お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

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

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

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

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

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

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

RENESAS

ユーザーズ・マニュアル

AS6133 Ver.2.21以上

アセンブラ

PC-9800（MS-DOS™ベース）

IBM PC/AT™（PC DOS™ベース）

対象デバイス

μPD6133シリーズ
μPD6604シリーズ
μPD63シリーズ
μPD67シリーズ

資料番号　U10115JJ3V0UM00（第３版）
発行年月　July 2001 N　CP(K)

〔メ　モ〕

目次要約

第Ⅰ編　言 語 編

第Ⅱ編　操 作 編

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

PC/AT, PC DOSは，米国IBM社の商標です。

- **本資料の内容は予告なく変更することがありますので，最新のものであることをご確認の上ご使用ください。**
- **文書による当社の承諾なしに本資料の転載複製を禁じます。**
- **本資料に記載された製品の使用もしくは本資料に記載の情報の使用に際して，当社は当社もしくは第三者の知的財産権その他の権利に対する保証または実施権の許諾を行うものではありません。上記使用に起因する第三者所有の権利にかかわる問題が発生した場合，当社はその責を負うものではありませんのでご了承ください。**
- **本資料に記載された回路，ソフトウエア，及びこれらに付随する情報は，半導体製品の動作例，応用例を説明するためのものです。従って，これら回路・ソフトウエア・情報をお客様の機器に使用される場合には，お客様の責任において機器設計をしてください。これらの使用に起因するお客様もしくは第三者の損害に対して，当社は一切その責を負いません。**

M7A 98.8

本版で改訂された主な箇所

箇　所	内　　容
はじめに	対象デバイスを追加
	２．１　PC-9800シリーズ，２．２　IBM PC/AT互換機の記述を変更
p.46	**第Ｉ編　３．４　疑似命令，制御命令一覧**に追加
p.83	**第II編　表３－１　記述可能なデバイス名と対応デバイス**の記述を変更
p.94	**４．４．２　アセンブラの起動方法**の［　例　］の記述変更
p.111	**５．４．１　許容ビット数を越えた命令のエラー・チェック**に記述を追加
p.112	**５．４．３　分岐命令の分岐先チェックBANK0, 1の自動判別**の記述を変更
p.114	**５．４．５　存在しないポートに対しての入力，出力命令チェック**に記述を追加

本文欄外の★印は，本版で改訂された主な箇所を示しています。

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

はじめに

★ １．AS6133アセンブラは，次の４ビット・マイクロコントローラをサポートしています。

シリーズ名	対応可能なデバイス
μPD6133	μPD6132, 6132A, 6133, 6134, 6135, 61P34B
μPD6604	μPD6603, 6604, 6605, 66P04B
μPD63	μPD62, 62A, 63, 63A, 64, 64A, 6P4B, 65, 6P5
μPD67	μPD67, 68, 69, 6P9

２．AS6133アセンブラは次の環境で動作します。

★ 2.1 PC-9800シリーズ

（1）PC-9800シリーズの対象OS

MS-DOS　Ver.5.0以降[注1]

Windows™ 3.1/95/98[注2]

WindowsNT™4.0[注2]

注１．Ver.5.00, 5.00Aにはタスク・スワップ機能がありますが，このソフトウエアではタスク・スワップ機能は使用できません。

２．MS-DOSプロンプト（Windows3.1/95/98）またはコマンド・プロンプト（Windows NT）でのみ使用可能です。

AS6133アセンブラは，当社提供のPC-9800シリーズ用MS-DOSまたはWindows上，またはマイクロソフト社提供のPC-9800シリーズ用Windows上にて動作いたします。

このほかの市販MS-DOSまたはWindows上における本アセンブラの動作については，その責を負いかねますのでご了承ください。

なお，MS-DOS内のCONFIG.SYSファイルの設定は次のようにしてください。

- files = 15（15以上）
- buffers = 10（10以上）

★　2.2　IBM PC/AT互換機

（1）IBM PC/AT互換機の対象機種

次のOSが動作するIBM PC/AT互換機パーソナル・コンピュータ

（2）IBM PC/AT互換機の対象OS

MS-DOS　Ver.6.0以降[注1]

PC DOS　Ver.6.1以降[注1]

Windows3.1/95/98[注2]

WindowsNT4.0[注2]

注1．Ver.6.0, 6.1にはタスク・スワップ機能がありますが，このソフトウエアではタスク・スワップ機能は使用できません。

2．MS-DOSプロンプト（Windows3.1/95/98）またはコマンド・プロンプト（Windows NT）でのみ使用可能です。

AS6133アセンブラは，マイクロソフト社提供のIBM PC/AT互換機用MS-DOSまたはWindows上，または日本IBM社提供のIBM PC/AT互換機用PC DOS上にて動作いたします。

このほかの市販MS-DOS，WindowsまたはPC DOS上における本アセンブラの動作については，その責を負いかねますのでご了承ください。

3．供給形態

3.1　アセンブラ

（1）ファイル名

AS6133.EXE

（2）フロッピィ・ディスクの形式

PC-9800シリーズ：高密度の3.5インチ・フロッピィ・ディスク（3.5"2HD）

IBM PC/AT互換機：高密度の3.5インチ・フロッピィ・ディスク（3.5"2HD）

４．本マニュアルで使用される記号の意味

…	同一の形式の連続
[　]	かっこ内を省略可能
{　}	かっこ内の１つを選択
△	１個の半角のスペースまたはTAB
“　”	ダブル・クォーテーション内の文字そのもの
CR	キャリッジ・リターン
LF	ライン・フィード
TAB	水平タブ
○○○	任意の文字列を意味します
×××	任意の文字列を意味します
□□□	任意の文字列を意味します
≡	対応する内容を表します
<>	かっこ内の内容相当のものを表します

５．ファイル名の規則

［ドライブ名：］　［￥ディレクトリ名￥…］ファイル名　　［．拡張子］

ドライブ名……ファイルを格納しているフロッピィ・ディスクがセットされているドライブ名。
ただし，ドライブ名が省略された場合はカレント・ドライブが選択されます。

ファイル名……半角８文字，または全角４文字以内の文字列

拡張子　……半角３文字以内の文字列

目　　次

第Ⅰ編　言 語 編

第II編　操 作 編

図の目次

図番号　　タイトル，ページ

第Ⅰ編　言 語 編

第Ⅱ編　操 作 編

表の目次

表番号　　タイトル，ページ

第Ⅰ編　言 語 編

第Ⅱ編　操 作 編

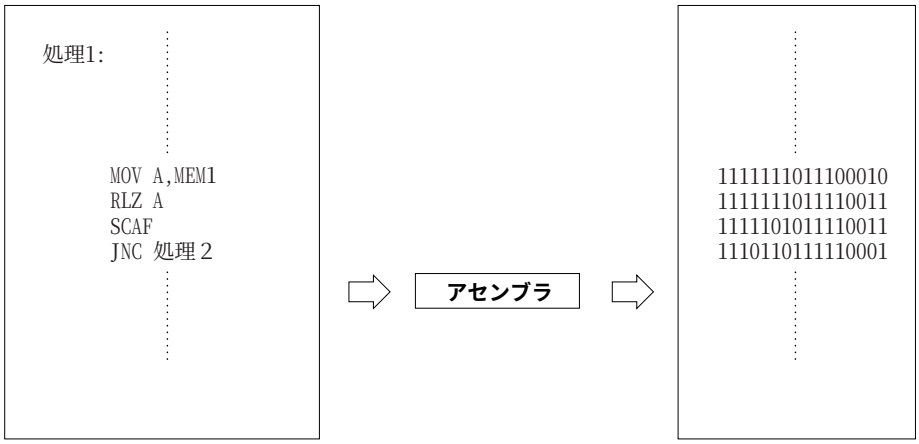
第 I 編
言 語 編

第１章　概　　説

１.１　アセンブラ概説

１.１.１　アセンブラとは

マイクロコントローラは，０と１の組み合わせからなる機械語と呼ばれる言語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語（シンボリック言語またはアセンブリ言語）を割り当てることにより，より解りやすくプログラムを記述することができます。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，マイクロコントローラが唯一理解できる機械語に変換するプログラムです。



アセンブラは，アブソリュート・アセンブラとリロケータブル・アセンブラに分類できます。

AS6133はアブソリュート・アセンブラに分類することができますが，従来のアブソリュート・アセンブラとは異なり分割プログラミングが可能です。したがってAS6133はアブソリュート・アセンブラでありながらリロケータブル・アセンブラに近い性質を持っているといえます。

１.１.２　アブソリュート・アセンブラとは

機械語は命令とデータによって構成されます。命令はマイクロコントローラに対し，動作の種類を指定するものであり，データはそのとき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付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，アセンブルする時点でアドレスやデータはすべて決定されていなければなりません。その情報は“ ORG ”というロケーション・カウンタ制御疑似命令でアセンブラに伝えられます。

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され，マイクロコントローラによって実行させることができます。この生成された機械語をアブソリュート・オブジェクト・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。

1.1.3　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生成されるアブソリュート・オブジェクト・モジュールは，データやアドレスが絶対的に決定しているものです。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）なオブジェクト・モジュールを生成するアセンブラをリロケータブル・アセンブラと呼びます。また，リロケータブル・アセンブラによって生成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。

リロケータブル・オブジェクト・モジュールの機械語は，そのままではプログラムとしてマイクロコントローラで実行させることはできません。これは，アドレスやデータが相対的な値（仮の値）となっているためです。このリロケータブル・オブジェクト・モジュールをマイクロコントローラで実行できる形に変換するのが「リンカ」です。

リンカとは

リンカは，リロケータブル・アセンブラから生成された複数のリロケータブル・オブジェクト・モジュールの配置を決定し，アドレスの参照関係を解決し，１つにまとめる働きをします。そして，相対的な値が与えられていたアドレスやデータに絶対的な値を与えます。

このリンカから出力された１つのまとまりをロード・モジュールと呼びます。ロード・モジュールはそのままの形ではマイクロコントローラで実行させることはできません。マイクロコントローラで実行させる形に変換する必要があります。

1.1.4　μPD6133シリーズ・システム開発フロー

μPD6133シリーズを用いたシステム開発の全工程のフローを図１－１に示します。またソフトウエア開発の詳細なフローは図１－２に示します。

図1－1　システム開発フロー

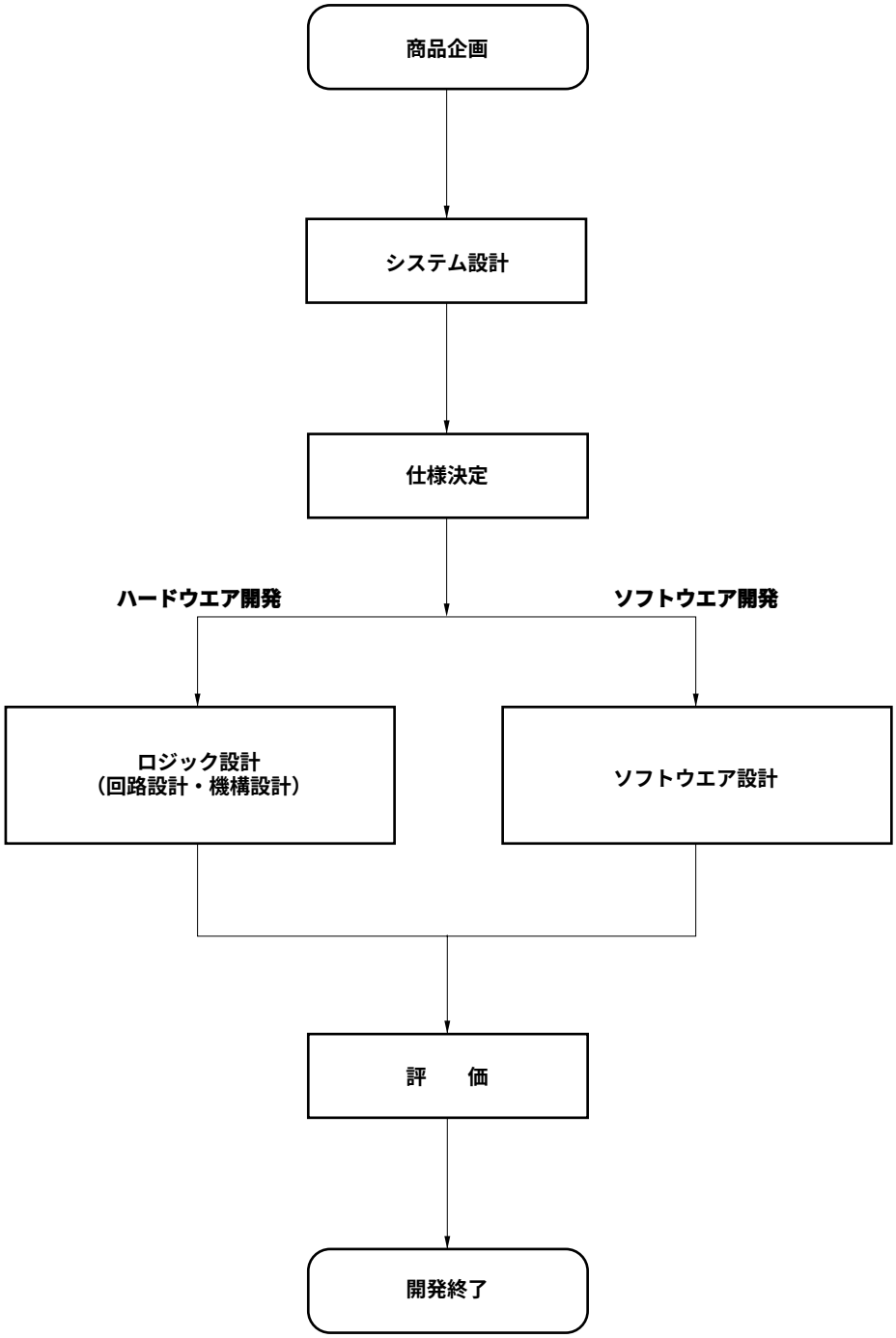
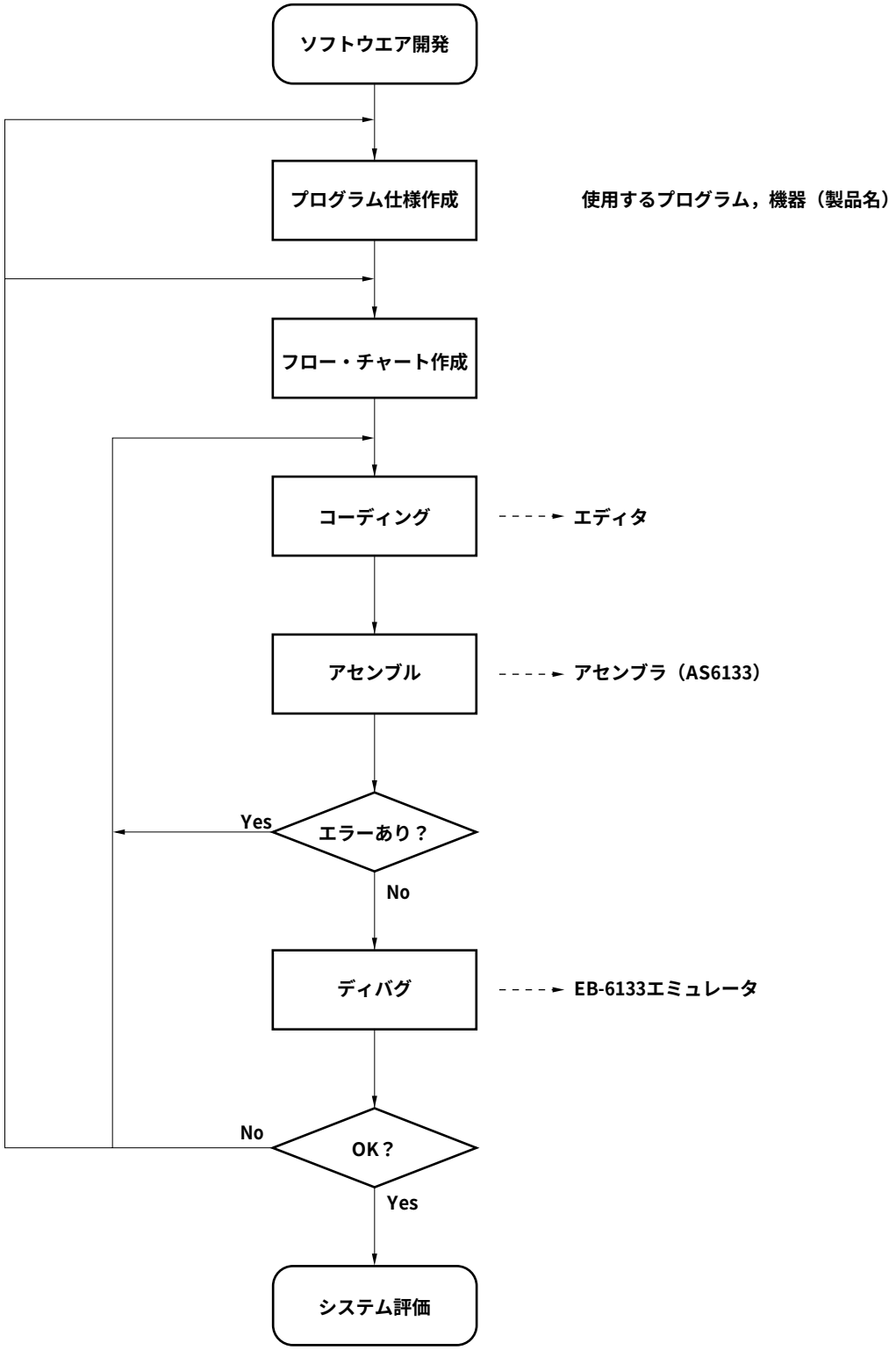


図1－2　ソフトウエア開発フロー



1.1.5　アセンブラの比較

アブソリュート・アセンブラとリロケータブル・アセンブラの特徴を表１－１に示します。

表１－１　アセンブラの比較

		アブソリュート・アセンブラ	リロケータブル・アセンブラ
アセンブル方式		一括アセンブル （ただしAS6133は疑似的に分割可能）	分割アセンブル ↓ リンク
アセンブル・リストの番地表示		絶対番地	相対番地
変数	オペランド部分の変数演算の制限	なし	リンカによって制限される。
	ローカル変数	定義不可能 （ただしAS6133は定義可能）	定義可能
その他		一括アセンブルのため，ソースを一部だけ修正した場合でもアセンブル時間は速くならない（ただしAS6133では疑似的に分割アセンブルができるためプログラミングの方法により速くすることが可能）。	ディバグ時にアドレス計算が必要。 分割アセンブル可能なため，複数人数で各モジュールごとにプログラミング可能。

1.2　μPD6133シリーズ・アセンブラ機能概要

1.2.1　シーケンス・ファイルの作成

μPD6133シリーズ・アセンブラ（AS6133）は，アブソリュート・アセンブラです。AS6133はアブソリュート・アセンブラながら，リロケータブル・アセンブラの特徴であるモジュール・プログラミングが可能です。ただし，リロケータブル・アセンブラ・パッケージのようにリンカというプログラムはありませんが，AS6133はリンクの機能を兼ね備えています。

ソース・モジュールを分割してプログラミングした場合，それらのソース・モジュール・ファイルの連結の順序などを記述したシーケンス・ファイルが必要です。シーケンス・ファイルは，各ソース・モジュール・ファイルの連結順序のほか，デバイス名，アセンブル実行時のオプションを指定します。

1.2.2　ソース・モジュール・ファイルの作成

プログラムは一般に機能ごとに幾つかのサブプログラムに分割して設計します。サブプログラムの独立性を機能的に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，あとの保守をしやすくすることにつながります。

１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・モジュールのコーディングが終わったらエディタ等を使ってファイルに書き込みます。こうしてできたファイルをソース・モジュール・ファイルと呼びます。

ソース・プログラムを分割した場合，シーケンス・ファイルには各ソース・モジュールの連結の順序などを記述します。

またINCLUDE文によるファイル分割は，上記のソース・モジュール分割とは意味が異なります。すなわちINCLUDE疑似命令により指定されたファイルは，そのINCLUDE疑似命令があるソース・モジュールの一部といえます。

