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

μPD70108,μPD70116

リロケータブル・アセンブラ・パッケージ

取扱説明書

INTELLEC™ベース用

NEC 日本電気株式会社

EEM−618
December 1986P

μPD70108,μPD70116

リロケータブル・アセンブラ・パッケージ

取扱説明書

INTELLECベース用

NEC 日本電気株式会社

ま え が き

このμPD70108，μPD70116リロケータブル・アセンブラ・パッケージは，INTELLEC Series ⅢのISIS™—ⅡのＯＳ下で動作するように作られたものであり，μPD70108（別名称Ｖ20™），μPD70116（別名称Ｖ30™），μPD70208（別名称Ｖ40™），μPD70216（別名称Ｖ50™）のプログラム開発に使用します．

1. システム構成

(1) ホスト・コンピュータ

INTELLEC Series Ⅲ

(2) オペレーティング・システム（ＯＳ）

ＩＳＩＳ—Ⅱ（バージョン　4.3以上）

(3) 使用メモリ・サイズ

１２８Ｋバイト以上（４０００Ｈ～５ＦＦＦＦＨ番地）

(4) コンソール

キーボードとＣＲＴ

2. リロケータブル・パッケージの供給形態

μPD70108，μPD70116リロケータブル・パッケージには次のソフトウエアが含まれております．

- μPD70108，μPD70116　リロケータブル・アセンブラ
- μPD70108，μPD70116　リンカ
- μPD70108，μPD70116　ＨＥＸ形式オブジェクト変換プログラム
- μPD70108，μPD70116　ライブラリアン

また，プログラム・ファイル名およびフロッピィ・ディスクの形式は次のとおりです．

(1) ファイル名

(a) リロケータブル・アセンブラ

ＲＡ１１６．８６

ＲＡ１１６．ＯＭｎ（ｎは順序番号）

(b) リンカ

ＬＫ１１６．８６

ＬＫ１１６．ＯＭｎ（ｎは順序番号）

注1　INTELLEC™，ＩＳＩＳ™は，米国インテル社の商標です．

注2　Ｖ20™，Ｖ30™，Ｖ40™，Ｖ50™は，日本電気株式会社の商標です．

(c) HEX形式オブジェクト変換プログラム

OC116.86

(d) ライブラリアン

LB116.86

(2) フロッピィ・ディスクの形式

片面単密度／片面倍密度の8インチ・フロッピィ・ディスク

3. 本マニュアルの構成

本マニュアルは，μPD70108，μPD70116のリロケータブル・アセンブラ・パッケージの概要，取扱い方法等について述べており，次の五編より構成されております．

第Ⅰ編 リロケータブル・アセンブラ・パッケージの概説

アブソリュート・アセンブラとリロケータブル・アセンブラの違いやリロケータブル・アセンブラの効果的な利用方法について述べております．

第Ⅱ編 リロケータブル・アセンブラの取扱方法

リロケータブル・アセンブラの機能，特徴，ソース・プログラムの様式，疑似命令およびプログラムの操作方法等について述べております．

第Ⅲ編 リンカの取扱方法

リンカの機能，操作方法などについて述べております．

第Ⅳ編 HEX形式オブジェクト変換プログラムの取扱方法

ソフトウエアの概要，機能，操作方法および拡張HEXオブジェクト・フォーマットについて述べております．

第Ⅴ編 ライブラリアンの取扱方法

ソフトウエアの概要，機能および操作方法について述べております．

本説明書中で使用される記号の意味

…	同一形式の連続
〔　〕	かっこ内を省略可能
{　}	かっこ内の一つを選択
"　"	" "で囲まれた文字そのもの
太字列	文字列そのもの
―	コマンドの説明での下線は，ユーザの入力箇所
CR	復帰（キャリッジ・リターン）
LF	改行（ライン・フィード）
HT	水平タブ
FD	フロッピィ・ディスク
HD	ハード・ディスク
LP	ライン・プリンタ
PT	紙テープ
TTY	テレタイプ（コンソール）
)	復改（$C_R \cdot L_F$）または復改（C_R）
OM(F)	オブジェクト・モジュール（ファイル）
LM(F)	ロード・モジュール（ファイル）
LBF	ライブラリ・ファイル
WKF	ワーク・ファイル
PRF	リスト・ファイル
HXF	拡張HEX形式オブジェクト・プログラム・ファイル

空白ページ

目　　次

第Ⅰ編　リロケータブル・アセンブラ・パッケージの概説

第Ⅱ編　リロケータブル・アセンブラの取扱方法

第Ⅲ編　リンカの取扱方法

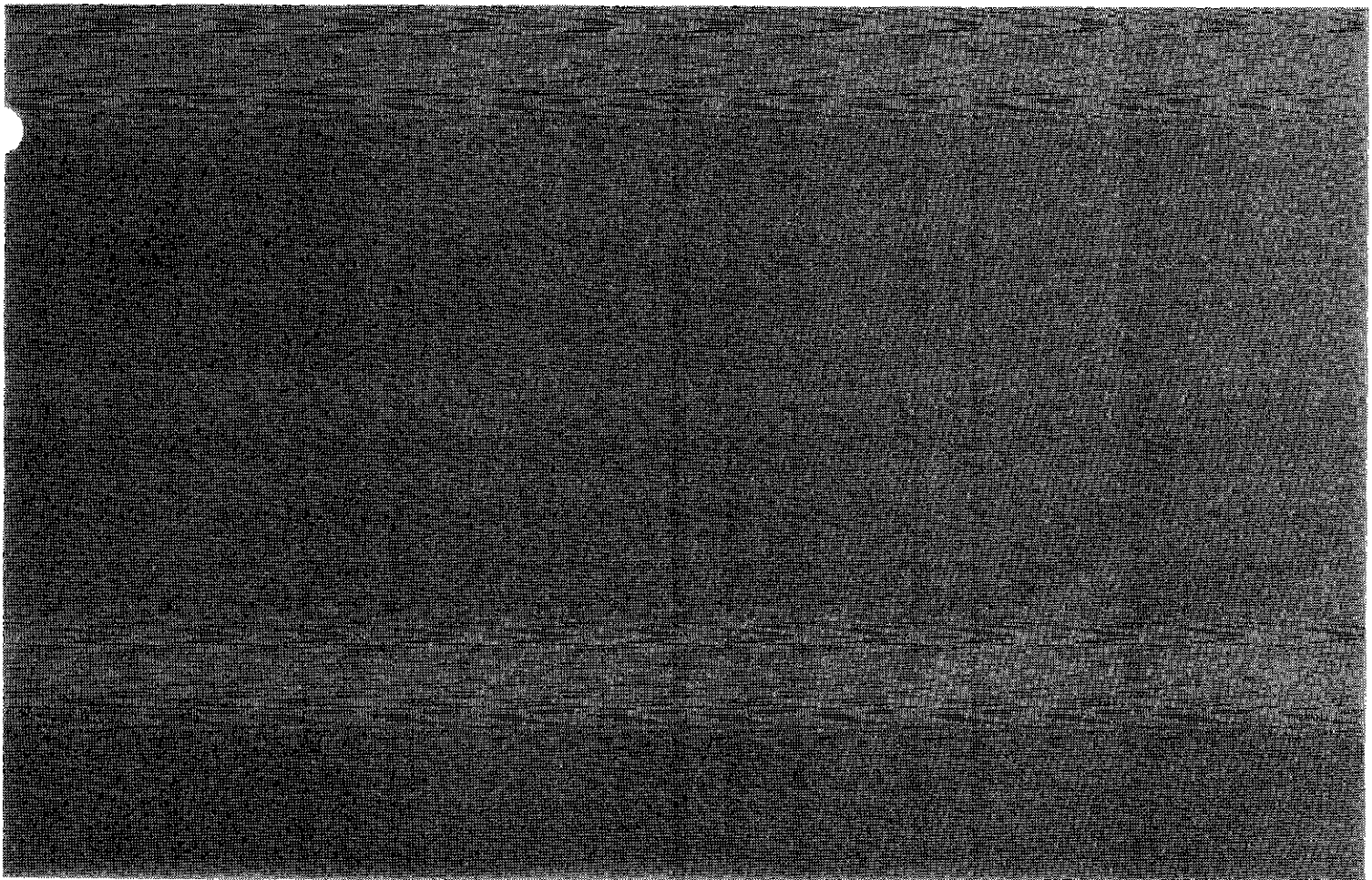
第Ⅳ編　HEX形式オブジェクト変換プログラムの取扱方法

第Ⅴ編　ライブラリアンの取扱方法

第Ⅰ編

μPD70108,μPD70116

リロケータブル・アセンブラ・パッケージの概説



ま え が き

本編では一般的なリロケータブル・アセンブラ・パッケージの利用について述べており，次の三章から構成されております．

第1章　概　　要

リロケータブル・アセンブラとアブソリュート・アセンブラの特徴を比較しながら，リロケータブル・アセンブラ・パッケージの概要について説明しております．

第2章　モジュラ・プログラミングの利点

リロケータブル・アセンブラ・パッケージを効果的に利用する場合に用いられるモジュラ・プログラミング手法について開発面や保守面から利点を述べています．

第3章　開発手順

モジュラ・プログラミングを用いた一般的な開発方法を，プログラムの設計から保守段階までの各ステップ毎に解説しています．

既にリロケータブル・アセンブラ・パッケージについて知識のある方でも，第3章の一読をおすすめします．

第1章 概　　要

1.1 アセンブラとは

マイクロプロセッサは，0と1の組合せからなる機械語と呼ばれる言葉しか判別することができません．しかし，機械語は人間にとっては非常に複雑であり，覚えにくいものです．そこで機械語に対応させて象徴的な言葉（シンボリック言語またはアセンブリ言語）をわりあてることにより，機械語の覚えにくさ，扱いにくさから解放されます．アセンブラは，人間にとって扱いやすい言葉であるシンボリック言語を，マイクロ・プロセッサが唯一理解できる機械語に変換するプログラムです．

1.2 アブソリュート・アセンブラとは

機械語はその用途により，命令とデータに分けられます．命令はマイクロ・プロセッサに対し，動作の種類を指定するものであり，データはその時に処理される値です．
データには，演算命令で処理される定数や変数があります．

アブソリュート・アセンブラは，機械語への変換の際，命令やデータに割りつけられた番地を絶対的（アブソリュート）に決定してしまうアセンブラです．したがって，機械語自身がメモリ上に格納される番地が決められていなければならず，その情報は“ORG”という番地指定疑似命令を使ってアセンブラに知らされます．

アブソリュート・アセンブラにより生成された機械語は，そのままメモリに格納（ロード）し，マイクロ・プロセッサによって実行させることができるので，別名ロード・モジュールと呼びます．それに対し，その原始（ソース）であるシンボリック言語のものをソース・モジュールと呼びます．

1.3 リロケータブル・アセンブラとは

アブソリュート・アセンブラによって出力される機械語は，メモリ上の配置が定められているものです．それに対して，メモリ上の任意の番地に再配置可能（リロケータブル）な機械語を出力するアセンブラをリロケータブル・アセンブラと呼び，リロケータブル・アセンブラによって出力される機械語をリロケータブル・オブジェクト・モジュールと呼びます．この機械語は，そのままメモリにロードして，実行させることはできません．

1.4 リンカとは

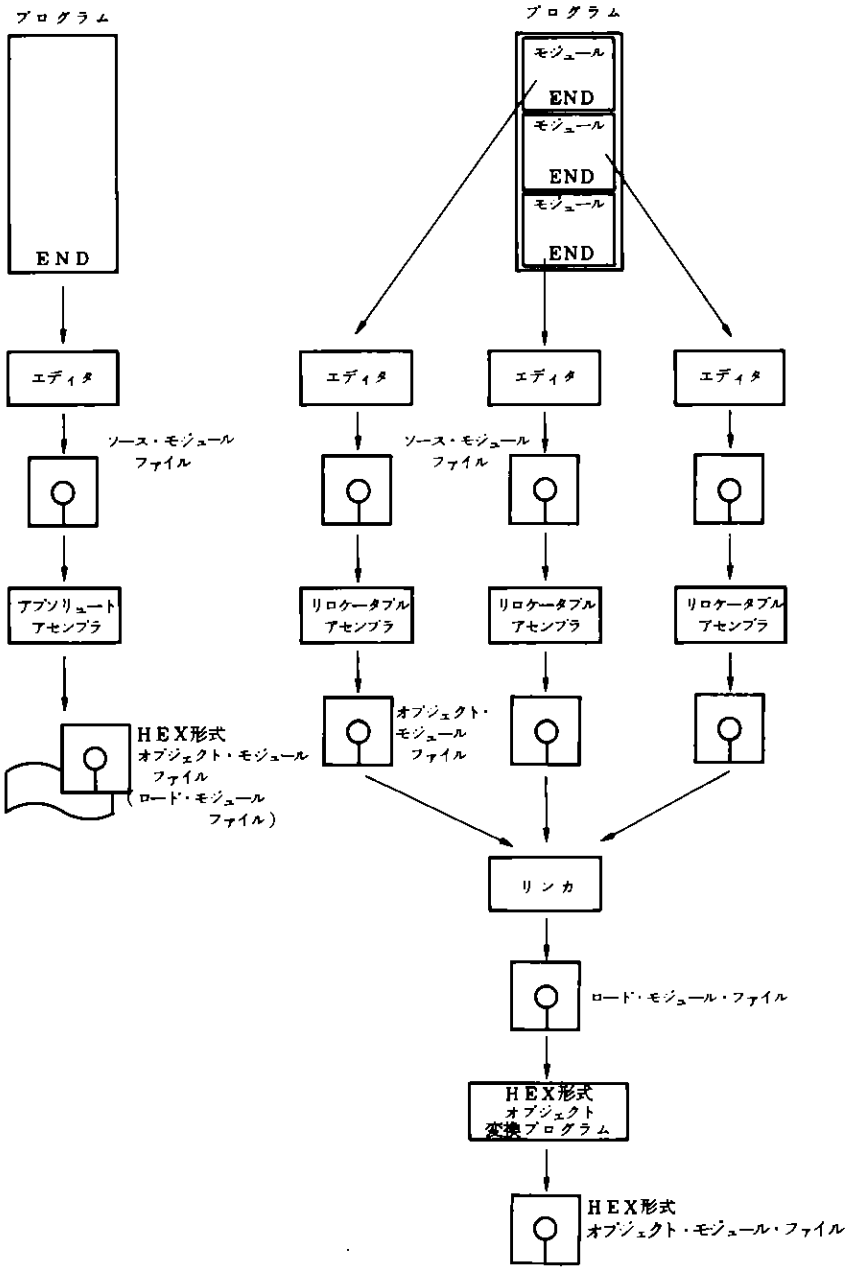
リロケータブル・オブジェクト・モジュールのメモリ上の配置場所を決定し，ロード・モジュールにするのがリンカです．また，リンカは，相互に番地の参照関係を持っている複数のリロケータブル・オブジェクト・モジュール群を結びつけて（結合と参照番地の解決），一つのロード・モジュールにする機能を備えています．

1．5　リロケータブル・アセンブラとリンカを使用してのプログラム開発

リロケータブル・アセンブラとリンカの機能を利用することにより，シンボリック言語で記述するプログラムは，機能的に100～500行ぐらいのサブプログラムに分割して並行開発できます．このような分割方法によるプログラミングをモジュラ・プログラミングと呼びます．

図1－1にアブソリュート・アセンブラを使った場合とリロケータブル・アセンブラ・パッケージを用いた場合との開発の流れを示します．

図1－1　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・パッケージとの開発比較



第２章　モジュラ・プログラミングの利点

アブソリュート・アセンブラで一つのプログラムを作成する際には，原則として，一度にプログラミングしなければなりませんでした．しかし，大きなプログラムを一度に作成するのは困難であり，また，保守する際にもプログラムの解析が大変であるため，一つのプログラムを機能ごとに，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）開発を行ないたい（これをモジュラ・プログラミングと言います）との要求がでてきました．

けれども，アブソリュート・アセンブラでは，飛び先の番地や，データの共通領域等，モジュール相互に関連する部分についていつもユーザが意識しなければならず，アブソリュート・アセンブラでモジュラー・プログラミングを行なうことはかなり困難でした．

そこで，ユーザがモジュール相互に関連する部分を意識せずに，モジュラ・プログラミングができるように考案されたアセンブラが，リロケータブル・アセンブラです．したがって，モジュラ・プログラミングの利点としては，

- ○効率のよい開発ができる
- ○保守がしやすい
- ○より高い信頼性が得られる

などがあげられます．

２．１　効率のよい開発

モジュラ・プログラミングにおいては，一つのプログラムを機能別に複数のモジュールに分割するため，それぞれのモジュールは小規模になります．すなわち，ある限定された機能の範囲でサブプログラムを作ればよいので，短時間で容易にそれぞれのモジュールのプログラミングができます．

また一般に，モジュール単位で独立して，エディット，アセンブル，ディバグ等を行なって，徐々に完全なプログラムにした方が，プログラム全体を一度にディバグするよりもバグを見つけやすく，ディバグする手間が少なくなります．

その上，複数の人間が各モジュールの開発を分担し，並行開発することも可能であるため，短期間に一つのプログラムを開発することも容易になります．

以上のように，モジュラ・プログラミングを利用することにより，効率よくプログラム開発ができるようになります．

２．２　保守のしやすさ

一つのプログラムが完成し，利用していくにつれ，場合によりプログラムの変更，追加をすることがあります．この場合，プログラムが完成してから日数が経過したため，プログラムの細かい点を忘れていたり，プログラム作成者以外の人がプログラムの変更，追加をしなければならないことがあります．

モジュラ・プログラミングにおいては，プログラムを機能的に分割した複数のモジュールで構成するため，変更，追加の機能に関連するモジュールのみを探すことができます．そして，この関連するモジュールについてのみ注目すれば，目的とする変更，追加ができるため，プログラムの解析をせねばならない部分が最小限ですみ，保守が容易になります．

２．３　より高い信頼性と汎用性

以前作成されたモジュールで，信頼性が高く，汎用性のあるものを保存しておき（ライブラリ化），別のプログラムを作る時に，そのモジュールを利用することができるのも，モジュラ・プログラミングの特徴のひとつです．つまり，さまざまなプログラムを作る場合に，このライブラリ化したモジュールを必要に応じてリンクすることにより，新規にプログラミングする部分を少なくすることができます．したがって，様々なプログラムを，正確にしかも早く作ることができます．

第３章　開発手順

ここでは，モジュラ・プログラミング手法を用いた一般的な開発手順を述べます．

３．１　ソース・モジュール・ファイルの作成

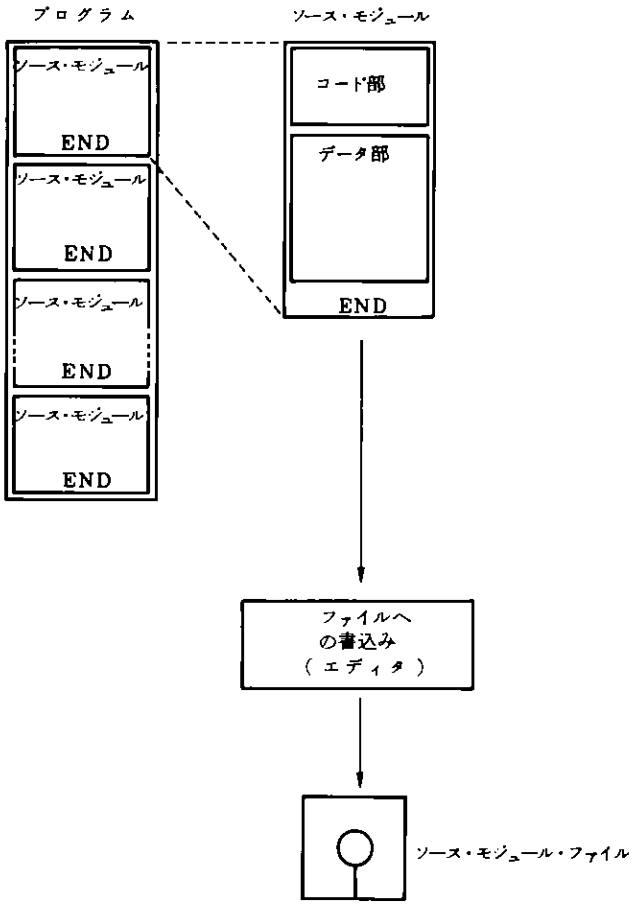
プログラムは機能的に幾つかのサブプログラムに分割して設計します．サブプログラムは機能的に独立性を高くしておくと個々のディバグが容易になり，開発効率を高めたり後の保守をしやすくすることにもつながります．

一つのサブプログラムはコーディングの単位になるもので，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はソース・モジュールと呼びます．

ソース・モジュールの記述は命令部分とデータ部分とに分けて行ないます．

ソース・モジュールのコーディングが終ったらエディタ等を使ってファイルに書込みます．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．

図３－１　ソース・モジュール・ファイルの作成



3.2 アセンブルの実行(オブジェクト・モジュール・ファイルの作成)

ソース・モジュール・ファイルができたら，アセンブラを使ってアセンブルします．アセンブラはリスト・ファイルとオブジェクト・モジュール・ファイルを出力します．

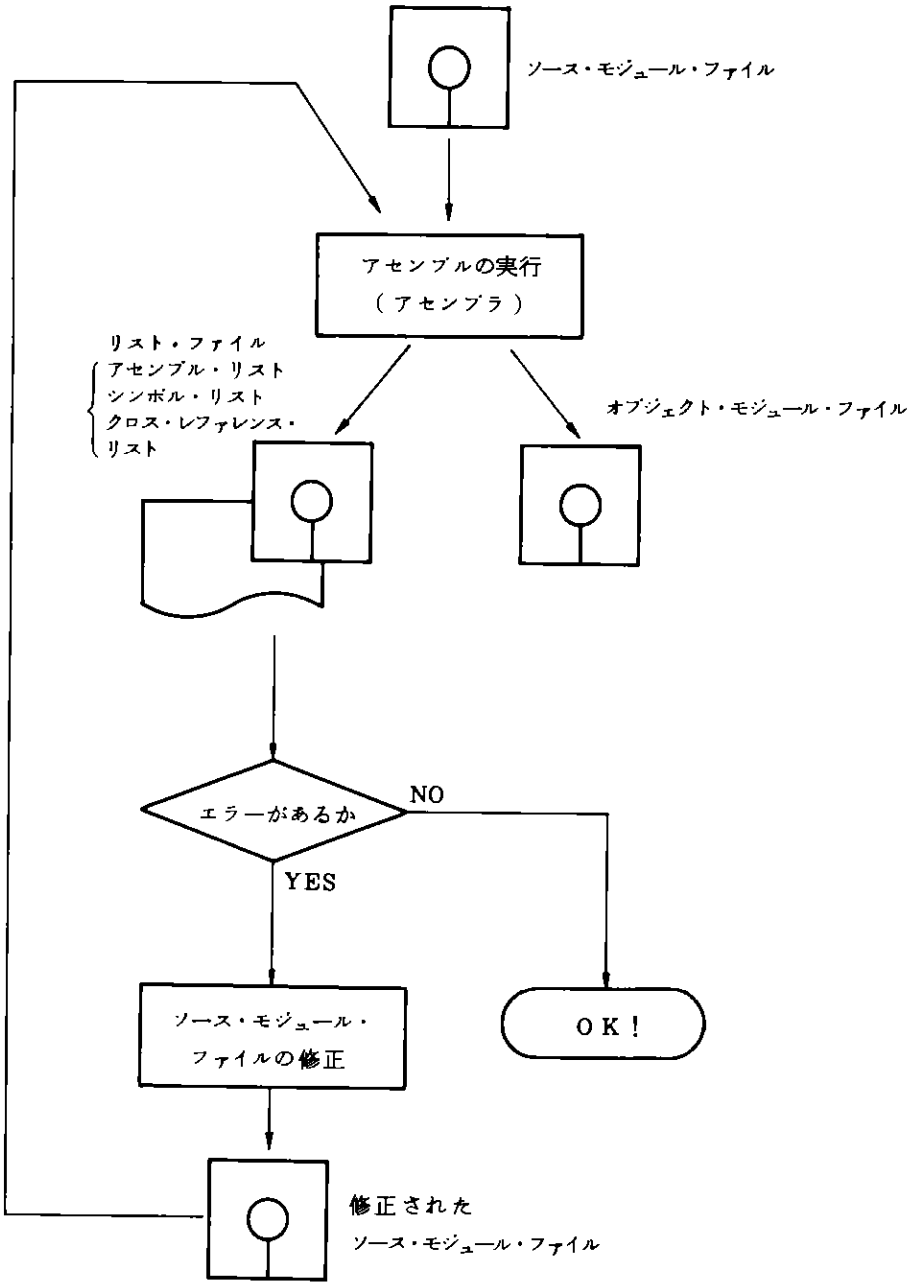
リスト・ファイルにはアセンブル・リスト，シンボル・リスト，クロス・レファレンス・リストが含まれます．

オブジェクト・モジュール・ファイルには，ソース・モジュールをアセンブルして得られた機械語情報と結合情報が含まれ（オブジェクト・モジュール），次のステップのリンカへの入力となります．

アセンブル・リスト上などでエラーを見つけたら，ソース・モジュールを修正し，エラーがなくなるまでアセンブルを繰返してください．

これらを必要なすべてのソース・モジュールに対して行ない，オブジェクト・モジュール・ファイルを作り上げます．

図８－２　オブジェクト・モジュール・ファイルの作成



3.3 オブジェクト・モジュールの再配置と結合（ロード・モジュール・ファイルの作成）

アセンブル・エラーのないオブジェクト・モジュール・ファイルができ上ったら，次にそれらの再配置と結合をします．これにはリンカを用います．

再配置というのは複数のオブジェクト・モジュールのメモリ上に配置される番地（アドレス）を決めることです．

結合というのはオブジェクト・モジュール間で共通に用いているシンボルの参照関係を調べ再配置後のアドレスを決定し，参照している機械語を決定することです．

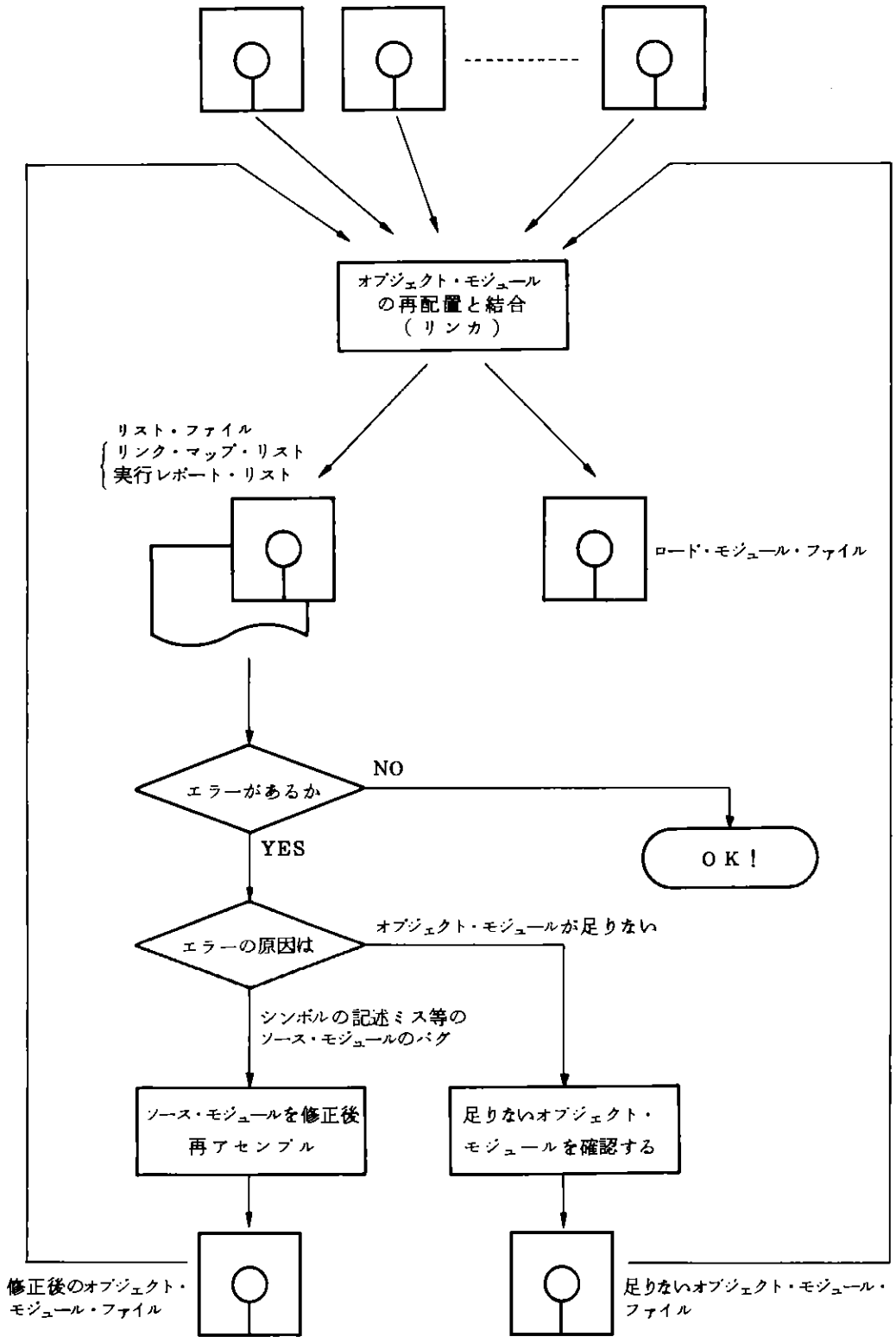
リンカは再配置と結合をした結果，ロード・モジュール・ファイルとリンク・マップ・リストなどのリスト・ファイルを出力します．

ロード・モジュール・ファイルは複数のオブジェクト・モジュールを再配置，結合したあとの機械語情報と結合情報を含んだものです．

またシンボルの参照関係が解決できなかった場合や，機械語のメモリ・サイズが許容範囲を超えてしまった場合には実行レポート・リストにエラーが印字されます．これはソース・モジュールの段階でシンボルの記述ミスをしたり，あるいはリンカに入力するオブジェクト・モジュールが足りないことが原因となります．前者の場合，ソース・モジュールを修正します．後者の場合は必要なモジュールをもう一度確かめてリンクしなおします．

エラーがなくなってできあがったロード・モジュールは実行可能になっています．

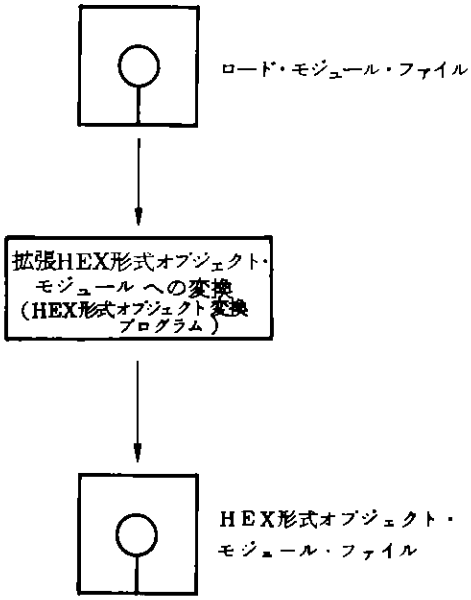
図８－３　ロード・モジュール・ファイルの作成



3.4 HEX形式オブジェクト・モジュールへの変換

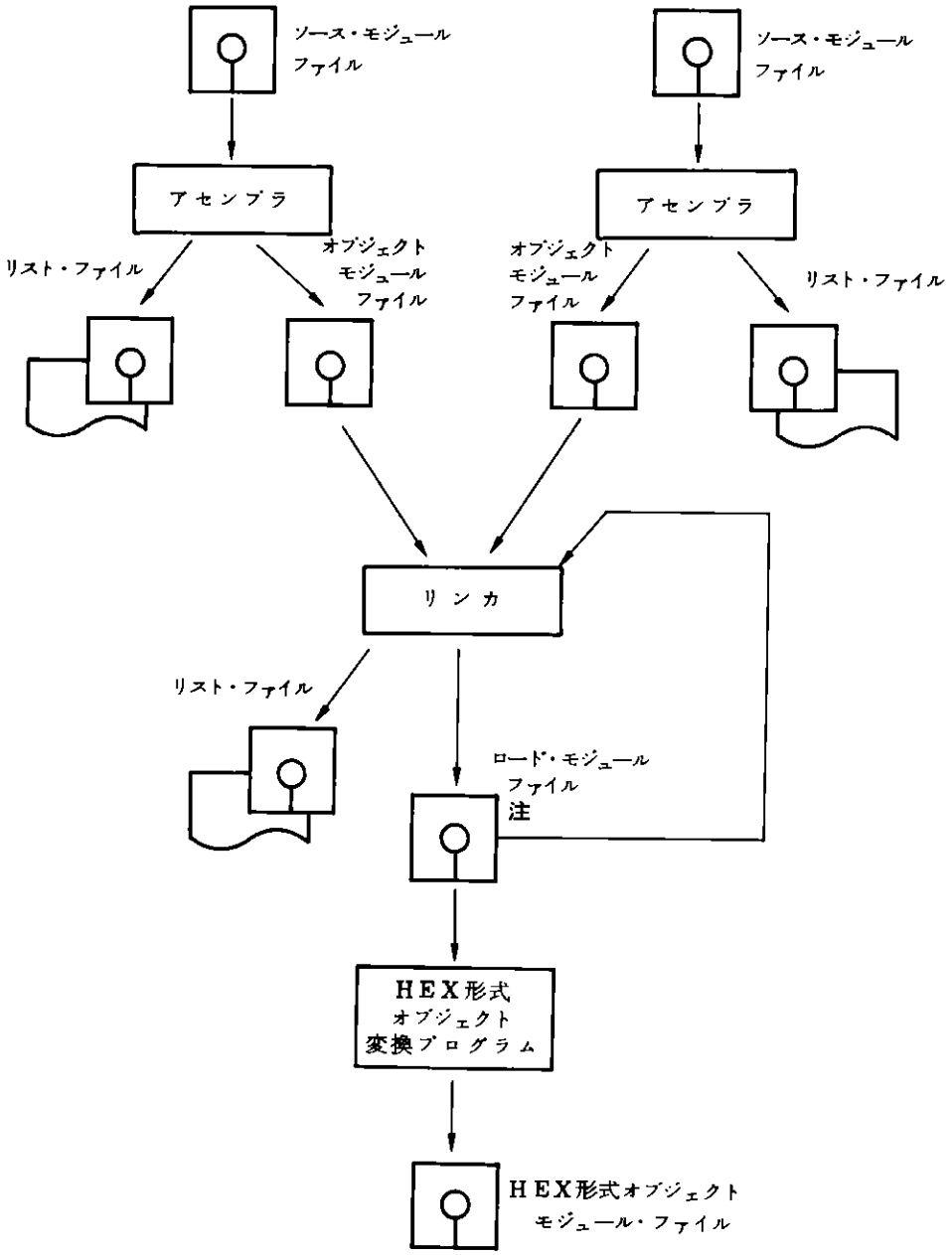
HEXローダの入力用などに拡張HEX形式のオブジェクト・モジュールが必要な場合は，HEX形式オブジェクト変換プログラムを使います．入力となるのはロード・モジュール・ファイルです．

図3－4　HEX形式オブジェクト・プログラム・ファイルの作成



アセンブラ，リンカ，HEX形式オブジェクト変換プログラムの流れを図示すると図3－5のようになります．

図 8－5　リロケータブル・アセンブラ・パッケージでの流れ



注　ロード・モジュールはリンカに再入力して，オブジェクト・モジュールとまったく同様に扱えます．

3.5 モジュールのライブラリ化

モジュラ・プログラミング手法で開発されたプログラムでは複数のソース・モジュールとオブジェクト・モジュールができあがります．これらを別個のファイルで持つと管理が繁雑になり保守もたいへんです．そこでこれらのモジュールを一つのファイルに集約し，管理を容易にする必要がでてきます．これを実現するものがライブラリアンであり，こうして管理することをモジュールのライブラリ化と呼びます．また，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．

