資 料

論理回路作成支援システム

取扱い説明書

昭 和 60 年 3 月

JIPDEC

財団法人 日本情報処理開発協会

この資料は、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５９年度に実施した「マイクロコンピュータの利用に関する共通的な技術開発」の一環としてとりまとめたものであります。

目　　次

1. 概　　要

1.1　本装置の目的

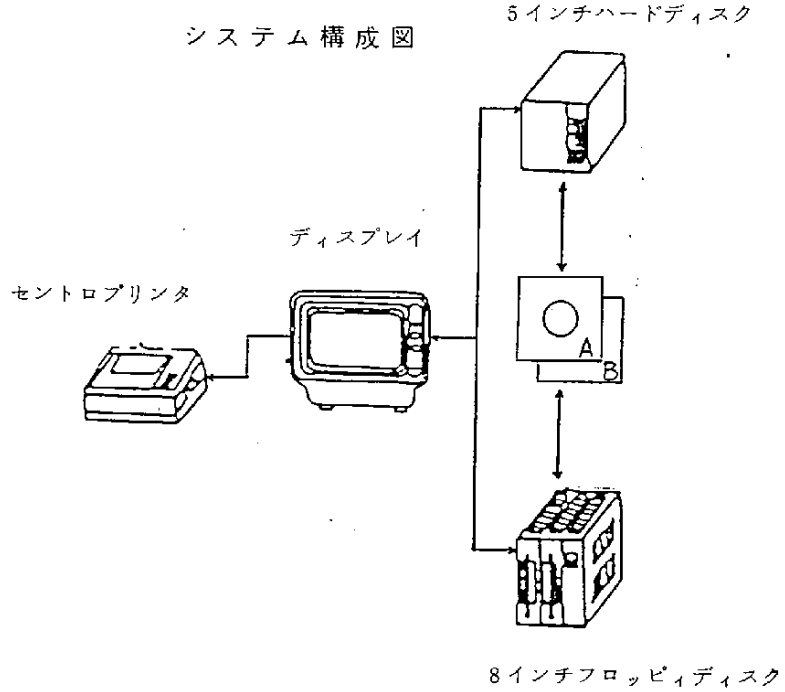
今回、１６ビットのマイクロコンピュータをベースにした"1000ゲート以下のＩＣの論理回路シミュレーションを行なう"ことを目的としたＩＣ／回路作成支援システムを開発しました。

本装置は、標準ＯＳの採用と１６ビットパソコンの今後の発展性を考慮しますと、機能拡張することによりさらに多くの要求に対応することができ、時代のニーズに応えることができると確信します。

ディジタル機器の開発に際して、本装置は特に下記の設計に対応できます。論理設計、計算機構成、計算機方式の各シミュレーションに使用されます。

1.2　システム構成

システム構成図

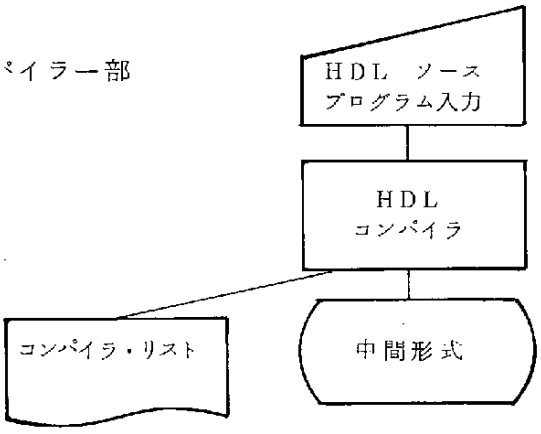


〈注〉　ディスクは5インチハードディスク又は8インチフロッピィディスクを使用して下さい。

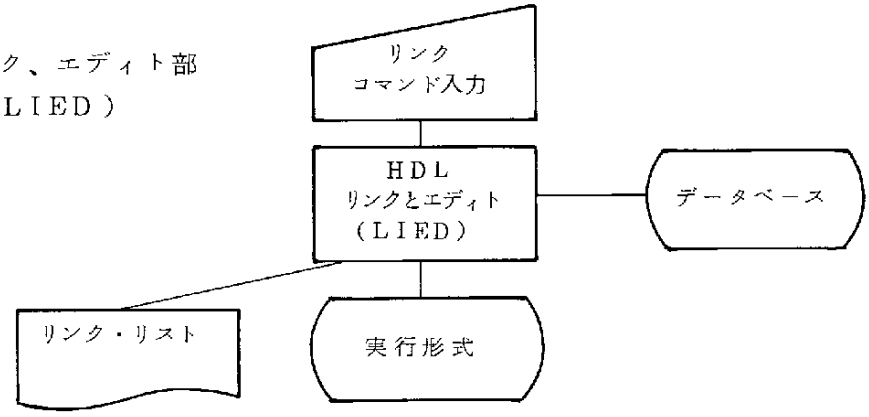
1.3 処理の概要

処理フローの概要を下図に示します。

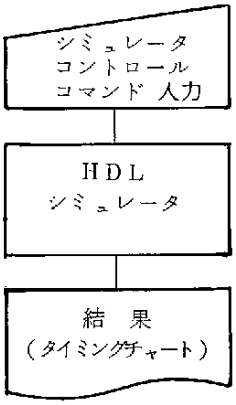
(1) コンパイラー部



(2) リンク、エディト部
(LIED)



(3) シミュレータ部



本装置は、ＭＳ－ＤＯＳの管理化にあり、ＣＰＭエミュレータの下で動作します。入力は、システムプログラムとしてのエディタにより入力されます。

回路を、ＨＤＬ（ハードウェア・デザイン・ランゲージ）で記述し、コンパイラに入力して中間形式を得ます。この中間形式は、リンカーによって必要なサブプログラムを結合して、実行形式となります。

サブプログラムは、中間形式の中に、メインプログラムと共に与えられますが、良くもちいられるサブプログラムは、データ・ベースとしてあらかじめ定義しておくことができます。

この実行形式を、シミュレータ・実行コマンドの命令にしたがって、シミュレータが実行します。結果は、タイミングチャートの形式でラインプリンタに出力されます。

2. ハードウェア記述言語

2.1 基本事項

ここでは、実際にＨＤＬシステム・記述プログラムを書く場合の基本的な規約及び注意事項を説明します。なお、ＨＤＬの文法については、その詳細は、基本システム設計書を参照して下さい。

