータのオプション一覧	294
フォーマットアジャスタのオプション 一覧	283
包含関係にあるオプションの指定例	25
ライブラリアンのオプション一覧	364
リンカのオプション一覧	92, 362
ロードモジュールコンバータのオプション 一覧	267
オプションパラメータ	
オプションパラメータでの数値表現	23
オプションファイル	
オプションファイル	42
オプションファイルからの読み込み指定 (-f)	31
オプションファイル指定による実行	42
オプションファイル中の継続指定	43
オプションファイル中のコメント指定	44
デフォルトオプションファイル	46

か

開始アドレス変更	
開始アドレス変更指定 (-ST)	290
外部シンボル情報出力指定	
外部シンボル情報出力指定 (-symtab)	103
外部シンボル情報出力抑止指定	
外部シンボル情報出力抑止指定 (-Xsymtab)	104
外部シンボル相互参照情報リスト	
外部シンボル相互参照情報リスト	172
外部シンボル相互参照情報リスト出力指定 (-xl)	161
外部シンボル相互参照情報リスト出力抑止指定 (-Xxl)	163
外部シンボル相互参照情報リスト ファイル	194
外部シンボル相互参照情報リストファイル名の 指定 (-xlf)	162
外部シンボル値	
外部シンボル値の仮設定 (-df)	132
外部シンボル値の設定	58
外部定義 / 参照シンボル情報 リスト出力概要	245
拡張セグメントアドレスレコード 拡張セグメントアドレスレコード (HEX16/HEX32)	348
拡張リニアアドレスレコード	
拡張リニアアドレスレコード (HEX32)	350

き

起動メッセージ	
起動メッセージと -V オプション	8
起動メッセージ表示の選択	55
強制終了	
強制終了	6

共通オプション	
共通オプション一覧	28

く

グループ化	
セクションのグループ化	56
クロスリファレンスリスト	
クロスリファレンスリスト	192

け

警告チェックレベル	
警告チェックレベルの選択	55
警告メッセージ	
警告メッセージ出力レベルの指定 (-w)	117
検索ライブラリ	
検索ライブラリの指定	71
検索ライブラリファイル	
検索ライブラリファイル指定	57
検索ライブラリファイルの指定 (-l)	126
シンボルごとの検索ライブラリファイル指定	57

こ

コマンド	
オブジェクト形式コンバータの	
コマンド一覧	365
コマンドライン	
コマンドラインの形式	5
コマンドラインの指定例	26
コントロールリスト	
コントロールリスト部のリスト出力	
フォーマット	174
コントロールリスト部のリスト表示例	175
コンバータ	
コンバータのエラーメッセージ	338

さ

最適化	
ファイル内容の最適化 (-O)	234
サポート範囲	
リンケージキットのサポート範囲	4

し

識別子	
識別子の区別	11
識別子の構成文字種	11
識別子文字数の制限	11
識別名表示	
リスト出力時の識別名表示	11
自動配置	
自動配置指定 (-AL)	124
終了コード	
終了コードの値と終了状態	7
ワーニング発生時の終了コードを 0 にする指定 (-Xcwno)	39

ワーニング発生時の終了コードを 1 にする指定 (-cwno)	38
終了メッセージ	
終了メッセージと -cmsg オプション	9
終了メッセージ表示指定 (-cmsg)	36
終了メッセージ表示の選択	55
終了メッセージ表示抑止指定 (-Xcmsg)	37
出力 HEX フォーマット	
出力 HEX フォーマット指定 (-I16/-I20/-I32)	270, 284, 289
出力 S フォーマット	
出力 S フォーマット指定 (-S1/-S2/-S3)	269, 284, 287
出力範囲	
出力範囲指定 (-ran)	284, 286, 295, 296
出力ファイル名	
出力ファイル名の変更 (-o)	261
出力フォーマット	
出力フォーマットの指定	55
出力レコード	
出力レコード内データ長指定 (-len)	285
出力ロードモジュールファイル	
出力ロードモジュールファイル名指定 (-o)	95
出力ロードモジュールファイル名の指定	55
シンボルリスト	
シンボルリスト部のリスト出力	
フォーマット	181
シンボルリスト部のリスト表示例	182

