資料

マルチプロセッサ開発
支援システム開発報告書

昭 和 57 年 3 月

JIPDEC

財団法人　日本情報処理開発協会

この報告書は，日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて，昭和５６年度に実施した「マイクロコンピュータの応用に関する調査研究」の一環としてとりまとめたものであります。

はじめに

当協会マイクロコンピュータ振興センター（ＭＣＣ）では，マイコン産業振興の一環として昭和53年度以来マイクロコンピュータ応用システムの高度化，システム開発の効率化などにつながる基礎的，共通的，先導的技術について，システムハウスを中心に委託開発を行うことになり，我が国のマイコン産業の技術力の育成・強化につとめているが，昭和５６年度においては次のテーマについて委託開発を行った。〔　〕内は委託先

① Ｉ／Ｏ・シミュレータ　〔㈱ティー・エス・ディ〕

② マイクロコンピュータ用リアルタイム・モニタプログラム　〔日本電気ソフトウェア㈱〕

③ インテリジェントディスクユニット　〔萩原電気㈱〕

④ マルチプロセッサ用開発支援システム　〔㈱デジタル〕

⑤ パケット交換網用汎用端末機　〔コンピュータ・ネットワーク・サービス㈱〕

⑥ リアルタイムＦＦＴ演算装置〔㈱エー・ディー・エス〕

本報告書は上記のテーマのうち「マルチプロセッサ用開発支援システム」の開発に関する成果をまとめたものである。

ここに委託開発にあたりご指導・ご協力いただいた関係各位に対し厚くお礼申し上げるとともに，これらの開発システムが広くマイクロコンピュータ応用システムの開発に携わる方々に利用され，我が国のマイクロコンピュータ産業の一層の発展に寄与することができれば幸いである。

昭和５７年３月

マイクロコンピュータプロジェクト委員会

（敬称略）

委員長	田村浩一郎	電子技術総合研究所　制御部論理システム研究室長
委員	寺田浩詔	大阪大学工学部　電子工学科教授
〃	福村晃夫	名古屋大学工学部　情報工学科教授
〃	山上喜吉	新エネルギー総合開発機構　太陽技術開発室副主任研究員
〃	出口光一郎	東京大学工学部　計数工学科助手
〃	前田英明	マイクロコンピュータシステムコンサルタント
オブザーバ	稲積義登	通商産業省機械情報産業局　情報処理振興課
〃	佐藤昌彦	通商産業省機械情報産業局　電子政策課

事務局　(財)日本情報処理開発協会マイクロコンピュータ振興センター

マイクロコンピュータプロジェクト委員会小委員会

（敬称略）

委　員	田　村　浩一郎	電子技術総合研究所　制御部論理システム研究室長
〃	山　上　喜　吉	新エネルギー総合開発機構　太陽技術開発室副主任研究員
〃	出　口　光一郎	東京大学工学部　計数工学科助手
〃	前　田　英　明	マイクロコンピュータシステムコンサルタント
〃	福　村　晃　夫	名古屋大学工学部　情報工学科教授
〃	吉　田　雄　二	名古屋大学工学部　電気工学第２学科助教授
〃	今　井　正　治	豊橋技術科学大学　第４工学系講師
〃	○寺　田　浩　詔	大阪大学工学部　電子工学科教授
〃	○島　崎　眞　昭	京都大学工学部　情報工学科助教授
〃	○河　田　　亨	大阪大学工学部　電子工学科助手

事　務　局　　(財) 日本情報処理開発協会マイクロコンピュータ振興センター

(注)　本開発においては，上記○印の委員にご担当いただきました。

目　　次

1. 概　　要

1.1 目的と特徴

1.1.1 開発の目的

マイクロコンピューターの処理内容が拡大するにつれ今後単一のＣＰＵから複数のＣＰＵを有機的に結合し処理を行う，システムのマルチ化が急速に進むと予想される，そこでマルチＣＰＵシステムのソフト・ハード両面の開発を効率よく行うため，「マルチプロセッサ開発支援システム」の開発を行った。

1.1.2 システムの特徴

本システムは会話形式でソースプログラムの作成よりシステムのシミュレーションまで一貫して行うことができる。特にＩＣＥ機能を強化してマルチＣＰＵシステムのデバッグを容易にした。以下に主な特徴を示す。

(1) １台の会話ユニットにより多数のＩＣＥユニットをコントロールできるのでターゲットが増えてもＩＣＥユニットを増すだけでよく高価なツールを何台も購入する必要がない。又，会話ユニットのみの操作でよいので取扱いが容易になる。

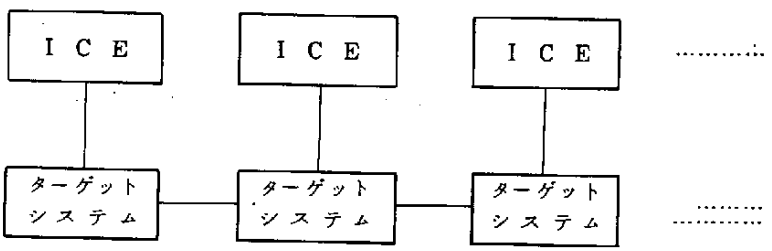


図１－１　従来のＩＣＥを使用した場合

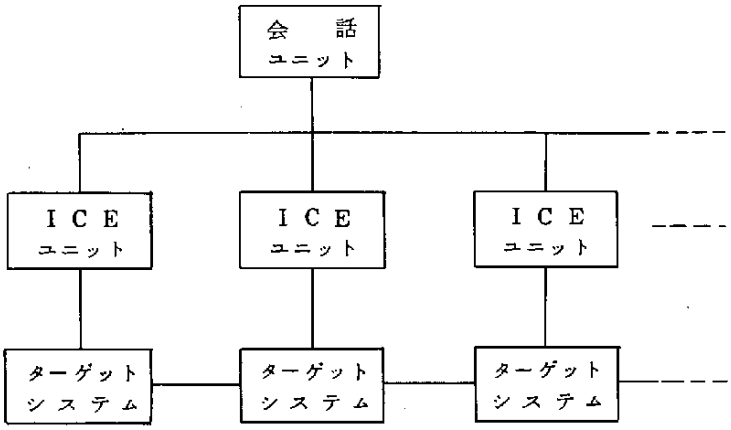
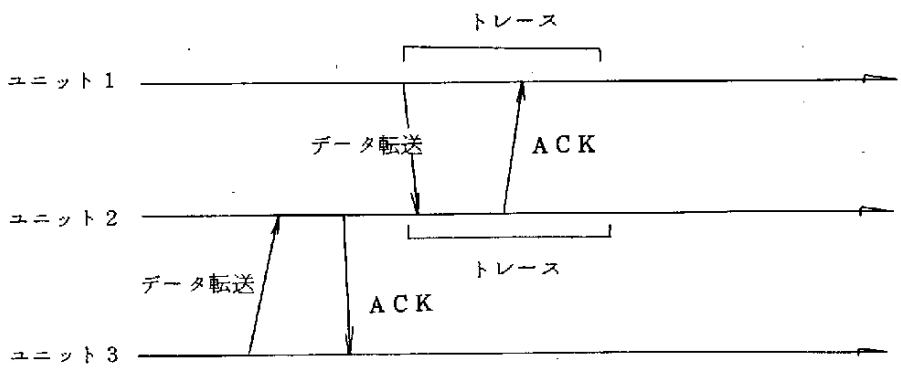


図１－２　本装置を使用した場合

(2)　複数のターゲットシステムを同時にシミュレートする場合，会話ユニットより同時にスタートをかけることができ，スタート時，ユニット間における実行命令のずれがない。

(3)　ＣＰＵ間のデータの授受において送信データと受信データを同一画面上で見ることができ，システムの切り分けが容易にできる。

(4)　トレースのスタート・ストップを外部より制御することができるので（ｔｒａｃｅ　ａｃｔｉｖｅ）不必要なトレースを行うことなく必要なトレースデータを得ることができる。



（ユニット１からユニット２へデータ転送した時のみトレースを行う。）

12 システム

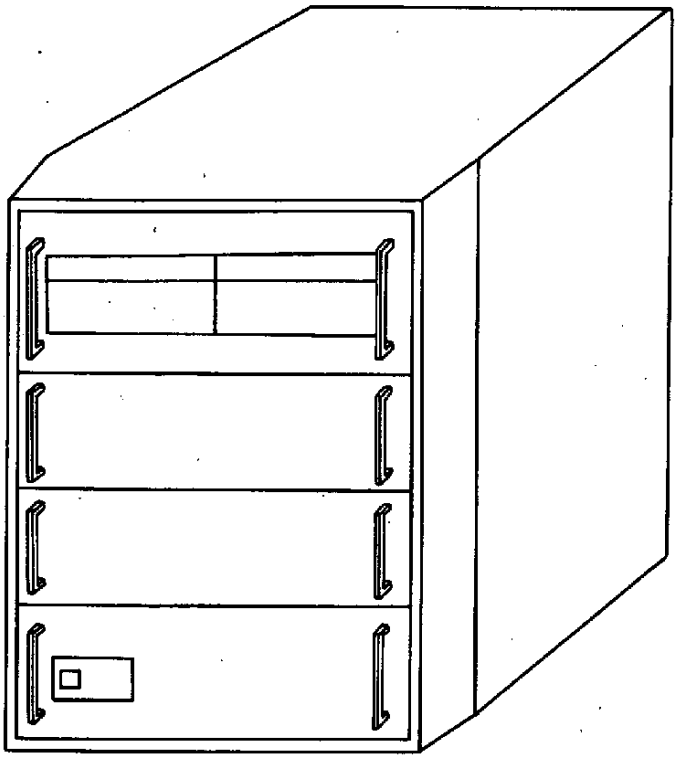
本システムは，下記のものより構成され，その外観は次図１－３に示すとおりである。

① 開発装置本体　　一式

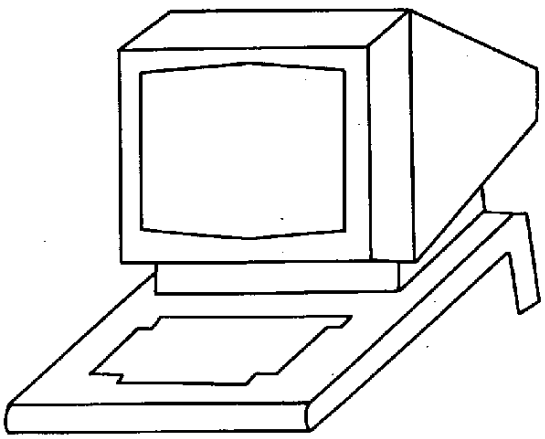
② ＣＲＴ端末　　一台

③ Ｚ８０用ＩＣＥアダプター　　二式

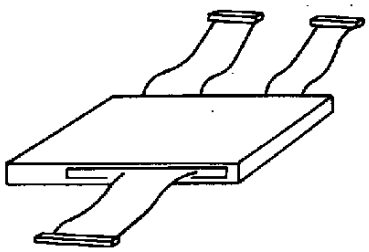
また，図１－４に結線概略図を，図１－５にコネクタ配置図を図１－６に結線図を示す。



開発装置　本体



ＣＲＴ端末



ＩＣＥアダプター

図１－３　外　観　図

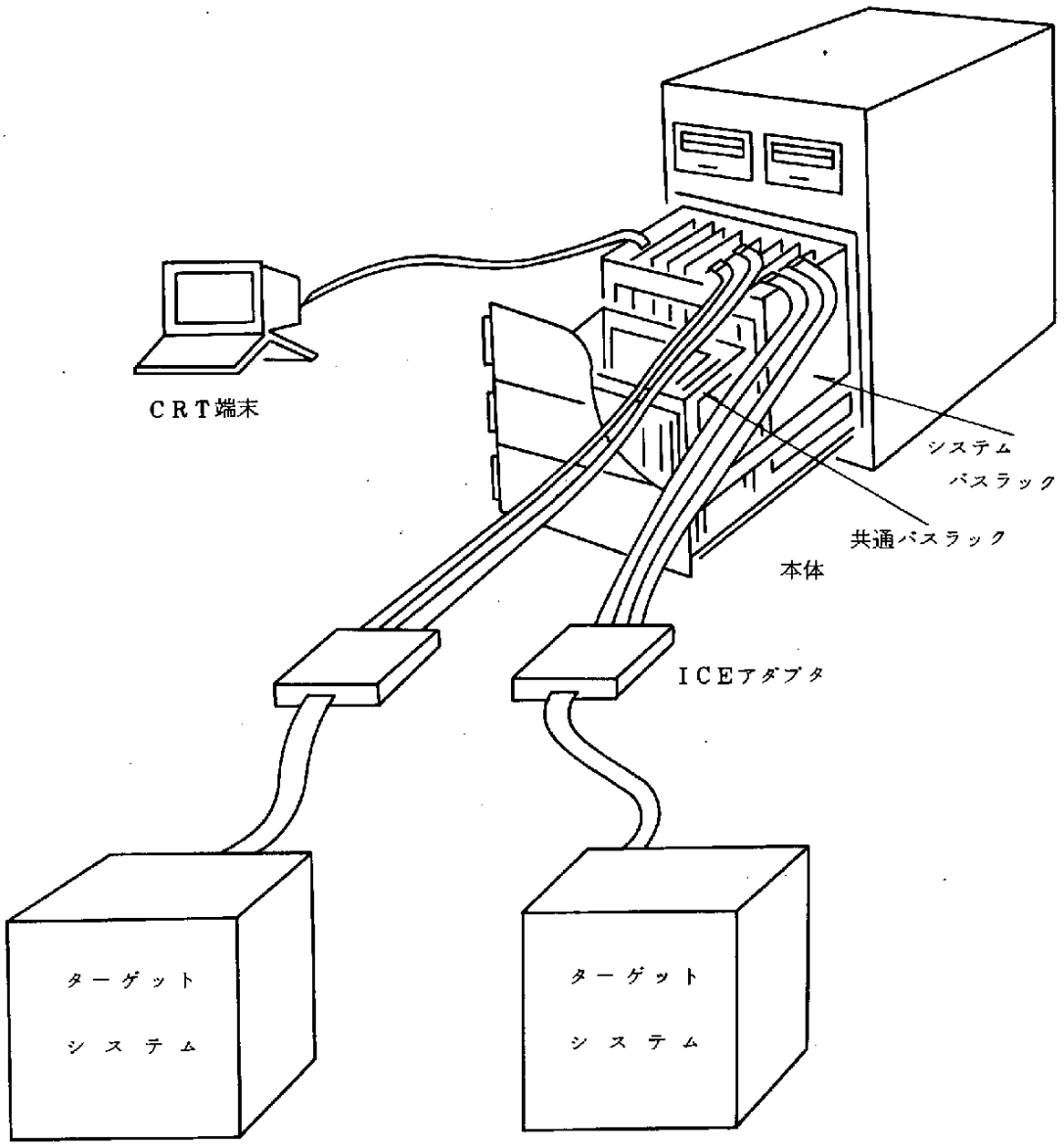


図１－４　結 線 概 略 図

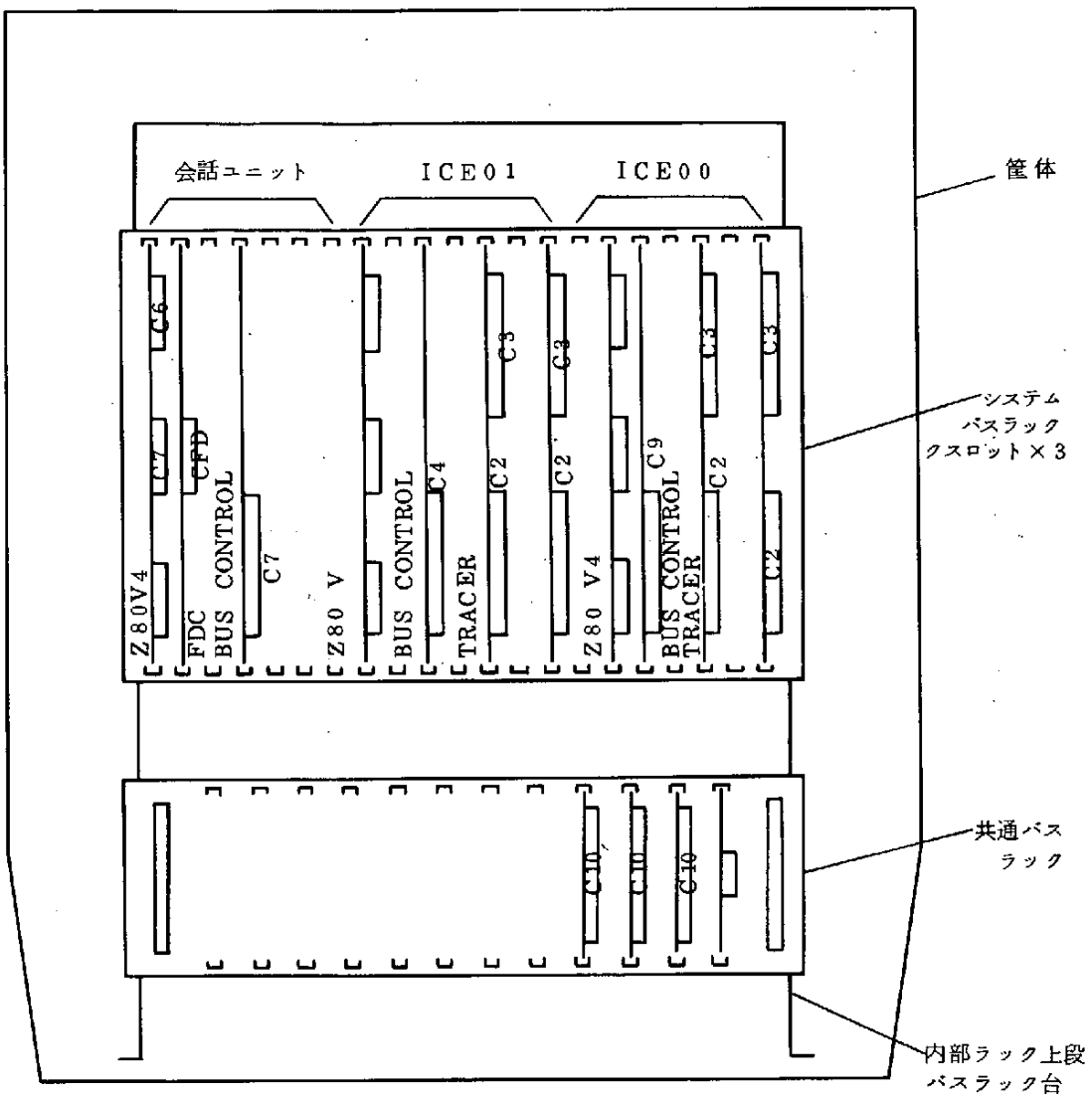


図１－５　コネクタ配置図

（本体ラック上側より見る）

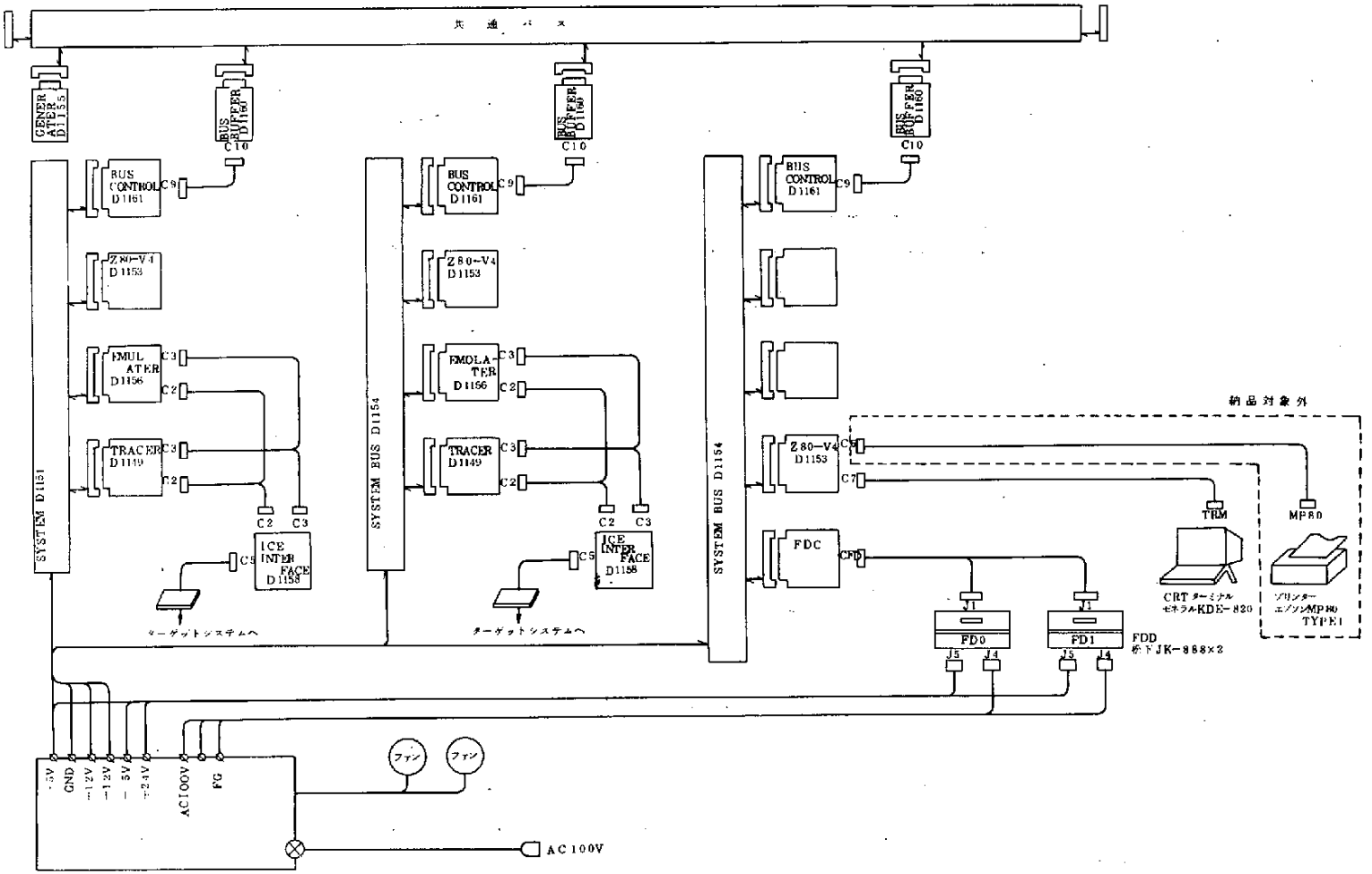

共 通 バ ス
GENERATER D1155
BUS BUFFER D1160
BUS CONTROL D1161
Z80-V4 D1153
EMULATER D1156
TRACER D1149
ICE INTER FACE D1158
SYSTEM BUS D1154
ターゲットシステムへ
納品対象外
TRM
MPB0
CRT ターミナル セキラルKDE-820
プリンター エプソンMPB0 TYPE1
FDC
FD0
FD1
FDD 形下JK-888×2
-5V
GND
-12V
+12V
+5V
+24V
AC100V
FG
ファン
AC100V