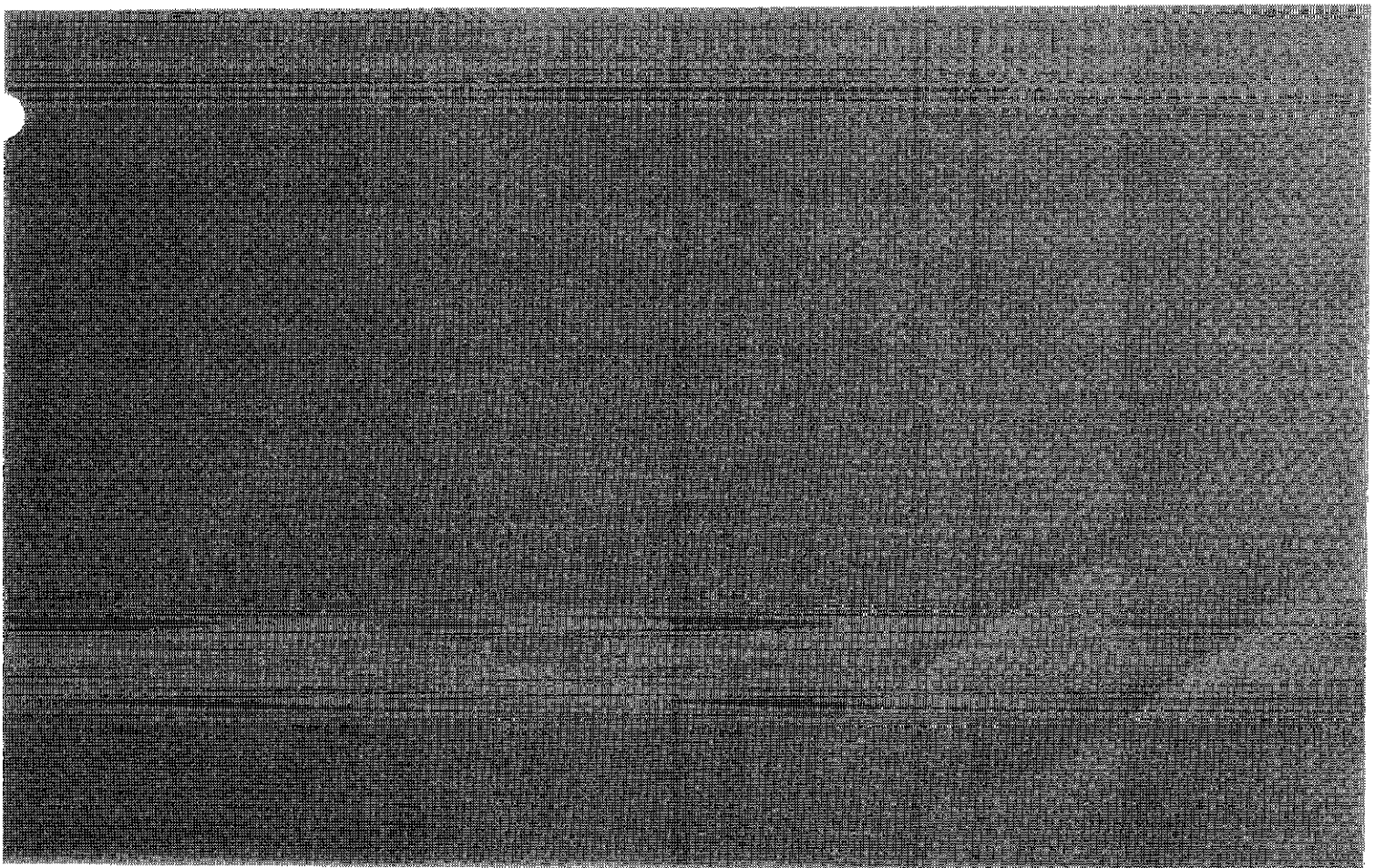
ライブラリアンは新規ライブラリ・ファイルの作成以外に，ライブラリ・ファイルのメンバであるモジュールの追加，削除，置きかえなどの機能を持っています．

またライブラリ・ファイルをリンカの入力に割り当てると，リンカはそのファイルから必要なモジュールだけを取り出して再配置，結合します．この機能を自動ライブラリ機能と呼びます．この機能を利用して，既にできあがった信頼性の高いモジュールからなるライブラリ・ファイルを作っておき，各プログラムで共有するようにすると，開発効率のアップと高い信頼性を期待できます．

第II編

μPD70108,μPD70116

リロケータブル・アセンブラの取扱方法



第１章　ソフトウエア概要

１．１　アセンブラの機能概要

本アセンブラはμPD70108，μPD70116の命令を使用して記述されたソース・プログラムを入力してマシン・コードに変換（アセンブル）し，オブジェクト・モジュールとして出力します．

本アセンブラは機械語命令のほかに２５種の疑似命令を備えており，ソース・プログラムの作成が容易です．

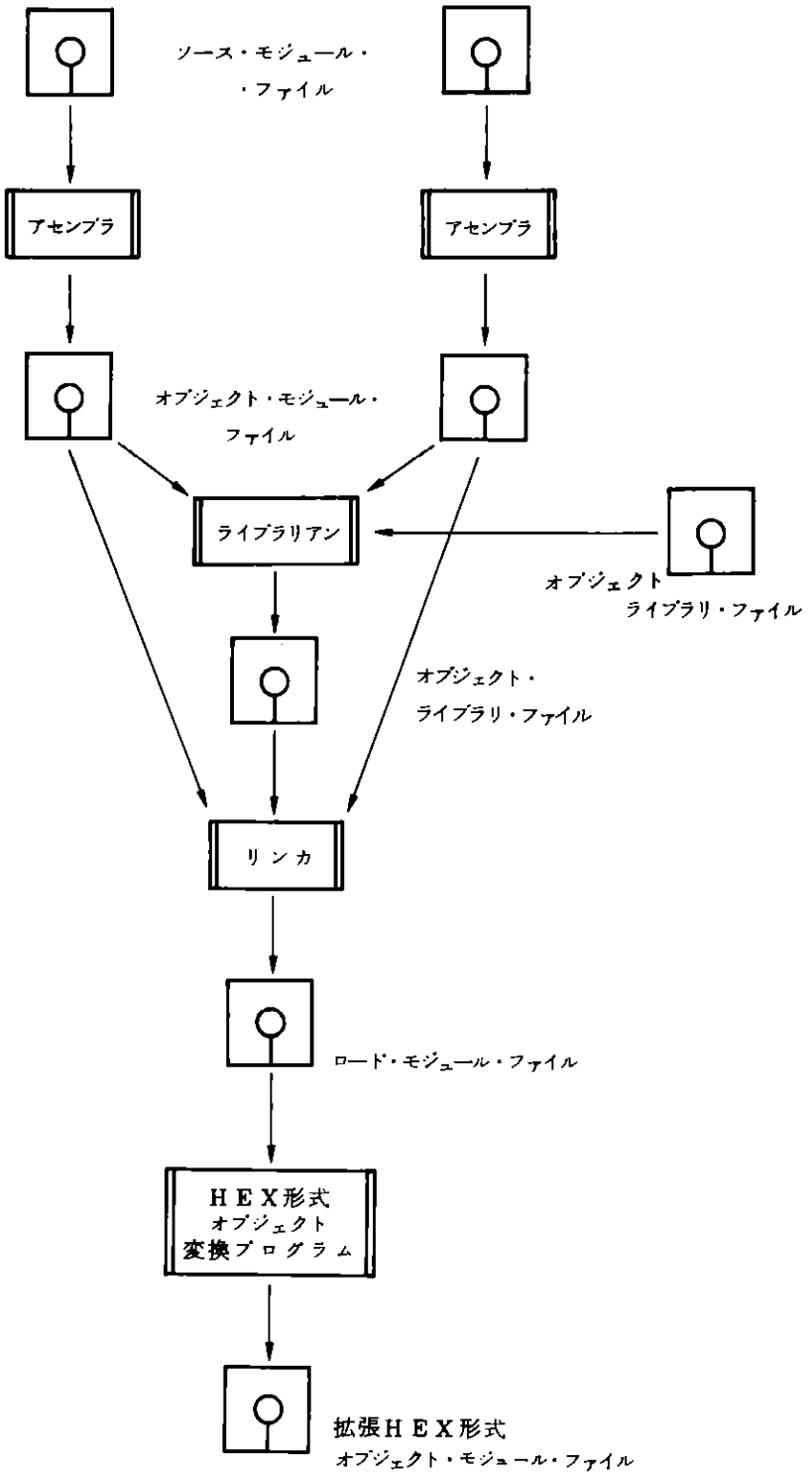
また本アセンブラはリロケータブル・アセンブル形式をとっており，関連プログラムにリンカ，ライブラリアンおよびＨＥＸ形式オブジェクト変換プログラムがあります．リンカは別々にアセンブルされた複数のオブジェクト・モジュールを連結します．ライブラリアンは複数のオブジェクト・モジュールを一つのファイル（これをライブラリと呼びます）に格納して管理し，ＨＥＸ形式オブジェクト変換プログラムはリンカによって連結されたモジュール（ロード・モジュール）を拡張ＨＥＸ形式のモジュールに変換します．

リンカ，ライブラリアンおよびＨＥＸ形式オブジェクト変換プログラムについては，次の編を参照してください．

「第Ⅲ編　μPD70108，μPD70116リンカの取扱方法」
「第Ⅳ編　μPD70108，μPD70116ＨＥＸ形式オブジェクト変換プログラムの取扱方法」
「第Ⅴ編　μPD70108，μPD70116ライブラリアンの取扱方法」

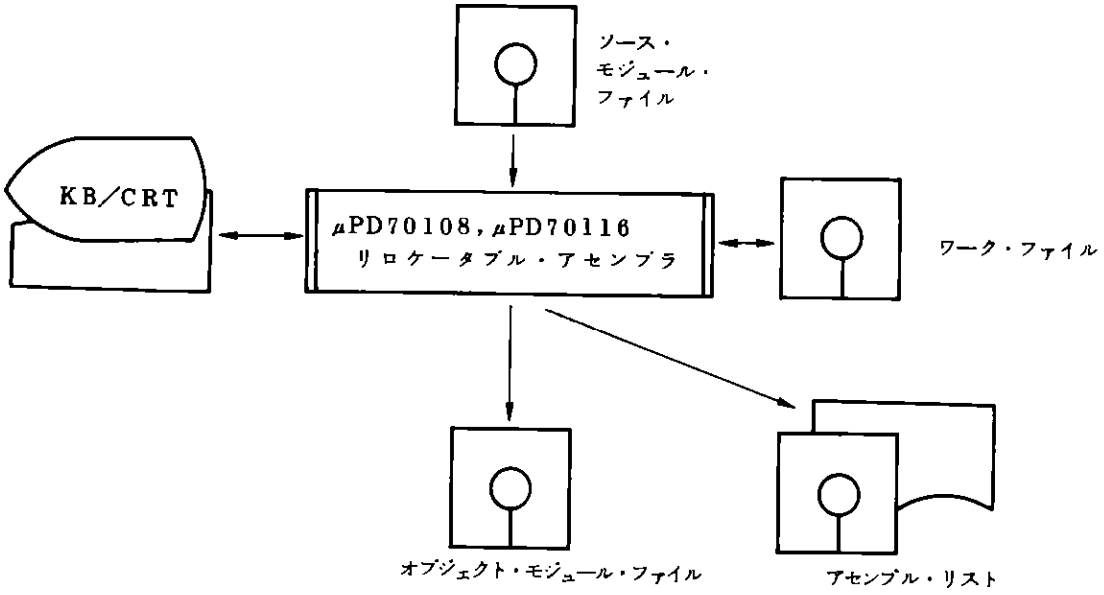
次ページにアセンブラ，リンカ，ライブラリアンおよびＨＥＸ形式オブジェクト変換プログラムの入出力関連を示します．

図１－１　アセンブラからHEXオブジェクト変換プログラムまでの流れ



1．2　システム構成

図1－2　システム構成



1．3　ワーク・ファイル

本アセンブラは，指定されたドライブのフロッピィ・ディスク上にワーク・ファイルを作り，アセンブル終了後に削除します．

プライマリ・ネーム	ファイル・タイプ
ソース・モジュール・ファイルと同じ	$$n　（nは順序番号）

1．4　シンボル数

本アセンブラで使用可能なシンボルの最大数は８００個です．[注]

注　アセンブラのバージョンによって多少変動することがあります．

第２章　本アセンブラの特徴

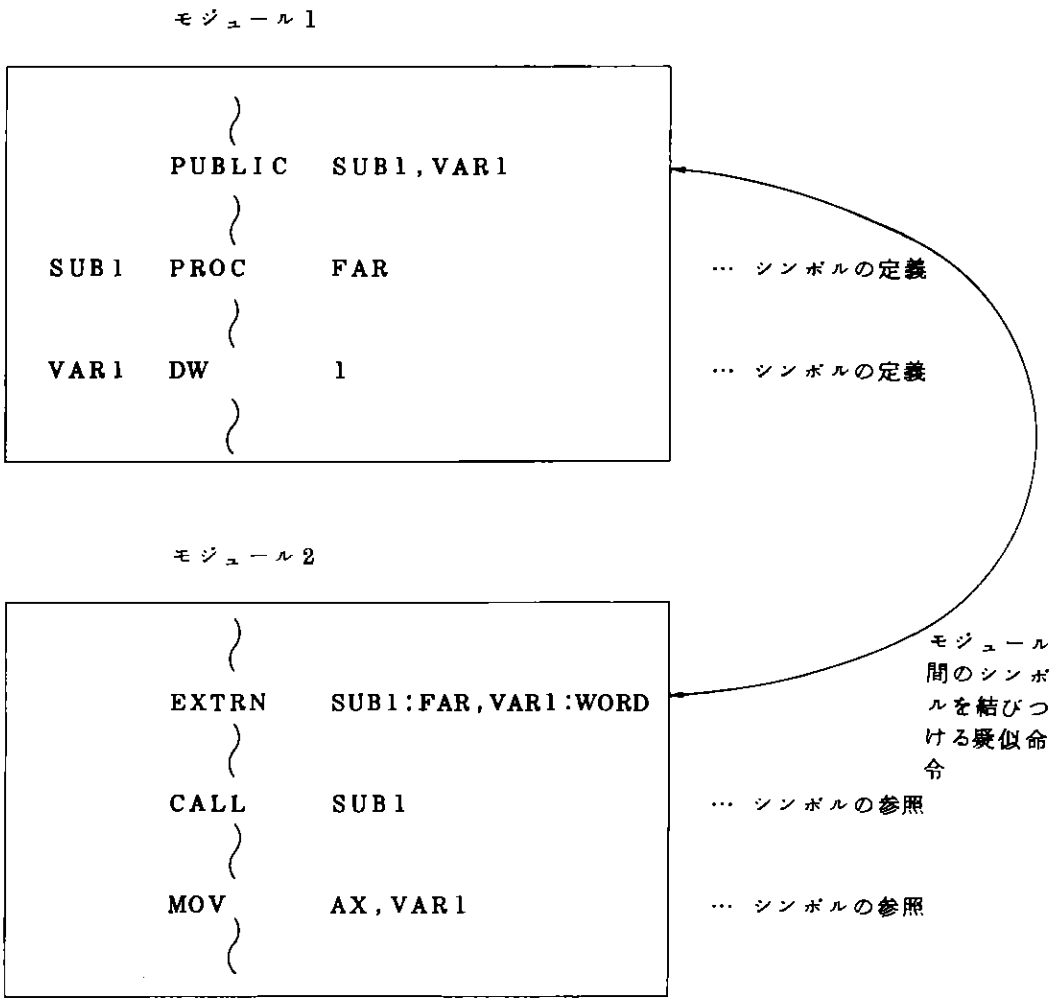
本アセンブラは，リロケータブル・アセンブル形式をとっています．

２．１　分割アセンブル

複数に分割したソース・モジュールを各々独立にアセンブルできます．

プログラムを複数のモジュールに分割した場合，モジュール間で互いにシンボルを参照する必要がありますのでプログラム・リンケージ疑似命令（ 4.7 参照 ）が用意されています．

例



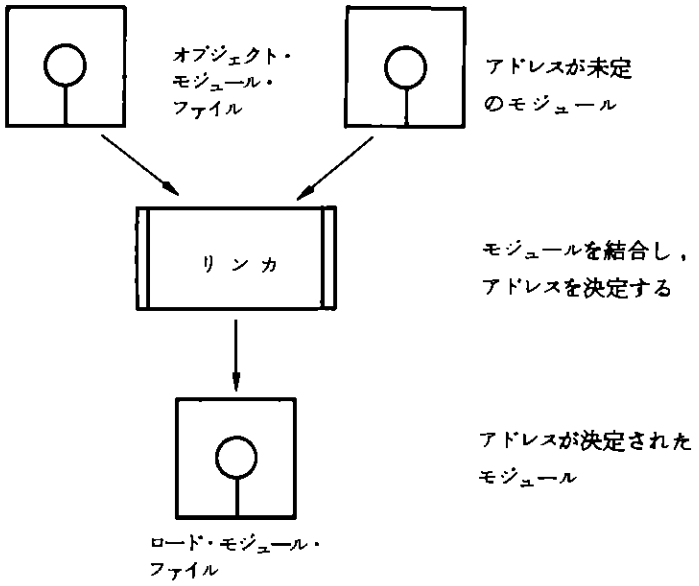
2.2 リロケータブル・アセンブル

複数のモジュールに分割してプログラム開発をする場合，各々のモジュールで記述した命令が最終的にメモリのどの番地（アドレス）に配置されるかを意識するというのは多大の工数を要します．（ただし，割込み処理ルーチン等の特殊な手続きを除きます．）

本アセンブラでは，プログラム開発中は特にアドレスを意識しないで済むようにリロケータブル・アセンブル形式をとっています．

分割した全モジュールのアセンブル終了後にリンカを使ってそれらを結合し，実際のアドレスを割付けます．

図２－１　リンク作業手順



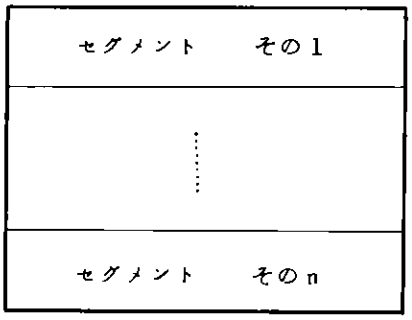
第３章　ソース・プログラム様式

３．１　ソース・プログラムの一般的な構成

μPD70108，μPD70116のソース・プログラムは論理的にはセグメントと呼ばれるブロックで構成します．（４.１.１　参照）

セグメントは一般には機能的または配置アドレス的に同種のルーチンまたはデータで構成します（例えば，コード，データ，…）

ソース・プログラム＝セグメントの集り



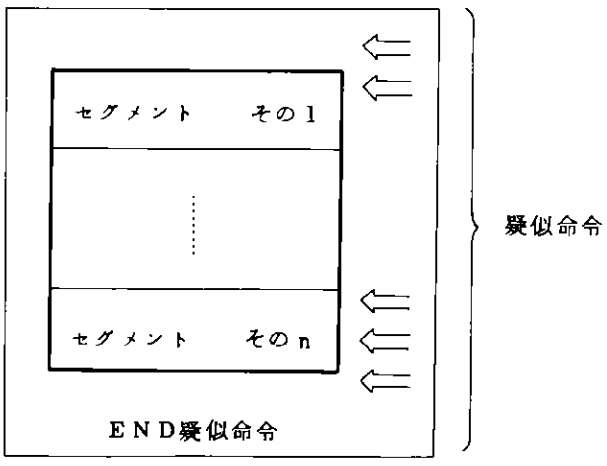
一つのセグメント内のシンボルは，セグメント・レジスタに参照先のセグメントのパラグラフ値（セグメントの先頭番地／１０Ｈ）をセットしてから初めて参照できます．また，セグメントのサイズは６４Ｋバイト以下という制約があります．

セグメントはμPD70108，μPD70116リンカで任意のアドレスに配置できます．ただし，絶対番地指定をしたアブソリュート・セグメントの場合はソース・プログラムで指定したアドレスから変更することはできません．

セグメントの内または外にアセンブラに対する指示文，すなわち疑似命令を置いて適切なオブジェクト・プログラムを生成するようにします．

ソース・プログラムの最後にはＥＮＤ疑似命令を置いて，アセンブラにアセンブルする範囲を指定します．（４.９　参照）

ソース・プログラム



3．1．1　疑似命令の記述できる位置

疑似命令は，記述できる位置から言って次の3種類に分類されます．各疑似命令の詳しい説明は第4章を参照してください．

(1)　セグメントの外にのみ記述できる疑似命令

END

(2)　セグメントの内にのみ記述できる疑似命令[注(2)]

PROC ～ ENDP，ORG，DB，DW，DD，DBS，DWS，DDS

LABEL，EVEN

(3)　任意の位置に記述できる疑似命令

SEGMENT～ENDS，STRUC～ENDS，RECORD，PUBLIC，EXTRN[注(1)]，ASSUME

EQU，PURGE，NAME，GROUP

なお，機械語命令は(2)と同様にセグメントの内でのみ記述できます．[注(2)]

注(1)　EXTRN疑似命令は，その記述する位置に意味があります（ 4.7.3　参照 ）．

注(2)　(2)の疑似命令および機械語命令をセグメントの外で記述した場合は，ディフォールト・セグメントの内で記述したものと見なされます．

ディフォールト・セグメントの名称は？？ＳＥＧです．

例

```
        )
        (
        DB      0
        )
        (
```

は次のように記述されたと見なされます．

```
        )
        (
??SEG   SEGMENT PARA PUBLIC
        DB      0
??SEG   ENDS
        )
        (
```

3．1．2　セグメントの構成

(1)　機械語命令を含むセグメント（ルーチン）

SEGMENT疑似命令で始め，ENDS疑似命令で終了します．（ 4.1.1　参照）

μPD70108，μPD70116ではコードまたはデータをアクセスする場合，そのアドレスはパラグラフ値とオフセット値で指定されます．

たとえば，コードはＰＳ（パラグラフ値）とＰＣ（オフセット値）でそのアドレスが指定されます．アセンブラはアクセスされるデータが，ASSUME疑似命令で，現在セットしてあるセグメント・レジスタの値でアクセス可能であるか否かをチェックします．

上記のチェックのために，現在セットしてあるセグメント・レジスタの値をASSUME疑似命令でアセンブラに指示する必要があります．（ 4.1.8　参照）

セグメント＝疑似命令＋機械語命令

```
セグメント名    SEGMENT    属性指定
                ASSUME  CS：セグメント名[ ,… ]
                  )
                  (
                機械語命令，疑似命令
                  )
                  (
セグメント名    ENDS
```

(2)　機械語を含まないセグメント（データ・ブロック等）

(1)と同様にSEGMENT疑似命令で始め，ENDS疑似命令で終了します．（ 4.1.1　参照）

この種類のセグメントには，データのみから成るものとプログラム・リンケージのためにEXTRN疑似命令のみからなるものがあります．（ 4.7.3　参照）

セグメント＝疑似命令のみ

```
セグメント名    SEGMENT    属性指定
                  )
                  (
                疑似命令
                  (特に，領域確保疑似命令，
                        プログラム・リンケージ疑似命令)
                  )
                  (
セグメント名    ENDS
```

3．1．3　ソース・プログラムの一般的な形式

ソース・プログラムの一般的な形式は次のようになります．

		ソース・プログラムの先頭にある方が望ましい疑似命令 （NAME，PUBLIC，EXTRN，EQU STRUC～ENDS，RECORD）
セグメント	参照する外部シンボル名の宣言	SEGMENT疑似命令 EXTRN疑似命令 ENDS疑似命令
	データ，スタック等	SEGMENT疑似命令 領域確保疑似命令 またはそれに準ずる命令（DB，DW，DD，DBS，DWS，DDS，LABEL） その他の疑似命令（ORG，EVEN） ENDS疑似命令
	ルーチン	SEGMENT疑似命令 ASSUME疑似命令 および機械語命令 PROC ～ ENDP疑似命令 その他の疑似命令 （DB，DW，DD，DBS，DWS，DDS，LABEL，ORG，EVEN，PURGE） ENDS疑似命令
		END疑似命令

ソース・プログラムに記述できるのは上記の疑似命令，機械語命令のほかにコントロールがあります．

コントロールについては第6章を参照してください．

3.2 文

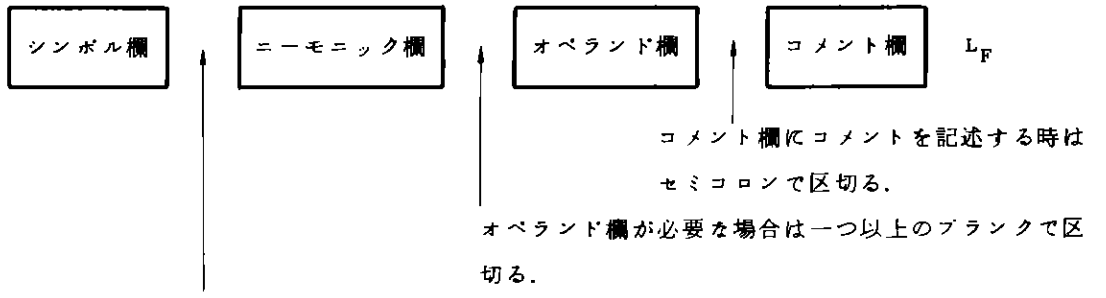
3.1の様に，ソースプログラムは疑似命令とセグメントから構成しますが，それらは文（Statement）で構成します．

一つの文は1行以内に，3.4で表される文字を使って記述します．

テキスト・エディタを使ってソース・プログラムを作成する場合，各文はC_R（キャリッジ・リターン），L_F（ライン・フィード）でターミネートされますが，アセンブラはこの内L_Fを文の終りと判断し，C_Rを無視して読み飛ばします．

文は下図のように，シンボル欄，ニーモニック欄，オペランド欄およびコメント欄の四つのフィールドで構成します．

各欄はブランク，コロン（：）またはセミコロン（；）で区切り，1行には最大132文字まで書けます．文の記述方法は自由記述形式で，シンボル欄，ニーモニック欄，オペランド欄の順序であれば任意のカラムから書けます．ただし，一つの文は1行以内で記述しなければなりません．



シンボル欄にシンボルを記述する場合は，コロンまたは一つ以上のブランクで区切る．
（コロンかブランクかはニーモニック欄で記述する機械語命令または疑似命令によってきまっています．）
シンボル欄にシンボルを記述しない場合は，任意の数のブランクが書けます．コロンおよびセミコロンの前後には任意の数のブランクが書けます．

ソース・プログラムの例

```
;  SAMPLE PROGRAM
;
        EXTRN  RETURN:FAR
;
SEG1    SEGMENT WORD PUBLIC 'CODE'
        ASSUME PS:SEG1,DS0:SEG2
        MOV     AW,SEG2   ;SET DS0
        MOV     DS0,AW
        MOV     V1,1
LAB1:   CALL    RETURN
SEG1    ENDS
;
SEG2    SEGMENT
        V1      DW   0
```

文

3.3 タビュレーション機能

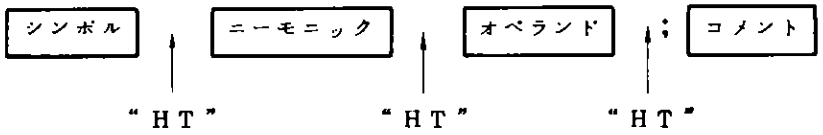
アセンブル・リストを見やすい形式にするため，本アセンブラにはタビュレーション機能が用意されています．タビュレーション機能とは，ソース・プログラムのレーベル欄，ニーモニック欄，オペランド欄，コメント欄がそれぞれ8の倍数のカラムより始まるように整理する機能です．

例

```
SEG1    SEGMENT
        ASSUME    PS:SEG1
        MOV       AW,SEG2    ;SET DS
```

8の整数倍カラム目＋1

タビュレーション動作を行なう場合は，ソース・プログラムのニーモニック欄，オペランド欄の先頭，およびコメント欄の始まりを示すセミコロン(；)の前に“HT”(Horizontal Tab，09H)文字を挿入します．このようにしますと，アセンブル・リスト上の各欄が8の倍数のカラム毎に整理されます．



3.4 文字セット

文は次の文字を使って記述してください．

(1) 英　字

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m
n o p q r s t u v w x y z

(2) 数　字

0 1 2 3 4 5 6 7 8 9

(3) 特殊文字

空白 ? @ , . + − ＊ ／ ( ) $ =

; : ' & # < > ! " % { }

^ ~ | _ ← C_R L_F HT FF NULL DEL

シンボルとして使用できる文字は3.5.2を参照してください．

英小文字をシンボル，予約語の中に使った場合，大文字として解釈されます．

3．4．1 英数字

英字と数字をまとめて英数字といいます．

3．4．2 数　字

2進文字　0，1の2個の文字を2進文字といいます．

8進文字　2進文字と2，3，4，5，6，7の8個の文字を8進文字といいます．

10進文字　8進文字と8，9の10個の文字を10進文字といいます．

16進文字　10進文字とA，B，C，D，E，Fの16個の文字を16進文字といいます．

3．4．3 特殊文字の用途

文　字	名　称	主な用途
	空白	各欄の区切り記号
?	疑問符	不定値記号または英字相当文字
@	単価記号	英字相当文字
_	アンダースコア	英字相当文字
,	カンマ	オペランド間の区切り記号
.	コンマ	シンボルとオフセットの区切り記号
+	プラス	正符号または加算演算子
−	マイナス	負符号または減算演算子
*	アステリスク	乗算演算子
/	スラッシュ	除算演算子
(	左かっこ	演算順序の変更
)	右かっこ	演算順序の変更
$	ドル記号	ロケーション・カウンタの値
=	等符号	レコードの初期値指定記号
;	セミコロン	コメントの開始記号
:	コロン	レーベルと機械語命令の区切り記号

'	引用符	文字定数の開始・終了記号
< >		} 構造体またはレコードによる値設定記号
^		不正文字のアセンブル・リスト上の表現
[ ]		} 配列のオフセット値指定記号またはベース・レジスタ，インデクス・レジスタの指定記号
HT	水平タブ	空白相当文字
LF	ライン・フィード	文の終了記号
CR	キャリッジ・リターン	アセンブラによって無視されます.
NULL	ヌル・コード	アセンブラによって無視されます.
FF	フォーム・フィード	アセンブラによって無視されます.
DEL	デリート・コード	アセンブラによって無視されます.

3.5 シンボル

本アセンブラでのシンボルは次の3種類に分けられます.

(1) ネ ー ム

NAME，EQU，SEGMENT，STRUC，RECORD 疑似命令で定義するシンボルです.
ただし，EQU疑似命令の場合，ネームはそのオペランドと同一のものとされます.

(2) レーベル

機械語命令に割付けるシンボルで，コロン（：）でターミネートするかLABEL疑似命令またはPROC疑似命令で定義します.

レーベルはアドレスと属性を持ちます.

属性：

NEAR……参照（BR，CALL）するのにPCのみを変更すればよいレーベル.
これは，レーベルを定義したセグメント内または同一グループ内でのみ参照できることを表します.

FAR……参照（BR，CALL）するのにPSとPCの両方を変更しなければならないレーベル.
これは，任意のセグメントから参照できることを表します.

これらの属性はアセンブラが適切なオペレーション・コードを生成するのに必要です.（8.5.1参照）

アセンブラは：でターミネートしたレーベルの属性をNEARとみなします.

(3) 変　数

LABEL疑似命令，エリア確保疑似命令（DB，DW，DD，DBS，DW，DDS）またはそれに準ずる命令（構造体名，レコード名）で定義するシンボルで，ブランクでターミネートします．変数はレーベル同様，アドレスと属性を持ちます．

属性：

BYTE……バイト領域であることを表します．

WORD……ワード（2バイト）領域であることを表します．

DWORD……ダブルワード（4バイト）領域であることを表します．

これらの属性はアセンブラが適切なオペレーション・コードを生成するためと，機械語命令のオペランド・チェックに用いられます．

例えば，本アセンブラでは次のようなオペランドの記述は禁止しています．

```
例  VAR01  DB  0       ;VAR01の属性はBYTE
    MOV    AW,VAR01 ;エラー
     …16ビット・レジスタAWにバイト・データを転送
```

3．5．1　属性による適切なオペレーション・コードの生成

シンボルを参照する機械語の適切なオペレーション・コードの生成には，オペランドのシンボル属性のほかにASSUME疑似命令も関係しますので4.1.8も参照してください（セグメント・プリフィックスの自動生成）．

(1) BR系／CALL命令の場合

(a) オペランドが属性NEARのレーベル

セグメント内BR系／CALLのオペレーション・コードを生成します．

(b) オペランドが属性FARのレーベル

セグメント間BR／CALLのオペレーション・コードを生成します．

前方参照である場合はPTR演算子でレーベルの属性をアセンブラに教えてやる必要があります．（前方参照とは，参照するシンボルの定義が参照する文より先（ファイルの終了側）にある場合をいいます．）

(c) オペランドが属性WORDの変数

間接BR／CALL（セグメント内またはグループ内）のオペレーション・コードを生成します．

(d) オペランドが属性DWORDの変数

間接BR／CALL（セグメント間）のオペレーション・コードを生成します．

(e) オペランドがベース・レジスタまたは（および）インデクス・レジスタで示される場合（3.8.1の(4)参照）.

アセンブラは属性が分りませんので，ＰＴＲ演算子でその属性を指示しなければなりません（3.8.2.7 参照）.

```
例  BR  〔BW+IX+4〕              ;エラー
    BR  WORD PTR 〔BW+IX+4〕;OK
```

(2) RET命令の場合

手続き（PROC～ENDPで定義します. 4.1.2参照）内の場合はその属性（NEAR，FAR）に応じて，次のオペレーション・コードを生成します.

NEAR …… セグメント内リターン・コード（0C3Hまたは0C2H）

FAR …… セグメント間リターン・コード（0CBHまたは0CAH）

手続きの外である場合は，無条件にセグメント内リターン・コードを生成します.

(3) その他の参照命令の場合（MOV，XOR，……）

(a) オペランドが属性BYTE，WORD，DWORDの場合

それぞれ，バイト，ワードおよびダブルワード参照のオペレーション・コードを生成します.

(b) オペランドが属性NEAR，FARの場合

レーベルの参照はＢＲ系／ＣＡＬＬ命令以外では，そのままでは使用禁止です.

そこで，PTR演算子で属性の変更を指示してから使います.

```
例  LAB01：XOR  AW，AW
                 )
    MOV   AW，LAB01              ;エラー
    MOV   AW，WORD PTR LAB01  ;OK
```

(c) オペランドがベース・レジスタまたは（および）インデクス・レジスタで示される場合

複数オペランドが必要な機械語命令の場合，アセンブラは残りのオペランドから判断します.

たとえば，MOV AW，〔BW〕は〔BW〕の属性を，AWが16ビット（ワード）レジスタであることからWORDと判断します.

オペランドから判断のつかない場合または一つのオペランドしかない場合はＰＴＲ演算子でその属性を指示しなければなりません.

```
例  MOV 〔BW〕，〔BP〕              ;エラー
    MOV 〔BW〕，WORD PTR 〔BP〕;OK
    DEC 〔BW〕                    ;エラー
    DEC BYTE PTR 〔BW〕        ;OK
```

3．5．2　シンボル記述上の規則

シンボルは次の(1)～(6)の規則を満たしていなければなりません．

(1)　シンボルは英数字および英字相当文字（@，?，_）で構成します．
　　ただし，先頭文字として10進文字（0～9）は使えません．

(2)　シンボルの長さは1～68文字です．
　　64文字以上のシンボルを記述してもかまいませんが，先頭63文字のみが有効となります．

(3)　シンボルとして予約語を使うことはできません（付録1参照）．

(4)　同一シンボルを2度以上定義することはできません．（ただし，セグメント名を除く）

(5)　ネームおよび変数はブランクでターミネートします．

(6)　レーベルは：でターミネートします．（ただし，LABEL疑似命令とPROC疑似命令による場合は空白でターミネートします．）

例(1)　正しいシンボルの例

```
CODE01    SEGMENT
VAR01     DW    0
LAB01     NOP
```

例(2)　誤ったシンボルの例

```
1ABC      EQU     3         … 数字は先頭文字に使えません．
LAB       MOV     AW,CW … レーベルには：が必要です．
FLAG:     DB      0         … 変数には：をつけません．
?         DW      0         … 予約語（?）は使えません．
```

例(3)　シンボルのみから成る文の例

```
ABC:
```

3.6 シンボル欄

シンボル欄にはシンボルを記述します．

シンボル欄にシンボルを記述することを，そのシンボルを定義すると言います．

3.6.1 シンボル記述上の注意

(1) ニーモニック欄に機械語命令を記述する場合は，シンボル欄にラベルを記述してもしなくてもかまいません．

(2) ニーモニック欄に疑似命令を記述する場合，シンボル欄にシンボルが必要なもの，シンボルを記述してもしなくてもよいものおよびシンボルを記述してはいけないものの3種があります．

(a) シンボルを必ず記述しなければならない疑似命令

```
SEGMENT  ,ENDS    ,PROC   ,ENDP
STRUC    ,RECORD  ,LABEL  ,EQU
```

(b) シンボルを記述しても，しなくても良い疑似命令

```
DB       ,DW      ,DD     ,DBS      ,DWS      ,DDS
```

(c) シンボルを記述してはいけない疑似命令

```
EXTRN    ,PUBLIC  ,ORG    ,END
ASSUME   ,PURGE   ,EVEN   ,NAME
```

例(1) 正しいシンボルの例

```
ABC      EQU     3
LAB:     MOV     AW,CW
FLAG     DB      0
```

例(2) 誤ったシンボルの例

```
         LABEL   BYTE  …… シンボルが必要です．
LAB01    ORG     100H  …… シンボルは記述できません．
```

3.7 ニーモニック欄

ニーモニック欄には，機械語命令または疑似命令を記述します．BUSLOCK,REP（およびその他のリピート命令）も他の機械語命令と一緒にこの欄に記述します．

例

```
NOP          …マシン・コード　90Hに変換されます．
DB 3         …コード03Hが生成されます．
REP MOVBK …マシン・コードF3H，A4Hに変換されます．
```

3.8 オペランド欄

オペランド欄には，ニーモニック欄で記述した機械語命令および疑似命令に必要なオペランドを記述します．

機械語命令には，オペランドを記述してはいけないもの，1～2個のオペランドを記述しなければならないものがあります．2個のオペランドを記述する場合，各オペランドをカンマ（，）で区切ります．疑似命令のオペランドは各疑似命令の説明を参照してください．

ここでは機械語命令の場合の説明をします．

3.8.1 オペランド欄の記述形式

オペランドに記述できるデータの形式には次の4種があります．

- 予約語
- シンボル（ネーム，レーベル，変数，レコード・フィールド名，構造体フィールド名等）
- 定　数
- 式

(1) 予 約 語

(a) レジスタ名

AW，AH，AL，BW，BH，BL，CW，CH，CL
DW，DH，DL，SP，BP，IX，IY，PS，DS0
DS1，SS

(b) ロケーション・カウンタ

$

(2) シンボル

シンボル欄で定義したシンボルです．

(3) 定　数

定数には数値定数と文字定数があります．
数値定数はさらに，2進定数，8進定数，10進定数および16進定数に分けられます．

(a) 2進定数

2進文字で構成し，最後に文字Bを付加して表します．

例：　011B …… 10進で3の数値を表します．

(b) 8進定数

8進文字で構成し，最後に文字OまたはQを付加して表します．

例：　037O …… 10進で31の数値を表します．
　　37Q …… 10進で31の数値を表します．

(c) 10進定数

10進文字で構成し，最後に文字Dを付加するか何も付加しないで表します．

例：　128D
　　0128　…… ともに10進で128を表します．

(d) 16進定数

16進文字で構成し，最後に文字Hを付加して表します．
16進定数の先頭文字は10進文字でなければなりません．

例：　13H …… 10進で19の数値を表します．
　　0FFH …… 10進で255の数値を表します．

(e) 文字定数

文字の列（長さは1または2）を引用符で囲んで表します．
文字はパリティ・ビット（MSB）を0とする7ビットASCIIコードに変換されます．
引用符自身を文字として確保するには引用符を2個続けて表します．

例：　'O'
　　'A'''

(4) 式

以下の項を演算子で連結したものです.

