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

RA78K0S

アセンブラ・パッケージ　Ver.1.40以上

操作編

対象デバイス
78K0Sシリーズ

資料番号　U16656JJ1V0UM00（第1版）
発行年月　April 2003 NS　CP(K)

〔メ　モ〕

目次要約

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

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

PC/AT**は，米国**IBM Corp.**の商標です。**

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

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

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

- 本資料に記載されている内容は2003年4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

〔メ　モ〕

は じ め に

このマニュアルは，RA78K0S　アセンブラ・パッケージ（以降，RA78K0Sとします）を用いてソフトウエア開発を行われる方に，RA78K0S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RA78K0S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RA78K0S **アセンブラ・パッケージ　ユーザーズ・マニュアル　言語編**（U16657J）（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RA78K0Sに関する記述は，Ver.1.40以上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（78K0Sシリーズ）の機能およびインストラクションについて理解しているユーザを対象としています。

【構　　成】

このマニュアルは次のように構成されています。

第1章　概　　説

マイクロコンピュータの開発におけるRA78K0Sの役割など，RA78K0S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

RA78K0S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　RA78K0S**の実行**

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，RA78K0Sを動作させてみたい方は，この章からお読みください。

第4章　構造化アセンブラ

第5章　アセンブラ

第6章　リ ン カ

第7章　オブジェクト・コンバータ

第8章　ライブラリアン

第9章　リスト・コンバータ

第10章　プログラムの出力リスト

各プログラムが出力する各種リストのフォーマットについて説明します。

第11章　RA78K0S**の活用法**

RA78K0S**をうまく使うための方法を紹介します。**

第12章　エラー・メッセージ

各プログラムが出力するエラー・メッセージについて説明します。

付　　録

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

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

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

【読 み 方】

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

実際にRA78K0S**を使用する場合は，第3章**　RA78K0S**の実行をお読みください。**

各プログラムの操作に慣れてからは，付録の一覧表をご活用ください。

【注　　意】

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

●ファイル名の形式が異なります。

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

●マニュアルに記載されている実行例や環境設定の方法が異なります。

【凡　例】

このマニュアルの中で共通に使用される記号などの意味を示します。

M	**；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
CC78K0S　C**コンパイラ** Ver.1.50**以上**	**操作編**	U16654J	U16654E
	言語編	U16655J	U16655E
RA78K0S　**アセンブラ・パッケージ**　Ver.1.40**以上**	**操作編**	**このマニュアル**	U16656E
	言語編	U16657J	U16657E
	構造化アセンブリ言語編	U11623J	U11623E
SM78K0S　**システム・シミュレータ**	**操作編**	**作成予定**	**作成予定**
ID78K0S-NS　**統合ディバッガ**　Ver.2.52**以上**	**操作編**	U16584J	U16584E
78K/0S**シリーズ用**OS　MX78K0S	**基礎編**	U12938J	U12938E
PM plus Ver.5.10		U16569J	**作成予定**

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（1/2）

図の目次（2/2）

表の目次

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RA78K0S**の役割など，**RA78K0S**全体の機能概要について説明します。**

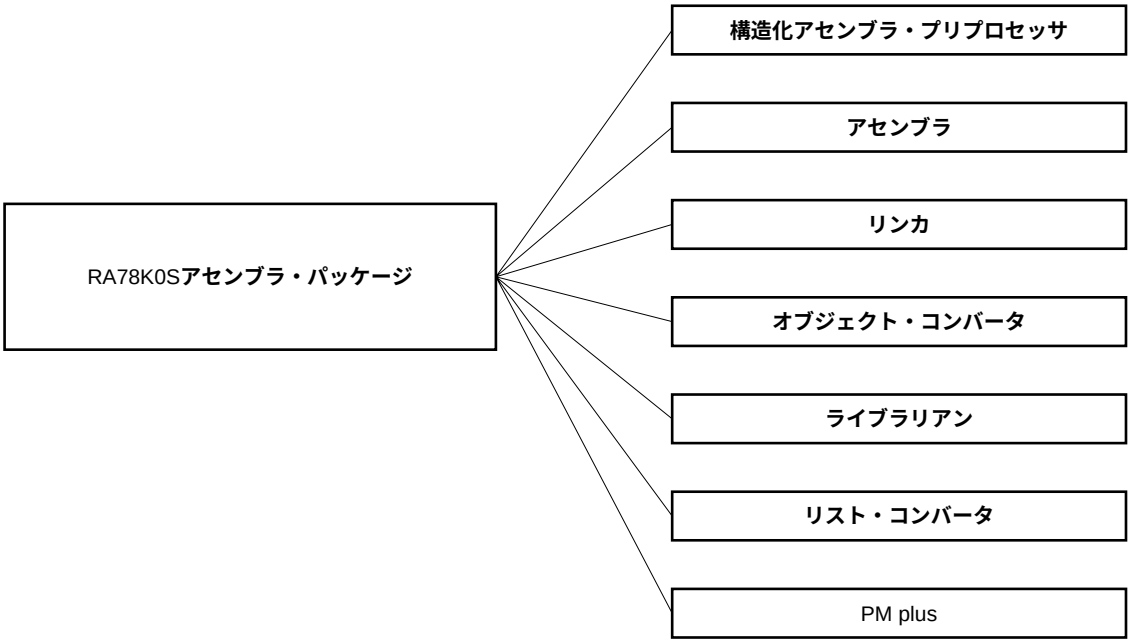
1. 1　アセンブラの概要

RA78K0Sアセンブラ・パッケージは，78K0Sシリーズのアセンブリ言語で記述されたソース・プログラムを機械語に変換する一連のプログラムの総称です。

RA78K0Sの中には，構造化アセンブラ・プリプロセッサ，アセンブラ，リンカ，オブジェクト・コンバータ，ライブラリアン，リスト・コンバータの6つのプログラムがあります。

さらに，エディット，コンパイル／アセンブル，リンクからディバグまでの一連の操作をWindows®上で簡単に行うことを可能にするPM plusもRA78K0Sに添付されています。

図1－1　RA78K0Sアセンブラ・パッケージ



1. 1. 1　アセンブラとは

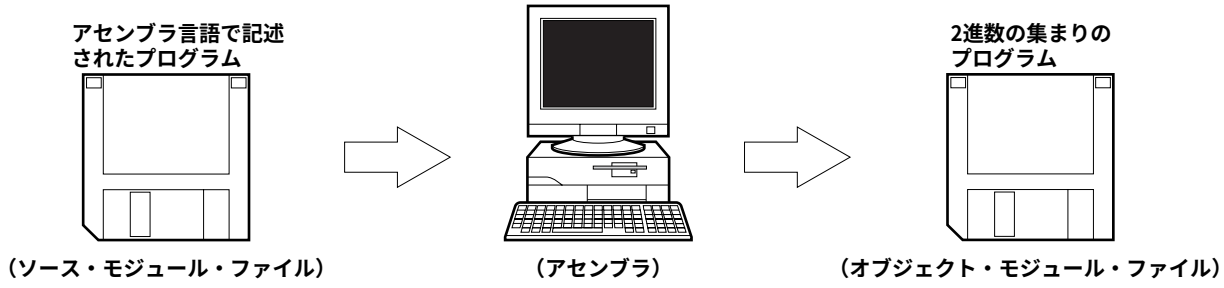
(1) アセンブリ言語と機械語

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

マイクロコンピュータに仕事をさせるためには，プログラムやデータが必要です。これを人間がプログラミングして，マイクロコンピュータのメモリ部に記憶させます。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，これを機械語といいます。機械語でプログラムを作るのは，人間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人間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言語体系をアセンブリ言語といいます。

マイクロコンピュータが扱えるのは機械語ですから，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。

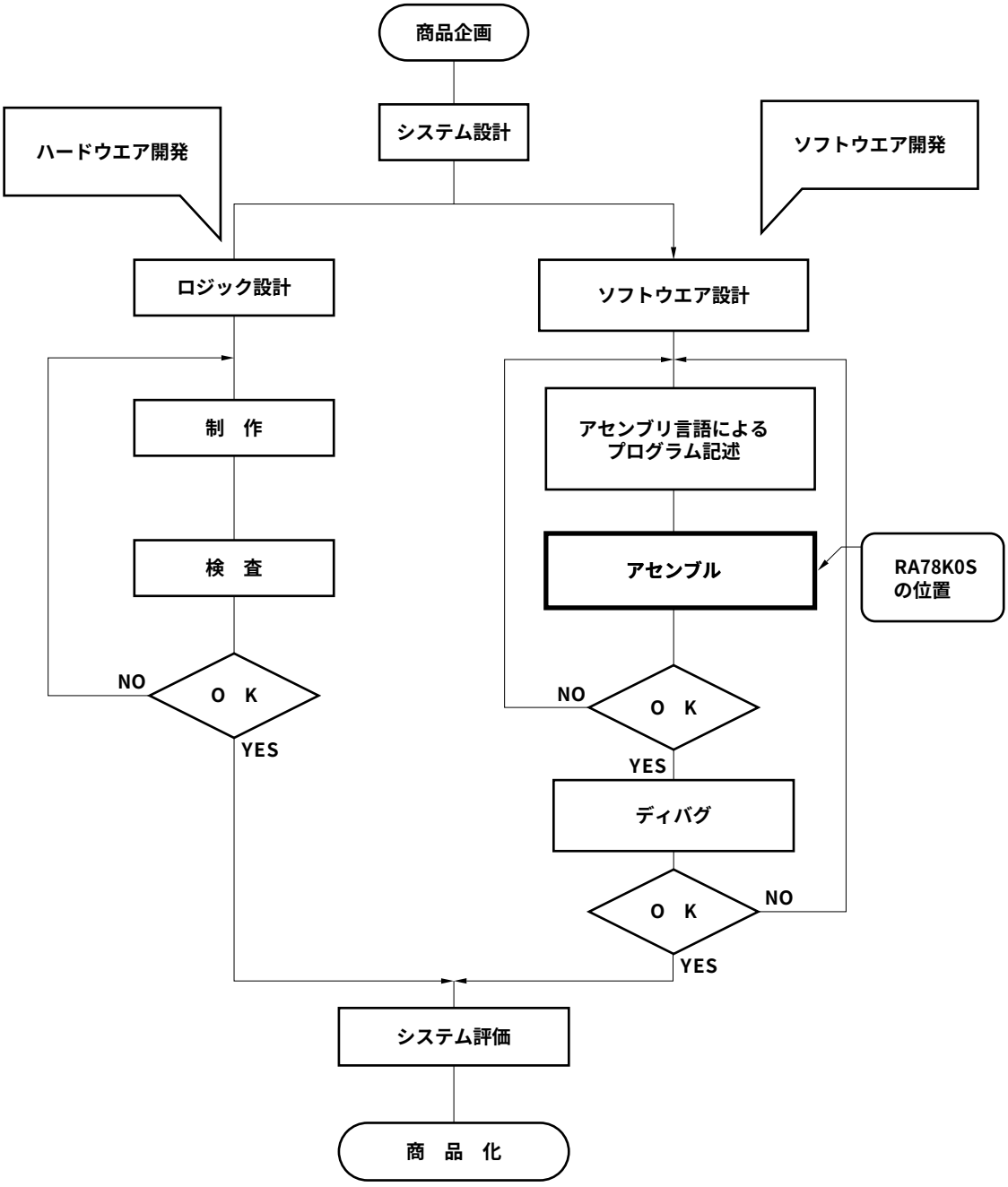
図1－2　アセンブラの流れ



(2) マイクロコンピュータ応用製品の開発とRA78K0Sの役割

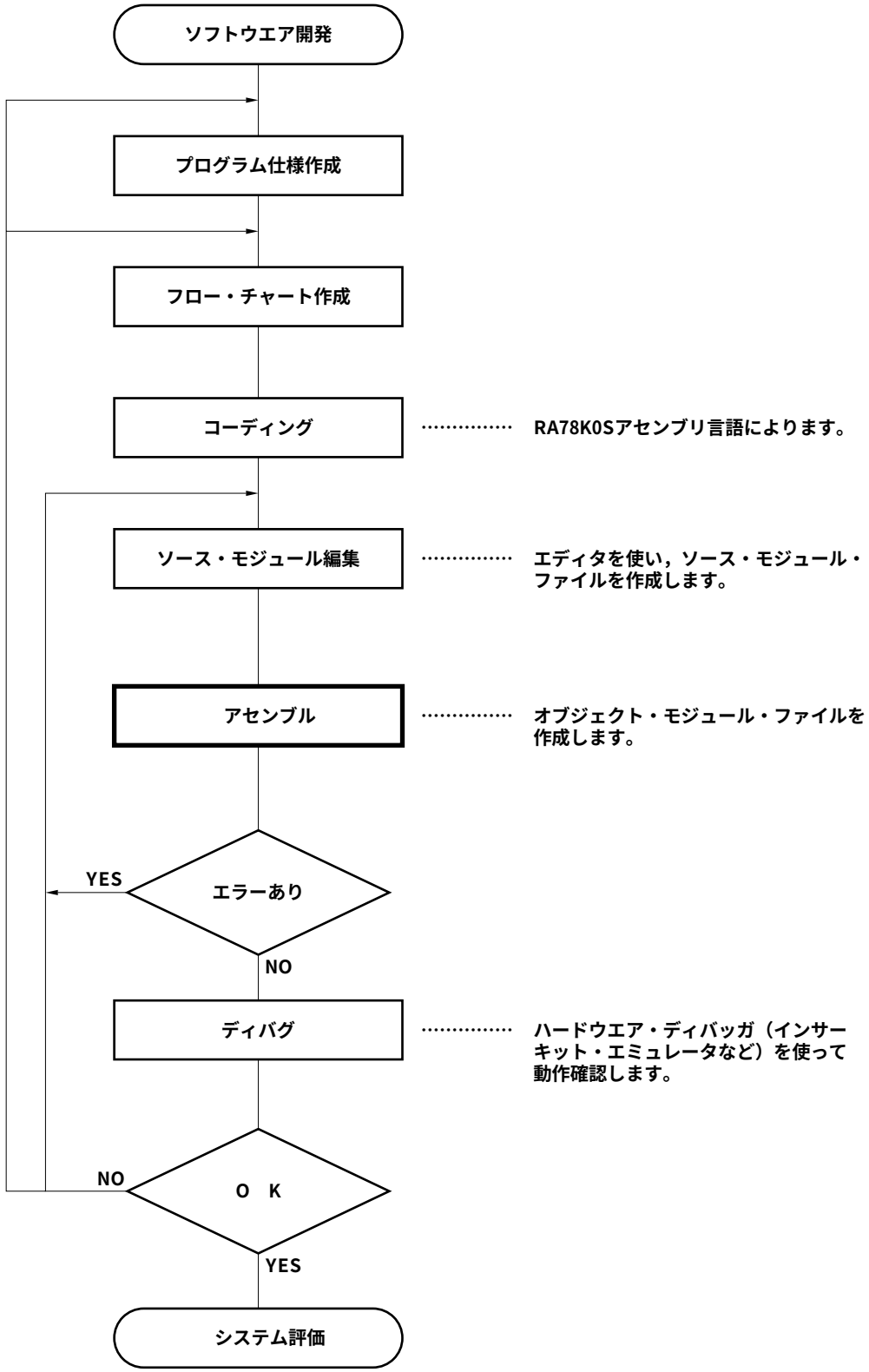
アセンブリ言語でのプログラミングは，製品開発のどこに位置するのかを図1－3 **マイクロコンピュータ応用製品の開発工程**に示します。

図1－3　マイクロコンピュータ応用製品の開発工程



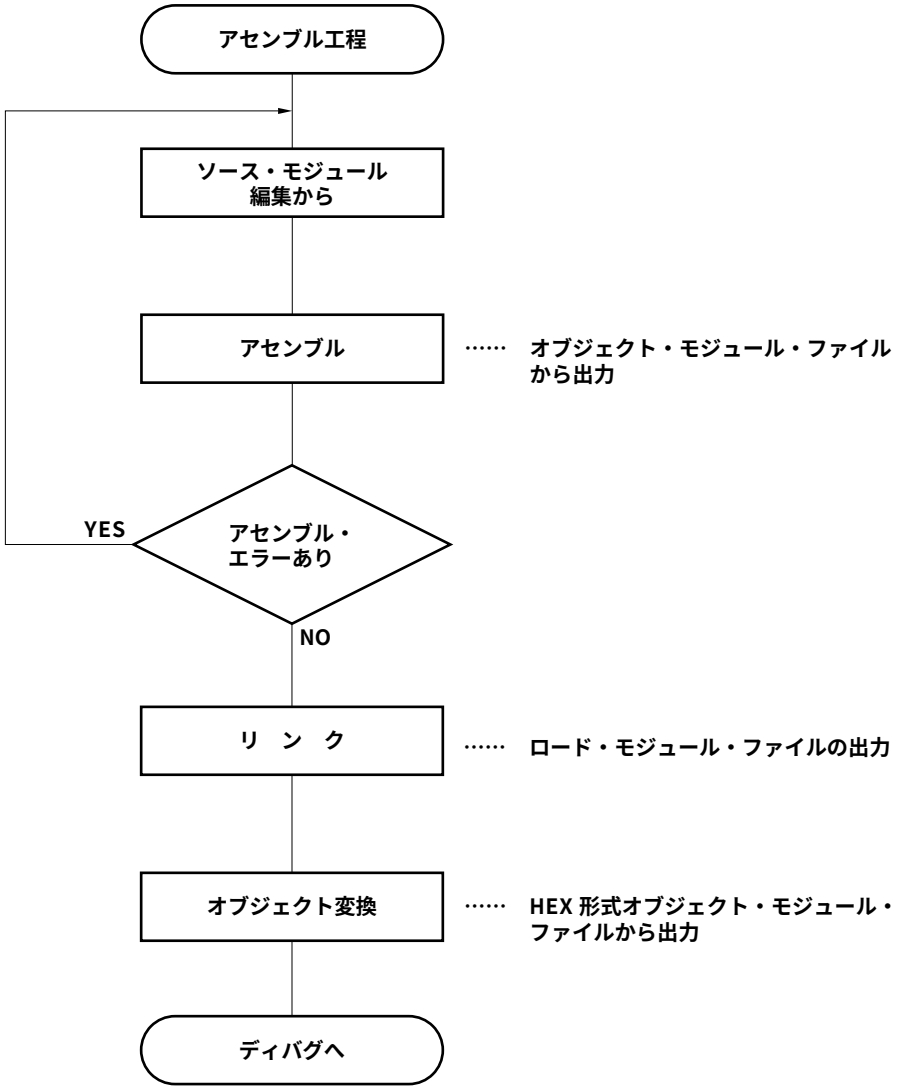
ソフトウエア開発の工程をもう少し詳しく，図1－4 **ソフトウエアの開発工程**で示します。

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，RA78K0Sを当てはめてみます。

図1－5　RA78K0Sのアセンブル工程



1. 1. 2　リロケータブル・アセンブラとは

アセンブラが変換した機械語は，マイクロコンピュータ内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，各機械語がメモリ中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」とに大別できます。

・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を，絶対的（アブソリュート）なアドレスに配置します。

・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，一時的なアドレスを与えます。絶対的なアドレスの決定は，リンカと呼ばれるプログラムにより行います。

アブソリュート・アセンブラで1つのプログラムを作成する際には，原則として一度にプログラミングしなければなりません。しかし，大きなプログラムを1つのまとまりとして一度に作成すると，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大変になります。そこで，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・プログラミングといいます。

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す。

リロケータブル・アセンブラを使用してモジュラ・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。

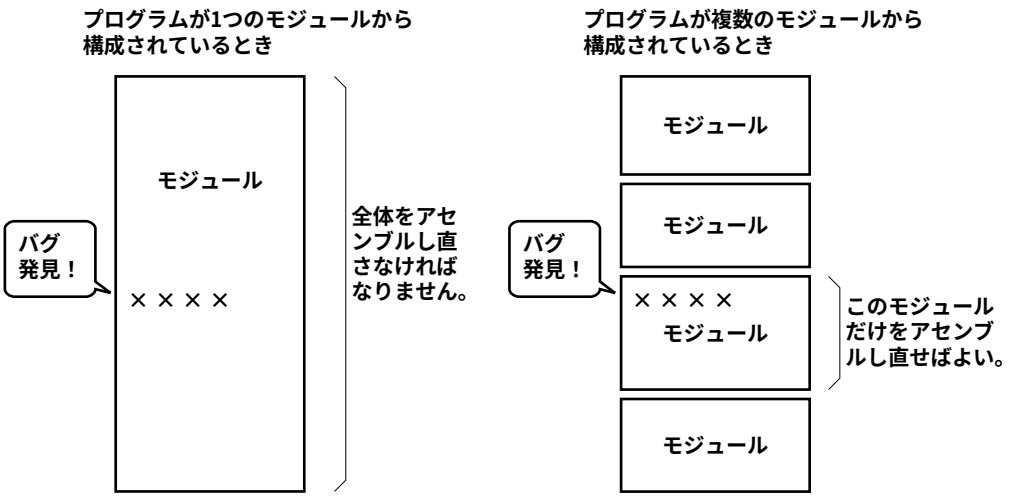
(1) 開発効率が上がる

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，プログラムを機能ごとにモジュール分割すれば，複数の人数でプログラムの並行開発ができ，開発効率が上がります。

また，プログラム中にバグが発見された場合，一部の修正の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ます。

これにより，ディバグ時間を短くすることができます。

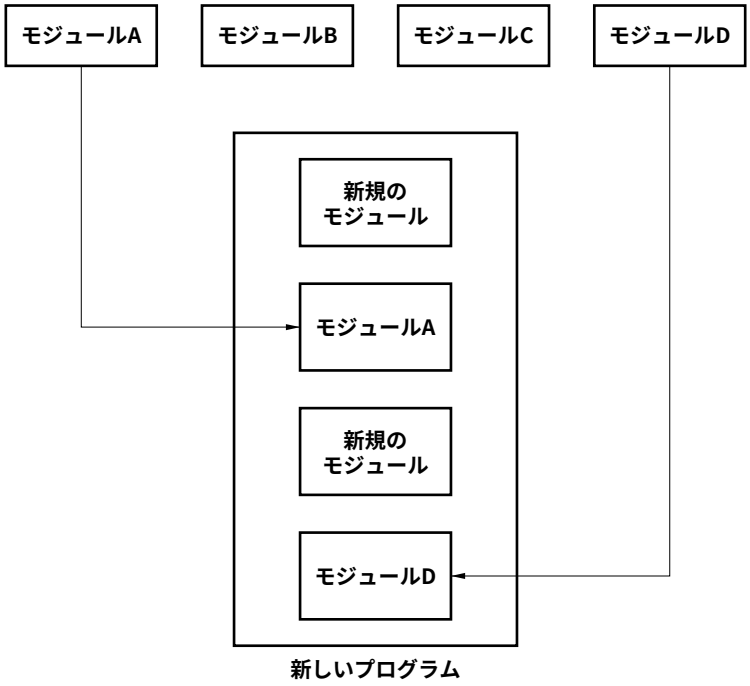
図1－6　アセンブルのやり直し



(2) 資源の活用ができる

以前に作成された信頼性，汎用性の高いモジュールを，他のプログラムの開発に利用することができます。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

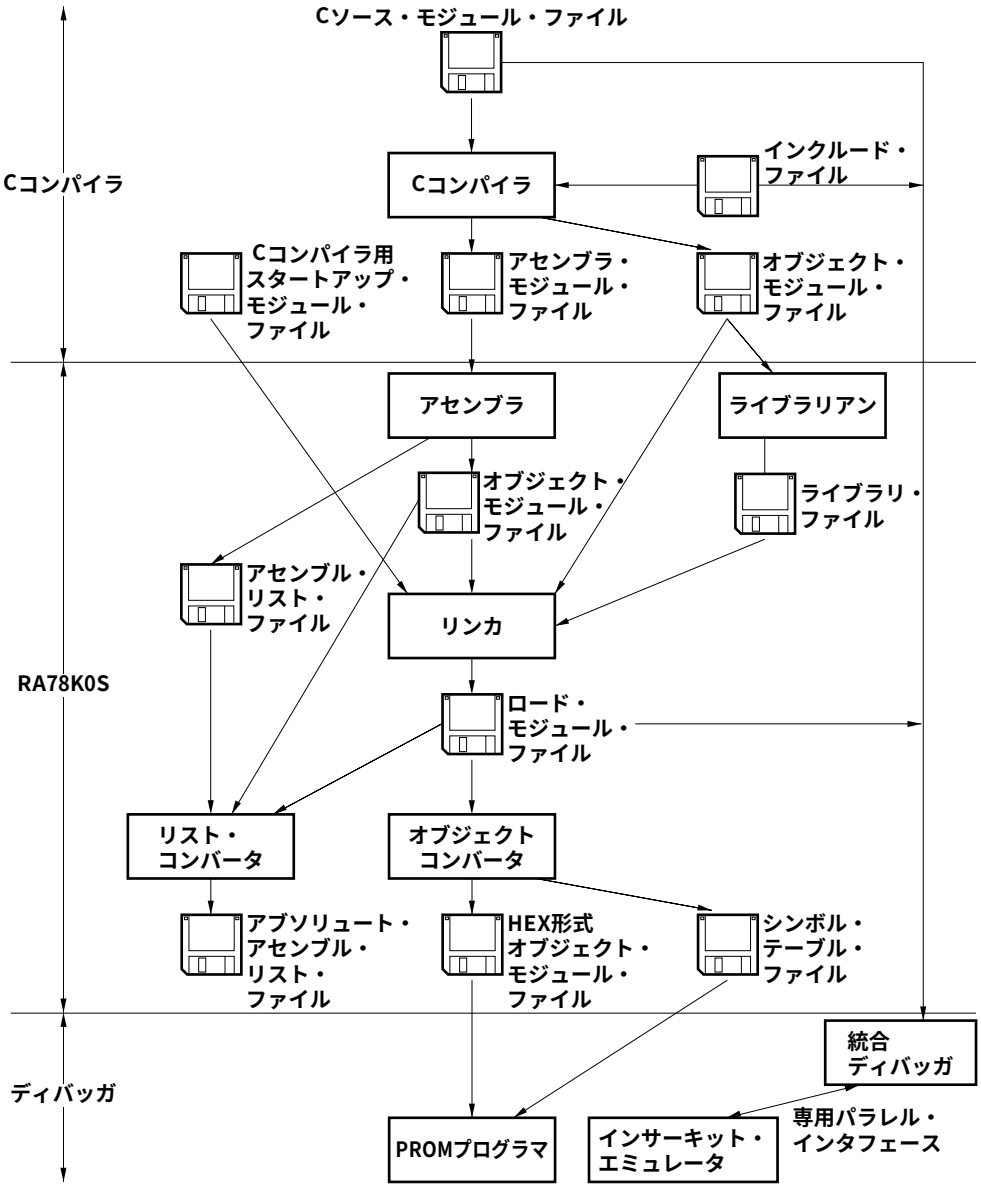


1. 2　RA78K0Sの機能概要

RA78K0Sにおける一般的なプログラム開発手順を，図1－8　RA78K0Sによるプログラム開発手順に示します。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“ RA78K0S ”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　RA78K0Sによるプログラム開発手順



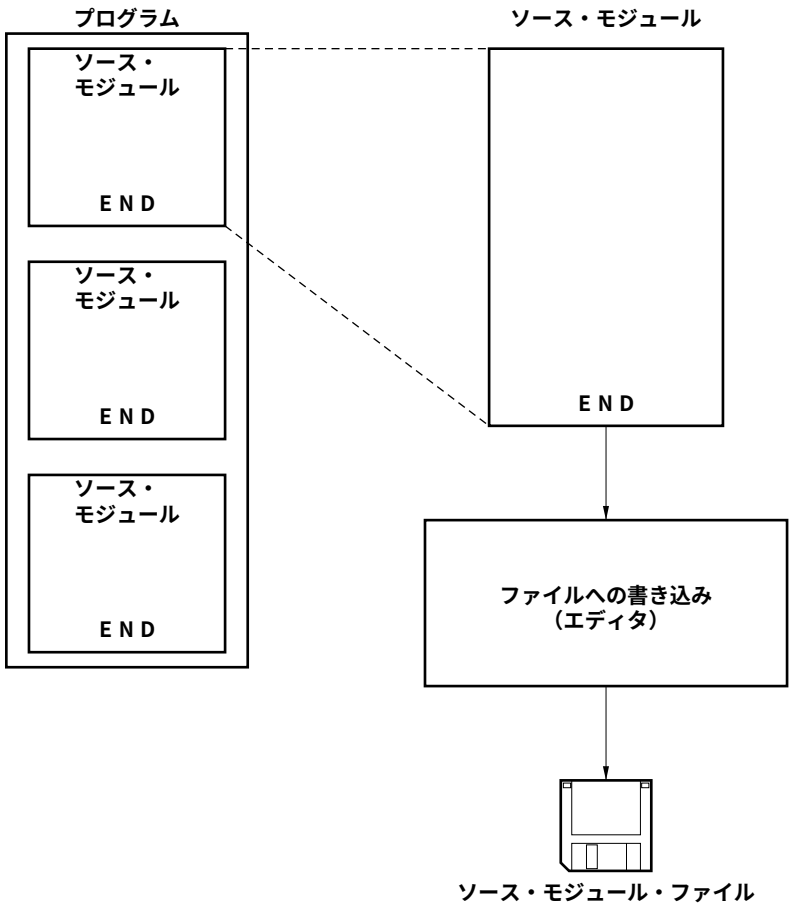
1. 2. 1　エディタによるソース・モジュール・ファイルの作成

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のもので，またアセンブラの入力単位ともなります。

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

ソース・モジュール・ファイルは，アセンブラの入力ファイルとなります。

図1－9　ソース・モジュール・ファイルの作成

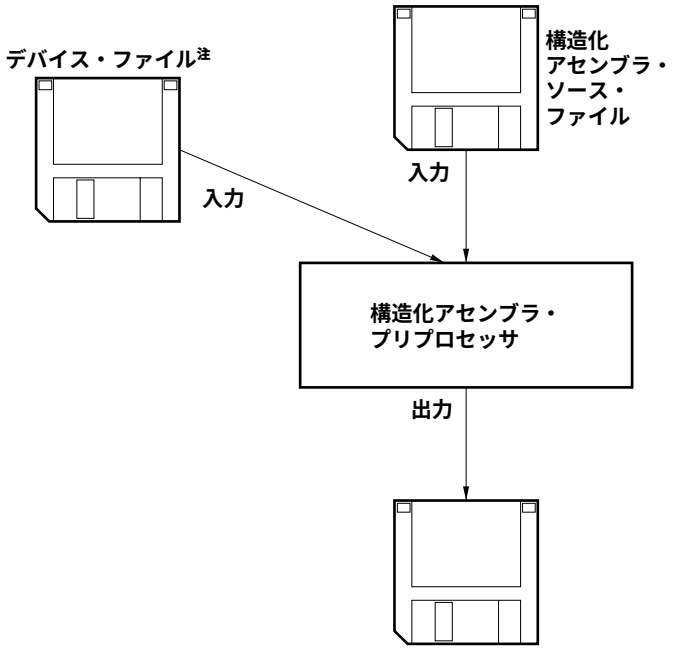


1. 2. 2　構造化アセンブラ・プリプロセッサ

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

構造化アセンブラ・プリプロセッサや構造化アセンブリ言語については，別冊のRA78K0S　**ユーザーズ・マニュアル　構造化アセンブリ言語編**（U11623J）をご覧ください。

図1－10　構造化アセンブラ・プリプロセッサの機能

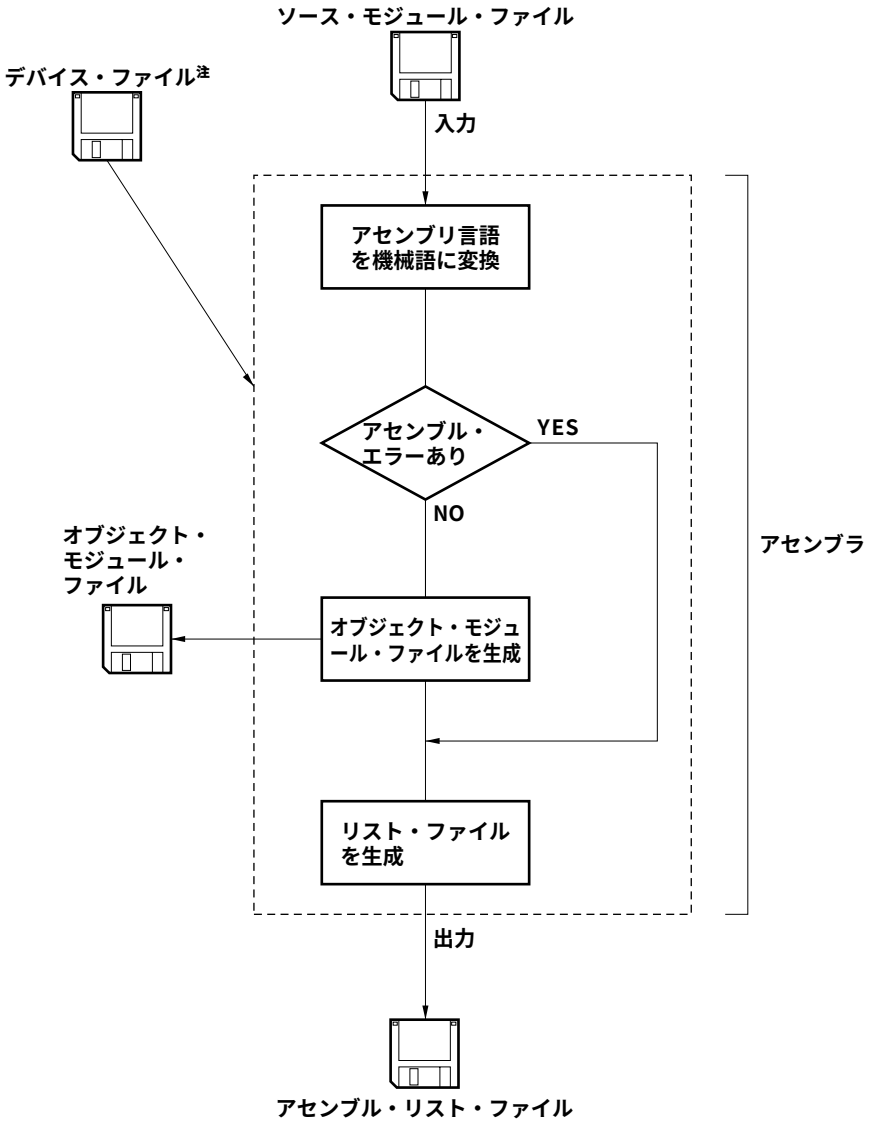


注　デバイス・ファイルは別途入手してください。

1. 2. 3　アセンブラ

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

図1－11　アセンブラの機能



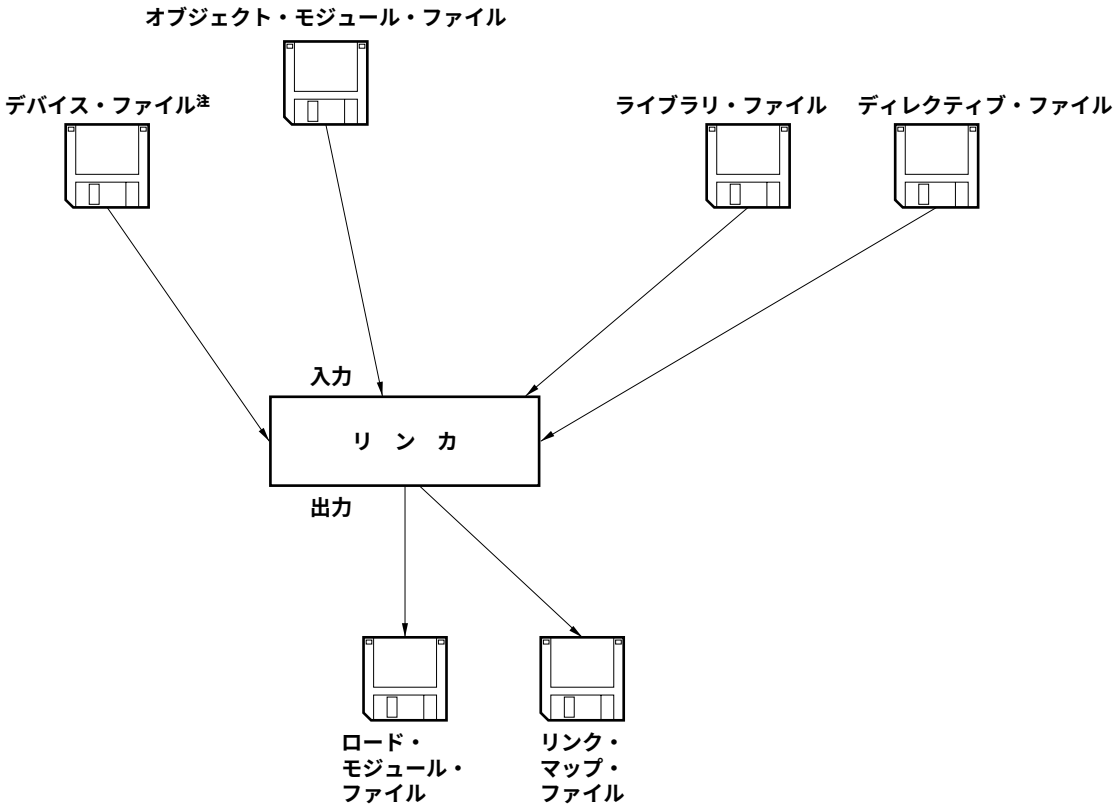
注　デバイス・ファイルは別途入手してください。

1. 2. 4　リ ン カ

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

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

図1－12　リンカの機能



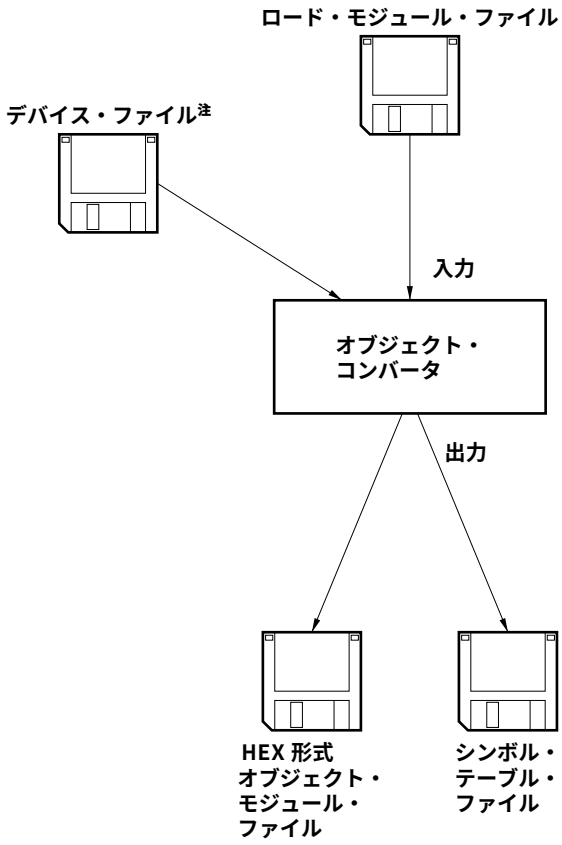
注　デバイス・ファイルは別途入手してください。

1. 2. 5　オブジェクト・コンバータ

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

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

図1－13　オブジェクト・コンバータの機能



注　デバイス・ファイルは別途入手してください。

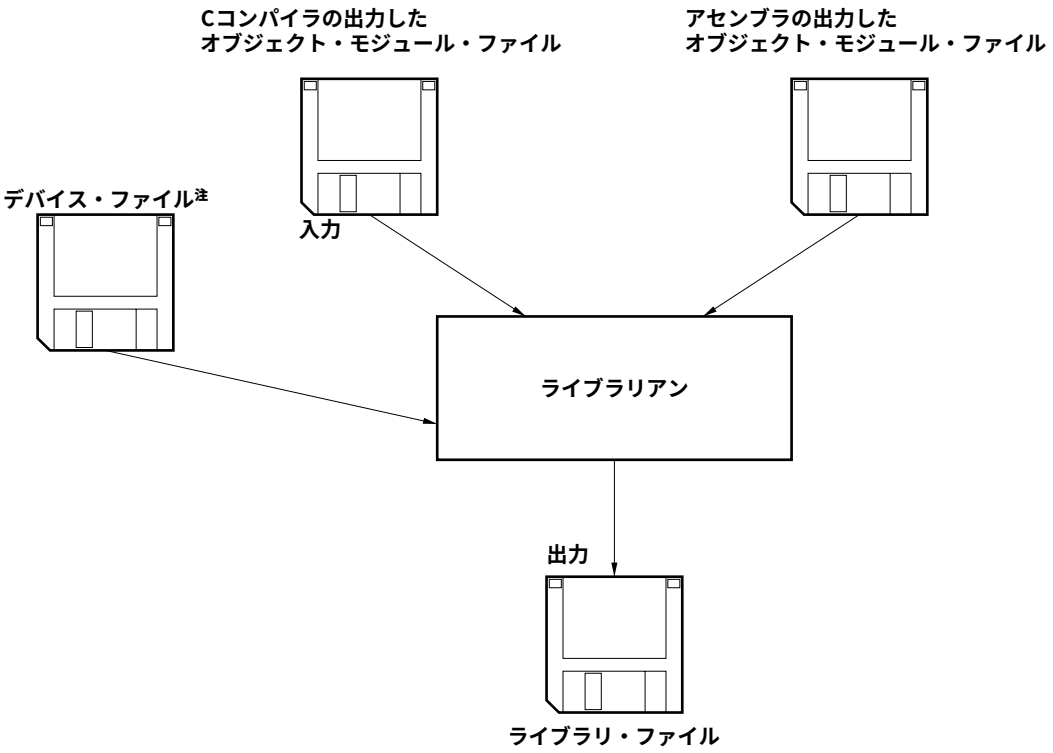
1. 2. 6　ライブラリアン

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

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

ライブラリ・ファイルの作成と更新にはライブラリアンを使用します。

図1－14　ライブラリアンの機能



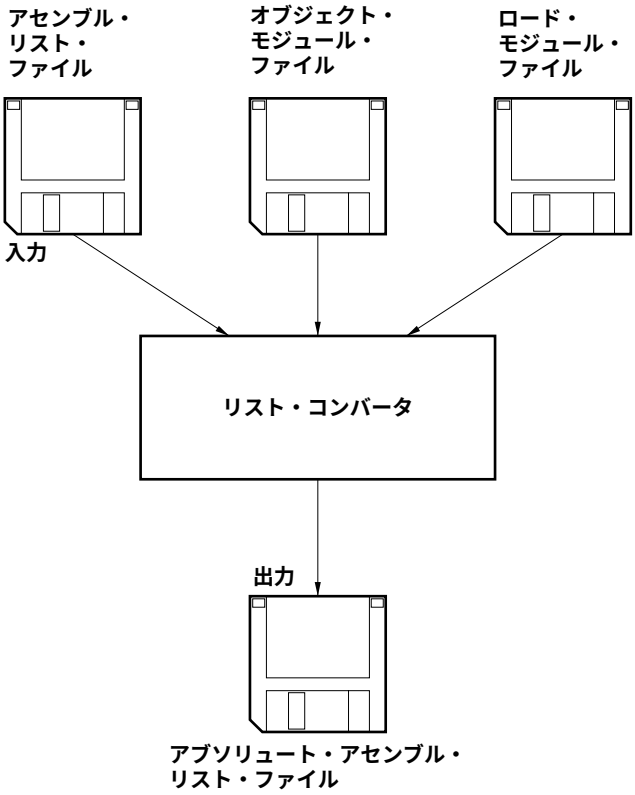
注　デバイス・ファイルは別途入手してください。

1. 2. 7　リスト・コンバータ

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

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

図1－15　リスト・コンバータの機能



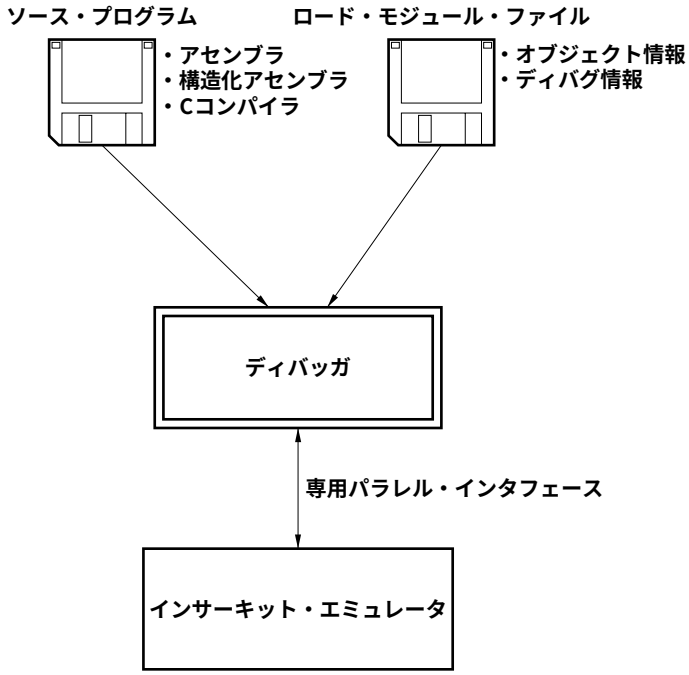
1. 2. 8　ディバッガ

78K0Sシリーズ用ディバッガは，ソース・プログラム，レジスタ，メモリなどの情報をそれぞれのウインドウに表示してディバグを行うソフトウエア・ツールです。

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

ディバッガ, IEは，別パッケージ（別売）になります。

図1－16　ディバッガの機能



1. 3　プログラム開発をはじめる前に

実際にプログラム開発をはじめる前に，次のことを頭に入れておいてください。

1. 3. 1　RA78K0Sの最大性能

(1) アセンブラの最大性能

表1－1　アセンブラの最大性能

項　　目	最大性能	
	PC版	EWS版
シンボル数（ローカル＋パブリック）	65535個[注1]	65535個[注1]
クロスレファレンス・リスト出力可能なシンボル数	65534個[注2]	65534個[注2]
1マクロ参照のマクロ・ボディ最大サイズ	1 Mバイト	1 Mバイト
全マクロ・ボディ合計のサイズ	10 Mバイト	10 Mバイト
1ファイル中のセグメント数	100個	100個
1ファイル中のマクロ，インクルード指定	10000個	10000個
1インクルード・ファイル中のマクロ，インクルード指定	10000個	10000個
リロケーション情報[注3]	65535個	65535個
行番号情報	65535個	65535個
1ファイル中のBR/CALL疑似命令数	32767個	32767個
ソース1行の文字数	2048文字[注4]	2048文字[注4]
シンボル長	256文字	256文字
スイッチ名の定義数	100個	100個
スイッチ名の文字長	31文字	31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	8レベル

