お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

ユーザーズ・マニュアル

RA78K0 Ver.3.80

アセンブラ・パッケージ

操作編

対象デバイス
78K0シリーズ

資料番号　U17199JJ1V0UM00（第1版）
発行年月　January 2005　CP(K)

〔メ　モ〕

目次要約

第1章　概　　説 … 17

第2章　製品概要とインストール方法 … 37

第3章　RA78K0の実行手順 … 46

第4章　構造化アセンブラ・プリプロセッサ … 68

第5章　アセンブラ … 94

第6章　リ ン カ … 146

第7章　オブジェクト・コンバータ … 200

第8章　ライブラリアン … 240

第9章　リスト・コンバータ … 273

第10章　プログラムの出力リスト … 292

第11章　RA78K0の活用法 … 312

第12章　エラー・メッセージ … 320

付録A　サンプル・プログラム … 374

付録B　使用上の注意一覧 … 381

付録C　コマンド・オプション … 386

付録D　サブコマンド … 398

付録E　総合索引 … 399

Windows**および**Windows NT**は，米国**Microsoft Corporation**の米国およびその他の国における登録商標または商標です。**

UNIX**は**X/Open**カンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標です。**

i386**は，米国**Intel Corp.**の商標です。**

PC/AT**は，米国**IBM Corp.**の商標です。**

HP9000**シリーズ**700, HP-UX**は，米国**Hewlett-Packard Corp.**の商標です。**

SPARCstation**は，米国**SPARC International, Inc.**の商標です。**

Solaris, SunOS**は，米国**Sun Microsystems, Inc.**の商標です。**

- 本資料に記載されている内容は2005年1月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品の使用に関連し発生した第三者の特許権，著作権その他の知的財産権の侵害等に関し，一切その責を負いません。当社は，本資料に基づき当社または第三者の特許権，著作権その他の知的財産権を何ら許諾するものではありません。
- 本資料に記載された回路，ソフトウエアおよびこれらに関する情報は，半導体製品の動作例，応用例を説明するものです。お客様の機器の設計において，回路，ソフトウエアおよびこれらに関する情報を使用する場合には，お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し，当社は，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「特定水準」に分類しております。また，各品質水準は，以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認ください。

  標準水準：コンピュータ，OA機器，通信機器，計測機器，AV機器，家電，工作機械，パーソナル機器，産業用ロボット

  特別水準：輸送機器（自動車，電車，船舶等），交通用信号機器，防災・防犯装置，各種安全装置，生命維持を目的として設計されていない医療機器

  特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

当社製品のデータ・シート，データ・ブック等の資料で特に品質水準の表示がない場合は，標準水準製品であることを表します。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNECエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいう。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E　02.11

はじめに

このマニュアルは，RA78K0　アセンブラ・パッケージ（以降，RA78K0とします）を用いてソフトウエア開発を行われる方に，RA78K0に含まれる各プログラムの機能および操作方法を正しく理解していただくことを目的としています。

このマニュアルではRA78K0の疑似命令やソース・プログラムの様式など，言語に関する解説はしていません。したがって，このマニュアルをお読みになる前にRA78K0　**アセンブラ・パッケージ　ユーザーズ・マニュアル　言語編**（U17198J）（以降，言語編とします）をお読みください。

このマニュアルでのRA78K0に関する記述は，Ver.3.80の製品に対応しています。

【対 象 者】

このマニュアルでは，開発対象となるマイクロコンピュータ（78K0シリーズ）の機能およびインストラクションについて理解しているユーザを対象としています。

【構　　成】

このマニュアルは次のように構成されています。

第1章　概　　説

マイクロコンピュータの開発におけるRA78K0の役割など，RA78K0全体の機能概要について説明します。

第2章　製品概要とインストール方法

RA78K0が提供するプログラムのファイル名，動作環境について説明します。

第3章　RA78K0の実行手順

サンプル・プログラムを使用して，開発手順について説明します。

この章は，各プログラムを実際に動作させることを目的としていますので，RA78K0を動作させてみたい方は，この章からお読みください。

第4章　構造化アセンブラ・プリプロセッサ

第5章　アセンブラ

第6章　リ ン カ

第7章　オブジェクト・コンバータ

第8章　ライブラリアン

第9章　リスト・コンバータ

第10章　プログラムの出力リスト

各プログラムが出力する各種リストのフォーマットについて説明します。

第11章　RA78K0**の活用法**

RA78K0**をうまく使うための方法を紹介します。**

第12章　エラー・メッセージ

各プログラムが出力するエラー・メッセージについて説明します。

付　　録

プログラムのオプション一覧表，サンプル・プログラム・リスト，使用上の注意一覧などを紹介します。

なお，このマニュアルではインストラクションについての詳細説明はしておりません。

インストラクションの詳細については，開発対象となるマイクロコンピュータのユーザーズ・マニュアルをお読みください。

【読 み 方】

アセンブラを初めて使われる方は，第1章　概説からお読みください。アセンブラに関する一般的知識のある方は読み飛ばされても結構です。

実際にRA78K0**を使用する場合は，第3章**　RA78K0**の実行手順をお読みください。**

各プログラムの操作に慣れてからは，付録の一覧表をご活用ください。

【注　　意】

このマニュアルでは，ホスト・マシンとしてPC-9800**シリーズ，**IBM PC/AT™**互換機の場合を対象にして書かれています。**HP9000**シリーズ**700™, SPARCstation™**ファミリの場合には，ホスト・マシン（**OS**）に依存した若干の違いがあります。次のような点にご注意ください。**

●ファイル名の形式が異なります。

- **実行形式の拡張子**.exe**は，**HP9000**シリーズ**700**などの**EWS**版ではつきません。**
- **バッチ・ファイルの拡張子**.bat**は，**HP9000**シリーズ**700**などの**EWS**版では**.sh**になります。**
- **大文字のファイル名は，**HP9000**シリーズ**700**などの**EWS**版では小文字になります。**

●マニュアルに記載されている実行例や環境設定の方法が異なります。

【凡　例】

このマニュアルの中で共通に使用される記号などの意味を示します。

⋮	**；同一の形式を繰り返します。**
[]	**；［］の中は省略可能です。**
「」	**；「」で囲まれた文字そのものを表します。**
‘ ’	**；‘ ’で囲まれた文字そのものを表します。**
＜＞	**；ダイアログ，ウインドウの名称を表します。**
“ ”	**；このマニュアルでの参照箇所（章，節，項，図，表）を表します。**
____	**；重要箇所，また，使用例での下線は入力文字を表します。**
□	**；1個の空白を表します。**
△	**；1個以上の空白またはタブを表します。**
▲	**；0個以上の空白またはタブ（省略可能の意）を表します。**
／	**；文字の区切りを表します。**
～	**；連続性を表します。**
↵	**；リターン・キーの入力を表します。**
注	**；本文中に付けた注の説明**
注意	**；特に気を付けて読んでいただきたい内容**
備考	**；本文中の補足説明**

【関連資料】

このマニュアルに関連する資料（ユーザーズ・マニュアル）を紹介します。

関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

開発ツールの資料（ユーザーズ・マニュアル）

資料名		資料番号	
		和文	英文
RA78K0　アセンブラ・パッケージ Ver.3.80	操作編	このマニュアル	U17199E
	言語編	U17198J	U17198E
	構造化アセンブリ言語編	U17197J	U17197E
CC78K0　Cコンパイラ Ver.3.70	操作編	U17201J	U17201E
	言語編	U17200J	U17200E
SM+　システム・シミュレータ	操作編	U17246J	U17246E
	ユーザ・オープン・インタフェース編	U17247J	U17247E
SM78Kシリーズ Ver.2.52　システム・シミュレータ	操作編	U16768J	U16768E
PM plus Ver.5.20		U16934J	U16934E
ID78K0-NS Ver.2.52　統合ディバッガ	操作編	U16488J	U16488E
ID78K0-QB Ver.2.81　統合ディバッガ	操作編	U16996J	U16996E
78K0シリーズ	命令編	U12326J	U12326E

注意　上記関連資料は予告なしに内容を変更することがあります。設計などには必ず最新の資料をご使用ください。

目次

第 1 章　概説 … 17
- 1.1 **概要** … 17
  - 1.1.1 **アセンブラとは** … 18
  - 1.1.2 **開発工程と** RA78K0 … 19
  - 1.1.3 **リロケータブル・アセンブラとは** … 22
  - 1.1.4 **リロケータブル・アセンブラの利点** … 22
- 1.2 RA78K0 **の機能概要** … 24
  - 1.2.1 **エディタによるソース・モジュール・ファイルの作成** … 25
  - 1.2.2 **構造化アセンブラ・プリプロセッサ** … 26
  - 1.2.3 **アセンブラ** … 27
  - 1.2.4 **リ ン カ** … 28
  - 1.2.5 **オブジェクト・コンバータ** … 29
  - 1.2.6 **ライブラリアン** … 30
  - 1.2.7 **リスト・コンバータ** … 31
  - 1.2.8 **ディバッガ** … 32
- 1.3 **プログラム開発をはじめる前に** … 33
  - 1.3.1 RA78K0 **の最大性能** … 33
- 1.4 RA78K0 **の特徴** … 36

第 2 章　製品概要とインストール方法 … 37
- 2.1 **ホスト・マシンと供給媒体** … 37
- 2.2 **インストール** … 38
  - 2.2.1 Windows **版のインストール** … 38
  - 2.2.2 UNIX **版のインストール** … 39
- 2.3 **デバイス・ファイルのインストール** … 40
  - 2.3.1 Windows **版のインストール** … 40
  - 2.3.2 UNIX **版のインストール** … 40
  - 2.3.3 **デバイス・ファイルのレジストリ登録** … 40
- 2.4 **ディレクトリ構成** … 41
  - 2.4.1 Windows **版のディレクトリ構成** … 41
  - 2.4.2 UNIX **版のディレクトリ構成** … 42
- 2.5 **アンインストール手順** … 43
  - 2.5.1 Windows **版のアンインストール** … 43
  - 2.5.2 UNIX **版のアンインストール** … 43
- 2.6 **環境設定** … 44
  - 2.6.1 **ホスト・マシン（**IBM PC/AT **互換機の場合）** … 44
  - 2.6.2 **環境変数** … 44
  - 2.6.3 **ソース・ファイル中の漢字コード** … 45

第 3 章　RA78K0 **の実行手順** … 46
- 3.1 RA78K0 **実行の前に** … 46
  - 3.1.1 **サンプル・プログラム** … 46
  - 3.1.2 **サンプル・プログラムの構成** … 49
- 3.2 RA78K0 **の実行手順** … 50
- 3.3 ST78K0 **の実行手順** … 56
- 3.4 **コマンド行（**DOS **プロンプト，**UNIX**）でのアセンブルからリンク，オブジェクト・コンバートの実行手順** … 62
- 3.5 **パラメータ・ファイルの使用** … 66

第 4 章　構造化アセンブラ・プリプロセッサ … 68
- 4.1 **構造化アセンブラ・プリプロセッサの入出力ファイル** … 68
- 4.2 **構造化アセンブラ・プリプロセッサの機能** … 69
- 4.3 **構造化アセンブラ・プリプロセッサの起動方法** … 70
  - 4.3.1 **構造化アセンブラ・プリプロセッサの起動** … 70
  - 4.3.2 **実行開始メッセージ，終了メッセージ** … 71

4.4 **構造化アセンブラ・オプション** … 74
4.4.1 **構造化アセンブラ・オプションの種類** … 74
4.4.2 **構造化アセンブラ・オプションの説明** … 74
4.5 PM plus **でのオプション設定** … 89
4.5.1 **オプションの設定方法** … 89
4.5.2 **各オプションの設定** … 91
4.5.3 **オプションの編集ダイアログ** … 93

第 5 **章　アセンブラ** … 94
5.1 **アセンブラの入出力ファイル** … 94
5.2 **アセンブラの機能** … 96
5.3 **アセンブラの起動方法** … 97
5.3.1 **コマンド行での起動** … 97
5.3.2 **パラメータ・ファイルによる起動** … 98
5.3.3 **実行開始メッセージ，終了メッセージ** … 99
5.4 **アセンブラ・オプション** … 101
5.4.1 **アセンブラ・オプションの種類** … 101
5.4.2 **アセンブラ・オプションの優先度** … 103
5.4.3 **アセンブラ・オプションの説明** … 104
5.5 PM plus **でのオプション設定** … 140
5.5.1 **オプションの設定方法** … 140
5.5.2 **各オプションの設定** … 142
5.5.3 **オプションの編集ダイアログ** … 145

第 6 **章　リンカ** … 146
6.1 **リンカの入出力ファイル** … 146
6.2 **リンカの機能** … 147
6.3 **メモリ空間とメモリ領域** … 148
6.4 **リンク・ディレクティブ** … 149
6.4.1 **ディレクティブ・ファイル** … 149
6.4.2 **メモリ・ディレクティブ** … 151
6.4.3 **セグメント配置ディレクティブ** … 153
6.5 **リンカの起動方法** … 156
6.5.1 **リンカの起動** … 156
6.5.2 **実行開始メッセージ，終了メッセージ** … 157
6.6 **リンカ・オプション** … 159
6.6.1 **リンカ・オプションの種類** … 159
6.6.2 **リンカ・オプションの優先度** … 161
6.6.3 **リンカ・オプションの説明** … 161
6.7 PM plus **でのオプション設定** … 193
6.7.1 **オプションの設定方法** … 193
6.7.2 **各オプションの設定** … 196
6.7.3 **オプションの編集ダイアログ** … 199

第 7 **章　オブジェクト・コンバータ** … 200
7.1 **オブジェクト・コンバータの入出力ファイル** … 200
7.2 **オブジェクト・コンバータの機能** … 202
7.2.1 **拡張空間への対応** … 202
7.2.2 **フラッシュ** ROM **セルフ書き換えモード対応** … 202
7.2.3 HEX **形式オブジェクト・モジュール・ファイル** … 203
7.2.4 **シンボル・テーブル・ファイル** … 215
7.3 **オブジェクト・コンバータの起動方法** … 217
7.3.1 **オブジェクト・コンバータの起動** … 217
7.3.2 **実行開始メッセージ，終了メッセージ** … 218
7.4 **オブジェクト・コンバータ・オプション** … 220
7.4.1 **オブジェクト・コンバータ・オプションの種類** … 220
7.4.2 **オブジェクト・コンバータ・オプションの説明** … 221
7.5 PM plus **でのオプション設定** … 236
7.5.1 **オプションの設定方法** … 236
7.5.2 **各オプションの設定** … 238

第 8 **章　ライブラリアン** … 240
8.1 **ライブラリアンの入出力ファイル** … 240

8.2 ライブラリアンの機能 … 242
8.3 ライブラリアンの起動方法 … 244
8.3.1 ライブラリアンの起動 … 244
8.3.2 実行開始メッセージ，終了メッセージ … 247
8.4 ライブラリアン・オプション … 248
8.4.1 ライブラリアン・オプションの種類 … 248
8.4.2 ライブラリアン・オプションの説明 … 248
8.5 サブコマンド … 256
8.5.1 サブコマンドの種類 … 256
8.5.2 サブコマンドの説明 … 256
8.6 PM plus でのオプション設定 … 267
8.6.1 オプションの設定方法 … 267
8.6.2 各オプションの設定 … 269
8.7 PM plus でのライブラリ・ファイルの操作 … 270
8.7.1 操作方法 … 270
8.7.2 各項目の設定 … 271

第 9 章　リスト・コンバータ … 273
9.1 リスト・コンバータの入出力ファイル … 273
9.2 リスト・コンバータの機能 … 275
9.3 リスト・コンバータの起動方法 … 278
9.3.1 リスト・コンバータの起動 … 278
9.3.2 実行開始メッセージ，終了メッセージ … 280
9.4 リスト・コンバータ・オプション … 281
9.4.1 リスト・コンバータ・オプションの種類 … 281
9.4.2 リスト・コンバータ・オプションの説明 … 281
9.5 PM plus でのオプション設定 … 289
9.5.1 オプションの設定方法 … 289
9.5.2 各オプションの設定 … 291

第 10 章　プログラムの出力リスト … 292
10.1 構造化アセンブラ・プリプロセッサの出力リスト … 292
10.1.1 エラー・リスト … 293
10.2 アセンブラの出力リスト … 294
10.2.1 アセンブル・リスト・ファイルのヘッダ … 295
10.2.2 アセンブル・リスト … 296
10.2.3 シンボル・リスト … 298
10.2.4 クロスレファレンス・リスト … 299
10.2.5 エラー・リスト … 301
10.3 リンカの出力リスト … 302
10.3.1 リンク・リスト・ファイルのヘッダ … 302
10.3.2 マップ・リスト … 304
10.3.3 パブリック・シンボル・リスト … 306
10.3.4 ローカル・シンボル・リスト … 307
10.3.5 エラー・リスト … 308
10.4 オブジェクト・コンバータの出力リスト … 309
10.4.1 エラー・リスト … 309
10.5 ライブラリアンの出力リスト … 310
10.5.1 ライブラリ情報出力リスト … 310
10.6 リスト・コンバータの出力リスト … 311
10.6.1 アブソリュート・アセンブル・リスト … 311
10.6.2 エラー・リスト … 311

第 11 章　RA78K0 の活用法 … 312
11.1 作業の効率化（EXIT ステータス機能） … 312
11.2 開発環境の整備（環境変数） … 314
11.3 プログラム実行の中断 … 315
11.4 アセンブル・リストを見やすくする … 316
11.5 プログラム起動時の手間を省く … 317
11.5.1 ソース・プログラムに制御命令を記述する … 317
11.5.2 PM plus を使用する … 317
11.5.3 パラメータ・ファイルやサブコマンド・ファイルを作成する … 318
11.6 オブジェクト・モジュールのライブラリ化 … 319

第 12 **章　エラー・メッセージ** … 320
12.1 **エラー・メッセージの概要** … 320
12.2 **構造化アセンブラ・プリプロセッサのエラー・メッセージ** … 321
12.3 **アセンブラのエラー・メッセージ** … 327
12.4 **リンカのエラー・メッセージ** … 339
12.5 **オブジェクト・コンバータのエラー・メッセージ** … 348
12.6 **ライブラリアンのエラー・メッセージ** … 352
12.7 **リスト・コンバータのエラー・メッセージ** … 356
12.8 PM plus **のエラー・メッセージ** … 360
12.8.1 **構造化アセンブラ・プリプロセッサ**（ST78K0）… 361
12.8.2 **アセンブラ**（RA78K0）… 364
12.8.3 **リンカ**（LK78K0）… 367
12.8.4 **オブジェクト・コンバータ**（OC78K0）… 370
12.8.5 **ライブラリアン**（LB78K0）… 372
12.8.6 **リスト・コンバータ**（LCNV78K0）… 373

付録 A　**サンプル・プログラム** … 374
A.1 k0main.asm … 375
A.2 k0sub.asm … 376
A.3 test1.s … 377
A.4 test2.s … 378
A.5 testinc.s … 379
A.6 st.bat … 380

付録 B　**使用上の注意** … 381

付録 C　**コマンド・オプション** … 386
C.1 **構造化アセンブラ・オプション** … 386
C.2 **アセンブラ・オプション** … 388
C.3 **リンカ・オプション** … 391
C.4 **オブジェクト・コンバータ・オプション** … 394
C.5 **ライブラリアン・オプション** … 396
C.6 **リスト・コンバータ・オプション** … 397

付録 D　**サブコマンド** … 398

付録 E　**総合索引** … 399

図の目次

図番号　タイトル,ページ

1-1　RA78K0 **アセンブラ・パッケージ** … 17
1-2　**アセンブラの流れ** … 18
1-3　**マイクロコンピュータ応用製品の開発工程** … 19
1-4　**ソフトウエアの開発工程** … 20
1-5　RA78K0 **のアセンブル工程** … 21
1-6　**アセンブルのやり直し** … 22
1-7　**既成モジュールを利用したプログラム作成** … 23
1-8　RA78K0 **によるプログラム開発手順** … 24
1-9　**ソース・モジュール・ファイルの作成** … 25
1-10　**構造化アセンブラ・プリプロセッサの機能** … 26
1-11　**アセンブラの機能** … 27
1-12　**リンカの機能** … 28
1-13　**オブジェクト・コンバータの機能** … 29
1-14　**ライブラリアンの機能** … 30
1-15　**リスト・コンバータの機能** … 31
1-16　**ディバッガの機能** … 32
2-1　**ディレクトリ構成** … 41
2-2　**ディレクトリ構成** … 42
3-1　**サンプル・プログラムの構造** … 46
3-2　RA78K0 **の実行手順** 1 … 54
3-3　RA78K0 **の実行手順** 2 … 55
3-4　ST78K0 **の実行手順** … 61
3-5　**リンク・ディレクティブ** … 63
4-1　**構造化アセンブラ・プリプロセッサの入出力ファイル** … 68
4-2　**〈構造化アセンブラオプションの設定〉ダイアログ（《出力》タブ選択時）** … 89
4-3　**〈アセンブラオプション〉ダイアログ（[アセンブラオプション** (S)**] ボタン選択時）** … 90
4-4　**〈構造化アセンブラオプションの設定〉ダイアログ（《その他》タブ選択時）** … 90
4-5　**〈オプションの編集〉ダイアログ** … 93
4-6　**〈オプションの追加〉ダイアログ** … 93
5-1　**アセンブラの入出力ファイル** … 95
5-2　**〈アセンブラオプションの設定〉ダイアログ（《出力** 1**》タブ選択時）** … 140
5-3　**〈アセンブラオプションの設定〉ダイアログ（《出力** 2**》タブ選択時）** … 141
5-4　**〈アセンブラオプションの設定〉ダイアログ（《その他》タブ選択時）** … 141
5-5　**〈オプションの編集〉ダイアログ** … 145
5-6　**〈オプションの追加〉ダイアログ** … 145
6-1　**メモリ領域名** … 151
6-2　**セグメント配置の具体例** … 155
6-3　**〈リンカオプションの設定〉ダイアログ（《出力** 1**》タブ選択時）** … 193
6-4　**〈リンカオプションの設定〉ダイアログ（《出力** 2**》タブ選択時）** … 194
6-5　**〈リンカオプションの設定〉ダイアログ（《ライブラリ》タブ選択時）** … 194
6-6　**〈リンカオプションの設定〉ダイアログ（《その他》タブ選択時）** … 195
6-7　**〈オプションの編集〉ダイアログ** … 199
6-8　**〈オプションの追加〉ダイアログ** … 199
7-1　**オブジェクト・コンバータの入出力ファイル** … 201
7-2　**インテル標準形式** … 204
7-3　**インテル拡張形式** … 205
7-4　**モトローラ** S **タイプ形式** … 211
7-5　**シンボル値のフォーマット** … 216
7-6　**〈オブジェクトコンバータオプションの設定〉ダイアログ（《出力** 1**》タブ選択時）** … 236
7-7　**〈オブジェクトコンバータオプションの設定〉ダイアログ（《出力** 2**》タブ選択時）** … 237
7-8　**〈オブジェクトコンバータオプションの設定〉ダイアログ（《その他》タブ選択時）** … 237
8-1　**ライブラリアンの入出力ファイル** … 241
8-2　**ライブラリ・ファイルの作成手順** … 243
8-3　**〈ライブラリアンオプションの設定〉ダイアログ（《出力》タブ選択時）** … 267

8-4 **〈ライブラリアンオプションの設定〉ダイアログ（《その他》タブ選択時）**… 268
8-5 **〈ライブラリ・ファイルの指定〉ダイアログ** … 270
8-6 **〈サブコマンド実行〉ダイアログ** … 271
9-1 **リスト・コンバータの入出力ファイル** … 274
9-2 **〈リストコンバータオプションの設定〉ダイアログ（《出力》タブ選択時）**… 289
9-3 **〈リストコンバータオプションの設定〉ダイアログ（《その他》タブ選択時）**… 290
B-1 **アドレスの見え方** … 383

表の目次

表番号　タイトル,ページ

1-1　**構造化アセンブラの最大性能** … 33
1-2　**アセンブラの最大性能** … 34
1-3　**リンカの最大性能** … 35
2-1　RA78K0 **アセンブラ・パッケージの供給媒体** … 37
4-1　**構造化アセンブラ・プリプロセッサの入出力ファイル** … 68
4-2　**構造化アセンブラ・オプション** … 74
5-1　**アセンブラの入出力ファイル** … 94
5-2　**アセンブラ・オプション** … 101
5-3　**アセンブラ・オプションの優先度** … 103
5-4　**タイトルとして記述可能な文字** … 122
6-1　**リンカの入出力ファイル** … 146
6-2　**セグメントの配置のグループ分け（外付け** ROM **など）** … 148
6-3　**ディレクティブの種類** … 149
6-4　**メモリ領域名指定とメモリ空間の組み合わせによるセグメントの配置** … 154
6-5　**リンカ・オプション** … 159
6-6　**リンカ・オプションの優先度** … 161
7-1　**オブジェクト・コンバータの入出力ファイル** … 200
7-2　**拡張空間に対する出力ファイルのファイル・タイプ** … 202
7-3　-ZF **オプション指定時のファイル・タイプ** … 202
7-4　**拡張テック・ヘッダ・フィールド** … 207
7-5　**チェック・サム評価のキャラクタの値** … 207
7-6　**拡張テックのデータ・ブロックのフォーマット** … 207
7-7　**拡張テックのターミネーション・ブロックのフォーマット** … 208
7-8　**拡張テックのシンボル・ブロックのフォーマット** … 209
7-9　**拡張テック・シンボル・ブロック・セクション定義フィールド** … 210
7-10　**拡張テック・シンボル・ブロック・シンボル定義フィールド** … 210
7-11　**モトローラ・ヘキサ・ファイルのレコードの種類** … 211
7-12　**各レコードの一般形** … 211
7-13　**フィールドの意味** … 212
7-14　**シンボル属性の値** … 216
7-15　**オブジェクト・コンバータ・オプション** … 220
7-16　-ZF **オプション指定時のファイル・タイプ** … 222
7-17　**拡張空間に対する** HEX **形式オブジェクト・モジュール・ファイルのファイル・タイプ** … 223
7-18　**拡張空間に対するシンボル・テーブル・ファイルのファイル・タイプ** … 224
8-1　**ライブラリアンの入出力ファイル** … 240
8-2　**ライブラリアン・オプション** … 248
8-3　**サブコマンド** … 256
9-1　**リスト・コンバータの入出力ファイル** … 273
9-2　**リスト・コンバータ起動時の指定ファイル・タイプ** … 278
9-3　**リスト・コンバータ・オプション** … 281
10-1　**構造化アセンブラ・プリプロセッサの出力ファイル** … 292
10-2　**エラーリストの出力項目の説明（構造化アセンブラ・プリプロセッサ起動時）** … 293
10-3　**アセンブラの出力リスト** … 294
10-4　**アセンブル・リスト・ファイルのヘッダの出力項目の説明** … 295
10-5　**アセンブル・リストの出力項目の説明** … 296
10-6　**シンボル・リストの出力項目の説明** … 298
10-7　**クロスレファレンス・リストの出力項目の説明** … 299
10-8　**エラーリストの出力項目の説明（アセンブラ起動時）** … 301
10-9　**リンカの出力リスト** … 302
10-10　**リンク・リスト・ファイルのヘッダの出力項目の説明** … 302
10-11　**マップ・リストの出力項目の説明** … 304
10-12　**パブリック・シンボル・リストの出力項目の説明** … 306
10-13　**ローカル・シンボル・リストの出力項目の説明** … 307
10-14　**エラーリストの出力項目の説明（リンカ起動時）** … 308

10-15 **オブジェクト・コンバータの出力リスト** … 309
10-16 **ライブラリアンの出力リスト** … 310
10-17 **ライブラリ情報出力リストの出力項目の説明** … 310
10-18 **リスト・コンバータの出力リスト** … 311
12-1 **構造化アセンブラ・プリプロセッサのエラー・メッセージ** … 321
12-2 **アセンブラのエラー・メッセージ** … 327
12-3 **リンカのエラー・メッセージ** … 339
12-4 **オブジェクト・コンバータのエラー・メッセージ** … 348
12-5 **ライブラリアンのエラー・メッセージ** … 352
12-6 **リスト・コンバータのエラー・メッセージ** … 356
12-7 **構造化アセンブラ・プリプロセッサ**（ST78K0）**用** DLL **の表示するエラー・メッセージ** … 361
12-8 **アセンブラ**（RA78K0）**用** DLL **の表示するエラー・メッセージ** … 364
12-9 **リンカ**（LK78K0）**用** DLL **の表示するエラー・メッセージ** … 367
12-10 **オブジェクト・コンバータ**（OC78K0）**用** DLL **の表示するエラー・メッセージ** … 370
12-11 **ライブラリアン**（LB78K0）**用** DLL **の表示するエラー・メッセージ** … 372
12-12 **リスト・コンバータ**（LCNV78K0）**用** DLL **の表示するエラー・メッセージ** … 373
B-1 **インテル拡張** HEX **形式** (**フラッシュ実アドレス**) **の出力例** … 384
B-2 **インテル拡張** HEX **形式** (**バンク番号** + CPU **アドレス**) **の出力例** … 385
C-1 **構造化アセンブラ・オプション** … 386
C-2 **アセンブラ・オプション** … 388
C-3 **リンカ・オプション** … 391
C-4 **オブジェクト・コンバータ・オプション** … 394
C-5 **ライブラリアン・オプション** … 396
C-6 **リスト・コンバータ・オプション** … 397
D-1 **サブコマンド一覧** … 398

第 1 章　概説

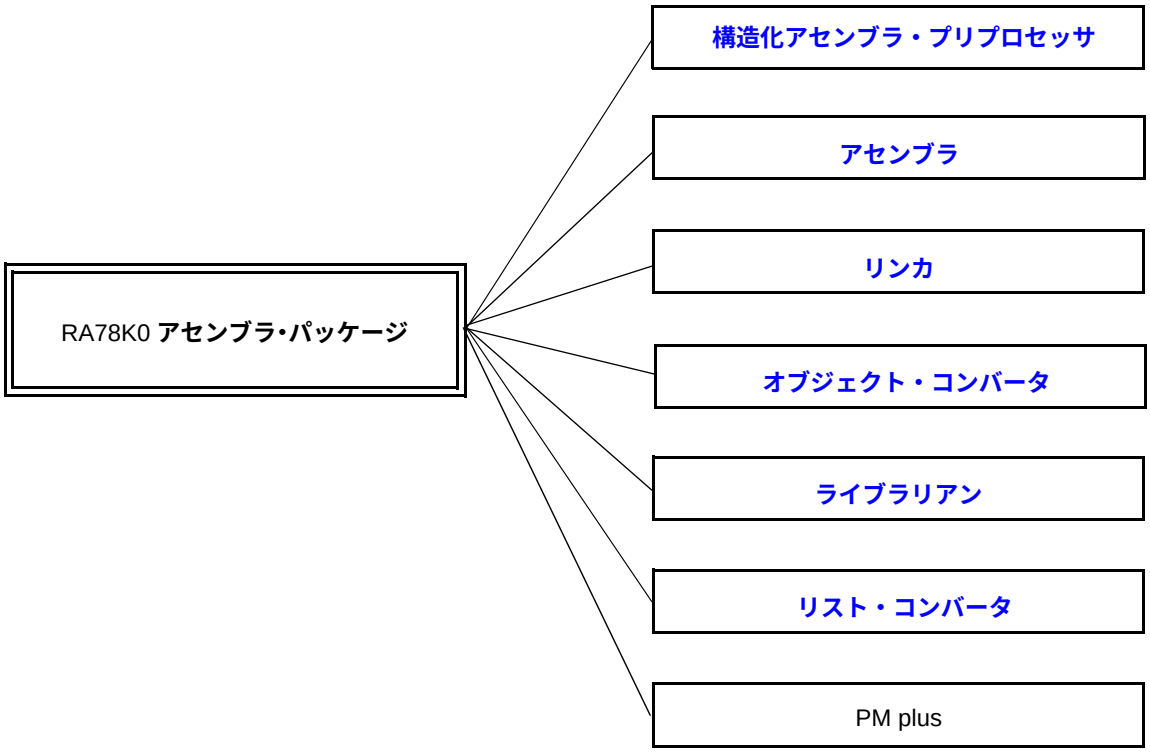
この章では，マイクロコンピュータの開発における RA78K0 の役割など，RA78K0 全体の機能概要について説明します。

1.1　概要

RA78K0 アセンブラ・パッケージ（以下，RA78K0）は，78K0 シリーズのアセンブリ言語で記述されたソース・プログラムを機械語に変換する言語処理プログラムの総称です。

RA78K0 には，構造化アセンブラ・プリプロセッサ，アセンブラ，リンカ，オブジェクト・コンバータ，ライブラリアン，リスト・コンバータといった 6 つのプログラムの他に，エディット，コンパイル／アセンブル，リンクからディバグまでの一連の操作を Windows 上で簡単に行うことを可能にする PM plus も標準添付されています。

図 1-1 RA78K0 アセンブラ・パッケージ



1.1.1 アセンブラとは

(1) アセンブリ言語と機械語

アセンブリ言語は，マイクロコンピュータ用の最も基本的なプログラミング言語です。

マイクロコンピュータに仕事をさせる際には，プログラムやデータが必要となります。これらの情報をユーザがプログラミングし，マイクロコンピュータのメモリに書き込みます。

なお，マイクロコンピュータが扱うことのできるプログラムやデータは 2 進数の集まりであり，これを機械語と呼びます。機械語でプログラムを作ることは，ユーザにとって覚えにくく，また誤りを起こしやすいものです。そこで機械語の意味をユーザにとって理解しやすい英語の略記号で表し，この記号を使ってプログラムを作成する方法があります。この記号によるプログラムの基本的な言語体系をアセンブリ言語と呼びます。

ただし，マイクロコンピュータが扱えるプログラミング言語は機械語に限定されるため，アセンブリ言語で作成したプログラムを機械語に翻訳するプログラムが必要となります。これをアセンブラと呼びます。

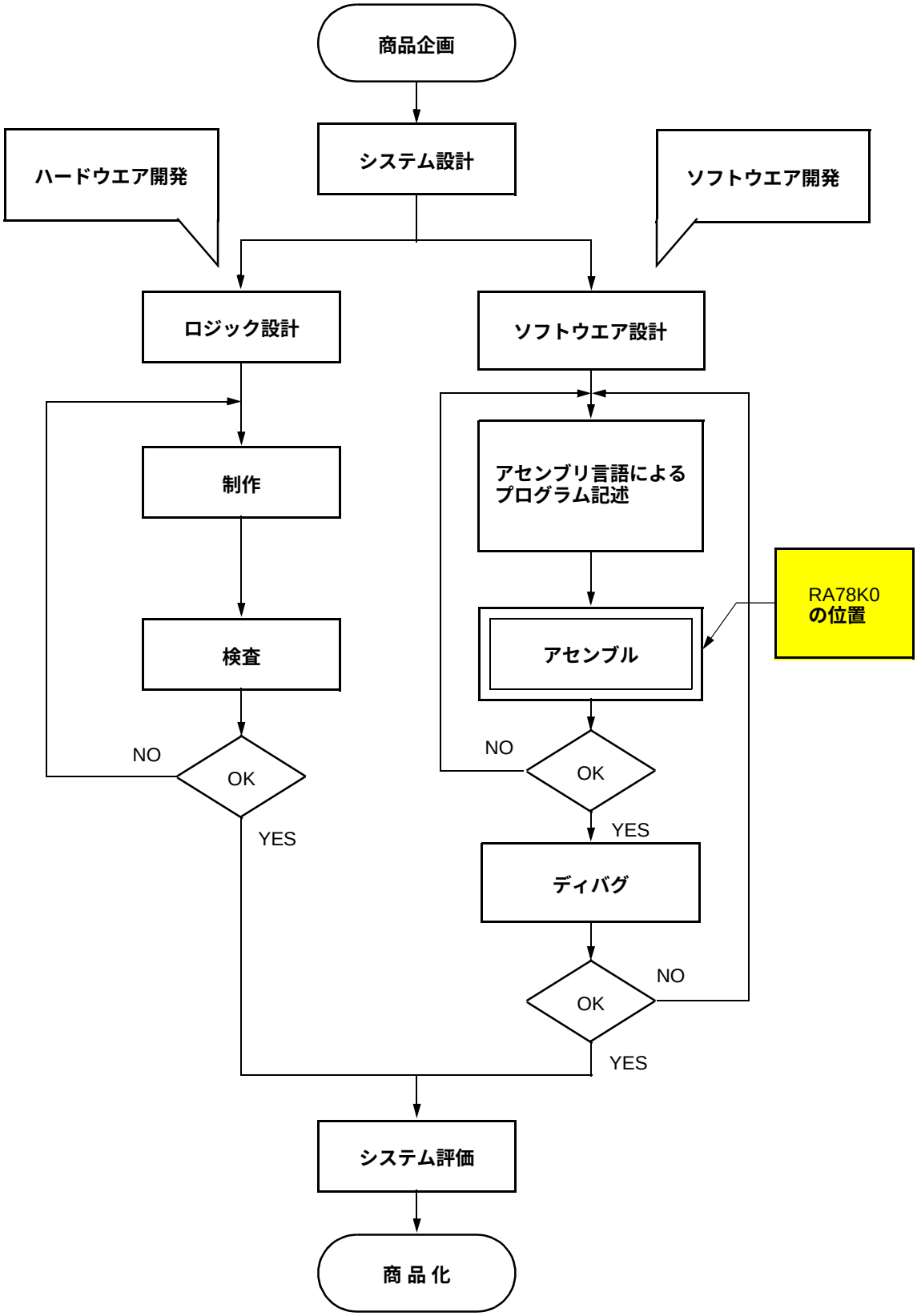
図 1-2 アセンブラの流れ

1.1.2 開発工程と RA78K0

(1) マイクロコンピュータ応用製品の開発と RA78K0 の役割

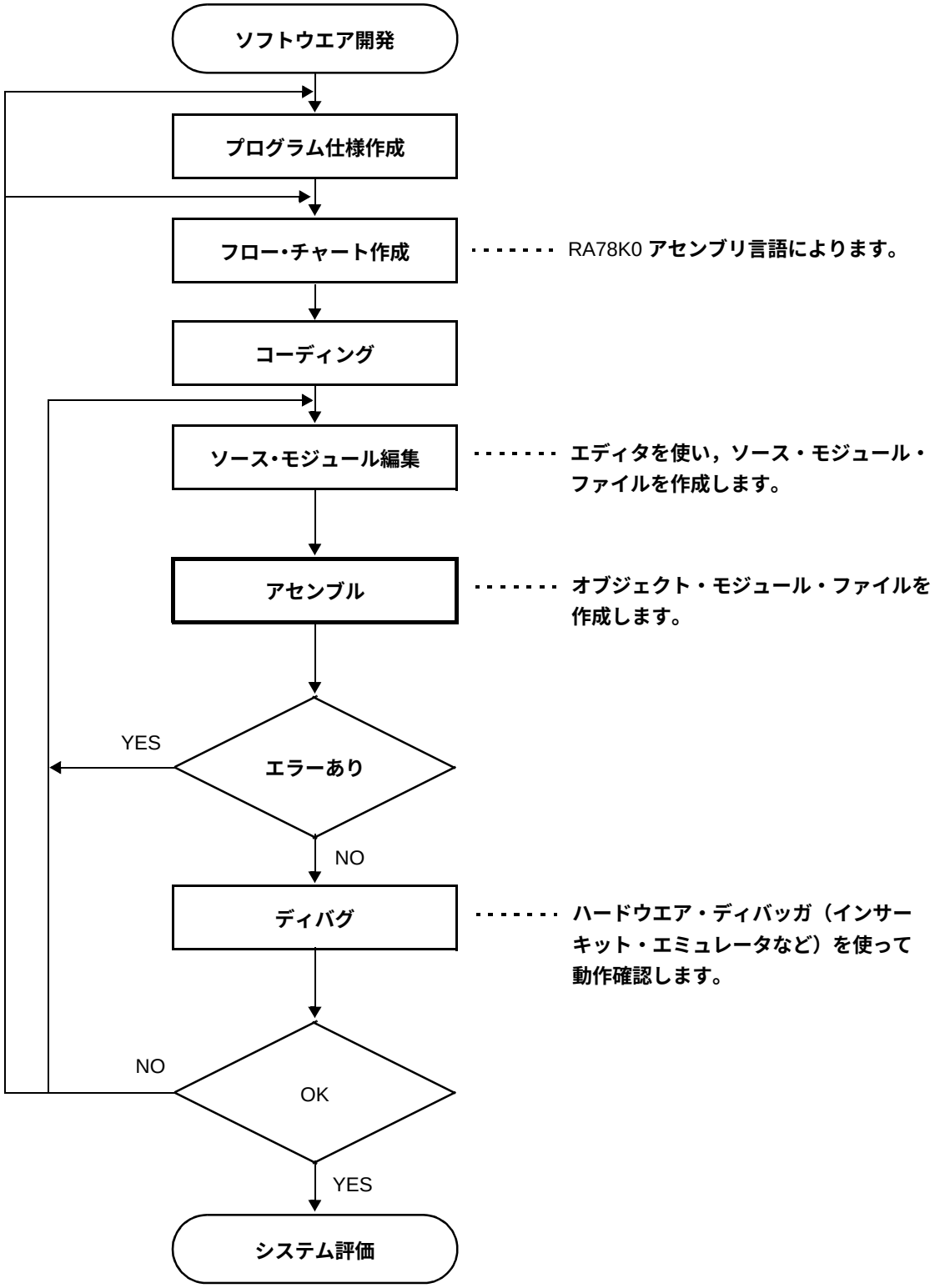
“ 製品開発におけるアセンブリ言語によるプログラム記述 ” の位置付けを図 1-3 に示します。

図 1-3 マイクロコンピュータ応用製品の開発工程



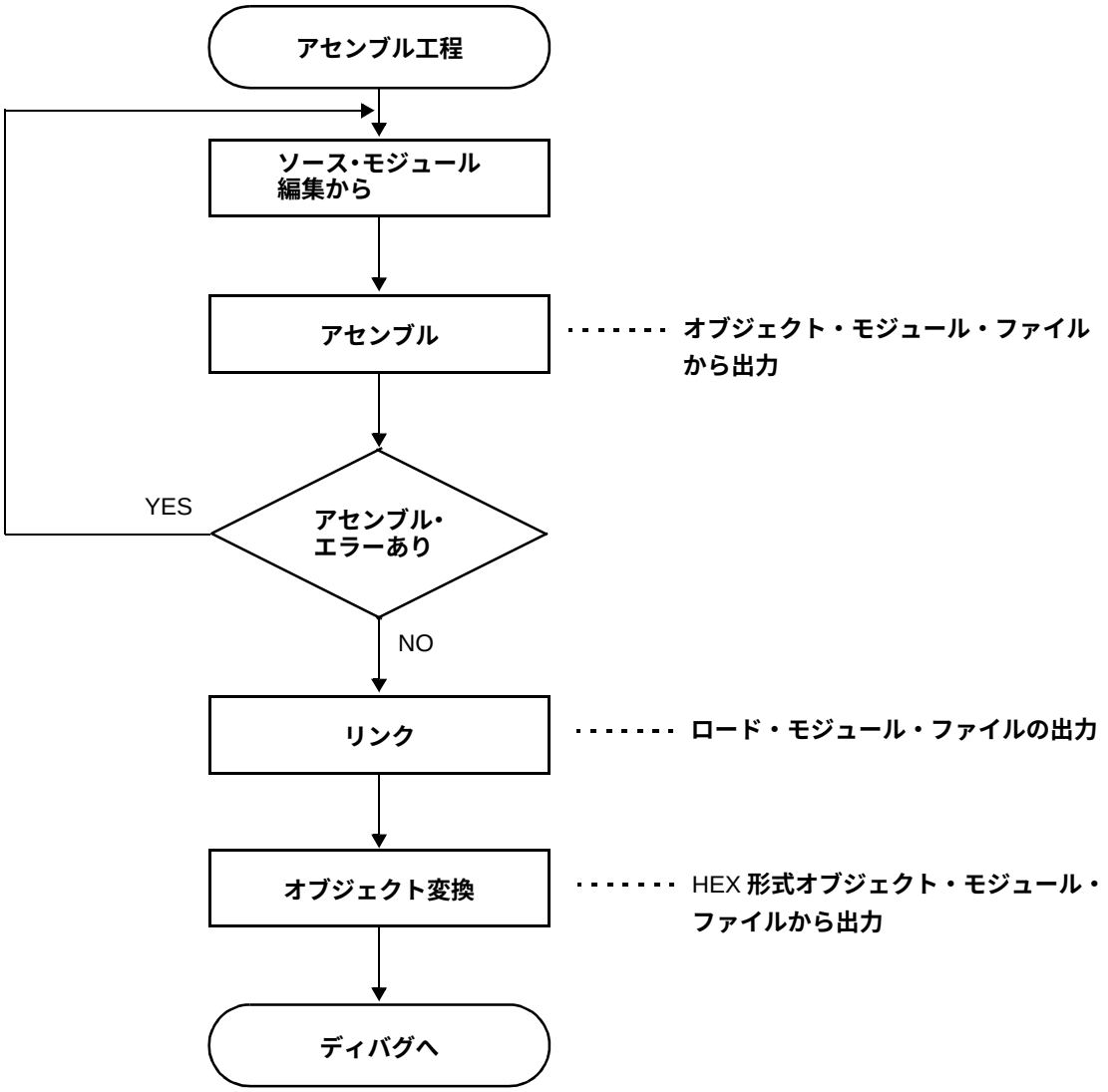
ソフトウエアの開発工程をもう少し詳しく，図 1-4 に示します。

図 1-4 ソフトウエアの開発工程



さらに，アセンブル工程に，RA78K0 を当てはめてみます。

図 1-5 RA78K0 のアセンブル工程



1.1.3 リロケータブル・アセンブラとは

アセンブラが変換した機械語は，マイクロコンピュータのメモリに書き込まれて使用されます。このとき，変換された機械語がメモリのどこに書き込まれるかが，決定されていなければなりません。

したがって，アセンブラの変換する機械語には，「各機械語がメモリのどのアドレスに配置されるべきか」という情報が付加されています。

機械語を配置するアドレスの決定方法により，アセンブラは“ アブソリュート・アセンブラ ”と“ リロケータブル・アセンブラ ”に大別されます。

- アブソリュート・アセンブラ
  アセンブリ言語から変換した機械語は，絶対的（アブソリュート）なアドレスに配置されます。
- リロケータブル・アセンブラ
  アセンブリ言語から変換した機械語には，一時的なアドレスが与えられます。
  なお，絶対的なアドレスは，リンカにより配置されます。

アブソリュート・アセンブラで 1 つのプログラムを作成する際には，原則として一度にプログラミングしなければなりません。しかし，大きなプログラムを 1 つのまとまりとして作成した場合，プログラムが複雑になり，また保守する際にもプログラムの解析が困難になります。

そこで，プログラム 1 つ 1 つの機能単位ごとにいくつかのサブプログラム（モジュール）に分割して，プログラム開発を行います。これをモジュラ・プログラミングと呼びます。

1.1.4 リロケータブル・アセンブラの利点

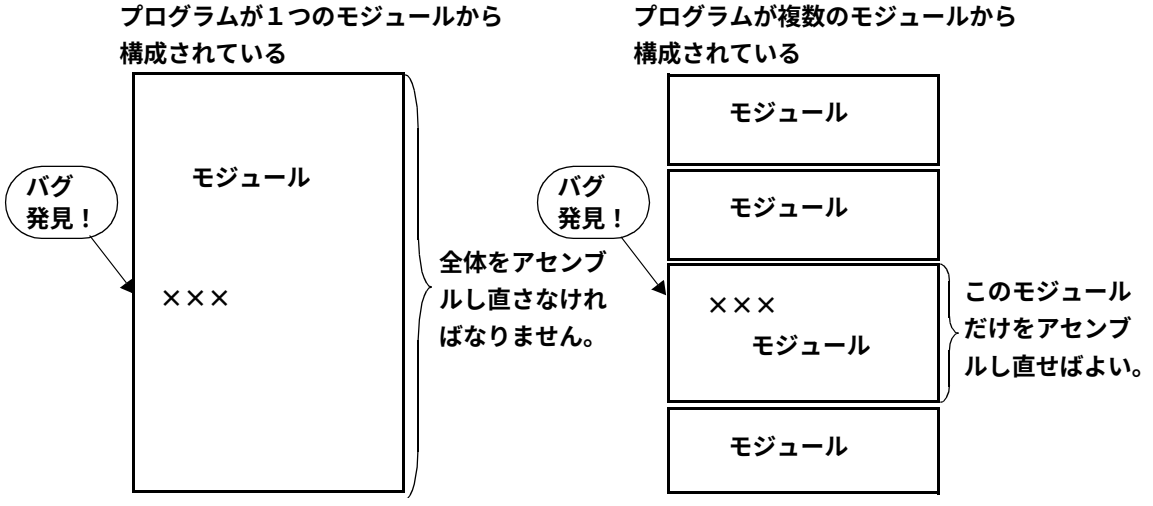
リロケータブル・アセンブラは，モジュラ・プログラミングに適したアセンブラであり，モジュラ・プログラミングを行うことにより，次の利点があげられます。

(1) 開発効率が上がる

大きなプログラムを一度にプログラミングすることは困難です。このような場合，プログラムを 1 つ 1 つの機能ごとにモジュール分割すれば，複数の人数でプログラムの並行開発ができ，開発効率が上がります。

また，プログラム中にバグが発見された場合，一部の修正を行うために全プログラムをアセンブルすることなく，修正が必要なモジュールだけアセンブルし直すことができ，ディバグ時間を短くできます。

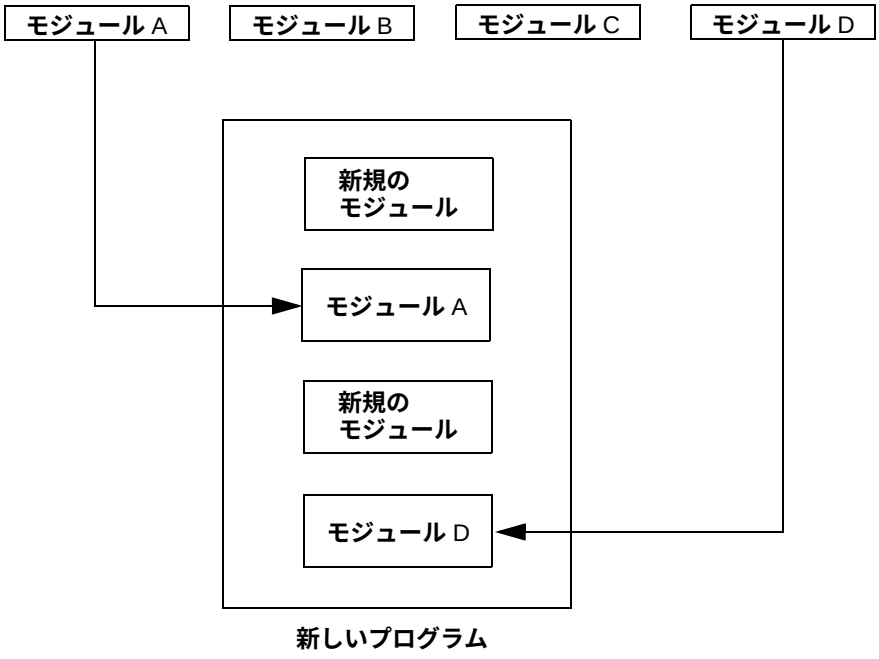
図 1-6 アセンブルのやり直し



(2) **資源の活用ができる**

以前に作成された信頼性，汎用性の高いモジュールを，他のプログラムの開発に再利用できます。このような汎用性の高いモジュールを蓄積することにより，新規にプログラム開発する部分を少なくすることができます。

図 1-7 既成モジュールを利用したプログラム作成

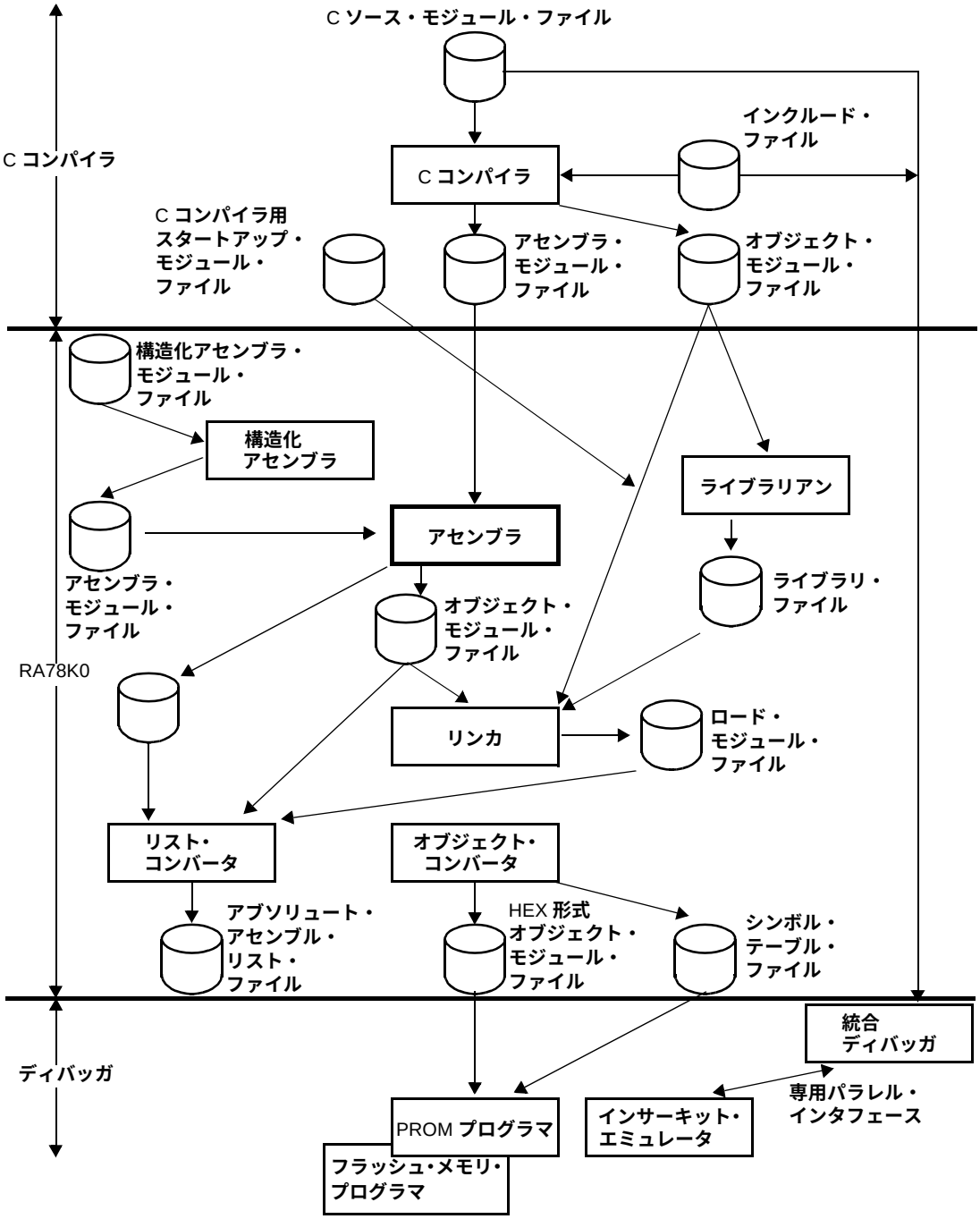


1.2 RA78K0 の機能概要

RA78K0 における一般的なプログラム開発手順を，図 1-8 に示します。プログラムの開発は，基本的にアセンブラ→リンカ→オブジェクト・コンバータを使用して行います。

なお，以降は，アセンブラ，リンカ，オブジェクト・コンバータなどの各プログラムを総称して「RA78K0」といい，アセンブラ・プログラムのことを「アセンブラ」といいます。

図 1-8 RA78K0 によるプログラム開発手順



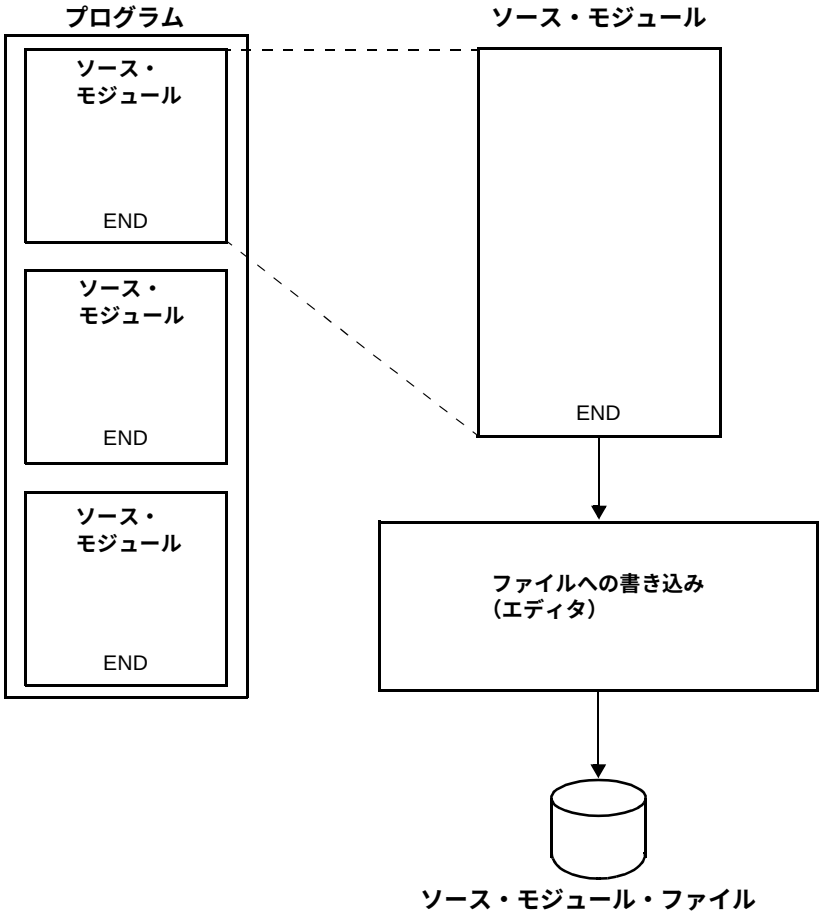
1.2.1 エディタによるソース・モジュール・ファイルの作成

1 つのプログラムを，機能的にいくつかのモジュールに分割します。1 つのモジュールは，コーディングの単位となるもので，またアセンブラの入力単位ともなります。

アセンブラの入力単位となるモジュールを，ソース・モジュールと呼びます。各ソース・モジュールのコーディング終了後，エディタを使ってソース・モジュールをファイルに書き込みます。こうしてできたファイルを，ソース・モジュール・ファイルと呼びます。

ソース・モジュール・ファイルは，アセンブラの入力ファイルとなります。

図 1-9 ソース・モジュール・ファイルの作成

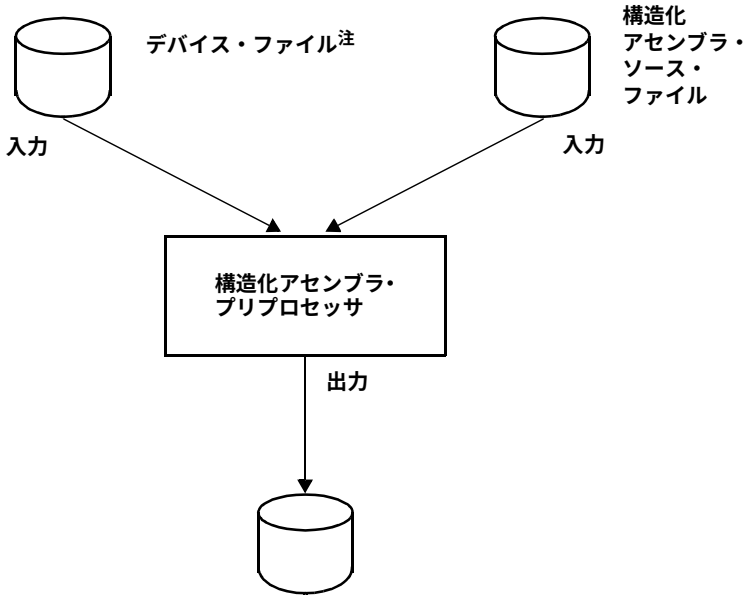


1.2.2 構造化アセンブラ・プリプロセッサ

構造化アセンブラ・プリプロセッサは，アセンブリ言語で構造化プログラミングを実現するためのプログラムです。構造化アセンブリ言語で書かれたソース・プログラムを入力し，アセンブラのソース・プログラムを入力します。

構造化アセンブラ・プリプロセッサや構造化アセンブリ言語については，「RA78K0 構造化アセンブリ言語編」のユーザーズ・マニュアルを参照してください。

図 1-10 構造化アセンブラ・プリプロセッサの機能



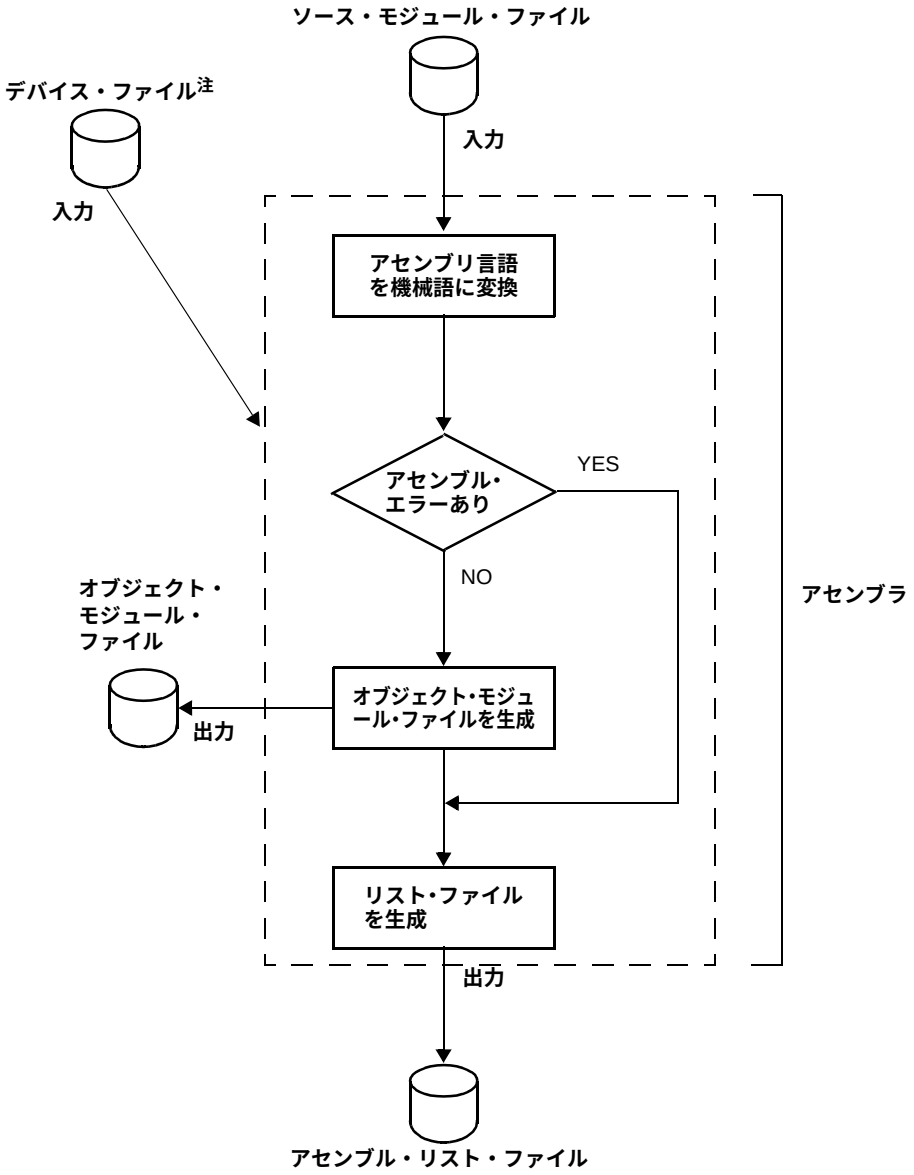
注 デバイス・ファイルはオンライン・デリバリ・サービス（ODS）から別途入手してください。下記 URL の「開発ツールダウンロード（ODS）」からジャンプできます。

http://www.necel.com/micro/ods/jpn/tool/DeviceFile/list.html

1.2.3 アセンブラ

アセンブラは，ソース・モジュール・ファイルを入力し，アセンブリ言語を 2 進数の集まり（機械語）に変換するプログラムです。ソース・モジュールの中に記述ミスを発見した場合は，アセンブル・エラーを出力します。アセンブル・エラーがない場合には，機械語情報と各機械語がメモリ中のどのアドレスに配置されるべきか，などの配置情報を含むオブジェクト・モジュール・ファイルを出力します。また，アセンブル時の情報をアセンブル・リスト・ファイルとして出力します。

図 1-11 アセンブラの機能



注 デバイス・ファイルはオンライン・デリバリ・サービス（ODS）から別途入手してください。下記 URL の「開発ツールダウンロード（ODS）」からジャンプできます。

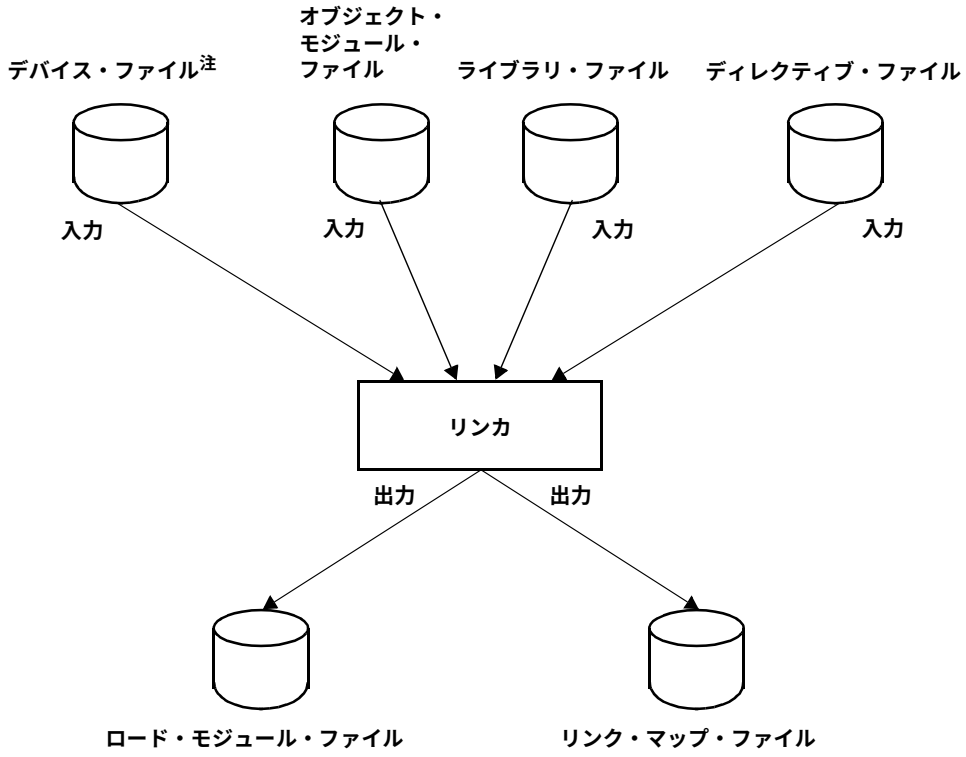
http://www.necel.com/micro/ods/jpn/tool/DeviceFile/list.html

1.2.4 リンカ

リンカは，コンパイラの出力したオブジェクト・モジュール・ファイル，またはアセンブラの出力したオブジェクト・モジュール・ファイルを複数個入力し，1 つのロード・モジュール・ファイルを出力します（オブジェクト・モジュール・ファイルが 1 つの場合でもリンクしなければなりません）。

この際リンカは，入力されたモジュール中のリロケータブル・セグメントの配置アドレスを決定します。これにより，リロケータブル・シンボルや外部参照シンボルの値を決定し，ロード・モジュール・ファイルに正しい値を埋め込みます。

図 1-12 リンカの機能



注　デバイス・ファイルはオンライン・デリバリ・サービス（ODS）から別途入手してください。下記 URL の「開発ツールダウンロード（ODS）」からジャンプできます。

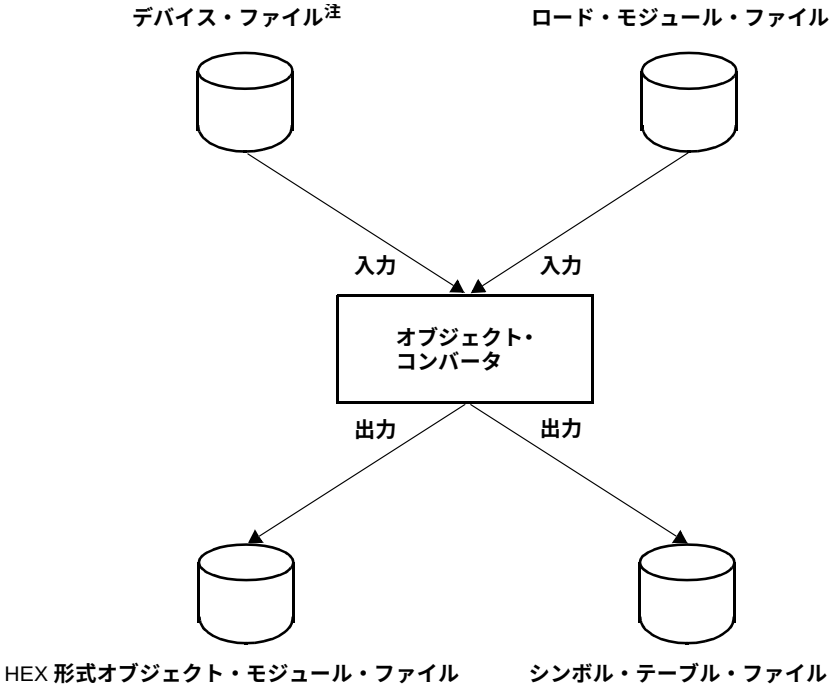
http://www.necel.com/micro/ods/jpn/tool/DeviceFile/list.html

1.2.5 オブジェクト・コンバータ

オブジェクト・コンバータは，リンカの出力したロード・モジュール・ファイルを入力し，ファイル形式を変換します。そして，その結果を HEX 形式オブジェクト・モジュール・ファイルとして出力します。

また，シンボリック・ディバグ時に必要となるシンボル情報をシンボル・テーブル・ファイルとして出力します。

図 1-13 オブジェクト・コンバータの機能



注　デバイス・ファイルはオンライン・デリバリ・サービス（ODS）から別途入手してください。下記 URL の「開発ツールダウンロード（ODS）」からジャンプできます。

http://www.necel.com/micro/ods/jpn/tool/DeviceFile/list.html

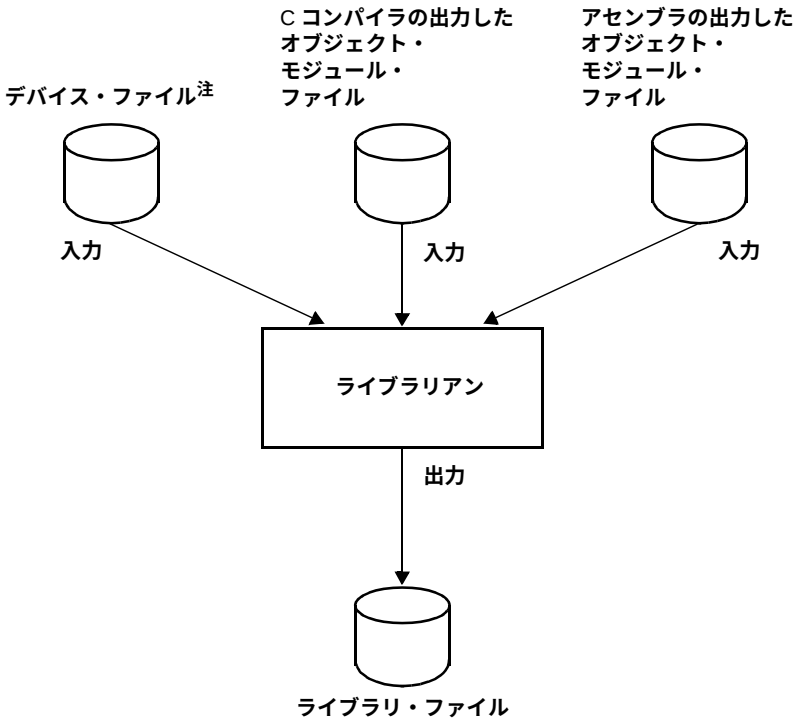
1.2.6 ライブラリアン

汎用性のある，インタフェースの明確なモジュールは，ライブラリ化しておくと便利です。ライブラリ化を行うことにより，多くのオブジェクト・モジュールも 1 つのファイルになり，扱いやすくなります。

リンカには，ライブラリ・ファイルの中から必要なモジュールだけを取り出してリンクする機能があります。したがって，複数のモジュールを 1 つのライブラリ・ファイルに登録しておけば，リンク時に必要なモジュール・ファイル名を個別に指定する必要がなくなります。

ライブラリ・ファイルの作成と更新にはライブラリアンを使用します。

図 1-14 ライブラリアンの機能



注　デバイス・ファイルはオンライン・デリバリ・サービス（ODS）から別途入手してください。下記 URL の「開発ツールダウンロード（ODS）」からジャンプできます。

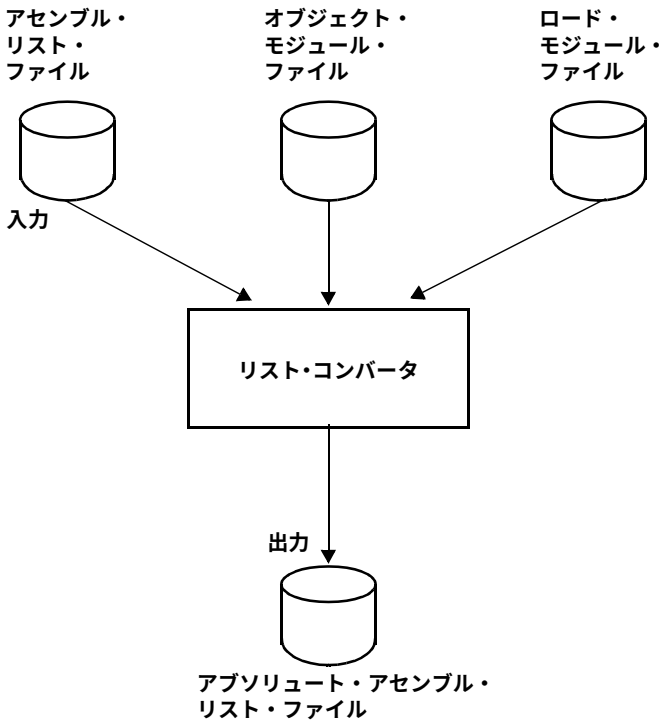
http://www.necel.com/micro/ods/jpn/tool/DeviceFile/list.html

1.2.7 リスト・コンバータ

リスト・コンバータは，アセンブラの出力したオブジェクト・モジュール・ファイルとアセンブル・リスト・ファイル，およびリンカの出力したロード・モジュール・ファイルを入力し，アブソリュート・アセンブル・リスト・ファイルを出力します。

リロケータブルなアセンブル・リストでは，リスト中のアドレスやリロケータブルな値は，実際の値とは異なるなどの欠点があります。しかし，アブソリュート・アセンブル・リストでは，これらが解決されるので，ディバグやプログラムの保守が容易になります。

図 1-15 リスト・コンバータの機能



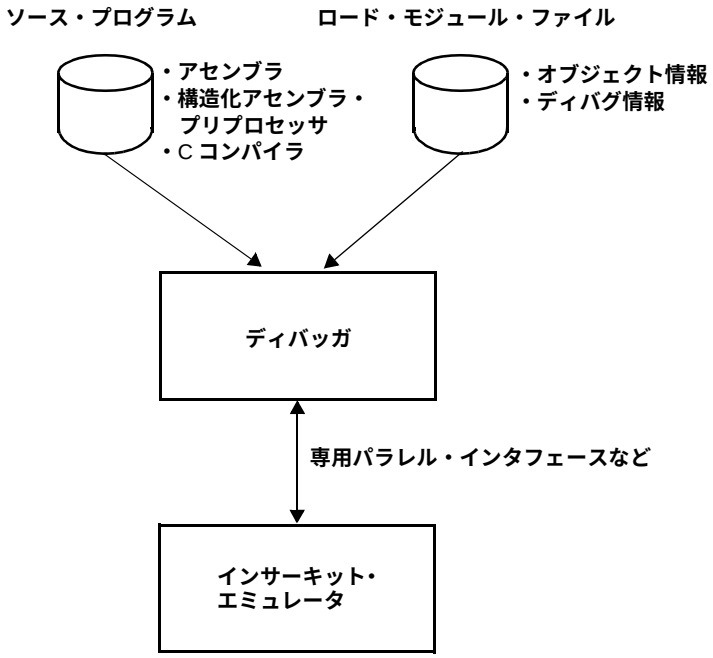
1.2.8 ディバッガ

78K0 シリーズ用ディバッガは，ソース・プログラム，レジスタ，メモリなどの情報をそれぞれのウインドウに表示してディバグを行うソフトウエア・ツールです。

リンカが出力したロード・モジュール・ファイルを，ターゲット・システムの IE（インサーキット・エミュレータ）にダウン・ロードし，さらにソース・プログラム・ファイルを読み込むことでソース・レベルでのディバグが可能になります。

ディバッガ，IE は，別パッケージ（別売）になります。

図 1-16 ディバッガの機能



1.3 プログラム開発をはじめる前に

実際にプログラム開発をはじめる前に，次のことをご参照願います。

1.3.1 RA78K0 の最大性能

(1) 構造化アセンブラの最大性能

表 1-1 構造化アセンブラの最大性能

項目	制限値
一行の長さ （LF，CR は含まない）	2048 文字
#define 疑似命令の登録シンボル数 （予約語は除く）	512 個
#define 疑似命令の登録シンボルの文字長	31 文字
制御文のネスティング・レベル	31 レベル
#ifdef 疑似命令のネスティング・レベル	8 レベル
#defcallt 疑似命令	32 個
#include 疑似命令のネスティング	できません
#define 疑似命令で再定義可能な回数	31 回
連続代入できるオペランド数	33 個[注 1]
論理演算子のオペランド	17 個[注 2]
-D オプションで定義可能なシンボル数	30 個
-I オプションで指定可能なインクルード・ファイル・パス数	64 個

注 1 以下のようになります。

S1 = S2 = … S32 = S33

シンボル数は 33 個，“ = ” は 32 個まで記述可能です。

注 2 以下のようになります。

式 1&& 式 2&& … && 式 16&& 式 17

式は 17 個，“ &&（または ||）” は 16 個まで記述可能です。

(2) アセンブラの最大性能

表 1-2 アセンブラの最大性能

項目	最大性能
シンボル数（ローカル + パブリック）	65535 個
クロスレファレンス・リスト出力可能なシンボル数	65534 個[注 1]
1 マクロ参照のマクロ・ボディ最大サイズ	1 M バイト
全マクロ・ボディ合計のサイズ	10 M バイト
1 ファイル中のセグメント数	256 個
1 ファイル中のマクロ，インクルード指定	10000 個
1 インクルード・ファイル中のマクロ，インクルード指定	10000 個
リロケーション情報[注 2]	65535 個
行番号情報	65535 個
1 ファイル中の BR 疑似命令数	32767 個
ソース 1 行の文字数	2048 文字[注 3]
シンボル長	256 文字
スイッチ名の定義数[注 4]	1000 個
スイッチ名の文字長[注 4]	31 文字
セグメント名の文字長	8 文字
モジュール名（NAME 疑似命令）の文字長	8 文字
MACRO 疑似命令の仮パラメータ数	16 個
マクロ参照の実パラメータ数	16 個
IRP 疑似命令の実パラメータ数	16 個
マクロ・ボディ内のローカル・シンボル数	64 個
マクロ展開のローカル・シンボル数合計	65535 個
マクロ（マクロ参照、REPT 疑似命令，IRP 疑似命令）のネスト数	8 レベル
TITLE 制御命令，-LH オプションで指定可能な文字数	60 文字[注 5]
SUBTITLE 制御命令で指定可能な文字数	72 文字
1 ファイル中のインクルード・ファイルのネスト数	8 レベル
条件アセンブルのネスト数	8 レベル
-I オプションで，指定可能なインクルード・ファイル・パス数	64 個
-D オプションで定義可能なシンボル数	30 個

注 1　モジュール名，セクション名の個数を引いた数です。

メモリを使用します。メモリがなければファイルを使用します。

注 2　アセンブラでシンボル値が解決できない場合に，リンカに渡す情報のことです。
たとえば，MOV 命令で外部参照シンボルを参照した場合，リロケーション情報が 2 個，.rel ファイルに生成されます。

注 3　復帰／改行コードを含みます。1 行に 2049 文字以上記述された場合，ワーニング・メッセージを出力し，2049 文字以降は無視します。

注 4　スイッチ名は SET/RESET 疑似命令で真 / 偽に設定され，$IF などで使用されるものです。

注 5　アセンブル・リスト・ファイルの 1 行の文字数指定（X とします）が 119 文字以下の場合，"X-60" 文字以内とします。

(3) リンカの最大性能

表 1-3 リンカの最大性能

項目	最大性能
シンボル数（ローカル + パブリック）	65535 個
同一セグメントの行番号情報	65535 個
セグメント数	65535 個
入力モジュール	1024 個
メモリ領域名の文字長	31 文字
メモリ領域数	100 個[注]
-B オプションで指定可能なライブラリ・ファイル数	10 個
-I オプションで指定可能なライブラリ・ファイル・パス数	64 個