- 定　　数
- シンボル（レーベル，変数，セグメント名，グループ名，構造体フィールド名，レコードフィールド名）
- ベース・レジスタ，インデクス・レジスタ（BW，BP，IX，IY）
- 予約語　BYTE，WORD，DWORD，NEAR，FAR
- レコード参照

演算子に関しては8.8.2を参照してください.

シンボルのうち，セグメント名を参照した場合，アセンブラはそのセグメントのパラグラフ値を参照しているものとみなし，グループ名を参照した場合はそのグループに属する先頭のセグメントのパラグラフ値を参照しているものとみなします.

例　MOV　AW，SEGNAM；SEGNAM＝セグメント名は，AWにセグメントSEGNAMのパラグラフ値を転送するオペコードを生成します.

ベース・レジスタ，インデクス・レジスタを式で使う場合，次の形式が許されます.

- 〔ベース・レジスタ名±定数〕
- 〔インデクス・レジスタ名±定数〕
- 〔ベース・レジスタ名±定数〕〔定数〕
- 〔インデクス・レジスタ名±定数〕〔定数〕
- 〔ベース・レジスタ名±定数〕〔インデクス・レジスタ名±定数〕
  または〔ベース・レジスタ名＋インデクス・レジスタ名±定数〕
- 〔ベース・レジスタ名±定数〕〔インデクス・レジスタ名＋定数〕〔定数〕

3.8.2 演　算　子

演算子とは1または複数のデータを入力して1または複数のデータを導く記号です．演算子とそのオペランド（入力データ）を記述したものを式と呼びます.

μPD70108，μPD70116アセンブリ言語の演算子は次の8種類に分けられ，演算の優先順序が定められています．（8.8.2.1　参照）

演算子が英字列から成る場合，演算子の前後に空白が必要です.

説明の便宜のために演算子の左側のオペランドを第1オペランド，右側のオペランドを第2オペランドと呼びます.

第１オペランド　演算子　第２オペランド　（２項演算子）

演算子　オペランド　　（単項演算子）

- 算術演算子

  ＋，－，＊，／，MOD

- 論理演算子

  OR，AND，NOT，XOR

- 比較演算子

  EQ，NE，LT，LE，GT，GE

- シフト演算子

  SHR，SHL

- バイト分離演算子

  LOW，HIGH

- 属性変更演算子

  PTR，：，SHORT，THIS

- 属性返却演算子

  SEG，OFFSET，TYPE，LENGTH，SIZE，MASK，WIDTH

- その他の演算子

  （），〔〕，．，<>

３．８．２．１　演算子の優先順序

優先順序は次のように定められていますが，（）を使うことによって演算順序を変更することができます（3.8.2.9　参照）．

式中に同一優先順位の演算子がある場合は左から演算されます．

次の表は優先順序の高い方から記述してあります．

(1)　（），<>，〔〕，．，LENGTH，SIZE，WIDTH，MASK

(2)　PTR，SEG，OFFSET，TYPE，THIS，：

(3)　HIGH，LOW

(4)　＊，／，MOD，SHR，SHL

(5)　＋，－

(6)　EQ，NE，LT，LE，GT，GE

(7)　NOT

(8)　AND

(9)　OR，XOR

(10)　SHORT

3．8．2．2 算術演算子

演算はすべて１７ビットで行なわれ，結果は１６ビットの値となります．

(1) ＋

第１オペランドと第２オペランドの値の和を返します．

単項演算子である場合は意味をもちません．

例　２＋３（＝５），＋２

(2) －

第１オペランドの値から第２オペランドの値を引いた値を返します．

単項演算子である場合は２の補数をとった値を返します．

例　３－２（＝１），－２

(3) ＊

第１オペランドの値と第２オペランドの値の積を返します．

例　３＊２（＝６）

(4) ／

第１オペランドの値を第２オペランドの値で割った値の整数部を返します．

例　５／２（＝２）

(5) MOD

第１オペランドの値を第２オペランドの値で割った余りを返します．

A MOD B＝A－（A／B）＊B

例　5 MOD 2(=1)

3．8．2．3 論理演算子

ビット毎の論理演算を行ないます．

(1) OR

オペランド間の論理和をとった値を返します．

例　１２３４H OR ５６７８H （＝５６７ＣＨ）

(2) AND

オペランド間の論理積をとった値を返します．

例　１２３４H AND ５６７８H （＝１２３０Ｈ）

(3) NOT

オペランドの論理否定をとった値を返します．

例　NOT １２３４H （＝－１２３５Ｈ）

(4) XOR

オペランド間の排他的論理和をとった値を返します．

例　１２３４H XOR ５６７８H （＝４４４ＣＨ）

3．8．2．4 比較演算子

オペランド間の比較を行ない，結果が真なら－1を，偽なら0を返します．

(1) EQ （EQUAL）

第1オペランドと第2オペランドの値が等しいときに－1，等しくないときに0を返します．

例 1234H EQ 3456H （=0）

(2) NE （NOT EQUAL）

第1オペランドと第2オペランドの値が等しくないときに－1，等しいときに0を返します．

例 1234H NE 3456H （=−1）

(3) LT （LESS THAN）

第1オペランドの値が第2オペランドの値より小さいときに－1，大きいまたは等しいときに0を返します．

例 1234H LT 5678H （=−1）

(4) LE （LESS THAN OR EQUAL）

第1オペランドの値が第2オペランドの値より小さいかまたは等しいときに－1，大きいときに0を返します．

例 1234H LE 5678H （=−1）

(5) GT （GREATER THAN）

第1オペランドの値が第2オペランドの値より大きいときに－1，小さいかまたは等しいときに0を返します．

(6) GE （GREATER THAN OR EQUAL）

第1オペランドの値が第2オペランドの値より大きいかまたは等しいときに－1，小さいときに0を返します．

例 1234H GE 5678H （=0）

3．8．2．5 シフト演算子

右または左シフトします．

(1) SHR （SHIFT RIGHT）

第1オペランドを第2オペランドの値のビット数だけ右シフトした値を返します．

MSB側にはシフトしたビット数の0が入ります．

例 1234H SHR 4 （=0123H）

(2) SHL （SHIFT LEFT）

第1オペランドを第2オペランドの値のビット数だけ左シフトした値を返します．

LSB側にはシフトしたビット数の0が入ります．

例 1234H SHL 4 （=2340H）

3．8．2．6 バイト分離演算子

2バイトまたは1バイトデータの下位または上位1バイトの値を返します．

(1) LOW

下位1バイトの値を返します．

一般形：

LOW 式

例 LOW 1234H （=0034H）

(2) HIGH

上位1バイトの値を返します．

一般形：

HIGH 式

例 HIGH 1234H （=0012H）

注) この演算子のオペランドがリロケータブルなシンボルの場合，さらにこれをオペランドとした式は記述できません．

例 3＋LOW SYM1 ；エラー

3．8．2．7　属性変更演算子

変数およびレーベルはアドレス（パラグラフ値とオフセット値）と属性を持ちますが，この演算子は属性を一時的に変更（または指定）するものです．たとえば，バイト変数VAR01に続く2バイトをAWへ転送する場合

```
MOV  AW,VAR01
```

はエラーですが

```
MOV  AW,WORD  PTR  VAR01
```

としますとAWに2バイト転送するオペレーション・コードが生成されます．

(1)　PTR

第2オペランドが変数名またはラベルの場合，それらの属性のみを一時的に第1オペランドで示す属性に変更（または指定）します．

第2オペランドが数の場合，オフセットをその数自身に，属性を第1オペランドで示す属性に指定します．この場合，所属セグメントは未定義ですので，次の(2)の：演算子を使って所属セグメントを明示する必要があります．

一般形：

型　PTR　{ 変数 / レーベル / 数* }

型＝｛NEAR｜FAR｜BYTE｜WORD｜DWORD｝

例
```
MOV  AW,WORD  PTR  VAR01
MOV  〔BW〕,WORD  PTR  〔BP〕
```

備考　アセンブラは，型　PTR　数の場合，“数”で示される番地の参照とみなします．

本形式の式の場合，どのセグメントに相対する番地であるかを(2)の：演算子で指示する必要があります．

例
```
MOV  AW,DS0:WORD  PTR  8
```

(2)　：

第2オペランドで表わされる変数の属するセグメントを一時的に第1オペランドで示すセグメントに変更します（または第1オペランドで示すセグメントに属することをアセンブラに通知します）．セグメント・レジスタ名を記述した場合，アセンブラはそのセグメント・レジスタにASSUMEされているセグメントに第2オペランドで示される変数が属するものとみなします．

一般形：

セグメント・レジスタ名：アドレス式　……形式　1

　または

セグメント名：アドレス式　……形式　2

グループ名：アドレス式　……形式　2

形式2の場合，

セグメント名またはグループ名はセグメント・レジスタのどれかにASSUMEされている必要があります（アセンブラはセグメント名またはグループ名をセグメント・レジスタに置き換えます）.

例　MOV　AW,DS1:VAR02　;VAR02の属するセグメントをDS1にASSUMEしてあるセグメントまたはグループとする.

(3) SHORT

相対アドレスが1バイト（−128〜+127）で表せることを宣言します.

ただし，後方参照である場合および外部参照シンボルの参照である場合は無視されます.

後方参照——相対アドレスの値によって1バイトまたは2バイトの値とします.

外部参照シンボルの参照—— 2バイトの値とします.

一般形：

```
SHORT アドレス式
```

例

```
LAB01: NOP
       NOP
       BR   LAB01          … ショート BR （2バイト）
       BR   SHORT LAB02 … ショート BR （2バイト）
LAB02: NOP
```

(4) THIS

カレント・ロケーションの値をアドレスとする変数またはレーベルを一時的に生成します.

一般形：

```
THIS {NEAR|FAR|BYTE|WORD|DWORD}
```

生成された変数またはレーベルのセグメント，オフセットおよび属性は次のようになります.

- セグメント……このTHIS演算子を使っている文を含むセグメント
- オフセット値……カレント・ロケーション・カウンタの値
- 属　性　……オペランドで指定した属性

例　BR　THIS NEAR　（=BR　$）

3．8．2．8 属性返却演算子

(1) SEG

オペランドで示す変数またはレーベルのパラグラフ値を返します．

一般形：

SEG ｛変数｜レーベル｝

例 SEG LAB01

(2) OFFSET

オペランドで示す変数，レーベルまたは構造体フィールドのオフセット値を返します．

一般形：

OFFSET ｛変数｜レーベル｜構造体フィールド名｝

例 OFFSET LAB01

OFFSET GROUP1：LAB01

GROUP1はグループ名

(3) TYPE

オペランドで示す変数，レーベル，構造体フィールド，構造体またはレコードの属性を値で返します．

NEAR ……－1 （0FFHまたは0FFFFH）

FAR ……－2 （0FEHまたは0FFFEH）

BYTE …… 1

WORD …… 2

DWORD…… 4

構造体名で

領域確保した変数……確保した基本バイト数（DUPなし，または1 DUPに相当）

レコード名で

領域確保した変数……1または2（フィールド幅の総計による）

構造体名……構造体を構成する領域のバイト数を返します．

レコード名……レコードのフィールド幅の総計によって異なります．

1以上8以下……1

9以上16以下……2

一般形：

TYPE ｛変数名｜レーベル｜構造体フィールド名｜構造体名｜レコード名｝

(4) LENGTH

オペランドで示す変数または構造体フィールドの要素数を返します．

オペランドが構造体名またはレコード名である場合は1を返します．

一般形：

LENGTH ｛変数名｜構造体フィールド名｜構造体名｜レコード名｝

例
```
VAR01  DW  100H DUP (0H)
       MOV   CW,LENGTH VAR01  ;=100H
```

(5) ＳＩＺＥ

オペランドで示す変数，構造体フィールド，構造体またはレコードのバイト数を返します．

一般形：

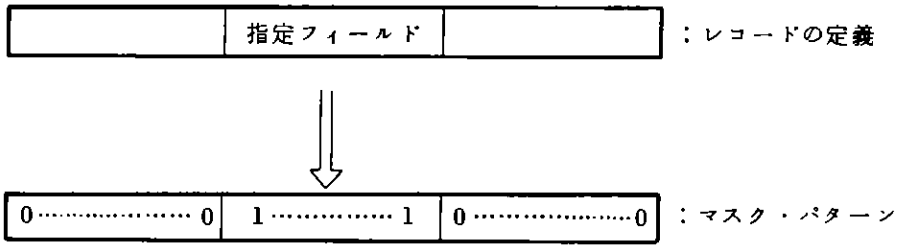
ＳＩＺＥ ｛変数名｜構造体フィールド名｜構造体名｜レコード名｝

例　VAR01 DW 100 DUP (0H)

ADD BW,SIZE VAR01 ;=200H

(6) ＭＡＳＫ

オペランドで示すレコード・フィールド名に対応したマスク・パターンを返します．マスク・パターンは対応するフィールドがすべて１，その他のフィールドがすべて０のパターンです．



一般形：

MASK レコード・フィールド名

例　REC01 RECORD FLD1:4=3,FLD2:12=12H

XOR AW,MASK FLD1 ;0F000H

(7) ＷＩＤＴＨ

オペランドで示すレコード名またはレコード・フィールド名に対応して，それぞれフィールド幅の総計またはフィールド幅数をビット数で返します．

一般形：

WIDTH ｛レコード名｜レコード・フィールド名｝

例　REC01 RECORD FLD1:4=8,FLD2:11=12H

MOV AW,WIDTH REC01 ;=15

MOV AW,WIDTH FLD1 ;=4

３．８．２．９　その他の演算子

(1) (　)

演算の順序を変更します．

演算の実行順序は予め定まっていますが，そのとおりには演算したくないときに用います．

一般形：

（式）

（）内の式の演算を先に行ないます．

（）が多重になっている場合は一番内側の（）内の式から演算します．

(2)

第1オペランドで示される変数またはレジスタ式のオフセット値に第2オペランドのSTRUCからのオフセット値を加算してオフセット値とします.

変数の属性は構造体フィールドの属性となります.

一般形：

{変数名｜レジスタ式}．構造体フィールド名

例
```
STR  STRUC
     F1  DB  0,8    ;オフセット＝0，属性＝バイト
     F2  DW  4      ;オフセット＝2，属性＝ワード
STR  ENDS
     MOV  〔BW〕,VAR01.F2        ;=VAR01+2(ワード転送)
     MOV  〔BW+1〕〔IX+2〕〔8〕.F1,8;=(バイト転送)
```

(3) 〔　〕（配列参照）

第1オペランドで示される変数のオフセット値に第2オペランドの値を加算してオフセット値とします.

変数の属性は変化しません.

一般形：

変数名〔数値〕

例
```
MOV  AW,VAR01〔2〕 ; =VAR01+2
```

(4) ＜　＞

オペランドで示すレコード名のフィールドの定義どおり（4.5.2）に＜＞内での各フィールド値をセットした値を返します．返す値の属性（BYTE，WORD）はレコードの定義における，フィールド幅の総計で与えられたレコード名の属性によります（4.5.6）.

フィールド幅の総計が

1以上8ビット以下 ⟶ バイト

9 〃 16ビット 〃 ⟶ ワード

一般形：

レコード名＜各フィールドの値リスト＞

レコードの定義で指定した初期値を使いたい場合は“，”のみを記述してください.

例　＜，，3＞…第1，第2フィールドの値は，レコード定義で指定した初期値になります.

第3フィールドの値は3になります.

値リストがレコードの定義で宣言したフィールドの数より少ない場合，残りのフィールドは定義したときの初期値になります.

例
```
REC01  RECORD  FLD1：4=4,FLD2：4=7
MOV  AL,REC01<5>  ;＝57H
```

逆に，値リストがレコードの定義で宣言したフィールドの数より多い場合は余分な（値リストの右側）フィールド値は捨てられます.

例
```
REC01  RECORD  FLD1：4=4,FLD2：4=7
MOV    AL,REC01<5,3,2>;=REC01<5,3>
```

3．8．3 演算子のオペランドに関する制限

演算子の種類とオペランドの種類の組み合わせは制限されてます．

例えば，VAR01，VAR02を変数とすると，

```
MOV  AW,VAR01+VAR02
```

は許されません．

オペランドがアブソリュートな数の場合，次の演算子は自由に使えます．

- 算術演算子
- 論理演算子
- 比較演算子
- シフト演算子

} オペランドがアブソリュートな数の場合にのみ使えます．

算術演算子の内，＊，／，MODもオペランドがアブソリュートな数の場合にのみ使える演算子です．

残る演算子のうち，次の演算子は記述できるオペランドが既に説明してありますので省きます．（8.8.2 参照）

- バイト分離演算子
- 属性変更演算子
- 属性返却演算子
- その他の演算子

あとに残る演算子は算術演算子の＋，－となるわけですがこれに関しては次の表を参照してください．

×は不可を表します．

A＋B：

A ＼ B	アブソリュートな数注(1)	リロケータブルな数注(2)	変数またはラベル
アブソリュートな数注(1)	アブソリュートな数注(1)	リロケータブルな数注(2)	変数またはラベル
リロケータブルな数注(2)	リロケータブルな数注(2)	×	×
変数またはラベル	変数またはラベル	×	×

A－B：

A ＼ B	アブソリュートな数	リロケータブルな数注(2)	変数またはラベル
アブソリュートな数注(1)	アブソリュートな数注(1)	×	×
リロケータブルな数注(2)	リロケータブルな数注(2)	アブソリュートな数注(1)または×注(3)	×
変数またはラベル	変数またはラベル	×	アブソリュートな数注(1)または×注(3)

注(1) アブソリュートな数
普通の数値，例えば０３Ｈ等

注(2) セグメント名またはグループ名（パラグラフ値とみなされる），SEG，OFFSETで導かれる数です．

注(3) Ａ，Ｂ共に同一のモジュールかつ同一のセグメントに属する同一属性のシンボルの場合のみ可能です．

第４章　疑似命令

４．１　プログラム構成疑似命令

４．１．１　SEGMENTとENDS

(1) 機　　能

セグメントを定義します．

SEGMENTとENDSは常に対で使用してください．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
セグメント名	SEGMENT	〔境界属性〕〔結合属性〕〔'クラス名'〕	〔;コメント〕
	⎫ ⎬ ⎭	機械語，疑似命令	
セグメント名	ENDS		

(3) 説　　明

(a) オペランド欄で指定する属性をもつセグメントを定義します．

(b) 境界属性は，リンカでこのセグメントにアドレスを割付けるときのアドレス境界条件です．

境界属性は，何も記述しないか，次の五つの予約語の一つを記述します．

BYTE …バイト境界に割付けます．（すなわち無条件）

WORD …偶数バイト境界に割付けます．

PARA …１０Ｈの整数倍のアドレスを割付けます．

PAGE …１００Ｈの整数倍のアドレスに割付けます．

INPAGE…セグメントがページ境界（１００Ｈの整数倍アドレス）をまたがらないようにアドレスを割付けます．

（つまりセグメントのサイズは１００Ｈバイト以下になるようコーディングしなければなりません．）

境界属性を省略した場合はＰＡＲＡとみなされます．

(c) 結合属性は，同じセグメント名でかつ同じクラス名のセグメント（同名セグメントと呼びます）がリンク時点で複数あった場合，それらをどのように結合するかを表します．

結合属性は，何も記述しないか次の五つの予約語の一つを記述します．

PUBLIC　…アドレスが連続となるように結合します．

COMMON　…各セグメントの先頭アドレスが同一となるように結合します．

AT　絶対式…絶対式の値＊１０Ｈのアドレスを割付けます．（アブソリュート・セグメント）

STACK　…各セグメントの最終アドレスが同一，かつ結合後のセグメントのサイズが各セグメントのサイズの和となるようにアドレスを割付けます．

MEMORY　…各セグメントの先頭アドレスが同一，かつMEMORY属性以外のすべてのセグメントの直後のアドレスを割付けます．

上記の予約語を記述しない場合，リンクの時点で同名セグメントがあればエラーとなります．

(d) クラス名は(c)の結合属性と組合せて，セグメントへの割付けアドレスをコントロールします．

(c)で説明しましたように，同名セグメントに対してリンカは特別の扱いを行ないます．また，リンカでこのクラス名を指定して，同じクラス名をもつセグメントに対して割付けアドレスを指定することもできます．

(e) 同じソース・モジュール内では，同じセグメント名のセグメントは境界属性，結合属性およびクラス名が同じでなければなりません．

同じセグメント名のセグメントで境界属性，結合属性またはクラス名を省略した場合は最初に記述したセグメントのそれがとられます．

アセンブラは同じソース・モジュール内にある同じセグメント名のセグメントを，単に分割されて記述してあるものとみなして一つのセグメントとしてオブジェクト・モジュールに出力します．

例

```
SEG1   SEGMENT   BYTE   PUBLIC   'CODE'
          Ⓐ
SEG1   ENDS
          ⎫
          ⎭
SEG1   SEGMENT   BYTE   PUBLIC   'CODE'
          Ⓑ
SEG1   ENDS
```

は次のように記述した場合と同じに扱われます．

```
          ⎫
          ⎭
SEG1   SEGMENT   BYTE   PUBLIC   'CODE'
          Ⓐ
          Ⓑ
SEG1   ENDS
```

(f) セグメントの，記述上のネスティングができます．

ただし，完全な入れ子になっていなければなりません．アセンブラはネストされている各々のセグメントを別々のセグメントとして扱います．

例

```
SEG1    SEGMENT
            Ⓐ
SEG2    SEGMENT
            Ⓑ
SEG2    ENDS
            Ⓒ
SEG1    ENDS
```

は次のように記述した場合と同じになります．（アセンブル・リストには，記述したとおりに印字されます．）

```
SEG2    SEGMENT
            Ⓑ
SEG2    ENDS
SEG1    SEGMENT
            Ⓐ
            Ⓒ
SEG1    ENDS
```

(4) 使 用 例

```
CODE01  SEGMENT
        ASSUME  PS:CODE01
        MOV     AW,SEG STACKE
        MOV     SS,AW
        MOV     SP,OFFSET STACKE
          ⎛
          ⎝
CODE01  ENDS
```

4．1．2 PROCとENDP

(1) 機 能

手続き（サブルーチン）を定義します．

PROCとENDPは常に対で使用してください．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
手続き名	PROC	[{NEAR / FAR}]	[；コメント]
	⎫		
	⎬ 機械語命令，疑似命令		
	⎭		
手続き名	ENDP		

(3) 説　明

(イ) 手続きはセグメントの中（SEGMENT～ENDS）でのみ定義できます．

(ロ) オペランド欄で，手続きの属性を定義します．

NEAR： この手続きを含むセグメント内またはグループ内からのみ参照できるサブルーチンであることを表します．

FAR ： 任意のセグメントから参照できる手続きであることを表します．

アセンブラは，オペランド欄に何の記述もない場合，NEARとみなします．

(ハ) 本疑似命令をネストして記述できます．

ネストする場合，完全入れ子になっていなければなりません．

(4) 使 用 例

```
CLEAR   PROC   FAR
        XOR    AW,AW
LOOP1:  MOV    [BW],AW
        ADD    BW,2
        DBNZ   LOOP1
        RET
CLEAR   ENDP
```

4．1．3 ASSUME

(1) 機　能

セグメント・レジスタにセットしてある，セグメントまたはグループのパラグラフ値をアセンブラに指示します．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
	ASSUME	{セグメント・レジスタ名：セグメント指定 / セグメント・レジスタ名：グループ名}[，…]	[；コメント]
	または		
	ASSUME	NOTHING	

(3) 説　明

(a) レーベルを定義する場合，プログラムの実行時（RUNタイム時）のPSの値をASSUME疑似命令で指示しなければなりません．

アセンブラはBR系／CALL命令がそのレーベルを参照しようとするとき，参照可能か否かをチェックします．

参照可能である条件は次のとおりです．

－NEAR レーベルに対して

参照する命令のあるセグメントでPSにASSUMEされているパラグラフ値と参照先ラベルの定義してあるセグメントでPSにASSUMEされているパラグラフ値が一致していなければなりません．

－FAR レーベルに対して

常に参照可能です．ただし，前方参照である場合，PTR演算子を使ってFARレーベルであることを明示しなければなりません．

(b) 変数を参照する機械語命令があると，アセンブラはその変数が参照可能であるか否かをチェックします．

参照可能であるためには，その変数の属するセグメントのパラグラフ値が，どれかのセグメント・レジスタにセットしてある必要があります．

この疑似命令で，アセンブラは参照する変数の属するセグメントのパラグラフ値がどのセグメント・レジスタにセットしてあるかを知り，参照可能か否かのチェックおよび適切なセグメント・プリフィックス（PS：，DS0：，DS1：，SS：）の生成を行ないます．ただし，前方参照の場合はセグメント・プリフィックスの自動生成を行ないませんので，必要なセグメント・プリフィックスがDS0，SS（ハードウェア・ディフォルト）でない場合はセグメント・プリフィックスを記述してください（8.8.2.7　参照）

(c) セットすべきセグメント・レジスタ名としては，PS，DS0，DS1，SSの4種類があります．

(d) セグメント指定として，次の3種類があります．

i) セグメント名

ii) SEG　シンボル名

iii) NOTHING

SEGはシンボルの属するセグメントのパラグラフ値を導く演算子です．

NOTHINGは以前のセグメント指定を解消します．

(e) ASSUME NOTHINGはすべてのセグメント指定を解消します．

すなわち，

ASSUME PS：NOTHING，DS0：NOTHING，DS1：NOTHING，SS：NOTHING

と同じ効果をもちます．

(f) ASSUME疑似命令によるアセンブラへの指示は，別のASSUME疑似命令を記述するまで有効です．

(g) 本疑似命令のオペランドにグループ名を記述する場合，グループは既に定義されていなければなりません．

(4) 使 用 例

```
ASSUME  PS：CODE01,DS0：DATA01,DS1：SEG  SYM02
        MOV   AW,SYM01
        MOV   BW,SYM02
```

4.2 グループ定義疑似命令

4.2.1 GROUP

(1) 機　能

グループに属するセグメント名を宣言します.

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
グループ名	GROUP	セグメント名[,…]	[;コメント]

(3) 説　明

(a) グループのサイズ（グループ内の先頭セグメントの先頭アドレスから最終セグメントの最終アドレスまでのオフセット）は64Kバイト以下でなければなりません.（64Kバイトを越えるとリンカでエラーとなります）

(b) 本疑似命令でグループ宣言をし，ASSUME疑似命令でセグメント・レジスタとグループ名を対応付けておきますと，グループ内の参照がセグメント内の参照と同じに扱われます.（使用例　参照）

(4) 使 用 例

```
CGROUP  GROUP   CODE1,CODE2
;
CODE1   SEGMENT WORD 'CODE'
        ASSUME  PS:CGROUP
          ⋮
PROC1   PROC    NEAR  ←──┐  セグメント間（グループ内）で
          ⋮               │  NEAR CALLができます.
PROC1   ENDP            │  （3バイト命令）
          ⋮               │
CODE1   ENDS            │
;                       │
CODE2   SEGMENT WORD 'CODE'
        ASSUME  PS:CGROUP
          ⋮               │
        CALL    PROC1 ──┘
          ⋮
CODE2   ENDS
```

4.3 シンボル制御疑似命令

4.3.1 EQU

(1) 機　　能

ネームを定義します.

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
ネーム	EQU	式 / レジスタ名 / 構造体フィールド名 / レコード・フィールド名	〔;コメント〕

(3) 説　　明

ネームはオペランド欄で記述した式の内容に応じて，機械語命令または疑似命令のオペランドに記述できます.

(4) 使 用 例

```
NAME01  EQU  SYM01
NAME02  EQU  〔BW+IY+2〕
NAME03  EQU  DS1
NAME05  EQU  REC01<1,2,3>
NAME06  EQU  ABC+DEF
```

4.3.2 LABEL

(1) 機　　能

レーベルまたは変数を定義します.

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
レーベルまたは変数	LABEL	NEAR / FAR / BYTE / WORD / DWORD / 構造体名 / レコード名	〔;コメント〕

(3) 説　明

現在のロケーション・カウンタの値をアドレスとするレーベルまたは変数を定義します．

変数／レーベルの属するセグメントは，現在のセグメントとなります．

(a) レーベルを定義する場合はオペランドにＮＥＡＲまたはＦＡＲを記述してください．（ 8.5 参照）

(b) 変数を定義する場合はオペランドに BYTE，WORD，DWORD，構造体名またはレコード名を記述してください．（ 8.5 参照）

構造体名またはレコード名を記述する場合は ，既に定義したものでなければなりません．

(4) 使 用 例

```
SYM01   LABEL   WORD
SYM02   DB      0
        )
        (
        MOV     AW,SYM01
        MOV     AL,SYM02
```

4．3．3 PURGE

(1) 機　能

シンボルをシンボル・テーブルから削除します．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
	PURGE	シンボル名〔,…〕	〔;コメント〕

(3) 説　明

(a) 削除するシンボルは，既に定義してあるシンボルでなければなりません．

削除できるシンボルの種類は次のとおりです．

- 変数名(注1)
- レーベル名(注1)
- ＥＱＵで定義したシンボル(注1)
- 構造体フィールド名
- レコード・フィールド名

（注1） PUBLICまたはEXTRN疑似命令で宣言したシンボルを除きます．

(b) 削除後は再定義しないかぎり参照することはできません．

(4) 使 用 例

```
PURGE SYM01,SYM02
```

4.4 ロケーション・カウンタ制御疑似命令

4.4.1 ORG

(1) 機　能

ロケーション・カウンタの値（セグメント内相対）を設定します.

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
	ORG	式	〔;コメント〕

(3) 説　明

(ア) 式の値はアブソリュートまたはリロケータブルな数でなければなりません.
つまり外部参照名を含んではいけません.

(イ) 式の項にシンボルを記述する場合，シンボルは既に定義されていなければなりません.

(ウ) 現在のロケーション・カウンタより小さいものを指定してはいけません.

(4) 使 用 例

```
ORG  100H
ORG  OFFSET $+10H
```

4.4.2 EVEN

(1) 機　能

ロケーション・カウンタの値を偶数にします.

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
	EVEN		〔;コメント〕

(3) 説　明

(a) 境界属性がBYTEまたはINPAGEのセグメント内では記述できません.
これは，境界属性がBYTEの場合にはそのセグメントの配置アドレスが偶数になるか奇数になるかわからないため，ロケーション・カウンタの値を偶数にしても意味をもたないためです.

(b) ロケーション・カウンタの値が奇数である場合，NOP命令（90H）が生成されます.

(4) 使 用 例

```
EVEN
```

4．5　領域定義疑似命令

4．5．1　STRUCとENDS

(1) 機　　能

構造体を定義します．

(2) 記述形式

シンボル名	ニーモニック欄	オペランド欄	コメント欄
構造体名	STRUC		〔；コメント〕
〔フィールド名〕	{DB|DW|DD}	式1 〔,…〕 式2 DUP(式1 〔,…〕)	〔；コメント〕
〔フィールド名〕	{DBS|DWS|DDS}	式1	〔；コメント〕
構造体名	ENDS		〔；コメント〕

(3) 説　　明

(a) 構造体とは領域のサイズおよびフィールド名のバイト・オフセット，属性（BYTE，WORD，DWORD）のみを定義しているもので実体（コード／データ）はありません．

つまり，メモリ上に領域を確保するものではありません．

(b) 構造体名は領域の確保に，フィールド名は機械語命令でのデータ参照におけるオフセットおよび属性指定（BYTE，WORD，DWORD）に使います（3.8.2.9，4.6.4 参照）．なお，フィールド名，定義方法は変数と同じですが，変数名ではありません．

(c) DB，DW，DD，DBS，DWS，およびDDS疑似命令の記述方法は4.6と同じです．ただし，オペランドにシンボルを記述する場合は既に定義または宣言（EXTRN疑似命令による）してあるシンボルでなければなりません．

オペランドに記述した式の値は，構造体名で領域確保したときの初期値となります．また，DB，DW，DD疑似命令に記述したオペランドが一つの場合に限って，のちに領域確保するときに初期値を変更することができます．

(d) 一つのDBまたはDBS疑似命令で定義された領域をバイト・フィールドと呼びます．

同様にDWまたはDWS疑似命令DDまたはDDS疑似命令で定義された領域をそれぞれワード・フィールド，ダブル・ワード・フィールドと呼びます．

(e) 領域確保の方法は，4.6.7を参照してください．

(4) 使 用 例

```
STR1    STRUC
S1      DB      0,1,'HELLO'     ;オフセット0
S2      DW      0,4,SYM01       ;オフセット7
S8      DD      SYM02,4         ;オフセット0DH
STR1    ENDS    ──────┘
                初期値
```

4．5．2 RECORD

(1) 機　　能

レコードを定義します．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
レコード名	RECORD	フィールド名：フィールド幅〔＝初期値〕〔，…〕	〔；コメント欄〕

(3) 説　　明

(a) レコードとは領域のサイズ（1バイトまたは2バイト），フィールド名のビット・オフセットおよびビット幅のみを定義しているもので，構造体同様実体（コード／データ）はありません．

(b) レコード名は領域の確保に，フィールド名は定数の定義に使います．（4.6.8，8.8.2.8，8.8.2.9 参照）

(c) オペランドに初期値を記述した場合，その値はのちにレコード名を使って領域確保するときの初期値となります．

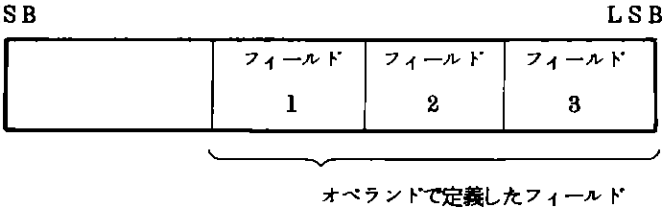
この初期値は領域確保するときに変更することもできます．

(d) フィールド幅の合計は16を越えることはできません．

フィールド幅の合計が8以下のときはバイト・フィールド，9以上16以下のときはワード・フィールドとなります．これは，変数定義（領域を確保したとき：4.6.8）の変数名がバイト，ワードの属性をもつことになります．

(e) フィールド幅の合計が8または16でないとき，バイトまたはワード領域のＬＳＢ側に詰めてフィールド幅ずつの領域が定義され，ＭＳＢ側の空いているフィールドは値0になります．

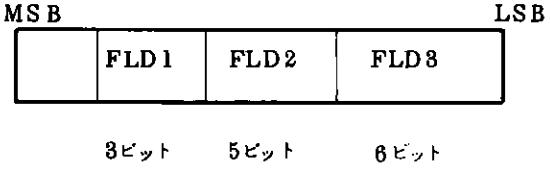
例



(f) 領域確保の方法は，4.6.8 を参照してください．

(4) 使 用 例

```
REC01   RECORD   FLD1:3=7,FLD2:5=10H,FLD3:6
```



4.6 領域確保疑似命令

4.6.1 DB

(1) 機　能

変数の定義およびバイト領域を確保します.

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
〔変数名〕	DB	式1〔,…〕 式2 DUP(式1〔,…〕)	〔;コメント〕

(3) 説　明

(a) オペランド欄に記述した式1の値をもつ領域を確保して指定の変数名にその先頭番地を割付けます.

(b) オペランド欄にDUPの指定をした場合，( )内で記述した式1の値をもつ領域を式2で記述した個数だけ確保して指定の変数名にその先頭番地を割付けます.

(c) 式1の記述形式は次のとおりです.

(i) 定数式(ただし，式の値は0FFH以下でなければなりません. また文字定数も含みます).

(ii) ?

(iii) 式2 DUP (式1〔,…〕)

(iv) {LOW / HIGH} アドレス式

?は不定値を表します.

，でいくつかの式を記述した場合および文字定数を記述した場合，メモリには下位アドレスから上位アドレスへと1バイトずつ格納されます.

(d) 式2の記述形式は次のとおりです.

(i) 定数式

定数式の値は1以上65535以下でなければなりません.

式の項にシンボルを記述する場合，シンボルは既に定義されていなければなりません.

(4) 使用例

```
①  SYM01   DB   00,?,'HELLO'
②  SYM02   DB   3H DUP(00,?,'HELLO')
③  SYM02   DB   00,?,'HELLO'
           DB   00,?,'HELLO'
           DB   00,?,'HELLO'
④  SYM03   DB   2H DUP (00,3 DUP(2,3))
⑤  SYM03   DB   00,2,3,2,3,2,3
           DB   00,2,3,2,3,2,3
```

上記の例の②と③，④と⑤はそれぞれ同じ意味になります.