注1. パブリック・シンボルが2001個以上の場合はテンポラリ・ファイルをアクセスするため，速度が遅くなります。

2. モジュール名，セクション名の個数を引いた数です。

3. 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。

4. 復帰／改行コードを含みます。1行に2049文字以上記述された場合，エラー・メッセージを出力し，処理を終了します。

(2) リンカの最大性能

表1－2　リンカの最大性能

項　　目	最大性能	
	PC版	EWS版
シンボル数（ローカル＋パブリック）	65535個	65535個
同一セグメントの行番号情報	65535個	65535個
セグメント数	65535個	65535個
入力モジュール	1024個	1024個

1. 4　RA78K0Sの特徴

RA78K0Sは，次の特徴的な機能を備えています。

(1) マクロ機能

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

マクロ機能を利用することにより，コーディングの効率を上げることができます。

(2) 分岐命令の最適化機能

分岐命令自動選択疑似命令（BR疑似命令，CALL疑似命令）を備えています。

メモリ効率のよいプログラムを生成するためには，分岐命令の分岐先範囲に応じたバイトの分岐命令を記述する必要があります。しかし，分岐先範囲をいちいち意識して，分岐命令を記述することは面倒です。BR疑似命令，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・プログラムの一部分を条件によりアセンブルするかしないかを設定できます。

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

第2章　製品概要とインストール方法

この章では，RA78K0Sの提供媒体に格納されているファイル群をユーザの開発環境(ホスト・マシン) 上にインストールする際の手順，および，ユーザの開発環境上からアンインストールする際の手順について説明します。

2. 1　ホスト・マシンと供給媒体

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

表2－1　アセンブラ・パッケージの供給媒体と記録形式

ホスト・マシン	OS	供給媒体	記録形式
PC-9800シリーズ	日本語Windows（98/Me/2000/XP/NT™4.0）注	CD-ROM	Windows標準のインストーラに対応
IBM PC/AT互換機	日本語Windows（98/Me/2000/XP/NT4.0）注 英語Windows（98/Me /2000/XP/NT4.0）注		
HP9000シリーズ700	HP-UX™（Rel. 10.10以降）	CD-ROM	cpコマンド
SPARCstationファミリ	SunOS™（Rel. 4.1.4以降） Solaris™（Rel. 2.5.1以降）		

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

2. 2　インストール

2. 2. 1　Windows版のインストール

以下に，RA78K0S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。

①Windows 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

RA78K0S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RA78K0S￥DISK1**に格納されている**SETUP.EXE**を起動します。**

③ファイル群の確認

Windowsの標準アプリケーションExplorerなどを用いて，RA78K0Sの提供媒体に格納されていたファイル群がホスト・マシン上にインストールされたことを確認します。
なお，各フォルダについての詳細は，2. 4 **ディレクトリ構成**を参照してください。

2. 2. 2　UNIX版のインストール

UNIX版のインストール手順は次のとおりです。なお，ここでは/nectools/binにインストールするものと仮定します。

①ログイン

ホスト・マシンにログインします。

②ディレクトリの移動

インストール・ディレクトリへ移動します。

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

③提供媒体のセット

CD-ROMをCDドライブにセットします。

④各種ファイルのコピー

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

⑤環境変数PATHの設定

環境変数PATHに/nectools/binを追加します。

2. 3　デバイス・ファイルのインストール

2. 3. 1　Windows版のインストール

デバイス・ファイルのインストールには，“ デバイスファイルインストーラ ” を使用します。“ デバイスファイルインストーラ ” は，RA78K0Sとともにインストールされます。

2. 3. 2　UNIX版のインストール

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

2. 3. 3　デバイス・ファイルのレジストリ登録

デバイス･ファイルがすでにインストールされている場合，RA78K0Sのインストール実行中に，デバイス･ファイルのレジストリ登録を促すメッセージが表示されることがあります。

現32ビット環境で使用する場合は，RA78K0S（Ver.1.30以前，16ビット環境）で使用していたデバイス・ファイルをレジストリ（32ビット環境）に登録してください。

なお，レジストリへの登録は，RA78K0Sのインストールが完了したあと，"デバイスファイルインストーラ" を使用することによっても可能です。

レジストリ登録の手順は次のとおりです。

① "デバイスファイルインストーラ"の起動

②ソース選択

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

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

③移行

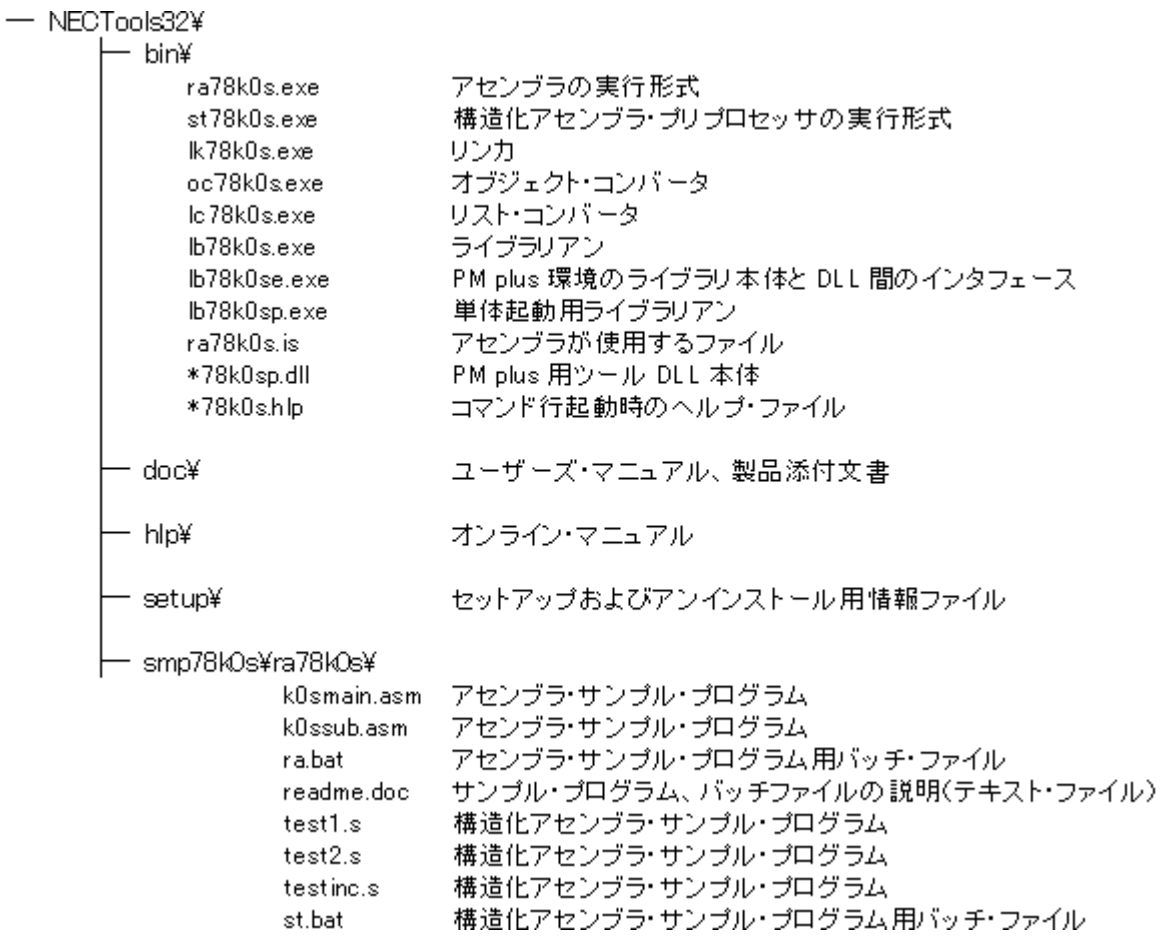
移行(M)ボタンをクリックすることでレジストリ(32ビット環境)に登録されます。

2. 4 ディレクトリ構成

2. 4. 1　Windows版のディレクトリ構成

インストール時に表示される標準ディレクトリは，Windowsシステムのあるドライブの “ ¥NECTools32 ” です。インストール・ディレクトリ下の構成は，次のようになります。ただし，ドライブやインストール・ディレクトリはインストール時に変更可能です。なお，PM plusとRA78K0Sは同じディレクトリにインストールしてください。

図2－1　ディレクトリ構成



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

2. 4. 2　UNIX版のディレクトリ構成

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

```
——nectools/
        bin/
          ra78k0s          アセンブラの実行形式
          st78k0s          構造化アセンブラ・プリプロセッサの実行形式
          lk78k0s          リンカの実行形式
          oc78k0s          オブジェクト・コンバータの実行形式
          lc78k0s          リスト・コンバータの実行形式
          lb78k0s          ライブラリアンの実行形式
          *.hlp            各プログラムに対応したヘルプ・ファイル（テキスト・ファイル）
          ra78k0s.is*      アセンブラが使用する命令セットを定義したテーブル・ファイル
          *.asm,*.s        インストール確認用サンプル・プログラム
          *.sh             インストール確認用バッチ・ファイル
          *.sh             インストール確認用バッチ・ファイル
          readme.doc       インストール確認用シェル・ファイルの使用説明（テキスト・ファイル）
```

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

2.5　アンインストール手順

2.5.1　Windows版のアンインストール

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

① Windows の起動

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

② ［コントロールパネル］ウィンドウのオープン

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

③［アプリケーションの追加と削除のプロパティ］ウィンドウのオープン

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

④ RA78K0S の削除

［アプリケーションの追加と削除のプロパティ］ウィンドウの［セットアップと削除］タブに表示されているインストール済みソフトウエア一覧の中から“NEC RA78K0S 78K/0 アセンブラ Vx.xx”を選択したあと，＜追加と削除(R)... ＞ ボタンをクリックしてください。

なお，［ファイル削除の確認］ウィンドウがオープンした場合は，＜はい(Y) ＞ ボタンをクリックしてください。

⑤ ファイル群の確認

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

2.5.2　UNIX版のアンインストール

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

2. 6　環境設定

2. 6. 1　環境変数

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

PATH	アセンブラの実行形式を置いたディレクトリを指定します
TMP	一時ファイルを作成するディレクトリを指定します （PC-9800シリーズ，IBM PC/AT互換機のみ有効）
INC78K0S	インクルード・ファイルを検索するディレクトリを指定します
LIB78K0S	ライブラリを使用する場合ライブラリを検索するディレクトリを指定します
LANG78K	コメントに記述した漢字の漢字コードを指定します

【指 定 例】

PC-9800シリーズ，IBM PC/AT互換機の場合

```
------------------------------------------------
  PATH = %PATH%;C:¥NECTools32¥bin
  set TMP = C:¥tmp
  set INC78K0S = C:¥NECTools32¥inc78k0s
  set LIB78K0S = C:¥NECTools32¥lib78k0s
  set LANG78K = SJIS
------------------------------------------------
```

HP9000シリーズ700，SPARCstationファミリの場合

・cshを使用している場合の例

```
  set path =($path /ra78k0s)
  setenv INC78K0S /ra78k0s
  setenv LIB78K0S /ra78k0s
  setenv LANG78K EUC
```

・shを使用している場合の例

```
  PATH = $PATH:/ra78k0s
  INC78K0S = /ra78k0s
  LIB78K0S = /ra78k0s
  export PATH INC78K0S LIB78K0S
  setenv LANG78K EUC
```

2. 6. 2　ソース・ファイル中の漢字コード

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

第3章　RA78K0Sの実行手順

アセンブラ・パッケージRA78K0S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‘ k0smain,asm ’，‘ k0ssub.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。また，構造化アセンブルからオブジェクト・コンバートまでの実行手順では，サンプル・プログラム‘ test1.s ’，‘ test2.s ’を使用します。

ここでは，PC-9800シリーズおよびIBM PC/AT互換機版については，PM plus上で実行する方法と，コマンド行で実行する方法を説明します。

3. 1　RA78K0S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「K0SMAIN.ASM」，「K0SSUB.ASM」は，動作確認用サンプル・プログラムのファイルです。

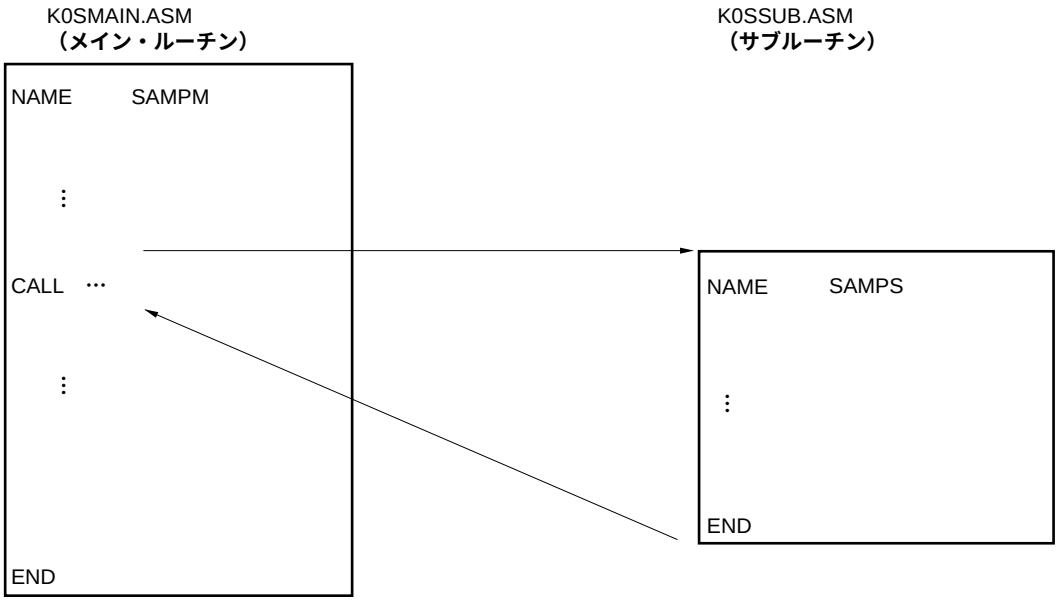
これらのファイルは，これからのアセンブラの操作でソース・プログラム・ファイルとして，アセンブラへの入力になります。

次に，サンプル・プログラムの内容について簡単に説明します。このサンプル・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（K0SMAIN.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（K0SSUB.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



■K0SMAIN.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   AX, #_@STBEG
        MOVW   SP, AX

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

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

        BR     $$

        END
```

■K0SSUB.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:
        MOV    A, [HL]
        ROR    A, 1
        ROR    A, 1
        ROR    A, 1
        ROR    A, 1
        AND    A, #0FH          ;hex upper code load
        CALL   !SASC
        MOV    B, A             ;store result

        XOR    A, A
        XCH    A, [HL]
        AND    A, #0FH          ;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
```

備考　このサンプル・プログラムは，RA78K0Sの機能や操作を習得していただく目的で用意した参考プログラムです。したがって，アプリケーション・プログラムとして，そのまま利用することはできません。

3. 1. 2　サンプルの構成

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。

K0smain.asm	メイン・モジュール
K0ssub.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　RA78K0Sの実行手順

RA78K0S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“k0smain.asm”，“k0ssub.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・ファイルは別途入手してください）。

ここでは，ターゲット・デバイスを“μPD789024”として説明します。

■ra.bat（RA78K0S動作確認用バッチ・プログラム）

```
echo off
cls
set      LEVEL=0

if "%1" == "" goto ERR_BAT

ra78k0s -C%1 k0smain.asm
if errorlevel 1 set LEVEL=1
ra78k0s -C%1 k0ssub.asm
if errorlevel 1 set LEVEL=1
if %LEVEL% == 1 echo Assemble error !!
if %LEVEL% == 1 goto END

cls
lk78k0s k0smain.rel k0ssub.rel -s -orasample.lmf -prasample.map
if errorlevel 1 echo Link error !!
if errorlevel 1 goto END

cls
oc78k0s rasample
if errorlevel 1 echo Object conversion error !!
if errorlevel 1 goto END
```

```
        cls
        set LEVEL=0
        lc78k0s -lrasample.lmf -rk0smain.rel k0smain.prn
        if errorlevel 1 set LEVEL=1
        lc78k0s -lrasample.lmf -rk0ssub.rel k0ssub.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。

RA78K0S**動作確認用バッチ・プログラムを，品種を指定して実行します。**

```
A>ra.bat 9024
```

次のメッセージがディスプレイに出力されます。

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

PASS_PARSE  Start
PASS_OUTOBJ Start

 Target chip : uPD789024
 Device file : Vx.xx

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

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

PASS_PARSE  Start
```

```
PASS_OUTOBJ Start

 Target chip : uPD789024
 Device file : Vx.xx

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

画面をクリアする

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

 Target chip : uPD789024
 Device file : Vx.xx

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

画面をクリアする

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

 Target chip : uPD789024
 Device file : Vx.xx

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

画面をクリアする

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

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

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

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

画面をクリアする

```
No error.
```

(2) ドライブA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。

k0smain.rel	：オブジェクト・モジュール・ファイル
k0smain.prn	：アセンブル・リスト・ファイル
k0ssub.rel	：オブジェクト・モジュール・ファイル
k0ssub.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
k0smain.p	：アブソリュート・アセンブル・リスト・ファイル
k0ssub.p	：アブソリュート・アセンブル・リスト・ファイル

（3）RA78K0Sの実行手順のまとめ

RA78K0Sの実行手順をまとめると次のようになります。

図3ー2　RA78K0Sの実行手順1

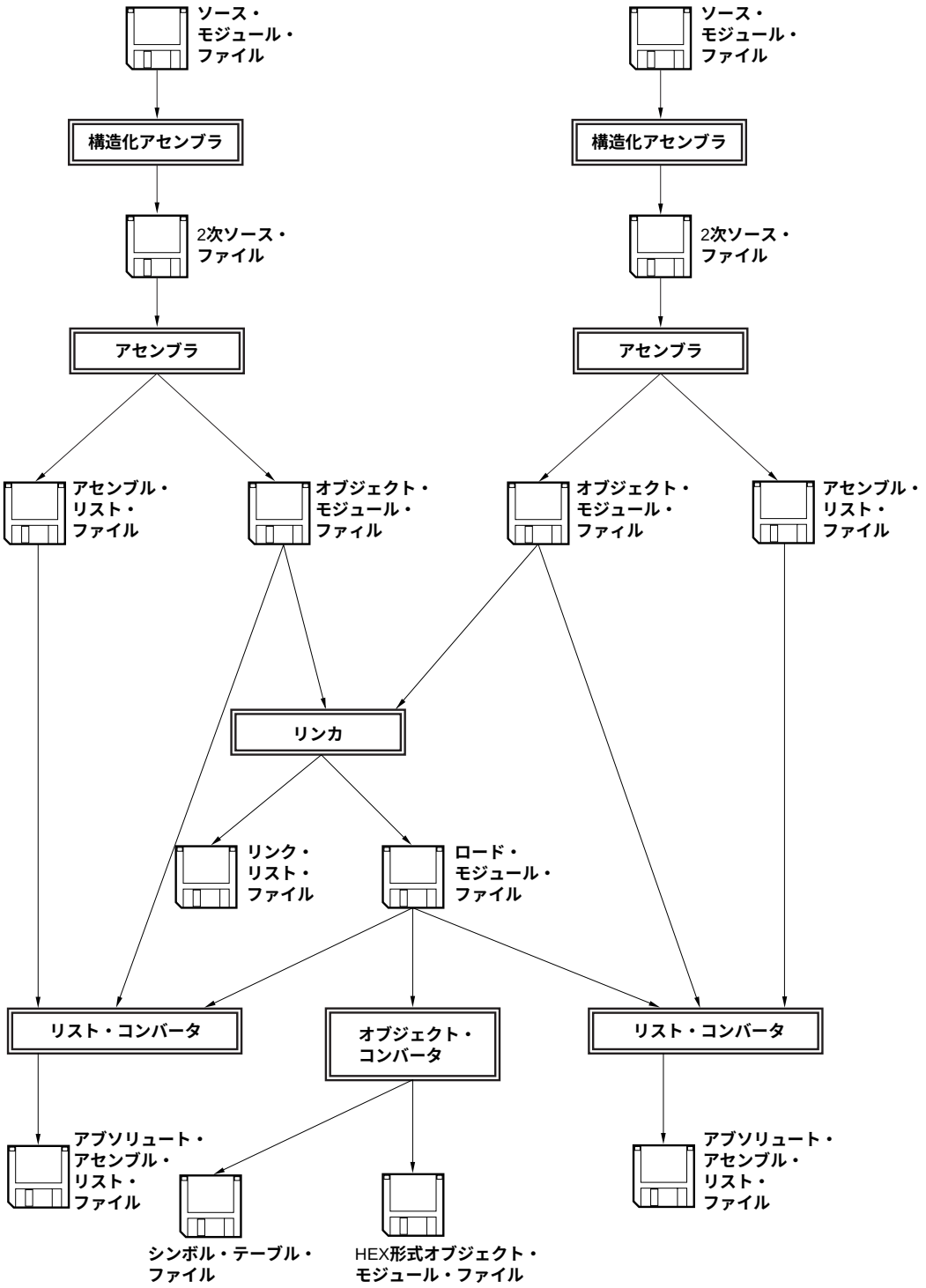
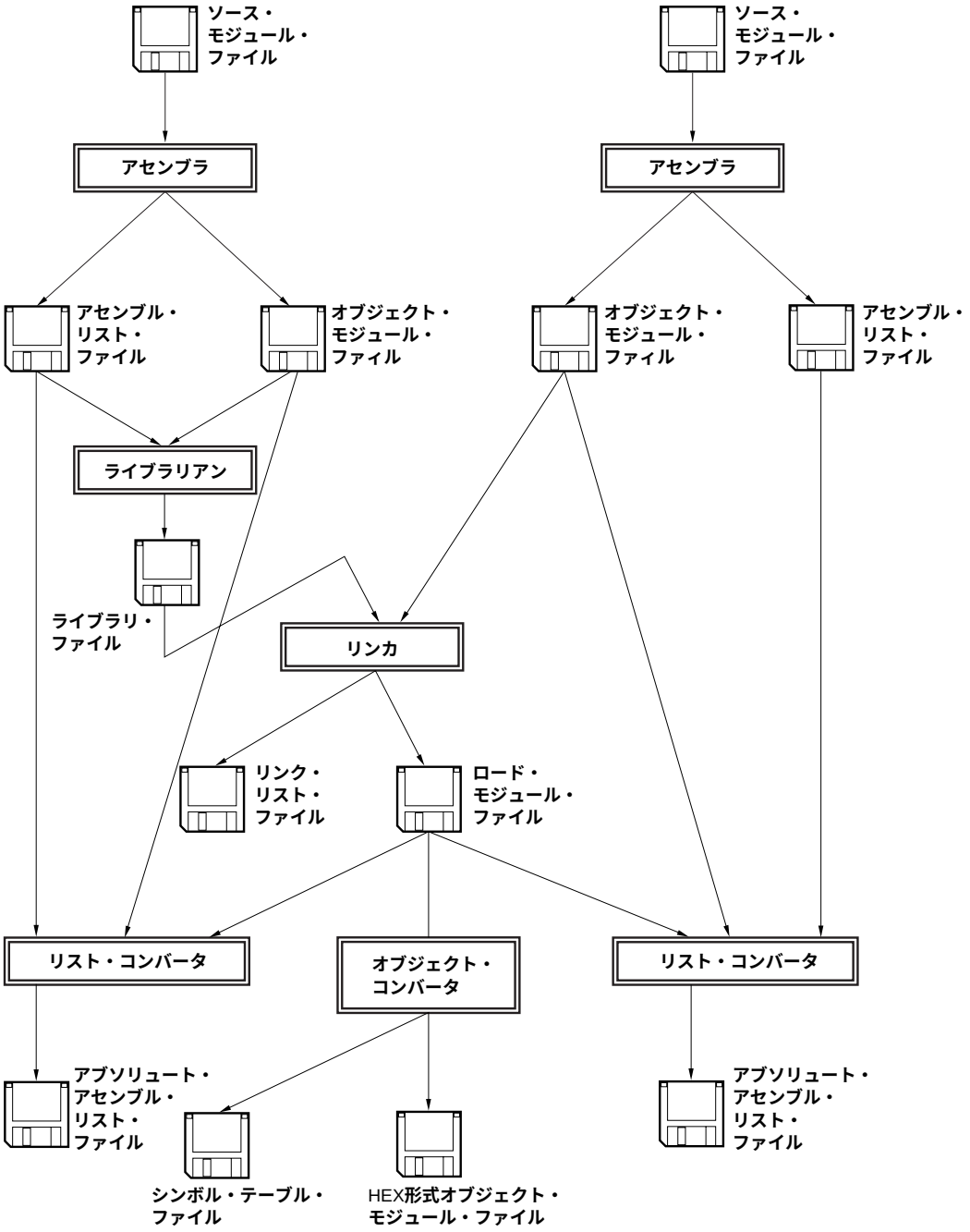


図3ー3　RA78K0の実行手順2



3. 3　ST78K0Sの実行手順

ST78K0Sの動作確認には，システム・ディスクの中に格納されているバッチ・ファイル（st.bat）を使用します。

st.batでは構造化アセンブリ言語で記述された“test1.s”，“test2.s”をソース・ファイルとして，構造化アセンブラ，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・ファイルは別途入手してください。）。

ここでは，ターゲット・デバイスを“μPD789024”として説明します。

■st.bat（ST78K0S動作確認用バッチ・プログラム）

```
echo off
cls
set      LEVEL=0

if "%1" == "" goto ERR_BAT

st78k0s -C%1 test1.s
ra78k0s test1.asm
if errorlevel 1 set LEVEL=1
st78k0s -C%1 test2.s
ra78k0s test2.asm
if errorlevel 1 set LEVEL=1
if %LEVEL% == 1 echo Assemble error !!
if %LEVEL% == 1 goto END

cls
lk78k0s test1.rel test2.rel -s -ostsample.lmf -pstsample.map
if errorlevel 1 echo Link error !!
if errorlevel 1 goto END

cls
oc78k0s stsample
if errorlevel 1 echo Object conversion error !!
if errorlevel 1 goto END

cls
set LEVEL=0
lc78k0s -lstsample.lmf -rtest1.rel test1.prn
if errorlevel 1 set LEVEL=1
```

```
        lc78k0s -lstsample.lmf -rtest2.rel test2.prn
        if errorlevel 1 set LEVEL=1
        if %LEVEL% == 1 echo List conversion error !!
        if %LEVEL% == 1 goto END

        cls
        echo No error.
        goto END

:ERR_BAT

        echo Usage : st.bat chiptype

:END
        echo on
```

(1) バッチ・ファイルを実行します。

ST78K0S動作確認用バッチ・プログラムを，品種を指定して実行します。

```
A>st.bat 9024
```

次のメッセージがディスプレイに出力されます。

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

start

 Target chip : uPD789024
 Device file : Vx.xx

Conversion complete, 0 error(s) found.

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

PASS_PARSE  Start
PASS_OUTOBJ Start

 Target chip : uPD789024
 Device file : Vx.xx

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

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

start

 Target chip : uPD789024
 Device file : Vx.xx

Conversion complete, 0 error(s) found.

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

PASS_PARSE  Start
PASS_OUTOBJ Start

 Target chip : uPD789024
 Device file : Vx.xx

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

画面をクリアする

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

 Target chip : uPD789024
 Device file : Vx.xx

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

画面をクリアする

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

 Target chip : uPD789024
 Device file : Vx.xx

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

画面をクリアする

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

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

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

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

画面をクリアする

```
No error.
```

(2) ドライブAの内容をチェックします。

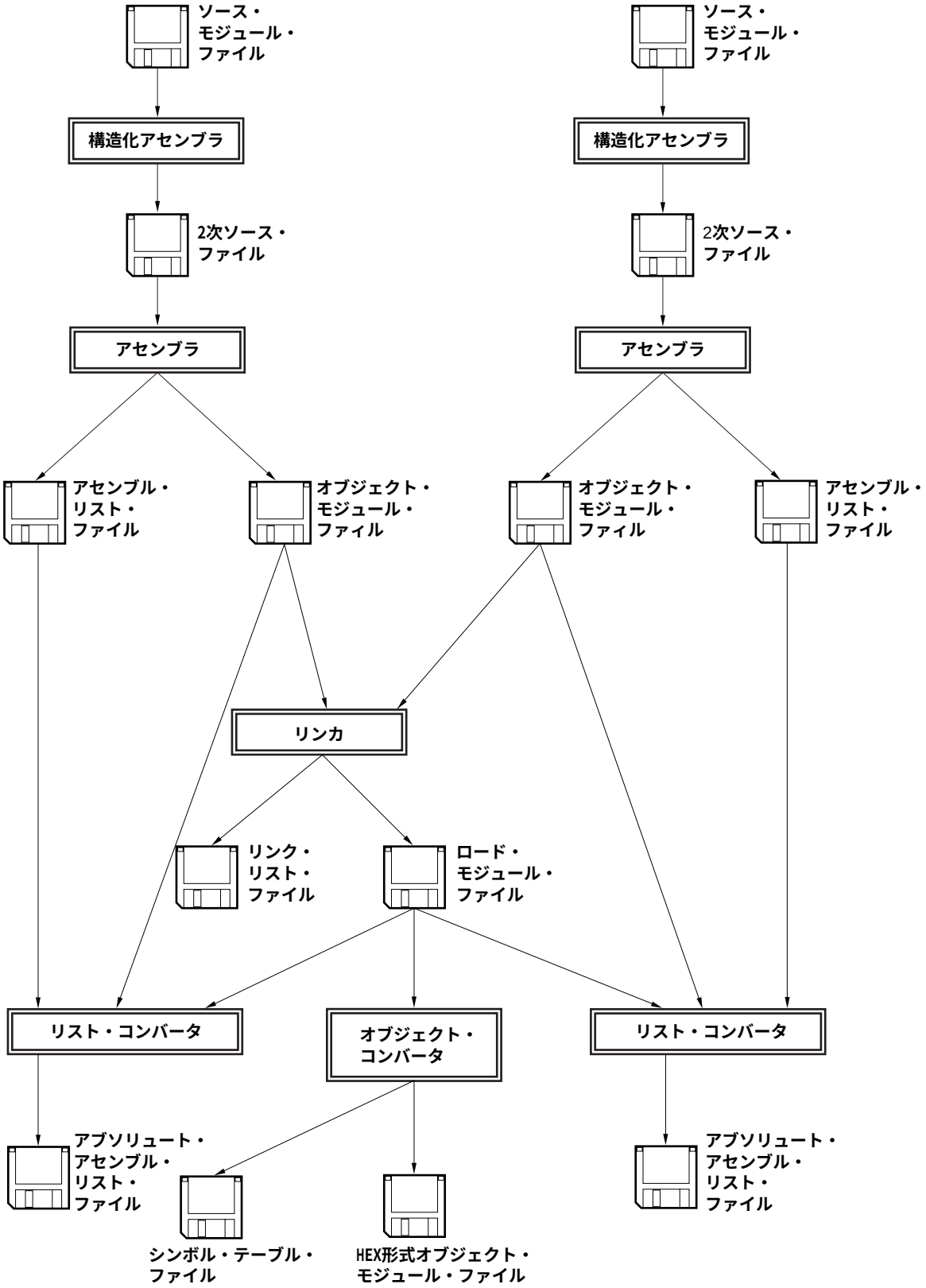
次のファイルが出力されています。

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

（3）ST78K0の実行手順のまとめ

ST78K0の実行手順をまとめると次のようになります。

図3ー4　ST78K0の実行手順



3. 4　コマンド行（DOSプロンプト，EWS）でのアセンブル～リンク，オブジェクト・コンバート

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（K0SMAIN.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>ra78k0s –c9176 k0smain.asm
```

次のメッセージがディスプレイに出力されます。

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

Pass1 Start
Pass2 Start

 Target chip:uPD789176
 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（K0SMAIN.REL）とアセンブル・リスト・ファイル（K0SMAIN.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（K0SSUB.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>ra78k0s –c9176 k0ssub.asm
```

次のメッセージがディスプレイに出力されます。

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

Pass1 Start
Pass2 Start

 Target chip:uPD789176
 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（K0SSUB.REL）とアセンブル・リスト・ファイル（K0SSUB.PRN）を出力します。

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

(5) ディレクティブ・ファイルを作成します。

ディレクティブ・ファイルは，リンカに対してセグメントの配置を指定します。

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

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

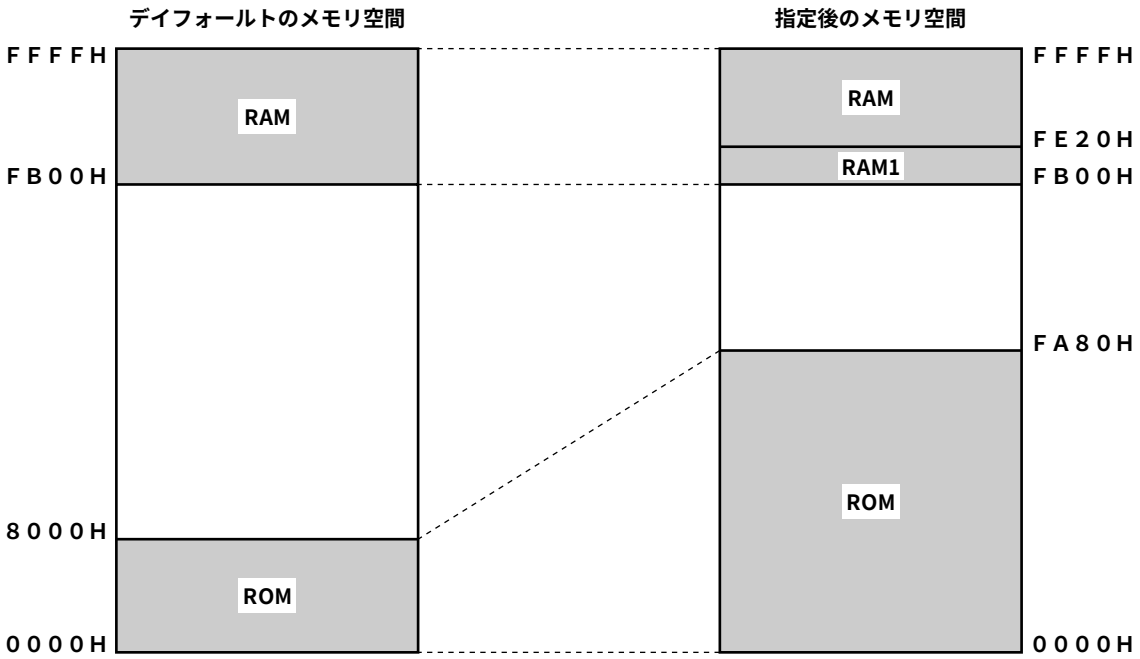
ディレクティブ・ファイルは，リンク時に，-Dオプションを用いてリンカに入力します。

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

ディレクティブ・ファイルには，次のように記述します。

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

図3ー5　リンク・ディレクティブ



(6) アセンブルの結果，出力されたオブジェクト・モジュール・ファイル「K0SMAIN.REL」と「K0SSUB.REL」をリンクします。

また，ディレクティブ・ファイルとして，K0S.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>lk78k0s k0main.rel k0ssub.rel –dk0s.dr[注1] –ok0s.lmf –pk0s.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。

2. スタック解決用シンボル（_@STBEG）生成オプションです。

次のメッセージがディスプレイに出力されます。

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

 Target chip:uPD789176
 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（K0S.lmf）とリンク・リスト・ファイル（K0S.MAP）を出力しました。

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

(8) リンクの結果出力されたロード・モジュール・ファイル（K0S.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>oc78k0s k0s.lmf –r –u0FFH
```

次のメッセージがディスプレイに出力されます。

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

 Target chip:uPD789176
 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（K0S.HEX）とシンボル・テーブル・ファイル（K0S.SYM）を出力しました。

(10) ライブラリ・ファイルを作成します。

アセンブラの出力したオブジェクト・モジュール・ファイル（K0SSUB.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。

```
A>lb78k0s < k0s.job
```

次のメッセージがディスプレイに出力されます。

```
78K/0S Series Librarian Vx. xx [xx xxx xx]
   Copyright (C) NEC Electronics Corportaion xxxx, xxxx
*create k0s.lib
*add k0s.lib k0ssub.rel
*exit
```

備考　上記の下線部は，“ k0s.job ”の内容です。

(11) ドライブA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（K0S.LIB）を出力しました。

(12) アブソリュート・アセンブル・リストを作成します。

K0SMAIN.ASMのアブソリュート・アセンブル・リストを作成するため「K0SMAIN.REL」，「K0SMAIN.ASM」および「K0S. 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。

```
A>lc78k0s k0smain -lk0s.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 RA78K0S Vx. xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation xxxx, xxxx

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

(13) ドライブA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（K0SMAIN.P）を出力しました。

3.5　パラメータ・ファイル使用時

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

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

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

パラメータ・ファイルによる起動方法は，次のようになります。

```
> [パス名] RA78K0S △ -Fパラメータ・ファイル名
> [パス名] LK78K0S △ -Fパラメータ・ファイル名
< [パス名] OC78K0S △ -Fパラメータ・ファイル名
```

次に使用例を示します。

```
例  A>ra78k0s -Fpara.pra
    A>lk78k0s -Fpara.plk
    A>oc78k0s -Fpara.poc
```

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

パラメータ・ファイルをエディタで作成した例を次に示します。

（para1.praの内容）

```
-c9024 k0smain.asm -e
```

（para.plkの内容）

```
k0smain.rel k0ssub.rel -bmylib.lib -osample.lmf -s
```

（para.pocの内容）

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

第4章　構造化アセンブラ

構造化アセンブラは，78K0Sシリーズの構造化アセンブリ言語で記述されたソース・モジュール・ファイルを入力し，それをアセンブリ言語に変換して2次ソース・モジュール・ファイルとして出力します。

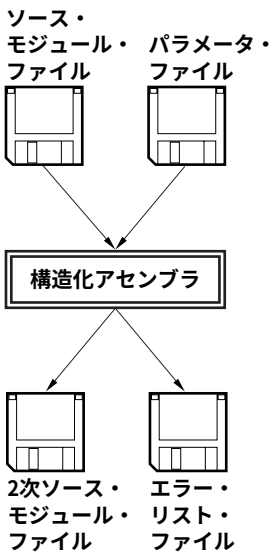
4. 1　構造化アセンブラの入出力ファイル

構造化アセンブラの入出力ファイルを以下に示します。

表4－1　構造化アセンブラの入出力ファイル

種　類	ファイル名	説　　明	ディフォールト・ファイル・タイプ
入力ファイル	ソース・モジュール・ファイル	○構造化アセンブリ言語で記述されたソース・モジュール・ファイルです。 ○ユーザ作成ファイルです。	なし
	パラメータ・ファイル	○構造化アセンブラのオプションをファイルから指定するための，オプションを内容とするファイルです。 ○ユーザ作成ファイルです。	.PST
出力ファイル	2次ソース・モジュール・ファイル	○アセンブリ言語で記述されたソース・モジュール・ファイルです。	.ASM
	エラー・リスト・ファイル	○構造化アセンブラのエラー情報を持つファイルです。	.EST

図4－1　構造化アセンブラの入出力ファイル



4. 2　構造化アセンブラの機能

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

(2) 構造化アセンブラは，ファイルやシステムに関するエラーがある場合は，アボート・エラーを出力し，ソース・モジュール中に記述エラーを発見した場合は，フェイタル・エラーやワーニング・エラーを出力します。

　アボート・エラー，フェイタル・エラーの場合は，通常2次ソース・ファイルは出力されません。ただし，-Jオプションが指定された場合には，フェイタル・エラーがある場合でも2次ソース・ファイルを出力します。

(3) 構造化アセンブラは，起動時に指定されたオプションに従って処理を行います。構造化アセンブラのオプションの詳細については，4. 4　**構造化アセンブラ・オプション**を参照してください。

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

4. 3　構造化アセンブラの起動方法

4. 3. 1　構造化アセンブラの起動

構造化アセンブラの起動には，次の2つの方法があります。

(1) コマンド行での起動

次のコマンドを入力して構造化アセンブラを起動します。

```
X> [パス名] st78k0s [△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・ディレクトリ名

③ 構造化アセンブラのコマンド・ファイル名

④ 構造化アセンブラに対して動作の詳細を指示します。

複数のオプションを指定する場合は，それぞれのオプション間を空白で区切ってください。アセンブラ・オプションの詳細については，4. 4　**構造化アセンブラ・オプション**を参照してください。

⑤ 構造化アセンブルするソース・モジュール・ファイル名

例　A>st78k0s -c9024 test1.s -e

(2) パラメータ・ファイルによる起動

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

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

パラメータ・ファイルによる起動方法は，次のようになります。

```
X> [パス名] st78k0s [△ソース・モジュール・ファイル名] △-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・ディレクトリ名
③ パラメータ・ファイル指定オプション
④ パラメータ・ファイル名

パラメータ・ファイル内での記述規則を次に示します。

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

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

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

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

例　パラメータ・ファイル（test1.pst）をエディタで作成します。

●test1.pstの内容

```
;Parameterfile
test1.s -osample.asm
-esample.est -c9024
```

●パラメータ・ファイル（test1.pst）を使用して構造化アセンブラを起動します。

```
A>st78k0s -ftest1.pst
```

4. 3. 2　実行開始，終了メッセージ

(1) 実行開始メッセージ

構造化アセンブラが起動すると，ディスプレイに実行開始メッセージが表示されます。

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

(2) 処理表示メッセージ

構造化アセンブラの100行処理ごとに ”.” が表示されます。

```
start……
```

(3) 実行終了メッセージ

エラーが検出されなかった場合，次のメッセージをディスプレイに出力して制御をOSに戻します。

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

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

エラーが検出された場合，エラーの数をディスプレイに出力して制御をOSに戻します。

```
TEST1.S (8) :F209 Syntax error

  Target chip:uPD78xxxx
  Device file:Vx.xx

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

構造化アセンブル中に構造化アセンブラ処理継続が不可能な致命的エラーが検出された場合，構造化アセンブラはメッセージをディスプレイに出力して構造化アセンブルを中止し，制御をOSに戻します。

例1.　存在しないソース・モジュール・ファイルを指定した場合

```
A>st78k0s sample.s

Structured assembler preprocessor for RA78K/0S Vx.xx [xx xxx xx]
   Copyright (C) NEC Electronics Corporation 1996

A006 File not found 'SAMPLE.S'

Program aborted.
```

例1では，存在しないソース・モジュール・ファイルを指定したためにエラーとなり，アセンブルを中止しました。

例2.　**存在しないオプションを指定した場合**

```
A>st78k0s test1.s -z

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

A012 Missing parameter '-z'
Please enter 'ST78K0S --', if you want help messages.

Program aborted.
```

例2**では，存在しないオプションを指定したためにエラーとなり，アセンブルを中止しました。**

構造化アセンブラがエラー・メッセージを出力してアセンブルを中止した場合，そのエラー・メッセージの原因を第12**章　エラー・メッセージで調べて対処してください。**

4. 4　構造化アセンブラ・オプション

4. 4. 1　構造化アセンブラ・オプションの種類

構造化アセンブラのオプションは，構造化アセンブラの動作に細かい指示を与えるものです。

構造化アセンブラのオプションは，次の13種類のオプションに分類できます。

表4－2　構造化アセンブラ・オプション

項番	分　類	オプション	説　明
1	デバイス種別指定	-C	対象とするデバイス種別を指定します。
2	ワード・シンボル文字指定	-SC	ワード・シンボル名の最後の文字を指定します。
3	シンボル定義	-D	#IFDEF疑似命令などに与えるシンボルを指定します。
4	タブ数指定	-WT	変換した命令を出力する位置を指定します。
5	インクルード・ファイル・パス指定	-I	インクルード・ファイルのドライブ，ディレクトリを指定します。
6	2次ソース・ファイル指定	-O	2次ソース・ファイル名を指定します。
7	エラー・リスト・ファイル指定	-E	エラー・リスト・ファイル名を指定します。
8	パラメータ・ファイル指定	-F	パラメータ・ファイル名を指定します。
9	ディバグ情報出力指定	-GS	構造化アセンブラ・ソース・レベルのディバグ情報の出力を指定します。
		-NGS	
10	2次ソース・ファイル強制出力指定	-J	2次ソース・ファイルの強制的出力を指定します。
11	漢字コード指定	-ZS	コメント文に記述される漢字コードの種別を指定します。
		-ZE	
		-ZN	
12	デバイス・ファイル・サーチ・パス指定	-Y	デバイス・ファイルをサーチするパスを指定します。
13	ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

4.4.2　構造化アセンブラ・オプションの説明

各構造化アセンブラ・オプションの詳細について説明します。

(1) デバイス種別指定　(-C)

記述形式　：-Cデバイス種別
省略時解釈：省略不可

【機　　能】

構造化アセンブラの対象となる対象デバイスを指定します。

【説　　明】

① -Cオプションは必ず指定してください。構造化アセンブラは指定された対象デバイスに対してプリプロセスを行い，アセンブラのソース・コードを生成します。
　なお，-Cオプションを省略した場合はエラーとなります。

② -Cオプションとプロセッサ品種指定制御命令で異なる品種が指定された場合は，ワーニングとなります。この際，構造化アセンブラは，オプションで指定した品種を優先します。

