コンピュータビジョン　32－2
（１９８４．９．２０）

画像処理プロセッサＴＩＰ－３　言語及び実行制御モニタ

Image Processor TIP-3　Language and Executing Monitor

緑川博子　天満勉　日本電気（株）　C&Cシステム研究所
H. Midorikawa　T. Temma　NEC Corp. C&C Systems Res. Labs.

Abstract　This paper describes the language and program executing mechanism in the TIP-3, which consists of Image Pipelined Processor, ImPPs(VLSI chips), and an M68000. The language employs "function formula" and enables using a mixed description of ImPP and M68000 programs and a hierarachical program description of programs and program modules. Moreover, it also expresses the concurrency in the data-driven program executing mechanism. The TIP-3 executing monitor executes program fire processing, ImPP programs and M68000 programs as a multi-task, and realizes data-driven program executing.

１．　はじめに

近年，ＦＡ，ＯＡなどにおいて，画像処理への要求が高まっており，大容量のデータを高速に処理できる並列性の高いプロセッサの開発が盛んである．

筆者等は並列処理向き，逐次処理向きの二つの処理部を持った高速画像処理システムＴＩＰ-3を構築中[(1)]である．

ＴＩＰ-3での並列処理部はイメージパイプラインドプロセッサ（ＩｍＰＰ）をリング状に接続して構成され，従来のノイマン型プロセッサとは異なるデータ駆動型プロセッサであり，画像処理等，データ量が多く並列性のある処理部分を高速に行うことを目的としている．

一方，逐次処理部は，M68000(以下M68Kと呼ぶ）とその周辺制御部から成り，並列処理部で行われる一連の処理の実行制御と逐次的な処理とを行うことを目的としている．

この様に処理方式の異なった処理部を持つシステムで，アプリケーションプログラムをいかに実行させていくかが問題となる．

このため，ＴＩＰ-3でのアプリケーションプログラムの実行制御を記述するＴＩＰ-3処理言語について検討した．この検討をもとに，ＴＩＰ-3処理言語用テーブルアセンブラと，アプリケーションプログラムの実行を制御するＴＩＰ-3実行制御モニタを作成したので報告する．

２．ＴＩＰ-3のソフトウエア環境

ＴＩＰ-3は図１の様にＰＳＵ(Process Support Unit)，ＩＰＵ(Image Processing Unit)，ＤＣＵ(Display Control Unit)，ＩＭ(Image Memory)の四つの部分から成る．この内並列処理部，逐次処理部に対応するのが，ＩＰＵ，ＰＳＵである．

テーブルアセンブラ及びＴＩＰ-3実行制御モニタは，ＴＩＰ-3のソフトウエア開発環境において，図２の様に位置付けられる．

ＴＩＰ-3処理言語は，この言語自体が画像処理や，数値計算を行うわけではない．個々に開発されたM68KプログラムやＩｍＰＰプログラムをまとめ，処理順序，データの受け渡しを記述するものである．テーブルアセン

プラは，このＴＩＰ-3処理言語で記述されたファイルをソースとし，アプリケーションプログラム実行時に用いるテーブルを作成する，ＰＣ9800上に開発されたソフトである．このオブジェクトファイルを，コマンドカタログと呼んでいる．

また，ＴＩＰ-3実行制御モニタは，コマンドカタログをもとに，ＴＩＰ-3でのＰＳＵとＩＰＵにおける処理をリンクし，並列的に効率良くアプリケーションプログラム実行させるモニタで，ＰＳＵ上にある．

３．ＴＩＰ-3における処理の特徴

一般にアプリケーションプログラムは，幾つかの実行ルーチンから構成され得る．画像処理では，処理データ量が多く処理並列性が高い実行ルーチンの高速化が重要であるが，幾つかの実行ルーチンは処理並列性が小さく逐次処理に向いている．