図１－６　結　　線　　図

2. ハードウェアの概説

本装置は外観図にもあるように，筐体とＩＣＥアダプターとに分かれる。そのブロック図を2.1で示す。

筐体内にはＣＰＵ，ＦＤＣ等のモジュール，マザーボード，FDD，電源等が納められている。以下各モジュールについて簡単に説明する。

21 ブロック図

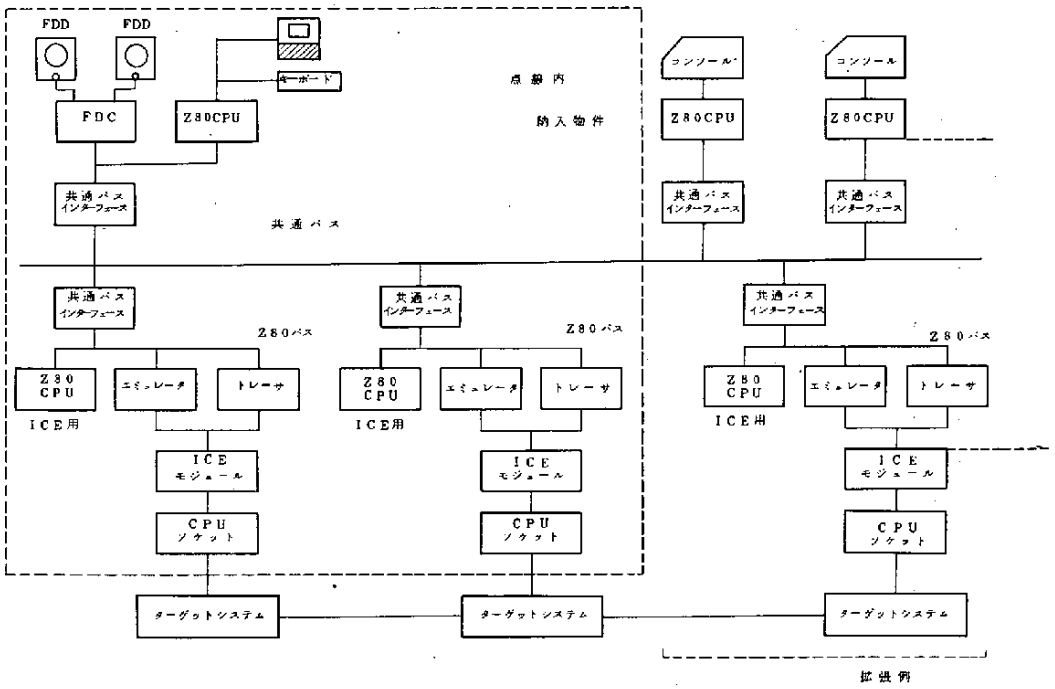


図２－１　ブロック図

22 ＣＰＵモジュール

ＣＰＵモジュールは，それ自身でワンボードＣＰＵとなるように下記の要素で構成されている。

① 4MHz　Z80CPU

② 65Kバイト9ビット（8ビット＋パリティビット）ーダイナミックRAM

③ 4Kバイトブートストラップ用ROM

④ インターバルタイマ　CTC（4MHz用）

⑤ ラインプリンター／テープリーダーインターフェイス用ＰＩＯ（4MHz 用）

⑥ コンソールターミナル／シリアルプリンタインターフェイス用ＳＩＯ（４MHz 用）

⑦ パリティジェネレータ，パリティチェッカ

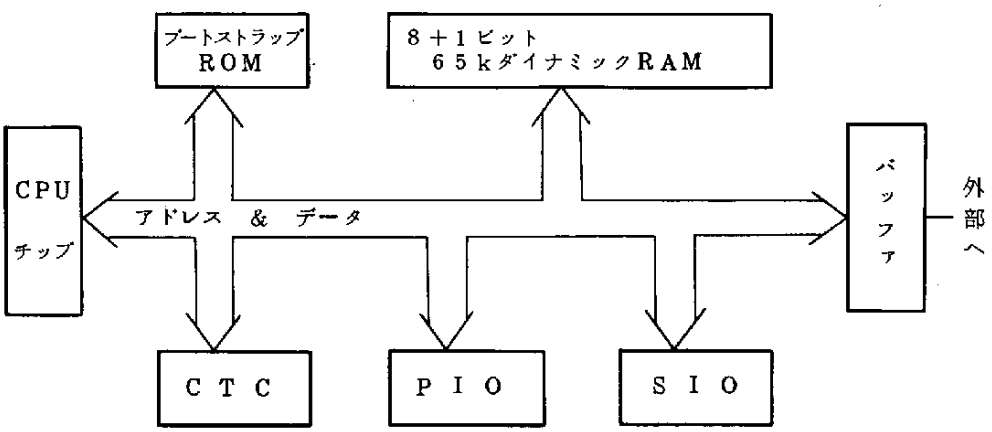


図２－２ ＣＰＵモジュールブロック図

2.3 ＦＤＣモジュール

ＦＤＣモジュールは，両面倍密度フロッピーディスクドライブを最大４台まで制御することができ，ＣＰＵとのデータ転送はＤＭＡで行っている。

モジュールは，下記の要素で構成されている。

① ＦＤＣ ＮＥＣμＰＤ７６５

② ＤＭＡＣ ８２５７

③ クロック発生回路

④ 書き込み補償回路

⑤ ＶＦＯ回路

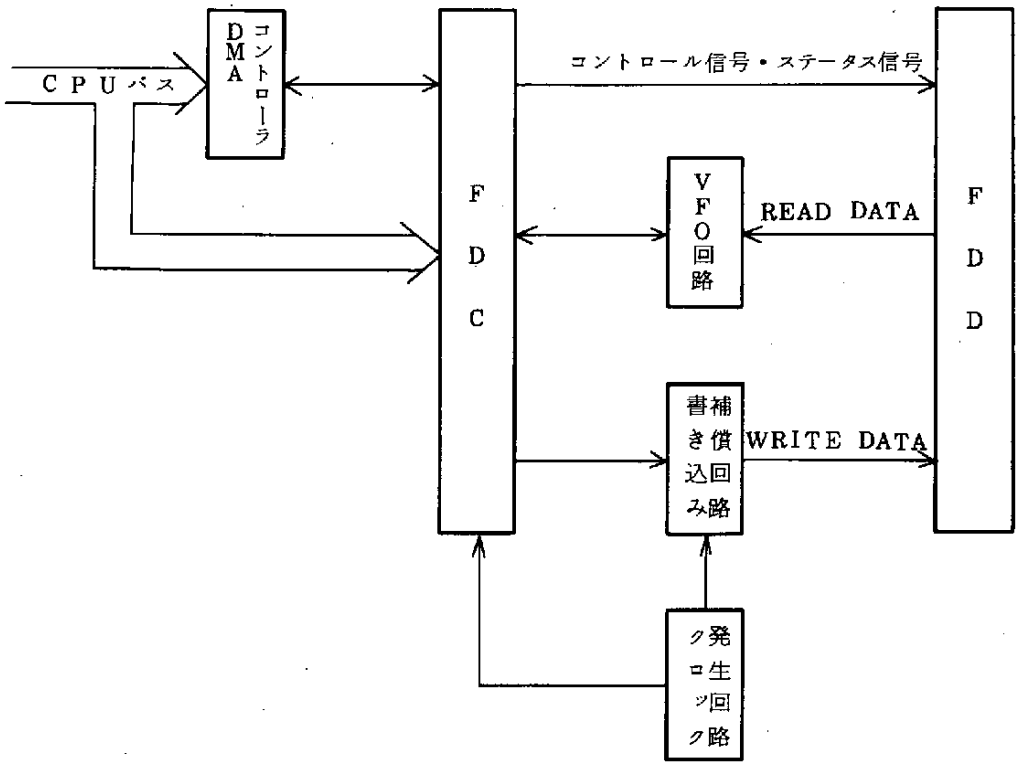


図２－３　ＦＤＣモジュールブロック図

2.4　ＩＣＥモジュール

ＩＣＥとはIn Circuit Emurator の略でターゲットシステムのデバッグを効率よく行う為のｔｏｏｌであり今回の開発支援装置製作の主眼点でもある。

本装置はマルチプロセッサシステムを対象としている為，一般のシングルプロセッサ用ＩＣＥに比べてトレース部に改良を加えている。

ＩＣＥは２枚のモジュール（トレーサ部，エミュレータ部）で構成されている。

以下にそのブロック図，動作概要を述べる。

2.4.1　ＩＣＥモジュールのブロック図

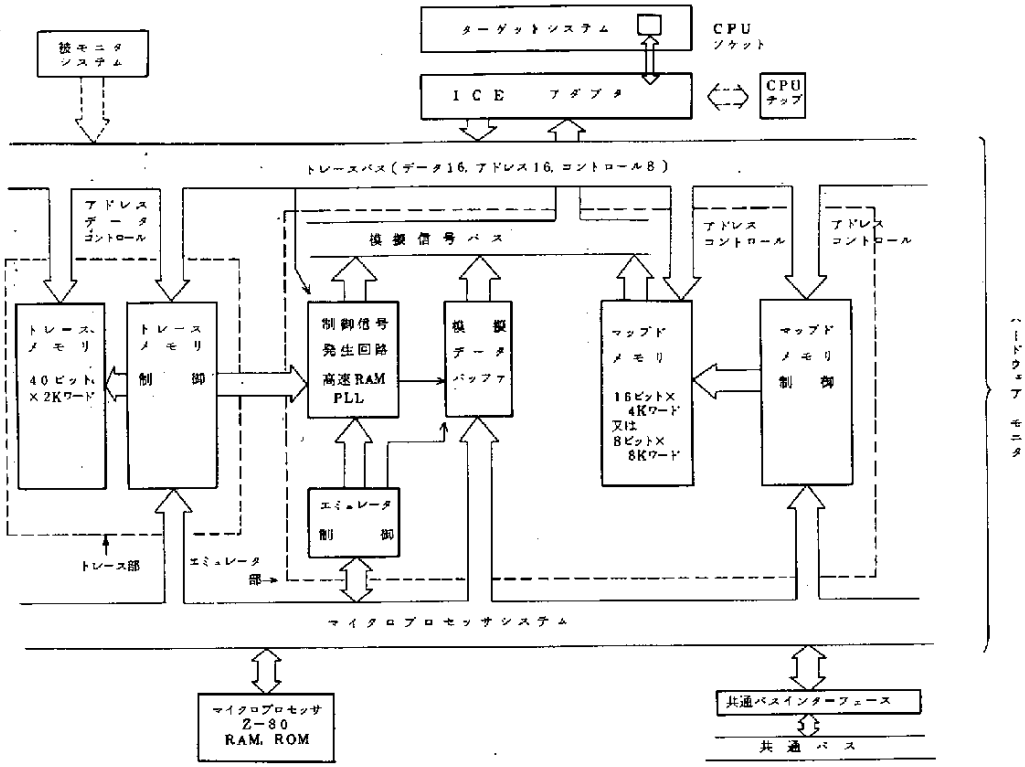


図２－４　ＩＣＥモジュールのブロック図

2.4.2　エミュレータ部

エミュレータ部はターゲット系に対する制御信号発生に用いられる。現在，ＩＣＥ応用においてはメモリに対するｒｅａｄ，ｗｒｉｔｅ，入力ポートに対するｒｅａｄ，出力ポートに対するｗｒｉｔｅの信号発生に用いられている。

エミュレータ部の構成は，エミュレート信号パターン格納用のエミュレータメモリ，そのコントロール回路，及び１６ビット２チャネルのパラレル出力ポートより成る。

エミュレータより出力される信号は４０ビット幅であり，１６ビット２チャネルと８ビット１チャネルに分かれている。

１６ビット・２チャネルの方はパラレルの出力ポートになっており，ホストプロセッサより書込まれた値を保持し，ターゲット系からのイネイブル信号によりＡチャネル，Ｄチャネル上にそれぞれ出力される。

８ビット幅のＣチャネルはエミュレータメモリとコントロール回路により，任意のパターンの波形を出力することができる。

エミュレータメモリは８ビット幅１６語のバイポーラ高速メモリを４重にインターリーブして

8ビット幅，６４語の容量を持っている。エミュレート信号は，エミュレートコントロール回路の内部可変クロック（最大３２MHz）を１～１６分周した基準クロックに従って出力される。

図２－５にエミュレータ部のブロック部を示す。

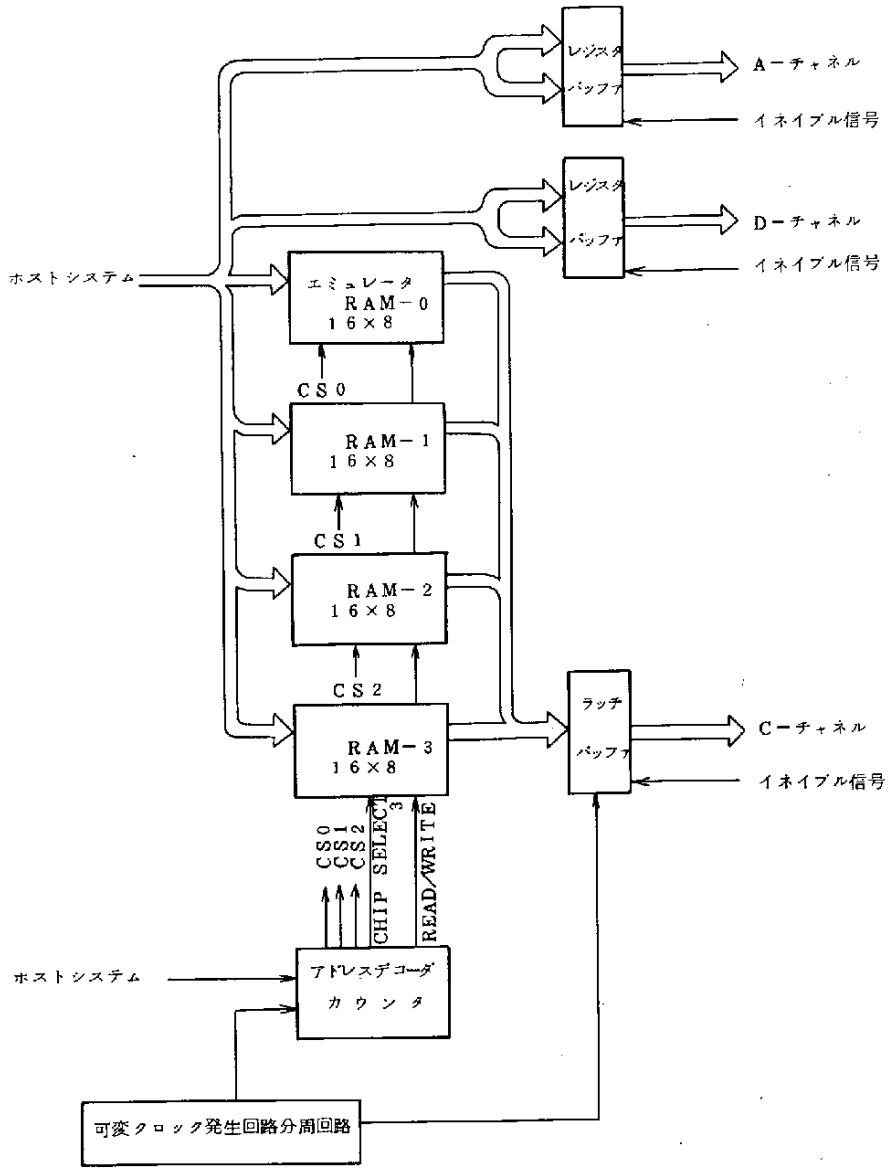


図２－５　エミュレータ部の構成

2.4.3　マップド・メモリ部

ＩＣＥ機能としては，ターゲット系のメモリが必要なエリアに存在しない場合，ターゲット系のメモリの動作が保障されていない場合，あるいはターゲット系のＲＯＭの内容を書き換えたい場合，それらを肩代りするメモリが必要である。この機能を実現するのが，マップドメモリであ

る。

マップド・メモリは８Ｋバイトの容量を持ち，ホスト側からは連続した８Ｋバイトのエリアを持つ通常のＲＡＭとして見える。ターゲット側からはホスト側の設定により，以下の３つの構成がとれる。

① ８ビット・４Ｋバイト×１

② ８ビット・４Ｋバイト×２

③ 16ビット・４Ｋバイト×１

ターゲット側がマップド・メモリをアクセスする場合，アドレスの上位４ビットがコンパレータを通り，あらかじめホスト側により設定してある値と比較される。一致した場合，ターゲット系のアドレスとデータバス，及びメモリ，コントロール信号がマップド・メモリ側に切り換わり，マップド・メモリがアクセスされる。このアドレス変換機構は２回路あり，４Ｋバイト単位に，２ブロックを任意のアドレス空間に配置できる。

また，４Ｋワード構成の場合，Ｃチャネルの下位ビットを持つことができる。

マップド・メモリのブロック図を図２－６に示す。

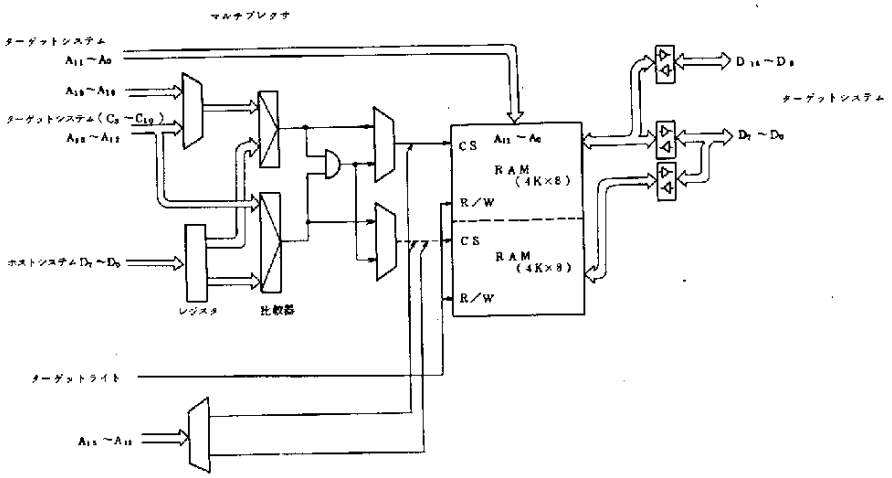


図２－６　マップド・メモリ部の構成

2.4.4　ＩＣＥアダプタ

先にも述べたように，本システムのＩＣＥモジュールはトレーサー，エミュレータ，マップド・メモリの部分が汎用化され，ターゲットＣＰＵに固有の部分がＩＣＥアダプタに集約されている。それ故，ＩＣＥアダプタのみの変更で各種のＣＰＵや回路に対することができる。