4．6．2 DW

(1) 機　能

変数の定義およびワード（２バイト）領域を確保します．

(2) 記述形式　　オペランド欄

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
〔変数名〕	DW	{ 式1 〔,…〕 式2 DUP(式1〔,…〕) }	〔;コメント〕

(3) 説　明

(a) オペランド欄の記述形式および意味はＤＢの場合と同様です．

ただし，式１の記述形式は次のとおりです．

(i) 定数式（文字定数を含む）

(ii) アドレス式

(iii) ？

(iv) 式２　DUP （式１〔，…〕）

？は不定値を表わします．

式１は１６ビットの値に評価され（65535＝0FFFFH以下），メモリには下位アドレスに下位８ビットの値，上位アドレスに上位８ビットの値が格納されます．

(b) オペランド欄に文字列定数を記述する場合，文字列の長さは２以下でなければなりません．この場合のメモリへの格納方法も(a)に従います．

(c) アドレス式に，変数またはラベルを記述した場合，オフセット値が領域に確保されます．

(4) 使用例

```
SYM01   DW   00,?
SYM02   DW   100H   DUP (00H,?)
SYM03   DW   100H   DUP (?,10 DUP(?,10 DUP(?,10 DUP(?))))
```

4．6．3 DD

(1) 機　能

変数の定義およびダブルワード（４バイト）領域を確保します．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
〔変数名〕	DD	{ 式1 〔,…〕 式2 DUP(式1〔,…〕) }	〔;コメント〕

(3) 説　明

(a) オペランド欄の記述形式および意味はＤＷの場合と同様です．

(b) 式1として定数式を記述する場合，式の値は65535以下でなければなりません．

メモリには次のように格納されます．

1284H

84	12	00	00

下位アドレス　　上位アドレス

(c) 式1として文字列定数を記述する場合の文字数は2文字以下です．

メモリには次のように格納されます．

'AB'

42	41	00	00

下位アドレス　　上位アドレス

(d) 式1としてのアドレス式に変数またはレーベルを記述した場合，最初の1ワードにオフセット値が，次の1ワードにパラグラフ値が領域の値として確保されます．

すなわち，

```
DD SYM01は，
DW OFFSET SYM01
DW SEG SYM01      と等価になります．
```

(4) 使用例

```
SYM01   DD   'AB',?
SYM02   DD   100 DUP(?,SYM01+30H)
SYM03   DD   20 DUP(?,8 DUP(SYM02,?))
```

4・6・4 DBS

(1) 機　能

変数の定義および領域を確保します．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
〔変数名〕	DBS	式	〔；コメント〕

(3) 説　明

(a) オペランド欄に記述した式の値をサイズとするバイト領域を確保し，指定の変数名にその先頭番地を割付けます．

'DBS 式'は'DB 式 DUP(?)' と等価です．

(b) 式の値は1以上65535以下でなければなりません．

(c) 式の項にシンボルを記述する場合，シンボルは既に定義されていなければなりません．

(4) 使用例

```
BUFF DBS 100H
```

4．6．5 DWS

(1) 機　能

変数の定義およびワード領域を確保します．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
〔変数名〕	DWS	式	〔；コメント〕

(3) 説　明

(a) オペランド欄に記述した式の値をサイズとするワード領域を確保し，指定の変数名にその先頭番地を割付けます．

'DWS 式'は'DW 式 DUP(？)'と等価です．

(b) 式の値は1以上32767以下でなければなりません．

(c) 式の項にシンボルを記述する場合，シンボルは既に定義されていなければなりません．

(4) 使用例

```
BUFF DWS 100H
```

4．6．6 DDS

(1) 機　能

変数の定義およびダブル・ワード領域を確保します．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
〔変数名〕	DDS	式	〔；コメント〕

(3) 説　明

(a) オペランド欄に記述した式の値をサイズとするダブル・ワード領域を確保し，指定の変数名にその先頭番地を割付けます．

'DDS 式'は'DD 式 DUP(？)'と等価です．

(b) 式の値は1以上16383以下でなければなりません．

(c) 式の項にシンボルを記述する場合，シンボルは既に定義されていなければなりません．

(4) 使用例

```
BUFF DDS 100H
```

4．6．7　構造体名による領域確保

(1) 機　　能

変数の定義および領域を確保します．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
〔変数名〕	構造体名	<〔式1〔，…〕〕> 式2 DUP （<〔式1〔，…〕〕>）	〔；コメント〕

(3) 説　　明

(a) 構造体名はSTRUCおよびENDS疑似命令で既に定義してあるものでなければなりません．

(b) オペランド欄は式を＜＞でくくりますが，記述方法はDB，DWおよびDD疑似命令の場合と同様です．ただし，式1の内にDUPは使えません．（すなわちネストはできません）

(c) オペランド欄にDUPの指定をした場合の意味はDB，DW，DD疑似命令の場合と同じです（式2個分の領域を作ります）．

(d) 構造体定義したバイト数分の領域を初期値の指定どおりに確保し，その先頭番地を変数名に割付けます．

(e) ＜＞内に記述した式1の値は構造体を定義した時の初期値を変更するのに使います．ただし，変更できるのは構造体のフィールドを定義したときのオペランドが1個だけの場合です．

```
例      S1  STRUC
            DB    1     ;変更可
            DB    1，2  ;変更不可
            ENDS
```

(f) また，構造体の定義で指定した初期値を使う場合は“，”のみを記述してください．全フィールドに対して定義での初期値を使う場合は＜＞内に何も記述しないでください．

(g) 式1の数が構造体の定義で宣言したフィールドの数より少ない場合，残りのフィールドには構造体を定義したときの初期値が割当てられます．

(h) 式1の数が構造体の定義で宣言したフィールドの数より多い場合，余分な（＞側）指定は捨てられます．

(4) 使 用 例

```
STR     STRUC
        F1      DB      'HELLO'     ;初期値の変更可
        F2      DW      1，2        ;    〃     不可
                DW      3           ;    〃     可
        F4      DD      5           ;    〃     可
STR     ENDS
VAR01   STR             <'A'，  ，7，8>
                          └，，，でないことに注意してください．
                            （DW 1，2は一つのワード・フィールドです）．
```

は次の記述と等価です．

```
VAR01   DB   'AELLO'
        DW   1,2
        DW   7
        DD   8
```

(5) 注　意

(a) ＜＞内に文字列を記述する場合

○長さ1のまたは2の文字列は，バイト，ワード，ダブル・ワード・フィールドに割付けることができます．ただし，

1バイト・フィールドには長さ2の文字列は割付けられません．

例　定義で“F4　DB　0”

＜……，'AB'，……＞は許されません．

F4フィールド

に対応

○長さ3以上の文字列は，文字列で定義したバイト・フィールドにのみ割付けることができます．

この場合，＜＞内で指定の文字列の長さが構造体の定義での対応するバイト・フィールドの長さより小さい場合は次のようになります．

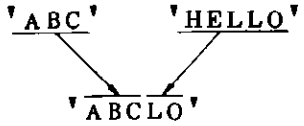
例　定義　F6　DB　'HELLO'

指定　＜……，'ABC'，……＞

F6フィールド

に対応

は，＜……，'ABCLO'，……＞と等価になります．



逆に，指定の文字列が長過ぎる場合は余分な文字列（文字列の後の方）は捨てられます．

例　定義　F6　DB　'HELLO'

指定　＜……，'ABCDEFG'，……＞

F6フィールドに対応

は，＜……，'ABCDE'，……＞と等価になります．

(b) ＜＞内に数値（アブソリュートまたはリロケータブルな数）を記述する場合の制限

○1バイトのバイト・フィールド，ワード・フィールドまたはダブル・ワード・フィールドには自由に割付けられます．ただし，バイト・フィールドにリロケータブルな数を記述する場合はバイト値になるようLOWまたはHIGH演算子をつけなければなりません．

例　定義　F7　DB　0

F8　DB　'A'

指定　＜……，1，2，……＞

F7，F8フィールドに対応

○ 2バイト以上のバイト・フィールドには割付けることができません．割付けることができるのは，文字列のみです．

例　定義　F9　DB　'AB'

　　指定　<……，4，……>；エラー

　　<……，'Z'，……>；OK

4．6．8　レコード名による領域確保

(1) 機　能

変数の定義および領域を確保します．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
〔変数名〕	レコード名	<〔式1〔，…〕〕> 式2 DUP(<〔式1〔，…〕〕>)	〔；コメント〕

式2の値はアブソリュートな数でなければなりません．

(3) 説　明

(a) レコード名はRECORD疑似命令で既に定義してあるものでなければなりません．

(b) オペランド欄は式を<>でくくりますが記述方法は，DB，DWおよびDD疑似命令の場合と同様です．ただし，式1の内にDUPは使えません．（すなわちネストはできません）

(c) レコードで定義したバイト数分の領域を初期値の指定どおりに確保し，その先頭番地を変数名に割付けます．

<>に記述した式1の値はレコードを定義した時の初期値を変更するのに使います．

(d) ただし，変更できるのはレコードのフィールドを定義した時のオペランドが1個だけの場合です．

(e) レコードの定義で指定した初期値を使う場合は“，”のみを記述してください．

全フィールドに対して，定義での初期値を使う場合は<>内に何も記述しないでください．

(f) レコードの定義でのビット・フィールドの数と<>内に記述した式の数が等しくないときの処理は，構造体名による領域確保の場合と同じです（4.6.7　参照）．

(g) 式1として'？'を記述した場合，対応するフィールドの値が0である領域が確保されます．

4.7 プログラム・リンケージ疑似命令

4.7.1 NAME

(1) 機　能

出力オブジェクト・モジュールのモジュール名を定義します．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
	NAME	モジュール名	〔；コメント〕

(3) 説　明

(a) モジュール名の記述上の規則は，シンボル名と同じです．

(b) 1回だけ記述できます．

(c) 本疑似命令を記述しない場合は，ソース・ファイル名のプライマリ・ネームがモジュール名としてとられます．

(4) 使用例

```
NAME    MAINPROG
```

4.7.2 PUBLIC

(1) 機　能

他のモジュール（別のアセンブル単位）で参照するシンボル（外部定義名）を宣言します．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
	PUBLIC	シンボル名〔，…〕	〔；コメント〕

(3) 説　明

(a) オペランド欄に記述したシンボルは他のモジュールで参照できるようになります．

(b) オペランド欄に記述したシンボルはそのモジュール内で定義するものでなければなりません．外部参照名（EXTRN疑似命令で宣言するシンボル）またはEQU疑似命令で外部参照名を割付けたシンボルは記述できません．

(c) オペランド欄に記述できるシンボルは次の3種類です．

(i) 定　数

(ii) 変　数

(iii) レーベル

(4) 使用例

```
        PUBLIC  SYM01,SYM02,SYM03
SYM01   EQU     03H
```

```
SYM02:   XOR      AW,AW
SYM03    DW       SYM01
```

4．7．3 EXTRN

(1) 機　能

他のモジュールで定義され，PUBLIC宣言されたシンボル（外部定義）を本モジュールで参照することを宣言します．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
	EXTRN	シンボル名：型	〔；コメント〕

(3) 説　明

(a) オペランド欄で記述したシンボル（他のモジュールで定義され，かつPUBLIC宣言されたもの）を本モジュールで参照できるようになります．

(b) オペランド欄で記述したシンボルは，他のモジュールで定義され，かつPUBLIC疑似命令で宣言されていなければなりません．

(c) 型の記述方法は次のいずれかでなければなりません．

(i) シンボルが定数の場合

ABS

(ii) シンボルがレーベルの場合

NEAR

FAR

(iii) シンボルが変数の場合

BYTE … 変数の属性がBYTEの場合

WORD … 〃　WORD 〃

DWORD… 〃　DWORD 〃

構造体名…変数が構造体名で定義してある場合

レコード名…変数がレコード名で定義してある場合

構造体名およびレーコド名は既に定義されていなければなりません．

(d) EXTRN疑似命令の位置

(i) シンボルを定義している他のモジュールのセグメント名と属性がわかっている場合は本モジュールの中で同名，同属性のセグメント定義をして，その中（SEGMENT ～ ENDS）で宣言してください．

例
```
CODE01   SEGMENT WORD PUBLIC 'CODE'
         EXTRN   SYM01:FAR,SYM02:FAR
         EXTRN   SYM03:BYTE,SYM04:DWORD
CODE01   ENDS
```

(ii) 不明の場合はセグメントの定義は行なわず，また本モジュールの中のどのセグメントの中でもない場所で宣言してください．

(iii) できるだけ(i)の方法を用いてください．

(ii)の方法ですと，外部参照名を参照する都度に所属セグメントをＳＥＧ演算子を使ってＡＳＳＵＭＥ疑似命令で指示する必要があります．

(4) 使 用 例

```
EXTRN  SYM01:FAR,SYM02:WORD
```

4.8 アセンブル終了疑似命令

4.8.1 END

(1) 機　　能

アセンブルを終了します．

(2) 記述形式

シンボル欄	ニーモニック欄	オペランド欄	コメント欄
	END	〔レーベル〕	〔；コメント〕

(3) 説　　明

オペランド欄にレーベルを記述すると，本モジュールは，メイン・モジュールになり，また，レーベルのアドレスをプログラムの実行開始アドレスとして，オブジェクト・モジュールに記録します．（ただし，実行開始アドレスはリンカで変更することもできます）

(4) 使 用 例

```
END
```

第5章 マクロ

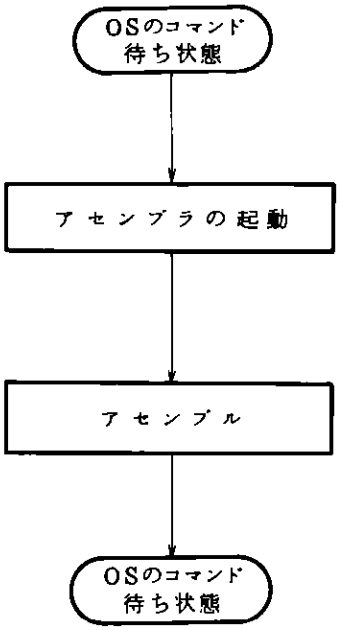
現在この機能は，サポートされていません．

第6章 操 作 法

6．1 操作手順

アセンブラの操作手順を図6－1に示します．

図 6 － 1 操 作 手 順



・次の指定によりアセンブラを起動します．

```
-RUN[:d:]RA116 △ソース・モジュール・ファイル名△基本コントロール[△・・・]↵
```

：d：RA116.86およびRA116.OMnを格納しているドライブ名

（省略時はＦ０となります．）

RA116：アセンブラのプログラム・ファイル名

ソース・モジュール・ファイル名：アセンブルするソース・モジュール・ファイルのファイル名

・媒体はＦＤまたはＨＤに限ります．

・ファイル名の形式は，6.3.1(1)ファイル名規則を参照してください．

基本コントロール：6.3.1基本コントロールを参照してください．

・アセンブラが起動されるとコンソールに実行開始メッセージが表示され，アセンブルが開始されます．

```
UPD70108/70116 ASSEMBLER  V×.×[××  ×××  ××]
Copyright(C)  1984  NEC  Corporation
```

実行開始メッセージ

6.2 プログラムの終了

アセンブラは，アセンブル処理が終了すると次のメッセージをコンソールに表示し制御をＯＳに戻します．

・正常終了の場合

```
ASSEMBLY COMPLETE,  ××××  ERROR(S) FOUND
```

・異常終了の場合

```
PROGRAM ABORTED
```

6.3 コントロール

アセンブラに対して詳細な動作指示を与えるものとして，次のコントロールを用意しております．

・基本コントロール

・汎用コントロール

6．3．1　基本コントロール

基本コントロールとは，プログラムの実行に先立ってプログラムの動作に必要な情報を指示するためのものです．アセンブラの起動時にコンソールから入力してください．

・同じコントロールまたは同種*のコントロールを複数回入力した場合は，後で入力したコントロールが有効となります．

*）OBJECTとNOOBJECTのようにNOの付いたコントロールと元のコントロールをいいます．

(1)　ファイル名規則

(a)　フロッピィ・ディスク上のファイルの場合

〔：ドライブ名：〕名称〔．拡張子〕

ドライブ名　　：ファイルを格納しているドライブ名
（F0｜F1…）

名　称　　：6文字以内の文字列

拡張子　　：3文字以内の文字列

＜省略時解釈＞

省略時の解釈はファイルの種類毎に異なり，表6－1のとおりとなります．

表6－1　ドライブ名および拡張子の省略時の解釈

項番	ファイルの種類	ドライブ名	拡張子
1	ソース・モジュール・ファイル	F0	ASM
2	オブジェクト・モジュール・ファイル		REL
3	リスト・ファイル		PRN

例

① ：F1：MDMAIN．ASM……ドライブ名がF1，名称がMDMAIN，拡張子がASM

② SUB1　　……ドライブ名と拡張子を省略

(b)　フロッピィ・ディスク以外の場合

：装置名：

装置名には，次のものが指定できます．

CO … コンソール
LP … ラインプリンタ

例

① PRINT(：LP：)…… リストをＬＰに出力

(2) 基本コントロールの種類

アセンブラでは，次の５種類の基本コントロールを提供しています．

- マクロ処理指定用
- 出力ファイル指定用
- 出力リスト項目指定用
- 出力リスト行指定用
- 処理日付指定用

基本コントロールの名前，指定形式および省略時の解釈についての一覧を表６－２に示します．

表６－２ 基本コントロール一覧

項番	分類		コントロール名	短縮形	省略時の既定値
1	マクロ処理指定用		MACRO NOMACRO	MC，MAC NOMC， NOMAC	NOMACRO
2	入出力ファイル指定用	オブジェクト・モジュール・ファイル	OBJECT〔(ファイル名)〕 NOOBJECT	OJ，OBJ NOOJ， NOOBJ	OJ(ソース・モジュール・ファイル名)ただし，拡張子は'REL'
		リスト・ファイル	PRINT〔(ファイル名)〕 NOPRINT	PR，PRI NOPR， NOPRI	PR(ソース・モジュール・ファイル名)ただし，拡張子は'PRN'
		ワーク・ファイル	WORKFILES (ドライブ名)	WF，WOR	WF(オブジェクト・モジュール・ファイルのドライブ名)
3	出力リスト項目指定用	シンボル・テーブル・リスト	SYMBOL NOSYMBOL	SB，SYM NOSB， NOSYM	NOSB
		クロス・レファレンス・リスト	XREF NOXREF	XR，XRE NOXR， NOXRE	NOXR
4	出力リスト行指定用	一行のカラム数	PAGEWIDTH(カラム数)	PW，WID	PW(１８０) ただし，PRINT(：CO：)の場合は８０
		１ページの行数	PAGELENGTH(行数)	PL，LEN	PL(６６)
5	処理日付指定		DATE(文字列)	DA，DAT	DA(ISIS-ⅡのDATEコマンドで最後に指定した日付)

(a) 出力ファイル指定用コントロール

出力ファイル指定のためのコントロールには，次のものがあります．

- オブジェクト・ファイル指定
- リスト・ファイル指定
- ワーク・ファイル指定

(i) オブジェクト・モジュール・ファイル指定（省略可）

＜形式＞

・オブジェクトを作成する場合

```
OBJECT[(ファイル名)]
```

・オブジェクトを作成しない場合

```
NOOBJECT
```

＜補足＞

・ＦＤファイル指定において，ドライブ名を省略した場合は，Ｆ０になります．

・オブジェクト・モジュール・ファイル指定を省略することができます．

省略時解釈：`OBJECT(ソース・ファイル名)`

拡張子は“ＲＥＬ”になります．

・出力媒体はＦＤまたはＨＤに限ります．

＜例＞ Ｆ１のドライブにＯＭＦ．ＲＥＬというファイル名でオブジェクトを作成

```
・・・ OBJECT(:F1:OMF) ・・・
```

(iii) リスト・ファイル指定（省略可）

＜形式＞

・リストを出力する場合

```
PRINT[(ファイル名)]
```

・リストを出力しない場合

```
NOPRINT
```

＜補足＞

・ＮＯＰＲＩＮＴの指定をしてリスト・ファイルを出力しない場合，エラーのある文およびエラー・メッセージがコンソールに出力されます．

・リスト・ファイルをＦＤまたはＨＤに作成する場合，ドライブ名を省略した場合は，Ｆ０となります．

・リスト・ファイル指定を省略することができます．

省略時解釈：`PRINT(ソース・ファイル名)`

拡張子は'ＰＲＮ'になります．

・出力媒体は，ＦＤ，ＨＤ，ＬＰ，コンソールのいずれかに限ります．

＜例＞

・・・ PRINT（：LP：） ・・・ ……ＬＰへのリスト出力

・・・ PRINT(：F1：PRF)・・・ ……ＦＤまたはＨＤファイルへのリスト出力ファイル名はＰＲＦ，ＰＲＮ

(iv) ワーク・ファイル指定（省略可）

＜形式＞

WORKFILES（ドライブ名）

＜補足＞

・省略した場合のドライブ名は，オブジェクト・モジュール・ファイルと同一のドライブ名になります．

省略時解釈 | WORKFILES（オブジェクト・モジュール・ファイルのドライブ名） |

・媒体はＦＤまたはＨＤに限ります．

＜例＞ Ｆ１のドライブにワーク・ファイルを割当てる．

・・・・ WORKFILES(：F1：)・・・・

(b) 出力リスト項目指定用コントロール

リスト・ファイルに出力すべきリストの種類を指定するためのコントロールです．

出力リスト項目には，次のものがあります．

- シンボル・テーブル・リスト用
- クロス・レファレンス・リスト用
- エラー・リスト用

なお，アセンブル・リストはＰＲＩＮＴコントロールを指定した場合は無条件に出力されます．

(i) シンボル・テーブル・リスト用（省略可）

＜形式＞

- シンボル・テーブル・リストを出力する場合

SYMBOL

- シンボル・テーブル・リストを出力しない場合

NOSYMBOL

＜補足＞

- NOPRINTコントロールを指定した場合は，SYMBOLSの指定が無視されます．
- このコントロールを省略することができます．

省略時解釈： | SYMBOL |
|---|

＜例＞

①シンボル・リストをＬＰに出力

・・・ PRINT(:LP:) SYMBOL ・・・

②シンボル・リストを出力しない

・・・ NOSYMBOL ・・・

(ii) クロス・レファレンス・リスト用（省略可）

＜形式＞

• クロス・レファレンス・リストを出力する場合

XREF

• クロス・レファレンス・リストを出力しない場合

NOXREF

＜補足＞

• NOPRINTコントロールを指定した場合は，XREFの指定が無視されます．

• このコントロールを省略することができます．

省略時解釈： | NOXREF |

＜例＞

①クロス・レファレンス・リストをＬＰに出力

・・・ PRINT(:LP:) XREF ・・・

②クロス・レファレンス・リストを出力しない

・・・ NOXREF ・・・

または

コントロールを指定しない

(c) 出力リスト行指定用コントロール

リスト・ファイルに出力されるリストの一行のカラム数（$C_R L_F$ を含む）と，1ページの行数を指定します．

指定項目には次の2種類のものがあります．

- 最大カラム数用
- ページ内行数用

なお，これらの指定はすべての種類のリストに共通です．

(i) 最大カラム数用（省略可）

＜形式＞

PAGEWIDTH(カラム数)

＜補足＞

- カラム数に記述できるのは80以上182以下の10進数です．
- このコントロールは省略することができます．

省略時解釈： PAGEWIDTH(180)

PRINTコントロールで，リスト・ファイルをコンソールに出力すると指定した場合は80になります．

＜例＞ 各種リストの1行の印字幅を132にする．

・・・ PAGEWIDTH(132) ・・・

(ii) ページ内行数用（省略可）

＜形式＞

PAGELENGTH（行数）

＜補足＞

- 行数に記述できるのは20以上255以下の10進数です．
- このコントロールは省略することができます．

省略時解釈： PAGELENGTH(66)

＜例＞ 各種リストの1ページの行数を80にする．

・・・ PAGELENGTH(80) ・・・

(d) 処理日付指定用（省略可）

アセンブルを行なった日の日付を通知するためのコントロールです．出力リストのヘッダ（ＤＡＴＥ欄）に指定された日付が印字されます．更に，オブジェクト・プログラム・ファイルの指定がある場合は，ファイルの中に処理日付として残ります．

＜形式＞

ＤＡＴＥ（日付）

日付：１２文字以内の文字列
（ただし，（　）を除く）

＜補足＞

- ＤＡＴＥコントロールを省略した場合は，ＩＳＩＳ－ⅡのＤＡＴＥコマンドで最後に指定した日付となります．

＜例＞　日付を８２／０８／０８とする．

・・・ DATE（８２／０８／０８） ・・・

６．３．２　汎用コントロール

汎用コントロールは，アセンブル実行中にアセンブラに対して動作指示を行なうためのものです．したがって，ソース・プログラムの中にソース・ステートメントとして指定します．

(1) 入力形式

$［　］コントロール名［　オペランド］↵

↵：復改

第１カラム

- 第１カラムに“＄”を指定してください．
- 継続行の指定はできません．
- １行に複数のコントロールは記述できません．

例

$INCLUDE（：F1：SMAC）↵

(2) 汎用コントロールの種類

アセンブラでは，次の２種類の汎用コントロールを提供しています．

- ソース・ファイル入力用
- リスト出力制御用

汎用コントロールの一覧を表６－３に示します．

表６－８　汎用コントロール一覧

項番	分　　類	コントロール名	短　縮　形	入　力　形　式
1	ソース・ファイル入力用	INCLUDE	IC，INC	$ INCLUDE （ファイル名）
2	リスト出力制御用	TITLE	TT，TIT	$ TITLE （文字列）
		EJECT	EJ，EJE	$ EJECT
		LIST	LI，LIS	$ LIST
		NOLIST	NOLI，NOLIS	$ NOLIST
		GEN	GE	$ GEN
		NOGEN	NOGE	$ NOGEN
		GENONLY	GO	$ GENONLY

(a)　ソース・ファイル入力用

ソース・ステートメント群を実行時に他のソース・ファイルから入力するためのコントロールです．

＜形式＞

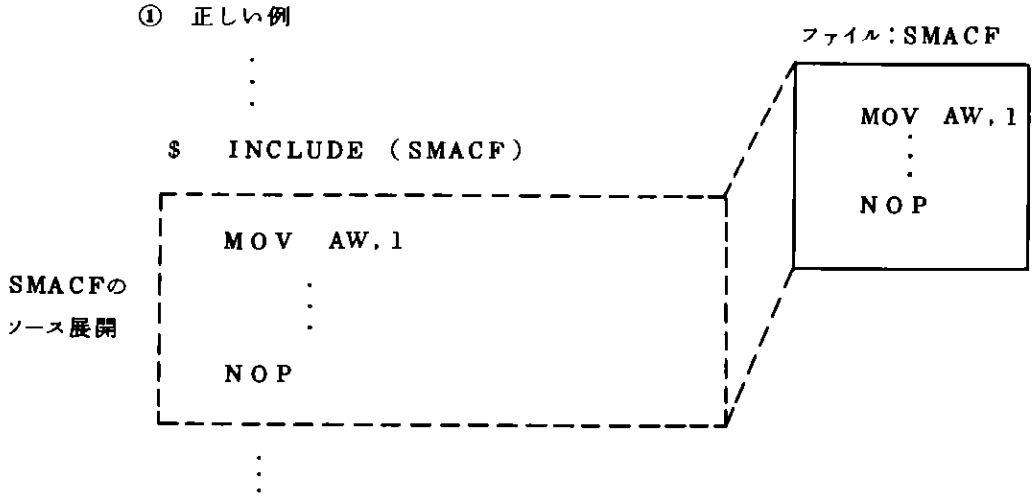
```
INCLUDE（ソース・ファイル名）
```

＜補足＞

- ファイル指定において，ドライブ名を省略した場合は，Ｆ０となります．
- ソース・ファイルの構成については，7.1.2の入出力ファイルを参照してください．

＜例＞

①　正しい例



(b) リスト出力制御用

アセンブル・リストの出力形式を制御するためのコントロールとして次のものがあります.

- タイトル印字
- 改頁
- リスト出力開始，中止
- マクロ展開リストの制御

(i) タイトル印字

改頁し，指定したタイトルがヘッダに印字されます.

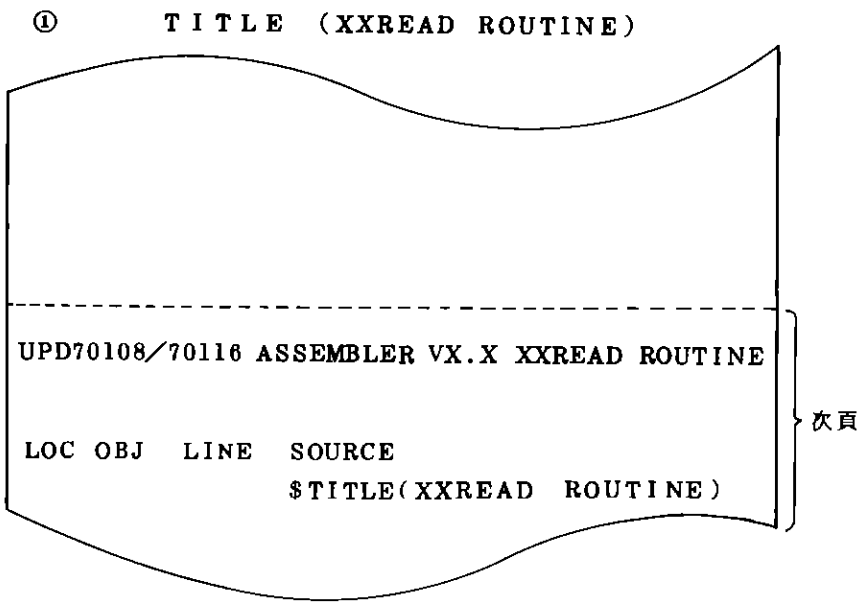
<形式>

```
TITLE( 68字以内の文字 )
```

<補足>

- TITLEコントロール自身のイメージは，改頁後の頁の先頭に印字されます.
- タイトルには，63文字以内の文字列が指定できます. 68文字をオーバした場合は，オーバ分が切りすてられます.
  ただし，PAGEWIDTHコントロールの指定値によりこの値は変動します.
  (68はPAGEWIDTH(132)の場合です.)

＜例＞



(ii) 改頁

アセンブル・リストが改頁されます.

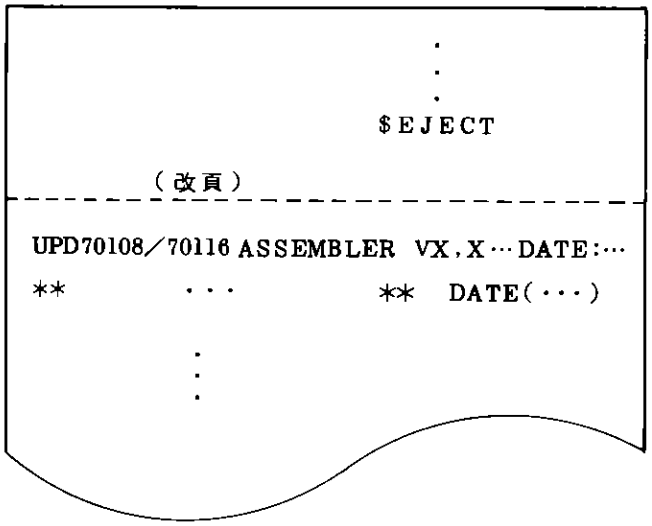
＜形式＞

EJECT

＜補足＞

・EJECTコントロール自身のイメージは改頁前の頁に印字されます.

＜例＞



(iii) リスト出力開始，中止

アセンブル・リストのリスト・ファイルへの出力を開始または中止します．（リストの出力量を制御するためだけのものであり，中止したあとでもアセンブルは続けられます．）

＜形式＞

・アセンブル・リストの出力を開始する場合

```
$   LIST
```

このコントロールの次の行から出力を開始します．

・アセンブル・リストの出力を中止する場合

```
$  NOLIST
```

このコントロールの次の行から出力を中止します．

＜補足＞

・アセンブルの始動時は'$ LIST'が指定された状態になっています．

＜例＞

```
            ⌇
        PUBLIC   SUM
$  NOLIST
GROUP1 GROUP   CODE01,CODE02
;
$ LIST
STR    STRUC
       PRICE  DW     ?
            ⌇
```

```
            ⌇


          PUBLIC   SUM
$ NOLIST                          ←$LISTコントロールの
                                    次の行から出力
STR   STRUC
      PRICE      DW    ?

            ⌇
```

(iii) マクロ展開リストの制御

マクロ定義，マクロ参照およびマクロ展開イメージのアセンブル・リストへの出力を制御します.

＜形式＞

・マクロ定義，マクロ参照およびマクロ展開イメージをアセンブル・リストへの出力を制御します.

```
$ GEN
```

・マクロ定義およびマクロ参照イメージをアセンブル・リストに出力する場合
（マクロ展開イメージは出力しません.）

```
$ NOGEN
```

・マクロ展開イメージをアセンブル・リストに出力する場合
（マクロ定義およびマクロ参照イメージは出力しません.）

```
$ GENONLY
```

＜補足＞

・アセンブラの起動時は'GEN'が指定された状態になっています.

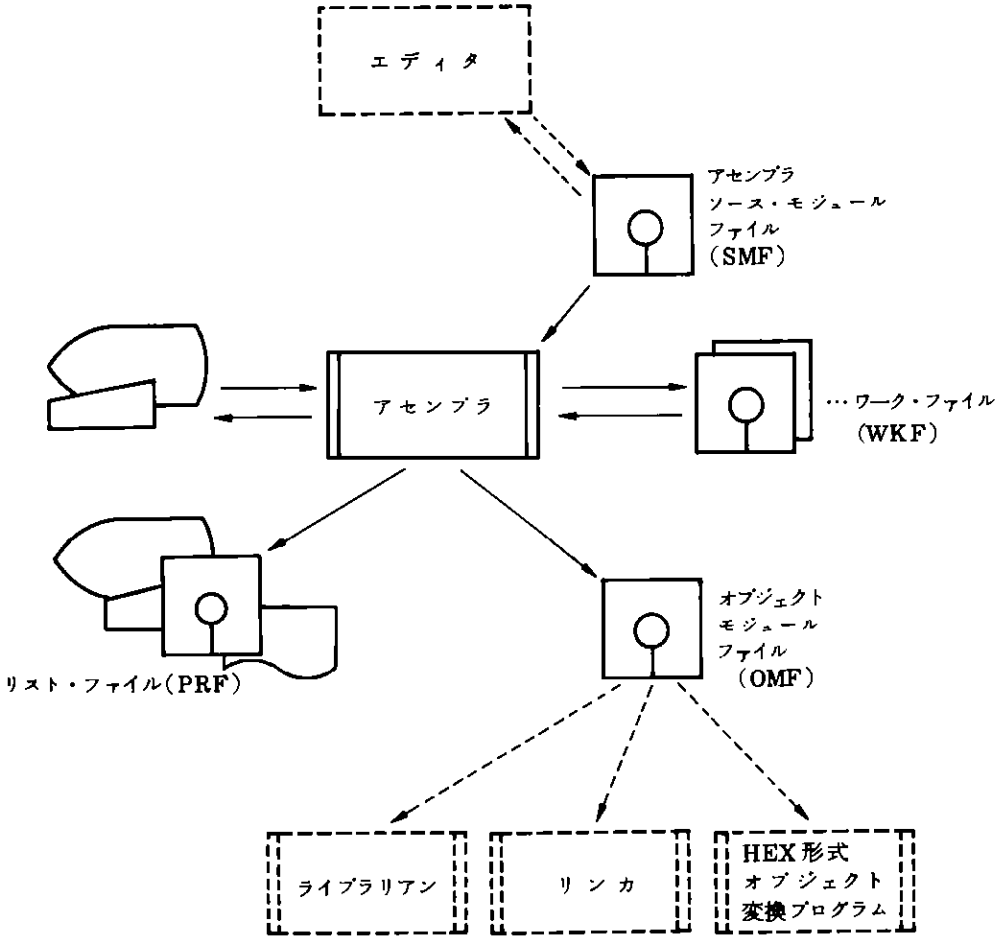
第7章　入 出 力

本章では本アセンブラが入出力するファイルの構成と，各種出力リストについて述べます．

7．1　入出力ファイル

アセンブラが扱う入出力ファイルの関連を図7－1に示します．

図7－1　入出力ファイルの関連



本リロケータブル・アセンブラ・パッケージ内のユーティリティ

7．1．1 種類と媒体

(1) 種　類

アセンブラが扱うファイルには，次の5種類があります．

- アセンブラ・ソース・モジュール・ファイル　（SMF）
- オブジェクト・モジュール・ファイル　（OMF）
- リスト・ファイル　（PRF）
- インクルード・ファイル　（INF）
- ワーク・ファイル　（WKF）

(2) 入出力種別と媒体

各ファイルに対する入出力種別および使用可能な媒体を表7－1に示します．

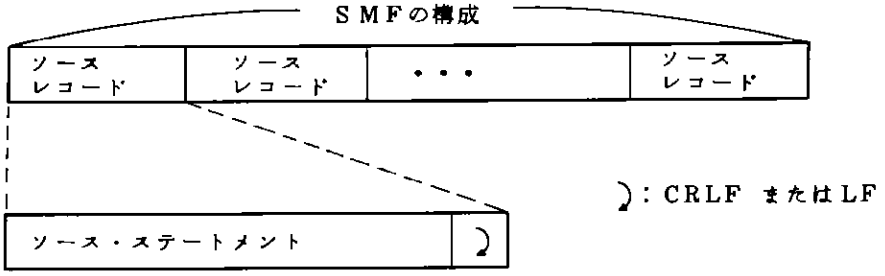
表7－1　入出力種別と媒体

項番	ファイル	入出力	媒　体
1	SMF	入力	FD／HD
2	OMF	出力	FD／HD
3	PRF	出力	FD/HD/コンソール/LP
4	INF	入力	FD／HD
5	WKF	入出力	FD／HD

7．1．2 ファイルの説明

(1) アセンブラ・ソース・モジュール・ファイル（SMF）

- アセンブラ言語で記述されたソース・ステートメント群を格納しているファイルです．



- ソース・ステートメントの区切りは“) ”です．
- ソース・レコードの長さは可変長ですが，182字以下でなければなりません．
- ソース・ステートメントに関しては第3章を参照ください．

(2) オブジェクト・モジュール・ファイル（OMF）

- アセンブル結果のオブジェクト・モジュールを格納するファイルです．（OBJECTコントロールで指定されたファイルです．）
- 絶対アセンブルを行ない，かつすべてのアドレス解決が完了している（外部参照がない）場合には，このファイルをHEX形式オブジェクト変換プログラムへ入力することによりHEX形式オブジェクト・プログラムを生成することができます．

- リロケータブル・アセンブル等により，アドレスが未解決の状態である場合は，このファイルをリンカに入力しロード・モジュール（すべてのアドレスが解決されている）を作成する必要があります．

(3) リスト・ファイル（ＰＲＦ）

アセンブル結果のリスト類およびメッセージを格納するファイルです．（ＰＲＩＮＴコントロールで指定されるファイルです．）

(4) インクルード・ファイル（ＩＮＦ）

ソース・ステートメント群をＳＭＦ以外のファイルからマクロ的に入力したい場合に使用するファイルです．ファイルの構成はＳＭＦと同じです．（ＩＮＣＬＵＤＥコントロールで指定されるファイルです．）

(5) ワーク・ファイル（ＷＫＦ）

アセンブラが処理の途中結果等を一時的に格納するための作業用ファイルです．

７．２　出力リスト

本アセンブラが出力するリストは次の３種類です．

(i) アセンブル・リスト

(ii) シンボル・リスト

(iii) クロス・レファレンス・リスト

上記の(i)～(iii)は同一のファイル（ＰＲＩＮＴコントロールによる）に出力されます．

7．2．1 アセンブル・リスト

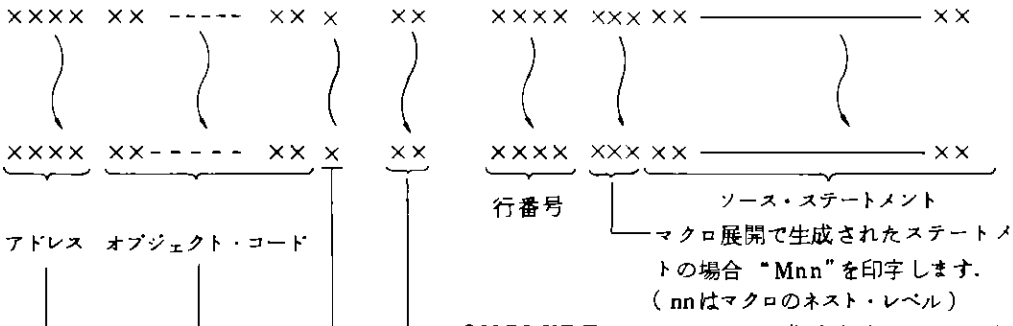
アセンブル結果を，エラー・メッセージと共に出力します．

(1) 形　式

UPD70108／70116 ASSEMBLER V×.×　タイトル　DATE：日付　PAGE：ページ

SOURCE FILE ：ソース・モジュール・ファイル名
OBJECT FILE ：オブジェクト・モジュール・ファイル名（モジュール名）
COMMAND　　：起動時のコマンド

LOC　OBJ　　LINE　SOURCE



マクロ展開で生成されたステートメントの場合 "Mnn"を印字します．
（nnはマクロのネスト・レベル）

INCLUDEコントロールで指定したファイルから入力したソース・ステートメントの場合，'In'を印字します．（nは，インクルード・レベル）

外部参照シンボル（EXTRN疑似命令で宣言したシンボル）を参照している場合，'E'を印字します．
リロケータブルなシンボルを参照している場合，'R'を印字します．

1行に7バイト以内のオブジェクト・コードを印字します．
パラグラフ値の参照である場合は'----'を印字します．
不定値(?)で初期化した場合は，'??'（バイト），'????'（ワード），'????????'（ダブル・ワード）を印字します．

オブジェクト・コードのセグメント内相対アドレスを印字します．ソース・ステートメントがSEGMENT，STRUCまたはENDS疑似命令である場合は'----'を印字します．
レコード定義の場合は'#'を印字します．

(2) ＥＱＵ疑似命令の場合のＬＯＣ欄，ＯＢＪ欄

ＥＱＵ疑似命令の場合，ＬＯＣ欄とＯＢＪ欄を使ってシンボルに割当てた値またはシンボルの属性を表します．

印字形式を表７－２に示します．

表７－２

EQUの内容	表　　示
レジスタ	'REG'
変数，ラベル	××××:〔　〕 ↑　↑ セグメント表示　インデックス表示
構造体フィールド，レコード・フィールド，外部参照，セグメント，グループ	'S FIELD','R FIELD', 'EXTRN','SEGMENT', 'GROUP'
定　数	'××××'または'−××××'（負のとき）

７．２．２　シンボル・リスト，クロス・レファレンス・リスト

ソース・プログラム内で定義したシンボル名，属性およびそのシンボルを定義・参照したステートメントの行番号を出力します．

(1) 形　式

```
UPD70108/70116 ASSEMBLER V××.××　タイトル　DATE：日付　PAGE：ページ
XREF   SYMBOL LIST
----   ------ ----

NAME        TYPE      VALUE ATTRIBUTES   XREFS

××・・・×××  ×××××××  ××××  ×————————×  ××  ××  ××---××

××・・・×××  ×××××××  ××××  ×————————×  ××  ××  ××---××
```

シンボル名　シンボルのタイプ　シンボルの値　シンボルの属性　定義・参照・削除したステートメントの行番号

14文字ずつ区切って印字

定義の場合は'#'，削除の場合は'P'，宣言の場合は'＊'を行番号の次に印字する．

詳しい内容は(2)〜(4)を参照してください．

(2) タイプ一覧

シンボル・リストのTYPE 欄に印字されるタイプ一覧を表7－8に示します.