ここで一例として，ＴＩＰ-3で，図３に示す様なある画像照合処理[(2)]を実行する場合を考えると，処理の実行を制御するために，次の四点が重要になる．

まず第一に，処理データが非常に多いもの，例えば画像処理などは，ＩＰＵで処理を行う事で，処理時間を大幅に短縮できるが，逐次的な処理は，ＰＳＵで処理を分担する方がかえって処理がはかどる場合もある．そこで，ＴＩＰ-3では，アプリケーションプログラムを幾つかの実行ルーチンに分け，逐次的なものは，ＰＳＵに分担させ，全体として効率良く処理できるようにすることが重要である．そのため逐次処理部，並列処理部の実行制御を統一的に行う必要がある．すなわち，ＩＰＵとＰＳＵの実行ルーチン間でのデータの受け渡しなどを実行制御処理モニタで行うことが必要になってくる．

第二に，アプリケーションプログラムは，場合，場合によって，画像のサイズ，処理パラメタ等が異なるので，アプリケーションプログラムを構成する各実行ルーチンのパラメタは，アプリケーション実行時に決定される

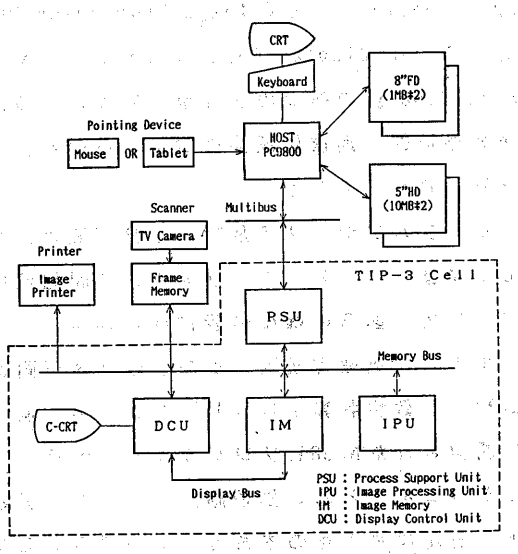

図　１　ＴＩＰ－３ハードウエア構成

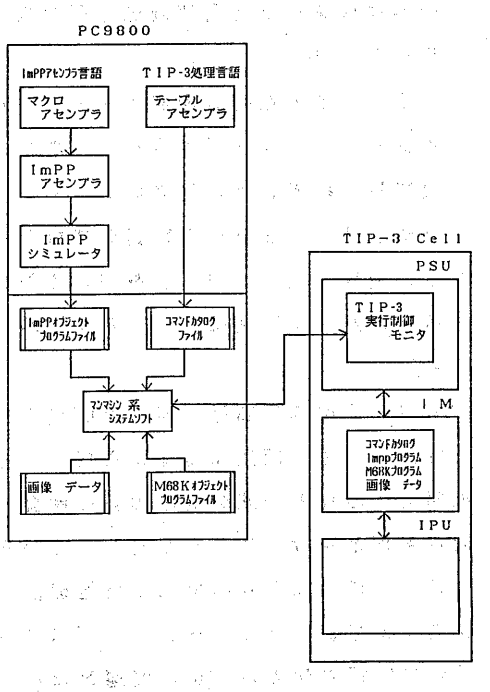

図２　ＴＩＰ－３ソフトウエア環境

ことが望まれる．こうする事により，実行ルーチンの汎用性が高められる．

第三に，すでに開発したある実行ルーチンの集まりを他の違うアプリケーションプログラムで使いたい場合，再びその同じ実行ルーチンの集まりを記述するのでは繁雑なので，使用頻度の高い実行ルーチンの集まりはサブルーチンのような形で用いる事が出来れば，都合がよい．すなわち，M68KやＩｍＰＰの実行ルーチンの集まりを一つの処理としてモジュール化し，この考えを階層的に利用できる事が望ましい．

さて，次に図２の画像照合処理の中の各実行ルーチンに注目すると，ある実行ルーチンが終了すると，他の実行ルーチンとは独立に，同時に処理を開始できる複数個の実行ルーチンがあることがわかる．たとえば，図２において，入力画像が決まれば，マスク生成（MGEN）とパッキング(PACK)はそれぞれ処理を開始することができる．また，標準図形と被照合図形のパッキング，拡大縮小，細線化，フーリエ変換，レーベリングなどの一連の処理は互いに独立で，他の実行ルーチンとは無関係に処理を始められる．この様に，一つのアプリケーションプログラムに，多くの並列処理できる実行ルーチンが含まれている．この様な実行ルーチンを並列に処理することが全体の処理を効率良く行うために重要である．従って，第四番目に処理並列性の記述ができる事が望まれる．