③ -Cオプションで指定した品種をプロセッサ品種指定制御命令として，2次ソース・ファイルに出力します。ただし，プロセッサ品種指定制御命令と同名の品種を指定した場合は，出力しません。

【使 用 例】

μPD789024をターゲットとして指定します。

```
A>st78k0s test.s -c9024
```

【注　　意】

-Cオプションは省略不可能です。ただし，ソース・ファイルの先頭でプロセッサ品種指定制御命令($PROCESSOR)を記述することで，コマンド行での指定を省略できます。

デバイス種別については，各デバイスの**デバイス・ファイル　使用上の留意点**を参照してください。

(2) ワード・シンボル文字指定　(-SC)

記述形式　：-SC文字
省略時解釈：PまたはP

【機　　能】

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

【説　　明】

① 構造化アセンブラは，扱うデータがバイトかワードかによって異なる命令を生成します。
代入であれば，バイトならばMOV命令，ワードならばMOVW命令を生成します。
ワード・シンボルの予約語であればワードの命令を生成します。

② 予約語でないシンボルが指定された場合は，そのシンボルの最後の文字によって，ワード・シンボルかバイト・シンボルかを判定して命令を生成します。

③ -SCオプションを指定しないと，‘Ｐ’か‘ｐ’で終わるシンボルがワード・シンボルと判断されます。

④ 判定文字は英字相当文字のみです。なお，英字の場合は大文字，小文字の区別はありません。

⑤ 重複指定した場合は，最後に指定したものを有効とします。

【使 用 例】

‘＠’で終わるシンボルがワード・シンボルと指定します。

```
A>st78k0s test.s -sc@
```

〈test.s〉

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

〈test.asm〉

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

(3) シンボル定義指定　(-D)

記述形式：-Dシンボル名［=数値］［，シンボル名［=数値］…］

【機　　能】

シンボルの定義を行います。

【説　　明】

① シンボルに与える数値は2進数，8進数，10進数，16進数とします。
　数値指定が省略された場合，値は1となります。
② 本オプションは，シンボルを#define疑似命令で定義したことと同一です。
③ コマンド行ではカンマで区切って30個まで定義できます。
④ 通常は，#ifdef疑似命令と組み合わせて使用します。
⑤ 重複して指定した場合，最後に指定したものを有効とします。
⑥ #define疑似命令と重複した場合，ワーニング・メッセージを出力し，#define疑似命令の定義を有効とします。
⑦ 英字の場合には，英大文字，英小文字どちらを指定しても良く，同一の意味とします。

【使 用 例】

“TRUE”というシンボルに1を定義します。

```
A>st78k0s test.s -dTRUE=1
```

(4) タブ数指定　(-WT)

記述形式　：-WT数値1
　　：-WT［数値1］，数値2
　　：-WT［数値1］，［数値2］，数値3
省略時解釈：-WT2, 3, 4

【機　　能】

交換されたアセンブリ言語を出力するまでのタブの数を指定します。

【説　　明】

① -WTオプションにより，アセンブラのソースの命令出力位置を自由に変更可能となり，プログラムの可読性が向上します。

② 数値1は，命令を出力するまでのタブの数を指定します。
　数値2は，命令のオペランドを出力するまでのタブの数を指定します。
　数値3は，命令のコメントを出力するまでのタブの数を指定します。

③ 数値は10進数で次の範囲内に指定してください。
　数値1：0～97
　数値2：1～98
　数値3：2～99
　数値1　＜　数値2　＜　数値3

④ 重複して指定した場合，最後に指定したものを有効とします。

【使 用 例】

数値1に3，数値2に4，数値3に5を指定します。

```
A>st78k0s test.s -wt3,4,5
```

(5) インクルード・ファイル・パス指定　(-I)

記述形式　：-I［ドライブ番号：］ディレクトリ
省略時解釈：カレント・ディレクトリ

【機　　能】

構造化アセンブラの入力となるインクルード・ファイルのパス名を指定します。

【説　　明】

① インクルード・ファイルが存在するドライブ番号，ディレクトリを指定します。
② -Iオプションを省略すると，インクルード・ファイルはカレントのドライブ番号，カレントのディレクトリにあると判断されます。
③ 重複して指定した場合は後者優先とします。

【使 用 例】

インクルード・ファイルのあるディレクトリをb：¥includeと指定します。

```
A>st78k0s test.s –ib:¥include
```

（6）2次ソース・ファイル指定　（-O）

記述形式　：-O［［［ドライブ番号：］ディレクトリ］ファイル名］
省略時解釈：-O入力ファイル名.ASM

【機　　能】

変換後の2次ソース・ファイルの出力先と，ファイル名を指定します。

【説　　明】

① 変換後の2次ソース・ファイルの出力ドライブ番号，ディレクトリ，ファイル名を指定します。
② -Oオプションを省略した場合には，出力ファイルは入力ファイルのファイル・タイプを“ASM”に置き換えたものがカレント・ディレクトリに作成されます。
③ ファイル名として“NUL”，“AUX”が指定できます。
④ フェイタル・エラー終了時には，2次ソース・ファイルを出力しません。
⑤ 重複して指定した場合は後者優先とします。

【使 用 例】

2次ソース・ファイルを“sample.asm”と指定します。

```
A>st78k0s test.s -osample.asm
```

(7) エラー・リスト・ファイル指定　(-E)

記述形式　：-E［［ドライブ番号：］［ディレクトリ］ファイル名］
省略時解釈：-E入力ファイル名.EST

【機　　能】

エラー・リスト・ファイルの出力先と，ファイル名を指定します。

【説　　明】

① エラー・リスト・ファイルを出力するドライブ番号，ディレクトリ，ファイル名を指定します。
② -Eオプションを省略した場合には，エラー・リスト・ファイルは入力ファイルのファイル・タイプを“.EST”に置き換えたものがカレント・ディレクトリに作成されます。
③ ファイル名として，“NUL”，“AUX”が指定できます。
④ 重複して指定した場合は後者優先とします。

【使 用 例】

エラー・リスト・ファイルを“sample.est”と指定します。

```
A>st78k0s test.s -esample.est
```

(8) パラメータ・ファイル指定　(-F)

記述形式：-F［［ドライブ番号：］ディレクトリ］ファイル名

【機　　能】

パラメータ・ファイルのファイル名を指定します。

【説　　明】

① パラメータ・ファイルの入力ドライブ番号，ディレクトリ，ファイル名を指定します。

② ファイル名は省略できません。ファイル・タイプを省略すると，“ .PST ”と解釈されます。

③ -Dオプションで，シンボルをコマンド行で数多く定義する際に有効です。

④ 重複して指定された場合はエラーとなります。

⑤ パラメータ・ファイルのネストは禁止します。指定された場合は，エラーとなります。

⑥ パラメータ・ファイル中で，“ ； ”，“ # ”以降の文字は，LHまたはEOFまですべてコメントとして解釈します。

【使 用 例】

パラメータ・ファイル“ sample.pst ”を指定します。

```
A>st78k0s -fsample.pst
```

(9) ディバグ情報出力指定　(-GS/-NGS)

記述形式　：-GS
　　：-NGS
省略時解釈：-GS

【機　　能】

構造化アセンブラ・ソース・レベルのディバグ情報の出力を指定します。

【説　　明】

① -GSオプションは，2次ソース・ファイルにディバグ情報の出力を指定します。
② -NGSオプションは，-GSオプションの設定を無効にします。
③ -GSオプションは，入力ソース・ファイル中にコンパイラのディバグ情報があった場合，先頭の‘ $ ’を‘ ; ’に置き換えます。
④ -GS/-NGSオプションを重複指定した場合は，最後に指定したものを有効とします。
⑤ 省略した場合は，-GSオプションを指定したものとみなされます。

【注　　意】

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

【使 用 例】

2次ソース・ファイルにディバグ情報の出力を指定します。

```
A>st78k0s test.s -gs
```

（10）2次ソース・ファイル強制出力指定　（-J）

記述形式：-J

【機　　能】

フェイタル・エラー終了時に，2次ソース・ファイルを強制的に出力させます。

【説　　明】

① フェイタル・エラー終了時に，2次ソース・ファイルを強制的に出力させます。

② フェイタル・エラー行は，2次ソース・ファイルに入力ソース・ファイルのイメージをそのまま出力します。

【使 用 例】

2次ソース・ファイルの強制出力を指定します。

```
A>st78k0s test.s -j
```

（11）漢字コード指定　（-ZS/-ZE/-ZN）

記述形式　：-ZS
：-ZE
：-ZN

省略時解釈：OSにより次のように解釈します。
Windows, HP-UX　：-ZS
SunOS, Solaris　：-ZE

【機　　能】

コメント文に記述される漢字コードの種別を指定します。

【説　　明】

① 漢字コードを次のように指定します。
-ZS：シフトJISコードとして解釈します。
-ZE：EUCコードとして解釈します。
-ZN：漢字として解釈しません。

② 本オプションは，次のように漢字コード指定制御命令に対応しています。
-ZS：$KANJICODE SJIS
-ZE：$KANJICODE EUC
-ZN：$KANJICODE NONE

③ 漢字コードの指定の優先順位は次のようになります。
（1）-ZS/-ZE/-ZNオプションの指定
（2）漢字コード指定制御命令（$KANJICODE）の指定
（3）環境変数LANG78Kの指定
（4）各OSのディフォールトの指定

【使 用 例】

漢字をシフトJISコードとして解釈するように指定します。

```
A>st78k0s test.s -zs
```

(12) デバイス・ファイル・サーチ・パス指定　(-Y)

記述形式　：-Y［ドライブ番号］ディレクトリ

省略時解釈：デバイス・ファイルを次の順序で検索します。

① 〈..¥dev〉（st78k0s.exeを起動したパスに対して）

② st78k0s.exeを起動したパス

③ カレント・パス

④ 環境変数PATHで指定されたパス

【機　　能】

デバイス・ファイルをサーチするパスを指定します。

【説　　明】

① デバイス・ファイルを指定されたパスから読み込みます。

② パス名以外が指定された場合にはエラーとなります。

③ ディレクトリの最後尾にディレクトリ指定記号[注]が記述されていなくても，記述されているものと解釈します。

④ デバイス・ファイルの検索は次の順序で行います。

1．-Yオプションで指定されたパス

2．〈..¥dev〉（st78k0s.exeを起動したパスに対して）

3．st78k0s.exeを起動したパス

4．カレント・パス

5．環境変数PATHで指定されたパス

注　PC-9800シリーズでは‘￥’，IBM PC/AT互換機では‘＼’です。

【使 用 例】

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

```
A>st78k0s test.s –ya:¥nectools¥dev
```

（13）ヘルプ指定　（--）

記述形式　：--
省略時解釈：表示しない

【機　　能】

--オプションは，ヘルプ・メッセージをディスプレイに表示します。

【用　　途】

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

【説　　明】

--オプションを指定すると，他の構造化アセンブラ・オプションはすべて無効となります。

注意　このオプションは，PM plus**上では指定できません。**
PM plus**上でヘルプを参照する場合は，「構造化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。

```
A>st78k0s --

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

Usage:st78k0s [option[...]] input-file [option[...]]

The option is as follows ([]means omissible, ...means repetition) .
  -cx             :Select target chip. (x = 9002, 9014, etc.) *Must be specified.
  -o[file]:Create the assembler source file[with the specified name].
  -e[file]:Create the error list file[with the specified name].
  -ffile  :Input options or source file name from specified file.
  -idirectory     :Set include search path.
  -sc[character]:Specify the last character of word symbol.
  -wtn1/-wt[n1], n2/-wt[n1], [n2], n3
          :Specify the number of tabs up to output position of each field.
            n1:Output position mnemonic field.
            n2:Output position operand field.  *Must be
            n3:Output position comment field.  0 <= n1 < n2 < n3 < 100.
  -dname[=data][,name[=data][...]]
          :Define name[with data].
  -gs/-ngs:Output the structured assembler source debug information to
            assembler source file / Not.
  -j              :Create the assembler source file if fatal error occurred.
  -zs/-ze/-zn     :Change source regulation.
            -zs:SJIS code usable in comment.
            -ze:EUC code usable in comment.

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

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

4. 5　PM plusからのオプション設定

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

4. 5. 1　オプションの設定方法

PM plusの [ツール(T)] メニューの [構造化アセンブラオプションの設定(S)...] を選択または  をクリックすると，<構造化アセンブラオプションの設定>ダイアログが現れます。

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

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

図4－3　<アセンブラオプション>ダイアログ

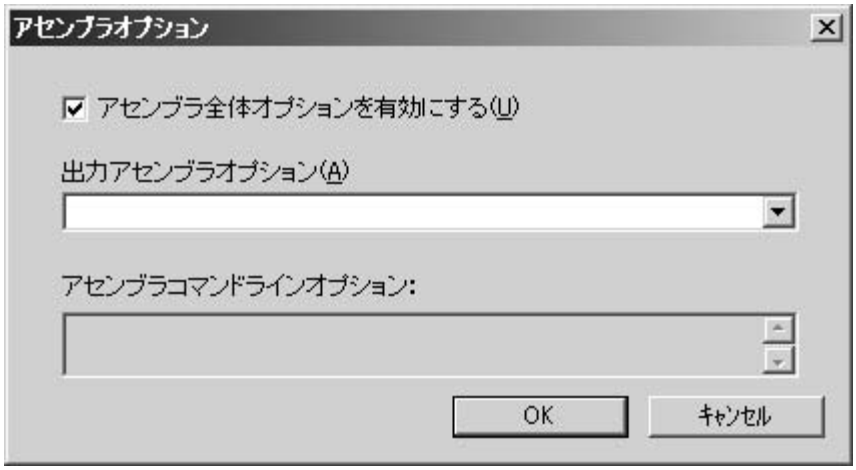


図4－4　<構造化アセンブラオプションの設定>ダイアログ（《その他》タブ選択時）

4.5.2　各オプションの設定

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

・ASMソース[-o](O)

出力パス名：

参照(B)ボタンまたは直接入力により，ASMソースの出力パスを指定します。

・タブ数[-wt](T)

変換されたアセンブリ言語を出力するまでのタブの数を指定します。

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

・ディバグ情報の出力[-gs](G)

2次ソース･ファイルにディバグ情報を出力します。

・コメント中の漢字コード(Z)

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

・エラー･リスト･ファイルの出力[-e](E)

出力パス名：

エラー･リスト･ファイルを出力したい場合は，入力ボックスにファイル名を入力します。

パスを指定する場合は参照(R)ボタンを使用します。

・アセンブラオプション(S)

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

　・アセンブラ全体オプションを有効にする(U)

　　<アセンブラオプションの設定>ダイアログで設定されている全体オプションを有効にします。

　・出力アセンブラオプション

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

・インクルード･ファイル･パス[-i](I)

参照(B)ボタンまたは直接入力によりインクルード･ファイルのパスを指定できます。

・ワードシンボル定義[-sc](S)

ワードとして定義したいシンボルの最終文字を指定します。以後，そのシンボルの最終文字によって，ワード･シンボルかバイト･シンボルかを判定して命令を生成します。

・シンボル定義[-d](D)

シンボルに定義づける数値を入力してください。

・コマンドファイルの作成(U)

コマンドファイルを作成する場合チェックしてください。

・パラメータ･ファイル(P)

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

・その他のオプション(O)

チェック･ボックスやラジオ･ボタンで選択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。

・リセット(R)

入力した内容をリセットします。

・オプション情報読込(E)

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

・オプション情報保存(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。

第5章　アセンブラ

アセンブラは，78K0Sのアセンブリ言語で記述されたソース・モジュール・ファイルを入力し，それを機械語に変換してオブジェクト・モジュール・ファイルとして出力します。

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

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

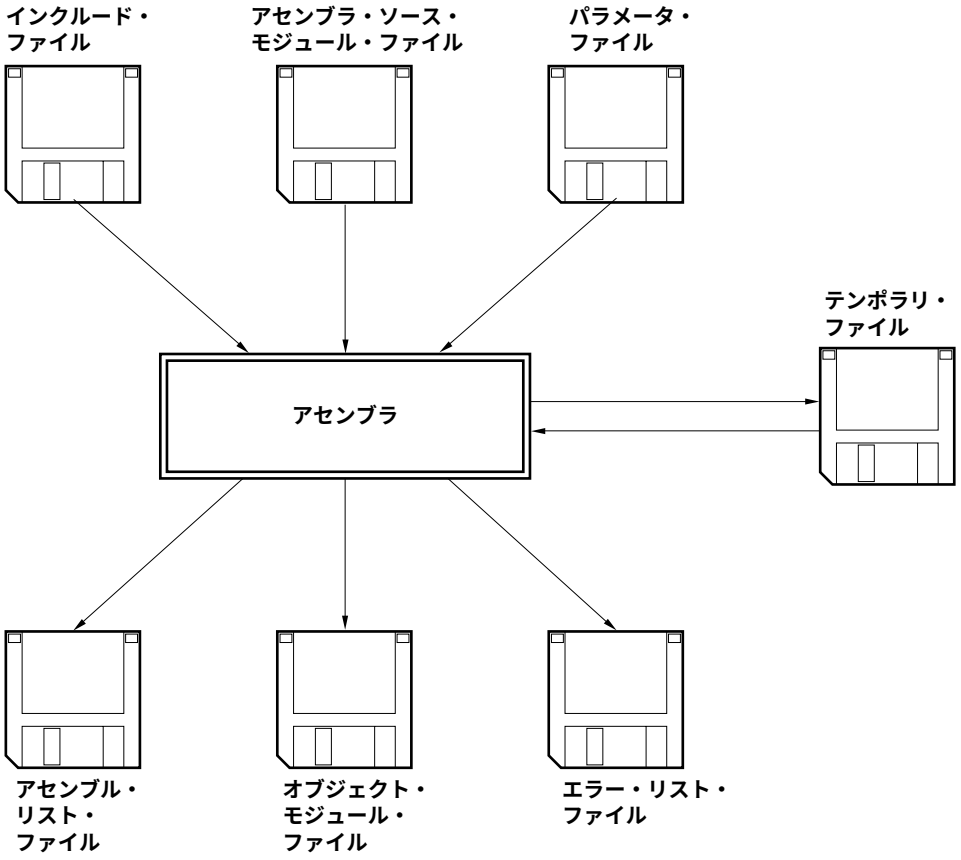
5. 1　アセンブラの入出力ファイル

アセンブラの入出力ファイルを次に示します。

表5－1　アセンブラの入出力ファイル

種　類	ファイル名	説　　明	ディフォールト・ファイル・タイプ
入力ファイル	アセンブラ・ソース・モジュール・ファイル	○78K0Sシリーズ用のアセンブリ言語で記述されたソース・モジュール・ファイルです。 ○ユーザ作成ファイルです。	.ASM
	インクルード・ファイル	○アセンブラ・ソース・モジュール・ファイルで参照するファイルです。 ○78K0Sシリーズ用のアセンブリ言語で記述されたファイルです。 ○ユーザ作成ファイルです。	なし
	パラメータ・ファイル	○実行プログラムのパラメータを内容とするファイルです。 ○ユーザ作成ファイルです。	.PRA
出力ファイル	オブジェクト・モジュール・ファイル	○機械語情報と機械語の配置アドレスに関する再配置情報およびシンボル情報を含んだバイナリ・ファイルです。	.REL
	アセンブル・リスト・ファイル	○アセンブル・リスト，クロスレファレンス・リストなどのアセンブル情報を持つファイルです。	.PRN
	エラー・リスト・ファイル	○アセンブル時のエラー情報を持つファイルです。	.ERA
入出力ファイル	テンポラリ・ファイル	○アセンブルのためにアセンブラが自動生成するファイルです。アセンブル終了時には消去されます。	RAxxxxx. $$n (n = 1-4)

図5－1　アセンブラの入出力ファイル



5. 2　アセンブラの機能

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

(2) アセンブラは，ファイルやシステムに関するエラーがある場合は，アボート・エラーを出力し，ソース・モジュール中に記述エラーを発見した場合は，フェイタル・エラーやワーニング・エラーを出力します。

　アボート・エラー，フェイタル・エラーの場合は，通常オブジェクト・モジュール・ファイルは出力されません。ただし，-Jオプションが指定された場合には，フェイタル・エラーがある場合でもオブジェクト・モジュール・ファイルを出力します。

(3) アセンブラは，アセンブル起動時に指定されたアセンブラ・オプションに従ってアセンブル処理を行います。アセンブラ・オプションの詳細については，5. 4　**アセンブラ・オプション**を参照してください。

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

5. 3　アセンブラの起動方法

5. 3. 1　アセンブラの起動

アセンブラの起動には，次の2つの方法があります。

(1) コマンド行での起動

X> [パス名] RA78K0S [△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・ディレクトリ名

③ アセンブラのコマンド・ファイル名

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。アセンブラ・オプションの詳細については，5. 4　**アセンブラ・オプション**を参照してください。

⑤ アセンブルするソース・モジュール・ファイル名

例　A>ra78k0s –c9024 k0smain.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>RA78K0S [△ソース・モジュール・ファイル] △-Fパラメータ・ファイル名
                                          |    |
                                          ②    ①
```

① アセンブラの起動に必要な情報を含んだファイル

② パラメータ・ファイル（指定オプション）

パラメータ・ファイルは，エディタなどで作成します。

パラメータ・ファイル内での記述規則を次に示します。

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

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

ソース・モジュール・ファイル名は，オプションのあとに記述することも可能です。

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

パラメータ・ファイルについての詳細は5. 4. 3　**アセンブラ・オプションの説明**を参照してください。

例　パラメータ・ファイル（K0SMAIN.PRA）をエディタで作成します。

●K0SMAIN.PRAの内容

```
;parameter file
k0smain.asm -osample.rel
-psample.prn
```

●パラメータ・ファイル（K0SMAIN.PRA）を使用してアセンブラを起動します。

```
A>ra78k0s –fk0smain.pra
```

5. 3. 2　実行開始，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，ディスプレイに実行開始メッセージが表示されます。

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

(2) 実行終了メッセージ

アセンブルの結果，アセンブル・エラーが検出されなかった場合，アセンブラは次のメッセージをディスプレイに出力して制御を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ディスプレイに出力して制御をOSに戻します。

```
Pass1 Start
K0SMAIN. ASM (12) :F201 Syntax error
Pass2 Start
K0SMAIN. ASM (12) :F201 Syntax error
K0SMAIN. ASM (29) :F407 Undefined symbol reference 'CONVAH'
K0SMAIN. ASM (29) :F303 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ディスプレイに出力してアセンブルを中止し，制御をOSに戻します。

例1.　存在しないソース・モジュール・ファイルを指定した場合

```
A>ra78k0s sample.asm

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

A006 File not found 'SAMPLE. ASM'
Program aborted.
```

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

を中止しました。

例2.　**存在しないアセンブラ・オプションを指定した場合**

```
A>ra78k0s k0smain.asm -z

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

AO12 Missing parameter '-z'
Please enter 'RA78K0S--', if you want help messages.
Program aborted.
```

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

アセンブラがエラー・メッセージを出力してアセンブルを中止した場合，そのエラー・メッセージの原因を**第**12**章　エラー・メッセージ**で調べて対処してください。

5. 4　アセンブラ・オプション

5. 4. 1　アセンブラ・オプションの種類

アセンブラ・オプションは，アセンブラの動作に細かい指示を与えるものです。アセンブラ・オプションは，15種類のオプションに分類できます。

表5－2　アセンブラ・オプション

項番	分　類	オプション	説　明
1	デバイス種別指定	-C	対象デバイスの種別を指定します。
2	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	
3	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	
4	ディバグ情報出力指定	-G	ディバグ情報（ローカル・シンボル情報）をオブジェクト・モジュール・ファイルへ出力します。
		-NG	
		-GA	アセンブラ・ソース・ディバグ情報をオブジェクト・モジュール・ファイルへ出力します。
		-NGA	
5	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。
6	アセンブル・リスト・ファイルの出力指定	-P	アセンブル・リスト・ファイルの出力を指定します。
		-NP	
7	アセンブル・リスト・ファイル情報指定	-KA	アセンブル・リスト・ファイル中にアセンブル・リストを出力します。
		-NKA	
		-KS	アセンブル・リスト・ファイル中にシンボル・リストを出力します。
		-NKS	
		-KX	アセンブル・リスト・ファイル中にクロスレファレンス・リストを出力します。
		-NKX	
8	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	
9	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	
10	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。
11	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。
12	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。
13	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。
14	シンボル定義指定	-D	シンボルの定義を行います。
15	ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

5.4.2　アセンブラ・オプションの優先度

表5－3に示すアセンブラ・オプションのうち，縦軸のものと横軸のものを同時に2つ以上指定した場合の優先度について説明します。

表5－3　アセンブラ・オプションの優先度

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

←横軸

↑
縦軸

【×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　A>ra78k0s –c9024 k0smain.asm -no -lw80 -lf

-LW, -LFオプションは，無効となります。

【△で記した箇所】

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

例　A>ra78k0s –c9024 k0smain.asm -p -nka -nks -nkx

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

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

例　A>ra78k0s –c9024 k0smain.asm -o -no

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

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

5.4.3　アセンブラ・オプションの説明

各アセンブラ・オプションの詳細について説明します。

(1) デバイス種別指定　(-C)

記述形式　：-Cデバイス種別
省略時解釈：省略不可

【機　　能】

-Cオプションは，アセンブルの対象となる対象デバイスを指定します。

【用　　途】

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

【説　　明】

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

【注　　意】

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

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

制御命令については言語編**第4章　制御命令**を参照してください。

【使 用 例】

コマンド行で指定します。

```
A>ra78k0s –c9024 k0smain.asm
```

(2) オブジェクト・モジュール・ファイル出力指定　(-O/-NO)

記述形式　：-O[出力ファイル名]
　　：-NO
省略時解釈：-O入力ファイル名.REL

【機　　能】

① -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。
② -NOオプションは，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】

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】

① ［出力ファイル名］には，ディスク型ファイル名，デバイス型ファイル名のNUL, AUX，パス名を指定できます。デバイス型ファイル名CON, PRN, CLOCKが指定されたときは，アボート・エラーとなります。
② -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。
③ -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。
④ -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。
⑤ -Oと-NOの両オプションを同時に指定した場合は，あとで指定した方を優先します。

【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。

```
A>ra78k0s –c9024 k0smain.asm -osample.rel
```

(3) オブジェクト・モジュール・ファイル強制出力指定　(-J/-NJ)

記述形式　：-J
　　：-NJ
省略時解釈：-NJ

【機　　能】

① -Jオプションは，フェイタル・エラーの場合でもオブジェクト・モジュール・ファイルを出力するように指定します。

② -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，フェタル・エラーがあるのを承知でプログラムを実行させたい場合には，-Jオプションを指定したオブジェクト・モジュール・ファイルを出力します。

【説　　明】

① -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。

② -Jと-NJの両オプションを同時に指定した場合は，あとで指定した方を優先します。

【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・ファイルを出力するよう指定します。

```
A>ra78k0s –c9024 k0smain.asm -j
```

（4）ディバグ情報出力指定　（-G/-NG, -GA/-NGA）

（a）-G/-NG

記述形式　：-G
　　：-NG
省略時解釈：-G

【機　　能】

① -Gオプションは，オブジェクト・モジュール・ファイル中に，ディバグ情報（ローカル・シンボル情報）を付加する指定をします。

② -NGオプションは，-Gオプションを無効にします。

【用　　途】

① -Gオプションは，ローカル・シンボルも含めシンボリック・ディバグを行うときに使用します。

② -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・ディバグ
2．シンボルなしでのディバグ
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，あとに指定した方を有効とします。

【注　　意】

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

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

制御命令については，言語編の第4章　制御命令を参照してください。

【使 用 例】

オブジェクト・モジュール・ファイルにディバグ情報を付加します。

```
A>ra78k0s –c9024 k0smain.asm -g
```

（b）-GA/-NGA

記述形式　：-GA
　　：-NGA
省略時解釈：-GA

【機　　能】

① -GAオプションは，オブジェクト・モジュール・ファイル中に構造化アセンブラでソース・ディバグ用の情報を出力する指定をします。

② -NGAオプションは，-GAオプションを無効にします。

【用　　途】

① -GAオプションは，アセンブラまたは構造化アセンブラのソース・レベルでディバグするときに使用します。ソース・レベルでのディバグには「統合ディバッガ（別売）」が必要です。

② -NGAオプションは，次の2種類の場合に使用します。

1．アセンブラ・ソースなしでのディバグ

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）

【説　　明】

① -GAと-NGAの両オプションが同時に指定された場合は，あとに指定した方を有効とします。

② -GAオプションは指定した位置にかかわらず優先されます。

【注　　意】

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

```
▲$▲DEBUGA
▲$▲NODEBUGA
```

制御命令については，言語編の第4章　制御命令を参照してください。

【使 用 例】

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

```
A>ra78k0s –c9024 k0smain.asm -ga
```

(5) インクルード・ファイル読み込みパス指定　(-I)

記述形式　：-Iパス名[，パス名]…（複数指定可能）
省略時解釈：ソース・ファイルのあるパス
　　：環境変数（INC78K0S）により指定されたパス

【機　　能】

ソース・モジュール中の‘ $include ’で指定されたインクルード・ファイルを指定したパスから入力する指定をします。

【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】

① ‘ , ’で区切ることにより，一度に複数のパス名を指定できます。
② ‘ , ’の前後には空白を入れることはできません。
③ -Iに続いてパス名が複数指定されるか，あるいは-Iオプションが複数指定された場合，指定された順番に‘ $include ’で指定したファイルを検索します。その後，省略時解釈と同じ順序で検索します。
④ -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。
⑤ -Iが9個以上指定された場合には，アボート・エラーとなります。

【使 用 例】

インクルード・ファイルをディレクトリB：¥SAMPLEから読み込みます。

```
A>ra78k0s –c9024 k0smain.asm –ib:¥sample
```

(6) アセンブル・リスト・ファイル出力指定　(-P/-NP)

記述形式　：-P[出力ファイル名]
　　：-NP
省略時解釈：-P入力ファイル名.PRN

【機　　能】

① -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。
② -NPオプションは，-Pオプションを無効にします。

【用　　途】

① 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。
② 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】

① ファイル名としてディスク型ファイル名とデバイス型ファイル名を指定できます。
　指定できるデバイス型ファイル名は，COM, PRN, NULおよびAUXです。CLOCKを指定した場合，アボート・エラーとなります。
② -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。
③ -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。
④ -Pと-NPの両オプションを同時に指定した場合は，あとで指定した方を優先します。

【使 用 例】

アセンブル・リスト・ファイル（SAMPLE.PRN）を作成します。

```
A>ra78k0s –c9024 k0smain.asm -psample.prn
```

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

（a）-KA/-NKA

記述形式　：-KA
　　：-NKA
省略時解釈：-KA

【機　　能】

① -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・リストを出力する指定をします。

② -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】

① -KAと-NKAの両オプションを同時に指定した場合は，あとで指定した方を優先します。

② -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。

【使 用 例】

アセンブル・リストを出力します。

```
A>ra78k0s –c9024 k0smain.asm -ka
```

K0SMAIN.PRNを参照します。

```
   Assemble list

 ALNO  STNO  ADRS  OBJECT  M I  SOURCE STATEMENT

    1     1                          
    2     2                                NAME    SAMPM
    3     3                        ;**********************************
    4     4                        ;
    5     5                        ;    HEX ->ASCII Conversion Program
    6     6                        ;
    7     7                        ;         main-routine
    8     8                        ;
    9     9                        ;**********************************
   10    10                        
   11    11                               PUBLIC MAIN, START
   12    12                               EXTRN  CONVAH
   13    13                               EXTRN  _@STBEG
   14    14                        
   15    15 ----             DATA   DSEG  saddr
   16    16 0000             HDTSA: DS  1
   17    17 0001             STASC: DS  2
   18    18                        
   19    19 ----             CODE   CSEG  AT 0H
   20    20 0000 R0000       MAIN:  DW  START
   21    21                        
   22    22 ----                    CSEG
   23    23 0000             START:
   24    24                        
   25    25                         ;chip initialize
   26    26 0000 RF00000            MOVW   AX, #_@STBEG
   27    27 0003 E61C               MOVW   SP, AX
   28    28                        
   29    29 0005 RF5001A            MOV    HDTSA, #1AH
   30    30 0008 RFC0000            MOVW   HL, #HDTSA  ;set hex 2-code data
in HL registor
   31    31                        
   32    32 000B R220000            CALL   !CONVAH     ;convert ASCII<- HE
X
   33    33                                            ;output BC-register
<- ASCII code
   34    34 000E RF80100            MOVW   DE, #STASC  ;set DE <- store ASC
II code table
   35    35 0011 0A27               MOV    A, B
   36    36 0013 EB                 MOV    [DE], A
   37    37 0014 88                 INCW   DE
   38    38 0015 0A25               MOV    A, C
   39    39 0017 EB                 MOV    [DE], A
   40    40                        
   41    41 0018 30FE               BR     $$
   42    42                        
   43    43                         END
```

（b）-KS/-NKS

記述形式　：-KS
**　　：**-NKS
省略時解釈：-NKS

【機　　能】

① -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・ファイル中に出力する指定をします。**

② -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】

① -KS**と**-NKS**の両オプションを同時に指定した場合は，あとで指定した方を優先します。**

② -KS**と**-KX**を同時に指定した場合，**-KS**を無視します。**

③ -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。**

【使 用 例】

シンボル・リストを出力します。

```
A>ra78k0s –c9024 k0smain.asm -ks
```

K0SMAIN.PRN**を参照します。**

（アセンブル・リストに続いてシンボル・リストが出力されています）。

```
   Symbol Table List

VALUE    ATTR  RTYP  NAME              VALUE    ATTR  RTYP  NAME

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

（c）-KX/ーNKX

記述形式　：-KX
　　：-NKX
省略時解釈：-NKX

【機　　能】

① -KXオプションは，アセンブル・リストに続いてクロスレファレンス・リストをアセンブル・リスト・ファイル中に出力する指定をします。

② -NKXオプションは，-KXオプションを無効にします。

【用　　途】

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

【説　　明】

① -KXと-NKXの両オプションを同時に指定した場合は，あとで指定した方を優先します。

② -KSと-KXを同時に指定した場合，-KSを無視します。

③ -NKA, -NKSおよび-NKXを同時に指定した場合，アセンブル・リスト・ファイルは出力されません。

【注　　意】

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

```
▲$▲XREF
▲$▲XR                          ；短縮形
▲$▲N0XREF
▲$▲N0XR                        ；短縮形
```

制御命令については，言語編の第4章　制御命令を参照してください。

【使 用 例】

クロスレファレンス・リストを出力します。

```
A>ra78k0s –c9024 k0smain.asm -kx
```

K0SMAIN.PRN**を参照します。**

アセンブル・リストに続いてクロスレファレンス・リストが出力されています。

```
        Cross-Reference List

NAME          VALUE    R ATTR  RTYP  SEGNAME XREFS

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

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

（a）-LW

記述形式　：-LW[文字数]
省略時解釈：-LW132（ディスプレイ出力の場合は80文字とします）

【機　　能】

-LWオプションは，リスト・ファイルの1行の文字数を指定します。

【用　　途】

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

【説　　明】

① -LWオプションで指定できる文字数の範囲を次に示します。
（ディスプレイ出力の場合は80文字まで）。

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。

② 文字数が省略された場合は，132を指定したものとなみします。
ただし，アセンブル・リスト・ファイルの出力先がディスプレイの場合は80とします。

③ 指定する文字数にはターミネータ（CR, LF）は含みません。

④ -NPオプションを指定した場合，-LWオプションは無効となります。

【注　　意】

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

```
▲$▲WIDTH
```

制御命令については，言語編の第4章　制御命令を参照してください。

【使 用 例】

アセンブル・リスト・ファイルの1行の文字数を80文字に指定します。

```
A>ra78k0s –c9024 k0smain.asm -lw80
```

アセンブル・リストを参照します。

```
   Assemble list

ALNO  STNO ADRS  OBJECT    M I    SOURCE STATEMENT

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

（b）-LL

記述形式　：-LL[行数]
省略時解釈：-LL66（ディスプレイ出力の場合は改頁しません）

【機　　能】

-LLオプションは，アセンブル・リスト・ファイルの1頁の行数を指定します。

【用　　途】

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

【説　　明】

① -LLオプションで指定できる行数の範囲を次に示します。

20　≦1頁に印字する行数　≦　32767

範囲外の数値や数値以外のものが指定された場合には，アボート・エラーとなります。

② 行数を省略した場合，66を指定したものとみなします。

③ 行数の0を指定した場合は改頁しません。

④ -NPオプションを指定した場合，-LLオプションは無効となります。

【注　　意】

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

```
▲$▲LENGTH
```

制御命令については，言語編の第4章　制御命令を参照してください。

【使 用 例】

アセンブル・リスト・ファイルの1頁の行数を20行に指定します。

```
A>ra78k0s –c9024 k0smain.asm -ll20
```

K0SMAIN.PRNを参照します。

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

Command:-c9024 k0smain.asm -ll20
Para-file:
In-file:K0SMAIN.ASM
Obj-file:K0SMAIN.REL
Prn-file:K0SMAIN.PRN

     Assemble list

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


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


ALNO  STNO ADRS  OBJECT    M I    SOURCE STATEMENT

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

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


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


ALNO  STNO ADRS  OBJECT    M I    SOURCE STATEMENT

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

（c）-LH

記述形式　：-LH文字列
省略時解釈：なし

【機　　能】

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

【用　　途】

① 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。
② 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】

① 指定可能な文字列は，60文字以内です。ただし，文字列内に空白は記述できません。
② 61文字以上記述された場合には，先頭の60文字を有効とし，エラーは出力されません。
なお，漢字，ひらがなは，1文字を2文字として計算します。
1行の最大文字数が119以下の場合，タイトルとしての文字列の有効長は次のとおりです。

有効長 =（1行の最大文字数）－60

③ 文字列が指定されなかった場合，アボート・エラーとなります。
④ -NPオプションを指定した場合，-LHオプションは無効となります。
⑤ -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。
⑥ 記述可能な文字セットを次に示します。

表5－4　タイトルとして記述可能な文字

文　　字	コマンド行	パラメータ・ファイル中
＊？＞＜｜	”” でくくることで記述可能	記述可能 ”” でくくってもコマンド行と同じように解釈します
；	”” でくくることで記述可能	記述不可 （コメントとみなされます）
＃	記述可能	記述不可 （コメントとみなされます）
”（ダブル・クォーテーション）	有効文字としては記述不可	有効文字としては記述不可
00H	記述不可	記述可能 ただし，文字列が切れたとみなされます
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・ファイル中では‘！’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　コマンド行上の＊は，ワイルド・カード展開の対象とならない場合は”” でくくらなくても記述可能です。

【注　　意】

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

次に記述形式を示します。

```
▲$▲TITLE▲（▲ ′文字列′ ▲）
▲$▲TT▲（▲ ′文字列′ ▲）                    ；短縮形
```

制御命令については，言語編の**第4章　制御命令**を参照してください。

【使 用 例】

アセンブル・リスト・ファイルのヘッダにタイトルを印字します。

```
A>ra78k0s –c9024 k0smain.asm -lhRA78K0S_MAINROUTINE
```

K0SMAIN. PRN**を参照します。**

```
78K/0S Series Assembler Vx. xx  RA78K0S_MAINROUTINE   Date:xx xxx xxx Page:  1
                                        └── タイトル

Command:  -c9024 k0smain. asm -lhRA78K0S_MAINROUTINE
Para-file:
In-file:K0SMAIN.ASM
Obj-file:K0SMAIN.REL
Prn-file:K0SMAIN.PRN

   Assemble list

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

（d）-LT

記述形式　：-LT[文字数]
省略時解釈：-LT8

【機　　能】

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

【用　　途】

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

【説　　明】

① -LTオプションで指定できる文字数の範囲は次のとおりです。

0 ≦ 指定できる文字数 ≦ 8

範囲外の数値や数値以外のものを指定した場合は，アボート・エラーとなります。

② -LT0を指定した場合，タビューレーション処理は行わず，タブ・コードを出力します。

③ -NPオプションを指定した場合，-LTオプションは無効となります。

【注　　意】

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

▲$▲TAB▲タブ数

制御命令については，言語編の第4章　制御命令を参照してください。

【使 用 例1】

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

```
   Assemble list

ALNO  STNO  ADRS  OBJECT  M I  SOURCE STATEMENT

   1    1                          NAME      SAMPLE
   2    2
   3    3  ----              CODE    CSEG
   4    4  0000  0A27                MOV       A, B
   5    5  0002  0A12                SET1      A. 1
   6    6                          END
```

【使 用 例2】

HT**コードによるブランクを**1**に指定します。**

```
   A>ra78k0s –c9024 sample.asm -lt1
```

sample.prn**を参照します。**

```
   Assemble list

ALNO  STNO  ADRS  OBJECT  M I  SOURCE STATEMENT

   1    1                    NAME SAMPLE
   2    2
   3    3  ----              CODE CSEG
   4    4  0000  0A27          MOV A, B
   5    5  0002  0A12          SET1 A. 1
   6    6                     END
```

備考　HT**コードによるブランクは**1**つです。**

（e）-LF/-NLF

記述形式　：-LF
　　：-NLF
省略時形式：-NLF

【機　　能】

① -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）を付加します。
② -NLFオプションは，-LFオプションを無効にします。

【用　　途】

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

【説　　明】

① -NPオプションを指定した場合，-LFオプションは無効となります。
② -LFと-NLFの両オプションを同時に指定した場合は，あとで指定した方を優先します。

【注　　意】

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

```
▲$▲FORMFEED
▲$▲NOFORMFEED
```

制御命令については，言語編の第4章　制御命令を参照してください。

【使 用 例】

アセンブル・リスト・ファイルの最後に改頁コードを付加します。

```
A>ra78k0s –c9024 k0smain.asm -pprn -lf
```

(9) エラー・リスト・ファイル出力指定　(-E/-NE)

記述形式　：-E[出力ファイル名]
　　：-NE
省略時形式：-NE

【機　　能】

① -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。

② -NEオプションは，-Eオプションを無効にします。

【用　　途】

① 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。

② 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】

① ファイル名としてディスク型ファイル名とデバイス型ファイル名を指定できます。
ただし，デバイス型ファイル名CLOCKを指定した場合は，アボート・エラーとなります。

② -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。

③ -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。

④ -Eと-NEの両オプションを同時に指定した場合は，あとで指定した方を優先します。

【使 用 例】

エラー・リスト・ファイル（sample.era）**を作成します。**

```
A>ra78k0s -c9024 k0smain.asm -esample.era

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

PASS_PARSE  Start
K0SMAIN.ASM(26) :F202 Illegal operand
PASS_OUTOBJ Start
K0SMAIN.ASM(32) :F407 Undefined symbol reference 'F'
K0SMAIN.ASM(41) :F407 Undefined symbol reference 'F'

  Target chip:uPD78xxxx
  Device file:Vx.xx

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

エラー・リスト・ファイル (sample.era) **を参照します。**

```
PASS_PARSE  Start
K0SMAIN.ASM(26) :F202 Illegal operand
PASS_OUTOBJ Start
K0SMAIN.ASM(32) :F407 Undefined symbol reference 'F'
K0SMAIN.ASM(41) :F407 Undefined symbol reference 'F'
Pass1 Start
```

（10）パラメータ・ファイル指定　（-F）

記述形式　：-Fファイル名
省略時解釈：コマンド行上からのみオプションまたは入力ファイル名が入力可能。

【機　　能】

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

【用　　途】

① 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。
② 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】

① ‘ファイル名’として指定できるのは，ディスク型ファイル名のみです。
　デバイス型ファイル名を指定するとアボート・エラーとなります。
② ファイル名を省略するとアボート・エラーとなります。
③ 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。
④ パラメータ・ファイル中に記述できる文字数の制限はありません。
⑤ 空白とタブおよび‘↵’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。
⑥ 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。
⑦ 展開されたオプションは，あとで指定したものを優先します。
⑧ ‘；’または‘＃’以降に記述された文字は‘↵’またはEOFの前まですべてコメントと解釈します。
⑨ -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（K0SMAIN.PRA）の内容は次のように設定します。

```
;parameter file
k0smain.asm -osample.rel –g –c9024
-psample.prn
```

コマンド行には，次のように入力します。

```
A>ra78k0s –fk0smain.pra
```

（11）テンポラリ・ファイル作成パス指定　（-T）

記述形式　：-Tパス名
省略時解釈：環境変数TMPにより指定されたパスに作成します
　　指定されていない場合は，カレント・パスに作成します

【機　　能】

-Tオプションは，テンポラリ・ファイルを作成するパスを指定します。

【用　　途】

テンポラリ・ファイルの作成場所を指定できます。

【説　　明】

① パス名としてパス以外のものは指定できません。
② パス名は省略できません。
③ 以前に作成されたテンポラリ・ファイルが存在している場合でも，ファイル保護がされていなければ，上書きします。
④ 必要とするメモリ・サイズがある間は，テンポラリ・ファイルをメモリに展開します。
　メモリが足りなくなった時点でメモリに展開していたテンポラリ・ファイルの内容をディスクに書き出します。
　以降のテンポラリ・ファイルへのアクセスは，セーブしたディスク・ファイルに対して行います。
⑤ 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。
⑥ 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。

a．-Tオプションで指定されたパス
b．環境変数TMPに設定されているパス（-Tオプション省略の場合）
c．カレント・パス（TMPが設定されていない場合）

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

【使 用 例】

テンポラリ・ファイルをディレクトリA：¥TMPに出力するよう指定します。

```
A>ra78k0s –c9024 k0smain.asm –ta:¥tmp
```

（12）漢字コード指定　（-ZS/-ZE/-ZN）

記述形式　：-ZS
　　：-ZE
　　：-ZN
省略時形式：OSにより次の通りとなります。
　　：-ZS（Windows/HP-UX）
　　：-ZE（SunOS/Solaris）