本システムのＩＣＥアダプタはＺ－８０用のものであり，ターゲット系エミュレート用のＣＰＵ（Ｚ－８０）とその制御回路，バススイッチ，ターゲット系に対するリフレッシュ・コントローラ，トレーサに対するトレース・制御用信号発生回路，及びエミュレータに対するポート・イネイブル信号発生回路より成る。

図２－７にＩＣＥアダプタ（Ｚ－８０）のブロック図を示す。バス・スイッチはエミュレート信号とターゲットＣＰＵからの信号との衝突を防ぐ為と，ターゲット系とターゲットＣＰＵ（ＩＣＥアダプタボード上にある）を切り離すのに用いられる。リフレッシュ・コントローラはターゲットＣＰＵを停止させているとき，ターゲット系のメモリのリフレッシュを行う。また，エミュレートやテストの信号はターゲット系のクロックには同期していないので，このインタフェイスにおいて同期化してバスの制御を行なっている。

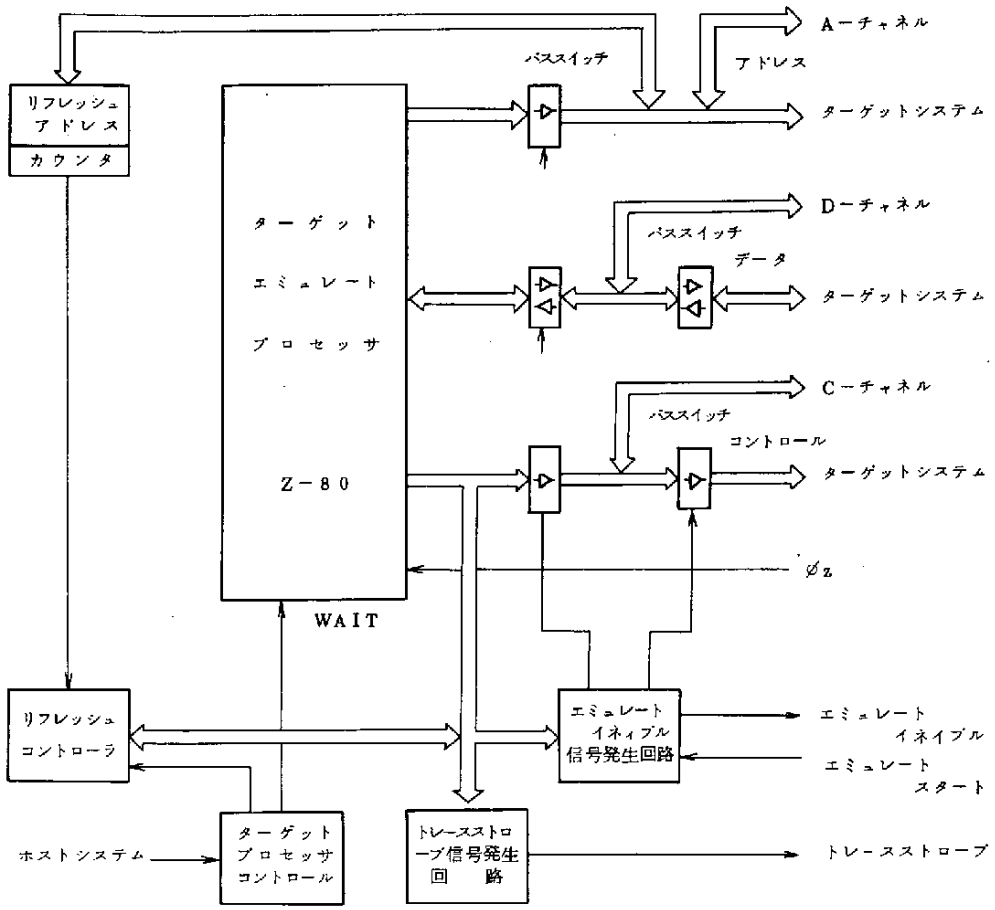


図２－７　ＩＣＥアダプタ部の構成

2.5 共通バスモジュール

本装置はマルチマイクロプロセッサ用の開発援助システムであるが，その処理の同時性，拡張性等，を考え装置自身を，マルチマイクロプロセッサシステムで構築した。

各ＣＰＵ間のデータ伝送を効率良く行う為，共通バスをもうけることにし，このモジュールでバスの制御を行う。

3. 操 作 説 明

本装置には，プログラムの作成を援助する応用プログラムとしてアセンブラ，エディタ，各種ユーティリティ，ＩＣＥなどがある。

プログラムはオーバーレイ構造をとりモニターにより各々のプログラムを管理し，会話型式で応用プログラムを呼出し実行することができる。以下にプログラムの構成を示す。

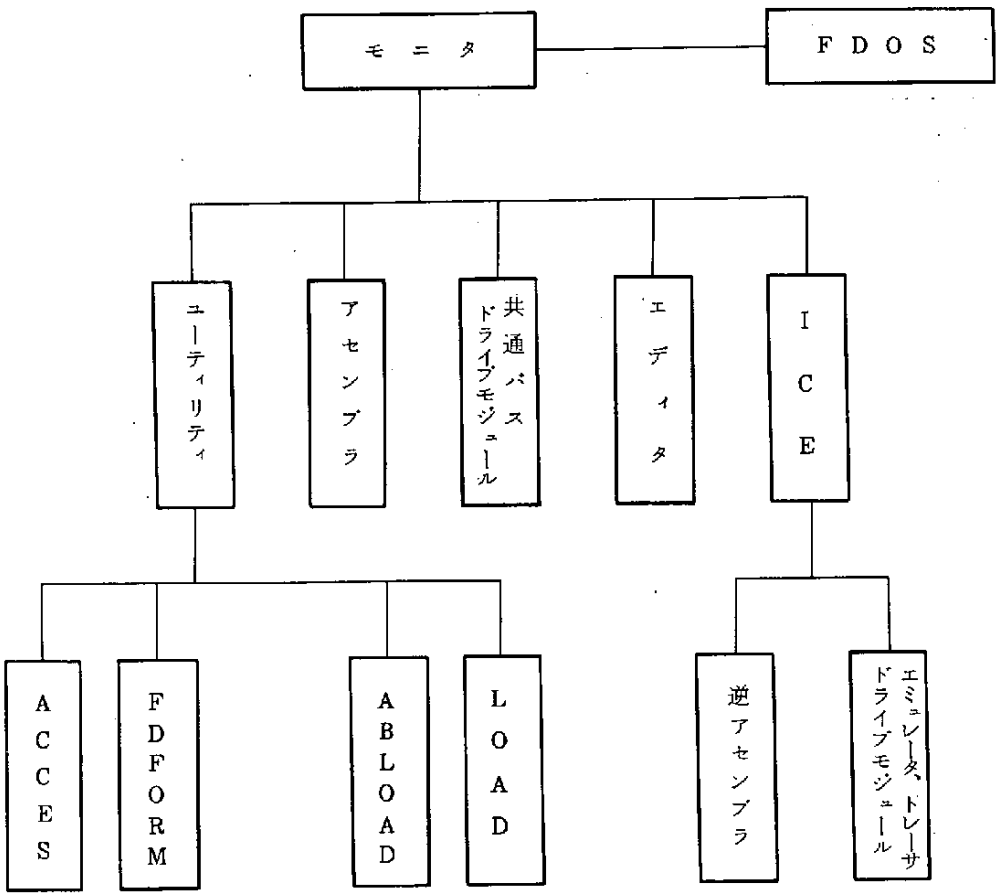


図３－１　プログラムの構成

3.1 装置の起動

システムの起動を次の手順で行う。

電源ＯＮでＣＲＴは次の表示を行う。

###　COLD－START　OR　HOT－START（C／H）[C]

ここでコールドスタート又はホットスタートの選択を行う。電源投入時システムがロードされて

いないので"C"を選びコールドスタートを行なう。(デフォルトをCとしているのでリターンキーを押せばよい。)

表示は次のようになる。

###　SET　IPL−DISK　&　HIT　ANY　KEY

FD0　又は　FD1にBOOTディスクをセットしリターンキーを押す。

表示は次のようになる。

###　MONITOR　OR　DEBUGGER　　M

モニタへ入るか(M)DEBUGGERに入るか(D)を決定する。デフォルト値はモニタへの入力とする。

モニタを選択すると表示は次のようになる。

##　DOS−BOOT　##

次にシステムディスクをFD0又はFD1へセットしリターンキーを押すとシステムディスケット内のDOSがローディングされユーザー確認のためのシステムーIDとパスワードの打込みと日付けの設定を要求する。

SYSTEM　ID=A

PAS−WORD　=A

DATE(YY/MM/DD)=820315

(年)(月)(日)

上記では，システムーIDが"A"パスワードが"A"である例を示す。以上でシステムの初期起動を終了し，

##　SYSTEM　?

を表示して，応用プログラムの選択コマンド待ちとなる。

以上の操作でユーザーはシステムが持つユーティリティープログラムを利用し，システムディスケットを媒体としてプログラムの作成を行うことができる。

本システムでは2台以上のFDDを同時に使用し，システムディスケットとは別にユーザープログラム専用のディスケットにおいて作成プログラムの管理を行うことができる。ユーティリティにある"ACCESS"コマンドによりFDDの"MOUNT"を行うことでFDDを意識せず作成プログラムをファイル名のみにより管理することができるようになる。

oリセットスイッチ

リセットスイッチを押した時

###COLD−START　OR　HOT−RTART(C/H)

を表示し，コールドスタート又はホットスタートの選択を要求する。コールドスタートは全システムの初期化を行う。操作は電源ONの時と同様とする。

ホットスタートでは，システムディスケットよりモニタプログラムがすでにローディングされていることを想定している。

##　SYSTEM　?

を表示し，応用プログラムの選択コマンド待ちとなる。

○MNI スイッチ

MNIスイッチを押すとシステムに常駐しているディバッガーへ制御が無条件で移る。

○デバッガー

ユーティリテーコマンドの“ABLOAD”を参照の事

○エラー表示

画面の右肩へエラー表示を行なう。キー入力エラーの場合

“＜KEY　DATA　ERROR＞”

を表示し，会話ユニット，ＩＣＥユニット間のエラーの場合

ＩＣＥ　ERROR－－＊＊　　（＊＊：エラーコード）

を表示する。エラーコードの内容を次に示す。

０１……タイムアウト（一定時間経過しても，応答がない）

０２……ＩＣＥユニット番号エラー

０３……他の入力パラメータエラー

32 エディタ

エディタはアセンブラなどのソースファイルの作成・修正・編集を可能にするユーティリティプログラムである。ＣＲＴディスプレイを使って会話的にできるラインエディタである。エディタは以下のコマンドを持っており，ディスクにソースファイルを作成しまたソースファイルの修正を行える。

321 制御 command

(1) Command string に対する制御

○ＲＵＢＯＵＴ

１つ前に入力された文字を消去する。消去された文字はＣＲＴにecho－back される。もし，すべてのＳtring が消去されていた場合にはprompt を印字する。

○Ｃｏｎｔｒｏｌ－Ｒ〔↑Ｒ〕

入力されたCommand　string　全部を消去する。

ＣＲＴには'＋Ｒ'をecho back する。

○ＥＳＣＡＰＥ　〔ＥＳＣ〕

Command と Command の区切りを示す。

ＣＲＴには'＄'を echo back する。

○ＬＩＮＥ　ＦＥＥＤ〔ＬＦ〕

Command　string の入力が終了した事を示す。

CRTに'CR／LF'をecho backした後，入力されたCommand string の解釈・実行を開始する。

○Control−w〔↑W〕

それまでに入力されたCommand string を表示する。

表示後続けてCommand string を Key−in できる。

(2) EDITORに対する制御

○Control−P〔↑P〕

EDIT BUFFER 内の TEXT を出力用File に格納して，EDIT を終了する。

CRTには'＋P'がecho back される。

○Control−F〔↑F〕

EDIT を終了する。

CRTには'＋F'がecho back される。

322 Edit command

(1) CP（Character Pointer）の移動 command

○B

CPをedit buffer の先頭へ移動する。

○E

CPをedit buffer の最後へ移動する。

○nJ

CPをn行目の先頭へ移動する。

- 1J＝B ； edit buffer の先頭へ移動する。
- 0J＝J ； 移動しない。

○nL

現在のCPを基準として｜n｜行移動する。

- n＞0 ； edit buffer の最後に向ってCRがnコカウントされるまで移動する。
- n＜0 ； edit buffer の先頭に向ってCRが｜n｜コカウントされるまで移動し，さらにその行の先頭に移動する。
- 0＝L ； 現在CPのある行の先頭に移動する。

○nM

現在のCPを基準として｜n｜文字分移動する。

- n＞0 ； edit buffer の最後に向ってn文字分移動する。
- n＜0 ； edit buffer の先頭に向って｜n｜文字分移動する。
- n＝0 ； 移動しない。

(2) Text 印字 command

○ n T

現在のＣＰを基準としてｎ行印字する。

ＣＰは移動しない。

- ｎ＞０ ； edit buffer の最後に向ってＣＲがｎコカウントされるまで印字する。
- ｎ＜０ ；edit buffcrの先頭に向ってＣＲが｜ｎ｜＋１コカウントされるまでの
  すべての行を印字する。
- ０Ｔ＝Ｔ；ＣＰのある行を先頭から，ＣＰの示す１つ前の文字まで印字する。

(3) 検索 command

○ S String $ or LF

現在のＣＰより，edit buffer の最後に向って，String を検索する。

実行後，ＣＰは検索されたｓｔｒｉｎｇ の後に位置する。

ただし，String がみつからなかった場合には，

'STRING NOT FOUND'

を印字し，ＣＰはedit buffer の先頭に移動する。

(4) 修正 command

○ C string 1 $ string 2 $ or LF

現在のＣＰより，edit buffer の最後に向ってString 1 を検索し，見つかった場合は，

String 2 と交換する。

見つからなかった場合は，

'STRING NOT FOUND'

を印字し，ＣＰをedit buffer の先頭へ移動する。

○ C string $$ or LF

現在のＣＰよりedit buffer の最後に向って，String を検索し，見つかった場合は，

string を消去する。

見つからなかった場合は，

'STRING NOT FOUND'

を印字し，ＣＰをedit buffer の先頭へ移動する。

○ I string $ or LF

現在のＣＰの示す位置にＳｔｒｉｎｇ を挿入する。

○ n D

現在のＣＰよりｎ文字分消去する。

- ｎ＞０ ； edit buffer の最後へ向ってｎ文字分消去する。
- ｎ＜０ ； edit buffer の先頭へ向って｜ｎ｜文字分消去する。
- ０Ｄ＝Ｄ；消去しない。

○ n K

現在のＣＰよりｎ行分消去する。

- ｎ＞０ ； edit buffer の最後に向って，ＣＲがｎコ検出されるまでの範囲内に含まれるすべての行を消去する。
- ｎ＜０ ； edit buffer の先頭へ向ってＣＲが｜ｎ｜＋１コ検出されるまでの範囲内に含まれるすべての文字を消去する。
- ０Ｋ＝Ｋ；ＣＰのある行の先頭からＣＰの示す１文字前までの文字を消去する。

(5) Parameter 表示 commad

- ：text の全行数を１０進表示する。
- ．ＣＰのある行番号を１０進表示する。
- ＝text の全文字数を１０進表示する。
- ？挿入可能な文字数を１０進表示する。

3.2.3 Macro command

(1) Macro 定義 commad

oＸＭ String ＬＦ

oＹＭ String ＬＦ

oＺＭ String ＬＦ

Macro command X,Y,Z として string で示されるCommand string を定義する。

(2) Macro 実行 command

oｎＸ

oｎＹ

oｎＺ

Macro command X,Y,Zをn回実行する。

(3) Macro 表示 command

oＸ／

oＹ／

oＺ／

Macro command X,Y,Z を表示する。

3.2.4 操 作

モニタのコマンド待ちでＥＤＩＴ〔ＣＲ〕と入力することで，エディタがロードされる。

(1) 入出力ファイルの設定

oＩＮＰＵＴ－ＦＩＬＥ

出力ファイルには，＄＄ＳＹＳ／ＴＥＭＰ１が，デフォルト表示されていて，特にファイルを指定する必要がなければ〔ＣＲ〕を入力すればよい。

○ＳＯＵＲＣＥ－ＦＩＬＥ

入力ファイルは，今からエディットするファイルを指定する。入力ファイルが複数の場合は，続けてファイルの設定が可能であり，〔ＣＲ〕のみを入力すれば，設定が終わる。

複数の入力ファイルを設定すれば，エディタバッファには，設定順にロードされる。

新しくソースプログラムを作成する場合は，入力ファイルの設定はしない。

(例)

(i) エディタを使って，新しくソースプログラムを作成する。出力はテンポラリーファイルＴＥＭＰ１（ＡＳＣ）にとり，入力ファイルは設定しない。

```
## SYSTEM ?   EDIT
##### EDITER #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB. R/W BUFF. STATUS
 00 OUTPUT FILE           $$SYS/TEMP1            ASC   W   0007  SUCCESS
 01 INPUT FILE
 1D EDITER BODY           $$SYS/$EDIT            HEX   R   0001  SUCCESS

****** EDITOR-Z-F ******

*
```

(ii) 出力ファイルにＴＥＭＰ１（ＡＳＣ），入力ファイルにＴＥＭＰ２（ＡＳＣ）を設定する。

```
## SYSTEM ?   EDIT
##### EDITER #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB. R/W  BUFF. STATUS
 00 OUTPUT FILE           $$SYS/TEMP1            ASC   W    0007  SUCCESS
 01 INPUT FILE            $$SYS/TEMP2            ASC   R    0007  SUCCESS
 02 INPUT FILE
 1D EDITER BODY           $$SYS/$EDIT            HEX   R    0001  SUCCESS

****** EDITOR-Z-F ******

*        ;エディタコマンド待ち。
```

入力ファイル，ＴＥＭＰ２（ＡＳＣ）がエディタバッファにロードされ，コマンド待ちになる。

3.3 ユーティリティ

本装置で使用するファイルに関するサービスを行うために，表３－１に示すユーティリティプログラムがある。

3.3.1 入出力ファイルの設定

ユーティリティプログラムの使用に際して，入出力ファイルを設定する場合，ボリューム名，

ファイル名，と必要に応じて，属性，R／Wをそれぞれ指定しなければならない。

(例)

```
FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
00 OUTPUT FILE             vol1/file1           atrb   r/w   0008   SUCCESS
01 INPUT FILE              vol2/file2           atrb   r/w   0008   SUCCESS
```

voln，filen は，それぞれボリューム名，ファイル名を表わす。

atrb は，ファイルの属性を示すもので，ASC（アスキーコード），RB．（リロケータブル・バイナリ），AB．（アブソリュード・バイナリ），HEX（ヘキサファイル），LST（リストファイル）がある。

R／Wで，リードかライトかを指定する。

入力は，カーソルによって誘導され，入力不要の所には，カーソルは移動しない。vol／file，atrb が，デフォルト表示されている場合，変更がなければ，〔CR〕で指定ができ，変更する場合は，そのまま，vol／file，atrb を入力すればよい。

出力ファイルに新しいファイルを設定する場合は，vol／file の頭に@（アットマーク）を付けて入力し，適当なブロック数を指定する。新しいファイルは，出力ファイルであれば，どこでも設定可能である。

(例)

新しいファイルA／MAIN（ASC）を，出力ファイルに指定し，ファイルのサイズを，3ブロック（1ブロック＝256 Byte）とする。

```
FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
00 OUTPUT FILE             @A/MAIN               ASC
                                ##### FILE SIZE (BLOCK) =3          SUCCESS
00 OUTPUT FILE             A/MAIN                ASC    W     0008   SUCCESS
01 INPUT FILE              $$SYS/TEMP1           ASC    R     0008   SUCCESS
```