1.2.3　日本語使用可能

AS6133は，日本語（８ビットJISコード，シフトJISコード）で記述したソース・プログラムを，アセンブルすることができます。

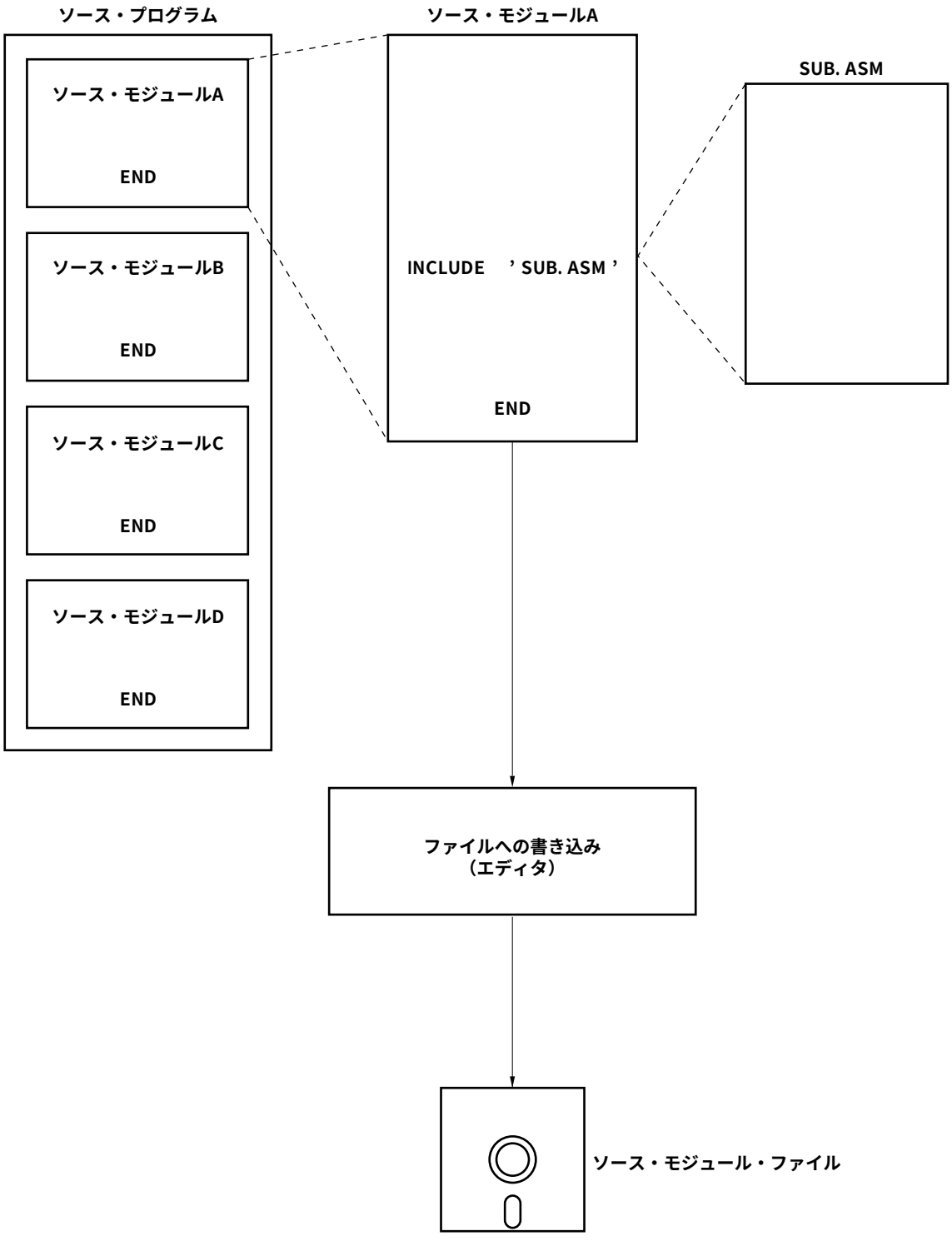
コメント欄はもちろんシンボル欄にも日本語を使用できますので，非常にわかりやすいソース・リストを作成することができます。

1.2.4　外部モジュール定義参照機能

PUBLIC, EXTRN疑似命令により外部モジュールで定義されたシンボルの参照が可能になります。PUBLIC宣言されたシンボルは，EXTRN宣言によりいつでも参照することができます。

後方モジュールで定義されているシンボルは，アセンブル時に参照を行い，前方モジュールで定義されているシンボルは，リンク時に参照します。

図1－3　ソース・モジュール・ファイルの作成



1.2.5　アセンブルの実行

ソース・モジュールをアセンブルするには次のファイルが必要です。

●アセンブラ　　（AS6133.EXE）
●ソース・モジュール・ファイル（○○○○．ASM, ××××．ASMなど）
●シーケンス・ファイル　　（□□□□．SEQ）

AS6133起動時にコンソールより直接，あるいはシーケンス・ファイル中のアセンブル・オプションの指定により出力リストの制限が行えます。

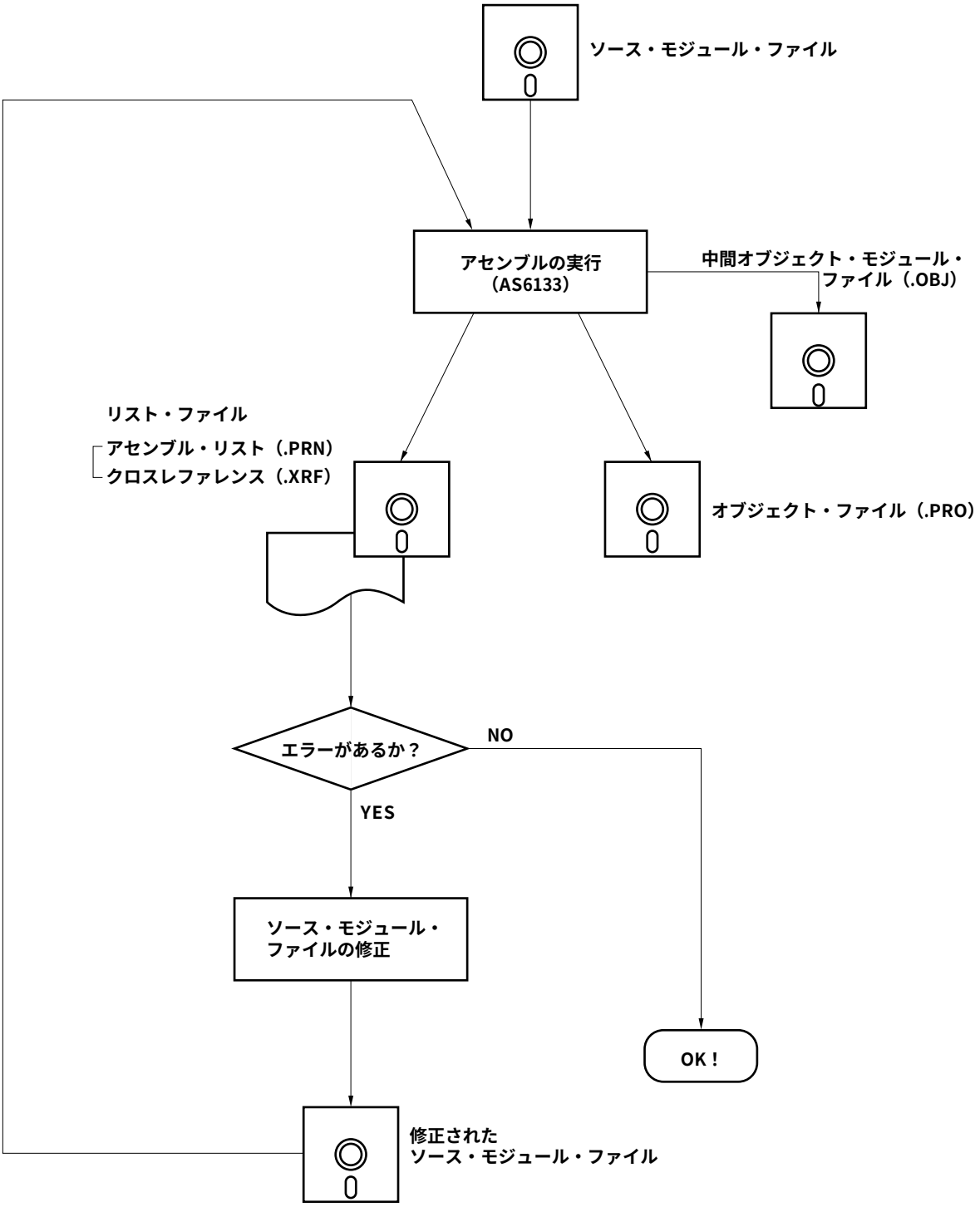
アセンブル・リスト上などでエラーを見つけたら，ソース・モジュールを修正し，エラーがなくなるまでアセンブルを繰り返してください。

ソース・プログラムをモジュール分割した場合，AS6133はアセンブル実行時に中間オブジェクト・モジュール・ファイル（.OBJ）を作成します。この中間オブジェクト・モジュール・ファイルは，ソース・プログラムを一部修正して再アセンブルするときに利用されます。

AS6133ではアセンブル時間を短縮するため，修正されたソース・モジュールのみ再アセンブルし，未修正のソース・モジュールは，すでに作成された中間オブジェクト・モジュール・ファイルを利用しています。修正されたかどうかはソース・モジュール・ファイルと，それと同名の中間オブジェクト・モジュール・ファイルの作成日時を比較して，ソース・モジュール・ファイルの方が新しい場合，修正されたと判断されます。したがって中間オブジェクト・モジュール・ファイルがない場合や，ソース・モジュール・ファイルの方が古い場合，中間オブジェクト・モジュール・ファイルの作成からアセンブラが自動的に判断してアセンブルを実行します。

このアセンブル時間短縮機能により，ディバグが進むに従い，アセンブル時間は大幅に短縮されていきます。

図1－4　オブジェクト・ファイルの作成



1.3　プログラム開発をはじめる前に

本節では，AS6133をスムーズに使用するために，あらかじめ注意する必要のある項目について述べます。具体的な説明については後節で述べます。

1.3.1　シンボルの制限について

（1）数の制限

１つのソース・モジュール内で使用できるシンボルのテーブル領域は，最大64 Kバイトあります。

AS6133では，１つのシンボルの文字数を最大253文字（１バイト／文字）[注]まで定義できます。また，使用できるシンボル数は下記のとおりです。

すべてのシンボル長を253文字とした場合 ……240個設定可

すべてのシンボル長を８文字とした場合……3368個設定可

注　全角文字（シフトJISコード）は，２バイト／文字として扱われます。

（2）マクロ内シンボル

グローバルな宣言を行っていないシンボルは，ローカル・シンボルとして扱われます。

1.3.2　疑似命令の制限

MACRO, REPT, IF文のネスティング・レベルは，トータルで40レベルです。疑似命令定義中に別の疑似命令を展開する場合，40レベル以下になるように注意することが必要です。

ただし，マクロ内でのマクロ名参照はできますがマクロ定義はできません。

表１－２　ネスティング可能な疑似命令と最大ネスティング・レベル

ネスティング可能な疑似命令	最大ネスティング・レベル	トータル
REPT～EXITR～ENDR	8	40
IF～ELSE～ENDIF	40	
MACRO～ENDM	40	
INCLUDE	8	8[注]

注　INCLUDE文のネスティングは，上記疑似命令とは独立して使用できます。

1.3.3　日本語使用時の注意

ソース・リストは，日本語のエディタで作成できます。

使用できる文字は，８ビットJISコード（半角）とシフトJISコード（全角）です。

ただし，予約語は，半角で記述してください。

また，全角のスペース，コロン，セミコロンをシンボル欄，ニモニック欄，オペランド欄を区切るために使用しないでください。全角文字と半角文字は，コードが異なります。

たとえば，スペースを全角で記述すると区切り記号でなく，空白という文字として扱われます。

1.3.4　ホスト・マシンの日付，時刻設定

ホスト・マシン（PC-9800シリーズ）上でMS-DOSを起動したとき，必ず現在の日付と時刻の確認をしてください。

AS6133では，アセンブル実行時にソース・モジュール・ファイルと同名の中間オブジェクト・モジュール・ファイルとの作成日時の比較を行います。比較した結果，中間オブジェクト・モジュール・ファイル作成日時の方が新しいと判断されると，そのソース・モジュールはアセンブルされません。

ホスト・マシンの時計の時刻がソース・モジュール・ファイル作成日時より遅れている場合，ソース・モジュール・ファイルを変更したにもかかわらず，アセンブルした結果が変更前の状態と同じになりますので注意が必要です。

1.3.5　ソース・モジュール数の制限

AS6133では，ソース・プログラムを30個までのモジュールに分割してプログラミングすることができます。

分割されたソース・モジュールは，シーケンス・ファイル（.SEQ）上で，アセンブル処理順序を記述することにより，1つのソース・プログラムとして扱われます。

第２章　ソース・プログラムの記述方法

２.１　ソース・プログラムの基本構成

ソース・プログラムは，図１－３のように少なくとも１つのソース・モジュールより構成されます。ソース・モジュールは文により構成します。文の構成については**２.２　文の構成**を参照してください。

ソース・モジュールのサイズに制限はありません。したがって記述できる文の数は無制限です。しかし分割できるソース・モジュール数は最大30モジュールです。

ソース・プログラム中では，命令，疑似命令，制御命令などを，任意の位置に記述できますが，END疑似命令のみは，各ソース・モジュールの最後に記述する必要があります。

また，ソース・モジュール中のINCLUDE疑似命令で，ソース・モジュールに読み込まれるインクルード・ファイルには，END疑似命令を記述する必要はありません。

２.２　文の構成

アセンブラ言語のソース・プログラムは，文（statement）で構成します。

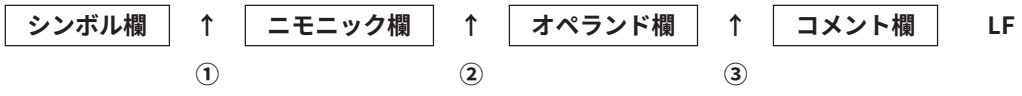
１つの文は，**２.４　文字セット**で表される文字を使って記述します。

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CR（キャリッジ・リターン）コード，LF（ライン・フィード）コードでターミネートされますが，アセンブラはこのうちLFコードを文の終わりと判断し，CRコード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，ニモニック欄，オペランド欄，コメント欄の４つのフィールドで構成します。

各欄は，半角のスペース（　）（８ビットJISコードの20H），TABコード（09H），半角のコロン（：）（3AH ）または，半角セミコロン（；）（3BH）で区切ります。また１行は最大256文字まで記述することができます。

文は自由記述形式で，シンボル欄，ニモニック欄，オペランド欄，コメント欄の順であれば任意のカラムから書けます。つまりシンボルのみ，コメントのみ，あるいは空文の記述も可能です。



① シンボル欄にシンボルを記述する場合は，半角のコロンまたは，ブランク（１つ以上の半角のスペースかTABコード）で区切ります。

　コロンかブランクかはニモニック欄で記述する命令により異なります。

② オペランド欄が必要な場合は，ブランクで区切ります。

③ コメント欄にコメントを記述する場合は，半角のセミコロンで区切ります。

④　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。

シンボル欄とニモニック欄の間にコロンを記述する場合を例１に，ブランク記述する場合を例２に示します。

[　例１　]

```
AAA  :  MOV        A,#8H        ;  A=8H
```

[　例２　]

```
AAA  EQU  7
```

２.３　タビュレーション機能

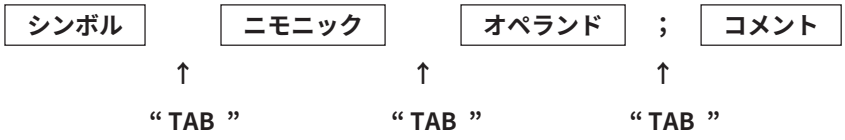
アセンブル・リストを見やすい形式にするため，AS6133には，タビュレーション機能が用意されています。タビュレーション機能とは，ソース・プログラムのシンボル欄，ニモニック欄，オペランド欄，コメント欄がそれぞれ８の倍数のカラムより始まるように整理する機能です。

[　例　]　足し算：

```
MOV        A,#8H
MOV        R01,A        ;R01=8H
```

8の整数（タビュレーションの数）倍カラム目

タビュレーション動作を行う場合はソース・プログラムのニモニック欄，オペランド欄の先頭，およびコメント欄の始まりを示す半角のセミコロン（；）の前に“TAB（Horizontal TAB,09H）”コードを挿入します。



AS6133では使用するプリンタに応じTABのコード（09H）を送るか半角のスペースで埋めて送るかをアセンブル・オプションにより選択することができます。

これはTABコードを認識できないプリンタに対応するために用意してあります。

AS6133ではこのような場合TABコードの変換に，半角のスペースをプリンタに送るように指定することができます。

備考　ディスク容量を効率的に使用するためTABコードを使用するようお奨めします。

2.4　文字セット

文の記述は，8ビットJISコードおよびシフトJISコードを使用してください。

シンボルとして使用できる文字には制限があります。詳細については，**2.5.1　シンボル記述上の規則**を参照してください。また，予約語には半角の英小文字と半角の英大文字との区別はありませんが，ユーザが定義するシンボルには，区別があります。

[　例1　]

```
AAA         EQU  3
AAa         EQU  5
```

AAAとAAaは別のシンボルと判断します。

[　例2　]

```
MOV         MEM1,#1
MOV         mem1,#3
```

MEM1とmem1は別のシンボルです。MEM1には1が，mem1には3が設定されます。また予約語のMOVはMovでも同じに解釈されます。

2.4.1　英 数 字

半角の英字と半角のアラビア数字をまとめて英数字といいます。

2.4.2　数　　字

2進数字　0と1の2個の文字を2進数字といいます。

8進数字　0，1，2，3，4，5，6，7の8個の文字を8進数字といいます。

10進数字　0，1，2，3，4，5，6，7，8，9の10個の文字を10進数字といいます。

16進数字　0，1，2，3，4，5，6，7，8，9，A，B，C，D, E，Fの16個の文字を16進数字といいます。

２.４.３　特殊文字

以下の特殊文字は半角文字を使用したとき有効となります。全角文字の場合は，特殊文字としては使用できず，単なる文字として解釈されます。

また，これらの特殊文字は文字列（文字定数）やコメント欄では，文字そのものとして使用できます（LFコードを除く）。

半角文字	名　　称	主な用途
空白	空白（スペース）	各欄の区切り記号
?	疑問符	英字相当文字
@	単価記号	間接アドレシング指定記号
_	下線	英字相当文字
,	コンマ	オペランド間の区切り記号
.	ピリオド	ビット区分演算子
+	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
*	アスタリスク	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	
$	ドル記号	ロケーション・カウンタの値
=	等符号	比較演算子
;	セミコロン	コメントの開始記号
;;	ダブルセミコロン	マクロ内コメントの開始記号
:	コロン	レーベルの区切り記号
'	引用符	文字定数の開始・終了記号
<		比較演算子
>		
#		イミーディエト・データ指定記号
&	アンパーサンド	マクロ内での文字列の結合を指定する記号
%	式演算子	マクロ・パラメータの直前に記述され，値の引き渡しであることを示す記号
TABコード	水平タブ	８文字の空白相当文字
LFコード	ライン・フィード	文の終了記号
CRコード	キャリッジ・リターン	通常はアセンブラによって無視されます。

２.５　シンボル欄

AS6133では，命令や疑似命令で使用するデータやアドレスを数値や数値式で記述するとエラーとなります。したがって，データやアドレスを使用する場合には，その内容を簡潔に示した「名前」を用いてください。そして，データやアドレスに付けた名前を「シンボル」と呼びます。

アセンブラのシンボル欄にはこの「シンボル」を記述し，このことを「シンボルを定義する」といいます。

プログラムで使用するシンボルは，使用する前に宣言しておけば，どこに記述しても構いません。ただし，宣言する位置によりその適用範囲（スコープ）が違います。

１つのモジュールでは，同じ名前のモジュールを使用することはできませんが，シンボルは基本的にモジュール内でのみ使用可能ですので，異なるモジュールでは，同じ名前を使用することができます。１つのシンボルを複数のモジュールに渡って使用したい場合は，PUBLIC宣言を行います。

シンボルは，その使用目的，定義の方法によって，レーベルとネームに分類されます。

（１）ネーム

EQU，SET疑似命令で定義するシンボルをネームと呼びます。ネームには数値データやアドレスが割り付けられます。それらの数値データやアドレスに代わって，プログラム上では定義されたネームを使用することができます。

つまり数値データをそのまま扱わず，その数値データに名前を付けて利用するときに使用します。

［　例　］

```
DATA1       EQU     8H
```

8Hという数値にDATA1というネームを定義

（２）レーベル

命令（ニモニック）のアドレスまたはORG，DW，DT疑似命令に割り付けるシンボルです。レーベルは，それが付けられた命令および疑似命令に割り付けられたプログラム・メモリ・アドレス（ロケーション・カウンタの値）を参照するために使用します。

