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。

RENESAS

ユーザーズ・マニュアル

RA78K0R Ver.1.00

アセンブラ・パッケージ

操作編

対象デバイス
78K0Rマイクロコントローラ

資料番号　U17836JJ1V0UM00（第1版）
発行年月　June 2006　CP(K)

〔メ　モ〕

目次要約

Windows**および**WindowsXP**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。**

PC/AT**は，米国**IBM Corp.**の商標です。**

- 本資料に記載されている内容は2006年6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，RA78K0R　アセンブラ・パッケージ（以降，RA78K0Rとします）を用いてソフトウエア開発を行われる方に，RA78K0R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RA78K0R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RA78K0R　**アセンブラ・パッケージ　ユーザーズ・マニュアル　言語編**（U17835J）（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RA78K0Rに関する記述は，Ver.1.00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（78K0Rマイクロコントローラ）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RA78K0Rの役割など，RA78K0R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

RA78K0R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　RA78K0R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，RA78K0R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　アセンブラ

第5章　リ ン カ

第6章　オブジェクト・コンバータ

第7章　ライブラリアン

第8章　リスト・コンバータ

第9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10章　RA78K0Rの活用法

RA78K0Rをうまく使うための方法を紹介します。

第11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RA78K0Rを使用する場合は，**第3章**　RA78K0R**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。

【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。
＜＞	；ウインドウ名，ダイアログ名を表します。
“ ”	；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
CC78K0R Ver.1.00　Cコンパイラ	操作編	U17838J	U17838E
	言語編	U17837J	U17837E
RA78K0R Ver.1.00　アセンブラ・パッケージ	操作編	このマニュアル	U17836E
	言語編	U17835J	U17835E
SM+　システム・シミュレータ	操作編	U18010J	U18010E（作成予定）
PM+ Ver.6.20		U17990J	U17990E
ID78K0R-QB Ver.3.20　統合ディバッガ	操作編	U17839J	U17839E

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

図の目次

図番号　タイトル, ページ

表の目次

表番号　タイトル, ページ

第 1 章 概説

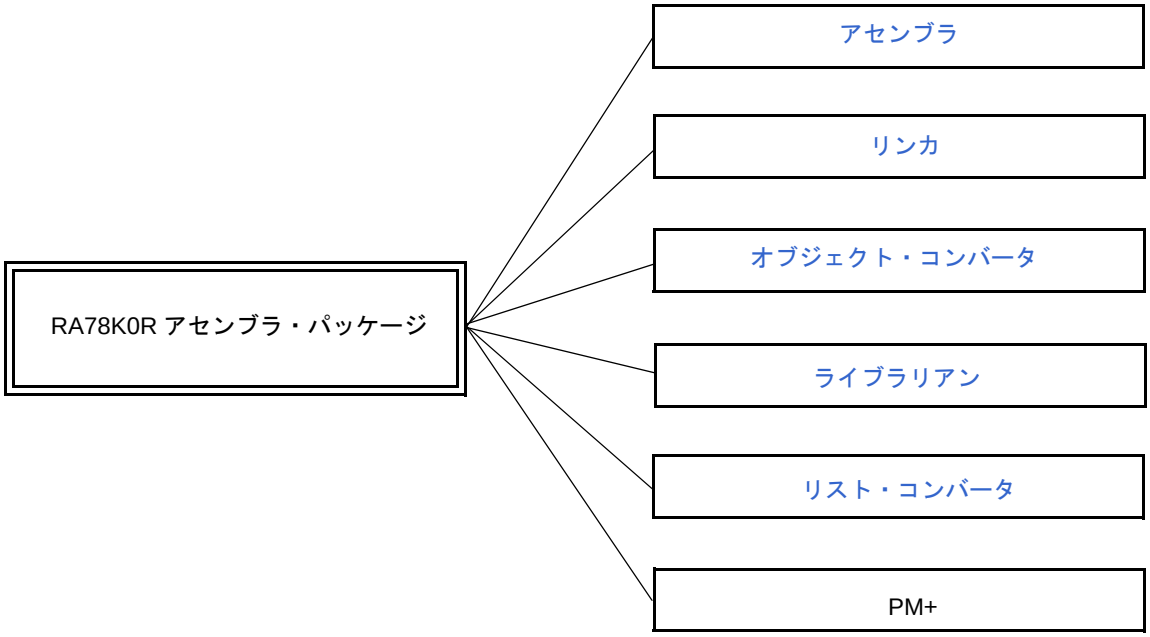
この章では，マイクロコンピュータの開発における RA78K0R の役割など，RA78K0R 全体の機能概要について説明します。

1.1 概要

RA78K0R アセンブラ・パッケージ（以下，RA78K0R）は，78K0R シリーズのアセンブリ言語で記述されたソースを機械語に変換する言語処理プログラムの総称です。

RA78K0R には，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 5 個のプログラムのほかに，エディット，コンパイル / アセンブル，リンクからデバッ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+ が標準添付されています。

図 1-1 RA78K0R アセンブラ・パッケージ



1.1.1 アセンブラ

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行わせる際には，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。

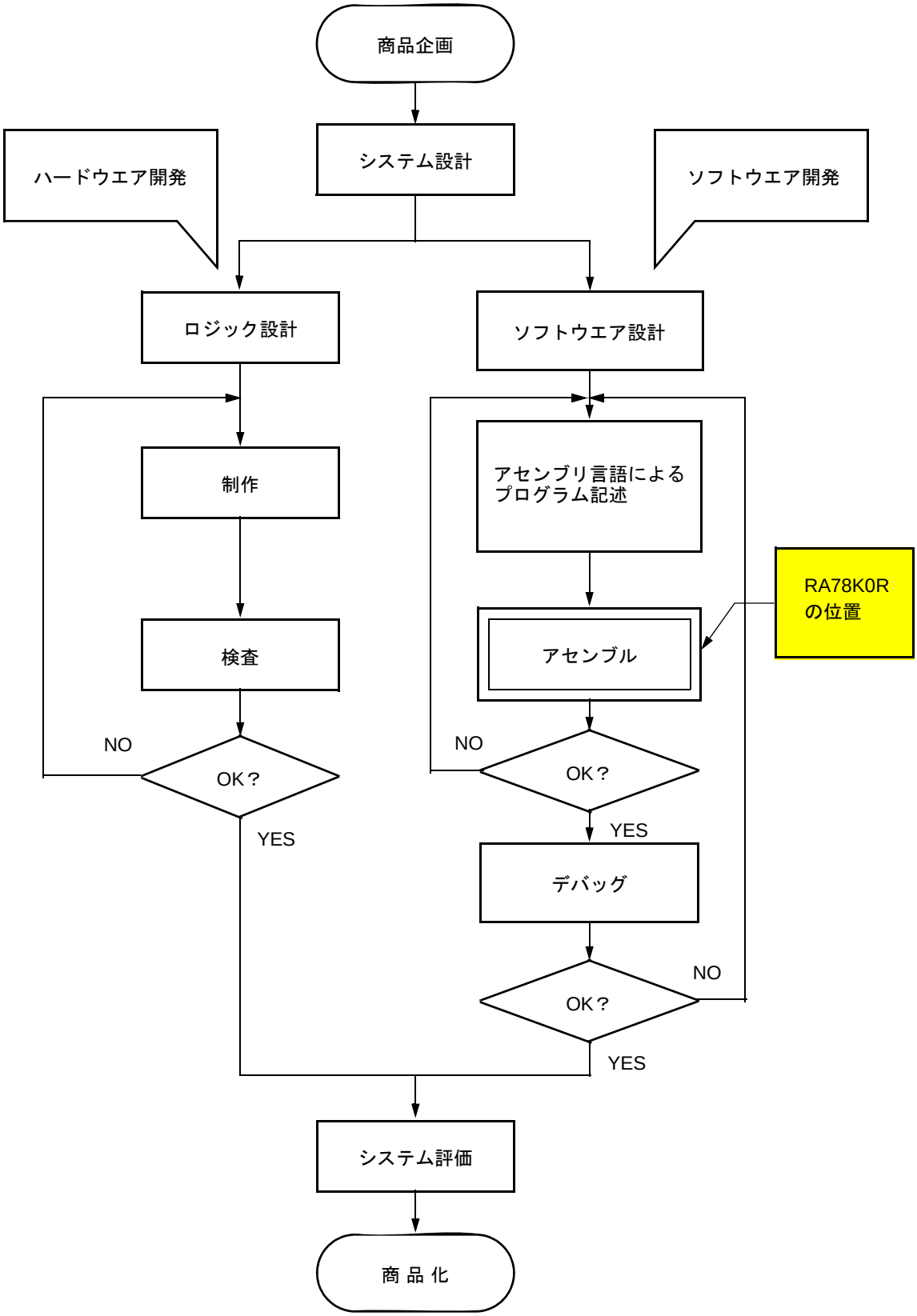
ただし，マイクロコンピュータが扱うことのできるプログラミング言語は機械語に限定されるため，アセンブリ言語で作成したプログラムを機械語に翻訳するプログラムが必要となります。これをアセンブラと呼びます。

図 1-2 アセンブラの流れ

1.1.2 マイクロコンピュータ応用製品の開発と RA78K0R の役割

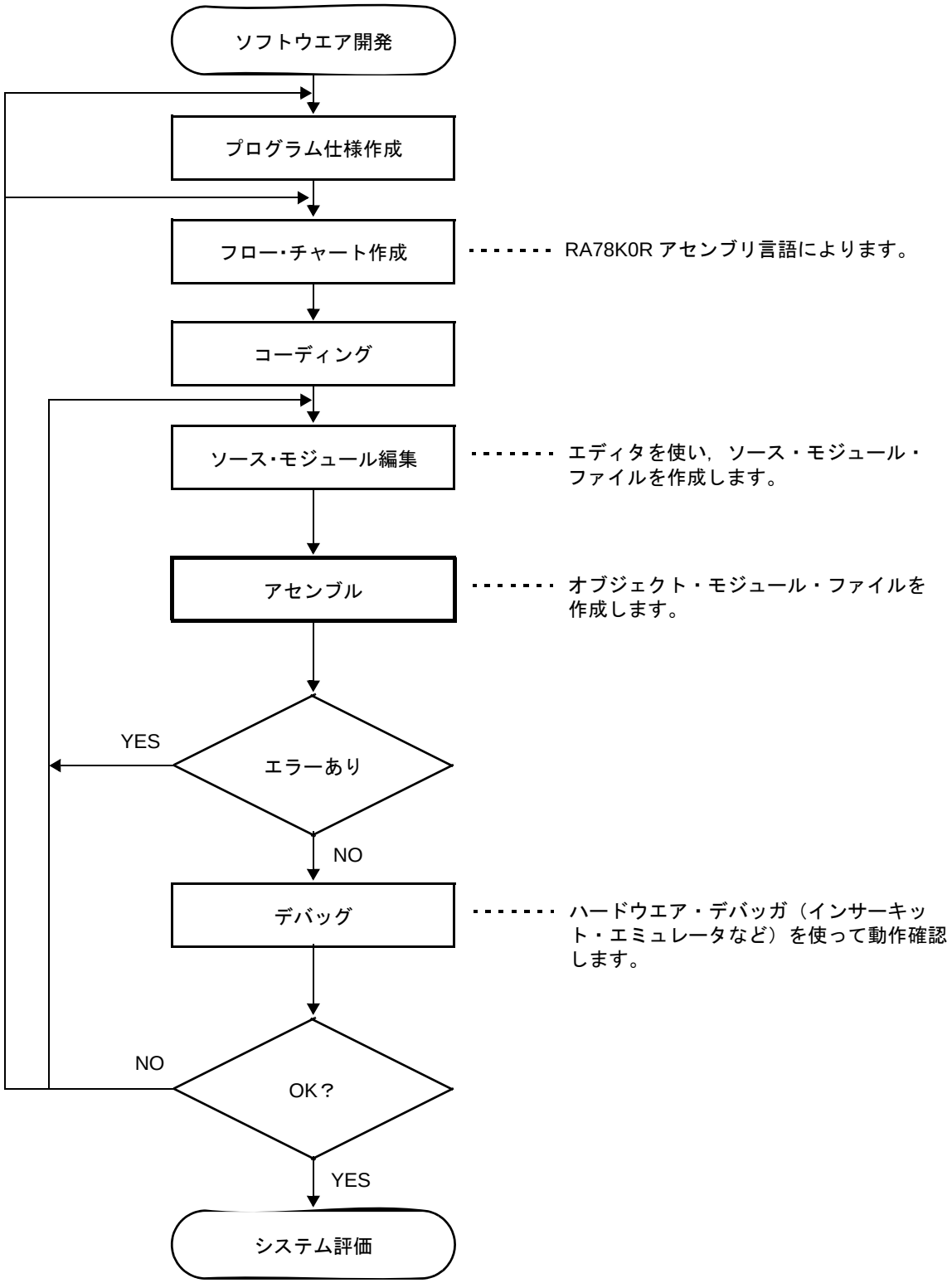
製品開発における“アセンブル”の位置づけを，次に示します。

図 1-3 マイクロコンピュータ応用製品の開発工程



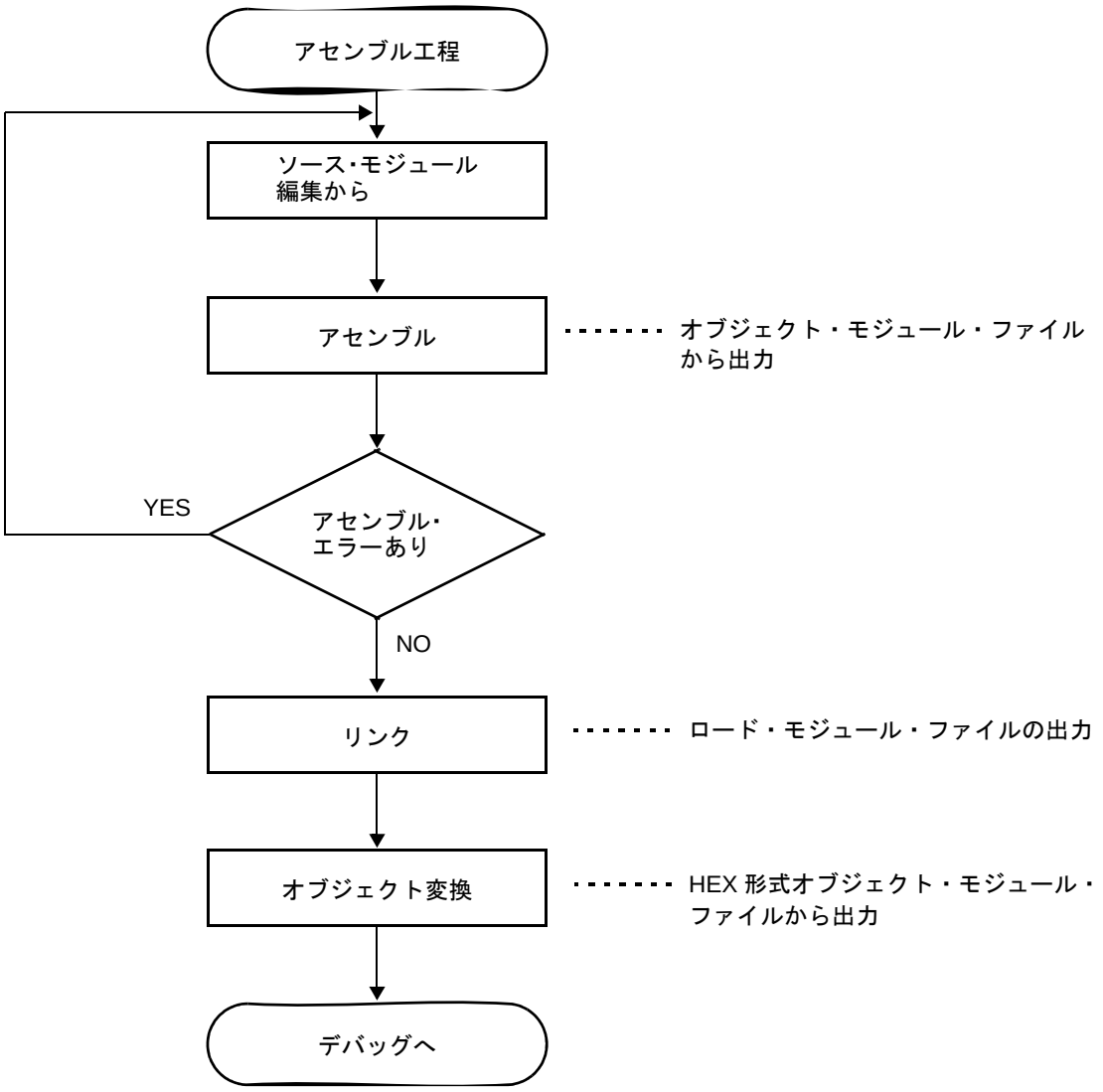
ソフトウエアの開発工程をもう少し詳しく，次に示します。

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，RA78K0R を当てはめてみます。

図 1-5 RA78K0R のアセンブル工程



1.1.3 リロケータブル・アセンブラ

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，RA78K0R では，リロケータブル・アセンブラを採用しています。

- 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，リンカにより，絶対的なアドレスが配置されます。

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でした。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。

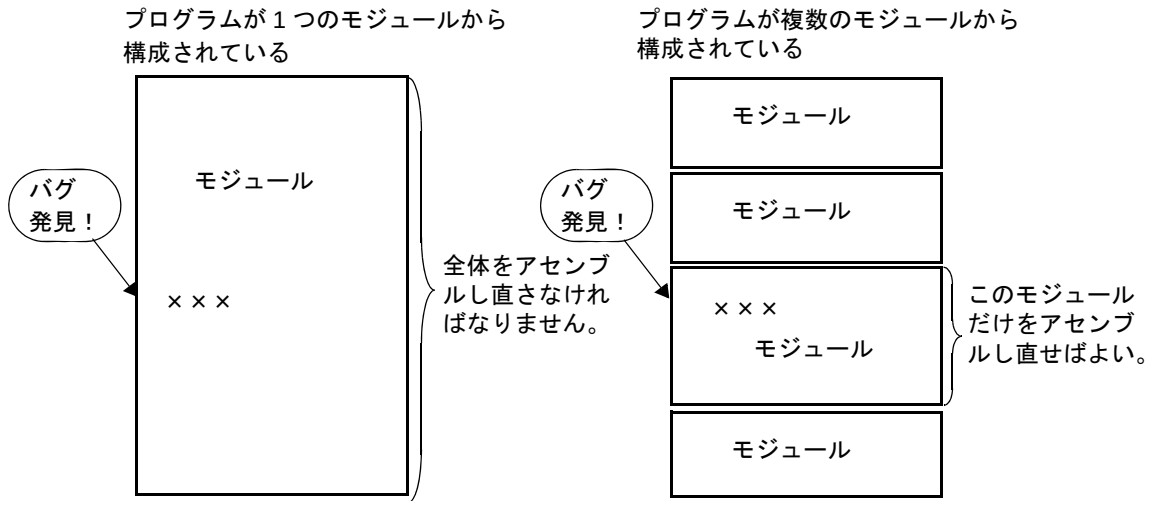
リロケータブル・アセンブラは，モジュラ・プログラミングに適したアセンブラであり，モジュラ・プログラミングを行うことにより，次の利点があげられます。

(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 つの機能単位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，デバッグ時間を短くできます。

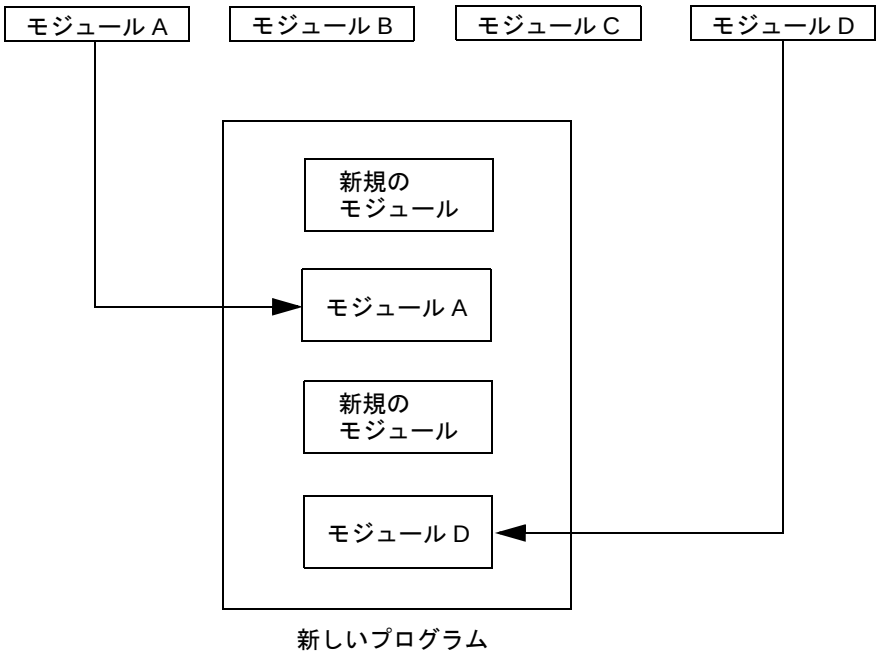
図 1-6 アセンブルのやり直し



(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，ほか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

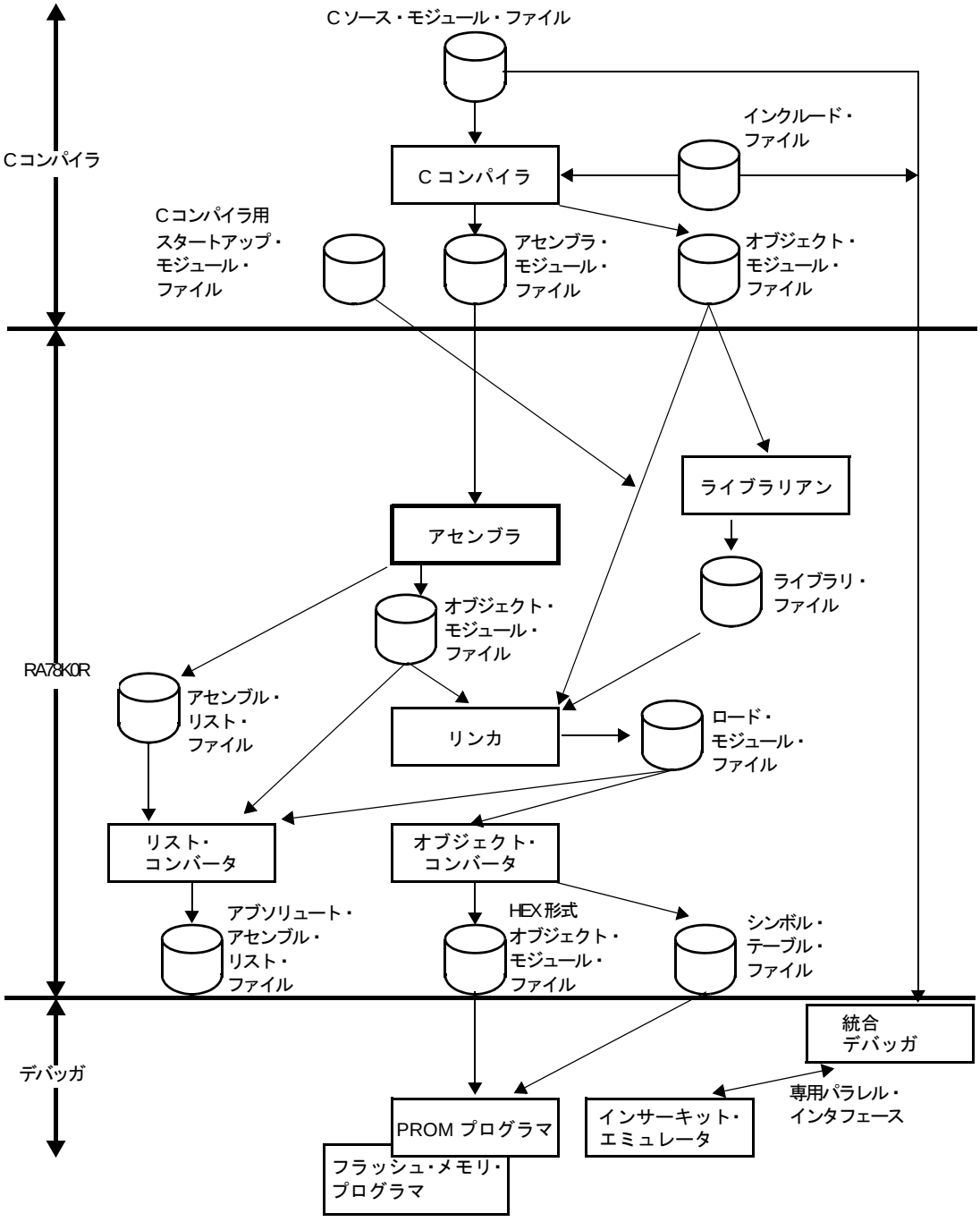


1.2 RA78K0R の機能概要

RA78K0R における一般的なプログラム開発手順を，次に示します。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「RA78K0R」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 RA78K0R によるプログラム開発手順



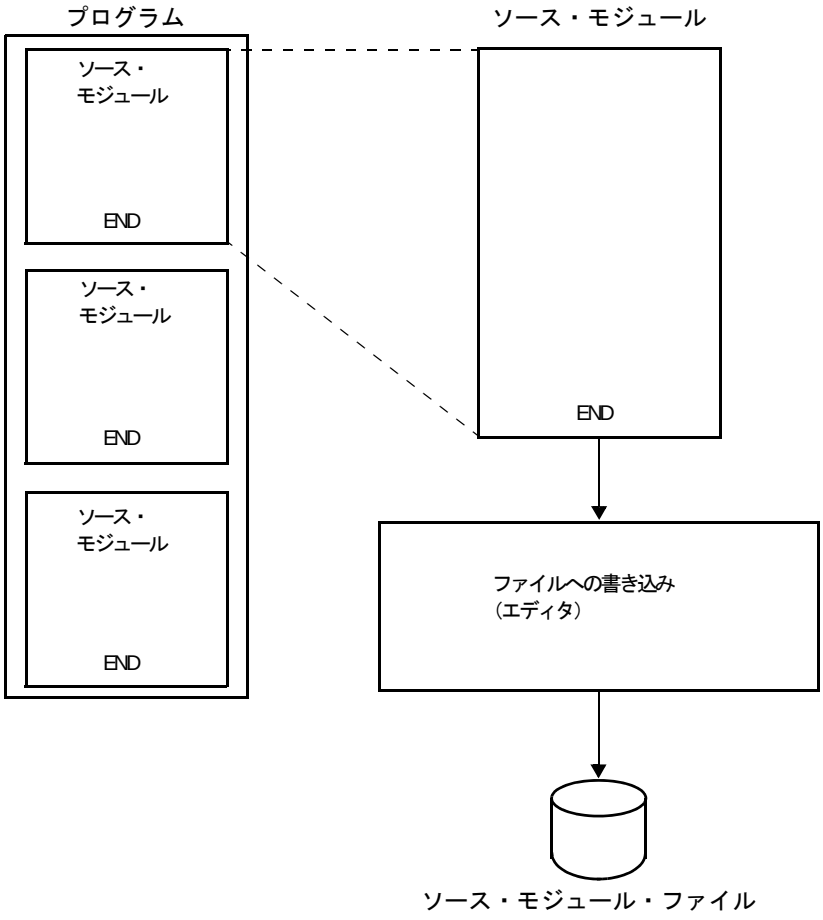
1.2.1 エディタによるソース・モジュール・ファイルの作成

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。

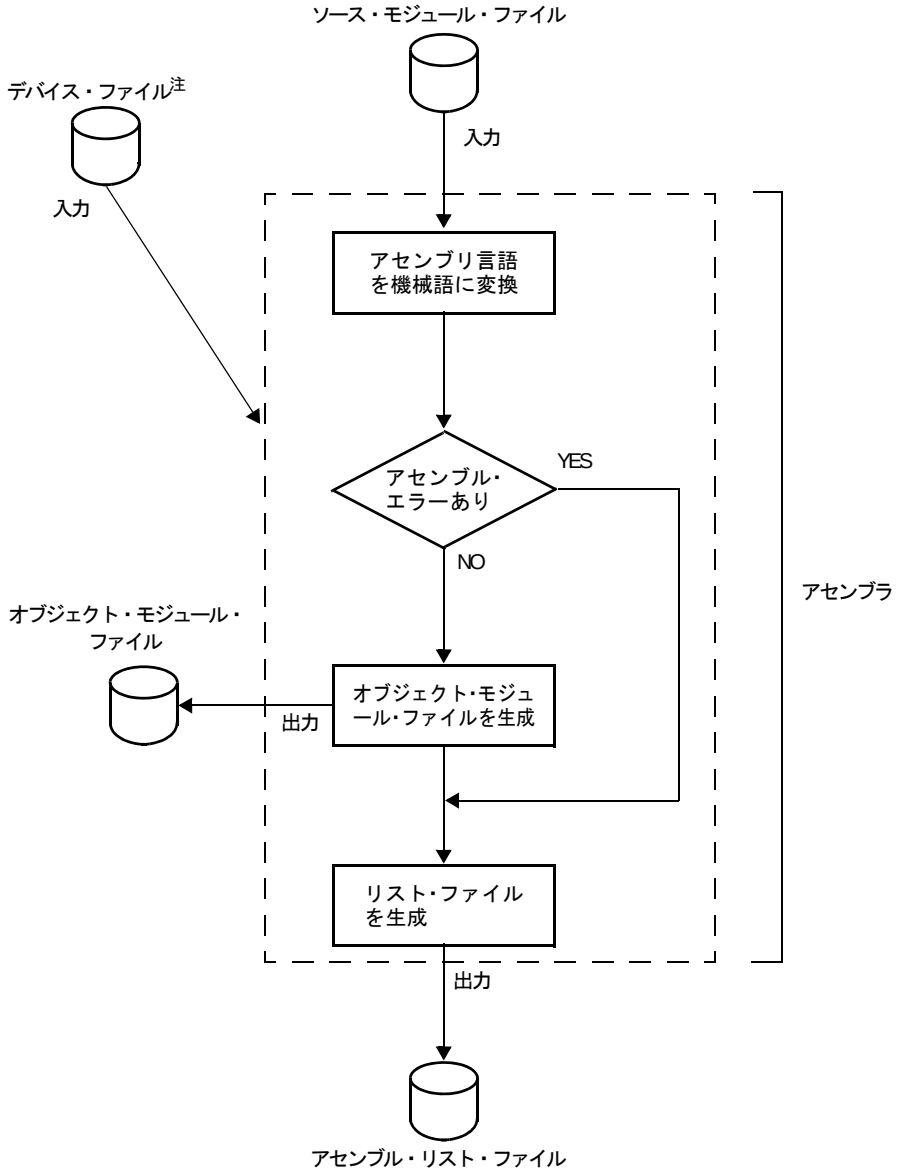
図 1-9 ソース・モジュール・ファイルの作成



1.2.2 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 1-10 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 から取得可能です。

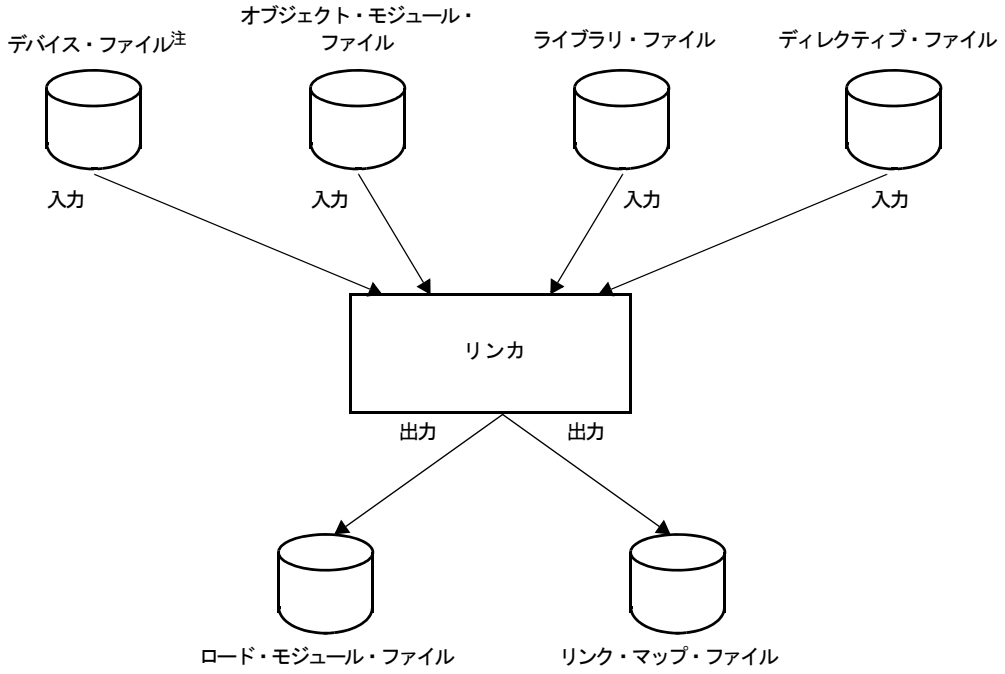
http://www.necel.com/micro/ods/jpn/index.html

1.2.3 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 1-11 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 から取得可能です。

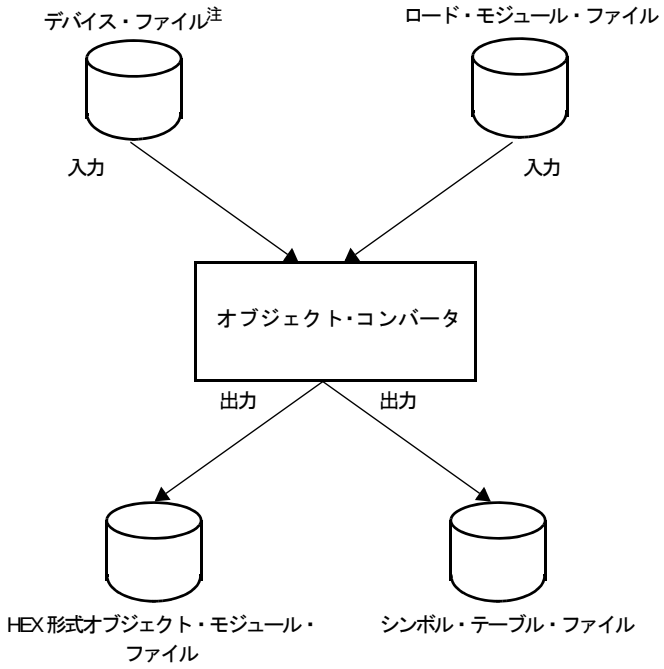
http://www.necel.com/micro/ods/jpn/index.html

1.2.4 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・デバッ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 1-12 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 から取得可能です。

http://www.necel.com/micro/ods/jpn/index.html

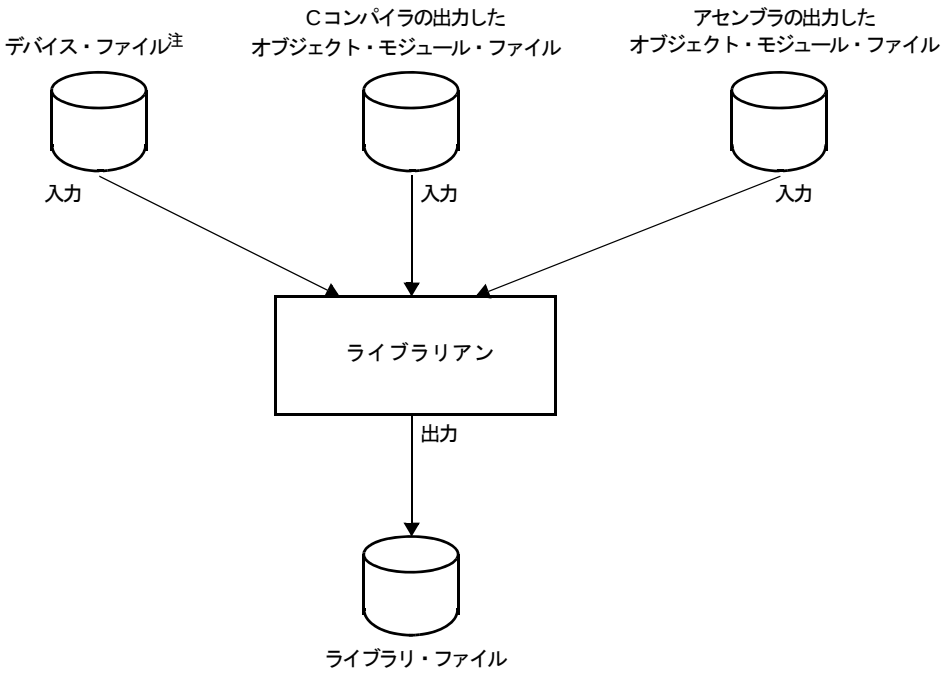
1.2.5 ライブラリアン

汎用性のある，インタフェースの明確なモジュールは，ライブラリ化を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 1-13 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 から取得可能です。