表7－3

表　　示	タ イ プ の 内 容
BYTE	バイト変数
WORD	ワード変数
DWORD	ダブルワード変数
NEAR	NEARレーベル
FAR	FARレーベル
NUMBER	定　　数
REG	レジスタ
------	未定義シンボル
SEGMENT	セグメント
STRUC	構 造 体
RECORD	レコード
GROUP	グループ
S FIELD	構造体フィールド
R FIELD	レコードのフィールド
STRUCV	構造体変数

(3) 値　一　覧

シンボル・リストのVALUE欄に印字される値の一覧を表7－4に示します.

表7－4

タ　イ　プ	VALUE欄
変数，ラベル	セグメント内オフセット値
セグメント，グループ，構造体，レコード，未定義シンボル	空　白
構造体フィールド	構造体内オフセット値
レコード・フィールド	シフト・カウント
外部シンボル	常に'000H'

(4) 属性一覧

シンボル・リストのATTRIBUTES欄に印字される属性一覧を表7－5に示します．

表7－5

タイプ	ATTRIBUTES 欄
グループ	グループに属するセグメント名
変数，レーベル	それが属するセグメント名
レコード	レコードのビット幅
構造体	構造体のバイト・サイズ，フィールド数
レコード・フィールド	レコード名，フィールドのビット幅
モジュール名	▼–MODULE NAME–▼
属性が認識できなかったシンボル	▼–UNRECOGNIZED▼

7．3 エラー・メッセージ

エラー・メッセージにはコンソール上に表示されるものとアセンブル・リスト上に印字されるメッセージがあります．

7．3．1 コンソールに表示されるエラー・メッセージ

コンソールに表示されるエラー・メッセージには次の4種類があります．

(a) 起動コマンドに対するエラー・メッセージ

(b) アセンブラの内部処理上のエラー・メッセージ

(c) 致命的なファイル I／O エラーに対するエラー・メッセージ

(d) アセンブル・エラーに対するエラー・メッセージ

注 (d)のエラー・メッセージはNOPRINTコントロールを指定してリストを出力しない場合にのみ表示されます．

エラー・メッセージの内容は7.3.2を参照してください．

(1) 起動コマンドに対するエラー・メッセージ

形式：＊＊＊ ERROR エラー番号 エラー・メッセージ

PROGRAM ABORTED

エラー番号 F001	メッセージ	MISSING FILE SPECIFICATION
	原因	ソース・モジュール・ファイル名の指定がありません.
	プログラムの処理	OSにコントロールを戻します.
	ユーザの処置	ソース・モジュール・ファイル名を指定して再実行してください.
エラー番号 F002	メッセージ	ILLEGAL FILE SPECIFICATION －ファイル名
	原因	ファイル名が正しくありません.
	プログラムの処理	OSにコントロールを戻します.
	ユーザの処置	正しいファイル名を指定して再実行してください.
エラー番号 F003	メッセージ	FILE SPECIFICATION CONFLICTED －ファイル名
	原因	互いに異なるファイル名を指定しなければならないコントロールに同じ名前のファイル名を指定しています.
	プログラムの処理	OSにコントロールを戻します.
	ユーザの処置	正しくファイル名を指定して再実行してください.
エラー番号 F005	メッセージ	FILE NOT FOUND －ファイル名
	原因	ソース・モジュール・ファイルが存在しません.
	プログラムの処理	OSコントロールを戻します.
	ユーザの処置	正しいソース・モジュール・ファイル名を指定して再実行してください.
エラー番号 F006	メッセージ	ILLEGAL FILE －ファイル名
	原因	出力ファイルまたはワーク・ファイルが，書込み保護されています．または入出力ファイル名として指定できないディバイス形のファイルが指定されています.
	プログラムの処理	OSにコントロールを戻します.
	ユーザの処置	原因により，出力ファイルまたはワーク・ファイルの書込み保護を外すかまたは別のファイル名を指定して再実行してください.
エラー番号 F007	メッセージ	ILLEGAL OR MISSING PARAMETER：パラメータ
	原因	パラメータが必要なコントロールにパラメータが指定されていないか不当なパラメータが指定されています.
	プログラム の処理	OSにコントロールる戻します.
	ユーザの処置	正しいパラメータを指定して再実行してください.
エラー番号 F008	メッセージ	CONTROL IS NOT RECOGNIZED：文字列
	原因	コントロールとして不当な文字列が指定されています.
	プログラムの処理	OSにコントロールを戻します.
	ユーザの処置	正しいコントロールを指定して再実行してください.

エラー番号 F009	メッセージ	INVALID SYNTAX：X
	原因	パラメータの無いコントロールにパラメータを指定しています.
	プログラムの処理	OSにコントロールを戻します.
	ユーザの処置	正しくコントロールを指定して再実行してください.

(2) アセンブラの内部処理上のエラー・メッセージ

形式：＊＊＊ ERROR エラー番号 エラー・メッセージ

エラー番号 A901	メッセージ	WORKING TABLE SPACE EXHAUSTED
	原因	シンボル登録用のワーク・エリア（メモリ）が足りません.
	プログラムの処理	OSにコントロールを戻します.
	ユーザの処置	シンボル数を減らして再実行してください.
エラー番号 A999	メッセージ	UTILITY PROGRAM ERROR：エラー・コード
	原因	プログラム（アセンブラ）のバグを検出しました.
	プログラムの処理	OSにコントロールを戻します.
	ユーザの処置	なし. プログラム名，バージョンおよびエラー・コードを弊社に連絡してください.

(3) 致命的なファイルI／Oエラーに対するエラー・メッセージ

形式：

ERROR nnn USER PC mmmm

FATAL ERROR nnn － RUN TERMINATED

くわしくはISIS－Ⅱシステム・ユーザーズ・ガイド等を参照ください.

7．3．2 アセンブル・リストに印字されるエラー・メッセージ

エラーを検出した次の行に印字されます．

形式：＊＊＊ ERROR エラー番号，エラー・メッセージ

エラー番号 1	メッセージ	SYNTAX ERROR
	原因	構文に誤りがあります．
	プログラムの処理	文を無視します．
	ユーザの処置	正しい文を記述して再実行してください．
エラー番号 2	メッセージ	OPERAND TYPE IS UNSUITABLE
	原因	機械語のオペランドが記述できない形式です．
	プログラムの処理	最大命令語長のＮＯＰ命令を生成します．
	ユーザの処置	正しい形式のオペランドを記述して再実行してください．
エラー番号 3	メッセージ	OBJECT SIZE ESTIMATION FAILED－TOO BIG
	原因	機械語命令のオブジェクト・サイズが，アセンブラの評価した値を越えています．（シンボルの前方参照の場合）
	プログラムの処理	アセンブラの評価した値のバイト数のＮＯＰ命令を生成します．
	ユーザの処置	ＰＴＲ演算子で型を明示するか，セグメント・オーバライドを記述して再実行してください．
エラー番号 4	メッセージ	TYPE INFORMATION IS AMBIGUOUS
	原因	オペランドの型情報が不足でオブジェクト・コードが決定できません．
	プログラムの処理	文を無視します．
	ユーザの処置	ＰＴＲ演算子を使って型を明示し，再実行してください．
エラー番号 5	メッセージ	CANNOT REFER TO THIS OPERAND-MISSING ASSUME
	原因	現在のＡＳＳＵＭＥ状態では，オペランドに記述したシンボルを参照できません．
	プログラムの処理	シンボルの属するセグメントが，ハードウェア・ディフォルトのセグメント・レジスタにＡＳＳＵＭＥされているとみなします．
	ユーザの処置	正しくＡＳＳＵＭＥして再実行してください．
エラー番号 6	メッセージ	ILLEGAL NEAR JUMP(BRANCH)-MISSING OR DIFFERENT PS ASSUME
	原因	ＰＳレジスタにセグメントまたはグループを正しくＡＳＳＵＭＥしていないため，NEAR JUMPできません．
	プログラムの処理	PSレジスタに現在のセグメント名をＡＳＳＵＭＥしてあるものとみなします．
	ユーザの処置	正しくＡＳＳＵＭＥして再実行してください．

エラー番号 7	メッセージ	NEAR LABEL CANNOT BE DEFIND-MISSING PS ASSUME
	原因	PSレジスタにセグメントまたはグループをASSUMEしていないためNEARレーベルを定義できません.
	プログラムの処理	レーベルを定義しているセグメントをPSレジスタにASSUMEしてあるとみなします.
	ユーザの処置	正しくASSUMEして再実行してください.
エラー番号 8	メッセージ	SEGMENT OVERRIDE IS ILLEGAL TO THIS INSTRUCTION
	原因	ハードウェア・ディフォールトでセグメント・レジスタが固定されている命令（ストリング命令等）にハードウェア・ディフォールト以外のセグメント・オーバライドをしています.
	プログラムの処理	セグメント・オーバライドの記述を無視します.
	ユーザの処置	セグメント・オーバライドを削除してください.
エラー番号 9	メッセージ	LABEL WITH COLON IS NOT ALLOWED
	原因	:の記述が許されないシンボルの定義に:を記述しています.（疑似命令で定義するシンボル）
	プログラムの処理	文を無視します.
	ユーザの処置	:を削除して再実行してください.
エラー番号 10	メッセージ	LABEL WITHOUT COLON IS REQUIRED
	原因	:なしのシンボルが必要な疑似命令にシンボルを記述していません.
	プログラムの処理	文を無視します.
	ユーザの処置	正しくシンボルを記述して再実行してください.
エラー番号 11	メッセージ	LABEL IS NOT ALLOWED TO THIS STATEMENT
	原因	シンボル欄にシンボルを記述できない疑似命令にシンボルを記述しています.
	プログラムの処理	シンボル欄に記述されたシンボルを無視します.
	ユーザの処置	シンボル欄のシンボルを削除して再実行してください.
エラー番号 12	メッセージ	ALREADY DEFINED SYMBOL-IGNORED
	原因	シンボルを多重に定義しています.
	プログラムの処理	2度目以降のシンボルの定義を無視します.
	ユーザの処置	正しくシンボルを記述して再実行してください.
エラー番号 13	メッセージ	ARITHMETIC OVERFLOW
	原因	式の値が17ビットで表現できる範囲を越えています.
	プログラムの処理	キャリーまたはボローを無視します.
	ユーザの処置	式を正しく記述して再実行してください.

エラー番号 14	メッセージ	NUMERIC ERROR
	原因	定数の記述形式に誤りがあります.
	プログラムの処理	定数の値を0と評価します.
	ユーザの処置	正しく定数を記述して再実行してください.
エラー番号 15	メッセージ	FORWARD REFERENCE OR UNDEFINED SYMBOL IS NOT ALLOWED
	原因	前方参照の許されない文で前方参照または未定義シンボルの参照をしています.
	プログラムの処理	シンボルの値を0と評価します.
	ユーザの処置	正しいシンボルを記述して再実行してください.
エラー番号 16	メッセージ	ILLEGAL EXPRESSION
	原因	式の記述形式に誤りがあります.
	プログラムの処理	式の評価値は不定となります.
	ユーザの処置	正しく式を記述して再実行してください.
エラー番号 17	メッセージ	ILLEGAL CHARACTER
	原因	使用できない文字を使っています.
	プログラムの処理	空白が記述されたとみなします. アセンブル・リスト上は^で表現します.
	ユーザの処置	正しい文字を記述して再実行してください.
エラー番号 18	メッセージ	TOO MANY ERRORS-QUIT REPORTING
	原因	一つの文にあやまりが，8個を越えています.
	プログラムの処理	8個を越えるエラー・メッセージは表示しません.
	ユーザの処置	誤りを修正して再実行してください.
エラー番号 19	メッセージ	INVALID MNEMONIC CODE
	原因	ニーモニック欄に機械語命令／疑似命令以外の文字列を記述しています.
	プログラムの処理	文を無視します.
	ユーザの処置	正しい命令を記述して再実行してください.
エラー番号 20	メッセージ	UNDEFINED SYMBOL-ZERO USED
	原因	未定義シンボルを参照しています.
	プログラムの処理	シンボルの値を0とみなします.
	ユーザの処置	正しいシンボルを記述して再実行してください.
エラー番号 21	メッセージ	ILLEGAL USE OF STRUCTURE OR RECORD NAME
	原因	構造体名またはレコード名を不正に使っています.
	プログラムの処理	文を無視します.
	ユーザの処置	構造体名またはレコード名を，他の正しいシンボル名に変更して再実行してください. （シンボル名を誤った場合）

エラー番号 22	メッセージ	THIS TYPE OF EXPRESSION IS NOT ALLOWED IN STORAGE INITIALIZATION
	原因	領域の初期値に，記述できない式を記述しています.
	プログラムの処理	領域の初期値を0にします.
	ユーザの処置	正しい式を記述して再実行してください.
エラー番号 23	メッセージ	LABEL CANNOT BE OVERRIDDEN BY OTHER THAN PS
	原因	レーベルにセグメント・レジスタDS0，DS1またはSSでオーバライドしています.
	プログラムの処理	セグメント・オーバライドの記述を無視します.
	ユーザの処置	セグメント・オーバライドを削除して再実行してください.
エラー番号 24	メッセージ	SOURCE LINE IS TOO LONG-ALLOWED ONLY UP TO 132
	原因	ソース・モジュールの1行の長さが132文字を超えています.
	プログラムの処理	133文字目以降を無視します.
	ユーザの処置	1行が132文字以内となるように修正して再実行してください.
エラー番号 25	メッセージ	OPERAND IS ILLEGAL
	原因	疑似命令のオペランドに，同じ意味の指定を複数回行なっています.
	プログラムの処理	最初の指定のみを有効とし，残りの指定を無視します.
	ユーザの処置	余分な指定を削除して再実行してください.
エラー番号 26	メッセージ	LABEL ON ENDS OR ENDP IS MISMATCHED
	原因	SEGMENT，STRUCまたはPROC疑似命令とENDSまたはENDP疑似命令の対応がとれていません.
	プログラムの処理	直前のSEGMENT，STRUCまたはPROC疑似命令と対応づけます.
	ユーザの処置	疑似命令の対応をとって再実行してください.
エラー番号 27	メッセージ	ONLY ONE NAME STATEMENT IS ALLOWED
	原因	NAME疑似命令を複数回記述しています.
	プログラムの処理	2回目以降のNAME疑似命令を無視します.
	ユーザの処置	NAMEを疑似命令を1回のみにして再実行してください.
エラー番号 28	メッセージ	TEXT EXISTS BEHIND END STATEMENT-IGNORED
	原因	END疑似命令文の後に文を記述しています.
	プログラムの処理	END疑似命令文の後にある文をコメントとみなします.
	ユーザの処置	END疑似命令が正しい位置にあるか否かをチェックし，誤っている場合はEND疑似命令の位置を修正して再実行してください.
エラー番号 29	メッセージ	NO END STATEMENT IN SOURCE TEXT-ASSUMED
	原因	ソース・モジュールにEND疑似命令文がありません.
	プログラムの処理	ソース・モジュールの最後にEND疑似命令文があったものとみなします.
	ユーザの処置	END疑似命令文を記述して再実行してください.

エラー番号 30	メッセージ	TOO MANY SEGMENT IN A GROUP-ALLOWED ONLY UP TO 16
	原因	グループ疑似命令のオペランドに17個以上のセグメント名を記述しています.
	プログラムの処理	先頭16個のセグメントを有効とし，17個目以降のセグメントを無視します.
	ユーザの処置	1グループを構成するセグメントの数を16個以内にして再実行してください.
エラー番号 31	メッセージ	PUBLIC SYMBOL IS OF ILLEGAL TYPE
	原因	PUBLIC疑似命令のオペランドに不当な属性のシンボルを記述しています.
	プログラムの処理	不当な属性のシンボルの記述を無視します.
	ユーザの処置	不当な属性をもつシンボルをPUBLIC疑似命令のオペランドから削除して再実行してください.
エラー番号 32	メッセージ	PUBLIC SYMBOL IS NOT DEFINED
	原因	PUBLIC疑似命令のオペランドに，未定義シンボル名を記述しています.
	プログラムの処理	未定義シンボル名の記述を無視します.
	ユーザの処置	シンボルを定義して再実行してください.
エラー番号 33	メッセージ	ASSUME SEGMENT IS NOT DEFINED
	原因	ASSUME疑似命令のオペランドに未定義シンボルを記述しています.
	プログラムの処理	未定義シンボルの代りに予約語NOTHING を記述したものとみなします.
	ユーザの処置	正しいセグメント名またはグループ名を記述して再実行してください.
エラー番号 34	メッセージ	GROUP ELEMENT IS ILLEGAL
	原因	GROUP疑似命令のオペランドにセグメント名以外のシンボルを記述しています.
	プログラムの処理	セグメント名以外のシンボル名の記述を無視します.
	ユーザの処置	正しいセグメント名を記述して再実行してください.
エラー番号 35	メッセージ	INTERNAL STACK OVERFLOW-TOO COMPLICATED SYNTAX
	原因	式の構文が複雑過ぎます.
	プログラムの処理	式の評価値は不定となります.
	ユーザの処置	単純な式に置き換えて再実行してください.

エラー番号 36	メッセージ	SIZE OVERFLOW
	原因	一つの領域確保疑似命令で確保する領域のサイズまたはセグメント・サイズが64Kバイトを越えています.
	プログラムの処理	(i)一つの領域確保疑似命令による場合,その疑似命令を無視します. (ii)セグメント・サイズの場合，そのまま処理を続けます.
	ユーザの処置	セグメント・サイズが64Kバイトを越えないように修正して再実行してください.
エラー番号 37	メッセージ	THIS DIRECTIVE CANNOT BE USED IN THIS VERSION
	原因	現バージョンのプログラムではサポートされていない疑似命令を記述しています.
	プログラムの処理	サポートしていない疑似命令を無視します.
	ユーザの処置	なし
エラー番号 38	メッセージ	TOO MANY SEGMENT IN SOURCE TEXT-ALLOWED ONLY UP TO 32
	原因	ソース・モジュールに32個を越えるセグメントを記述しています.
	プログラムの処理	先頭から32個迄を有効とし，33個目以降を無視します.
	ユーザの処置	セグメント数を32個以内にして再実行してください.
エラー番号 39	メッセージ	SEGMENT/PROC NESTING ALLOWED ONLY TO A DEPTH OF 16
	原因	SEGMENTまたはPROC疑似命令のネスト・レベルの和が16を越えています.
	プログラムの処理	16レベル迄を有効とし，17レベル目以降を無視します.
	ユーザの処置	ネスト・レベルの和が16以下となるように修正して再実行してください.
エラー番号 40	メッセージ	CANNOT HAVE STRING LONGER THAN 2 CHARACTERS
	原因	DB疑似命令のオペランド以外で，長さが2を越える文字列を記述しています.
	プログラムの処理	先頭2文字を有効とし，3文字目以降を無視します.
	ユーザの処置	長さが2以下の文字列にして再実行してください.
エラー番号 41	メッセージ	LITERAL IS ILLEGAL
	原因	長さが0の文字列定数（''）を記述しています.
	プログラムの処理	定数0が記述されているものとみなします.
	ユーザの処置	正しい文字列定数に修正して再実行してください.

エラー番号 42	メッセージ	OPERAND OF OFFSET OPERATOR IS ILLEGAL
	原因	OFFSET演算子のオペランドに，記述できない項を記述しています.
	プログラムの処理	OFFSET演算子およびそのオペランドを無視します.
	ユーザの処置	OFFSET演算子のオペランドに正しい項を記述して再実行してください.
エラー番号 43	メッセージ	OPERAND OF SEG OPERATOR IS ILLEGAL
	原因	SEG演算子のオペランドに，記述できない項を記述しています.
	プログラムの処理	SEG演算子およびそのオペランドを無視します.
	ユーザの処置	SEG演算子のオペランドに正しい項を記述して再実行してください.
エラー番号 44	メッセージ	OPERAND OF TYPE OPERATOR IS ILLEGAL
	原因	TYPE演算子のオペランドに，記述できない項を記述しています.
	プログラムの処理	TYPE演算子およびそのオペランドを無視します.
	ユーザの処置	TYPE演算子のオペランドに正しい項を記述して再実行してください.
エラー番号 45	メッセージ	SEGMENT OVERRIDE IS ILLEGAL
	原因	セグメント・レジスタ名以外でセグメント・オーバライドしています.
	プログラムの処理	セグメント・オーバライドを無視します.
	ユーザの処置	正しくセグメント・オーバライドを記述して再実行してください.
エラー番号 46	メッセージ	OPERAND OF SEGMENT OVERRIDE OPERATOR IS ILLEGAL
	原因	：演算子のオペランドに，記述できない項を記述しています.
	プログラムの処理	：演算子およびそのオペランドを無視します.
	ユーザの処理	：演算子のオペランドに正しい項を記述して再実行してください.
エラー番号 47	メッセージ	OPERAND OF LENGTH OPERATOR IS ILLEGAL
	原因	LENGTH演算子のオペランドに，記述できない項を記述しています.
	プログラムの処理	LENGTH演算子およびそのおペランドを無視します.
	ユーザの処置	LENGTH演算子のオペランドに正しい項を記述して再実行してください.
エラー番号 48	メッセージ	OPERAND OF PTR OPERATOR IS ILLEGAL
	原因	PTR演算子のオペランドに，記述できない項を記述しています.
	プログラムの処理	PTR演算子およびそのオペランドを無視します.
	ユーザの処置	PTR演算子のオペランドに正しい項を記述して再実行してください.

エラー番号 49	メッセージ	INDEX EXPRESSION IS ILLEGAL
	原因	不当なレジスタ式を記述しています.
	プログラムの処理	レジスタ式の評価値は不定となります.
	ユーザの処置	正しいレジスタ式を記述して再実行してください.
エラー番号 50	メッセージ	TOO DUP NESTING-ALLOWED ONLY UP TO 8
	原因	DUPのネスト・レベルが8を越える記述をしています.
	プログラムの処理	DUPを使っている疑似命令を無視します.
	ユーザの処置	DPUのレベルを8以下に修正して再実行してください.
エラー番号 51	メッセージ	NUMBER OF OPERANDS DO NOT MATCH
	原因	機械語命令のオペランドの数が少な過ぎるまたは多過ぎます.
	プログラムの処理	その機械語命令の最大命令語長分のNOP命令を生成します.
	ユーザの処置	正しいオペランドを記述して再実行してください.
エラー番号 52	メッセージ	CANNOT INTERPRET OPERAND TYPE
	原因	機械語命令のオペランドにエラーがあります.
	プログラムの処理	その機械語命令の最大命令語長分のNOP命令を生成します.
	ユーザの処置	正しいオペランドを記述して再実行してください.
エラー番号 53	メッセージ	ALREADY DEFINED AS PUBLIC SYMBOL
	原因	PUBLIC疑似命令のオペランドに，既に他のPUBLIC疑似命令で宣言したシンボルを記述しています.
	プログラムの処理	重複宣言しているシンボル名の記述を無視します.
	ユーザの処置	重複しているシンボル名を削除して再実行してください.
エラー番号 54	メッセージ	CANNOT USE HIGH OR LOW OPERATOR IN THIS INSTRUCTION
	原因	HIGHおよびLOW演算子を使用できない機械語命令で，HIGHまたはLOW演算子を記述しています.
	プログラムの処理	文を無視します.
	ユーザの処置	HIGHおよびLOW演算子を使用しないように修正して再実行してください.
エラー番号 55	メッセージ	VALUE FOR RECORD FIELD IS TOO LARGE
	原因	対応するレコード・フィールドでは表しきれない値をレコード・フィールドの初期値として記述しています.
	プログラムの処理	初期値のLSB側のレコード・フィールド長分のビット・パターンで初期化します.
	ユーザの処置	正しい初期値またはレコード・フィールド長に修正して再実行してください.

エラー番号 56	メッセージ	TOO MANY OVERRIDING RECORD VALUE
	原因	レコード定義でのレコード・フィールド数を越える初期値の数でオーバライドしています.
	プログラムの処理	先頭の定義レコード・フィールド数分の初期値のみを有効とし，残りを無視します.
	ユーザの処置	レコードの定義を変更（フィールド数を増やす）か，オーバライドする初期値の数を少くして再実行してください.
エラー番号 57	メッセージ	SEGMENT REGISTER INITIALIZATION IS NOT SEGMENT OR GROUP
	原因	ASSUME疑似命令のオペランドに，セグメント名またはグループ名以外のシンボルを記述しています.
	プログラムの処理	セグメント名またはグループ名以外を記述している指定（セグメント・レジスタ名：シンボル）を無視します.
	ユーザの処置	セグメント名またはグループ名を記述して再実行してください.
エラー番号 58	メッセージ	TOO MANY GROUP IN SOURCE TEXT-ALLOWED ONLY UP TO 16
	原因	ソース・モジュール内に16個を越えるグループを定義しています.
	プログラムの処理	先頭から16個迄を有効とし，17個目以降を無視します.
	ユーザの処置	グループの数を16個以下に修正して再実行してください.
エラー番号 59	メッセージ	GROUP ELEMENT IS NOT SEGMENT
	原因	GROUP疑似命令のオペランドに，セグメント名以外のシンボルを記述しています.
	プログラムの処理	セグメント名以外のシンボルを無視します.
	ユーザの処置	セグメント名を記述して再実行してください.
エラー番号 60	メッセージ	OPERAND MUST BE A LABEL
	原因	END疑似命令のオペランドに記述したシンボルが当モジュール内のレーベルでありません.
	プログラムの処理	オペランドに記述したシンボルを無視します.
	ユーザの処置	当モジュール内のレーベルをオペランドに記述して再実行してください.
エラー番号 61	メッセージ	CHAIN OF EQU IS CIRCULAR
	原因	EQU疑似命令の定義がループしています.
	プログラムの処理	EQU疑似命令を無視します.
	ユーザの処置	正しくEQU疑似命令を記述して再実行してください.

エラー番号 62	メッセージ	SAME SEGMENT MUST HAVE SAME ALIGN-TYPE, COMBINE-TYPE AND CLASS
	原因	セグメントの分割記述において，2個目以降のセグメントの属性に1個目と異なる記述をしています.
	プログラムの処理	1個目のセグメントの属性と同じ属性が記述されているものとみなします.
	ユーザの処置	2個以降のセグメントの属性を削除して再実行してください.
エラー番号 63	メッセージ	THIS SYMBOL CANNOT BE PURGED
	原因	PURGE疑似命令のオペランドにPURGEできないシンボルを記述しています.
	プログラムの処理	PURGEできないシンボルの記述を無視します.
	ユーザの処置	正しいシンボル名を記述して再実行してください.
エラー番号 64	メッセージ	RECORD FIELD WIDTH MUST BE FROM 1 UP TO 16 BITS
	原因	1個のRECORD疑似命令で定義しているレコード・フィールド長が0または16を越えています.
	プログラムの処理	RECORD疑似命令を無視します.
	ユーザの処置	レコード・フィールド長を16以下にして再実行してください.
エラー番号 65	メッセージ	RECORD WIDTH IS TOO BIG-ALLOWED ONLY UP TO 16 BITS
	原因	1個のRECORD疑似命令で定義しているレコード・フィールド長の合計が16を越えています.
	プログラムの処理	RECORD疑似命令を無視します.
	ユーザの処置	レコード・フィールド長の合計を16以下にして再実行してください.
エラー番号 66	メッセージ	CANNOT USE EVEN DIRECTIVE IN BYTE-ALIGNED SEGMENT
	原因	境界属性がBYTEまたはINPAGEのセグメント内でEVEN疑似命令を記述しています.
	プログラムの処理	EVEN疑似命令を無視します.
	ユーザの処置	EVEN疑似命令を削除して再実行してください.
エラー番号 67	メッセージ	ONLY POSITIVE NUMBER IS ALLOWED
	原因	正整数を記述しなければならない構文で，負または0を記述しています.
	プログラムの処理	数値1を記述しているものとみなします.
	ユーザの処置	正しい正整数を記述して再実行してください.
エラー番号 68	メッセージ	CANNOT OVERRIDDEN WITH OTHER BUT STRING
	原因	文字列で初期化した構造体フィールドに，文字列以外でオーバライドしています.
	プログラムの処理	文字列による初期値を有効とし，オーバライド指定を無視します.
	ユーザの処置	構造体フィールドの初期化を文字列以外とするかオーバライド指定を文字列で行なって再実行してください.

エラー番号 69	メッセージ	TOO MANY OVERRIDING
	原因	構造体名またはレコード名による初期化で，オーバライドしている項の数を定義よりも多く記述しています.
	プログラムの処理	先頭から定義しているフィールド数分の項を有効とし，残る項を無視します.
	ユーザの処置	構造体またはレコードの定義のフィールド数を増やすかオーバライドしている項の数を減らして再実行してください.
エラー番号 70	メッセージ	ILLEGAL DIRECTIVE IN STRUC
	原因	STRUC～ENDS内に記述できない文を記述しています.
	プログラムの処理	記述できない文を無視します.
	ユーザの処置	記述できない文を削除して再実行してください.
エラー番号 71	メッセージ	SYMBOL IS NOT STRUCTURE OR RECORD NAME
	原因	構造体名またはレコード名を記述すべき構文（領域の確保または定数の定義）で，構造体名またはレコード名以外のシンボルを記述しています.
	プログラムの処理	文を無視します.
	ユーザの処置	シンボル名として構造体名またはレコード名を記述して再実行してください.
エラー番号 72	メッセージ	THIS STRUCTURE DEFINITION WILL BE IGNORED
	原因	構造体のサイズが64Kバイトを越えているまたは構造体フィールドが1個もありません.
	プログラムの処理	構造体名および構造体フィールド名の定義を無視します.
	ユーザの処置	正しく構造体を定義して再実行してください.
エラー番号 73	メッセージ	OPERAND OF SHORT OPERATOR IS ILLEGAL
	原因	SHORT演算子のオペランドにレーベル以外を記述しています.
	プログラムの処理	記述した機械語の最大バイト数分のＮＯＰ命令を生成します.
	ユーザの処置	SHORT演算子を削除するかレーベルを記述して再実行してください.
エラー番号 74	メッセージ	OPERAND OF SIZE OPERATOR IS ILLEGAL
	原因	ＳＩＺＥ演算子のオペランドに，記述できない項を記述しています.
	プログラムの処理	ＳＩＺＥ演算子およびそのオペランドを無視します.
	ユーザの処置	ＳＩＺＥ演算子のオペランドに正しい項を記述して再実行してください.
エラー番号 76	メッセージ	OPERAND MUST BE POSITIVE OR ZERO
	原因	0または正整数を記述しなければならない構文に負の数を記述しています.
	プログラムの処理	数値0が記述してあるものとみなします.
	ユーザの処置	0または正整数を記述して再実行してください.

エラー番号 77	メッセージ	RELOCATABLE VALUE IS ILLEGAL IN THIS INSTRUCTION
	原因	オペランドにリロケータブルなシンボル（式）を記述できない機械語にリロケータブルなシンボル（式）を記述しています.
	プログラムの処理	リロケータブルなシンボル（式）を無視します.
	ユーザの処置	数値（シンボル）を記述して再実行してください.
エラー番号 79	メッセージ	ONLY RECORD OR RECORD FIELD NAME IS ALLOWED TO WIDTH
	原因	WIDTH演算子のオペランドにレコード名またはレコード・フィールド名以外を記述しています.
	プログラムの処理	WIDTH演算子及びそのオペランドを無視します.
	ユーザの処置	WIDTH演算子のオペランドにレコード名またはレコード・フィールド名を記述して再実行してください.
エラー番号 80	メッセージ	ONLY RECORD FIELD NAME IS ALLOWED TO MASK
	原因	MASK演算子のオペランドにレコード・フィールド名以外を記述しています.
	プログラムの処理	MASK演算子およびそのオペランドを無視します.
	ユーザの処置	MASK演算子のオペランドにレコード・フィールド名を記述して再実行してください.
エラー番号 81	メッセージ	RIGHT OPERAND TO DOT(.) IS ILLEGAL
	原因	演算子の右オペランドに構造体フィールド名以外を記述しています.
	プログラムの処理	演算子およびその右オペランドを無視します.
	ユーザの処置	演算子のオペランドに構造体フィールド名を記述して再実行してください.
エラー番号 82	メッセージ	THIS FIELD CANNOT BE OVERRIDDEN
	原因	構造体名による領域確保で，オーバライドできないフィールドに対してオーバライドしています.
	プログラムの処理	構造体の定義時の初期値を有効とし，オーバライドの指定を無視します.
	ユーザの処置	構造体の定義に於ける対応するフィールドをオーバライド可能な形式に変更するかオーバライドの指定を削除して再実行してください.
エラー番号 83	メッセージ	TOO LONG OVERRIDING STRING-TRUNCATED
	原因	構造体名による領域確保で，オーバライドしている文字列の長さが構造体定義時のフィールドの長さを超えています.
	プログラムの処理	対応するフィールド長分の文字列のみを有効とし余分な（文字列の後の方）を無視します.
	ユーザの処置	構造体フィールドの長さを大きくするか，オーバライドする文字列の長さを構造体定義時のフィールド長以下にして再実行してください.