注　ディフォールトで定義されているものを含みます。

1.4 RA78K0 の特徴

RA78K0 は，次の特徴的な機能を備えています。

(1) マクロ機能

ソース・プログラム中で同じ命令群を何回も記述する場合，その一連の命令群を，1 つのマクロ名に対応させてマクロ定義をすることができます。

マクロ機能を利用することにより，コーディングの効率を上げることができます。

(2) 分岐命令の最適化機能

分岐命令自動選択疑似命令（BR 疑似命令）を備えています。

メモリ効率のよいプログラムを生成するためには，分岐命令の分岐先範囲に応じたバイトの分岐命令を記述する必要があります。しかし，分岐先範囲をいちいち意識して，分岐命令を記述することは面倒です。BR 疑似命令を記述することにより，アセンブラが分岐先範囲に応じて適切な分岐命令のコードを生成します。これを分岐命令の最適化機能といいます。

(3) 条件付きアセンブル機能

ソース・プログラムの一部分を条件によりアセンブルする／しないを設定できます。

ソース・プログラム中にディバグ文などを記述した場合，ディバグ文を機械語に変換する／しないを条件付きアセンブルのスイッチ設定により選択できます。ディバグ文が必要なくなった場合でも，ソース・プログラムに大幅な変更を加えることなくアセンブルできます。

第 2 章　製品概要とインストール方法

この章では，RA78K0 の提供媒体に格納されているファイル群をユーザの開発環境（ホスト・マシン）上にインストールする際の手順，およびユーザの開発環境上からアンインストールする際の手順について説明します。

2.1　ホスト・マシンと供給媒体

この RA78K0 アセンブラ・パッケージは，表 2-1 に示す開発環境に対応しています。また，ホスト・マシンにより，供給媒体が異なります。

表 2-1 RA78K0 アセンブラ・パッケージの供給媒体

ホスト・マシン	OS	供給媒体
IBM PC/AT 互換機	日本語 Windows（98/Me/2000/XP/NT™4.0）注 英語 Windows（98/Me /2000/XP/NT4.0）注	CD-ROM
HP9000 シリーズ 700	HP-UX™（Rel.10.10 以降）	CD-ROM
SPARCstationシリーズ	SunOS™（Rel.4.1.4 以降） Solaris™（Rel.2.5.1 以降）	

注　アセンブラを Windows 上で使用するためには，PM plus が必要です。
PM plus を使用しない場合は，DOS プロンプト（Windows98/Me），またはコマンド・プロンプト（Windows2000/XP/NT4.0）で RA78K0 アセンブラ・パッケージに含まれる各ツールを起動できます。

2.2 インストール

以下に，RA78K0 の提供媒体に格納されているファイル群をホスト・マシン上にインストールする手順について説明します。

2.2.1 Windows 版のインストール

(1) Windows の起動

ホスト・マシン，周辺機器などの電源を投入し，Windows を起動します。

(2) 提供媒体のセット

RA78K0 の提供媒体をホスト・マシンの該当デバイス装置（CD-ROM ドライブ）にセットすることにより，セットアップ・プログラムが自動実行します。

以降，モニタ画面に表示されるメッセージに従ってインストール作業を実行してください。

注意 セットアップ・プログラムが自動実行しない場合には，フォルダ RA78K0\DISK1 に格納されている SETUP.EXE を起動します。

(3) ファイル群の確認

Windows の標準アプリケーション Explorer などを用いて，RA78K0 の提供媒体に格納されていたファイル群がホスト・マシン上にインストールされたことを確認します。

なお，各フォルダについての詳細は，「2.4 ディレクトリ構成」を参照してください。

2.2.2 UNIX 版のインストール

ここでは /nectools/bin にインストールするものと仮定して説明を行っています。

(1) ログイン

ホスト・マシンにログインします。

(2) ディレクトリの移動

インストール・ディレクトリへ移動します。

```
% cd  /nectools/bin
```

(3) 提供媒体のセット

CD-ROM を CD ドライブにセットします。

(4) 各種ファイルのコピー

cp コマンドを実行し，CD-ROM から各種ファイルをコピーします（CD-ROM がマウントされていることを確認してからコピーしてください）。

(5) 環境変数 PATH の設定

環境変数 PATH に /nectools/bin を追加します。

2.3 デバイス・ファイルのインストール

デバイス・ファイルはオンライン・デリバリ・サービス（ODS）から別途入手してください。下記 URL の「開発ツールダウンロード（ODS）」からジャンプできます。

http://www.necel.com/micro/ods/jpn/tool/DeviceFile/list.html

2.3.1 Windows 版のインストール

デバイス・ファイルのインストールには，“ デバイスファイルインストーラ ” を使用します。“ デバイスファイルインストーラ ” は，RA78K0 とともにインストールされます。

2.3.2 UNIX 版のインストール

デバイス・ファイルは，-y オプションにより，ディレクトリを指定するか（例：-y/nectools/dev），アセンブラの実行形式のあるディレクトリ（例：/nectools/bin）にコピーしてご使用ください。

2.3.3 デバイス・ファイルのレジストリ登録

デバイス・ファイルがすでにインストールされている場合，RA78K0 のインストール実行中に，デバイス・ファイルのレジストリ登録を促すメッセージが表示されることがあります。

現 32 ビット環境で使用する場合は，RA78K0（Ver.3.30 以前，16 ビット環境）で使用していたデバイス・ファイルをレジストリ（32 ビット環境）に登録してください。

なお，レジストリへの登録は，RA78K0 のインストールが完了したあと，“ デバイスファイルインストーラ ” を使用することによっても可能です。

レジストリ登録の手順は次のとおりです。

(1) “ デバイスファイルインストーラ ” の起動

(2) ソース選択

［参照 (R)］ボタンをクリックして 16 ビット環境で使用していた “ NECDEV.INI ” を選択します。

ソース・リスト・ボックスに表示されているデバイス・ファイルからレジストリに登録したいものを選択します。

(3) 移行

［移行 (M)］ボタンをクリックすることでレジストリ（32 ビット環境）に登録されます。

2.4 ディレクトリ構成

2.4.1 Windows 版のディレクトリ構成

インストール時に表示される標準ディレクトリは，Windows システムのあるドライブの“ NECTools32\ ”です。インストール・ディレクトリ下の構成は，次のようになります。ただし，ドライブやインストール・ディレクトリはインストール時に変更可能です。なお，PM plus と RA78K0 は同じディレクトリにインストールしてください。

図 2-1 ディレクトリ構成

\NECTools32\

bin\	
st78k0.exe	構造化アセンブラ・プリプロセッサ
ra78k0.exe	アセンブラ
lk78k0.exe	リンカ
oc78k0.exe	オブジェクト・コンバータ
lb78k0.exe	ライブラリアン
lcnv78k0.exe	リスト・コンバータ
lb78k0e.exe	PM plus 環境のライブラリ本体と DLL 間インタフェース
lb78k0p.exe	単体起動用ライブラリアン
ra78k0.is*	アセンブラが使用するファイル
*78k0p.dll	PM plus 用ツール DLL 本体
*78k0.hlp	コマンド行起動時のヘルプ・ファイル（テキスト・ファイル）
doc\	ユーザーズ・マニュアル，製品添付文章
hlp\	オンライン・マニュアル
setup\	インストール，およびアンインストール用情報ファイル
smp78k0\ra78k0\	
k0main.asm	アセンブラ・サンプル・プログラム
k0sub.asm	アセンブラ・サンプル・プログラム
ra.bat	アセンブラ・サンプル・プログラム用バッチ・ファイル
readme.doc	サンプル・プログラム，バッチ・ファイルの説明（テキスト・ファイル）
test1.s	構造化アセンブラ・サンプル・プログラム
test2.s	構造化アセンブラ・サンプル・プログラム
testinc.s	構造化アセンブラ・サンプル・プログラム
st.bat	構造化アセンブラ・サンプル・プログラム用バッチ・ファイル

備考　このマニュアルでは，セットアップ・プログラムのディフォールトの指示に従って，ディフォールトのプログラム・フォルダ名“ NECTools32 ”で，標準ディレクトリにインストールしたこととして説明します。

2.4.2 UNIX 版のディレクトリ構成

インストール後のファイル構成は，次のとおりです。ここでは，/nectools/bin にインストールしたものと仮定します。

図 2-2 ディレクトリ構成

```
/nectools/
└─ bin/
```

st78k0	構造化アセンブラ・プリプロセッサ
ra78k0	アセンブラ
lk78k0	リンカ
oc78k0	オブジェクト・コンバータ
lb78k0	ライブラリアン
lcnv78k0	リスト・コンバータ
*.hlp	各プログラムに対応したヘルプ・ファイル（テキスト・ファイル）
ra78k0.is*	アセンブラが使用する命令セットを定義したテーブル・ファイル
*.asm , *.s	インストール確認用サンプル・プログラム
*.sh	インストール確認用バッチ・ファイル
*.sh	インストール確認用バッチ・ファイル
readme.doc	インストール確認用シェル・ファイルの使用説明（テキスト・ファイル）

C コンパイラ，統合ディバッガ，システム・シミュレータ，デバイス・ファイルは，アセンブラと同じディレクトリにインストールすることをお勧めします。

2.5 アンインストール手順

2.5.1 Windows 版のアンインストール

ここでは，ホスト・マシンにインストールされているファイル群をアンインストールする際の手順について説明します。

(1) Windows の起動

ホスト・マシン，および周辺機器などの電源を投入し，Windows を起動します。

(2) ［コントロールパネル］ウインドウのオープン

［スタート］ボタンの [設定 (S)] メニューから [コントロールパネル (C)] を選択し，［コントロールパネル］ウインドウを開きます。

(3) ［アプリケーションの追加と削除］ウインドウのオープン

［コントロールパネル］ウインドウの［アプリケーションの追加と削除］アイコンをダブル・クリックし，［アプリケーションの追加と削除］ウインドウを開きます。

注意　Windows XP の場合，［アプリケーションの追加と削除］は [プログラムの追加と削除] となります。

(4) RA78K0 の削除

［アプリケーションの追加と削除］ウインドウの［セットアップと削除］タブに表示されているインストール済みソフトウエア一覧の中から “ NEC RA78K0 78K/0 アセンブラ Vx.xx ” を選択したあと，［追加と削除 (R) ］ボタンを押下してください。

なお，［ファイル削除の確認］ウインドウがオープンした場合は，［はい (Y) ］ボタンを押下してください。

(5) ファイル群の確認

Windows の標準アプリケーション Explorer などを用いて，ホスト・マシンにインストールされていたファイル群がアンインストールされたことを確認してください。なお，各フォルダについての詳細は，「2.4 ディレクトリ構成」を参照してください。

2.5.2 UNIX 版のアンインストール

「2.2.2 UNIX 版のインストール」でコピーしたファイルを，rm コマンドで削除します。

2.6 環境設定

2.6.1 ホスト・マシン（IBM PC/AT 互換機の場合）

RA78K0 は，32 ビット化されており，i386™ 以上の CPU を搭載した機種で動作します。

32 ビット化は DOS Extender を使用する形で実現されていますので，次の OS で動作するように設計されています。

- Windows98/Me の DOS プロンプト
- Windows2000/XP/NT4.0 のコマンド・プロンプト

なお，動作時には 7 M バイト以上のプロテクトメモリが必要です。

各 OS における環境変数の制限について次に示します。

2.6.2 環境変数

次の環境変数を設定してください。また，Windows インストーラでインストールした場合は，必要な環境変数が自動的に設定されます。

PATH	：アセンブラの実行形式を置いたディレクトリを指定します
TMP	：一時ファイルを作成するディレクトリを指定します （IBM PC/AT 互換機のみ有効）
INC78K0	：インクルード・ファイルを検索するディレクトリを指定します
LIB78K0	：ライブラリを使用する場合ライブラリを検索するディレクトリを指定します
LANG78K	：コメントに記述した漢字の漢字コードを指定します

【指定例】

- IBM PC/AT 互換機の場合

```
PATH = %PATH% ; C:\NECTools32\bin
set      TMP = C:\tmp
set      INC78K0 = C:\NECTools32\inc78k0
set      LIB78K0 = C:\NECTools32\lib78k0
set      LANG78K = SJIS
```

- HP9000 シリーズ 700，SPARCstation シリーズの場合

＜ csh を使用している場合の例＞

```
set      path = ( $path /ra78k0 )
setenv   INC78K0 /ra78k0
setenv   LIB78K0 /ra78k0
setenv   LANG78K EUC
```

＜ sh を使用している場合の例＞

```
PATH = $PATH:/ra78k0
INC78K0 = /ra78k0
LIB78K0 = /ra78k0
LANG78K = EUC
export    PATH INC78K0 LIB78K0 LANG78K
```

2.6.3 ソース・ファイル中の漢字コード

- ソース・ファイル中の特定の場所（コメントなど）には，漢字を使用できます。
- 漢字コードの種類は環境変数（LANG78K），漢字コード制御命令（KANJICODE），または漢字コード指定オプション（-ZE/-ZS/-ZN）で指定してください。

第 3 章　RA78K0 の実行手順

この章では，RA78K0 アセンブラ・パッケージを使用し，アセンブルからオブジェクト・コンバートまでを行う手順を説明します。

実行手順に従って実際にサンプル・プログラム“ k0main.asm ”,“ k0sub.asm ”をアセンブルからリンク，オブジェクト・コンバートまでを行います。

ここでは，コマンド行で実行する方法を説明します。

3.1　RA78K0 実行の前に

3.1.1　サンプル・プログラム

システム・ディスクに格納されているファイルのうち“ k0main.asm ”,“ k0sub.asm ”は，動作確認用サンプル・プログラムのファイルです。

これらのファイルは，これからのアセンブラの操作でソース・プログラム・ファイルとして，アセンブラへの入力になります。

k0main.asm　：メイン・モジュール

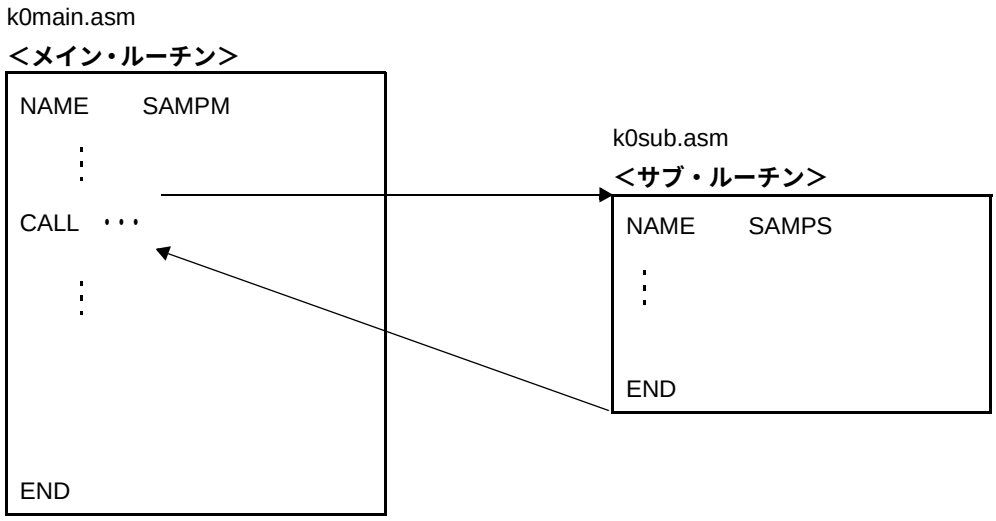
k0sub.asm　：サブ・モジュール

これらサンプル・プログラムは，16 進数のデータを ASCII コードに変換するもので，1 つのプログラムをメイン・ルーチンとサブルーチンの 2 つのモジュールに分けています。

メイン・ルーチンのモジュール名は SAMPM で，ソース・モジュール・ファイル（k0main.asm）に格納されています。

また，サブルーチンのモジュール名は SAMPS で，ソース・モジュール・ファイル（k0sub.asm）に格納されています。

図 3-1 サンプル・プログラムの構造



■ k0main.asm（メイン・ルーチン）

```
        NAME    SAMPM
; ************************************************
; *                                              *
; *       HEX -> ASCII Conversion Program        *
; *                                              *
; *       main-routine                           *
; *                                              *
; ************************************************

        PUBLIC  MAIN , START
        EXTRN   CONVAH
        EXTRN   _@STBEG

DATA    DSEG    saddr
HDTSA : DS      1
STASC : DS      2

CODE    CSEG    AT 0H
MAIN:   DW      START

        CSEG
START :
        ; chip initialize
        MOVW    SP , #_STBEG

        MOV     HDTSA , #1AH
        MOVW    HL , #HDTSA             ; set hex 2-code data in HL register

        CALL    !CONVAH                 ; convert ASCII <- HEX
                                        ; output BC-register <- ASCII code
        MOVW    DE , #STASC             ; set DE <- store ASCII code table
        MOV     A , B
        MOV     [ DE ] , A
        INCW    DE
        MOV     A , C
        MOV     [ DE ] , A

        BR      $$

        END
```

■ k0main.asm（サブルーチン）

```
        NAME    SAMPS
;*************************************************************************
;*                                                                       *
;*      HEX -> ASCII Conversion Program                                  *
;*                                                                       *
;*      sub-routine                                                      *
;*                                                                       *
;*      input condition : ( HL ) <- hex 2 code                           *
;*                                                                       *
;*      output condition   :BC-register <- ASCII 2 code                  *
;*                                                                       *
;*************************************************************************

        PUBLIC CONVAH

        CSEG
CONVAH : XOR   A , A
        ROL4    [ HL ]                  ; hex upper code load
        CALL    !SASC
        MOV     B , A                   ; store result

        MOV     A , A
        ROL4    [ HL ]                  ; hex lower code load
        CALL    !SASC
        MOV     C , A                   ; store result

        RET

;*************************************************************
;*      subroutine convert ASCII code                        *
;*      input Acc ( lower 4bits ) <- hex code                *
;*      output Acc          <-  ASCII code                   *
;*************************************************************

SASC :  CMP     A , #0AH                ; check hex code > 9
        BC      $SASC1
        ADD     A , #07H                ; bias ( +7 )
SASC1 : ADD     A , #30H                ; bias ( +30 )
        RET

        END
```

備考 1 このサンプル・プログラムは，RA78K0 の機能や操作を習得していただく目的で用意した参考プログラムです。したがって，アプリケーション・プログラムとして，そのまま利用することはできません。

備考 2 このサンプル・プログラムでは，レジスタ・バンク選択フラグ（RBS0，RBS1）の初期値設定を行っていません。したがって，次のようになります。

レジスタ・バンク 0（FEF8H-FEFFH）

3.1.2 サンプル・プログラムの構成

これ以後の説明で，操作例に使用するサンプル・プログラムの構成を次に示します。

k0main.asm **：メイン・モジュール**

k0sub.asm **：サブ・モジュール**

mylib.lib **：ライブラリ・ファイル（ここでは使用しません）**

sample.dr **：ディレクティブ・ファイル**

3.2 RA78K0 の実行手順

RA78K0 の動作には，システム・ディスクの中に格納されているバッチ・ファイル（ra.bat）を使用します。

ra.bat ではアセンブリ言語で記述された “ k0main.asm ”, “ k0sub.asm ” をソース・ファイルとして，アセンブラ，リンカ，オブジェクト・コンバータ，リスト・コンバータを順に実行し，エラーが出力されたらメッセージを出力してバッチ・ファイルを終了します。

このバッチ・ファイルは，ターゲットとなるデバイスの品種指定を入力するようになっています（デバイス・ファイルはオンライン・デリバリ・サービス（ODS）から別途入手してください）。

ここでは，ターゲット・デバイスを “ uPD780058 ” として説明します。

■ ra.bat（RA78K0 動作確認用バッチ・プログラム）

```
        echo off
        cls
        set         LEVEL = 0

        if " %1 " == "" goto ERR_BAT

        ra78k0 -C%1 k0main.asm
        if errorlevel 1 set LEVEL = 1
        ra78k0 -C%1 k0sub.asm
        if errorlevel 1 set LEVEL = 1
        if %LEVEL% == 1 echo Assemble error !!
        if %LEVEL% == 1 goto END

        cls
        lk78k0 k0main.rel k0sub.rel -s -orasample.lmf -prasample.map
        if errorlevel 1 echo Link error !!
        if errorlevel 1 goto END

        cls
        oc78k0 rasample
        if errorlevel 1 echo Object conversion error !!
        if errorlevel 1 goto END
        cls
        set LEVEL = 0
        lcnv78k0 -lrasample.lmf -rk0main.rel k0main.prn
        if errorlevel 1 set LEVEL = 1
        lcnv78k0 -lrasample.lmf -rk0sub.rel k0sub.prn
        if errorlevel 1 set LEVEL = 1
        if %LEVEL% == 1 echo List conversion error !!
        if %LEVEL% == 1 goto END

        cls
        echo No error.
        goto END

: ERR_BAT

        echo Usage : ra.bat chiptype

: END
        echo on
```

(1) **バッチ・ファイルを実行します。**

RA78K0 **動作確認用バッチ・プログラムを，品種を指定して実行します。**

```
C>ra.bat 0058
```

次のメッセージがディスプレイに出力されます。

```
78K/0 Series Assembler Vx.xx  [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD780058
 Device file : Vx.xx

Assembly complete ,     0 error ( s ) and     0 warning ( s ) found.

78K/0 Series Assembler Vx.xx  [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD780058
 Device file : Vx.xx

Assembly complete ,     0 error ( s ) and     0 warning ( s ) found.
```

画面をクリアする

```
78K/0 Series Linker Vx.xx  [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

 Target chip : uPD780058
 Device file : Vx.xx

Link complete ,     0 error ( s ) and     0 warning ( s ) found.
```

画面をクリアする

```
78K/0 Series Object Converter Vx.xx  [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

 Target chip : uPD780058
 Device file : Vx.xx

Object Conversion Complete ,      0 error ( s ) and     0 warning ( s ) found.
```

画面をクリアする

```
List Conversion Program for RA78K/0 Vx.xx  [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

Pass1 : start...
Pass2 : start...
Conversion complete.

List Conversion Program for RA78K/0 Vx.xx  [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

Pass1 : start...
Pass2 : start...
Conversion complete.
```

画面をクリアする

```
No error.
```

(2) **ドライブ** C **の内容をチェックします。**

次のファイルが出力されています。

k0main.rel	**：オブジェクト・モジュール・ファイル**
k0main.prn	**：アセンブル・リスト・ファイル**
k0sub.rel	**：オブジェクト・モジュール・ファイル**
k0sub.prn	**：アセンブル・リスト・ファイル**
rasample.lmf	**：ロード・モジュール・ファイル**
rasample.map	**：リンク・リスト・ファイル**
rasample.hex	**：** HEX **形式オブジェクト・モジュール・ファイル**
rasample.sym	**：シンボル・テーブル・ファイル**
k0main.p	**：アブソリュート・アセンブル・リスト・ファイル**
k0sub.p	**：アブソリュート・アセンブル・リスト・ファイル**

(3) RA78K0 の実行手順のまとめ

RA78K0 の実行手順をまとめると次のようになります。

図 3-2 RA78K0 の実行手順 1

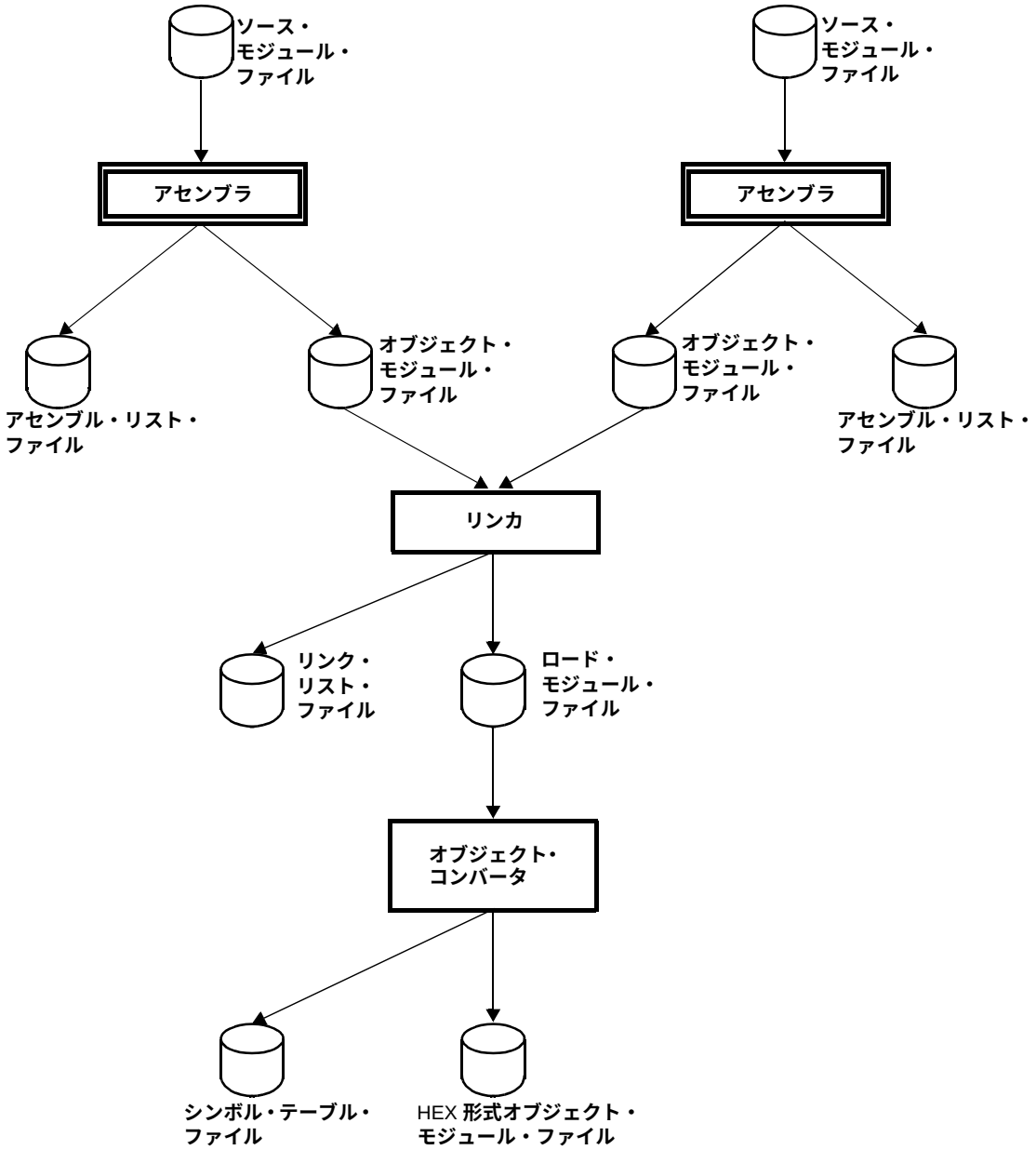
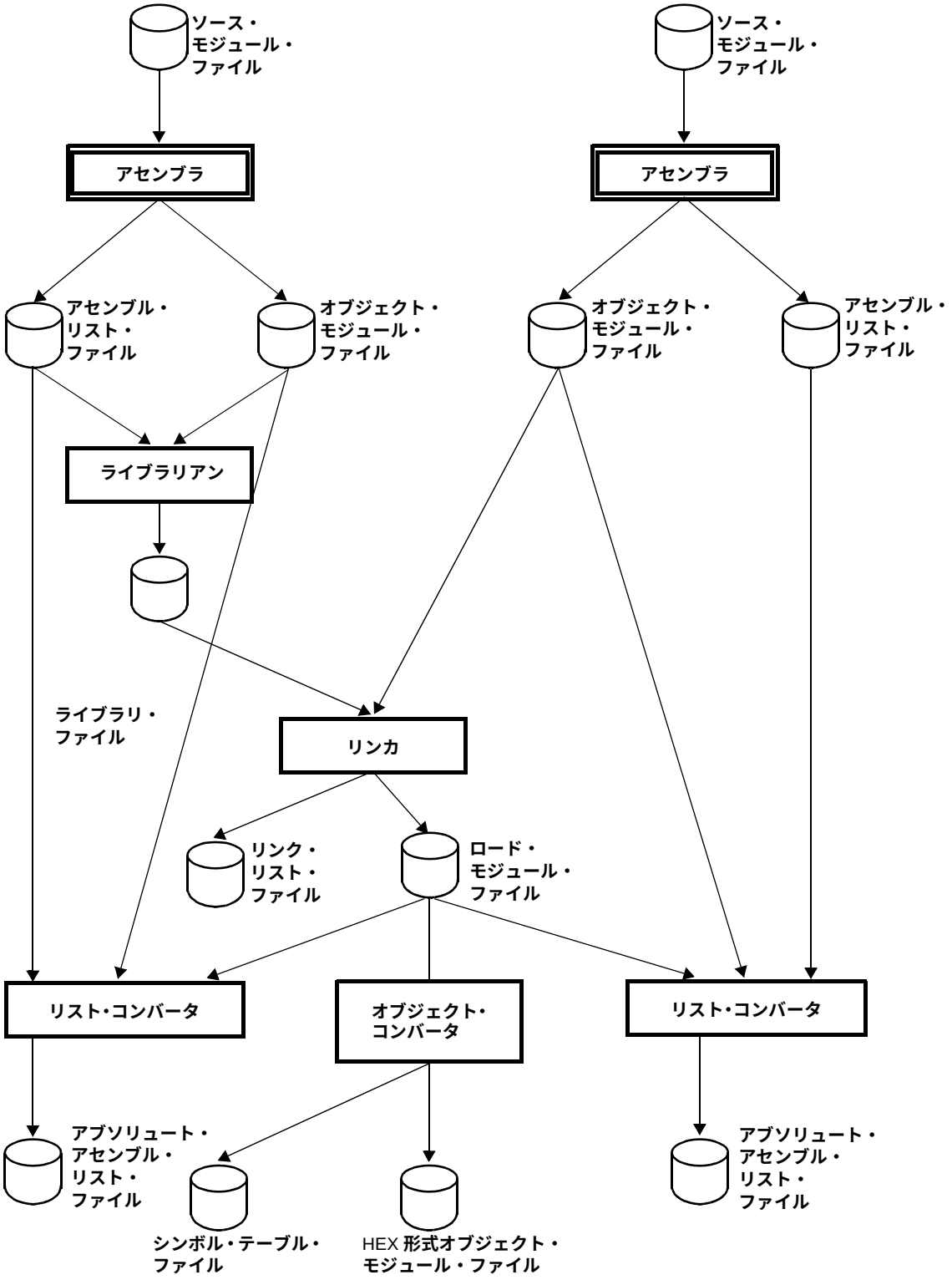


図 3-3 RA78K0 の実行手順 2



3.3 ST78K0 の実行手順

ST78K0 の動作確認には，システム・ディスクの中に格納されているバッチ・ファイル（st.bat）を使用します。

st.bat では構造化アセンブリ言語で記述された “test1.s”，“test2.s” をソース・ファイルとして，構造化アセンブラ・プリプロセッサ，アセンブラ，リンカ，オブジェクト・コンバータ，リスト・コンバータを順に実行し，エラーが出力されたらメッセージを出力してバッチ・ファイルを終了します。

このバッチ・ファイルは，ターゲットとなるデバイスの品種指定を入力するようになっています（デバイス・ファイルはオンライン・デリバリ・サービス（ODS）から別途入手してください）。

ここでは，ターゲット・デバイスを “ uPD780058 ” として説明します。

■ st.bat（ST78K0 動作確認用バッチ・プログラム）

```
        echo off
        cls
        set         LEVEL = 0

        if " %1 " == "" goto ERR_BAT

        st78k0 -C%1 test1.s
        ra78k0 test1.asm
        if errorlevel 1 set LEVEL = 1
        st78k0 -C%1 test2.s
        ra78k0 test2.asm
        if errorlevel 1 set LEVEL = 1
        if %LEVEL% == 1 echo Assemble error !!
        if %LEVEL% == 1 goto END

        cls
        lk78k0 test1.rel test2.rel -s -ostsample.lmf -pstsample.map
        if errorlevel 1 echo Link error !!
        if errorlevel 1 goto END

        cls
        oc78k0 stsample
        if errorlevel 1 echo Object conversion error !!
        if errorlevel 1 goto END

        cls
        set LEVEL = 0
        lcnv78k0 -lstsample.lmf -rtest1.rel test1.prn
        if errorlevel 1 set LEVEL = 1
        lcnv78k0 -lstsample.lmf -rtest2.rel test2.prn
        if errorlevel 1 set LEVEL = 1
        if %LEVEL% == 1 echo List conversion error !!
        if %LEVEL% == 1 goto END

        cls
        echo No error.
        goto END

: ERR_BAT

        echo Usage : st.bat chiptype

: END
        echo on
```

(1) **バッチ・ファイルを実行します。**

ST78K0 **動作確認用バッチ・プログラムを，品種を指定して実行します。**

```
C>st.bat 0058
```

次のメッセージがディスプレイに出力されます。

```
Structured assembler preprocessor for RA78K/0 Vx.xx  [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

start

 Target chip : uPD780058
 Device file : Vx.xx

Conversion complete , 0 error ( s ) found.

78K/0 Series Assembler Vx.xx  [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD780058
 Device file : Vx.xx

Assembly complete ,     0 error ( s ) and     0 warning ( s ) found.

Structured assembler preprocessor for RA78K/0 Vx.xx  [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

start

 Target chip : uPD780058
 Device file : Vx.xx

Conversion complete , 0 error ( s ) found.

78K/0 Series Assembler Vx.xx  [xx xxx xxxx]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD780058
 Device file : Vx.xx

Assembly complete ,     0 error ( s ) and     0 warning ( s ) found.
```

画面をクリアする

```
78K/0 Series Linker Vx.xx  [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

 Target chip : uPD780058
 Device file : Vx.xx

Link complete ,     0 error ( s ) and     0 warning ( s ) found.
```

画面をクリアする

```
78K/0 Series Object Converter Vx.xx  [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

 Target chip : uPD780058
 Device file : Vx.xx

Object Conversion Complete ,      0 error ( s ) and     0 warning ( s ) found.
```

画面をクリアする

```
List Conversion Program for RA78K/0 Vx.xx  [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

Pass1 : start..
Pass2 : start..
Conversion complete.

List Conversion Program for RA78K/0 Vx.xx  [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

Pass1 : start..
Pass2 : start..
Conversion complete.
```

画面をクリアする

```
No error.
```

(2) ドライブ C の内容をチェックします。

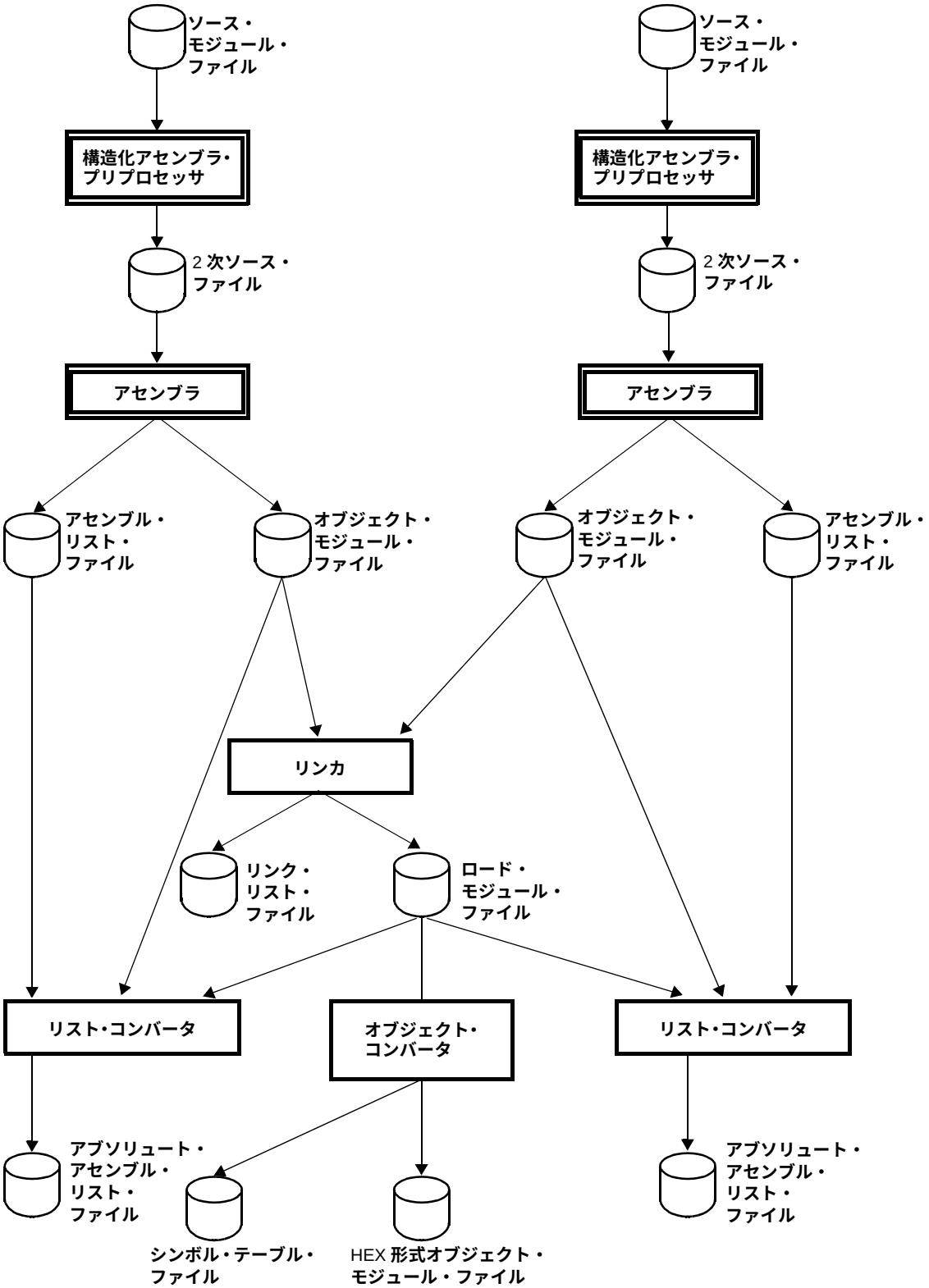
次のファイルが出力されています。

test1.asm	： 2 次ソース・ファイル
test1.rel	： オブジェクト・モジュール・ファイル
test1.prn	： アセンブル・リスト・ファイル
test2.asm	： 2 次ソース・ファイル
test2.rel	： オブジェクト・モジュール・ファイル
test2.prn	： アセンブル・リスト・ファイル
stsample.lmf	： ロード・モジュール・ファイル
stsample.map	： リンク・リスト・ファイル
stsample.hex	： HEX 形式オブジェクト・モジュール・ファイル
stsample.sym	： シンボル・テーブル・ファイル
test1.p	： アブソリュート・アセンブル・リスト・ファイル
test2.p	： アブソリュート・アセンブル・リスト・ファイル

(3) ST78K0 の実行手順のまとめ

ST78K0 の実行手順をまとめると次のようになります。

図 3-4 ST78K0 の実行手順



3.4 コマンド行（DOS プロンプト，UNIX）でのアセンブルからリンク，オブジェクト・コンバートの実行手順

コマンド行でアセンブルからオブジェクト・コンバートを行う方法を説明します。

(1) サンプル・プログラム（k0main.sam）をアセンブルします。

コマンド行には次のように入力します。

```
C>ra78k0 -c054 k0main.asm
```

次のメッセージがディスプレイに出力されます。

```
78K/0 Series Assembler Vx.xx [ xx xxx xx ]
    Copyright ( C ) NEC Electronics Corporation xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD78054
 Device file : Vx.xx

Assembly complete ,       0 error ( s ) and       0 warning ( s ) found.
```

(2) ドライブ C の内容をチェックします。

アセンブラは，オブジェクト・モジュール・ファイル（k0main.rel）とアセンブル・リスト・ファイル（k0main.prn）を出力します。

なお，アセンブル時に，-E オプションを指定することにより，アセンブラはエラー・リスト・ファイル（アセンブル・エラーになった行とそのエラー・メッセージの内容のリスト）を出力します。

(3) サンプル・プログラム（k0sub.asm）をアセンブルします。

コマンド行には次のように入力します。

```
C>ra78k0 -c054 k0sub.asm
```

次のメッセージがディスプレイに出力されます。

```
78K/0 Series Assembler  Vx.xx [ xx xxx xx ]
    Copyright (C) NEC Electronics Corporation xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD78054
 Device file : Vx.xx

Assembly complete ,       0 error ( s ) and       0 warning ( s ) found.
```

(4) ドライブ C の内容をチェックします。

アセンブラは，オブジェクト・モジュール・ファイル（k0sub.rel）とアセンブル・リスト・ファイル（k0sub.prn）を出力します。

なお，アセンブル時に，-E オプションを指定することにより，アセンブラはエラー・リスト・ファイルを出力します。

(5) ディレクティブ・ファイルを作成します。

ディレクティブ・ファイルは，リンカに対してセグメントの配置を指定します。

したがって，セグメントを配置するためにディフォールトの ROM/RAM 領域を拡張したり，新たにメモリ領域を定義する必要のある場合には，ディレクティブ・ファイルを作成します。

また，ソース・モジュール中でアブソリュート・セグメントとして定義していないセグメントをメモリ上の特定番地に配置しようとする場合にもディレクティブ・ファイルを作成する必要があります。

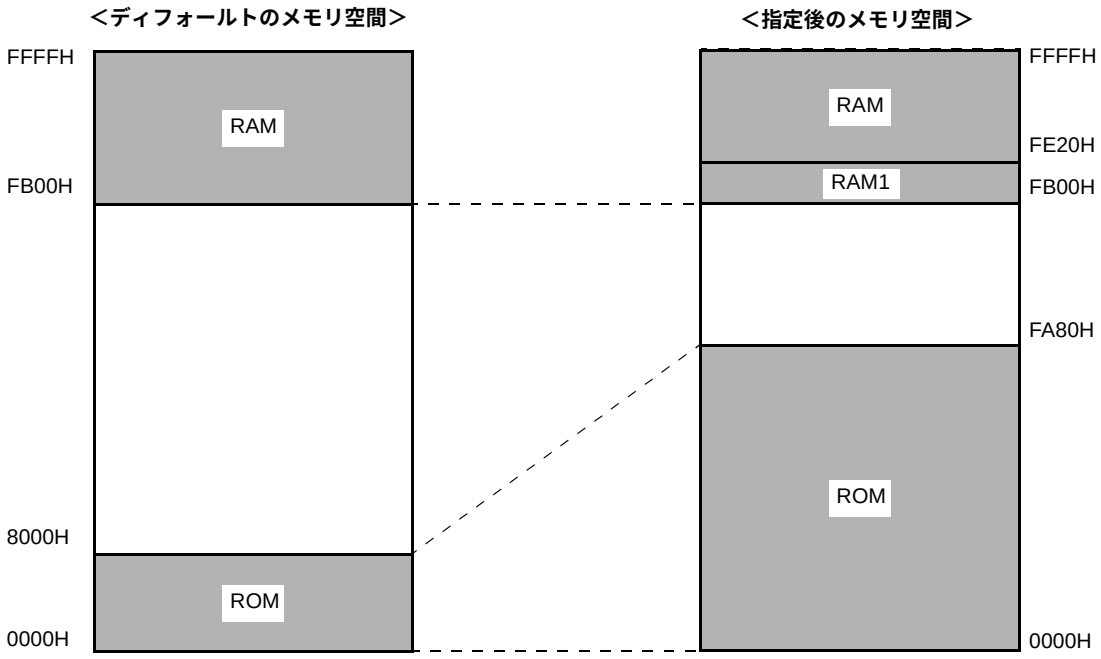
ディレクティブ・ファイルは，リンク時に，-D オプションを用いてリンカに入力します。

例　ROM（0H-7FFFH）領域を ROM（0H-FA7FH）領域に拡張し，RAM 領域を RAM（FE20H-FFFFH）領域と RAM1（FB00H-FE1FH）領域に拡張する場合

ディレクティブ・ファイルには，次のように記述します。

```
MEMORY ROM : ( 0H , 0FA80H )
MEMORY RAM1 : ( 0FB00H , 320H )
MEMORY RAM : ( 0FE20H , 1E0H )
```

図 3-5 リンク・ディレクティブ



(6) アセンブルの結果，出力されたオブジェクト・モジュール・ファイル“ k0main.rel ”と“ k0sub.rel ”をリンクします。

また，ディレクティブ・ファイルとして，k0.dr を入力します。

コマンド行には次のように入力します。

```
C>lk78k0 k0main.rel k0sub.rel –dk0.dr 注1 –ok0.lmf –pk0.map -S 注2
```

注 1　ディレクティブ・ファイルを指定しない場合は不要です。

注 2　スタック解決用シンボル（_@STBEG）生成オプションです。

次のメッセージがディスプレイに出力されます。

```
78K/0 Series Linker Vx.xx [ xx xxx xx ]
   Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

 Target chip : uPD78054
 Device file : Vx.xx

Link complete ,       0 error ( s ) and       0 warning ( s ) found.
```

(7) ドライブ C の内容をチェックします。

リンカは，ロード・モジュール・ファイル（k0.lmf）とリンク・リスト・ファイル（k0.map）を出力しました。

なお，リンク時に -E オプションを指定することにより，リンカはエラー・リスト・ファイルを出力します。

(8) リンクの結果出力されたロード・モジュール・ファイル（k0.lmf）を HEX 形式ファイルに変換します。

コマンド行には次のように入力します。

```
C>oc78k0 k0.lmf –r –u0FFH
```

次のメッセージがディスプレイに出力されます。

```
78K/0 Series Object Converter Vx.xx [ xx xxx xx ]
   Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

 Target chip : uPD78054
 Device file : Vx.xx

Object Conversion Complete ,       0 error ( s ) and       0 warning ( s ) found.
```

(9) ドライブ C の内容をチェックします。

オブジェクト・コンバータは，HEX 形式オブジェクト・モジュール・ファイル（k0.hex）とシンボル・テーブル・ファイル（k0.sym）を出力しました。

(10) ライブラリ・ファイルを作成します。

アセンブラの出力したオブジェクト・モジュール・ファイル（k0sub.rel）をライブラリ・ファイルとして登録します。

コマンド行には次のように入力します。

```
C>lb78k0 < k0.job
```

次のメッセージがディスプレイに出力されます。

```
78K/0 Series Librarian Vx.xx [ xx xxx xx ]
    Copyright ( C ) NEC Electronics Corporation xxxx , xxxx
*create k0.lib                                          ; “ k0.job ” の内容
*add k0.lib k0sub.rel                                   ; “ k0.job ” の内容
*exit
```

(11) ドライブ C の内容をチェックします。

ライブラリアンは，ライブラリ・ファイル（k0.lib）を出力しました。

(12) アブソリュート・アセンブル・リストを作成します。

k0main.asm のアブソリュート・アセンブル・リストを作成するため “ k0main.rel ”, “ k0main.asm ”，および “ k0.lmf ” をリスト・コンバータに入力します。

コマンド行には次のように入力します。

```
C>lcnv78k0 k0main -lk0.lmf
```

次のメッセージがディスプレイ出力されます。

```
List Conversion Program for RA78K/0 Vx.xx [ xx xxx xx ]
    Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

Pass1 : start...
Pass2 : start...
Conversion complete.
```

(13) ドライブ C の内容をチェックします。

リスト・コンバータは，アブソリュート・アセンブル・リスト・ファイル（k0main.p）を出力しました。

3.5 パラメータ・ファイルの使用

アセンブラ，リンカ起動時に複数のオプションを入力する際に，コマンド行で起動に必要な情報を指定しきれない場合や，同じ指定を何回も繰り返すことがあります。このようなときにパラメータ・ファイルを使用します。

パラメータ・ファイルを使用する場合は，パラメータ・ファイル指定オプションをコマンド行の中で指定してください。

注意　パラメータ・ファイルは，PM plus **でのオプション設定では指定できません。**

パラメータ・ファイルによる起動方法は，次のようになります。

```
>[ パス名 ] ra78k0 △ -F パラメータ・ファイル名
>[ パス名 ] lk78k0 △ -F パラメータ・ファイル名
>[ パス名 ] oc78k0 △ -F パラメータ・ファイル名
```

次に使用例を示します。

```
C>ra78k0 -Fpara.pra
C>lk78k0 -Fpara.plk
C>oc78k0 -Fpara.poc
```

パラメータ・ファイルは，エディタで作成し，コマンド行で指定すべきすべてのオプション，出力ファイル名を記述できます。

パラメータ・ファイルをエディタで作成した例を次に示します。

＜ para1.pra の内容＞

```
-c054 k0main.asm -e
```

＜ para.plk の内容＞

```
k0main.rel k0sub.rel -bmylib.lib -osample.lmf -S
```

＜ para.poc の内容＞

```
sample.lmf -u0FFH -osample.hex -r
```

第 4 章　構造化アセンブラ・プリプロセッサ

この章では，構造化アセンブラ・プリプロセッサは，78K0 シリーズの構造化アセンブリ言語で記述されたソース・モジュール・ファイルを入力し，それをアセンブリ言語に変換して 2 次ソース・モジュール・ファイルとして出力します。

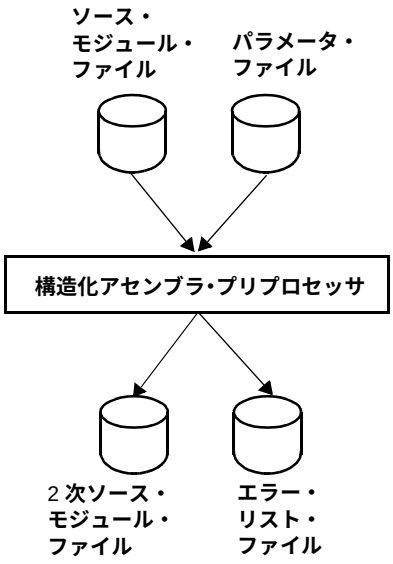
4.1　構造化アセンブラ・プリプロセッサの入出力ファイル

構造化アセンブラ・プリプロセッサの入出力ファイルを以下に示します。

表 4-1 構造化アセンブラ・プリプロセッサの入出力ファイル

種類	ファイル名	説明	ディフォールト・ファイル・タイプ
入力ファイル	ソース・モジュール・ファイル	- 構造化アセンブリ言語で記述されたソース・モジュール・ファイルです。 - ユーザ作成ファイルです。	なし
	パラメータ・ファイル	- 構造化アセンブラ・プリプロセッサのオプションをファイルから指定するための，オプションを内容とするファイルです。 - ユーザ作成ファイルです。	.pst
出力ファイル	2 次ソース・モジュール・ファイル	- アセンブリ言語で記述されたソース・モジュール・ファイルです。	.asm
	エラー・リスト・ファイル	- 構造化アセンブラ・プリプロセッサのエラー情報を持つファイルです。	.est

図 4-1 構造化アセンブラ・プリプロセッサの入出力ファイル



4.2 構造化アセンブラ・プリプロセッサの機能

(1) 構造化アセンブラ・プリプロセッサは，ソース・モジュール・ファイルを読み込み，アセンブラの入力ソース・ファイルとなる，アセンブラ・ソース・ファイルを出力します。

(2) 構造化アセンブラ・プリプロセッサは，ファイルやシステムに関するエラーがある場合は，アボート・エラーを出力し，ソース・モジュール中に記述エラーを発見した場合は，フェータル・エラーやワーニング・エラーを出力します。
アボート・エラー，フェータル・エラーの場合は，通常 2 次ソース・ファイルは出力されません。ただし，-J オプションが指定された場合には，フェータル・エラーがある場合でも 2 次ソース・ファイルを出力します。

(3) 構造化アセンブラ・プリプロセッサは，起動時に指定されたオプションに従って処理を行います。構造化アセンブラ・プリプロセッサのオプションの詳細については，「4.4 構造化アセンブラ・オプション」を参照してください。

(4) 構造化アセンブラ・プリプロセッサは，その処理を正常に終了すると終了メッセージを出力し，制御を OS に戻します。

4.3 構造化アセンブラ・プリプロセッサの起動方法

4.3.1 構造化アセンブラ・プリプロセッサの起動

構造化アセンブラ・プリプロセッサの起動には，次の 2 つの方法があります。

(1) コマンド行での起動

次のコマンドを入力して構造化アセンブラ・プリプロセッサを起動します。

```
X>[ パス名 ] st78k0 [ △オプション ]  …ソース・モジュール・ファイル名  [ △オプション ] …
↑      ↑       ↑          ↑                    ↑                          ↑
(a)    (b)     (c)        (d)                  (e)                        (d)
```

(a) カレント・ドライブ名

(b) カレント・ディレクトリ名

(c) 構造化アセンブラ・プリプロセッサのコマンド・ファイル名

(d) 構造化アセンブラ・プリプロセッサに対して動作の詳細を指示します。

複数のオプションを指定する場合は，それぞれのオプション間を空白で区切ってください。アセンブラ・オプションの詳細については，「4.4 構造化アセンブラ・オプション」を参照してください。

空白を含むパスを設定する場合には，ダブルクオーテーション（“ ”）で囲んでください。

(e) 構造化アセンブルするソース・モジュール・ファイル名

空白を含むパスのファイル名を指定する場合には,ダブルクオーテーション(“ ”)で囲んでください。

例　C>st78k0 –c054 test1.s -e

(2) パラメータ・ファイルによる起動

構造化アセンブラ・プリプロセッサを起動するときに，何度も同じ指定をするのは面倒です。このような場合にパラメータ・ファイルを使用して構造化アセンブラ・プリプロセッサを起動すると便利です。

パラメータ・ファイルを使用する場合には，コマンド行にパラメータ・ファイル指定オプション（-F）を指定します。

パラメータ・ファイルによる起動方法は，次のようになります。

```
X> [ パス名 ] st78k0 [ △ソース・モジュール・ファイル名 ] △ -F パラメータ・ファイル名
↑      ↑                    ↑                          ↑
(a)    (b)                  (c)                        (d)
```

(a) カレント・ドライブ名

(b) カレント・ディレクトリ名

(c) パラメータ・ファイル指定オプション

(d) パラメータ・ファイル名

パラメータ・ファイル内での記述規則を次に示します。

```
[ [ [ △ ] オプション [ △オプション ] … [ △ ] △ ] ] …
```

コマンド行でソース・モジュール・ファイル名を省略した場合は，パラメータ・ファイル内でソース・モジュール・ファイル名を 1 つだけ指定できます。

ソース・モジュール・ファイル名は，オプションのあとに記述することもできます。

パラメータ・ファイルには，コマンド行で指定するすべてのオプション，出力ファイル名を記述します。

例　パラメータ・ファイル（test1.pst）をエディタで作成します。

＜ test1.pst の内容＞

```
; Parameterfile
test1.s -osample.asm
-esample.est –c054
```

パラメータ・ファイル（test1.pst）を使用して構造化アセンブラ・プリプロセッサを起動します。

```
C>st78k0 -ftest1.pst
```

4.3.2　実行開始メッセージ，終了メッセージ

(1) 実行開始メッセージ

構造化アセンブラ・プリプロセッサが起動すると，次の実行開始メッセージが表示されます。

```
Structured assembler preprocessor for RA78K/0 Vx.xx [ xx xxx xx ]
    Copyright ( C ) NEC Electronics Corporation xxxx , xxxx
```

(2) 処理表示メッセージ

構造化アセンブラ・プリプロセッサの 100 行処理ごとに“ . ”が表示されます。

```
start……
```

(3) 実行終了メッセージ

エラーが検出されなかった場合，次のメッセージを表示して制御を OS に戻します。

```
   Target chip : uPD78xxxx
   Device file : Vx.xx

Conversion complete , 0 error ( s ) found.
```

エラーが検出された場合，エラーの数を表示して制御を OS に戻します。

```
TEST1.S ( 8 ) :   RA78K0 error E1209 : Syntax error

   Target chip : uPD78xxxx
   Device file : Vx.xx

Conversion complete , 1 error ( s ) found.
```

構造化アセンブル中に構造化アセンブラ・プリプロセッサ処理継続が不可能な致命的エラーが検出された場合，構造化アセンブラ・プリプロセッサはメッセージを表示して構造化アセンブルを中止し，制御を OS に戻します。

例 1　存在しないソース・モジュール・ファイルを指定した場合

```
C>st78k0 sample.s

Structured assembler preprocessor for RA78K/0 Vx.xx [ xx xxx xx ]
      Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

RA78K0 error F1006 : File not found ' SAMPLE.S '

Program aborted.
```

この例では，存在しないソース・モジュール・ファイルを指定したためにエラーとなり，アセンブルを中止しました。

例 2　存在しないオプションを指定した場合

```
C>st78k0 test1.s -z

Structured assembler preprocessor for RA78K/0 Vx.xx [ xx xxx xx ]
      Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

RA78K0 error F1012 : Missing parameter ' -z '
Please enter ' ST78K0 -- ' , if you want help messages.

Program aborted.
```

この例では，存在しないオプションを指定したためにエラーとなり，アセンブルを中止しました。

構造化アセンブラ・プリプロセッサがエラー・メッセージを出力してアセンブルを中止した場合，そのエラー・メッセージの原因を「第 12 章 エラー・メッセージ」で調べて対処してください。

4.4 構造化アセンブラ・オプション

4.4.1 構造化アセンブラ・オプションの種類

構造化アセンブラ・プリプロセッサのオプションは，構造化アセンブラ・プリプロセッサの動作に細かい指示を与えるものです。

構造化アセンブラ・プリプロセッサのオプションは，次の 13 種類のオプションに分類できます。

表 4-2 構造化アセンブラ・オプション

分類	オプション	説明
デバイス種別指定	-C	対象とするデバイス種別を指定します。
ワード・シンボル文字指定	-SC	ワード・シンボル名の最後の文字を指定します。
シンボル定義指定	-D	#IFDEF 疑似命令などに与えるシンボルを指定します。
タブ数指定	-WT	変換した命令を出力する位置を指定します。
インクルード・ファイル読み込みパス指定	-I	インクルード・ファイルのドライブ，ディレクトリを指定します。
2 次ソース・ファイル指定	-O	2 次ソース・ファイル名を指定します。
エラー・リスト・ファイル指定	-E	エラー・リスト・ファイル名を指定します。
パラメータ・ファイル指定	-F	パラメータ・ファイル名を指定します。
ディバグ情報出力指定	-GS	構造化アセンブラ・ソース・レベルのディバグ情報の出力を指定します。
	-NGS	
2 次ソース・ファイル強制出力指定	-J	2 次ソース・ファイルの強制的出力を指定します。
漢字コード指定	-ZS	コメント文に記述される漢字コードの種別を指定します。
	-ZE	
	-ZN	
デバイス・ファイル・サーチ・パス指定	-Y	デバイス・ファイルをサーチするパスを指定します。
ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

4.4.2 構造化アセンブラ・オプションの説明

次ページ以降に，各構造化アセンブラ・オプションの詳細について説明します。

(1) デバイス種別指定

デバイス種別指定 （-C）

【記述形式】

-C デバイス種別

- **省略時解釈**

  省略不可

【機能】

- **構造化アセンブラ・プリプロセッサの対象となる対象デバイスを指定します。**

【説明】

- -C **オプションは必ず指定してください。構造化アセンブラ・プリプロセッサは指定された対象デバイスに対してプリプロセスを行い，アセンブラのソース・コードを生成します。**
  なお，-C **オプションを省略した場合はエラーとなります。**
- -C **オプションとプロセッサ品種指定制御命令で異なる品種が指定された場合は，ワーニングとなります。この際，構造化アセンブラ・プリプロセッサは，オプションで指定した品種を優先します。**
- -C **オプションで指定した品種をプロセッサ品種指定制御命令として，**2 **次ソース・ファイルに出力します。ただし，プロセッサ品種指定制御命令と同名の品種を指定した場合は，出力しません。**

【使用例】

- uPD78054 **をターゲットとして指定します。**

```
C>st78k0 test.s –c054
```

【注意】

- -C **オプションは省略不可能です。ただし，ソース・ファイルの先頭でプロセッサ品種指定制御命令（**$PROCESSOR**）を記述すれば，コマンド行での指定を省略できます。**
  デバイス種別については，各デバイスの「デバイス・ファイル　使用上の留意点」を参照してください。

(2) ワード・シンボル文字指定

ワード・シンボル文字指定 （-SC）

【記述形式】

-SC **文字**

- **省略時解釈**

  -SCP

【機能】

- **シンボル名でバイト／ワードを区別する必要がある場合に，判定の対象となるシンボルの最終文字を指定します。**

【説明】

- **構造化アセンブラ・プリプロセッサは，扱うデータがバイトかワードかによって異なる命令を生成します。代入であれば，バイトならば** MOV **命令，ワードならば** MOVW **命令を生成します。**
  ワード・シンボルの予約語であればワードの命令を生成します。
- **予約語でないシンボルが指定された場合は，そのシンボルの最後の文字によって，ワード・シンボルかバイト・シンボルかを判定して命令を生成します。**
- -SC **オプションを指定しないと，“** P **”か“** p **”で終わるシンボルがワード・シンボルと判断されます。**
- **判定文字は英字相当文字のみです。なお，英字の場合は大文字，小文字の区別はありません。**
- **重複指定した場合は，あとで指定した方が有効となります。**

【使用例】

- **“ @ ”で終わるシンボルがワード・シンボルと指定します。**

```
C>st78k0 test.s -sc@
```

〈 test.s 〉

```
A = #3
AX = #3
SYM = #3
SYM@ = #3
```

〈 test.asm 〉

```
MOV     A , #3      ; A = #3
MOVW    AX , #3     ; AX = #3
MOV     SYM , #3    ; SYM = #3
MOVW    SYM@ , #3   ; SYM@ = #3
```

(3) シンボル定義指定

シンボル定義指定 （-D）

【記述形式】

-D シンボル名 [= 数値] [, シンボル名 [= 数値] …]

- 省略時解釈

  なし

【機能】

- シンボルの定義を行います。

【説明】

- シンボルに与える数値は 2 進数，8 進数，10 進数，16 進数とします。
  数値指定が省略された場合，値は 1 となります。
- 本オプションは，シンボルを #define 疑似命令で定義したことと同一です。
- コマンド行ではカンマで区切って 30 個まで定義できます。
- シンボル名は，31 文字まで記述できます。
- 通常は，#ifdef 疑似命令と組み合わせて使用します。
- 重複して指定した場合は，あとで指定した方が有効となります。
- #define 疑似命令と重複した場合，ワーニング・メッセージを出力し，#define 疑似命令の定義を有効とします。
- 英字の場合には，英大文字，英小文字どちらを指定しても良く，同一の意味とします。

【使用例】

- “ TRUE ” というシンボルに 1 を定義します。

```
C>st78k0 test.s -dTRUE = 1
```

(4) タブ数指定

タブ数指定 （-WT）

【記述形式】

-WT **数値** 1 -WT [**数値** 1] , **数値** 2 -WT [**数値** 1] , [**数値** 2] , **数値** 3

- **省略時解釈**

  -WT2 , 3 , 4

【機能】

- **変換されたアセンブリ言語のタブの数を指定します。**

【説明】

- -WT **オプションにより，アセンブラのソースの命令出力位置を自由に変更可能となり，プログラムの可読性が向上します。**
- **数値** 1 **は，命令を出力するまでのタブの数を指定します。**
  数値 2 **は，命令のオペランドを出力するまでのタブの数を指定します。**
  数値 3 **は，命令のコメントを出力するまでのタブの数を指定します。**
- **数値は** 10 **進数で次の範囲内に指定してください。**
  数値 1：0 ～ 97
  数値 2：1 ～ 98
  数値 3：2 ～ 99
  数値 1 ＜ **数値** 2 ＜ **数値** 3
- **重複して指定した場合は，あとで指定した方が有効となります。**

【使用例】

- **数値** 1 **に** 3**，数値** 2 **に** 4**，数値** 3 **に** 5 **を指定します。**

```
C>st78k0 test.s -wt3 , 4 , 5
```

(5) インクルード・ファイル読み込みパス指定

インクルード・ファイル読み込みパス指定 （-I）

【記述形式】

-I パス名 [, パス名] …（複数指定可能）

- 省略時解釈

  インクルード・ファイルを次の順序で検索します。

  (i) ソース・ファイルのあるパス

  (ii) 環境変数 (INC78K0) により指定されたパス

【機能】

- 構造化アセンブラ・プリプロセッサの入力となるインクルード・ファイルのパス名を指定します。

【説明】

- “ , ”で区切ることにより，一度に複数のパス名を指定できます。
- “ , ”の前後に空白を入れることはできません。
- インクルード・ファイルの検索順序は，次の検索順になります。

  (i) -I オプションに続いてパス名が複数指定された場合は，指定された順番でインクルード・ファイルを検索します。

  (ii) -I オプションが複数指定された場合は，後者の指定を優先する順番でインクルード・ファイルを検索します。

  (iii) -I オプションで指定したパスの検索後に，省略時解釈と同じ順序でインクルード・ファイルを検索します。

- -I に続けてパス名以外のものを指定した場合やパス名を省略した場合は，アボート・エラーとなります。
- -I が 65 個以上指定された場合には，アボート・エラーとなります。

【使用例】

- インクルード・ファイルをディレクトリ C：\include から読み込みます。

```
C>st78k0 test.s –ic:\include
```

(6) 2 次ソース・ファイル指定

2 次ソース・ファイル指定 （-O）

【記述形式】

-O [[[ドライブ番号 :] ディレクトリ] ファイル名]

- 省略時解釈

  -O 入力ファイル名 .asm

【機能】

- 変換後の 2 次ソース・ファイルの出力先と，ファイル名を指定します。

【説明】

- 変換後の 2 次ソース・ファイルの出力ドライブ番号，ディレクトリ，ファイル名を指定します。
- -O オプションを省略した場合には，出力ファイルは入力ファイルのファイル・タイプを “ ASM ” に置き換えたものがカレント・ディレクトリに作成されます。
- ファイル名として “ NUL ”, “ AUX ” が指定できます。
- フェータル・エラー終了時には，2 次ソース・ファイルを出力しません。
- 重複して指定した場合は，あとで指定した方が有効となります。

【使用例】

- 2 次ソース・ファイルを “ sample.asm ” と指定します。

```
C>st78k0 test.s -osample.asm
```

(7) エラー・リスト・ファイル指定

エラー・リスト・ファイル指定 （-E）

【記述形式】

-E [[**ドライブ番号** :] [**ディレクトリ**] **ファイル名**]

- **省略時解釈**

  -E **入力ファイル名** .est

【機能】

- **エラー・リスト・ファイルの出力先と，ファイル名を指定します。**

【説明】

- **エラー・リスト・ファイルを出力するドライブ番号，ディレクトリ，ファイル名を指定します。**
- -E **オプションを省略した場合には，エラー・リスト・ファイルは入力ファイルのファイル・タイプを“** .est **”に置き換えたものがカレント・ディレクトリに作成されます。**
- **ファイル名として，“** NUL **”，“** AUX **”が指定できます。**
- **重複して指定した場合は，あとで指定した方が有効となります。**

【使用例】

- **エラー・リスト・ファイルを “** sample.est **” と指定します。**

```
C>st78k0 test.s -esample.est
```

(8) パラメータ・ファイル指定

パラメータ・ファイル指定 （-F）

【記述形式】

-F [[ドライブ番号 :] ディレクトリ] ファイル名

- 省略時解釈

  入力ファイルなし

【機能】

- パラメータ・ファイルのファイル名を指定します。

【説明】

- パラメータ・ファイルの入力ドライブ番号，ディレクトリ，ファイル名を指定します。
- ファイル名は省略できません。ファイル・タイプを省略すると，“ .pst ” と解釈されます。
- -D オプションで，シンボルをコマンド行で数多く定義する際に有効です。
- 重複して指定された場合はエラーとなります。
- パラメータ・ファイルのネストは禁止します。指定された場合は，エラーとなります。
- “ ; ”,または‘ # ’以降に記述された文字は改行文字（LF），またはEOFの前まですべてコメントと解釈します。

【使用例】

- パラメータ・ファイル “ sample.pst ” を指定します。

```
C>st78k0 -fsample.pst
```

(9) ディバグ情報出力指定

ディバグ情報出力指定 （-GS/-NGS）

【記述形式】

```
-GS
-NGS
```

- **省略時解釈**

  -GS

【機能】

- **構造化アセンブラ・ソース・レベルのディバグ情報の出力を指定します。**

【説明】

- -GS **オプションは，**2 **次ソース・ファイルにディバグ情報の出力を指定します。**
- -NGS **オプションは，**-GS **オプションを無効にします。**
- -GS **オプションは，入力ソース・ファイル中にコンパイラのディバグ情報があった場合，先頭の“**$**”を“**;**”に置き換えます。**
- -GS **と** -NGS **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**

【注意】

- **構造化アセンブラ・ソース・レベルでディバグをする場合は，構造化アセンブラ・プリプロセッサでディバグ情報出力指定オプション（**-GS**）を指定してください。**2 **次ソース・ファイルをアセンブルするときには，アセンブラ・オプションのディバグ情報出力指定オプション（**-G/-GA**）を指定しないでください。必要なオプションは，構造化アセンブラ・プリプロセッサが** 2 **次ソース・ファイル中に制御命令として出力します。**

【使用例】

- 2 **次ソース・ファイルにディバグ情報の出力を指定します。**

```
C>st78k0 test.s -gs
```

(10) 2 次ソース・ファイル強制出力指定

2 次ソース・ファイル強制出力指定 （-J）

【記述形式】

```
-J
```

- 省略時解釈

  フェータル・エラー終了時には，2 次ソース・ファイルは出力されない。

【機能】

- フェータル・エラー終了時に，2 次ソース・ファイルを強制的に出力させます。

【説明】

- フェータル・エラー終了時に，2 次ソース・ファイルを強制的に出力させます。
- フェータル・エラー行は，2 次ソース・ファイルに入力ソース・ファイルのイメージをそのまま出力します。

【使用例】

- 2 次ソース・ファイルの強制出力を指定します。

```
C>st78k0 test.s -j
```

(11) 漢字コード指定

漢字コード指定 （-ZS/-ZE/-ZN）

【記述形式】

```
-ZS
-ZE
-ZN
```

- **省略時解釈**

  OS **により次のように解釈します。**

  Windows , HP-UX ：-ZS

  SunOS , Solaris ：-ZE

【機能】

- **コメント文に記述される漢字コードの種別を指定します。**

【説明】

- **漢字コードを次のように指定します。**

  -ZS：**シフト** JIS **コードとして解釈します。**

  -ZE：EUC **コードとして解釈します。**

  -ZN：**漢字として解釈しません。**

- **本オプションは，次のように漢字コード指定制御命令に対応しています。**

  -ZS：$KANJICODE SJIS

  -ZE：$KANJICODE EUC

  -ZN：$KANJICODE NONE

- **漢字コードの指定の優先順位は次のようになります。**

  (i) -ZS/-ZE/-ZN **オプションの指定**

  (ii) **漢字コード指定制御命令（**$KANJICODE**）の指定**

  (iii) **環境変数** LANG78K **の指定**

  (iv) **各** OS **のディフォールトの指定**

【使用例】

- **漢字をシフト** JIS **コードとして解釈するように指定します。**

```
C>st78k0 test.s -zs
```

(12) デバイス・ファイル・サーチ・パス指定

デバイス・ファイル・サーチ・パス指定 （-Y）

【記述形式】

-Y [ドライブ番号] ディレクトリ

- 省略時解釈

  デバイス・ファイルを次の順序で検索します。

  (i) < ..\dev >（st78k0.exe を起動したパスに対して）

  (ii) st78k0.exe を起動したパス

  (iii) カレント・パス

  (iv) 環境変数 PATH で指定されたパス

【機能】

- デバイス・ファイルをサーチするパスを指定します。

【説明】

- デバイス・ファイルを指定されたパスから読み込みます。
- パス名以外が指定された場合にはエラーとなります。
- ディレクトリの最後尾にディレクトリ指定記号が記述されていなくても，記述されているものと解釈します。
- デバイス・ファイルの検索は次の順序で行います。

  (i) -Y オプションで指定されたパス

  (ii) 〈 ..\dev 〉（st78k0.exe を起動したパスに対して）

  (iii) st78k0.exe を起動したパス

  (iv) カレント・パス

  (v) 環境変数 PATH で指定されたパス

【使用例】

- デバイス・ファイルをディレクトリ c:\NECTools32\dev から読み込むように指定します。

```
C>st78k0 test.s –yc:\NECTools32\dev
```

(13) ヘルプ指定

ヘルプ指定 (--)

【記述形式】

```
--
```

- **省略時解釈**

  表示しない

【機能】

- **-- オプションは，ヘルプ・メッセージをディスプレイに出力します。**

【用途】

- **ヘルプ・メッセージは，構造化アセンブラ・オプションとその説明の一覧です。構造化アセンブラ・プリプロセッサを実行するときに参照してください。**

【説明】

- **-- オプションを指定すると，他の構造化アセンブラ・オプションはすべて無効となります。**
- **ヘルプ・メッセージの続きをお読みになる場合は，リターン・キーを入力してください。表示を途中で終了する場合は，リターン・キー以外の文字を入力したあとで，リターン・キーを入力してください。**

注意 このオプションは，PM plus **上では指定できません。**

PM plus **上でヘルプ・メッセージを参照する場合は，〈構造化アセンブラ・プリプロセッサオプションの設定〉ダイアログで［ヘルプ］ボタンをクリックしてください。**

【使用例】

- -- **オプションを指定すると，ヘルプ・メッセージがディスプレイに出力されます。**

```
C>st78k0 --

Structured assembler preprocessor for RA78K/0 Vx.xx [ xx xxx xx ]
      Copyright ( C ) NEC Electronics Corporation xxxx

Usage : st78k0 [ option [ ... ] ] input-file [ option [ ... ] ]

The option is as follows ( [ ] means omissible , ...means repetition ).
   -cx              : Select target chip. ( x = 000 , 012 , etc. ) * Must be specified.
   -o [ file ]      : Create the assembler source file [ with the specified name ].
   -e [ file ]       : Create the error list file [ with the specified name ].
   -ffile           : Input options or source file name from specified file.
   -idirectory      : Set include search path.
   -sc [ character ] :Specify the last character of word symbol.
   -wtn1/-wt [ n1 ] , n2/-wt [ n1 ] , [ n2 ] , n3
                    : Specify the number of tabs up to output position of each field.
                      n1 : Output position mnemonic field.
                      n2 : Output position operand field.    *Must be
                      n3 : Output position comment field.    0 <= n1 < n2 < n3 < 100.
   -dname [ = data ] [ , name [ = data ] [ ... ] ]
                    : Define name [ with data ].
   -gs/-ngs         : Output the structured assembler source debug information to assembler source file
                      / Not.
   -j               : Create the assembler source file if fatal error occurred.
   -zs/-ze/-zn      : Change source regulation.
                      -zs      : SJIS code usable in comment.

Press RETURN to continue...
   -ze              : EUC code usable in comment.
   -zn              : no multibyte code in comment.
   -ydirectory      : Set device file search path.
   --               : Show this message.

DEFAULT ASSIGNMENT : -o -e -scp -wt2 , 3 , 4 -gs
```

4.5 PM plus でのオプション設定

PM plus から構造化アセンブラ・オプションを設定する方法について説明します。

4.5.1 オプションの設定方法

PM plus の [ツール (T)] メニューの [構造化アセンブラオプションの設定 (S)] を選択するか，ツール・バーの [ST] ボタンを押下すると，〈構造化アセンブラオプションの設定〉ダイアログが現れます。

ダイアログ内で必要なオプションを入力することにより，各構造化アセンブラ・オプションを設定できます。

図 4-2 〈構造化アセンブラオプションの設定〉ダイアログ（《出力》タブ選択時）

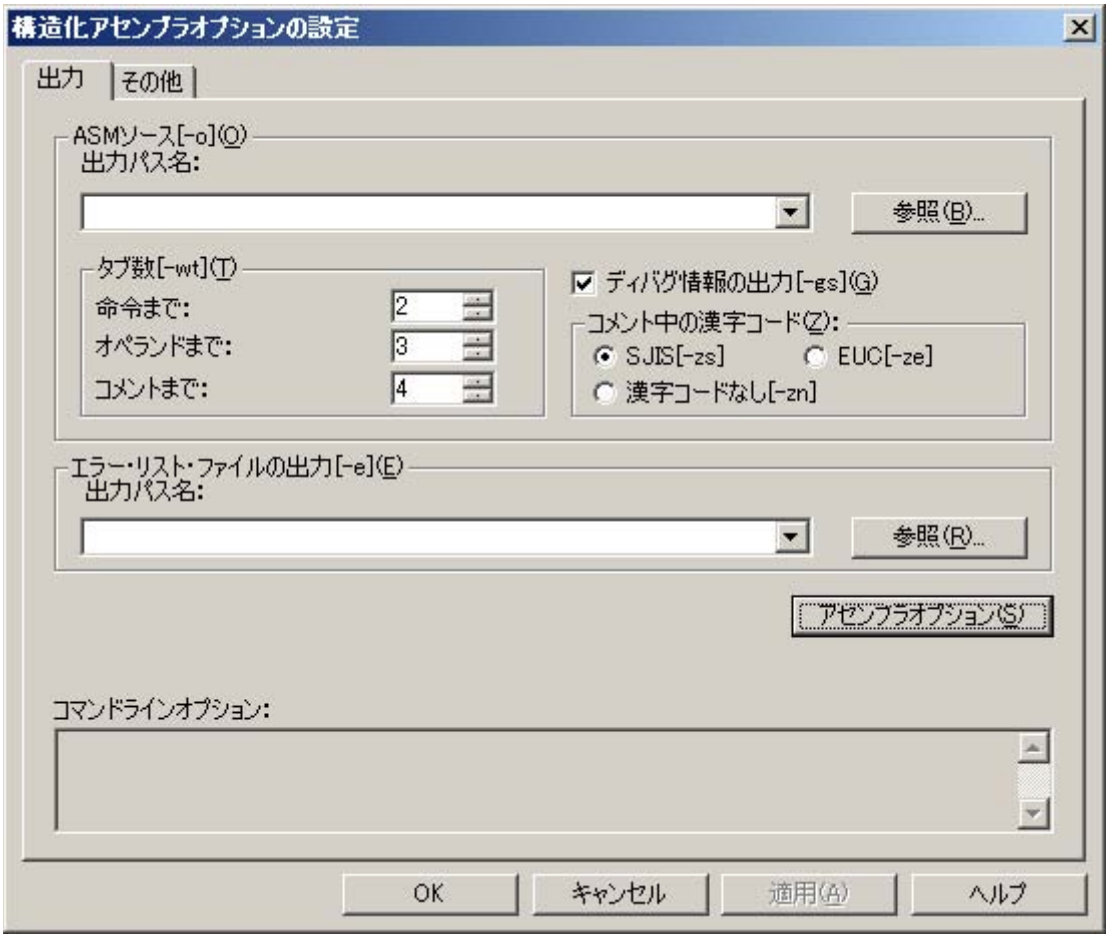


図 4-3 〈アセンブラオプション〉ダイアログ（[アセンブラオプション (S)] ボタン選択時）

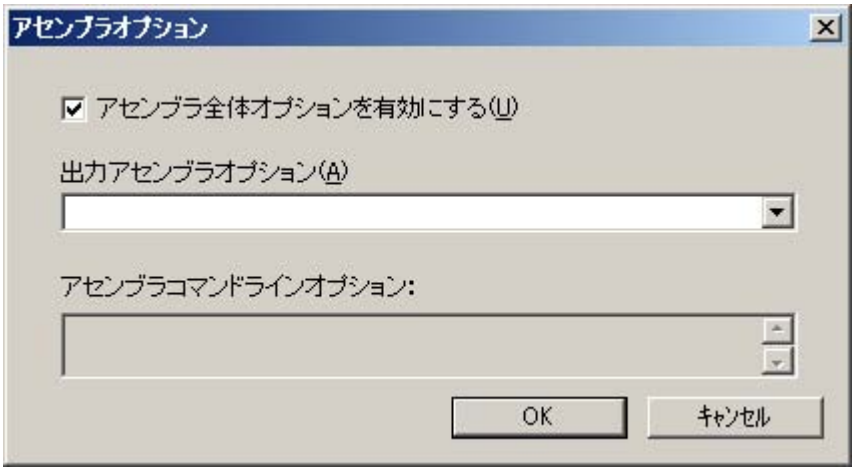
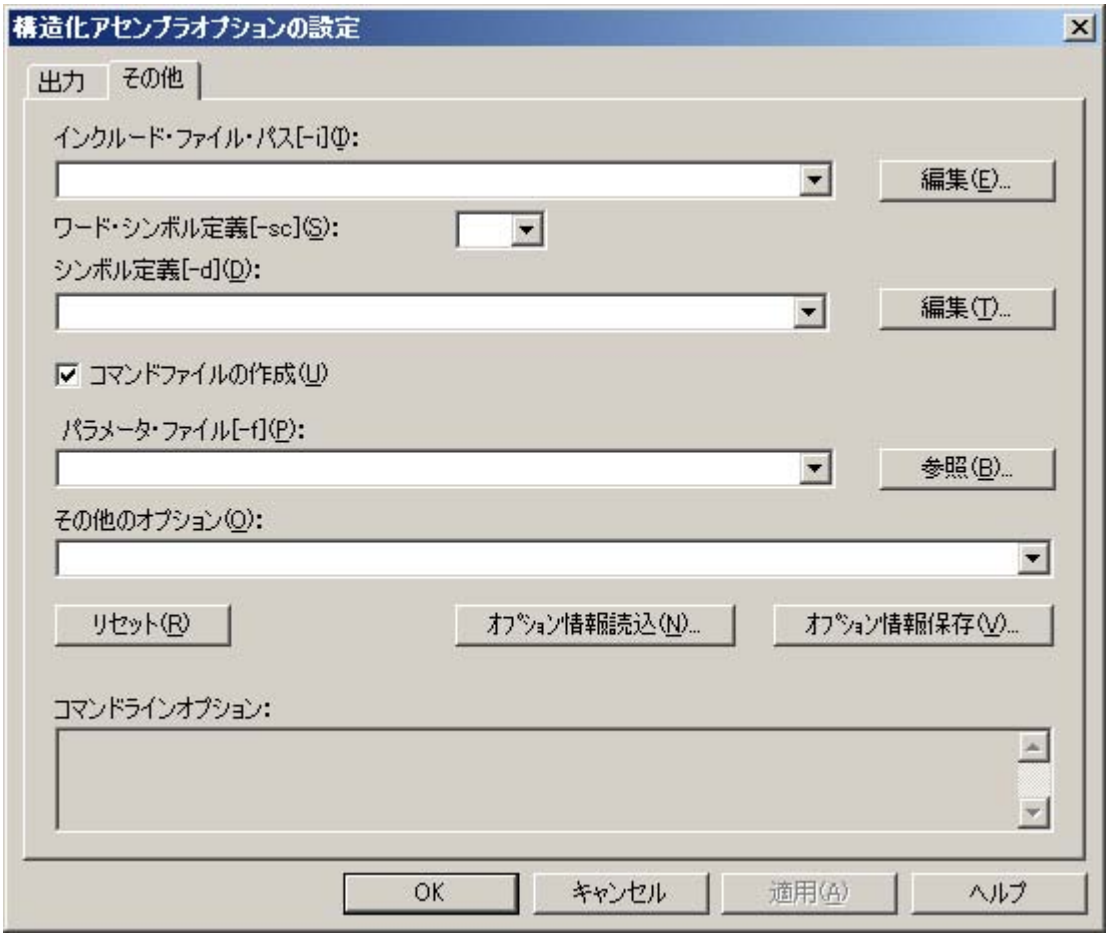