４．ＴＩＰ－３処理言語

４．１　ＴＩＰ-3処理言語の特徴

筆者等は，ＴＩＰ-3でのプログラム実行と処理の記述に関して必要と思われる前節の四つの点を考慮し，ＴＩＰ-3処理言語を検討し，作成した．まず第一，第三の点に関しては，ＩｍＰＰ，M68K実行ルーチン，これらの実行ルーチンを幾つかまとめたモジュールの混在記述を可能とすること．第二に関しては，実行ルーチンの可変部分を入力パラメタとして記述できること．またＩｍＰＰ実行ルーチンでは，頻繁に用いる入力パラメタはプログラム開発時にデフォルト値として設定してお

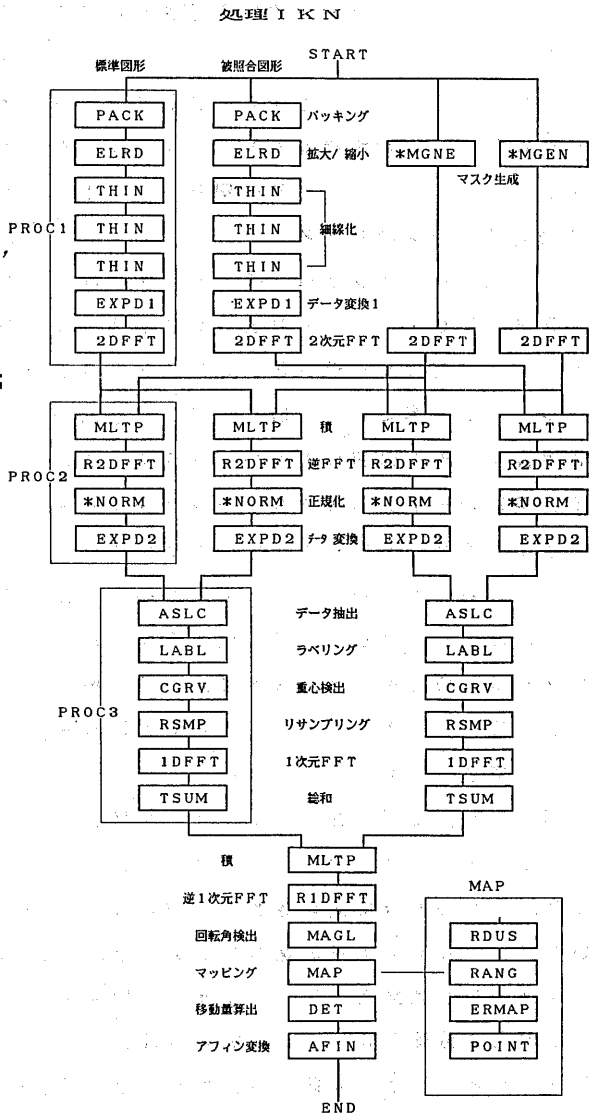

図３　画像照合処理フロー

く事により，パラメタの省略を可能とした．第四に関しては，一般に知られた関数型記述を用いることにした．この記述によって，従来の様にプログラムの記述順序で実行順序が決まるのではなく，処理に必要なデータが揃ったら処理を開始するという実行ルーチンの並列処理性を表現した．また，アプリケーションプログラムを，モジュールを使って階層化した場合にも，階層化された個々のレベルで実行ルーチンが並列的に実行される様にした．

