制御フロー図面の入力と論理合成

伊藤誠，大平駿介，倉橋英彦，牧田敏彦，夢沼良一
（山梨大学 工学部）

1．はじめに

本研究室の論理設計自動化システムの中の、制御論理回路の設計システムを紹介する。制御回路設計用には様々な言語が開発されているが、本システムでは特別な言語を用いず、フローチャートを直接計算機へ図面入力して設計を行ない、直列型制御回路のＰＬＡパターンや、機能素子を得るものである。

フローチャートの図面入力は会話型の図形エディタで行ない、インタラクティブな編集が可能となっている。

ＰＬＡパターンや機能素子の合成にあたっては、状態割当や論理式の簡単化を行ない、より簡単な回路を出力する。

2．制御論理回路設計システム

本システム全体の流れを図1に示す。全体は、図面エディタ、制御情報抽出（図1－Ａ）と、論理式生成、回路合成（図1－Ｂ）の二つの部分で構成される。

図面エディタ部は、図面より情報抽出して文字型中間ファイル1を出力するトランスレータを内蔵した、会話型のフローチャート専用図面エディタである。計算機との会話形式で図面を入力して行くため図面上のエラーがすぐにわかり、またトランスレータをエディタから起動できるため、トランスレート時に判明したエラーの修正もすぐに行なえる。

さらに、制御回路設計において重要な状態割当の作業を補助する、自動状態割当の機能も持っている。図1－Ｂはフローチャートより得られた制御回路記述である中間ファイル1より論理式を生成し、（中間ファイル2）さらに式の簡単化を行なった後、ＰＬＡパターンもしくは機能素子記述を出力する。

機能素子記述は、本研究室で開発した機能素子レベルシミュレータで論理検証できるほか、基板への実装も可能なゲートレベルの記述に変換できる。

言語を用いた設計は、その言語の文法を覚える必要があり、特に制御の流れの直観的な把握がむずかしい。そこで本システムでは、アルゴリズム記述に用いられているフローチャートになるべく近い形の制御回路用の記述法を用いて、制御の流

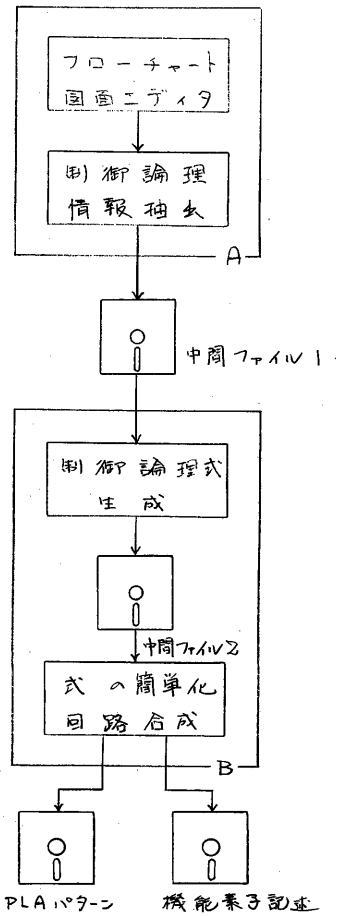


図1　システムの流れ

れを記述する方法を採用した。計算機のグラフィック画面に描かれたフローチャートを見るだけで、制御の流れを把握することは容易である。

3．システム構成

システムは8ビットおよび16ビットマイクロコンピュータ上にインプリメントされている。グラフィック画面は640×400（8ビット：白黒、16ビット：カラー7色）のものと、使用している。画面入力でカーソルを動かすには、マウスまたはキーボードを用いる。入力された図面はファイルに保存されるほか、プリンタでハードコピーをとることができる。

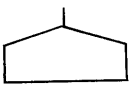
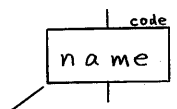
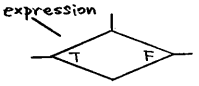
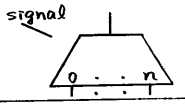
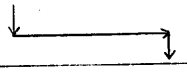
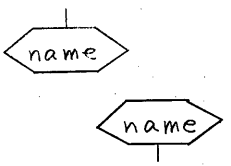
4．フローチャート図形仕様

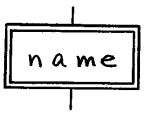
制御論理回路の動作を記述するには、状態名、状態割当、出力信号、条件信号、条件分岐、状態遷移を明確にする必要がある。図面エディタは、これらの情報を一般的なフローチャートに近い形で入力できるようになっている。