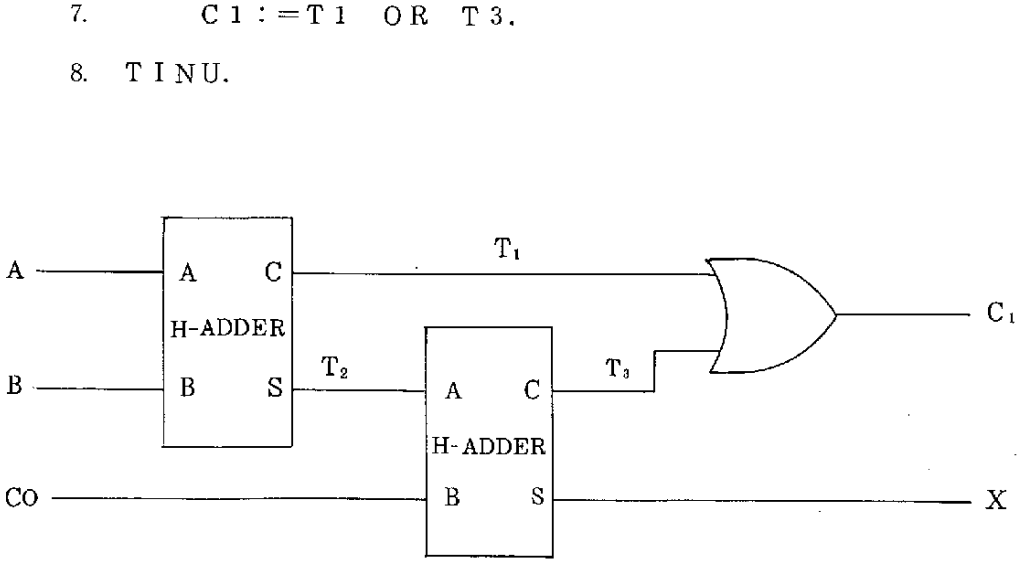
2.1.1 プログラムの作成

まず最初に、簡単なＨＤＬシステム・記述プログラムを作成します。

ここで、半加算機を２つ用いて全加算機を作ります。

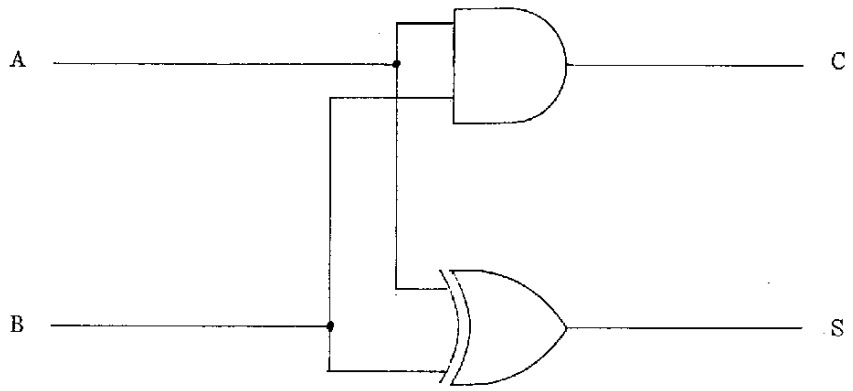
〔全加算機〕

```
1.  UNIT  F−ADDER  (MAIN).
2.        SWITCH  A, B, CO.
3.        LIGHT  C1, X.
4.        TERMINAL  T1,  T2.  T3.
5.        CONNECT  H−ADDER  (A, B; T1, T2).
6.        CONNECT  H−ADDER  (T2, CO; T3, X).
7.        C1 := T1  OR  T3.
8.  TINU.
```



〔半加算機〕

```
1.  UNIT  H−ADDER  (A, B ; C, S).
2.        TERMINAL  A, B, C, S.
3.        C  :=A  AND  B.
4.        S  :=A  XOR  B.
5.  TINU.
```



上記のプログラムの各部分について詳しい説明は後で述べます。

ここでは概要を説明します。全加算機のプログラムにおいて、第１行は、プログラムの開始を示す。ＵＮＩＴ文と呼ばれるものでＦ－ＡＤＤＥＲは、このプログラムの名前です。このプログラムが主プログラムであることを示します。

ここで注意することは、一般のプログラムと違い第２行から第７行までの順番は何の意味もありません。なぜならば、ＨＤＬシステム・記述プログラムは対応するハードウェアの存在と構成を記述しているからです。

また、第５行と第６行は、Ｈ－ＡＤＤＥＲというプログラム名を持つ副プログラムを呼び出すＣＯＮＮＥＣＴ文です。

全加算機の回路を見るとわかるようにＨ－ＡＤＤＥＲが２つ存在しています。第８行のＴＩＮＵ文は、プログラムの終りを示す文で、ＵＮＩＴ文と対応しています。

半加算機のプログラムは、全加算機で呼び出される副プログラムであり、第1行のＵＮＩＴで（Ａ，Ｂ：Ｃ，Ｓ）は、主プログラムと副プログラムとの引数で、ＡとＢは、副プログラムへの入力、ＣとＳは出力を示す。

これらはＣＯＮＮＥＣＴ文と対応しています。プログラム間の結合は、この引数だけで行なわれます。

2.1.2　使用可能な文字

ＨＤＬシステム・記述プログラムを書くために使用できる文字を以下に示します。

数　字	0　から　9　までの数字
英　字	A　から　Z　までのアルファベット
カナ文字	ア　から　ン　までのカタカナ
特殊文字	$　　／ @　　（ #　　） ：　　， ；　　． ％ 、 ？ ＝ ＋ －　　＿　下線 ＊　　空白

2.1.3　区切り記号

単語と単語を区切ったり、結合したりするのに用いられる文字を区切記号と呼び、区切記号を下に示します。

:=	=<	>=	=
<	>	;	:
,	.	(	)
+	−	★	／
#	ゐ	空白	

2.1.4　予　約　語

あらかじめ、ＨＤＬシステム記述・コンパイラに用意されていて、特定の意味をもつものを予約語といい、予約語と同じ識別子を別の目的で使用できません。予約語には、以下のものがあります。

UNIT	TINU	MAIN	REGISTER
SUBREGISTER	CASREGISTER	CONSTANT	TERMINAL
SUBTERMINAL	CASTERMINAL	DELAY	BY
NS	MUS	DECODER	ENCODER
TAG	SWITCH	LIGHT	CLOCK
DEMUX	MUX	IF	THEN
ELSE	FI	CASE	OF
ESAC	BIBUS	OUTBUS	AT
DO	TA	ON	NO
CONNECT	TRIST	OR	
NOT	INCR	DECR	REFL
XFER	PRR	PRI	NEGPRR

NEGPRL	UPRR	UPRL	NEG_UPRR
NEG_UPRL	COIN	SXOR	NOR
NAND	MOD	SHR	SHL
CIR	CIL	DSHR	DSHL
CSHR	CSHL	ESHL	CONV
@	@HIGH	@LOW	@FLOAT
@UP	@DOWN		

2.1.5 識 別 子 （ identifier ）

単独の文字あるいは、いくつかを組み合わせてある意味を持った文字の並びを識別子といい、プログラム中で使用する素子名、プログラム名（ユニット名）をこの識別子で表わします。識別子をプログラム名として用いるときは、長さに制限はないが、１０文字をこえるときは、１１文字以後は無視されます。この他に次の条件を満していなければなりません。

1) 識別子の１文字目は数字であってはならない。

2) 区切り記号を含んではならない。

3) 予約語と同じものであってはならない。

例 ＡＢＣＤＥＦＧＨＩＪＫＬ

ＡＢＣＤＥＦＧＨＩＪＪＸ

この２つは同じ識別子として解釈されます。

2.1.6 値 （literal）

値として、次の４種類を使うことができます。

1) １０進数 ０から９までの数字を組み合わせて作ったもの

2) １６進数 ０から９までの数字とＡ，Ｂ，Ｃ，Ｄ，Ｅ，Ｆ，の英字を組み合わせて作ったもので最後にＨをつける。

ただし、最初の１文字は数字でなければならない。

3) ８進数 ０から７までの数字を組み合わせて作ったもので最後にＱま

たはOをつける。

4)　2進数　0と1の数字を組み合わせて作ったもので最後にBをつける。

例　15

OFH

17Q　または　17O

1111B

それぞれ、10進数、16進数、8進数、2進数で書かれた15dです。

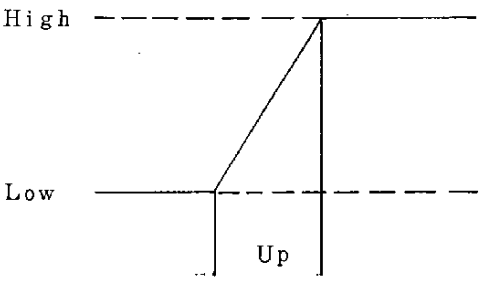
2.1.7　内部表現

HDLでは、回路のシミュレーションを行なうにあたり、回路の状態を次の6つに分けて表わしています。

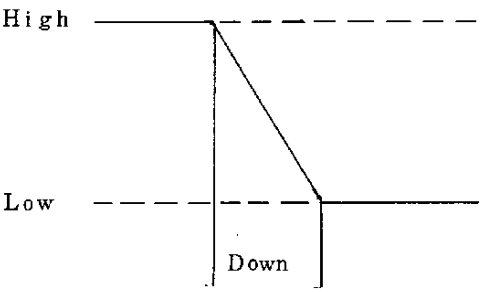
High　：　信号がハイ・レベルである

Low　：　信号がロー・レベルである

Up　：　信号がローからハイに上がる過渡状態



Down　：　信号がハイからローに下がる過渡状態



Floating　：　ハイ・インピーダンス出力であることを示す

Warning　：　回路の状態が不定であることを示す

2.1.8　素子の接続

ＨＤＬ記述では、素子の接続を▼：＝▼で表わします。

これは、一般の言語の代入に相当するものであるが、左右両辺のビット数が同じ長さでなければなりません。そこで、左右のビット数が異なる場合、ダミーを用いて、ビット数の調節を行なわなければなりません。