エラー番号 84	メッセージ	DEFAULT VALUE FOR RECORD FIELD IS TOO BIG
	原因	RECORD疑似命令のオペランドで，レコード・フィールドの初期値が対応するフィールド長で表しきれない．
	プログラムの処理	初期値のＬＳＢ側のレコード・フィールド長分を有効とし，残るＭＳＢ側のビット・パターンを無視する．
	ユーザの処置	レコード・フィールド長を大きくするか，初期値をフィールド長に合わせた値にして再実行してください．
エラー番号 85	メッセージ	TOO BIG VALUE FOR DB
	原因	ＤＢ疑似命令のオペランド値に－２５５～２５５の範囲外の記述をしています．
	プログラムの処理	数値０が記述されたものとみなします．
	ユーザの処置	−255～255 の値を記述して再実行してください．
エラー番号 86	メッセージ	VARIABLE OR LABEL IS NOT ALLOWED FOR DB
	原因	ＤＢ疑似命令のオペランドにＨＩＧＨまたはＬＯＷ演算子を使っていないレーベル／変数を記述しています．
	プログラムの処理	ＬＯＷ演算子が記述されたものとみなします．
	ユーザの処置	ＨＩＧＨまたはＬＯＷ演算子を記述して再実行してください．
エラー番号 87	メッセージ	RELOCATABLE VALUE IS NOT ALLOWED FOR DB
	原因	ＤＢ疑似命令のオペランドにセグメント名またはグループ名を記述しています．
	プログラムの処理	数値０が記述されたものとみなします．
	ユーザの処置	ＤＢ疑似命令をＤＷ疑似命令に変更して再実行してください．
エラー番号 88	メッセージ	OPERAND TO ORG MUST BE ABSOLUTE OR IN THIS SEGMENT
	原因	ＯＲＧ疑似命令のオペランドに不当な式を記述しています．
	プログラムの処理	ＯＲＧ疑似命令を無視します．
	ユーザの処置	ＯＲＧ疑似命令のオペランドにアブソリュートな数値または当セグメント内のリロケータブルな数値を記述して再実行してください．
エラー番号 89	メッセージ	ORG VALUE IS LESS THAN CURRENT LOCATION
	原因	ＯＲＧ疑似命令のオペランドに，カレント・ロケーション・カウンタの値より小さな値を記述しています．
	プログラムの処理	ＯＲＧ疑似命令を無視します．
	ユーザの処置	カレント・ロケーション　カウンタの値より大きな値をＯＲＧ疑似命令のオペランドに記述して再実行してください．

第8章 実 行 例

(1) 入 力 例

```
-RUN RA116 :F1:FORMAL XR

UPD70108/70116 ASSEMBLER V1.0  [30 Oct 85]
   Copyright (C) 1984 NEC Corporation

ASSEMBLY COMPLETE.     0 ERROR(S) FOUND
```

(2) アセンブル・リスト

```
UPD70108/70116 ASSEMBLER V1.0 FORMAL_MAIN                                DATE:18/03/85          PAGE:    1

SOURCE FILE : :F1:FORMAL.ASM
OBJECT FILE : :F1:FORMAL.REL (FORMAL_MAIN)
COMMAND     : :F1:FORMAL XR

LOC  OBJ                    LINE      SOURCE

                               1              NAME    FORMAL_MAIN
                               2
                               3      ;       ************************************************
                               4      ;       *                                              *
                               5      ;       *       FORMAL MAIN ROUTINE UNDER MP/M-86      *
                               6      ;       *                                              *
                               7      ;       ************************************************
                               8
                               9              PUBLIC  CODE_LEN.CODE_BASE.M8080
                              10              PUBLIC  DATA_LEN.DATA_BASE
                              11              PUBLIC  COMMANDBUFF
                              12
                              13      CGROUP  GROUP   CODE
                              14      DGROUP  GROUP   DATA.CONST.STACK.MEMORY
                              15
----                          16      DATA    SEGMENT WORD PUBLIC 'DATA'
0000 ????                     17              CODE_LEN        DW      ?               ; BASE PAGE
0002 ??                       18                              DB      ?
0003 ????                     19              CODE_BASE       DW      ?
0005 ??                       20              M8080           DB      ?
0006 ????                     21              DATA_LEN        DW      ?
0008 ??                       22                              DB      ?
0009 ????                     23              DATA_BASE       DW      ?
                              24
000B                          25              ORG             80H
0080 (128                     26              COMMANDBUFF     DB      80H DUP (?)
     ??
     )
----                          27      DATA    ENDS
                              28
----                          29      CONST   SEGMENT WORD PUBLIC 'CONST'
----                          30      CONST   ENDS
                              31
----                          32      STACK   SEGMENT WORD STACK 'STACK'
0000 (16                      33              DB      10H DUP (?)
     ??
     )
0010                          34              STACK_BOTTOM    LABEL   WORD
----                          35      STACK   ENDS
                              36
----                          37      MEMORY  SEGMENT WORD MEMORY 'MEMORY'
0000                          38              MEMORY_BOTTOM   LABEL   BYTE
----                          39      MEMORY  ENDS
                              40
----                          41      CODE    SEGMENT WORD PUBLIC 'CODE'
                              42              ASSUME  PS:CGROUP,DS0:DGROUP,SS:DGROUP,DS1:DGROUP
                              43
                              44              EXTRN   MAIN:NEAR,EXIT:NEAR
                              45
0000 8CD8                     46              MOV     AW,DS0

UPD70108/70116 ASSEMBLER V1.0 FORMAL_MAIN                                DATE:18/03/85          PAGE:    2

LOC  OBJ                    LINE      SOURCE

0002 8ED0                     47              MOV     SS,AW
0004 BC1000             R     48              MOV     SP,OFFSET DGROUP:STACK_BOTTOM
                              49
0007 E80000             E     50              CALL    MAIN                    ; CALL ACTUAL MAIN ROUTINE
000A E80000             E     51              CALL    EXIT                    ; RETURN CONTROL TO MP/M-86
                              52
----                          53      CODE    ENDS
                              54
                              55              END

UPD70108/70116 ASSEMBLER V1.0 FORMAL_MAIN                                DATE:18/03/85          PAGE:    3

*** XREF SYMBOL LISTING ***

NAME              TYPE     VALUE ATTRIBUTES   XREF

??SEG             SEGMENT        SIZE=0000H PARA PUBLIC
CGROUP            GROUP          CODE  13# 42
CODE              SEGMENT        SIZE=000DH WORD PUBLIC 'CODE'  13# 41# 53#
CODE_BASE         WORD      0003 DATA PUBLIC  9# 19#
CODE_LEN          WORD      0000 DATA PUBLIC  9# 17#
COMMANDBUFF       BYTE      0080 DATA PUBLIC  11# 26#
CONST             SEGMENT        SIZE=0000H WORD PUBLIC 'CONST'  14# 29# 30#
DATA              SEGMENT        SIZE=0100H WORD PUBLIC 'DATA'  14# 16# 27#
DATA_BASE         WORD      0009 DATA PUBLIC  10# 23#
DATA_LEN          WORD      0006 DATA PUBLIC  10# 21#
DGROUP            GROUP          DATA CONST STACK MEMORY  14# 42 42 42 42 46
EXIT              NEAR      0000 CODE EXTRN  44# 51
M8080             BYTE      0005 DATA PUBLIC  9# 20#
MAIN              NEAR      0000 CODE EXTRN  44# 50
MEMORY            SEGMENT        SIZE=0000H WORD MEMORY 'MEMORY'  14# 37# 39#
MEMORY_BOTTOM     BYTE      0000 MEMORY  38#
STACK             SEGMENT        SIZE=0010H WORD STACK 'STACK'  14# 32# 35#
STACK_BOTTOM      WORD      0010 STACK  34# 48

ASSEMBLY COMPLETE.    0 ERROR(S) FOUND
```

空白ページ

付録1　予約語 一覧

(1) レジスタ，フラグ

AH	AL	AW	BH	BL	BP
BW	CH	CL	PS	CW	DH
IY	DL	DS0	DW	DS1	IX
SP	SS	PSW	R	DIR	CY

(2) 機 械 語

ADD	ADD4S	ADDC	ADJ4A	ADJ4S	ADJBA
ADJBS	AND	BC	BCWZ	BE	BGE
BGT	BH	BL	BLE	BLT	BN
BNC	BNE	BNH	BNL	BNV	BNZ
BP	BPE	BPO	BR	BRK	BRKEM
BRKV	BUSLOCK	BV	BZ	CALL	CALLN
CHKIND	CLR	CLR1	CMP	CMP4S	CMPBK
CMPBKB	CMPBKW	CMPM	CMPMB	CMPMW	CVTBD
CVTBW	CVTDB	CVTWL	DBNZ	DBNZE	DBNZNE
DEC	DI	DISPOSE	DIV	DIVU	EI
EXT	FPO1	FPO2	HALT	IN	INC
INM	INS	LDEA	LDM	LDMB	LDMW
MOV	MOVBK	MOVBKB	MOVBKW	MUL	MULU
NEG	NOP	NOT	NOT1	OR	OUT
OUTM	POLL	POP	PREPARE	PUSH	REP
REPC	REPE	REPNC	REPNE	REPNZ	REPZ
RET	RETEM	RETI	ROL	ROL4	ROLC
ROR	ROR4	RORC	SET	SET1	SHL
SHR	SHRA	STM	STMB	STMW	SUB
SUB4S	SUBC	TEST	TEST1	TRANS	TRANSB
XCH	XOR				

(3) 疑似命令，演算子及びそのオペランド

ABS	ASSUME	AT	BYTE	COMMON	DB
DBS	DD	DDS	DUP	DW	DWS
DWORD	END	ENDP	ENDS	EQ	EQU
EVEN	EXTRN	FAR	GE	GROUP	GT
HIGH	INPAGE	LABEL	LE	LENGTH	LOW
LT	MASK	MEMORY*	MOD	NAME	NE
NEAR	NOTHING	OFFSET	ORG	PAGE	PARA
PROC	PTR	PUBLIC	PURGE	RECORD	SEG
SEGMENT	SHORT	SIZE	STACK*	STRUC	THIS
TYPE	WIDTH	WORD	?	??SEG	

注） ＊印のものはシンボルとして使用することができます．

付録２　疑似命令 一覧

疑似命令	機　　能	参照ページ
SEGMENTと ENDS	セグメントを定義します.	2－82
PROCと ENDP	手続き（サブルーチン）を定義します.	2－84
ASSUME	セグメント・レジスタにセットしてある，セグメントまたはグループのパラグラフ値をアセンブラに指示します.	2－85
GROUP	グループに属するセグメントを宣言します.	2－87
EQU	ネームを定義します.	2－88
LABEL	レーベルまたは変数を定義します.	2－88
PURGE	シンボルをシンボル・テーブルから削除します.	2－89
ORG	ロケーション・カウンタの値（セグメント内相対）を設定します.	2－40
EVEN	ロケーション・カウンタの値を偶数にします.	2－40
STRUCと ENDS	構造体を定義します.	2－41
RECORD	レコードを定義します.	2－42
DB	変数の定義およびバイト領域を確保します.	2－48
DW	変数の定義およびワード（２バイト）領域を確保します.	2－44
DD	変数の定義およびダブルワード（４バイト）領域を確保します.	2－44
DBS	変数の定義およびバイト領域を確保します.	2－45
DWS	変数の定義およびワード領域を確保します.	2－46
DDS	変数の定義およびダブル・ワード領域を確保します.	2－46
NAME	出力オブジェクト・モジュールのモジュール名を定義します.	2－50
PUBLIC	他のモジュール（別のアセンブル単位）から参照するシンボルを宣言します.	2－50
EXTRN	本モジュールで参照する他のモジュールのシンボルを宣言します.	2－51
END	アセンブルを終了します.	2－52

付録3　コントロール 一覧

基本コントロール 一覧

項番	分類		コントロール名	短縮形	省略時の既定値
1	入出力ファイル指定用	オブジェクト・モジュール・ファイル	OBJECT〔(ファイル名)〕 NOOBJECT	OJ,OBJ NOOJ, NOOBJ	OJ(ソース・モジュール・ファイル名)ただし，ファイル・タイプは'REL'
		リスト・ファイル	PRINT〔(ファイル名)〕 NOPRINT	PR,PRI NOPR, NOPRI	PR(ソース・モジュール・ファイル名)ただし，ファイル・タイプは'PRN'
		ワーク・ファイル	WORKFILES(ドライブ名)	WF,WOR	WF(オブジェクト・モジュール・ファイルのドライブ名)
2	出力リスト項目指定用	シンボル・テーブル・リスト	SYMBOL NOSYMBOL	SB,SYM NOSB, NOSYM	NOSB
		クロス・レファレンス・リスト	XREF NOXREF	XR,XRE NOXR, NOXRE	NOXR
3	出力リスト行指定用	一行のカラム数	PAGEWIDTH(カラム数)	PW,WID	PW(130) ただし，コンソールの場合は80
		1ページの行数	PAGELENGTH(行数)	PL,LEN	PL(66)
4	処理日付指定		DATE(文字列)	DA,DAT	DA(ISIS-IIのDATEコマンドで最後に指定した日付)

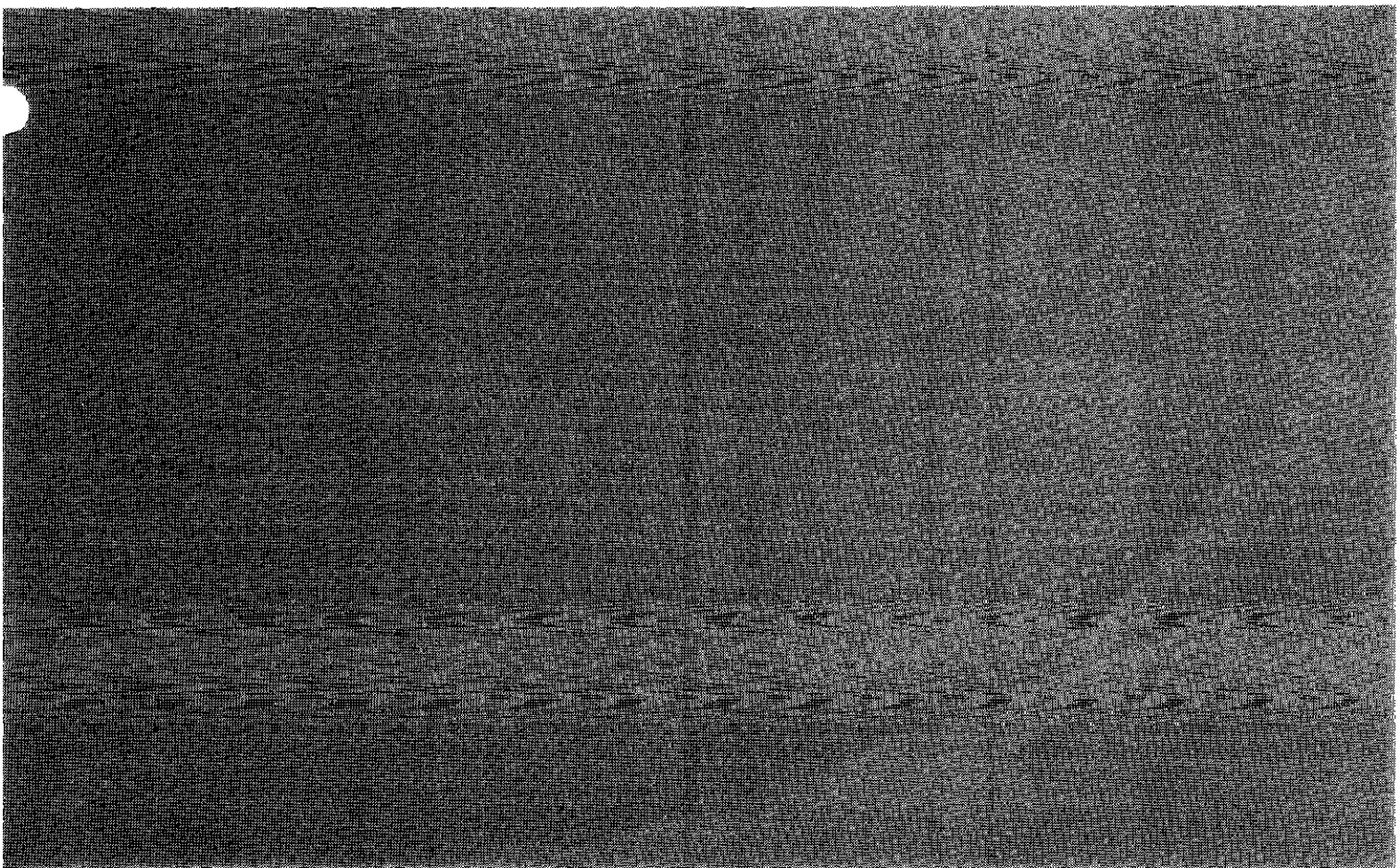
汎用コントロール一覧

項番	分類	コントロール名	短縮形	入力形式
1	ソース・ファイル入力用	INCLUDE	IC,INC	$ INCLUDE(ファイル名)
2	リスト出力制御用	TITLE	TT,TIT	$ TITLE (文字列)
		EJECT	EJ,EJE	$ EJECT
		LIST	LI,LIS	$ LIST
		NOLIST	NOLI, NOLIS	$ NOLIST
		GEN	GE	$ GEN
		NOGEN	NOGE	$ NOGEN
		GENONLY	GO	$ GENONLY

第III編

μPD70108,μPD70116

リンカの取扱方法



第１章　ソフトウエア概要

１．１　リンカの機能概要

リンカはμPD70108，μPD70116リロケータブル・アセンブラの出力したオブジェクト・モジュール（またはμPD70108，μPD70116リンカの出力したロード・モジュール）を結合してロード・モジュールを出力します．

リンカは次の処理を行なって，モジュールを結合します．

(1)　入力モジュールの統合

(2)　入力モジュール内で解決されない外部名を定義しているモジュールをライブラリから探して統合

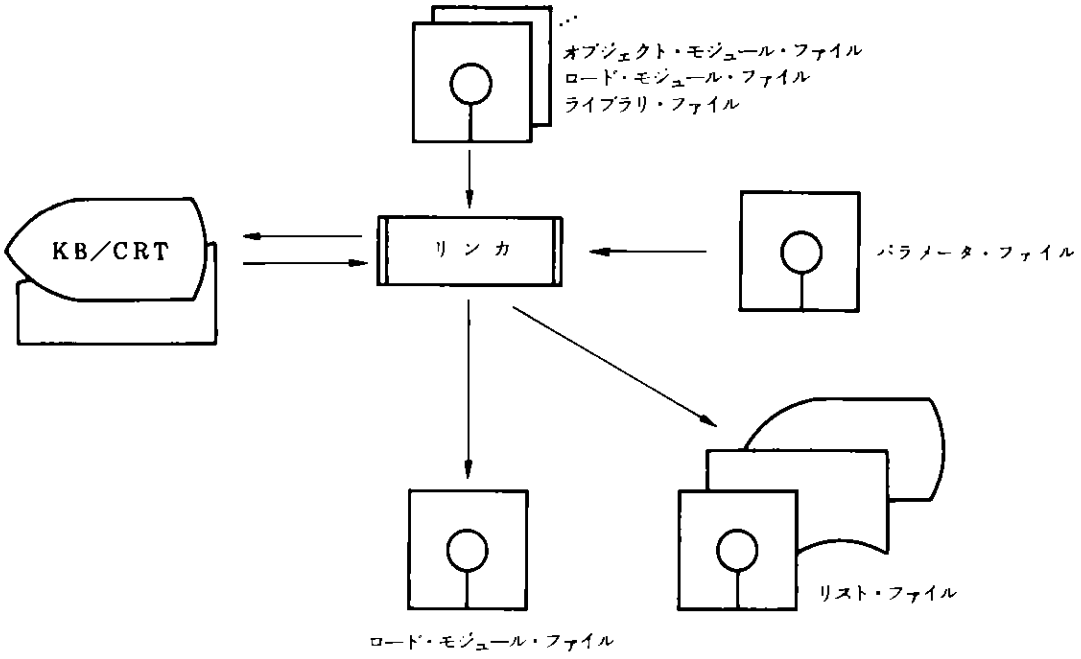
(3)　統合したモジュール内セグメントのアドレス決定

(4)　統合したモジュール内テキスト（オペレーション・コード）の修正

１．２　システム構成

リンカのシステム構成を図１－１に示します．

図１－１　システム構成



第2章 機　　能

2.1 入力ファイルの種類

(1) オブジェクト・モジュール・ファイル

アセンブラの出力するオブジェクト・モジュールのファイルです．

(2) ロード・モジュール・ファイル

リンカの出力するロード・モジュールのファイルです．

(3) ライブラリ・ファイル

ライブラリアンの出力するファイルです．

ライブラリ・ファイルは複数のオブジェクト・モジュールを含みます．

2.2 機　　能

本リンカは，次の処理を行なってモジュールを結合してロード・モジュールを出力します．

(1) 入力モジュールの統合

コントロールで指定されたファイルのオブジェクト・モジュールまたはロード・モジュールを統合します．

(2) 自動ライブラリ入力

(1)で統合したモジュール内で参照している外部参照名が未定義である場合，未定義である外部参照名を参照しているモジュールをライブラリ・ファイルから探して統合します．

ただし，自動ライブラリ入力には次の順序性がありますので注意が必要です．

リンカはライブラリの指定以前に指定されたオブジェクト・モジュール，自動ライブラリ入力したオブジェクト・モジュールおよびロード・モジュール内で未定義である外部参照名を参照しているオブジェクト・モジュールを探して統合します．

例　LK70116 OMF1，OMF2，LBF1，OMF3，LBF2 TO LMF

- オブジェクト・モジュールOMF1とOMF2に対してライブラリLBF1から自動ライブラリ入力を行ないます．
- オブジェクト・モジュールOMF1，OMF2，ライブラリ LBF1から自動ライブラリ入力したオブジェクト・モジュールおよびオブジェクト・モジュールOMF3に対してライブラリLBF2から自動ライブラリ入力を行ないます．
  したがって，オブジェクト・モジュールOMF3に対してライブラリLBF1からは自動ライブラリ入力は行ないません．

(3) 配置アドレスの決定

統合したモジュール内のセグメントの配置アドレスをコントロール（ORDER，ADDRESSES，

RESERVE）にもとづいて決定します．

(4) 統合したモジュール内のテキスト（オブジェクト・コード）を修正します．

2・2・1 配置アドレスの割付け

リンカは次の規則に従ってセグメントに配置アドレスを割付けます．

(1) 同名セグメントは連続したまたは重複したアドレスに割付けられます．注(1)

同名セグメントとは，セグメント名とクラス名がまったく同じセグメントを言います．

(2) 同じクラス名をもつセグメントは連続した順序でアドレスが割付けられます．注(2)（連続したアドレスとは限りません．）

(3) 結合属性（combination type）が，MEMORYであるセグメントは，結合属性がMEMORY以外のセグメントの後に配置されます．

(4) セグメントの配置アドレスは，境界属性を満たす値が割付けられます．

同名セグメントの境界属性は，各セグメントの境界属性をすべて満たす属性となります．

(5) ORDERまたはADDRESSESコントロールでセグメント名とクラス名の指定をした場合，セグメント名の指定が優先されます．

例 S1，S4：セグメント名

CODE：クラス名とし，S1のクラス名をCODEとすると，

ORDER(SEGMENTS(S1，S4))

ORDER(CLASSES(CODE))

の指定では，S1，S4，S1以外のクラス名がCODEのセグメントの順に配置されます．

(6) アブソリュート・セグメントおよびADDRESSESコントロールで配置アドレスを指定したセグメントは指定のアドレスに配置されます．

ただし，指定したセグメントの境界属性（align type）と指定アドレスが矛盾する場合は境界属性が優先されます．

(7) RESERVEコントロールで指定した領域には，(5)の場合を除きセグメントは配置されません．

(8) ORDERコントロールで指定した最初のセグメント（ORDERコントロールを記述しない場合は最初に入力したセグメント）の先頭アドレスはそのセグメントにADDRESSESコントロールで配置アドレスを指定しない限りは00200H番地となります．ORDERコントロールまたはADDRESSESコントロールでクラス名を指定した場合はそのクラス名をもつセグメントの内，最初に入力されたセグメントの先頭アドレスが上記のアドレスになります．

最初に入力されたセグメントとは，入力ファイル・リストの最初に指定したファイルのモジュール内先頭セグメントです．

(9) ORDERコントロールを記述しない場合のセグメントの配置順序は(1)，(2)，(3)および入力されたセグメントの順序から決定されます．

注(1) 同一名セグメント内のセグメントの順序は入力順となります．

注(2) 同じクラス名をもつセグメントの順序は入力順となります．

注(8) リンカはグループ（GROUP）に対する配置アドレス上の配慮はしませんのでORDERコントロール，ADDRESSESコントロールまたはセグメントのクラス名でグループに属する全セグメントが64Kバイト以内に納まるように指定してください．64Kバイト以内に納まらない場合はエラーとなります．

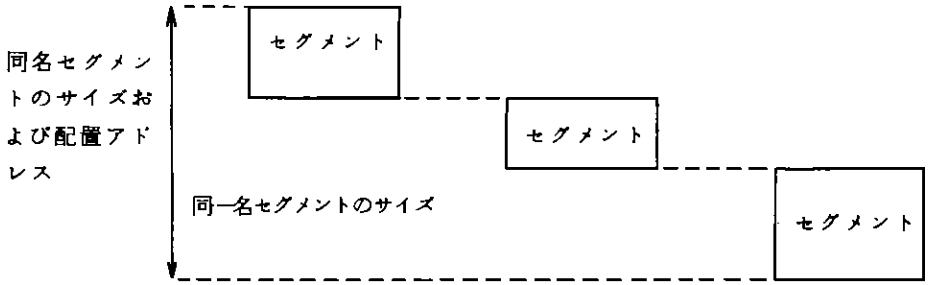
2・2・2 同一名セグメントのサイズおよびアドレス割付け

セグメント名とクラス名がまったく同一であるセグメントに対してリンカはセグメントの結合属性（combination type）によってサイズおよび配置アドレスを決定します．

(1) 結合属性がPUBLICの場合

各セグメントのサイズの和を同名セグメントのサイズとします．各セグメントは連続した領域に割付けられます．

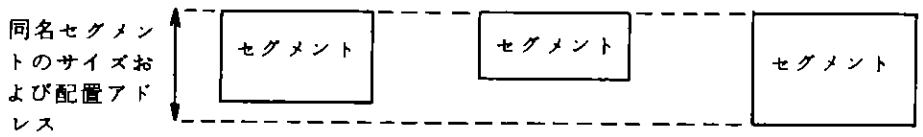
図2－1 結合属性がPUBLICの場合



(2) 結合属性がCOMMON，ABSOLUTEまたはMEMORYの場合

各セグメントのサイズの最大値を同名セグメントのサイズとします．各セグメントの先頭アドレスが同じになるように配置アドレスが割付けられます．

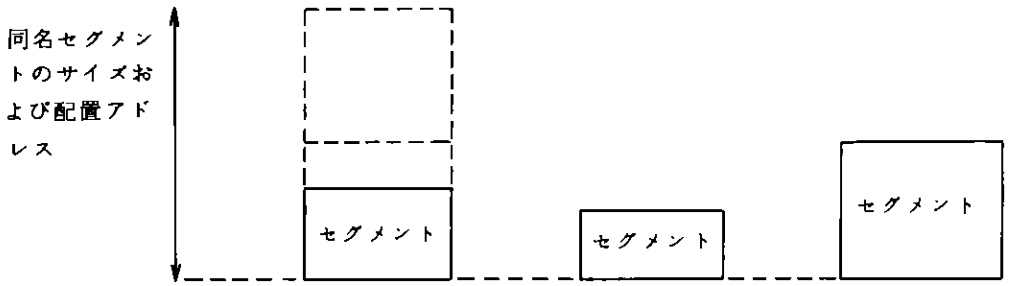
図2－2 結合属性がCOMMON，ABSOLUTEまたはMEMORYの場合



(3) 結合属性がSTACKの場合

各セグメントのサイズの和を同名セグメントのサイズとします．各セグメントの最終アドレスが同一となる様に配置アドレスが割付けられます．

図２－３　結合属性がＳＴＡＣＫの場合



２・２・３ オブジェクト・コードの更新

セグメントの配置アドレスを決定後，オブジェクト・コードをリロケーション情報に基づいて更新します．

例

入力モジュール

ファイル名　：FILE1	ファイル名　：FILE2
モジュール名：MOD1	モジュール名：MOD2
セグメント名　CODE 結合属性　PUBLIC 境界属性　PARA クラス名　CODE サイズ　50H	セグメント名　CODE 結合属性　PUBLIC 境界属性　WORD クラス名　CODE サイズ　145H
セグメント名　DATA 結合属性　PUBLIC 境界属性　WORD クラス名　DATA サイズ　80H	セグメント名　DATA 結合属性　PUBLIC 境界属性　PARA クラス名　DATA サイズ　33H
セグメント名　STACKS 結合属性　PUBLIC 境界属性　WORD クラス名　STACK サイズ　13H	セグメント名　STACKS 結合属性　PUBLIC 境界属性　WORD クラス名　STACK サイズ　27H

コントロール

```
FILE1,FILE2 TO FILE3
ORDER(SEGMENTS(CODE,DATA,STACKS))
ADDRESSES(SEGMENTS(CODE(200H)))
NAME(MOD3)
```

出力モジュール

ファイル名　：FILE3

モジュール名：MOD3

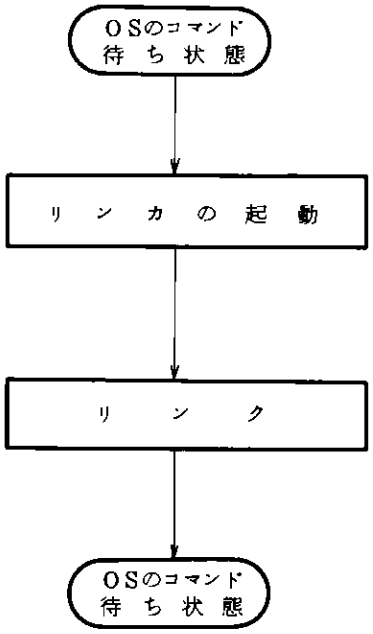
200H	セグメント名　CODE
	結合属性　PUBLIC
	境界属性　PARA
	クラス名　CODE
24FH	サイズ　50H
250H	セグメント名　CODE
	結合属性　PUBLIC
	境界属性　WORD
	クラス名　CODE
394H	サイズ　145H
3A0H	セグメント名　DATA
	結合属性　PUBLIC
	境界属性　WORD
	クラス名　DATA
3CFH	サイズ　30H
3D0H	セグメント名　DATA
	結合属性　PUBLIC
	境界属性　PARA
	クラス名　DATA
402H	サイズ　33H
404H	セグメント名　STACKS
	結合属性　PUBLIC
	境界属性　WORD
	クラス名　STACK
416H	サイズ　13H
417H	セグメント名　STACKS
	結合属性　PUBLIC
	境界属性　WORD
	クラス名　STACK
43DH	サイズ　27H

第3章 操作法

3.1 操作手順

・リンカの操作手順を図3－1に示します.

図3－1 操作手順



・次の指定によりリンカを起動します.

```
-RUN 〔:d:〕LK116 <パラメータ・ファイル名〔コントロール〔…〕〕↵
```

または

```
-RUN 〔:d:〕LK116入力ファイル名〔,…〕TO出力ファイル名〔コントロール〔…〕〕↵
```

パラメータ・ファイル名：リンカへのコマンドを格納しているファイル（パラメータ・ファイル）のファイル名（3.1.1を参照してください.）

入力ファイル名　：リンクするオブジェクト・モジュール・ファイル，ロード・モジュール・ファイルまたはライブラリ・ファイルのファイル名

出力ファイル名　：リンクした結果のロード・モジュール・ファイルのファイル名

コントロール　　：3.3を参照してください.

d　　：LK116.86およびLK116.OMnを格納しているＦＤドライブ名（省略時はＦ０となります．）

LK116：リンカのプログラム・ファイル名

・リンカが起動されるとコンソールに実行開始のメッセージが表示され，リンクが開始されます．

```
UPD70108／70116 LINKER V×.× 〔×× ××× ××〕 ←---------実行開始メッセージ
    Copyright （C） 1984 NEC Corporation
```

3．1．1　パラメータ・ファイルの形式

次の形式のファイルをエディタで作成してください．

```
入力ファイル名〔,…〕 TO 出力ファイル名〔 コントロール名〔 …〕〕;
```

・１行は１８２文字以内にしてください．
・１行におさまらない場合は，複数行に分けて記述してください．
・セミコロン（；）から行の終り迄はリンカへのコマンドとしては解釈されません．（コメント）
・空白のみの行も記述できます．
・英小文字は英大文字として解釈されます．（ただし，セグメント名，グループ名およびクラス名は小文字のまま解釈されます．）

例

```
;
; program main linking command
;
MAIN,SUB1,SUB2,SUBLIB.LIB
TO MAIN
ORDER(SEGMENTS(CODE,CONST,DATA,STACK))
ADDRESSES(SEGMENTS(CODE(1000H),DATA(10000H)))
```

(1)　ファイル名規則

(a)　フロッピィ・ディスク上のファイルの場合

```
〔：ドライブ名：〕名称〔．拡張子〕
```

ドライブ名：ファイルを格納しているＦＤがセットされているドライブ名
（Ｆ０｜Ｆ１…）

名称　　：６文字以内の文字列

拡張子　：３文字以内の文字列

＜省略時解釈＞

省略時解釈はファイルの種類毎に異なり，表8－1のとおりとなります．

表8－1

項番	ファイルの種類	ドライブ名	拡張子
1	入力ファイル	F0	REL
2	ロード・モジュール・ファイル		LNK
3	リスト・ファイル		MAP

例① ：F1：MDMAIN.REL……ドライブ名がF1，名称がMDMAIN，拡張子がREL

② SUB1 ……ドライブ名と拡張子を省略

(b) フロッピィ・ディスク以外の場合

：装置名：

装置名には，次のものが指定できます．

- CO…コンソール
- LP…ラインプリンタ

例

① PRINT(：LP：)……リストをLPに出力

3.2 プログラムの終了

リンカは，リンク処理が終了すると次のメッセージをコンソールに表示し，制御をOSに戻します．

・正常終了の場合

```
LINK COMPLETED
```

・異常終了の場合

```
PROGRAM ABORTED
```

3.3 コントロールの指定

コントロールとは，プログラムの実行に先立ってプログラムの動作に必要な情報を指示するためのものです．

リンカの起動時またはパラメータ・ファイルで指定してください．

コントロールには次の１５種類があります．

コントロールは短縮形で入力することもできます．

表３－２　コントロールの種類

No	コントロール名	短縮形	既定値
1	START	ST，STA	——
2	BOOTSTRAP	BS，BOO	——
3	ADDRESSES	AD，ADD	——
4	ORDER	OD，ORD	——
5	RESERVE	RS，RES	——
6	NAME	NA，NAM	NA（出力ロード・モジュール・ファイル名のプライマリネーム）
7	MAP	MA	MA
8	NOMAP	NOMA	
9	PUBLICS	PL，PUB	PL
10	NOPUBLICS	NOPL，NOPUB	
11	SYMBOLS	SB，SYM	SB
12	NOSYMBOLS	NOSB，NOSYM	
13	PRINT〔（ファイル名）〕	PR，PRI	PR（出力ロード・モジュール・ファイル名）ただし，ファイル・タイプはMAP
14	NOPRINT	NOPR，NOPRI	
15	DATE（日付）	DA，DAT	DA（ISIS－ⅡのDATEコマンドで最後に指定した日付）

ADDRESSESおよびORDERコントロールでは，次のキーワードを使います．

No	キーワード	短縮形
1	SEGMENTS	SM，SEG
2	CLASSES	CS，CLA
3	GROUPS	GR，GRO

・次のコントロールは，同じコントロールまたは同種のコントロールを複数回入力した場合，後で指定したコントロールが有効となります．

START，BOOTSTRAP，NAME，MAP／NOMAP，PUBLICS／NOPUBLICS，SYMBOLS／NOSYMBOLS，PRINT／NOPRINT，DATE

起動時のコントロールとパラメータ・ファイルのコントロールの場合は，起動時のコントロールが有効となります．

・次のコントロールは，指定したすべてのコントロールが有効となります．

ADDRESSES，ORDER，RESERVE

3．3．1 プログラム開始アドレスの指定

(1) 機　　能

出力ロード・モジュールのプログラム開始アドレスを指定します．

(2) 形　　式

START（外部定義名）

(3) 説　　明

(a) プログラム開始アドレスを外部定義名で指定します．

(b) STARTコントロールを指定しない場合，入力モジュール内のメイン・モジュールのプログラム開始アドレスが出力ロード・モジュールのプログラム開始アドレスとなります．
また，本指定もメイン・モジュールも無い場合は00000Hになります．

3．3．2 ブート・ストラップの指定

(1) 機　　能

アドレス0FFFF0Hにプログラム開始アドレスへのFAR JMP命令を生成します．

(2) 形　　式

BOOTSTRAP

(3) 説　　明

(a) 出力ロード・モジュールにプログラム開始アドレスがない場合，このコントロールは無視されます．

3．3．3 配置アドレスの指定

(1) 機　　能

セグメントの配置アドレスを指定します．

(2) 形　　式

$$\mathrm{ADDRESSES}\left(\left\{\begin{array}{l}\mathrm{SEGMENTS}(\text{セグメント名}[/\text{クラス名}](\text{アドレス})[,\cdots])\\ \mathrm{CLASSES}\ \ (\text{クラス名}(\text{アドレス})[,\cdots])\\ \mathrm{GROUPS}\ \ \ (\text{グループ名}(\text{アドレス})[,\cdots])\end{array}\right\}[,\cdots]\right)$$

(3) 説　　明

(a) クラス名で配置アドレスを指定した場合，そのクラス名をもつ全セグメントのうちの最初のセグメントのアドレスを指定したものとみなされます．ただし，セグメント名指定でアドレスを指定したセグメントは含まれません．