４．２　ＴＩＰ-3処理言語の記述形式

図４（ｂ）に示す様な簡単なアプリケーションプログラムＭＡＩＮを一例として，ＴＩＰ-3処理言語を説明する．

アプリケーションプログラムＭＡＩＮは，ＩＰＰＰＲＧ１，ＩＰＰＰＲＧ２という二つのＩｍＰＰ実行ルーチンと，ＣＭＤという実行ルーチンの集まり，Ｍ68ＰＲＧというＭ68Ｋ実行ルーチンの四つの部分から成る．ここで，実行ルーチンの集まりをモジュールと呼ぶ事にすると，アプリケーションプログラムＭＡＩＮも，その中のＣＭＤも一つのモジュールとみなすことができる．ＭＡＩＮに対し，ＣＭＤは下位のモジュールと呼ぶ．言い換えれば，モジュールは，モジュールや実行ルーチンの集まりである．

図４（ａ）に示す様に，モジュールは，ＭＯＤＵＬＥ文，複数のステートメント，ＥＮＤ文の三つの部分で記述される．図４（ｂ）のモジュールＭＡＩＮの場合に対応させれば，モジュール名はＭＡＩＮ，モジュールＭＡＩＮの出力変数はＥＯＵＴ，入力変数はＰ１，Ｐ２，Ｐ３，Ｐ４，同期フラグはＳＴである．入力変数と同期フラグの違いは前者はデータ値そのものを用い，後者はプログラムの実行のタイミングをとるためだけの入力パラメタである．このＭＡＩＮが実行可能になるには，すべての入力変数，同期フラグが揃う必要がある．

ステートメントは左辺に出力変数，右辺に入力変数とファンクションを書いて等号で結んだ関数型で表される．これは，ＭＡＩＮというモジュールを構成する実行ルーチンや下位モジュールを記述するためのものである．ファンクション名には，ＩｍＰＰプログラム名，Ｍ68Ｋプログラム名，下位モジュール名の３種が用いられる．ファンクションが下位モジュールの場合には，さらに，これらは別途，モジュールとして記述されている必要がある．また，これらのモジュールに更に下位モジュールを含むこともできる．

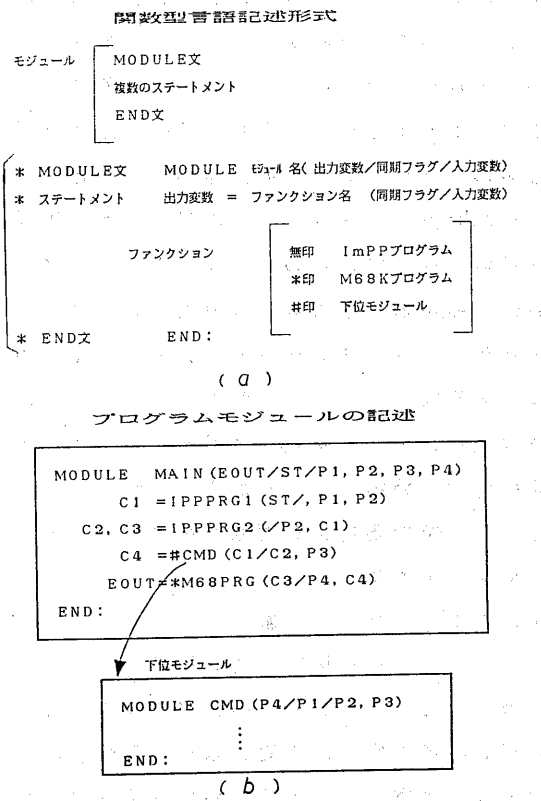

図　４　ＴＩＰ－３処理言語

５．ＴＩＰ－３実行制御モニタ

５．１　プログラム実行手順

前章で述べた実行ルーチンの並列実行を実現するため，図５の様な，ＴＩＰ-3実行制御モニタを設計した．これは，Ｍ68Ｋのリアルタイム処理用ＯＳとして発表されているＲＭＳ68Ｋの下に，五つのタスクがある様なマルチタスク構成である．この中の，実行ルーチンの処理に関連したコマンド管理，ＩｍＰＰプログラム実行管理，Ｍ68Ｋプログラム実行管理の部分は，次節５．２で説明する．