ダミーには、以下の６種類があります。

@ＨＩＧＨ

@ＬＯＷ

@ＵＰ

@ＤＯＷＮ

@ＦＬＯＡＴ

@　　……コンパイラのconsol入力のcommand文によって属性が与えられます。

例　４ビットのキャリー付の加算について

1)　ＣＹ：Ａ（３：０）：＝Ｂ（３：０）＋Ｃ（３：０）

2)　ＣＹ：Ａ（３：０）：＝@：Ｂ（３：０）＋@：Ｃ（３：０）

１）は左辺が４ビットとキャリーの１ビットで長さが５ビットですが、右辺は、４ビットだけで、長さが同じでないので誤りです。

２）は、@を付けて調節したものです。

2.1.9　コーディングの方法

ＨＤＬのシステム・パートのプログラムは、ＭＳ－ＤＯＳの管理化にある既成のエディタの下で入力され、コンパイラーで処理します。

エディタの一行は、最大２５４文字まで入力できます。コーディングは、比較的自由に行なえますが、区切り記号は、行をまたがらないようにして下さい。

2.1.10　コンパイラ・オプション

(1)　コンパイル時に、オプションとして以下のパラメータを指定できます。

このオプションによって、コンパイル方法の指定及びコンパイラからの必要な情報を得ることができます。

i) オプションは、任意個指定できます。

ii) 一行の入力文字数は２５４文字です。

iii) オプションの前後には、任意個の空白があってかまいません。

iv) 各オプションの指定順位は自由です。

v) 対立するオプションを同時には指定できません。

(2) コンパイラ時オプションの一覧表を下に示します。

No.	コンパイル時オプション			省略時の解釈
1	SOURCE	又は	NOSOURCE	SOURCE
2	SYMBOL	又は	NOSYMBOL	NOSYMBOL
3	LITERAL	又は	NOLITERAL	NOLITERAL
4	UNIFORM	又は	NOUNIFORM	NOUNIFORM
5	PARSING	又は	NOPARSING	NOPARSING
6	ACTION	又は	NOACTION	NOACTION
7	STACK	又は	NOSTACK	NOSTACK
8	AID	又は	NOAID	NOAID
9	ACLOCK	又は	NOACLOCK	NOACLOCK
10	MATRIX	又は	NOMATRIX	NOMATRIX
11	MID	又は	NOMID	NOMID
12	MCLOCK	又は	NOMCLOCK	NOMCLOCK
13	OBJECT	又は	NOOBJECT	NOOBJECT
14	MEMORY	又は	NOMEMORY	NOMEMORY
15	DIRECTORY	又は	NODIRECTORY	NODIRECTORY
16	TIME	又は	NOTIME	NOTIME
17	HIGH	又は	LOW	HIGH
	又はUP	又は	DOWN	HIGH
18	DTEST	又は	NODTEST	NODTEST
19	DIMENSION	又は	NODIMENSION	NODIMENSION
20	SIZE＝"LITERAL"			SIZE＝100

注意　SIZE＝"LITERAL"と続けて書きます。

(3) オプションの説明

i) SOURCE　　ソース・プログラムを印刷する

NOSOURCE　　ソース・プログラムを印刷しない

ii) №2より№16までは、主にコンパイラのディバッグのため出力指定です。

iii) №17　　ソース・プログラム中で使用されたダミー▼@▼に属性を与えます

iv) DTEST (DIMENSION TEST)　　配列の添字の値の範囲が正しいかを調べる命令を挿入します

NODTEST　　配列の添字のための処理はしません

v) DIMENSION　　配列の範囲の書き方によって、配列の各ビットの重みの付け方を決定します

例　TERMINAL　　A(3:0),B(0:3).

A(3:0):=B(3:0).

は、下記の解釈となります。

A(3:0):=Reflect B(0:3)

NODIMENSION　上記の例で

A(3:0):=B(3:0).

A(3:0):=B(0:3).

この二つは、同じものであると解釈します。

vi) SIZE　　コンパイラが実行時に使用する作業領域の大きさを指定します。この領域は小さいほど実行時間は短かくなりますが、小さすぎるとコンパイラの処理ができなくなります。

3. リンクとエディト（ＬＩＥＤ）

結合編集プログラム（ＬＩＥＤ）を実行させるに、エディタよりコマンドを入力します。

コマンドには、動作コマンド、出力コマンド及び作業領域指定コマンドの３種類のコマンドあります。この章では、これらのコマンドの指定のしかたを説明します。

3.1 動作コマンド

動作コマンドには、プログラムの結合編集およびデータベースの管理を行うために下記のコマンドがあります。

NOR　（NORMAL）
SUB　（SUB UNIT）
ADD
DLT　（DELETE）
NEW　（NEW DIRECTORY）

使用法は、実行時に、１つだけコマンドを与えます。ただし、コマンドを省略した場合には、ＮＯＲを指定したものと処理されます。

ⅰ）ＮＯＲ

翻訳された中間形式を実行形式へ結合編集するためのコマンドです。

入力方法は、ＮＯＲと入力するだけです。

ⅱ）ＳＵＢ

翻訳された中間形式を実行形式へ結合編集する動作は、ⅰ）のNORと同じですが、指定したサブ・ユニットごとに実行できるようにするために編集します。

入力方法は、ＳＵＢ＝＜実行したいサブ・ユニット名＞

iii) ＡＤＤ

翻訳された中間形式を、データ・ベース（ユニット・パッケージ）へ登録します。

入力方法は、

ＡＤＤ＝＜データ・ベースへ登録するサブ・ユニット名＞

iv) ＤＬＴ

既にデータ・ベースに登録してあるサブ・ユニットを、データ・ベースから消去します。

入力方法は、

ＤＬＴ＝＜データ・ベースから消去したいサブ・ユニット名＞

v) ＮＥＷ

新しくデータ・ベースを作るときに必ず'ＮＥＷ'を実行しなければなりません。

3.2 出力コマンド

結合編集プログラム（ＬＩＥＤ）の出力（実行形式）を出力装置へ出力させるためのコマンドです。

出力は、ＩＤファイル、ＭＡＴＲＩＸファイル、ＭＥＭＯＲＹファイル、ＣＬＯＣＫファイルの四つのファイルがあり、それぞれに対して出力させるか否かをコマンドで与えます。

出力する	出力しない／省略時解釈
ＩＤ	ＮＯＩＤ
ＭＡＴＲＩＸ	ＮＯＭＡＴＲＩＸ
ＭＥＭＯＲＹ	ＮＯＭＥＭＯＲＹ
ＣＬＯＣＫ	ＮＯＣＬＯＣＫ

3.3　作業領域指定コマンド

結合編集プログラム（ＬＩＥＤ）実行時に使用する作業領域の大きさを指定するものです。領域は小さいほど実行時間は短かくなりますが、小さすぎると結合編集ができなくなります。

このコマンドとして、下記の二つがあります。

ＩＤＴ＿ＳＩＺＥ＝＜リテラル＞

ＭＴＸ＿ＳＩＺＥ＝＜リテラル＞

＜リテラル＞の所に、領域の大きさを１０進数で書きます。省略の場合には、それぞれ次のように指定したものと解釈されます。

ＩＤＴ＿ＳＩＺＥ＝　５００

ＭＴＸ＿ＳＩＺＥ＝１０００

3.4　コマンドのまとめ

動作コマンド

ＮＯＲ
ＳＵＢ＝＜ユニット名＞
ＡＤＤ＝＜ユニット名＞
ＤＬＴ＝＜ユニット名＞
ＮＥＷ

出力コマンド

ＩＤ
ＮＯＩＤ

ＭＡＴＲＩＸ
ＮＯＭＡＴＲＩＸ

```
MEMORY
NOMEMORY
```

```
CLOCK
NOCLOCK
```

作業領域指定コマンド

```
IDT_SIZE=<リテラル>
```

```
MTX_SIZE=<リテラル>
```

省略時は、下線のコマンドが入ります。すべてを省略するときには、コマンドは必要ありません。

4. シミュレータ

4.1 シミュレーションの方法

実際のハードウェアの論理値は1（HIGH）と0（LOW）の2値であるが、シミュレータではつぎに示す値をとります。

記号	意　味	
L	LOW	論理0を示す
U	UP	論理0から論理1への過渡状態を示す
H	HIGH	論理1を示す
D	DOWN	論理1から論理0への過渡状態を示す
X	TRISTRTE	トライステート出力を意味し、ハイ・インピーダンス状態を示す
W	WARNING	UPとDOWNのアンド演算などの結果として値がはっきり定まらない状態を示す
T	TRIGGER	トリガーパルスを示す

上記の状態に対して、NOT、OR、ANDの演算を表1に示します。

表1　演算表

NOT	
L	H
U	D
H	L
D	U
X	H
W	W
T	T

OR	L	U	H	D	X	W	T
L	L						
U	U	U					
H	H	H	H				
D	D	W	H	D			
X	H	H	H	H	H		
W	L	U	H	D	H	W	
T	L	U	H	D	H	T	T

AND

	L	U	H	D	X	W	T
L	L						
U	L	U					
H	L	U	H				
D	L	W	D	D			
X	L	U	H	D	H		
W	L	U	H	D	H	W	
T	L	T	T	T	T	T	T

シミュレーションの結果は、以下のキャラクター表示を使用します。

L は 0
H は 1
D は I
U は I
X は TRG
W は W
T は T

4.2 仮想マシン

仮想マシンによってディジタルシステムのシミュレーションを行う。図1から明らかなように、メインコントローラのもとに、イニシャライズ・コントローラ、ディレイ・コントローラ、クロック・コントローラ、I／Oコントローラ、エグゼキュータなどのサブ・プロセッサから構成される。データエリアはシミュレーションされるディジタル・システムの論理素子の値を示し、エディタがこのデータエリアを作成します。