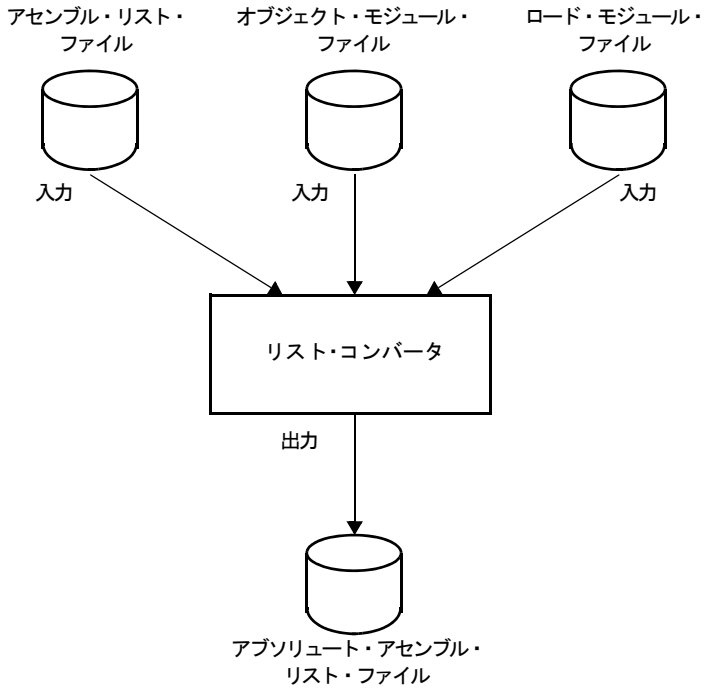
http://www.necel.com/micro/ods/jpn/index.html

1.2.6 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短所があります。しかし，アブソリュート・アセンブル・リストでは，これらの短所が解決されるので，デバッグやプログラムの保守が容易になります。

図 1-14 リスト・コンバータの機能



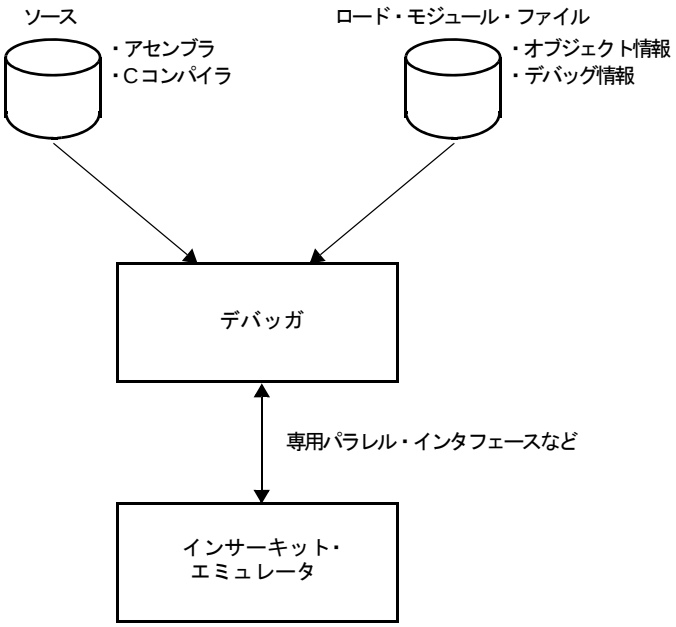
1.2.7 デバッガ

78K0R シリーズ用デバッガは，ソース，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バッ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・ファイルを読み込むことで，ソース・レベルでのデバッグが可能になります。

デバッガ，IE は，別パッケージ（別売）になります。

図 1-15 デバッガの機能



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，次のことを参照してください。

1.3.1 RA78K0R の最大値

(1) アセンブラの最大値

項目	最大値
シンボル数（ローカル + パブリック）	65,535 個
クロスリファレンス・リスト出力可能なシンボル数	65,534 個[注 1]
1 マクロ参照のマクロ・ボディ最大サイズ	1M バイト
全マクロ・ボディ合計のサイズ	10M 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	10,000 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	10,000 個
リロケーション情報[注 2]	65,535 個
行番号情報	65,535 個
1 ファイル中の BR/CALL 疑似命令数	32,767 個
ソース 1 行の文字数	2,048 文字[注 3]
シンボル長	256 文字
スイッチ名の定義数[注 4]	1,000 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	256 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	65,535 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

項目	最大値
-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 文字以上記述した場合，ワーニング・メッセージが出力され，2,049 文字以降は無視されます。

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。

(2) リンカの最大値

項目	最大値
シンボル数（ローカル + パブリック）	65,535 個
同一セグメントの行番号情報	65,535 個
セグメント数	65,535 個[注]
入力モジュール	1,024 個
メモリ領域名の文字長	256 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	64 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 デフォルトで定義されているものを含みます。

1.4 RA78K0R の特徴

RA78K0R 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として，BR 疑似命令，CALL 疑似命令を備えています。

メモリ効率の良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，CALL 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

ソースの一部分を条件によりアセンブルする / しないに設定することができます。

ソース中にデバッグ文などを記述した場合，デバッ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することができます。デバッグ文が必要なくなった場合でも，ソースに大幅な変更を加えることなくアセンブルすることができます。

第 2 章　製品概要とインストール方法

この章では，RA78K0R 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・マシンと提供媒体

この RA78K0R アセンブラ・パッケージは，次に示す開発環境に対応しています。

表 2-1　RA78K0R アセンブラ・パッケージの提供媒体

ホスト・マシン	OS	提供媒体
IBM PC/AT™ 互換機	日本語 Windows（2000/XP）注	CD-ROM

注　アセンブラを Windows 上で使用するためには，PM+ が必要です。
PM+ を使用しない場合は，コマンド・プロンプトで，RA78K0R アセンブラ・パッケージに含まれる各ツールを起動することができます。

2.2　インストール

以下に，RA78K0R 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。

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

   RA78K0R 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 RA78K0R¥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“エクスプローラ”などを用いて，RA78K0R 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 フォルダ構成」を参照してください。

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 から取得可能です。

http://www.necel.com/micro/ods/jpn/index.html

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，RA78K0R とともにインストールされます。

2.4 フォルダ構成

インストール時に表示される標準フォルダは，Windows システムのあるドライブの“Program Files¥NEC Electronics Tools¥RA78K0R¥V*x.xx*¥”です。インストール・フォルダ以下の構成は，次のようになります。ただし，ドライブやインストール・フォルダはインストール時に変更してもかまいません。

PM+ と RA78K0R は同じフォルダにインストールしてください。

なお，このマニュアルでは，セットアップ・プログラムのデフォルトの指示に従って，デフォルトのプログラム・フォルダ名“Program Files¥NEC Electronics Tools”で標準フォルダにインストールしたとして説明しています。

図 2-1 フォルダ構成

Program Files¥NEC Electronics Tools¥

- RA78K0R¥V*x.xx*¥bin¥ … アセンブラ等のプログラムが格納されているフォルダ
- RA78K0R¥V*x.xx*¥doc¥ … ユーザーズ・マニュアル，製品添付文書が格納されているフォルダ
- RA78K0R¥V*x.xx*¥hlp¥ …… オンライン・ヘルプ・ファイルが格納されているフォルダ
- RA78K0R¥V*x.xx*¥inc78k0r¥… インクルード・ファイルが格納されているフォルダ
- RA78K0R¥V*x.xx*¥smp78k0r¥ra78k0r¥ … サンプル・プログラムが格納されているフォルダ

2.5 ファイル構成

各フォルダの内容を，次に示します。

フォルダ構造，およびファイル構成は，インストーラ使用時のものです。

表 2-2 ファイル構成

フォルダ名	ファイル名	概要
RA78K0R¥V*x.xx*¥bin¥	ra78k0r.exe	アセンブラ
	lk78k0r.exe	リンカ
	oc78k0r.exe	オブジェクト・コンバータ
	lb78k0r.exe	ライブラリアン
	lc78k0r.exe	リスト・コンバータ
	lb78k0re.exe	PM+ 環境のライブラリ本体と DLL 間インタフェース
	lb78k0rp.exe	単体起動用ライブラリアン
	ra78k0r.is*	アセンブラが使用するファイル
	*78k0rp.dll	PM+ 用ツール DLL 本体
	*78k0r.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）
RA78K0R¥V*x.xx*¥doc¥	*.pdf	ユーザーズ・マニュアル，製品添付文書
RA78K0R¥V*x.xx*¥hlp¥	78k0rp.chm	オンライン・ヘルプ・ファイル
RA78K0R¥V*x.xx*¥inc78k0r¥	compati.inc	アセンブラ・オプション（-compati）用インクルード・ファイル
RA78K0R¥V*x.xx*¥smp78k0r¥ra78k0r¥	k0rmain.asm	アセンブラ・サンプル・プログラム（メイン・ルーチン）
	k0rsub.asm	アセンブラ・サンプル・プログラム（サブルーチン）
	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.txt	サンプル・プログラム，バッチ・ファイルの説明書（テキスト・ファイル）

備考 * ： 英数字

2.6 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. RA78K0R の削除

   コントロール・パネルの [アプリケーションの追加と削除]，または [プログラムの追加と削除] において，“NEC EL RA78K0R V*x.xx*”を選択してください。

3. ファイル群の確認

   Windows の標準アプリケーション“エクスプローラ”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 フォルダ構成」を参照してください。

2.7 環境設定

2.7.1 ホスト・マシン

次の OS で動作するように設計されています。

- Windows 2000/XP のコマンド・プロンプト

2.7.2 環境変数

次の環境変数を設定してください。

表 2-3 環境変数

環境変数	説明
PATH	アセンブラの実行形式を置いたフォルダを指定します。
TMP	一時ファイルを作成するフォルダを指定します。
INC78K0R	インクルード・ファイルを検索するフォルダを指定します。
LIB78K0R	ライブラリを使用する場合，ライブラリを検索するフォルダを指定します。
LANG78K	コメントに記述した漢字の漢字コードを指定します。

【設定例】

```
PATH=%PATH%;C:¥Program Files¥NEC Electronics Tools¥RA78K0R¥Vx.xx¥bin
set TMP=C:¥tmp
set INC78K0R=C:¥Program Files¥NEC Electronics Tools¥RA78K0R¥Vx.xx¥inc78k0r
set LIB78K0R=C:¥Program Files¥NEC Electronics Tools¥XXXXXX¥Vx.xx¥lib78k0r
set LANG78K=SJIS
```

2.7.3 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，漢字を使用することができます。
- 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（-zs/-ze/-zn）で指定してください。

第 3 章 RA78K0R の実行手順

この章では，RA78K0R 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“k0rmain.asm”，“k0rsub.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 RA78K0R 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，“k0rmain.asm”，“k0rsub.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において，ソース・ファイルとしてアセンブラへの入力になります。

k0rmain.asm　：メイン・モジュール

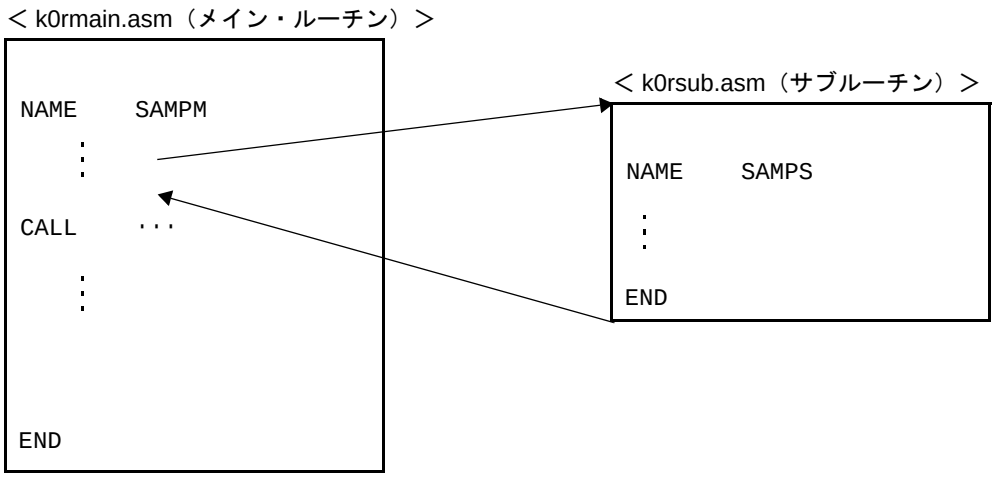
k0rsub.asm　：サブ・モジュール

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 k0rmain.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 k0rsub.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



＜ k0rmain.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
```

＜ k0rmain.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 (a)
                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，RA78K0R の機能や操作を習得する目的で用意され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（FFEF8H - FFEFFH）

備考 3　サンプル・プログラム中の (a) の ROL4 命令は，78K0R シリーズではサポートしていない 78K0 シリーズ用の命令であるため，アセンブラ・オプション（-compati）の指定が必要です。
アセンブラ・オプション（-compati）については，「4.4 アセンブラ・オプション」を参照してください。

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。

k0rmain.asm	：メイン・モジュール
k0rsub.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 RA78K0R の実行手順

RA78K0R 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“k0rmain.asm”，“k0rsub.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“uPD78F1166_A0”として説明します。

＜ ra.bat（RA78K0R 動作確認用バッチ・プログラム）＞

```
        echo off

        cls
        set LEVEL = 0

        if " %1 " == "" goto ERR_BAT

        ra78k0r -c%1 k0rmain.asm
        if errorlevel 1 set LEVEL = 1
        ra78k0r -c%1 k0rsub.asm
        if errorlevel 1 set LEVEL = 1
        if %LEVEL% == 1 echo Assemble error !!
        if %LEVEL% == 1 goto END

        cls
        lk78k0r k0rmain.rel k0rsub.rel -s -orasample.lmf -prasample.map
        if errorlevel 1 echo Link error !!
        if errorlevel 1 goto END

        cls
        oc78k0r rasample
        if errorlevel 1 echo Object conversion error !!
        if errorlevel 1 goto END
        cls
        set LEVEL = 0
        lc78k0r -lrasample.lmf -rk0rmain.rel k0rmain.prn
        if errorlevel 1 set LEVEL = 1
        lc78k0r -lrasample.lmf -rk0rsub.rel k0rsub.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。

   RA78K0R 動作確認用バッチ・プログラムを，ターゲット・デバイスの品種指定名を指定して実行します。

   C>ra.bat f1166a0

   次のメッセージがディスプレイに出力されます。

```
78K0R Series Assembler Vx.xx  [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD78f1166a0
 Device file : Vx.xx

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

78K0R Series Assembler Vx.xx  [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD78f1166a0
 Device file : Vx.xx

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

   画面をクリアする

```
78K0R Series Linker Vx.xx  [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx

 Target chip : uPD78f1166a0
 Device file : Vx.xx

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

   画面をクリアする

```
78K0R Series Object Converter Vx.xx  [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx

 Target chip : uPD78f1166a0
 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 RA78K0R Vx.xx  [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx

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

List Conversion Program for RA78K0R Vx.xx  [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation 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。

k0rmain.rel	: オブジェクト・モジュール・ファイル
k0rmain.prn	: アセンブル・リスト・ファイル
k0rsub.rel	: オブジェクト・モジュール・ファイル
k0rsub.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
k0rmain.p	: アブソリュート・アセンブル・リスト・ファイル
k0rsub.p	: アブソリュート・アセンブル・リスト・ファイル

RA78K0R の実行手順をまとめると次のようになります。

図 3-2 RA78K0R の実行手順 1

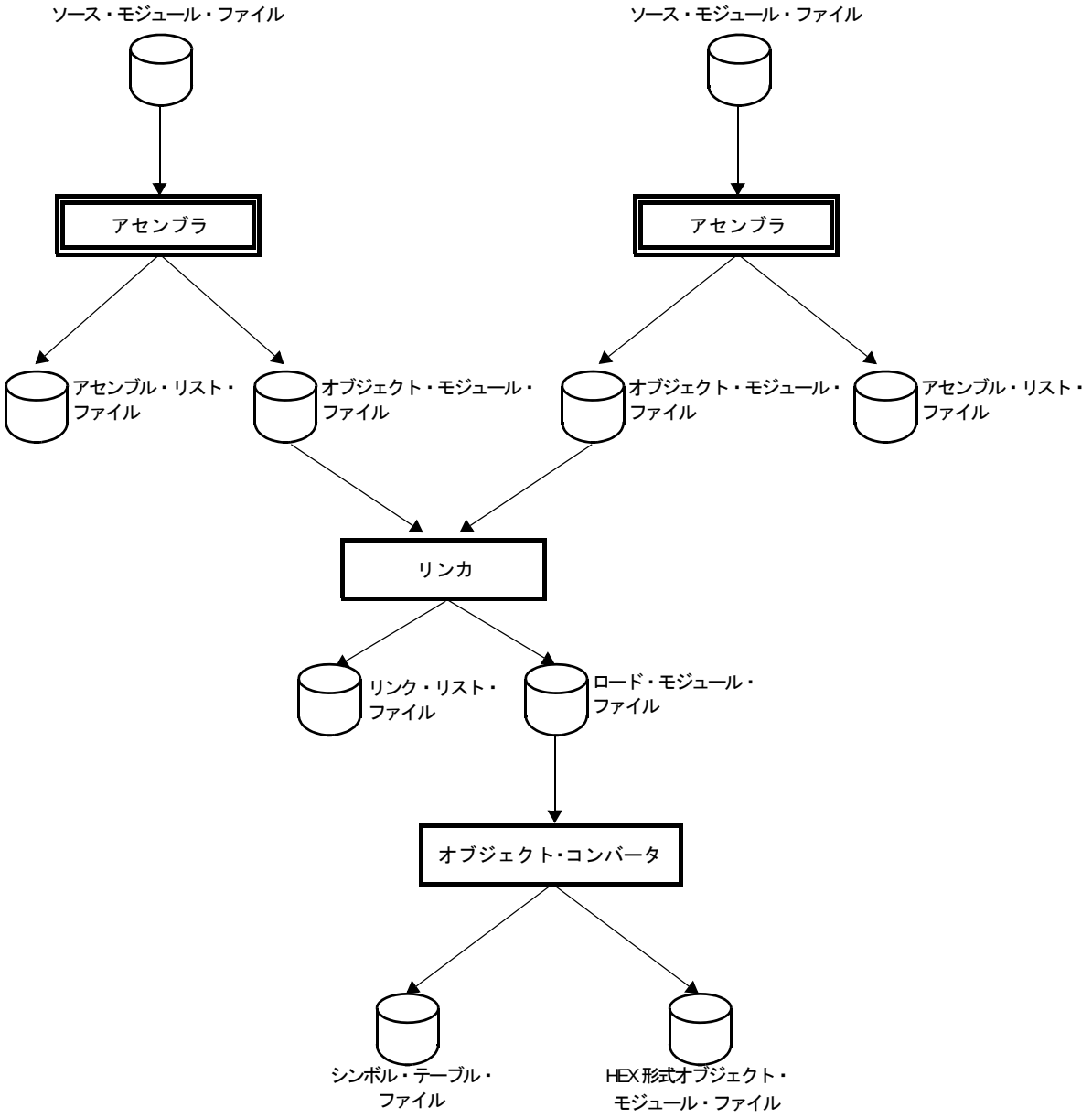
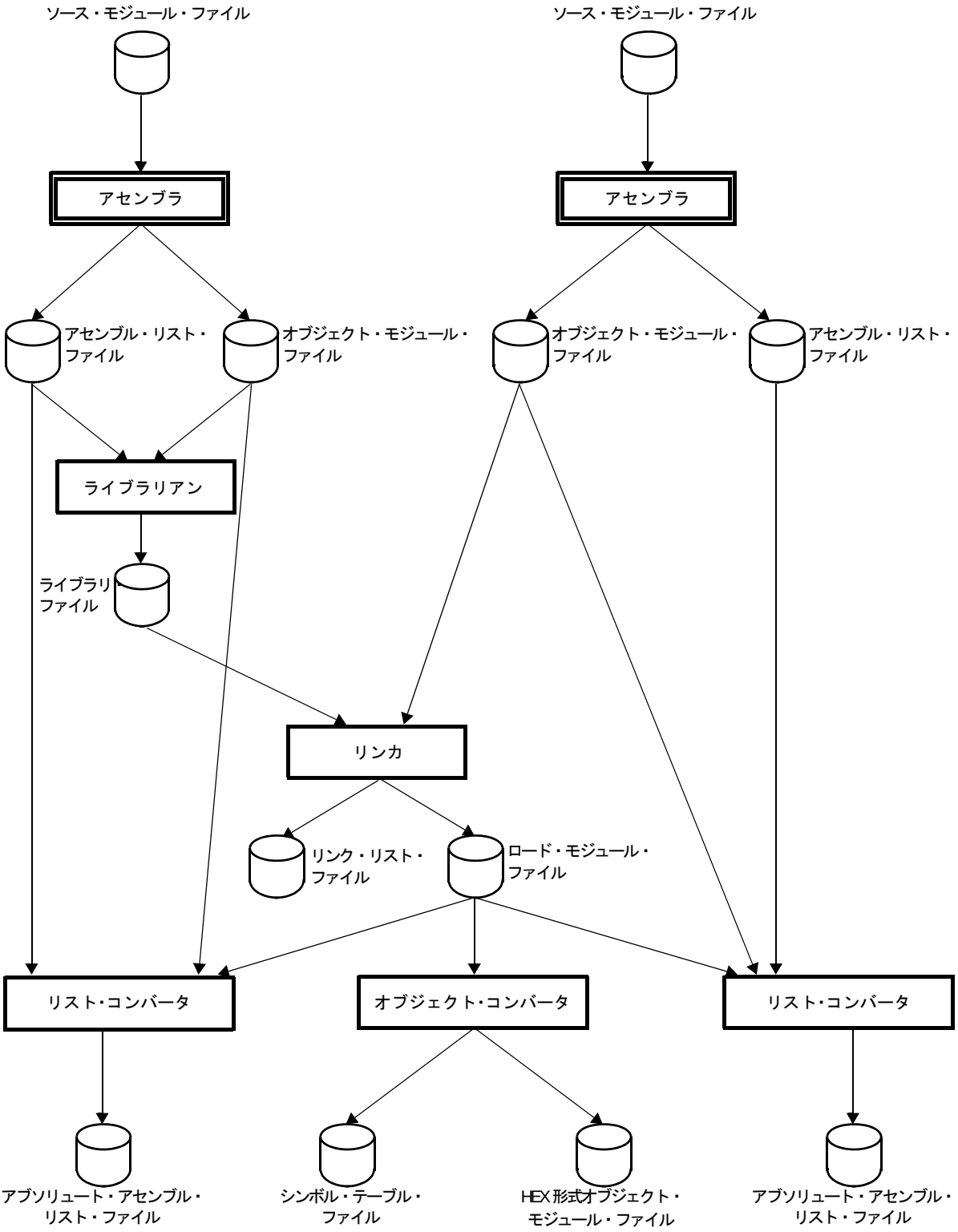


図 3-3 RA78K0R の実行手順 2



3.3 コマンド行での実行手順

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 k0rmain.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>ra78k0r -cf1166a0 k0rmain.asm

   次のメッセージがディスプレイに出力されます。

```
78K0R Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD78F1166_A0
 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 k0rmain.rel とアセンブル・リスト・ファイル k0rmain.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 k0rsub.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>ra78k0r -cf1166a0 k0rsub.asm

   次のメッセージがディスプレイに出力されます。

```
78K0R Series Assembler  Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

Pass1 Start
Pass2 Start

 Target chip : uPD78F1166_A0
 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 k0rsub.rel とアセンブル・リスト・ファイル k0rsub.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。

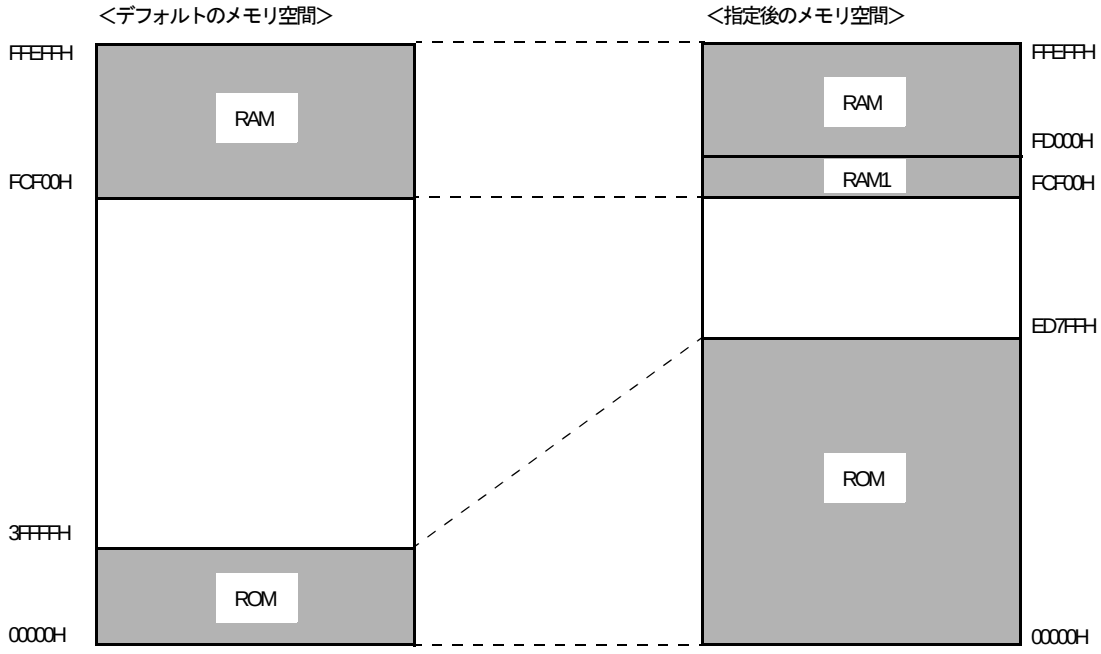
   ディレクティブ・ファイルは，リンカに対してセグメントの配置を指定します。
   したがって，セグメントを配置するためにデフォルトの ROM/RAM 領域を拡張したり，新たにメモリ領域を定義する必要のある場合には，ディレクティブ・ファイルを作成します。
   また，ソース・モジュール中でアブソリュート・セグメントとして定義していないセグメントをメモリ上の特定番地に配置しようとする場合にも，ディレクティブ・ファイルを作成する必要があります。
   ディレクティブ・ファイルは，リンク時に -d オプションを用いてリンカに入力します。

【例】

- ROM（0H - 3FFFFH）領域を ROM（0H - ED7FFH）領域に拡張し，RAM 領域を RAM（FCF00H - FFEFFH）領域と RAM1（FCF00H - FCFFFH）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 , 0ED800H )
MEMORY RAM1 : ( 0FCF00H , 100H )
MEMORY RAM : ( 0FD000H , 2F00H )
```



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“k0rmain.rel”と“k0rsub.rel”をリンクします。

   また，ディレクティブ・ファイルとして，k0r.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>lk78k0r k0rmain.rel k0rsub.rel -dk0r.dr 注 1 -ok0r.lmf -pk0r.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。

```
78K0R Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

 Target chip : uPD78F1166_A0
 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 k0r.lmf とリンク・リスト・ファイル k0r.map を出力します。

   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 k0r.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>oc78k0r k0r.lmf -r -u0FFH

   次のメッセージがディスプレイに出力されます。

```
78K0R Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

 Target chip : uPD78F1166_A0
 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 k0r.hex とシンボル・テーブル・ファイル k0r.sym を出力します。

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 k0rsub.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>lb78k0r < k0r.job
```

次のメッセージがディスプレイに出力されます。

```
78K0R Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx
*create k0r.lib                 ;“k0r.job”の内容
*add k0r.lib k0rsub.rel         ;“k0r.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 k0r.lib を出力します。

12. アブソリュート・アセンブル・リストを作成します。

k0rmain.asm のアブソリュート・アセンブル・リストを作成するため，“k0rmain.rel”，“k0rmain.asm”，および“k0r.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>lc78k0r k0rmain -lk0r.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 RA78K0R Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation 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 k0rmain.p を出力します。

3.4 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，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+ 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 ]ra78k0r △ -f パラメータ・ファイル名
>[ パス名 ]lk78k0r △ -f パラメータ・ファイル名
>[ パス名 ]oc78k0r △ -f パラメータ・ファイル名
```

次に，使用例を示します。

```
C>ra78k0r -fpara.pra
C>lk78k0r -fpara.plk
C>oc78k0r -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することができます。

次に，パラメータ・ファイルをエディタで作成した例を示します。

＜ para1.pra の内容＞

```
-cf1166a0 k0rmain.asm -e
```

＜ para.plk の内容＞

```
k0rmain.rel k0rsub.rel -bmylib.lib -osample.lmf -s
```

＜ para.poc の内容＞

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

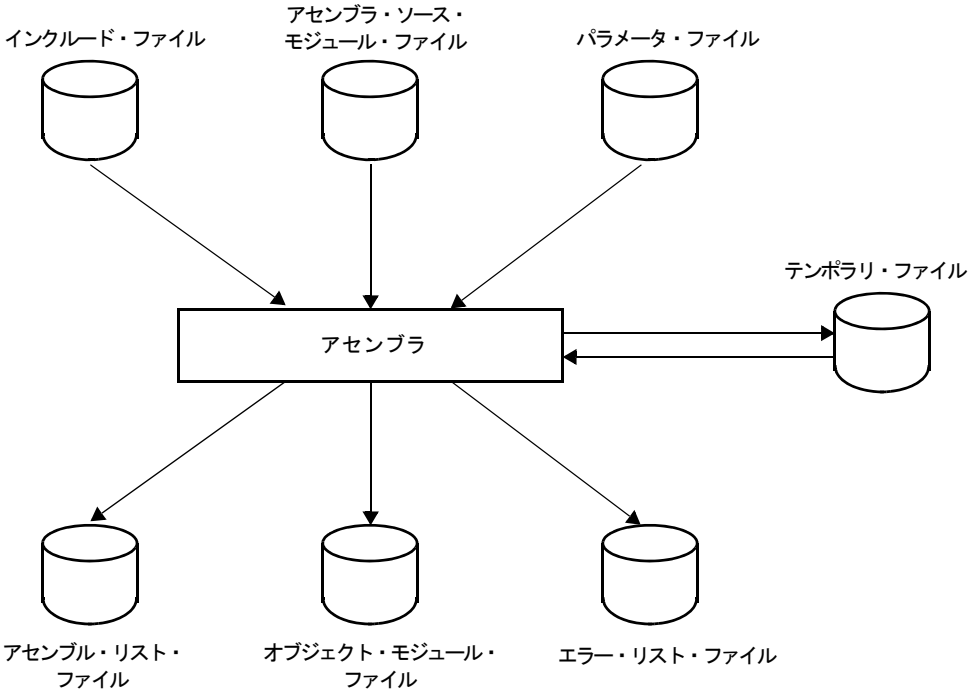
第 4 章 アセンブラ

アセンブラは，78K0R 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。

図 4-1 アセンブラの入出力ファイル



4.1 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 4-1 アセンブラの入出力ファイル

種類	ファイル名	説明	デフォルト・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	- 78K0R 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（ユーザ作成ファイル）	.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 - 78K0R 用のアセンブリ言語で記述されたファイル（ユーザ作成ファイル）	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（ユーザ作成ファイル）	.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	.rel
	アセンブル・リスト・ファイル	- アセンブル・リスト，クロスリファレンス・リストなどのアセンブル情報を持つファイル	.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	.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 アセンブル終了時には消去されます。	RA*xxxxx*.$$n （n = 1 - 4）

4.2 アセンブラの機能

(1) アセンブリ言語を機械語に変換

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。

4.3 アセンブラの起動

4.3.1 アセンブラの起動方法

アセンブラの起動には，2 つの方法があります。

(1) コマンド行での起動

```
 X>[ パス名 ]ra78k0r[ △オプション] … ソース・モジュール・ファイル名 [ △オプション] … [ △ ]
 ↑    ↑       ↑          ↑                    ↑                    ↑
(1)  (2)     (3)        (4)                  (5)                  (4)
```

(1) カレント・ドライブ名

(2) カレント・フォルダ名

(3) アセンブラのコマンド名

(4) 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，それぞれのオプション間を空白で区切ってください。なお，アセンブラ・オプションに大文字，小文字の区別はありません。アセンブラ・オプションの詳細については，「4.4 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は，ダブルクオーテーション（”“）で囲んでください。

(5) 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は，ダブルクオーテーション（”“）で囲んでください。

【例】

- エラー・リスト・ファイル k0rmain.era を出力します。

```
C>ra78k0r -cf1166a0 k0rmain.asm -e -np
```

(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>ra78k0r[ △ ソース・モジュール・ファイル ] △ -f パラメータ・ファイル名
                                          ↑          ↑
                                         (1)        (2)
```

(1) パラメータ・ファイル指定オプション

(2) アセンブラ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 つだけ指定することができます。
- 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，「3.4 パラメータ・ファイルの使用」を参照してください。

【例】

1. パラメータ・ファイル k0rmain.pra をエディタで作成します。

```
; parameter file
k0rmain.asm -osample.rel
-psample.prn
```

2. パラメータ・ファイル k0rmain.pra を使用してアセンブラを起動します。

```
C>ra78k0r -fk0rmain.pra
```

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

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。

```
78K0R Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation 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
k0rmain.asm ( 12 ) : RA78K0R error E2201 : Syntax error
PASS2 Start
k0rmain.asm ( 12 ) : RA78K0R error E2201 : Syntax error
k0rmain.asm ( 29 ) : RA78K0R error E2407 : Undefined symbol reference
'CONVAH'
k0rmain.asm ( 29 ) : RA78K0R 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】

＜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>ra78k0r sample.asm

78K0R Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

RA78K0R 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，アセンブルが中止されます。

＜存在しないアセンブラ・オプションを指定した場合＞

```
C>ra78k0r k0rmain.asm -z

78K0R Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

RA78K0R error F2012 : Missing parameter '-z'
Please enter 'RA78K0R--' , 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，アセンブルが中止されます。

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・メッセージの原因については，「第 11 章 エラー・メッセージ」で調べて対処してください。

4.4 アセンブラ・オプション

4.4.1 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・オプションの分類と説明を示します。

表 4-2 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	
デバッグ情報出力指定	-g	デバッグ情報（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・デバッ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リファレンス・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	

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	78K0R 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。
78K0 シリーズ互換マクロ	-compati	78K0 シリーズ用アセンブラで作成したアセンブラ・ソース・ファイルをアセンブル可能にします。
	-ncompati	
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 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 4-3 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>ra78k0r -cf1166a0 k0rmain.asm -np -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>ra78k0r -cf1166a0 k0rmain.asm -p -nka -nks -nkx
```

-nka，-nks，および -nkx を同時に指定したので，-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>ra78k0r -cf1166a0 k0rmain.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 4-3 に記述されていないオプションは，ほか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。

デバイス種別指定

-c

(1) -c

【記述形式】

-c *デバイス種別*

- 省略時解釈

　省略することはできません。

【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 オプションで指定可能な対象デバイスについては，各デバイスのユーザーズ・マニュアル，または「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 オプションは省略できません。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（$PROCESSOR）を記述すれば，コマンド行での指定を省略することができます。

△ $ △ 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，「RA78K0R 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】

- uPD78F1166_A0 をターゲットとして指定します。

```
C>ra78k0r -cf1166a0 main.asm
```

オブジェクト・モジュール・ファイル出力指定

-o/-no

(1) -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，パス名を指定することができます。デバイス型ファイル名 CON，PRN を指定した場合は，アボート・エラーとなります。
- -o オプションを指定しても，フェイタル・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>ra78k0r -cf1166a0 k0rmain.asm -osample.rel
```

オブジェクト・モジュール・ファイル強制出力指定

-j/-nj

(1) -j/-nj

【記述形式】

```
-j
-nj
```

- 省略時解釈

  -nj

【機能】

- -j オプションは，フェイタル・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，フェイタル・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，フェイタル・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てオブジェクト・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，フェイタル・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】

- フェイタル・エラーの場合でもオブジェクト・モジュール・ファイル k0rmain.rel を出力するよう指定します。

  C>ra78k0r -cf1166a0 k0rmain.asm -j

デバッグ情報出力指定

-g/-ng, -ga/-nga

(1) -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・ファイル中にデバッグ情報（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・シンボルも含めてシンボリック・デバッ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。

  (1) グローバル・シンボルのみのシンボリック・デバッグ

  (2) シンボルなしでのデバッグ

  (3) 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（DEBUG/NODEBUG，または DG/NODG）を記述することができます。

  記述形式を次に示します。

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

  制御命令については，「RA78K0R 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 k0rmain.rel にデバッグ情報（ローカル・シンボル情報）を付加します。

```
C>ra78k0r -cf1166a0 k0rmain.asm -g
```

(2) -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・ソース・デバッグ情報を付加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・レベルでデバッグするときに使用します。ソース・レベルでのデバッグには「統合デバッガ（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。

  (1) アセンブラ・ソースなしでのデバッグ

  (2) 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）

  (3) C コンパイラ・ソース・レベルでのデバッグ

【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（DEBUGA/NODEBUGA）を記述することができます。
  記述形式を次に示します。

