普 及 課

49－S001

コンピュータ・ネットワークJIPNETの研究開発

昭 和 50 年 3 月

JIPDEC

財団法人 日本情報処理開発センター

この報告書は，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49年度に実施した「コンピュータ・ネットワークシステムの研究開発」の成果をとりまとめたものであります。

序

当財団は情報処理技術の研究開発の一環として，昭和48年度より，「コンピュータ・ネットワーク・システム」の研究開発に着手いたしました。

複数個のコンピュータを相互に連結し，ハードウェア，ソフトウェアおよびデータ・ベースを共用する，リソース・シェアリング・コンピュータ・ネットワークの開発は，現在世界的に急速に進展しつつあります。

タイムシェアリング・システムの次に来るものとして，ネットワークを通してのコンピュータの利用法は，今後の情報処理形態の一つの焦点になるものと考えられます。

我が国においては，この分野の研究開発が，欧米にくらべ，やや立ち遅れていると言われておりますが，最近になってその気運はかなり高まってまいりました。しかし，実用的なネットワーク・システムを構築するには，まだ相当の技術の蓄積と，経験が必要であると考えられます。

当財団は異なったメーカの複数台のコンピュータを持ち，このようなネットワークの研究開発を実施しうる環境をそなえていると判断いたし，昭和48年度より数カ年の計画で，当財団内の3機種を結合したリソース・シェアリング・ネットワークの建設を進めております。このネットワークは将来，外部への拡張も考慮いたしておりますが，本年度は，前年度に行なった基本設計をもとに，基本的なファシリティとしてのコンピュータ・ネットワーク・システムのインプリメンテーションを行ない，本書はシステムの中間報告として，その成果をまとめたものであります。

本報告書が，この方面に興味のある方々に広く利用され，我が国情報処理技術向上の一助として寄与できることを念願いたす次第であります。

昭和50年3月

財団法人　日本情報処理開発センター

副会長　斎　藤　　有

ま え が き

JIPNET（JIPDEC Integrated Project NETwork）はJIPDECで開発中の異機種結合，分散型コンピュータ・ネットワークである。

第1ステップとしてインハウスの3台の国産コンピュータ，ACOS 77 NEAC 700, FACOM 230—75，およびHITAC 8450と，数台の端末が，3台のノード・プロセッサ（IMP）と48KBPS通信回線を介して結合されている。

このプロジェクトは1973年に開始され，4，5年間の開発および内部での試行期間を経て，将来は外部への拡張も考慮している。

コンピュータ・ネットワークＪＩＰＮＥＴの研究開発

目　　次

1. ＪＩＰＮＥＴの概要

1. ＪＩＰＮＥＴの概要

1.1 ＪＩＰＮＥＴの目的

ＪＩＰＮＥＴは実用と実験の両目的を持つものである。

先づ実用の目的としては，

(1) 現在，設置されている異なった機種間のハードウェア，ソフトウェア，データファイル等の制約をとり除き総合したリソースを利用する。

(2) プログラムやデータファイルの互換性を標準化とは異なるアプローチにより確保する。

(3) 漢字入出力，プロッタ，図形処理装置，マーク・シート・リーダ等の特殊周辺装置を，どのＣＰＵからも共用する。

(4) １つの端末から複数のＴＳＳにアクセスする。

(5) 機種によるロードのアンバランスをなくし平均化する。

(6) コンピュータ・ネットワーク・システムの効率評価のデータを収集する。

等がある。一方実験的目的としては，

(1) 個々のシステムの結合したが故の負荷やオーバヘッドの増大度，あるいは運用上の変更の必要度等を検討する。

(2) 実用上，ユーザレベルのプロトコルとしてどの様なものが必要であるか。またそれらの処理が異なる機種間でどの程度効率よくおこなわれるか。

(3) ネットワーク利用のメリットをコスト効率という観点から定量的にとらえる。

(4) 異機種間の分散型データベースが常識的な効率下においてマネージできる方式および限界を研究する。

(5) 異機種間のオートマティックなハードおよびソフトのシェアリングを実現し，バーチャル ネットワーク（Virtual network）の実用性および有効性を検討する。

1.2 ＪＩＰＮＥＴの構成

ＪＩＰＮＥＴは３台の代表的な国産の中・大型コンピュータ，NEAC，FACOM，HITACをHOSTコンピュータとし，３台のIMP，NEAC3200／50と48KBPSの回線を介して図１－１に示す結合形態によりネットワークのモデルを形成している。

各HOSTとIMP間はセレクター・チャネルによる直結とし，インタフェース・アダプタを介し

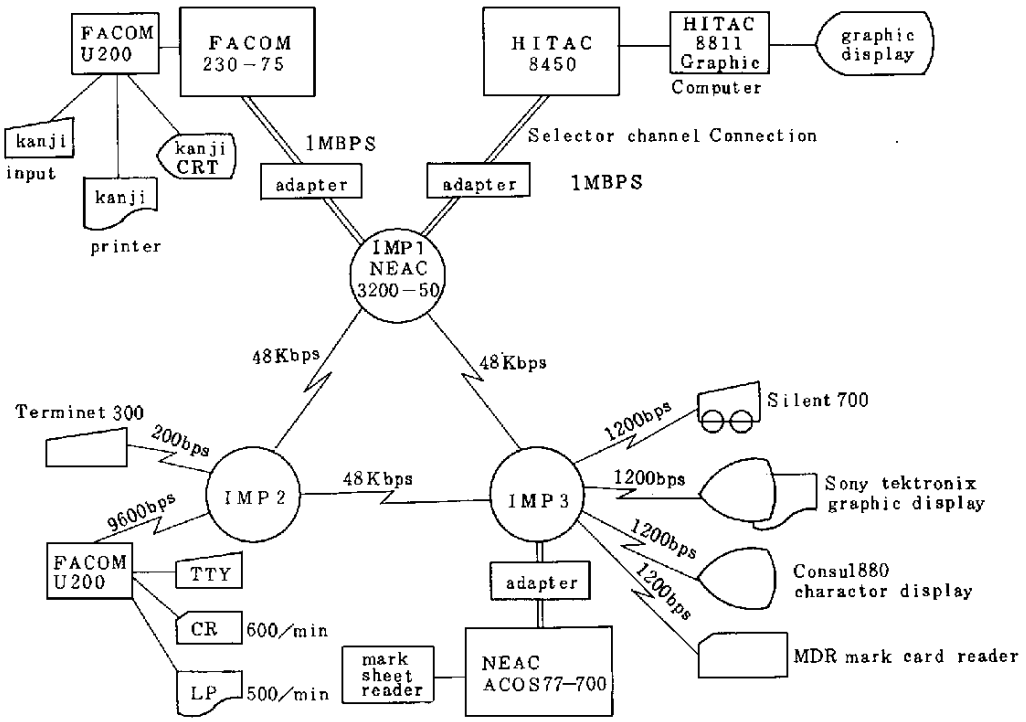


図１－１　ＪＩＰＮＥＴの構成

てキャラクター単位の８ビットのパラレル転送を半二重により行なっている。

結合にあたっては，次のような点に留意した。

(1) システム構成はサブネットのコントロール，ＩＭＰに直接結合された端末からのネットワークリソースの利用，またはネットワークの効率測定等を含むコンピュータ・ネットワークの基本機能の殆んどすべてが実験しうるものであること。

(2) コンピュータ・コンプレックス・システムの一種として，ＪＩＰＤＥＣで実施する他の研究，開発にも適した構成であること。

(3) 将来，容易に他のコンピュータ・ネットワークまたはシステムに結合できる拡張性を持つ。

(4) 情報ネットワーク・システムのモデルとして，分散型データベース・マネジメント・システムの実験が可能であること。

(5) 現在までに各ＨＯＳＴで実行されてきたすべてのプログラムが何等修正なしに引き続き稼働可能であること。

1.3 JIPNETの機能

JIPNETは次の諸機能を持つ。

(1) 他HOSTにプロセス(HOSTにおける処理の単位)を発生させ,プロセス間通信により処理を依頼する。入力データ・ファイル,プログラム・ファイル等の所在は問わない。

(2) 各HOSTのTSS端末あるいはリモート・バッチ・ステーションから他HOSTのTSS処理あるいはリモート・バッチ処理が使用できる。

(3) 漢字I/O,プロッター,マーク・シート・リーダー等の特殊I/Oをすべての HOSTで共用することができる。

(4) IMPに直接連結された会話型端末,あるいはリモート・バッチ端末より,任意のHOSTのTSSあるいはリモート・バッチ処理が使用できる。

(5) Cooperated Job Processing(CJP)

ネットワーク内のHOST・コンピュータ群を一種のコンピュータ・コンプレックスとして考え複数HOSTにわたるいくつかの作業を並行的に処理し,一つの作業を複数HOSTの協業により行なう。

(6) Automatic Job Dispatching(AJD)

AJDはネットワーク・ジョブに対する自動的なHOST選択機能であり,HOSTを指定していないジョブに対し,各HOSTの機能,負荷などを考慮し,最適のHOSTを割り当てる。特にIMPに直接連結された端末からのジョブに対して,利用者にHOSTマシーンを意識させないでも済む利点がある。この機能は仮想ネットワーク(Virtual network)指向を目ざすものである。

(7) Distributed Data Base Accessing(DDBA)

各HOSTに分散したデータ・ベースに自由にアクセスできる機能である。異機種HOST内の種々の構造のデータ・ベースの処理および互換性を可能とする。

2. プロトコル

2. プロトコル

2.1 概　要

プロトコル（Protocol）とは「議定書」を意味するが，コンピュータ・ネットワークにおいては複数のプロセス間で授受されるデータの形，タイミングおよびセマンティクス（Semantics）に関して決められた約束ごとの集合の意味を持っている。

コンピュータ・ネットワークは，HOST・コンピュータ，端末，IMPおよび回線等いくつかの異なった要素から成る複合体である。プロトコルはこれらの諸要素の間で相互の関連および分担を明確にするという役割も果たす。また多くの要求を一つのレベルのプロトコルで規定すると相互の関係が複雑になり，プロトコルの一部の追加，変更が全体に影響を及ぼす可能性があるため，基本的な機能をはたすプロトコルから，順次より高度な機能をはたすプロトコルへと積み上げる，すなわち階層的構造をプロトコルに持たせる方法がとられる。

このように階層的構造を持たせることにより，1つのレベルのプロトコルの変更が他のレベルのプロトコルに影響を与えぬ利点がある。

JIPNETにおけるプロトコルは次の4種類であり，図2－1に示す階層関係を持つ。

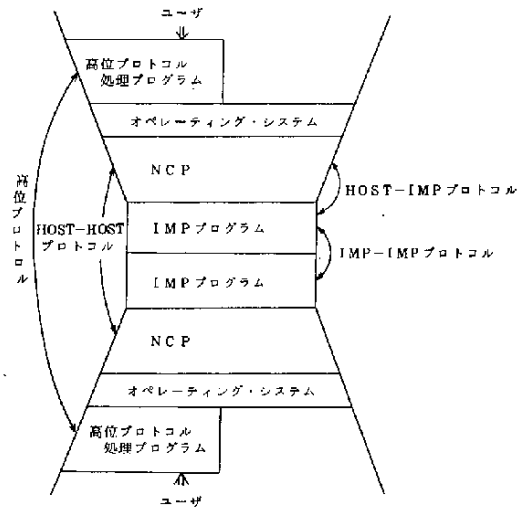


図2－1　プロトコルの関係

① IMP－IMPプロトコル

パケットの送信および受信，確認の手続き，エラー検出と回復，パケット・フォーマット等，相隣る2つのIMP間のプロトコルと，フロー・コントロール，シークエンスコントロール，メッ

セージ伝送にともなうエラー検出と回復等，発信地IMPと目的地IMP間のプロトコルとがある。

② HOST－IMPプロトコル

HOSTとIMP間のメッセージの形式や転送のコントロール，相互の障害管理等のプロトコルで主としてHOSTとIMP間のアダプターに対するコマンドの形で規定される。

③ HOST－HOSTプロトコル

HOST間のコネクションの確立，コネクションを通してのデータの送受，他HOSTのプロセスの起動とプロセス間の情報伝達，フロー制御等HOST同志のコミュニケーションに関するプロトコル。

④ 高位プロトコル

機能レベルのユーザプロトコルであり，ネットワーク内のTSS，リモート・バッチ処理等をユーザが使用する場合のコマンドを始め，ファイル転送，図形処理，特殊IOの使用等，ネットワークのすべての機能を操作性高くユーザが使用することを可能とするプロトコルである。

このうち①②③はシステムレベルのものであり，④はユーザレベルのものである。④をユーザレベルプロトコルと呼ぶ場合もある。

2.2 IMP—IMPプロトコル

2.2.1 隣接IMP間プロトコル

隣接IMP間プロトコルは，高速回線により結合された2つのIMP間でのパケットの授受，パケットの処理方法を規定したものである。

A パケットの送信および受信

隣接IMP間でのパケットの送信および受信の確認は，受信側はACKを同一回線を通じて送ることによりおこなう。送信側はACKを受け取ったパケットを捨ててもよい。

JIPNETにおいては，1本の全二重回線を，それぞれの方向ともに0番～3番までのチャネルという論理的なパスに分割し，パケットをこのチャネルに割り当てて伝送する。

4つのチャネルには，図2－2に示すように，両方向に，それぞれのチャネルの使用状態を示すチャネル・ユース語(Channel Use Word)と送信状態を記録するトランスミット語(Transmit Word)と受信状態を記録するレシーブ語(Receive Word)が対応している。

初期状態において各々のワードの対応するビットは図の様に初期化されている。即ちチャネルユース語はそのチャネルが全て未使用状態であるから，この状態を全ビットオフで表わしている。またトランスミット語，レシーブ語は双方の対応するビットが反転するように，それぞれオール0，オール1になっている。

今IMP iからIMP jに向けてパケット転送が要求されると，まずIMP iではそのチャネル

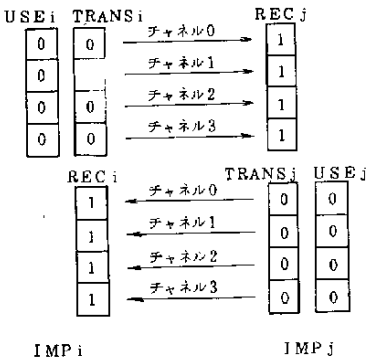


図２－２　チャネル

・ユース語USEｉを調べて空いているチャネルがあればそのチャネルを割り当て（説明ではチャネル０とする），対応するビットをオンにする。またパケット・ヘッダ中のACK語に，トランスミット語TRSｉの対応するビットを記録し，レシーブ語RECｉの全ビットと使用したチャネル番号もパケット・ヘッダに記録してそのパケットをIMPｊに向けて送りそのパケットをチャネル０に関してACK待ちの状態に入れる。

今チャネル０を使ってパケットを送出すると図２－３のような状態になる。

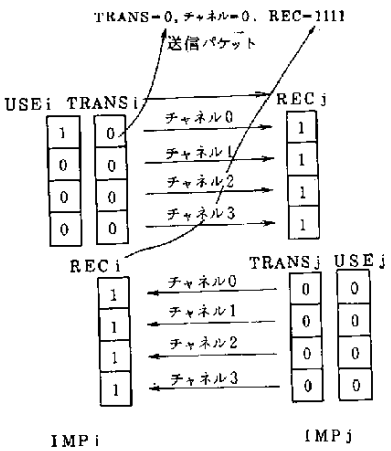


図２－３　IMPｉからチャネル０を使用してIMPｊにパケットを送信した時（パケットはIMPｊに未到着）

このパケットがIMPｊに到着するとIMPｊはまずパケット・ヘッダからTRANSの情報を取り出し，レシーブ語RECｊの対応するビットと比較する。もし両ビットが一致すれば，送られてきたパケットは，既に受信済みの重複パケットとみなし何もしないでそのパケットを棄却する。

もし両ビットが逆の関係にあればこのパケットを受け入れREC jの対応するビットを反転させる。次に，パケット・ヘッダに記録されているRECを取り出し，トランスミット語TRS jと全ビットの比較を行なう。もし一致のとれたビットがあればそのビットに対応するチャネルに関するACK信号とみてまずTRS. jのそのビットを反転させそのチャネルを通してIMP j→IMP iに送られたACK待ちのパケットを解放しまたチャネルを解放する。即ちチャネル・ユース語 USE jの対応するビットをオフにする。図2－4は前の例で，パケットがIMP jに到着した時の状態を示したものである。

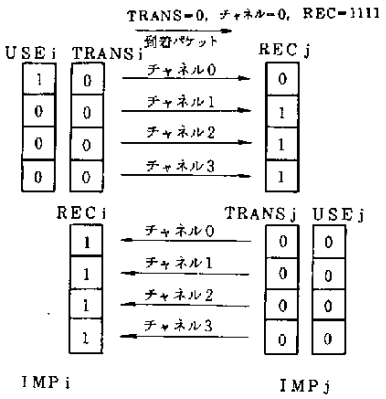


図2－4　パケットが到着した後の状態

同様にして，IMP iからチャネル1を使って次のパケットをIMP jに送った時の状態を図2－5に示す。

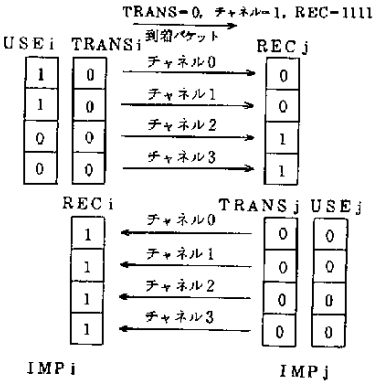


図2－5　IMP iからチャネル1を使用して次のパケットを
IMP jに送信した時

図2－6，図2－7はこの状態でIMP jからチャネル0を使ってIMP iにパケットを送った

時及び到着した時の状態を示したものである。

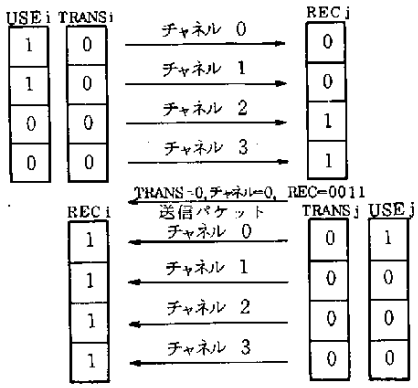


図２－６　IMP j からチャネル０を使用して IMP i にパケットを送信した時（パケットは未到着）

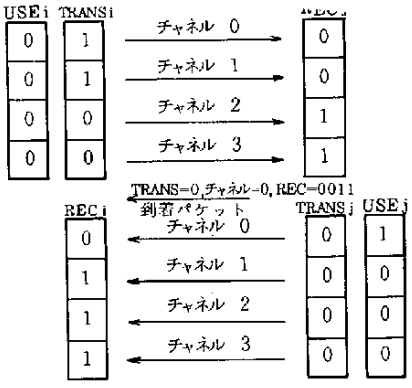


図２－７　パケットが到着した後の状態

一方的な流れしかないような場合，例えばＩＭＰｉからＩＭＰｊに向うパケットはあるが，ＩＭＰｊからＩＭＰｉに向うパケットが無い場合は，ＩＭＰｊはその回線のトラフィックを監視していて，もし，送信キューも未確認パケット（ＡＣＫを受け取っていないパケット）も無くなった場合は，ＡＣＫ情報のみをヌルパケットとして送出する。

B　エラーの検出および回復

隣接IMP間で検出するエラーは，パケットの伝送エラーおよび回線の故障（通信制御装置およびモデルも含む）である。

パケットの伝送エラーは，受信側のＩＭＰで検出されるが，この場合には受信ＩＭＰは当該パケットを捨てる。このため，送信側ＩＭＰは，パケットを送信した後，一定時間たってもACKが返ってこない場には，そのパケットを同一のチャネルを通して再送しなければならない。

回線の故障には２種類ある。すなわち，数ｍｓ間何らかの原因によって瞬断する場合と，回線が全く使用不能になる場合である。

瞬断と見なし得る場合には，回線の回復手順をふまないでただちにパケットの伝送を再開する。

回線が使用不能になった場合には，次のような回復手順をふまなければならない。（ IMP の初期手順においても，これと全く同じ手順をふむ）

(1)　通信制御装置を初期化する。

(2)　通信制御装置を送受信可能な状態とする。

(3)　Helloパケットを400ms 間隔で送信する。相手側からHelloパケットを受け取ったならば必ずI heard youパケットを同一回線を通じて送らなければならない。

(4)　I heard youパケットを80個受信した時点で回線を回復したものとみなし，Ａで述べたチ

ャネル・ユース語，トランスミット語，レシーブ語を初期化する。

(注) (3)の状態で受信したHello，I heard you以外のパケットは無視する。

C　コントロール・パケット

IMP間で伝送する単位であるパケットには，一般データを送るための通常パケットと，IMP内で制御情報を送るためのコントロール・パケットがある。隣接IMP間プロトコルとして利用されるコントロール・パケットは回線復帰パケット，回線障害パケット，Update Vectorパケット，ヌルパケット，Helloパケット，I heard youパケットである。このうち，ヌル，Hello，I heard youパケットについてはすでに述べたとおりである。

回線復帰，回線障害，Update Vectorパケットは，サブネットの状態，ルーティング情報の通知をおこなうコントロール・パケットである。

(1) 回線復帰パケット

このパケットは，自己IMPに結合されている回線が使用不能状態から使用可能状態になった時に作り出すもので，これを作り出したIMPは，その時点で使用可能状態になっているすべての回線に対して送り出す。回線復帰パケットを受け取ったIMPは，現在自分の持っているコネクション・マトリックスに対して論理和をとり，新しいコネクション・マトリックスとする。もしマトリックスが変わったならば，回線復帰パケットが送られてきた回線をのぞいて，現在使用可能状態になっている回線に対して回線復帰パケットを送らなければならない。

(2) 回線障害パケット

このパケットは自己IMPに結合されている回線が使用可能状態から使用不可能状態になった時に作り出すもので，これを作り出したIMPは，その時点で使用可能状態になっているすべての回線に対して送り出す。回線障害パケットを受け取ったIMPは，現在自分の持っているコネクション・マトリックスに対して論理積をとり，新しいコネクション・マトリックスとする。その後disconnectになっているIMPが無いかどうか調べ，もしあればそのIMPに結合されている回線は使用不能になっているものとみなす。コネクション・マトリックスが変わったならば，回線障害パケットが送られてきた回線をのぞいて，現在使用可能状態になっている回線に対して回線障害パケットを送らなければならない。

(3) Update Vectorパケット

このパケットは，Bias tableのupdateを指示するためのものであり，回線復帰，回線障害を起点として作り出される。

Update Vectorは，あるIMPから目的地IMPにパケットを送る場合に，最少どれだけの時間がかかるかを見積ったものである。

Update Vectorを受け取ったIMPは，Bias tableをupdateし，新しくUpdate Vectorを作りなおす。もしUpdate Vectorが変わったならば現在使用可能状態となってい

るすべての回線に対してUpdate Vectorパケットを送り出す。

2.2.2 発信地IMP—目的地IMPプロトコル

データ伝送においては，各階層におけるエラーの検出，回復が行なわれるが，最終的にはend—endにおいて制御することによってより確かなものにされる。IMP—IMPプロトコルにおいては発信地IMP—目的地IMPプロトコルがこのend—endの制御をすることになる。

発信地IMP—目的地IMPプロトコルは機能的に次の3種に大別される。

(1) フロー制御

サブネット内のメッセージの流れをコントロールする。

(2) シーケンス制御

目的地HOSTにメッセージを正しい順序で送り込む機能。

(3) エラーの検出と回復

IMPにおいて，これらの機能はすべてパイプを仲介として達成されている。

サブネットにおいて各IMP間は，パケットの発信側を主体としてパイプという論理的なパスで結合されている。すなわち，図2—8で示すが如く，発信地IMP—目的地IMP間に設定されたパイプにより種々の機能をはたす。

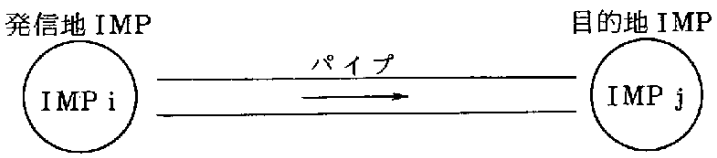


図2—8 パイプ

パイプには，1パケット・メッセージを流すためのS—pipeと，多重パケット・メッセージを流すためのL—pipeの2種類があり，S—pipeには同時に4個まで，L—pipeは同時に1個メッセージを流すことができる。

パイプを流れるメッセージは，8ビットのメッセージ番号を持っており，この番号はパイプが初期化された時に0に設定され，メッセージを流すごとに1増加してゆき，255の次は0とround—ingして使用される。

このメッセージ番号は，このパイプに関係するHOST down, RFA, allocate 8, inquiry, retransmit, give back, receive, free, inquiry—reply retransmit packetにも適用される。これらのコントロール・パケットはそれぞれのメッセージに対応するメッセージ番号を使用する。

またシークエンシングもメッセージ番号によりパイプ毎にコントロールされる。

A　基本手順

HOSTより受け取ったメッセージは，パケット化され，それぞれ独立なルートを通って目的地IMPに到着する。目的地IMPにおいてパケットはメッセージ化され，目的地Hostに送られる。目的地HOSTではメッセージを受け取ると，ただちにreceiveを返す。receiveはメッセージを発信したHOSTに送られ，メッセージの伝送が完了する。

single packet messageはパイプがあくと直ちに目的地に送られる。もし目的地がメッセージを受け取る余裕がない場合にはメッセージが捨てられ，目的地IMPは発信地IMPに向かってretransmit packetを送り，再送を要求する。これを受け取った発信地IMPでは，メッセージを再送する。このために，single packet messageは，発信地IMPにおいてcopyが持たれている。

multi packets messageは1～8 packetより成るメッセージであって，このメッセージを送る場合には，目的地IMPにおいてメッセージ組み立て用のバッファが確保されていなければならない。

メッセージ組み立て用のバッファが確保されるのは，RFA packetを送り，allocate 8 packetが返送された場合と，multi packets messageを送り，receiveを受け取った場合の2つである。このため目的地IMPにおいてはRFAを受け取って，allocate 8を返すとき，あるいはreceiveを返すのはバッファを8個確保してからでなければならない。

また，receiveを受け取ったときには，目的地IMPにおいてバッファが予約されているためにmulti packets messageを送る必要のない場合には，バッファ予約をキャンセルするためのgive back packetを送り，その確認を示すfree packetを待つ。

B　エラーの検出と回復

メッセージが回線において誤って伝送されたりするエラーは，隣接IMP間プロトコルによって規定されているので，ここでは，メッセージの紛失と，重複，目的地downについて規定する。

メッセージの重復はメッセージ番号を参照することにより簡単に知ることができる。重複メッセージは目的地IMPにおいて捨てられる。

メッセージの紛失は，発信地IMPにおいて次のようにして監視する。

メッセージを送る場合にtimerをかけ，receiveを受け取ると，timerをリセットする。もしtime outしたらinquiry packetを発信する。

inquiry packetはtime outをしたメッセージと同一のメッセージ番号を持っており，これを受け取った目的地IMPにおいては，その番号を持ったメッセージを調べ，すでに受け取っていればreceiveが紛失したものとしてreceiveを再送する。受け取っていなければメッセージが紛失したのであるから，再送要求をする。

目的地downは2種類ある。すなわち，目的地IMP down（あるいはdisconnect）と目的地HOST downである。

目的地IMP downは発信地IMPでチェックされ目的地IMPがdisconnectしたというreceiveがHOSTに送られる。

目的地HOST downは，目的地IMPで検出され，HOST downのreceiveを目的地IMPが作って送る。

以下にsingle packet，multi packets messageの伝送手順およびmulti packets messageの発信地，目的地の状態の遷移を示す。

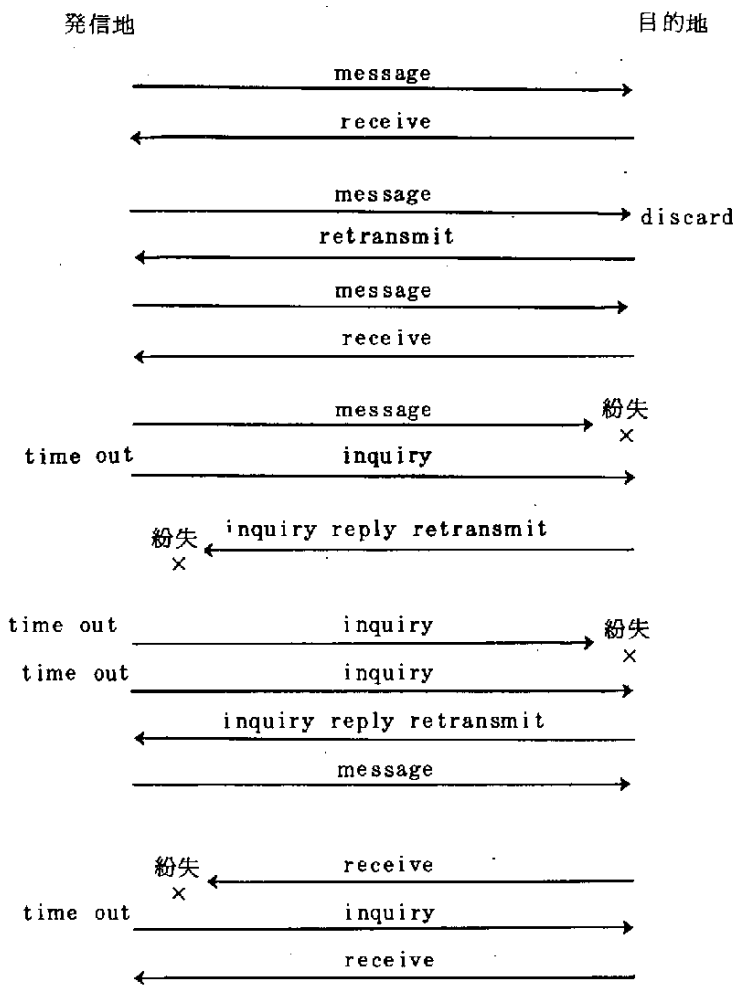


図2－9　単一パケットメッセージの伝送手順

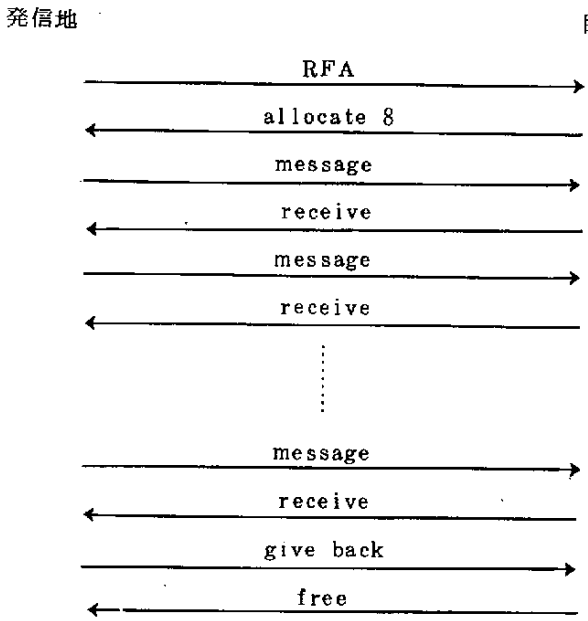


図２－10　多重パケットメッセージの伝送手順

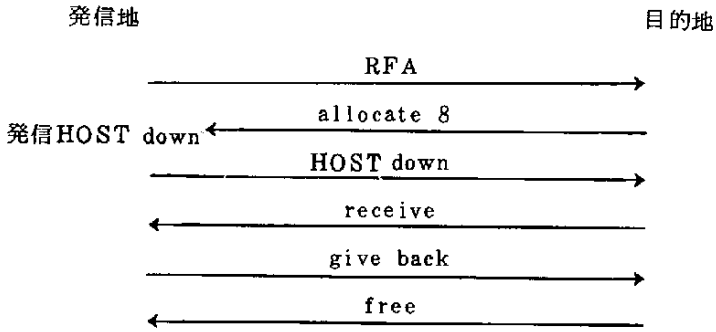


図２－11　多重パケットメッセージの伝送手順
―― 送信側HOSTがdownした場合

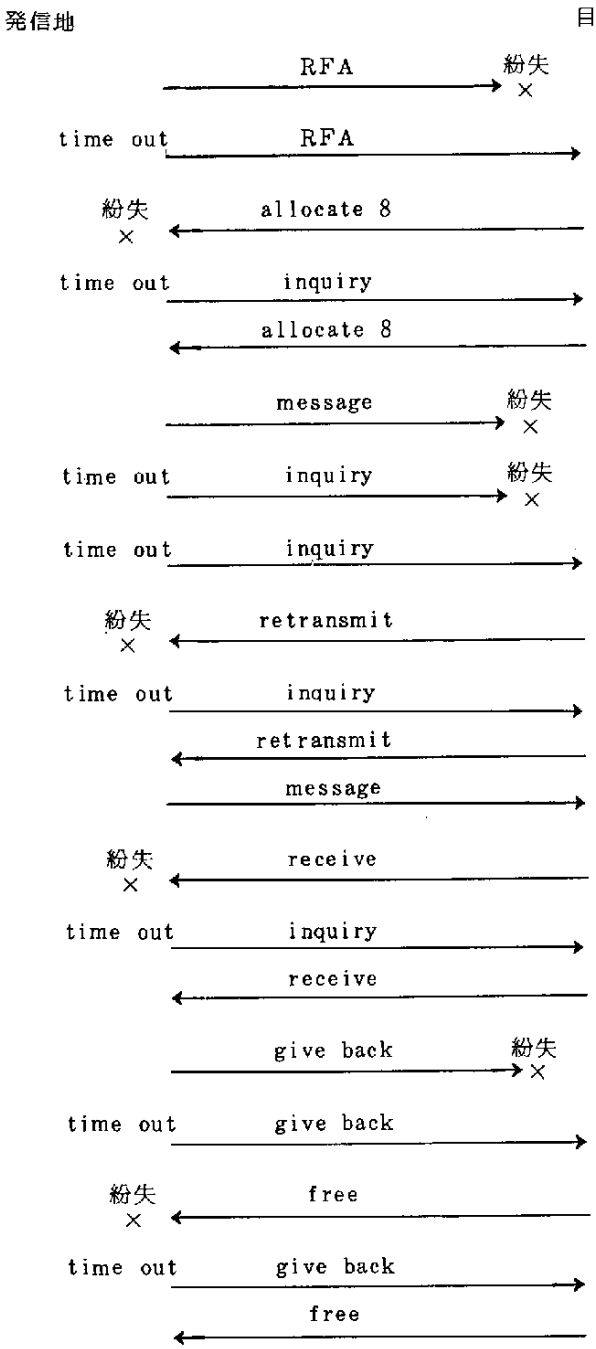


図２－12　多重パケットメッセージの伝送手順

―― メッセージ・パケットが紛失した場合

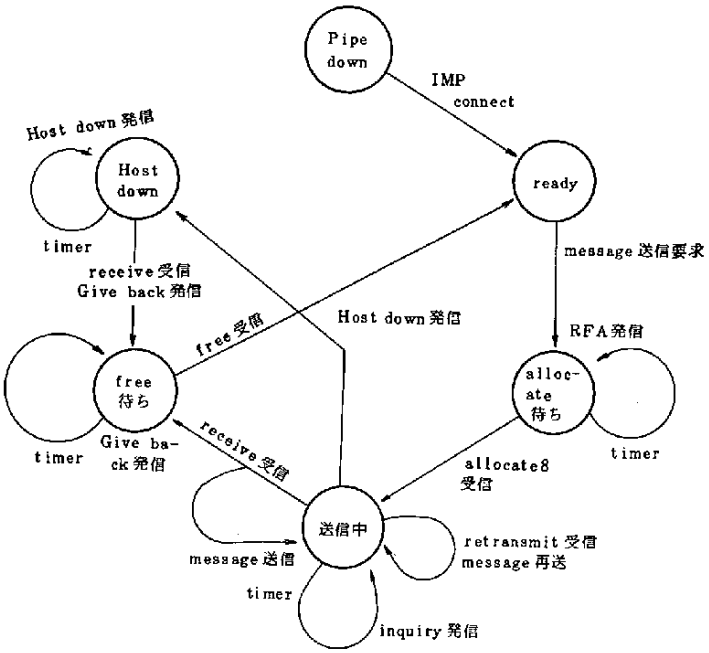


図２－１３　多重パケットメッセージの発信地のｓｔａｔｕｓ

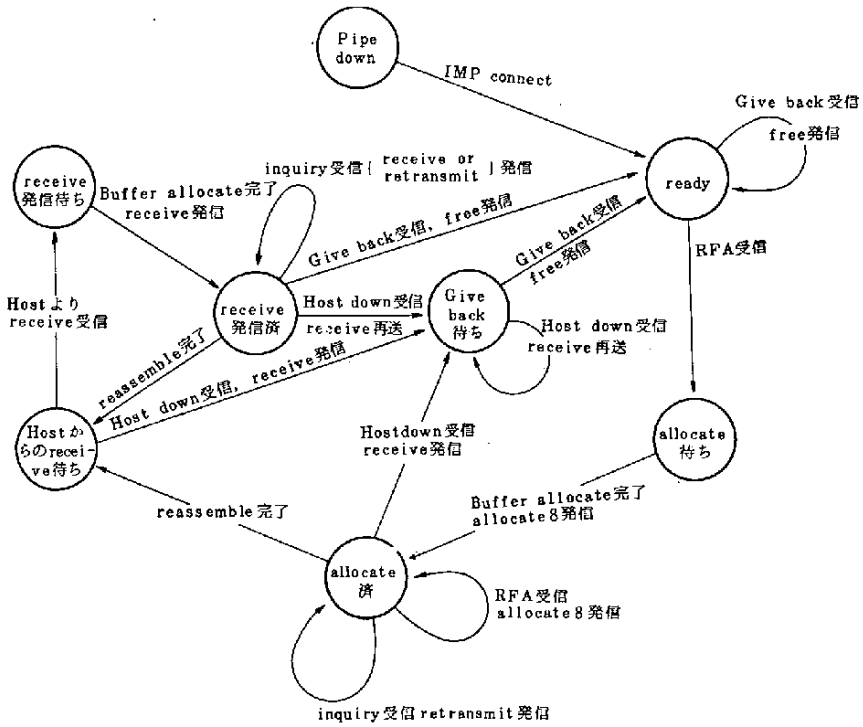
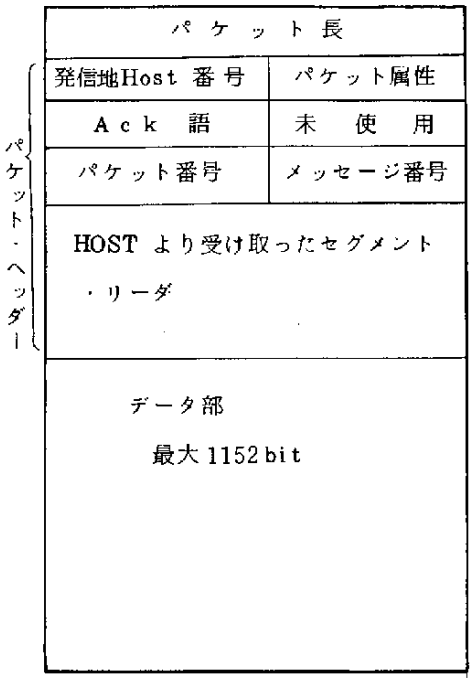


図２－１４　多重パケットメッセージの目的地のｓｔａｔｕｓ

C　IMP－IMPプロトコルに関するテーブル，パケット・フォーマット一覧表

a．パケット・フォーマット



パケット長：パケット長をバイトで示す。（発信地HOST以降，データ部までの長さである）
発信地Host 番号：このパケットを発信したHostを示す。
パケット属性：このパケットが何の目的で利用されるかを示す。詳しくはbを参照
Ack 語：隣接IMP間でAck操作をおこなうときに利用する。
第1ビット：トランスミットビット
第2～4ビット：チャネル番号
（右づめで利用）
第5～8ビット：rec 語
パケット番号：メッセージの第何番号のパケットであるかを示す。
（0～7までの値をとる）
メッセージ番号：シーケンシングのために利用する番号

図2－15　Packet format

b．パケット属性

Packet headerの第1語の下半語（9～16bit）を使用して，このパケットがどのような属性を持っているかを示す。attributeは次のものがある。

	9ビット	16ビット
通常Packet		00000001_2
Control Packet		$\times\times\times\times\times\times\times0_2$ ……××…×は種類を示す。

9ビット　16ビット

00000000 ：HOSTdown Packet
00000010 ：別の目的に使用（my line status change piece）
00000100 ：回線復帰情報Packet
00000110 ：回線障害情報Packet
00001000 ：RFA Packet
00001010 ：allocate 8 Packet
00001100 ：INQ（inquiry）Packet
00001110 ：Retransmit Packet

00010000 : Give Back Packet

00010010 : Update Vector Packet

00010100 : Receive Packet

00010110 : Null Packet

00011000 : Hello Packet

00011010 : I heard you Packet

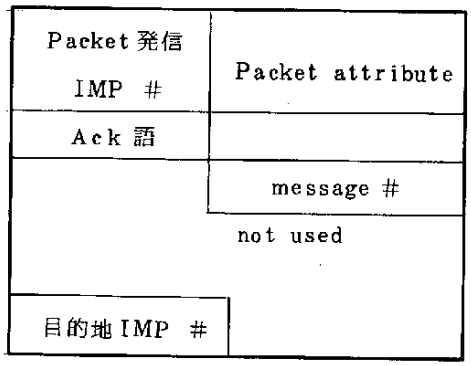
00011100 : free Packet

00100000 : inquiry reply retransmit Packet

c コントロール・パケット・フォーマット

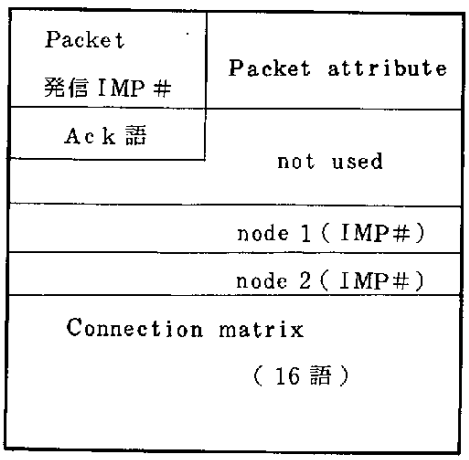
① HOST down Packet

Packet header のみでデータ部を持たない 80 ビットのパケットであり次の形をしている。



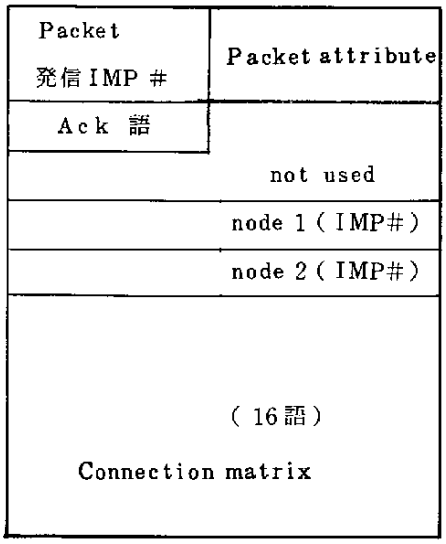
② 回線復帰情報Packet

Packet header とデータ部からなる 336 ビットのパケットでありデータ部にはコネクション・マトリックス, IMP status table を持ち次の形をしている。



③ 回線障害情報Packet

Packet header とデータ部からなる336ビットのパケットであり，データ部にはコネクション・マトリックスを持ち，次の形をしている。

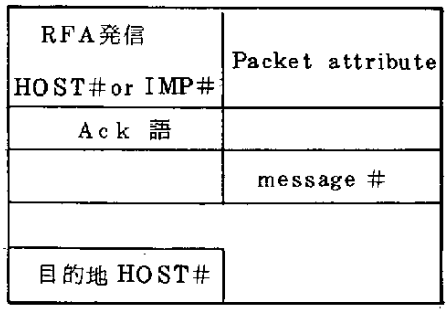


：00000110_2

これによりどのIMP間のlineかを識別する。

④ RFA Packet

Packet header のみでデータ部を持たない80ビットのパケットであり，次の形をしている。



：00001000_2

⑤ Allocate 8 Packet

Packet header のみでデータ部を持たない80ビットのパケットであり，次の形をしている。

alloc 8発信 HOST# or IMP#	Packet attribute	: 00001010_2
Ack 語		
	message #	
目的地 HOST#		

⑥ Inquiry Packet

Packet header のみでデータ部を持たない 80 ビットのパケットであり，次の形をしている。

発信HOST# or IMP#	Packet attribute	: 00001100_2
Ack 語	not used	
Pipe種類 S:0 L:1	message #	
目的地HOST#		

⑦ Retransmit Packet

Packet header のみでデータ部を持たない 80 ビットのパケットであり，次の形をしている。

Retransmit 発信HOST# or IMP#	Packet attribute	: 00001110_2
Ack 語	not used	
Pipe種類 S:0 L:1	message #	
目的地HOST#		

⑧ Give back Packet

Packet header のみでデータ部を持たない 80 ビットのパケットであり，次の形をしている。

Give Back発信 HOST# or IMP#	Packet attribute	: 00010000_2
Ack 語	not used	
not used	message #	
目的地Host#		

⑨ Update Vector Packet

Packet header とデータ部からなる336ビットのパケットであり，データ部にはUpdate Vector（そのIMPから目的地IMPに達するための最小delayを示す）を持ち，次の形をしている。

Packet 発信IMP#	Packet attribute	: 00010010_2
Ack 語		
not used		
IMP 1への最小delay		} Update Vector（16語）
⋮		
IMP 16への最小delay		

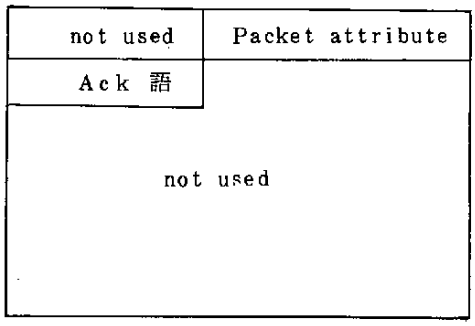
⑩ Receive Packet

Packet header のみでデータ部を持たない80ビットのパケットであり次の形をしている。

Receive 発信 HOST# or IMP#	Packet attribute	: 00010100_2
Ack 語	not used	
not used	message #	
segment leader		segment leader のHost#はmessage を発信したHost#を入れる。

⑪ null, Hello, I heard you Packet

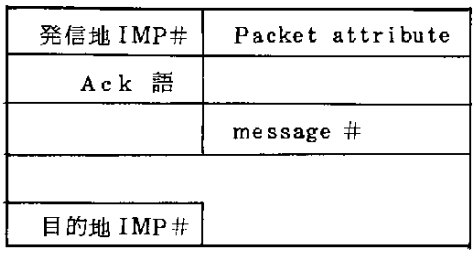
Packet header のみでデータ部を持たない 80 ビットのパケットであり，次の形をしている。



null : 00010110_2
hello : 00011000_2
I heard you : 00011010_2

⑫ free Packet

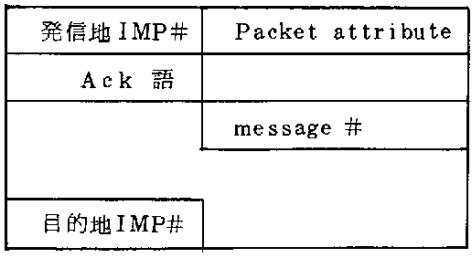
Packet header のみでデータ部を持たない 80 ビットのパケットであり，次の形をしている。



: 00011100_2

⑬ inquiry reply retransmit

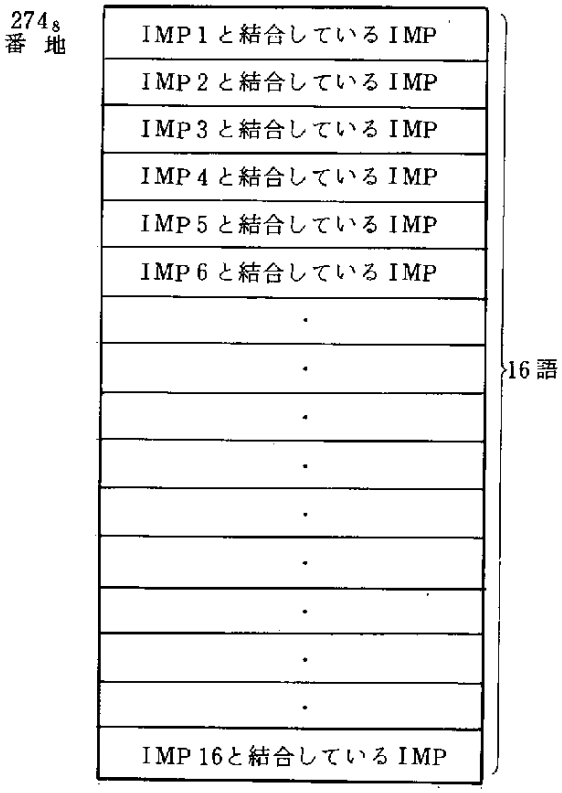
Packet header のみでデータ部を持たない 80 ビットのパケットであり，次の形をしている。



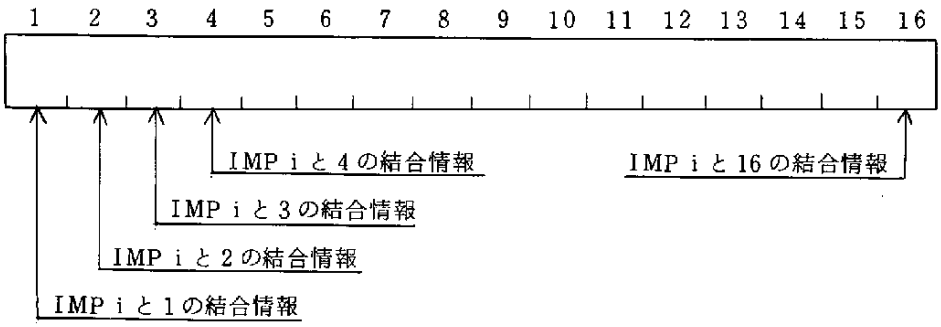
: 00100000_2

d. Connection matrix

コネクション・マトリックスは，そのIMPと結合しているIMPに対して，どのＩＭＰとどのIMPが結合しているかを示しているものであり次の形をしている。



コネクション・マトリックスの各語は次のことを示す。ここでIMP iと結合しているIMPの情報とする。



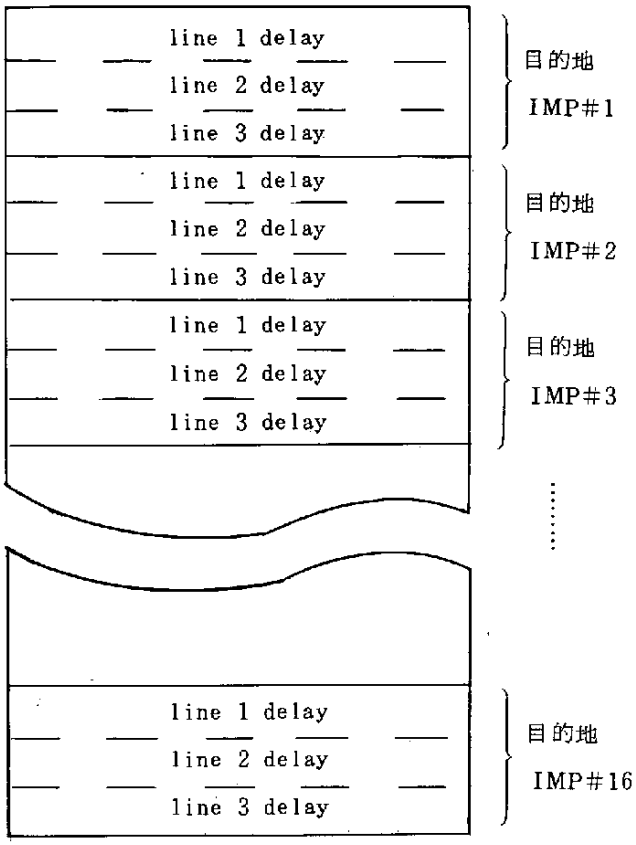
IMP i とkの結合情報は，両者が直接回線でつながっている場合に1となり，それ以外の時は0となる。

コネクション・マトリックスは初期設定ルーチンによりall 0にセットされ，それ以降はcontrolタスクがline up, downの情報により対応ビットのセット，リセットをおこなう。

e．BIAS table

Bias tableは，そのIMPから目的地IMPに対してパケットを送った場合に，途中のIMPにおいてmodem出力キューが1個と仮定したときに目的地に到着するまでにかかる時間がは

いっており，次の形をしている。



line k delayは，そのＩＭＰから目的地ＩＭＰ＃ℓにパケットを送った場合にline Kに出力したとすれば目的地ＩＭＰ＃ℓに到着するまでの予想時間がはいる。この値が20000_8のとき，そのlineがdownしているか，あるいはlineが無いかのいずれかである。

このtableのupdateはcontrol taskが次の場合におこなう。

○自己line down（およびup）pieceをmonitorよりキュー onされた場合。

○line up，line down，update vectorを他IMPから送られた場合。

2.3　HOST－IMPプロトコル

A　チャネル・アダプタのコマンド

IMPのチャネルとHOSTコンピュータのチャネルを結合するチャネル・アダプタは次のコマンドを解読，実行する機能を持っている。

これらのコマンドのコマンドコード，表現形式はそれぞれのHOSTおよびIMPにおいて違っている。

① READ DATA（RDと略称する）

相手側の発したWDI，WDコマンドによるデータを読み取る。

相手側の発したコマンドがWDIあるいはWDのいずれであるかを知ることができる。

② WRITE DATA with INTERRUPT(WDIと略称する)

相手側CPUに対して割り込みを行なう。RDコマンドが相手側がら発せられるとデータを転送する。

WDIを発する以前に相手側がRDを発していた場合には，割り込みはおこさない。

③ WRITE DATA（WDと略称する）

RDコマンドが相手側で発せられるとデータを転送する。このコマンドは相手CPUに割り込みはおこさない。

④ NO OPERATION（NOPと略称する）

何もおこなわない。

⑤ SENSE

アダプターの状態を聞くコマンドである。

⑥ LOCK PSW（LKと略称する）

自分側アダプターのプログラム・スイッチをlockする。

プログラム・スイッチは，チャネル・アダプターには2個あり，この2個がともにunlock状態のときデータを転送することができる。いずれかがlockされているときにはデータ転送をしないで異常終了する。

SENSEコマンドで聞くことのできるプログラム・スイッチは自分側，相手側のOR状態である。

電源が投入された初期状態では，このスイッチはlock状態となっている。

⑦ UNLOCK PSW（ULKと略称する）

自分側アダプターのプログラム・スイッチをunlockする。

3 データ・フォーマット

HOST－IMP間において授受されたデータは，制御情報と一般データの2種類ある。

制御情報は（図2－17）に示すように32ビットで構成され，WDIコマンドにより転送される。

一般データは初期設定によって決められたセグメントサイズ以下の不定長でありWDコマンドにより転送される。

ユーザ・プロセスが送信したデータは，メッセージごとに制御情報とメッセージを分割して1個以上のセグメントを作りサブネットに投入される。

目的地IMPにおいては到着したメッセージを送信側NCPがサブネットに投入した時と全く同

じ様式になおして受信側NCPに渡し，それを送信側プロセスが渡したメッセージと全く同じ様式にして受信側プロセスに渡す。

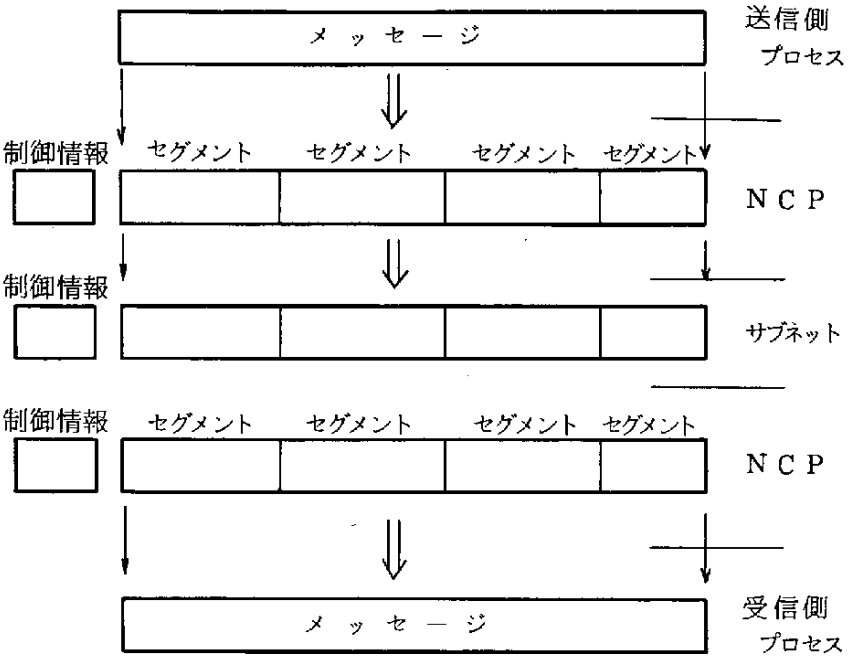


図2－16　メッセージの様式

以下に制御情報の形式を示す。

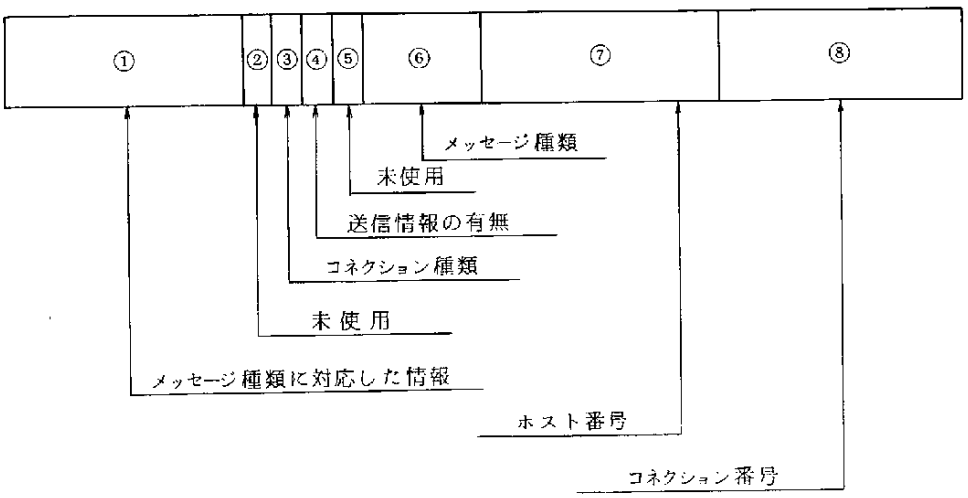


図2－17　HOST－IMP間の制御情報

① メッセージ種類に対応した情報（８ビット）

⑥のメッセージ種類により使用法が異なる。（表２－３）参照のこと。

② 未使用（１ビット）

③ コネクション種類（１ビット）

１のときこのコネクションは多重パケットメッセージを送り出すことを示す。このビットはコネクション確立時の最大メッセージサイズにより，メッセージの転送ごとにセットされる。

すなわち，最大メッセージ長が複数セグメントになる場合には，たとえ転送メッセージが１セグメントであっても１にされる。

④ 送信情報の有無（１ビット）

この情報を送ったあとに，次に相手側に送る情報の有無を示す。１のとき送信情報があるのを示す。

⑤ 未使用（１ビット）

常に０とする。

⑥ メッセージ種類（４ビット）

制御情報が何を意味するかを示す。（表２－１），（表２－２）を参照のこと。

⑦ HOST番号（８ビット）

HOST→IMP制御情報の場合には，目的地HOST番号を示す。

IMP→HOST制御情報の場合には，発信地HOST番号を示す。

⑧ コネクション番号（８ビット）

プロセス間コミュニケーションにおいてNCPがプロセス中のportを識別するための番号。

表２－１　IMP→HOST制御情報

メッセージ種類	意　味
0	通常メッセージ送信要求（Regular message） 他HOSTより，そのHOST宛の一般データを転送する要求があった時，プレフィックスとして使用される。このあと続いてWDコマンドによりメッセージ本体が転送される。
1	セグメント長変更指令 サブネットの初期設定の結果，セグメントサイズが標準値でない場合にIMP→HOSTの初期設定時に発信される。 このセグメント長変更指令はサブネットの全部が，まずwake upされ，その後HOSTがwake upされた場合以外は指示できない。この長さは576～1152ビットである。
2	初期設定指令 サブネットの初期設定の結果，セグメントサイズが標準値の場合に発信される。
3	データ受信完了（RECEIVE） 他HOSTに発信したメッセージが正常に送られたことをIMPが確認した時，発信する。
4	接続中止要求 IMPがネットワーク・サービスを中止したい時にHOSTに対して発信する。 この応答はHOST→IMP制御情報のメッセージ種類＝５により行われ，この応答により当該IMPはnot ready状態となる。 接続中止要求から切断完了までの間はIMP稼動状態にある。
5	切断完了 HOSTがネットワークサービスの中止要求を発信したのに対する応答。
6～11	未　使　用
12	複数セグメント受け入れ準備完了 HOST側から複数セグメントの転送要求があった場合に，受け入れ準備ができたら，これを発信する。これによりHOST側から第１セグメントから最後のセグメントまでの１メッセージがWDコマンドにより送られてくる。

メッセージ種類	意　　味
13	メッセージ再送要求 サブネットの障害などでパケットが消えた場合に，HOSTに対してメッセージの要求をする。これを受け取ったHost は次のタイミングでWDコマンドにより当該コネクションにおいて，RECEIVEを受け取っていないメッセージのうち 最も古いメッセージを発信しなければならない。この応答は，多重パケットメッセージの場合だけ発信される。

表２－２　HOST→IMP制御情報

メッセージ種類	意　　味
0	通常メッセージ送信要求（Regular　message） 他HOSTに一般データを送信したい場合にプレフィックスとして使用される。１セグメント・メッセージの場合には，この後続いてWDコマンドによりメッセージが転送される。複数セグメント・メッセージの場合にはＩＭＰ→HOST制御情報の複数セグメント受け入れ準備完了を受け取るとWDコマンドにより続いてメッセージを転送する。
1	初期設定完了表示（Ready）
2	未　使　用
3	データ受信完了（RECEIVE） 目的地HOSTがメッセージを受け取った時に発信する。
4	接続中止要求 HOSTがネットワークとの結合を切断したい時に発信する。 この応答はIMP→HOST制御情報のメッセージ種類＝５によって行なわれ，この応答によりHOSTは not　ready 状態となる。 この要求に先だってHOSTは必ずすべてのコネクションを解放しなければならない。
5	切断完了 ＩMPがネットワーク・サービスの中止要求を発信したのに対しての応答。
6～15	未　使　用

表２－３　制御情報のメッセージ種類に対応した情報部分の用途

メッセージ種類	メッセージ種類に対応した情報の用途
0	送信メッセージのセグメント数（１～８） 初期設定するセグメント長をユニット数（４～８）で指定する （１ユニット＝144ビット）
1	IMP→HOST制御情報の時のみ使用
2	未　使　用
3	RECEIVEの状態を数字で入れる。 ＝１の時正常受信 ＝２の時　バッファ余裕なし　discard. ＝３の時　コネクション切り換え中　discard. ＝４の時　コネクション　close discard. ＝５の時　目的地IMP　disconnect or HOST　down. ＝６の時　目的地IMPなし
4～15	未　使　用

C　HOST—IMPのデータ転送制御

HOST，IMPのそれぞれが相手側にデータを転送するタイミングは非同期に起こるが，アダプタは半二重であり，同時には一方向だけにしかデータを流すことができない。このために両側からアダプタ利用権の取り合いがおこり，これをうまく制御してやらなければならない。

現在JIPNETにおいて，実際HOST－IMPで行っているデータ転送のタイミング調整について述べる。

HOST－IMP間で転送されるデータは，基本的にはWDIによって制御情報を送ることにより，おこなわれ，WDによって送られるデータはWDIを受けた結果として送るのであるから，手順としてはWDIのみを基本として考える。

HOST－IMP間では同期をとって転送するデータがある。これは以下の通りである。

① HOSTは一般データを受信したならばただちにRECEIVEを返送する。すなわち，HOSTは一般データを受信した直後は，RECEIVE以外のものを発信してはいけないし，他のものも受信できない。

IMPは一般データを送り出したならば，RECEIVEを受ける準備をしなければならない。

② HOSTはIMPからmulti packets messageの転送要求があったならば，ただちにそのmessageを送らなければならない。

③ single packet message およびIMPからHOSTに送るmulti packets message は制御情報につづいて転送される。

上記の手順を満たしてかつコマンドの衝突をできるだけ起こさないように，次の手順をとる。ここで述べる順序は優先度の順である。

① コマンドの衝突がおこった場合には，IMPを優先し，IMP側はWDIをretryし，HOST側は受信準備をする。

② receiveを発信した場合には次の情報を発信してもよい。

③ 現在送信しようとする制御情報に，次に送るべき制御情報があるかないかのビットがあり，制御情報あるいはそれに付随するデータ送信後に相手側から次に送信する情報ありの状態になっていれば受信準備をしなければならない。

（図2－18）はその例を示している。いまIMPからHOSTに対して①～④までのメッセージがありそのうち②，④はRECEIVEである。またHOSTからIMPに対して⑪～⑭までのメッセージを送信するとしよう。

ここで①と⑪が衝突したとする（衝突がおこらなくても，①がHOST側に送られれば同じ手順をとることになる）と図中の順序で以後メッセージは衝突なく転送されることになる。

すなわち，一度衝突が起るかあるいはたがいに相手から自分側に対するデータの転送があることを知らせ合うことによって，その時点で待ち行列にあるものまたは交互に転送しあっている最中に来たものについては衝突およびデータ交換の制御をうまくおこなうことができる。

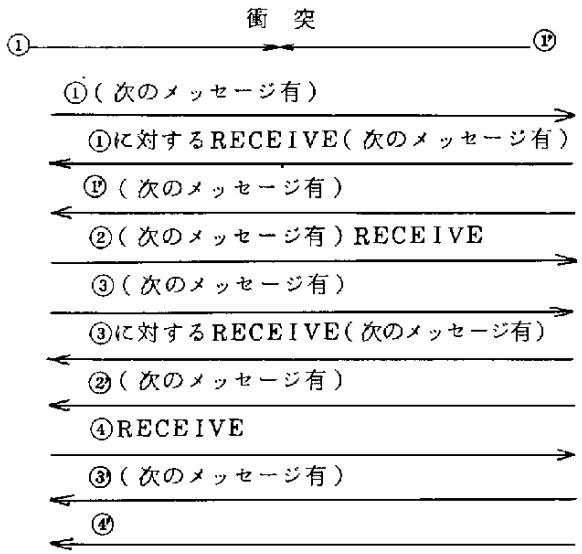


図２－18　データ転送の制御

D　初期状態処理手順

ネットワークのサービス開始時には，自分側アダプタの電源がonの状態になっているのを確認したのち，自分側のpswをunlockし相手側のアダプタの状態をきく。

相手側がinoperableであればネットワークのサービスは開始できない状態であるので，適当な時間後に始めからやりなおす。

HOST側

相手側がoperable（電源on）であれば，IMPからのactionを待つ。IMPからは，初期設定あるいはセグメント長変更指令のいずれかがくるので，それにしたがって初期設定を行ない，完了したら，初期設定完了を送る。以上でネットワークとしてサービス可能となる。

IMP側

相手側がoperableであれば両者のpswが伴にunlockになった時点で割り込みが発生するので，これをきっかけにしてIMPは次のactionをとる。

初期設定あるいはセグメント長変更指令のいずれかを送り，相手からの初期設定完了を待つ。

初期設定完了を受け取ったとき，HOSTはサブネットと結合状態になる。

E　データの正規転送手順

HOST－IMP間における情報の授受はすべてこの手順によっておこなわれる。

この手順によって転送される情報には，NCP－IMP間であらかじめ決められた制御情報とHOST中のプロセスの要求により作られる一般データである。

制御情報は32ビットで構成され，WDIコマンドにより転送される。一般データは初期状態処理手順により決定されたセグメントサイズ（これをnビットとする）によって，(n)ビットのセグメントに分割されて（最終セグメントは(n)ビット以下の不定）WDコマンドにより転送される。

制御情報には，それだけで意味をもつものと，一般データのプレフィックスとして使用されるものの2種類がある。

データの正規転送手順において，ハードウェアエラーが検出された時は，両側の処理系に知らされるのでその時点でHOSTとIMPは切りはなされる。

○一般データが1セグメントの場合

データを送信する側は，WDIコマンドにより制御情報（メッセージ種類コード＝0）を送り，これを終了したのちWDコマンドにより1セグメント転送する。これを図示すれば次のようになる。

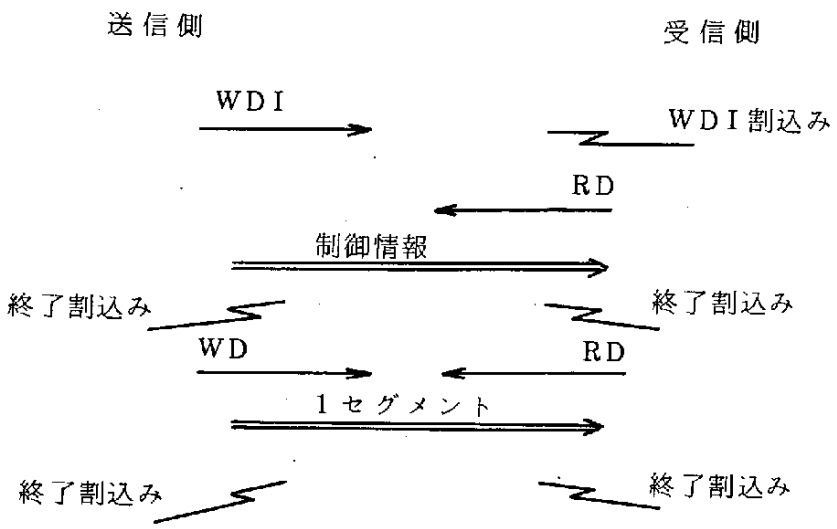


図2－19　データの転送

この場合，WDに対してWDIを衝突させることは許さない。

○一般データが複数セグメントになる場合

データを送信するHOSTは，WDIコマンドにより，制御情報（メッセージ種類コード＝０）を送り，IMPから対応するコネクション番号を持つものに対する制御情報（メッセージ種類コード＝12：複数セグメント受入れ準備完了）が送られてくるのを待つ。

この間に他の情報ならば任意のものを送ることができる。

IMPから制御情報が送られてきたならば，ただちにWDコマンドにより一般データを構成する全セグメントを送らなければならない。

IMPからHOSTに送る場合にはWDIコマンドによって制御情報（メッセージ種類 コード＝０）を送り，これに続いて第１セグメントから最終セグメントまでをWDコマンドにより送る。

第１セグメントを送ってから最終セグメントを送るまでの間にHOST（あるいはIMP）はPSW lock以外でデータ転送を中止してはいけない。PSW lockになった場合には少くとも一方の処理系はダウンしたものとみなす。

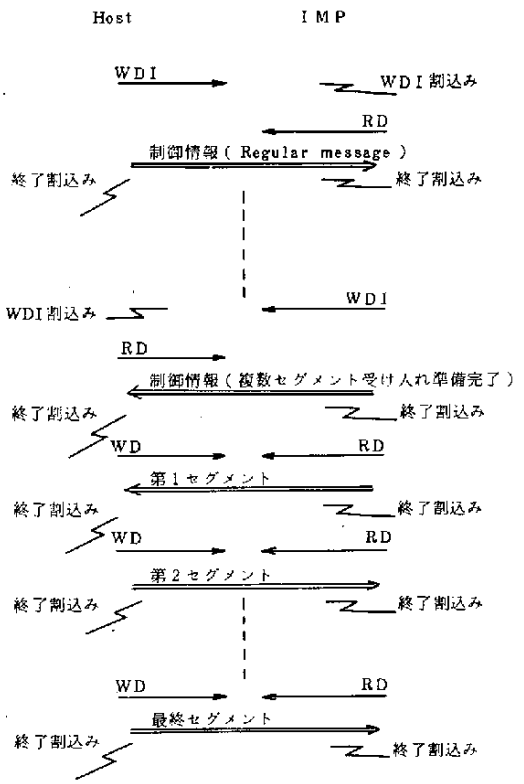


図２－20　多重セグメントデータの転送

F　衝突手順

アダプタの両側から同時にコマンドが発信された場合に，アダプタあるいはチャネルなどの場所でコマンドの衝突がおこる。

これはチャネル正常手順でおこるとすれば，WDIとWDIだけである。衝突があった場合は必ずIMP側のコマンドを優先させなければならない。

すなわち，HOST側は，データの受け入れ準備をしなければならない。

G　終了処理手順

終了処理手順は通常の状態においてHOSTあるいはIMPが正常に処理を終了する手順を決めたものである。

終了処理の開始は，一方が他方に対して，接続中止要求制御情報（メッセージ種類＝4）を発信することにより，双方においておこなう。これを受けた処理系は，同様の制御情報を確認のために送らなければならない。

終了処理は，開始してから5分以内に終わらなければならない。

終了処理が終ると，切断完了（メッセージ種類＝5）を発信して，相手が死んだことにし，初期状態待ちにはいる。

2.4　HOST—HOSTプロトコル

HOST—HOSTプロトコルは，HOST同士が交信し，データを転送しあう方法，手順に関する規約である。

HOST対間の転送は，コネクションを通じ行なわれる。NCPは，その機能を遂行するために他のHOSTのNCPと通信を行なう必要がある。その目的のために，各HOST対間には前もってコネクション番号0のコネクションが割当てられている。それをコントロール・コネクションと呼ぶ。コントロール・コネクション上を行来するメッセージをコントロール・メッセージと呼び，これにはコントロール・コマンドが入っている。

ユーザ・プロセス対間の通信には，コネクション確立要求によって割当てられるコネクションが使われる。それを一般コネクションと呼ぶ。一般コネクション上を行来するメッセージを一般メッセージと呼び，これにはユーザが指定したデータ（メッセージ）が入ってくる。

A　フロー制御

NCPは，送信側プロセスが受信側プロセスで処理（受信）されていくより早くメッセージを送ると，受信側HOSTにメッセージがたまってしまうので円滑なデータ転送のためのフロー制御を行なう。

プロセス対間で1つのコネクションが確立すると，そのコネクションに1個のバッファを割当

てる。それを基底バッファと呼び，そのコネクションが解放されるまで保持される。それ以上は必要に応じてバッファ・プールなどから取り出して供給され，使用後はプールに返却する。あるコネクションに対するバッファ占有量が大きくなると，他のコネクションに対するサービスに支障をきたすので，バッファ・プールに残ったバッファ数が，ある限度を切った場合，多量にバッファを占有しているコネクションに対してデータの送出の一時停止を要求する。このコネクションに対しては，基底バッファのデータがユーザ・プロセスに渡された時点で送信の再開を許す。

B　コントロール・コマンド

コントロール・コマンドは，ネットワーク・ジョブに対してサービスをするために使用され，コントロール・コネクションを通じて目的地HOSTに転送される。

各HOSTのNCPは，コントロール・コマンドを発信・送信し，解釈・実行できなければならない。

コントロール・コマンドは，図2－21に示すように，8ビットのコマンド・コードと568ビット以下のコマンド・パラメータから構成されている。

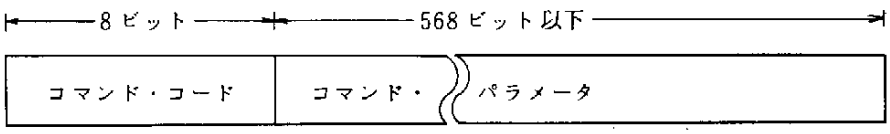


図2－21　コントロール・コマンドの形式

・コマンド・コード（8ビット）

HOST－HOSTプロトコルで定めたコントロール・コマンドのコード

・コマンド・パラメータ（0～568ビット）

各コマンドで決められたパラメータ

コントロール・コマンドは，次の4種類に大別される。

① コネクションの確立・解放

② イベント情報の通知

③ プロセスの起動・消去

④ その他

(1) コネクションの確立・解放

NCPは，ユーザ・プロセスからコネクション確立要求を受け付けるとRFC（Request－For－Connection）コマンドまたはCOK（Connection－OK）を目的地NCPへ送る。これらのコントロール・コマンドのパラメータのport－id（高位プロトコル図2－24参照）が一致する一対のコマンドが交換させるとコネクションが確立する。

NCPは，ユーザ・プロセスからコネクション解放要求を受け付けるとCLS（Close）コマンドを目的地NCPへ送る。どちら側もCLSコマンドを送り，受け取り，port－idが一致するとコネクションが解放される。

① コネクション確立要求コマンド（RFC，Request For Connection）

・一般形

RFC　<my－port－id>，<your－port－id>，<ecb－id>，

$$\left\{\begin{array}{l}<\text{コネクション番号}>\\<\text{最大メッセージ長}>\end{array}\right\}，<\text{passward}>$$

・コマンド・コード

RFCのコマンド・コードは，16進で01である。

・パラメータ

<my－port－id>　コネクション確立要求を発したユーザ・プロセスが指定したmy－port－idである。パラメータ長は40ビットである。

<your－port－id>　コネクション確立要求を発したユーザ・プロセスが指定したyour－port－idである。パラメータ長は40ビットである。

<ecb－id>　NCPが割当てたecb－idである。このパラメータは拡張用であり，現バージョンでは使用してない。パラメータ長は40ビットである。

<コネクション番号>　ユーザ・プロセスが指定したmy－port－idが受信側portである場合，NCPがそのコネクションに割り当てたコネクション番号である。コネクション確立後は，この番号によってプロセス対間のデータの送受を行なう。パラメータ長は8ビットである。

<最大メッセージ長>　ユーザ・プロセスが指定したmy－port－idが送信側portである場合，ユーザ・プロセスが指定した最大メッセージ長（ℓビット）を次の計算式で求めた値（ℓ'）である。

計算式

$$\ell' = \left[\frac{\ell + 143}{144}\right] - 1$$

但し，［　］は小数点以下切捨てを表わす。

パラメータ長は8ビットである。このため，最大メッセージ長は36,864ビット以下である。

<passward>　ユーザ・プロセスが指定したパスワードである。パラメータ長

は48ビットである。

・補足事項

ユーザ・プロセスがコネクション確立要求を出し，相手側より一致するRFCコマンドが到着してない場合，NCPはRFCコマンドを送出する。

このコマンドに対する応答は，コネクションの確立に成功した場合はCOKコマンドであり，コネクションの確立に失敗した場合はCLSコマンドである。

② コネクション確立応答コマンド（COK，Connection OK）

・一般形

COK　＜my－port－id1＞，＜your－port－id＞，｛＜コネクション番号＞／＜最大メッセージ長＞｝
〔，＜my－port－id2＞〕

・コマンド・コード

COKのコマンド・コードは，16進で02である。

・パラメータ

＜my－port－id1＞　コネクション確立要求を発したユーザ・プロセスが指定したmy－port－idである。パラメータ長は40ビットである。

＜your－port－id＞　コネクション確立要求を発したユーザ・プロセスが指定したyour－port－idである。パラメータ長は40ビットである。

＜コネクション番号＞　ユーザ・プロセスが指定したmy－port－idが受信側portである場合，NCPがそのコネクションに割当てたコネクション番号である。この番号によってプロセス対間のデータの送受を行なう。パラメータ長は8ビットである。

＜最大メッセージ長＞　ユーザ・プロセスが指定したmy－port－idが送信側portである場合，ユーザ・プロセスが指定した最大メッセージ長（ℓビット）を次の計算式で求めた値（ℓ'）である。

計算式

$$\ell' = \left[\frac{\ell + 143}{144}\right] - 1$$

但し，〔　〕は小数点以下切捨てを表わす。

パラメータ長は8ビットである。このため，最大メッセージ長は36，864ビット以下である。

＜my－port－id 2＞　ユーザ・プロセスがListen完了後のオープンの場合に指定したmy－port－id 2（Listenで指定したportと同一）である。これは，Listenで指定したmy－port－idとオープンで指定

したmy－port－idが異なるために使われる。

・補足説明

ユーザ・プロセスがコネクション確立要求を出し，相手側より一致するRFCコマンドが到着している場合，NCPはCOKコマンドを送出する。またすでに送出したRFCコマンドに一致するRFCコマンドが到着し，自系側portが入力側portである場合，RFCコマンドのすれ違いであるので，NCPは新めてCOKコマンドを送出する。

③ コネクション解放コマンド（CLS，Close）

・一般形

CLS ＜my－port－id＞，＜your－port－id＞，＜最終メッセージ番号＞

・コマンド・コード

CLSのコマンド・コードは，16進で03である。

・パラメータ

＜my－port－id＞　コネクション解放要求を発したユーザ・プロセスが指定したmy－port－idである。パラメータ長は40ビットである。

＜your－port－id＞　コネクション解放要求を出したユーザ・プロセスが指定したyour－port－idである。パラメータ長は40ビットである。

＜最終メッセージ番号＞　my－port－idが送信側portの場合はこのコネクション上でメッセージを送信した個数であり，my－port－idが受信側portの場合は受信側ユーザ・プロセスに渡されたメッセージの個数である。パラメータ長は32ビットである。

・補足説明

ユーザ・プロセスがコネクション解放要求を出すと，NCPはCLSコマンドを送出し，他系より一致するCLSコマンドを受け取った時コネクションが解放される。また，すでに受取ったRFCコマンドのパラメータとユーザ・プロセスより発せられたコネクション確立要求のパラメータとが一部しか一致しない場合にも，NCPはCLSコマンドを送出する。

④ メッセージ送信一時停止コマンド（CEASE）

・一般形

CEASE　＜コネクション名＞

・コマンド・コード

CEASEのコマンド・コードは，16進で04である。

・パラメータ

＜コネクション名＞　メッセージの送信を一時停止するコネクションのコネクション名である。パラメータ長は16ビットである。

・補足説明

受信側NCPは，受信側プロセスが処理（受信）するよりも早くメッセージが送られ，そのコネクションに対する受信側バッファの占有量が大きくなると，送信側NCPに送信を一時停止するように，CEASEコマンドを送出する。

⑤ メッセージ送信再開コマンド（RESUME）

・一般形

RESUME　＜コネクション名＞

・コマンド・コード

RESUMEのコマンド・コードは，16進で05である。

・パラメータ

＜コネクション名＞　メッセージの送信を再開するコネクションのコネクション名である。パラメータ長は16ビットである。

・補足説明

受信側NCPは，受信側プロセスがメッセージ送信の一時停止されているコネクションにたまっているメッセージをすべて受け取った場合，送信側NCPに送信を再開するよにRESUMEコマンドを送出する。

(2) イベント情報の通知

NCPは，小量メッセージに対しては，ユーザ・プロセスがこのためのコネクションを確立してメッセージを送信するわずらわしさから開放するため，受信側ecb－idを指定するだけで配達する。

① ポスト・コマンド（POST）

・一般形

POST　＜ecb－id＞，＜通知情報＞

・コマンド・コード

POSTのコマンド・コードは，16進で10である。

・パラメータ

＜ecb－id＞　ポスト要求を発したユーザ・プロセスが指定した受信側ecb－idである。パラメータ長は40ビットである。

＜通知情報＞　ユーザ・プロセスが指定したポストすべき内容である。パラメータ長は32ビットである。

・補足説明

POSTコマンドを受取ったNCPは，そのイベント情報を待っているユーザ・プロセスにそれを渡し，ユーザ・プロセスの待ち状態を解く。

(3) プロセスの起動・消去

NCPは，ユーザ・プロセスの依頼により他系HOSTにプロセスを起動・消去要求を目的地NCPへ送出する。これを受取ったNCPは，指定されたプロセスを起動し，消滅させる。

① プロセス発生・実行コマンド（ LOADE，Load and Execute ）

・一般形

LOADE ＜プロセス名＞，＜プロセス情報ヘッダー＞，＜ecb−id1＞

〔，＜ecb−id2＞〕

・コマンド・コード

LOADEのコマンド・コードは，16進で20である。

・パラメータ

＜プロセス名＞　ユーザ・プロセスのプロセス発生要求に対し，発信側NCP がそのプロセスを識別するために付けた名前である。パラメータ長は56ビットである。始めの24ビットはユーザ名である。

＜プロセス情報ヘッダー＞　このパラメータは，＜実行プログラム名＞，＜ジョブ情報＞，＜ファイル個数＞の3つから構成されている。各々の項目の長さは，64ビット，432ビット，そして16ビットであり，全体で512ビットである。

＜実行プログラム名＞　ユーザ・プロセスが指定した実行すべきプログラムの名前である。

＜ジョブ情報＞　ユーザ・プロセスが指定したジョブ情報であり，表2−4の形式である。

＜ファイル個数＞　ユーザ・プロセスが指定したプログラムを実行するために必要なCNT情報（ファイル）の個数である。

＜ecb−id1＞　ユーザ・プロセスが指定したプロセスのロード結果を通知するecb−idである。パラメータ長は40ビットである。

＜ecb−id2＞　ユーザ・プロセスが指定したプロセスの完了状態を通知するecb−idである。パラメータ長は40ビットである。

表2－4　ジョブ情報

項　　目	長　さ（バイト）	内　　容
ユーザ番号	6	ネットワーク・ユーザ番号（10進6桁，JISコード）
パスワード	6	ネットワーク・パスワード
最大コアー使用容量	2	1024バイト単位で指定
CPU使用時間	2	秒単位で指定
優　先　度	1	0：通常，1：優先
実行時パラメータ長	1	実行時パラメータのバイト数
実行時パラメータ	36	実行時にプロセスに渡すパラメータ

② プロセス消去コマンド（DELP，Delete Process）

・一般形

DELP　＜プロセス名＞，＜ecb－id＞

・コマンド・コード

DELPのコマンド・コードは，16進で21である。

・パラメータ

＜プロセス名＞　　発信側NCPが，LOADPコマンドでプロセスに割当てたプロセス名である。パラメータ長は56ビットである。

＜ecb－id＞　　プロセス消去要求を発したユーザ・プロセスが指定したecb－idである。このecb－idには，このコマンドの実行完了状態が通知される。

(4) そ　の　他

① 追加情報コマンド（CNT，Continue）

・一般形

CNT　＜情報コード＞，＜情報＞

・コマンド・コード

CNTのコマンド・コードは，16進で40である。

・パラメータ

＜情報コード＞　　追加情報が何を意味するかを示す。パラメータ長は8ビットである。

コード（16進）	内　容
01	ファイル情報

＜情　報＞　　情報コードに対応した情報である。パラメータ長は640ビット以下である。

ファイル情報の形式を表２－５に示す。

表２－５　ファイル情報（つづく）

項　　目	長　さ（バイト）	内　　容
ファイル定義名	8	プログラムで定義したファイル名であり，実際のファイルとのつき合わせに使用する（ＪＩＳコード）。
ファイル名	１６	実際のファイルの名前（ＪＩＳコード）
装置の種類	1	装置の種類を指定する。 コード（16進）　装置の種類 ００　ディスク装置 ０１　磁気テープ装置 ０２～ＦＦ　その他
ファイル編成法	1	ファイルがどのように編成されているかを指定する コード（16進）　編　成　法 ０１　順編成 ０２　分割形順編成 ０４　直接編成 ０８　順インデックス編成
ファイルの存在場所	6	様式未定
ファイルの種類	1	ファイルの種類を指定する。 コード（16進）　ファイルの種類 ０１　システム・ファイル ０２　パブリック・ファイル ０４　コモン・ファイル ０８　ユーザ・ファイル
ファイルの前処置	1	ファイルの前処置の方法を指定する。 コード（16進）　前処置の方法 ０１　ワーク（テンプ）ファイル ０２　新規ファイル ０４　旧ファイル

表２－５　ファイル情報　（つづき）

項　　目	長　さ（バイト）	内　　容
ファイルの後処置	1	ファイルの使用後の処置を指定する。 コード（16進）　後処置の方法 01　保　存 02　消　去 04　登録（カタログ）
パスワード	6	パスワードを指定する。
拡　張　用	10	拡　張　用

2.5　高位プロトコル

2.5.1　Demand Service Protocol

Demand Service Protocol（DSPと略称する）は，通常TSSサービスと称している Demand Service を他のHOSTから利用するさいにサポートしなければならない高位のプロトコルを規定したものである。

A　概　　要

DSPは図２－22で示すような構成になっている。

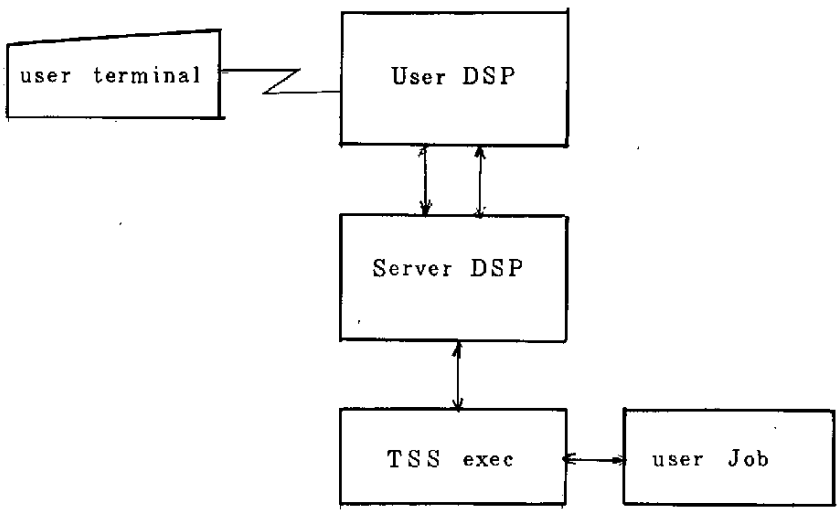


図２－22　DSPの構成図

DSPの実体は実際の端末ユーザが属するHOST内に存在するプロセスUser DSPと，ユーザ

ジョブが発生するHOSTに存在するプロセスServer DSPより構成され，このUser DSP とServerDSPとの間でのデータの受け渡し手順，タイミングなどを規定したものである。

DSPの基本部分はInitial Connection，Network Virtual Terminal（NVT），Flow Control，Optionの4つから構成されている。

○ Initial Connection

User DSPとServer DSP間で，両方向のコネクションを確立する手順を定めたものである。

○ Network Virtual Terminal

Server DSPはremote terminalに対してメッセージのやりとりをおこなうわけであるが，このremote terminalは各種の端末があり，これのコントロールをすべてServer側でおこなうことは不可能である。このため，Network に共通な仮想端末を規定して，すべてこれをコントロールする方式をとっている。

○ Flow Control

DSPは，相互にデータの転送を行なうが，Server DSPからUserDSPに対しては，データの転送が端末のスピードを上まわってUser側にデータが蓄積されることになる。この現象を起こさないようにServer－User間でデータ・フローを制御するものである。

○ Option

NVTは，特定の機能の多い端末に対して十分なサポートをおこなうことができないため，User－Serverの同意によりNVTの仕様を変更できるようにしてある。これがOptionである。

B Initial Connection

Initial ConnectionはUser DSPとServer DSPとの間で両方向のコネクションを確立する手順を決めたものである。

あるHOSTのNetworkおよびTSSサービス開始時にServer DSPは必ずmy－port－id＝0をNCPに対してListen状態にしておかなければならない。

User DSPは，Server DSPのport－id＝0に対してUser番号，HOST番号，識別番号よりなるport－idを指定してコネクションの確立要求をだす。このとき識別番号（port－idの下7ビット）はUser DSPが任意に決めてよい……ここでこれをnとする。

Server DSPはUserDSPのコネクション確立要求により，port－idがpostされるので，自分側のport－idを決め，postされたpost－idとコネクションを確立する要求を出す。……このとき自分側のport－idはUser番号はpostされたport－idのUser番号部分と同一，HOST番号は自己HOSTの番号，識別番号はn－1とする。

以上の動作によりUser DSP → Server DSPのコネクションが確立する。

こののち，User DSPはmy port－id，your port－idの識別番号をそれぞれn－1，nと

して確立を要求する。Server DSPはmy port－id，your－idの識別番号をそれぞれn，n－1としてコネクションの確立を要求する。……いずれのport－idもUser番号部分は，User DSPがコネクション要求をだした時のものでありHOST番号は対応する番号を使用する。……

このServer DSP→User DSPのコネクションの確立においては，User DSP側はServer DSPからのコネクション確立要求（RFCコマンド）をwaitし，その後OPENをおこなう（i，e，COKコマンドを返す）ような手順をとっても良い。しかし，Server DSP側はUser DSPからのRFCコマンドにも対応できるようになっていなければならない。

以上の動作によりServer DSP→UserDSPのコネクションが確立する。

いずれの場合もコネクションの確立時に指定するメッセージ最大長は，1セグメントの長さとするが，ここでは144バイト（1152ビット）と仮りに決めておく。

両方向のコネクションの確立で成功した時点でInitial コネクションの動作を終了するが，いずれか一方のコネクションの確立が失敗した場合にはすでに確立済のコネクションを解放して処理を終了する。

User DSPがコネクション要求時に決定するport－idの識別番号nは，本来はどのような方法によってもよいはずであるが，各DSPで勝手な方法で決めると，DSPにより発生させたuser Jobがnetwork accessをおこなう場合，他の高位プロトコルを使用する場合に，識別番号の一致という状態におちいる可能性がある。このようなことを避けるために次の手順によってnを決定することとする。

識別番号00_{16}～$3F_{16}$までの64個は高位プロトコルのためにリザーブされており，ユーザが何らかの形で高位プロトコルを使用する場合にはそれ以外の識別番号40_{16}～$7F_{16}$をユーザプロセス間で利用しなければならない。

DSPはport－idの識別番号として00_{16}，01_{16}がリザーブされておりdefaultの場合はこれを利用してコネクションを確立する。ただし同一User番号を持つユーザが同時に2台の端末を動かす場合には，同じport－idができることになるので，このような場合にはユーザの指示により，ユーザの使用できる識別番号40_{16}～$7F_{16}$の中から一対をとり出して利用することとする。これにより同一User番号を持つユーザは同時に複数個の端末を利用できることになる。（図2－23）はInitial connectionの確立過程の例である。

HOST#08 User DSP	HOST#06_{16} Server DSP
	LISTEN
	my port−id＝0000000600_{16}
OPEN	WAIT
my port−id＝ 060007080_{16}	→port−id：0600070801_{16}
your port−id＝0000000600_{16}	がpostされる。
	OPEN
	my port−id 1＝0600070600_{16}
	my port−id 2＝0000000600_{16}
	your port−id＝0600070801_{16}

user DSP → Server DSPのconnection確立

OPEN	OPEN
my port−id ＝0600070800	my port−id ＝0600070601
your port−id＝0600070601	your port −id＝0600070800

Server DSP → User DSP の connection確立

connection確立完了

注

user番号：060007_{16}
user HOST番号：08_{16}
serverHOST番号：06_{16}
} とする。

図2－23　Initial Connectionの確立例

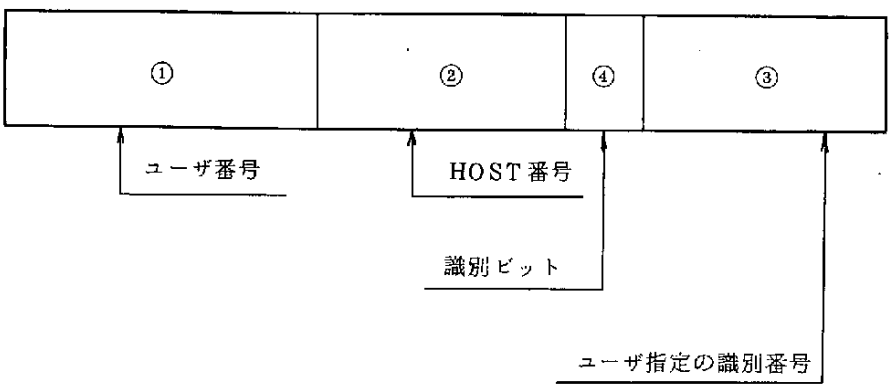


図2－24　Port−idおよびecb−idの形

① ユーザ番号（24ビット）

ユーザを識別するために使用される。最初の8ビットはユーザが所属するホスト番号のために使用する。

残りの16ビットは各ホストにおいて自由に使用して良い。

② HOST番号（8ビット）

Portあるいはecbが属するプロセスの存在HOSTを示す。

③ ユーザ指定の識別番号

Port—idとして使用する場合には，7ビット目が1の時，Write port—idであり，0のときRead port—idとなる。

④ 識別ビット（1ビット）

ecb—idならば1，port—idならば0，ecb—idとはevent control block—idでプロセスの同期をとる時に使用する。

C Network Virtual Terminal

Server DSPはremote terminalに対してメッセージのやりとりをおこなうが，remote terminalは多種多様な機能を持っているために，一元的な制御をおこなうことができない。このような状態では，Server DSPはすべての端末に対する制御方式を用意しなければならなくなり，実際上のインプリメンテーションは非常に困難である。

これをのがれるためには，端末の種類の限定をおこなうかあるいはServer DSPは端末の種類を全く意識しないで標準的なメッセージのやりとりだけをおこない，端末の制御はUser DSP側の責任においておこなうかのいずれかの方法を選択しなければならない。

NVTは，後者の考え方からできたものである。すなわちServer DSPはUser DSPをNVTというnetworkにおいて共通な仮想端末として制御し，User DSPはその制御にしたがって実際の端末を制御する方式である。（図2—25）はそのモデルである。

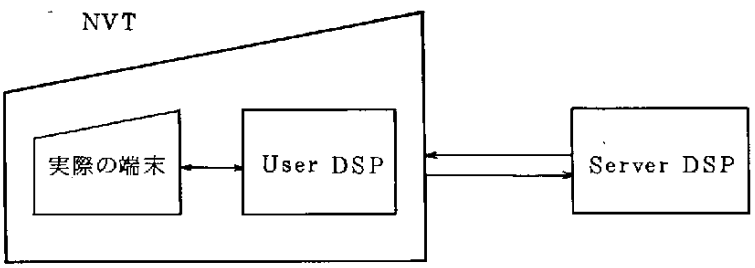


図2—25 NVTの制御モデル

NVTは基本的には全二重回線に接続され半二重動作をおこなう端末タイプライタと考えることができる。しかしながらNVTの機能は固定したものではなく，それを制御するプロセス（たとえばServer DSP）とNVT自身の間で会話することにより両者の合意が得られればその機能を変更することができる。

このために，まずＮＶＴの基本機能およびその制御方法を述べ，つづいて変更可能な機能および変更方法について述べる。

○基本機能

文字単位：８ビット

符号方式：最上位のビットが０のとき，ＪＩＳ情報交換用符号C6220の７単位符号のローマ文字・カナ文字併用符号とする。

最上位のビットが１のとき，NVTで定めた特殊符号とする。

伝送方式：ブロック伝送（不定長）

動　　作：半二重

○符号方式

付記１に示す如く，codeとしてはＪＩＳ７単位を使用するが，機能キャラクタとしては次のものが意味を持つ。ここで示した以外の機能キャラクタの意味は不定である。

機能キャラクタ名	コード（８進）	意　味
ＮＵＬ	0.0.0.	意味を持たない。
ＢＥＬ	００７	警報または合図をする。
ＢＳ	０１０	印字位置を同一行で１文字分後退させる。印字位置が第１文字目の時は意味を持たない。
ＨＴ	０１１	印字行に沿って，あらかじめ定めてある印字位置のすぐつぎの印字位置まで変換させる。
ＬＦ	０１２	次の印字行に移動させる。
ＶＴ	０１３	印字行をあらかじめ定めてある印字行のすぐ次の印字行まで移動させる。
ＦＦ	０１４	印字位置を次のページの定められた最初の印字位置まで移動させる。
ＣＲ	０１５	印字位置を同一行の初めの位置にもどす。
ＳＯ	０１６	カナ文字用符号であることを指定する。
ＳＩ	０１７	ローマ字用符号であることを指定する。