つまりレーベルは，あるルーチンの先頭番地にそのルーチンの処理内容を表すような名前を付け，他のルーチンからそのルーチンへの分岐や参照する場合に使用します。

［　例　］

```
LOOP:MOV        A,@ROH
      ⋮
     JMP        LOOP
```

この例で“ LOOP ”をレーベルと呼びます。

２.５.１　シンボル記述上の規則

シンボル記述上の規則は，次のとおりです。

（１）シンボルには半角の特殊文字（下線と疑問符を除く）以外の８ビットJISコードとシフトJISコードが使用できます。

先頭文字に半角の数字は使用できません。

（２）シンボルの長さは１～253文字（半角文字の場合）です。シンボルが254文字以上になるとＳエラー（構文誤り）が発生します。ただし，シフトJISコードを使用した文字は１文字で半角の英数字２文字分となります。

（３）レーベルは半角のコロン（：）（3AH）でターミネートします。（レーベルとコロンの間に半角のスペースまたはTABコードがあってもかまいません）。

（４）EQU疑似命令，SET疑似命令，MACRO疑似命令を使用するときは，必ずシンボル欄にネームを記述してください。ネームは半角のスペースまたはTABコードでターミネートします。

（５）同一シンボルを二度以上定義することはできません。二度以上定義すると，Ｓエラー（シンボル二重定義）が発生します。ただしSET疑似命令で定義するシンボル[注]，およびマクロ内で定義されたグローバル宣言されていないシンボルは除きます。

またパブリック宣言されていなければ，同一シンボルを別のモジュールで使用することが可能です。このときそれらは異なったシンボルとみなされます。

（６）予約語をシンボルとして定義することはできません。ただし，予約語を含む語をシンボルとして定義することはできます。

（７）シンボルには，英大文字と英小文字の区別があります。

[例１]

正しい例	誤った例
FIF4:	1F4F:……数字で始まっています。
LABEL:	LABEL ……コロンが付いていません（レーベルの場合）。
HERE:	HE RE: ……シンボル中にブランクがあります。
ANH:	ANL: ……命令は使用できません。
ENDX:	END: ……疑似命令は使用できません。

[例２]

```
ABC     EQU     3
XYZ     EQU     ABC
```

ABCとXYZには同じデータ“３”が割り付けられます。

注　SET疑似命令で定義されたシンボルは，値を変更することができます。値を変更する場合もSET疑似命令を使用します。

2.6　ニモニック欄

ニモニック欄には，命令，疑似命令，制御命令を記述します。

オペランドの必要な命令の場合，ニモニック欄とオペランド欄を区別するために，ブランク（1つ以上の半角のスペースかTABコード）が必要です。

[　例　]	正しい例	誤った例
	JMP LOOP	JMPLOOP …… ニモニックとオペランド間にブランクがありません。
	RET	RE T …… ニモニック中にブランクが挿入されています。
	SCAF	SCA …… μPD6133シリーズの命令にSCAはありません。

２.７　オペランド欄

オペランド欄には命令の実行に必要なデータ（オペランド）を書きます。命令によりオペランドの必要ない命令や，１個または２個のオペランドを必要とする命令があります。

２個のオペランドを必要とする場合，各オペランドは“ , ”で区切ります。

ニモニック欄とオペランド欄の間には，ブランクが必要です。

２.７.１　オペランド欄の記述形式

（1）定　　数

定数は，数字だけで構成する数値定数と文字で構成する文字定数があります。

数値定数には，２進定数，８進定数，10進定数，16進定数があり，半角で記述します。

（a）２進定数

２進文字列の最後に半角文字“ B ”を付加して表現します。

（b）８進定数

８進文字列の最後に半角文字“ O ”または半角文字“ Q ”を付加して表現します。

（c）10進定数

10進文字列の最後に半角文字“ D ”を付加するか，あるいは何も付加せずに表現します。

（d）16進定数

16進文字列の最後に半角文字“ H ”を付加して表現します。先頭の文字が０～９以外の半角文字で始まる場合には先頭に“ 0 ”を付加します。

（ｅ）文字定数

文字定数は８ビットJISコード（LFコードを除く）およびシフトJISコードを半角の引用符（’）で囲んだものです。

AS6133では，TITLE，INCLUDE疑似命令でのみ使用することができます。

半角の引用符で囲んだ文字はアセンブルの結果，８ビットJISコードまたはシフトJISコードに変換されます。

半角の引用符を文字定数として使用する場合は，引用符を引用符で挟んで記述します。

なお文字定数は演算することはできません。

［　例　］’Ａ’　……41H

（半角）

’Ａ’　……8260H

（全角）

’’’’　……27H

（半角）　２個の引用符を記述すると，１個の半角の引用符が定数として確保されます。

’Ａ’’’　……4127H

（半角）

’’　……20H

（半角のスペース）

’＜’　……203CH

’日’　……93FAH

’日本電気’　……93FA967B93648B43H

（２）$（ロケーション・カウンタ）

“ $ ”はロケーション・カウンタの値を示します。すなわち，“ $ ”が使用されているその命令のプログラム・メモリ・アドレスを表します。

［　例　］

```
アドレス
  101            MOV     A,R11
  102    LOOP:   INC     A
  103            JNC     $-1
  105            JMP     $+20H
```

“JNC　$－１”の$は103H番地を示します。したがって$－１は102H番地を示します。また“JMP $＋20H”の場合の$は105H番地を示します。

“JNC　$－１”はレーベルを用いて“JNC　LOOP”としても同じ動作をします。

（３）シンボル

シンボルをオペランド欄に記述した場合は，そのシンボル（レーベル，ネーム）に割り付けられた値をオペランドの値とします。

［　例１　］

```
A1:     JC      A2
          :
A2:     ANL     A,R10
```

［　例２　］

```
VALUE   EQU     1H
        MOV     A,#VALUE
```

“ MOV　A, #VALUE ”は“ MOV　A, #1H ”と同じ意味を持ちます。

（４）式（Expression）

前記の定数，$，シンボルを演算子により結合した文字式，数値式を“ 式 ”と呼びます。演算子は17種類（＋，－，＊，／，MOD，NOT，AND，OR，XOR，SHR，SHL，EQまたは=，NEまたは＜＞，GTまたは＞，GEまたは＞=，LTまたは＜，LEまたは＜=）あり，演算実行上の優先順位が定められています。

詳しくは**２.９　式と演算子**を参照してください。

２.８　コメント欄

コメント欄は半角のセミコロン（；）で始まり，そのあとにコメント（Comment）を記述します。コメントは，プログラマがアセンブル・リストを参照する場合にプログラム内容の理解を助けるための注釈で，アセンブル・リスト上には出力されますがアセンブラは無視します。

コメントにはすべての８ビットJISコード（LFコードを除く），シフトJISコードを使用できます。

マクロ定義内でセミコロンを続けて（;;）使用した場合には，アセンブラはマクロ定義内コメントとみなし，マクロ展開時，そのコメントは印字されません。

コメントは，LFコードでターミネートされますので，コメントが１行におさまらない場合は，次の行の最初にセミコロン（；）を記述してください。

２.９　式と演算子

２.９.１　式

オペランド欄に，定数，$，シンボル，演算子を用いて表現した文字式，数値式を式と呼びます。

２.９.２　演算子概要

（１）概　　要

AS6133アセンブリ言語の演算子は次の５種類に分けられ，演算実行上の優先順位が定められています。

１．算術演算子

＋，－，＊，／，MOD

２．論理演算子

OR，AND，XOR，NOT

３．比較演算子

EQ，NE，LT，LE，GT，GE

=，<>，<，<=，>，>=

４．シフト演算子

SHR，SHL

５．その他

（　）　（演算順序指定記号）

＆　　（置換演算子）

（２）演算子の優先順位

優先順位は下表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。

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。

次の表は優先順位の高い方を１として記述してあります。

表２－１　演算子の優先順位

優先順位	演　　算　　子
1	（　）　（演算順序指定記号）
2	＊，／，MOD，SHL，SHR
3	＋，－
4	EQ，NE，LT，LE，GT，GE，=，<>，<，<=，>，>=
5	NOT
6	AND
7	OR，XOR

2.9.3　算術演算子

（1）加算演算子（+）

オペランド間の加算をします。

[　例　]

アドレス	シンボル	ニモニック	オペランド
0010	START:	JMP	$+6

JMP命令により16H番地にジャンプします。

（2）減算演算子（−）

オペランド間の減算をします。

[　例　]

アドレス	シンボル	ニモニック	オペランド
0020	BACK:	JMP	BACK-6

JMP命令の実行により1AH番地へジャンプします。

（3）乗算演算子（*）

オペランド間の乗算をします。

[　例　]

アドレス	シンボル	ニモニック	オペランド
	A6:	MOV	A,#(2*3)

MOV命令実行により６（２＊３）がAccにロードされます。

（4）除算演算子（／）

オペランド間の除算をします。余りは捨てられます。

[　例　]

アドレス	シンボル	ニモニック	オペランド
	A5:	MOV	A,#(256/50)

MOV命令実行により５（256÷50）がAccにロードされます。

（５）MOD演算子

オペランドを除算した余りを求めます。

[　例　]	アドレス	シンボル	ニモニック	オペランド
		A5:	MOV	A,#MOD

MOV命令実行により６（256÷50の余り）がAccにロードされます。

２.９.４　論理演算子

（１）OR演算子

オペランド間の論理和を求めます。

[　例　]	アドレス	シンボル	ニモニック	オペランド
		MDFY1:	MOV	A,#(0AH OR 5H)

MOV命令実行により0FHがAccにロードされます。

（２）AND演算子

オペランド間の論理積を求めます。

[　例　]	アドレス	シンボル	ニモニック	オペランド
		MASK:	MOV	A,#(1AH AND 0FH)

MOV命令実行により0AHがAccにロードされます。

（３）XOR演算子

オペランド間の排他的論理和を求めます。

[　例　]	アドレス	シンボル	ニモニック	オペランド
		MDFY2:	MOV	A,#(9AH XOR 9DH)

MOV命令実行により7HがAccにロードされます。

（４）NOT演算子

オペランドの値の１の補数を求めます。

[　例　]

アドレス	シンボル	ニモニック	オペランド
	COMPL:	MOV	A,#(NOT 3H AND 0FH)

MOV命令実行により0CH（NOT 3H で0FFFCH，0FFFCH AND　0FH）がAccにロードされます。

２.９.５　比較演算子

（１）EQ（EQual）演算子

左辺と右辺の値が等しいとき0FFFFH，等しくないとき０が得られます。

“ EQ ”は“ = ”と記述することもできます。

[　例　]

アドレス	シンボル	ニモニック	オペランド
	COMP1:	MOV	A,#(ONE EQ 1)

MOV命令実行により，ONEが１ならAccに0FHがロードされ，ONEが１でなければ０がロードされます。

（２）NE（Not Equal）演算子

左辺が右辺の値と等しくないとき0FFFFH，等しいとき０が得られます。

“ NE ”は“ <> ”と記述することもできます。

[　例　]

アドレス	シンボル	ニモニック	オペランド
	COMP2:	MOV	A,#(ONE NE 1)

MOV命令実行により，ONEが１でなければAccに0FHがロードされ，ONEが１なら０がロードされます。

（３）LT（Less Than）演算子

左辺が右辺の値より小さいとき0FFFFH，そうでないとき０が得られます。

“ LT ”は“ ＜ ”と記述することもできます。

[　例　]	アドレス	シンボル	ニモニック	オペランド
		COMP3:	MOV	A,#(MINI LT 5)

MOV命令実行により，MINIが５より小さければAccに0FHがロードされ，MINIが５と等しいか大きければ０がロードされます。

（４）LE（Less Than or Equal）演算子

左辺が右辺の値より小さいか等しいとき0FFFFH，そうでないとき０が得られます。

“ LE ”は“ <= ”と記述することもできます。

[　例　]	アドレス	シンボル	ニモニック	オペランド
		COMP4:	MOV	A,#(MINI LE 5)

MOV命令実行により，MINIが５より小さいか等しければAccに0FHがロードされ，MINIが５より大きければ０がロードされます。

（５）GT（Greater Than）演算子

左辺が右辺の値より大きいとき0FFFFH，そうでないとき０が得られます。

“ GT ”は“ ＞ ”と記述することもできます。

[　例　]	アドレス	シンボル	ニモニック	オペランド
		COMP5:	MOV	A,#(MAX GT 5)

MOV命令実行により，MAXが５より大きければAccに0FHがロードされ，MAXが５と等しいか小さければ０がロードされます。

（６）GE（Greater Than or Equal）演算子

左辺が右辺の値より大きいか等しいとき0FFFFH，そうでないとき０が得られます。

“ GE ”は“ >= ”と記述することもできます。

[　例　]	アドレス	シンボル	ニモニック	オペランド
		COMP6:	MOV	A,#(MAX GE 5)

MOV命令実行により，MAXが５より大きいか等しければAccに0FHがロードされ，MAXが５より小さければ０がロードされます。

2.9.6　シフト演算子

（1）SHR（Shift Right）演算子

左辺の値を右辺の値だけ右へビット・シフトします。

シフトした結果MSBには0が入ります。

[　例　]	アドレス	シンボル	ニモニック	オペランド
	01FA	FIELD:	MOV	A,#($ SHR 5)

$（アドレス：01FAH）を右に5ビット・シフトします。

この結果，Accには0DHがロードされます。

（2）SHL（Shift Left）演算子

左辺の値を右辺の値だけ左へビット・シフトします。

シフトした結果LSBには0が入ります。

[　例　]	アドレス	シンボル	ニモニック	オペランド
	0021	FLY:	JMP	FLY SHL 2

FLY（アドレス：0021H）を左に2ビット・シフトします。

この結果，0084H番地にジャンプします。

2.9.7　その他の演算子

（1）（　）（演算順位指定記号）

演算子の優先順位に関係なく，カッコでくくられた中を先に演算します。

（　）のネストは16まで記述できます。

[　例　]		
	5+8-6*2/4	= 10
	5+(8-6)*2/4	= 6
	(5+8-6)*2/4	= 3
	2*(0FH-(0BH AND (0AH OR 0FH)))	= 8
	2*0FH-0BH AND 0AH OR 0FH	= 0FH

（2）&（置換）演算子

マクロ定義文中に記述し，マクロ展開時に＆の両側の文字を結合します。＆自身はNULLコードに置き換わります。

[　例　]

```
MOVI  MACRO X
      MOV A,#&X
      ENDM
MOVI  1
MOV   A,#1
```

第３章　疑似命令と制御命令

３.１　疑似命令と制御命令の概要

アセンブラの基本機能は，命令を機械語に変換することです。疑似命令と制御命令は，よりアセンブラを使いやすくし，見やすいリストを得るために用意されています。

疑似命令と制御命令は，機械語には変換されず，アセンブラ自身に対して指示するものです。ただし，組み込みマクロ疑似命令は機械語に変換されます。

３.２　疑似命令

AS6133のニモニック欄には，疑似命令が記述できます。

（１）ロケーション・カウンタ制御疑似命令

- ORG

（２）シンボル定義疑似命令

シンボル定義疑似命令は，任意の数値，データ・メモリ・アドレス，フラグ，レーベルの定義に使用します。

- EQU
- SET

シンボル定義疑似命令により割り付けられた値の変更はできませんが，SET疑似命令で定義されたシンボルは，SET疑似命令により変更することができます。したがって，SET疑似命令はアセンブル時にのみ意味を持つ変数を定義するために利用します。

（３）外部定義，外部参照疑似命令

外部定義，外部参照疑似命令は，モジュール間で共通に使用するシンボルを定義，参照します。

- PUBLIC～BELOW～ENDP（外部定義疑似命令）
- EXTRN（外部参照疑似命令）

（４）データ定義疑似命令

データ定義疑似命令は，テーブル・エリアのデータを定義します。

- **DW**：８ビット長のデータを定義します。
- **DT**：10ビット長のタイマ・テーブル用データを定義します。

（５）条件付きアセンブル疑似命令

条件付きアセンブル疑似命令を有効に活用することにより，プログラミングの効率化および，ソース・プログラムのライブラリ化を図ることができます。

- **IF～ELSE～ENDIF**

（６）繰り返し疑似命令

繰り返し疑似命令を有効に活用することにより，プログラミングの効率化を図ることができます。

- **REPT～（EXITR）～ENDR**

（７）マクロ定義疑似命令

プログラム中で，同一のルーチンを何回も使用する場合は，一般にサブルーチン化により，プログラム・ステップ数を節減する手法が用いられます。サブルーチン化できない似かよった処理ルーチンで，パラメータが異なる場合，効率よくプログラミングするためにマクロ機能が用いられます。

マクロ定義疑似命令は，このマクロを定義するときに使用します。詳細は**3.5 マクロ機能**を参照してください。

- **MACRO～ENDM**

（８）マクロ内シンボル・グローバル宣言疑似命令

- **GLOBAL**

（９）アセンブル終了疑似命令

アセンブル終了疑似命令は，各ソース・（プログラム・）モジュールの終了を指示します。

- **END**

（10）マスク・オプション指定疑似命令

- **OPTION～ENDOP**

３.３　制御命令

AS6133のニモニック欄には，制御命令を記述することができます。制御命令はアセンブル後，機械語に変換されず，出力リストの形式およびソース入力の制御を行います。

なお，すべての制御命令は同一モジュール内でのみ有効です。

（１）出力リスト制御命令

出力リスト制御命令は，アセンブル・リストを見やすくするために使用します。

- **TITLE**：アセンブル・リストにタイトルを印字します。
- **EJECT**：改ページします。

（２）ソース入力用制御命令

ソース入力用制御命令は，１つのプログラム（ソース・モジュール）中で，ファイル分割したい場合，つまりファイルが大きくなりすぎるために分割したい場合や，すでに完成したプログラム（ライブラリ化したプログラム）を利用したい場合に使用します。

- **INCLUDE**

ファイルを参照する場合は，INCLUDE制御命令を用いてそのファイル名を指定します。

３.４　疑似命令，制御命令一覧

次頁より，それぞれの命令を説明します。

表３－１　疑似命令，制御命令一覧

命　令		名　称	参照頁
疑似命令	ORG	ロケーション・カウンタ制御疑似命令	p.47
	EQU	シンボル定義疑似命令	p.48
	SET		p.49
	PUBLIC～BELOW～ENDP	外部定義疑似命令	p.50
	EXTRN	外部参照疑似命令	p.52
	DW	データ定義疑似命令	p.53
	DT		p.54
	IF～ELSE～ENDIF	条件付きアセンブル疑似命令	p.55
	REPT～（EXITR）～ENDR	繰り返し疑似命令	p.56, p.57
	MACRO～ENDM	マクロ定義疑似命令	p.58
	GLOBAL	マクロ内シンボル・グローバル宣言疑似命令	p.60
	END	アセンブル終了疑似命令	p.61
	OPTION～ENDOP	マスク・オプション指定疑似命令	p.62
	USEPOC/NOUSEPOC		p.63
	USECAP/NOUSECAP		p.64
制御命令	TITLE	出力リスト制御命令	p.65
	EJECT		p.66
	INCLUDE	ソース入力用制御命令	p.67

ORG　　ORIGIN　　ORG

シンボル欄	ニモニック欄	オペランド欄	コメント欄
[レーベル：]	ORG	＜式＞	[；コメント]

[機　　能]

ロケーション・カウンタに値を設定します。

[用　　途]

（１）プログラム・メモリの開始番地を指定します。各セグメントの先頭で，ORG疑似命令を記述してください。

（２）テーブル・エリアの開始番地を指定します。これによりテーブル・エリア番地以前で変更が生じても，テーブル・エリア番地には，影響ありません。

[説　　明]

（１）オペランド欄の式としてシンボルも使用できますが，そのシンボルは，それ以前に定義されているものでなければなりません。

（２）プログラムの先頭でORG疑似命令によるアドレス指定をしなければ，アセンブラはロケーション・カウンタに0000番地を割り付けます。

（３）ORG疑似命令で指定したアドレスの値が直前のロケーション・カウンタ値以下の場合，Aエラー（アドレス指定誤り）が発生します。このときオペランドに記述された評価値は無視され，ORG命令の直前のロケーション・カウンタから連続した値が割り付けられます。