す

スタートアドレスレコード出力指定	
スタートアドレスレコード出力指定 (-entry)	271
スタートアドレスレコード出力抑止指定	
スタートアドレスレコード出力抑止指定 (-Xentry)	273
スタートセグメントアドレスレコード	
スタートセグメントアドレスレコード (HEX16/HEX32)	349
スタートリニアアドレスレコード	
スタートリニアアドレスレコード (HEX32)	351
スプリットモード	
スプリットモード指定 (-sp)	295, 297
スプリットモードの概要	293
スプリットモード抑止指定 (-Xsp)	295, 298

せ

整形指定	
整形指定 (-adjust)	274
セクション	
ROM/RAM 領域の設定とセクション配置	82
ROM RAM 転送セクション	83
ROM RAM 転送セクションの使用方法	83
ROM RAM 転送セクションの注意	84
セクショングループの指定 (-gr)	122
セクショングループの指定がある場合の配置例	64
セクション種別と配置先	68
セクション内容種別	59

セクションの共有結合	60
セクションのグループ化	56
セクションの結合順序が指定された場合の 配置例	63
セクションの結合順序が指定されなかった場合の 配置例	63
セクションの結合順序が指定されなかった場合の 配置例	62
セクションの結合属性	59
セクションの識別	59
セクションの自動配置	65
セクションの単純連結結合	60
セクションの配置順と配置アドレスの 指定	56
セクションの配置属性	59
セクションのリンク	61
セクション配置順 / アドレスの指定 (-sc)	120
セクション名	59
セクションセクション詳細マップリスト セクション詳細マップリスト出力抑止指定 (-Xml)	169
セクション詳細マップリストファイル名の指定 (-mlf)	168
セクション詳細マップリスト セクション詳細マップリスト	172
セクション詳細マップリスト出力指定 (-ml)	167
セクション情報リスト セクション情報リスト	190
セクションの結合 / 配置 セクションの結合 / 配置に関する制御	54
セクション配置 セクション配置領域チェック指定 (-check_locate)	143
セクション配置領域チェック抑止指定 (-Xcheck_locate)	146
セクション配置詳細情報リスト セクション配置詳細情報リストファイル	198
セクション配置チェック サイズ 0 のセクション配置チェック指定 (-check_size0_sec)	147
セクション配置チェック抑止 サイズ 0 のセクション配置チェック抑止指定 (-Xcheck_size0_sec)	149
絶対形式アセンブルリスト 絶対形式アセンブルリスト	172
絶対形式アセンブルリスト出力指定 (-als)	153
絶対形式アセンブルリスト出力ディレクトリ指定 (-alout)	152
絶対形式アセンブルリスト出力モジュール指定 (-alsf)	154
絶対形式アセンブルリスト出力抑止指定 (-Xals)	155
絶対形式アセンブルリストの形式	183
絶対形式ロードモジュール f2es(絶対形式ロードモジュール HEX16 フォーマット変換)	278
f2hs(絶対形式ロードモジュール HEX フォーマット変換)	276
f2is(絶対形式ロードモジュール HEX8 フォーマット変換)	277
f2ms(絶対形式ロードモジュール S フォーマット変換)	275

絶対形式ロードモジュールの出力指定 (-a)	98
------------------------------	----

そ

相対アセンブルリスト 相対アセンブルリスト入力ディレクトリ指定 (-alin)	151
相対形式ロードモジュール 相対形式ロードモジュールの出力指定 (-r)	99
その他のコンバータ その他のコンバータの種類	257