4－1　図形

全図形の形状と意味を示す。

表1　図形の形状と意味

図形	意味
	BEGIN フローチャートの始まりを示す。 nameはフローチャート名である。
	HALT フローチャートの終了を示す。
	STATE 制御の状態を示す。nameは状態名、codeは状態割当された符号、outputはその状態における出力信号である。出力信号は複数列挙できる。
	IF 論理条件式expressionのTrue(1)、False(0)によって分岐する制御の流れを記述する。T－F、F－Tの方向が選択できる。
	CASE 2または3ビットの配列型条件信号signalのデコード値によって、4または8（n＝3 or 7）方向に分岐する制御の流れを示す。
	LINE 図形間をつなぎ、制御の流れを示す。矢印の方向に流れる。
	GOTO，FROM LINEにラベル名をつけ、LINEでつなぐかわりに、GOTO→FROMへ制御が流れることを示す。同一nameのラベルで、GOTO側は複数個使用できる。ただし、後述するMACRO内へのジャンプは許さない。



MACRO

グラフィック画面にウィンド機能を持たせていないので、一画面ではフローチャート全体を入力できないことがある。このため一部をマクロ化してMACRO図形を書いておき、MACRO内は別の画面で入力する。

MACROのサブルーチン化は行なっていない。

IF，CASE図形はネスティングが可能となっており、分岐条件は各図形条件の論理積をとったものになる。

MACRO内の記述も、BEGIN—— HALTのフローチャート形式で入力し、その中に別のMACROがあっても良い。

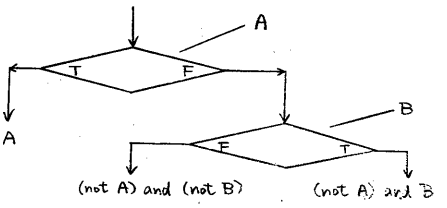


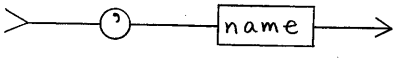
図2　分岐条件のネスティング

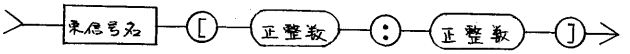
4－2　テキスト

表1中のテキスト部分の構文図を次に示す。

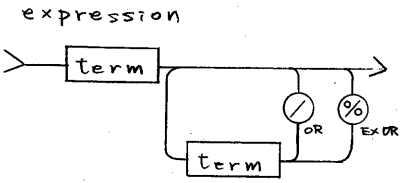
(1) name —— 8文字以内の英数字　　(2) code —— 6ビットのバイナリ

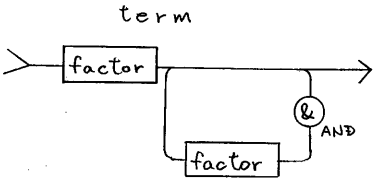
(3) 単信号，束信号名，output　　(4) signal

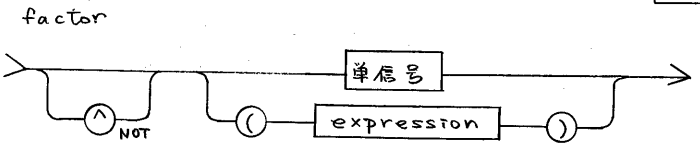




(5) expression —— 80文字以内







5．図面エディタのコマンド

エディタは会話型であり、設計者がコマンドを指定し、計算機からの指示に従って図面を入力して行く。エラーがあった場合は、すぐにメッセージを表示してエディタの実行を一時停止させる。実行は設計者の確認を待って再開される。このため、エラーの発見、修正は容易である。また、図面入力の自由度が高く、場合に応じて様々な入力形式をとれる。

表2　メインコマンド

	コマンド	説明
編集	Insert	図形入力
	Delete	図形消去
	Modify	図形修正
		} サブコマンドを持つ

補助	Show_L	指定したLINEの点滅表示
	Show_A	全STATEの状態割当状況の表示
	Show_M	MACROの内容の簡易表示
	Find	nameを持つ図形のサーチ
ファイル	Load	画面データのロード } MACRO画面ごとに可能
	Save	画面データのセーブ
システム	Init	システムの初期化
	Redraw	画面の書き直し
	Edit_M	編集画面（MACRO）の変更
	Exit	CP/Mへ戻る
その他	Hdcopy	画面のハードコピー
	Assign	自動状態割当
	Trans	トランスレータの起動