（４）ORG疑似命令に付けたレーベル自身には，直前のロケーション・カウンタの値が割り当てられます。

[使 用 例]

```
015D            INC     A
015E            RET
0200    STRT:   ORG     200H
0200            MOV     A,#1
```

レーベルSTRTに15FHが割り付けられます。また，ORG疑似命令のオペランドは200Hなので，MOV命令はアドレス200Hに割り付けられます。

EQU　　EQUATE　　EQU

シンボル欄	ニモニック欄	オペランド欄	コメント欄
ネーム	EQU	＜式＞	[；コメント]

[機　　能]

オペランドで指定された式の値をシンボル欄に記述したネームに割り付けます。

[用　　途]

データ・メモリのアドレス定義に使用します。

[説　　明]

（1）オペランド欄にシンボルを記述する場合，それはすでに定義されているものでなければなりません。

（2）シンボル欄とニモニック欄，ニモニック欄とオペランド欄は，ブランクで区切ってください。

（3）シンボルまたはニモニックの記述にエラーがあるとそのネームは登録されませんので，そのネームを参照した文もエラーとなります。また，オペランドの記述エラーの場合は，ネームには０が割り付けられます。

（4）EQU疑似命令で定義されたネームは，同一モジュール内では他の値に再定義できません。再定義するとSエラー（シンボル二重定義）が発生します。

（5）EQU疑似命令で定義されたネームは，命令のオペランドで指定する場合のみ定義以前に参照することができます。

（6）定義する式の値はμPD6133用コードに変換されず，そのままの値が割り付けられます。

[使 用 例]

```
P3_INIT     EQU     12H
P3_MOD      EQU     P3_INIT
            ;
            OUT     P3,#P3_MOD
```

SET　　SET　　SET

シンボル欄	ニモニック欄	オペランド欄	コメント欄
ネーム	SET	＜式＞	[；コメント]

[機　　能]

オペランド欄に記述した式の値を，シンボル欄に記述したネームに割り付けます。

オペランド欄には，式のほかにメモリ名R_0～R_F，R_{10}～R_{1F}，R_{00}～R_{0F}が記述できます。

[用　　途]

条件付きアセンブル疑似命令（IF～ELSE～ENDIF）や繰り返し操作疑似命令（REPT～ENDR，EXITR）の仮パラメータを設定する場合に使用します。

[説　　明]

（１）シンボル欄とニモニック欄，ニモニック欄とオペランド欄は，ブランクで区切ってください。

（２）シンボルまたはニモニックの記述にエラーがあるとそのネームは登録されませんので，そのネームを参照した文もエラーになります。また，オペランドの記述エラーの場合は，ネームには０が割り付けられます。

（３）SET疑似命令で定義されたネームに，ほかの値を再定義することができます。
定義した値は次のSET疑似命令が来るまで保持されます。

（４）SET疑似命令で定義されたネームは，命令のオペランドで指定する場合のみ定義以前に参照することができます。

（５）定義する式の値はμPD6133用コードに変換されず，そのままの値が割り付けられます。

[使 用 例]

```
IMMED   SET   5
        ANL   A,#IMMED     ;IMMED=5
        ;
IMMED   SET   6
        ANL   A,#IMMED     ;IMMED=6
```

PUBLIC	PUBLIC	PUBLIC
BELOW	BELOW	BELOW
ENDP	END PUBLIC	ENDP

書式１

シンボル欄	ニモニック欄	オペランド欄	コメント欄
［レーベル：］	PUBLIC	＜シンボル群＞	［；コメント］

書式２

シンボル欄	ニモニック欄	オペランド欄	コメント欄
［レーベル：］	PUBLIC	BELOW	［；コメント］
［ネーム	EQU	＜式（EQU型）＞］	［；コメント］
	ENDP		

［機　　能］

外部定義疑似命令の書式は２種類あります。

書式１では，オペランド欄に記述したシンボルを他のモジュールで参照することを宣言します。

書式２では，“ PUBLIC BELOW ”と“ ENDP ”で囲まれたブロック内で定義されたシンボルを他のモジュールで参照することを宣言します。

［用　　途］

他のモジュールで参照するシンボルを宣言します。

［説　　明］

（１）外部定義疑似命令は，ソース・プログラム中のどの位置でも記述可能です。

（２）書式１では，パブリック宣言するシンボルを同一モジュール内でシンボル定義疑似命令を用いて定義する必要があります。もし，シンボル定義が同一モジュール内でなされていないシンボルを記述するとSエラー（Undefined Symbol）が発生します。

（３）書式２では，“ PUBLIC BELOW ”と“ ENDP ”で囲まれたブロック内にシンボル定義疑似命令以外の命令を記述するとＳエラー（Syntax Error）が発生します。

（４）文はLFコードでターミネートされますので，複数のシンボルを記述して１行に収まらない場合は，次の行から再びPUBLIC宣言を行ってください。

（５）“ PUBLIC BELOW ”に対する“ ENDP ”が記述されていない場合，END疑似命令でPエラー（No ENDP Statement）が発生します。

（６）PUBLIC宣言されているシンボルが外部モジュールで参照されていない場合，リンク時にワーニング（Unreference Symbol）が発生します。

PUBLIC	PUBLIC	PUBLIC
BELOW	BELOW	BELOW
ENDP	END PUBLIC	ENDP

[使 用 例]

```
        PUBLIC      VAL1,VAL2
        ・
        ・
        ・
VAL1    EQU         1
VAL2    EQU         2
        ・
        ・
        ・
        PUBLIC      BELOW
VAL3    EQU         3
VAL4    EQU         4
        ENDP
```

EXTRN　　EXTERN　　EXTRN

シンボル欄	ニモニック欄	オペランド欄	コメント欄
[レーベル：]	EXTRN	＜シンボル群＞	[；コメント]

[機　　能]

オペランド欄に記述された，シンボル（他のモジュールでパブリック宣言されたシンボル）を本モジュールで参照することを宣言します。

[用　　途]

他のモジュールでパブリック宣言されたシンボルの中で，本モジュールに必要なシンボルがある場合，EXTRN疑似命令を利用して本モジュールでの使用を可能にします。

[説　　明]

（１）EXTRN疑似命令は，そこで宣言されるシンボルがそのモジュール内で参照されるより前に，記述されなければいけません。

（２）同じモジュール内でEXTRN宣言されているシンボルをシンボル定義すると，Ｓエラー（Symbol Multi Defined）が発生します。

DW　　DEFINE WORD　　DW

シンボル欄	ニモニック欄	オペランド欄	コメント欄
[レーベル：]	DW	＜式＞	[；コメント]

[機　　能]

オペランド欄に記述された式または文字を現在のロケーション・カウンタの値（プログラム・メモリ・アドレス）に，８ビットのオブジェクト・コードとして確保します。

[用　　途]

テーブル・エリアの８ビット・データを定義するときに使用されます。

[説　　明]

（１）＜式＞には，８ビットで表現できる式を１つだけ記述することができます。式の評価値が10ビットを越えた場合は，Vエラー（不正な値）が発生し，ビット８または９が“１”の場合は，ワーニング・メッセージが発生します（オブジェクト・コードのビット８，９は“０”になります）。また，オペランド欄に式を複数記述した場合は，Oエラー（オペランドの数が不正）が発生します。

（２）オペランド欄にシンボルを記述し，記述されたシンボルが未定義の場合，Sエラー（未定義シンボル）が発生します。

（３）オペランド欄に記述した式が適当でない場合，オブジェクト・コードとしてNOP（E0E0H）が生成されます。

注意　DW命令は，タイマ用テーブル・エリア参照（MOV T,@R0）用以外のテーブル・エリアに使用します。タイマ用テーブル・エリア参照には，DT命令を使用してください。

[使 用 例]

```
E.      LOC.    OBJ.      M  I    SOURCE STATEMENT
                E2E0              DW      20H
                E4E0              DW      340H          ;①
```

①では，ワーニングが発生しますが，オブジェクト・コードにはビット８，９を０にした値が入ります。

DT　　DEFINE TIMER　　DT

シンボル欄	ニモニック欄	オペランド欄	コメント欄
[レーベル：]	DT	＜式＞	[；コメント]

[機　　能]

オペランド欄に記述された式または文字を現在のロケーション・カウンタの値（プログラム・メモリ・アドレス）に10ビットのオブジェクト・コードとして確保します。

[用　　途]

テーブル・エリアのタイマ用データを定義するときに使用されます。

[説　　明]

（１）＜式＞には，10ビットで表現できる式を１つだけ記述することができます。式の評価値が11ビットを越えた場合は，Vエラー（不正な値）が発生します。また，オペランド欄に式を複数記述した場合は，Oエラー（オペランドの数が不正）が発生します。

（２）オペランド欄にシンボルを記述し，記述されたシンボルが未定義の場合，Sエラー（未定義シンボル）が発生します。

（３）オペランド欄に記述した式が適当でない場合，オブジェクト・コードとしてNOP（E0E0H）が生成されます。

注意　DT命令は，タイマ用テーブル・エリア参照（MOV T,@R0）用にオブジェクト・コードを変換しますので，通常のテーブル参照命令には使用しないでください（通常のテーブル参照命令にはDW命令を使用してください）。

[使 用 例]

```
E.      LOC.    OBJ.        M  I    SOURCE STATEMENT
                                    ;** TIME DATA **
                F8F7                DT      21FH            ;CARRY  ON
                F1F7                DT      05FH            ;CARRY  OFF
```

IF	IF	IF
ELSE	ELSE	ELSE
ENDIF	ENDIF	ENDIF

シンボル欄	ニモニック欄	オペランド欄	コメント欄
[レーベル：]	IF	＜式＞	[；コメント]
[ステートメント			]
	[ELSE]		[；コメント]
[ステートメント			]
	ENDIF		[；コメント]

[機　　能]

IF文のオペランド欄の評価値が“０”（偽）以外の値の場合は，IFとELSEで囲まれたステートメントがアセンブルの対象となり，ELSEとENDIFで囲まれたステートメントはアセンブルされません。

IF文のオペランド欄の評価値が“０”（偽）の場合は，IFとELSEで囲まれたステートメントはアセンブルされず，ELSEとENDIFで囲まれたステートメントがアセンブルの対象となります。

[用　　途]

プログラム内の任意のルーチンでその使用条件に応じて展開するステートメントを選択する場合に用います。

[説　　明]

（１）IFとそれに対応するENDIFの間に含まれるすべてのステートメントは，IF～ENDIFブロックとして定義されます。

（２）ELSEはオプションですので，必ずしも指定する必要はありません。ただし，指定する場合には，IF～ENDIFブロックに対して１つだけしか使用できません。１つのIF～ENDIFブロックに対してELSEを複数個指定すると２つめのELSEからSエラー（構文誤り）が発生します。

（３）IF文のオペランド欄にシンボルを記述する場合，それはすでに定義されているものでなければなりません。

（４）ネスティングはマクロ参照文，REPT文を含め，最大40レベルまで可能です。

（５）ELSEとENDIF文にはレーベルを記述することはできません。

[使 用 例]

```
IF      ZZZ0    EQ  0
        NOP
        HALT    #3H
ELSE
        NOP
        HALT    #ZZZ0
ENDIF
```

REPT　　REPEAT　　REPT
ENDR　　END REPEAT　　ENDR

シンボル欄	ニモニック欄	オペランド欄	コメント欄
[レーベル：]	REPT	＜式（EQU型）＞	[；コメント]
[ステートメント			]
	[EXITR]		[；コメント]
[ステートメント			]
	ENDR		[；コメント]

[機　　能]

REPTとENDRで囲まれたステートメントを＜式（EQU型）＞の評価値の回数だけ繰り返し展開します。

REPTとENDRの途中でEXITRが現れると展開を終了し，ENDRの次のステートメントからアセンブルを行います。

[用　　途]

同じステートメントを繰り返す場合に使用します。

ディバグ中，一時的に繰り返し疑似命令を禁止したり，途中で中断したいときにEXITRを挿入します。

[説　　明]

（１）ネスティングは最大８レベルまで可能で，マクロ参照文，IF文を含め最大40レベルまで可能です。

（２）＜式（EQU型）＞中にシンボルを記述する場合，それはすでに定義されていなければなりません。未定義もしくは前頁で定義している場合は，Sエラー（Undefined Symbol）が発生します。

（３）REPT～ENDRブロックに記述した疑似命令のオペランドに記述するシンボルは，すでに定義されている必要があります。ソース・プログラムの前方で定義されていたり，未定義のシンボルが記述されるとＳエラー（Undefined Symbol）が発生します。

（４）REPTに対するENDRがない場合は，モジュールの最後のEND疑似命令でPエラー（No ENDR Statement）が発生します。

[使 用 例]

```
REPT        3       ;“DW  0”命令を３回繰り返します。
DW          0
ENDP
```

EXITR　　EXIT REPEAT　　EXITR

シンボル欄	ニモニック欄	オペランド欄	コメント欄
	EXITR		[；コメント]

[機　　能]

REPT文中にEXITRが現れると展開を終了し，ENDRの次のステートメントからアセンブルを行います。

[説　　明]

（１）REPT～ENDR中でのみ，有効な疑似命令です。

（２）上記のブロック内以外の箇所でEXITRを記述するとＰエラー（不正なEXITR文）が発生します。

MACRO	MACRO	MACRO
ENDM	END MACRO	ENDM

シンボル欄	ニモニック欄	オペランド欄	コメント欄
ネーム	MACRO	＜仮パラメータ群＞	［；コメント］
［ステートメント（マクロ・ボディ）			］
	ENDM		

［機　　能］

MACROとENDMとの間にある一連のステートメント（マクロ・ボディ）に対してネームで示されるマクロ名を割り付けます。

ネームはマクロ参照時の定義名として使用されます。

［用　　途］

マクロ定義するときに用います。

［説　　明］

（１）マクロ・ボディ

マクロ・ボディは，シンボル，命令，疑似命令（MACRO, ENDMは除く），コメント，自分自身を含む他のマクロ文で構成されます。

（２）仮パラメータ群

・仮パラメータは，コンマ（，）で区切って32個，253文字まで記述することができます。

・仮パラメータはマクロ・ボディ内でのみ有効です。

・マクロ・ボディ内に記述した仮パラメータには，そのマクロが参照されたときに実パラメータが代入されます。

・仮パラメータは，シンボル欄，ニモニック欄，オペランド欄のいずれにも記述することができます。

（３）マクロ・ボディ内でセミコロン（；）を２つ続けた場合，そのあとの文字列はマクロ内コメントとして処理され，マクロ参照時には展開されません。

MACRO	MACRO	MACRO
ENDM	END MACRO	ENDM

[使用例１] パラメータを持たないマクロ

```
ADDR01      MACRO                   ;マクロ定義
            MOV         A,R01
            INC         A
            MOV         R01,A
            ENDM
```

[使用例２] パラメータを持つマクロ

```
ADDRNO      MACRO       RNO         ;マクロ定義
            MOV         A,RNO
            INC         A
            MOV         RNO,A       ;;RNO+1
            ENDM
            ADDRNO      R10         ;マクロ参照
                ↓(展開)
            MOV         A,R10
            INC         A
            MOV         R10,A
```

[解　　説]

このようにオペランド欄にパラメータ記述されているマクロは，マクロ参照時のパラメータに置き換えることができます。マクロ定義文のパラメータを仮パラメータといいます。

仮パラメータRNOの位置にR10が代入されています。

“ ；；”は，マクロ内コメントですので，参照時には展開されません。

GLOBAL　　GLOBAL　　GLOBAL

シンボル欄	ニモニック欄	オペランド欄	コメント欄
［レーベル：］	GLOBAL	＜シンボル群＞	［；コメント］

［機　　能］

マクロ内で使用されているシンボルをマクロ外で参照可能なシンボルとして宣言します。

［用　　途］

マクロ内で使用されているシンボルをマクロ外で使用するには，GLOBAL疑似命令を使用します。

［説　　明］

（１）GLOBAL疑似命令は，マクロ定義内（MACRO～ENDMで囲まれたブロック）にしか使用できません。もし，マクロ定義外で使用すると，Mエラー（Invalid Mnemonic）が発生します。

（２）グローバル宣言はそのシンボルが定義されるより前に，記述しなければなりません。もし，GLOBAL宣言の方があとにあるとSエラー（Symbol Multi Defined）が発生します。

（３）グローバル宣言されたシンボルの有効範囲は，同一ソース・モジュール・プログラム内です。

（４）GLOBAL疑似命令のオペランドには，１行以内（最大255文字）ならば，シンボル名を複数指定できます。

ただし，255文字を越えるとＳエラー（Syntax Error）が発生し，そのステートメントは無効となります。

［使 用 例］

```
OBJ.        M  I    SOURCE  STATEMENT
                    STMAC   MACRO               ;マクロ定義
                            GLOBAL  SYMA        ;グローバル宣言
                    SYMA    SET     00H
0000                        DW      SYMA
                            ENDM
                    ;
                    STMAC                       ;マクロ参照
                    ;
0000                DW      SYMA                ;ローカル・シンボルをマクロ外で参照
```

［解　　説］

マクロ内でグローバル宣言されたシンボルは，マクロ展開を終了しても値はそのまま使用できます。

END　　END　　END

シンボル欄	ニモニック欄	オペランド欄	コメント欄
[レーベル：]	END		

[機　　能]

ソース・（プログラム・）モジュールの終了をアセンブラに指示します。

[用　　途]

ソース・（プログラム・）モジュールの最終行に記述します。

[説　　明]

（１）END疑似命令のあとにLFコード（８ビットJISコード；0AH）がないとエラーになります。

スクリーン・エディタを使用した場合，LFコードがなくても登録することができますので注意してください。

（２）ENDのあとにコメントCR/LFコード以外の記述があると，ワーニング・メッセージが発生します。ただし，それらのステートメントは無視されます。

（３）ソース・ファイルの最後まで読み込んで，END文が記述されていなかったり，END文の次にCRコードがないなどの理由で，アセンブラがEND疑似命令を認識しない場合，Pエラー（END文がない）が発生します。ただし，ファイルの最後にEND文があると仮定し，オブジェクト・ファイルを生成します。

[使 用 例]

```
        .
        .
        .
        .
END
```

[解　　説]

上記は，ソース・プログラム・モジュールの最終行にあるEND疑似命令を示します。

OPTION
ENDOP

シンボル欄	ニモニック欄	オペランド欄	コメント欄
[レーベル：]	OPTION ENDOP	マスク・オプション疑似命令	[；コメント]

[機　　能]

OPTIONとENDOPで囲まれたブロックを，マスク・オプション定義ブロックと呼びます。マスク・オプション定義ブロックには，マスク・オプション疑似命令を記述することができます。マスク・オプション疑似命令は，各デバイスごとに異なります。

[説　　明]

（１）OPTION疑似命令は，ENDOP疑似命令で終わる必要があります。OPTION疑似命令とENDOP疑似命令の間にEND疑似命令等があるとPエラー（No OPTION Directive）が発生します。

（２）OPTION疑似命令と，ENDOP疑似命令の間にオブジェクトを生成する命令があるとワーニングが発生します。このときこの間に記述されたオブジェクトは生成されません。

（３）OPTION疑似命令とENDOP疑似命令は，１つのソース・プログラム中に一度だけ記述することができます。二度以上記述された場合は，後に記述されたOPTION疑似命令に対してPエラー（Duplicated OPTION Diretive）が発生します。このときこの間に記述されたオブジェクトは生成されません。

また，OPTION疑似命令，ENDOP疑似命令は，２つのモジュールにまたがって記述することはできません。

（４）マスク・オプションが必要なデバイスのソース・プログラム中にOPTION疑似命令が記述されていない場合はリンク時にOエラー（Not Found Mask Option Block）が発生します。

[使 用 例]

```
OPTION          ;低電圧検出回路内蔵
USEPOC
ENDOP
```

★ USEPOC　　USEPOC　　USEPOC
NOUSEPOC　　NOUSEPOC　　NOUSEPOC

シンボル欄	ニモニック欄	オペランド欄	コメント欄
		USEPOC	[；コメント]
		NOUSEPOC	

[機　　能]

マスク・オプションの低電圧検出回路あり／なしを指定します。

USEPOCが低電圧検出回路あり，NOUSEPOCが低電圧検出回路なしを示します。

[注意事項]

USEPOC, NOUSECOPのいずれかを指定しなければ，エラーとなります。

★ USECAP　　USECAP　　USECAP
NOUSECAP　　NOUSECAP　　NOUSECAP

シンボル欄	ニモニック欄	オペランド欄	コメント欄
		USECAP	[；コメント]
		NOUSECAP	

[機　　能]

マスク・オプションの発振器用コンデンサのあり／なしを指定します。

USECAPが発振器用のコンデンサあり，NOUSECAPが発振器用のコンデンサなしを示します。

[注意事項]