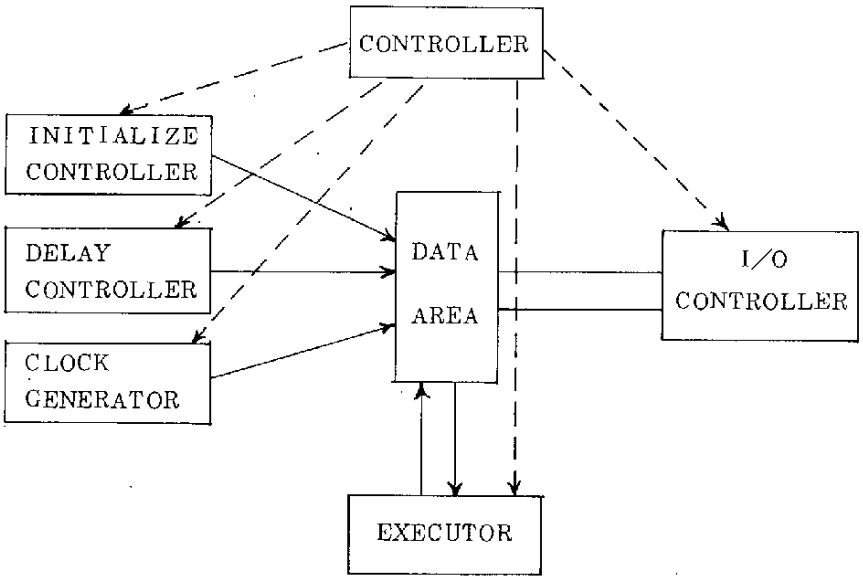


図 1.　仮想マシンの概略

以下、サブ・プロセッサについて簡単に説明します。

○　コントローラ

サブ・プロセッサ全体のコントロールを行います。

○　イニシャライズ・コントローラ

コントロール・コマンドで示されたデータ・エリアの一部を示された値に初期化する。

○　ディレイ・コントローラ

ディレイカウンタ（アップ・カウンタまたはダウン・カウンタ）の値を調べ、0であれば、値（論理1(H)または論理0(L)）をヴァリューにセットします。

○　クロック・コントローラ

エディターの作製したクロック・テーブルをもとに、クロックを更新します。

○　エグゼキュータ

実行形式のプログラムにしたがって、データエリアの値を変化させる。実際には、アップ・カウンタまたは、ダウン・カウンタをセットするだけであ

る。このプログラムはシステム部で記述されたディジタル・システムの各素子間の結合関係を示しています。

○　I／Oコントローラ

コントローラ・テーブルをもとに、スイッチ入力かプリント出力かを調べ、入出力のコントロールを行います。

4.3　シミュレーションのフロー

シミュレーションのフローを図2に示し、フローに従って簡単に説明します。

1. エディターの作製した実行形式のプログラムをロードします。実行形式のプログラムは、アイデンティファイヤー・テーブル、プログラム・エリア、クロック・テーブル、データ・エリアから構成されています。
2. コントロール部のコマンドを翻訳し、アイデンティファイヤー・テーブルをもとに、コントロール・テーブルを作成します。
3. コントロール・テーブルによって、シミュレーションするディジタル・システムの初期化を行います。
4. データ・エリア（ディジタル・システムの各論理素子を表現している）に論理的に矛盾がないように初期化を行います。
5. スイッチ入力か否かを調べ、そうであれば対応するスイッチにデータを入力します。
6. 実行形式のプログラムにしたがって、エグゼキュータがシミュレーションを行う。しかし、データ・エリアの値は直接変えず、ディレイ・カウンタ（アップ・カウンタまたはダウン・カウンタ）の値をセットするだけです。
7. ディレイ・コントローラによって、データ・エリアのディレイ・カウンタの値を調べ、0であればヴァリューに新しく値をセットします。
8. クロック・テーブルによってクロックの更新を行います。
9. コントロール・テーブルを調べ、出力する条件が成立しておれば、データ・エリアの値などを出力します。

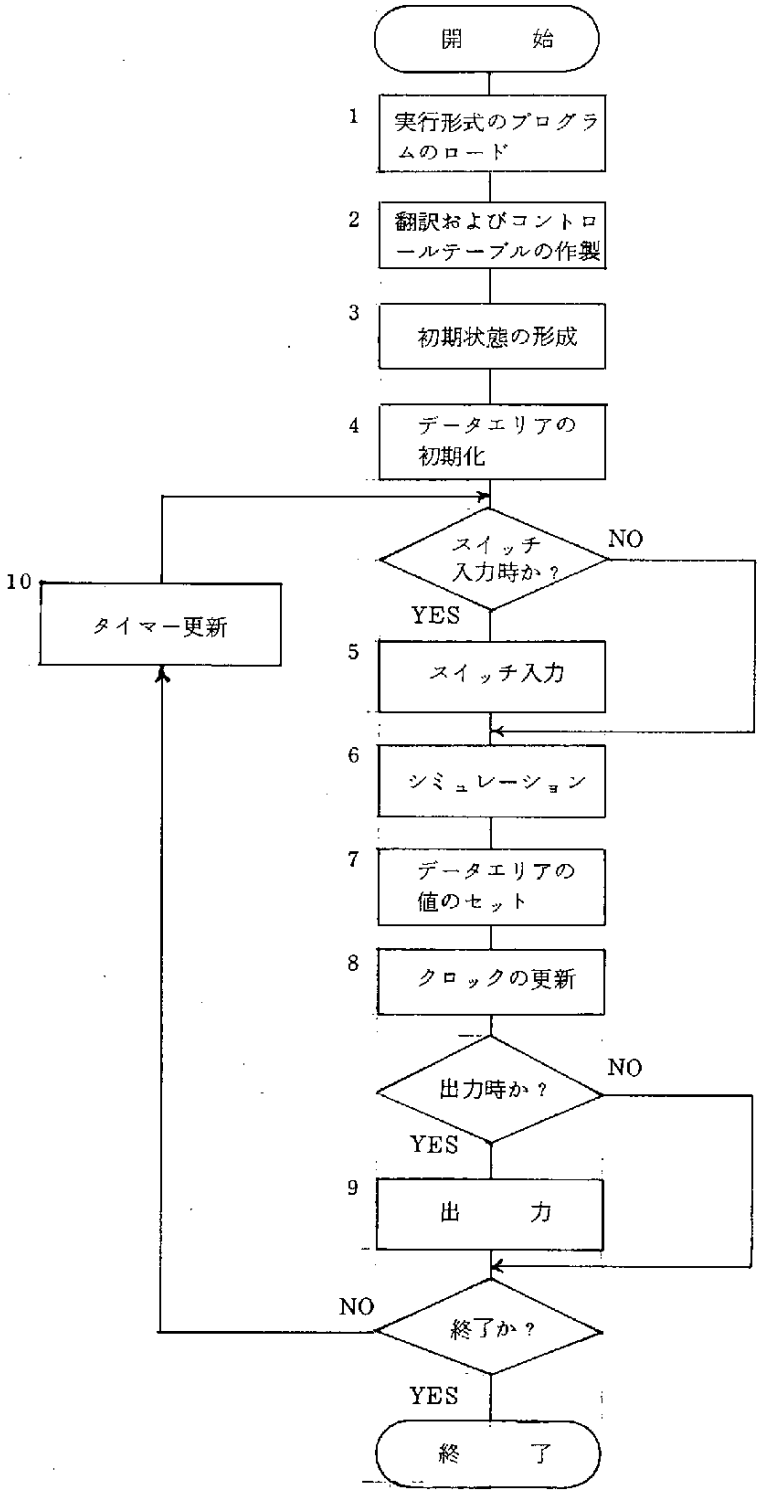


図 2.　シミュレーション・フロー

10. コントロール・テーブルによって、シミュレーションを続けるか否かを調べる。続行であればタイマーを更新し、シミュレーションをくり返します。

4.4 シミュレーション・コマンドの使用法

シミュレータ・コントロール部は、シミュレーションの条件を指定する。同一のソースプログラムで種々の条件でシミュレーションができるように、システム部で記述したディレイの値やクロックの周期などを変更できます。また、シミュレーションを行う前の初期値のセットが可能です。システム部で記述したシンボル名を用いて、スイッチ属性やライト属性をもつものを含めて、すべての論理素子を指定することができます。

