会話型設計援助システム

伊藤 誠，洞野豪之，和田義弥
（山梨大学 工学部）

1. システムの目的

小型計算機と図形端末を用いた，論理設計援助システムを開発する。対象は，IC 100個程度の基板とする。大型機の開発が目的ではなく，インタフェース回路や小型特殊目的のμプログラム制御プロセサの開発を目的とする。したがって，大規模回路の分割，基板端子間（バックボード）の配線の自動生成は行なわない。また，研究用試作では，1台しか実装しないことが多いから，テスタ用の治具さえないことが多いので，テスト・シーケンスの生成はとりあえず考えない。同じ理由で，プリントパタンの作成も半自動とする。利用者は，論理回路の設計経験者とし，ある程度の回路設計知識を要求する。

2. ハードウェア構成

使用計算機はLSI－11（56KB）（DEC社）を（当面）用いる。画像端末はタブレットと蓄積型線表示装置とし，ハードコピーにXYプロッタを用いる。2次記憶は，フロッピーディスク2台（512KB）を用い，CRT文字表示器とプリンタを文字入力に用いる。（図1）

ただし，目的とするシステムを構成するには，主記憶と二次記憶の容量がかなり不足すると考えられるので，システム構造の積み上げが必要であろう。

モニタは，RT-11（DEC社）を用い，言語はPASCAL，FORTRAN，アセンブラを用いる。ファイルは，主としてモニタ標準の順次および乱ファイルを利用するが，目的により，その上（中？）にB氏木型ソートファイルを作成する予定である。

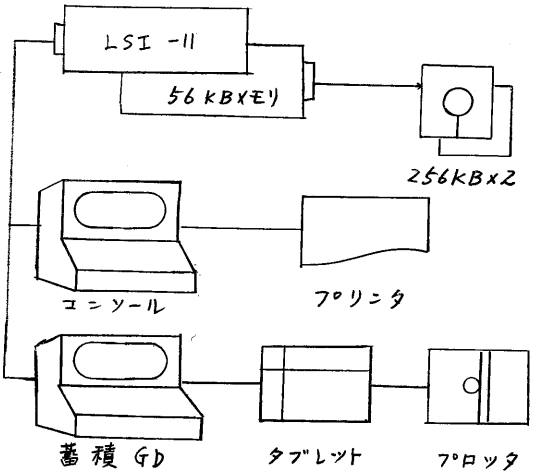


図 1

3. システム構成

システムは複数個のシステムに分割し，その間をファイルで結合する。したがって，ファイル間の無矛盾性はすべて利用者の責任とする。システムも複数個のボリュームに分割される。

3.1 図形エディタ

素子およびマクロ素子（後述）レベルの回路図の入力に用いる。入力は，タブレットおよび鍵盤，出力は（マクロ）素子ファイルであり，図形ファイルを用いる。図形データは多角形（素子）と網（ネット）とする。素子および回路図は，

あらかじめ登録し，サイズ，位置，角度を指定して展開できる．図形表示器が図面より小さいので，図面はあらかじめ分割して入力するが，窓処理により画面を移動しても，前の図面の信号線の接続端子が，次の図面の端子として表示されるようにする。

図形コマンドは通常のコマンド以外に，網を扱う機能が必要である．網は許容点のみを端点とするグラフ状の線の集合である．許容端点は，素子の端子点，網上の点，指定された経由点である。回路図，網には名前をつけることができる．タブレット，図形表示器がなくても回路図入力を可能とするため，図形コマンドは，一度文字型コマンドに変換してから処理をする。