表3　Insert系サブコマンド

コマンド	説明	コマンド	説明
State	STATE（とoutput）の入力	Begin	BEGINの入力
If	IF（とexpression）の入力	Halt	HALTの入力
Case	CASE（とsignal）の入力	Line	LINEの入力
Macro	MACRO図形の入力	Assign	人手による状態割当
From	FROMの入力	Outsig	outputの付加
Goto	GOTOの入力	Expres	expression, signalの入力

表4　Delete系サブコマンド

コマンド	説明	コマンド	説明
All	図形全情報の消去	Assign	状態割当を未定義にする
From	FROMの消去	Outsig	出力信号群の消去
Goto	GOTOの消去	Expres	expression, signalの消去
Line	LINEの消去	Box	指定した枠内の消去

表5　Modify系サブコマンド

コマンド	説明	コマンド	説明
Txtpos	テキスト表示位置の変更	Replac	出力信号名の変更、消去
Figpos	図形の表示位置の変更	Outsig	} Insert系と同じ
Name	nameの変更	Expres	
If_dir	IFのT-Fの方向の反転		

6．エディタのデータ構造

汎用の画面エディタと違い、情報抽出を目的としたフローチャート専用のエディタであるため、図形ごとに固有のテーブルを持っている。LINEについては、各図形データから線形リストで、始点 —— 折れ曲り点 —— 終点の座標がつながれている。

図2に主なテーブルの構造を示す。

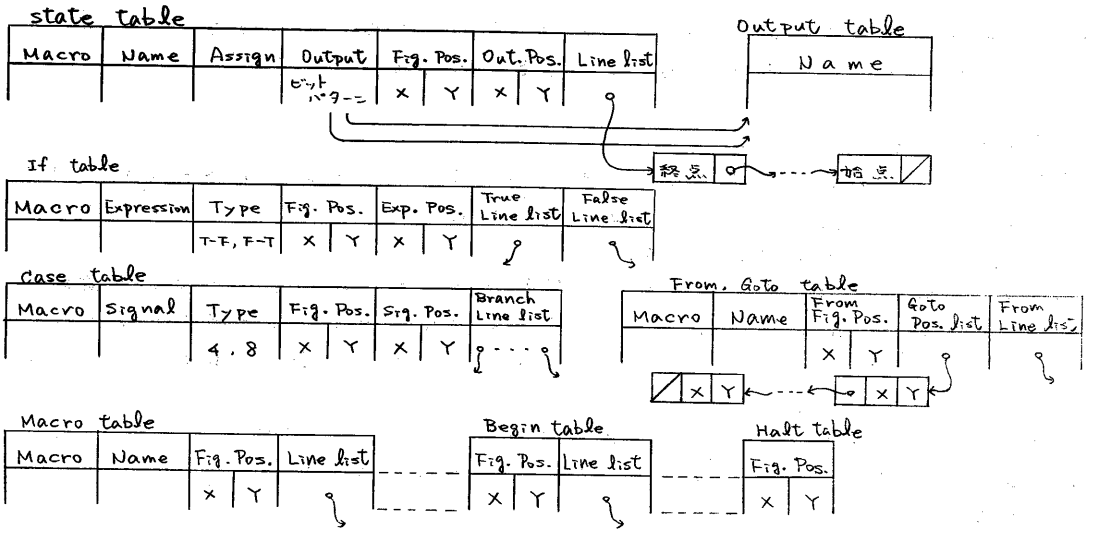


図2　エディタの主なテーブル

各テーブル中のMacroはMacro tableをさし、その図形がどのMacroの画面に書かれているかを示す。

7．制御情報抽出

制御論理回路と合成するために図面から抽出しなければならない情報は、D型FF数、状態割当、状態遷移、遷移条件、出力信号である。トランスレータは図面よりこれらの情報を抽出して、テキストの形で出力する。この中間ファイルのフォーマットを図3に示す。$FFはD-FFの情報、$ASNは状態割当、$STは状態遷移と遷移条件、$OUTは出力信号が出力される状態を表わす。$FF，$ASN，$OUTはテーブルよりすぐ求められる。