1) ＩＮＩＴＩＡＬＩＺＥ文

機能　シミュレーション実行時の初期状態をセットします。初期化したい論理素子と初期化データを指定して下さい。ＩＮＩＴＩＡＬＩＺＥ文で指定されないものは、システムが０をセットします。

例　　ＩＮＩＴＩＡＬＩＺＥ　　Ａ（３：０）＝１００１；

レジスタまたはターミナルＡを１００１Ｂに初期化します。

2) ＤＥＬＡＹＳＥＴ文

機能　ディレイ値をセットします。又システム部で既に与えられているディレイの値を変更するために使用します。変更したい論理素子とディレイ値を指定します。単位はｎ秒である。

例　　ＤＥＬＡＹＳＥＴ　　Ｑ＝（１００、５００）；

Ｑの立上りのディレイを１００ｎ秒、立下りのディレイ値を５００ｎ秒に変更します。

3) ＣＬＯＫＳＥＴ文

機能　システム部で記述しているクロックの周期やパルス幅などを変更する時に使用します。クロック名、パルス幅などを指定します。単位はｎ秒である。

例　　ＣＬＯＣＫＳＥＴ　　ＣＬ＝（１００、１５０、１５０）；

クロックＣＬを下図に示すように変更します。

かっこの中の第１項は、初期ＬＯＷのパルス幅

4)　ＳＷＩＴＣＨＩＮ文

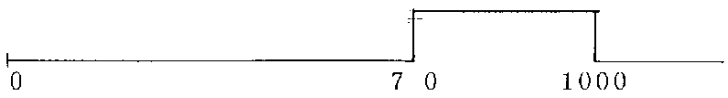
機能　システム部でスイッチとして宣言されたものに外部からデータを入力するのに使用します。スイッチ名、入力データ、スイッチ入力する時間（システムの持つ内部タイマーによる）を指定します。

例　　ＳＷＩＴＣＨＩＮ　　ＡＴ　　７５０　　ＳＷ＝１；

ＳＷＩＴＣＨＩＮ　　ＡＴ　　１０００　　ＳＷ＝０；

スイッチＳＷを７５０ｎ秒にオン、１０００ｎ秒にオフ；

この波形を下図に示す。



5)　ＰＲＩＮＴＯＵＴ文

機能　シミュレーション結果を出力したい論理素子とその時間あるいは条件などを指定します。

例　　ＰＲＩＮＴＯＵＴ　　ＢＹ　ＣＬＯＫ（ＣＬ）　Ｘ；

クロックＣＬの立上り時のＸの内容が出力されます。

例　　ＰＲＩＮＴＯＵＴ　　ＢＹ　ＣＹＣＬＥ（１００）　Ｙ；

１００ｎ秒ごとにＹの内容が出力されます。

例　　ＰＲＩＮＴＯＵＴ　ＡＴ　１０００　Ａ；

タイマーが１０００ｎ秒の時Ａの内容が出力されます。

例　　ＰＲＩＮＴＯＵＴ　ＡＴ　Ｚ＝１　Ｂ；

Ｚ＝１のとき、Ｂの内容が出力されます。

6) ＲＵＮ文

機能　シミュレーションの開始を示すとともに、シミュレーション時間を指定します。

例　　ＲＵＮ（１０００）；

タイマーが１０００ｎ秒までシミュレーションを行います。

例　　ＲＵＮ（１０００、２０００）；

１０００ｎ秒までタイマーを動かし、１０００ｎ秒から２０００ｎ秒までシミュレーションを行います。

例　　ＲＵＮ（Ｆ＝１）；

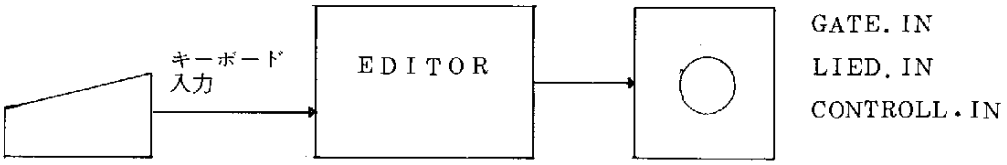
Ｆ＝１になるまでシミュレーションを行います。

5. 操 作 説 明

5.1 操作の概要

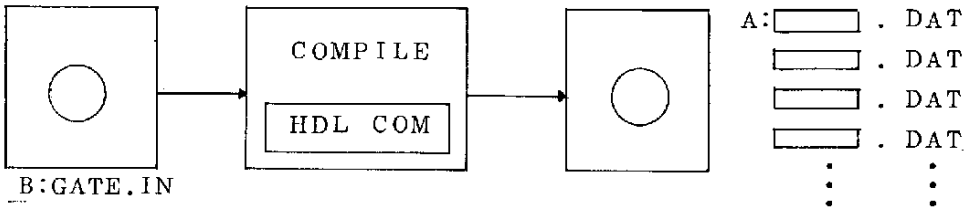
① ＭＳ－ＤＯＳの起動

② エディト



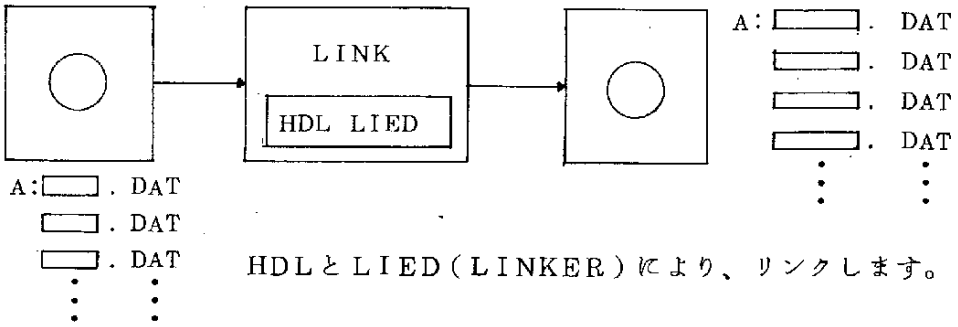
エディタを使い、GATE. IN（コンパイル時のシステム・ソース）、LIED. IN（リンク時のリード・ソース）、CONTROLL. IN（シミュレータ時のコントロール・ソース）というファイル名のソースプログラムを作成します。

③ コンパイル



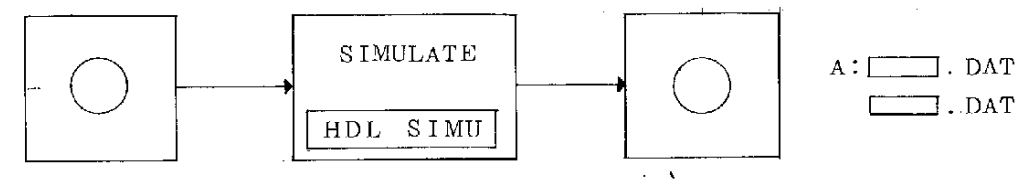
HDL（CP/M EMULATOR）とCOM(COMPILER)により、GATE. INをコンパイルします。