表3－1　ユーティリティコマンド一覧表

コマンド	説　　明
ACCESS	MOUNT　：ディスケットの使用開始宣言を行う（Open） DEMOUNT：ディスケットの使用終了宣言を行う（Close） FPURGE　：ファイルの抹消を行う
BYE	モニターのサービスを終了する
FDCOPY	ディスケットに入っているすべてのデータを同一フォーマットで他のディスケットにコピーする。
FDPACK	ディスケットに入っているファイルを他のディスケットにコピーする。 （ボリューム名は変わらない）
FDFORM	ディスケットの初期化を行う。 ディスケットにボリュームラベルを設定する。

コマンド	説　　明
FCATA	ボリュームに属するファイルのディレクトリーを出力する。
FLENG	ファイルの長さを測る。
FMERGE	ファイルのコピー
FMODIFY	ファイル名，属性，等を変更する。
LOAD	アセンブラで出力されたリロケータブル・バイナリ形式のオブジェクトを実行可能なアブソリュートバイナリ形式のオブジェクトに変換する。
ABLOAD	アブソリュートバイナリ形式のオブジェクトを，メモリー内にロードし，実行，もしくはデバッグを行う。
LPOUT	ファイルの内容をプリンタに出力する
HEXCNV	AB. ,HEX変換

3.3.2 コマンドの説明

ACCESS

（内容） MOUNT　：Disk の Open
DEMOUNT：Disk の Close
FPURGE　：ファイルの抹消

“ACCESS” を起動すると，次のメッセージを出して，コマンド待ちになる。

FUNCTION＝

MOUNT，DEMOUNT，FPURGE のいずれかを入力し，各操作に移る。ここで〔CR〕のみをキーインすると，モニタに戻る。

(1) MOUNT

ドライブユニットに入れられたディスケットのボリュームをOpenし，利用可能にする。その際ユーザID，パスワードのチェックを行う。

ユーザボリュームの入ったディスケットを利用する時は，あらかじめ，MOUNT を行わなければならない。

(例)

ドライブユニット1に入っているDisk を Open する。ボリューム名はAユーザID＝A パスワード＝A とする。

```
## SYSTEM ?   ACCESS
##### FILE ACCESS UTILITY #####

 FC.   *** COMMENT ***      *** V/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
 1D ACCESS BODY           $$SYS/$ACCESS          HEX     R    0001   SUCCESS

    FUNCTION = MOUNT           ;MOUNT の起動

  UNIT NO      = 1             ;ドライブユニットの指定
  VOLUME NAME  = A
  USER ID.     = A
  PASS WORD    = XXXXXXXX

  ** SUCCESS **               ;ボリュームOpen のステータス


    FUNCTION =    ;再びコマンド待ちになる。[CR]を入力すればモニタに戻る。
```

(2) DEMOUNT

すでにOPENされているディスケットのボリュームをClose する。

(例)

ボリューム名"A"のディスケットをCloseする。

```
   FUNCTION = DEMOUNT    ;DEMOUNT の起動

  VOLUME NAME = A

  ** SUCCESS **          ;Close 完了
```

(3) FPURGE

抹消するファイルの属するボリューム名，属性を入力して，ファイルの抹消を行う。

(例)

ボリューム名＝A，ファイル名＝SUB 2，属性（ATRIBUTE）＝RB．のファイルを抹消する。

```
   FUNCTION = FPURGE      ;FPURGE の起動

  VOLUME NAME = A
  FILE NAME   = SUB2
  ATRIBUTE    = RB.

  ** SUCCESS **
```

BYE

（内容）

モニターのサービスを終了する。

ボリューム，ファイル名はすべてclose される。

FDCOPY

（内容）

ドライブユニット１に入っているディスケットのデータ（ディレクトリーを含む全てのデータ）を，ドライブユニット０に入っているディスケットにコピーする。

(例)

```
## SYSTEM ?   FDCOPY
##### DISK COPY #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB. R/W  BUFF. STATUS
 1D FDCOPY BODY           $$SYS/$FDCOPY          HEX   R     001  SUCCESS

** DISK COPY **

  PICK OFF ALL DISK                ;すべてのドライブユニットからDisk を外す。
  SET ORIGINAL DISK TO UNIT-1      ;ドライブユホット１にオリジナルDiskを入れる。
  SET NEW DISK TO UNIT-0           ;ドライブユニット０に新しいDiskを入れる。
```

ディスケットが挿入されると，すぐ，コピーを始める。

ディスケットを挿入する時，ドライブユニットをまちがうとオリジナルディスケットの内容をこわしてしまうので，注意を要する。

コピーが終了すると，以下のメッセージを表示する。

COPY COMPLETE RESET DISK & HIT A KEY

外したDisk をもとに戻してキーを押すと，モニターにもどる。

FDPACK

（内容）

ディスケット間でファイルのみをコピーする。

作成ファイルはディスケットに順序よくコピーされるのでディスケットのガーベッジコレクションが行える。

```
## SYSTEM ?   FDPACK
##### DISK PACK & COPY #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB. R/W  BUFF. STATUS
 1D FDPACK BODY           $$SYS/$FDPACK          HEX   R    0001  SUCCESS
```

```
** DISK PACK & COPY **

  PICK OFF ALL DISK
  SET ORIGINAL DISK TO UNIT-1
  SET NEW DISK TO UNIT-0

     ## AUTO PACK OR MANUAL PACK (A/M) ?  M     ;マニュアルパックを指定
```

オートパック「A」を指定すると，すべてコピーされる。

マニュアルパック「M」を指定すると，ドライブユニット1のファイルが表示されコピーする（Yを押す）と，次の行に，同じファイルが表示され，次のファイルに移る。

ドライブユニット1のディスケットにMAIN(ASC),MAIN(RB.) SUB1(ASC), SUB1(RB.), SUB2(ASC), SUB2(RB.)が入っており，SUB1，SUB2，のそれぞれ(ASC)と(RB.)を，コピーする例を下に示す。

```
SEQ. NO.  FILE-NAME ATRB. ORG-DATE ACC-DATE VER. ADDRESS  SIZE RECORD PRM.
  1 ***** SUB1        ASC  82/ 2/26 82/ 2/26   3     6   1     2      9
  1 ***** SUB1        ASC  82/ 2/26 82/ 2/26   3     6   1     2      9

SEQ. NO.  FILE-NAME ATRB. ORG-DATE ACC-DATE VER. ADDRESS  SIZE RECORD PRM.
  2 ***** SUB2        ASC  82/ 2/26 82/ 2/26   2     6   3     2      9
  2 ***** SUB2        ASC  82/ 2/26 82/ 2/26   2     6   3     2      9

SEQ. NO.  FILE-NAME ATRB. ORG-DATE ACC-DATE VER. ADDRESS  SIZE RECORD PRM.
  3 ***** MAIN        ASC  82/ 2/26 82/ 2/26   1     6   5     5     10

SEQ. NO.  FILE-NAME ATRB. ORG-DATE ACC-DATE VER. ADDRESS  SIZE RECORD PRM.
  4 ***** MAIN        RB.  82/ 2/26 82/ 2/26   1     6  10     2      5

SEQ. NO.  FILE-NAME ATRB. ORG-DATE ACC-DATE VER. ADDRESS  SIZE RECORD PRM.
  5 ***** SUB1        RB.  82/ 2/26 82/ 2/26   3     6  12     2      4
  3 ***** SUB1        RB.  82/ 2/26 82/ 2/26   3     6   5     2      4

SEQ. NO.  FILE-NAME ATRB. ORG-DATE ACC-DATE VER. ADDRESS  SIZE RECORD PRM.
  6 ***** SUB2        RB.  82/ 2/26 82/ 2/26   3     6  14     2      4
  4 ***** SUB2        RB.  82/ 2/26 82/ 2/26   3     6   7     2      4
```

それぞれのファイルに対して操作が終わると，以下のメッセージを出力する。

```
  PACK COMPLETE  RESET DISK & HIT A KEY
```

ディスケットをもとに戻して，キーを押すとモニターに戻る。

FDFORM

（内容）

ディスケットの初期化を行ない，そのディスケットのボリューム名，ユーザID，パスワードを指定する。

(例)

```
## SYSTEM ?    FDFORM
##### FD-FORMATER #####

 FC.   *** COMMENT ***       *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
 1D  FD-FORMAT BODY         $$SYS/$FDFORM          HEX     R    0001   SUCCESS


## FD-FRMATTER ##
  FUNCTION PASS = 37-1101

PICK OFF ALL DISK
INSERT A VERSIN-DISK TO UNIT-1

 * DENSITY? (D/S) = D
```

危険防止用パスワード（FUNCTION PASS），ディスケットの単密・倍密（DENSITY）の指定を行う。

以上の操作でフォーマットが始まる。

フォーマットが終われば，ボリューム名，ユーザID，パスワードを任意に指定できる。

```
   * VOLUME NAME  = A
   * USER-ID      = A
   * PASS-WORD    = A

  RESET DISK & HIT KEY
```

指定が終れば，ディスケットをもとに戻して，キーを押しモニターに戻る。

FCATA

（内容）

すでにOpenされているディスケット内のファイル名，属性，日付，サイズ，等を表示する。1画面に表示されるファイルの数は15で，スペースを入力すれば，次のファイルの内容を見ることができる。

(1) FILE-NAME

- ファイル名（10文字まで）
- $で始まるのは，システム用ファイル

(2) ATRB.

ファイルの属性

(3) ORG-DATE

ファイル作成をした日付

(4) ACC-DATE

最後にファイルをアクセスした日付け

(5) VER

ファイルの内容を変更した回数

(6) ADDRESS

ディスケット上のアドレス(トラック，セクター)１０進数

(7) SIZE

ブロック単位で示したファイルのサイズ(１ブロック＝256 Byte)

(8) RECODE

ファイルに書かれたレコード数

(9) PRM

＊R＊が，示されたファイルは書き込み禁止

(例)

ボリューム名$$SYSのディスケット内のファイルを見る。

```
## SYSTEM ?  FCATA
##### F-CATALOG #####

 FC.   *** COMMENT ***      *** V/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
 1D  F-CATA BODY           $$SYS/$FCATA           HEX     R    0001   SUCCESS



## VOLUME = $$SYS

 SEQ.  NO.   FILE-NAME  ATRB.  ORG-DATE  ACC-DATE  VER.  ADDRESS    SIZE  RECORD  PRM.
   1  *****  $*DOS*$     HEX   81/04/28  81/11/ 5   11     6    1   156     637   *R*
   2  *****  $*TBL*$     HEX   81/04/28  81/11/ 5    2    12    1    52     290   *R*
   3  *****  $ACCESS     HEX   81/04/28  81/11/ 5    7    14    1    45     637   *R*
   4  *****  $TAPEIO     HEX   81/04/28  81/11/ 5    3    15   20    16     229   *R*
   5  *****  $FMERGE     HEX   81/04/28  81/11/ 5    3    16   10    10     137   *R*
   6  *****  $FCATA      HEX   81/04/28  81/11/ 5    5    16   20    20     246   *R*
   7  *****  $FLENG      HEX   81/04/28  81/11/ 5    2    17   14     9     116   *R*
   8  *****  $LPOUT      HEX   81/04/28  81/11/ 5    3    17   23    20     144   *R*
   9  *****  $EDIT       HEX   81/04/28  81/11/ 5    1    18   17     8     104   *R*
  10  *****  $SEDIT      HEX   81/04/28  82/ 2/ 2    3    18   17     8     111   *R*
  11  *****  $ASMZ80     HEX   81/04/28  81/04/28    1    19    7    70     949   *R*
  12  *****  $LOAD       HEX   81/04/28  81/05/14    2    21   25    20     228   *R*
  13  *****  $HEXCNV     HEX   81/04/28  81/11/ 5    2    22   19    12     144   *R*
  14  *****  $FMODIFY    HEX   81/04/28  81/11/ 5    2    23    5    20     234   *R*
  15  *****  $FDCOPY     HEX   81/04/28  81/11/ 5    2    23   25    12     109   *R*
```

スペースを押せば，次のSEQ.＝16からのファイルを見ることができる。全てのファイルの表示が終れば，使用されている全ブロック数を示して再びボリューム名の入力待ちになる。

```
SEQ. NO.  FILE-NAME ATRB. ORG-DATE ACC-DATE VER. ADDRESS  SIZE RECORD PRM.
 61 ***** $AC       HEX  82/ 1/28 82/ 1/28   1   71  25    1      4
 62 ***** $PIP      HEX  82/ 1/28 82/ 1/28   2   71  25    1      4
 63 ***** $REN      HEX  82/ 2/17 82/ 2/17   1   72   1    1      6
 64 ***** $ASCSEND  HEX  82/ 2/17 82/ 2/18   1   72   2    1      5
 65 ***** LIST1     HEX  82/ 2/17 82/ 2/17   2   92   1  500     74
 66 ***** LIST      HEX  82/ 2/17 82/ 2/17  16  111   7 1000    200
 67 ***** $LEN      HEX  82/ 2/19 82/ 2/19   1   72   3    1      5

      ** COMPLETE **

                ** RESERVED AREA  =  3219   (BLOCKS)

## VOLUME =
```

［ＣＲ］を押せばモニターに戻る。

ＦＬＥＮＧ

（内容）

ファイルのサイズ（長さ）を測定する。単位はブロックで示され，１ブロック＝256 バイトである。

新しいパーマネントファイルを作り，テンポラリファイルの内容をコピーする時は，ファイルのサイズを測定し求めたブロック数の１０～２０％増のファイルを作成するとよい。

(例)

ボリューム名＝ＳＳＳＹＳ，ファイル名＝ＴＥＭＰ１，属性＝ASC，のファイルのブロックサイズを測定する。

```
## SYSTEM ?   FLENG
##### FILE LENGTH MESUREMENT #####

 FC.  *** COMMENT ***      *** V/F-NAME ***     ATRB. R/W  BUFF. STATUS
 00 MESURE FILE           $$SYS/TEMP1           RB.   R   0014  SUCCESS
 1D MESURE BODY           $$SYS/$FLENG          HEX   R   0001  SUCCESS


### FILE-LENGTH ###

* FILE LENGTH   =      00258   (     1 BLOCK )
* RECODE COUNT  =          5
** TOTAL LENGTH (BLOCK) =     1
```

FMERGE

（内容）

出力ファイルに入力ファイルのデータをコピーする。複数の入力ファイルを指定して，結合させることができる。

(例)

(1) 入力ファイル$$SYS／TEMP1の内容を，出力ファイル$$SYS／TEMP2にコピーする。

```
## SYSTEM ?   FMERGE
##### FILE MERGE #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB. R/W  BUFF. STATUS
 00 OUTPUT FILE           $$SYS/TEMP2            ASC   W    0008  SUCCESS
 01 INPUT FILE            $$SYS/TEMP1            ASC   R    0008  SUCCESS
 02 INPUT FILE
 1D F-MERGE BODY          $$SYS/$FMERGE          HEX   R    0001  SUCCESS

### FILE-MERGE (COPY) ###

MERGE  FC=  1

 FILE SIZE (BYTE)      00185
 RECORD COUNT             10
```

(2) 入力ファイルA／MAIN1，A／MAIN2を結合して，$$SYS／TEMP2にコピーする。

```
## SYSTEM ?   FMERGE
##### FILE MERGE #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB. R/W  BUFF. STATUS

 00 OUTPUT FILE           $$SYS/TEMP2            ASC   W    0008  SUCCESS
 01 INPUT FILE            A/MAIN1                ASC   R    0008  SUCCESS
 02 INPUT FILE            A/MAIN2                ASC   R    0002  SUCCESS
 03 INPUT FILE
 1D F-MERGE BODY          $$SYS/$FMERGE          HEX   R    0001  SUCCESS

### FILE MERGE (COPY) ###

MERGE  FC=  1

### FILE-MERGE (COPY) ###

MERGE  FC=  2

 FILE SIZE (BYTE)      00180
 RECORD COUNT             10
```

FMODIFY

（内容）

ファイル名，パーミッション（ライトプロテクト）の変更を行う。既に存在するファイル名に

変更することはできない。パーミッションを，Rに指定されたファイルの抹消・変更はできない。

(例)

ボリューム名A，ファイル名ＭＡＩＮ（ＡＳＣ）のファイルをライトプロテクトする。続けてボリューム名A，ファイル名ＳＵＢ１（ＲＢ．）をファイル名ＳＵＢ０１に変え，ライトプロテクトする。

```
## SYSTEM ?   FMODIFY
##### FILE MODIFY #####

 FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
 1D FMODIFY BODY           $$SYS/FMODIFY          HEX    R    0001   SUCCESS


### FILE-NAME & PERMISSION MODIFY ###

## VOLUME NAME = A          ;ボリューム名の指定
## FILE NAME   = MAIN       }変更するファイル名，属性
## ATRIBUTE    = ASC
* FILE NAME = MAIN          ---> ;ファイル名は変更しない〔CR〕
* PERMISSION = W  --->            ;ライト プロテクト
## FILE NAME   = SUB1       ;ボリューム名が同じなので，つづけてファイル名から入力
## ATRIBUTE    = RB.
* FILE NAME = SUB1          --->  SUB01
* PERMISSION = W  --->  R
## FILE NAME   =            ;〔ＣＲ〕入力で，ボリューム名入力待ちとなる。

## VOLUME NAME =            ;〔ＣＲ〕入力で，モニターに戻る。
```

LOAD

（内容）

アセンブラで出力されたＲＢ．形式のオブジェクトを実行可能なＡＢ．形式のオブジェクトに変換する。

（操作）

ＡＢ．出力ファイル，シンボルテーブル出力ファイル，ＬＩＮＫするＲＢ．入力ファイルを指定する。入力ファイルは複数指定可能である。

次に，各入力ファイルのロードアドレスを決定し，以下，メッセージにしたがって入力していく。

(例)

ＲＢ．ファイルにＡ／ＭＡＩＮ，Ａ／ＳＵＢ１，Ａ／ＳＵＢ２を指定しリンクして，ＡＢ．ファイル＄＄ＳＹＳ／ＴＥＭＰ１を作成する。シンボルテーブルファイルには，＄＄ＳＹＳ／ＳＹＭＢＯＬ（ＡＳＣ）を指定する。

```
## SYSTEM ?   LOAD
##### RELOCATABLE LORDER #####

 FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
 00 AB-OUTPUT FILE         $$SYS/TEMP1           AB.    W     0002   SUCCESS
 01 SYMBOL TABLE           $$SYS/SYMBOL          ASC    W     0001   SUCCESS
 02 RB-INPUT FILE          A/MAIN                RB.    R     0002   SUCCESS
 03 RB-INPUT FILE          A/SUB1                RB.    R     0001   SUCCESS
 04 RB-INPUT FILE          A/SUB2                RB.    R     0001   SUCCESS
 05 RB-INPUT FILE
 1D LOADER BODY            $$SYS/$LOAD           HEX    R     0001   SUCCESS
****** LOADER-Z-F START ******
```

ファイルの指定が終われば，ロードが始まり各モジュールの先頭番地を決めていく。

```
MODULE NAME : MAIN

START ADDRESS :                    ;モジュール名

 B-SECTOR :   0000                 ;ベースページ形の先頭アドレス（この例では不要）
 T-SECTOR :   4000   OK ?  Y       ;トップオフセット形の先頭アドレス
                                    （この例ではX'4000を指定）

START-END ADDRESS :

 T-SECTOR :   4000-400C            ;モジュール'MAIN'がロードされた領域

MODULE NAME : SUB1                 ;次のモジュール名

START ADDRESS :

 B-SECTOR :   0000
 T-SECTOR :   4100   OK ?  Y       ;プログラムSUB1のアドレスをMAINに続けてリンクした
                                    い時は〔CR〕を押し，アドレスを指定したい時はアドレス
                                    を入力する。ここでは4100（  ）番地にSUB1を設定し
START-END ADDRESS :                 ている。

 T-SECTOR :   4100-4102

MODULE NAME : SUB2

START ADDRESS :

 B-SECTOR :   0000
 T-SECTOR :   4103                 ;〔CR〕を入力し，前のモジュールにつづけてロードする。


START-END ADDRESS :

 T-SECTOR :   4103-4105
SYMBOL ERROR : NOT EXIST
```

シンボルが未定義の場合，アドレスの指定を行なうことができる。

```
SYMBOL ERROR:

SUB1       UNDEFINED
SUB2       UNDEFINED     } SUB1，SUB2が未定義
MODIFY ? Y
SYMBOL NAME = SUB1   VALUE = 1000   OK ? Y
```

SUB1の値を1000にする。

オブジェクトファイルの出力指定を行う。

```
OBJECT OUT ?  Y
```

AB．ファイルの出力領域を指定する。

```
ALL DUMP ?  Y
ADDRESS :  4000-400C
ADDRESS :  4100-4105
```

