資　料

データ通信システムプログラム自動作成システム: PASDAC*

杉　浦　宜　紀**

Abstract

This paper describes PASDAC, a system for automatic production of data communication system programs.

In PASDAC, the skeleton of data communication processing functions is considered as a train of consecutive file processing operations; and the system is put into a model in accordance with the attributes of each file processing, connective relations among file processing operations, operation conditions and so forth.

Based on this concept, the user can obtain programs of a target system, with a little input, by principally describing the data flow and the file format in a nonprocedural manner.

The number of input steps needed to obtain a desired program is one percent to several percent of that needed in the case of description in assembler; and the memory required is roughly the same or 2 percent larger, at most.

1. まえがき

ソフトウェアの生産性向上，高信頼度化等を目的として，各方面でプログラム作成の自動化に関する研究が行われている[1)~3)]．それらの多くは，機能別に細分化された多数のモジュールをファイル化し，必要に応じて結合，編集して新しいプログラムを生成するもので，コーディング，デバッグに要する労力の削減と，プログラムの信頼性向上に大きな効果が期待されている．このようなシステムにおいては，対象とする分野のシステムをいかにモデル化し，必要な機能を抽出し，分割してファイル化するか，また入力言語をどのようなものにするかが重要な問題となる．機能の抽出，分割を誤ると，類似機能の重複を招き，ファイルはぼう大なものになりかねないし，またモジュールの数にくらべて適用範囲がせまく，融通性の乏しいものになる可能性がある．記述言語は，できるだけ少ないステップ数でシステム・モデルが記述でき，モデルの把握が容易なこと，および，記述順序の関係しない，非手続的言語であることが，誤り率，プログラム作成時間，デバッグ効率の点から望ましい．

本論文では，主として，データ通信網における集中局，中継局あるいは端末局として用いられる比較的小規模なデータ通信システムのプログラムを自動作成するためのシステムとして開発したPASDAC(Program Automation System for Data Communication Systems)について述べる．PASDACでは，まず，データ通信の処理機能の骨格を，継起するファイル(ここでは理論的に一つのかたまりとなった情報のことをすべてファイルと称する)処理の列としてとらえ，各ファイル処理の属性，ファイル処理間の接続関係，動作条件などによりシステムをモデル化する．使用者はこのような概念にもとづいて，主としてデータの流れと，ファイル形式を非手続的に記述することにより，わずかの入力でプログラムを得ることができる．

本システムは，ミニコンピュータ OKITAC-4300を用いるデータ通信システムに適用され，対象システム作成に要するステップ数は，アセンブラで記述する場合の100～数10分の1，できあがったシステムの所要メモリ量は，ほぼ同等か，多くとも20%増程度である．

* Program Automation System for Data Communication Systems: PASDAC by Nobunori SUGIURA (Software Systems Division, Oki Electric Industry Co., Ltd.).

** 沖電気工業(株) ソフトウェア事業部

この論文では，PASDAC のモデル化の概念，システム構成，入力言語，適用例等について述べる．

2. モデル化の概念

データ通信システムのソフトウェアを，オペレーティング・システム（OS）部分と，アプリケーション部分にわけ，アプリケーション・プログラムを OS の管理下で動作するジョブとみなす．

アプリケーション・プログラムはシステムに要求される機能をハードウェアおよび OS に適合させて作られるものであり，その構成は次のような概念でとらえることができる．

アプリケーション・プログラム
- アプリケーション機能
- 手続
  - アプリケーション内手続
  - OS インタフェース手続
  - ハードウェア・インタフェース手続

アプリケーション機能は，システムに要求される機能仕様であり，手続は，その機能を実現するための手段に相当する．手続は，アプリケーション自体の内部における順序とか，実現手段を与えるものと，OS とのインタフェースをとるもの，ハードウェアとのインタフェースをとるものからなっていると考えられる．アプリケーション機能は，通常，自然言語によって仕様書に記述され，それをもとにプログラム作成者が手続を加えてプログラムに仕上げる．従来，プログラム用言語は，アプリケーション機能と手続を区別して記述できるように作られていないため，両者が一体になった形で記述が行われている．プログラム・ミスは，主にこのような両者一体になった記述に起因することが多い．システムの使用者にとっては，アプリケーション機能だけが必要であり，その機能がどのような手続によって実現されているかは問題ではない．そこで，アプリケーション機能のみを記述することにより，自動的にそれに必要とされる手続をつけ加えて，プログラムを生成するシステムが実現できれば，プログラム作成上の誤りは取り除かれる．