【機　　能】

① 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。

② 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】

① -ZS/-ZE/-ZNオプションが同時に指定された場合，あとに指定した方を優先します。

② ソース・モジュールの先頭で，-ZS/-ZE/-ZNオプションと同機能の制御命令を記述できます。
次に記述形式を示します。

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

制御命令については，言語編の第4章　制御命令を参照してください。

③ 環境変数（LANG78K）でも漢字コードを指定することができます。環境変数については，11. 2　開発環境の整備（環境変数）を参照してください。

【使 用 例】

漢字コードをEUCコードとして解釈します。

```
A>ra78k0s k0smain.asm –c9024 -ze
```

（13）デバイス・ファイル・サーチ・パス指定　（-Y）

記述形式　：-Yパス名

省略時形式：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＜..¥dev＞（ra78k0s.exeの起動されたパスに対して）

②RA78K0Sの起動されたパス

③カレント・ディレクトリ

④環境変数PATH

【機　　能】

デバイス・ファイルを指定されたパスから読み込みます。

【用　　途】

デバイス・ファイルが存在するパスを指定します。

【説　　明】

① -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。

② -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。

③ 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。

a．-Yオプションで指定されたパス

b．＜..¥dev＞（ra78k0s.exeの起動されたパスに対して）

c．RA78K0Sの起動されたパス

d．カレント・ディレクトリ

e．環境変数PATH

【使 用 例】

デバイス・ファイルのパスをa：¥78k0s¥devディレクトリに指定します。

```
A>ra78k0s k0smain.asm –c9024 –ya:¥78k0s¥dev
```

（14）シンボル定義指定（-D）

記述形式　：-Dシンボル名［=数値］［，シンボル名［=数値］…］
省略時形式：なし

【機　能】

シンボルの定義を行います。

【用　途】

シンボルの定義を行いたい場合，-Dオプションを指定します。

【説　明】

① 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。
② シンボルはカンマで区切ることにより，30個まで指定できます。
③ シンボル名は，30文字まで記述できます。
④ 同名のシンボル名が重複された場合，あとに指定した方を有効とします。
⑤ シンボル名の英字は，大文字，小文字を区別します。
⑥ -Dに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。

```
A>ra78k0s k0smain.asm –c9024 –dSYM=2
```

（15）ヘルプ指定　（--）

記述形式　：--
省略時形式：表示しない

【機　　能】

--オプションは，ヘルプ・メッセージをディスプレイに表示します。

【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】

① --オプションを指定すると他のアセンブラ・オプションは，すべて無効となります。

② ヘルプ・メッセージの続きをお読みになる場合は，リターン・キーを入力してください。
表示を途中で終了する場合は，リターン・キー以外の文字を入力したあとで，リターン・キーを入力してください。

注意　このオプションは，PM plus**上では指定できません。**
PM plus**上でヘルプを参照する場合は，〈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。

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

usage : ra78k0s [option[...]] input-file [option[...]]
The option is as follows ([] means omissible).
-cx          :Select target chip. ( x = 9012,p9014 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].
-zs/-ze/-zn  :Change source regulation.
              -zs:SJIS code usable in comment.
              -ze:EUC code usable in comment.
              -zn:no multibyte code in comment.
--           :Show this message.
DEFAULT ASSIGNMENT:
  -o -ne -p -ka -nks -nkx -lw132 -ll66 -nlf -lt8 -g -nj -ga
```

5. 5　PM plusでのオプション設定

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

5. 5. 1　オプションの設定方法

PM plusの［ツール(T)］メニューの［アセンブラオプションの設定(A)...］を選択または  をクリックすると，<アセンブラオプションの設定>ダイアログが現れます。

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

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

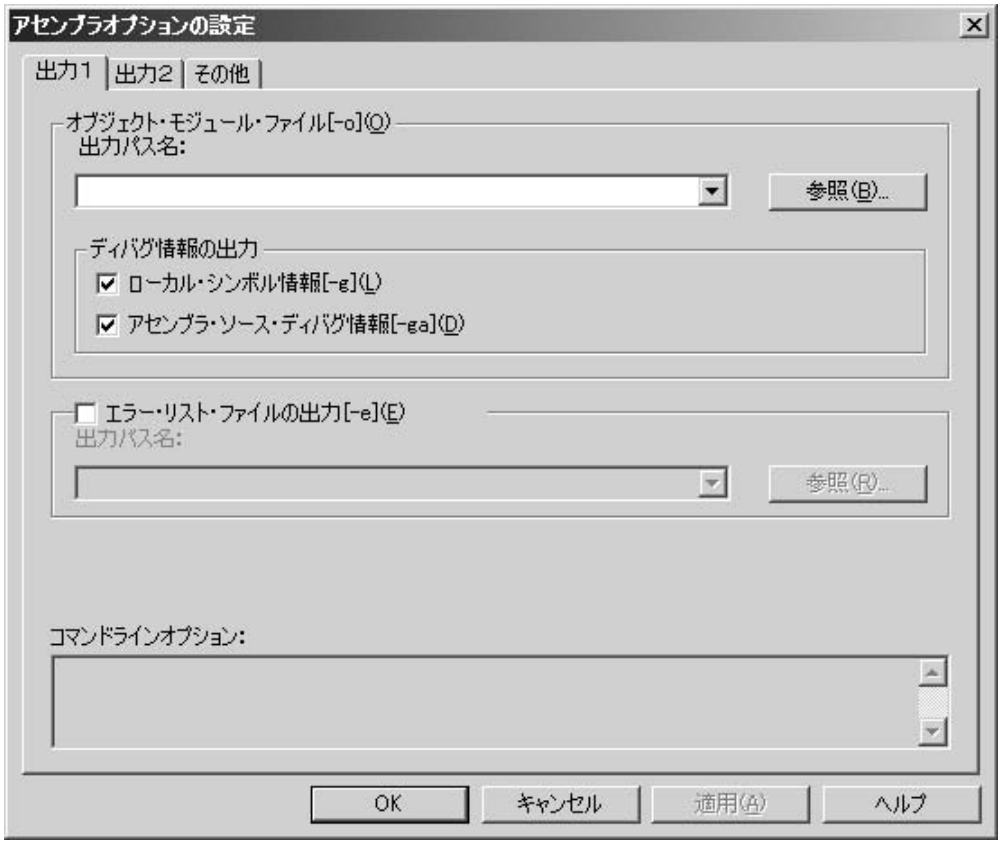


図5－3　<アセンブラオプションの設定> ダイアログ（《出力2》タブ選択時）

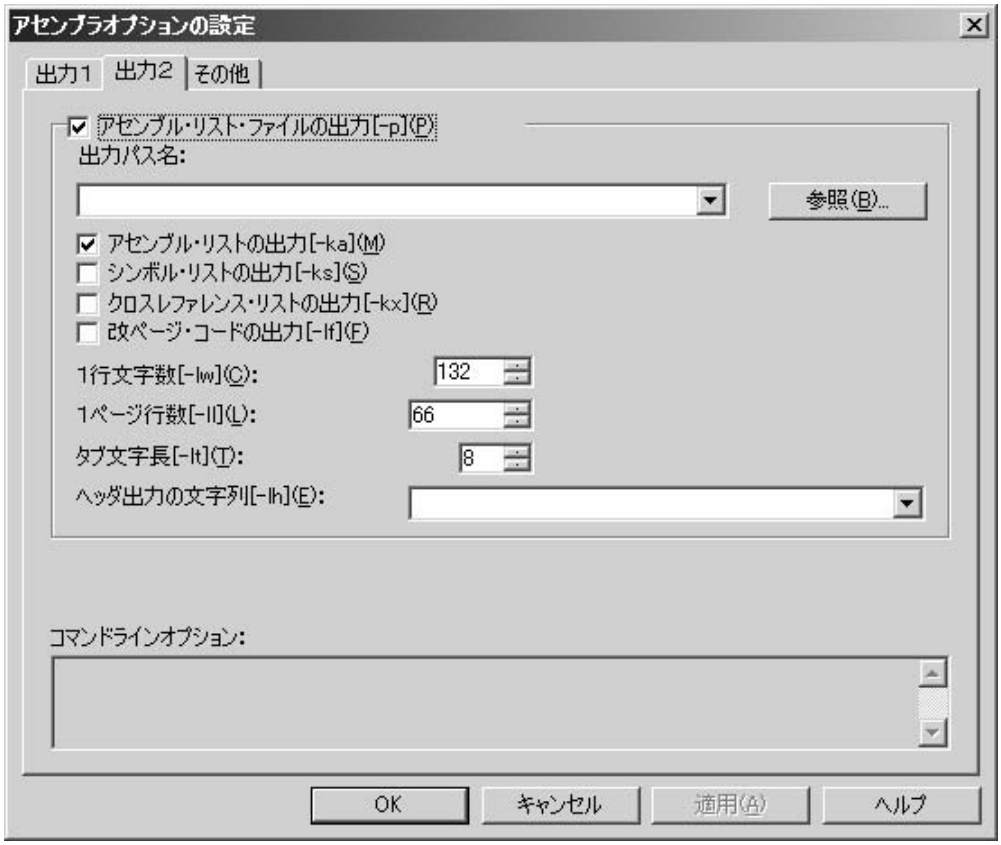
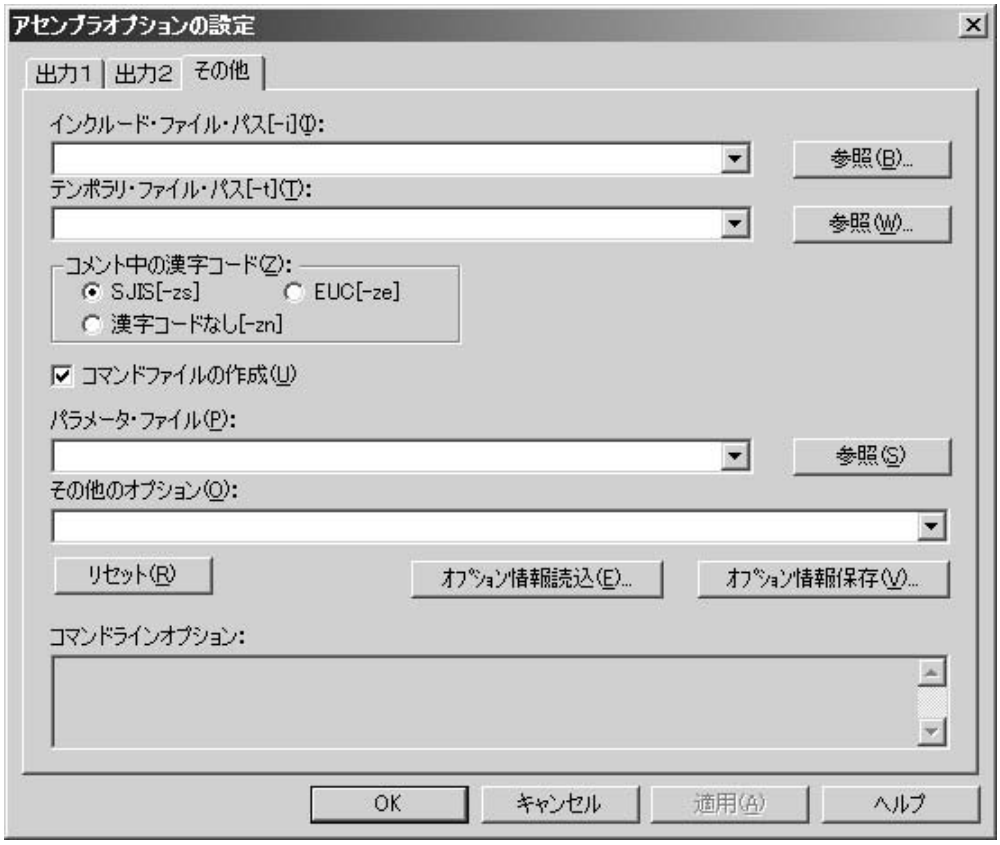


図5－4　<アセンブラオプションの設定> ダイアログ（《その他》タブ選択時）



5.5.2　各オプションの設定

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

・オブジェクト･モジュール･ファイル[-o](O)

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･ファイルの出力パスを指定します。

・ローカル･シンボル情報[-g](L)

ローカル･シンボルも含めシンボリック･ディバグを行います。

・アセンブラ･ソース･ディバグ情報[-ga](D)

アセンブラ･ソース･ディバグ情報を付加します。

・エラー･リスト･ファイルの出力[-e](E)

出力パス名：

エラー･リスト･ファイルを出力したい場合は，入力ボックスにファイル名を入力します。

パスを指定する場合は参照(R)ボタンを使用します。

・アセンブル･リスト･ファイルの出力[-p](P)

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･ファイルの出力パスを指定します。

・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･リストを出力します。

・シンボル･リストの出力[-ks](S)

シンボル･リストをアセンブル･リストに続いて出力します。

・クロスレファレンス･リストの出力[-kx](R)

クロスレファレンス･リストをアセンブル･リストに続いて出力します。

・改ページ･コードの出力[-lf](F)

アセンブル･リスト･ファイルの内容を印字したあとで改ページ･コードを付加します。

・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(72～2046文字まで選択可能)。

・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(20～32767文字まで選択可能)。

・タブ文字長[-lt](T)

タブ文字長を指定します(0～8文字まで選択可能)。

・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(60文字まで)。

・インクルード･ファイル･パス[-i](I)

参照(B)ボタンまたは直接入力によりインクルード･ファイルを読み込むパスを指定します。

・テンポラリ･ファイル･パス[-t](T)

参照(W)ボタンまたは直接入力によりテンポラリ･ファイルの作成場所を指定します。

・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。

・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チェックしてください。

・パラメータ･ファイル(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･ファイルを読み込みます。

・その他のオプション(O)

チェック･ボックスやラジオ･ボタンで選択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。

・リセット(R)

入力した内容をリセットします。

・オプション情報読込(E)

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

・オプション情報保存(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。

第6章　リ ン カ

リンカは，78K0Sシリーズ用のアセンブラが出力したいくつかのオブジェクト・モジュール・ファイルを入力し，配置アドレスを決定して1つにまとめたロード・モジュール・ファイルを出力します。

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

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

6. 1 リンカの入出力ファイル

リンカの入出力ファイルを次に示します。

表6－1 リンカの入出力ファイル

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

6.2 リンカの機能

リンカの機能を次に示します。

(1) 入力セグメントの結合

リンカは，セグメントごとに配置するアドレスを決定し，管理します。

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

(2) 入力モジュールの決定

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

(3) 入力セグメントの配置アドレス決定

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

(4) オブジェクト・コードの修正

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

6. 3　メモリ空間とメモリ領域

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

注意　メモリ空間が1つしかないため，/REGULAR**空間以外は指定できません。**

メモリ空間：64 K**バイトごと**
メモリ領域：1つのメモリ空間をさらにいくつかの領域に分割する。
実装しているメモリのアドレスを宣言する。

表6－2　**セグメントの配置のグループ分け（外付け**ROM**など）**

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

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

2.　**具体例は，図**3－15　**リンク・ディレクティブ・ファイル（**sample.dr**）の内容を参照してください。**

6.4 リンク・ディレクティブ

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

ディレクティブ・ファイルの役割

(1) 実装メモリのアドレスを宣言。
(2) メモリをいくつかの領域に分割。

例 CALLT領域
内蔵ROM
外付けROM
SADDR領域
SADDR領域以外の内蔵RAM

(3) セグメントの配置をリンカで指定する。
各セグメントに対し，次の内容を指定します。

・アブソリュート・アドレス
・メモリ領域のみ指定

エディタなどでディレクティブ・ファイル（デイレクティブを記述したファイル）を作成し，リンカの起動時に，-Dオプションを指定して作成したファイルを読み込みます。
リンカは，ファイルからディレクティブを読み込み，解釈しながらリンク処理を行います。
デイレクティブには，次の2種類があります。

表6－3 ディレクティブの種類

項番	ディレクティブの種類	説 明
1	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。
2	セグメント配置ディレクティブ	○セグメントの配置を指定します。

6.4.1 ディレクティブ・ファイル

ディレクティブ・ファイル中に記述するディレクティブの記述フォーマットを次に示します。
ディレクティブは，1つのディレクティブ・ファイル中に複数記述できます。

① 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］[注]

注 省略時は/REGULARと解釈されるため，実際は記述する必要はありません。

② セグメント配置ディレクティブ

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

(1) 予 約 語

ディレクティブ・ファイル中での予約語を次に示します。

MEMORY, MERGE, AT, SEQUENT, COMPLETE

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

例
```
MEMORY
memory
Memory    :  使用不可
```

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

SEQUENT ：セグメントを出現順に，順次空きを作らないようにマージします。BSEGはビット単位で出現順にマージします。
COMPLETE：同名のセグメントが複数存在する場合はエラーとします。

例
```
MERGE DSEG1:COMPLETE=RAM
```

(2) シンボル

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

(3) 数　値

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

例
```
23H, 0FC80H
```

(4) コメント文

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

例 下線部がコメントとなります。

```
;DIRECTIVE FILE FOR 789024
MEMORY MEM1: (01000H, 1000H) #SECOND MEMORY AREA
```

6.4.2 メモリ・ディレクティブ

メモリ・ディレクティブは，メモリ領域（実装するメモリのアドレスと名前）を定義するディレクティブです。

定義したメモリ領域は，その名前（メモリ領域名）によってセグメント配置ディレクティブで参照できます。

メモリ領域は，ディフォールトで定義されているメモリ領域を含め100個まで定義できます。

【構　文】

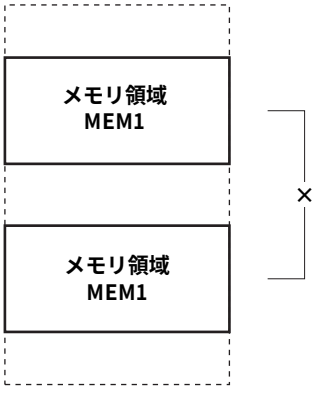
MEMORY△**メモリ領域名▲**:**▲**（**▲スタート・アドレス▲，▲サイズ▲**）[/**▲メモリ空間名**]

(1) メモリ領域名

定義するメモリ領域の名前を指定します。指定時の条件は次のとおりです。

① 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。
② 大文字と小文字は別の文字として区別します。
③ 大文字と小文字は混在できます。
④ メモリ領域名の長さは，最大31文字です。32文字以上記述するとエラーとなります。
⑤ 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許されません。

図6－1 メモリ領域名



(2) スタート・アドレス

定義するメモリ領域の先頭アドレスを指定します。

0H-FFFFH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 1以上の数値定数を記述します。

② リンカがディフォールトで定義しているメモリ領域のサイズを指定し直す場合には，定義可能な範囲の制約があります。

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

(4) メモリ空間名

メモリ空間名は， 64 K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。

① メモリ空間名は，大文字で記述します。

② 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。

③ ‘ / ’ 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】

① 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。

② 1つのメモリ・ディレクティブで1つのメモリ領域を定義できます。

③ 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。

④ ディフォールトのメモリ領域は，メモリ・ディレクティブで同一のメモリ領域を再定義しないかぎり有効です。メモリ・ディレクティブの記述を省略した場合，リンカが持つ各デバイスごとのディフォールトのメモリ領域のみを指定したものとします。

⑤ ディフォールトのメモリ領域を使用せずに，別の領域名でご使用になるときは，ディフォールトの領域名のサイズを “ 0 ” に設定してください。

【使 用 例】

メモリ空間のアドレス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)
```

6.4.3 セグメント配置ディレクティブ

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

【構　文】

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

(1) セグメント名

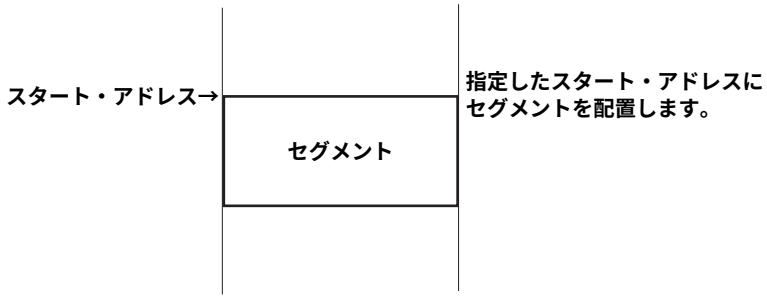
リンカに入力するオブジェクト・モジュール・ファイル中に含まれるセグメント名です。

① 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。
② 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。

① 予約語ATは，大文字または小文字のいずれか一方で記述しなければなりません。
なお，大文字と小文字を混同してはなりません。
② 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。
2. セグメント定義疑似命令のAT指定またはORG疑似命令によって配置アドレスを指定したセグメントに対して，リンク・ディレクティブでスタート・アドレスは指定できません。

(3) メモリ空間名

メモリ空間名は，セグメントを配置するメモリ空間を指定します。

① メモリ空間名として指定できるのは，REGULARのみです。
② メモリ空間名は，大文字で記述します。
③ メモリ空間名を省略した場合，REGULARを指定したものとみなします。

次にセグメントの配置先を示します。

表6－4 メモリ領域名指定とメモリ空間の組み合わせによるセグメントの配置

メモリ領域名指定	メモリ空間	セグメントの配置先
×	×	REGULAR空間中のディフォールト状態のとき配置されるメモリ領域
メモリ領域名	×	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。

【注 意】

① セグメント配置ディレクティブが指定されなかった入力セグメントは，アセンブル時にセグメント定義疑似命令で指定した再配置属性に従って配置アドレスを決定します。
② セグメント名として指定したセグメントが存在しない場合はエラーです。
③ 同一のセグメントに対して，セグメント配置ディレクティブを複数回指定した場合エラーとなります。

【使 用 例】

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

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

① 入力セグメントSEG1をROM領域中の500Hに割り付ける場合

```
MERGE  SEG1:AT (500H)
```

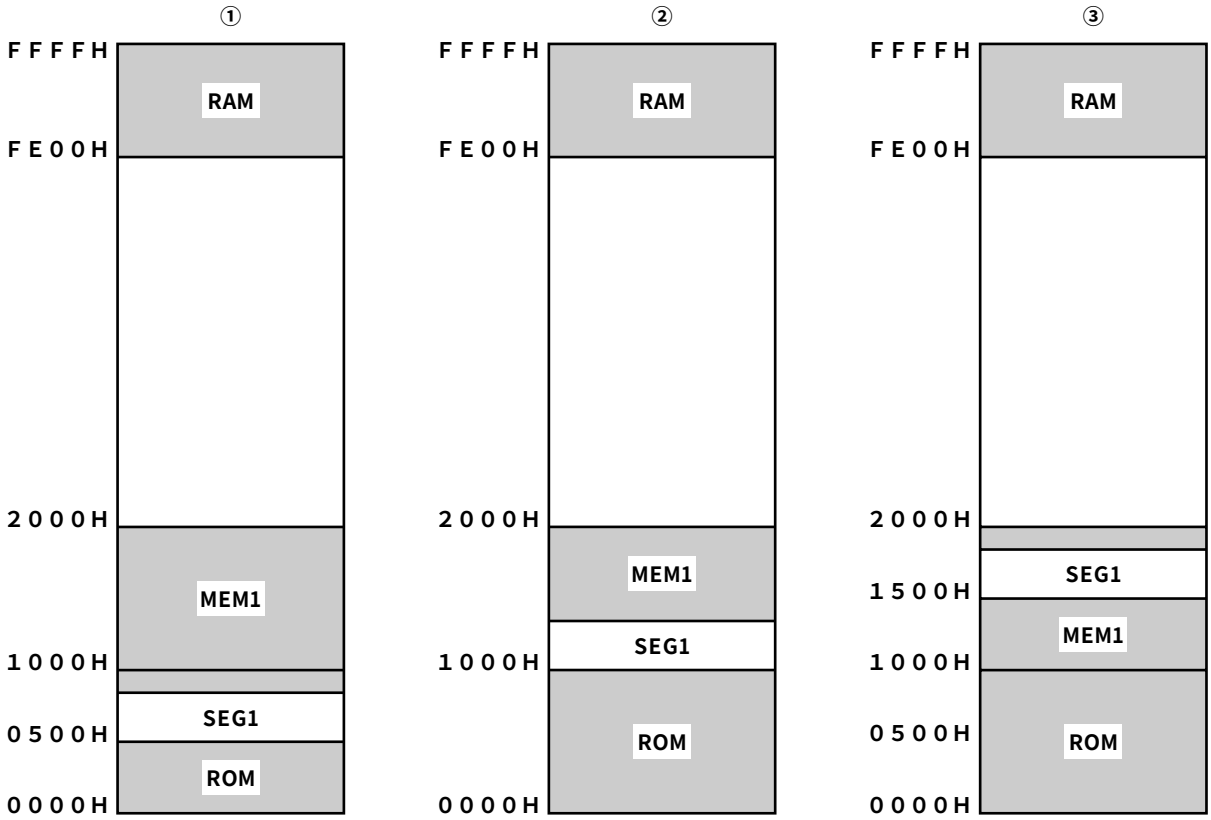
② 入力セグメントSEG1をメモリ領域MEM1中に割り付ける場合

```
MERGE  SEG1:= MEM1
```

③ 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

```
MERGE  SEG1:AT (1500H) = MEM1
```

図6－2　セグメント配置の具体例



6.5 リンカの起動方法

6.5.1 リンカの起動

リンカの起動には，次の2つの方法があります。

(1) コマンド行での起動

```
X> [パス名] lk78k0s [△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・ディレクトリ名

③ リンカのコマンド・ファイル名

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，それぞれのリンカ・オプション間を空白で区切ってください。

⑤ リンカに対して動作の詳細を指示します。

入力モジュールとして，最大256個入力できます。

例 A>lk78k0s k0smain.rel k0ssub.rel –ok0s.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>LK78K0S [△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
```

① パラメータ・ファイル指定オプション

② 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] … [△] △] ] …
```

① 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。

② 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。

③ パラメータ・ファイルには，コマンド行で指定すべきすべてのリンカ・オプション，出力ファイル名を記述します。

例 パラメータ・ファイル（K0S.PLK）をエディタで作成します。

K0S.PLKの内容

```
;parameter  file
k0smain.rel k0ssub.rel –ok0s.lmf –pk0s.map -e
-ta:¥tmp
```

パラメータ・ファイルK0S.PLKを使用してリンカを起動します。

```
A>lk78k0s –fk0s.plk
```

6.5.2 実行開始，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，ディスプレイに次の実行開始メッセージを表示します。

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

(2) 実行終了メッセージ

リンクの結果，リンク・エラーが検出されなかった場合，リンカは次のメッセージをディスプレイに出力して制御を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ディスプレイに出力して制御を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ディスプレイに出力してリンクを中止し，制御をOSに戻します。

例1. 存在しないオブジェクト・モジュール・ファイルを指定した場合

```
A>lk78k0s samp1.rel samp2.rel

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

A006 File not found 'SAMP1.REL'
A006 File not found 'SAMP2.REL'
Program Aborted.
```

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

例2. **存在しないリンカ・オプションを指定した場合**

```
A>lk78k0s k0smain.rel k0ssub.rel -z

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

A018 Option is not recognized '-z'
Please enter 'LK78K0S --', if you want help messages.
Program Aborted.
```

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

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

6.6 リンカ・オプション

6.6.1 リンカ・オプションの種類

リンカ・オプションはリンカの動作に細かい指示を与えるものです。リンカ・オプションは，16種類のオプションに分類できます。

表6－5 リンカ・オプション

項番	分　類	オプション	説　明
1	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	
2	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	
3	ディバッグ情報出力指定	-G	ディバグ情報をロード・モジュール・ファイルへ出力します。
		-NG	
4	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	
5	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。
6	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	
7	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	
8	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行コードを付加します。
		-NLF	
9	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	
10	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。
11	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。
12	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。
13	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。
14	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。
15	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。
16	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。

備考 リンカ・オプションの詳細については，C. 3 **リンカ・オプション一覧**を参照してください。

6.6.2 リンカ・オプションの優先度

表6－6に示すリンカ・オプションのうち，縦軸のものと横軸のものを同時に2つ以上指定した場合の優先度について説明します。

表6－6 リンカ・オプションの優先度

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

←横軸

↑
縦軸

【×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 A>lk78k0s k0smain.rel k0ssub.rel -np -km -s

-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 A>lk78k0s k0smain.rel k0ssub.rel -p -nkm -nkp -nkl -s

-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 A>lk78k0s k0smain.rel k0ssub.rel -o -no -s

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

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

6.6.3 リンカ・オプションの説明

各リンカ・オプションの詳細について説明します。

(1) ロード・モジュール・ファイル出力指定 (-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】

① -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。
② -NOオプションは，ロード・モジュール・ファイルを出力しない指定をします。

【用 途】

① 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。
② リンク・リスト・ファイルの出力のみが目的でリンクする場合などに，-NOオプションを指定します。この指定によってリンク時間が短縮されます。

【説 明】

① 出力ファイルとしてディスク型ファイル名とデバイス型ファイル名のNUL, AUXを指定できます。
② -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。
③ -Oオプションを指定する際に‘出力ファイル名’を省略すると，カレント・ディレクトリにロード・モジュール・ファイル‘入力ファイル名.lmf’が出力されます。
④ ‘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。
⑤ -Oと-NOの両オプションを同時に指定した場合は，あとで指定した方を優先します。

【使 用 例】

ロード・モジュール・ファイルk0s.lmfを出力します。

```
A>lk78k0s k0smain.rel k0ssub.rel –ok0s.lmf
```

(2) ロード・モジュール・ファイル強制出力指定 (-J/-NJ)

記述形式 ：-J
：-NJ
省略時解釈：-NJ

【機 能】

① -Jオプションは，フェイタル・エラーの場合でもロード・モジュール・ファイルを出力するように指定します。

② -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】

① -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。

② -Jと-NJの両オプションを同時に指定した場合は，あとで指定した方を優先します。

【使 用 例】

ロード・モジュール・ファイルを強制的に出力します。

```
A>lk78k0s k0smain.rel k0ssub.rel -j
```

(3) ディバグ情報出力指定 (-G/-NG)

記述形式 ：-G
：-NG
省略時解釈：-G

【機 能】

① -Gオプションは，ロード・モジュール・ファイル中にディバグ情報（ローカル・シンボル情報）を付加する指定をします。

② -NGオプションは，-Gオプションを無効にします。

【用 途】

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

【説 明】

① -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。

② -Gオプションを省略するとディバグ情報は付加されません。

③ -Gと-NGの両オプションを同時に指定した場合は，あとで指定した方を優先とします。

④ -NGオプションが指定された場合，-KP, -KLオプションにかかわらず，パブリック・シンボル・リストおよびローカル・シンボル・リストは出力しません。

【使 用 例】

ロード・モジュール・ファイルにディバグ情報を付加します。

```
A>lk78k0s k0smain.rel k0ssub.rel -g
```

（4）スタック解決用シンボル生成指定 （-S/-NS）

記述形式 ：-S[領域名]
：-NS
省略時解釈：-NS

【機 能】

① -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。
② -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】

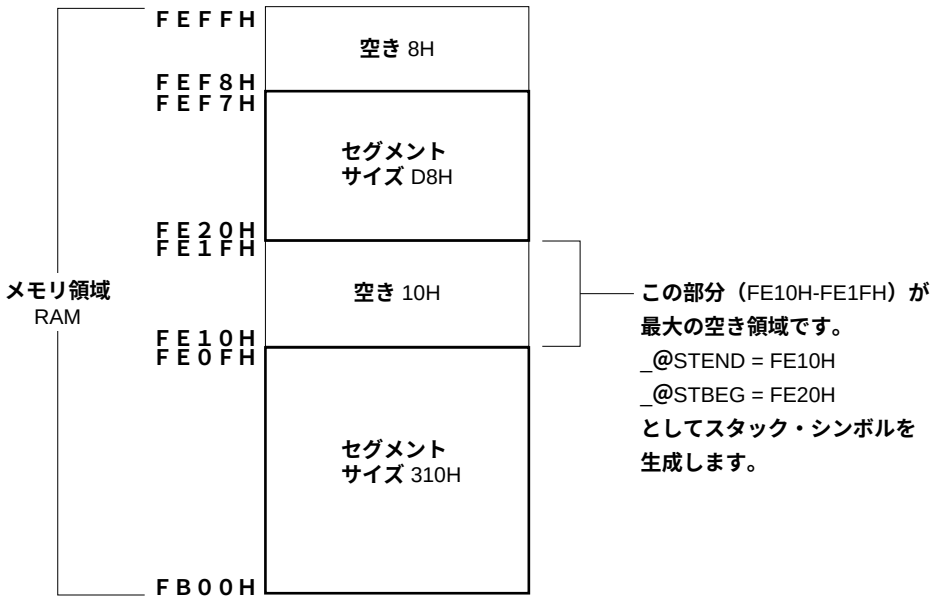
① ‘ 領域名 ’には，ユーザが定義したメモリ領域名，またはディフォールトで定義されているメモリ領域名を指定します。
② ‘ 領域名 ’は，大文字と小文字を区別します。
③ リンカは，-Sオプションで指定したメモリ領域の中で，セグメントが配置されていない領域のうち，最大のアドレスを探します。そして，見つかった最大のアドレス領域の先頭アドレスを値として持つパブリック・シンボル‘ __@STEND ’と，最終アドレス＋1を値として持つパブリック・シンボル‘ __@STBEG ’を生成します。
これらのシンボルは，パブリック宣言されたNUMBER属性のシンボルとして扱われ，リンカのシンボル・テーブル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。
④ 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。
⑤ 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。
⑥ ‘ 領域名 ’が省略された場合，‘ RAM ’が指定されたものとします。
⑦ -Sと-NSの両オプションを同時に指定した場合は，あとで指定した方を優先します。

【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）

```
A>lk78k0s k0smain.rel k0ssub.rel -s
```



(5) ディレクティブ・ファイル指定 (-D)

記述形式 ：-Dファイル名
省略時解釈：なし

【機 能】

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

【用 途】

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

【説 明】

① ‘ファイル名’として指定できるのは，ディスク型ファイル名のみです。デバイス型ファイル名を指定するとアボート・エラーとなります。
② ファイル名を省略するとアボート・エラーとなります。
③ ディレクティブ・ファイルのネストは許されません。
④ ディレクティブ・ファイル中に記述できる文字数の制限はありません。
⑤ -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。
⑥ ディレクティブ・ファイルの詳細については，6. 4 **リンク・ディレクティブ**を参照してください。

【使 用 例】

ディフォールトのメモリ領域ROM/RAMを再定義します。

ディレクティブ・ファイルK0S.DRの内容

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

K0S.DRを使用してリンクします。

```
A>lk78k0s k0smain.rel k0ssub.rel –dk0s.dr –s
```

(6) リンク・リスト・ファイル出力指定 (-P/-NP)

記述形式 ：-P[出力ファイル名]
：-NP
省略時解釈：-P入力ファイル名.MAP

【機 能】

① -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。

② -NPオプションは，-Pオプションを無効にします。

【用 途】

① 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。

② 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】

① ファイル名としてディスク型ファイル名とデバイス型ファイル名を指定できます。
ただし，指定できるデバイス型ファイル名はCON, PRN, NULおよびAUXです。CLOCKを指定した場合，アボート・エラーとなります。

② -Pオプションを指定する際に，‘ 出力ファイル名 ’ を省略すると，カレント・ディレクトリにリンク・リスト・ファイル ‘ 入力ファイル名.MAP ’ を出力します。

③ ‘ 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。

④ -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優先となります。

【使 用 例】

リンク・リスト・ファイル（K0S.MAP）を作成します。

```
A>lk78k0s k0smain.rel k0ssub.rel –pk0s.map –s
```

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

(a) -KM/-NKM

記述形式 ：-KM
：-NKM
省略時解釈：-KM

【機 能】

① -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。
② -NKMオプションは，-KM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】

① -KMと-NKMの両オプションを同時に指定した場合は，あとで指定した方を優先します。
② -NKMオプションを指定した場合，-KDオプションを指定してもリンク・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。
③ -NKM, -NKPおよび-NKLオプションがすべて指定された場合，-Pオプションを指定してもリンク・リスト・ファイルは出力されません。

【使 用 例】

リンク・リスト・ファイルK0S.MAPにマップ・リストを出力します。

```
A>lk78k0s k0smain.rel k0ssub.rel –pk0s.map -km –s
```

K0S.MAPを参照します。

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

Command: k0smain.rel k0ssub.rel -km -pk0s.map -s
Para-file:
Out-file: K0SMAIN.LMF
Map-file: K0S.MAP
Direc-file:
Directive:

*** Link information ***

     3 output segment(s)
   37H byte(s) real data
    25 symbol(s) defined

*** Memory map ***

  SPACE=REGULAR

  MEMORY=ROM
  BASE ADDRESS=0000H  SIZE=4000H
         OUTPUT    INPUT     INPUT     BASE      SIZE
         SEGMENT   SEGMENT   MODULE    ADDRESS
         CODE                          0000H     0002H     CSEG AT
                   CODE      SAMPM     0000H     0002H
* gap *                                0002H     007EH
         ?CSEG                         0080H     0035H     CSEG
                   ?CSEG     SAMPM     0080H     0015H
                   ?CSEG     SAMPS     0095H     0020H
* gap *                                00B5H     3F4BH

  MEMORY=RAM
  BASE ADDRESS=FD00H  SIZE=0300H
         OUTPUT    INPUT     INPUT     BASE      SIZE
         SEGMENT   SEGMENT   MODULE    ADDRESS
* gap *                                FD00H     0120H
         DATA                          FE20H     0003H     DSEG AT
                   DATA      SAMPM     FE20H     0003H
* gap *                                FE23H     00DDH
* gap (Not Free Area) *                FF00H     0100H

  Target chip:uPD789026
  Device file:Vx.xx
```

マップ・リスト

（b）-KD/-NKD

記述形式 ：-KD
：-NKD
省略時解釈：-KD

【機 能】

① -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。
② -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に，-KDオプションを指定します。

【説 明】

① -KDと-NKDの両オプションを同時に指定した場合は，あとで指定した方を優先します。
② -NKMオプションを指定した場合，-KDオプションを指定してもリンク・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。
③ -NKM, -NKPおよび-NKLオプションがすべて指定された場合，-Pオプションを指定してもリンク・リスト・ファイルは出力されません。

【使 用 例】

リンク・リスト・ファイル（K0S.MAP）にリンク・ディレクティブ・ファイルを出力します。

```
A>lk78k0s k0smain.rel k0ssub.rel –dk0s.dr –pk0s.map -kd –s
```

K0S.MAPを参照します。

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

Command:k0smain.rel k0ssub.rel –dk0s.dr –pk0s.map -kd -s
Para-file:
Out-file:K0SMAIN.LMF
Map-file:K0S.MAP
Direc-file:K0S.DR                                  ←ディレクティブ・ファイル名
Directive:memory ROM: (0h, 1000h)                  ←ディレクティブ・ファイルの内容
          memory RAM: (0fe20h, 1e0h)

*** Link information ***

     3 output segment(s)
   37H byte(s) real data
    23 symbol(s) defined

*** Memory map ***

  SPACE = REGULAR

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

（c）-KP/-NKP

記述形式 ：-KP
：-NKP
省略時解釈：-NKP

【機 能】

① -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。
② -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】

① -KPと-NKPの両オプションを同時に指定した場合は，あとで指定した方を優先します。
② -NKM, -NKPおよび-NKLオプションがすべて指定された場合，-Pオプションを指定してもリンク・リスト・ファイルは出力されません。
③ -NGオプションを指定した場合，-KPオプションを指定してもパブリック・シンボル・リストは出力されません。

【使 用 例】

リンク・リスト・ファイル（K0S.MAP）に，パブリック・シンボル・リストを出力します。

```
A>lk78k0s k0smain.rel k0ssub.rel -g –pk0s.map -kp –s
```

K0S.MAPを参照します。

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

Command:k0smain.rel k0ssub.rel -g –pk0s.map -kp -s
Para-file:
Out-file:K0SMAIN.LMF
Map-file:K0S.MAP
Direc-file:
Directive:

*** Link information ***

     3 output segment(s)
   37H byte(s) real data
    23 symbol(s) defined

*** Memory map ***

  SPACE = REGULAR
        ⋮

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

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

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

SAMPM         ADDR      0000H     MAIN
SAMPM         ADDR      0080H     START
SAMPS         ADDR      009AH     CONVAH
              NUM       FE20H     _@STBEG
              NUM       FD00H     _@STEND

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

パブリック・シンボル・リスト

（d）-KL/-NKL

記述形式 ：-KL
：-NKL
省略時解釈：-NKL

【機 能】

① -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。
② -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】

① -KLと-NKLの両オプションを同時に指定した場合は，あとで指定した方を優先します。
② -NKM, -NKPおよび-NKLオプションが，すべて指定された場合，-Pオプションを指定してもリンク・リスト・ファイルは出力されません。
③ -NGオプションを指定した場合は，-KLオプションを指定してもローカル・シンボル・リストは出力されません。

【使 用 例】

リンク・リスト・ファイル（K0S.MAP）に，ローカル・シンボル・リストを出力します。

```
A>lk78k0s k0smain.rel k0ssub.rel –pk0s.map -kl –s
```

K0S.MAPを参照します。

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

Command:k0smain.rel k0ssub.rel -g –pk0s.map -kl -s
Para-file:
Out-file:K0SMAIN.LMF
Map-file:K0S.MAP
Direc-file:
Directive:

*** Link information ***

    3  output segment(s)
  37H  byte(s) real data
   23  symbol(s) defined

*** Memory map ***

  SPACE = REGULAR

       ⋮

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

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

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

SAMPM    MOD                SAMPM
SAMPM    DSEG               DATA
SAMPM    ADDR     FE20H     HDTSA
SAMPM    ADDR     FE21H     STASC
SAMPM    CSEG               CODE
SAMPM    CSEG               ?CSEG
SAMPS    MOD                SAMPS
SAMPS    CSEG               ?CSEG
SAMPS    ADDR     00ACH     SASC
SAMPS    ADDR     00B2H     SASC1

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

ローカル・シンボル・リスト

(8) リンク・リスト・ファイル形式指定 (-LL, -LF/-NLF)

(a) -LL

記述形式 ：-LL[行数]
省略時解釈：-LL66（ディスプレイ出力の場合は改頁しません）

【機 能】

-LLオプションは，リンク・リスト・ファイルの1頁の行数を指定します。

【用 途】

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

【説 明】

① -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。

② 行数が省略された場合，66が指定されたものとみなします。

③ 行数に0を指定した場合は改頁しません。

④ -NPオプションが指定された場合，-LLオプションは無効となります。

【使 用 例】

リンク・リスト・ファイルの1頁の行数を20行に指定します。

```
A>lk78k0s k0smain.rel k0ssub.rel –pk0s.map -ll20 –s
```

K0S.MAPを参照します。

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

Command:k0smain.rel k0ssub.rel –pk0s.map -ll20 –s
Para-file:
Out-file:K0SMAIN.LMF
Map-file:K0S.MAP
Direc-file:
Directive:

*** Link information ***

     3  output segment(s)
   37H  byte(s) real data

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

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

   23 symbol(s) defined

*** Memory map ***

  SPACE = REGULAR

  MEMORY = ROM
  BASE ADDRESS = 0000H    SIZE = 2000H
        OUTPUT      INPUT     INPUT     BASE      SIZE
        SEGMENT     SEGMENT   MODULE    ADDRESS

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

78K/0S Series Linker Vx. xx               Date:xx xxx xxxx Page:  3
        CODE                            0000H     0002H  CSEG AT
                    CODE      SAMPM     0000H     0002H
* gap *                                 0002H     007EH
        ?CSEG                           0080H     0035H  CSEG
                    ?CSEG     SAMPM     0080H     0015H
                    ?CSEG     SAMPS     0095H     0020H
* gap *                                 00B5H     1F4BH

  MEMORY = RAM
  BASE ADDRESS = FE00H    SIZE = 0200H
        OUTPUT      INPUT     INPUT     BASE      SIZE
       ⋮
```

（b）-LF/-NLF

記述形式 ：-LF
：-NLF
省略時解釈：-NLF

【機 能】

① -LFオプションは，リンク・リスト・ファイルの最後に，改頁コード（FF）を付加する指定をします。
② -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】

① -NPオプションを指定した場合，-LFオプションは無効となります。
② -LFと-NLFの両オプションを同時に指定した場合，あとで指定した方を優先します。

【使 用 例】

リンク・リスト・ファイルの最後に改頁コードを付加します。

```
A>lk78k0s k0smain.rel k0ssub.rel –pk0s.map -lf –s
```

(9) エラー・リスト・ファイル出力指定 (-E/-NE)

記述形式 ：-E[ファイル名]
：-NE
省略時解釈：-NE

【機 能】

① -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。

② -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】

① ファイル名として，ディスク型ファイル名とデバイス型ファイル名を指定できます。
ただし，デバイス型ファイル名としてCLOCKを指定した場合は，アボート・エラーとなります。

② -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。

③ -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。

④ -Eと-NEの両オプションを同時に指定した場合は，あとで指定した方を優先します。

【使 用 例】

エラー・リスト・ファイル（K0S.ELK）を作成します。

```
A>lk78k0s k0smain.rel k0ssub.rel –dk0s.dr –ek0s.elk –s
```

ディレクティブ・ファイルの内容に誤りがありました。K0S.ELKを参照します。

```
K0S.DR(3) :F102 Directive syntax error
```

（10）ライブラリ・ファイル指定 （-B）

記述形式 ：-Bファイル名
省略時解釈：なし

【機 能】

-Bオプションは，指定ファイルをライブラリ・ファイルとして入力することを指定します。

【用 途】

リンカは入力モジュールが参照しているモジュールを，ライブラリ・ファイルから探し出し，そのモジュールだけを入力モジュールと結合します。

ライブラリ・ファイルは，あらかじめ複数のモジュールを登録して1つのファイルにまとめておくためのものです。

共通的なモジュールをライブラリ・ファイルとして作成しておけば，ファイル管理の面でも操作性の面においても効率がよくなります。ライブラリ・ファイルをリンカに入力するには，-Bオプションを指定します。

【説 明】

① ファイル名としてディスク型ファイル名以外は指定できません。
② ファイル名は省略できません。
③ ファイル名としてパス名を含めて指定した場合は，指定されたパスからライブラリ・ファイルを入力します。指定されたパスにライブラリ・ファイルが存在しない場合は，エラーとなります。
④ ファイル名としてパス名を含まずに指定した場合には，-Iオプションの指定およびディフォールトのサーチ・パスからライブラリ・ファイルを入力します。
⑤ -Bオプションが複数指定された場合は，指定順にライブラリ・ファイルの入力を行います。-Bオプションは最大10個まで指定できます。
⑥ ライブラリ・ファイルの作成方法の詳細は，**第8章 ライブラリアン**を参照してください。

【使 用 例】

ライブラリ・ファイル（K0S.LIB）を入力します。
（ライブラリ・ファイルには，K0SSUB.RELが登録されています）。