＜図形コマンド例＞

NA : NAND (11,5)/1, 90°　; NAND素子を原点(11,5)に配置し，名前をNAとする．サイズは標準とし，90°回転する。

X ; NET (NA.1, NB.3, IA.4)　; NAの1ピン，NBの3ピン，IAの4ピンをつなぐネットXを定める．

P1 : TER (1, 30)　; 図形上に端子点を定義する．素子上の端子点は，素子のピンとなる。

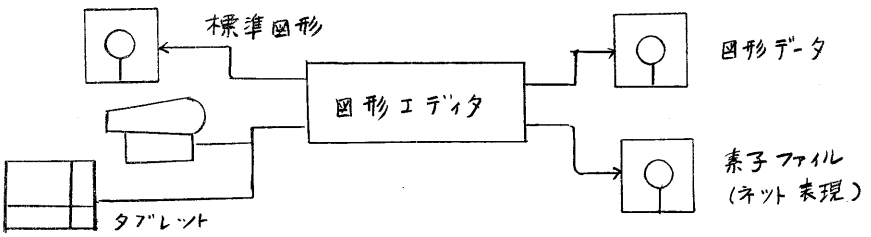
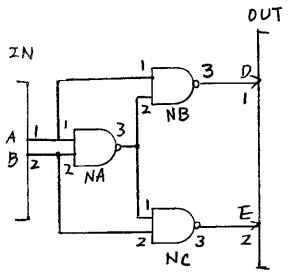


図 2 図形システム

図形エディタの出力は，回路図の描画用図形データファイルと，ネット表現された素子ファイルである．ネット型素子ファイルは，コンバータにより，素子ファイルに変換することができる．（図3）



回路図

```
A : IN.1, NA.1, NB.1 ;
B : IN.2, NA.2, NC.2 ;
C : NA.3, NA.2, NC.1 ;
D : NB.3, OUT.1 ;
E : NC.3, OUT.2 ;
NA , NB, NC = NAND ;
IN , OUT = EXT
```

ネット表現

```
NA : NAND ( A, B, C )
NB : NAND ( A, B, C )
NC : NAND ( C, D, E )
IN : EXT ( A, B )
OUT : EXT ( D, E )
```

素子表現

図 3

また，網にはサイズが指定できる．サイズ8の網は，8ビットの信号線である．サイズ2以上の信号を束信号と呼ぶ．束信号からの分岐，複数の信号を一つの束信号とする併合操作は，特別の素子として表現する．

3.2 論理設計システム

論理設計システムは RTL（レジスタ転送）レベルの言語をまずマクロ素子レベル記述に変換する。RTL（以後機能レベルと呼ぶ）言語は，従来提案されているもののサブセットとする予定であるが，インタフェース設計を記述するため，若干，どろくさい表現が必要となる．図4に簡単な機能レベル言語と対応するマクロ素子の記述の例を示す．

例

```
in    BD[7..0]                    IN(BD[7..0])
sig   DD[15..0], DT[7..0]         SIZE(DD[15,0], DT[7,0])
reg   DH[7..0], DL[7..0]          SIZE(DH[7,0], DL[7,0])
DT = RCV(BD)                      RCV(BD, DT)
at    HSTRB   DH := DT            REG(DT, HSTRB, DH)
at    LSTRB   DL := DT            REG(DT, LSTRB, DL)
DA    (DD, OUT)                   DA(DD, OUT)
```

(a) 機能レベル　　(b) マクロ素子

図4　機能レベルとマクロ素子

クロック動作のため，at（エッジ），on（マスタスレーブ），while（ラッチ）等のキーワードを使用する。デコーダ，マルチプロセッサに，case，select 文を，非同期カウンタに aup（<レジスタ>，#<モード>）等を用いる．また，特定の状態でのみ機能する（active となる）ブロックと，状態推移文を用意する．これらは，同期型順序回路として PLA または線型論理で合成する．

機能レベル言語は制御フローの記述部と　その他（主としてデータフロー部）に分割して変換される．前者は制御フロー記述言語，後者はマクロ素子言語となる．

```
S1  begin
    if  CH="H3"  then do S3
    ...
    end
```