○特殊符号

特殊符号は８ビットの文字コード中の最上位のビットが１のものであってNVTとのやりとりにおいて特殊な働きをなすものである。特殊符号には次のものがある。

符号名	コード(16進)	意味
SMH	FF	Special mark header 以下に記述する特殊符号の前に必ずこの文字をつけなければならない。
GA	FE	Go ahead NVTを制御するプロセスからNVTに対して送られる符号であって，これを受けたNVTは送信状態となる。
RRT	FD	Request for the right of transmission NVTが自分を制御するプロセスに対して送信権を要求する。これを受け取ったプロセスはGAを送信して受信状態となる。
IP	FC	Interrupt Process DSPにより利用されているTSSに対して割り込みをかける。
AO	FB	Abort output TSSプロセスより転送されるメッセージを捨てる。この符号によりServer DSPは端末に対するread命令がくるまでメッセージを捨てなければならない。
OH	FA	Option header NVTの機能変更をおこなう会話に使用するヘッダーである。
FS	EF	Force Send 強制的に送信権をServer DSP側に持ってくる。User側にあった送信権はなくなり，すでにメッセージが送られている時には捨てられる。

○ NVTの制御

NVTとそれを制御するプロセス間における初期設定（すなわちコネクションの確立）はInitial connectionにおいてすでに述べた。

このようにして初期設定が終了した時点でNVTは受信状態，制御プロセスは送信状態になる。

NVTは通常受信状態になっており，この状態ではプロセスからのメッセージを受信するか，特殊符号のRRTを発信するかのいずれかしかおこなうことができない。NVTがRRTを発信しそれに対するGAが返ってきたときにNVTは送信状態になる。

NVTが受信状態から送信状態に切り換わるのは，上記の場合と，制御プロセスから特殊符号のGAを受信した場合だけである。

NVTが送信状態から受信状態に切り換わるのは，1行分のメッセージを制御プロセスに送

信した時である。

制御プロセスは，NVTに対してメッセージを送信する場合には自分が送信状態になっていることを確認してメッセージを送りだす。もし自分が受信状態ならば，メッセージにＦＳをつけて送り出す。

制御プロセスはNVTに対してメッセージを要求する場合には，GAあるいは送信メッセージの頭にGAをつけて送信する。

GAだけの場合には，NVTは送信状態となり，メッセージの入力が完了し次第送信してくる。GAつきメッセージの場合には，NVTに対してメッセージを出力したのち，GAだけの場合と同じ処理をする。

D　フローコントロール

User DSPからServer DSPへのメッセージはuser terminal orientedに発生するのでServer DSP側に蓄積することは余り考えられない。また蓄積することがあったとしてもServer HOSTはTSSをサービスするコンピュータであるからバッファの余裕もありそんなに問題にならない。

しかしながら，ServerからUserへのメッセージはその出力がterminalに対するものであるから処理速度が遅いにもかかわらず，送信側はそれをはるかに上まわる速さでメッセージを送り出すことができる。この場合も，User DSP側に十分なバッファがあれば（たとえば大記憶経由で端末に出力する）問題はないが，ＴＩＰのような場合であると，このようなことは許されなくなる。このような場合，NCPでメッセージはdiscardされて，せっかく送ったものを再送しなければならなくなる。

このために，DSPにおいては，Server DSPからUser DSPに対して送るメッセージに関してflow controlをおこなう。

これは次のような方法によっておこなう。

User DSPにおいてあらかじめある個数（これをｎ個とする）のバッファを用意しておく。最初にUser DSPからServer DSPに対してｎを通知（POST）する。そしてServer DSPからUser DSPに対してメッセージを送るたびに両者で１をひき，この値が０になるまでメッセージを送り続けることができる。０になった時点でUser DSPからServer DSPに対して新らしく用意されたバッファの個数がPOST（HOST—HOSTプロトコル参照）される（必ずｎ個）。このためServer DSPはPOSTを受けるためにwaitしなければならない。

バッファ個数のリセット（ｎにする）はUser DSPが送信状態になる場合もおこなう。すなわち，Server DSPがGAを送った場合，両者においてバッファ未使用数はｎにセットされる。

このとき使用するecb—idはuser番号，HOST番号としてそれぞれユーザの番号，Serverが存在するHOSTの番号であり，識別番号としてUser DSP → Server DSPのコネクションの

識別番号のServer DSPのそれと同一のものを使用する。

E　Option

NVTとServer DSPの間において初期設定時には，NVTが基本機能だけで動作することになっているが，両者の了解のもとにその機能を変更することができる。ここでは，変更項目および両者での了解のとり方について述べる。このうち変更項目については現時点でのものであり，将来拡張される可能性はあるが，拡張された部分に対する変更要求がきた場合にはこれを拒否することにより，今までどうりの使用が可能である。

○ OPtion の会話

Optionの会話はWILL，WON'T，DO，DON'T，ＳＢの5つのオペレーションを使用することによりおこなう。

WILLとはこのコマンドの送信側が送信メッセージを変更したいことを通知する。あるいはDOに対する承認を示す。

WON'Tとは，送信側になっているコネクションのメッセージを現在の状態からdefaultの状態にもどすことを通知する。あるいはDOに対する拒否を示す。

DOとは自分が受信側になっているコネクションに対して受信メッセージを変更したいことを通知する。あるいはWILLに対する承認を示す。

DON'Tとは，自分が受信側になっているコネクションに対して受信メッセージをdefaultの状態にもどすように通知する。あるいはWILLに対する拒否を示す。

SBとはNVTの機能変更の了解がついた時点で，その機能変更の内容を詳細に打合せるために使用する。SBの使用法は，そのOptionに従属している。

SMH	OH	オペレーション	内　　容

オペレーション

00_{16} ： WILL

01_{16} ： WON'T

02_{16} ： DO

03_{16} ： DON'T

04_{16} ： SB

内　容

optionの種類により異なるが，SB以外は８ビットでoptionの種類を示す。SBの場合は任意の長さである。

図２－26　Optionの会話コマンド

○変更項目

現在なし

以下の表はＪＩＳ情報交換用符号よりそのまま転載した。

付　記　1

日　本　工　業　規　格　　ＪＩＳ

情　報　交　換　用　符　号　　C6220－1969

Code for Information Interchange

1. 適用範囲　この規格は，情報処理およびデータ伝送を行なうシステムで情報交換に用いる符号について規定する。

2. 用語の意味　この規格で用いるおもな用語の意味は，つぎのとおりとする。

(1) 情報交換　異なるシステム間でも相互に情報が利用できるように，一つのシステムから他のシステムへ情報を伝えること。

(2) キャラクタ　データの構成　制御または表現に用いる要素となるもので，文字，数字，記号および特殊な機能を表現するものからなる。

(3) 機能キャラクタ　特殊な機能を表現するキャラクタで，制御機能の開始，変更，停止などが指定される。

(4) 媒　体　データを物理的状態の変化として表わしうる物質，たとえば紙テープ，カード，磁気テープなどをいう。

3. 符　号　符号の単位は，7単位および8単位とする。

3.1　7単位符号　7単位符号は，ローマ文字用符号およびカナ文字用符号とし，この両者は単独で使用してもまた併用してもよい。

3.1.1　ローマ文字用符号　ローマ文字用符号は，表1のとおりとする。

3.1.2　カナ文字用符号　カナ文字用符号は，表2のとおりとする。

3.1.3　ローマ文字，カナ文字併用符号　ローマ文字，カナ文字併用符号は，7単位でローマ文字とカナ文字を併用する場合に使うものとし，表1，表2を次の方法で結合する。

表2の機能符号（未定義）部分の0/14および0/15(1)に表1と同義の機能キャラクタSOおよびSIを設け，表1と表2のSOおよびSIの相互利用によって結合する。この場合，SOが先行する符号群は，カナ文字用符号表に示す符号を意味し，SIが先行する符号群は，ローマ字用符号表に示す符号を意味するものとする。

注　(1)符号表上の位置については6.1参照。

3.2　8単位符号　8単位符号は，表3のとおりである。

表1　ローマ文字用符号

ビット番号 b_7	b_6	b_5	b_4	b_3	b_2	b_1	行＼列	0	0	0	0	1	1	1	1
								0	0	1	1	0	0	1	1
								0	1	0	1	0	1	0	1
								0	1	2	3	4	5	6	7
			0	0	0	0	0	NUL	(TC_7)DLE	SP	0	@	P	`(2)(5)	p
			0	0	0	1	1	(TC_1)SOH	DC_1	!	1	A	Q	a	q
			0	0	1	0	2	(TC_2)STX	DC_2	"(2)	2	B	R	b	r
			0	0	1	1	3	(TC_3)ETX	DC_3	#(4)	3	C	S	c	s
			0	1	0	0	4	(TC_4)EOT	DC_4	$	4	D	T	d	t
			0	1	0	1	5	(TC_5)ENQ	(TC_8)NAK	%	5	E	U	e	u
			0	1	1	0	6	(TC_6)ACK	(TC_9)SYN	&	6	F	V	f	v
			0	1	1	1	7	BEL	(TC_{10})ETB	'(2)	7	G	W	g	w
			1	0	0	0	8	FE_0(BS)	CAN	(	8	H	X	h	x
			1	0	0	1	9	FE_1(HT)	EM	)	9	I	Y	i	y
			1	0	1	0	10	FE_2(LF)(1)	SUB	*(3)	:	J	Z	j	z
			1	0	1	1	11	FE_3(VT)	ESC	+(3)	;	K	[	k	{(5)
			1	1	0	0	12	FE_4(FF)	IS_4(FS)	,	<	L	¥	l	|(5)(5)
			1	1	0	1	13	FE_5(CR)(1)	IS_3(GS)	−	=	M	]	m	}(5)
			1	1	1	0	14	SO	IS_2(RS)	.	>	N	^(2)	n	‾(2)
			1	1	1	1	15	SI	IS_1(US)	/(3)	?	O	_	o	DEL

注 (1)　0/10 "LF" および0/13 "CR" の動作を1動作で行なう装置では，0/10 を "NL" として使用する。

(2)　2/2 "¨", 2/7 "´", 5/14 "^" および6/0 "`" の記号は，それぞれドイツ語系のウムラウト，フランス語系のアクサンにも使う。その場合には0/8 "BS" を先行させる。

(3)　2/10 "*", 2/13 "−", 2/15 "/", 7/12 "|", 7/14 "‾" などは付属書2に示すように多重の意味を有するが，これらは情報送受間の相互了解のもとに記号の意味が混乱しないように固定する。

(4)　一つの符号位置に対して二重にキャラクタを配置した箇所はない。しかし，国際間の情報交換の場合にかぎり，情報送受間の相互了解のもとに2/3 "#" を "£" に置きかえて使用してもよい。

(5)　6/0 "`", 7/11 "{", 7/12 "|" および7/13 "}" の四つの記号は，表1，表2および表3に盛られている以外の記号および文字と置きかえてもよい。

表２　カナ文字用符号

ビット番号 b_7	b_6	b_5	b_4	b_3	b_2	b_1	行＼列	0	0	0	0	1	1	1	1
								0	0	1	1	0	0	1	1
								0	1	0	1	0	1	0	1
								0	1	2	3	4	5	6	7
			0	0	0	0	0			SP.	—	タ	ミ		
			0	0	0	1	1			。	ア	チ	ム		
			0	0	1	0	2			「	イ	ツ	メ		
			0	0	1	1	3			」	ウ	テ	モ		
			0	1	0	0	4			、	エ	ト	ヤ		
			0	1	0	1	5			・	オ	ナ	ユ		
			0	1	1	0	6			ヲ	カ	ニ	ヨ		
			0	1	1	1	7			ァ	キ	ヌ	ラ		
			1	0	0	0	8			ィ	ク	ネ	リ		
			1	0	0	1	9			ゥ	ケ	ノ	ル		
			1	0	1	0	10			ェ	コ	ハ	レ		
			1	0	1	1	11			ォ	サ	ヒ	ロ		
			1	1	0	0	12			ャ	シ	フ	ワ		
			1	1	0	1	13			ュ	ス	ヘ	ン		
			1	1	1	0	14			ョ	セ	ホ	゛		
			1	1	1	1	15			ッ	ソ	マ	゜		DEL

機能符号（未定義）　機能符号（未定義）　国字符号部分　国字符号部分

2.5.2　Remote Batch Protocol

Remote Batch Protocol（RBP）は他のHOSTからBatchあるいはRemote Batchを利用するためにサービスしなければならない高位プロトコルを規定したものである。

A　概　　要

RBPは図２－27で示すような構成になっている。

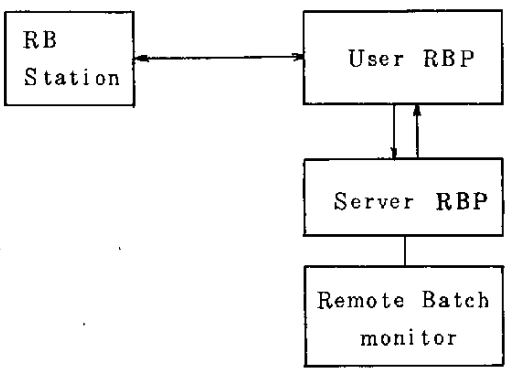


図２－27　RBPの構成図

RBPの実体は，Remote Batchを依頼するユーザが属するHOSTに存在するプロセスUser RBPと，Remote Batch Jobが発生するHOSTに存在するプロセスServer RBPにより構成され，このUser RBPとServer RBPとの間でのデータの受け渡し手順，タイミングなどを規定したものである。

RBPの基本部分は，Initial connection，DSP connection，Option extentionの3つから構成されている。

○ Initial connection

User RBPとServer RBPの間で両方向のコネクションを確立する手順を定めたものである。

○ DSP connection

DSP connectionとは，DSPにおいてUser DSP ↔ Server DSPの間で設定されるコネクションのことであって，NVT，フローコントロール，Optionの3つを含む。

RBPにおいてもこのコネクションを利用している。

○ Option extention

DSP connectionにおけるOptionは会話形を主体として構成されているが，RBPにおいてはこれを拡張して，より大量のデータの伝送を可能にするものを設定する。

B Initial connection

Initial connectionは，User RBPとServer RBPとの間で両方向のコネクションを確立する手順を定めたものである。この手順はDSPにおけるInitial connectionと次の一点を別にして全く同じである。この一点とは，Server RBPが最初にListenしているport－idが2であることである。

あるHOSTのNetworkおよびRemote Batchサービス開始時にServer RBPは必ずmy port－id＝2をNCPに対してListen状態にしておかなければならない。

User RBPは，Server RBPのport－id＝2に対してUser番号，HOST番号，識別番号よりなるport－idを指定して，コネクションの確立要求をだす。このとき識別番号（port－idの下7ビット）はUser DSPが任意に決めてよい……ここでこれをnとする。

Server RBPは，User RBPのコネクション確立要求により，port－idがpostされるので，自分側のport－idを決めpostされたport－idとコネクションを確立する要求をだす。……このとき自分側のport－idは，User番号がpostされたport－idのUser番号部分と同一，HOST番号が自己HOSTの番号，識別番号はn－1とする。

以上の動作によりUser RBP → Server RBPのコネクションが確立する。

こののち，User RBPはmy port－id，your port－idの識別番号をそれぞれn－1，nとして確立要求をする。Server RBPはmy port－id，your port－idの識別番号をそれぞれ

n，n－1としてコネクションの確立を要求する。……いずれのport－idもUser番号部分は，User RBPがコネクション要求をだした時のものであり，HOST番号は対応する番号を使用する。

以上の動作により，Server RBP→User RBPのコネクションが確立する。

いずれの場合もコネクション確立時に指定するメッセージ最大長は1セグメントの長さとするが，ここでは144バイト（1152ビット）と仮りに決めておく。

両方向のコネクションの確立が成功した時点でInitial connectionの動作を終了するが，いずれか一方のコネクションを確立が失敗した場合にはすでに確立済のコネクションを解放して処理を終了する。

RBPはport－idの識別番号として02_{16}，03_{16}がリザーブされておりdefaultの場合には，User RBPはこれを利用してコネクションを確立する。

ただし識別番号40_{16}～$7F_{16}$の中からユーザの指定にしたがいコネクションの確立をおこなってもよい。

図2－28はInitial connectionの確立過程の例である。

図2－28　Initial connectionの確立例

C　DSP connection

DSP connectionは，RBPにおいてコマンド，データなどが転送されるコネクションである。この仕様はDSPのそれと全く同じである。

D　Option extension

Optionに関する会話は，DSPのoptionにしたがっておこなわれる。変更項目は次の4種である。

a．EBCDIC data option

option種類＝00_{16}

このoptionが了解されたとき，一方から他方に送られるdataはEBCDIC codeとなる。ただし，optionの変更に関するものはＪＩＳ codeのままである。

b．unit address option

option種類＝01_{16}

このoptionが了解されたとき，Serverからくるメッセージ（GAだけのものも含む）の頭に必ずunit address byteが含まれている。unit address byteは，1バイトより成りメッセージ本文（NVTの特殊符号のあとで，通常出力される部分）の最初にこれがはいる。

このバイトは次のdeviceを示す。

device code

00_{16}	端末タイプライタ
01_{16}	ディスプレイ装置
02_{16}	磁気テープ装置
03_{16}	カードリーダ
04_{16}	紙テープリーダ
05_{16}	未使用
06_{16}	ライン・プリンタ
07_{16}	カードパンチャー
08_{16}	紙テープ・パンチャー
09_{16}	ＸＹプロッター

このoptionが了解されたとき，User側からServer側にSBによって上記端末種類が通知される。このときのformatは次のとおりである。

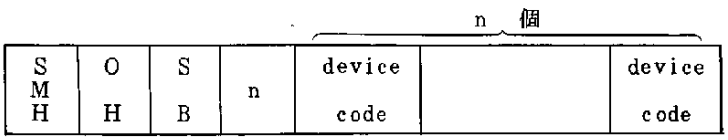


図2－29

c. Full duplex option

option 種類＝02_{16}

DSP connection を full duplex として利用する。

d. exchange right of transmission

option 種類＝03_{16}

RBPの送信権を常に端末側にもってくる。これにより，GAなくしてServer RBP へデータを送ることができる。また，このようになった場合には，Server →User のデータはRRTを発信してから送らなければならない。

2.5.3 File Transfor Protocol

File Transfor Protocol（FTP）はネットワーク内のHOST間でFile を転送するための高位プロトコルを規定したものである。

A 概　要

FTP は図２－30 で示すような構成になっている。

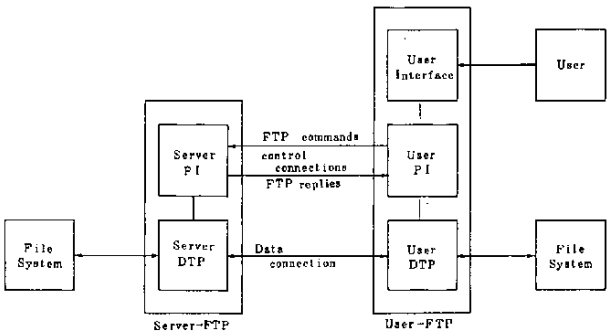


図２－30　FTP のモデル(1)

(1) Server—FTP プロセス

User—FTP あるいは他の Server—FTP と協調してFile 転送の機能を果たすプロセスであり，プロトコル・インタプリータ（PI）とデータ転送プロセス（DTP）から構成される。

(2) Server—PI

Server—PI はmy—port id＝4 に対してListen 状態にしておき，User—PIからのコネクション確立要求を待ち，コントロール・コネクションを確立する。Server—PI の役目はUser—PI からFTP コマンドを受取り，その応答を返すことと，Server—DTPを管理することである。

(3) Server—DTP

Server—DTPはServer—PIから指示により転送パラメータをセットし他系のUser—DTPあるいはServerのPortとのデータ・コネクションを確立し，データ転送をおこなう。Serverサイトのfile accessもServer—DTPの役割である。

(4) User—FTPプロセス

Server—FTP（1あるいは複数個）と協調してFile転送の機能を果たすプロセスであり，プロトコル・インタプリータ，データ転送プロセス，ユーザ・インタフェースから構成される。

(5) User—PI

User—PIはServer—FTPのport＝4に対してイニシャル・コネクション確立の要求をしコントロール・コネクションを確立する。コントロール・コネクションを通してFTPコマンドを送出し，その応答を受取る。

User—FTPがFile転送を行なう場合は，User—DTPを管理する。

(6) User—DTP

User—DTPは，User—PIからの指示により，転送パラメータをセットしServer—DTPとのデータ・コネクションを確立する。Server—FTPとのデータ・コネクションが確立されたら，Serverとデータの転送をおこなう。2つのServer間でデータが転送される時には，User—DTPは働かない。

図2－30に示すFTPの動作を以下に概説する。

Userからの指示により，User—PIはServer—FTPとInitial connectionを確立する。確立されたコントロール・コネクションを通して，User—PIはServer—FTPプロセスにFTPコマンドを送る。

Server—PIからUser—PIへの応答は同様にコントロール・コネクションを通じて返送される。

FTPコマンドは，データ・コネクションおよびFileシステムへのアクセスのためのパラメータ群を規定するものであり，Port，データの表現型式，ファイルのStore，Retrieve等のためのコマンドが用意されている。

User—DTP，Server—DTPはFTPコマンドの内容にしたがい転送パラメータを決め，データ・コネクションを確立する。

User—DTP，Server—DTPは指定されたパラメータにしたがって各ファイル・システムへアクセスし，データ・コネクションを通してデータを転送する。

上記のケースとは異なり，Userが存在するHOSTとは異なる2つのHOST間でfile転送をおこなう場合は図2－31のようなFTPの構成になる。

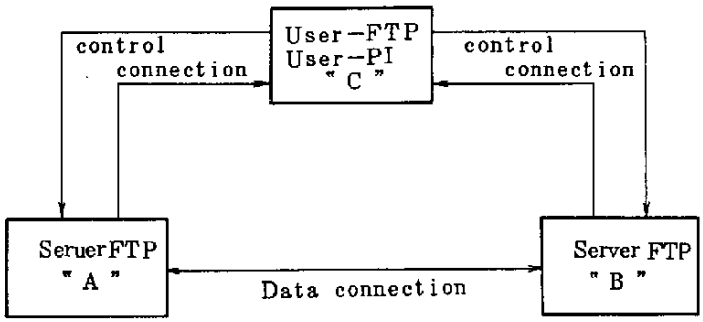


図２－31　FTPのモデル(2)

この場合は，User−FTPは２つのServer−FTPとコントロール・コネクションを確立した後，２つのServer−FTP間にデータ・コネクションを確立し，Server FTP間でデータ転送がおこなわれる。

B　データ表現型式とデータ転送方式

a．データ表現型式

データ転送においてFTPがあつかうデータの表現型式にはBinary型と Character 型とがある。いずれの場合もDTP間では転送の単位は８ビットを１単位とするが，HOST系のコア上での表現の違いにより上記２つの型を設ける。

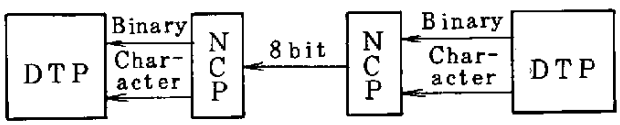


図２－32

(1)　Binary 型(B)

コア上連続する8bit単位のbit列を，そのままのimageでデータ転送するものである。たとえば１語32bitのHOSTと１語36bitのHOST間で４バイト分転送すると図２－33のようになる。

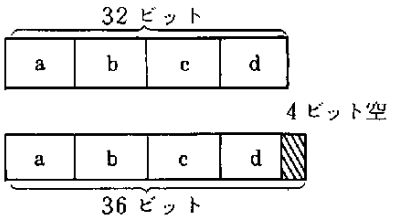


図 2 — 33

(2) Character 型(C)

8bit の転送単位をHOSTの文字表現単位に（から）あわせて変換し転送するものである。例えば 1 語 32bit 8bit／Byte のHOST系と 1 語 36 ビット 9bit／Byte のHOST系との間でデータ転送すると図 2 — 34 のようになる。

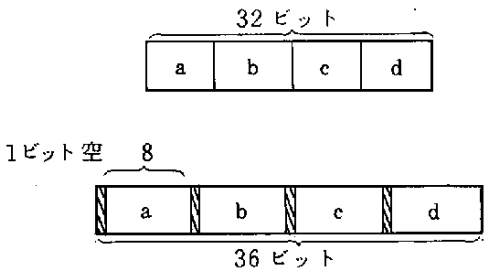


図 2 — 34

b．データ転送方式

FTP であつかうファイルは，複数のレコードの集りからなるレコード構造のファイルとする。

FTPはこのレコードを，データ・コネクションを通じて転送するわけであるが，レコード長がHOST—HOSTプロトコルのメッセージ長を超える場合，そのレコードは複数個のメッセージに分割されて転送される。分割した各メッセージには，そのメッセージの種類を示すDescriptor と，メッセージ内のデータ・バイト数を示すデータ・カウント等を含むメッセージ・ヘッダが付加される。

FTP であつかうレコードの最大長は仮に 2048 バイトとしメッセージ最大長は 1152 バイトとする。

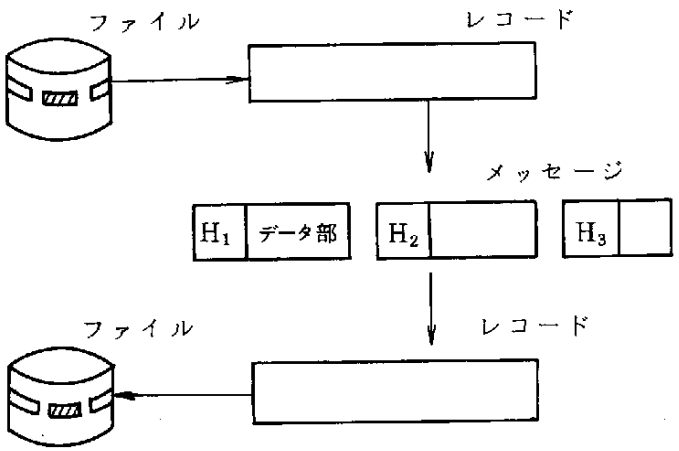


図２－35

ヘッダ情報の詳細は以下の通りである。

header format

8bit	8	16	0～nbit
HL	D	D.C.	filler

（内容はバイナリ値）

図２－36

(1) ＨＬ：header length（８bit）

ヘッダ部の長さを示し，８ビットを１バイトとするバイト数で表す。

(2) D　：Descriptor（８bit）

ヘッダ情報の種類を示す（下記以外は０）

EOR：End of Record：1

EOF：End of file　：2

(3) D.C.：Data Count（16bit）

データ部のデータ・バイト数を表す。

(4) filler：バウンダリ調整のためのエリアで，各HOST特有のバウンダリはこのfillerビットで調整する。

注　header formatはデータ・コネクション上でのビットパターンを示しており，データ表現型式の違いは考慮していない。したがって特有のバイト表現を有するHOSTのFTPにおいてはこのheaderの組立，解読に十分注意しなければならない。

c. Header length の決定法

filler バイトは，各HOSTによってデータ入出力のためのバウンダリが違うため，そのバウンダリ調整用に設けたものである。header length を決めるための手順を以下に示す。

① User—PIは，転送データ表現型式をTYPEコマンドによりServerに指示する。

② Server—PIは，送られたTYPEコマンドの内容と自HOSTのデータ表現方法とから，自HOST内データ入出力バウンダリを判断し，バウンダリ・バイト数をTYPEコマンドの応答として返答する。

③ User—PIはデータ・コネクションの両端DTPのバウンダリ・バイト数の最小公倍数を計算し，その値より大きいか等しい場合はその値をheader lengtk とする。

最小公倍数が4より小さい場合は4以上の公倍数をheader lengthとする。

その後User—PIはServerに対しBYTEコマンドによりheader lengthを指示する。

④ 各DTPはheader lengthを加味したメッセージ長によりデータ・コネクションを確立する。

C コネクション

FTPには，FTPコマンドが転送されるコントロール・コネクションとデータが転送されるデータ・コネクションがある。

a. コントロール・コネクション

コントロール・コネクションはDSPのInitial connection とDSP connection の基本機能部分をそのまま利用することとする。但し，Server—FTPが最初にListen している Port—id は4である。

FTPサービスの開始時に，FTP Server—PIはmy—port id＝4をNCPに対してLis—ten状態にしておかなければならない。

User—PI は Server の port—id＝4に対してコネクション確立を要求する。この手順はDSP の Initial connection Protocol と全く同一の手順とする。

User—PI，Server—PI 間のFTPコマンドおよびその返答は，すべてコントロール・コネクションを通して転送される。

Server はファイル転送が終了した後User—PI からの指示（BYE）によりコントロール・コネクションをcloseする。

User—PIが2つ目のServer—PIとコントロール・コネクションを確立する場合，User—PI はmy port—id＝22として2つ目のServer—PIのport—id＝4にコネクション確立要求をおこなう。

b．データ・コネクション

データ・コネクションは，User—DTPをServer—DTP間あるいは2つのServer—DTP間に用意されるデータ転送の為のコネクションである。

(1) コネクションの確立

Serverは，コントロール・コネクションを通しFTPコマンドをUser—PI から受け取り，転送パラメータを決めデータ・コネクションを確立する。その手順は以下に示すとおりである。

先ず，PORTコマンドによりデータ・コネクションの相手先port—idを知る。データ・コネクションのport—idは通常は6，7である。次にTYPEコマンドによりデータ表現型式がCharacter，Binaryのいずれであるかを知る。データ表現型式と自ホストのデータ表現方法とにより，自ホストに最適な入出力バウンダリ・バイト数を決定し，TYPEコマンドの応答としてその値を返答する。TYPEコマンドの応答を返すと，BYTEコマンドにより，データ・コネクションのheader lengthが知らされる。

さらに転送の方向を示すRETR iveあるいはSTORe コマンドによりファイル転送要求が送られてくるので，この時点でSeverはport id 6あるいは7を選び他DTPとのデータ・コネクションを確立すれば良い。

2つのServer間でデータ転送がおこなわれる場合は，User—PIは両Serverとコントロール・コネクションを確立した後両Server—PIにFTPコマンドを送出し，Server—DTPはPORT，STOR or RETRコマンドによって示されたHOSTのDTPとデータ・コネクションを確立する。

(2) コネクションの閉鎖

データ・コネクションは下記のような条件が発生したらCLOSEされる。

(1) header情報内の Decriptor＝EOFを送受した場合。

(2) 回復不能のエラーが発生した場合。

(3) フロー・コントロール

DTP間のデータ転送のフロー・コントロールは以下のように行う。

入力側DTPにおいてあらかじめある個数（n）のバッファを用意しておく。最初に入力側DTPで出力側DTPに対してnをPOSTする。そして出力側DTPから入力側DTPに対してメッセージを送るたびに両者で1を引き，この値が0になるまでメッセージを送り続ける。0になった時点で，入力側DTPは新しくバッファをn個用意し出力側DTPにnをPOSTするので，出力側DTPはPOSTを受けるために必らずWaitしなければならない。

データ・コネクションの確立例　1

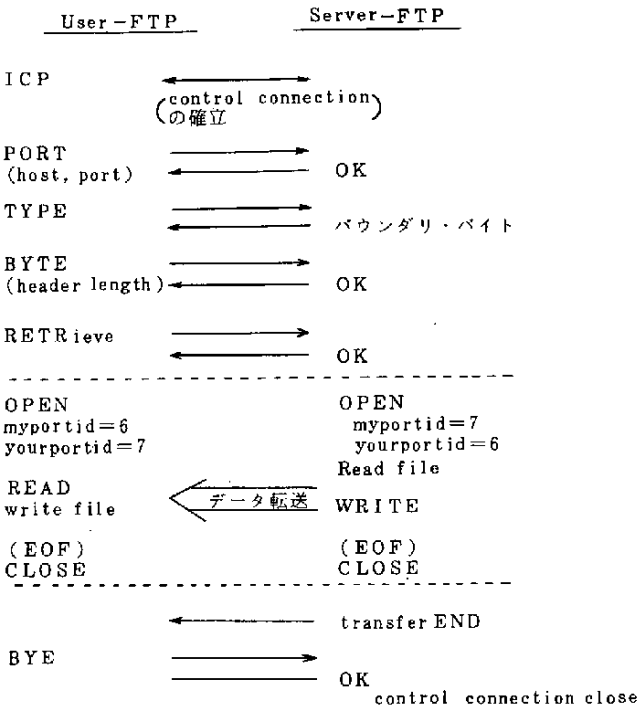


図２－37

データ・コネクションの確立例　2

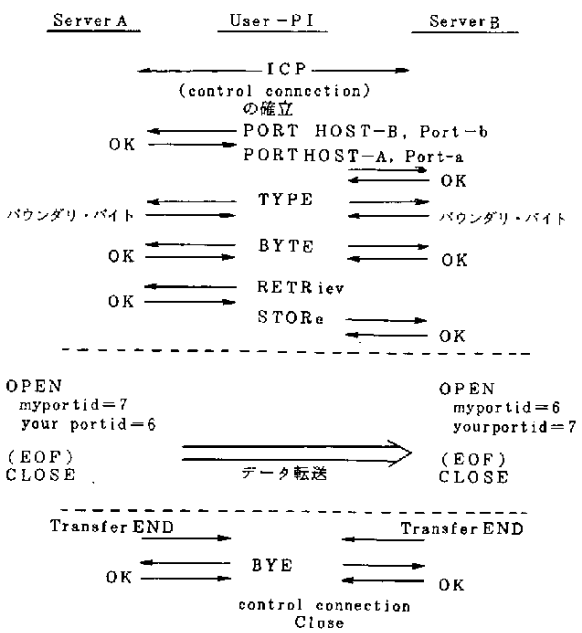


図２－38

D　FTPコマンド

Server--PIはコントロール・コネクションを通して，User－PIから各種のコマンドを受取り，その応答を返送する。

FTPコマンドはNVTのCRを最終文字とするNVT文字ストリングによって表わされる。

FTPコマンドはFTPアクセス・コントロール用のコマンドと，データ転送のパラメータ設定用のコマンド，FTPサービスを要求するコマンドの3グループのコマンド群からなる。

（FTPコマンドの形式は表2－5参照）

表2－5　FTPコマンドの形式

USER SP <user name> CR

BYE CR

PORT SP <port－id> CR

TYPE SP <type code> CR

BYTE SP <header length> CR

RETR SP <file identifier> CR

STOR SP <file identifier> CR

LIST CR

<user name> ::=<string>

<string> ::=<char>|<char><string>

<char> ::=CRを除くNVTコード

<port id> ::=JIPNET port-idを示す16進10桁のNVTストリング

<type code> ::=C|B

<header length> ::=データ・コネクションヘッダ長で10進2桁のNVTストリング

<file identifier> ::=<file name>

<file name> ::=<string>

a．FTPアクセス・コントロール・コマンド。（　）内はcommand code

(1) User name（USER）

パラメータとしてUserの識別名をNVTストリングで指定する。これは，fileシステムへのアクセス・コントロール用に使用され，コントロール・コネクション確立後最初に転送されるコマンドである。

(2) BYE（BYE）

パラメータなしのコマンドでファイル転送中でなければコントロール・コネクションを，

CLOSEする。ファイル転送中であればファイル転送終了後，終了応答を返し，コントロールコネクションをCLOSEする。

b．転送パラメータ設定用コマンド

(1) Port（PORT）

パラメータとしてPort idを持つ。

PORTコマンドはServerに相手DTPのport idを知らせるためのコマンドで，16進10桁のNVTストリングでport idを表す。Userからの特別な指定がない限り，port-id識別番号は6あるいは7である。

(2) Representation type（TYPE）

パラメータとしてデータ表現型式（2.1）を表すNVT文字BあるいはCを指定する。

Server—PIはTYPEコマンドの応答として自HOSTの入出力バウンダリ・バイト数を返答する。

(3) header dyte size（BYTE）

パラメータとして10進2桁のNVTストリングでheader lengthを示す。DTPはheader lengthを加味してmessage sizeを決める。

c．FTPサービス・コマンド

(1) Retrieve（RETR）

パラメータとしてfile識別情報を持つ。

RETRコマンドを受取ったServer DTPは，指定されたfileのコピーをUser—DTPあるいは他のServerにデータ・コネクションを通して転送する。

転送後，fileの状態，内容を変更してはならない。

(2) Store（STOR）

パラメータとしてfile識別情報を持つ。

STORコマンドを受取ったServer DTPは，データ・コネクションを通して転送されてくるデータをfileとして記録する。指定されたfileと同名のfileが存在する場合は新しく転送されたデータと置き換える。

同名のfileがない場合は新しくfileを創成する。

(3) List（LIST）

パラメータなしのコマンドである。

LISTコマンドを受取ったServer FTPは，そのユーザのファイル名一覧をコントロール・コネクションを通して送る。ファイル名は応答コードに続きSPで仕切り5個連続させ最後にCRを置いたものを1セグメントとし複数のセグメントで送出する。最終セグメント以外は1次応答コードを使用し，最終セグメントは2次応答コードを先頭に置く。

d．FTPの応答

Server—PIはUser—PIから送られてきたFTPコマンドに対し何らかの応答を返さなくてはならない。

応答は3桁の数字からなる応答コードと必要に応じて説明用の文字ストリングが付加された形式をとり最終文字位置にはCRLFを置く。（いずれもNVTコード）

・nnn CRLF

・nnn……CRLF

応答の形式

また，コマンドによっては，コマンドを正しく受取ったという1次応答を返した後，コマンド動作終了後2次応答を返すという2段階の応答形態をとる場合がある。

3桁の応答コードは先頭の1桁で応答の大まかな分類を示し，下2桁でその細分類を示す。（現在Xは何でも良い）

0XX：コマンド動作正常終了

1XX：コマンドを受取り動作は必要な時に行う。

2XX：コマンドを受取った（2次応答あり）

5XX：コマンドのシンタックスエラー

6XX：コマンドの実行は不可能

7XX：コマンドの動作が異常終了

各コマンドに対する応答を表2—6に示す。

表2—6

コマンド	応答コードの1桁目		
	1次 ＼ 2次	正常	異常
USER		0	5，6
BYE		0，1	5，6
PORT		0	5，6
TYPE		0	5，6
BYTE		0	5，6
RETR	1次	2	5，6
	2次	0	7
STOR	1次	2	5，6
	2次	0	7
LIST	1次	2	5，6，7
	2次	0	7

2.6 HOST－HOSTプロトコルの検討

現在，稼動中あるいは開発中のコンピュータ・ネットワークは多数あるが，HOST－HOSTプロトコル（以下H－Hと略称する）に関する報告は数少ない。この理由は，稼動中の汎用目的のコンピュータ・ネットワークが少ない点と，開発中の場合には技術的な詰めが，かなりのところまで終了しないとH－Hができあがらない点にあるということができる。

この項では，ARPA－net，Cyclades，NPL，JIPNETのH－Hの概要をのべ，それぞれのH－H比較検討をおこなうことにする。

2.6.1 HOST－HOSTプロトコルの概要

H－Hは，HOSTが他のHOSTと協力してベーシックなサービスを利用者に対しておこなう 規約であり，具体的にはNCPがこの機能をはたす。

H－Hの機能は3つに大別される。

① プロセスあるいは利用者の識別

② 授受されるメッセージフォーマットと意味づけ

③ コントロール・コマンド（NCP間会話コマンド）

- ○利用者のメッセージ交換手順
- ○メッセージ・フローの制御
- ○エラーに対する制御
- ○割り込み信号の授受

以下それぞれの項目について4つのコンピュータ・ネットワークのH－Hの概要をのべ，最後に一覧表をのせる。

A コミュニケーション識別番号（port－id）

(1) ARPA－net

socketと呼び40ビットより構成される。その内容は所属HOST番号，ユーザ番号および任意の8ビット番号（AENと呼ぶ）である。

コントロール・コマンドのパラメータとして使用する場合には，所属HOST番号を省略して用いる。

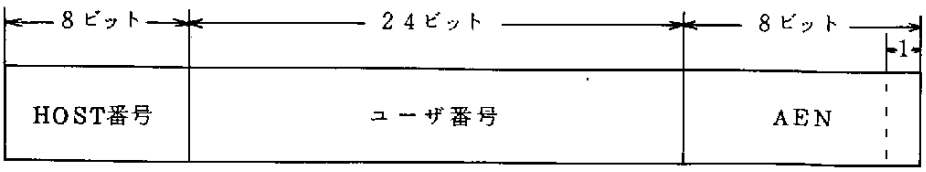
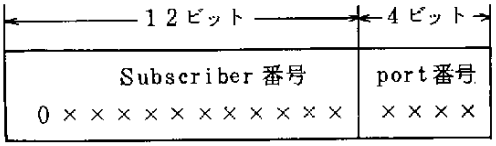


図2－39 ARPA—net の socket

AENは，ユーザが種々のportを作り出すためのものであって，最後のビットがonのとき，送信port，offのとき受信portとなる。（図2－39の説明）

(2) CYCLADES

Subscriber番号，port番号の16ビットより構成され，次の2つの形を持っている。



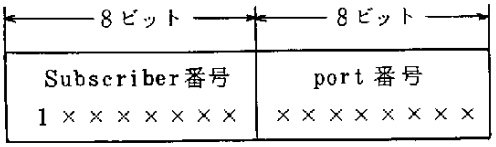


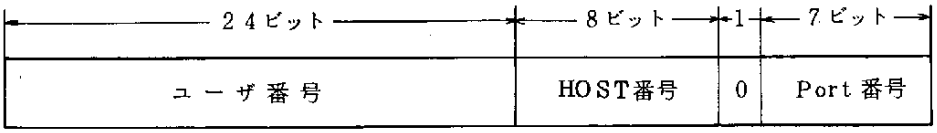
図2－40　Cyclades のport－id

(3) NPL

Logical channelと呼び8ビットより構成される。この番号は，network control executiveで管理されている。

(4) JIPNET

ユーザ番号，HOST番号，port番号の40ビットで構成されている。



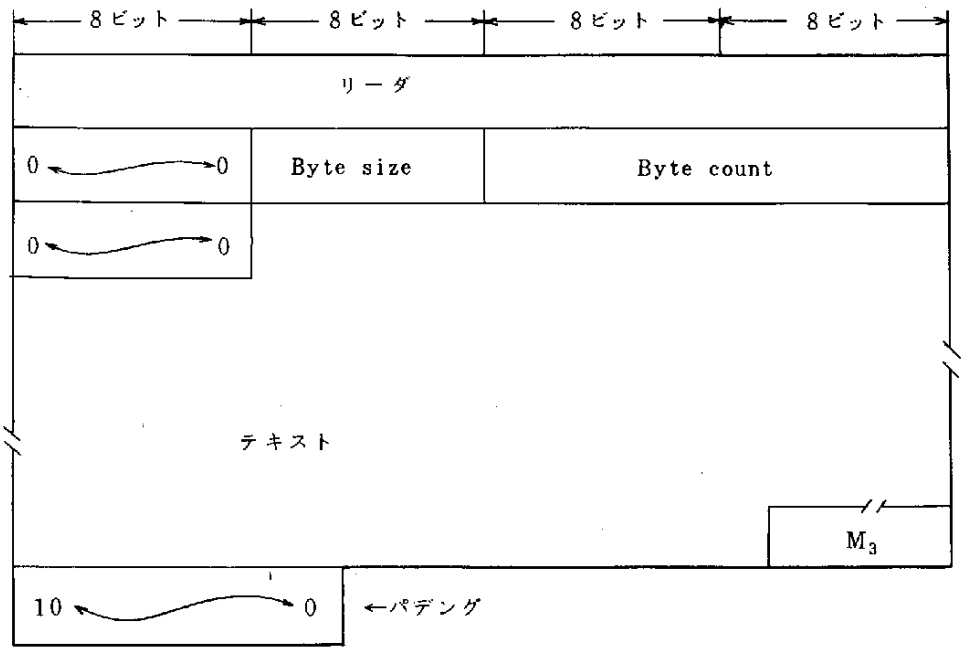
port番号はARPA－netのAENと同じで最後のビットのオン，オフにより，送信，受信port－idを示す。

図2－41　JIPNETのport－id

B　メッセージ形式

(1) ARPA－net

ヘッダー，テキスト，送信バウンダリー調整フィールド，パディングより構成され，テキストは8023ビット以下である。



リーダ（32ビット）　：HOST−IMPプロトコルによる。

Byte size （8ビット）：テキストが何ビットを単位としたものであるかを示す。

Byte count（16ビット）：テキスト長をByte size × Byte count で示す。

M_3（0ビット以上）　：送信バウンダリー調整フィールドで必ずall 0である。

パディング　：ハードウェア・パディング

図2−42　ARPA−netのH−Hメッセージ

(2) CYCLADES

CIGALEのパケット形式をそのままHOSTに持ち込んでいる。

パケット形式は次のとおりである。

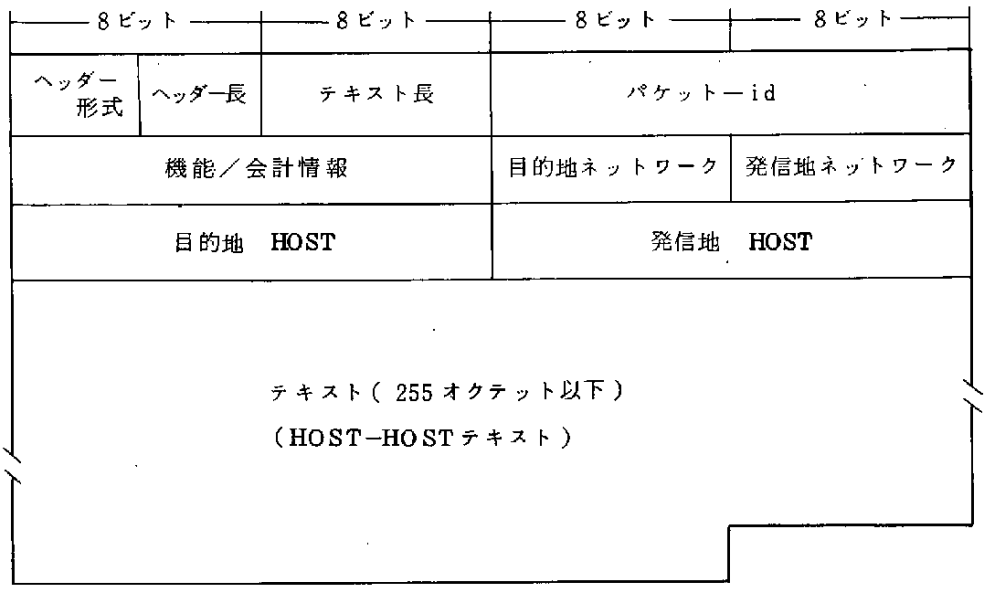


図2－43　CycladesのH－Hメッセージ

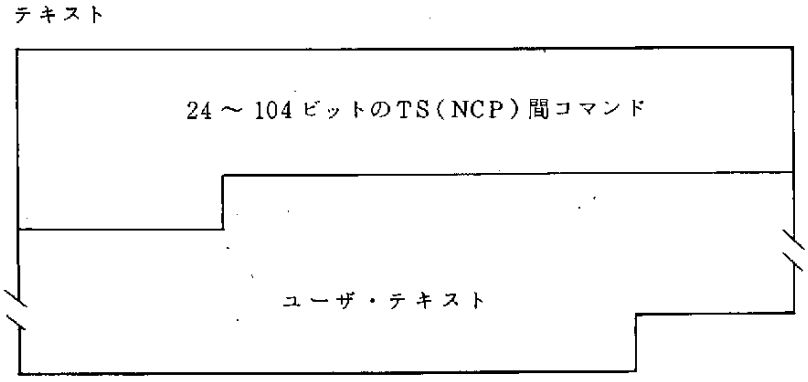
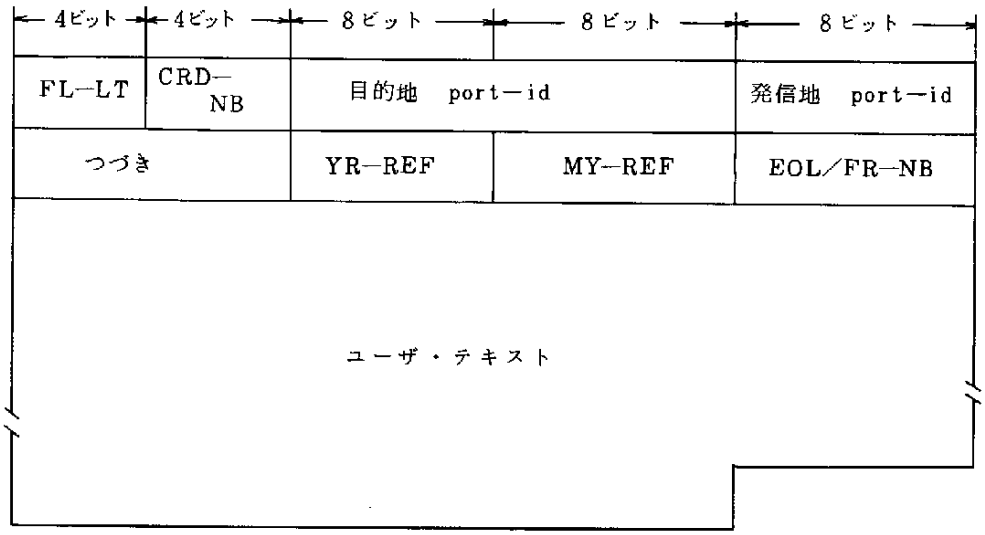


図2－44　CycladesのHOST－HOSTテキスト

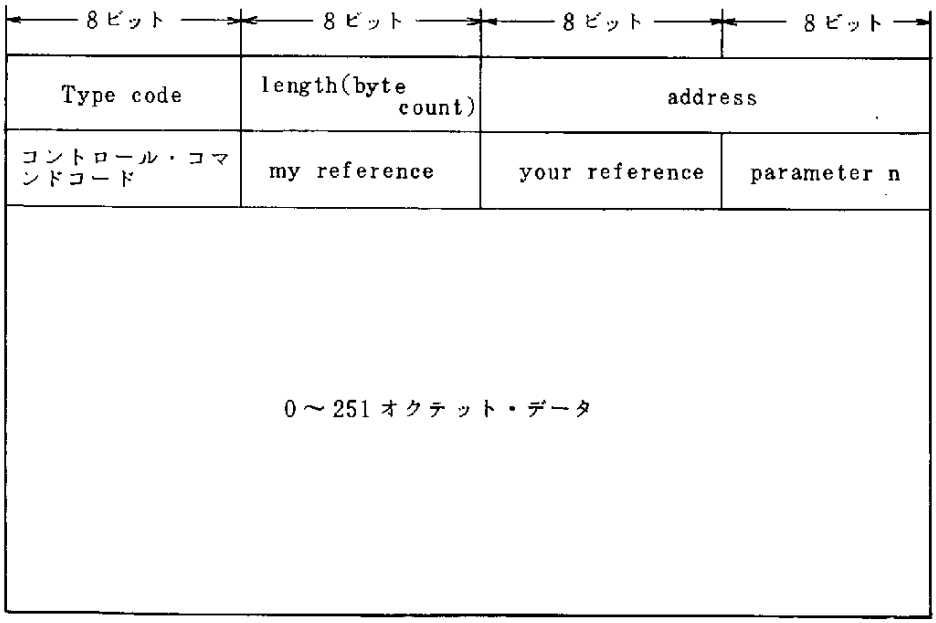


内容については，コントロール・コマンド参照のこと。

図２－45　CycladesのHOST—HOSTテキストの例

(3) ＮＰＬ

パケット・ヘッダー，コマンド，データより構成されデータ長は251オクテット（2008ビット）以下である。



Type code

0 : normal data traffic

1 : destination not currently available

2 : destination nonexistent

3 : error found in byte count

4 : error found in type code

my reference, your reference : logical channel 番号を入れる。

parameter n : コントロール・コマンドにパラメータが必要なときにここを利用する。

図2－46　NPLのH－Hメッセージ

(4) JIPNET

32ビットのリーダ，9216ビット以下のメッセージ本体から構成され，リーダ，本体は別のi／o命令によって送られる。メッセージ本体が1152ビットを越えるときは1152ビット単位に区切って送られ，最後のものだけ1152ビット以下となる。

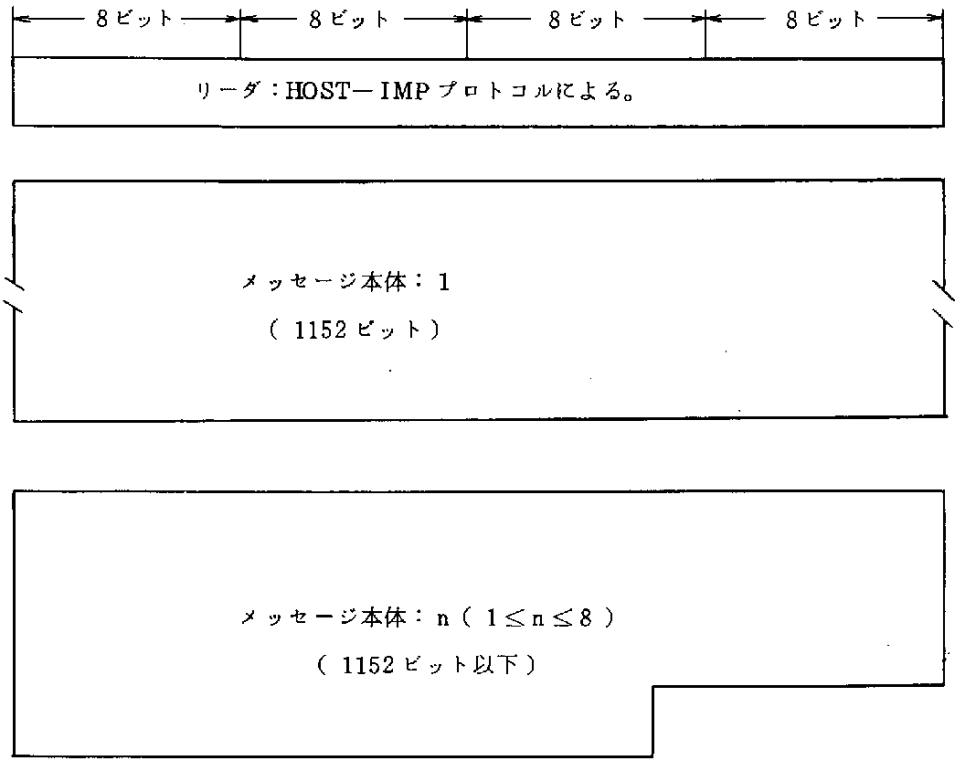


図2－47　JIPNETのH－Hメッセージ

C　コントロール・コマンド

(1)　ARPA—net

○ STR(Sender to Receiver)

my socket, your socket, byte sizeを指定して，コネクションの確立を要求する。送信プロセス側のNCPから受信側のNCPに送られる。

○ RTS(Receiver to Sender)

my socket, your socket, link番号を指定して，コネクションの確立を要求する。受信プロセス側のNCPから送信側のNCPに送られる。

RTS，STRの両者によってコネクションが確立する。link番号はコネクション確立後のメッセージ伝送時に，送受信socketの対に対する名前として利用される。

○ CLS(close)

my socket, your socketを指定してコネクションの解放を要求する。STR，RTSに対する拒否にも利用される。

○ ALL(Allocate)

link番号，メッセージ個数，バッファ容量を指定して，メッセージ個数あるいはバッファ容量がつきるまで当該linkに対してメッセージを送ってよいことを送信側に通知する。受信プロセス側のNCPから送信側のNCPに送られる。

○ GVB(Give back)

link番号，メッセージ個数，バッファ容量を指定して，当該linkに対して割り付けたバッファの返却を求める。受信プロセス側NCPから送信側のNCPに送られる。

メッセージ個数(f_m)，バッファ容量(f_b)は，送信側プロセスのNCPにおいて該当リンクに割り付けられたもののうちf_m／128，f_b／128以上の整数値を返却することを意味する。

○ RET(Return)

link番号，メッセージ個数，バッファ容量を指定して，GVBに対する応答として，バッファを返却することを通知する。

○ INR(Interrupt by Receiver)

link番号を指定して，送信側プロセスに対して割り込み信号の如き働きをする特殊な信号を送ることを指示する。受信側プロセスのNCPから送信側NCPに送られる。

○ INS(Interrupt by Sender)

INRの送信側，受信側が入れ換ることをのぞいて機能は全く同じ。

○ ECO(Echo request)

8ビットのデータを持って，NCPに対してエコーを返すよう要求する。

○ ERP(Echo reply)

ECOに対する応答で，ECOの8ビットデータを持って返る。

○ ERR(Error detect)

エラー・コードとエラー・メッセージを指定して，コントロール・コマンド中にあやまりがあったことを通知する。

○ RST(Reset)

RSTを発信したHostと受信したHost間のコネクションを初期化することを指示する。

○ RRP(Reset reply)

RSTに対する応答で，初期化の終了を通知する。

○ NOP(No operation)

何の意味も持たないが，複数のコマンドを同一メッセージとしてNCPに送ることができるので，コマンドの切れ目に任意にそう入することができる。

(2) Cyclades

基本サービス

○ FL—LT(Flow letter)

発信，受信port—id，Credit—no，YR—REF，MY—REF，FR—NB/EOL，テキストより構成され，ユーザ・メッセージの送受信のために使用される。

Credit—noは，付加機能としてフロー制御を利用することを指定されたときに使われる。(YR—REF)＋1から(YR—REF)＋(credit—no)のメッセージを送ってよいことを意味する。

YR—REF，MY—REFはNCP間で利用するメッセージ番号であって，YR—REFは付加機能としてエラー制御を利用する場合に，Ackの番号として使用，MY—REFは送信メッセージの番号として利用。

FR—NBは，NCPにおいてメッセージをリアセンブルする場合に何番号のブロックかを指定する。

EOLはリアセンブルするメッセージの最終ブロックを意味する。

○ FL—TG(Flow telegram)

発信，受信port—idと16ビットの情報より構成される。割り込み信号などの情報伝送に利用される。

付加サービス

○ FL—INIT(Flow initialize)

発信，受信port—id相手側に送るメッセージの最大長および付加機能のサービス要求のパラメータより構成されコネクションの確立を要求する。同様のものが送られてきた場合に

コネクションは確立する。

付加機能としては，エラー制御，フロー制御，縮少（reduced）アドレスの指定ができる。縮少アドレスとは，コネクションに対して特定の名前をつける方法をいう。

○ FL—TERM（Flow terminate）

発信，受信port—idと終了理由をパラメータより構成され，コネクションの解放，拒否に利用される。

○ FL—ACK

発信，受信port—id，YR—REF，Credit—noをパラメータより構成され，エラー制御がが指定されたときのメッセージに対するAckおよびフロー制御のときのcreditを送るために利用される。

○ FL—ERROR

発信，受信port—idおよび64ビットのエラー情報より構成され，正常でないコマンドを受信したことを通知するために利用される。

○ PG－LT

縮少アドレス指定のメッセージ伝送（FL—LT）に利用。

○ PG—ACK

縮少アドレス指定のメッセージAck（FL—ACK）に利用。

○ PG—TG

縮少アドレス指定の割り込み信号（FL—TG）に利用。

(3) NPL

○ CALL

my reference, your reference（以上の2つのパラメータはすべてのコマンドに共通して必要ないので以降省略する），my send／receive capabilityを指示してコネクションの確立を要求する。your reference＝0のとき，loggerを示し，実際のyour referenceはHelloにより確定する。

○ HELLO

パラメータはCALLと同じであり，コネクションの確立したことを通知する。

○ GOODBYE

終了理由を指示してコネクションを解放する。CALLに対する拒否にも利用される。

○ UM DATA（User machine data）

メッセージ番号を持ち，コネクションに対して通常メッセージを送る。

○ INC DATA（Incomplete data）

メッセージ番号を持ち，コネクションに対して通常メッセージを送る。このコマンドは，

表2－7　Host—Host プロトコルの比較

	基本思想	プロセス間コミュニケーションの形態	メッセージの到着の確認	フロー制御		割り込み信号	エラーの通知	メッセージ長	その他
ARPA—net	プロセス間コミュニケーション	○パスの設定 ○パスは一方向	RFNM （目的地IMPが発信）	バッファ確保	コントロール・コマンドによりallocateを通知	INS，INRコマンドによる	ERRコマンドによる	8023ビット （1パケットのとき936ビット）	NCPのリセット機能
JIPNET	同上	同上	ACK （NCPが発信）	WABT	送信中止，再開のコントロール・コマンド	POSTコマンドによる。	POSTコマンドによる。	9216ビット （1パケットのとき1152ビット）	プロセスの発生
NPL	同上	○パスの設定 ○パスは両方向	なし メッセージ番号の不連続により紛失メッセージを検出し通知	バッファ確保	コントロール・コマンドによりallocateを通知	INTERROPTコマンドによる	ERROR，DROPコマンドによる。	2008ビット	ターミナルの制御
Cyclades	同上	同上 ただし，パスの設定を行なわない方式もある。	ACK （NCPが発信）	同上	ACKに便乗	FL—TGコマンドによる。	FL－ERRORコマンドによる。	1976ビット Hostのリアセンブル能力により1976×127ビットまで可	サービスに対して階層性がある

Terminal Processor（TP）がUMに対して送るもので，入力メッセージは完了しておらず，後続メッセージにより完了することを示す。

○ NEXT

パラメータとして，next 個数を持ち，この個数だけ，UM DATA，INC DATAを送ってよいことを指示する。

NEXTを受信した場合に，以前に保有していたnext 個数はリセットされ最新のものが有効となる。

○ INTERRUPT

TPからの interrupt 信号を意味する。

○ CANCEL

TPに対して出力メッセージをキャンセルさせる。

○ DROP

メッセージギャップの個数をパラメータとして持ち，直前に受け取ったメッセージと新らしく受け取ったもののメッセージ番号の間が不連続であったため，メッセージが紛失したことを送信側に知らせる。

○ ERROR

不完全なパケットを受信したことを知らせる。

2.6.2 比較検討

前項においては，4つのコンピュータ・ネットワークのH－Hについて概要をのべたが，この項では，それぞれのH－Hについての問題点などを比較検討してみることとする。

A プロセスあるいは利用者の識別

ARPA－net，Cyclades，JIPNETは，形態の違いはあれ全く同じ考え方にたっている。すなわちport－idを利用するわけであるが，これの一部にユーザ番号を入れることによってネットワーク内に共通する名前とし，同一の名前が他のユーザでは使用できない方式となっている。

この利点は，ユーザが自分の責任においてport－idを定義すればネットワーク全体にまたがってプロセスを分散させ，それぞれの仮想的ネットワークの構成が容易にできることにある。この意味において汎用性はきわめて大きいということができる。

NPLの方法では，各Hostに存在するnetwork control executiveがLogical channelを一括管理しているために，汎用性を持たせるには特別のはからいをする必要がある。

すなわち，メッセージ交換をしたいとする2つのプロセスは，他方のプロセスのLogical channelを知らないばかりでなく，自分のLogical channelも，前もって割り付けておかなければならないという理由からこのようなことになる。

このために，loggerという仲介役をもうけ，最初にloggerに対してメッセージ交換をおこない自分の意図を知らせて，相手のLogical channel番号を知る。こののち実際のメッセージ交換にはいるわけである。

ここで問題となるのは，loggerに，メッセージ交換をしたいプロセスの名前を登録しておく必要があることであり，この意味において新らしいサービスを追加する場合にはloggerの修正が必要となってくる。

NPLの思想としては，ユーザに対して自由にプロセス間コミュニケーションをさせるということではなく，各HOSTに対して用意されているサービス機能（たとえばBASIC，文献検索，テキスト・エディタ，ファイル・ストア）を利用させるということを基本において設計した為このような形になったと考えるべきであろう。

B　コントロール・コマンド

コントロール・コマンドにおいて必要となる機能は，4つのネットワークのH—Hにおいて形態の違いはあるがすべて満足されている。しかしながら若干問題のある方法を採用しているものもあるのでこれを対象にコメントをつけておく。

a．メッセージ交換手順

メッセージ交換手順としては，コネクションの設定方式とWaldenの提案しているメッセージ交換方式〔90〕，データ・グラム方式があるが，4つのH—Hともにコネクションの設定方式をとっている。

(1)　単方向あるいは両方向コネクション

コネクションの設定方式をとった場合に，1つのコネクションが単方向のデータのみをあつかうか，両方向であるべきかという問題点が発生する。

ARPA—net，JIPNETは単方向であり，Cyclades，NPLは両方向である。

これを決めるポイントは，プロセス間コミュニケーションにおいて単方向だけのコネクションが，どの程度利用されるかということである。実際問題としては，プロセス間コミュニケーションのほとんどは両方向（したがって単方向の場合は2つのコネクションを設定する）になっており，両方向にした方がコネクションの設定にともなうオーバーヘッドが減少すると考えられる。ここで注意しなければならないのは，フロー制御としてコネクションごとにバッファを予約しておく方式をとっている場合には単方向しか使わないコネクションに対しては，バッファが活用されないことになるので，それに対する考慮がほしい。

(2)　メッセージ長の指定

コネクションを通るメッセージの最大長をコネクション設定要求をするのに指定する必要があるかどうかという問題である。

CycladesとJIPNETにおいては，メッセージ最大長を指定するようになっている。この2つのH—Hにおいて指定させている理由は，受信NCPにおいて受信用バッファの割り

あてがメッセージを単位として考えている点にある。ＪＩＰＮＥＴにおいては，これに加えて，多重パケットメッセージと単一パケットメッセージが別々にシーケンシングされるために，シーケンシングを保障するためにも必要となる。

メッセージ最大長を指定しない場合に発生する問題点は，フロー制御において詳しく述べるのでここでは省略する。

結論をいえば，メッセージ長の指定はあった方が良いと考えられる。しかしながら，両方向のコネクションの場合には，Cyclades のように違った長さを指定できるようにすべきである。

ついでながら，メッセージ長＝０の指定ができるようにしておけば，両方向のコネクションにおいて，一方を全く利用しないことを示すようにすることもできる。

(3) その他

ＪＩＰＮＥＴのＨ－Ｈにおいては，コネクションの確立要求，確立応答のコマンドがあるがこのような形にすると図２－48のようにコマンドのすれ違いという現象が起こることがあるので，ARPA－net，Cyclades のように要求と応答のコマンドが同一になっていた方がよいであろう。

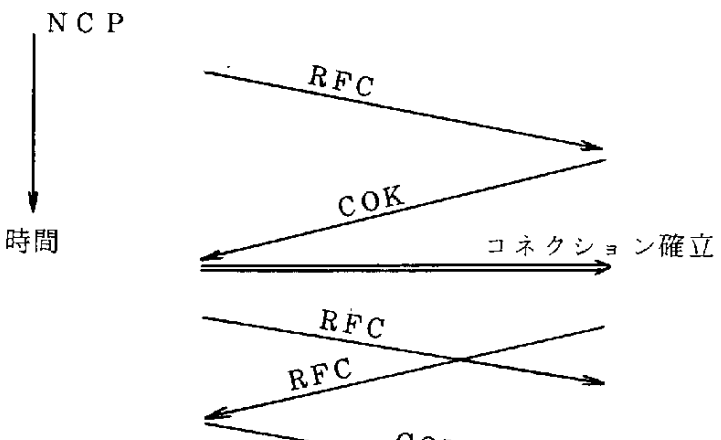


図２－48　コマンドのすれ違い

ＪＩＰＮＥＴのＨ－Ｈにおいては，コネクションの確立応答コマンドにおいてＲＦＣに指定された自分側のport－idに代えて，新らしいport－idによってコネクションを確立させる方式もサービスされている。

この方法は，高位プロトコルのコネクションを設定する場合には有効になる。すなわち，この方法をとっていないARPA－net の場合にはどのport－id に対してコネクションを設

定するのかという情報を受け取るためのコネクションを最初に設定する必要があり，オーバーヘッドが多くなっている。

b．フロー制御

フロー制御は，受信側Hostにおいて，メッセージを処理する能力以上のものが流入するのを防止するために必要なものである。

この手法としては，以下にあげるようなものがある。

① バッファ確保方式

○バッファ要求送信方式〔26〕

○受信側準備方式〔54〕

② クレート方式〔54〕

③ 受信命令待ち方式〔54〕

④ Window方式〔100〕

⑤ Wait before transmit（WABT）方式

ARPA−net，Cyclades，NPLでは，受信側準備方式を用いており，JIPNETではWABT方式をとっている。

(1) WABT方式の問題点

JIPNETにおいては，コネクション単位にこの方式を採用しているが，メッセージの再送に関する問題点がある。すなわち，受信側NCPはバッファがなくなると当該コネクションに対する送信一時中止を送り，バッファに余裕ができるのを待って送信再開を送る。

送信側NCPにおいては，送信一時中止を受け取ると，当該コネクションに対して送信再開がくるまで，メッセージの送信を中止している。

このような方法では，受信側の送信一時中止が送信側NCPに到着するまでに時間遅れがあるために図２−49に示すように途中のメッセージ（Ｍ４，Ｍ５）が廃棄されて，そのあとに受け取るべきメッセージ（Ｍ６）が受け取られる現象が起こる。

また，TIPの端末によってTSSを利用していた場合に起こるものであるが，計算結果が多量になったが，出力装置の印字速度が遅いために出力メッセージがTIP内にたまってしまう。もともとTIPバッファは少ないために，送信一時中止，送信再開がひんぱんにでることになる。

この２つの問題点を解決するためには，より高位のプロトコルにおいてプロセス同志がメッセージを受信したので，次に送ってよいという確認メッセージを授受するようにしなければならない。

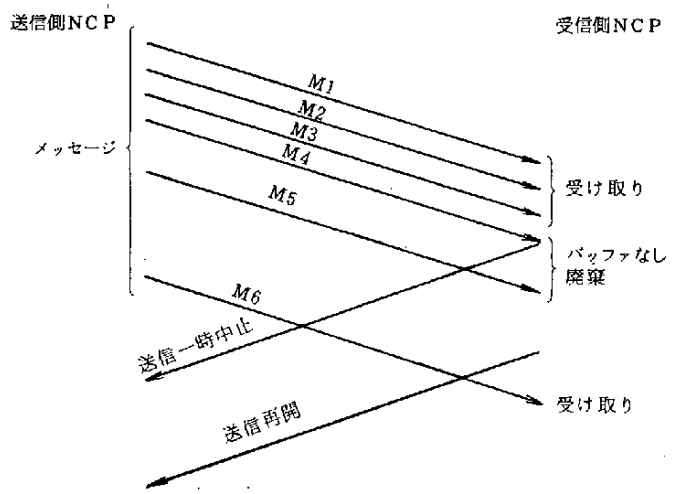


図２－49　WABT方式の問題点

(2)　ARPA—netの問題点

ARPA—netのフロー制御の方法は（図２－50）に示すとおりである。この方法では，JIPNETにおいて起ったような問題は発生しない。しかしながらコネクション設定時にメッセージ長を指定しないために次のような問題点が起こる。

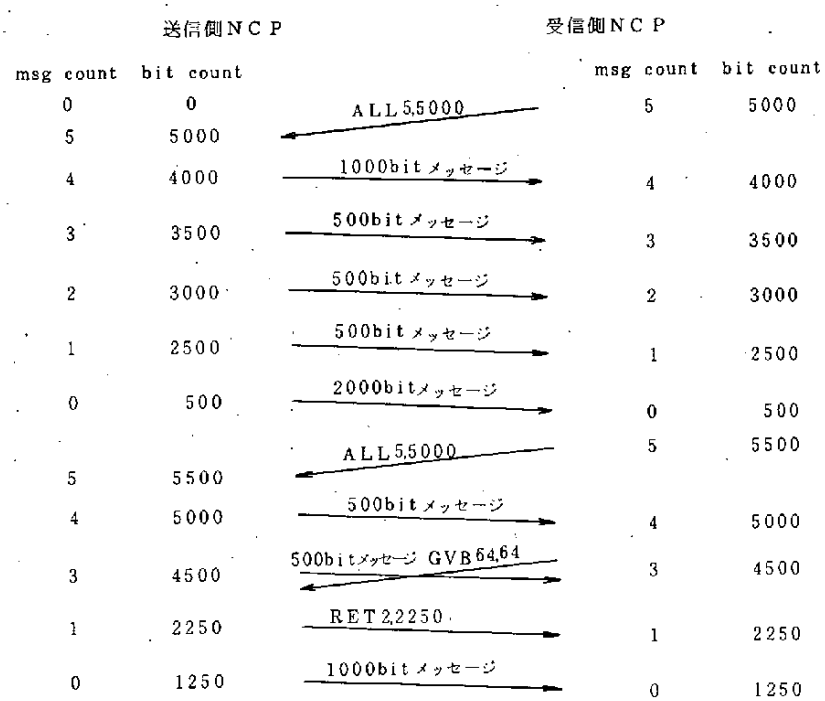


図２－50　ARPA—netのフロー制御

受信側NCPが5000ビットのバッファを確保して送信側NCPに通知したとする。送信側は，8000ビットメッセージを送りたい場合には，受信側では5000ビットのバッファしか確保していないために，メッセージを送ることができない。受信側NCPに対してバッファの確保を要求するコマンドもないために（図2－51）で示すように，0ビットメッセージを送ることによりmsg countを0にしてバッファ確保をうながす方法をとらなければならない。

このような複雑さをのがれるためには，より高位のプロトコルにおいてメッセージをリアセンブルする機能を追加しておくか，メッセージをストリームとしてあつかい，メッセージの区切り記号をもうける必要性がある。

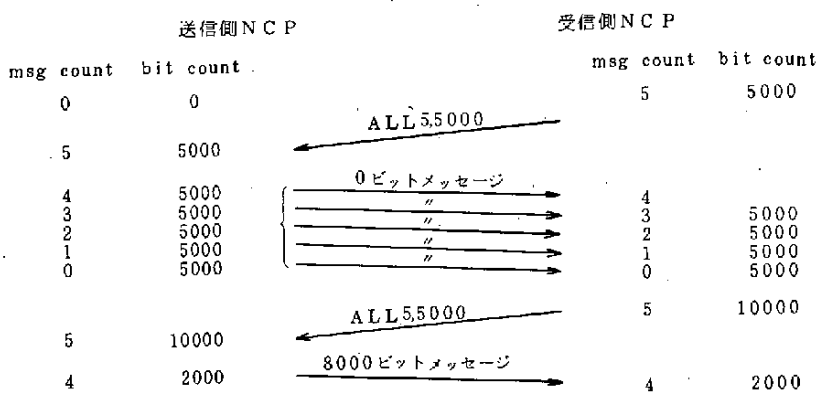


図2－51　ARPA—netのフロー制御の問題点

(3)　NPLの問題点

バッファの確保を通知するNEXTコマンドに問題がある。すなわち，NEXTのパラメータに指定されたnext個数だけのデータを送ることができることを通知するものであるが，NEXTを受け取ると，以前に保持していたnext個数がリセットされ，新らしく送られてきた方が有効になることから発生するものである。

これは，（図2－52）で示すように，NEXTの送受信タイミングによって受信側で送ったと考えているnext個数と，送信側の保持しているnext個数が違ってくることである。

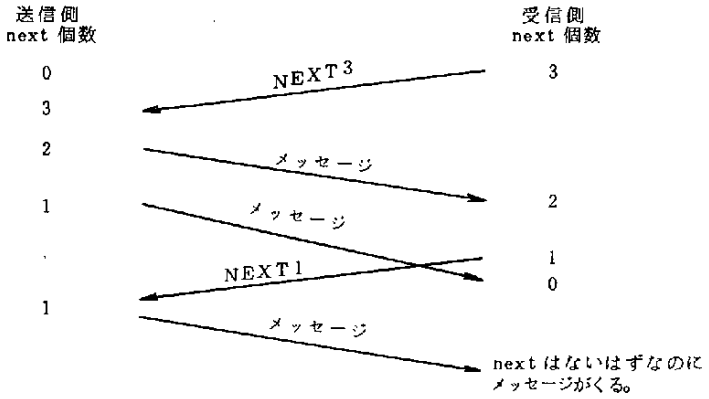


図２－52　NPLのフロー制御の問題点

C　メッセージの誤り

NCP間で伝送されるメッセージの誤り率がどの程度であるかということはコンピュータ・ネットワークにおいて，機能分担をどのようにするかということにかかわる問題であり，高い見地から判断する必要がある。

ARPA—net，ＪIPNETにおいては 10^{-10} 程度は保障され，Cyclades の場合には 10^{-4} 程度である。

この誤り率がある域値を越える場合には，NCPにおいてエラー制御をおこなわねばならぬがCyclades はその例である。

但しCyclades においておこなっているエラー制御は，NCP間でやりとりするすべてのメッセージではなく，ユーザ間でやりとりするメッセージに対してだけおこなっている。

このために，コネクションの確立要求（FL—INIT），割り込み信号（FL—TG）などは 10^{-4} の確率で重複，紛失することになり，NCPのサービスが不十分になる問題点がでてくると思われる。

D　機能面での検討

Cyclades において基本機能と付加機能という形にＨ—Ｈを階層化した背景について考察する必要がある。

Cyclades の付加機能および他の３つのＨ—Ｈは，すべてコネクション本位にメッセージ交換をおこなう方法をとっている。このように，コネクションを利用するメッセージ交換方式に問題がないかどうかを検討してみよう。

コネクションは２つのプロセス間に最低１本設定されるものである。このためにｎ個のプロセスが相互に会話する場合にはｎ×（ｎ－１）本のコネクションが必要となる。このような使用形

態は十分に考えられるものである。

すなわち，すべてのHOSTのプロセスからデータ収集をするプロセスを作った場合などである。

このような場合には，コネクションの個数を多く使用して，NCPの用意しているコネクション・テーブルを全部使ってしまい，他のものの利用ができなくなってしまう可能性がある。

これを避けるためには，コネクションという概念をとりはずしてしまう必要がある。この結果としてできたのがCyclades の基本機能であろう。同様な機能は，ユーザに対して一般には公開されていないが，ARPA—net においても，NCPのコントロール・リンク，UCLAにおいてネットワークの性能測定に利用しているリンクという形で使用されている。

このような機能は，非常に有効なものであるが，フロー制御をどのようにしておこなうかという問題点を残したままユーザに公開してもよいものかどうかという問題点が残る。

3. サブネット

3. サブネット

3.1 サブネットの論理設計

コンピュータ・ネットワークにおいて最も基本的な事項は，HOSTコンピュータ間の通信を可能にすることである。少数のコンピュータを結合する場合には，HOSTコンピュータを直接結合することは，色々な意味において有効なことが多い。しかしながらHOSTコンピュータの数が多くなると，HOST内において蓄積交換をおこなわなければならなくなり，HOSTの仕事量が増大し，HOSTに負荷をかけるばかりではなく，通信機能の信頼性がHOSTに従属し，高信頼性を実現するのが困難になってくる。

このような理由から，JIPNETにおいては，コミュニケーション・サブシステムと，HOSTコンピュータを明確に分離して，HOSTコンピュータはコミュニケーション・サブシステムであるサブネットに結合することにより，いかなるHOSTとも通信ができる方式をとった。

JIPNETのサブネットは，高い信頼性をもつパケット交換ネットワークであり，次のような基準にたって作られている。

3.1.1 設計目標

サブネットの設計においては次の4つの点に留意した。

(1) メッセージ伝送の誤り率を低くするだけではなくサブネットの要素に障害が起こっても全体的機能は失なわれないこと。

(2) 会話形処理を可能にするために，短いメッセージに対して十分な応答特性が得られると共に大容量のデータ伝送に対しても十分なスループットが得られること。

(3) ネットワークの拡張性を考慮したソフトウエアであること。

(4) 同一IMPに結合されたHOST間では，特に高速な伝送を保証すること。

A．高信頼性

サブネットの信頼性はネットワーク自体の信頼性を左右する重要な要素である。信頼性を決定づける要素は2つあり，1つはメッセージ伝送の誤り率の減少であり，もう1つはフェイル・ソフト機能の実現である。

サブネットに於て，メッセージ伝送の誤り率を減少させることによって，HOSTまたはFront end processorはHOST－HOST間における伝送エラーの検出や，そのリカバリに大きな負荷を費す必要がなくなり，少なくともこの点でのHOSTの負荷を減少させることが可能とな

る。ＪＩＰＮＥＴに於てはメッセージ伝送のエラー・チェックは完全にサブネットの分担としている。

同期伝送に於て，一般的に使用される16 bitsのCyclic check sumのＩＳＯ standardに於ては，ビットあたりの誤り率は10^{-10}以下と言われており，ＪＩＰＮＥＴに於てもＣＲＣを採用し，これと同程度の誤り率を目標としている。

分散型ネットワークは一般に集中型のものより信頼性が高いといわれ，そのフェイル・ソフト機能としては，いづれかのHOSTの障害はもとより，一部のＩＭＰや回線の障害時に於ても，それを切り離した形でサブネットは活動を続けることが要求されている。

ＪＩＰＮＥＴに於ては，何等かのサブネット要素の障害は単にサブネットトポロジーの変化とみなし，その障害の影響を最小限にとどめるように配慮している。

B．メッセージの応答性とスループット

サブネットの効率に対しては2種類の要求がある。1つは会話型ジョブに対する即応性であり，もう1つは大量のファイル転送に対する高スループットである。

両者に共通した条件としてはまずメッセージ伝送の速度が早いという事であろう。ネットワークに於ける会話型ジョブのために充分な応答時間を得るには短かいメッセージの伝送の遅延は少なくとも0.5 sec以下であることが望ましい。

一方，大量のファイル転送に於ては，メッセージ伝送の速度を基本的に早くするばかりでなく，より長いメッセージを扱うことによって処理のオーバヘットの減少や，回線の利用率を高める工夫をせねばならない。

ＪＩＰＮＥＴは長短両メッセージを扱う機能を持ち，それぞれのメッセージ長は差しあたり，現在144バイト以下と1152バイトとしている。メッセージ長の決定は既存のいくつかのHOSTのタイムシェアリングやリモートバッチ処理のメッセージサイズにもとづいたものである。

C．拡張性

HOSTとＩＭＰの間の負荷分担はネットワーク設計における難しい問題の1つである。これはＩＭＰのプログラムが，HOSTに依存する部分を持つべきか否かという問題と関連する。

拡張性の立場から言えば，ＩＭＰ内のプログラムは，それにつながるHOSTとは無関係なものであることが望ましい。

ＪＩＰＮＥＴにおけるＩＭＰのプログラムはHOSTとは無関係なものとし，且つネットワークの拡張やトポロジの変化にも出来るだけ左右されぬものとした。例えばＩＭＰやHOSTの数の増加に対して，たゞ数語の定数の変更のみを行えばよい。

D．同一ＩＭＰ結合HOST間の高速伝送

同一ＩＭＰに結合されたHOSTは，複数のHOSTコンピュータを直接結合したコンピュータ・コンプレックスとして使用するという目的を兼ねることも可能となる。実際ＪＩＰＮＥＴにお

いても1つのＩＭＰにグラフィック・ディスプレイを持つコンピュータと，大容量ファイル，高速演算を持つコンピュータとを結合し，コンプレックス・システムとしてインタラクティブ・グラフィック・ジョブに利用したいという要求がある。

このような形に結合されたHOST間では伝送手順を変えてより高速な伝送をおこなうことも可能ではあるが，ネットワークの全体的な見地からデータ伝送手順は，これらのHOST間も他のＩＭＰに結合されたHOSTと全く同様とし，且つこの様な場合にも対処し得る高速伝送効率を得るべく配慮している。

3.1.2　機能概要

A．パケットとメッセージの伝送

図３－１に示すように発信HOSTからサブネットに流入するメッセージは先づ目的地ＩＭＰでパケット化され，各パケット毎に任意のルートを通って目的地ＩＭＰに到着し，そこで再びメッセージにリアセンブル（ reassemble ）され受信HOSTにおくられる。

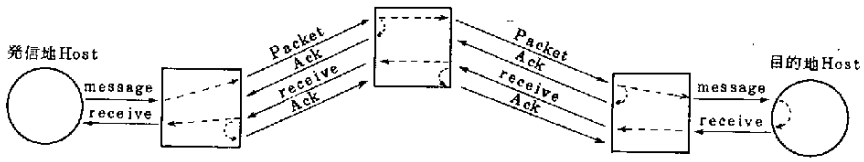


図３－１　メッセージ，パケットの伝送手順

２つの隣接ＩＭＰ間ではＩＭＰ－ＩＭＰプロトコルにしたがいパケットの受信確認信号としてＡＣＫを返し，また発信HOSTと受信HOST間ではメッセージの受信確認として receive が返送される。これらの確認信号を受けると発信側のＩＭＰまたはHOSTはそれぞれのコピーを棄却することが出来る。ある一定的間内にそれらの確認信号が返送されぬ場合は原則として夫々のコピーを再送する。

各パケット毎にＡＣＫを返送するコントロールトラフィックの増大をさけるため，ＡＣＫは数パケット分まとめて，逆方向のトラフィックに便乗して返される。逆方向のトラフィックがない場合はＡＣＫのみがヌルパケットとして返される。これらの事項は，IMP／IMPプロトコル，HOST／IMPプロトコルとしてきちんと定められており，詳しくはそこを参照されたい。

２つのＩＭＰ間の全二重回線は両方向の夫々４つの論理的なチャネルに分けられており，１つのパケットの伝送にはその１チャネルが割り当てられる。チャネル数の決定は出来るだけ回線の使用効率を高めるという目的にもとづいており，その要素は次のようなものである。

伝播遅れ $t\alpha(\ell)$：あるＩＭＰで１パケットを送出してから，隣のＩＭＰがその最初のビットを受けとるまでの時間であり，これは両ＩＭＰ間の距離に比例する。

伝送遅れ tm(s) ：あるＩＭＰでパケットの最初のビットを受信してから，その最後のビットを受信するまでの間の時間で，これはパケット長に比例し，回線速度に反比例する。

伝送処理遅れ tp ： modemキューに入っているパケットが，出力状態になってから実際に出力されるまでの時間。

ＡＣＫの処理遅れ t_A ：パケットのACKを返すため逆方向のトラフィックに便乗させてから実際に，出力させるまでの時間。

こゝで

$$T = tp + t_A + 2\ t\alpha(\ell) + 2\ tm(s)$$

とすれば，Tはパケットの出力要求が出てからＡＣＫを受け取るまでの時間である。Tはまた1パケットの伝送によりチャネルが占有される時間でもある。

そこで必要なチャネル数は次の式で表わされる。

$$n = \frac{T}{tm(s)} = \frac{tp + t_A + 2\ t\alpha(\ell)}{tm(s)} + 2$$

回線速度を48Ｋbit／sec とすれば，tm(s)は100 bitあたり2ｍsec，td(ℓ)は100 km あたり0.5ｍsec，tpおよびtAはそれぞれ1ｍsec 以下である。この様な前提により計算を行なうと距離とパケット長に対する必要なチャネル数は図３－２に示すようになる。

この結果からＪIPNET では4チャネルを採用することとした。

パケット長 / IMP距離	200 bit	300 bit	500 bit	1000 bit
100 km	2.8	2.5	2.3	2.2
500 km	3.8	3.1	2.7	2.4
800 km	4.5	3.7	3.0	2.5

図３－２

B．ルーティング

ＪIPNET ではルーティング手法として原理的には Shortest Queue ＋ Bias（ＳＱ＋Ｂ）を使用している。即ち各ＩＭＰに於て，各回線の出力キューの長さと，静的なバイアス（Bias）項の和を回線毎に比較し，その最少のものを選んでパケットを出力する。この手法は固定ルーティングと適応ルーティングの両方の利点を持つと考えられる。

図３－３に示すようにバイアス項は出力回線をｉと目的地ＩＭＰをｊとを行，列とする表になっており，各要素Bias（ｉ，ｊ）はパケットが，そのＩＭＰから出力回線ｉを経て目的地ＩＭＰｊに到着するまでの推定時間を示す。

通常のＳＱ＋Ｂ法では，バイアス項はサブネットトポロジにしたがって予め計算された固定的なものであり，若しバイアス項の荷重の方がキューの長さの項の荷重より高ければ，これは固定ルーティングに近づき逆の荷重にすれば適応ルーティングに近づくこととなる。

行先 回線	1	2	3	4	…… 16
1	3 1	1 2	3 5		
2	2 5	9	4 2		
3	1 8	3 8	2 7		

図３－３　Biasテーブル

またバイアス項は，通常は各ＩＭＰの無負荷状態を前提として計算されるが，ＪＩＰＮＥＴでは各ＩＭＰに出力キューが１個づゝあるという仮定でバイアス項を算出し，バイアス項に若干多くの重みをかけている。

静的なBias Table はサブネットの一部の障害によるＩＭＰの数やトポロジの変化に対する適応性がないため，その点を改良する手段として，ＪＩＰＮＥＴではサブネットにおける重大な変化，即ちＩＭＰや回線のup／down に際して動的にBias Table の更新をおこなうことにしている。

Bias tableは次のようなアルゴリズムによってジェネレートされる。先づＩＭＰの初期状態に於てはすべてのBias（ｉ，ｊ）は∞の値を持っている。回線ｉがup したという知らせにより，回線ｉの他端のＩＭＰ名をｊとすれば，Bias（ｉ，ｊ）にはC(i)が入る。こゝでC(i)は図３－４に示すように回線速度によってことなる回線の容量を表わす定数である。逆に回線ｉがdownしたことがしらされるとすべてのｊに対しBias（ｉ，ｊ）は∞にされる。

回　線　速　度	C(i)
4 8 K bps	1
9.6 K bps	5
4.8 K bps	10
2.4 K bps	20

図３－４　各回線速度とC(i)

各ＩＭＰに於てBias table に何等かの変化があると，Update Vectorも更新される。

Update Vectorとは他のすべてのＩＭＰに対するその時点での最少伝送時間を示すもので，Bias table をもとに次の式で計算される。

$$\mathrm{Update}(j) = \min_{\text{all } i} \{ \mathrm{Bias}(i, j) + C(i) * \mathrm{Const} \} \qquad \text{for all } j$$

こゝでｉはline 名

ｊは目的地ＩＭＰのＩＭＰ名

Constは１or２の値をとり得る定数であるが現在は１としている。

Bias table の更新によってこのUpdate Vectore も変化を起したならば，このVectorはそのＩＭＰにつながるすべての回線に送り出される。

隣接ＩＭＰからline ｉを通してUpdate Vector を受け取ったＩＭＰは次の式によりBias tableが変化するか否かを試みる。

$$\text{Bias}_{\text{new}}(i, j) = MIN\{\text{Bias}_{\text{old}}(i, j), \text{Update}(j) + C(i)\} \qquad \text{for all } j$$

その結果Bias table に変化が生じればこのＩＭＰに於ても変化したすべての目的地ｊに対して，Update Vectorも再計算され，Update Vectorに変化が生じれば，更に次の隣接ＩＭＰへ更新情報が伝達され，変化が生じなければそこで伝達が停止される。

以上の様なアルゴリズムにより作成，更新されるBias table と各回線の出力キューの長さとにより目的地ｊに対する出力回線の選択は次の式によりおこなわれる

$$L(j) = \underset{\text{all } i}{MIN}\{\text{Bias}(i, j) + C(i) * Q(i)\}$$

こゝでL(j)はそれを通じてパケットが出力されるline№，ｉはこのＩＭＰに結合されているline№，Q(i)はline ｉの出力キューの長さである。

若しBias（ｉ，ｊ）がすべてのｉに対して∞である場合には，目的地ｊに対するパケットはこのＩＭＰで棄却される。

C．障害の検出

ＩＭＰと回線の障害およびサブネットのディスコネクション（disconnection）状態の検出は次のようにおこなわれる。

ＩＭＰ障害　：　一般にＩＭＰの障害は隣接ＩＭＰによってのみ検出される。連続した20パケットの伝送に対して何等の応答もない場合，その隣のＩＭＰは障害とみなされる。この場合応答のないパケットに対しては200ｍsec毎に再伝送がおこなわれるので，この障害を検出するには最大４秒かゝることとなる。ＩＭＰの障害は，それにつながるすべての回線の障害とみなして処理される。

回線障害　：　回線障害はハードウエア割込みによって検出される。この割込みは隣接ＩＭＰのmodemからのcarrier がとだえた時，あるいはmodem が故障した場合におこる。回線の障害には２種類あり，その１つは数ｍsecの瞬断であり，もう１つが本当の障害である。そこで割込みがおこってから約300ｍsec後にcarrier が来ているかどうかを調べてこの２つを区別している。

回線障害の検出によって，ルーティングのためのBias table が更新され，そのＩＭＰ内で出力を待っているパケットや，ＡＣＫの返送を待っているパケットの再ルーティングが行われる。またサブネットの結合状態を示すコネクション・マトリックス（Connection matrix）の更新

がおこなわれ，それを隣接IMPに送ることにより，回線の障害状態をサブネット全体にしらせることとなる。

コネクション・マトリックスとディスコネクション：コネクション・マトリックスはサブネットの結合状態を保持している表であり，図３－５に示すように16×16のビット行列である。この行列の各要素は，ＩＭＰ間の結合の状態を示し，１ならば行と列の両ＩＭＰ間が結合され，０ならば結合されていない。そしてこの行列はＩＭＰや回線のup／downによって更新される。

このマトリックスはサブネットのディスコネクション状態を検出するために使われる。

$$M = \begin{array}{c} \begin{array}{cccccc} IMP\ 1 & IMP\ 2 & IMP\ 3 & \cdots\cdots & IMP\ 16 & \end{array} \\ \left[\begin{array}{cccccc} 0 & 1 & 0 & \cdots\cdots & 1 & \\ 1 & 0 & 1 & \cdots\cdots & 0 & \\ 0 & 1 & 0 & \cdots\cdots & 0 & \\ \vdots & & & & & \\ 1 & 0 & 0 & \cdots\cdots & 0 & \end{array}\right] \end{array} \begin{array}{l} IMP\ 1 \\ IMP\ 2 \\ IMP\ 3 \\ \vdots \\ IMP\ 16 \end{array}$$

図３－５　Connection matrix

このビット行列Mの各要素をm_{ij}とし，Mに対し，次の操作を定義する。

M×Mは次の要素Aijを持つ新しい行列を発生するオペレーションとする。

$$A_{ij} = (m_{i1} \cap m_{1j}) \cup (m_{i2} \cap m_{2j}) \cup \cdots \cup (m_{i16} \cap m_{16j})$$

M1｜M2 は次の要素A_{ij}を持つ新しい行列を発生するオペレーションとする。

$$A_{ij} = m1_{ij} \cup m2_{ij}$$

ここで$m1_{ij}$と$m2_{ij}$は夫々M1とM2の要素である。

そこでコネクション・マトリックスMによる行列

$$M_2 = (M \times M) \mid M$$

の各要素をm^2_{ij}とすればm^2_{ij}はＩＭＰ$_i$とＩＭＰ$_j$が２ホップ(hop)以内で結合されているか否かをあらわす。

更に，　$M_4 = M_2 \times M_2$，$M_8 = M_4 \times M_4$，$M_{16} = M_8 \times M_8$ を計算すれば，M_{16}の各要素m^{16}_{ij}はIMP$_i$とjの間に16ホップ以内で到達する経路があるか否かを示す（附録参照）。ＪＩＰＮＥＴの最大ＩＭＰ数は16であるので，ある要素m^{16}_{ij}が０であればIMP$_i$とＩＭＰ$_j$とは互いにディスコネクション状態である事を示すこととなる。

このように，ＩＭＰや回線の障害により各ＩＭＰでコネクション・マトリックスを更新し，自分のＩＭＰに対して，どのＩＭＰがディスコネクションであるかを検出するには，各ＩＭＰで４回のビット行列演算を行なえばよい。この計算に要する時間は約３ｍsecである。ディスコネクションＩＭＰに対するメッセージをHOSTから受け取った場合，ＩＭＰは相手がディスコネクシ

ョンである旨の疑似のreceiveをHOSTに返す。またディスコネクション状態になったＩＭＰのBias項には∞が入れられる。

D．フローコントロール

サブネットに於けるメッセージのラウンドトリップタイム（round trip time）を短かくし，且つスループット（ throughput ）を高めるため，フローコントロールによりサブネット内のメッセージの流れを規制する。

ＪＩＰNETにおけるフローコントロールはARPANET に於けるものと類似である。即ち，発信地ＩＭＰと目的地ＩＭＰの間にパイプ（ pipe ）と呼ぶ論理的なパスを張り，そのパイプの容量を制限することによりサブネット内のメッセージ量をコントロールする。

ＩＭＰ数をｎとするとパイプの本数はサブネット全体でｎ（ｎ－１）×２本である。

メッセージにはsingle packet message（ 144 bytes以下）とmulti packets message（ 1152 bytes以下）とがあり，single packet用のパイプ，Ｓ－pipe と呼ぶ，とmulti packets用のパイプ，Ｌ－pipeと呼ぶ，とをもうけ，両者はそれぞれ独立にコントロールされている。こゝで２種類のパイプを創った理由は，長いメッセージの伝送によるオーバヘッドの影響を短いメッセージの伝送に与えないためである。

Ｓ－pipe の容量は４である。即ち，２つのＩＭＰのペア（ pair ）の間で，４パケットまでが同時に伝送を許される。そして５番目のメッセージは，最初のメッセージの receive が返って来るまで送出が待たされる。

Multi packets message はバッファ確保のためのパケットが先づ目的地に伝送され，それにより目的地ＩＭＰはそのmulti packets messageのリアセンブルのためのバッファ領域を割当てる。メッセージ本体はパケット化され，この予約の確認を得て初めて発信ＨＯＳＴより送出される。

Ｌ－pipe の容量は現在のところ１としている。

E．シークェンシングとエラーコントロール

伝送されたメッセージはシークェンシングによって正しい順序で受信HOSTに届けられる。

シークェンシングはＳ－pipeとＬ－pipe と夫々独立に管理されている。２つのHOST間の１つのコネクション（ connection ）には，Ｓ－pipeかＬ－pipe か一方のパイプが割当てられるため，２つのパイプは夫々独立に管理し得る。

各メッセージにはパイプ毎に０から255の番号がラウンディング（ rounding ）に付けられ，目的地ＩＭＰはこの番号によりシークェンシングを行ない，受信HOSTには正しい順序でメッセージを送り込む。

パイプの容量の決定は，目的地ＩＭＰに於て，より若い番号のメッセージの到着を待つ既着のメッセージのために，待ち合わせ用のバッファが何個必要であるかというところからＩＭＰのメ

モリ容量の制約をうける。例えばS－pipe の容量を4とすればシークェンシング用のバッファは最大4パケット分でよい。

シークェンシングはフローコントロールと密接な関係があるばかりでなくメッセージ伝送におけるエラー検出に対しても大きな役割を果たしている。

例えば，シークェンシングのためにメッセージ番号をチェックすることにより，喪失メッセージ（missing message）や重複メッセージ（duplicate message）を検出することが出来ると同時にreceiveの紛失も発見し得る。receiveには対応するメッセージ番号がつけられているため送信側ではどのメッセージのreceiveが返送されていないかをチェックすることができる。発信地IMPは各メッセージの送出毎にタイマー（timer）をセットし，4 sec経ってもreceiveが返送されてこないメッセージに対しては，そのメッセージ番号を付して質問パケット（inquiry packet）を送る。質問パケットを受けた目的地IMPは，若しメッセージをすでに受け取っていればreceiveが紛失したとみなし，receiveを再送し，まだメッセージを受け取っていなければメッセージ自体の再伝送を要請する。