$STは、図形の接続関係を本の校正ゲラと同様に追跡して抽出する。さらに、そのLineにたどりつくまでの条件を記憶しておく必要がある。このため、条件のスタックを用意している。図4にその様子を示す。Lineの横の数字は追跡順序である。また、条件スタックの変化も示した。

抽出方法を要約すると、「状態aに

$FF（Q1..Qn / D1..Dn）
$ASN（状態名，状態割当コード）
　⋮
$ST（状態名a，条件式，状態名b）
　⋮
$OUT（出力信号名，{状態名}/{}/…）
　⋮

図3　中間ファイルのフォーマット

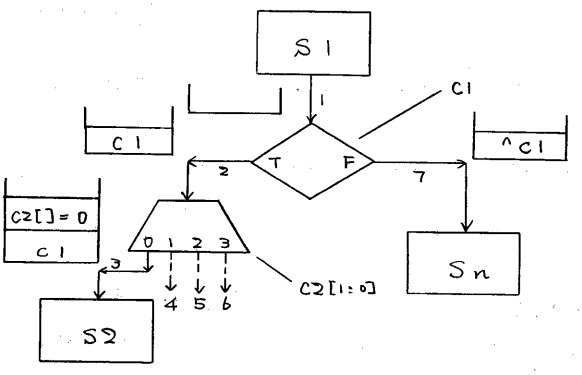


図4　状態遷移情報の抽出

着目し、派生するすべてのLineについて、条件とスタックの保存しながら、次の状態にたどりつくまで追跡する」ということになる。

8. 自動状態割当

状態割当はマニュアルで行なうことも可能となっているが、状態数が多くなるとこの作業は大変である。本システムでは、グレイコードを用い、FF数が最小となるような状態割当を自動的に行なう。

制御論理回路を最適にする実用的な状態割当法はないが、ここでは状態遷移の前後で値の変化するFF数が少ない方が回路が簡単になるという傾向に着目して自動状態割当を行なっている。グレイコードはこの方法に適した数列を与えてくれる。

制御は一般には分岐するので、メインフローチャートのBeginよりStateにグレイコードを割当てながらLineを追跡し、なるべく長い範囲にグレイコードが割当られるようにしている。

このあと、設計者は状態割当を検討して、マニュアルで修正することもできる。また、グレイコードのかわりに、バイナリコードをそのまま割当てることもできる。

9. 制御論理式生成

トランスレータによって出力された中間ファイルは、制御論理式生成プログラムを通して制御論理式と冗長項を出力する。冗長項とは使用されていない状態割当コードであり、式の簡単化に用いられる。制御論理式の生成例を示す。

```
$FF( Q2, Q1 / D2, D1 )
$ASN( S1, 00)
$ASN( S2, 01)
$ASN( S3, 11)
$ST ( S1, C, S2)
$ST ( S1,^C, S1)
$ST ( S2, , S3)
$OUT( P1, {S1} / {S3})
$OUT( P2, {S2} )
```

⟹

```
$EXP( D2, ^Q2&Q1, Q2&^Q1)
$EXP( D1, ^Q2&^Q1&C, Q2&^Q1)
$EXP( P1, ^Q2&^Q1 / Q2&Q1, Q2&^Q1)
$EXP( P2, ^Q2&Q1, Q2&^Q1)
```

出力信号　論理式　冗長項

図5　制御論理式の生成

冗長項が複数項あるときは、論理和がとられる。図6にデータ構造を示す。

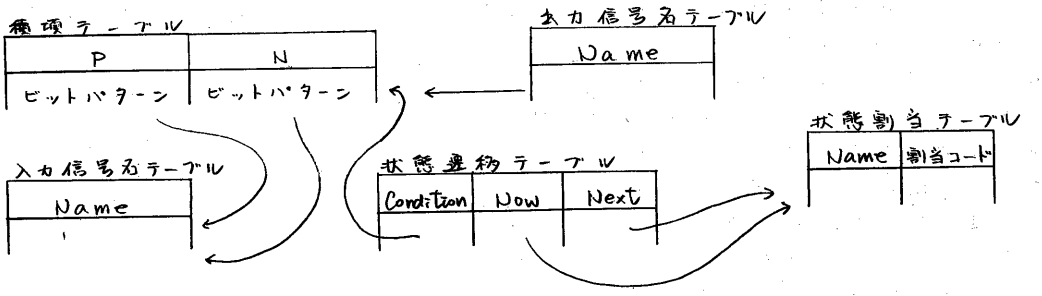


図6　データ構造

積項テーブルは論理式を積項に分けて記憶する。積項はビットパターンで入力信号名テーブルとさし、どの信号が入力されるかを示す。Pは信号の肯定，Nは否定である。遷移条件の論理式も、積項テーブルの一部を用いて記憶する。