S1 ブロック
S1 が active のとき
実行される．

ラベルなしブロック
常時 active である．

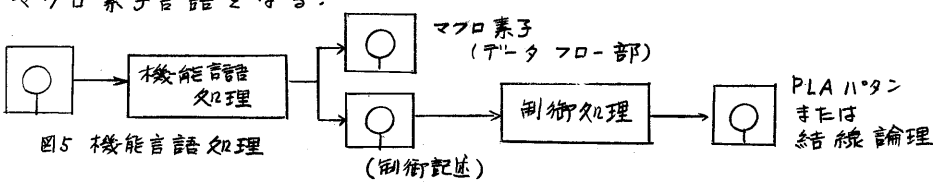


図5 機能言語処理

機能言語の図入力は考えていないが，制御部をフロー図で，データ部をマクロ素子図面で入力することを考慮中である。

マクロ素子は，IC の機能と 1:1 に対応する素子レベル言語に展開される．素子レベルに変換されると，論理設計は完了する．マクロ素子から素子への展開は，マクロ素子定義ファイルとマクロ展開ファイルによる．マクロ定義ファイルの選択により，異なるファミリーの IC に展開することができるし，新しいテクノロジへの対応が容易になるだろう．

```
SIZE (DD[16..0])                  DEF8(DD[7..0] STRB,"CLR, BUF[7..0])
SIZE (BUF[16..0])                 DEF8 (D5[15..8] STRB,"CLR,BUF[15..8])
REG  (DD, STRB,"CLR BUF)          DEF8 は SN74273 に相当
```

(a) マクロ素子　　(b) 素子

図6. マクロ素子から素子への変換

3.3 実装システム

実装システムへの入力は素子ファイルと，使用する基板を記述した基板情報ファイル，外部信号の端子を記述した端子情報ファイル，および，IC情報を記述したIC情報ファイルである。（図7）

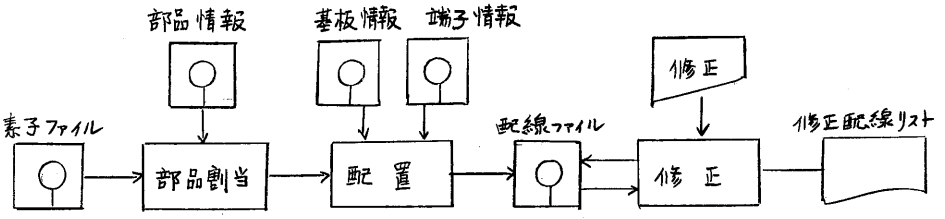


図7 実装システム

素子ファイルにIC情報ファイルをつきあわせて部品割当を行なう。論理素子のICへの割当は単純な方法とする。他の部品もすべて素子名との 1:1 対応とする。次に基板情報から部品を配置するが，部品の形状が一様でない場合は，人手指定とする。配置はクラスタリング手法を用いるが，初期設計入力が図入力のとき，元の素子の配置関係をクラスタリングに反映できるようにしたい。この場合図形データファイルから，素子配置情報を抽出して用いる。

配置が終了すると，配線ファイルを出力する。これは，IC相対または基板で指定されたピン名を，指定したネットリストで出力順に wrap 配線をする。

配線の修正を，配線ファイルを更新する形で行なう。これにより，un wrap（配線の戻し）と wrap の順を知ることができる。修正情報の入力形式については考慮中であるが，素子ファイルの編集コマンド形式とする予定である。

配線ファイルからのプリントパタンの自動生成は，このシステム構成と目的から考えて不要と思われるが，図形エディタと合わせて半自動配線方式を考えたい。

3.4 機能シミュレータ

マクロ素子を対象とした会話型シミュレータを開発する予定である。会話型としては，BASIC インタプリタシステムのような形式を考えている。すなわち，コマンドモードで，レジスタ，外部端子の値，イベントを登録しシミュレータを呼び出す。BRAKE コマンドで，特定の信号の変化でシミュレーションを中断して，コマンドモードに戻ることも可能にする。マクロ素子の動作の記述には，特別の記述言語を用意し，min－max－typical の遅延時間を記述する。シミュレーションはイベント driven 形式とし，スパイクの抑圧をする。動作時間よりも操作性を重視した設計をする。