しかしながら，メッセージの紛失は非常にまれな場合と考えてよい。これがおこるのは，中継IMPがあるパケットを受けとり，それのACKを返送した後，次のIMPにそのパケットを送り出すまでのわずかな間にdownするか，まわりの回線の障害によってディスコネクト状態に陥いるかした場合のみである。

F．コンジェスションとロックアップ

サブネット内に多量のパケットが流入することはフローコントロールによっておさえられているが，1つのIMPに沢山のパケットが集中する場合にコンジェスション（congestion）がおこり，ひいてはロックアップ（lock up）の状態が起る可能性がある。

サブネットに於て，処理能力以上のパケットはIMPのバッファ内に残留しているので，ロックアップはバッファの割当てアルゴリズムにより避けることが出来る。

JIPNETに於て考慮したロックアップは次の4種類である。

(1) Store & Forward ロックアップ

(2) Reassemble ロックアップ

(3) Sequencing ロックアップ

(4) HOST／IMP ロックアップ

a．Store & Forward ロックアップ

このロックアップは各IMPが隣のIMPの状態を全く知らずにパケットを送ってくるために起るロックアップである。図3－6に示すようにIMPiとjの間で両方向のパケット伝送があり，IMPiのバッファはすでに全部占有され，IMPjからのパケットをもうそれ以上受ける事が不可能な状態とする。このことは，IMPiはすでにjに送ったパケットに対する

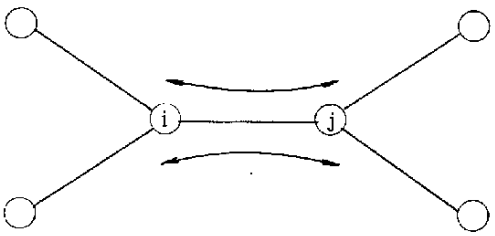


図３－６　Store & Forward　ロックアップ

ＡＣＫも受け取れないという事を意味する。ＡＣＫが返ってくれば，それに対応するコピーを捨てあきバッファを得る事が可能となるが，それもかなわずＩＭＰ$_i$内のバッファは一向に解放されず，両者が硬着状態に陥いりロックアップが発生する。

この現象を避けるためには，次の事項が満足されねばならない。

＊ＡＣＫは必ず受け取る事が出来るようにする。

＊少なくとも１個のバッファはstore & forwardのために確保しておくこと。

ＪＩＰＮＥＴに於ては次の手続きでこのロックアップを防いでいる。

(1) 各ＩＭＰの初期状態セット時に，そのＩＭＰに結合されたすべての回線に対し，夫々１個づゝの入力バッファを割当てる。

(2) パケットを受け取った時，若し伝送エラーがあれば棄却され，なければ，先づ到着しているＡＣＫを調べそれによって保存しているコピーのいづれかが不用になればこれを棄却し，あきバッファを得る。

(3) 次にstore & forward バッファがあればそれを，なければ通常のバッファをプール（pool）から得て，次の入力にそなえる。

(4) 上の場合でバッファがない場合，そのパケットは棄却され，そのパケットが使用していたバッファが次の入力のために再利用される。

(5) 残りバッファが20以下の場合は，自ＩＭＰを目的地としている通常パケット（コントロールパケットではない）は受け取られない。

(6) フリーバッファが存在する時は少なくとも１個のstore & forward バッファを必ず確保しておく。

b．Reassemble　ロックアップ

このロックアップはmulti packets message の伝送の際，目的地ＩＭＰに於てそのメッセージの１部のパケットが未着のうちにバッファがすでに一杯となり，reassembleが不可能のためにおこるものである。

前述のように，multi packets message は，目的地ＩＭＰに於けるリアセンブル内のバッファ領域の予約が出来たという確認を得た後，始めて発信側でパケットに分解されてサブネットに流出される。これによりリアセンブルロックアップは完全に防ぐことが出来る。

c．Sequencing　ロックアップ

このロックアップは幾つかのＩＭＰから沢山のsingle packet message が同時に１つのＩＭＰに送られる場合に発生するロックアップである。前述のように一度に各Ｓ－pipe を通して流れるsingle packet message の数は４個までと制限されているが，同一目的地に対して集中するため不規則な順序で到着するメッセージのシークェンシングのためのバッファの不足によりロックアップが生ずる。

このロックアップを避けるため，single packet message は目的地ＩＭＰに於て次のような処理方法がとられる。

(1) シークェンシング上HOST にたゞちに出力できるメッセージは，必ず受け取られHOSTに出力される。

(2) シークェンシング待ちに入るメッセージは，20バッファ以上の余裕がない場合は棄却される。

(3) バッファ事情が好転した場合，バッファに余裕がないために棄却されていたメッセージのうち，HOSTに直ちに送出できるものに対しては発信地ＩＭＰに再伝送が要求される。この要求順序は対象とする全パイプに対してround robbin 方式で均等にサービスされる。

(4) 発信地ＩＭＰはすべてのsingle packet message のコピーを保管しており，目的地ＩＭＰからの要求に対し直ちに再伝送をおこなう。

d．HOST／IMP ロックアップ

このロックアップはHOSTとＩＭＰ間のインタフェースアダプタ（ interface adapter）が半二重であることにより発生するものである。

ＩＭＰ内のバッファが無くなった状態で，若しHOSTからメッセージ伝送の要求が来るとＩＭＰのHOST／ＩＭＰタスク（ task ）はバッファ待ち状態に入る。一方，ＩＭＰ内にHOST向けのメッセージがあれば，それをＨＯＳＴに出力することにより，バッファがあき，HOSTからの要求を受け入れることが出来るようになる。

これは，１つのタスクの中でチャネル利用権とバッファ使用権が競合している状態といえる。

この状態をさけるために，ＨＯＳＴ－IMP protocal はＩＭＰ側に優先権を与えている。即ち，ＩＭＰとＨＯＳＴの両側からの出力がぶつかった場合，ＩＭＰ側の出力が先づ実行され，ＨＯＳＴ側の出力は，ＩＭＰ内のバッファ数が調べられ，ＨＯＳＴからのメッセージが受け入れ可能の場合にのみおこなわれる。

3.2 ソフトウエア

3.2.1 概　要

ＩＭＰの機能をはたすプログラムをSubnet Control Program （ＳＣＰ）と呼ぶことにする。ＳＣＰは割り込み禁止モードで動作するモニター（monitor），割り込み可能モードで動作するＣＰＵディスパッチャー（dispatcher），およびそれぞれのタスク（task）より構成される。これらの関係は（図３－７），（図３－８）に示すとおりである。

A．コントロール・プログラム

コントロール・プログラムはモニターおよびＣＰＵディスパッチャーに対する総称である。

モニターは割り込み分析，各種の割り込みに対するそれぞれの処理，バッファ管理，Ｉ／Ｏサービス，タスク・ターミネイタより構成され，割り込みによる各タスク間の連絡およびサービスを主なる機能としている。

ＣＰＵディスパッチャーは，ＣＰＵがあいた時点で，各タスクを優先順序にしたがって調べてゆき，動作可能なものにＣＰＵを渡す機能をはたす。

B．タスク

以下に各タスクの機能を優先順序にしたがって簡単に述べる。くわしくは，各タスクの説明の項を参照されたい。

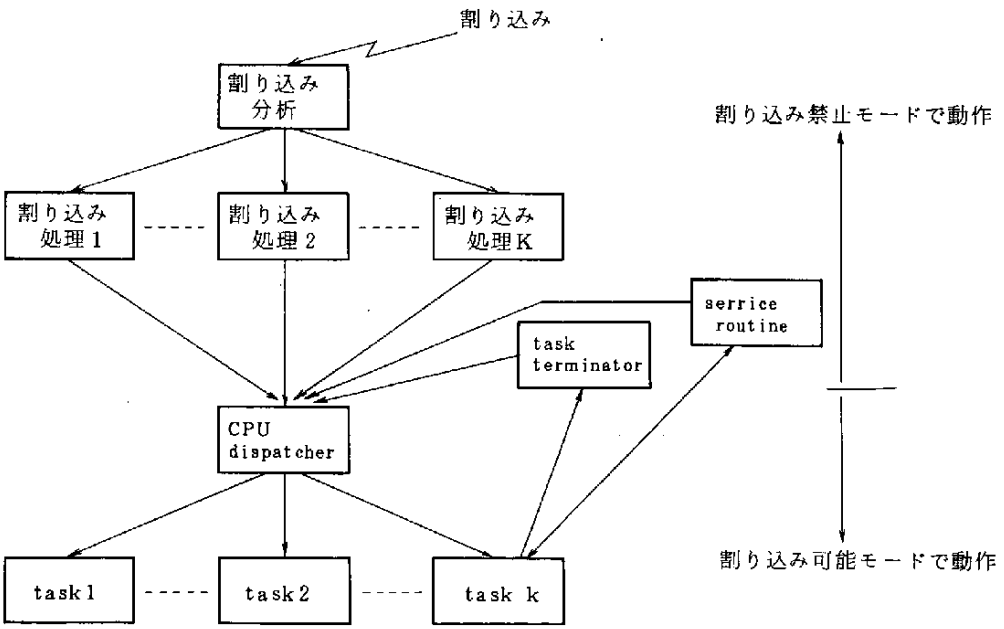


図３－７

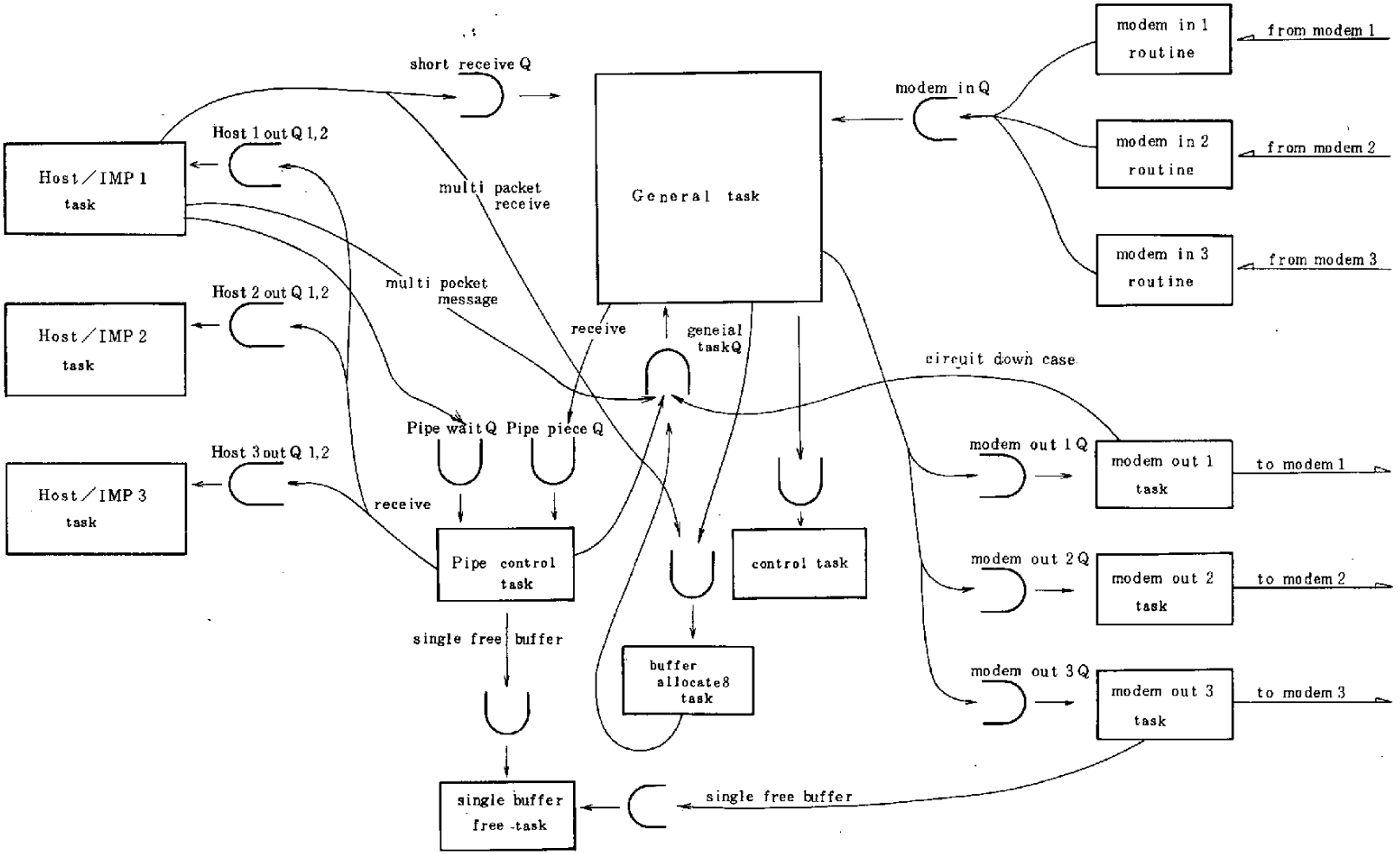

short receive Q
Host 1 out Q 1,2
Host／IMP 1 task
multi packet receive
General task
modem in Q
modem in 1 routine
from modem 1
modem in 2 routine
from modem 2
modem in 3 routine
from modem 3
multi pocket message
Host 2 out Q 1,2
Host／IMP 2 task
receive
geneial taskQ
circuit down case
Pipe wait Q
Pipe piece Q
Host 3 out Q 1,2
Host／IMP 3 task
receive
modem out 1 Q
modem out 1 task
to modem 1
modem out 2 Q
modem out 2 task
to modem 2
Pipe control task
control task
single free buffer
buffer allocate8 task
modem out 3 Q
modem out 3 task
to modem 3
single buffer free task
single free buffer

図3－8

○ Time out タスク

ハードウエアタイマーにより，200 m sec ごとに起動され，ハードウエアの障害監視，紛失メッセージの監視などをおこなう。

○ Pipe管理タスク

フローコントロールを主たる役割とするが，目的地ＩＭＰがディスコネクト状態になったとき，発信地ＨＯＳＴに対してディスコネクト状態になったというreceiveの作成，紛失メッセージの再送をおこなう。

○ Controlタスク

回線のup／down にともないコントロールパケットを発生するとともにupdate Vectorにより Bias table の更新をおこなう。

○ Generalタスク

パケットのルーティング，他ＩＭＰあて，パケットの蓄積交換，メッセージのシークェンシング，リアセンブルをおこなう。

○ Retransmitタスク

目的地ＩＭＰのバッファに余裕がないために捨てられたsingle packet massage に対する再送要求パケットを作り出す。

○ Modem outタスク１～3

回線に出力する必要のあるパケットを送り出す。回線が故障したときには，出力すべきパケットを generalタスクに返す。

○ＨＯＳＴ／ＩＭＰタスク１～3

ＨＯＳＴに対するメッセージの入出力，ＨＯＳＴが障害のときのreceive の作成をおこなう。

○ＨＯＳＴ check タスク

ＨＯＳＴの初期設定，終了会話の処理の指令などをおこなう。

○ Buffer allocate 8タスク

multi packets message の受信のためのバッファ割当てをする。

○ＴＴＹ outタスク

ＡＳＲに対するメッセージ出力，ＩＭＰのコンソールタイプライター間の通信をおこなう。

○ Single buffer freeタスク

Single packet message に利用されたバッファのフリーを管理する。

○ Back groundタスク

ＡＳＲからのメッセージ入力などをあつかう，このタスクは待ち状態にならない。

3.2.2 コントロール・プログラム

コントロール・プログラムは，割り込み処理ルーチン，ＣＰＵディスパッチャ，バッファ管理，Ｉ／Ｏサービス，タスクターミネイタの５個の管理機構から成り立っており，ＣＰＵディスパッチャ以外はすべて割込禁止モードで動作する。

割り込み処理ルーチンは，標準割込を扱い，タスクの状態推移を制御する。タスクの状態は，各タスクに対応するＴＣＢに記述される。ディスパッチャは，優先順位に配列されたＴＣＢをサーチし，READY 状態にあるタスクを見つけ出し，実際にそのタスクを起動する。バッファ管理は，バッファの割付け，解放，予約を監視する。又，各種キューの管理も行なう。バッファ管理は通常のタスクからサブルーチンコールの形式で呼び出され，サービスするが，サービス終了後ディスパッチャに制御を渡すものと，直接呼び出しプログラムに復帰するものがある。

Ｉ／Ｏサービスは，入出力に関係したタスクからサブルーチンコールの形式で呼び出され，実際のＩ／Ｏインストラクションの実行及び入出力を要求したタスクの状態管理（Ｉ／Ｏイベント待ち状態に置く等）を行なう。

タスクターミネイタは，タスクが終了した時，ジャンプの形式で制御を渡され，タスクの後処理を行ない，ディスパッチャに制御を渡す。

A．タスク制御に関する基本的な事項

(1) タスク状態

ＳＣＰ内タスクは，必ず次のいずれかの状態に属する。

ＤＯＲＭ：起動要因が発生する前のアイドル状態

ＲＥＡＤＹ：ＣＰＵ以外のリソースはすべて確保済みの実行待ち状態

ＲＵＮ　：ＣＰＵ及び他の資材を確保し，現在，実行状態

ＩＮＴ　：実行中に割込みが発生し，処理を中断された状態

ＷＡＩＴ：資材の要求待ち又は，事象の完了待ちの状態

（以上の分類は，概念上の分類で，実際の処理上は，

ＤＯＲＭ，ＲＥＡＤＹ，ＷＡＩＴ Ｉ／Ｏ，ＷＡＩＴコアーの４状態を設定する）

(2) タスクの記述

ディスパッチャは，タスク優先順に配列されたＴＣＢ群を調べて優先順位のいちばん高いＲＥＡＤＹタスクを起動する。

ＴＣＢは，そのタスクに関する必要情報を記述したもので，次の情報から構成される。

タスク名称　　：ＴＣＢの位置（アドレス）そのもの

タスク状態　　：ＤＯＲＭ，ＲＥＡＤＹ等を表わす

実行アドレス　：ディスパッチャが制御を渡すアドレスＤＯＲＭの時及びターミネイトし

た時は，次の開始アドレスが，割込まれた時は，中断アドレスがはいる

開始アドレス　：タスクの起動アドレス

Ｘレジスター　：割込まれた時のインデックスレジスターの退避エリア

Ａレジスター　：　　〃　　Ａ　　〃

タスク起動要因：タスク起動の要因がセットされる

システムリザーブ：コントロールプログラムが使用するエリア
バッファの予約に使用される

ＴＣＢの構造

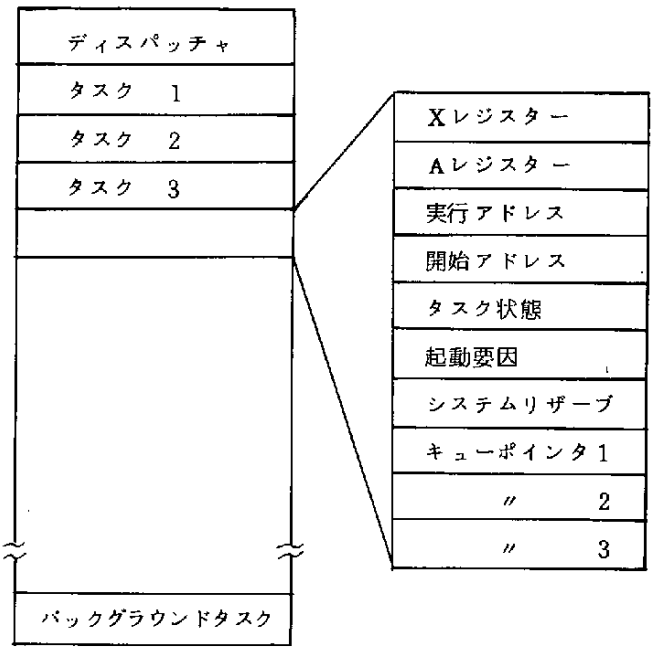


図３－９

(3)　タスク表示

ディスパッチャは，タスクを起動する時に，そのタスク名称（ＴＣＢのアドレス）を走行タスク（Current task）表示エリアにセットする。

タスクコントローラ，バッファ管理，Ｉ／Ｏサービス，タスクターミネイタは，これによって，自分がどのタスクから制御を渡されたかを知る事ができる。

(4)　割込要因

ＳＣＰに関する割込要因として以下のものがある

Ｈ－ＳＬＣ

ＭＬＣ

ホストインターフェイスアダプタ

タイマー

ＰＴＲ

ＴＴＹ

＊メモリパリティエラー

＊電源異常（標準割込みではない）

＊印のついたものは，特に処理を行なわずＣＰＵはストップする

B．割り込み処理ルーチン，ＣＰＵディスパッチャ

標準割込みが発生すると，割込み分析ルーチンに制御が渡される。割込み分析ルーチンは，走行タスクを知り，そのＴＣＢにレジスター，中断アドレスを退避する。次に割込み要因を調べ，直接処理を行なうものに関しては対応するルーチンに直接制御を渡し，それ以外の割込みに対しては，対応するタスクのＴＣＢを適当にセットしディスパッチャに制御を渡す。

ディスパッチャは，走行タスク表示エリアに自分のタスク名称（ディスパッチャＴＣＢアドレス）をセットして割込可とし，ＴＣＢ群を２番目から順にサーチし，最初に発見したＲＥＡＤＹ状態タスクに制御を渡す。この時，割込不可として，走行タスク表示エリアにそのタスク名称をセットし，インデックスイジスタ，Ａレジスターをロードしたのち，割込可にして実行アドレスにジャンプする。ＴＣＢ群の先頭は，ディスパッチャＴＣＢであり，ディスパッチャは，処理が割込みで中断されても常に始めから処理を行なうのでディスパッチャＴＣＢのサーチは行なわない。

C．タスクターミネイタ

タスクが終了した時の後処理をする。

走行タスク表示エリアによりターミネイトタスクのＴＣＢを知り，ＴＣＢを初期状態にセットする。この時，起動条件と対応キューテーブルをチェックし，起動要因が無ければＤＯＲＭに，有ればＲＥＡＤＹにセットして，ディスパッチャに制御を渡す。

D．Ｉ／Ｏサービス

入出力インストラクションの実行及びＷＡＩＴ処理を行なう。

デバイススティタスをチェックし，Dead 状態なら，タスクにエラーリターンする。Wake up 状態なら，走行タスク表示エリアにより，コーリングタスクのＴＣＢを知り，タスクをＷＡＩＴ状態にしたのち，Ｉ／Ｏスタートし，ディスパッチャに制御を渡す。

E．バッファ管理

バッファ管理は，バッファの割付け，予約，解放及び各種キューの管理を行なうモジュール群から構成される。

初期状態においては，バッファは，すべてフリーバッファチェインにつながっており，タスクの要求によって割付けられていく。

バッファの循環経路は，次の（図３－10）のようになる。

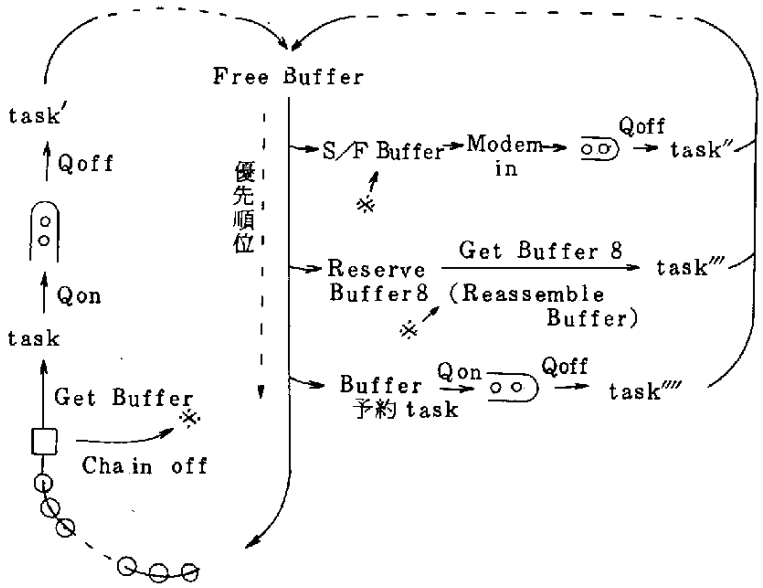


図３－10

F. TOP SEAT CALL

タスクは，一時的に自己の優先順位を最高位にするようリクエストを発することができる。これがTop Seat Call である。

このリクエストは，最高位を要求するSitting down call と，最高位をあけわたして定順位に戻るStanding up call とのペアで使われる。

このリクエストは，他タスクをブロックして，特別な処理を行なうためのものであるから，Top Seat に居る間，タスクがWAITするようなモニターコール（Ｉ／Ｏ，バッファ要求，etc）を行なってはならない。又，Sitting down call を行なったら，必らずターミネイトの前にStanding up call を行なわなければならない。

コーリングシークェンス

```
        ⌇
    INH
    JST*  '131        ;  Sitting down
      ⋮
```

```
  :                          } on the Top Seat
INH
JST*   '132          ;  Standing up
  ≀
```

3.2.3 各タスクの説明

A. Time out タスク

このタスクは，ＩＭＰ内の各種タイマーのタイム・アウト処理をつかさどるものであり下記の機能をはたす。

○ 200ｍsec のタイムアウトが発生した場合

200ｍsec のタイムアウトが発生すると以下の各種タイマーのクロックを１だけ進めタイム・アウトが発生しているものに関しては，タイム・アウト処理をおこなう。

(1) ＴＴＹＴタイマー

入出力タイプライターのタイマー監視をおこないタイム・アウトが発生した場合に制御をＴＴＹタイム・アウトルーチンに渡す。

(2) ＰＴＲＴタイマー

紙テープ入力装置ＰＴＲのタイマー監視をおこないＰＴＲルーチンに制御を渡す。

(3) ＭＬＣＴタイマー

Multi Line Controller(ＭＬＣ)のタイマー監視をおこなう。

(4) ＨＩＡＴタイマー

HOST Interface Adapter（ＨＩＡ）の入出力コマンドのタイマー監視をおこなう。このタイマーは30秒でタイム・アウトとなる。

タイム・アウトが発生するとＨＩＡのステータスを調べDead であり，かつＨＩＡの電源がonであればＨＩＡのプログラム・スウイッチＰＳＷをアンロックする。しかし電源がoff であればＨＩＡのタイム・アウト値に30秒をセットする。

またＨＩＡのステータスがDeadではなく，かつＨＩＡタスクがI/O WAIT であれば当該ＩＭＰをdownとする。

(5) ＳＥＣ１タイマー

このタイマーは，single packet message を送出してから，そのパケットの receive を受けとるまでのタイマー監視をおこなうものであり，約４秒でタイム・アウトを発生する。

このタイマーは図３－11のように current 1 sec timer pointer（１語）と１－sec timer counter（４語）から構成されている。

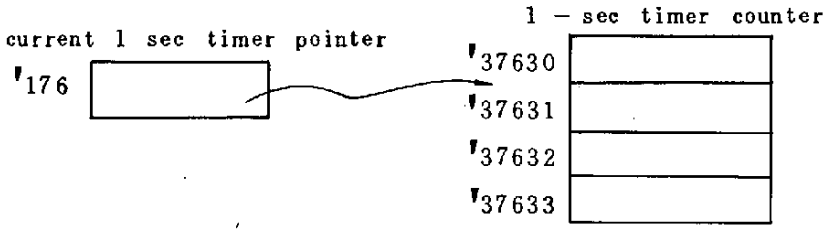


図3－11

1 secのタイム・アウトが生じるごとに，1－sec timer conterのアドレスを示すcurrent 1 sec timer pointer を1だけポインターをアップする。いまポインターが'37632 番地を示しているとき，ポインターを1アップした場合には，ポインターは'37633 番地を示す。

またポインターが '37633 番地を示していて，ポインターを1アップした場合ポインターは，'37630 番地を示す。

すなわち，current 1 sec timer pointer （'176番地）が示す範囲は '37630 ≦ current 1 sec timer pointer ≦ '37633 である。以上の方法により

1－sec timer pointer で示されたアドレスに対してsingle packet message 送出のタイマーはセットされ約4秒でタイム・アウトが発生する。タイム・アウトが発生するとsingle pipe entrance timer のテーブル（ '37660 番地以降）を調べcurrent 1 sec timer pointer が示すアドレスと一致するものがあれば該当するパイプ管理テーブルのsingle entrance につながれている receive 待ちメッセージに関して質問パケットを発信する。

タイマーのセットおよびリセットは，下記の通りである。

○タイマーのセット

```
LDX    Single pipe K entrance timer nのアドレス
LDA    '176
STA    0,1
STA    0
IRS    0,1
```

○タイマーのリセット

```
LDX    Single pipe K entrance timer nのアドレス
CRA
IMA    0,1
STA    0
```

```
LDA     0,1
SUB     '163    /* '163 contains 1 */
STA     0,1
```

37630	1-sec timer counter 0	1-sec timer counter 1	1-sec timer counter 2	1-sec timer counter 3
37634	4-sec 〃	4-sec 〃	4-sec 〃	4-sec 〃
37640	L-pipe 1 entrance timer	L-pipe 2 entrance timer	L-pipe 3 entrance timer	L-pipe 4 entrance timer
37644	〃 5 〃	〃 6 〃	〃 7 〃	〃 8 〃
37650	〃 9 〃	〃 10 〃	〃 11 〃	〃 12 〃
37654	〃 13 〃	〃 14 〃	〃 15 〃	〃 16 〃
37660	S-pipe 1 entrance timer 0	S-pipe 1 entrance timer 1	S-pipe 1 entrance timer 2	S-pipe 1 entrance timer 3
37664	〃 2 〃	〃 2 〃	〃 2 〃	〃 2 〃
37670	〃 3 〃	〃 3 〃	〃 3 〃	〃 3 〃
37674	〃 4 〃	〃 4 〃	〃 4 〃	〃 4 〃
37700	〃 5 〃	〃 5 〃	〃 5 〃	〃 5 〃
37704	〃 6 〃	〃 6 〃	〃 6 〃	〃 6 〃
37710	〃 7 〃	〃 7 〃	〃 7 〃	〃 7 〃
37714	〃 8 〃	〃 8 〃	〃 8 〃	〃 8 〃
37720	〃 9 〃	〃 9 〃	〃 9 〃	〃 9 〃
37724	〃 10 〃	〃 10 〃	〃 10 〃	〃 10 〃
37730	〃 11 〃	〃 11 〃	〃 11 〃	〃 11 〃
37734	〃 12 〃	〃 12 〃	〃 12 〃	〃 12 〃
37740	〃 13 〃	〃 13 〃	〃 13 〃	〃 13 〃
37744	〃 14 〃	〃 14 〃	〃 14 〃	〃 14 〃
37750	〃 15 〃	〃 15 〃	〃 15 〃	〃 15 〃
37754	S-pipe 16 entrance timer 0	S-pipe 16 entrance timer 1	S-pipe 16 entrance timer 2	S-pipe 16 entrance timer 3

図 3 — 12

(6) ＳＥＣ４タイマー

パイプがdownしたときsingle exit のメッセージポインターにつながれているバッファをフリーにするための監視タイマーである。

パイプがdownしたときsingle exit のメッセージポインターにつながっているバッファが，modem out タスクキューに再送パケットとしてキューon されている場合が存在するためパイプ管理タスクでは，single exit につながっているバッファをフリーにすることはできない。

このためパイプ管理タスクは，パイプのdown時にsingle exit につながっているバッファをタイム・アウトタスクでEntry 宣言した領域DFPTIMをExternal 参照してDFPTIM＋16からの16語をパイプk (k＝1，…，16)のチェーンヘッドとし，バッファ＋81をチェーン領域としてつなぎDFPTIMからの16語にタイム・アウトを20秒後に発生するための値－5をセットする。

ＳＥＣ４タイマーは，DFPTIMから16 語を順次調べタイム・アウトになったパイプ関係のバッファをフリーにする。

ＥＮＴで宣言されているDFPTIMは図３－13の形をしている。

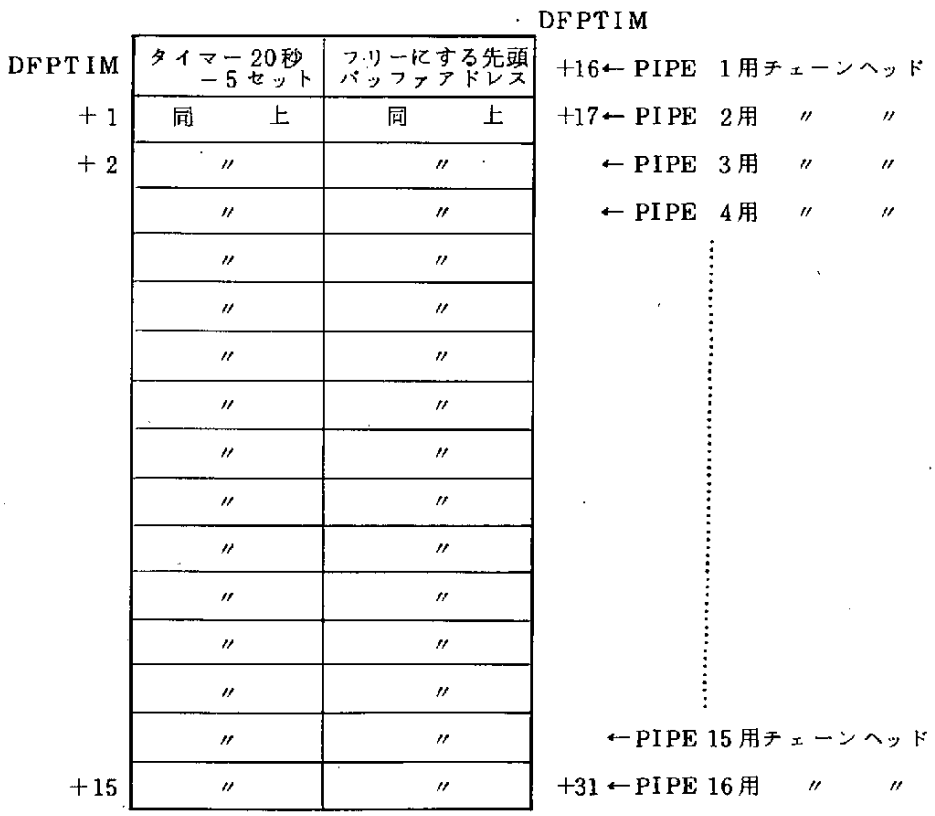


図３－13

図３－14は，DFPTIMのパイプ用領域にsingle exitのフリーにするバッファが３個タイマー値－５がセットされ，バッファの先頭＋81番地でチェーンされていることを示す。

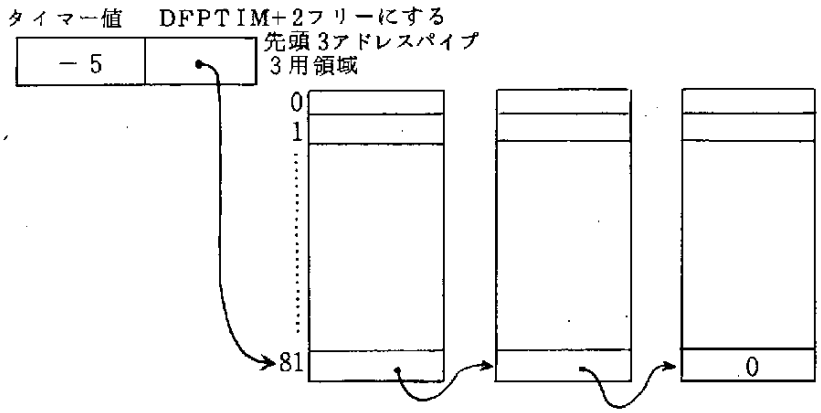


図３－14

(7) ＨＯＵＲタイマー

パイプがdownのときsingle exit のメッセージポインターにつながれたバッファをフリーにするための監視タイマーがセットされていれば，その値を再セットし，されていなければタイマー値０にする。

Ｈ－ＳＬＣタイマー１～３，ＨＩＡタイマー１～３，ＴＴＹタイマー，ＰＴＲタイマー，ＭＬＣタイマー，1SECタイマー，４SECタイマー，１HOURタイマー，Ｈ－ＳＣＬtimingタイマー１～３を順次調べタイマーがセットされていないものに対してタイマー値０をセットする。

(8) ＴＭＧタイマー

48 kbps モデムを制御するHigh Speed Single Line Controller （Ｈ－ＳＬＣ）の回線制御タイマーである。400ｍsec でタイム・アウトを発生させ，下記のステップ状態によって回線の制御を行なう。

○ステップ０　回線初期状態

Ｈ－ＳＬＣを送受信可能の状態にする。ＥＲオン実行する。

next ステップ　１

○ステップ１　Ｈ－ＳＬＣ送受信可能の状態

モデムが送受信可能か確認する。ＤＲオン実行する。

○可能のとき

回線を送信状態にする。ＲＳオン実行する。

next ステップ　２

○不可能のとき

ステップ0でERオン実行の後200msec たってもモデムより送受信可能の信号DRオンが確認できなかった場合である。ERオン実行する。

nextステップ 1

200msecごとのDRオン信号が2回とも確認できなかった場合H－SLCのステータスがDeadであればステップ0から実行する。

ステータスがDeadでなければ，モデム・アウトタスクのコミュニケーションエリアのDead フラッグをオンにし，H－SLCのステータスをDead，Helloカウンター，I heard you カウンターに初期値－80をそれぞれセットしステップ0から実行する。

○ステップ2　モデム送受信可能の状態

受信回線のキャリア検出を行なう。CDオン実行する。

○検出できた場合

H－SLCの割込み情報転送アドレスと，受信バッファアドレスをセットし受信スタートをする。

H－SLCのステータスを調べDeadであれば，モデム・アウトタスクのコミュニケーションエリアのHello フラッグをオンにする。ステータスがDead でなければ，H－SLCのステップを0，modem outタスクコミュニケーションエリアDead フラッグをオン，H－SLCのステータスをDead，Helloカウンター I heard you カウンターに初期値－80をセットする。

○検出できなかった場合

回線を送信状態にする。RSオン実行する。

nextステップ 3

○ステップ3　回線に送信可能の状態

受信回線のキャリア検出を行なう。CDオン実行する。

○キャリアが検出できた場合

ステップ2のキャリア確認後から実行する。

○キャリアが検出できない場合

H－SLCのステータスがDeadであればステップ0から実行する。ステータスがDead でなければモデム・アウトタスクのコミュニケーションエリアDead フラッグをオン，H－SLCのステータスをDead，Helloカウンター，I heard you カウンターに初期値－80をセット

しステップ0から実行する。

○ステップ4　Hello状態

モデム・アウトタスクコミュニケーションエリアHelloフラッグをオンにする。

H－SLCのステータスがDeadでなければ，H－SLCのステップ0，モデム・アウトタスクコミュニケーションエリアDeadフラッグをオン，H－SLCのステータスをDead，Helloカウンター，I heard youカウンター初期値－80セットする。

○ステップ5　回線upの状態

受信回線のキャリア確認を行なう。CDオン実行する。キャリアが検出できない場合その回線を接断しH－SLCのステータスがDeadであればステップ0から実行する。なければモデム・アウトタスクのコミュニケーションエリアDeadフラッグをオン，H－SLCのステータスをDead，Helloカウンター，I heard youカウンターに初期値－80をセットする。

(9) H－SLCタイマー

High Speed Single Line Controller（H－SLC）の回線障害監視タイマーである。

モデム・アウトタスクよりメッセージ送出と同時にタイマー400 m secがセットされる。タイム・アウトが発生すると該当するモデム・アウトタスクを調べI／O待ちであれば，回線障害が発生したものとして回線を接断し当該H－SLCを初期状態としてHelloカウンター，I heard youカウンターに初期値－80をセットする。

また，該当H－SLCの回線状態がDeadのとき，モデム・アウトタスクコミュニケーションエリアのDeadフラッグをオンにする。

○ HOST Interface Adapter（HIA）のHalt I／O処理

HIAから下記の割込みが発せられたとき，HOSTのI／O障害としてtime outタスクのコミュニケーションエリアに情報がセットされHOSTの障害処理が行なわれる。

(1) HOST側からHaltコマンドが発生し当該IMPのHIAステータスがアンロックのとき。

(2) インコレクトレングスの割込みが発生したとき。

(3) 当該IMPのHIAステータスがDeadまたは，Halt I／Oのとき。

上記（1, 2, 3）の要因により time out タスクが起動されると当該HOSTのステータスを調べdown状態であればHIAのコマンド監視タイマー30秒をセットする。

ステータスがdownでなければ当該HOSTのHIAステータスをDead状態，プログラム・スウィッチPSWをロック状態とし，当該HOSTを切りはなしHalt I／Oを発信する。

このタスクは，200 m secハードウエアタイマーまたは，ＨＯＳＴ interface adapter 接断の割込みが発生し，time outタスクのＴＣＢコミュニケーションエリアに情報がセットされることによりディスパッチャによって起動される。200 m secハードウエアタイマーによる起動であれば，各タイマークロックを進め計数のカウントがタイム・アウト値にたっしたものに対してタイム・アウト処理を行なう。

B．Pipe管理タスク

Pipe 管理タスクは他ＩＭＰに向けてでていくメッセージフローをコントロールする役割をはたす。

このタスクはパイプ待ちキュー，パイプ pieceキューに対するキュー on およびコミュニケーションエリアへの起動条件セット（コントロールタスクによってセットされる）により起動される。pieceとは4語より構成される情報でありモニタが管理している。

パイプ待ちキューには発信ＲＦＡ pieceおよびsingleパケットがキュー on される。パイプ piece キューにはallocate 8 piece，inquiry piece，retransmit piece，give back piece，receive piece および free pieceがキューonされる。

以下に上記piece，パケットがキュー on されて起動された場合，およびコミュニケーションエリア起動の場合のpipe 管理タスクの動作概要を述べる。

○　コミュニケーションエリア起動

Connection matrixを参照し，新たにデイスコネクトなIMPを見つけた場合，パイプ down処理を行なう，またコネクトなＩＭＰを見つけた場合にはパイプup処理を行なう。

○　発信ＲＦＡ piece キュー on

目的地ＩＭＰがディスコネクトならば目的地down piece に作り変え，発信Ｈ－Ｉタスクの HOST outキュー2へキュー onする。

目的地ＩＭＰがコネクトならば発信ＲＦＡ pieceを目的ＩＭＰへのパイプ待ちチェーンへつなぐ。そのときパイプが休止状態ならばＲＦＡパケットを作成しgeneral タスクキューにキュー onする。

○　Single Packetキュー on

目的地ＩＭＰがディスコネクトならばパケットをフリーにし，発信Ｈ－Ｉタスクの HOST outキュー2へ目的地down piece をキュー onする。

目的地ＩＭＰがコネクトならば目的地ＩＭＰへのパイプに入っているメッセージ数を調べる。メッセージ数が4ならば該当パケットをパイプ待ちチェーンへつなぐ。メッセージ数が4以下ならばメッセージ番号を付け加えgeneralタスクキューへキュー onするとともにパケットのチェーン領域を利用してreceive待ちメッセージ・ポインターへつなぐ。

○ Allocate 8 pieceキュー on

受け取ったallocate 8 pieceのメッセージ番号を調べ有効か否かのチェックを行なう。有効でなければpieceをフリーし処理を終る。有効ならばpipe entranceステータスが割当て待ちか否かを調べる。割当て待ちでなければpieceをフリーし処理を終る。割当て待ちならばパイプ待ちチェーンの先頭につながっているＲＦＡ pieceの発信ＨＯＳＴが生きているか否かのチェックを行なう。死んでいるならばallocate 8 pieceをフリーし，pipe entrance ステータスをHOST down とし処理を終る。生きているならば，pipe entranceステータスを割当て待ちとしパイプ待ちチェーンの先頭につながっているＲＦＡpieceを発信したＨ－ＩタスクのHost outキュー2へＯＫpieceをキューonする。

○ Retransmit pieceキュー on

受け取ったretransmit pieceがＬ－pipeに対するものかＳ－pipeに対するものかを調べる。

Ｌ－pipeに対するものならばメッセージ番号が有効でかつpipe entranceステータスが送信中ならばretransmit pieceを目的とするＨ－ＩタスクのHost outキュー2へ キューonする。ステータスがそれ以外ならばpieceをフリーし処理を終了する。

Ｓ－pipeに対するものならばメッセージ番号が有効か否かのチェックを行ない有効ならばreceive 待ちパケットを再度HOSTタスクキューへキューonする。有効でなければpieceをフリーし処理を終了する。

○ Give back pieceキュー on

受け取ったgive back pieceのメッセージ番号をチェックする。有効でなければpieceをフリーし処理を終了する。有効な場合，(1) long exitステータスがgive back 待ちあるいはreceive 発信済み，(2)ステータスは休止かのチェックを行なう。(1)の場合，long exit ステータスを休止とした後フリーパケットを作成しgeneral タスクキューにキューonする。(2)の場合，フリーパケットを作成しgeneralタスクキューにキューonする。(1)，(2)以外のとき，pieceをフリーし処理を終了する。

○ Free pieceキュー on

受け取ったフリーpieceのメッセージ番号が有効かつlong entranceステータスがフリー待ちならばlong entranceを休止にする。上記以外ならばpieceをフリーし処理を終了する。

○ Inquiry pieceキュー on

受け取ったinquiry pieceがＬ－pipeに対するものかＳ－pipeに対するものかを調べる。

Ｌ－pipeに対するものだった場合，pieceのメッセージ番号が(1) exit pipeのメッセージ番号と同じか，(2) 1だけ少ないかのチェックを行なう。(1)(2)どちらでもなかった場合，

pieceをフリーし処理を終る。(1)の場合，exit pipe ステータスがallocate 8発信済みかreceiveを発信した状態ならば再送パケットを作成しgeneral タスクキューへキューonする。それ以外のステータスならばpieceをフリーし処理を終る。(2)の場合，exit pipe ステータスがreceiveを発信した状態ならばreceiveコピーからreceive パケットを作成しgeneral タスクキューにキューonする。

S－pipe の場合，メッセージ番号が有効か否かのチェックを行なう。無効な場合pieceをフリーし処理を終る。メッセージ番号が有効な場合でsingle exit のメッセージ番号より若い番号ならばパイプテーブル上にreceiveコピーがあるのだからreceive パケットを作成しgeneralタスクキューにキューonする。さもないときは再送パケットを作成しgeneral タスクキューへキューonする。

○ Receive pieceキューon

受け取ったreceive pieceがL－pipe に対するものかS－pipe に対するものかを調べる。

L－pipeに対するものだった場合，メッセージ番号のチェックを行ない，有効でないならばpieceをフリーし処理を終了する。有効ならばlong entrance ステータスが(1)送信中，(2)HOST downかどうかを調べる。(1)(2)以外ならばpieceをフリーし処理を終了する。(1)ならばlong パイプ待ちチェーンの先頭につながっているＲＦＡpieceを発信したＨ－Ｉタスクの HOST outキュー2へreceive pieceをキューon すると同時に上記ＲＦＡpiece をチェーンからはずしフリーする。そのときlong パイプ待ちチェーンにまだpieceがあるならばそれの発信HOSTが生きていることを確認の上，Ｈ－ＩタスクのHOST outキュー2へＯＫpieceを作成してキューon し処理を終了する。ＯＫpieceをキューonするＨ－Ｉタスクがないならば(2)の場合と同様の処理を行なう。

long entrance ステータスが(2)のときステータスをフリー待ちとしgive back パケットを作成しgeneralタスクキューにキューonする。

S－pipe に対するものだった場合，メッセージ番号のチェックを行ない，有効でないならばpieceをフリーし処理を終了する。有効ならばreceiveをパイプテーブルのsingle entranceへコピーし，対応するメッセージバッファをＦＰＱへキューonする。受け取ったreceive pieceがすぐにHOSTに渡せるものならば該当Ｈ－ＩタスクのHOST out キュー2へキューonする。receive piece を受けとることによりパイプ内のメッセージ数が4以下になるので，パイプ待ちメッセージがあれば，それをgeneral タスクキューにキューonする。

(注) 有効なメッセージ番号

メッセージ番号はL－pipe，S－pipe およびpipe 番号ごとに与えられるものであり，

通常のメッセージはこの番号により他のメッセージと区別される。

コントロールパケットは通常のメッセージに付随して発生するものと考えられる。コントロールパケットにおいて有効なメッセージ番号は次のものである。

○ RFA

RFA送信側

L－pipeを通じて新たに送り出したいメッセージのメッセージ番号。

RFA受信側

L－pipeを通じて新たに受け取ってよいメッセージのメッセージ番号。

○ Allocate 8

RFAの送信側と受信側を逆にしたもの。

○ Inquiry

INQ送信側

パイプに入っているがreceiveを受け取っていないメッセージのメッセージ番号。

INQ受信側

パイプに入っている可能性のあるメッセージのメッセージ番号。

○ Retransmit

retransmit送信側

パイプに入っている可能性があるメッセージのメッセージ番号。

retransmit受信側

パイプに入っているがreceiveを受け取っていないメッセージのメッセージ番号。

○ Receive

receive送信側

受け取ったメッセージのメッセージ番号。

receive受信側

パイプに入っているメッセージのメッセージ番号。

○ Give back

give back送信側

次に送るべきメッセージのメッセージ番号。

give back受信側

次に受けとるメッセージのメッセージ番号。

○ Free

give backの送信側と受信側を反対にしたもの。

a．Communication area 起動

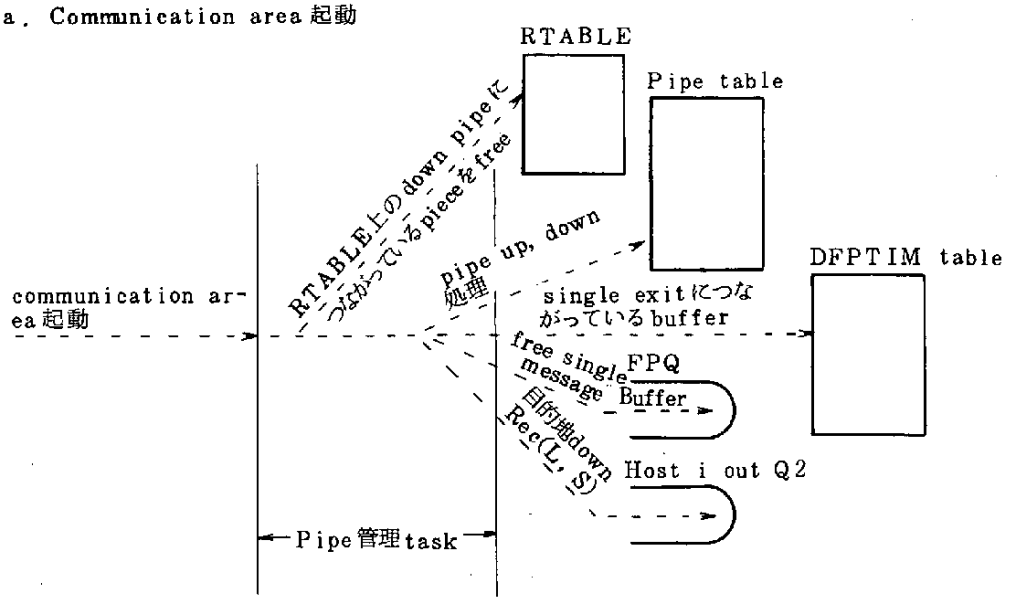


図３－15

b．Pipe 待ちＱ起動

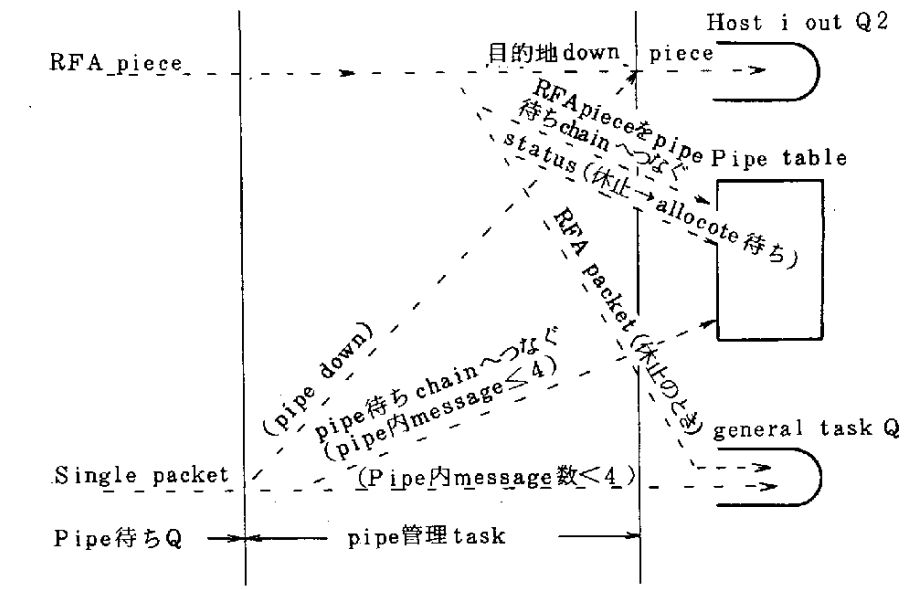


図３－16

c. Pipe piece Q起動（receive piece以外）

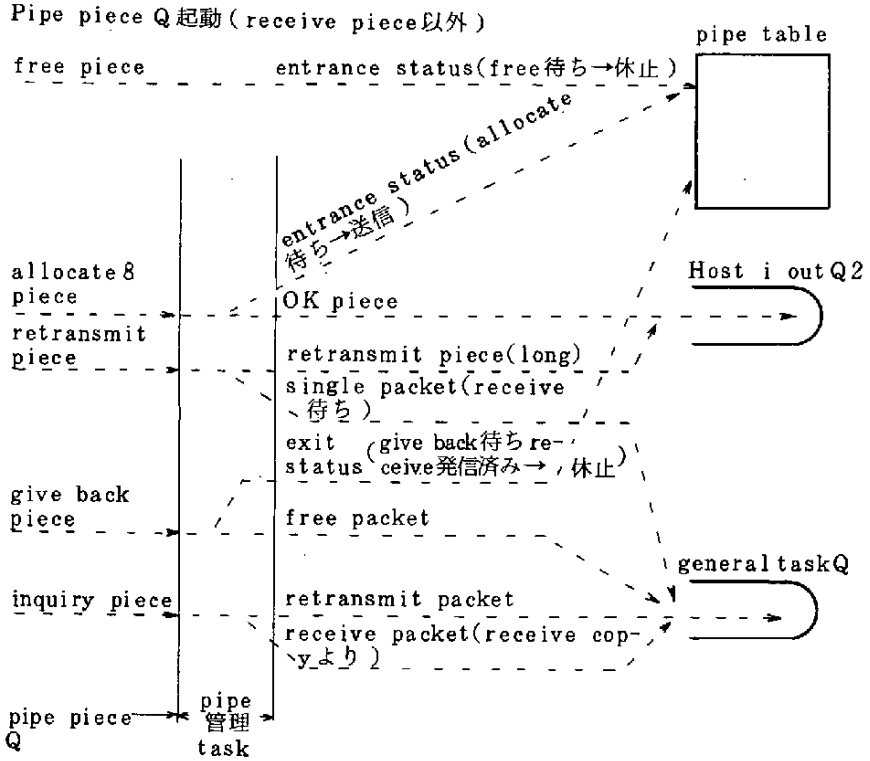


図3－17

d. Pipe piece Q起動（receive piece）

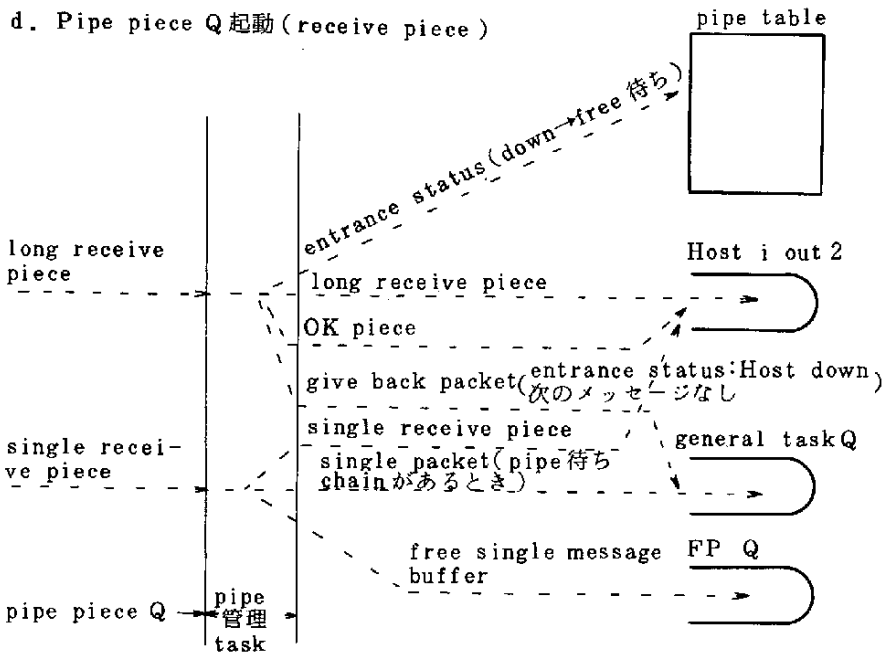


図3－18

C. Control タスク

Controlタスクは Bias table, Connection matrix の更新をその役割りとしている。

このタスクはルーティング information キューに回線障害情報パケット，回線復帰情報パケット，Update Vector パケット，my line status change piece がキュー on されることにより起動される。

起動されると Bias table, Connection matrix の更新を行ない，隣接IMPに前記パケットの形で更新情報を送る。

また更新の結果，新らたにＩＭＰディスコネクト，コネクトのいずれかの状態が発生した場合には pipe 管理タスクを起動させ，パイプテーブルの更新を行なわせる。

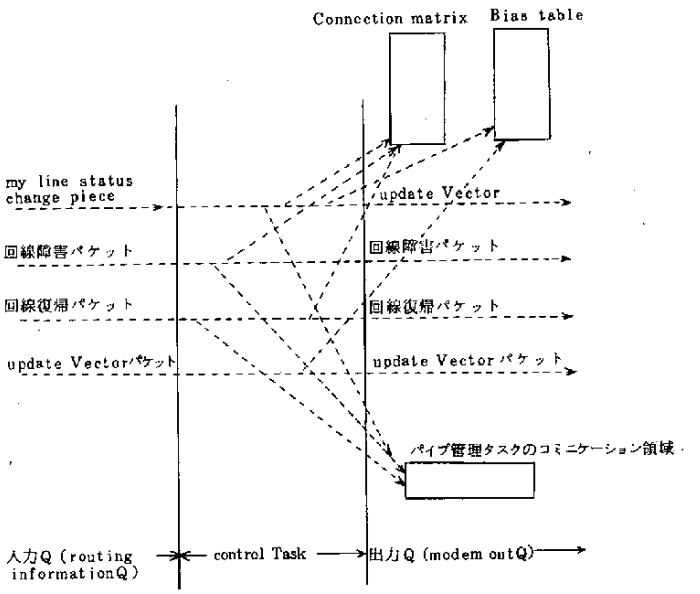


図３－19

D. General タスク

Generalタスクはパケットの蓄積交換，メッセージのシークェンシング，リアセンブルをおこなう。このタスクは，modem in キュー，short rec キュー，general タスクキューに対するキュー on により起動される。

(1) Modem in キュー

a. Line up, Line down, Update Vector パケットについては，フリーバッファに余裕があれば受け入れ，ルーティング Information キューにキュー on する。

b. 目的地が他ＩＭＰのパケットは，受け入れ後ルーティング処理により最適ルートを見つけだし蓄積交換をする。

c. 自己ＩＭＰ宛コントロールパケットは，Host down 以外は pieceにコピーして対応する

キューにキューon（Alloc－8キュー，Pipe Pieceキュー）

d．HOST downパケットはパイプステータスに応じてパイプテーブルを変更する。

e．自己IMP宛Singleパケットは，パイプテーブル参照によりシークェンシングを行ない，対応するHOST outキューにキューon。この時，シークェンシング待ちパケットは，フリーバッファに余裕があれば，パイプテーブルにつなぎ，なければ，retransmit request Pieceを作って再送テーブル（restransmit table）につなぎ，バッファはフリーにする。

f．自己IMP宛longパケットは，パイプテーブル参照によりリアセンブルを行ない，リアセンブルが終了したら，対応するHOST outキューにキューon。

(2) Short Recキュー

Recieve パケットをパイプテーブルの対応するエリアにコピー後ルーティング処理

(3) Generalタスクキュー

パケットのルーティングを行なう。

目的IMP番号により，Bias table を参照し，到着時間が最短のモデム番号を決定し，該当modem outキューにキューon。

最短時間が∞の場合は，

自己発信single パケットはFMGキューにキューon

自己発信single 再送パケットはそのまま何もしない

その他のパケットは，バッファフリーを行なう。

(4) パケットの受け入れ

パケットヘッダーにセットされているACK語のTransビット，チャネル番号により，自己IMPの対応RECビットをとりだし，Transビット＝RECビットなら重複パケット（Duplicate Packet）として，フリー。Transビット≒RECビットなら，RECビットを反転し，パケットを受け入れる。この処理はmodem inキューにキューonされたバッファに対しておこなう。

E．Retransmitタスク

目的地においてバッファの余裕がないために捨てられたsingle packet message に対する再送パケットを発信する。このタスクはバッファに余裕ができると起動される。

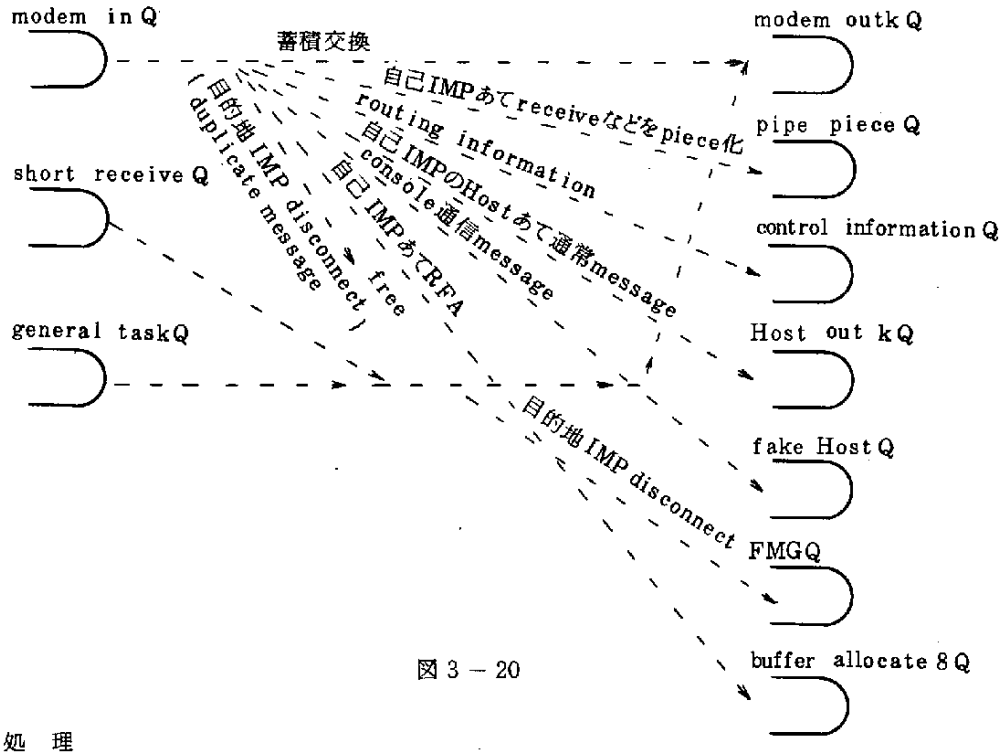


図3－20

○処　理

再送テーブルをサーチし，retransmit request pieceがあったら，プットoffし，再送パケットを作り，パイプテーブルの対応するメッセージポインターに割当て済みとしてつなぎ，HOSTタスクキューにキューonする。

再送テーブル（49W）

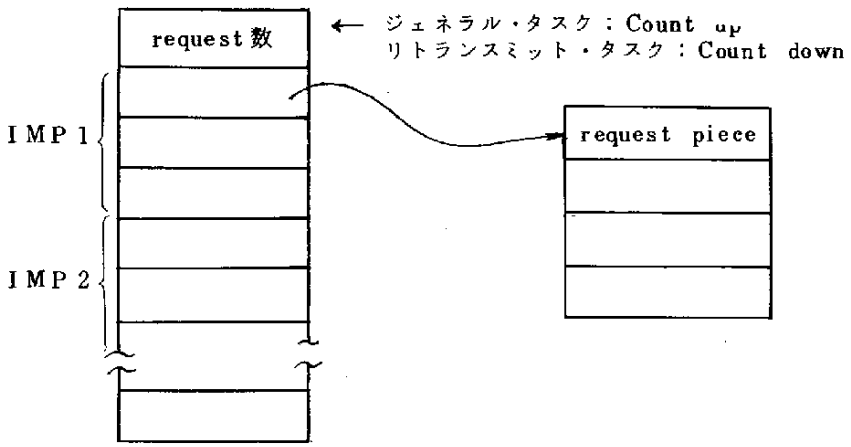


図3－21

F. Modem out タスク

Modem out タスクは高速通信回線の1回線につき1つ対応するタスクであり，1～3までの3個ある。

このタスクは，コミュニケーションエリアのDeadフラッグ，Helloフラッグ，null フラッグ，channelフラッグのセットおよびmodem outキューに対するキューon により起動され，通信回線にパケットを出力する役割をはたす。

(1) Deadフラッグ

○チャネル及びmodem out キューにつながっているバッファを調べ，Line up パケット，Line downパケット，Update Vectorパケットについては，バッファフリー，その他のパケットについては，generalタスクキューにキューonする。

○ルーティングInformationキューにdown pieceをキューonする。

(2) Helloフラッグ，IHYフラッグ

Helloパケット及びI Heard You パケットをモデムに送出する。

(3) Nullフラッグ

Modem outキューの有無を調べ，有ればキューの処理へ，無ければ，Null パケットをモデムに送出する。

(4) Channelフラッグ

このフラッグは，time out タスクが200ms毎にセットするフラッグで処理は次のキュー処理と同じ。

(5) Modem outキューにキューon

チャネルのuseビットを調べ，使用中であれば，つながっているバッファを再送出，使用中でなければ，つながっているバッファのプットoff操作を行ない，modem outキューをキューoff，キューがあったらチャネルを使用中にし，バッファをつないで，モデムに送出する。

(6) プットoff操作

使用中でないチャネルにバッファがつながっていればそれを調べ，自己発信singleパケットは，FMGキューにキューon。自己発信single retransmit packet は，チャネルからはずすだけ，その他のパケットは，バッファフリーを行なう。

(7) ACK語のセット

パケットヘッダーの第2語目上位8ビットに図3－22のようにセットする。

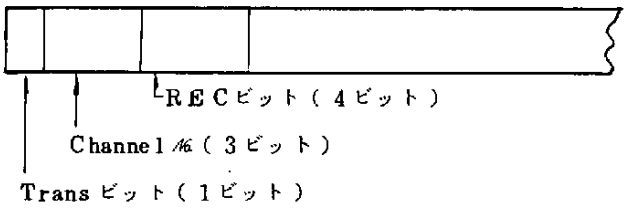


図3－22

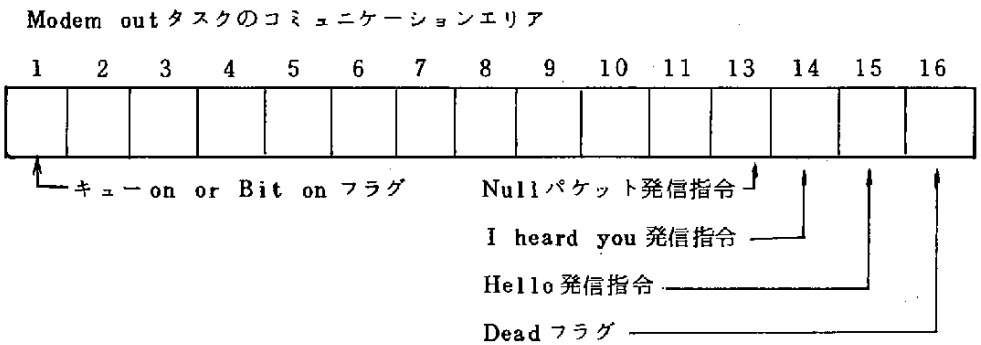


図3－23

G. HOST－IMPタスク

HOST－IMPタスク（H－Iタスク）はチャネル経由でHOSTとメッセージの送受を行なうタスクである。このタスクはIMP内に3個あり各1つのタスクが1つのHOSTにサービスする。タスクの起動はHOSTよりのメッセージ送信要求，HOST outキュー1およびHOST outキュー2へのキューonによりなされる。

HOST outキュー1はHOSTへ出力すべき通常メッセージ（バッファから構成）がキューイングされており，その型式は図3－24のとおりである。

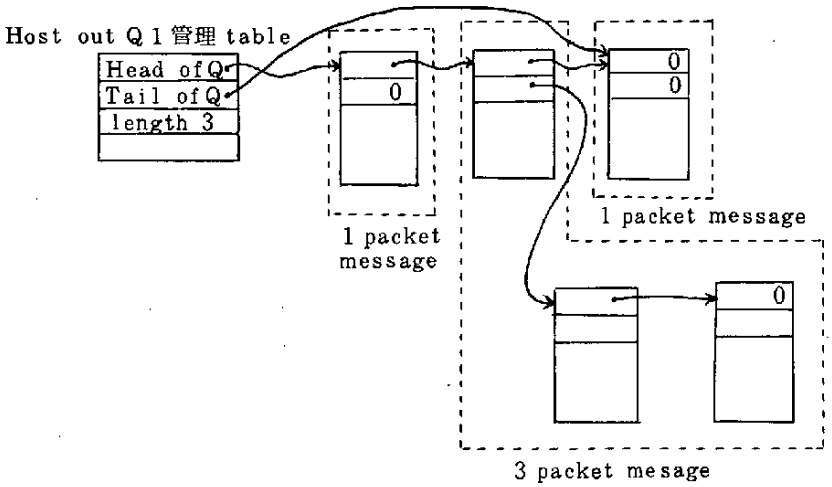


図３－24

HOST outキューは receive 等の通常メッセージ以外の制御情報がpieceの形でキューイングされている。

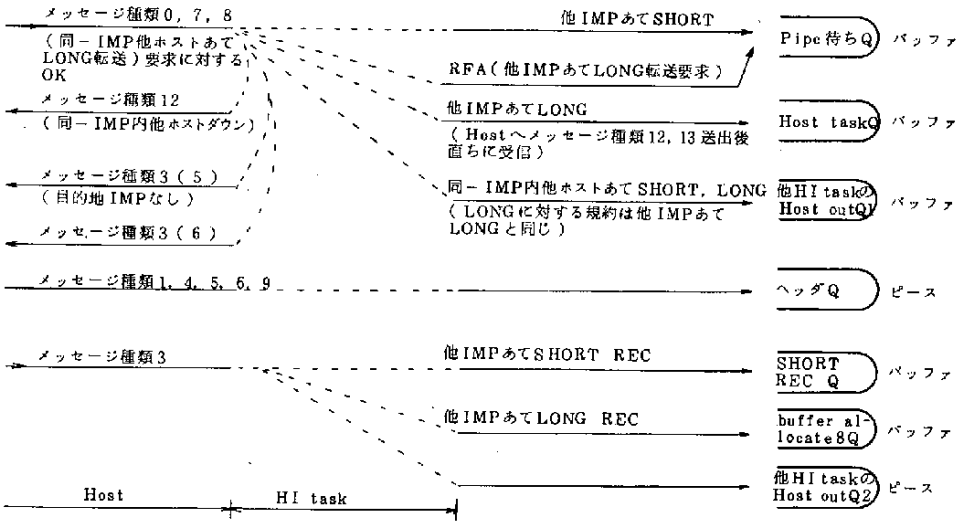


図３－25

注)

同一ＩＭＰ内他HOST downあるいは目的地ＩＭＰなしでＲＥＣをホストへかえすとき，Shortメッセージに関してはメッセージ本体を受け取ってからそれを捨て，Long メッセージに関してはメッセージ本体を受け取る必要はない。

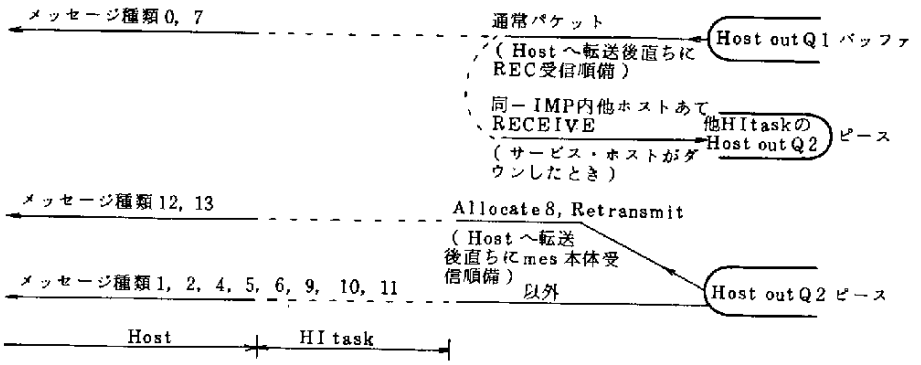


図 3 － 26

注） HOST outキュー2を構成するpieceは 2，3語目にセグメントリーダーが入っているのでそのままHOSTへ転送する。ただしセグメントリーダーの11ビット目（送信情報の有無）をセットする必要がある。

メッセージ種類 12，13に対応するメッセージ本体パケットには，ＡＬ８，ＲＥＣピースに付記されているメッセージ番号をつける。（通常パケット）

サービス・HOST が down した場合，HOST outキュー１，HOST outキュー２の内容を全て捨てる，ただし同一ＩＭＰ内他 HOSTよりの通常メッセージに関してはHOSTdownのＲＥＣを返す。

H．HOST checkタスク

HOST interface adapterから原因不明の割り込みが発生して，当該HOSTが down のとき，当該HOSTを初期設定中としてセグメント・サイズが標準であれば初期設定指令を，標準でなければセグメント長変更指令をそれぞれ当該 HOSTに発信するタスクである。またHOST→ＩＭＰ制御情報のうち下記のメッセージに対応した処理も行なう。

- ○初期設定完了
- ○接続中止要求
- ○切断完了
- ○稼動応答

このタスクは，HOST interface adapterから原因不明の割り込みが発生し，タスクのコミュニケーションエリアのビットがオンにセットされた場合，およびヘッダーキューに初期設定完了pieceがキューon された場合に起動される。

○　コミュニケーションエリア起動

当該 HOSTの状態を調べ down であれば，当該 HOSTを初期設定中としてセグメント・サイズが標準であれば初期設定指令を，標準でなければ，セグメント長変更指令を当該 HOSTout

キュー2にキューonする。

○ 初期設定完了pieceキューon

当該HOSTの状態を調べ，当該HOSTが初期設定中であればup状態とする。

○ 接続中止要求pieceキューon

当該HOSTの状態を調べ，up状態であれば，当該HOSTをdownとし，切断完了を当該HOST outキュー2にキューonする。

○ 切断完了pieceキューon

当該HOSTの状態を調べ，当該HOSTが切断中のとき，HOSTをdown状態とする。また，当該ＩＭＰのすべてのHOSTがdown状態であればタスク終了処理を行なう。

○ 稼動応答pieceキューon

キューoffしたpieceをフリーにするだけである。

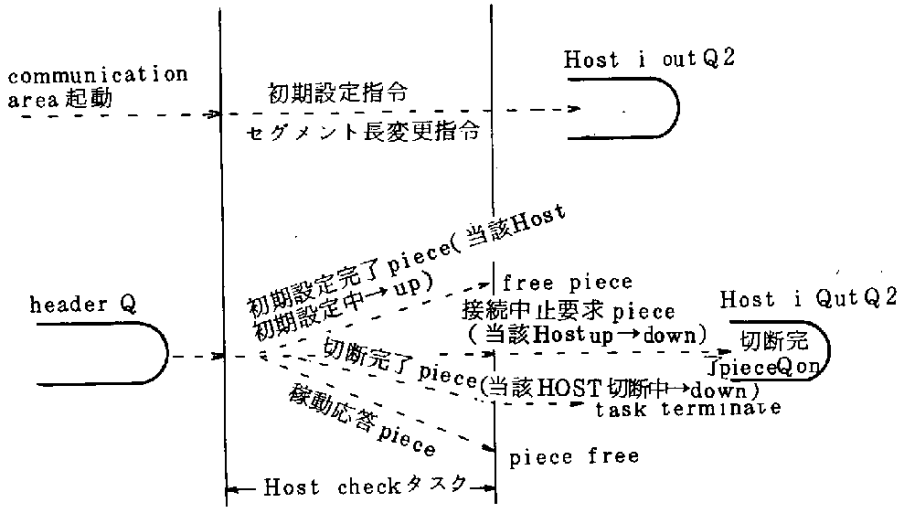


図３－27　HOST checkタスクの起動

HOST checkタスクのコミュニケーションエリアを図３－28に示す。

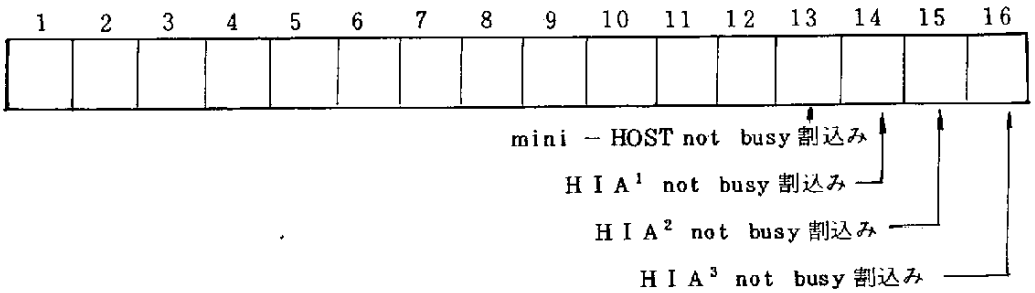


図３－28

I. Buffer allocate 8 タスク

Buffer allocate 8 タスクは longメッセージ用バッファをアロケートするタスクである。

このタスクは buffer allocate 8 キューにＲＦＡ piece, long receive パケットがキューonされることにより起動される。

起動されると long exit pipe ステータスを調べ，有効なステータスならば，8バッファを確保し，pipe controll table へセットするとともに long exit pipe ステータスを所定の値に変更する。

有効なステータスでなければ起動要因となったＲＦＡ pieceあるいは long receive パケットを捨ててターミネイトする。

(注) 有効なステータス

起動要因	有効ステータス
ＲＦＡ piece	1（休　止）
long Receive Packet	4（HOSTよりの Receive 待ち）

図３－29

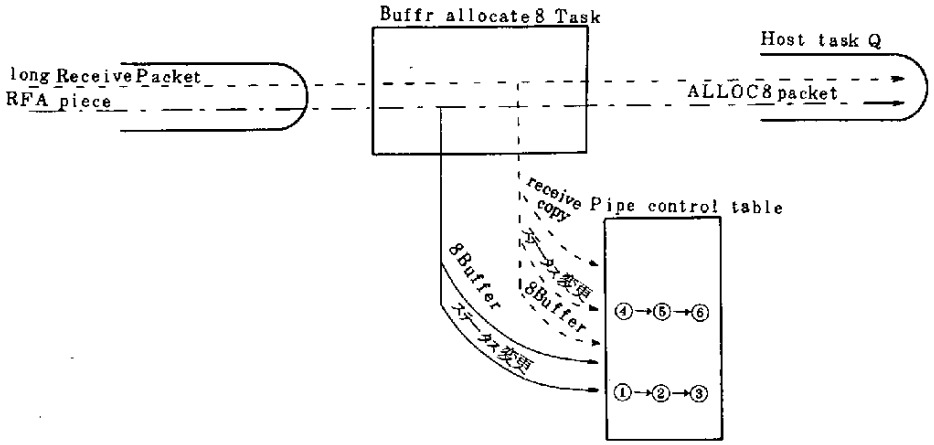


図３－30

J. ＴＴＹ out タスク

ＩＭＰに接続している入出力タイプライターにメッセージを出力するタスクである。

このタスクは，Console キューまたは，ＴＴＹ outキューにキューon されることにより起動される。

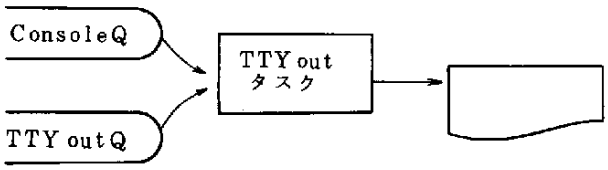


図３ －31　ＴＴＹ　out の関連するキュー

○　Consoleキューにキューonされるバッファの形式

このキューにのせられるバッファは，他ＩＭＰから自己Fake HOST宛に向けられたパケットであり図３－32の形式に従っていなければならない。

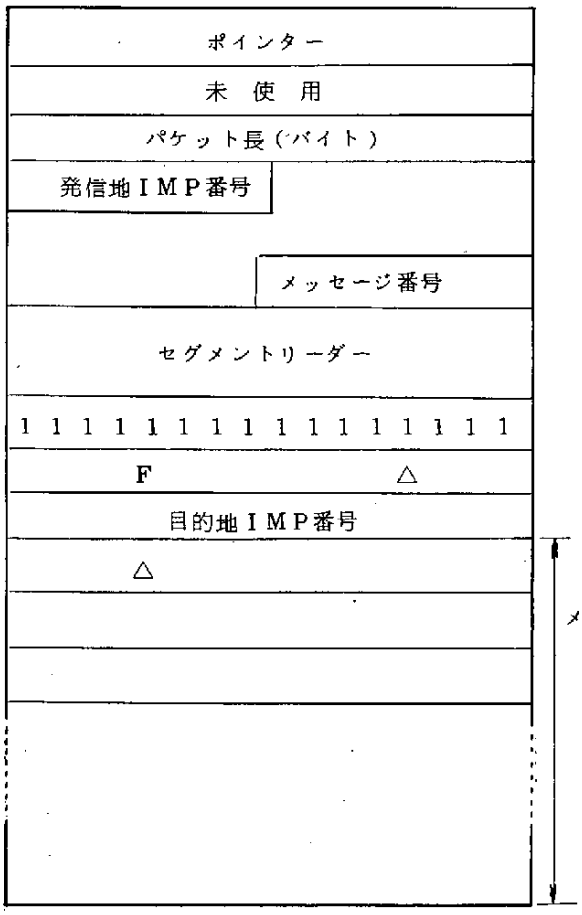
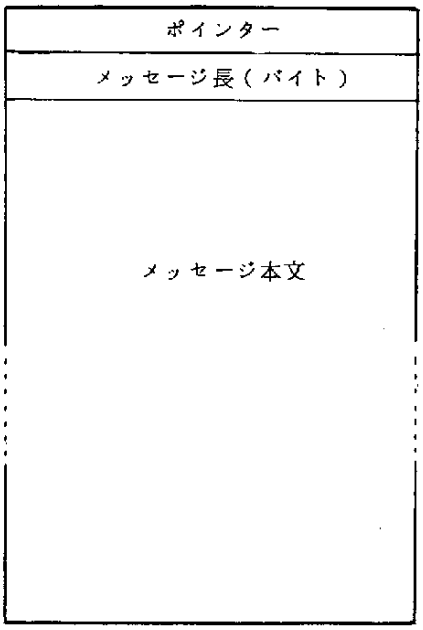


図３ －32

○　ＴＴＹ out キューにキュー on　されるバッファの形式

このキューにのせられるバッファは，ＩＭＰ内循環系バッファを利用しバッファの内容は，図3 －33のようになっていなければならない。



メッセージ本文は，バッファ長により決められた最大文字数まで可能であるが72文字目を印字した後自動的にnew lineとなるので注意しなければならない。

図3 －33

K．Single buffer free タスク

このタスクは発信地が自分であるsingle packet message のバッファをフリーにする機能をはたす。

このタスクの起動要因はＦＰキュー，ＦＭＧキューに対するキュー onである。ＦＰキューはpipe管理タスクがsingle packet messageの receive を受け取ったときバッファの先頭＋81番地をチェーン領域としてキュー on される。

ＦＭＧキューはmodem out タスクが発信地が自分であるsingle packet message のＡＣＫを受け取ったとき，バッファの先頭をチェーン領域としてキュー on される。

通常の状態では，必ずＦＰキューにキュー onされるバッファとＦＭＧキューにキュー onされるバッファは同一のものがあるはずであるから，同一のものがでてきた時点でバッファのフリー操作をする。

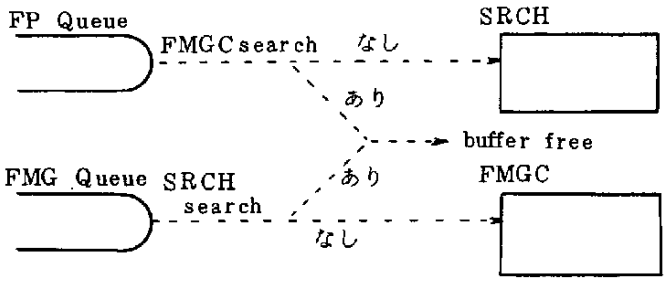


図３－34

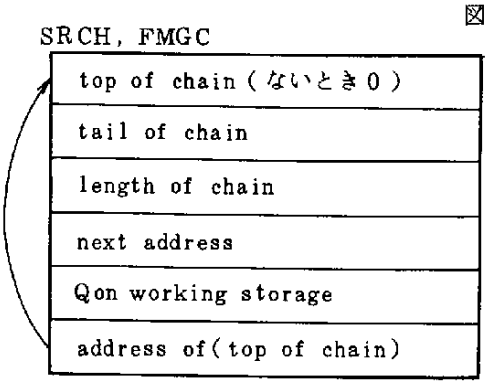


図３－35

ＳＲＣＨはバッファの先頭＋81番地を使用してチェーンされ，ＦＭＧＣはバッファの先頭を使用してチェーンされる。

ＳＲＣＨはＦＰキューにキューon されたバッファと同一のものがＦＭＧＣにチェーンされていないときにのせられるチェーンであり，ＦＭＧＣはＦＭＧチェーンにキューon されたバッファと同一のものがＳＲＣＨにチェーンされていないときにのせられるチェーンである。

L. Back ground タスク

このタスクは，下記に示すコマンドの処理を行なう。これらのコマンドをＩＭＰに接続された入出力タイプライターからタイプ・インすることによりＩＭＰ／ＴＩＰに対し動作の指示を行なうことができる。

下記のコマンドの利用にあたっては，入出力タイプライターから(ESC)キーをタイプ・インするとＩＭＰ側から@キーを打ち返してくるので，続けてコマンドをタイプ・インする。

○　ＩＭＰに対するコマンド

ＩＭＰに対するコマンドは，

Ｉ△×××．．．．×(CR)

の形式である。

(1) 日付のセット

I△D△××××(CR)

月 日

月，日の指示は，0をサプレスしてはいけない。

(2) 時刻のセット

I△T△××××××(CR)

時 分 秒

時，分，秒の指示は0をサプレスじてはいけない。また時に関しては24時間法をもちいること。

(3) 他IMPとの交信（タイプライターからタイプライター）

I△F△##△message(CR)

相手IMP番号 送信すべきメッセージ

相手IMP番号は1～16までの10進数

送信すべきメッセージは，71文字以下でなければならない。

(4) コアーの内容の変更

I△A△××××××，××××××(CR)

変更番地 変更すべき内容

8進数で示された変更番地の内容を変更すべき内容で示されたものでおきかえる。

(5) コアーの内容の参照

I△R△××××××(CR)

参照番地

8進数で示された参照番地の内容を8進数でタイプ・アウトする。

その型式は，次のとおりである。

××××××△YYYYYY

参照番地 内容

(6) IMPのサービス終了

I△END(CR)

○ TIPに対するコマンド

TIPに対するコマンドは

T△××××．．．．×(CR)

の形式である。

(1) TIPのサービス開始

T△START(CR)

(2) ＴＩＰのサービス終了

T△END (CR)

このタスクは，入出力タイプライターからＩＭＰバッファに打鍵されたメッセージがＩＭＰバッファの形でＴＴＹ inキューにキューonされた場合，および他ＩＭＰから自己Fake HOST宛に送られたメッセージがFake HOSTキューにキューon された場合に起動される。

○ ＴＴＹ inキュー

ＩＭＰに用意されているサービス・コマンドとして，日付セット・コマンド，時刻のセット・コマンド，コアーの内容の変更コマンド，コアーの内容の参照コマンド，ＴＩＰのサービス開始および終了コマンドに関してコマンド解析を行ない，必要があれば，ＩＭＰバッファに解析結果を作成してＴＴＹout キューにキューonする。

他ＩＭＰとの交信コマンドに関しては，他ＩＭＰのFake HOST宛に作成されたメッセージをパケット化しパイプ待ちキューにキューonする。

○ Fake HOSTキュー

他ＩＭＰから自己ＩＭＰのFake HOST宛に送られたパケットのフォーマットチェックを行なう。

チェックの結果正しければ，そのバッファをConsoleキューにキューonする。

誤りであれば，そのバッファをreceive パケットに作り変えshort Rec キューにキューonする。

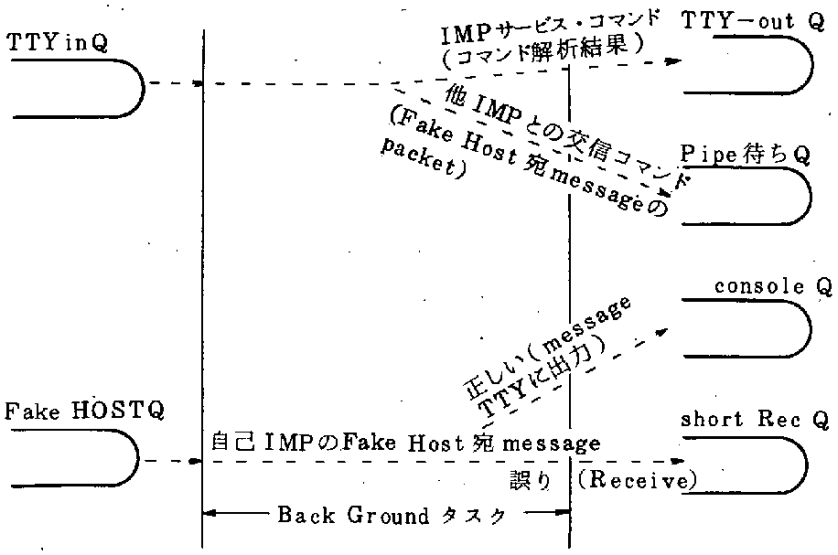


図３ −36　Back Groundタスクの起動

3.3 サブネットの性能測定

システムの開発に於て，その機能，性能を測定するということは重要な仕事である。JIPNETにおいても，コンピュータネットワークのメッセージ伝送などに関して測定をおこなっているが，ここでは，サブネットだけを使用した測定法およびその結果について記述する。

測定の目的は，サブネットのスループット（throughput）とメッセージのラウンドトリップタイム（round trip time）である。

3.3.1 測定方法

本来サブネットはHOSTよりメッセージを受け取って指定されたHOSTに正確にメッセージを渡たすのが役割りであるので，自分自身でメッセージを発生することはない。このためにサブネット中のプログラムに変更を加えてメッセージを発生する機能，receive を作成する機能を追加した。

メッセージを発生する機能をはたすプログラムをartificial traffic generator という。これは200msecあるいは100msec毎に任意のsingle packet massage を発生することができる。

この機構はtime outタスク中に組み込まれており，fast タイムアウトがかかると，バッファの余裕があるかぎり（バッファの残り個数が30個になるまで）指定された個数ずつのメッセージを作り出して，パイプ待ちキューにキューonする。

このメッセージの行き先は指定されたＩＭＰであり，これを受け取るのはＨＯＳＴ／ＩＭＰ３タスクである。このタスクではメッセージを受け取ると，受け取ったメッセージ数をカウントし，receiveを作成して送り返す。

送り返えされたreceiveは，メッセージを発信したＩＭＰのパイプ管理において，処理が終わり次第捨てられる。

スループットの測定は，ＨＯＳＴ／ＩＭＰ３タスクがおこなっている受け取ったメッセージのカウンタを0にして，100秒後に再びカウンタを参照することによりおこなった。100 秒の計測はストップウオッチでおこなったが，この測定の誤差は±0.2％以内である。

ラウンドトリップタイムはサブネットにおいては，メッセージがパイプにはいってからreceiveを受け取り，ＨＯＳＴ／ＩＭＰタスクにキューon するまでの時間と定義して測定した。

この測定は，パイプ管理がメッセージをパイプに入れる時の時刻を（ハードウエアタイマーを利用しているのでm secまで計測できる）憶えておき，対応するreceive を受け取りＨＯＳＴ／ＩＭＰタスクにキューon しようとするときの時刻と比較し，その差をカウンタに加算して入れる。このとき他のカウンタを＋１する。

これらのカウンタをある時点で０にして，４秒後に両者を参照することにより計測した。

2.3.2 測定結果

測定にあたっては，次のパターンを設定し，それぞれのパターンでおこなった。

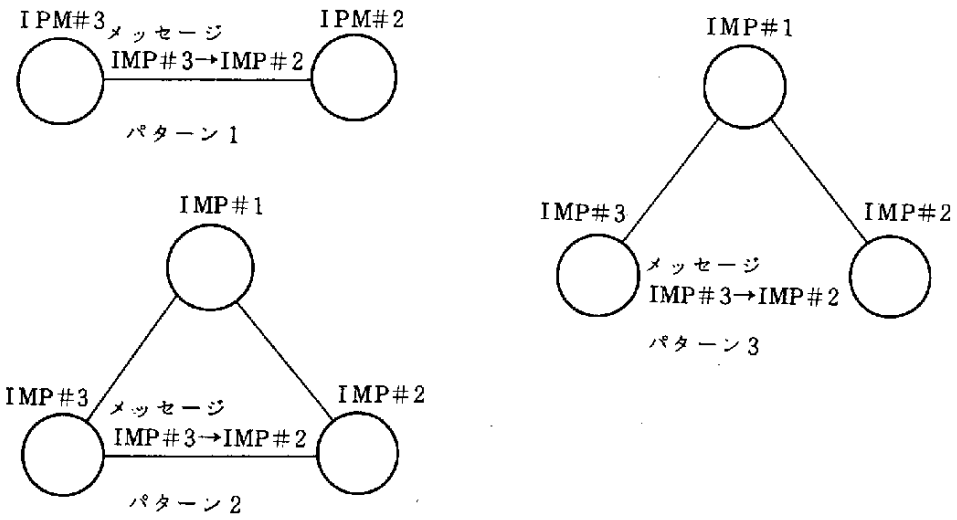


図3－37　測定パターン

A．スループットの測定

a．Single packet message

それぞれのパターンに対する測定結果は（表3－1）～（表3－12）までのとおりである。この表中でオーバーヘッドを含むスループットとは，ソフトウエアオーバーヘッド12バイト，ハードウエア・オーバーヘッド3バイト合計15バイトをメッセージサイズに加算し，これを送った時（すなわち回線を通る情報全体）のスループットである。

これらの表をグラフ化したのが（図3－38）～（図3－41）である。

このグラフによれば10バイトメッセージのように短いメッセージにおいては，サブネットのフローコントロール機構によって伝送量がおさえられるが，メッセージが長くなるにしたがって48 K bit／sec の回線容量によっておさえられるのがわかる。

また，パスが1個のものと2個のものを比較すると，パスが1個のものが回線バウンドになっても，パスが2個の場合は目的地にゆくためのルートがルーティングにより分流され，フロ

ーコントロールの機構でおさえられたままである。

（図３－41）において，10バイトメッセージは，パターン１がスループットが一番大きく，次にパターン２，最後がパターン３である。フローコントロールにより伝送量がおさえられている場合には，メッセージのラウンドトリップタイムが小さい方が伝送量は大きくなるので，パターン３が他のものよりスループットが小さいのは納得できるが，パターン１がパターン２より大きい点に興味がある。

これに関しては現在のところ必ずしも明確な理由づけは得られていないが，しかし考え得ることとしては，再伝送のアルゴリズムを変えると，この状態が若干変化するところから，この測定の過程で現実には再伝送がかなり発生し，そのアルゴリズムが影響しているのではないかという事と，modem out タスクの優先度がやや低いので，パターン２に於て，経路が２つになると，割り込みが多発し，そのオーバーヘッドが大きくなるため，modem out タスクに対してＣＰＵの供給が充分おこなわれないせいではないか等の理由が考えられる。

表３－１　throughput - pattern 1 －10バイトメッセージ

項目 ＼ 発生個数個数／秒	10	40	80	120	160	170	180	200
伝送個数	10	40	80	120	160	170	175	174.9
throughput kbit/sec	0.8	3.2	6.4	9.6	12.8	13.6	14	14
throughput kbit/sec（含む overhead）	2.0	8.0	16	24	32	34	35	35

表３－２　throughput - pattern 1 －50バイトメッセージ

項目 ＼ 発生個数個数／秒	10	20	40	80	90	120	140
伝送個数	10	20	40	80	90	90	90
throughput kbit/sec	4.0	8.0	16	32	36	36	36
through put kbit/sec（含む overhead）	5.2	10.4	20.8	41.6	46.8	46.8	46.8

表３－３　throughput - pattern 1 －100バイトメッセージ

項目 ＼ 発生個数個数／秒	10	30	50	60	80
伝送個数	10	30	50	52	52
throughput kbit/sec	8	24	40	41.6	41.6
throughput kbit/sec（含む overhead）	9.2	27.6	46	47.8	47.8

表３－４　throughput - pattern 1 － 144バイトメッセージ

項目 ＼ 発生個数 個数／秒	10	20	40	60	80
伝送個数	10	20	37.6	37.6	37.6
throughput kbit／sec	11.52	23.04	43.3	43.3	43.3
throughput kbit／sec（含むoverhead）	12.72	25.44	47.8	47.8	47.8

表３－５　throughput - pattern 2 － 10バイトメッセージ

項目 ＼ 発生個数 個数／秒	10	20	40	80	150	160	180	200
伝送個数	10	20	40	80	150	160	172.2	172.1
throughput kbit／sec	0.8	1.6	3.2	6.4	12	12.8	13.8	13.8
throughput kbit／sec（含むoverhead）	2.0	4.0	8.0	16	30	32	34.4	34.4

表３－６　throughput - pattern 2 － 50バイトメッセージ

項目 ＼ 発生個数 個数／秒	10	20	40	80	100	110	120	140
伝送個数	10	20	40	80	100	105.3	105.5	105.6
throughput kbit／sec	4.0	8.0	16	32	40	42.1	42.2	42.2
throughput kbit／sec（含むoverhead）	5.2	10.4	20.8	41.6	52	54.8	54.9	54.9

表３－７　throughput- pattern 2 － 100バイトメッセージ

項目 ＼ 発生個数 個数／秒	10	20	40	60	70	80	100
伝送個数	10	20	40	60	66.7	66.7	66.7
throughput kbit／sec	8	16	32	48	53.4	53.4	53.4
throughput kbit／sec（含むoverhead）	9.2	18.4	36.8	55.2	61.4	61.4	61.4

表３－８　throughput - pattern 2 － 144バイトメッセージ

項目 ＼ 発生個数 個数／秒	10	20	40	50	60	80
伝送個数	10	20	40	50	50	50
throughput kbit／sec	11.52	23.04	46.08	57.6	57.6	57.6
throughput kbit／sec（含むoverhead）	12.72	25.44	50.88	63.6	63.6	63.6