入力信号名テーブル　A & B & ^C / ^A & B & D

Name
A
B
C
D

	P	N
OR	1 1 0 0 … 0	0 0 1 0 … 0
	0 1 0 1 … 0	1 0 0 0 … 0
	A B C D	^A ^B ^C ^D

図7　論理式の記憶

10．式の簡単化とPLA、機能素子の合成

生成された制御論理式は不要項、不要変数の消去、共通項のくくりだしを行なう簡単化をしたのち、PLAまたは機能素子化される。くくりだしにはPLA用、機能素子用があり、出力の指定に応じて使い分けられる。データ構造は図6とほぼ同じである。

10－1　不要項の消去

不要項の消去には、集合Aから集合Bの部分を取り去る、排除演算（＃演算）A＃Bを用いる。＃演算については参考文献(3)を参照されたい。ここでは手法のみ紹介する。

step 1 :　調べる積項を決める。
step 2 :　式内の他のすべての積項で＃演算する。
step 3 :　結果がnullならばその積項は不要項なので消去する。
step 4 :　すべての積項について1～3を行なう。

10－2　不要変数の消去

不要変数の消去にも＃演算を用いる。

step 1 :　調べる積項Cを決める。
step 2 :　調べる変数Lを決める。
step 3 :　Cの中でLの否定をとったものを$\overline{C(L)}$とする。
step 4 :　式内の他の積項と＃演算し、nullになるものがあれば不要変数なので消去する。
step 5 :　すべての変数のすべての積項に対して、1～4を行なう。

プログラムでは不要項および不要変数の消去のあと、再び不要項の消去を行なう式を簡単化する。また、冗長項も利用し、（実際の式）／（冗長項）に対して簡単化を行なっている。

10－3　くくりだし

(1)機能素子用

式間の共通項の併合を行ない、配線数を減らすことを目的としている。

Y = X1 / X2 & ^X3 & X4
Z = X2 / ^X3 & X4

⟶

r = ^X3 & X4
Y = X1 / X2 & r
Z = X2 / r

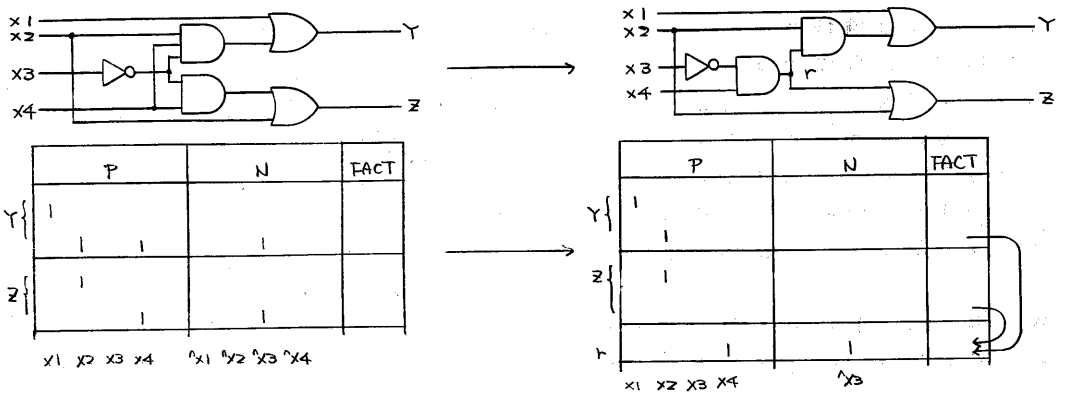


図8　くくりおとしの例

(2) ＰＬＡ用

ＰＬＡでは積項の数を減らすことが重要である。したがって図８のような部分的なくくりおとしは行なわず、積項が完全に一致したもののみをくくりおとす。このほか、次の二つの概念にくくりおとす。