PASDAC はこのような考えにもとづいて作成されたもので，入力言語は，データ通信システムのアプリケーション機能のみを記述するように作られており，一切の手続は自動的に生成され，プログラムが作成される．

アプリケーション機能は，次のような要素によって表現される．

（1） ファイル

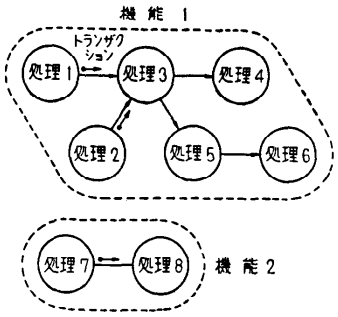


Fig. 1 Graphic expression of functions of a data communication system.

論理的にまとまった情報のかたまりを収容するもので，場所はどんな装置上，あるいは回線上にあってもよい．

（2） 処 理

ファイル内の情報を他のファイルに転送するもので，転送するに際して情報を加工する場合もある．

（3） 機 能

継起する一連の処理列を機能とよび，**Fig. 1** のようなグラフで表現する．機能には，始点および終点となる処理があるものとする．対象システムの全機能は，このようなグラフの集合によって表現される．

（4） トランザクション

各々の機能は，その始点となる処理が，何らかの原因によって起動され，順々に次の処理にうつって行くという過程ではたされる．このような過程は，あたかもある粒子が始点で発生し，連続する処理の経路を流れて行くように考えることができる．このような仮想的な粒子をトランザクションと呼ぶ．

（5） システム状態

システムを構成するハードウェアの状態，あるいは処理の進み具合など，アプリケーション機能を記述するうえで必要な状態をシステム状態という．

対象システムのアプリケーション機能は，以上5種類の要素によって記述される．

トランザクションは，通信回線からのデータ入力，周辺端末からの割込等の外部励起，あるいは，その他のシステム状態の変化により発生し，機能の始点となる処理を起動する．1つの処理が終了すると，次に進むべき収理が動作可能の場合は，トランザクションは次の処理に進む．動作不可能の場合は，可能になるまで待たされる．次に進むべき処理が2つ以上あるときは，トランザクションはその数だけ複製され，それぞ

れの処理に伝送される．次に進むべき処理が2つ以上あり，システム状態によって，そのうちのどれかを選択してトランザクションが伝送される場合もある．また，互いに別の経路にある処理として記述されているものでも，同期して動作する必要のある場合がある．それらは同期処理として記述され，トランザクションは互いに同期をとりながら伝送される．機能の終点にあたる処理が終了した時点で，トランザクションは消滅する．一般には，トランザクションはシステム中に複数個存在し，複数個の処理が並行して行われる．システム状態は，トランザクションが移動する時変化する．

処理の基本的な操作は，ファイル情報の移動であり，次の4つを標準操作としている．

（i） GET： ファイルから1レコードのデータを読み込む

（ii） PUT： ファイルへ1レコードのデータを書込む

（iii） CHANGE： ファイルの，あるレコードの内容を変更する

（iv） DELETE： ファイルの，あるレコードを消去する

レコードの内容を変更する場合の変更関数は，処理の結果作られるファイルのレコードに対する属性として記述され，四則演算，論理演算等が関数として記述できるが，それ以外の変更を行う場合はその内容を外部サブルーチンとして記述し，PASDAC にリンクすることができる．