〔Y〕を入力すると，ロードしたすべての領域が出力される。

〔N〕を入力して，出力領域を指定することができる。次にシンボルテーブルの出力指定を行う。

```
SYMBOL TABLE OUT ?  Y
```

以上で，出力ファイルにそれぞれ指定されたデータが出力され，エラーがなければ，次のメッセージを表示し，モニターへ戻る。

```
SYMBOL ERROR : NOT EXIST
```

ABLOAD

（内容）

(1) AB．形式のオブジェクトをメモリーへロードする。

(2) ロード後，デバッガにてデバッグを行う。

(1) AB．形式のオブジェクトをメモリーへロードする。

ロードされる領域は，ユーティリティ 'LOAD' で設定された領域である。

(例)

AB．ファイルTEMP1（RB．）を，メモリーへロードする。

```
## SYSTEM ?   ABLOAD
##### AB-LOADER #####

 FC.   *** COMMENT ***       *** U/F-NAME ***     ATRB.  R/W  BUFF.  STATUS

 00 AB-INPUT FILE          $$SYS/TEMP1              AB.    R    0004   SUCCESS
 01 AB-INPUT FILE          [CR]
```

AB．ファイルが，ロードされた後，自動的にデバッガが起動される。

(2) デバッガ

DOS内に常駐しメモリにあるプログラムのデバッグを行う。

デバッガは，"ABLOAD"コマンドの終る時点，又はコールドスタートからのエントリー，又はMNIスイッチONで実行を開始する。実行を開始すると"＊"を表示し，コマンド入力待ちである事を示す。以下にコマンドを示す。

コマンド	実　行　内　容
C	モニターコールドスタート
D	メモリダンプ
G	実行開始
M	メモリリード・ライト
P	フラグデータ読出し
R	レジスタデータの変更
U	4000番地スタート
W	モニターウォームスタート

コマンドの実行方法（␣はユーザーが打込む）

＊ C　モニターをコールドスタートさせる。

##　DOS−BOOT　##　を表示する。以後の手続きは3－1を参照

＊ D　1 0 0 0 , 1 1 0 0

スタートアドレス　デリミタ　エンドアドレス

スタートアドレスからエンドアドレスまでのメモリデーターを表示する。

＊ M　1 0 0 0 ␣ 3E＝3F ␣ 01＝0 0

スペースキー　メモリ内容　書込みデーター　スペースキー

アドレス又はデータのあとスペースキーを押すことによりメモリデータ又は，次のメモリデータを読出し又は変更することができる。

リターンキーを押すと終了する。

＊ P

フラグの内容を表示する。

＊ R

全レジスタを表示する。

＊ R　␣ PC＝1 2 3 4

レジスタ名　レジスタデータ

＊ U　無条件で4000番地よりスタートする。

＊ W　モニターをウォームスタートさせる。

##　SYSTEM　？　を表示する。

LPOUT

（内容）

ファイル（ＡＳＣ），（ＬＳＴ）をプリンタへ出力する。

出力装置は，パラレルポートで接続されたプリンタ（ドットインパクト）カレントループで接続されたプリンタ（インクジェット）の選択可能である。

(例)

ファイル＄＄ＳＹＳ／ＬＩＳＴ（ＬＳＴ）を，ドットインパクトプリンタへ出力する。

```
## SYSTEM ?   LPOUT
##### LINE-PRINTER OUTPUT #####

 FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS

 00 PRINT FILE             $$SYS/LIST             LST    R    0008   SUCCESS
 1D LPOUT BODY             $$SYS/$LPOUT           HEX    R    0001   SUCCESS


## PRINT ##

  **I/O SELECT**
     DOT-INPACT PRINTER (0)     OR     INK-JET PRINTER (1) ? 0

## LP-READY ? ##

   HIT A KEY
```

出力中に〔ＥＳＣ〕を入力すると，出力が一時的に中断され次のメッセージが画面に表示される。

```
    * BREAK? (Y/N) =
```

〔Ｙ〕を入力すると，出力は中止されモニターへ戻る。

〔Ｎ〕を入力すると，プリントを続行する。プリントが終われば モニターへ戻る。

HEXCNV

（内容）

ＡＢ．ファイルをシステム・ユーティリティの標準形式であるＨＥＸファイルに変換する。

ＨＥＸファイルに変換できるのは，最大１６ＫＢｙｔｅ の連続領域である。

(例)

ＡＢ．ファイルＴＥＭＰ１をＨＥＸファイルに変換する。

ＨＥＸ形式の出力ファイル，ＡＢ．形式の入力ファイルを指定する。次に，ＨＥＸ形式に変更する範囲の先頭アドレスを指定する。入力ＡＢ．ファイルは，この範囲内にないとエラーとなる。

```
## SYSTEM ?   HEXCNV
##### AB.--> HEX #####

 FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS

 00 "HEX" OUTPUT           $$SYS/TEMP1           HEX    W     0008   SUCCESS
 01 "AB." INPUT            $$SYS/TEMP1           AB.    R     0004   SUCCESS
 02 "AB." INPUT
 10 HEXCNV BODY            $$SYS/$HEXCNV         HEX    R     0001   SUCCESS


### AB-->HEX ###
  LANGE = 0000-3FF0

  ** COMPLETE **
```

3.4 アセンブラ

エディタ等でディスケット上に作成したソースファイルをアセンブルしオブジェクトプログラムを作成する。

ソースプログラムは以下に述べる書式で作成する。

アセンブリ言語としてニモニックはザイログ社オリジナルのニモニックを使用し，他に擬似命令を１６種類用意した。

- マシン・コードに変換されるニモニック（命令テーブルを参照の事）

  LD，ADD，ＪＰなど６８種類

- 擬似命令

  .ORG，.END，.WORD　など１６種類

各行（ソースライン）の終端は'ＣＲ'（キャリッジ・リターン）とする。

3.4.1 行の書式

1行内にラベル，ニモニック，オペランド，コメント文の順に書き，行終端は必ず〔ＣＲ〕（キャリッジ・リターン）でなければならない。

行構成は次のようなものがある。

- コメント文のみ
- ラベルのみ
- ニモニックのみ
- ニモニック＋オペランド
- ラベル＋ニモニック＋オペランド
- 上記４種類の後にコメント文がついた行

ソースラインは，一般に次のような形をとる。

ラベルｄ１ニモニック␣　オペランドｄ２オペランド　ｄ３コメント〔ＣＲ〕

ここで

- d1 ; ラベル・デリミタ
- d2 ; オペランド・セパレータ
- d3 ; コメント・デリミタ

本アセンブラでは，特に指定のない限り，d1＝':'（コロン），d2＝','（コンマ），d3＝';'（セミコロン）とする。

3.4.2 ラ ベ ル

ラベルは，アルファベット・数字から成る6文字以内の文字列で表わされる。特に，ラベルの先頭文字は，必ずアルファベットでなければならない。

ラベルは，その文字列の後尾に，ラベル・デリミタを付加しなければならない。なお，ラベルの文字数が6文字を越える場合，越えた分については無視される。同一名のラベルを複数個定義することは許されない。また，レジスタ名をラベルとして使用する事は望ましくない。

3.4.3 ニモニック

ニモニックとしては，ザイログ社オリジナルのニモニック及び本アセンブラの擬似命令が使用できる。

3.4.4 オペランド

オペランドとしてはレジスタ名，条件名など命令テーブルに明記されている所定の文字，数値，シンボル，式などが使える。

(1) 数 値

- ○10進数 ； 0～9の数字の組み合わせ。

  例) 12, 1024, ……

- ○16進数 ； 0～9の数字，A～Fのアルファベット及び16進表示記号の組合わせ。

  本アセンブラでは，特に指定のない限り16進表示記号としてはX'を用いる。

  例) X'12, X'1024, …

(2) シンボル

オペランドとして使用可能なシンボルとしては，次の3つがある。

- ○ プログラム内でラベルとして定義されているシンボル
- ○ プログラム内で．EQUによって定義されているシンボル
- ○ プログラム内でラベルとして，もしくは．EQUによって定義されていないが，GLOBL によって，外部参照記号と宣言されているシンボル。

シンボルは値と共に，モードを持つ。すなわち．ASECT でアドレス・モードが絶対形であると宣言されたプログラム中でラベルとして定義されているシンボルのモードは'A'である。

また．BSECTでベース・ページ再配置形である場合のラベルとしてのシンボルのモードは'B'であり．TSECTの場合（カレント・ページ再配置形）のラベルは'T'モードである。さらに，各部参照記号の場合，モードは'E'である。

なお数値，シンボル以外に，シンボルと等価な記号として，PC Value Mark がある。PC Value Mark はその行に対するロケーションの値とモードと等価な意味をもつ。

例) JR ＊＋4

3.4.5 式の形式

式は数値，シンボル，PC Value Mark，演算記号と，(,)で記述される。

演算記号としては，'+'，'−'，'＊'，'／'，'&'，'/' の6種類がある。

- ＋（プラス）　；加算記号
- −（マイナス）　；減算記号
- ＊（アスタリスク）；乗算記号
- ／（スラッシュ）；除算記号
- &（アンド）　；論理積
- /（感嘆符）　；論理和

3.4.6 擬似命令

擬似命令とは，プログラミングの手助けをする為のアセンブラに対する命令である。ある種のものを除いて機械語には変換されない。

擬似命令として，以下の16種類がある。

(1) アドレス・モードの設定

```
.ASECT
.BSECT
.TSECT
```

- .ASECT；以下のプログラムのアドレス・モードが絶対形であることを示す。
- .BSECT；〃　がベース・ページ再配置形であることを示す。
- .TSECT；〃　がカレント・ページ再配置形であることを示す。

(2) ロケーション・カウンタの設定

```
.ORG␣expression
```

ロケーション・カウンタの値を，expression のもつ値に設定する。

この命令は，アドレス・モードが絶対形の時，すなわち，.ASECTで宣言されている場合にのみ許される。

(3) 領域の確保

```
.DEFS ␣ expression
```

ロケーション・カウンタの値を，expression のもつ値だけ増加させる。すなわち，メモリ領域の確保を行う。

(4) 外部記号の宣言

```
.GLOBL ␣ symbol (,…, symbol)
```

symbol が外部記号であることを宣言する。

symbol の扱いはラベルと同じである。

複数の symbol の定義を一度に行うことができる。

(5) シンボルの値の割り当て

```
symbol : .EQU ␣ expression
```

symbol に expression のもつ値とモードを割り当てる。

expression の中にシンボルが使用されている場合，そのシンボルはその行以前に，ラベルとして定義されていなければならない。

(6) データの定義

```
.WORD ␣ expression (,…, expression)
.DWORD ␣ expression (,…, expression)
```

- WORD ； 1 byte データを定義する。expression のモードは 'A' でなければならない。
- DWORD； 2 byte データを定義する。expression のモードは何であってもよい。

複数のデータの定義を一度に行うことができる。

(7) ASCII CODE データの定義

```
.ASCII ␣ 'string'
```

string で表わされる文字列のASCII CODE（パリティ・ビットを削除したもの）をデータとして定義する。

(8) アセンブル・リスト出力の有無の指定

```
.LIST ␣ expression
```

パス－2におけるアセンブル・リストの出力の有無を指定する。

expression ≥ 0 ；リストを出力する。

expression ＜0 ；リストを出力しない。

指定の無い場合は，アセンブル・リストを出力する。

(9) プログラム・セグメント名の宣言

```
.TITLE ␣ symbol (,'coment')
```

プログラム・セグメントの名前の定義を行う。

プログラム・セグメント内にこの擬似命令のない場合，初期設定でキーインされたタイトル名もしくはMAINPRを名前とする。

symbol の扱いはラベルと同じである。

また comment は２６文字以内の文字列で，リスト上部にタイトル・コメントとして印字される。

(10) 改ページの指定

```
.PAGE ('comment')
```

・アセンブル・リスト印字時にリストのページの変更を行う。

comment は２６文字以内の文字列で，リスト上部にタイトルコメントとして印字される。

(11) 空白行の指定

```
.SPACE ␣ expression
```

アセンブル・リスト印字時に，同一ページ内に expression の表わす数だけの空白行を挿入する。

(12) プログラムの終了表示

```
.END
```

プログラム・セグメントの終了を表示する。

アセンブル対象のソースプログラムの最後には，必ずこの擬似命令がなければならない。

(13) ソースプログラムファイルの終了表示

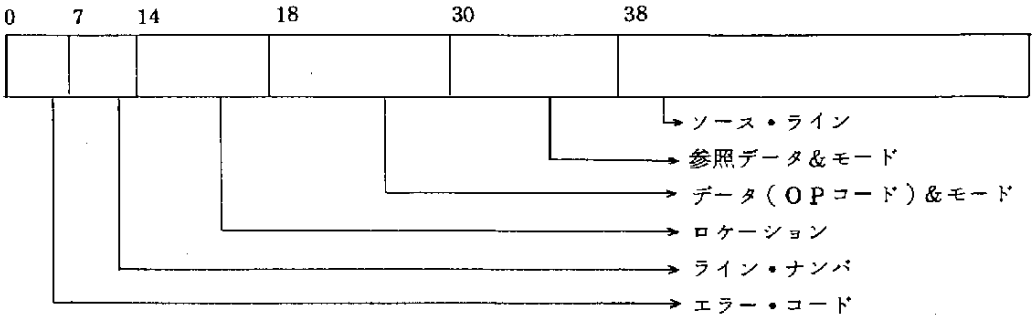
```
.EOF
```

複数のソースプログラムファイルを一度にアセンブルするとき，最終ソースプログラム以外のソースファイルは必ずこの擬似命令で終らなければならない。

3.4.7 リ ス ト

ＰＡＳＳ－１実行中，もしくはＰＡＳＳ－２ノーリスト・モード時にエラーが発生すると，本アセンブラは，エラー・コード，行番号，ソースラインの順に，エラー・リストを印字する。

また，ＰＡＳＳ－２　リストモード時，アセンブル・リストを印字する。



3.4.8 操　作

システムのコマンド待ちで，ＡＳＭＺ８０，又はＡＳＭＺ８０Ｎ とキーインすることで，アセンブラを起動できる。後尾にＮを付けた場合は，アセンブルリストの出力を禁止できる。

(1) 入力ファイルの設定

○ＲＢ－ＦＩＬＥ

ＲＢ出力ファイルには，＄＄ＳＹＳ／ＴＥＭＰ１がデフォルトで表示されていて，特にファイルを指定する必要がなければＣＲを入力すればよい。

○ＬＩＳＴ－ＦＩＬＥ

アセンブルリスト出力ファイルには，＄＄ＳＹＳ／ＬＩＳＴがデフォルトで表示される。ただし，ＡＳＭＺ８０Ｎでアセンブラを起動させた場合はデフォルト表示はされず．〔ＣＲ〕を入力すればアセンブルリストは作成されない。

○ＳＯＵＲＣＥ－ＦＩＬＥ

デフォルト表示は，＄＄ＳＹＳ／ＴＥＭＰ１で，上記同様の操作となる。ソースファイルが２つ以上ある場合は，つづけてファイルの設定が可能であり，〔ＣＲ〕のみを入力すれば設定が終る。

(2) ソーステキスト名及びデリミタの設定

ソースファイル内に，．ＴＩＴＬＥでモジュール名を指定しない場合に，ソーステキスト名を入力すれば，これがモジュール名として用いられる。

デリミタは3.4.1で示した通りであるが，ソーステキスト内で他のデリミタを使用した場合に，変更することができる。

(3) アセンブル実行

以上の操作で，アセンブルが開始され，結果をコンソールへ表示する。

ＲＢ．の出力指定をして，システムに戻る。

(4) 実　行　例

(i) ソースファイルＴＥＭＰ１（ＡＳＣ）を，アセンブルする。

アセンブルリスト出力ファイルには，ＬＩＳＴ（ＬＳＴ）を指定し，ＲＢ．出力ファイルには，ＴＥＭＰ１（ＲＢ．）を指定するが，エラーが発生したのでＲＢ．は作成しない。

```
## SYSTEM ?   ASMZ80
##### ASSEMBLER Z-80 #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W  BUFF.  STATUS
 00 RB-FILE               $$SYS/TEMP1           RB.    W    0001   SUCCESS
 01 LIST-FILE             $$SYS/LIST            LST    W    0008   SUCCESS
 02 SOURCE-FILE           $$SYS/TEMP1           ASC    R    0005   SUCCESS
 03 SOURCE-FILE
 1D ASSEMBLER BODY        $$SYS/$ASMZ80         HEX    R    0001   SUCCESS

*** CRAMBO-Z-F Z-80 VERSION ***


SOURCE TEXT NAME ? :
DELIMITER NORMAL(0) OR NOT(1) ? :  0
 PASS-1 START

 PASS-1 END

 PASS-2 START
E-13     7.  0007  1000      A  0009 T          DJNZ     LOP2
E-13     8.  0009  CD0000    A  0000 A          CALL     SUB2
 PASS-2 END
ERROR EXIST

 R.B. OUT(0) OR NOT(1) ? :  1     エラーが発生したのでＲＢ．は出力しない。

## SYSTEM ?
```

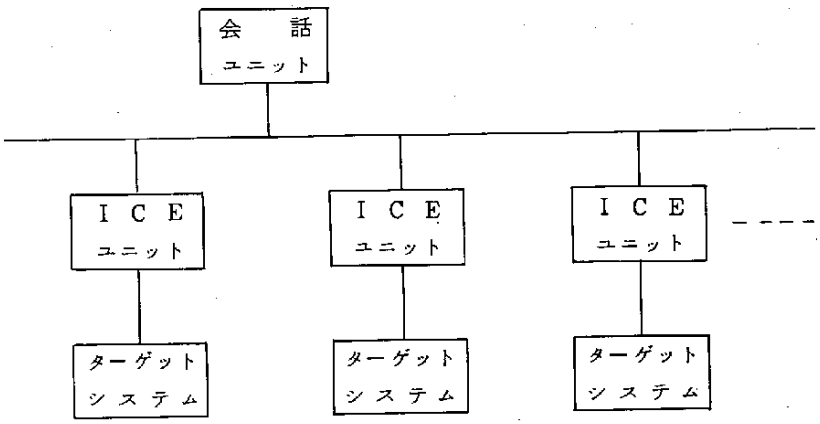
(ii) ソースファイルＭＡＩＮ１，ＭＡＩＮ２.（ＡＳＣ）を，１つのソースモジュールとして，アセンブルして，ＴＥＭＰ１（ＲＢ．），ＬＩＳＴ（ＬＳＴ）を作成する。

ＭＡＩＮ１の最終行は'．ＥＯＦ'命令でなければならない。

```
## SYSTEM ?   ASMZ80
##### ASSEMBLER Z-80 #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W  BUFF.  STATUS
 00 RB-FILE               $$SYS/TEMP1           RB.    W    0001   SUCCESS
 01 LIST-FILE             $$SYS/LIST            LST    W    0008   SUCCESS
 02 SOURCE-FILE           A/MAIN1               ASC    R    0005   SUCCESS
 03 SOURCE-FILE           A/MAIN2               ASC    R    0008   SUCCESS
 04 SOURCE-FILE
 1D ASSEMBLER BODY        $$SYS/$ASMZ80         HEX    R    0001   SUCCESS

*** CRRMBO-Z-F Z-80 VERSION ***


SOURCE TEXT NAME ? :
DELIMITER NORMAL(0) OR NOT(1) ? :  0
 PASS-1 START

 .EOF COMES

 PASS-1 END

 PASS-2 START

 PASS-2 END

 R.B. OUT(0) OR NOT(1) ? :  0     RB.を出力する。
```

3.5 ICE

CRTを見ながら会話形式で複数のターゲットシステムのエミュレーション，デバッグを行う。



全構成概略

3.5.1 動作機能

次のコマンドがあり，コマンドメニューより，それぞれの処理を呼び出すことができる。

(A) initialize …… ターゲットシステムと会話システムの結合，切離し，又ターゲットシステムのネーミングを行う。

(B) memory & register … ターゲットシステムのメモリ内容の読出し，メモリへのデータの書込み，およびレジスタの読出し，書込みを行う。

(C) I/O port & register ……ターゲットシステムに結合しているI/Oポートからのデータ出力およびI/Oポートデータの読み込みを行う。又レジスタの読出し，書込みを行う。