```
△ $ △ DEBUGA
△ $ △ NODEBUGA
```

  制御命令については，「RA78K0R 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 k0rmain.rel にアセンブラ・ソース・デバッグ情報を付加します。

```
C>ra78k0r -cf1166a0 k0rmain.asm -ga
```

インクルード・ファイル読み込みパス指定

-i

(1) -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。

  (1) ソース・ファイルのあるパス

  (2) 環境変数 INC78K0R で指定したパス

【機能】

- -i オプションは，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することができます。
- “ , ”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。

  (1) -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。

  (2) -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。

  (3) -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・ファイルをフォルダ C:¥sample から読み込みます。

```
C>ra78k0r -cf1166a0 k0rmain.asm -iC:¥sample
```

アセンブル・リスト・ファイル出力指定

-p/-np

(1) -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することができます。
  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。
- -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 を出力します。

```
C>ra78k0r -cf1166a0 k0rmain.asm -psample.prn
```

アセンブル・リスト・ファイル情報指定

-ka/-nka, -ks/-nks, -kx/-nkx

(1) -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・ファイル k0rmain.prn 中にアセンブル・リストを出力します。

C>ra78k0r -cf1166a0 k0rmain.asm -ka

＜ k0rmain.prn の内容＞

```
        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                          EXTRN   _@STBEG
14    14
15    15    -----                 DATA    DSEG    AT      0FFE20H
16    16    FFE20                 HDTSA : DS      1
17    17    FFE21                 STASC : DS      2
18    18
19    19    -----                 CODE    CSEG    AT      0H
20    20    00000 R0000           MAIN :  DW      START
21    21
22    22    -----                         CSEG
23    23    00000                 START :
24    24
25    25                                  ; chip initialize
26    26    00000 RCBF80000               MOVW    SP , #_@STBEG
27    27
28    28    00004 CD201A                  MOV     HDTSA , #1AH
29    29    00007 3620FE                  MOVW    HL , #LOWW ( HDTSA ) ; set hex
2-code data in HL registor
        :
```

(2) -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 は無視されます。
- -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】

- アセンブル・リスト・ファイル k0rmain.prn 中に，アセンブル・リストに続いてシンボル・リストを出力します。

  C>ra78k0r -cf1166a0 k0rmain.asm -ks

＜ k0rmain.prn の内容＞

```
        Symbol Table List

VALUE   ATTR    RTYP    NAME            VALUE   ATTR    RTYP    NAME

        CSEG            ?CSEG                   CSEG            CODE
-----H          EXT     CONVAH                  DSEG            DATA
FFE20H  ADDR            HDTSA              0H   ADDR    PUB     MAIN
        MOD             SAMPM              0H   ADDR    PUB     START
FFE21H  ADDR            STASC           -----H          EXT     _@STBEG
```

(3) -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リファレンス・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，クロスリファレンス・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。
- -ks と -kx を同時に指定した場合は，-ks は無視されます。
- -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（XREF/NOXREF，または XR/NOXR）を記述することができます。

  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，「RA78K0R 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 k0rmain.prn 中に，アセンブル・リストに続いてクロスリファレンス・リストを出力します。

```
C>ra78k0r -cf1166a0 k0rmain.asm -kx
```

＜ k0rmain.prn の内容＞

```
        Cross-Reference List

NAME    VALUE   R       ATTR    RTYP    SEGNAME XREFS

?CSEG                   CSEG            ?CSEG   22#
CODE                    CSEG            CODE    19#
CONVAH  -----H  E               EXT             12@     31
DATA                    DSEG            DATA    15#
HDTSA   FFE20H          ADDR            DATA    16#     28      29
MAIN        0H          ADDR    PUB     CODE    11@     20#
SAMPM                   MOD                      2#
START       0H  R       ADDR    PUB     ?CSEG   11@     20      23#
STASC   FFE21H          ADDR            DATA    17#     33
_@STBEG -----H  E               EXT             13@     26
```

アセンブル・リスト・ファイル形式指定

-lw, -ll, -lh, -lt, -lf/-nlf

(1) -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 オプションで指定可能な文字数の範囲（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。
- 文字数を省略した場合は，132 を指定したものとみなされます。

  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（WIDTH）を記述することができます。

  記述形式を次に示します。

```
△ $ △ WIDTH
```

  制御命令については，「RA78K0R 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 k0rmain.prn の 1 行の文字数を 80 文字に指定します。

C>ra78k0r -cf1166a0 k0rmain.asm -lw80

＜アセンブル・リスト・ファイル k0rmain.prn の内容＞

```
        Assemble list

ALNO  STNO  ADRS  OBJECT  M I   SOURCE STATEMENT

 1     1                                NAME  SAMPM
 2     2                          ; ***********************************
 3     3                          ; *                                 *
 4     4                          ; * HEX -> ASCII Conversion Program  *
 5     5                          ; *                                 *
 6     6                          ; *           main-routine          *
 7     7                          ; *                                 *
 8     8                          ; ***********************************
 9     9
10    10
11    11                          PUBLIC  MAIN , START
12    12                          EXTRN   CONVAH
13    13                                  EXTRN   _@STBEG
14    14
15    15    -----                 DATA    DSEG    AT      0FFE20H
16    16    FFE20                 HDTSA : DS      1
17    17    FFE21                 STASC : DS      2
18    18
19    19    -----                 CODE    CSEG    AT      0H
20    20    00000   R0000         MAIN :  DW      START
21    21
22    22    -----                         CSEG
23    23    00000                 START :
24    24
25    25                                  ; chip initialize
26    26    00000   RCBF80000            MOVW    SP , #_@STBEG
27    27
28    28    00004   CD201A               MOV     HDTSA , #1AH
29    29    00007   3620FE               MOVW    HL , #LOWW ( HDTSA ) ; set hex
2-code data in HL registor
        :
```

(2) -ll

【記述形式】

```
-ll[ 行数 ]
```

- 省略時解釈

  -ll0（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，-ll オプションを指定します。

【説明】

- -ll オプションで指定可能な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した場合には，アボート・エラーとなります。
- 行数を省略した場合，0 を指定したものとみなされます。
- 行数の 0 を指定した場合は，改頁されません。
- -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（LENGTH）を記述することができます。記述形式を次に示します。

```
△ $ △ LENGTH
```

制御命令については，「RA78K0R 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 k0rmain.prn の 1 頁の行数を 20 行に指定します。

```
C>ra78k0r -cf1166a0 k0rmain.asm -ll20
```

＜ k0rmain.prn の内容＞

```
78K0R Series Assembler Vx.xx             Date:xx xxx xxxx Page:  1

Command : -cf1166a0 k0rmain.asm -ll20
Para-file :
In-file : k0rmain.asm
Obj-file : k0rmain.rel
Prn-file : k0rmain.prn

        Assemble list

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


78K0R 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            *
 8     8                          ;

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


78K0R Series Assembler Vx.xx            Date : xx xxx xxxx Page:  3


ALNO  STNO  ADRS  OBJECT   M I  SOURCE  STATEMENT

 9     9                          ; ********************************
10    10
11    11                          PUBLIC  MAIN , START
12    12                          EXTRN   CONVAH
13    13                          EXTRN   _@STBEG
14    14
15    15    -----                 DATA    DSEG    AT      0FFE20H
16    16    FFE20                 HDTSA : DS      1
        :
```

(3) -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，文字列内に空白を記述することはできません。
- 61 文字以上記述した場合には，先頭の 60 文字が有効となり，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 文字として計算されます。

  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
- 文字列を指定しなかった場合は，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。

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されます
;	”” 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
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（TITLE，または TT）を記述することができます。

  記述形式を次に示します。

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

  制御命令については，「RA78K0R 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 k0rmain.prn のヘッダにタイトル“RA78K0R_MAINROUTINE”を印字します。

C>ra78k0r -cf1166a0 k0rmain.asm -lhRA78K0R_MAINROUTINE

＜ k0rmain.prn の内容＞

```
78K0R Series Assembler Vx.xx  RA78K0R_MAINROUTINE Date:xx xxx xxx Page : 1
                                        ↑
                                      タイトル

Command : -cf1166a0 k0rmain.asm -lhRA78K0R_MAINROUTINE
Para-file :
In-file : k0rmain.asm
Obj-file : k0rmain.rel
Prn-file : k0rmain.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             *
       :
```

(4) -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 オプションで指定可能な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 を指定した場合は，タビュレーション処理は行わず，タブ・コードが出力されます。
- -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（TAB）を記述することができます。

  記述形式を次に示します。

△ $ △ TAB △ *タブ数*

  制御命令については，「RA78K0R 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】

- -lt オプションを省略した場合のアセンブル・リスト・ファイル sample.prn を参照します。

C>ra78k0r -cf1166a0 sample.asm

＜ sample.prn の内容＞

```
       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                          EXTRN   _@STBEG
14    14
15    15    -----                 DATA    DSEG    AT      0FFE20H
16    16    FFE20                 HDTSA : DS      1
17    17    FFE21                 STASC : DS      2
18    18
19    19    -----                 CODE    CSEG    AT      0H
20    20    00000 R0000           MAIN :  DW      START
       :
```

- HT コードによるブランクを 1 に指定します。

C>ra78k0r -cf1166a0 sample.asm -lt1

＜ sample.prn の内容＞

```
       Assemble list

ALNO  STNO  ADRS  OBJECT  M I   SOURCE STATEMENT

 1     1                                NAME  SAMPM
 2     2                          ; ****************************************
 3     3                          ;
 4     4                          ;       HEX -> ASCII Conversion Program
 5     5                          ;
 6     6                          ;             main-routine
 7     7                          ;
 8     8                          ; ****************************************
 9     9
10    10                          PUBLIC MAIN , START
11    11                          EXTRN CONVAH
12    12                          EXTRN _@STBEG
13    13
14    14     -----                DATA DSEG AT 0FFE20H
15    15     FFE20                HDTSA : DS 1
16    16     FFE21                STASC : DS 2
17    17
18    18     -----                CODE CSEG AT 0H
19    19     00000 R0000          MAIN : DW START
20    20       :
```

備考　HT コードによるブランクは 1 つです。

(5) -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，-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，あとで指定した方が優先されます。

【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（FORMFEED/NOFORMFEED）を記述できます。
  記述形式を次に示します。

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

制御命令については，「RA78K0R 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 k0rmain.prn の最後に，改頁コードを付加します。

```
C>ra78k0r -cf1166a0 k0rmain.asm -p -lf
```

エラー・リスト・ファイル出力指定

-e/-ne

(1) -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することができます。
- -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 k0rmain.era を作成します。

C>ra78k0r -cf1166a0 k0rmain.asm -ek0rmain.era

```
78K0R Series Assembler Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

PASS1 Start
k0rmain.asm ( 31 ) : RA78K0 error E2202 : lllegal operand
PASS2 Start
k0rmain.asm ( 26 ) : RA78K0 error E2312 : Operand out of range ( byte )
k0rmain.asm ( 31 ) : RA78K0 error E2202 : lllegal operand

 Target chip : uPD78F1166_A0
 Device file : Vx.xx

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

＜ k0rmain.era の内容＞

```
PASS1 Start
k0rmain.asm ( 31 ) : RA78K0R error E2202 : lllegal operand
PASS2 Start
k0rmain.asm ( 26 ) : RA78K0R error E2312: Operand out of range ( byte )
k0rmain.asm ( 31 ) : RA78K0R error E2202: lllegal operand
```

パラメータ・ファイル指定

-f

(1) -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となります。

【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行います。

【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することができるのは，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・ファイル中に記述可能な文字数に，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が優先されます。
- “;”，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されます。
- -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 k0rmain.pra の内容は，次のように設定します。

```
; parameter file
k0rmain.asm -osample.rel -g -cf1166a0
-psample.prn
```

コマンド行には，次のように入力します。

```
C>ra78k0r -fk0rmain.pra
```

テンポラリ・ファイル作成パス指定

-t

(1) -t

【記述形式】

-t *パス名*

- 省略時解釈

  環境変数 TMP で指定したパス

  環境変数 TMP を指定していない場合は，カレント・パス

【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することができます。

【説明】

- パス名として，パス以外のものは指定できません。
- パス名を省略することはできません。
- 以前に作成したテンポラリ・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されます。
- 必要とするメモリ・サイズがある間は，テンポラリ・ファイルはメモリに展開されます。
  メモリが足りなくなった時点で，メモリに展開されていたテンポラリ・ファイルの内容がディスクに書き出されます。
  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われます。
- 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。

  (1) -t オプションで指定されたパス

  (2) 環境変数 TMP で設定したパス（-t オプション省略の場合）

  (3) カレント・パス（TMP を設定していない場合）

  なお，(1)，または (2) を指定した場合，指定し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・ファイルをフォルダ C:¥tmp に出力します。

```
C>ra78k0r -cf1166a0 k0rmain.asm -tC:¥tmp
```

漢字コード指定

-zs/-ze/-zn

(1) -zs/-ze/-zn

【記述形式】

```
-zs
-ze
-zn
```

- 省略時解釈

  -zs

【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。
- 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（KANJICODE）を記述することができます。

  記述形式を次に示します。

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

  制御命令については，「RA78K0R 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，「10.2 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>ra78k0r k0rmain.asm -cf1166a0 -ze
```

デバイス・ファイル・サーチ・パス指定

-y

(1) -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，次の順序で調べて決定されます。

  (1) デバイス・ファイル・インストーラで登録されたパス

  (2) RA78K0R の起動されたパス

  (3) カレント・フォルダ

  (4) 環境変数 PATH

【機能】

- -y オプションは，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。

  (1) -y オプションで指定されたパス

  (2) デバイス・ファイル・インストーラで登録されたパス

  (3) RA78K0R の起動されたパス

  (4) カレント・フォルダ

  (5) 環境変数 PATH

【使用例】

- デバイス・ファイルのパスを C:¥78k0r¥dev フォルダに指定します。

```
C>ra78k0r k0rmain.asm -cf1166a0 -yC:¥78k0r¥dev
```

シンボル定義指定

-d

(1) -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】

- -d オプションは，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 個まで指定することができます。
- シンボル名は，31 文字まで記述することができます。
- 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>ra78k0r k0rmain.asm -cf1166a0 -dSYM=2
```

シリーズ共通オブジェクト指定

-common

(1) -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，78K0R 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，78K0R 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。

  78K0R 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】

- 78K0R 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】

- 78K0R 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。

```
C>ra78k0r k0rsub.c -cf1166a0 -common
```

セルフ・プログラミング指定

-self

(1) -self

【記述形式】

```
-self
```

- 省略時解釈

  なし

【機能】

- -self オプションは，xxxxxH 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 ! xxxxxH”の記述に対してエラーを出力しません。

  なお，アドレス xxxxxH 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。

【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 ! xxxxxH”の記述に対してエラーとなる場合に指定してください。

【使用例】

- セルフ・プログラミング使用時に，“CALL ! xxxxxH”の記述に対してエラーを出力しないよう指定します。

  C>ra78k0r k0rsub.asm -cf1166a0 -self

78K0 シリーズ互換マクロ

-compati/-ncompati

(1) -compati/-ncompati

【記述形式】

```
-compati
-ncompati
```

- 省略時解釈

  -ncompati

【機能】

- -compati オプションは，78K0 シリーズ用アセンブラで作成したアセンブラ・ソース・ファイルをアセンブル可能にします。
- -ncompati オプションは，-compati オプションを無効にします。

【用途】

- 通常，78K0R で使用できない 78K0 の命令が記述されたアセンブラ・ソースをアセンブルした場合，フェイタル・エラー（E2337）となります。

  下記の 78K0R で使用できない 78K0 の命令をソースの記述を変更せずにアセンブルしたい場合，-compati オプションを指定します。

  78K0R で使用できない 78K0 の命令　：DIVUW/ROR4/ROL4/ADJBA/ADJBS/CALLF/DBNZ

【説明】

- -compati オプションを指定すると，「..¥inc78k0r¥compati.inc」（ra78k0r.exe の起動されたパスに対して）をインクルードして，78K0R で使用することができない 78K0 の命令をマクロ変換します。

【使用例】

- 78K0R で削除された 78K0 の命令をマクロ変換します。

  C>ra78k0r k0rsub.asm -cf1166a0 -compati

ヘルプ指定

--

(1) --

【記述形式】

```
--
```

- 省略時解釈

  表示しません。

【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，ほかのアセンブラ・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・キーを押下してください。
  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・キーを押下してください。

注意 本オプションは，PM+ 上では指定することはできません。
PM+ 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>ra78k0r --

```
78K0R Series Assembler Vx.xx [xx xxx xxxx]
   Copyright (C) NEC Electronics Corporation xxxx

usage : ra78k0r [option[...]] input-file [option[...]]
The option is as follows ([ ] means omissible).
-cx             : Select target chip. (x = f1166a0 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 78K0R 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 -ll0 -nlf -lt8 -g -nj -ga
```

4.5 PM+ でのオプション設定

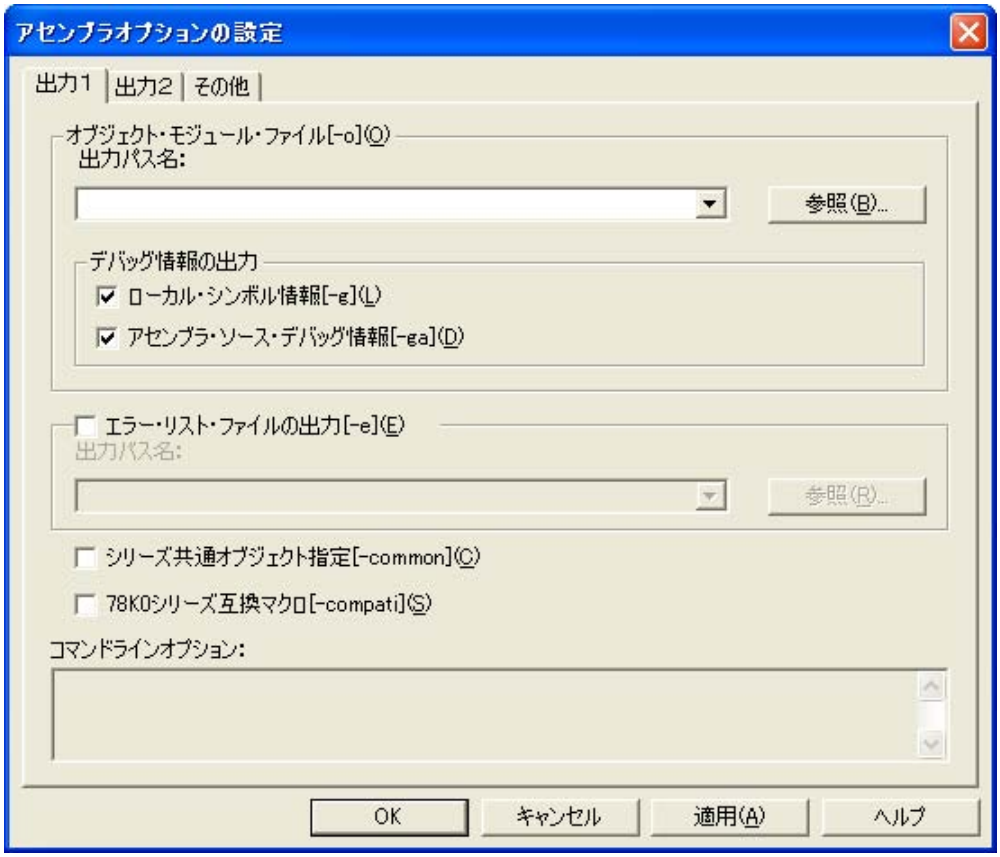
PM+ からアセンブラ・オプションを設定する方法について説明します。

4.5.1 オプションの設定方法

PM+ の [ツール (T)] メニュー→ [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・オプションを設定することができます。

図 4-2 [アセンブラオプションの設定] ダイアログ

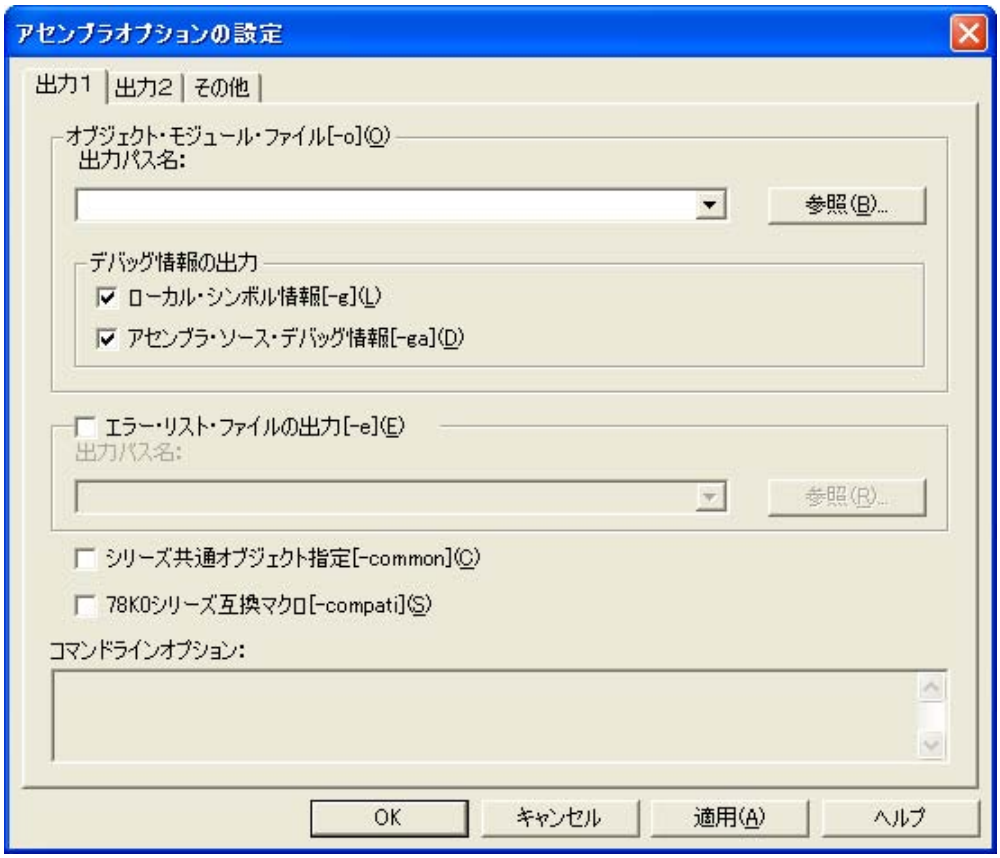


4.5.2 ダイアログの説明

[アセンブラオプションの設定] ダイアログの各タブについて，次に説明します。

(1) [出力 1] タブ

図 4-3 [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）



- 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，デバッグ情報（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・デバッグ情報 [-ga](D)

オブジェクト・モジュール・ファイル中に，ソース・デバッ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)

  78K0R 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。

- 78K0 シリーズ互換マクロ [-compati](S)

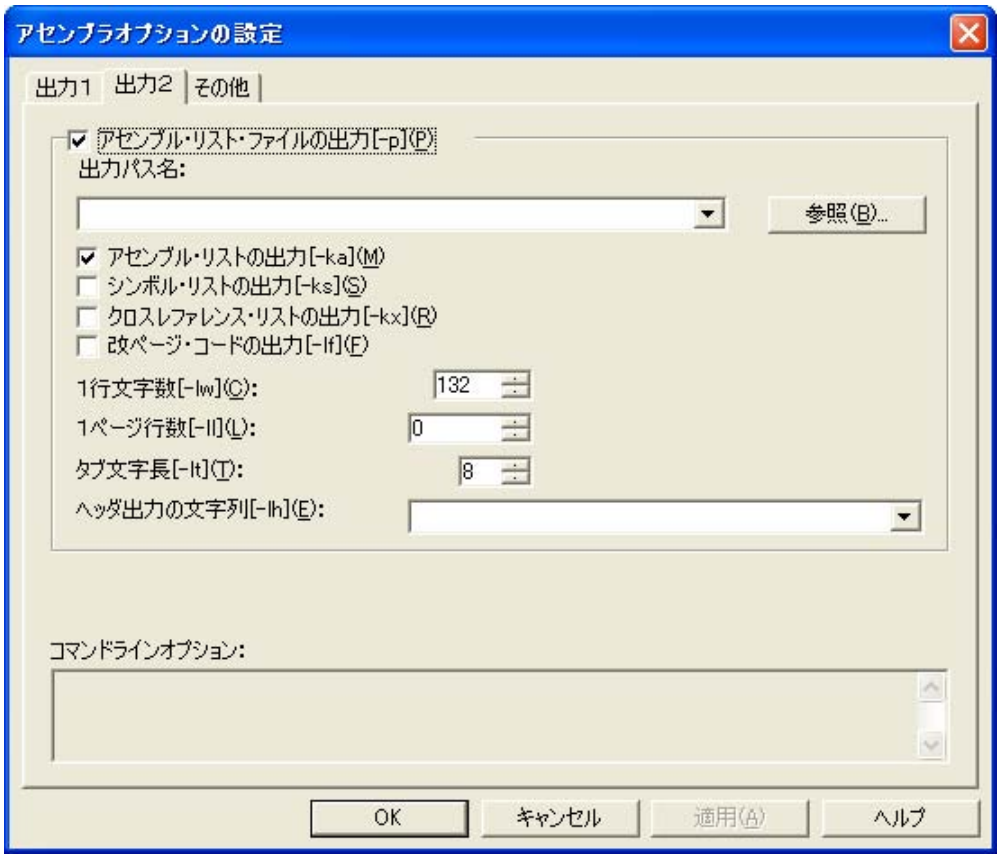
  78K0 シリーズ用アセンブラで作成したアセンブラ・ソース・ファイルをアセンブル可能にします。

- 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。

(2) [出力 2] タブ

図 4-4 [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）



- 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。

- クロスリファレンス・リストの出力 [-kx](R)

  アセンブル・リストに続いてクロスリファレンス・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 〜 2,046 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，0，および 20 〜 32,767 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 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。

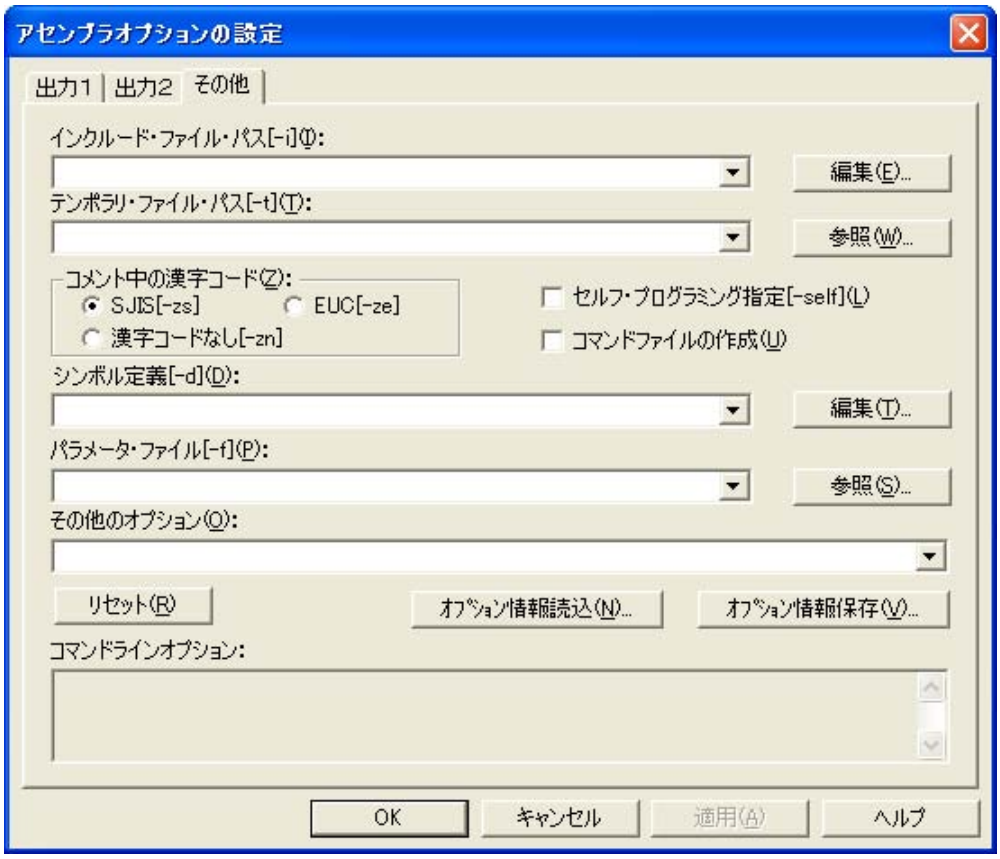
  入力可能な文字数は 60 文字までです。

- 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。

(3) [その他] タブ

図 4-5 [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）



- インクルード・ファイル・パス [-i](I)

  ［編集 (E)...］ボタン（[オプションの編集] ダイアログをオープン），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。

  パスは，カンマで区切って 64 個まで指定することができます。

- 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，チェックします。

- シンボル定義 [-d](D)

［編集 (T)...］ボタン（[オプションの編集] ダイアログをオープン），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。

シンボル定義は，カンマで区切って 30 個まで指定することができます。各シンボルの指定可能な文字数は，31 文字までです。

- 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+ 上では指定することはできません。

- 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)...

[オプション情報の読込み] 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

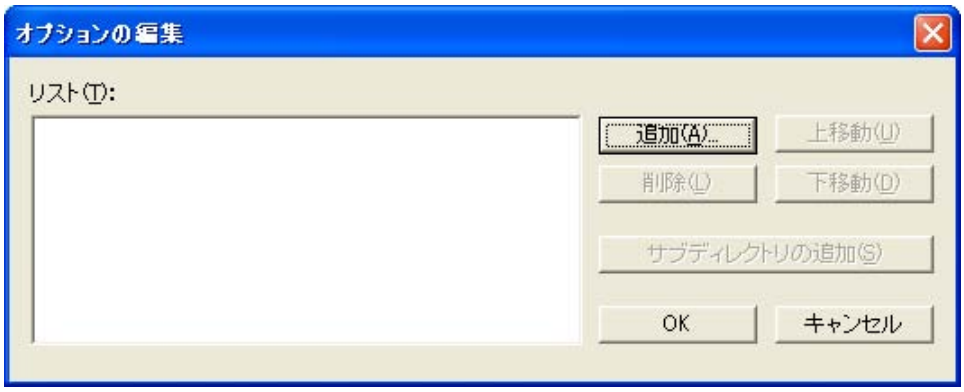
このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

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 4-6 [オプションの編集] ダイアログ



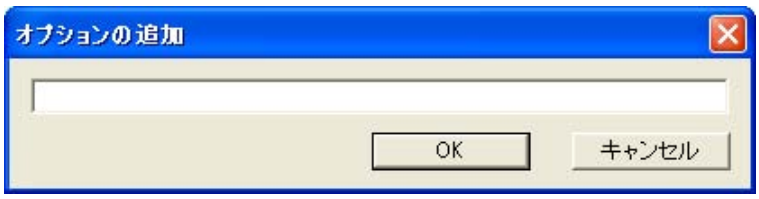
- ［追加 (A)］ボタン

  リストの項目を追加します。

  ファイルやフォルダ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。

図 4-7 [オプションの追加] ダイアログ



- ［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)

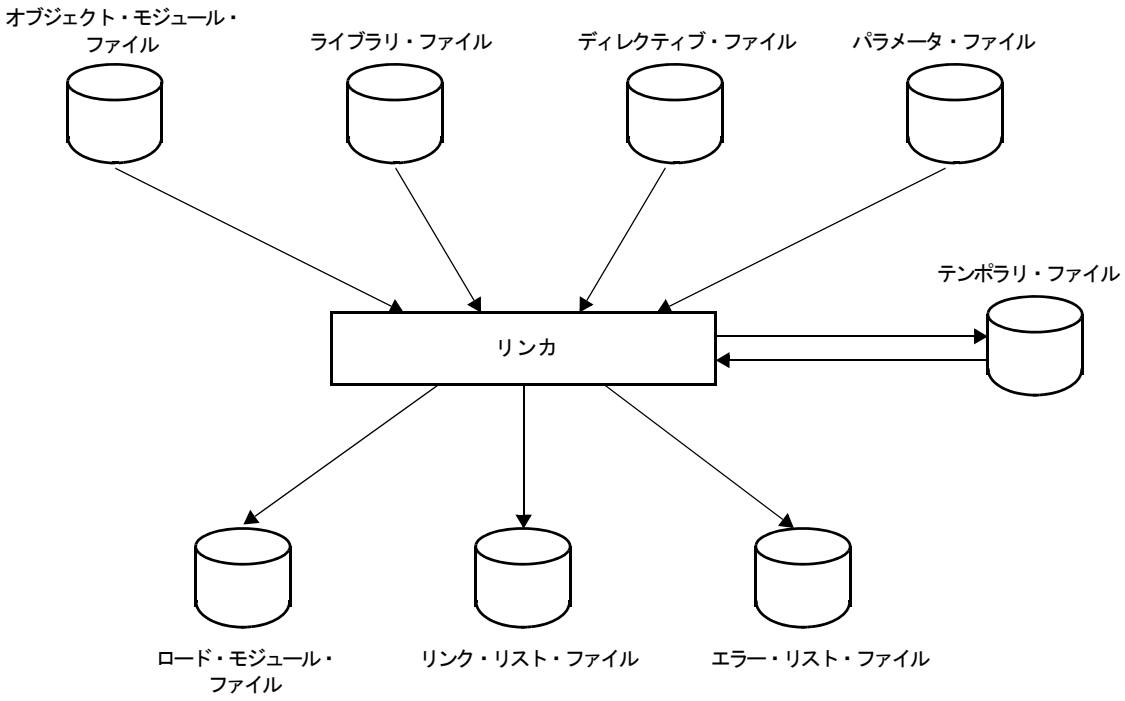
第 5 章 リンカ

リンカは，78K0R 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。

図 5-1 リンカの入出力ファイル



5.1 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 5-1 リンカの入出力ファイル

種類	ファイル名	説明	デフォルト・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 - アセンブラの出力したファイル	.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 - ライブラリアンの出力したファイル	.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（ユーザ作成ファイル）	.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（ユーザ作成ファイル）	.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 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	.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	.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 リンク終了時には消去されます。	LK*xxxxx*.$$n （n = 1 - 3）

5.2 リンカの機能

(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。

5.3 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。

注意 メモリ空間は 1 つだけです。メモリ空間名として REGULAR 以外の指定を行うことはできません。

表 5-2 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	デフォルトのアドレス	デフォルトで配置されるセグメント
ROM	内蔵 ROM （ROM レス品の場合は RAM 領域の前まで）	CSEG
RAM	内蔵 RAM	DSEG，BSEG

備考 1 メモリ領域のデフォ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，「3.3 コマンド行での実行手順　5.」を参照してください。

5.4 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略します）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・ファイルの役割を次に示します。

ディレクティブには，次の 2 種類があります。

ディレクティブの種類	役割
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。 **【例】** CALLT 領域 内蔵 ROM 外付け ROM SADDR SADDR 以外の内蔵 RAM
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。 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

エディタなどを使用してディレクティブを記述したファイル（ディレクティブ・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ます。

これにより，リンカはディレクティブ・ファイルを読み込み，解釈しながらリンク処理を行います。

5.4.1 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。

- 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*]

MERGE *セグメント名* : [*結合属性*] [= *メモリ領域名指定*] [/ *メモリ空間名*]

なお，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することができます。

各ディレクティブの詳細については，「5.4.2 メモリ・ディレクティブ」，「5.4.3 セグメント配置ディレクティブ」を参照してください。

(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。

予約語は，ディレクティブ・ファイル中で，ほかの意味（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	説明
MEMORY	メモリ・ディレクティブを指定
MERGE	セグメント配置ディレクティブを指定
AT	セグメント配置ディレクティブの配置属性（スタート・アドレス）を指定
SEQUENT	セグメント配置ディレクティブの結合属性（セグメントを結合する）を指定
COMPLETE	セグメント配置ディレクティブの結合属性（セグメントを結合しない）を指定

注意 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して記述することはできません。

【例】

MEMORY : 使用可

memory : 使用可

Memory : 使用不可

(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，大文字と小文字は区別されます。

(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 進数を記述することができます。

記述方法はソースと同じで，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 78F1166_A0

MEMORY MEM1 : (40000H , 10000H) #SECOND MEMORY AREA

5.4.2 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することができます。

メモリ領域は，デフォルトで定義されているメモリ領域を含め，100 個まで定義することができます。

【構文】

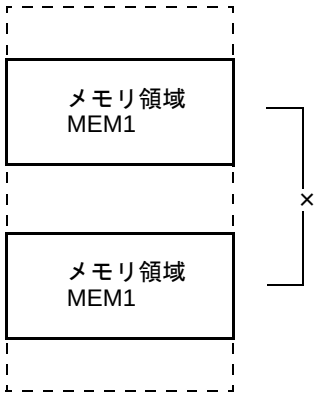
```
MEMORY △メモリ領域名△：△(△スタート・アドレス△,△サイズ△)[/△メモリ空間名]
```

(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 256 文字です。
  257 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。



(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。
- リンカがデフォ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。