(b) アドレスは絶対番地（00000H～0FFFFFH）です．

(c) セグメント名／クラス名（アドレス）の指定をした場合，指定のクラス名をもつセグメントのみが，配置アドレス指定の対象となります．

(d) グループへのアドレス指定では，グループのパラグラフ値のみを決定します．
グループを構成するセグメントのアドレス指定を別に行なう必要があります．

3．3．4 配置順序の指定

(1) 機　能

セグメントの配置順序を指定します．

(2) 形　式

ORDER({ SEGMENTS(セグメント名〔／クラス名〕〔,…〕)
　　　 { CLASSES (クラス名〔(セグメント名〔,…〕)〕〔,…〕) } 〔,…〕)

(3) 説　明

(a) クラス名で配置順序を指定した場合，指定のクラス名の全セグメントのうち，セグメント名で配置順序を指定したセグメント以外の配置順序を指定したものとみなされます．

(b) セグメント名／クラス名の指定をした場合，指定のクラス名をもつセグメントのみが，配置順序指定の対象となります．

(c) クラス名(セグメント名〔,…〕の指定をした場合，指定のクラス内でのセグメントの配置順序が(　)内で指定した順序になります．

3．3．5 配置禁止エリアの指定

(1) 機　能

セグメントの配置禁止エリアを指定します．

(2) 形　式

RESERVE(addr1 TO addr2〔,…〕)

addr1≦addr2

(3) 説　明

アドレスは絶対番地（00000H～0FFFFFH）です．

3．3．6 出力ロード・モジュール名の指定

(1) 機　能

出力ロード・モジュール名を指定します．

(2) 形　式

NAME(モジュール名)

(3) 説　明

モジュール名は，“@”，“?”，“_”，英数字から成る68文字以内の文字列です（先頭文字は，“@”，“?”，“_”，英数字でなければなりません）．

3．3．7 リンク・マップの出力指定

(1) 機　能

リンク・マップを出力するか否かを指定します．

(2) 形　式

MAP　…リンク・マップを出力します．

NOMAP…リンク・マップを出力しません．

(3) 説　　明

(a) リンク・マップの内容は次のとおりです．

入力モジュール・リスト，セグメント・マップ，グループ・マップ

(b) リンク・マップは，PRINTコントロールで指定したファイルに出力されます．

ただし，NOPRINTコントロールを指定した場合は出力されません．

3．3．8 外部定義名リストの出力指定

(1) 機　　能

外部定義名リストを出力するか否かを指定します．

(2) 形　　式

PUBLICS　…外部定義名リストを出力します．

NOPUBLICS…外部定義名リストを出力しません．

(3) 説　　明

外部定義名リストは，PRINTコントロールで指定したファイルに出力されます．

ただし，NOPRINTコントロールを指定した場合は出力されません．

3．3．9 シンボル・リストの出力指定

(1) 機　　能

シンボル・リストを出力するか否かを指定します．

(2) 形　　式

SYMBOLS　…シンボル・リストを出力します．

NOSYMBOLS…シンボル・リストを出力しません．

(3) 説　　明

シンボル・リストは，PRINTコントロールで指定したファイルに出力されます．

ただし，NOPRINTコントロールを指定した場合は出力されません．

3．3．10 日付の指定

(1) 機　　能

出力リストに印字かつ出力ロード・モジュールに登録する日付を指定します．

(2) 形　　式

DATE（日付）

日付：“（”，“）”，“&”を除く１２文字以内の文字列

3．3．11 リスト・ファイルの生成指定

(1) 機　　能

リスト・ファイルを生成するか否かを指定します．

(2) 形　　式

PRINT[（ファイル名）]…リスト・ファイルを生成します．

NOPRINT ……リスト・ファイルを生成しません．

(3) 説　明

- ＰＲＩＮＴで指定するファイル名はリスト・ファイルのファイル名です．また，リスト・ファイルには8.3.7から8.3.9で指定したリストのみが格納されます．
- ＮＯＰＲＩＮＴを指定した場合，8.3.7から8.3.9でリストの出力を指定しても何も生成されません．

第4章　出力形式

本リンカが出力する情報は次のとおりです．

- ロード・モジュール・ファイル
- リスト・ファイル
- エラー・メッセージ

エラー・メッセージは付録を参照してください．

4.1　リスト・ファイル

出力形式は次のとおりです．

```
UPD70108/70116 LINKER   V×.×        日付        PAGE:      ページ
LOAD MODULE FILE：ロード・モジュール・ファイル名（モジュール名）
COMMAND:
入力コマンド
[COMMAND FILE ：パラメータ・ファイル名
 パラメータ・ファイルの内容]  …パラメータ・ファイルを指定したときのみ印字されます．
（警告エラー・メッセージ）
        LINK MAP LIST
**  INPUT FILE/MODULE LIST  **
     ×××      入力ファイル名      （モジュール名）
     ~~~
     モジュール
     番号

**  SEGMENT MAP  **
    START  STOP   SIZE   SEGMENT          TYPE       ALIGN        CLASS
    ×××××  ×××××  ×××××  ××××××××××××××  ×××××××××  ××××××××××××  ××××××××××××××
    ~~~~~  ~~~~~  ~~~~~  ~~~~~~~~~~~~~~  ~~~~~~~~~  ~~~~~~~~~~~~  ~~~~~~~~~~~~~~
    セグメントの先頭アドレス（16進）
           セグメントの終了アドレス（16進）
                  セグメント・サイズ（16進）
                         セグメント名
                                  結合属性
                                             境界属性
                                                          クラス名
```

```
** GROUP MAP **
    GROUP              BASE   SEGMENT
    ××××××××××××××××  ×××××  ×××××××××××××××× •••
                               ×××××××××××××××× •••
```

グループ名　　グループのパラグラフ値（16進）　　セグメント名

```
** START ADDRESS **
    PARAGRAPH=×××× OFFSET=××××
** PUBLIC SYMBOLLIST **
    SYMBOL             MOD ADDRESS     •••
    ××××××××××××××××  ××× ××××:××××  •••
```

シンボル名　　モジュール番号　　アドレス（16進）

アドレスの形式は，パラグラフ値：オフセット値です．

```
** SYMBOL LIST **
MODULE：モジュール名
    SYMBOL             TYPE ADDRESS     •••
    ××××××××××××××××  ×××  ××××:××××  •••
```

シンボル名　　タイプ　　アドレス（16進）

アドレスの形式は，パラグラフ：オフセットです．
シンボルが定数シンボルである場合は，定数値です．
また，外部参照名でアドレス／値が未解決なシンボルの場合は，
−−−−：−−−−と表示されます．

- SYM … ローカル・シンボル名
- PUB … 外部定義名
- EXT … 外部参照名
- LIN … 行番号または文番号
- SPC … ローカル・スコープ・ブロック名（開始）
- SPE … ローカル・スコープ・ブロック名（終了）

```
MODULE：モジュール名
    〈
```

第5章 実 行 例

(1) 入 力 例 ……パラメータ・ファイル使用

```
-RUN LK116 <:F1:LKCLEA.JOB

UPD70108/70116 LINKER V1.0  [30 Oct 85]
   Copyright (C) 1984 NEC Corporation

LINK COMPLETE

-
```

(2) リンク・マップ

```
UPD70108/70116 LINKER V1.0                                          18/03/85        PAGE :    1

LOAD MODULE FILE : :F1:CLEAR.LNK   (CLEAR)

COMMAND :
  <:F1:LKCLEA.JOB

COMMAND FILE : :F1:LKCLEA.JOB

  :F1:FORMAL.:F1:CLEAR.:F1:BDOS.LIB TO :F1:CLEAR
  ORDER(SEGMENTS(CODE,DATA,CONST,STACK)) ADDRESS(CODE(1000H),DATA(10000H))
```

```
UPD70108/70116 LINKER V1.0                                          18/03/85        PAGE :    2

          L I N K   M A P   L I S T

**  INPUT FILE/MODULE LIST  **

    1  :F1:FORMAL.REL  (FORMAL_MAIN)
    2  :F1:CLEAR.REL   (CLEAR_MEMORY)
    3  :F1:BDOS.LIB    (BDOS_INTERFACE0)
    4  :F1:BDOS.LIB    (BDOS_INTERFACE47)

**  SEGMENT MAP  **

  START  STOP   SIZE   SEGMENT          TYPE       ALIGN        CLASS

  01000  01032  00033  CODE             PUBLIC           WORD  CODE
  10000  100FF  00100  DATA             PUBLIC           WORD  DATA
  10100  10100  00000  CONST            PUBLIC           WORD  CONST
  10100  1010F  00010  STACK            STACK            WORD  STACK
  10110  10110  00000  ??SEG            PUBLIC           PARA
  10110  10110  00000  MEMORY           MEMORY           WORD  MEMORY

**  GROUP MAP  **

  GROUP           BASE  SEGMENTS

  CGROUP          0100  CODE
  DGROUP          1000  DATA            CONST           STACK           MEMORY
```

```
UPD70108/70116 LINKER V1.0                                          18/03/85        PAGE :    3

**  PUBLIC SYMBOL LIST  **

  SYMBOL          MOD  ADDRESS    SYMBOL          MOD  ADDRESS    SYMBOL          MOD  ADDRESS    SYMBOL          MOD  ADDRESS

  CHAIN             4  0100:002E  CODE_BASE         1  1000:0003  CODE_LEN          1  1000:0000  COMMANDBUFF       1  1000:0080
  DATA_BASE         1  1000:0009  DATA_LEN          1  1000:0006  EXIT              3  0100:002A  M8080             1  1000:0005
  MAIN              2  0100:000E
```

```
UPD70108/70116 LINKER V1.0                                          18/03/85        PAGE :    4

**  SYMBOL LIST  **

  MODULE : FORMAL_MAIN

    SYMBOL          TYPE  ADDRESS    SYMBOL          TYPE  ADDRESS    SYMBOL          TYPE  ADDRESS    SYMBOL          TYPE  ADDRESS

    CODE_LEN         PUB  1000:0000  CODE_BASE        PUB  1000:0003  M8080            PUB  1000:0005  DATA_LEN         PUB  1000:0006
    DATA_BASE        PUB  1000:0009  COMMANDBUFF      PUB  1000:0080  STACK_BOTTOM     SYM  1010:0010  MEMORY_BOTTOM    SYM  1011:0000
    MAIN             EXT  0100:000E  EXIT             EXT  0100:002A

  MODULE : CLEAR_MEMORY

    SYMBOL          TYPE  ADDRESS    SYMBOL          TYPE  ADDRESS    SYMBOL          TYPE  ADDRESS    SYMBOL          TYPE  ADDRESS

    MAIN             PUB  0100:000E  BLANK            SYM       0020  DATA_LEN         EXT  1000:0006  COMMANDBUFF      EXT  1000:0080
    STACK_BOTTOM     SYM  1010:0010  MEMORY_BOTTOM    SYM  1011:0000  CHAIN            EXT  0100:002E  EXIT             EXT  0100:002A
    CODE_BOTTOM      SYM  0100:002A

  MODULE : BDOS_INTERFACE0

    SYMBOL          TYPE  ADDRESS    SYMBOL          TYPE  ADDRESS    SYMBOL          TYPE  ADDRESS    SYMBOL          TYPE  ADDRESS

    EXIT             PUB  0100:002A  BDOS_ENTRY       SYM       00E0  RESET_CODE       SYM       0000

  MODULE : BDOS_INTERFACE47

    SYMBOL          TYPE  ADDRESS    SYMBOL          TYPE  ADDRESS    SYMBOL          TYPE  ADDRESS    SYMBOL          TYPE  ADDRESS

    CHAIN            PUB  0100:002E  BDOS_ENTRY       SYM       00E0  CHAIN_CODE       SYM       002F
```

付録1　起動コマンドに対するエラー・メッセージ

形式：＊＊＊ ERROR エラー番号　エラー・メッセージ
　　PROGRAM ABORTED

エラー番号 F001	メッセージ	MISSING FILE SPECIFICATION
	原因	入力ファイル名の指定がありません.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	入力ファイル名を指定して再実行してください.
エラー番号 F002	メッセージ	ILLEGAL FILE SPECIFICATION−ファイル名
	原因	ファイル名が正しくありません.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	正しいファイル名を指定して再実行してください.
エラー番号 F003	メッセージ	FILE SPECIFICATION CONFLICTED−ファイル名
	原因	互いに異なるファイル名を指定しなければならないコントロールに同じ名前のファイル名を指定しています.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	正しくファイル名を指定して再実行してください.
エラー番号 F005	メッセージ	FILE NOT FOUND−ファイル名
	原因	入力ファイルが存在しません.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	正しい入力ファイル名を指定して再実行してください.
エラー番号 F006	メッセージ	ILLEGAL FILE−ファイル名
	原因	入力ファイルが，オブジェクト・モジュール・ファイルまたはロード・モジュール・ファイルではありません.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	正しいファイル名を指定して再実行してください.
エラー番号 F007	メッセージ	ILLEGAL OR MISSING PARAMETER：パラメータ
	原因	パラメータが必要なコントロールにパラメータが指定されていないか不当なパラメータが指定されています.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	正しいパラメータを指定して再実行してください.
エラー番号 F008	メッセージ	CONTROL IS NOT RECOGNIZED：文字列
	原因	コントロールとして不当な文字列が指定されています.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	正しいコントロールを指定して再実行してください.
エラー番号 F009	メッセージ	INVALID SYNTAX：X
	原因	起動コマンドに構文上のエラーがあります.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	入力ファイル名，コントロールを正しく指定して再実行してください.

付録２　致命的なファイルI/Oエラーに対するエラー・メッセージ

形式：

ERROR nnn USER PC mmmm

FATAL ERROR nnn － RUN TERMINATED

くわしくはＩＳＩＳ－Ⅱシステム・ユーザーズ・ガイド等を参照ください．

付録３　リンカのエラー・メッセージ

形式：

＊＊＊ ERROR エラー番号　エラー・メッセージ

エラー番号 Ｆ１０１	メッセージ	SEGMENT COMBINATION ERROR(MODULE： モジュール名　SEGMENT：セグメント名）
	原　　因	同名セグメントの結合属性が異っています.
	リンカの処理	コントロールをＯＳへ戻す.
エラー番号 Ｗ１０２	メッセージ	TYPE MISMATCH(MODULE：モジュール名 SYMBOL：シンボル名）
	原　　因	外部参照名の属性が定義と異っています.
	リンカの処理	処理続行.
エラー番号 Ｗ１０３	メッセージ	MULTIPLE DEFINED(MODULE：モジュール名 SYMBOL：シンボル名）
	原　　因	外部定義名が２重定義されています.
	リンカの処理	最初に入力されたモジュールの外部定義名を有効として処理続行.
エラー番号 Ｗ１０４	メッセージ	UNRESOLVED SYMBOL(MODULE：モジュール名 SYMBOL：シンボル名）
	原　　因	外部参照名が定義されていません.
	リンカの処理	外部参照名の値またはアドレスを０００００Hとして処理続行.
エラー番号 Ｆ１０５	メッセージ	SEGMENT OVERFLOW(SEGMENT：セグメント名 CLASS：クラス名）
	原　　因	同名セグメントのサイズが６４Ｋバイトを越えています.
	リンカの処理	ＯＳにコントロールを戻す.
エラー番号 Ｗ１０６	メッセージ	EXTRA START ADDRESS IGNORED （MODULE：モジュール名）
	原　　因	スタート・アドレス指定のあるモジュール（メイン・モジュール）が複数入力されました.
	リンカの処理	最初のメイン・モジュールのスタート・アドレスのみを有効として処理続行.
エラー番号 Ｗ１０７	メッセージ	MORE THAN ONE MEMORY SEGMENT （MODULE：モジュール名　SEGMENT：セグメント名）
	原　　因	MEMORY属性のセグメントが複数あります.
	リンカの処理	最初に入力されたMEMORY属性のセグメントのみを有効とし，他のMEMORY属性のセグメントをCOMMON属性とみなして処理続行.

エラー番号 W108	メッセージ	GROUP DEFINED BY AN EXTERNAL REFERENCE(MODULE：モジュール名)
	原因	グループ名を外部参照名として参照しています.
	リンカの処理	グループのベース・アドレス(パラグラフ・ナンバ)を外部参照名の値として処理続行.
エラー番号 F109	メッセージ	SIZE OF GROUP EXCEEDS 64K(GROUP：グループ名)
	原因	同一グループに属するセグメントが64Kバイトの範囲内に配置されていません.
	リンカの処理	OSにコントロールを戻す.
エラー番号 W110	メッセージ	BOOTSTRAP SPECIFIED FOR MODULE WITHOUT START ADDRESS
	原因	入力モジュールにメイン・モジュールが無いまたはスタート・コントロールが無いのにBOOTSTRAP 指定をしています.
	リンカの処理	BOOTSTRAP 指定を無視して処理続行
エラー番号 F111	メッセージ	PUBLIC SYMBOL NOT FOUND(SYMBOL：シンボル名)
	原因	STARTコントロールで指定した外部定義名が定義されていません.
	リンカの処理	OSにコントロールを戻す.
エラー番号 F112	メッセージ	SPECIFIED SEGMENT IS ABSOLUTE (SEGMENT：セグメント名)
	原因	アブソリュートなセグメントに対してADDRESS コントロールで配置アドレスを指定しています.
	リンカの処理	OSにコントロールを戻す.
エラー番号 W113	メッセージ	PAGE RESIDENT SEGMENT CROSSES PAGE BOUNDARY(SEGMENT：セグメント名)
	原因	INPAGE属性のセグメントのサイズが256バイトを越えています.
	リンカの処理	INPAGE属性を無視して配置する.
エラー番号 F114	メッセージ	SPECIFIED CLASS NOT FOUND IN INPUT MODULE(CLASS：クラス名)
	原因	ADDRESSまたはORDERコントロールで指定したクラス名のセグメントは入力モジュールに存在しません.
	リンカの処理	OSにコントロールを戻す.

エラー番号 F115	メッセージ	SPECIFIED SEGMENT NOT FOUND IN INPUT MODULE(SEGMENT：セグメント名)
	原因	ADDRESSまたはORDERコントロールで指定したセグメント名は入力モジュール内に存在しません.
	リンカの処理	OSにコントロールを戻す.
エラー番号 W116	メッセージ	SEGMENT OVERLAP(SEGMENTS： セグメント名ANDセグメント名)
	原因	セグメント間で配置アドレスの重複があります.
	リンカの処理	重複させたまま処理続行.
エラー番号 F117	メッセージ	MEMORY REQUIRED BY INPUT MODULE EXCEEDS 1 MEGABYTE
	原因	出力モジュールのサイズが1Mバイトを越えています.
	リンカの処理	OSにコントロールを戻す.
エラー番号 W118	メッセージ	MEMORY SEGMENT NOT PLACED HIGHEST IN MEMORY
	原因	MEMORY属性のセグメントのアドレスが他のセグメントより高いアドレスにできません.
	リンカの処理	そのままの配置アドレスで処理続行
エラー番号 F119	メッセージ	SPECIFIED CLASS OUT OF ORDER (CLASS：クラス名)
	原因	クラスの指定にORDERコントロールとADDRESSESコントロールで矛盾があります.
	リンカの処理	OSにコントロールを戻す.
エラー番号 F120	メッセージ	SPECIFIED SEGMENT OUT OF ORDER (SEGMENT：セグメント名)
	原因	セグメントの指定にORDERコントロールとADDRESSESコントロールで矛盾があります.
	リンカの処理	OSにコントロールを戻す.
エラー番号 W121	メッセージ	CAN NOT MAINTAIN SPECIFIED ORDER (SEGMENT：セグメント名)
	原因	ORDERコントロールで指定した順序でセグメントを配置できません.
	リンカの処理	そのままの配置アドレスで処理続行.
エラー番号 F123	メッセージ	ADDRESS FOR CLASS SPECIFIED MORE THAN ONCE(CLASS：クラス名)
	原因	同じクラスに対して複数回ADDRESSESコントロールでアドレスを指定しています.
	リンカの処理	OSにコントロールを戻す.

エラー番号 W209	メッセージ	PARAMETER OVER
	原　　因	パラメータが多過ぎます.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	正しいサブコマンドで再実行してください.
エラー番号 W210	メッセージ	PUBLIC SYMBOL シンボル名 IN ファイル名（モジュール名）
	原　　因	ADDまたはREPLACEサブコマンドで指定したファイル内のモジュールで定義している外部定義シンボルが，既に出力ライブラリ・ファイル内のモジュールに存在しています.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	正しいサブコマンドで再実行してください.
エラー番号 W211	メッセージ	MISSING FILE SPECIFICATION
	原　　因	サブコマンドのパラメータにファイル名の指定がありません.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	正しいサブコマンドで再実行してください.
エラー番号 W212	メッセージ	ILLEGAL FILE SPECIFICATION−ファイル名
	原　　因	サブコマンドのパラメータに不当なファイル名を指定しています.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	正しいサブコマンドで再実行してください.
エラー番号 W218	メッセージ	FILE SPECIFICATION CONFILCTED−ファイル名
	原　　因	サブコマンドのパラメータで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.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	正しいサブコマンドで再実行してください.
エラー番号 W214	メッセージ	ILLEGAL FILE−ファイル名
	原　　因	サブコマンドのパラメータで指定したファイルがOMF，LMFまたはLBFでありません．または書き込み保護されたファイルを出力ファイルとして指定しています.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	正しいサブコマンドで再実行してください.
エラー番号 W215	メッセージ	CHECK SUM ERROR−ファイル名
	原　　因	ファイルにチェック・サム・エラーがありました.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	チェック・サム・エラーのあったファイルを再度作成して，再実行してください.
エラー番号 W216	メッセージ	EXIT SUBCOMMAND NOT FOUND−ファイル名
	原　　因	起動時に指定したパラメータ・ファイルにEXITサブコマンドがありません.
	プログラムの処理	パラメータ・ファイルの最後にEXITサブコマンドがあったものとして処理する．（制御をOSに戻す．）
	ユーザの処置	パラメータ・ファイルにEXITサブコマンドを入れてください.

エラー番号 A901	メッセージ	WORKING TABLE SPACE EXHAUSTED
	原　　因	作業用ワーク・エリア（メモリ）が足りません.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	なし
エラー番号 A902	メッセージ	INVALID FILE SYNTAX －ファイル名
	原　　因	ファイルの形式に誤りがあります.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	正しいファイル名で再実行してください.
エラー番号 A999	メッセージ	PROGRAM ERROR ：エラー・コード
	原　　因	プログラムのバグを検出しました.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	なし. プログラム名，バージョンおよびエラー・コードを弊社にご連絡ください.

付録３　致命的なファイルI/Oエラーに対するエラー・メッセージ

形式：

ERROR nnn USER PC mmmm

FATAL ERROR nnn － RUN TERMINATED

くわしくはISIS－Ⅱシステム・ユーザーズ・ガイド等を参照ください。

付録4　ライブラリアンのエラー・メッセージ

エラー番号 W201	メッセージ	INVALID COMMAND
	原　　因	サブコマンド名が誤っています.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	正しいサブコマンドで再実行してください.
エラー番号 W202	メッセージ	INVALID SYNTAX
	原　　因	サブコマンドのパラメータ指定が誤っています.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	正しいサブコマンドで再実行してください.
エラー番号 W203	メッセージ	FILE NOT FOUND
	原　　因	指定ファイルが存在しません.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	正しいサブコマンドで再実行してください.
エラー番号 W204	メッセージ	MODULE NOT FOUND−ファイル名（モジュール名）
	原　　因	指定モジュールが存在しません.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	正しいサブコマンドで再実行してください.
エラー番号 W205	メッセージ	NOT LIBRARY FILE−ファイル名
	原　　因	ＬＢＦとして指定したファイルがＬＢＦでありません.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処理	正しいサブコマンドで再実行してください.
エラー番号 W206	メッセージ	NOT OBJECT FILE−ファイル名
	原　　因	ＯＭＦとして指定したファイルがＯＭＦでありません.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	正しいサブコマンドで再実行してください.
エラー番号 W207	メッセージ	FILE ALREADY EXISTS−ファイル名
	原　　因	CREATEコマンドで指定したファイルが既に存在しています.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	正しいサブコマンドで再実行してください.
エラー番号 W208	メッセージ	MODULE ALREADY EXISTS−ファイル名（モジュール名）
	原　　因	出力ＬＢＦ内に，入力ファイル内のモジュール名と同名のモジュールが存在します.
	プログラムの処理	サブコマンドを無視し，次のサブコマンド入力を待つ.
	ユーザの処置	正しいサブコマンドで再実行してください.

付録1　サブコマンド一覧

項番	サブコマンド名	短縮形	オペランド形式	備考
1	CREATE	C	LBF名	LBF：ライブラリ ファイル
2	ADD	A	{OMF名 / LBF名 [((OM名[,…]))]} [,…] TO LBF名	OMF：OMファイル LBF：ライブラリ・ファイル
3	DELETE	D	LBF名 (OM名[,…])	LBF：ライブラリ・ファイル
4	REPLACE	R	{OMF名 / LBF名[((OM名[,…]))]} [,…] FROM LBF名	OMF：OMファイル LBF：ライブラリ・ファイル
5	LIST	L	LBF名 [((OM名[,…]))] [,…] [TO LSF名] [{PUBLICS / PUB / PL}]	LSF：リスト・ファイル LBF：ライブラリ・ファイル
6	EXIT	E	なし	

付録２　起動コマンドに対するエラー・メッセージ

形式：＊＊＊ ERROR エラー番号　エラー・メッセージ
　　PROGRAM ABORTED

エラー番号 F001	メッセージ	MISSING FILE SPECIFICATION
	原　　因	パラメータ・ファイル名の指定がありません.
	プログラムの処理	ＯＳにコントロールを戻す.
	ユーザの処置	パラメータ・ファイル名を指定して再実行してください.
エラー番号 F002	メッセージ	ILLEGAL FILE SPECIFICATION－ファイル名
	原　　因	ファイル名が正しくありません.
	プログラムの処理	ＯＳにコントロールを戻す.
	ユーザの処置	正しいファイル名を指定して再実行してください.
エラー番号 F005	メッセージ	FILE NOT FOUND －ファイル名
	原　　因	パラメータ・ファイルが存在しません.
	プログラムの処理	ＯＳにコントロールを戻す.
	ユーザの処置	正しいパラメータ・ファイル名を指定して再実行してください.
エラー番号 F007	メッセージ	ILLEGAL OR MISSING PARAMETER ：パラメータ
	原　　因	パラメータが必要なコントロールにパラメータが指定されていないか不当なパラメータが指定されています.
	プログラムの処理	ＯＳにコントロールを戻す.
	ユーザの処置	正しいパラメータを指定して再実行してください.
エラー番号 F008	メッセージ	CONTROL IS NOT RECOGNIZED：文字列
	原　　因	コントロールとして不当な文字列が指定されています.
	プログラムの処理	ＯＳにコントロールを戻す.
	ユーザの処置	正しいコントロールを指定して再実行してください.
エラー番号 F009	メッセージ	INVALID SYNTAX ： X
	原　　因	起動コマンドに構文上のエラーがあります.
	プログラムの処理	ＯＳにコントロールを戻す.
	ユーザの処置	コマンド・ファイル名，コントロールを正しく指定して再実行してください.

4．1　ライブラリ・ファイル情報リスト

(1)　出力形式

```
    UPD70108/70116 LIBRARIAN V×.×  DATE(  ①  )  PAGE:  ②
    LIB-FILE NAME:  ③                (  ④  )
⑤   nnnn  MM … M  (  ⑥  )
        UPDATE:⑩  PP…P  ⑦  UPD70108/116
                                                    注1
            SS  …  S
                .
                .
                .
                ⑧
                    NUMBER OF PUBLIC SYMBOLS    :  ⑨
.                   .
.                   .
.                   .
```

注1　　点線の中は，ＰＵＢＬＩＣＳの指定がある場合にだけプリントされます．

(2)　出力項目

リスト出力項目の内容を表4－1に示します．

表4－1　出　力　項　目

項番〔注1〕	内　　容
①	日付（起動時のDATEコントロールでの指定値）
②	出力リストのページ番号（１０進表示）
③	ライブラリ・ファイル名
④	ライブラリ・ファイルの作成日付および更新日付
⑤	モジュール通番　　（１０進表示）
⑥	モジュール作成日付（アセンブル／リンク処理日）
⑦	モジュールを作成したソフトウェア
⑧	ＰＵＢＬＩＣシンボル名
⑨	モジュール内で定義されているＰＵＢＬＩＣシンボルの総数
⑩	更新回数（１０進表示）

第5章 実行例

実行例

```
-RUN LB116 <:F1:LBBDOS.JOB

UPD70108/70116 LIBRARIAN V1.0  [30 Oct 85]
   Copyright (C) 1984 NEC Corporation

*CREATE :F1:BDOS
*ADD    :F1:BDOS0.:F1:BDOS47 TO :F1:BDOS
*LIST   :F1:BDOS TO :F1:BDOS PUBLICS
*EXIT

-
```

BDOS.PRNの内容（ライブラリ・ファイル情報リスト）

```
UPD70108/70116 LIBRARIAN V1.0     DATE(18/03/85          )      PAGE :     1

    LIB-FILE NAME : :F1:BDOS.LIB    (18/03/85          18/03/85          )

    1     BDOS_INTERFACE      (18/03/85                              )
          0
          UPDATE :     0       ASSEMBLER      UPD70108/116

          EXIT

               NUMBER OF PUBLIC SYMBOLS :     1

    2     BDOS_INTERFACE      (18/03/85                              )
          47
          UPDATE :     0       ASSEMBLER      UPD70108/116

          CHAIN

               NUMBER OF PUBLIC SYMBOLS :     1


               NUMBER OF MODULES :     2
```

3．4．6 EXIT

(1) 機　能

ライブラリアンを終了させます．

(2) 形　式

```
*{EXIT}↵
 {E   }
```

(3) 使 用 例

*EXIT↵

3．5 入出力ファイル

(1) ファイルの種類

ライブラリアンへの入出力ファイルには，次の4種類があります．

- 入力オブジェクト　　（OMF，LBF）
- 更新／出力ライブラリ・ファイル（LBF）
- リスト・ファイル　　（PRF）

(2) 入出力種別と媒体

各ファイルに対する入出力種別および使用可能な媒体を表3－1に示します．

表3－1　入出力種別と媒体

項番	ファイル	入出力	媒　体
1	入力OMF	入力	FD
2	入力LMF	入力	FD
3	入力LBF	入力	FD
4	更新／出力 LBF	入出力	FD
5	PRF	出力	FD／LP／コンソール

(3) 入力オブジェクト・ファイル

編集対象とするモジュール（OM）を格納しているファイルです．

- オブジェクト・モジュール・ファイル（OMF）

  アセンブラが作成したオブジェクト・モジュール・ファイルです．

- ライブラリ・ファイル（LBF）

  ライブラリアンが作成したファイルです．ファイル内の特定のモジュール，または全モジュールを入力することができます．

第4章　出力リスト

ライブラリアンは，ＬＩＳＴサブコマンドによりライブラリ・ファイル情報リストを出力します．

(1) リストの出力先

ＬＩＳＴサブコマンドを参照してください．

(2) 出力リストの頁フォーマット

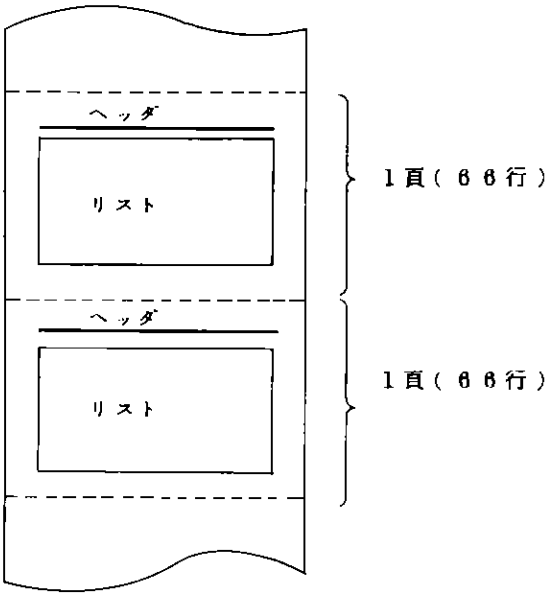
出力リストは，次のような頁編集をしています．

- １頁は６６行
- 頁の先頭には次のヘッダを出力

＜形　式＞

UPD70108／70116 LIBRARIAN V×.× DATE(日付) PAGE：ページ

＜頁フォーマット＞



すべてのモジュールが対象となります．

・更新ＬＢＦ名

追加される側のＬＢＦ名です．

(3) 注意事項

・追加するモジュール名と同名のモジュールが更新ＬＢＦの中に存在してはいけません．

(4) 使用例

```
*ADD F8, XLIB(MX) TO CLIB↵
```

3．4．3 DELETE

(1) 機　能

既存のライブラリ・ファイル内のモジュール（ＯＭ）を削除します．

(2) 形　式

```
*{DELETE / D} ＬＢＦ名（モジュール名〔，…〕）↵
```

・ＬＢＦ名

削除するモジュールを格納しているライブラリ・ファイルの名前です．

・モジュール名（ＯＭ名）

削除対象のモジュール名です．

(3) 注意事項

指定したモジュールは，ＬＢＦ内に存在しなければいけません．

(4) 使用例

```
*DELETE CLIB(MX)↵
```

3．4．4 REPLACE

(1) 機　能

既存のライブラリ・ファイル内のモジュールを，他のオブジェクト・モジュール・ファイルまたはライブラリ・ファイル内のモジュールで置換します．

(2) 形　式

```
*{REPLACE / R} {OMF名 / LBF名〔（モジュール名〔，…〕）〕} FROM 更新LBF名↵
```

○ ＯＭＦ名／ＬＢＦ名

置換するモジュールを格納しているファイルの名前です．

＜省略時解釈＞拡張子が省略されるとＲＥＬとみなします．

○ モジュール名

入力ファイルがＬＢＦの場合，置換するモジュールの名前です．モジュール名を省略すると，ＬＢＦ内のすべてのモジュールが置換対象となります．

○ 更新ＬＢＦ名

置換される側のＬＢＦ名です．

(3) 注意事項

○ 置換するモジュール名と同名のモジュールが更新ＬＢＦの中に存在しなければなりません．

○ 入力ＬＢＦ名（ＦＲＯＭの左側）と更新ＬＢＦ名は異なっていなければなりません．

(4) 使用例

```
*REPLACE SUB1,SXSUB(SUB2,SUB8) FROM SUBLIB )
```

3.4.5 LIST

(1) 機　能

ライブラリ・ファイル内のモジュール情報をリスト・ファイルへ出力します．

(2) 形　式

```
* { LIST } LBF名 [ (モジュール名 [ ,… ]) ] [ ,… ] [ TO リスト・ファイル名 ] [ { PUBLICS } ] )
  { L    }                                                                  { PUB     }
                                                                            { PL      }
```

・ＬＢＦ名

プリント対象のライブラリ・ファイルの名前です．

・モジュール名（ＯＭ名）

特定のモジュールの情報だけをプリントする場合に，該当モジュールの名前を指定します．モジュール名を省略した場合は，ＬＢＦ内のすべてのモジュールがプリントの対象となります．

・リスト・ファイル名

プリント情報を出力するファイルの名前です．

・ＰＵＢＬＩＣＳ

モジュール内で定義されているＰＵＢＬＩＣシンボル情報のプリントを指示します．

(3) 注意事項

・リスト・ファイル名を省略した場合は，コンソールへプリントされます．

・リスト出力形式については，第４章を参照してください．

(4) 使用例

```
*LIST CLIB TO :LP: )
```

←--------ＣＬＩＢ内の全モジュール情報をＬＰへ出力

```
*LIST CLIB(M1,M2) PUBLICS )
```

←----ＣＬＩＢ内のモジュールＭ１とＭ２の情報をコンソールへ出力ＰＵＢＬＩＣシンボル情報も出力

(b) フロッピィ・ディスク以外の場合

```
装置名：
```

装置名には次のものが指定できます．

{ CO … コンソール
{ LP … ラインプリンタ

例

① ：LP：……リストをLPに出力

3.2 プログラムの終了

プログラムは実行を終了すると，次のメッセージをコンソールに出力し，制御をOSに戻します．

・正常終了の場合

メッセージは出力されません．

・処理中に異常が発生し，処理を中止する場合

```
PROGRAM ABORTED
```

3.3 コントロール

コントロールには，次のものがあります．

(a) DATEコントロール

処理日付を指定するコントロールです．

＜形 式＞

```
DATE(日付)
```

または

```
DA(日付)
```

日付：12文字以内の文字列
（“(”，“)”，“&”を除く）

＜補 足＞

・この日付は，出力リストのヘッダに印字されます．更に，出力LBF内にも処理日付として残ります．

・DATEコントロールを省略した場合は，次のように解釈されます．

```
DATE(ISIS-ⅡのDATEコマンドで最後に指定した日付)
```

3.4 サブコマンド

(1) サブコマンドの種類

ライブラリアンのサブコマンドには，次の6種類があります．

- CREATE　サブコマンド
- ADD　サブコマンド
- DELETE　サブコマンド
- REPLACE　サブコマンド
- LIST　サブコマンド
- EXIT　サブコマンド

（サブコマンド名およびオペランド形式の一覧については，付録1の付1－1および付1－2を参照してください．）

3.4.1 CREATE

(1) 機　能

ライブラリ・ファイルを新規に生成します．

(2) 形　式

```
*{CREATE} LBF名 )
 {C     }
```

(3) 注意事項

・生成するライブラリ・ファイルの名前をオペランドに指定してください．

(4) 使用例

```
*CREATE   :F1:COMLIB )
```

3.4.2 ADD

(1) 機　能

既存のライブラリ・ファイルに対して，別のファイル（OMFまたはLBF）内のモジュール（OM）を追加します．

(2) 形　式

```
*{ADD} {OMF名                          } [, …] TO 更新LBF名)
 {A  } {LBF名[ ( モジュール名[, …] ) ] }
```