4. 制御処理サブシステム

制御処理サブシステムは，制御状態の推移（シーケンス）と各状態での制御出力信号を，記述した言語より，2値の状態推移表を作成するのが目的である。記述は大きな制御フローを分割して記述でき，特殊な技法を用いれば，副シーケンスをサブルーチンのように利用できる。入力信号は，2値信号の束を束信号として扱うことができ，束の値を推移条件とすることができる。その他，推移条件と

して，クロックの整数倍の遅延を指定することもでき，この場合，遅延時間に相当するカウンタを自動生成する．記述形式，変換手法については，文献(2)を参考にした．このシステムの出力をSPLAに書きこむことにより，制御処理が実現できる．

4.1 記述形式

図8に全体の構文図を示す．先頭に宣言文，次に主制御部記述，最後に複数の副制御記述部が続く．

(a) 宣言文

宣言文は入力と出力信号名の記述に始まる．入力は配列のように範囲を指定することにより，束信号を指定できる（例 CMND[2..0]）．

internalは，内部で信号を一時記憶するときに用いる．これにより，後述する副制御シーケンスのサブルーチン的利用が可能となる．constは束信号の値を定義する．これにより，束信号の一致条件が名前で指定できる．

(例) in CNT = [2..0] ; const STP = [7:3]