  各デバイスのデフォ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，64K バイトの空間 REGULAR で表されます。

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することができます。
- 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。
- デフォ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フォルトのメモリ領域のみを指定したものとします。
- デフォ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，デフォ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 )
```

5.4.3 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 ]
MERGE △セグメント名△：△ [ 結合属性 ] [ △ = △メモリ領域名 ] [ △ / △メモリ空間名 ]
```

(1) セグメント名

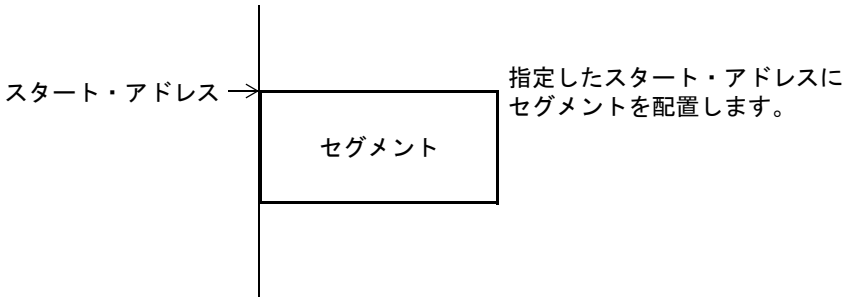
リンカに入力するオブジェクト・モジュール・ファイル中に含まれるセグメント名です。

- 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。大文字と小文字を混在することはできません。
- 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・アドレスを指定することはできません。

(3) 結合属性

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（デフォルト）”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
```

(4) 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，REGULAR のみです。
- 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。

次にセグメントの配置先を示します。

メモリ領域	メモリ空間	セグメントの配置先
指定なし	指定なし	REGULAR 空間中のデフォルト状態のとき配置されるメモリ領域
メモリ領域名	指定なし	REGULAR 空間の指定されたメモリ領域

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 BASE”であるセグメントに，メモリ名“REGULAR”が指定された場合，セグメントが 0000H - FFFFH 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し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が決定されます。
- 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，エラーとなります。
- 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は，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 )
```

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

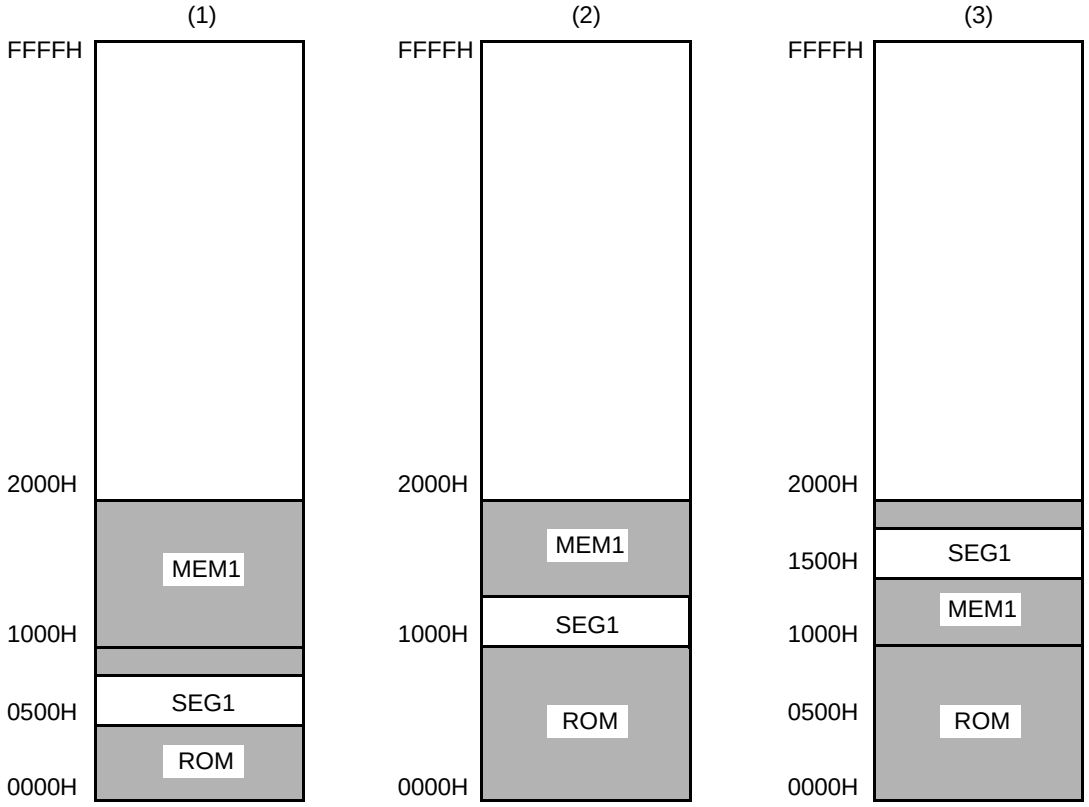
  MERGE　SEG1 : AT (500H)

- 入力セグメント SEG1 をメモリ領域 MEM1 中に割り付ける場合（次図 (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（次図 (3) 参照）

  MERGE　SEG1 : AT (1500H) = MEM1



5.5 リンカの起動

5.5.1 リンカの起動方法

リンカの起動には，2 つの方法があります。

(1) コマンド行での起動

```
 X>[ パス名 ]lk78k0r[ △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 ] …
[ △ ]
 ↑    ↑       ↑          ↑                      ↑                      ↑
(1)  (2)     (3)        (4)                    (5)                    (4)
```

(1) カレント・ドライブ名

(2) カレント・フォルダ名

(3) リンカのコマンド名

(4) 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，それぞれのオプション間を空白で区切ってください。なお，リンカ・オプションに大文字，小文字の区別はありません。リンカ・オプションの詳細については，「5.6 リンカ・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。

(5) リンクするオブジェクト・モジュール・ファイル名

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】

- デバッグ情報を付加したロード・モジュール・ファイル k0r.lmf を出力します。

```
C>lk78k0r k0rmain.rel k0rsub.rel -ok0r.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>LK78K0R[ △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
  ↑     ↑                                          ↑          ↑
(1)    (2)                                        (3)        (4)
```

(1) カレント・ドライブ名

(2) カレント・フォルダ名

(3) パラメータ・ファイル指定オプション

(4) 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るすべてのリンク・オプション，出力ファイル名を記述します。パラメータ・ファイルについての詳細は，「3.4 パラメータ・ファイルの使用」を参照してください。

【例】

1. パラメータ・ファイル k0r.plk をエディタで作成します。

```
; parameter file
k0rmain.rel k0rsub.rel -ok0r.lmf -pk0r.map -e
-tC:¥tmp
```

2. パラメータ・ファイル k0r.plk を使用してリンカを起動します。

```
C>lk78k0r -fk0r.plk
```

5.5.2 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。

```
78K0R Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation 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】

＜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>lk78k0r samp1.rel samp2.rel

```
78K0R Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

RA78K0R error F3006 : File not found 'samp1.rel'
RA78K0R 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す。

＜存在しないリンカ・オプションを指定した場合＞

C>lk78k0r k0rmain.rel k0rsub.rel -z

```
78K0R Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

RA78K0R error F3018 : Option is not recognized '-z'
Please enter 'LK78K0R --' , 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す。

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・メッセージの原原因については，「第 11 章 エラー・メッセージ」で調べて対処してください。

5.6 リンカ・オプション

5.6.1 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・オプションの分類と説明を示します。

表 5-3 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	
デバッグ情報出力指定	-g	デバッグ情報をロード・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。

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・メモリ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・メモリ領域の先頭アドレスを指定します。
オンチップ・デバッグ・オプション・バイト指定	-go	オンチップ・デバッグ・オプション・バイトを指定します。
セキュリティ ID 指定	-gi	セキュリティ ID を指定します。
ユーザ・オプション・バイト指定	-gb	ユーザ・オプション・バイトに設定する値を指定します。
ミラー領域指定	-mi	ミラー元のセグメントの配置先を指定します。
64K バイト境界配置指定	-ccza	各 64K バイト境界の領域の最後の 1 バイトにセグメントの配置を行うかどうかを指定します。
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。

5.6.2 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 5-4 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>lk78k0r k0rmain.rel k0rsub.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>lk78k0r k0rmain.rel k0rsub.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>lk78k0r k0rmain.rel k0rsub.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 5-4 に記述されていないオプションは，ほか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。

ロード・モジュール・ファイル出力指定

-o/-no

(1) -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。これにより，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 を指定することができます。
- -o オプションを指定してもフェイタル・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・フォルダにロード・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 k0r.lmf を出力します。

```
C>lk78k0r k0rmain.rel k0rsub.rel -ok0r.lmf
```

ロード・モジュール・ファイル強制出力指定

-j/-nj

(1) -j/-nj

【記述形式】

```
-j
-nj
```

- 省略時解釈

  -nj

【機能】

- -j オプションは，フェイタル・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，フェイタル・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，フェイタル・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，フェイタル・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 k0rsub.lmf を強制的に出力します。

```
C>lk78k0r k0rmain.rel k0rsub.rel -j
```

デバッグ情報出力指定

-g/-ng

(1) -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・ファイル中にデバッグ情報（ローカル・シンボル情報）を付加することを指定します。
- -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・デバッガでシンボリック・デバッグを行うときには，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 k0rsub.lmf にデバッグ情報を付加します。

```
C>lk78k0r k0rmain.rel k0rsub.rel -g
```

スタック解決用シンボル生成指定

-s/-ns

(1) -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，またはデフォ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。

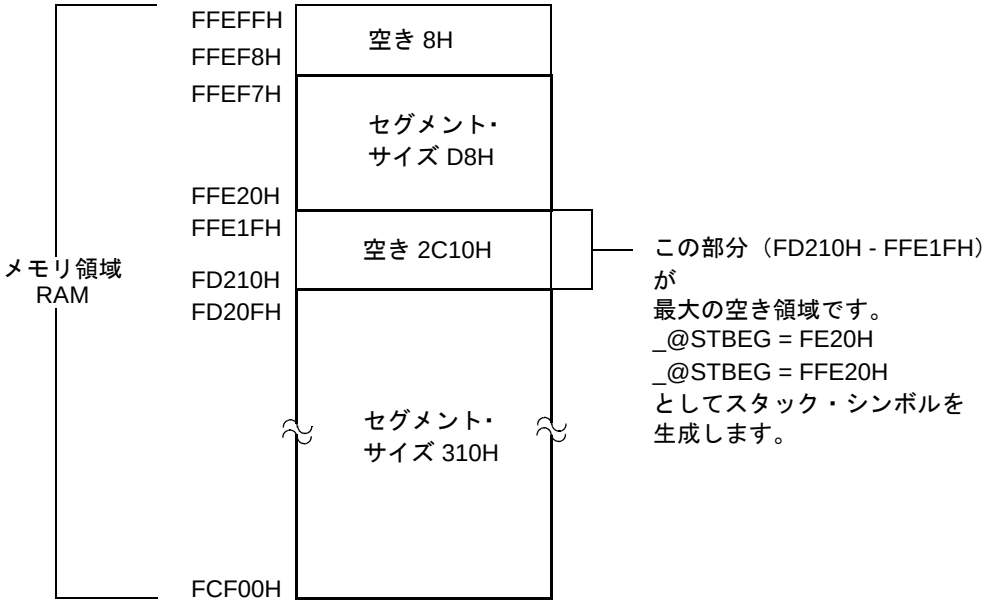
  これらのシンボルは，パブリック宣言された NUMBER 属性のシンボルとして扱われ，リンカのシンボル・テーブルの最後に登録されます。また，リンク・リスト・ファイルに出力される際，モジュール名欄は空白となります。
- 最大の空き領域のサイズが 10 バイト以下の場合，ワーニング・メッセージが出力されます。
- 空き領域が存在しない場合，ワーニング・メッセージが出力され，“_@STEND”と“_@STBEG”は指定したメモリ領域の最終アドレス + 1 を持ちます。
- 領域名を省略した場合，“RAM”が指定されたものとします。
- -s と -ns の両オプションを同時に指定した場合は，あとで指定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>lk78k0r k0rmain.rel k0rsub.rel -s
```



ディレクティブ・ファイル指定

-d

(1) -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，デフォ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することができるのは，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・ファイル中に記述可能な文字数に，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，「5.4 リンク・ディレクティブ」を参照してください。

【使用例】

- デフォルトのメモリ領域 ROM/RAM を再定義します。

  ＜ディレクティブ・ファイル k0r.dr の内容＞

```
memory ROM : ( 0H , 40000H )
memory RAM : ( 0FCF00H , 3000H )
```

  ディレクティブ・ファイル k0r.dr をリンクします。

```
C>lk78k0r k0rmain.rel k0rsub.rel -dk0r.dr
```

リンク・リスト・ファイル出力指定

-p/-np

(1) -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することができます。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。
- -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・フォルダにリンク・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 k0r.map を作成します。

  C>lk78k0r k0rmain.rel k0rsub.rel -pk0r.map

リンク・リスト・ファイル情報指定

-km/-nkm, -kd/-nkd , -kp/-nkp , -kl/-nkl

(1) -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 オプションをすべて指定した場合は，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 k0r.map にマップ・リストを出力します。

C>lk78k0r k0rmain.rel k0rsub.rel -s -pk0r.map -km

＜ k0r.map の内容＞

```
78K0R Series Linker Vx.xx                        Date : xx xxx xxxx Page : 1

Command : k0rmain.rel k0rsub.rel -s -pk0r.map -km
Para-file :
Out-file : k0rmain.lmf
Map-file : k0r.map
Direc-file :
Directive :

*** Link information ***

    4 output segment(s)
  5FH byte(s) real data
   41 symbol(s) defined

*** Memory map ***

  SPACE = REGULAR

  MEMORY = ROM
  BASE ADDRESS = 00000H   SIZE = 40000H
        OUTPUT  INPUT   INPUT   BASE    SIZE
        SEGMENT SEGMENT MODULE  ADDRESS
        CODE                    00000H  00002H  CSEG    AT
                CODE    SAMPM   00000H  00002H
* gap *                          00002H  000BEH
        ?CSEGOB0                000C0H  00004H  CSEG    OPT_BYTE
        ?CSEG                   000C4H  00059H  CSEG
                ?CSEG   SAMPM   000C4H  00017H
                ?CSEG   SAMPS   000DBH  00042H
* gap *                          0011DH  3FEE3H

  MEMORY = LRAM
  BASE ADDRESS = FCF00H   SIZE = 03100H
        OUTPUT  INPUT   INPUT   BASE    SIZE
        SEGMENT SEGMENT MODULE  ADDRESS
* gap *                          FCF00H  02F20H
        DATA                    FFE20H  00003H  DSEG    AT
                DATA    SAMPM   FFE20H  00003H
* gap *                          FFE23H  000DDH
* gap ( Not Free Area ) *        FFF00H  00100H
```

マップ・リスト

(2) -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 オプションをすべて指定しれた場合は，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 k0r.map にリンク・ディレクティブ・ファイルを出力します。

C>lk78k0r k0rmain.rel k0rsub.rel -s -dk0r.dr -pk0r.map -kd

＜ k0r.map の内容＞

```
78K0R Series Linker Vx.xx                          Date:xx xxx xxxx Page :  1

Command : k0rmain.rel k0rsub.rel -s -dk0r.dr -pk0r.map -kd
Para-file :
Out-file : k0rmain.lmf
Map-file : k0r.map
Direc-file : k0r.dr                                ←ディレクティブ・ファイル名
Directive : MEMORY ROM : ( 0H , 0ED800H )          ←ディレクティブ・ファイルの内容
            MEMORY RAM : ( 0FCF00H , 1100H )
            MEMORY RAM : ( 0FE000H , 1F00H )

*** Link information ***

    6 output segment(s)
  9DH byte(s) real data
   40 symbol(s) defined

*** Memory map ***

  SPACE = REGULAR

  MEMORY = ROM
  BASE ADDRESS = 00000H   SIZE = ED800H
        OUTPUT  INPUT   INPUT   BASE    SIZE
        SEGMENT SEGMENT MODULE  ADDRESS
        CODE                    00000H  00002H  CSEG    AT
          :
```

(3) -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 オプションをすべて指定した場合は，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 k0r.map に，パブリック・シンボル・リストを出力します。

C>lk78k0r k0rmain.rel k0rsub.rel -s -g -pk0r.map -kp

＜ k0r.map の内容＞

```
78K0R Series Linker Vx.xx               Date : xx xxx xxxx Page : 1

Command : k0rmain.rel k0rsub.rel -s -g -pk0r.map -kp
Para-file :
Out-file : k0rmain.lmf
Map-file : k0r.map
Direc-file :
Directive :

*** Link information ***

    6 output segment(s)
  9DH byte(s) real data
   40 symbol(s) defined

*** Memory map ***

  SPACE = REGULAR

  MEMORY = ROM
  BASE ADDRESS = 00000H   SIZE = 40000H
    :

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

78K0R Series Linker Vx.xx               Date : xx xxx xxx Page : 2

*** Public symbol list ***

MODULE  ATTR    VALUE   NAME

SAMPM
        ADDR    00000H  MAIN
        ADDR    000D2H  START
SAMPS
        ADDR    000E9H  CONVAH
        NUM     FFE20H  _@STBEG
        NUM     FCF00H  _@STEND

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

パブリック・シンボル・リスト

(4) -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 k0r.map に，ローカル・シンボル・リストを出力します。

C>lk78k0r k0rmain.rel k0rsub.rel -s -g -pk0r.map -kl

＜ k0r.map の内容＞

```
78K0R Series Linker Vx.xx                Date : xx xxx xxxx Page : 1

Command : k0rmain.rel k0rsub.rel -s -g -pk0r.map -kl
Para-file :
Out-file : k0rmain.lmf
Map-file : k0r.map
Direc-file :
Directive :

*** Link information ***

    6 output segment(s)
  9DH byte(s) real data
   40 symbol(s) defined

*** Memory map ***

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

78K0R Series Linker Vx.xx                 Date : xx xxx xxx Page : 2

*** Local symbol list ***

MODULE  ATTR    VALUE   NAME

SAMPM
        MOD             SAMPM
        DSEG            DATA
        ADDR    FFE20H  HDTSA
        ADDR    FFE21H  STASC
        CSEG            CODE
        CSEG            ?CSEG
SAMPS
        MOD             SAMPS
        CSEG            ?CSEG
        ADDR    0015CH  SASC
        ADDR    00162H  SASC1

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

ローカル・シンボル・リスト

リンク・リスト・ファイル形式指定

-ll, -lf/-nlf

(1) -ll

【記述形式】

```
-ll[ 行数 ]
```

- 省略時解釈

  -ll0（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。

- 行数を省略した場合，0 を指定したものとみなされます。
- 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 k0r.map の 1 頁の行数を 20 行に指定します。

C>lk78k0r k0rmain.rel k0rsub.rel -s -pk0r.map -ll20

＜ k0r.map の内容＞

```
78K0R Series Linker Vx.xx                        Date : xx xxx xxxx Page : 1

Command : k0rmain.rel k0rsub.rel -s -pk0r.map -km -ll20
Para-file :
Out-file : k0rmain.lmf
Map-file : k0r.map
Direc-file :
Directive :

*** Link information ***

    4 output segment(s)
  5FH byte(s) real data
   41 symbol(s) defined

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

78K0R Series Linker Vx.xx                        Date : xx xxx xxxx Page : 2

*** Memory map ***

 SPACE = REGULAR

 MEMORY = ROM
 BASE ADDRESS = 00000H  SIZE = 40000H
        OUTPUT  INPUT   INPUT   BASE    SIZE
        SEGMENT SEGMENT MODULE  ADDRESS
        CODE                    00000H  00002H  CSEG    AT
                CODE    SAMPM   00000H  00002H
* gap *                          00002H  000BEH
        ?CSEGOB0                000C0H  00004H  CSEG    OPT_BYTE
        ?CSEG                   000C4H  00059H  CSEG
                ?CSEG   SAMPM   000C4H  00017H
------------------------------------------------------------------------

78K0R Series Linker Vx.xx                        Date : xx xxx xxxx Page : 3

                ?CSEG   SAMPS   000DBH  00042H
* gap *                          0011DH  3FEE3H

 MEMORY = RAM
 BASE ADDRESS = FCF00H  SIZE = 03100H
        OUTPUT  INPUT   INPUT   BASE    SIZE
        SEGMENT SEGMENT MODULE  ADDRESS
* gap *                          FCF00H  02F20H
        DATA                    FFE20H  00003H  DSEG    AT
                DATA    SAMPM   FFE20H  00003H
* gap *                          FFE23H  000DDH
* gap (Not Free Area) *          FFF00H  00100H


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

(2) -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）を付加することを指定します。
- -nlf オプションは，-lf オプションを無効にします。

【用途】

- 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 k0r.map の最後に改頁コードを付加します。

  C>lk78k0r k0rmain.rel k0rsub.rel -pk0r.map -lf

エラー・リスト・ファイル出力指定

-e/-ne

(1) -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することができます。
- -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 k0r.elk を作成します。

  C>lk78k0r k0rmain.rel k0rsub.rel -dk0r.dr -ek0r.elk

  ディレクティブ・ファイル k0r.dr の内容に誤りがありました。

  ＜エラー・リスト・ファイル k0r.elk の内容＞

```
k0r.dr ( 3 ) : RA78K0R error E3102: Directive syntax error
```

ライブラリ・ファイル指定

-b

(1) -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良くなります。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を指定することはできません。
- ファイル名を省略することはできません。
- ファイル名としてパス名を含めて指定した場合は，指定したパスからライブラリ・ファイルを入力します。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，およびデフォルトのサーチ・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 64 個まで指定することができます。

  注意 PM+ 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については，「第 7 章 ライブラリアン」を参照してください。

【使用例】

- ライブラリ・ファイル k0r.lib を入力します。

  ライブラリ・ファイルには，k0rsub.rel が登録されています。

```
C>lk78k0r k0rmain.rel -bk0r.lib
```

ライブラリ・ファイル読み込みパス指定

-i

(1) -i

【記述形式】

-i *パス名* [, *パス名*] …（複数指定可能）

- 省略時解釈

  環境変数 LIB78K0R で指定したパス

  環境変数 LIB78K0R が指定されていない場合は，カレント・パス

【機能】

- -i オプションは，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】

- -i 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。
- -i は複数指定することができます。また，“ , ”で区切ることにより，一度に複数のパスを指定することができます。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 個まで指定可能です。パス名を複数指定した場合，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，エラーにはなりません。
- パス名を省略した場合，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。

  (1) -i オプションで指定したパス

  (2) 環境変数 LIB78K0R で指定したパス

  (3) 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・ファイルをフォルダ C:¥lib から検索します。

```
C>lk78k0r k0rmain.rel k0rsub.rel -bk0r.lib -iC:¥lib
```

パラメータ・ファイル指定

-f

(1) -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となります。

【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ことを指定します。

【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することができるのは，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・ファイル中に記述可能な文字数に，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 k0r.plk を使用してリンクします。

＜パラメータ・ファイル k0r.plk の内容＞

```
; parameter file
k0rmain.rel k0rsub.rel -ok0r.lmf -pk0r.map -e
-tC:¥tmp -g
```

コマンド行には，次のように入力します。

```
C>lk78k0r -fk0r.plk
```

テンポラリ・ファイル作成パス指定

-t

(1) -t

【記述形式】

-t *パス名*

- 省略時解釈

  環境変数 TMP で指定したパス

  環境変数 TMP を指定していない場合は，カレント・パス

【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することができます。

【説明】

- パス名として，パス以外のものは指定できません。
- パス名を省略することはできません。
- 以前に作成したテンポラリ・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されます。
- 必要とするメモリ・サイズがある間は，テンポラリ・ファイルはメモリに展開されます。
  メモリが足りなくなった時点で，メモリに展開されていたテンポラリ・ファイルの内容がディスクに書き出されます。
  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われます。
- 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。

  (1) -t オプションで指定されたパス

  (2) 環境変数 TMP で設定したパス（-t オプション省略の場合）

  (3) カレント・パス（TMP を設定していない場合）

  なお，(1)，または (2) 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・ファイルをフォルダ C:¥tmp に出力するよう指定します。

```
C>lk78k0r k0rmain.rel k0rsub.rel -tC:¥tmp
```

デバイス・ファイル・サーチ・パス指定

-y

(1) -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，次の順序で調べて決定されます。

  (1) デバイス・ファイル・インストーラで登録されたパス

  (2) LK78K0R の起動されたパス

  (3) カレント・フォルダ

  (4) 環境変数 PATH

【機能】

- -y オプションは，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。

  (1) -y オプションで指定されたパス

  (2) デバイス・ファイル・インストーラで登録されたパス

  (3) LK78K0R の起動されたパス

  (4) カレント・フォルダ

  (5) 環境変数 PATH

【使用例】

- デバイス・ファイルのパスを C:¥78k0r¥dev フォルダに指定します。

```
C>lk78k0r k0rmain.rel k0rsub.rel -yC:¥78k0r¥dev
```

ワーニング・メッセージ出力指定

-w

(1) -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することができます。

【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，「11.3 リンカのエラー・メッセージ」を参照してください。

【使用例】

- 詳細なワーニング・メッセージを出力します。

```
C>lk78k0r k0rmain.rel k0rsub.rel -w2
```

フラッシュ・メモリ内蔵製品のブート領域 ROM プログラムのリンク指定

-zb

(1) -zb

【記述形式】

-zb *アドレス*

- 省略時解釈

  配置範囲の制限はありません。

【機能】

- -zb オプションは，フラッシュ・メモリ領域の先頭アドレスを指定します。

【説明】

- フラッシュ・メモリ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・メモリ領域の先頭アドレスを指定します。
- アドレスを省略した場合は，エラーとなります。
- 設定したアドレス以降には，コードは配置できません。

注意 フラッシュ・メモリ領域セルフ書き換え機能を持たないデバイスでは，本オプションを使用しないでください。

【使用例】

- フラッシュ・メモリ領域の先頭アドレスとして 2000h を指定します。

```
C>lk78k0r k0rmain.rel -zb2000h
```

オンチップ・デバッグ・オプション・バイト指定

-go

(1) -go

【記述形式】

-go *制御値*，*スタート・アドレス* [，*サイズ*]

- 省略時解釈

  オンチップ・デバッグを使用しません。

  C3H 番地はデバイス・ファイルの初期値

【機能】

- -go オプションは，オンチップ・デバッグを使用するか否かを指定します。

【用途】

- オンチップ・デバッグの制御値，スタート・アドレス，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】

- 制御値には，オンチップ・デバッグ動作の制御値を設定してください。

  制御値として指定できない値を指定した場合は，アボート・エラーとなります。

  制御値の詳細については，「ID78K0R の添付文書」を参照してください。

- スタート・アドレスには，オンチップ・デバッグ・プログラムの配置スタート・アドレスを設定してください。

  スタート・アドレスとして指定可能な値の範囲は，次のとおりです。

  0 ≦ スタート・アドレス ≦ 0FFFFFH

  スタート・アドレスを省略した場合，D8H を指定したものとみなされます。

  スタート・アドレスの詳細については，「ID78K0R の添付文書」を参照してください。

- サイズには，オンチップ・デバッグ・プログラムのサイズを設定してください。

  サイズとして指定可能な値の範囲は，次のとおりです。

  88 ≦ サイズ ≦ 1,024

- プログラム・サイズを省略した場合，88 バイトを指定したものとみなされます。

  プログラム・サイズの詳細については，「ID78K0R の添付文書」を参照してください。

- 制御値，スタート・アドレス，サイズに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。

- -go オプションが指定された場合，C3H 番地には，制御値が配置されます。
  2H，3H，CEH ～ D7H 番地と，-go オプションで指定したスタート・アドレスから指定されたプログラム・サイズ分の領域は，FFH で充てんされるため，セグメントを配置することはできません。
  なお，C0 ～ C2H 番地は，-gb オプションにより，ユーザ・オプション・バイトの領域として確保されます。
- -go オプションが指定されなかった場合でも，C3H 番地は予約領域とするため，ユーザ・コードを配置することはできません。
- C3H 番地の制御値は，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，C0H 番地からのユーザ・オプション・バイトとあわせて 4 バイトで定義してください。

```
[ 任意のセグメント名 ]      CSEG    OPT_BYTE
                            DB      11H
                            DB      22H
                            DB      33H
                            DB      44H
```

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，本オプションが優先されます。

【使用例】

- C3H 番地に，制御値として 0FFH を埋め込みます。
- スタート・アドレス（12345H 番地）から 256 バイトをプログラム領域として確保します。

```
C>lk78k0r k0rmain.rel -go0FFH,12345H,256
```

セキュリティ ID 指定

-gi

(1) -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 を設定しません。

【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 を補てんします。
- セキュリティID は，C4H ～ CDH に設定されます。セキュリティID を設定した場合，C4H ～ CDH 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，本オプションが優先されます。

【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“112233445566778899aah”を指定します。

```
C>lk78k0r k0rmain.rel -gi112233445566778899aah
```

ユーザ・オプション・バイト指定

-gb

(1) -gb

【記述形式】

-gb *ユーザ・オプション・バイト値*

- 省略時解釈

  デバイス・ファイルの初期値

【機能】

- -gb オプションは，ユーザ・オプション・バイトに設定する値を指定します。

【用途】

- ユーザ・オプション・バイトの値を設定したいときに，-gb オプションで指定します。

【説明】

- ユーザ・オプション・バイト値として指定可能な値の範囲は，次のとおりです。

  0 ≦ ユーザ・オプション・バイト ≦ 0FFFFFFH

- ユーザ・オプション・バイト値として指定することができない値を指定した場合は，アボート・エラーとなります。
- “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は，アボート・エラーとなります。
- ユーザ・オプション・バイトは，C0H ～ C2H 番地に設定されます。
- -gb オプションが指定されなかった場合は，C0 ～ C2H 番地は予約領域とするため，ユーザ・コードを配置することはできません。
- 3 バイト以内で指定してください。3 バイトに満たない場合には，上位ビットに 0 を補てんします。
- C0H ～ C2H 番地のユーザ・オプション・バイト値は，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，C3H 番地の制御値とあわせて 4 バイトで定義してください。

```
[ 任意のセグメント名 ]      CSEG    OPT_BYTE
                            DB      11H
                            DB      22H
                            DB      33H
                            DB      44H
```

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，本オプションが優先されます。

【使用例】

- ユーザ・オプション・バイト値として C0H 番地に A1H，C1H 番地に B2H，C2H 番地に C3H を指定します。

```
C>lk78k0r k0rmain.rel -gb0A1B2C3H
```

ミラー領域指定

-mi

(1) -mi

【記述形式】

```
-mi0，または -mi1
```

- 省略時解釈

  -mi0

【機能】

- -mi オプションは，ミラー元のセグメントの配置先を指定します。

【用途】

- ミラー元のセグメントの配置先を指定したいときに，-mi オプションで指定します。

【説明】

- CSEG MIRRORP の再配置属性のセグメントの配置先をリンカで指定します。
- -mi0 指定時には，MAA に 0 を設定したときのミラー領域にセグメントを配置，-mi1 指定時には，MAA に 1 を設定したときのミラー領域にセグメントを配置します。
  ミラー領域の詳細については，デバイスのユーザーズ・マニュアルを参照してください。
- パブリック・シンボル“_@MAA”を生成します。このシンボルは，-mi0 指定時には“0”，-mi1 指定時には“1”を値として持つ NUMBER 属性のシンボルです。

【使用例】

- MAA に 1 を設定したときのミラー領域にセグメントを配置します。

  C>lk78k0r k0rmain.rel -mi1

  ミラー領域が F1000H 番地～のデバイスの場合，CSEG MIRRORP のセグメントは 11000H 番地から配置されます。

  ＜コンパイラが提供するスタープアップ・ルーチンでの使用例＞

```
MOVW     PMC , #_@MAA
```

  この場合，PMC には“1”が格納されます。

64K バイト境界配置指定

-ccza

(1) -ccza

【記述形式】

```
-ccza
-nccza
```

- 省略時解釈

  -ccza（入力ファイルがアセンブラ出力ファイルのみの場合）

  -nccza（入力ファイルにコンパイラ出力ファイルがある場合）

【機能】

- -ccza オプションは，各 64K バイト領域の境界の最後の 1 バイト（xFFFFH 注）にセグメントの配置を行うかどうかを指定します。

  注 x：0H - EH

【用途】

- 各 64K バイト領域の境界の最後の 1 バイトにセグメントの配置を行うかどうかを指定したいときに，-ccza オプションで指定します。

【説明】

- アセンブラでのみ開発する場合には，自動的に各 64K バイト領域の境界の最後の 1 バイトにも配置を行うため，本オプションを指定する必要はありません。
- コンパイラ出力のオブジェクト・モジュール・ファイルをリンカに入力した場合には，自動的に -nccza と指定されたものとみなし，各 64K バイト領域の境界の最後の 1 バイトに配置を行いません。
- コンパイラにて -za オプションを指定した場合には，各 64K バイト領域の境界の最後の 1 バイトへの配置が可能となるため，-ccza を指定してください。
- 各 64K バイト領域の境界の最後の 1 バイトへの配置についての詳細は，「CC78K0R C コンパイラ 言語編」のユーザーズ・マニュアルを参照してください。

ヘルプ指定

--

(1) --

【記述形式】

```
--
```

- 省略時解釈

  表示しません。

【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，ほかのリンカ・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読む場合は，リターン・キーを押下してください。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・キーを押下してください。

注意 本オプションは，PM+ 上では指定することはできません。

PM+ 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>lk78k0r --

```
78K0R Series Linker Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

usage : lk78k0r [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).
-godata,address[,size] : Change On-Chip Debug Option Bytes, start address,
size (size = 88 to 1024).
-giid            : Set Security ID.
-gbdata          : Set User Option Bytes.
-mi[0 or 1]      : Select allocation for MIRRORP segment.
-ccza/-nccza     : Allocate user code to nFFFFH/Not.
--               : Show this message.