図 4-4 〈構造化アセンブラオプションの設定〉ダイアログ（《その他》タブ選択時）



4.5.2 各オプションの設定

〈構造化アセンブラオプションの設定〉ダイアログの各オプションについて，次に説明します。

《出力》タブ

- ASM ソース [-o](O)

  （全体オプションで指定する場合）出力パス名：

  ［参照 (B)］ボタン，または直接入力により，ASM ソース（2 次ソース・ファイル）のパスを指定します。

  （個別オプションで指定する場合）出力ファイル名：

  ［参照 (B)］ボタン，または直接入力により，ASM ソース（2 次ソース・ファイル）のパスとファイル名を指定します。

- タブ数 [-wt](T)

  変換されたアセンブリ言語のタブの数を指定します。

  命令までのタブ数，オペランドまでのタブ数，コメントまでのタブ数をそれぞれ個別に指定できます。

- ディバグ情報の出力 [-gs](G)

  ASM ソース（2 次ソース・ファイル）中にディバグ情報を出力する場合に，チェックします。

- コメント中の漢字コード (Z)

  ソースのコメント中で使用する漢字コードの種類 (SJIS[-zs]，EUC[-ze]，漢字コードなし [-zn]) を選択します。

- エラー・リスト・ファイルの出力 [-e](E)

  （全体オプションで指定する場合）出力パス名：

  エラー・リスト・ファイルを出力したい場合は，［参照 (R)］ボタン，または直接入力により，エラー・リスト・ファイルのパスを指定します。

  （個別オプションで指定する場合）出力ファイル名：

  エラー・リスト・ファイルを出力したい場合は，［参照 (R)］ボタン，または直接入力により，エラー・リスト・ファイルのパスとファイル名を指定します。

- アセンブラオプション (S)

  アセンブラ・ソース・モジュール・ファイルに対し，アセンブラ・オプションを指定します。

- コマンドラインオプション

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

- アセンブラ全体オプションを有効にする (U)

  〈アセンブラオプションの設定〉ダイアログで設定されている全体オプションを有効にする場合に，チェックします。

- 出力アセンブラオプション

  出力アセンブラ・ソースに対してオプションを有効にする場合に，オプション名を含めた文字列を入力します。

- アセンブラコマンドラインオプション

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

《その他》タブ

- **インクルード・ファイル・パス** [-i](I)

  ［編集 (E)**］ボタン，または直接入力により，インクルード・ファイルを読み込むパスを指定します。**

- **ワード・シンボル定義** [-sc](S)

  シンボル名でバイト／ワードを区別する必要がある場合に，判定の対象となるシンボルの最終文字を指定します。

- **シンボル定義** [-d](D)

  ［編集 (T)**］ボタン，または直接入力により，シンボルに定義づける数値を入力します。**

- **コマンドファイルの作成** (U)

  コマンドファイルを作成する場合に，チェックしてください。

- **パラメータ・ファイル** [-f](P)

  ［参照 (B)**］ボタン，または直接入力によりユーザ定義のパラメータ・ファイルを読み込みます。**

- **その他のオプション** (O)

  ダイアログで設定可能なオプション以外のオプションを指定したい場合に入力ボックスに入力します。

  注意　ヘルプ指定 (--) **オプションは，**PM plus **上では指定できません。**

- **リセット** (R)

  入力した内容をリセットします。

- **オプション情報読込** (N)

  〈オプション情報読込み〉ダイアログが開き，オプション情報ファイルを指定後，読み込みます。

- **オプション情報保存** (V)

  〈オプション情報の保存〉ダイアログが開き，オプション情報ファイルに名前をつけて保存します。

- **コマンドラインオプション**

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

4.5.3 オプションの編集ダイアログ

〈オプションの編集〉ダイアログは，項目をリストで編集します。

〈オプションの編集〉ダイアログについて，次に説明します。

図 4-5 〈オプションの編集〉ダイアログ

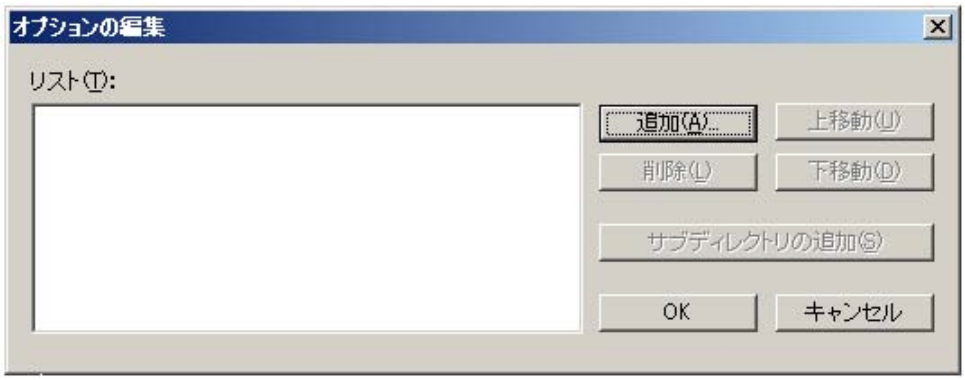
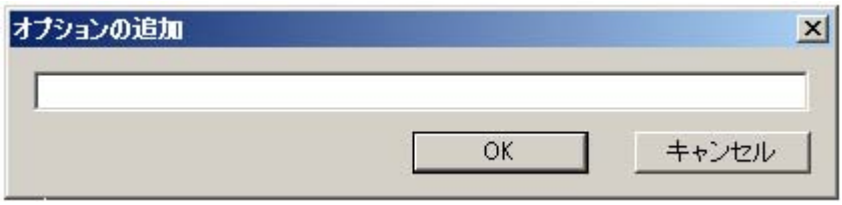


図 4-6 〈オプションの追加〉ダイアログ



- ［追加 (A)］ボタン

  リストの項目を追加します。

  ファイルやディレクトリを指定する項目の場合は，それぞれの〈参照〉ダイアログがオープンします。

  それ以外の場合は内容を入力する〈オプションの追加〉ダイアログがオープンします。

- ［削除 (L)］ボタン

  選択中のリストの項目を削除します。

- ［上移動 (U)］ボタン

  選択中のリストの項目を上に移動します。

- ［下移動 (D)］ボタン

  選択中のリストの項目を下に移動します。

- ［サブディレクトリの追加 (S)］ボタン

  次の場合は，選択中のリストの項目にサブディレクトリを追加できます。

  《その他》タブのインクルード・ファイル・パス [-i](I)

第 5 章　アセンブラ

この章では，アセンブラは，78K0 のアセンブリ言語で記述されたソース・モジュール・ファイルを入力し，それを機械語に変換してオブジェクト・モジュール・ファイルとして出力します。

さらに，アセンブル・リスト・ファイルやエラー・リスト・ファイルなどのリスト・ファイルを出力します。

アセンブル・エラーがある場合は，エラー・メッセージをアセンブル・リスト・ファイルやエラー・リスト・ファイルに出力し，エラーの原因を明示します。

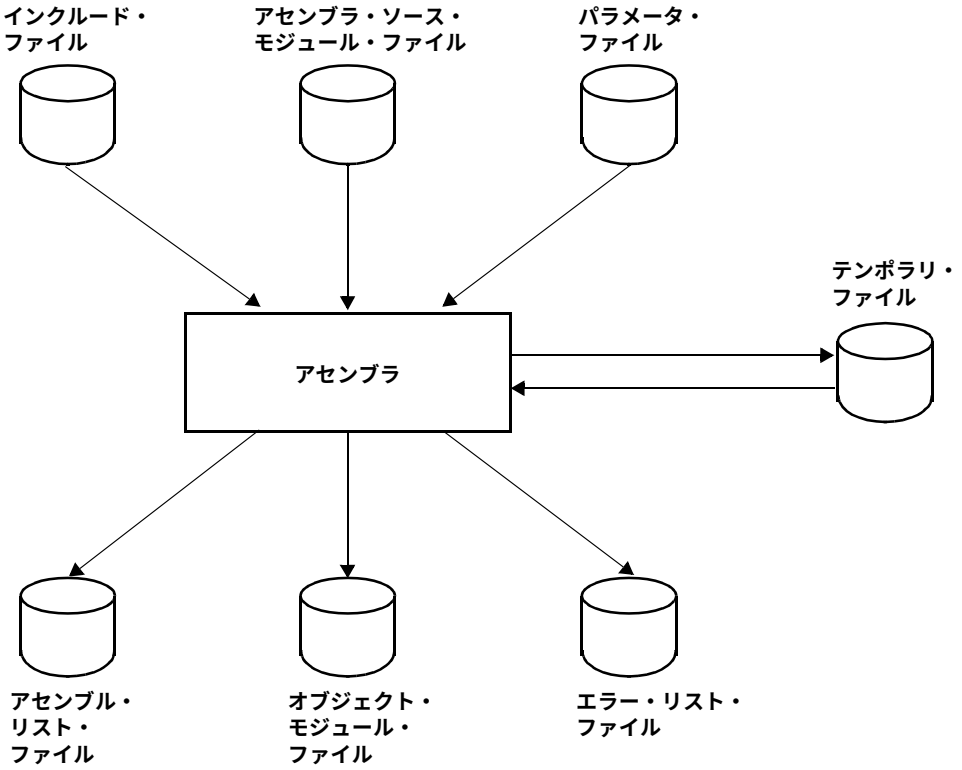
5.1　アセンブラの入出力ファイル

アセンブラの入出力ファイルを次に示します。

表 5-1 アセンブラの入出力ファイル

種類	ファイル名	説明	ディフォールト・ファイル・タイプ
入力ファイル	アセンブラ・ソース・モジュール・ファイル	- 78K0 用のアセンブリ言語で記述されたソース・モジュール・ファイルです。 - ユーザ作成ファイルです。	.asm
	インクルード・ファイル	- アセンブラ・ソース・モジュール・ファイルで参照するファイルです。 - 78K0 用のアセンブリ言語で記述されたファイルです。 - ユーザ作成ファイルです。	なし
	パラメータ・ファイル	- 実行プログラムのパラメータを内容とするファイルです。 - ユーザ作成ファイルです。	.pra
出力ファイル	オブジェクト・モジュール・ファイル	- 機械語情報と機械語の配置アドレスに関する再配置情報，およびシンボル情報を含んだバイナリ・ファイルです。	.rel
	アセンブル・リスト・ファイル	- アセンブル・リスト，クロスレファレンス・リストなどのアセンブル情報を持つファイルです。	.prn
	エラー・リスト・ファイル	- アセンブル時のエラー情報を持つファイルです。	.era
入出力ファイル	テンポラリ・ファイル	- アセンブルのためにアセンブラが自動生成するファイルです。アセンブル終了時には消去されます。	RAxxxxx.$$n （n = 1-4）

図 5-1 アセンブラの入出力ファイル



5.2 アセンブラの機能

(1) アセンブラは，ソース・モジュール・ファイルを読み込み，アセンブリ言語を機械語に変換します。

(2) アセンブラは，ファイルやシステムに関するエラーがある場合は，アボート・エラーを出力し，ソース・モジュール中に記述エラーを発見した場合は，フェータル・エラーやワーニング・エラーを出力します。
アボート・エラー，フェータル・エラーの場合は，通常オブジェクト・モジュール・ファイルは出力されません。ただし，-J オプションが指定された場合には，フェータル・エラーがある場合でもオブジェクト・モジュール・ファイルを出力します。

(3) アセンブラは，アセンブル起動時に指定されたアセンブラ・オプションに従ってアセンブル処理を行います。
アセンブラ・オプションの詳細については，「5.4 アセンブラ・オプション」を参照してください。

(4) アセンブラは，その処理を正常に終了すると終了メッセージを出力し，制御を OS に戻します。

5.3 アセンブラの起動方法

アセンブラの起動には，次の 2 つの方法があります。

5.3.1 コマンド行での起動

X>[**パス名**] ra78k0 [△**オプション**] …**ソース・モジュール・ファイル名** [△**オプション**] … [△]

↑ (a) ↑ (b) ↑ (c) ↑ (d) ↑ (e) ↑ (d)

(a) **カレント・ドライブ名**

(b) **カレント・ディレクトリ名**

(c) **アセンブラのコマンド・ファイル名**

(d) **アセンブラに対して動作の詳細を指示します。**
複数のアセンブラ・オプションを指定する場合は，それぞれのアセンブラ・オプション間を空白で区切ってください。アセンブラ・オプションの詳細については，「5.4 アセンブラ・オプション」を参照してください。
空白を含むパスを設定する場合には，ダブルクオーテーション（”“）で囲んでください。

(e) **アセンブルするソース・モジュール・ファイル名**
空白を含むパスのファイル名を指定する場合には，ダブルクオーテーション（”“）で囲んでください。

例 C>ra78k0 –c054 k0main.asm -e -np

5.3.2 パラメータ・ファイルによる起動

パラメータ・ファイルは，起動に必要な情報がコマンド行に指定しきれない場合や，アセンブルするたびに同じオプションを繰り返し指定するような場合に使用します。
パラメータ・ファイルを使用する場合には，コマンド行にパラメータ・ファイル指定オプション（-F）を指定します。

パラメータ・ファイルによる起動方法は，次のようになります。

```
X>ra78k0 [ △ソース・モジュール・ファイル ] △ -F パラメータ・ファイル名
                                        ↑          ↑
                                       (b)        (a)
```

(a) アセンブラの起動に必要な情報を含んだファイル
(b) パラメータ・ファイル（指定オプション）
パラメータ・ファイルは，エディタなどで作成します。

パラメータ・ファイル内での記述規則を次に示します。

```
[ [ [ △ ] オプション [ △オプション ] … [ △ ] △ ] ] …
```

コマンド行でソース・モジュール・ファイル名を省略した場合は，パラメータ・ファイル内でソース・モジュール・ファイル名を 1 つだけ指定できます。
ソース・モジュール・ファイル名は，オプションのあとに記述することも可能です。
パラメータ・ファイルには，コマンド行で指定するすべてのアセンブラ・オプション，出力ファイル名を記述します。
パラメータ・ファイルについての詳細は「5.4.3 アセンブラ・オプションの説明」を参照してください。

例　パラメータ・ファイル（k0main.pra）をエディタで作成します。

＜ k0main.pra の内容＞

```
; parameter file
k0main.asm -osample.rel
-psample.prn
```

パラメータ・ファイル（k0main.pra）を使用してアセンブラを起動します。

```
C>ra78k0 –fk0main.pra
```

5.3.3 実行開始メッセージ，終了メッセージ

(1) **実行開始メッセージ**

アセンブラが起動すると，次の実行開始メッセージが表示されます。

```
78K/0 Series Assembler Vx.xx [ xx xxx xx ]
    Copyright ( C ) NEC Electronics Corporation xxxx , xxxx
```

(2) **実行終了メッセージ**

アセンブルの結果，アセンブル・エラーが検出されなかった場合，アセンブラは次のメッセージを表示して制御を OS **に戻します。**

```
Pass1 Start
Pass2 Start

 Target chip : uPD78xxx
 Device file : Vx.xx

Assembly complete ,       0 error ( s ) and       0 warning ( s ) found.
```

アセンブルの結果，アセンブル・エラーが検出された場合・アセンブラはエラーの数を表示して制御を OS **に戻します。**

```
Pass1 Start
K0MAIN.ASM ( 12 ) : RA78K0 error E2201 : Syntax error
Pass2 Start
K0MAIN.ASM ( 12 ) : RA78K0 error E2201 : Syntax error
K0MAIN.ASM ( 29 ) : RA78K0 error E2407 : Undefined symbol reference ' CONVAH '
K0MAIN.ASM ( 29 ) : RA78K0 error E2303 : Illegal expression

 Target chip : uPD78xxx
 Device file : Vx. xx

Assembly complete ,       3 error ( s ) and       0 warning ( s ) found.
```

アセンブル中にアセンブラ処理継続が不可能な致命的エラーが検出された場合，アセンブラはメッセージを表示してアセンブルを中止し，制御を OS **に戻します。**

例 1　存在しないソース・モジュール・ファイルを指定した場合

```
C>ra78k0 sample.asm

78K/0 Series Assembler Vx.xx [ xx xxx xx ]
    Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

RA78K0 error F2006 : File not found 'SAMPLE.ASM'
Program aborted.
```

この例では，存在しないソース・モジュール・ファイルを指定したためにエラーとなり，アセンブルを中止しました。

例 2　存在しないアセンブラ・オプションを指定した場合

```
C>ra78k0 k0main.asm -z

78K/0 Series Assembler Vx.xx [ xx xxx xx ]
    Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

RA78K0 error F2012 : Missing parameter ' -z '
Please enter ' RA78K0-- ' , if you want help messages.
Program aborted.
```

この例では，存在しないアセンブラ・オプションを指定したためにエラーとなり，アセンブルを中止しました。
アセンブラがエラー・メッセージを出力してアセンブルを中止した場合，そのエラー・メッセージの原因を「第 12 章 エラー・メッセージ」で調べて対処してください。

5.4 アセンブラ・オプション

5.4.1 アセンブラ・オプションの種類

アセンブラ・オプションは，アセンブラの動作に細い指示を与えるものです。アセンブラ・オプションは，17種類のオプションに分類できます。

表 5-2 アセンブラ・オプション

分類	オプション	説明
デバイス種別指定	-C	対象デバイスの種別を指定します。
オブジェクト・モジュール・ファイル出力指定	-O	オブジェクト・モジュール・ファイルの出力を指定します。
	-NO	
オブジェクト・モジュール・ファイル強制出力指定	-J	強制的にオブジェクト・モジュール・ファイルを出力します。
	-NJ	
ディバグ情報出力指定	-G	ディバグ情報（ローカル・シンボル情報）をオブジェクト・モジュール・ファイルへ出力します。
	-NG	
	-GA	アセンブラ・ソース・ディバグ情報をオブジェクト・モジュール・ファイルへ出力します。
	-NGA	
インクルード・ファイル読み込みパス指定	-I	インクルード・ファイルを指定したパスから読み込みます。
アセンブル・リスト・ファイル出力指定	-P	アセンブル・リスト・ファイルの出力を指定します。
	-NP	
アセンブル・リスト・ファイル情報指定	-KA	アセンブル・リスト・ファイル中にアセンブル・リストを出力します。
	-NKA	
	-KS	アセンブル・リスト・ファイル中にシンボル・リストを出力します。
	-NKS	
	-KX	アセンブル・リスト・ファイル中にクロスレファレンス・リストを出力します。
	-NKX	
アセンブル・リスト・ファイル形式指定	-LW	アセンブル・リスト・ファイルの 1 行に印字する文字数を変更します。
	-LL	アセンブル・リスト・ファイルの 1 頁に印字する行数を変更します。
	-LH	アセンブル・リスト・ファイルのヘッダに指定された文字列を出力します。
	-LT	タブの展開文字数を変更します。
	-LF	アセンブル・リスト・ファイルの最後に改頁コードを付加します。
	-NLF	
エラー・リスト・ファイル出力指定	-E	エラー・リスト・ファイルを出力します。
	-NE	

表 5-2 アセンブラ・オプション

分類	オプション	説明
パラメータ・ファイル指定	-F	入力ファイル名，オプションを指定したファイルより入力します。
テンポラリ・ファイル作成パス指定	-T	テンポラリ・ファイルを指定したパスに作成します。
漢字コード指定	-ZS	コメントに記述された漢字をシフト JIS コードとして解釈します。
	-ZE	コメントに記述された漢字を EUC コードとして解釈します。
	-ZN	コメントに記述された文字を漢字として解釈しません。
デバイス・ファイル・サーチパス指定	-Y	デバイス・ファイルを指定されたパスから読み込みます。
シンボル定義指定	-D	シンボルの定義を行います。
シリーズ共通オブジェクト指定	-COMMON	78K0 シリーズ共通オブジェクト・モジュール・ファイルの出力を指定します。
セルフ・プログラミング指定	-SELF	セルフ・プログラミングを使用する際に指定します。
ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

5.4.2 アセンブラ・オプションの優先度

次の表に示すアセンブラ・オプションのうち，縦軸のものと横軸のものを同時に 2 つ以上指定した場合の優先度について説明します。

表 5-3 アセンブラ・オプションの優先度

	-NO	-NP	-NKA	-NKS	-KX	-NKX	--
-J	×						×
-G	×						×
-P			△	△		△	×
-KA		×					×
-KS		×			×		×
-KX		×					×
-LW		×					×
-LL		×					×
-LH		×					×
-LT		×					×
-LF		×					×

【×で記した箇所】

横軸に示したオプションを指定した場合，縦軸に示したオプションは無効となります。

例 C>ra78k0 –c054 k0main.asm -no -lw80 -lf

-LW，-LF オプションは，無効となります。

【△で記した箇所】

横軸に示したオプション 3 つをすべて同時に指定した場合，縦軸に示したオプションは無効となります。

例 C>ra78k0 –c054 k0main.asm -p -nka -nks -nkx

-NKA，-NKS，および -NKX が同時に指定されたので，-P オプションは無効となります。

また，-O/-NO オプションのようにオプション名の前に “ N ” を付加できるオプションを同時に指定した場合，あとで指定した方が有効となります。

例 C>ra78k0 –c054 k0main.asm -o -no

-NO オプションがあとに指定されているので，-O オプションは無効となり，-NO オプションが有効となります。

表 5-3 に記述されていないオプションは，他のオプションの影響を特に受けません。しかし，ヘルプ指定オプション（ -- ）が指定された場合には，すべてのオプション指定が無効となります。

5.4.3 アセンブラ・オプションの説明

次ページ以降に，各アセンブラ・オプションの詳細について説明します。

(1) デバイス種別指定

デバイス種別指定 （-C）

【記述形式】

-C デバイス種別

- 省略時解釈

  省略不可

【機能】

- -C オプションは，アセンブルの対象となる対象デバイスを指定します。

【用途】

- -C オプションは必ず指定してください。アセンブラは指定された対象デバイスに対してアセンブルを行い，それに対応したオブジェクト・コードを生成します。

【説明】

- -C オプションで指定できる対象デバイスは各デバイスの「デバイス・ファイル　使用上の留意点」を参照してください。

【注意】

- -C オプションは省略不可能です。ただし，ソース・モジュールの先頭で，-C オプションと同機能の制御命令を記述すれば，コマンド行での指定を省略できます。

△ $ △ PROCESSOR △ (△デバイス種別△) △ $ △ PC △ (△デバイス種別△)　　; 短縮形

制御命令については，「RA78K0 アセンブラ・パッケージ 言語編」のユーザーズ・マニュアルを参照してください。

【使用例】

- コマンド行で指定します。

```
C>ra78k0 –c054 k0main.asm
```

(2) オブジェクト・モジュール・ファイル出力指定

オブジェクト・モジュール・ファイル出力指定 （-O/-NO）

【記述形式】

```
-O [ 出力ファイル名 ]
-NO
```

- 省略時解釈

  -O 入力ファイル名 .rel

【機能】

- -O オプションは，オブジェクト・モジュール・ファイルの出力を指定します。また，その出力先や出力ファイル名を指定します。
- -NO オプションは，-O，-J，-G，-GA オプションを無効にします

【用途】

- オブジェクト・モジュール・ファイルの出力先や出力ファイル名を変更したいときに，-O オプションを指定します。
  アセンブル・リスト・ファイルの出力のみが目的でアセンブルする場合などは，-NO オプションを指定します。これによりアセンブル時間が短縮されます。

【説明】

- [出力ファイル名] には，ディスク型ファイル名，デバイス型ファイル名の NUL , AUX，パス名を指定できます。デバイス型ファイル名 CON , PRN , CLOCK が指定されたときは，アボート・エラーとなります。
- -O オプションを指定しても，フェータル・エラーがある場合には，オブジェクト・モジュール・ファイルは出力されません。
- -O オブジェクトを指定する際にドライブ名を省略すると，カレント・ドライブにオブジェクト・モジュール・ファイルが出力されます。
- -O オプションを指定する際に出力ファイル名を省略すると，オブジェクト・モジュール・ファイル名は，“ 入力ファイル名 .rel ” となります。
- -O と -NO の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

【使用例】

- オブジェクト・モジュール・ファイル（sample.rel）を出力します。

```
C>ra78k0 –c054 k0main.asm –osample.rel
```

(3) オブジェクト・モジュール・ファイル強制出力指定

オブジェクト・モジュール・ファイル強制出力指定 （-J/-NJ）

【記述形式】

```
-J
-NJ
```

- 省略時解釈

  -NJ

【機能】

- -J オプションは，フェータル・エラーでもオブジェクト・モジュール・ファイルを出力するように指定します。
- -NJ オプションは，-J オプションを無効にします。

【用途】

- 通常フェータル・エラーがある場合には，オブジェクト・モジュール・ファイルは出力されません。したがって，フェータル・エラーがあるのを承知でプログラムを実行させたい場合には，-J オプションを指定したオブジェクト・モジュール・ファイルを出力します。

【説明】

- -J オプションを指定すると，フェータル・エラーがある場合でも，オブジェクト・モジュール・ファイルが出力されます。
- -J と -NJ の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -NO オプションを指定した場合は，-J オプションは無効となります。

【使用例】

- フェータル・エラーの場合でもオブジェクト・モジュール・ファイルを出力するよう指定します。

```
C>ra78k0 –c054 k0main.asm -j
```

(4) ディバグ情報出力指定

ディバグ情報出力指定 （-G/-NG , -GA/-NGA）

(a) -G/-NG

【記述形式】

```
-G
-NG
```

- **省略時解釈**

  -G

【機能】

- -G **オプションは，オブジェクト・モジュール・ファイル中に，ディバグ情報（ローカル・シンボル情報）を付加するよう指示します。**
- -NG **オプションは，**-G **オプションを無効にします。**

【用途】

- -G **オプションは，ローカル・シンボルも含めシンボリック・ディバグを行うときに使用します。**
- -NG **オプションは，次の** 3 **種類の場合に使用します。**

  (i) **グローバル・シンボルのみのシンボリック・ディバグ**
  (ii) **シンボルなしでのディバグ**
  (iii) **オブジェクトのみを必要とするとき（**PROM **による評価時など）**

【説明】

- -G **と** -NG **の両オプションが同時に指定された場合は，あとで指定した方が有効となります。**
- -G/-NG **オプションと** -GA/-NGA **オプションが同時に指定された場合は，指定した位置にかかわらず** -GA/-NGA **オプションが有効となります。**
- -NO **オプションを指定した場合は，**-G **オプションは無効となります。**

【注意】

- **ソース・モジュールの先頭で，**-G , -NG **オプションと同機能の制御命令が記述できます。**
  次に記述形式を示します。

```
△ $ △ DEBUG
△ $ △ DG              ; 短縮形
△ $ △ NODEBUG
△ $ △ NODG            ; 短縮形
```

制御命令については，「RA78K0 **アセンブラ・パッケージ 言語編」のユーザーズ・マニュアルを参照してください。**

【使用例】

- **オブジェクト・モジュール・ファイルにディバグ情報を付加します。**

```
C>ra78k0 –c054 k0main.asm -g
```

(b) -GA/-NGA

【記述形式】

```
-GA
-NGA
```

- **省略時解釈**

  -GA

【機能】

- -GA **オプションは，オブジェクト・モジュール・ファイル中にアセンブラでソース・ディバグ用の情報を出力するよう指示します。**
- -NGA **オプションは，**-G，-GA **オプションを無効にします。**

【用途】

- -GA **オプションは，アセンブラのソース・レベルでディバグするときに使用します。ソース・レベルでのディバグには「統合ディバッガ（別売）」が必要です。**
- -NGA **オプションは，次の** 3 **種類の場合に使用します。**

  (i) **アセンブラ・ソースなしでのディバグ**
  (ii) **オブジェクトのみを必要とするとき（**PROM **による評価時など）**
  (iii) C **コンパイラ／構造化アセンブラ・ソース・レベルでのディバグ**

【説明】

- -GA **と** -NGA **の両オプションが同時に指定された場合は，あとで指定した方が有効となります。**
- -G/-NG **オプションと** -GA/-NGA **オプションが同時に指定された場合は，指定した位置にかかわらず** -GA/-NGA **オプションが有効となります。**
- -NO **オプションを指定した場合は，**-GA **オプションは無効となります。**

【注意】

- **ソース・モジュールの先頭で，**-GA，-NGA **オプションと同機能の制御命令が記述できます。次に記述形式を示します。**

```
△ $ △ DEBUGA
△ $ △ NODEBUGA
```

制御命令については，「RA78K0 **アセンブラ・パッケージ 言語編」のユーザーズ・マニュアルを参照してください。**

【使用例】

- **オブジェクト・モジュール・ファイルにアセンブラ・ソース・ディバグ情報を付加します。**

```
C>ra78k0 –c054 k0main.asm -ga
```

(5) インクルード・ファイル読み込みパス指定

インクルード・ファイル読み込みパス指定 （-I）

【記述形式】

-I パス名 [, パス名] …（複数指定可能）

- 省略時解釈

  インクルード・ファイルを次の順序で検索します。

  (i) ソース・ファイルのあるパス
  (ii) 環境変数（INC78K0）により指定されたパス

【機能】

- ソース・モジュール中の“ $include ”で指定されたインクルード・ファイルを指定したパスから入力するよう指示します。

【用途】

- インクルード・ファイルを，あるパスから検索したいときに指定します。

【説明】

- “ , ”で区切ることにより，一度に複数のパス名を指定できます。
- “ , ”の前後には空白を入れることはできません。
- “ $include ”で指定したインクルード・ファイルの検索順序は，次の検索順になります。

  (i) -I オプションに続いてパス名が複数指定された場合は，指定された順番でインクルード・ファイルを検索します。
  (ii) -I オプションが複数指定された場合は，後者の指定を優先する順番でインクルード・ファイルを検索します。
  (iii) -I オプションで指定したパスの検索後に，省略時解釈と同じ順序でインクルード・ファイルを検索します。

- -I に続けてパス名以外のものを指定した場合やパス名を省略した場合は，アボート・エラーとなります。
- -I が 65 個以上指定された場合には，アボート・エラーとなります。

【使用例】

- インクルード・ファイルをディレクトリ C:\SAMPLE から読み込みます。

```
C>ra78k0 –c054 k0main.asm –ic:\sample
```

(6) アセンブル・リスト・ファイル出力指定

アセンブル・リスト・ファイル出力指定 （-P/-NP）

【記述形式】

-P [**出力ファイル名**] -NP

- **省略時解釈**

  -P **入力ファイル名** .prn

【機能】

- -P **オプションは，アセンブル・リスト・ファイルの出力を指定します。**
  また，その出力先や出力ファイル名を指定します。
- -NP **オプションは，**-P，-KA，-KS，-KX，-LW，-LL，-LH，-LT，-LF **のオプションを無効にします。**

【用途】

- **アセンブル･リスト･ファイルの出力先や出力ファイル名を変更したいときに,**-P **オプションを指定します。**
- **オブジェクト・モジュール・ファイルの出力のみが目的でアセンブルする場合などに，**-NP **オプションを指定します。これによりアセンブル時間が短縮されます。**

【説明】

- **ファイル名としてディスク型ファイル名とデバイス型ファイル名を指定できます。**
  指定できるデバイス型ファイル名は，COM , PRN , NUL**，および** AUX **です。**CLOCK **を指定した場合，アボート・エラーとなります。**
- -P **オプションを指定する際に出力ファイル名を省略するとアセンブル・リスト・ファイル名は,“ 入力ファイル名** .prn **” になります。**
- -P **オプションを指定する際にドライブ名を省略すると，カレント・ドライブにアセンブル・リスト・ファイルが出力されます。**
- -P **と** -NP **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**

【使用例】

- **アセンブル・リスト・ファイル（**sample.prn**）を作成します。**

  C>ra78k0 –c054 k0main.asm -psample.prn

(7) アセンブル・リスト・ファイル情報指定

アセンブル・リスト・ファイル情報指定 （-KA/-NKA , -KS/-NKS , -KX/-NKX）

(a) -KA/-NKA

【記述形式】

```
-KA
-NKA
```

- **省略時解釈**

  -KA

【機能】

- -KA **オプションは，アセンブル・リスト・ファイル中にアセンブル・リストを出力します。**
- -NKA **オプションは，**-KA **オプションを無効にします。**

【用途】

- **アセンブル・リストを出力したいときに** -KA **オプションを指定します。**

【説明】

- -KA **と** -NKA **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**
- -NKA，-NKS，**および** -NKX **オプションを，すべて指定した場合は，アセンブル・リスト・ファイルは出力されません。**
- -NP **オプションを指定した場合は，**-kA **オプションは無効となります。**

【使用例】

- **アセンブル・リストを出力します。**

  C>ra78k0 –c054 k0main.asm -ka

  k0main.prn **を参照します。**

```
          Assemble list

ALNO  STNO   ADRS   OBJECT  M I    SOURCE STATEMENT
1     1
2     2                             NAME   SAMPM
3     3                      ; ************************************************
4     4                      ; *                                              *
5     5                      ; *       HEX -> ASCII Conversion Program        *
6     6                      ; *                                              *
7     7                      ; *              main-routine                    *
8     8                      ; *                                              *
9     9                      ; ************************************************
10    10
11    11                            PUBLIC MAIN , START
12    12                            EXTRN  CONVAH
13    13
14    14     ----                   DATA   DSEG   AT 0FE20H
15    15     FE20                   HDTSA: DS     1
16    16     FE21                   STASC: DS     2
17    17
18    18     ----                   CODE   CSEG   AT 0H
19    19     0000   R0000           MAIN:  DW     START
20    20
21    21     ----                          CSEG
22    22     0000                   START:
23    23
24    24
25    25
26    26     0000   11201A                 MOV    HDTSA , #1AH
27    27     0003   1620FE                 MOVW   HL , #HDTSA  ; set hex 2-code data in HL
        :
```

(b) -KS/-NKS

【記述形式】

```
-KS
-NKS
```

- **省略時解釈**

  -NKS

【機能】

- -KS **オプションは，アセンブル・リストに続いてシンボル・リストをアセンブル・リスト・ファイル中に出力します。**
- -NKS **オプションは，**-KS **オプションを無効にします。**

【用途】

- **シンボル・リストを出力したいときに，**-KS **オプションを指定します。**

【説明】

- -NKA , -NKS**，および** -NKX **オプションがすべて指定された場合は，アセンブル・リスト・ファイルは出力されません。**
- -KS **と** -KX **を同時に指定した場合は，**-KS **を無視します。**
- -KS **と** -NKS **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**
- -NP **オプションを指定した場合は，**-KS **オプションは無効となります。**

【使用例】

- **シンボル・リストを出力します。**

  C>ra78k0 –c054 k0main.asm -ks

  k0main.prn **を参照します。**

  アセンブル・リストに続いてシンボル・リストが出力されています。

```
          Symbol Table List

VALUE  ATTR    RTYP     NAME              VALUE  ATTR    RTYP     NAME

       CSEG             ?CSEG                    CSEG             CODE
----H          EXT      CONVAH                   DSEG             DATA
FE20H  ADDR              HDTSA               0H   ADDR    PUB      MAIN
       MOD     SAMPM                        0H   ADDR    PUB      START
FE21H  ADDR              STASC
```

(c) -KX/-NKX

【記述形式】

```
-KX
-NKX
```

- **省略時解釈**

  -NKX

【機能】

- -KX **オプションは，アセンブル・リストに続いてクロスレファレンス・リストをアセンブル・リスト・ファイル中に出力します。**
- -NKX **オプションは，**-KX **オプションを無効にします。**

【用途】

- **ソース・モジュール・ファイルで定義された各シンボルが，ソース・モジュール中のどこでどれだけ参照されているか，また，アセンブル・リストの何行目の記述で，そのシンボルを参照しているのかなどの情報を知りたいときにクロスレファレンス・リストを出力します。**

【説明】

- -NKA , -NKS**，および** -NKX **オプションがすべて指定された場合は，アセンブル・リスト・ファイルは出力されません。**
- -KS **と** -KX **を同時に指定した場合，**-KS **を無視します。**
- -KX **と** -NKX **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**
- -NP **オプションを指定した場合は，**-KX **オプションは無効となります。**

【注意】

- **ソース・モジュールの先頭で，**-KX/-NKX **オプションと同機能の制御命令を記述できます。**
  その記述形式を次に示します。

```
△ $ △ XREF
△ $ △ XR              ; 短縮形
△ $ △ NOXREF
△ $ △ NOXR            ; 短縮形
```

制御命令については，「RA78K0 **アセンブラ・パッケージ 言語編」のユーザーズ・マニュアルを参照してください。**

【使用例】

- **クロスレファレンス・リストを出力します。**

```
C>ra78k0 –c054 k0main.asm -kx
```

k0main.prn **を参照します。**

アセンブル・リストに続いてクロスレファレンス・リストが出力されています。

```
            Cross-Reference List

NAME        VALUE   R       ATTR    RTYP    SEGNAME     XREFS

?CSEG                       CSEG            ?CSEG       21#
CODE                        CSEG            CODE        18#
CONVAH      ----H   E               EXT                 12@     29
DATA                        DSEG            DATA        14#
HDTSA       FE20H           ADDR            DATA        15#     26      27
MAIN        0H              ADDR    PUB     CODE        11@     19#
SAMPM                       MOD                         2#
START       0H      R       ADDR    PUB     ?CSEG       11@     19      22#
STASC       FE21H           ADDR            DATA        16#     31
```

(8) アセンブル・リスト・ファイル形式指定

アセンブル・リスト・ファイル形式指定 （-LW , -LL , -LH , -LT , -LF/-NLF）

(a) -LW

【記述形式】

-LW [文字数]

- 省略時解釈

  -LW132（ディスプレイ出力の場合は 80 文字とします）

【機能】

- -LW オプションは，リスト・ファイルの 1 行の文字数を指定します。

【用途】

- 各種リスト・ファイルの 1 行の文字数を変更したいとき，-LW オプションを指定します。

【説明】

- -LW オプションで指定できる文字数の範囲（ディスプレイ出力の場合は 80 文字まで）を次に示します。

  72 ≦ 1 行に印字する文字数 ≦ 2046

  範囲外の数値や数値以外を指定した場合は，アボート・エラーとなります。
- 文字数が省略された場合は，132 を指定したものとなみします。
  ただし，アセンブル・リスト・ファイルの出力先がディスプレイの場合は 80 となります。
- 指定する文字数にはターミネータ（CR , LF）は含みません。
- -NP オプションを指定した場合は，-LW オプションは無効となります。

【注意】

- ソース・モジュールの先頭で，-LW オプションと同機能の制御命令を記述できます。
  その記述形式を次に示します。

△ $ △ WIDTH

  制御命令については，「RA78K0 アセンブラ・パッケージ 言語編」のユーザーズ・マニュアルを参照してください。

【使用例】

- **アセンブル・リスト・ファイルの** 1 **行の文字数を** 80 **文字に指定します。**

```
C>ra78k0 –c054 k0main.asm -lw80
```

アセンブル・リストを参照します。

```
          Assemble list

ALNO  STNO    ADRS    OBJECT  M I     SOURCE STATEMENT

1       1
2       2                                       NAME    SAMPM
3       3                               ; ***************************************************
4       4                               ; *                                                 *
5       5                               ; *         HEX -> ASCII Conversion Program         *
6       6                               ; *                                                 *
7       7                               ; *                 main-routine                    *
8       8                               ; *                                                 *
9       9                               ; ***************************************************
10      10
11      11                                      PUBLIC MAIN , START
12      12                                      EXTRN  CONVAH
13      13
14      14      ----                    DATA    DSEG    AT 0FE20H
15      15      FE20                    HDTSA : DS      1
16      16      FE21                    STASC : DS      2
17      17
18      18      ----                    CODE    CSEG    AT 0H
19      19      0000 R0000              MAIN :  DW      START
20      20
21      21      ----                            CSEG
22      22      0000                    START :
23      23
24      24
25      25
          :
```

(b) -LL

【記述形式】

-LL [**行数**]

- **省略時解釈**

  -LL66（**ディスプレイ出力の場合は改頁しません**）

【機能】

- -LL **オプションは，アセンブル・リスト・ファイルの** 1 **頁の行数を指定します。**

【用途】

- **アセンブル・リスト・ファイルの** 1 **頁の行数を変更したいときに，**-LL **オプションで指定します。**

【説明】

- -LL **オプションで指定できる行数の範囲を次に示します。**

  20 ≦ 1 **頁に印字する行数** ≦ 32767

  範囲外の数値や数値以外のものが指定された場合には，アボート・エラーとなります。
- **行数を省略した場合，**66 **を指定したものとみなします。**
- **行数の** 0 **を指定した場合は改頁しません。**
- -NP **オプションを指定した場合は，**-LL **オプションは無効となります。**

【注意】

- **ソース・モジュールの先頭で，**-LL **オプションと同機能の制御命令を記述できます。**
  その記述形式を次に示します。

△ $ △ LENGTH

  制御命令については，「RA78K0 **アセンブラ・パッケージ 言語編」のユーザーズ・マニュアルを参照してください。**

【使用例】

- **アセンブル・リスト・ファイルの** 1 **頁の行数を** 20 **行に指定します。**

C>ra78k0 –c054 k0main.asm -ll20

k0main.prn **を参照します。**

```
78K/0 Series Assembler Vx.xx                                Date:xx xxx xxxx Page:   1

Command:-c054 k0main.asm -ll20
Para-file:
In-file:K0MAIN.ASM
Obj-file:K0MAIN.REL
Prn-file:K0MAIN.PRN

          Assemble list

------------------------------------------------------------------------------



78K/0 Series Assembler Vx.xx                                Date:xx xxx xxxx Page:   2

ALNO  STNO    ADRS    OBJECT  M I     SOURCE STATEMENT

1       1
2       2                                       NAME      SAMPM
3       3                                ; ************************************************
4       4                                ; *                                              *
5       5                                ; *       HEX -> ASCII Conversion Program  *
6       6                                ; *                                              *
7       7                                ; *               main-routine *

------------------------------------------------------------------------------



78K/0 Series Assembler Vx.xx                                Date : xx xxx xxxx Page:   3


ALNO  STNO    ADRS    OBJECT  M I     SOURCE STATEMENT

8       8                                ; *                                              *
9       9                                ; ************************************************
10      10
11      11                                       PUBLIC    MAIN , START
          :
```

(c) -LH

【記述形式】

-LH 文字列

- **省略時解釈**

  なし

【機能】

- -LH **オプションは，アセンブル・リスト・ファイルのヘッダのタイトル欄に印字する文字列を指定します。**

【用途】

- **アセンブル・リスト・ファイルの内容を端的に表すようなタイトルを表示したいときに，**-LH **オプションを指定します。**
- **タイトルを各頁に印字することで，アセンブル・リスト・ファイルの内容がひと目でわかります。**

【説明】

- **指定可能な文字列は，**60 **文字以内です。ただし，文字列内に空白は記述できません。**
- 61 **文字以上記述された場合には，先頭の** 60 **文字を有効とし，エラーは出力されません。**

  なお，漢字，ひらがなは，1 **文字を** 2 **文字として計算します。**

  1 **行の最大文字数が** 119 **以下の場合，タイトルとしての文字列の有効長は次のとおりです。**

  有効長 =（1 **行の最大文字数**）- 60
- **文字列が指定されなかった場合，アボート・エラーとなります。**
- -NP **オプションを指定した場合は，**-LH **オプションは無効となります。**
- -LH **オプションを省略した場合，アセンブル・リスト・ファイルのタイトル欄は，空白となります。**
- **記述可能な文字セットを次に示します。**

表 5-4 **タイトルとして記述可能な文字**

文字	コマンド行	パラメータ・ファイル中
* ? > < |	”” **でくくることで記述可能**	**記述可能** ”” **でくくってもコマンド行と同じように解釈します**
;	”” **でくくることで記述可能**	**記述不可** **（コメントとみなされます）**
#	**記述可能**	**記述不可** **（コメントとみなされます）**
”**（ダブル・クォーテーション）**	**有効文字としては記述不可**	**有効文字としては記述不可**
00H	**記述不可**	**記述可能** **ただし，文字列が切れたとみなされます**

表 5-4 タイトルとして記述可能な文字

文字	コマンド行	パラメータ・ファイル中
03H , 06H , 08H , 0DH , 0EH , 10H , 15H , 17H , 18H , 1BH , 7FH	記述不可	記述可能 ただしアセンブル・リスト・ファイル中では“！”で表示 （単独の 0DH はリストには出力されません）
01H , 02H , 04H , 05H , 07H , 0BH , 0CH , 0FH , 11H , 12H , 13H , 14H , 16H , 19H , 1CH , 1DH , 1EH , 1FH	記述可能 ただしアセンブル・リスト・ファイルでは“！”で表示	記述可能 ただしアセンブル・リスト・ファイル中では“！”で表示
1AH	記述可能 ただしアセンブル・リスト・ファイルでは“！”で表示	記述不可 （ファイルの終わり）
英字	大／小文字がそのまま入力されます	大／小文字がそのまま入力されます
その他	記述可能	記述可能

備考　起動行上の * は，ワイルド・カード展開の対象とならない場合は " " でくくらなくても記述可能です。

【注意】

- ソース・モジュールの先頭で，-LH オプションと同機能の制御命令を記述できます。

  次に記述形式を示します。

```
△$△TITLE△(△'文字列'△)
△$△TT△(△'文字列'△)              ；短縮形
```

  制御命令については，「RA78K0 アセンブラ・パッケージ 言語編」のユーザーズ・マニュアルを参照してください。

【使用例】

- **アセンブル・リスト・ファイルのヘッダにタイトルを印字します。**

C>ra78k0 –c054 k0main.asm -lhRA78K0_MAINROUTINE

78k0main.prn **を参照します。**

```
78K/0 Series Assembler Vx.xx    RA78K0_MAINROUTINE              Date:xx xxx xxx Page :    1
                                          ↑
                                       タイトル

Command :    -c054 k0main.asm -lhRA78K0_MAINROUTINE
Para-file :
In-file : K0MAIN.ASM
Obj-file : K0MAIN.REL
Prn-file : K0MAIN.PRN

           Assemble list

ALNO  STNO    ADRS    OBJECT  M I      SOURCE STATEMENT
1       1
2       2                                       NAME      SAMPM
3       3                                ; ************************************************
4       4                                ; *                                              *
5       5                                ; *         HEX -> ASCII Conversion Program      *
6       6                                ; *                                              *
7       7                                ; *                  main-routine*
         :
```

(d) -LT

【記述形式】

-LT [**文字数**]

- **省略時解釈**

  -LT8

【機能】

- -LT **オプションは，ソース・モジュール中の** HT（Horizontal Tabulation）**コードを，各種リスト上でいくつかのブランク（空白）に置き換えて出力する（タビュレーション処理）ための基本となる文字数を指定します。**

【用途】

- -LW **オプションで，各種リストの** 1 **行の文字数を少なく指定した場合に，**HT **コードによるブランクを少なくし，文字数を節約するために，**-LT **オプションを指定します。**

【説明】

- -LT **オプションで指定できる文字数の範囲は次のとおりです。**

  0 ≦ **指定できる文字数** ≦ 8

  範囲外の数値や数値以外のものを指定した場合は，アボート・エラーとなります。
- -LT0 **を指定した場合，タビュレーション処理は行わず，タブ・コードを出力します。**
- -NP **オプションを指定した場合は，**-LT **オプションは無効となります。**

【注意】

- **ソース・モジュールの先頭で，**-LT **オプションと同機能の制御命令を記述できます。**

  次に記述形式を示します。

△ $ △ TAB △**タブ数**

  制御命令については，「RA78K0 **アセンブラ・パッケージ 言語編」のユーザーズ・マニュアルを参照してください。**

【使用例】

例 1　　-LT **オプションを省略した場合の** sample.prn **を参照します。**

```
                Assemble list

ALNO    STNO    ADRS     OBJECT   M I  SOURCE STATEMENT

1       1                                    NAME SAMPLE
2       2
3       3       ----                 CODE    CSEG
4       4       0000     63                  MOV   A , B
5       5       0001     619A                SET1  A.1
6       6                                    END
```

例 2　HT **コードによるブランクを** 1 **に指定します。**

C>ra78k0 -c054 sample.asm -It1

sample.prn **を参照します。**

```
                Assemble list

ALNO    STNO    ADRS     OBJECT   M I  SOURCE  STATEMENT

1       1                                    NAME SAMPLE
2       2
3       3       ----                 CODE    CSEG
4       4       0000     63                  MOV   A , B
5       5       0001     619A                SET1  A.1
6       6                                    END
```

備考　HT **コードによるブランクは** 1 **つです。**

(e) -LF/-NLF

【記述形式】

```
-LF
-NLF
```

- **省略時解釈**

  -NLF

【機能】

- -LF **オプションは，アセンブル・リスト・ファイルの最後に改頁コード**（FF）**の付加を指定します。**
- -NLF **オプションは，**-LF **オプションを無効にします。**

【用途】

- **アセンブル・リスト・ファイルの内容を印字したあとで改頁しておきたい場合には，**-LF **オプションを指定して改頁を付加します。**

【説明】

- -NP **オプションを指定した場合は，**-LF **オプションは無効となります。**
- -LF **と** -NLF **の両オプションを同時に指定した場合は，あとで指定した方を優先します。**

【注意】

- **ソース・モジュールの先頭で，**-LF/-NLF **オプションと同機能の制御命令を記述できます。**
  次に記述形式を示します。

```
△ $ △ FORMFEED
△ $ △ NOFORMFEED
```

  制御命令については，「RA78K0 **アセンブラ・パッケージ 言語編」のユーザーズ・マニュアルを参照してください。**

【使用例】

アセンブル・リスト・ファイルの最後に改頁コードを付加します。

```
C>ra78k0 –c054 k0main.asm -pprn -lf
```

(9) エラー・リスト・ファイル出力指定

エラー・リスト・ファイル出力指定 （-E/-NE）

【記述形式】

-E [出力ファイル名] -NE

- **省略時解釈**

  -NE

【機能】

- -E **オプションは，エラー・リスト・ファイルの出力を指定します。また，その出力先や出力ファイル名を指定します。**
- -NE **オプションは，**-E **オプションを無効にします。**

【用途】

- **エラー・メッセージをファイルに保存しておきたい場合，**-E **オプションを指定します。**
- **エラー・リスト・ファイルの出力先や出力ファイル名を変更したいときに，**-E **オプションで指定します。**

【説明】

- **ファイル名としてディスク型ファイル名とデバイス型ファイル名を指定できます。**
  ただし，デバイス型ファイル名 CLOCK **を指定した場合は，アボート・エラーとなります。**
- -E **オプションを指定する際に，出力ファイル名を省略するとエラー・リスト・ファイル名は，“ 入力ファイル名** .era **” となります。**
- -E **オプションを指定する際にドライブ名を省略すると，カレント・ドライブにエラー・リスト・ファイルが出力されます。**
- -E **と** -NE **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**

【使用例】

- **エラー・リスト・ファイル（**k0main.era**）を作成します。**

```
C>ra78k0 –c054 k0main.asm -ek0main.era
```

```
78K/0 Series Assembler Vx.xx [ xx xxx xx ]
    Copyright ( C ) NEC Electronics Corporation xxxx ,  xxxx

Pass1 Start
K0MAIN.ASM ( 31 ) : RA78K0 error E2202 : Illegal operand
Pass2 Start
K0MAIN.ASM ( 26 ) : RA78K0 error E2312 : Operand out of range ( byte )
K0MAIN.ASM ( 31 ) : RA78K0 error E2202 : Illegal operand

 Target chip : uPD78054
 Device file : Vx.xx

Assembly complete ,       2 error ( s) and       0 warning ( s ) found.
```

エラー・リスト・ファイル（k0main.era**）を参照します。**

```
Pass1 Start
K0MAIN.ASM ( 31 ) : RA78K0 error E2202 : Illegal operand
Pass2 Start
K0MAIN.ASM ( 26 ) : RA78K0 error E2312: Operand out of range ( byte )
K0MAIN.ASM ( 31 ) : RA78K0 error E2202: Illegal operand
```

(10) パラメータ・ファイル指定

パラメータ・ファイル指定 （-F）

【記述形式】

-F ファイル名

- 省略時解釈

  入力ファイルなし

【機能】

- -F オプションは，オプションあるいは入力ファイル名を指定のファイルから入力する指定をします。

【用途】

- コマンド行ではアセンブラの起動に必要な情報を指定しきれないときに，-F オプションを指定します。
- アセンブルするたびに繰り返し同じようにオプションを指定する際には，それらをパラメータ・ファイルに記述しておき，-F オプションで指定します。

【説明】

- “ ファイル名 ” として指定できるのは，ディスク型ファイル名のみです。
  デバイス型ファイル名を指定するとアボート・エラーとなります。
- ファイル名を省略するとアボート・エラーとなります。
- パラメータ・ファイルのネストは許されません。パラメータ・ファイル中で，-F オプションを指定するとアボート・エラーとなります。
- パラメータ・ファイル中に記述できる文字数の制限はありません。
- 空白とタブ，および改行文字（LF）をオプションあるいは入力ファイル名の区切りとします。
- パラメータ・ファイル中に記述したオプションあるいは入力ファイル名はコマンド行上のパラメータ・ファイル指定のあった位置に展開されます。
- 展開されたオプションは，あとで指定したものを優先します。
- “ ; ”，または “ # ” 以降に記述された文字は改行文字（LF），または EOF の前まですべてコメントと解釈します。
- -F オプションを複数指定するとアボート・エラーとなります。

【使用例】

- パラメータ・ファイルを使用してアセンブルします。
  パラメータ・ファイル（k0main.pra）の内容は次のように設定します。

```
; parameter file
k0main.asm -osample.rel –g –c054
-psample.prn
```

コマンド行には，次のように入力します。

```
C>ra78k0 –fk0main.pra
```

(11) テンポラリ・ファイル作成パス指定

テンポラリ・ファイル作成パス指定 （-T）

【記述形式】

-T パス名

- 省略時解釈

  環境変数 TMP により指定されたパスに作成します

  指定されていない場合は，カレント・パスに作成します

【機能】

- -T オプションは，テンポラリ・ファイルを作成するパスを指定します。

【用途】

- テンポラリ・ファイルの作成場所を指定できます。

【説明】

- パス名としてパス以外のものは指定できません。
- パス名は省略できません。
- 以前に作成されたテンポラリ・ファイルが存在している場合でも，ファイル保護がされていなければ，上書きします。
- 必要とするメモリ・サイズがある間は，テンポラリ・ファイルをメモリに展開します。
  メモリが足りなくなった時点でメモリに展開していたテンポラリ・ファイルの内容をディスクに書き出します。
  以降のテンポラリ・ファイルへのアクセスは，セーブしたディスク・ファイルに対して行います。
- テンポラリ・ファイルは，アセンブル終了時に削除されます。また，キー入力（CTRL-C）によってアセンブルが中止されたときにも削除されます。
- テンポラリ・ファイルの作成パスは，次の順番で決定されます。

  (i) -T オプションで指定されたパス

  (ii) 環境変数 TMP に設定されているパス（-T オプション省略の場合）

  (iii) カレント・パス（TMP が設定されていない場合）

  なお，(i) または (ii) を指定した場合，指定されたパスにテンポラリ・ファイルが作成できなければアボート・エラーとなります。

【使用例】

- テンポラリ・ファイルをディレクトリ C:\TMP に出力するよう指定します。

```
C>ra78k0 –c054 k0main.asm –tc:\tmp
```

(12) 漢字コード指定

漢字コード指定 （-ZS/-ZE/-ZN）

【記述形式】

```
-ZS
-ZE
-ZN
```

- **省略時解釈**

  OS **により次の通りとなります。**

  -ZS（Windows/HP-UX）

  -ZE（SunOS/Solaris）

【機能】

- **コメントに記述された漢字を，指定された漢字コードとして解釈します。**
- **オプションにより，漢字コードを次のように解釈します。**

  -ZS：**シフト** JIS **コード**

  -ZE：EUC **コード**

  -ZN：**漢字として解釈しません。**

【用途】

- **コメント行の，漢字の漢字コードの解釈を指定するときに使用します。**

【説明】

- -ZS/-ZE/-ZN **オプションが同時に指定された場合は，あとで指定した方が有効となります。**
- **ソース・モジュールの先頭で，**-ZS/-ZE/-ZN **オプションと同機能の制御命令を記述できます。**

  次に記述形式を示します。

```
△ $ △ KANJICODE △ SJIS
△ $ △ KANJICODE △ EUC
△ $ △ KANJICODE △ NONE
```

  制御命令については，「RA78K0 **アセンブラ・パッケージ 言語編」のユーザーズ・マニュアルを参照してください。**
- **環境変数（**LANG78K**）でも漢字コードを指定することができます。環境変数については，「**11.2 **開発環境の整備（環境変数）」を参照してください。**

【使用例】

- **漢字コードを** EUC **コードとして解釈します。**

```
C>ra78k0 k0main.asm –c054 -ze
```

(13) デバイス・ファイル・サーチパス指定

デバイス・ファイル・サーチパス指定 （-Y）

【記述形式】

-Y パス名

- 省略時解釈

  デバイス・ファイルを読み込むパスは，次の順序で調べ決定します。

  (i) < .. \dev >（ra78k0.exe の起動されたパスに対して）

  (ii) RA78K0 の起動されたパス

  (iii) カレント・ディレクトリ

  (iv) 環境変数 PATH

【機能】

- デバイス・ファイルを指定されたパスから読み込みます。

【用途】

- デバイス・ファイルが存在するパスを指定します。

【説明】

- -Y オプションに続けてパス名以外が指定された場合，アボート・エラーとなります。
- -Y オプションに続けて指定するパス名が省略された場合，アボート・エラーとなります。
- デバイス・ファイルを読み込むパスは，次の順序で調べ決定します。

  (i) -Y オプションで指定されたパス

  (ii) < .. \dev >（ra78k0.exe の起動されたパスに対して）

  (iii) RA78K0 の起動されたパス

  (iv) カレント・ディレクトリ

  (v) 環境変数 PATH

【使用例】

- デバイス・ファイルのパスを c:\78k0\dev ディレクトリに指定します。

```
C>ra78k0 k0main.asm –c054 –yc:\78k0\dev
```

(14) シンボル定義指定

シンボル定義指定 （-D）

【記述形式】

-D **シンボル名** [= **数値**] [, **シンボル名** [= **数値**] …]

- **省略時解釈**

  なし

【機能】

- **シンボルの定義を行います。**

【用途】

- **シンボルの定義を行いたい場合，**-D **オプションを指定します。**

【説明】

- **シンボルに与える数値は，**2 **進数，**8 **進数，**10 **進数，および** 16 **進数とします。数値指定が省略された場合は，**1 **が指定されたものとします。**
- **シンボルはカンマで区切ることにより，**30 **個まで指定できます。**
- **シンボル名は，**31 **文字まで記述できます。**
- **同名のシンボル名が重複された場合，あとで指定した方が有効となります。**
- **シンボル名の英字は，大文字，小文字を区別します。**
  -D **で定義したシンボルは，**EQU/$SET/$RESET **の代わりとなります。**-D **に指定したシンボル名がソースでも定義されていた場合，エラーとなります。**

【使用例】

- **シンボルの定義を** 2 **と指定します。**

```
C>ra78k0 k0main.asm –c054 –dSYM = 2
```

(15) シリーズ共通オブジェクト指定

シリーズ共通オブジェクト指定 （-COMMON）

【記述形式】

-COMMON

- **省略時解釈**

  指定したデバイスに対応したオブジェクト・ファイルを出力します。

【機能】

- -COMMON **オプションは，**78K0 **シリーズ共通オブジェクト・モジュール・ファイルの出力を指定します。**

【用途】

- **デバイス種別指定（**-C**）オプションに関係なく，**78K0　**シリーズ共通で使用することができるオブジェクト・コードを生成します。**

  78K0 **シリーズの異なるデバイスを指定されたオブジェクト・ファイルとのリンクが可能になります。**

【説明】

- 78K0 **シリーズ共通で使用することができるオブジェクト・コードを生成したい場合に指定してください。**

【注意】

- -COMMON **オプションを指定した場合でも，デバイス種別指定（**-C**）オプション，または同機能の制御命令を省略することはできません。**

  リンクの際に，入力したオブジェクト・モジュール・ファイルすべてに対してシリーズ共通オブジェクト指定（-COMMON**）オプションが指定されているとエラーになります。**

【使用例】

```
C>ra78k0 k0sub.c –c054 –common
```

(16) セルフ・プログラミング指定

セルフ・プログラミング指定 （-SELF）

【記述形式】

-SELF

- **省略時解釈**

  なし

【機能】

- 8100H **番地がアクセス範囲外（内蔵** ROM **が存在しない）の場合でも，“** CALL !8100H **”の記述に対してエラーを出力しません。**

【用途】

- -SELF **オプションは，セルフ・プログラミングを使用する際に指定します。**

【説明】

- **セルフ・プログラミング使用時に，“** CALL !8100H **”の記述に対してエラーとなる場合に指定してください。**

(17) ヘルプ指定

ヘルプ指定 (--)

【記述形式】

```
--
```

- 省略時解釈

  表示しない

【機能】

- -- オプションは，ヘルプ・メッセージをディスプレイに出力します。

【用途】

- ヘルプ・メッセージは，アセンブル・オプションとその説明の一覧です。アセンブラを実行するときに参照してください。

【説明】

- -- オプションを指定すると他のアセンブラ・オプションは，すべて無効となります。
- ヘルプ・メッセージの続きを読む場合は，リターン・キーを入力してください。
  表示を途中で終了する場合は，リターン・キー以外の文字を入力したあとで，リターン・キーを入力してください。

  注意　このオプションは，PM plus 上では指定できません。
  PM plus 上でヘルプを参照する場合は，〈アセンブラオプションの設定〉ダイアログで［ヘルプ］ボタンをクリックしてください。

【使用例】

- -- **オプションを指定すると，ヘルプ・メッセージがディスプレイに出力されます。**

C>ra78k0 --

```
78K/0 Series Assembler Vx.xx [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

usage : ra78k0 [ option [ ... ] ] input-file [ option [ .. .] ]
The option is as follows ( [ ] means omissible ).
-cx                : Select target chip. ( x = 012 , 014 etc. ) * Must be specified.
-o [ file ] / -no  : Create the object module file [ with the specified name ] / Not.
-e [ file ] / -ne  : Create the error list file [ with the specified name ] / Not.
-p [ file ] / -np  : Create the print file [ with the specified name ] / Not.
-ka / -nka         : Output the assemble list to print file / Not.
-ks / -nks         : Output the symbol table list to print file / Not.
-kx / -nkx         : Output the cross reference list to print file / Not.
-lw [ width ]      : Specify print file columns per line.
-ll [ length ]     : Specify print file lines per page.
-lf / -nlf         : Add Form Feed at end of print file / Not.
-lt [ n ]          : Expand TAB character for print file ( n = 1 to 8 ) / Not expand ( n = 0 ).
-lhstring          : Print list header with the specified string.
-g / -ng           : Output debug information to object file / Not.
-j / -nj           : Create object file if fatal error occurred / Not.
-idirectory [ , directory .. ]  : Set include search path.
-tdirectory        : Set temporary directory.
-ydirectory        : Set device file search path.
-ffile             : Input option or source module file name from specified file.
-ga / -nga         : Output assembler source debug information to object file / Not.
-dname [ = data ] [ , name [ = data ] [ ... ] ] : Define name [ with data ].
-common            : Create the common object module file for 78K0 Series.
-self              : Use Self-programming.
-zs / -ze / -zn    : Change source regulation.
                        -zs : SJIS code usable in comment.
                        -ze : EUC code usable in comment.
                        -zn : no multibyte code in comment.
--                 : Show this message.
DEFAULT ASSIGNMENT :
          -o -ne -p -ka -nks -nkx -lw132 -ll66 -nlf -lt8 -g -nj -ga
```

5.5 PM plus でのオプション設定

PM plus からアセンブラ・オプションを設定する方法について説明します。

5.5.1 オプションの設定方法

PM plus の [ツール (T)] メニューの [アセンブラオプションの設定 (A)] を選択するか，ツール・バーの［RA］ボタンを押下すると，〈アセンブラオプションの設定〉ダイアログが現れます。

ダイアログ内で必要なオプションを入力することにより，各アセンブラ・オプションを設定できます。

図 5-2 〈アセンブラオプションの設定〉ダイアログ（《出力 1》タブ選択時）

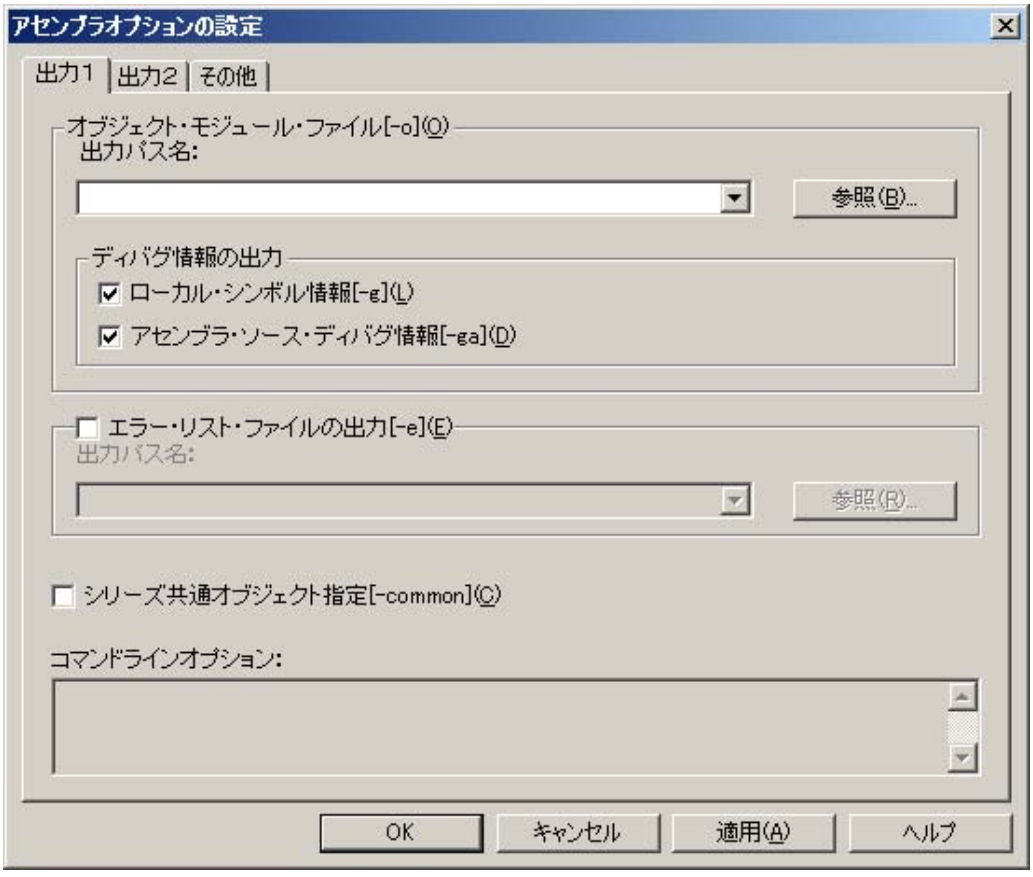


図 5-3 〈アセンブラオプションの設定〉ダイアログ（《出力 2》タブ選択時）

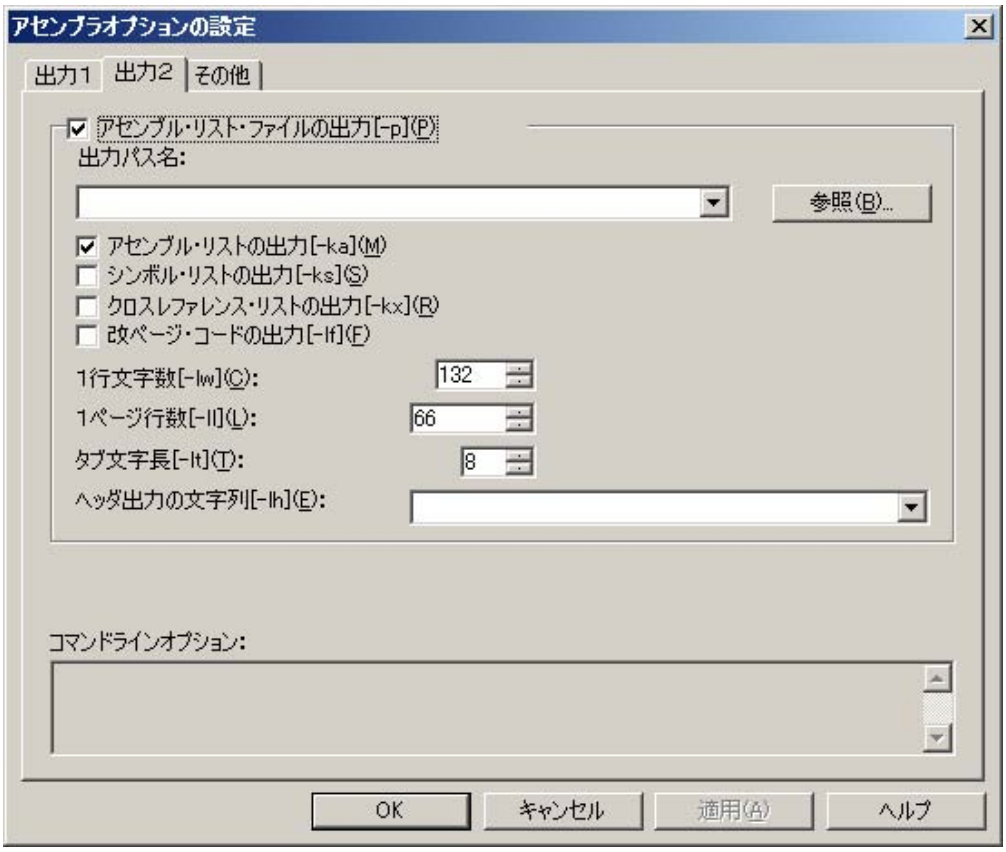
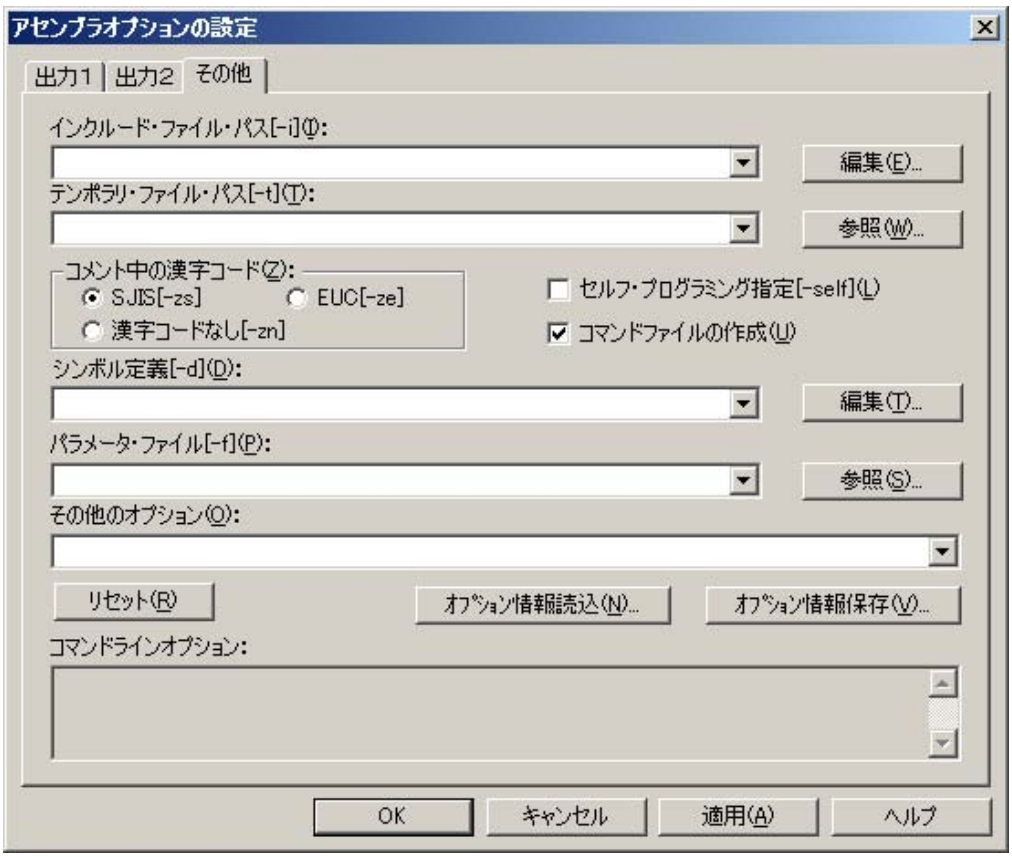


図 5-4 〈アセンブラオプションの設定〉ダイアログ（《その他》タブ選択時）



5.5.2 各オプションの設定

〈アセンブラオプションの設定〉ダイアログの各オプションについて，次に説明します。

《出力 1》タブ

- **オブジェクト・モジュール・ファイル** [-o](O)

  （全体オプションで指定する場合）出力パス名：

  ［参照 (B)］ボタン，または直接入力により，オブジェクト・モジュール・ファイルのパスを指定します。

  （個別オプションで指定する場合）出力ファイル名：

  ［参照 (B)］ボタン，または直接入力により，オブジェクト・モジュール・ファイルのパスとファイル名を指定します。

- **ローカル・シンボル情報** [-g](L)

  オブジェクト・モジュール・ファイル中に，ディバグ情報（ローカル・シンボル情報）を付加する場合に，チェックします。

- **アセンブラ・ソース・ディバグ情報** [-ga](D)

  オブジェクト・モジュール・ファイル中に，ソース・ディバグ情報を付加する場合に，チェックします。

- **エラー・リスト・ファイルの出力** [-e](E)

  エラー・リスト・ファイルを出力する場合に，チェックします。

  （全体オプションで指定する場合）出力パス名：

  ［参照 (R)］ボタン，または直接入力により，エラー・リスト・ファイルのパスを指定します。

  （個別オプションで指定する場合）出力ファイル名：

  ［参照 (R)］ボタン，または直接入力により，エラー・リスト・ファイルのパスとファイル名を指定します。

- **シリーズ共通オブジェクト指定** [-common](C)

  78K0 **シリーズ共通オブジェクト・モジュール・ファイルを出力する場合に，チェックします。**

- **コマンドラインオプション**

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

《出力 2》タブ

- **アセンブル・リスト・ファイルの出力** [-p](P)

  アセンブル・リスト・ファイルを出力する場合に，チェックします。

  （全体オプションで指定する場合）出力パス名：

  ［参照 (B)］ボタン，または直接入力により，アセンブル・リスト・ファイルのパスを指定します。

  （個別オプションで指定する場合）出力ファイル名：

  ［参照 (B)］ボタン，または直接入力により，アセンブル・リスト・ファイルのパスとファイル名を指定します。

- **アセンブル・リストの出力** [-ka](M)

  アセンブル・リスト・ファイル中にアセンブル・リストを出力する場合に，チェックします。

- **シンボル・リストの出力** [-ks](S)

  アセンブル・リストに続いてシンボル・リストをアセンブル・リスト・ファイル中に出力する場合に，チェック

します。

- **クロスレファレンス・リストの出力** [-kx](R)
  アセンブル・リストに続いてクロスレファレンス・リストをアセンブル・リスト・ファイル中に出力する場合に，チェックします。
- **改ページ・コードの出力** [-lf](F)
  アセンブル・リスト・ファイルの最後に改頁コード (FF) **を付加する場合に，チェックします。**
- 1 **行文字数** [-lw](C)
  アセンブル・リスト・ファイルの１行の文字数を指定します。指定可能な文字数は，72 **〜** 2046 **の範囲です。**
- 1 **ページ行数** [-ll](L)
  アセンブル・リスト・ファイルの１ページの行数を指定します。指定可能な行数は，20 **〜** 32767 **の範囲です。**
- **タブ文字長** [-lt](T)
  タブ文字長を指定します。指定可能なタブ文字長は，0 **〜** 8 **までの範囲です。**
- **ヘッダ出力の文字列** [-lh](E)
  アセンブル・リスト・ファイルのヘッダのタイトル欄に印字する文字列を指定します。入力文字数は 60 **文字までです。**
- **コマンドラインオプション**
  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

《その他》タブ

- **インクルード・ファイル・パス** [-i](I)
  ［編集 (E)**］ボタン，または直接入力によりインクルード・ファイルを読み込むパスを指定します。**
- **テンポラリ・ファイル・パス** [-t](T)
  ［参照 (W)**］ボタン，または直接入力によりテンポラリ・ファイルの作成するパスを指定します。**
- **コメント中の漢字コード** (Z)
  ソースのコメント中で使用する漢字コードの種類 (SJIS[-zs]**，**EUC[-ze]**，漢字コードなし** [-zn]) **を選択します。**
- **セルフ・プログラミング指定** [-self](L)
  セルフ・プログラミングを使用する場合にチェックしてください。
- **コマンドファイルの作成** (U)
  コマンドファイルを作成する場合に，チェックします。
- **シンボル定義** [-d](D)
  ［編集 (T)**］ボタン，または直接入力により，シンボルに定義づける数値を入力します。**
- **パラメータ・ファイル** [-f](P)
  ［参照 (S)**］ボタン，または直接入力により，ユーザ定義のパラメータ・ファイルとして入力するファイルを指定します。**

- **その他のオプション** (O)

  ダイアログで設定可能なオプション以外のオプションを指定したい場合に入力ボックスに入力します。

  注意　ヘルプ指定 (--) **のオプションは，**PM plus **上では指定できません。**

- **リセット** (R)

  入力した内容をリセットします。

- **オプション情報読込** (N)

  〈オプション情報読込み〉ダイアログが開き，オプション情報ファイルを指定後，読み込みます。

- **オプション情報保存** (V)

  〈オプション情報の保存〉ダイアログが開き，オプション情報ファイルに名前をつけて保存します。

- **コマンドラインオプション**

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

5.5.3 オプションの編集ダイアログ

〈オプションの編集〉ダイアログは，項目をリストで編集します。

〈オプションの編集〉ダイアログについて，次に説明します。

図 5-5 〈オプションの編集〉ダイアログ

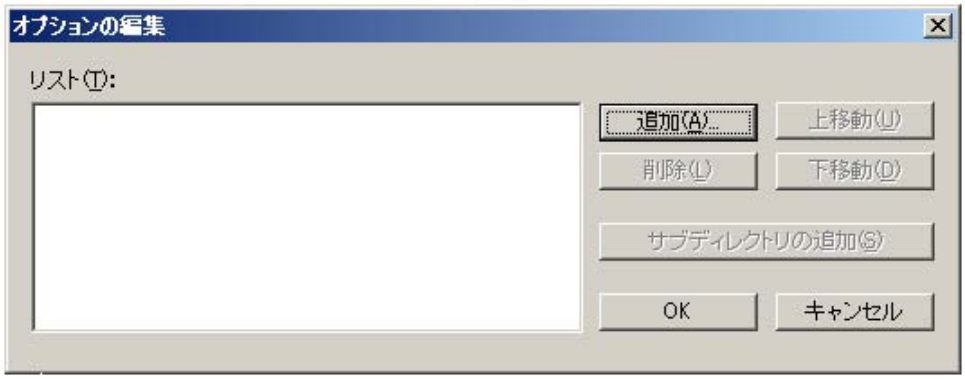
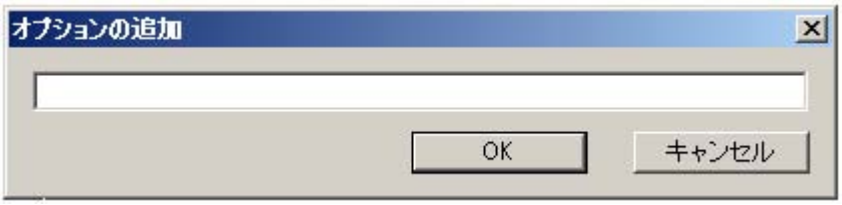


図 5-6 〈オプションの追加〉ダイアログ



- ［追加 (A)］ボタン

  リストの項目を追加します。

  ファイルやディレクトリを指定する項目の場合は，それぞれの〈参照〉ダイアログがオープンします。

  それ以外の場合は内容を入力する〈オプションの追加〉ダイアログがオープンします。

- ［削除 (L)］ボタン

  選択中のリストの項目を削除します。

- ［上移動 (U)］ボタン

  選択中のリストの項目を上に移動します。

- ［下移動 (D)］ボタン

  選択中のリストの項目を下に移動します。

- ［サブディレクトリの追加 (S)］ボタン

  次の場合は，選択中のリストの項目にサブディレクトリを追加できます。

  《その他》タブのインクルード・ファイル・パス [-i](I)