条件式 CNT = STP は論理式 CNT[2] & CNT[1] & CNT[0] と同じである．

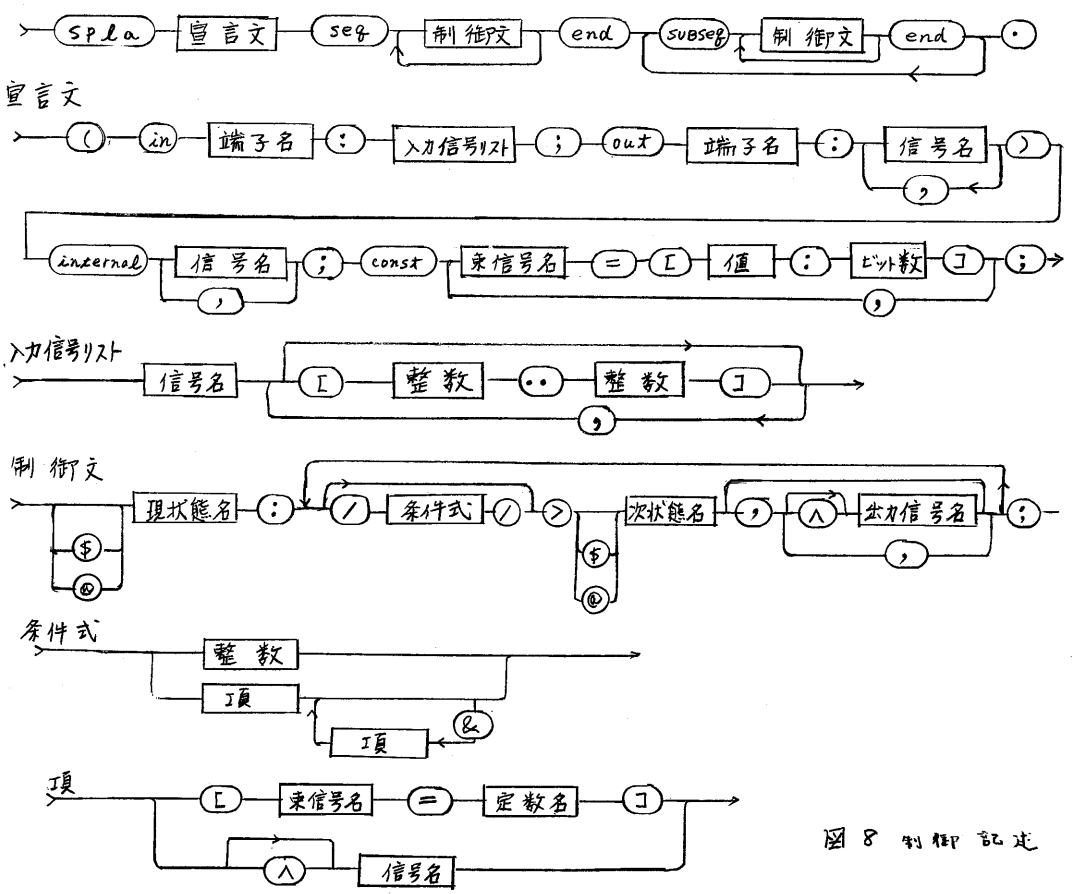


図8 制御記述

(b) 主制御部

制御（シーケンス）の記述は主制御部と副制御部に分割して記述する．各制御部は制御文で記述される制御ブロックの集合として記述される．制御ブロックは，1つの現状態（条件付）との次状態への推移，および，そのときの出力信号から構成される．

（例）　S1 : / C1 / > S2, 011, ^012
　　/ c2 / > S3, 021, 022 ;

これは，状態 S1 から，条件 c1 が満たされたとき，出力 011に1，012に0を出して，状態 S2 に推移することを示す．同じように条件 c2 が満足されたら，S3 へ推移する．条件が満たされないとき，状態は S1 にとどまる．一つの制御ブロック内での条件式は排他的で，同時に二つ以上の状態への推移することは許されない．したがって，任意時点で active な状態ブロックは一つのみである．

(c) 副制御部

副制御部は，入口と出口の状態を1つづつ持ち，＄と@をつけて他の状態と区別する．状態 Si から副制御部 A に制御を渡すには

Si : > ＄A

と記述する．副制御部での制御部記述は

subseg
＄A : /　　/ > ----

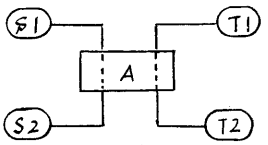
---/　/ > @A　end.

という形式を持ち，＄Aから始まり，@A への推移で終る．A での制御終了後の主制御部での動作記述は

@A : /　/ > Sj

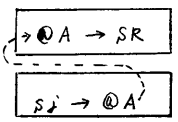
により指定する．

副制御部は，そのままではサブルーチンのように共用することはできない．ただ，内部FFを用いた次の技法により共用が可能となる．




```
internal  SUB
seg   S1 : /      / > ＄A, SUB
      T1 : /      / > ＄A, ASUB
      @A : / SUB/ > S2
      @A : / ASUB/ > T2
```

主制御部と副制御部の記述を統合して，一つの〝状態推移表〟を作るが，このとき，副制御部の出力への推移と主制御部での戻りの推移は統合されることがある．



@A : > SR
Sj : > @A
→ Sj : > SR

(C) 条件式

条件式は，論理条件と遅延条件がある．論理条件には，〝来信号の値が，定数値に等しいとき〟という条件が指定できる．

（例）　in　CMD［2..0］; const　COM=［7..3］

条件式：（CMD = CONT）& /:SRVIN

現在，条件式の中で，論理 OR は許していない．

Si : /A/B/ > Sj は　　Si : /A/ > Sj

/B/ > Sj

で記述できる。

遅延条件は，必要なビット数のカウンタを自動生成して処理する．カウンタは余分の内部 FF を使用するため，遅延の利用が少ない場合は，状態を追加する方が良いが，この判断は行なっていない．

4.2　システムの implementation

システムは，PASCAL で書かれており約1500行である．付録に，文献(2)と同じ例を記述した結果を示す．PLAに書きこむには，システムの出力形式を変換する後処理が必要である．