3. 入力言語

3.1 システム・モデル

PASDAC では，システム・モデルは次のような要素によって表現される．

$$SF=\langle H, S, S_0, F, P, T\rangle$$

ここで，各要素は次のような意味をもつ

H: ハードウェア構成

S: システムの状態変数集合

S_0: 状態変数の初期値集合

F: ファイル集合

P: 処理集合

T: トランザクション集合

3.2 ハードウェア構成の定義

システムのハードウェア構成は，次のような要素によって定義される．

$$H=\{(d, n)_i\}\quad (i=1, 2, \cdots, m_h)$$

d は装置の名称，n は台数，m_h はシステムを構成する装置の種類（名称）の数である．

主記憶装置も一つの装置とみなし，台数はキロワード数で定義される．

回線構成は特に次のように定義される．

$$H=\{(l, n, s, k, c)_i\}\quad (i=1, 2, \cdots, m_l)$$

l は回線名，n は本数，s は回線の速度，k は単方向（送信，受信），半2重，全2重の別，c は制御方式である．m_l は回線の種類の数である．

3.3 システムの状態変数の定義

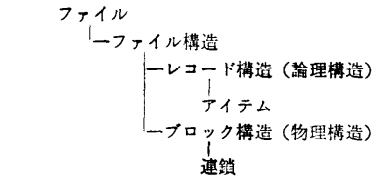
システムの状態変数は，次のように定義される．

$$S=\{(s, D, s_0)_i\}\quad (i=1, 2, \cdots, m_s)$$

s は状態変数名，D は状態変数の領域，s_0 は状態変数の初期値を表わす．m_0 は状態変数の数である．状態変数には，ハードウェアの状態を示すものと，使用者が記述上，必要に応じて設定するものとがある．前者の状態変数名には装置名を用い，とりうる値は READY, INHIBIT, BUSY, PAUSE 等である．後者は，処理の流れを制御するパラメータや，各種の判定条件など，自由に設定，使用することができる．

3.4 ファイル構造の定義

ファイルは次のような構成で定義される．



ファイルの記述は，次のような要素によりなされる．

$$F=\{(f, d, a, u, fs)_i\}\quad (i=1, 2, \cdots, m_f)$$

f はファイル名，d は装置名，a はそのファイルが使用されるときの該装置の動作（READ, WRITE 等），u はその装置内のユニット番号，f_s はファイル構造名である．ファイル構造は次の要素から定義される．

$$F_S=\{f_s, r, r_n, r_s, b_s)_i\}\quad (i=1, 2, \cdots, m_{fs})$$

r はレコード名，r_n はレコードの数，r_s はレコード構造名，b_s はブロック構造名である．

さらにファイルの論理的構造を規定するレコード構造の定義は，次のような要素によって行われる．

$$R_s=\{(r_s, I, I_l, S_P, F_{act})_i\}\quad (i=1, 2, \cdots, m_{rs})$$

I はアイテム名，I_l はアイテムの長さ，S_P はアイテム間の分離に用いる符号，F_{act} は変更関数である．アイテム名の先頭文字はアイテムの種類を表わす文字と

されており，固定長，可変長，その他全部で7種類の種別ができる．

変更関数は，アイテムを生成するときの関数を算術式または論理式等を用いて表わすもので，式中の要素としては，アイテム名，状態変数名，状態値，定数，文字符号などが許される．四則演算，論理演算だけでは記述できない場合は，サブルーチン名を記入することにより，使用者の作成した外部サブルーチンとリンクすることができる．

ブロック構造は，ファイルの物理的構造を規定するもので，次のように定義される．

$$B_S=\{(b_s, t, l, n, e, ch, cha)_i\} \quad (i=1, 2, \cdots, m_{b_s})$$

t はブロック構造の形式で，b 種類の形式の中から選択される．l はブロックの長さ，n はレコード数，e は空欄に入れるデータ，ch は連鎖名，cha は連鎖のアドレスである．l〜cha はブロック構造の形式により必要なもののみ指定する．

3.5 処理の定義

処理は，ファイル操作を基本とする最小単位の機能であり，処理の集合によってアプリケーション機能が表現される．処理は次のように定義される．

$$P=\{(P, P_t, EC, OP, f_r, \mathrm{Sel}, \mathrm{Sch})_i\} \quad (i=1, 2, \cdots, m_p)$$

P は処理名，P_t は処理の種類である．処理の種類は次の4種からなる．

始点処理：機能の起点となる処理．トランザクションはここで発生する．

終点処理：機能の終点となる処理．トランザクションはここで消滅する．

一般処理：機能の途中の処理で，トランザクションの発生，消滅はなく，通過するのみ．処理に対するトランザクションの入口と出口は一般には複数個ある．

同期処理：ファイル操作は行わず，異なった入口から入ったトランザクション間の同期をとるもの．トランザクションの発生，消滅はない．

E_c は処理が動作可能となる条件を与えるもので，システム状態変数を用いた論理式で与えられる．O_P は第2章で述べたファイル操作 (GET, PUT, CHANGE, DELETE のうちの1つ) を示す．これ以外の操作を行う場合は外部サブルーチンをリンクすることができる．f_r は操作をうけるファイルのファイル名，レコード名，レコードの何番目かを指定するものである．Sel は，次にトランザクションが伝送されるべき処理が複数個ある場合，そのうちのどれを選択するかを決定する論理を与えるものである．選択論理の変数には，前述の変更関数の所で述べた各要素が使用できる．Sch は，処理を終了した時点で状態を変化させるためのもので，変更条件と，変更指定よりなる．