④ リンク



HDLとLIED（LINKER）により、リンクします。

⑤ シミュレート



A: . DAT
. DAT
. DAT
. DAT

ＨＤＬとＳＩＭＵ（ＳＩＭＵＬＡＴＥ）により、シミュレートします。

尚、3.4.5に関し、結果はＣＲＴにのみ出力されます。但し、コンパイル、リンク、シミュレート、それぞれの実行に際し、

ＨＤＬ　ＣＯＭ　又は

ＨＤＬ　ＬＩＥＤ　又は

ＨＤＬ　ＳＩＭＵ

とキー入力した後、ＲＴＮ　ＫＥＹを押す前に、[CNTL]＋Ｐを入力しますと、ＣＲＴ出力と同時にプリンタにも出力可能となります。

5.2　ＭＳ－ＤＯＳの起動

① ＭＳ－ＤＯＳの起動

1) ＭＳ－ＤＯＳ　ＶＥＲＳＩＯＮ　2.0以上を使用して下さい

2) メモリは最低３８４ＫＢを用意して下さい

3) ＭＳ－ＤＯＳ起動後、ディスプレイの下部のコマンドを消す為に、[CNTL]とファンクション・キー[F7]を２度押します。

② ＨＤＬディスクの検査

ＭＳ－ＤＯＳのＤＩＲＥＣＴＯＲＹコマンドを使い、ディスクの内容を検査します。

```
A>DIR

COMMAND  COM    ****** **-**-**  **:**
EDL      COM    ****** **-**-**  **:**
COM      COM    ****** **-**-**  **:**
PARAMETR OVR    ****** **-**-**  **:**
PASS1    OVR    ****** **-**-**  **:**
PASS2    OVR    ****** **-**-**  **:**
PASS3    OVR    ****** **-**-**  **:**
PASS4    OVR    ****** **-**-**  **:**
ERPRINT  OVR    ****** **-**-**  **:**
LIED     COM    ****** **-**-**  **:**
SIDT     COM    ****** **-**-**  **:**
PAGING   OVR    ****** **-**-**  **:**
MACHINE  OVR    ****** **-**-**  **:**
VGA_GET  OVR    ****** **-**-**  **:**
VGA_PUT  OVR    ****** **-**-**  **:**
CLK_SET  OVR    ****** **-**-**  **:**
IND      OVR    ****** **-**-**  **:**
DLY_SET  OVR    ****** **-**-**  **:**
DLY_SYNC OVR    ****** **-**-**  **:**
MESSAGE  DAT    ****** **-**-**  **:**
R_TABLE  DAT    ****** **-**-**  **:**
B_TABLE  DAT    ****** **-**-**  **:**
N_TABLE  DAT    ****** **-**-**  **:**
C_TABLE  DAT    ****** **-**-**  **:**
D_TABLE  DAT    ****** **-**-**  **:**
TRM1     DAT    ****** **-**-**  **:**
UNIFORM1 DAT    ****** **-**-**  **:**
REDUCT   DAT    ****** **-**-**  **:**
BIT_OPT  DAT    ****** **-**-**  **:**
TRM2     DAT    ****** **-**-**  **:**
```

5.3 エディト

① ソース・ファイル作成

1) ＭＳ－ＤＯＳ上で使用可能なＥＤＩＴＯＲ、たとえば"ＷＳ"（WORDSTAR）、"ＥＤＬＩＮ"などを使用して下さい。

2) ソースプログラムの一行の最大文字数として、２５４文字までコンパイラはサポートしていますが、それ以上は無視します。

3) ＴＡＢ、ＳＰＡＣＥ、ＣＡＲＲＩＡＧＥ　ＲＥＴＵＲＮなどは自由に挿

入可能です。

4) ソースのファイル名は、

ＧＡＴＥ．ＩＮ（コンパイラにかけるシステム・ソース）

ＬＩＥＤ．ＩＮ（リンカにかけるリード・ソース）

ＣＯＮＴＲＯＬＬ．ＩＮ（シミュレータにかけるコントロール・ソース）

など、それぞれの実行に於いて固定ファイル名の為、異なるファイル名を作成するとＥＲＲＯＲが発生します。

② サンプル・ソース・ファイル

例 1) ＧＡＴＥ．ＩＮ

```
UNIT SAMPLE(MAIN).

        TERMINAL I1,I2,I3,G1,G2,G3,Y1,Y2.
        CLOCK X=5 BY 5 NS.
        I1 := NOT X.
        I2 := NOT Y2.
        I3 := NOT Y1.
        G1 := I1 AND I2 AND Y1.
        G2 := I1 AND I3 AND Y2.
        G3 := X AND Y1.
        Y1 := G1 OR G2 OR G3.
        Y2 := X.

TINU SAMPLE.
```

例　2) CONTROLL. IN

```
INITIALIZE Y2=0;
INITIALIZE Y1=0;
INITIALIZE G3=0;
INITIALIZE G2=0;
INITIALIZE G1=0;
INITIALIZE I3=1;
INITIALIZE I2=1;
INITIALIZE I1=1;
DELAYSET G2=(1,3);
DELAYSET Y2=(1,3);
PRINTOUT BY CYCLE(1) X;
PRINTOUT BY CYCLE(1) I1;
PRINTOUT BY CYCLE(1) I2;
PRINTOUT BY CYCLE(1) I3;
PRINTOUT BY CYCLE(1) G1;
PRINTOUT BY CYCLE(1) G2;
PRINTOUT BY CYCLE(1) G3;
PRINTOUT BY CYCLE(1) Y1;
PRINTOUT BY CYCLE(1) Y2;
RUN(50);
```

5.4　コンパイル

① コンパイル

1) HDL. COM(CP/M EMULATOR)とCOM. CMD (COMPILER)をAドライブにセットします。

2) エディタで作成したGATE. IN(ソース・ファイル)をBドライブにセットします。

3) コンパイルの例を以下に示します。

A＞HDL　COM　とキー入力すると・・・・

```
############################################################

                        HDL  COMPILER

                        Vervion 1.0
                        01-March-1985
                    Serial # MIE0930-C001
                    All rights reserved
                  Copyright (c) 1984.1985

                      SOLITON SYSTEMS K.K.

############################################################


HARDWARE DESIGN LANGUAGE COMPILE

OPTIOIONS SPECIFIED ARE AS FOLLOWS --


LIST OF OPTIONS USED DURING PROCESSING IS --
                                                GATE
                                                NOSYMBOL
                                                NOLITERAL
                                                NOUNIFORM
                                                NOPARSING LIST
                                                NOACTION
                                                NOSTACK
                                                NOACTION ID
                                                NOCLOCK
                                                NOMATRIX
                                                NOMODIFY ID
                                                NOOBJECT
                                                NOMEMORY
                                                NOMODIFY CLOCK
                                                NOMODIFY
                                                NODIMENSION TEST
                                                NODIRECTRY
                                                NODIMENSION TYPE
                                                NOTIME
                                                DUMMY=HIGH
                                                ID TABLE SIZE=   100
```

```
PASS1
     LEXICAL ANALYSIS OUTPUT.
 NO.     GATE

    1   UNIT SAMPLE(MAIN).
    2   TERMINAL I1.I2.I3.G1.G2.G3.Y1.Y2.
    3   CLOCK X=5 BY 5 NS.
    4   I1 := NOT X.
    5   I2 := NOT Y2.
    6   I3 := NOT Y1.
    7   G1 := I1 AND I2 AND Y1.
    8   G2 := I1 AND I3 AND Y2.
    9   G3 := X AND Y1.
   10   Y1 := G1 OR G2 OR G3.
   11   Y2 := X.
   12   TINU SAMPLE.
PASS2
PASS3
PASS4
End of Execution
```

② コンパイル後の中間ファイル

1) コンパイルにより、以上の［　　］．ＤＡＴファイル（中間ファイル）がＡドライブに自動生成されます。

2) この中間ファイルはＭＳ－ＤＯＳのＤＵＭＰコマンドにより、内容をチェックする事ができます。

3) これらの［　　］．ＤＡＴファイルはリンカ、シミュレータが使用します。

```
A>DIR

ERR_FILE DAT      *****   **-**-**   **:**
UNIFORM2 DAT      *****   **-**-**   **:**
ID       DAT      *****   **-**-**   **:**
LIT      DAT      *****   **-**-**   **:**
ACTION   DAT      *****   **-**-**   **:**
CLK      DAT      *****   **-**-**   **:**
MATRIX   DAT      *****   **-**-**   **:**
MEMORY   DAT      *****   **-**-**   **:**
OBJECT   DAT      *****   **-**-**   **:**
IDTFILE  DAT      *****   **-**-**   **:**
MTXFILE  DAT      *****   **-**-**   **:**
CLKFILE  DAT      *****   **-**-**   **:**
MEMFILE  DAT      *****   **-**-**   **:**
EXCLK    DAT      *****   **-**-**   **:**
EXMEM    DAT      *****   **-**-**   **:**
EXMTX    DAT      *****   **-**-**   **:**
EXIDT    DAT      *****   **-**-**   **:**
```

5.5 リ ン ク

① リンク

1) ＨＤＬ．ＣＯＭ（ＣＰ／Ｍ ＥＭＵＬＡＴＯＲ）とＬＩＥＤ．ＣＭＤ（ＬＩＮＫＥＲ）をＡドライブにセットします。

2) コンパイラにより自動生成された中間ファイルは、そのままＡドライブにセットします。

3) エディタで作成したＬＩＥＤ．ＩＮ（リンクの為のオプション・コマンド＝リード・ソース）をＢドライブにセットします。但し、このコマンド・ソースは省略可能です。その場合デフォルト・コマンドが使用されます。（リンク例 参照）

4) リンクの例を以下に示します。

Ａ＞ＨＤＬ ＬＩＥＤとキー入力すると・・・・

```
############################################################

                    HDL LINKER

                    Vervion 1.0
                   01-March-1985
                Serial # MIE0120-L001
                All rights reserved
              Copyright (c) 1984.1985

                  SOLITON SYSTEMS K.K.

############################################################
```

```
COMMAND= 5
UNITNAME=
IDT_LIST= 0
MTX_LIST= 0
MEM_LIST= 0
CLK_LIST= 0
TIM_LIST= 0
IDT_SIZE= 500
MTX_SIZE=1000


      End of Execution
```