USECAP, NOUSECAPのいずれかを指定しなければ，エラーとなります。

このマスク・オプションは，D67, D68, D69の製品のみ指定できます。

TITLE　　TITLE　　TITLE

シンボル欄	ニモニック欄	オペランド欄	コメント欄
[レーベル：]	TITLE	'文字列 '	[；コメント]

[機　　能]

アセンブル・リストの改ページを行い，オペランド欄の文字列をアセンブル・リストのヘッダ・ラインに出力します。

[用　　途]

アセンブル・リストにタイトルを印字することにより，リストを見やすくするために使用します。

[説　　明]

（１）文字列は最大78文字（８ビットJISコード換算）まで記述することができます。なお，79文字以上記述するとIエラー（不正なデータ長）が発生します。

（２）TITLE制御命令が現れるとアセンブラは改ページを行い，指定したタイトル（文字）をヘッダに印字します。ただしTITLE制御命令が１行目にある場合は改ページは行いません。またTITLE制御命令自身は改ページ後の先頭の行に出力されます。

（３）引用符（ ' ）で狭まないで，文字列を記述した場合，Sエラー（構文誤り）が発生します。

[使 用 例]

ソース・プログラム・リスト

```
          .
          .
          .
        TITLE         'SUBROUTINE'
          .
          .
```

EJECT　　EJECT　　EJECT

シンボル欄	ニモニック欄	オペランド欄	コメント欄
[レーベル：]	EJECT		[；コメント]

[機　　能]

アセンブル・リストの改ページを行います。

[用　　途]

ルーチンの切れ目で強制的に改ページするときに使用します。改ページすることによりアセンブル・リストが見やすくなります。

[説　　明]

（１）EJECT制御命令が現れると，アセンブラは改ページを行います。

（２）EJECT制御命令自身の文字列は改ページ前のページに印字されます。

[使 用 例]

ソース・プログラム・リスト

```
                .
                .
                JMP         ABC
                                        EJECT
        DEF:
                .
                .
```

INCLUDE　　INCLUDE　　INCLUDE

シンボル欄	ニモニック欄	オペランド欄	コメント欄
[レーベル：]	INCLUDE	'ファイル名 '	[；コメント]

（ファイル名の規則については「はじめに」を参照してください）

[機　　能]

ファイル名で指定されたソース・プログラムを読み込んで，ソース・プログラムの一部として処理します。

[用　　途]

分割された他のファイルを組み込むときに用います。

[説　　明]

（１）INCLUDEで指定したソース・モジュール中に，INCLUDE文を含むことができます。INCLUDEのネスティングは８レベルまで可能です。９レベル以上設定されている場合は，ネスト・オーバフローのエラーになります。

（２）INCLUDE制御命令で指定したファイルの終了は，EOF文でなければなりません。EOFが記述されていない場合はワーニングが発生します。

（３）ファイル名で拡張子が指定されない場合は，ASMが拡張子とされます。

（４）INCLUDE制御命令で結合されるファイルは分割モジュールではないので，元のソース・プログラム内のシンボルをそのまま参照することができます。

（５）ファイル名を引用符で囲まないで記述した場合は，Sエラー（構文誤り）が発生し無効となります。

（６）ファイル名にはパス名が使用できます（ファイル名には最大64文字記述することができます）。

（７）ファイル名で指定したファイルが存在しない場合は，Fエラー（インクルード・ファイルがオープンできない）が発生します。

INCLUDE　　INCLUDE　　INCLUDE

［使用例１］

ソース・プログラム

```
      ・
      ・
      ・
INCLUDE          'SUB1.ASM'
      ・
      ・
      ・
INCLUDE          'SUB2.ASM'
      ・
      ・
      ・
   END
```

SUB1.ASM

```
      ・
      ・
      ・
      ・
```

SUB2.ASM

```
      ・
      ・
      ・
      ・
```

INCLUDE　　INCLUDE　　INCLUDE

［使用例２］

ソース・モジュールＡ

```
INCLUDE          'MACROFILE.ASM'
     .
     .
     .
   END
```

ソース・モジュールＢ

```
INCLUDE          'MACROFILE.ASM'
     .
     .
     .
   END
```

１ソース・モジュール中に記述できるINCLUDEファイルは，16ファイルまでです。

１ソース・モジュール中に記述できるINCLUDEファイル名の総文字数は255までです。

注意　/HOSTオプション指定時は，ソース・ファイル名にドライブ名，ディレクトリ名を記述することはできません。

INCLUDE　　INCLUDE　　INCLUDE

MACROFILE. ASM

```
MAC1        MACRO       A1,A2
              .
              .
              .
            ENDM
MAC2        MACRO       B1,B2
              .
              .
              .
            ENDM
              .
              .
              .
```

［解　　説］

複数のモジュールで使用するマクロのみを１つのファイルにまとめます。そしてINCLUDE制御命令を用いてそのファイルを組み込めば，PUBLICやEXTRN疑似命令を用いることなく，複数のソース・モジュールで共通にマクロを使用することができ便利です。もしPUBLICやEXTRN疑似命令を使用すると，使用するモジュールごとに使用するマクロ名を宣言する必要があります。

３.５　マクロ機能

プログラム中で，同一のルーチンを何回も使用する場合は，一般にサブルーチン化により，プログラム・ステップ数を節減する方法が用いられます。サブルーチン化できない似通った処理ルーチンで，パラメータが異なる場合，効率よくプログラミングするためにマクロ機能が用いられます。

３.５.１　マクロの定義と適用範囲

（１）マクロの定義

マクロを定義するためには，マクロ定義疑似命令（MACRO, ENDM）を使用します。

また，マクロを定義する場合に仮パラメータを用いて記述することができます。

マクロ定義疑似命令については，**表３－１　疑似命令，制御命令一覧**で参照してください。

（２）マクロの適用範囲

マクロ内で定義するシンボルには，マクロ内でのみ有効なローカル・シンボルと，そのマクロ以外のルーチンでも有効となるグローバル・シンボルの２種類があります。

グローバル・シンボルにするには，GLOBAL疑似命令を用いてマクロ内でグローバル宣言をする必要があります。グローバル宣言されないシンボルは，マクロ内のみ有効なローカル・シンボルとして扱われます。GLOBAL疑似命令については，**表３－１　疑似命令，制御命令一覧**で参照してください。

マクロは，ある一連のブロックにその手続きの内容が分かるような意味のある名前を付けることにより，プログラムの読みやすさを向上させることができます。また，マクロ定義文だけを格納した独立のファイルを作成し，ソース・プログラムの先頭でINCLUDE文によりファイルの内容を読み込んで使用するといった，いわばライブラリ的な活用の方法もあります。

・ローカル・シンボル

マクロ内で定義したシンボルは，特に宣言をしないと，そのシンボルを定義しているマクロ内でのみ有効なローカル・シンボルとなります。これにはマクロ内のマクロ参照文や，マクロ内のINCLUDE文も含まれています。したがって，同じ名前のシンボルをマクロの外で定義したり，同じマクロが２回以上参照されて，同じようなステートメントが生成されても，二重定義にはなりません。

・グローバル・シンボル

マクロ内で定義したシンボルをマクロ外で参照したい場合があります。このときはシンボルのグローバル宣言を行い，そのシンボルを同一モジュール内のすべてのステートメントからの参照を可能にします（シンボルのグローバル宣言の方法，使用例については**表３－１　疑似命令，制御命令一覧のGLOBAL疑似命令**で参照してください）。

しかし，SET疑似命令以外で定義されたシンボルが固定的にグローバル宣言されているマクロを２回以上参照し，一連のステートメントが生成されると，そのシンボルは二重定義エラーとなるので注意してください。

また，マクロ外で値がSET疑似命令により定義されたシンボルと同一のシンボルをマクロ内でセットした場合，そのシンボルはマクロ内ではローカル・シンボルとなり，マクロ外の同一名のシンボルとは関連がなくなります。もし，マクロ内でマクロ外のシンボルに値をセットしたい場合は，グローバル宣言する必要があります。

（3）マクロの使い方

マクロを使うためには，定義と参照という手続きが必要です。一連の命令，および疑似命令をマクロ名に割り当てることを「マクロ定義」といいます。

［ 例１ ］

```
ADDR01      MACRO                   ;マクロ定義
            MOV         A,R01
            INC         A
            MOV         R01,A
            ENDM
```

この例ではマクロ名ADDR01に

```
MOV         A,R01
INC         A
MOV         R01,A
```

という３つの命令を割り当てています。マクロ名は任意の名前で記述することができますが，ほかのシンボル名や予約語と同じ名前は使用できません。

すでに定義したマクロは，マクロ定義以降ならば同一モジュール内から何度でも使用することができます。

マクロ名を記述してマクロ定義した内容を使用することを「マクロ参照」といいます。

マクロ参照文は，ニモニック欄に記述します。

マクロが参照されると，アセンブラはマクロに割り付けられた一連の命令，および疑似命令を定義された順序とおりに展開します。これを「マクロ展開」といいます。

［ 例２ ］

```
ADDR01              ;マクロ参照
  ↓(展開)
MOV   A,R01
INC   A
MOV   R01,A
```

マクロ関連の疑似命令には次のものがあります。

MACRO～ENDM
GLOBAL
REPT～EXITR～ENDR

３.５.２　マクロの参照

[機　　能]

MACRO文とENDM文で定義したマクロ・ボディを参照します。

[記述形式]

シンボル	ニモニック	オペランド	コメント
［レーベル：］	ネーム	＜実パラメータ群＞	［；コメント］

[説　　明]

（１）ネームはMACRO文のシンボル欄に記述した“マクロ・ネーム”であり，参照する以前に定義されているものでなければいけません。

（２）実パラメータとして記述できる形式は次の５種類で，16ビットのデータとして評価されます。

（ａ）式
（ｂ）文字定数（引用符で囲んだ８ビットJISコード，またはシフト8JISコード例）
（ｃ）スペース，または空き（記述なし，コンマのみ）
（ｄ）シンボル
（ｅ）定数

（３）仮パラメータから実パラメータへの置き換えは，それぞれの記述順に対応させて左から順に行われます。ただし，実パラメータ数＞仮パラメータ数の場合，Ｏエラー（Operand count error）となります。

実パラメータ数＜仮パラメータ数の場合，対応しない残りの仮パラメータにはNULLコードが割り付けられ，マクロ参照時にはエラーは発生しません。ただし，マクロ展開時にNULLコードによって，エラーが発生する場合があります。

（４）実パラメータとして空白，コンマ，引用符，セミコロン，タブなどを記述する場合には，文字列として引用符で囲まなければいけません。

（５）マクロ・ボディにマクロ参照文を記述することができます。ただし，ネスティングは繰り返し疑似命令，マクロ参照文，IF文を含め最大40レベルまで可能です。これを越えた場合にはＮエラー（Nesting overflow）が発生し，アセンブルの対象とはなりません。もしくは，Ｍエラー（Macro area overflow）が発生し，マクロは展開されません。

[　例　]

```
ADMAC       2,5
```

“ADMAC”は，マクロ定義疑似命令で定義されたマクロのネームで２，５は“ADMAC”を参照する際，必要となる実パラメータです。

３.５.３　マクロの展開

マクロを使用したソース・プログラムは，次の順序でアセンブルされます。

① マクロ定義があるとアセンブラの内部のメモリ領域にマクロ・ボディをそのままストアします（マクロ登録）。

② 次にマクロ参照を見つけると，それに対応するマクロ・ボディをシンボル・テーブルより探しだしてマクロ・ネームの位置に挿入します。

③ 展開したプログラムをアセンブルします。ただし，マクロ・ボディ内でダブルセミコロン（；；）が記述されている場合，“；；”のあとから，その行の終わりまでをマクロ定義内コメントとみなし，マクロ参照時には展開されません。

[説　　明]

（１）マクロ展開は，モジュールのアセンブル・フェーズのパス１で行われます。

（２）マクロ内に記述した疑似命令のオペランドでマクロ外定義シンボルを参照する場合は，マクロ参照より先に定義されている必要があります。定義されていない，もしくはマクロの参照よりあとに定義されたシンボルをマクロ内に記述すると，Sエラー（Symbol undefined）が発生します。

[　例　]

```
HTIMER      MACRO       TIMEVAL,HALTVAL     ┐
            MOV         T,#TIMEVAL          │  ①
            HALT        #HALTVAL            │
            ENDM                            ┘

            HTIMER      100H,0101B          —  ②
```

①：“HTIMER”というネームのマクロを定義します。
“TIMEVAL”，“HALTVAL”は仮パラメータです。

②：“HTIMER”というネームのマクロを参照します。
“100H，0101B”は実パラメータで仮パラメータの“TIMEVAL, HALTVAL”に対応しています。

HTIMERを参照した結果，次のように展開されます。

```
MOV         T,#100H
HALT        #0101B
```

第 Ⅱ 編
操 作 編

第１章　製品概要

１.１　製品内容

プログラム名	ファイル名	ファイルの種類
アセンブラ	AS6133. EXE	コマンド・ファイル

コマンド・ファイルとは，プログラムが起動したとき最初にメモリ内に読み込まれるファイルです。

１.２　対応ディバッガ

AS6133アセンブラを使用する場合，ディバッガは次の製品を使用してください。当社製のSM6133 V1.02，V1.06は使用できません。

メーカ名：株式会社　内藤電誠町田製作所
製品名　：EB-6133エミュレータ

１.３　システム構成

本節ではAS6133を動作させるために必要な動作環境について説明します。

（１）ホスト・マシン

使用できるパーソナル・コンピュータは，“ はじめに ” を参照してください。

（２）オペレーティング・システム

使用できるオペレーティング・システムは，“ はじめに ” を参照してください。

（３）ユーザ・メモリ・サイズ

512 Kバイト以上

（４）AS6133起動時必要となるファイル

１．ソース・ファイル（××××. ASM ）

アセンブルの対象となるファイルです。

２．シーケンス・ファイル（××××. SEQ）

アセンブラ起動時に，デバイス・ファイル名，アセンブル・オプション，ソース・ファイル名を指定するための情報を格納したファイルです。

複数のソース・モジュール・ファイルをアセンブルする場合には，あらかじめシーケンス・ファイル内にすべてのソース・ファイル名を指定しなければなりません。

３．MS-DOSの環境設定ファイル（CONFIG. SYS）

設定値：files = 15（15以上にする）

buffers = 10（10以上にする）

第２章　実行の前に

２.１　バックアップ・ファイルの作成

AS6133を実際のアセンブルに使用する場合は，万一フロッピィ・ディスク本体やその内容が破壊された場合に備え，オリジナル・ディスクの内容を作業用ディスクにコピーしたあとに使用してください。

オリジナル・ディスクは大切に保管してください。

バックアップ・ファイルの作成手順

１．MS-DOSを起動します。

２．ドライブＡにMS-DOSのシステム・ディスクを，ドライブＢに新しいフロッピィ・ディスクをセットします。

３．FORMATコマンドによりドライブＢのフロッピィ・ディスクをフォーマッティングし，システムをコピーします。

```
A>FORMAT B:/S↵
A>
```

４．次にドライブＡにAS6133のオリジナル・ディスクをセットします。COPYコマンドでドライブＡのAS6133.EXEをドライブＢへ転送します。

```
A>COPY A:*.* B:/V↵
A>
```

５．以上でドライブＢへドライブＡの内容がすべて転送されました。

```
A>DIR B:/W↵
    AS6133.EXE
A>
```

２.２　インストール

（１）出荷媒体中のファイル（AS6133.EXE）をインストール先にコピーします。

たとえば，出荷媒体をセットしたフロッピィ・ディスク・ドライブがA:，インストール先がC:\nectools\binの場合，次のようにcopyコマンドを実行します。

```
X> copy A:\*.*   C:\nectools\bin
```

（２）環境変数PATHに，インストール先のディレクトリを追加します。

上の例では，次の行をAUTOEXEC.BATに追加します。

```
PATH   C:\nectools\bin;%PATH%
```

第３章　シーケンス・ファイル

３.１　概　　要

アセンブラを起動し，アセンブル処理を実行する際には，アセンブル対象となるデバイス・ファイル，ソース・モジュール・ファイル，アセンブル・オプション[注]（これらを総称しアセンブル条件という）を指定します。

アセンブル条件の指定は，シーケンス・ファイルにより行います。

シーケンス・ファイルでアセンブル条件を指定すると，多くのアセンブル条件の指定が１つのシーケンス・ファイル名の指定だけで行えます。

また，ディバグ中にソース・モジュール・ファイルを削除したり，追加したりする場合に，シーケンス・ファイル内の記述を変更するだけで容易に実行できます。

上記に説明したように，シーケンス・ファイルを有効に活用することにより，ディバグの効率化を実現することができます。

注　アセンブル・オプションとは，アセンブル実行時にリスト出力の有無などを制御するものです。詳細は**４.５　アセンブル・オプション**を参照してください。

3.2　シーケンス・ファイルの記述形式

シーケンス・ファイルは，エディタまたはCOPYコマンドを用いて記述します。

シーケンス・ファイル名の拡張子は必ず“.SEQ”にしてください。

3.2.1　全体の記述形式

```
[;コメント]
        デバイス名                                          [;コメント]      ;①
        ／オプション［／オプション／オプション／…／…／……］  [;コメント]      ;②
        ソース・ファイル名                                  [;コメント]
                ・                                            ・
                ・                                            ・          ③
                ・                                            ・
        ソース・ファイル名                                  [;コメント]
```

[説　明]

（1）①には，デバイス名指定を行います。

（2）②には，アセンブル・オプションの指定を行います。半角のスラッシュ（/）とスラッシュの間にはアセンブル・オプションを１項目のみ指定してください。

　複数のアセンブル・オプションを指定する場合は，スラッシュで区切りながら連続して記述します。アセンブル・オプション指定が２行以上となる場合は，２行目以降の行の先頭にもスラッシュ記号を記述してください。

　②で記述したアセンブル・オプションは，全ソース・ファイルのアセンブルに対して有効となります。

（3）③には，ソース・モジュール・ファイル名を指定します。

（4）シーケンス・ファイル内でコメントを記述する場合は，ソース・プログラムと同様に半角のセミコロン（;）を使用してください。コメント文はファイルのどの位置でも記述可能です。

（5）記述は，デバイス名，アセンブル・オプション，ソース・ファイル名の順番で行ってください。この順番以外で記述を行うと，エラーが発生します。

3.2.2　デバイス名の記述形式

```
[；コメント]
    デバイス名          [；コメント]
```

[機　　能]

アセンブル対象とする製品のデバイス名を指定します。

[説　　明]

（1）デバイス名は，シーケンス・ファイルの先頭に記述してください。

ただしコメント文は，デバイス名より前の行に記述することができます。

（2）拡張子はありません。

指定されたデバイス名以外を記述した場合は，アセンブル実行時に以下のようなエラーが発生し，アセンブルの実行を中止します。

```
NOT FOUND DEVICE STATEMENT
```

規定外の場所に記述した場合は，アセンブル実行時にエラーが発生します。

★　（３）シーケンス・ファイル中で記述可能なデバイス名とデバイスとの対応を，表３－１に示します。

表３－１　記述可能なデバイス名と対応デバイス

デバイス名	対応デバイス
D6133	μPD6133
D6134	μPD6134
D6135	μPD6135
D6604	μPD6604
D6605	μPD6605
D63	μPD63
D63A	μPD63A
D64	μPD64
D64A	μPD64A
D62	μPD62
D62A	μPD62A
D65	μPD65
D6132	μPD6132
D6132A	μPD6132
D67	μPD67
D68	μPD68
D69	μPD69

注意　μPD61P34B, 66P04B, 6P4B, 6P5, 6P9の場合は，対応するROM版デバイスのデバイス名を使用してください。

[　例　]　μPD6133をアセンブル対象とする場合

```
D6133        ;μPD6133
```

注意　必ず“μP”を省略した製品名で記述してください。

3.2.3　アセンブル・オプションの記述形式

```
［／オプション］［／オプション］［／……］［／オプション］
［／オプション］［／……］［／オプション］　［：コメント］
```

[機　　能]

アセンブル・オプションを指定します。

[説　　明]

（1）アセンブル・オプションの指定は，デバイス・ファイル名の次の行より記述してください。ただし，コメント行が間に入ってもかまいません。

（2）アセンブル・オプションの記述はスラッシュ（／）で始まります。

（3）複数のアセンブル・オプションを指定する場合は，オプションごとにスラッシュ（／）で区切ってください。また，オプションとオプションの間にブランクを入れてもかまいません。

（4）アセンブル・オプションの指定は2行以上にわたって記述することができます。この場合，その行の最後にCR/LFコードを入力し次の行の先頭にスラッシュ（／）を記述します。