まずここで一般のＴＩＰ-3におけるアプリケーションプログラムの実行手順を示す．

（１）　スタートアップ

ＰＣ9800のシステムソフトを起動後，Ｍ68ＫのＯＳであるＲＭＳ68Ｋと前述の五つのタスクプログラム群（ＴＩＰ-3実行制御モニタ）をＭ68Ｋ側へ転送する．転送終了後，ＲＭＳ68Ｋに起動をかける．

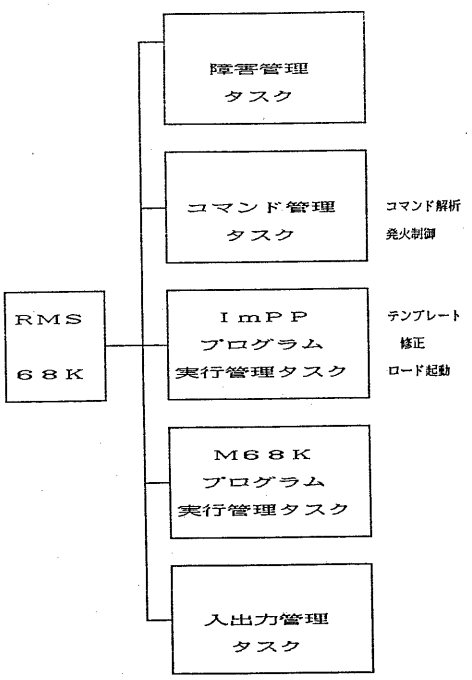

図５　ＴＩＰ－３実行制御モニタ

（２）ファイルの転送

コマンドカタログ，ＩｍＰＰオブジェクトプログラム，Ｍ68Ｋオブジェクトプログラム，処理しようとする画像などをあらかじめＩＭへロードする．

（３）プログラム実行開始コマンド入力

ＰＣ9800から実行させたいアプリケーションプログラムのモジュール名と入力変数を入力し，ＰＳＵへ送る．

（４）プログラムの実行制御

ＰＳＵのＴＩＰ-3実行制御モニタは，ＰＣ9800から受け取ったモジュール名に対応するコマンドカタログを参照しながら，入力変数が揃ったものから，ファンクションを実行していく．

５．２　タスクの概要

ＴＩＰ-3実行制御モニタにおいて，必要なデータが揃った実行ルーチンから処理を開始していくという（データ駆動型プログラム実行）方式がどのように実現されているかを述べる．

（１）コマンド管理タスク

＊ファイルの管理

ＰＣ9800からファイルをＩＭへロードすると，コマンド管理タスクは，これらのファイル名とタイプ（コマンドカタログ，ＩｍＰＰオブジェクトプログラム，Ｍ68Ｋオブジェクトプログラム，画像データ）を管理するテーブルを作成する．アプリケーションプログラムの実行時にこれを参照する．

＊コマンドカタログの展開，実行

ＰＣ9800よりモジュール名とそのモジュールの入力データが入力されると，コマンド管

理タスクは，その名前に対応するコマンドカタログを参照し，各ファンクション毎にパラメタ格納エリアをＰＳＵ内に確保する．これは，ファンクション間の受け渡しパラメタを一時格納するためのエリアである．

モジュールの入力データはコマンドカタログを参照することにより，どのファンクションの何番目の入力データとなるかがわかる．そこで，コマンド管理タスクは，データを該当するファンクションのパラメタ格納エリアに格納し，そのファンクションに必要なデータが揃ったかチェックする．揃った場合には，そのファンクションを実行する．すなわち，ファンクションがＩｍＰＰ実行ルーチンの場合は，ＩｍＰＰ実行管理タスクへ，Ｍ68Ｋ実行ルーチンの場合は，Ｍ68Ｋ実行管理タスクへプログラム名とパラメタ格納エリアの先頭アドレスを渡す．ファンクションがモジュールの場合には該当するコマンドカタログをさらに展開し，そのモジュールの入力データとして，データを渡す．