第 6 章　リンカ

この章では，リンカは，78K0 用のアセンブラが出力したいくつかのオブジェクト・モジュール・ファイルを入力し，配置アドレスを決定して 1 つにまとめたロード・モジュール・ファイルを出力します。

さらに，リンク・リスト・ファイルやエラー・リスト・ファイルなどのリスト・ファイルを出力します。

リンク・エラーがある場合は，エラー・メッセージをエラー・リスト・ファイルに出力し，エラーの原因を明示します。なお，エラーがある場合，ロード・モジュール・ファイルを出力しません。

6.1　リンカの入出力ファイル

リンカの入出力ファイルを次に示します。

表 6-1 リンカの入出力ファイル

種類	ファイル名	説明	ディフォールト・ファイル・タイプ
入力ファイル	オブジェクト・モジュール・ファイル	- 機械語情報と機械語の配置アドレスに関する再配置情報，およびシンボル情報を含んだバイナリ・ファイルです。 - アセンブラの出力したファイルです。	.rel
	ライブラリ・ファイル	- 複数のオブジェクト・モジュール・ファイルが登録されたファイルです。 - ライブラリアンの出力したファイルです。	.lib
	ディレクティブ・ファイル	- リンクに対するリンク指示を記述したファイルです。 - ユーザ作成ファイルです。	.dr
	パラメータ・ファイル	- 実行プログラムのパラメータを内容とするファイルです。 - ユーザ作成ファイルです。	.plk
出力ファイル	ロード・モジュール・ファイル	- リンク結果の全情報を持つバイナリ・イメージのファイルです。オブジェクト・コンバータの入力ファイルとなります。	.lmf
	リンク・リスト・ファイル	- リンク結果を表示するリスト・ファイルです。	.map
	エラー・リスト・ファイル	- リンク時のエラー情報を持つファイルです。	.elk
入出力ファイル	テンポラリ・ファイル	- リンクのためにリンカが自動生成するファイルです。アセンブル終了時には消去されます。	LKxxxxx.$$n （n = 1-3）

6.2 リンカの機能

リンカの機能を次に示します。

(1) 入力セグメントの結合

リンカは，セグメントごとに配置するアドレスを決定し，管理します。

別々のオブジェクト・モジュール・ファイルにあっても，セグメントが同じであれば，そのセグメントを 1 つとみなして結合します。

(2) 入力モジュールの決定

入力としてライブラリ・ファイルが指定されていると，入力オブジェクト・モジュール・ファイルが参照しているモジュールをライブラリから探し出して，入力モジュールとして扱います。

(3) 入力セグメントの配置アドレス決定

入力モジュールの配置アドレスをセグメントごとに決定します。ソース・モジュール・ファイル中で配置属性が指定されていれば，その属性に従って配置します。また，リンカの「リンク・ディレクティブ・ファイル」で配置属性を指定できます。

(4) オブジェクト・コードの修正

配置アドレスがオブジェクト・コードに埋め込まれるものは，(3) で決定した配置アドレスに従ってオブジェクト・コードを修正します。

6.3 メモリ空間とメモリ領域

メモリ空間とは，メモリ領域を定義するための空間です。また，メモリ領域とはセグメントを配置するためにメモリ空間中に定義した領域です。

メモリ空間 ： 64 K バイトごと

メモリ領域 ： 1 つのメモリ空間をさらにいくつかの領域に分割する。
実装しているメモリのアドレスを宣言する。

表 6-2 セグメントの配置のグループ分け（外付け ROM など）

メモリ領域名	ディフォールトのアドレス	ディフォールトで配置されるセグメント
ROM	内蔵 ROM：ROM レスは RAM の前まで	CSEG
RAM	内蔵 RAM	DSEG , BSEG

備考 1 メモリ領域のディフォールトのアドレスを変更したい場合やプログラムで記述した各セグメントの配置を指定したい場合に，ディレクティブ・ファイルを使用します。

備考 2 具体例については，「3.4 (5) ディレクティブ・ファイルを作成します。」を参照してください。

6.4 リンク・ディレクティブ

リンク・ディレクティブ（以降ディレクティブとします）とは，リンカに対して入力ファイルや使用可能なメモリ領域，セグメントの配置など，リンク時の各種指示を行うための命令群です。

ディレクティブ・ファイルの役割

(1) 実装メモリのアドレスを宣言

(2) メモリをいくつかの領域に分割

例 CALLT 領域

内蔵 ROM

外付け ROM

SADDR

SADDR 以外の内蔵 RAM

(3) セグメントの配置をリンカで指定する

各セグメントに対し，次の内容を指定します。

- アブソリュート・アドレス
- メモリ領域のみ指定

エディタなどでディレクティブ・ファイル（ディレクティブを記述したファイル）を作成し，リンカの起動時に，-D オプションを指定して作成したファイルを読み込みます。

リンカは，ファイルからディレクティブを読み込み，解釈しながらリンク処理を行います。

ディレクティブには，次の 2 種類があります。

表 6-3 ディレクティブの種類

ディレクティブの種類	説明
メモリ・ディレクティブ	- 実装メモリのアドレスを宣言します。 - メモリをいくつかの領域に分割して，メモリ領域を指定します。
セグメント配置ディレクティブ	- セグメントの配置を指定します。

6.4.1 ディレクティブ・ファイル

ディレクティブ・ファイル中に記述するディレクティブの記述フォーマットを次に示します。

ディレクティブは，1 つのディレクティブ・ファイル中に複数記述できます。

- メモリ・ディレクティブ

  MEMORY　メモリ領域名 : (スタート・アドレス値 , サイズ) [/ メモリ空間名]

- セグメント配置ディレクティブ

  MERGE　セグメント名 : [AT △ (△スタート・アドレス△)] [= メモリ領域名指定] [/ メモリ空間名]

(1) 予 約 語

ディレクティブ・ファイル中での予約語を次に示します。

MEMORY , MERGE , AT , SEQUENT , COMPLETE

ディレクティブ・ファイル中で予約語を他の意味（セグメント名やメモリ領域名など）に使用することはできません。
予約語の記述は，大文字でも小文字でもかまいません。ただし，大文字と小文字を混在させた記述はできません。

例　MEMORY
　　memory
　　Memory　：　使用不可

ソース中に同名のセグメントが複数あった場合，結合させずにエラーとしたいときは“ COMPLETE ”，結合したいときは“ SEQUENT（ディフォールト）”をディレクティブ中に指定します。

SEQUENT　：セグメントを出現順に，順次空きを作らないようにマージします。
　　BSEG はビット単位で出現順にマージします。

COMPLETE　：同名のセグメントが複数存在する場合はエラーとします。

　　例　MERGE DSEG1 : COMPLETE = RAM

(2) シンボル

セグメント名，メモリ領域名，メモリ空間名の記述では大文字と小文字を区別します。

(3) 数　値

各ディレクティブの項目のうち数値定数を記述する場合は，10 進数または 16 進数を記述できます。
記述方法はソース・プログラムと同じで，16 進数の場合は最後に“ H ”を付けます。また，先頭が A-F の場合は前に“ 0 ”を付けます。

例　23H , 0FC80H

(4) コメント文

ディレクティブ・ファイル中に，“ ; ”または“ # ”を記述した場合，そこから改行文字（LF）までは，コメントとして扱われます。なお，改行文字が現れる前にディレクティブ・ファイルが終了した場合は，終了までをコメントとして扱います。

例　下線部がコメントとなります。

```
; DIRECTIVE FILE FOR 78054
MEMORY MEM1 : ( 01000H , 1000H ) #SECOND MEMORY AREA
```

6.4.2 メモリ・ディレクティブ

メモリ・ディレクティブは，メモリ領域（実装するメモリのアドレスと名前）を定義するディレクティブです。
定義したメモリ領域は，その名前（メモリ領域名）によってセグメント配置ディレクティブで参照できます。
メモリ領域は，ディフォールトで定義されているメモリ領域を含め 100 個まで定義できます。

【構文】

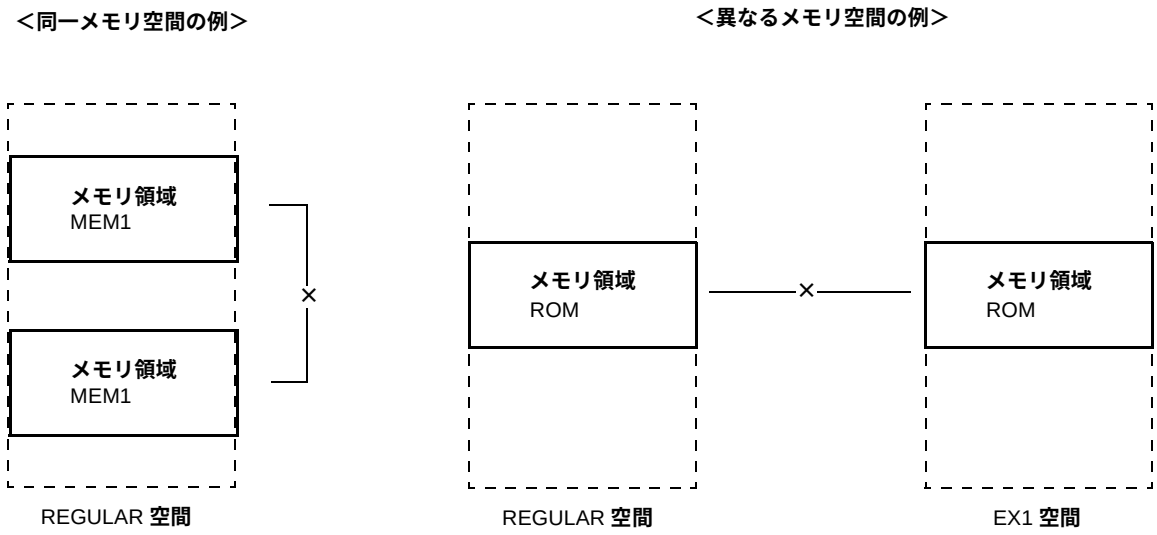
MEMORY △**メモリ領域名**△ : △ (△**スタート・アドレス**△ , △**サイズ**△) [/ △**メモリ空間名**]

(1) **メモリ領域名**

定義するメモリ領域の名前を指定します。指定時の条件は次のとおりです。

- **メモリ領域名に使用できる文字は，**A-Z , a-z , 0-9 , _ , ? , @ **です。ただし，**0-9 **はメモリ領域名の先頭には使用できません。**
- **大文字と小文字は別の文字として区別します。**
- **大文字と小文字は混在できます。**
- **メモリ領域名の長さは，最大** 31 **文字です。**32 **文字以上記述するとエラーとなります。**
- **各メモリ領域名は，全メモリ空間を通じて** 1 **つでなくてはなりません。異なるメモリ領域に同じメモリ領域名を付けることは，メモリ空間が同一である場合でも，異なる場合でも許されません。**

図 6-1 **メモリ領域名**



(2) **スタート・アドレス**

定義するメモリ領域の先頭アドレスを指定します。

0H-FFFFFH **までの数値定数を記述します。**

(3) **サ イ ズ**

定義するメモリ領域のサイズを指定します。指定時の条件は次のとおりです。

- 1 **以上の数値定数を記述します。**

- リンカがディフォールトで定義しているメモリ領域のサイズを指定し直す場合には，定義可能な範囲の制約があります。

各デバイスのディフォールトで定義されているメモリ領域のサイズと再定義可能な範囲は，各デバイス・ファイルの「使用上の留意点」を参照してください。

(4) メモリ空間名

メモリ空間名は，メモリ空間を 64 K バイトごとに分けた次の 16 個の名前で表されます。

REGULAR , EX1, EX2 , EX3 , EX4 , EX5 , EX6 , EX7 , EX8 , EX9 , EX10 , EX11 , EX12 , EX13 , EX14 , EX15

メモリ空間名は，メモリ領域をどのメモリ空間に割り付けるかを指定するときに使います。次に指定時の条件を示します。

- メモリ空間名は，すべて大文字で記述します。
- メモリ空間名を省略した場合，REGULAR を指定したものとみなされます。
- “ / ” を記述したあとにメモリ空間名を省略した場合は，エラーとなります。

【機能】

- メモリ領域名で指定した名前を持つメモリ領域を，指定したメモリ空間に定義します。
- 1 つのメモリ・ディレクティブで 1 つのメモリ領域を定義できます。
- メモリ・ディレクティブ自体は，複数の記述が可能です。このとき，指定した順番に複数回定義された場合は，エラーとなります。
- ディフォールトのメモリ領域は，メモリ・ディレクティブで同一のメモリ領域を再定義しないかぎり有効です。メモリ・ディレクティブの記述を省略した場合，リンカが持つ各デバイスごとのディフォールトのメモリ領域のみを指定したものとします。
- ディフォールトのメモリ空間を使用せずに，別の領域名で使用するときは，ディフォールトの領域名のサイズを “ 0 ” に設定してください。

【使用例】

- メモリ空間（EX1）のアドレス 0H から 1FFH までをメモリ領域 ROMA として定義します。

```
MEMORY ROMA : ( 0H , 200H ) / EX1
```

6.4.3 セグメント配置ディレクティブ

セグメント配置ディレクティブは，指定したセグメントを指定したメモリ領域上か，特定番地に配置するディレクティブです。

【構文】

MERGE△**セグメント名**△:△[AT△(△**スタート・アドレス**△)] [△=△**メモリ領域名**] [△/△**メモリ空間名**]

(1) **セグメント名**

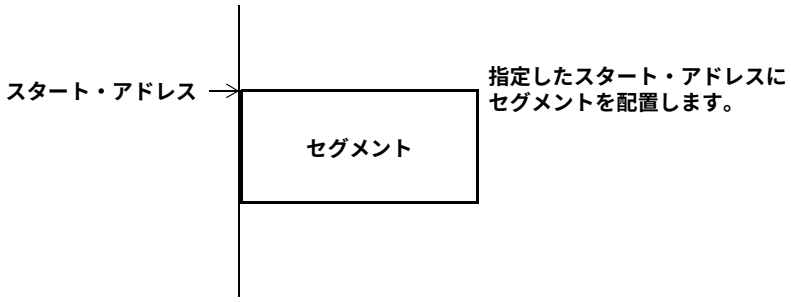
リンカに入力するオブジェクト・モジュール・ファイル中に含まれるセグメント名です。

- **セグメント名として入力セグメント以外は指定できません。**
- **セグメント名は，アセンブル・ソース上に記述したとおりに指定しなければなりません。**

(2) **スタート・アドレス**

セグメントを“スタート・アドレス”で指定した領域に配置します。

- **予約語** AT **は，大文字または小文字のいずれか一方で記述しなければなりません。**
  なお，大文字と小文字を混同してはなりません。
- **スタート・アドレスには，数値定数を記述します。**



注意 1　**指定したスタート・アドレスによって配置を行うと，セグメントが配置されるメモリ領域の範囲を越えてしまう場合はエラーとなります。**

注意 2　**セグメント疑似命令の** AT **指定，または** ORG **疑似命令によって配置アドレスを指定したセグメントに対して，リンク・ディレクティブでスタート・アドレスは指定できません。**

(3) **メモリ空間名**

メモリ空間名は，セグメントを配置するメモリ空間を指定します。

- **メモリ空間名として指定できるのは，次の** 16 **種類のうちのいずれかです。**
- REGULAR , EX1 , EX2 , EX3 , EX4 , EX5 , EX6 , EX7 , EX8 , EX9 , EX10 , EX11 , EX12 , EX13 , EX14 , EX15
- **メモリ空間名は，すべて大文字で記述します。**
- **メモリ空間名を省略した場合，**REGULAR **を指定したものとみなします。**

次にセグメントの配置先を示します。

表 6-4 メモリ領域名指定とメモリ空間の組み合わせによるセグメントの配置

メモリ領域名指定	メモリ空間	セグメントの配置先
指定なし	指定なし	REGULAR 空間中のディフォールト状態のとき配置されるメモリ領域
指定なし	メモリ空間名	指定されたメモリ空間中の任意のメモリ領域
メモリ領域名	指定なし	REGULAR 空間の指定されたメモリ領域
メモリ領域名	メモリ空間名	指定されたメモリ空間の指定されたメモリ領域

この表では，セグメントの配置の対象となるメモリ領域を定義するということを中心として説明しています。なお，実際の配置アドレス決定時には，“ AT（スタート・アドレス）” が指定されていれば，そのアドレスからセグメントを配置します。

たとえば，再配置属性が “ CSEG FIXED ” であるセグメントに，メモリ名 “ EX1 ” が指定された場合，セグメントが 800H-FFFH の中に納まるように配置します。

【注意】

- **セグメント配置ディレクティブが指定されなかった入力セグメントは，アセンブル時にセグメント定義疑似命令で指定した再配置属性に従って配置アドレスを決定します。**
- **セグメント名として指定したセグメントが存在しない場合はエラーです。**
- **同一のセグメントに対して，セグメント配置ディレクティブを複数回指定した場合エラーとなります。**

【使用例】

- **セグメント・タイプ，再配置属性が“** CSEG UNIT **”であるセグメント** SEG1 **に対して，アドレスを割り付けます。領域は次のように宣言してあるものとします。**

```
MEMORY ROM : ( 0000H , 1000H )
MEMORY MEM1 : ( 1000H , 2000H )
MEMORY RAM : ( 0FE00H , 200H )
```

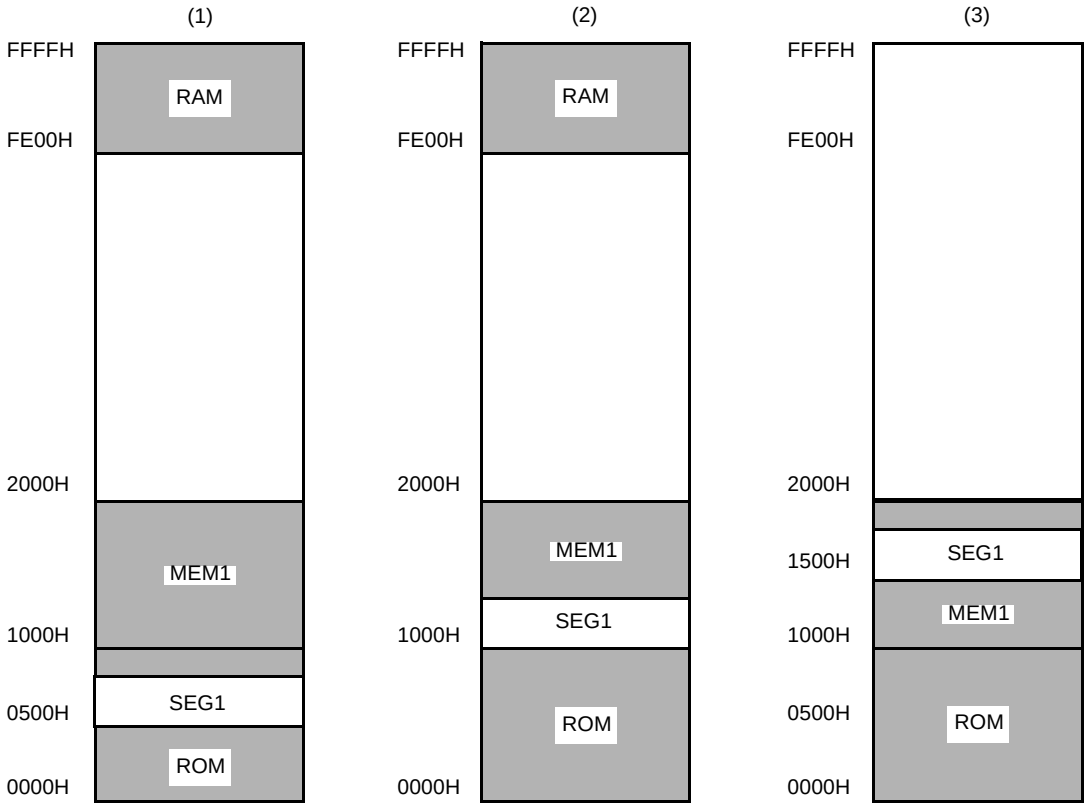
  - **入力セグメント** SEG1 **を** ROM **領域中の** 500H **に割り付ける場合（**図 6-2 (1) **参照）**

    MERGE　SEG1 : AT (500H)
  - **入力セグメント** SEG1 **をメモリ領域** MEM1 **中に割り付ける場合（**図 6-2 (2) **参照）**

    MERGE　SEG1 : = MEM1
  - **入力セグメント** SEG1 **をメモリ領域** MEM1 **中の** 1500H **に割り付ける場合（**図 6-2 (3) **参照）**

    MERGE　SEG1 : AT (1500H) = MEM1

図 6-2 **セグメント配置の具体例**



6.5 リンカの起動方法

6.5.1 リンカの起動

リンカの起動には，次の 2 **つの方法があります。**

(1) **コマンド行での起動**

```
X>[ パス名 ] lk78k0 [ △オプション ] …オブジェクト・モジュール・ファイル名 [ △オプション ] … [ △ ]
↑      ↑       ↑          ↑                    ↑                          ↑
(a)    (b)     (c)        (d)                  (e)                        (d)
```

(a) **カレント・ドライブ名**

(b) **カレント・ディレクトリ名**

(c) **リンカのコマンド・ファイル名**

(d) **リンカに対して動作の詳細を指示します。**

複数のリンカ・オプションを指定する場合は，それぞれのリンカ・オプション間を空白で区切ってください。

空白を含むパスを設定する場合には，ダブルクオーテーション（“ ”）で囲んでください。

(e) **リンカに対して動作の詳細を指示します。**

入力モジュールとして，最大 1024 **個入力できます。**

空白を含むパスのファイル名を指定する場合には，ダブルクオーテーション（“ ”）で囲んでください。

例 C>lk78k0 k0main.rel k0sub.rel –ok0.lmf -g

(2) **パラメータ・ファイルによる起動**

パラメータ・ファイルは，起動に必要な情報がコマンド行に指定しきれない場合やリンクするたびに同じオプションを繰り返し指定するような場合に使用します。

パラメータ・ファイルを使用する場合には，コマンド行にパラメータ・ファイル指定オプション（-F**）を指定します。**

パラメータ・ファイルによる起動方法は，次のようになります。

```
X>LK78K0 [ △オブジェクト・モジュール・ファイル ] △ -f パラメータ・ファイル名
                                                  ↑          ↑
                                                 (a)        (b)
```

(a) **パラメータ・ファイル指定オプション**

(b) **リンカの起動に必要な情報を含んだファイル**

備考　パラメータ・ファイルは，エディタなどで作成します。

パラメータ・ファイル内での記述規則を次に示します。

```
[ [ [ △ ] オプション [ △オプション ] … [ △ ] △ ] ] …
```

- コマンド行でオブジェクト・モジュール・ファイル名を省略した場合は，パラメータ・ファイル内でオブジェクト・モジュール・ファイル名を指定します。
- オブジェクト・モジュール・ファイル名は，オプションの後に記述することも可能です。
- パラメータ・ファイルには，コマンド行で指定すべきすべてのリンク・オプション，出力ファイル名を記述します。

例　パラメータ・ファイル（k0.plk）をエディタで作成します。

＜ k0.plk の内容＞

```
; parameter file
k0main.rel k0sub.rel –ok0.lmf –pk0.map -e
-tc:\tmp
```

パラメータ・ファイル k0.plk を使用してリンカを起動します。

```
C>lk78k0 –fk0.plk
```

6.5.2　実行開始メッセージ，終了メッセージ

(1) 実行開始メッセージ

リンカが起動すると，次の実行開始メッセージが表示されます。

```
78K/0 Series Linker Vx.xx [ xx xxx xx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx
```

(2) 実行終了メッセージ

リンクの結果，リンク・エラーが検出されなかった場合，リンカは次のメッセージを表示して制御を OS に戻します。

```
Target chip : uPD78xxx
 Device file : Vx.xx

Link complete ,       0 error ( s ) and       0 warning ( s ) found.
```

リンクの結果，リンク・エラーが検出された場合，リンカはエラーの数を表示して制御を OS に戻します。

```
Target chip : uPD78xxx
 Device file : Vx.xx

Link complete ,       1 error ( s ) and       0 warning ( s ) found.
```

リンク中にリンカの処理継続が不可能な致命的エラーが検出された場合，リンカはメッセージを表示してリンクを中止し，制御を OS に戻します。

例 1　存在しないオブジェクト・モジュール・ファイルを指定した場合

```
C>lk78k0 samp1.rel samp2.rel
```

```
78K/0 Series Linker Vx.xx [ xx xxx xx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

RA78K0 error F3006 : File not found ' SAMP1.REL '
RA78K0 error F3006 : File not found ' SAMP2.REL '
Program Aborted.
```

この例では，存在しないオブジェクト・モジュール・ファイルを指定したためにエラーとなり，リンクが中止されました。

例 2　存在しないリンカ・オプションを指定した場合

```
C>lk78k0 k0main.rel k0sub.rel -z
```

```
78K/0 Series Linker Vx.xx [ xx xxx xx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

RA78K0 error F3018 : Option is not recognized ' -z '
Please enter ' LK78K0 -- ' , if you want help messages.
Program Aborted.
```

この例では，存在しないリンカ・オプションを指定したためにエラーとなり，リンクが中止されました。

リンカがエラー・メッセージを出力してリンクを中止した場合，そのエラー・メッセージの原因を「第 12 章 エラー・メッセージ」で調べて対処してください。

6.6 リンカ・オプション

6.6.1 リンカ・オプションの種類

リンカ・オプションはリンカの動作に細かい指示を与えるものです。リンカ・オプションは，19 種類のオプションに分類できます。

表 6-5 リンカ・オプション

分類	オプション	説明
ロード・モジュール・ファイル出力指定	-O	ロード・モジュール・ファイルの出力を指定します。
	-NO	
ロード・モジュール・ファイル強制出力指定	-J	強制的にロード・モジュール・ファイルの出力をします。
	-NJ	
ディバグ情報出力指定	-G	ディバグ情報をロード・モジュール・ファイルへ出力します。
	-NG	
スタック解決用シンボル生成指定	-S	スタック解決用のパブリック・シンボルを自動生成します。
	-NS	
ディレクティブ・ファイル指定	-D	指定のファイルをディレクティブ・ファイルとして入力します。
リンク・リスト・ファイル出力指定	-P	リンク・リスト・ファイルの出力を指定します。
	-NP	
リンク・リスト・ファイル情報指定	-KM	リンク・リスト・ファイル中に，マップ・リストを出力します。
	-NKM	
	-KD	リンク・リスト・ファイル中に，リンク・ディレクティブ・ファイルを出力します。
	-NKD	
	-KP	リンク・リスト・ファイル中に，パブリック・シンボル・リストを出力します。
	-NKP	
	-KL	リンク・リスト・ファイル中に，ローカル・シンボル・リストを出力します。
	-NKL	
リンク・リスト・ファイル形式指定	-LL	リストの 1 頁に印字する行数を変更します。
	-LF	リスト・ファイルの最後に改頁コードを付加します。
	-NLF	
エラー・リスト・ファイル出力指定	-E	エラー・リスト・ファイルを出力します。
	-NE	
ライブラリ・ファイル指定	-B	指定のファイルをライブラリ・ファイルとして入力します。
ライブラリ・ファイル読み込みパス指定	-I	ライブラリ・ファイルを指定したパスから読み込みます。

表 6-5 リンカ・オプション

分類	オプション	説明
パラメータ・ファイル指定	-F	入力ファイル名，オプションを指定したファイルより入力します。
テンポラリ・ファイル作成パス指定	-T	テンポラリ・ファイルを指定したパスに作成します。
デバイス・ファイル・サーチ・パス指定	-Y	デバイス・ファイルを指定されたパスから読み込みます。
ワーニング・メッセージ出力指定	-W	ワーニング・メッセージをコンソールへ出力するか否かを指定します。
フラッシュ ROM 内蔵製品のブート領域 ROM プログラムのリンク指定	-ZB	フラッシュ ROM 領域の先頭アドレスを指定します。
オンチップ・ディバグのプログラム・サイズ指定	-GO	オンチップ・ディバグのプログラム・サイズを指定します。
セキュリティ ID 指定	-GI	セキュリティ ID を指定します。
ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

6.6.2 リンカ・オプションの優先度

次の表に示すリンカ・オプションのうち，縦軸のものと横軸のものを同時に 2 つ以上指定した場合の優先度について説明します。

表 6-6 リンカ・オプションの優先度

	-NO	-NG	-NP	-NKM	-NKP	-NKL	--
-J	×						×
-G	×						×
-P				△	△	△	×
-KM			×				×
-KD			×	×			×
-KP		×	×				×
-KL		×	×				×
-LL			×				×
-LF			×				×

【×で記した箇所】

横軸に示したオプションを指定した場合，縦軸に示したオプションは無効となります。

例 C>lk78k0 k0main.rel k0sub.rel -np -km

-KM オプションは，無効となります。

【△で記した箇所】

横軸に示したオプション 3 つをすべて指定した場合，縦軸に示したオプションは無効となります。

例 C>lk78k0 k0main.rel k0sub.rel -p -nkm -nkp -nkl

-NKM , -NKP，および -NKL が同時に指定されたので，-P オプションは無効となります。

また，-O/-NO オプションのようにオプション名の前に “ N ” を付加できるオプションを同時に指定した場合，あとで指定した方が有効となります。

例 C>lk78k0 k0main.rel k0sub.rel -o -no

-NO オプションがあとに指定されているので，-O オプションは無効となり，-NO オプションが有効となります。

表 6-6 に記述されていないオプションは，他のオプションの影響を特に受けません。しかし，ヘルプ指定オプション（ -- ）が指定された場合には，すべてのオプション指定が無効となります。

6.6.3 リンカ・オプションの説明

次ページ以降に，各リンカ・オプションの詳細について説明します。

(1) ロード・モジュール・ファイル出力指定

ロード・モジュール・ファイル出力指定 （-O/-NO）

【記述形式】

-O [出力ファイル名] -NO

- 省略時解釈

  -O 入力ファイル名 .lmf

【機能】

- -O オプションは，ロード・モジュール・ファイルの出力を指定します。
  また，その出力先や出力ファイル名を指定します。
- -NO オプションは，-O，-J，-G オプションを無効にします。

【用途】

- ロード･モジュール･ファイルの出力先や出力ファイル名を変更したいときに -O オプションを指定します。
- リンク・リスト・ファイルの出力のみが目的でリンクする場合などに，-NO オプションを指定します。この指定によってリンク時間が短縮されます。

【説明】

- 出力ファイルとしてディスク型ファイル名とデバイス型ファイル名の NUL , AUX を指定できます。
- -O オプションを指定してもフェータル・エラーがある場合は，ロード・モジュール・ファイルは出力されません。
- -O オプションを指定する際に “ 出力ファイル名 ” を省略すると，カレント・ディレクトリにロード・モジュール・ファイル “ 入力ファイル名 .lmf ” が出力されます。
- “ 出力ファイル名 ” にパス名のみを指定すると，指定したパスに “ 入力ファイル名 .lmf ” が出力されます。
- -O と -NO の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

【使用例】

- ロード・モジュール・ファイル k0.lmf を出力します。

```
C>lk78k0 k0main.rel k0sub.rel –ok0.lmf
```

(2) ロード・モジュール・ファイル強制出力指定

ロード・モジュール・ファイル強制出力指定 （-J/-NJ）

【記述形式】

```
-J
-NJ
```

- **省略時解釈**

  -NJ

【機能】

- -J **オプションは，フェータル・エラーの場合にでもロード・モジュール・ファイルを出力するよう指示します。**
- -NJ **オプションは，**-J **オプションを無効にします。**

【用途】

- **通常フェータル・エラーがある場合には，ロード・モジュール・ファイルは出力されません。**
  したがって，フェータル・エラーがあるのを承知でプログラムを実行させたい場合には，-J **オプションを指定しロード・モジュール・ファイルを出力します。**

【説明】

- -J **オプションを指定すると，フェータル・エラーがある場合でも，ロード・モジュール・ファイルが出力されます。**
- -J **と** -NJ **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**
- -NO **オプションを指定した場合は，**-J **オプションは無効となります。**

【使用例】

- **ロード・モジュール・ファイルを強制的に出力します。**

```
C>lk78k0 k0main.rel k0sub.rel -j
```

(3) ディバグ情報出力指定

ディバグ情報出力指定 （-G/-NG）

【記述形式】

```
-G
-NG
```

- **省略時解釈**

  -G

【機能】

- -G **オプションは，ロード・モジュール・ファイル中にディバグ情報（ローカル・シンボル情報）を付加する指定をします。**
- -NG **オプションは，**-G，-KP，-KL **オプションを無効にします。**

【用途】

- **ソース・ディバッガでシンボリック・ディバグを行うときには，必ず，**-G **オプションを指定します。**

【説明】

- -NG **オプションが指定された場合，パブリック・シンボル・リスト，およびローカル・シンボル・リストは出力されません。**
- -G **と** -NG **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**
- -NO **オプションを指定した場合，**-G **オプションは無効となります。**

【使用例】

- **ロード・モジュール・ファイルにディバグ情報を付加します。**

```
C>lk78k0 k0main.rel k0sub.rel -g
```

(4) スタック解決用シンボル生成指定

スタック解決用シンボル生成指定 （-S/-NS）

【記述形式】

-S [領域名] -NS

- 省略時解釈

  -NS

【機能】

- -S オプションは，スタック解決用のパブリック・シンボル“ _@STBEG ”と“ _@STEND ”を生成します。
- -NS オプションは，-S オプションを無効にします。

【用途】

- スタック領域を確保するために -S オプションを指定します。

【説明】

- “ 領域名 ”には，利用者が定義したメモリ領域名，またはディフォールトで定義されているメモリ領域名を指定します。
- “ 領域名 ”は，大文字と小文字を区別します。
- リンカは，-S オプションで指定したメモリ領域の中で，セグメントが配置されていない領域のうち，最大の空き領域を探します。そして，見つかった最大の空き領域の先頭アドレスを値として持つパブリック・シンボル“ _@STEND ”と，最終アドレス + 1 を値として持つパブリック・シンボル“ _@STBEG ”を生成します。

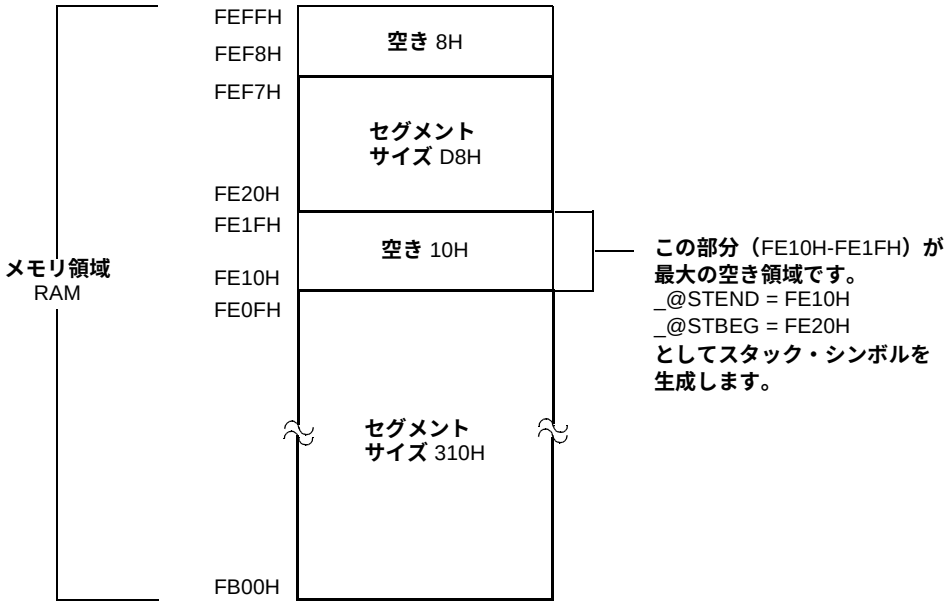
  これらのシンボルは，パブリック宣言された NUMBER 属性のシンボルとして扱われ，リンカのシンボル・テーブルの最後に登録されます。また，リンク・リスト・ファイルに出力される際，モジュール名欄は空白となります。
- 最大の空き領域のサイズが 10 バイト以下の場合，ワーニング・メッセージが出力されます。
- 空き領域が存在しない場合，ワーニング・メッセージが出力され，“ _@STEND ”と“ _@STBEG ”は指定したメモリ領域の最終アドレス + 1 を持ちます。
- “ 領域名 ”が省略された場合，“ RAM ”が指定されたものとします。
- -S と -NS の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

【使用例】

- **メモリ領域** RAM **にスタック領域を確保します。**

  （ただし，RAM **領域にサイズ** 310H **のセグメントと** saddr **領域に配置するサイズ** D8H **のセグメントを入力すると仮定します）**

  C>lk78k0 k0main.rel k0sub.rel -s



(5) ディレクティブ・ファイル指定

ディレクティブ・ファイル指定 （-D）

【記述形式】

-D ファイル名

- 省略時解釈

  なし

【機能】

- -D オプションは，指定ファイルをディレクティブ・ファイルとして入力することを指定します。

【用途】

- メモリ領域を新たに定義したり，ディフォールトのメモリ領域を再定義する場合，またはセグメントを特定のアドレスやメモリ領域に配置したい場合，ディレクティブ・ファイルを作成します。このディレクティブ・ファイルをリンカに入力するために，-D オプションを指定します。

【説明】

- “ファイル名”として指定できるのは，ディスク型ファイル名のみです。デバイス型ファイル名を指定するとアボート・エラーとなります。
- ファイル名を省略するとアボート・エラーとなります。
- ディレクティブ・ファイルのネストは許されません。
- ディレクティブ・ファイル中に記述できる文字数の制限はありません。
- -D オプションを複数指定したり，ファイル名を複数指定するとアボート・エラーとなります。
- ディレクティブ・ファイルの詳細については，「6.4 リンク・ディレクティブ」を参照してください。

【使用例】

- ディフォールトのメモリ領域 ROM/RAM を再定義します。

＜ディレクティブ・ファイル k0.dr の内容＞

```
memory ROM: ( 0000h , 1000h )
memory RAM: ( 0FE20h , 1E0h )
```

k0.dr を使用してリンクします。

```
C>lk78k0 k0main.rel k0sub.rel –dk0.dr
```

(6) リンク・リスト・ファイル出力指定

リンク・リスト・ファイル出力指定 （-P/-NP）

【記述形式】

-P [出力ファイル名] -NP

- **省略時解釈**

  -P 入力ファイル名 .map

【機能】

- -P **オプションは，リンク・リスト・ファイルの出力を指定します。また，その出力先や出力ファイル名を指定します。**
- -NP **オプションは，**-P**，**-KM**，**-KD**，**-KP**，**-KL**，**-LL**，**-LF **オプションを無効にします。**

【用途】

- **リンク・リスト・ファイルの出力先や出力ファイル名を変更する場合，**-P **オプションを指定します。**
- **ロード・モジュール・ファイルの出力だけを目的でリンクする場合などに，**-NP **オプションを指定します。これによりリンク時間が短縮されます。**

【説明】

- **ファイル名としてディスク型ファイル名とデバイス型ファイル名を指定できます。**
  ただし，指定できるデバイス型ファイル名は CON , PRN , NUL**，および** AUX **です。**CLOCK **を指定した場合，アボート・エラーとなります。**
- -P **オプションを指定する際に，“ 出力ファイル名 ” を省略すると，カレント・ディレクトリにリンク・リスト・ファイル “ 入力ファイル名** .map **” を出力します。**
- **“ 出力ファイル名 ” にパス名のみを指定すると，指定したパスに “ 入力ファイル名** .map **” を出力します。**
- -P **と** -NP **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**

【使用例】

- **リンク・リスト・ファイル（**k0.map**）を作成します。**

```
C>lk78k0 k0main.rel k0sub.rel –pk0.map
```

(7) リンク・リスト・ファイル情報指定

リンク・リスト・ファイル情報指定 （-KM/-NKM , -KD/-NKD , -KP/-NKP , -KL/-NKL）

(a) -KM/-NKM

【記述形式】

-KM -NKM

- **省略時解釈**

  -KM

【機能】

- -KM **オプションは，リンク・リスト・ファイル中にマップ・リストを出力します。**
- -NKM **オプションは，**-KM**，**-KD **オプションを無効にします。**

【用途】

- **リンク・リスト・ファイル中にマップ・リストを出力したいときに，**-KM **オプションを指定します。**

【説明】

- -NKM , -NKP**，および** -NKL **オプションがすべて指定された場合は，リンク・リスト・ファイルは出力されません。**
- -NKM **オプションを指定した場合は，リンク・リスト・ファイル中にリンク・ディレクティブ・ファイルは出力されません。**
- -KM **と** -NKM **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**
- -NP **オプションを指定した場合は，**-KM **オプションは無効となります。**

【使用例】

- **リンク・リスト・ファイル** k0.map **にマップ・リストを出力します。**

```
C>lk78k0 k0main.rel k0sub.rel –pk0.map -km
```

k0.map **を参照します。**

```
78K/0 Series Linker Vx.xx                                   Date : xx xxx xxxx Page :   1

Command:k0main.rel k0sub.rel –pk0.map -km
Para-file :
Out-file : K0MAIN.LMF
Map-file : K0.MAP
Direc-file :
Directive :

*** Link information ***

  3           output segment ( s )
  2FH         byte ( s ) real data
  23          symbol ( s ) defined

*** Memory map ***

  SPACE = REGULAR

  MEMORY = ROM
  BASE ADDRESS = 0000H      SIZE = 8000H
                OUTPUT      INPUT       INPUT       BASE        SIZE
                SEGMENT     SEGMENT     MODULE      ADDRESS
                CODE                                0000H       002H        CSEG AT
                CODE        SAMPM                   0000H       0002H
* gap *                                  0002H       007EH
                ?CSEG                               0080H       0020H       CSEG
                            ?CSEG       SAMPM       0080H       0013H
                            ?CSEG       SAMPS       0093H       001AH
* gap *                                              00ADH       7F53H

  MEMORY = LRAM
  BASE ADDRESS = FAC0H    SIZE = 0020H
                OUTPUT      INPUT       INPUT       BASE        SIZE
                SEGMENT     SEGMENT     MODULE      ADDRESS
* gap *
  MEMORY = RAM
  BASE ADDRESS = FB00H    SIZE = 0500H
                OUTPUT      INPUT       INPUT       BASE        SIZE
                SEGMENT     SEGMENT     MODULE      ADDRESS
* gap *                                              FB00H       03H
                DATA                                FE20H       0003H       DSEG AT
                            DATA        SAMPM       FE20H       0003H
* gap *                                              FE23H       00DDH
* gap ( Not Free Area ) *                            FF00H       0100H
```

マップ・リスト

(b) -KD/-NKD

【記述形式】

```
-KD
-NKD
```

- **省略時解釈**

  -KD

【機能】

- -KD **オプションは，リンク・リスト・ファイル中にリンク・デイレクティブ・ファイルを出力します。**
- -NKD **オプションは，**-KD **オプションを無効にします。**

【用途】

- **リンク・リスト・ファイル中に，リンク・ディレクティブ・ファイルを出力したい場合は，**-KD **オプションを指定します。**

【説明】

- -NKM，-NKP，**および** -NKL **オプションがすべて指定された場合は，リンク・リスト・ファイルは出力されません。**
- -NKM **オプションを指定した場合は，リンク・リスト・ファイル中にリンク・ディレクティブ・ファイルは出力されません。**
- -KD **と** -NKD **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**
- -NP **オプションを指定した場合は，**-KD **オプションは無効となります。**

【使用例】

- **リンク・リスト・ファイル（**k0.map**）にリンク・ディレクティブ・ファイルを出力します。**

  C>lk78k0 k0main.rel k0sub.rel –dk0.dr –pk0.map -kd

  k0.map **を参照します。**

```
78K/0 Series Linker Vx.xxDate:xx xxx xxxx Page :   1

Command : k0main.rel k0sub.rel –dk0.dr –pk0.map -kd
Para-file :
Out-file : K0MAIN.LMF
Map-file : K0.MAP
Direc-file : K0.DR                                  ←ディレクティブ・ファイル名
Directive : memory ROM : ( 0h , 4000h )             ←ディレクティブ・ファイルの内容
            memory RAM : ( 0fe20h , 1000h )

*** Link information ***

        3       output segment ( s )
        48H     byte ( s ) real data
        23      symbol ( s ) defined

*** Memory map ***

        SPACE = REGULAR

        MEMORY = ROM
        BASE ADDRESS = 0000H     SIZE = 1000H
                OUTPUT          INPUT           INPUT           BASE    SIZE
                SEGMENT         SEGMENT         MODULE          ADDRESS
                CODE                                            0000H   0002H   CSEG AT
                  :
```

(c) -KP/-NKP

【記述形式】

```
-KP
-NKP
```

- **省略時解釈**

  -NKP

【機能】

- -KP **オプションは，リンク・リスト・ファイル中にパブリック・シンボル・リストを出力します。**
- -NKP **オプションは，**-KP **オプションを無効にします。**

【用途】

- **リンク・リスト・ファイル中に，パブリック・シンボル・リストを出力する場合に** -KP **オプションを指定します。**

【説明】

- -NKM**，**-NKP**，および** -NKL **オプションがすべて指定された場合は，リンク・リスト・ファイルは出力されません。**
- -NG **オプションを指定した場合，パブリック・シンボル・リストは出力されません。**
- -KP **と** -NKP **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**
- -NP **オプションを指定した場合は，**-KP **オプションは無効となります。**

【使用例】

- **リンク・リスト・ファイル**（k0.map）**に，パブリック・シンボル・リストを出力します。**

```
C>lk78k0 k0main.rel k0sub.rel -g –pk0.map -kp
```

k0.map **を参照します。**

```
78K/0 Series Linker Vx.xx                                  Date : xx xxx xxxx Page :   1

Command : k0main.rel k0sub.rel -g –pk0.map -kp
Para-file :
Out-file : K0MAIN.LMF
Map-file : K0.MAP
Direc-file :
Directive :

*** Link information ***

          3         output segment ( s )
          2FH       byte ( s ) real data
          23        symbol ( s ) defined

*** Memory map ***

          SPACE = REGULAR

          MEMORY = ROM
          BASE ADDRESS = 0000   SIZE = 8000H
             :

--------------------------------------------------------------------------------

78K/0 Series Linker Vx.xx                                  Date:xx xxx xxx Page :   2

*** Public symbol list ***
MODULE          ATTR    VALUE   NAME

SAMPM           ADDR    0000H   MAIN
SAMPM           ADDR    0080H   START
SAMPS           ADDR    0093H   CONVAH

 Target chip : uPD78xxx
 Device file : Vx.xx
```

パブリック・シンボル・リスト

(d) -KL/-NKL

【記述形式】

```
-KL
-NKL
```

- **省略時解釈**

  -NKL

【機能】

- -KL **オプションは，リンク・リスト・ファイル中にローカル・シンボル・リストを出力します。**
- -NKL **オプションは，**-KL **オプションを無効にします。**

【用途】

- **リンク・リスト・ファイル中にローカル・シンボル・リストを出力する場合に，**-KL **オプションを指定します。**

【説明】

- -NKM**，**-NKP**，および** -NKL **オプションがすべて指定された場合は，リンク・リスト・ファイルは出力されません。**
- -NG **オプションを指定した場合は，ローカル・シンボル・リストは出力されません。**
- -KL **と** -NKL **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**
- -NP **オプションを指定した場合は，**-KL **オプションは無効となります。**

【使用例】

- **リンク・リスト・ファイル（**k0.map**）に，ローカル・シンボル・リストを出力します。**

C>lk78k0 k0main.rel k0sub.rel –pk0.map -kl

k0.map **を参照します。**

```
78K/0 Series Linker Vx.xx                                Date:xx xxx xxxx Page :  1

Command:k0main.rel k4sub.rel -g –pk0.map -kl
Para-file :
Out-file : K0MAIN.LMF
Map-file : K0.MAP
Direc-file :
Directive :

*** Link information ***

          3        output segment ( s )
          2FH      byte ( s ) real data
          23       symbol ( s ) defined

*** Memory map ***

          SPACE = REGULAR
             :
--------------------------------------------------------------------------------

78K/0 Series Linker Vx.xx                                Date:xx xxx xxx Page :  2

*** Local symbol list ***
MODULE          ATTR            VALUE   NAME

SAMPM           MOD                     SAMPM
SAMPM           DSEG                    DATA
SAMPM           ADDR            FE20H   HDTSA
SAMPM           ADDR            FE21H   STASC
SAMPM           CSEG                    CODE
SAMPM           CSEG                    ?CSEG
SAMPS           MOD                     SAMPS
SAMPS           CSEG                    ?CSEG
SAMPS           ADDR            00A4H   SASC
SAMPS           ADDR            00AAH   SASC1

 Target chip : uPD78xxx
 Device file : Vx.xx
```

ローカル・シンボル・リスト

(8) リンク・リスト・ファイル形式指定

リンク・リスト・ファイル形式指定 （-LL , -LF/-NLF）

(a) -LL

【記述形式】

-LL [**行数**]

- **省略時解釈**

  -LL66（**ディスプレイ出力の場合は改頁しません**）

【機能】

- -LL **オプションは，リンク・リスト・ファイルの** 1 **頁の行数を指定します。**

【用途】

- **リンク・リスト・ファイルの** 1 **頁の行数を変更したいときに，**-LL **オプションで指定します。**

【説明】

- -LL **オプションで指定できる行数の範囲は次のとおりです。**

  20 ≦ 1 **頁に印字する行数** ≦ 32767

  範囲外の数値や数値以外のものが指定された場合にはアボート・エラーとなります。
- **行数が省略された場合，**66 **が指定されたものとみなします。**
- **行数に** 0 **を指定した場合は改頁しません。**
- -NP **オプションを指定した場合は，**-LL **オプションは無効となります。**

【使用例】

- **リンク・リスト・ファイルの** 1 **頁の行数を** 20 **行に指定します。**

```
C>lk78k0 k0main.rel k0sub.rel –pk0.map -ll20
```

k0.map **を参照します。**

```
78K/0 Series Linker Vx.xx                                    Date : xx xxx xxxx Page :   1

Command:k0main.rel k0sub.rel –pk0.map -ll20
Para-file :
Out-file : K0MAIN.LMF
Map-file : K0.MAP
Direc-file :
Directive :

*** Link information ***

          3          output segment ( s )
          2FH        byte ( s ) real data

--------------------------------------------------------------------------------

78K/0 Series Linker Vx.xxDate:xx xxx xxxx Page :   2

          23 symbol ( s ) defined

*** Memory map ***

 SPACE = REGULAR

 MEMORY = ROM
 BASE ADDRESS = 0000H      SIZE = 8000H
          OUTPUT          INPUT           INPUT           BASE            SIZE
          SEGMENT         SEGMENT         MODULE          ADDRESS

--------------------------------------------------------------------------------

78K/0 Series Linker Vx.xx                                    Date : xx xxx xxxx Page :   3
          CODE                            0000H   00000002H   CSEG AT
                          CODE            SAMPM           0000H           00000002H
* gap *                                                   0002H           0000007EH
          ?CSEG                                         0080H           00000046H   CSEG
                          ?CSEG           SAMPM           0080H           0000002AH
                          ?CSEG           SAMPS           0093H           0000001CH
* gap *                                                   00ADH           0000FF3AH

 MEMORY = LRAM
 BASE ADDRESS = FAC0H      SIZE = 0020H
          OUTPUT          INPUT   INPUT   BASE    SIZE
            :
```

(b) -LF/-NLF

【記述形式】

```
-LF
-NLF
```

- **省略時解釈**

  -NLF

【機能】

- -LF **オプションは，リンク・リスト・ファイルの最後に，改頁コード**（FF）**を付加する指定をします。**
- -NLF **オプションは，**-LF **オプションを無効にします。**

【用途】

- **リンク・リスト・ファイルの内容を印字したあとで改頁しておきたい場合，**-LF **オプションを指定して改頁コードを付加します。**

【説明】

- -NP **オプションを指定した場合は，**-LF **オプションは無効となります。**
- -LF **と** -NLF **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**

【使用例】

- **リンク・リスト・ファイルの最後の改頁コードを付加します。**

```
C>lk78k0 k0main.rel k0sub.rel –pk0.map -lf
```

(9) エラー・リスト・ファイル出力指定

エラー・リスト・ファイル出力指定 （-E/-NE）

【記述形式】

```
-E [ ファイル名 ]
-NE
```

- **省略時解釈**

  -NE

【機能】

- -E **オプションは，エラー・リスト・ファイルの出力を指定します。また，その出力先や出力ファイル名を指定します。**
- -NE **オプションは，**-E **オプションを無効にします。**

【用途】

- **エラー・リスト・ファイルの出力先や出力ファイル名を変更したいときに，**-E **オプションを指定します。**

【説明】

- **ファイル名として，ディスク型ファイル名とデバイス型ファイル名を指定できます。**
  ただし，デバイス型ファイル名として CLOCK **を指定した場合は，アボート・エラーとなります。**
- -E **オプションを指定する際に出力ファイル名を省略すると，エラー・リスト・ファイル名は，“ 入力ファイル名** .elk **” となります。**
- -E **オプションを指定する際にドライブ名を省略すると，カレント・ドライブにエラー・リスト・ファイルが出力されます。**
- -E **と** -NE **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**

【使用例】

- **エラー・リスト・ファイル（**k0.elk**）を作成します。**

  C>lk78k0 k0main.rel k0sub.rel –dk0.dr –ek0.elk

  ディレクティブ・ファイルの内容に誤りがありました。k0.elk **を参照します。**

  ```
  K0.DR ( 3 ) : RA78K0 error E3102: Directive syntax error
  ```

(10) ライブラリ・ファイル指定

ライブラリ・ファイル指定 （-B）

【記述形式】

-B ファイル名

- 省略時解釈

  なし

【機能】

- -B オプションは，指定ファイルをライブラリ・ファイルとして入力することを指定します。

【用途】

- リンカは入力モジュールが参照しているモジュールを，ライブラリ・ファイルから探し出し，そのモジュールだけを入力モジュールと結合します。
- ライブラリ・ファイルは，あらかじめ複数のモジュールを登録して 1 つのファイルにまとめておくためのものです。
- 共通的なモジュールをライブラリ・ファイルとして作成しておけば，ファイル管理の面でも操作性の面においても効率がよくなります。ライブラリ・ファイルをリンカに入力するには，-B オプションを指定します。

【説明】

- ファイル名としてディスク型ファイル名以外は指定できません。
- ファイル名は省略できません。
- ファイル名としてパス名を含めて指定した場合は，指定されたパスからライブラリ・ファイルを入力します。指定されたパスにライブラリ・ファイルが存在しない場合は，エラーとなります。
- ファイル名としてパス名を含まずに指定した場合には，-I オプションの指定，およびディフォールトのサーチ・パスからライブラリ・ファイルを入力します。
- -B オプションが複数指定された場合は，指定順にライブラリ・ファイルの入力を行います。-B オプションは最大 10 個まで指定できます。

  注意　PM plus 上の〈リンカオプションの設定〉ダイアログでライブラリを複数指定する場合は，ファイルの間をコンマ（ , ）で区切ってください。
- ライブラリ・ファイルの作成方法の詳細は，「第 8 章 ライブラリアン」をお読みください。

【使用例】

- ライブラリ・ファイル（k0.lib）を入力します。

  ライブラリ・ファイルには，k0sub.rel が登録されています。

```
C>lk78k0 k0main.rel –bk0.lib
```

(11) ライブラリ・ファイル読み込みパス指定

ライブラリ・ファイル読み込みパス指定 （-l）

【記述形式】

-l パス名 [, パス名] …（複数指定可能）

- 省略時解釈

  環境変数 “ LIB78K0 ” により指定されたパス
  指定されていない場合は，カレント・パス

【機能】

- ライブラリ・ファイルを指定したパスから入力するよう指示します。

【用途】

- ライブラリ・ファイルをあるパスから検索したいときに指定します。

【説明】

- -l オプションは，-B オプションでパス名を含まないライブラリ・ファイル名が指定された場合にのみ有効です。
- -l は複数指定できます。また，“ , ” で区切ることにより，1 度に複数のパスを指定できます。この際 “ , ” の前後には空白を入れることはできません。
- パス名は，1 回のリンクで最大 64 個まで指定できます。パス名が複数指定された場合リンカは指定順にライブラリ・ファイルのサーチを行います。
- 指定したパスにライブラリ・ファイルが存在しない場合でもエラーとはなりません。
- パス名が省略された場合，アボート・エラーとなります。
- -B オプションでパス名を含まずにライブラリ・ファイルを指定した場合，リンカは次に示す順序でパスをサーチします。

  (i) -l オプションで指定したパス
  (ii) 環境変数 “ LIB78K0 ” で指定されたパス
  (iii) カレント・パス

  いずれのパスにも指定された名前のライブラリ・ファイルが存在しない場合にはエラーとなります。

【使用例】

- ライブラリ・ファイルをディレクトリ C：\LIB から検索します。

```
C>lk78k0 k0main.rel k0sub.rel –bk0.lib –ic:\lib
```

(12) パラメータ・ファイル指定

パラメータ・ファイル指定 （-F）

【記述形式】

-F ファイル名

- 省略時解釈

  コマンド行上からのみオプション，入力ファイル名の入力が可能

【機能】

- -F オプションは，オプションあるいは入力ファイル名を指定のファイルから入力する指定をします。

【用途】

- コマンド行では，リンカの起動に必要な情報を指定しきれないときに，-F オプションを指定します。リンクするたび繰り返し同じようにオプションを指定する場合には，それらをパラメータ・ファイルに記述しておき，-F オプションで指定します。

【説明】

- “ ファイル名 ” として指定できるのは，ディスク型ファイル名のみです。デバイス型ファイル名を指定するとアボート・エラーとなります。
- ファイル名を省略するとアボート・エラーとなります。
- パラメータ・ファイルのネストは許されません。パラメータ・ファイル中で，-F オプションを指定するとアボート・エラーとなります。
- パラメータ・ファイル中に記述できる文字数の制限はありません。
- 空白とタブ，および改行文字（LF）をオプションあるいは入力ファイル名の区切りとします。
- パラメータ・ファイル中に記述したオプションあるいは入力ファイル名は，コマンド行上のパラメータ・ファイル指定のあった位置に展開されます。
- 展開されたオプションは，あとで指定した方が有効となります。
- “ ; ”,または‘ # ’以降に記述された文字は改行文字（LF）,またはEOFの前まですべてコメントと解釈します。
- -F オプションを複数指定するとアボート・エラーとなります。

【使用例】

- パラメータ・ファイルを使用してリンクします。

  ＜パラメータ・ファイル（k0.plk）の内容＞

```
; parameter file
k0main.rel k0sub.rel –ok0.lmf –pk0.map -e
-tc:\tmp -g
```

  コマンド行には，次のように入力します。

```
C>lk78k0 –fk0.plk
```

(13) テンポラリ・ファイル作成パス指定

テンポラリ・ファイル作成パス指定 （-T）

【記述形式】

-T パス名

- 省略時解釈

  環境変数 “ TMP ” により指定されたパスに作成します。

  指定されていない場合は，カレント・パスに作成します。

【機能】

- -T オプションは，テンポラリ・ファイルを作成するパスを指示します。

【用途】

- テンポラリ・ファイルの作成場所を指定できます。

【説明】

- パス名として，パス以外のものは指定できません。
- パス名は省略できません。
- 以前に作成されたテンポラリ・ファイルが存在している場合でも，ファイル保護がされていなければ，上書きします。
- 必要とするメモリ・サイズがある間は，テンポラリ・ファイルをメモリに展開します。メモリが足りなくなった時点で，メモリに展開していたテンポラリ・ファイルの内容をディスクに書き出します。以降のテンポラリ・ファイルへのアクセスは，セーブしたディスク・ファイルに対して行います。
- テンポラリ・ファイルは，リンク終了時に削除されます。また，キー入力（ CTRL-C ）によってリンクが中止されたときも，削除されます。
- テンポラリ・ファイルの作成パスは，次の順番で決定されます。

  (i) -T オプションで指定されたパス

  (ii) 環境変数 TMP に設定されているパス（-T オプション省略の場合）

  (iii) カレント・パス（TMP が設定されていない場合）

  なお，(i) または (ii) を指定した場合，指定されたパスにテンポラリ・ファイルが作成できなければアボート・エラーとなります。

【使用例】

- テンポラリ・ファイルをディレクトリ “ TMP ” に作成するように指定します。

```
C>lk78k0 k0main.rel k0sub.rel -t\tmp
```

(14) デバイス・ファイル・サーチ・パス指定

デバイス・ファイル・サーチ・パス指定 （-Y）

【記述形式】

-Y パス名

- 省略時解釈

  デバイス・ファイルを読み込むパスは，次の順序で調べ決定します。

  (i) < ..\dev >（lk78k0.exe の起動されたパスに対して）

  (ii) LK78K0 の起動されたパス

  (iii) カレント・ディレクトリ

  (iv) 環境変数 PATH

【機能】

- デバイス・ファイルを指定されたパスから読み込みます。

【用途】

- デバイス・ファイルが存在するパスを指定します。

【説明】

- -Y オプションに続けてパス名以外が指定された場合，アボート・エラーとなります。
- -Y オプションに続けて指定するパス名が省略された場合，アボート・エラーとなります。
- デバイス・ファイルを読み込むパスは，次の順序で調べ決定します。

  (i) -Y オプションで指定されたパス

  (ii) < ..\dev >（ lk78k0.exe の起動されたパスに対して）

  (iii) LK78K0 の起動されたパス

  (iv) カレント・ディレクトリ

  (v) 環境変数 PATH

【使用例】

- デバイス・ファイルのパスを c:\78k0\dev ディレクトリに指定します。

```
C>lk78k0 k0main.rel k0sub.rel –yc:\78k0\dev
```

(15) ワーニング・メッセージ出力指定

ワーニング・メッセージ出力指定 （-W）

【記述形式】

-W [**レベル**]

- **省略時解釈**

  -W1

【機能】

- -W **オプションは，ワーニング・メッセージをコンソールへ出力するか否かを指定します。**

【用途】

- **ワーニング・メッセージを出力させるレベルを指定できます。**

【説明】

- -W **オプションに続けてレベル以外が指定された場合，アボート・エラーとなります。**
- **レベルには，**0，1，2 **以外は指定できません。**
- **出力させるレベルには以下のものがあります。**

  0 **：ワーニング・メッセージを出力しません。**

  1 **：通常のワーニング・メッセージを出力します。**

  2 **：詳細なワーニング・メッセージを出力します。**

  なお，具体的に出力されるか否かは，「12.4 **リンカのエラー・メッセージ」を参照してください。**

【使用例】

- **レベルを** 2 **として** -W **オプションを指定します。**

```
C>lk78k0 k0main.rel k0sub.rel -w2
```

(16) フラッシュ ROM 内蔵製品のブート領域 ROM プログラムのリンク指定

フラッシュ ROM 内蔵製品のブート領域 ROM プログラムのリンク指定 （-ZB）

【記述形式】

-ZB **アドレス**

- **省略時解釈**

  リンク指定なし

【機能】

- **フラッシュ** ROM **領域の先頭アドレスを指定します。**

【説明】

- **フラッシュ** ROM **内蔵製品のブート領域** ROM **プログラムのリンク指定を行い，フラッシュ** ROM **領域の先頭アドレスを指定します。**
- **アドレスが省略された場合はエラーとなります。**

注意　フラッシュ ROM **領域セルフ書き換え機能を持たないデバイスでは，このオプションを使用しないでください。**

【使用例】

```
C>lk78k0 k0main.rel -zb2000h
```

(17) オンチップ・ディバグのプログラム・サイズ指定

オンチップ・ディバグのプログラム・サイズ指定 （-GO）

【記述形式】

-GO [**サイズ**]

- **省略時解釈**

  オンチップ・ディバグを使用しない

【機能】

- -GO **オプションは，オンチップ・ディバグを使用するか否かを指定します。**

【用途】

- **オンチップ・ディバグのプログラム・サイズを変更したいときに，**-GO **オプションで指定します。**

【説明】

- **数値以外が指定された場合にはアボート・エラーとなります。**
- **プログラム・サイズを省略された場合，**256 **バイトが指定されたものとみなします。**
  プログラム・サイズの詳細については，「ID78K0 **の添付文書」を参照してください。**
- -GO **オプションが指定された場合，**02H **番地から** 03H **番地と，**8FH **から指定されたプログラム・サイズ分** +1 **の領域へ，セグメントを配置することはできません。**
- **オンチップ・ディバグ機能を持たないデバイスに対して，本オプションが指定されていた場合には，エラーとなります。**

【使用例】

- 8FH～18FH**番地（**256**バイト** +1**バイト）を，オンチップ・ディバグのプログラム配置領域として確保します。**

```
C>lk78k0 k0main.rel –go256
```

(18) セキュリティ ID 指定

セキュリティ ID 指定 （-GI）

【記述形式】

-GI セキュリティ ID

- 省略時解釈

  セキュリティ ID を設定しない

【機能】

- -GI オプションは，セキュリティ ID を指定します。

【用途】

- セキュリティ ID を設定したいときに，-GI オプションで指定します。

【説明】

- “H”で終わる 16 進数の数値で指定してください。それ以外が指定された場合にはアボート・エラーとなります。
- 10 バイト以内で指定してください。10 バイトに満たない場合には，上位ビットに 0 を補填します。
- セキュリティ ID は 85H ～ 8EH 番地に設定されます。セキュリティ ID を設定した場合，85H ～ 8EH 番地へセグメントを配置することはできません。
- セキュリティ ID 機能を持たないデバイスに対して，本オプションが指定されていた場合には，エラーとなります。
- セキュリティ ID は，アセンブラ・ソース中に以下の再配置属性のセグメントを定義することでも指定可能です。ただし，セグメントの再配置属性には必ず SECUR_ID を指定してください。

```
[ 任意のセグメント名 ]   CSEG   SECUR_ID
                         DB     11H
                         DB     22H
                         DB     33H
                         DB     44H
                         DB     55H
                         DB     66H
                         DB     77H
                         DB     88H
                         DB     99H
                         DB     0AAH
```

アセンブラ・ソースの指定と本オプションの指定が重なった場合には，本オプションを優先します。

【注意】

- **セキュリティ** ID **機能を持ったデバイスに対して，本オプションが指定されていない場合には，任意のコードが配置されることがありますので，注意してください。**

【使用例】

- **上記アセンブラ・ソースでの指定と同じ“** 112233445566778899AA **”を指定します。**

```
C>lk78k0 k0main.rel -gi112233445566778899aah
```

(19) ヘルプ指定

ヘルプ指定 (--)

【記述形式】

```
--
```

- 省略時解釈

  表示しない

【機能】

- -- オプションは，ヘルプ・メッセージをディスプレイに出力します。

【用途】

- ヘルプ・メッセージは，リンカ・オプションとその説明の一覧です。リンカを実行するときに参照してください。

【説明】

- -- オプションを指定すると，他のリンカ・オプションはすべて無効となります。
- ヘルプ・メッセージの続きをお読みになる場合は，リターン・キーを入力してください。表示を途中で終了する場合は，リターン・キー以外の文字を入力したあとで，リターン・キーを入力してください。

  注意　このオプションは，PM plus 上では指定できません。

  PM plus 上でヘルプを参照する場合は，〈リンカオプションの設定〉ダイアログで［ヘルプ］ボタンをクリックしてください。

【使用例】

- -- **オプションを指定するとヘルプ・メッセージがディスプレイに出力されます。**

C>lk78k0 --

```
78K/0 Series Linker Vx.xx [ xx xxx xx ]
    Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

usage:lk78k0 [ option [ ... ] ] input-file [ option [ ... ] ]
The option is as follows ( [ ] means omissible ).
-ffile              : Input option or input-file name from specified file.
-dfile              : Read directive file from specified file.
-bfile              : Read library file from specified file.
-idirectory [ , directory.. ] : Set library file search path.
-o [ file ] / -no   : Create load module file [ with specified name ] / Not.
-p [ file ] / -np   : Create link map file [ with specified name ] / Not.
-e [ file ] / -ne   : Create error list file [ with specified name ] / Not.
-tdirectory         : Set temporary directory.
-km / nkm           : Output map list to link map file / Not.
-kd / -nkd          : Output directive file image to link map file / Not.
-kp / -nkp          : Output public symbol list to link map file / Not.
-kl / -nkl          : Output local symbol list to link map file / Not.
-ll [ page length ] : Specify link map file lines per page.
-lf / -nlf          : Add Form Feed at end of the link map file / Not.
-s [ memory area ] / -ns   : Create stack symbol [ in specified memory area ] / Not.
-g / -ng            : Output symbol information to load module file / Not.
-ydirectory         : Set device file search path.
-j / -nj            : Create load module file if fatal error occurred / Not.
-w [ n ]            : Change warning level ( n = 0 to 2 ).
-zbaddress          : Create Boot file ( address:flash start address ).
-go [ n ]           : Change On-chip debug program size ( n = 256 to 1024 ).
-giid               : Set Security ID.
--                  : Show this message.

   Press RETURN to continue ...
DEFAULT ASSIGNMENT : -o -p -ne -km -kd -nkp -nkl -ll66 -nlf -ns -g -nj -w1

directive file usage :
 MEMORY memory-area-name : ( origin-value , size ) [ / memory-space-name ]
 MERGE segment-name : [ location-type-definition ] [ merge-type-definition ]
         [ = memory-area-name] [ / memory-space-name ]

 example : MEMORY ROM : ( 0H , 4000H )
                MEMORY RAMA : ( 0H , 100H ) / EX1
                MERGE CSEG1 : = ROM
                MERGE DSEG1 : AT ( 80H )
```

6.7 PM plus でのオプション設定

PM plus からリンカ・オプションを設定する方法について説明します。

6.7.1 オプションの設定方法

PM plus の [ツール (T)] メニューの [リンカオプションの設定 (L)] を選択するか，ツール・バーの［LK］ボタンを押下すると，〈リンカオプションの設定〉ダイアログが現れます。

ダイアログ内で必要なオプションを入力することにより，各リンカ・オプションを設定できます。

図 6-3 〈リンカオプションの設定〉ダイアログ（《出力 1》タブ選択時）

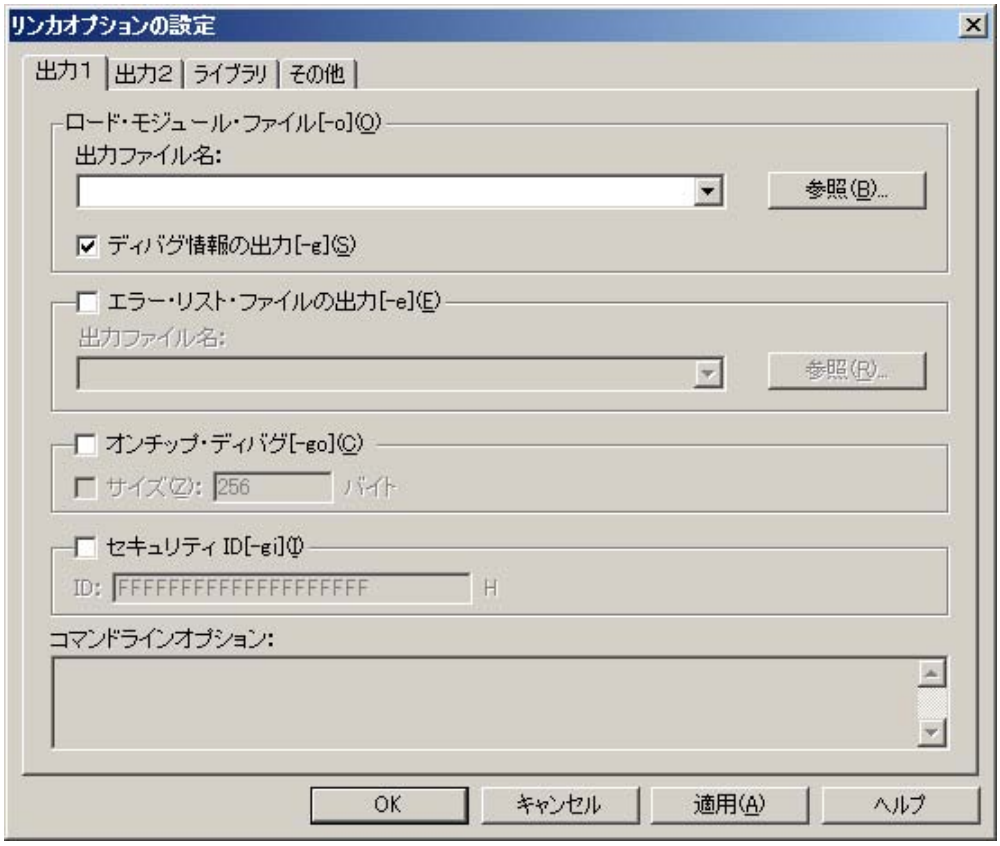


図 6-4 〈リンカオプションの設定〉ダイアログ（《出力 2》タブ選択時）

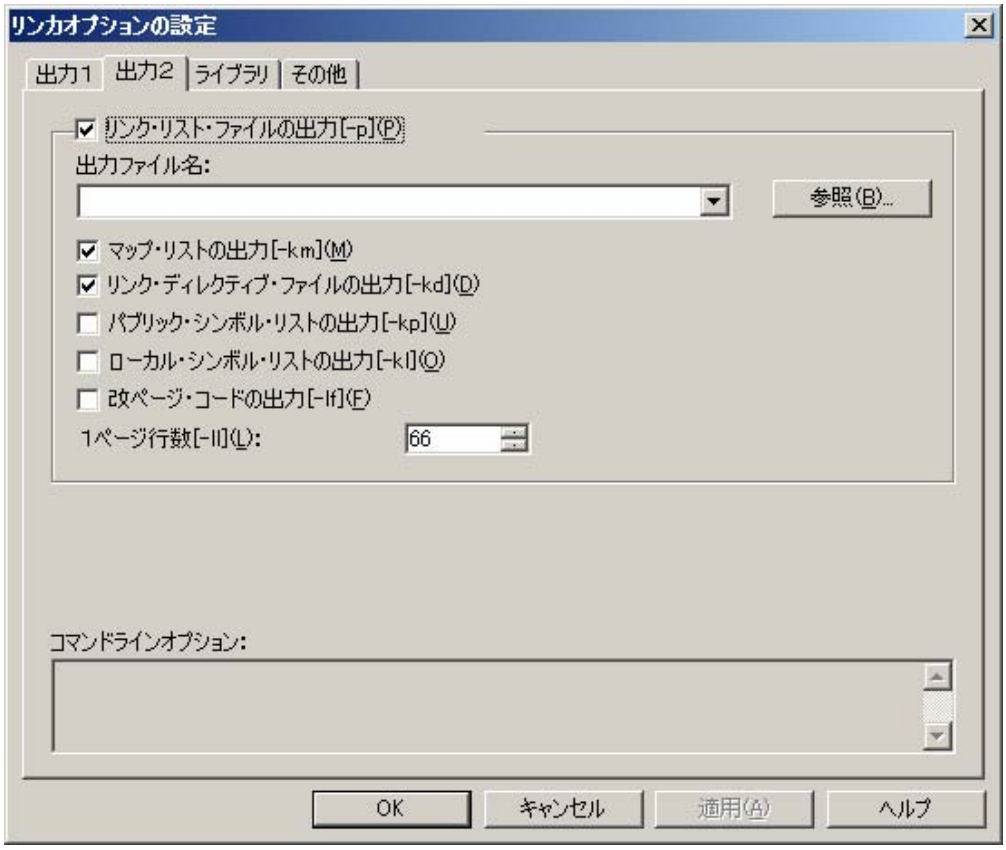


図 6-5 〈リンカオプションの設定〉ダイアログ（《ライブラリ》タブ選択時）

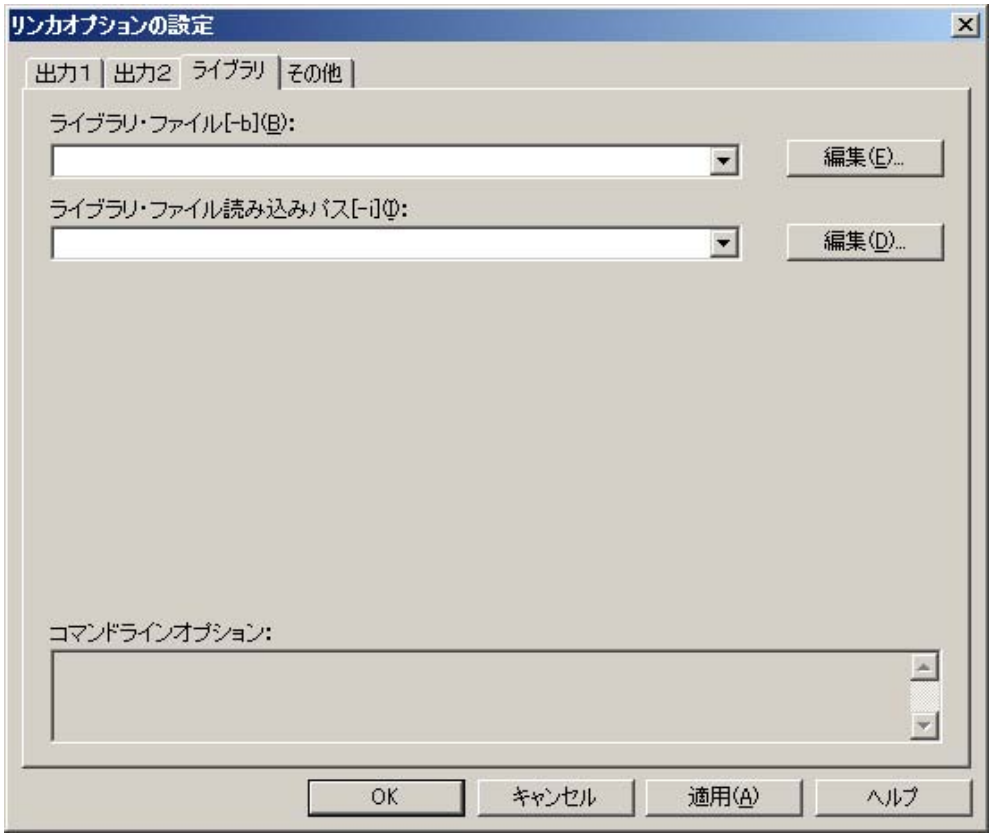
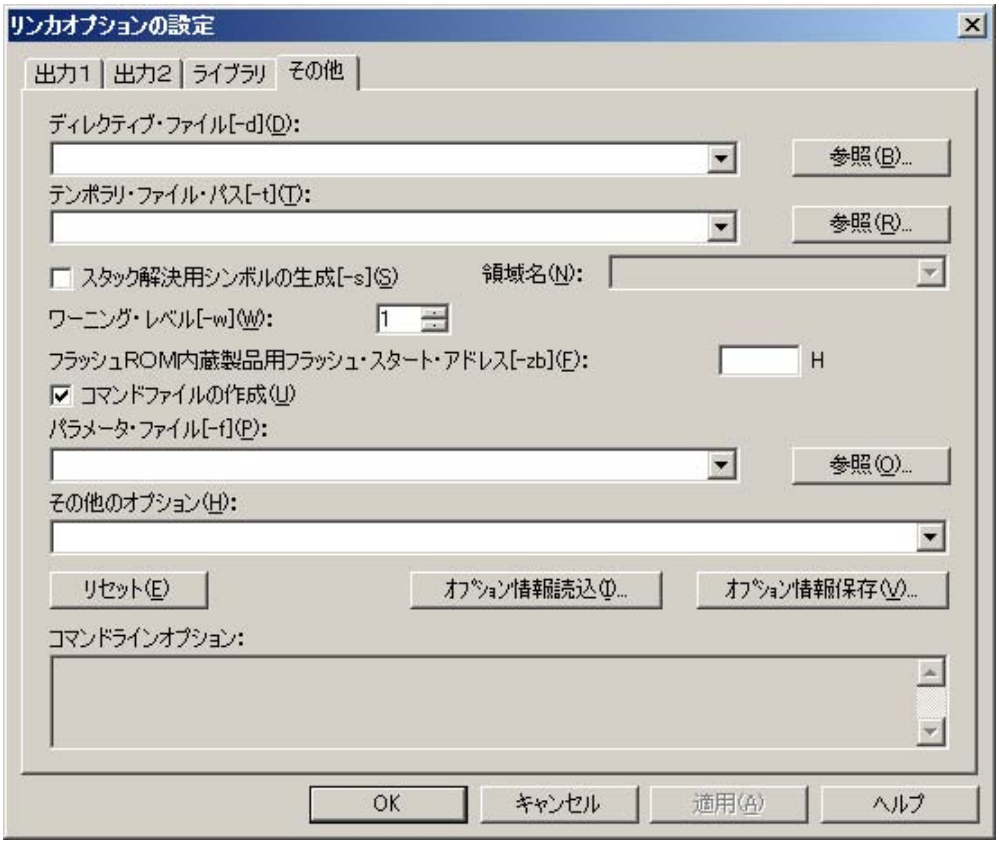


図 6-6 〈リンカオプションの設定〉ダイアログ（《その他》タブ選択時）



6.7.2 各オプションの設定

〈リンカオプションの設定〉ダイアログの各オプションについて，次に説明します。

《出力 1》タブ

- ロード・モジュール・ファイル [-o](O)

  出力ファイル名：

  [参照 (B)] ボタン，または直接入力により，ロード・モジュール・ファイルのパスとファイル名を指定します。

- ディバグ情報の出力 [-g](S)

  ロード・モジュール・ファイル中にディバグ情報 (ローカル・シンボル情報) を付加する場合に，チェックします。

- エラー・リスト・ファイルの出力 [-e](E)

  エラー・リスト・ファイルを出力する場合に，チェックします。

  出力パス名：

  [参照 (R)] ボタン，または直接入力により，エラー・リスト・ファイルのパスとファイル名を指定します。

- オンチップ・ディバグ [-go](C)

  オンチップ・ディバグを使用する場合に，チェックします。

  サイズ (Z)：

  オンチップ・ディバグのプログラム・サイズを指定します。

  注意　オンチップ・ディバグ機能を持たないデバイスでは，本オプションは指定できません。

- セキュリティ ID[-gi](I)