DEFAULT ASSIGNMENT : -o -p -ne -km -kd -nkp -nkl -ll0 -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 : ( 0FEF00H , 100H )
           MERGE CSEG1 : = ROM
           MERGE DSEG1 : AT ( 0FF000H )
```

5.7 PM+ でのオプション設定

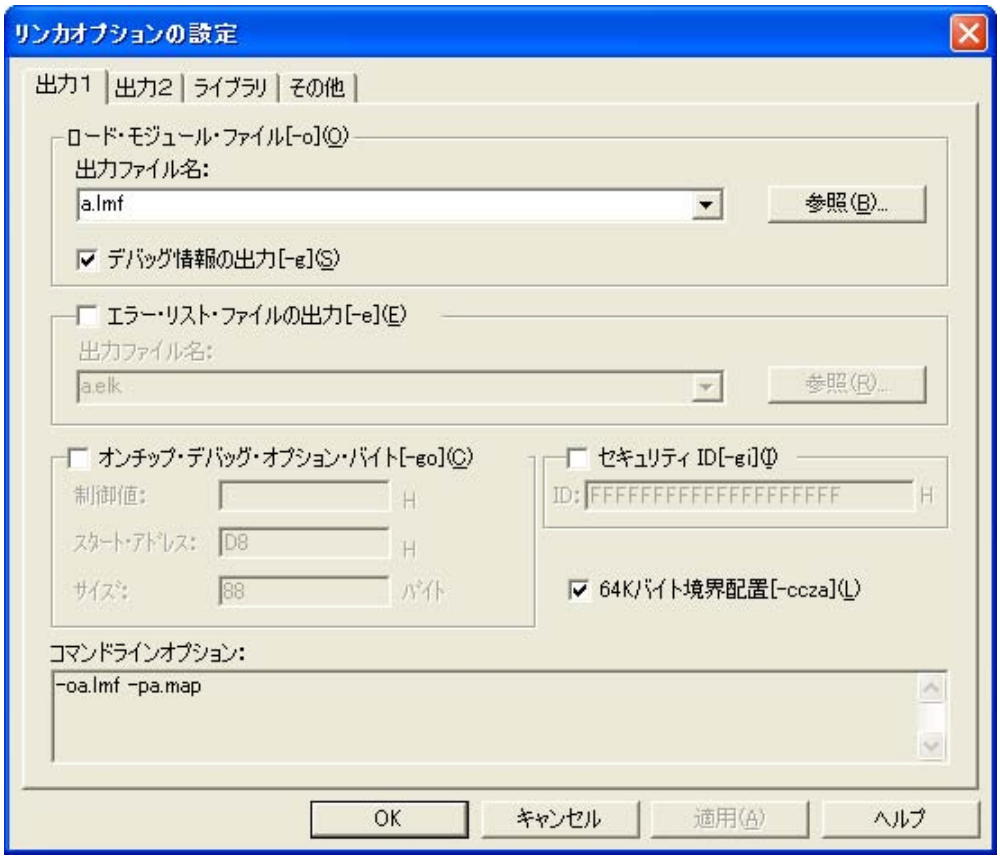
PM+ からリンカ・オプションを設定する方法について説明します。

5.7.1 オプションの設定方法

PM+ の [ツール (T)] メニュー→ [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・オプションを設定することができます。

図 5-2 [リンカオプションの設定] ダイアログ

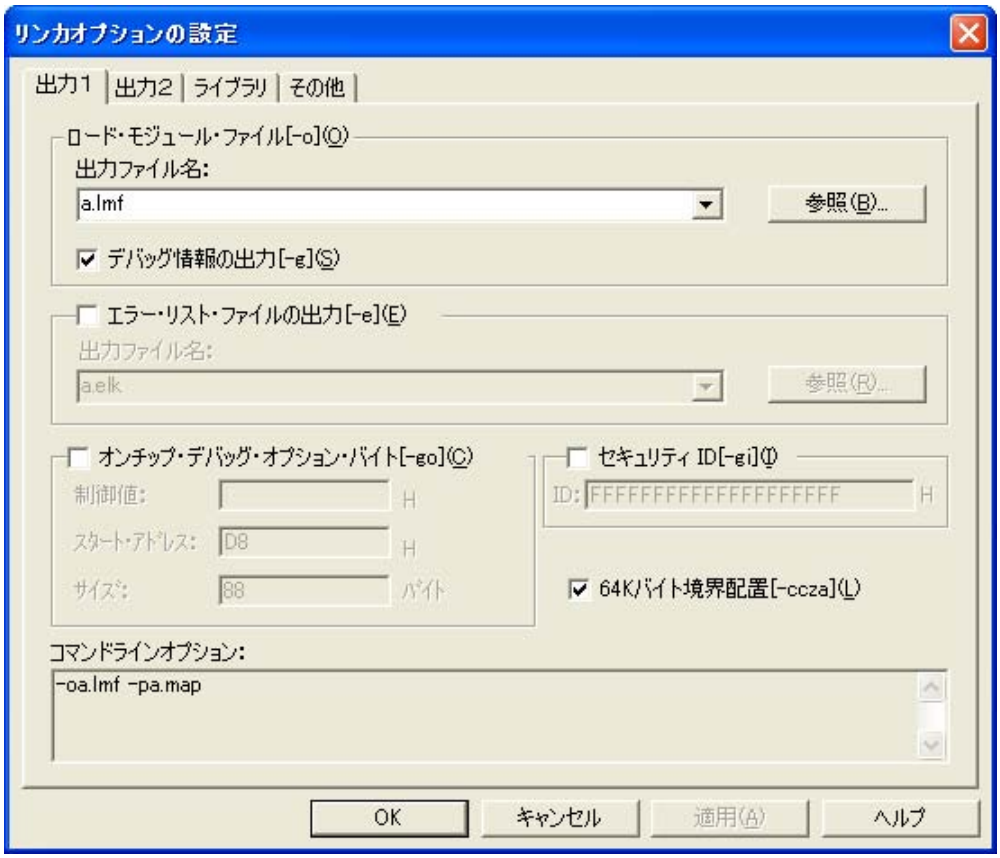


5.7.2 ダイアログの説明

[リンカオプションの設定] ダイアログの各タブについて，次に説明します。

(1) [出力 1] タブ

図 5-3 [リンカオプションの設定] ダイアログ（[出力 1] タブ選択時）



- 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。

初期値として a.lmf が設定されます。

- デバッグ情報の出力 [-g](S)

ロード・モジュール・ファイル中にデバッグ情報 (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。

- オンチップ・デバッグ・オプション・バイト [-go](C)

オンチップ・デバッグ・オプション・バイトを使用する場合に，チェックします。

制御値：

オンチップ・デバッグ動作の制御値を指定します。

制御値の詳細については，「ID78K0R の添付文書」を参照してください。

スタート・アドレス：

オンチップ・デバッグ・プログラムの配置スタート・アドレスを指定します。

指定可能な値の範囲は，次のとおりです。

0 ≦ スタート・アドレス ≦ 0FFFFFH

省略した場合，D8H を指定したものとみなされます。

サイズ：

オンチップ・デバッグ・プログラムのサイズを指定します。

指定可能な値の範囲は，次のとおりです。

88 ≦ サイズ ≦ 1024

省略した場合，88 バイトを指定したものとみなされます。

注意 オンチップ・デバッグ機能を持たないデバイスでは，本オプションを指定することはできません。

- 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，本オプションを指定することはできません。

- 64K バイト境界配置 [-ccza](L)

64K バイト境界の最後の 1 バイト（xFFFFH[注]）に配置を行う場合にチェックします。

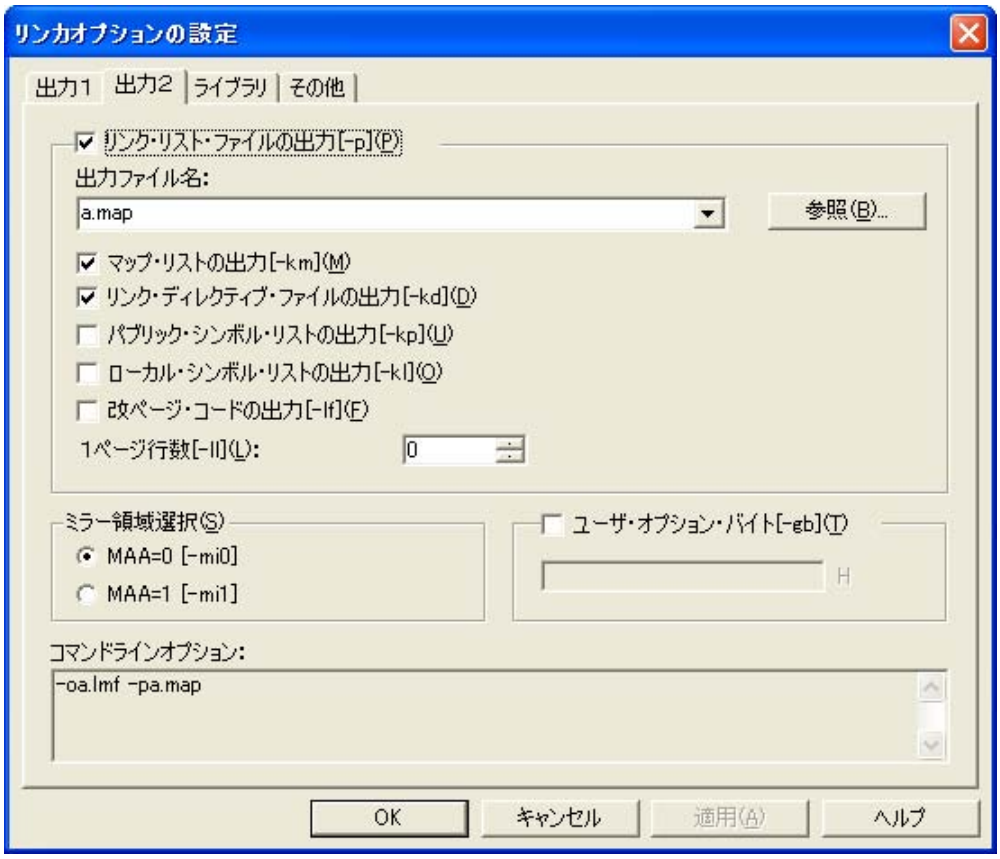
注 x：0H - EH

- 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。

(2) [出力 2] タブ

図 5-4 [リンカオプションの設定] ダイアログ（[出力 2] タブ選択時）



- 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。

  初期値として a.map が設定されます。

- 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。

  指定可能な行数は，0，および 20 ～ 32767 までの範囲です。

- ミラー領域選択 (S)

  RAM 空間にミラーされるセグメントを配置する領域を選択します。

  MAA=0 [-mi0] ： MAA に 0 を設定したときにミラーされる領域に配置します。

  MAA=1 [-mi1] ： MAA に 1 を設定したときにミラーされる領域に配置します。

  ミラー領域の詳細については，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。

- ユーザ・オプション・バイト [-gb](T)

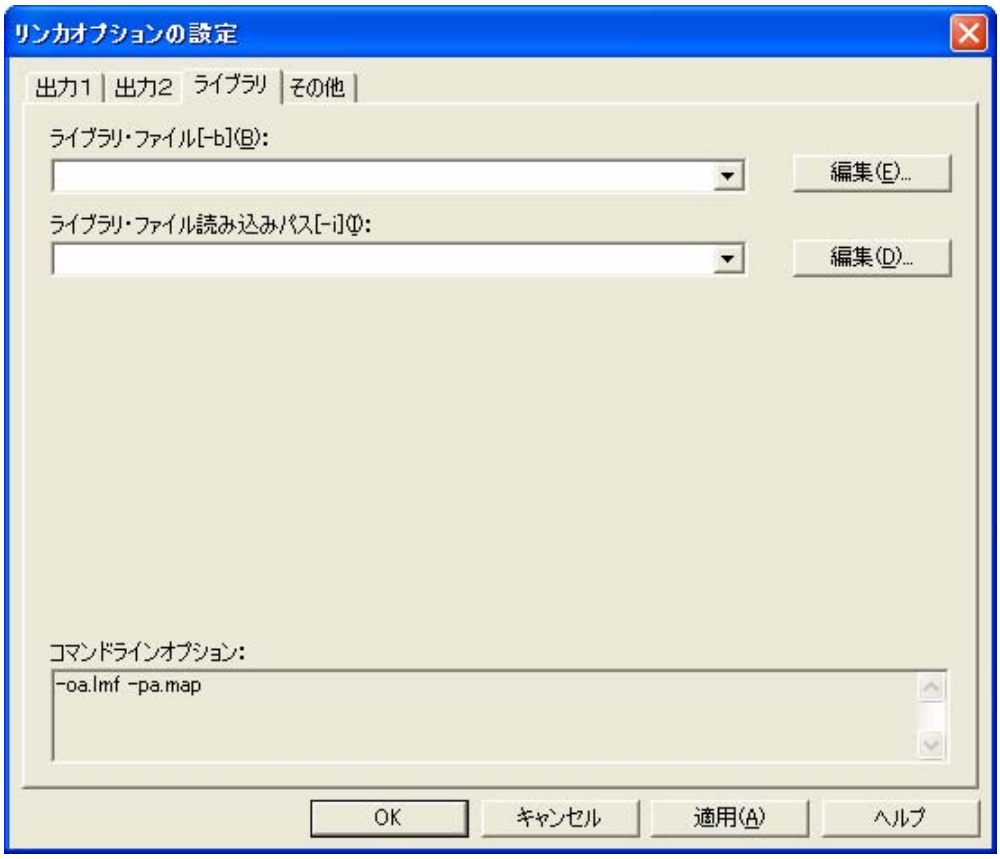
  ユーザ・オプション・バイトに値を設定する場合にチェックし，値を指定します。

- 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。

(3) [ライブラリ] タブ

図 5-5 [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）



- ライブラリ・ファイル [-b](B)

  [編集 (E)...] ボタン（[オプションの編集] ダイアログをオープン），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。

  ファイル名は，カンマで区切って 64 個まで指定することができます。

- ライブラリ・ファイル読み込みパス [-i](I)

  [編集 (D)...] ボタン（[オプションの編集] ダイアログをオープン），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。

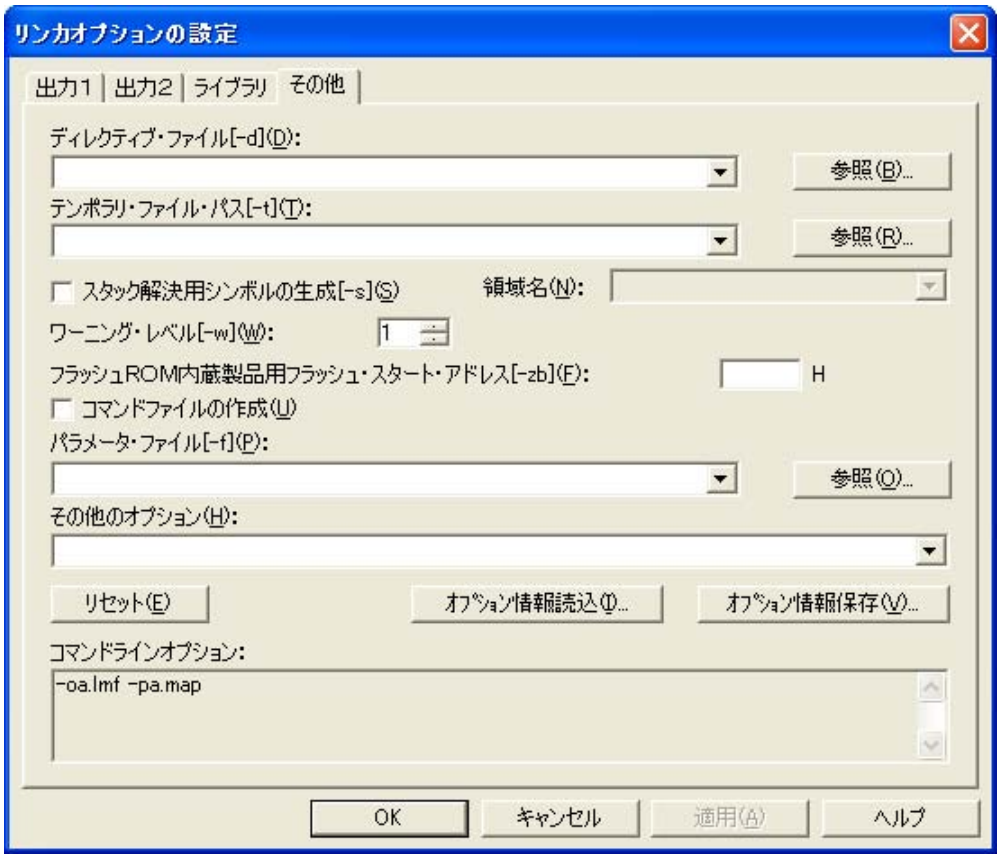
  パス名は，カンマで区切って 64 個まで指定することができます。

- 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) [その他] タブ

図 5-6 [リンカオプションの設定] ダイアログ（[その他] タブ選択時）



- 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，またはデフォルトで定義されているメモリ領域名を指定します。
入力可能な文字数は，256 文字までです。

- 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・メモリ内蔵製品のブート領域スタート・アドレスを指定します。
指定可能な値の範囲は，次のとおりです。

0H ≦ zb ≦ 0FFFFH

注意 フラッシュ・メモリ領域セルフ書き換え機能を持たないデバイスでは，本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+ 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)...

[オプション情報の読込み] 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

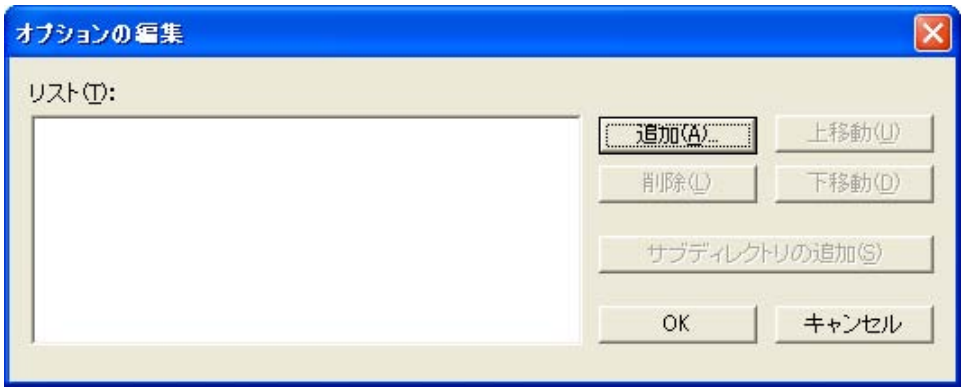
このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

5.7.3 [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 5-7 [オプションの編集] ダイアログ



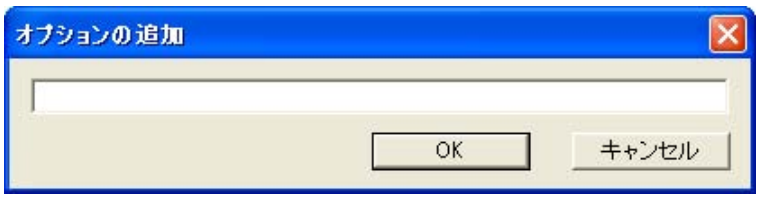
- ［追加 (A)］ボタン

  リストの項目を追加します。

  ファイルやフォルダ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。

図 5-8 [オプションの追加] ダイアログ



- ［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することができます。

  [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)

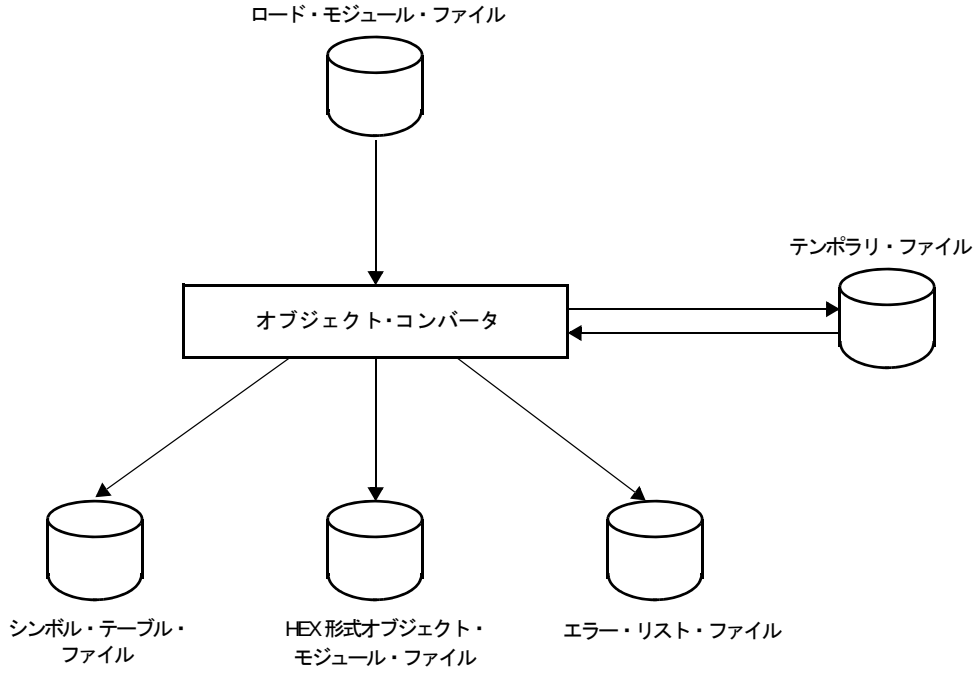
第 6 章 オブジェクト・コンバータ

オブジェクト・コンバータは，RA78K0R 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・デバッ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。

図 6-1 オブジェクト・コンバータの入出力ファイル



6.1 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 6-1 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	デフォルト・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 - リンカの出力したファイル	.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（ユーザ作成ファイル）	.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 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	.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	.eoc

6.2 オブジェクト・コンバータの機能

6.2.1 フラッシュ・メモリの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・メモリのセルフ書き換えモード使用時に，フラッシュ・メモリ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・ファイルを生成することができます。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 6-2 -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

6.2.2 HEX 形式オブジェクト・モジュール・ファイル

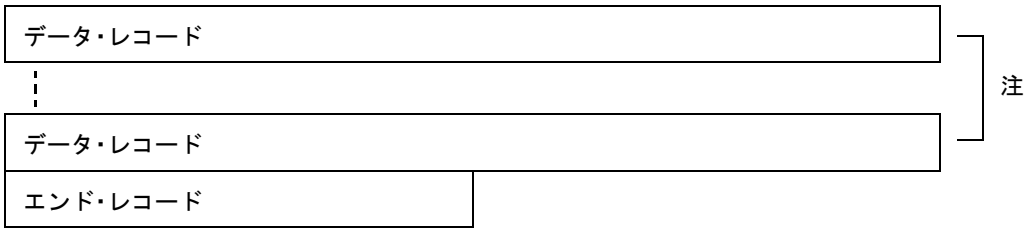
オブジェクト・コンバータの出力する HEX 形式オブジェクト・モジュール・ファイルは，PROM プログラマやデバッガなどの HEX ローダに入力可能です。

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
```

(1) 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 6-2 インテル標準形式



注 データ・レコードは繰り返されます。

(a) データ・レコード

:	02	0000	00	8000	7E
(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。

(b) エンド・レコード

```
 :       00       0000    01       FF
(i)     (ii)     (iii)  (iv)     (v)
```

(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

00 で固定です。

(iii) 0000 で固定です。

(iv) レコード・タイプ

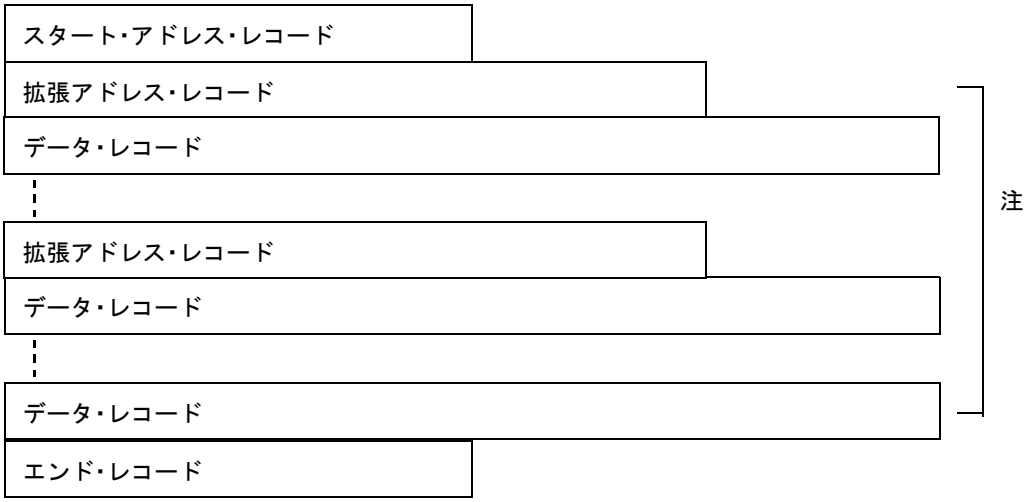
01 で固定です。

(v) チェック・サム

FF で固定です。

(2) 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 6-3 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。

(a) 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。

(b) データ・レコード

```
 :       02      0000    00      80000   7E
(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。

(c) 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

レコードの始まりを示します。

(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。

(d) 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。

(3) 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。

- データ・ブロック
- 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）。
- 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。

次に，共通なヘッダ・フィールドの形式を示します。

表 6-3 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。 キャラクタの値を表 6-4 に示します。

表 6-4 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	0 - 9
A - Z	10 - 35
$	36
%	37
.（ピリオド）	38
_（アンダースコア）	39
a - z	40 - 65

(a) データ・ブロック

データ・ブロックのフォーマットを，次に示します。

表 6-5 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
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

(b) 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，次に示します。

表 6-6 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

(c) 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。

表 6-7 拡張テックのシンボル・ブロックの属性

項目	属性
シンボル自体	1 - 16 個の英大小文字，数字，ピリオド，およびアンダースコア。 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，次に示します。

表 6-8 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できます。 表 6-9 にこのフォーマットを示します。
シンボル定義	各 5 - 35	表 6-10 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，次に示します。

表 6-9 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 6-10 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。

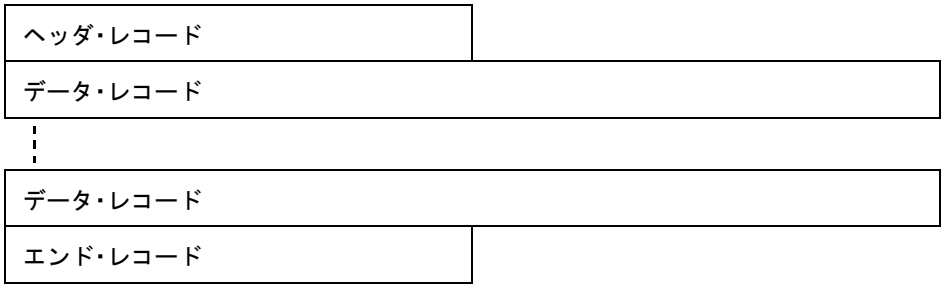
(4) モトローラ S タイプ形式

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 6-4 のようになります。レコードの種類を，次に示します。

表 6-11 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 6-4 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，次に示します。

表 6-12 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 6-13 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

(a) 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

(b) 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

(c) 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

(d) 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

(e) 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

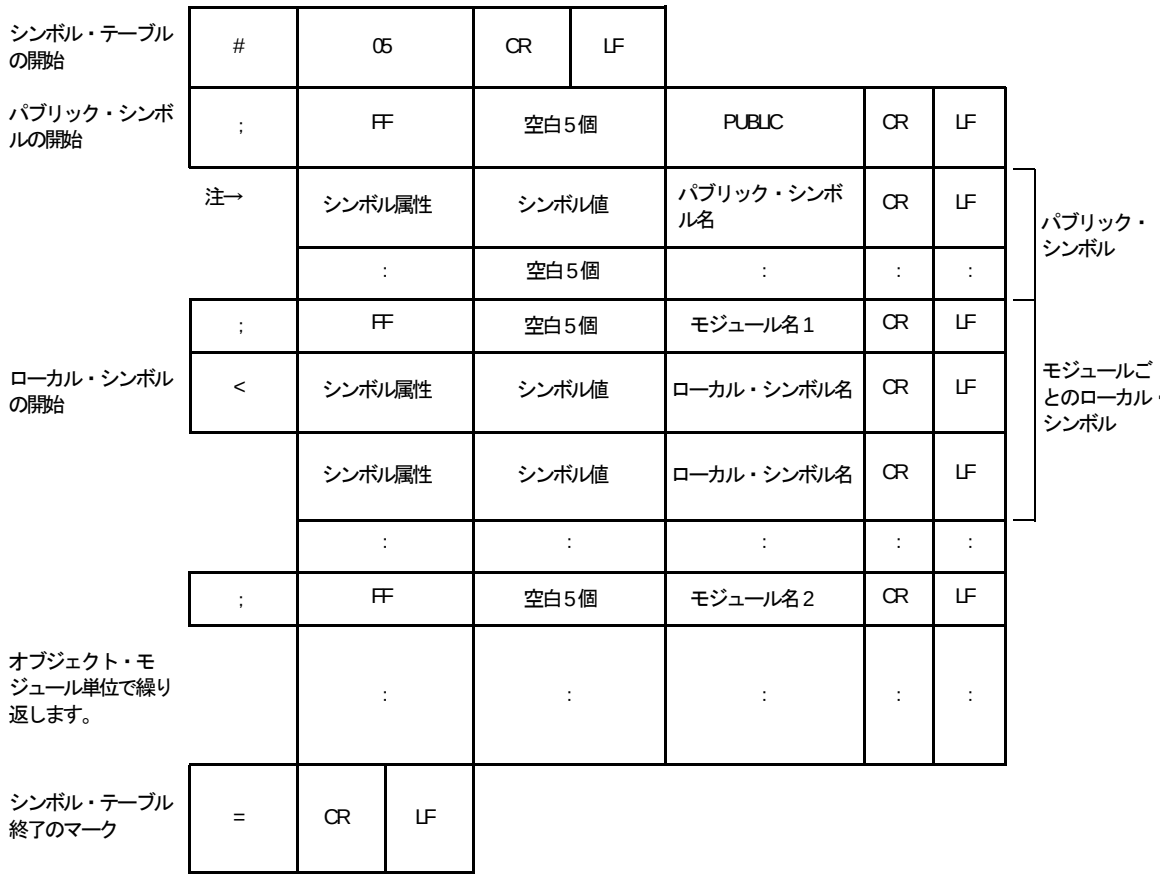
6.2.3 シンボル・テーブル・ファイル

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。

```
#05
; FF    PUBLIC
01000E9CONVAH
0100000MAIN
01000D2START
00FFE20_@STBEG
00FCF00_@STEND
; FF    SAMPM
<02FFE20HDTSA
02FFE21STASC
; FF    SAMPS
<010015CSASC
0100162SASC1
=
```

図 6-5 シンボル・テーブル・ファイルのフォーマット



注 シンボル属性は，次の値をとります。

シンボル値のフォーマットは，図 6-6 を参照してください。

値	シンボル属性
00	EQU で定義した定数
01	コード・セグメント内のラベル
02	データ・セグメント内のラベル
03	ビット・シンボル
FF	モジュール名

図 6-6 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）

6.3 オブジェクト・コンバータの起動

6.3.1 オブジェクト・コンバータの起動方法

オブジェクト・コンバータの起動には，2 つの方法があります。

(1) コマンド行での起動

```
 X>[ パス名 ]oc78k0r[ △オプション] … ロード・モジュール・ファイル名 [ △オプション] … [ △ ]
 ↑    ↑       ↑          ↑                          ↑                  ↑
(1)  (2)     (3)        (4)                        (5)                (4)
```

(1) カレント・ドライブ名

(2) カレント・フォルダ名

(3) オブジェクト・コンバータのコマンド・ファイル名

(4) オブジェクト・コンバータに対して動作の詳細を指定

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，それぞれのオプション間を空白で区切ってください。なお，オブジェクト・コンバータ・オプションに大文字，小文字の区別はありません。オブジェクト・コンバータ・オプションの詳細については，「6.4 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。

(5) コンバートするロード・モジュール・ファイル名

【例】

- HEX 形式オブジェクト・モジュール・ファイル samle.hex を出力します。

```
C>oc78k0r k0r.lmf -osamle.hex
```

(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は，次のようになります。

```
 X>oc78k0r[ △ロード・モジュール・ファイル ] △ -f パラメータ・ファイル名
  ↑     ↑                                  ↑          ↑
(1)   (2)                                (3)        (4)
```

(1) カレント・ドライブ名

(2) カレント・フォルダ名

(3) パラメータ・ファイル指定オプション

(4) 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 ] … [ △ ] △ ]] …
```

- 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 つだけ指定することができます。
- 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。パラメータ・ファイルについての詳細は，「3.4 パラメータ・ファイルの使用」を参照してください。

【例】

1. パラメータ・ファイル k0r.poc をエディタで作成します。

```
; parameter file
k0r.lmf -osample.hex
-ssample.sym -r
```

2. パラメータ・ファイル k0r.poc を使用してオブジェクト・コンバータを起動します。

```
C>oc78k0r -fk0r.poc
```

6.3.2 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。

```
78K0R Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation 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 : uPD78xxx
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 : uPD78xxx
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】

＜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>oc78k0r sample.lmf

```
78K0R Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

RA78K0R 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。

＜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>oc78k0r k0r.lmf -a

```
78K0R Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

RA78K0R error F4018 : Option is not recognized '-a'
Please enter 'OC78K0R --' , 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・メッセージの原因については，「第 11 章 エラー・メッセージ」で調べて対処してください。

6.4 オブジェクト・コンバータ・オプション

6.4.1 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・オプションの分類と説明を示します。

表 6-14 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（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・メモリ内蔵製品用ファイル分割出力指定	-zf	ブート領域とそれ以外の領域を別々のファイルに分割出力します。
ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

HEX 形式オブジェクト・モジュール・ファイル出力指定

-o/-no

(1) -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

【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・フォルダに 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。

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

【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>oc78k0r k0r.lmf -osample.hex
```

シンボル・テーブル・ファイル出力指定

-s/-ns

(1) -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

【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・フォルダにシンボル・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。

【使用例】

- シンボル・テーブル・ファイル sample.sym を出力します。

  C>oc78k0r k0r.lmf -ssample.sym

オブジェクト・アドレス順ソート指定

-r/-nr

(1) -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>oc78k0r k0r.lmf -r
```

オブジェクト充てん値指定

-u/-nu

(1) -u/-nu

【記述形式】

```
-u 充てん値 [,[ スタート・アドレス ], 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可能な値の範囲は，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・アドレスには，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。
  指定可能な値の範囲は，次のとおりです。

  0H ≦ スタート・アドレス ≦ 0FFEFFH

  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。また，スタート・アドレスを省略した場合は，0 を指定したものとみなされます。

- 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。
  指定可能な値の範囲は，次のとおりです。

  1H ≦ サイズ ≦ 0FFF00H

  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。また，スタート・アドレスを指定している場合は，サイズを省略することはできません。

- スタート・アドレス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・アドレス，サイズの指定形式とその解釈は，次のようになります。

  (1) -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

  (2) -u *充てん値* , *サイズ*

  0 番地から，サイズ - 1 番地まで

  (3) -u *充てん値* , *スタート・アドレス* , *サイズ*

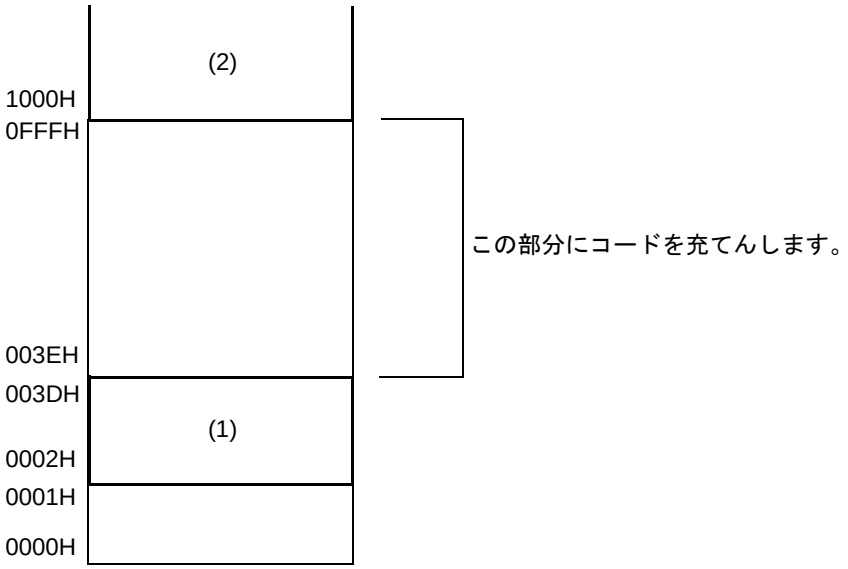
  スタート・アドレスから，スタート・アドレス + サイズ - 1 番地まで

【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>oc78k0r k0r.lmf -u00h,003eh,0fc2h
```

エラー・リスト・ファイル出力指定

-e/-ne

(1) -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することができます。
- -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 k0r.eoc を作成します。

```
C>oc78k0r k0r.lmf -ek0r.eoc
```

パラメータ・ファイル指定

-f

(1) -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となります。

【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ことを指定をします。

【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することができるのは，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・ファイル中に記述可能な文字数に，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 78k0r.poc 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 78k0r.poc の内容＞

```
; parameter file
k0r.lmf -osample.hex
-ssample.sym -r
```

コマンド行には，次のように入力します。

C>oc78k0r k0r.lmf -f78k0r.poc

HEX 形式指定

-ki/-kie/-kt/-km/-kme

(1) -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。

オプション	出力形式	範囲
-ki	インテル標準 HEX 形式	0H - FFFFH （64K バイトまで）
-kie	インテル拡張 HEX 形式	0H - FFFFFH （1M バイトまで）
-kt	拡張テック形式	0H - FFFFFFFFH （4G 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 （16M 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 （4G バイトまで）

【使用例】

- 出力する HEX 形式オブジェクト・モジュール・ファイルをモトローラ S タイプ形式（スタンダード・アドレス）に指定します。

```
C>oc78k0r k0r.lmf -km
```

デバイス・ファイル・サーチ・パス指定

-y

(1) -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，次の順序で調べて決定されます。

  (1) デバイス・ファイル・インストーラで登録されたパス

  (2) OC78K0R の起動されたパス

  (3) カレント・フォルダ

  (4) 環境変数 PATH

【機能】

- -y オプションは，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を指定した場合，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を省略した場合，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。

  (1) -y オプションで指定されたパス

  (2) デバイス・ファイル・インストーラで登録されたパス

  (3) OC78K0R の起動されたパス

  (4) カレント・フォルダ

  (5) 環境変数 PATH

【使用例】

- デバイス・ファイルのパスを C:¥78k0r¥dev フォルダに指定します。

```
C>oc78k0r k0r.lmf -yC:¥78k0r¥dev
```

フラッシュ・メモリ内蔵製品用ファイル分割出力指定

-zf

(1) -zf

【記述形式】

```
-zf
```

- 省略時解釈

  分割出力しません。

【機能】

- -zf オプションは，ブート領域とそれ以外の領域を別々のファイルに分割出力します。

【説明】

- フラッシュ・メモリ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 オプションを指定した場合，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・メモリ領域セルフ書き換え機能を持たないデバイスでは，本オプションを使用しないでください。

【使用例】

- ブート領域とそれ以外の領域を HEX フォーマット・ファイル k0r.hxb，k0r.hxf に分割出力します。

```
C>oc78k0r k0r.lmf -zf
```

ヘルプ指定

--

(1) --

【記述形式】

```
--
```

- 省略時解釈

  表示しません。

【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，ほかのオブジェクト・コンバータ・オプションはすべて無効となります。

注意 本オプションは，PM+ 上では指定することはできません。
PM+ 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>oc78k0r --