表３－９　throughput - pattern ３－10バイトメッセージ

項目 ＼ 発生個数 個数／秒	10	40	80	120	130	160	200
伝送個数	10	40	80	120	128.8	129.1	129.1
throughput kbit/sec	0.8	3.2	6.4	9.6	10.3	10.3	10.3
throughput kbit/sec（含むoverhead）	2.0	8.0	16	24	25.8	25.8	25.8

表３－10　throughput - pattern ３－50バイトメッセージ

項目 ＼ 発生個数 個数／秒	10	20	40	80	90	120	160
伝送個数	10	20	40	80	88.5	88.5	88.9
throughput kbit/sec	4.0	8.0	16	32	35.4	35.4	35.6
throughput kbit/sec（含むoverhead）	5.2	10.4	20.8	41.6	46.0	46.0	46.2

表３－11　throughput - pattern ３－100バイトメッセージ

項目 ＼ 発生個数 個数／秒	10	30	50	60	80
伝送個数	10	30	50	51.8	51.9
throughput kbit/sec	8	24	40	41.4	41.5
throughput kbit/sec（含むoverhead）	9.2	27.6	46	47.7	47.7

表３－12　throughput - pattern ３－144バイトメッセージ

項目 ＼ 発生個数 個数／秒	10	20	40	80
伝送個数	10	20	37.6	37.6
throughput kbit/sec	11.52	23.04	43.3	43.3
throughput kbit/sec（含むoverhead）	12.72	25.44	47.8	47.8

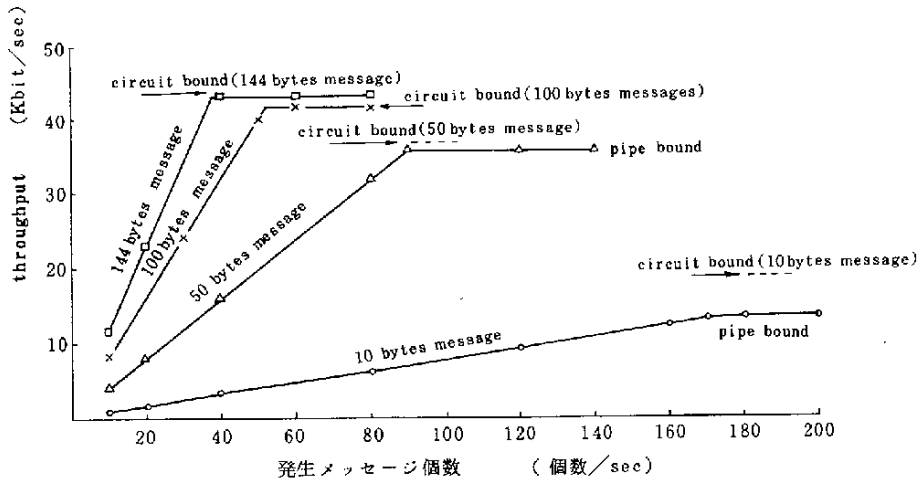


図3 −38 Throughput - pattern 1

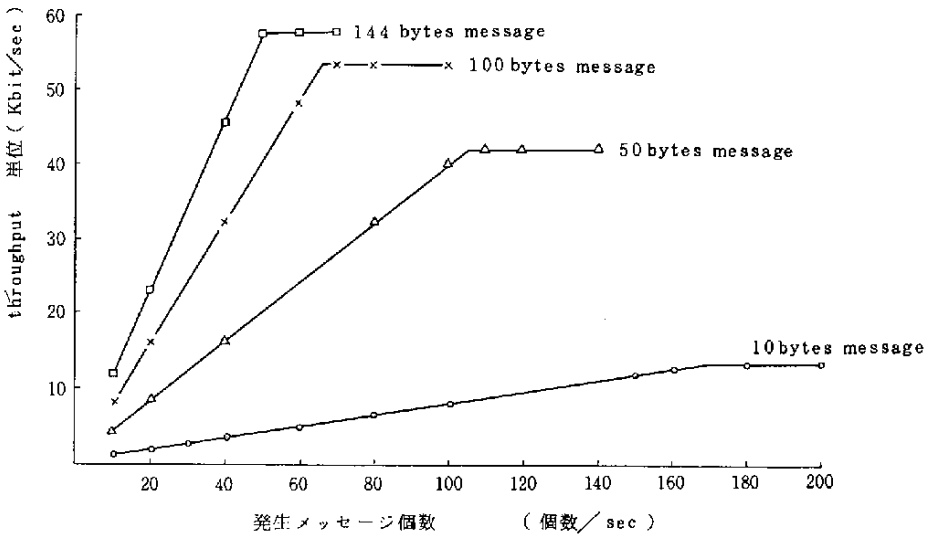


図3 −39 Throughput - pattern 2 (目的地までの path が 1 Hop, 2 Hop の 2 個)

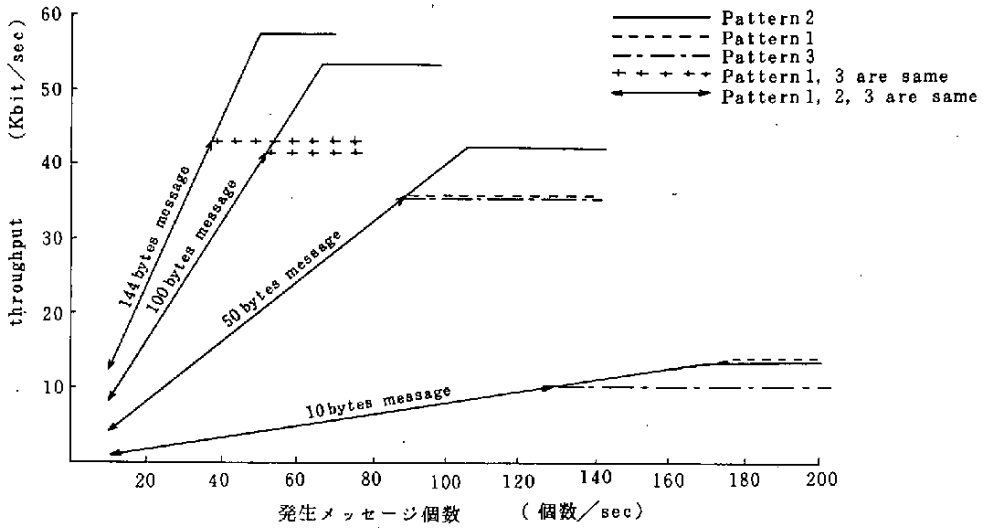


図 3 － 40　Throughput - pattern 3 （目的地まで 2 Hop の path が 1 個）

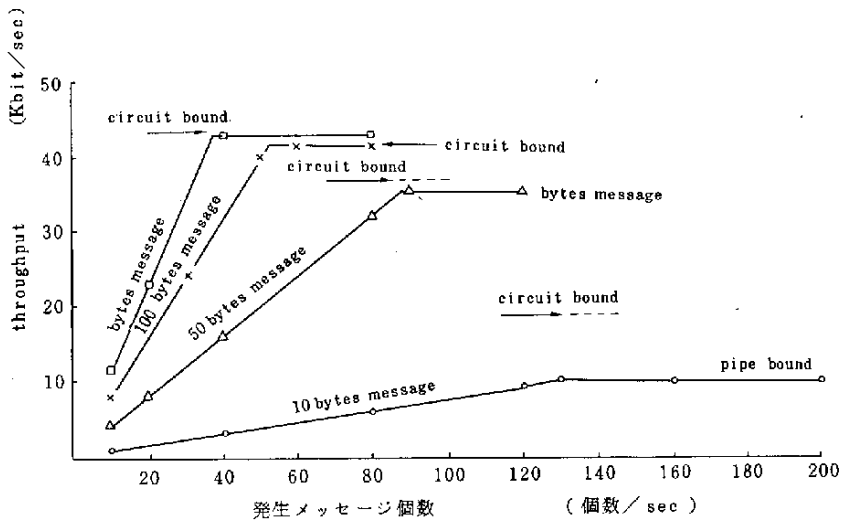


図 3 － 41　Throughput - all patterns

b．Multi packets massage

それぞれのパターンに対する測定結果を（表３－13）～（表３－36）に示す。この表でスループットは１パケット 144 バイトで計算してある。

パターン２に於てＩＭＰ３からＩＭＰ２に伝送される８パケットメッセージはルーティング機構により図３－42のように分流される。１つのメッセージのみを考えると２パケットメッセージまではパターン１とパターン２は同じスループットを示す。図３－43 は各パターンに対するメッセージ長とスループットのグラフである。

パターン２に於てパケット数が３，５，７の場合スループットが急に上昇するのは，図３－42 でわかる様に２つの経路が使用されることの効果である。またパケット数が８になると７の場合より下がるのは長いメッセージのラウンドトリップの影響であろう。

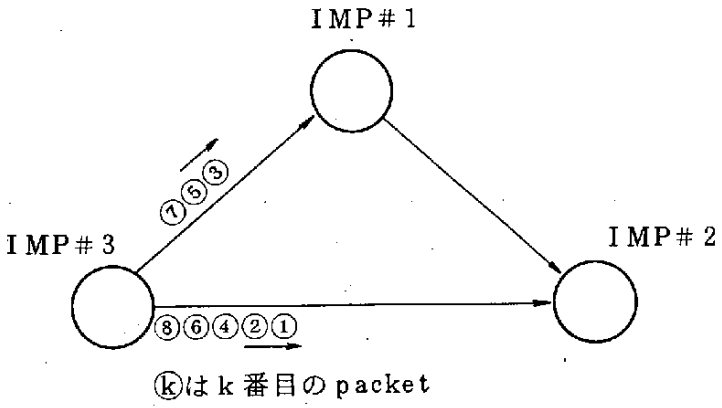


図３－42　パケットの分流

表３－13　Pattern 1　8 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	4.4	4.4	4.4	4.4	4.4	4.4	4.4	4.4	4.4	4.4	4.4		
throughput kbit/sec	41.5	41.5	41.5	41.5	41.5	41.5	41.5	41.5	41.5	41.5	41.5		

表３－14　pattern 1　7 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0		
throughput kbit/sec	40.3	40.3	40.3	40.3	40.3	40.3	40.3	40.3	40.3	40.3	40.3		

表３－15　Pattern 1　6 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	5.8	5.8	5.8	5.8	5.8	5.8	5.8					
throughput kbit/sec	34.6	40.1	40.1	40.1	40.1	40.1	40.1	40.1					

表３－16　Pattern 1　5 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	6.9	6.9	6.9	6.9	6.9	6.9	6.9					
throughput kbit/sec	28.8	39.7	39.7	39.7	39.7	39.7	39.7	39.7					

表３－17　Pattern 1　4 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	8.4	8.4	8.4	8.4	8.5	8.5	8.5	8.5	8.5			
throughput kbit/sec	23.0	38.7	38.7	38.7	38.7	39.2	39.2	39.2	39.2	39.2			

表３－18　Pattern 1　3 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	10.0	10.7	10.9	10.8	10.7	10.7	10.8	10.7	10.8			
throughput kbit/sec	17.3	34.6	37.0	37.7	37.3	37.0	37.0	37.3	37.0	37.3			

表3－19　Pattern 1　2 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	10.0	15.0	15.5	15.5	15.5	15.6	15.6	15.5	15.5			
throughput kbit／sec	11.5	23.0	34.6	35.7	35.7	35.7	35.9	35.9	35.7	35.7			

表3－20　Pattern 1　1 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	10.0	15.0	20.0	25.0	26.2	26.1	26.1	26.1	26.0	26.0	26.0	26.0
throughput kbit／sec	5.8	11.5	17.3	23.0	28.8	30.2	30.1	30.1	30.1	30.0	30.0	30.0	30.0

表3－21　Pattern 2　8 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	6.8	6.8	6.9	6.8	6.8	6.8						
throughput kbit／sec	46.1	62.7	62.7	63.6	62.7	62.7	62.7						

表3－22　Pattern 2　7 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	7.9	7.8	7.9	7.9	7.9	7.9						
throughput kbit／sec	40.3	63.7	62.9	63.7	63.7	63.7	63.7						

表3－23　Pattern 2　6 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	8.4	8.4	8.4	8.4	8.4	8.4	8.4	8.4	8.4			
throughput kbit／sec	34.6	58.1	58.1	58.1	58.1	58.1	58.1	58.1	58.1	58.1			

表3－24　Pattern 2　5 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	10.0	10.0	10.0	10.0	10.1	10.0	10.1	10.1	10.1			
throughput kbit／sec	28.8	57.6	57.6	57.6	57.6	58.2	57.6	58.2	58.2	58.2			

表 3 － 25　Pattern 2　4 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	10.0	10.8	10.8	10.8	10.9	10.8	10.8	10.8	10.8			
throughput kbit/sec	23.0	46.1	49.8	49.8	49.8	50.2	49.8	49.8	49.8	49.8			

表 3 － 26　Pattern 2　3 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	10.0	13.9	13.9	14.0	14.0	13.9	14.0	14.0	14.0			
throughput kbit/sec	17.3	34.6	48.0	48.0	48.4	48.4	48.0	48.4	48.4	48.4			

表 3 － 27　Pattern 2　2 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	10.0	15.0	15.5	15.5	15.5	15.5	15.5	15.6	15.5			
throughput kbit/sec	11.5	23.0	34.6	35.7	35.7	35.7	35.7	35.7	35.9	35.7			

表 3 － 28　Pattern 2　1 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	10.0	15.0	20.0	25.0	26.0	26.0	26.1	26.1	26.0	26.2	26.2	26.2
throughput kbit/sec	5.8	11.5	17.3	23.0	28.8	30.0	30.0	30.1	30.1	30.0	30.2	30.2	30.2

表 3 － 29　Pattern 3　8 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	3.9	3.9	3.9		3.9	3.9							3.9
throughput kbit/sec	35.9	35.9	35.9		35.9	35.9							35.9

表 3 － 30　Pattern 3　7 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	4.3	4.3	4.3	4.3	4.3	4.3	4.3						5.0
throughput kbit/sec	34.7	34.7	34.7	34.7	34.7	34.7	34.7						40.3

表３－31　Pattern 3　6 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	4.9	4.9	4.9	4.9	4.9		4.9						4.9
throughput kbit/sec	33.9	33.9	33.9	33.9	33.9		33.9						33.9

表３－32　Pattern 3　5 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.6	5.6	5.6	5.6	5.6	5.6	5.6						5.6
throughput kbit/sec	28.8	32.3	32.3	32.3	32.3	32.3	32.3						32.3

表３－33　Pattern 3　4 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	6.7	6.7	6.7				6.7	6.7				6.7
throughput kbit/sec	23.0	30.9	30.9	30.9				30.9	30.9				30.9

表３－34　Pattern 3　3 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	8.1	8.1	8.1				8.1	8.1				8.1
throughput kbit/sec	17.3	28.0	28.0	28.0				28.0	28.0				28.0

表３－35　Pattern 3　2 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	10.0	10.4	10.4	10.4			10.4					10.4
throughput kbit/sec	11.5	23.0	24.0	24.0	24.0			24.0					24.0

表３－36　Pattern 3　1 Packet

発生個数／秒	5	10	15	20	25	30	35	50	75	100	150	200	250
伝送個数	5.0	10.0	14.4	14.4	14.4			14.4					14.4
throughput kbit/sec	5.8	11.5	16.6	16.6	16.6			16.6					16.6

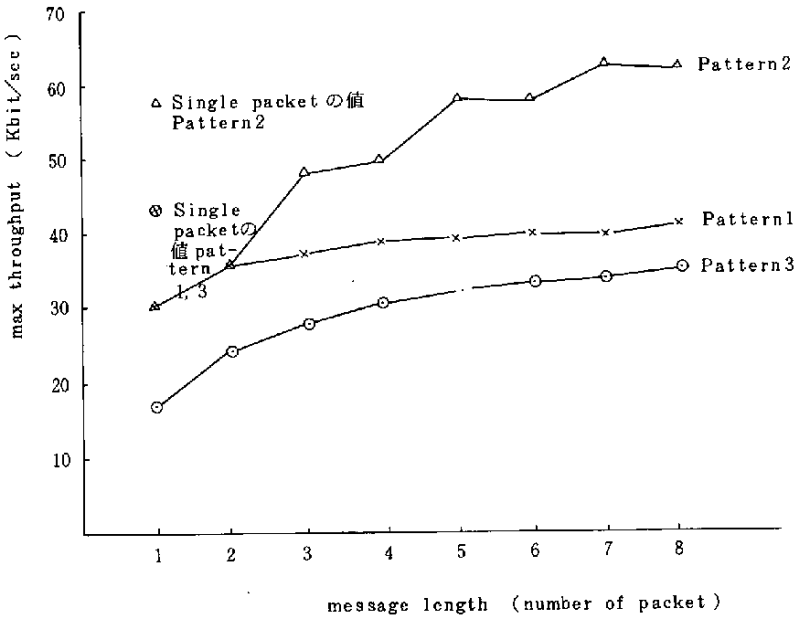


図３－43　throughput-multi packets message

B. Single packet message ラウンドトリップタイムの測定

それぞれのパターンに対する測定結果は（表３－37）～（表３－39）までのとおりである。この表中で，最大スループット時のラウンドトリップタイムの推定値は，次の式により計算した。

$$\text{Estimate max} = \frac{4000}{\text{最大 throughput 時の秒あたりの message 個数}} \text{ m sec}$$

それぞれの表をグラフにしたのが（図３－44）～（図３－46）までである。これらのグラフで興味をひかれるのは，パターン３のとき最大スループット時の予測値の方が 200 m sec に４メッセージのラウンドトリップタイムよりも小さい点である。これはサブネットにおいてのラウンドトリップタイムをパイプにはいってから receive が返るまでの時間と定義したことによる。すなわち，200 m sec に４メッセージであると，１度にパイプにはいるため，４番目のメッセージは回線のチャネル待ちの時間が長くかかる。そして 200 m sec で４個のメッセージは伝送できてしまう。このため毎回４番目のメッセージのラウンドトリップタイムが大きくなる。他方，最大スループット時には，通常 receive が返ってきてから，次のメッセージが送られるが，このときは，少くとも３個より少い個数しか回線のチャネル待ちになっていない。すなわち最大スループット時にはメッセージが４個単位にはいってくるのではなく，ほぼ一定の時間間隔でメッセージがパイプにはいるために，すでに出力待ちになっているメッセージ個数の平均値が小さくなる。

表３－37　round trip time －pattern 1

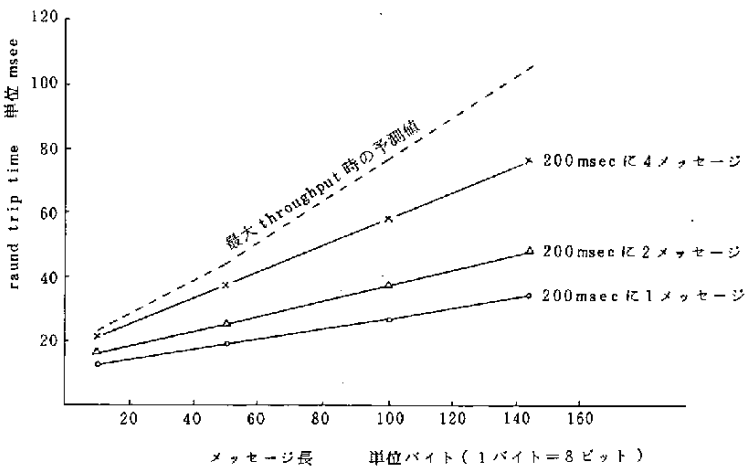
メッセージ個数 / メッセージ長（バイト）	200 msecに1個	200 msecに2個	200 msecに4個	最大 throughput 時（計算値）
10	12.0 msec	16.1msec	20.8msec	22.9msec
50	19.0	24.9	37.2	44.4
100	27.0	37.4	58.2	76.9
144	34.9	48.2	76.5	106.4

表３－38　round trip time －pattern 2

メッセージ個数 / メッセージ長（バイト）	200 msecに1個	200 msecに2個	200 msecに4個	最大 throughput 時（計算値）
10	12.0 msec	16.3 msec	22.5 msec	23.2 msec
50	19.0	25.0	34.2	37.9
100	27.1	37.7	50.7	60.0
144	35.0	48.6	65.6	80.0

表３－39　round trip time －pattern 3

メッセージ個数 / メッセージ長（バイト）	200 msecに1個	200 msecに2個	200 msecに4個	最大throughput 時（計算値）
10	21.4 msec	25.9 msec	33.1 msec	31.0 msec
50	35.0	40.9	54.4	45.2
100	51.7	61.9	83.5	77.1
144	66.0	80.4	109.4	106.4



３－44　message の round trip time －pattern 1

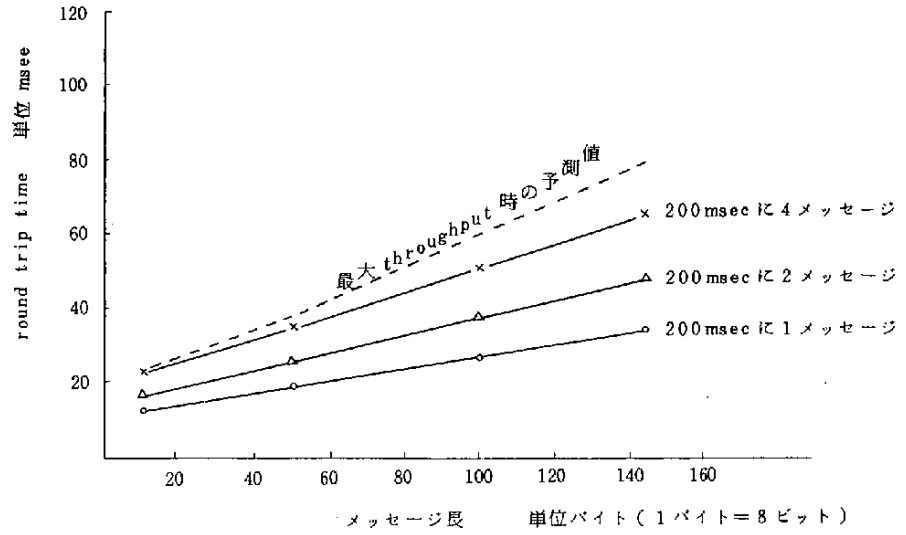


図3－45　messageのround trip time－pattern 2

（目的地まで1Hop，2Hopの2つのpath）

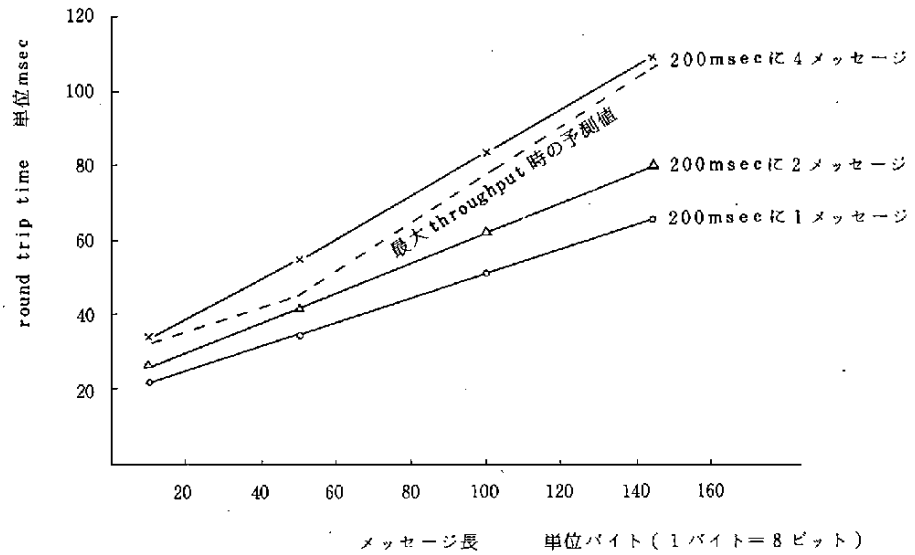


図3－46　messageのround trip time－pattern 3

（目的地まで2Hopのpathが1個）

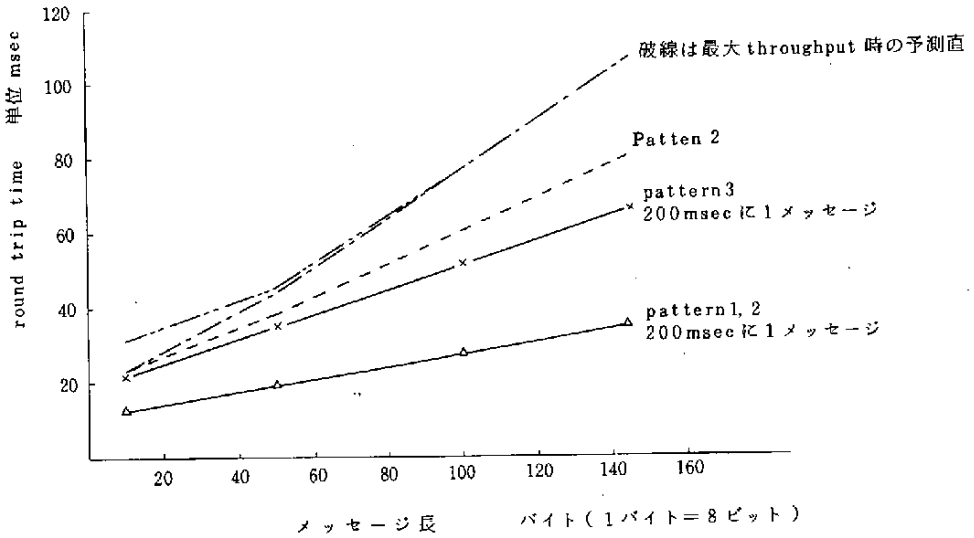


図３－47　message の round trip time

測定したデータから，ＩＭＰが１パケットを蓄積交換をするに必要な時間を推定することができる。すなわち（図３－47）の200ｍsecに１メッセージを送った場合のパターン１，パターン３の両グラフを比較する。そして，ＩＭＰの内部でどのような経路を通っていたかを調べることにより可能である。（図３－48），（図３－49）はそれぞれパターン１，パターン３の処理経過を示している。ここでそれぞれの過程でかかる時間を表３－40のようにおいてパターン１とパターン３の伝送時間の差をT(s)とすれば

$$T(s) = Tmd(s) + Tmr(s) + 2 \cdot Tmin + 2 \cdot Tgen + 3 \cdot Tmout$$

Tmd(s)，Tmr(s) はそれぞれ計算でててくるので，$2 \cdot Tmin + 2 \cdot Tgen + 3 \cdot Tmout$ を求めることができる。

蓄積交換に必要な時間（TS/F）は本来ならばTS/F＝Tmin＋Tgen＋Tmout であるが，ヌルパケットが発信される可能性もあるのでここでは

$$TS/F = Tmin + Tgen + 1.5\,Tmout$$

とすることにした。これによればTS/F≒1.16 msecとなる。

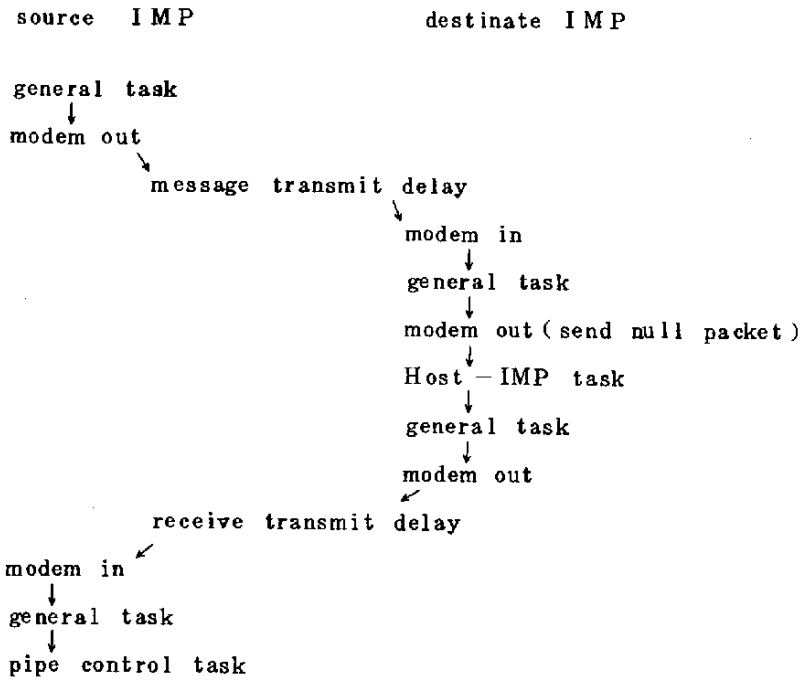


図３－48

表３－40

general task走行時間	:	Tgen	Msec
modem in 〃	:	Tmin	〃
modem out 〃	:	Tmout	〃
Host －IMP 〃	:	Th－i	〃
pipe control 〃	:	Tpipe	〃
message transmit delay	:	Tmd(s)	〃
receive 〃	:	Tmr	〃

Tmd(s)のSはmessage長を示す。

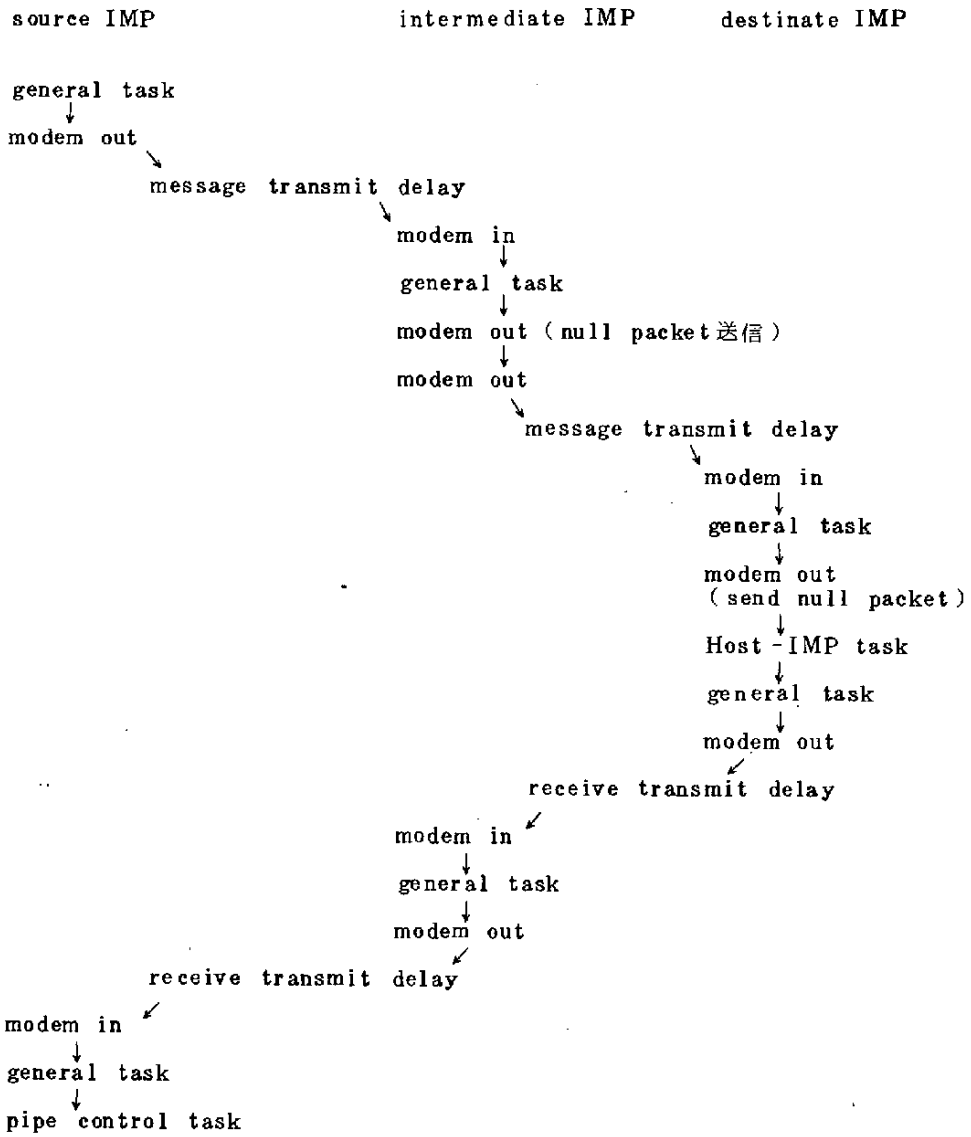

source IMP
intermediate IMP
destinate IMP
general task
modem out
message transmit delay
modem in
general task
modem out (null packet送信)
modem out
message transmit delay
modem in
general task
modem out
(send null packet)
Host-IMP task
general task
modem out
receive transmit delay
modem in
general task
modem out
receive transmit delay
modem in
general task
pipe control task

図3－49

3.4 作成上の諸問題

3.4.1 ソフトウエアの構造

ＳＣＰ作成にあたって考慮したのは次の点である。

(1) プログラム構造を単純化する。

(2) 機能の追加修正が簡単である。

(3) 短期間にインプリメントできる。

(4) プログラムのモジュール化ができる。

(5) ＨＳＣＬからの入力をタイミングよく処理できる。

この結果としてできあがったのが現在のＳＣＰである。(1)～(4)はプログラムの作成にかかる期間が約４ケ月間であり，出来得る限り，システムの早期安定をはかること，且つ論理的な誤りや不備な点を発見しやすい構造にする必要があったからである。(5)は，高速回線の性能を生かすための要求である。

ソフトウエア構造として，我々がとったような，ＣＰＵディスパッチャーが各タスクにＣＰＵをあたえる方式と割り込みを起点として割り込み原因を起こしたタスクにＣＰＵを渡たし，優先度の高いタスクは，低いタスクの割り込み原因をマスクする方式が考えられる。

前者は，大形コンピュータでユーザータスクを管理するときによくもちいられている。

後者はたとえばプロセスコントロールなどのように割り込み原因とタスクの関係がはっきり決っている場合によくもちいられる手法である。

我々が前者を採用したのは，(2)の機能の追加修正がきわめて簡単だからである。もし後者を採用したならば，予定期日にできあがらなかったであろうと確信している。

現在のＳＣＰの欠点としてＣＰＵを使いすぎるのではないかということがある。

たとえばＩＭＰが毎秒250パケットの伝送処理を行う場合，ＣＰＵがback groundの処理を行っている時間は測定によれば全体の16％である。

しかし，残り74％のうち，約半分（全体の35％）はＩＭＰのコントロールプログラムが消費しており，夫々のタスクがＩＭＰ本来の処理を行なうために使用するＣＰＵの利用率は全体の30～40％である。

１個の割り込み毎にCPUがback groundタスクに渡たるまでにコントロールプログラムで費される時間は現在のプログラムでは毎回385 μseeである。また1パケットを処理するために発生する割り込みは３回であり，250パケットを処理するには毎秒750回の割り込みが発生していることとなる。

これらの結果から，現在のシステムは当初の予想以上に割り込みが多発し，且つＣＰＵの有効利用率が低く，コントロールプログラム形式を採用した事は，システムの早期の安定化には極めて有

効であったが，効率の点からは，割り込みオリエンテッドにＣＰＵを与え，より低い優先度の割り込みを禁止する方式が望ましいと言える。

3.4.2　デバッグ方法

コンピュータネットワークの開発は，多数のグループによっておこなわれるために，多くのトラブルが予想された。このために，プログラムのデバッグ過程においてのエラーの切り分けのために，メッセージダンプとＮＤＰの２つのプログラムを作成した。

メッセージダンプは，ＩＭＰがHOSTに対して入出力したデータをコアー上に残しておくもので，ＩＭＰ＃１において，FacomとHitac がやりとりしたメッセージの新しいもの約200個分を保存することができる。これは，ＩＭＰ＃１においてHOSTに対して入出力命令をだすたびに，そのデータをＩＭＰ＃２に送り，そこでこれを保存しておく方法をとった。

Ｈ－ＮＣＰ，Ｆ－ＮＣＰ，ＩＭＰグループ間のトラブル切り分けはほとんどこれによって解決することができた。

ＮＤＰは，Ｎ－ＮＣＰ，Ｆ－ＮＣＰのプログラム・デバッグ・スケジュールが一致していなくとも，それぞれのＮＣＰの基本的デバッグができるように作成したものである。

これの機能は，ＮＣＰをシュミレートするもので，ＮＣＰが何かをＮＤＰに送ってきたならば，この返答をＰＴＲから読み込んで，その内容をＮＣＰに送り返す。

3.4.3　アルゴリズム上の問題

A．タスクの優先順序

ＳＣＰにおいてタスクの優先順序は非常に重要である。これはＩＭＰの性能に直接影響をあたえるからである。現時点においては，ＩＭＰの性能に関してはっきりした値はでていないので，現在使用しているアルゴリズムから，このような優先順序にした理由を述べるにとどめる。

現在使用している優先順になったのは，回線断の処理に対する対応策からである。

回線の断のときには図３－50の手順がとられる。

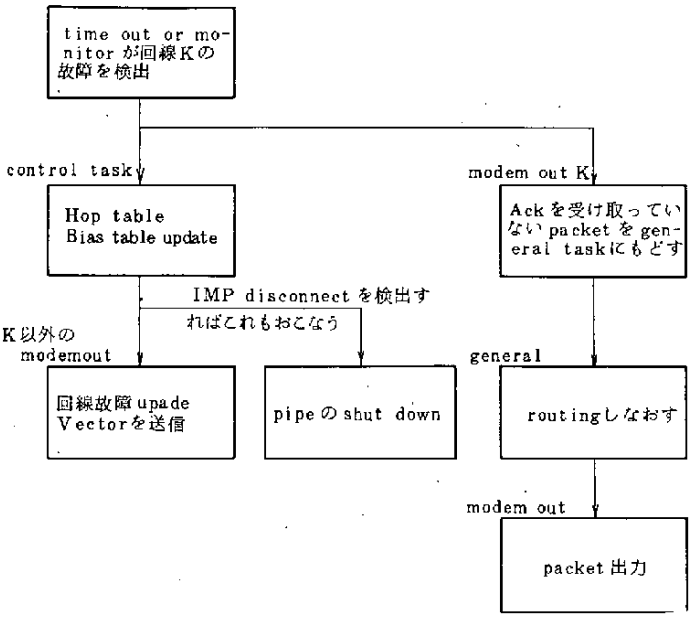


図3－50

この手順においてgeneralタスクが動くときには，必ずcontrolタスクがBias tableを更新しておかなければならない。もし，そうしないとgeneralタスクは同じmodem outタスクに対してパケットをキューonしてしまうからである。

この理由から，modem outあるいはgeneralタスクの少くとも一方がcontrol タスクより優先順序が低くなければならない。

回線効率を100％にもってゆくためには，modem outがエンドオブレンジを検出してから667～833μのあいだに次の命令を発信すればよいのでパイプ管理，control，modem outのタスク順序にした。

B．タスクの分割

今回インプリメントしたサブネットは，タスクに分割する場合に主として機能本位におこなった。この分割が決まったところで，ディバッグ過程においてなるべくトラブルがおこらないように担当者を決めていった。

実際の担当の分割は，modem in，general，modem out を一人が担当し，HOST－IMPパイプ管理，controlを別の人が担当した。この分割により担当者間のトラブルはかなり減り，それなりの効果はあったと考えている。

しかしながら一方これはプログラムの処理効率がおとす結果を招いてしまった。すなわち，general，パイプ管理，controlの3つのタスクは，からみ合いが多いために，1つのタスクとしてインプリメントした方がはるかに効率がよい。また同時に各タスク間で同一のテーブルの参

照をしているために異なる担当名間でこのテーブルの更新タイミングが問題になることがあった。なおこれはtop seat callを利用することにより，このような問題を除去できることがわかった。

C．メッセージ番号の対応

ＩＭＰにおいて，発信地 ― 目的地ＩＭＰを end－to－end とするエラーチェックをおこなっているが，このためにＩＭＰ内でメッセージ番号を使用している。

この番号は，シーケンシングにも利用されているが，発信地－目的地で同一番号のメッセージを授受できるように考えてある。

この方法をとったときに，次のような場合にメッセージ番号の扱いに注意しなければならない。

(1) 図３－51に示すように発信地HOSTがメッセージ転送要求をだしたままdown してしまった場合，すなわち，発信地ＩＭＰがバッファアロケート要求を出したにもかゝわらずメッセージ本体が無いという状態が起こった場合。

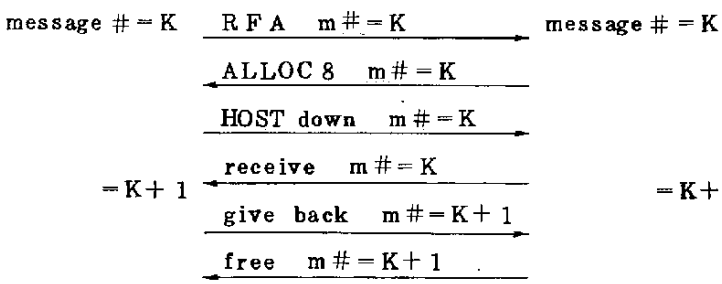


図３－51

ＪＩＰＮＥＴではHOST downというヌルメッセージを目的地に送ることにより，これをさけている。

(2) 回線断の検出の遅れからくる問題点

ＩＭＰが回線の故障を検出すると，他のＩＭＰにその情報を送るが，これを送出する時間遅れがあるために，次のような問題が起こる。

図３－52に於てline １のdown→up，line ４のup→down がほゞ同時に，この順で起こったとしよう。ＩＭＰ＃４は，line ４のdownを先に知るために，ＩＭＰ＃１とはディスコネクト状態になり，パイプ管理が使用するメッセージ番号が０にリセットされてしまう。一方ＩＭＰ＃１は，lineのupを先に知るために，line ４がdownしたとしてもＩＭＰ＃４とはディスコネクト状態にはならない。このためにメッセージ番号は以前のものを保持したままである。

このことから，ＩＭＰ＃１と＃４では，メッセージ番号に違いがあるために，一方から他方へのメッセージは受信側でメッセージ番号エラーで捨てられてしまい，いつまでたってもとどかないことになる。

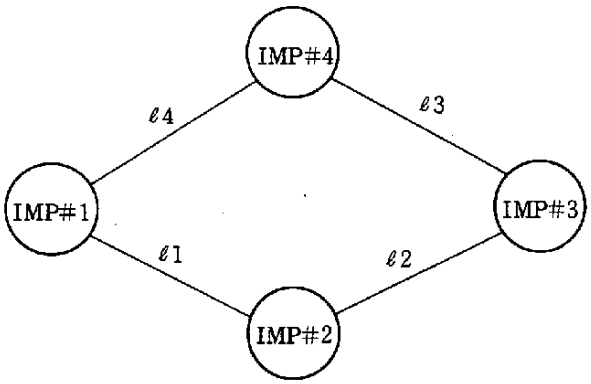


図３－５２

これをのがれるためには，ＩＭＰがディスコネクト→コネクトに移ったときには，メッセージ番号のリセット（ reset ）とそれに対する応答が必要になってくる。即ちこの例ではＩＭＰ＃１と＃４が共にメッセージ番号をリセットすればよい。しかしながら，この現象が起こるのはきわめて稀であるために現在のところＪＩＰＮＥＴにおいては何らの対策もおこなっていない。

3.4.4　ハードウエア

ＩＭＰ本体は，初期においてメモリーパリティーが多発して問題になったが，これは初期段階でほゞ収束した。

ハードウエアで問題になったのは，interface adaptor と 48Kbit／sec の高速制御装置（Ｈ－ＳＬＣ）である。interface adaptor の最大の欠点は，ノイズに弱いということであった。これはノイズ源を調べて，ノイズを発生しないようにすることによって解決をはかった。もう一つは，次の場合の動作である。

である。

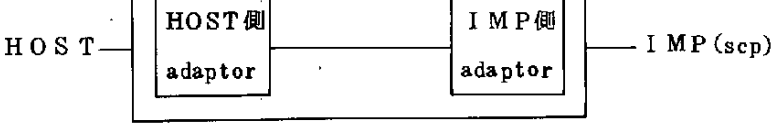


図３－５３

図３－５３に示すHOST側 adaptor の電源が off のまま，ＩＭＰ側 adaptor の電源を onにしてＳＣＰを run する。ＩＭＰ側では 30秒間隔でHOST側の adaptor の電源が on になっているかどう

かを調べている。この時にHOST側 adaptor の電源を onにしても，ＩＭＰ側では，いつまでたってもHOST側 adaptor の電源がハードウエア的な不備により on にならないことがある。

また adaptor の機能の設定として，半二重を採用したのは疑問点がある。Design optionですでに述べたように，HOST－ＩＭＰロックアップ（ lock up ）が起こるばかりでなく，adaptor 上でのコマンド（ command ）の衝突，HOSTのモニター上でのコマンドの衝突など各種の衝突を生む結果となった。これらの問題はすべて全二重を採用すれば，さけられる問題であろう。

次にＨ－ＳＬＣに関していえば，機能の設定が我々が考えていた手法を採用するには不向であったと思われる。このＨ－ＳＬＣは，４本までのチャネルを利用し割り込み情報用，データ受信用にそれぞれ１本ずつ使用し，送信用に１～２本使用している（２本の場合には，データを送信おわると直ちに次のデータを自動的に送ることができる）

く受信側にチャネルを２本用意しておくべきではないかということである。Ｈ－ＳＬＣは，我々のような受信アルゴリズムを使用すると，１パケットの受信につき２回割り込みが起こる。これは，Ｈ－ＳＬＣが受信したデータをコアーに書き込む時に，直前に書き込んだアドレス（ address)を比較し，アドレスが＋１していなければアドレス不連続という割り込みが発生するためである。

Ｈ－ＳＬＣは，我々のような受信アルゴリズムを使用すると，１パケットの受信につき２回割り込みが起こる。これは，Ｈ－ＳＬＣが受信したデータをコアーに書き込む時に，直前に書き込んだアドレス（ address）を比較し，アドレスが＋１していなければアドレス不連続という割り込みが

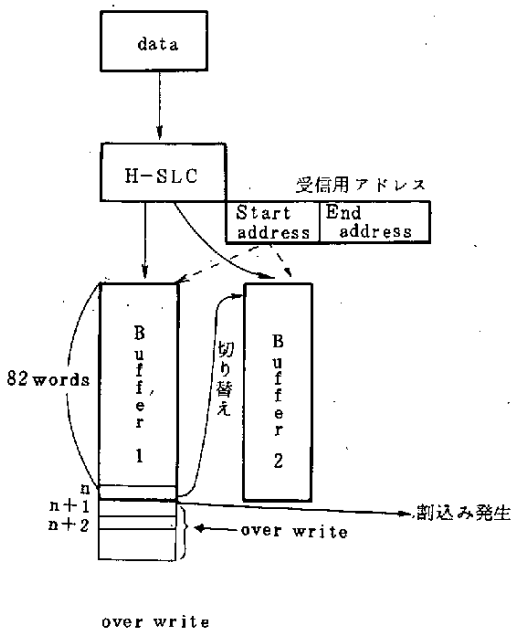


図３－54

H－SLCを通して読み込まれるデータは図3－54に示すように受信開始時にセットされているスタートアドレスより順次一方のバッファ1に書き込まれる。この場合1つのバッファが一杯になると割り込みが発生し，もう1ケのバッファに切り替えられ新しいスタートアドレスがセットされる。しかしこの割り込み処理に手間どると，始めのバッファの続きにどんどんover writeされてゆく。割り込み処理が終りバッファ切りかえがおこなわれるとまず最初に現在のstore addre-ssがしらべられるが，それがn＋1であれば（nはバッファ1の最終アドレス），丁度バッファ1が一杯になったところでバッファ2に切り替った正常状態であり，問題がないが，例えばn＋5位になってしまっていると5 wordsもover writeしたことになりこのover write 分は捨てられ，再度受信を受けることとなる。

バッファ切り替えにかけ得る時間は，図3－55のようにFseqがしらべられてからdataが入ってくるまでの330 μ secが最短のケースである。

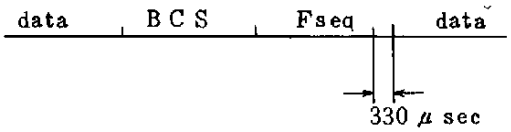


図3－55

この時間内に割り込み処理およびバッファ切り替えを完了しないと，不必要なover writeの発生およびその受信データの棄却をせねばならなくなる。現在はソフトウエアによりこれらの処理をすべて行なっているが，ハードウエアにより自動的にバッファ切り替えが出来る機能が要求される。すなわち受信チャネルを2本にすればより効果がある。もし受信バッファを2個用意してハードウエアが自動的に切り換えてくれるような形になっていれば，バッファ切り換えが間に合わないことは考えられないので，やはり受信チャネルを2本にすべきではないかと考えている。

割り込み情報は，送信，受信いずれの情報も転送されるが，割り込みsource flopは1個であるため，割り込み情報エリアをみて送信か受信かを判別しなければならない。若し，送受信の割り込みが同時に発生すると，1語の割り込み情報に，送受信2個の情報がのってくるために割り込みの発生の都度送受信の区別をせねばならずこの分別も含め割り込み処理ルーチンをぬけるまでかなりの時間がかかることとなる。少なくとも送信，受信の区別はハードウエアで行なえることが望ましい。

3.4.5 まとめ

現在インプリメントを完了したSCPは前述のように効率の点においては若干問題点が残っているがハードウエア障害（modemの電源断，HOST interfaceの電源断およびノイズによる誤動作など）や，HOSTのNCPの誤動作に対してはほゞ完璧な防御機能をもっており，障害の切りわけ，

障害部分の切りはなし動作，その回復手順等が完備しており，ＳＣＰの安全性が充分保障されている。

しかしながら，サブネットが拡張された場合に種々の問題が発生することもわかっている。その例の１つが前述の回線断の検出の遅れからくる問題点である。

今回の経験にもとづいたサブネット設計上の反省点を２，３あげてみる。

○ multi packet message

multi packet message は，バッファを占有するばかりでなく，ＳＣＰのプログラムを非常に複雑にする。packet のシーケンシングさえ完全におこなっていれば，HOST においてリアセンブルするのは比較的簡単であるので，リアセンブルはHOSTにまかせた方がよいのではないか。

○ HOSTのreceive発信

HOST／ＩＭＰ protocol において，RECEIVEをHOSTが発信するように規定した。これは，コネクションの切り換えおよびcease，resume によるコントロールにおいてこの方法をとらざるを得なかったからである。

現在ふり返ってみるに，コネクションの切り換えは高位のプロトコル（high level protocol）において行い，またフローコントロールは別の手法によった方がよいであろうという考えにたっている。それによって，RECEIVE を目的地ＩＭＰにおいて発信できれば，コネクション当りのスループットが上がるばかりでなく，パイプ管理テーブルもかなり小さくすることができる。

○ ルーティングについて

ＪIPNETが採用したShortestQ＋Bias法では（一般にadaptive routing では同様のことがいえる），ルーピング現象，return back path select について充分な解決がなされていない。

ルーピング現象とは図３－56で目的地をｄとしたときに，例えばｊ，ｋ，ｌのＩＭＰのあいだをパケットが繰返し循環する状態をいう。

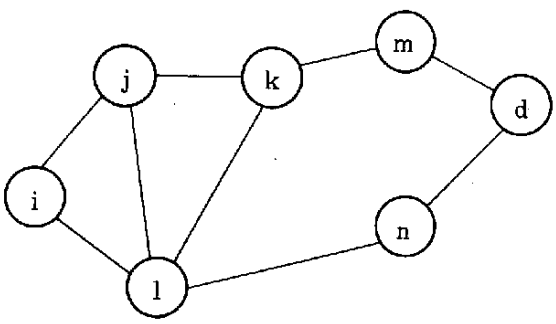


図３－56

この現象を完全な形で回避し得る，効率の良いアルゴリズムは開発されていない。

return back path select とは，例えば図３－57に於てｉ→ｄのパケットに対して，ｉがｊにパケットを流すことである。

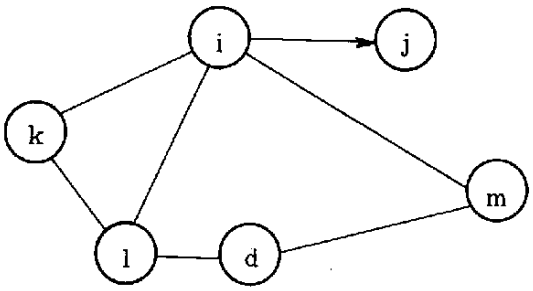


図３－57

このようなパケットは必ず自分のところへ返ってくる。

ＪＩＰＮＥＴの場合には，コネクションマトリックス を持っているのでこれを検出することができる。すなわち，ｉからｊに出力するのが目的地ｄに対して return back path select であるかどうかは，ｉに結合されているすべての line が故障したものとして，ディスコネクトの計算をおこなう。この結果，ｊとｄはディスコネクトの状態になる（ｋとｄはディスコネクトにはならない）のでｉ→ｄのパケットはｊが return back path であるのが分る。

しかしながらルーティングの際，毎度この様な検出を行なうのは適当ではなく，むしろ回線やＩＭＰのup／downのイベント発生時に１回だけ検出処理をおこない，ルーティング時の参照情報をすべきであろうが，それにしてもノード数の多いサブネットでは負担となる。

いづれにしろ，適応ルーティングにともなう無駄なトラフィックの減少等に関しては更に検討の必要がある。

○　コネクションマトリックスとディスコネクションの検出

ＪＩＰＮＥＴにおいては，このマトリックスの大きさは16×16である。したがってサブネットにおけるＩＭＰの数は16までである。これを大きくすると，ディスコネクトの計算をするのに時間がかかる。現在のところほぼ3 m secがかかるので，ＩＭＰ数を２倍にすれば，約2.5倍時間がかかるであろう。

図３－58に示すように，ＩＭＰの結合が幹線と支線のように次のような形になっていれば，ＩＭＰ数をふやしてコネクション　マトリックスは拡大する必要がない。しかし，いずれにせよ幹線ノードが100を越すような規模のネットワークにおけるディスコネクションの検出法は再検討の要があろう。

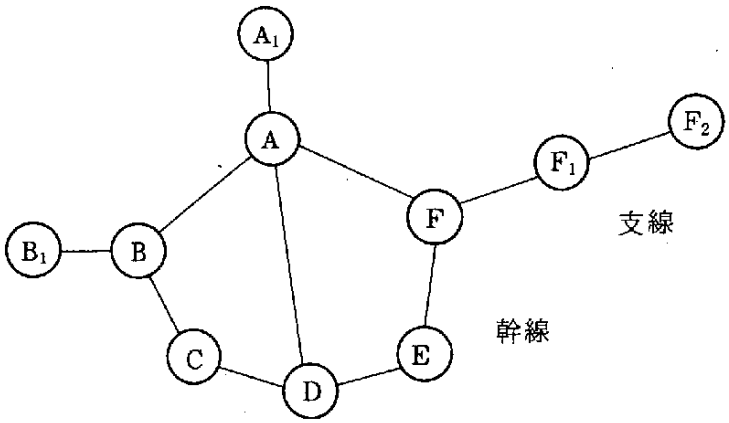


図３－58

なお図３－58のような形のネットワークではＩＭＰのアドレスづけは，グループとグループ内の番号という形になる。

○　ソフトウエア構造

すでに述べたように，割り込みオリエンテッド（ oriented ）な方法により作成する。

○　ＩＭＰの選定

ＩＭＰの選定にあたって，ＪＩＰＮＥＴではＣＰＵのスピードをリアルタイムミックス値で，またチャネルに関しては，１語転送に要するメモリーサイクル（ memory cycle ）のスティールの率によって判断した。

ＳＣＰ作成にあたって実際に問題となったのは，Ｉ／Ｏディバイスの処理コマンドの機能と割り込みに対する応答をいかにはやくするかということであった。したがって接続するＩ／Ｏディバイスのコマンド，割り込み処理のハードウエア化に関して十分な検討を必要とする。

また，core to core の内容の転送のスピードもはやくできることが望まれる。

○　adaptor

すでに述べたようにHOST－ＩＭＰ lock up を防ぐためにも全二重にすべきである。

3.5 Terminal IMP

3.5.1 TIPの機能

A. 概　要

JIPNETのTIPはARPA－net のそれとほゞ同じような機能をはたすものである。すなわち，メッセージ交換をするIMP機能に，端末装置の制御機能と，HOST機能の一部を付加したものである。

TIPがuser 端末に対してサービスするのは，Demand Service，Remote Batch および端末間通信だけである。

TIPにおいては，station がそれぞれのサービスを受ける単位となる。

stationとは，Type-witer，display のように会話形で使用できる端末（control 端末と呼ぶ）が1台と，2台までの入力専用端末，1台までの出力専用端末より構成されている。すなわち，stationはcontrol 端末と，必要に応じて入力専用あるいは出力専用端末を付加したものである。

userは任意にstationを構成し，このstation からDSP，RBP，端末間通信のサービスを受けることができる。

B. TIPの使用法

端末より，サービスを受けようとするuserは，次の段階の手順をふまなければならない。

① TIPと端末の接続

② 端末特性の設定およびstationの定義

③ サービスの要求

④ サービスを受ける

⑤ サービスの終了要求

a. TIPと端末の接続

userは，TIPからサービスを受けたい場合には，まず第1にconsole 端末から会話開始要求としてBreak keyをtype-in しなければならない。

この動作により，TIPは？を端末に対して出力し，次の段階にはいる。

b. 端末の設定

必要があればInsert linefeedコマンドを入力する。

c. stationの定義

必要があれば，assignコマンドによりstationに入力あるいは出力専用端末を付加する。

d. サービスの要求

callコマンドにより，high level protcol のサービスを要求する。Tip側からuser名，pass word の入力要求がくるのでこれに答えなければならない。この応答手順は次のとおりである。

@CALL△DSP△06

#USER△NO＝＊＊＊＊＊＊

#PASS△WORD＝＊＊＊＊＊＊＊＊＊＊＊＊

DSP△SERVICE△START

以下，Host のＴＳＳにしたがう。

㊟ ＿＿＿＿はuserが入力する

＝＝＝＝＝はsystemがオーバ・プリントして黒くぬりつぶしたのち，user が入力する。

e．サービスを受ける

systemがSERVICE△START と出力したのち，この状態にはいる。この状態におけるmessage 入出力は，すべてHOSTの規定にしたがっておこなう。

ＴＩＰのコマンドを入力する場合には，Breakキーを押し，system が？を出力したら，@につづいてTipコマンドを入力することができる。

f．サービスの終了要求

Breakキーを押し，systemが？を出力したら，@CLOSEを入力する。

応答形式は次のとおりである。

?@CLOSE

DSP△SERVICE△END

C．ＴＩＰのコマンド

ASSIGNコマンド

一般形

@ASSIGN〔△INPUT△#〔，#〕〕△OUTPUT△#〕(CR)

#は16進数で端末番号を指定する。

機能

このコマンドによりstationを定義する。このコマンドで指定した入力あるいは出力専用端末はreadyの状態になっていなければならず，このうち１台でも，他人が利用しているとコマンドはrejectされる。

このコマンドは，指定した端末すべてはreleaseされたのちでなければ再び入力できない。

省略形

@A〔△I△#〔，#〕〕〔△O△#〕(CR)

CALLコマンド

一般形

@CALL△Protocol名△ #(CR)

#は16進数でHost番号を指定する。

Protocol名はDSPあるいはRBPである。

機能

このコマンドにより，TIP側の対応するprotocol処理processを呼び出す。

省略形

@C△{D / R}△#(CR)

Clear insert linefeedコマンド

一般形

@CLEAR△INSERT△LINEFEED(CR)

機能

user端末がNLのcodeで復帰改行をおこなうことを指定する。

省略形

@C△I△L(CR)

CLOSEコマンド

一般形

@CLOSE

機能

DSP，RBPのサービスの終了を要求する。

省略形

@C(CR)

GIVE BACKコマンド

一般形

@GIVE△BACK△RIGHT△OF△{INPUT / OUTPUT}(CR)

機能

入力あるいは出力専用端末に対しておこなっていたdata入出力をcontrol端末からするように指示する。

省略形

@G△R△{I / O}(CR)

Insert linefeedコマンド

一般形

@INSERT△LINEFEED (CR)

機能

端末に出力するmessageの最後の文字がlinefeedでないならば，これをそう入することを指示する。

clear insert linefeedが指定されていないとき，初期状態として@I△L(CR)が指定されているものとみなす。

省略形

@I△L(CR)

Reinput startコマンド

一般形

@REINPUT△START(CR)

機能

入力専用端末からの入力dataが受信エラーを起こしたとき，そのrestartの準備が終ったことを指示する。

省略形

@R△S(CR)

Releaseコマンド

一般形

@RELEASE△#〔,#〔,#〕〕(CR)

#は16進数で端末番号を示す。

機能

入力あるいは出力専用端末をstationから切り離す。このコマンドを入力しなくとも，@C(CR)コマンドによりstationから切り離される。

省略形

@R△#〔,#〔,#〕〕(CR)

Set codeコマンド

一般形

@SET△CODE△{EBCDIC / STANDARD}(CR)

機能

DSP，RBPに対するコマンドであって，送信するdataのcodeを指定する。standard

はＮＶＴ code である。

省略形

@S△C△{E/S}(CR)

Send コマンド

一般形

@SEND△{ABORT−OUTPUT / INTERRUPT−PROCESS}(CR)

機能

ＤＳＰ，ＲＢＰに対するコマンドであって，ＮＶＴで決められた特殊コードを server protocol process に送る。

省略形

@S△{A/I}(CR)

Terminal status コマンド

一般形

@TERMINAL△STATUS△#〔,# ，…，#〕(CR)

＃は 16 進数で端末番号を示す。

機能

指定した端末番号の端末が，station で占有されているかどうかを出力してくる。

省略形

@T△S△#〔，#，…，#〕(CR)

Transfer コマンド

一般形

@TRANSFER△RIGHT△OF△{INPUT△#/OUTPUT}(CR)

＃は 16 進数で，端末番号を示す。

機能

入力あるいは出力専用端末に対して，入出力動作の開始を指定する。

入力権が入力専用端末から放棄されるのは，@G△R△I (CR) のコマンドが console端末から入力されたときおよびTip の入力エラーなどにより，control 端末から一時的にコマンドの入力を期待する場合である。

省略形

@T△R△{I△#/O}(CR)

Low level コマンド

一般形

@LOW△ { INIT △#,# / SEND △message / CLOSE } (CR)

機能

端末間交信をおこなうためのコマンドである。

ＩＮＩＴ△#,# は端末間でのコネクションの確立を要求するもので相手側のHOST番号，端末番号をこの順で16進数で指定する。

ＳＥＮＤ△message は相手側端末に対してmessage を送る。

CLOSEは端末間の交信を終わる指示である。

省略形

@L△ { Ｉ△#, # / Ｓ△message / Ｃ } (CR)

3.5.2 ＩＭＰとのインターフェース

すでに述べたように，ＴＩＰは，ＩＭＰに対してHOST機能（ＤＳＰ，ＲＢＰ）と端末制御機能を付加したものである。

ＴＩＰの追加にあたって考慮したのは，ＩＭＰ機能は全く変更せずにおこなうということである。この結果採用したのは，次の方法である。

- ＴＩＰに必要なタスク数だけＴＣＢを追加し，back ground より priority が高いところにおく。
- ＭＬＣ（多重通信制御部）の割り込みを受け付けたら，ＩＭＰの割り込み処理ルーチンは，ＴＩＰの割り込み処理ルーチンに制御を渡たす。
- 40000（８進数）番地以降に，ＴＩＰの task および buffer をおく。
- mini - HOST interface task を作り，ＩＭＰ側からみた場合に，H-Ｉ task と全く同じ方法でHOST側と interface がとれるように設定し，Tip との interface の違いは，この task においてすべて吸収する。
- ＴＩＰのサービス開始，終了は console type - writer よりおこない，これは console からmessage が入力されると，ＴＩＰのサービス開始，終了ルーチンに制御を渡たす。

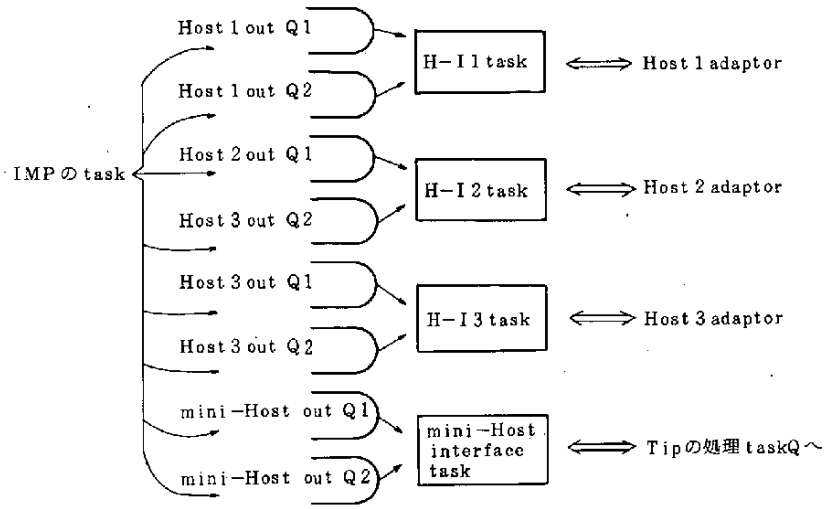


図３－59　ＴＩＰとＩＭＰの結合方法

3.5.3　論理設計

TIPの処理機能としては，次の３つが必要となる。

(1)　端末制御

(2)　TIPコマンドの解析

(3)　リモートHostとの，コネクション管理

これらを処理する為に，必要となるタスク，キューの関係を（図３－60）に示す。

各タスクを以下に説明する。

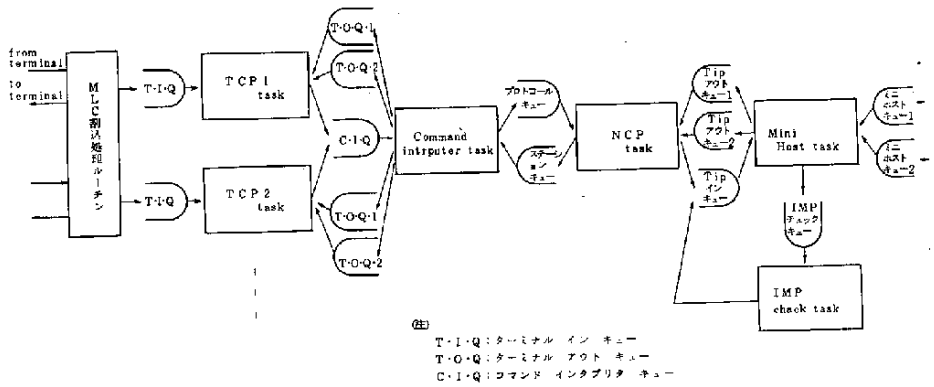


図３－60

A．ＴＣＰ（Terminal Control Program）

端末を制御するタスクで，端末制御手順ごとに作成しなければならなく，新しい手順の端末が付けられると，作成し追加しなければならないタスクである。

機能として

(1) 端末の制御

(2) コードの変換（実端末コード↔ＮＶＴコード）

である。

ＴＣＰが起動される条件として，キュー起動，コミュニケーション起動とがある。

(a) キュー起動

○ ターミナルインキュー

端末ユーザーの入力データキューで，ＭＬＣ割込処理ルーチンによりキューオンされる。

● ターミナルアウトキュー１，２

端末への出力データのキューで各ＴＣＰ毎ターミナルアウトキュー１，２の２つある。

ターミナルアウトキュー２は，Tipコマンドの応答メッセージやBREAKキューに対する応答促進文字（？）のキューで，ターミナルアウトキュー１より先立って処理される。ターミナルアウトキュー１はリモートホストよりの出力メッセージキューである。

(b) コミュニケーション起動

端末へ出力中に，同一端末あてに，キューオンされたメッセージ（出力待メッセージ）は，出力中メッセージの出力終了後，続いて出力しなければならない。出力待ちメッセージのキュー管理用エリヤは，端末テーブルにある。ＭＬＣ割込処理ルーチンは，端末への送信終了割込を検知した時，ＴＣＢ（対応するＴＣＰ）のコミュニケーション起動エリヤのセット，及び，Ｉ／Ｏバッファテーブルのコミュニケーション起動要因ステータスをセットする。コミュニケーション起動要因ステータスは，出力待ちメッセージの出力要求と，入力データ中のエラーに対する再入力要求メッセージ＂RETRANS×××＂の出力要求を区分する為のステータスエリヤである。ＭＬＣ割込処理ルーチンによりコミュニケーション起動されると，ＴＣＰは，コミュニケーション起動要因ステータスにより，＂RETRANS×××＂メッセージを出力するか，出力待メッセージキューのメッセージを出力する。端末がTipに対して，入力権のない時の入力データは棄却され，端末に＂ＩＬＬＥＧＡＬ ＭＥＳＳＡＧＥ＂が出力される。端末がTipに対し入力権を得るのは，BREAKキーの打鍵後及び，リモートホストよりのＧＡコマンドを受け取った時である。

B．コマンド・インタプリタ・タスク

BREAKキー打鍵後，TIPコマンドが入力される。このコマンドを解析するタスクである。ＴＣＰは個々の端末をサービスするか，このタスクは，ステーションを単位として管理する。

機能としては，

(1) 端末，ステーション間の変換

(2) TIPコマンドの解析

がある。

ＮＣＰタスクは，このタスクで作られた，ステーションをＮＶＴと仮想して，処理する。

本タスクが起動される条件は，キュー起動のみである。

⊘ コマンド・インタプリンタ・キュー

ＴＣＰよりキューオンされたメッセージで，＂BREAK＂データ TIPコマンド，プロセスデータが含まれる。これらの入力データと処理後の対象キューの関係を図３－61に示す。

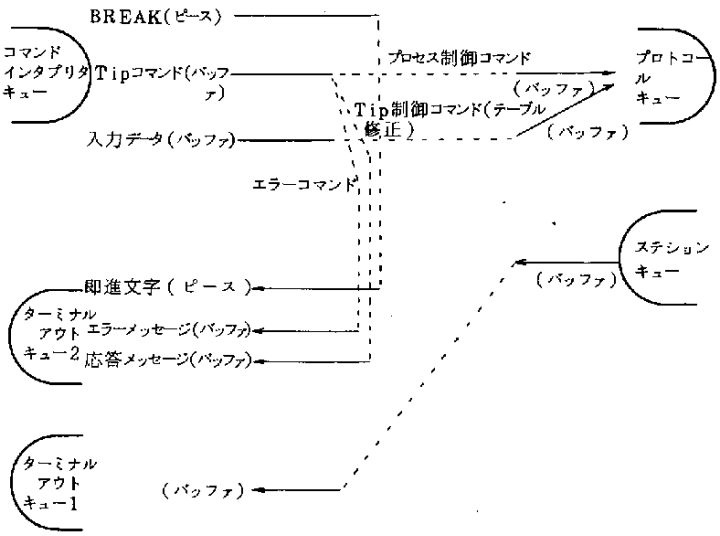


図３－61

キューには，ピース，バッファが含まれている。両者を識別する為ピースの第４ワード目は全ビットオンにする。TIP コマンドの内容により処理後，キューオンされる対象が変わる。端末の機能を変える為のTIP制御コマンドは，TIP内のテーブルを修正し，応答メッセージをターミナルアウトキュー２にキューオンする。コネクションの確立，切断要求コマンド，プロセス制御コマンドはテーブルに登録後，プロトコールキューにキューオンする。誤りのコマンドに対してはエラーメッセージをターミナルアウトキュー２にキューオンする。

⊘ ステーションキュー

リモートHOSTより端末への出力メッセージのキューで，出力メッセージを対応する端末のターミナルアウトキュー１にキューオンされる。又，TIPコマンドのリモートHOSTとの応答結果もキューオンされ，応答結果を，応答コマンドメッセージに変換して，ターミナルアウトキュー２にキューオンされる。

C．ＮＣＰタスク

このタスクは，リモートHOSTとのメッセージのやりとりを行なうタスクで，高位のプロトコル，低位のプロトコルの両者をサービスするタスクである。

機能としては，

(1) コネクションの管理

(2) リモートHOSTとのデータフロー管理

(3) ＮＣＰコマンド，高位プロトコルコマンドの解析

を含むタスクである。

このタスクが起動される条件はキュー起動のみである。

キューの関係を図３－62に示す。

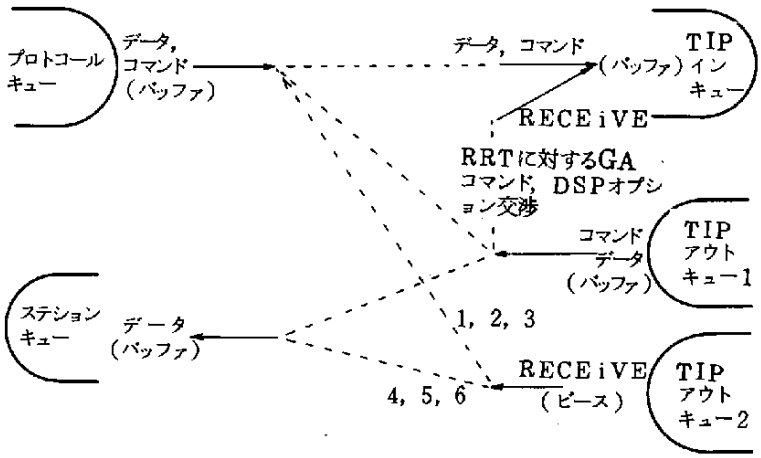


図３－62

○　プロトコルキュー

コマンドインタプリタタスクによりキューオンされるキューである。キューの内容データには，処理入力データ，プロセスコマンド（例えばＴＳＳコマンド），プロトコルコマンドが含まれ編集されるデータもあるが，（プロトコルコマンド）全てのデータともTIPインキューにキューオンされる。

リモートHOSTへメッセージ送信中に送信メッセージがキューオンされた時は，送信中メッセージの受取承認メッセージ（ＲＥＣＥＩＶＥ）を受けとる迄送信待ちになる。又TIPが，リモートホストにデータを送る送信権のない時，TIPは先ずTIPに送信権を得る為のＲＲＴコマンドを送る。次にリモートHOSTより承認コマンド（ＧＡ）を受けとる。この間，ＲＲＴを要求したメッセージは送信待ちになる。又ＲＥＣＥＩＶＥを受け取っても，リモートHOSTのＮＣＰで棄却されたメッセージでは，再送しなければならない。

- TIPアウトキュー1

リモートHOSTよりのバッファ形式のデータに対するキューで，端末に出力されるデータ及び，リモートHOST間の交渉コマンド（ローレベルプロトコル コマンド，ハイレベルプロトコル コマンド）が含まれる。この時の応答結果は，ステションキューにキューオンされるもの，（TIPコマンドの入力時）端末に知らせずに，そのコマンドの応答をTIPインキューにキューオンするものとする。（ＤＳＰコネクション時のオプション交渉）

- TIPアウトキュー2

ピース形式のメッセージがキューオンされる。

TIPは1パケット・メッセージのみがサービス対象である為RECEIVE のみキューオンされる。

（注：現在1パケット・メッセージのみサービスしている）

D．ＩＭＰチェックタスク

TIPに接続されているＩＭＰとの初期設定，接続中止を行なうタスクである。

このタスクはTIPイニシャライザールーチン，TIPターミネイトルーチンと関連する。これらの関係を図3－63に示す。

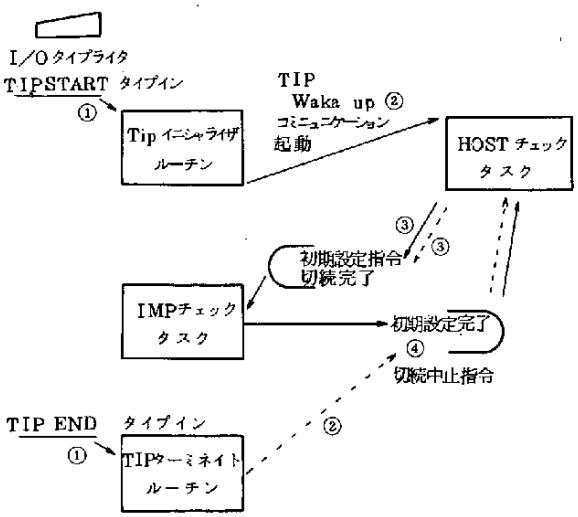


図3－63

E．その他ルーチン

TIPが保有するルーチンを次に示す。

(1) ＭＬＣ割込処理ルーチン

(2) 端末出力ルーチン

(3) TIPイニシャライザルーチン

(4) TIPターミネイトルーチン

(5) TIPバッファゲットルーチン

(6) TIPバッファフリールーチン

上記以外はＩＭＰのモニタを使用する。

各ルーチンについて次に説明する。

(1) ＭＬＣ割込処理ルーチン

ＩＭＰの割込処理ルーチンより，ＭＬＣの割込に対して，コントロールが渡され，ＭＬＣの割込分析を行なう。端末よりの入力データの終了割込に対しては，入力データをターミナルインキューにキューオンする。

入力エラー（受信オーバーラン等）の時に，再送要求メッセージ出力の為，又，送信終了割込の時端末送信待メッセージの送信の為にＴＣＰにコミュニケーション起動をかける。

(2) 端末出力ルーチン

端末への出力メッセージを出力するルーチンである。

(3) TIPイニシャライザルーチン

Ｉ／Ｏタイプライタに＂TIP START＂をタイプインすることにより起割されるルーチンで，TIPバッファののフリーチェインの作成，テーブル類の初期化，ＭＬＣ回線のオープン，Tipの初期設定を行なうルーチンである。

(4) TIPターミネイトルーチン

Ｉ／Ｏタイプライタに"TIP END"をタイプインすることにより起動されるルーチンで，Tipの切断を行なうルーチンである。

(5) TIPバッファゲットルーチン

TIPバッファをフリーチェイン上より，一個得るルーチンで，フリーのTIPバッファがなければ，要求タスクのＴＣＢをバッファリクエストキューに登録し，要求タスクのＴＣＢ上のタスクスティタスをコアウェイトにする。

(6) TIPバッファフリールーチン

TIPバッファを解放し，フリーチェイン上に登録する。もしバッファリクエストキュー上に，バッファ要求タスクがあれば，そのタスクにそのフリーバッファを与える。

4. HOSTのNCP

4. ＨＯＳＴのＮＣＰ

4.1 ＮＣＰ概論

この節では，ＮＣＰの一般論について述べる。議論を具体化する場合はARPANETなど既存のものを実例としてとり上げる。

ＮＣＰは各HOSTに存在するネットワークのコントロール・プログラムである。ＮＣＰは，自系ユーザ・プロセスが他系HOSTのユーザ・プロセスと自由に簡単に交信できるようにサービスするプログラムであり，一般に各HOSTのＯＳの一部，あるいはそれに準ずる形で存在することになる。

異機種結合によるコンピュータ・ネットワークでは，各HOSTに存在するＯＳはそれぞれまったく独自の思想によって作られており，各ＯＳとのインタフェースを十分考慮したうえでＮＣＰをどのように実現すべきかを検討せねばならない。

ＮＣＰの機能は，前述のHOST-HOSTプロトコル，HOST-IMP プロトコルの処理とユーザ・プロセスへのサービスである。

4.1.1 ＮＣＰ実現上のHOST ＯＳの必須条件

ＮＣＰは前述のHOST－HOSTプロトコルに従いユーザ・プロセスにネットワーク機能をサービスするため，HOSTの既存のＯＳには最低限以下のような機能が備わっていることが望ましい。

① 多重処理機能をそなえていること。

② 他マシン（ＩＭＰ）とのインタフェース・アダプタが，チャネルの１つに連結された周辺装置の一種として扱い得ること。

③ インタフェース・アダプタを介してビット・ストリングのトランスペアレント(transparent)な送受が可能であること。

④ ＮＣＰとユーザ・プロセスとの間で情報の受け渡しが可能であること。

⑤ ＮＣＰがプロセス（ジョブ）の起動，監視および強制終了をおこなえること。

⑥ プロセスに必要なファイルなどが動的に割り付けられ，また消去が可能であること。

⑦ 言語プロセッサ，ユーティリティ・プログラムなどの既存の処理プログラムがＮＣＰから利用できること。

⑧ アカウンティングなどのためにリソース使用情報が取り出せること。

⑨ 情報の機密保持が万全であること。

まず①はネットワークに結合されるホストの最低条件であり，ネットワークを通した複数ジョブ，あるいはローカル・ジョブとネットワーク・ジョブの多重処理は必須のものであり，ＯＳにそのコントロール機能がそなわっていなければならない。

②はHOSTとＩＭＰがチャネルによる直結の場合は絶対必要条件であり，これが満されない場合はＮＣＰを作成できない。しかし，両者が回線結合の場合は，すでに既存ＯＳにコミュニケーション機能がサポートされているはずである。HOSTとＩＭＰの結合は，両者間が短距離の場合はチャネル結合が可能であるが，長距離の場合は通信回線接続となり，今後は回線の高速化，接続しやすさ，標準化などの点で回線接続が標準となるであろう。

③は異種コンピュータ間通信の場合，各コンピュータ毎にデータ転送の最低単位（ビット，バイト，語など）や文字のビット配列が異なるため，２進データを送る際にbit-orientedで，メッセージの長さは可変長ビットをとることができ，いかなるビット配列（符号）に対してトランスペアレンシーが保障されていなければならない。もし，この条件が実現できないと，ＮＣＰ又はＳＣＰが肥大化し，オーバヘッドが増大する。

④はＮＣＰが自系HOST内に存在する複数の独立のユーザ・プロセスにコンカレントにサービスするために絶対必要条件であり，これが満されない場合はＮＣＰを作成できない。

⑤はＮＣＰが他系ＮＣＰ（ユーザ・プロセス）の依頼により自系プロセスの起動・監視ができれば，ネットワーク内の複数ホストにわたるCooperated Jobの処理が容易になる。

⑥は各HOSTに散存するファイルなどが自由に使えるようにし，特に高位プロトコル（ＦＴＰなど）をサービスするために必要である。

⑦は各HOSTの既存のソフトウェア・リソースの共用を可能とする機能であり，これはコンピュータ・ネットワークの基本機能の一つである。

⑧は各HOSTのリソースの利用状況を管理するために必要であり，この情報はＡＪＤ（Automatic Job Dispatching)に於て最適HOSTの割当てに利用し得る。これによりコンピュータ・ネットワークをバーチャル・ネットワークへと発展させる可能性が生ずる。また商用コンピュータ・ネットワークに於ては，各HOST利用に対する課金のためにぜひ必要である。

⑨に関して，ネットワークのオーバオールな機密保護対策は，HOST，サブネット，ハードウェア，ソフトウェアなどの総合的なロード分担の問題となるが，最もベーシックな機能として各HOSTに先づこの機能がサポートされていることが必要である。

4.1.2　ＨＯＳＴ間のコミュニケーション機能

HOST－HOSTプロトコルの基本的な役割りは，ネットワークにまたがるプロセス－プロセス通信を円滑に行なう手順を提供することである。

A　プロセス間のコネクションの確立と閉鎖

図４－１に示すように，２つのHOST A，Bのそれぞれのプロセスａ，ｂが各々のＮＣＰを通して相互に交信するには，まず両者の間でコネクションを確立せねばならない。コネクションの確立とは，一方のプロセスの出力が他方のプロセスの入力となるロジカルなパスを作るこ

とである。そして，そのロジカルなパスを閉じるのがコネクションの閉鎖である。コネクションが一方向に定義されている場合，両方向に通信するには２つのコネクションが必要である。

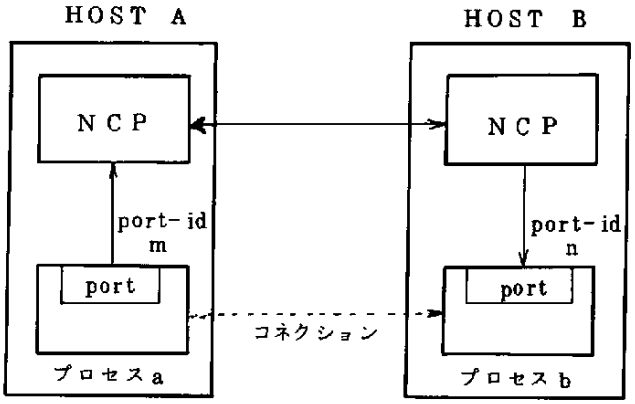


図４－１　プロセス間コネクション

プロセス間のコネクションを確立する場合は，まず他のHOST内の相手プロセスを指示する手段が必要である。一般にプロセスの識別名は各HOST特有の規則が夫々存在し，それらを一元化してネットワークに対してグローバルな名前を付ける必要がある。これを port-id と呼ぶ。１つのコネクションは２つのグローバルな port-idのペア（図ではｍ、ｎ）に対して確立される。

図４－２は port-id のフォーマットであり，HOSTを識別するための番号，ユーザを識別するための番号などからなる。ユーザ番号には，自分が所属するホームHOSTの番号が含まれている。なお補足番号は，１つのプロセスが複数のコネクションを同時に張る場合に必要となる。

HOST番号	ユーザ番号	補足番号

図４－２　port-id のフォーマット

コネクションの識別を簡単化するため，図４－３に示すコネクション名を付ける場合がある。コネクション名を付ける場合，受信側NCPがネットワークにグローバルな名前を割り当てる。

HOST番号	コネクション番号

図4－3　コネクション名のフォーマット

ユーザ・プロセスからコネクション確立要求を受けると，ＮＣＰは他系ＮＣＰにコネクション確立要求コマンド（Request For Connection）を送り，他系ＮＣＰはそのコマンドを受け取り，そのHOST内のプロセスがコネクション確立要求を出すと，他系ＮＣＰへコネクション確立要求コマンドを返送する。このコマンドのパラメータである両者のport－id が一致する一対のコマンドが交換されると，コネクションが確立される。

またユーザ・プロセスからコネクション閉鎖要求を受けると，ＮＣＰは他系ＮＣＰへコネクション閉鎖要求コマンド（Close）を送出し，それに対し他系ＮＣＰからも一致するコネクション閉鎖要求コマンドを受け取った時，コネクションが閉鎖される。

B　プロセス間のデータの授受

コネクションが確立されると，自系プロセスの送信要求（Write）によりコネクション上で他系ＮＣＰへデータを送る。データを受信したＮＣＰはそれをバッファ内に貯えておき，自系のユーザ・プロセスが受信要求（Read）を出した時，ユーザ領域にデータを転送する。

送／受信オペレーションが完了するまでユーザ・プロセスを停止させるやり方（これをSynchronous operationと呼ぶ）と，完了する前に他のオペレーションを実行できるよう制御を戻し，その完了をＥＣＢ（Event Control Block)に通知するやり方（これをAsynchronous operationと呼ぶ）の２通りの場合がある。

C　プロセス間の同期

ネットワークを通して１つの仕事を複数のHOSTで実行する場合を考慮すると，当然プロセス間で同期をとり合う機能が必要となる。

プロセス間の一般のデータの授受はコネクションを通しておこなうが，同期用などの小量データの送受信はプロセス間でいちいちコネクションを確立せずに行なう方法もある。

以上のようなHOST間のコミュニケーション機能は，ネットワークの利用形態やユーザ・プロセスのプロフィールによる影響が大きく，たとえば単発の短いメッセージ転送を主体とするネットワーク・システムでは，あらかじめコネクションを確立するという手続きをはぶき，送受信の具体的要求が出た時点で，そのつどコネクションに相当する手続きをとるという方法も考えられる。しかし，大量データの連続転送や頻度の高い会話型転送では，かえってオーバヘッドが増える可能性がある。この方式はデータグラム（Datagram）と呼ばれ，一方コネクションを確立してメッセージを転送する方式はバーチャル・コール（Virtual call）と呼ばれ

る。

4.1.3 ＨＯＳＴ間のフロー・コントロール

サブネット間のフロー・コントロールとは別に，HOST間にもフロー・コントロールが必要である。各HOSTは一般に複数HOSTと同時に交信しており，１つのコネクションにおいて送信側HOSTが大量のデータを送り込んでも，受信側は常にそれにマッチした速度で処理し得るとは限らない。そこで受信側HOSTが送信側HOSTからのフローを抑制できるなんらかの機構が必要となる。

HOST間のフロー・コントロールには，ＷＡＢＴ（Wait before transmit）方式とバッファ予約方式とがある。

ＷＡＢＴ方式は次の通りである。HOST間で１つのコネクションが確立すると，最低1個の基底バッファが与えられるが，それ以上は必要に応じてバッファ・プール（buffer pool）から取り出して供給され，使用後はプールに返却される。プールに残ったバッファ数がある限度を切った場合，多量にバッファを占有しているコネクションに対しては送信側HOSTにデータ送出の一時停止（ＷＡＢＴ信号）を要求する。このコネクションに対する受信側HOSTの処理が進行するか，他のコネクションとの関連でバッファ事情が好転すれば送信の再開を許す。

一方，バッファ予約方式はARPANETで採用されており，次の通りである。受信側HOSTは各々のコネクションに対しバッファ・スペース（buffer space）をビット単位で割り当て，どれだけのスペースが利用できるかをＡＬＬ（Allocate）コマンドで送信側HOSTへ知らせる。送信側HOSTはその累計送信ビットだけデータを送信し，そこでいったん送信を止めて，次のＡＬＬコマンドを待つ。

いずれの方式を採用するにせよ，HOST間のフロー・コントロールは基本的にはＮＣＰ内のバッファ利用状況の管理の問題となる。各ＮＣＰはそれぞれ適当な個数のバッファ・プールを持つ。一般には一度に確立し得るコネクションの数を基本とし，１コネクションにつき２，３個とか，ファイル処理，会話型処理などの性質によるメッセージ長の考慮などを含め，そのHOSTでとり得るメモリ・スペースとの兼合いでバッファ個数あるいはバッファ容量を定める。いずれにしろ，コネクションが少ない時は多くのバッファが利用でき，混雑時にも少なくとも１個の基底バッファは確保できるような配慮が望ましい。

4.1.4 コントロール・コマンド

ＮＣＰはその機能を果たすために，他のHOSTのＮＣＰと通信を行なう必要がある。この目的のために，各HOST間で前もって１つのコネクションを割当てておく。そのコネクションをコントロール・コネクションと呼び，このコネクションを通してコントロール・メッセージを伝送する。コ

ントロール・メッセージは1つまたはそれ以上のコントロール・コマンドが連らなってできており，相手ＮＣＰによって分離・解釈される。両ＮＣＰは互いにコントロール・コマンドを交換してプロセス間通信の制御をおこなう。

コントロール・コマンドは即ちHOST－HOSTプロトコルのコマンドであり，これに関しては，1.4で述べたのでここでは省略する。

4.1.5　ＮＣＰのサービス・コマンド

ユーザ・プロセスはＮＣＰに対してネットワーク・アクセスを要求するわけであるが，その手段としてHOST－HOSTプロトコルのコマンドをそのままの形で利用させることができない。これはＮＣＰが多重コネクションの管理，フローのコントロールなどを含んでいるからである。ユーザ・プロセスからの依頼は，マクロなサービス・コマンドの形で行なわれる。

基本的なサービス・コマンドには以下のものがある。

① Open　：コネクションの確立を要求する。
(Connect)

② Close　：コネクションの閉鎖を要求する。
(Disconnect)

③ Read　：コネクションを通じてメッセージを受信する。
(Receive)

④ Write　：コネクションを通じてメッセージを送信する。
(Send)

⑤ Post：ステータスやイベント情報などの短いメッセージをコネクションの確立なしに送信する。

⑥ Wait：サービス・コマンドの終了あるいはイベントの通知を待つ。

⑦ Listen：コネクション要求持ち（4.1.6で後述）。

⑧ Set ecb-id：Post受信用のecb-idを宣言する。

⑨ Set interrupt entry：相手側より割込み信号が来たときに割込む入口を宣言する。

サービス・コマンドは，周辺入出力装置のアクセス・メソードに近い形でプロセス間コミュニケーション・アクセス・メソードの提供ができることが望ましい。また，後述の高位プロトコルを容易に実現できるようにこの他のサービス・コマンドが追加されるであろう。

HOST－HOSTプロトコルのコマンドとユーザが利用するサービス・コマンドの関係を図4－4にそって述べる。

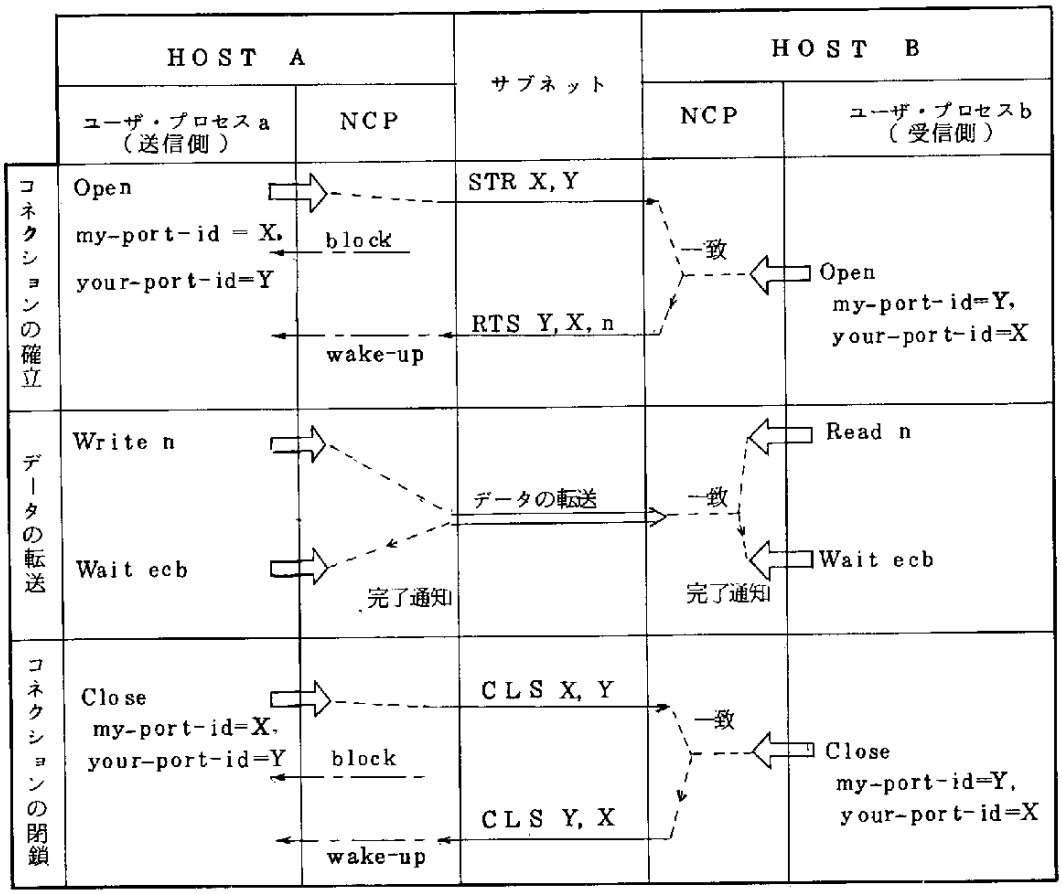


図４－４　プロセス間コミュニケーションの過程

HOST　Aの送信側プロセスaがコネクション確立を要求するOpenサービス・コマンドを同HOSTのNCPに発すると，NCPはSTR（Sender-To-Receiver）を受信側HOST　BのNCPに送る。送信側NCPは受信側NCPからの一致するRTS（Receiver-To-Sender）コマンドが来るまでユーザ・プロセスaをblockしておく。

一方，HOST　Bの受信側プロセスbがOpenサービス・コマンドによりコネクション確立要求を発すると，HOST　BのNCPはRTSコマンドを送信側HOST AのNCPに送る。これら２つのコマンドSTRとRTSはRequest-For-Connectionと呼ばれる。両コマンドは夫々my port-idとyour-port-idとをパラメータとして持ち，両者のid同士（ここではXとY）が一致するときSTRとRTSは一致するといい，一致した一対のコマンドが交換されるとコネクションが確立する。コネクションの確立が完了すると，両NCPはユーザ・プロセスa，bを夫々wake-upする。

コネクションが確立されると，プロセスaはWriteサービス・コマンドを発し，データの送信を要求し，送信側NCPはそのデータをNCPの送信用バッファにコピーし，送信待ち行列に登録し，

プロセスaに制御を戻す。Write nのnはコネクション番号である。具体的なデータの送信はユーザ・プロセスaとは非同期に動作し，この間ユーザ・プロセスは他のオペレーションを実行することも可能であり，ユーザ・プロセスaは適当な時点でWait サービス・コマンドで送信の完了を持つ。これに対し，受信側プロセスbはRead コマンドを発し，Write との両者のnが一致すれば，データ転送が開始される。

送信側プロセスaがClose サービス・コマンドを発し，コネクションの閉鎖を要求すると，NCPはCLSコマンドを送信側NCPに送る。送信側NCPよりport-idが一致するCLSコマンドを受け取るまで送信側プロセスaはblockされる。コネクションの閉鎖を完了するためには，どちら側のNCPもプロセスからのClose サービス・コマンドによりCLSコマンドを送り，また受けとらねばならない。

Open サービス・コマンドとCloseサービス・コマンドが完了するまでユーザ・プロセスをblockしておくと，図4－5に示すようなデッドロック（deadlock）現象が起きる可能性がある。

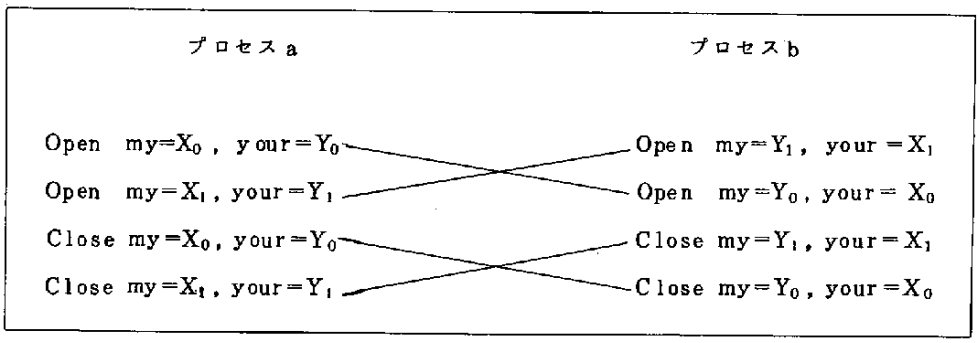


図4－5　コネクション確立・閉鎖のデッドロック

デッドロックが発生したことを検出することは，不可能ではないが，むしろデッドロックを引きおこさないようなシステム設計が必要である。いかなる場合にもデッドロックを避け得るという定石はないが，以下に述べるような手段で，デッドロックの防御を行なうことが望ましい。

① 一括要求

処理途中で必要になるリソースをできるだけまとめて一度に要求する。例えば1つのサービス・コマンドで両方向のコネクションの確立，あるいは閉鎖をおこなう。

② 定順要求

消極的解決方法であるが，リソースに対する要求をどのプロセスも同じ一定順序で発するものと約束する方式である。例えば両プロセス対間でコネクション確立・閉鎖のためのサービスコマンドを発する順序をとりきめておく。

③ 時間監視

コネクション確立要求のサービス・コマンドのパラメータとしてタイムアウト時間を指定させ（省略時は標準時間を指定したとみなす），タイムアウト後はそのコネクション確立要求をとり下げる。

4.1.6 高位プロトコルの処理

高位プロトコルはＮＣＰの上位のソフトウェアで，NCPの機能を使って処理をおこなう。高位プロトコルとしては，デマンド・サービス・プロトコル（ＤＳＰ），リモート・バッチ・プロトコル（ＲＢＰ），ファイル転送プロトコル（ＦＴＰ）などはもっとも基本的であろう。ネットワーク・サービスの拡大により，これらの高位プロトコルは数も種類も増加してゆくものであり，その処理形態も拡張性を十分考慮に入れる必要がある。たとえば，これらの処理を高位プロトコル処理プロセスというユーティリティ・プロセスの形でおこなうのは一つの方法である。