  セキュリティ ID を設定する場合に，チェックします。

  ID：

  セキュリティ ID を指定します。

  注意　セキュリティ ID 機能を持たないデバイスでは，本オプションは指定できません。

- コマンドラインオプション

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

《出力 2》タブ

- リンク・リスト・ファイルの出力 [-p](P)

  リンク・リスト・ファイルを出力する場合に，チェックします。

  出力ファイル名：

  [参照 (B)] ボタン，または直接入力により，リンク・リスト・ファイルのパスとファイル名を指定します。

- マップ・リストの出力 [-km](M)

  リンク・リスト・ファイル中にマップ・ファイルを出力する場合に，チェックします。

- リンク・ディレクティブ・ファイルの出力 [-kd](D)

  リンク・リスト・ファイル中にリンク・ディレクティブ・ファイルを出力する場合に，チェックします。

- **パブリック・シンボル・リストの出力** [-kp](U)

  リンク・リスト・ファイル中にパブリック・シンボル・リストを出力する場合に，チェックします。

- **ローカル・シンボル・リストの出力** [-kl](O)

  リンク・リスト・ファイル中にローカル・シンボル・リストを出力する場合に，チェックします。

- **改ページ・コードの出力** [-lf](F)

  リンク・リスト・ファイルの最後に改頁コード（FF**）を付加する場合に，チェックします。**

- 1 **ページ行数** [-ll](L)

  リンク・リスト・ファイルの１ページの行数を指定します。指定できる行数は，20 **〜** 32767 **までの範囲です。**

- **コマンドラインオプション**

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

《ライブラリ》タブ

- **ライブラリ・ファイル** [-b](B)

  ［編集 (E)**］ボタン，または直接入力により，ライブラリ・ファイルとして入力するファイルを指定します。**

- **ライブラリ・ファイル読み込みパス** [-i](I)

  ［編集 (D)**］ボタン，または直接入力により，ライブラリ・ファイルを読み込むパスを指定します。**

- **コマンドラインオプション**

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

《その他》タブ

- **ディレクティブ・ファイル** [-d](D)

  ［参照 (B)**］ボタン，または直接入力により，ディレクティブ・ファイルとして入力するファイルを指定します。**

- **テンポラリ・ファイル・パス** [-t](T)

  ［参照 (R)**］ボタン，または直接入力により，テンポラリ・ファイルを作成するパスを指定します。**

- **スタック解決用シンボルの生成** [-s](S)

  メモリ領域の中で最大の空き領域をスタック領域として確保する場合に，チェックします。

- **領域名** (N)

  利用者が定義したメモリ領域名，またはディフォールトで定義されているメモリ領域名を指定します。

- **ワーニング・レベル** [-w](W)

  ワーニング・メッセージを出力させるレベルを指定します。

  0 **：ワーニング・メッセージを出力しません。**

  1 **：通常のワーニング・メッセージを出力します。**

  2 **：詳細なワーニング・メッセージを出力します。**

- **フラッシュ** ROM **内蔵製品用フラッシュ・スタート・アドレス** [-zb](F)

  フラッシュ ROM **内蔵製品のブート領域スタート・アドレスを指定します。**

  注意　フラッシュ ROM **領域セルフ書き換え機能を持たないデバイスでは，このオプションを指定しないでください。**

- **コマンドファイルの作成** (U)

  コマンドファイルを作成する場合に，チェックします。

- **パラメータ・ファイル** [-f](P)

  ［参照 (C)**］ボタン，または直接入力により，ユーザ定義のパラメータ・ファイルとして入力するファイルを指定します。**

- **その他のオプション** (H)

  ダイアログで設定可能なオプション以外のオプションを指定したい場合に入力ボックスに入力します。

  注意　ヘルプ指定 (--) **オプションは，**PM plus **上では指定できません。**

- **リセット** (E)

  入力した内容をリセットします。

- **オプション情報読込** (I)

  〈オプション情報読込み〉ダイアログが開き，オプション情報ファイルを指定後，読み込みます。

- **オプション情報保存** (V)

  〈オプション情報の保存〉ダイアログが開き，オプション情報ファイルに名前をつけて保存します。

- **コマンドラインオプション**

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

6.7.3 オプションの編集ダイアログ

〈オプションの編集〉ダイアログは，項目をリストで編集します。

〈オプションの編集〉ダイアログについて，次に説明します。

図 6-7 〈オプションの編集〉ダイアログ

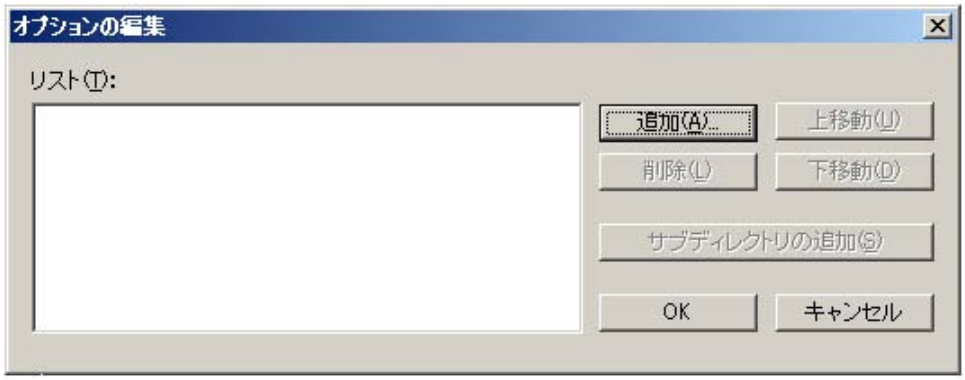


図 6-8 〈オプションの追加〉ダイアログ

- ［追加 (A)］ボタン

  リストの項目を追加します。

  ファイルやディレクトリを指定する項目の場合は，それぞれの〈参照〉ダイアログがオープンします。

  それ以外の場合は内容を入力する〈オプションの追加〉ダイアログがオープンします。

- ［削除 (L)］ボタン

  選択中のリストの項目を削除します。

- ［上移動 (U)］ボタン

  選択中のリストの項目を上に移動します。

- ［下移動 (D)］ボタン

  選択中のリストの項目を下に移動します。

- ［サブディレクトリの追加 (S)］ボタン

  次の場合は，選択中のリストの項目にサブディレクトリを追加できます。

  《ライブラリ》タブのライブラリ・ファイル読み込みパス [-i](I)

第 7 章　オブジェクト・コンバータ

この章では，オブジェクト・コンバータは，RA78K0 のリンカが出力したロード・モジュール・ファイル（すべての参照アドレス情報が解決されていなければなりません）を入力し，それを HEX 形式オブジェクト・モジュール・ファイルとして出力します。

さらに，シンボリック・ディバグ時に使用するシンボル情報をシンボル・テーブル・ファイルとして出力します。

オブジェクト・コンバータ・エラーがある場合は，エラー・メッセージを出力し，エラーの原因を明示します。

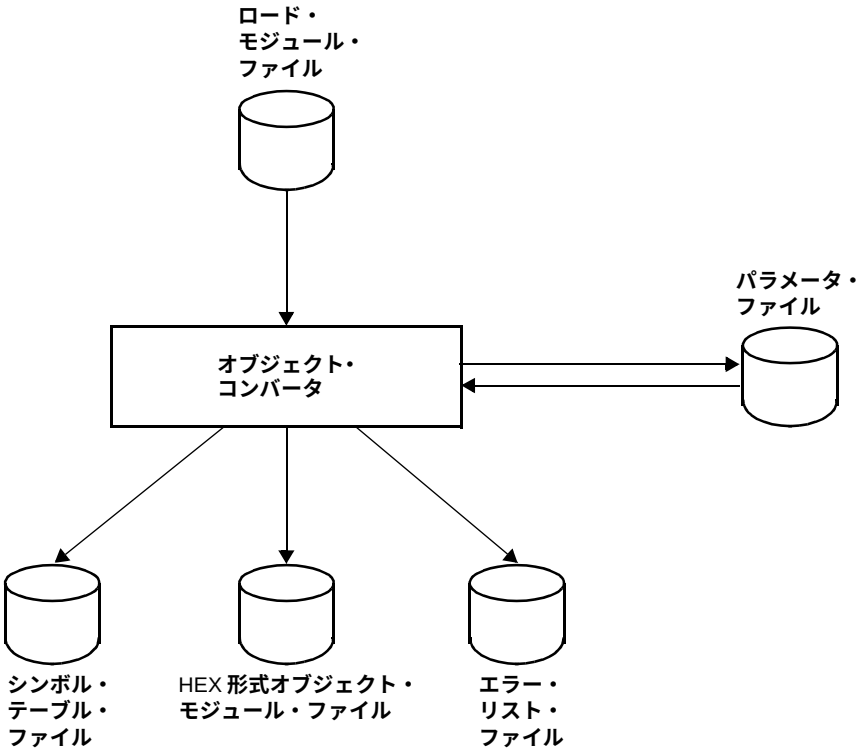
7.1　オブジェクト・コンバータの入出力ファイル

オブジェクト・コンバータの入出力ファイルを次に示します。

表 7-1 オブジェクト・コンバータの入出力ファイル

種類	ファイル名	説明	ディフォールト・ファイル・タイプ
入力ファイル	ロード・モジュール・ファイル	- リンクの結果のオブジェクト・コードのバイナリ・イメージ・ファイルです。 - リンカの出力したファイルです。	.lmf
	パラメータ・ファイル	- 実行プログラムのパラメータを内容とするファイルです。 - ユーザ作成ファイルです。	.poc
出力ファイル	HEX 形式オブジェクト・モジュール・ファイル	- ロード・モジュール・ファイルを，HEX 形式オブジェクト・フォーマットで変換したファイルです。 - マスク ROM 発注時，または PROM プログラマ使用時に使います。	.hex
	シンボル・テーブル・ファイル	- 入力ファイルの各モジュールに含まれるシンボルの情報を持つファイルです。	.sym
	エラー・リスト・ファイル	- オブジェクト・コンバート時のエラー情報を持つファイルです。	.eoc

図 7-1 オブジェクト・コンバータの入出力ファイル



7.2 オブジェクト・コンバータの機能

7.2.1 拡張空間への対応

オブジェクト・コンバータは，拡張空間に配置されたセグメントにコードが出力されているときには，空間ごとに別々の HEX 形式オブジェクト・モジュール・ファイルを生成します。

空間ごとに別々の HEX ファイルを出力するには，リンク・ディレクティブ・ファイルにおいて，memory , merge ディレクティブ両方に，空間の指定を行います。リンク・ディレクティブについては，「6.4.1 ディレクティブ・ファイル」を参照してください。

また，拡張空間に配置されたセグメントに ADDRESS 属性，または BIT 属性を持つシンボルが定義されているときには，空間ごとに別々のシンボル・テーブル・ファイルを生成します。NUMBER 属性を持つシンボルは，すべて通常空間に対するシンボル・テーブル・ファイルに出力します。

次に拡張空間に対する，HEX 形式オブジェクト・モジュール・ファイルとシンボル・テーブル・ファイルのファイル・タイプを示します。

表 7-2 拡張空間に対する出力ファイルのファイル・タイプ

ファイル	通常空間	拡張空間							
	REGULAR	EX1	EX2	EX3	EX4	…	EX13	EX14	EX15
HEX	.hex	.H1	.H2	.H3	.H4	…	.H13	.H14	.H15
シンボル	.sym	.S1	.S2	.S3	.S4	…	.S13	.S14	.S15

7.2.2 フラッシュ ROM セルフ書き換えモード対応

オブジェクト・コンバータは，フラッシュ ROM のセルフ書き換えモード使用時に，フラッシュ ROM に配置されたコードに対して，ブート領域とフラッシュ領域で別々の HEX 形式オブジェクト・モジュール・ファイルを生成できます。別々の HEX ファイルを出力するには，オブジェクト・コンバータ・オプション -ZF を指定します。ファイル・タイプは，次のようになります。

表 7-3 -ZF オプション指定時のファイル・タイプ

ファイル	ファイル・タイプ
ブート領域 ROM プログラム側の出力ファイル	.hxb
ブート領域 ROM 以外のプログラム側の出力ファイル	.hxf

7.2.3 HEX 形式オブジェクト・モジュール・ファイル

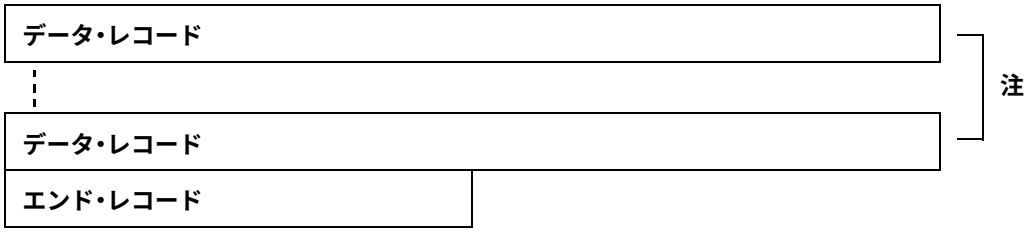
オブジェクト・コンバータの出力する HEX 形式オブジェクト・モジュール・ファイルは，PROM プログラマやディバッガなどの HEX ローダに入力可能です。

次にサンプル・プログラムの HEX 形式オブジェクト・モジュール・ファイルを示します。

```
: 0200000080007E
: 1000800011201A1620FE9A93001421FE63958462B3
: 1000900095FAFE617131809AA40073617131809A82
: 0D00A000A40072AF4D8D020D070D30AFA8
: 00000001FF
```

【インテル標準 HEX 形式オブジェクト・モジュール・ファイルのフォーマット】

図 7-2 インテル標準形式



注　データ・レコードは繰り返されます。

(1) **データ・レコード**

:	XX	XXXX	00	DD … DD	SS
↑	↑	↑	↑	↑	↑
(i)	(ii)	(iii)	(iv)	(v)	(vi)

(i) **レコード・マーク**

レコードの始まりを示します。

(ii) **コード数**（2 **桁**）

レコードに納められるコードのバイト数です。最大 16 **バイトになります。**

(iii) **ロケーション・アドレス（オフセット）**

そのレコードで表すコードの先頭アドレス（オフセット）を 16 **進** 4 **桁で示します。**

(iv) **レコード・タイプ**（2 **桁**）

00 **で固定します。**

(v) **コード（最大** 32 **桁）**

オブジェクト・コードを 1 **バイトずつ上位** 4 **ビット，下位** 4 **ビットに分けて示します。**

コードは，最大 16 **バイトまで表現されます。**

(vi) **チェック・サム**（2 **桁**）

コード数からコードまでのデータを 0 **から順に減算した値が入ります。**

(2) **エンド・レコード**

:	00	0000	01	FF
↑	↑	↑	↑	↑
(i)	(ii)	(iii)	(iv)	(v)

(i) **レコード・マーク**

(ii) **コード数。**00 **で固定です。**

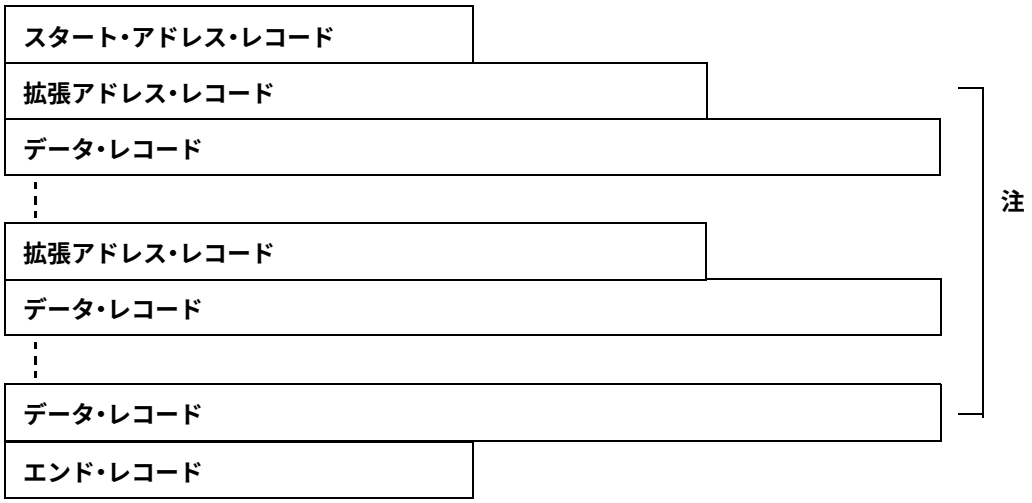
(iii) 0000 **で固定です。**

(iv) **レコード・タイプ。**01 **で固定です。**

(v) **チェック・サム。**FF **で固定です。**

【インテル拡張 HEX 形式オブジェクト・モジュール・ファイルのフォーマット】

図 7-3 インテル拡張形式



注　拡張アドレス・レコード，データ・レコードは繰り返されます。

(1) 拡張アドレス・レコード

:	02	0000	02	XXXX	SS
↑	↑	↑	↑	↑	↑
(i)	(ii)	(iii)	(iv)	(v)	(vi)

(i) レコード・マーク
レコードの始まりを示します。

(ii) コード数。02 で固定です。

(iii) 0000 で固定です。

(iv) レコード・タイプ。02 で固定です。

(v) セグメントのパラグラフ値
セグメントのパラグラフ値を 16 進 4 桁で示します。

(vi) チェック・サム（2 桁）
コード数からアドレスの上位 8 ビット値までのデータを 0 から順に減算した値が入ります。

(2) データ・レコード

:	XX	XXXX	00	DD … DD	SS
↑	↑	↑	↑	↑	↑
(i)	(ii)	(iii)	(iv)	(v)	(vi)

(i) レコード・マーク
レコードの始まりを示します。

(ii) **コード数（2 桁）**

レコードに納められるコードのバイト数です。最大 16 バイトになります。

(iii) **ロケーション・アドレス（オフセット）**

そのレコードで表すコードの先頭アドレス（オフセット）を 16 進 4 桁で示します。

(iv) **レコード・タイプ（2 桁）**

00H **で固定です。**

(v) **コード（最大 32 桁）**

オブジェクト・コードを 1 バイトずつ上位 4 ビット，下位 4 ビットに分けて示します。

コードは，最大 16 バイトまで表現されます。

(vi) **チェック・サム（2 桁）**

コード数からコードまでのデータを 0 から順に減算した値が入ります。

(3) **スタート・アドレス・レコード**

:	04	0000	03	0000	0000	F9
↑	↑	↑	↑	↑	↑	↑
(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)

(i) **レコード・マーク**

(ii) 04 **で固定です。**

(iii) 0000 **で固定です。**

(iv) 03 **で固定です。**

(v) 0000 **で固定です。**

(vi) 0000 **で固定です。**

(vii) F9 **で固定です。**

(4) **エンド・レコード**

:	00	0000	01	FF
↑	↑	↑	↑	↑
(i)	(ii)	(iii)	(iv)	(v)

(i) **レコード・マーク**

(ii) 00 **で固定です。**

(iii) 0000 **で固定です。**

(iv) 01 **で固定です。**

(v) FF **で固定です。**

【拡張テック HEX 形式オブジェクト・モジュール・ファイルのフォーマット】

ヘキサ・ファイルは，次の 3 種類のブロックから構成されます。

(i) **データ・ブロック**

(ii) **シンボル・ブロック（未使用ブロックです。シンボル情報は，シンボル・テーブル・ファイルを用います）。**

(iii) **ターミネーション・ブロック**

各ブロックは，共通した 6 文字のヘッダ・フィールドで始まり，end-of-line **で終了します。**

ブロックの最大長は，先頭の文字%と end-of-line **を含まずに** 255 **です。**

表 7-4 **に共通なヘッダ・フィールドの形式を示します。**

表 7-4 **拡張テック・ヘッダ・フィールド**

項目	ASCII キャラクタ数	説明
%	1	パーセント記号により，ブロックか拡張テック・フォーマットであることが指定されます。
ブロック長	2	ブロック内のキャラクタ数を示す 2 桁の 16 進数です。 このキャラクタ数には，先頭の%記号，および end-of-line は含まれません。
ブロックの種類	1	6 = データ・ブロック 3 = シンボル・ブロック 8 = ターミネーション・ブロック
チェック・サム	2	先頭の %，チェック・サムの桁，および end-of-line を除くブロック内の全キャラクタの値の合計を 256 で割った余りを表す 2 桁の 16 進数です。 全キャラクタの値を表 7-5 に示します。

表 7-5 **チェック・サム評価のキャラクタの値**

キャラクタ	値（10 進）
0-9 A-Z $ % .（ピリオド） _（アンダスコア） a-z	0-9 10-35 36 37 38 39 40-65

(1) **データ・ブロック**

データ・ブロックのフォーマットを表 7-6 **に示します。**

表 7-6 **拡張テックのデータ・ブロックのフォーマット**

フィールド	ASCII キャラクタ数	説明
ヘッダ	6	標準ヘッダ・フィールド ブロックの種類 = 6

表 7-6 拡張テックのデータ・ブロックのフォーマット

フィールド	ASCII キャラクタ数	説明
ロード・アドレス	2-17	オブジェクト・コードがロードされるアドレスです。 キャラクタ数可変です。
オブジェクト・コード	2n	2 桁の 16 進数として表されるバイト n 個を表します。

注意　拡張テックでは，特定のフィールドで文字数 2-17（実際のデータとしては 1-16）まで変えることができます。可変フィールドの最初の文字は 16 進数で，残りのフィールド長を示します。数字の 0 は 16 文字の文字列が続けられることを示します。したがって文字列は 1-16 文字となるため，文字列長を 1 文字加えて，可変長フィールドの長さは 2-17 となります。

```
%    15    6     1C    3     100    020202020202
↑    ↑     ↑     ↑     ↑     ↑      ↑
(i)  (ii)  (iii) (iv)  (v)   (vi)   (vii)
```

(i)　ヘッダ・キャラクタ
(ii)　ブロック長：15H = 21
(iii)　ブロックの種類：6
(iv)　チェック・サム：1CH
(v)　ロード・アドレスの桁数
(vi)　ロード・アドレス：100H
(vii)　オブジェクト・コード：6 バイト

(2)　ターミネーション・ブロック

ターミネーション・ブロックのフォーマットを表 7-7 に示します。

表 7-7 拡張テックのターミネーション・ブロックのフォーマット

フィールド	ASCII キャラクタ数	説明
ヘッダ	6	標準ヘッダ・フィールド ブロックの種類 = 8
ロード・アドレス	2-17	プログラム実行の開始アドレスです。 キャラクタ数可変です。

```
%    08    8     1A    2     80
↑    ↑     ↑     ↑     ↑     ↑
(i)  (ii)  (iii) (iv)  (v)   (vi)
```

(i)　ヘッダ・キャラクタ
(ii)　ブロック長：8H

(iii) **ブロックの種類：**8

(iv) **チェック・サム：**1AH

(v) **ロード・アドレスの桁数**

(vi) **ロード・アドレス：**80H

(3) **シンボル・ブロック（未使用）**

拡張テックのシンボル・ブロックは，シンボリック・ディバッグ用に使用されるデータであり，次に示す属性を想定しています。

シンボル自体 ： 1-16 **個の英大小文字，数字，ピリオド，およびアンダスコア。先頭文字に数字は許されません。**

値 ： 64 **ビットまで（**16 **進数** 16 **桁）可能です。**

種類 ： **アドレス，またはスカラ（スカラはアドレスを除くすべての数値を示します）。アドレスはコード・アドレス（インストラクションのアドレス）とデータ・アドレス（データ項目のアドレス）に分けられます。**

グローバル／ローカル指定 ： **シンボルがグローバル（外部参照可能）かローカルかを示します。**

セクション・メンバシップ ： **セクションは，メモリの名前の付いている範囲と考えることができます。**
プログラムの各アドレスは，必ず 1 **つにセクションに属しており，スカラはセクションに属していません。**

シンボル・ブロックの形式を表 7-8 **に示します。**

表 7-8 **拡張テックのシンボル・ブロックのフォーマット**

フィールド	ASCII キャラクタ数	説明
ヘッダ	6	標準ヘッダ・フィールド ブロックの種類 = 3
セクション名	2-17	ブロック内で定義されるシンボルを含むセクション名です。キャラクタ数は可変です。
セクション定義	5-35	このフィールドは各セクションのシンボル・ブロック 1 つによって表示されなければなりません。このフィールドは任意の数のシンボル定義フィールドの前，または後に続けることが出来ます。表 7-9 にこのフォーマットを示します。
シンボル定義	おのおの 5-35	表 7-10 に示されるゼロ以上のシンボル定義フィールドです。

プログラム内のシンボルは，シンボル・ブロックとして転送されます。各シンボル・ブロックには，セクション名，およびこのセクションの属するシンボルのリストが含まれます（必要に応じて，どのセクションにもスカラを入れることができます）。

同じセクションのシンボルを 1 **つ以上のブロックに入れることもできます。**

シンボル・ブロック中のセクション定義フィールドとシンボル定義フィールドの形式を表 7-9**，表** 7-10 **に示します。**

表 7-9 拡張テック・シンボル・ブロック・セクション定義フィールド

フィールド	ASCII キャラクタ数	説明
0	1	0 によりセクション定義フィールドであることが指定されます。
ベース・アドレス	2-17	セクション開始アドレスです。 キャラクタ数は可変です。
長さ	2-17	セクションの長さを示します。 キャラクタ数は可変で，次の式より算出されます。 1 -（上位アドレス - ベース・アドレス）

表 7-10 拡張テック・シンボル・ブロック・シンボル定義フィールド

フィールド	ASCII キャラクタ数	説明
種類	1	シンボルのグローバル／ローカルの指定，およびシンボルの示す値の種類を示す 1 桁の 16 進数です。 1 = グローバル・アドレス 2 = グローバル・スカラ 3 = グローバル・コード・アドレス 4 = グローバル・データ・アドレス 5 = ローカル・アドレス 6 = ローカル・スカラ 7 = ローカル・コード・アドレス 8 = ローカル・データ・アドレス
シンボル	2-17	シンボル長，可変です。
数値	2-17	シンボルに対応する値で，キャラクタ数は可変です。

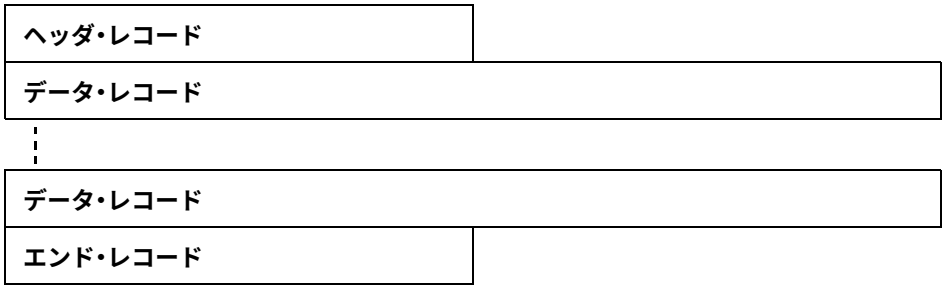
【モトローラ S タイプ形式】

変更生成されるヘキサ・ファイルは，次の 3 種類，5 レコードからなり，ファイル全体の構成は図 7-4 のようになります。レコードの種類を表 7-11 に示します。

表 7-11 モトローラ・ヘキサ・ファイルのレコードの種類

種類	レコード・タイプ
ヘッダ・レコード（オプショナル）	S0
データ・レコード	S2（スタンダード 24 ビット） S3（32 ビット）
エンド・レコード	S8（スタンダード 24 ビット） S7（32 ビット）

図 7-4 モトローラ S タイプ形式



モトローラ・ヘキサ・フォーマットは，アドレスが 24 ビットのスタンダード・アドレスと 32 ビット・アドレスに分けられますが，スタンダード・アドレスの場合は，S0 , S2 , S8 レコード，32 ビット・アドレスの場合は，S0 , S3 , S7 レコードで構成されます。なお，ヘッダ・レコード S0 はオプションであり出力されません。各 S レコードの末尾には，CR 文字が置かれます。

各レコードのフィールドの一般形式とその意味を，表 7-12，表 7-13 に示します。

表 7-12 各レコードの一般形

レコード・タイプ	一般形
S0	S0XXYY … YYZZZZ
S2	S2XXWWWWWWDD … DDZZ
S3	S3XXWWWWWWWWDD … DDZZ
S7	S7XXWWWWWWWWZZ
S8	S8XXWWWWWWZZ

表 7-13 フィールドの意味

フィールド	意味
Sn	レコード・タイプ
XX	データ・レコード長 アドレスと 16 進データとチェック・サムのバイト数です。
YY … YY	ファイル名 入力ファイル名の ASCII コードを 16 進数で表現したものです。
WWWWWW [WW]	24 [32] ビット目アドレス
DD … DD	16 進データ データ 1 バイトを 2 桁の 16 進数で表現したものです。
ZZ	チェック・サム レコード長，アドレス，および 16 進データのバイトごとの和の 1 の補数の下位 1 バイトを 2 桁の 16 進数で表現したものです。

```
S2   08     00FF11  D4520A20  A0
↑    ↑      ↑       ↑         ↑
(i)  (ii)   (iii)   (iv)      (v)
```

(i) **レコード・タイプ：**S2

(ii) **レコード長：**8 **バイト**

(iii) **ロード・アドレス（**24 **ビット・アドレス）**

(iv) 16 **進データ**

(v) **チェック・サム**

(1) S0 **レコード**

```
S0   XX   YYYYYYYY   ZZ
↑    ↑    ↑         ↑
(i)  (ii) (iii)     (iv)
```

(i) **レコード・タイプ**

(ii) **レコード長**

(iii) **のバイト数** + (iv) **のバイト数です。**

(iii) **ファイル名**

(iv) **チェック・サム**

(2) S2 **レコード**

```
S2   XX   WWWWWW   DD … DD   ZZ
↑    ↑    ↑       ↑         ↑
(i)  (ii) (iii)   (iv)      (v)
```

(i) **レコード・タイプ**

(ii) **レコード長**

(iii) **のバイト数** + (iv) **のバイト数** + (v) **のバイト数です。**

(iii) **ロード・アドレス**

(iv) **のデータのロード・アドレスであり，**24 **ビットで** 0H-FFFFFFH **までの範囲です。**

(iv) **データ**

ロードされるデータそのものです。

(v) **チェック・サム**

(3) S3 **レコード**

S3	XX	WWWWWWWW	DD … DD	ZZ
↑	↑	↑	↑	↑
(i)	(ii)	(iii)	(iv)	(v)

(i) **レコード・タイプ**

(ii) **レコード長**

(iii) **のバイト数** + (iv) **のバイト数** + (v) **のバイト数です。**

(iii) **ロード・アドレス**

(iv) **のデータのロード・アドレスであり，**32 **ビットで** 0H-FFFFFFFFH **までの範囲です。**

(iv) **データ**

ロードされるデータそのものです。

(v) **チェック・サム**

(4) S7 **レコード**

S7	XX	WWWWWWWW	ZZ
↑	↑	↑	↑
(i)	(ii)	(iii)	(iv)

(i) **レコード・タイプ**

(ii) **レコード長**

(iii) **のバイト数** + (iv) **のバイト数です。**

(iii) **エントリ・アドレス**

エントリ・アドレスであり，32 **ビットで** 0H-FFFFFFFFH **までの範囲です。**

(iv) **チェック・サム**

(5) S8 **レコード**

S8	XX	WWWWWWWW	ZZ
↑	↑	↑	↑
(i)	(ii)	(iii)	(iv)

(i) **レコード・タイプ**

(ii) **レコード長**

(iii) **のバイト数** + (iv) **のバイト数です。**

(iii) **エントリ・アドレス**

エントリ・アドレスであり，24 **ビットで** 0H-FFFFFFH **までの範囲です。**

(iv) **チェック・サム**

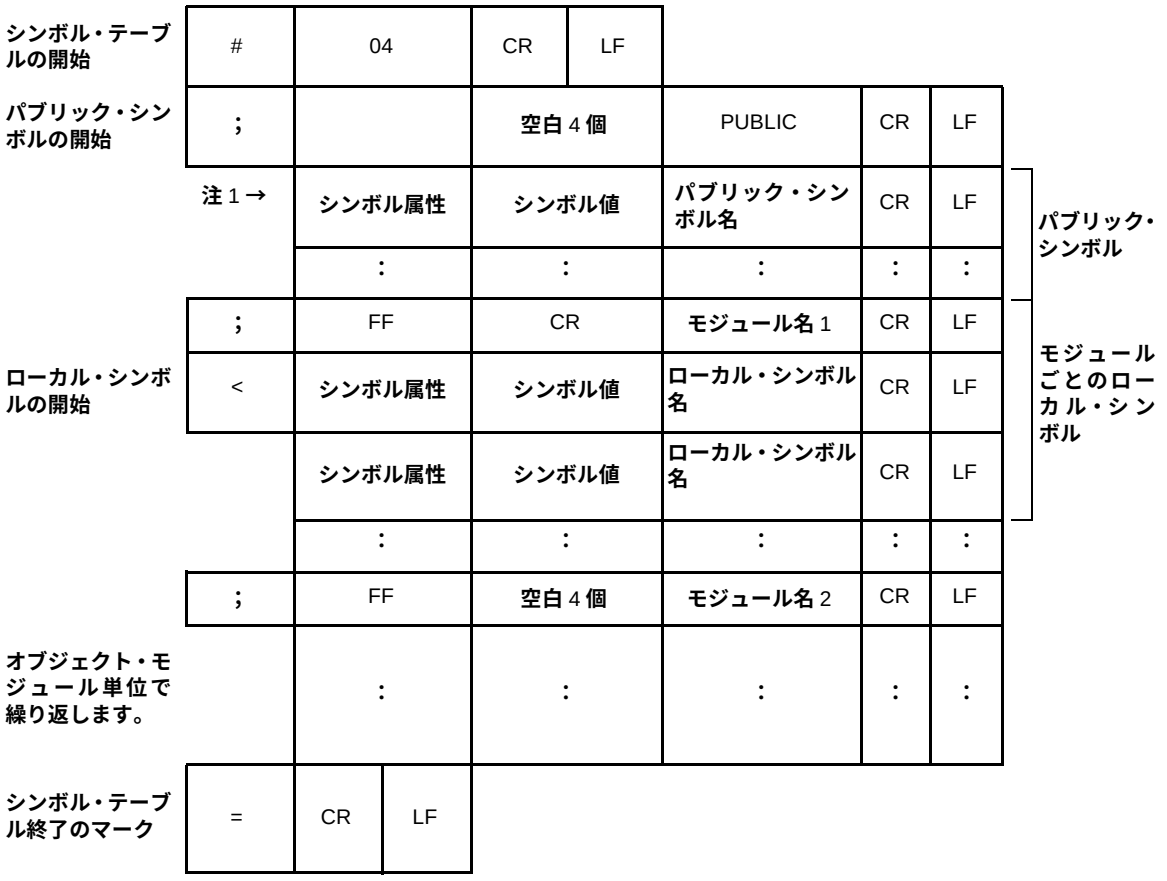
7.2.4 シンボル・テーブル・ファイル

オブジェクト・コンバータの出力するシンボル・テーブル・ファイルは，ディバッガへの入力となります。

次にサンプル・プログラムのシンボル・テーブル・ファイルを示します。

```
#04
; FF       PUBLIC
010097CONVAH
010000MAIN
010080START
00FE20_@STBEG
00FB00_@STEND
; FF       SAMPM
<02FE20HDTSA
02FE21STASC
; FF       SAMPS
<0100A8SASC
0100AESASC1
=
```

【シンボル・テーブル・ファイルのフォーマット】



注 1　シンボル属性は，表 7-14 の値をとります。

注 2　シンボル値は図 7-5 を参照してください。

表 7-14 シンボル属性の値

値	シンボル属性
00	EQU で定義した定数
01	コード・セグメント内のレーベル
02	データ・セグメント内のレーベル
03	ビット・シンボル
FF	モジュール名

図 7-5 シンボル値のフォーマット

- シンボル属性が NUMBER のとき

定数値　4 桁

- シンボル属性が LABEL のとき

アドレス値　4 桁

- シンボル属性がビット・シンボルのとき

上位 13 ビット	下位 3 ビット

上位 13 ビット　：0FE00H からの相対アドレス

下位 3 ビット　：ビット位置（0 ～ 7）

7.3 オブジェクト・コンバータの起動方法

7.3.1 オブジェクト・コンバータの起動

オブジェクト・コンバータの起動には，次の 2 つの方法があります。

(1) コマンド行での起動

```
X>[ パス名 ] oc78k0 [ △オプション ] …ロード・モジュール・ファイル名 [ △オプション ] … [ △ ]
↑    ↑      ↑          ↑                          ↑                                  ↑
(a)  (b)    (c)        (d)                        (e)                                (d)
```

(a) カレント・ドライブ名
(b) カレント・ディレクトリ名
(c) オブジェクト・コンバータのコマンド・ファイル名
(d) オブジェクト・コンバータに対して動作の詳細を指定
(e) コンバートするロード・モジュール・ファイル名

例 C>oc78k0 k0.lmf -osamle.hex

注意 複数のオブジェクト・コンバータ・オプションを指定する場合は，それぞれのオブジェクト・コンバータ・オプション間を空白で区切ってください。オブジェクト・コンバータ・オプションの詳細については，「7.4 オブジェクト・コンバータ・オプション」を参照してください。

(2) パラメータ・ファイルによる起動

パラメータ・ファイルは，起動に必要な情報がコマンド行に指定しきれない場合やオブジェクト・コンバートするたびに同じオプションを繰り返し指定するような場合に使用します。

パラメータ・ファイルを使用する場合には，コマンド行にパラメータ・ファイル指定オプション（-F）を指定します。

パラメータ・ファイルによる起動方法を次に示します。

```
X>oc78k0 [ △ロード・モジュール・ファイル ] △ -f パラメータ・ファイル名
                                              ↑          ↑
                                             (a)        (b)
```

(a) パラメータ・ファイル指定オプション
(b) オブジェクト・コンバータの起動に必要な情報を含んだファイル

備考 パラメータ・ファイルはエディタなどで作成します。

パラメータ・ファイル内での記述規則を次に示します。

```
[ [ [ △ ] オプション [ △オプション ] … [ △ ] △ ] ] …
```

備考 1 コマンド行でロード・モジュール・ファイル名を省略した場合，パラメータ・ファイル内でロード・

モジュール・ファイル名を 1 つだけ指定できます。

備考 2 ロード・モジュール・ファイル名は，オプションのあとでも記述できます。

備考 3 パラメータ・ファイルには，コマンド行で指定すべきすべてのオブジェクト・コンバータ・オプション，出力ファイル名を記述します。

例　　パラメータ・ファイル（k0.poc）をエディタで作成します。

＜ k0.poc の内容＞

```
; parameter file
k0.lmf -osample.hex
-ssample.sym -r
```

パラメータ・ファイル（k0.poc）を使用してオブジェクト・コンバータを起動します。

```
C>oc78k0 –fk0.poc
```

7.3.2　実行開始メッセージ，終了メッセージ

(1) 実行開始メッセージ

オブジェクト・コンバータが起動すると，次の実行開始メッセージを表示します。

```
78K/0 Series Object Converter Vx.xx [ xx xxx xx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx
```

(2) 実行終了メッセージ

オブジェクト・コンバートの結果，エラーが検出されなかった場合には，オブジェクト・コンバータは，次のメッセージを表示して制御を OS に戻します。

```
 Target chip : uPD780xxx
 Device file : Vx.xx

Object Conversion Complete ,       0 error ( s ) and       0 warning ( s ) found.
```

オブジェクト・コンバートの結果，エラーが検出された場合は，オブジェクト・コンバータはエラーの数を表示して制御を OS に戻します。

```
Target chip : uPD780xxx
 Device file : Vx.xx

Object Conversion Complete ,       3 error ( s ) and       0 warning ( s ) found.
```

オブジェクト・コンバート中に，オブジェクトコンバータの処理継続が不可能な致命的エラーが検出された場合，オブジェクト・コンバータはメッセージを表示してオブジェクト・コンバートを中止し，制御を OS に戻します。

例 1　存在しないロード・モジュール・ファイル名を指定した場合

```
C>oc78k0 sample.lmf

78K/0 Series Object Converter Vx.xx [ xx xxx xx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

RA78K0 error F4006 : File not found ' SAMPLE.LMF '
Program aborted.
```

この例では，存在しないロード・モジュール・ファイルを指定したためにエラーとなり，オブジェクト・コンバートが中止されました。

例 2　存在しないオブジェクト・コンバータ・オプションを指定した場合

```
C>oc78k0 k0.lmf -a

78K/0 Series Object Converter Vx.xx [ xx xxx xx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

RA78K0 error F4018 : Option is not recognized '-a'
Please enter ' OC78K0 -- ' , if you want help messages.
Program aborted.
```

この例では，存在しないオブジェクト・コンバータ・オプションを指定したために，エラーとなり，オブジェクト・コンバートが中止されました。
オブジェクト・コンバータがエラー・メッセージを出力して，オブジェクト・コンバートを中止した場合そのエラー・メッセージの原因を「第 12 章 エラー・メッセージ」で調べて対処してください。

7.4 オブジェクト・コンバータ・オプション

7.4.1 オブジェクト・コンバータ・オプションの種類

オブジェクト・コンバータ・オプションは，オブジェクト・コンバータの動作に細かい指示を与えるものです。オブジェクト・コンバータ・オプションは，10 種類のオプションに分類できます。

次にオブジェクト・コンバータ・オプションの分類と説明を示します。

表 7-15 オブジェクト・コンバータ・オプション

分類	オプション	説明
HEX 形式オブジェクト・モジュール・ファイル出力指定	-O	HEX 形式オブジェクト・モジュール・ファイルの出力を指定します。
	-NO	
シンボル・テーブル・ファイル出力指定	-S	シンボル・テーブル・ファイルの出力を指定します。
	-NS	
オブジェクト・アドレス順ソート指定	-R	HEX 形式オブジェクトをアドレス順にソートします。
	-NR	
オブジェクト充てん値指定	-U	HEX形式オブジェクトが出力されないアドレス領域に対して，指定した充てん値をオブジェクト・コードとして出力します。
	-NU	
エラー・リスト・ファイル出力指定	-E	エラー・リスト・ファイルを出力します。
	-NE	
パラメータ・ファイル指定	-F	入力ファイル名，オプションを指定したファイルより入力します。
HEX 形式指定	-KI	インテル標準 HEX 形式
	-KIE	インテル拡張 HEX 形式
	-KT	拡張テック形式
	-KM	モトローラ S タイプ形式（24 ビット・スタンダード・アドレス）
	-KME	モトローラ S タイプ形式（32 ビット・アドレス）
デバイス・ファイル・サーチ・パス指定	-Y	デバイス・ファイルを指定されたパスから読み込みます。
フラッシュ ROM 内蔵製品用ファイル分割出力指定	-ZF	- フラッシュ ROM 内蔵製品のブート領域 ROM プログラムのリンク指定時において，ブート領域とそれ以外の領域を別の HEX フォーマット・ファイルに分割出力するオプションを追加します。 - -ZF オプションが指定された場合，ブート領域 ROM プログラム側の出力ファイル・タイプは “HXB”，それ以外のプログラム側の出力ファイル・タイプは “HXF” になります。
ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

7.4.2 オブジェクト・コンバータ・オプションの説明

次ページ以降に，各オブジェクト・コンバータ・オプションの詳細について説明します。

(1) HEX 形式オブジェクト・モジュール・ファイル出力指定

HEX 形式オブジェクト・モジュール・ファイル出力指定 （-O/-NO）

【記述形式】

-O [出力ファイル名] -NO

- 省略時解釈

  -O 入力ファイル名 .hex

  （拡張空間に対してのファイル・タイプは，‘ .H1 ’ ～ ‘ .H15 ’ ）

【機能】

- -O オプションは，HEX 形式オブジェクト・モジュール・ファイルの出力を指定します。
  また，その出力先や出力ファイル名を指定します。
- -NO オプションは，HEX 形式オブジェクト・モジュール・ファイルを出力しないことを指定します。

【用途】

- HEX 形式オブジェクト・モジュール・ファイルの出力先や出力ファイル名を変更したいときに，-O オプションを指定します。
- シンボル・テーブル・ファイルの出力のみが目的でオブジェクト・コンバートする場合などに，-NO オプションを指定します。これによりオブジェクト・コンバート時間が短縮されます。

【説明】

- “ 出力ファイル名 ” には，ディスク型ファイル名を指定してください。
  デバイス型ファイル名を指定すると，アボート・エラーとなります。
- -O オプションを指定する際に “ 出力ファイル名 ” を省略すると，カレント・ディレクトリに HEX 形式オブジェクト・モジュール・ファイル（“ 入力ファイル名 .hex ”）が出力されます。
- “ 出力ファイル名 ” にパス名のみを指定すると，指定したパスに “ 入力ファイル名 .hex ” が出力されます。
- -O と -NO の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -ZF オプションが指定された場合，次に示すファイル・タイプになります。

表 7-16 -ZF オプション指定時のファイル・タイプ

ファイル	ファイル・タイプ
ブート領域 ROM プログラム側の出力ファイル	.hxb
ブート領域 ROM 以外のプログラム側の出力ファイル	.hxf

オブジェクト・コンバータは，拡張空間に配置されたセグメントにコードが出力されているときには，空間ごとに別々の HEX 形式オブジェクト・モジュール・ファイルを生成します。

次に拡張空間に対する HEX 形式オブジェクト・モジュール・ファイルのファイル・タイプを示します。

表 7-17 拡張空間に対する HEX 形式オブジェクト・モジュール・ファイルのファイル・タイプ

ファイル	通常空間	拡張空間								
	REGULAR	EX1	EX2	EX3	EX4	EX5	…	EX13	EX14	EX15
HEX	.hex	.H1	.H2	.H3	.H4	.H5	…	.H13	.H14	.H15

【使用例】

- HEX 形式オブジェクト・モジュール・ファイル（sample.hex）を出力します。

```
C>oc78k0 k0.lmf -osample.hex
```

(2) シンボル・テーブル・ファイル出力指定

シンボル・テーブル・ファイル出力指定 （-S/-NS）

【記述形式】

```
-S [ 出力ファイル名 ]
-NS
```

- **省略時解釈**

  -S **入力ファイル名** .sym

  （拡張空間に対してのファイル・タイプは，‘.S1’ **～** ‘.S15’**）**

【機能】

- -S **オプションは，シンボル・テーブル・ファイルの出力を指定します。また，その出力先や出力ファイル名を指定します。**
- -NS **オプションは，シンボル・テーブル・ファイルを出力しないことを指定します。**

【用途】

- **シンボル･テーブル･ファイルの出力先や出力ファイル名を変更したいときに,**-S **オプションを指定します。**
- HEX **形式オブジェクト・モジュール・ファイルの出力のみが目的でオブジェクト・コンバートする場合などに，**-NS **オプションを指定します。**

  -NS **オプションの指定により，オブジェクト・コンバート時間が短縮されます。**

【説明】

- **“ 出力ファイル名 ” には，ディスク型ファイル名を指定してください。**

  デバイス型ファイル名を指定した場合，アボート・エラーとなります。
- -S **オプションを指定する際に“ 出力ファイル名 ”を省略すると，カレント・ディレクトリにシンボル・テーブル・ファイル（‘ 入力ファイル名** .sym **’）が出力されます。**
- **“ 出力ファイル名 ” にパス名のみを指定すると，指定したパスに “ 入力ファイル名** .sym **” が出力されます。**
- -S **と** -NS **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**

拡張空間に配置されたセグメントに ADDRESS **属性，または** BIT **属性を持つシンボルが定義されている場合，空間ごとに別々のシンボル・テーブル・ファイルを生成します。**

NUMBER **属性を持つシンボルは，すべて通常空間に対するシンボル・テーブル・ファイルに出力します。**

次に拡張空間に対するシンボル・テーブル・ファイルの，ファイル・タイプを示します。

表 7-18 **拡張空間に対するシンボル・テーブル・ファイルのファイル・タイプ**

ファイル	通常空間	拡張空間								
	REGULAR	EX1	EX2	EX3	EX4	EX5	…	EX13	EX14	EX15
HEX	.hex	.S1	.S2	.S3	.S4	.S5	…	.S13	.S14	.S15

【使用例】

- **シンボル・テーブル・ファイル**（sample.sym）**を出力します。**

```
C>oc78k0 k0.lmf -ssample.sym
```

(3) オブジェクト・アドレス順ソート指定

オブジェクト・アドレス順ソート指定 （-R/-NR）

【記述形式】

```
-R
-NR
```

- **省略時解釈**

  -R

【機能】

- -R **オプションは，**HEX **形式オブジェクトをアドレス順にソートして出力します。**
- -NR **オプションは，**HEX **形式オブジェクトをロード・モジュール・ファイルに格納されている順に出力します。**

【用途】

- HEX **形式オブジェクトがアドレス順にソートされる必要のない場合に，**-NR **オプションを指定します。**

【説明】

- -R **と** -NR **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**
- -NO **オプションを指定した場合，**-R/-NR **オプションは無効となります。**

【使用例】

- HEX **形式オブジェクトをアドレス順にソートします。**

```
C>oc78k0 k0.lmf -r
```

(4) オブジェクト充てん値指定

オブジェクト充てん値指定 （-U/-NU）

【記述形式】

-U **充てん値** [, [**スタート**] , **サイズ**] -NU

- **省略時解釈**

  -U0FFH（0FFH **を充てんします**）

【機能】

- -U **オプションは，**HEX **形式オブジェクトが出力されないアドレス領域に対して，指定した充てん値をオブジェクト・コードとして出力します。**
- -NU **オプションは，**-U **オプションを無効にします。**

【用途】

- HEX **形式オブジェクトが出力されていないアドレス領域には，不要なコードが書き込まれてしまうことがあります。このようなアドレスをプログラムが何らかの原因でアクセスした場合，プログラムの動作は予測できません。このため，あらかじめ，**-U **オプションを指定して** HEX **形式オブジェクトが出力されていないアドレス領域にコードを書き込んでおきます。**

【説明】

- **充てん値として指定できる値の範囲は，次のとおりです。**

  0H ≦ **充てん値** ≦ 0FFH

  2 **進，**8 **進，**10 **進数字，または** 16 **進数字で指定します。範囲外の数値，または数値以外を指定した場合は，アボート・エラーとなります。**
- **スタートには，充てんを行うアドレス領域の先頭アドレスを指定します。**
  指定できる値の範囲は，次のとおりです。

  0H ≦ **スタート** ≦ SFR **領域を除くプログラム空間の最大アドレス**

  2 **進，**8 **進，**10 **進数字，または** 16 **進数字で指定します。範囲外の数値，または数値以外を指定した場合には，アボート・エラーとなります。また，スタートを省略した場合には，**0 **を指定したものとみなします。**
- **サイズには，充てんを行うアドレス領域のサイズを指定します。指定できる値の範囲は，次のとおりです。**

  1H ≦ **サイズ** ≦ SFR **領域を除くプログラム空間の最大アドレス** + 1

  2 **進，**8 **進，**10 **進数字，または** 16 **進数字で指定します。範囲外の数値，または数値以外を指定した場合には，アボート・エラーとなります。また，スタートを指定している場合には，サイズを省略できません。**
- **スタートとサイズの両方の指定を省略した場合，オブジェクト・コンバータは内蔵** ROM **の範囲が指定されたものとみなして処理を行います。**
- -U **と** -NU **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**

- -NO オプションを指定した場合は，-U/-NU オプションは無効になります。
- -U オプションで，複数のアドレス範囲を指定することはできません。
- -U オプションでの，スタート，サイズの指定形式とその解釈は次のようになります。

(a) -U 充てん値

：対象デバイスに内蔵 ROM がある場合は，内蔵 ROM の範囲

(b) -U 充てん値,,サイズ

：0 番地からサイズ - 1 番地まで

(c) -U 充てん値，スタート，サイズ

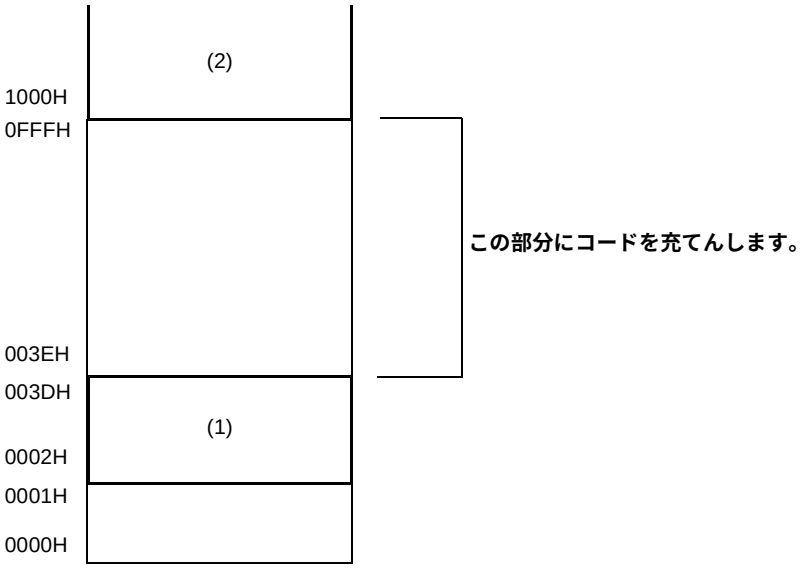
：スタート番地から，スタート番地 + サイズ - 1 番地まで

【使用例】

- HEX 形式オブジェクトが出力されていないアドレス領域にコードを充てんします。

次のような HEX 形式オブジェクト・モジュール・ファイルがあると仮定します。この場合，003EH-0FFFH のアドレス領域にはコードが書き込まれていません。

```
: 020000000200FC
: 100002002B41000BFC80FE2B40000944F7083A20EC        ; (1)
: 100012001A6720FE2822006521FED350D25014FE1A        ; (1)
: 10002200B900059F2835002431B900059F28350005        ; (1)
: 0C003200242156AF0A8302A807A830560C
: 01000003B5D0d0026A3…                              ; (2)
: 1010100024A5F622B667…                             ; (2)
   :
: 00000001FF
```



003EH-0FFFH のアドレス領域に，00H を充てんします。

C>oc78k0 k0.lmf -u00h , 003eh , 0fc2h

(5) エラー・リスト・ファイル出力指定

エラー・リスト・ファイル出力指定 （-E/-NE）

【記述形式】

```
-E [ 出力ファイル名 ]
-NE
```

- **省略時解釈**

  -NE

【機能】

- -E **オプションは，エラー・リスト・ファイルの出力を指定します。**
  また，その出力先や出力ファイル名を指定します。
- -NE **オプションは，**-E **オプションを無効にします。**

【用途】

- **エラー・リスト・ファイルの出力先や出力ファイル名を変更したいときに，**-E **オプションを指定します。**

【説明】

- **ファイル名として，ディスク型ファイル名とデバイス型ファイル名を指定できます。ただし，デバイス型ファイル名として** CLOCK **を指定した場合は，アボート・エラーとなります。**
- -E **オプションを指定する際に出力ファイル名を省略するとエラー・リスト・ファイル名は，“ 入力ファイル名** .eoc **” となります。**
- -E **オプションを指定する際にドライブ名を省略するとカレント・ドライブにエラー・リスト・ファイルが出力されます。**
- -E **と** -NE **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**

【使用例】

- **エラー・リスト・ファイル（**k0.eoc**）を作成します。**

```
C>oc78k0 k0.lmf –ek0.eoc
```

(6) パラメータ・ファイル指定

パラメータ・ファイル指定 （-F）

【記述形式】

-F ファイル名

- 省略時解釈

  コマンド行上からのみオプション，または入力ファイル名の入力が可能

【機能】

- -F オプションは，オプション，または入力ファイル名を指定のファイルから入力する指定をします。

【用途】

- コマンド行ではオブジェクト・コンバータの起動に必要な情報を指定しきれないときに，-F オプションを指定します。
- オブジェクト・コンバートするたびに繰り返し同じようにオプションを指定する場合には，それらをパラメータ・ファイルに記述しておいて，-F オプションで指定します。

【説明】

- “ ファイル名 ” として指定できるのは，ディスク型ファイル名のみです。
  デバイス型ファイル名を指定するとアボート・エラーとなります。
- ファイル名を省略するとアボート・エラーとなります。
- パラメータ・ファイルのネストは許されません。パラメータ・ファイル中で，-F オプションを指定するとアボート・エラーとなります。
- パラメータ・ファイル中に記述できる文字数は制限はありません。
- 空白とタブ，および改行文字（LF）をオプションあるいは入力ファイル名の区切りとします。
- パラメータ・ファイル中に記述したオプション，または入力ファイル名は，コマンド行上のパラメータ・ファイル指定のあった位置に展開されます。
- 展開されたオプションは，あとで指定した方が有効となります。
- “ ; ”,または‘ # ’以降に記述された文字は改行文字(LF),またはEOFの前まですべてコメントと解釈します。
- -F オプションを複数指定するとアボート・エラーとなります。

【使用例】

- パラメータ・ファイルを使用してオブジェクト・コンバートします。

＜パラメータ・ファイル 78k0.poc の内容＞

```
; parameter file
k0.lmf -osample.hex
-ssample.sym -r
```

コマンド行には，次のように入力します。

```
C>oc78k0 k0.lmf -f78k0.poc
```

(7) HEX 形式指定

HEX 形式指定 （-KI/-KIE/-KT/-KM/-KME）

【記述形式】

```
-KI
-KIE
-KT
-KM
-KME
```

- **省略時解釈**

  -KIE

【機能】

- **出力する** HEX **形式オブジェクト・モジュール・ファイルの形式を指定します。**

【用途】

- **出力する** HEX **形式オブジェクト・モジュール・ファイルの形式を「インテル標準** HEX **形式」，「インテル拡張** HEX **形式」，「拡張テック形式」，「モトローラ** S **タイプ形式（スタンダード・アドレス）」，「モトローラ** S **タイプ形式（**32 **ビット・アドレス）」の中から指定します。**

【説明】

- **各オプションは，次のように指定します。**

  -KI **：インテル標準** HEX **形式**

  0H-FFFFH（64 K **バイトまで）**

  -KIE **：インテル拡張** HEX **形式**

  0H-FFFFFH（1 M **バイトまで）**

  -KT **：拡張テック形式**

  0H-FFFFFFFFH（4 G **バイトまで）**

  -KM **：モトローラ** S **タイプ形式（スタンダード・アドレス）**

  0H-FFFFFFH（16 M **バイトまで）**

  -KME **：モトローラ** S **タイプ形式（**32 **ビット・アドレス）**

  0H-FFFFFFFFH（4 G **バイトまで）**

【使用例】

- **モトローラ** S **タイプ形式（スタンダード・アドレス）オブジェクトを指定します。**

```
C>oc78k0 k0.lmf -km
```

(8) デバイス・ファイル・サーチ・パス指定

デバイス・ファイル・サーチ・パス指定 （-Y）

【記述形式】

-Y パス名

- 省略時解釈

  デバイス・ファイルを読み込むパスは，次の順序で調べ決定します。

  (i) < ..\dev >（oc78k0.exe の起動されたパスに対して）

  (ii) OC78K0 の起動されたパス

  (iii) カレント・ディレクトリ

  (iv) 環境変数 PATH

【機能】

- デバイス・ファイルを指定されたパスから読み込みます。

【用途】

- デバイス・ファイルが存在するパスを指定します。

【説明】

- -Y オプションに続けてパス名以外が指定された場合，アボート・エラーとなります。
- -Y オプションに続けて指定するパス名が省略された場合，アボート・エラーとなります。
- デバイス・ファイルを読み込むパスは，次の順序で調べ決定します。

  (i) -Y オプションで指定されたパス

  (ii) < ..\dev >（oc78k0.exe の起動されたパスに対して）

  (iii) OC78K0 の起動されたパス

  (iv) カレント・ディレクトリ

  (v) 環境変数 PATH

【使用例】

- デバイス・ファイルのパスを c:\78k0\dev ディレクトリに指定します。

```
C>oc78k0 k0.lmf -yc:\78k0\dev
```

(9) フラッシュ ROM 内蔵製品用ファイル分割出力指定

フラッシュ ROM 内蔵製品用ファイル分割出力指定 （-ZF）

【記述形式】

-ZF

- 省略時解釈

  分割出力しない

【機能】

- フラッシュ ROM 領域の先頭アドレスを指定します。

【説明】

- フラッシュ ROM 内蔵製品のブート領域 ROM プログラムのリンク指定時において，ブート領域とそれ以外の領域を別の HEX フォーマット・ファイルに分割出力するオプションを追加します。
- -ZF オプションが指定された場合，ブート領域 ROM プログラム側の出力ファイル・タイプは “HXB”，それ以外のプログラム側の出力ファイル・タイプは “HXF” になります。

  注意　フラッシュ ROM 領域セルフ書き換え機能を持たないデバイスでは，このオプションを使用しないでください。

【使用例】

```
C>oc78k0 k0.lmf -zf
```

(10) ヘルプ指定

ヘルプ指定 (--)

【記述形式】

```
--
```

- 省略時解釈

  表示しない

【機能】

- -- オプションは，ヘルプ・メッセージをディスプレイに出力します。

【用途】

- ヘルプ・メッセージは，オブジェクト・コンバータ・オプションとその説明の一覧です。
  オブジェクト・コンバータを実行するときに参照してください。

【説明】

- -- オプションを指定すると，他のオブジェクト・コンバータ・オプションは，すべて無効となります。

  注意 このオプションは，PM plus 上では指定できません。
  PM plus 上でヘルプを参照する場合は，〈オブジェクトコンバータオプションの設定〉ダイアログで［ヘルプ］ボタンをクリックしてください。

【使用例】

- -- オプションを指定するとヘルプ・メッセージがディスプレイに出力されます。

```
C>oc78k0 --

78K/0 Series Object Converter Vx.xx [ xx xxx xx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

usage : oc78k0 [ option [ ... ] ] input-file [ option [ ... ] ]
The option is as follows ( [ ] means omissible ).
-ffile              :Input option or input-file name from specified file.
-o [ file ] / -no   :Create HEX module file [ with specified name ] / Not.
-s [ file ] / -ns   :Create symbol table file [ with specified name ] / Not.
-r/-nr              :Sort HEX object by address / Not.
-uvalue [ , [ start ] , size]/-nu :Fill up HEX object with specified value / Not.
-kkind              : Select hex format.I ; intel format IE ; intel extend format T ; tex format M
                      ; s format ME ; s-32bit format
-ydirectory         : Set device file search path.
-zf                 : Create boot hex module file ( HXB ) , and flash hex module file ( HXF ).
--                  : Show this message.
DEFAULT ASSIGNMENT : -o -s –r –u0ffh
```

7.5 PM plus でのオプション設定

PM plus からオブジェクトコンバータ・オプションを設定する方法について説明します。

7.5.1 オプションの設定方法

PM plus の [ツール (T)] メニューの [オブジェクトコンバータオプションの設定 (O)] を選択するか，ツール・バーの［OC］ボタンを押下すると，〈オブジェクトコンバータオプションの設定〉ダイアログが現れます。

ダイアログ内で必要なオプションを入力することにより，各オブジェクトコンバータ・オプションを設定できます。

図 7-6 〈オブジェクトコンバータオプションの設定〉ダイアログ（《出力 1》タブ選択時）

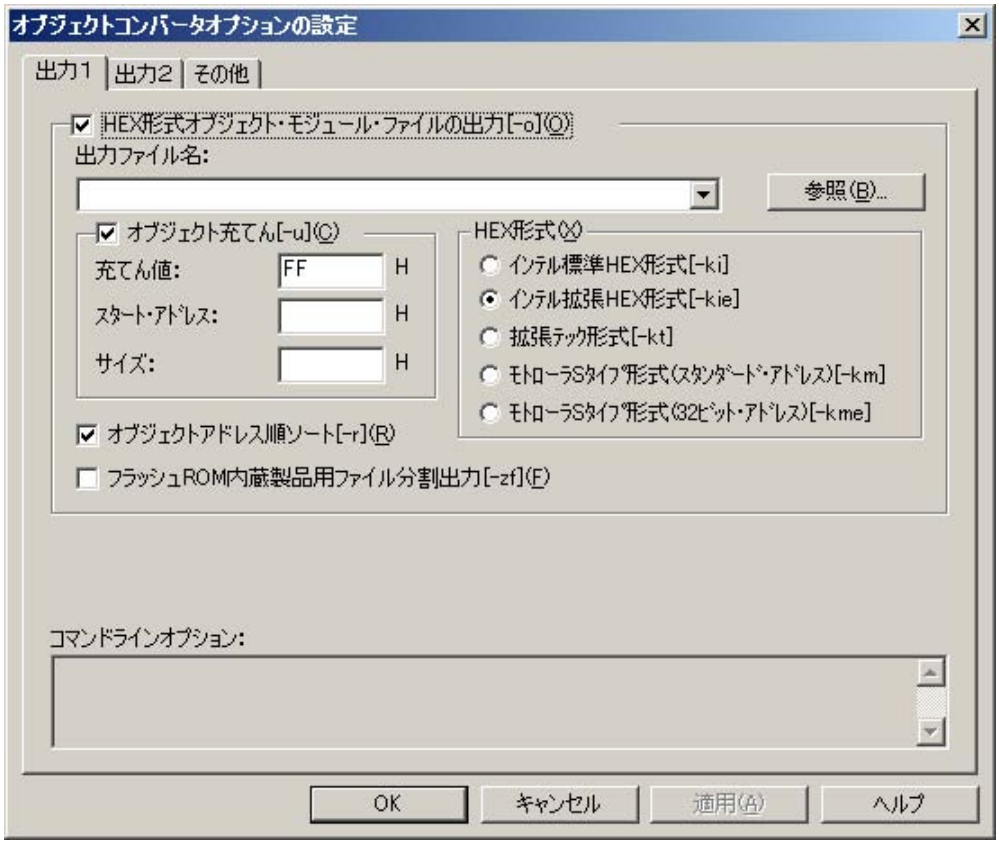


図 7-7 〈オブジェクトコンバータオプションの設定〉ダイアログ（《出力 2》タブ選択時）

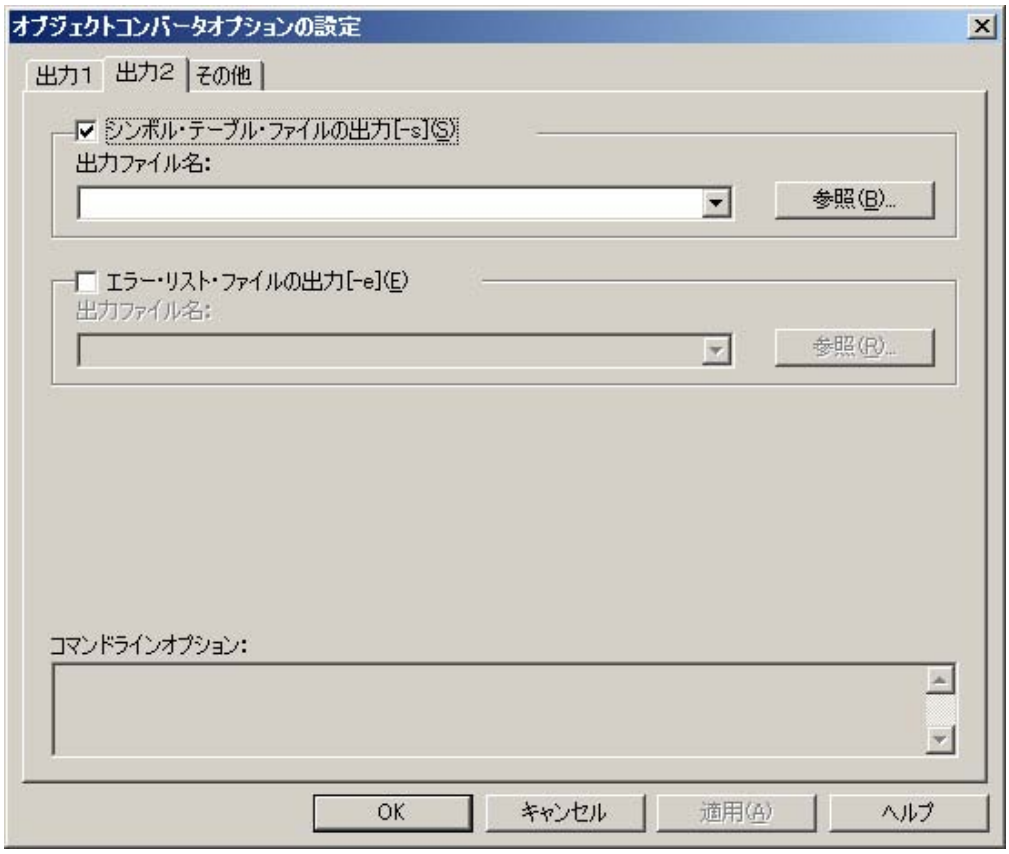
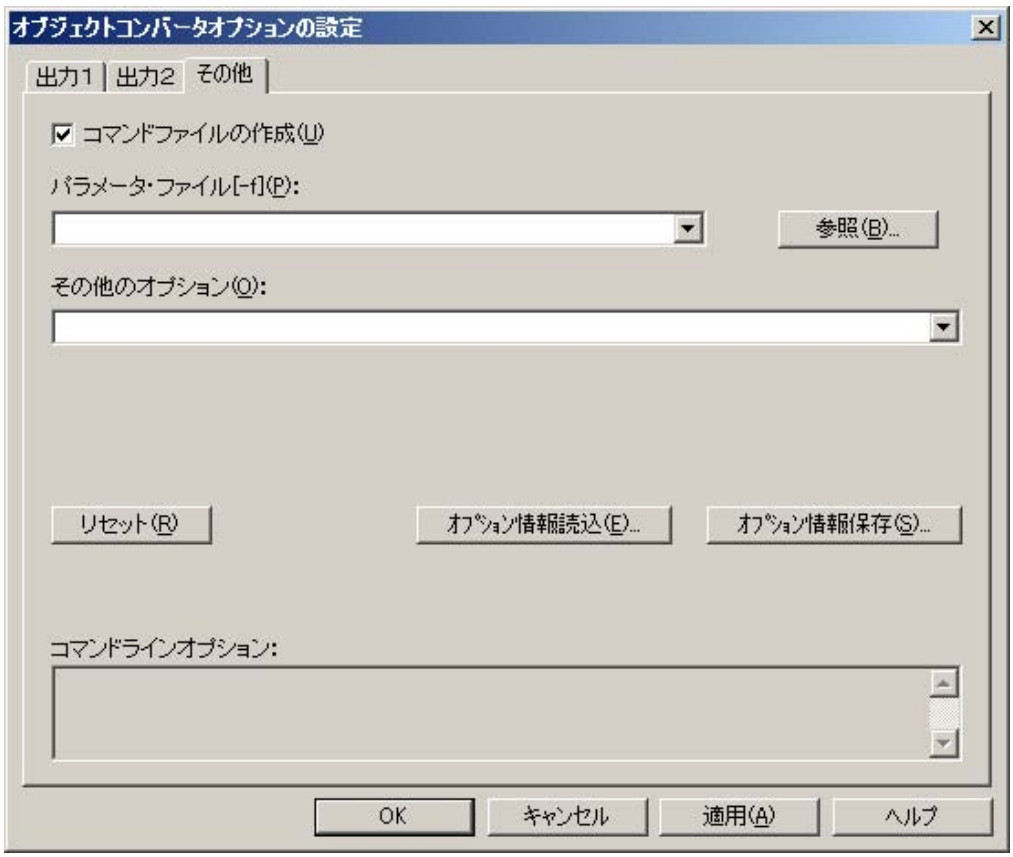


図 7-8 〈オブジェクトコンバータオプションの設定〉ダイアログ（《その他》タブ選択時）



7.5.2 各オプションの設定

〈オブジェクトコンバータオプションの設定〉ダイアログの各オプションについて，次に説明します。

《出力 1》タブ

- HEX 形式オブジェクト・モジュール・ファイルの出力 [-o](O)

  HEX 形式オブジェクト・モジュール・ファイルを出力する場合に，チェックします。

  出力ファイル：

  [参照 (B)] ボタン，または直接入力により，形式オブジェクト・モジュール・ファイルのパスとファイル名を指定します。

- オブジェクト充てん [-u](C)

  HEX 形式オブジェクトが出力されていないアドレスに不要なコードが書き込まれプログラムが暴走するのを防ぐためにあらかじめコードを書き込む場合に指定します。

- HEX 形式 (X)

  出力するオブジェクトの HEX 形式を選択します。

  注意　メモリバンク機能を持たないデバイスでは，本オプションは指定できません。

- オブジェクトアドレス順ソート [-r](R)

  HEX 形式オブジェクトがアドレス順にソートされている必要がある場合に，チェックします。

- フラッシュ ROM 内蔵製品用ファイル分割出力 [-zf](F)

  フラッシュ ROM 内蔵製品においてブート領域とそれ以外の領域を別の HEX フォーマット・ファイルに分割出力する場合に，チェックします。

  注意　フラッシュ ROM 領域セルフ書き換え機能を持たないデバイスでは，このオプションを指定しないでください。

- コマンドラインオプション

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

《出力 2》タブ

- シンボル・テーブル・ファイルの出力 [-s](S)

  シンボル・テーブル・ファイルを出力する場合に，チェックします。

  出力ファイル名：

  [参照 (B)] ボタン，または直接入力により，シンボル・テーブル・ファイルのパスとファイル名を指定します。

- エラー・リスト・ファイルの出力 [-e](E)

  エラー・リスト・ファイルを出力する場合に，チェックします。

  出力パス名：

  [参照 (R)] ボタン，または直接入力により，エラー・リスト・ファイルのパスとファイル名を指定します。

- コマンドラインオプション

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

《その他》タブ

- **コマンドファイルの作成** (U)

  コマンドファイルを作成する場合に，チェックします。

- **パラメータ・ファイル** [-f](P)

  ［参照 (B)**］ボタン，または直接入力により，ユーザ定義のパラメータ・ファイルとして入力するファイルを指定します。**

- **その他のオプション** (O)

  ダイアログで設定可能なオプション以外のオプションを指定したい場合に入力ボックスに入力します。

  注意　ヘルプ指定 (--) **オプションは，**PM plus **上では指定できません。**

- **リセット** (R)

  入力した内容をリセットします。

- **オプション情報読込** (E)

  〈オプション情報読込み〉ダイアログが開き，オプション情報ファイルを指定後，読み込みます。

- **オプション情報保存** (S)

  〈オプション情報の保存〉ダイアログが開き，オプション情報ファイルに名前をつけて保存します。

- **コマンドラインオプション**

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

第 8 章　ライブラリアン

この章では，ライブラリアンは，RA78K0 のオブジェクト・モジュール・ファイル，またはライブラリ・ファイルに対してモジュール単位で編集を行います。

さらに，リスト・ファイルを出力します。

ライブラリアン・エラーがある場合は，エラー・メッセージを出力し，エラーの原因を明示します。

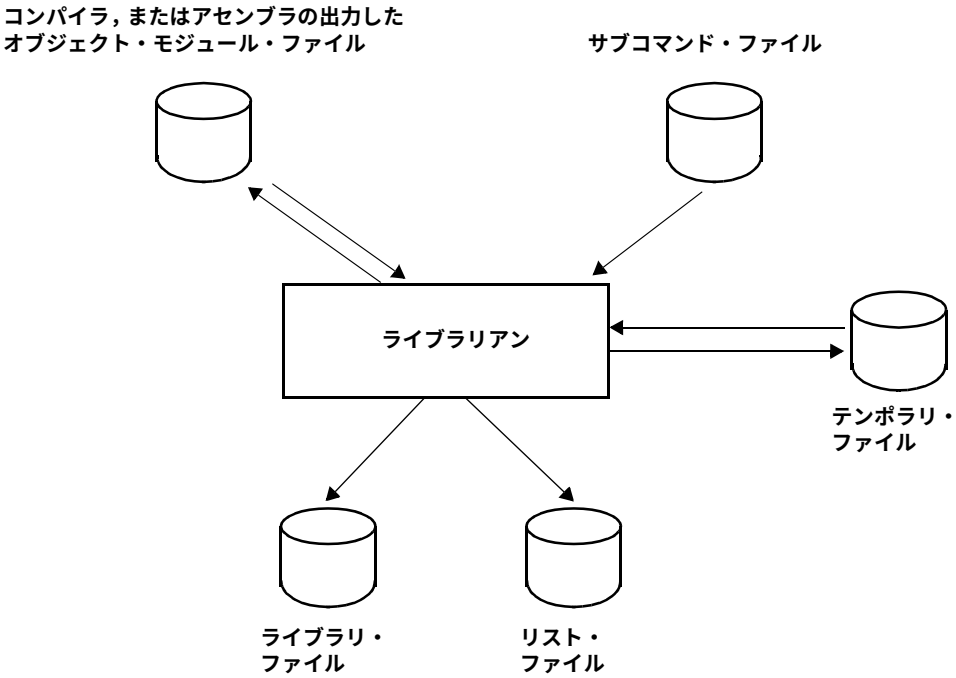
8.1　ライブラリアンの入出力ファイル

ライブラリアンの入出力ファイルを次に示します。

表 8-1 ライブラリアンの入出力ファイル

種類	ファイル名	説明	ディフォールト・ファイル・タイプ
入力ファイル	サブコマンド・ファイル	- 実行プログラムのコマンドとパラメータを内容とするファイルです。 - ユーザ作成ファイルです。	なし
出力ファイル	リスト・ファイル	- ライブラリ・ファイルの情報を出力した結果のファイルです。	.lst
入出力ファイル	オブジェクト・モジュール・ファイル	- アセンブラ，またはコンパイラの出力したオブジェクト・モジュール・ファイルです。	.rel
	ライブラリ・ファイル	- ライブラリアンが出力したライブラリ・ファイルを入力し，内容を更新します。	.lib
	テンポラリ・ファイル	- ライブラリ化のためにライブラリアンが自動生成するファイルです。ライブラリアンの実行終了時には消去されます。	Lbxxxxxx.$$y (y = 1-6)

図 8-1 ライブラリアンの入出力ファイル



8.2 ライブラリアンの機能

(1) モジュールのライブラリ化

アセンブラ，およびリンカは，1 個の出力モジュールを 1 個のファイルに作成します。

したがって，モジュールの数が多い場合，ファイルの数も増加します。このため，複数のモジュールを 1 個のファイルにまとめる機能を用意しました。これをモジュールのライブラリ化と呼び，ライブラリ化されたファイルをライブラリ・ファイルと呼びます。

ライブラリ・ファイルは，リンカに入力できます。したがって，モジュラ・プログラミングを行った場合，共通的なモジュールをライブラリ・ファイルとして作成しておけば，ファイル管理の面でも操作性の面においても効率がよくなります。

(2) ライブラリ・ファイルに対する編集

ライブラリアンには，ライブラリ・ファイルに対して次に示す編集機能があります。

(a) ライブラリ・ファイルへのモジュールの追加
(b) ライブラリ・ファイル内のモジュールの削除
(c) ライブラリ・ファイル内のモジュールの置換
(d) ライブラリ・ファイル内のモジュールの抽出

機能の詳細については，「8.5 サブコマンド」を参照してください。

(3) ライブラリ・ファイル情報の出力

ライブラリアンには，ライブラリ・ファイルの中に格納されている情報のうち，次に示すものを編集し，出力する機能があります。

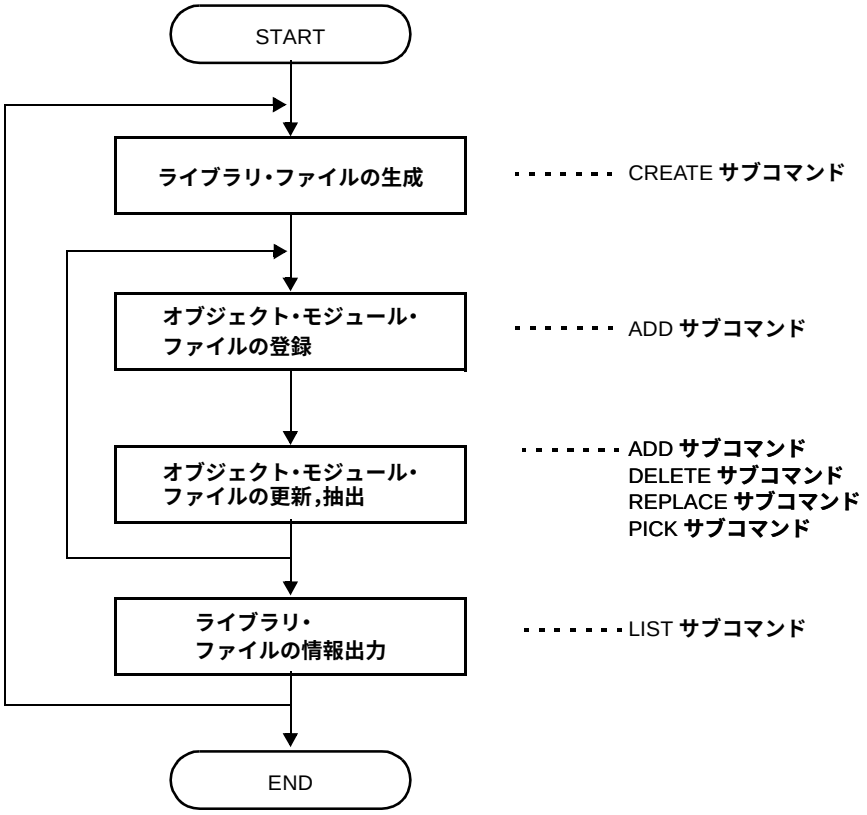
(a) モジュール名
(b) 作成プログラム
(c) 登録日付
(d) 更新日付
(e) PUBLIC シンボル情報

注意 ライブラリアンでは，(2)，(3) で説明した機能をそれぞれサブコマンドとして提供しています。ライブラリアンは，各サブコマンドを順次解説しながら処理を行います。サブコマンドの操作については，「8.5 サブコマンド」を参照してください。

(4) ライブラリ・ファイルの作成手順

一般的なライブラリ・ファイルの作成手順を次に示します。

図 8-2 ライブラリ・ファイルの作成手順



8.3 ライブラリアンの起動方法

8.3.1 ライブラリアンの起動

ライブラリアンの起動には，次の 2 つの方法があります。

(1) コマンド行での起動

```
X>[ パス名 ] lb78k0 [ △オプション ] …
↑    ↑          ↑              ↑
(a)  (b)        (c)            (d)
```