② リンク後の中間ファイル

1) リンクにより、以下の[　　]．ＤＡＴファイル（中間ファイル）がＡドライブに自動生成されます。

2) この中間ファイルはＭＳ－ＤＯＳのＤＵＭＰコマンドにより、内容をチェックする事ができます。

3) これらの[　　]．ＤＡＴはシミュレータが使用します。

```
A>DIR

ERR_FILE  DAT    *****   **-**-**   **:**
UNIFOR12  DAT    *****   **-**-**   **:**
ID        DAT    *****   **-**-**   **:**
LIT       DAT    *****   **-**-**   **:**
ACTION    DAT    *****   **-**-**   **:**
CLK       DAT    *****   **-**-**   **:**
MATRIX    DAT    *****   **-**-**   **:**
MEMORY    DAT    *****   **-**-**   **:**
OBJECT    DAT    *****   **-**-**   **:**
IDTFILE   DAT    *****   **-**-**   **:**
MTXFILE   DAT    *****   **-**-**   **:**
CLKFILE   DAT    *****   **-**-**   **:**
MEMFILE   DAT    *****   **-**-**   **:**
EXCLK     DAT    *****   **-**-**   **:**
EXMEM     DAT    *****   **-**-**   **:**
EXMTX     DAT    *****   **-**-**   **:**
EXIDT     DAT    *****   **-**-**   **:**
MTPAGE    DAT    *****   **-**-**   **:**
MEPAGE    DAT    *****   **-**-**   **:**
WORK_ID   DAT    *****   **-**-**   **:**
WORK_MT   DAT    *****   **-**-**   **:**
WORK_CL   DAT    *****   **-**-**   **:**
WORK_ME   DAT    *****   **-**-**   **:**
```

5.6 シミュレート

① シミュレータ

1) HDL. COM(CP/M EMULATOR)とSIMU. CMD(SIMULATOR)をAドライブにセットします。

2) コンパイラ、リンカにより自動生成された中間ファイルは、そのままAドライブにセットします。

3) エディタで作成したCONTROLL. IN(シミュレータのコントロール・ソース)をBドライブにセットします。

4) シミュレータの例を以下に示します。

A>HDL SIMUとキー入力すると・・・・

```
##########################################################

                    HDL SIMULATOR

                    Vervion 1.0
                    01-March-1985
                Serial # MIE0129-S001
                All rights reserved
            Copyright (c) 1984.1985

                  SOLITON SYSTEMS K.K.

##########################################################
```

SIMULATION COMMAND STATEMENTS LIST(HDL)

ST_NO	STATEMENTS	ERR MESSAGE
1	INITIALIZE Y2=0;	
2	INITIALIZE Y1=0;	
3	INITIALIZE G3=0;	
4	INITIALIZE G2=0;	
5	INITIALIZE G1=0;	
6	INITIALIZE I3=1;	
7	INITIALIZE I2=1;	
8	INITIALIZE I1=1;	
9	DELAYSET G2=(1,3);	
10	DELAYSET Y2=(1,3);	
11	PRINTOUT BY CYCLE(1) X;	
12	PRINTOUT BY CYCLE(1) I1;	
13	PRINTOUT BY CYCLE(1) I2;	
14	PRINTOUT BY CYCLE(1) I3;	
15	PRINTOUT BY CYCLE(1) G1;	
16	PRINTOUT BY CYCLE(1) G2;	
17	PRINTOUT BY CYCLE(1) G3;	
18	PRINTOUT BY CYCLE(1) Y1;	
19	PRINTOUT BY CYCLE(1) Y2;	
20	RUN(50);	

```
*****************************************
*                                       *
*          SIMULATION  RESULTS          *
*                                       *
*****************************************
```

	X	I 1	I 2	I 3	G 1	G 2	G 3	Y 1	Y 2
1	0	1	1	1	0	0	0	0	0
2	0	1	1	1	0	0	0	0	0
3	0	1	1	1	0	0	0	0	0
4	0	1	1	1	0	0	0	0	0
5	0	1	1	1	0	0	0	0	0
6	-	1	1	1	0	0	0	0	0
7	1	1	1	1	0	0	0	0	0
8	1	-	1	1	0	0	0	0	0
9	1	0	1	1	0	0	0	0	-
10	1	0	1	1	0	0	0	0	1
11	-	0	-	1	0	0	0	0	1
12	0	0	0	1	0	0	0	0	1
13	0	-	0	1	0	0	0	0	1
14	0	1	0	1	0	0	0	0	1
15	0	1	0	1	0	0	0	0	1
16	-	1	0	1	0	-	0	0	-
17	1	1	0	1	0	1	0	0	0
18	1	-	-	1	0	1	0	-	0
19	1	0	1	1	0	1	0	1	-
20	1	0	1	-	0	1	-	1	1
21	-	0	-	0	0	-	1	1	1
22	0	0	0	0	0	0	1	1	1
23	0	-	0	0	0	0	-	1	1
24	0	1	0	0	0	0	0	1	1
25	0	1	0	0	0	0	0	-	1
26	-	1	0	0	0	0	0	0	-
27	1	1	0	-	0	0	0	0	0
28	1	-	-	1	0	0	0	0	0
29	1	0	1	1	0	0	0	0	-
30	1	0	1	1	0	0	0	0	1
31	-	0	-	1	0	0	0	0	1
32	0	0	0	1	0	0	0	0	1
33	0	-	0	1	0	0	0	0	1
34	0	1	0	1	0	0	0	0	1
35	0	1	0	1	0	0	0	0	1
36	-	1	0	1	0	-	0	0	-
37	1	1	0	1	0	1	0	0	0
38	1	-	-	1	0	1	0	-	0
39	1	0	1	1	0	1	0	1	-
40	1	0	1	-	0	1	-	1	1
41	-	0	-	0	0	-	1	1	1
42	0	0	0	0	0	0	1	1	1
43	0	-	0	0	0	0	-	1	1
44	0	1	0	0	0	0	0	1	1
45	0	1	0	0	0	0	0	-	1
46	-	1	0	0	0	0	0	0	-
47	1	1	0	-	0	0	0	0	0
48	1	-	-	1	0	0	0	0	0
49	1	0	1	1	0	0	0	0	-
50	1	0	1	1	0	0	0	0	1

End of Execution

6. エラーメッセージ

エラーメッセージとして、

システムエラー

コンパイルエラー

リンクエラー

シミュレータエラー

があります。以下、この順に説明します。

6.1 システムエラー

Error	Description
INSUFFICIENT MEMORY	The loaded program cannot run in the memory size allocated. If possible, increase the size of the Transient Program Area.
※ ERROR(1) "Conversion"*	This error occurs whenever the run-time system cannot perform the required conversion between data types. This error can be signaled during arithmetic operations, assignments, and I/O processing with GET and PUT statements.
ERROR(2) "I/O Stack Overflow"*	The run-time I/O stack exceeds 16, simultaneous, nested I/O operations. You must simplify the program and try again.
ERROR(3)*	A transcendental function argument is out-of-range.

※ ERROR(4) "I/O Conflict x"*

A file is explicitly or implicitly opened with one set of attributes, and subsequently accessed with a statement requiring conflicting attributes. The value of x is one of the following:

- STREAM/RECORD
- SEQUEN/DIRECT
- INPUT/OUTPUT
- KEYED Access

The first conflict arises when ASCII files are processed using READ or WRITE, but the INTO or FROM option does not specify a varying character string.

Error	Description
ERROR(5) "Format Overflow"*	The nesting level of embedded formats exceeds 32. You must simplify the program and try again.
ERROR(6) "Invalid Format Item"*	The format processor encounters a format item that cannot be processed. The P format is not implemented in PL/I.
ERROR(7) "Free Space Exhausted"*	No more free space is available. If you intercept this error with an ON-unit, do not execute an ALLOCATE, OPEN, or recursion without first releasing storage.
ERROR(8) "OVERLAY, NO FILE d:filename"	The overlay manager cannot find the indicated file.
ERROR(9) "OVERLAY, DRIVE d:filename"	An invalid drive code is passed as a parameter to an overlay.
ERROR(10) "OVERLAY, SIZE d:filename"	The indicated overlay is too large and overwrites the PL/I stack and/or free space if loaded.