HOST間コミュニケーションで述べた各HOSTのプロセスには，ユーザ各自のユーザ・プロセスとシステムがサポートするユーティリティ・プロセスとがある。ユーザ・プロセスとユーティリティ・プロセスとではコネクションを確立する場合の手順がやや異なる。

ユーティリティ・プロセスは夫々リザーブされた個有のport－idを持つ。そして，一般に，ユーティリティ・プロセスは，不特定多数の他のプロセスからコネクションの要求を受けることが想定されるが，自分のport-idは勿論わかるが，相手のport－idがわからない。そこで自分のport-idを宣言し，他からのコネクション要求待ちの状態に入るＮＣＰサービス・コマンドが用意される。これはListenサービス・コマンドと呼ばれる。他系からのコネクション確立要求と，自分のport－idが一致すると要求元のport－idが通知され要求待ちが解除され，改めて通知されたport－idをパラメータとしてOpenサービス・コマンドを発する。これによってUser-Server間のコネクションが確立される。

A　ＤＳＰの実現方法

ＤＳＰ（Demand Service Protocol）は，他のHOSTのＴＳＳをネットワークを通して使用するものである。

図４－６にＤＳＰの処理の概念図を示す。ユーザHOSTのＴＳＳ端末からサーバHOSTのＴＳＳを利用するには，ユーザHOSTのNCPのもとにユーザDSPというユーティリティ・プロセスがあり，それに対してサーバHOSTのNCPのもとにもサーバＤＳＰと呼ばれる仮想端末プロセスが存在する。ユーザＨＯＳＴ，サーバＨＯＳＴとも既存のＯＳ機能としてＴＳＳモニタがあり，たとえばサーバHOSTにおける通常のＴＳＳ使用はサーバHOSTの実端末（点線）からサーバHOSTのＴＳＳモニタを経由してユーザ・ジョブと会話をおこなうわけである。しかしネットワークを通してユーザHOSTの端末からサーバHOSTのこのユーザ・プロセスと会話をするばあいは，点線で示した通常の端末へのつながりが，サーバHOSTのＴＳＳ

モニタ中でサーバＤＳＰ（仮想端末プロセス）へのつながりへと切り換えられ，両ＨＯＳＴのＮＣＰを通して会話がおこなわれることになる。

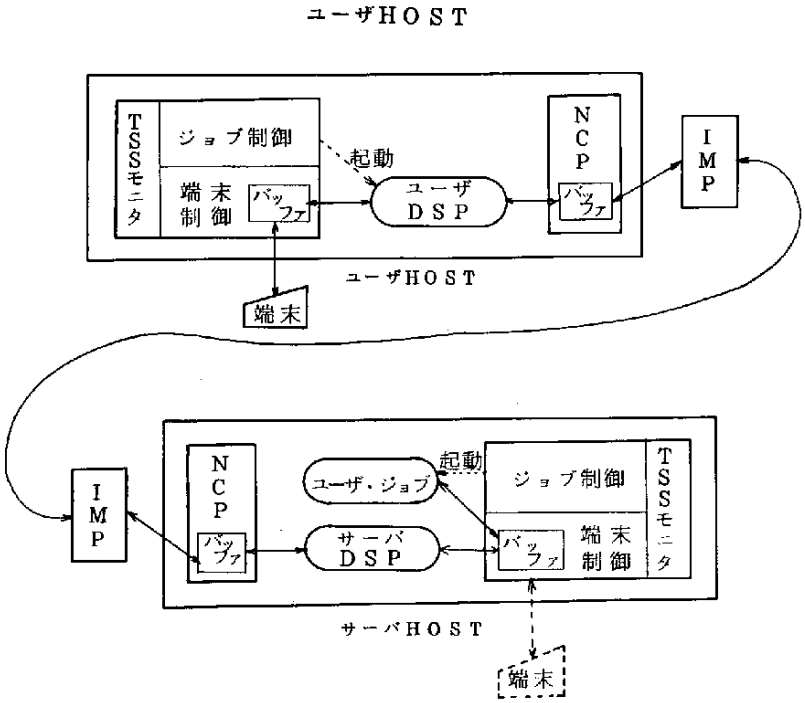


図４－６　ＤＳＰの実現方法

通常のシステムではこのような機能をＯＳの外に持たせる（例えば，ユーザ・プログラムとして）ことは困難であるが，１つの方法は，ＯＳの中の端末制御を改造してサーバＤＳＰを普通の端末とみなすことができるようにすることである。最近のＯＳでは，あらゆる種類の端末装置や回線をサポートできるように，端末の物理的特性をサポートする部分を完全な独立モジュールとしており，その部分に新規の端末が追加されたような形で仮想端末用端末制御プログラムを組み込むとよい。

ＩＢＭで最近発表したシステム・ネットワーク体系（SNA, System Network Architecture）では，アプリケーション・プログラムと端末装置のオペレーションを同一概念でとらへ，最終情報授受者（End User）と呼んでいる。

このサーバ，ユーザ両ＤＳＰをユーザ・プログラムとして実現できると作成も簡単になり，拡張性や融通性に富む。また，ネットワークの標準コードとして，ＪＩＳコードを採用した場合，受信側端末にカナ文字印字機構がない時はやや面倒であるが，カナ文字対ローマ字変換の機能をＤＳＰの基本的機能の一部として含むことが望ましい。そのほか端末の行送りなどの端

末制御をシミュレートしなければならない。

B　ＲＢＰの実現方法

ＲＢＰ（Remote Batch Protocol）は，端末からソース・カード・デック（ローカル・ファイル）をリモートHOSTへ送って処理し，結果を手元に返送させる機能をもったものである。

図４－７にＲＢＰの処理の概念図を示す。

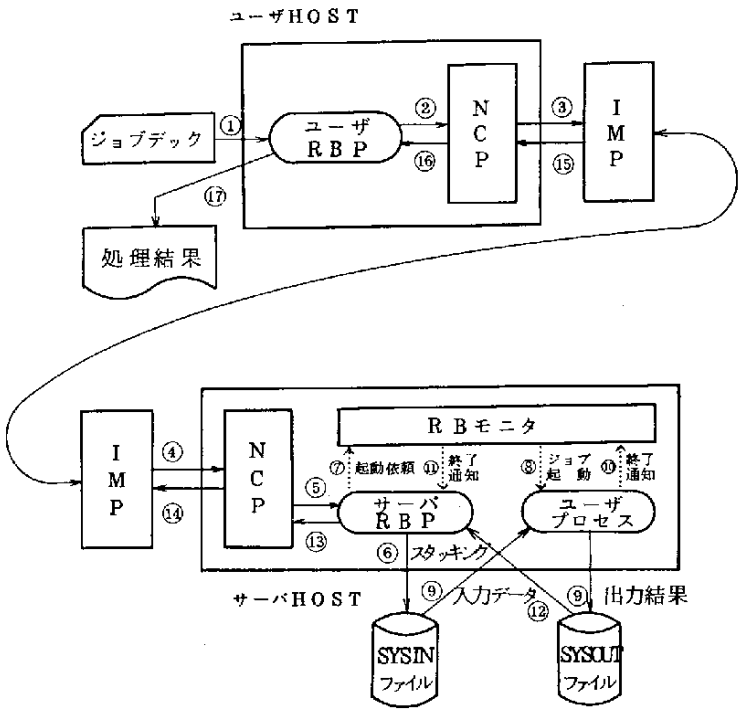


図４－７　ＲＢＰの実現方法

ＲＢＰは，ＤＳＰと同様にユーザHOSTにユーザＲＢＰプロセス，サーバHOSTにサーバＲＢＰプロセスが存在する形で実現される。図中の番号はメッセージおよび制御の順序を示す。サーバＲＢＰは既存のＲＢモニタの入力制御をシミュレートする。

サーバＲＢＰはサーバＤＳＰと同様にユーザ・プロセスとして実現できることが望ましい。ＲＢＰ用のSYSIN ファイル・エリアを正規（既存）のSYSIN ファイル・エリアとは別の所に確保できると，既存の入力制御との競合がさけられサーバが簡単になる。

異種HOST間でのリモート・バッチの場合，文字コードに関しては例えばＪＩＳコードや

EBCDIC コードを採用すればあまり問題はないが，ラインプリンタの改行制御コード（Slow code）およびその動作は各機種まちまちであり，この方面の標準化も望まれる。具体的には各HOSTごとに異なる改行制御コードを一度共通の標準コードに変換し，ユーザRBPがローカル・コードに変換する。

RBPで授受されるデータは大量であり，ブロッキングや（空白が非常に多いので）空白の圧縮（Compactation）を行なうと，ユーザーサーバ間の実効データ転送率が向上する。

C．FTPの実現方法

FTP（File Transfer Protocol）は，HOST間のファイルの転送機能であり，付随する機能としてリモートHOST上のファイルのディレクトリ・リストの出力，ファイル名の変更，ファイルの抹消などの手続きを定めるものである。

FTPは，転送するファイルのロジカルな内容にどの程度立ち入るかによって如何様な規模のものにもなり得る。たとえば，シーケンシャル・ファイルのみでなくランダム・ファイルも対象とするか，COBOL，FORTRANなどの言語に依存するファイルの異機種間の互換性をどこまで考慮するか，ネットワークにおけるファイルの機密保護をどの程度まで管理するかなど，そのネットワークの利用範囲，目的などにより効率との兼合いで実現可能な限度が検討されるべきであろう。互換性の解決には，①意味の表現を統一する方向と，②コンパチブルでない点を遂一カバーする方向と，2つの方向が考えられる。ARPAネットワークのプロトコルの大部分は統一した表現を用いるという考えに沿って開発されたもので，変換法を幾つか規定しておけば，HOSTごとの差異を吸収して統一表現でネットワークを使えるであろうという思想がその根拠になっている。

図4－8にFTPの処理の概念図を示す。

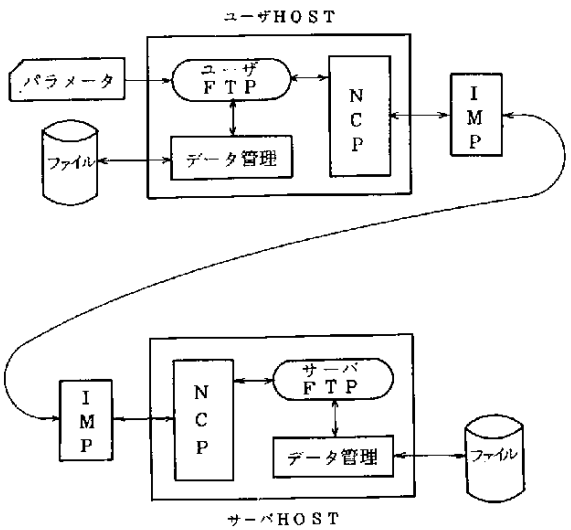


図4－8　FTPの実現方法

サーバＦＴＰをユーザ・プロセスとして作成する場合は，次の問題点がある。

① 現在のＯＳではファイルの創成・抹消などはイニシエータ，ターミネータが行なっており，ユーザにはこの様な機能が開放されていない。

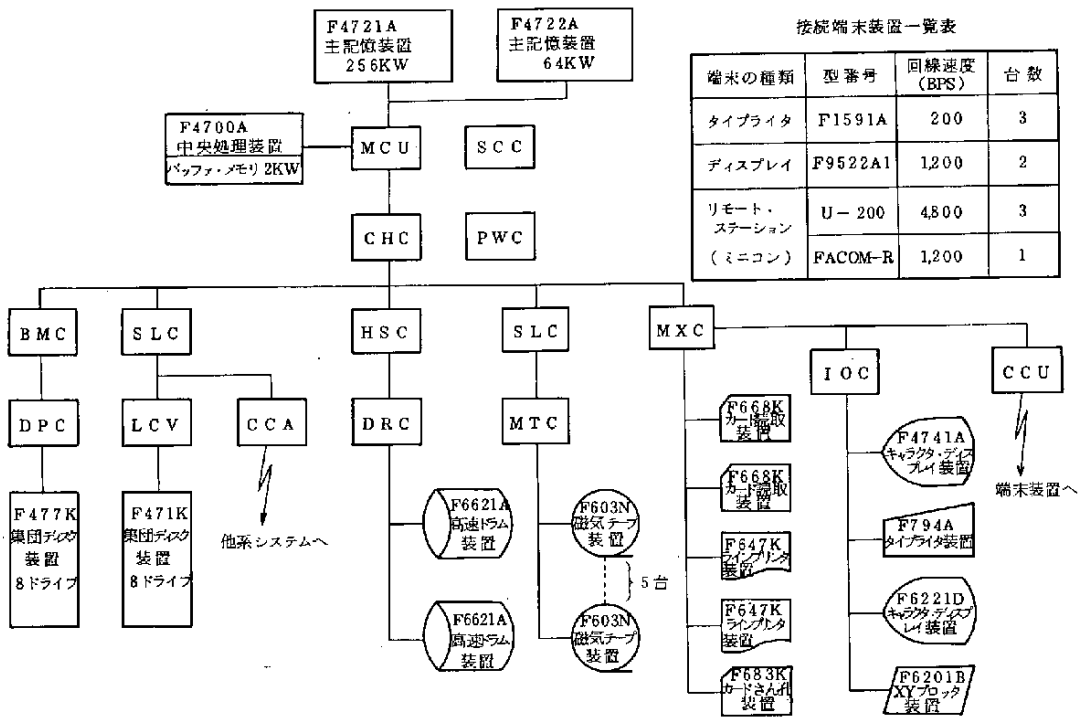
② ファイル保護のため，ユーザ・プログラムでは他人のファイルを読んだり書いたりすることができない。

③ ユーザ・プログラムはデータ管理を介してファイルにアクセスするので色々な制約がある。

4.2 FACOM NCP

4.2.1 F－NCPの概要

FACOM230-75 システムは高性能超大型システムであり，超高速演算機能，大容量主記憶装置（320K語），大容量ランダム・アクセス装置（800Mバイト）などの豊富なリソースを具備し（図4－9参照），バッチ処理，ＴＳＳ処理などをコンカレントにサービスしている。この豊富なリソースやサービスをコンピュータ・ネットワークを介して他系システムのユーザに使用させるため，FACOM 230-75 システム内にＮＣＰ（Network Control Program，以下Ｆ－ＮＣＰと略す）を作成した。



接続端末装置一覧表

端末の種類	型番号	回線速度（BPS）	台数
タイプライタ	F1591A	200	3
ディスプレイ	F9522A1	1,200	2
リモート・ステーション（ミニコン）	U－200	4,800	3
	FACOM-R	1,200	1

図4－9　FACOM 230-75 ハードウェア構成

F－ＮＣＰは，図４－１０ に示すようにベーシック機能（HOST－HOSTプロトコル，HOST－IMPプロトコル，ＮＣＰサービス・コマンド）をサポートするモジュール，ハイレベル・プロトコルをサポートするモジュール，そして他系にＴＳＳを使用させるために追加されたモジュールから構成されている。ベーシック機能用モジュールは，Ｆ－ＮＣＰサブモニタ，NOPEN，NCLOSE，NREAD，NWRITE，LISTEN，SETECBID，NPOST，WAIT，そしてSWAIT サービス・モジュールである。ハイレベル・プロトコルのサービス用モジュールは，DSP User／Server，RBP User／Server，そしてＦＴＰ User／Serverである。ＴＳＳ用追加モジュールは，VOPEN，VCLOSE，VGETS，VPUTS，そしてＮＴＣＰサービス・モジュールである。

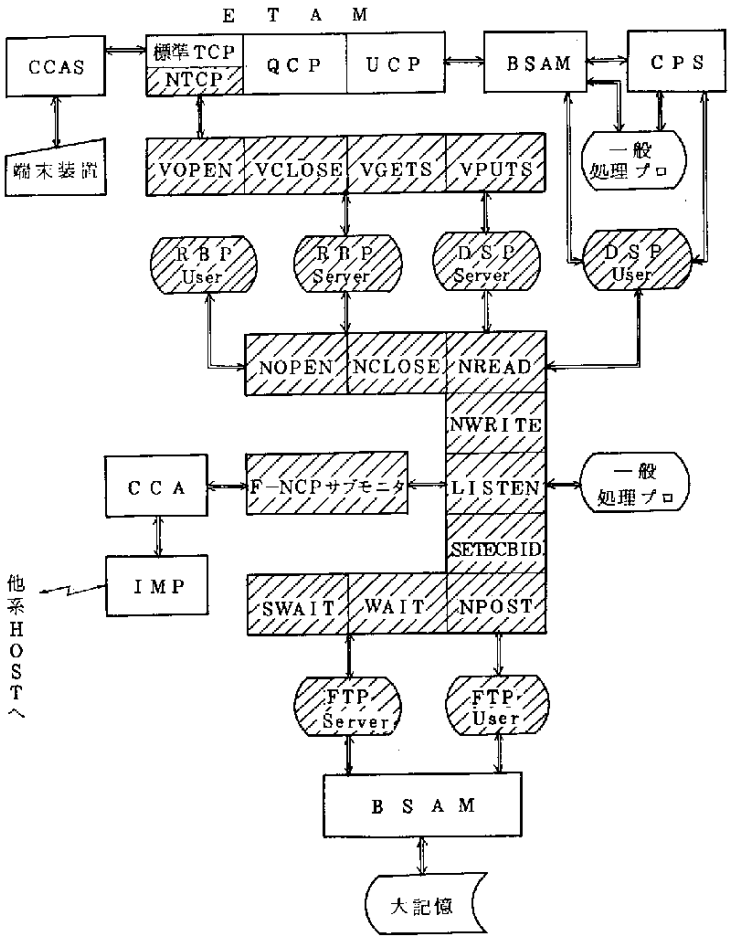


図４－１０　Ｆ－ＮＣＰのソフトウェア構成

F－ＮＣＰではプロセス間コミュニケーション用のアクセス・マクロ命令として，第１バージョンでは，次の７つの機能をサービスしている。

1）コネクション確立要求コマンド …… ＮＯＰＥＮ

2）コネクション切断要求コマンド …… ＮＣＬＯＳＥ

3）メッセージ受信コマンド …… ＮＲＥＡＤ

4）メッセージ送信コマンド …… ＮＷＲＩＴＥ

5）コネクション待ち宣言コマンド …… ＬＩＳＴＥＮ

6）イベント受信宣言コマンド …… ＳＥＴＥＣＢＩＤ

7）イベント通知コマンド …… ＮＰＯＳＴ

以下に，各々のアセス・マクロ命令について述べ，簡単なプログラム例を示す。

(1) ＮＣＢマクロ命令

コネクション確立に必要な情報を含むのがＮＣＢ（Network Control Block）であり，ＮＣＢマクロ命令により作られる。

(a) 書き方

名札欄	命令欄	オペランド欄
ncb名	ＮＣＢ	MACRF＝ アクセス・マクロ命令の種類 ,BLKSIZE＝最大メッセージ長 ,TMOD＝転送モード ,MYPORTID＝ my－port－id ,YRPORTID＝ your－port－id ,PASSWORD＝ パスワード 〔,MYPORTID2＝my－port－id２〕

(b) オペランド

・MACRF　使用するアクセス・マクロ命令の種類を指定する。

（R, N）　NREAD マクロ命令を使用する。

（W, N）　NWRITE マクロ命令を使用する。

・BLKSIZE　伝送されるメッセージの最大長をバイト数で指定する。

・TMOD　送受信されるメッセージの転送モードを指定する。

8　８ビット・モードの転送を意味する。

9　９ビット・モードの転送を意味する。

・MYPORTID　　コネクションの自系側のport-id名を指定する。

'X_1 X_2 ···X_{10}'　16進10桁で指定する。

・YRPORTID　　コネクションの他系側のport-id名を指定する。

'X_1 X_2 ···X_{10}'　16進10桁で指定する。

・PASSWORD　　コネクションのパスワードを指定する。

'C_1 C_2 ···C_6'　6個の文字列で指定する。

・MYPORTID2　　Listen後のオープンの場合，Listenで指定したport名を指定する。

'X_1 X_2 ···X_{10}'　16進10桁で指定する。

(2) NOPENマクロの命令

NOPENマクロ命令はコネクションの確立を要求し，コネクションを介してメッセージの送受を可能にする。

(a) 書き方

名札欄	命令欄	オペランド欄
〔記号名〕	NOPEN	ncb名，MGTYP＝メッセージ形式

(b) オペランド

・ncb名　　コネクションを確立しようとしているNCBの先頭番地を記号名で指定する。

・MGTYP　　メッセージの処理目的を指定する。パラメータとしてはI，O，L，Tがあり，適当に組合せて指定する。指定方法は（〔I〕〔，O〕〔，L〕〔，T〕）である。

I　　入力メッセージとして処理することを指定する。

O　　出力メッセージとして処理することを指定する。

L　　Listen後のオープンであることを指定する。

T　　ネットワーク測定用プログラムからのオープンであることを指定する。

(3) NCLOSEマクロ命令

NCLOSEマクロ命令はコネクションの切断を要求する。

(a) 書き方

名札欄	命令欄	オペランド欄
〔記号名〕	NCLOSE	ncb名〔，OPT＝オプション〕

(b) オペランド

・ncb名　　コネクションを切断しようとしているNCBの先頭番地を記号名で指定する。

・OPT　　ネットワーク測定用プログラムからのクローズの場合指定する。

T　　ネットワーク測定用プログラムからのクローズであることを指定する。

(4) NREADマクロ命令

NREADマクロ命令はコネクションを介して伝送されてきたメッセージを読込むことを指令する。

(a) 書き方

名札欄	命令欄	オペランド欄
〔記号名〕	NREAD	necb名，処理形式，ncb名，領域番地，最大読込み長

(b) オペランド

- necb名　展開した制御表NECB(Network Event Control Block)の先頭に付ける記号名を指定する。
- 処理形式　メッセージの処理方法を指定する。

SM　他系から送信された順にメッセージを読込むことを指定する。

- ncb名　リードするコネクションのNCBの先頭番地を指定する。
- 領域番地　メッセージを読込むべき領域の先頭番地を指定する。但し，領域の先頭番地は偶数番地でなければならない。
- 最大読込み長　読込むべきメッセージの最大長をバイト数で指定する。伝送されてきたメッセージが，指定された最大長を越えた場合，指定されたメッセージ長だけが領域に読込まれる。

(c) 使用上の注意事項

NREAD マクロ命令はサービス依頼を受け付けるといったん制御を処理プログラムに戻すので，適当な時点で necb 番地をパラメータにしてWAIT/SWAITマクロ命令を発信し，NREAD マクロ命令の完了を待たなければならない。その後，正しく完了したかどうかをNECBを調べて確認しなければならない。

(5) NWRITEマクロ命令

NWRITE マクロ命令はコネクションを介してメッセージを送信することを指令する。

(a) 書き方

名札欄	命令欄	オペランド欄
〔記号名〕	NWRITE	necb名，処理形式，ncb名，領域番地，書込み長

(b) オペランド

- necb名　展開した制御表NECB(Network Event Control Block)の先頭に付ける記号名を指定する。
- 処理形式　メッセージの処理方法を指定する。

SM　　自系から発した順に他系へメッセージを送信することを指定する。

・ｎｃｂ名　　ライトするコネクションのＮＣＢの先頭番地を指定する。

・領域番地　　送信すべきメッセージが格納されている領域の先頭番地を指定する。但し，領域の先頭番地は偶数番地でなければならない。

・書込み長　　送信すべきメッセージの長さをバイト数で指定する。

(c) 使用上の注意事項

ＮＷＲＩＴＥマクロ命令はサービス依頼を受け付けるといったん制御を処理プログラムに戻すので，適当な時点でｎｅｃｂ 番地をパラメータにしてＷＡＩＴ／ＳＷＡＩＴマクロ命令を発信し，ＮＷＲＩＴＥマクロ命令の完了を待たなければならない。その後，正しく完了したかどうかをＮＥＣＢを調べて確認しなければならない。

(6) ＬＩＳＴＥＮマクロ命令

ＬＩＳＴＥＮマクロ命令は他系からのコネクション待ちを宣言する。

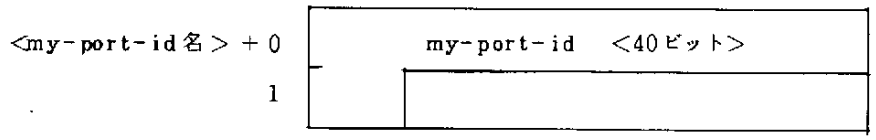
(a) 書き方

名札欄	命令欄	オペランド欄
〔記号名〕	ＬＩＳＴＥＮ	ｅｃｂ 番地，my-port-id 番地，your-port-id 番地

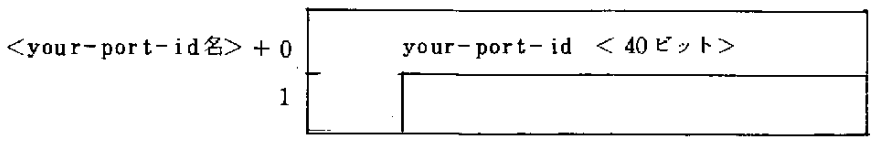
(b) オペランド

・ｅｃｂ番地　　Ｌｉｓｔｅｎの完了を通知する領域の番地を指定する。

・my-port-id番地　　my-port-idが格納されている領域の先頭番地を指定する。その領域には，次の形式でport-id が記述されていること。



・your-port-id番地　　他系よりのオープンで指定された他系のmy-port-id がセットされる領域の先頭番地を指定する。その領域の大きさは２語で，次の形式でport-idがセットされる。



(c) 使用上の注意事項

ＬＩＳＴＥＮ マクロ命令はサービス依頼を受け付けるといったん制御を処理プログラムに戻すので，適当な時点でecb 番地をパラメータにしてＷＡＩＴ／ＳＷＡＩＴマクロ命令を発信し，ＬＩＳＴＥＮマクロ命令の完了を待たなければならない。その後，正しく完了したかどうかを

ＥＣＢを調べて確認しなければならない。

(7) SETECBIDマクロ命令

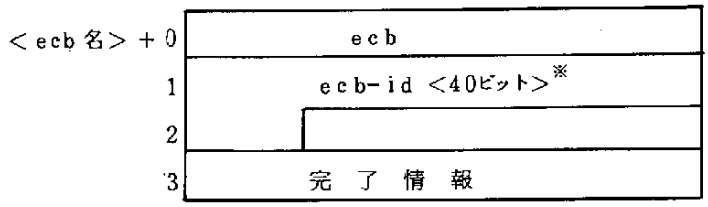
SETECBID マクロ命令は他系からのイベントを受信する領域を宣言する。

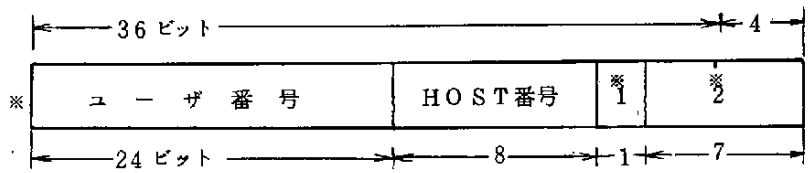
(a) 書き方

名札欄	命令欄	オペランド欄
〔記号名〕	SETECBID	ｅｃｂ番地

(b) オペランド

・ｅｃｂ番地　他系からのイベント受信完了を通知する領域の先頭番地を指定する。その領域の大きさは4語で，次の形式である。





※1：識別ビット（必ず1）

※2：ユーザ指定の識別番号

(c) 使用上の注意事項

SETECBID マクロ命令は受信側 ecb-id を登録する機能だけであり，処理プログラムは適当な時点でｅｃｂ番地をパラメータにしてWAIT/SWAITマクロ命令を発信し，他系からのポストを待たなければならない。その後，正しく完了したかどうかをＥＣＢを調べて確認しなければならない。

(8) NPOSTマクロ命令

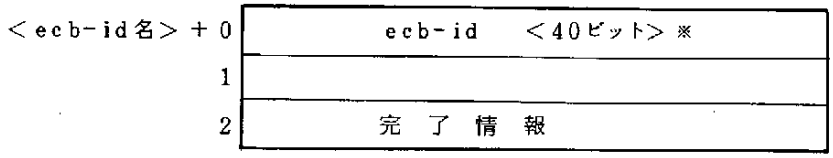
NPOSTマクロ命令は他系にイベントの発生を通知する。

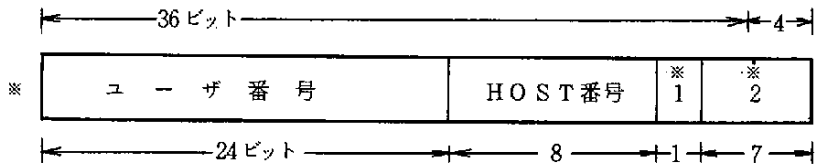
(a) 書き方

名札欄	命令欄	オペランド
〔記号名〕	NPOST	ecb-id番地，通知情報番地

(b) オペランド

・ecb-id番地　通知する他系のecb-id が入っている領域の先頭番地を指定する。その領域の大きさは3語であり，次の形式である。

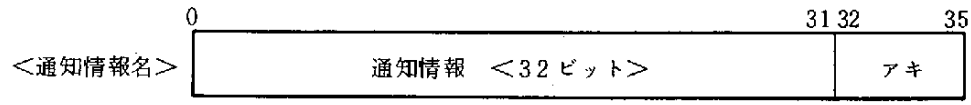




※1：識別ビット（必ず1）

※2：ユーザ指定の識別番号

・通知情報番地　他系に通知する情報が入っている領域の番地を指定する。通知情報は，次の形式である。



(9) プログラム例

図4-11に示すような簡単なプロセス間コミュニケーションの自系側プログラムの例を次に示す。

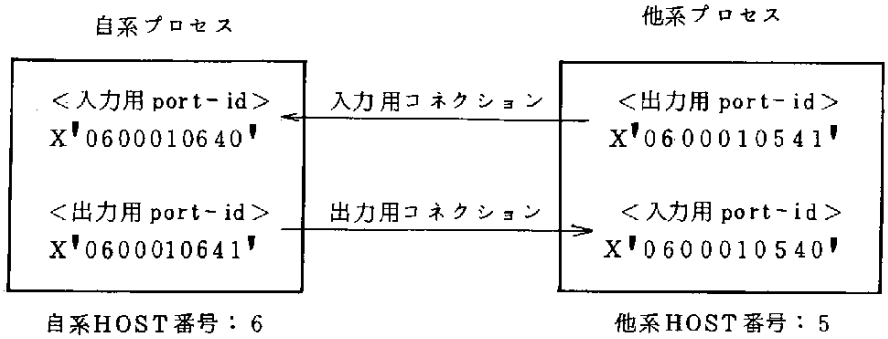


図4－11　簡単なプロセス間コミュニケーションの例

```
            ELEMENT     SAMPLE
START       :                                                    ①
CONNECT     NOPEN       NCBIPT, MGTYP=I                          ②
            TB0F, 0     NCBIPT+15                                ③
            J           OPNERR1                                  ④
            NOPEN       NCBOPT, MGTYP=O                          ⑤
            TB0F, 0     NCBOPT+15                                ⑥
            J           OPENERR2                                 ⑦
            :                                                    ⑧
RECEIVE     NREAD       NECBIPT, SM, NCBIPT, BUFIPT, 144         ⑨
            :                                                    ⑩
            WAIT        NECBIPT                                  ⑪
            TB0F, 0     NECBIPT                                  ⑫
            J           REDERR                                   ⑬
            :                                                    ⑭
SEND        NWRITE      NECBOPT, SM, NCBOPT, BUFOPT, 144         ⑮
            :                                                    ⑯
            WAIT        NECBOPT                                  ⑰
            TB0F, 0     NECBOPT                                  ⑱
            J           WRTERR                                   ⑲
            :                                                    ⑳
            TNE         EOFMARK                                  ㉑
            J           RECEIVE                                  ㉒
BREAK       NCLOSE      NCBIPT                                   ㉓
            TB1F, 0     NCBIPT+15                                ㉔
            J           CLSERR1                                  ㉕
            NCLOSE      NCBOPT                                   ㉖
            TB1F, 0     NCBOPT+15                                ㉗
            J           CLSERR2                                  ㉘
            :                                                    ㉙
```

```
                RETURN    0                                          ㉘
NCBIPT          NCB       MACRF=(R, N),
                          BLKSIZE=144,
                          TMOD=9,                                    ㉙
                          MYPORTID='0600010640',
                          YRPORTID='0600010541',
                          PASSWORD='HIMITU'
NCBOPT          NCB       MACRF=(W, N)
                          BLKSIZE=144,
                          TMOD=9,                                    ㉚
                          MYPORTID='0600010641',
                          YRPORTID='0600010540',
                          PASSWORD='HIMITU'
BUFIPT          RESERVE   36                                         ㉛
BUFOPT          RESERVE   36                                         ㉜
                :                                                    ㉝
                END       START                                      ㉞
```

＜プログラムの説明＞

① このプログラムの実行のための前処理を行なう。

② 入力用コネクションを確立する。コネクション確立に必要なパラメータは，NCBIPTの名前を付けてＮＣＢマクロ命令で宣言してある。

③ NOPENマクロ命令が正常に完了したかどうかをチェックする。

④ コネクションの確立に失敗した場合，OPNERR1にジャンプする。

⑤ 出力用コネクションを確立する。コネクション確立に必要なパラメータは，NCBOPT の名前を付けてＮＣＢマクロ命令で宣言してある。

⑥ NOPENマクロ命令が正常に完了したかどうかをチェックする。

⑦ コネクションの確立に失敗した場合，OPNERR2にジャンプする。

⑧ コネクション確立後，必要ならばメッセージの送受信を行なう前の前処理を行なう。

⑨ メッセージを読みたい場合，NREAD マクロ命令により，そのオペレーションが開始される。既にメッセージが自系内に到着していれば直ちにこのメッセージが指定されたバッファ（BUFIPT）に転送される。そうでなければ，メッセージが到着するまで，このオペレーシ

ョンは完了しない。その終了は適当な時点でWAIT／SWAITマクロ命令によってECBを調べることで知ることができる。

⑩ この間，他の処理を実行することも可能である。

⑪ WAITマクロ命令により，前に発信したNREAD マクロ命令の完了を待つ。

⑫ NREAD マクロ命令が正常に完了したかどうかをチェックする。

⑬ 失敗した場合，REDERRにジャンプする。

⑭ 読込んだメッセージを処理し，出力メッセージを作成する。

⑮ メッセージを送りたい場合，NWRITE マクロ命令によりそのオペレーションが開始される。送信オペレーションが完了する前に，他のオペレーションを実行できるように制御を処理プログラムに戻す。

⑯ この間，他の処理を実行することも可能である。

⑰ WAITマクロ命令により前に発信したNWRITE マクロ命令の完了を待つ。

⑱ NWRITEマクロ命令が正常に完了したかどうかをチェックする。

⑲ 失敗した場合，WRTERRにジャンプする。

⑳ 必要ならば，またRECEIVEに戻りメッセージの送受信を繰り返す。

㉑ このようにして必要なメッセージの交換処理が終了すると，まず入力用コネクションを切断する。

㉒ NCLOSEマクロ命令が正常に完了したかどうかをチェックする。

㉓ コネクションの切断に失敗した場合，CLSERR 1にジャンプする。

㉔ 出力用コネクションを切断する。

㉕ NCLOSE マクロ命令が正常に完了したかどうかをチェックする。

㉖ コネクションの切断に失敗した場合，CLSERR2にジャンプする。

㉗ このプログラムのための後処理を行なう。

㉘ このプログラムの終了をシステムに告げる。

㉙ 入力用コネクションに関係する情報をNCBマクロ命令で宣言する。

㉚ 出力用コネクションに関係する情報をNCBマクロ命令で宣言する。

㉛ 入力用メッセージのバッファを36語確保する。

㉜ 出力用メーセージのバッファを36語確保する。

㉝ この他に必要なエリアを確保する。

㉞ プログラムの実行開始番地（ START ）を指定する。

4.2.3 MONITOR －VIの概要

この節ではMONITOR-VI について，F－NCPを理解するに必要な予備知識を述べる。

A 概 説

MONITOR-VI（以下M-VIと略す）はFACOM230-60／75用のオペレーティング・システムであり，バッチ処理，リモート・バッチ処理，会話型処理，オンライン処理などの処理形態をそれぞれ独立のサブモニタとして作成し，コンカレント処理を行なうことができる。

制御プログラムは，独立に動作可能ないくつかの機能部，すなわち1つのスーパバイザ（特権的サブモニタ），いくつかのサブモニタ（バッチ・サブモニタ，会話型処理サブモニタなど）とサブモニタおよび処理プログラムから依頼されたサービスを行なうサービス・モジュール群から構成される。

(1) スーパバイザとサブモニタ

サブモニタはセンタ・オペレータの操作指令で起動するかまたはシステム編集（System Generation）時の指定により自動的に起動され，操作指令により終了する。スーパバイザはサブモニタおよび処理プログラムをジョブステップという単位でとらえ，これらを統一的に管理する。ジョブステップの管理のために各ジョブステップごとに作業域（これをジョブステップ作業域と呼ぶ）が用意され，制御表という単位で目的・用途別に使用される。サブモニタという概念の導入により，サブモニタの追加など，機能の拡張性などが容易である。さらに各サブモニタは独立に運用可能であり，システム障害により受ける影響を当該サブモニタに限定することができる。

(2) サービス・モジュール

スーパバイザ，サブモニタおよび処理プログラムが，それぞれの処理を進める上で共通に必要とする機能は，システム・マクロ命令にまとめられている。システム・マクロ命令が発せられた場合に，依頼されたサービス機能に対する処理をする部分をサービス・モジュールと呼んでいる。

サービス・モジュールの機能には，ＣＰＵ，主記憶，入出力装置，ボリューム，ファイルなどの各種リソースを有効に使用するための管理や，各種のファイルに対するアクセスなどがある。サービス・モジュールには，タスクとして動作するサービス・タスクと割込み禁止状態（特権命令の使用可，主記憶の保護なし，外部割込み禁止）で動作するサービス・ルーチンの2種がある。どちらも常駐／非常駐の区別があり，さらにいつでも使用可能なもの（初期登録サービス・モジュール）と，サブモニタ起動時などに登録され，ある期間だけ使用できるもの（実行時登録サービス・モジュール）がある。後者はあるサブモニタに固有なモジュールなどに適用され，不要なモジュールによるモニタ・レジデントの増大を防いでいる。サービス・モジュールの作業域はそれを呼び出したジョブステップのジョブステップ作業域を使用する。サービス・モジュールがサービス・モジュールを呼び出した場合には，そのサービス・モジュールの呼出し元をたどっていってみつかったジョブステップ（大もと呼出

し元ジョブステップ）のジョブステップ作業域を使用する。入出力要求の完了を告げる割込みは，要求を出したジョブステップ以外を処理中に発生し，サービス・モジュールが呼び出されることがある。このような場合に備えて，特殊なモニタ専用マクロ命令（ INTRPT）によって，指定したジョブステップから指定したサービス・モジュールが呼び出された状態を擬似的に実現することができる。

(3) その他のモジュール

この他にシステム・サブルーチンと呼ばれる絶対番地で動作するモジュールがある。

以上をまとめると，モニタ・モジュールは図４－12のように分類できる。

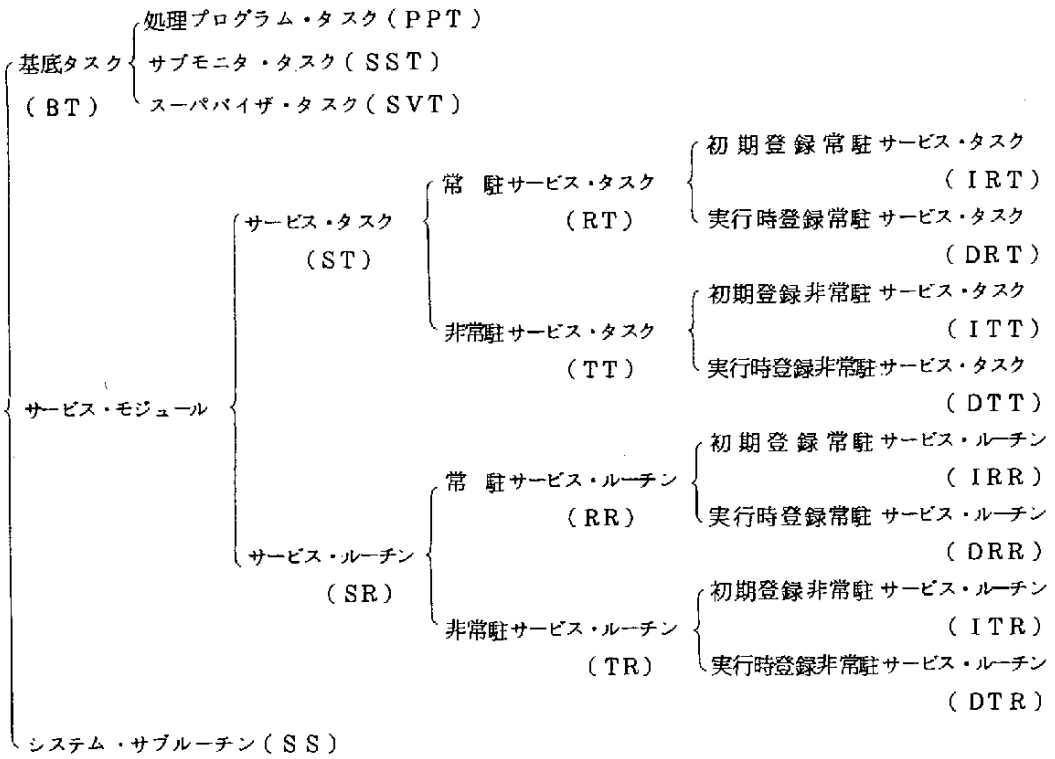


図4－12　モニタ・モジュールの分類

(4) MONITORを構成するファイル

M－VIは数十個のファイルから構成されており，ここではこれらのファイルのうち，M－VIの特徴をよく表わしているものについて述べる。

(a) システム定数ファイル（ＳＣＦ）

オペレーティング・システムは各システムごとに異なる様々な機器構成のもとで稼動し，

また各センターごとに異なる運用形態にも適用しなければならない。従来はシステム編集プログラム（System Generation，略してＳＧ）によって，各々のセンターに適したシステムを作成していた。Ｍ－Ⅵについても各センターに合致したシステムを提供することに変りないが，センター固有の情報およびシステム・パラメータをＳＣＦ（System Constant File）というファイルに収めることによってシステムに柔軟性をもたせ，システム編集作業が簡単に行なえるようになっている。

ＳＣＦの内容はカード・イメージの文字情報であり，ユーティリティ・プログラムLIBE（LIBrary Editor)によって変更することができる。従って，センターの機器構成の変更や運用方法の変更に伴って，センター固有の情報を変更するには，システム全体の大がかりなシステム編集作業を必要とせず，ＳＣＦの一部を変更するだけでよい。

(b) モニタ・プログラム・ライブラリ

モニタ・プログラムは，モニタ・プログラム・ライブラリに格納されている。モニタ・プログラム・ライブラリはSYS1.SUP，SYS２.SUP，……，SYS８.SUP というファイル名をもつ８つのファイルから構成されている。プログラムを開設（ローディング）する際のプログラムを捜す順序はSYS1.SUP，SYS2.SUP，…… の順にさがす。

モニタをいくつかのファイルに分散すると，次のような利点が生じる。

- プログラムを常駐のものと非常駐のものに分け，非常駐のものだけを高速ドラムなどに格納し，ローディング時間を短縮し，しかも高速ドラムの容量をあまり占有しない。
- 非常駐プログラムをいくつかのファイルに分散し，かつそのファイルが占有されているDASDを違ったチャネルに接続することによって，プログラム・ロードのためのチャンネル負荷を均等にできる。
- デバックの場合，モニタ・プログラム・ライブラリ中のプログラムを捜す順序をうまく利用することによってプログラムの入換えが簡単にできる。
- 作成グループごとにファイルを分けてデバッグをすることが可能である。
- ユーザにモニタ・プログラム・ライブラリの一部を開放することができる。

B ジョブステップ

(1) 概 説

Ｍ－Ⅵのスーパバイザは，スーパバイザ，サブモニタおよび処理プログラムのジョブステップを「ジョブステップ」として一律に扱っている。スーパバイザが扱うジョブステップというのは，イニシエートされ，まだターミネートされていないジョブステップだけであり，ジョブ管理でいう普通の意味でのジョブステップとは異なる。

ジョブステップには必ず０～ｎ（ｎ≦511，ｎはシステム編集時に指定）のジョブステップ番号が割り当てられる。

スーパバイザはジョブステップ番号0のジョブステップであり，イニシャル・ロード時にSIP（System Initial Program）によってイニシエートされ，以後主記憶に常駐し，ターミネートされることはない。

サブモニタは，それぞれ1から始まる固有のジョブステップ番号（システム編集時に指定）をもち，スーパバイザや他のサブモニタによってイニシエートおよびターミネートされる。

処理プログラムは，当該サブモニタのジョブステップ制御プログラムによってイニシエートおよびターミネートされる。処理プログラムのジョブステップ番号は，大きい番号nから降順にさがし，使われていない番号が割り当てられる。

ジョブステップは，ジョブステップ毎に専用のジョブステップ作業域をもっている。このジョブステップ作業域は各ジョブステップの起動時に開設され，終了時に閉鎖される。スーパバイザ，サブモニタそして処理プログラムのジョブステップ作業域（単に作業域と呼ぶことがある）は，それぞれスーパバイザ作業域，サブモニタ作業域，そして処理プログラム作業域と呼ぶ。

ジョブステップは，そのジョブステップに1対1に対応する制御表JSCB（Job-Step Control Block）をもつ。JSCBはスーパバイザ作業域の間接参照部分（後述）につくられる。

ジョブステップはジョブステップ毎に領域番号0からある値までの領域番号をもち，それぞれの領域番号に対応した領域をもつことができる。領域番号0の領域は，そのジョブステップの作業域である。

ジョブステップには少なくとも1つのタスクがある。ジョブステップがイニシエートされるときタスクが作られる。このタスクは主タスクと呼ばれ，ジョブステップがターミネートされるまで存在しつづける。

以上をまとめると，ジョブステップは

- ジョブステップ番号
- JSCB（Job-Step Control Block）
- ジョブステップ作業域（領域番号0の領域）
- 領域番号の集合
- 主タスク

と1対1に対応する（図4－13参照）。

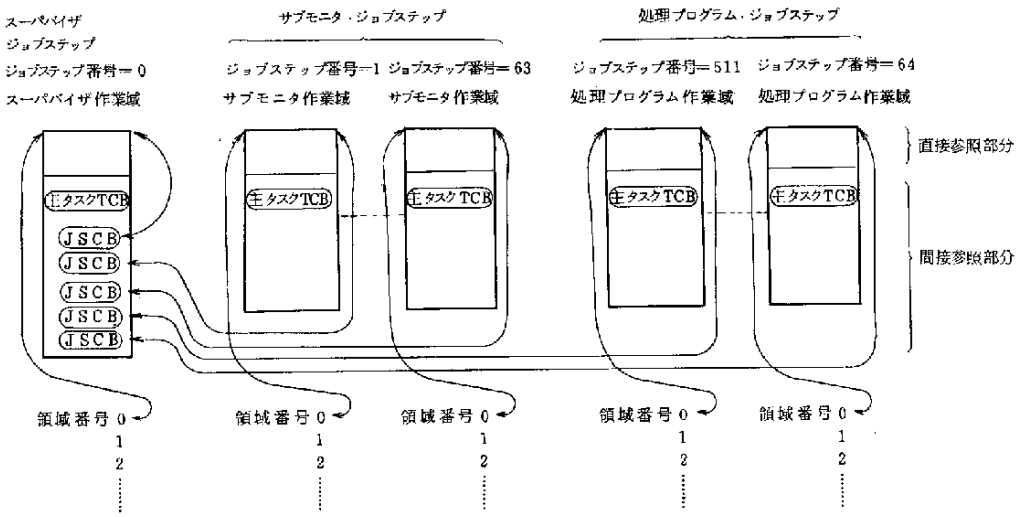


図４－１３　ジョブステップの管理図

(2)　ジョブステップのイニシエートとターミネート

ジョブステップのイニシエータは，最初にＩＮＩＴマクロ命令（ Initiate ）*によりジョブステップ作業域を確保し，次にＩＪＥＸＥＣマクロ命令（ Interjobstep Execute）*により主タスクを起動する。ジョブステップを起動したタスクはＩＪＥＸＥＣ マクロ命令の パラメータで指定したＥＣＢへの通知（ post ）を待つことによって，ジョブステップの終了を検知する。

ジョブステップのターミネータは，ＳＲＣＭマクロ命令（ Scratch Memory)*でジョブステップ作業域以外の領域を返却し，ＴＥＲＭマクロ命令（ Terminate)*でジョブステップを終了する。

処理手順の概略を図４－１４に示す。

＊　モニタ専用マクロ命令

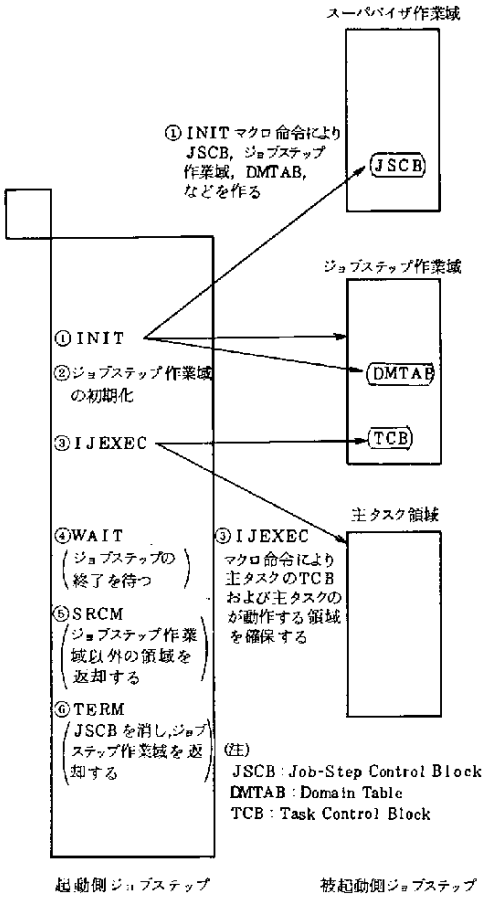


図4－14　ジョブステップのイニシエート／ターミネート

(3)　ジョブステップ間通信

各ジョブステップはお互に干渉しないことを原則とするが，場合によってはお互いに通信したい場合がある。そのためM－VIでは次に示すモニタ専用マクロ命令がある。

- IJABORT　指定されたジョブステップまたはそのジョブステップ内のタスクを異常終了させる。
- IJENQ　あるジョブステップのタスクと別のジョブステップのタスクがあるリソースを共用し，同時に使用することができない場合，このマクロ命令で使用申請を行なう。
- IJEXEC　新しくジョブステップの主タスクを発生し，起動する。
- IJDEQ　IJENQマクロ命令で行なった使用申請の終了を宣言する。
- IJPOST　別のジョブステップのタスクがWAIT／SWAIT マクロ命令で指定されたECB(Event Control Block)に事象が発生したことを通知する。

・IJSGB　　指定された別のジョブステップの指定された領域にベース・レジスタとガード・レジスタを設定する。

・INTRPT　　指定されたサービス・モジュールを指定された別のジョブステップの指定されたタスクが呼出した状態にする。

C 領　域

(1) 概　説

M－VI スーパバイザが，番地が連続している主記憶空間として認識する単位を領域と呼ぶ。一般には，領域の主記憶空間上での位置（先頭番地）は可変であり，領域中の番地の相対位置は一定である。領域と領域間の位置の差は可変である。

領域は記憶保護，プログラムのロード（領域単位の開設）の単位でもある。

領域は連続した主記憶空間上におかれ，ロール・アウト（スワープ・アウト）されるとロール・アウト・ファイル上におかれる。しかし，スーパバイザやサブモニタはロール・アウトされることはない。

(2) 番地の表現形式

ジョブステップを固定すると，そのジョブステップ内1語の主記憶上での位置は，領域番号と領域内相対番地からなる表現形式（＜dw＞形式と呼ぶ）で一意に定まる。<dw>形式は，領域がロール・アウトや主記憶内のリアロケーションが起っても不変であり，ロール・アウトや主記憶内リアロケーションが起りうる場合＜dw＞形式で記録しておく必要がある。その他の番地の表現形式には，＜db＞，＜dvw＞，＜bvw＞，<bw>,および絶対番地がある。ただし，dは領域番号，bは修飾するベース・レジスタの番号，vは修飾するインデックスレジスタの番号，そしてwは領域内相対番地を示す。

他のジョブステップに影響を与えないような一般のシステム・マクロ命令のパラメータの番地の表現形式は，＜db＞形式が使われる。

D ジョブステップ作業域

(1) 概　説

ジョブステップ作業域は，ジョブステップ毎につくられる作業域で，スーパバイザ作業域，サブモニタ作業域，そして処理プログラム作業域の3種がある。

あるジョブステップの属するタスク及びそのタスクが呼び出したサービス・タスクは，作業域としてそのジョブステップ作業域を使用することを原則とする。

ジョブステップ作業域は直接参照部分と間接参照部分の2つに分けて使用される（図4－15参照）。直接参照部分には，特定の個数だけ存在し，大きさが変化しないような制御表をおく。このような制御表は直接制御表と呼ばれる。間接参照部分には，必要とする個数や大きさが変化する制御表をおく。このような制御表は間接制御表と呼ばれる。間接参照部分

はその中のどの部分が使用中であるか，どの部分が「空」かの管理がなされていて，サービス・モジュールなどが必要とする作業域はここからとられる。

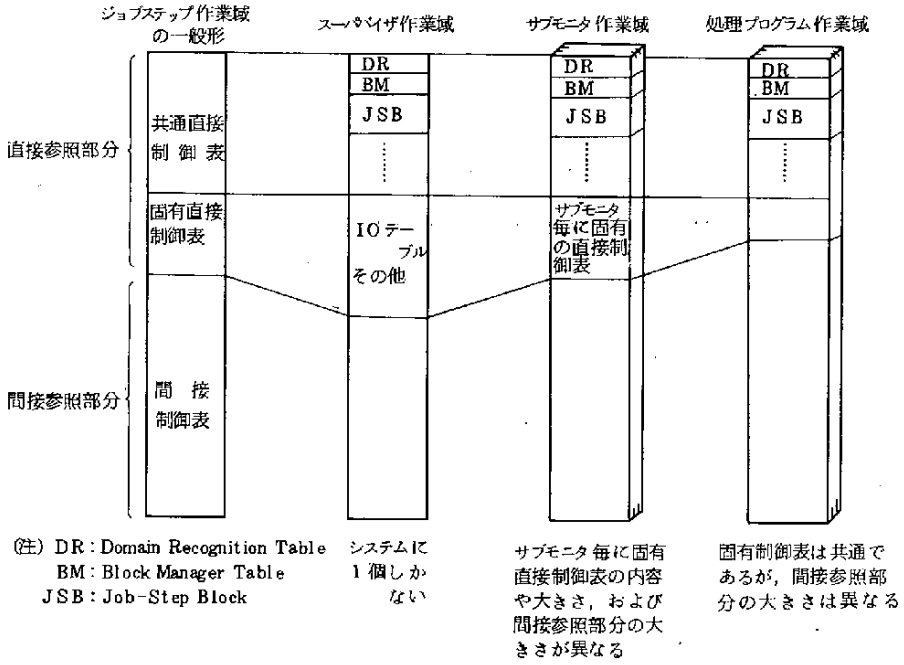


図４－１５　ジョブステップ作業域の構造

(2)　直接制御表

直接制御表には，スーパバイザ，サブモニタ，そして処理プログラムに共通する共通直接制御表と，各ジョブステップ毎に独立な内容をもつ固有直接制御表がある。

共通直接制御表には，DR(Domain Recognition Table)，BM(Block Manager Table)，JSB(Job-Step Block)，RBWQ(Request Block Wait Queue) などがある。

直接制御表の内容は，初期設定時に初期化される。すなわち，スーパバイザの直接制御表は，スーパバイザの初期設定時にＳＩＰによって，ＳＣＦの内容に従って初期化が行なわれる。サブモニタの直接制御表は，各サブモニタをイニシエートするモジュールやサブモニタの主タスクによって初期化される。処理プログラムの直接制御表は，イニシエータによって初期化される。

(3)　間接制御表

間接制御表は，間接参照部分からGETBi マクロ命令*によって借受け使用し，不要になったらFREEBiマクロ命令*によって間接参照部分に返却する。間接制御表の大きさは 2^n語（$1 \leqq n \leqq 9$）であり，先頭の１語は間接参照部分の管理のため使用されるので，各間接制

＊モニタ専用マクロ命令

御表の実際に使用できる大きさは $2^n - 1$ 語である。

間接制御表にはすべて制御表識別番号がつけられている。すなわち，間接制御表の先頭の1語には，制御表識別番号，使用中／空きの表示，制御表の大きさ（語数）が記入されている（図4－16参照）。

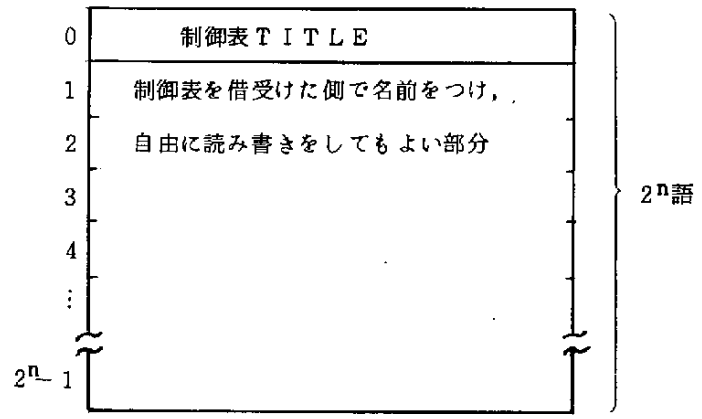


図4－16　間接制御表の一般形

E　サブモニタ

(1) 概　説

M－VIで現在サポートしているサブモニタとしては，BATCH，ETAM，CPSがある。サブモニタのジョブステップ番号は，BATCHなら1，ETAMならば2，ＣＰＳならば3というように，サブモニタ毎に決まっている。

サブモニタは，操作員がコンソールより打鍵したSTART指令により，指定されたサブモニタがイニシエートされる。サブモニタをイニシエートするに必要な情報はＳＣＦに記入されている。

ＳＣＦの内容としては，

- サブモニタ名
- サブモニタ作業域の大きさ
- サブモニタ作業域の直接参照部分の大きさ
- ジョブステップ番号
- 使用する領域の個数
- 主タスクの実行優先権
- 当該サブモニタを起動する前に起動されていなければならないサブモニタ名

- 最大主記憶容量
- システム初期設定時に自動スタートするかどうかの区別

などである。

サブモニタの正常な完了は，操作員がコンソールより打鍵したＳＴＯＰ指令により行なわれる。STOP指令で，サブモニタは「ジョブステップの完了作業をせよ」という作業依頼を受け種々の後処理を行なったのち，RETURNまたはABORTマクロ命令でジョブステップの完了を宣言する。サブモニタの完了が通知されると，スーパバイザは，サブモニタが使用していた領域を返却し，ジョブステップの消滅処理を行なう。

(2) レジスタの使用規約

サブモニタで使用可能なレジスタのうち，いくつかのレジスタは決められた目的以外に使うことができない。M－VIの元でサブモニタを作成した場合に使用可能なレジスタは，７個のベース・レジスタ（B0，B1，B2，B4，B5，B6，B7），５個のガード・レジスタ（G1，G2，G3，G4，G5），８個のインデックス・レジスタ（X0，X1，X2，……，X7），Ａレジスタ，Ｒレジスタ，Ｚレジスタ，そして２個のバイト・カウンタ（ＢＣ６，ＢＣ７）などである。

ベース・レジスタとガード・レジスタは機械語命令で直接変えることはできず，必ずスーパバイザ・マクロ命令によって値を変える。

以下に，各レジスタの使用規約を示す（図４－１７参照）。

Ｂ１　実行領域の先頭絶対番地

Ｂ２　空，一般にはジョブステップ作業域内の絶対番地，またはGETDM領域内の絶対番地のうちのどれかに設定

Ｂ４　ジョブステップ作業域の先頭絶対番地

Ｂ５　一般には他のジョブステップのジョブステップ作業域かGETDM領域，またはその他のうちのどれかの先頭絶対番地を設定

Ｂ６　Ｂ５と同じ

Ｂ７　Ｂ５と同じ

Ｇ１　Ｂ４に設定されている領域を読み書き可能（ＲＷ属性）にする。

Ｇ２　Ｂ５に設定されている領域を読み書き可能（ＲＷ属性）にする。

Ｇ３　全主記憶を読み実行可能（ＲＸ属性）にする。

G4，G5 未使用

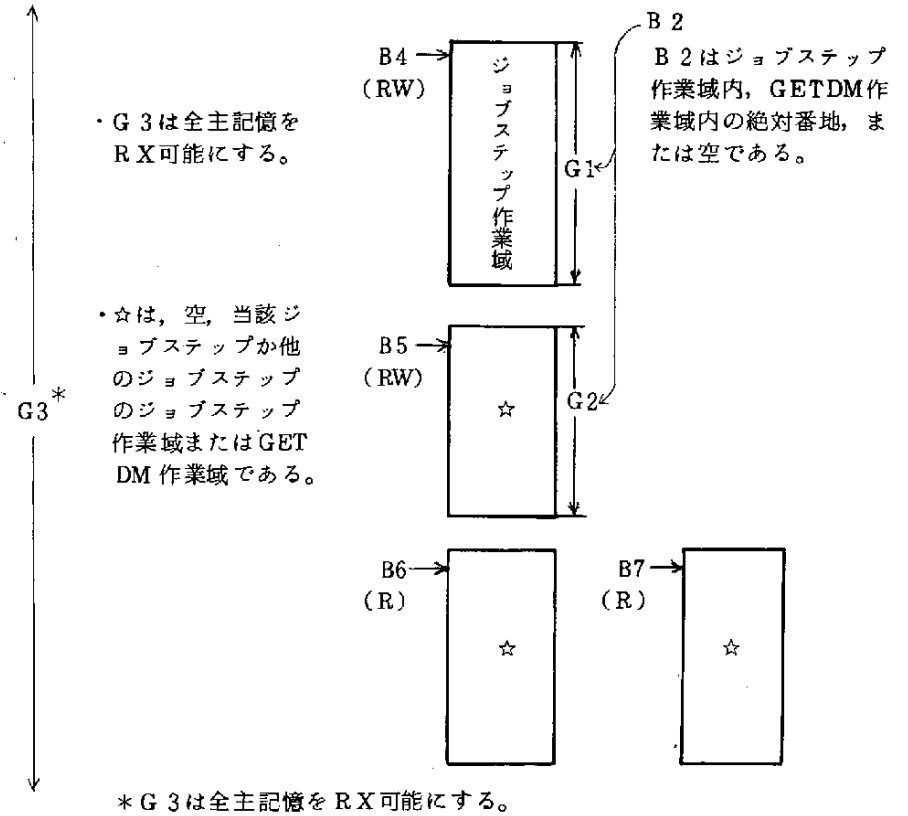
Ｘ０ ～ Ｘ７　自由に使用してよい（初期値：０）。

A　　自由に使用してよい（初期値：０）。

R　　自由に使用してよい（初期値：実行領域の開始番地＜ｄｗ＞）。

Z　　自由に使用してよい（初期値：タスク番号）。

ＢＣ６
ＢＣ７　　自由に使用してよい（初期値：０）。



＊Ｇ３は全主記憶をＲＸ可能にする。

図４－１７　サブモニタのレジスタ使用規約

F　サービス・モジュール

(1) 概　説

基底タスクが要求するサービスをするモジュールをサービス・モジュールと呼ぶ。サービス・モジュールは，タスク構造かルーチン構造かの区別，常駐か非常駐かの区別，そして初期登録か実行時登録かの区別があり，これらの組合せで８種類に分けられる。

サービス・モジュールの所属を明確にするには，サービス・モジュールの領域の所属，サービス・モジュールの登録元，サービス・モジュールの呼出し元の三つについて述べる必要がある。サービス・モジュールの領域の所属は，どんな場合でもスーパバイザ・ジョブステッ

プである。サービス・モジュールの登録元ジョブステップは，サブモニタまたはスーパバイザである。サービス・モジュールはいくつものサブモニタから登録されることがあり，このような場合には，サービス・モジュールは複数個の登録元をもつことになる。サービス・モジュールの呼出し元はサービス・モジュールを呼出したジョブステップであり，登録元とは必ずしも一致しない。サービス・モジュールは呼出し元のジョブステップのジョブステップ作業域を使ってサービスをする。

サービス・モジュールの領域の所属はスーパバイザであり，これらのサービス・モジュールの領域を一意に識別するため，領域番号（Domain Number）とセグメント番号（Segment Number）が付けられる。ＤＳ番号とは領域番号と相対セグメント番号を合せたものの総称であり，サービス・モジュールとＤＳ番号は１対１に対応する。

処理プログラムから呼び出されたり，広く使われるサービス・モジュールには，ＤＳ番号の他にＳＦ番号（Supervisor Function Number）が付けられる。しかし，全サービス・モジュールがＳＦ番号をもっているわけではない。ＳＦ番号もＤＳ番号と同様にサービス・モジュールを識別するのに使うことができ，両者をあわせてモジュール番号と呼ぶことがある。

(2) サービス・モジュールの呼出し関係

サービス・モジュールの呼出し関係を述べる場合，「呼出し」，「受渡し」そして「戻り」という用語をはっきりさせておく必要がある。

あるサービス・モジュールが呼び出された場合，そのモジュールだけですべての処理が終ってしまえば，呼出し元に戻って一連の処理（サービス）が終了する。しかし呼び出されたモジュールだけではプログラムの大きさの制限があるため，全処理が終らない場合がある。このような場合は，呼び出されたモジュールは，別のモジュールに制御を受け渡し，受け渡されたモジュールから呼出し元に戻ることになる。このような制御の受け渡しは，何重にもすることができる（図４－１８参照）。

サービス・モジュールは基底タスクから機械語のＴＲＰ命令*で呼び出される。基底タスクから呼び出されたサービス・モジュールが，またサービス・モジュールを呼び出すことがある。サービス・モジュールからサービス・モジュールの呼出しは，呼出し関係の許す範囲で何重にもすることができる（図４－１９参照）。

＊　モニタ・コール用機械語命令

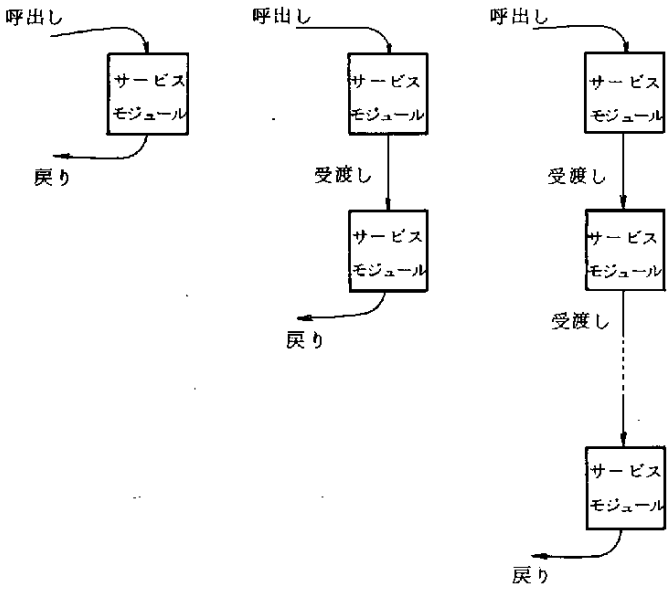


図 4-18　サービス・モジュールの呼出し／受渡し／戻りの関係

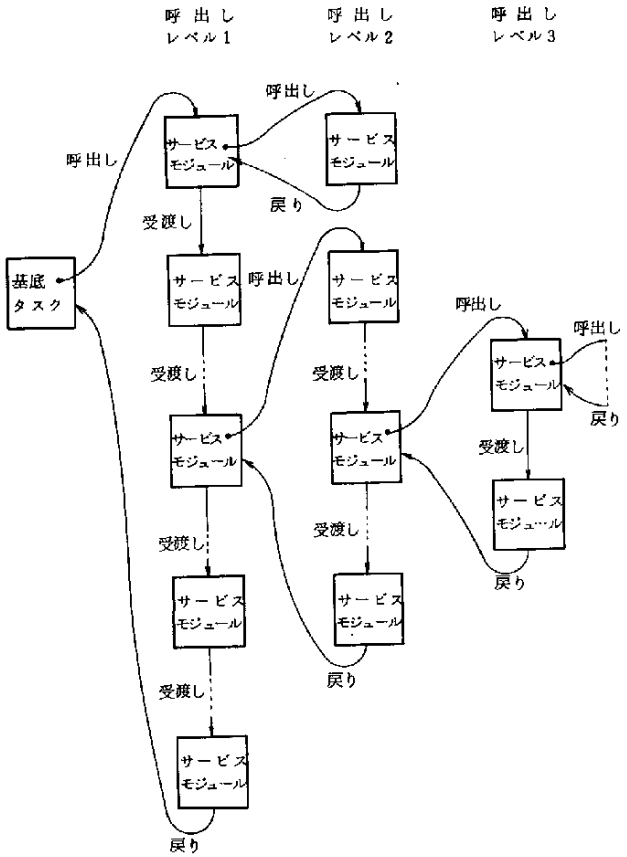


図 4－19　サービス・モジュールの呼出しレベル

(3) サービス

ある基底タスクがサービス・モジュールを呼出した時，その基底タスクに対する1つのサービスが発生したと考える。サービスは呼出しレベル1からの戻りで消滅すると考える。

サービスは次の2つの場合に発生する。

① 基底タスクがTRP命令でサービス・モジュールを呼出した場合

② モニタ・モジュールがある基底タスクに対してINTRPTマクロ命令を発した場合

いずれの場合にも，発生したサービスは基底タスクに対して何らかの処理をする。この一連の処理はシーケンシャルになされ，並行処理はされない。

しかし，呼出し元基底タスクとサービスは並行に処理されることがある。これは，サービスが消滅しないうちに，呼出し元基底タスクを動作(wake-up)させた場合に起きる。

ある基底タスクに対するサービスは同時に複数個存在する可能性がある。これは次の理由による。ある基底タスクがTRP命令でサービス・モジュールを呼出すことによって，1つのサービスを発生させたとする。このサービスが消滅する前に，呼出し元の基底タスクを実行可能にすることができる。そして，実行可能になった基底タスクはまたサービス・モジュールを呼出すことによって，サービスを発生させることができる。この場合，以前に発生させたサービスがまだ消滅していなければ，この基底タスクに対するサービスが同時に複数個存在することになる。もう1つの場合はINTRPTマクロ命令によるサービスの発生であり，対象の基底タスクと非同期的に起きることによる。ある基底タスクに関して，複数個のサービスが動作している場合，これらは並行的に処理される。

(4) サービス・ルーチン

サービス・ルーチンは(一般的いうと)TRP命令で呼び出されるサブルーチンである。サービス・ルーチンは外部割込み禁止モードで動作し，全主記憶が実行可で，読出し可で，書込み可となり，全命令セット(特権命令も)を使用することができる。

サービス・ルーチンの中では，WAIT，SWAIT，ENQ，WTIMEマクロ命令などのような呼出し元のタスクを止めてしまうようなマクロ命令は使用することができない(このようなマクロ命令はタスク構造のモジュールしか呼出すことができない)。

サービス・ルーチンはベース・レジスタB1修飾で動作する。

サービス・ルーチンに制御を移す方法には

- 呼出し
- 制御の受渡し

の2種がある。

呼出しは

・タスクからのＴＲＰ命令による呼出し

・サービス・ルーチンからのＴＲＰ命令による呼出し

・サービス・タスクからのＴＥＸＩＴマクロ命令*による呼出し

の３種がある。

制御の受渡しとは，サービス・ルーチンからの制御の受渡しを言う。

(a) タスクからのＴＲＰ命令による呼出し

タスクがＴＲＰ命令でサービス・ルーチンを呼出すと，サービス・モジュール制御は呼出し時のレジスタを呼出し元ジョブステップのジョブステップ作業域にあるＴＣＢ（Task Control Block）にセーブし，そのＴＣＢの相対番地をインデックス・レジスタ２（Ｘ２）にセットし，スーパバイザ作業域にＨＫ領域（House-Keep Area）を確保し，ＨＫの相対番地をインデックス・レジスタ１（Ｘ１）にセットし，サービス・ルーチンに制御を移す（図４－２０参照）。

サービス・ルーチンは携えてきたＨＫの一部を作業域として使用することができる。ＨＫの作業域として使用できる部分（16語）はHKWKと呼ぶ。HKWKは，制御を受渡す場合に，制御を渡すべきモジュールへ渡すパラメータをおさめるためにも使用される。

図４－２０　タスクからサービス・ルーチンの呼出し

(b) サービス・タスクからのＴＥＸＩＴマクロ命令による制御の受渡し

サービス・タスクがサービス・タスクの終了を告げるＴＥＸＩＴマクロ命令によって，サービス・ルーチンへ制御を渡すと，サービス・モジュール制御はＨＫを携えてサービス・ルーチンに制御を渡す。この場合，ＨＫのHKWKには，サービス・タスクからのパラメータ（RBPRM）がコピーされる（図４－２１参照）。

＊　サービス・タスクの終了を告げるモニタ専用マクロ命令

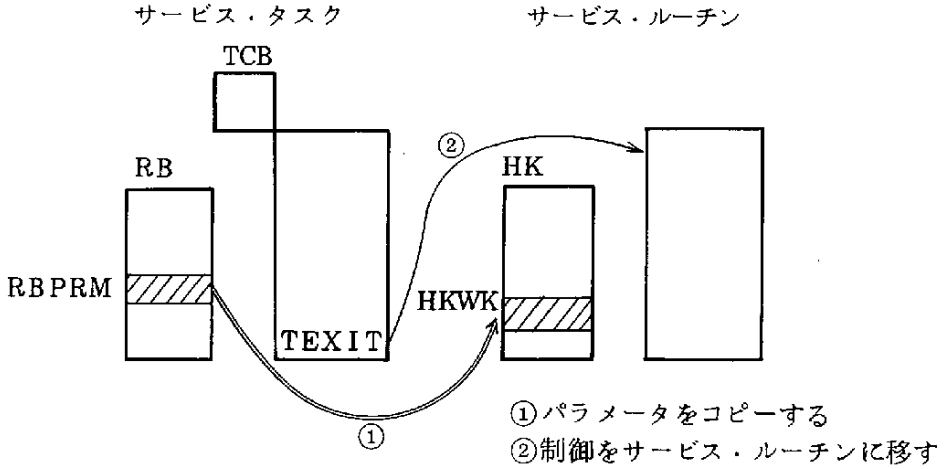


図４－２１　サービス・タスクからサービス・ルーチンへの制御の受渡し

(c)　サービス・ルーチンからの制御の受渡し

サービス・ルーチンAからサービス・ルーチンの終了を告げるREXIT マクロ命令によってサービス・ルーチンBに制御を渡す場合，サービス・モジュール制御はＨＫそのものを携えて制御を受け渡すべきサービス・ルーチンＢに制御を移す（図４－２２参照）。

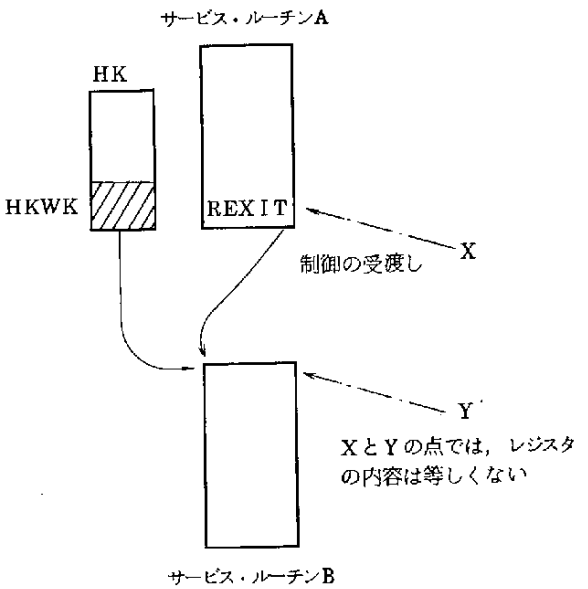


図４－２２　サービス・ルーチンからサービス・ルーチンへの制御の受渡し

制御を受渡す場合，サービス・ルーチンAの出口時点のレジスタの値が，サービス・ルーチンBの入口時点ではこわされている。

(d) レジスタの使用規約

サービス・ルーチンで使用可能なレジスタの使用目的は，各レジスタ毎に決められており，決められた目的以外に使用することができない。

以下に，各レジスタの使用規約を示す（図4－23参照）。

B1　実行領域の先頭絶対番地

B2　初期値不定，自由に使用してよい。

B4　呼出しのおお元の基底タスクのジョブステップ作業域の先頭絶対番地

B5　スーパバイザ作業域の先頭絶対番地

B6　B4またはB5の内容と同じ，自由に使用してよい。

B7　初期値不定，自由に使用してよい。

X1　HKの相対番地

X2　TH（呼出し元のTCBまたはHK）の相対番地

X3　CPU番号

X0，X4～X7　初期値不定，自由に使用してよい。

その他　初期値不定，自由に使用してよい。

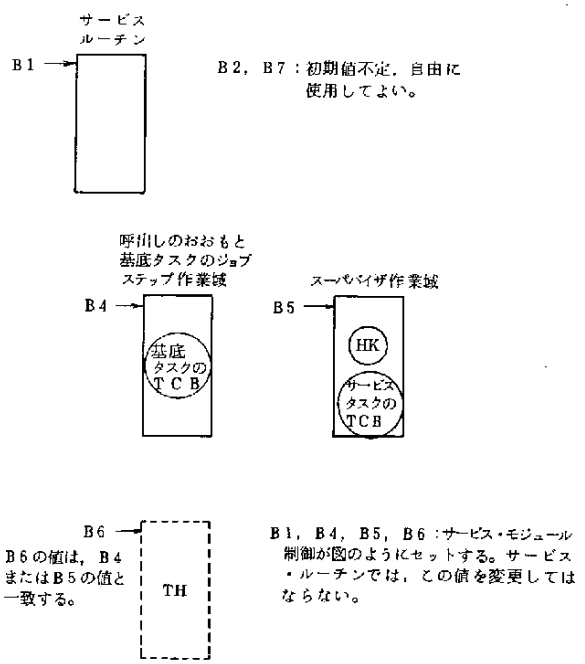


図4－23　サービス・ルーチンのベース・レジスタ使用規約

(5) サービス・タスク

サービス・タスクはタスクとして動作する。サービス・タスクは外部割込み可能状態で動作し，外部割込みによりタスクの実行は中断されるが，タスク・スイッチ中断が起きないように制御されている。サービス・タスクでは，サービス・ルーチンではできなかった待ちを必要とする処理をすることができる。サービス・タスクは外部割込み可で動作するため，主記憶保護機構に従い，特権命令を実行することができない。

サービス・タスクはベースレジスタＢ１修飾で動作する。サービス・タスクのＴＣＢは，スーパバイザ作業域に作られる。

サービス・タスクに制御を移す方法には

- ＩＮＴＲＰＴマクロ命令による呼出し
- ＴＥＸＩＴマクロ命令による呼出し
- 制御の受渡し
- その他の呼出し（ＴＲＰ命令による呼出しなど）

の４種類がある。

(a) タスクからＴＲＰ命令による呼出し

タスクがＴＲＰ命令によってサービス・タスクを呼出すと，サービス・モジュール制御は，呼出し時のレジスタを呼出し元ジョブステップのジョブステップ作業域にあるＴＣＢにセーブし，そのジョブステップ作業域にＲＢ領域（Request Block)を確保する。ＲＢ中のRBBTCBに呼出し元のＴＣＢの相対番地をセットし，ＲＢの相対番地をＸ１にセットし，サービス・タスクに制御を移す（図４－２４参照）。

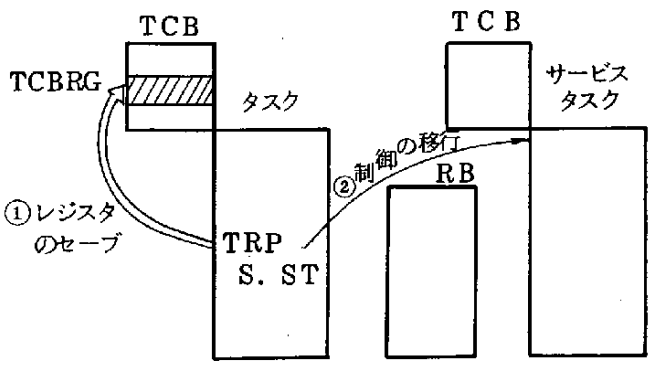


図４－２４　タスクからサービス・タスクの呼出し

この他の場合については，サービス・ルーチンとほぼ同様であるので説明は省略する。

(b) レジスタの使用規約

サービス・タスクで使用可能なレジスタの使用目的は，各レジスタ毎に決められており，決められた目的以外に使うことができない。

以下に，各レジスタの使用規約を示す（図4－25参照）。

B1　実行領域の先頭絶対番地

B2　自由に使用してよい。

B4　呼出しおお元基底タスクのジョブステップ作業域の先頭番地

B5　サービス・タスク毎に決められており，サブモニタ作業域かスーパバイザ作業域の先頭番地

B6　スーパバイザ作業域の先頭番地，自由に使用してよい。

B7　自由に使用してよい。

G1　B4に設定されている領域を読み書き可能（RW属性）にする。ただし，処理プログラム基底タスクの場合，処理プログラム域のすべてを書き込み可能にする。

G2　B5に設定されている領域を読み書き可能（RW属性）にする。

G3　全主記憶を読み実行可能（RX属性）にする。

X1　RBの先頭相対番地

X0，X2～X7　自由に使用してよい。

その他　自由に使用してよい

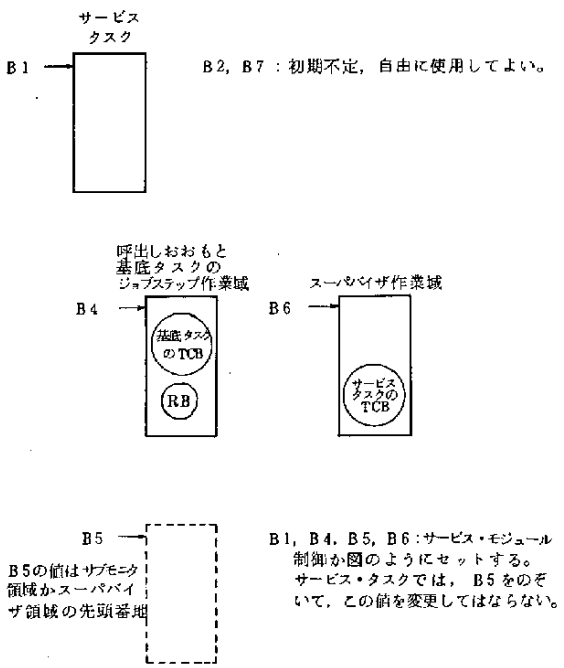


図4－25　サービス・タスクのベース・レジスタ使用規約

4.2.4　F－NCPの基本構想

F－NCPは，次の諸点を考慮し設計された制御プログラムである。

① バッチ処理プログラムと同程度の知識でプロセス間コミュニケーション用処理プログラムの作成を可能にする。

② プロセス間コミュニケーション用アクセス・マクロ命令は順編成アクセス・マクロ命令と同一形式にする。

③ 他系HOSTやＴＩＰよりＴＳＳサービスを利用できるようにする。

F－NCPは実際に他系HOSTとメッセージの交信を行なうサブモニタ部と，処理プログラムからのサービス依頼を受け付けるサービス・モジュール部から構成されている（図４－２６参照）。

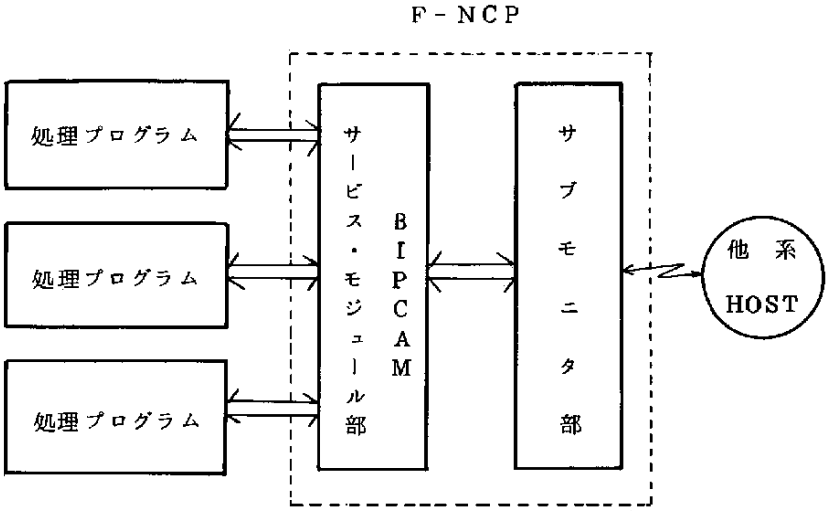


図４－２６　　F－NCPと処理プログラムの連繋

サービス・モジュール部は処理プログラムからのメッセージ送信依頼の場合，そのメッセージをサブモニタ部のメッセージ送信待ち行列に登録し，メッセージ受信依頼の場合，他系からメッセージが到着した時サブモニタ部から呼出され，メッセージを処理プログラムの指定された領域に転送するなどの機能を受け持つ。

処理プログラムとのインタフェースは，他系のプロセスを仮想順編成ファイルとみなし，順編成アクセス・マクロ命令と同一インタフェースとしている（ただし，アクセス・マクロ名は異なる）。このアクセス法をBIPCAM（Basic Inter-Process Communication Access Method）と呼ぶ。

4.2.5　F－NCPサブモニタ部

A　M－VIシステムにおけるF－NCPサブモニタの位置

F－NCPサブモニタはM－VIシステム内の1サブモニタである。サブモニタとはあるまとまった処理単位の制御プログラムであり，M－VIシステムではBATCH，ETAM，CPSなどがある（図4－27参照）。

①　F－NCPサブモニタの独立

各サブモニタは完全に独立しており，各サブモニタの管轄内で生じる各種異常状態は，他サブモニタに影響を及ぼすことはなく，当該サブモニタ単位内で処理される。従って，ネットワーク・サービス中に他のサブモニタのトラブルで，ネットワーク・サービスの中断という事態をさけることができる。

②　起動と停止

F－NCPサブモニタの起動はシステム・コンソールからの操作員指令の1つであるSTART 指令で行なわれ，停止は操作員指令のSTOP 指令による。

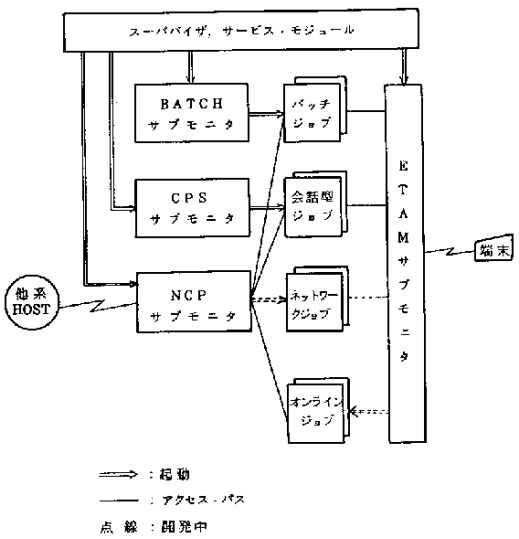


図4－27　F－NCPを組込んだM－VIシステム

B　F－NCPサブモニタ部を構成するモジュール

F－NCPサブモニタ部は，次の6個の機能モジュールから構成されている。

①　主モジュール……N．MAIN

主モジュールはF－NCPサブモニタ制御の中核部分であり，F－NCPサブモニタ全体

の制御を行なう。

② 初期設定モジュール……N．INTLZ

初期設定モジュールは，サブモニタの初期化を行なう。

③ ＣＣＡ用入出力モジュール……N．CCARTN

ＣＣＡ用入出力モジュールは，ＣＣＡを介してHOSTとＩＭＰとの間で物理的にメッセージの送受を行なう。

④ 時間監視モジュール……N．TMSPVR

時間監視モジュールは，Ｆ－ＮＣＰサブモニタの時間監視を行なう。

⑤ 制御メッセージ受信モジュール……N．CTLRCV

制御メッセージ受信モジュールは，他系より受信した制御メッセージの一括管理を行なう。

⑥ 終了処理モジュール……N．TERM

終了処理モジュールは，Ｆ－ＮＣＰサブモニタの終了処理を行なう。

コンソールよりSTART コマンド投下で，スーパバイザによりIJEXEC マクロ命令（①）で起動された主モジュールは，初期設定モジュールをLINKマクロ命令（②）で起動する。初期設定モジュールは初期設定後，EXECマクロ命令（③，④，⑤）によって順次ＣＣＡ用入出力モジュール，時間監視モジュール，制御メッセージ受信モジュールを起動し，RETURNマクロ命令（⑥）を発し消滅する。初期設定完了後，主モジュールはサブモニタの初期設定終了，すなわち準備完了をスーパバイザに通知するため，スーパバイザ作業域にあるSTARTECBをＩJPOST マクロ命令（⑦）でポストする。

コンソールよりSTOPコマンドが投下されると，スーパバイザは主モジュールにIJQONマクロ命令（⑧）で通知する。主モジュールは終了処理モジュールをLINKマクロ命令（⑨）で起動する。終了処理モジュールは終了処理を完了してRETURNマクロ命令（⑩）を発し消滅する。主モジュールはすべてが終了したのでRETURNマクロ命令（⑪）を発し消滅する。以上により，Ｆ－ＮＣＰサブモニタは閉鎖される。

これらの流れを図４－２８に示す。

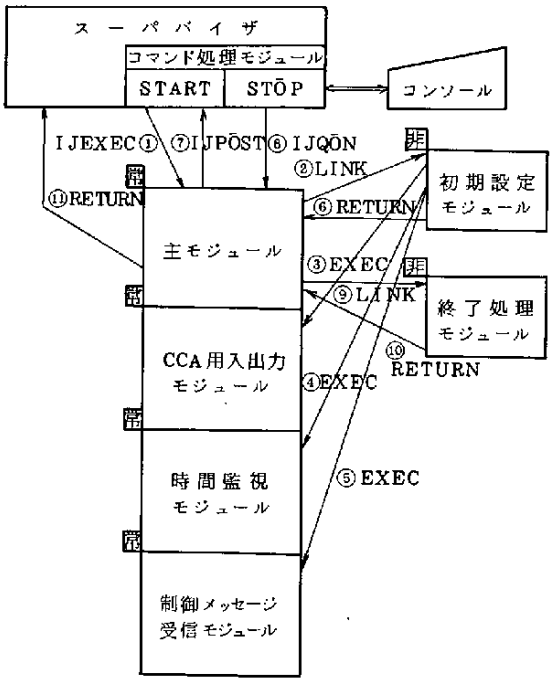


図4－28　F－NCPサブモニタ部の起動・停止

以下に，各モジュールの機能について述べる。

(1)　N．MAINモジュール

N.MAINモジュールは，M－Ⅵスーパバイザより NCP スタート指令によって起動され，次の機能を遂行する常駐モジュールである。

①　初期設定モジュール（ N. INTLZ ）を呼び出し，初期設定をする。初期設定完了後，スーパバイザにサブモニタの初期設定が終了したことを通知する。

②　NCPのストップ指令を受け付けるため，スーパバイザからのサブモニタ終了依頼を待つ。

③　スーパバイザからの終了依頼を受け，終了処理モジュール（ N. TERM ）を呼び出し，終了処理を行なう。

④　終了処理完了後，スーパバイザにサブモニタの終了処理の終了を通知する。これを受けてスーパバイザはサブモニタを消滅させる。

(2)　N. INTLZモジュール

N. INTLZ モジュールは，N.MAIN モジュールによって起動され，次の機能を遂行する非常駐モジュールである。

① サブモニタ・ジョブステップ作業域の固有直接制御表群の初期化をする。

② N.CCARTNモジュール，N.TMSPVRモジュール，N.CTLRCVモジュールを起動する。

③ 以上の処理を完了し，制御をN.MAINモジュールに戻し，消滅する。

(3) N.CCARTNモジュール

N.CCARTNモジュールは，N.INTLZモジュールによって起動され，次の機能を遂行する常駐モジュールである。

① 自系HOSTのネットワーク・サービスの開始を自系IMPにHOST-IMPプロトコルの初期状態処理手順に従って通知する。

② CCAからのアテンション割込みを受け付け，HOST-IMPプロトコルに従ってIMPからメッセージを受信し，Receive信号を返送する。

③ 受信したメッセージが制御メッセージの場合，そのメッセージを制御メッセージ受信モジュール(N.CTLRCV)に依頼(QON)する。

④ 受信したメッセージが一般メッセージの場合，すでにそのメッセージに対するリード要求が処理プログラムから発せられていれば，NREADマクロ命令の後処理サービス・モジュールを呼び起し(INTRPT)，処理プログラムにメッセージを渡し，まだリード要求が処理プログラムから発せられていなければ，各コネクション管理表中にあるリード待ち行列にメッセージを登録する。

⑤ 送信待ち行列に登録されているメッセージを取出し(QOFF)，HOST-IMPプロトコルに従ってメッセージをCCAを介してIMPに送り出す。メッセージが，目的地HOSTでdiscardされた場合は再送する。

⑥ HOSTとIMPが同時にWDIコマンドを発信した場合，アダプタ又はチャネルなどの所でコマンドの衝突がおこる。この場合，まずIMP側のメッセージを受信し，それからHOST側のメッセージを送信する。

(4) N.TMSPVR モジュール

N.TMSPVR モジュールは，N.INTLZ モジュールによって起動され，時間監視に関する次の機能を遂行する常駐モジュールである。

① 自系の処理プログラムからコネクションの接続依頼が出ていて，コネクションの接続が完了してないコネクションの時間監視を1分毎に行ない，規定された時間内(現バージョンでは30分)をオーバーした場合，S.OPNTUPサービス・モジュールを呼び出し，処理プログラムにコネクション確立失敗を通知する。

② この他の時間監視を行なう(現バージョンではやっていない)。

(5) N.CTLRCVモジュール

N.CTLRCVモジュールは，N.INTLZモジュールによって起動され，受信した制御メッセージに対応したサービス・モジュールへ振分ける常駐モジュールである。

次に，N.CTLRCV モジュールの処理手順を示す。

① QOFFマクロ命令を発し，制御メッセージの到着，すなわちN.CCARTNモジュールからの依頼（ＱＯＮ）を待つ。

② 制御メッセージの種類を調べ，各々の制御メッセージに１対１に対応したサービス・モジュールを呼び出し，サービスを依頼する。

③ サービス終了後，又，①に戻り，以上の処理をくり返す。

(6) N.TERMモジュール

N.TERM モジュールは，N.MAIN モジュールによって起動され，次の機能を遂行する非常駐モジュールである。

① 確立されていてまだ切断されていないコネクションを切断する。

② 自系HOSTのネットワーク・サービス終了を自系ＩＭＰにHOST－IMPプロトコルの終了手順に従って通知する。

③ サブモニタ・ジョブステップ作業域中の使用中の制御表ブロックを返却する。

C F－ＮＣＰサブモニタ部のプログラム構造

F－ＮＣＰサブモニタ部は，リエントラント構造であり，N.MAIN, N.CCARTN, N.TMSPVR, N.CTLRCV, N.INTLZ, N.TERMの６個のタスクから構成されている（図４－２９参照）。

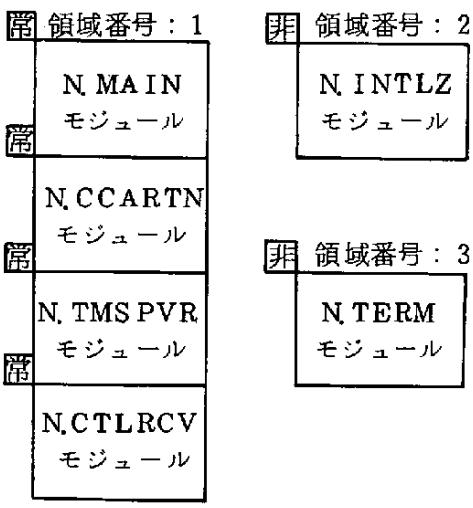


図４－２９ F－ＮＣＰサブモニタ部のプログラム構造

この内，N.MAIN，N.CCARTN，N.TMSPVR，N. CTLRCVは常駐タスクで，同一領域に属している。これらのタスクを同一領域に属させることによって，別々の領域に属させた場合にくらべ，領域は128語単位に割当てられるので，各領域の未使用領域（内部フラグメンテーション）がなくなり（図４－３０参照），又，領域管理表も１個でたりる。

N.INTLZとN.TERMは１回しか使用されないので，非常駐タスクとし，主記憶容量の節約を行っている。

表４－１に，各モジュールの大きさを示す。常駐部の大きさは約２K語，非常駐部の大きさは128語である。

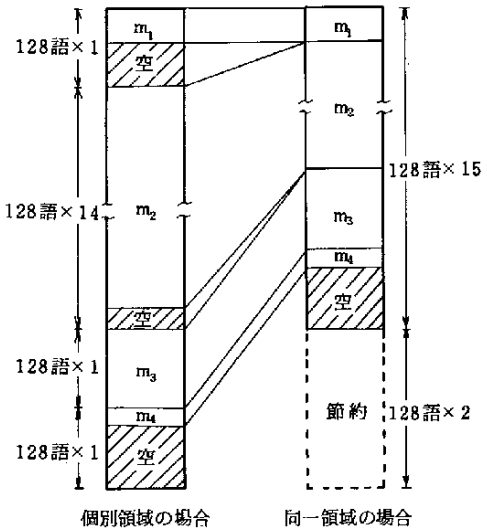


図４－３０　主記憶の有効利用

表４－１　モジュールの大きさ

モジュール名	大きさ（語）	常駐の有無
N.MAIN	52	○
N.CCARTN	1,746	○
N.TMSPVR	32	○
N.CTLRCV	128	○
N.INTLZ	92	×
N.TERM	124	×

4.2.6　BIPCAMをサポートするサービス・モジュール

ここでは，BIPCAMをサポートするために作成されたサービス・モジュールについて述べる（制御表に関しては，4.2.7にまとめて述べられているので随時参照して下さい）。

A　NOPEN マクロ命令をサポートするサービス・モジュール

NOPEN マクロ命令のサービスは，4つのサービス・モジュールの協同作業で遂行する。

(1)　N.NOPENサービス・モジュール

N.NOPENサービス・モジュールは処理プログラムからＴＲＰ　S.NOPEN で呼ばれ，パラメータ（ＮＣＢなど）のチェックを行ない，N.NOPEN1サービス・モジュールへ制御を受け渡す。

(2)　N.NOPEN1サービス・モジュール

N.NOPEN1サービス・モジュールは，ＮＣＢからＣＭＢ（Connection Management Black）を作成し，CNTCHN に登録し，＜ケース1＞ 他系 port がオープン済の場合は他系へＣＯＫ制御メッセージ，＜ケース2＞他系 port がオープンしてない場合はＲＦＣ制御メッセージを送信（IJQON）し，処理プログラムを quit し，サービスを一時中断する。