ファンクションの処理が終了すると，処理結果がコマンド管理タスクに戻されてくる．ファンクションがＩｍＰＰの場合にはＩＰＵからのハードウエア割り込み，Ｍ68Ｋプログラム，下位モジュールの場合には，ＰＳＵ内のソフトウエア割り込みとして，コマンド管理タスクへ処理結果が渡される．コマンド管理タスクは，このデータがどのカタログの何番目のパラメタなのか，予めファンクション実行開始時にテーブルを作成して管理している．

あるファンクションから出力されたデータのパラメタ番号がわかると，このデータを入力とする次のファンクションを見つけ，そのファンクションの実行に必要なパラメタが揃ったかチェックする．また値の受け渡しが必要な入力パラメタについては，パラメタ格納エリアに格納する．

この様に，各ファンクションはパラメタが揃うと次々に実行され，プログラムの出力が戻ってくる度に，データの受け渡しが行われる．

（２）ＩｍＰＰプログラム実行管理タスク

ＩＭにあるＩｍＰＰオブジェクトプログラム中の可変パラメタ部分を，与えられたパラメタ（パラメタ格納エリア中のデータ）に置き換え，ＩＰＵにＩｍＰＰオブジェクトプログラムをロードし，起動をかけるタスクである．

ＩｍＰＰ実行ルーチン終了後の出力結果データはＩＰＵから直接コマンド管理タスクへ

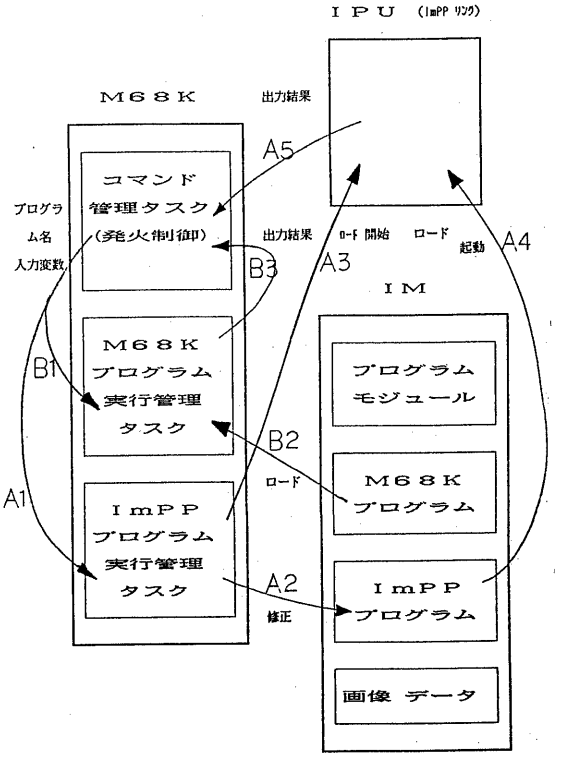

図６ ＴＩＰ－３実行制御モニタの動作

返される．

（３）M68Kプログラム実行管理タスク

M68K実行ルーチンを実行するためのタスクで，IMからM68Kオブジェクトプログラムをロードし，パラメタ格納エリアにあるパラメタを使って処理を行い，出力結果をコマンド管理タスクに返す．

図６にこの様子を示す．図中のＡ１～Ａ５，Ｂ１～Ｂ３の２種の矢印は，各々ＩＰＵ，ＰＳＵにおけるＩｍＰＰ実行ルーチンとM68K実行ルーチンの処理順序を示す．ＩｍＰＰプログラム実行管理，M68Kプログラム実行管理，コマンド管理の各タスクが，それぞれ独立に，また，マルチタスクとして並列的に実行される様子を示している．