```
A>lk78k0s k0smain.rel –bk0s.lib
```

(11) ライブラリ・ファイル読み込みパス指定 (-l)

記述形式 ：-lパス名[，パス名]…（複数指定可能）
省略時解釈：環境変数‘LIB78K0S’により指定されたパス
指定されていない場合は，カレント・パス

【機 能】

ライブラリ・ファイルを指定したパスから入力するよう指定します。

【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】

① -lオプションは，-Bオプションでパス名を含まないライブラリ・ファイル名が指定された場合にのみ有効です。

② -lは複数指定できます。また，‘，’で区切ることにより，1度に複数のパスを指定できます。この際‘，’の前後には空白を入れることはできません。

③ パス名は，1回のリンクで最大10個まで指定できます。パス名が複数指定された場合リンカは指定順にライブラリ・ファイルのサーチを行います。

④ 指定したパスにライブラリ・ファイルが存在しない場合でもエラーとはなりません。

⑤ パス名が省略された場合，アボート・エラーとなります。

⑥ -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．-lオプションで指定されたパス
2．環境変数‘LIB78K0S’で指定されたパス
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・ファイルをディレクトリA：¥LIBから検索します。

```
A>lk78k0s k0smain.rel k0ssub.rel –bk0s.lib –ia:¥lib –s
```

(12) パラメータ・ファイル指定 (-F)

記述形式 ：-Fファイル名
省略時解釈：コマンド行上からのみオプション，入力ファイル名の入力が可能

【機 能】

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

【用 途】

コマンド行では，リンカの起動に必要な情報を指定しきれないときに，-Fオプションを指定します。リンクするたび繰り返し同じようにオプションを指定する場合には，それらをパラメータ・ファイルに記述しておき，-Fオプションで指定します。

【説 明】

① ‘ファイル名’として指定できるのは，ディスク型ファイル名のみです。デバイス型ファイル名を指定するとアボート・エラーとなります。
② ファイル名を省略するとアボート・エラーとなります。
③ 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。
④ パラメータ・ファイル中に記述できる文字数の制限はありません。
⑤ 空白とタブおよび‘↵’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。
⑥ 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。
⑦ 展開されたオプションは，あとで指定したものが優先です。
⑧ ‘；’以降に記述された文字は‘↵’またはEOFの前まですべてコメントと解釈します。
⑨ -Fオプションを複数指定するとアボート・エラーとなります。

【使 用 例】

パラメータ・ファイルを使用してリンクします。

パラメータ・ファイル（K0S.PLK）の内容

```
;parameter file
k0smain.rel k0ssub.rel –ok0s.lmf –pk0s.map -e -s
-ta:¥tmp -g
```

コマンド行には，次のように入力します。

```
A>lk78k0s –fk0s.plk –s
```

(13) テンポラリ・ファイル作成パス指定 (-T)

記述形式 ：-Tパス名
省略時解釈：環境変数‘TMP’により指定されたパスに作成します。
指定されていない場合は，カレント・パスに作成します。

【機 能】

-Tオプションは，テンポラリ・ファイルを作成するパスを指定します。

【用 途】

テンポラリ・ファイルの作成場所を指定できます。

【説 明】

① パス名として，パス以外のものは指定できません。
② パス名は省略できません。
③ 以前に作成されたテンポラリ・ファイルが存在している場合でも，ファイル保護がされていなければ，上書きします。
④ 必要とするメモリ・サイズがある間は，テンポラリ・ファイルをメモリに展開します。メモリが足りなくなった時点で，メモリに展開していたテンポラリ・ファイルの内容をディスクに書き出します。以降のテンポラリ・ファイルへのアクセスは，セーブしたディスク・ファイルに対して行います。
⑤ テンポラリ・ファイルは，リンク終了時に削除されます。また，キー入力（CTRL-C）によってリンクが中止されたときも，削除されます。
⑥ 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オプションで指定されたパス
2．環境変数TMPに設定されているパス（-Tオプション省略の場合）
3．カレント・パス（TMPが設定されていない場合）

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・ファイルをディレクトリ‘TMP’に作成するように指定します。

```
A>lk78k0s k0smain.rel k0ssub.rel -t¥tmp -s
```

（14）デバイス・ファイル・サーチ・パス指定 （-Y）

記述形式 ：-Yパス名

省略時形式：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 ＜..¥dev＞（lk78k0s.exeの起動されたパスに対して）

② LK78K0Sの起動されたパス

③ カレント・ディレクトリ

④ 環境変数PATH

【機 能】

デバイス・ファイルを指定されたパスから読み込みます。

【用 途】

デバイス・ファイルが存在するパスを指定します。

【説 明】

① -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。

② -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。

③ 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。

a．-Yオプションで指定されたパス

b．＜..¥dev＞（lk78k0s.exeの起動されたパスに対して）

c．LK78K0Sの起動されたパス

d．カレント・ディレクトリ

e．環境変数PATH

【使 用 例】

デバイス・ファイルのパスをa：¥78k0s¥devディレクトリに指定します。

```
A>lk78k0s k0smain.rel k0ssub.rel –ya:¥78k0s¥dev –s
```

（15）ワーニング・メッセージ出力指定 （-W）

記述形式 ：-W[レベル]
省略時形式：通常のエラー・メッセージを出力します。

【機 能】

-Wオプションは，ワーニング・メッセージをコンソールへ出力するか否かを指定します。

【用 途】

ワーニング・メッセージを出力させるレベルを指定できます。

【説 明】

① -W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。
② レベルには，0，1，2以外は指定できません。
③ 出力させるレベルには次のものがあります。
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，表12ー3 **リンカ・エラー・メッセージ**を参照してください。

【使 用 例】

レベルを2として-Wオプションを指定します。

```
A>lk78k0s k0smain.rel k0ssub.rel -w2 -s
```

（16）ヘルプ指定　（--）

記述形式　：--
省略時解釈：表示しない

【機　　能】

--オプションは，ヘルプ・メッセージをディスプレイに表示します。

【用　　途】

ヘルプ・メッセージは，リンカ・オプションとその説明の一覧です。リンカを実行するときに参照してください。

【説　　明】

--オプションを指定すると，他のリンカ・オプションはすべて無効となります。

注意　このオプションは，PM plus**上では指定できません。**
PM plus**上でヘルプを参照する場合は，「リンカオプションの設定」ダイアログでヘルプ・ボタンをクリックしてください。**

【使 用 例】

--オプションを指定するとヘルプ・メッセージがディスプレイに出力されます。

```
   A>lk78k0s --

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

usage:lk78k0s [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                 :Change warning level (n = 0 to 2).
--                 :Show this message.

  Press RETURN to continue ...
DEFAULT ASSIGNMENT :-o -p -ne -km -kd -nkp -nkl -ll66 -nlf -ns -g -nj -w1

directive file usage:
 MEMORY memory-area-name: (origin-value, size) [/memory-space-name]
 MERGE segment-name: [location-type-definition] [merge-type-definition]
      [= memory-area-name] [/memory-space-name]

 example:MEMORY ROM:(0H,4000H)
         MEMORY RAMA:(0H,100H)
         MERGE CSEG1:= ROM
         MERGE DSEG1:AT(80H)
```

6. 7 PM plusでのオプション設定

PM plusからのリンカ･オプションを設定する方法について，説明します。

6. 7. 1 オプションの設定方法

PM plusの [ツール(T)] メニューの [リンカオプションの設定(L)...] を選択または  をクリックすると，<リンカオプションの設定>ダイアログが現れます。

ダイアログ内で必要なオプションを入力することにより，各リンカ･オプションを設定できます

図6－3　<リンカオプションの設定>ダイアログ（《出力1》タブ選択時）

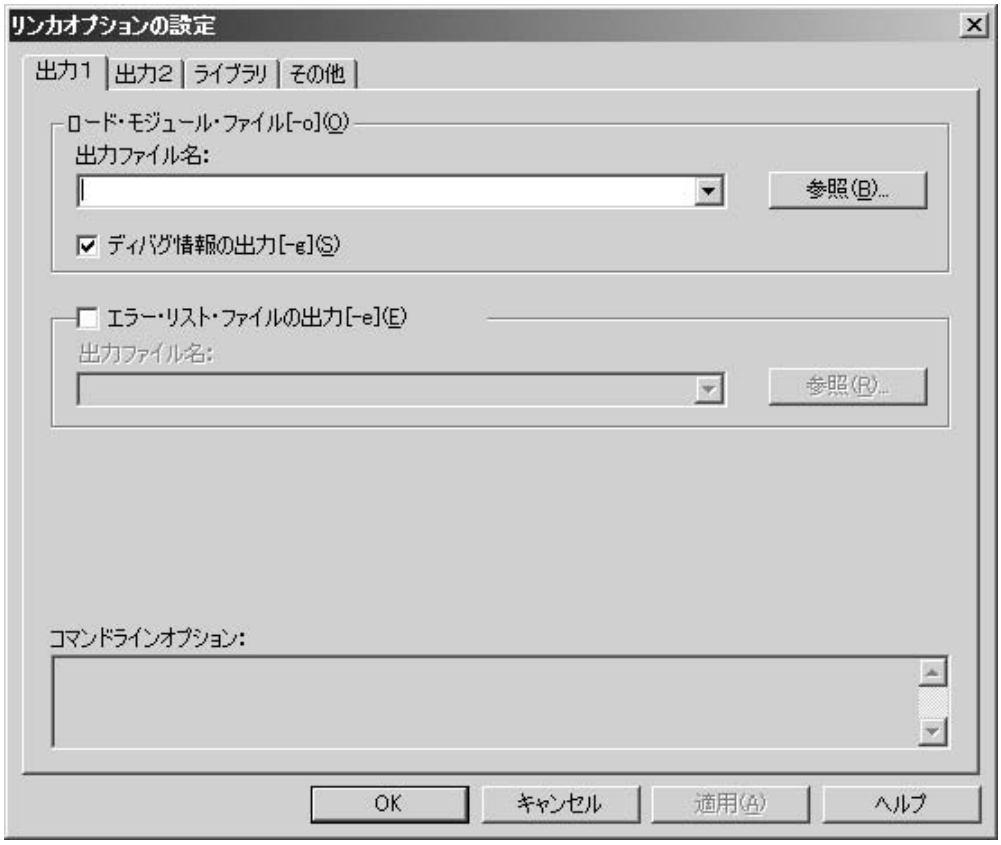


図6－4　<リンカオプションの設定>ダイアログ（《出力2》タブ選択時）

図6－5　<リンカオプションの設定>ダイアログ（《ライブラリ》タブ選択時）

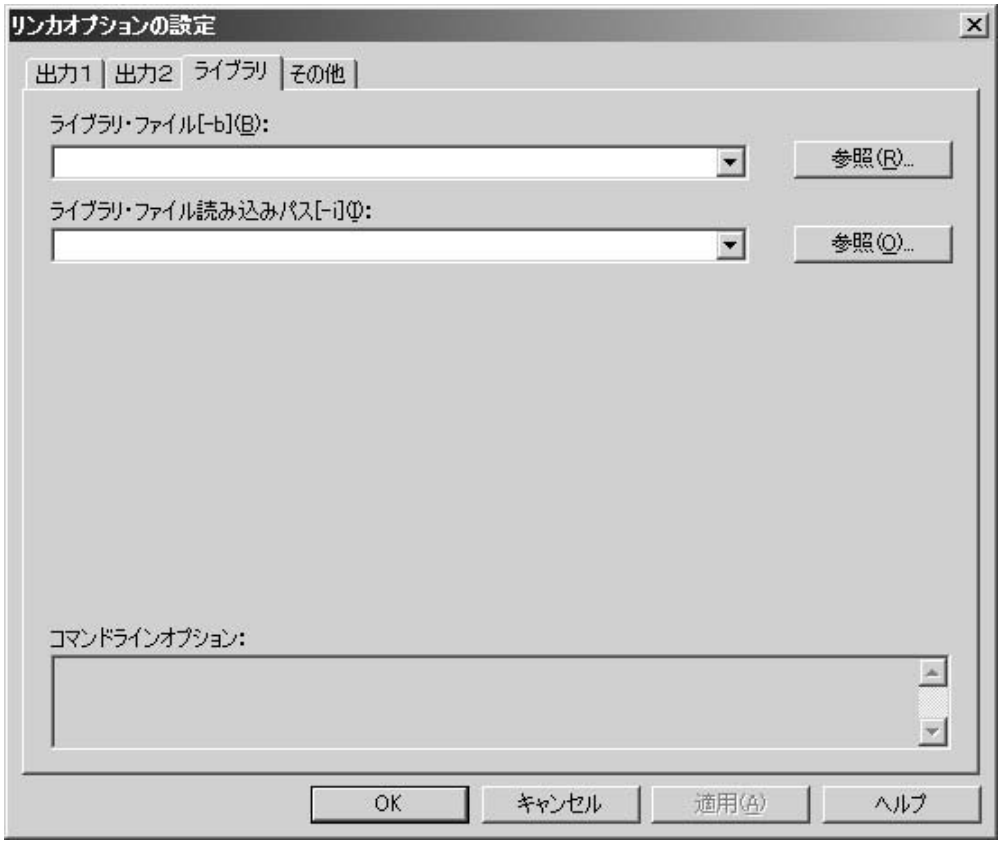
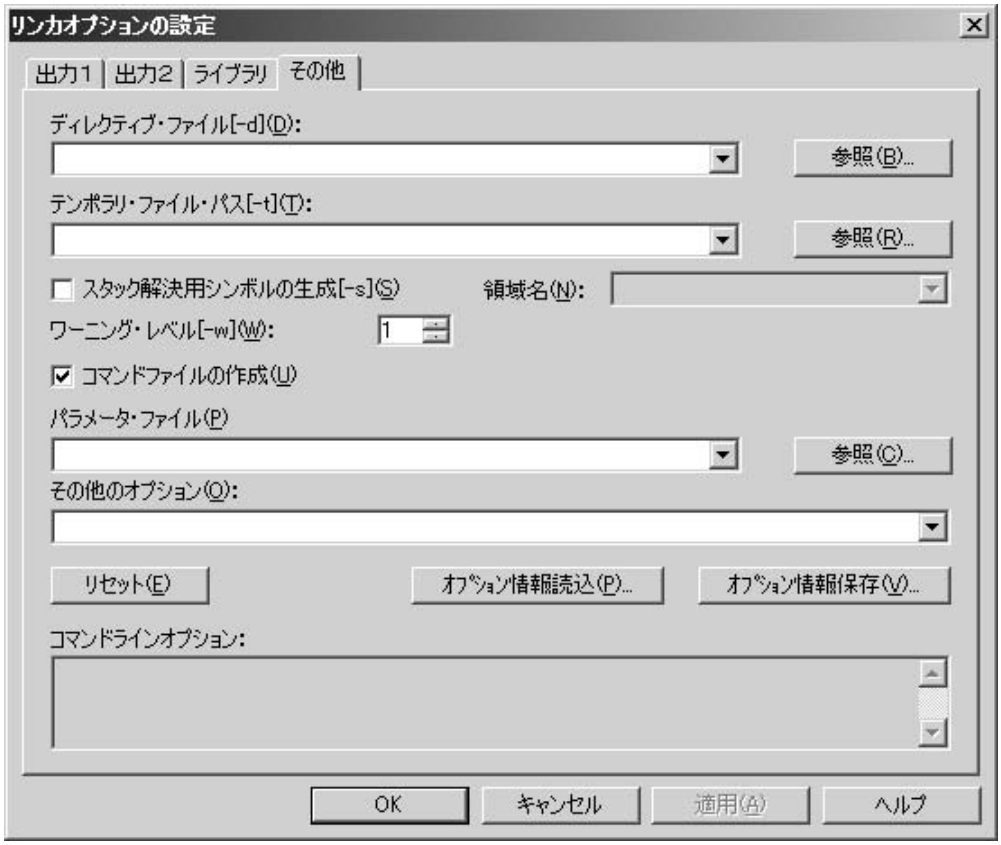


図6－6　<リンカオプションの設定>ダイアログ（《その他》タブ選択時）



6.7.2 各オプションの設定

<リンカオプションの設定>ダイアログの各オプションについて，次に説明します。

- 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･ファイルの出力パスを指定します。
- ディバグ情報の出力[-g](S)

  ロード･モジュール･ファイル中にディバグ情報(ローカル･シンボル情報)を付加します。
- エラー･リスト･ファイルの出力[-e](E)

  出力パス名：

  エラー･リスト･ファイルを出力したい場合は，入力ボックスにファイル名を入力します。

  パスを指定する場合は参照(R)ボタンを使用します。
- 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，チェックします。

  参照(B)ボタンまたは直接入力により，リンク･リスト･ファイルの出力パスを指定します。
- マップ･リストの出力[-km](M)

  リンク･リスト･ファイル中にマップ･ファイルを出力します。
- 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･ファイルの内容を印字したあとで改ページ･コードを付加します。
- 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(20～32767文字まで選択可能)。
- ライブラリ･ファイル[-b](B)

  参照(R)ボタンまたは直接入力により，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。
- ライブラリ･ファイル読み込みパス[-i](I)

  参照(O)ボタンまたは直接入力により，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。
- 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，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。
- 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)

  チェックすると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します。
- 領域名(N)

  利用者が定義したメモリ領域名，またはディフォールトで定義されているメモリ領域名を指定します。

・ワーニング･レベル[-w](W)

ワーニング･メッセージを出力させるレベルを指定します。

0:ワーニング･メッセージを出力しません。

1:通常のワーニング･メッセージを出力します。

2:詳細なワーニング･メッセージを出力します。

・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チェックしてください。

・パラメータ･ファイル(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･ファイルを読み込みます。

・その他のオプション(O)

チェック･ボックスやラジオ･ボタンで選択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。

・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(P)

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

・オプション情報保存(V)

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

・コマンドラインオプション

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

第7章　オブジェクト・コンバータ

オブジェクト・コンバータは，RA78K0Sのリンカが出力したロード・モジュール・ファイル（すべての参照アドレス情報が解決されていなければなりません）を入力し，それをHEX形式オブジェクト・モジュール・ファイルとして出力します。

さらに，シンボリック・ディバグ時に使用するシンボル情報をシンボル・テーブル・ファイルとして出力します。

オブジェクト・コンバータ・エラーがある場合は，エラー・メッセージをディスプレイに出力し，エラーの原因を明示します。

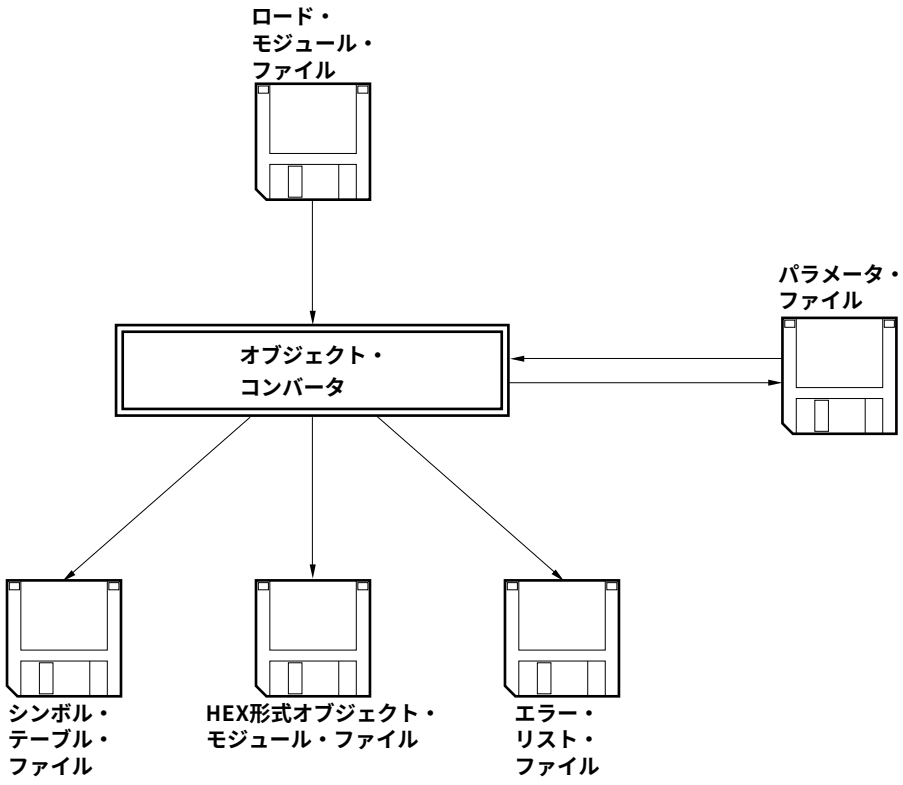
7. 1　オブジェクト・コンバータの入出力ファイル

オブジェクト・コンバータの入出力ファイルを次に示します。

表7－1　オブジェクト・コンバータの入出力ファイル

種　類	ファイル名	説　　明	ディフォールト・ファイル・タイプ
入力ファイル	ロード・モジュール・ファイル	○リンクの結果のオブジェクト・コードのバイナリ・イメージ・ファイルです。 ○リンカの出力したファイルです。	.LMF
	パラメータ・ファイル	○実行プログラムのパラメータを内容とするファイルです。 ○ユーザ作成ファイルです。	.POC
出力ファイル	HEX形式オブジェクト・モジュール・ファイル	○ロード・モジュール・ファイルを，HEX形式オブジェクト・フォーマットで変換したファイルです。 ○マスクROM発注時またはPROMプログラマ使用時に使います。	.HEX
	シンボル・テーブル・ファイル	○入力ファイルの各モジュールに含まれるシンボルの情報を持つファイルです。	.SYM
	エラー・リスト・ファイル	○オブジェクト・コンバート時のエラー情報を持つファイルです。	.EOC

図7－1　オブジェクト・コンバータの入出力ファイル



7. 2　オブジェクト・コンバータの機能

(1) HEX形式オブジェクト・モジュール・ファイル

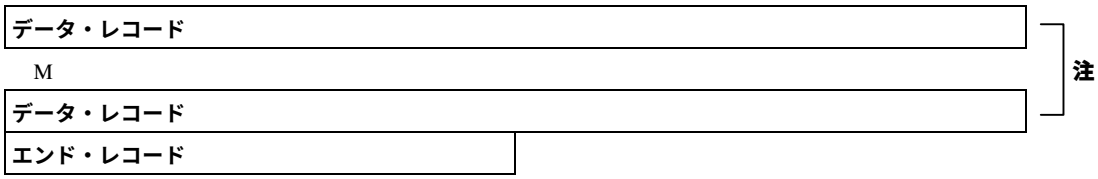
オブジェクト・コンバータの出力するHEX形式オブジェクト・モジュール・ファイルは，PROMプログラマやディバッガなどのHEXローダに入力可能です。

次にサンプル・プログラムのHEX形式オブジェクト・モジュール・ファイルを示します。

```
:0200000080007E
:10008000F5201AFC20FE229500F821FE0A27EB88B5
:100090000A25EB30FE2F00000000630F22AC000A9F
:1000A000E70A430F630F22AC000AE520130A380267
:0500B0008307833020EE
:00000001FF
```

【インテル標準HEX形式オブジェクト・モジュール・ファイルのフォーマット】

図7－2　インテル標準形式



注　データ・レコードは繰り返されます。

(a) データ・レコード

：	XX	XXXX	00	DD…DD	SS
①	②	③	④	⑤	⑥

① レコード・マーク

レコードの始まりを示します。

② 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。

③ 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。

④ レコード・タイプ（2桁）

00で固定します。

⑤ 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。

⑥ 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
|  |   |    |   |
①  ②   ③    ④   ⑤
```

① **レコード・マーク**

② **コード数。**00**で固定です。**

③ 0000**で固定です。**

④ **レコード・タイプ。**01**で固定です。**

⑤ **チェック・サム。**FF**で固定です。**

(2) シンボル・テーブル・ファイル

オブジェクト・コンバータの出力するシンボル・テーブル・ファイルは，ディバッガへの入力となります。

次にサンプル・プログラムのシンボル・テーブル・ファイルを示します。

```
#04
;FF    PUBLIC
010095CONVAH
010000MAIN
010080START
;FF    SAMPM
<02FE20HDTSA
02FE21STASC
;FF    SAMPS
<0100ACSASC
0100B2SASC1
=
```

【シンボル・テーブル・ファイルのフォーマット】

シンボル・テーブルの開始	#	04		CR	LF			
パブリック・シンボルの開始	;	FF		空白4個		PUBLIC	CR	LF
	注1→	シンボル属性		シンボル値		パブリック・シンボル名	CR	LF
		M		M		M	M	M
	;	FF		空白4個		モジュール名1	CR	LF
ローカル・シンボルの開始	<	シンボル属性		シンボル値[注2]		ローカル・シンボル名	CR	LF
		シンボル属性		シンボル値		ローカル・シンボル名	CR	LF
		M		M		M	M	M
	;	FF		空白4個		モジュール名2	CR	LF
オブジェクト・モジュール単位で繰り返します。		M		M		M	M	M
シンボル・テーブル終了のマーク	=	CR	LF					

パブリック・シンボル

モジュールごとのローカル・シンボル

注1.　シンボル属性は，**表**7－2　**シンボル属性の値**の値をとります。

2.　シンボル値は**図**7－3　**シンボル値のフォーマット**を参照してください。

表7－2　**シンボル属性の値**

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

図7－3　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ADDRESS**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

0　n	**アドレス値**　4**桁**

↑

n**：ビット位置（**0-7**）**

7.3　オブジェクト・コンバータの起動方法

7.3.1　オブジェクト・コンバータの起動

オブジェクト・コンバータの起動には，次の2つの方法があります。

（1）コマンド行での起動

```
X> [パス名] oc78k0s [△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・ディレクトリ名**
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**
⑤ **コンバートするロード・モジュール・ファイル名**

例　A>oc78k0s k0s.lmf -osamle.hex

注意　複数のオブジェクト・コンバータ・オプションを指定する場合は，それぞれのオブジェクト・コンバータ・オプション間を空白で区切ってください。オブジェクト・コンバータ・オプションの詳細については，7.4　**オブジェクト・コンバータ・オプションを参照してください。**

(2) パラメータ・ファイルによる起動

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

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

パラメータ・ファイルによる起動方法を次に示します。

```
X>oc78k0s [△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
```

① パラメータ・ファイル指定オプション

② オブジェクト・コンバータの起動に必要な情報を含んだファイル

備考　パラメータ・ファイルはエディタなどで作成します。

パラメータ・ファイル内での記述規則を次に示します。

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

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

2. ロード・モジュール・ファイル名は，オプションのあとでも記述できます。

3. パラメータ・ファイルには，コマンド行で指定すべきすべてのオブジェクト・コンバータ・オプション，出力ファイル名を記述します。

例　パラメータ・ファイル（K0S.POC）をエディタで作成します。

K0S.POCの内容

```
;parameter file
k0s.lmf -osample.hex
-ssample.sym -r
```

パラメータ・ファイル（K0S.POC）を使用してオブジェクト・コンバータを起動します。

```
A>oc78k0s –fk0s.poc
```

7. 3. 2　実行開始，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，ディスプレイに次の実行開始メッセージを表示します。

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

（2）実行終了メッセージ

オブジェクト・コンバートの結果，エラーが検出されなかった場合には，オブジェクト・コンバータは，次のメッセージをディスプレイに出力して制御をOSに戻します。

```
Target chip:uPD789xxx
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ディスプレイに出力して制御をOSに戻します。

```
Target chip:uPD789xxx
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ディスプレイに出力してオブジェクト・コンバートを中止し，制御をOSに戻します。

例1.　存在しないロード・モジュール・ファイル名を指定した場合

```
A>oc78k0s sample.lmf

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

A006 File not found 'SAMPLE.LMF'
Program aborted.
```

この例では，存在しないロード・モジュール・ファイルを指定したためにエラーとなり，オブジェクト・コンバートが中止されました。

例2.　**存在しないオブジェクト・コンバータ・オプションを指定した場合**

```
A>oc78k0s k0s.lmf -a

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

A018 Option is not recognized '-a'
Please enter 'OC78K0S --', if you want help messages.
Program aborted.
```

この例では，存在しないオブジェクト・コンバータ・オプションを指定したために，エラーとなり，オブジェクト・コンバートが中止されました。

オブジェクト・コンバータがエラー・メッセージを出力して，オブジェクト・コンバートを中止した場合，そのエラー・メッセージの原因を第12**章　エラー・メッセージで調べて対処してください。**

7. 4　オブジェクト・コンバータ・オプション

7. 4. 1　オブジェクト・コンバータ・オプションの種類

オブジェクト・コンバータ・オプションは，オブジェクト・コンバータの動作に細かい指示を与えるものです。オブジェクト・コンバータ・オプションは，8種類のオプションに分類できます。

次にオブジェクト・コンバータ・オプションの分類と説明を示します。

表7－3　オブジェクト・コンバータ・オプション

項番	分　類	オプション	説　明
1	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	
2	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	
3	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	
4	オブジェクト充てん指定指定	-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。
5	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	
6	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。
7	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。
8	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。

備考　オブジェクト・コンバータ・オプションの詳細については，C. 4　**オブジェクト・コンバータ・オプション一覧**を参照してください。

7. 4. 2　オブジェクト・コンバータ・オプションの説明

各オブジェクト・コンバータ・オプションの詳細について説明します。

（1）HEX形式オブジェクト・モジュール・ファイル出力指定　（-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】

① -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。

② -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】

① 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。

② 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】

① ‘ 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。

② -Oオプションを指定する際に‘ 出力ファイル名 ’を省略すると，カレント・ディレクトリにHEX形式オブジェクト・モジュール・ファイル（‘ 入力ファイル名.HEX ’）が出力されます。

③ ‘ 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。

④ -Oと-NOの両オプションを同時に指定した場合は，あとで指定した方を優先します。

【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。

```
A>oc78k0s k0s.lmf -osample.hex
```

(2) シンボル・テーブル・ファイル出力指定　(-S/-NS)

記述形式　：-S［出力ファイル名］
　　：-NS
省略時解釈：-S入力ファイル名.SYM

【機　　能】

① -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。

② -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】

① 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。

② 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】

① ‘ 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。

② -Sオプションを指定する際に ‘ 出力ファイル名 ’ を省略すると，カレント・ディレクトリにシンボル・テーブル・ファイル（ ‘ 入力ファイル名.SYM ’ ）が出力されます。

③ ‘ 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。

④ -Sと-NSの両オプションを同時に指定した場合は，あとで指定した方を優先します。

【使 用 例】

シンボル・テーブル・ファイル（SAMPLE.SYM）を出力します。

```
A>oc78k0s k0s.lmf -ssample.sym
```

(3) オブジェクト・アドレス順ソート指定　(-R/-NR)

記述形式　：-R
：-NR
省略時解釈：-NR

【機　　能】

① -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。**

② -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ている必要のある場合に，**-R**オプションを指定します。**

【説　　明】

① -R**と**-NR**の両オプションを同時に指定した場合は，あとで指定した方を優先します。**

② -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。**

```
A>oc78k0s k0s.lmf -r
```

(4) オブジェクト充てん指定　(-U)

記述形式　：-U充てん値［,［スタート］, サイズ］
省略時解釈：なし

【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。

【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】

① 充てん値として指定できる値の範囲は，次のとおりです。

0H　≦　充てん値　≦　0FFH

2進，8進，10進数字または16進数字で指定します。範囲外の数値または数値以外を指定した場合は，アボート・エラーとなります。

② スタートには，充てんを行うアドレス領域の先頭アドレスを指定します。
指定できる値の範囲は，次のとおりです。

0H　≦　スタート　≦　SFR領域を除くプログラム空間の最大アドレス

2進，8進，10進数字または16進数字で指定します。範囲外の数値または数値以外を指定した場合には，アボート・エラーとなります。また，スタートを省略した場合には，0を指定したものとみなします。

③ サイズには，充てんを行うアドレス領域のサイズを指定します。指定できる値の範囲は，次のとおりです。

1H　≦　サイズ　≦　SFR領域を除くプログラム空間の最大アドレス

2進，8進，10進数字または16進数字で指定します。範囲外の数値または数値以外を指定した場合には，アボート・エラーとなります。また，スタートを指定している場合には，サイズを省略できません。

④ スタートとサイズの両方の指定を省略した場合，オブジェクト・コンバータは次の処理を行います。

(a) アセンブルの対象デバイスが，内蔵ROMを持っている場合には，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ます。
(b) アセンブルの対象デバイスが，内蔵ROMを持たない場合，オブジェクト・コンバータはエラーとみなして実行を中止します。

⑤ -Uオプションを複数指定した場合は，最後に指定したものを優先します。
⑥ -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。
⑦ -Uオプションでの，スタート，サイズの指定形式とその解釈は次のようになります。

(a) -U充てん値

：対象デバイスに内蔵ROMがある場合は，内蔵ROMの範囲

：対象デバイスに内蔵ROMのない場合は，アボート・エラー

(b) -U充てん値，サイズ

：0番地からサイズ番地まで

(c) -U充てん値，スタート，サイズ

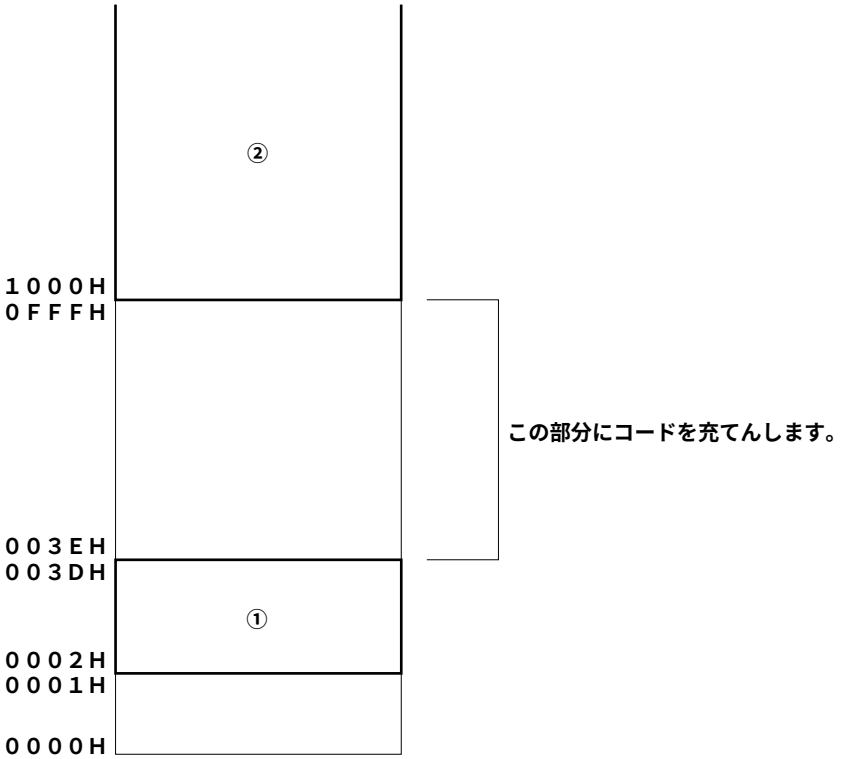
：スタート番地からサイズ番地まで

【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 ┐
:100012001A6720FE2822006521FED350D25014FE1A │ ①
:10002200B900059F2835002431B900059F28350005 ┘
:0C003200242156AF0A8302A807A830560C
:01000003B5D0d0026A3…                       ┐ ②
:1010100024A5F622B667…                      ┘
     ⋮
:00000001FF
```



003EH-0FFFHのアドレス領域に，00Hを充てんします。

```
A>oc78k0s k0s.lmf -u00h,003eh,0fc2h
```

(5) エラー・リスト・ファイル出力指定　(-E/-NE)

記述形式　：-E[出力ファイル名]
　　：-NE
省略時解釈：-NE

【機　　能】

① -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。
② -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】

① ファイル名として，ディスク型ファイル名とデバイス型ファイル名を指定できます。ただし，デバイス型ファイル名としてCLOCKを指定した場合は，アボート・エラーとなります。
② -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。
③ -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。
④ -Eと-NEの両オプションを同時に指定した場合は，あとで指定した方を優先します。

【使 用 例】

エラー・リスト・ファイル（K0S.EOC）を作成します。

```
A>oc78k0 k0s.lmf –ek0s.eoc
```

(6) パラメータ・ファイル指定　(-F)

記述形式　：-Fファイル名
省略時解釈：コマンド行上からのみオプションまたは入力ファイル名の入力が可能

【機　　能】

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

【用　　途】

① 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。
② 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】

① ‘ ファイル名 ’ として指定できるのは，ディスク型ファイル名のみです。
デバイス型ファイル名を指定するとアボート・エラーとなります。
② ファイル名を省略するとアボート・エラーとなります。
③ 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。
④ パラメータ・ファイル中に記述できる文字数は制限はありません。
⑤ 空白とタブおよび ‘ ↵ ’ 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。
⑥ 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。
⑦ 展開されたオプションは，あとで指定したものを優先します。
⑧ ‘ ; ’ または ‘ # ’ 以降に記述された文字は ‘ ↵ ’ またはEOFの前まですべてコメントと解釈します。
⑨ -Fオプションを複数指定するとアボート・エラーとなります。

【使 用 例】

パラメータ・ファイルを使用してオブジェクト・コンバートします。

パラメータ・ファイルK0S.POCの内容

```
;parameter file
k0s.lmf -osample.hex
-ssample.sym -r
```

コマンド行には，次のように入力します。

```
A>oc78k0s k0s.lmf -fk0s.poc
```

（7）デバイス・ファイル・サーチ・パス指定　（-Y）

記述形式　：-Yパス名

省略時形式：デバイス・ファイルを読み込むバスは，次の順序で調べ決定します。

① ＜..¥dev＞（oc78k0s. exeの起動されたパスに対して）

② OC78K0Sの起動されたパス

③ カレント・ディレクトリ

④ 環境変数PATH

【機　　能】

デバイス・ファイルを指定されたパスから読み込みます。

【用　　途】

デバイス・ファイルが存在するパスを指定します。

【説　　明】

① -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。

② -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。

③ 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。

a．-Yオプションで指定されたパス

b．＜..¥dev＞（oc78k0s.exeの起動されたパスに対して）

c．OC78K0Sの起動されたパス

d．カレント・ディレクトリ

e．環境変数PATH

【使 用 例】

デバイス・ファイルのパスをa:¥78k0s¥devディレクトリに指定します。

```
A>oc78k0s k0s.lmf -ya:¥78k0s¥dev
```

(8) ヘルプ指定　(--)

記述形式　：--
省略時解釈：表示しない

【機　　能】

--オプションは，ヘルプ・メッセージをディスプレイに表示します。

【用　　途】

ヘルプ・メッセージは，オブジェクト・コンバータ・オプションとその説明の一覧です。
オブジェクト・コンバータを実行するときに参照してください。

【説　　明】

--オプションを指定すると，他のオブジェクト・コンバータ・オプションは，すべて無効となります。

【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。

```
A>oc78k0s --

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

usage : oc78k0s [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] :Fill up HEX object with specified value.
-ydirectory    :Set device file search path.
--             :Show this message.
DEFAULT ASSIGNMENT: -o -s -nr
```

7. 5　PM plusでのオプション設定

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

7. 5. 1　オプションの設定方法

PM plusの［ツール(T)］メニューの［オブジェクトコンバータオプションの設定　(O)...］を選択または  をクリックすると，<オブジェクトコンバータオプションの設定>ダイアログが現れます。

ダイアログ内で必要なオプションを入力することにより，各オブジェクトコンバータ･オプションを設定できます。

図7－4　<オブジェクトコンバータオプションの設定>ダイアログ（《出力1》タブ選択時）

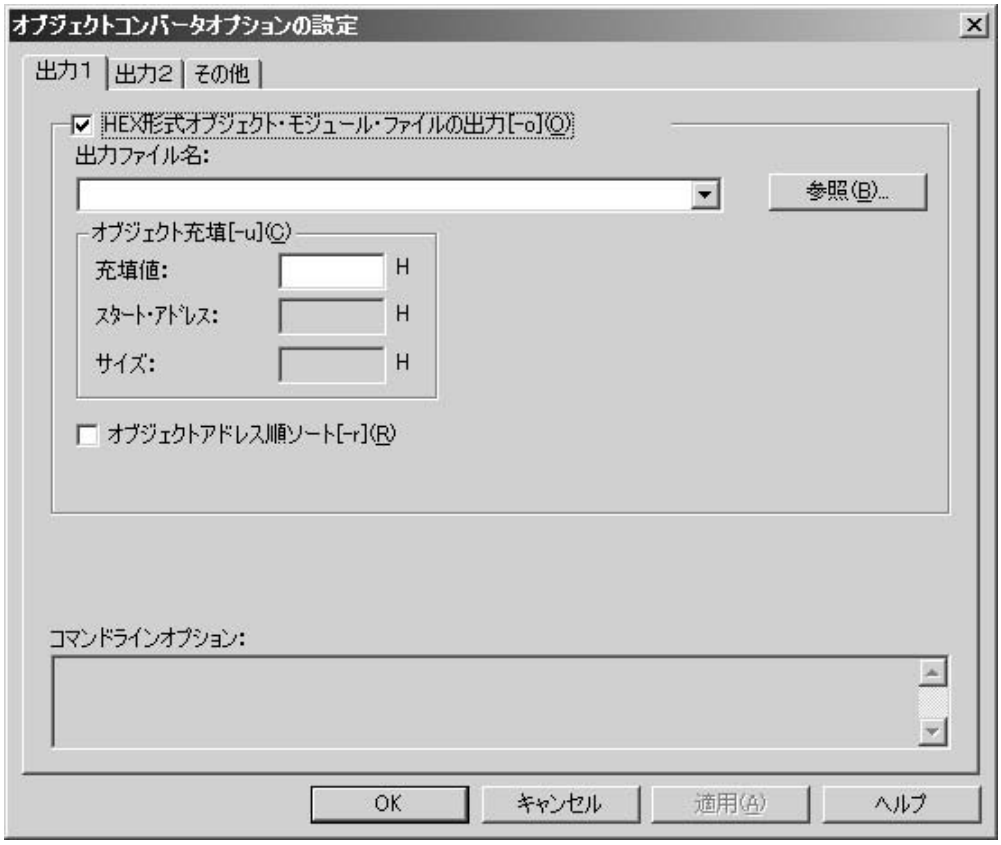


図7－5　<オブジェクトコンバータオプションの設定>ダイアログ（《出力2》タブ選択時）

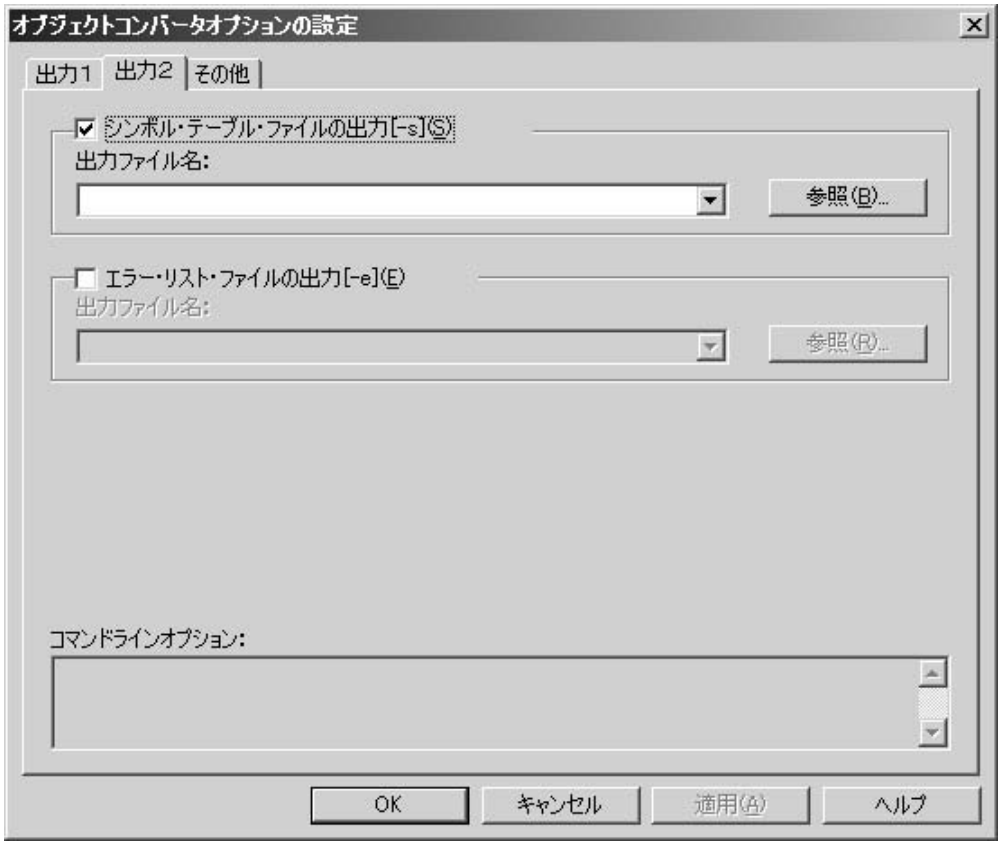
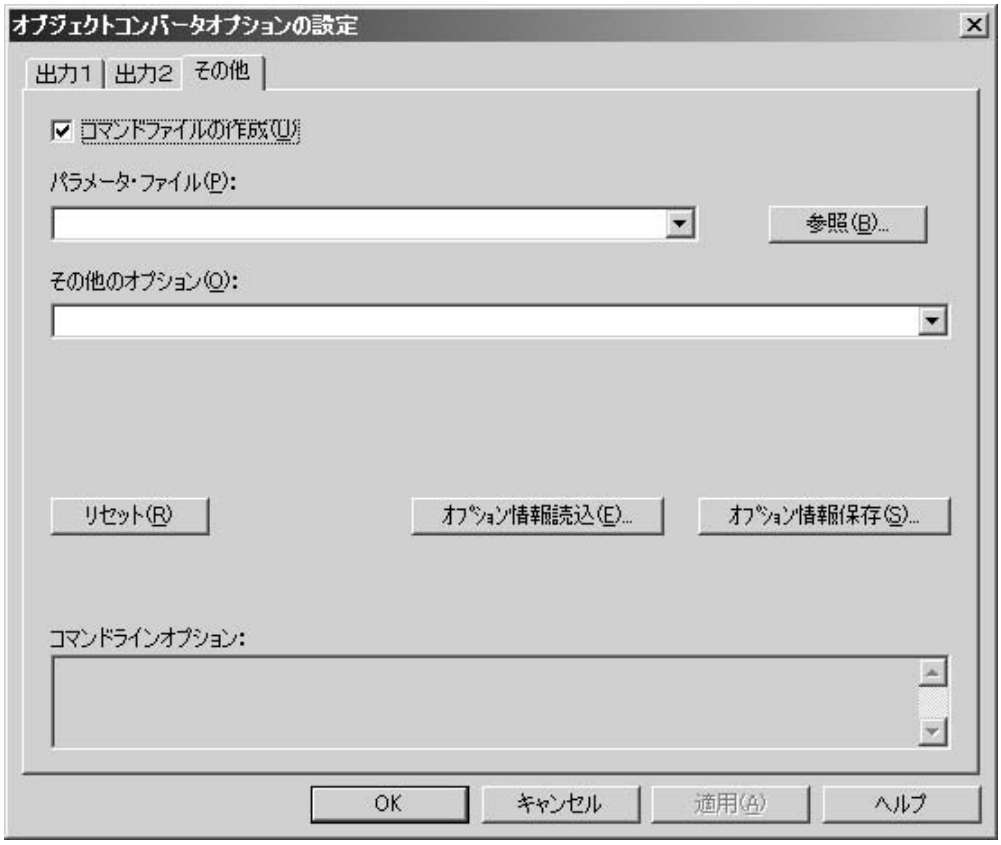