（5）相反するアセンブル・オプションを指定した場合は，あとで記述されたアセンブル・オプションが有効となります。

（6）アセンブル・オプション指定は省略することができます。

（7）アセンブル・オプションの詳細については，**4.5　アセンブル・オプション**を参照してください。

規定外の場所に記述した場合は，アセンブル実行時にエラーが発生します。

3.2.4　ソース・ファイル名の記述形式

```
ソース・ファイル名　［；コメント］
ソース・ファイル名　［；コメント］
　　：
ソース・ファイル名　［；コメント］
```

[機　　能]

アセンブル対象となるソース・ファイル名を指定します。

[説　　明]

1行に2つ以上のソース・ファイル名を記述することはできません。

注意　/HOSTオプション指定時は，ソース・ファイル名にドライブ名，ディレクトリ名を記述することはできません。

3.3　シーケンス・ファイルの記述例

以下にシーケンス・ファイル（SAMPLE. SEQ）の記述例を示します。

```
;DEVICE  NAME
            D6134           ;     ①
;OPTION
            /HOST                 ②
            /WORK=B:
;SOURCE  MODULE
            INIT.ASM
            MAIN.ASM
            SUB1.ASM              ③
            SUB2.ASM
            DATA.TBL
```

[説　　明]

①はアセンブル対象となるデバイス名の指定です。

②はアセンブル・オプションの指定です。

③はアセンブル対象となるソース・モジュールです。

シーケンス・ファイルの作成方法には，MS-DOS上で動作するエディタを使用する方法と，MS-DOSのシステム・コマンドである“ COPY ”コマンドを使用する方法とがあります。

短い内容の記述時は，COPYコマンドで十分ですが，以前に記述したシーケンス・ファイルの修正や，多量の指定を記述する場合はエディタを使用すると便利です。

注意　/HOSTオプション指定時は，ソース・ファイル名にドライブ名，ディレクトリ名を記述することはできません。

第４章　アセンブラの機能

4.1　概　　要

AS6133は，指定されたソース・モジュール・ファイルを読み込み，そのソース・モジュール・ファイルに記述されたステートメントよりオブジェクト・ファイル，アセンブル・リスト・ファイルなどを生成します。

AS6133のアセンブル方式は，２パス方式を採用しています。１パス目ではシンボル・テーブルを作成するとともに，ニモニックを機械語に変換します。シンボルは未定義のまま，領域のみを確保します。

２パス目では，１パス目で確保したシンボル領域に機械語を割り付けます。２パス終了後，中間ファイルである中間オブジェクト・モジュール・ファイルが生成されます。この中間オブジェクト・モジュール・ファイルでは，モジュール・ファイル間にまたがって分岐するアドレス情報は決定されていません。

次にAS6133は，中間オブジェクト・モジュール・ファイルをリンク処理し，オブジェクト・ファイルを生成します。リンク処理は，自動的に実行されます。

またAS6133はアセンブル処理を効率よく実行するためのアセンブル時間短縮機能を有しています。２パス終了時に生成される中間オブジェクト・モジュール・ファイルには生成された日時が登録されます。ソース・モジュール・ファイルを一部修正して再アセンブルする際，同一名の中間オブジェクト・モジュール・ファイルと，ソース・モジュール・ファイルの作成日時を比較し，ソース・モジュール・ファイルの作成日時が新しい場合のみ，そのソース・モジュール・ファイルをアセンブルします。

中間オブジェクト・モジュール・ファイルの作成日時が新しい場合は，対応するソース・モジュール・ファイルは，修正されていないと判断し，そのソース・モジュール・ファイルはアセンブルされません。この場合には，アセンブラ起動時にすでに作成されてある中間オブジェクト・モジュール・ファイルを用いて，リンク処理が実行されます。

４.２　アセンブラの入出力ファイル

AS6133の入力ファイルには以下のものがあります。

入力ファイル名	ファイルの種類	ファイル・タイプ
ソース・ファイル	エディタを用いて作成したソース・ファイルです。	.ASM
シーケンス・ファイル	デバイス名，アセンブル・オプション指定，ソース・モジュール・ファイルを登録したファイルです。 ＊シーケンス・ファイルを採用することにより，アセンブラ起動ごとにデバイス名，アセンブル・オプション，ソース・モジュール・ファイルの指定をする必要がなく効率よく，アセンブル実行ができます。	.SEQ

備考　.　（アンダ・ライン）拡張子名の変更可能なファイルです。

AS6133の出力ファイルには以下のものがあります。

出力ファイル名	ファイルの種類	ファイル・タイプ
PROMファイル	インテルHEX形式のオブジェクト・コードとIFL/DFLが格納されるファイルです。 インテルHEX形式の終了コードがIFL/DFLのあとにあります。 PROMライタに“PRO”ファイルをダウンロードすると，オブジェクト・コードとIFL/DFLを一度で書き込み処理ができます。	.PRO
アセンブル・リスト・ファイル	ソース・モジュール・ファイルごとのアセンブル・リストが格納されるファイルです。	.PRN
クロス・レファレンス・リスト・ファイル	ソース・モジュール・ファイルごとのクロスレファレンス・リストが格納されるファイルです。 ただし，リスト出力がない場合の拡張子は，XRFとなります。	.PRN
ロギング・ファイル	アセンブル処理中にコンソール上に出力されるエラー・メッセージ，ワーニング・メッセージなどの出力を格納するファイルです。 このファイル名は“AS6133.LOG”に固定されます。	.LOG
中間オブジェクト・モジュール・ファイル	アセンブル実行時にソース・ファイルごとに生成される中間ファイルです。 リンク処理時は，入力ファイルになります。	.OBJ

備考　.　（アンダ・ライン）拡張子名の変更可能なファイルです。

４.３　アセンブラの機能

４.３.１　中間オブジェクト・モジュール・ファイル出力機能

アセンブラ起動時に，指定されるソース・モジュール・ファイル“ .ASM ”を機械語に変換し，ソース・モジュール・ファイルと同一名の中間オブジェクト・モジュール・ファイル“ .OBJ ”を出力します。

また，この中間オブジェクト・モジュール・ファイルには，出力された日時が登録されます。

４.３.２　リンク機能

AS6133は，アブソリュート・アセンブラですが，モジュール分割されたソース・ファイルのアセンブルを可能とするために，リンク機能を有しています。

ソース・モジュール・ファイルをアセンブルすると，おのおののソース・モジュール・ファイルに対応した中間オブジェクト・モジュール・ファイルが出力されます。その後自動的に中間オブジェクト・モジュール・ファイルを入力ファイルとして，リンク処理が実行されます。

４.３.３　PROファイル出力機能

中間オブジェクト・モジュール・ファイルをリンク処理した結果，PROファイルを出力します。PROファイルは，オブジェクト部とIFL/DFL部が２つに分かれているファイルで，マスクROM発注用PROMのデータ・ファイルです。

詳しくは**第５章　アセンブラの出力リスト**を参照してください。

４.３.４　アセンブル時間短縮機能

AS6133は，効率のよいディバグを実現するために，アセンブル時間短縮機能を有しています。

ソース・モジュール・ファイルのアセンブルを実行する前に，ソース・モジュール・ファイルの作成された日時と同一名の中間オブジェクト・モジュール・ファイルの作成された日時とを比較します。

そして中間オブジェクト・モジュール・ファイルの作成日時が新しい場合は，同一名のソース・モジュール・ファイルは，変更されていないと判断し，そのソース・モジュール・ファイルのアセンブル処理を実行しません。

ソース・モジュール・ファイルの作成日時より，同一名の中間オブジェクト・モジュール・ファイルの作成日時が古い場合，そのソース・モジュール・ファイルはアセンブル処理されます。そしてそのソース・モジュール・ファイル以降に指定されているソース・モジュール・ファイルもすべて無条件にアセンブル処理されます。

また，ソース・モジュール・ファイルの指定の記述順序の変更追加，削除を行った場合前回アセンブル処理してから変更のあったソース・モジュール・ファイル以降のファイルを無条件にアセンブル処理します。

ディバグ済みのソース・モジュール・ファイルを先に記述し，ディバグ中のソース・モジュール・ファイルをそのあとに記述すると，アセンブル時間短縮機能を有効に使用できます。

図４－１　アセンブル時間短縮機能の処理手順（1/2）

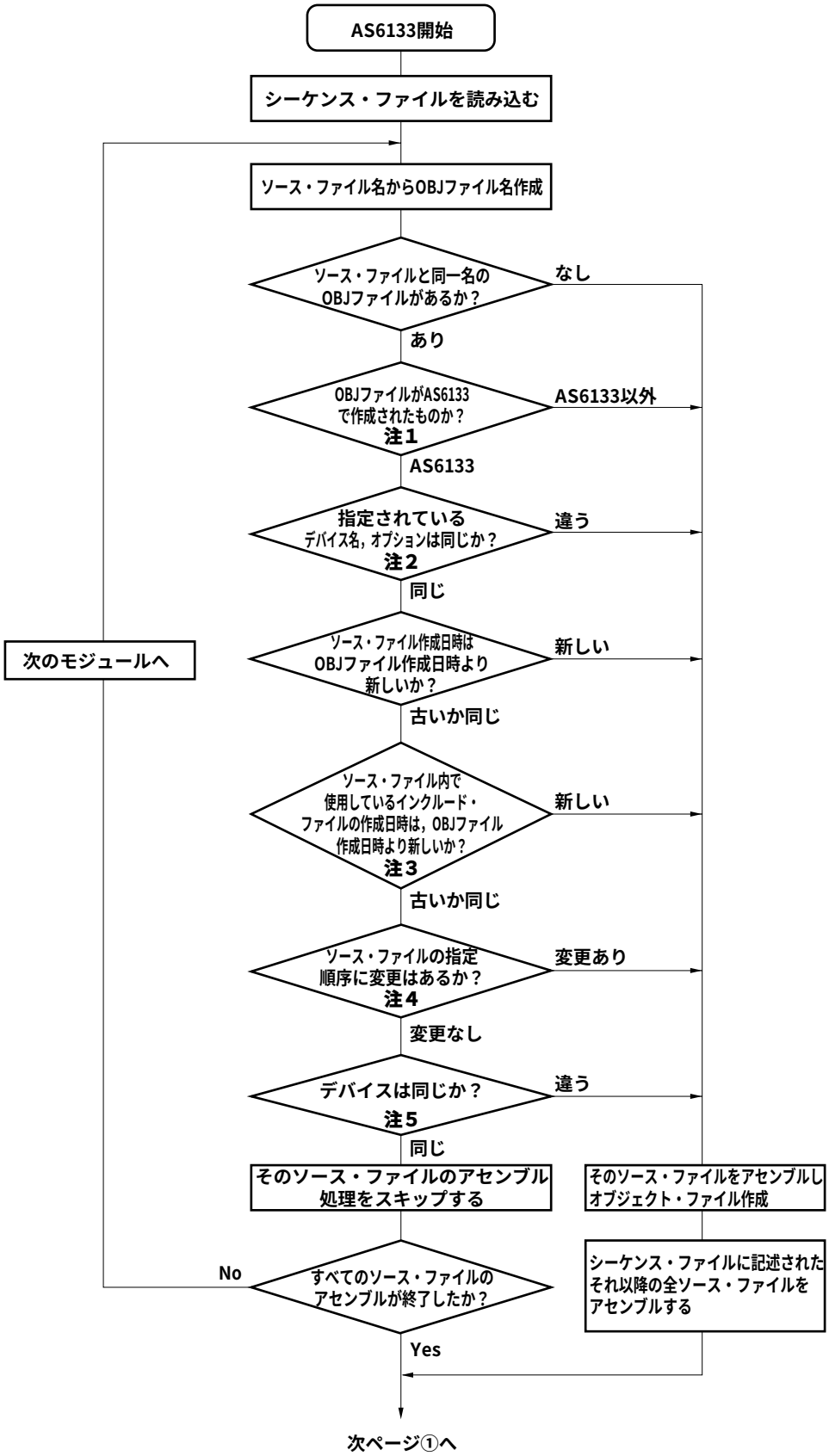
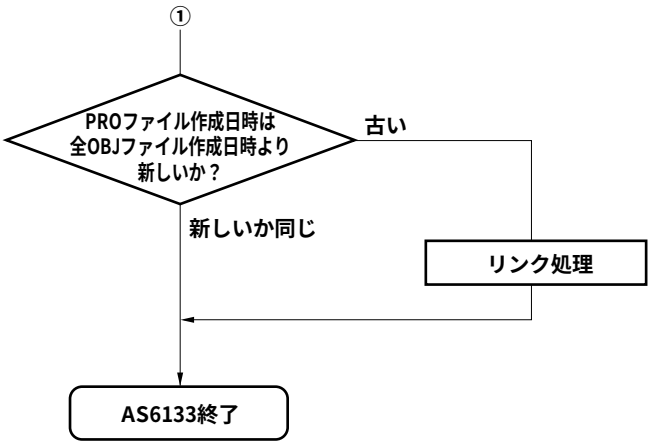


図４－１　アセンブル時間短縮機能の処理手順（2/2）



注１．AS6133で作成されたOBJファイルには，先頭に“ AS61 ”の文字列がある。

２．SEQファイルで指定されたデバイス名，オプションとOBJファイルにセットされているデバイス名，オプションをチェックする。

３．OBJファイルからインクルード・ファイル名を得る。

４．OBJファイルから直前のソース・ファイル名を得る。

５．OBJファイルからデバイス情報を得る。

４.３.５　アセンブル・リスト・ファイル出力機能

アセンブル終了後，アセンブル・リスト・ファイルが出力されます。アセンブル・リスト・ファイルはアセンブル・オプションで出力の制御を行うことができます。

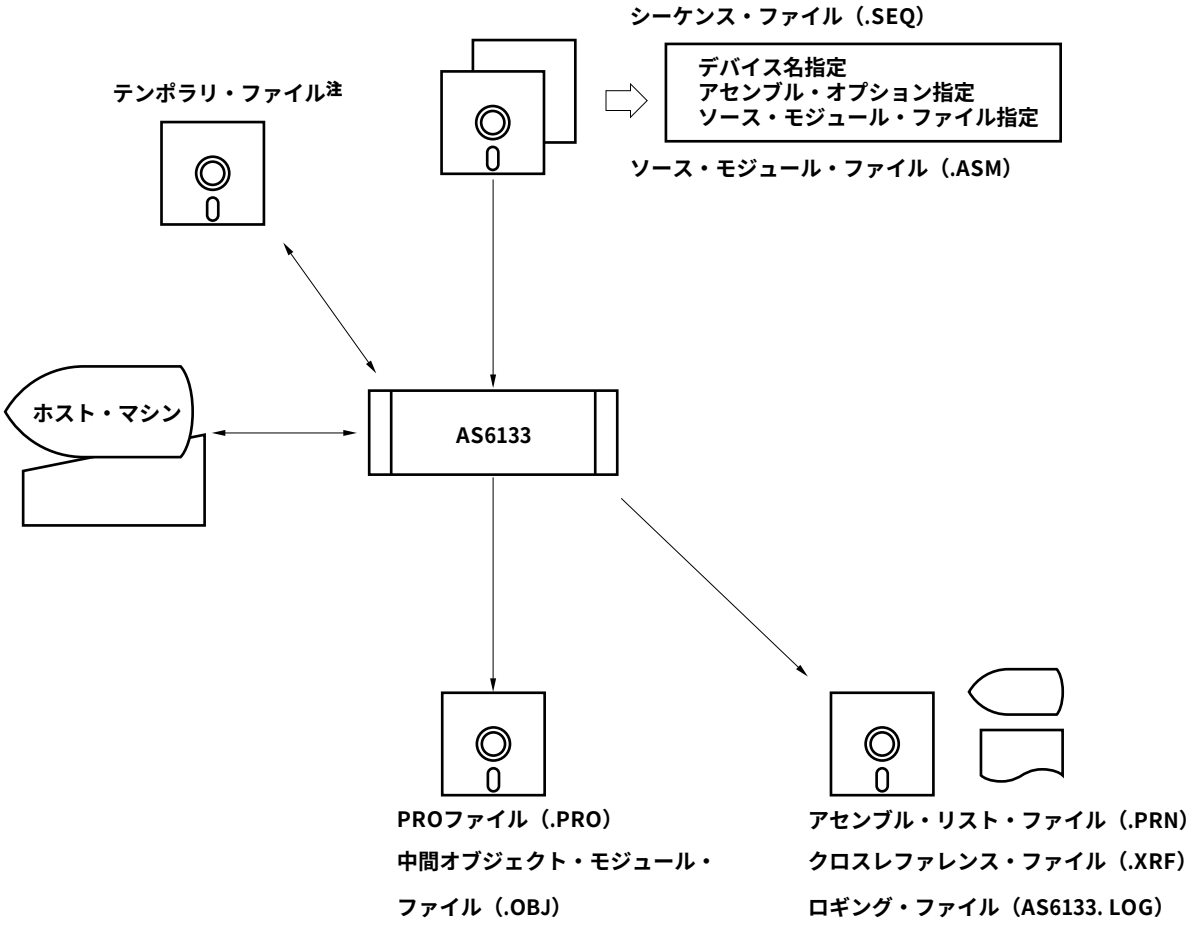
詳しくは，**第５章　アセンブラの出力リスト**を参照してください。

４.３.６　クロスレファレンス・リスト・ファイル出力機能

AS6133ではクロスレファンス・リスト・ファイルが生成されます。

詳しくは，**第５章　アセンブラの出力リスト**を参照してください。

図４－２　AS6133入出力ファイル構成



注　テンポラリ・ファイルはアセンブル終了時に消去されます。

４.４　アセンブラの起動方法

４.４.１　アセンブラ起動時必要な入力ファイル

アセンブラを起動するためには，以下の入力ファイルが必要です。

（１）シーケンス・ファイル（.SEQ）

アセンブルに必要なデバイス名とアセンブル・オプション，ソース・プログラム・ファイル名を記述したファイルです。

（２）ソース・モジュール・ファイル（.ASM）

ソース・プログラムを記述したファイルです。

詳しくは，４.２　アセンブラの入出力ファイルを参照してください。

４.４.２　アセンブラの起動方法

ここでは，アセンブラを実際に起動させる手順を説明します。

アセンブラを起動させる場合，次の２通りの入力方法があります。

入力方法

（１）

```
×>[ディレクトリ]AS6133 ↵
```

（２）

```
×>[ディレクトリ]AS6133△<シーケンス・ファイル名> ↵
```

×：カレント・ドライブ名

注意１．AS6133の［ディレクトリ］を省略するために，環境変数のPATHを設定しておいてください。

２．シーケンス・ファイルとソース・ファイルは，必ず同じディレクトリ内に入れてください。

本節では上記２通りの入力方法別にアセンブラの動作を説明します。

（１）

```
×>[ディレクトリ]AS6133 ↵
```

による起動

ドライブＡには，アセンブラを格納したディスクを，ドライブＢには，シーケンス・ファイルとソース・ファイルを格納したディスクをセットします。

プロンプトをシーケンス・ファイルとソース・ファイルの入ったドライブＢにし，“Ａ：AS6133”と入力します。

```
B>A:AS6133 ↵
```

アセンブラがメモリにロードされ動作を開始します。

アセンブラは動作開始後カレント・ディレクトリ内のシーケンス・ファイル（.SEQ）を検索し，以下のような動作を行います。

１．シーケンス・ファイルがカレント・ディレクトリ内に１つ存在する場合

　自動的にそのシーケンス・ファイルが読み込まれ，その内容に基づいてアセンブルが実行されます。

２．シーケンス・ファイルがカレント・ディレクトリ内に２つ以上存在する場合

　すべてのシーケンス・ファイル名に１から順番に番号が割りふられた一覧表がモニタに表示されます。ここでアセンブル対象のシーケンス・ファイルの番号を入力します。

３．選択するシーケンス・ファイルがない場合は動作を終了します。設定をし直してください。

[　例　]

（a）MS-DOS起動の場合

```
B>A:AS6133 ↵
μPD6133 SERIES ASSEMBLER V××.××[×× ××× ××]
        Copyright(c)NEC Corporation 1995, 2000

=== SEQ FILE LIST IN CURRENT DIRECTORY ===

1) TEST1.SEQ       2) TEST.SEQ       3) TEST2.SEQ       4) TEST3.SEQ

SEQファイル番号を入力してください : 2 ↵
TEST.ASM  アセンブル開始  HH:MM:SS MM/DD/YY
```