3.6 トランザクションの定義

トランザクションは，継起する処理の中を流れる仮想的な粒子で，次のように定義される．

$$T=\{(t, P_t)_i\} \quad (i=1, 2, \cdots, m_t)$$

t はトランザクション名，P_t は，t が発生する処理名を表わす．P_t において発生したトランザクションは後続する各々の処理に示される論理によって経路を選択しながら伝送され，終点処理で消滅する．

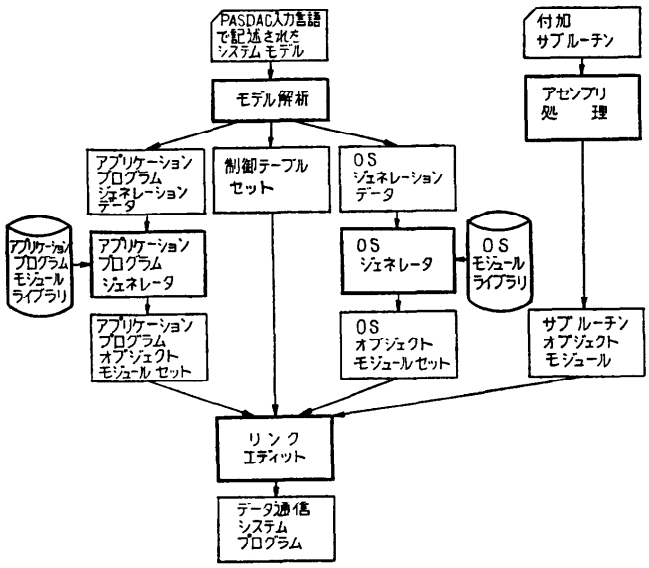


Fig. 2 PASDAC process flow.

4. システム構成とプログラム生成の流れ

PASDAC のシステム構成と，プログラム生成の流れを **Fig. 2** に示す．

図中太線で示した部分が PASDAC システムの行う処理であり，細線の部分は PASDAC により生成されるプログラムおよびテーブルである．

モデル解析プログラムは，入力言語によって記述されたシステム・モデルを解析し，最適 OS を生成するための OS ジェネレ

ーション・データ，必要なアプリケーション・プログラムを生成するためのアプリケーション・プログラムジェネレーション・データ，および，処理を制御するための制御テーブル・セットを作成する．OS ジェネレーション・データをもとに，OS ジェネレータは，OS モジュール・ライブラリの中から必要なモジュールを選択し，OS オブジェクト・モジュール・セットを作成する．同様にアプリケーション・プログラム・ジェネレータはアプリケーション・プログラム・モジュール・ライブラリの中から，必要なモジュールを選択して，アプリケーション・プログラム・オブジェクト・モジュール・セットを作成する．以上のようにして作成された制御テーブル・セットおよび2つのオブジェクト・モジュール・セットをリンク・エディットすることにより，目的とするデータ通信システム・プログラムが作成される．PASDAC に付加される外部サブルーチンは，別にアセンブリ処理をうけた後，他のオブジェクト・モジュールと一緒にリンク・エディットされる．

モデル解析プログラムは，アセンブラで約 15 kW (16 bit/W)．アプリケーション・プログラム・ライブラリは全体で約 8 kW である．

5. 生成されるプログラムの構成と動作

PASDAC によって生成されるプログラムの構成を **Fig. 3** に示す．OSはスーパバイザ，各種管理プログラム，端末制御プログラム，通信制御プログラム等からなる．アプリケーション・プログラムは，始点処理，終点処理，一般処理，同期処理の4つの処理プログラムの下にサブルーチン群が結合された形で構成されている．各処理は，モデル解析により生成された処理制御テーブル，および，トランザクション制御テーブルにより制御された形で動作し，その他の補助テーブル，データテーブルを参照しながら処理を進める．したがって，生成されるプログラムの機能の大小は，主としてテーブル・セットの大小に影響し，プログラムの大きさにはさほど影響を与えない（一般にはサブルーチンの数が増減するが，それほど大きな差は出ない）．