図7－6　<オブジェクトコンバータオプションの設定>ダイアログ（《その他》タブ選択時）



7. 5. 2　各オプションの設定

<オブジェクトコンバータオプションの設定>ダイアログの各オプションについて，次に説明します。

- HEX形式オブジェクト･モジュール･ファイルの出力[-o](O)

  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･ファイルの出力パスを指定します。
- オブジェクト充填[-u](C)

  HEX形式オブジェクトが出力されていないアドレスに不要なコードが書き込まれプログラムが暴走するのを防ぐためにあらかじめコードを書き込んでおきます。
- オブジェクトアドレス順ソート[-r](R)

  HEX形式オブジェクトがアドレス順にソートされている必要がある場合に指定します。
- シンボル･テーブル･ファイルの出力[-s](S)

  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･ファイルの出力パスを指定します。
- エラー･リスト･ファイルの出力[-e](E)

  出力パス名：

  エラー･リスト･ファイルを出力したい場合は，入力ボックスにファイル名を入力します。

  パスを指定する場合は参照(R)ボタンを使用します。
- 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チェックしてください。
- パラメータ･ファイル(P)

  参照(B)ボタンまたは直接入力によりユーザ定義のパラメータ･ファイルを読み込みます。
- その他のオプション(O)

  チェック･ボックスやラジオ･ボタンで選択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。
- リセット(R)

  入力した内容をリセットします。
- オプション情報読込(E)

  <オプション情報読込み>ダイアログが開き，オプション情報ファイルを指定後，読み込みます。
- オプション情報保存(S)

  <オプション情報の保存>ダイアログが開き，オプション情報ファイルに名前をつけて保存します。
- コマンドラインオプション

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

第8章　ライブラリアン

ライブラリアンは，RA78K0Sのオブジェクト・モジュール・ファイルまたはライブラリ・ファイルに対してモジュール単位で編集を行います。

さらに，リスト・ファイルを出力します。

ライブラリアン・エラーがある場合は，エラー・メッセージをディスプレイに出力し，エラーの原因を明示します。

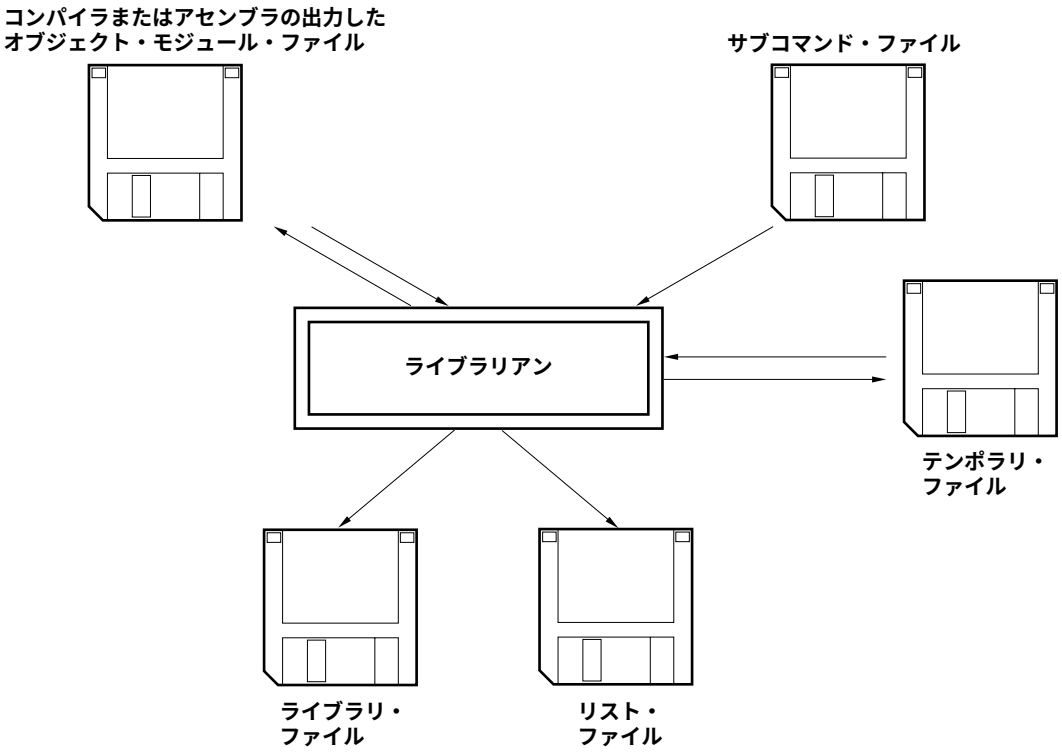
8. 1　ライブラリアンの入出力ファイル

ライブラリアンの入出力ファイルを次に示します。

表8－1　ライブラリアンの入出力ファイル

種　類	ファイル名	説　　明	ディフォールト・ファイル・タイプ
入力ファイル	サブコマンド・ファイル	○実行プログラムのコマンドとパラメータを内容とするファイルです。 ○ユーザ作成ファイルです。	なし
出力ファイル	リスト・ファイル	○ライブラリ・ファイルの情報を出力した結果のファイルです。	.LST
入出力ファイル	オブジェクト・モジュール・ファイル	○アセンブラまたはコンパイラの出力したオブジェクト・モジュール・ファイルです。	.REL
	ライブラリ・ファイル	○ライブラリアンが出力したライブラリ・ファイルを入力し，内容を更新します。	.LIB
	テンポラリ・ファイル	○ライブラリ化のためにライブラリアンが自動生成するファイルです。ライブラリアンの実行終了時には消去されます。	Lbxxxxxx.$$y （y = 1-6）

図8－1　ライブラリアンの入出力ファイル



8.2　ライブラリアンの機能

(1) モジュールのライブラリ化

アセンブラおよびリンカは，1個の出力モジュールを1個のファイルに作成します。

したがって，モジュールの数が多い場合，ファイルの数も増加します。このため，複数のモジュールを1個のファイルにまとめる機能を用意しました。これをモジュールのライブラリ化と呼び，ライブラリ化されたファイルをライブラリ・ファイルと呼びます。

ライブラリ・ファイルは，リンカに入力できます。したがって，モジュラ・プログラミングを行った場合，共通的なモジュールをライブラリ・ファイルとして作成しておけば，ファイル管理の面でも操作性の面においても効率がよくなります。

(2) ライブラリ・ファイルに対する編集

ライブラリアンには，ライブラリ・ファイルに対して次に示す編集機能があります。

① ライブラリ・ファイルへのモジュールの追加
② ライブラリ・ファイル内のモジュールの削除
③ ライブラリ・ファイル内のモジュールの置換
④ ライブラリ・ファイル内のモジュールの抽出

（機能の詳細については，8.5　**サブコマンド**をお読みください）。

(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。

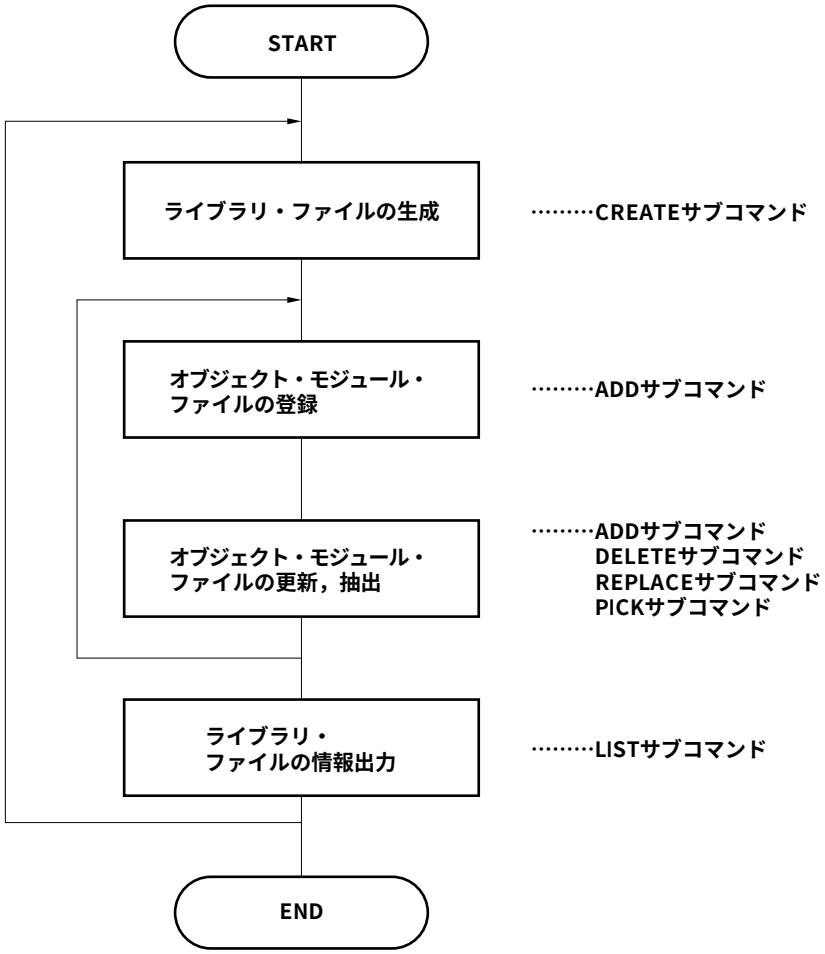
① モジュール名
② 作成プログラム
③ 登録日付
④ 更新日付
⑤ PUBLICシンボル情報

注意　ライブラリアンでは，(2)，(3) で述べ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決しながら処理を行います。サブコマンドの操作については，8.5　**サブコマンドをお読みください。**

(4) ライブラリ・ファイルの作成手順

一般的なライブラリ・ファイルの作成手順を次に示します。

図8－2　ライブラリ・ファイルの作成手順



8.3　ライブラリアンの起動方法

8.3.1　ライブラリアンの起動

ライブラリアンの起動には，次の2つの方法があります。

(1) コマンド行での起動

```
X> [パス名] lb78k0s [△オプション] …
|     |       |         |
①     ②       ③         ④
```

① カレント・ドライブ名
② カレント・ディレクトリ名
③ ライブラリアンのコマンド・ファイル名
④ ライブラリアンに対して動作の詳細を指示します。

例　A>lb78k0s -ll20 -lw80

注意　複数のライブラリアン・オプションを指定する場合は，それぞれのライブラリアン・オプション間を空白で区切ります。
ライブラリアン・オプションの詳細については，8.4　ライブラリアン・オプションを参照してください。

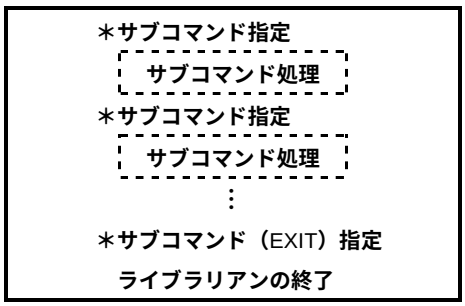
ライブラリアンが起動すると，ディスプレイに以下の起動メッセージが表示されます。

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

‘＊’に続いて，ライブラリアンのサブコマンドを指定します。

```
*create k0s.lib
*add k0s.lib k0smain.rel k0ssub.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‘＊’がディスプレイに出力され，次のサブコマンドの入力待ち状態となります。終了サブコマンド（EXIT）が入力されるまで，この動作は繰り返されます。



1行で指定可能な文字数は，128文字です。

1行に必要なオペランド情報をすべて入力できない場合は，‘＆’を用いて継続行の指定ができます。なお，継続できる行数は15行です。

(2) サブコマンド・ファイルによる起動

サブコマンド・ファイルとはライブラリアンへのコマンドを格納しているファイルです。

ライブラリアンの起動時にサブコマンド・ファイルを指定しない場合，‘＊’のあとに複数のサブコマンドを入力しなければなりません。しかし，サブコマンド・ファイルを作成することにより，これら複数のサブコマンドの処理が一度にできます。

また，ライブラリ化のたびに同じサブコマンドを繰り返し指定することがあります。このような場合にサブコマンド・ファイルを使用してライブラリ化を行います。

サブコマンド・ファイルを使用する場合には，ファイル名の前に‘＜’を記述します。

サブコマンド・ファイルによる起動方法を次に示します。

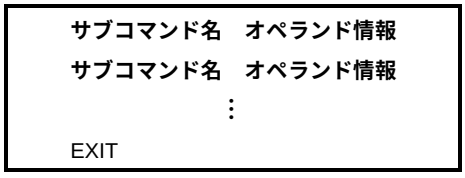
```
X>lb78k0s△  <  サブコマンド・ファイル名   [△オプション] …
            |              |
            ①              ②
```

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

(a) サブコマンド・ファイルは，エディタなどで作成してください。

(b) サブコマンド・ファイル内での記述規則を次に示します。



(c) 1つのサブコマンドが複数行にわたる場合，各行の行末に行の継続を示す‘ & ’を記述します。

(d) ‘ ; ’（セミコロン）から行末までは，ライブラリアンのコマンドとしては解釈されず，コメントとしてみなされます。

(e) サブコマンド・ファイルの最後のサブコマンドがEXITサブコマンドでない場合には，自動的にEXITサブコマンドがあったものと解釈されます。

(f) ライブラリアンは，サブコマンドをサブコマンド・ファイルから読み込んで処理を行います。
サブコマンド・ファイル内のすべてのサブコマンドの処理を終了するとライブラリアンは終了します。

例　サブコマンド・ファイル（K0S.SLB）**をエディタなどで作成します。**

K0S. SLB**の内容**

```
;library creation command
create k0s.lib
add k0s.lib k0smain.rel &
k0ssub.rel
;
exit
```

サブコマンド・ファイル（K0S.SLB）**を使用してライブラリアンを起動します。**

```
A>lb78k0s <k0s.slb
```

8. 3. 2　実行開始，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，ディスプレイに次の実行開始メッセージが表示されます。

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

(2) 実行終了メッセージ

ライブラリアンは，実行終了メッセージを出力しません。各処理を終了したあとにユーザがEXITコマンドを入力することにより，ライブラリアンは制御をOSに戻します。

```
*create k0s.lib
*add k0s.lib k0smain.rel k0ssub.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ディスプレイに出力して，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

```
A>lb78k0s -a

78K/0S Series Librarian Vx. xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx, xxxx
A018 Option is not recognized '-z'
Usage: LB78K0S [options]
```

この例では，存在しないライブラリアン・オプションを指定したためにエラーとなり，ライブラリアンの実行が中止されました。

ライブラリアンがエラー・メッセージを出力してライブラリ化を中止した場合には，そのエラー・メッセージの原因を**第12章　エラー・メッセージ**で調べて対処してください。

8.4　ライブラリアン・オプション

8.4.1　ライブラリアン・オプションの種類

ライブラリアン・オプションは，リスト・ファイルの形式やテンポラリ・ファイルの作成パスなどの指定を行います。ライブラリアン・オプションは，4種類のオプションに分類できます。

表8－2　ライブラリアン・オプション

項番	分　類	オプション	説　明
1	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行コードを付加します。
		-NLF	
2	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。
3	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。
4	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。

備考　ライブラリアン・オプションの詳細についてはC. 5　**ライブラリアン・オプション一覧**を参照してください。

8. 4. 2　ライブラリアン・オプションの説明

次に，各ライブラリアン・オプションの詳細を説明します。

（1）リスト・ファイル形式指定　（-LW, -LL, -LF/-NLF）

（a）-LW

記述形式　：-LW[文字数]
省略時解釈：-LW132（ディスプレイ出力の場合は80文字とします）

【機　　能】

-LWオプションは，リスト・ファイルの1行の文字数を指定します。

【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】

① -LWオプションで指定できる文字数の範囲は次のとおりです
（ディスプレイ出力の場合は，80文字までです）。

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。

② 文字数を省略した場合は，132を指定したものとみなします。ただし，リスト・ファイルの出力先がディスプレイの場合は，80となります。

③ 指定する文字数には，ターミネータ（CR, LF）は含みません。

④ LISTサブコマンドが指定されない場合，-LWオプションは無視されます。

⑤ -LWオプションを複数指定した場合は，あとで指定した方を優先します。

【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。

```
A>lb78k0s -lw80
```

（b）-LL

記述形式　：-LL[行数]
省略時解釈：-LL66（ディスプレイ出力の場合は改頁しません）

【機　　能】

-LLオプションは，リスト・ファイルの1頁の行数を指定します。

【用　　途】

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

【説　　明】

① -LLオプションで指定できる行数の範囲は次のとおりです。

20　≦　1頁に印字する行数　≦　32767

範囲外の数値や数値以外を指定した場合は，アボート・エラーとなります。

② 行数が省略された場合，66が指定されたものとみなします。

③ 行数の0を指定した場合は改頁しません。

④ LISTサブコマンドが指定されない場合，-LLオプションは無視されます。

⑤ -LLオプションを複数指定した場合は，あとで指定した方を優先します。

【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。

```
A>lb78k0s -ll20
```

（c）-LF/-NLF

記述形式　：-LF
　　：-NLF
省略時解釈：-NLF

【機　　能】

① -LFオプションは，リスト・ファイルの最後に改頁コード（FF）を付加する指定をします。
② -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】

① LISTサブコマンドが指定されない場合，-LFオプションは無視されます。
② -LFと-NLFの両オプションを同時に指定した場合は，あとで指定した方を優先します。

【使 用 例】

リスト・ファイルに改頁コードを付加します。

```
A>lb78k0s -lf
```

(2) テンポラリ・ファイル作成パス指定　(-T)

記述形式　：-Tパス名
省略時解釈：環境変数TMPにより指定されたパスに作成します。
　　指定されていない場合は，カレント・パスに作成します。

【機　　能】

-Tオプションは，テンポラリ・ファイルを作成するパスを指定します。

【用　　途】

テンポラリ・ファイルの作成場所を指定できます。

【説　　明】

① パス名としてパス以外のものは指定できません。
② パス名は省略できません。
③ 以前に作成されたテンポラリ・ファイルが存在している場合でも，ファイル保護をしていなければ，上書きします。
④ 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。
⑤ 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。
⑥ 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オプションで指定されたパス
2．環境変数TMPに設定されているパス（-Tオプション省略の場合）
3．カレント・パス（TMPが設定されていない場合）

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・ファイルをディレクトリb:¥TMPに出力するよう指定します。

```
A>lb78k0s -tb:¥tmp
```

(3) デバイス・ファイル サーチ・パス指定　(-Y)

記述形式　：-Yパス名

省略時形式：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 ＜..¥dev＞（lb78k0s.exeの起動されたパスに対して）

② LB78K0Sの起動されたパス

③ カレント・ディレクトリ

④ 環境変数PATH

【機　　能】

デバイス・ファイルを指定されたパスから読み込みます。

【用　　途】

デバイス・ファイルが存在するパスを指定します。

【説　　明】

① -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。

② -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。

③ 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。

a．-Yオプションで指定されたパス

b．＜..¥dev＞（lb78k0s.exeの起動されたパスに対して）

c．LB78K0Sの起動されたパス

d．カレント・ディレクトリ

e．環境変数PATH

【使 用 例】

デバイス・ファイルのパスをa:¥78k0s¥devディレクトリに指定します。

```
A>lb78k0s -ya:¥78k0s¥dev
```

(4) ヘルプ指定　(--)

記述形式　：--
省略時形式：表示しない

【機　　能】

--オプションは，ヘルプ・メッセージをディスプレイに表示します。

【用　　途】

ヘルプ・メッセージは，サブコマンドとその説明の一覧です。ライブラリアンを実行するときに参照してください。

【説　　明】

--オプションを指定すると他のライブラリアン・オプションは，すべて無効となります。

【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。

```
A>lb78k0s --

78K/0S Series Librarian Vx. xx [xx xxx xx]
  Copyright (C) NEC Electronics Corporation xxxx, xxxx
+-------------------------------------------------------------------------+
|  Subcommands : create, add, delete, replace, pick, list, help, exit     |
|                                                                         |
|    Usage : subcommand[ option]masterLBF[ option]transaction[ option]    |
|                                                                         |
|             transaction :== OMFname                                     |
|                            LBFname [ (modulename [,...] )]              |
|                                                                         |
|    <create  >: create masterLBF [ transaction]                          |
|    <add     >: add masterLBF transaction                                |
|    <delete  >: delete masterLBF (modulename [,...])                     |
|    <replace >: replace masterLBF transaction                            |
|    <pick    >: pick masterLBF (modulename [,...])                       |
|    <list    >: list [ option] masterLBF [(modulename [,...])            |
|                  option : -p  = output public symbol                    |
|                           -np = no output public symbol                 |
|                           -o filename = specify output file name        |
|    <help    >: help                                                     |
|    <exit    >: exit                                                     |
+-------------------------------------------------------------------------+
```

8.5　サブコマンド

8.5.1　サブコマンドの種類

サブコマンドは，ライブラリアンの動作に細かい指示を与えます。サブコマンドは，8種類のオプションに分類できます。

表8－3　サブコマンド

項番	サブコマンド名	短縮名	説　明
1	CREATE	C	ライブラリ・ファイルを新規に作成します。
2	ADD	A	ライブラリ・ファイルにモジュールを追加します。
3	DELETE	D	ライブラリ・ファイル内のモジュールを削除します。
4	REPLACE	R	ライブラリ・ファイル内のモジュールを他のモジュールと置き換えます。
5	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。
6	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。
7	HELP	H	ディスプレイにヘルプ・メッセージを出力します。
8	EXIT	E	ライブラリアンを終了します。

備考　サブコマンドの詳細については，**付録**D　**サブコマンド一覧**を参照してください。

8.5.2　サブコマンドの説明

各サブコマンドの詳細について説明します。

コマンド・ファイルの一般形式

＊サブコマンド［△オプション］△ライブラリ・ファイル名［△オプション］トランザクション［△オプション］

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［▲，…］）］

（1）CREATE

記述形式：CREATE△ライブラリ・ファイル名［△トランザクション］
短縮形　：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】

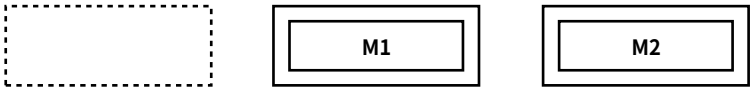
① 作成されたライブラリ・ファイルのサイズは0となります。
② トランザクションを指定した場合は，ライブラリ・ファイルの作成と同時にモジュールを登録します。
③ ライブラリ・ファイル名：
指定したファイルがすでに存在している場合には，上書きします。
④ トランザクション：
ライブラリ・ファイル中にパブリック・シンボルと同一のパブリック・シンボルを持つオブジェクト・モジュール・ファイルは，登録できません。
またライブラリ・ファイル中にあるモジュールと同一の名前のモジュールは登録できません。
⑤ エラーが発生した場合は処理を中断し，ライブラリ・ファイルは作成されません。

【使 用 例】

ライブラリ・ファイルを作成し同時にモジュールM1とM2を登録します。

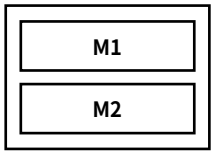
```
＊create  k0s.lib m1.rel m2.rel
```

〈作成前〉



〈作成後〉

K0S.LIB



（2）ADD

記述形式：ADD△ライブラリ・ファイル名△トランザクション
短縮形　：A

【機　　能】

既存のライブラリ・ファイルに対してモジュールを追加します。

【説　　明】

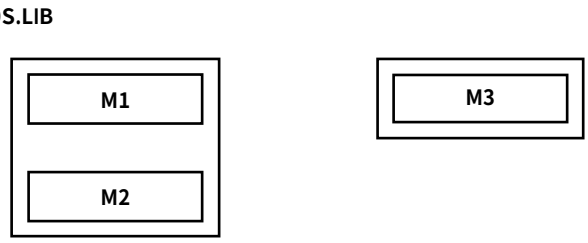
① 追加するライブラリ・ファイル中には，モジュールが存在していなくてもかまいません。
② 追加するモジュールと同名のモジュールがライブラリ・ファイル内に存在する場合，エラーとなります。
③ 追加するモジュール中のパブリック・シンボルがライブラリ・ファイル内に存在する場合，エラーとなります。

【使 用 例】

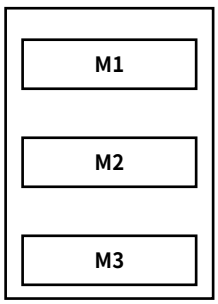
ライブラリ・ファイル（K0S.LIB）にモジュール（M3）を追加します。

```
*add k0s.lib m3.rel
```

〈追加前〉



〈追加後〉



（3）DELETE

記述形式：DELETE△ライブラリ・ファイル名▲（▲モジュール名［▲, …］▲）
短縮形　：D

【機　　能】

既存のライブラリ・ファイルからモジュールを削除します。

【説　　明】

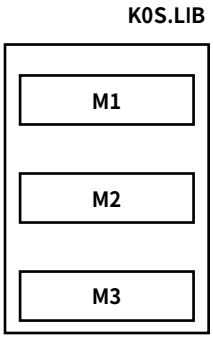
① 指定したモジュールがライブラリ・ファイルに存在しない場合，エラーとなります。
② エラーが発生した場合は処理を中断し，ライブラリ・ファイルの状態は変化しません。

【使 用 例】

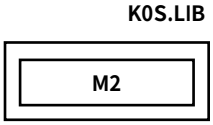
ライブラリ・ファイル（K0S.LIB）からモジュール（M1, M3）を削除します。

```
＊delete k0s.lib (m1.rel m3.rel)
```

〈削除前〉



〈削除後〉



（4）REPLACE

記述形式：REPLACE△ライブラリ・ファイル名△トランザクション
短縮形　：R

【機　　能】

既存のライブラリ・ファイルのモジュールを，他のオブジェクト・モジュール・ファイルのモジュールと置き換えます。

【説　　明】

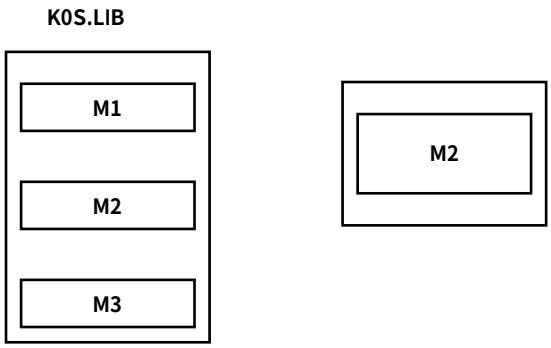
① 置換するモジュール名と同名のモジュールが，更新するライブラリ・ファイル内に存在しない場合はエラーとなります。
② 置換するモジュール中のパブリック・シンボルが，ライブラリ・ファイル内に存在する場合はエラーとなります。
③ 置換するオブジェクト・モジュール・ファイル名は，登録時と同じファイル名でなければなりません。
④ エラーが発生した場合は処理を中断し，ライブラリ・ファイルの状態は変化しません。

【使 用 例】

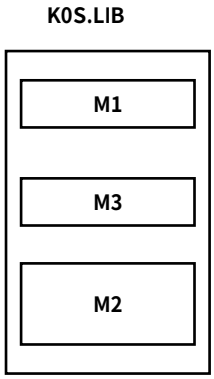
ライブラリ・ファイル（K0S.LIB）中のモジュール（M2）を置換します。

```
＊replace k0s.lib m2.rel
```

〈置換前〉



〈置換前〉



ライブラリ・ファイル中のモジュール（M2）を削除したあと，新たにモジュール（M2）を登録するため，ライブラリ・ファイル中のモジュール（M2）の順序は最後となります。

（5）PICK

記述形式：PICK△ライブラリ・ファイル名▲（▲モジュール名［▲, …］▲）
短縮形　：P

【機　　能】

既存のライブラリ・ファイルのモジュールから指定したモジュールを取り出します。

【説　　明】

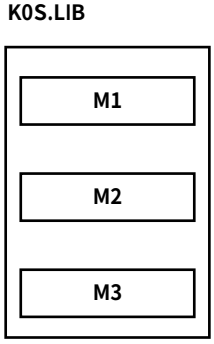
① 取り出したモジュールは，登録時のファイル名を持つオブジェクト・モジュール・ファイルとなります。
② 指定したモジュール名が，ライブラリ・ファイル内に存在しない場合はエラーとなります。
③ エラーの場合は処理を中断します。ただし，複数のモジュールが指定されているときにエラーが発生した場合には，エラーとなったモジュールの直前までに取り出されたモジュールは有効となりディスク上に保存されます。

【使 用 例】

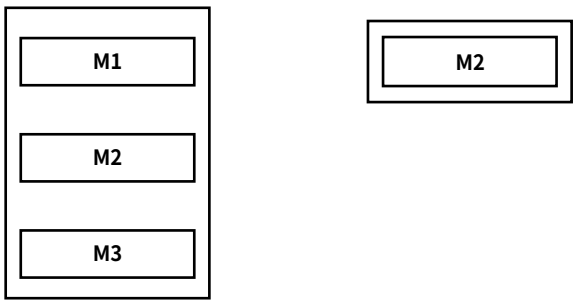
ライブラリ・ファイル（K0S.LIB）中のモジュール（M2）を取り出します。

```
＊pick k0s.lib (m2.rel)
```

〈抽出前〉



〈抽出後〉



（6）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
：-○▲ファイル名

短縮形　：L

【機　　能】

ライブラリ・ファイル内のモジュール情報を出力します。

【説　　明】

① オプションは，複数指定できます。

② -O：
出力ファイル名には，デバイス型ファイル名を指定できます。
出力ファイル名を省略した場合は，エラーとなります。
ファイル・タイプを省略した場合は，‘ 入力ファイル名.LST ’ が入力されたものとします。

③ -PUBLIC/-NOPUBLIC：
下線部のみの指定も可能です。
-PUBLICは，パブリック・シンボル情報の出力を指示します。
-NOPUBLICは，-PUBLICを無効にします。
-PUBLICと-NOPUBLICの両方を指定した場合は，あとで指定した方を優先します。

【使 用 例】

ライブラリ・ファイル（K0S.LIB）のモジュール情報をリスト・ファイル（K0S.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 –ok0s.lst k0s.lib
```

リスト・ファイル（K0S.LST）を参照します。

```
78K/0S Series librarian Vx. xx    DATE : xx xxx xx          PAGE   1
LIB-FILE NAME : K0S.LIB     (xx xxx xx)
0001  M1.REL     (xx xxx xx)
  sym1    sym2    sym3
  NUMBER OF PUBLIC SYMBOLS :   3
0002  M3.REL     (xx xxx xx)
  NUMBER OF PUBLIC SYMBOLS :   0
0003  M2.REL     (xx xxx xx)
  bit1    bit2
  NUMBER OF PUBLIC SYMBOLS :   2
```

(7) HELP

記述形式：HELP
短縮形　：H

【機　　能】

ヘルプ・メッセージをディスプレイに出力します。

【説　　明】

ヘルプ・メッセージは，サブコマンドとその説明の一覧です。HELP**コマンドまたは，**--**オプションを指定してライブラリアンを実行するときに参照してください。**

【使 用 例】

HELP**コマンドを指定するとヘルプ・メッセージが出力されます。**

```
*help

+-------------------------------------------------------------------------+
|  Subcommands : create, add, delete, replace, pick, list, help, exit     |
|                                                                         |
|    Usage : subcommand[ option]masterLBF[ option]transaction[ option]    |
|                                                                         |
|             transaction :== OMFname                                     |
|                             LBFname [ (modulename [,...])]              |
|                                                                         |
|    <create  >: create masterLBF [ transaction]                          |
|    <add     >: add masterLBF transaction                                |
|    <delete  >: delete masterLBF (modulename [,...])                     |
|    <replace >: replace masterLBF transaction                            |
|    <pick    >: pick masterLBF (modulename [,...])                       |
|    <list    >: list [ option] masterLBF [(modulename [,...])            |
|                  option : -p  = output public symbol                    |
|                           -np = no output public symbol                 |
|                           -o filename = specify output file name        |
|    <help    >: help                                                     |
|    <exit    >: exit                                                     |
+-------------------------------------------------------------------------+
```

（8）EXIT

記述形式：EXIT
短縮形　：E

【機　　能】

ライブラリアンを終了します。

【説　　明】

ライブラリアンを終了するときに使用します。

【使 用 例】

ライブラリアンを終了します。

```
＊exit
```

8. 6　PM plusでのオプション設定

PM plusからのライブラリ･ファイルの指定をする方法について，説明します。

8. 6. 1　オプションの設定方法

PM plusの［ツール(T)］メニューの［外部ツールの起動(T)］から［1 lb78k0sp］を選択または  をクリックすると，<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］を押すと，<サブコマンド実行ダイアログ>が現れます。

ダイアログ内で必要なオプションを入力することにより，各ライブラリアンのオプションを設定できます。

図8－3　<ライブラリ・ファイルの指定>ダイアログ

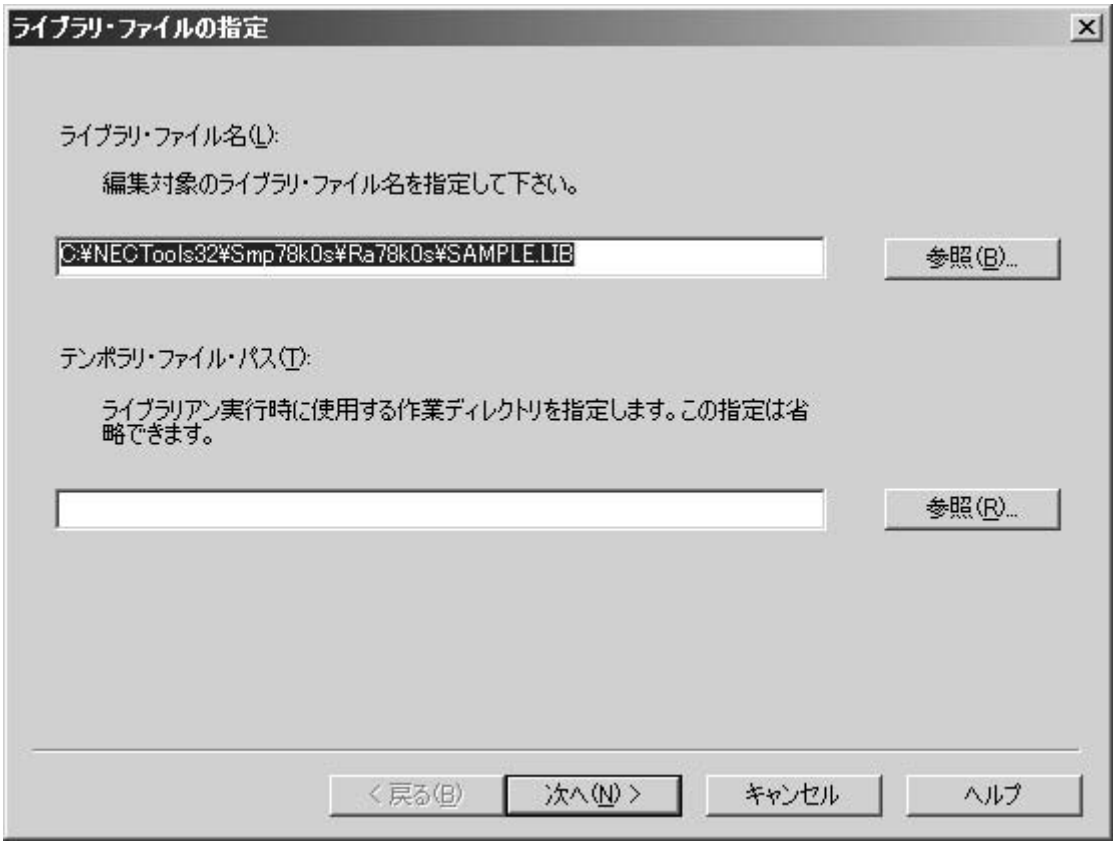


図8－4　<サブコマンド実行>ダイアログ

8. 6. 2　各オプションの設定

<ライブラリアン >ダイアログの各オプションについて，次に説明します。

・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。

・ファイルの場所(T)

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･ファイルを選択します。

- ADD

既存のライブラリ･ファイルに対してモジュールを追加します。

- PICK

既存のライブラリ･ファイルのモジュールから指定したモジュールを取り出します。

- REPLACE

既存のライブラリ･ファイルのモジュールを他のオブジェクト･モジュール･ファイルのモジュールと置き換えます。

- DELETE

既存のライブラリ･ファイルからモジュールを削除します。

- LIST

ライブラリ･ファイル内のモジュール情報を出力します。

第9章　リスト・コンバータ

リスト・コンバータは，アセンブラが出力するアセンブル・リスト・ファイル，オブジェクト・モジュール・ファイルと，リンカが出力するロード・モジュール・ファイルを入力します。

そして，入力ファイル中のリロケータブルなアドレスやシンボルに実際のアドレスを埋め込んで，アブソリュート・アセンブル・リスト・ファイルとして出力します。こうすることによって，リンク・マップを参照しながらアセンブル・リストを見るというわずらわしさが軽減されます。

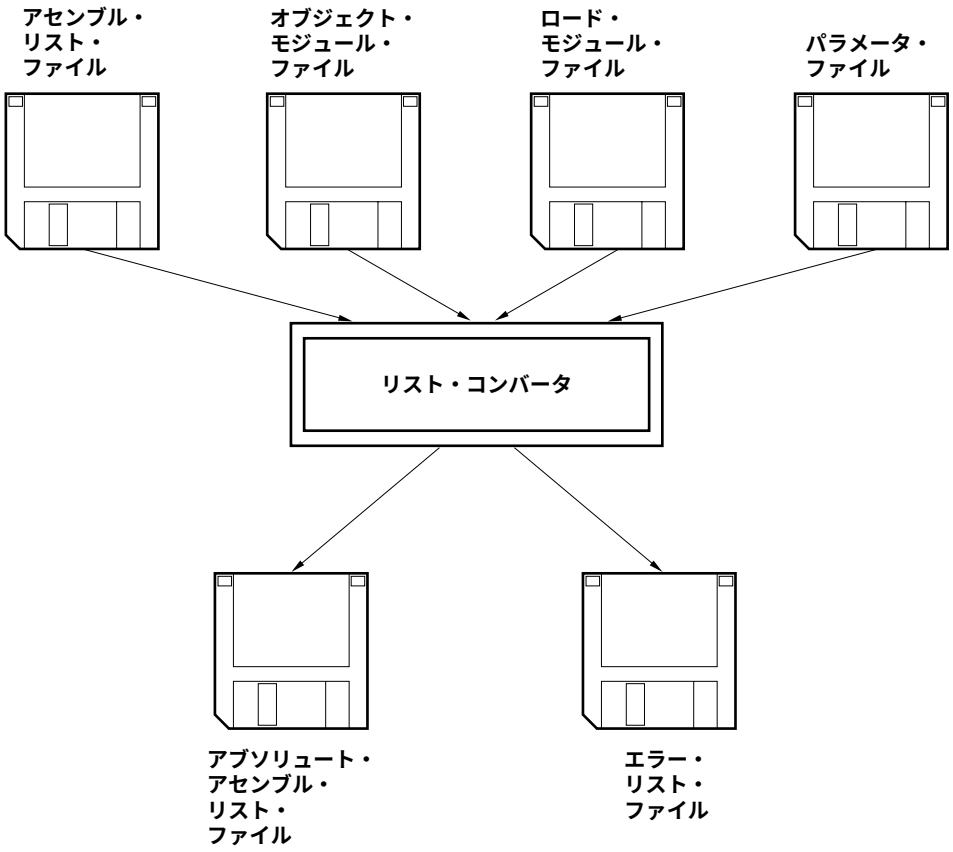
9. 1　リスト・コンバータの入出力ファイル

リスト・コンバータの入出力ファイルを次に示します。

表9－1　リスト・コンバータの入出力ファイル

種　類	ファイル名	説　　明	ディフォールト・ファイル・タイプ
入力ファイル	オブジェクト・モジュール・ファイル	○機械語情報と機械語の配置アドレスに関する再配置情報およびシンボル情報を含んだバイナリ・ファイルです。	.REL
	アセンブル・リスト・ファイル	○アセンブル・リスト，クロスレファレンス・リストなどのアセンブル情報を持つファイルです。	.PRN
	ロード・モジュール・ファイル	○リンク結果のオブジェクト・コードのバイナリ・イメージ・ファイルです。	.LMF
	パラメータ・ファイル	○実行プログラムのパラメータを内容とするファイルです。 ○ユーザ作成ファイルです。	.PLV
出力ファイル	アブソリュート・アセンブル・リスト・ファイル	○入力ファイル中のリロケータブルなアドレスやシンボルに実際のアドレスを埋め込んだリスト・ファイルです。	.P
	エラー・リスト・ファイル	○リスト・コンバート時のエラー情報を持つファイルです。	.ELV

図9－1　リスト・コンバータの入出力ファイル



9.2　リスト・コンバータの機能

リロケータブル・アセンブラをアブソリュート・アセンブラと比較した場合の長所，短所を次に示します。

[長所]

① 複数の人数で開発可能。
② モジュール分割により開発，保守が容易。
③ ライブラリ管理。
④ 大規模プログラムの開発に適している。

[短所]

① アセンブル・リストのアドレスが実際の物理アドレスと一致しない。
② 外部シンボルの値がアセンブル・リスト中では0になっており，実際の値は，リンク・マップを参照しなければならない。
③ アセンブル・リスト中のリロケータブルな値は，実際の値と異なる。

上記の短所は，特にディバグ時や保守のためのドキュメント面で生産性の低下を招きます。
リスト・コンバータは，リロケータブル・アセンブラ・パッケージの上記の欠点を解決できます。

① リスト・コンバータの出力したアブソリュート・アセンブル・リストは，動作時のアドレスと完全に一致しています。
② 外部シンボルの実際の値がリスト上に埋め込まれます。
③ リロケータブルな値がリスト上に実際の値として埋め込まれます。
④ シンボル・テーブルあるいはクロスレファレンス・リスト上のシンボル値に対しても，実際の値が埋め込まれます。

例1.　**ロケーションの埋め込み**

●アセンブル・リスト

```
 21  21 ----            CSEG
 22  22 0000        START:
 23  23
 24  24                ;chip initialize
 25  25
 26  26 0000  F5201A  MOV    HDTSA, #1AH
 27  27 0003  FC20FE  MOVW   HL, #HDTSA ;set hex 2-code data in HL registor
 28  28
 29  29 0006 R220000  CALL   !CONVAH    ;convert ASCII<- HEX
 30  30                                 ;output BC-register<- ASCII code
 31  31 0009  F821FE  MOVW   DE, #STASC ;set DE<- store ASCII code table
 32  32 000C  0A27    MOV    A, B
 33  33 000E  EB      MOV    [DE], A
 34  34 000F  88      INCW   DE
 35  35 0010  0A25    MOV    A, C
 36  36 0012  EB      MOV    [DE], A
```

●アブソリュート・アセンブル・リスト

```
 21  21 ----            CSEG
 22  22 0080        START:
 23  23
 24  24                ;chip initialize
 25  25
 26  26 0080  F5201A  MOV    HDTSA, #1AH
 27  27 0083  FC20FE  MOVW   HL, #HDTSA ;set hex 2-code data in HL registor
 28  28
 29  29 0086 R229500  CALL   !CONVAH    ;convert ASCII<- HEX
 30  30                                 ;output BC-register<- ASCII code
 31  31 0089  F821FE  MOVW   DE, #STASC ;set DE<- store ASCII code table
 32  32 008C  0A27    MOV    A, B
 33  33 008E  EB      MOV    [DE], A
 34  34 008F  88      INCW   DE
 35  35 0090  0A25    MOV    A, C
 36  36 0092  EB      MOV    [DE], A
```

例2.　**オブジェクト・コードの埋め込み**

●アセンブル・リスト