(3)　N.NOPENR サービス・モジュール

N.NOPENRサービス・モジュールは，前のN.NOPEN1サービス・モジュールが送信した制御メッセージ（ＣＯＫ又はＲＦＣ）に対するReceiveが到着した時に呼び出され（INTRPT），＜ケース1＞他系 port がオープン済の場合は処理プログラムをwake-upし，サービスを終了し，＜ケース2＞他系 portがオープン済でない場合，処理プログラムを quit 状態のままにしてサービスを一時中断する。

(4)　N.OPENXサービス・モジュール

①　ＲＦＣ受信サービス・モジュール（N.RFCRCV）からINTRPTマクロ命令で呼ばれた場合

ＲＦＣ制御メッセージのすれ違いであり，自系 port が入力側の場合，他系HOSTへＣＯＫ制御メッセージを送信（ＱＯＮ）する。

②　ＣＯＫ受信サービス・モジュール（N.COKRCV）からINTRPT マクロ命令で呼ばれた場合

処理プログラムをwake-up し，サービスを終了する。

以上の４つのサービス・モジュールの制御の関係を図４－３１に示す。

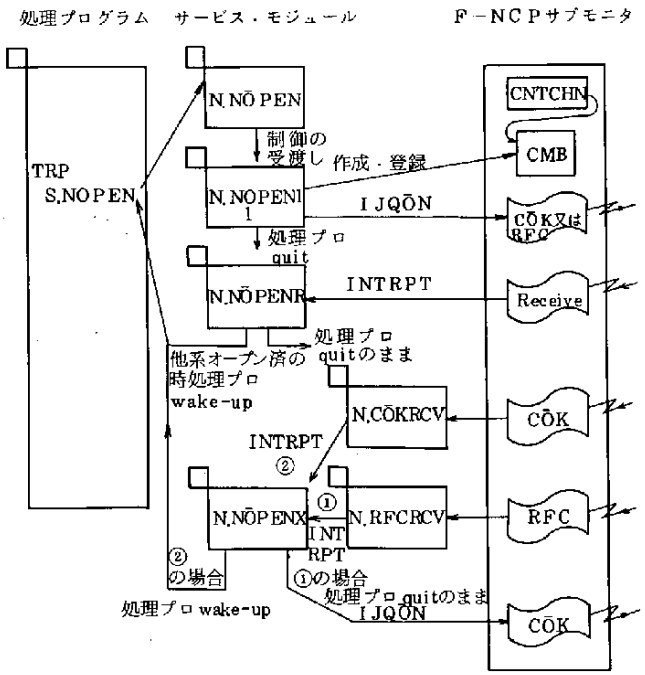


図４－３１　S.OPENサービス処理過程

B　NCLOSEマクロ命令をサポートするサービス・モジュール

NCLOSEマクロ命令のサービスは，３つのサービス・モジュールの協同作業で遂行する。

(1)　N.NCLOSEサービス・モジュール

N.NCLOSEサービス・モジュールは，処理プログラムからTRP　S.NCLOSEで呼ばれ，パラメータのチェックを行ない，他系HOSTへCLS制御メッセージを送信（IJQON）し，処理プログラムをquitし，サービスを一時中断する。

(2)　N.NCLOSRサービス・モジュール

N.NCLOSRサービス・モジュールは，前のN.NCLOSEサービス・モジュールが送信したCLS制御メッセージに対するReceiveが到着した時呼び出され（INTRPT），＜ケース１＞他系portがクローズ済の場合，コネクションの切断処理（コネクション番号の返却，CMBの削除・返却など）し，処理プログラムをwake-upし，サービスを終了し，＜ケース２＞他系がクローズ済でない場合は，処理プログラムをquit状態のままにしてサービスを一時中断する。

(3)　N.NCLOSXサービス・モジュール

N.NCLOSXサービス・モジュールは，他系よりCLS制御メッセージを受信した時，CLS受信サービス・モジュール（N.CLSRCV）からINTRPTマクロ命令で呼ばれ，他系port

をクローズ済とし，＜ケース１＞自系 port がクローズ済の場合，コネクションの切断処理し，処理プログラムを wake-up し，サービスを終了し，＜ケース２＞自系 port がクローズ済でない場合は，ただ単にフローを終了する。

以上の３つのサービス・モジュールの制御の関係を図４－３２に示す。

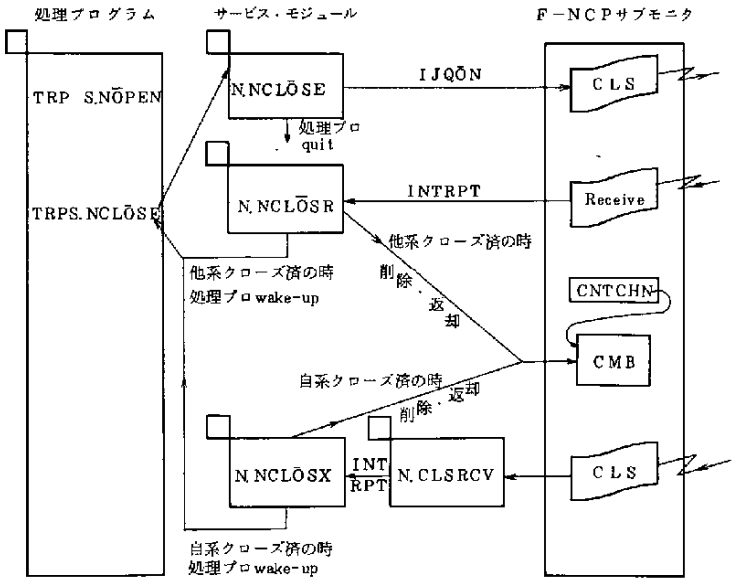


図４－３２　S.NCLOSEサービス処理過程

C　NREAD マクロ命令をサポートするサービス・モジュール

NREAD マクロ命令のサービスは，４つのサービス・モジュールの協同作業で遂行する。

(1)　N.NREADS サービス・モジュール

N.NREADS サービス・モジュールは，処理プログラムから TRP N.NREADS で呼ばれ，ＲＢ（Request Block）中のRBWKBIT のNREAD 表示ビットをオンにし，N.NRDWTC サービス・モジュールへ制御を受け渡す。

(2)　N.NRDWTCサービス・モジュール

N.NRDWTCサービス・モジュールは，パラメータのチェックを行ない，NREAD 表示を判別してN.NREAD2 サービス・モジュールへ制御を渡す。

(3)　N.NREAD2サービス・モジュール

N.NREAD2サービス・モジュールは，ＲＤＢ（Read Block）を作成し，＜ケース１＞すでに他系よりメッセージが到着している場合，ＣＭＢより当該メッセージ（ＳＧＢ）をはずし，ＲＤＢに結合し，N.NREADX サービス・モジュールへ制御を渡し，＜ケース２＞他系よりメッセージが到着してない場合，ＲＤＢをＣＭＢに結合し，処理プログラムを wake-up し，サービスを一時中断する。

(4) N.NREADX サービス・モジュール

① N.NREAD 2サービス・モジュールよりの制御の受渡しの場合

N.NREADX サービス・モジュールは，メッセージを処理プログラムの領域へ転送し，完了情報をセット（IJPOST）し，処理プログラムをwake-up し，サービスを終了する。

② INTRPTマクロ命令で呼ばれた場合

N. CCARTNモジュールは，メッセージを受信し，当該CMBにRDBが登録してある場合，RDBをはずし，RDBにそのメッセージ（SGB）を結合し，INTRPTマクロ命令でN.NREADXサービス・モジュールを呼ぶ。呼ばれたN.NREADX サービス・モジュールは①と同じ機能を遂行し，サービスを終了する。ただし，処理プログラムはすでにwake-upされているので，再度wake-up する必要はない。

以上の4つのサービス・モジュールの制御の関係を図4-33に示す。

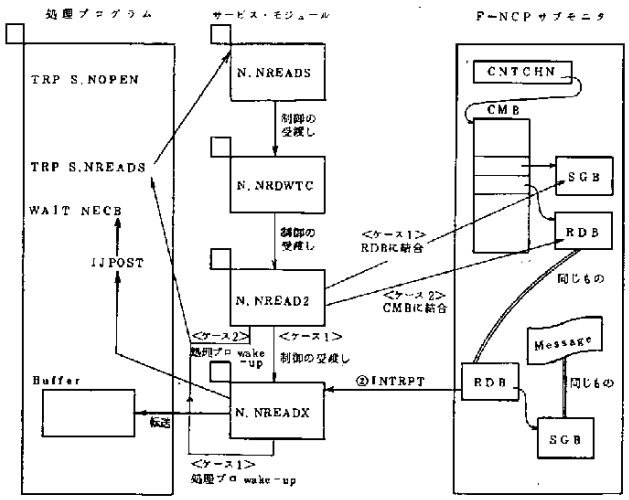


図4-33 S.NREADSサービス処理過程

D NWRITE マクロ命令をサポートするサービス・モジュール

NWRITE マクロ命令のサービスは，4つのサービス・モジュールの協同作業で遂行される。

(1) N.NWRITEサービス・モジュール

N.NWRITEサービス・モジュールは，処理プログラムからTRP S.NWRITES で呼ばれ，RB中のRBWKBITのNWRITE表示ビットをオンにし，N.NRDWTCサービス・モジュールに制御を受け渡す。

(2) N.NRDWTCサービス・モジュール

N.NRDWTCサービス・モジュールは，パラメータのチェックを行ない，NWRITE 表示

を判別してN.NWRIT2サービス・モジュールへ制御を受け渡す。

(3) N.NWRIT2サービス・モジュール

N.NWRIT2サービス・モジュールは，処理プログラム内のデータ（メッセージ）をSNDB（Send Block）へコピーし，他系HOSTへ送信（IJQON）し，処理プログラムをwake-upし，サービスを一時中断する。

(4) N.NWRITRサービス・モジュール

N.NWRITRサービス・モジュールは，送信したメッセージが他系で受信され，Receiveが自系に到着した時，N.CCARTNモジュールによってINTRPTマクロ命令で呼ばれ，Receive情報を処理プログラムのNECBに通知（IJPOST）し，サービスを終了する。ただし，処理プログラムはすでにwake-upされているので，再度wake-upする必要がない。

以上の4つのサービス・モジュールの制御の関係を図4－34に示す。

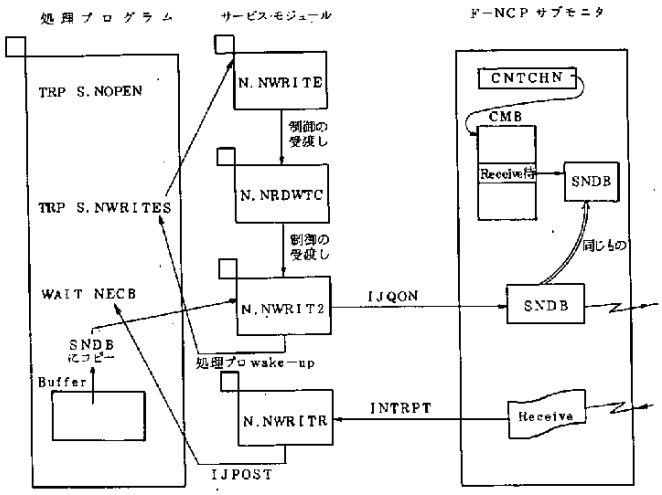


図4－34　S.NWRITESサービス処理過程

E　LISTEN マクロ命令をサポートするサービス・モジュール

LISTEN マクロ命令のサービスは，2つのサービス・モジュールの協同作業で遂行される。

(1) N.LISTENサービス・モジュール

N.LISTENサービス・モジュールは，処理プログラムからTRP S.LISTENで呼ばれ，パラメータのチェックを行ない，LSNB（Listen Block）を作成し，LSNQTに登録し，＜ケース1＞他系よりRFC制御メッセージが到着している場合は，OPNQTより当該OPNB（Open Block）をはずし，LSNBに結合し，N.LISTNXサービス・モジュールへ制御を受け渡し，＜ケース2＞他系よりRFC制御メッセージが到着してない場合は，処

理プログラムをwake-upし，サービスを一時中断する。

(2) N.LISTNX サービス・モジュール

N.LISTNXサービス・モジュールは，OPNB中の（他系の）my-port-idをLISTENマクロ命令で指定した（自系の）your-port-id番地にセットし，ECBに対してポスト（IJPOST）し，サービスを終了する。サービスの終了の仕方としては，＜ケース1＞N.LISTENより制御の受渡しの場合は，処理プログラムをwake-upし，＜ケース2＞他系よりRFC制御メッセージが到着して，RFC受信モジュール（N.RFCRCV）よりINTRPTマクロ命令で呼ばれた場合，処理プログラムはすでにwake-up されているので再度wake-up しない。

以上の2つのサービス・モジュールの制御の関係を図4－35に示す。

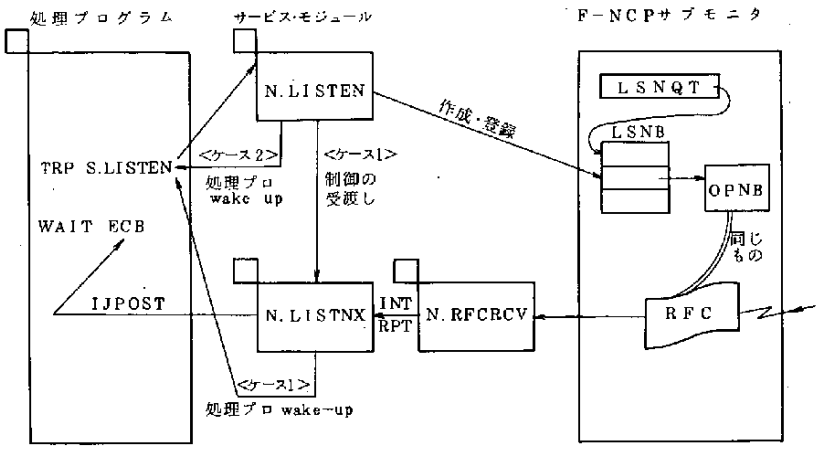


図4－35　S.LISTENサービス処理過程

F　SETECBID マクロ命令をサポートするサービス・モジュール

SETECBID マクロ命令のサービスは，1つのサービス・モジュールで遂行される。

(1) N.SETEID サービス・モジュール

N.SETEID サービス・モジュールは，3種類の方法で呼ばれる。

① 処理プログラムからTRP S.SETECBID で呼ばれた場合は，パラメータのチェックを行ない，EIDB(ECB-ID Block)を作成し，＜ケース1＞他系よりPOST制御メッセージが到着している場合，PSTQTより該当するPSTB(Post Block)をはずし，処理プログラムにPOST情報を通知し，処理プログラムをwake-upし，サービスを終了し，＜ケース2＞他系よりPOST制御メッセージが到着してない場合，EIDBをEIDQTに登録し，処理プログラムをwake-upし，サービスを一時中断する。

② POST制御メッセージを受信し，制御メッセージ受信モジュール（N.CTLRCV）が T

RPで呼び出した場合，PSTBを作成し，＜ケース1＞すでに該当するEIDBがEIDQTに登録されている場合，当該PSTBをはずし，INTRPTマクロ命令で再びN.SETEIDサービス・モジュールを呼び出し，制御メッセージ受信モジュールをwake-upし，サービスを終了し，＜ケース2＞該当するEIDBがEIDQTに登録されていない場合，PSTBをPSTQTに登録し，制御メッセージ受信モジュールをwake-upする。

③ N.SETEIDサービス・モジュールからINTRPTマクロ命令で呼ばれた場合，PSTB中のPOST情報を処理プログラムに通知し，処理プログラムをwake-upせずにサービスを終了する。

以上のサービス・モジュールの制御を図4－36に示す。

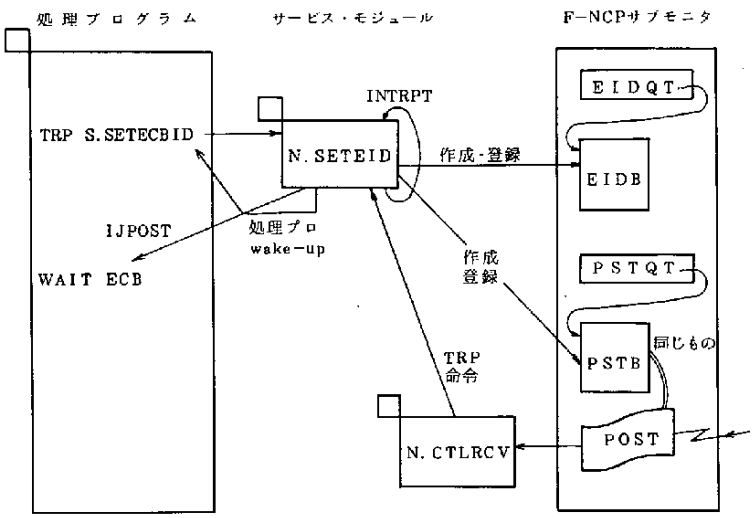


図4－36　S.SETECBIDサービス処理過程

G　NPOSTマクロ命令をサポートするサービス・モジュール

NPOSTマクロ命令のサービスは，1つのサービス・モジュールで遂行される。

(1) N.NPOSTサービス・モジュール

N.NPOSTサービス・モジュールは，2種類の方法で呼ばれる。

① 処理プログラムからTRP S.NPOSTで呼ばれた場合，パラメータのチェックを行ない，SNDB上にPOST制御メッセージを作成し，他系に送信（IJQON）し，処理プログラムをquitし，サービスを一時中断する。

② 他系に送信したPOST制御メッセージに対するReceiveが自系に到着した時，N.CCARTNモジュールによりINTRPTマクロ命令で呼ばれ，Receive情報を処理プログラムの完了情報エリアにセットし，処理プログラムをwake-upし，サービスを終了する。

以上のサービス・モジュールの制御を図4－37に示す。

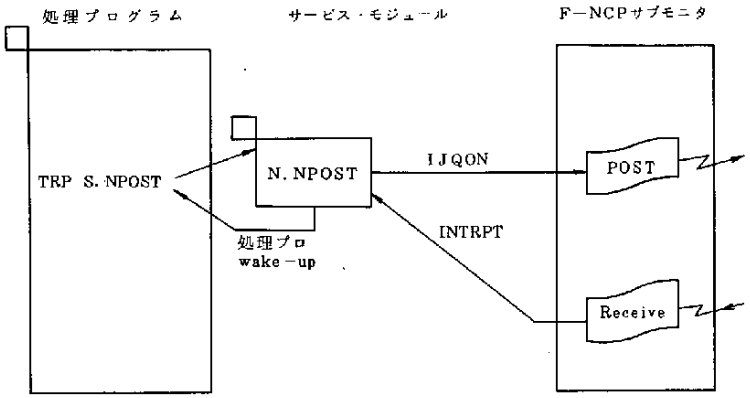


図４－37　S．NPOSTサービス処理過程

H　サービス・モジュールの属性

BIPCAMを構成するサービス・モジュールの属性を表4-2にまとめて示す。

表４－２　サービス・モジュールの属性一覧表（その１）

モジュール名	モジュールの種類	モジュールの大きさ（語）	SF番号	DS番号（8進）	RBWKの大きさ（語）
N.NOPEN	ITT	142	1000	604000	16
N.NOPEN1	ITT	508	－	604001	16
N.NOPENR	ITT	104	－	604002	16
N.NOPENX	ITT	304	－	604003	16
N.NCLOSE	ITT	248	1001	605000	16
N.NCLOSR	ITT	104	－	605001	16
N.NCLOSX	ITT	100	－	605002	16
N.NREADS	IRT	12	1002	606000	16
N.NWRITE	IRT	11	1003	606001	16
N.NRDWTC	IRT	96	－	607000	16
N.NREAD2	IRT	140	－	607001	16
N.NREADX	IRT	152	－	607002	16
N.NWRIT2	IRT	228	－	607003	16
N.NWRITR	IRT	54	－	607004	16
N.LISTEN	ITT	182	1004	610000	16
N.LISTNX	ITT	66	－	610001	16

表４－２　サービス・モジュールの属性一覧表（その２）

モジュール名	モジュールの種類	モジュールの大きさ（語）	ＳＦ番号	ＤＳ番号（８進）	RBWKの大きさ（語）
N.NPOST	IRT	146	1005	611000	16
N.SETEID	IRT	276	1006	612000	16

4.2.7 制御表

A 概要

サブモニタ・ジョブステップ作業域は，直接参照部分（640語）と間接参照部分（15,104語）から成り，全体で15,744語（128語×123）である。直接参照部分は共通直接制御表（161語）と固有直接制御表（479語）から構成されている（図４－３８参照）。

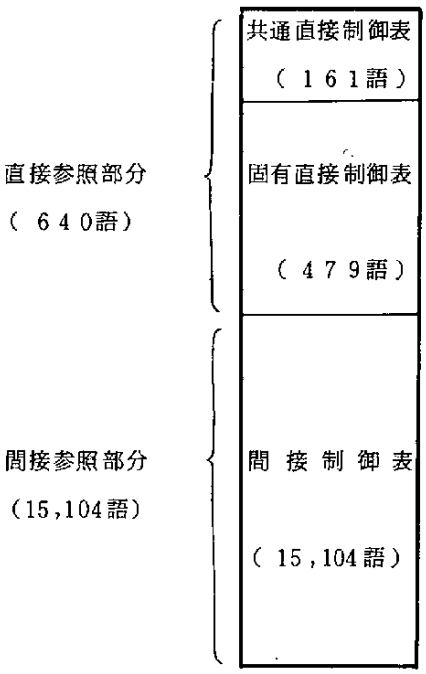


図４－３８　サブモニタ・ジョブステップ作業域の構造

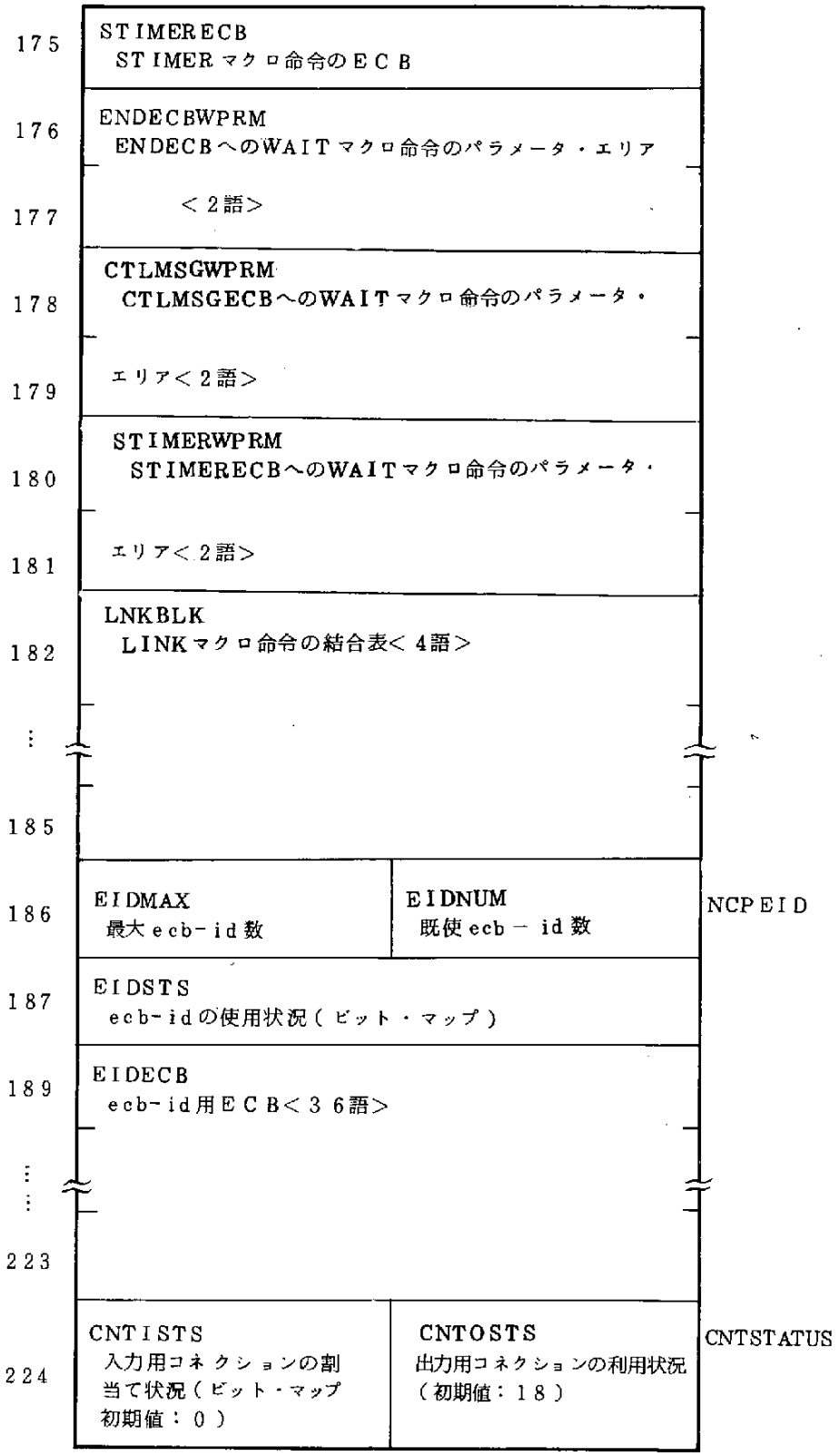
共通直接制御表はM－Ⅵのサブモニタすべてに共通であるので，ここでは説明を省略する。

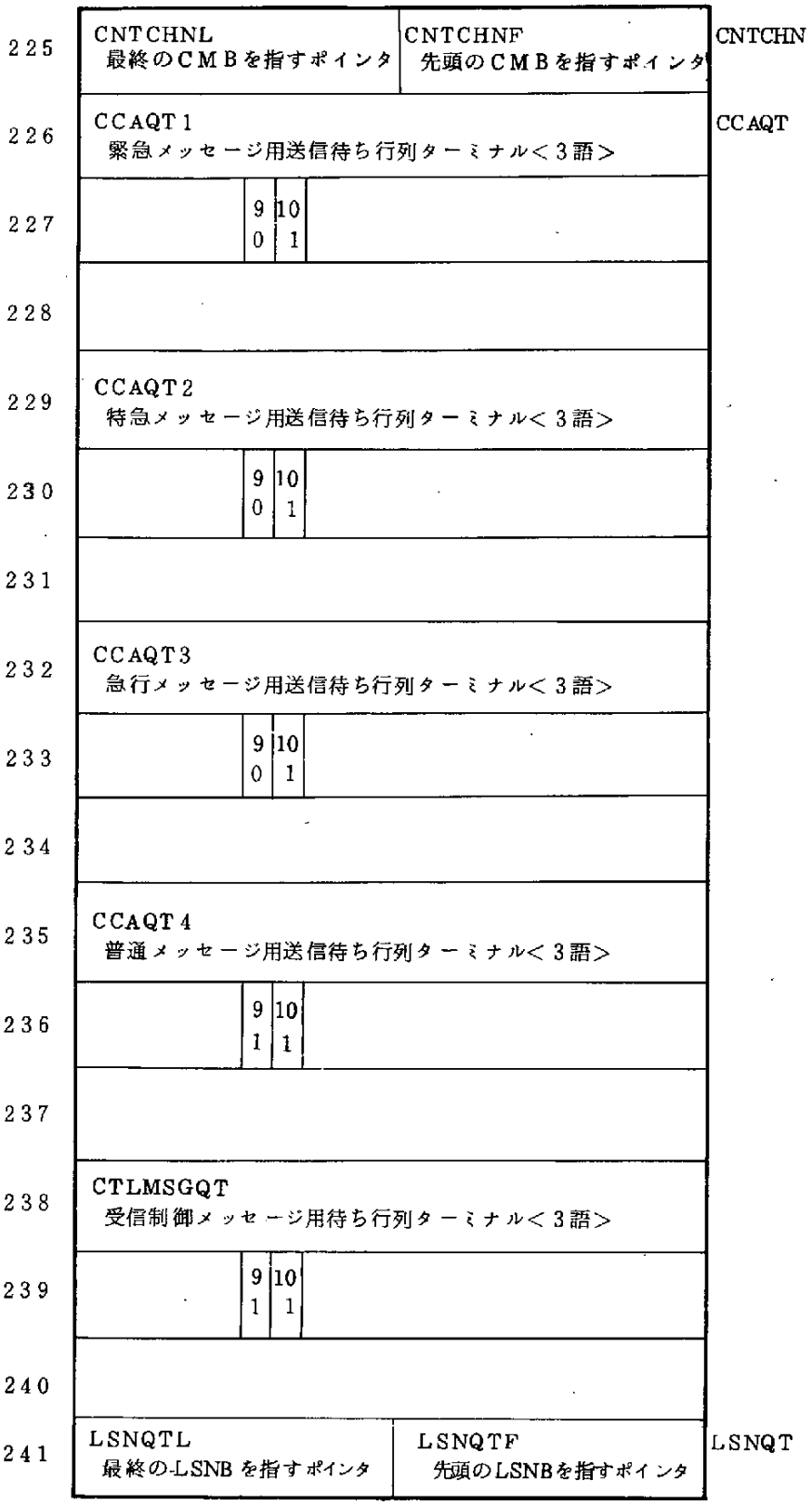
固有直接制御表には特定の個数だけ存在し，大きさの変化しない制御表をとり，間接参照部分には必要とする個数や大きさが変化する制御表をとる。

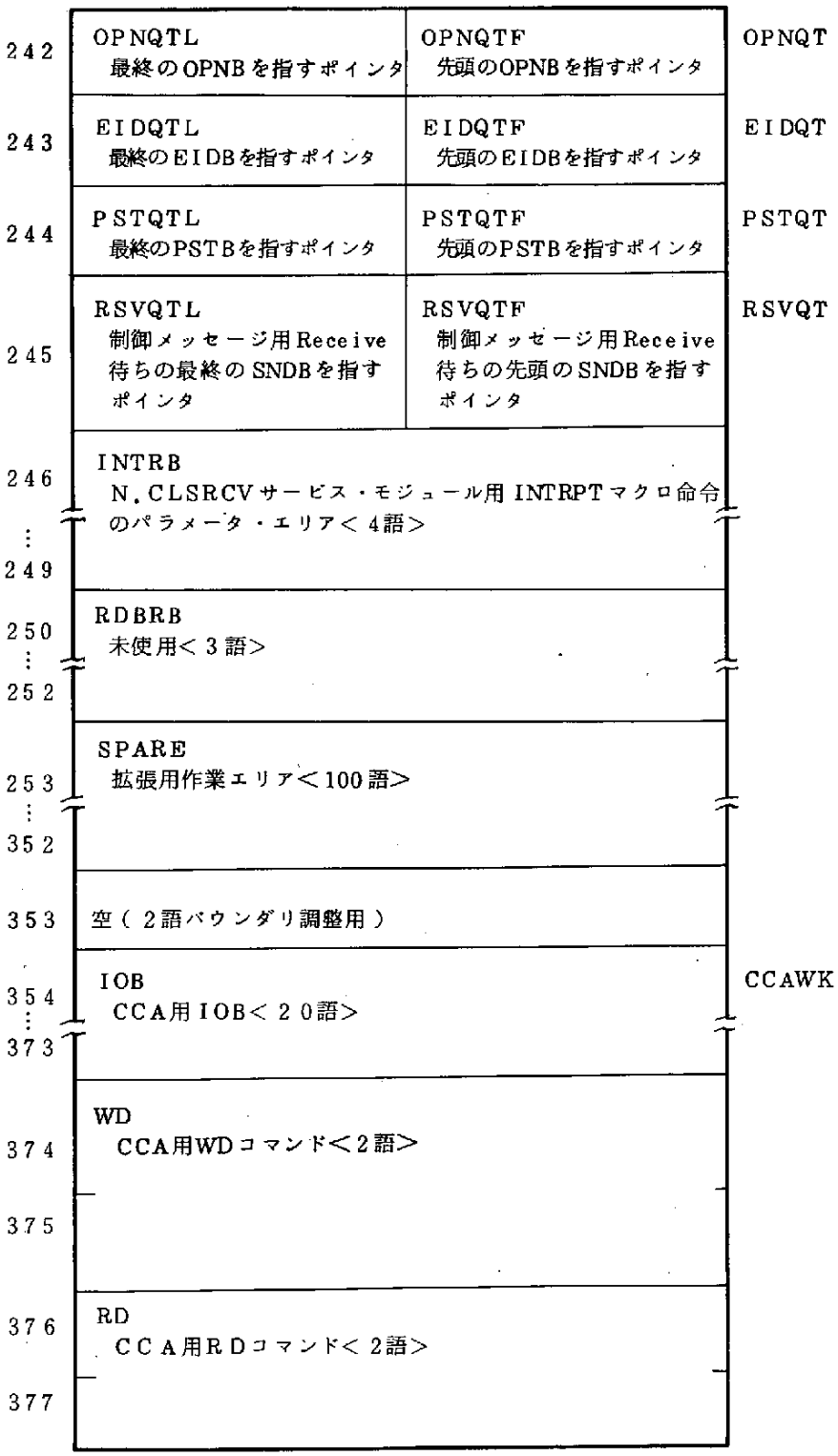
B　固有直接制御表

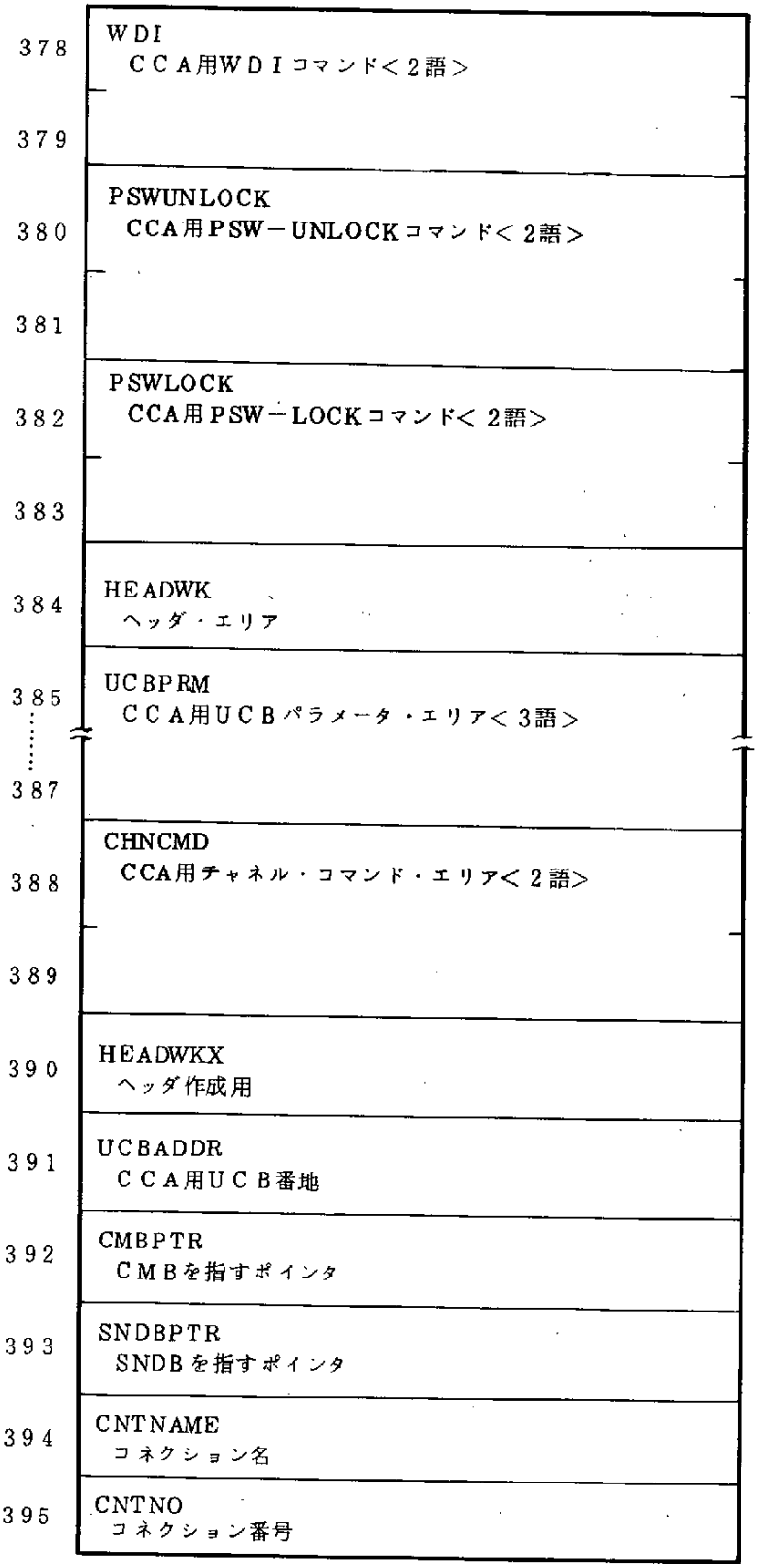
図４－３９に固有直接制御表のフォーマットを示す。固有直接制御表の初期値は，サブモニタの初期設定時にN.INTLZモジュールによってセットされる。

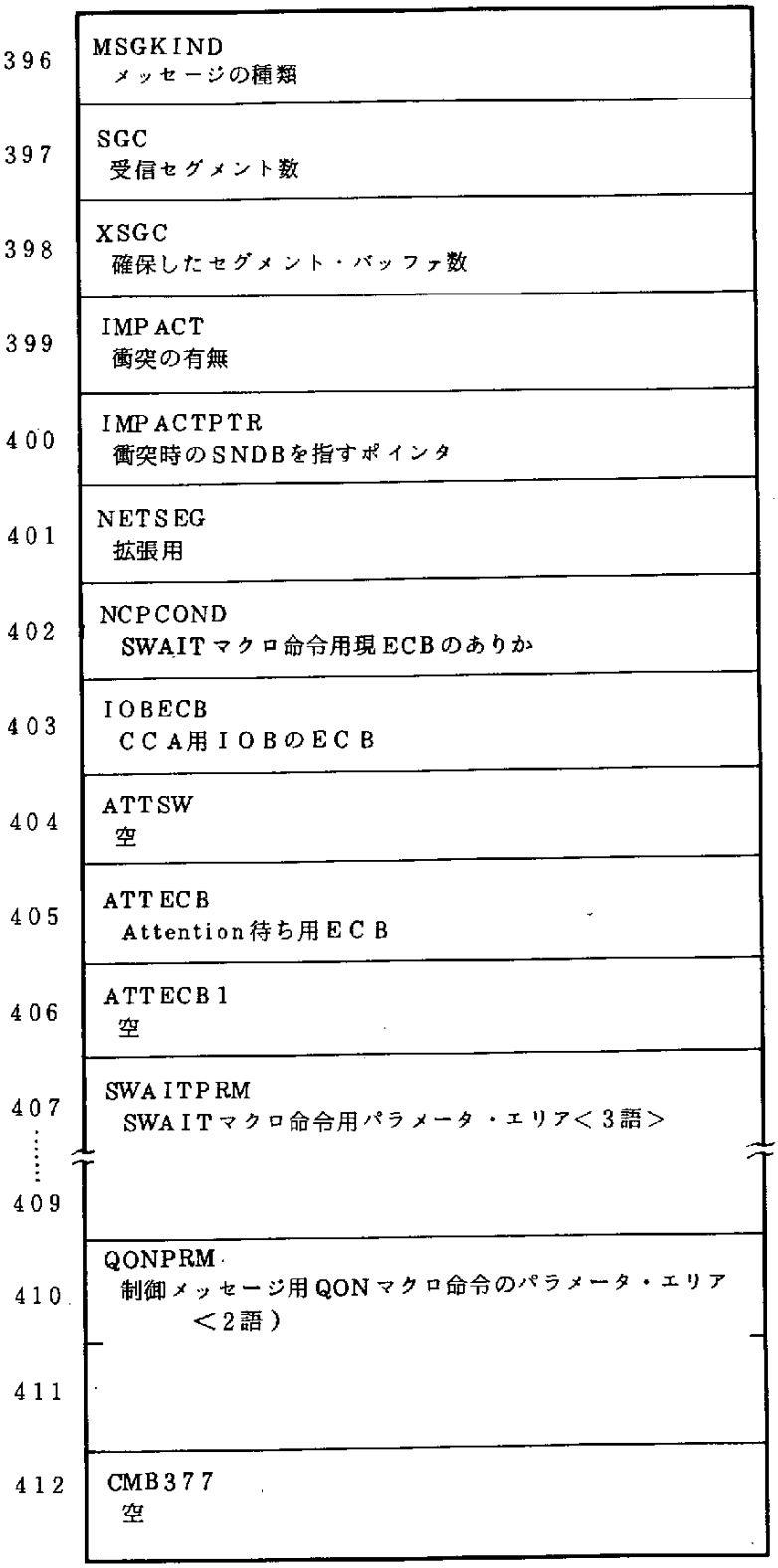
161	NCPHOSTSTS F-NCP サブモニタの状態表示	NCB
162	NCPIMPSTS 自系 IMP の状態表示	
163	NCPVL F-NCP サブモニタのバージョン・レベル	
164	NCPASW ASWの内容	
165	NCPMAXINIT 最大多重度 / NCPCRTINIT 現在の多重度	
166	NCPABTCODE サブモニタ・アボート時の内部コード	
167	NCPABTINF サブモニタ・アボート時の通知情報	
168	NCPHOSTNO 自系HOST番号 （006_8） / NCPSGSIZE セグメント長 （初期値：144）	
169	ENDECB SMBQTへのQOFFマクロ命令のECB	NCPECB
170	ENDPA SMBQTから受け取るパラメータの番地	
171	CCAECB CCAQTへのQOFFマクロ命令のECB	
172	CCAPA CCAQTから受け取るパラメータの番地	
173	CTLMSGECB CTLMSGQTへのQOFFマクロ命令のECB	
174	CTLMSGPA CTLMSGQTから受け取るパラメータの番地	

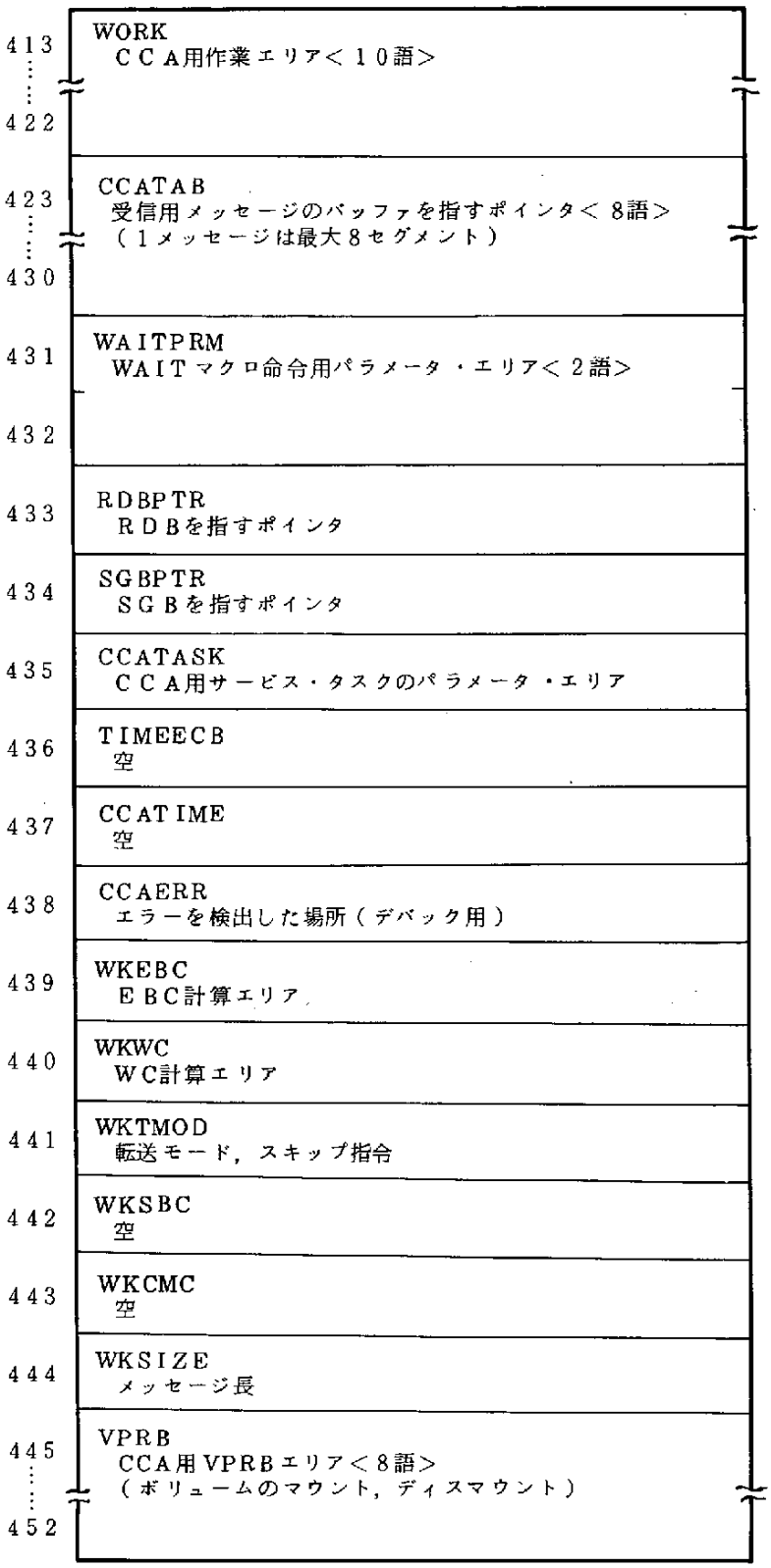

175
STIMERECB
STIMERマクロ命令のECB
176
ENDECBWPRM
ENDECBへのWAITマクロ命令のパラメータ・エリア
177
<2語>
178
CTLMSGWPRM
CTLMSGECBへのWAITマクロ命令のパラメータ・
179
エリア<2語>
180
STIMERWPRM
STIMERECBへのWAITマクロ命令のパラメータ・
181
エリア<2語>
182
LNKBLK
LINKマクロ命令の結合表<4語>
185
186
EIDMAX
最大ecb-id数
EIDNUM
既使ecb－id数
NCPEID
187
EIDSTS
ecb-idの使用状況(ビット・マップ)
189
EIDECB
ecb-id用ECB<36語>
223
224
CNTISTS
入力用コネクションの割当て状況(ビット・マップ
初期値：0)
CNTOSTS
出力用コネクションの利用状況
(初期値：18)
CNTSTATUS


225
CNTCHNL
最終のCMBを指すポインタ
CNTCHNF
先頭のCMBを指すポインタ
CNTCHN
226
CCAQT1
緊急メッセージ用送信待ち行列ターミナル<3語>
CCAQT
227
9 0
10 1
228
229
CCAQT2
特急メッセージ用送信待ち行列ターミナル<3語>
230
9 0
10 1
231
232
CCAQT3
急行メッセージ用送信待ち行列ターミナル<3語>
233
9 0
10 1
234
235
CCAQT4
普通メッセージ用送信待ち行列ターミナル<3語>
236
9 1
10 1
237
238
CTLMSGQT
受信制御メッセージ用待ち行列ターミナル<3語>
239
9 1
10 1
240
241
LSNQTL
最終のLSNBを指すポインタ
LSNQTF
先頭のLSNBを指すポインタ
LSNQT


242
OPNQTL
最終のOPNBを指すポインタ
OPNQTF
先頭のOPNBを指すポインタ
OPNQT
243
EIDQTL
最終のEIDBを指すポインタ
EIDQTF
先頭のEIDBを指すポインタ
EIDQT
244
PSTQTL
最終のPSTBを指すポインタ
PSTQTF
先頭のPSTBを指すポインタ
PSTQT
245
RSVQTL
制御メッセージ用Receive待ちの最終のSNDBを指すポインタ
RSVQTF
制御メッセージ用Receive待ちの先頭のSNDBを指すポインタ
RSVQT
246
INTRB
N.CLSRCVサービス・モジュール用INTRPTマクロ命令のパラメータ・エリア＜4語＞
249
250
RDBRB
未使用＜3語＞
252
253
SPARE
拡張用作業エリア＜100語＞
352
353
空（2語バウンダリ調整用）
354
IOB
CCA用IOB＜20語＞
CCAWK
373
374
WD
CCA用WDコマンド＜2語＞
375
376
RD
CCA用RDコマンド＜2語＞
377


378
WDI
CCA用WDIコマンド<2語>
379
380
PSWUNLOCK
CCA用PSW－UNLOCKコマンド<2語>
381
382
PSWLOCK
CCA用PSW－LOCKコマンド<2語>
383
384
HEADWK
ヘッダ・エリア
385
UCBPRM
CCA用UCBパラメータ・エリア<3語>
387
388
CHNCMD
CCA用チャネル・コマンド・エリア<2語>
389
390
HEADWKX
ヘッダ作成用
391
UCBADDR
CCA用UCB番地
392
CMBPTR
CMBを指すポインタ
393
SNDBPTR
SNDBを指すポインタ
394
CNTNAME
コネクション名
395
CNTNO
コネクション番号


396
MSGKIND
メッセージの種類
397
SGC
受信セグメント数
398
XSGC
確保したセグメント・バッファ数
399
IMPACT
衝突の有無
400
IMPACTPTR
衝突時のSNDBを指すポインタ
401
NETSEG
拡張用
402
NCPCOND
SWAITマクロ命令用現ECBのありか
403
IOBECB
CCA用IOBのECB
404
ATTSW
空
405
ATTECB
Attention待ち用ECB
406
ATTECB1
空
407
SWAITPRM
SWAITマクロ命令用パラメータ・エリア<3語>
409
410
QONPRM
制御メッセージ用QONマクロ命令のパラメータ・エリア
<2語)
411
412
CMB377
空


413
422
WORK
CCA用作業エリア＜10語＞
423
430
CCATAB
受信用メッセージのバッファを指すポインタ＜8語＞
（1メッセージは最大8セグメント）
431
432
WAITPRM
WAITマクロ命令用パラメータ・エリア＜2語＞
433
RDBPTR
RDBを指すポインタ
434
SGBPTR
SGBを指すポインタ
435
CCATASK
CCA用サービス・タスクのパラメータ・エリア
436
TIMEECB
空
437
CCATIME
空
438
CCAERR
エラーを検出した場所（デバック用）
439
WKEBC
EBC計算エリア
440
WKWC
WC計算エリア
441
WKTMOD
転送モード，スキップ指令
442
WKSBC
空
443
WKCMC
空
444
WKSIZE
メッセージ長
445
452
VPRB
CCA用VPRBエリア＜8語＞
（ボリュームのマウント，ディスマウント）

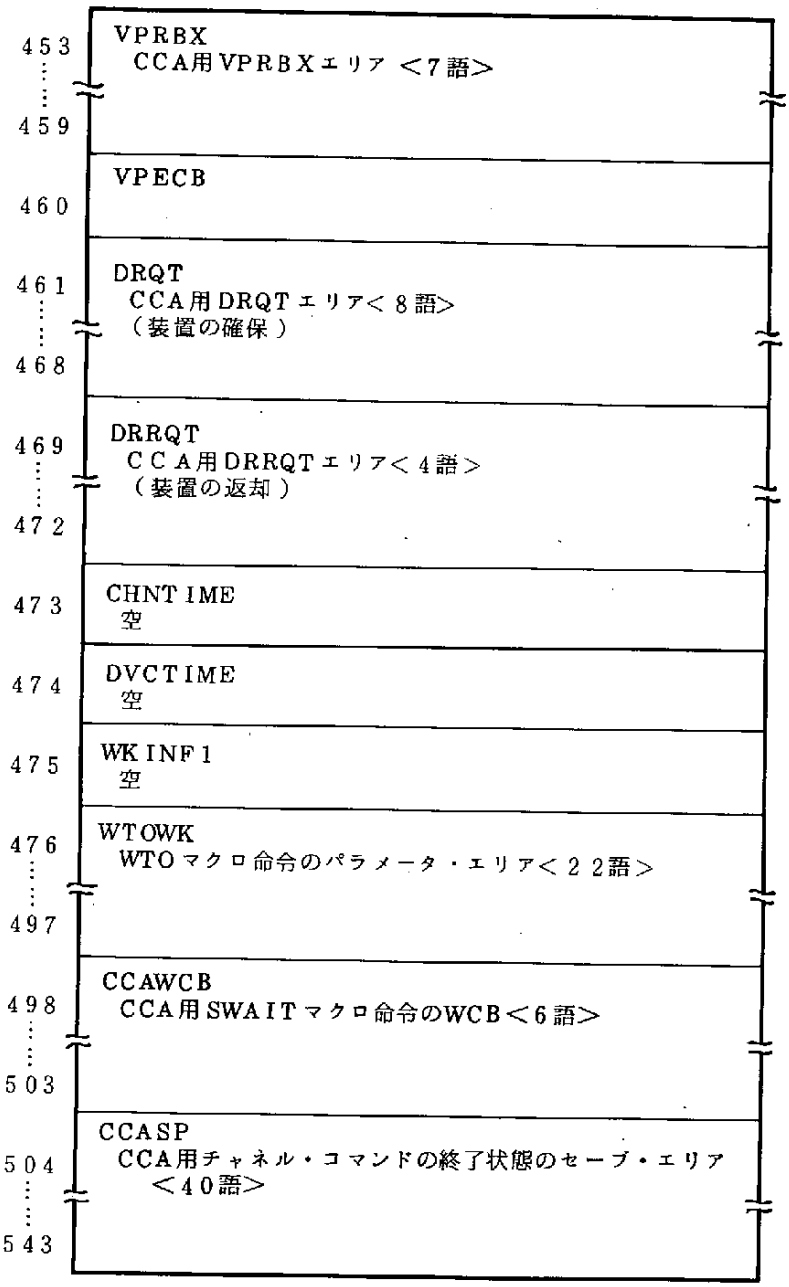


図4－39　固有直接制御表のフォーマット

C　間接制御表

以下に間接参照部分に動的に割り付けられる間接制御表を示す。

(1)　CMB

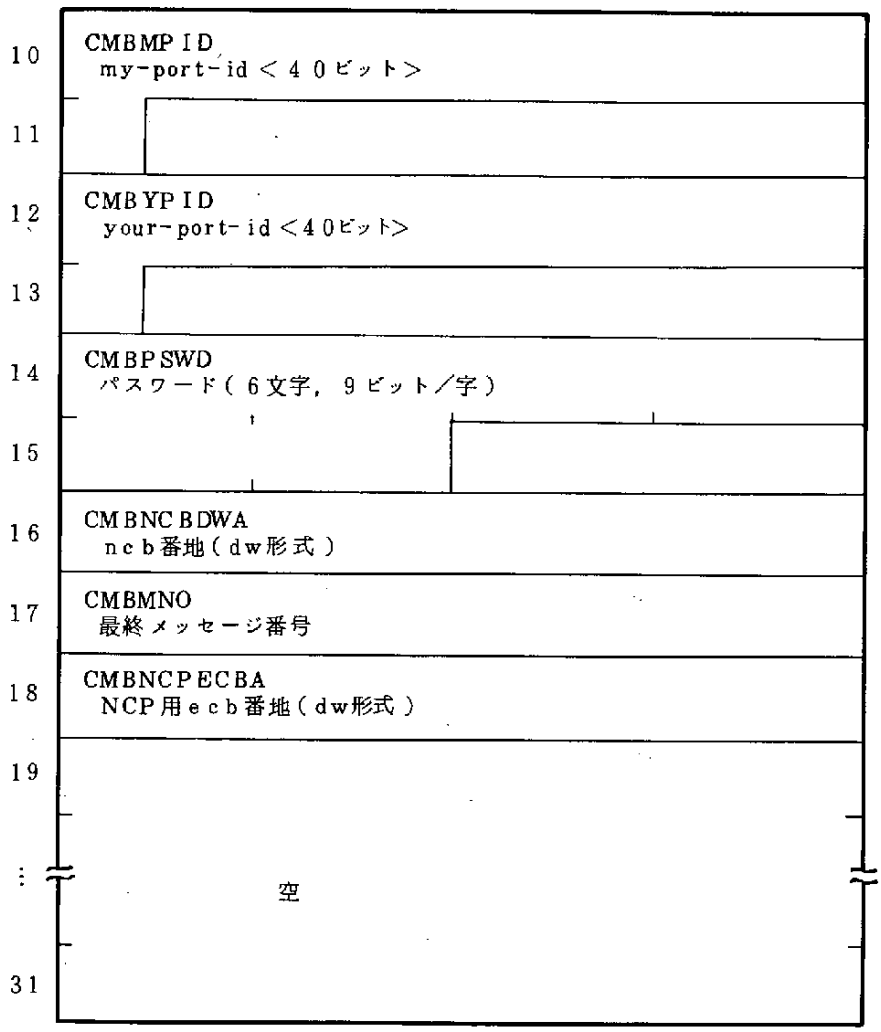
CMB（Connection Management Block）は，コネクションを管理するために間接参照部分に動的に割り付けられる32語の制御表である。

処理プログラムからNOPEN マクロ命令が発せられた時，N.OPENサービス・モジュールがGETB5 マクロ命令によって間接参照部分にCMBを割り付ける。

処理プログラムからNCLOSEマクロ命令が発せられ，かつ他系よりCLS制御メッセージを受信した時，N.CLOSEX サービス・モジュールがFREEB5 マクロ命令によって間接参照部分にCMBを返却する。

図4-40～図4-43にCMBのフォーマットを示す。

0	CMBTITLE 制御表タイトル（452_8）		
1	CMBPRE 前のCMBを指すポインタ		CMBNXT 次のCMBを指すポインタ
2	CMBCNM コネクション名		空
3	CMBMNUM 先読みメッセージの個数／CCAQTへの登録回数		CMBMFST 最初の先読みメッセージのSGBを指すポインタ
4	CMBIONUM（注1） 先出し入出力要求の個数		CMBIOFST（注1） 最初の先出し入出力要求ブロック（RDB）を指すポインタ
5	CMBRNUM（注2） Receive待ちメッセージの個数		CMBRFST（注2） 最初のReceive待ちメッセージ（SNDB）を指すポインタ
6	CMBJSNO ジョブステップ番号	CMBTASK タスク番号	CMBSMNO タイム・アウト後に呼び出すサービス・モジュール番号
7	CMBTLMT 他系オープン待ち残り時間（初期値：30分）		
8	CMBTMOD 転送モード		CMBMSIZE 最大メッセージ長
9	CMBMSTS 自系のコネクション状態		CMBYSTS 他系のコネクション状態



（注１）　CEASE中の場合，送信待ちメッセージ・ターミナルにも使う。

（注２）　複数セグメント・メッセージの場合は，バッファ予約待ちターミナルにも使う。

図４－４０　ＣＭＢのフォーマット

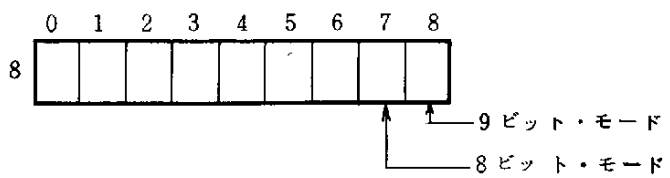


図４－４１　　CMBTMODの詳細

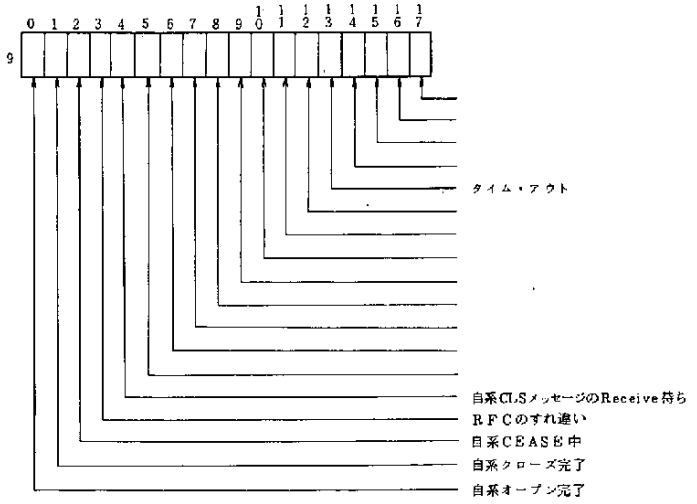


図４－４２　CMBMSTSの詳細

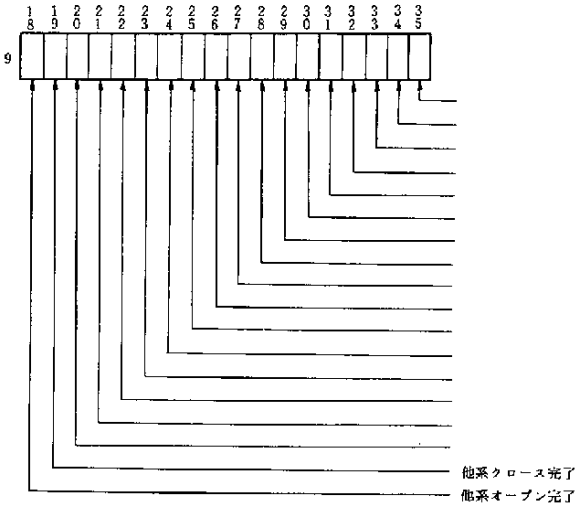


図４－４３　CMBYSTSの詳細

CMBは，図４－44に示すように直接参照部分にあるCNTCHNに登録され管理される。CNTCHNの右半分CNTCHNFは先頭ＣＭＢを指し，CNTCHNの左半分CNTCHNＬは最終CMBを指す。

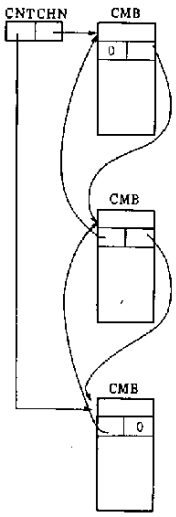


図４－４４　ＣＭＢの管理

(2)　ＳＮＤＢ

ＳＮＤＢ（Send Block）は，送信メッセージをバッファしておく間接参照部分に動的に割り付けられる可変長の制御表である。

処理プログラムからのサービス依頼によって他系にメッセージを送らなければならなくなった時，各サービスに対応したサービス・モジュールが，まずGETB４マクロ命令によって処理プログラムのジョブステップ作業域の間接参照部分にSNDB（これを元SNDBと呼ぶ）を割り付け，次にＩJQONマクロ命令によってスーパバイザのジョブステップ作業域の間接参照部分にSNDB（これを前SNDBと呼ぶ）を割り付け，前SNDBに元SNDBの内容をコピーし，元ＳＮＤを処理プログラムのジョブステップ作業域の間接参照部分に返却し，さらに，サブモニタのジョブステップ作業域の間接参照部分にSNDB（これを現SNDBと呼ぶ）を割り付け，現SNDBに前SNDBの内容をコピーし，前SNDBをスーパバイザのジョブステップ作業域の間接参照部分に返却し，新SNDBをサブモニタのジョブステップ作業域の直接参照部分にあるＣＣＡ用待ち行列ターミナル・リストCCAQTに登録される（図４－４５参照）。

N. CCARTNモジュールがQOFFマクロ命令によってSNDBを取り出し，他系にメッセージを送信し，そのメッセージに対するReceiveを待つため，制御メッセージの場合は制御メッセージReceive待ちターミナル（RSVQT）に登録され，Receive受信後FREEB４マクロ命令によって間接参照部分にSNDBを返却する。

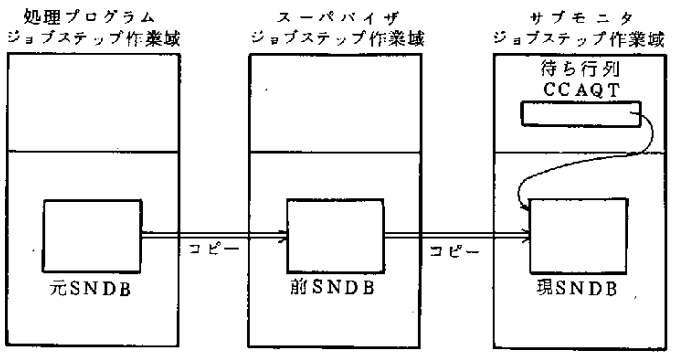


図4－45　IJQONマクロ命令の動作

SNDBは，最大512語の可変長制御表であり，固定部と可変部から構成されている（図4－46参照）。固定部にはメッセージを送信するのに必修の制御情報が入っており（図4－47～図4－49参照），可変長部にはメッセージの本体（テキスト）が入っている（図4－50参照）。

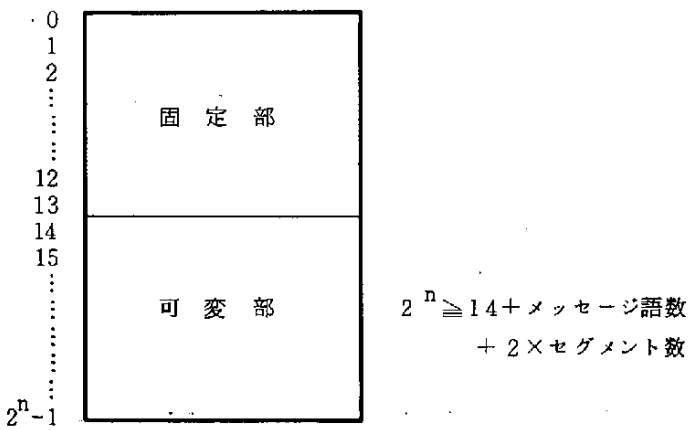


図4－46　SNDBのフォーマット

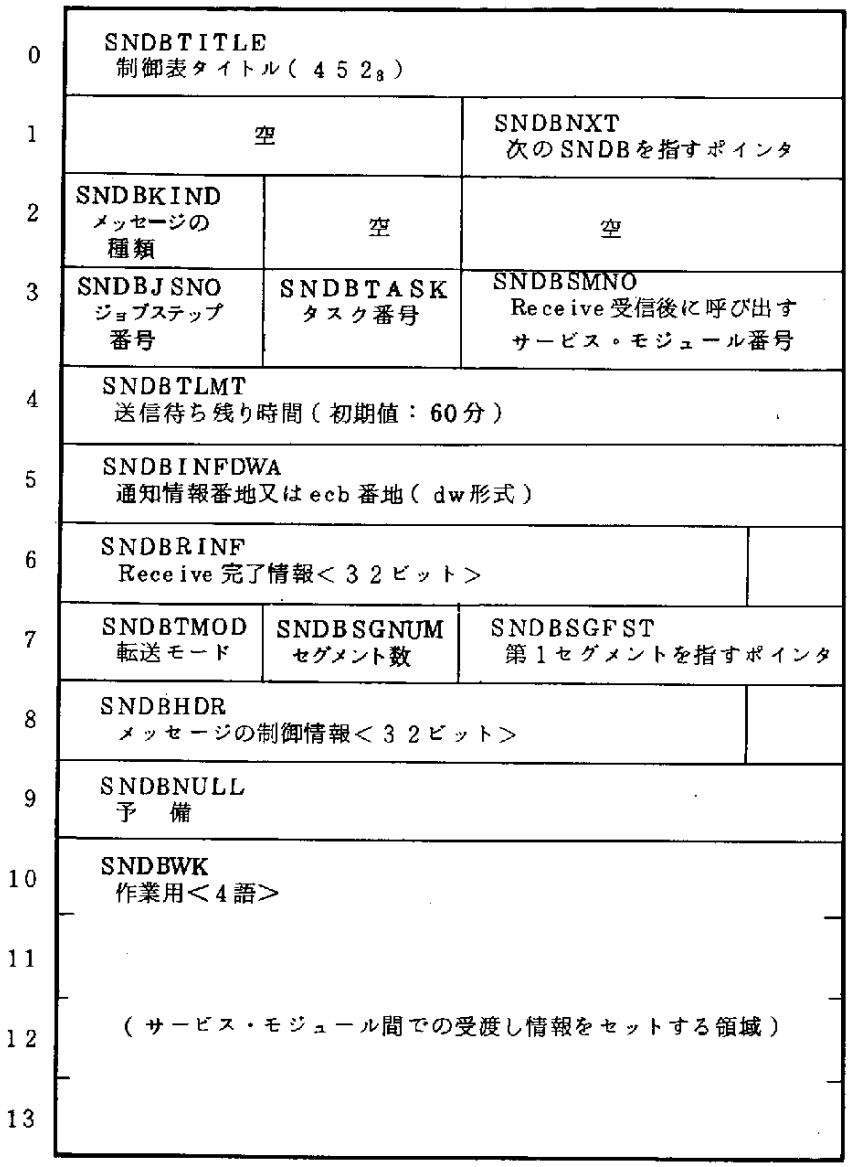


図4-47　SNDB固定部のフォーマット

コード	内　　容
1	HOST-IMP制御メッセージ
2	Receiveメッセージ
3	制御コマンド・メッセージ
4	一般メッセージ

図4-48　SNDBKINDの詳細

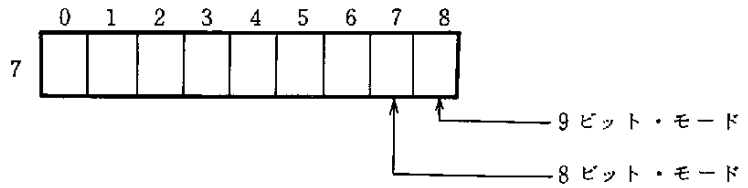


図 4－49　SNDBTMODの詳細

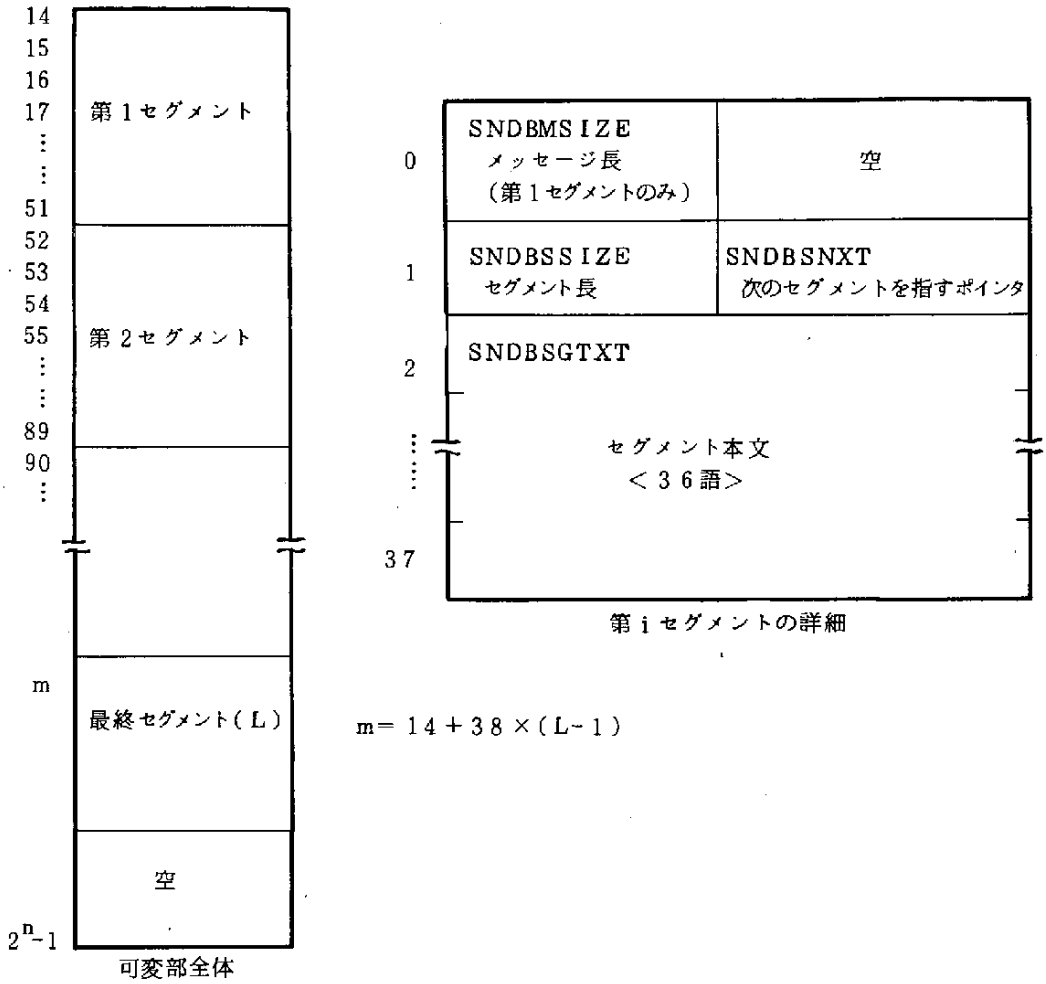


図4－50　SNDB可変部のフォーマット

図4-51にメッセージ長が300バイト，転送モードが9ビットの一般メッセージの場合のSNDBを示す。

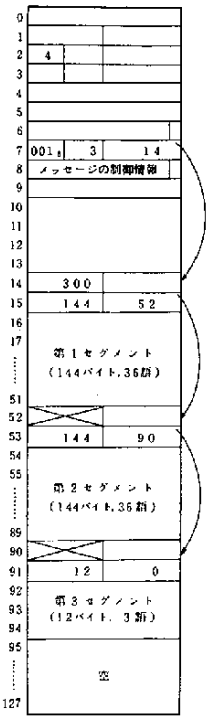


図4－51　SNDBの実例

（一般メッセージ，メッセージ長：300バイト，転送モード：9ビット）

(3)　SGB

SGB（Segment Block）は，他系より受信したセグメントをバッファしておく間接参照部分に動的に割り付けられる64語の制御表である。

N. CCARTNモジュールは，IMPよりのアテンションを検知し，制御情報を読み込み，つづいて送られてくるセグメントを読み込むためGETB 4マクロ命令によって間接参照部分にSGBを割り付ける。

N.READXサービス・モジュールは，SGBのセグメントを処理プログラムの領域にコピーし，FREEB5 マクロ命令によって間接参照部分にSGBを返却する。

図4－52にSGBのフォーマットを示す。

図4－53にあるコネクションに対して処理プログラムからNREADマクロ命令が発信される以前に，他系からすでに3メッセージが到着している場合のSGBの連結関係を示す。

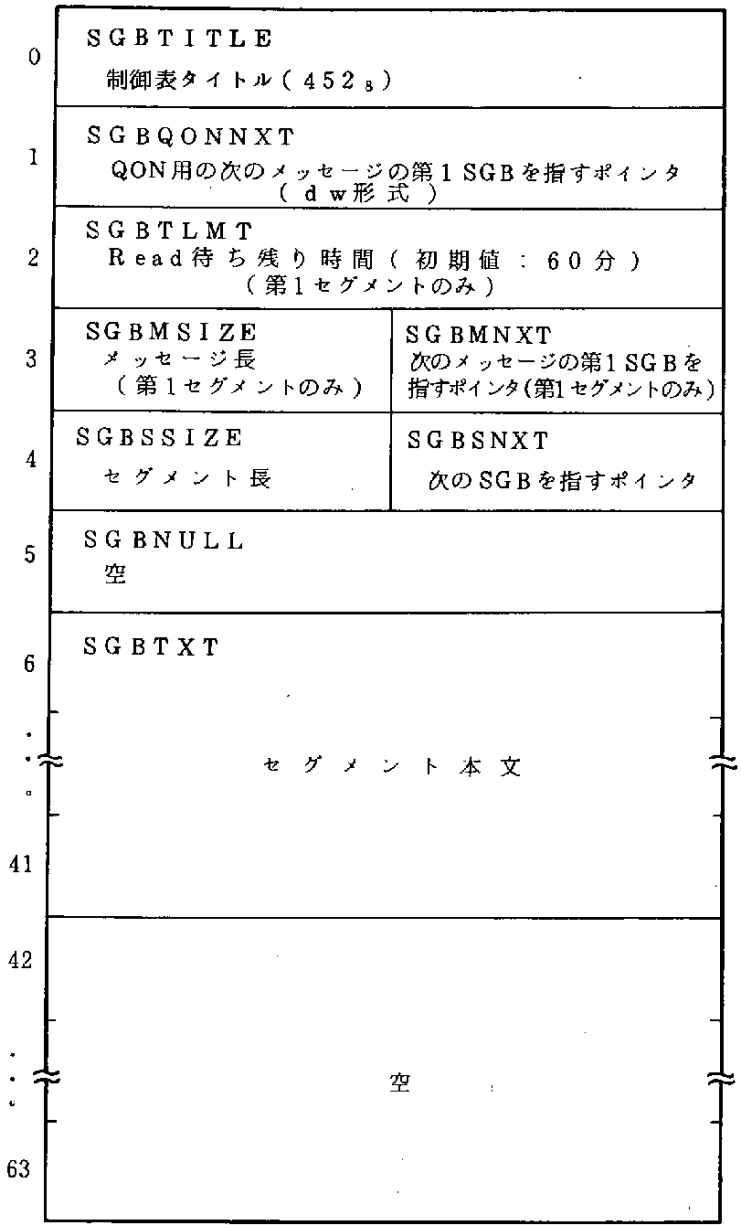


図4－52　SGBのフォーマット

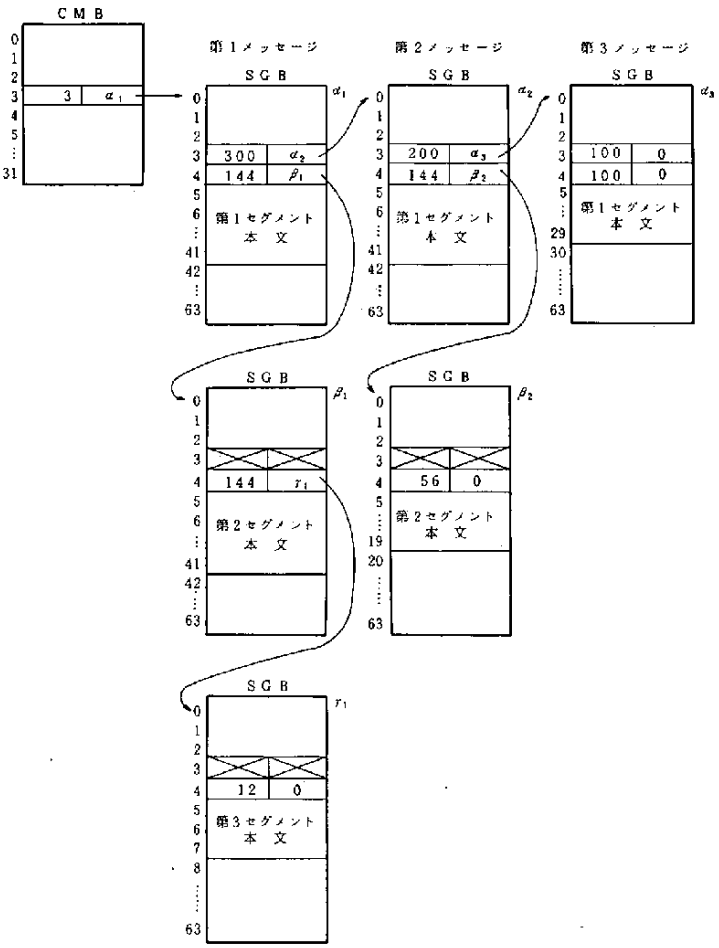


図4－53　先読みメッセージの管理

(4)　RDB

RDB（Read Block）は，処理プログラムのメッセージ受信要求を管理するために間接参照部分に動的に割り付けられる16語の制御表である。

処理プログラムからNREADマクロ命令が発せられた時，N.READ2サービス・モジュールがGETB5マクロ命令によって間接参照部分にRDBを割り付ける。

他系から当該メッセージが到着した時，N.READXサービス・モジュールがFREEB5マクロ命令によって間接参照部分にRDBを返却する。

図4－54にRDBのフォーマットを示す。

図4－55にNREADマクロ命令発信後，他系より3セグメント・メッセージが到着した場合のRDBの状態を示す。

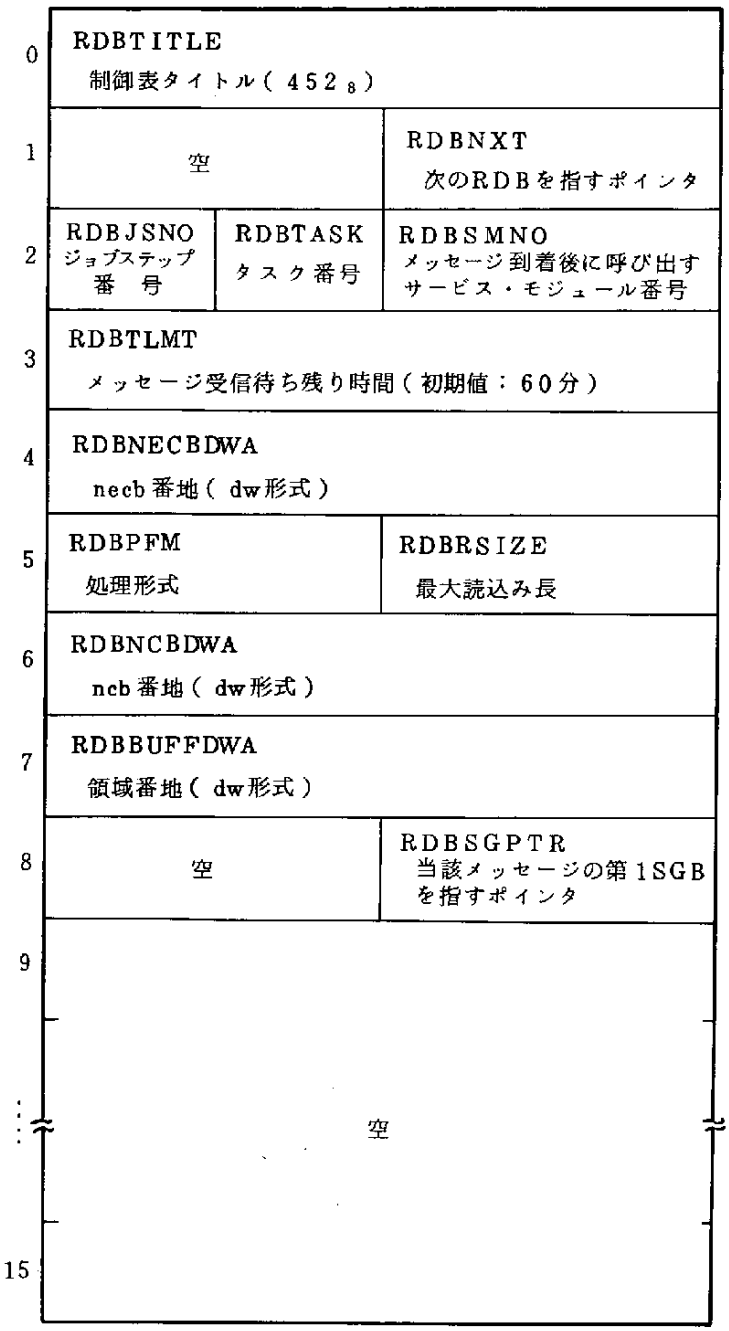


図4－54　RDBのフォーマット

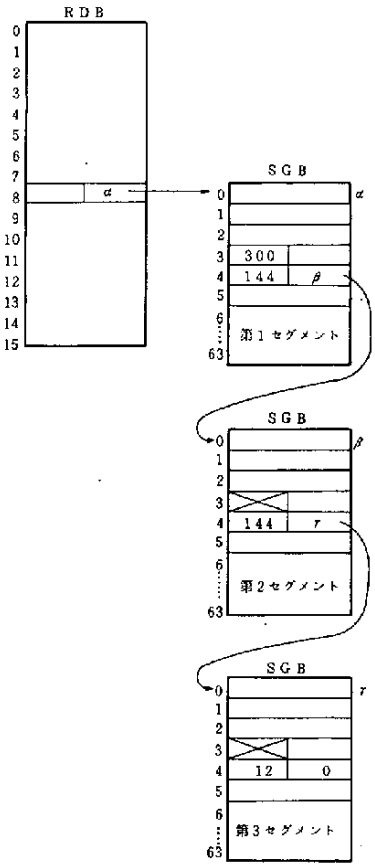


図４－５５　メッセージを受信した場合のＲＤＢの状態

(5)　ＬＳＮＢ

ＬＳＮＢ（Listen Block From My Host)は，Listen要求を管理するために間接参照分に割り付けられる16語の制御表である。

処理プログラムからLISTENマクロ命令が発せられた時，N.LISTENサービス・モジュールがGETB ５マクロ命令によって間接参照部分にLSNBを割り付ける。

他系からＲＦＣ制御メッセージが到着し，Listenが完了し，処理プログラムからListen後指定のNOPEN マクロ命令が発せられた時，N.NOPEN1 サービス・モジュールがFREEB ５マクロ命令によって間接参照部分にLSNBを返却する。

図４－５６，図４－５７にLSNBのフォーマットを示す。

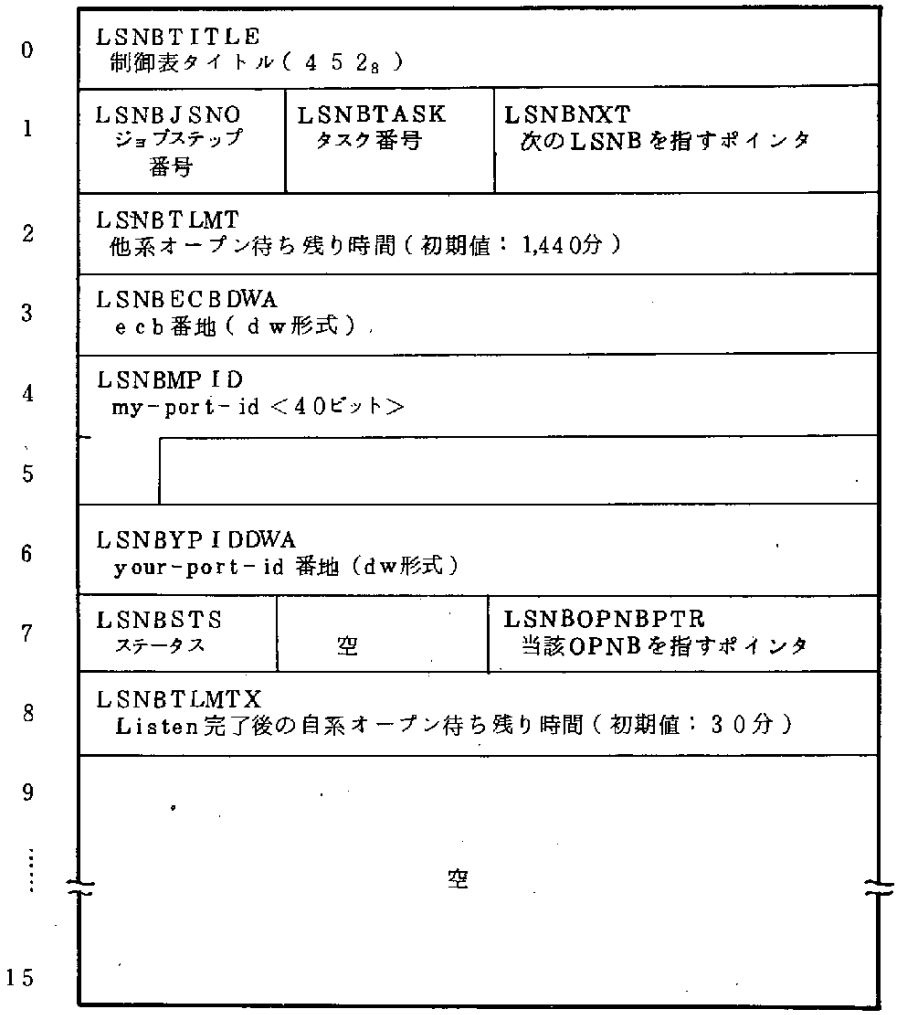


図4－56　LSNBのフォーマット

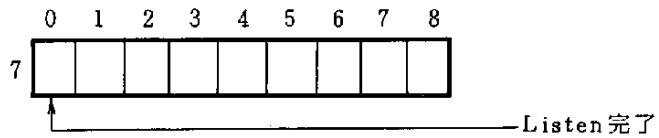


図4－57　LSNBSTSの詳細

LSNBは，図4－58に示すように直接参照部分にあるLSNQTに登録され管理される。LSNQT の右半分 LSNQTFは先頭 LSNBを指し， LSNQT の左半分LSNQTLは最終 LSNB を指す。

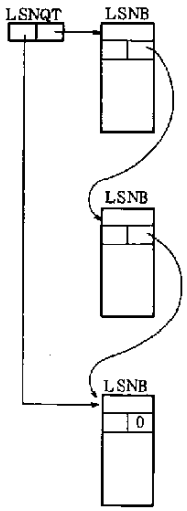


図4-58　LSNBの管理

(6)　ＯＰＮＢ

ＯＰＮＢ（Open Block From Your Host）は，他系から来たＲＦＣ制御メッセージを管理するために間接参照部分に動的に割り付けられる16語の制御表である。

他系よりＲＦＣ制御メッセージを受信した時，N.RFCRCV サービス・モジュールがGETB 4マクロ命令によって間接参照部分にOPNBを割り付ける。

処理プログラムからNOPENマクロ命令が発せられた時，N.OPEN 1サービス・モジュールがFREEB 5マクロ命令によって間接参照部分にOPNBを返却する。

図4-59にOPNBのフォーマットを示す。

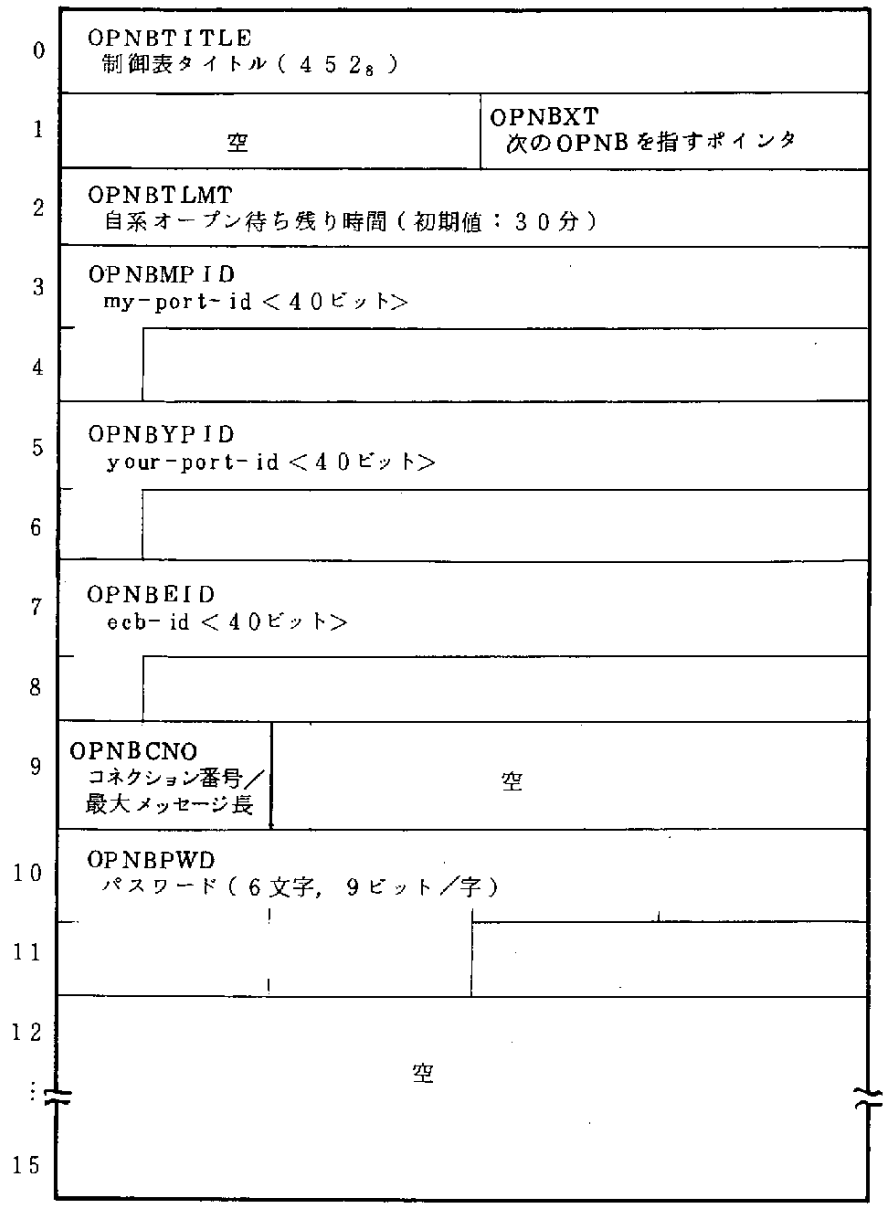


図4－59　OPNBのフォーマット

OPNBは，すでに該当するLSNBがLSNQTに登録されていない場合，図４－６０に示すように直接参照部分にあるOPNQTに登録され管理される。OPNQTの右半分OPNQTFは先頭OPNBを指し，OPNQTの左半分OPNQTLは最終OPNB指す。

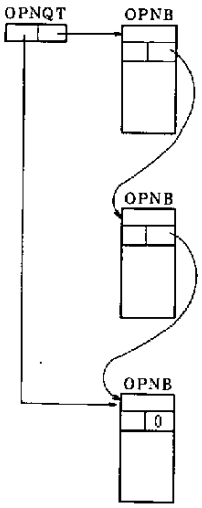


図 4－60　OPNBの管理

すでに該当するLSNBがLSNQTに登録されている場合，OPNBは当該LSNB に連結する（図４－６１参照）。

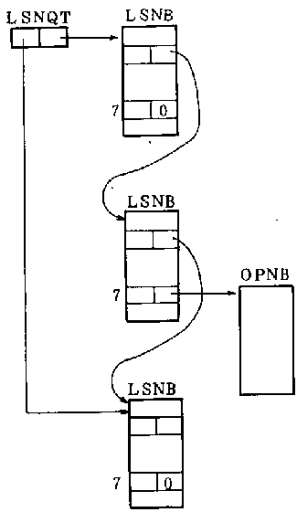


図4-61　OPNB と LSNB の連結

(7)　ＥＩＤＢ

ＥＩＤＢ（ECB-ID Block From My Host）は，他系からのポストの受信申込みを管

理するために間接参照部分に動的に割り付けられる８語の制御表である。

処理プログラムからSETECBID マクロ命令が発せられた時，N.SETEID サービス・モジュールがGETB ５マクロ命令によって間接参照部分にEIDBを割り付ける。

他系よりPOST制御メッセージが到着した時，N.SETEIDサービス・モジュールがＦＲＥＥＢ５マクロ命令によって間接参照部分にＥＩＤＢを返却する。

図４－６２にEIDBのフォーマットを示す。

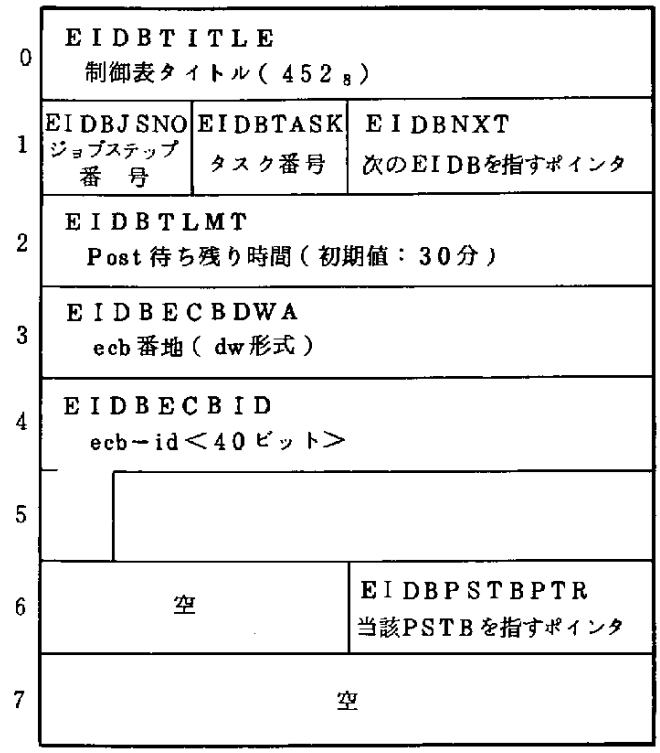


図４－６２　ＥＩＤＢのフォーマット

EIDBは，すでに該当するPSTBがPSTQTに登録されていない場合，図４－６３に示すように直接参照部分にあるEIDQT に登録され管理される。EIDQTの右半分EIDQTFは先頭EIDBを指し，EIDQT の左半分EIDQTLは最終EIDBを示す。

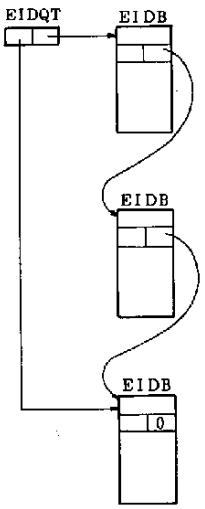


図４－６３　ＥＩＤＢの管理

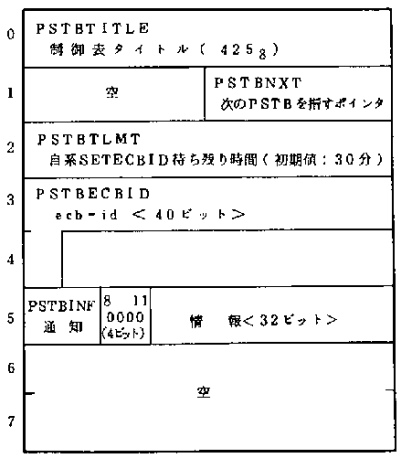
(8)　ＰＳＴＢ

ＰＳＴＢ（Post Block From Your Host）は，他系から来たPOST制御メッセージを管理するために間接参照部分に動的に割り付けられる８語の制御表である。

他系よりPOST制御メッセージを受信した時，N.SETEID サービス・モジュールがGETB4 マクロ命令によって間接参照部分にPSTBを割り付ける。

処理プログラムからSETECBID マクロ命令が発せられた時，N.SETEIDサービス・モジュールがFREEB５マクロ命令によって間接参照部分にＰＳＴＢを返却する。

図４－６４にPSTBのフォーマットを示す。



すでに該当するEIDBがEIDQTに登録されている場合，当該EIDBをEIDQTからはずし，PSTBをそのEIDBに連結する（図4-65参照）。

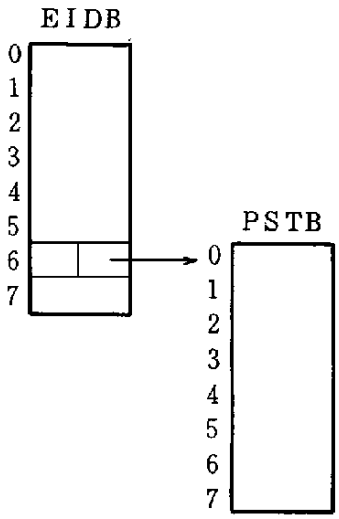


図4-65　PSTBとEIDBの連結

4.2.8　DSP－Serverシステム

A　DSPプロトコルの実現方法

F－NCPでは，他系にTSSサービスを利用させるためDSP-Serverシステムを開発した。

図4－66に示すように，サーバ側HOSTのユーザ・プロセスとしてサーバDSPを形成し，サーバ側のTSSシステムのバッファとNCPのバッファを接続することによって，サーバ側のTSS機能を実行する。