・OMF名／LBF名

追加するモジュールが格納されているファイルの名前です．

＜省略時解釈＞拡張子が省略されるとRELとみなします．

・モジュール名（OM名）

入力ファイルがLBFの場合，追加するモジュールの名前です．省略時は，LBF内の

3．1．2 会話形式でのサブコマンドの入力方法

起動時にコマンド・ファイルを指定しない場合には，実行開始メッセージが表示された後にサブコマンドの入力要求記号“＊”が表示され，サブコマンドの入力待ちとなります．

“＊”に続いて，ライブラリアンのサブコマンドを指定します．

```
＊サブコマンド名　オペランド情報↵
```

サブコマンドの入力が終了すると，各サブコマンド処理が開始されます．

一つのサブコマンドの処理が完了すると再び“＊”が印字され，次のサブコマンド入力待ち状態となります．終了サブコマンド（ＥＸＩＴサブコマンド）が入力されるまで，この動作がくり返されます．

```
＊　サブコマンド指定↵
    ┌ ─ ─ ─ ─ ─ ─ ┐
    │ サブコマンド処理 │
    └ ─ ─ ─ ─ ─ ─ ┘

＊　サブコマンド指定↵
    ┌ ─ ─ ─ ─ ─ ─ ┐
    │ サブコマンド処理 │
    └ ─ ─ ─ ─ ─ ─ ┘
        ⋮
＊　終了サブコマンド指定↵

    ライブラリアンの終了
```

１行で指定可能な文字数は８０字（“↵”を含む）までです．

なお，１行で必要なオペランド情報をすべて入力することができない場合は，継続行の指定ができます．

```
                          ---継続行の指示
＊サブコマンド名　オペランド情報&↵
－オペランド情報↵
＊
```

行の最後に“＆↵”を指定することで，オペランド入力要求記号“－”が次の行に印字され，前の行にひき続いてオペランド情報を入力できます．

〔例〕

```
*CREATE   LBF )
*ADD      F1, F2, F8, F4 & )
-TO LBF )                    <……継続行
          :
          :
*LIST     LBF TO LST: PUBLICS )
*EXIT
```

(1) ファイル名規則

(a) フロッピィ・ディスク上のファイルの場合

〔ドライブ名：〕名称〔．拡張子〕

ドライブ名：ファイルを格納しているFDまたはHDがセットされているドライブ名（F0，F1…）

名　称：6文字以内の文字列

拡張子：3文字以内の文字列

＜省略時解釈＞

省略時解釈はファイルの種類毎に異なり，表3－2のとおりとなります。

表3－2　ドライブ名およびファイル・タイプの省略時解釈

項番	ファイルの種類	ドライブ名	ファイル・タイプ
1	オブジェクト・モジュール・ファイル	F0	REL
2	ライブラリ・ファイル		LIB
3	ロード・モジュール・ファイル		LNK
4	リスト・ファイル		PRN

例

① ：F1：MDMAIN．LIB …… ドライブ名がF1，名称がMDMAIN，拡張子がLIB

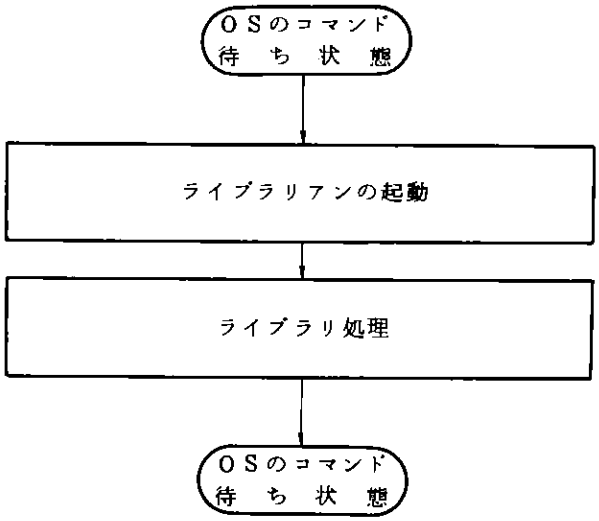
② SUB1 …… ドライブ名と拡張子がLIB

第3章　操 作 法

3.1　操作手順

ライブラリアンの操作手順を図３－１に示します．

図３－１　操作手順



次の指定によりライブラリアンを起動します．

```
—RUN[：d：]LB116[<パラメータ・ファイル名>][　コントロール]↵
```

d：LB116.86を格納しているＦＤまたは
ＨＤドライブ名
（省略時はＦ０とみなします．）

LB116：ライブラリアンのプログラム・ファイル名

パラメータ・ファイル名：ライブラリアンへのコマンドを格納しているファイル（パラメータ・ファイル）のファイル名

ライブラリアンが起動されるとコンソールに実行開始のメッセージが表示され，ライブラリ処理が開始されます．

```
UPD70108/70116 LIBRARIAN  V×.×  〔×× ××× ××〕
    Copyright (C) 1984 NEC Corporation
```
←実行開始メッセージ

起動時のオペランドにパラメータ・ファイルを指定した場合は，サブコマンドをパラメータ・ファイルから読込んで処理を行ない，パラメータ・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ＯＳコマンド待ちになります．

パラメータ・ファイルを指定しない場合はキーボードからサブコマンドを読込み，会話形式で処理を行ないます．

３．１．１ パラメータ・ファイルの形式

次の形式のファイルをエディタで作成してください．

```
サブコマンド名  オペランド情報 )
サブコマンド名  オペランド情報 )
      )
      (
EXIT )
```

- １行は１３２文字以内にしてください．
- 一つのサブコマンドが１行におさまらない場合は，行の最後に“ &)”（継続行の指定）を入力して複数行に分けて入力してください．
- 一つのサブコマンドの最後の行には，“ &)”は入力しないでください．
- セミコロン（ ; ）から行の終り迄はライブラリアンへのコマンドとしては解釈されません．（コメント．）
- 空白のみの行も記述できます．
- 英小文字は英大文字として解釈されます．（ただし，モジュール名を除きます．）
- パラメータ・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，自動的にＥＸＩＴサブコマンドがあったものと解釈されます．

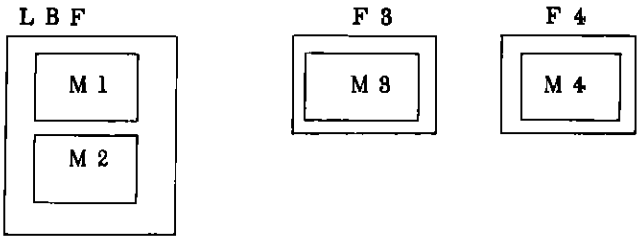
例

```
;
; library creation command
;
  CREATE SUBLIB
;
  ADD SUB1,SUB2,SUB3 &                    ←継続行
  SUB4 SUB5 TO SUBLIB
;
  LIST SUBLIB TO LIST PUBLICS
  EXIT
```

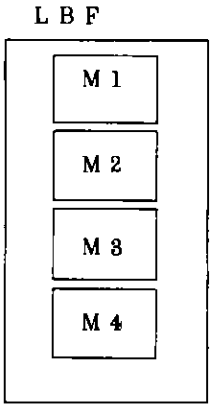
例(2) 追加登録

モジュールＭ３とＭ４をＬＢＦへ登録

＜登録前＞



＜登録後＞



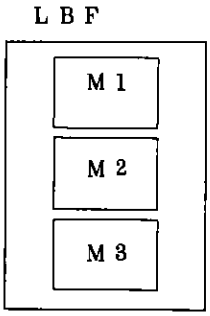
２・２・２ モジュール（ＯＭ）の削除

既存のＬＢＦ（モジュールが登録済）内のモジュールを削除します.

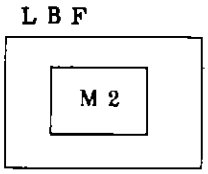
ＬＢＦ内のモジュールの削除にはＤＥＬＥＴＥサブコマンドを用います.

例 ＬＢＦ内のモジュールＭ１とＭ３を削除

＜削除前＞



＜削除後＞



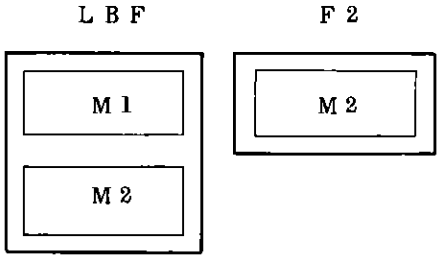
2．2．3 モジュール（OM）の置換

既存のＬＢＦ（モジュールが登録済）内のモジュールを，他のＯＭＦまたはＬＢＦの中のモジュールで置換します．

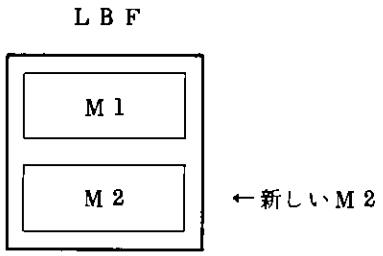
ＬＢＦ内のモジュールの置換にはREPLACEサブコマンドを使います．

例　ＬＢＦ内のモジュールＭ２を置換

＜置換前＞



＜置換後＞



2．3　ライブラリ・ファイル(LBF)情報のプリント

既存のＬＢＦ内に登録されているモジュールの情報を編集し，リスト・ファイルへ出力します．

出力情報には，次のものがあります．

＜ライブラリ情報＞

- 作成日付，更新日付
- 登録モジュール数

＜モジュール情報＞

- モジュール（ＯＭ）名
- 作成プログラム名
- 登録日付，更新日付
- 更新回数
- モジュール内で定義されているＰＵＢＬＩＣシンボル 注(1)
- ＰＵＢＬＩＣシンボルの個数

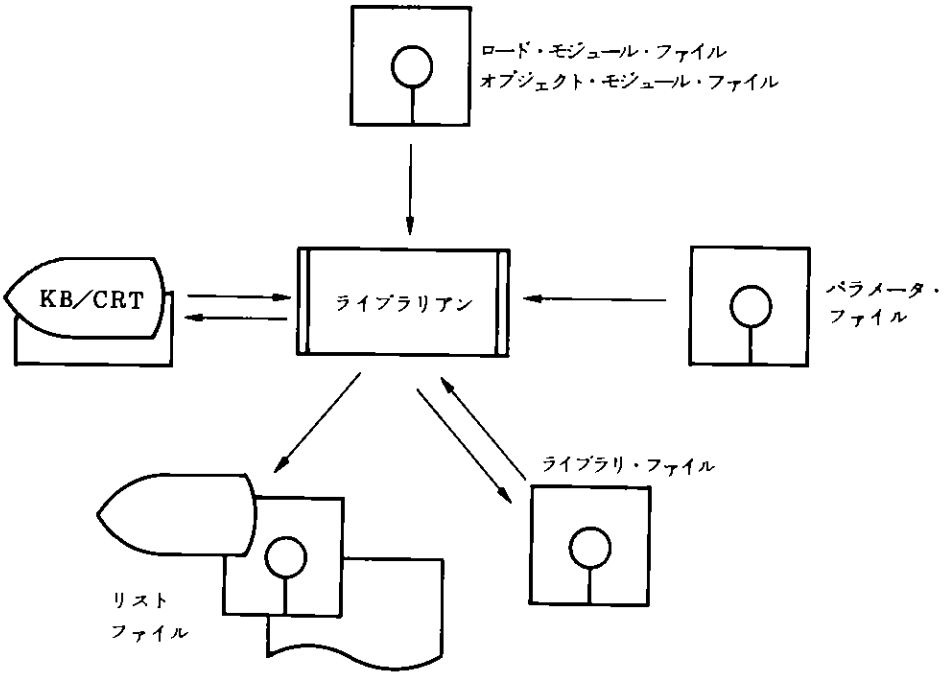
注(1)　ＰＵＢＬＩＣ疑似命令（アセンブラ）で宣言されたシンボルです．

ＬＩＳＴサブコマンドによりプリントします．（出力形式については，第３章を参照してください．）

1．2　システム構成

システムの構成図を図1－2に示します．

図1－2　システム構成



第２章　機　　能

２．１　ライブラリ・ファイル(LBF)の生成

ＬＢＦを初期化します．なお，初期生成されたＬＢＦの中にはモジュール（OM／LM）は存在しません．

ＬＢＦの初期化には，CREATEサブコマンドを用います．

２．２　ライブラリ・ファイル(LBF)の更新

既存のＬＢＦに対して，モジュールの追加，削除または置換を行ないます．

２・２・１　モジュール（ＯＭ）の追加

既存のＬＢＦに，他のＯＭＦあるいはＬＢＦの中のモジュール（ＯＭ）を新規（または追加）登録します．

＜新規登録＞

初期生成直後のＬＢＦへのモジュール追加

＜追加登録＞

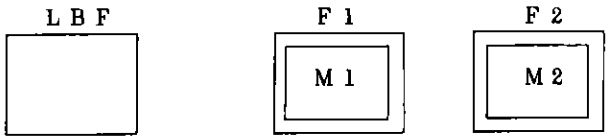
既にモジュールが登録されているＬＢＦへのモジュール追加

ＬＢＦへのモジュールの追加にはＡＤＤサブコマンドを用います．

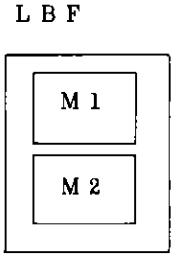
例(1)　新規登録

モジュールＭ１とＭ２をＬＢＦへ登録

＜登録前＞



＜登録後＞



第１章　ソフトウエア概要

１．１　ライブラリアンの機能概要

ライブラリアンは，本パッケージで扱っているオブジェクト・モジュール（ＯＭ）ファイル（OMF）とロード・モジュール（ＬＭ）・ファイル（LMF）に対してモジュール単位で編集を行なうものです．

(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．OM（LM）の数が多い場合，ファイルの数も増加します．そこで，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（ＬＢＦ）と呼びます．

ライブラリ・ファイルをリンカに入力することもできます．したがって，モジュラ・プログラミングを行なった場合共通的なモジュールをＬＢＦ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（ＬＢＦ）に対して，次に示すような編集機能があります．

- ＬＢＦへのモジュールの追加
- ＬＢＦ内のモジュールの削除
- ＬＢＦ内のモジュールの置換

（機能の詳細については，第２章2.2を参照してください．）

(3) ライブラリ・ファイル情報のプリント

ライブラリアンには，ライブラリ・ファイル（ＬＢＦ）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プリントする機能があります．

- モジュールの名前，作成プログラム，登録日付，更新日付
- PUBLICシンボル情報

（機能の詳細については，第２章2.3を参照してください．）

(4) ライブラリアンの操作

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．

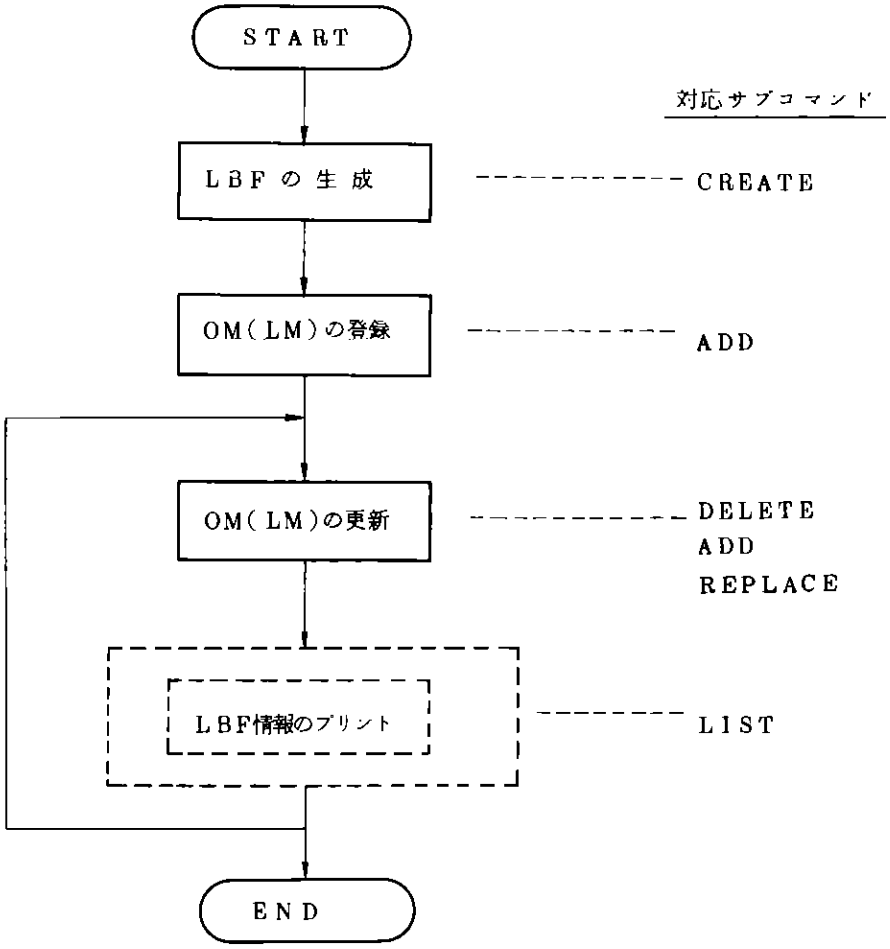
ライブラリアンは，各サブコマンドを順次解釈しながら処理を行ないます．

（操作法の説明については第３章を参照してください．）

- ライブラリ・ファイルの作成手順

一般的なＬＢＦ作成手順を，図１－１に示します．

図１－１　ＬＢＦの作成手順



付録1　起動コマンドに対するエラー・メッセージ

形式：＊＊＊ ERROR エラー番号　エラー・メッセージ
　　PROGRAM ABORTED

エラー番号 F001	メッセージ	MISSING FILE SPECIFICATION
	原　　因	入力ファイル名の指定がありません.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	入力ファイル名を指定して再実行してください.
エラー番号 F002	メッセージ	ILLEGAL FILE SPECIFICATION－ファイル名
	原　　因	ファイル名が正しくありません.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	正しいファイル名を指定して再実行してください.
エラー番号 F003	メッセージ	FILE SPECIFICATION CONFLICTED－ファイル名
	原　　因	互いに異なるファイル名を指定しなければならないのに同じ名前のファイル名を指定しています.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	正しくファイル名を指定して再実行してください.
エラー番号 F005	メッセージ	FILE NOT FOUND －ファイル名
	原　　因	入力ファイルが存在しません.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	正しい入力ファイル名を指定して再実行してください.
エラー番号 F006	メッセージ	ILLEGAL FILE－ファイル名
	原　　因	入力ファイルがオブジェクト・モジュールまたはロード・モジュール・ファイルでありません.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	正しい入力ファイル名を指定して再実行してください.
エラー番号 F008	メッセージ	CONTROL IS NOT RECOGNIZED：文字列
	原　　因	コントロールとして不当な文字列が指定されています.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	正しいコントロールを指定して再実行してください.
エラー番号 F009	メッセージ	INVALID SYNTAX ： X
	原　　因	起動コマンドに構文上のエラーがあります.
	プログラムの処理	OSにコントロールを戻す.
	ユーザの処置	入力ファイル名，コントロールを正しく指定して再実行してください.

付録２　致命的なファイルI/Oに対するエラー・メッセージ

形式：

ERROR nnn USER PC mmmm

FATAL ERROR nnn － RUN TERMINATED

くわしくはＩＳＩＳ－Ⅱシステム・ユーザーズ・ガイド等を参照ください．

(b) データ・レコード（ＤＲ）

```
: ×× ×××× 00 DD … DD SS CR LF
① ②  ③   ④    ⑤    ⑥
```

①……レコード・マーク

②……コード数

③……ロケーション・アドレス（オフセット）

④……レコード・タイプ．「００」固定です．

⑤……コード

⑥……チェック・サム

(c) スタート・アドレス・レコード（ＳＡＲ）

アセンブラ・ソース・プログラムのＥＮＤ疑似命令でオペランドにラベルを指定した場合，あるいは，リンク時にＳＴＡＲＴコントロールの指定がある場合にのみ作成されるレコードです．プログラムの実行開始アドレスを示すレコードです．

```
: 04 0000 03 ×××× YYYY SS CR LF
① ②  ③   ④   ⑤    ⑥   ⑦
```

①……レコード・マーク

②……コード数．「０４」固定です．

③……「００００」固定です．

④……レコード・タイプ．「０８」固定です．

⑤……実行開始アドレスのパラグラフ番号を１６進４桁で示します．

⑥……実行開始アドレスのオフセット値を１６進４桁で示します．

⑦……チェック・サム

(d) 最終レコード（EOFR）

```
: 00 0000 01 FF CR LF
① ②   ③   ④  ⑤
```

①……レコード・マーク

②……コード数．「００」固定です．

③……「００００」固定です．

④……レコード・タイプ．「０１」固定です．

⑤……チェック・サム．「ＦＦ」固定です．

オブジェクト・コードの終了を示すレコードです．

備考 各レコード共，パリティなし７ビットＡＳＣＩＩコードで出力します．

第4章 実 行 例

(1) 入 力 例

```
-RUN OC116 :F1:CLEAR TO CLEAR

UPD70108/70116 OBJECT CONVERTER V1.0  [30 Oct 85]
   Copyright (C) 1984 NEC Corporation

OBJECT CONVERSION COMPLETE

-
```

(2) ＣＬＥＡＲ．Ｈ８６の内容

```
:020000020100FB
:0D0000008CD88ED0BC1001E80400E81D0073
:10000E008B0E0600BF10012BCF8CD88EC033C0F3E1
:0C001E00AAC606800020E80700E80000E9
:04002A00B100CDE074
:05002E00B12FCDE0C37D
:00000001FF
```

3.4 入出力ファイル

(1) ファイルの種類

HEX形式オブジェクト変換プログラムへの入出力ファイルには，次の種類があります．

- オブジェクト／ロード・モジュール・ファイル（OMF，LMF）
- 拡張HEX形式オブジェクト・モジュール・ファイル（H86）

(2) 入出力種別と媒体

各ファイルに対する入出力種別および使用可能な媒体を表3－1に示します．

表3－1 入出力種別と媒体

項番	ファイル	入出力	媒体
1	入力OMF	入力	FD／HD
2	入力LMF	入力	FD／HD
3	H86	出力	FD／HD

(3) オブジェクト／ロード・モジュール

拡張HEX形式に変換すべきオブジェクト・モジュールを格納しているファイルです．

- オブジェクト・モジュール・ファイル（OMF）

アセンブラが作成したオブジェクト・モジュール・ファイルです．

- ロード・モジュール・ファイル（LMF）

リンカが作成したロード・モジュール・ファイルです．

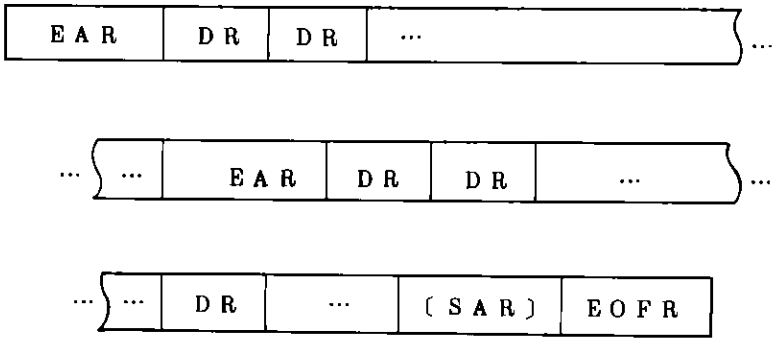
オブジェクト／ロード・モジュール・ファイルは，外部参照アドレスがすべて解決済みでなければいけません．

(4) 拡張HEX形式オブジェクト・モジュール・ファイル（H86）

拡張HEX形式に変換します．

拡張HEX形式の構成は，図3－2のとおりです．

図3－2 拡張HEX形式



〔　〕：存在しない場合があることを示します．

(a) 拡張アドレス・レコード（ＥＡＲ）

：０２ ０００００ ０２ ××××× ＳＳ (CR) (LF)
① ② ③ ④ ⑤ ⑥

①……レコード・マーク『：』

②……コード数『０２』固定です．

③……『０００００』固定です．

④……レコード・タイプ『０２』固定です．

⑤……セグメントのパラグラフ値を１６進４桁で示します．

⑥……チェック・サム注(1)

ＥＡＲは，セグメントの先頭および１６個の連続したＤＲに対して作成します．

注(1) チェック・サム……コロン，C_R，L_Fを除くすべてのデータを初期値０から順に減算した値が入ります．この値は１６進数２桁で示されます．

：１００７８０００８ＣＡＦＣＤ８Ａ０７……ＦＦ８Ｂ (CR) (LF)

チェック・サムの算出方法

８ビットの２進減算器を仮定してボローを無視した減算を行ないます．上記のデータ例ではまず初期値００Ｈからコード数の１０Ｈを減算します．

```
  0000 0000
- 0001 0000
  1111 0000
```

次にロケーション・アドレスの上位８ビット０７Ｈをこの結果から減算します．

```
  1111 0000
- 0000 0111
  1110 1001
```

更にロケーション・アドレスの下位８ビット８０Ｈを減算します．

```
  1110 1001
- 1000 0000
  0110 1001
```

最後のデータＦＦまでこの減算方法で繰返します．

```
  0110 1001
- 0011 1100   ……３ＣＨ
  0010 1101
- 1010 1111   ……０ＡＦＨ
  0111 1110
       ・
       ・
       ・
- 1111 1111   ……０ＦＦＨ
  1000 1011
```

チェック・サム　８ＢＨ

・ＨＥＸ形式オブジェクト変換プログラムが起動されるとコンソールに実行開始のメッセージが表示され，変換処理が開始されます．

```
UPD70108/70116 OBJECT CONVERTER Vx.x 〔xx xxx xx〕 ←――実行開始メッセージ
    Copyright (C) 1984 NEC Corporation
```

(1) ファイル名規則

```
〔ドライブ名：〕名称〔．拡張子〕
```

〔　〕：省略可能を示します．
ドライブ名：ファイルを格納しているＦＤがセットされているドライブ名（Ｆ０｜Ｆ１…）
名　称：６文字以内の文字列
拡張子：３文字以内の文字列

＜省略時解釈＞

省略時解釈はファイルの種類毎に異なり，表３－２のとおりとなります．

表３－２

項番	ファイルの種類	ドライブ名	ファイル・タイプ
1	入力ファイル	Ｆ０	ＬＮＫ
2	出力ファイル		Ｈ８６

例

① ：F1:MDMAIN.H86 ……ドライブ名がＦ１，名称がMDMAIN，拡張子がＨ８６

② SUB1 ……ドライブ名と拡張子を省略

３．２　プログラムの終了

プログラムは実行を終了すると，次のメッセージをコンソールに出力し，制御をＯＳに戻します．

・正常終了の場合

```
OBJECT CONVERSION COMPLETE
```

・処理中に異常が発生し，処理を中止する場合

```
PROGRAM ABORTED
```

3.3 コントロール

コントロールとは，プログラムの実行に先立ってプログラムの動作に必要な情報を指示するためのものです．

ＨＥＸ形式オブジェクト変換プログラムの起動時にコンソールから入力してください．

・コントロールを複数回入力した場合は，後で入力したコントロールが，有効となります．

コントロールの名称，短縮形および省略時の解釈についての一覧を表６－２に示します．

表６－２

項番	コントロール名	短縮形	省略時
1	ORDER	OD，ORD	ORDER
	NOORDER	NOOD，NOORD	

(a) アドレス順序指定用コントロール

拡張ＨＥＸ形式オブジェクト・モジュール内のアドレス順序を指定します．

＜形式＞

・アドレスを昇順にソーティングして出力する場合

```
ORDER
```

・アドレスを任意（入力順）で出力する場合

```
NOORDER
```

＜補足＞

・NOORDER コントロールを指定した場合変換スピードは早くなりますが，アドレスが昇順となりませんので注意[*1]が必要です．

＊1）・本プログラムの出力をそのまま紙テープにパンチしてＲＯＭ発注することはできません．

第２章　機　　能

２．１　HEX形式オブジェクト・プログラムの作成

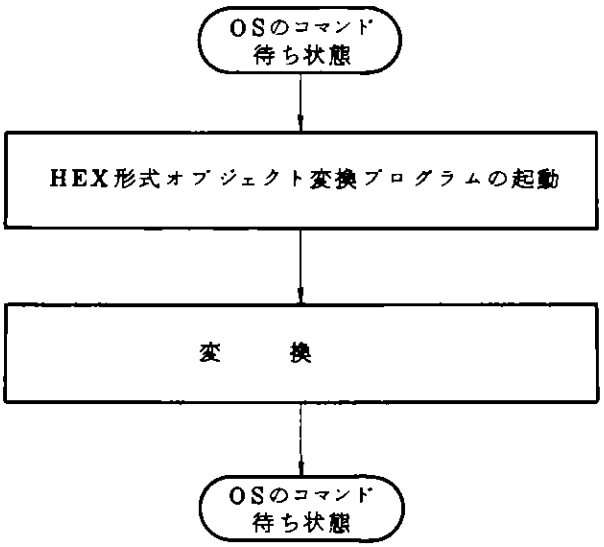
アセンブラまたはリンカが作成したオブジェクト／ロード・モジュール（すべての参照アドレスが解決されているもの）を拡張ＨＥＸ形式オブジェクト・モジュールに変換し，ＦＤへ出力します．

第3章　操 作 法

3．1　操作手順

ＨＥＸ形式オブジェクト変換プログラムの操作手順を図３－１に示します．

図３－１　操作手順



・次の指定により，ＨＥＸ形式オブジェクト変換プログラムを起動します．

```
—RUN[:d:]OC116 入力ファイル名　TO　出力ファイル名　コントロール　[　...　]
```

d　：OC116.86を格納しているFDまたはHDドライブ名
（省略時はF0となります．）

OC116：HEX形式オブジェクト変換プログラムのファイル名

入力ファイル名：拡張ＨＥＸ形式オブジェクト・モジュールに変換する，オブジェクト・モジュールまたはロード・モジュールのファイル名
・媒体はＦＤまたはＨＤに限ります．

出力ファイル名：拡張ＨＥＸ形式オブジェクト・モジュール・ファイル名
・媒体はＦＤまたはＨＤに限ります．

ファイル名の形式は，(1)ファイル名規則を参照してください．

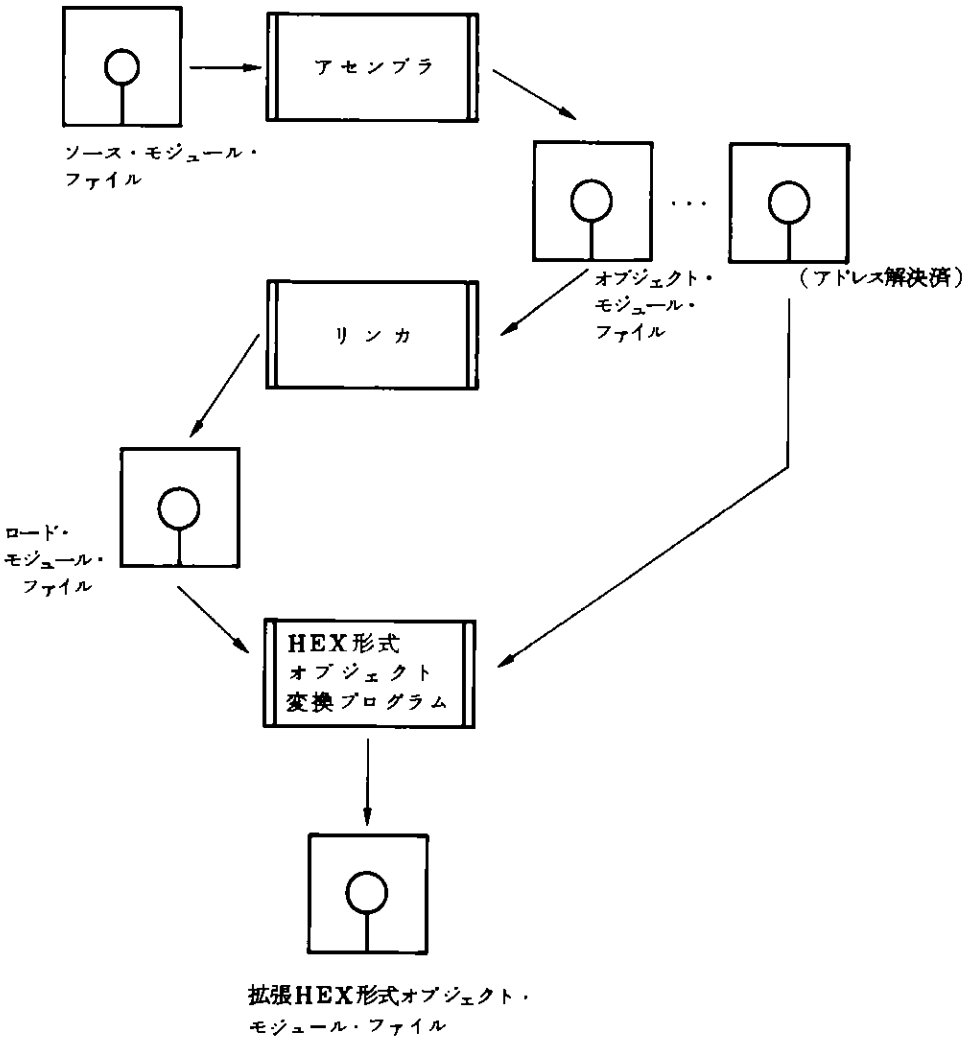
第１章　ソフトウエア概要

１．１　HEX形式オブジェクト変換プログラムの機能概要

・ＨＥＸ形式オブジェクト変換プログラムは，アセンブラまたはリンカが作成したオブジェクト／ロード・モジュール（すべての参照アドレスが解決されているもの）を入力し，拡張ＨＥＸ形式オブジェクト・モジュールに変換します．

・アセンブラからＨＥＸ形式オブジェクト変換プログラムまでの手順の概要を図１－１に示します．

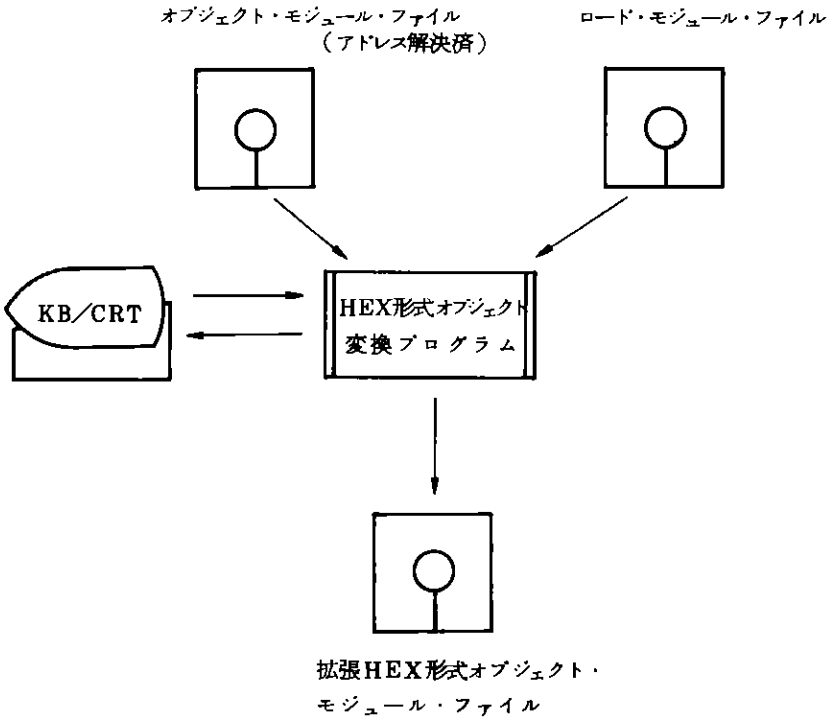
図１－１　アセンブラからＨＥＸ形式オブジェクト変換プログラムまでの流れ



1.2 システム構成

システムの構成図を，図1－2に示します．

図1－2 システム構成



エラー番号 F115	メッセージ	SPECIFIED SEGMENT NOT FOUND IN INPUT MODULE(SEGMENT：セグメント名)
	原因	ADDRESSまたはORDERコントロールで指定したセグメント名は入力モジュール内に存在しません.
	リンカの処理	OSにコントロールを戻す.
エラー番号 W116	メッセージ	SEGMENT OVERLAP(SEGMENTS： セグメント名ANDセグメント名)
	原因	セグメント間で配置アドレスの重複があります.
	リンカの処理	重複させたまま処理続行.
エラー番号 F117	メッセージ	MEMORY REQUIRED BY INPUT MODULE EXCEEDS 1 MEGABYTE
	原因	出力モジュールのサイズが1Mバイトを越えています.
	リンカの処理	OSにコントロールを戻す.
エラー番号 W118	メッセージ	MEMORY SEGMENT NOT PLACED HIGHEST IN MEMORY
	原因	MEMORY属性のセグメントのアドレスが他のセグメントより高いアドレスにできません.
	リンカの処理	そのままの配置アドレスで処理続行
エラー番号 F119	メッセージ	SPECIFIED CLASS OUT OF ORDER (CLASS：クラス名)
	原因	クラスの指定にORDERコントロールとADDRESSESコントロールで矛盾があります.
	リンカの処理	OSにコントロールを戻す.
エラー番号 F120	メッセージ	SPECIFIED SEGMENT OUT OF ORDER (SEGMENT：セグメント名)
	原因	セグメントの指定にORDERコントロールとADDRESSESコントロールで矛盾があります.
	リンカの処理	OSにコントロールを戻す.
エラー番号 W121	メッセージ	CAN NOT MAINTAIN SPECIFIED ORDER (SEGMENT：セグメント名)
	原因	ORDERコントロールで指定した順序でセグメントを配置できません.
	リンカの処理	そのままの配置アドレスで処理続行.
エラー番号 F123	メッセージ	ADDRESS FOR CLASS SPECIFIED MORE THAN ONCE(CLASS：クラス名)
	原因	同じクラスに対して複数回ADDRESSESコントロールでアドレスを指定しています.
	リンカの処理	OSにコントロールを戻す.