参考文献

(1)　ブルーア〝ディジタル計算機の自動設計〟産業図書

(2)　NEレポート〝PLAによる論理回路設計をサポートするシステム開発〟　日経エレクトロニクス　12.25.78

	******** PATTERN *******																					
	CNT			CMD		SRVOUT	DMARQ	RW	TIMESW	TIME CON	CURRENT STATE	SRVIN	IOWR	IOEN	DMAEN	IOSTP	IORST	RW	TIMESW	TIME CON	NEXT STATE
	******* NORMAL OUTPUTS ARE											H	H	L	L	H	H				
1	-	-	-	-	-	-	-	-	-	-	RIOSEQ	-	-	-	-	-	H	-	-	-	**********
2	-	-	-	-	-	-	-	-	-	0	RIOSEQ	-	-	-	-	-	-	-	S	-	**********
3	-	-	-	-	-	-	-	-	H	5	RIOSEQ	-	-	-	-	-	-	-	R	0	WAITSEQ
4	L	H	H	-	-	H	-	-	-	-	WAITSEQ	-	-	-	-	-	-	-	-	-	$DIOSEQ
5	H	L	L	H	L	H	-	-	-	-	WAITSEQ	-	-	-	-	-	-	S	-	-	$START
6	H	L	L	L	H	H	-	-	-	-	WAITSEQ	-	-	-	-	-	-	R	-	-	$START
7	-	-	-	-	-	-	-	H	-	-	@START	-	-	-	-	-	-	-	-	-	$DMAREAD
8	-	-	-	-	-	-	-	L	-	-	@START	-	-	-	-	-	-	-	-	-	$DMAWRT
9	-	-	-	-	-	-	-	-	-	-	CHCSTP	-	-	-	-	H	-	-	-	-	**********
10	-	-	-	-	-	-	-	-	-	0	CHCSTP	-	-	-	-	-	-	-	S	-	**********
11	-	-	-	-	-	-	-	-	H	3	CHCSTP	-	-	-	-	-	-	-	R	0	WAITSEQ
12	-	-	-	-	-	-	-	-	-	-	$DIOSEQ	-	H	-	-	-	-	-	-	-	DIOSEQ1
13	-	-	-	-	-	-	-	-	-	-	DIOSEQ1	-	H	L	-	-	-	-	-	-	DIOSEQ2
14	-	-	-	-	-	-	-	-	-	-	DIOSEQ2	H	H	L	-	-	-	-	-	-	**********
15	-	-	-	-	-	L	-	-	-	-	DIOSEQ2	-	-	-	-	-	-	-	-	-	WAITSEQ
16	-	-	-	-	-	-	-	-	-	-	$START	-	-	-	-	-	-	-	-	-	START1
17	-	-	-	-	-	-	-	-	-	-	START1	-	-	L	-	-	-	-	-	-	START2
18	-	-	-	-	-	-	-	-	-	-	START2	H	-	-	-	-	-	-	-	-	**********
19	-	-	-	-	-	L	-	-	-	-	START2	-	-	-	-	-	-	-	-	-	@START
20	-	-	-	-	-	-	-	-	-	-	$DMAREAD	-	H	-	-	-	-	-	-	-	**********
21	-	-	-	-	-	-	H	-	-	-	$DMAREAD	-	-	-	-	-	-	-	-	-	DMAREAD1
22	-	-	-	-	-	-	-	-	-	-	DMAREAD1	-	H	-	L	-	-	-	-	-	DMAREAD2
23	-	-	-	-	-	-	-	-	-	-	DMAREAD2	H	H	-	L	-	-	-	-	-	**********
24	-	-	-	-	-	H	-	-	-	-	DMAREAD2	-	-	-	-	-	-	-	-	-	DMAREAD3
25	-	-	-	-	-	-	-	-	-	-	DMAREAD3	H	H	-	-	-	-	-	-	-	**********
26	L	L	L	-	-	-	L	-	-	-	DMAREAD3	-	-	-	-	-	-	-	-	-	$DMAREAD
27	H	H	H	-	-	-	L	-	-	-	DMAREAD3	-	-	-	-	-	-	-	-	-	CHCSTP