(a) カレント・ドライブ名

(b) カレント・ディレクトリ名

(c) ライブラリアンのコマンド・ファイル名

(d) ライブラリアンに対して動作の詳細を指示します。
空白を含むパスを設定する場合には，ダブルクオーテーション（” “）で囲んでください。

例 C>lb78k0 -ll20 -lw80

注意 複数のライブラリアン・オプションを指定する場合は，それぞれのライブラリアン・オプション間を空白で区切ります。
ライブラリアン・オプションの詳細については，「8.4 ライブラリアン・オプション」を参照してください。

ライブラリアンが起動すると，次の起動メッセージが表示されます。

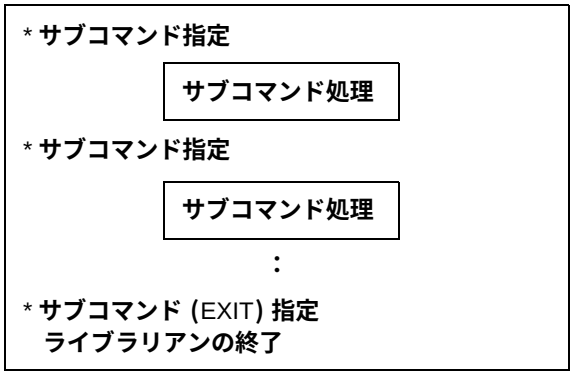
```
78K/0 Series Librarian Vx.xx [ xx xxx xx ]
   Copyright ( C ) NEC Electronics Corporation xxxx , xxxx
*
```

“ * ”に続いて，ライブラリアンのサブコマンドを指定します。

```
*create k0.lib
*add k0.lib k0main.rel k0sub.rel
*exit
```

サブコマンドの入力を終了すると，各サブコマンドの処理を開始します。1 つのサブコマンドの処理が完了すると再び“ * ”が出力され，次のサブコマンドの入力待ち状態となります。終了サブコマンド（EXIT）が

入力されるまで，この動作は繰り返されます。



1 行で指定可能な文字数は，128 文字です。

1 行に必要なオペランド情報をすべて入力できない場合は，“ & ” を用いて継続行の指定ができます。なお，継続できる行数は 15 行です。

(2) サブコマンド・ファイルによる起動

サブコマンド・ファイルとはライブラリアンへのコマンドを格納しているファイルです。

ライブラリアンの起動時にサブコマンド・ファイルを指定しない場合，“ * ” のあとに複数のサブコマンドを入力しなければなりません。しかし，サブコマンド・ファイルを作成することにより，これら複数のサブコマンドの処理が一度にできます。

また，ライブラリ化のたびに同じサブコマンドを繰り返し指定することがあります。このような場合にサブコマンド・ファイルを使用してライブラリ化を行います。

サブコマンド・ファイルを使用する場合には，ファイル名の前に “ < ” を記述します。

サブコマンド・ファイルによる起動方法を次に示します。

```
X>lb78k0 △  <  サブコマンド・ファイル名  [ △オプション ] …
            ↑            ↑
           (i)          (ii)
```

(i) サブコマンド・ファイルを指定する場合は，必ず付加してください。

(ii) サブコマンドが格納されているファイル

(a) サブコマンド・ファイルは，エディタなどで作成してください。

(b) サブコマンド・ファイル内での記述規則を次に示します。

```
サブコマンド名  オペランド情報
サブコマンド名  オペランド情報
  ：
EXIT
```

(c) 1 つのサブコマンドが複数行にわたる場合，各行の行末に行の継続を示す “ & ” を記述します。

(d) “ ; ”（セミコロン）から行末までは，ライブラリアンのコマンドとしては解釈されず，コメントとしてみなされます。

(e) サブコマンド・ファイルの最後のサブコマンドが EXIT サブコマンドでない場合には，自動的に EXIT サブコマンドがあったものと解釈されます。

(f) ライブラリアンは，サブコマンドをサブコマンド・ファイルから読み込んで処理を行います。
サブコマンド・ファイル内のすべてのサブコマンドの処理を終了するとライブラリアンは終了します。

例　サブコマンド・ファイル（k0.slb）をエディタなどで作成します。

＜ k0.slb の内容＞

```
; library creation command
create k0.lib
add k0.lib k0main.rel &
k0sub.rel
;
exit
```

サブコマンド・ファイル（k0.slb）を使用してライブラリアンを起動します。

```
C>lb78k0 <k0.slb
```

8.3.2 実行開始メッセージ，終了メッセージ

(1) 実行開始メッセージ

ライブラリアンが起動すると，次の実行開始メッセージが表示されます。

```
78K/0 Series Librarian Vx.xx [ xx xxx xx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx
*
```

(2) 実行終了メッセージ

ライブラリアンは，実行終了メッセージを出力しません。各処理を終了したあとにユーザが EXIT コマンドを入力することにより，ライブラリアンは制御を OS に戻します。

```
*create k0.lib
*add k0.lib k0main.rel k0sub.rel
*exit
```

ライブラリアンの処理継続が不可能な致命的エラーが検出された場合ライブラリアンはメッセージを表示して，制御を OS に戻します。

例　存在しないライブラリアン・オプションを指定した場合

```
C>lb78k0 -a

78K/0 Series Librarian Vx.xx [ xx xxx xx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx
RA78K0 error F5018 : Option is not recognized ' -z '
Usage : LB78K0 [ options ]
```

この例では，存在しないライブラリアン・オプションを指定したためにエラーとなり，ライブラリアンの実行が中止されました。

ライブラリアンがエラー・メッセージを出力してライブラリ化を中止した場合には，そのエラー・メッセージの原因を「第 12 章 エラー・メッセージ」で調べて対処してください。

8.4 ライブラリアン・オプション

8.4.1 ライブラリアン・オプションの種類

ライブラリアン・オプションは，リスト・ファイルの形式やテンポラリ・ファイルの作成パスなどの指定を行います。ライブラリアン・オプションは，4 種類のオプションに分類できます。

表 8-2 ライブラリアン・オプション

分類	オプション	説明
リスト・ファイル形式指定	-LW	リスト・ファイルの 1 行に印字する文字数を変更します。
	-LL	リスト・ファイルの 1 頁に印字する行数を変更します。
	-LF	リスト・ファイルの最後に改頁コードを付加します。
	-NLF	
テンポラリ・ファイル作成パス指定	-T	テンポラリ・ファイルを指定したパスに作成します。
デバイス・ファイル・サーチ・パス指定	-Y	デバイス・ファイルを指定されたパスから読み込みます。
ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

8.4.2 ライブラリアン・オプションの説明

次ページ以降に，各ライブラリアン・オプションの詳細について説明します。

(1) リスト・ファイル形式指定

リスト・ファイル形式指定 （-LW , -LL , -LF/-NLF）

(a) -LW

【記述形式】

-LW [**文字数**]

- **省略時解釈**

  -LW132（**ディスプレイ出力の場合は** 80 **文字とします**）

【機能】

- -LW **オプションは，リスト・ファイルの** 1 **行の文字数を指示します。**

【用途】

- **リスト・ファイルの** 1 **行の文字数を変更したいときに，**-LW **オプションで指定します。**

【説明】

- -LW **オプションで指定できる文字数の範囲（ディスプレイ出力の場合は，**80 **文字まで）は次のとおりです。**

  72 ≦ 1 **行に印字する文字数** ≦ 260

  範囲外の数値や数値以外を指定した場合には，アボート・エラーとなります。
- **文字数を省略した場合は，**132 **を指定したものとみなします。ただし，リスト・ファイルの出力先がディスプレイの場合は，**80 **となります。**
- **指定する文字数には，ターミネータ（**CR , LF**）は含みません。**
- LIST **サブコマンドが指定されない場合，**-LW **オプションは無視されます。**
- -LW **オプションを複数指定した場合は，あとで指定した方が有効となります。**

【使用例】

- **リスト・ファイルの** 1 **行の文字数を** 80 **文字に指定します。**

```
C>lb78k0 -lw80
```

(b) -LL

【記述形式】

-LL [**行数**]

- **省略時解釈**

  -LL66（**ディスプレイ出力の場合は改頁しません**）

【機能】

- -LL **オプションは，リスト・ファイルの** 1 **頁の行数を指定します。**

【用途】

リスト・ファイルの 1 **頁の行数を変更したいときに，**-LL **オプションを指定します。**

【説明】

- -LL **オプションで指定できる行数の範囲は次のとおりです。**

  20 ≦ 1 **頁に印字する行数** ≦ 32767

  範囲外の数値や数値以外を指定した場合は，アボート・エラーとなります。
- **行数が省略された場合，**66 **が指定されたものとみなします。**
- **行数の** 0 **を指定した場合は改頁しません。**
- LIST **サブコマンドが指定されない場合，**-LL **オプションは無視されます。**
- -LL **オプションを複数指定した場合は，あとで指定した方が有効となります。**

【使用例】

- **リスト・ファイルの** 1 **頁の行数を** 20 **行に指定します。**

```
C>lb78k0 -ll20
```

(c) -LF/-NLF

【記述形式】

```
-LF
-NLF
```

- **省略時解釈**

  -NLF

【機能】

- -LF **オプションは，リスト・ファイルの最後に改頁コード**（FF）**を付加する指定をします。**
- -NLF **オプションは，**-LF **オプションを無効にします。**

【用途】

- **リスト・ファイルの内容を印字したあとで改頁しておきたい場合に，**-LF **オプションを指定して改頁コードを付加します。**

【説明】

- LIST **サブコマンドが指定されない場合，**-LF **オプションは無視されます。**
- -LF **と** -NLF **の両オプションを同時に指定した場合は，あとで指定した方が有効となります。**

【使用例】

- **リスト・ファイルに改頁コードを付加します。**

```
C>lb78k0 -lf
```

(2) テンポラリ・ファイル作成パス指定

テンポラリ・ファイル作成パス指定 （-T）

【記述形式】

-T パス名

- 省略時解釈

  環境変数 TMP により指定されたパスに作成します。

  指定されていない場合は，カレント・パスに作成します。

【機能】

- -T オプションは，テンポラリ・ファイルを作成するパスを指定します。

【用途】

- テンポラリ・ファイルの作成場所を指定できます。

【説明】

- パス名としてパス以外のものは指定できません。
- パス名は省略できません。
- 以前に作成されたテンポラリ・ファイルが存在している場合でも，ファイル保護をしていなければ，上書きします。
- 必要とするメモリ・サイズが残っている間は，テンポラリ・ファイルをメモリに展開します。
  なお，メモリが足りなくなった時点で，メモリに展開していたテンポラリ・ファイルの内容をディスクに書き出します。以降のテンポラリ・ファイルへのアクセスは，セーブしたディスク・ファイルに対して行います。
- テンポラリ・ファイルは，ライブラリ化終了時に削除されます。また，キー入力（CTRL-C）によってライブラリ化が中止されたときも，削除されます。
- テンポラリ・ファイルの作成パスは，次の順序で決定されます。

  (i) -T オプションで指定されたパス

  (ii) 環境変数 TMP に設定されているパス（-T オプション省略の場合）

  (iii) カレント・パス（TMP が設定されていない場合）

  なお，(i) または (ii) を指定した場合，指定されたパスにテンポラリ・ファイルが作成できなければアボート・エラーとなります。

【使用例】

- テンポラリ・ファイルをディレクトリ c:\TMP に出力するよう指定します。

```
C>lb78k0 -tc:\tmp
```

(3) デバイス・ファイル・サーチ・パス指定

デバイス・ファイル・サーチ・パス指定 （-Y）

【記述形式】

-Y パス名

- 省略時解釈

  デバイス・ファイルを読み込むパスは，次の順序で調べ決定します。

  (i) < ..\dev >（lb78k0.exe の起動されたパスに対して）

  (ii) LB78K0 の起動されたパス

  (iii) カレント・ディレクトリ

  (iv) 環境変数 PATH

【機能】

- デバイス・ファイルを指定されたパスから読み込みます。

【用途】

- デバイス・ファイルが存在するパスを指定します。

【説明】

- -Y オプションに続けてパス名以外が指定された場合，アボート・エラーとなります。
- -Y オプションに続けて指定するパス名が省略された場合，アボート・エラーとなります。
- デバイス・ファイルを読み込むパスは，次の順序で調べ決定します。

  (i) -Y オプションで指定されたパス

  (ii) < ..\dev >（lb78k0.exe の起動されたパスに対して）

  (iii) LB78K0 の起動されたパス

  (iv) カレント・ディレクトリ

  (v) 環境変数 PATH

【使用例】

- デバイス・ファイルのパスを c:\78k0\dev ディレクトリに指定します。

```
C>lb78k0 -yc:\78k0\dev
```

(4) ヘルプ指定

ヘルプ指定 (--)

【記述形式】

```
--
```

- 省略時解釈

  表示しない

【機能】

- -- オプションは，ヘルプ・メッセージをディスプレイに出力します。

【用途】

- ヘルプ・メッセージは，サブコマンドとその説明の一覧です。ライブラリアンを実行するときに参照してください。

【説明】

- -- オプションを指定すると他のライブラリアン・オプションは，すべて無効となります。

  注意 このオプションは，PM plus 上では指定できません。

  PM plus 上でヘルプを参照する場合は，〈ライブラリ・ファイルの指定〉ダイアログで［ヘルプ］ボタンをクリックしてください。

【使用例】

- -- **オプションを指定するとヘルプ・メッセージがディスプレイに出力されます。**

```
C>lb78k0 --

78K/0 Series Librarian Vx.xx [ xx xxx xx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx
+-------------------------------------------------------------------------------------------------------+
|  Subcommands : create , add , delete , replace , pick , list , help , exit                            |
|                                                                                                       |
|    Usage : subcommand [ option ] masterLBF [ option ] transaction [ option ]                          |
|                                                                                                       |
|                    transaction : ==                        OMFname                                    |
|                                                  LBFname [ ( modulename [ , ... ] ) ]                 |
|                                                                                                       |
|    <create      > :      create masterLBF [ transaction ]                                             |
|    <add         > :      add masterLBF transaction                                                    |
|    <delete      > :      delete masterLBF ( modulename [ , ... ] )                                    |
|    <replace     > :      replace masterLBF transaction                                                |
|    <pick        > :      pick masterLBF ( modulename [ , ... ] )                                      |
|    <list        > :      list [ option ] masterLBF [ ( modulename [ , ... ] )                         |
|                          option :  -p                = output public symbol                           |
|                                    -np               = no output public symbol                        |
|                                    -o filename = specify output file name                             |
|    <help        > : help                                                                              |
|    <exit        > : exit                                                                              |
+-------------------------------------------------------------------------------------------------------+
```

8.5 サブコマンド

8.5.1 サブコマンドの種類

サブコマンドは，ライブラリアンの動作に細かい指示を与えます。サブコマンドは，8 種類のオプションに分類できます。

表 8-3 サブコマンド

サブコマンド名	短縮名	説明
CREATE	C	ライブラリ・ファイルを新規に作成します。
ADD	A	ライブラリ・ファイルにモジュールを追加します。
DELETE	D	ライブラリ・ファイル内のモジュールを削除します。
REPLACE	R	ライブラリ・ファイル内のモジュールを他のモジュールと置き換えます。
PICK	P	ライブラリ・ファイル内のモジュールを取り出します。
LIST	L	ライブラリ・ファイル内のモジュール情報を出力します。
HELP	H	コンソールにヘルプ・メッセージを出力します。
EXIT	E	ライブラリアンを終了します。

8.5.2 サブコマンドの説明

各サブコマンドの詳細について説明します。

【コマンド・ファイルの一般形式】

```
*サブコマンド [△オプション]△ライブラリ･ファイル名[△オプション]トランザクション[△オプション]
                                  ↑                    ↑
                                 (a)                  (b)
```

(a) 直前に指定されたライブラリ・ファイル名は“ . ”で置き換えできます。

(b) トランザクション = △オブジェクト・モジュール・ファイル名

△ライブラリ・ファイル名 [△ (△モジュール名 [△ , …])]

(1) CREATE

CREATE

【記述形式】

CREATE △**ライブラリ・ファイル名** [△**トランザクション**]

- **省略時解釈**

  C

【機能】

- CREATE **サブコマンドは，ライブラリ・ファイルを新規に作成します。**

【説明】

- **作成されたライブラリ・ファイルのサイズは** 0 **となります。**
- **トランザクションを指定した場合は，ライブラリ・ファイルの作成と同時にモジュールを登録します。**
- **ライブラリ・ファイル名：**

  指定したファイルがすでに存在している場合には，上書きします。
- **トランザクション：**

  ライブラリ・ファイル中にパブリック・シンボルと同一のパブリック・シンボルを持つオブジェクト・モジュール・ファイルは，登録できません。

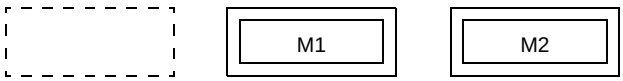
  またライブラリ・ファイル中にあるモジュールと同一の名前のモジュールは登録できません。
- **エラーが発生した場合は処理を中断し，ライブラリ・ファイルは作成されません。**

【使用例】

- **ライブラリ・ファイルを作成し同時にモジュール** M1 **と** M2 **を登録します。**

```
*create k0.lib m1.rel m2.rel
```

〈作成前〉



〈作成後〉

k0.lib

(2) ADD

ADD

【記述形式】

ADD △ライブラリ・ファイル名△トランザクション

- **省略時解釈**

  A

【機能】

- **既存のライブラリ・ファイルに対してモジュールを追加します。**

【説明】

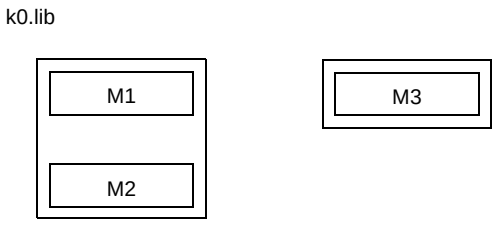
- **追加するライブラリ・ファイル中には，モジュールが存在していなくてもかまいません。**
- **追加するモジュールと同名のモジュールがライブラリ・ファイル内に存在する場合，エラーとなります。**
- **追加するモジュール中のパブリック・シンボルがライブラリ・ファイル内に存在する場合，エラーとなります。**

【使用例】

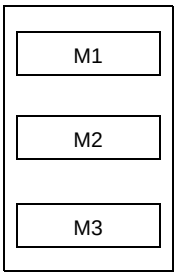
- **ライブラリ・ファイル（**k0.lib**）にモジュール（**M3**）を追加します。**

```
*add k0.lib m3.rel
```

〈追加前〉



〈追加後〉



(3) DELETE

DELETE

【記述形式】

DELETE △**ライブラリ・ファイル名**△ (△**モジュール名** [△ , …] △)

- **省略時解釈**

  D

【機能】

- **既存のライブラリ・ファイルからモジュールを削除します。**

【説明】

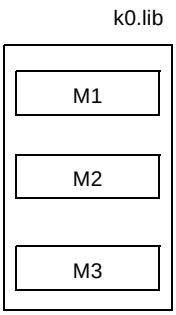
- **指定したモジュールがライブラリ・ファイルに存在しない場合，エラーとなります。**
- **エラーが発生した場合は処理を中断し，ライブラリ・ファイルの状態は変化しません。**

【使用例】

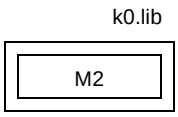
- **ライブラリ・ファイル（**k0.lib**）からモジュール（**M1 , M3**）を削除します。**

```
*delete k0.lib ( m1.rel , m3.rel )
```

〈削除前〉



〈削除後〉



(4) REPLACE

REPLACE

【記述形式】

REPLACE △ライブラリ・ファイル名△トランザクション

- **省略時解釈**
  R

【機能】

- **既存のライブラリ・ファイルのモジュールを，他のオブジェクト・モジュール・ファイルのモジュールと置き換えます。**

【説明】

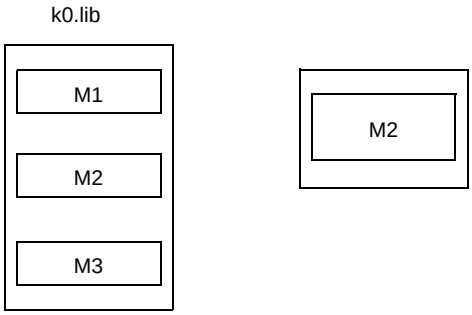
- **置換するモジュール名と同名のモジュールが,更新するライブラリ・ファイル内に存在しない場合はエラーとなります。**
- **置換するモジュール中のパブリック・シンボルが，ライブラリ・ファイル内に存在する場合はエラーとなります。**
- **置換するオブジェクト・モジュール・ファイル名は，登録時と同じファイル名でなければなりません。**
- **エラーが発生した場合は処理を中断し，ライブラリ・ファイルの状態は変化しません。**

【使用例】

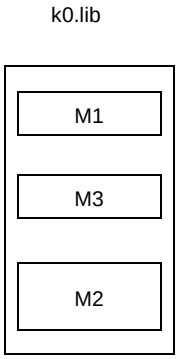
- **ライブラリ・ファイル**（k0.lib）**中のモジュール**（M2）**を置換します。**

*replace k0.lib m2.rel

〈置換前〉



〈置換後〉



ライブラリ・ファイル中のモジュール（M2）を削除したあと，新たにモジュール（M2）を登録するため，ライブラリ・ファイル中のモジュール（M2）の順序は最後となります。

(5) PICK

PICK

【記述形式】

PICK △**ライブラリ・ファイル名**△ (△**モジュール名** [△ , …] △)

- **省略時解釈**

  P

【機能】

- **既存のライブラリ・ファイルのモジュールから指定したモジュールを取り出します。**

【説明】

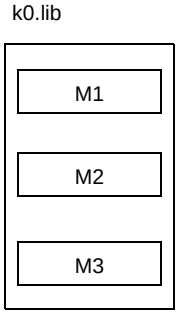
- **取り出したモジュールは，登録時のファイル名を持つオブジェクト・モジュール・ファイルとなります。**
- **指定したモジュール名が，ライブラリ・ファイル内に存在しない場合はエラーとなります。**
- **エラーの場合は処理を中断します。ただし，複数のモジュールが指定されているときにエラーが発生した場合には，エラーとなったモジュールの直前までに取り出されたモジュールは有効となりディスク上に保存されます。**

【使用例】

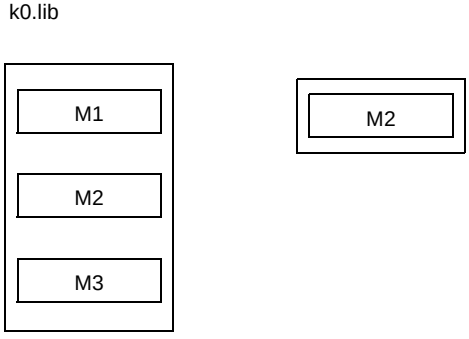
- **ライブラリ・ファイル（**k0.lib**）中のモジュール（**M2**）を取り出します。**

```
*pick k0.lib ( m2.rel )
```

〈抽出前〉



〈抽出後〉



(6) LIST

LIST

【記述形式】

LIST [△**オプション**] △**ライブラリ・ファイル名** [△ (△**モジュール名** [△ , …] △)]
オプション：-PUBLIC/-NOPUBLIC
：-O △**ファイル名**

- **省略時解釈**
  L

【機能】

- **ライブラリ・ファイル内のモジュール情報を出力します。**

【説明】

- **オプションは，複数指定できます。**
- -O**：**
  出力ファイル名には，デバイス型ファイル名を指定できます。
  出力ファイル名を省略した場合は，エラーとなります。
  ファイル・タイプを省略した場合は，“ 入力ファイル名 .lst **” が入力されたものとします。**
- -PUBLIC/-NOPUBLIC**：**
  下線部のみの指定も可能です。
  -PUBLIC **は，パブリック・シンボル情報の出力を指示します。**
  -NOPUBLIC **は，**-PUBLIC **を無効にします。**
  -PUBLIC **と** -NOPUBLIC **の両方を指定した場合は，あとで指定した方を優先します。**

【使用例】

- **ライブラリ・ファイル（**k0.lib**）のモジュール情報をリスト・ファイル（**k0.list**）に出力します。この際，パブリック・シンボル情報が出力されるように，**-P **オプションを指定します。**

```
*list -p –ok0.lst k0.lib
```

リスト・ファイル（k0.lst**）を参照します。**

```
78K/0 Series librarian Vx.xx    DATE : xx xxx xx PAGE   1

LIB-FILE NAME : K0.LIB    ( xx xxx xx )

0001  M1.REL                ( xx xxx xx )

   sym1       sym2       sym3

   NUMBER OF PUBLIC SYMBOLS :   3

0002  M3.REL                ( xx xxx xx )

   NUMBER OF PUBLIC SYMBOLS :   0

0003  M2.REL                ( xx xxx xx )

   bit1       bit2

   NUMBER OF PUBLIC SYMBOLS :   2
```

(7) HELP

HELP

【記述形式】

```
HELP
```

- **省略時解釈**

  H

【機能】

- **ヘルプ・メッセージをディスプレイに出力します。**

【説明】

- **ヘルプ・メッセージは，サブコマンドとその説明の一覧です。**HELP **コマンド，または** -- **オプションを指定してライブラリアンを実行するときに参照してください。**

【使用例】

- HELP **コマンドを指定するとヘルプ・メッセージが出力されます。**

  *help

```
+-------------------------------------------------------------------------------------------------------------------------+
|  Subcommands : create , add , delete , replace , pick , list , help , exit                                              |
|                                                                                                                         |
|    Usage : subcommand [ option ] masterLBF [ option ] transaction [ option ]                                            |
|                                                                                                                         |
|                  transaction : ==          OMFname                                                                      |
|                                            LBFname [ ( modulename [ , ... ] ) ]                                         |
|                                                                                                                         |
|    <create    > :    create masterLBF [ transaction ]                                                                   |
|    <add       > :    add masterLBF transaction                                                                          |
|    <delete    > :    delete masterLBF ( modulename [ , ... ] )                                                          |
|    <replace   > :    replace masterLBF transaction                                                                      |
|    <pick      > :    pick masterLBF ( modulename [ , ... ] )                                                            |
|    <list      > :    list [ option ] masterLBF [ ( modulename [ , ... ] )                                               |
|                              option :          -p        = output public symbol                                         |
|                                                -np       = no output public symbol                                      |
|                                                -o filename = specify output file name                                   |
|    <help      > :    help                                                                                               |
|    <exit      > :    exit                                                                                               |
+-------------------------------------------------------------------------------------------------------------------------+
```

(8) EXIT

EXIT

【記述形式】

```
EXIT
```

- **省略時解釈**

  E

【機能】

- **ライブラリアンを終了します。**

【説明】

- **ライブラリアンを終了するときに使用します。**

【使用例】

- **ライブラリアンを終了します。**

```
*exit
```

8.6 PM plus でのオプション設定

PM plus からライブラリアン・オプションの指定をする方法について説明します。

8.6.1 オプションの設定方法

PM plus の [ツール (T)] メニューの [ライブラリアンオプションの設定 (B)] を選択するか，ツール・バーの [LB] ボタンを押下すると，〈ライブラリアンオプションの指定〉ダイアログが現れます。

ダイアログ内で必要なオプションを入力することにより，各ライブラリアンのオプションを設定できます。

図 8-3 〈ライブラリアンオプションの設定〉ダイアログ（《出力》タブ選択時）

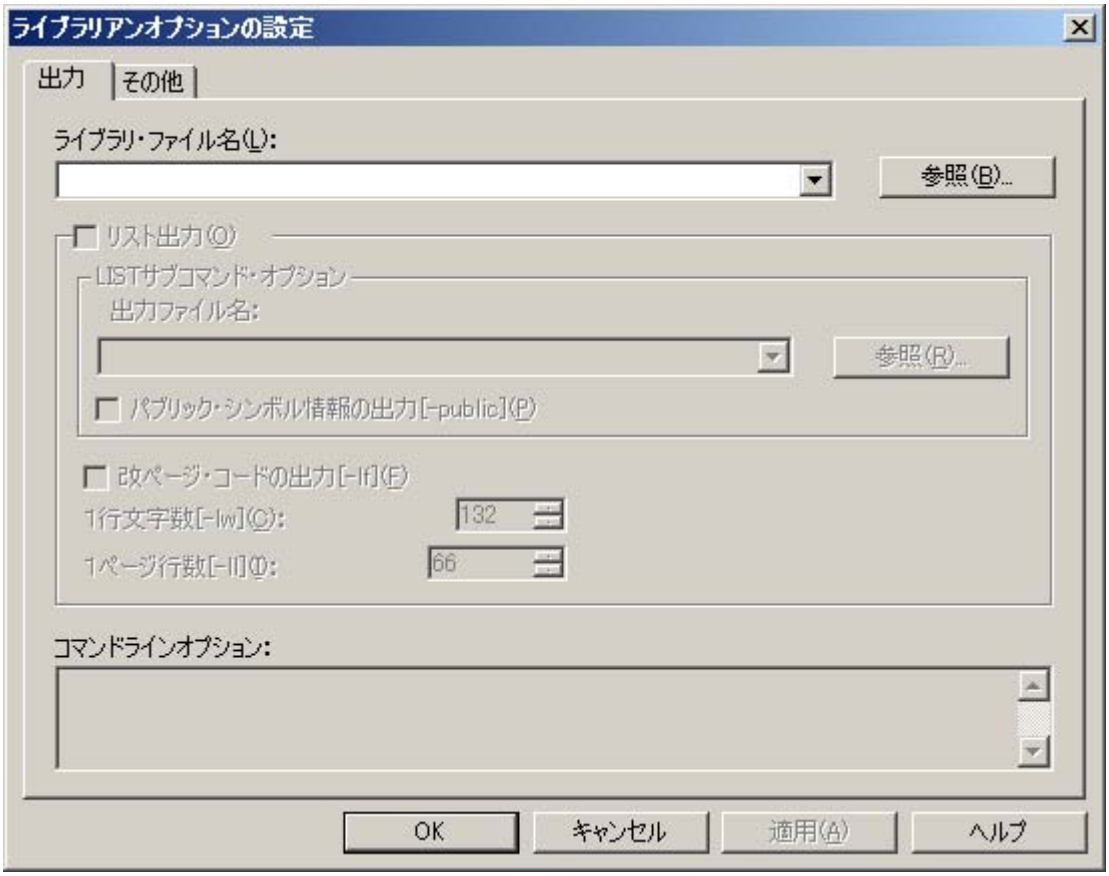
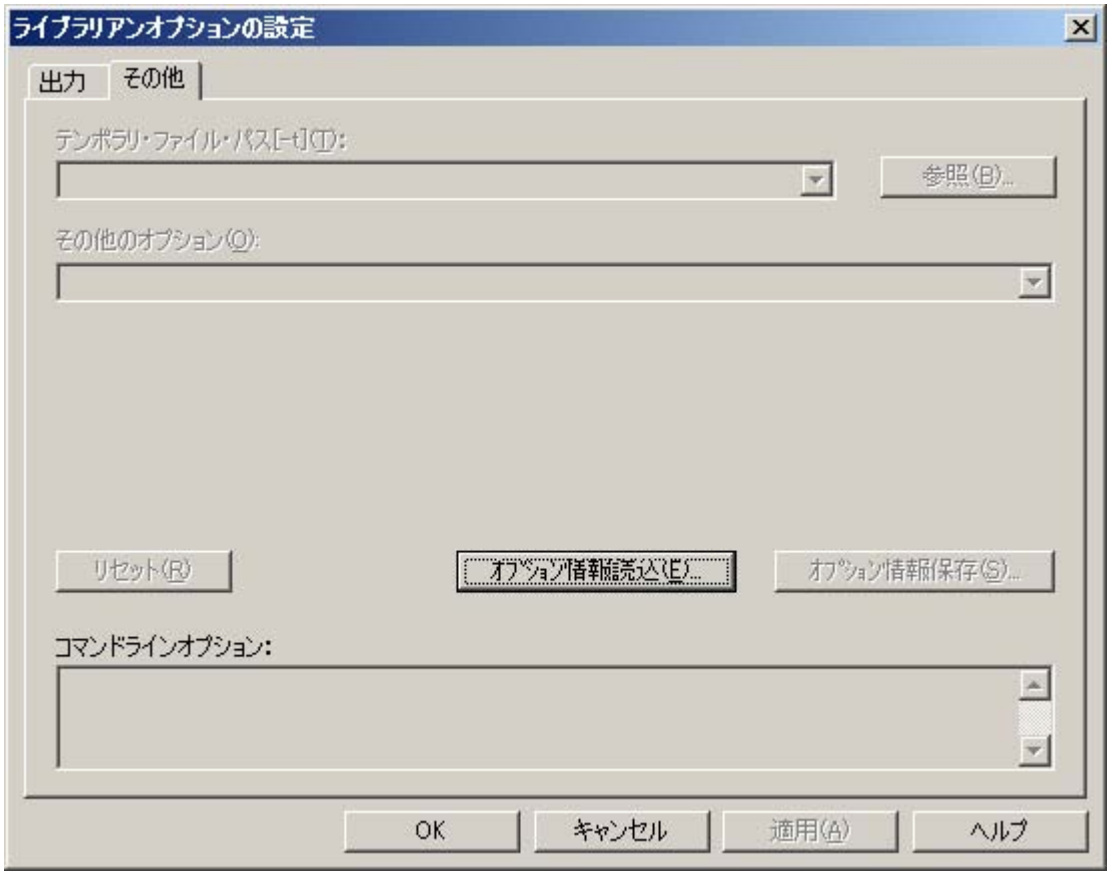


図 8-4 〈ライブラリアンオプションの設定〉ダイアログ（《その他》タブ選択時）



8.6.2 各オプションの設定

〈ライブラリアンオプションの設定〉ダイアログの各オプションについて，次に説明します。

《出力》タブ

- ライブラリ・ファイル名 (L)

  ［参照 (B)］ボタン，または直接入力により，ライブラリ・ファイル名を指定します。

- リスト出力 (O)

  リスト・ファイルを出力する場合に，チェックします。

- 出力ファイル名：

  ［参照 (R)］ボタン，または直接入力により，リスト・ファイルのパスとファイル名を指定します。

- パブリック・シンボル情報の出力 [-public](P)

  リスト・ファイル中にパブリック・シンボル情報を付加する場合に，チェックします。

- 改ページ・コードの出力 [-lf](F)

  ライブラリ・ファイルの最後に改頁コード (FF) を付加する場合に，チェックします。

- 1 行文字数 [-lw](C)

  ライブラリ・ファイルの１行の文字数を指定します。指定可能な文字数は，72 ～ 260 の範囲です。

- 1 ページ行数 [-ll](l)

  ライブラリ・ファイルの１ページの行数を指定します。指定可能な行数は，20 ～ 32767 の範囲です。

- コマンドラインオプション

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

《その他》タブ

- テンポラリ・ファイル・パス [-t](T)

  ［参照 (B)］ボタン，または直接入力によりテンポラリ・ファイルの作成するパスを指定します。

- その他のオプション (O)

  ダイアログで設定可能なオプション以外のオプションを指定したい場合に入力ボックスに入力します。

  注意　ヘルプ指定 (--) のオプションは，PM plus 上では指定できません。

- リセット (R)

  入力した内容をリセットします。

- オプション情報読込 (E)

  〈オプション情報読込み〉ダイアログが開き，オプション情報ファイルを指定後，読み込みます。

- オプション情報保存 (S)

  〈オプション情報の保存〉ダイアログが開き，オプション情報ファイルに名前をつけて保存します。

- コマンドラインオプション

  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

8.7 PM plus でのライブラリ・ファイルの操作

PM plus からライブラリ・ファイルを操作する方法について，説明します。

8.7.1 操作方法

PM plus の [ツール(T)] メニューの [外部ツールの登録(X)] メニューから，LB 単体起動用実行形式（lb78k0p.exe）を登録します。

登録したアイコンを選択すると，〈ライブラリ・ファイルの指定〉ダイアログが現れます。

パスとファイル名を指定後，［次へ］ボタンを押下すると〈サブコマンド実行〉ダイアログが現れます。

図 8-5 〈ライブラリ・ファイルの指定〉ダイアログ

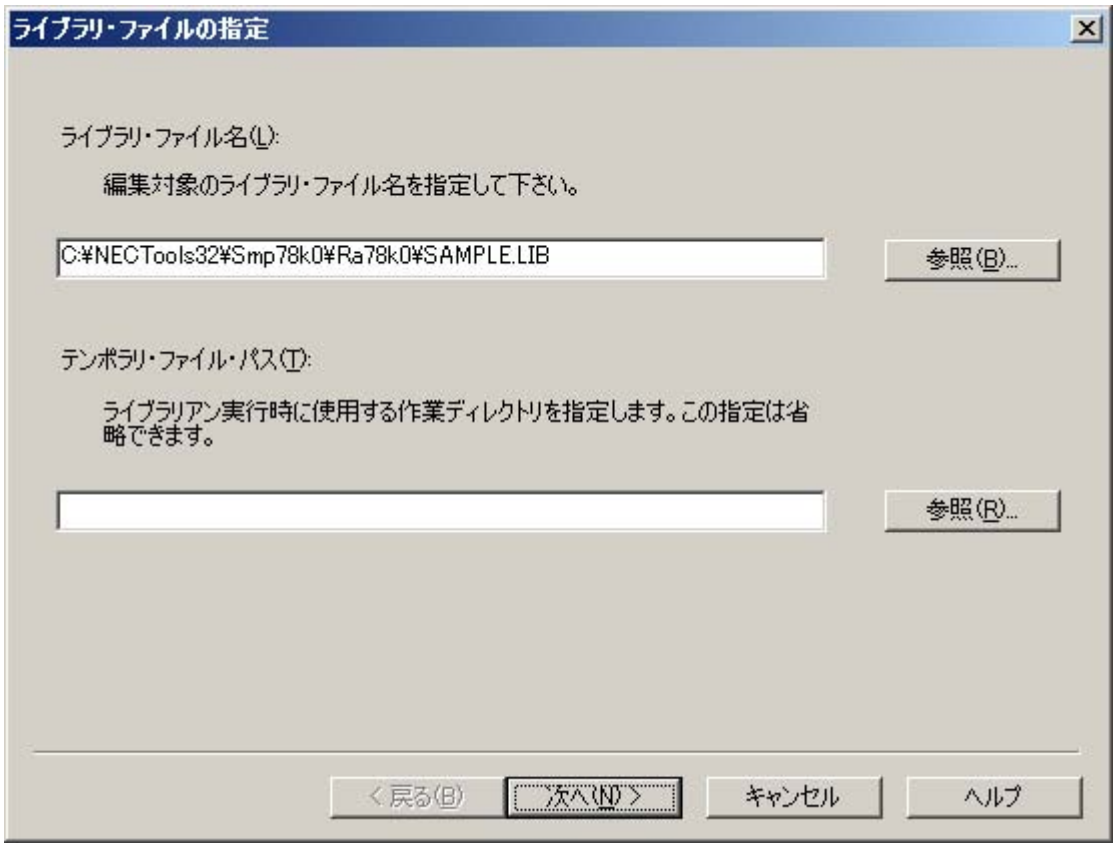
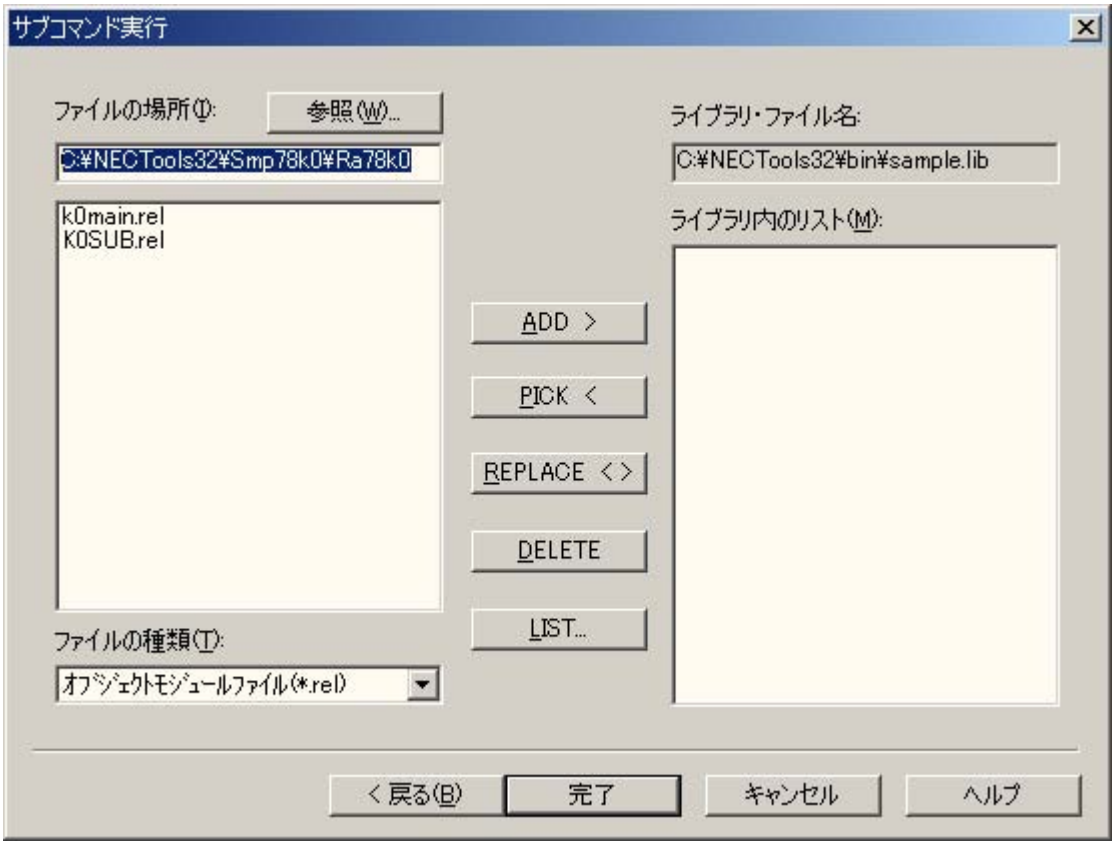


図 8-6 〈サブコマンド実行〉ダイアログ



8.7.2 各項目の設定

〈ライブラリ・ファイルの指定 〉ダイアログ，および〈サブコマンド実行 〉ダイアログの各項目について，次に説明します。

〈ライブラリ・ファイルの指定 〉ダイアログ

- **ライブラリ・ファイル名** (L)

  [参照 (B)] ボタン，または直接入力により，編集対象のライブラリ・ファイルのパスとファイル名を指定します。

- **テンポラリ・ファイル・パス** (T)

  [参照 (R)] ボタン，または直接入力により，テンポラリ・ファイルを作成するパスを指定します。

〈サブコマンド実行 〉ダイアログ

- **ファイルの場所** (I)

  [参照 (W)] ボタン，または直接入力により，ライブラリ化するオブジェクト・モジュール・ファイルのあるパスを指定します。

  パスを指定するとファイルの一覧が表示されます。

- **ファイルの種類** (T)

  ファイルの一覧に表示するファイルの種類を指定します。

- **ライブラリ・ファイル名**
  このエディット・ボックスは読み取り専用です。現在指定されているライブラリのファイル名が表示されます。
- **ライブラリ内のリスト** (M)
  指定されているライブラリ内のオブジェクト・モジュール・ファイルの一覧が表示されます。
- ADD
  既存のライブラリ・ファイルに対してモジュールを追加します。
- PICK
  既存のライブラリ・ファイルのモジュールから指定したモジュールを取り出します。
- REPLACE
  既存のライブラリ･ファイルのモジュールを他のオブジェクト･モジュール･ファイルのモジュールと置き換えます。
- DELETE
  既存のライブラリ・ファイルからモジュールを削除します。
- LIST
  ライブラリ・ファイル内のモジュール情報を出力します。

第 9 章　リスト・コンバータ

この章では，リスト・コンバータは，アセンブラが出力するアセンブル・リスト・ファイル，オブジェクト・モジュール・ファイルと，リンカが出力するロード・モジュール・ファイルを入力します。

そして，入力ファイル中のリロケータブルなアドレスやシンボルに実際のアドレスを埋め込んで，アブソリュート・アセンブル・リスト・ファイルとして出力します。こうすることによって，リンク・マップを参照しながらアセンブル・リストを見るというわずらわしさが軽減されます。

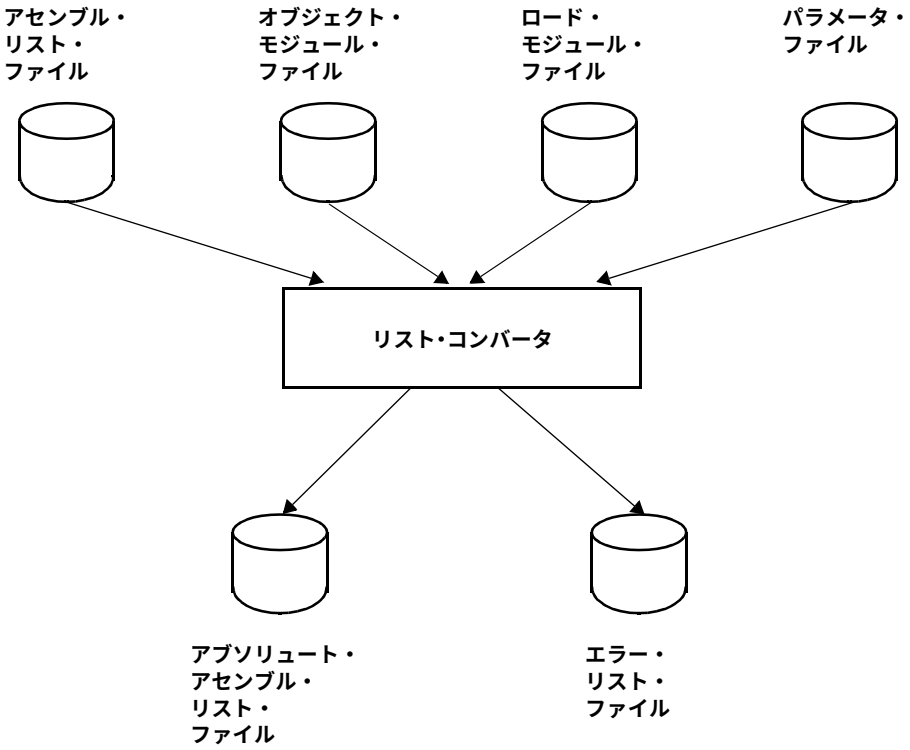
9.1　リスト・コンバータの入出力ファイル

リスト・コンバータの入出力ファイルを次に示します。

表 9-1 リスト・コンバータの入出力ファイル

種類	ファイル名	説明	ディフォールト・ファイル・タイプ
入力ファイル	オブジェクト・モジュール・ファイル	- 機械語情報と機械語の配置アドレスに関する再配置情報，およびシンボル情報を含んだバイナリ・ファイルです。	.rel
	アセンブル・リスト・ファイル	- アセンブル・リスト，クロスレファレンス・リストなどのアセンブル情報を持つファイルです。	.prn
	ロード・モジュール・ファイル	- リンク結果のオブジェクト・コードのバイナリ・イメージ・ファイルです。	.lmf
	パラメータ・ファイル	- 実行プログラムのパラメータを内容とするファイルです。 - ユーザ作成ファイルです。	.plv
出力ファイル	アブソリュート・アセンブル・リスト・ファイル	- 入力ファイル中のリロケータブルなアドレスやシンボルに実際のアドレスを埋め込んだリスト・ファイルです。	.p
	エラー・リスト・ファイル	- リスト・コンバート時のエラー情報を持つファイルです。	.elv

図 9-1 リスト・コンバータの入出力ファイル



9.2 リスト・コンバータの機能

リロケータブル・アセンブラをアブソリュート・アセンブラと比較した場合の長所，短所を次に示します。

[長所]

(1) 複数の人数で開発可能。
(2) モジュール分割により開発，保守が容易。
(3) ライブラリ管理が容易。
(4) 大規模プログラムの開発に適している。

[短所]

(1) アセンブル・リストのアドレスが実際の物理アドレスと一致しない。
(2) 外部シンボルの値がアセンブル・リスト中では 0 になっており，実際の値は，リンク・マップを参照しなければならない。
(3) アセンブル・リスト中のリロケータブルな値は，実際の値と異なる。

上記の短所は，特にディバグ時や保守のためのドキュメント面で生産性の低下を招きます。
リスト・コンバータは，リロケータブル・アセンブラ・パッケージの上記の欠点を解決できます。

(1) リスト・コンバータの出力したアブソリュート・アセンブル・リストは，動作時のアドレスと完全に一致しています。
(2) 外部シンボルの実際の値がリスト上に埋め込まれます。
(3) リロケータブルな値がリスト上に実際の値として埋め込まれます。
(4) シンボル・テーブルあるいはクロスレファレンス・リスト上のシンボル値に対しても，実際の値が埋め込まれます。

例 1　ロケーションの埋め込み

＜アセンブル・リスト＞

```
21  2   ----                    CSEG
22  22  0000            START :
23  23
24  24                          ; chip initialize
25  25
26  26  0000    11201A          MOV     HDTSA , #1AH
27  27  0003    1620FE          MOVW    HL , #HDTSA     ; set hex 2-code data in HL registor
28  28
29  29  0006 R9A0000           CALL    !CONVAH         ; convert ASCII <- HEX
30  30                                                  ; output BC-register <- ASCII code
31  31  0009    1421FE          MOVW    DE , #STASC     ; set DE <- store ASCII code table
32  32  000C    63              MOV     A , B
33  33  000D    95              MOV     [ DE ] , A
34  34  000E    84              INCW    DE
35  35  000F    62              MOV     A , C
36  36  0010    95              MOV     [ DE ] , A
37  37
38  38  0011    FAFE            BR      $$
39  39
40  40                          END
```

＜アセンブソリュート・アセンブル・リスト＞

```
21  21  ----                    CSEG
22  22  0080            START :
23  23
24  24                          ; chip initialize
25  25
26  26  0080    11201A          MOV     HDTSA , #1AH
27  27  0083    1620FE          MOVW    HL , #HDTSA     ; set hex 2-code data in HL registor
28  28
29  29  0086 R9A9300           CALL    !CONVAH         ; convert ASCII<- HEX
30  30                                                  ; output BC-register <- ASCII code
31  31  0089    1421FE          MOVW    DE , #STASC     ; set DE <- store ASCII code table
32  32  008C    63              MOV     A , B
33  33  008D    95              MOV     [ DE ] , A
34  34  008E    84              INCW    DE
35  35  008F    62              MOV     A , C
36  36  0090    95              MOV     [ DE ] , A
37  37
38  38  0091    FAFE            BR      $$
39  39
40  40                          END
```

例 2　オブジェクト・コードの埋め込み

＜アセンブル・リスト＞

```
21  21   ----                    CSEG
22  22   0000            START:
23  23
24  24                           ; chip initialize
25  25
26  26   0000    11201A          MOV     HDTSA , #1AH
27  27   0003    1620FE          MOVW    HL , #HDTSA     ; set hex 2-code data in HL registor
28  28
29  29   0006 R9A0000            CALL    !CONVAH         ; convert ASCII <- HEX
30  30                                                   ; output BC-register <- ASCII code
31  31   0009    1421FE          MOVW    DE , #STASC     ; set DE <- store ASCII code table
32  32   000C    63              MOV     A , B
33  33   000D    95              MOV     [ DE ] , A
34  34   000E    84              INCW    DE
35  35   000F    62              MOV     A , C
36  36   0010    95              MOV     [ DE ] , A
37  37
38  38   0011    FAFE            BR      $$
39  39
40  40                           END
```

＜アセンブソリュート・アセンブル・リスト＞

```
21  21   ----                                CSEG
22  22   0080            START:
23  23
24  24                                                   ; chip initialize
25  25
26  26   0080    11201A          MOV     HDTSA , #1AH
27  27   0083    1620FE          MOVW    HL , #HDTSA     ; set hex 2-code data in HL registor
28  28
29  29   0086 R9A9300            CALL    !CONVAH         ; convert ASCII <- HEX
30  30                                                   ; output BC-register <- ASCII code
31  31   0089    1421FE          MOVW    DE , #STASC     ; set DE <- store ASCII code table
32  32   008C    63              MOV     A , B
33  33   008D    95              MOV     [ DE ] , A
34  34   008E    84              INCW    DE
35  35   008F    62              MOV     A , C
36  36   0090    95              MOV     [ DE ] , A
37  37
38  38   0091    FAFE            BR      $$
39  39
40  40                           END
```

9.3 リスト・コンバータの起動方法

9.3.1 リスト・コンバータの起動

リスト・コンバータの起動には，次の 2 つの方法があります。

(1) コマンド行での起動

```
X>lcnv78k0 [ △オプション ] …入力ファイル名 [ △オプション ] … [ △ ]
↑     ↑          ↑              ↑            ↑
(a)   (b)        (c)            (d)          (c)
```

(a) カレント・ドライブ名

(b) リスト・コンバータのコマンド・ファイル名

(c) リスト・コンバータに対して動作の詳細を指示します。
空白を含むパスを設定する場合には，ダブルクオーテーション（” “）で囲んでください。

(d) アセンブル・リストのプライマリ・ネーム
空白を含むパスのファイル名を指定する場合には，ダブルクオーテーション（” “）で囲んでください。

例 C>lcnv78k0 k0main –lk0.lmf

注意 1 上記 (c) で，複数のリストコンバータ・オプションを指定する場合には，それぞれのリスト・コンバータ・オプション間を空白で区切ってください。
なお，オプションの詳細については，「9.4 リスト・コンバータ・オプション」を参照してください。

注意 2 上記 (d) のファイルの拡張子は “ .prn ” にしてください。

注意 3 上記 (d) で，コマンド行にアセンブル・リストのプライマリ・ネームのみを指定する場合には，オブジェクト・モジュール・ファイル，ロード・モジュール・ファイルのプライマリ・ネームは，アセンブル・リスト・ファイルのプライマリ・ネームと同一でなければなりません。また，ファイル・タイプは次のようになっていなければなりません。

表 9-2 リスト・コンバータ起動時の指定ファイル・タイプ

ファイル名	タ イ プ
オブジェクト・モジュール・タイプ	.rel
ロード・モジュール・ファイル	.lmf

プライマリ・ネームの異なるファイルを指定する場合は，オプションを使用します。

(2) パラメータ・ファイルによる起動
パラメータ・ファイルは，起動に必要な情報がコマンド行に指定しきれない場合や，リスト・コンバートするたびに同じオプションを繰り返し指定するような場合に使用します。

パラメータ・ファイルを使用する場合には，コマンド行にパラメータ・ファイル指定オプション（-F）を指定します。

パラメータ・ファイルによる起動方法を次に示します。

```
C>lcnv78k0 [ △入力ファイル名 ] △ -f パラメータ・ファイル名
                                  ↑          ↑
                                 (a)        (b)
```

(a) パラメータ・ファイル指定オプション

(b) リスト・コンバータの起動に必要な情報を含んだファイル

備考　パラメータ・ファイルはエディタなどで作成してください。

パラメータ・ファイル内での記述規則を次に示します。

```
[ [ [ △ ] オプション [ △オプション ] … [ △ ] △ ] ] …
```

- コマンド行で入力ファイル名を省略した場合，パラメータ・ファイル内に入力ファイル名を記述します。
- 入力ファイル名は，オプションのあとにも記述できます。
- パラメータ・ファイルには，コマンド行で指定すべきすべてのリスト・コンバータ・オプション，出力ファイル名を記述します。

例　　パラメータ・ファイル（k0.plv）をエディタで作成します。

＜ k0.plv の内容＞

```
; parameter file
k0main –lk0.lmf
-ek0.elv
```

パラメータ・ファイル（k0.plv）を使用してリスト・コンバータを起動します。

```
C>lcnv78k0 –fk0.plv
```

9.3.2 実行開始メッセージ，終了メッセージ

(1) 実行開始メッセージ

リスト・コンバータが起動すると，次の実行開始メッセージが表示されます。

```
List Conversion Program for RA78K/0 Vx.xx [ xx xxx xx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

Pass1 : start ...
Pass2 : start ...
```

(2) 実行終了メッセージ

リスト・コンバートの結果，リスト・コンバート・エラーが検出されなかった場合，リスト・コンバータは次のメッセージを表示して制御を OS に戻します。

```
Conversion complete.
```

リスト・コンバート中に，リスト・コンバータ処理継続が不可能な致命的エラーが検出された場合，リスト・コンバータはメッセージを表示して処理を中止し，制御を OS に戻します。

例　存在しないリスト・コンバータ・オプションを指定した場合

```
C>lcnv78k0 k0main.prn -a

List Conversion Program for RA78K/0 Vx.xx [ xx xxx xx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

RA78K0 error F6018 : Option is not recognized ' -a '
Program aborted.
```

リスト・コンバータがエラー・メッセージを出力して処理を中止した場合は，そのエラー・メッセージの原因を「第 12 章 エラー・メッセージ」で調べて対処してください。

9.4 リスト・コンバータ・オプション

9.4.1 リスト・コンバータ・オプションの種類

リスト・コンバータ・オプションは，リスト・コンバータの動作に細かい指示を与えるものです。リスト・コンバータ・オプションは，6 種類のオプションに分類できます。

表 9-3 リスト・コンバータ・オプション

分類	オプション	説明
オブジェクト・モジュール・ファイル入力指定	-R	オブジェクト・モジュール・ファイルを入力します。
ロード・モジュール・ファイル入力指定	-L	ロード・モジュール・ファイルを入力します。
アブソリュート・アセンブル・リスト・ファイル出力指定	-O	アブソリュート・アセンブル・リスト・ファイルを出力します。
エラー・リスト・ファイル出力指定	-E	エラー・リスト・ファイルを出力します。
パラメータ・ファイル指定	-F	入力ファイル名，オプションを指定したファイルより入力します。
ヘルプ指定	--	ディスプレイにヘルプ・メッセージを表示します。

9.4.2 リスト・コンバータ・オプションの説明

次ページ以降に，各リスト・コンバータ・オプションの詳細について説明します。

(1) オブジェクト・モジュール・ファイル入力指定

オブジェクト・モジュール・ファイル入力指定 （-R）

【記述形式】

-R [入力ファイル名]

- 省略時解釈

  -R アセンブル・リスト・ファイル名 .rel

【機能】

- -R オプションは，オブジェクト・モジュール・ファイルの入力を指示します。

【用途】

- オブジェクト・モジュール・ファイルのプライマリ・ネームが，アセンブル・リスト・ファイルのプライマリ・ネームと異なる場合，またはファイル・タイプが“ .rel ”でない場合は，-R オプションを指定します。

【説明】

- フェータル・エラーがある場合は，アブソリュート・アセンブル・リスト・ファイルは出力されません。
- 入力ファイル名のプライマリ・ネームのみを指定した場合は，ファイル・タイプとして “ .rel ” を付加してファイルを入力します。

【使用例】

- アセンブル・リスト・ファイル名が k0main.prn で，オブジェクト・モジュール・ファイル名が sample.rel で，ロード・モジュール・ファイル名が k0.lmf の場合

```
C>lcnv78k0 k0main.prn -rsample.rel –lk0.lmf
```

(2) ロード・モジュール・ファイル入力指定

ロード・モジュール・ファイル入力指定 （-L）

【記述形式】

-L [入力ファイル名]

- 省略時解釈

  -L アセンブル・リスト・ファイル名 .lmf

【機能】

- -L オプションは，ロード・モジュール・ファイルの入力を指定します。

【用途】

- ロード・モジュール・ファイルのプライマリ・ネームがアセンブル・リスト・ファイルのプライマリ・ネームと異なる場合，またはファイル・タイプが “ .lmf ” でない場合に，-L オプションを指定します。

【説明】

- フェータル・エラーがある場合は，アブソリュート・アセンブル・リスト・ファイルは出力されません。
- 入力ファイル名のプライマリ・ネームのみを指定した場合は，ファイル・タイプとして “ .lmf ” を付加してファイルを入力します。

【使用例】

- アセンブル・リスト・ファイル名が k0main.prn で，ロード・モジュール・ファイル名が sample.lmf の場合

```
C>lcnv78k0 k0main.prn -lsample.lmf
```

(3) アブソリュート・アセンブル・リスト・ファイル出力指定

アブソリュート・アセンブル・リスト・ファイル出力指定 （-O）

【記述形式】

-O [出力ファイル名]

- 省略時解釈

  -O アセンブル・リスト・ファイル名 .p

【機能】

- -O オプションは，アブソリュート・アセンブル・リスト・ファイルの出力を指定します。
- また，その出力先や出力ファイル名も指定できます。

【用途】

- アブソリュート・アセンブル・リスト・ファイルの出力先や出力ファイル名を変更したいときに，-O オプションを指定します。

【説明】

- ファイル名として，ディスク型ファイル名とデバイス型ファイル名を指定できます。指定できるデバイス型ファイル名はCON , PRN , NUL，およびAUXです。CLOCKを指定した場合，アボート・エラーとなります。
- ファイル名にエラー・ファイルと同一のデバイスが指定された場合，アボート・エラーとなります。
- -O オプションを指定する際に出力ファイル名を省略するとアブソリュート・アセンブル・リスト・ファイル名は，“ アセンブル・リスト・ファイル名 .p ” となります。
- 出力ファイル名のプライマリ・ネームのみを指定した場合は，ファイル・タイプとして，“ .p ” を付加してファイルを出力します。
- -O オプションを指定する際にドライブ名を省略すると，カレント・ドライブにアブソリュート・アセンブル・リスト・ファイルが出力されます。

【使用例】

- アブソリュート・アセンブル・リスト・ファイル（sample.p）を作成します。

```
C>lcnv78k0 k0main.prn -osample.p –lk0.lmf
```

(4) エラー・リスト・ファイル出力指定

エラー・リスト・ファイル出力指定 （-E/-NE）

【記述形式】

```
-E [ 出力ファイル名 ]
-NE
```

- **省略時解釈**

  -NE

【機能】

- -E **オプションは，エラー・リスト・ファイルの出力を指定します。また，その出力先や出力ファイル名も指定できます。**
- -NE **オプションは，**-E **オプションを無効にします。**

【用途】

- **エラー・メッセージをファイルに保存しておきたい場合には，**-E **オプションを指定します。**

【説明】

- **ファイル名として，ディスク型ファイル名とデバイス型ファイル名を指定できます。ただし，デバイス型ファイル名として，**CLOCK **を指定した場合は，アボート・エラーとなります。**
- **ファイル名にアブソリュート・アセンブル・リスト・ファイルと同一のデバイスが指定された場合，アボート・エラーとなります。**
- -E **オプションを指定する際に出力ファイル名を省略するとエラー・リスト・ファイル名は，“ アセンブル・リスト・ファイル名** .elv **” となります。**
- **出力ファイル名のプライマリ・ネームのみを指定した場合は，ファイル・タイプとして “** .elv **” を付加してファイルを出力します。**
- -E **オプションを指定する際にドライブ名を省略するとカレント・ドライブにエラー・リスト・ファイルが出力されます。**
- -E **と** -NE **オプションを同時に指定した場合は，あとで指定した方が有効となります。**

【使用例】

- **エラー・リスト・ファイル（**sample.elv**）を作成します。**

  ```
  C>lcnv78k0 k0main.prn -esample.elv
  ```

  エラー・リスト・ファイル（sample.elv**）を参照します。**

```
RA78K0 warning W6701: Load module file is older than object module file 'K0MAIN.LMF , K0MAIN.REL'
Pass1: start
RA78K0 error F6105: Segment name is not found is load module file 'DATA'
```

(5) パラメータ・ファイル指定

パラメータ・ファイル指定 （-F）

【記述形式】

-F ファイル名

- 省略時解釈

  入力ファイルなし

【機能】

- -F オプションは，オプションあるいは入力ファイル名を指定のファイルから入力する指定をします。

【用途】

- コマンド行ではリスト・コンバータの起動に必要な情報を指定しきれないときに，-F オプションを指定します。
- リスト・コンバートするたびに繰り返し同じようにオプションを指定し，リスト・コンバートする場合には，それらをパラメータ・ファイルに記述しておき，-F オプションを指定します。

【説明】

- “ ファイル名 ”として指定できるのは，ディスク型ファイル名のみです。デバイス型ファイル名を指定するとアボート・エラーとなります。
- ファイル名を省略するとアボート・エラーとなります。
- ファイル名のプライマリ・ネームのみを指定した場合は，ファイル・タイプとして“ .plv ”を付加してファイルをオープンします。
- パラメータ・ファイルのネストは許されません。パラメータ・ファイル中で，-F オプションを指定するとアボート・エラーとなります。
- パラメータ・ファイル中に記述できる文字数の制限はありません。
- 空白とタブ，および改行文字（LF）をオプションあるいは入力ファイル名の区切りとします。
- パラメータ・ファイル中に記述したオプションあるいは入力ファイル名はコマンド行上のパラメータ・ファイル指定のあった位置に展開されます。
- 展開されたオプションは，あとで指定した方が有効となります。
- -F オプションを複数指定するとアボート・エラーとなります。
- “ ; ”，または“ # ”以降に記述された文字は改行文字（LF），または EOF の前まですべてコメントと解釈します。

【使用例】

- パラメータ・ファイルを使用してリスト・コンバータを起動させます

  パラメータ・ファイル（k0.plv）の内容

```
: parameter file
k0main –lk0.lmf
-ek0.elv
```

コマンド行には，次のように入力します

```
C>lcnv78k0 –fk0.plv
```

(6) ヘルプ指定

ヘルプ指定 (--)

【記述形式】

```
--
```

- **省略時解釈**

  表示しない

【機能】

- **-- オプションは，ヘルプ・メッセージをディスプレイに出力します。**

【用途】

- **ヘルプ・メッセージは，リスト・コンバータ・オプションとその説明の一覧です。リスト・コンバータを実行するときに参照してください。**

【説明】

- **-- オプションを指定すると他のリスト・コンバータ・オプションは，すべて無効となります。**

注意 このオプションは，PM plus **上では指定できません。**

PM plus **上でヘルプを参照する場合は，〈リストコンバータオプションの設定〉ダイアログで［ヘルプ］ボタンをクリックしてください。**

【使用例】

- **-- オプションを指定するとヘルプ・メッセージがディスプレイに出力されます。**

```
C>lcnv78k0 --

List Conversion Program for RA78K/0 Vx.xx [ xx xxx xx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

usage : LCNV78K0 [ option [ ... ] ] input-file [ option [ ... ] ]
The option is as follows ( [ ] means omissible ).
-R [ file ] : Specify object module file.
-L [ file ] : Specify load module file.
-O [ file ] : Specify output list file ( absolute assemble list file ).
-Ffile      : Input option or input-file name from specified file.
-E [ file ] : Create error list file.
--          : Show this message.
```

9.5 PM plus でのオプション設定

PM plus からリストコンバータ・オプションを設定する方法について説明します。

9.5.1 オプションの設定方法

PM plus の [ツール (T)] メニューの [リストコンバータオプションの設定 (N)] を選択するか，ツール・バーの [LC] ボタンを押下すると，〈リストコンバータオプションの設定〉ダイアログが現れます。

ダイアログ内で必要なオプションを入力することにより，各リストコンバータ・オプションを設定できます。

図 9-2 〈リストコンバータオプションの設定〉ダイアログ（《出力》タブ選択時）

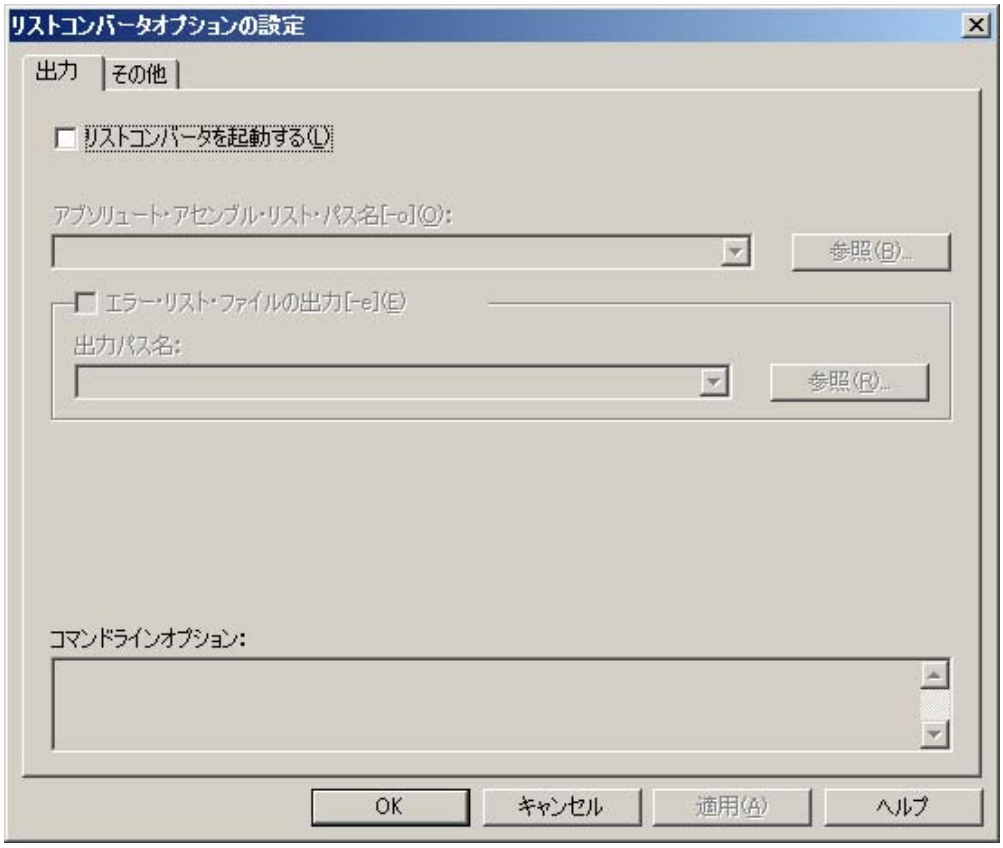
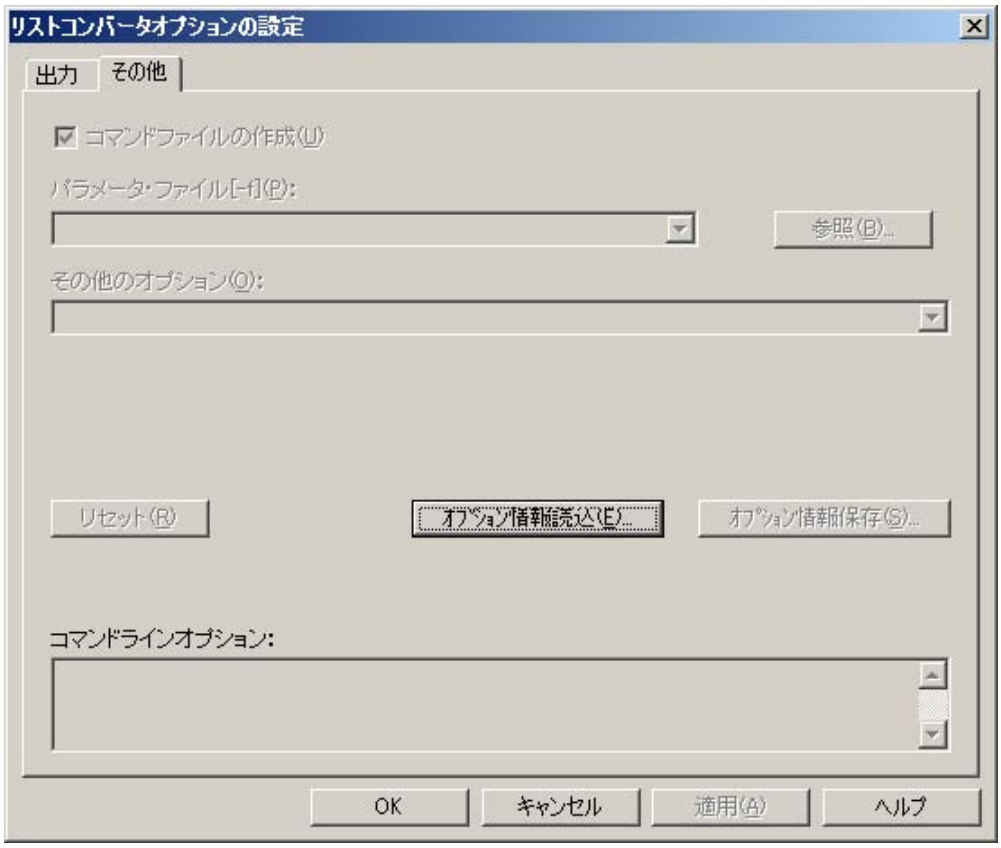


図 9-3 〈リストコンバータオプションの設定〉ダイアログ（《その他》タブ選択時）



9.5.2 各オプションの設定

〈リストコンバータオプションの設定〉ダイアログの各オプションについて，次に説明します。

《出力》タブ

- リストコンバータを起動する (L)
  リスト・コンバータを起動する場合に，チェックします。
- アブソリュート・アセンブル・リスト・パス名 [-o](O)
  [参照 (B)] ボタン，または直接入力により，アブソリュート・アセンブル・リストのパスを指定します。
- エラー・リスト・ファイルの出力 [-e](E)
  エラー・リスト・ファイルを出力する場合に，チェックします。
- 出力パス名：
  [参照 (R)] ボタン，または直接入力により，エラー・リスト・ファイルのパスを指定します。
- コマンドラインオプション
  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

《その他》タブ

- コマンドファイルの作成 (U)
  コマンドファイルを作成する場合に，チェックします。
- パラメータ・ファイル [-f](P)
  [参照 (B)] ボタン，または直接入力により，ユーザ定義のパラメータ・ファイルとして入力するファイルを指定します。
- その他のオプション (O)
  ダイアログで設定可能なオプション以外のオプションを指定したい場合に入力ボックスに入力します。

  注意　ヘルプ指定 (--) オプションは，PM plus 上では指定できません。
- リセット (R)
  入力した内容をリセットします。
- オプション情報読込 (F)
  〈オプション情報読込み〉ダイアログが開き，オプション情報ファイルを指定後，読み込みます。
- オプション情報保存 (S)
  〈オプション情報の保存〉ダイアログが開き，オプション情報ファイルに名前をつけて保存します。
- コマンドラインオプション
  このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

第 10 章　プログラムの出力リスト

この章では，次に示す各プログラムが出力する各種リストのフォーマットなどについて説明します。

- 構造化アセンブラ・プリプロセッサの出力リスト
  エラー・リスト
- アセンブラの出力リスト
  アセンブル・リスト・ファイルのヘッダ
  アセンブル・リスト
  シンボル・リスト
  クロスレファレンス・リスト
  エラー・リスト
- リンカの出力リスト
  リンク・リスト・ファイルのヘッダ
  マップ・リスト
  パブリック・シンボル・リスト
  ローカル・シンボル・リスト
  エラー・リスト
- オブジェクト・コンバータの出力リスト
  エラー・リスト
- ライブラリアンの出力ファイル
  ライブラリ情報出力リスト
- リスト・コンバータの出力リスト
  アブソリュート・アセンブル・リスト
  エラー・リスト

10.1　構造化アセンブラ・プリプロセッサの出力リスト

構造化アセンブラ・プリプロセッサは，次のリストを出力します。

表 10-1 構造化アセンブラ・プリプロセッサの出力ファイル

出力リスト・ファイル名	出力リスト名
エラー・リスト・ファイル	エラー・リスト

10.1.1 エラー・リスト

構造化アセンブラ・プリプロセッサ起動時に出力されたエラー・メッセージが格納されます。

【出力形式】

Start

(1) TTT.S ((2) 4) : RA78K0 (3) error (4) E1221: (5) Missing ENDIF

【出力項目の説明】

表 10-2 エラーリストの出力項目の説明（構造化アセンブラ・プリプロセッサ起動時）

項目	内容
(1)	エラーの発生したソース・モジュール・ファイル名
(2)	エラーの発生行
(3)	エラーの種類
(4)	エラー番号
(5)	エラー・メッセージ

補足　ファイル名，エラーの発生行は表示されない場合もあります。

10.2 アセンブラの出力リスト

アセンブラは，次のリストを出力します。

表 10-3 アセンブラの出力リスト

出力リスト・ファイル名	出力リスト名
アセンブル・リスト・ファイル	アセンブル・リスト
	シンボル・リスト
	クロスレファレンス・リスト
エラー・リスト・ファイル	エラー・リスト

10.2.1 アセンブル・リスト・ファイルのヘッダ

ヘッダ部は，常にアセンブル・リスト・ファイルの先頭に出力されます。

【出力形式】

```
78K/0 Series Assembler (1) Vx.xx (2)          Date:(3) xx xxx xxxx Page: (4)   1
(5)
Command: (6) k0main.asm –c054
Para-file: (7)
In-fine:   (8)   K0MAIN.ASM
Obj-file:  (9)   K0MAIN.REL
Prn-file:  (10) K0MAIN.PRN
```

【出力項目の説明】

表 10-4 アセンブル・リスト・ファイルのヘッダの出力項目の説明

項目	内容
(1)	アセンブラのバージョン番号
(2)	タイトル文字列 -LH オプション，または TITLE 制御命令によって指定された文字列
(3)	アセンブル・リストの作成年月日
(4)	ページ番号
(5)	サブタイトル文字列 SUBTITLE 制御命令によって指定された文字列
(6)	コマンド行のイメージ
(7)	パラメータ・ファイルの内容
(8)	入力ソース・モジュール・ファイル名
(9)	出力オブジェクト・モジュール・ファイル名
(10)	アセンブル・リスト・ファイル名

10.2.2 アセンブル・リスト

アセンブル・リストは，アセンブル結果をエラー・メッセージ（エラーがある場合のみ）とともに出力します。

【出力形式】

```
          Assemble list

ALNO   STNO   ADRS    OBJECT (3) M   (4) I     SOURCE STATEMENT
(1) 1  (2) 1
(2) 2  (2) 2                          (5)       NAME    SAMPM
  :
28     28
29     29     (6) 0006 (8) R220000 (5)         CALL    !CONVAH          ; convert ASCII <- HEX
30     30                            (5)                                 ; output BC-register <-
                                                                           ASCII code
31     31     (6) 0009    00000000              MOV       DE , #STASC    ; set DE <- store
                                                                           ASCII code table
(7) **   ERROR E2202 , STNO      31 (   0 ) Illegal operand
                (6) 000D              00
32     32     (6) 000E (8) 0A27      (5)       MOV       A , B
33     33     (6) 0010 (8) EB        (5)       MOV       [ DE ] , A
  :
Segment informations :

ADRS   LEN     NAME

(9) FE20 (10) 0003H (11) DATA
(9) 0000 (10) 0002H (11) CODE
(9) 0000 (10) 0017H (11) ?CSEG

Target chip :(12) uPD78xxx
Device file :(13) Vx. xx
Assembly complete ,     (14) 1 error ( s ) and     (15) 0 warning ( s ) found. (   (16) 31 )
```

【出力項目の説明】

表 10-5 アセンブル・リストの出力項目の説明

項目	内容
(1)	ソース・モジュールのイメージの行番号
(2)	行番号（INCLUDE ファイルの展開，マクロ展開も含みます）
(3)	マクロ表示 M　：マクロ定義行です。 #n　：マクロ展開行です。n はネスト・レベルです。 空白　：マクロ定義行／マクロ展開行ではありません。

表 10-5 アセンブル・リストの出力項目の説明

項目	内容
(4)	INCLUDE 表示 In ：INCLUDE ファイル中です。n はネスト・レベルです。 空白 ：INCLUDE ファイル未使用
(5)	ソース・プログラム・ステートメント
(6)	ロケーション・カウンタ値（4 桁または 5 桁）
(7)	エラーの発生行
(8)	リロケーション情報 R ：リンカによってオブジェクト・コード，またはシンボル値が変更されます。 空白 ：オブジェクト・コード，またはシンボル値が変更されません。
(9)	セグメント・アドレス（4 桁または 5 桁）
(10)	セグメント・サイズ（4 桁または 5 桁）
(11)	セグメント名
(12)	RA78K0 の対象デバイス
(13)	デバイス・ファイルのバージョン番号
(14)	フェータル・エラーの個数
(15)	ワーニングの個数
(16)	最終エラー行

10.2.3 シンボル・リスト

ソース・モジュール内で定義されているシンボル（ローカル・シンボルを含む）の情報を出力します。

【出力形式】

```
Symbol Table List

  VALUE        ATTR     RTYP   NAME          VALUE   ATTR      RTYP   NAME
               (2) CSEG         (4) ?CSEG             (2) CSEG        (4) CODE
(1)  ----H                 (3) EXT (4) CONVAH           (2) DSEG        (4) DATA
(1)  FE20H     ADDR              (4) HDTSA     (1)  0H (2) ADDR  (3) PUB (4) MAIN
               MOD               (4) SAMPM     (1)  0H (2) ADDR  (3) PUB (4) START
(1)  FE21H     ADDR              (4) STASC
```

【出力項目の説明】

表 10-6 シンボル・リストの出力項目の説明

項目	内容	
(1)	シンボル値（4 桁または 5 桁）	
(2)	シンボル属性 CSEG：コード・セグメント名 DSEG：データ・セグメント名 BSEG：ビット・セグメント名 MAC ：マクロ名 MOD ：モジュール名 SET ：SET 疑似命令によって定義されたシンボル NUM ：NUMBER 属性シンボル	ADDR：ADDRESS 属性シンボル BIT ：BIT 属性シンボル（addr.bit） SABIT：BIT 属性シンボル（saddr.bit） SFBIT：BIT 属性シンボル（sfr.bit） RBIT ：BIT 属性シンボル（A.bit , X.bit , PSW.bit） 空白 ：EXTRN，または EXTBIT 宣言された外部参照シンボル ***** ：未定義シンボル
(3)	シンボル参照形式 EXT ：EXTRN 宣言された外部参照シンボル（SADDR 属性） EXTB ：EXTBIT 宣言された外部参照シンボル（saddr.bit） PUB ：PUBLIC 宣言された外部定義シンボル 空白 ：ローカル・シンボル , セグメント名 , マクロ名 , モジュール名 ***** ：未定義シンボル	
(4)	定義されたシンボル名	