(D) single step …… ターゲットシステムの次の1命令を実行し，実行結果を逆アセンブルしCRTへ表示する。又，レジスタの読出し，書込みを行う。

(E) Set trace condition ……トレーサーのトレース条件の設定，変更を行う。

(F) mapped memory …… マップドメモリの接続・分離・マップドメモリのアドレス設定を行う。

(G) Display trace data …… トレーサーに保持されているトレースデータを逆アセンブルして，ＣＲＴへ表示する。

(H) Emulation & trace start/stop …… エミュレーション，トレースの開始，終了を行う。

(I) File control …… ターゲット実行プログラムをＦＤからターゲットへロードする。

(J) Data Fill …… ターゲットのＲＡＭの指定範囲を特定のデータで埋める。

画面に表示されたデータは，スクリーン上で変更が可能である。カーソルを該当データ上へ移動し，変更データを打込むことによりデータの設定変更を行うことができる。カーソル移動キーを次に示す。

control−J （Left）……カーソルは左へシフトする。カーソルが左端にある場合，その行の右端へ移る。

control−K （Right）……カーソルは右へシフトする。カーソルが右端にある場合，その行の左端へ移る。

control−I （Up）……１行上の行のうち右方向で最も近い変更データエリアへカーソルを移す。該当エリアがない場合は順に上の行をサーチしてゆく。カーソルの位置より上に該当エリアがない場合，最下端より順に上の行をサーチしてゆく。

control−M （Down）……１行下の行のうち右方向で最も近い変更データエリアへカーソルを移す。該当エリアがない場合は，順に下の行をサーチしてゆく。

カーソルより下に該当エリアがない場合最上端よりサーチしてゆく。

○画面のハードコピー

Control−Pを押すことにより画面のハードコピーをとることができる。

○サイクルワードの設定

〔　　〕はサイクルワードであることを示し，数種類の動作モードのうちの一つを選択することができる。

カーソルを〔　〕内に移動させControl−Aを押すことにより動作モードの変更を行う。

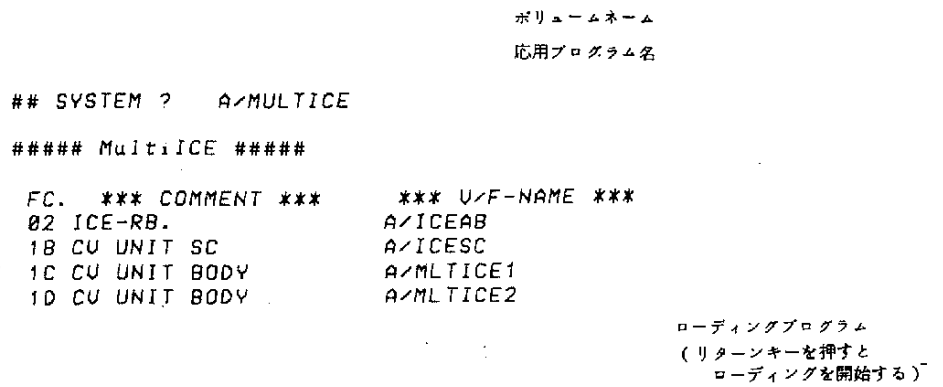
カーソルが〔　〕内にありcontrol−A以外のキーを押すと（ESC以外）エラー表示を行う。

3.5.2 ＩＣＥの起動，終了

(1) ＩＣＥ会話処理へのエントリー

モニタでのコマンド待ちの状態により"A／MLTICE"コマンドを打込むことによりＩＣＥ

会話処理ルーチンを呼び出し実行することができる。



ＦＤより会話ユニット及びＩＣＥユニットへのプログラムローディングを終了すると画面へＩＣＥ会話処理のコマンドメニューを表示し，コマンド待ち状態となる。

(2) ＩＣＥユニットのネーミング

各コマンドへ入る前に，会話ユニットと結合しているＩＣＥユニットのネーミングを行う必要がある。コマンドＡを押しイニシャライズ処理の中で各ユニットの名前を決定する。又ＩＣＥユニットと結合しているターゲットのＣＰＵ種別を設定する。ＩＣＥ会話処理では，結合ＩＣＥをこの名前により区別する。

(3) ＩＣＥ会話処理の終了

コマンドメニュー表示でＥＳＣキーを押すと会話ユニットと結合していたＩＣＥユニットを分離しモニタへ戻る。

3.6.3 各コマンドの内容

コマンドメニュー

ＩＣＥ処理で用意しているコマンドの内容を表示する。コマンドに対応しているＡ～Ｊのキーを押す事により各コマンドの実行へ移る。ＥＳＣキーを押すと，ＩＣＥ処理の終了とみなしモニタへ制御を移す。

```
###  Command menu  ###

     KEY          CONTENT

     (A) ------- Initialize
     (B) ------- Memory & Register
     (C) ------- I/O port & Register
     (D) ------- Single step
     (E) ------- Set trace condiotion
     (F) ------- Mapped memory
     (G) ------- Display trace data
     (H) ------- Emulation & Trace START/STOP
     (I) ------- File controle
     (J) ------- Data fill
     (ESC) ----- Exit
```

イニシャライズ

現在結合中のＩＣＥユニット番号，ＩＣＥユニットネーム，ＣＰＵの種別の表示を行う。又，ＩＣＥユニットネームおよびＣＰＵの種別の登録，変更を行う。

§ 画面内容

Available ＩＣＥ unit

結合中の全ＩＣＥユニットを表示する。ユニット番号はＩＣＥが個々に持つ固有の番号である。

Name

ＩＣＥ会話処理が各ＩＣＥユニットに与えるネームでユーザが任意に設定することができる。

ＩＣＥ会話処理において，ＩＣＥユニットは全てここで設定されたネームにより区別される。８文字のＡＳＣＩＩ文字を入力可能とする。

ＣＰＵ

結合中ＩＣＥのＣＰＵ種別を表わす。ＩＣＥ会話処理の開始時及びRequest ＩＣＥを行い新しくＩＣＥを結合した場合，結合中のＣＰＵの種別を設定しておく必要がある。

§ ユニット結合操作

イニシャライズ時

Name へユニットの名前を設定しＣＰＵへ結合ユニットのＣＰＵの種別を登録する。

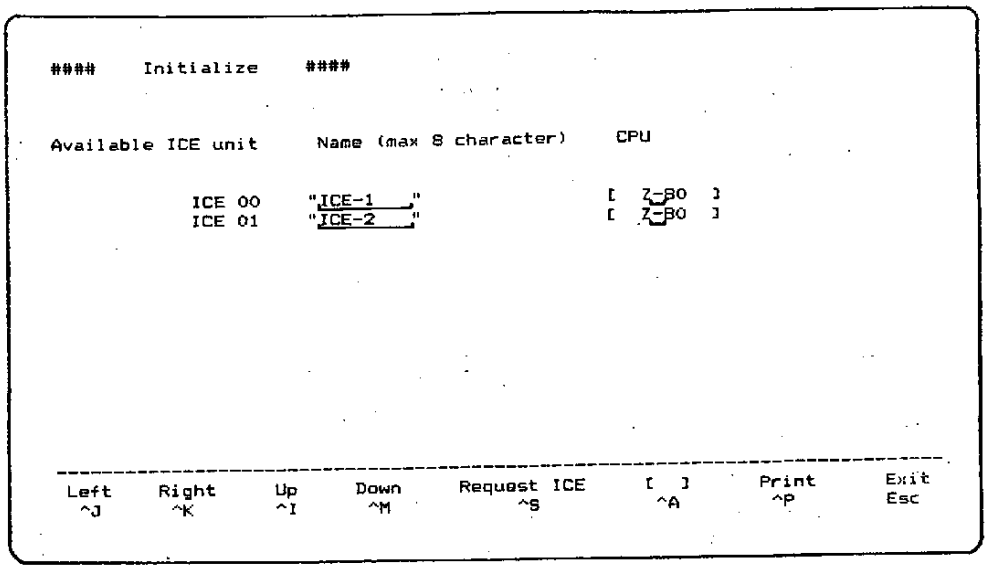
ＩＣＥ ＯＰＥＮ

現在使用中のＩＣＥユニットのほかに他の未使用のＩＣＥユニットを結合したい場合 Request ＩＣＥを行う。

新しく結合されたＩＣＥユニットが現在使用中のＩＣＥユニットの下に表示される。ユニットネームを設定し，ＣＰＵ種別を登録する。

ＩＣＥ　ＣＬＯＳＥ

現在使用中のＩＣＥユニットを使用中のＩＣＥ会話ユニットから分離したい場合．Name エリアをスペースとする。イニシャライズルーチンよりぬける時ＩＣＥユニットは分離される。



␣：カーソル移動可能エリア

メモリアンドレジスタ

モジュールネーム１及びモジュールネーム２で示されるターゲットシステムのメモリデータ及びレジスタデータの表示，並びに変更を行う。

§　画面内容

Module name

表示中データのＩＣＥユニットネームを示す。

Address

表示中メモリの中心アドレスを示す。カーソルがアドレスエリアよりぬけ出した時，又は１桁目のアドレスが更新された時，メモリ内容の更新を行う。

メモリ内容

Address で指定された番地の上下４０～４７バイト分のデータを８バイト単位で表示する。表示は１６進表示とＡＳＣＩＩ表示の両方を行う。データがＡＳＣＩＩ 表示不可能の場合“．”を表示する。

カーソルを変更するデータ上へ移動させ変更データを打込みデータの変更を行う。１６進及びＡＳＣＩＩによりメモリデータの変更が可能である。

レジスタデータ

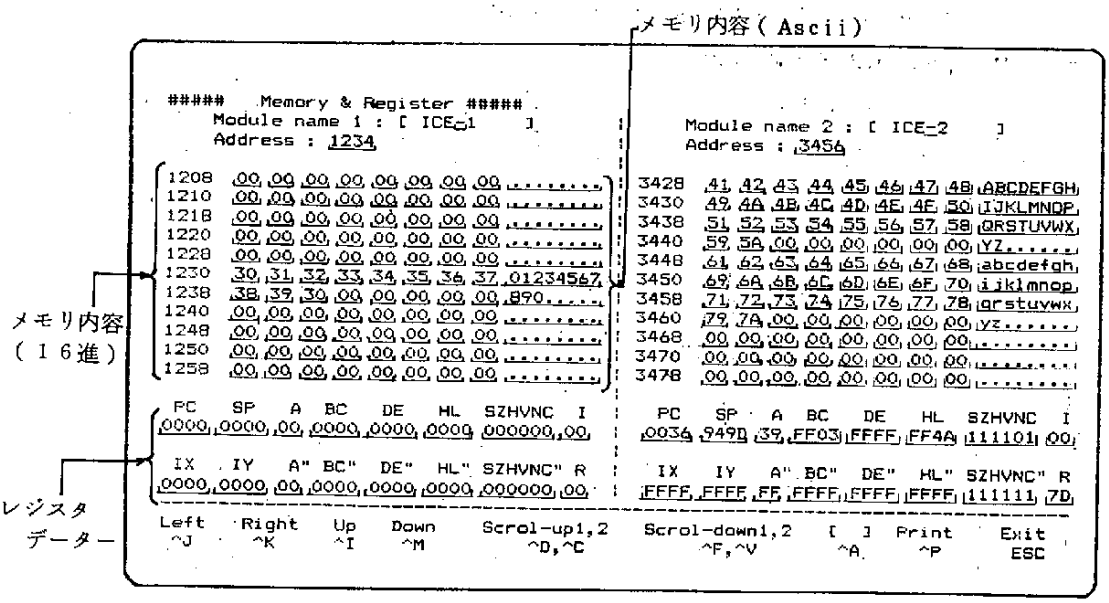
現在のレジスタの内容を１６進で表示する。フラグデータは“０”“１”のみ入力可能とする。

Scrol−up 1, 2

メモリ内容のスクロールアップを行う。

Scrol−down 1,2

メモリ内容のスクロールダウンを行う。



[ ]：カーソル移動可能エリア

Ｉ／Ｏポート＆レジスタ

モジュールネーム1.及びモジュールネーム2.で示されたＩＣＥユニットのＩ／Ｏポートから IN／OUTを実行し，入出力アドレス，入出力データの表示を行なう。（I／O ポートアドレスの表示はアドレスバスの１６ビットを行う。）

§ 画面内容

Module name

表示中データのＩＣＥユニットネームを示す。

Input port address

ＩＮを行う時のアドレスバスの内容を示す。

Output port address

outを行う時のアドレスバスの内容を示す。

Output port data

outを行う時のデータバスの内容を示す。

Input, Output 履歴

表示中のＩＣＥユニットで行ったＩＮ　ＯＵＴのアドレス及び入出力データを表示する。

レジスタデータ

memory and registerコマンドのレジスタ表示を参照

Input 1,2 (へD, へC)

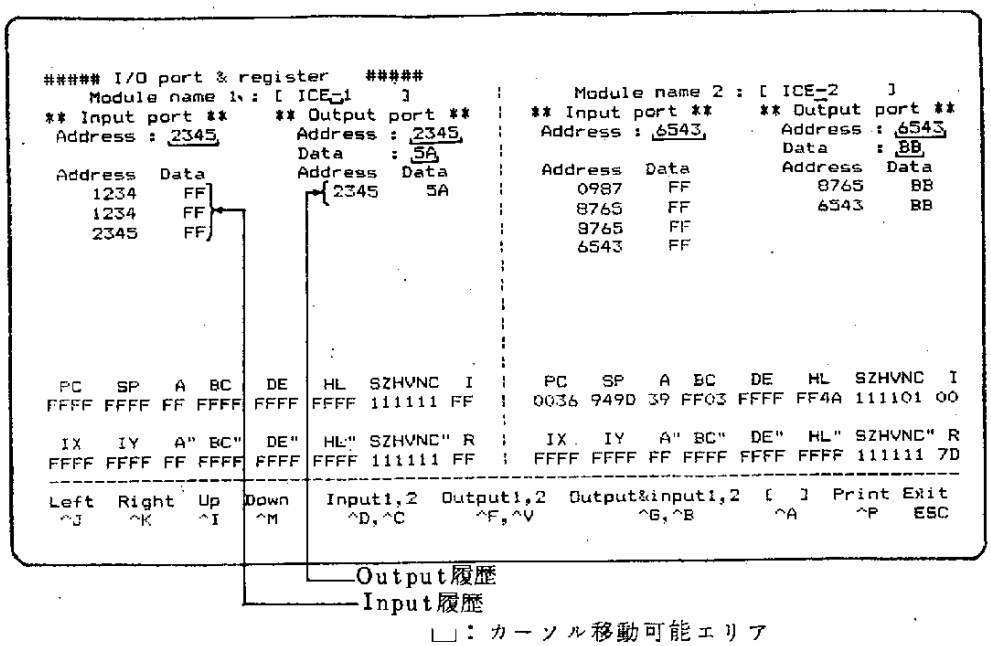
モジュール1,及びモジュール2.で示されるＩＣＥユニットでＩＮ命令を実行する。入力アドレス，入力データをinput 履歴へ出力する。

Output 1,2 (へF, へV)

モジュール1.及びモジュール2.で示されるＩＣＥユニットでout 命令を実行する。出力アドレス，出力データをoutput 履歴へ出力する。

Output & input 1,2 (へG, へB)

モジュール1.及びモジュール2.で示されるＩＣＥユニットでoutputのアドレス及びデータでout命令を実行し続いて同じアドレスでＩＮ命令を実行する。結果はoutput 履歴input 履歴へ出力する。



トレースコンディションセット

トレースデータの設定を行う。設定画面は２面あり，ＰＡＧＥにより画面を変更する。

§ 画面設定

module name

トレース条件を設定しようとするＩＣＥユニットのユニットネーム

＊＊＊Trace condition set＊＊＊

トレーサの動作モードの設定

① Start trace　トレーススタート条件成立よりトレースを開始し，トレースメモリFULLで停止する。

② End trace　トレーススタート条件成立よりトレースを始め，トレーストップ条件成立でトレースを終了する。
トレースメモリFULLで停止しないため，トレーストップ以前の２ｋ ward 分のデータしか保障されない。

以上の制限のもとで1度だけトレーススタートストップをする（Single trace），トレーススタートストップを繰り返す（itrative trace）があるが，これは使用者のトレース条件設定に依存するものである。

itrative trace では後述するPass counter trigger，Pass counter breakの値が初回だけ満足されるが，２回目以後はデフォルト値となる。

以上のようなトレーサの基本的な動作パターンを画面右上に表示する。

Pass counter ： trigger

トレーススタートを行うタイミングを決める。Trigger 条件成立をカウントし，何回目のTrigger でトレーススタートするかを表示する。

Pass counter ： break

トレーストップを行うタイミングを決める。break 条件成立をカウントし，何回目の break でトレーストップするかを表示する。

以上のPass counter のデフォルト値は００１で，そのときカウンタはないものとみなされる。

Delay counter

Delay counter のクロックを決める。トレーストップ条件成立後一定の間，トレースを続けたい場合のトレース時間または区間を決める。

Trace active

トレーサをトレース可能な状態にする。（トレース可能な状態とはtrace active からtrace inactive 迄であり，これらはevent A,B 及びEXT IN によって決定される。）

ストローブ表示

ターゲットシステムのアドレスバス，データバス，コントロールバスのサンプリングを各々のストローブ信号の立上りで行うか立下りで行うかを表示する。トレーサの状態はこのサンプリングされたデータで変化するが動作はtrace clock に選択されたストローブ信号に同期する。

＊＊＊ General counter set ＊＊＊

汎用的に使用するジェネラルカウンタの使用条件を設定する。

Clock source

クロックソースの規定を行う。ここでF５を指定すると画面の右に示すジェネラルカウンタ動作条件が有効となる。

ジェネラルカウンタ動作条件

A−comp, B−comp, trace active の右側の数値が " 1 " でそれぞれのソースをトリガー条件とし, event ON／OFFで" 1 "としたソースの組合わせをジェネラルカウンタのトリガー条件とする。画面の例ではB−comp の一致条件が成立した時ジェネラルカウンタをアップ／ダウンする。

count Up／Down

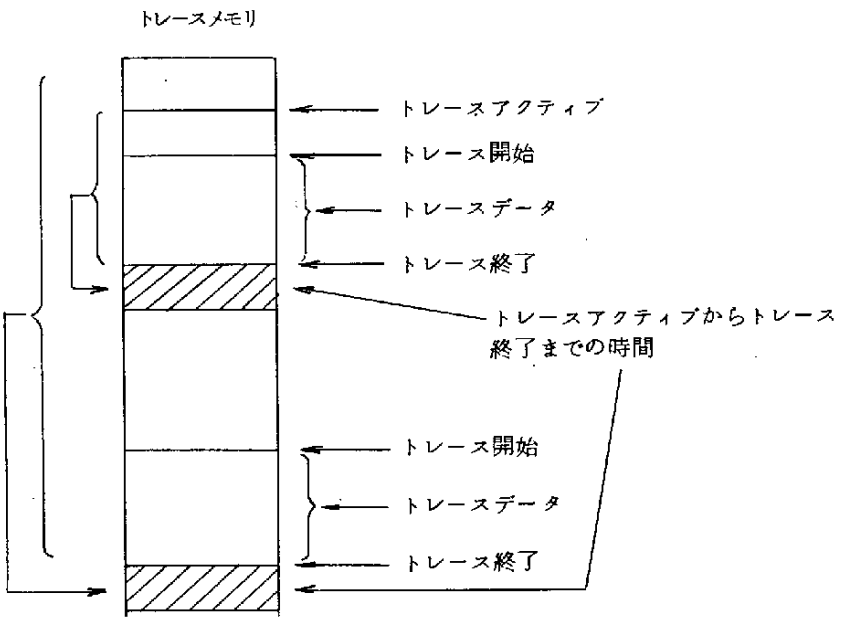
カウンタをupカウンタとして用いるかdown カウンタとして用いるかを規定する。

initial value

カウンタの初期値の設定

write time in trace memory

トレーサの動作終了時トレースアクティブになってからそこまでのジェネラルカウンタの値をメモリに書込む場合はONとする。



＊＊＊ Event A, B set ＊＊＊

イベントコンパレーターA, Bとバスライン（アドレスデータバス, コントロールバス）とを比較し, トレーサーのスタート条件, 終了条件を設定する。

Source

イベントコンパレータと比較するバスラインの組合わせを指定する。

universal は16ビットデータバスへの拡張を目的としたものであり，Z-80 では無視される。

Value

イベントコンパレータの値を定める。"X"を入力すると，そのビットは全ての値をトリガー条件とみなす。

＊＊＊ trace start condition ＊＊＊

＊＊＊ trace stop condition ＊＊＊

トレースの下記条件をA-comp，B-comp，external 1，external 2．の組合せにより指定する。

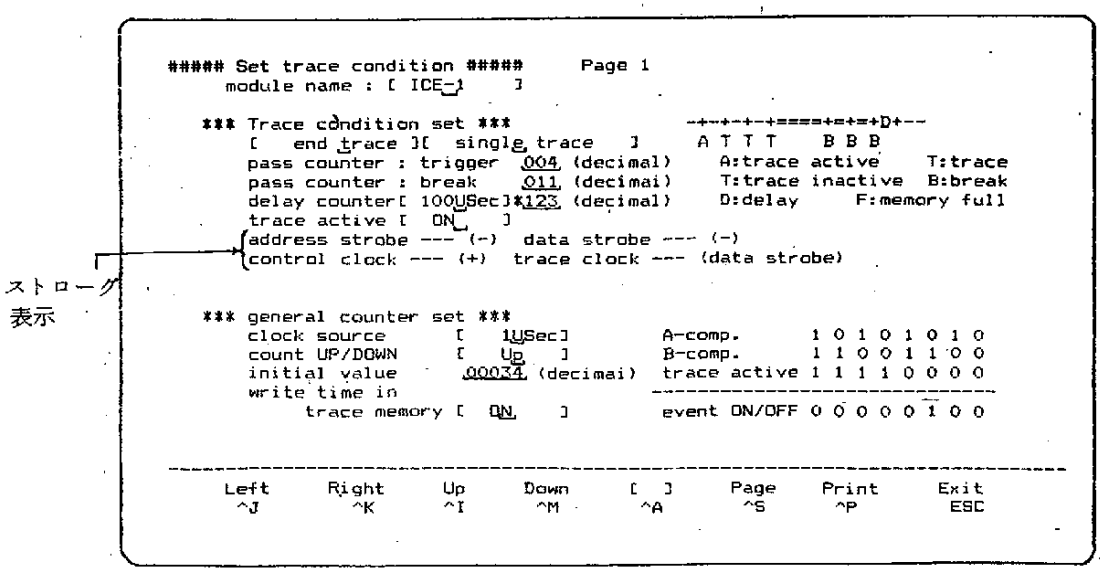
trace trigger	トレースのTrigger 条件。
trace break	トレースのBreak 条件。
trace active	トレース可能な状態条件。
trace inactive	トレース可能な状態条件の取消し。