i．積項が、それを部分項として持つ別の積項で置き換えられるとき。

A = X2 / ^X1 ⟶ A = B / ^X1
B = X1 & X2 ⟶ B = X1 & X2

ii．積項が二つ以上の積項の論理和で置き換えられるとき。

A = X2 ⟶ A = B & C
B = X1 & X2 ⟶ B = X1 & X2
C = ^X1 & X2 ⟶ C = ^X1 & X2

ＰＬＡ用のくくりおとしでは、積項テーブルの使用法が違い、ＦＡＣＴがその積項を含むかを記憶しておく。

Y1 = ^X1 & X2 / X1 & X2 & X3
Y2 = ^X1 & X3 / X1 & ^X2 & X3 / ^X1 & X2

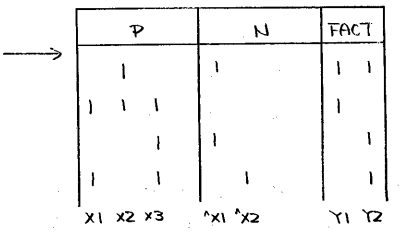


図9　ＰＬＡ用の論理式の記憶

11．機能素子合成

簡単化された論理式は、次の機能素子の列でおおわれ、回路記述となる。

.CON(X, Y)
　Xと Yを結線する
.INV(X, Y)
　Xの否定を Yへ出力する
.ORS((X1..Xn), Y)
　X1 / .. / Xnを Yへ出力する
.ANDS((X1..Xn), Y)
　X1 & .. & Xnを Yへ出力する

12. PLAパターン合成

PLAパターンの出力フォーマットを図10に示す。

```
Y1 = ^x1 & x2 / x1 & x2 & x3
Y2 = ^x1 & x3 / x1 & ^x2 & x3 / ^x1 & x2

$PLA ( x1 , x2 , x3 / Y1 , Y2 )
$PLT (  0   1   -  /  1   1
        1   1   1  /  1   0
        0   -   1  /  0   1
        1   0   1  /  0   1 )
        AND PLANE     OR PLANE
```

図10 PLAパターン

13. 制御論理合成例

図面入力から最終出力までの例を図11～17に示す。図18は機能素子レベルシミュレータの出力である。各ステップに要した時間を表6に示す。

表6 所要時間（16ビット）

ステップ	時間
図面入力	15分
自動状態割当	1秒
情報抽出	10秒
制御論理式の生成	1分
式の簡単化とPLA合成	2分
式の簡単化と機能素子合成	2分

14. おわりに

プログラムはPASCALで記述されており、モニタはCP/Mである。本システムにより制御部と記述し論理回路を合成することが可能となった。今後、RTLレベルの記述を図面化しその論理を自動合成することにより、総合的な論理設計自動化システムを開発する予定である。

参考文献

(1) 日本電気：「FDAシステム、FDA-TL言語仕様」
(2) S. Kang, W.M. van Cleemput : "AUTOMATIC PLA SYNTHESIS FROM A DDL-P DESCRIPTION" IEEE 18th Design Automation Conference, pp. 391-397, 1981
(3) M. A. Bruerer : Digital System Design Automation Computer Science Press 1972
(4) 石川純一：「論理設計自動化と機能シミュレータ」，設計自動化研資 1983, 5