た

ターゲット CPU ターゲット CPU 指定 (-cpu)	133, 238
ターミネータレコード S7 タイプ (ターミネータレコード)	359
S8 タイプ (ターミネータレコード)	360
S9 タイプ (ターミネータレコード)	361

て

ディスク容量 必要なディスク容量についての注意	248
データ長 出力レコード内データ長指定 (-len)	284, 285
データレコード S1 タイプ (データレコード: 2 バイトアドレス)	355
S2 タイプ (データレコード: 3 バイトアドレス)	356
S3 タイプ (データレコード: 4 バイトアドレス)	357
データレコード (HEX8/HEX16/HEX32)	346
デバッグ情報 デバッグ情報存在チェック抑止指定 (-NCI0302LIB)	137
デバッグ情報の継承	55
デバッグ情報の削除	213
デバッグ情報の削除指定 (-Xg)	97, 236
デバッグ情報の出力指定 (-g)	96
デバッグ情報の未削除指定 (-g)	235
デフォルトオプションファイル デフォルトオプションファイル	46
デフォルトオプションファイル抑止指定 (-Xdof)	30

な

内蔵 ROM/RAM 領域自動設定 内蔵 ROM/RAM 領域自動設定の抑止指定	86
---	----

に

入出力ファイル / メッセージ 入出力ファイル / メッセージに関する制御	53
入力オブジェクトファイル 入力オブジェクトファイルの指定	55

は

配置アドレス	
配置アドレスの決定方法	66, 68
バイナリコンバータ	
バイナリコンバータ	256
バイナリコンバータのオプション一覧	294
バイナリコンバータの概要	292
バイナリコンバータ, アジャスタの制限事項	310
バックアップファイル	
バックアップファイルの作成 (-b)	231
バックアップファイルの作成抑止 (-Xb)	232
バックリンク	
バックリンク指定 (-pk)	123
パディング	
パディング (-p)	263
パディングデータ	
パディングデータ指定 (-p)	100
パラメータ	
パラメータ	22
版数 / メッセージ出力	
版数 / メッセージ出力指定 (-V)	34
版数 / メッセージ出力抑止 (-XV)	35

ふ

ファイル名	
ファイル名の文字コード	12
ファイル名の文字数	12
フォーマットアジャスタ	
フォーマットアジャスタ	256
フォーマットアジャスタのオプション一覧	283
フォーマットアジャスタの概要	280
フォーマットアジャスタの機能	281
ブレリンク処理	
ブレリンク処理の抑止指定 (-XPLNK)	150

へ

ヘッダ	
ヘッダ形式	185
ヘッダレコード	
S0 タイプ (ヘッダレコード)	353
ヘルプメッセージ	
ヘルプメッセージ	10
ヘルプメッセージの表示 (-help)	33

ま

マップリスト	
マップリスト出力の抑止指定 (-Xm)	106
マップリストファイルの作成指定 (-m)	295, 299
マップリストファイルの作成抑止指定 (-Xm)	295, 301
マップリストファイル名の指定 (-m)	105
マップリスト部のリスト出力	
フォーマット	176
マップリスト部のリスト表示例	177

マングル名	
マングル名	11

め

メモリ使用情報リスト	
メモリ使用情報リストの出力指定 (-mmi)	108
メモリ使用情報リスト部のリスト出力	
フォーマット	178
メモリ使用情報リスト部のリスト表示例	180

も

モジュール	
モジュールの置換 (登録) (-r)	223
モジュールの削除 (-d)	224
モジュールの抽出 (-x)	225
モジュールの追加 (登録) (-a)	222