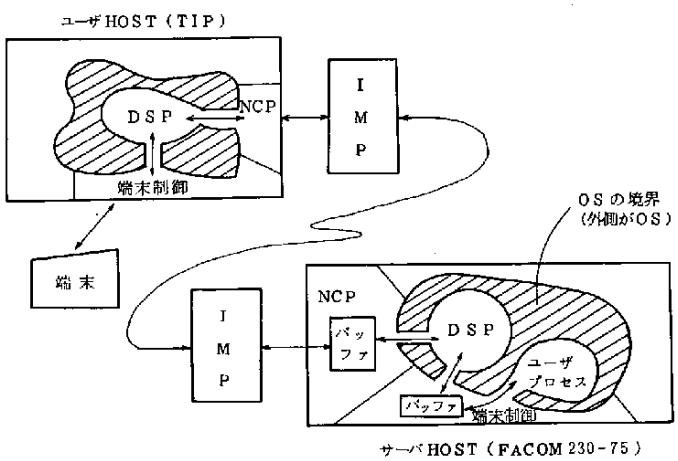


図4－66　DSPプロトコルの実現方法

B　ＣＰＳの概要

Ｍ－ⅥシステムのＴＳＳサービス用サブモニタはＣＰＳ（Conversational Programming System）と呼ばれ，多種類のプログラミング言語，ファイル操作，テキスト編集，会話形リモート・バッチ等の諸機能をサポートしている。

図４－６７にＣＰＳの動作概念図を示す。図中，点線はデータの転送およびその方向を示し，実線は制御の渡り方とその方向を示す。

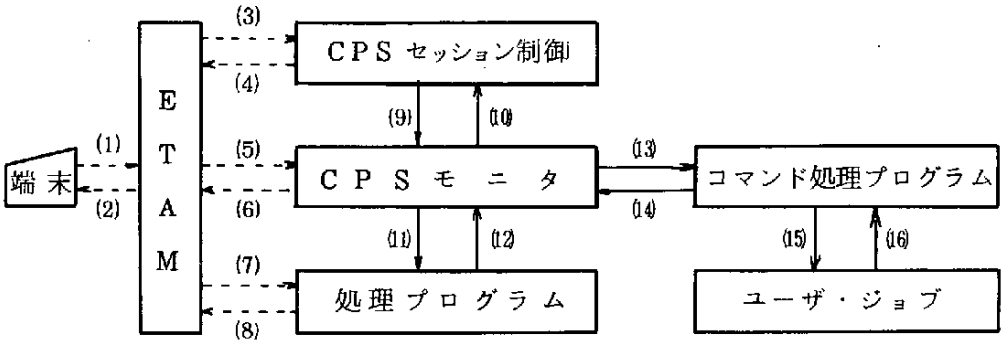


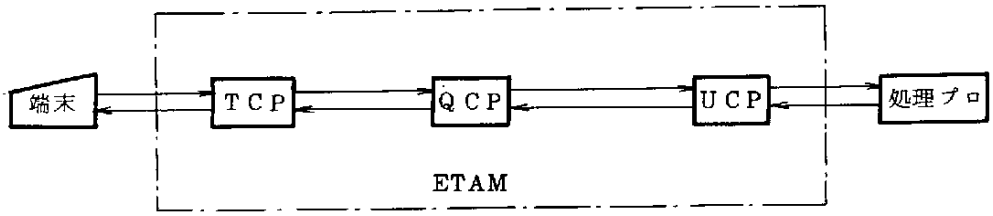
図４－６７　ＣＰＳの動作概念図

まず，端末よりユーザが¥¥ＣＰＳコマンド（ＣＰＳの利用開始を宣言するコマンド）を打鍵すると，このコマンドを受信したETAMは，ＣＰＳセッション制御に通知する。ＣＰＳセッション制御はETAMを通じてユーザの資格審査（ユーザ名，パスワードのチェック）を行ない，資格審査に合格するとＣＰＳモニタを起動する。ＣＰＳモニタは，セッション開設制御を呼び，セッションを開設する。セッションは１ジョブステップであり，かつ１ジョブとして動作する。セッションが開設されると，端末のユーザは，サブシステム等の機能を用いてプログラムを任意のプログラミング言語で作成，修正，実行などをする。端末からBYEコマンドが入力されると，制御はＣＰＳモニタに戻る。ＣＰＳモニタは，セッション閉設制御を呼びセッションを閉じ，制御をＣＰＳセッション制御に戻す。ＣＰＳセッション制御は，ユーザとETAMにセッションの終了を通知する。

C　ＥＴＡＭの概要

ETAM（Extended Telecommunication Aceess Method）は，Ｍ－Ⅵの通信制御プログラムである。

図４－６８にETAMの動作概念図を示す。図中，実線はデータの転送およびその方向を示す。



ＴＣＰ：　Terminal Control Program

ＱＣＰ：　Queue Control Program

ＵＣＰ：　User Control Program

図４－６８　ETAMの動作概念図

ETAMは大別して，ＴＣＰ，ＱＣＰ，そしてＵＣＰの３つのモジュールから構成されており，各々次のような役割りを受け持っている。

ＴＣＰ：端末装置の物理的特性，操作性の制御

ＱＣＰ：データ（メッセージ）の入出力管理

ＵＣＰ：処理プログラムと直接的なインタフェースを持ち，入出力要求の受付け，完了処理を行う

ＴＣＰは同一物理的性格を有する端末装置と対応したモジュール群からなり，ＱＣＰとＵＣＰは端末装置の性格に全く関係しない。従って，新しい端末装置をサポートするには，一定の規約に基づいてＴＣＰモジュールを作成し，これを組み込めば良い。端末の種類，回線の種類に関して多種類のものが新規に登録してくることが予想され，将来の展望に対し容易に対処できるように，端末の物理的特性をサポートする部分を徹底した独立モジュールとしそのインタフェースを外部に公開している。ETAM側からは，ＤＳＰ－Serverを特殊な端末装置（仮想端末装置）と見なし，新たにこの仮想端末装置をサポートするＴＣＰモジュール（N-TCPと呼ぶ）を開発した。

D　ＤＳＰ－Server システムの構成

ＤＳＰ－Server システムは，ＤＳＰ－Server 部とＴＳＳインタフェース部から構成されている（図４－６９参照）。

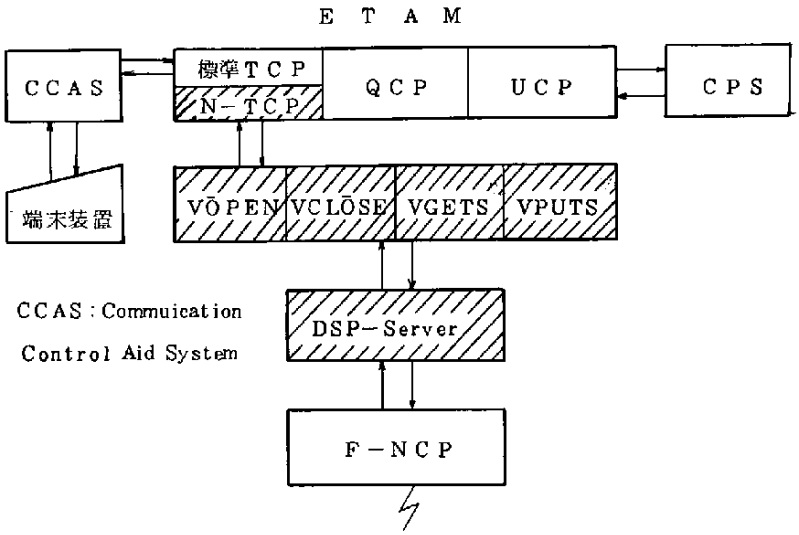


図4－69　DSP－Server システム構成

DSP－Server部はDSP－Serverからなり，処理プログラム・モードで走行する。TSSインタフェース部はN－TCPモジュール，VOPEN モジュール，VCLOSEモジュール，VGETS モジュール，そしてVPUTS モジュールから成り，サービス・ルーチン・モード（割込み不可モード）で走行する。

E　ETAM連絡用マクロ命令

DSP－ServerとETAMのN－TCPとの間のコミュニケーションのために，次の4つのマクロ命令を作成した。

① 仮想端末オープン命令 …… VOPEN

② 仮想端末クローズ命令 …… VCLOSE

③ ゲットETAM命令 …… VGETS

④ プットETAM命令 …… VPUTS

以下に，各マクロ命令について述べる。

(1) VCBマクロ命令

仮想端末をオープンするに必要な情報を含むのがVCB（Virtual Control Block）であり，VCBマクロ命令により作られる。

(a) 書き方

名札欄	命令欄	オペランド欄
vcb名	VCB	TCODE ＝伝送コード， DELCH ＝境界文字， BLKSIZE ＝最大メッセージ長， ECBAD＝ecb番地， MODE ＝プログラム・モード

(b) オペランド

・TCODE　　使用する伝送コードを指定する。

JIS　　ＪＩＳコード

EBCDIC　　EBCDICコード

・DELCH　　メッセージの終りを示す境界文字を指定する。

'O_1 O_2 O_3'　8進３桁で指定する。

・BLKSIZE　　伝送されるメッセージの最大長をバイト数で指定する。

・ECBAD　　ETAM（Ｎ－ＴＣＰ）より作業要請を受け付けるecb番地を指定する。

＜ecb名＞

0	8	9	10	11	12	35
通知情報１		W	C	S	通知情報２	

W：Wait bit, ビット9

C：Completion bit, ビット10

S：Single wait bit, ビット11

① 通知情報１

現在は使用せず。

② 通知情報２

内容	意　味	アクション
1	送信依頼	VGETS, NWRITE
2	受信依頼	NREAD, VPUTS
3	促進文字付受信依頼	VGETS, NWRITE, NREAD, VPUTS
4	閉塞依頼	VCLOSE

・MODE　　プログラムのモードを指定する。

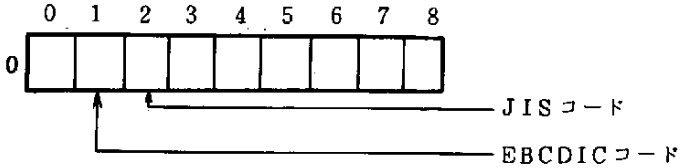
SERVER　　DSP-Server全体を制御するプログラムの場合指定する。

TERMINAL　仮想端末１台をサポートするプログラムの場合指定する。

(c) 展開形

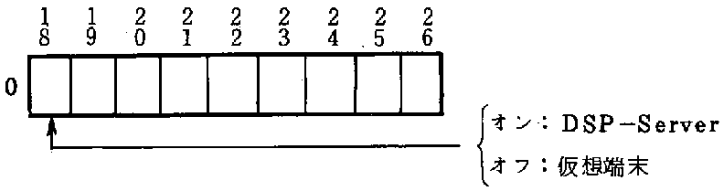
＜vcb名＞＋0	TCODE	DELCH	MODE	
1	BLKSIZE		RCDSIZE	
2	TMLNO			
3	ECBAD			
4	＜システム作業域＞			

・TCODE　　指定する伝送コードに対応するビットをオンにする。



・DELCH　　境界文字を9ビットでセットする。

・MODE　　指定するプログラム・モードに応じてビットをオン・オフする。



・BLKSIZE　メッセージの最大長をバイト数でセットする。

・RCDSIZE　実際のメッセージ長がセットされる。

・TMLNO　　ETAM(N-TCP）が割り当てた端末番号がセットされる。

・ECBAD　　ETAM(N-TCP）より作業依頼を受け付けるecb番地をセットする。

(2) VOPENマクロ命令

VOPENマクロ命令は，ETAMにDSP-Serverあるいは仮想端末がサービスを開始したことを通知する。

(a) 書き方

名札欄	命令欄	オペランド欄
〔記号名〕	VOPEN	ｖｃｂ名

(b) オペランド

・ｖｃｂ名　　オープンしようとしているＶＣＢの先頭番地を記号名で指定する。

(c) 展開形

0	LR	＊＋3
1	TRP	S．VOPEN
2	J	＊＋3
3	ADCON	＊＋1
4	ADCON	ｖｃｂ名

(d) 通知情報

システムは，VOPENマクロ命令完了後Aレジスタに完了状態をセットする。

内　容	意　　味
0	正常完了
1	空き仮想端末がない
2	ＤＣＢの内容が誤っている
3	パラメータ番地が誤っている
4	処理プログラム・ジョブステップ作業域に空がない

(e) 補足事項

・仮想端末用プログラムの場合，VOPEN マクロ命令完了後ＶＣＢのTMLNO にこのプログラムに割り当てた端末番号がセットされる。

(3) VCLOSEマクロ命令

VCLOSEマクロ命令は，ETAMにDSP-Server あるいは仮想端末がサービスを終了したことを通知する。

(a) 書き方

名札欄	命令欄	オペランド欄
〔記号名〕	VCLOSE	ｖｃｂ名

(b) オペランド

・ｖｃｂ名　　クローズしようとしているＶＣＢの先頭番地を記号名で指定する。

(c) 展開形

0	ＬＲ	＊＋3
1	ＴＲＰ	Ｓ．ＶＣＬＯＳＥ
2	Ｊ	＊＋3
3	ＡＤＣＯＮ	＊＋1
4	ＡＤＣＯＮ	ｖｃｂ名

(d) 通知情報

システムは，VCLOSEマクロ命令完了後Aレジスタに完了状態をセットする。

内　容	意　　味
0	正常完了
2	ＤＣＢエラー
3	パラメータ番地が誤っている
4	処理プログラム・ジョブステップ作業域に返却できない　い

(4)　VGETSマクロ命令

VGETSマクロ命令は，ETAMよりメッセージを読み込むことを指令する。

(a)　書き方

名札欄	命令欄	オペランド欄
〔記号名〕	VGETS	ｖｃｂ名，領域番地

(b)　オペランド

・ｖｃｂ名　　Getするvcbの先頭番地を指定する。

・領域番地　　メッセージを読込むべき領域の先頭番地を指定する。

(c)　展開形

0	ＬＲ	＊＋3
1	ＴＲＰ	Ｓ．ＶＧＥＴＳ
2	Ｊ	＊＋4
3	ＡＤＣＯＮ	＊＋1
4	ＡＤＣＯＮ	ｖｃｂ名
5	ＡＤＣＯＮ	領域番地

(d)　通知情報

システムは，VGETSマクロ命令完了後Ａレジスタに完了状態をセットする。

内　容	意　　味
0	正常完了
2	ＩＪＰＯＳＴマクロ命令が正常に完了しない
3	パラメータ番号が誤っている
4	処理プログラム・ジョブステップ作業域に返却できない

(e)　補足事項

・VGETマクロ命令は，ＶＣＢで指定されたＥＣＢに送信依頼あるいは促進文字付受信依

頼がポストされた時発信される。

・VGETS マクロ命令完了後，ＶＣＢのRCDSIZEに実際のメッセージ長がセットされる。

(5) VPUTSマクロ命令

VPUTSマクロ命令は，ETAM にメッセージを書き込むことを指令する。

(a) 書き方

名札欄	命令欄	オペランド欄
〔記号名〕	VPUTS	ｖｃｂ名，領域番地

(b) オペランド

・ｖｃｂ名　　PutするＶＣＢの先頭番地を指定する。

・領域番地　　書き込むべきメッセージが格納されている領域の先頭番地を指定する。

(c) 展開形

0	ＬＲ	＊＋3
1	ＴＲＰ	Ｓ．ＶＰＵＴＳ
2	Ｊ	＊＋4
3	ＡＤＣＯＮ	＊＋1
4	ＡＤＣＯＮ	ｖｃｂ名
5	ＡＤＣＯＮ	領域番地

(d) 通知情報

システムは，VPUTS マクロ命令完了後Ａレジスタに完了状態をセットする。

内　容	意　　味
0	正常完了
1	ETAMジョブステップ作業域に空がない
3	パラメータ番地が誤っている
4	処理プログラム・ジョブステップ作業域に返却できない

(e) 補足事項

・VPUTS 命令完了後，ＶＣＢのRCDSIZEに境界文字までのメッセージ長がバイト数でセットされるので，確認に利用できる。

・ETAMジョブステップ作業域に空がない（Ａレジスタの内容が１）の場合，一定時間経過後，再試行すること。

F　ＤＳＰ－Server

ＤＳＰ－Server は，同時に最高５台の仮想端末として動作できるようになっており，Server 全体を管理・制御するServer 制御タスク（ SCT，Server Control Task）と，１台の仮想端末として動作するServer端末タスク（ SVT，Server Terminal Task ）からなる（図４－７０参照）。

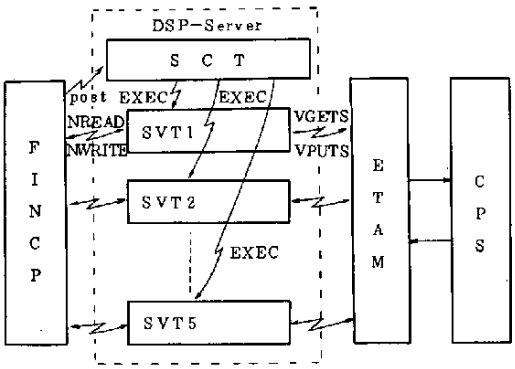


図４－７０　DSP－Server の構成

(1)　Server 制御タスク（ ＳＣＴ）

Server 制御タスクは，Listenを出し，他系のＤＳＰ－Userからのオープンを待つ。Listen が完了すると，空いている仮想端末があるかどうかを調べ，もし空いていればVOPENマクロ命令を発し，Ｎ－ＴＣＰに空き端末を割り当ててもらう。空き端末がない場合は，他系のＤＳＰ－User にそのむねを通知する。仮想端末が割り当てられると，コネクションを確立し，Server端末タスクを起動し，１サイクルのサービスを終了する。以下同様。

又，Server 端末タスクから終了通知を受け取ると，そのServer 端末タスクが使用していたコネクションを切断する。

(2)　Server 端末タスク（ ＳＶＴ）

Server 端末タスクは，Server 制御タスクによってListen完了後，子タスクとして起動され，ＤＳＰプロトコルに従ってＤＳＰ－User とメッセージの授受を遂行し，受信したメッセージをＣＰＳに渡し，ＣＰＳ（ ETAM ）より送信依頼の出ているメッセージを受け取り，ＤＳＰ－User に送信する送受信の中継媒体として動作する。

4.3 HITAC—NCP

4.3.1 H—NCPの概要

HITACにおいてネットワーク・サービスを管理するH—NCPは，モニタ的性格を持つNCPと，ユーザ・プログラムにリンケージ・バインドされ，NCPと連絡をするディスパッチ・モジュールからなる。

ネットワーク・サービスを受けたいと考えるユーザは，ユーザ・プログラム中で，NCPサービス・コマンド（ユーザ・レベル・コマンド）を用い，ディスパッチ・モジュールをバインドしたネットワーク・ジョブ（プロセス）を作成する。このプロセスは，NCPと各種の連絡をとり，ネットワーク・リソースを利用することが可能である。

以下にNCPの位置付け，ロジカル構成，データ・フロー，コントロール・フロー及び設計方針について概説する。

A．H—NCPの位置付け

NCPはHITAC8450処理装置内で，リアルタイム・クラスのジョブとして動作し，EDOS—MSOの助けをかりネットワーク・モニタとしての機能を果す。一方ディスパッチ・モジュールは，ユーザ・プログラムに付加され，ネットワーク・プロセス特有のNCPとの連絡機能を果す。

NCPは機能実現のために各種のエグゼクティブ連絡マクロを用いて，EDOS—MSOとSVC（スーパバイザ・コール）の形でコミュニケーションを行ない，HITAC内のプロセス及びIMPとのデータ転送を行なう。

EDOS—MSO側からみれば，NCPはハイ・プライオリティな1連のタスクにすぎないが，ある種のコントロール・タスク（例えばリモート・リーダ／ライタ・タスク）より優先権が高いタスクであり，EDOS—MSOにおける非常駐コントロール・タスク的性格を持つ。またプロセスは一般のバッチ・ジョブとして取り扱われる。

NCPとプロセスの関係について考えると，プロセスからみるとNCPはネットワーク・サービス用のモニタであり，ネットワークの直接利用を望むユーザは，必ずNCPと会話を行なう必要がある。但し間接的にネットワーク利用を考えるユーザは，ハイ・レベル・プロトコルを用いる事ができ，NCPとの直接的な会話を行なわなくてもネットワーク・サービスを受ける事ができる。また，NCPはプロセスの発生，消却の機能を持ち，必要に応じてプロセスを発生し会話を行なったり，一方的な指示を行なう事もある。

次にNCPとサブネットの関係を考えると，NCPはインターフェース・アダプタを通じてIMPとのデータ転送を行なう。すなわちサブネット系とプロセス系を結ぶパイプ的役割を果たすものであり，同時にサブネット系とプロセス系の監視をするプログラムであると言える。

NCPとEDOS—MSO及びIMP，プロセスの関連を図4—71に示す。プロセスA，B，Cは

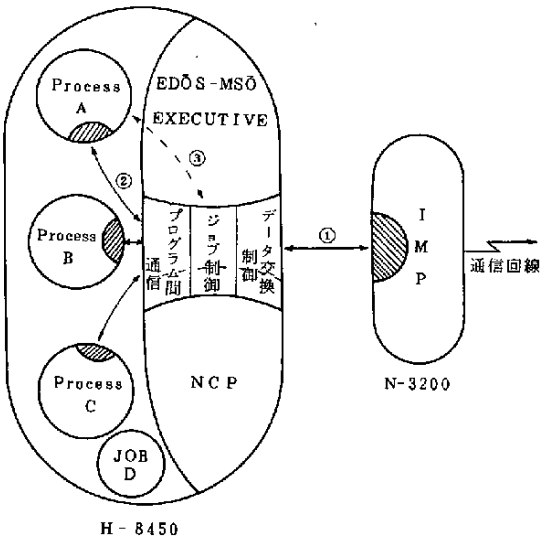


図4－71　H—NCPの位置付け

ネットワークの利用を要求し，NCPと連絡をとっている。ジョブDはネットワークの利用を要求してないバッチ・ジョブである。各プロセスの斜線の部分は，ディスパッチ・モジュールで，NCPサービス・コマンドを用いることによって付加されたものである。

プロセスAはNCPにより起動されたジョブであり，プロセスB，C及びジョブDはEDOS—MSOのジョブ・コントローラにより直接起動されたジョブである。

①に示すNCPとIMPとのデータ伝送は，フィジカルI／Oマクロを用いる事によって行なわれ，OSのデータ交換制御を利用している。

②に示すNCPとプロセスとの会話は，プログラム間通信マクロを用いる事によって行なわれ，OSのプログラム間通信制御を利用している。

③に示すプロセスAの起動は，キー・イン／アウト・シミュレーション・マクロを用い，EDOS—MSOのジョブ・コントローラの制御を行なう事により実現される。

B　H—NCPのロジカル構造

HITAC内のNCPは各種の機能を持つが，その論理的な構造を図4－72に示す。

割込み管理は，IMP，プロセス，エグゼクティブから発生する各種の割込みのコントロールおよびスケジューリングを行ない，非割込み管理は，割込みモードで処理されていない時のコントロールを行なう。また，マルチタスク管理は，NCPサブタスクの管理を行なう。

コネクション管理，プロセス管理は割込み管理，マルチタスク管理から呼び出され，各種の処理を行なった後呼び出された管理へ制御をかえす。

ネットワーク・プロセスは，NCPからのデータ転送による割込み処理及び，ユーザ・コマンドの処理を行なうディスパッチ・モジュールと，ユーザー処理部分に大きく分かれ，ユーザ・プロ

グラムが各種のNCPサービスコマンドを用いる事によって，ネットワークを利用する。

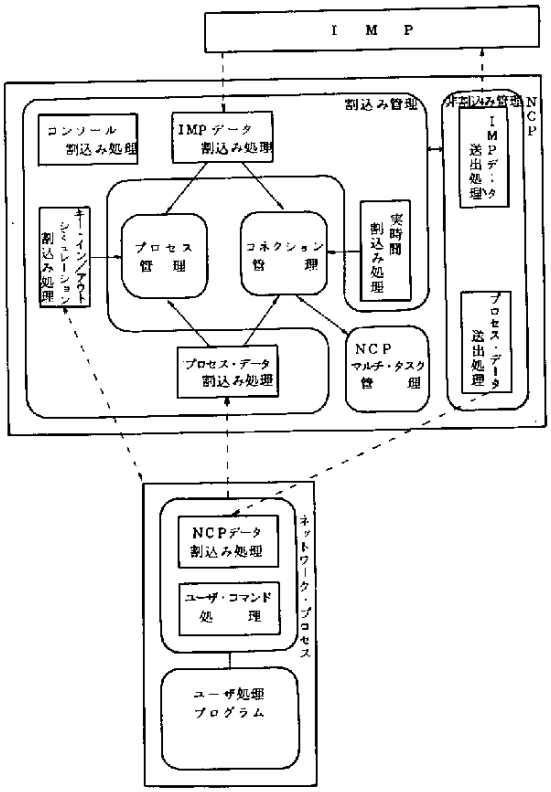


図4－72　H—NCPのロジカル構造

C．データ・フロー

IMP，NCP，プロセス間のデータは非周期に発生し，それぞれの目的に応じた流れを形成する。NCP内でのデータの流れはIMP，プロセス両者の同期をとって行なわれ，データ蓄積のためのキューを作成する。

キューよりIMP，プロセスへのデータ転送が行なわれるのは，両者からデータが発生してない状態であり，しかも，NCP内にデータが流れていない状態の時である。NCP内のデータの流れを図4－73に示す。

IMPあるいはプロセスより送られてきたデータは，メッセージ種類別に処理され，IMP又はプロセスへのWriteキュー（IMP Write Queue, IPC Write Queue）にキューオンされる。この時メッセージが数パケットに及ぶものについては，さらに読込みの処理を行なう。又，IMPからのメッセージが"receive"の時にはreceiveキューよりキュー・オフの処理を行なう事がある。データの流れが停止した時，NCPは，IMP又はIPC Writeキューよりデータを取り出し，IMP又はプロセスにデータ転送を行なう。この時IMPへのデータ転送終了後receiveキューへの登録処理を行なう事がある。

NCP内でのデータ・ストリームは，IMP，プロセスのアクションによって生じる事がほとんどであり，そのデータによって生じる処理が必要に応じて新しいデータを作成し，リアクションを発生する。

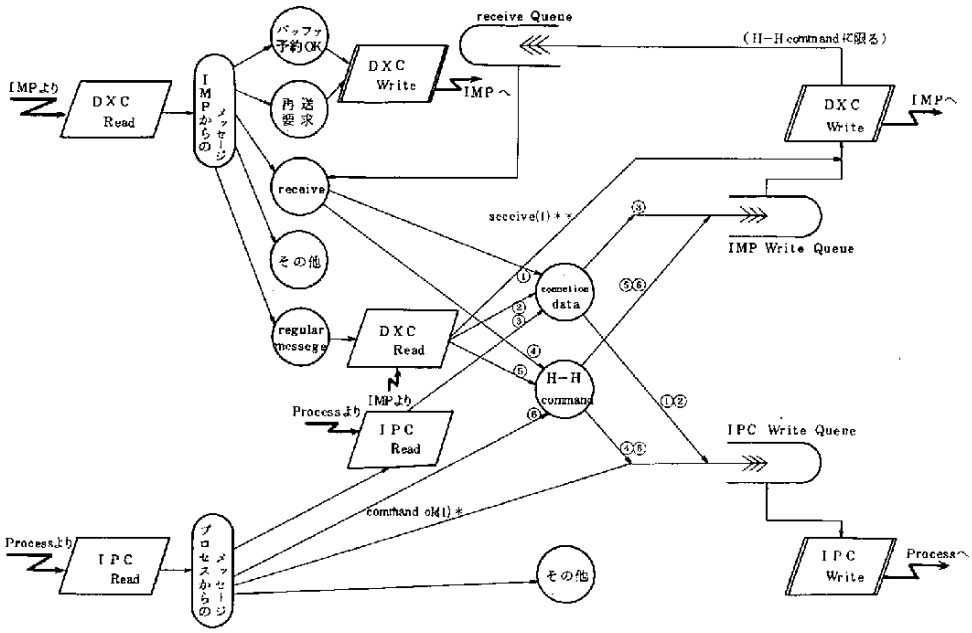


図4－73　IMP－NCP－Process間のデータ・フロー

B　コントロール・フロー

IMPからのデータ，プロセスからのデータをNCPで処理する場合の制御は，割込み処理の実施から非割込み処理の実施へと移行する。IMP，プロセスからのデータは，割込みという型式でNCPに連絡され，NCPではコネクションに関する処理およびプロセス制御に関する処理を行ない割込み処理を抜け出す。非割込み処理の状態では，IMPまたはプロセスに送出するデータがあるか否かを調べ，送出データがある場合にIMP又はプロセスにデータ転送を行なう。送出データがない時には，IMPまたはプロセスからのアクションを待つ。すなわちNCPのタスクは待ち状態となり制御は他のタスクに移される。

この時制御の移行として重視されるのは，割込み処理状態，非割込み処理状態のいずれにあっても，非同期に割込みが発生する事にある。NCPでは，これら非同期に発生する割込みに関して特別な制御を行なう必要がある。すなわち多重割込み管理の方式として，優先順位スケジュールと状態遷移のスケジュールの方式を併用した形をとっている。

また，各割込み処理から呼出される各種の処理は，同時に呼ばれる事があるのでシリアル・リューザブルな処理を用いてコントロールされる。

図4－74に割込み状態と非割込み状態で使用する処理モデルを示す。

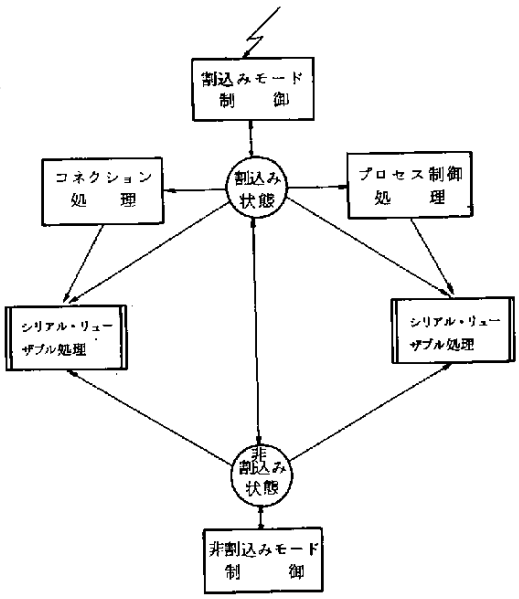


図4－74　処理モデルのステート

E　設計方針

H–NCPの設計において基本的には，ベーシックなネットワーク・サービスをサポートする事を方針とする。すなわち，HITACのユーザが，ネットワークを利用したい時簡単な手続きで利用できる様なマクロ体系を用意し，プロセスに追加するディスパッチ・モジュールによってNCPと基本的な会話を行なうための機能をサポートする。

NCPの設計上の留意点は以下の通りとする。

① 他のプログラムへの影響を少なくする。

ジョブ・スケジューリング，タスク・スケジューリング等は，EDOS–MSOに任せ，NCPが直接スケジューリングを行なわない。又，NCPはコア・メモリの独占（常駐）は行なわず，ネットワーク・サービスを必要としない時にはコア上にいない。

NCPの用いるIMPとの会話用のチャネルは特定チャネルであり，他のプログラムとの競合は行なわない。

② オーバー・ヘッドを少なくする。

ネットワーク・プロセスがNCPを利用している時はそのプロセスに対するオーバー・ヘッドが増大している事となるが，そのオーバー・ヘッドはなるべく少なくする方針で設計する。すなわちNCPの処理は，割込みオリエンティドに処理され，NCPの使用されない時にはCPUを利用しないアイドル・タスクとなる方式をとる。又，オーバレイ構造を少なくし，オーバレイのためのOSのオーバ・ヘッドを少なくする。

③ プログラム・サイズの最少化

モニタ的役割のNCPはなるべく，プログラム・サイズの小さいものでしかも機能を満足させるものでなければならない。

④ 利用者側のニーズを考慮する。

HITACに存在するプロセス及びHITAC利用者のニーズを考慮に入れ，エラーの状態が分るデバック機能をもち，いつでも利用可能であり，しかもハング・アップしないコントロール・プログラムを考え，プロセスの動作時間が大きい事も設計方針の中に取り入れる。

HITAC 8450はデータ処理に適合するバランスのとれた周辺機器構成を持つが，当財団での利用形態もデータ処理用に広く利用されている。

これは，ファイルの取り扱いにフレキシビリティがある事と，ファイルの取り扱いが容易である事により，データ処理用のプログラムの多くがHITACを用いて作成されている為である。

また，グラフィック・ディスプレイ装置HITAC 8811を結合した構成を持ち，グラフィック・サポート・プログラムも各種開発されている。

このHITAC 8450を他の大型計算機と比較すると，計算速度，ジョブ・スケジューリング等の面で難点があり，他の大型計算機で処理する方が効率のよいプログラム（例えば技術計算用のFORTRAN等）は，他の計算機に実行させる方がよい場合もあり，機能分担のあるネットワークを利用すると効果的となる。又，コアサイズによるプログラム・サイズの制限，タイム・スライスの問題等の影響を考える必要も生じている。

現状におけるHITAC 8450の利用を考えると，データ処理，グラフィック処理で広く利用されている。ネットワークに結合された状況であっても今までのサービスをそのまま続ける事が必要となる。その為にメモリの追加がなされた。

又現状のバッチ・ジョブでのリソース・アロケーションにおいては，ほとんどのDISCパックが，ワーク・パックとして使用されているので，ネットワーク専用のパックをリザーブする事がある程度容易である。

NCP設計に当ってコントロール・プログラムとしてNCPをどの位置におくかの選択は非常に難しい問題である。先ず考えられる事が，EDOS－MSO EXECUTIVEの中に組み込む方式，次にEXECUTIVEと同等な1つのサブ・モニタとする方式，最後にユーザ・プログラムとしてEXECUTIVEの下でスレーブ・モードで動作する方式がある。

現在のNCPは第3番目のEXECUTIVEの下で，リアルタイム・クラスのジョブとして動作するNCPである。これを設定した理由は以下の通りである。

① 既存のエグゼクティブ・連絡マクロでネットワーク・モニタとしての機能を果す事が可能である。

② モニタの中に組み込んだり，サブ・モニタとする場合EDOS－MSOの詳細な部分を知る必要

がある。

③ モニタの改造により他のプログラムへの影響が考えられる。（例えばモニタを拡張する事によるオーバーヘッド，ユーザ領域の縮小，バージョン・アップによる修正の必要性，メインテナンスの困難さ）

④ リアル・タイム・クラスのジョブとしてかなりのハイ・プライオリティ・タスクとして存在する事が可能

⑤ 割込み処理のカバーが可能

4.3.2 EDOS—MSO

H—NCPはEDOS—MSOの助けをかりてネットワーク処理を行なう。ここでEDOS—MSOの概要及び，H—NCP機能実現の為に特に必要となる，データ交換制御機能，プログラム間通信機能，キー・イン／アウト・シミュレーション機能について述べる。

A．EDOS—MSOの概要

EDOS—MSOは次の様な特長を持つ。

① マルチ・ステージ方式

メモリ資源は利用形態により多様な要求に応じて配分される必要がある。このメモリ資源を効果的に配分するためにマルチ・ステージ方式を用い，ユーザ・メモリ領域の一部を最大4個までのステージに分割し，ステージ間の独立性とステージ内の可変性を持つ。1つのステージは特権エリアと非特権エリアの2つの領域（パーティション）からなり，特権エリア内では1本，非特権エリア内では2本のジョブ・ストリームが実行可能である。特権エリア内のプログラムは，非特権エリア内のプログラムをロール・アウトさせ自分のメモリ領域として使用できる。また非特権エリア内のプログラムは互いにメモリ領域を融通しあって実行することができる。

② 入出力チャネル・キュー制御方式

高トラフィック・リアルタイム処理での応答を迅速にし，特定サブタスクの処理の遅れを防ぐため，メインタスクおよびサブタスクからの要求は，要求順に従って入出力サービスを行なうFIFO方式である。

③ ジョブ・スケジューリング方式

ジョブ・スケジューリングの要素として，プライオリティ，開始時刻，システム資源の確保を用いる。プライオリティにより優先ジョブの割込みができ，開始時刻指定により入力データの完成を待つことができる。さらにシステム資源の確保は，ジョブの中途での待ちをなくすジョブ単位確保方式と，融通のきくジョブ・ステップ単位確保方式がある。

④ 入力リーダと出力ライタ

ジョブの入力ラン，実行ラン，出力ランを独立に並行して実行し，入出力装置及び処理装置

の使用効率をあげる。入力ラン用の入力リーダは最大3本，出力ラン用の出力ライタは最大6本まで同時動作可能である。

⑤　システム区域とユーザ区域

システム資源のうち主記憶装置は，システム区域とユーザ区域に大別される。システム区域ではコントロール・プログラム（BCSを含む）が実行され，ユーザ区域では処理プログラム及び管理プログラムのうちの，入力リーダ，出力ライタ，FCP，ジョブ・コントロール等が実行される。また処理装置に記憶装置保護機構が付加されているシステムでは，メモリ・プロテクト機能によってコントロール・プログラムおよびユーザの処理プログラムを破壊するのを防止することができる。

EDOS−MSOのソフトウェア構成は図4−75の通りである。

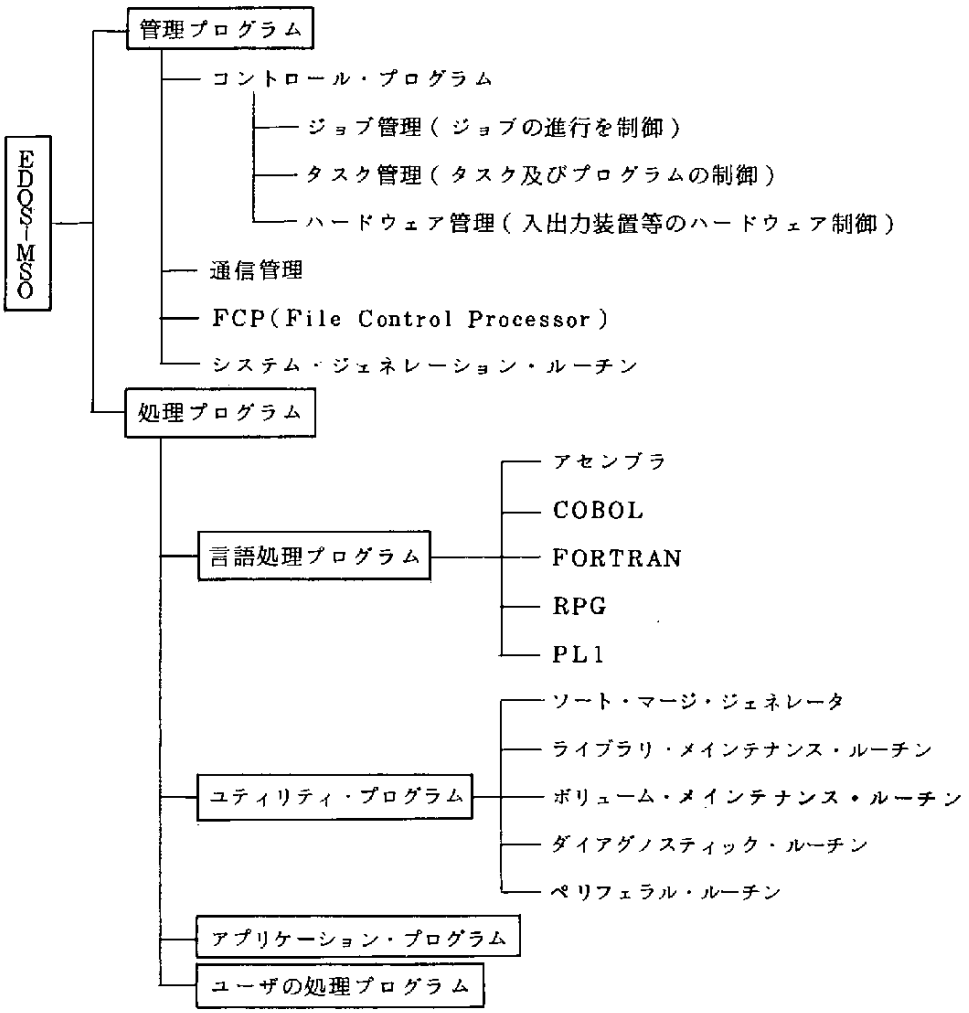


図4−75　EDOS−MSOのソフトウェア構成

a．ジョブの構成とシステムでの処理

バッチ処理，リアルタイム処理のジョブは，システム・リーダ（カード読取り装置，磁気テープ装置，ランダム・アクセス装置）によりシステムへ入力され，実行される段階でジョブ・ストリームを形成する。EDOS—MSOでは最高12本のジョブ・ストリームを形成する事ができる。

1つのジョブは1つのメインタスクと複数個のサブタスクから構成される。このジョブの性格はJOBステートメントによって記述する。

システムへのジョブの入力は，ジョブを構成するジョブ・コントロール・ステートメントとデータを入力する事によって行なわれ，処理の実行順序はシステムで決定する。実行時に必要なデータはファイルとして用意されている。ただしリアルタイム・ジョブでは，実行時に不規則な時間間隔で発生し，ジョブへの入力となるデータがある。システムからのジョブの出力は，いったんランダム・アクセス装置上におかれ，システムでスケジュールしてから出力装置へ出力する方法と，直接出力装置へ出力する方法とがある。

b．ジョブの実行とジョブ・クラス

入力リーダで入力ジョブ待合せキューに登録されたジョブは，ジョブ・コントロールによって各ジョブ・クラスごとに引出されて実行される。ジョブ・コントロールが各ジョブをスケジューリングするときは，そのジョブを実行するエリアでスケジューリングする。このスケジューリングでは，開始優先権で順序決定を行ない，同一優先権で開始時刻の指定があれば早い開始時刻を優先する。又システム資源（主記憶装置，入出力装置）不足のためにジョブを削除したり，抑止状態にしたときなどは，次に高い優先権のジョブを実行する。

ジョブ・ストリームにはジョブの実行に必要なシステム資源が割当てられる。ジョブ・ストリームに与えられたシステム資源の集まりをジョブ・クラスと呼び，1つのステージだけを考えた場合，R／A，B，Cの3個のジョブ・クラスがシステム資源を分けあっている。このジョブ・クラスはその状態により，実行可能状態，停止状態，休止状態，中断状態，開始待ち状態，実行状態の6つに分けられる。又，ジョブ・クラス・A／Rは特権エリアでのみ実行され，ジョブ・クラスB，Cは非特権エリアで実行される。すなわちジョブ・クラスA／Rはジョブ・クラスB，Cのプログラムをロール・アウトする権利を持っている。非特権エリアはジョブ・クラスBとCで2分されて使用される。Bクラスのジョブは非特権エリアの先頭から，Cクラスのジョブでは非特権エリアの後方から主記憶装置を占有し，ジョブ・クラスBとジョブ・クラスCの境界はジョブ単位で動的に変化する。図4—76に1つのステージについて示す。

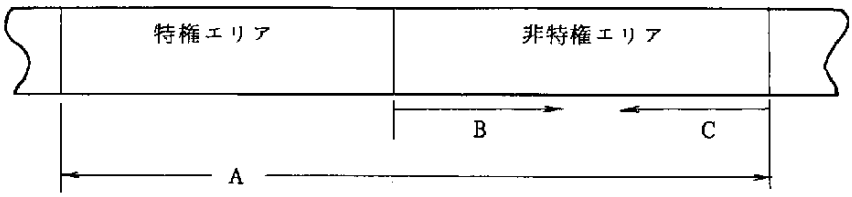


図4－76　ステージとジョブ・クラス

c．入力リーダと出力ライタ

ジョブ実行に必要なジョブ・コントロール・ステートメント群とカード形式のデータをシステムに登録するプログラムが入力リーダであり，これには常駐入力リーダと非常駐入力リーダがある。入力ジョブ待合せキューに入力情報を，出力ジョブ待合せキューに出力情報を一時的にランダム・アクセス装置に格納する（ジョブ・スタック）目的は次の通りである。

(i)　ジョブを入力ラン，実行ラン，出力ランに分ける事により，お互が他へ影響せず独立並行的に処理され，入出力装置と処理装置の使用効率が向上する。

(ii)　入出力装置各1台でマルチ・ジョブ・ストリームをサポートするため，ジョブ実行中1つのジョブ・クラスで入出力装置の占有を行なわない。

(iii)　プライオリティ・ジョブ・スケジュール方式によるジョブの優先権，開始時刻によるジョブの実行開始をサポートする。

(iv)　出力情報により出力ランのスケジューリングをサポートする。

入力リーダは出力ライタとともにジョブ管理プログラムを構成するルーチンであり，常駐入力リーダは専用エリア（入力リーダ・パーティション・グループ）で実行され，非常駐入力リーダはマルチ・ステージ・エリア又は，オペレータ・パーティション・グループ・エリアで実行される。常駐入力リーダはSRTコマンドによりロードされ，STPコマンドを入れない限り主記憶装置上に常駐するが，非常駐入力リーダは，ENDステートメント又はシステム・リーダ装置からのデータ終了を検出した時主記憶装置から消されるので，再び実行するためにはSRTコマンドを入れなければならない。

入力ジョブ・スタック・エリア上でのジョブの状態は次の3種類である。

(i)　実行中の状態：ジョブ・コントロールにより選択され実行が開始されているジョブで，実行が終了すれば入力ジョブ・スタック・エリアより自動的に削除される。

(ii)　実行待ち状態：未だ実行されてなく，しかも実行抑止の状態でないジョブで，入力リーダで登録された直後のジョブはこの状態である。

(iii)　実行抑止状態：未だ実行されてなく，しかも実行が抑止の状態のジョブでHLDコマンド，JOBステートメントのHD指定，開始時刻指定の時にこの状態となる。

出力ライタはジョブの実行結果とシステム出力データを出力ジョブの待合せキューから取り出してラインプリンタおよびカード・せん孔機に出力するルーチンであり，コントロール・プログラムの一部として，ライタ・パーティション・エリアで実行される。このライタ・パーティションは8KBのメモリを必要とする。

d．EDOS−MSOのテーブル

ECR（executive communication region）

バイト	内　　容
0—　1	月（01～12）
2—　3	日（01～31）
4—　5	年（01～99）西暦の下2桁
6—　8	年間通算日（001～366）
9—29	コントロール・プログラムで使用
30—31	オンライン・カタログ・エリアの先頭アドレス
31—207	コントロール・プログラムで使用

PT（program table）

バイト		内　　容
0—8		コントロール・プログラムで使用
9	2	ジョブ・コントロール・ステートメント・フラグ
10—23		コントロール・プログラムで使用
24—27		ESA先頭アドレス
28—49		コントロール・プログラムで使用

OLC（on−line catalog）

バイト	
0—5	ボリューム・シリアル番号
6—7	デバイス・リストの先頭アドレス（AH）
8	使用しない

（最後のエントリの後に"FF"が付加されている。）

ESA(executive storage area)

バイト	内　　容
0—107	コントロール・プログラムで使用。
108—111	プログラム・エリアの最終アドレス
112—115	ジョブ・コントロール・ステートメント・エリアの先頭アドレス
116—183	コントロール・プログラムで使用。

JCSA(job control statement area)

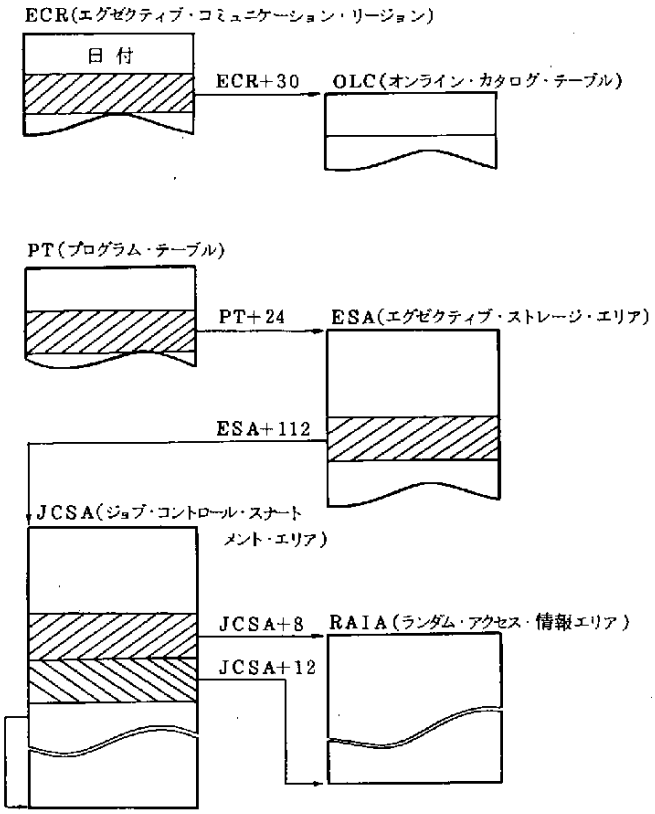
バイト	内　　容	
0- 3	EQUATE情報エリアの先頭アドレス	(コントロール・プログラムで使用)
4- 7	磁気テープ情報エリアの先頭アドレス	
8-11	ランダム・アクセス情報エリアの先頭アドレス	
12-15	DUMP情報エリアの先頭アドレス	(エグゼクティブで使用)
16-19	IPATCH情報エリアの先頭アドレス	(エグゼクティブで使用)
20-23	JCSAの最後＋１のアドレス	

RAIA(random access information area)

バイト	
0- 1	RAIAのサイズ
2- 8	DTFステートメントで指定されたDTF名称 (SFN)
9-10	エクステント・マトリクス・サイズ
11-n+10	エクステント・マトリクス・エリア
n+11-n +54	ファイルＩＤ
n+55-n +56+6x	ボリューム・シリアル番号
n+55+6x +1～	バイト 0—n+54+6x のくりかえし

n：エクステント・マトリクス・エリアのサイズ
x：指定されたボリュームの個数

e．EDOS—MSOテーブル一覧



B．データ交換制御機能

EDOS—MSOではDXC(Data exchange controler)を介して2台の処理装置間で両方向にデータの転送を行なうためのDXCサポート・マクロを備えている。両処理装置間でのデータ転送は，一時点では一方向に限られる。

コントロール・プログラムはDXCを1つの入出力として取扱うので，一般のノンランダム・アクセス装置と同じく処理プログラム中でファイルとして割当てる必要がある。

DXCと一般のノンランダム・アクセス装置との違いは，DXCの場合処理装置内の演算とは非同期に，他の処理装置側でのDXCへのWriteコマンドの実行により割込みが入る。このためユーザ・プログラムでDXCからの割込みを処理するためのDXC割込み処理ルーチンを用意する必要がある。

DXC割込み処理ルーチンでは他の処理装置からの割込みに対して，EXCPまたはEXCPWなどのフィジカルI／Oマクロを用いて，DXCに対するReadを行なう事が必要となる。また必要に応じてフィジカルI／Oを用いて他の処理装置にデータを転送する事ができる。フィジカルI／Oで指定するチャネル・コントロール・ブロック(CCB)は，一般のCCBに16バイトのCCBエクステンションを付加したものを使用する。

1つのDXC割込み処理中に発生した新たなDXC割込み要求は，コントロール・プログラムで記憶し，実行中のDXC割込み処理が終了した時点で割込まれる。

以下にDXCマクロ及びCCBについて述べる。

(1) STDXC(Set address of DXC Routine)

このマクロは，DXCからの割込み処理の準備を行なう。このマクロの実行に先だってDXCの装置割当てが完了していなければならない。このマクロではCCB名称を指定するが，指定されたCCBは，DXC用のCCBであり，DXC割込み処理ルーチンのアドレスがセットされていなければならない。

(2) DXCXT(Exit from User DXC Routine)

STDXCマクロで準備されたDXC割込み処理ルーチンから割込まれた元のアドレスに制御を戻すために，DXC割込み処理ルーチンの出口で発行される。DXC割込み処理ルーチンでは一般にDXC Readが出されるが，これは必ずしも必要とされるものではなく，割込み処理ルーチンを出た後でもよい。但し，頻繁に割込みが発生する処理装置とのデータ交換では，この方式は不適当である。また，相手側からの割込みに対していつまでもReadを出さないと，タイム・オーバーのエラーが発生したり，間違ってWriteを出すとデータの衝突という不都合な事態が生じるので，注意を必要とする。

(3) DXC用CCB(Channel Control Block about DXC)

DXC用のCCBは一般のCCB定義マクロにDXC割込み処理ルーチン名称を記述することによって作成される。この結果展開されるCCBは，56バイトのエリアが割当てられる。表4－3にDXC用のCCBを示す。

表4－3　DXC用CCB

CCB

Byte	Content	
0 - 5	SDN or SFN	○
6	Device class	○
7	User flag	○
8-11	CCW address	○
12-13	control program use	
14-15	AH(assignment half word)	○
16-19	CAR(channel address register)	
20-23	CCR-Ⅱ(channel command reg-Ⅱ)	
24-27	CCR-Ⅰ(channel command reg-Ⅰ)	
28-30	ASR(device status reg)control	
31	SDB of ASR	◎
32-34	Sense byte	◎
35	Executive flag	◎
36-39	CAR of PCI	
40-43	DXC interrupt routine address	○
44-47	Program counter p1 store area	○
48-51	General register 10 store area	○
52-55	General register 11 store area	○

Standard status device byte(SDB)

bit	内　　容	略 称
0	装置要求割込	MRI
1	終了割込	TI
2	Device busy	DBY
3	Control busy	CBY
4	———————— (1)	
5	Secondary Indicator	SI
6	Inoperable	IOP
7	———————— (0)	

Sense byte ※1(32byte)

bit	内　　容
0	———————— (0)
1	Halt I／O
2	PSW off
3	loop test
4	WDI
5	Incorrect length
6	Incompatible
7	Invalid command

Executive flag

bit	内　　容
0	終了ビット
1	異常状態ビット
2	回復不能エラービット
3	ＰＣＩフラグ
4,5	SDV時のCC
6	センス情報そう失
7	NOファイル

C．プログラム間通信機能

HITAC8450内で動作する2つ以上のプログラム間の連絡は，プログラム間通信 機能に関するマクロを用いる事によって可能となる。このマクロをIPC(Inter Program Communication)マクロと呼び，5種類の実行マクロと1種類の定義マクロよりなる。これらのマクロはメイン・タスクによってのみ出される。図4－77にその過程を示す。

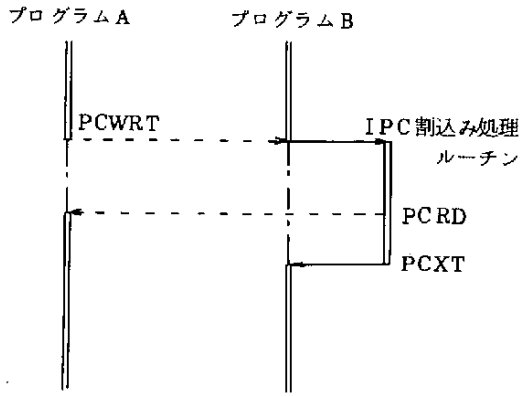


図4－77　プログラム間通信機能

プログラム間通信機能は，"割込み"という形で実現される為，この機能を利用するプログラムは割込み処理ルーチンを用意する必要がある。以下に各マクロについて述べる。

(1) PCSET(Set address of Inter Program Communication Control Blok)

エグゼクティブに対しユーザのプログラム間通信コントロール・ブロック(PCCB)のアドレスを知らせる。このマクロは他のIPCマクロに先だって出さなければならず，しかもIPC割込み処理ルーチン内で出してはならない。

(2) PCWRT(PC Write)

他のプログラムにデータ転送を行なう。この時PCCBには次の情報をセットしておかなければならない。

(i) 送信したい相手のプログラム名

(ii) 送信コード

(iii) 送信データ・アドレスとその長さ(256 Byte 以下)

このマクロを実行すると相手のプログラムにIPC割込みを起し，相手のプログラムが PCRDマクロを実行して始めてデータ転送が行なわれる。同じプログラム名称を持つプログラムが複数個存在する時には，送信する相手のステージ番号とジョブ・クラスを指定する事により，ユニークな相手を決定する事ができる。このマクロの実行される以前に相手のプログラムでもPCSETが出されている事が必要である。なお，IPC割込み処理ルーチン内でPCRD実行後

PCWRTマクロを実行するとプログラム間でデッド・ロックを生じる可能性がある。

(3) PCRD(PC Read)

PCWRTにより他のプログラムから割込まれたプログラムが，IPC割込み処理ルーチンで出す。このマクロが実行されて始めてデータの転送が行なわれる。このマクロを実行するときPCCBには，データ読込みエリアの先頭アドレスとその長さ(256Byte 以下)がセットされていなければならない。なお送信側のデータ長と受信側のデータ長が一致しない場合には，短い方を採用し，差を双方のPCCBに入れ，フラグをセットする。

IPC割込み処理ルーチンに入る時にはPCCBに次のデータがセットされているので，ユーザは参照可能である。

(i) 割込みをかけてきた相手のプログラム名称

(ii) 割込みをかけてきた相手のステージ番号とジョブ・クラス

(iii) 送信されてきたコード

このPCRDマクロにより割込みをかけてきた相手のプログラムの待ち状態は解除される。

(4) PCWT(PC Wait)

PCWRTマクロの後で出され，PCWRTマクロの完了するまでプログラムを待ち状態におく。すなわち最後に出されたPCWRTマクロで指定したPCCBの終了フラグがセットされていればPCWTマクロの次に制御が渡る。

(5) PCXT(Exit from PC Interrupt Routine)

IPC割込み処理ルーチンから出て割込まれたもとのアドレスへコントロールを戻す。このマクロはIPC割込み処理ルーチン以外で出すことはできない。またIPC割込み処理中に別のプログラムからPCWRTマクロがこのプログラムに対して出されると，その割込みは登録され，このPCXTマクロが出たあとサービスされる。

(6) PCCB(Define Inter Program Communication Blok)

PCCBを定義するエリア定義マクロであり，このマクロにより48ByteのPCCBエリアが定義される。このマクロの展開形及びステータス・バイト1，2とフラグ・バイトを表4－4に示す。

表4－4　PCCB

PCCB

Distinate Program name (WRITE)		Stage №・Job №	
Data length	Data address (write)		
Code	Status 1	Status 2	Flag
Distinate Program name (READ)		Stage №・Job №	
Data length	Data address (receive)		
Code	Status 1	Status 2	Flag
IPC interrupt routine address			
Program counter P1 store area			
General register 10 store area			
General register 11 store area			

4 bytes

Status 1

ビット	意味
0	終了フラグ
1	異常終了フラグ
2	相手プログラムが存在しない PCSETが出されてない
3	相手プログラム・ロールアウト中
4	データ・エリアに誤り IPC割込み処理ルーチンで PCRD出ない。
5	PCCBアドレス・エラー 読込みエリアに誤り
6	データ・エリアが長すぎる
7	データ・エリアが短かすぎる

Status 2

ビット	意味
0～7	データ長の差

Flag

ビット	意味
0	最後に指定したPCCBアドレスと異ったPCCBアドレス指定
1～7	未使用

D．キー・イン／アウトシミュレーション機能

あるジョブのコンソール・オペレーションを他のプログラムでコントロールするためと，エグゼクティブと各種の連絡をとるために用いられる機能である。

キー・イン／アウト・シミュレーションを行なうプログラムをイニシェータと呼び，イニシェータによってコントロールされるプログラムをサボーディネータと呼ぶ。コンソール・オペレーションを代行するためには，イニシェータはサボーディネータをロードする必要がある。また，オペレータ・コマンドによるエグゼクティブとの連絡は，キー・イン／アウト・シミュレーションによって擬似オペレータ・コマンドを発行する事により行なわれる。すなわち各種のオペレー

タ・コマンドをユーザ・プログラムから擬似的に実行したり，擬似オペレータ・コマンドによってロードされたプログラムから出されたTYPEメッセージを受け取り，それに対する応答をシミュレートできる機能をもつ。

イニシェータからの擬似オペレータ・コマンドまたは応答等は，SVC命令によってエグゼクティブに知らされ，エグゼクティブはキー・インされたと同様な処理を行なう。サボーディネータでTYPEマクロを実行したり，サボーディネータがコールしたエグゼクティブ・オーバレイからのメッセージが出されたとき，エグゼクティブはそのむねをイニシェータに知らせる。イニシェータは指定されたメッセージ読込みエリアを参照する事により，どのプログラムからどの様なメッセージが出されたかを知る事ができ，さらにそのメッセージに対する応答ができる。

この機能を行なうためにEDOS—MSOは3個のSVCマクロと1個の定義マクロをサポートしている。以下に各マクロについて述べる。

(1) STKEY(Set Address of Key—in／out Simulation Control Block)

エグゼクティブに対してキー・イン／アウト・シミュレーション・コントロール・ブロック(KEYCB)のアドレスを知らせ，イニシェータとなる事を宣言する。このマクロは他のKey—Simマクロに先立って発行されなければならない。また，ユーザのKey—Sim割込みルーチン内で発行する事はできない。

KEYCBの内容を変更したとき，または別のKEYCBに変更しようとするときは，必ずこのマクロを再発行しなければならない。また，KEYCBの内容を変更しようとする時には，変更からマクロの発行までの間にKey—Sim割込みがかからないことを確認する必要がある。

このマクロの実行以前に次の情報をセットしておく必要がある。

(i) メッセージ受取りバッファのアドレス及び長さ

(ii) メッセージ受取りバッファの個数

(iii) Key—Sim割込みルーチンのアドレス

(iv) オプション・フラグ

(2) KEYSM(Execute Key—in／out Simulation)

エグゼクティブに対し，オペレータ・コマンド，応答コマンドのシミュレーションを依頼する。メッセージ・エリアにはメッセージの長さ(1～80)とそれに続くメッセージをセットしておく必要がある。

このマクロの実行後の復帰アドレスは．メッセージが受付けられた時と，拒絶された時で異なる。そのアドレスは次の通りである。

(i) 受付けられた時……KEYSMマクロの次の命令＋4

(ii) 拒絶された時……KEYSMマクロの次の命令

(3) KEYXT(Exit from Key—in／out Simulation Routine)

ユーザのKey—Sim割込みルーチンから出て割込まれたアドレスにコントロールを戻す。Key—Sim割込みルーチンにコントロールが渡るとき，先頭のメッセージ受取りバッファには，タイプ・アウト・メッセージがセットされている。タイプ・アウト・メッセージの長さが，メッセージ受取りバッファの長さより短い時は残りの部分にスペースが入り，長い時はメッセージ受取りバッファの長さ分だけセットされ，残りは無視される。

Key—Sim割込みルーチンではメッセージ受取りバッファの先頭だけ処理すればよい。

Key—Sim割込みが不要な場合には，KEYCBオプション・フラグにその旨セットしておかなければならない。

(4) KEYCB(Define Key—in／out Simulation Control Block)

キー・イン／アウト・シミュレーション・コントロール・ブロックを定義する。KEYCB，エグゼクティブ・フラグ，オプション・フラグの内容を表4—5に示す。

表4—5 KEYCB

KEYCB

Message receive buffer address			
Buffer length	Buffer number	Executive flag	Option flag
Key—Sim interrupt routine address			
Program counter P1 store area			
General register 10 store area			
General register 11 store area			

4 bytes

Exective flag

ビット	意味
0	メッセージ受取りバッファが一杯のためKey—Sim割込み抑止(resetはユーザが行なう)
1〜7	未使用

Option flag

ビット	意味
0〜5	未使用
6	KEYSMマクロによる模擬オペレータコマンドのタイプ・アウト禁止
7	Key—Sim割込みを禁止

4.3.3 NCPサービス・コマンド

ここで述べるサービス・コマンドは，ユーザが基本的なネットワーク・サービスを受けようとする際に必要となる各種のマクロ・コマンドで，定義マクロとしてのDTFNT，ECBST，LSNST，実行マクロとしてのOPEN，CLOSE，GET，PUT，GETN，PUTN，LOADP，CNT，LISTN，POST，SWAITなどがある。これらの実行マクロは，機能上コネクションとデータ授受に関するもの，事

象待ち及び通知に関するもの，プロセスの起動等に関するものに分類することができ，このマクロ・コマンドを，ユーザがアセンブラ・ソース・コーディング上に記述することによって，ユーザ・プロセスとNCPとの会話を行なうためのディスパッチ・モジュールを，ＬＩＮＫ　ＥＤＩＴ時にユーザ・プログラムに付加する事ができる。

A．ネットワーク・ファイル・ディスクリプタ・マクロ（DTFNTマクロ）

DTFNTマクロは，ロジカル・ファイルの特徴を記述し，ファイル処理形式を示し，また，ネットワーク・ファイルで使用されるEDCBT（Extended Data access Control Block Table）に，Port－ID，転送バイト数，パスワード等をセットする。

このマクロは，シリアル・アクセスを必要とするファイルに使用するもので，STARTカードの次で，かつプログラム・ソース・カードの前で定義されていなければならない。DTFNTマクロによって定義されたファイルは，利用方法によって，ネットワーク・プロセス間コミュニケーション用ファイルとしても，また，一般のファイルとしても使用する事ができる。

DTFNTマクロの記述形式は，DTFSRマクロに準拠し，ネットワーク特有のキー・ワードは，NMPID，NYPID，NECBN，NFFLE，NFVOL，NFSTA，NFEND，NPASSの8通りである。

(1)　DTFNTによるネットワーク・ファイル　定義の例

NAME（1～7キャラクタ）	OPERATION	OPERAND	
ファイル名称	DTFNT	BLKSIZE＝n，	
		CAPREC＝NO，	
		DEVICE＝DISC　78，	
		EOFADDR＝名称，	
		IOAREA1＝名称，	
		LABNAME＝名称，	
		RECFORM＝FIXUNB，	
		RECSIZE＝n，	
		TYPEFLE＝INPUT，	
		VERIFY＝YES，	
		NMPID＝0500030500，	……(1)
		NYPID＝0600030601，	……(2)
		NECBN＝ECBN03，	……(3)
		NFFLE＝NETWORKFILE，	……(4)
		NFVOL＝TEST03，	……(5)
		NFSTA＝N，	……(6)
		NFEND＝D，	……(7)
		NPASS＝CCCCCC	……(8)

(2) OPERAND

BLKSIZE ：数字nは，IOAREA1で定義された入出力エリアからのデータ転送最大バイト長を定義し，最大1152バイト（8パケット）まで指定できる。

DEVICE ：このエントリーは，ロジカル・ファイルに割り当てる入出力装置の型を指定する。

IOAREA1 ：データ入出力のエリアの名称を定義する。入出力ルーチンは，このエリアから，または，このエリアにデータを転送する。

RECFORM ：このエントリーは，入出力ファイルのレコードの型を指定する。

NMPID ：プロセス間コミュニケーションを行なうために，全ネットワークに対してグローバルに付けた名前で，コネクションを張ろうとする自系のPort－ID。

NYPID ：コネクションを張ろうとする相手ポートPort－ID。

NECBN ：イベント情報がポストされるエリア（Event Control Block）に付けられる名称。

NFFLE ：自分のプロセス以外で使用するネットワーク・ファイルのファイル名を指定する。NFVOL，NFSTA，NFENDの指定との組み合わせにより，標準FDRを構成する。

NFVOL ：ネットワーク・ファイルのボリューム・シリアル・ナンバーを6桁の英数字で指定する。

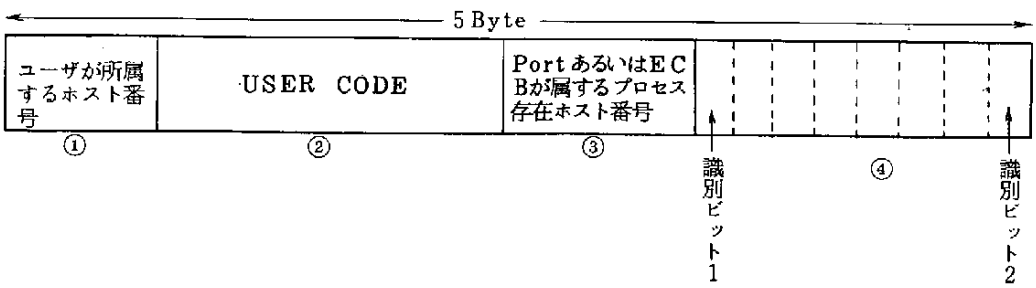
NFSTA ：ネットワーク・ファイルのステータスを指定する。
Temporary＝T，New＝N，Old＝Oのいずれか1文字を指定する。

NFEND ：ネットワーク・ファイルのジョブステップ終了後のファイルの状態を指定する。Catalog＝C，Delete＝D，Pass＝P

NPASS ：プロセス間コミュニケーションのためのパスワードを指定する。

(注) NMPIDからNPASSまでの8個は，実行時に／DTFステートメントによっても与える事ができる。この場合は，／DTFによって与えられた情報が最優先する。

(3) Port－ID及びECB－IDの構成



① ユーザが所属するホスト番号をセットするエリア（1byte）

② ユーザ識別コード・エリア（2byte）

③ Port—ID，あるいはECB—IDが属しているプロセス存在ホスト番号をセットするエリア（1byte）

④ 各種情報識別ビット・エリア（1byte）

識別ビット1：" 0 "の時→Port—ID

" 1 "の時→ECB—ID

識別ビット2：" 0 "の時→Read Port—ID

" 1 "の時→Write Port—ID

(4) 使用例

```
00001  SOKUTE   START (                          S0000100
00002           PRINT P                          S0000200
00003  DISC1    DTFNT B                          CS0000300
00004                  R                         CS0000400
00005                  CAPREC=NO,                CS0000500
00006                  DEVICE=DISC78,            CS0000600
00007                  IOAREA1=BUFFER,           CS0000700
00008                  EOFADDR=TERM,             CS0000800
00009                  LABNAME=LABNAME,          CS0000900
00010                  RECFORM=FIXUNB,           CS0001000
00011                  TYPEFLE=INPUT,            CS0001100
00012                  VERIFY=YES,               CS0001200
00013                  NMPID=0500010500,         CS0001300
00014                  NYPID=0600010601,         CS0001400
00015                  NFFLE=NETWORKFILE01,      CS0001500
00016                  NFVOL=TEST01,             CS0001600
00017                  NECBN=ECBN01,             CS0001700
00018                  NFSTA=N,                  CS0001800
00019                  NFEND=D,                  CS0001900
00020                  NPASS=AAAAAA              S0002000
00145  DISC2    DTFNT BLKSIZE=1152,             CS0002100
00146                  RECSIZE=1152,             CS0002200
00147                  CAPREC=NO,                CS0002300
00148                  DEVICE=DISC78,            CS0002400
00149                  IOAREA1=BUFFER,           CS0002500
00150                  EOFADDR=TERM,             CS0002600
00151                  LABNAME=LABNAME,          CS0002700
00152                  RECFORM=FIXUNB,           CS0002800
00153                  TYPEFLE=INPUT,            CS0002900
00154                  VERIFY=YES,               CS0003000
00155                  NMPID=0500020500,         CS0003100
00156                  NYPID=0600020601,         CS0003200
00157                  NFFLE=NETWORKFILE02,      CS0003300
00158                  NFVOL=TEST02,             CS0003400
00159                  NECBN=ECBN02,             CS0003500
00160                  NFSTA=N,                  CS0003600
00161                  NFEND=D,                  CS0003700
00162                  NPASS=BBBBBB              S0003800
00284  DISC3    DTFNT BLKSIZE=1152,             CS0003900
00285                  RECSIZE=1152,             CS0004000
```

B．コネクションとデータ授受に関するマクロ

他のネットワーク・プロセスとのコミュニケーションのために使用されるマクロで，プロセス間のコネクションの確立，解放を行なうOPEN，CLOSEマクロ，データの授受のためのGET，PUT，GETN，PUTNの各マクロがある。

a．OPENマクロ

OPENマクロは，ユーザがプログラム中で使用するファイルを，使用可能な状態にするもので，ローカル・ファイル以外のネットワーク・コミュニケーション用ファイルに対しては，コネクションの確立要求をNCPに出す。

(1) フォーマット

NAME	OPERATION	OPERAND
記　号 （ブランク）	OPEN	Port 名称1，Port 名称2，…… Port名称n

(2) OPERAND

Port 名称：Port 名称は，ロジカル・ファイルの記号名称で，DTF定義マクロで与えられた名称である。複数個のPort 名称を書くことによって，1つのOPENマクロで16個までのPort 名称を指示できる。

(3) 処　理

ネットワーク用ファイル以外のファイル（ローカル・ファイル）については既存のOPEN処理と同様の処理が行なわれる。

ネットワーク用ファイルについては，ディスパッチ・モジュールが，すでに互いの相手Port－IDを知っている場合のコネクション確立か，既存のコネクションの切り替えの処理かをEDCBTをチェックして判断する。

(4) 展開形

NAME	OPERATION	OPERAND
	CNOP	0，4
NAME	LM	14，15，*+8
	BAL	1，0(15)
	DC	AL4(*+12+4*n)　n：指定Port数
	DC	AL4(IFNT)
	DC	AL4(IFCP)
	DC	AL4(Port 名称1)
	⋮	⋮
	DC	AL4(Port 名称n)
	EXTRN	IFNT
	(EXTRN	IFCP)
	(EXTRN	Port 名称1)
	⋮	⋮
	(EXTRN	Port 名称n)

b．CLOSEマクロ

CLOSEマクロは，ユーザがプログラム中で使用したファイルをクローズ処理するもので，ネットワーク・コミュニケーション用のファイルに対しては，コネクションの解放を行なう。

(1) フォーマット

NAME	OPERATION	OPERAND
記　号 （ブランク）	CLOSE	Port名称1，Prot名称2，…… Port名称n

(2) OPERAND

Port名称：Port名称は，DTF定義マクロで割り当てられたロジカル・ファイルの記号名称である。複数個のPort名称を記述することによって，16個までのPortを1つのCLOSEマクロで指示できる。

(3) 処　理

ローカル・ファイルについては，既存のCLOSEマクロの処理を行なう。

ネットワーク・ファイルについては，ディスパッチ・モジュールでCLOSEに応じたコネクション解放のための情報を作成し，NCPに連絡する。

(4) 展開形

NAME	OPERTION	OPERAND
	CNOP	0，4
NAME	LM	14，15，*+8
	BAL	1，4(15)
	DC	AL4(*+12+4*n)　n：Port数
	DC	AL4(IFNT)
	DC	AL4(IFCP)
	DC	AL4(Port名称1)
	⋮	⋮
	DC	AL4(Port名称n)
	EXTRN	IFNT
	(EXTRN	IFCP)
	(EXTRN	Port名称1)
	⋮	⋮
	(EXTRN	Port名称n)

c．GETNマクロ

プロセス間コミュニケーションにおいて，順編成ファイルのブロック・レベルのアクセス方法で実現させるための読み込みマクロ。

(1) フォーマット

NAME	OPERATION	OPERAND
記　号 (ブランク)	GETN	Port名称，ECB名称，バッファ・アドレス

(2) OPERAND

Port名称：Port名称は，DTFNTマクロで割り当てられたファイルの記号名称。

ECB名称：GETNマクロの実行完了情報がセットされる領域の名称であるとともに，ECB−IDがセットされているエリアの名称である。領域は，ECBSTマクロによって与えられる。

バッファ・アドレス：読み込まれたデータがセットされる領域の名称

(3) 処　理

DTFNTで定義されたブロック長が転送バイト長として使用され，GETNマクロの実行が終了すると，DTFNTで定義されたECB領域には結果コードが，Read Lengthポスト・エリアには，実際の読み込み長がセットされる。

このマクロを実行すると，制御は次の命令に戻されるので，実行の終了はイベント待ちマクロによって知る事ができる。

(4) 展開形

NAME	OPERATION	OPERAND
	CNOP	0，4
NAME	LM	14，15，*+8
	BAL	1，16(15)
	DC	AL4(*+24)
	DC	AL4(IFNT)
	DC	AL4(Port名称)
	DC	AL4(ECB名称)
	DC	AL4(バッファ・アドレス)
	DC	F"00"
	EXTRN	IFNT

d．PUTNマクロ

プロセス間コミュニケーションにおいて，順編成ファイルのブロック・レベルのアクセス方法で実現させるための書き出しマクロ。

(1) フォーマット

NAME	OPERATION	OPERAND
記　号 （ブランク）	PUTN	Port名称，ECB名称，バッファ・アドレス

(2) OPERAND

Port名称：Port名称は，DTFNTマクロで割り当てられたファイルの記号名称。

ECB名称：PUTNマクロの実行完了情報がセットされる領域の名称であるとともに，ECB−IDがセットされているエリアの名称である。領域は，ECBSTマクロによって与えられる。

バッファ・アドレス：DTFNTで定義された転送バイト分だけ，このエリアより他のプロセスに転送する。そのためのデータ・エリアの名称。

(3) 処　理

DTFNTで定義されたブロック長が転送バイト長として使用され，PUTNマクロの実行が終了すると，DTFNTで定義されたECB領域には結果コードがセットされる。

このマクロを実行すると，制御は次の命令に戻されるので，実行の終了はイベント待ちマクロによって知る事ができる。

(4) 展開形

NAME	OPERATION	OPERAND
	CNOP	0，4
NAME	LM	14，15，*+8
	BAL	1，20(15)
	DC	AL4(*+24)
	DC	AL4(IFNT)
	DC	AL4(Port名称)
	DC	AL4(ECB名称)
	DC	AL4(バッファ・アドレス)
	DC	F"00"
	EXTRN	IFNT

e．GETマクロ

GETマクロは，コネクションを介して転送されてきたデータをDTF定義で与えられたエリアに読み込む命令である。

(1) フォーマット

NAME	OPERATION	OPERAND
記　号 （ブランク）	GET	Port名称

(2) OPERAND

Port名称：Port名称は，DTF定義マクロで割り当てられたロジカル・ファイルの記号名称である。

(3) 処　理

ローカル・ファイルに関しては，既存のＧＥＴマクロと同様な処理を行なう。ネットワーク・ファイルに関しては，コネクションを介して転送されてきたデータを，DTFNTエントリーで与えられたIOAREA1のエリアに読み込む。データの授受が完了するまで制御は次に移らない。

(4) 展開形

NAME	OPERATION	OPERAND
	CNOP	0，4
NAME	LM	14，1，*+8
	B	24(15)
	DC	AL4(*+20)
	DC	AL4(IFNT)
	DC	AL4(0)
	DC	AL4(Port名称)
	DC	AL4(IFCP)
	EXTRN	IFNT

f．PUTマクロ

PUTマクロは，コネクションを介してDTF定義で与えられたエリアからデータを転送する書き出し命令である。

(1) フォーマット

NAME	OPERATION	OPERAND
記　号 （ブランク）	PUT	Port名称

(2) OPERAND

Port名称：Port名称は，DTF定義マクロで割り当てられたロジカル・ファイルの記号名称である。

(3) 処　理

ローカル・ファイルに関しては，既存のPUTマクロと同様な処理を行なう。

ネットワーク・ファイルに関しては，DTFNTエントリーで与えられたIOAREA1のエリアから，BLKSIZEの項で与えられたレングス分をコネクションを介して転送する書き出しの処理をする。書き出しが終了するまで制御は次に移らない。

(4) 展開形

NAME	OPERATION	OPERAND
	CNOP	0，4
NAME	LM	14，1，*+8
	B	28(15)
	DC	AL4(*+20)
	DC	AL4(IFNT)
	DC	AL4(0)
	DC	AL4(Port名称)
	DC	AL4(IFCP)
	EXTRN	IFNT

C．プロセスの起動に関するマクロ

HITAC内のネットワーク・プロセスが，他のプロセスを発生させようとするためのマクロがLOADPマクロである。このマクロを実施しようとするネットワーク・プロセスは，プロセスの起動に関する情報を用意しLOADPマクロの後でCNTマクロを利用することにより，NCPに情報を転送することができる。

a．LOADPマクロ

実行プログラム名を与えて，任意のホストにプロセスを発生させ，起動させるマクロである。

(1) フォーマット

NAME	OPERATION	OPERAND
記　号 （ブランク）	LOADP	プログラム名称，発生情報名称，プロセス情報名称，ECB名称1，ECB名称2

(2) OPERAND

プログラム名　　：実行形式のプログラムに付けられる名前で，8文字以内で与える。

発生情報名称　　：発生情報名称は，発生情報が格納されているエリアの名称で，ホスト番号，プロセス名称から構成されている。

プロセス情報名称：プロセス情報名称は，プロセス情報が格納されているエリアに付けられている名称で，ファイル個数，ジョブ情報，ファイル情報等から構成されている。

ECB名称1　　：LOADPマクロによって，プロセスの発生が成功したかどうかをポストするエリアの名称。

ECB名称2　　：発生されたプロセスの実行が，終了した時にその完了情報がポストされるエリアの名称。

(3) 処　理

LOADPマクロの実行により，NCPは発生情報，プロセス情報をもとにして，指定ホストにプロセスを発生させる。発生されたプロセスには，オペランドで与えられたプログラム名が付けられる。

(4) 展開形

NAME	OPERATION	OPERAND
	CNOP	0，4
NAME	LM	14，15，*+8
	BAL	1，32(15)
	DC	AL4(*+32)
	DC	AL4(IFNT)
	DC	AL4（発生情報名称）
	DC	AL4（プロセス情報名称）
	DC	CL8"プログラム名称"
	DC	AL4(ECB名称1)
	DC	AL4(ECB名称2)
	EXTRN	IFNT

b．CNTマクロ

ユーザが，連続した情報をNCPに転送するためのマクロ。

(1) フォーマット

NAME	OPERATION	OPERAND
記　号 （ブランク）	CNT	バッファ・アドレス名称，C or E

(2) OPERND

バッファ・アドレス名称：転送すべき情報がセットされているエリアの名称。

C　　：転送すべきデータがまだ残っている状態を表わす。

E　　：最終転送データであることを示す。

(3) 処　理

ユーザが用意したプロセス起動に関する情報を，NCPに転送する。

(4) 展開形

NAME	OPERATION	OPERAND
	CNOP	0，4
NAME	LM	14，15，*+8
	BAL	1，40(15)
	DC	AL4(*+16)
	DC	AL4(IFNT)
	DC	AL4(バッファ・アドレス名称)
	DC	XL4"00000000" or "FFFFFFFF" ※
	EXTRN	IFNT

※　オペランドで'E'が指定された時にAll"F"コードとなる。

D．事象待ち及び通知に関するマクロ

ネットワークにおいては，複数個のプロセスが互にある種の関連を保ちながら処理を進めていく必要が生じてくる。この過程において，ネットワーク・プロセス間の通知に使用されるのがLISTN，POSTの各マクロであり，事象待ち用に用いられるのがSWAITマクロである。

a．LISTNマクロ

他系のプロセスから，LISTNマクロで指定したPort名称に対してOPENコマンドが発行されたかどうかをチェックするマクロ。

(1) フォーマット

NAME	OPERATION	OPERAND
記　号 （ブランク）	LISTN	Port 名称，ＥＣＢ名称

(2) OPERAND

Port 名称：ＬＩＳＴＮの対象となる，My port－ＩＤがあるポート名称。

ＥＣＢ名称：結果情報がポストされるエリアであるとともに，ECB－ＩＤがセットされているエリアの名称

(3) 処　理

他系のプロセスから，自分のPort に対してＯＰＥＮコマンドが発信されたかどうかをチェックし，他系からの発信があった場合は，Your port－ＩＤをEDCBTのYour port－ＩＤのセット・エリアに格納する。

(4) 展開形

NAME	OPERATION	OPERAND
	CNOP	0，4
NAME	LM	14，15，＊+8
	BAL	1，44(15)
	DC	AL4(＊+16)
	DC	AL4(ＩＦＮＴ)
	DC	AL4(Port 名称)
	DC	AL4(ＥＣＢ名称)
	EXTRN	ＩＦＮＴ

b．POSTマクロ

他のプロセスに，イベント情報を転送するためのマクロ。

(1) フォーマット

NAME	OPERATION	OPERAND
記　号 （ブランク）	POST	相手側のＥＣＢ名称，＂イベント情報＂，ＥＣＢ名称

(2) OPERAND

相手側のＥＣＢ名称 ：相手側のＥＣＢ－ＩＤがセットされているＥＣＢセット・エリアの名称。

イベント情報　　：相手側に転送するイベント情報を４バイトでセットする。

ＥＣＢ名称　　：結果情報及びECB−ＩＤがセットされているエリアの名称。

(3) 処　理

他のプロセスと同期をとるためのイベント情報を転送する。

(4) 展開形

NAME	OPERATION	OPERAND
	CNOP	0，4
NAME	LM	14，15，*+8
	BAL	1，48(15)
	DC	AL4(*+20)
	DC	AL4(IFNT)
	DC	AL4(相手側のECB名称)
	DC	XL4"イベント情報"
	DC	AL4(ECB名称)
	EXTRN	IFNT

c．SWAITマクロ

指定されたECB−ＩＤに対して，いずれかの情報がポストされるまで，待ち状態とするためのマクロ。

(1) フォーマット

NAME	OPERATION	OPERAND
記　号 (ブランク)	SWAIT	ECB指定個数，Post エリア，ECB指定1，ECB指定2， ……ECB指定n

(2) OPERAND

ECB指定個数　：オペランド内のECB指定の個数(最大８個まで)を示す。

Post エリア　　：待ち状態が解かれた結果情報がセットされるエリアの名称(１ワード確保すること)。

ECB指定n　　：情報がポストされるまで待ち状態としたいECB−ＩＤが，セットされているエリアの名称(最大８個まで指定可能)。

(3) Post エリアの結果情報のセットのされ方

＜Post エリア名称＞(バウンダリ調整がなされていること)

			1	0	0	1	0	0	0	1

SWAITマクロで，ECB指定が8個セットされており，ポスト情報がECB指定1と，ECB指定5及びECB指定8に対してあり，この3つのポストによりSWAITが解除になった場合には，Postエリアの下位1バイトに前頁のようなビット構成で結果情報がセットされる。

(4) 処　理

指定されたECB—IDに対して，いずれかの情報がポストされるまで待ち状態とする。指定したいずれかのECBに情報がポストされると，ポストされたECBをビット単位でPostエリアに知らせ，待ち状態は解除となる。

他のプロセスからのイベント通知を待つ場合やマクロの実行完了をチェックするためなどに使用する。

(5) 展開形

NAME	OPERATION	OPERAND
	CNOP	0，4
NAME	LM	14，15，*+8
	BAL	1，56(15)
	DC	AL4(*+16+4*n)　n：ECB指定個数
	DC	AL4(IFNT)
	DC	F"ECB指定個数"
	DC	AL4(Postエリア)
	DC	AL4(ECB指定1)
	⋮	
	DC	AL4(ECB指定n)
	EXTRN	IFNT

E．データ・セットに関するマクロ

ユーザが必要とするデータ及び，他のマクロを実行する際に必要となるデータをセットするのがここで述べるLSNSTマクロ，およびECBSTマクロである。

a．LSNSTマクロ

コネクション切り替えのために必要となるPort—IDをセットするマクロ。

(1) フォーマット

NAME	OPERATION	OPERAND
記　号 (ブランク)	LSNST	Port名称，"LSNSET定義Port—ID"

(2) OPERAND

Port 名称　　：定義したPort－IDをセットすべきPort 名称。

LSNSET定義：ユーザがセットしようとするPort－IDを10桁の16進数で表わす。
Port－ID

(3) 処　理

コネクションの切り替えに必要となるMy port－ID 2を，EDCBTのセット・エリアにセットし，S.Lフラグを ONとする。

(4) 展開形

NAME	OPERATION	OPERAND
	CNOP	0，4
	LM	14，15，＊+8
	BAL	1，64(15)
	DC	AL4(＊+20)
	DC	AL4(IFNT)
	DC	AL4(Port名称)
	DC	XL5"LSNSET定義Port－ID"
	DC	XL3"000000"
	EXTRN	IFNT

b．ECBSTマクロ

ECB－IDを定義するマクロ

(1) フォーマット

NAME	OPERATION	OPERAND
記　号 (必ず指定)	ECBST	"セットECB－ID"

(2) OPERAND

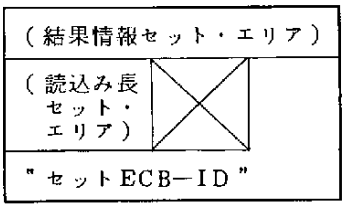
セットECB－ID：セットするECB－IDを10桁の16進数で記述する。

(3) 処　理

オペランドの項で指定されたECB－IDをエリアにセットする。

イベントに関係のあるマクロを使用する場合には，必ずこのマクロでECB－IDをセットしておかなければならない。

NAME

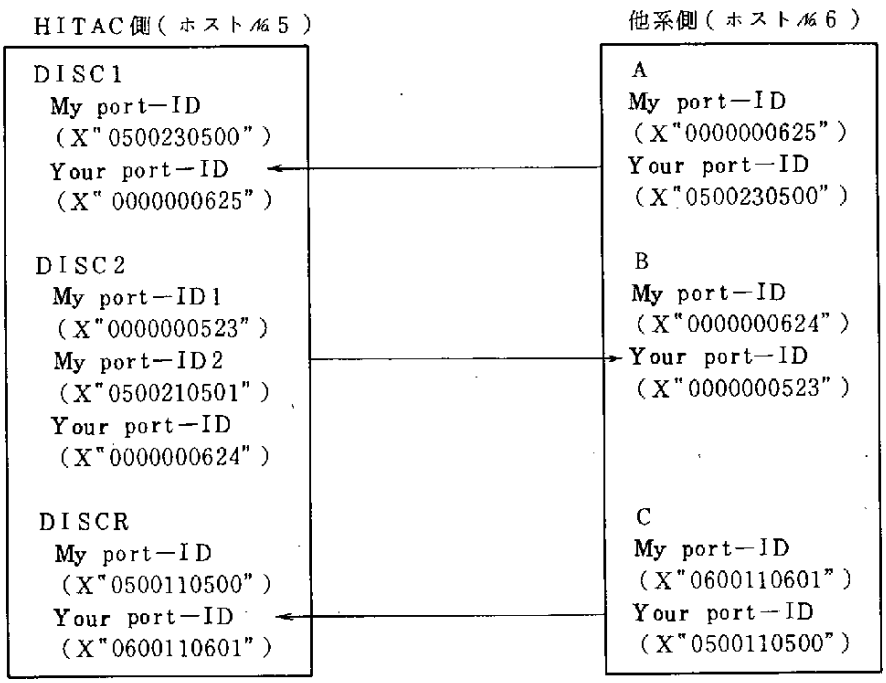


(4) 展開形

NAME	OPERATION	OPERAND
	CNOP	0，4
※NAME	DC	XL4"00000000"
	DC	XL2"0000"
	DC	XL1"00"
	DC	XL5"セットECB−ID"

※必ず指定する

F．マクロの一般的な使用例



簡単なネットワーク間のコミュニケーション・プログラムの例を示す。この例では，DISC2のファイルに対してYour port−IDを他系から読み込み（⑩，⑪），My port−ID2 をセットする（⑥）ことによって，オープンの切り替え（⑦）を行なっている。そして他系のプロセスにOUTPUTエリアから800バイトのデータを転送（⑧）している。ここまでのコミュニケーショ

ンが他系もともに完了しているかどうか同期をとるために，ポスト情報を受けるため待ち状態（⑨）をつくる。次に，DISC1のファイルをオープン（⑩）してコネクションを確立する。このファイルのINPUTエリアに最大1152バイトまでデータを読み込む（⑪）。ここまでの処理が終了したことを他系のプロセスにポストする。（⑫）。今まで使用してきたDISC1，DISC2のファイルに対してコネクションの解放を行なう（⑬）。つづいて，DISCRのファイルをオープンし他系とコネクションを持つ（⑭）。GETNマクロを利用してBUFFRというエリアにデータを読み込み（⑮），データの読み込みが完了したかどうかをチェックする（⑯）。データ読み込みの終了をチェックした後，DISCRのファイルをクローズし，コネクションの解放を行なう。各マクロおよびデータ授受のためのエリアを確保する（⑱，⑲）。

```
TEST     START 0
*************************************
         PRINT NOGEN
DISC1    DTFNT BLKSIZE=1152,
               DEVICE=DISC78,
               IOAREA1=DISCA,
               TYPEFLE=INPUT,
               CAPREC=NO,
               RECFORM=FIXUNB,
               RECSIZE=800,
               EOFADDR=ABC,
               VERIFY=YES,
               LABNAME=LABN1,
               NMPID=0500230500,
               NYPID=0000000625,
               NPASS=AAAAAA
*************************************
DISC2    DTFNT BLKSIZE=800,
               DEVICE=DISC78,
               IOAREA1=DISCB,
               TYPEFLE=OUTPUT,
               CAPREC=NO,
               RECFORM=FIXUNB,
               RECSIZE=800,
               EOFADDR=ABC,
               VERIFY=YES,
               LABNAME=LABN2,
               NMPID=0000000523,
               NYPID=0000000604,
               NPASS=BBBBBB

DISCR    DTFNT BLKSIZE=1152,
               DEVICE=DISC78,
               IOAREA1=BUFFR,
               TYPEFLE=INPUT,
               CAPREC=NO,
               RECSIZE=1152,
               RECFORM=FIXUNB,
               LABNAME=LABNAMER,
               EOFADDR=TERM,
               VERIFY=YES,
               NMPID=0500110500,
               NYPID=0600110601,
               NFCHN=FCHNNR,
               NFFLE=NETWORKFILR,
               NFVOL=TEST01,
               NFSTA=N,
               NFEND=D,
               NPASS=BBBBBB
         DTFEN
```

①

```
**********************************
BEGIN    BALR  10,0                    ..... (2)
         USING *,10,11
BASEADR  L     11,BASEAD                ..... (3)

LISTN    LISTN DISC2,LISNAREA           ..... (4)
**********************************
SWAIT    SWAIT 1,LISNAREA,SWECB1        ..... (5)


LSNSET   LSNST DISC2,'0500210501'       ..... (6)
**********************************
OPEN2    OPEN DISC2                     ..... (7)

**********************************
*****    WRITE(PUT)
WRITE2   PUT  DISC2                     ..... (8)


**********************************
SWAIT2   SWAIT 1,SWPOST2,SWECB2         ..... (9)


**********************************
OPEN1    OPEN DISC1                     ..... (10)
**********************************
*****    READ1(GET1)
READ1    GET  DISC1                     ..... (11)


**********************************
POST1    POST  YOURECB,'99999999',MYECBID ..... (12)


**********************************
CLOSE12  CLOSE DISC1,DISC2              ..... (13)


READER   OPEN DISCR                     ..... (14)


READL    GETN  DISCR,ECBR,BUFFR         ..... (15)

         SWAIT 1,POSTR,ECBR             ..... (16)

         CLOSE DISCR                    ..... (17)
**********************************
ABC      TERM
**********************************
LISNAREA ECBST '5500000580'             ..... (18)

SWPOST2  DC    2F'0'
SWECB1   ECBST '5500000580'             ..... (19)


         END   BEGIN                    ..... (20)
```

各部の説明

① DTFNTマクロによって転送バイト長，バッファエリア，Port—IDなどの定義を行なう。

②，③ ベースの設定

④ LISTNマクロによりYour port—IDを他系のプロセスより受け取る。

⑤ LISTN処理が完了するまで待ち状態をつくる。

⑥ LSNSTマクロにより，My port—ID2をセットする。

⑦ DISC2のファイルをオープンし，他系のプロセスとコネクションを確立する。

⑧ DISC2のOUTPUTエリアより800バイトのデータを転送する。

⑨ 他系のプロセスと同期をとるために，待ち状態をつくる。

⑩ DISC1のファイルをオープンし，他系のプロセスとコネクションを確立する。

⑪ DISC1のファイルのINPUTエリアにデータを読み込む。最大1152バイトまで読み込む事が可能である（実際の読み込み長はEDCBTにセットされる）。

⑫ 他系のプロセスと同期をとるために，イベント情報を送る。

⑬ DISC1，DISC2のファイルをクローズし，コネクションの解放を行なう。

⑭ 更にDISCRのファイルをオープンし，他系のプロセスとコネクションを確立する。

⑮ GETNマクロによりBUFFRエリアにデータを読み込む。
実際の読み込み長は，EDCBTに結果情報とともにセットされる。

⑯ GETNマクロが完了するまで，待ち状態をつくる。

⑰ DISCRのファイルをクローズし，コネクションの解放を行なう。

⑱ ECBSTマクロによりLISTNで使用するECB—IDおよび結果情報ポスト・エリアを確保する。

⑲ 各マクロで使用するエリアの確保

⑳ ユーザ・プロセスのプログラムの終了

4.3.4 H—NCPのソフトウェア

H—NCPは，HITAC内のプロセスと会話をし，サブネットへメッセージ転送を行なうNCPと，プロセスにバインドされてユーザのサービス・コマンドを処理するディスパッチ・モジュールに大別される。さらにNCPは，NCP機能の主要な部分を処理するメイン・モジュールと，メイン・モジュールを補助するサブ・モジュールから構成される。メイン・モジュールはその処理の発生する条件から割込み処理モジュールと非割込み処理モジュールに区別できる。図4—78に各モジュールの構成を示す。

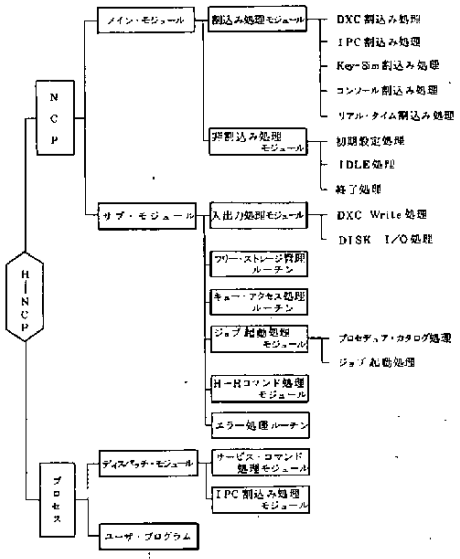


図4－78　H-NCPのソフトウェア構成

A．メイン・モジュール

このモジュールはNCPの基本的なモジュールであり，IMP，プロセス・スーパバイザからの連絡により起動される割込み処理モジュールと，IMP，プロセスへのデータ転送のスケジュールを行なう非割込み処理モジュールからなる。メイン・モジュールは各処理を行なう時，必要に応じてサブ・モジュールを呼び出す事がある。

a．割込み処理モジュール

IMPからの装置要求割込みによるDXC割込み，プロセスからのプログラム間通信によるIPC割込み，サボーディネータによるKey－Sim割込み，オペレータ・コマンドによるコンソール割込み，実時間連絡によるリアルタイム割込みの5種の割込み処理モジュールから構成される。各割込みは非同期に発生し，それらのコントロールは各割込み処理モジュールの入口でスケジュールされる。以下に各割込み処理モジュールの機能及び処理概要について述べる。

(1)　DXC割込み処理モジュール（DXCINT）

IMPからアダプタを経由して送られてくるデータは，DXC割込みをともなってNCPに連絡される。DXC割込みモジュールでは，このデータを受信し，解析を行なう。その後プロセスへの連絡のためのデータを作成し，ＩＰＣ出力キューにキュー・オンを行なう事や，

IMPへのデータを作成し，IMP出力キューにキュー・オンを行なう事がその主な機能である。

この割込み処理モジュールは，IMPとの間のIMP—HOSTプロトコルの処理及び他系NCPとの間のHOST—HOSTプロトコルの処理を行なう中核となるモジュールである。

IMPからの非同期に発せられたデータによって起動される本モジュールでは割込み手順を実施し，IMPからの制御情報を入手するために受信用マクロを発行する。その後制御情報を解析し，通常メッセージであれば，さらにデータ受信のための受信用マクロを発行する。制御情報が複数セグメント受入れ準備完了またはメッセージ再要求の場合には，このモジュールの中で送信用マクロを発行する。さらに，制御情報およびその後送られてくるメッセージによって各種の処理を行なう。これらの処理が，HITAC内のプロセスへの連絡を伴なうものであればIPC出力キューへのキュー・オン，IMPへの応答やメッセージを要求するものであればIMP出力キューへのキュー・オンを行なう。又データ受信完了情報の場合には，receiveキューからキュー・オフの処理を行なう事がある。これらの処理が終了すると，割込み終了手順を行ない，割込まれた元のアドレスに制御をもどすためのマクロを発行する。