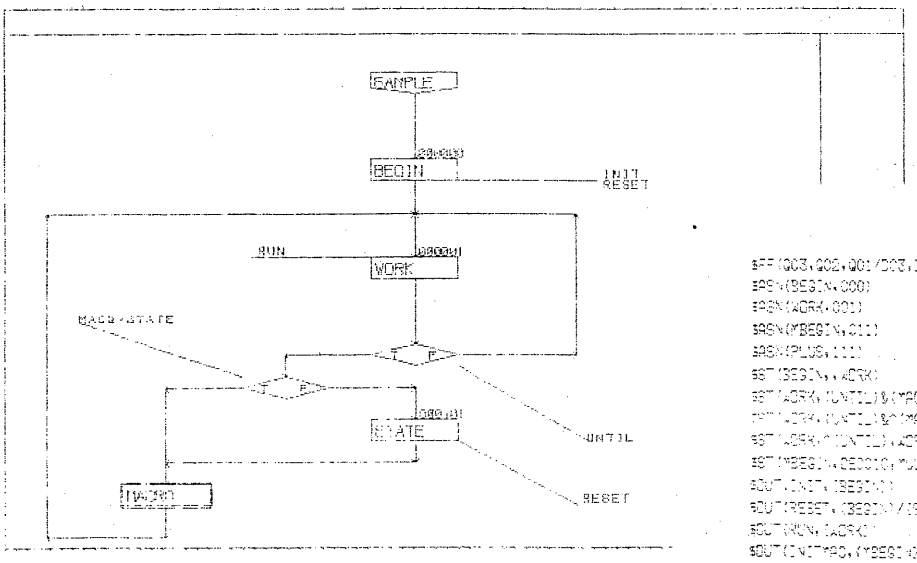


図11 主フロー図面

```
$FF(Q03,Q02,Q01/D03,D02,D01)
$ASN(BEGIN,000)
$ASN(WORK,001)
$ASN(MYBEGIN,011)
$ASN(PLUS,111)
$ST(BEGIN,,WORK)
$ST(WORK,(UNTIL)&(MAC0&^STATE),MBEGIN)
$ST(WORK,(UNTIL)&(MAC0&^STATE),STATE)
$ST(WORK,^(UNTIL),WORK)
$ST(MBEGIN,BEGIN,MULT)
$OUT(INIT,(BEGIN))
$OUT(RESET,(BEGIN)/(STATE))
$OUT(RUN,(WORK))
$OUT(INITMAC,(MYBEGIN))
```

図12 抽出情報（一部のみ）

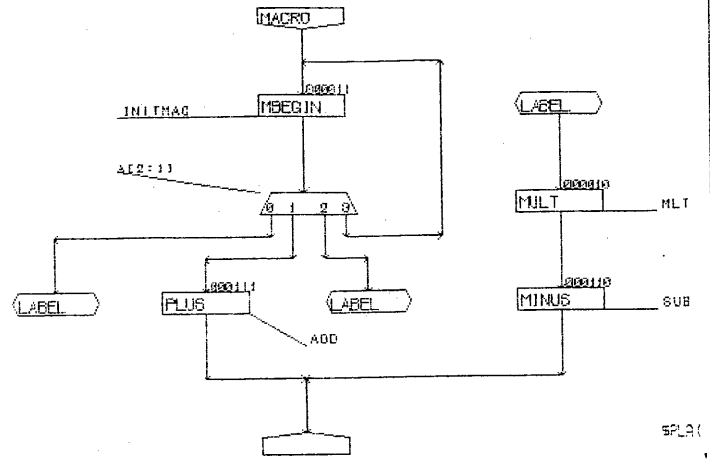


図13　マクロ画面

```
$EXP(D03,^Q03&^Q02&Q01&UNTIL&^MAC/^Q03&^Q02&Q01&UNTIL&MAC&STATE
        /^Q03&Q02&Q01&DEC011/^Q03&Q02&^Q01,Q03&^Q02&^Q01)
$EXP(D02,^Q03&^Q02&Q01&UNTIL&MAC&^STATE/^Q03&Q02&Q01&DEC010
        /^Q03&Q02&Q01&DEC011/^Q03&Q02&Q01&DEC012
        /^Q03&Q02&Q01&DEC013/^Q03&Q02&^Q01/Q03&^Q02&Q01
        /^Q03&Q02&Q01&^DEC010&^DEC011&^DEC012&^DEC013
       ,Q03&^Q02&^Q01)
$EXP(D01,^Q03&^Q02&^Q01/^Q03&^Q02&Q01&UNTIL&MAC&^STATE
        /^Q03&^Q02&Q01&UNTIL&^MAC/^Q03&^Q02&Q01&UNTIL&MAC&STATE
        /^Q03&^Q02&Q01&^UNTIL/^Q03&Q02&Q01&DEC011
        /^Q03&Q02&Q01&DEC013/Q03&Q02&Q01/Q03&Q02&^Q01
        /Q03&^Q02&Q01/^Q03&Q02&Q01&^DEC010&^DEC011&^DEC012&^DEC013
       ,Q03&^Q02&^Q01)
$EXP(INIT,^Q03&^Q02&^Q01,Q03&^Q02&^Q01)
$EXP(RESET,^Q03&^Q02&^Q01/Q03&^Q02&Q01,Q03&^Q02&^Q01)
$EXP(RUN,^Q03&^Q02&Q01,Q03&^Q02&^Q01)
$EXP(INITMAC,^Q03&Q02&Q01,Q03&^Q02&^Q01)
$EXP(ADD,Q03&Q02&Q01,Q03&^Q02&^Q01)
$EXP(MLT,^Q03&Q02&^Q01,Q03&^Q02&^Q01)
$EXP(SUB,Q03&Q02&^Q01,Q03&^Q02&^Q01)
```