処理制御テーブルは，入力モデルで記述された各処理に対して作成され，OS とのインタフェース情報，処理可動条件，ファイル操作，状態変更，その処理をうけているトランザクションの連鎖情報，ファイル情報，次の処理の選択法等の情報が一連のテーブル群で記憶されている．

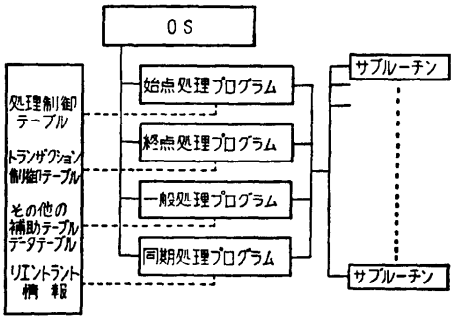


Fig. 3 Program structure of a system generated by PASDAC.

トランザクション制御テーブルは，各トランザクションに対して作成され，分岐点で複製 (split) されたトランザクションに対しても同様に作成される．このテーブルは，トランザクション番号，始点処理名，連鎖情報，同期情報，複製情報，処理後のシステム状態，処理されるデータに対する情報等からなる．

処理されるファイルの内容は，トランザクションのデータとしていったん主記憶装置内に蓄えられ，処理をうける．

各処理プログラムは，同種類の，異なる番号の処理，異なる番号のトランザクションに対して，同じ処理プログラムがリエントラントに動作し，リエントラント情報は各トランザクションに対して蓄えられる．

このような各種プログラム・テーブルにより，OS インタフェース，バッファ管理，待ち行列管理，同期管理，状態管理，スケジューリング等が自動的に行われ，また，システムの初期設定，システム状態の出力，管理情報の出力，障害対策，復帰等の機能も自動的に付加される．

6. モデル化の例

データ通信システムを PASDAC によって記述した場合の簡単な例を **Fig. 4**(次頁参照)，**Fig. 5** (537頁参照) に示す．

このシステムは，回線 (LS 0 1) (Fig. 4 (b)) から受信したデータをいったん磁気テープ(MT) に書き込み，異なった通信制御手順の他の回線 (LS 0 2) (Fig. 4 (c)) に送信するデータ中継システムで，コンソール・タイプライタ(TYP)への各種メッセージの印字，テキスト通数の管理等を行うものである．

システムの機能は，次のような要素からなる．

(1) システム始動機能

TYP からのシステム開始コマンド入力により，各カウンタをクリアし，開始メッセージを印字して通信可能状態とする．

（2） 受 信 機 能

LS 01 から受信したブロックを MT に書き込む．ブロックの末尾が ETB ならば次のブロックを受信する．ETX または EOT ならばテキスト数，CNRE，総テキスト数 TCNR を更新する．ETX ならば次のテキストを受信し，EOT ならば受信終了メッセージとともに受信したテキスト数 CNRE を印字した後，CNRE をクリアする．

（3） 送 信 機 能

TYP からの送信コマンド入力により，磁気テープからブロックを読み込み，回線 LS 02 に送出する．ブロックの末尾が ETB ならば次のブロックを読み込む．ETX または EOT ならば，テキスト数 CNSE，総テキスト数 TCNS を更新する．ETX ならば次のテキストを読み込み，EOT ならば送信終了メッセージとともに送信したテキスト数 CNSE を印字し，CNSE をクリアする．

（4） システム停止機能

TYP からのシステム停止コマンド入力により終了メッセージと総送受信テキスト数 TCNS, TCNR を印字し，システムを停止状態にする．

これらの機能を，PASDAC の入力モデル作成に便利なように図式表現したのが Fig. 4 である．

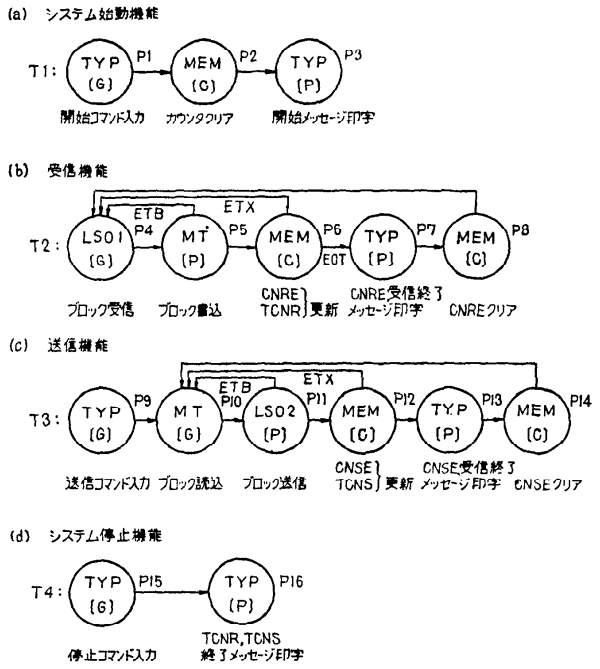


Fig. 4 Functions of an example model.