```
21  21 ----          CSEG
22  22 0000        START:
23  23
24  24               ;chip initialize
25  25
26  26 0000  F5201A  MOV    HDTSA, #1AH
27  27 0003  FC20FE  MOVW   HL, #HDTSA ;set hex 2-code data in HL registor
28  28
29  29 0006 R220000  CALL   !CONVAH    ;convert ASCII<- HEX
30  30                                 ;output BC-register<- ASCII code
31  31 0009  F821FE  MOVW   DE, #STASC ;set DE<- store ASCII code table
32  32 000C  0A27    MOV    A, B
33  33 000E  EB      MOV    [DE], A
34  34 000F  88      INCW   DE
35  35 0010  0A25    MOV    A, C
36  36 0012  EB      MOV    [DE], A
```

●アブソリュート・アセンブル・リスト

```
21  21 ----          CSEG
22  22 0080        START:
23  23
24  24               ;chip initialize
25  25
26  26 0080  F5201A  MOV    HDTSA, #1AH
27  27 0083  FC20FE  MOVW   HL, #HDTSA ;set hex 2-code data in HL registor
28  28
29  29 0086 R229500  CALL   !CONVAH    ;convert ASCII<- HEX
30  30                                 ;output BC-register<- ASCII code
31  31 0089  F821FE  MOVW   DE, #STASC ;set DE<- store ASCII code table
32  32 008C  0A27    MOV    A, B
33  33 008E  EB      MOV    [DE], A
34  34 008F  88      INCW   DE
35  35 0090  0A25    MOV    A, C
36  36 0092  EB      MOV    [DE], A
```

9.3　リスト・コンバータの起動方法

9.3.1　リスト・コンバータの起動

リスト・コンバータの起動には，次の2つの方法があります。

(1) コマンド行での起動

```
X>lc78k0s [△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。
④ アセンブル・リストのプライマリ・ネーム

例　A>lc78k0s k0smain –lk0s.lmf

注意1. 上記③で，複数のリストコンバータ・オプションを指定する場合には，それぞれのリスト・コンバータ・オプション間を空白で区切ってください。
なお，オプションの詳細については，9.4　リスト・コンバータ・オプションを参照してください。

2. 上記④のファイルの拡張子は‘.PRN’にしてください。

3. 上記④で，コマンド行にアセンブル・リストのプライマリ・ネームのみを指定する場合には，オブジェクト・モジュール・ファイル，ロード・モジュール・ファイルのプライマリ・ネームは，アセンブル・リスト・ファイルのプライマリ・ネームと同一でなければなりません。
また，ファイル・タイプは次のようになっていなければなりません。

表9－2　リスト・コンバータ起動時の指定ファイル・タイプ

ファイル名	タ イ プ
オブジェクト・モジュール・タイプ	.REL
ロード・モジュール・ファイル	.LMF

プライマリ・ネームの異なるファイルを指定する場合は，オプションを使用します。

(2) パラメータ・ファイルによる起動

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

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

パラメータ・ファイルによる起動方法は，次のようになります。

```
X>lc78k0s [△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・ファイルでの生成規則を次に示します。

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

① 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。

② 入力ファイル名は，オプションのあとにも記述できます。

③ パラメータ・ファイルには，コマンド行で指定すべきすべてのリスト・コンバータ・オプション，出力ファイル名を記述します。

例　パラメータ・ファイル（K0S.PLV）をエディタで作成します。

K0S.PLVの内容

```
;parameter file
k0smain –lk0s.lmf
-ek0s.elv
```

パラメータ・ファイル（K0S.PLV）を使用してリスト・コンバータを起動します。

```
A>lc78k0s –fk0s.plv
```

9. 3. 2　実行開始，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，ディスプレイに実行開始メッセージが表示されます。

```
List Conversion Program for RA78K/0S 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ディスプレイに出力して制御を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ディスプレイに出力して処理を中止し，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

```
A>lc78k0s k0smain -a

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

A018 Option is not recognized '-a'
Program aborted.
```

リスト・コンバータがエラー・メッセージを出力して処理を中止した場合は，そのエラー・メッセージの原因を第12章　エラー・メッセージで調べて対処してください。

9.4　リスト・コンバータ・オプション

9.4.1　リスト・コンバータ・オプションの種類

リスト・コンバータ・オプションは，リスト・コンバータの動作に細かい指示を与えるものです。リスト・コンバータ・オプションは，6種類のオプションに分類できます。

表9－3　リスト・コンバータ・オプション

項番	分　類	オプション	説　明
1	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。
2	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。
3	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。
4	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。
5	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。
6	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。

備考　リスト・コンバータ・オプションの詳細については，C. 6　**リスト・コンバータ・オプション一覧**を参照してください。

9. 4. 2　リスト・コンバータ・オプションの説明

各リスト・コンバータ・オプションの詳細について説明します。

（1）オブジェクト・モジュール・ファイル入力指定　（-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定します。

【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】

① 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。
② 入力ファイル名のプライマリ・ネームのみを指定した場合は，ファイル・タイプとして‘ .REL ’を付加してファイルを入力します。

【使 用 例】

アセンブル・リスト・ファイル名がK0SMAIN.PRNで，オブジェクト・モジュール・ファイル名がSAMPLE.RELで，ロード・モジュール・ファイル名がK0S.LMFの場合

```
A>lc78k0s k0smain -rsample.rel –lk0s.lmf
```

(2) ロード・モジュール・ファイル入力指定　(-L)

記述形式　：-L[入力ファイル名]
省略時解釈：-Lアセンブル・リスト・ファイル名.LMF

【機　　能】

-Lオプションは，ロード・モジュール・ファイルの入力を指定します。

【用　　途】

ロード・モジュール・ファイルのプライマリ・ネームがアセンブル・リスト・ファイルのプライマリ・ネームと異なる場合またはファイル・タイプが“.LMF”でない場合に，-Lオプションを指定します。

【説　　明】

① 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。
② 入力ファイル名のプライマリ・ネームのみを指定した場合は，ファイル・タイプとして‘.LMF’を付加してファイルを入力します。

【使 用 例】

アセンブル・リスト・ファイル名がK0SMAIN.PRNで，ロード・モジュール・ファイル名がSAMPLE.LMFの場合

```
A>lc78k0s k0smain -lsample.lmf
```

(3) アブソリュート・アセンブル・リスト・ファイル出力指定　(-O)

記述形式　：-O[出力ファイル名]
省略時解釈：-Oアセンブル・リスト・ファイル名.P

【機　　能】

-Oオプションは，アブソリュート・アセンブル・リスト・ファイルの出力を指定します。
また，その出力先や出力ファイル名も指定できます。

【用　　途】

アブソリュート・アセンブル・リスト・ファイルの出力先や出力ファイル名を変更したいときに，-Oオプションを指定します。

【説　　明】

① ファイル名として，ディスク型ファイル名とデバイス型ファイル名を指定できます。指定できるデバイス型ファイル名はCON, PRN, NULおよびAUXです。CLOCKを指定した場合，アボート・エラーとなります。
② 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。
③ -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。
④ 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。
⑤ -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。

```
A>lc78k0s k0smain -osample.p –lk0s.lmf
```

(4) エラー・リスト・ファイル出力指定　(-E/-NE)

記述形式　：-E[出力ファイル名]
　　：-NE
省略時解釈：-NE

【機　　能】

① -Eオプションは，エラー・リスト・ファイルの出力を指定します。また，その出力先や出力ファイル名も指定できます。
② -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】

① ファイル名として，ディスク型ファイル名とデバイス型ファイル名を指定できます。ただし，デバイス型ファイル名として，CLOCKを指定した場合は，アボート・エラーとなります。
② ファイル名にアブソリュート・アセンブル・リスト・ファイルと同一のデバイスが指定された場合，アボート・エラーとなります。
③ -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。
④ 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。
⑤ -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。
⑥ -Eと-NEオプションを同時に指定した場合は，あとで指定した方を優先します。

【使 用 例】

例　エラー・リスト・ファイル（SAMPLE.ELV）を作成します。

```
A>lc78k0s k0smain -esample.elv
```

エラー・リスト・ファイル（SAMPLE.ELV）を参照します。

```
*** WARNING W101 Load module file is older than object module file 'K0SMAIN.LMF,
K0SMAIN.REL'
Pass1: start
*** ERROR A105 Segment name is not found is load module file 'DATA'
```

(5) パラメータ・ファイル指定　(-F)

記述形式　：-Fファイル名
省略時解釈：コマンド行上からのみオプション，入力ファイル名の入力が可能

【機　　能】

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

【用　　途】

① 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。
② 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】

① ‘ ファイル名 ’ として指定できるのは，ディスク型ファイル名のみです。デバイス型ファイル名を指定するとアボート・エラーとなります。
② ファイル名を省略するとアボート・エラーとなります。
③ 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。
④ 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。
⑤ パラメータ・ファイル中に記述できる文字数の制限はありません。
⑥ 空白とタブおよび ‘ ↵ ’ 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。
⑦ 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。
⑧ 展開されたオプションは，あとで指定したものが優先です。
⑨ -Fオプションを複数指定するとアボート・エラーとなります。

【使 用 例】

パラメータ・ファイルを使用してリスト・コンバータを起動させます

パラメータ・ファイル（K0S.PLV）の内容

```
:parameter file
k0smain –lk0s.lmf
-ek0s.elv
```

コマンド行には，次のように入力します

```
A>lc78k0s –fk0s.plv
```

(6) ヘルプ指定　(--)

記述形式　：--
省略時解釈：表示しない

【機　　能】

--オプションは，ヘルプ・メッセージをディスプレイに表示します。

【用　　途】

ヘルプ・メッセージは，リスト・コンバータ・オプションとその説明の一覧です。リスト・コンバータを実行するときに参照してください。

【説　　明】

--オプションを指定すると他のリスト・コンバータ・オプションは，すべて無効となります。

【使 用 例】

--オプションを指定するとヘルプ・メッセージかディスプレイに出力されます。

```
A>lc78k0s --

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

usage : LC78K0S [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).
-f [file]:Input option or input-file name from specified file.
-e [file]:Create error list file.
--       :Show this message.
```

9. 5　PM plusでのオプション設定

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

9. 5. 1　オプションの設定方法

PM plusの［ツール(T)］メニューの［リストコンバータオプションの設定(N)...］を選択または  をクリックすると，<リストコンバータオプションの設定>ダイアログが現れます。

ダイアログ内で必要なオプションを入力することにより，各リストコンバータ･オプションを設定できます。

図9－2　<リストコンバータオプションの設定> ダイアログ（《出力》タブ選択時）

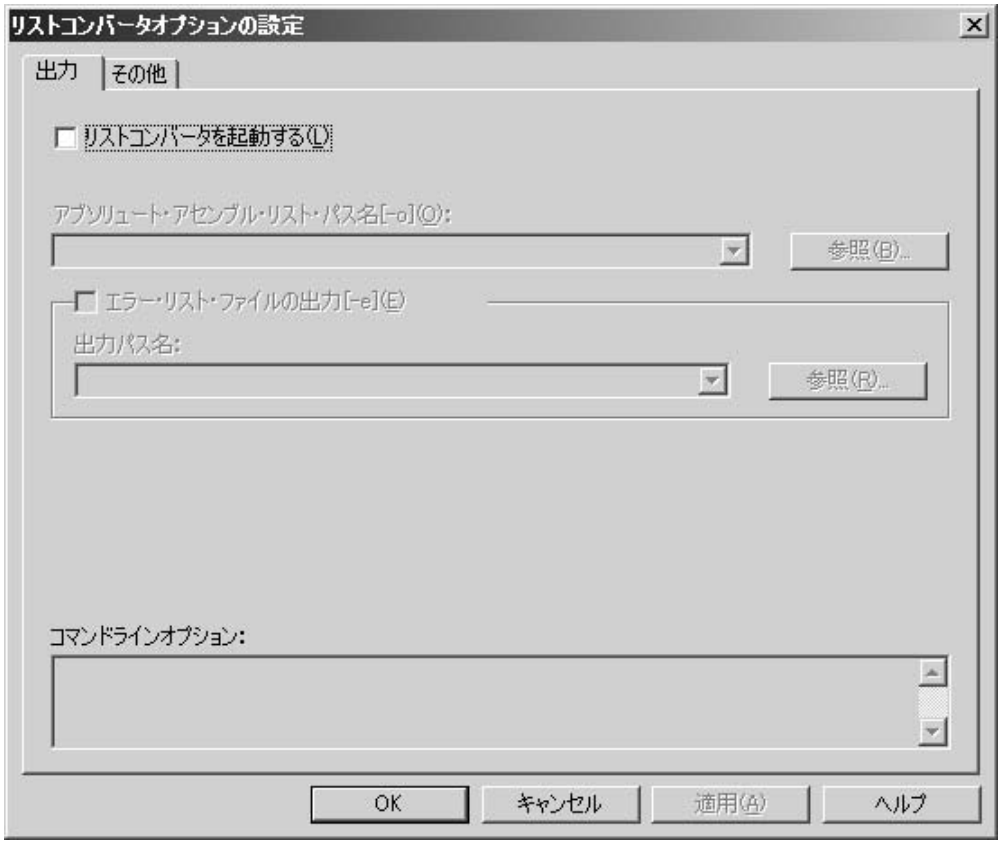


図9－3　<リストコンバータオプションの設定> ダイアログ（《その他》タブ選択時）

9. 5. 2　各オプションの設定

<リストコンバータオプションの設定>ダイアログの各オプションについて，次に説明します。

・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はチェックします。

・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)

出力パス名：

エラー･リスト･ファイルを出力したい場合は，入力ボックスにファイル名を入力します。

パスを指定する場合は参照(R)ボタンを使用します。

・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チェックしてください。

・パラメータ･ファイル(P)

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

・その他のオプション(O)

チェック･ボックスやラジオ･ボタンで選択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。

・リセット(R)

入力した内容をリセットします。

・オプション情報読込(E)

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

・オプション情報保存(S)

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

・コマンドラインオプション

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

第10章　プログラムの出力リスト

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。

○アセンブルの出力リスト

アセンブル・リスト・ファイルのヘッダ

アセンブル・リスト

シンボル・リスト

クロスレファレンス・リスト

エラー・リスト

○リンカの出力リスト

リンク・リスト・ファイルのヘッダ

マップ・リスト

パブリック・シンボル・リスト

ローカル・シンボル・リスト

エラー・リスト

○オブジェクト・コンバータの出力リスト

エラー・リスト

○ライブラリアンの出力ファイル

ライブラリ情報出力リスト

○リスト・コンバータの出力リスト

アブソリュート・アセンブル・リスト

エラー・リスト

10. 1　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。

出力リスト・ファイル名	出力リスト名
アセンブル・リスト・ファイル	アセンブル・リスト
	シンボル・リスト
	クロスレファレンス・リスト
エラー・リスト・ファイル	エラー・リスト

10. 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】

```
78K/0S Series Assembler ①Vx.xx ②          Date:③xx xxx xxxx Page:④  1
⑤
Command:  ⑥k0smain.asm –c9024
Para-file:⑦
In-fine:  ⑧K0SMAIN.ASM
Obj-file: ⑨K0SMAIN.REL
Prn-file: ⑩K0SMAIN.PRN
```

【出力項目の説明】

項　目	内　　容
①	アセンブラのバージョン番号
②	タイトル文字列 -LHオプションまたはTITLE制御命令によって指定された文字列
③	アセンブル・リストの作成年月日
④	ページ番号
⑤	サブタイトル文字列 SUBTITLE制御命令によって指定された文字列
⑥	コマンド行のイメージ
⑦	パラメータ・ファイルの内容
⑧	入力ソース・モジュール・ファイル名
⑨	出力オブジェクト・モジュール・ファイル名
⑩	アセンブル・リスト・ファイル名

10. 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

   ALNO  STNO ADRS  OBJECT  M I  SOURCE STATEMENT
               ③④
   ①1   ②1
   ②2   ②2                  ⑤    NAME     SAMPM

        ⋮

   28    28
   29    29 ⑥0006 ⑧R220000  ⑤    CALL  !CONVAH  ;convert ASCII<- HEX
   30    30                 ⑤                   ;output BC-register<- ASCII code
   31    31 ⑥0009  00000000        MOV   DE, #STASC;set DE<-store ASCII code table
⑦**    ERROR F202, STNO    31 (  0) Illegal operand
           ⑥000D  00
   32    32 ⑥000E ⑧0A27     ⑤     MOV    A, B
   33    33 ⑥0010 ⑧EB       ⑤     MOV    [DE], A

        ⋮

   Segment informations:

   ADRS    LEN     NAME

  ⑨FE20  ⑩0003H  ⑪DATA
  ⑨0000  ⑩0002H  ⑪CODE
  ⑨0000  ⑩0017H  ⑪?CSEG

   Target chip:⑫uPD78xxx
   Device file:⑬Vx. xx
   Assembly complete,    ⑭1 error(s) and    ⑮0 warning(s) found. (  ⑯31)
```

【出力項目の説明】

項　目	内　　容
①	ソース・モジュールのイメージの行番号
②	行番号（INCLUDEファイルの展開，マクロ展開も含みます）
③	マクロ表示 M　：マクロ定義行です。 #n　：マクロ展開行です。nはネスト・レベルです。 空白 ：マクロ定義行／マクロ展開行ではありません。
④	INCLUDE表示 In　：INCLUDEファイル中です。nはネスト・レベルです。 空白 ：INCLUDEファイル未使用
⑤	ソース・プログラム・ステートメント
⑥	ロケーション・カウンタ値
⑦	フェイタル・エラー／ワーニング発生行
⑧	リロケーション情報 R　：リンカによってオブジェクト・コードまたはシンボル値が変更されます。 空白 ：オブジェクト・コードまたはシンボル値が変更されません。
⑨	セグメント・アドレス
⑩	セグメント・サイズ
⑪	セグメント名
⑫	RA78K0S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

10. 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

   VALUE     ATTR      RTYP      NAME              VALUE     ATTR      RTYP      NAME
          ②CSEG                ④?CSEG                      ②CSEG                ④CODE
①  ----H              ③EXT     ④CONVAH                     ②DSEG                ④DATA
①  FE20H    ADDR                ④HDTSA            ①  0H    ②ADDR     ③PUB      ④MAIN
           MOD                  ④SAMPM            ①  0H    ②ADDR     ③PUB      ④START
①  FE21H    ADDR                ④STASC
```

【出力項目の説明】

項　目	内　容
①	**シンボル値**
②	**シンボル属性** CSEG：**コード・セグメント名**　　NUM　：NUMBER**属性シンボル** DSEG：**データ・セグメント名**　　ADDR：ADDRESS**属性シンボル** BSEG：**ビット・セグメント名**　　SABIT：BIT**属性シンボル**（saddr.bit） MAC　：**マクロ名**　　SFBIT：BIT**属性シンボル**（sfr.bit） MOD　：**モジュール名**　　RBIT　：BIT**属性シンボル**（A.bit, X.bit） SET　：SET**疑似命令によって定義されたシンボル**　　**空白**　：EXTRN**または**EXTBIT**宣言された外部参照シンボル** *****　：**未定義シンボル**
③	**シンボル参照形式** EXT　：EXTRN**宣言された外部参照シンボル**（SADDR**属性**） EXTB　：EXTBIT**宣言された外部参照シンボル**（saddr.bit） PUB　：PUBLIC**宣言された外部定義シンボル** **空白**　：**ローカル・シンボル，セグメント名，マクロ名，モジュール名** *****　：**未定義シンボル**
④	**定義されたシンボル名**

10. 1. 4　クロスレファレンス・リスト

ソース・モジュール内で定義されたシンボルがソース・モジュールのどこで（行番号）参照されているかという情報が出力されます。

【出力形式】

```
    Cross-Reference List

  NAME         VALUE      R     ATTR       RTYP       SEGNAME   XREFS
①?CSEG                        ④CSEG                  ⑥?CSEG    ⑦21#
①CODE                         ④CSEG                  ⑥CODE     ⑦18#
①CONVAH      ②----H     ③E               ⑤EXT                  ⑦12     29
①DATA                         ④DSEG                  ⑥DATA     ⑦14#
①HDTSA       ②FE20H           ④ADDR                  ⑥DATA     ⑦15#    26    27
①MAIN        ②0H              ④ADDR      ⑤PUB        ⑥CODE     ⑦11@    19#
①SAMPM                        ④MOD                               ⑦2#
①START       ②0H        ③R    ④ADDR      ⑤PUB        ⑥?CSEG    ⑦11@    19    22#
①STASC       ②FE21H           ④ADDR                  ⑥DATA     ⑦16#    31
```

【出力項目の説明】

項　目	内　容
①	**定義されたシンボル名**
②	**シンボル値**
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**　NUM　：NUMBER**属性シンボル** DSEG：**データ・セグメント名**　ADDR　：ADDRESS**属性シンボル** BSEG：**ビット・セグメント名**　SABIT　：BIT**属性シンボル**（saddr.bit） MAC　：**マクロ名**　SFBIT　：BIT**属性シンボル**（sfr.bit） MOD　：**モジュール名**　RBIT　：BIT**属性シンボル**（A.bit, X.bit） SET　：SET**疑似命令によって定義されたシンボル**　**空白**　：EXTRN**または**EXTBIT**宣言された外部参照シンボル** *****　：**未定義シンボル**
⑤	**シンボル参照形式** EXT　：EXTRN**宣言された外部参照シンボル**（SADDR**属性**） EXTB　：EXTBIT**宣言された外部参照シンボル**（saddr.bit） PUB　：PUBLIC**宣言された外部定義シンボル** **空白**　：**ローカル・シンボル，セグメント名，マクロ名，モジュール名** *****　：**未定義シンボル**
⑥	**定義されたシンボル名**
⑦	**定義・参照行番号** **定義行：**×××××# **参照行：**×××××▲　**（▲は空白1つ）** EXTRN**宣言，**EXTBIT**宣言，**PUBLIC**宣言：**×××××@

10. 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
①ERROR.ASM (②26) :③F202④Illegal operand
①ERROR.ASM (②32) :③F202④Illegal operand
  Pass2 Start
①ERROR.ASM (②26) :③F202④Illegal operand
①ERROR.ASM (②29) :③F407④Undefined symbol reference ’ DTSA’
①ERROR.ASM (②29) :③F303④Illegal expression
①ERROR.ASM (②32) :③F202④Illegal operand
①ERROR.ASM (②37) :③F407④Undefined symbol reference ’ F’
①ERROR.ASM (②37) :③F303④Illegal expression
```

【出力項目の説明】

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	エラー番号
④	エラー・メッセージ

10. 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。

出力リスト・ファイル名	出力リスト名
リンク・リスト・ファイル	マップ・リスト
	パブリック・シンボル・リスト
	ローカル・シンボル・リスト

10. 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】

```
78K/0S Series Linker ①Vx. xx       Date:②xx xxx xxxx Page:③1

Command:   ④k0smain.rel k0ssub.rel –ok0s.map –dk0s.dr
Para-file: ⑤
Out-file:  ⑥K0S.LMF
Map-File:  ⑦K0SMAIN.MAP
Direc-File:⑧
Directive: ⑨

*** Link information ***
⑩  3 output segment(s)
⑪  37H byte(s) real data
⑫  23  symbol(s) defined
```

【出力項目の説明】

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

10. 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 =②ROM
    BASE ADDRESS =③0000H  SIZE =④2000H
         OUTPUT   INPUT     INPUT    BASE        SIZE
         SEGMENT            SEGMENT             MODULE      ADDRESS
       ⑥CODE                        ⑨0000H     ⑩0002H      ⑪CSEG AT
              ⑦CODE     ⑧SAMPM   ⑨0000H     ⑩0002H
⑤* gap *                          ⑨0002H     ⑩007EH
       ⑥?CSEG                     ⑨0080H     ⑩0035H      ⑪CSEG
              ⑦?CSEG    ⑧SAMPM   ⑨0080H     ⑩0015H
              ⑦?CSEG    ⑧SAMPS   ⑨0095H     ⑩0020H
⑤* gap *                          ⑨00B5H     ⑩1F4BH

    MEMORY = RAM
    BASE ADDRESS = ③FE00H      SIZE =④0200H
         OUTPUT INPUT       INPUT    BASE        SIZE
         SEGMENT           SEGMENT   MODULE      ADDRESS
⑤* gap *                          ⑨FE00H     ⑩0020H
       ⑥DATA                      ⑨FE20H     ⑩0003H      ⑪DSEG AT
              ⑦DATA     ⑧SAMPM   ⑨FE20H     ⑩0003H
⑤* gap *                          ⑨FE23H     ⑩00DDH
⑤* gap (Not Free Area) *          ⑨FE00H     ⑩0100H

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

【出力項目の説明】

項　目	内　容
①	メモリ空間名
②	メモリ領域名
③	メモリ領域の先頭アドレス
④	メモリ領域のサイズ
⑤	出力グループ 何も配置されていないエリアがある場合‘ gap ’を表示します。
⑥	ロード・モジュール・ファイルに出力されるセグメント名
⑦	オブジェクト・モジュール・ファイルから読み込まれたセグメント名
⑧	入力モジュール名
⑨	セグメントの先頭アドレス
⑩	出力セグメント／入力セグメントのサイズ
⑪	セグメント・タイプ，再配置属性
⑫	このアセンブルの対象製品
⑬	デバイス・ファイルのバージョン番号

10. 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 ***

MODULE    ATTR    VALUE      NAME

①SAMPM    ②ADDR   ③0000H     ④MAIN
①SAMPM    ②ADDR   ③0080H     ④START
①SAMPS    ②ADDR   ③0095H     ④CONVAH
```

【出力項目の説明】

項　目	内　容
①	パブリック・シンボルが定義されたモジュール名
②	シンボル属性 CSEG：コード・セグメント名　NUM　：NUMBER属性シンボル DSEG：データ・セグメント名　ADDR　：ADDRESS属性シンボル BSEG：ビット・セグメント名　SABIT　：BIT属性シンボル（saddr.bit） MAC　：マクロ名　SFBIT　：BIT属性シンボル（sfr.bit） MOD　：モジュール名　RBIT　：BIT属性シンボル（A.bit, X.bit） SET　：SET疑似命令によって定義されたシンボル　空白　：EXTRNまたはEXTBIT宣言された外部参照シンボル *****　：未定義シンボル
③	シンボル値
④	パブリック・シンボル名

10. 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 ***

  MODULE    ATTR    VALUE       NAME

①SAMPM    ②MOD                ④SAMPM
①SAMPM    ②DSEG               ④DATA
①SAMPM    ②ADDR   ③FE20H      ④HDTSA
①SAMPM    ②ADDR   ③FE21H      ④STASC
①SAMPM    ②CSEG               ④CODE
①SAMPM    ②CSEG               ④?CSEG
①SAMPS    ②MOD                ④SAMPS
①SAMPS    ②CSEG               ④?CSEG
①SAMPS    ②ADDR   ③00ACH      ④SASC
①SAMPS    ②ADDR   ③00B2H      ④SASC1
```

【出力項目の説明】

項　目	内　容	
①	ローカル・シンボルが定義されたモジュール名	
②	シンボル属性	
	CSEG：コード・セグメント名	NUM　：NUMBER属性シンボル
	DSEG：データ・セグメント名	ADDR：ADDRESS属性シンボル
	BSEG：ビット・セグメント名	SABIT：BIT属性シンボル（saddr.bit）
	MAC　：マクロ名	SFBIT：BIT属性シンボル（sfr.bit）
	MOD　：モジュール名	RBIT　：BIT属性シンボル（A.bit, X.bit）
	SET　：SET疑似命令によって定義されたシンボル	空白　：EXTRNまたはEXTBIT宣言された外部参照シンボル
		*****　：未定義シンボル
③	シンボル値	
④	ローカル・シンボル名	

10. 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】

```
*** ERROR ①F405 ②Undefined symbol 'CONVAH' in file 'K0SMAIN.REL'
```

【出力項目の説明】

項　目	内　容
①	エラー番号
②	エラー・メッセージ

10. 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。

出力リスト・ファイル名	出力リスト名
エラー・リスト・ファイル	エラー・リスト

10. 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。

10. 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。

出力リスト・ファイル名	出力リスト名
リスト・ファイル	ライブラリ情報出力リスト

10. 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】

```
  78K/0S Series librarian Vx. xx    DATE:① xx xxx xx       PAGE②1
  LIB-FILE NAME:③K0S.LIB     (④xx xxx xx)
⑤0001  ⑥K0SMAIN.REL     (⑦xx xxx xx)
   ⑧MAIN                          ⑧START
   NUMBER OF PUBLIC SYMBOLS:  ⑨2
⑤0002  ⑥K0SSUB.REL     (⑦xx xxx xx)
   ⑧CONVAH
   NUMBER OF PUBLIC SYMBOLS:  ⑨1
```

【出力項目の説明】

項　目	内　容
①	リストの作成年月日
②	ページ数
③	ライブラリ・ファイル名
④	ライブラリ・ファイルの作成年月日
⑤	モジュールの通番（0001から番号をつけます）
⑥	モジュール名
⑦	モジュール作成年月日
⑧	パブリック・シンボル名
⑨	モジュール内で定義されているパブリック・シンボル数

10. 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。

出力リスト・ファイル名	出力リスト名
アブソリュート・アセンブル・リスト・ファイル	アブソリュート・アセンブル・リスト
エラー・リスト・ファイル	エラー・リスト

10. 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。

10. 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11章　RA78K0Sの活用法

RA78K0Sを効率よく使用するための方法を紹介します。

11. 1　作業の効率化（EXITステータス機能）

RA78K0Sの各プログラムは，処理終了時に，処理中に発生した最大のエラー・レベルをEXITステータスとして，OSに返します。

EXITステータスを次に示します。

- 正常終了時　　：0
- WARNINGあり　　：0
- FATAL ERRORあり　：1
- ABORT時　　：2

これらを利用してバッチ・ファイルを作成することで効率良く作業できます。

【使 用 例】

バッチ・ファイル（RA.BAT）の内容

```
ra78K0s –c9024 k0smain.-g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo¥
echo on
ra78K0s –c9024 k0ssub.asm -g -e
echo off
IF ERRORLEVEL 1 GOTO ERR
echo¥
echo on
lk78K0s k0smain.rel k0ssub.rel –ok0s.lmf -g
echo off
IF ERRORLEVEL 1 GOTO ERR
echo¥
echo on
oc78K0s k0s.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。

```
A>ra.bat
```

11. 2　開発環境の整備（環境変数）

RA78K0Sでは開発環境を整えるため，次の環境変数をサポートしています。

PATH	：実行形式のサーチ・パス
INC78K0S	：インクルード・ファイルのサーチ・パス（アセンブラのみ）
LIB78K0S	：ライブラリ・ファイルのサーチ・パス（リンカのみ）
TMP	：テンポラリ・ファイルを作成するパス
LANG78K	：漢字種別の指定

プログラム開発を行う場合，関連ファイル別にサブディレクトリを作成し，関連ファイルをひとまとめにしておくと作業が円滑にできるようになります。

【使 用 例】

AUTOEXEC.BATの内容

```
;AUTOEXEC.BAT
Verify on
break on
PATH A:¥BIN;A:¥BAT;A:¥RA78K0S;      ←①
SET INC78K0S = A:¥RA78K0S¥INCLUDE  ←②
SET LIB78K0S = A:¥RA78K0S¥LIB      ←③
SET TMP = A:¥TMP                   ←④
SET LANG78K = SJIS                 ←⑤
```

① パス指定により，A：¥BIN, A：¥BAT, A：¥RA78K0S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A：¥RA78K0S¥INCLUDEから検索します。

③ リンカは，ライブラリ・ファイルをA：¥RA78K0S¥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A：¥TMPに作成します。

⑤ コメント文の漢字を，シフトJISコードとして解釈します。

注意　Windowsインストーラでインストールした場合には，必要な環境変数が自動的に設定されます。

11. 3　プログラム実行の中断

キー入力（CTRL＋C）により各プログラムの実行を中断できます。

バッチ・ファイル中で‘ break on ’を指定した場合は，キー入力のタイミングに関係なしに制御をOSに戻し，‘ break off ’を指定した場合には画面表示中にのみ制御をOSに戻します。そして，オープン中のすべてのテンポラリ・ファイル，出力ファイルを削除します。

11. 4　アセンブル・リストを見やすくする

-LHオプションやTITLE制御命令を使用してアセンブル・リストのヘッダにタイトルを表示します。アセンブル・リストの内容を端的に表すようなタイトルを表示しておくことでアセンブル・リストの内容がわかりやすくなります。

また，SUBTITLE制御命令を使用すればサブタイトルが表示できます。制御命令については，言語編の第4章 制御命令をお読みください。

【使 用 例】

アセンブル・リスト・ファイルのヘッダにタイトルを印字します。

```
A>ra78k0s –c9024 k0smain.asm -lhRA78K0S_MAINROUTINE
```

K0SMAIN.PRNを参照します。

```
                                                    Date:xx xxx xxxx Page:  1
78K/0S Series Assembler Ex.xx RA78K0S_MAINROUTINE
                                                    └── タイトル

Command: -c9024 k0smain.asm -lhRA78K0S_MAINROUTINE
Para-file:
In-file: K0SMAIN.ASM
Obj-file:K0SMAIN.REL
Prn-file:K0SMAIN.PRN

   Assemble list

 ALNO STNO ADRS OBJECT      M I SOURCE STATEMENT

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

11. 5　プログラム起動時の手間を省く

11. 5. 1　ソース・プログラムに制御命令を記述する

アセンブラ起動時に常に指定するオプションと同一の機能を持つ制御命令は，あらかじめソース・プログラム中で指定しておきます。これにより，アセンブラを起動するたびにオプションを指定する必要がなくなります。

【使 用 例】

```
$ PROCESSOR (9024)  ┐
$ XREF              ┘ 制御命令

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

         ⋮
```

11. 5. 2　PM plusを使用する

RA78K0Sの各プログラムのオプションは，PM plus上でプロジェクト・ファイル（.PRJ）に自動的に保存されます。2回目以降のビルド（MAKE）では，保存されたオプションが使用されますので，毎回オプションを指定する必要がなくなります。

11. 5. 3　パラメータ・ファイルやサブコマンド・ファイルを作成する

プログラム（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することで，オブジェクト・モジュールのライブラリ化が容易にできます。

【使用例1】

パラメータ・ファイルを使用してアセンブルします。

パラメータ・ファイルK0SMAIN.PRAの内容

```
;parameter file
k0smain.asm -osample.rel -g
-psample.prn
```

コマンド行には，次のように入力します。

```
A>ra78k0s –fk0smain.pra
```

【使用例2】

パラメータ・ファイルを使用してアセンブルします。

パラメータ・ファイルK0S.SLBの内容

```
;
;library creation command
;
create k0s.lib
;
add k0s.lib k0smain.rel &
k0ssub.rel
;
exit
```

コマンド行には，次のように入力します。