図14　制御論理式

```
$EXP(D03,^Q03&^Q02&Q01&UNTIL&^MAC/^Q03&^Q02&Q01&UNTIL&STATE
        /^Q03&Q02&DEC011/^Q03&Q02&^Q01)
$EXP(D02,^Q02&Q01&UNTIL&MAC&^STATE/Q03&^Q02/^Q03&Q02)
$EXP(D01,Q01&DEC011/Q01&DEC013/Q03/^Q02/Q01&^DEC010&^DEC012)
$EXP(INIT,^Q02&^Q01)
$EXP(RESET,^Q02&^Q01/Q03&^Q02)
$EXP(RUN,^Q03&^Q02&Q01)
$EXP(INITMAC,^Q03&Q02&Q01)
$EXP(ADD,Q03&Q02&Q01)
$EXP(MLT,^Q03&Q02&^Q01)
$EXP(SUB,Q03&^Q01)
```

図15　簡単化した論理式
（機能素子用）

```
DFF((D03,D02,D01),(@T,@T,@T)
 ,(@RESET,@RESET,@RESET)
 ,(@CLOCK,@CLOCK,@CLOCK)
 ,(Q03,Q02,Q01),('Q03,'Q02,'Q01))
.DECOD(A[2:1],@F,('DEC013
 ,'DEC012,'DEC011,'DEC010))
.INV('DEC013,DEC013)
.INV('DEC012,DEC012)
.ANDS((Q01,'DEC010,'DEC012),@PDA8)
.ANDS((Q03,Q02,Q01),ADD)
.ANDS((UNTIL,MAC,'STATE,@PDA3),@PDA9)
.ORS((MLT,@PDA7,@PDA10,@PDB1),D03)
.ORS((@PDA1,@PDA2,@PDA9),D02)
.ORS((Q03,'Q02,@PDA5,@PDA6,@PDA8),D01)
```

図16　機能素子出力
（一部のみ）

```
$PLA( Q03     ,Q02     ,Q01     ,UNTIL   ,MAC
    ,STATE   ,DEC011  ,DEC010  ,DEC012  ,DEC013
   / D03     ,D02     ,D01     ,INIT    ,RESET
    ,RUN     ,INITMAC ,ADD     ,MLT     ,SUB   )
$PLT( 0 0 1 1 0 - - - - / 1 0 1 0 0 0 0 0 0 0
      0 0 1 1 - 1 - - - / 1 0 1 0 0 0 0 0 0 0
      0 1 - - - - 1 - - / 1 1 0 0 0 0 0 0 0 0
      - 0 1 1 1 0 - - - / 0 1 1 0 0 0 0 0 0 0
      - - 1 - - - 1 - - / 0 0 1 0 0 0 0 0 0 0
      - - 1 - - - - - 1 / 0 0 1 0 0 0 0 0 0 0
      - - 1 - - - - 0 0 / 0 0 1 0 0 0 0 0 0 0
      - 0 0 - - - - - - / 0 0 1 1 1 0 0 0 0 0
      1 0 - - - - - - - / 0 1 1 0 1 0 0 0 0 0
      0 0 1 - - - - - - / 0 0 0 0 0 1 0 0 0 0
      0 1 1 - - - - - - / 0 0 0 0 0 0 1 0 0 0
      1 1 1 - - - - - - / 0 0 1 0 0 0 0 1 0 0
      0 1 0 - - - - - - / 1 0 0 0 0 0 0 0 1 0
      1 - 0 - - - - - - / 0 0 1 0 0 0 0 0 0 1 )
```

図17　ＰＬＡパターン出力

```
                S Q U M S A I R S I A Y S Q Q Q D D D
                R C N A T ^ N E U N D L U 0 0 0 0 0 0
                E L T C A 2 I S N I D T B 3 2 1 3 2 1
                S C I   T T E   T
                E C L   E 1   T   M
                T K       V       A
                                  C
time
   1   **0001**************
   2   *00001*************
  12   110001110000000001
  32   110001001000001001
  52   110001001000000101
  72   111103001000000111
  92   111113000100001101
 112   111013000100001101
 132   111012000100001010
 152   111012000001001110
 172   111012000000111001
 192   111012001000001101
 212   111012010000010111
 232   111011000100001111
 252   111011000001001101
 272   111011001000001101
 292   111011010000010111
```

図18　シミュレーション結果