６．言語の記述性についての検討

ＴＩＰ-3処理言語を用いて，実際の処理がどの程度記述できるか検討してみた．

図７は前述の図３の画像照合処理を記述した例である．

この例では，画像処理（細線化，パッキング，ＦＦＴ，レーベリング，リサンプリングなど）がＩＰＵで，マスク生成，正規化などの割り算を含む数値計算がＰＳＵで処理され

ＩＫＮ

```
MODULE  IKN(DX,DY,TH,P//SSA,SIA,DSA,DIA,DM1A,DM2A,R1,R2,R3,
        TSAB,TIAB,TBL1,TBL2,TBL3,FMS,FM1,FM1,FM2,GSM1,GSM2,
        GIM1,GIM2,GRVTS,GRVTI,DFMS,DFMI,DFM1,DFM2,RSSP,RISP,
        F1DS,F1DI,MAP1,MTH,ISI,RIS,NOR1,NOR2,SEL1,SEL2,L1,H1,
        L2,H2,CX,CY,PINT,M,N,DM,DN,NN1,NN2,STH,ERMP)
!
T11= #PROC1(/SSA,M,N,DSA,DM,DN,TBL1,TBL2,TBL3,
                TSAB,NN1,NN2,FMS,SEL1)
T12= #PROC1(/SIA,M,N,DIA,DM,DN,TBL1,TBL2,TBL3,
                TIAB,NN1,NN2,FM1,SEL1)
!
T3= *MGEN(/R1,NOR1,M,N,DM1A)                MASK GENERATE
T4= *MGEN(/R2,NOR1,M,N,DM2A)                MASK GENERATE
T9= 2DFFT(T3/DM1A,FM1,SEL1)                 2D-FFT  MASK1
T10= 2DFFT(T4/DM2A,FM2,SEL1)                2D-FFT  MASK2
!
T25= #PROC2(T9 ,T11/FM1,FMS,FMS,GSM1,DFMS,SEL2,M,N,NOR2)
T26= #PROC2(T10,T11/FM2,FMS,FM1,GSM2,DFM1,SEL2,M,N,NOR2)
T27= #PROC2(T9, T12/FM1,FMI,FMI,GIM1,DFM1,SEL2,M,N,NOR2)
T28= #PROC2(T10,T12/FM2,FMI,FM2,GIM2,DFM2,SEL2,M,N,NOR2)
!
T39,NS= #PROC3(T25,T26/FMS,FM1,GSM1,GSM2,L1,H1,
                GRVTS,R3,RSSP,F1DS,SEL1,SSA)
T40,NI= #PROC3(T27,T28/FMI,FM1,GIM1,GIM2,L2,H2,
                GRVTI,R3,RISP,F1DI,SEL1,SIA)
!
T41= MLTP(T39,T40/SSA,SIA,ISI)              Product
T42= 1DFFT(T41/ISI,RIS,SEL2)                1D-FFT(REV)
T43= MAGL(T42/RIS,MTH,NS,NI)            an Angle of Revolution
!
T44= #RUDS (T43 /MTH,MAP1,CX,CY,NS)         a Radius of  :
T45= #RANG (T44/MAP1,STH,NS)
T46= #ERMAP(T45/MTH,STH,MAP1,ERMP)
T47= #POINT(T46/ERMP,PINT)
!
DX,DY,TH= DET (T44/MTH,PINT)                err-circle MAP
!
P= AFIN(/DSA,DIA,DX,DY,TH)
!
END:
```

ＰＲＯＣ１

```
MODULE PROC1(T11//SSA,M,N,DSA,
!
T1= PACK(/SSA,M,N,DSA)
T5= ELRD(T1/DSA,SSA,M,N,DM,DN)
TSA1= THIN(T5/SSA,TSAB,NN1,DM,
TSA2= THIN(/TSAB,TSA1,NN1,DM,D
TSAE= THIN(/TSA1,TSA2,NN2,DM,D
T7= EXPD1(/TSA2,TSAE)
T11= 2DFFT(T7/TSA2,FM,SEL)
!
END:
```

ＰＲＯＣ２

```
MODULE PROC2(T25/T9,T11/FIN1,F
!
T13= MLTP(T9 ,T11/FIN1,FIN2,BF
T17 = 2DFFT(T13/BF1,BF2,SEL2)
T21= *NORM(T17/BF2,BF1,M,N,NOR
T25= EXPD2(T21/BF1,FOUT)
!
END:
```