設定方法はジェネラルカウンタ動作条件の設定と同様とする。

External out

external out 1,2 source

他のトレーサのEXTIN1，EXTIN2 としてどの信号を用いるかを設定する。



ストローク表示

␣：カーソル移動可能エリア

```
##### Set trace condition #####      Page 2
     module name : [ ICE-1     ]

   *** Event A,B set ***
  # event comparater A                    # event comparater B
     source --- [     address     ]          source --- [     address     ]
     value ---- 0000 H (don"t care:X)       value ---- 0000 H (don"t care:X)

  # trace start condition                 # trace stop condition
          A-comp.      1 0 1 0 1 0 1 0          A-comp.      1 0 1 0 1 0 1 0
          B-comp.      1 1 0 0 1 1 0 0          B-comp.      1 1 0 0 1 1 0 0
          external 1   1 1 1 1 0 0 0 0          external 2   1 1 1 1 0 0 0 0
         ---------------------------------      ---------------------------------
     trace trigger     0 0 0 0 0 0 0 0      trace break       0 0 0 0 0 0 0 0
     trace active      0 0 0 0 0 0 0 0      trace inactive    0 0 0 0 0 0 0 0

   *** External out ***
        external out 1 source [  trace start ]
        external out 2 source [  trace start ]

-------------------------------------------------------------------------------
     Left      Right      Up       Down      [  ]      Page      Print      Exit
      ^J        ^K        ^I        ^M        ^A        ^S        ^P         ESC
```

␣：カーソル移動可能エリア

シングルステップ

モジュールネームで示されるターゲットシステムの１命令を実行し，結果を逆アセンブルし表示する。

§ 画面内容

Module name

表示中データのＩＣＥユニットネームを示す。

Display mode

実行結果の表示形式を指定する。表示は次の２種類とする。

① instruction Register

命令の逆アセンブル表示と現在のレジスタ内容の表示を行う。

② Trace data Register

命令の逆アセンブル表示に加えてオペランドのフェッチデータやメモリ I/O のリードライトサイクルを表示する。

実行結果

実行された順に逆アセンブルデータを表示する。最下行に表示されている命令は次に実行する予定の命令である。ＰＣを変更した場合，最下行の命令は実行されずＰＣが示す命令を実行する。

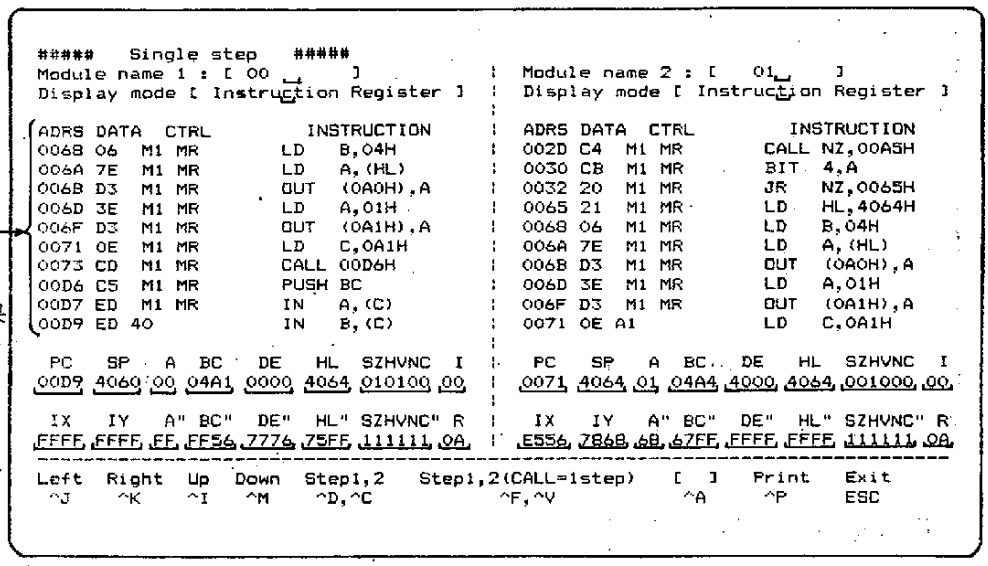
CTRLはコントロールバスの内容（M1, READ, WRITE, I/O）を示す。

Step 1,2 （←D, ←C）

モジュール1,又はモジュール2.で示しているＩＣＥユニットのステップ動作を行う。

Step 1,2 (call＝1step) (^F, ^V)

CALL命令又はブロック転送命令により実行されたルーチンを1ステップとみなし，CALL命令の実行結果は表示しない。



␣: カーソル移動可能エリア

マップドメモリ

ICEユニットに用意されている4Kバイト×2のマップドメモリの使用条件を指定する。

§ 表示内容

Module name

表示中のICEユニットネームを示す。

Mapped memory

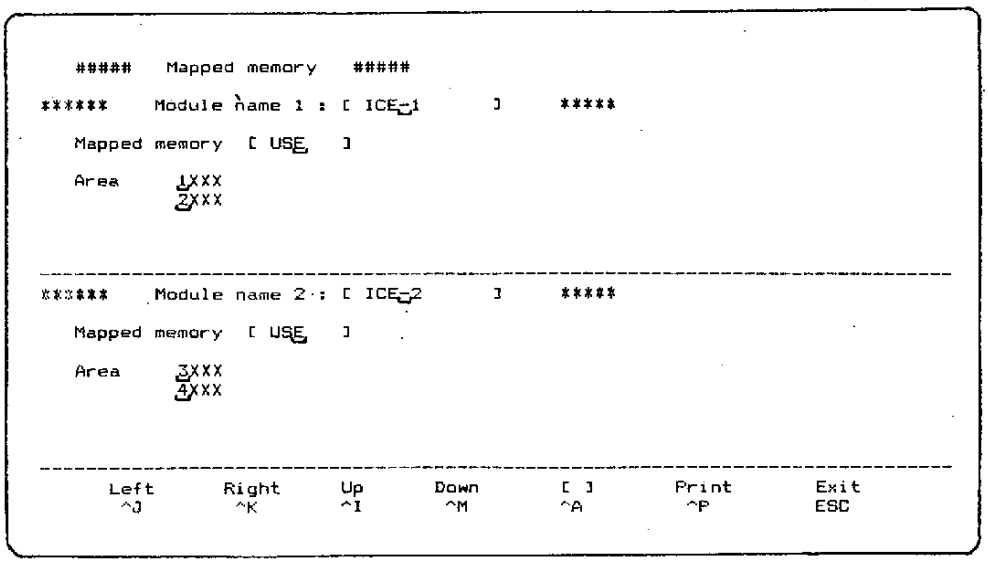
Area で表示するメモリ空間へマップドメモリを設定するかしないかを指定する。USE又は UNUSE へ変化させた時点でマップドメモリの設定が変化する。

Area

マップドメモリを使用する場合のアドレスを指定する。使用状態をUSEとしAreaを1×××とすると1000(H)～1FFF(H) へマップドメモリが設定される。

アドレスへ"X"を入力するとそのエリアはUNUSEとなり画面よりアドレスが消える。

アドレスを変更した時点でマップドメモリの設定は変更される。



␣：カーソル移動可能エリア

ディスプレイトレースデータ

トレーサに蓄えられたトレースデータを逆アセンブルし表示する。

§ 表示内容

Module name

表示中のＩＣＥユニットネームを示す。

トレースデータ表示エリア

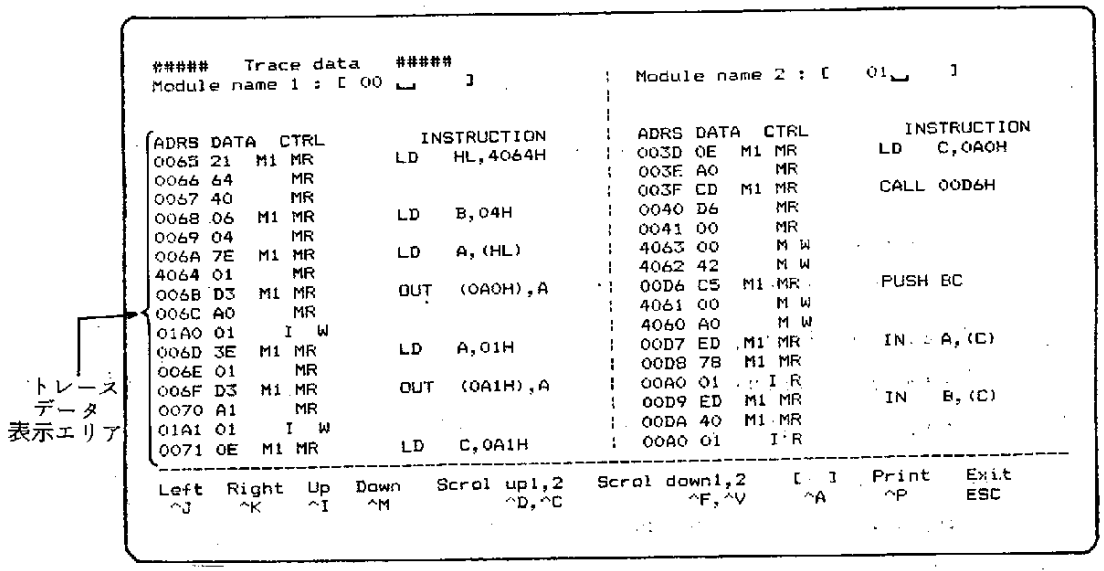
表示形式はシングルステップの instruction Regist と同様とする。

Scrol up 1,2 （＾D，＾C）

画面をスクロールアップし，トレースデータの上位アドレスのデータを表示する。

Scrol down 1,2 （＾F，＾V）

画面をスクロールダウンし，トレースデータの下位アドレスのデータを表示する。



␣：カーソル移動可能エリア

エミュレーショントレーススタート／ストップ

ＩＣＥユニットのエミュレーション又はトレースのスタート・ストップを指定する。

§ 画面内容

Name

現在のＩＣＥ会話システムに結合しているＩＣＥユニットのユニットネームを表示する。

ＰＣ　現在のＰＣの値を示す。

Group

〔ON〕となっている，ＩＣＥユニットを１つのグループとみなす。このグループで同時にエミュレーションのスタートストップを行う。

Group master

グループエミュレーションを行う場合のスタート又はストップの原因となるモジュールを決める。

Concurrent stop with trace

トレースの終了によりエミュレーションをストップするかどうかの設定を行う。

Emulation

ターゲットプロセッサがＲＵＮ状態かＨＡＬＴ状態かを表示する。

Trace

トレーサがトレースをしているかどうかを表示する。

Emulation :〔　　〕

シングルエミュレーションを行う場合のユニットネームを〔　〕内に表示する。

Trace　　:〔　　〕

シングルトレースを行う場合のターゲットシステムのユニットネームを〔　〕内に表示する。

Page （へD）

結合ＩＣＥが８個以上で１画面に入らない場合，別のページの呼び出しを行う。

Emulation （Group）

Start　Group で ON を指定したユニットのエミュレーションを開始する。

Stop　上記のエミュレーションを強制的にストップする。

Emulation （Single）

Emulation :〔　〕の項で指定したユニットのエミュレーションを強制的にスタート・ストップする。

Trace

trace :〔　〕の項で指定したユニットのトレースを強制的にスタートストップする。

```
#####    Emulation & Trace   Start / Stop     #####

Name        PC      Group    Group master   Concurrent Stop     Emulation   Trace
                                             whith trace

ICE-1     FFFF    [ ON  ]    [ OFF ]          [ OFF ]            HALT      DONE
ICE-2     0038    [ ON  ]    [ ON  ]          [ ON  ]            HALT      DONE




Emulation : [ ICE-1    ]   START (V)  or  STOP (N)
Trace     : [ ICE-1    ]   START (M)  or  STOP (,)
-------------------------------------------------------------------------------
      Left     Right      Up       Down      [  ]        Print      Exit      Page
       ^J       ^K        ^I        ^M        ^A          ^P         ESC        ^D

     Emulation(Group)              Emulation (Single)              Trace
      START     STOP               START     STOP              START     STOP
        Z         X                  V         N                 M
```

␣：カーソル移動可能エリア

ファイルコントロール

ＦＤ内の実行可能プログラム（属性＝ＡＢ．）をターゲットへロードする。

§　画面内容

Ｆｕｎｃｔｉｏｎ　　機能の説明

Ａｖａｉｌａｂｌｅ　ｆｉｌｅ　ｎａｍｅ

ロードしようとするＦＤ内のファイル名

ＩＣＥ　Ｕｎｉｔ　ｎａｍｅ

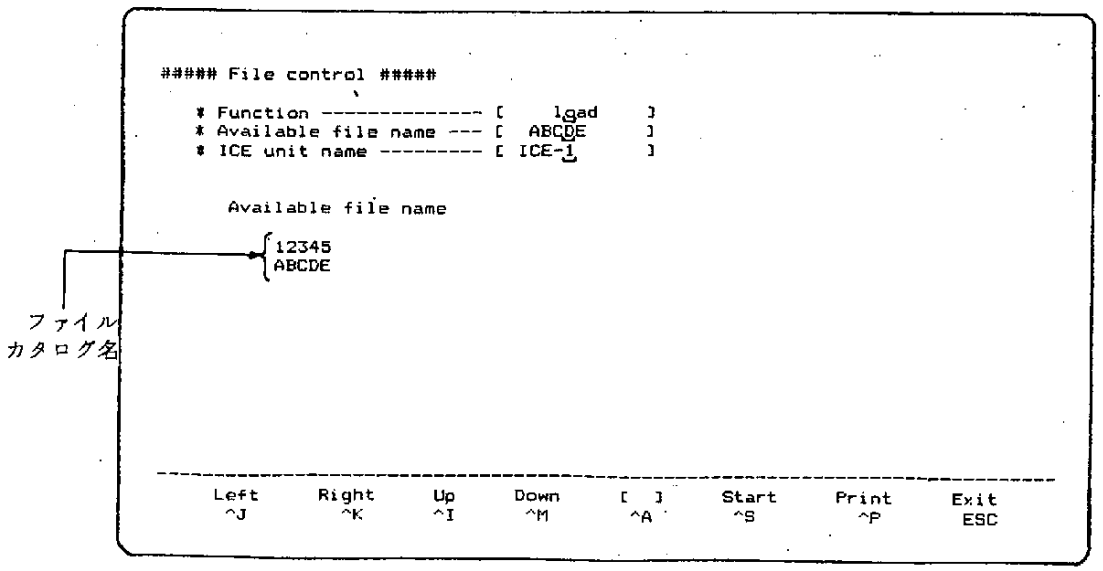
ロードを行おうとするターゲットのＩＣＥユニット名

ファイル・カタログ名

ＦＤ内に作成されている使用可能なファイルのリスト

Ｓｔａｒｔ

実行を開始する。実行中は＜Ｐｒｏｓｅｃｕｔｉｎｇ＞を表示し正常終了時に＜Ｓｕｃｃｅｓｓ＞を表示する。



␣：カーソル移動可能エリア

データフィル

ターゲットシステムのＲＡＭエリアを指定データでうめる。

§　画面内容

module name

対象となるターゲットが，結合しているユニットネーム

set data　　書込みデータ

start address

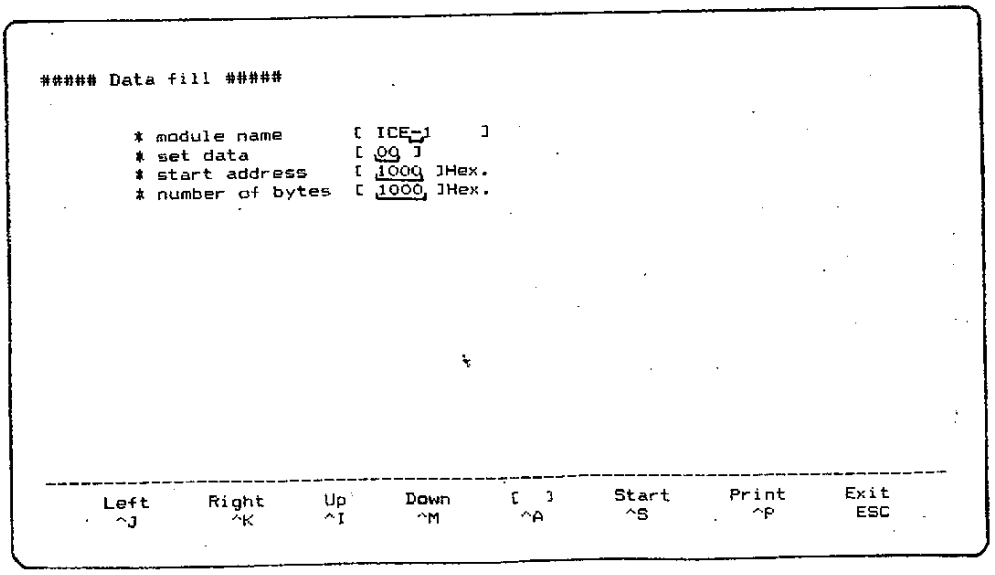
書込みを開始しようとするアドレス

number of byte

書込みバイト数

Start（^S）

書込みを開始する。実行中は，＜prosecuting＞を表示し，正常終了時に＜success＞を表示する。



␣：カーソル移動可能エリア

3.6 使 用 例

新しくプログラムを作成し，AB.ファイルを作る場合の本システムの使用例を以下に述べる。尚，AB。ファイルは，ＩＣＥおよびユーティリティ'ABLOAD'を使って，実行，デバッグができる。

(1)　プログラム'MAIN'を新しく作成し，アセンブルして，MAIN(RB.)を作成する。

(2)　MAIN(RB。)，SUB1(RB.)，SUB2(RB.)をリンクしてAB.形式のファイルを作成する。

ただし，SUB1，SUB2(RB)はあらかじめ，ユーザ用ディスケット，ボリューム名，'A'上に作成されたプログラムモジュールであると仮定する。

以上の(1)，(2)のステップに従って，プログラムの作成を行う。

3.6.1 概　要

(1) 電源ON～モニタ起動

メインスイッチをONにし，ブート用ディスケット，システムディスケットをドライブユニットに挿入し，モニタを起動する。

(2) ユーザ用ディスケット(ボリューム'A')のオープン

ACCESS-MOUNTを実行し，ユーザ用ディスケットをアクセスできるようにする。単にディスケットをセットしただけでは，アクセス可能とならない。

(3) テキストエディタによるプログラム'MAIN'作成

ソースプログラムをキーボードより入力し，テンポラリファイルTEMP1(ASC)へ出力する。

○モニタ用ディスケットには，次のテンポラリーファイルがある。

TEMP1(ASC)，TEMP1(RB.)