このシステムを PASDAC の入力言語で記述した場合の例を Fig. 5 に示す．図中 # 印のある項目は PASDAC のもつ標準的な状態，構造等を表わす．本システムのプログラムは，アセンブラで記述した場合，モニタ，通信制御処理，各種誤り制御，故障対策処理を含めて約 6 キロステップ程度となるが，PASDAC では，入力ステップ数は約 60 ステップである．また，所要メモリ数はアセンブラで記述した場合とほぼ同等である．

7. む す び

プログラムの自動作成システムを開発する場合，システム機能をいかに抽出し，モデル化して，プログラム構造と結びつけるかが重要な問題となる．PASDAC では，データ通信システムの基本機能を 4 種類のファイル処理としてモデル化し，そのファイル処理に各々処理プログラムを対応させ，それらを制御テーブルで制御するというプログラム構造を採用することにより入力のモデル記述が図式的に単純化され，プログラム構造との結合も容易になった．また，入力言語はアプリケーション機能のみを非手続的に記述するもので，必要入力ステップ数はきわめて少なく，記述誤り，仕様上の矛盾の発見等が容易になった．

生成されるプログラムは，機能が増大しても，テーブル類が増加するだけで，プログラムの大きさにはあまり影響がなく，メモリは有効に活用される．反面，一つの機能に対する処理が，その都度多数のテーブルを参照して行われるため，ダイナミック・ステップ数は通常の専用プログラムより増加する可能性があり，特に高速を要するシステムでは適用できない場合が考えられる．しかし，一般には，それ程高速処理を要求されるシステムは少なく，PASDAC の適用範囲は十分広い．