ら

ライブラリ	
SOFTUNE V3/V5 言語ツールで作成されたライブラリ	216
シンボル個別のライブラリの指定 (-el)	128
ライブラリアン	
ライブラリアンのエラーメッセージ	331
ライブラリアンのオプション一覧	364
ライブラリアンの制限事項	248
ライブラリアンの役割	208
ライブラリファイルの作成に関する Q&A	249
ライブラリ検索	
デフォルトライブラリ検索の抑止指定 (-nd)	130
ライブラリ検索の抑止	57
ライブラリ検索の抑止指定 (-nl)	129
ライブラリ検索パス	
ライブラリ検索パス指定	57
ライブラリ検索パスの指定 (-L)	127
ライブラリの検索	
ライブラリの検索に関する制御	54
ライブラリファイル	
ライブラリファイルが 1 つの場合の検索例 1	72
ライブラリファイルが 1 つの場合の検索例 2	74
ライブラリファイルが 1 つの場合の検索例 3	75
ライブラリファイルが個別に指定された場合の処理	80
ライブラリファイルが複数の場合の検索例 1	76
ライブラリファイルが複数の場合の検索例 2	78
ライブラリファイル内のモジュール抽出	212
ライブラリファイルの検索順序	71
ライブラリファイルの新規作成	210
ライブラリファイルの内容検査 (-c)	233
ライブラリファイルの内容表示	214
ライブラリファイルの内容チェック	214

ライブラリファイルの編集	
ライブラリファイルの編集	210

リ

リスト	
リスト 1 行の桁数指定 (-pw)	230
リスト 1 ページの行数指定 (-pl)	112, 229
リスト行の桁数指定 (-pw)	111
リスト出力概要	243, 244, 245
リスト表示の名前の省略解除 (-dt)	107
リスト表示のデマングルシンボル名	
リスト表示のデマングルシンボル名の出力指定 (-demangle)	110
リスト表示のデマングルシンボル名の出力 抑止指定 (-Xdemangle)	109
リストファイル	
リストファイルの形式変更	55
リストファイルの構成	242
リストファイルの出力指定 (-m)	226
リストファイルの出力抑止指定 (-Xm)	227
リストファイルの詳細情報の出力指定 (-dt)	228
領域チェック	
ユーザ指定領域のチェック指定 (-check_rora)	140
ユーザ指定領域のチェック抑止指定 (-Xcheck_rora)	142
リンカ	
リンカの使用上の Q&A	203
リンカのエラーメッセージ	316
リンカのオプション一覧	92, 362
リンカの概要	52
リンカの制限事項	202
リンカの予約名	202
リンクリストファイル	
リンクリストファイル	172
リンクリストファイルの構成	173

リンケージキット	
リンケージキットのエラーメッセージの 表示形式	315
リンケージキットのエラーメッセージ 分類	314
リンケージキットのサポート範囲	4

れ

レコード開始アドレス変更	
レコード開始アドレス変更指定 (-ST)	284
レコード数管理レコード	
S5 タイプ (レコード数管理レコード)	358

ろ

ローカルシンボル情報リスト	
ローカルシンボル情報リスト	172
ローカルシンボル情報リスト出力指定 (-sl)	164
ローカルシンボル情報リスト出力抑止指定 (-Xsl)	166
ローカルシンボル情報リストファイル名の指定 (-slf)	165
ローカルシンボル情報リストファイル	
ローカルシンボル情報リストファイル	196
ロードモジュールコンバータ	
ロードモジュールコンバータのオプション 一覧	267
ロードモジュールコンバータの概要	266
ロードモジュールコンバータの種類	256

わ

ワークディレクトリ	
TMP(ワークディレクトリ)	14
ワーニング発生時の終了コード	
ワーニング発生時の終了コードを 0 にする指定 (-Xcwno)	39
ワーニング発生時の終了コードを 1 にする指定 (-cwno)	38

CM71-00327-5

富士通マイクロエレクトロニクス・CONTROLLER MANUAL

FR ファミリ

SOFTUNETM リンケージキット マニュアル

V6 対応

2008 年 6 月 第 5 版 発行

発行	富士通マイクロエレクトロニクス株式会社
編集	マーケティング統括部 ビジネス推進部