TEMP1(AB.)，TEMP2(ASC)

ユーザは，テンポラリーファイルを自由に利用できる。

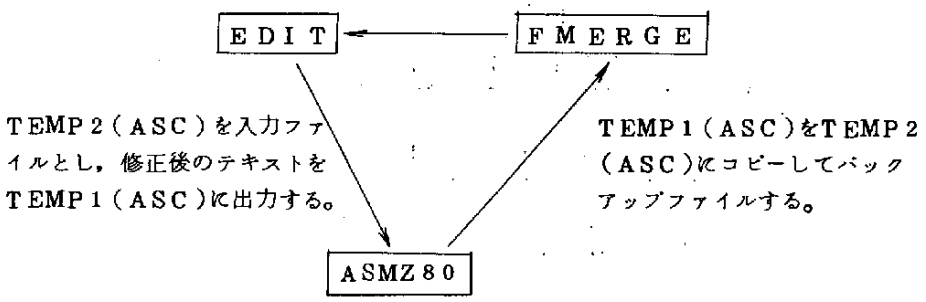
(4) 'MAIN'のアセンブル

TEMP1(ASC)に入っているソースプログラムをアセンブルし，エラーがなければRB.を，TEMP1(RB.)アセンブルリストを，LIST(LST)へ出力する。

(5) アセンブルエラーが発生した場合

ユーティリティ'FMERGE'を使って，TEMP1(ASC)に入っているソースプログラムを，TEMP2(ASC)に移す。

○この例のように，1つのプログラムのエディト，アセンブルを反復する時，TEMP2(ASC)が，TEMP1(ASC)のバックアップファイルとなるように利用すると便利である。



(6) アセンブルエラーがない場合

エラーをすべて取りさった後，ソースプログラム及びアセンブラで作成されたRB．を，パーマネントファイルにして，保存する。

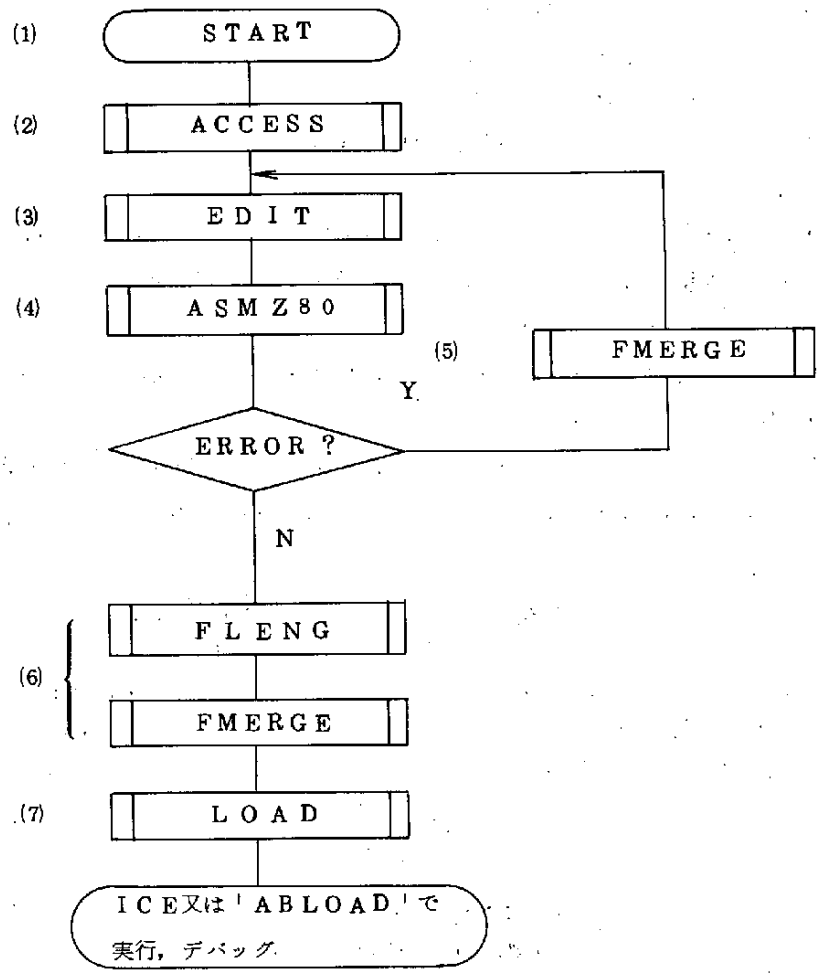
ユーティリティ'FLENG'を使って，TEMP1(ASC)及び(RB．)のサイズを測定し，それに応じた長さのパーマネントファイルMAIN(ASC)，(RB．)を，ユーティリティ'FMERGE'で作り，TEMP1(ASC)，(RB。)をそれぞれコピーする。

(7) RB．をリンクしてAB．を作成する。

(6)で作成したMAIN(RB．)と，すでに作成されているSUB1(RB．)，SUB2(RB．)をリンクして，AB．を作成し，TEMP1(AB．)へ出力する。

3.6.2 操作フローチャート

操作手順のフローチャートを示す。(番号は概説と対応している。)



3.6.3 実行画面

電源ＯＮからの画面を以下に示す。

左側に付けた番号は概説と対応している。

```
###  COLD-START OR HOT-START (C/H) ? C
                                ;コールドスタート
###  SET IPL-DISK & HIT ANY KEY

(1)
***  MONITOR OR DEBUGGER (M/D) ? M
                                ;モニタ起動
## DOS-BOOT ##
SYSTEM-ID =DIGI                 }モニタDisk の利用資格チェック
PASS-WORD =37-1101
DATE (YYMMDD) =820303

## SYSTEM ?   ACCESS            ;ユーティリティコマンド'ACCESS'
##### FILE ACCESS UTILITY #####

 FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
 1D ACCESS BODY            $$SYS/$ACCESS          HEX     R    0001   SUCCESS



  FUNCTION = MOUNT          ;MOUNTの起動

(2)
UNIT NO      = 1            ;今からOpenするDiskの入ったドライブユニット番号
VOLUME NAME = A             ;ボリューム名
USER ID.     = A            ;ユーザID
PASS WORD    = XXXXXXXX     ;パスワード'A'入力後'X'でかくされる

** SUCCESS **

  FUNCTION =                ;リターンキーでシステムに戻る。
## SYSTEM ?   EDIT          ;'EDIT'を用いてプログラムを作成する。
##### EDITER #####

 FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W  BUFF.  STATUS
 00 OUTPUT FILE            $$SYS/TEMP1            ASC     W   0007   SUCCESS
 01 INPUT FILE
 1D EDITER BODY            $$SYS/$EDIT            HEX     R   0001   SUCCESS

(3)
****** EDITOR-Z-F ******

*I;*** MAIN ***
          .TITLE   MAIN     }
          .GLOBL   SUB1     }
MAIN      LD       C,3      }
          LD       B,2      }
LOP1:     CALL     SUB1     }  テキスト入力
          DJNZ     LOP2     }
          CALL     SUB2     }
          HALT              }
          .END              }
```

```
*B$22T
;*** MAIN ***
        .TITLE  MAIN
        .GLOBL  SUB1
MAIN    LD      C,3
        LD      B,2
LOP1:   CALL    SUB1
        DJNZ    LOP2
        CALL    SUB2
        HALT
        .END
* [^P]

## PUT TEXT ##
```

入力されているか確認

コントロール'P'を入力してTEMP1(ASC)にテキストバッファの内容を出力する。

```
 ## SYSTEM ?   ASMZ80
 ##### ASSEMBLER Z-80 #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W  BUFF.  STATUS
 00 RB-FILE                $$SYS/TEMP1           RB.    W    0001   SUCCESS
 01 LIST-FILE              $$SYS/LIST            LST    W    0008   SUCCESS
 02 SOURCE-FILE            $$SYS/TEMP1           ASC    R    0005   SUCCESS
 03 SOURCE-FILE
 1D ASSEMBLER BODY         $$SYS/$ASMZ80         HEX    R    0001   SUCCESS

(4) *** CRAMBO-Z-F Z-80 VERSION ***


SOURCE TEXT NAME ? :[CR]
DELIMITER NORMAL(0) OR NOT(1) ? :  0
 PASS-1 START
E-11    4.                                MAIN    LD      C,3

 PASS-1 END
ERROR EXIST
 CONTINUE(0) OR NOT(1) ? :  1

## SYSTEM ?   FMERGE
 ##### FILE MERGE #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W  BUFF.  STATUS
 00 OUTPUT FILE            $$SYS/TEMP2           ASC    W    0000   SUCCESS
 01 INPUT FILE             $$SYS/TEMP1           ASC    R    0008   SUCCESS
 02 INPUT FILE
(5) 1D F-MERGE BODY        $$SYS/$FMERGE         HEX    R    0001   SUCCESS

 ### FILE-MERGE (COPY) ###
MERGE  FC=  1

 FILE SIZE (BYTE)     00185
 RECORD COUNT            10

## SYSTEM ?   EDIT
 ##### EDITER #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W  BUFF.  STATUS
 00 OUTPUT FILE            $$SYS/TEMP1           ASC    W    0007   SUCCESS
(3)' 01 INPUT FILE         $$SYS/TEMP2           ASC    R    0007   SUCCESS
 02 INPUT FILE
 1D EDITER BODY            $$SYS/$EDIT           HEX    R    0001   SUCCESS

****** EDITOR-Z-F ******
```

;作成したプログラムをアセンブルする。

;ソースプログラム内でTITEL文を用いてモジュール名'MAIN'を付けてあるので不要

;エラーが発生したのでパス2は実行しない。

;TEMP1(ASC)をTEMP2(ASD)にコピーする。

;エラーを修正する。

```
*10T
;*** MAIN ***
        .TITLE  MAIN
        .GLOBL  SUB1
MAIN    LD      C,3
        LD      B,2
LOP1:   CALL    SUB1
        DJNZ    LOP2
        CALL    SUB2
        HALT
        .END
*3L2T
MAIN    LD      C,3
        LD      B,2
*L2T
*CMAIN$MAIN:
*L2T
MAIN:   LD      C,3          } エラーの修正
      LD      B,2            }
*C  $
*L2T
MAIN:   LD      C,3          } 確認
        LD      B,2          }
* [^P]

## PUT TEXT ##
```

```
## SYSTEM ?   ASMZ80          ;(4)と同様の作業
##### ASSEMBLER Z-80 #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB. R/W  BUFF. STATUS
 00 RB-FILE                $$SYS/TEMP1           RB.   W    0001  SUCCESS
 01 LIST-FILE              $$SYS/LIST            LST   W    0008  SUCCESS
 02 SOURCE-FILE            $$SYS/TEMP1           ASC   R    0005  SUCCESS
 03 SOURCE-FILE
 1D ASSEMBLER BODY         $$SYS/$ASMZ80         HEX   R    0001  SUCCESS

*** CRAMBO-Z-F Z-80 VERSION ***
```

(4)'

```
SOURCE TEXT NAME ? :
DELIMITER NORMAL(0) OR NOT(1) ? :  0
 PASS-1 START

 PASS-1 END

 PASS-2 START
E-13    7.  0007  1000      A  0009 T          DJNZ    LOP2
E-13    8.  0009  CD0000    A  0000 A          CALL    SUB2
 PASS-2 END
ERROR EXIST

 R.B. OUT(0) OR NOT(1) ? :  1
```

```
## SYSTEM ?   FMERGE          ;(5)と同様の作業
##### FILE MERGE #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB. R/W  BUFF. STATUS
 00 OUTPUT FILE            $$SYS/TEMP2           ASC   W    0008  SUCCESS
 01 INPUT FILE             $$SYS/TEMP1           ASC   R    0008  SUCCESS
 02 INPUT FILE
 1D F-MERGE BODY           $$SYS/$FMERGE         HEX   R    0001  SUCCESS
```

(5)'

```
### FILE-MERGE (COPY) ###

MERGE  FC=  1

 FILE SIZE (BYTE)      00185
 RECORD COUNT             10
```

```
## SYSTEM ?   EDIT          ;(3), (3)' と同様
##### EDITER #####

 FC.  *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
 00 OUTPUT FILE           $$SYS/TEMP1           ASC    W     0007   SUCCESS
 01 INPUT FILE            $$SYS/TEMP2           ASC    R     0007   SUCCESS
 02 INPUT FILE
 1D EDITER BODY           $$SYS/$EDIT           HEX    R     0001   SUCCESS

****** EDITOR-Z-F ******

*10T
;*** MAIN ***
        .TITLE  MAIN
        .GLOBL  SUB1
MAIN:   LD      C,3
        LD      B,2
LOP1:   CALL    SUB1
        DJNZ    LOP2
        CALL    SUB2
        HALT
        .END
(3)'*CSUB1$SUB1,SUB2
*L2T
        .GLOBL  SUB1,SUB2
MAIN:   LD      C,3
*CLOP2$LOP1
*L2T
        DJNZ    LOP1
        CALL    SUB2
*L10T
        DJNZ    LOP1
        CALL    SUB2
        HALT
        .END
*S$L10T
;*** MAIN ***
        .TITLE  MAIN
        .GLOBL  SUB1,SUB2
MAIN:   LD      C,3
        LD      B,2
LOP1:   CALL    SUB1
        DJNZ    LOP1
        CALL    SUB2
        HALT
        .END


## PUT TEXT ##
```

```
(4)'## SYSTEM ?   ASMZ80          ;(4), (4)' と同様
##### ASSEMBLER Z-80 #####
```

```
 FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
 00 RB-FILE                 $$SYS/TEMP1           RB.     W    0001   SUCCESS
 01 LIST-FILE               $$SYS/LIST            LST     W    0008   SUCCESS
 02 SOURCE-FILE             $$SYS/TEMP1           ASC     R    0005   SUCCESS
 03 SOURCE-FILE
 1D ASSEMBLER BODY          $$SYS/$ASMZ80         HEX     R    0001   SUCCESS

*** CRRMBO-Z-F Z-80 VERSION ***


SOURCE TEXT NAME ? :
DELIMITER NORMAL(0) OR NOT(1) ? :  0
 PASS-1 START

 PASS-1 END

 PASS-2 START

 PASS-2 END

 R.B. OUT(0) OR NOT(1) ? :  0              ;エラーがなかったのでRB.を出力する。
```

```
## SYSTEM ?   FLENG                        ;アセンブルエラーのなくなったテキストのサイズを測定する。
##### FILE LENGTH MESUREMENT #####

 FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
 00 MESURE FILE             $$SYS/TEMP1           ASC     R    0014   SUCCESS
 1D MESURE BODY             $$SYS/$FLENG          HEX     R    0001   SUCCESS


### FILE-LENGTH ###

* FILE LENGTH   =     00190   (     1 BLOCK )
* RECODE COUNT =          10
** TOTAL LENGTH (BLOCK) =     1          ;測定結果テキストの長さは1ブロックである。


## SYSTEM ?   FMERGE                       ;ユーザDisk上に新ファイルMAIN(ASC)を作るTEMP1
##### FILE MERGE #####                     (ASC)の内容をコピーする。

(6)
 FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
 00 OUTPUT FILE             @A/MAIN               ASC
                                 ##### FILE SIZE (BLOCK) =3          SUCCESS
 00 OUTPUT FILE             A/MAIN                ASC     W    0008   SUCCESS
 01 INPUT FILE              $$SYS/TEMP1           ASC     R    0008   SUCCESS
 02 INPUT FILE
 1D F-MERGE BODY            $$SYS/$FMERGE         HEX     R    0001   SUCCESS

### FILE-MERGE (COPY) ###                 'FLENG'での測定結果を参考にして，新ファイルMAIN(ASC)のサイズを決める。

MERGE  FC=  1
FILE SIZE (BYTE)      00190
RECORD COUNT             10


## SYSTEM ?   FLENG                       ;アセンブラで作成したRB.ファイルのサイズを測定する。
##### FILE LENGTH MESUREMENT #####

 FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
 00 MESURE FILE             $$SYS/TEMP1           RB.     R    0014   SUCCESS
 1D MESURE BODY             $$SYS/$FLENG          HEX     R    0001   SUCCESS
```

```
### FILE-LENGTH ###

* FILE LENGTH   =      00258    (     1 BLOCK )
* RECODE COUNT =           5
** TOTAL LENGTH (BLOCK) =     1


## SYSTEM ?    FMERGE                          ;新ファイルMAIN(RB.)にTEMP1(RB.)をコピーする。
##### FILE MERGE #####

 FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF   STRTUS
 00 OUTPUT FILE            @A/MAIN                RB.
                                ##### FILE SIZE (BLOCK) =3          SUCCESS
 00 OUTPUT FILE            A/MAIN                 RB.    W     0008   SUCCESS
 01 INPUT FILE             $$SYS/TEMP1            RB.    R     0008   SUCCESS
 02 INPUT FILE
 1D F-MERGE BODY           $$SYS/$FMERGE          HEX    R     0001   SUCCESS

### FILE-MERGE (COPY) ###

MERGE  FC=  1

 FILE SIZE (BYTE)      00250
 RECORD COUNT              5
```

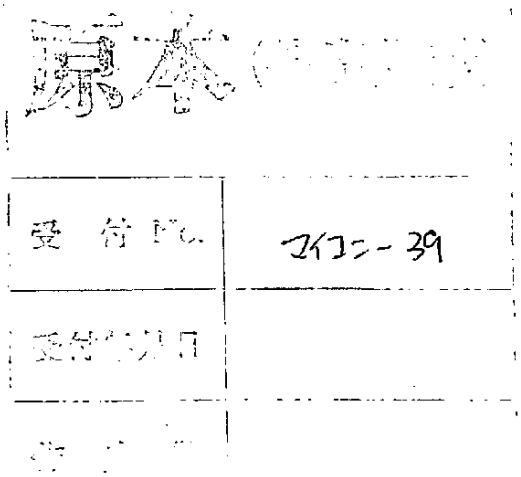
```
## SYSTEM ?    LOAD                            ;今作成したMAIN(RB.)と以前に作成してあった，SUB1
##### RELOCATABLE LORDER #####                  (RB.), SUB2(RB.)をリンクしてAB.を作成する。

 FC.   *** COMMENT ***      *** U/F-NAME ***     ATRB.  R/W   BUFF.  STATUS
 00 AB-OUTPUT FILE         $$SYS/TEMP1            AB.    W     0002   SUCCESS
 01 SYMBOL TABLE           $$SYS/SYMBOL           ASC    W     0001   SUCCESS
 02 RB-INPUT FILE          A/MAIN                 RB.    R     0002   SUCCESS
 03 RB-INPUT FILE          A/SUB1                 RB.    R     0001   SUCCESS
 04 RB-INPUT FILE          A/SUB2                 RB.    R     0001   SUCCESS
 05 RB-INPUT FILE
 1D LOADER BODY            $$SYS/$LOAD            HEX    R     0001   SUCCESS
****** LOADER-Z-F START ******


 MODULE NAME : MAIN                              ;モジュール

 START ADDRESS :

(7)  B-SECTOR :   0000                           ;B-SECTORは使用しない〔CR〕のみ入力
     T-SECTOR :   4000   OK ?   Y                ;T-SECTORはX'4000を先頭アドレスにする。


 START-END ADDRESS :

  T-SECTOR :   4000-400C                         ;モジュール名'MAIN'が使用した領域

 MODULE NAME : SUB1                              ;次のモジュール名'SUB1'

 START ADDRESS :

    FUNCTION =                                   ;リターンキーでシステムに戻る。
 ## SYSTEM ?    EDIT                            ;'EDIT'を用いてプログラムを作成する。
 ##### EDITER #####

    B-SECTOR :   0000                            ;〔CR〕入力(使用せず)
    T-SECTOR :   4100   OK ?   Y                 ;X'4100を先頭アドレスにする
                                                  (デバッグする時に便利)
```

```
START-END ADDRESS :

 T-SECTOR :   4100-4102          ; 'SUB1' が使用した領域

MODULE NAME : SUB2               ;次のモジュール名 'SUB2'

START ADDRESS :

 B-SECTOR :   0000               ; [CR] 入力(使用せず)
 T-SECTOR :   4103               ; SUB1に続くアドレスへリンクするため [CR] を入力


START-END ADDRESS :

 T-SECTOR :   4103-4105          ; 'SUB2' が使用した領域
SYMBOL ERROR : NOT EXIST         ) シンボルエラーがあればここに表示される。又，簡単な修正もできる。


 OBJECT OUT ?  Y                 ; AB.の出力
 ALL DUMP ?  Y                   ; リンクした領域をすべて出力する
 ADDRESS :   4000-400C           ) 出力された領域
 ADDRESS :   4100-4105

 SYMBOL TABLE OUT ?  Y           ; シンボルテーブルを出力する。
 SYMBOL ERROR : NOT EXIST
```

昭和57年3月発行

発行所　財団法人　日本情報処理開発協会
東京都港区芝公園3−5−8
機械振興会館内
TEL（434）8211（代表）

印刷所　山陽株式会社
TEL（591）0248



受付No.	コイコン-39
受付年月日	
備考	