（b）PC DOS起動の場合

```
B>A:AS6133 ↵
μPD6133 SERIES ASSEMBLER V××.××[×× ××× ××]
        Copyright(c)NEC Corporation 1995, 2000

=== SEQ FILE LIST IN CURRENT DIRECTORY ===

1) TEST1.SEQ       2) TEST.SEQ       3) TEST2.SEQ       4) TEST3.SEQ

SEQ FILE ?(SELECT NUMBER) = 2 ↵
TEST.ASM  << ASSEMBLE START >>  HH:MM:SS MM/DD/YY
```

（2）×>[ディレクトリ]AS6133△<シーケンス・ファイル名>↵　による起動

ドライブAにアセンブラを格納したディスクを，ドライブBにシーケンス・ファイルとソース・ファイルを格納したディスクをセットします。

プロンプト（A>）に対し，“ AS6133△B：SAMPLE.SEQ ”を入力します。

```
A>AS6133△B:SAMPLE.SEQ↵
```

入力後，アセンブラがメモリにロードされ，ドライブBのディスクにセットされた“ SAMPLE.SEQ ”という名のシーケンス・ファイルに従ってアセンブル処理を行います。

またシーケンス・ファイル名の拡張子“ .SEQ ”は，省略して記述することができます。この場合“ .SEQ ”が自動的に割り付けられます。

選択するシーケンス・ファイルがない場合は動作を終了します。設定をし直してください。

4.4.3　アセンブル実行中の中断

アセンブラを起動後途中で実行を中断させたい場合は，コンソールよりコントロール＋C（＾C）を入力してください。アセンブラは，＾Cを受け付けると，開いているファイルをすべて閉じたのち動作を終了します。

動作終了後，MS-DOSのプロンプト（A>）に復帰します。

４.５　アセンブル・オプション

アセンブル・オプションは，アセンブル実行時出力するファイルや，そのファイルの形式の指定，変数の指定，作業ドライブの指定を行うために使用します。

アセンブル・オプションの指定はシーケンス・ファイル内に記述します。詳しくは**４.４　アセンブラの起動方法**を参照してください。

アセンブル・オプションをまったく指定しなかった場合のアセンブル・オプションは，あらかじめアセンブラに設定されてあるディフォールト値（既定値）に従います。

表４－１　アセンブル・オプション一覧

オプション	ディフォールト[注]	内　　容	参照頁
HOS［T］ NOH［OST］	HOST	EB-6133エミュレータ用情報の出力制御	p.97
OBJ［=＜ディレクトリ＞］ NOO［BJ］	OBJ（無効）	オブジェクトの出力制御	p.98
PRO［=ファイル名［.PRO］］ NOPRO	PRO（無効）	ロード・モジュールの出力制御	p.99
LIS［T］［=ファイル名［.PRN］］ NOL［IST］	LIST（無効）	アセンブル・リストの出力制御	p.100
XREF［=ファイル名［.XRF］］ NOX［REF］	XREF（NOX）	クロスレファレンス・リストの出力制御	p.101
ROW［=n］	ROW = 66（有効）	リスト出力のページ内行数の指定（50～250）	p.102
COL［UMN］［=n］	COL = 80（col = 132）	リスト出力の１ラインのカラム数の指定（72～256）	p.102
SEQ NOS［EQ］	SEQ（NOS）	オプション情報の出力制御	p.103
TAB NOT［AB］［=n］	NOTAB = 8（有効）	タブの制御（１～255）	p.104
FOR［M］ NOF［ORM］	FORM（有効）	フォーム・フィード制御	p.105
ZZZn = m	ZZZn = 0（有効）	アセンブル変数の制御	p.106
WOR［K］=ドライブ名：	カレント・ドライブ（有効）	作業ドライブの指定	p.106
HEAD NOHEAD	HEAD（HEAD）	リスト・ヘッダの出力制御	p.107
HEL［P］	－	ヘルプ表示	p.107

注　（　）内は，/HOST指定時の設定値です。無効ではディフォールト値固定となり，有効以外は設定することはできません。

注意　EB-6133エミュレータを使用する際には，必ずHOSTオプションを指定してください。

４.５.１　EB-6133エミュレータ用情報出力制御オプション

[記述形式]

[{HOS［T] / NOH［OST]}]	ディフォールト値…/HOST

[機　　能]

μPD6133シリーズの開発ツールであるEB-6133エミュレータを使用するときに，必要となる情報を出力するかしないかを制御するオプションです。

[説　　明]

（１）HOS［T]

EB-6133エミュレータ用の情報を“.OBJ”ファイルへ出力します。

次のアセンブル・オプションが強制的に設定されます。

/OBJ/PRO/LIST/NOXREF/COL = 132/NOSEQ

また，上記のファイルはすべてSEQファイルと同じディレクトリに出力されます。

注意　/HOST選択時は，関連する全入力ファイル（ソース・ファイル）はすべてSEQファイルと同じディレクトリになければいけません。

（２）NOH［OST]

EB-6133エミュレータ用の情報を出力しません。

注意　オプション指定をしなかった場合は，ディフォールト値より，/HOSTを指定したとみなします。

４.５.２　オブジェクト・ファイル出力制御オプション

[記述形式]

[{OBJ［=＜ディレクトリ＞］ NOO［BJ］}]	ディフォールト値…/OBJ /HOST指定時…無効

[機　　能]

中間オブジェクト・ファイルを出力するかしないかの制御と，出力する場合の出力先のディレクトリの指定を行います。

また，指定したディレクトリに，ソース・モジュール・ファイルと同じファイル名の中間オブジェクト・ファイルが存在し，かつソース・モジュール・ファイルより中間オブジェクト・ファイルの作成時刻が新しい場合は，アセンブル処理を実行しません。

[説　　明]

（１）/OBJ［=＜ディレクトリ＞］

中間オブジェクト・ファイルを出力する場合のオプションです。

（２）/NOO［BJ］

中間オブジェクト・ファイルの出力を行わない場合のオプションです。

（３）本オプションで指定できるのは出力先のディレクトリです。中間オブジェクト・ファイル名を指定することはできません。

（４）中間オブジェクト・ファイルの出力を行わない（/NOOを指定時）場合は，/PROオプションは無効となります。

（５）/HOST（EB-6133エミュレータ用の情報を出力）オプションが指定されている場合，本オプションは無効となり，必ず中間オブジェクト・ファイルが，シーケンシャル・ファイルと同じディレクトリに出力されます。

４.５.３　ロード・モジュール・ファイル（PROファイル）出力制御オプション

[記述形式]

[{ PRO［=ファイル名［.PRO］］ / NOP［RO］ }]	ディフォールト値…/PRO /HOST指定時…無効

[機　　能]

ロード・モジュール・ファイル（PROファイル）を出力するかしないかの制御と，出力する場合のファイル名の指定を行います。

[説　　明]

（１）PRO［=ファイル名］

PROファイルを出力する場合のオプションです。

●ファイル名無記述

シーケンス・ファイルが存在するディレクトリ

シーケンス・ファイル名.PROという名前で出力されます。

●ファイル名記述

指定されたファイル名のファイルを作成します。ファイル名にはAUX, CON, PRN, NULを記述することができます。この場合出力先は以下のとおりです。

- AUX　　RS-232-C
- CON　　コンソール（通常はCRT）
- PRN　　プリンタ
- NUL　　ファイル出力なし

ファイル名の記述は“［ドライブ名：［￥ディレクトリ￥］］ファイル名”の形式で行ってください。

拡張子の記述を省略した場合，拡張子は“.PRO”となります。

（２）NOP［RO］

PROファイルの出力を行わない場合のオプションです。

（３）/HOST（EB-6133エミュレータ用の情報を出力）オプションが指定されている場合，本オプションは無効となり，必ずPROファイルがシーケンス・ファイルと同じディレクトリに出力されます。

４.５.４　アセンブル・リスト・ファイル出力制御オプション

[記述形式]

[{ LIS［T］［=ファイル名［.PRN］］ / NOL［IST］ }]	ディフォールト値…/LIST /HOST指定時…無効

[機　　能]

アセンブル・リスト・ファイルを出力するかしないかの制御とアセンブル・リスト・ファイルを出力する場合の出力先のファイル名指定を行います。

[説　　明]

（１）LIS［T］

アセンブル・リスト・ファイルを出力する場合のオプションです。

出力先の指定方法には，以下の二通りがります。

ファイル名無記述

ソース・ファイルと同一ディレクトリに，ソース・ファイル名と同一名のファイルを作成します。

ソース・プログラムがモジュール分割されているときは，各ソース・モジュール・ファイルが格納されているディレクトリに同一名のファイルを作成します。

拡張子は“ .PRN ”となります。

ファイル名記述

指定されたファイル名のファイルを作成します。ファイル名にはAUX, CON, PRN, NULを記述することができます。

ファイル名の記述は“［ドライブ名：［￥ディレクトリ￥］］ファイル名”の形式で行ってください。

拡張子の記述を省略した場合，拡張子は“ .PRN ”となります。

（２）NOL［IST］

アセンブル・リスト・ファイルを出力しない場合のオプションです。

（３）/HOST（EB-6133エミュレータ用の情報を出力）オプションが指定されている場合，本オプションは無効となり，必ずソース・ファイル名と同一ファイルをシーケンス・ファイルと同じディレクトリに作成し，拡張子は“ PRN ”となります。

４.５.５　クロスレファレンス・リスト・ファイル出力制御オプション

[記述形式]

[{ XRE［F］［=ファイル名［.XRF］］ / NOX［REF］ }]	ディフォールト値…XREF /HOST指定時…/NOX

[機　　能]

クロスレファレンス・リスト・ファイルを出力するかしないかの制御と，クロスレファレンス・リスト・ファイルを出力する場合の出力先のファイル名指定を行います。

また，クロスレファレンス・リスト・ファイルの出力指定を行った場合は，１つのソース・モジュール・ファイルに対し１つのクロスレファレンスが出力されます。

[説　　明]

（１）XRE［F］

クロスレファレンス・リスト・ファイルを出力する場合のオプションです。

出力先の指定方法には以下の二通りがあります。

ファイル名無記述

① アセンブル・リストが出力される場合は，アセンブル・リストの出力されるファイルに一緒に出力されます。したがってファイル名もアセンブル・リストと同じものです。

② アセンブル・リストが出力されない場合，つまりNOLが指定されている場合のファイル名は，ソース・ファイルと同一ディレクトリにソース・ファイル名と同一名のファイルを出力します。この場合の拡張子は“ .XRF ”となります。

ファイル名記述

指定されたファイル名のファイルを作成します。

アセンブル・リストと異なるファイルに出力させたい場合に使用します。

ファイル名にはAUX, CON, PRN, NULを記述することができます。

ファイル名の記述は，“［ドライブ名：［￥ディレクトリ￥］］ファイル名”の形式で行ってください。

拡張子の記述を省略した場合，拡張子は“ .XRF ”となります。

（２）NOX［REF］

クロスレファレンス・リスト・ファイルを出力しない場合のオプションです。

４.５.６　リスト出力ページ内行数（ROW NO.）制御オプション

[記述形式]

[ROW = n]	ディフォールト値…/ROW = 66 /HOST指定時…有効

[機　　能]

すべてのリスト・ファイル（アセンブル・リスト，クロスレファレンス・リストなど）で１ページ内の行数を指定します。

[説　　明]

ｎは１ページ内の行数を表します。

10進数で記述し，50≦n≦250の範囲で指定できます。

４.５.７　リスト出力カラム数制御オプション

[記述形式]

[COL［UMN］ = n]	ディフォールト値…/COL = 80 /HOST指定時…/COL = 132

[機　　能]

すべてのリスト出力（アセンブル・リスト，メモリ・マップ，クロスレファレンス・リスト）の１行のカラム数を指定するオプションです。

[説　　明]

ｎは10進数の整数で，72≦n≦255の範囲で指定できます。

４.５.８　オプション情報出力制御オプション

[記述形式]

[SEQ NOS［EQ］]	ディフォールト値…/SEQ /HOST指定時…/NOS

[機　　能]

各ソース・モジュールのアセンブル・リストの第１ページに，以下の内容を出力するかしないかを制御するオプションです。

・アセンブラ起動時に指定されたシーケンス・ファイル名とシーケンス・ファイル内に記述されている内容（SEQ=）

[説　　明]

（１）SEQ

アセンブル・リスト・ファイルの第１ページ目にオプション情報を出力します。

（２）NOSEQ

アセンブル・リスト・ファイルの第１ページ目にオプション情報（［機　　能］の内容）を出力しません。オプション情報は，アセンブル・リストと分離して出力することはできません。

（３）アセンブル・リスト・ファイル出力制御オプションで/NOLISTを指定した場合，本オプションは無効となります。

４.５.９　タブ制御オプション

[記述形式]

[{ TAB / NOTAB［= n］ }]	ディフォールト値…NOT = 8 /HOST指定時…有効

[機　　能]

アセンブル・リスト中でTABコードを使用するかしないかを制御するオプションです。

[説　　明]

（１）TAB

アセンブル・リスト中でTABコードを使用します。

本オプションを選択した場合は，アセンブル実行スピードが速くなり，使用するファイル容量が小さくなります。

（２）NOT［AB］

アセンブル・リスト中でTABコードを使用しません。TABコードの次の文字が行の先頭からnの倍数のカラムにくるようにTABコードの位置にスペースを埋めます。nは10進数で記述し，1≦n≦255の範囲で指定できます。この範囲を越えた場合は，エラーとなり，アセンブルを中止します。

本オプションは，TABコードを認識できないプリンタを使用時に利用してください。

４.５.10　フォーム・フィード制御オプション

[記述形式]

[{FOR［M］ NOF［ORM］}]	ディフォールト値…/FOR /HOST指定時…有効

[機　　能]

出力リストの改ページをフォーム・フィード・コード（８ビットJISコード：0CH）で行うか，CR/LFコードで行うかを制御するオプションです。

[説　　明]

（１）FOR［M］

出力リストの改ページをフォーム・フィード・コードで実行します。

（２）NOF［ORM］

出力リストの改ページをリスト出力ページ内行数制御オプション（ROW）で指定された行までCR/LFコードを出力して実行します。

（３）本オプションは，フォーム・フィード・コードを認識できないプリンタを使用時に利用してください。

FOR［M］を選択した場合は，アセンブル実行時間が速くなり，使用ファイル容量は小さくなります。

４.５.11　アセンブル時変数制御オプション

[記述形式]

[ZZZn = m]	0≦n≦9	ディフォールト値…ZZZn = 0
	0H≦m≦0FFFFH	/HOST指定時…有効

[機　　能]

アセンブル時変数ZZZnを値（m）で指定された値に初期化するオプションです。

[説　　明]

（１）mの評価値の有効範囲は0H-0FFFFHです。0FFFFHを越える場合は０になります。

（２）mは２，８，10，16進のどれでも記述可能です。ただしmに式または文字列を記述すると（不当オプション）エラーが発生し，アセンブルを中止します。

（３）起動時に指定がない場合，初期値として０が割り当てられます。これは，SET疑似命令による変更が現れるまで有効です。

４.５.12　作業ドライブ制御オプション

[記述形式]

[WOR［K］ = ドライブ名：]		ディフォールト値…カレント・ドライブ
		/HOST指定時…有効

[機　　能]

アセンブル実行中に使用する作業ファイルを確保するドライブ名を指定します。

[説　　明]

（１）ドライブ名指定

ドライブ名は１つのみ指定できます。

例　WORK = A：

（２）アセンブル終了後作業ファイルはすべて消去されます。

４.５.13　リスト・ヘッダ出力制御オプション

[記述形式]

HEAD	ディフォールト値…HEAD
NOHEAD	/HOST指定時…HEAD

[機　　能]

アセンブル・リスト，クロスレファレンスなどのリストのヘッダを出力するかしないかの制御を行います。

[説　　明]

（１）/HEAD

各ページにヘッダを出力する場合のオプションです。

（２）/NOHEAD

最初のページのみヘッダを出力し，以降のページには出力しない場合のオプションです。

（３）本オプションにより，ヘッダの出力が制御されるのは，以下のリストです。

アセンブル・リスト・ファイル

クロスレファレンス・リスト・ファイル

４.５.14　ヘルプ表示

[記述形式]

HEL［P］

[機　　能]

AS6133のオプションの説明を表示します。

[説　　明]

本オプションは，シーケンス・ファイル内には指定できません。

AS6133△/HEL［P］の形式でのみ指定できます。

第５章　アセンブラの出力リスト

５.１　出力リストの種類

AS6133では，アセンブラの実行後に以下のリストを出力する機能を備えています。

表５－１　出力リスト

出力ファイル	出力ファイル拡張子	アセンブル・オプション	/HOST指定時のリスト出力の有無
オブジェクト・ファイル	.OBJ	/OBJ	○
PROファイル	.PRO	/PRO	○
アセンブル・リスト	.PRN	/LIS［T］	○
オプション情報リスト	.PRN	/SEQ	
クロスレファレンス・リスト	.XRF .PRN	/XRE［F］	
ロギング情報ファイル	AS6133.LOG		○

上記で示したリストを出力したい場合は，アセンブラ起動時にアセンブル・オプションで指定します。指定方法は，**４.５　アセンブル・オプション**を参照してください。

これらのリストを出力したくない場合は，アセンブル・オプションの前に“ NO ”を付けます（**例**　/NOLIST, /NOSEQなど）。

５.２　各出力リストのリスト出力書式の制御

（１）ページ内行数

アセンブル・オプションの“ ROW = n ”に従います（50≦n≦250）。

ディフォールトはn = 66です。

（２）行内カラム数

アセンブル・オプションの“ COL = n ”に従います（72≦n≦255）。

リスト出力が指定された行内カラム数を越える場合は，指定を越えた部分がカットされます。

また，カットされる位置が全角文字をまたぐ場合は１カラム手前でカットされます。

ディフォールトはn = 80ですが，/HOST指定時はn = 132固定となります。

（３）改頁制御

アセンブル・オプションの“ FORM/NOFORM ”に従います。

FORM　……リストの改頁をFFコードで行います（ディフォールト）。

NOFORM ……ROWオプションで指定された行までCR/LFコードを出力します。

注　FF（フォーム・フィード）コード　……８ビットJISコードの0CH

LF（ライン・フィード）コード　……８ビットJISコードの0AH

CR（キャリッジ・リターン）コード……８ビットJISコードの0DH

（４）TABコード制御

アセンブル・オプションの［TAB/NOTAB］に従います。

NOTAB = n……リスト出力時，TABコードに続く文字が行の先頭からｎの倍数のカラムにくるように，TABコードの位置にスペースを埋めます（ディフォールトはn = 8）。

TAB　……リスト出力時，TABコードはそのまま出力されます。

5.3　ヘッダ部の出力

ドキュメント以外のリストには，以下のヘッダ部が各ページの冒頭に出力されます。

（1）アセンブラ名，アセンブラのバージョン・ナンバ
（2）デバイス名
（3）リスティング・タイトル
（4）アセンブル日時，ページ（モジュール順番号－モジュール内ページ番号）
（5）モジュール名
　　例　UPD6133.ASM

このヘッダ部の出力をアセンブル・オプションで制御することができます。

/HEADオプションを指定（ディフォールト）した場合は，アセンブル・リストの各ページにこのヘッダ部が出力されます。

/NOHEADオプションを指定した場合は，アセンブル・リストの先頭ページのみにこのヘッダ部が出力され，以降のページにはヘッダ部が出力されません。

５.４　アセンブラのチェック機能

記述されたプログラムの実行エラーを少なくするために，アセンブラでは，記述された命令のチェックを行います。

５.４.１　許容ビット数を越えた命令のエラー・チェック

記述された命令が許容ビット数を越えた場合は，メッセージを出力します。

★　（１）イミーディエト・データ値指定命令

ANL	A,	#data	：５ビット以上ならエラー
ORL	A,	#data	：５ビット以上ならエラー
XRL	A,	#data	：５ビット以上ならエラー
OUT	Pp,	#data	：11ビット以上ならエラー ビット８，または９がONならワーニング・メッセージを出力し， ビット８，９は，０になります。
MOV	A,	#data	：５ビット以上ならエラー
MOV	Rr,	#data	：11ビット以上ならエラー ビット８，または９がONならワーニング・メッセージを出力し， ビット８，９は，０になります。
MOV	T,	#data	：11ビット以上ならエラー
MOV	M0,	#data	：11ビット以上ならエラー
MOV	M1,	#data	：11ビット以上ならエラー
STTS		#data	：５ビット以上ならエラー

（２）DT, DW命令

DT命令	：11ビット以上ならエラー
DW命令	：11ビット以上ならエラー
DW命令	：ビット８，または９がONならワーニング・メッセージを出力し， オブジェクト値のビット８，９は，０になります。