10.2.4 クロスレファレンス・リスト

ソース・モジュール内で定義されたシンボルがソース・モジュールのどこで（行番号）参照されているかという情報が出力されます。

【出力形式】

Cross-Reference List

NAME	VALUE	R	ATTR	RTYP	SEGNAME	XREFS		
(1) ?CSEG			(4) CSEG		(6) ?CSEG	(7) 21#		
(1) CODE			(4) CSEG		(6) CODE	(7) 18#		
(1) CONVAH	(2) ----H	(3) E		(5) EXT		(7) 12	29	
(1) DATA			(4) DSEG		(6) DATA	(7) 14#		
(1) HDTSA	(2) FE20H		(4) ADDR		(6) DATA	(7) 15#	26	
(1) MAIN	(2) 0H		(4) ADDR	(5) PUB	(6) CODE	(7) 11@	19#	
(1) SAMPM			(4) MOD			(7) 2#		
(1) START	(2) 0H	(3) R	(4) ADDR	(5) PUB	(6) ?CSEG	(7) 11@	19	22#
(1) STASC	(2) FE21H		(4) ADDR		(6) DATA	(7) 16#	31	

【出力項目の説明】

表 10-7 クロスレファレンス・リストの出力項目の説明

項目	内容	
(1)	定義されたシンボル名	
(2)	シンボル値（4 桁または 5 桁）	
(3)	リロケーション属性 R ：リロケータブルなシンボル E ：external なシンボル 空白 ：アブソリュートなシンボル * ：未定義シンボル	
(4)	シンボル属性 CSEG：コード・セグメント名 DSEG：データ・セグメント名 BSEG：ビット・セグメント名 MAC ：マクロ名 MOD ：モジュール名 SET ：SET 疑似命令によって定義されたシンボル NUM ：NUMBER 属性シンボル	ADDR：ADDRESS 属性シンボル BIT ：BIT 属性シンボル（addr.bit） SABIT：BIT 属性シンボル（saddr.bit） SFBIT：BIT 属性シンボル（sfr.bit） RBIT ：BIT 属性シンボル（A.bit , X.bit , PSW.bit） 空白 ：EXTRN，または EXTBIT 宣言された外部参照シンボル ***** ：未定義シンボル
(5)	シンボル参照形式 EXT ：EXTRN 宣言された外部参照シンボル（SADDR 属性） EXTB ：EXTBIT 宣言された外部参照シンボル（saddr.bit） PUB ：PUBLIC 宣言された外部定義シンボル 空白 ：ローカル・シンボル , セグメント名 , マクロ名 , モジュール名 ***** ：未定義シンボル	
(6)	定義されたシンボル名	

表 10-7 クロスレファレンス・リストの出力項目の説明

項目	内容
(7)	**定義・参照行番号** **定義行：**×××××# **参照行：**××××× △ **（△は空白 1 つ）** EXTRN **宣言** , EXTBIT **宣言** , PUBLIC **宣言：**×××××@

10.2.5 エラー・リスト

アセンブラ起動時に出力されたエラー・メッセージが格納されています。

【出力形式】

```
   Pass1 Start
(1) ERROR.ASM ( (2) 26 ) : RA78K0 (3) error (4) E2202 :      (5) Illegal operand
(1) ERROR.ASM ( (2) 32 ) : RA78K0 (3) error (4) E2202 :      (5) Illegal operand
   Pass2 Start
(1) ERROR.ASM ( (2) 26 ) : RA78K0 (3) error (4) E2202 :      (5) Illegal operand
(1) ERROR.ASM ( (2) 29 ) : RA78K0 (3) error (4) E2407 :      (5) Undefined symbol reference ' DTSA '
(1) ERROR.ASM ( (2) 29 ) : RA78K0 (3) error (4) E2303 :      (5) Illegal expression
(1) ERROR.ASM ( (2) 32 ) : RA78K0 (3) error (4) E2202 :      (5) Illegal operand
(1) ERROR.ASM ( (2) 37 ) : RA78K0 (3) error (4) E2407 :      (5) Undefined symbol reference ' F '
(1) ERROR.ASM ( (2) 37 ) : RA78K0 (3) error (4) E2303 :      (5) Illegal expression
```

【出力項目の説明】

表 10-8 エラーリストの出力項目の説明（アセンブラ起動時）

項目	内容
(1)	エラーの発生したソース・モジュール・ファイル名
(2)	エラーの発生行
(3)	エラーの種類
(4)	エラー番号
(5)	エラー・メッセージ

補足　ファイル名，エラーの発生行は表示されない場合もあります。

10.3 リンカの出力リスト

リンカは，次のリストを出力します。

表 10-9 リンカの出力リスト

出力リスト・ファイル名	出力リスト名
リンク・リスト・ファイル	マップ・リスト
	パブリック・シンボル・リスト
	ローカル・シンボル・リスト

10.3.1 リンク・リスト・ファイルのヘッダ

ヘッダ部は常にリンク・リスト・ファイルの先頭に出力されます。

【出力形式】

```
78K/0 Series Linker (1) Vx.xx                    Date :(2) xx xxx xxxx Page : (3) 1

Command :       (4) k0main.rel k0sub.rel –ok0.map –dk0.dr
Para-file :     (5)
Out-file :      (6) K0.MAP
Map-File :      (7) K0MAIN.MAP
Direc-File :    (8)
Directive :     (9)

*** Link information ***
(10)  3 output segment ( s )
(11)  37H byte ( s ) real data
(12)  23   symbol ( s ) defined
```

【出力項目の説明】

表 10-10 リンク・リスト・ファイルのヘッダの出力項目の説明

項目	内容
(1)	リンカのバージョン番号
(2)	リンク・リスト・ファイルの作成年月日
(3)	ページ番号
(4)	コマンド行のイメージ
(5)	パラメータ・ファイルの内容
(6)	出力ロード・モジュール・ファイル名
(7)	リンク・リスト・ファイル名
(8)	ディレクティブ・ファイル名

表 10-10 リンク・リスト・ファイルのヘッダの出力項目の説明

項目	内容
(9)	ディレクティブ・ファイルの内容
(10)	ロード・モジュール・ファイルに出力されるセグメント数
(11)	ロード・モジュール・ファイルに出力されるデータの大きさ
(12)	ロード・モジュール・ファイルに出力されるシンボル数

10.3.2 マップ・リスト

セグメントの配置に関する情報を出力します。

【出力形式】

```
      *** Memory map ***

(1)   SPACE = REGULAR

  MEMORY = (2) ROM
  BASE ADDRESS = (3) 0000H              SIZE = (4) 2000H
          OUTPUT          INPUT         INPUT          BASE          SIZE
          SEGMENT         SEGMENT       MODULE         ADDRESS
          (6) CODE                                     (9) 0000H     (10) 0002H
                                                                          (11) CSEG AT
                          (7) CODE      (8) SAMPM      (9) 0000H     (10) 0002H
(5) *gap *                                             (9) 0002H     (10) 007EH
          (6) ?CSEG                                    (9) 0080H     (10) 0035H
                                                                          (11) CSEG
                          (7) ?CSEG     (8) SAMPM      (9) 0080H     (10) 0015H
                          (7) ?CSEG     (8) SAMPS      (9) 0095H     (10) 0020H
(5) *gap *                                             (9) 00B5H     (10) 1F4BH

      MEMORY = RAM
      BASE ADDRESS = (3) FE00H          SIZE = (4) 0200H
          OUTPUT          INPUT         INPUT          BASE          SIZE
          SEGMENT         SEGMENT       MODULE         ADDRESS
(5) * gap *                                            (9) FE00H     (10) 0020H
          (6) DATA                                     (9) FE20H     (10) 0003H
                                                                          (11) DSEG AT
                          (7) DATA      (8) SAMPM      (9) FE20H     (10) 0003H
(5) *gap *                                             (9) FE23H     (10) 00DDH
(5) *gap ( Not Free Area ) *                           (9) FE00H     (10) 0100H

   Target chip : (12) uPD78xxx
   Device File : (13) Vx.xx
```

【出力項目の説明】

表 10-11 マップ・リストの出力項目の説明

項目	内容
(1)	**メモリ空間名**
(2)	**メモリ領域名**
(3)	**メモリ領域の先頭アドレス（4 桁または 5 桁）**

表 10-11 マップ・リストの出力項目の説明

項目	内容
(4)	メモリ領域のサイズ（4 桁または 5 桁）
(5)	出力グループ 何も配置されていないエリアがある場合 “ gap ” を表示します。
(6)	ロード・モジュール・ファイルに出力されるセグメント名
(7)	オブジェクト・モジュール・ファイルから読み込まれたセグメント名
(8)	入力モジュール名
(9)	セグメントの先頭アドレス（4 桁または 5 桁）
(10)	出力セグメント／入力セグメントのサイズ（4 桁または 5 桁）
(11)	セグメント・タイプ，再配置属性
(12)	このアセンブルの対象製品
(13)	デバイス・ファイルのバージョン番号

10.3.3 パブリック・シンボル・リスト

入力モジュール内で定義されているパブリック・シンボルの情報を出力します。

【出力形式】

```
*** Public symbol list ***

MODULE          ATTR          VALUE          NAME

(1) SAMPM       (2) ADDR      (3) 0000H      (4) MAIN
(1) SAMPM       (2) ADDR      (3) 0080H      (4) START
(1) SAMPS       (2) ADDR      (3) 0095H      (4) CONVAH
```

【出力項目の説明】

表 10-12 パブリック・シンボル・リストの出力項目の説明

項目	内容	
(1)	パブリック・シンボルが定義されたモジュール名	
(2)	シンボル属性 CSEG：コード・セグメント名 DSEG：データ・セグメント名 BSEG：ビット・セグメント名 MAC ：マクロ名 MOD ：モジュール名 SET ：SET 疑似命令によって定義されたシンボル NUM ：NUMBER 属性シンボル	ADDR：ADDRESS 属性シンボル BIT ：BIT 属性シンボル（addr.bit） SABIT：BIT 属性シンボル（saddr.bit） SFBIT：BIT 属性シンボル（sfr.bit） RBIT ：BIT 属性シンボル（A.bit ，X.bit ，PSW.bit） 空白 ：EXTRN または EXTBIT 宣言された外部参照シンボル ***** ：未定義シンボル
(3)	シンボル値（4 桁または 5 桁）	
(4)	パブリック・シンボル名	

10.3.4 ローカル・シンボル・リスト

入力モジュール内で定義されているローカル・シンボルの情報を出力します。

【出力形式】

*** Local symbol list ***

MODULE	ATTR	VALUE	NAME
(1) SAMPM	(2) MOD		(4) SAMPM
(1) SAMPM	(2) DSEG		(4) DATA
(1) SAMPM	(2) ADDR	(3) FE20H	(4) HDTSA
(1) SAMPM	(2) ADDR	(3) FE21H	(4) STASC
(1) SAMPM	(2) CSEG		(4) CODE
(1) SAMPM	(2) CSEG		(4) ?CSEG
(1) SAMPS	(2) MOD		(4) SAMPS
(1) SAMPS	(2) CSEG		(4) ?CSEG
(1) SAMPS	(2) ADDR	(3) 00ACH	(4) SASC
(1) SAMPS	(2) ADDR	(3) 00B2H	(4) SASC1

【出力項目の説明】

表 10-13 ローカル・シンボル・リストの出力項目の説明

項目	内容	
(1)	ローカル・シンボルが定義されたモジュール名	
(2)	シンボル属性 CSEG：コード・セグメント名 DSEG：データ・セグメント名 BSEG：ビット・セグメント名 MAC ：マクロ名 MOD ：モジュール名 SET ：SET 疑似命令によって定義されたシンボル NUM ：NUMBER 属性シンボル	ADDR：ADDRESS 属性シンボル BIT ：BIT 属性シンボル（addr.bit） SABIT：BIT 属性シンボル（saddr.bit） SFBIT：BIT 属性シンボル（sfr.bit） RBIT ：BIT 属性シンボル（A.bit ，X.bit ，PSW.bit） 空白 ：EXTRN または EXTBIT 宣言された外部参照シンボル ***** ：未定義シンボル
(3)	シンボル値（4 桁または 5 桁）	
(4)	ローカル・シンボル名	

10.3.5 エラー・リスト

リンカ起動時に出力されたエラー・メッセージが格納されています。

【出力形式】

```
RA78K0 (1) error          (2) E3405 :       (3) Undefined symbol ' CONVAH ' in file ' K0MAIN.REL '
```

【出力項目の説明】

表 10-14 エラーリストの出力項目の説明（リンカ起動時）

項目	内容
(1)	**エラーの種類**
(2)	**エラー番号**
(3)	**エラー・メッセージ**

10.4 オブジェクト・コンバータの出力リスト

オブジェクト・コンバータは，次のリストを出力します。

表 10-15 オブジェト・コンバータの出力リスト

出力リスト・ファイル名	出力リスト名
エラー・リスト・ファイル	エラー・リスト

10.4.1 エラー・リスト

オブジェクト・コンバータ起動時に出力されたエラー・メッセージが格納されています。

【出力形式】

リンカの出力するエラー・リストと同一です。

10.5 ライブラリアンの出力リスト

ライブラリアンは，次のリストを出力します。

表 10-16 ライブラリアンの出力リスト

出力リスト・ファイル名	出力リスト名
リスト・ファイル	ライブラリ情報出力リスト

10.5.1 ライブラリ情報出力リスト

ライブラリ・ファイル内のモジュールに関する情報を出力します。

【出力形式】

```
78K/0 Series librarian (1) Vx.xx      DATE : (2) xx xxx xx              PAGE (3) 1
  LIB-FILE NAME : (4) K0.LIB          ( (5) xx xxx xx )
(6) 0001   (7) K0MAIN.REL             ( (8) xx xxx xx )
   (9) MAIN                                  (9) START
   NUMBER OF PUBLIC SYMBOLS :   (10) 2
(6) 0002   (7) K0SUB.REL              ( (8) xx xxx xx )
   (9) CONVAH
   NUMBER OF PUBLIC SYMBOLS :   (10) 1
```

【出力項目の説明】

表 10-17 ライブラリ情報出力リストの出力項目の説明

項目	内容
(1)	ライブラリアンのバージョン番号
(2)	リストの作成年月日
(3)	ページ数
(4)	ライブラリ・ファイル名
(5)	ライブラリ・ファイルの作成年月日
(6)	モジュールの通番（0001 から番号を付けます）
(7)	モジュール名
(8)	モジュール作成年月日
(9)	パブリック・シンボル名
(10)	モジュール内で定義されているパブリック・シンボル数

10.6 リスト・コンバータの出力リスト

リスト・コンバータは，次のリストを出力します。

表 10-18 リスト・コンバータの出力リスト

出力リスト・ファイル名	出力リスト名
アブソリュート・アセンブル・リスト・ファイル	アブソリュート・アセンブル・リスト
エラー・リスト・ファイル	エラー・リスト

10.6.1 アブソリュート・アセンブル・リスト

アセンブル・リストにアブソリュートな値を埋め込んで出力します。

【出力形式】

アセンブラの出力するアセンブル・リストと同一です。

10.6.2 エラー・リスト

リスト・コンバータ起動時に出力されたエラー・メッセージが格納されています。

【出力形式】

アセンブラの出力するエラー・リストと同一です。

第 11 章　RA78K0 の活用法

この章では，RA78K0 を効率よく使用するための方法を紹介します。

11.1　作業の効率化（EXIT ステータス機能）

RA78K0 の各プログラムは，処理終了時に，処理中に発生した最大のエラー・レベルを EXIT ステータスとして，OS に返します。

EXIT ステータスを次に示します。

- 正常終了時　：0
- WARNING あり　：0
- FATAL ERROR あり　：1
- ABORT 時　：2

これらを利用してバッチ・ファイルを作成することで効率良く作業できます。

【使 用 例】

＜バッチ・ファイル（ra.bat）の内容＞

```
ra78K0 –c014 k0main.-g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
ra78K0 –c014 k0sub.asm -g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
lk78K0 k0main.rel k0sub.rel –ok0.lmf -g
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
oc78K0 k0.lmf
echo off
IF ERRORLEVEL 1 GOTO ERR
GOTO EXIT
:ERR
echo　エラーが発生しました。
: EXIT
```

バッチ・ファイル（ra.bat**）を使用して処理を行います。**

```
C>ra.bat
```

11.2 開発環境の整備（環境変数）

RA78K0 では開発環境を整えるため，次の環境変数をサポートしています。

PATH	：実行形式のサーチ・パス
INC78K0	：インクルード・ファイルのサーチ・パス（構造化アセンブラ・プリプロセッサ，アセンブラ）
LIB78K0	：ライブラリ・ファイルのサーチ・パス（リンカのみ）
TMP	：テンポラリ・ファイルを作成するパス
LANG78K	：漢字種別の指定

【使用例】

＜ autoexec.bat の内容＞

```
; AUTOEXEC.BAT
Verify on
break on
PATH C:\BIN ; C:\BAT ; C:\RA78K0 ;          ← (1)
SET INC78K0 = C:\RA78K0\INCLUDE             ← (2)
SET LIB78K0 = C:\RA78K0\LIB                 ← (3)
SET TMP = C:\TMP                            ← (4)
SET LANG78K = SJIS                          ← (5)
```

(1) パス指定により，C：\BIN , C：\BAT , C：\RA78K0 という順に実行形式ファイルを検索します。

(2) アセンブラは，インクルード・ファイルを C：\RA78K0\INCLUDE から検索します。

(3) リンカは，ライブラリ・ファイルを C：\RA78K0\LIB から検索します。

(4) 各プログラムは，テンポラリ・ファイルを C：\TMP に作成します。

(5) コメント文の漢字を，シフト JIS コードとして解釈します。

注意 Windows インストーラでインストールした場合には，必要な環境変数が自動的に設定されます。

11.3 プログラム実行の中断

キー入力（CTRL-C）により各プログラムの実行を中断できます。

バッチ・ファイル中で “ break on ” を指定した場合は，キー入力のタイミングに関係なしに制御を OS に戻し，“ break off ” を指定した場合には画面表示中にのみ制御を OS に戻します。そして，オープン中のすべてのテンポラリ・ファイル，出力ファイルを削除します。

11.4 アセンブル・リストを見やすくする

-LH オプションや TITLE 制御命令を使用してアセンブル・リストのヘッダにタイトルを表示します。アセンブル・リストの内容を端的に表すようなタイトルを表示しておくことでアセンブル・リストの内容がわかりやすくなります。

また，SUBTITLE 制御命令を使用すればサブタイトルが表示できます。制御命令については，「RA78K0 アセンブラ・パッケージ 言語編」のユーザーズ・マニュアルを参照してください。

【使用例】

アセンブル・リスト・ファイルのヘッダにタイトルを印字します。

```
C>ra78k0 –c054 k0main.asm -lhRA78K0_MAINROUTINE
```

k0main.prn を参照します。

```
78K/0 Series Assembler Ex.xx RA78K0_MAINROUTINE          Date : xx xxx xxxx Page :   1
                                        |
                                     タイトル

Command : -c054 k0main.asm -lhRA78K0_MAINROUTINE
Para-file :
In-file :   K0MAIN.ASM
Obj-file : K0MAIN.REL
Prn-file : K0MAIN.PRN

     Assemble list

 ALNO   STNO    ADRS    OBJECT          M   I   SOURCE STATEMENT

    1    1
    2    2                                          NAME SAMPM
    3    3                                  ; ***************************************************
    4    4                                  ; *                                                 *
    5    5                                  ; *      HEX -> ASCII Conversion Program    *
    6    6                                  ; *                                                 *
    7    7                                  ; *               main-routine                      *
      :
```

11.5 プログラム起動時の手間を省く

11.5.1 ソース・プログラムに制御命令を記述する

アセンブラ起動時に常に指定するオプションと同一の機能を持つ制御命令は，あらかじめソース・プログラム中で指定しておきます。これにより，アセンブラを起動するたびにオプションを指定する必要がなくなります。

【使用例】

```
$ PROCESSOR ( 054 )                ; 制御命令
$ XREF                             ; 制御命令

   NAME            SAMPM
; ****************************************************
; *                                                  *
; *      HEX -> ASCII Conversion Program             *
; *                                                  *
; *               main-routine                       *
; *                                                  *
; ****************************************************
   :
```

11.5.2 PM plus を使用する

RA78K0 の各プログラムのオプションは，PM plus 上でプロジェクト・ファイル（.PRJ）に自動的に保存されます。2 回目以降のビルド（MAKE）では，保存されたオプションが使用されますので，毎回オプションを指定する必要がなくなります。

11.5.3 パラメータ・ファイルやサブコマンド・ファイルを作成する

プログラム（構造化アセンブラ・プリプロセッサ，アセンブラ，リンカ，オブジェクト・コンバータ，およびリスト・コンバータ）の起動時にコマンド行に起動に必要な情報を指定しきれない場合やプログラムを起動するたびに同じオプションを指定するような場合にはパラメータ・ファイルを使用します。

また，ライブラリアンにおいては，サブコマンド・ファイルにサブコマンドを登録指定してオブジェクト・モジュールのライブラリ化が容易にできます。

【使用例 1】

パラメータ・ファイルを使用してアセンブルします。

＜パラメータ・ファイル k0main.pra の内容＞

```
; parameter file
k0main.asm -osample.rel -g
-psample.prn
```

コマンド行には，次のように入力します。

```
C>ra78k0 –fk0main.pra
```

【使用例 2】

パラメータ・ファイルを使用してアセンブルします。

＜パラメータ・ファイル k0.slb の内容＞

```
;
; library creation command
;
create k0.lib
;
add k0.lib k0main.rel &
k0sub.rel
;
exit
```

コマンド行には，次のように入力します。

```
C>lb78k0 <k0.slb
```

11.6 オブジェクト・モジュールのライブラリ化

アセンブラ，およびリンカは，1 つの出力モジュールを 1 つのファイルに作成します。したがって，モジュールの数が多い場合はファイルの数も増加します。このため，複数のモジュールを 1 つのファイルにまとめる機能を用意しました。これをモジュールのライブラリ化と呼び，ライブラリ化されたファイルをライブラリ・ファイルと呼びます。

ライブラリ・ファイルは，リンカに入力できます。したがって，モジュラ・プログラミングを行った場合，共通的なモジュールをライブラリ・ファイル化として作成しておけば，ファイル管理の面でも操作性の面においても効率がよくなります。

第 12 章　エラー・メッセージ

この章では，RA78K0（構造化アセンブラ・プリプロセッサ，アセンブラ，リンカ，オブジェクト・コンバータ，ライブラリアン，リスト・コンバータ）の出力するエラー・メッセージの原因，ユーザの処置などについて説明します。

12.1　エラー・メッセージの概要

RA78K0 のエラー・メッセージは次の 4 種類にレベル分けしています。

(1) アボート・エラー（F ××××）

処理続行が不可能なエラーが発生したため，ただちに処理を終了（中断）します。
なお，コマンド行のアボート・エラーに関しては，他のコマンド行のエラーを検出してから処理を終了します。

(2) フェータル・エラー（E ××××）

実行プログラム・エラーが発生したため，他のエラーを検出後，出力オブジェクトを生成せずに処理を終了（中断）します。
なお，フェータル・エラー時に出力オブジェクトを生成しないことを明示するため，同名のオブジェクトが存在する場合は消去して終了します。

(3) 内部エラー（C ××××）

内部エラーが発生したため，ただちに処理を終了（中断）します。

(4) ワーニング・エラー（W ××××）

ユーザが意図したものとは異なる可能性がありますが，出力オブジェクトを生成します。

備考　対話形式の実行プログラムでは，アボート・エラーの発生以外はすべて正常終了とします。

RA78K0 アセンブラ・パッケージのエラー・メッセージは次のように分類されています。

- Fn0 ××…コマンド行解析部のエラー - Fn9 ××…ファイル・システムに関するエラー - Fn1 ××…その他のアボート・エラー - Cn ×××…内部エラー - En2 ××…文の記述に関するエラー - En3 ××…式に関するエラー - En4 ××…シンボルに関するエラー - En5 ××…セグメントに関するエラー - En6 ××…制御命令，マクロに関するエラー - Wn ×××…各種ワーニング・エラー	n = 1～6 - 1…構造化アセンブラ・プリプロセッサ - 2…アセンブラ - 3…リンカ - 4…オブジェクト・コンバータ - 5…ライブラリアン - 6…リスト・コンバータ

12.2 構造化アセンブラ・プリプロセッサのエラー・メッセージ

表 12-1 構造化アセンブラ・プリプロセッサのエラー・メッセージ

エラー番号	エラー・メッセージ	
F1001	メッセージ	Missing input file
	原　因	入力ファイルを指定していません。
	ユーザの処置	入力ファイルを指定してください。
F1002	メッセージ	Too many input files
	原　因	入力ファイルが 2 つ以上指定されました。
	ユーザの処置	入力ファイルを 1 つだけ指定してください。
F1004	メッセージ	Illegal file name 'ファイル名'
	原　因	ファイル名に不当な文字があるか，または文字数が制限を越えています。
	ユーザの処置	ファイル名を正しい文字，および文字数にしてください。
F1005	メッセージ	Illegal file specification 'ファイル名'
	原　因	不当なファイルが指定されました。
	ユーザの処置	正しいファイル名を指定してください。
F1006	メッセージ	File not found 'ファイル名'
	原　因	指定された入力ファイルが存在しません。
	ユーザの処置	存在するファイル名を指定してください。
F1008	メッセージ	File specification conflicted 'ファイル名'
	原　因	入出力ファイル名が重複して指定されました。
	ユーザの処置	入出力ファイル名は異なるものを指定してください。
F1009	メッセージ	Unable to make file 'ファイル名'
	原　因	指定されたファイルにライト・プロテクトがかかっています。
	ユーザの処置	ファイルのライト・プロテクトを解除してください。
F1010	メッセージ	Directory not found 'ファイル名'
	原　因	出力ファイル名中に存在しないドライブ，またはディレクトリが含まれています。
	ユーザの処置	存在するドライブ，およびディレクトリ名を指定してください。
F1011	メッセージ	Illegal path 'オプション'
	原　因	パラメータにパスを指定するオプションで，パス名以外が指定されました。
	ユーザの処置	正しいパス名を指定してください。
F1012	メッセージ	Missing parameter 'オプション'
	原　因	必要なパラメータが指定されていません。
	ユーザの処置	パラメータを指定してください。

表 12-1 構造化アセンブラ・プリプロセッサのエラー・メッセージ

エラー番号	エラー・メッセージ	
F1013	メッセージ	Parameter not needed ' オプション '
	原　因	不要なパラメータが指定されています。
	ユーザの処置	不要なパラメータを削除してください。
F1014	メッセージ	Out of range 'オプション '
	原　因	指定数値が範囲外です。
	ユーザの処置	正しい数値を指定してください。
F1015	メッセージ	Parameter is too long 'オプション '
	原　因	パラメータの文字数が制限を越えています。
	ユーザの処置	パラメータの文字数を制限内にしてください。
F1016	メッセージ	Illegal parameter 'オプション '
	原　因	パラメータの文法が誤っています。
	ユーザの処置	正しいパラメータを指定してください。
F1017	メッセージ	Too many parameters 'オプション '
	原　因	パラメータの総数が制限を越えています。
	ユーザの処置	パラメータの総数を制限内にしてください。
F1018	メッセージ	Option is not recognized 'オプション '
	原　因	オプション名が誤っています。
	ユーザの処置	正しいオプション名を指定してください。
F1019	メッセージ	Parameter file nested
	原　因	パラメータ・ファイル中に -F オプションが指定されました。
	ユーザの処置	パラメータ・ファイル中に -F オプションを指定しないでください。
F1020	メッセージ	Parameter file read error 'ファイル名 '
	原　因	パラメータ・ファイルの読み込みができません。
	ユーザの処置	正しいパラメータ・ファイルを指定してください。
F1021	メッセージ	Memory allocation failed
	原　因	メモリが足りません。
	ユーザの処置	必要なメモリを確保してください。
F1100	メッセージ	Can't set Control-C
	原　因	構造化アセンブラ・プリプロセッサ実行中止のためのコントロール C の設定ができません。
	ユーザの処置	構造化アセンブラ・プリプロセッサを，もう一度実行してください。
F1101	メッセージ	Open/read/write/close error on 'ファイル名 '
	原　因	ファイルの入出力にエラーがあり，オープン，リード，ライト，クローズが正常にできません。
	ユーザの処置	正しいファイル名を指定してください。

表 12-1 構造化アセンブラ・プリプロセッサのエラー・メッセージ

エラー番号	エラー・メッセージ	
F1102	メッセージ	Can't find 'ファイル名'
	原　因	インクルード・ファイルがないか，またはインクルード・ファイル名と入力ファイル名，出力ファイル名が重複しています。
	ユーザの処置	正しいパス，ディレクトリ，ファイル名を指定してください。
F1103	メッセージ	Illegal include file 'ファイル名'
	原　因	インクルード・ファイルに不当なものが指定されました。
	ユーザの処置	正しいファイルを指定してください。
F1104	メッセージ	Illegal(-sc) character
	原　因	-SC オプションで，シンボルとして使用できない文字が指定されました。
	ユーザの処置	正しい文字を指定してください。
F1105	メッセージ	Can't define the reserved symbol
	原　因	-D オプションで予約語を指定しました。
	ユーザの処置	-D オプションで予約語を指定しないでください。
F1106	メッセージ	Duplicate PROCESSOR control
	原　因	ソース・ファイル中で PROCESSOR 制御命令を重複指定しました。 -C オプションと異なる品種が指定されました。
	ユーザの処置	PROCESSOR 制御命令の指定は 1 回にしてください。 品種名を修正してください。
F1107	メッセージ	No processor specified
	原　因	デバイス種別の指定がありません。
	ユーザの処置	デバイス種別を指定してください。
F1108	メッセージ	Illegal processor type specified
	原　因	ソース・ファイル中の PROCESSOR 制御命令で，デバイス種別の指定が誤っています。
	ユーザの処置	正しいデバイス種別を指定してください。
F1109	メッセージ	Illegal processor type specified -C
	原　因	-C オプションで，デバイス種別の指定が誤っています。
	ユーザの処置	正しいデバイス種別を指定してください。
F1110	メッセージ	Can't use this control outside module header
	原　因	ソース・モジュール・ヘッダに記述すべき制御命令が，通常のソース行に記述されています。
	ユーザの処置	制御命令をソース・モジュール・ヘッダに記述してください。
F1111	メッセージ	Syntax error in module header
	原　因	ソース・モジュール・ヘッダに記述されている制御命令の記述形式に誤りがあります。
	ユーザの処置	正しい形式で制御命令を記述してください。

表 12-1 構造化アセンブラ・プリプロセッサのエラー・メッセージ

エラー番号	エラー・メッセージ	
C1112	メッセージ	Structured assembler preprocessor internal error
	原　因	構造化アセンブラ・プリプロセッサ自身に内部エラーが発生しました。
	ユーザの処置	特約店，または当社までご連絡ください。
E1201	メッセージ	Illegal #ELSE/#ENDIF
	原　因	#ELSE , #ENDIF 文の記述位置に誤りがあります。
	ユーザの処置	#ELSE , #ENDIF 文を正しい位置に記述してください。
E1202	メッセージ	Illegal #ENDCALLT
	原　因	#ENDCALLT 文の記述位置に誤りがあります。
	ユーザの処置	#ENDCALLT 文を正しい位置に記述してください。
E1203	メッセージ	Missing #ENDIF
	原　因	#ENDIF 文がありません。
	ユーザの処置	#ENDIF 文を正しい位置に記述してください。
E1204	メッセージ	Missing #ENDCALLT
	原　因	#ENDCALLT 文がありません。
	ユーザの処置	#ENDCALLT 文を正しい位置に記述してください。
E1205	メッセージ	Too many #DEFCALLT definition
	原　因	callt 命令変換パターンの登録数が制限を越えています。
	ユーザの処置	#DEFCALLT の登録数を減らしてください。
E1206	メッセージ	Too many CALL instructions
	原　因	#DEFCALLT 〜 #ENDCALLT で定義する命令が多すぎます。
	ユーザの処置	#DEFCALLT 〜 #ENDCALLT で定義する命令を 1 つにしてください。
E1207	メッセージ	Duplicate definition
	原　因	同一変換パターンの再定義をしました。
	ユーザの処置	#DEFCALLT の登録を修正してください。
E1208	メッセージ	Symbol table overflow
	原　因	シンボル数が制限を越えています。
	ユーザの処置	シンボル数を減らしてください。
E1209	メッセージ	Syntax error
	原　因	記述した文の構文に誤りがあります。
	ユーザの処置	正しい構文を記述してください。
E1210	メッセージ	Nest level error
	原　因	ネスティングに誤りがあります（オーバフロー，ネストの対など）。
	ユーザの処置	正しい制御文を記述してください。

表 12-1 構造化アセンブラ・プリプロセッサのエラー・メッセージ

エラー番号	エラー・メッセージ	
E1211	メッセージ	Too many characters in a line
	原　因	1 行の長さが制限を越えています。
	ユーザの処置	1 行の長さを 2046 文字以内にしてください。
E1212	メッセージ	Too many include files
	原　因	インクルード・ファイルの中にインクルード疑似命令があります。
	ユーザの処置	インクルード・ファイル中にインクルード疑似命令を指定しないでください。
E1214	メッセージ	Illegal BREAK
	原　因	BREAK 文の記述位置に誤りがあります。
	ユーザの処置	BREAK 文を正しい位置に記述してください。
E1215	メッセージ	Illegal CONTINUE
	原　因	CONTINUE 文の記述位置に誤りがあります。
	ユーザの処置	CONTINUE 文を正しい位置に記述してください。
E1216	メッセージ	Illegal CASE/DEFAULT/ENDS
	原　因	CASE/DEFAULT/ENDS 文の記述位置に誤りがあります。
	ユーザの処置	CASE/DEFAULT/ENDS 文を正しい位置に記述してください。
E1217	メッセージ	Illegal ELSEIF/ELSE/ENDIF
	原　因	ELSEIF/ELSE/ENDIF 文の記述位置に誤りがあります。
	ユーザの処置	ELSEIF/ELSE/ENDIF 文を正しい位置に記述してください。
E1218	メッセージ	Illegal NEXT
	原　因	NEXT 文の記述位置に誤りがあります。
	ユーザの処置	NEXT 文を正しい位置に記述してください。
E1219	メッセージ	Illegal ENDW
	原　因	ENDW 文の記述位置に誤りがあります。
	ユーザの処置	ENDW 文を正しい位置に記述してください。
E1220	メッセージ	Illegal UNTIL/UNTIL_BIT
	原　因	UNTIL , UNTIL_BIT 文の記述位置に誤りがあります。
	ユーザの処置	UNTIL , UNTIL_BIT 文を正しい位置に記述してください。
E1221	メッセージ	Missing ENDIF
	原　因	ENDIF 文がありません。
	ユーザの処置	ENDIF 文を正しい位置に記述してください。
E1222	メッセージ	Missing ENDS
	原　因	ENDS 文がありません。
	ユーザの処置	ENDS 文を正しい位置に記述してください。

表 12-1 構造化アセンブラ・プリプロセッサのエラー・メッセージ

エラー番号	エラー・メッセージ	
E1223	メッセージ	Missing ENDW
	原　因	ENDW 文がありません。
	ユーザの処置	ENDW 文を正しい位置に記述してください。
E1224	メッセージ	Missing NEXT
	原　因	NEXT 文がありません。
	ユーザの処置	NEXT 文を正しい位置に記述してください。
E1225	メッセージ	Missing UNTIL/UNTIL_BIT
	原　因	UNTIL，または UNTIL_BIT 文がありません。
	ユーザの処置	UNTIL，または UNTIL_BIT 文を正しい位置に記述してください。
E1226	メッセージ	Illegal character in a line
	原　因	ソース・ライン中に不正な文字の記述があります。
	ユーザの処置	ソース・ライン中の不正な文字の記述を削除してください。
E1227	メッセージ	Illegal operand in a line
	原　因	代入式，比較条件式のデータ・サイズに誤りがあります。
	ユーザの処置	正しいデータ・サイズを指定してください。
E1228	メッセージ	Illegal SFR access in operand
	原　因	代入式にアクセスできない sfr シンボルを記述しました。
	ユーザの処置	sfr シンボルのアクセス状態を確認して，正しい sfr シンボルを記述してください。
E1229	メッセージ	This symbol is reserved“ シンボル名 ”
	原　因	使用したシンボルが予約語になっています。
	ユーザの処置	シンボル名を変更してください。
E1230	メッセージ	Out source line overflow
	原　因	ソース・ラインの文字数が制限値を超えました。
	ユーザの処置	ソース・ライン中の不要な記述を削除してください。
W1301	メッセージ	Symbol redefinition
	原　因	#define 文によりシンボルの再定義をしています。
	プログラムの処置	最後に定義されたシンボルを有効とします。
	ユーザの処置	最初に定義したシンボルを有効にしたい場合は構文を修正してください。
W1302	メッセージ	Duplicate PROCESSOR option and control
	原　因	-C オプションと $PROCESSOR 制御命令で異なる品種を指定しています。
	プログラムの処置	-C オプションで指定された品種を有効とし，$PROCESSOR 制御命令は無視します。
	ユーザの処置	-C オプションで指定した品種で正しかったことを確認してください。

12.3 アセンブラのエラー・メッセージ

表 12-2 アセンブラのエラー・メッセージ

エラー番号	エラー・メッセージ	
F2001	メッセージ	Missing input file
	原　因	入力ファイルを指定していません。
	ユーザの処置	入力ファイルを指定してください。
F2002	メッセージ	Too many input files
	原　因	入力ファイルが 2 つ以上指定されました。
	ユーザの処置	入力ファイルを 1 つだけ指定してください。
F2004	メッセージ	Illegal file name ‘ ファイル名 ’
	原　因	ファイル名に不当な文字があるか，または文字数が制限を越えています。
	ユーザの処置	ファイル名を正しい文字，および文字数にしてください。
F2005	メッセージ	Illegal file specification ‘ ファイル名 ’
	原　因	不当なファイルが指定されました。
	ユーザの処置	正しいファイル名を指定してください。
F2006	メッセージ	File not found ‘ ファイル名 ’
	原　因	指定された入力ファイルが存在しません。
	ユーザの処置	存在するファイル名を指定してください。
F2008	メッセージ	File specification conflicted ‘ ファイル名 ’
	原　因	入出力ファイル名が重複して指定されました。
	ユーザの処置	入出力ファイル名は異なるものを指定してください。
F2009	メッセージ	Unable to make file ‘ ファイル名 ’
	原　因	指定されたファイルにライト・プロテクトがかかっています。
	ユーザの処置	ファイルのライト・プロテクトを解除してください。
F2010	メッセージ	Directory not found ‘ ファイル名 ’
	原　因	出力ファイル名中に存在しないドライブ，またはディレクトリが含まれています。
	ユーザの処置	存在するドライブ，およびディレクトリ名を指定してください。
F2011	メッセージ	Illegal path ‘ オプション ’
	原　因	パラメータにパスを指定するオプションで，パス名以外が指定されました。
	ユーザの処置	正しいパス名を指定してください。
F2012	メッセージ	Missing parameter ‘ オプション ’
	原　因	必要なパラメータが指定されていません。
	ユーザの処置	パラメータを指定してください。

表 12-2 アセンブラのエラー・メッセージ

エラー番号	エラー・メッセージ	
F2013	メッセージ	Parameter not needed ‘ オプション ’
	原　因	不要なパラメータが指定されています。
	ユーザの処置	不要なパラメータを削除してください。
F2014	メッセージ	Out of range ‘ オプション ’
	原　因	指定数値が範囲外です。
	ユーザの処置	正しい数値を指定してください。
F2015	メッセージ	Parameter is too long ‘ オプション ’
	原　因	パラメータの文字数が制限を越えています。
	ユーザの処置	パラメータの文字数を制限内にしてください。
F2016	メッセージ	Illegal parameter ‘ オプション ’
	原　因	パラメータの文法が誤っています。
	ユーザの処置	正しいパラメータを指定してください。
F2017	メッセージ	Too many parameters ‘ オプション ’
	原　因	パラメータの総数が制限を越えています。
	ユーザの処置	パラメータの総数を制限内にしてください。
F2018	メッセージ	Option is not recognized ‘ オプション ’
	原　因	オプション名が誤っています。
	ユーザの処置	正しいオプション名を指定してください。
F2019	メッセージ	Parameter file nested
	原　因	パラメータ・ファイル中に -F オプションが指定されました。
	ユーザの処置	パラメータ・ファイル中に -F オプションを指定しないでください。
F2020	メッセージ	Parameter file read error ‘ ファイル名 ’
	原　因	パラメータ・ファイルの読み込みができません。
	ユーザの処置	正しいパラメータ・ファイルを指定してください。
F2021	メッセージ	Memory allocation failed
	原　因	メモリが足りません。
	ユーザの処置	必要なメモリを確保してください。
F2101	メッセージ	Source file size 0　‘ファイル名 ’
	原　因	サイズが 0 バイトのソース・ファイルを入力しました。
F2102	メッセージ	Illegal processor type specified
	原　因	対象デバイスの指定がまちがっています。
F2103	メッセージ	Syntax error in module header
	原　因	ソース・モジュール・ヘッダに記述可能な制御命令の記述形式がまちがっています。

表 12-2 アセンブラのエラー・メッセージ

エラー番号	エラー・メッセージ	
F2104	メッセージ	Can't use this control outside module header
	原　因	ソース・モジュール・ヘッダに記述する制御命令が，通常のソースに記述されています。
F2105	メッセージ	Duplicate PROCESSOR control
	原　因	ソース・モジュール・ヘッダの中で PROCESSOR 制御命令が重複して記述されています。
F2106	メッセージ	Illegal source file name for module name
	原　因	ソース・ファイル名のプライマリ・ネームがシンボルの構成文字に反しているためモジュール名が作成できません。
F2107	メッセージ	Default segment ? CSEG is already used
	原　因	セグメント定義省略時にディフォールト・セグメントを定義しようとしました。
F2108	メッセージ	Symbol table overflow ' シンボル名 '
	原　因	定義可能なシンボル数の制限を越えています。
F2109	メッセージ	Too many DS
	原　因	DS 疑似命令が多くあるために，セグメント内のオブジェクト・コードの間隔が空きすぎて，オブジェクト・ファイルに情報を出力できません。
F2110	メッセージ	String table overflow
	原　因	ストリング・テーブルの制限を越えました。
	ユーザの処置	9 文字以上のシンボル数を減らしてください。
F2111	メッセージ	Object code more than 128bytes
	原　因	オブジェクト・コードがソース・ステートメント 1 行につき，128 バイトを越えました。
F2112	メッセージ	No processor specified
	原　因	対象デバイスがコマンド行にも，ソース・モジュール・ファイルにも指定されていません。
F2114	メッセージ	Local symbol name of asm statement must begin with ‘?L’ in C source.
	原　因	C ソースの #asm 中に ‘?L’ で始まらないローカル・シンボルが記述されています。
E2201	メッセージ	Syntax error
	原　因	文の記述形式がまちがっています。
E2202	メッセージ	Illegal operand
	原　因	オペランドの記述が不正です。
E2203	メッセージ	Illegal register
	原　因	記述できないレジスタが指定されました。
E2204	メッセージ	Illegal character
	原　因	ソース・モジュール中に不正な文字の記述があります。

表 12-2 アセンブラのエラー・メッセージ

エラー番号	エラー・メッセージ	
E2205	メッセージ	Unexpected LF in string
	原　因	文字列が閉じる前に改行コードが現れました。
E2206	メッセージ	Unexpected EOF in string
	原　因	文字列が閉じる前にファイルの終わりになりました。
E2207	メッセージ	Unexpected null code in string
	原　因	文字列中にヌル・コード（00H）が記述されました。
E2209	メッセージ	Too many line number
	原　因	1 ファイルに記述可能な行数を超えています。
E2301	メッセージ	Too complex expression
	原　因	式が複雑すぎます。
E2302	メッセージ	Absolute expression expected
	原　因	リロケータブルな式が記述されています。
E2303	メッセージ	Illegal expression
	原　因	式の記述形式に誤りがあります。
E2304	メッセージ	Illegal symbol in expression 'ファイル名'
	原　因	式の中に使用できないシンボルが記述されています。
E2305	メッセージ	Too long string as constant
	原　因	文字定義の長さの制限（4 文字）を越えています。
E2306	メッセージ	Illegal number
	原　因	数値の記述に誤りがあります。
E2307	メッセージ	Division by zero
	原　因	0 で除算をしています。
E2308	メッセージ	Too large integer
	原　因	定数の値が 16 ビットを越えています。
E2309	メッセージ	Illegal bit value
	原　因	ビット値の記述に誤りがあります。
E2310	メッセージ	Bit value out of range
	原　因	ビット値が 0-7 の範囲を越えました。
E2311	メッセージ	Operand out of range（n）
	原　因	指定された値が，n（0-7）の範囲を越えました。
E2312	メッセージ	Operand out of range（byte）
	原　因	オペランドの値が範囲（00H-FFH）を越えたか，あるいはオペランド中の byte の値が範囲（-128 〜 +127）を越えました。

表 12-2 アセンブラのエラー・メッセージ

エラー番号	エラー・メッセージ	
E2313	メッセージ	Operand out of range（addr5）
	原　因	addr5 として記述可能な範囲（40H-7EH）を越えました。
E2314	メッセージ	Operand out of range（addr11）
	原　因	addr11 として記述可能な範囲（800H-FFFH）を越えました。
E2315	メッセージ	Operand out of range（saddr）
	原　因	saddr として記述可能な範囲（0FE20H-0FF1FH）を越えました。
E2316	メッセージ	Operand out of range（addr16）
	原　因	addr16 として記述可能な範囲（対象デバイスによって異なる）を越えました。
E2317	メッセージ	Even expression expected
	原　因	ワード・アクセスに奇数アドレスを記述しています。
E2318	メッセージ	Operand out of range（sfr）
	原　因	SFR/SFRP 疑似命令のオペランドが記述可能な範囲を越えているか，あるいは SFRP 疑似命令のオペランドとして奇数の値が記述されています。
E2326	メッセージ	Illegal SFR access in operand
	原　因	オペランドのアクセスできない SFR シンボルを記述しています。
E2327	メッセージ	Illegal bank access in operand
	原　因	オペランドにアクセスできないシンボルを記述しています。
E2401	メッセージ	Illegal symbol for PUBLIC　'シンボル名'
	原　因	このシンボルは PUBLIC 宣言できません。
E2402	メッセージ	Illegal symbol for EXTRN/EXTBIT　'シンボル名'
	原　因	このシンボルは EXTRN/EXTBIT 宣言できません。
E2403	メッセージ	Can't define PUBLIC symbol　'シンボル名'
	原　因	すでに PUBLIC 宣言されたシンボルに，PUBLIC 宣言できないシンボル定義しました。
	ユーザの処置	saddr.bit 以外のビット項を定義したシンボルは PUBLIC 宣言できないので PUBLIC 宣言を取り消すか，EQU の定義を変更してください。
E2404	メッセージ	Public symbol is undefined　'シンボル名'
	原　因	PUBLIC 宣言されたシンボルが定義されていません。
E2405	メッセージ	Illegal bit symbol
	原　因	機械語命令のオペランドのビット・シンボルに，前方参照のシンボルあるいはビット・シンボルとして不当なシンボルを使用しています。
	ユーザの処置	ビット・シンボルには，後方参照あるいは EXTBIT 宣言したシンボルを記述してください。

表 12-2 アセンブラのエラー・メッセージ

エラー番号	エラー・メッセージ	
E2406	メッセージ	Can't refer forward bit symbol 'シンボル名'
	原　因	ビット・シンボルを前方参照しているか，または式の中にビット・シンボルを記述しています。
E2407	メッセージ	Undefined symbol reference 'シンボル名'
	原　因	未定義シンボルを使用しています。
E2408	メッセージ	Multiple symbol definition 'シンボル名'
	原　因	シンボル名が重複して定義されています。
E2409	メッセージ	Too many symbols in operand
	原　因	1 行以内に記述可能なオペランドのシンボル個数が制限を越えました。
E2410	メッセージ	Phase error
	原　因	アセンブル中にシンボルの値が変化しました（たとえば，BR 疑似命令の最適化処理によって変化したレーベルをオペランドの中に用いて定義した EQU シンボルなど）。
E2411	メッセージ	This symbol is reserved 'シンボル名'
	原　因	指定したシンボルは予約語になっています。
E2502	メッセージ	Illegal segment name
	原　因	セグメント名として不正なシンボルが記述されています。
E2503	メッセージ	Different segment type 'セグメント名'
	原　因	同名セグメント定義において，セグメントのタイプが異なっています。
E2504	メッセージ	Too many segments
	原　因	定義できるセグメントの制限（256 個）を越えています。
E2505	メッセージ	Current segment is not exist
	原　因	ENDS 疑似命令が，セグメントが作られる前，あるいは一度セグメントが終了したあとに，次のセグメントが作られる前に記述されました。
E2506	メッセージ	Can't describe DB, DW, DS, ORG, label in BSEG
	原　因	DB , DW , DS , ORG 疑似命令をビット・セグメント内で記述しています。
E2507	メッセージ	Can't describe opcodes outside CSEG
	原　因	機械語命令，BR 疑似命令をコード・セグメント以外で記述しています。
E2508	メッセージ	Can't describe DBIT outside BSEG
	原　因	DBIT 疑似命令をビット・セグメント以外で記述しています。
E2509	メッセージ	Illegal address specified
	原　因	アブソリュート・セグメントとして配置したアドレスが，そのセグメントに対応する範囲を越えています。
E2510	メッセージ	Location counter overflow
	原　因	ロケーション・カウンタがセグメントに対応した範囲を越えました。

表 12-2 アセンブラのエラー・メッセージ

エラー番号	エラー・メッセージ	
E2511	メッセージ	Segment name expected
	原　因	再配置属性が AT のセグメント定義疑似命令でセグメント名が指定されていません。
E2512	メッセージ	Segment size is odd numbers　'セグメント名'
	原　因	再配置属性 callt0 のセグメントが奇数サイズで記述されています。
E2515	メッセージ	Security ID is not supported for this device
	原　因	セキュリティ ID は，指定されているデバイスでは使用できません。
E2517	メッセージ	Illegal bank number
	原　因	不正なバンク指定が記述されています。
E2601	メッセージ	Nesting over of include
	原　因	インクルード・ファイルのネスティングできる制限（8 レベル）を越えています。
E2602	メッセージ	Must be specified switches
	原　因	スイッチ名が指定されていません。
E2603	メッセージ	Too many switches described
	原　因	スイッチ名の記述が制限（1 モジュール内で 1000 個以内）を越えています。
E2604	メッセージ	Nesting over of IF-classes
	原　因	IF/_IF 節のネスティングの制限（8 レベル）を越えています。
E2605	メッセージ	Needless ELSE statement exists
	原　因	必要のないところに ELSE 文が存在しています。
E2606	メッセージ	Needless ENDIF statement exists
	原　因	必要のないところに ENDIF 文が存在しています。
E2608	メッセージ	Missing ENDIF
	原　因	IF 文，または _IF 文と ENDIF 文の対応がとれていません。
E2609	メッセージ	Illegal ELSEIF statement
	原　因	ELSE 文のあとに ELSEIF 文，または _ELSEIF 文が記述されています。
E2610	メッセージ	Multiple symbol definition（MACRO）'シンボル名'
	原　因	マクロ名として定義しようとしたシンボルが，すでに定義されています。
E2611	メッセージ	Illegal syntax of parameter
	原　因	マクロの仮パラメータの記述に誤りがあります。
E2612	メッセージ	Too many parameter
	原　因	1 マクロ定義の仮パラメータの個数が制限（16 個）を越えています。
E2613	メッセージ	Same name parameter described　'シンボル名'
	原　因	1 マクロ定義の仮パラメータとして同名のシンボルが指定されました。

表 12-2 アセンブラのエラー・メッセージ

エラー番号	エラー・メッセージ	
E2614	メッセージ	Can’t nest macro definition
	原　因	マクロ定義の中でマクロ定義を行っています。
E2615	メッセージ	Illegal syntax of local symbol
	原　因	LOCAL 疑似命令のオペランド記述に誤りがあります。
E2616	メッセージ	Too many local symbols
	原　因	1 つのマクロ・ボディ内で記述できるローカル・シンボル数の制限（64 個）を越えています。
E2617	メッセージ	Missing ENDM
	原　因	マクロ定義疑似命令に対応する ENDM 文がありません。
E2618	メッセージ	Illegal syntax of ENDM
	原　因	ENDM 文の記述に誤りがあります。
E2619	メッセージ	Illegally defined macro
	原　因	参照したマクロは定義時に誤りがあります。
E2620	メッセージ	Illegal syntax of actual parameter
	原　因	マクロの実パラメータの記述に誤りがあります。
E2621	メッセージ	Nesting over of macro reference
	原　因	マクロ参照において，ネスティングできる制限（8 レベル）を越えています。
E2622	メッセージ	Illegal syntax of EXITM
	原　因	EXITM 文の記述に誤りがあります。
E2623	メッセージ	Illegal operand of REPT
	原　因	REPT 疑似命令のオペランドに許されていない式が記述されています。
E2624	メッセージ	More than ??RAFFFF
	原　因	マクロ展開の際にローカル・シンボルの置き換えが，65535 個を越えました。
E2625	メッセージ	Unexpected ENDM
	原　因	余分な ENDM 文が現れました。
E2626	メッセージ	Can’t describe LOCAL macro definition
	原　因	マクロ・ボディ以外の通常ソース・ステートメント中に LOCAL 疑似命令が記述されました。
E2627	メッセージ	More than two segments in this include/macro
	原　因	インクルード・ファイル，マクロ・ボディ，rept-endm ブロック，irp-endm ブロック中に，2 つ以上のセグメントが存在しています。
W2701	メッセージ	Too long source line
	原　因	ソース・ステートメント 1 行が 2048 文字を越えています。
	プログラムの処理	2049 文字以降を無視します。

表 12-2 アセンブラのエラー・メッセージ

エラー番号	エラー・メッセージ	
W2702	メッセージ	Duplicate PROCESSOR option and control
	原　因	コマンド・ライン上の対象デバイスと指定オプション（-C）と，ソース・ヘッダ中の PROCESSOR 制御命令が両方とも指定されています。
	プログラムの処理	コマンド・ライン上の対象デバイスと指定オプションを有効とし，ソース・ヘッダ中の PROCESSOR 疑似命令を無視します。
W2703	メッセージ	Multiple defined module name
	原　因	NAME 疑似命令が二度以上定義されています。
	プログラムの処理	その NAME 疑似命令を無効として，すでに定義されたモジュール名を有効とします。
W2704	メッセージ	Already declared EXTRN symbol　'シンボル名'
	原　因	このシンボルはすでに EXTRN 宣言されています。
	ユーザの処置	1 つのシンボルの EXTRN 宣言は，1 モジュールにつき 1 回にしてください。
W2705	メッセージ	Already declared EXTBIT symbol　'シンボル名'
	原　因	このシンボルはすでに EXTBIT 宣言されています。
	ユーザの処置	1 つのシンボルの EXTBIT 宣言は，1 モジュールにつき 1 回にしてください。
W2706	メッセージ	Missing END statement
	原　因	ソース・ファイルの最後に END 文が記述されていません。
	プログラムの処理	ソース・ファイルの最後に END 文があったものとみなします。
W2707	メッセージ	Illegal statement after END directive
	原　因	END 文のあとにコメント，空白，タブ，改行コード以外のものが記述されました。
	プログラムの処理	END 文のあとを無視します。
W2708	メッセージ	Already declared LOCAL symbol　'シンボル名'
	原　因	このシンボルは，すでに LOCAL 宣言されています。
	ユーザの処置	1 つのシンボルの LOCAL 宣言は，1 マクロにつき 1 回にしてください。
W2709	メッセージ	Few count of actual parameter
	原　因	実パラメータの個数が仮パラメータの個数よりも少なく設定されています。
	プログラムの処理	足りない個数分，仮パラメータはヌル・ストリングが与えられます。
W2710	メッセージ	Over count of actual parameter
	原　因	実パラメータの個数が仮パラメータの個数よりも多く設定されています。
	プログラムの処理	超過分の実パラメータは，無視されます。

表 12-2 アセンブラのエラー・メッセージ

エラー番号	エラー・メッセージ	
W2711	メッセージ	Too many errors to report
	原　因	この行に対するエラーが多すぎます（エラーが 6 個以上）。
	プログラムの処理	6 個目以降のエラー・メッセージは出力せずに処理を続けます。
W2712	メッセージ	Insufficient cross-reference work area
	原　因	クロスレファレンス・リストの出力処理を行うためのメモリが不足しています。
	プログラムの処理	クロスレファレンス・リストの出力せずに，処理を続けます。
W2717	メッセージ	Normal, callt and callf functions must be described together respectively.
	原　因	「通常関数」，「callt 関数」，「callf 関数」がそれぞれまとめて記述されていないため，ディバグ情報が不正になる場合があります。
	プログラムの処理	通常関数と callt 関数もまとめて記述してください。
E2801	メッセージ	Illegal debug information
	原　因	ソース・ファイル中のディバグ情報が不正です。
	ユーザの処置	コンパイル，または構造化アセンブルを，もう一度実行してください。
F2901	メッセージ	Can't open source file　'ファイル名'
	原　因	ソース・ファイルがオープンできません。
F2902	メッセージ	Can't open parameter file　'ファイル名'
	原　因	パラメータ・ファイルがオープンできません。
F2903	メッセージ	Can't open include file　'ファイル名'
	原　因	インクルード・ファイルがオープンできません。
F2904	メッセージ	Illegal include file　'ファイル名'
	原　因	インクルード・ファイル名として，ドライブ名のみ，パス名のみ，デバイス型ファイル名のいずれかが指定されました。
F2905	メッセージ	Can't open overlay file　'ファイル名'
	原　因	オーバレイ・ファイルがオープンできません。
	ユーザの処置	オーバレイ・ファイルがアセンブラの実行形式と同じディレクトリにあるかどうかを調べてください。
F2906	メッセージ	Illegal overlay file　'ファイル名'
	原　因	オーバレイ・ファイルの内容が不正です。
F2907	メッセージ	Can't open object file　'ファイル名'
	原　因	オブジェクト・ファイルがオープンできません。
	ユーザの処置	ディレクトリに空き領域のあるディスクを使用してください。

表 12-2 アセンブラのエラー・メッセージ

エラー番号	エラー・メッセージ	
F2908	メッセージ	Can't open print file 'ファイル名'
	原　因	アセンブル・リスト・ファイルがオープンできません。
	ユーザの処置	ディレクトリに空き領域のあるディスクを使用してください。
F2909	メッセージ	Can't open error list file 'ファイル名'
	原　因	エラー・リスト・ファイルがオープンできません。
	ユーザの処置	ディレクトリに空き領域のあるディスクを使用してください。
F2910	メッセージ	Can't open temporary file 'ファイル名'
	原　因	テンポラリ・ファイルがオープンできません。
	ユーザの処置	ディレクトリに空き領域のあるディスクを使用してください。
F2913	メッセージ	Can't read source file 'ファイル名'
	原　因	ソース・ファイルにファイル I/O エラーが発生しました。
F2914	メッセージ	Can't read parameter file 'ファイル名'
	原　因	パラメータ・ファイルにファイル I/O エラーが発生しました。
F2915	メッセージ	Can't read include file 'ファイル名'
	原　因	インクルード・ファイルにファイル I/O エラーが発生しました。
F2916	メッセージ	Can't read overlay file 'ファイル名'
	原　因	オーバレイ・ファイルにファイル I/O エラーが発生しました。
F2917	メッセージ	Can't write object file 'ファイル名'
	原　因	オブジェクト・ファイルにファイル I/O エラーが発生しました。
	ユーザの処置	オブジェクト・ファイルを他のディレクトリに出力するか，指定したディスクに空き領域を作ってください。
F2918	メッセージ	Can't write print file 'ファイル名'
	原　因	アセンブル・リスト・ファイルにファイル I/O エラーが発生しました。
	ユーザの処置	アセンブル・リスト・ファイルを他のディレクトリに出力するか，指定したディスクに空きを領域を作ってください。
F2919	メッセージ	Can't write error list file 'ファイル名'
	原　因	エラー・リスト・ファイルにファイル I/O エラーが発生しました。
	ユーザの処置	エラー・リスト・ファイルを他のディレクトリに出力するか，指定したディスクに空き領域を作ってください。
F2920	メッセージ	Can't read/write temporary file 'ファイル名'
	原　因	テンポラリ・ファイルにファイル I/O エラーが発生しました。
	ユーザの処置	テンポラリ・ファイルを他のディレクトリに出力するか，指定したディスクに空き領域を作ってください。

表 12-2 アセンブラのエラー・メッセージ

エラー番号	エラー・メッセージ	
C2921	メッセージ	Assembler internal error
	原　因	アセンブラ自身に内部エラーが発生しました。
	ユーザの処置	もう一度アセンブルを実行してください。 エラーが解決できない場合には，特約店または当社までご連絡ください。
F2922	メッセージ	Insufficient memory in hostmachine
	原　因	システムにアセンブラを実行するための十分なメモリがありません。
F2923	メッセージ	Insufficient memory for macro in hostmachine
	原　因	マクロ処理の途中で内部メモリが不足しました。
	ユーザの処置	マクロ定義を少なくしてください。

12.4 リンカのエラー・メッセージ

表 12-3 リンカのエラー・メッセージ

エラー番号	エラー・メッセージ	
F3001	メッセージ	Missing input file
	原　因	入力ファイルを指定していません。
	ユーザの処置	入力ファイルを指定してください。
F3004	メッセージ	Illegal file name ‘ ファイル名 ’
	原　因	ファイル名に不当な文字があるか，または文字数が制限を越えています。
	ユーザの処置	ファイル名を正しい文字，および文字数にしてください。
F3005	メッセージ	Illegal file specification ‘ ファイル名 ’
	原　因	不当なファイルが指定されました。
	ユーザの処置	正しいファイル名を指定してください。
F3006	メッセージ	File not found ‘ ファイル名 ’
	原　因	指定された入力ファイルが存在しません。
	ユーザの処置	存在するファイル名を指定してください。
F3007	メッセージ	Input file specification overlapped ' ファイル名 '
	原　因	入力ファイル名が重複して指定されました。
	ユーザの処置	正しいファイル名を指定してください。
F3008	メッセージ	File specification conflicted ‘ ファイル名 ’
	原　因	入出力ファイル名が重複して指定されました。
	ユーザの処置	入出力ファイル名は異なるものを指定してください。
F3009	メッセージ	Unable to make file ‘ ファイル名 ’
	原　因	指定されたファイルにライト・プロテクトがかかっています。
	ユーザの処置	ファイルのライト・プロテクトを解除してください。
F3010	メッセージ	Directory not found ‘ ファイル名 ’
	原　因	出力ファイル名中に存在しないドライブ，またはディレクトリが含まれています。
	ユーザの処置	存在するドライブ，およびディレクトリ名を指定してください。
F3011	メッセージ	Illegal path ‘ オプション ’
	原　因	パラメータにパスを指定するオプションで，パス名以外が指定されました。
	ユーザの処置	正しいパス名を指定してください。
F3012	メッセージ	Missing parameter ‘ オプション ’
	原　因	必要なパラメータが指定されていません。
	ユーザの処置	パラメータを指定してください。

表 12-3 リンカのエラー・メッセージ

エラー番号	エラー・メッセージ	
F3013	メッセージ	Parameter not needed ‘ オプション ’
	原　因	不要なパラメータが指定されています。
	ユーザの処置	不要なパラメータを削除してください。
F3014	メッセージ	Out of range ‘ オプション ’
	原　因	指定数値が範囲外です。
	ユーザの処置	正しい数値を指定してください。
F3015	メッセージ	Parameter is too long ‘ オプション ’’
	原　因	パラメータの文字数が制限を越えています。
	ユーザの処置	パラメータの文字数を制限内にしてください。
F3016	メッセージ	Illegal parameter ‘ オプション ’
	原　因	パラメータの文法が誤っています。
	ユーザの処置	正しいパラメータを指定してください。
F3017	メッセージ	Too many parameters ‘ オプション ’
	原　因	パラメータの総数が制限を越えています。
	ユーザの処置	パラメータの総数を制限内にしてください。
F3018	メッセージ	Option is not recognized ‘ オプション ’
	原　因	オプション名が誤っています。
	ユーザの処置	正しいオプション名を指定してください。
F3019	メッセージ	Parameter file nested
	原　因	パラメータ・ファイル中に -F オプションが指定されました。
	ユーザの処置	パラメータ・ファイル中に -F オプションを指定しないでください。
F3020	メッセージ	Parameter file read error ‘ ファイル名 ’
	原　因	パラメータ・ファイルの読み込みができません。
	ユーザの処置	正しいパラメータ・ファイルを指定してください。
F3021	メッセージ	Memory allocation failed
	原　因	メモリが足りません。
	ユーザの処置	必要なメモリを確保してください。
F3030	メッセージ	On-chip debug is not supported for this device
	原　因	オンチップ・ディバグ機能は，指定されているデバイスでは使用できません。
	ユーザの処置	オンチップ・ディバグ機能を指定しないでください。
F3031	メッセージ	Security ID is not supported for this device
	原　因	セキュリティ ID は，指定されているデバイスでは使用できません。
	ユーザの処置	セキュリティ ID を指定しないでください。

表 12-3 リンカのエラー・メッセージ

エラー番号	エラー・メッセージ	
F3101	メッセージ	'ファイル名 'invalid input file（or made by different hostmachine）
	原　因	オブジェクト・モジュール・ファイル以外のファイルを入力しようとしたか，互換性のないホスト・マシンで作成されたオブジェクト・モジュール・ファイルをリンクしようとしました。
F3102	メッセージ	Directive syntax error
	原　因	ディレクティブの記述がまちがっています。
F3103	メッセージ	'ファイル名 'Illegal processor type
	原　因	アセンブル，またはコンパイルの対象デバイスが，このリンカの対象デバイスではありません。
	ユーザの処置	オブジェクト・モジュールファイルが正しいことを確認してください。リンカの扱えるアセンブル，またはコンパイルの対象デバイスを確認してください。また，オーバレイ・ファイルが正しいバージョンであることを確認してください（リンカは，アセンブラのオーバレイ・ファイルの一部を参照して対象デバイス固有の情報を得ています）。
F3104	メッセージ	ファイル名 'Different processor type from first input file　最初に入力したファイル名 '
	原　因	最初に入力したオブジェクト・モジュール・ファイルと対象デバイスの異なるオブジェクト・モジュール・ファイルを入力しました。
W3105	メッセージ	Library file　'ファイル名 ' has no public symbol
	原　因	ライブラリ・ファイルにパブリック・シンボルが存在しません。そのため，ライブラリ・ファイルに含まれるオブジェクト・モジュールはリンクされません。
F3106	メッセージ	Can't create temporary file　'ファイル名 '
	原　因	テンポラリ・ファイルが作成できません。
E3107	メッセージ	Name　'名前 ' in directive has already defined
	原　因	ディレクティブのメモリ領域として，予約語，またはすでに定義している名前を定義しようとしました。 この名前（予約語，メモリ空間名，メモリ領域名）は，すでに登録されています。
E3108	メッセージ	Overlapped memory area　'メモリ領域 1' and　'メモリ領域 2'
	原　因	メモリ・ディレクティブでメモリ領域のアドレスが重複しています。
E3109	メッセージ	Memory area　'メモリ領域名 ' too long name（up to 31 characters）
	原　因	ディレクティブ中でのメモリ領域名の指定が長すぎます。 ディレクティブ中でのメモリ領域名の長さの制限は 32 文字です。
E3110	メッセージ	Memory area　'メモリ領域名 ' already defined
	原　因	メモリ・ディレクティブで指定されたメモリ領域は，すでに登録されています。
E3111	メッセージ	Memory area　'メモリ領域名 ' redefinition out of range
	原　因	メモリ・ディレクティブで指定されているメモリ領域の範囲は再定義可能な範囲を越えています。

表 12-3 リンカのエラー・メッセージ

エラー番号	エラー・メッセージ	
E3112	メッセージ	Segment 'セグメント名' wrong allocation type
	原　因	マージ・ディレクティブでセグメントの配置型の指定が間違っています。
C3113	メッセージ	Linker internal error
	原　因	内部エラー
	ユーザの処置	特約店，または当社までご連絡ください。
E3114	メッセージ	Illegal number
	原　因	ディレクティブ中の数値の記述に誤りがあります。
E3115	メッセージ	Too large value（up to 1048575/0FFFFFH）
	原　因	ディレクティブ中で 1048575（0FFFFFH）を越える値が記述されました。
E3116	メッセージ	Memory area 'メモリ領域名' definition out of range
	原　因	メモリ・ディレクティブにおいて，メモリ領域の先頭アドレスとサイズの和が，1048575（0FFFFFH）を越えました。
E3117	メッセージ	Can't find target chip in all modules
	原　因	入力したオブジェクト・モジュール・ファイルすべてに対してシリーズ共通オブジェクト指定 (-COMMON) オプションが指定されているため，対象デバイスが判別できません。
	ユーザの処置	必要のないシリーズ共通オブジェクト指定 (-COMMON) オプションを外してください。
E3201	メッセージ	Multiple segment definition 'セグメント名' in merge directive
	原　因	マージ・ディレクティブで指定されたセグメントは，すでに登録されています（同じセグメントを複数のマージ・ディレクティブで割り付け指定しようとしています）。
E3202	メッセージ	Segment type mismatch 'セグメント 1' in file 'セグメント 2' -ignored
	原　因	このセグメントと同じ名前で，異なるセグメント・タイプの再配置属性を持つセグメントが存在しています。
F3203	メッセージ	Segment 'セグメント名' unknown segment type
	原　因	入力したオブジェクト・モジュール・ファイルのセグメント情報に誤りがあります（出力セグメントの結合型の指定がまちがっています）。
E3204	メッセージ	Memory area/space '名前' not defined
	原　因	マージ・ディレクティブで指定されたメモリ領域名／メモリ空間名は定義されていません。
E3205	メッセージ	Name '名前' in directive has bad attribute
	原　因	ディレクティブのセグメント名，メモリ領域名，メモリ空間名のいずれかに指定できないものを記述しています（メモリ領域名を指定すべきところにメモリ空間名を指定したなど）。
E3206	メッセージ	Segment 'セグメント名' can't allocate to memory-ignored
	原　因	セグメントをメモリ領域に割り付けることができません（セグメントを割り付けるのに十分なメモリ領域が存在しません）。

表 12-3 リンカのエラー・メッセージ

エラー番号	エラー・メッセージ	
E3207	メッセージ	Segment 'セグメント名' has illegal segment type
	原　因	このセグメントの型情報が不正です。
E3208	メッセージ	Segment 'セグメント名' may not change attribute
	原　因	アセンブル時に再配置属性を AT xxxxH' としたセグメント，または ORG 疑似命令により作成したセグメントに対し，ディレクティブで結合型を変更しようとしました。
E3209	メッセージ	Segment 'セグメント名' may not change arrangement
	原　因	アセンブル時に再配置属性を AT xxxxH' としたセグメント，または ORG 疑似命令により作成したセグメントに対し , ディレクティブで配置アドレスを変更しようとしました。
	ユーザの処置	リンク時に結合型を指定するセグメントに対しては，アセンブル時に配置アドレスを指定しないでください。
E3210	メッセージ	Segment 'セグメント名' is not exist-ignored
	原　因	ディレクティブで指定されたセグメントが存在しません。
E3211	メッセージ	Bank type mismatch 'シンボル名' in file 'ファイル名' –ignored
	原　因	シンボルのバンク番号の指定に矛盾があります。
	ユーザの処置	シンボルのバンク番号が正しいことを確認してください。
E3301	メッセージ	Relocatable object code address out of range（file 'ファイル名'，segment 'セグメント名'，address xxxxH, type 'アドレシング・タイプ'）
	原　因	入力したオブジェクト・モジュール・ファイル・中に含まれるリロケータブル・オブジェクト・コードの修正情報が，オブジェクト・コードの存在しないアドレスに対して出力されています（リロケーション・エントリのアドレスが，オリジン・データの範囲外にあります。
	ユーザの処置	シンボルの参照が正しいことを確認してください。
E3302	メッセージ	Illegal symbol index in line number（file 'ファイル名'，segment 'セグメント名'）
	原　因	入力したオブジェクト・モジュール・ファイル中に含まれるディバグ用行番号情報に誤りがあり，シンボル情報を正しく参照していません。行番号のインデクスとシンボル・インデクスの対応がとれていません。
E3303	メッセージ	Can't find symbol index in relocatable object code（file 'ファイル名'，segment 'セグメント名'，address xxxxH, type 'アドレシング・タイプ'）
	原　因	入力したオブジェクト・モジュール・ファイル中に含まれるリロケータブル・コードの修正情報に誤りがあり，シンボル情報を正しく参照していません。リロケーション・エントリとシンボル・インデクスの対応がとれていません。
	ユーザの処置	シンボル，変数などの参照方法が正しいことを確認してください。
E3304	メッセージ	Operand out of range（segment 'セグメント名'，address xxxxH, type 'アドレシング・タイプ'）
	原　因	リロケータブル・オブジェクト・コードの解決に用いているオペランド値が，命令に対応したオペランドの値の範囲を越えています。
	ユーザの処置	オペランド値をアドレシング・タイプごとに定められているオペランドの範囲に納まるようにソース・プログラムを記述してください。

表 12-3 リンカのエラー・メッセージ

エラー番号	エラー・メッセージ	
E3305	メッセージ	Even value expected（segment 'セグメント名 '，address xxxxH, type 'アドレシング・タイプ '）
	原　因	callt，または saddrp アドレシングのリロケータブル・オブジェクト・コード解決に用いているオペランド値が奇数になりました（callt，または saddrp アドレシングのオペランドは偶数でなければなりません）。
E3306	メッセージ	A multiple of 4 value expected（segment 'セグメント名 '，address xxxxH, type'アドレシング・タイプ '）
	原　因	saddr アドレシングのリロケータブル・オブジェクト・コードの解決に用いているオペランド値が 4 の倍数になりませんでした。
F3401	メッセージ	'ファイル名 ' Bad symbol table
	原　因	入力したオブジェクト・モジュール・ファイルのシンボル情報が不正です。入力ファイルのシンボル・エントリが　file' シンボルで始まっていません。
F3402	メッセージ	File　'ファイル名 ' has no string table for symbol
	原　因	入力したオブジェクト・モジュール・ファイルのシンボル情報が不正です。
	ユーザの処置	もう一度アセンブル，またはコンパイルし直してください。アセンブラのシンボル認識文字数を 8 文字，コンパイラの認識文字数を 7 文字にすることで回避可能な場合があります。
E3403	メッセージ	Symbol　'シンボル名 ' unmatched type in file ' ファイル名 1'. First defined in file　'ファイル名 2'
	原　因	同名外部定義／参照シンボルの型がファイル 1 とファイル 2 で異なっています。
E3404	メッセージ	Multiple Symbol definition　'シンボル名 ' in file 'ファイル名 1'. First defined in file　'ファイル名 2'
	原　因	オブジェクト・モジュール・ファイル 1 中で定義されている PUBLIC シンボルは，オブジェクト・モジュール・ファイル 2 ですでに PUBLIC 宣言されています。
E3405	メッセージ	Undefined symbol　'シンボル名 ' in file　'ファイル名 '
	原　因	ファイルで EXTRN 宣言されているシンボルは，他のファイルで PUBLIC 宣言されていません。
W3406	メッセージ	Stack area less than 10 bytes
	原　因	確保したスタック領域の大きさが 10 バイト以下です（-S オプションで指定されたメモリ領域に確保できたスタック領域の大きさが 10 バイト以下です）。
W3407	メッセージ	Can't allocate stack area
	原　因	スタック領域を確保するメモリ領域に，空き領域がありません（-S オプションで指定されたメモリ領域にスタック領域を確保できません）。
E3410	メッセージ	Multiple module name definition ' モジュール名 ' in file ' ファイル 1'. First defined in file ' ファイル 2'
	原　因	オブジェクト・モジュール・ファイル 1 のモジュール名とオブジェクト・モジュール・ファイル 2 のモジュール名が同じです。

表 12-3 リンカのエラー・メッセージ

エラー番号	エラー・メッセージ	
W3411	メッセージ	Different REL type in 'ファイル名'
	原　因	オブジェクト・モジュール・ファイルの型バージョンに相違があります。
	ユーザの処置	最新版でアセンブル，またはコンパイルし直してください。
E3415	メッセージ	-RD/QF/etc. and Not -RD/QF/etc. REL are mixed
	原　因	プログラム全体で同じ指定でなければいけないコンパイラの最適化オプションに関して，異なる指定をしたオブジェクト・ファイルが入力されました。同じ指定でコンパイルし直してください。
W3416	メッセージ	Multiple CAP/NOCAP are in file 'ファイル名（オプション）'. Defined first one in file 'ファイル名（オプション）'
	原　因	全入力オブジェクト・モジュール・ファイルを対象にアセンブル，またはコンパイル・オプションの CAP/NOCAP が一致していません。
W3417	メッセージ	The version of ツール名 in file 'ファイル名' are more than one. Used the first one in file 'ファイル名'
	原　因	全入力オブジェクト・モジュール・ファイルを対象にリンクまでに使用した各ツール（CC78K0，ST78K0，RA78K0），およびデバイス・ファイルのバージョンに相違があります。
W3418	メッセージ	File 'ファイル名' is old. Can't find TOOL infomation
	原　因	入力したオブジェクト・モジュール・ファイルに TOOL 情報がない場合出力します。 通常，旧（DF 非対応）をリンクすると必ず出力します。
W3420	メッセージ	File 'ファイル名' already has had error(s) /warning(s) by 'ツール名'
	原　因	リンクまでに使用していた各ツール（CC78K0，ST78K0，RA78K0）においてエラー，またはワーニング・メッセージを出力しています。
E3424	メッセージ	-ZF REL and not -ZF REL are mixed in file 'ファイル名'
	原　因	フラッシュ ROM 内蔵製品のブート領域 ROM プログラムのロード・モジュールとフラッシュ領域プログラムのオブジェクト・モジュールをリンクする際，このオブジェクト・モジュールの中にコンパイル時に -ZF オプションを指定していないものがあります。
E3425	メッセージ	There are different function ID in same name 関数名'（file 'ファイル名'）
	原　因	コンパイラで EXT_FUNC 宣言された同名の関数が，異なる ID 値を持っています。
E3426	メッセージ	Multiple input BOOT file 'ファイル名'
	原　因	フラッシュ ROM 内蔵製品のブート領域 ROM プログラムのロード・モジュールとフラッシュ領域プログラムのオブジェクト・モジュールをリンクする際，ブート領域 ROM プログラムのロード・モジュールが複数入力されました。
E3427	メッセージ	BOOT REL and -ZF REL are mixed in file 'ファイル名'
	原　因	-ZB オプション指定時のリンクにおいて，コンパイル時に -ZF オプション指定されたオブジェクト・モジュールが入力されています。

表 12-3 リンカのエラー・メッセージ

エラー番号	エラー・メッセージ	
E3428	メッセージ	FLASH start address larger than ROM max address
	原　因	フラッシュ ROM 領域の先頭のアドレスが対象デバイスの ROM エンド・アドレスより大きくなっています。
E3429	メッセージ	BOOT segment　'セグメント名 ' are found in FLASH file　'ファイル名 '
	原　因	フラッシュ ROM 内蔵製品のブート領域 ROM プログラムのロード・モジュールとフラッシュ領域 ROM プログラムのオブジェクト・モジュールのリンク時に，このオブジェクト・モジュールにフラッシュ ROM 領域の先頭のアドレスより小さい配置アドレスのセグメントが存在しています。
E3430	メッセージ	Different FLASH address in file　'ファイル名 '
	原　因	入力ファイルのフラッシュ ROM 領域の先頭アドレスがすべて同じではありません。
E3431	メッセージ	There are different function name in same ID（関数名）(file　'ファイル名 ')
	原　因	コンパイラで EXT_FUNC 宣言された複数の関数が，同じ ID 値をもっています。
E3432	メッセージ	Illegal allocate an EXT_FUNC function　'関数名 '（file　'ファイル名 '）
	原　因	-ZB オプション指定時のリンクにおいて , コンパイラ EXT_FUNC 宣言された関数の実体が存在しています。
F3901	メッセージ	Can't open overlay file　'ファイル名 '
	原　因	オーバレイ・ファイルがオープンできません。
	ユーザの処置	オーバレイ・ファイルが正しいディレクトリ（実行形式プログラムがあるディレクトリ）にあることを確認してください。
F3902	メッセージ	file　'ファイル名 ' file not found
	原　因	指定されたライブラリ・ファイルをオープンできません。
F3903	メッセージ	Can't read input file　'ファイル名 '
	原　因	入力ファイルとして指定されたオブジェクト・モジュール・ファイルを読むことができません。
F3904	メッセージ	Can't open output file　'ファイル名 '
	原　因	出力ファイルをオープンできません。
	ユーザの処置	出力ファイルを作成しようとしたディスクの状態（空き容量，メディアの状態など）を確認してください。
F3905	メッセージ	Can't create temporary file　'ファイル名 '
	原　因	シンボル・エントリ用のテンポラリ・ファイルを作成することができません。
	ユーザの処置	テンポラリ・ファイルを作成しようとしたディスクの状態（空き容量，メディアの状態など）を確認してください。
F3906	メッセージ	Can't write map file　'ファイル名 '
	原　因	リンク・リスト・ファイルにデータを書き込めません。
	ユーザの処置	リンク・リスト・ファイルを作成しようとしたディスクの状態（空き容量，メディアの状態など）を確認してください。