ＰＲＯＣ３

```
MODULE PROC3(T39,NS/T25,T26/FMS,FM1,GSM1,GSM2,L1,H1,
        GRVT,R3,RSSP,F1DS,SEL,SSA)
!
T29= ASLC(T25,T26/FMS,FM1,GSM1,L1,H1)       Area Extraction
T31= LABL(T29/GSM1,GSM2)                    Label
T33,NS= CGRV(T31/GSM2,GRVT)                 Center of Gravat
T35= RSMP(T33/GSM2,RSSP,GRVT,R3)            Resampling
T37= 1DFFT(T35/RSSP,F1DS,SEL)               1D-FFT
T39= TSUM(T37/GSM2,SSA,NS)                  SUM Total
!
END:
```

図７　ＴＩＰ－３処理言語記述例

カタログ名	サイズ (B)	使用パラメタ数	使用 ﾓｼﾞｭｰﾙ数
IKN	1388	242 (80種)	21
PROC1	120	58 (21種)	7
PROC2	276	34 (15種)	4
PROC3	132	49 (21種)	6

表１　カタログ中の使用パラメタ数とモジュール数

ている．この様に，２つの異なる種類の実行ルーチンから成るモジュールＩＫＮが統一的に記述できた．

また，実行ルーチンの可変部分が入力パラメタとして動的に決まるので，汎用性の高い実行ルーチンが作られた．表１にこれらのカタログ中のパラメタ数，モジュール数を示す．

また，処理中に何度も使われる一連の実行ルーチンの集まりをＰＲＯＣ１～ＰＲＯＣ３としてまとめてモジュール化し，上位モジュールＩＫＮではこのモジュールを用いることにより記述を簡素化できた．

さらにこのＴＩＰ-3処理言語により，どの実行ルーチンとどの実行ルーチンが並列に処理を進められるかが表現できた．

今後の追加機能としては，パラメタ群をひとまとめにした記述（例えば一つの画像についてサイズ，ベースアドレスをまとめたもの），定数の記述，繰り返し処理の記述が望まれる．

６．おわりに

このＴＩＰ-3処理言語により，並列処理向きのＩＰＵと逐次処理向きのＰＳＵという処理系の異なった実行ルーチン間のデータの受け渡しが可能になり，一つのアプリケーションプログラムをこれらの実行ルーチンの混在した形で記述できた．また，実行ルーチンの可変部分を実行時に動的に決定することにより，実行ルーチンの汎用化が図れた．また，幾つかの実行ルーチンをひとまとめにしてモジュール化し，これを下位モジュールとしてさらに上位のモジュールで用いることができるような，ＩｍＰＰ実行ルーチン，Ｍ68Ｋ実行ルーチンを最下位とした階層的なアプリケーションプログラムの記述が可能になった．

さらに，一つのアプリケーションプログラムの中に含まれる実行ルーチンの並列性に着目し，処理に必要なデータが揃った実行ルーチンから次々と実行していくデータ駆動によるプログラム実行方式を表現することができた．

また，ＴＩＰ-3実行制御モニタでは，データ一時格納，プログラム発火制御を行うコマンド管理タスク，ＩｍＰＰプログラムの可変部分に入力変数を埋め込むＩｍＰＰプログラム実行管理タスク，Ｍ68Ｋユーザプログラムを実行するＭ68Ｋプログラム実行管理タスクがマルチタスクとして，それぞれ，並列的に処理を行なう事ができる．

現在，テーブルアセンブラとＴＩＰ-3実行制御モニタの開発はほぼ終了した．今後，階層化されたアプリケーションプログラムをデータ駆動により並列実行した場合，どの程度の処理並列性が実現できるか，さらに検討を加える予定である．

最後に，本研究の機会を与えて下さった周辺機器研究部花木部長，首藤課長，日頃ご指導頂いている岩下主任，プログラム開発をして頂いたＮＳＩＳ谷口氏に感謝致します．

参考文献

(1) 森下他 ”画像処理プロセッサTIP-3 ハードウエア構成” 1984 ｺﾝﾋﾟｭｰﾀﾋﾞｼｮﾝ研資

(2) 森下他 ”部分領域マッチングによる印影パターンの位置正規化” S59 情処第28回全国大会 予稿集 2N-1 PP.967-968