```
A>lb78k0s <k0s.slb
```

11. 6　オブジェクト・モジュールのライブラリ化

アセンブラおよびリンカは，1つの出力モジュールを1つのファイルに作成します。したがって，モジュールの数が多い場合はファイルの数も増加します。このため，複数のモジュールを1つのファイルにまとめる機能を用意しました。これをモジュールのライブラリ化と呼び，ライブラリ化されたファイルをライブラリ・ファイルと呼びます。

ライブラリ・ファイルは，リンカに入力できます。したがって，モジュラ・プログラミングを行った場合，共通的なモジュールをライブラリ・ファイル化として作成しておけば，ファイル管理の面でも操作性の面においても効率がよくなります。

第12章　エラー・メッセージ

この章では，アセンブラ・パッケージ（構造化アセンブラ，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）の出力するエラー・メッセージの原因，ユーザの処置などについて説明します。

12. 1　エラー・メッセージの概要

RA78K0Sのエラー・メッセージは次の3種類にレベル分けしています。

(1) アボート・エラー（A×××）

処理続行が不可能なエラーが発生したため，ただちに処理を終了（中断）します。

なお，コマンド行のアボート・エラーに関しては，他のコマンド行のエラーを検出してから処理を終了します。

(2) フェータル・エラー（F×××）

実行プログラム・エラーが発生したため，他のエラーを検出後，出力オブジェクトを生成せずに処理を終了（中断）します。

なお，フェータル・エラー時に出力オブジェクトを生成しないことを明示するため，同名のオブジェクトが存在する場合は消去して終了します。

(3) ワーニング・エラー（W×××）

ユーザが意図したものとは異なる可能性がありますが，出力オブジェクトを生成します。

備考　対話形式の実行プログラムでは，アボート・エラーの発生以外はすべて正常終了とします。

RA78K0Sのエラー・メッセージは次のように分類されています。

次頁よりRA78K0Sのエラー・メッセージについて説明します。

- A0××…コマンド行解析部のエラー
- A9××…ファイル，システムに関するエラー
- A1××…その他のアボート・エラー
- F2××…文の記述に関するエラー
- F3××…式に関するエラー
- F4××…シンボルに関するエラー
- F5××…セグメントに関するエラー
- F6××…制御命令，マクロに関するエラー
- W7××…各種ワーニング・エラー

12. 2　構造化アセンブラのエラー・メッセージ

表12－1　構造化アセンブラ・エラー・メッセージ（1/5）

A001	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。
A002	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。
A004	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。
A005	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。
A006	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。
A008	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。
A009	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。
A010	メッセージ	Directory not found　ファイル名’
	原　因	出力ファイル名中に存在しないドライブまたはディレクトリが含まれています。
	ユーザの処置	存在するドライブおよびディレクトリ名を指定してください。
A011	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。
A012	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。
A014	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。
A015	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。
A016	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。
A017	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。

表12－1　構造化アセンブラ・エラー・メッセージ（2/5）

A018	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。
A019	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。
A020	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。
A021	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。
A101	メッセージ	Open/read/write/close error on　ファイル名’
	原　因	ファイルの入出力にエラーがあり，オープン，リード，ライト，クローズが正常にできません。
	ユーザの処置	正しいファイル名を指定してください。
A102	メッセージ	Can't find　ファイル名’
	原　因	インクルード・ファイルがないか，またはインクルード・ファイル名と入力ファイル名，出力ファイル名が重複しています。
	ユーザの処置	正しいパス，ディレクトリ，ファイル名を指定してください。
A103	メッセージ	Illegal include file　ファイル名’
	原　因	インクルード・ファイルに不当なものが指定されました。
	ユーザの処置	正しいファイルを指定してください。
A104	メッセージ	Illegal(-sc) character
	原　因	-SCオプションで，シンボルとして使用できない文字が指定されました。
	ユーザの処置	正しい文字を指定してください。
A105	メッセージ	Can't define the reserved symbol
	原　因	-Dオプションで予約語を指定しました。
	ユーザの処置	-Dオプションで予約語を指定しないでください。
A106	メッセージ	Duplicate PROCESSOR control
	原　因	ソース・ファイル中でPROCESSOR制御命令を重複指定しました。 -Cオプションと異なる品種が指定されました。
	ユーザの処置	PROCESSOR制御命令の指定は1回にしてください。 品種名を修正してください。
A107	メッセージ	No processor specified
	原　因	デバイス種別の指定がありません。
	ユーザの処置	デバイス種別を指定してください。
A108	メッセージ	Illegal processor type specified
	原　因	ソース・ファイル中のPROCESSOR制御命令で，デバイス種別の指定が誤っています。
	ユーザの処置	正しいデバイス種別を指定してください。
A109	メッセージ	Illegal processor type specified -C
	原　因	-Cオプションで，デバイス種別の指定が誤っています。
	ユーザの処置	正しいデバイス種別を指定してください。

表12－1　構造化アセンブラ・エラー・メッセージ（3/5）

A110	メッセージ	Can't use this control outside module header
	原　因	ソース・モジュール・ヘッダに記述すべき制御命令が，通常のソース行に記述されています。
	ユーザの処置	制御命令をソース・モジュール・ヘッダに記述してください。
A111	メッセージ	Syntax error in module header
	原　因	ソース・モジュール・ヘッダに記述されている制御命令の記述形式に誤りがあります。
	ユーザの処置	正しい形式で制御命令を記述してください。
A112	メッセージ	Structured assembler preprocessor internal error
	原　因	構造化アセンブラ・プリプロセッサ自身に内部エラーが発生しました。
	ユーザの処置	NECエレクトロニクスに連絡してください。

F201	メッセージ	Illegal #ELSE/#ENDIF
	原　因	#ELSE, #ENDIF文の記述位置に誤りがあります。
	ユーザの処置	#ELSE, #ENDIF文を正しい位置に記述してください。
F202	メッセージ	Illegal #ENDCALLT
	原　因	#ENDCALLT文の記述位置に誤りがあります。
	ユーザの処置	#ENDCALLT文を正しい位置に記述してください。
F203	メッセージ	Missing #ENDIF
	原　因	#ENDIF文がありません。
	ユーザの処置	#ENDIF文を正しい位置に記述してください。
F204	メッセージ	Missing #ENDCALLT
	原　因	#ENDCALLT文がありません。
	ユーザの処置	#ENDCALLT文を正しい位置に記述してください。
F205	メッセージ	Too many #DEFCALLT definition
	原　因	callt命令変換パターンの登録数が制限を越えています。
	ユーザの処置	#defcalltの登録数を減らしてください。
F206	メッセージ	Too many CALL instructions
	原　因	#DEFCALLT～#ENDCALLTで定義する命令が多すぎます。
	ユーザの処置	#DEFCALLT～#ENDCALLTで定義する命令を1つにしてください。
F207	メッセージ	Duplicate definition
	原　因	同一変換パターンの再定義をしました。
	ユーザの処置	#DEFCALLTの登録を修正してください。
F208	メッセージ	Symbol table overflow
	原　因	シンボル数が制限を越えています。
	ユーザの処置	シンボル数を減らしてください。
F209	メッセージ	Syntax error
	原　因	記述した文の構文に誤りがあります。
	ユーザの処置	正しい構文を記述してください。
F210	メッセージ	Nest level error
	原　因	ネスティングに誤りがあります（オーバフロー，ネストの対など）。
	ユーザの処置	正しい制御文を記述してください。
F211	メッセージ	Too many characters in a line
	原　因	1行の長さが制限を越えています。
	ユーザの処置	1行の長さを218文字以内にしてください。

表12－1　構造化アセンブラ・エラー・メッセージ（4/5）

F212	メッセージ	Too many include files
	原　因	インクルード・ファイルの中にインクルード疑似命令があります。
	ユーザの処置	インクルード・ファイル中にインクルード疑似命令を指定しないでください。
F214	メッセージ	Illegal BREAK
	原　因	BREAK文の記述位置に誤りがあります。
	ユーザの処置	BREAK文を正しい位置に記述してください。
F215	メッセージ	Illegal CONTINUE
	原　因	CONTINUE文の記述位置に誤りがあります。
	ユーザの処置	CONTINUE文を正しい位置に記述してください。
F216	メッセージ	Illegal CASE/DEFAULT/ENDS
	原　因	CASE/DEFAULT/ENDS文の記述位置に誤りがあります。
	ユーザの処置	CASE/DEFAULT/ENDS文を正しい位置に記述してください。
F217	メッセージ	Illegal ELSEIF/ELSE/ENDIF
	原　因	ELSEIF/ELSE/ENDIF文の記述位置に誤りがあります。
	ユーザの処置	ELSEIF/ELSE/ENDIF文を正しい位置に記述してください。
F218	メッセージ	Illegal NEXT
	原　因	NEXT文の記述位置に誤りがあります。
	ユーザの処置	NEXT文を正しい位置に記述してください。
F219	メッセージ	Illegal ENDW
	原　因	ENDW文の記述位置に誤りがあります。
	ユーザの処置	ENDW文を正しい位置に記述してください。
F220	メッセージ	Illegal UNTIL/UNTIL_BIT
	原　因	UNTIL, UNTIL_BIT文の記述位置に誤りがあります。
	ユーザの処置	UNTIL, UNTIL_BIT文を正しい位置に記述してください。
F221	メッセージ	Missing ENDIF
	原　因	ENDIF文がありません。
	ユーザの処置	ENDIF文を正しい位置に記述してください。
F222	メッセージ	Missing ENDS
	原　因	ENDS文がありません。
	ユーザの処置	ENDS文を正しい位置に記述してください。
F223	メッセージ	Missing ENDW
	原　因	ENDW文がありません。
	ユーザの処置	ENDW文を正しい位置に記述してください。
F224	メッセージ	Missing NEXT
	原　因	NEXT文がありません。
	ユーザの処置	NEXT文を正しい位置に記述してください。
F225	メッセージ	Missing UNTIL/UNTIL_BIT
	原　因	UNTILまたはUNTIL_BIT文がありません。
	ユーザの処置	UNTILまたはUNTIL_BIT文を正しい位置に記述してください。
F226	メッセージ	Illegal character in a line
	原　因	ソース・ライン中に不正な文字の記述があります。
	ユーザの処置	ソース・ライン中の不正な文字の記述を削除してください。

表12－1　構造化アセンブラ・エラー・メッセージ（5/5）

F227	**メッセージ**	Illegal operand in a line
	原　因	**代入式，比較条件式のデータ・サイズに誤りがあります。**
	ユーザの処置	**正しいデータ・サイズを指定してください。**
F228	**メッセージ**	Illegal SFR access in operand
	原　因	**代入式にアクセスできない**sfr**シンボルを記述しました。**
	ユーザの処置	sfr**シンボルのアクセス状態を確認して，正しい**sfr**シンボルを記述してください。**
F229	**メッセージ**	This symbol is reserved **“ シンボル名 ”**
	原　因	**使用したシンボルが予約語になっています。**
	ユーザの処置	**シンボル名を変更してください。**

W301	**メッセージ**	Symbol redefinition
	原　因	#define**文によりシンボルの再定義をしています。**
	プログラムの処置	**最後に定義されたシンボルを有効とします。**
	ユーザの処置	**最初に定義したシンボルを有効にしたい場合は構文を修正してください。**
W302	**メッセージ**	Duplicate PROCESSOR option and control
	原　因	-C**オプションと**$PROCESSOR**制御命令で異なる品種を指定しています。**
	プログラムの処置	-C**オプションで指定された品種を有効とし，**$PROCESSOR**制御命令は無視します。**
	ユーザの処置	-C**オプションで指定した品種で正しかったことを確認してください。**

12. 3　アセンブラのエラー・メッセージ

表12－2　アセンブラ・エラー・メッセージ（1/8）

A101	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。
A102	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。
A103	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。
A104	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。
A105	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。
A106	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。
A107	メッセージ	Default segment ? CSEG is already used
	原　因	セグメント定義省略時にディフォールト・セグメントを定義しようとしました。
A108	メッセージ	Synbol 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。
A109	メッセージ	Too many DS
	原　因	DS疑似命令が多くあるために，セグメント内のオブジェクト・コードの間隔が空きすぎて，オブジェクト・ファイルに情報を出力できません。
A110	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。
A111	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。
A112	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。
A114	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。
A115	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。

表12－2　アセンブラ・エラー・メッセージ（2/8）

F201	メッセージ	Syntax error
	原　因	文の記述形式がまちがっています。
F202	メッセージ	Illegal operand
	原　因	オペランドの記述が不正です。
F203	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。
F204	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。
F205	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。
F206	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。
F207	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。

F301	メッセージ	Too complex expression
	原　因	式が複雑すぎます。
F302	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。
F303	メッセージ	Illegal expression
	原　因	式の記述形式がまちがっています。
F304	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。
F305	メッセージ	Too long string constant
	原　因	文字定義の長さの制限（4文字）を越えています。
F306	メッセージ	Illegal number
	原　因	記述された数値がまちがっています。
F307	メッセージ	Division by zero
	原　因	0で除算をしています。
F308	メッセージ	Too large number
	原　因	定数の値が16ビットを越えています。
F309	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。
F310	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。
F311	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。
F312	メッセージ	Operand out of range（byte）
	原　因	オペランドの値が範囲（00H-FFH）を越えたか，あるいはオペランド中のbyteの値が範囲（－128～＋127）を越えました。
F313	メッセージ	Operand out of range（addr5）
	原　因	addr5として記述可能な範囲（40H-7EH）を越えました。

表12－2　アセンブラ・エラー・メッセージ（3/8）

F314	メッセージ	Operand out of range（addr11）
	原　因	addr11として記述可能な範囲（800H-FFFH）を越えました。
F315	メッセージ	Operand out of range（saddr）
	原　因	saddrとして記述可能な範囲（0FE20H-0FF1FH）を越えました。
F316	メッセージ	Operand out of range（addr16）
	原　因	addr16として記述可能な範囲（対象デバイスによって異なる）を越えました。
F317	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。
F318	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SFR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。

F401	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。
F402	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。
F403	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を定義しました。
	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。
F404	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。
F405	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。
F406	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。
F407	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。
F408	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。
F409	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。
F410	メッセージ	Phase error
	原　因	アセンブル中にシンボルの値が変化しました（たとえば，BR疑似命令の最適化処理によって変化したレーベルをオペランドの中に用いて定義したEQUシンボルなど）
F411	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。

表12－2　アセンブラ・エラー・メッセージ（4/8）

F502	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。
F503	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。
F504	メッセージ	Too many segments
	原　因	定義できるセグメントの制限（100個）を越えています。
F505	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。
F506	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。
F507	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。
F508	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。
F509	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。
F510	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て範囲を越えました。
F511	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。
F512	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。

F601	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（2レベル）を越えています。
F602	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。
F603	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5個以内）を越えています。
F604	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。
F605	メッセージ	Needless ELSE statement exists
	原　因	必要のないところにELSE文字が存在しています。
F606	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。
F607	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。
F608	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。
F609	メッセージ	Illegal ELSEIF statement
	原　因	ELSE文のあとにELSEIFまたは_ELSEIF文が記述されています。

表12－2　アセンブラ・エラー・メッセージ（5/8）

F610	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。
F611	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。
F612	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。
F613	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。
F614	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。
F615	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。
F616	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。
F617	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。
F618	メッセージ	Illegal syntax of ENDM
	原　因	ENDM文の記述が間違っています。
F619	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。
F620	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。
F621	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。
F622	メッセージ	Illegal syntax of EXITM
	原　因	EXITM文の記述に誤りがあります。
F623	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。
F624	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。
F625	メッセージ	Unexpected ENDM
	原　因	余分なENDMが現れました。
F626	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。
F627	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。

表12－2　アセンブラ・エラー・メッセージ（6/8）

W702	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。
W703	メッセージ	Multiple defined module name
	原　因	NAME疑似命令が二度以上定義されています。
	プログラムの処理	そのNAME疑似命令を無効として，すでに定義されたモジュール名を有効とします。
W704	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。
W705	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。
W706	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。
W707	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。
W708	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。
W709	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。
W710	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。
W711	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。
W712	メッセージ	Insufficient cross-reference work area
	原　因	クロスレファレンス・リストの出力処理を行うためのメモリが不足しています。
	プログラムの処理	クロスレファレンス・リストの出力せずに，処理を続けます。

表12－2　アセンブラ・エラー・メッセージ（7/8）

A901	メッセージ	Can't open source file　ファイル名’
	原　因	ソース・ファイルがオープンできません。
A902	メッセージ	Can't open parameter file　ファイル名’
	原　因	パラメータ・ファイルがオープンできません。
A903	メッセージ	Can't open include file　ファイル名’
	原　因	インクルード・ファイルがオープンできません。
A904	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。
A905	メッセージ	Can't open overlay file　ファイル名’
	原　因	オーバレイ・ファイルがオープンできません。
	ユーザの処置	オーバレイ・ファイルがアセンブラの実行形式と同じディレクトリにあるかどうかを調べてください。
A906	メッセージ	Illegal overlay file　ファイル名’
	原　因	オーバレイ・ファイルの内容が不正です。
A907	メッセージ	Can't open object file　ファイル名’
	原　因	オブジェクト・ファイルがオープンできません。
	ユーザの処置	ディレクトリに空き領域のあるディスクを使用してください。
A908	メッセージ	Can't open print file　ファイル名’
	原　因	アセンブル・リスト・ファイルがオープンできません。
	ユーザの処置	ディレクトリに空き領域のあるディスクを使用してください。
A909	メッセージ	Can't open error list file　ファイル名’
	原　因	エラー・リスト・ファイルがオープンできません。
	ユーザの処置	ディレクトリに空き領域のあるディスクを使用してください。
A910	メッセージ	Can't open temporary file　ファイル名’
	原　因	テンポラリ・ファイルがオープンできません。
	ユーザの処置	ディレクトリに空き領域のあるディスクを使用してください。
A911	メッセージ	System error
	原　因	システム・エラーが発生しました。
	ユーザの処置	実行環境を確かめて，もう一度アセンブルを実行してください。
A912	メッセージ	Can’t set Control-C
	原　因	アセンブラ実行中止のためのCTRL＋Cの設定ができません。
	ユーザの処置	実行環境を確かめ，もう一度アセンブルしてください。
A913	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。
A914	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。
A915	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。
A916	メッセージ	Can't read overlay file　ファイル名’
	原　因	オーバレイ・ファイルにファイルI/Oエラーが発生しました。

表12－2　アセンブラ・エラー・メッセージ（8/8）

A917	**メッセージ**	Can't write object file　**ファイル名’**
	原　因	**オブジェクト・ファイルにファイル**I/O**エラーが発生しました。**
	ユーザの処置	**オブジェクト・ファイルを他のディレクトリに出力するか，指定したディスクに空き領域を作ってください。**
A918	**メッセージ**	Can't write print file　**ファイル名’**
	原　因	**アセンブル・リスト・ファイルにファイル**I/O**エラーが発生しました。**
	ユーザの処置	**アセンブル・リスト・ファイルを他のディレクトリに出力するか，指定したディスクに空きを領域を作ってください。**
A919	**メッセージ**	Can't write error list file　**ファイル名’**
	原　因	**エラー・リスト・ファイルにファイル**I/O**エラーが発生しました。**
	ユーザの処置	**エラー・リスト・ファイルを他のディレクトリに出力するか，指定したディスクに空き領域を作ってください。**
A920	**メッセージ**	Can't read/write temporary file　**ファイル名’**
	原　因	**テンポラリ・ファイルにファイル**I/O**エラーが発生しました。**
	ユーザの処置	**テンポラリ・ファイルを他のディレクトリに出力するか，指定したディスクに空き領域を作ってください。**
A921	**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。**
A922	**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。**
A923	**メッセージ**	Insufficient memory for macro in hostmachine
	原　因	**マクロ処理の途中で内部メモリが不足しました。**
	ユーザの処置	**マクロ定義を少なくしてください。**

12. 4　リンカのエラー・メッセージ

表12－3　リンカ・エラー・メッセージ（1/6）

A101	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。
F102	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。
A103	メッセージ	ファイル名’Illegal processor type
	原　因	アセンブルまたはコンパイルの対象デバイスが，このリンカの対象デバイスではありません。
	ユーザの処置	オブジェクト・モジュールファイルが正しいことを確認してください。リンカの扱えるアセンブルまたはコンパイルの対象デバイスを確認してください。また，オーバレイ・ファイルが正しいバージョンであることを確認してください（リンカは，アセンブラのオーバレイ・ファイルの一部を参照して対象デバイス固有の情報を得ています）。
A104	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。
W105	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。
A106	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。
F107	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。
F108	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。
F109	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。
F110	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。
F111	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。
F112	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まちがっています。
A113	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。
F114	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。
F115	メッセージ	Too large value（up to 65535/0FFFFH）
	原　因	ディレクティブ中で65535（0FFFFH）を越える値が記述されました。

表12－3　リンカ・エラー・メッセージ（2/6）

F116	**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，**65535（0FFFFH）**を越えました。**

F201	**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）**
F202	**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。**
A203	**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）。**
F204	**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。**
F205	**メッセージ**	Name　**名前’** in directive has bad attribute
	原　因	**ディレクティブのセグメント名，メモリ領域名，メモリ空間名のいずれかに指定できないものを記述しています（メモリ領域名を指定すべきところにメモリ空間名を指定したなど）。**
F206	**メッセージ**	Segment　**セグメント名’** can’t allocate to memory-ignored
	原　因	**セグメントをメモリ領域に割り付けることができません（セグメントを割り付けるのに十分なメモリ領域が存在しません）。**
F207	**メッセージ**	Segment　**セグメント名’** has illegal segment type
	原　因	**このセグメントの型情報が不正です。**
F208	**メッセージ**	Segment　**セグメント名’** may not change attribute
	原　因	**アセンブル時に再配置属性を**　AT xxxxH’**としたセグメント，または**ORG**疑似命令により作成したセグメントに対し，ディレクティブで結合型を変更しようとしました。**
F209	**メッセージ**	Segment　**セグメント名’** may not change arrangement
	原　因	**アセンブル時に再配置属性を**　AT xxxxH’**としたセグメント，または**ORG**疑似命令により作成したセグメントに対し，ディレクティブで配置アドレスを変更しようとしました。**
	ユーザの処置	**リンク時に結合型を指定するセグメントに対しては，アセンブル時に配置アドレスを指定しないでください。**
F210	**メッセージ**	Segment　**セグメント名’** is not exist-ignored
	原　因	**ディレクティブで指定されたセグメントが存在しません。**
F211	**メッセージ**	Bank type mismatch　**シンボル名’** in file　**ファイル名’** –ignored
	原　因	**シンボルのバンク番号の指定に矛盾があります。**
	ユーザの処置	**シンボルのバンク番号が正しいことを確認してください。**

表12－3　リンカ・エラー・メッセージ（3/6）

F301	メッセージ	Relocatable object code address out of range（file　ファイル名’，segment　セグメント名’，address xxxxH, type　アドレシング・タイプ’）
	原　因	入力したオブジェクト・モジュール・ファイル・中に含まれるリロケータブル・オブジェクト・コードの修正情報が，オブジェクト・コードの存在しないアドレスに対して出力されています（リロケーション・エントリのアドレスが，オリジン・データの範囲外にあります）。
	ユーザの処置	シンボルの参照が正しいことを確認してください。
F302	メッセージ	Illegal symbol index in line number（file　ファイル名’，segment　セグメント名’）
	原　因	入力したオブジェクト・モジュール・ファイル中に含まれるディバグ用行番号情報に誤りがあり，シンボル情報を正しく参照していません。行番号のインデクスとシンボル・インデクスの対応がとれていません。
F303	メッセージ	Can't find symbol index in relocatable object code（file　ファイル名’，segment　セグメント名’，address xxxxH, type　アドレシング・タイプ’）
	原　因	入力したオブジェクト・モジュール・ファイル中に含まれるリロケータブル・コードの修正情報に誤りがあり，シンボル情報を正しく参照していません。リロケーション・エントリとシンボル・インデクスの対応がとれていません。
	ユーザの処置	シンボル，変数などの参照方法が正しいことを確認してください。
F304	メッセージ	Operand out of range（segment　セグメント名’，address xxxxH, type　アドレシング・タイプ’）
	原　因	リロケータブル・オブジェクト・コードの解決に用いているオペランド値が，命令に対応したオペランドの値の範囲を越えています。
	ユーザの処置	オペランド値をアドレシング・タイプごとに定められているオペランドの範囲に納まるようにソース・プログラムを記述してください。
F305	メッセージ	Even value expected（segment　セグメント名’，address xxxxH, type　アドレシング・タイプ’）
	原　因	calltまたはsaddrpアドレシングのリロケータブル・オブジェクト・コード解決に用いているオペランド値が奇数になりました（calltまたはsaddrpアドレシングのオペランドは偶数でなければなりません）。
F306	メッセージ	A multiple of 4 value expected（segment　セグメント名’, address xxxxH, type　アドレシング・タイプ’）
	原　因	saddrアドレシングのリロケータブル・オブジェクト・コードの解決に用いているオペランド値が4の倍数になりませんでした。

注意　F301-F306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。

表12－3　リンカ・エラー・メッセージ（4/6）

A401	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。
A402	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。
F403	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。
F404	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。
F405	メッセージ	Undefined symbol　シンボル名’ in file　ファイル名’
	原　因	ファイルでEXTRN宣言されているシンボルは，他のファイルでPUBLIC宣言されていません。
W406	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）。
W407	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）。
F408	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。
F409	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。
F410	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。
W411	メッセージ	Different REL type in file　ファイル名’
	原　因 ユーザの処置	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。 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。
F415	メッセージ	-QD/QF/etc. and Not-QD/QF/etc. REL are mixed
	原　因	プログラム全体で同じ指定でなければいけないコンパイラの最適化オプションに関して，異なる指定をしたオブジェクト・ファイルが入力されました。同じ指定でコンパイルし直してください。
W416	メッセージ	Multiple CAP/NOCAP are in file　ファイル名（オプション）’ First defined 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。

表12－3　リンカ・エラー・メッセージ（5/6）

W417	**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**（OC78K0S, ST78K0S, RA78K0S）**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。**
W418	**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，旧タイプの**（DF**非対応）をリンクすると必ず出力します。**
F420	**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**（CC78K0S, ST78K0S, RA78K0S）**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。**
F425	**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。**
F431	**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**値を持っています。**

A901	**メッセージ**	Can't open overlay file　**ファイル名’**
	原　因	**オーバレイ・ファイルがオープンできません。**
	ユーザの処置	**オーバレイ・ファイルが正しいディレクトリ（実行形式プログラムがあるディレクトリ）にあることを確認してください。**
A902	**メッセージ**	file　**ファイル名’** file not found
	原　因	**指定されたライブラリ・ファイルをオープンできません。**
A903	**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。**
A904	**メッセージ**	Can't open output file　**ファイル名’**
	原　因	**出力ファイルをオープンできません。**
	ユーザの処置	**出力ファイルを作成しようとしたディスクの状態（空き容量，メディアの状態など）を確認してください。**
A905	**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。**
A906	**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。**
A907	**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。**
A908	**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。**

表12－3　リンカ・エラー・メッセージ（6/6）

A909	メッセージ	Can't read device file　デバイス・ファイル名’
	原　因	-Cオプション，または$PROCESSOR制御命令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。

12. 5　オブジェクト・コンバータのエラー・メッセージ

表12－4　オブジェクト・コンバータ・エラー・メッセージ（1/2）

A006	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。
A100	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。
A101	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。
A103	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。
A104	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。
A105	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。
A106	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。
A107	メッセージ	Can’t specify -U option for ROMless devide
	原　因	内部ROMのない品種に充填オプション（-U）を指定しています。

F200	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。

表12－4　オブジェクト・コンバータ・エラー・メッセージ（2/2）

F201	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。

W300	メッセージ	xxxxxxH-yyyyyyH overlapped
	原　因	xxxxxxHからyyyyyyHまでのアドレスに対するオブジェクトが重複して出力されています。
W301	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CSEG内にDB/DWが記述されている場合は，DSに変更されるか、DSEG内でDB/DW命令を記述するようにしてください。

A900	メッセージ	Can't open file　ファイル名’
	原　因	ファイルがオープンできません。
A901	メッセージ	Can't close file　ファイル名’
	原　因	ファイルがクローズできません。
A902	メッセージ	Can't read file　ファイル名’
	原　因	ファイルが正しく読めません。
A903	メッセージ	Can't access file　ファイル名’
	原　因	ファイルが正しく読み込みまたは書き込みができません。
A904	メッセージ	Can't write file　ファイル名’
	原　因	出力ファイルに正しくデータが書き込めません。

12. 6　ライブラリアンのエラー・メッセージ

表12－5　ライブラリアンのエラー・メッセージ（1/3）

A001	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。
A002	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。
A003	メッセージ	Unrecognized string　'???'
	原　因	会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。
A004	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。
A005	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。
A006	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。
A007	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。
A008	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。
A009	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。
A010	メッセージ	Directory not found　'ファイル名'
	原　因	出力ファイル名中に存在しないドライブ，またはディレクトリが含まれています。
A011	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。
A012	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。
A013	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。
A014	メッセージ	Out of range　'オプション'
	原　因	指定値が範囲外です。
A015	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。
A016	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。
A017	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。
A018	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。
A019	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。
A020	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。
A021	メッセージ	Memory allocation failed
	原　因	メモリ・アロックに失敗しました。

表12－5　ライブラリアンのエラー・メッセージ（2/3）

A022	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。
A023	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。
A024	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。
A025	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。
A026	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。

A100	メッセージ	Internal error
	原　因	内部エラーが発生しました。
F101	メッセージ	Invalid sub command
	原　因	サブコマンド名が誤っています。
F102	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。
F103	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。
F104	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。
F105	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。
F106	メッセージ	Module already exists（module：ファイル名）
	原　因	同名のモジュールが，すでに更新ライブラリ・ファイルまたは他の入力ファイル内に存在しています。
F107	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。
F108	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。
F109	メッセージ	Public symbol already exists（symol：シンボル名）
	原　因	同名の外部定義シンボル名が，すでに更新ライブラリ・ファイルまたは他の入力ファイル内に存在しています。
F110	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。
F111	メッセージ	Illegal file format（file：ファイル名）
	原　因	更新ライブラリ・ファイルまたは，他の入力ファイルのフォーマットが異常です。
F112	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。
F113	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。

表12－5　ライブラリアンのエラー・メッセージ（3/3）

F114	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。
F115	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。
F116	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。
F117	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。
F118	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。

A901	メッセージ	File open error（file：ファイル名）
	原　因	ファイルがオープンできません。
A902	メッセージ	File read error（file：ファイル名）
	原　因	ファイルが正しく読めません。
A903	メッセージ	File write error（file：ファイル名）
	原　因	ファイルに正しくデータが書き込めません。
A904	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。
A905	メッセージ	File close error（file：ファイル名）
	原　因	ファイルがクローズできません。

12. 7　リスト・コンバータのエラー・メッセージ

表12－6　リスト・コンバータのエラー・メッセージ（1/2）

A101	メッセージ	File is not 78K/0S　ファイル名’
	原　因	入力ファイル名が78K0Sのものではありません。
W101	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。
A102	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。
W102	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。
A103	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。
W103	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。
A104	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。
W104	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。
A105	メッセージ	Segment name is not found in load list 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。
W105	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。
A106	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。
A107	メッセージ	Not enough memory
	原　因	作業用メモリが足りません。
A108	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。
A109	メッセージ	Overlay file can not open　パス名’
	原　因	アセンブラのオーバーレイ・ファイルがオープンできません。
A110	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。

表12－6　リスト・コンバータのエラー・メッセージ（2/2）

A901	メッセージ	File open error has occured　ファイル名 ’
	原　因	ファイルがオープンできません。
A902	メッセージ	File read error has occured　ファイル名 ’
	原　因	ファイルが正しく読めません。
A903	メッセージ	File write error has occured　ファイル名 ’
	原　因	ファイルに正しくデータが書き込めません。
A904	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。
A999	メッセージ	Internal error
	原　因	プログラム内部エラーです。

12. 8　PM plusのエラー・メッセージ

PM plusのヘルプに記載されていないエラー・メッセージについて，説明します。PM plusのその他のエラー・メッセージについては，PM plusのオンライン・ヘルプを参照してください。

①　構造化アセンブラ（ST78K0S）用DLLの表示するエラー・メッセージ

!	日本語メッセージ	ST78K0S.EXEがPATH環境変数で示されるディレクトリに登録されていません。
	英語メッセージ	Cannot find ST78K0S.EXE shown in environment variable PATH.
	原　因	ST78K0S.EXE実行形式がPATH環境変数で示されるディレクトリに登録されていません。
	ユーザの処置	ST78K0S.EXEなどのST78K0S関連ファイルをPATH環境変数で示されるディレクトリに入れてください。
	ボタン	“ OK ”…メッセージを閉じます。
×	日本語メッセージ	メモリを確保できません。
	英語メッセージ	Not enough memory.
	原　因	メモリが足りません。
	ユーザの処置	他のアプリケーションを終了させたあとで，再度試してください。
	ボタン	“ OK ”…メッセージを閉じます。
×	日本語メッセージ	メモリをロックできません。
	英語メッセージ	Cannot lock the memory.
	原　因	メモリが足りないか，Windowsシステムが壊れている可能性があります。
	ユーザの処置	他のアプリケーションを終了させるか，Windowsを再起動させたあとで，再度試してください。
	ボタン	“ OK ”…メッセージを閉じます。

②　アセンブラ（RA78K0S）用DLLの表示するエラー・メッセージ

!	日本語メッセージ	RA78K0S.EXEがPATH環境変数で示されるディレクトリに登録されていません。
	英語メッセージ	Cannot find RA78K0S.EXE shown in environment variable PATH.
	原　因	RA78K0S.EXE実行形式が，PATH環境変数で示されるディレクトリに登録されていません。
	ユーザの処置	RA78K0S.EXEなどのRA78K0S関連ファイルを，PATH環境変数で示されるディレクトリに入れてください。
	ボタン	“ OK ”…メッセージを閉じます。
×	日本語メッセージ	メモリを確保できません。
	英語メッセージ	Not enough memory.
	原　因	メモリが足りません。
	ユーザの処置	他のアプリケーションを終了させたあとで，再度試してください。
	ボタン	“ OK ”…メッセージを閉じます。
×	日本語メッセージ	メモリをロックできません。
	英語メッセージ	Cannot lock the memory.
	原　因	メモリが足りないか，Windowsシステムが壊れている可能性があります。
	ユーザの処置	他のアプリケーションを終了させるか，Windowsを再起動させたあとで，再度試してください。
	ボタン	“ OK ”…メッセージを閉じます。

③　リンカ（LK78K0S）用DLLの表示するエラー・メッセージ

!	日本語メッセージ	LK78K0S.EXEが，PATH環境変数で示されるディレクトリに登録されていません。
	英語メッセージ	Cannot find LK78K0S.EXE shown in environment variable PATH.
	原　因	LK78K0S.EXE実行形式が，PATH環境変数で示されるディレクトリに登録されていません。
	ユーザの処置	LK78K0S.EXEなどのLK78K0S関連ファイルを，PATH環境変数で示されるディレクトリに入れてください。
	ボタン	“ OK ”…メッセージを閉じます。
×	日本語メッセージ	メモリを確保できません。
	英語メッセージ	Not enough memory.
	原　因	メモリが足りません。
	ユーザの処置	他のアプリケーションを終了させたあとで，再度試してください。
	ボタン	“ OK ”…メッセージを閉じます。
×	日本語メッセージ	メモリをロックできません。
	英語メッセージ	Cannot lock the memory.
	原　因	メモリが足りないか，Windowsシステムが壊れている可能性があります。
	ユーザの処置	他のアプリケーションを終了させるか，Windowsを再起動させたあとで，再度試してください。
	ボタン	“ OK ”…メッセージを閉じます。

④　オブジェクト・コンバータ（OC78K0S）用DLLの表示するエラー・メッセージ

!	日本語メッセージ	OC78K0S.EXEが，PATH環境変数で示されるディレクトリに登録されていません。
	英語メッセージ	Cannot find OC78K0S.EXE shown in environment variable PATH.
	原　因	OC78K0S.EXE実行形式が，PATH環境変数で示されるディレクトリに登録されていません。
	ユーザの処置	OC78K0S.EXEなどのOC78K0S関連ファイルを，PATH環境変数で示されるディレクトリに入れてください。
	ボタン	“ OK ”…メッセージを閉じます。
×	日本語メッセージ	メモリを確保できません。
	英語メッセージ	Not enough memory.
	原　因	メモリが足りません。
	ユーザの処置	他のアプリケーションを終了させたあとで，再度試してください。
	ボタン	“ OK ”…メッセージを閉じます。
×	日本語メッセージ	メモリをロックできません。
	英語メッセージ	Cannot lock the memory.
	原　因	メモリが足りないか，Windowsシステムが壊れている可能性があります。
	ユーザの処置	他のアプリケーションを終了させるか，Windowsを再起動させたあとで，再度試してください。
	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

RA78K0S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　K0smain.asm

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

PUBLIC MAIN, START
EXTRN  CONVAH
EXTRN  _@STBEG

DATA   DSEG   saddr
HDTSA: DS  1
STASC: DS  2

CODE   CSEG   AT 0H
MAIN:  DW  START

       CSEG
START:

   ;chip initialize
   MOVW   AX, #_@STBEG
   MOVW   SP, AX

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

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

   BR     $$

   END
```

A. 2　K0ssub.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:
   MOV    A, [HL]
   ROR    A, 1
   ROR    A, 1
   ROR    A, 1
   ROR    A, 1
   AND    A, #0FH        ;hex upper code load
   CALL   !SASC
   MOV    B, A           ;store result

   XOR    A, A
   XCH    A, [HL]
   AND    A, #0FH        ;hex lower code load
   CALL   !SASC
   MOV    C, A           ;store result

   RET

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

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

   END
```

A. 3　test1.s

```
EXTRN   SEARCH, STABLE
EXTRN   _@STBEG
PUBLIC  MAIN, START
;*****************************************
;          String data search
;*****************************************
SDATA:
    DB  04,12H,34H,56H,78H
;
;
CODE    CSEG    AT 0H
START:  DW      MAIN

MAIN:
    MOVW    AX, #_@STBEG
    MOVW    SP, AX
    DE = #STABLE
    HL = #SDATA
    CALL  !SEARCH
    if_bit (!CY)
SLI:
        repeat
        until (forever)
    else
SERR:
        repeat
        until (forever)
    endif
    END
```

A. 4　test2.s

```
#include   “testinc.s”

PUBLIC SEARCH

   CSEG
;*****************************************************
;*  Data search                                      *
;*    input    HL     search data address           *
;*             DE     table top address             *
;*    output   CY = 1 not find                       *
;*             CY = 0 find (DE<-table address)       *
;*****************************************************

SEARCH:
    while ([DE]!= #0) (A)
       BC = #0
       A =[DE]
       C = A
       PUSH   HL
       PUSH   DE
       while ([DE]== [HL]) (A)
          DE++
          HL++
          if (C == #0) (A)

             POP DE
             POP HL
             CLR1  CY
             RET
          endif
          C--
       endw
       POP  DE
       POP  HL
       A =[DE]
       A += E
       E = A
       if (CY)
       D++
       endif
        endw
    SET1   CY
    RET
    END
```

A. 5　testinc.s

```
PUBLIC  STABLE

;**********************************
;         Data table
;**********************************

   CSEG
STABLE:
    DB  03, 12H, 34H, 78H
    DB  04, 55H, 66H, 77H, 88H
    DB  05, 12H, 34H, 56H, 78H, 10H
    DB  03, 12H, 34H, 56H
    DB  04, 12H, 34H, 0AH, 78H
    DB  04, 12H, 34H, 56H, 70H
    DB  04, 12H, 34H, 56H, 78H
    DB  01, 0ABH
    DB  02, 34H, 78H
    DB  00
```

A. 6　st.bat

```
        echo off
        cls
        set      LEVEL=0

        if "%1" == "" goto ERR_BAT

        st78k0s -C%1 test1.s
        ra78k0s test1.asm
        if errorlevel 1 set LEVEL=1
        st78k0s -C%1 test2.s
        ra78k0s test2.asm
        if errorlevel 1 set LEVEL=1
        if %LEVEL% == 1 echo Assemble error !!
        if %LEVEL% == 1 goto END

        cls
        lk78k0s test1.rel test2.rel -s -otest.lmf -ptest.map
        if errorlevel 1 echo Link error !!
        if errorlevel 1 goto END

        cls
        oc78k0s test
        if errorlevel 1 echo Object conversion error !!
        if errorlevel 1 goto END

        cls
        set LEVEL=0
        lc78k0s -ltest.lmf -rtest1.rel test1.prn
        if errorlevel 1 set LEVEL=1
        lc78k0s -ltest.lmf -rtest2.rel test2.prn
        if errorlevel 1 set LEVEL=1
        if %LEVEL% == 1 echo List conversion error !!
        if %LEVEL% == 1 goto END

        cls
        echo No error.
        goto END

:ERR_BAT

        echo Usage : st.bat chiptype

:END
        echo on
```

付録B　使用上の注意一覧

RA78K0Sを使用するときの注意事項を示します。

(1) デバイス・ファイルに関する注意事項

RA78K0S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RA78K0Sパッケージには含まれておりません。別途ご入手ください。

(2) メモリ・ディレクティブについての注意事項

各デバイスのディフォールトのメモリ領域名は，消去できません。

ディフォールトにあり，ご使用にならないメモリ領域名のサイズは0にしてください。

ディフォールトのメモリ領域名につきましては，各デバイスの**デバイス・ファイル使用上の留意点**を参照してください。

ただし，セグメントによってはディフォールトの領域に割り付けられるものもありますので，領域名を変更するときにはご注意ください。

(3) ディバグ・オプションに関する注意事項

Cコンパイラ／構造化アセンブラ・プリプロセッサでディバグ情報を出力して，コンパイル／構造化アセンブルした場合，その出力アセンブル・ソースをアセンブルするときには，ディバグ情報を出力しないようにしてください（-NGAオプションを指定してください）。ディバグ情報を出力すると，Cコンパイラ／構造化アセンブラ・ソース・レベルでディバグできないことがあります。

(4) メモリ初期化疑似命令に関連した注意事項

メモリ初期化疑似命令（DW，DB）をデータ・セグメント（DSEG）中で記述した場合でも，コードを出力します。

ROMコード発注時には，内蔵ROM以外のアドレスにコードが存在するとエラーとなりますので，注意してください。

(5) SFR名のEQU定義に関する注意事項

EQU擬似命令のオペランドにはSFR名を指定できますが，saddrr領域外のSFRの名前をPUBLICに指定した場合，アセンブル・エラーになります。

(6) C78K0Sに関連した注意事項

CC78K0Sが出力したアセンブラ・ソースをアセンブルして使用し，Cソース・レベル・ディバグを行う場合，何点かの注意事項があります。

詳細は，C**コンパイラ・パッケージの製品添付文書（使用上の留意点）**を参照してください。

（7）D78K0S-NS, SM78K0Sに関する注意事項

ID78K0S-NS, SM78K0Sでディバグを行う場合にシンボル数，ソース行数の制限に関して，ID78K0S-NS, SM78K0Sの制限以内でご使用ください。

詳細は，**ディバッガ／シミュレータの製品添付文書（使用上の留意点）**を参照してください。

（8）ネットワーク使用時の注意事項

一時ファイルを作成するディレクトリをネットワーク上で共有されているファイル・システムに置くと，ファイルの競合が生じて異常動作を起こす場合があります。オプションや環境変数の設定によって，このような競合を避けてください。

（9）オブジェクト・コンバータの操作仕様に関する注意事項

オブジェクト・コンバータ・オプション-Uで，スタートを指定した場合，スタート・アドレスまたはコードが配置されたアドレスの小さい方のアドレスから充てんを開始します。SFR領域（FF00H-FFFFH）には充てんを行いません。

記述形式：-U充てん値［，［スタート］，サイズ］

［　］内は省略可能です。

付録C　オプション一覧

プログラムのオプションを表形式でまとめました。
プログラム開発の際にお役立てください。
このオプション一覧は索引としても使用できます。

C. 1　構造化アセンブラ・オプション一覧

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照頁
1	デバイス種別指定	-Cデバイス種別	対象デバイスの種別を指定します。	独立	省略不可	75
2	ワード・シンボル文字指定	-SC文字	ワード・シンボル名の最後の文字を指定します。	独立	-SCP	76
3	シンボル定義指定	-Dシンボル[数値]	#IFDEF疑似命令などに与えるシンボルを指定します。	独立	なし	77
4	タブ数指定	-WT数値，数値，数値	変換した命令を出力する位置を指定します。	独立	-WT2, 3, 4	78
5	インクルード・ファイル・パス指定	-Iパス名…（複数指定可）	インクルード・ファイルを指定したパスから読み込みます。	独立	環境変数‘INC78K0S’指定パス	79
6	2次ソース・ファイル指定	-O[ファイル名]	2次ソース・ファイル名を指定します。	独立	-O[入力ファイル名.ASM]	80
7	エラー・リスト・ファイル指定	-E[ファイル名]	エラー・リスト・ファイルを出力します。	独立	-E[入力ファイル名.EST]	81
8	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	82
9	ディバグ情報出力指定	-GS	構造化アセンブラ・ソース・レベルのディバグ情報の出力を指定します。	-GSと-NGSを同時に指定した場合は，後で指定した方が有効です。	-GS	83
		-NGS	-GSオプションを無効にします。			
10	2次ソース・ファイル強制出力指定	-J	2次ソース・ファイルを強制的に出力します。	独立	強制出力しません。	84

項番	分　　類	記述形式	機　　能	他のオプションとの関係	省略時解釈	参照頁
11	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，後で指定した方が有効です。	Windows/HP-UXのとき-ZS SunOSのとき-ZE	85
		-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。			
12	デバイス・ファイル・サーチ・パス指定	-Yパス名	デバイス・ファイルを指定されたパスから読み込みます。	独立	〈..¥dev〉ST78K0S の起動されたパスに対して	86
13	ヘルプ指定	--	ディスプレイ（コンソール）にヘルプ・メッセージを出力します。	他のオプションをすべて無効にします。	表示しません。	87

C. 2　アセンブラ・オプション一覧

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照頁
1	デバイス種別指定	-Cデバイス種別	対象デバイスの種別を指定します。	独立	省略不可	102
2	オブジェクト・モジュール・ファイル出力指定	-O[ファイル名]	オブジェクト・モジュール・ファイルを出力します。	-Oと-NOを同時に指定した場合は，あとで指定した方が有効です。	-O[入力ファイル名.REL]	103
		-NO	オブジェクト・モジュール・ファイルを出力しません。			
3	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	104
		-NJ	-Jオプションを無効にします。			
4	ディバグ情報出力指定	-G	ローカル・シンボル情報をオブジェクト・モジュール・ファイルへ出力します。	-Gと-NGを同時に指定した場合は，あとで指定した方が有効です。	-G	105
		-NG	-Gオプションを無効にします。			
		-GA	ソース・ディバグ情報をオブジェクト・モジュール・ファイルへ出力します。	-GAと-NGAを同時に指定した場合は，あとで指定した方が有効です。	-GA	106
		-NGA	-GAオプションを無効にします。			
5	インクルード・ファイル読み込みパス指定	-Iパス名[，パス名]…（複数指定可）	インクルード・ファイルを指定したパスから読み込みます。	独立	環境変数‘INC78K0S’指定パス	107
6	アセンブル・リスト・ファイル出力指定	-P[ファイル名]	アセンブル・リスト・ファイル出力します。	-Pと-NPを同時に指定した場合は，あとで指定した方が有効です。	-P[入力ファイル名.REL]	108
		-NP	アセンブル・リスト・ファイル出力しません。			
7	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	109
		-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。	-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。	-NKS	111
		-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レファレンス・リストを出力します。	-NKA, -NKS, -NKXを同時に指定した場合は，-Pを無視します。	-NKX	112
		-NKX	-KXオプションを無効にします。			

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照頁
8	アセンブル・リスト・ファイル形式指定	-LW[文字数]	アセンブル・リスト・ファイルの1行に印字する文字数を変更します。	-NPを指定した場合は，-LWが無視されます。	-LW132	114
		-LL[行数]	アセンブル・リスト・ファイルの1頁に印字する行数を変更します。	-NPを指定した場合は，-LLが無視されます。	-LL66	116
		-LH文字列	アセンブル・リスト・ファイルのヘッダに，指定された文字列を出力します。	-NPを指定した場合は，-LHが無視されます。	なし	118
		-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	121
		-LF	アセンブル・リスト・ファイルの最後に改行コードを付加します。	-LFと-NLFを同時に指定した場合は，あとで指定した方が有効です。	-NLF	123
		-NLF	-LFオプションを無効にします。	-NPを指定した場合は，-LFが無視されます。		
9	エラー・リスト・ファイル出力指定	-E[ファイル名]	エラー・リスト・ファイルを出力します。	-Eと-NEを同時に指定した場合は，あとで指定した方が有効です。	-NE	124
		-NE	-Eオプションを無効にします。			
10	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	126
11	テンポラリ・ファイル作成パス指定	-Tパス名	テンポラリ・ファイルを，指定したパスに作成します。	独立	環境変数，‘TMP’指定パス	127
12	漢字コード指定	-ZS	コメントの漢字をシフトJISコードとして解釈します。	-ZSと-ZEと-ZNを同時に指定した場合は，あとで指定した方が有効です。	Windows HP-UXのとき-ZS SunOSのとき-ZE	128
		-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。			
13	デバイス・ファイル・サーチパス指定	-Yパス名	デバイス・ファイルを指定されたパスから読み込みます。	独立	〈..¥dev〉RA78K0S の起動されたパスに対して	129
14	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	130
15	ヘルプ指定	--	ディスプレイ（コンソール）にヘルプ・メッセージを出力します。	他のオプションをすべて無効にします。	表示しません。	131

C. 3　リンカ・オプション一覧

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照頁
1	ロード・モジュール・ファイル出力指定	-O[ファイル名]	ロード・モジュール・ファイルを出力します。	-CAと-NCA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]	155
		-NO	ロード・モジュール・ファイルを出力しません。			
2	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	156
		-NJ	-Jオプションを無効にします。			
3	ディバグ情報出力指定	-G	ディバグ情報をロード・モジュール・ファイルへ出力します。	-Gと-NGを同時に指定した場合は，あとで指定した方が有効です。 -NGが指定された場合は，-KP, -KLの指定にかかわらず，ローカル・シンボル・リスト，パブリック・シンボル・リストは出力されません。	-G	157
		-NG	-Gオプションを無効にします。			
4	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	158
		-NS	-Sオプションを無効にします。			
5	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	160
6	リンク・リスト・ファイル出力指定	-P[ファイル名]	リンク・リスト・ファイルの出力を指定します。	-PとNPを同時に指定した場合は，あとで指定した方が有効です。	-P[ファイル名.MAP]	161
		-NP	-Pオプションを無効にします。			
7	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。 -NKM, -NKP, -NKLをすべて指定した場合は，-Pは無効となります。	-KM	162
		-NKM	-KMオプションを無効にします。			
		-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	164
		-NKD	-KDオプションを無効にします。	-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。		
		-KP	リンク・リスト・ファイル中に，クロレファレンス・リストを出力します。		-NKP	166
		-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	168
		-NKL	-KLオプションを無効にします。			

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照頁
8	リンク・リスト・ファイル形式指定	-LL[行数]	リスト1頁に印字する行数を指定します。	-NPを指定した場合は，-LLは無視されます。	-LL66	170
		-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。	-NLF	172
		-NLF	-LFオプションを無効にします。	-NPを指定した場合は,後で指定した方が有効です。		
9	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	173
		-NE	-Eオプションを無効にします。			
10	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	174
11	ライブラリ・ファイル読み込みパス指定	-Iパス名[，パス名]…（複数指定可）	ライブラリ・ファイルを指定したパスから読み込みます。	-Bオプションで，パス名含まないライブラリ・ファイルを指定した場合は無効となります。	環境変数，‘LIB78K0S’指定パス	175
12	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	176
13	テンポラリ・ファイル作成パス指定	-Tパス名	テンポラリ・ファイルを，指定したパスに作成します。	独立	環境変数，‘TMP’指定パス	177
14	デバイス・ファイル・サーチ・パス指定	-Yパス名	デバイス・ファイルを指定されたパスから読み込みます。	独立	〈..¥dev〉LK78K0S の起動されたパスに対して	178
15	ワーニング・メッセージ出力指定	-W[レベル]	ワーニング・メッセージをコンソールへ出力するか否かを指定します。	独立	通常のエラーメッセージを出力	179
16	ヘルプ指定	--	ディスプレイ（コンソール）にヘルプ・メッセージを出力します。	他のオプションをすべて無効にします。	表示しません。	180

C. 4　オブジェクト・コンバータ・オプション一覧

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照頁
1	HEX形式オブジェクト・モジュール・ファイル出力指定	-O[ファイル名]	HEX形式オブジェクト・モジュール・ファイルを出力します。	-Oと-NOを同時に指定した場合は，あとで指定した方が有効です。	-O[入力ファイル名].HEX（拡張空間に対するファイルタイプH1～H15）	198
		-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。			
2	シンボル・テーブル・ファイル出力指定	-S[ファイル名]	シンボル・テーブル・ファイルを出力します。	-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。	-S[入力ファイル名].HEX（拡張空間に対するファイルタイプH1～H15）	199
		-NS	シンボル・テーブル・ファイルを出力しません。			
3	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。	-NR	200
		-NR	-Rオプションを無効にします。			
4	オブジェクト充てん指定	-U充てん値[，[スタート]，サイズ]	HEX形式オブジェクトが出力されない領域に対して，指定した充てん値をオブジェクト・コードとして出力します。	-NOを指定した場合は，-Uは無視されます。	なし	201
5	エラー・リスト・ファイル出力指定	-E[ファイル名]	エラー・リスト・ファイルを出力します。	-Eと-NEを同時に指定した場合は，あとで指定した方が有効です。	-NE	203
		-NE	-Nオプションを無効にします。			
6	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	204
7	デバイス・ファイル・サーチ・パス指定	-Yパス名	デバイス・ファイルを指定されたパスから読み込みます。	独立	〈..¥dev〉OC78K0Sの起動されたパスに対して	205
8	ヘルプ指定	--	ディスプレイ（コンソール）にヘルプ・メッセージを出力します。	他のオプションをすべて無効にします。	表示しません。	206

C. 5　ライブラリアン・オプション一覧

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照頁
1	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。-	-LW132	220
		-LL[行数]	リスト・ファイルの1頁の行数を変更します。		-LL66	221
		-LF	リスト・ファイルの最後に，改頁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。	-NLF	222
		-NFL	-LFオプションを無効にします。			
2	テンポラリ・ファイル作成パス指定	-Tパス名	テンポラリ・ファイルを，指定したパスに作成します。	独立	環境変数，‘TMP’指定パス	223
3	デバイス・ファイル・サーチ・パス指定	-Yパス名	デバイス・ファイルを指定されたパスから読み込みます。	独立	〈..¥dev〉LB78K0S の起動されたパスに対して	224
4	ヘルプ指定	--	ディスプレイ（コンソール）にヘルプ・メッセージを出力します。	他のオプションをすべて無効にします。	表示しません。	225

C. 6 リスト・コンバータ・オプション一覧

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照頁
1	オブジェクト・モジュール・ファイル入力指定	-R[ファイル名]	オブジェクト・モジュール・ファイルを入力します。	独立	-R[アセンブル・リスト・ファイル名.REL]	247
2	ロード・モジュール・ファイル入力指定	-L[ファイル名]	ロード・モジュール・ファイルを入力します。	独立	-L[アセンブル・リスト・ファイル名.LNK]	248
3	アブソリュート・アセンブル・リスト・ファイル出力指定	-O[ファイル名]	アブソリュート・アセンブル・リスト・ファイルを出力します。	独立	-O[アセンブル・リスト・ファイル名.P]	249
4	エラー・リスト・ファイル出力指定	-E[ファイル名]	エラー・リスト・ファイルを出力します。	-Eと-NEを同時に指定した場合は，あとで指定した方が有効です。	-NE	250
		-NE	-Nオプションを無効にします。			
5	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	251
6	ヘルプ指定	--	ディスプレイ（コンソール）にヘルプ・メッセージを出力します。	他のオプションをすべて無効にします。	表示しません。	252

付録D　サブコマンド一覧

サブコマンドを，一覧表にまとめて示します。
プログラム開発の際にお役立てください。
このサブコマンド一覧は，索引としても使用できます。

項番	分　類	記述形式	機　能	短縮形	参照頁
1	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	227
2	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	228
3	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	229
4	REPLACE	REPLACE△ライブラリ・ファイル名△トランザクション	ライブラリ・ファイル内のモジュールを他のモジュールと置き換えます。	R	230
5	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	231
6	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	232
7	HELP	HELP	ティスプレイ（コンソール）にヘルプ・メッセージを出力します。	H	233
8	EXIT	EXIT	ライブラリアンを終了します。	E	234

付録E　索　　引

50音で始まる語句の索引

記号，アルファベットで始まる語句の索引

【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

【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/

【営業関係お問い合わせ先】

下記のページに最新版のお問い合わせ先が記載されています。

URL（アドレス）　http://www.necel.com/ja/contact/contact_j.html

【技術的なお問い合わせ先】

半導体テクニカルホットライン

（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−435−9494

FAX　：044−435−9608

E-mail　：info@lsi.nec.co.jp

【資料請求先】

NECエレクトロニクス特約店または上記ホームページ記載の営業関係お問い合わせ先へお申し付けください。