```
78K0R Series Object Converter Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

usage : oc78k0r [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.
-e[file]/-ne : Create the error list file [with the 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
```

6.5 PM+ でのオプション設定

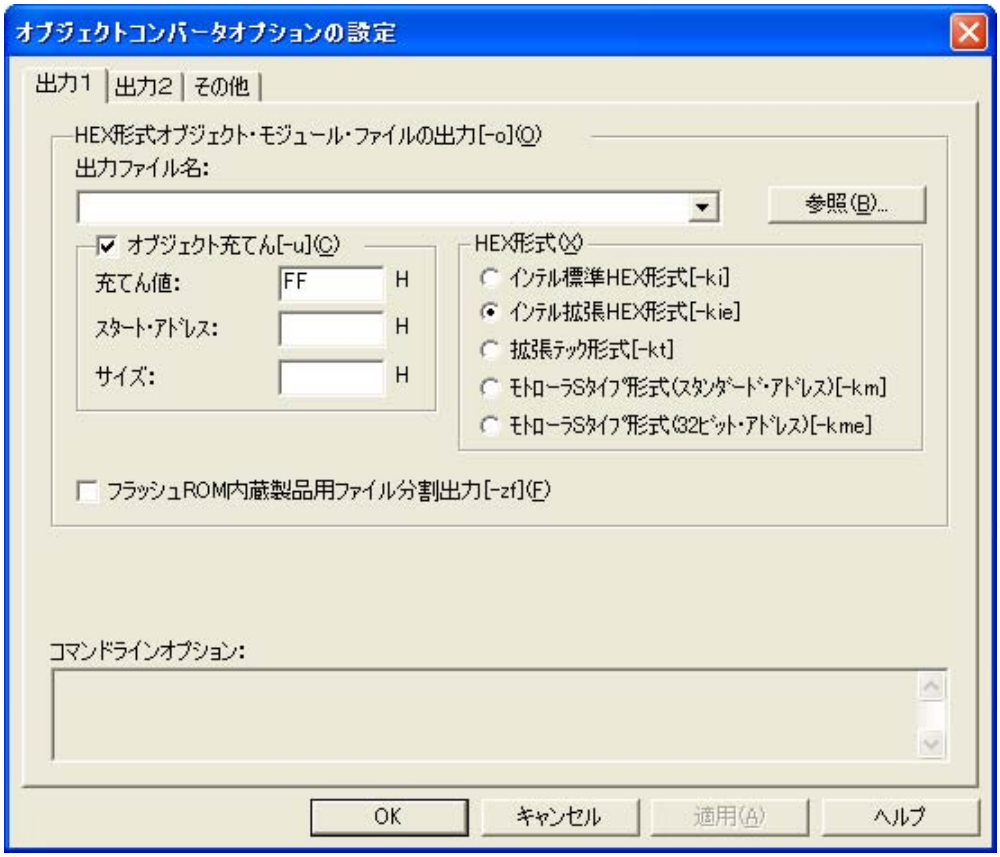
PM+ からオブジェクトコンバータ・オプションを設定する方法について説明します。

6.5.1 オプションの設定方法

PM+ の [ツール (T)] メニュー→ [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・オプションを設定することができます。

図 6-7 [オブジェクトコンバータオプションの設定] ダイアログ

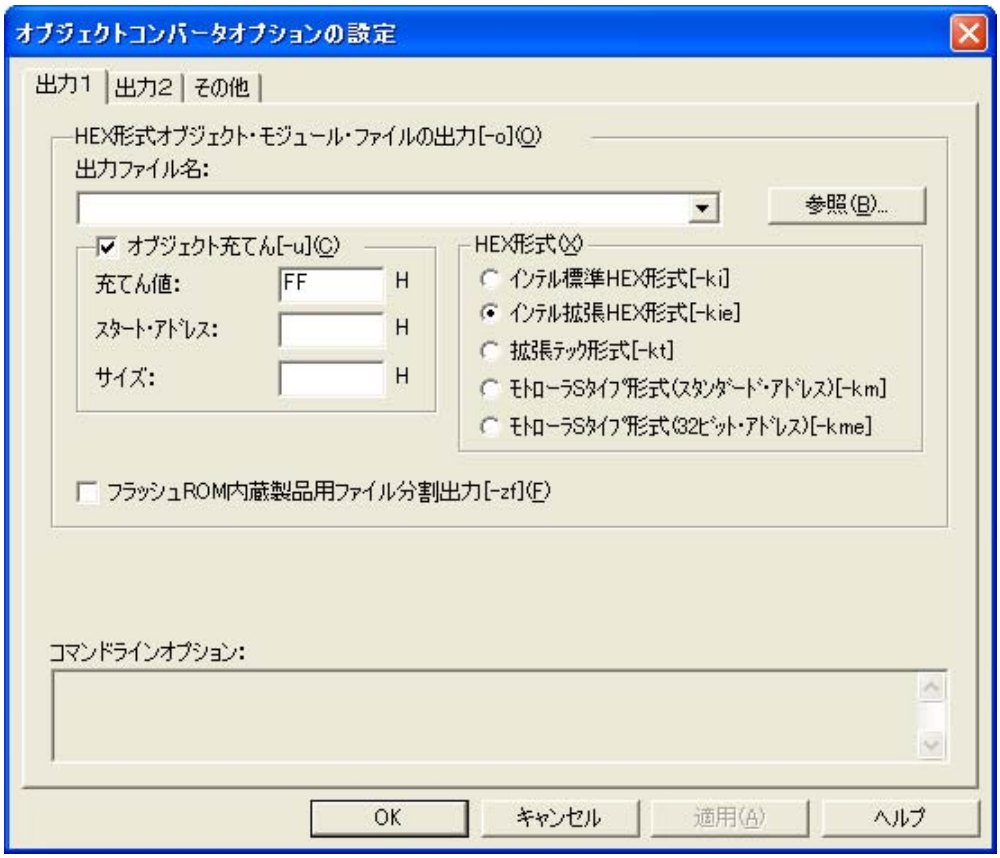


6.5.2 ダイアログの説明

[オブジェクトコンバータオプションの設定] ダイアログの各タブについて，次に説明します。

(1) [出力 1] タブ

図 6-8 [オブジェクトコンバータオプションの設定] ダイアログ（[出力 1] タブ選択時）



- 出力ファイル名

［参照 (B)...］ボタン，または直接入力により，HEX 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，あらかじめコードを書き込む場合に，チェックします。

充てん値：

充てんする値を指定します。

指定可能な値の範囲は，次のとおりです。

0H ≦ 充てん値 ≦ 0FFH

スタート・アドレス：

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。

指定可能な値の範囲は，次のとおりです。

0H ≦ スタート・アドレス ≦ 0FFEFFH

サイズ：

充てんを行うアドレス領域のサイズを指定します。

指定可能な値の範囲は，次のとおりです。

1H ≦ サイズ ≦ 0FFF00H

- HEX 形式 (X)

出力するオブジェクトの HEX 形式（インテル標準 HEX 形式 [-ki]，インテル拡張 HEX 形式 [-kie]，拡張テック形式 [-kt]，モトローラ S タイプ形式（スタンダード・アドレス）[-km]，モトローラ S タイプ形式（32 ビット・アドレス）[-kme]）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，本オプションを指定することはできません。

- 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・メモリ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。

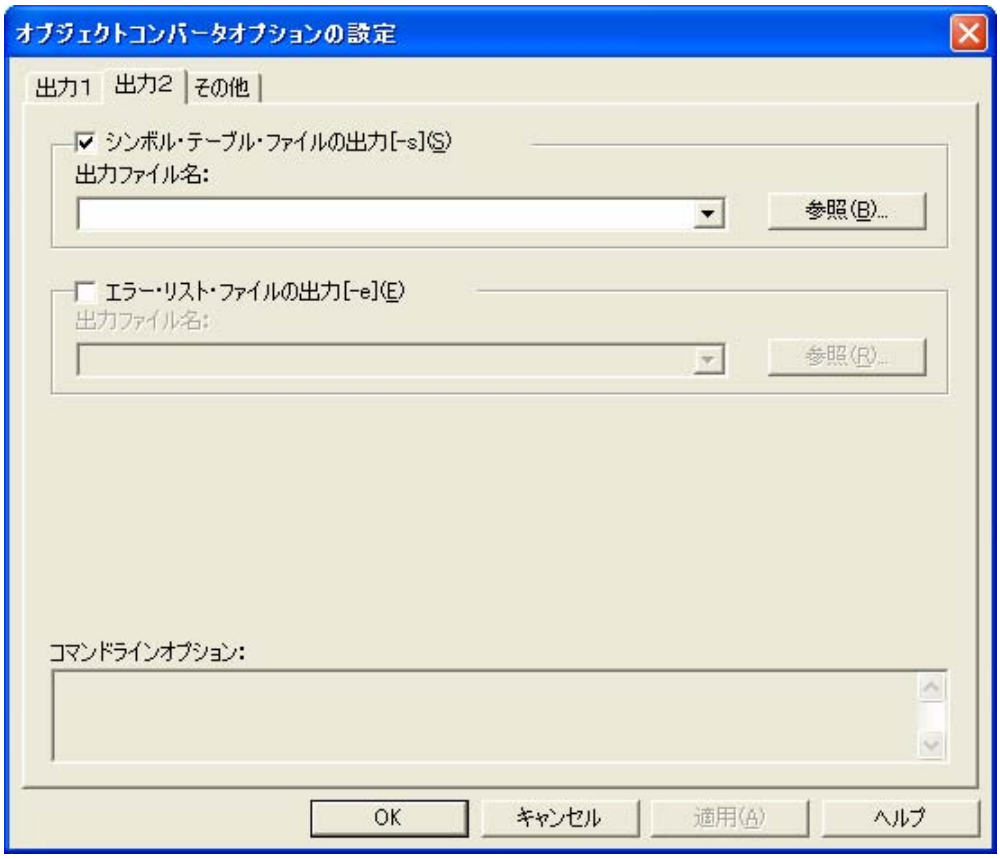
注意 フラッシュ・メモリ領域セルフ書き換え機能を持たないデバイスでは，本オプションを指定しないでください。

- 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。

(2) [出力 2] タブ

図 6-9 [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）



- 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：

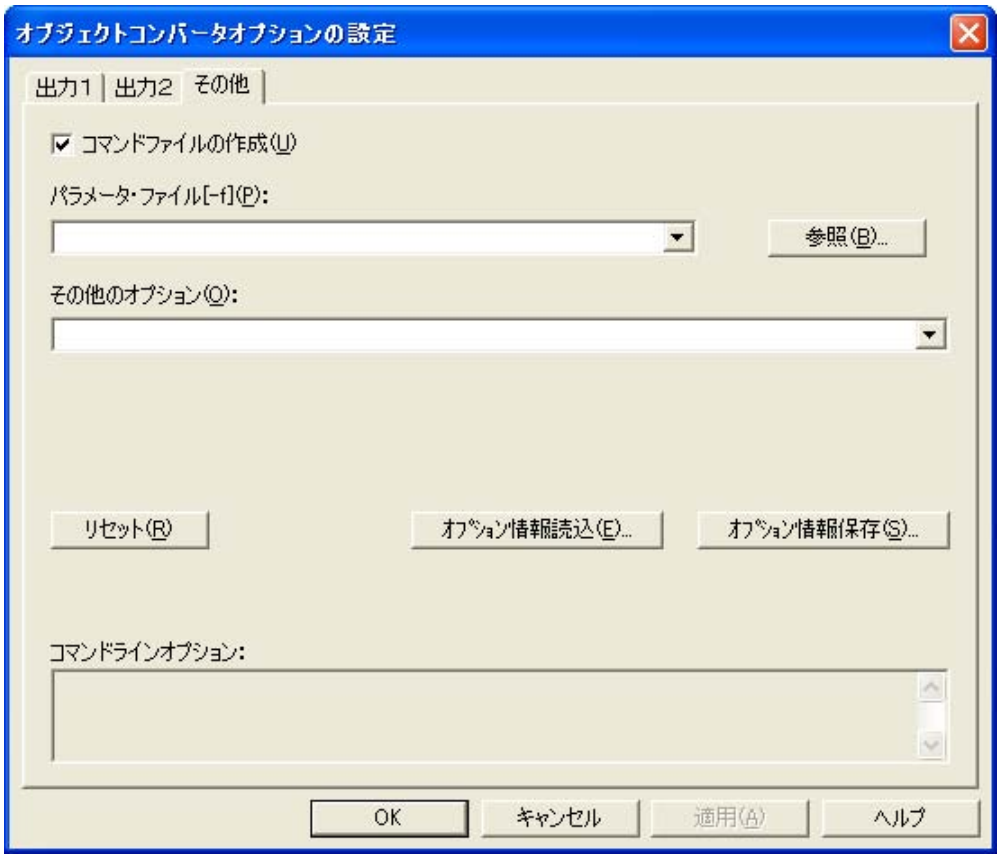
  ［参照 (R)...］ボタン，または直接入力により，エラー・リスト・ファイルのパスとファイル名を指定します。

- 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。

(3) [その他] タブ

図 6-10 [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）



- 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+ 上では指定することはできません。

- 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)...

  [オプション情報の読込み] 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

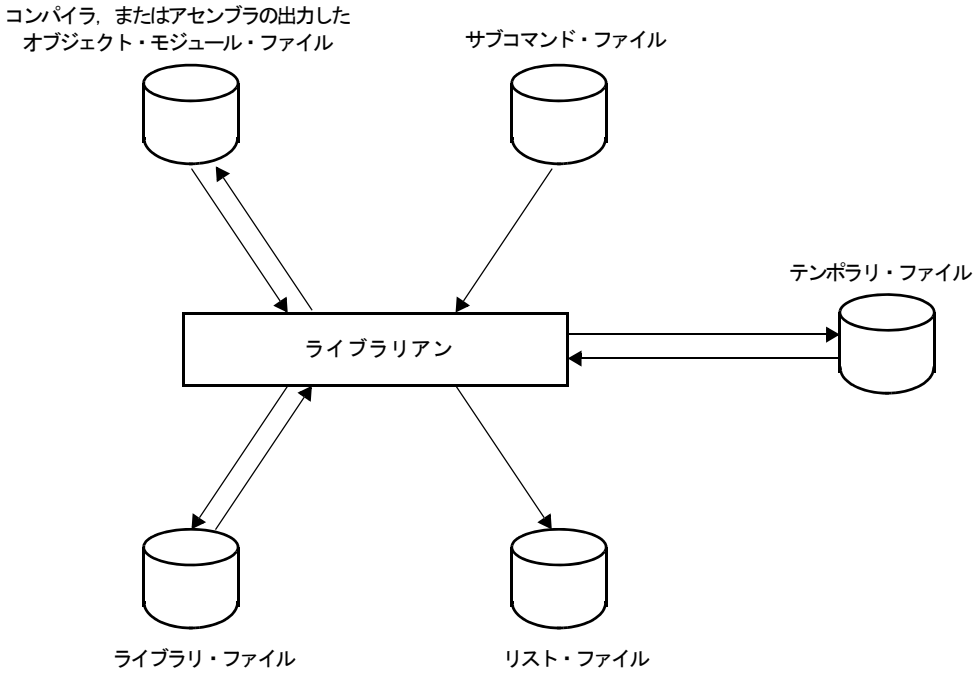
このエディット・ボックスは読み取り専用です。現在設定されているオプション文字列が表示されます。

第 7 章 ライブラリアン

ライブラリアンは，RA78K0R 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，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。

図 7-1 ライブラリアンの入出力ファイル



7.1 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 7-1 ライブラリアンの入出力ファイル

種類	ファイル名	説明	デフォルト・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（ユーザ作成ファイル）	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	.lst
入出力ファイル	オブジェクト・モジュール・ファイル	- コンパイラ，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	.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 ライブラリアンの実行終了時には消去されます。	Lb*xxxxxx*.$$y (y = 1 - 6)

7.2 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が用意されています。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，リンカに入力することができます。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良くなります。

(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。

- ライブラリ・ファイルへのモジュールの追加
- ライブラリ・ファイル内のモジュールの削除
- ライブラリ・ファイル内のモジュールの置換
- ライブラリ・ファイル内のモジュールの抽出

各機能の詳細については，「7.5 サブコマンド」を参照してください。

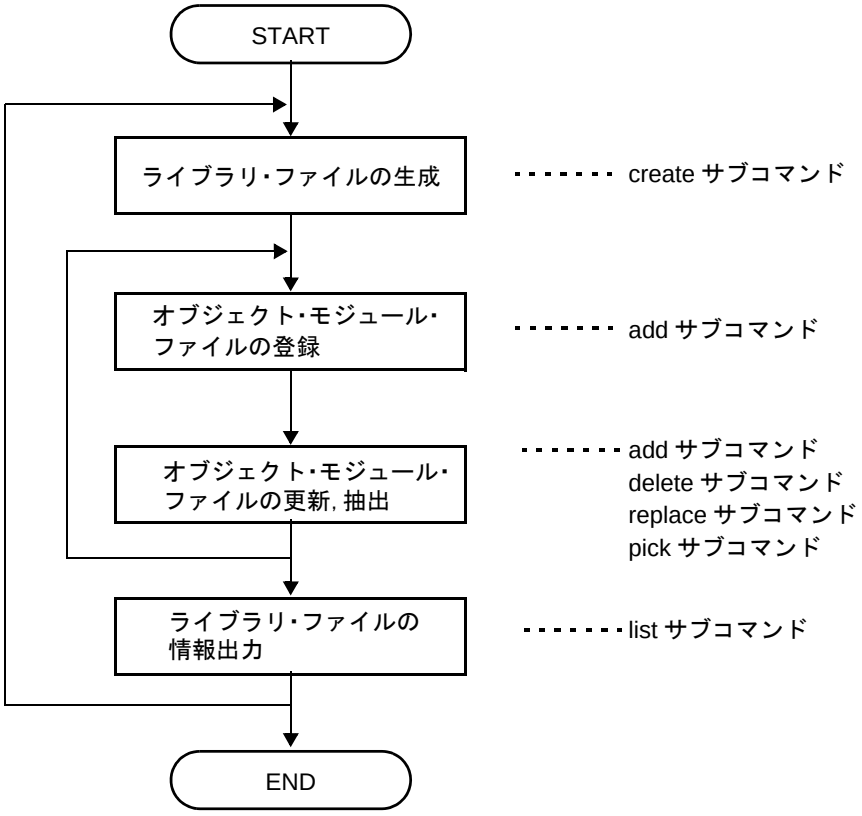
(3) ライブラリ・ファイル情報の出力

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。

- モジュール名
- 作成プログラム
- 登録日付
- 更新日付
- 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，「7.5 サブコマンド」を参照してください。

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。



7.3 ライブラリアンの起動

7.3.1 ライブラリアンの起動方法

ライブラリアンの起動には，2 つの方法があります。

(1) コマンド行での起動

```
 X>[ パス名 ]lb78k0r[ △オプション] …
  ↑    ↑       ↑          ↑
(1)  (2)     (3)        (4)
```

(1) カレント・ドライブ名

(2) カレント・フォルダ名

(3) ライブラリアンのコマンド名

(4) ライブラリアンに対して動作の詳細を指示します。

複数のライブラリアン・オプションを指定する場合は，それぞれのオプション間を空白で区切ります。

なお，ライブラリアン・オプションに大文字，小文字の区別はありません。ライブラリアン・オプションの詳細については，「7.4 ライブラリアン・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。

【例】

C>lb78k0r -ll20 -lw80

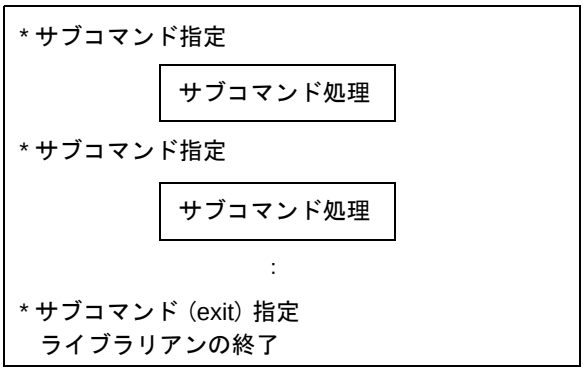
ライブラリアンが起動すると，次の起動メッセージが表示されます。

```
78K0R Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation 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 k0r.lib
*add k0r.lib k0rmain.rel k0rsub.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を行うことができます。なお，継続可能な行数は，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>lb78k0r △< サブコマンド・ファイル名 [ △オプション ] …
           ↑          ↑
         (1)        (2)
```

(1) 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。

(2) サブコマンドが格納されているファイル

備考 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。

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
```

- 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。
- “;”（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。
- 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。
- 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】

1. サブコマンド・ファイル k0r.slb をエディタなどで作成します。

```
; library creation command
create k0r.lib
add k0r.lib k0rmain.rel &
k0rsub.rel
;
exit
```

2. サブコマンド・ファイル k0r.slb を使用してライブラリアンを起動します。

```
C>lb78k0r <k0r.slb
```

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

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。

```
78K0R Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation 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 k0r.lib
*add k0r.lib k0rmain.rel k0rsub.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>lb78k0r -a

78K0R Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx
RA78K0R error F5018 : Option is not recognized '-z'
Usage : LB78K0R [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・メッセージの原因については，「第 11 章 エラー・メッセージ」で調べて対処してください。

7.4 ライブラリアン・オプション

7.4.1 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，ライブラリアンの動作に細かい指示を与えるものです。

ライブラリアン・オプションの分類と説明を示します。

表 7-2 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。
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。

リスト・ファイル形式指定

-lw , -ll , -lf/-nlf

(1) -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 オプションで指定可能な文字数の範囲（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されます。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 サブコマンドを指定していない場合，-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>lb78k0r -lw80
```

(2) -ll

【記述形式】

```
-ll[ 行数 ]
```

- 省略時解釈

  -ll0（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 オプションで指定可能な行数の範囲は，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。

- 行数を省略した場合，0 を指定したものとみなされます。
- 行数の 0 を指定した場合は，改頁されません。
- list サブコマンドを指定していない場合，-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>lb78k0r -ll20
```

(3) -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）を付加することを指定をします。
- -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 サブコマンドを指定していない場合，-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>lb78k0r -lf

テンポラリ・ファイル作成パス指定

-t

(1) -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

  指定されていない場合は，カレント・パス

【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することはできません。
- パス名を省略することはできません。
- 以前に作成したテンポラリ・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されます。
- 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。

  (1) -t オプションで指定したパス

  (2) 環境変数 TMP で設定したパス（-t オプション省略の場合）

  (3) カレント・パス（TMP を設定していない場合）

  なお，(1)，または (2) 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・ファイルをフォルダ C:¥tmp に出力します。

```
C>lb78k0r -tC:¥tmp
```

ヘルプ指定

--

(1) --

【記述形式】

```
--
```

- 省略時解釈

  表示しません。

【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ほかのライブラリアン・オプションは，すべて無効となります。

注意 本オプションは，PM+ 上では指定することはできません。
PM+ 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>lb78k0r --

```
78K0R Series Librarian Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation 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                                                       |
+-------------------------------------------------------------------------+
```

7.5 サブコマンド

7.5.1 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るものです。

サブコマンドの説明を示します。

表 7-3 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ほか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。

7.5.2 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 ]
                                    ↑                                  ↑
                                   (1)                                (2)
```

(1) 直前に指定されたライブラリ・ファイル名は，“ . ”で置き換えることができます。

(2) *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* [△ , …])]

備考 サブコマンド，オプションに大文字，小文字の区別はありません。

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，登録することができません。

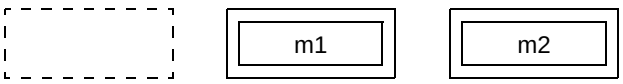
  また，ライブラリ・ファイル中にあるモジュールと同一の名前のモジュールは，登録することができません。
- エラーが発生した場合は，処理を中断し，ライブラリ・ファイルは作成されません。

【使用例】

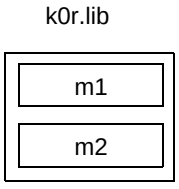
- ライブラリ・ファイル k0r.lib を作成し，同時にモジュール m1 と m2 を登録します。

```
*create k0r.lib m1.rel m2.rel
```

〈作成前〉



〈作成後〉



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】

- add サブコマンドは，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】

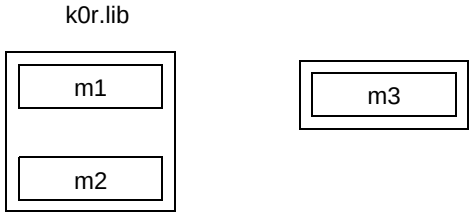
- 追加するライブラリ・ファイル中には，モジュールが存在していなくてもかまいません。
- 追加するモジュールと同名のモジュールがライブラリ・ファイル内に存在する場合，エラーとなります。
- 追加するモジュール中のパブリック・シンボルがライブラリ・ファイル内に存在する場合，エラーとなります。

【使用例】

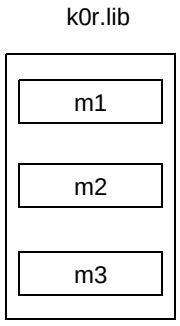
- ライブラリ・ファイル k0r.lib にモジュール m3 を追加します。

```
*add k0r.lib m3.rel
```

〈追加前〉



〈追加後〉



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】

- delete サブコマンドは，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】

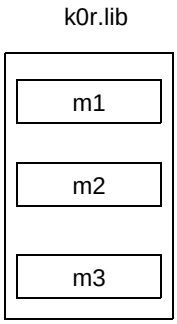
- 指定したモジュールがライブラリ・ファイルに存在しない場合，エラーとなります。
- エラーが発生した場合は処理を中断し，ライブラリ・ファイルの状態は変化しません。

【使用例】

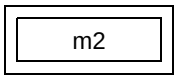
- ライブラリ・ファイル k0r.lib からモジュール m1，m3 を削除します。

```
*delete k0r.lib ( m1.rel , m3.rel )
```

〈削除前〉



〈削除後〉



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】

- replace サブコマンドは，既存のライブラリ・ファイルのモジュールを，ほかのオブジェクト・モジュール・ファイルのモジュールと置き換えます。

【説明】

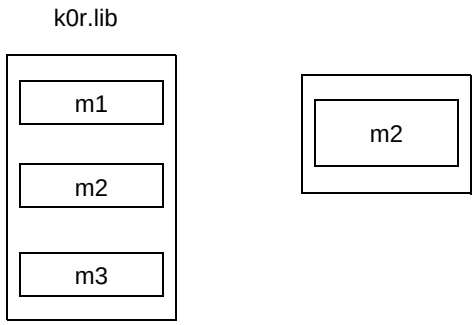
- 置換するモジュール名と同名のモジュールが更新するライブラリ・ファイル内に存在しない場合は，エラーとなります。
- 置換するモジュール中のパブリック・シンボルがライブラリ・ファイル内に存在する場合は，エラーとなります。
- 置換するオブジェクト・モジュール・ファイル名は，登録時と同じファイル名でなければなりません。
- エラーが発生した場合は処理を中断し，ライブラリ・ファイルの状態は変化しません。

【使用例】

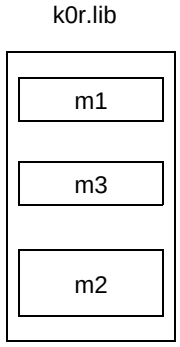
- ライブラリ・ファイル k0r.lib 中のモジュール m2 を置換します。

*replace k0r.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。

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】

- pick サブコマンドは，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】

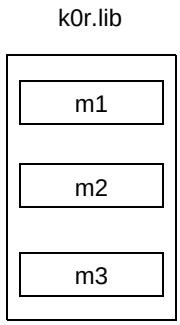
- 取り出したモジュールは，登録時のファイル名を持つオブジェクト・モジュール・ファイルとなります。
- 指定したモジュール名がライブラリ・ファイル内に存在しない場合は，エラーとなります。
- エラーの場合は，処理を中断します。ただし，複数のモジュールが指定されているときにエラーが発生した場合には，エラーとなったモジュールの直前までに取り出されたモジュールは有効となり，ディスク上に保存されます。

【使用例】

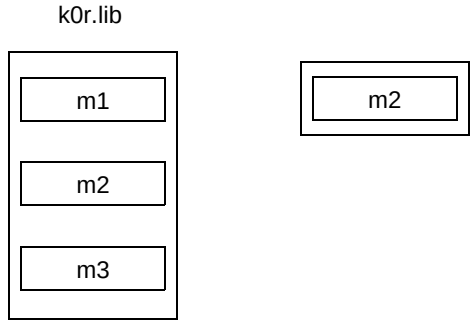
- ライブラリ・ファイル k0r.lib 中のモジュール m2 を取り出します。

*pick k0r.lib (m2.rel)

〈抽出前〉



〈抽出後〉



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

  I

【機能】

- list サブコマンドは，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，複数指定することができます。なお，オプションに大文字，小文字の区別はありません。
- -o :

  出力ファイル名には，デバイス型ファイル名を指定することができます。

  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 :

  -p/-np と指定することも可能です。

  -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，あとで指定した方が優先されます。

【使用例】

- ライブラリ・ファイル k0r.lib のモジュール情報をリスト・ファイル k0r.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 -ok0r.lst k0r.lib

＜リスト・ファイル k0r.lst の内容＞

```
78K0R Series librarian Vx.xx                  DATE : xx xxx xx    PAGE  1

LIB-FILE NAME : k0r.lib          ( xx xxx xx )

0001  k0rmain.rel        ( xx xxx xx )

  MAIN                           START

  NUMBER OF PUBLIC SYMBOLS :     2

0002    k0rsub.rel       ( xx xxx xx )

  CONVAH

  NUMBER OF PUBLIC SYMBOLS :     1
```

help

【記述形式】

```
help
```

- 省略時解釈

  h

【機能】

- help サブコマンドは，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                                                        |
+-------------------------------------------------------------------------+
```

exit

【記述形式】