28	-	-	-	-	-	-	-	-	-	-	$DMAWRT	H	-	-	-	-	-	-	-	-	**********
29	-	-	-	-	-	-	H	-	-	-	$DMAWRT	-	-	-	-	-	-	-	-	-	DMAWRT1
30	-	-	-	-	-	H	-	-	-	-	DMAWRT1	-	-	-	-	-	-	-	-	-	DMAWRT2
31	-	-	-	-	-	-	-	-	-	-	DMAWRT2	H	-	-	L	-	-	-	-	-	DMAWRT3
32	-	-	-	-	-	-	-	-	-	-	DMAWRT3	H	-	-	-	-	-	-	-	-	**********
33	L	L	L	-	-	-	L	-	-	-	DMAWRT3	-	-	-	-	-	-	-	-	-	$DMAWRT
34	H	H	H	-	-	-	L	-	-	-	DMAWRT3	-	-	-	-	-	-	-	-	-	CHCSTP

```
SORCE FILE NAME
WRK:SPLA.TEXT
    0 SPLA
    1 (IN<CPU:CNT[0..2],CMD[0..1],SRVOUT;
    2 OUT<CPU:SRVIN;
    3 IN<CIO:DMARQ;
    4 OUT<CIO:IOWR,IOEN,DMAEN,IOSTP,IORST)
    5
    6 INTERNAL RW;
    7
    8 CONST CON=[0;3],DIO=[3;3],CHC=[4;3],STP=[7;3],RD=[2;2],WR=[1;2];
    9
 10 SEQ
   11 $MAIN:>RIOSEQ,SRVIN,IOWR,^IOEN,^DMAEN,IOSTP,IORST;
   12 RIOSEQ:IORST;
   13 /#5/>WAITSEQ;
   14 WAITSEQ:/SRVOUT&[CNT=DIO]/>$DIOSEQ;
   15 /SRVOUT&[CNT=CHC]&[CMD=RD]/>$START,RW;
   16 /SRVOUT&[CNT=CHC]&[CMD=WR]/>$START,^RW;
   17 @START:/RW/>$DMAREAD;
   18 /^RW/>$DMAWRT;
   19 @DMAWRT:>CHCSTP;
   20 @DMAREAD:>CHCSTP;
   21 CHCSTP:IOSTP;
   22 /#3/>WAITSEQ;
   23 @DIOSEQ:>WAITSEQ      END


   24 SUBSEQ
   25 $DIOSEQ:>DIOSEQ1,IOWR;
   26 DIOSEQ1:>DIOSEQ2,IOWR,^IOEN;
   27 DIOSEQ2:SRVIN,IOWR,^IOEN;
   28 /^SRVOUT/>@DIOSEQ       END


   29 SUBSEQ
   30 $START:>START1;
   31 START1:>START2,^IOEN;
   32 START2:SRVIN;
   33 /^SRVOUT/>@START     END


   34 SUBSEQ
   35 $DMAREAD:IOWR;
   36 /DMARQ/>DMAREAD1;
   37 DMAREAD1:>DMAREAD2,IOWR,^DMAEN;
   38 DMAREAD2:SRVIN,IOWR,^DMAEN;
   39 /SRVOUT/>DMAREAD3;
   40 DMAREAD3:SRVIN,IOWR;
   41 /[CNT=CON]&^DMARQ/>$DMAREAD;
   42 /[CNT=STP]&^DMARQ/>@DMAREAD        END


   43
   44 SUBSEQ
   45 $DMAWRT:SRVIN;
   46 /DMARQ/>DMAWRT1;
   47 DMAWRT1:/SRVOUT/>DMAWRT2;
   48 DMAWRT2:>DMAWRT3,SRVIN,^DMAEN;
   49 DMAWRT3:SRVIN;
   50 /[CNT=CON]&^DMARQ/>$DMAWRT;
   51 /[CNT=STP]&^DMARQ/>@DMAWRT      END


   52 END.
****   ^100
```

付録 A　記述例