５.４.２　暴走防止のためのチェック

動作中に電源電圧の変動が生じたり，電源投入時にパワーオン・リセットがかからなかったりした場合に，プログラム・カウンタが不定になってしまい，暴走する可能性があります。そのときプログラム・カウンタがプログラム中の命令でない番地を指してしまったときに，ジャンプ命令やHALT命令のコードと一致してしまうと無限ループに入る可能性があります。

それを防止するために，プログラム・カウンタが指したオブジェクト・コードが分岐命令，またはHALT命令のコードと一致した場合は，ワーニング・メッセージを出力し，生成される命令を表示します。

このとき，生成される命令は，JMP, JC, JNC, JF, JNF, CALL, RET, およびHALT命令です。

ワーニング・メッセージが出た場合は，生成される命令を見て，無限ループに入る可能性がある場合はプログラム変更などを行うことを推奨します。

★ 5.4.3　分岐命令の分岐先チェック（BANK0～3の自動判別）

ROMのワード数が，1024命令以上の場合のデバイスについて行います。

ニモニックに分岐先のBANK番号を記述せずに分岐させる場合は，次のように記述します。

- BANK0（0～1023命令）内へ分岐する場合は，J×0またはCALL0のオブジェクト・コード
- BANK1（1024～2047命令）内へ分岐する場合は，J×1またはCALL1のオブジェクト・コード
- BANK2（2048～3071命令）内へ分岐する場合は，J×2またはCALL2のオブジェクト・コード
- BANK3（3072命令以上）内へ分岐する場合は，J×3またはCALL3のオブジェクト・コード

BANK番号とJ×0, J×1, J×2, J×3, CALL0, CALL1, CALL2, CALL3のいずれかを記述した場合は，エラーとなります。

・分岐命令		ソース記述命令	
JMP0	addr	JMP	addr
JMP1	addr		
JMP2	addr		
JMP3	addr		
JC0	addr	JC	addr
JC1	addr		
JC2	addr		
JC3	addr		
JNC0	addr	JNC	addr
JNC1	addr		
JNC2	addr		
JNC3	addr		
JF0	addr	JF	addr
JF1	addr		
JF2	addr		
JF3	addr		
JNF0	addr	JNF	addr
JNF1	addr		
JNF2	addr		
JNF3	addr		

・サブルーチン命令　　ソース記述命令

CALL0	addr	CALL　　addr
CALL1	addr	
CALL2	addr	
CALL3	addr	

５.４.４　入力専用ポートに対する出力チェック

入力専用のポートに対して，出力命令が記述された場合，エラー・メッセージを出力します。

・入力専用ポート

P11（KI3～0）

P01（S1/LED, S0）

・出力命令

```
OUT     P11, A
OUT     P01, A
OUT     P1, #data
```

★　5.4.5　存在しないポートに対しての入力，出力命令チェック

デバイスによって，存在しないポートに対しての入力，出力命令が記述された場合は，ワーニング・メッセージを出力します。

ポート / デバイス	P10 (KI/O7-4)	P00 (KI/O0-3)	P11 (KI3-0)	P01 S1LED S0	P12 I/OPull I/OMode	P02 (I/O3-0)
D6133	○	○	○	○	×	×
D6134	○	○	○	○	×	×
D6135	○	○	○	○	○	○
D6603	○注1	○	○	○注2	×	×
D6604	○	○	○	○	×	×
D6605	○	○	○	○	○	○
D63	○	○	○	○	×	×
D63A	○	○	○	○	×	×
D64	○	○	○	○	×	×
D64A	○	○	○	○	×	×
D65	○	○	○	○	○	○
D62	○	○	○	○	×	×
D62A	○	○	○	○	×	×
D6132	○	○	○	○	×	×
D6132A	○	○	○	○	×	×
D67	○	○	○	○	×	×
D68	○	○	○	○	×	×
D69	○	○	○	○	×	×

○：ワーニング・メッセージ出力なし　　×：ワーニング・メッセージ出力あり

注意　μPD61P34B, 66P04B, 6P4B, 6P5, 6P9の場合には，対応するマスクROM版のデバイスのデバイス名を参照してください。

注1．D6603には，KI/O7-5が存在しませんが，ワーニング・メッセージの出力はありません。

2．D6603には，S0が存在しませんが，ワーニング・メッセージの出力はありません。

第６章　エラー・メッセージ一覧

６.１　起動時，実行時のエラー

AS6133では，アセンブル起動時に指定されたパラメータの記述に誤りがあったり，実行時に異常が発生するとエラー・メッセージを表示してアセンブルを中止します。

メッセージ	file not found
原因	起動時に指定したファイルが指定したドライブおよびディレクトリ内にない。
プログラムの処理	アセンブルを終了する。
ユーザの処理	正しいfileを指定する。
メッセージ	invalid option
原因	オプションの設定が不正（オプション名やパラメータが不正など）
プログラムの処理	不正オプションを出力しアセンブルを中止する。
ユーザの処理	正しいオプションを指定する。
メッセージ	invalid option value
原因	オプションの値が不正（オプションで記述可能な値の範囲外の値を記述）
プログラムの処理	不正オプションを出力しアセンブルを中止する。
ユーザの処理	正しいオプションを指定する。
メッセージ	out of memory
原因	メモリ容量が不足している。
プログラムの処理	アセンブルを中止する。
ユーザの処理	オプションの数を減らす。メモリを増やす/WORKドライブ指定を変える。

次の場合は，メッセージを表示しアセンブルを実行します。

メッセージ	HALT table overflow
原因	HALT領域オーバフロー
プログラムの処理	HALT命令が32個を越えたため，HALT命令情報を保存しない。
ユーザの処理	HALT命令の数を減らす。

（1）エラーメッセージ形式

エラーが発生したソース行が出力され，次の行にソース・ファイル名，行番号，エラー種別，エラー番号，メッセージが出力されます。最後には，エラーと警告の総数が出力されます。

［　例　］

```
        INC      C                                ←エラー発生ソース行
DIRTEST.ASM (3) : S エラー番号 058 : 未定義シンボル  ←ソース・ファイル名，行番号，エラー・メッセージ
                  ①  ②  ③        ④      ⑤
エラー数 = 1     警告数 = 0                        ←全エラー，警告数
```

①：ソース・ファイル名

②：行番号

③：エラー種別

- １：ROM領域オーバフロー
- １：プログラム・メモリを越えた領域を参照している
- A：アドレス操作誤り
- F：インクルード・ファイル・オープン・エラー
- I：ORG疑似命令のオペランド値が不正
- I：不正なデータ長
- I：INCLUDEファイル関連エラー
- M：MACRO管理ファイルI/Oエラー
- M：メモリ・オーバフロー
- M：不正な命令
- N：NESTオーバフロー
- O：マスク・オプション定義ブロック・エラー
- O：同一のマスク・オプションが２重指定
- O：不正なオペランド
- P：不正な疑似命令
- P：不正な文
- R：REPT領域オーバフロー
- R：プログラム・メモリを越えた領域に文が記述されている
- R：予約語が使われた
- S：シンボル２重定義／未定義シンボル
- S：シンボル領域オーバフロー
- S：構文スタック・オーバフロー
- S：文法誤り／構文誤り
- T：不正なオペランド
- U：シンボル未定義
- V：オペランド値／オペランド数誤り
- W：警告

④：エラー番号

⑤：エラー・メッセージ

注意　「5.4.2　暴走防止のためのチェック」のワーニング・メッセージでは，エラー発生ソース行を出力しないことがあります。

11	コード	メッセージ	Illegal first operand type
	O	原因	第１オペランドが不正。
		ユーザの処理	正しい式を記述する。
12	コード	メッセージ	Illegal second operand type
	O	原因	第２オペランドが不正。
		ユーザの処理	正しい式を記述する。
14	コード	メッセージ	Illegal first operand value
	V	原因	第１オペランド値誤り。
		ユーザの処理	その品種で許されているオペランドの値を確認する。
20	コード	メッセージ	Unreference symbol
	W	原因	未参照シンボル
		ユーザの処理	シンボルが必要かを確認する。もし必要でない場合は削除する。必要な場合はそれを参照する。
21	コード	メッセージ	No IF directive
	P	原因	IF文がない。
		ユーザの処理	正しい位置にIF文を記述する。
25	コード	メッセージ	Symbol define error
	S	原因	シンボル定義誤り。
		ユーザの処理	シンボル定義疑似命令とオペランドを正しく記述する。
27	コード	メッセージ	No OPTION statement
	P	原因	OPTION文がない。
		ユーザの処理	OPTION文を記述せずにENDOPが記述された。OPTION文を記述する。
28	コード	メッセージ	No END directive
	P	原因	END文がない。
		ユーザの処理	END文を記述する。
29	コード	メッセージ	No ENDIF directive
	P	原因	ENDIFがない。
		ユーザの処理	正しい位置にENDIF文を記述する。
31	コード	メッセージ	No ENDR directive
	P	原因	ENDR文がない。
		ユーザの処理	正しい位置にENDR文がない。
32	コード	メッセージ	No ENDM directive
	P	原因	MACROに対するENDM文がない。
		ユーザの処理	正しい位置にENDM文を記述する。
33	コード	メッセージ	No ENDP directive
	P	原因	ENDP文がない。
		ユーザの処理	正しい位置にENDP文を記述する。
35	コード	メッセージ	Nesting overflow
	N	原因	NESTがオーバフロー。
		ユーザの処理	IF文，REPT-ENDRを合わせたネスティングを40レベル以下まで減らす。

36	コード	メッセージ	Operand count error
	O	原因	オペランドの数が不正。
		ユーザの処理	正しい数のオペランドを記入する。
37	コード	メッセージ	Syntax error
	S	原因	構文の誤り。
		ユーザの処理	正しい構文を記述する。
39	コード	メッセージ	Symbol area overflow
	S	原因	シンボル領域オーバフロー。
		ユーザの処理	シンボルの数を減らす。または使用可能メモリを増やす。
41	コード	メッセージ	Invalid ENDR statement
	P	原因	不正なENDR文。
		ユーザの処理	正しい箇所に記述する。
42	コード	メッセージ	Invalid EXITR statement
	P	原因	不正なEXITR文。
		ユーザの処理	正しい箇所に記述する。
43	コード	メッセージ	Invalid ENDM statement
	P	原因	不正なENDM文。
		ユーザの処理	正しい箇所に記述する。
44	コード	メッセージ	Invalid value
	V	原因	不正な値。
		ユーザの処理	正しい値を記述する。
45	コード	メッセージ	Invalid operand
	T	原因	不正なオペランド。
		ユーザの処理	正しいオペランドを記述する。
47	コード	メッセージ	Out of address range（3）
	R	原因	プログラム・メモリを越えた領域に文が記述されている（３）。
		ユーザの処理	プログラム・メモリ領域内に命令を記述する。
49	コード	メッセージ	Used reserved word
	R	原因	予約語が使われた。
		ユーザの処理	シンボルを予約語とは別の名称に変更する。
51	コード	メッセージ	Invalid data length
	I	原因	不正なデータ長。
		ユーザの処理	正しい文字数で記述する。
52	コード	メッセージ	Include nesting error
	N	原因	インクルード多重。
		ユーザの処理	インクルードが８重までとする。
53	コード	メッセージ	Duplicated OPTION directive
	O	原因	OPTION疑似命令が２重定義されている。
		ユーザの処理	オプション・ブロックは１つのソース・プログラム内に１つだけ記述する。

55	コード	メッセージ	Rept area overflow
	R	原因	REPT領域オーバフロー。
		ユーザの処理	リピート定義のネスティングを８レベル以下にする。
57	コード	メッセージ	Symbol multi defined
	S	原因	シンボル２重定義。
		ユーザの処理	シンボルを違う名前にする。
58	コード	メッセージ	Undefined symbol
	S	原因	未定義シンボル。
		ユーザの処理	定義したシンボルを記述する。または，シンボルを定義する。
59	コード	メッセージ	Invalid Pseudo
	P	原因	不正な疑似命令。
		ユーザの処理	疑似命令を正しく記述する。
61	コード	メッセージ	Include file open error
	F	原因	インクルード・ファイルがオープンできない。
		ユーザの処理	正しいインクルード・ファイルを指定する。または，メモリ領域を拡張する。
62	コード	メッセージ	Parser stack overflow
	S	原因	構文スタック・オーバフロー。
		ユーザの処理	（　）のネスティングが16以上，演算子数が31以上を越えた。それぞれの限度数以内にする。
65	コード	メッセージ	Statement after END
	W	原因	END文の後ろに文がある。
		ユーザの処理	END文の後ろには文を記述しない。
67	コード	メッセージ	Address error
	A	原因	アドレス指定誤り。
		ユーザの処理	その品種で許されているアドレスを指定する。
68	コード	メッセージ	Operation in OPTION block
	W	原因	マスク・オプション定義ブロックに命令文がある。
		ユーザの処理	命令を削除する。
71	コード	メッセージ	Illegal first operand type and value
	O	原因	第１オペランドの値が不正。
		ユーザの処理	正しいオペランドを記述する。
72	コード	メッセージ	Illegal second operand type and value
	O	原因	第２オペランドの値が不正。
		ユーザの処理	正しいオペランドを記述する。
74	コード	メッセージ	Undefined first operand symbol
	U	原因	第１オペランド・シンボル未定義。
		ユーザの処理	定義したシンボルを記述する。または，シンボルを定義する。
75	コード	メッセージ	Undefined second operand symbol
	U	原因	第２オペランド・シンボル未定義。
		ユーザの処理	定義したシンボルを記述する。または，シンボルを定義する。

77	コード	メッセージ	Not found Mask-option block
	O	原因	マスク・オプション定義ブロックがない。
		ユーザの処理	OPTION疑似命令でマスク・オプションを指定する。
85	コード	メッセージ	Invalid ENDP statement
	P	原因	不正なENDP文。
		ユーザの処理	PUBLIC BELOWに対応するENDP文を記述する。
98	コード	メッセージ	Invalid instruction of last address in program
	W	原因	最終の命令がJMP, RET命令ではない。
		ユーザの処理	最後の命令がDW, DT命令でなければ，JMP, RET命令のいずれかを記述する。
99	コード	メッセージ	Over max value
	V	原因	記述値が許容ビット数を越えている。 ファイル名［アドレス］
		ユーザの処理	許されるビット数以内の値を記述する。 LINK時に数値が決定した場合は，ファイル名，アドレスが出力される。
100	コード	メッセージ	Over effective value
	W	原因	記述値が有効なビット数を越えている。
		ユーザの処理	無効なビットを０にする。 許されるビット数以内の値を記述する。
101	コード	メッセージ	Output request for read only port
	P	原因	入力専用ポートに対する出力命令。
		ユーザの処理	入出力ポートまたは出力専用ポートにのみ出力命令を記述する。
102	コード	メッセージ	Input/Output request for non-existent port
	W	原因	存在しないポートに対する入出力命令。
		ユーザの処理	存在するポートに対して入出力命令を記述する。
103	コード	メッセージ	Same operand value with branch or HALT
	W	原因	オペランド値が分岐命令，HALT命令と一致する。
		ユーザの処理	命令の第２オペランド以降の値，またはDefine命令の定義値が分岐命令，またはHALT命令と一致している。メッセージには，分岐命令の場合は分岐先アドレス，HALT命令の場合はオペランド値が出力されている。 プログラム・カウンタが不定になり，命令が実行された場合の動作に注意する。
104	コード	メッセージ	The source file does not exist in the same directory with SEQ file
	S	原因	/HOSTオプション指定時，ソース・ファイルがSEQファイルと同一ディレクトリにない。
		ユーザの処理	ソース・ファイルをSEQファイルと同じディレクトリに移動する。
105	コード	メッセージ	Too many INCLUDE file
	I	原因	INCLUDEファイルの数が多すぎる。
		ユーザの処理	INCLUDEファイルは，１ソース・ファイルについて16ファイルまでにする。
106	コード	メッセージ	Too long INCLUDE file name
	I	原因	INCLUDEファイルの文字数が多すぎる。
		ユーザの処理	INCLUDEファイルは，１ソース・ファイルについて，ファイル名の総文字数を255文字までとする。

付録Ａ　制限事項一覧

ここでは，AS6133 V2.21以上の制限事項について説明します。

項番	制　限　事　項
1	/HOSTオプション指定時は，シーケンス・ファイナル中に記述するソース・ファイル名にドライブ名，ディレクトリ名を含めることはできません。 また，/NOH［OST］オプション指定時も，シーケンス・ファイル中に記述するソース・ファイル名の指定方法として相対パス指定することはできません。 ソース・ファイル名の記述形式に関しては，**3.2.4　ソース・ファイル名の記述形式**を参照してください。
2	μPD6P4をプログラム・メモリ2016ワードで使用する場合には対応していません。

★

付録Ｂ　改版履歴

これまでの改版履歴を次に示します。なお，適用箇所は各版での章を示します。

版	前版からの主な改版内容	適用箇所
第２版	AS6133アセンブラがSM6133シミュレータから分離され単体の製品となったため，“ SM6133別冊 ”または“ SM6133に添付されます ”という記述を削除	全般
	対応するディバッガを当社製SM6133シミュレータから株式会社内藤電誠町田製作所製のEB-6133に変更	
	対象デバイスにμPD63シリーズを追加	
	PC-9800シリーズ，IBM PC/AT互換機，アセンブラの記述を変更	はじめに
	対応ディバッガに関する記述を追加	第Ⅱ編　操作編　第１章　製品概要
	インストールに関する記述を追加	第Ⅱ編　操作編　第２章　実行の前に
	記述可能なデバイス名と対応デバイスに関する記述を追加	第Ⅱ編　操作編　第３章　シーケンス・ファイル
	追加	付録Ａ　制限事項一覧
第３版	シリーズ名，対応可能なデバイスを追加，PC-9800シリーズ，IBM PC/AT互換機の記述を変更	はじめに
	USEPOC/NOUSEPOC, USECAP/NOUSECAPを追加	第Ⅰ編　第３章　疑似命令と制御命令
	記述可能なデバイス名と対応デバイスの記述を変更	第Ⅱ編　第３章　シーケンス・ファイル
	アセンブラの起動方法の［　例　］の記述を変更	第Ⅱ編　第４章　アセンブラの機能
	イミーディエト・データ値指定命令に記述を追加	第Ⅱ編　第５章　アセンブラの出力リスト
	分岐命令の分岐先チェックの記述を変更	
	存在しないポートに対しての入力，出力命令チェックにデバイスを追加	

〔メ　モ〕

お問い合わせ先

【技術的なお問い合わせ先】

ＮＥＣ半導体テクニカルホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−435−9494
FAX　：044−435−9608
E-mail　：s-info@saed.tmg.nec.co.jp

【営業関係お問い合わせ先】

第一販売事業部
東　京　(03)3798−6106, 6107, 6108
大　阪　(06)6945−3178, 3200, 3208, 3212
仙　台　(022)267−8740
郡　山　(024)923−5591
千　葉　(043)238−8116

第二販売事業部
東　京　(03)3798−6110, 6111, 6112
立　川　(042)526−5981, 6167
松　本　(0263)35−1662
静　岡　(054)254−4794
金　沢　(076)232−7303
松　山　(089)945−4149

第三販売事業部
東　京　(03)3798−6151, 6155, 6586, 1622, 1623, 6156
水　戸　(029)226−1702
広　島　(082)242−5504
前　橋　(027)243−6060
鳥　取　(0857)27−5313
太　田　(0276)46−4014
名古屋　(052)222−2170, 2190
福　岡　(092)261−2806

【資料の請求先】

上記営業関係お問い合わせ先またはNEC特約店へお申しつけください。

【NECエレクトロンデバイス　ホームページ】

NECエレクトロンデバイスの情報がインターネットでご覧になれます。　URL(アドレス)　http://www.ic.nec.co.jp/

アンケート記入のお願い

お手数ですが，このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

［ドキュメント名］AS6133 Ver.2.21以上　ユーザーズ・マニュアル
（U10115JJ3V0UM00（第３版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他）（　　）
ご住所（　　）
お電話番号（　　）
お仕事の内容（　　）
お名前（　　）

１．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
その他（　　）					
（　　）					

２．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

３．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

４．ご意見，ご要望

５．このドキュメントをお届けしたのは

NEC販売員，特約店販売員，その他（　　）

ご協力ありがとうございました。
下記あてにFAXで送信いただくか，最寄りの販売員にコピーをお渡しください。

日本電気（株）NECエレクトロンデバイス
半導体テクニカルホットライン
FAX：（044）435-9608

2000.6

キリトリ