```
exit
```

- 省略時解釈

  e

【機能】

- exit サブコマンドは，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

7.6 PM+ でのオプション設定

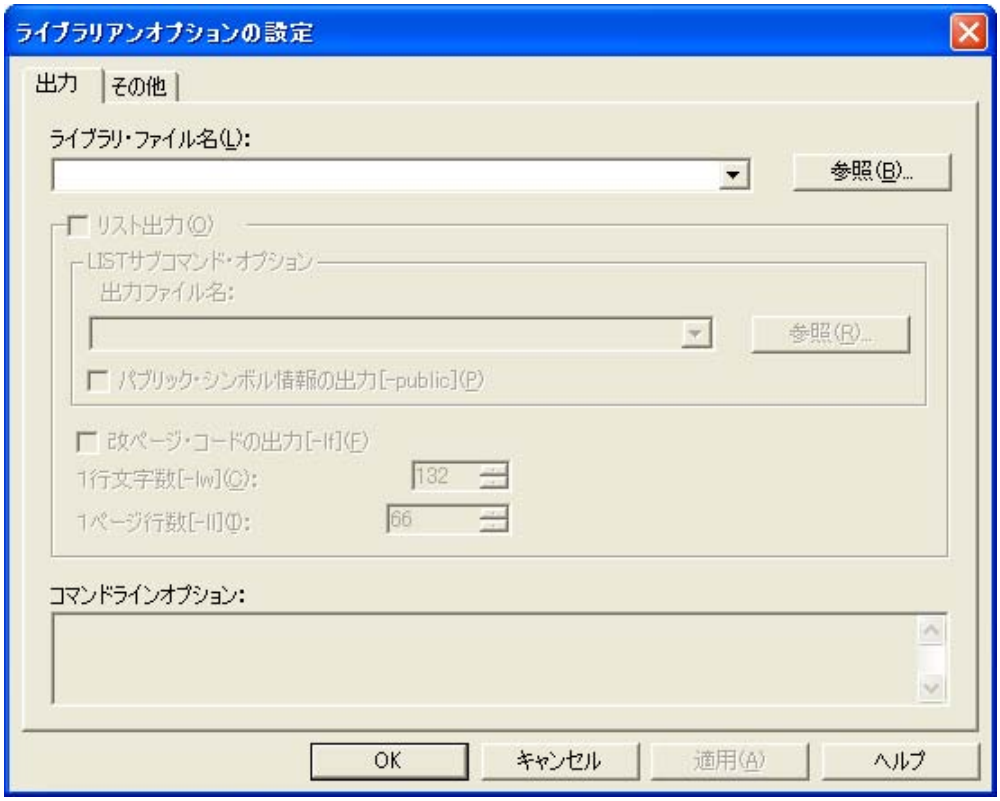
PM+ からライブラリアン・オプションの指定をする方法について説明します。

7.6.1 オプションの設定方法

PM+ の [ツール (T)] メニュー→ [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，各ライブラリアンのオプションを設定することができます。

図 7-2 [ライブラリアンオプションの設定] ダイアログ

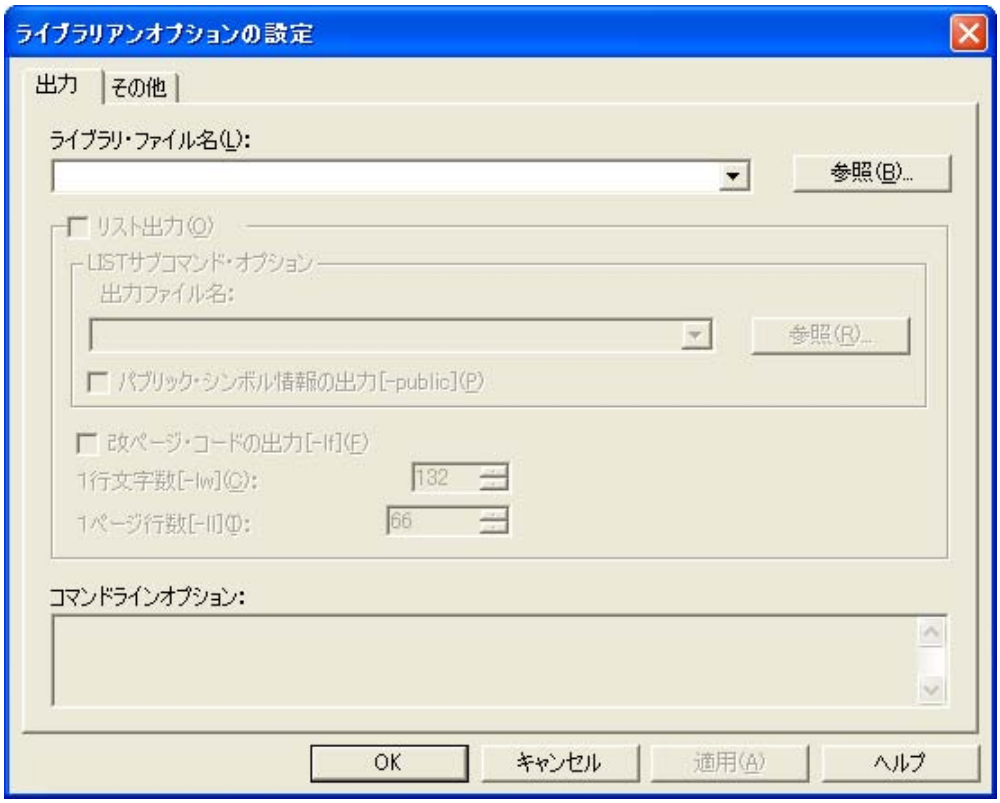


7.6.2 ダイアログの説明

[ライブラリアンオプションの設定] ダイアログの各タブについて，次に説明します。

(1) [出力] タブ

図 7-3 [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）



- 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。

  指定可能な文字数の範囲は，次のとおりです。

  72 ～ 260

- 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。

  指定可能な行数の範囲は，次のとおりです。

  0，および 20 ～ 32,767

- 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。

(2) [その他] タブ

図 7-4 [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）

- 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+ 上では指定することはできません。

- 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)...

[オプション情報の読込み] 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。

7.7 PM+ でのライブラリ・ファイルの操作

PM+ 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。

7.7.1 操作方法

PM+ の [ツール (T)] メニュー→ [外部ツールの登録 (X)] から，LB 単体起動用実行形式（lb78k0rp.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。

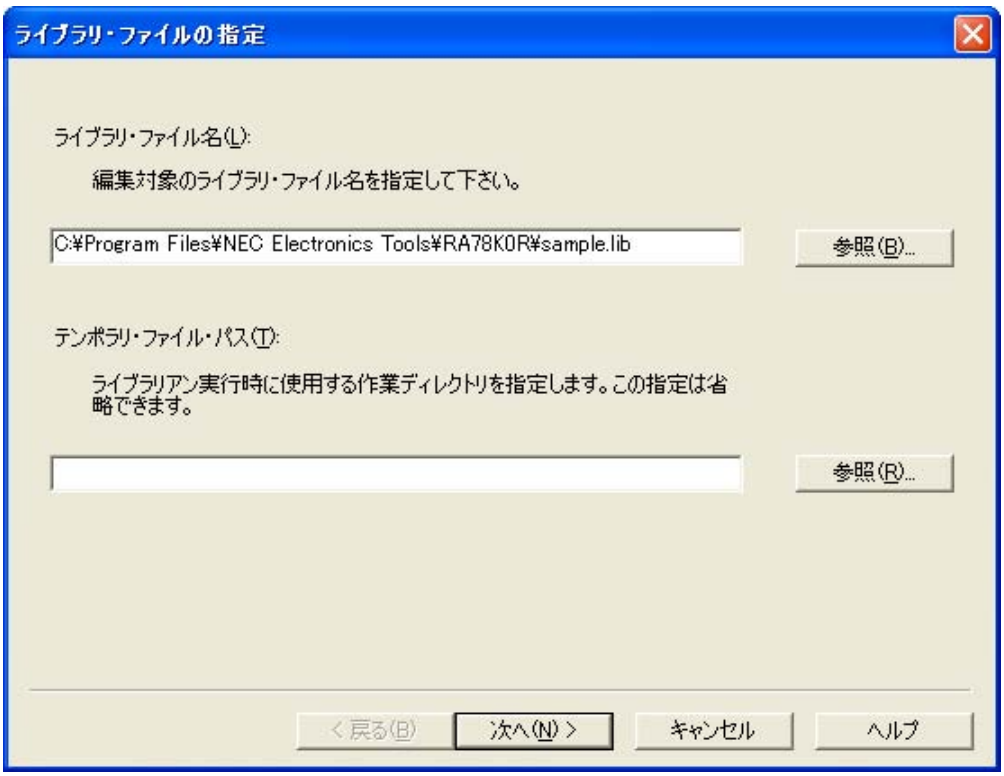
パスとファイル名を指定後，［次へ］ボタンを押下すると，[サブコマンド実行] ダイアログが現れます。

7.7.2 各ダイアログの説明

[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。

(1) [ライブラリ・ファイルの指定] ダイアログ

図 7-5 [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)

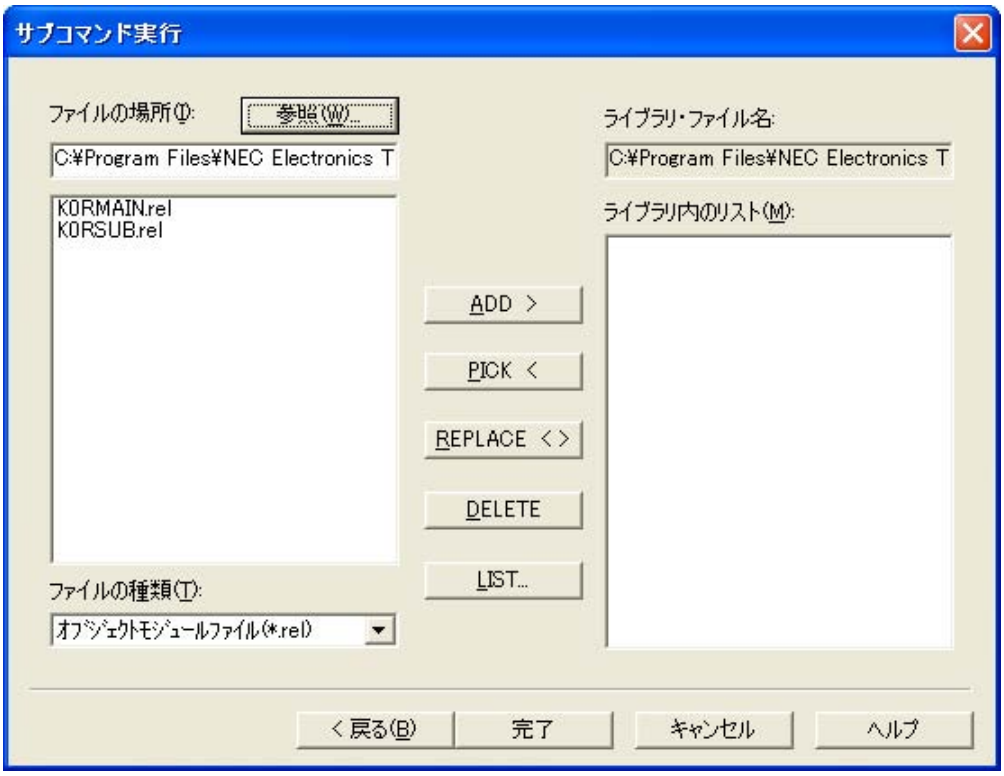
  ［参照 (B)...］ボタン，または直接入力により，編集対象のライブラリ・ファイルのパスとファイル名を指定します。

- 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。

(2) [サブコマンド実行] ダイアログ

図 7-6 [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ほかのオブジェクト・モジュール・ファイルのモジュールと置き換えます。

- DELETE

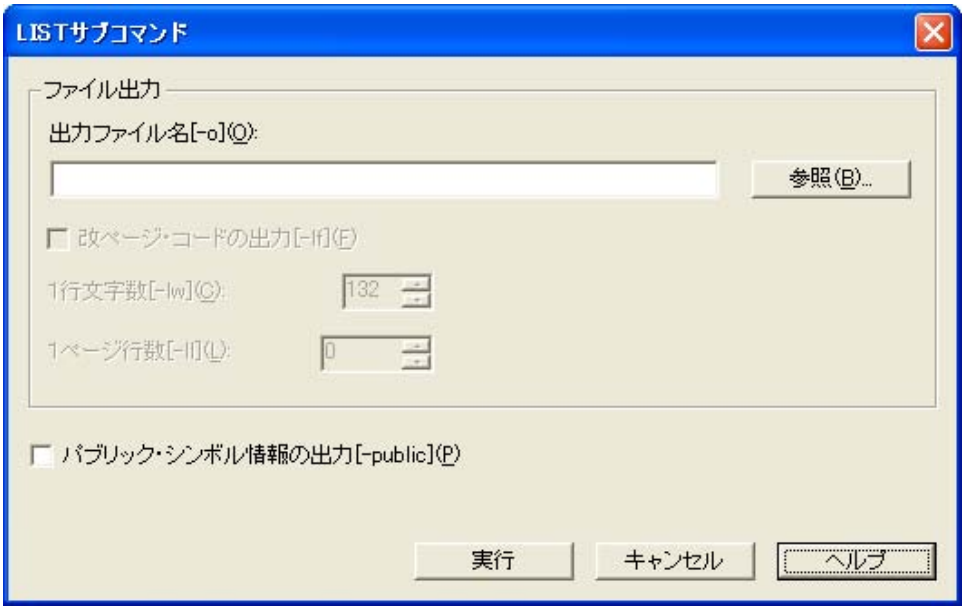
既存のライブラリ・ファイルからモジュールを削除します。

- 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。

[LIST サブコマンド] ダイアログを開きます。

表 7-4 [LIST サブコマンド] ダイアログ



- 出力ファイル名 [-o](O)

［参照 (B)...］ボタン，または直接入力により，出力ファイルのパスとファイル名を指定します。

- 改ページ・コードの出力 [-lf](F)

出力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) :

出力ファイルの 1 行の文字数を指定します。

指定可能な文字数の範囲は，次のとおりです。

72 ～ 260

1 ページ行数 [-ll](L) :

出力ファイルの 1 ページの行数を指定します。

指定可能な行数の範囲は，次のとおりです。

0，および 20 ～ 32767

- パブリック・シンボル情報の出力 [-public](P)

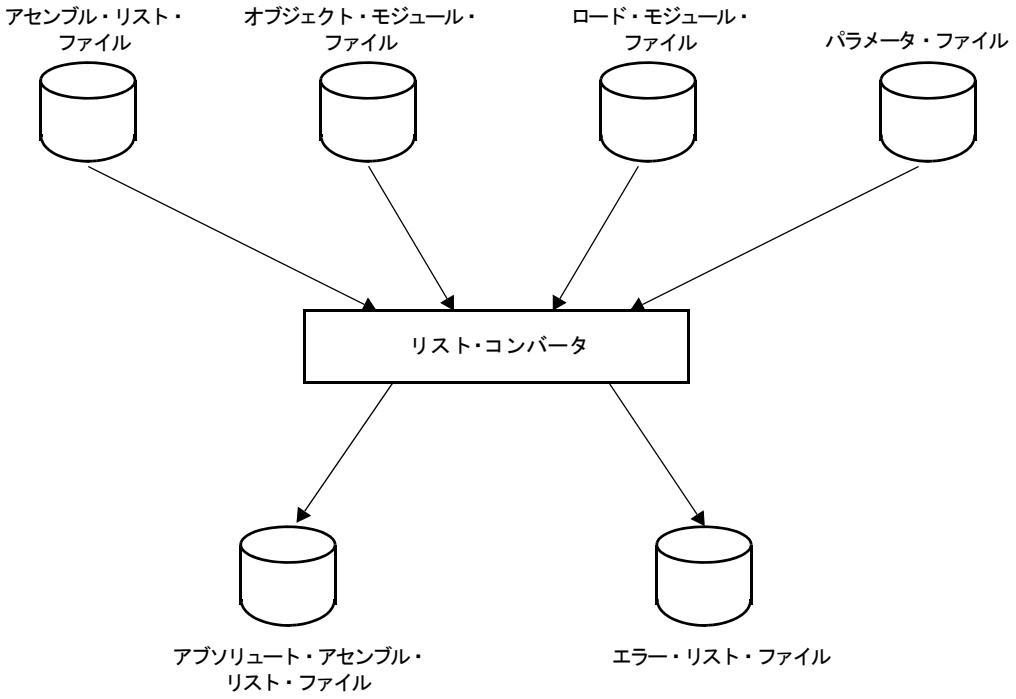
出力ファイル中にパブリック・シンボル情報を付加する場合に，チェックします。

第 8 章 リスト・コンバータ

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。

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み，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。

図 8-1 リスト・コンバータの入出力ファイル



8.1 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 8-1 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	デフォルト・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	.rel
	アセンブル・リスト・ファイル	- アセンブル・リスト，クロスリファレンス・リストなどのアセンブル情報を持つファイル	.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	.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（ユーザ作成ファイル）	.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	.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	.elv

8.2 リスト・コンバータの機能

(1) アセンブラ（リロケータブル・アセンブラ）の短所を解決

リスト・コンバータは，アセンブル・リスト・ファイルにロケーション，オブジェクト・コードを埋め込むことにより，リロケータブル・アセンブラの短所を解決します。

- 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。
- 外部シンボルの実際の値がリスト上に埋め込まれます。
- 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。
- シンボル・テーブル，あるいはクロスリファレンス・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】，**【例 2】**に，リスト・コンバータによって得られるアブソリュート・アセンブル・リストの例を示します。

【例 1】

ロケーションの埋め込みは，次のようになります。

＜アセンブル・リスト＞

```
22  22  -----           CSEG
23  23  00000       START :
24  24
25  25                  ; chip initialize
26  26  00000 RCBF80000 MOVW    SP , #_@STBEG
27  27
28  28  00004 CD201A    MOV     HDTSA , #1AH
29  29  00007 3620FE    MOVW    HL , #LOWW ( HDTSA )   ; set hex 2-code data in
HL registor
30  30
31  31  0000A RFD0000   CALL    !CONVAH     ; convert ASCII <- HEX
32  32                                      ; output BC-register <- ASCII code
33  33  0000D 3421FE    MOVW    DE , #LOWW ( STASC )    ; set DE <- store ASCII
code table
34  34  00010 63        MOV     A , B
35  35  00011 99        MOV     [ DE ] , A
36  36  00012 A5        INCW    DE
37  37  00013 62        MOV     A , C
38  38  00014 99        MOV     [ DE ] , A
39  39
40  40  00015 EFFE      BR      $$
41  41
42  42                  END
```

＜アブソリュート・アセンブル・リスト＞

```
22  22  -----           CSEG
23  23  000D2       START :
24  24
25  25                  ; chip initialize
26  26  000D2 RCBF820FE MOVW    SP , #_@STBEG
27  27
28  28  000D6 CD201A    MOV     HDTSA , #1AH
29  29  000D9 3620FE    MOVW    HL , #LOWW ( HDTSA )   ; set hex 2-code data in
HL registor
30  30
31  31  000DC RFDE900   CALL    !CONVAH     ; convert ASCII <- HEX
32  32                                      ; output BC-register <- ASCII code
33  33  000DF 3421FE    MOVW    DE , #LOWW ( STASC )    ; set DE <- store ASCII
code table
34  34  000E2 63        MOV     A , B
35  35  000E3 99        MOV     [ DE ] , A
36  36  000E4 A5        INCW    DE
37  37  000E5 62        MOV     A , C
38  38  000E6 99        MOV     [ DE ] , A
39  39
40  40  000E7 EFFE      BR      $$
41  41
42  42                  END
```

【例 2】

オブジェクト・コードの埋め込みは，次のようになります。

＜アセンブル・リスト＞

```
22  22  -----           CSEG
23  23  00000       START :
24  24
25  25                  ; chip initialize
26  26  00000 RCBF80000 MOVW    SP , #_@STBEG
27  27
28  28  00004 CD201A    MOV     HDTSA , #1AH
29  29  00007 3620FE    MOVW    HL , #LOWW ( HDTSA )   ; set hex 2-code data in
HL registor
30  30
31  31  0000A RFD0000   CALL    !CONVAH     ; convert ASCII <- HEX
32  32                                      ; output BC-register <- ASCII code
33  33  0000D 3421FE    MOVW    DE , #LOWW ( STASC )    ; set DE <- store ASCII
code table
34  34  00010 63        MOV     A , B
35  35  00011 99        MOV     [ DE ] , A
36  36  00012 A5        INCW    DE
37  37  00013 62        MOV     A , C
38  38  00014 99        MOV     [ DE ] , A
39  39
40  40  00015 EFFE      BR      $$
41  41
42  42                  END
```

＜アブソリュート・アセンブル・リスト＞

```
22  22  -----           CSEG
23  23  000D2       START :
24  24
25  25                  ; chip initialize
26  26  000D2 RCBF820FE MOVW    SP , #_@STBEG
27  27
28  28  000D6 CD201A    MOV     HDTSA , #1AH
29  29  000D9 3620FE    MOVW    HL , #LOWW ( HDTSA )   ; set hex 2-code data in
HL registor
30  30
31  31  000DC RFDE900   CALL    !CONVAH     ; convert ASCII <- HEX
32  32                                      ; output BC-register <- ASCII code
33  33  000DF 3421FE    MOVW    DE , #LOWW ( STASC )    ; set DE <- store ASCII
code table
34  34  000E2 63        MOV     A , B
35  35  000E3 99        MOV     [ DE ] , A
36  36  000E4 A5        INCW    DE
37  37  000E5 62        MOV     A , C
38  38  000E6 99        MOV     [ DE ] , A
39  39
40  40  000E7 EFFE      BR      $$
41  41
42  42                  END
```

8.3 リスト・コンバータの起動

8.3.1 リスト・コンバータの起動方法

リスト・コンバータの起動には，2 つの方法があります。

(1) コマンド行での起動

```
 X>lc78k0r[ △オプション] … 入力ファイル名 [ △オプション] … [ △ ]
  ↑    ↑          ↑              ↑            ↑
(1)  (2)        (3)            (4)          (3)
```

(1) カレント・ドライブ名

(2) リスト・コンバータのコマンド・ファイル名

(3) リスト・コンバータに対して動作の詳細を指示します。
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，それぞれのオプション間を空白で区切ってください。なお，リスト・コンバータ・オプションに大文字，小文字の区別はありません。リスト・コンバータ・オプションの詳細については，「8.4 リスト・コンバータ・オプション」を参照してください。

(4) 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。
ファイルの拡張子は“.prn”にしてください。
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。

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

【例】

- アセンブル・リスト・ファイル k0rmain.prn とロード・モジュール・ファイル sample.lmf のプライマリ・ネームが異なる場合，ロード・モジュール・ファイル sample.lmf の入力を指定します。

```
C>lc78k0r k0rmain.prn -lsample.lmf
```

(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は，次のようになります。

```
C>lc78k0r[ △入力ファイル名 ] △ -f パラメータ・ファイル名
                              ↑          ↑
                             (1)        (2)
```

(1) パラメータ・ファイル指定オプション

(2) 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することができます。
- 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】

1. パラメータ・ファイル k0r.plv をエディタで作成します。

```
; parameter file
k0rmain -lk0r.lmf

-ek0r.elv
```

2. パラメータ・ファイル k0r.plv を使用してリスト・コンバータを起動します。

```
C>lc78k0r -fk0r.plv
```

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

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 RA78K0R Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation 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>lc78k0r k0rmain.prn -a

```
List Conversion Program for RA78K0R Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

RA78K0R error F6018 : Option is not recognized '-a'
Please enter 'LC78K0R --', if you want help messages.
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・メッセージの原因については，「第 11 章 エラー・メッセージ」で調べて対処してください。

8.4 リスト・コンバータ・オプション

8.4.1 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・オプションの分類と説明を示します。

表 8-2 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。

オブジェクト・モジュール・ファイル入力指定

-r

(1) -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】

- フェイタル・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 k0rmain.prn とオブジェクト・モジュール・ファイル sample.rel のプライマリ・ネームが異なる場合，ロード・モジュール・ファイル sample.rel の入力を指定します。

```
C>lc78k0r k0rmain.prn -lsample.rel
```

ロード・モジュール・ファイル入力指定

-l

(1) -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】

- フェイタル・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 k0rmain.prn とロード・モジュール・ファイル sample.lmf のプライマリ・ネームが異なる場合，ロード・モジュール・ファイル sample.lmf の入力を指定します。

```
C>lc78k0r k0rmain.prn -lsample.lmf
```

アブソリュート・アセンブル・リスト・ファイル出力指定

-o

(1) -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することができます。

【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することができます。
  指定することのできるデバイス型ファイル名は，CON，PRN，NUL，および AUX です。
- 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>lc78k0r k0rmain.prn -osample.p -lk0r.lmf
```

エラー・リスト・ファイル出力指定

-e/-ne

(1) -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することができます。
- -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することができます。
- 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を指定した場合，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>lc78k0r k0rmain.prn -esample.elv

＜エラー・リスト・ファイル sample.elv の内容＞

```
RA78K0R warning W6701 : Load module file is older than object module file
'k0rmain.lmf, k0rmain.rel'
Pass1 : start

RA78K0R warning W6702 : Load module file is older than assemble module file
'k0rmain.lmf, k0rmain.prn'

Pass2 : start
```

パラメータ・ファイル指定

-f

(1) -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となります。

【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ことを指定をします。

【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することができるのは，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・ファイル中に記述可能な文字数に，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されます。

【使用例】

- パラメータ・ファイル k0r.plv 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 k0r.plv の内容＞

```
: parameter file
k0rmain -lk0r.lmf
-ek0r.elv
```

コマンド行には，次のように入力します。

```
C>lc78k0r -fk0r.plv
```

ヘルプ指定

--

(1) --

【記述形式】

```
--
```

- 省略時解釈

  表示しません。

【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，ほかのリスト・コンバータ・オプションはすべて無効となります。

注意 本オプションは，PM+ 上では指定することはできません。
PM+ 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>lc78k0r --

```
List Conversion Program for RA78K0R Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx

usage : LC78K0R [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.
```

8.5 PM+ でのオプション設定

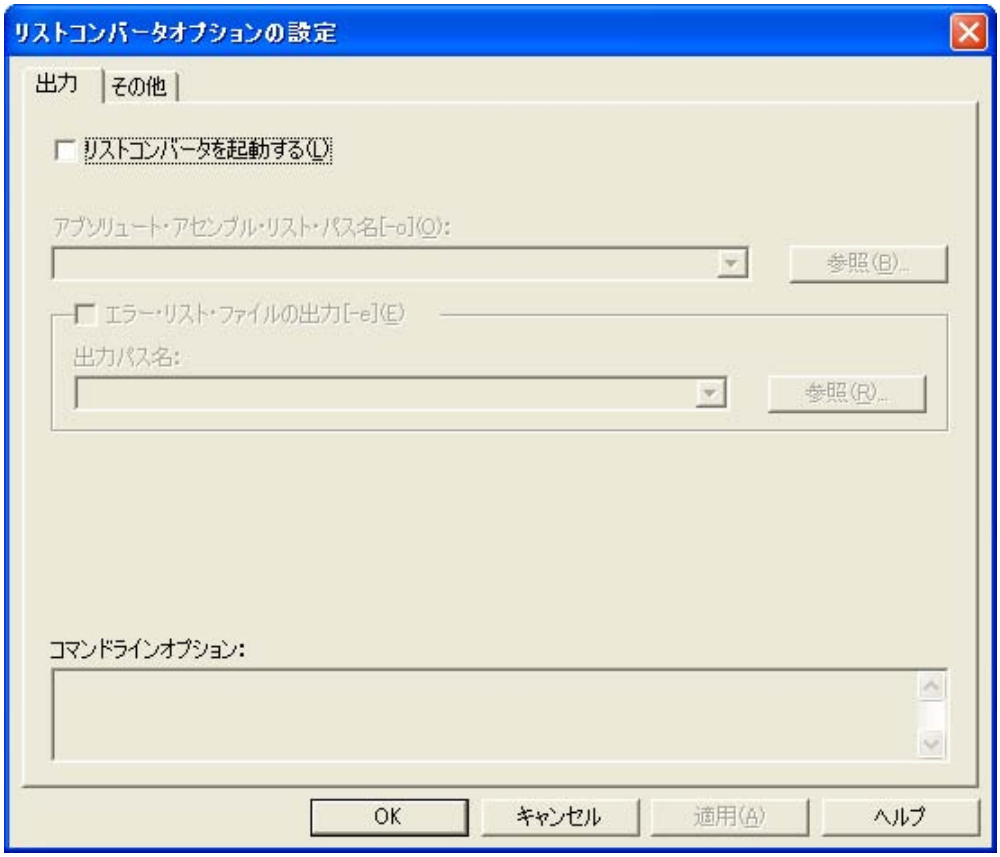
PM+ からリストコンバータ・オプションを設定する方法について説明します。

8.5.1 オプションの設定方法

PM+ の [ツール (T)] メニュー→ [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・オプションを設定することができます。

図 8-2 [リストコンバータオプションの設定] ダイアログ

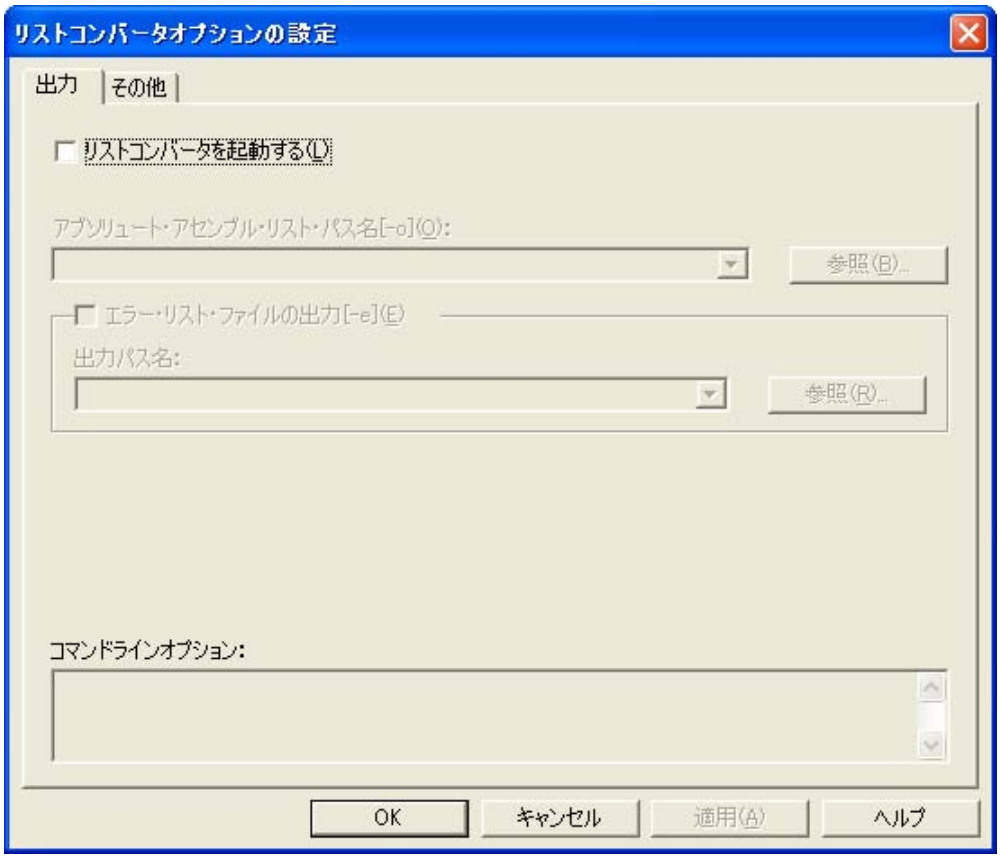


8.5.2 ダイアログの説明

[リストコンバータオプションの設定] ダイアログの各タブについて，次に説明します。

(1) [出力] タブ

図 8-3 [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）



- 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：

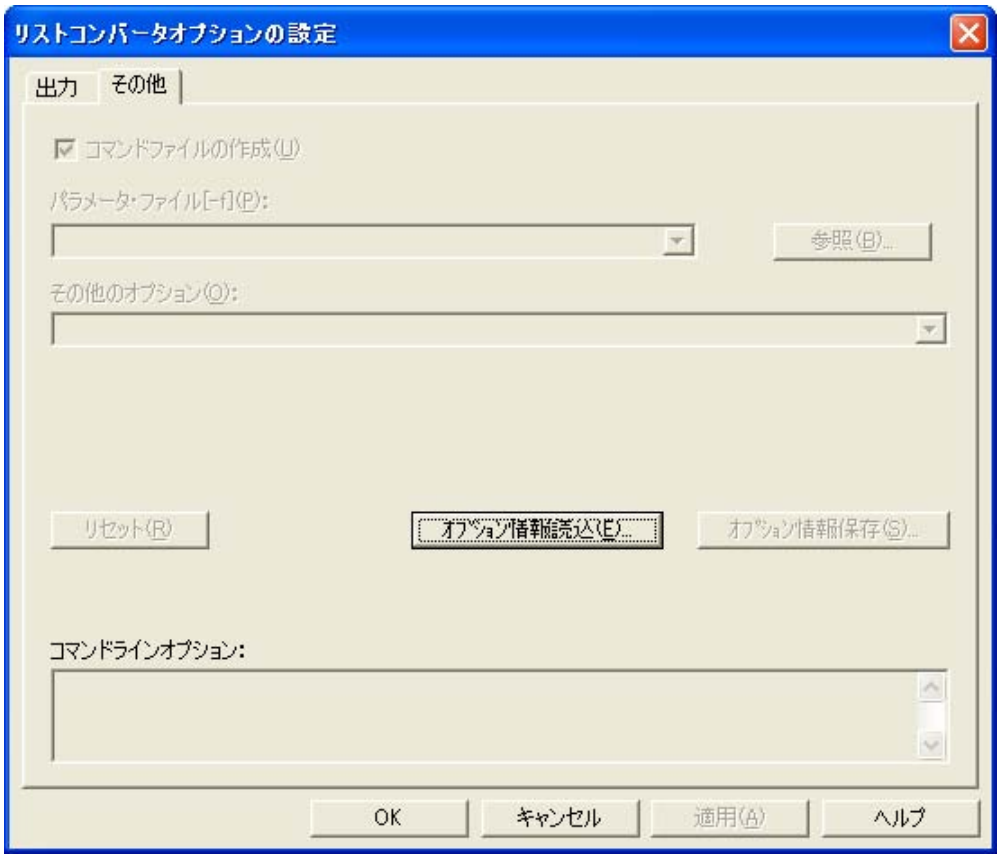
  [参照 (R)...] ボタン，または直接入力により，エラー・リスト・ファイルのパスを指定します。

- 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。

(2) [その他] タブ

図 8-4 [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）



- 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+ 上では指定することはできません。

- 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)...

  [オプション情報の読込み] 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 9 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。

- アセンブラの出力リスト
- リンカの出力リスト
- オブジェクト・コンバータの出力リスト
- ライブラリアンの出力リスト
- リスト・コンバータの出力リスト

9.1 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 9-1 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・ファイルのヘッダ
	アセンブル・リスト
	シンボル・リスト
	クロスリファレンス・リスト
エラー・リスト・ファイル	エラー・リスト

9.1.1 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】

```
78K0R Series Assembler (1)Vx.xx (2)SAMPLE_TITLE Date:(3)xx xxx xxxx Page:(4)1
(5)SAMPLE_SUBTITLE
Command :   (6)  k0rmain.asm -cf1166a0
Para-file : (7)  -ks -kx
In-fine :   (8)  k0rmain.asm
Obj-file :  (9)  k0rmain.rel
Prn-file :  (10) k0rmain.prn
```

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

9.1.2 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

(1)ALNO  (2)STNO  (6)ADRS (8)OBJECT (3)M (4)I (5)SOURCE STATEMENT
    1        1
    2        2                                      NAME  SAMPM
    :
   31       31       0000A   RFD0000                CALL  !CONVAH
                                                 ; convert ASCII <- HEX
   32       32                                     ; output BC-register <- ASCII code
   33       33       0000D   00000000                MOV   DE , #LOWW ( STASC )
                                                 ; set DE <- store ASCII code table
                     00011   00
(7) ** ERROR E2202 , STNO 33 ( 33 ) Illegal operand
   34       34       00012   63                     MOV   A , B
   35       35       00013   99                     MOV   [ DE ] , A
    :
Segment informations :

(9)ADRS (10)LEN  (11)NAME

   FFE20    00003H   DATA
   00000    00002H   CODE
   00000    00019H   ?CSEG

Target chip : (12)uPD78xxx
Device file : (13)Vx.xx
Assembly complete, (14)1 error(s) and (15)0 warning(s) found. ( (16)33 )
```

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。
(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・ステートメント
(6)	ロケーション・カウンタ値（5 桁）
(7)	エラーの発生行

項目	内容
(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（5 桁）
(10)	セグメント・サイズ（8 桁）
(11)	セグメント名
(12)	RA78K0R 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)	フェイタル・エラーの個数
(15)	ワーニングの個数
(16)	最終エラー行

9.1.3 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

(1)VALUE (2)ATTR  (3)RTYP  (4)NAME      (1)VALUE (2)ATTR  (3)RTYP (4)NAME
           CSEG             ?CSEG                CSEG             CODE
   ----H            EXT     CONVAH               DSEG             DATA
   FFE20H  ADDR             HDTSA           0H   ADDR     PUB     MAIN
           MOD              SAMPM           0H   ADDR     PUB     START
   FFE21H  ADDR             STASC        -----H           EXT     _@STBEG
```

項目	内容
(1)	シンボル値（8 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） SFR ： SFR を EQU 疑似命令で定義したネーム SFRP ： SFRP を EQU 疑似命令で定義したネーム 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

9.1.4 クロスリファレンス・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

(1)NAME   (2)VALUE (3)R     (4)ATTR   (5)RTYP   (6)SEGNAME    (7)XREFS
   ?CSEG                      CSEG                ?CSEG         22#
   CODE                       CSEG                CODE          19#
   CONVAH    -----H    E                  EXT                   12@    31
   DATA                       DSEG                DATA          15#
   HDTSA     FFE20H           ADDR                DATA          16#    28     29
   MAIN          0H           ADDR      PUB       CODE          11@    20#
   SAMPM                      MOD                                2#
   START         0H    R      ADDR      PUB       ?CSEG         11@    20     23#
   STASC     FFE21H           ADDR                DATA          17#    33
   _@STBEG   -----H    E                  EXT                   13@    26
```

項目	内容
(1)	定義されたシンボル名
(2)	シンボル値（8 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） SFR　：SFR を EQU 疑似命令で定義したネーム SFRP　：SFRP を EQU 疑似命令で定義したネーム 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

項目	内容
(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
(7)	定義・参照行番号 定義行　：xxxxx# 参照行　：xxxxx △　（△は空白 1 つ） EXTRN 宣言，EXTBIT 宣言，PUBLIC 宣言　：xxxxx@

9.1.5　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 ) : RA78K0R (3)error (4)E2202 : (5)Illegal operand
(1)ERROR.ASM ( (2)32 ) : RA78K0R (3)error (4)E2202 : (5)Illegal operand
PASS2 Start
(1)ERROR.ASM ( (2)26 ) : RA78K0R (3)error (4)E2202 : (5)Illegal operand
(1)ERROR.ASM ( (2)29 ) : RA78K0R (3)error (4)E2407 : (5)Undefined symbol
reference 'DTSA'
(1)ERROR.ASM ( (2)29 ) : RA78K0R (3)error (4)E2303 : (5)Illegal expression
(1)ERROR.ASM ( (2)32 ) : RA78K0R (3)error (4)E2202 : (5)Illegal operand
(1)ERROR.ASM ( (2)37 ) : RA78K0R (3)error (4)E2407 : (5)Undefined symbol
reference 'F'
(1)ERROR.ASM ( (2)37 ) : RA78K0R (3)error (4)E2303 : (5)Illegal expression
```

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。

9.2 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 9-2 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	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・ファイル	エラー・リスト

9.2.1　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】

```
78K0R Series Linker (1)Vx.xx                Date :(2)xx xxx xxxx Page : (3)1

Command :       (4) k0rmain.rel k0rsub.rel -s -ok0r.map -dk0r.dr
Para-file :     (5)
Out-file :      (6) k0rmain.lmf
Map-File :      (7) k0r.map
Direc-File :    (8) k0r.dr
Directive :     (9) MEMORY ROM  : ( 0H , 0ED800H )
                (9) MEMORY RAM1 : ( 0FCF00H , 1100H )
                (9) MEMORY RAM  : ( 0FE000H , 1F00H )

*** Link information ***

(10)   6 output segment(s)
(11) 9DH byte(s) real data
(12)  40 symbol(s) defined
```

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
(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

9.2.2 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)00000H    SIZE = (4)ED800H
      (6)OUTPUT   (7)INPUT    (8)INPUT  (9)BASE     (10)SIZE
         SEGMENT     SEGMENT     MODULE    ADDRESS
         CODE                            00000H      00002H  (11)CSEG  AT
                     CODE        SAMPM     00000H      00002H
(5)* gap *                               00002H      000BEH
         ?CSEGOB0                        000C0H      00004H  (11)CSEG  OPT_BYTE
         ?CSEG                           000C4H      00059H  (11)CSEG
                     ?CSEG       SAMPM     000C4H      00017H
                     ?CSEG       SAMPS     000DBH      00042H
(5)* gap *                               0011DH      ED6E3H

  MEMORY = RAM1
  BASE ADDRESS = (3)FCF00H    SIZE = (4)01100H
      (6)OUTPUT   (7)INPUT    (8)INPUT  (9)BASE     (10)SIZE
         SEGMENT     SEGMENT     MODULE    ADDRESS
(5)* gap *                               FCF00H      01100H

  MEMORY = RAM
  BASE ADDRESS = (3)FE000H    SIZE = (4)01F00H
      (6)OUTPUT   (7)INPUT    (8)INPUT  (9)BASE     (10)SIZE
         SEGMENT     SEGMENT     MODULE    ADDRESS
(5)* gap *                               FE000H      01E20H
         DATA                            FFE20H      00003H  (11)DSEG  AT
                     DATA        SAMPM     FFE20H      00003H
(5)* gap *                               FFE23H      000DDH

 Target chip : (12)uPD78xxx
 Device File : (13)Vx.xx
```

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（5 桁）
(4)	メモリ領域のサイズ（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

項目	内容
(9)	セグメントの先頭アドレス（8 桁）
(10)	出力セグメント / 入力セグメントのサイズ（8 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

9.2.3 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 ***

(1)MODULE    (2)ATTR      (3)VALUE     (4)NAME

   SAMPM
                ADDR         00000H       MAIN
                ADDR         000D2H       START
   SAMPS
                ADDR         000E9H       CONVAH

                NUM          FFE20H       _@STBEG

                NUM          FE000H       _@STEND
```

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） SFR ： SFR を EQU 疑似命令で定義したネーム SFRP ： SFRP を EQU 疑似命令で定義したネーム 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（8 桁）
(4)	パブリック・シンボル名

9.2.4 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 ***

(1)MODULE    (2)ATTR      (3)VALUE     (4)NAME

   SAMPM
                MOD                       SAMPM
                DSEG                      DATA
                ADDR         FFE20H       HDTSA
                ADDR         FFE21H       STASC
                CSEG                      CODE
                CSEG                      ?CSEG
   SAMPS
                MOD                       SAMPS
                CSEG                      ?CSEG
                ADDR         00114H       SASC
                ADDR         0011AH       SASC1
```

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） SFR ： SFR を EQU 疑似命令で定義したネーム SFRP ： SFRP を EQU 疑似命令で定義したネーム 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（8 桁）
(4)	ローカル・シンボル名

9.2.5 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】

```
RA78K0R (1)error (2)E3405 : (3)Undefined symbol 'CONVAH' in file 'k0rmain.rel'
```

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

9.3 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 9-3 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

9.3.1 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。

9.4 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 9-4 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

9.4.1 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】

```
78K0R Series librarian (1)Vx.xx  DATE : (2)xx xxx xx        PAGE (3)1

LIB-FILE NAME : (4)k0r.lib        ( (5)xx xxx xx )

(6)0001   (7)k0rmain.rel  ( (8)xx xxx xx )

  (9)MAIN                         (9)START

  NUMBER OF PUBLIC SYMBOLS : (10)2

(6)0002   (7)k0rsub.rel   ( (8)xx xxx xx )

  (9)CONVAH

  NUMBER OF PUBLIC SYMBOLS : (10)1
```

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

9.5 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 9-5 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