Error	Description
ERROR(11) "OVERLAY, NESTING d:filename"	Loading the indicated overlay exceeds the maximum nesting depth.
ERROR(12) "OVERLAY, READ d:filename"	There has been a disk read error while loading an overlay. This is probably caused by a premature EOF.

Error	Description
ERROR(13) "Invalid OS Version"	Any operation that generates an operating system call not supported under the current operating system causes this error.
ERROR(14) "Unsuccessful Write"	Any unsuccessful write operation on a file due to lack of directory space, lack of disk space, and so on, cause this error.
ERROR(15) "File Not Open"	Any attempt to lock or unlock a record in a file that is not open causes this error.
ERROR(16) "File Not Keyed"	Any attempt to lock or unlock a record in a file that does not have the KEYED attribute causes this error.
UNDEFINEDFILE	If this error occurs on input, the run-time system cannot find the named file on the disk, or an input device is opened for output. If the error occurs on output, the run-time system cannot create an output file, or an output device is opened for input.

<注> ※印のＥＲＲＯＲに関し、ソースファイルの SYNTAX ERROR の場合発生します。

6.2 コンパイラーのプログラム・エラー

コンパイラへの入力プログラムにエラーがある場合、コンパイラーは、エラー・メッセージをコンソールに出力します。エラー・メッセージを以上に示します。

```
LITERAL LENGTH OVER 33.
MULTIPLE LABEL DEFINE.
LABEL PART INCLUDE INVALID LABEL.
COMPARE PART INVALIDE INVALID KEY_WORD.
ACTION OVER 5.
INVALID ACTION LITERAL.
ACTION PART INCLUDE INVALID ACTION NUMBER.
STACK OVER 5.
INVALID STACK LITERAL.
STACK PART INCLUDE INVALID STACK ACTION.
INVALID NEXT LITERAL.
NOT DEFINE LABEL.
USE CALL & RETURN.
NEXT PART INCLUDE INVALID WORD.
END IS NOT '$'.
COMPARE OVER 5.
SYNTAX ERROR.
':' OF UNIT PARAMETER OVER 2.
DIMENSION OF REGISTER DECLARE PART OVER 5.
EXPRESSION OPERATER MORE.
EXPRESSION OPERAND MORE.
SYNBOLIC RANGE HAVE DIMENSION.
DIMENSION OF OPERAND OVER 5.
SYMBOLIC RANGE DIMENSION ERROR.
NOT DEFINE SYMBOLIC RANGE.
DIMENSION OF OPERAND OVER 5.
PRHIBITIVE ID INDEX.
DIMENSION OF OPERAND OVER 5.
DIMENSION OF OPERAND OVER 5.
DIMENSION OF DUMMY OVER 5.
EXPRESION OPERAND MORE.
CLOCK SIZE OVER 100.
CLOCK HIGH & FIRST LOW MISS MATCH.
CLOCK SIZE & HIGH MISS MATCH.
CLOCK FIRST LOW OVER SIZE.
CLOCK HIGH OVER SIZE.
CLOCK MUS OVERFLOW.
CONSTANT LITERAL OVER.
CONSTANT LITERAL & SIZE MISS MATCH.
```

```
CASCADE LENGTH MISS MATCH.
DIMENSION OF CASCADE OVER 5.
INVALID SYMBOLIC RANGE.
INVALID CASCADE OF SYMBOLIC RANGE.
PROHIBITIVE ID INDEX.
DIMENSION OF CASCADE OVER 5.
DIMENSION OF CASCADE OVER 5.
INVALID MUX FI.
INVALID MUX ELSE.
MUX OR DEMUX DO NOT HAVE DIMENSION.
DEMUX VARIABLE LENGTH MISS MATCH.
':' OF CONNECT PARAMETER OVER 2.
ON STATEMENT INCLUDE IF STATEMENT.
MISSING FI.
MISSING FI.
ON STATEMENT INCLUDE CASE STATEMENT.
MISSING ESAC.
MISSING ESAC.
MISSING ESAC.
ON STATEMENT INCLUDE AT STATEMENT.
MISSING TA.
ON STATEMENT INCLUDE ON STATEMENT.
MISSING ON.
MISSING END.
MISSING IF.
MISSING AT.
MISSING CASE.
MISSING MUX IF.
MISSING DEMUX IF.
MULTIPLE DECLARATION.
ILLIGAL LITERAL.
DIMENSION OVER 5.
EXPRESION OPERATION MORE.
STRUCTURE STATEMENT ERROR.
MISSING ')'.
MISSING EXPRESION.
ILLIGAL OPERATION.
':=' LENGTH MISS MATCH.
BINAEY OPERAND LENGTH MISS MATCH.
DELAY MUS OVERFLOW.
UNIT PARAMETER OUTPUT INCLUDE SWITCH.
UNIT PARAMETER INPUT INCLUDE LIGHT.
UNIT PARAMETER INCLUDE CLOCK.
UNIT PARAMETER INCLUDE CONSTANT.
CONNECT PARAMETER INCLUDE CONSTANT.
CONNECT PARAMETER INCLUDE CASCADE.
CONSTANT TYPE(2) ERROR.
CLOCK TYPE(2) ERROR.
```

```
ILLEGAL LITERAL.
LITERAL ERROR.
ILLEGAL CHARACTER IN LITERAL.
ILLEGAL LITERAL TYPE ('B','O','H').
LITERAL OVER FLOW.
STORE TYPE ('CS','CL','SU','TG') ERROR.
STORE TYPE ('SM','* ','SU','  ') ERROR.
LOAD TYPE ('LG') ERROR.
LOAD TYPE ('SM','* ','SU','  ') ERROR.
VARIABLE RANGE ERROR.
DIMENSION ERROR.
RANGE OVERFLOW.
RANGE UNDERFLOW.
CASCADE LENGTH ERROR.
CASCADE DECLARE ERROR.
RANGE OVERFLOW.
RANGE UNDERFLOW.
CONNECT PARAMETER INCLUDE CASCADE.
OBJECT FILE ACCESS ERROR.
```

```
MULTIPLE COMMAND EXIST IN PARA STATEMENT.
YOU MUST NOT WRITE ID & NOID AT A PARA STATEMENT.
YOU MUST NOT WRITE MATRIX & NOMATRIX AT A PARA STATEMENT.
YOU MUST NOT WRITE MEMOTY & NOMEMORY AT A PARA STATEMENT.
YOU MUST NOT WRITE CLOCK & NOCLOCK AT A PARA STATEMENT.
YOU MUST NOT WRITE TIME & NOTIME AT A PARA STATEMENT.
MULTIPLE IDT_SIZE EXIST IN PARA STATEMENT.
MULTIPLE MTX_SIZE EXIST IN PARA STATEMENT.
ILLEGAL ID EXIST IN PARA STATEMENT.
MISSING . IN PARA STATEMENT.
ID LENGTH OVER 8.
MISSING = IN PARA STATEMENT.
THERE IN NO UNIT_NAME.
MISSING = IN PARA STATEMENT.
THERE IN NO SIZE.
ILLEGAL LITERAL.
DO NOT EXIST.
DO NOT EXIST.
ID FILE CLOSE ERROR.
MATRIX FILE CLOSE ERROR.
CLOCK FILE CLOSE ERROR.
MEMORY FILE CLOSE ERROR.
DO NOT EXIST.
DO NOT EXIST.
DO NOT EXIST.
DO NOT EXIST.
CAN NOT PERMIT TO USE UNIT PARAMETER.
PARAMETER MISS MATCH.
PARAMETER MISS MATCH.
PARAMETER MISS MATCH.
PARAMETER MISS MATCH.
CAN NOT PERMIT RECURSIVE CALL.
```

6.4　シミュレータ・コマンドのエラー

入力するシミュレータ・コマンドにエラーがある場合、シミュレータは、実行中に、コンソールにエラーメッセージを出力します。

エラー番号	メッセージ及び対処
CMERR 1	NOT　CLOSE　BY ; 入力文字が８１以上である、又は；がない
CMERR 2	= MISSING コマンド文の中に、＝がない
CMERR 3	(MISSING コマンド文の中に、左かっこがない
CMERR 4	) MISSING コマンド文の中に、右かっこがない
CMERR 5	TIME　SPECIFICATION　ERROR CLOCK関係のコマンドにエラーあり
CMERR 6	COMMAND　ERROR コマンド名エラー
CMERR 7	NOT　DEFINED　SYMBOL　USED コンパイラーで、入力されていないＩＤを使用した
CMERR 8	TOKEN　LENGTH　＞　３２ 最大トークン長３２を越えている

昭和６０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　株式会社　昌　文　社
東京都港区芝５－２６－３０
TEL（４５２）４９３１（代表）