表 12-3 リンカのエラー・メッセージ

エラー番号	エラー・メッセージ	
F3907	メッセージ	Can't write output file 'ファイル名'
	原　因	ロード・モジュール・ファイルに書き込みができません。
	ユーザの処置	出力ファイルを作成しようとしたディスクの状態（空き容量，メディアの状態など）を確認してください。
F3908	メッセージ	Can't access temporary file 'ファイル名'
	原　因	テンポラリ・ファイルに書き込みができません。
	ユーザの処置	テンポラリ・ファイルを作成しようとしたディスクの状態（空き容量，メディアの状態など）を確認してください。
F3909	メッセージ	Can't read DEVICE_File file 'デバイス・ファイル名'
	原　因	リンクまでに使用した各ツール（CC78K0，ST78K0，RA78K0）で指定したデバイスに対応したデバイス・ファイルの読み込みができません。

注意　E3301- E3306 のメッセージの中で，‘address xxxxH’ として表示されるアドレスは，セグメント配置後の絶対アドレスです

12.5 オブジェクト・コンバータのエラー・メッセージ

表 12-4 オブジェクト・コンバータのエラー・メッセージ

エラー番号	エラー・メッセージ	
F4001	メッセージ	Missing input file
	原　因	入力ファイルを指定していません。
	ユーザの処置	入力ファイルを指定してください。
F4002	メッセージ	Too many input files
	原　因	入力ファイルが 2 つ以上指定されました。
	ユーザの処置	入力ファイルを 1 つだけ指定してください。
F4004	メッセージ	Illegal file name ‘ ファイル名 ’
	原　因	ファイル名に不当な文字があるか，または文字数が制限を越えています。
	ユーザの処置	ファイル名を正しい文字，および文字数にしてください。
F4005	メッセージ	Illegal file specification ‘ ファイル名 ’
	原　因	不当なファイルが指定されました。
	ユーザの処置	正しいファイル名を指定してください。
F4006	メッセージ	File not found ’ファイル名 ’
	原　因	指定された入力ファイルが存在しません。
	ユーザの処置	C コンパイラのスタートアップ・ルーチンをリンクしている場合は，「“ スタートアップ・ルーチン名 ”.lmf」として出力されます。この場合，リンカ・オプションで「-o*.lmf」のように出力ファイル名を指定してください。
F4008	メッセージ	File specification conflicted ‘ ファイル名 ’
	原　因	入出力ファイル名が重複して指定されました。
	ユーザの処置	入出力ファイル名は異なるものを指定してください。
F4009	メッセージ	Unable to make file ‘ ファイル名 ’
	原　因	指定されたファイルにライト・プロテクトがかかっています。
	ユーザの処置	ファイルのライト・プロテクトを解除してください。
F4010	メッセージ	Directory not found ‘ ファイル名 ’
	原　因	出力ファイル名中に存在しないドライブ，またはディレクトリが含まれています。
	ユーザの処置	存在するドライブ，およびディレクトリ名を指定してください。
F4011	メッセージ	Illegal path ‘ オプション ’
	原　因	パラメータにパスを指定するオプションで，パス名以外が指定されました。
	ユーザの処置	正しいパス名を指定してください。

表 12-4 オブジェクト・コンバータのエラー・メッセージ

エラー番号	エラー・メッセージ	
F4012	メッセージ	Missing parameter ‘ オプション ’
	原　因	必要なパラメータが指定されていません。
	ユーザの処置	パラメータを指定してください。
F4013	メッセージ	Parameter not needed ‘ オプション ’
	原　因	不要なパラメータが指定されています。
	ユーザの処置	不要なパラメータを削除してください。
F4014	メッセージ	Out of range ‘ オプション ’
	原　因	指定数値が範囲外です。
	ユーザの処置	正しい数値を指定してください。
F4015	メッセージ	Parameter is too long ‘ オプション ’
	原　因	パラメータの文字数が制限を越えています。
	ユーザの処置	パラメータの文字数を制限内にしてください。
F4016	メッセージ	Illegal parameter ‘ オプション ’
	原　因	パラメータの文法が誤っています。
	ユーザの処置	正しいパラメータを指定してください。
F4017	メッセージ	Too many parameters ‘ オプション ’
	原　因	パラメータの総数が制限を越えています。
	ユーザの処置	パラメータの総数を制限内にしてください。
F4018	メッセージ	Option is not recognized ‘ オプション ’
	原　因	オプション名が誤っています。
	ユーザの処置	正しいオプション名を指定してください。
F4019	メッセージ	Parameter file nested
	原　因	パラメータ・ファイル中に -F オプションが指定されました。
	ユーザの処置	パラメータ・ファイル中に -F オプションを指定しないでください。
F4020	メッセージ	Parameter file read error ‘ ファイル名 ’
	原　因	パラメータ・ファイルの読み込みができません。
	ユーザの処置	正しいパラメータ・ファイルを指定してください。
F4021	メッセージ	Memory allocation failed
	原　因	メモリが足りません。
	ユーザの処置	必要なメモリを確保してください。

表 12-4 オブジェクト・コンバータのエラー・メッセージ

エラー番号	エラー・メッセージ	
F4100	メッセージ	'ファイル名 ' Illegal processor type
	原　因	アセンブル，またはコンパイルの対象デバイスが，このプログラムの対象デバイスと異なります。
	ユーザの処置	ロード・モジュール・ファイルが正しいかどうか，そしてアセンブル，またはコンパイルの対象デバイスを確認してください。また，デバイス・ファイルのバージョンが正しいかどうかを確認してください。
F4101	メッセージ	'ファイル名 ' invalid input file（or made by different hostmachine）
	原　因	ロード・モジュール・ファイル以外のファイルを入力しようとしたか，または互換性のないホスト・マシンで作成されたロード・モジュール・ファイルをコンバートしようとしました。
F4103	メッセージ	Symbol　'シンボル名 ' Illegal attribute
	原　因	入力ファイルのシンボル属性に誤りがあります。
F4104	メッセージ	'ファイル名 ' Illegal input file-not linked
	原　因	オブジェクト・モジュール・ファイルを入力しようとしています。
F4105	メッセージ	Insufficient memory in hostmachine
	原　因	プログラムが動作するために十分なメモリがありません。
F4106	メッセージ	Illegal symbol table
	原　因	入力したロード・モジュール・ファイルのシンボル・テーブルに誤りがあります。
	ユーザの処置	ソースが C 言語で記述されている場合は，C ソースのアセンブラ記述が次の注意事項に該当しないかを確認してください。 〈注意事項〉 ローカル・シンボルを使用している場合は，?L の文字列で始まるシンボル（?L@01 , ?L@symmailto:?L@01 など）を使用してください。ただし，このシンボルは 8 文字以下にしてください。また，このシンボルを外部定義（PUBLIC 宣言）しないでください。
F4107	メッセージ	Can't specify -U option for ROMless device
	原　因	内部 ROM のない品種にオブジェクト充てん値指定（-U）オプションを指定しています。
E4200	メッセージ	Undefined symbol　'シンボル名 '
	原　因	アドレスが解決していないシンボルがあります。
	ユーザの処置	シンボル値の定義をしてください。このシンボルを外部参照シンボルとして参照しますが，外部定義していないときはこのシンボル値を定義しているモジュ ール外で外部定義してください。
E4201	メッセージ	Out of address range
	原　因	ロード・モジュール・ファイルのオブジェクトのアドレスが範囲を越えています。
W4300	メッセージ	xxxxxxH-yyyyyyH overlapped
	原　因	xxxxxxH から yyyyyyH までのアドレスに対するオブジェクトが重複して出力されています。

表 12-4 オブジェクト・コンバータのエラー・メッセージ

エラー番号	エラー・メッセージ	
W4301	メッセージ	Can’t initialize RAM area ’アドレス’- ’アドレス’
	原　因	RAM 領域に初期値データが出力されています。
	ユーザの処置	アセンブリ・ソースで DSEG 内に DB/DW が記述されている場合は，DS に変更するか，CSEG 内で DB/DW 命令を記述するようにしてください。
F4900	メッセージ	Can't open file ’ファイル名’
	原　因	ファイルがオープンできません。
F4901	メッセージ	Can't close file ’ファイル名’
	原　因	ファイルがクローズできません。
F4902	メッセージ	Can't read file ’ファイル名’
	原　因	ファイルが正しく読めません。
F4903	メッセージ	Can't access file ’ファイル名’
	原　因	ファイルが正しく読み込み，または書き込みができません。
F4904	メッセージ	Can't write file ’ファイル名’
	原　因	出力ファイルに正しくデータが書き込めません。
F4905	メッセージ	Can't open overlay file ‘ ファイル名’
	原　因	オーバレイ・ファイルがオープンできません。
	ユーザの処置	オーバレイ・ファイルが実行形式と同じディレクトリにあるかどうかを調べてください。
C4999	メッセージ	Object Converter internal error
	原　因	内部エラーです。
	ユーザの処置	特約店，または当社までご連絡ください。

12.6 ライブラリアンのエラー・メッセージ

表 12-5 ライブラリアンのエラー・メッセージ

エラー番号	エラー・メッセージ	
F5001	メッセージ	Missing input file
	原　因	オプションのみの指定で，入力ファイルが 1 つも指定されていません。
F5002	メッセージ	Too many input file
	原　因	入力ファイルの総数が制限を越えて指定されました。
F5003	メッセージ	Unrecognized string　'???'
	原　因	対話形式のコマンド行にオプション以外のものが指定されました。
F5004	メッセージ	Illegal file name　'ファイル名'
	原　因	ファイル名に OS で許されない文字があるか，文字数が制限を越えています。
F5005	メッセージ	Illegal file specification　'ファイル名'
	原　因	ファイル名に不当なものが指定されました。
F5006	メッセージ	File not found　'ファイル名'
	原　因	指定された入力ファイルが存在しません。
F5007	メッセージ	Input file specification overlapped　'ファイル名'
	原　因	入力ファイル名が重複して指定されました。
F5008	メッセージ	File specification conflicted　'ファイル名'
	原　因	入出力ファイル名が重複して指定されました。
F5009	メッセージ	Unable to make file　'ファイル名'
	原　因	指定された出力ファイルが作成できません。
F5010	メッセージ	Directory not found　'ファイル名'
	原　因	出力ファイル名中に存在しないドライブ，またはディレクトリが含まれています。
F5011	メッセージ	Illegal path　'ファイル名'
	原　因	パラメータにパス名を指定するオプションで，パス名以外が指定されました。
F5012	メッセージ	Missing parameter　'オプション'
	原　因	必要なパラメータが指定されていません。
F5013	メッセージ	Parameter not needed　'オプション'
	原　因	不要なパラメータが指定されました。
F5014	メッセージ	Out of range　'オプション'
	原　因	指定値が範囲外です。
F5015	メッセージ	Parameter is too long　'オプション'
	原　因	パラメータの文字数が制限を越えて指定されました。

表 12-5 ライブラリアンのエラー・メッセージ

エラー番号	エラー・メッセージ	
F5016	メッセージ	Illegal parameter 'オプション'
	原　因	パラメータの文法に誤りがあります。
F5017	メッセージ	Too many parameter 'オプション'
	原　因	パラメータの総数が制限を越えました。
F5018	メッセージ	Option is not recognized 'オプション'
	原　因	誤ったオプションが指定されました。
F5021	メッセージ	Memory allocation failed
	原　因	メモリ・アロックに失敗しました。
F5024	メッセージ	Illegal character
	原　因	不当な文字，または文字列があります。
F5025	メッセージ	Qualifier is not unique.
	原　因	修飾子の省略形がユニークではありません。
F5026	メッセージ	Umbigous input redirect.
	原　因	'<' の後にファイル名がない。または '< △ファイル名' が 2 回以上指定されています。
C5100	メッセージ	Internal error
	原　因	内部エラーが発生しました。
E5101	メッセージ	Invalid sub command
	原　因	サブコマンド名が誤っています。
E5102	メッセージ	Invalid syntax
	原　因	サブコマンドのパラメータ指定に誤りがあります。
E5103	メッセージ	Illegal input file-different target chip（file：ファイル名）
	原　因	入力したオブジェクト・モジュール・ファイルの対象デバイス指定に誤りがあります。
E5104	メッセージ	Illegal library file-different target chip（file：ファイル名）
	原　因	指定ライブラリ・ファイルの対象デバイスに誤りがあります。
E5105	メッセージ	Module not found（module：ファイル名）
	原　因	指定モジュールがライブラリ・ファイル中に存在しません。
E5106	メッセージ	Module already exists（module：ファイル名）
	原　因	同名のモジュールが，すでに更新ライブラリ・ファイル，または他の入力ファイル内に存在しています。
E5107	メッセージ	Master library file is not specify
	原　因	以前のオペレーションで，まだ更新ライブラリ・ファイルの指定がされていないのに，'.' での置き換えが指定されました。

表 12-5 ライブラリアンのエラー・メッセージ

エラー番号	エラー・メッセージ	
E5108	メッセージ	Multiple transaction file（file：ファイル名）
	原　因	入力オブジェクト・モジュール・ファイル名が重複しています。
E5109	メッセージ	Public symbol already exists（symol：シンボル名）
	原　因	同名の外部定義シンボル名が，すでに更新ライブラリ・ファイル，または他の入力ファイル内に存在しています。
E5110	メッセージ	File specification conflicted（file：ファイル名）
	原　因	指定した入力ファイル名と出力ファイル名が一致しています。
E5111	メッセージ	Illegal file format（file：ファイル名）
	原　因	更新ライブラリ・ファイル，または他の入力ファイルのフォーマットが異常です。
E5112	メッセージ	Library file not found（file：ファイル名）
	原　因	指定したライブラリ・ファイルが見つかりません。
E5113	メッセージ	Object module file not found（file：ファイル名）
	原　因	指定したオブジェクト・モジュール・ファイルが見つかりません。
E5114	メッセージ	No free space for temporary file
	原　因	ディスク上のテンポラリ・ファイルを作成するための十分な空き容量がありません。
E5115	メッセージ	Not enough memory
	原　因	プログラムが動作するための，十分なメモリが確保できません。
E5116	メッセージ	Sub command Buffer full
	原　因	サブコマンドの継続行の長さが制限（128×15 文字）を越えています。 サブコマンド・ファイル中のサブコマンドの 1 行の長さが制限（128 文字）を越えています。
E5117	メッセージ	Can not use device file
	原　因	入力ファイルにデバイス型のファイルが指定されました。 list コマンドの入出力ファイルに CLOCK が指定されました。 出力オブジェクト・モジュール・ファイル，または出力ライブラリ・ファイルに PRN，CON，CLOCK が指定されました。
E5118	メッセージ	Illegal path（file：ファイル名）
	原　因	指定ファイルのパス名に誤りがあります。
W5201	メッセージ	Module not found（module：ファイル名）
	原　因	REPLACE 指定されたモジュールがライブラリ・ファイル中に存在しません。
F5901	メッセージ	File open error（file：ファイル名）
	原　因	ファイルがオープンできません。
F5902	メッセージ	File read error（file：ファイル名）
	原　因	ファイルが正しく読めません。

表 12-5 ライブラリアンのエラー・メッセージ

エラー番号	エラー・メッセージ	
F5903	メッセージ	File write error（file：ファイル名）
	原　因	ファイルに正しくデータが書き込めません。
F5904	メッセージ	File seek error（file：ファイル名）
	原　因	ファイル・シーク・エラーが発生しました。
F5905	メッセージ	File close error（file：ファイル名）
	原　因	ファイルがクローズできません。

12.7 リスト・コンバータのエラー・メッセージ

表 12-6 リスト・コンバータのエラー・メッセージ

エラー番号	エラー・メッセージ	
F6001	メッセージ	Missing input file
	原　因	入力ファイルを指定していません。
	ユーザの処置	入力ファイルを指定してください。
F6002	メッセージ	Too many input files
	原　因	入力ファイルが 2 つ以上指定されました。
	ユーザの処置	入力ファイルを 1 つだけ指定してください。
F6004	メッセージ	Illegal file name ‘ ファイル名 ’
	原　因	ファイル名に不当な文字があるか，または文字数が制限を越えています。
	ユーザの処置	ファイル名を正しい文字，および文字数にしてください。
F6005	メッセージ	Illegal file specification ‘ ファイル名 ’
	原　因	不当なファイルが指定されました。
	ユーザの処置	正しいファイル名を指定してください。
F6006	メッセージ	File not found ‘ ファイル名 ’
	原　因	指定された入力ファイルが存在しません。
	ユーザの処置	存在するファイル名を指定してください。
F6008	メッセージ	File specification conflicted ‘ ファイル名 ’
	原　因	入出力ファイル名が重複して指定されました。
	ユーザの処置	入出力ファイル名は異なるものを指定してください。
F6009	メッセージ	Unable to make file ‘ ファイル名 ’
	原　因	指定されたファイルにライト・プロテクトがかかっています。
	ユーザの処置	ファイルのライト・プロテクトを解除してください。
F6010	メッセージ	Directory not found ‘ ファイル名 ’
	原　因	出力ファイル名中に存在しないドライブ，またはディレクトリが含まれています。
	ユーザの処置	存在するドライブ，およびディレクトリ名を指定してください。
F6011	メッセージ	Illegal path ‘ オプション ’
	原　因	パラメータにパスを指定するオプションで，パス名以外が指定されました。
	ユーザの処置	正しいパス名を指定してください。
F6012	メッセージ	Missing parameter ‘ オプション ’
	原　因	必要なパラメータが指定されていません。
	ユーザの処置	パラメータを指定してください。

表 12-6 リスト・コンバータのエラー・メッセージ

エラー番号	エラー・メッセージ	
F6013	メッセージ	Parameter not needed ‘ オプション ’
	原　因	不要なパラメータが指定されています。
	ユーザの処置	不要なパラメータを削除してください。
F6014	メッセージ	Out of range ‘ オプション ’
	原　因	指定数値が範囲外です。
	ユーザの処置	正しい数値を指定してください。
F6015	メッセージ	Parameter is too long ‘ オプション ’
	原　因	パラメータの文字数が制限を越えています。
	ユーザの処置	パラメータの文字数を制限内にしてください。
F6016	メッセージ	Illegal parameter ' オプション '
	原　因	パラメータの文法が誤っています。
	ユーザの処置	正しいパラメータを指定してください。
F6017	メッセージ	Too many parameters ' オプション '
	原　因	パラメータの総数が制限を越えています。
	ユーザの処置	パラメータの総数を制限内にしてください。
F6018	メッセージ	Option is not recognized ' オプション '
	原　因	オプション名が誤っています。
	ユーザの処置	正しいオプション名を指定してください。
F6019	メッセージ	Parameter file nested
	原　因	パラメータ・ファイル中に -F オプションが指定されました。
	ユーザの処置	パラメータ・ファイル中に -F オプションを指定しないでください。
F6020	メッセージ	Parameter file read error ' ファイル名 '
	原　因	パラメータ・ファイルの読み込みができません。
	ユーザの処置	正しいパラメータ・ファイルを指定してください。
F6021	メッセージ	Memory allocation failed
	原　因	メモリが足りません。
	ユーザの処置	必要なメモリを確保してください。
F6101	メッセージ	File is not 78K/0　’ファイル名 ’
	原　因	入力ファイル名が 78K/0 のものではありません。
F6102	メッセージ	Load module file is not executable　’ファイル名 ’
	原　因	ロード・モジュール・ファイル以外のファイルを入力しようとしたか，互換性のないホスト・マシンで作成されたロード・モジュール・ファイルをコンバートしようとしました。

表 12-6 リスト・コンバータのエラー・メッセージ

エラー番号	エラー・メッセージ	
F6103	メッセージ	Load module file has relocation data 'ファイル名'
	原　因	ロード・モジュール・ファイルのアドレスが解決されていません。
F6104	メッセージ	Object module file is executable 'ファイル名'
	原　因	オブジェクト・モジュール・ファイルが実行形式です。
F6105	メッセージ	Segment name is not found in load module file 'セグメント名'
	原　因	ロード・モジュール・ファイル内にオブジェクト・モジュール・ファイルのセグメント名が見つかりません。
F6106	メッセージ	Segment name is not found in object module file 'ファイル名'
	原　因	オブジェクト・モジュール・ファイル内にアセンブル・リスト・ファイルのセグメント名が見つかりません。
F6107	メッセージ	Not enough memory
	原　因	作業用メモリが足りません。
F6108	メッセージ	Load module file has no symbol date 'ロード・モジュール名'
	原　因	リンカで -NG オプションを指定したためロード・モジュール中にシンボル情報が出力されていません。
F6109	メッセージ	Overlay file can not open 'パス名'
	原　因	アセンブラのオーバーレイ・ファイルがオープンできません。
F6110	メッセージ	Illegal assembler list file 'ファイル名'
	原　因	入力されたアセンブル・リストがアセンブル・リスト以外のファイルです。
W6701	メッセージ	Load module file is older than object module file 'ロード・モジュール・ファイル名，オブジェクト・モジュール・ファイル名'
	原　因	オブジェクト・モジュール・ファイル名よりも古いロード・モジュール・ファイル名が指定されました。
W6702	メッセージ	Load module file is older than assemble module file 'ロード・モジュール・ファイル名，アセンブル・リスト・ファイル名'
	原　因	アセンブル・リスト・ファイル名よりも古いロード・モジュール・ファイル名が指定されました。
W6703	メッセージ	Assemble list has error statement 'ファイル名'
	原　因	アセンブル・リスト内にエラー行があります。
W6704	メッセージ	Segment name is not found in assemble list file 'セグメント名'
	原　因	アセンブル・リスト内にオブジェクト・モジュール・ファイルのセグメント名が見つかりません。
W6705	メッセージ	Segment data length is different 'セグメント名'
	原　因	アセンブル・リスト・ファイル上のセグメント・データの長さとオブジェクト・モジュール・ファイル上のデータの長さが異なります。
	プログラムの処理	余分なセグメントのデータは無視して処理を実行します。

表 12-6 リスト・コンバータのエラー・メッセージ

エラー番号	エラー・メッセージ	
F6901	メッセージ	File open error has occured 'ファイル名'
	原　因	ファイルがオープンできません。
F6902	メッセージ	File read error has occured 'ファイル名'
	原　因	ファイルが正しく読めません。
F6903	メッセージ	File write error has occured 'ファイル名'
	原　因	ファイルに正しくデータが書き込めません。
F6904	メッセージ	File seek error has occured 'ファイル名'
	原　因	ファイル・シーク・エラーが発生しました。
C6999	メッセージ	Internal error
	原　因	プログラム内部エラーです。

12.8 PM plus のエラー・メッセージ

PM plus のヘルプに記載されていないエラー・メッセージについて説明します。PM plus のその他のエラー・メッセージについては，PM plus のオンライン・ヘルプを参照してください。

12.8.1 構造化アセンブラ・プリプロセッサ（ST78K0）

表 12-7 構造化アセンブラ・プリプロセッサ（ST78K0）用 DLL の表示するエラー・メッセージ

エラー種別	エラー・メッセージ	
×	メッセージ	ST78K0.EXE が PATH 環境変数で示されるディレクトリに登録されていません。
	原　因	ST78K0.EXE 実行形式が規定のディレクトリにありません。
	ユーザの処置	RA78K0 アセンブラ・パッケージを再インストールしてください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	フォルダが存在しません。 作成しますか？
	原　因	指定されたフォルダが存在しません。
	ユーザの処置	フォルダを作成するか，別のフォルダを選択してください。
	ボタン	“ OK ” …フォルダを作成し，メッセージを閉じます。 “ キャンセル ” …メッセージを閉じます。
!	メッセージ	フォルダの作成に失敗しました。
	原　因	指定されたフォルダを作成することができません。
	ユーザの処置	別のフォルダを指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	タブ数で指定した値の大小関係が不正です。
	原　因	タブ数として，大小関係が不正な値が指定されました。
	ユーザの処置	指定可能範囲の数値を指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	命令までのタブ数で指定された値が不正です。 指定可能範囲は　0 ≦命令まで≦ 97 です。
	原　因	命令までのタブ数として入力可能範囲外の数値が記述されました。
	ユーザの処置	指定可能範囲の数値を指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	オペランドまでのタブ数で指定された値が不正です。 指定可能範囲は　0 ≦オペランドまで≦ 98 です。
	原　因	オペランドまでのタブ数として入力可能範囲外の数値が記述されました。
	ユーザの処置	指定可能範囲の数値を指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	コメントまでのタブ数で指定された値が不正です。 指定可能範囲は　0 ≦コメントまで≦ 99 です。
	原　因	コメントまでのタブ数として入力可能範囲外の数値が記述されました。
	ユーザの処置	指定可能範囲の数値を指定してください。
	ボタン	“ OK ” …メッセージを閉じます。

表 12-7 構造化アセンブラ・プリプロセッサ（ST78K0）用 DLL の表示するエラー・メッセージ

エラー種別	エラー・メッセージ	
!	メッセージ	ワード・シンボル文字が不正です。
	原　因	シンボルとして使用できない文字が指定されています。
	ユーザの処置	指定可能な文字を指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	インクルード・ファイル・パスで指定したパスの指定数が不正です。
	原　因	インクルード・ファイル・パスが入力可能範囲外の数で記述されています。
	ユーザの処置	指定可能範囲の数で再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	インクルード・ファイル・パスで指定したパスの長さが不正です。
	原　因	インクルード・ファイル・パスの長さが入力可能範囲外の文字数で記述されています。
	ユーザの処置	指定可能範囲の文字数で再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	(インクルード・ファイル・パス) インクルード・ファイル・パスで指定したパスが重複しています。
	原　因	インクルード・ファイル・パスが重複して記述されています。
	ユーザの処置	重複しているパスを削除して再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	シンボル定義で指定したシンボルの指定数が不正です。 シンボルは 30 個まで指定可能です。
	原　因	シンボル定義でシンボルの数が入力可能範囲外の数で記述されています。
	ユーザの処置	指定可能範囲の数で再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	(シンボル) シンボル定義で指定したシンボルの長さが不正です。 シンボルは 31 文字まで指定可能です。
	原　因	シンボル定義でシンボルの長さが入力可能範囲外の文字数で記述されています。
	ユーザの処置	指定可能範囲の文字数で再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	(シンボル) シンボル定義で指定したシンボルが重複しています。
	原　因	シンボル定義でシンボルが重複して記述されています。
	ユーザの処置	重複しているシンボルを削除して再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。

表 12-7 構造化アセンブラ・プリプロセッサ（ST78K0）用 DLL の表示するエラー・メッセージ

エラー種別	エラー・メッセージ	
!	メッセージ	変更内容を個別オプションを設定している (ソース・ファイル名) に反映できませんでした。
	原　因	全体オプションを個別オプションへ反映した場合，個別オプションが不正な指定になってしまいます。
	ユーザの処置	個別オプションの指定を確認して再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。

12.8.2 アセンブラ（RA78K0）

表 12-8 アセンブラ（RA78K0）用 DLL の表示するエラー・メッセージ

エラー種別	エラー・メッセージ	
×	メッセージ	RA78K0.EXE が PATH 環境変数で示されるディレクトリに登録されていません。
	原　因	RA78K0.EXE 実行形式が規定のディレクトリにありません。
	ユーザの処置	RA78K0 アセンブラ・パッケージを再インストールしてください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	フォルダが存在しません。 作成しますか？
	原　因	指定されたフォルダが存在しません。
	ユーザの処置	フォルダを作成するか，別のフォルダを選択してください。
	ボタン	“ OK ” …フォルダを作成し，メッセージを閉じます。 “ キャンセル ” …メッセージを閉じます。
!	メッセージ	フォルダの作成に失敗しました。
	原　因	指定されたフォルダを作成することができません。
	ユーザの処置	別のフォルダを指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	1 行文字数で指定された値が不正です。 指定可能範囲は　72　≦　LW　≦　2046　です。
	原　因	1 行文字数として入力可能範囲外の数値が指定されました。
	ユーザの処置	指定可能範囲の数値を指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	1 ページ行数で指定された値が不正です。 指定可能範囲は　0,20　≦　LL　≦　32767　です。
	原　因	1 ページ行数として入力可能範囲外の数値が指定されました。
	ユーザの処置	指定可能範囲の数値を指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	タブ文字長で指定された値が不正です。 指定可能範囲は　0　≦　LT　≦　8　です。
	原　因	オペランドまでのタブ数として入力可能範囲外の数値が記述されました。
	ユーザの処置	指定可能範囲の数値を再度指定してください。
	ボタン	“ OK ” …メッセージを閉じます。

表 12-8 アセンブラ（RA78K0）用 DLL の表示するエラー・メッセージ

エラー種別	エラー・メッセージ	
!	メッセージ	インクルード・ファイル・パスで指定したパスの指定数が不正です。 パスは 64 個まで指定可能です。
	原　因	インクルード・ファイル・パスが入力可能範囲外の数で記述されています。
	ユーザの処置	指定可能範囲の数で再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	（インクルード・ファイル・パス） インクルード・ファイル・パスで指定したパスの長さが不正です。
	原　因	インクルード・ファイル・パスの長さが入力可能範囲外の文字数で記述されています。
	ユーザの処置	指定可能範囲の文字数で再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	（インクルード・ファイル・パス） インクルード・ファイル・パスで指定したパスが重複しています。
	原　因	インクルード・ファイル・パスが重複して記述されています。
	ユーザの処置	重複しているパスを削除して再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	シンボル定義で指定したシンボルの指定数が不正です。 シンボルは 30 個まで指定可能です。
	原　因	シンボル定義でシンボルの数が入力可能範囲外の数で記述されています。
	ユーザの処置	指定可能範囲の数で再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	（シンボル） シンボル定義で指定したシンボルの長さが不正です。 シンボルは 31 文字まで指定可能です。
	原　因	シンボル定義でシンボルの長さが入力可能範囲外の文字数で記述されています。
	ユーザの処置	指定可能範囲の文字数で再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	（シンボル） シンボル定義で指定したパスが重複しています。
	原　因	シンボル定義でシンボルが重複して記述されています。
	ユーザの処置	重複しているシンボルを削除して再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。

表 12-8 アセンブラ（RA78K0）用 DLL の表示するエラー・メッセージ

エラー種別	エラー・メッセージ	
!	メッセージ	変更内容を個別オプションを設定している (ソース・ファイル名) に反映できませんでした。
	原　因	全体オプションを個別オプションへ反映した場合，個別オプションが不正な指定になってしまいます。
	ユーザの処置	個別オプションの指定を確認して再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。

12.8.3 リンカ（LK78K0）

表 12-9 リンカ（LK78K0）用 DLL の表示するエラー・メッセージ

エラー種別	エラー・メッセージ	
×	メッセージ	LK78K0.EXE が，PATH 環境変数で示されるディレクトリに登録されていません。
	原　因	LK78K0.EXE 実行形式が規定のディレクトリにありません。
	ユーザの処置	RA78K0 アセンブラ・パッケージを再インストールしてください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	ワーニング・レベルで指定された値が不正です。 指定可能範囲は　0　≦　W　≦　2　です。
	原　因	ワーニング・レベルとして入力可能範囲外の数値が指定されました。
	ユーザの処置	指定可能範囲の数値を指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	フラッシュスタートアドレスで指定された値が不正です。 指定可能範囲は　0h　≦　ZB　≦　0fffffh　です。
	原　因	フラッシュスタートアドレスとして入力可能範囲外の数値が指定されました。
	ユーザの処置	指定可能範囲の数値を指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	パス，またはファイルが見つかりません。 パス，およびファイル名を確認してください。
	原　因	指定されたパス，またはファイルが見つかりません。
	ユーザの処置	正しいパス，またはファイル名を指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	フォルダが存在しません。 作成しますか？
	原　因	指定されたフォルダが存在しません。
	ユーザの処置	フォルダを作成するか，別のフォルダを選択してください。
	ボタン	“ OK ” …フォルダを作成し，メッセージを閉じます。 “ キャンセル ” …メッセージを閉じます。
!	メッセージ	フォルダの作成に失敗しました。
	原　因	指定されたフォルダを作成することができません。
	ユーザの処置	別のフォルダを指定してください。
	ボタン	“ OK ” …メッセージを閉じます。

表 12-9 リンカ（LK78K0）用 DLL の表示するエラー・メッセージ

エラー種別	エラー・メッセージ	
!	メッセージ	オンチップ・ディバグで指定されたサイズが不正です。 指定可能な範囲は　256 バイト ≦ サイズ ≦ 1024 バイトです。
	原　因	オンチップ・ディバグで指定されたサイズが入力可能範囲外の数値で記述されています。
	ユーザの処置	指定可能範囲の数値で再度試してください。
	ボタン	“OK”... メッセージを閉じます。
!	メッセージ	セキュリティ ID で指定した ID が不正です。 ID は 20 桁で指定してください。
	原　因	セキュリティ ID で指定されたセキュリティ ID の指定形式が正しくありません。
	ユーザの処置	指定可能な形式で再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	セキュリティ ID で指定した ID が不正です。 ID は 16 進数で指定してください。
	原　因	セキュリティ ID で指定されたセキュリティ ID の指定形式が正しくありません。
	ユーザの処置	指定可能な形式で再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	セキュリティ ID で指定した ID が不正です。 セキュリティ ID を指定する場合は，ID を入力してください。
	原　因	セキュリティ ID が入力されていません。
	ユーザの処置	セキュリティ ID を入力してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	ライブラリ・ファイルで指定したファイルの指定数が不正です。 ファイルは 10 個まで指定可能です。
	原　因	ライブラリ・ファイルが入力可能範囲外の数で記述されています。
	ユーザの処置	指定可能範囲の数で再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	（ライブラリ・ファイル） ライブラリ・ファイルで指定したファイル名の長さが不正です。
	原　因	ライブラリ・ファイルで指定したファイル名が入力可能範囲外の文字数で記述されています。
	ユーザの処置	指定可能範囲の文字数で再度試してください。
	ボタン	“ OK ” …メッセージを閉じます。

表 12-9 リンカ（LK78K0）用 DLL の表示するエラー・メッセージ

エラー種別	エラー・メッセージ	
!	メッセージ	(ライブラリ・ファイル) ライブラリ・ファイルで指定したファイルが重複しています。
	原　因	ライブラリ・ファイルが重複して記述されています。
	ユーザの処置	重複しているファイルを削除して再度試してください。
	ボタン	" OK " …メッセージを閉じます。
!	メッセージ	ライブラリ・ファイル読み込みパスで指定したパスの指定数が不正です。 パスは 64 個まで指定可能です。
	原　因	ライブラリ・ファイル読み込みパスが入力可能範囲外の数で記述されています。
	ユーザの処置	指定可能範囲の数で再度試してください。
	ボタン	" OK " …メッセージを閉じます。
!	メッセージ	(ライブラリ・ファイル読み込みパス) ライブラリ・ファイル読み込みパスで指定したパスの長さが不正です。
	原　因	ライブラリ・ファイル読み込みパスが入力可能範囲外の文字数で記述されています。
	ユーザの処置	指定可能範囲の文字数で再度試してください。
	ボタン	" OK " …メッセージを閉じます。
!	メッセージ	(ライブラリ・ファイル読み込みパス) ライブラリ・ファイル読み込みパスで指定したパスが重複しています。
	原　因	ライブラリ・ファイル読み込みパスが重複して記述されています。
	ユーザの処置	重複しているパスを削除して再度試してください。
	ボタン	" OK " …メッセージを閉じます。

12.8.4 オブジェクト・コンバータ（OC78K0）

表 12-10 オブジェクト・コンバータ（OC78K0）用 DLL の表示するエラー・メッセージ

エラー種別	エラー・メッセージ	
×	メッセージ	OC78K0.EXE が，PATH 環境変数で示されるディレクトリに登録されていません。
	原　因	OC78K0.EXE 実行形式が規定のディレクトリにありません。
	ユーザの処置	RA78K0 アセンブラ・パッケージを再インストールしてください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	オブジェクト充填の指定形式が不正です。
	原　因	オブジェクトの充填値に指定された形式が不正です。
	ユーザの処置	記述可能な形式で指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	充てん値で指定された値が不正です。 指定可能範囲は　0h　≦　充てん値　≦　0ffh　です。
	原　因	充填値として入力可能範囲外の数値が指定されました。
	ユーザの処置	指定可能範囲の数値を指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	スタートアドレスで指定された値が不正です。 指定可能な範囲は　0h　≦　スタートアドレス　≦　SFR 領域を除くプログラム空間の最大アドレス h　です。
	原　因	スタートアドレスとして入力可能範囲外の数値が指定されました。
	ユーザの処置	指定可能範囲の数値を指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	サイズで指定された値が不正です。 指定可能範囲は　0h　≦　サイズ　≦　（SFR 領域を除くプログラム空間の最大アドレス）-（スタートアドレス）+ 1h　です。
	原　因	サイズとして入力可能範囲外の数値が指定されました。
	ユーザの処置	指定可能範囲の数値を指定してください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	フォルダが存在しません。 作成しますか？
	原　因	指定されたフォルダが存在しません。
	ユーザの処置	フォルダを作成するか，別のフォルダを選択してください。
	ボタン	“ OK ” …フォルダを作成し，メッセージを閉じます。 “ キャンセル ” …メッセージを閉じます。

表 12-10 オブジェクト・コンバータ（OC78K0）用 DLL の表示するエラー・メッセージ

エラー種別	エラー・メッセージ	
!	メッセージ	フォルダの作成に失敗しました。
	原　因	指定されたフォルダを作成することができません。
	ユーザの処置	別のフォルダを指定してください。
	ボタン	“ OK ” …メッセージを閉じます。

12.8.5 ライブラリアン（LB78K0）

表 12-11 ライブラリアン（LB78K0）用 DLL の表示するエラー・メッセージ

エラー種別	エラー・メッセージ	
×	メッセージ	LB78K0.EXE が，PATH 環境変数で示されるディレクトリに登録されていません。
	原　因	LB78K0.EXE 実行形式が規定のディレクトリにありません。
	ユーザの処置	RA78K0 アセンブラ・パッケージを再インストールしてください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	フォルダが存在しません。 作成しますか？
	原　因	指定されたフォルダが存在しません。
	ユーザの処置	フォルダを作成するか，別のフォルダを選択してください。
	ボタン	“ OK ” …フォルダを作成し，メッセージを閉じます。 “ キャンセル ” …メッセージを閉じます。
!	メッセージ	フォルダの作成に失敗しました。
	原　因	指定されたフォルダを作成することができません。
	ユーザの処置	別のフォルダを指定してください。
	ボタン	“ OK ” …メッセージを閉じます。

12.8.6 リスト・コンバータ（LCNV78K0）

表 12-12 リスト・コンバータ（LCNV78K0）用 DLL の表示するエラー・メッセージ

エラー種別	エラー・メッセージ	
×	メッセージ	LCNV78K0.EXE が，PATH 環境変数で示されるディレクトリに登録されていません。
	原　因	LCNV78K0.EXE 実行形式が規定のディレクトリにありません。
	ユーザの処置	RA78K0 アセンブラ・パッケージを再インストールしてください。
	ボタン	“ OK ” …メッセージを閉じます。
!	メッセージ	フォルダが存在しません。 作成しますか？
	原　因	指定されたフォルダが存在しません。
	ユーザの処置	フォルダを作成するか，別のフォルダを選択してください。
	ボタン	“ OK ” …フォルダを作成し，メッセージを閉じます。 “ キャンセル ” …メッセージを閉じます。
!	メッセージ	フォルダの作成に失敗しました。
	原　因	指定されたフォルダを作成することができません。
	ユーザの処置	別のフォルダを指定してください。
	ボタン	“ OK ” …メッセージを閉じます。

付録 A　サンプル・プログラム

この章では，RA78K0 **に添付されているサンプル・プログラムのリストを紹介します。**

A.1 k0main.asm

```
  NAME          SAMPM
; ***********************************************
;
;     HEX -> ASCII Conversion Program
;
;           main-routine
;
; ***********************************************

PUBLIC          MAIN , START
EXTRN           CONVAH
EXTRN           _@STBEG

DATA            DSEG    saddr
HDTSA :         DS  1
STASC :         DS  2

CODE            CSEG    AT 0H
MAIN :          DW      START

                CSEG
START :

        ; chip initialize
        MOVW    SP , #_@STBEG

        MOV     HDTSA , #1AH
        MOVW    HL , #HDTSA     ; set hex 2-code data in HL registor

        CALL    !CONVAH         ; convert ASCII <- HEX
                                ; output BC-register <- ASCII code
        MOVW    DE , #STASC     ; set DE <- store ASCII code table
        MOV     A , B
        MOV     [ DE ] , A
        INCW    DE
        MOV     A , C
        MOV     [ DE ] , A

        BR      $$

        END
```

A.2 k0sub.asm

```
   NAME          SAMPS
;*************************************************************
;
;        HEX -> ASCII Conversion Program
;              sub-routine
;
;  input condition : ( HL )          <- hex 2 code
;  output condition : BC-register    <- ASCII 2 code
;
;*************************************************************

PUBLIC          CONVAH

          CSEG
CONVAH :
          XOR     A , A
          ROL4    [ HL ]            ; hex upper code load
          CALL    !SASC
          MOV     B , A             ; store result

          XOR     A , A
          ROL4    [ HL ]            ; hex lower code load
          CALL    !SASC
          MOV     C , A             ; store result

          RET

;*****************************************************************
;     subroutine   convert ASCII code
;
;  input   Acc ( lower 4bits ) <- hex code
;  output  Acc                 <- ASCII code
;*****************************************************************

SASC :
          CMP     A , #0AH          ; check hex code > 9
          BC      $SASC1
          ADD     A , #07H          ; bias ( +7H )
SASC1 :
          ADD     A , #30H          ; bias ( +30H )
          RET

          END
```

A.3 test1.s

```
EXTRN           SEARCH , STABLE
EXTRN           _@STBEG
PUBLIC          MAIN , START
; ********************************************
;     Searching data
; ********************************************
SDATA :
        DB      04 , 12H , 34H , 56H , 78H
;
;
CODE    CSEG    AT 0H
START : DW      MAIN

MAIN :
        MOVW    SP , #_@STBEG
        DE = #STABLE
        HL = #SDATA
        CALL    !SEARCH
        if_bit ( !CY )
SLI :
                repeat
                until ( forever )
        else
SERR :
                repeat
                until ( forever )
        endif
        END
```

A.4 test2.s

```
#include          " testinc.s "

PUBLIC SEARCH

        CSEG
; ************************************************************
; *  Data search                                             *
; *     input     HL      search data address                *
; *               DE      table top address                  *
; *     output    CY = 1  not find                           *
; *               CY = 0  find ( DE <- table address )       *
; ************************************************************

SEARCH :
        while ( [ DE ] !=  #0 ) ( A )
                BC = #0
                A = [ DE ]
                C = A
                PUSH    HL
                PUSH    DE
                while ( [ DE ] == [ HL ] ) ( A )
                        DE++
                        HL++
                        if ( C == #0 ) ( A )

                                POP     DE
                                POP     HL
                                CLR1    CY
                                RET
                        endif
                        C--
                endw
                POP      DE
                POP      HL
                A = [ DE ]
                E += A
                A = B
                ADDC    D , A
        endw
        SET1    CY
        RET
        END
```

A.5 testinc.s

```
PUBLIC          STABLE

; ************************************
;               Data table
; ************************************

        CSEG
STABLE :
        DB      03 , 12H , 34H, 78H
        DB      04 , 55H, 66H, 77H, 88H
        DB      05 , 12H, 34H, 56H, 78H, 10H
        DB      03 , 12H, 34H, 56H
        DB      04 , 12H, 34H, 0AH, 78H
        DB      04 , 12H, 34H, 56H, 70H
        DB      04 , 12H, 34H, 56H, 78H
        DB      01 , 0ABH
        DB      02 , 34H , 78H
        DB      00
```

A.6 st.bat

```
        echo off
        cls
        set         LEVEL = 0

        if " %1 " == "" goto ERR_BAT

        st78k0 -C%1 test1.s
        ra78k0 test1.asm
        if errorlevel 1 set LEVEL = 1
        st78k0 -C%1 test2.s
        ra78k0 test2.asm
        if errorlevel 1 set LEVEL = 1
        if %LEVEL% == 1 echo Assemble error !!
        if %LEVEL% == 1 goto END

        cls
        lk78k0 test1.rel test2.rel -s -otest.lmf -ptest.map
        if errorlevel 1 echo Link error !!
        if errorlevel 1 goto END

        cls
        oc78k0 test
        if errorlevel 1 echo Object conversion error !!
        if errorlevel 1 goto END

        cls
        set LEVEL = 0
        lcnv78k0 -ltest.lmf -rtest1.rel test1.prn
        if errorlevel 1 set LEVEL = 1
        lcnv78k0 -ltest.lmf -rtest2.rel test2.prn
        if errorlevel 1 set LEVEL = 1
        if %LEVEL% == 1 echo List conversion error !!
        if %LEVEL% == 1 goto END

        cls
        echo No error.
        goto END

: ERR_BAT

        echo Usage : st.bat chiptype

: END
        echo on
```

付録 B　使用上の注意

この章では，RA78K0 を使用するときの注意事項を示します。

(1) デバイス・ファイル

RA78K0 を実行するには，デバイス・ファイルが必要です。デバイス・ファイルは RA78K0 パッケージには含まれておりません。別途ご入手ください。

デバイス・ファイルは NEC エレクトロニクス　マイクロコンピュータ　ホームページよりダウンロードしてください。

http://www.necel.com/micro/ods/jpn/tool/DeviceFile/list.html　→　開発ツールダウンロード（ODS）

(2) オブジェクト・コンバータ

オブジェクト・コンバータは，-R（オブジェクトのアドレス・ソート），および -U（充てん値指定）オプションを指定してご使用ください。

本オプションは，ディフォールトでは指定されています。

オブジェクトがアドレス・ソートされていない場合，ROM コード発注（アクロス処理，テープ・アウトと呼ばれている作業です）を行うとエラーになりますので，-R は必ず指定してください（指定の解除をしないでください）。

(3) メモリ初期化疑似命令

メモリ初期化疑似命令 DW，DB をデータ・セグメント（DSEG）で記述した場合，オブジェクト・コードは出力されますが，オブジェクト・コンバータでワーニング W4301 になります。これは，ROM 領域（コード領域）以外のアドレスにコードが存在するためにワーニングになっています。

この状態で ROM コード発注（アクロス処理，テープ・アウトと呼ばれている作業です）を行うとエラーになります。

(4) オブジェクト・コンバータの操作仕様

オブジェクト・コンバータ・オプション -U で，スタートを指定した場合，スタート・アドレス，またはコードが配置されたアドレスの小さい方のアドレスから充てんを開始します。SFR 領域（FF00H-FFFFH）には充てんを行いません。

記述形式：-U 充てん値 [, [スタート] , サイズ]

[　] 内は省略可能です。

(5) メモリ・ディレクティブ

各デバイスのディフォールトのメモリ領域名は消去できません。

ご使用にならないディフォールトのメモリ領域名のサイズは 0 にしてください。

ただし，セグメントによってはディフォールトの領域に割り付けられるものもありますので，領域名を変更する際にはご注意ください。

なお，ディフォールトのメモリ領域名については，各デバイスのユーザーズ・マニュアルを参照してください。

(6) ディバグ・オプション

C コンパイラ／構造化アセンブラ・プリプロセッサでディバグ情報を出力して，コンパイル／構造化アセンブルした場合，その出力アセンブル・ソースをアセンブルするときには，ディバグ情報を出力しないようにしてください（-NGA オプションを指定してください）。ディバグ情報を出力すると，C コンパイラ／構造化アセンブラ・ソース・レベルでディバグできないことがあります。

(7) CC78K0 関連

CC78K0 が出力したアセンブラ・ソースをアセンブルして使用し，C ソース・レベル・ディバグを行う場合，何点かの注意事項があります。

詳細は，C コンパイラ・パッケージの製品添付文書（使用上の留意点）を参照してください。

(8) ID78K0/ID78K0-NS/ID78K0-QB，SM78K0

ID78K0/ID78K0-NS/ID78K0-QB，SM78K0 でディバグを行う場合にシンボル数，ソース行数の制限に関して，ID78K0/ID78K0-NS/ID78K0-QB，SM78K0 の制限以内でご使用ください。

詳細は，ディバッガ／シミュレータの製品添付文書（使用上の留意点）を参照してください。

(9) セグメント名

セグメント名を記述する場合，ソース・ファイル名のプライマリ名と同名のセグメント名を記述しないでください。アセンブル時にアボート・エラー F2106 になります。

(10) 構造化アセンブラ・ソース中でのマクロ定義

構造化アセンブラ・プリプロセッサのソース中にマクロ定義を記述する場合は，構造化アセンブリ言語ではなく，アセンブリ言語で記述してください。ラベルの二重定義エラーとなる場合があります。

(11) SFR 名の EQU 定義

EQU 疑似命令のオペランドには SFR 名を指定できますが，saddr 領域外の SFR の名前を PUBLIC に指定した場合，アセンブル・エラーとなります。

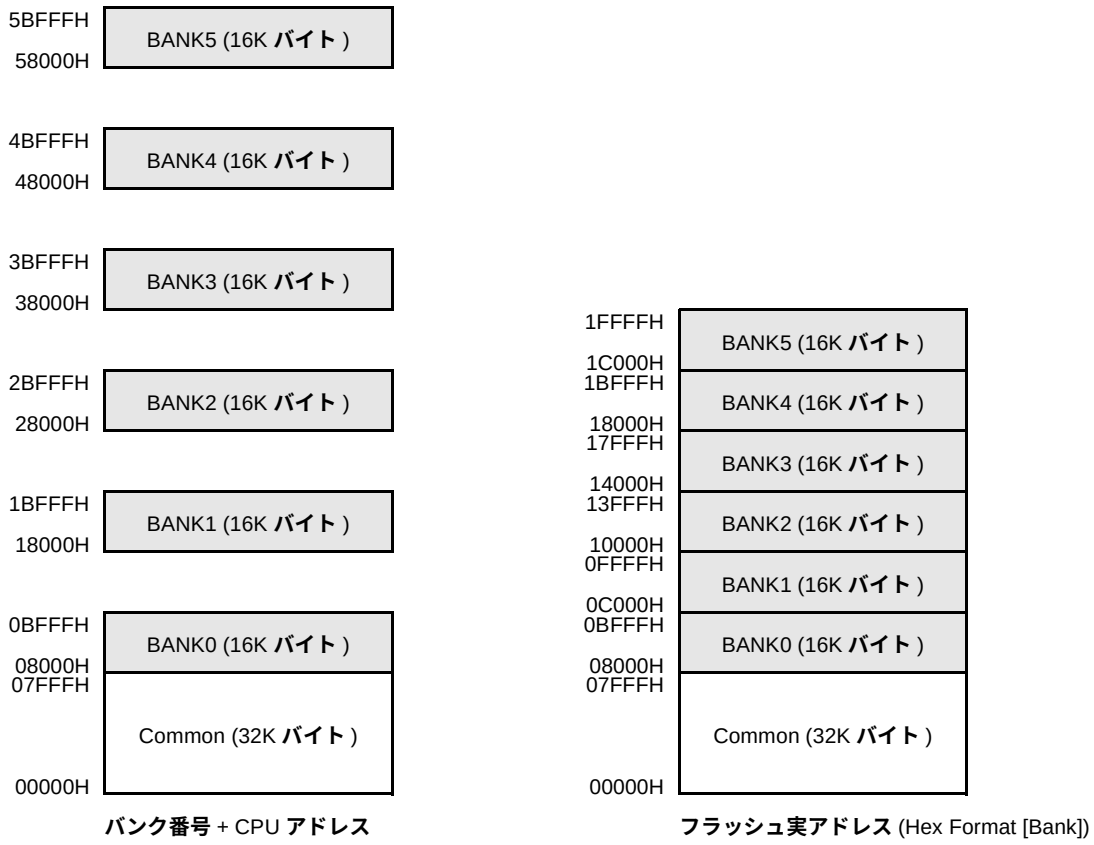
(12) ネットワーク使用時

一時ファイルを作成するディレクトリをネットワーク上で共有されているファイル・システムに置くと，ファイルの競合が生じて異常動作を起こす場合があります。オプションや環境変数の設定によって，このような競合を避けてください。

(13) バンク対応製品における HEX 出力方法

バンク対応製品では，アドレスの見え方として「バンク番号 + CPU アドレス」，「フラッシュ実アドレス (Hex Format [Bank])」の 2 種類があります。

図 B-1 アドレスの見え方



アセンブラでのアドレス参照はバンク番号 + CPU アドレスで行っていますので，ユーザはバンク番号 + CPU アドレスを意識していることになります。

ただし，フラッシュ・メモリへのセルフ，またはオンボード・プログラミングにおいてはフラッシュ実アドレスで書き込みをしなければなりません。そこで，オブジェクト・コンバータが HEX 形式オブジェクト・モジュール・ファイルをフラッシュ実アドレスで出力することにより，セルフ・プログラミング内，およびライタでの変換処理 (バンク番号 + CPU アドレス→フラッシュ実アドレス) が不要になります。

OC78K0 V3.80 以降は，フラッシュ実アドレスによる HEX 出力に対応しています。バンク対応製品の場合，ディフォールトで「インテル拡張 HEX 形式」，「フラッシュ実アドレス」で出力されますが，-k オプション指定により他の出力形式が可能となっています。64K バイトを超える製品の場合，インテル標準形式で出力しても動作しませんので，必ずインテル拡張形式，あるいはモトローラ S タイプの 24 ビット・スタンダード・アドレス，32 ビット・アドレスで出力指定するようにしてください。

```
          Assemble list

ALNO    STNO    ADRS    OBJECT    M   I   SOURCE          STATEMENT
   1       1
   2       2    -----                      main_c          CSEG    AT 100H
   3       3
   4       4    00100   13F301                             MOV     BANK, #BANKNUM lab_bk1
   5       5    00103   R9A0080                            CALL    !lab_bk1
   7       7
   8       8    -----                                      CSEG    BANK    ; 18000->0C000H
   9       9    18000                      label_bk1 :
  10      10    18000   00                                 NOP
  11      11    18001   00                                 NOP
  12      12    18002   00                                 NOP
  13      13    18003   AF                                 RET
  14      14
  15      15
  16      16                               END
```

上記のプログラムでは，lab_bk1 **が** 18000H **番地に配置されています。**

表 B-1 **インテル拡張** HEX **形式 (フラッシュ実アドレス) の出力例**

```
:1000F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF10
:020000020000FC
:0601000013F3019A0080D8
:020000020000FC
:10010600FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9
:10011600FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE9
:10012600FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD9

   :
   :
   :

:10BFF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF51
:020000020000FC
:04C000000000000AF8D
:020000020000FC
:10C00400FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3C
:10C01400FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2C
```

フラッシュ実アドレスの場合，18000H **番地に見えるコードは** 0C000H **番地に書かれていますが，**10 **バイト目以降のコード部分は変わりません。コード部分はそのままで，アドレス値を詰めた形になっています (それに伴い各行最終バイトのチェック・サム値も変更になります)。このことに関し，ユーザは特別な配慮をする必要はありません。**

表 B-2 インテル拡張 HEX 形式 (バンク番号 + CPU アドレス) の出力例

```
:1000F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF10
:020000020000FC
:0601000013F3019A0080D8
:020000020000FC
:10010600FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9
:10011600FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE9
:10012600FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD9

   :
   :
   :

:10BFF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF51
:0200000210000EC
:0480000000000AFCD
:020000021000EC
:10800400FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7C
:10801400FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6C
```

18000H **番地のコードは，そのまま** 18000H **番地に書かれています。**

付録 C　コマンド・オプション

この章では，プログラムのオプションを表形式でまとめました。
プログラム開発の際にお役立てください。
このオプション一覧は索引としても使用できます。

C.1　構造化アセンブラ・オプション

表 C-1 構造化アセンブラ・オプション

分類	記述形式	機能	他のオプションとの関係	省略時解釈
デバイス種別指定	-Cデバイス種別	対象デバイスの種別を指定します。	独立	省略不可
ワード・シンボル文字指定	-SC 文字	ワード・シンボル名の最後の文字を指定します。	独立	-SCP
シンボル定義指定	-D シンボル名 [= 数値] [, シンボル名 [= 数値] …]（複数指定可）	#IFDEF 疑似命令などに与えるシンボルを指定します。	独立	なし
タブ数指定	-WT数値,数値,数値	変換した命令を出力する位置を指定します。	独立	-WT2 , 3 , 4
インクルード・ファイル読み込みパス指定	-I パス名 [， パス名] …（複数指定可）	インクルード・ファイルを指定したパスから読み込みます。	独立	ソース・ファイルのあるパス 環境変数 ‘ INC78K0 ’ 指定パス
2 次ソース・ファイル指定	-O [ファイル名]	2 次ソース・ファイル名を指定します。	独立	-O 入力ファイル名 .asm
エラー・リスト・ファイル指定	-E [ファイル名]	エラー・リスト・ファイルを出力します。	独立	-E 入力ファイル名 .est
パラメータ・ファイル指定	-F ファイル名	入力ファイル名，オプションを指定したファイルから入力します。	独立	コマンド行上からのみオプション，入力ファイル名の入力が可能

表 C-1 構造化アセンブラ・オプション

分類	記述形式	機能	他のオプションとの関係	省略時解釈
ディバグ情報出力指定	-GS	構造化アセンブラ・ソース・レベルのディバグ情報の出力を指定します。	-GSと-NGSを同時に指定した場合は，あとで指定した方が有効です。	-GS
	-NGS	-GS オプションを無効にします。		
2 次ソース・ファイル強制出力指定	-J	2 次ソース・ファイルを強制的に出力します。	独立	強制出力しません。
漢字コード指定	-ZS	コメントの漢字をシフト JIS コードとして解釈します。	-ZS と -ZE と -ZNを同時に指定した場合は，あとで指定した方が有効です。	Windows/ HP-UX のとき -ZS SunOS , Solaris のとき -ZE
	-ZE	コメントの漢字を EUC コードとして解釈します。		
	-ZN	コメントの漢字を漢字として解釈しません。		
デバイス・ファイル・サーチ・パス指定	-Y パス名	デバイス・ファイルを指定されたパスから読み込みます。	独立	〈..\dev〉 ST78K0 の起動されたパスに対して
ヘルプ指定	- -	ディスプレイ（コンソール）にヘルプ・メッセージを出力します。	他のオプションをすべて無効にします。	表示しません。

C.2 アセンブラ・オプション

表 C-2 アセンブラ・オプション

分類	記述形式	機能	他のオプションとの関係	省略時解釈
デバイス種別指定	-Cデバイス種別	対象デバイスの種別を指定します。	独立	省略不可
オブジェクト・モジュール・ファイル出力指定	-O［ファイル名］	オブジェクト・モジュール・ファイルを出力します。	-O と -NO を同時に指定した場合は，あとで指定した方が有効です。	-O 入力ファイル名 .rel
	-NO	オブジェクト・モジュール・ファイルを出力しません。		
オブジェクト・モジュール・ファイル強制出力指定	-J	オブジェクト・モジュール・ファイルを強制的に出力します。	-J と -NJ を同時に指定した場合は，あとで指定した方が有効です。	-NJ
	-NJ	-J オプションを無効にします。		
ディバグ情報出力指定	-G	ローカル・シンボル情報をオブジェクト・モジュール・ファイルへ出力します。	-G と -NG を同時に指定した場合は，あとで指定した方が有効です。	-G
	-NG	-G オプションを無効にします。		
	-GA	ソース・ディバグ情報をオブジェクト・モジュール・ファイルへ出力します。	-GAと-NGAを同時に指定した場合は，あとで指定した方が有効です。	-GA
	-NGA	-GA オプションを無効にします。		
インクルード・ファイル読み込みパス指定	-I パス名［，パス名］…（複数指定可）	インクルード・ファイルを指定したパスから読み込みます。	独立	ソースファイルのあるパス 環境変数 ‘INC78K0’ 指定パス
アセンブル・リスト・ファイル出力指定	-P［ファイル名］	アセンブル・リスト・ファイル出力します。	-P と -NP を同時に指定した場合は，あとで指定した方が有効です。	-P 入力ファイル名 .rel
	-NP	アセンブル・リスト・ファイル出力しません。		

表 C-2 アセンブラ・オプション

分類	記述形式	機能	他のオプションとの関係	省略時解釈
アセンブル・リスト・ファイル情報指定	-KA	アセンブル・リスト・ファイル中にアセンブル・リストを出力します。	-KSと-KXを同時に指定された場合は，-KS を無視します。 -KA と -NKA , -KS と -NKS , -KX と -NKX を同時に指定した場合は，あとで指定した方が有効です。 -NKA , -NKS , -NKX を同時に指定された場合は，-P を無視します。	-KA
	-NKA	アセンブル・リスト・ファイルを出力しません。		
	-KS	アセンブル・リスト・ファイル中にシンボル・リストを出力します。		-NKS
	-NKS	-KS オプションを無効にします。		
	-KX	アセンブル・リスト・ファイル中にクロレファレンス・リストを出力します。		-NKX
	-NKX	-KX オプションを無効にします。		
アセンブル・リスト・ファイル形式指定	-LW [文字数]	アセンブル・リスト・ファイルの 1 行に印字する文字数を変更します。	-NP を指定した場合は，-LW が無視されます。	-LW132
	-LL [行数]	アセンブル・リスト・ファイルの 1 頁に印字する行数を変更します。	-NP を指定した場合は，-LL が無視されます。	-LL66
	-LH 文字列	アセンブル・リスト・ファイルのヘッダに，指定された文字列を出力します。	-NP を指定した場合は，-LHが無視されます。	なし
	-LT [文字数]	タブの展開文字数を変更します。	-NP を指定した場合は，-LT が無視されます。	-LT8
	-LF	アセンブル・リスト・ファイルの最後に改行コードを付加します。	-LF と -NLF を同時に指定した場合は，あとで指定した方が有効です。 -NP を指定した場合は，-LF が無視されます。	-NLF
	-NLF	-LF オプションを無効にします。		
エラー・リスト・ファイル出力指定	-E [ファイル名]	エラー・リスト・ファイルを出力します。	-E と -NE を同時に指定した場合は，あとで指定した方が有効です。	-NE
	-NE	-E オプションを無効にします。		

表 C-2 アセンブラ・オプション

分類	記述形式	機能	他のオプションとの関係	省略時解釈
パラメータ・ファイル指定	-F ファイル名	入力ファイル名，オプションを指定したファイルから入力します。	独立	コマンド行上からのみオプション，入力ファイル名の入力が可能
テンポラリ・ファイル作成パス指定	-T パス名	テンポラリ・ファイルを，指定したパスに作成します。	独立	環境変数，‘ TMP ’ 指定パス
漢字コード指定	-ZS	コメントの漢字をシフト JIS コードとして解釈します。	-ZS と -ZE と -ZNを同時に指定した場合は，あとで指定した方が有効です。	Windows / HP-UX のとき -ZS SunOSのとき -ZE
	-ZE	コメントの漢字を EUC コードとして解釈します。		
	-ZN	コメントの漢字を漢字として解釈しません。		
デバイス・ファイル・サーチパス指定	-Y パス名	デバイス・ファイルを指定されたパスから読み込みます。	独立	〈..\dev〉 RA78K0 の起動されたパスに対して
シンボル定義指定	-D シンボル名 [=数値] [，シンボル名 [= 数値 …] （複数指定可）	シンボルの定義を行います。	独立	なし
シリーズ共通オブジェクト指定	-COMMON	78K0 シリーズ共通オブジェクト・モジュール・ファイルの出力を指定します。	独立	なし
セルフ・プログラミング指定	-SELF	セルフ・プログラミングを使用する際に指定します。	独立	なし
ヘルプ指定	--	ディスプレイ（コンソール）にヘルプ・メッセージを出力します。	他のオプションをすべて無効にします。	表示しません。

C.3 リンカ・オプション

表 C-3 リンカ・オプション

分類	記述形式	機能	他のオプションとの関係	省略時解釈
ロード・モジュール・ファイル出力指定	-O [ファイル名]	ロード・モジュール・ファイルを出力します。	-O と -NO を同時に指定した場合は，あとで指定した方が有効です。	-O 入力ファイル名 .lmf
	-NO	ロード・モジュール・ファイルを出力しません。		
ロード・モジュール・ファイル強制出力指定	-J	ロード・モジュール・ファイルを強制的に出力します。	-J と -NJ を同時に指定した場合は，あとで指定した方が有効です。	-NJ
	-NJ	-J オプションを無効にします。		
ディバグ情報出力指定	-G	ディバグ情報をロード・モジュール・ファイルへ出力します。	-G と -NG を同時に指定した場合は，あとで指定した方が有効です。-NG が指定された場合は，-KP , -KL の指定にかかわらず，ローカル・シンボル・リスト，パブリック・シンボル・リストは出力されません。	-G
	-NG	-G オプションを無効にします。		
スタック解決用シンボル生成指定	-S [領域名]	スタック解決用のパブリック・シンボルを自動生成します。	-S と -NS を同時に指定した場合は，あとで指定した方が有効です。	-NS
	-NS	-S オプションを無効にします。		
ディレクティブ・ファイル指定	-D ファイル名	特定のファイルをディレクティブ・ファイルとして指定します。	独立	なし
リンク・リスト・ファイル出力指定	-P [ファイル名]	リンク・リスト・ファイルの出力を指定します。	-P と NP を同時に指定した場合は，あとで指定した方が有効です。	-P ファイル名 .map
	-NP	-P オプションを無効にします。		

表 C-3 リンカ・オプション

分類	記述形式	機能	他のオプションとの関係	省略時解釈
リンク・リスト・ファイル情報指定	-KM	リンク・リスト・ファイル中に，マップ・リストを出力します。	-KM と -NKM を同時に指定した場合は，あとで指定した方が有効です。	-KM
	-NKM	-KM オプションを無効にします。	-NKM , -NKP , -NKLをすべて指定した場合は，-P は無効となります。	
	-KD	リンク・リスト・ファイル中に，リンク・ディレクティブ・ファイルを出力します。	-NKM を指定した場合，-KD は無効となります。 -KDと-NKD , -KP と -NKP , -KL と -NKL を同時に指定した場合は，あとで指定した方が有効です。 -NG が指定された場合は，-KP , -KL の指定に関わらず，ローカル・シンボル・リスト，パブリック・シンボル・リストは出力されません。	-KD
	-NKD	-KD オプションを無効にします。		
	-KP	リンク・リスト・ファイル中に，パブリック・シンボル・リストを出力します。		-NKP
	-NKP	-KP オプションを無効にします。		
	-KL	リンク・リスト・ファイル中に，ローカル・シンボル・リストを出力します。		-NKL
	-NKL	-KL オプションを無効にします。		
リンク・リスト・ファイル形式指定	-LL [行数]	リスト 1 頁に印字する行数を指定します。	-NP を指定した場合は，-LL は無視されます。	-LL66
	-LF	リスト・ファイルの最後に，改行コードを付加します。	-LF と -NLF を同時に指定した場合は，あとで指定した方が有効です。 -NP を指定した場合は，後で指定した方が有効です。	-NLF
	-NLF	-LF オプションを無効にします。		
エラー・リスト・ファイル出力指定	-E [ファイル名]	エラー・リスト・ファイルを出力します。	-E と -NE を同時に指定した場合は，あとで指定方が有効です。	-NE
	-NE	-E オプションを無効にします。		

表 C-3 リンカ・オプション

分類	記述形式	機能	他のオプションとの関係	省略時解釈
ライブラリ・ファイル指定	-B ファイル名	特定のファイルを，ライブラリ・ファイルとして入力します。	独立	なし
ライブラリ・ファイル読み込みパス指定	-Iパス名[,パス名] …（複数指定可）	ライブラリ・ファイルを指定したパスから読み込みます。	-B オプションで，パス名含まないライブラリ・ファイルを指定した場合は無効となります。	環境変数，‘LIB78K0’ 指定パス
パラメータ・ファイル指定	-F ファイル名	入力ファイル名，オプションを指定したファイルから入力します。	独立	コマンド行上からのみオプション，入力ファイル名の入力が可能
テンポラリ・ファイル作成パス指定	-T パス名	テンポラリ・ファイルを，指定したパスに作成します。	独立	環境変数，‘ TMP ’ 指定パス
デバイス・ファイル・サーチ・パス指定	-Y パス名	デバイス・ファイルを指定されたパスから読み込みます。	独立	〈..\dev〉 LK78K0 の起動されたパスに対して
ワーニング・メッセージ出力指定	-W [レベル]	ワーニング・メッセージをコンソールへ出力するか否かを指定します。	独立	通常のエラーメッセージを出力
フラッシュ ROM 内蔵製品のブート領域 ROM プログラムのリンク指定	-ZB アドレス	フラッシュ ROM 領域の先頭アドレスを指定します。	独立	なし
オンチップ・ディバグのプログラム・サイズ指定	-GO[サイズ]	オンチップ・ディバグのプログラム・サイズを指定します。	独立	なし
セキュリティ ID 指定	-GI セキュリティ ID	セキュリティ ID を指定します。	独立	なし
ヘルプ指定	--	ディスプレイ（コンソール）にヘルプ・メッセージを出力します。	他のオプションをすべて無効にします。	表示しません。

C.4 オブジェクト・コンバータ・オプション

表 C-4 オブジェクト・コンバータ・オプション

分類	記述形式	機能	他のオプションとの関係	省略時解釈
HEX 形式オブジェクト・モジュール・ファイル出力指定	-O [ファイル名]	HEX 形式オブジェクト・モジュール・ファイルを出力します。	-O と -NO を同時に指定した場合は，あとで指定した方が有効です。	-O 入力ファイル名 .hex (拡張空間に対するファイルタイプ H1 ～ H15)
	-NO	HEX 形式オブジェクト・モジュール・ファイルを出力しません。		
シンボル・テーブル・ファイル出力指定	-S [ファイル名]	シンボル・テーブル・ファイルを出力します。	-S と -NS を同時に指定した場合は，あとで指定した方が有効です。	-S 入力ファイル名 .hex (拡張空間に対するファイルタイプ H1 ～ H15)
	-NS	シンボル・テーブル・ファイルを出力しません。		
オブジェクト・アドレス順ソート指定	-R	HEX 形式オブジェクトをアドレス順にソートします。	-R と -NR を同時に指定した場合は，あとで指定した方が有効です。-NO を指定した場合は，-R は無視されます。	-R
	-NR	-R オプションを無効にします。		
オブジェクト充てん値指定	-U充てん値 [, [スタート] , サイズ]	HEX 形式オブジェクトが出力されない領域に対して，指定した充てん値をオブジェクト・コードとして出力します。	-U と -NU を同時に指定した場合は，あとで指定した方が有効です。-NO を指定した場合は，-U は無視されます。	-U0FFH
	-NU			
エラー・リスト・ファイル出力指定	-E [ファイル名]	エラー・リスト・ファイルを出力します。	-E と -NE を同時に指定した場合は，あとで指定した方が有効です。	-NE
	-NE	-N オプションを無効にします。		
パラメータ・ファイル指定	-F ファイル名	入力ファイル名，オプションを指定したファイルから入力します。	独立	コマンド行上からのみオプション，入力ファイル名の入力が可能

表 C-4 オブジェクト・コンバータ・オプション

分類	記述形式	機能	他のオプションとの関係	省略時解釈
デバイス・ファイル・サーチ・パス指定	-Y パス名	デバイス・ファイルを指定されたパスから読み込みます。	独立	〈..\dev〉OC78K0 の起動されたパスに対して
フラッシュ ROM 内蔵製品用ファイル分割出力指定	-ZF	フラッシュ ROM 内蔵製品のブート領域 ROM プログラムのリンク指定時において，ブート領域とそれ以外の領域を別の HEX フォーマット・ファイルに分割出力するオプションを追加します。	独立	なし
ヘルプ指定	--	ディスプレイ（コンソール）にヘルプ・メッセージを出力します。	他のオプションをすべて無効にします。	表示しません。

C.5 ライブラリアン・オプション

表 C-5 ライブラリアン・オプション

分類	記述形式	機能	他のオプションとの関係	省略時解釈
リスト・ファイル形式指定	-LW [文字数]	リスト・ファイルの 1 行に印字する文字数を変更します。	LIST（サブコマンド）を指定しない場合は無効です。-LF と -NLF を同時に指定した場合は，あとで指定した方が有効です。	-LW132
	-LL [行数]	リスト・ファイルの 1 頁の行数を変更します。		-LL66
	-LF	リスト・ファイルの最後に，改頁コードを付加します。		-NLF
	-NFL	-LF オプションを無効にします。		
テンポラリ・ファイル作成パス指定	-T パス名	テンポラリ・ファイルを，指定したパスに作成します。	独立	環境変数，'TMP' 指定パス
デバイス・ファイル・サーチ・パス指定	-Y パス名	デバイス・ファイルを指定されたパスから読み込みます。	独立	〈..\dev〉LB78K0 の起動されたパスに対して
ヘルプ指定	--	ディスプレイ（コンソール）にヘルプ・メッセージを出力します。	他のオプションをすべて無効にします。	表示しません。

C.6 リスト・コンバータ・オプション

表 C-6 リスト・コンバータ・オプション

分類	記述形式	機能	他のオプションとの関係	省略時解釈
オブジェクト・モジュール・ファイル入力指定	-R [ファイル名]	オブジェクト・モジュール・ファイルを入力します。	独立	-R アセンブル・リスト・ファイル名 .rel
ロード・モジュール・ファイル入力指定	-L [ファイル名]	ロード・モジュール・ファイルを入力します。	独立	-L アセンブル・リスト・ファイル名 .lnk
アブソリュート・アセンブル・リスト・ファイル出力指定	-O [ファイル名]	アブソリュート・アセンブル・リスト・ファイルを出力します。	独立	-O アセンブル・リスト・ファイル名 .p
エラー・リスト・ファイル出力指定	-E [ファイル名]	エラー・リスト・ファイルを出力します。	-E と -NE を同時に指定した場合は，あとで指定した方が有効です。	-NE
	-NE	-N オプションを無効にします。		
パラメータ・ファイル指定	-F ファイル名	入力ファイル名，オプションを指定したファイルから入力します。	独立	コマンド行上からのみオプション，入力ファイル名の入力が可能
ヘルプ指定	--	ディスプレイ（コンソール）にヘルプ・メッセージを出力します。	他のオプションをすべて無効にします。	表示しません。

付録 D　サブコマンド

この章では，サブコマンドを，一覧表にまとめて示します。

プログラム開発の際にお役立てください。

このサブコマンド一覧は，索引としても使用できます。

表 D-1 サブコマンド一覧

分類	記述形式	機能	短縮形
CREATE	CREATE △ライブラリ・ファイル名 [△トランザクション]	ライブラリ・ファイルを新規に作成します。	C
ADD	ADD　△ライブラリ・ファイル名△トランザクション	ライブラリ・ファイルにモジュールを追加します。	A
DELETE	DELETE △ライブラリ・ファイル名△ (△モジュール名 [△ , …] △)	ライブラリ・ファイル内のモジュールを削除します。	D
REPLACE	REPLACE △ライブラリ・ファイル名△トランザクション	ライブラリ・ファイル内のモジュールを他のモジュールと置き換えます。	R
PICK	PICK △ライブラリ・ファイル名△ (△モジュール名 [△ , …] △)	ライブラリ・ファイル内のモジュールを取り出します。	P
LIST	LIST [△オプション] ライブラリ・ファイル名 [△ (△モジュール名 [△ , …] △)]	ライブラリ・ファイル内のモジュール情報を出力します。	L
HELP	HELP	ディスプレイ（コンソール）にヘルプ・メッセージを出力します。	H
EXIT	EXIT	ライブラリアンを終了します。	E

付録 E　総合索引

Symbols
-- （LB78K0）… 254
-- （LCNV78K0）… 288
-- （LK78K0）… 191
-- （OC78K0）… 235
-- （RA78K0）… 138
-- （ST78K0）… 87

A
ADD … 258
.asm … 68, 94
AT … 149

B
-B （LK78K0）… 181

C
-C （RA78K0）… 105
-C （ST78K0）… 75
-COMMON （RA78K0）… 136
COMPLETE … 149
CREATE … 257

D
-D （LK78K0）… 167
-D （RA78K0）… 135
-D （ST78K0）… 77
DELETE … 259
.dr （LK78K0）… 146

E
-E （LCNV78K0）… 285
-E （LK78K0）… 180
-E （OC78K0）… 229
-E （RA78K0）… 128
-E （ST78K0）… 81
.elk … 146
.elv … 273
.eoc … 200
.era … 94
.est … 68
EXIT … 266

F
-F （LCNV78K0）… 286
-F （LK78K0）… 183
-F （OC78K0）… 230
-F （RA78K0）… 130
-F （ST78K0）… 82

G
-G （LK78K0）… 164
-G （RA78K0）… 108
-GA （RA78K0）… 110
-GI （LK78K0）… 189
-GO （LK78K0）… 188
-GS （ST78K0）… 83

H
HELP … 265
.hex … 200

I
-I （LK78K0）… 182
-I （RA78K0）… 111
-I （ST78K0）… 79
INC78K0 … 314

J
-J （LK78K0）… 163
-J （RA78K0）… 107
-J （ST78K0）… 84

K
-KA （RA78K0）… 113
-KD （LK78K0）… 171
-KI （OC78K0）… 232
-KIE （OC78K0）… 232
-KL （LK78K0）… 175
-KM （LK78K0）… 169
-KM （OC78K0）… 232
-KME （OC78K0）… 232
-KP （LK78K0）… 173
-KS （RA78K0）… 115
-KT （OC78K0）… 232
-KX （RA78K0）… 116

L
-L （LCNV78K0）… 283
LANG78K … 314
-LF （LB78K0）… 251
-LF （LK78K0）… 179
-LF （RA78K0）… 127
-LH （RA78K0）… 122
.lib … 146, 240
LIB78K0 … 314
LIST … 263
-LL （LB78K0）… 250
-LL （LK78K0）… 177
-LL （RA78K0）… 120
.lmf … 146, 200, 273
.lst … 240

-LT （RA78K0）… 125
-LW （LB78K0）… 249
-LW （RA78K0）… 118

M
.map … 146
MEMORY … 149
MERGE … 149

N
-NE （LCNV78K0）… 285
-NE （LK78K0）… 180
-NE （OC78K0）… 229
-NE （RA78K0）… 128
-NG （LK78K0）… 164
-NG （RA78K0）… 108
-NGA （RA78K0）… 110
-NGS （ST78K0）… 83
-NJ （LK78K0）… 163
-NJ （RA78K0）… 107
-NKA （RA78K0）… 113
-NKD （LK78K0）… 171
-NKL （LK78K0）… 175
-NKM （LK78K0）… 169
-NKP （LK78K0）… 173
-NKS （RA78K0）… 115
-NKX （RA78K0）… 116
-NLF （LB78K0）… 251
-NLF （LK78K0）… 179
-NLF （RA78K0）… 127
-NO （LK78K0）… 162
-NO （OC78K0）… 222
-NO （RA78K0）… 106
-NP （LK78K0）… 168
-NP （RA78K0）… 112
-NR （OC78K0）… 226
-NS （LK78K0）… 165
-NS （OC78K0）… 224
-NU （OC78K0）… 227

O
-O （LCNV78K0）… 284
-O （LK78K0）… 162
-O （OC78K0）… 222
-O （RA78K0）… 106
-O （ST78K0）… 80

P
.p … 273
-P （LK78K0）… 168
-P （RA78K0）… 112
PATH … 314
PICK … 262
.plk … 146
.plv … 273
PM plus … 89, 140, 193, 236, 267, 289, 317, 360
.poc … 200
.pra … 94
.prn … 94, 273
.pst … 68

R
-R （LCNV78K0）… 282
-R （OC78K0）… 226
RAM … 148
REGULAR … 152, 153
.rel … 94, 146, 240, 273
REPLACE … 260
ROM … 148

S
-S （LK78K0）… 165
-S （OC78K0）… 224
-SC （ST78K0）… 76
-SELF （RA78K0）… 137
SEQUENT … 149
.sym … 200

T
-T （LB78K0）… 252
-T （LK78K0）… 184
-T （RA78K0）… 132
TMP … 314

U
-U （OC78K0）… 227

W
-W （LK78K0）… 186
-WT （ST78K0）… 78

Y
-Y （LB78K0）… 253
-Y （LK78K0）… 185
-Y （OC78K0）… 233
-Y （RA78K0）… 134
-Y （ST78K0）… 86

Z
-ZB （LK78K0）… 187
-ZE （RA78K0）… 133
-ZE （ST78K0）… 85
-ZF （OC78K0）… 234
-ZN （RA78K0）… 133
-ZN （ST78K0）… 85
-ZS （RA78K0）… 133
-ZS （ST78K0）… 85

【あ行】
アセンブラ … 17, 27
アセンブル・リスト … 275, 296, 316
アブソリュート・アセンブル・リスト … 275, 311
アボート・エラー … 320
インストール … 38
インテル標準 HEX 形式 … 204
エラー・リスト … 293, 301, 308, 309, 311
オブジェクト充てん値指定 … 227
オブジェクト・コンバータ … 29, 200

【か行】
環境変数 … 44, 314
漢字コード … 45
クロスレファレンス・リスト … 299
構造化アセンブラ … 26, 68

【さ行】
最大性能 … 33
サブコマンド … 256
サンプル・プログラム … 374
実行手順 … 46, 50, 56
セグメント配置ディレクティブ … 153

【た行】
ディレクティブ・ファイル … 149

【な行】
内部エラー … 320

【は行】
パブリック・シンボル・リスト … 306
パラメータ・ファイル … 68, 70, 94, 98, 146, 156, 200, 217, 273, 278
フェータル・エラー … 320

【ま行】
マップ・リスト … 304
メモリ空間 … 148
メモリ領域 … 148
メモリ・ディレクティブ … 151

【ら行】
ライブラリアン … 30, 240
リ ン カ … 28
リスト・コンバータ … 31, 273
リンク・ディレクティブ … 149
リンク・リスト・ファイル … 302
ローカル・シンボル・リスト … 307
ロード・モジュール・ファイル … 146, 200, 273

【わ行】
ワーニング・エラー … 320

【発　行】
NECエレクトロニクス株式会社
〒211-8668　神奈川県川崎市中原区下沼部1753
電話（代表）：044(435)5111

お問い合わせ先

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。
URL(アドレス)　http://www.necel.co.jp/

【営業関係，技術関係お問い合わせ先】

半導体ホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）
電　話　：044−435−9494
E-mail　：info@necel.com

【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクスの販売特約店へお申し付けください。