9.5.1 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。

9.5.2 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 10 章 RA78K0R の活用法

この章では，RA78K0R を効率良く使用するための方法を紹介します。

10.1 作業の効率化（EXIT ステータス機能）

RA78K0R 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。

表 10-1 EXIT ステータス

処理	EXIT ステータス
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，効率良く作業することができます。

【使用例】

＜バッチ・ファイル ra.bat の内容＞

```
ra78K0R -cf1166a0 k0rmain.-g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
ra78K0R -cf1166a0 k0rsub.asm -g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
lk78K0R k0rmain.rel k0rsub.rel -ok0r.lmf -g
echo off
IF ERRORLEVEL 1 GOTO ERR
echo\
echo on
oc78K0R k0r.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
```

10.2 開発環境の整備（環境変数）

RA78K0R 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。

表 10-2 環境変数

環境変数	説明
PATH	実行形式のサーチ・パス
INC78K0R	インクルード・ファイルのサーチ・パス（アセンブラ）
LIB78K0R	ライブラリ・ファイルのサーチ・パス（リンカ）
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:¥ra78k0r;  ← (1)
SET INC78K0R=C:¥ra78k0r¥include ← (2)
SET LIB78K0R=C:¥ra78k0r¥lib     ← (3)
SET TMP=C:¥tmp                  ← (4)
SET LANG78K=SJIS                ← (5)
```

(1) パス指定により，C:¥bin，C:¥bat，C:¥ra78k0r 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:¥ra78k0r¥include から検索します。

(3) リンカは，ライブラリ・ファイルを C:¥ra78k0r¥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。

10.3 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することができます。

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く制御を 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。

10.4 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，サブタイトルを表示することができます。制御命令については，「RA78K0R 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 k0rmain.prn のヘッダに，タイトルを印字します。

```
C>ra78k0r -cf1166a0 k0rmain.asm -lhRA78K0R_MAINROUTINE
```

＜ k0rmain.prn の内容＞

```
78K0R Series Assembler Ex.xx RA78K0R_MAINROUTINE Date : xx xxx xxxx Page : 1
                                  |
                               タイトル

Command : -cf1166a0 k0rmain.asm -lhRA78K0R_MAINROUTINE
Para-file :
In-file : k0rmain.asm
Obj-file : k0rmain.rel
Prn-file : k0rmain.prn

  Assemble list

ALNO    STNO    ADRS    OBJECT  M I     SOURCE STATEMENT

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

10.5 プログラム起動時の手間を省く

10.5.1 ソース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 ( f1166a0 ) ; 制御命令
$ XREF                  ; 制御命令

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

10.5.2 PM+ を使用する

RA78K0R の各プログラムのオプションは，PM+ 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。

10.5.3 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（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を容易に行うことができます。

【使用例 1】

パラメータ・ファイル k0rmain.pra を使用してアセンブルします。

＜パラメータ・ファイル k0rmain.pra の内容＞

```
; parameter file
k0rmain.asm -osample.rel -g
-psample.prn
```

コマンド行には，次のように入力します。

```
C>ra78k0r -fk0rmain.pra
```

【使用例 2】

パラメータ・ファイル k0r.slb を使用してアセンブルします。

＜パラメータ・ファイル k0r.slb の内容＞

```
;
; library creation command
;
create k0r.lib
;
add k0r.lib k0rmain.rel &
k0rsub.rel
;
exit
```

コマンド行には，次のように入力します。

```
C>lb78k0r <k0r.slb
```

10.6 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が用意されています。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，リンカに入力することができます。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良くなります。

第 11 章 エラー・メッセージ

この章では，RA78K0R（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。

11.1 エラー・メッセージの概要

RA78K0R 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（Fxxxx）

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，ほか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) フェイタル・エラー（Exxxx）

実行プログラム・エラーが発生したため，ほかのエラーを検出後，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，フェイタル・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（Cxxxx）

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（Wxxxx）

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。

RA78K0R 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。

種類	説明
Fn0xx	コマンド行解析部のエラー
Fn9xx	ファイル・システムに関するエラー
Fn1xx	その他のアボート・エラー
Cnxxx	内部エラー
En2xx	文の記述に関するエラー
En3xx	式に関するエラー
En4xx	シンボルに関するエラー
En5xx	セグメントに関するエラー
En6xx	制御命令，マクロに関するエラー
Wnxxx	各種ワーニング・エラー

備考 n = 2 ～ 6

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

11.2 アセンブラのエラー・メッセージ

表 11-1 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，またはフォルダが含まれています。
	ユーザの処置	存在するドライブ，およびフォルダ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。

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。

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，デフォルト・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することができません。
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。
F2115	メッセージ	Too long source line
	原因	1 行の長さが制限（2048 文字）を越えています。
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。

エラー番号	エラー・メッセージ	
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。
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。
F2209	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越えています。
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	byte として記述可能な範囲（00H - 0FFH）を越えました。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。

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（80H - BFH）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（0FFE20H - 0FFF1FH）を越えました。
E2316	メッセージ	Operand out of range（$!addr20）
	原因	addr20 として記述可能な範囲（00000H - 0FFFFFH）を越えたか，あるいは分岐命令の次のアドレスからの相対距離を計算した結果，（-32768 ～ +32767）の範囲を越えています。
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。
E2319	メッセージ	Operand out of range (word)
	原因	word として記述可能な範囲（0000H - 0FFFFH）を越えました。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。
E2320	メッセージ	Operand out of range (20bit)
	原因	20bit として記述可能な範囲（00000H - 0FFFFFH）を越えました。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。
E2321	メッセージ	Operand out of range (addr20)
	原因	addr20 として記述可能な範囲（0000H - 0FFFFFH）を越えました。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。
E2322	メッセージ	Illegal operand, 2ndSFR is used as addr16
	原因	オペランドの記述が不正です。 2ndSFR は，addr16 として使われます。
	ユーザの処置	オペランドを addr16 と同様な記述へ変更してください。
E2323	メッセージ	Illegal operand, 2ndSFR.bit is used as addr16.bit
	原因	オペランドの記述が不正です。 2ndSFR.bit は，addr16.bit として使われます。
	ユーザの処置	オペランドを addr16.bit と同様な記述へ変更してください。
E2324	メッセージ	Illegal operand, SFR can't be used as addr16
	原因	オペランドの記述が不正です。 SFR は，addr16 として使うことはできません。
	ユーザの処置	オペランドを SFR で記述してください。

エラー番号	エラー・メッセージ	
E2325	メッセージ	Illegal operand, SFR.bit can't be used as addr16.bit
	原因	オペランドの記述が不正です。 SFR.bit は，addr16.bit として使うことはできません。
	ユーザの処置	オペランドを SFR.bit で記述してください。
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	Operand out of range (addr20)
	原因	addr20 として記述可能な範囲（00000H - 0FFFFFH）を越えたか，あるいは分岐命令の次のアドレスからの相対距離を計算した結果，（-128 ～ +127）の範囲を越えています。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。
E2328	メッセージ	Operand out of range (n)
	原因	指定された値が，n（1 - 7）の範囲を越えました。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。
E2329	メッセージ	Operand out of range (n)
	原因	指定された値が，n（1 - 15）の範囲を越えました。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。
E2330	メッセージ	Operand out of range (addr16 / BR or CALL)
	原因	addr16 として記述可能な範囲（0H - FFFFH）を越えました。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。
E2331	メッセージ	Operand out of range (addr16 / NUMBER)
	原因	addr16（数値定数，および NUMBER 属性のシンボル）として記述可能な範囲（0H - FFFFH）を越えました。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。
E2332	メッセージ	Operand out of range (!addr16 / ADDRESS)
	原因	addr16（ADDRESS 属性のシンボル）として記述可能な範囲を越えました。
	ユーザの処置	以下の (1)，または (2) の記述可能な範囲内のオペランドに変更してください。 (1) F0000H ～ FFFFFH (2) MAA に 0 を設定したときにミラーされる領域，または MAA に 1 を設定したときにミラーされる領域 ミラー領域の詳細については，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。
E2333	メッセージ	Operand out of range (ES:!addr16 / ADDRESS)
	原因	ES:!addr16 (ADDRESS 属性のシンボル) として記述可能な範囲（0H - FFFFFH）を越えました。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。

エラー番号	エラー・メッセージ	
E2334	メッセージ	Operand out of range (!addr16.bit / ADDRESS)
	原因	!addr16.bit として記述可能な範囲を越えました。
	ユーザの処置	以下の (1)，または (2) の記述可能な範囲内のオペランドに変更してください。 (1) F0000H - FFFFFH (2) MAA に 0 を設定したときにミラーされる領域，または MAA に 1 を設定したときにミラーされる領域 ミラー領域の詳細については，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。
E2335	メッセージ	Operand out of range (ES:!addr16.bit / ADDRESS)
	原因	ES:!addr16.bit として記述可能な範囲（0H - FFFFFH）を越えました。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。
E2336	メッセージ	Operand out of range (addr / BR or CALL)
	原因	疑似命令 BR/CALL のオペランド addr として記述可能な範囲を越えました。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。
E2337	メッセージ	Illegal mnemonic, use another mnemonic or option -COMPATI
	原因	78K0R で使用できない 78K0 の命令を使っています。
	ユーザの処置	別の命令で記述するか，-compati オプションを使ってください。
E2338	メッセージ	Operand out of range (EQU operand)
	原因	疑似命令 EQU のオペランドとして記述可能な範囲（0H - 0FFFFFH）を越えました。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。
E2339	メッセージ	Operand out of range (word / ADDRESS)
	原因	word（ADDRESS 属性のシンボル）として記述可能な範囲を越えました。
	ユーザの処置	以下の (1)，または (2) の記述可能な範囲内のオペランドに変更してください。 (1) F0000H ～ FFFFFH (2) MAA に 0 を設定したときにミラーされる領域，または MAA に 1 を設定したときにミラーされる領域 ミラー領域の詳細については，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。
E2340	メッセージ	Operand out of range (ES:word / ADDRESS)
	原因	ES:word（ADDRESS 属性のシンボル）として記述可能な範囲（0000H - 0FFFFFH）を越えました。
	ユーザの処置	記述可能な範囲内のオペランドに変更してください。

エラー番号	エラー・メッセージ	
E2341	メッセージ	Illegal size for Option Bytes
	原因	ユーザ・オプション・バイト，およびオンチップ・デバッグ・オプション・バイトを指定するセグメントが 4 バイトで指定されていません。
	ユーザの処置	ユーザ・オプション・バイト，およびオンチップ・デバッグ・オプション・バイトを指定するセグメントを 4 バイトで指定してください。
E2342	メッセージ	Illegal value for Option Bytes
	原因	ユーザ・オプション・バイト，およびオンチップ・デバッグ・オプション・バイトを指定するセグメントに配置した値が不正です。
	ユーザの処置	正しい値を配置してください。配置する値については，デバイスのユーザーズ・マニュアルを参照してください。
E2343	メッセージ	Illegal Option Bytes segment
	原因	ユーザ・オプション・バイト，およびオンチップ・デバッグ・オプション・バイトを指定するセグメントが複数指定されています。
	ユーザの処置	ユーザ・オプション・バイト，およびオンチップ・デバッグ・オプション・バイトを指定するセグメントを 1 つにしてください。
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。
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。

エラー番号	エラー・メッセージ	
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ラ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。
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。

エラー番号	エラー・メッセージ	
E2516	メッセージ	Option Bytes is not supported for this device
	原因	オプション・バイトは，指定されているデバイスでは使用できません。
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。
E2607	メッセージ	Missing ELSE or ENDIF
	原因	IF 文，または _IF 文に対となる ELSE 文，ENDIF 文の対応がとれていません。
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。
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。

エラー番号	エラー・メッセージ	
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。

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	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。

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	クロスリファレンス・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	クロスリファレンス・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，デバッ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・ファイル中のデバッグ情報が不正です。
	ユーザの処置	コンパイルを，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・ファイルをオープンすることができません。
F2902	メッセージ	Can't open parameter file　'ファイル名 '
	原因	パラメータ・ファイルをオープンすることができません。
F2903	メッセージ	Can't open include file　'ファイル名 '
	原因	インクルード・ファイルをオープンすることができません。
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	オーバーレイ・ファイルをオープンすることができません。
	ユーザの処置	オーバーレイ・ファイルがアセンブラの実行形式と同じフォルダ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	オーバーレイ・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・ファイルをオープンすることができません。
	ユーザの処置	フォルダに空き領域のあるディスクを使用してください。
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・ファイルをオープンすることができません。
	ユーザの処置	フォルダに空き領域のあるディスクを使用してください。

エラー番号	エラー・メッセージ	
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・ファイルをオープンすることができません。
	ユーザの処置	フォルダ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・ファイルをオープンすることができません。
	ユーザの処置	フォルダ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	オーバーレイ・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・ファイルをほかのフォルダに出力するか，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・ファイルをほかのフォルダに出力するか，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・ファイルをほかのフォルダに出力するか，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・ファイルをほかのフォルダに出力するか，指定したディスクに空き領域を作ってください。
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。

エラー番号	エラー・メッセージ	
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。

11.3 リンカのエラー・メッセージ

表 11-2 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。
F3002	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。
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，またはフォルダが含まれています。
	ユーザの処置	存在するドライブ，およびフォルダ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。

エラー番号	エラー・メッセージ	
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。
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。
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。

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。
E3102	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。 リンカの扱うことができるアセンブル，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，オーバーレイ・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ーレイ・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。

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	Too Many line number data (up to 65535/0FFFFH) in the same name segment 'segment'
	原因	1 セクションの最大ライン・ナンバ・エントリ数 65535 を越えています。
E3118	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オプション（-common）が指定されているため，対象デバイスを判別することができません。
	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オプション（-common）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。

エラー番号	エラー・メッセージ	
E3205	メッセージ	Name 'name ' 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 'segment name ' 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）。
E3207	メッセージ	Segment 'segment name ' 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 'segment name ' may not change attribute
	原因	アセンブル時に再配置属性を AT xxxxxH' 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 'segment name ' may not change arrangement
	原因	アセンブル時に再配置属性を AT xxxxxH' 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 'segment name ' 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。
E3212	メッセージ	Default segment can't allocate to memory-ignored
	原因	デフォルト・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。
	ユーザの処置	ROM 範囲内に -gb，-gi，-go のデータを配置することができるかを確認してください。
E3301	メッセージ	Relocatable object code address out of range（file 'file name ', segment 'segment name ', address xxxxxH, type 'addressing type '）
	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 'file name ', segment 'segment name '）
	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バッ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。

エラー番号	エラー・メッセージ	
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 xxxxxH, 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（file ' ファイル名 ', segment 'セグメント名 ', symbol ' シンボル名 ', address xxxxxH, 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，ソースを記述してください。
E3305	メッセージ	Even value expected（file ' ファイル名 ', segment 'セグメント名 ', symbol ' シンボル名 ', address xxxxxH, 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 xxxxxH, 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。

エラー番号	エラー・メッセージ	
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，ほかのファイルで 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）。
E3408	メッセージ	Can't find -A symbol
	原因	リンカ・オプションのプログラム・エントリ・アドレス指定の -a 以降に記述したシンボルが，パブリック・シンボルに存在しません。
E3409	メッセージ	-A symbol ’ シンボル名’is unmatched type
	原因	リンカ・オプションのプログラム・エントリ・アドレス指定の -a によりサーチしたシンボルのタイプに，誤りがありました。
	ユーザの処置	プログラム・エントリ・アドレス指定によってサーチするシンボルが，許されるタイプのシンボルにしてください。
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。
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	Compiler options are mixed in file ’ ファイル名 1’ First defined in file ’ ファイル名 2’
	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（CC78K0R，RA78K0R），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。

エラー番号	エラー・メッセージ	
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）をリンクを行うと必ず出力します。
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（CC78K0R，RA78K0R）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・メモリ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。
F3426	メッセージ	Multiple input BOOT file "ファイル名 1'．First input file ' ファイル名 2'
	原因	フラッシュ・メモリ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。
	ユーザの処置	ブート領域 ROM プログラムのロード・モジュール・ファイルを 1 つだけ指定してください。
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。
E3428	メッセージ	FLASH start address larger than ROM max address
	原因	フラッシュ・メモリ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・メモリ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・メモリ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。
F3430	メッセージ	Different FLASH address in file ' ファイル名 1'．First specified in file ' ファイル名 2'
	原因	入力ファイルのフラッシュ・メモリ領域の先頭アドレスが，すべて同じではありません。
	ユーザの処置	-zb オプションには，すべて同じ値を指定してください。

エラー番号	エラー・メッセージ	
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。
W3434	メッセージ	Can't specify User Option Bytes/On-Chip Debug Option Bytes/Security ID with LMF
	原因	入力ファイルにロード・モジュール・ファイルを指定した場合，-gb/-go/-gi オプションを指定することはできません。
	ユーザの処置	入力ファイルにロード・モジュール・ファイルを指定して再リンクを行う場合，-gb/-go/-gi オプションは指定しないでください。
F3435	メッセージ	ext_table address in file '%s'. First specified in file '%s'
	原因	C ソースで指定した“#pragma ext_table”の値が不正です。
	ユーザの処置	スタート・アップ・ルーチンで指定した“ITBLTOP”の値と，C ソースで指定した“#pragma ext_table”の値を，すべて同じ値にしてください。
F3502	メッセージ	Too many segment (up to 65535/0FFFFH)
	原因	入力セグメントの総数が 65,535 個を越えました。
F3901	メッセージ	Can't open overlay file 'ファイル名 '
	原因	オーバーレイ・ファイルをオープンすることができません。
	ユーザの処置	オーバーレイ・ファイルが正しいフォルダ（実行形式プログラムがあるフォルダ）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することができません。
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することができません。
	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。
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。

エラー番号	エラー・メッセージ	
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。
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（CC78K0R，RA78K0R）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。

11.4 オブジェクト・コンバータのエラー・メッセージ

表 11-3 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，またはフォルダが含まれています。
	ユーザの処置	存在するドライブ，およびフォルダ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。

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。

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@sym 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	xxxxxH-yyyyyH overlapped
	原因	xxxxxH から yyyyyH 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。

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	ファイルをオープンすることができません。
F4901	メッセージ	Can't close file ’ファイル名 ’
	原因	ファイルをクローズすることができません。
F4902	メッセージ	Can't read file ’ファイル名 ’
	原因	ファイルを正しく読むことができません。
F4903	メッセージ	Can't access file ’ファイル名 ’
	原因	ファイルを正しく読み込む，または書き込むことができません。
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を書き込むことができません。
F4905	メッセージ	Can't open overlay file ‘ ファイル名 ’
	原因	オーバーレイ・ファイルをオープンすることができません。
	ユーザの処置	オーバーレイ・ファイルが実行形式と同じフォルダ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。

11.5 ライブラリアンのエラー・メッセージ

表 11-4 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，またはフォルダが含まれています。
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。

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。
F5019	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。
F5020	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に失敗しました。
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。
F5022	メッセージ	Memory allocation failed
	原因	メモリ・アロケーションに失敗しました。
F5023	メッセージ	Illegal character ’,’before file name
	原因	入力ファイルの前に必要な’,’があります。
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。

エラー番号	エラー・メッセージ	
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，またはほか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。
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，またはほか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，またはほか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。

エラー番号	エラー・メッセージ	
F5901	メッセージ	File open error（file：ファイル名）
	原因	ファイルをオープンすることができません。
F5902	メッセージ	File read error（file：ファイル名）
	原因	ファイルを正しく読むことがでません。
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を書き込むことができません。
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	ファイルをクローズすることができません。

11.6 リスト・コンバータのエラー・メッセージ

表 11-5 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，またはフォルダが含まれています。
	ユーザの処置	存在するドライブ，およびフォルダ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。

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 78K0R ’ファイル名 ’
	原因	入力ファイル名が 78K0R 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。
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。

エラー番号	エラー・メッセージ	
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・ファイルをオープンすることができません。
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。
F6901	メッセージ	File open error has occured　'ファイル名 '
	原因	ファイルをオープンすることができません。

エラー番号	エラー・メッセージ	
F6902	メッセージ	File read error has occured 'ファイル名'
	原因	ファイルを正しく読むことができません。
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を書き込むことができません。
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。

11.7 PM+ のエラー・メッセージ

PM+ 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+ のその他のエラー・メッセージについては，PM+ のオンライン・ヘルプを参照してください。

11.7.1 アセンブラ（RA78K0R）

表 11-6 アセンブラ（RA78K0R）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	RA78K0R.EXE が PATH 環境変数で示されるフォルダに登録されていません。
	原　因	RA78K0R.EXE 実行形式が規定のフォルダにありません。
	ユーザの処置	RA78K0R 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。

エラー種別	エラー・メッセージ	
!	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。
!	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。

エラー種別	エラー・メッセージ	
!	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。
!	メッセージ	（シンボル） シンボル定義で指定した文字に不適切な文字が含まれています。
	原　因	シンボル定義で，不正な文字が記述されています。
	ユーザの処置	適切な文字でシンボルを記述して，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	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。

11.7.2 リンカ（LK78K0R）

表 11-7 リンカ（LK78K0R）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	LK78K0R.EXE が，PATH 環境変数で示されるフォルダに登録されていません。
	原　因	LK78K0R.EXE 実行形式が規定のフォルダにありません。
	ユーザの処置	RA78K0R 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。

エラー種別	エラー・メッセージ	
!	メッセージ	オンチップ・デバッグ・オプション・バイトで指定した制御値が不正です。
	原　因	オンチップ・デバッグ・オプション・バイトで指定した制御値が，指定デバイスで指定可能な値で記述されていません。
	ユーザの処置	デバイスのユーザーズ・マニュアルで，指定可能な値を確認してください。
	ボタン	“OK”　　：メッセージを閉じます。
!	メッセージ	オンチップ・デバッグ・オプション・バイトで指定した制御値が不正です。 オンチップ・デバッグ・オプション・バイトを指定する場合は，制御値を入力してください。
	原　因	オンチップ・デバッグ・オプション・バイトの制御値が入力されていません。
	ユーザの処置	オンチップ・デバッグ・オプション・バイトの制御値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	オンチップ・デバッグ・オプション・バイトで指定した制御値が不正です。 制御値は 16 進数で指定してください。
	原　因	オンチップ・デバッグ・オプション・バイトで指定した制御値の指定形式が，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	オンチップ・デバッグ・オプション・バイトで指定したスタート・アドレスが不正です。 指定可能な範囲は　0h　≦　スタート・アドレス　≦　0FFFFFh バイト　です。
	原　因	オンチップ・デバッグ・オプション・バイトで指定したスタート・アドレス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	オンチップ・デバッグ・オプション・バイトで指定したスタート・アドレスが不正です。 オンチップ・デバッグ・オプション・バイトを指定する場合は，スタート・アドレスを入力してください。
	原　因	オンチップ・デバッグ・オプション・バイトのスタート・アドレスが入力されていません。
	ユーザの処置	オンチップ・デバッグ・オプション・バイトのスタート・アドレスを入力してください。
	ボタン	“OK”　　：メッセージを閉じます。

エラー種別	エラー・メッセージ	
!	メッセージ	オンチップ・デバッグ・オプション・バイトで指定したスタート・アドレスが不正です。 スタート・アドレスは 16 進数で指定してください。
	原　因	オンチップ・デバッグ・オプション・バイトで指定したスタート・アドレスの指定形式が，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	オンチップ・デバッグ・オプション・バイトで指定し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・デバッグ・オプション・バイトで指定したサイズが，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	オンチップ・デバッグ・オプション・バイトで指定したサイズが不正です。 サイズは 10 進数で指定してください。
	原　因	オンチップ・デバッグ・オプション・バイトで指定したサイズの指定形式が，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	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。

エラー種別	エラー・メッセージ	
!	メッセージ	ユーザ・オプション・バイトで指定した値が不正です。 指定可能な範囲は 0h ≦ ユーザ・オプション・バイト ≦ 0FFFFFFFFh バイト です。
	原 因	ユーザ・オプション・バイトで指定した値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	ユーザ・オプション・バイトで指定した制御値が不正です。 ユーザ・オプション・バイトを指定する場合は，値を入力してください。
	原 因	ユーザ・オプション・バイトの値が入力されていません。
	ユーザの処置	ユーザ・オプション・バイトの値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	ユーザ・オプション・バイトで指定した値が不正です。 値は 16 進数で指定してください。
	原 因	ユーザ・オプション・バイトで指定した値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	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 64 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。
!	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。

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。

11.7.3 オブジェクト・コンバータ（OC78K0R）

表 11-8 オブジェクト・コンバータ（OC78K0R）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	OC78K0R.EXE が，PATH 環境変数で示されるフォルダに登録されていません。
	原　因	OC78K0R.EXE 実行形式が規定のフォルダにありません。
	ユーザの処置	RA78K0R 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	オブジェクト充てんの指定形式が不正です。
	原　因	オブジェクトの充てん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	充てん値として，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　≦　0FFEFFH 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　≦　0FFF00H 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。
!	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。

11.7.4 ライブラリアン（LB78K0R）

表 11-9 ライブラリアン（LB78K0R）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	LB78K0R.EXE が，PATH 環境変数で示されるフォルダに登録されていません。
	原　因	LB78K0R.EXE 実行形式が規定のフォルダにありません。
	ユーザの処置	RA78K0R 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。

表 11-10 ライブラリアン単体起動用実行形式（lb78k0rp.exe）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名が指定されていません。
	原　因	ライブラリ・ファイル名の指定エディット・ボックスにファイル名が指定されていません。
	ユーザの処置	ファイル名を正しく指定してください。
	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	指定した出力パスが，存在していません。
	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 ”　　：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	フォルダの作成を選択した場合に，フォルダの作成時にエラーが発生しました。
	ユーザの処置	システムのフォルダ構造に問題がないか確認してください。
	ボタン	“ OK ”　　：メッセージを閉じます。
？	メッセージ	LB 実行時にエラーが発生しました。 ログを表示しますか？
	原　因	LB 実行時に，何らかのエラーが発生しました。
	ユーザの処置	“OK” ボタン（LB 実行ログを表示），“ キャンセル ” ボタン（表示しない）のどちらかを選択してください。
	ボタン	“ OK ”　　：LB 実行ログを表示します。 “ キャンセル ”：メッセージを閉じます。
!	メッセージ	左のリストボックスでファイルが選択されていません。
	原　因	add，replace サブコマンドの実行に先立ち，左のリスト・ボックスでファイルが 1 つも選択されていません。
	ユーザの処置	1 つ以上のファイルを選択してください。
	ボタン	“ OK ”　　：メッセージを閉じます。
!	メッセージ	右のリストボックスでファイルが選択されていません。
	原　因	pick，delete サブコマンドの実行に先立ち，右のリスト・ボックスでファイルが 1 つも選択されていません。
	ユーザの処置	1 つ以上のファイルを選択してください。
	ボタン	“ OK ”　　：メッセージを閉じます。

エラー種別	エラー・メッセージ	
?	メッセージ	ライブラリ内の選択したモジュールを削除しますか？
	原　因	ライブラリ内のモジュール削除の確認メッセージです。
	ユーザの処置	“OK” ボタン（削除する），“ キャンセル ” ボタン（削除しない）のどちらかを選択してください。
	ボタン	“ OK ”　　：ライブラリ内の選択したモジュールを削除します。 “ キャンセル ”：メッセージを閉じます。
!	メッセージ	出力ファイル名の指定が不正です。
	原　因	list 出力ファイル名の指定欄に，正しくファイル名が指定されていません。
	ユーザの処置	出力ファイル名を正しく指定してください。
	ボタン	“ OK ”　　：メッセージを閉じます。
!	メッセージ	一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　≦　260　です。
	原　因	list ファイル出力の一行文字数として，入力可能範囲外の数値が記述されています。
	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	一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	list ファイル出力の一ページ行数として，入力可能範囲外の数値が記述されています。
	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。

11.7.5 リスト・コンバータ（LC78K0R）

表 11-11 リスト・コンバータ（LC78K0R）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	LC78K0R.EXE が，PATH 環境変数で示されるフォルダに登録されていません。
	原　因	LC78K0R.EXE 実行形式が規定のフォルダにありません。
	ユーザの処置	RA78K0R 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，RA78K0R 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 k0rmain.asm

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

PUBLIC  MAIN , START
EXTRN   CONVAH
EXTRN   _@STBEG

DATA    DSEG    AT      0FFE20H
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 , #LOWW ( HDTSA )    ; set hex 2-code data in HL registor

        CALL    !CONVAH                 ; convert ASCII <- HEX
                                        ; output BC-register <- ASCII code
        MOVW    DE , #LOWW ( 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 k0rsub.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 lower 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
```

付録 B 使用上の注意

この章では，RA78K0R 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

RA78K0R 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 RA78K0R パッケージには含まれていません。別途，入手してください。

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 から取得可能です。

http://www.necel.com/micro/ods/jpn/index.html

(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使用してください。

本オプションは，デフォ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 DB，DW，DG 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です。

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・アドレスを指定した場合，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。内部 RAM 領域（ED800H - FFFFFH）には充てんを行いません。

記述形式　：-u *充てん値* [,[*スタート・アドレス*], *サイズ*]

備考　[　] 内は省略可能です。

(5) メモリ・ディレクティブ

各デバイスのデフォルトのメモリ領域名は，消去できません。

使用しないデフォ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フォルトの領域に割り付けられるものもあるため，領域名を変更する際には注意してください。

なお，デフォ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) デバッグ・オプション

C コンパイラでデバッグ情報を出力して，コンパイルした場合，その出力アセンブル・ソースをアセンブルするときには，デバッ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）。デバッグ情報を出力すると，C コンパイラ・ソース・レベルでデバッグすることができないことがあります。

(7) CC78K0R

CC78K0R 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・デバッグを行う場合，何点かの注意事項があります。

詳細については，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) ID78K0R-QB，SM+ for 78K0R

ID78K0R-QB，SM+ for 78K0R でデバッグを行う場合に，シンボル数，ソース行数の制限に関して，ID78K0R-QB，SM+ for 78K0R の制限以内で使用してください。

詳細については，デバッガ / 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)SFR 名の EQU 定義

EQU 疑似命令のオペランドには SFR 名を指定することができますが，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。

(11)ネットワーク使用時

一時ファイルを作成するフォルダ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，このような競合は避けてください。

(12)セルフ・プログラミング使用時

セルフ・プログラミングのための各ツールのオプションについて，次に示します。

(a) アセンブラ

RA78K0 では，セルフ・プログラミング用ファームウエアがデバイス・ファイルの内蔵 ROM 領域外であるとき，“CALL !8100H”の記述に対してエラーとなる場合があり，-self オプションを指定することでこのエラーを回避していました。
これに対し，RA78K0R では，全空間において“CALL !xxxxxH”の記述が可能であるため，セルフ用ファームウエアを呼び出す際は“CALL !!addr20”を記述してください。

(b) リンカ

ブート領域用ロード・モジュール・ファイルを作成する場合は，-zb オプションを使用します。
-zb オプションの後には，フラッシュ ROM 領域の先頭アドレスを指定してください。

次に，フラッシュ領域用ロード・モジュール・ファイルを作成する場合は，ブート領域用ロード・モジュール・ファイルとフラッシュ領域用オブジェクト・モジュール・ファイルを入力し，再リンクします。
なお，フラッシュ領域用オブジェクト・モジュール・ファイルは，-zb オプションで指定したアドレス以降に配置してください。

(c) オブジェクト・コンバータ

ブート領域用ロード・モジュール・ファイルとフラッシュ領域用オブジェクト・モジュール・ファイルを入力し，再リンクして出力されたロード・モジュール・ファイルを，OC78K0R に入力します。
このとき，-zf オプションを指定することで，ブート領域用 HEX フォーマット・ファイル（*.hxb）とフラッシュ領域用 HEX フォーマット・ファイル（*.hxf）を分割して出力することができます。

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一覧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。
このオプション一覧は，索引としても使用することができます。

C.1 アセンブラ・オプション

表 C-1 アセンブラ・オプション

分類	記述形式	機能	ほか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	省略することはできません。
オブジェクト・モジュール・ファイル出力指定	-o[*出力ファイル名*]	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。		
デバッグ情報出力指定	-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・デバッ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*] …（複数指定可能）	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 INC78K0R で指定したパス
アセンブル・リスト・ファイル出力指定	-p[*出力ファイル名*]	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* .prn
	-np	アセンブル・リスト・ファイル出力しません。		

分類	記述形式	機能	ほか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・ファイル中にクロスリファレンス・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・リストのファイルの 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・リストのファイルの 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。	-ll0
	-lh *文字列*	アセンブル・リストの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・リストのファイルの最後に，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[*出力ファイル名*]	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。		
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となります。
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 で指定したパス

分類	記述形式	機能	ほかのオプションとの関係	省略時解釈
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。	-zs
	-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	注
シンボル定義指定	-d *シンボル名* [= *数値*][, *シンボル名* [= *数値*] …]	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	78K0R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	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。
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
78K0シリーズ互換マクロ	-compati	78K0シリーズ用アセンブラで作成したアセンブラ・ソース・ファイルをアセンブル可能にします。	独立	-ncompati
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。	ほかのオプションをすべて無効にします。	表示しません。

注 デバイス・ファイルを読み込むパスは，次の順序で調べて決定されます。

(1) デバイス・ファイル・インストーラで登録されたパス

(2) RA78K0Rの起動されたパス

(3) カレント・フォルダ

(4) 環境変数 PATH

C.2 リンカ・オプション

表 C-2 リンカ・オプション

分類	記述形式	機能	ほか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[*出力ファイル名*]	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。		
デバッグ情報出力指定	-g	デバッグ情報をロード・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[*出力ファイル名*]	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 *入力ファイル名* .map
	-np	-p オプションを無効にします。		

分類	記述形式	機能	ほか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。	-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。	-ll0
	-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。		
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

分類	記述形式	機能	ほかのオプションとの関係	省略時解釈
ライブラリ・ファイル読み込みパス指定	-i *パス名* [, *パス名*] …（複数指定可能）	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 LIB78K0R で指定したパス
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となります。
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 で指定したパス
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	注
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	-w1
フラッシュ・メモリ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・メモリ領域の先頭アドレスを指定します。	なし	配置範囲の制限はありません。
オンチップ・デバッグ・オプション・バイト指定	-go *制御値* , *スタート・アドレス* [, *サイズ*]	オンチップ・デバッグ・オプション・バイトを指定します。	なし	オンチップ・デバッグを使用しません。C3H 番地はデバイス・ファイルの初期値
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。	なし	セキュリティ ID を設定しません。
ユーザ・オプション・バイト指定	-gb *ユーザ・オプション・バイト値*	ユーザ・オプション・バイトに設定する値を指定します。	なし	デバイス・ファイルの初期値
ミラー領域指定	-mi0，または -mi1	ミラー元のセグメントの配置先を指定します。	なし	-mi0
64K バイト境界配置指定	-ccza	64K バイト境界の最後の 1 バイトに配置を行うかどうかを指定します。	なし	-ccza（入力ファイルがアセンブラ出力ファイルのみの場合），-nccza（入力ファイルにコンパイラ出力ファイルがある場合）
	-nccza	-ccza オプションを無効にします。	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。	ほかのオプションをすべて無効にします。	表示しません。

注 デバイス・ファイルを読み込むパスは，次の順序で調べて決定されます。

(1) デバイス・ファイル・インストーラで登録されたパス

(2) LK78K0R の起動されたパス

(3) カレント・フォルダ

(4) 環境変数 PATH

C.3 オブジェクト・コンバータ・オプション

表 C-3 オブジェクト・コンバータ・オプション

分類	記述形式	機能	ほか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[*出力ファイル名*]	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
	-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[*出力ファイル名*]	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* .sym
	-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・アドレス*], *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[*出力ファイル名*]	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。		
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，または入力ファイル名の入力が可能となります。
HEX 形式指定	-ki	インテル標準 HEX 形式	独立	-kie
	-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（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）		

分類	記述形式	機能	ほか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	注
フラッシュ・メモリ内蔵製品用ファイル分割出力指定	-zf	フラッシュ・メモリ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	分割出力しません。
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。	ほかのオプションをすべて無効にします。	表示しません。

注 デバイス・ファイルを読み込むパスは，次の順序で調べて決定されます。

(1) デバイス・ファイル・インストーラで登録されたパス

(2) OC78K0R の起動されたパス

(3) カレント・フォルダ

(4) 環境変数 PATH

C.4 ライブラリアン・オプション

表 C-4 ライブラリアン・オプション

分類	記述形式	機能	ほか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。		-ll0
	-lf	リスト・ファイルの最後に，改頁コードを付加します。		-nlf
	-nlf	-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 により指定されたパス
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。	ほかのオプションをすべて無効にします。	表示しません。

C.5 リスト・コンバータ・オプション

表 C-5 リスト・コンバータ・オプション

分類	記述形式	機能	ほか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[*入力ファイル名*]	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[*入力ファイル名*]	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* .lmf
アブソリュート・アセンブル・リスト・ファイル出力指定	-o[*出力ファイル名*]	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[*出力ファイル名*]	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。		
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，または入力ファイル名の入力が可能となります。
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。	ほか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，索引としても使用することができます。

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ほか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

総合索引

N

O

P

R

S

T

U

W

Y

Z

【あ行】

【か行】

【さ行】

〔メ　モ〕

【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。