ファイル処理以外の複雑な処理を要するシステムについては外部サブルーチンとのリンクが可能であり，自由度は高い．

```
SYSTEM NAME: PASDAC EXAMPLE:
HARDWARE: TYP,1,MEM,20,MT,1:
LINE: LS01,TLP(1200),F(1),OCB,LS02,TLP(1200),F(1),OLV:
STATE: #STY01,#SM102,#SRL01,#SSL02,SCAL,(INHT,REDY,BUSY),INHT:
FILE: FTYP,TYP(1),FS10,FTAP,MT(1),FS20,FMEM,MEM,FS30,
      FMS1,LS01(1),FS40,FMS2,LSD2(2),FS50:
FILE STRUCTURE:
      FS10,IBEG,RS01,#BS01/IEND,RS02,#BS02/IRED,RS03,#BS03/
           ISED,RS04,#BS04/IKIN,#RS05,#BS05/IKOF,#RS06,#BS06/
           IRDR,#RS07,#BS07.
      FS20,MTRE,RS11,#BS11/MTSE,RS12,#BS12.
      FS30,CNRE,RS21,#BS21/CNSE,RS22,#BS22/
           TCNR,RS23,#BS23/TCNS,RS24,#BS24.
      FS40,TXT1,RS31,#BS31,FS50,TXT2,RS41,#BS41:
RECORD STRUCTURE:
      RS01,FBMS,C(11),,$C(SYSTEM OPEN).
      RS02,FEMS,C(50),,$C(SYSTEM CLOSE : TOTAL COUNT = ),
           (FMEM-TCNR-FRTC)+(FMEM-TCNS-FSTC).
      RS03,FEM1,C(20),,$C(RECIEVE END)+(FMEM-CNRE-FRCN).
      RS04,FEM2,C(20),,$C(SEND END)+(FMEM-CNSE-FSCN).
      RS11,UMTR,C(256),C(4),,(FMS1-TXT1-ULNR)/
           FMTC,C(2),,(FMS1-TXT1-FLNC).
      RS12,UMTS,C(256),C(4),(FMS2-TXT2-ULNS)/
           FMTC,C(2),,(FMS2-TXT2-FLNC).
      RS21,FRCN,D(10),<SCAL.EQ.BUSY>,,
           $(FMEM-CNRE-FRCN)+1$(FMEM-CNRE-FRCN)=(FMEM-CNRE-FRCN).
      RS22,FSCN,D(10),<SCAL.EQ.BUSY>,,
           $(FMEM-CNSE-FSCN)+1$(FMEM-CNSE-FSCN)=(FMEM-CNSE-FSCN).
      RS23,FRTC,D(10),<SCAL.EQ.BUSY>,,
           $(FMEM-TCNR-FRTC)+1$(FMEM-TCNR-FRTC)=(FMEM-TCNR-FRTC).
      RS24,FSTC,D(10),<SCAL.EQ.BUSY>,,
           $(FMEM-TCNS-FSTC)+1$(FMEM-TCNS-FSTC)=(FMEM-TCNS-FSTC).
      RS31,ULNR,C(256),C(4),(FTAP-MTRE-UMTR)/
           FLNC,C(2),,(FTAP-MTRE-FMTC).
      RS41,ULNS,C(256),C(4),(FTAP-MTSE-UMTS)/
           FLNC,C(2),,(FTAP-MTSE-FMTC):
PROCESS:
      P1,S,(FTYP-IKIN)(G),<$GOD>(P2).
      P2,G,(FMEM-CNRE)(C)/(FMEM-CNSE)(C)/(FMEM-TCNR)(C)/
           (FMEM-TCNS)(C),<$GOD>(P3),(SCAL=REDY).
      P3,G,(FTYP-IBEG)(P).
      P4,G,(FMS1-TXT1)(G),<$GOD>(P5).
      P5,G,(FTAP-MTRE)(P),<(FMS1-TXT1-FLNC).EQ.C(17)>(P4)/
           <(FMS1-TXT1-FLNC).NE.C(17)>(P6).
      P6,G,(FMEM-MTRE)(C),<(FMS1-TXT1-FLNC).EQ.C(03)>(P4)/
           <(FMS1-TXT1-FLNC).NE.C(03)>(P7).
      P7,G,(FTYP-IRED)(P),<$GOD>(P8).
      P8,G,(FMEM-CNRE)(C),<$GOD>(P4),(SCAL=REDY).
      P9,G,(FTYP-IRDR)(G),<$GOD>(P10).
      P10,G,(FTAP-MTSE)(G),<$GOD>(P11).
      P11,G,(FMS2-TXT2)(P),<(FTAP-MTSE-FMTC).EQ.C(17)>(P10)/
           <(FTAP-MTSE-FMTC).NE.C(17)>(P12).
      P12,G,(FMEM-CNSE)(C),<(FTAP-MTSE-FMTC).EQ.C(03)>(P10)/
           <(FTAP-MTSE-FMTC).NE.C(03)>(P13).
      P13,G,(FTYP-ISED)(P),<$GOD>(P14).
      P14,G,(FMEM-CNSE)(C),<$GOD>(P10),(SCAL=REDY).
      P15,G,(FTYP-IKOF)(G),<$GOD>(P16).
      P16,T,(FTYP-IEND)(P),<$STP>,(SCAL=INHT):
TRANSACTION: T1,P1.T2,P4.T3,P10.T4,P15:
END
```

Fig. 5 PASDAC input data list of the example model.

本論文作成にあたっては，名古屋大学教授福村晃夫博士に御懇切なる御指導，御討論をいただいた．また，PASDAC の開発に対しては，沖電気工業株式会社社長室主幹山本正隆博士，ソフトウェア事業部技術管理部生産技術開発課長椎野努博士，SE 部課長林徹也氏に多大の御協力をいただいた．ここに深く感謝申し上げる．

参 考 文 献

1) G. E. Heidorn : Automatic Programming Through Natural Language Dialogue : A Survey, IBM J. Res. Develop., Vol. 20, No. 4, pp. 302～313 (July 1976).
2) J. F. Nunamaker Jr. and B.R. Konsynski Jr. : Computer-Aided Analysis and Design of Information Systems, Comm. ACM, Vol. 19, No. 12, pp. 674～687 (1976).
3) H. Kawashima, N. Sugiura and F. Moriguchi : Switching Program System Generation, ISS '74, pp. 431/1-431/8 (1974).

(昭和52年3月16日受付)
(昭和52年9月14日再受付)