(2) IPC割込み処理モジュール（IPCINT）

プロセスからIPCマクロによって送られてくるデータは，IPC割込みとしてNCPに連絡される。このモジュールでは，プロセスからのデータを受信し，IMPへの出力データを作成し，IMP出力キューにキュー・オンを行なう。IPC割込み処理モジュールの主な機能は，HOST—HOSTプロトコル処理と，NCPと会話するプロセスの制御である。

プロセスからのデータによって起動される本モジュールは，割込み手順を実施し，プロセスからのデータを受信するための受信マクロを発行する。このデータのうち，他プロセスへのコネクションを通じたデータ転送を要求するものに関しては，さらに複数メッセージを送信してくる事があるが，1度の割込みに対しては1度の受信マクロを発行するだけであり，複数メッセージに対しては，割込み処理モジュールを抜け出し次のデータ転送を待つ形態をとる。このようにして受信されたデータは，プロセス制御用の処理が行なわれ，さらにIMPへの出力キューにキュー・オンされる。各種処理が終了すると，割込み終了手順を行ない，割込まれた元のアドレスに制御をもどすためのマクロを発行する。

(3) Key—Sim割込み処理モジュール（KSMINT）

Key—Sim割込み処理モジュールは，NCPがエグゼクティブにキー・イン／アウト・シミュレーションを要求した時に起動される。このモジュールでは，シミュレートされたオペレータ・コマンドを解析し，IMP出力キューまたはIPC出力キューにキュー・オンする。このモジュールの機能は，HOST—HOSTプロトコルのうちのジョブの起動，終了に関する処

理と，起動したジョブの制御である。

エグゼクティブからキー・イン／アウト・シミュレーションを通じて割込まれたデータは，本モジュールで解析され，ジョブ・起動，終了の処理が行なわれる。また，必要に応じて本モジュールの中でキー・イン／アウト・シミュレーションを要求するマクロを発行する。その後ジョブ起動又は終了の処理が完了すると，起動，終了の要求を出したプロセスへその状態をポストする為に他系プロセスの場合にはIMP出力キューへ，自系プロセスの場合にはIPC出力キューへキュー・オンする。この割込み処理モジュールでも，割込みの入口，出口で，割込み手順および割込み終了手順を実施し，終了手順が終ると，割込まれた元のアドレスに制御をもどす為のマクロを発行する。

(4) コンソール割込み処理モジュール（CNSINT）

コンソール・オペレータ・コマンド（E—INT）を発行した時起動されるモジュールで，NCPの終了手順，強制終了及びテストモードの指定等の処理を行なう。本モジュールは，プロセス間コミュニケーションに直接作用する性質のものではなく，NCPに対する制御を主な機能とする。すなわち，コンソール・コマンドによって，NCPの終了，テスト・モードの設定等を行ない，NCPの状態を特殊な状態に移行させる時に使用される。

このモジュールに，コンソール・コマンドによる割込みが入ると，コンソール，出力マクロによって，何を要求しているのかをオペレータに質問する。オペレータのコマンドによって上記の処理を行ない，割込まれた元のアドレスに制御をもどす。

(5) リアルタイム割込み処理モジュール（RTTINT）

NCPがタスクに対する実時間割込みの値をセットすると，実時間経過後このモジュールに制御が渡る。すなわち，時間監視の機能および処理を行なう割込みモジュールである。この時間設定の間隔は1～2000秒までの間であるのでI／Oの時間監視用としては不適当であるが，かなり長い時間を経過した後のNCPの状態をチェックしたり，テスト・モードの際の経時時計として利用される。このモジュールではプロトコルに関する処理はほとんど行なわれないが，NCPのロックアウトを防止するために使用されることがある。ここで述べるロックアウトとは，他の仕事があるにもかかわらず，NCPが永久的な待ち状態に入ることを言い，各種割込みのタイミングで非常に低い確率ではあるが生じる事がある。なおこの割込み処理ルーチンでも処理終了後に割込まれた元のアドレスへ制御をもどす。

b．非割込み処理モジュール

非割込み処理モジュールは割込みモード以外で処理されるモジュールで，NCP起動時のイニシャライズ，各出力キューの取出し及び出力，ＮＣＰ終了手順の処理の各モジュールから構成される。非割込み処理モジュールは，その処理中に割込みが発生すると待ち状態となり，割込み処理が全て完了した時再開する。以下に非割込み処理モジュールの機能及び処理概要に

ついて述べる。

(1) 初期設定処理モジュール(HNSTR)

NCPが起動された時，装置割当て，ファイルのイニシャルライズを行ない，さらに，各種割込み処理ルーチンのエグゼクティブへの連絡，テーブルのイニシャライズを行なう。最後にHOST—IMPプロトコルの初期設定手順を実施する。その後IMPから初期設定指令を待つためにアイドル状態に入る。

(2) IDLE処理モジュール(IDLEP)

NCPは初期設定処理及び割込み処理が終了するとIDLE処理モジュールに制御を渡す。IDLE処理モジュールでは，IMP出力キュー及び，IPC出力キューを調べ，キューがある場合には，IMP，プロセスにデータ転送を行なう。全てのキューがなくなった時には待ちタスクとなり，他のジョブに制御が移行する。この処理モジュールは，割込みオリエンティドに動作するNCPの中で，割込み処理終了後のNCPスケジューリングの機能を果し，非割込み処理モジュールの中核をなす部分である。

(3) 終了処理モジュール(ENDS)

NCPが正規終了を行なった後，終了状態を表示するモジュールであり，NCPをネットワークから切り離すか否かをオペレータに連絡する機能を持つ。オペレータの応答によりNCPを終了させたり，初期手順に戻したりする事が可能である。

B サブ・モジュール

サブ・モジュールはメイン・モジュールから呼ばれるサブ・ルーチンの形態を持ち，サブ・モジュールはその機能実現のために，シリアル・リューザブル・ルーチンと通常のサブ・ルーチンに分類でき，クローズドな各種の処理を行なう。

a. シリアル・リューザブル・ルーチン

このルーチンは割込み処理，非割込み処理から頻繁に呼ばれるサブ・ルーチンで，キュー操作，フリー・ストレージの管理等を行なう。NCP機能実現のためにこれらのルーチンがクロス・コールされる事があるため，メモリおよびテーブル類の保護のために，あるモジュールがこのルーチンを呼んでいる時には，他のモジュールが呼べない様なルーチン構成を持つ。すなわち，処理にプロテクションを設けているルーチンである。これらのルーチンはシリアル・リューザブルなため，他のリューザブル・ルーチンから呼ばれる事はない。その種類は，フリー・ストレージ管理ルーチン，キュー・アクセス処理ルーチン，プロセデュア・カタログ処理ルーチンおよびプロセス・ポインター操作ルーチンの4種に限定される。以下に各ルーチンの機能および概要を述べる。

(1) フリー・ストレージ管理ルーチン(BLKGTFR)

NCPはセグメント用バッファ，制御用バッファ，イベント用バッファを持ち，各モジュー

ルはそのバッファを要求したり開放したりする。これらのフリー・ストレージを統一的に管理し，バッファのアロケート，フリーを行なうルーチンが，フリー・ストレージ管理ルーチンである。このルーチンではメイン・モジュールの指定により，バッファの選択を行ない，アロケート，フリーの処理をする。また，その処理の結果を呼ばれたモジュールにポストする役割を持つ。

(2) キュー・アクセス処理ルーチン（QUEACCSS）

IMPに対するIMP出力キュー，プロセスに対するIPC出力キュー，NCP内で利用するRECキューの3種のキュー操作を行なうルーチンである。このルーチンは，メイン・モジュールの要求に従ってキューへの登録，キューからの取出しの処理を行なう。また，処理終了前に呼び出されたモジュールに処理の結果をポストする。要求を出すモジュールは，取扱うキュー及び処理形式（キュー・オン，オフ）を指定する。

(3) プロセデュア・カタログ処理ルーチン（PRCCATR）

NCPがジョブを起動する場合に用いられるルーチンで，自系プロセス，他系プロセスの要求によって呼び出される。すなわち割込み処理モジュールのうちDXC割込み及びIPC割込み処理によって要求が発生する。このジョブ起動に際してはHOST—HOSTプロトコルで定められたコマンドによって各割込み処理モジュールがジョブ起動に必要なデータを作成し，プロセデュア・カタログ処理ルーチンを呼び出す。このルーチンが処理されるとその処理結果が呼び出されたモジュールにポストされる。

(4) プロセス・ポインタ操作ルーチン

HITAC内にあるプロセスは，NCPのプロセス管理テーブルによって管理されているが，その時の管理テーブル内の位置を計算するルーチンである。このルーチンは，プロセスの持つジョブ・クラス，ステージ番号よりポインターを求める機能及び，ポインタより，ジョブ・クラス，ステージ番号を求める機能の2つからなり，メイン・モジュールから呼ばれる。このルーチンの処理が終了するとその結果を呼び出されたモジュールにポストする。

b．通常のサブ・ルーチン

通常のサブ・ルーチンは，メイン・モジュールからクロス・コールされる事のないサブ・ルーチンで，入出力処理を扱うDXC Writeルーチン，DISK・I／O処理ルーチン，HOST－HOST間のコマンドを処理するH—Hコマンド処理モジュール，エラー処理ルーチン等から構成される。特にH—Hコマンド処理モジュールは，各種H—Hコマンド処理のためのサブ・ルーチンより構成される。以下に通常のサブ・ルーチンの主なものについて述べる。

(1) DXC Write処理ルーチン（DXCWRT）

IMPへ出力するデータをキュー・オフし，出力マクロを出すルーチンで，出力するメッセージのタイプによって，コマンド・チェインを行なう事がある。出力終了後メッセージのタ

イプに従って処理し，H—Hコマンドの場合にはRECキュー（レシーブ・キュー）にキュー・オンを行なう。

(2) H—Hコマンド処理モジュール

IMPまたはプロセスからのメッセージで，H—Hコマンドの処理が必要な時呼び出されるルーチンの集合であり，多種のクローズドなサブ・ルーチンで構成される。その主なものは，コネクションの確立，開放のためのサブ・ルーチン，イベント・ポストのためのサブ・ルーチン，コネクション上のデータ転送処理のサブ・ルーチン，ジョブ起動終了のためのサブ・ルーチンである。

これらの処理ルーチンはDXC割込み処理，IPC割込み処理，Key—Sim割込み処理から呼ばれ，コネクション・テーブル，プロセス・テーブルへのアクセスを行なう。

(3) エラー処理ルーチン（ERPRT）

IMPとのデータ転送時のI／Oエラー，ディスク・アクセス時のエラー，NCP初期設定手順上のエラー，HOST—HOSTプロトコル，HOST—IMPプロトコル上のエラー，プロセスとの会話時のエラー，フリー・ストレージ及びキューへのアクセス中のエラー等の各種のエラーが生じた時呼び出されるルーチンで，警告メッセージをコンソール・タイプライタに出力する。その後，エラーの種類に従ってNCPを終了手順に導くかまたは，呼び出された元のアドレスに制御を渡す。

C．ディスパッチ・モジュール

ディスパッチ・モジュールは，ユーザ・プロセスにLINKの時点で付加されるモジュールで，モジュールにとって必要なデータは，ネットワーク・サービス・コマンドから受け取る。このディスパッチ・モジュールは，大きく分けると，サービス・コマンド解析ルーチン，割り込み待ち状態作成ルーチン，割り込み処理ルーチンの3つからなり，ネットワーク・サービス・コマンドから得たデータを，サービスコマンド解析ルーチンでデータ・チェックし，NCPに転送すべきデータかどうかを判断し，必要データをNCPに送信する。データをNCPに送信した後は，割り込み待ち状態作成ルーチンに入り，NCPからの応答を持つ。NCPから，送信したデータに対する返答が送られて来ると，割り込み処理ルーチンでそのデータを解析し事後処理を行なった後，再びユーザ・プロセスに制御を戻す。

このようにして，ディスパッチ・モジュールは，ユーザ・プロセスが他系プロセスとのコミュニケーションを行なう場合に，ユーザ・プロセスとNCPとの中継ぎ的役割を果すモジュールである。

ディスパッチ・モジュールは，大きく分けて次の3つの機能から構成されている。

(1) サービス・コマンド解析ルーチン

(2) 割り込み待ち状態（アイドル状態）作成ルーチン

(3) 割り込み処理ルーチン

サービス・コマンド解析ルーチン及び割り込み処理の各ルーチンは，それぞれサービス・コマンドに対応した処理が行なわれるようになっており機能構成は図4－79のようになっている。

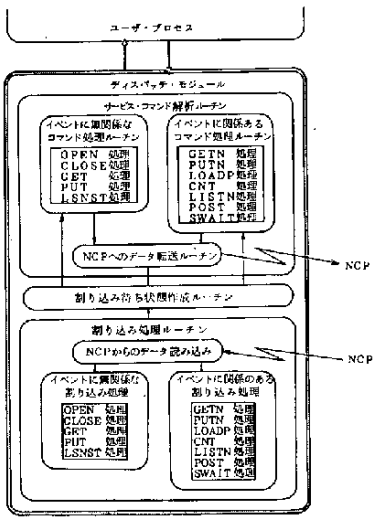


図4－79 ディスパッチ・モジュールの構成

ディスパッチ・モジュールにおけるデータの授受は，ユーザ・プロセスから連絡マクロの展開形としてサブルーチン・コールの形で行なわれる場合と，NCPとの間でIPC（Inter Program Communication）マクロを使用してのデータの授受という2通りの方法が行なわれる。

NCPとのデータの授受は，ディスパッチ・モジュールよりPCWRTマクロによってNCP に図4－80の形でデータが転送され，それに対してNCPから受信メッセージとして同様なフォーマットで返ってくる。従って，ディスパッチ・モジュールは，(1)～(3)の各処理ルーチンで，これらのデータに対してイベントに関係ある処理が必要であるかどうかのチェックをし，コントロールをユーザ・プロセスに戻すべきかどうか，また，イベント情報をテーブルにセットすべきどうかの判断を下さねばならない。

(イ)	(ロ)	(ハ)	(ニ)
148 $(0094)_{16}$	コマンド コ ー ド	INFO.	データ本文

図4－80　ディスパッチ・モジュールとNCP間のデータ授受フォーマット

(イ)　メッセージ長セット（2バイト）

NCPとのデータ授受メッセージ・サイズがセットされているエリアで，常に$(148)_{10}$がセットされている。

(ロ)　コマンド・コード・セット（1バイト）

各種処理モジュールとマッチしたコマンド・コードがセットされるエリア。

(ハ)　情報セット（1バイト）

各コマンドに対応して情報がセットされているエリア

(ニ)　データ本文（144バイト）

転送したり，転送されてくるデータ本文がセットされているエリアで，1度に転送できる最大データ長は144バイトである。従って144バイト以上のデータの授受は，これらの形式に従って何回かデータの転送が行なわれることになる。

a．サービス・コマンド解析ルーチン

このサービス・コマンド解析ルーチンは，ユーザ・プロセスが使用したマクロより，必要データを受け取り解析処理をする。サービス・コマンドには，イベントに関係のあるコマンド（例えばGETN，PUTNなど）と，イベントに無関係なコマンド（例えば，OPEN，CLOSEなど）があり，ルーチンは，受け取ったデータを解析し，ネットワークに関するものかどうかの判断をし，必要ならばNCPへの転送データを作成して送信する。また，ネットワークに関するファイルであるか，ローカルなファイルであるかの判断がなされるのもこのルーチンにおいてであり，ディスパッチ・ルーチンの窓口的役割を果している。

b．割り込み待ち状態作成ルーチン

このルーチンは，サービス・コマンド解析ルーチンで判断された結果，セットされたフラグを元に，待ち状態を作成するルーチンで，待ち状態が解除になるのは，NCPからの割り込みがあった時である。

c．割り込み処理ルーチン

このルーチンは，NCPから送られてくるデータを解析し，その解析結果に対処した各処理を行なうルーチンである。NCPから転送されてくるデータには，サービス・コマンド解析ルーチンから送られたデータに対する受信メッセージであるコマンド受信データと，NCPで処理された結果が送られてくるメッセージ本文データとの2種類がある。このメッセージ本文デ

ータは，更にイベントに関係のあるデータと，イベントには無関係なデータとに分類され，イベントに関係のあるデータに対しては，イベント・ポスト処理を行ない，イベントには関係のないデータに対しては，待ち状態の解除をセットしてユーザ・プロセスに制御を移す。

4.3.5 H—NCPの制御テーブル

NCPの制御テーブルはプロセスの制御を行なうプロセス・テーブル及び，コネクションの制御を行なうコネクション・テーブルを主体とし，その他にNCPのステートを表わすテーブルや，フリー・ストレージ・キューの状態を表わすテーブルがある。

ディスパッチ・モジュールのテーブルは，ユーザ・プログラムにバインドされる拡張ファイル制御テーブルと，ディスパッチ・モジュールが持つイベント制御用のテーブル，コマンド制御テーブル等である。

これらのテーブルは，各モジュールの処理中に参照，登録，削除が行なわれる。上記のテーブル以外にフリー・ストレージにあって，各種のデータ処理のために用いられる領域があるが，これらの領域の使われ方はほぼ一定したフォーマットを形成している。

以下にH—NCPの制御テーブル及び，フリー・ストレージのデータ・フォーマットについて述べる。

A．NCPの制御テーブル

NCPの制御テーブルは，ステート・テーブル，プロセス・テーブル，コネクション・テーブル，キュー管理テーブル，フリー・ストレージ管理テーブル，ジョブ起動用テーブルからなる。これらのテーブルは，NCPのメインモジュール，サブ・モジュールによってアクセスされる。各テーブルは固定長であり，NCPの起動時に初期化される。

a．ステート・テーブル

NCPの状態，割込み処理，リューザブル処理ルーチンの使用状況，キューの状態，特殊なサブ・ルーチンの使用状況などを管理するためのテーブルで，各割込み処理モジュールおよびIDLE処理モジュール等で参照され，NCPのフロー制御のために用いられる。図4—81にステート・テーブルを示す。

NSTATE1

Byte	内　　容	ラベル
0	NCP status	STATE
1	IMP status	
2	NCP PSW	
3	IMP PSW	
4	NCP接続中止要求	
5	IMP接続中止指令	
6	NCP切断完了	
7	IMP切断完了	
8	交信一時停止	
9	initial設定回数	
10	未　使　用	
11	未　使　用	
12	IMP 出力キューロック	DXCLOK
13	ロック用ワーク	
14	DXC Write 用ゲート	GATE
15	ゲート用ワーク	

NSTATE2

Byte	内　　容	ラベル
0	IMP用緊急出力フラグ	DQPRI
1	IMP出力キュー個数	DQFLG
2	IPC出力キュー個数	PQFLG
3	DXC割込み処理中	DXCFLG
4	IPC割込み処理中	IPCFLG
5	Key—Sim割込み処理中	KSMFLG
6	タスク・コントローラ動作中	TSKFLG
7	コンソール割込み処理中	CNSFLG
8	実時間割込み処理中	RTTFLG
9	キュー・アクセス・ルーチン使用中	QAFLG
10	フリー・ストレージ管理ルーチン使用中	FSFLG
11	ターミネータ使用中	TRFLG
12	プロセス・ポインタ操作ルーチン使用中	SAFLG
13	プロセデュア・カタログルーチン使用中	CTFLG
14	DXC Write ルーチン使用中	DXWFLG
15	未　使　用	

図4—81　ステート・テーブル

b．プロセス・テーブル

NCPと会話をしているプロセス，NCPが起動したプロセスを管理するテーブルで，ステージ・クラス毎のエントリーを有する。現在NCPが管理するプロセスは0A，0B，0C，1B，1Cの5種類のステージ・クラスであり，1つのステージ・クラスについて図4—82の様な管理テーブルをもつ。

このテーブルへのプロセスの登録は，プロセスがNCPとの初期会話を行なった時または，NCPがプロセスを起動した時のいずれかである。

PRCTAB（64 Byte／ステージ・クラス）

JOB state	Process state	Commond from Process	event state
Command to IMP	load—P state	load—E state	option
Program name			
Process —ID state	Process —ID		
Timer			Job account
Post used	message address to IMP		
message counter	long message use address		
counter	event wait pointer		
counter	event post pointer		
counter	SYSIN Job pointer		
counter	LOAD running pointer		
Connection flag	connection block pointer		
Wait flag	from process message pointer		

図4—82　プロセス・テーブル

テーブル詳細

(1) JOB state

このジョブ・クラスがネットワークに関連するか否かを示す。

"00"の時関係なし，"0F"の時関係有

(2) Process state

初期状態の時"00"，communication 中の時0n（n＝1……5）

(3) Command from process

プロセスから連絡されたコマンド（H—Pプロトコル）

(4) event state

プロセスのイベント待ちおよびイベント・ポストの情報

(5) command to IMP

IMP出力キューに登録したH—Hコマンド

(6) load—P state

プロセスからのジョブ起動要求を示すフラグ

(7) load—E state

他系プロセスからのジョブ起動要求を示すフラグ

(8) option

拡張バイト

(9) Program name

NCPと会話中のプログラム名

(10) Process—ID state

NCPが起動したプロセスであるか否かを示すフラグ

(11) Process—ID

NCPが起動したプロセスにつけられた名称

(12) Timer

将来の拡張用領域で実時間処理に用いられる。

(13) Job account

将来の拡張用領域で，JOBアカウント用エリア

(14) Post used & message address to IMP

H—H commandを（connection dataを除く）IMP出力キューに登録した時そのメッセージのアドレスをstackする領域で，先頭1バイトはPOSTコマンドの時のみ使用される。

(15) message counter & long message use address

プロセスから連絡されたデータがコネクション・データ（PUT，PUTNマクロ)の時，転送回数の制御及び転送アドレスの制御をするための領域。

(16) counter & event wait pointer

イベント待ちのECBブロックへのポインタ及びブロック数

(17) counter & event post pointer

イベント・ポスト済みのECBブロックへのポインタ及びブロック数

(18) counter & SYSIN Job pointer

NCPが起動要求を出したプロセスのうちSYSIN中のジョブを示すECBブロックへのポインタおよびブロック数。

⑲ counter & LOAD running pointer

LOADEまたはLOADP実行中の時用いられるポインタ及びカウンタ

⑳ Connection flag & connection block pointer

このプロセスが持つコネクション・ブロックへのポインタおよびコネクション数

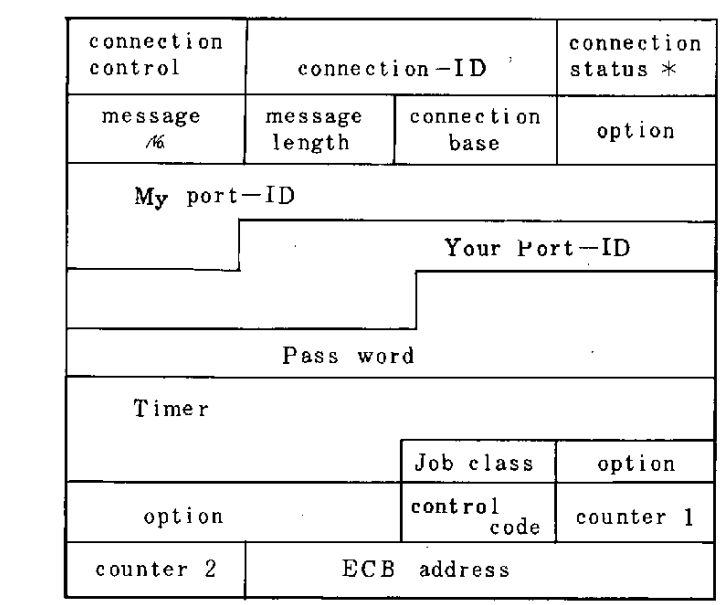
㉑ wait flag & from process message pointer

プロセスからのメッセージのアドレスをstackする領域で，先頭1バイトはWAIT 用に用いられる。

C．コネクション・テーブル

プロセスがコネクションを確立してプロセス間でデータ転送を行なう時，必要とされる制御テーブルで，他系または自系プロセスよりコネクション確立要求によって1ブロックが割当てられ，解放要求によってそのブロックが返還される。コネクション・テーブルは20ブロックからなり，NCPはデータ転送の制御をするために，コネクション・データが送られてくるとこのテーブルを操作する。1ブロックの内容は図4－83の通りである。

CNCTAB



＊ connection status

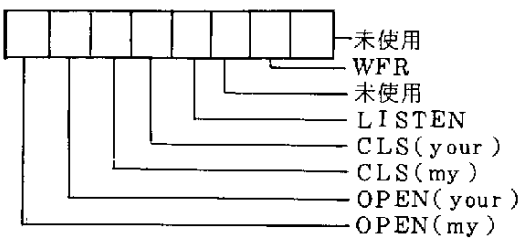


図4－83　コネクション・テーブル

テーブル詳細

(1) connection control

コネクション・ブロックの使用中フラグ（initial＝00，使用中＝FF）

(2) connection－ID

このコネクションに付けられた名称

(3) connection status

コネクション・ブロックの状態を表わすバイト

(4) message ㎞

転送されたメッセージ数

(5) message length

コネクション確立要求時の最大メッセージ長

(6) connection base

確立要求時，入力側である時connection－ID作成のために用いられる番号

(7) option

拡張バイト

(8) My port－ID

(9) Your port－ID

(10) Pass word

(11) Timer

将来の拡張用領域で実時間処理に用いられる。

(12) Job class

このコネクションの属するジョブ・クラス

(13) option

拡張バイト

(14) control code

コネクション・ブロックが入力用か出力用かを示す。入力用の時FF，出力用の時00である。また，LISTNで用いられる時にはこのコードをF0とする。

(15) counter－1

自系プロセスのメッセージ制御用カウンタ

(16) counter－2

他系プロセスのメッセージ制御用カウンタ

(17) ECB address

自系または他系プロセスよりメッセージの送受が行なわれる時ECBブロックによって制

御を行なう。この時のECBブロックへのポインター。このＥＣＢアドレスが存在するのは，カウンタ１または２が０以外の時である。

d．キュー管理テーブル

キュー・テーブルには，次の３種類があり，以下のように管理する。

1. IMP　キュー・テーブル
2. ＩＰＣ　キュー・テーブル
3. receive　キュー・テーブル

IMPおよび，ＩＰＣキュー・テーブルは原則としてＦＩＦＯ（First－In First－Out）で管理する。それぞれのキューは，各々のためのtop－pointerとtail－pointerにより管理する。

receiveキュー・テーブルは，指定されたJob class idのキューを取り出す必要があるために，キューの取り出しが起った時はpop－upせず，そのキューのあった場所を「空」としておき，テーブルを最後まで使いきった時点で「空」の部分を利用して，キューを全体的にpop－upする方式である。

図４－84に各キュー・テーブル及び管理方式を示す。

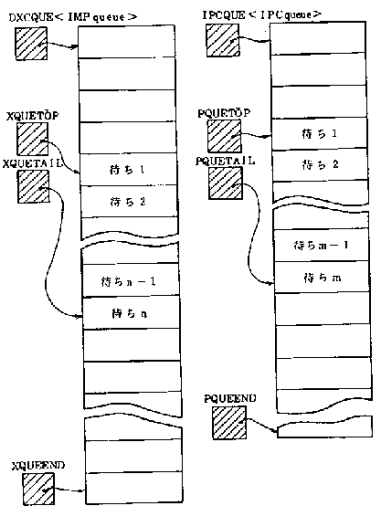


図４－84　キューテーブル及び管理方式

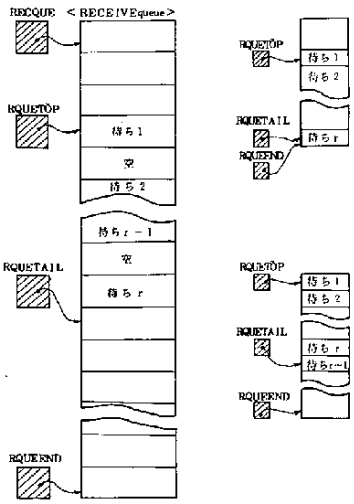


図4－84　続　き

e．フリー・ストレージ管理テーブル

セグメント・テーブル，ECBテーブル，コントロール・テーブルの各テーブルは，各々のブロック・サイズが指定されているので（セグメント・テーブル：148bytes，ECBテーブル：16bytes，コントロール・テーブル：8bytes）各ブロックを各々の種類のテーブルごとにリストしておき，このリストをフリー・リストとして管理する。

各テーブルのフリー・リストのエントリーは次のとおりである。

1. セグメント・テーブル　　SFREENT
2. ECB・テーブル　　EFREENT
3. コントロール・テーブル　　CFREENT

図4－85にフリー・ストレージ管理方式を示す。

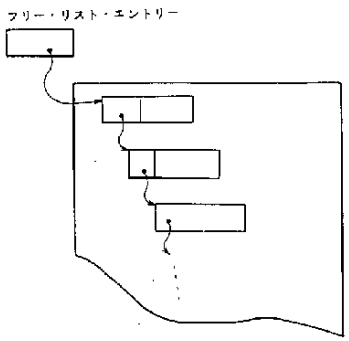


図４－85　フリー・ストレージ管理方式

f．ジョブ起動用テーブル

ジョブ起動の際，MSO．SYSCATファイルにジョブを登録するが，この際に使用するテーブルは，ジョブクラス名称と，プロシージャ名称，カード何枚分（最大４枚）のデータをＩ／Ｏバッファに登録中かのデータおよび与えられた入力カード・データをディスクに出力するためのＩ／Ｏバッファなどのためのエリアからなる。

その形式を図４－86に示す。

DISKBUF

00	m	procedure name	00	（0A用）
01			01	（0B用）
02			02	（0C用）
03			03	（1B用）
04			04	（1C用）

←――ディスクＩ／Ｏバッファ――→

図４－86　ディスク・Ｉ／Ｏバッファ

B．ディスパッチ・モジュールが使用する各テーブル

a．Extended Data Control Block Table（EDCBT）：８０バイト

このテーブルはDTFNTファイル定義マクロによって，各ファイルに追加されるテーブルで指定ファイルに関する各種データがセットされているテーブルである。

図４－87にEDCBTを示す。

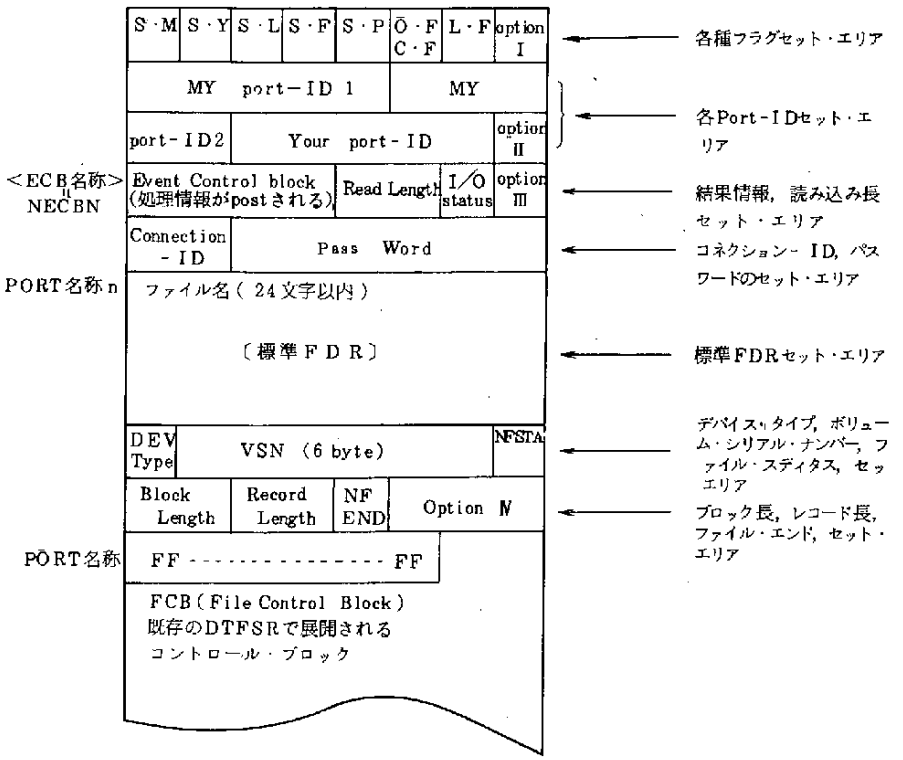


図4－87　EDCBT

b. Event Control Block Table（ECBT）（ECB−ID登録テーブル）：468バイト

イベントに関係のある処理が行なわれる場合，ECB−ID及びそれに付随したデータを登録しておくテーブルである。図4－88にECBTを示す。

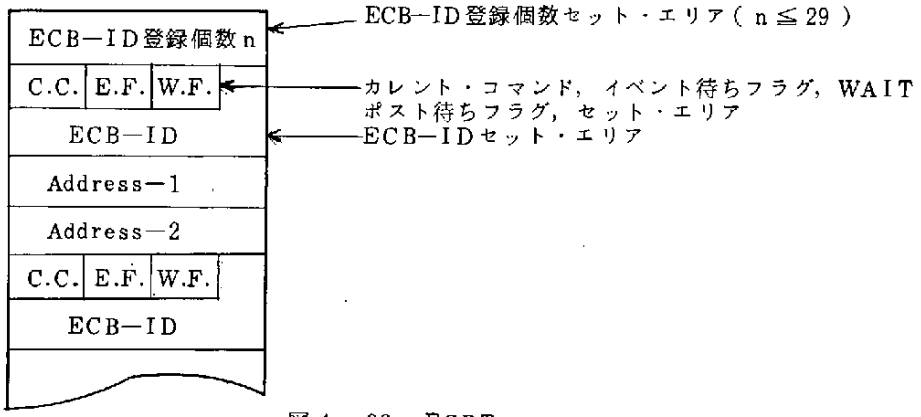


図4－88　ECBT

c. WAIT Current control Table（WAITCT）：68バイト

Wait マクロで待ち状態とするイベントを登録しておくテーブルである。

図４－89にWAITCTを示す。

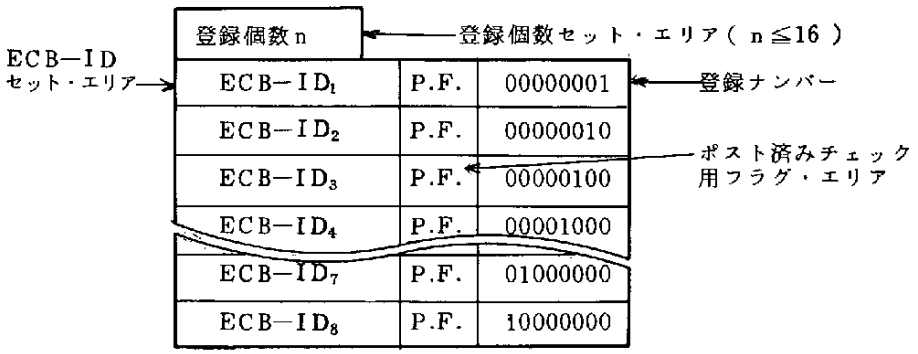


図４－89　WAITCT

d．Current Command Table（CCT）：9バイト

現在処理を実行しているコマンド解析処理ルーチンのコマンド・コード，各種アドレスがセットされる。図４－90にCCTを示す。

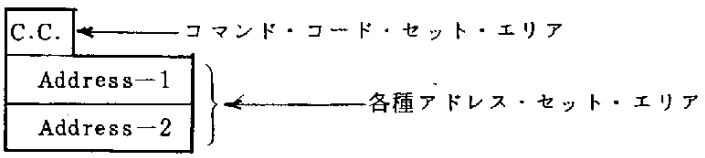


図４－90　CCT

e．ECB—ID set area（ECBSTマクロを使用）：12バイト

ECB—ID及び結果情報等をセットするためのエリアである。図４－91にECB—IDセット・エリアを示す。

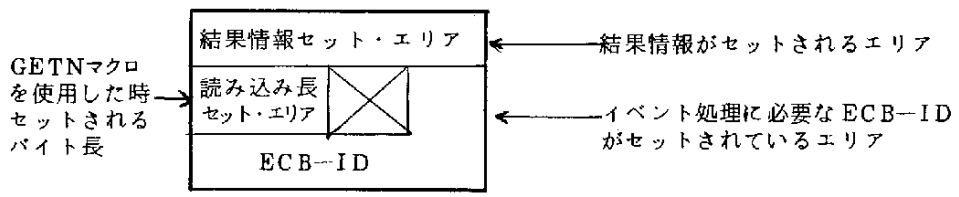


図４－91　ECB—ID　セット・エリア

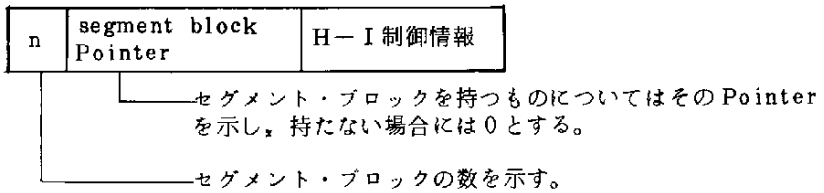
C．データ・フォーマット

フリー・ストレージは，３種類の固定長ブロックの集合体であり，それらは入出力処理用バッファ，イベント制御用バッファ，その他のバッファとして用いられ，そのデータ・フォーマットは各ブロックの形態によって種類を異にする。以下に各バッファの説明と使用される際のフォーマットについて説明する。

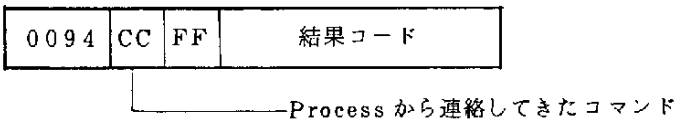
a．コントロール・ブロック

H－I間制御メッセージの読込みエリア，H－I間制御メッセージ，プロセスへのリジェクト・メッセージ作成エリア及び，プロセス管理テーブルのポインタが示すブロックとして用いられる1ブロック8バイトの領域である。そのフォーマットは用途によって以下の通りである。

(a) H－I間制御メッセージ

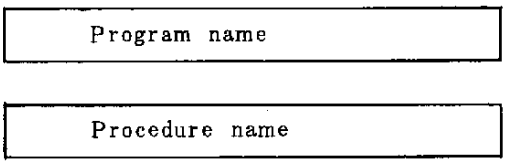


(b) H－P間制御メッセージ

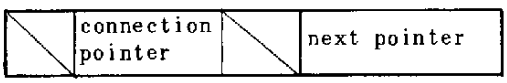


(c) プロセス管理テーブル用ブロック

(i) SYSIN, LOAD running test 用



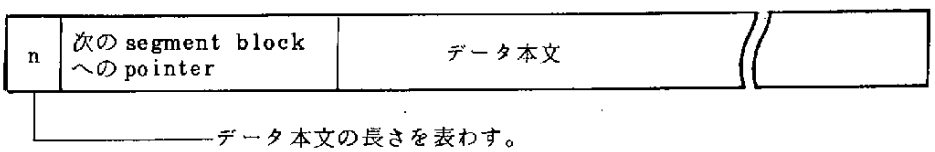
(ii) connection block pointer



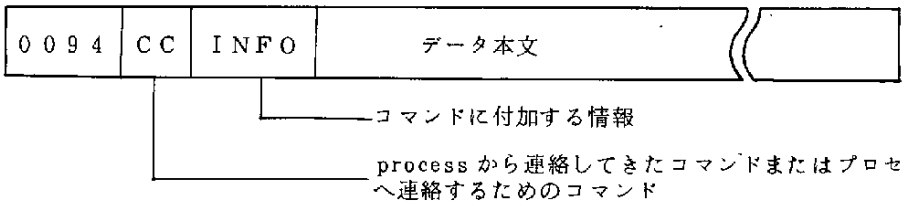
b．セグメント・ブロック

H－Hコマンドの読込みエリア，コネクション・データの読込みエリア，プロセスからのデータの読込みエリア及びプロセス，IMPへの連絡用データの作成エリアとして用いられる1ブロック148バイトの領域である。そのフォーマットは用途によって以下の通りである。

(a) H－Hコマンド，コネクション・データ

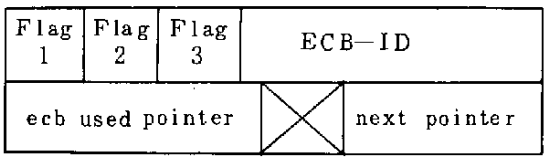


(b) プロセスからのデータ



c．イベント・ブロック

H—Hコマンド中のPOSTコマンド，コネクションデータの制御用，プロセス・テーブルのイベント管理用等に用いられる1ブロック16バイトの領域で，そのフォーマットは次の通りである。



Flag 1～Flag 3は用途によって異なり，ecb used pointerも利用用途によって異なる。

4.3.6 NCPのインプリメンテーション

NCPのインプリメンテーションにおける処理プログラムでの，割込み処理ルーチン，シリアル・リューザブル・ルーチンの取扱いおよび，そのアルゴリズム的な面に関するNCP特有な問題につき以下に述べる。また最後にNCPのインプリメンテーションにおける問題点につきまとめる。

A．プログラム構造と制限事項および条件

NCPは，各種の割込みによって動作し，そのプログラム構造は基本的にリューザブルな処理ルーチンの集合体となっている。しかし各割込み処理ルーチンの動作中に他の割込みが生じる事は充分考えられ，その制御も行なわなければならない。そのためNCPは各割込み処理ルーチンで，他の割込み処理ルーチンの動作状況を調べ，優先割込みがある場合にはその処理を先に行なうアルゴリズムを持つ。また，キューの取扱い，フリー・ブロックの管理などのシリアル・リューザブルなルーチンが動作中であればその処理を優先的に行なう。これらの処理が円滑に行なわれる様にNCPのプログラム構造は，割込み処理ルーチンを主体としている。図4—92にプログラム構造を示す。

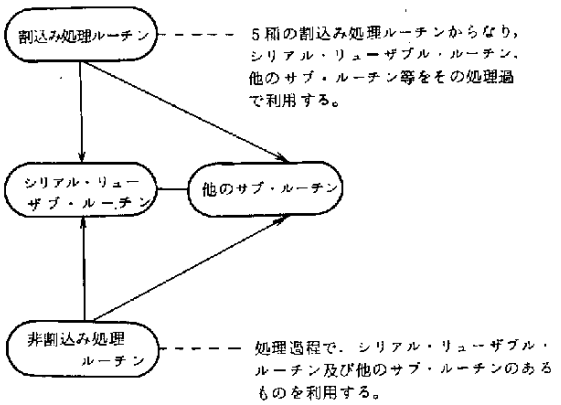


図4－92　プログラム構造

NCPがプロセスおよびバッファを管理する場合の物理的な制約条件は，以下の通りである。

(1)　プロセス数＝5

同時に管理できるプロセスの数は最大5本である。HITAC 8450　EDOS－MSOではジョブ・クラスの設定がなされているが，現在の機器構成と領域の制限により0A，0B，0C，1B，1Cの5ジョブ・クラスの管理のみを行なっている。

(2)　コネクション数＝20

同時に確立し得るコネクション数は最大20であり，各コネクションには夫々コネクション・ブロックが対応する。

(3)　コントロール・ブロック数＝100

1ブロック8バイトのコントロール・ブロックで100ブロック確保している。

(4)　ECBブロック数＝50

1ブロック16バイトのECBブロックで50ブロック確保している。

(5)　セグメント・ブロック数＝160

1ブロック148バイトのセグメント・ブロックで160ブロック確保している。8パケット・メッセージの場合8ブロックのセグメント・ブロックを用いる。

(6)　キュー・テーブル＝50

IMP出力キュー，IPC出力キュー，receiveキューはそれぞれ50個迄のキューを登録できる。

以上のような構造および制限のもとで，NCPは次のような条件および特色を持つ。

①　1 CPUの下で動作する。

②　EDOS—MSOの下で，リアル・タイム・クラスのジョブとして動作する。（JOB優先権33）

③　同一割込みが多重に発生する事はない。

（DXC割込み処理中DXC割込みが発生する事はない。）

④ 各種割込みは原則として非同期に生じる。

⑤ 割込み処理ルーチンとして，次の5種類を使用する。

a．DXC割込み処理ルーチン

b．IPC割込み処理ルーチン

c．キー・イン／アウト・シミュレーション割込み処理ルーチン

d．コンソール割込み処理ルーチン

e．リアル・タイム割込み処理ルーチン

⑥ NCPはマルチ・タスクのcontrolを行なう。

B．プログラム作成上の留意点

NCPの機能実現のために動作する各割込み処理ルーチンおよびIDLEルーチンを作成する上で，既存のEDOS—MSOがサポートするマクロの機能あるいは，非同期に発生する割込みの制御等に関し，若干問題点があり，下記のような点に留意した。

(1) DXC割込み処理

IMPからのデータ転送は完全に非同期な形で発生する。この時NCP側からIMPへのデータ転送を行なっている場合には，メッセージの衝突および，すれちがいという問題が発生する。また，IMPからのメッセージを受信し終ってOSがセンス情報確得のためのコマンドを発行するタイミングで再びIMPからの装置要求割込みが生じ，センス・コマンドの終了をさまたげる状況も存在する。これらのI／O処理に伴なうOSのタイミングの問題をかなり厳密に検討せねばならない。

(2) IPC割込み処理

IPC割込み処理の読込みマクロでは，一度に読込む事ができる長さは，256バイト以内であり，多量のデータ転送は数回に分割して読込む事が必要となる。しかも一度の割込み処理に対して1回の読込みしかできない。したがって，読込みマクロ実行後割込み処理を抜け出して連続するデータの読込みのため再び割込みを待つ必要が生じる。

(3) キー・イン／アウト・シミュレーション機能

NCPはプロセスの起動のためにEDOS—MSOのキー・イン／アウト・シミュレーション機能を利用しているが，キー・イン／アウト・シミュレーション・マクロで発行可能なコマンドの種類や機能は必ずしも充分ではない。また，既存OSにおけるこの機能は，一般ユーザの利用度も低いため，他のモジュールとの関連，出力メッセージ等不明確な点も多く，NCPをユーザプログラムのレベルで作成する上でのネックとなる。

(4) IDLE処理

IDLE処理中に割込みが発生した場合，各種の状態バイトを再チェックする必要がある。例えば，IMPへの出力キューが存在しないと判断している場合に，ある割込みが発生し，IMP

への出力キューの登録を行なうと再度IMP出力キューのチェックを行なう必要が生じる。

また，IDLE処理では，IMPの出力キューおよびプロセスへの出力キューのチェックを行なうが，その順序は，プロセス出力キューを先に調べている。これはNCP動作中にはプロセスが動作しない事から，プロセスへのコントロール移行を先ず行ない，プロセスの処理と，IMPへの出力処理の平行処理を可能とし，全体の効率を上げるべき配慮した結果である。

(5) IMPへのデータ転送

IMPへの出力を行なうルーチン内では，出力中の状態及び衝突手順を正しく把握するために，I/Oマクロ実行の前後にゲートを設けている。これにより，衝突手順及び異常な装置要求割込みの検査等の微妙なタイミングの問題を解決している。

C．NCPの割込み処理

一般に割込みの判断とその処理は割込み禁止モードで行なわれる事が望ましい。EDOS—MSOでは，割込み処理をユーザのプログラムに任せる方法をとっている。この場合割込み処理ルーチンは完全な割込み禁止モードで動作する事はできず，わずかに同一種類の割込みに対してのみ二重割込みが禁止されている。例えばDXC割込み処理中にIPC割込みが生じる事はあるが，DXC割込みは生じない。

NCPにおいては，IMPからのデータ転送，プロセスからの連絡，エグゼクティブからの連絡等を処理するための数種の割込み処理ルーチンを用意している。これらの割込みは非同期に生じ，しかもある割込み処理ルーチンの処理中であっても他の種類の割込みが生じるためNCP内で割込みの管理を行ない，バッファの保護，ルーチンの使用権，非同期性の制御を行なわなければならない。

割込み管理方式には，固有優先順位，状態遷移，キューイング，リエントラント処理などが考えられるが，NCPでは同一種類の割込みに対する二重割込み禁止に着眼して，状態遷移を主体にした管理方式を用いている。

以下に主要な割込み処理ルーチンの特徴，処理内容等について述べる。

a．各割込み処理ルーチンの特徴（①）およびNCPにおける処理内容（②）

(1) DXC割込み処理ルーチン（図4－93）

①ⅰ 装置要求割込みとして非同期に生じた割込みを処理する。

ⅱ 割込み処理ルーチン内でReadコマンドを発行しても，しなくてもよい。

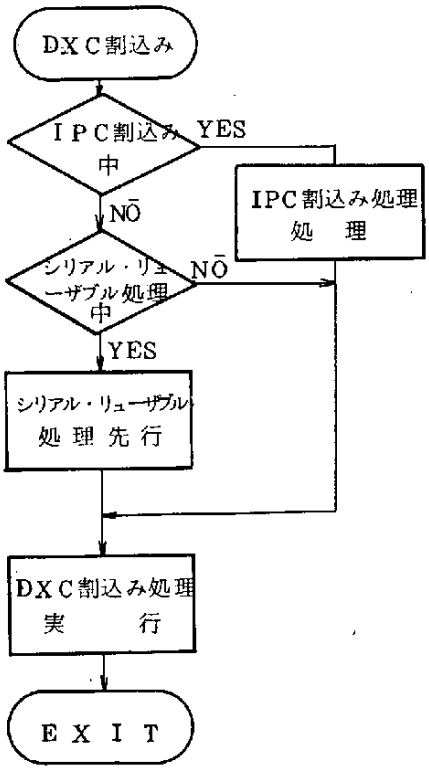
ⅲ 汎用レジスタ10，11が退避，回復される。

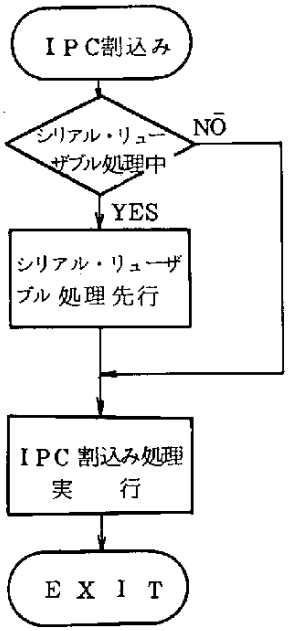
②ⅰ 割込み処理ルーチン内でReadを必ず出し，Writeを出す事もある。

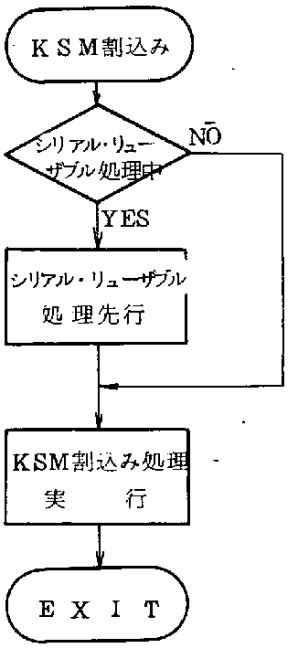
ⅱ IPC割込み処理中であればその処理を先行する。

ⅲ シリアル・リューザブル・ルーチン使用中であればその処理を先行する。

ⅳ キー・イン／アウト・シミュレーション・マクロを発行する事がある。


DXC割込み
IPC割込み中
YES
NO
IPC割込み処理
処　理
シリアル・リューザブル処理中
NO
YES
シリアル・リューザブル
処 理 先 行
DXC割込み処理
実　　行
E X I T


IPC割込み
シリアル・リューザブル処理中
NO
YES
シリアル・リューザブル処理先行
IPC割込み処理
実　　行
E X I T


KSM割込み
シリアル・リューザブル処理中
NO
YES
シリアル・リューザブル
処理先行
KSM割込み処理
実　　行
E X I T

(2) IPC割込み処理ルーチン（図4－94）

①ⅰ 他のプログラムからPC Writeがに対して生じた時動作する。

ⅱ PC Writeを発行したプロセスに制御がわたるのは，NCPがIdle中か，I／O待ち，およびSタイプ・マクロのIdleのいずれかである。

ⅲ 割込み処理ルーチン内でPC Readを発行しなければならない。

ⅳ 汎用レジスタ10，11が退避，回復される。

②ⅰ 割込み処理ルーチン内ではPC Readを必ず出す。

ⅱ シリアル・リューザブル・ルーチン使用中であれば，その処理を先行する。

ⅲ キー・イン／アウト・シミュレーション・マクロを発行する事がある。

(3) KSM割込み処理ルーチン（図4－15）

①ⅰ サボーディネータがTypeマクロを発生した時動作する。動作するタイミングは非同期である。

ⅱ 汎用レジスタ10，11が退避，回復される。

②ⅰ シリアル・リューザブル・ルーチン使用中であればその処理を先行する。

ⅱ キー・イン／アウト・シミュレーション・マクロを発行する事がある。

ⅲ 他の割込み処理ルーチンが使用するバッファ，テーブルの書替えを行なわない。

b．割込みとコンディション・コード

各割込みが発生した時P－counter退避エリアには，割込み発生時のプログラム・カウンタ（P1）がセーブされる。この時Pカウンタには，ILC（instruction length code），CC（condition code）およびPM（program mask）がセットされている。

割込み発生後，前の処理を先行する必要がある時には，このCCおよびPMをCPUに知らせる必要が生じる。このためレジスタにBranch addreesをセット（P－counter）して，SPM命令を実行しなければならない。

D．シリアル・リューザブル・ルーチンの制御

シリアル・リューザブル・ルーチンの利用中何らかの割込みが発生し，このルーチンを利用したい時には，利用されていたルーチンが終了した後，利用可能となる様な制御方式を必要とする。そのため，NCPでは図4－96に示す方法により，シリアル・リューザブルにしている。

先行処理モジュールがシリアル・リューザブル・ルーチンの使用を要求し，その処理中に何らかの割込みが生じた時，割込み処理ルーチンで，シリアル・リューザブル・ルーチンの復帰アドレスをAからBへ変更し，割込みexitのアドレスをCからAに変更する。その後Cのアドレスに制御をわたす事によってC～D迄の処理が先行される。割込み発生時にプログラム・カウンタが，次の実行する命令のアドレスを示している事と，それをユーザが参照する事が可能なことによって実現可能となる。

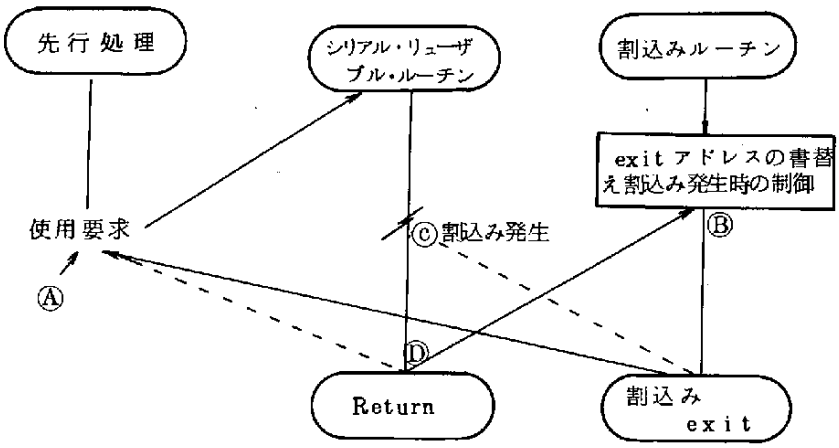


図4－96　シリアル・リューザブル・ルーチン

a．シリアル・リューザブル・ルーチンの使用方法

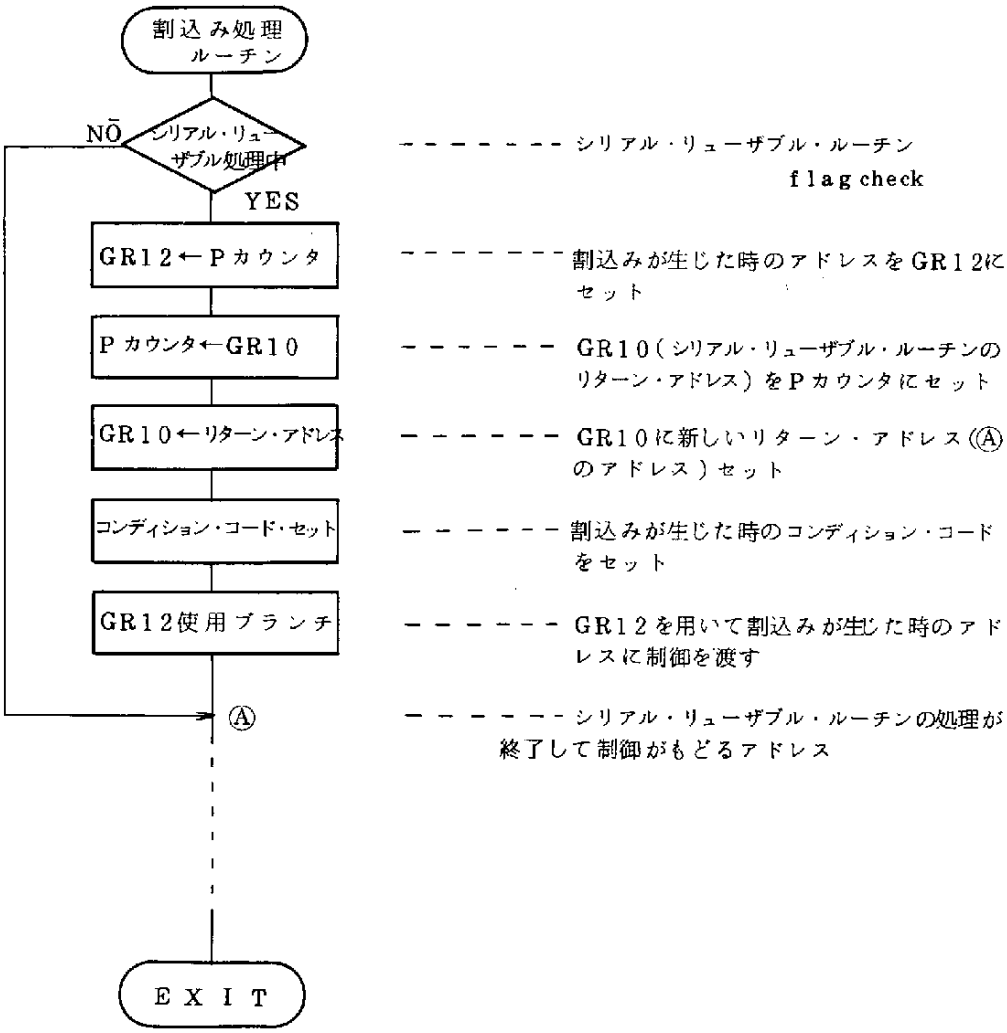
シリアル・リューザブル・ルーチンを使用する時には次のコーリング・シーケンスを用いる。

```
CNOP    0，4          ……バウンダリ調整
LA      10，＊＋12    ……GR10……ルーチン復帰アドレス
BAL     11，ルーチン名 ……GR11……アーギュメント・アドレス
DS      F             ……アーギュメント
```

この時GR10（汎用レジスタ10）は，割込み処理ルーチンの，シリアル・リューザブル・ルーチン制御によって変更される事があるが，GR11は変化しない。また，シリアル・リューザブル・ルーチン内ではGR10，11の退避，回復は行なわない。

b．割込み処理ルーチンのリューザブル・ルーチン制御

割込み処理ルーチンのシリアル・リューザブル・ルーチン制御の方法を図4－97に従って詳しくのべる。


割込み処理ルーチン
NO
シリアル・リューザブル処理中
YES
GR12←Pカウンタ
Pカウンタ←GR10
GR10←リターン・アドレス
コンディション・コード・セット
GR12使用ブランチ
Ⓐ
EXIT
シリアル・リューザブル・ルーチン flag check
割込みが生じた時のアドレスをGR12にセット
GR10（シリアル・リューザブル・ルーチンのリターン・アドレス）をPカウンタにセット
GR10に新しいリターン・アドレス（Ⓐのアドレス）セット
割込みが生じた時のコンディション・コードをセット
GR12を用いて割込みが生じた時のアドレスに制御を渡す
シリアル・リューザブル・ルーチンの処理が終了して制御がもどるアドレス

E　レジスタの設定

NCPインプリメント上汎用レジスタ16個について次の様に設定する。

表4－6にレジスタの使用目的を示す。

表4－6　レジスタの使用目的

GR	使用目的	詳細	備考
0	特殊目的	特殊timer用として利用	
1	ベース・レジスタ	領域ベース・レジスタ	CNCTAB以下の領域
2		固定ベース・レジスタ	ベース1
3			ベース2
4			ベース3
5			ベース4
6		特殊ベース・レジスタ	ベース5及びサブルーチン用(注1)
7	ワーク・レジスタ	WORK及びサブルーチンブランチ・レジスタとしても使用	H－Hコマンド処理モジュールへのブランチ・リターンで使用
8			
9			
10	割込み処理シリアル・リューザブル	割込み処理ルーチンで復帰されるレジスタ	シリアル・リューザブルのリターン・レジスタ及びアーギュメント・レジスタとして用いる。
11			
12	割込みスケジュール	割込み処理中優先割込み処理の時使用	
13	サブルーチン専用	シリアル・リューザブル，その他のサブルーチンでWORKレジスタとして使用	
14			
15			

（注1）　サブルーチン用ベース・レジスタとして用いる時には5種類の値をとる。

F．インプリメンテーション上の問題点

NCPを通常のジョブ形態を持つプログラムとして作成した理由は次の通りである。

① 既存のマクロ（ユーザ・プログラムで作成可能な）で，NCPの機能実現が可能である。すなわちIMPとの会話にDXC制御装置用マクロ，プロセスとの会話にIPCマクロ，ジョブ起動のためにKey—Simマクロを用いる事により，NCPの基本機能を満足する事ができる。

② サブ・モニタ作成のためには，EDOS—MSOの細部にわたる資料が必要となり，しかもその解析に時間を要する。また，サブ・モニタとOSとのインターフェース部分を作成するためにEDOS—MSOの設計にたずさわったスペシャリストが必ず必要となる。

③ NCPがリアルタイム・クラスのジョブとして，優先順位の高いタスクを利用する事ができ，他のバッチ・ジョブと比較して，優先的に動作する事が可能である。

以下にNCPをサブ・モニタとして作成した場合の比較および，OSに対する要望について述べる。

a．サブ・モニタとNCP

コントロール・プログラムに組込まれて特権モードで動作するNCPを仮定すると，EDOS—MSOにおけるBCS（basic communication support）と同様な位置付けが考えられる。すなわちサブ・モニタ的性格をもつコントロール・プログラムが設定される。この様な形態においてNCPを作成した時の利点を現NCPをもとに考えてみる。

① ディスパッチ・モジュールが不要となる。

ユーザ・プログラムにリンケージ・バインドされるディスパッチ・モジュールの必要性がなくなり，全てSVC（システム・マクロ）でNCPへ直接連絡が可能となり，連絡の終了はシステム・マクロの終了として検知できる。この事によりユーザ・プログラムが非同期な割込みで動作するという事はなくなる。

② OSの管理テーブルの参照が可能となる。

システム・リソースの使用状況，タスクの状態，ジョブの状態などを参照する事が可能であり，その操作も可能となる。

③ NCPの割込み管理が容易になる。

システム・ステータス（P2）で動作するプログラムのうち，あるものはインヒビット・モードでの動作が可能であり，特権命令の使用が許される。これらにより割込み処理の制御および多重割込みのスケジューリングが可能となる。

④ ジョブ起動が拡張される。

ジョブ起動及びスケジューリングを，ジョブ管理，SYSIN管理を参照して直接的に行なう事ができれば，Key—Simマクロの発行を必要としなくなる。

一方EDOS—MSOにおいてサブ・モニタ形式でNCPを作成するにあたっては次のような問題がある。

① 新規システム・マクロの登録およびOSの一部変更。

システム・マクロの定義，既存のSVC解析ルーチンの変更及び，SVC処理ルーチンの追加が必要となる。これに伴ない，標準EDOS—MSOのバージョン・アップの都度，変更部分の訂正も行なわねばならない。

② NCPが常時レジデントになる。

システム・イニシェーション時にサブ・モニタの起動が必要となり，NCPが不要な時にもコアに常駐することとなる。

③ エラー・リカバリーの難しさ。

NCPのダウンが直接システム・ダウンと結びつく可能性が大きいので，エラー・リカバリーに細心の注意を必要とする。またエラー状況のロギング，ダイナミック・ループ，システム・アイドルの極小化のために，プログラム構造を考慮する必要がある。

今回のNCPの作成に際し，サブ・モニタ形式をとらなかった理由は上記の問題点もさることながら，むしろNCP作成に必要な既存OSに関する細部にわたるドキュメントの入手が期待できぬという判断による。

b．OSに対する要望

NCPの処理プログラムで作成した場合OSの機能としてどのようなものがあればよりよいかを検討する。

(1) キュー・オン・オフ機能

各種のキュー・アクセス用のマクロがあり，キュー・オン・オフに伴なうイベント通知が行なわれれば，独自のキュー処理ルーチンを作成する必要がなく，また，このルーチンがリエントラントもしくは，リューザブルであれば各種割込み処理ルーチンから自由に扱う事ができる。

(2) フリー・ストレージ管理機能

現在あるメモリ確保および解放のマクロ以外に，ユーザが指定したエリアを指定したサイズだけ得られる様なマクロがあれば，領域管理が効果的に行なわれる。さらに領域の利用状況を把握する事ができれば，ネットワークにおける自動的なリソース割当ての際に便利である。

(3) 割込み禁止ルーチンの作成機能

あらゆる割込みを総合的に管理するために，インヒビット・モードのルーチンの作成が可能である事が望ましい。この事により各種割込み処理ルーチンのスケジューリング，領域へのアクセス方式の簡略化，競合問題の解決等でより簡単なプログラム構造とする事が可能となる。

(4) システム・マクロの登録

各種の機能を果すシステム・マクロ登録が簡単であり，しかもシステム・ステータス（P2）で動作可能であれば，現NCPの処理の多くの部分がシステム・マクロとして登録される事が可能であろう。

(5) タイマの時間単位

EDOS−MSOでは実時間割込みは，秒（1〜20000）単位であるが，NCP内で各種ルーチンのスケジュールを行なう場合，少くともミリ秒単位の設定ができることが望ましい。

4.4 NCPの性能測定

ＪIPNETの性能評価のための基礎データを得ることを目的として，ユーザ・プロセス，NCP，IMPの各結合点でのデータ転送の最大スループットを測定する。

ネットワークの実際の運用時におけるデータ伝送の状況は，コミュニケーションを行なっているプロセスの数・性質，コネクションの数，メッセージ長，転送要求の頻度・間隔，HOST のバック・グラウンドで流れているジョブ等各種の要因が複雑にからみ合った環境になっている。

しかし，第1段階のインプリメンテーションを一応区切りづける段階で，上記の状況を，かなり単純化した理想的環境におけるデータを把握する事は，性能評価の第1ステップとして必要な事である。

今回の測定の目的は，主にF，H両NCPのメッセージ伝送の最大能力を知るため，人工的なメッセージ・ジェネレータ／レシーバにより，理想的なプログラム環境で，メッセージ長，コネクション数を変化させ，単一伝送方向の最大スループットを測定することである。しかしながら，４９年度末までには，このすべての予定を完了する事が困難であったため，本節では測定方法，測定のために新たに用意したArtificial Message Generator／Receiverの機能について概説した後，現時点までに得られている測定結果とそれに対するコメントを付記し，さらに測定に付随して発生した問題点について若干ふれることにする。

A．測定の方法

測定用のArtificial Message Generator／Receiver（MG／MR）をHOST内ユーザ・プロセス，NCP，IMPの３点に夫々配置して，各パス毎に，メッセージ長，コネクション数を変化させた際の単方向最大スループットを求める。図４－98はその配置を示す。

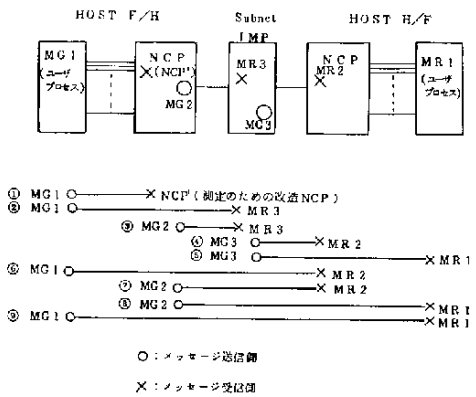


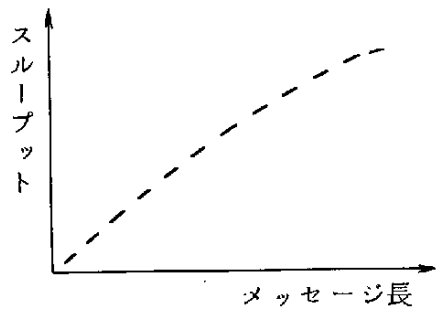
図４－98　測定ルーチン配置図

図4－98に記したMG—MRの組合せを次の3ケースに分けて，各ケースで次のような方法でスループットを測定する。

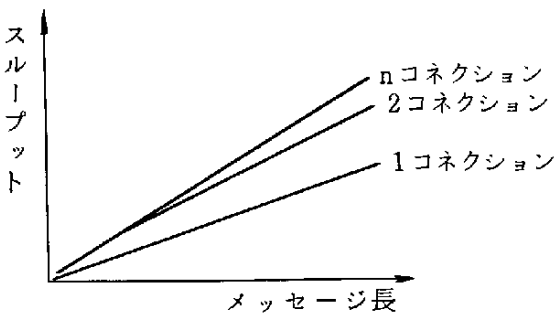
(1) ケース1（ユーザ・プロセスから各receiver宛）

①MG1—NCP', ②MG1—MR3, ⑥MG1—MR2, ⑨MG1—MR1

a．単一コネクションでメッセージ長をパラメータとしてスループットの変化を測定する。



b．上記aの測定にさらにコネクション数のパラメータを加え，コネクション2～18本に同一サイズのメッセージを流した時のトータルおよびコネクション毎のスループットを測定する。



c．MG1からMR1，2，3へのメッセージについてはコネクション毎のメッセージ・ラウンド・トリップ時間を測定する。

d．適当な分布で各コネクションのメッセージ長を変えて，b，cを測定する。

(2) ケース2（（NCPから各receiver宛）

③MG2－MR3，⑦MG2－MR2，⑧MG2－MR1

ケース1のaと同じ。

(3) ケース3（IMPから各receiver宛）

④MG3－MR2，⑤MG3－MR1

ケース1のaと同じ

B．Message Generator／Receiverの機能概要

1. MG1

MG1はHOST F，Hのユーザ・プロセスとして働くもので，下記の機能を果たす。

(1) 指定された数のコネクションを確立する。

(2) 各コネクション毎に指定されたサイズのメッセージを発生させMRに送出する。（1full packet～8full packet）

(3) 1定個数のメッセージ送出完了後，タイマーにより，トータル・スループット（K bps）とmessage round trip time平均を算出し記録する。

(4) コネクション数を1～18まで変化させ(1)～(3)を繰り返す。

2. MR1

MR1はHOST F，Hのユーザ・プロセスとして働くもので，下記の機能を果たす。

(1) 指定された数のコネクションを確立する。

(2) 各コネクション毎に指定されたサイズのメッセージをMG1から受信する（1full packet～8full packet）

(3) 1定個数のメッセージ受信完了後その経過時間からトータル・スループットを算出し記録する。

(4) コネクション数を1～18まで変化させ，(1)～(3)を繰り返す。

(5) 相手がMG2，MG3の場合は特定のport－idのコネクション1本のみでメッセージ受信数を記録する。

3. MG2，MG3

MG2はNCP中のHOST－IMPプロトコル処理部分に，またMG3はIMP中のHOST－IMPプロトコル処理部分に存在し特定のコネクションを通して指定されたサイズのメッセージを送出するものである。

メッセージ・サイズは1full packet～8full packetいずれかを指定する。1つのメッセージを送出してから次のメッセージを送出するのにReceiveを待つ必要はなく，返ってきたReceiveは捨てる。単位時間に送出したメッセージ数をカウントしスループットを算出する。

4. MR2，MR3

MR2，MR3はNCP中のHOST－IMPプロトコル処理部分に，またMR3はIMP中のHOST－IMPプロトコル処理部分に存在しMG1，2，3から送られてきたメッセージに対するReceiveを返し単位時間に受取ったメッセージ数からスループットを算出する。

5. NCP

NCP′はNCPに修正を加え，自系内ユーザ・プロセスMG1からのメッセージを処理するものである。

図４－99にこのような測定用ツールによって得られるデータの一部を示す。

TEST—NO＝MGO２

CONNECTION数＝２本

PACKET数＝１個（144バイト），２個（288バイト）

START＝10，END＝20，ADD＝10

** TEST-NO. MG02 **

TOTAL-WRITE	*CONNECTION*	*PACKET-SU*	*SEND-SU*	*TRTT*	*ARTT*
0010	0001	0001	0005	00000331	0066
	0002	0002	0005	00000349	0069
START-TIME= 2524160	END-TIME= 2524528	KBPS= 00000048	DISCARD-REC=		0000
0020	0001	0001	0010	00000533	0053
	0002	0002	0010	00000540	0054
START-TIME= 2525361	END-TIME= 2525945	KBPS= 00000061	DISCARD-REC=		0000

$ARTTi = TRTTi / (SEND—Su)i$

START—TIME・END—TIME＝MMSSmmm

KBPS＝転送ビット数／使用時間

$$転送ビット数 = \sum_{i=1}^{CONNECTION数} (PACKET-SU)i \times 1152 \times (SEND-SU)i$$

使用時間＝（END—TIME）—（START—TIME）

図４－99　測定結果出力の例（MG１—MR３）

C．測定の中間結果

前述のように，現時点では図４－98に示した９ケースのうち一部の結果しか得られていないが，その測定結果をグラフ化すると図４－100図のようになる。図４－100のグラフはＨＩＴＡＣ，FACOMの各HOST側にMGを置いた場合のスループットを表しており，横軸は１full packetから８full packetまで段階的に変化させたメッセージ・サイズ（パケット数）を示し縦軸は，各メッセージ・サイズでの最高スループット値（K bps）を示している。

これらのグラフからも明らかなように，メッセージ・サイズが段階的に増えるにつれて，スループット値はほぼ直線的に増大している。次に両HOST，IMPの各レベルでの最大スループットについて簡単にまとめると，まず，IMPからH—NCPに送出できるデータの最大スループット値は約500K bpsである。（図４－100のａ），またH—NCPからIMPあてのデータ転送の最大スループット値は400K bpsである。（ｂ）　一方F—NCPからIMPあての最大スループット値は約350K bpsである。（ｆ）

次にH—NCPからIMP経由でF—NCPに転送する時の最大スループット値は約350K bpsである。（ｃ）

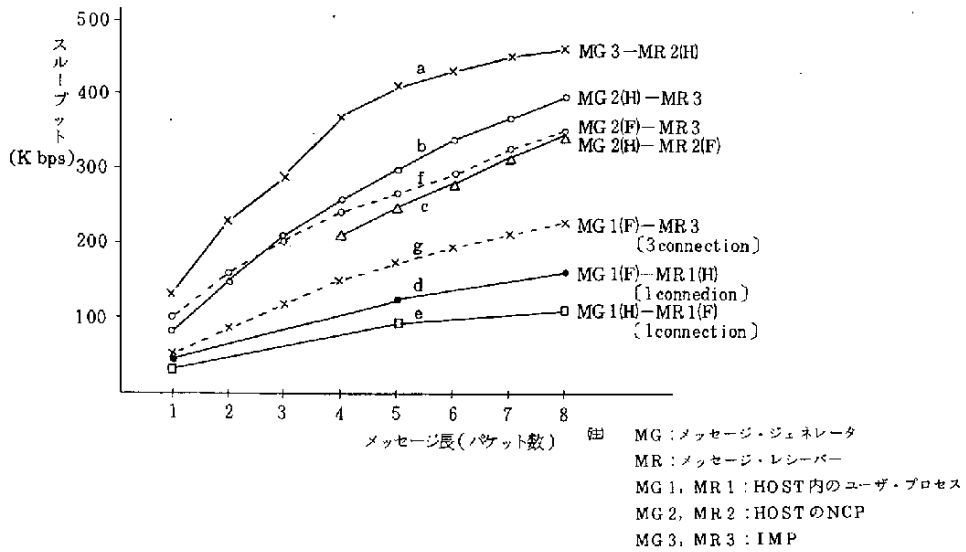


図 4 — 100　測定結果のグラフ

次にユーザ・プロセス間のMG 1—MR 1 ではF→H方向の 1 connection で最大スループットは 160K bps，H→F方向の 1 connection では 110K bps である。(d，e)

またFのユーザ・プロセスMG 1 から IMP あての最大スループットは 3 connection の場合，230K bps(g) となっている。

次に，これらの各グラフが示す値の相対的な関係について，簡単に解説する。

(1)　a と b の差について

a は IMP から H—NCP への，b は H—NCP から IMP への最大スループット値を示しているが，前者の方が後者より 50～100K bps 位高い値になっている。この理由の大部分は，HOST—IMP プロトコルによる HOST，IMP での送出手順の違いに起因している。すなわち，マルチ・パケット・メッセージの HOST→IMP の転送時は，実データの転送に先立って，バッファ予約の要求を出し，その確認応答が返された時点で実データを転送するといった手順になっており，一方 IMP→HOST には予約・確認の手順がない。この他に，HOST，IMP での HOST−IMP プロトコル処理上のオーバ・ヘッドの差が影響していると思われる。

(2)　b と f の差について

b は H—NCP から IMP への，f は F—NCP から IMP への最大スループット値を示している。これについては，逆方向の測定データが F 側で得られていないため，明確な考察はできないが，1 packet メッセージでは F−NCP の方が H−NCP よりも高い値になっていること，packet 数

が増えるにつれて（３～４packet時）その関係が逆転し８packet時にその差が最大となっている事から，F，H両NCPでのチャネル・コマンドの使用方法の違いによるものと思われる。すなわち，H−NCPではデータ部（複数セグメント）の送出をコマンド・チェインにして一回のI／O要求で行っているが，F−NCPの現バージョンではコマンド・チェイン機能を利用していないため，割込み処理の回数の差という形でb，fの差が現われてきているものと思われる。

(3) dとeの差について

d，eは両HOSTのユーザ・プロセス間で，dがF→H方向，eがH→F方向のデータ転送のスループット値を示しているが，F→Hの方がH→F方向よりも10～50Kbps程高くなっている。この理由についても，各ケースの詳細なデータがそろってから，綿密に検討しなければ，明確な考察はできないが，NCPとユーザ・プロセス間でのデータ転送が，F−NCPでは，モニタ・コールによる単純な処理であるが，H−NCPではIPC（Inter－program－communication）マクロを用いてユーザ・プロセス，H−NCP（これもモニターからは処理プログラムとして扱われる）間でパケット数分の割込みをかけ合っての転送方法となっているので，このためのオーバ・ヘッド分が影響しているものと思われる。

D．問　題　点

この測定を行なうことにより，ヘビー・ロード・トラフィックの下でNCPが動作する状況になり，それまでの実験的運用の段階では検知できなかったハード（アダプタ），ソフト（モニター，NCP）両面での問題が発生した。

この問題のほとんどは，非同期処理の微妙なタイミングに起因するものであり，その原因の究明には非常な苦労を重ねた。

これらの問題は，コマンド衝突等のタイミングに関係し，大別すると，NCP内の処理上，モニター内の処理およびハードウェア（アダプタ）の欠陥の３種に分ける事ができるが，実際に起る現象としては，複数の原因がからみ合って起るため現象が複雑になり，根本的な原因をつかみ難い。

発生した現象の主なものは次のようなものである。

(1) NCPとIMPとのデータ転送の手順で，IMPからの転送要求割込み（ATT）がつかまらない。

(2) NCPとIMPとのデータ転送の手順で，IMPからの転送要求割込みが残ってしまう。

(3) NCPがモニターに依頼した入出力コマンドが終了しない。

(4) NCPがモニターから報告される入出力コマンドの終了status情報がおかしい。

(1)はF−NCP特有の問題であるが，(2)(3)(4)はH−NCP，F−NCP共通に発生しているものである。

ここで，F—NCP，H—NCPにおけるIMPからの転送要求割込み（ATT）の処理方法（モニター・インタフェース）の概略は以下のようになっている。

a．F—NCPの場合

ATT用のECB(Event Control Block)をモニター・コール(SETECB)により登録し，ATTが入るとそのECBにPOSTされる。

POSTされたECBは再登録しないと次のATTが受取れない。

b．H—NCPの場合

ATT割込みが入った場合の割込み処理ルーチンの入口名をモニター・コールによって登録する。

ATTが入ると登録した入口名にコントロールが渡る。

割込み処理終了後EXITすると次の割込みを受付けることができる。

問題点の(1)は，aのような割込み処理の場合，次のようなタイミングが起ると，必らず表われ，このままでは救えない。すなわちATTのECBのPOSTとF—NCPからの転送要求がすれ違ってコマンド不整合になった後，F—NCPでSETECBを登録する前，あるいは登録依頼中にIMPからのリトライ（HOST—IMPプロトコルによる）の転送要求割込みが起るとこの割込み要求がつかまらない。

問題点の(2)は，やはりIMPからのATTとNCPからの転送要求がモニター内ですれ違うと，既にATTがPOSTされている状態でコマンド不整合になり，その時POSTされていたATTと，次のIMPからのリトライによるATTが続けてくることになり，ATTが一つ余分に入って来た事になる。このATTの性質を判断する必要があるが，現在のままでは救えない。

この(1)，(2)の問題に関しては，コマンド不整合後のIMPからのリトライのタイミングを少し遅らせ，NCPの転送要求の前後にGATEを設けてコントロールするようにしなければならない。

(3)の問題はアダプタの動作が，コマンド衝突の微妙なタイミングで誤動作してしまう事が発見された事が1つと，他にモニターでの衝突処理がまずいのではないかという点があり，現在，ハードに関してはより詳細な原因追求と処置を，モニターに関しては原因の調査を行なっている段階である。

(4)の問題は，チャネル—アダプタ間，アダプタ内部，アダプタ—IMP間等各所で起るコマンド衝突時のstatusが完全にチェックされ切っていないために起った現象である。

今後これらの問題点を解決すると同時に，前述の測定も併せて継続する予定である。

あ と が き

1975年1月現在，ＪＩＰＮＥＴはサブネットのＳＣＰとＦＡＣＯＭ，ＨＩＴＡＣの両ＨＯＳＴのＮＣＰおよびＤＳＰ，ＲＢＰ，ＦＴＰの基本的部分のインプリメントがほぼ完了し，ノードに直結された端末からＨＯＳＴのＴＳＳ使用も可能となっている。今後，これらの評価および修正を行なうとともに，ＣＪＰ，ＡＪＤ，ＤＤＢＡ等高度な利用法に対する検討および処理機能を加えてゆきたい。

参　考　文　献

参考文献

1. ネットワーク全般

1) コンピュータ・ネットワーク・システムの研究開発　　48－S001
日本情報処理開発センター　　1974年3月

2) Abramson, N.
The ALOHA system — another alternative for computer communications. Proceedings of AFIPS 1970 Fall Joint Computer Conference, Vol.37, pp.281-285.

3) McKay, D.B. and Karp, D.P.
IBM computer network/440. Courant Computer Science Symposium 3, December 1970, pp.27-44.

4) Herzog, B.
MERIT computer network. Courant Computer Science Symposium 3, December 1970, pp.46-47.

5) Aupperle, E.
MERIT computer network: Hardware considerations. Courant Computer Science Symposium 3, December 1970, pp.49-64.

6) Cocanower, A.
MERIT computer network: Software considerations. Courant Computer Science Symposium 3, December 1970, pp.66-77.

7) Aupperle, E.M.
MERIT network re-examined. Comp. Con., 1973, pp.25-30.

8) Kirstein, P.T.
The future network development in Europe and the USA. International Computer State of the Art Report: Computer Networks Infotech Information, 1971, Vol.6, pp.347-363.

9) Somia, M.
The approach of software problems in the SOC experimental computer network. ICCC, 1972, pp.390-396.

10) Roberts, L.G. and Wessler, B.D.
Computer network development to achieve resource sharing. SJCC, 1970.

11) Frank, H., Kahn, R.E. and Kleinrock, L.
Computer communication network design — Experience with theory and practice. Proceedings of AFIPS 1972 Spring Joint Computer Conference, Vol.40, pp.255-270.

12) McQuillan, J.M., Crowther, W.R., Cosell, B.P., Walden, D.C. and Heart, F.E.
Improvements in the design and performance of the ARPA network. Proceedings of AFIPS 1972 Fall Joint Computer Conference, Vol.41, pp.741-755.

13) Coleman, M.L.
ACCNET — A corporate computer network. '73 NCC, p.133.

14) Martel, C.C., Cunningham, I.M. and Grushcow, M.S.
The BNR network: a Canadian experience with packet switching technology. '75 IFIP Congress, pp.160-164.

15) Barber, D.L.A.
The European computer network project. '72 ICCC Washington D.C., pp.192-200.

16) Howell, R.H.
The integrated computer network system. '72 ICCC Washington D.C., pp.214-219.

17) De Mercado, J., Guindon, R., DaSilva, J. and Kadoch, M.
The Canadian Universities computer network - Topological considerations. '72 ICCC Washington D.C., pp.220-225

18) Aufenkamp, D.D. and Weiss, E.C.
NSF activities related to a national science computer network. '72 ICCC Washington D.C., pp.226-232.

19) Thies, A.W., Hawryszkiewycz, I.T. and Gannon, D.J.
Design of the Australian post office computer network. '74 ICCC Stockholm, pp.99-105.

20) Pinter, L.
Development of a Hungarian computer data center network. '74 ICCC Stockholm, pp.113-117.

21) Alarcía, G. and Herrera, S.
C.T.N.E.'s packet switching network - Its applications. '74 ICCC Stockholm, pp.163-170.

22) Despres, R.
RCP, the experimental packet switched data transmission service of the French PTT. '74 ICCC Stockholm, pp.171-185.

23) Barber, D.L.A.
Progress with the European informatics network. '74 ICCC Stockholm, pp.215-220.

2. ネットワーク・デザイン

○デザイン&インプリメンテーション

24) Pouzin, L.
CIGALE, the packet switching machine of the CYCLADES computer network. IFIP '74, Computer Networks II, pp.155-159.

25) 尾佐竹徇
"情報通信システム" システム工学講座 6 日本工業新聞社 昭和47年

26) Michael, L.W., Mills, D.L. and Zelkowitz, M.V.
Design of a distributed computer network for resource sharing. AIAA Computer Network Systems Conference, Huntsville, ALABAMA/'73.4.

27) Davies, D.W. and Barber, D.L.A.
Communication networks for computers. John Wiley & Sons, p.575.

28) The interface message processor program. BBN '73 November.

29) Kleinrock, L.
Resource allocation in computer systems and computer-communication networks (Invited Paper). '74 IFIP Congress, pp.11-18.

30) Casey, R.G. and Friedman, T.D.
Design techniques for Data-Base-Oriented computer networks. AD-763791, '73.5.

31) Hirota, K., Kato, M. and Yoshida, Y.
A design of packet switching system. '74 ICCC Stockholm, pp.151-162.

32) Pearson, D.J. and Wilkin, D.
Some design aspects of a public packet switched network. '74 ICCC Stockholm, pp.199-213.

33) Scantlebury, R.A. and Wilkinson, P.T.
The National Physical Laboratory data communication network. '74 ICCC Stockholm, pp.223-228.

34) Bailey, P.A. and Wood, B.M.
A central file store for the data communication network at the National Physical Laboratory. '74 ICCC Stockholm, pp.229-238.

○ＩＭＰに関して

35) Heart, F.E., Kahn, R.E., Ornstein, S.M., Crowther, W.R. and Walden D.C.
The interface message processor for the ARPA computer network. Proceedings of AFIPS 1970 Spring Joint Computer Conference, Vol.36, pp.551-567.

36) Roberts, L.G.
Initial design for interface message processors for the ARPA computer network. BBN Report No.1763, 1969.

37) Carr, S., Crocker, S. and Cerf, V.
Specifications for the interconnection of a Host and an IMP. BBN Report No.1822.

38) Heart, F.E., Ornstein, S.M., Crowther, W.R. and Barker, W.B.
A new minicomputer/multiprocessor for the ARPA network. '73 NCC, p.529.

39) Sharma, R.L., Shah, J.C., El-Bardai, M.T. and Sharma, K.K.
C-system: multiprocessor network architecture. '74 IFIP Congress, pp.19-23.

40) Sobolewski, J.S.
Programmable communication processors. '72 ICCC Washington D.C., pp.380-389.

○ＴＩＰに関して

41) Ornstein, S.M., Heart, F.E., Crowther, W.R., Rising, H.K., Russell, S.B. and Michel, A.
The terminal IMP for the ARPA computer network. Proceedings of AFIPS 1972 Spring Joint Computer Conference, Vol. 40, pp.243-254.

42) Kahn, R.E.
Terminal access to the ARPA computer network. Courant Computer Symposium 3 Computer Networks. Courant Institute, New York, November 1970 — Proceedings to be published by Prentice Hall, Englewood Cliffs, New Jersey, in preparation.

43) User's guide to the terminal IMP.
Bolt Beranck and Newman Inc. Report No.2183.
The BBN terminal interface message processor BBN Report 2184.

○ネットワーク・トポロジーに関して

44) Frank, H., Frisch, I.T. and Chou, W.
Topological considerations in the design of the ARPA computer network. Proceedings of AFIPS 1970 Spring Joint Computer Conference, Vol.36, pp.581-587.

45) Frank, H. and Chou, W.
Topological optimization of computer network. Proc. of IEEE, Vol.60, No.11, November 1972.

46) Kevin, V. and Whitney, M.
Comparison of network topology optimization algorithms.
'72 ICCC Washington D.C., pp.332-337.

47) Rosner, R.D.
Large scale network design considerations. '74 ICCC Stockholm, pp.189-197.

○メッセージ・スイッチング，パケッティング方式に関して

48) 岡田，宮原，真田，手塚，
"メッセージ交換処理網におけるパケッティング方式について"信学会，交換研資 (1971-10).

49) Abramson, N.
Packet switching with satellites. '73 NCC, p.695.

50) Kleinrock, L. and Lam, S.S.
Packet switching in slotted satellite channel. '73 NCC, p.703.

51) Roberts, L.G.
Dynamic allocation of satellite capacity through packet reservation. '73 NCC, p.711.

52) Davies, D.W.
Packet switching, message switching and future data communication networks. '74 IFIP Congress, pp.147-150.

○フロー・コントロールに関して

53) Davies, D.W.
The control of congestion in packet-switching networks. Trans. IEEE, Vol.COM-20, No.3, p.546, June 1972.

54) Kahn, R.E. and Crowther, W.R.
Flow control in a resource sharing computer network. Proc. of the Second ACM IEEE Symposium on Problems in the Optimization of Data Communications Systems, Palo Alto, California, October 1971, pp.108-116.

55) Price, W.L.
Simulation studies of an isarithmically controlled store and forward data communication network. '74 IFIP Congress, pp.151-154.

56) Jilek, P.
Flow control in computer networks. '74 ICCC Stockholm, pp.239-247.

○ルーティング（Routingに関して）

57) Frank, H. and Chou, W.
Routing in computer networks. Networks John Wiley, 1971, Vol.1, No.2, pp.99-112.

58) Fultz, G.L. and Kleinrock, L.
Adaptive routing techniques for store-and-forward computer communication networks. 1971 International Conference on Communication, Montreal, Canada, p.39-1, (June 1971).

59) Fultz, G.L.
Adaptive routing technique for message switching computer communication networks. UCLA-ENG-7252, July 1972.

60) Pickholtz, R.L. and McCoy, Jr., C.
Improvements in routing in a packet-switched network.
'74 ICCC Stockholm, pp.249-252.

61) Cegrell, T.
A routing procedure for the TIDAS message-switching network.
'74 ICCC Stockholm, pp.253-262.

○プロトコルに関して

62) Carr, S., Crocker, S. and Cerf, V.
Host/host protocol in the ARPA network. Proceedings of AFIPS 1970 Spring Joint Computer Conference, Vol.36, pp.589-597.

63) Crocker, S., Heafner, J., Metcalfe, R. and Postel, J.
Function-oriented protocols for the ARPA network.
Proceedings of ARPA 1972 Spring Joint Computer Conference, Vol.40, pp.271-280.

64) ARPA network current network protocols. August 1971
Available from the Network Information Center as NIC #7104 at Stanford Research Institute, Menlo Park, California 94025.

65) TELNET Protocol Specification, NIC #15372.

66) Mckenzie, A.
Host/host protocol for the ARPA network, NIC #8246.

67) 伊藤哲史
Host level の protocol. 情報処理学会講習会75.2
「コンピュータ・ネットワーク」TEXT

68) Demand Service Protocol. 日本情報処理開発センター内部資料

69) Remote Batch Protocol. 日本情報処理開発センター内部資料

70) File Transfer Protocol. 日本情報処理開発センター内部資料

71) ISO/TC 97/SC 6 (Tokyo-17) 1005.

72) Ziemmerman, H.Z., et al.
Transport protocol - Standard host-host protocol for heterogeneous computer networks. Reseau Cyclades SCH519.1.

73) Pouzin, L.
Network protocols. Reseau Cyclades SCH517, '73, 9.

3. パフォーマンス評価，解析，シミュレーション

74) Kleinrock, L.
Analytic and simulation methods in computer network design. Proceedings of AFIPS 1970 Spring Joint Computer Conference, Vol.36, pp.569-579.

75) Cole, G.D.
Performance measurements on the ARPA computer network. Proceedings of the Second ACM IEEE Symposium on Problems in the Optimization of Data Communications Systems, Palo Alto, California, October 1971, pp.39-45.

76) Kleinrock, L.
Models for computer networks. Proceedings of the International Conference on Communications, pp.21.9-21.16, June 1969.

77) Kleinrock, L.
Performance models and measurements of the ARPA computer network. ONLINE 72 Conference Proceedings, Vol.2.

78) Price, W.L.
Simulation of data transit networks. National Physical Laboratory, Com. Sci. 56, April 1972.

79) Kleinrock, L.
Comparison of solutions methods for computer network models. Proc. of the Computers and Communications Conference, Rome, New York, September 30 - October 2, 1969.

80) Bowdon, E.K., Sr., Mamrak, S.A. and Salz, F.R.
Simulation - A tool for performance evaluation in network computers. '73 NCC, p.121.

81) Friedman, T.D.
A system of APL functions to study computer networks. '73 NCC, p.141.

82) Van Slyke, R.M., Chow, W. and Frank, H.
Avoiding simulation in simulating computer communications networks. '73 NCC, p.165.

83) Kleinrock, L. and Naylor, W.E.
On measured behavior of the ARPA network. '74 NCC, pp.767-780.

84) コンピュータ・ネットワーク. Dr. Robert E. Kahn 講演録
日本情報処理開発センター 49年3月

85) Wedberg, G.H. and Hauschild, L.W.
The General Electric network monitor system. '74 IFIP Congress, pp.24-28.

86) Morgan, D.E., Banks, W., Colvin, W. and Sutton, D.
A performance measurement system for computer networks. '74 IFIP Congress, pp.29-33.

87) Coviello, G.J. and Rosner, R.D.
Cost considerations for a large data network. '74 ICCC Stockholm, pp.289-294.

88) Urano, Y., Ono, K. and Inoue, S.
Optimal design of distributed networks. '74 ICCC Stockholm, pp.413-420.

4. その他

89) Thomas, R.H.
A resource sharing executive for the ARPANET. National Computer Conference 1973, pp.155-163.

90) Walden, D.C.
A system for interprocess communication in a resource sharing computer network. C.ACM, April 1972, Vol.15, No.4, pp.221-230.

91) Thomas, R. and Henderson, D.A.
McROSS — A multi-computer programming system. Proceedings of AFIPS 1972 Spring Joint Computer Conference, Vol.40, pp.281-294.

92) Kahn, R.E.
Resource-sharing computer communications networks. Proceedings of IEEE, Vol.60, No.11, November 1972.

93) Elovitz, H.S. and Heitmeyer, C.L.
What is a computer network?. Proc. NTC '74, pp.1007-1014.

94) 付加価値通信網サービス (Value Added Network VAN).
日本情報処理開発センター「米国における情報処理の実態」 49−R001 PP. 63−78.

95) 大野豊, 田畑孝一
"コンピュータ・リンクの現状と将来". ビジネス・コミュニケーション '75, Vol.12, No.2, pp.22-29.

96) "コンピュータ・ネットワーク講習会テキスト" 情報処理学会 '75. 2.講習会

97) Krilloff, H.Z.
A high level language for use with computer networks. '73 NCC, p.149.

98) Thomas, R.H.
On the design of a resource sharing executive for the ARPANET. '73 NCC, p.155.

99) McKenzie, A.M.
Some computer network interconnection issues. '74 NCC, pp.857-860.

100) Cerf, V. and Robert, R.E.
A protocol for packet network interconnection. '74 IEEE Trans of COMM, Vol.COM-22, No.5, pp.637-648.

101) Chupin, J.C.
Control concepts of a logical network machine for data banks. '74 IFIP Congress, p.291.

102) Atkinson, M.P.
PIXIN: a data language for network modelling, '74 IFIP Congress, p.296.

103) Raymond, J. and du Masle, J.
NJCL, a network job control language. '74 IFIP Congress, p.301.

104) Eswaran, K.P.
Placement of records in a file and file allocation in a computer network. '74 IFIP Congress, p.304.

105) McKenzie, A.A. Cosell, B.P., McQuillan, J.M. and Thrope, M.J.
The network control center for the ARPA network. '72 ICCC Washington D.C., pp.185-191.

106) Smith, B.T.
Mixed computer networks: benefits, problems and guidelines. '72 ICCC Washington D.C., pp.201-208.

107) Farber, D.J. and Heinrich, F.R.
The structure of a distributed computer system - The distributed file system. '72 ICCC Washington D.C., pp.364-370.

108) Booth, G.M.
The use of distributed data bases in information networks. '72 ICCC Washington D.C., pp.371-376.

109) Somia, M.M.
Super system or subsystem in a distributed computer network. '74 ICCC Stockholm, pp.315-320.

110) Kuo, F.F.
Political and economic issues for internetwork connections. '74 ICCC Stockholm, pp.389-391.

附 録(ディスコネクション手法の証明)

附録　ディスコネクション検出手法の証明

コネクション・マトリックスMの各要素m_{ij}はIMPiとIMPjとの間に回線が直接張られていれば1，いなければ0である。言い換えればm_{ij}が1の場合はIMPiからIMPjに1ホップで到達する道のあることを示す。

グラフ理論によれば，マトリックスMを2乗した新しいマトリックスの各要素のうち，0以外のものをすべて1とおき換えたマトリックスM^2の各要素は，2つのIMP間に2ホップで行く道があるか否かを示す。

さらにマトリックスMのn乗に相当するM^nの各要素は，2つのIMP間にnホップで行く道があるか否かを示す。

まず最初にこれを証明する。証明に先立ち，以降の展開に必要となるオペレーション＊および｜を定義する。

要素が0，1からなるn次正方行列A，Bに対して，

(a) オペレーション＊の定義

行列A，Bに対して，C＝A＊Bの要素c_{ik}を次のように定義する。

$$c_{ik} = \bigcup_{j} (a_{ij} \cap b_{jk})$$

(b) オペレーション｜の定義

行列A，Bに対して，C＝A｜Bの要素c_{ij}を次のように定義する。

$$c_{ij} = a_{ij} \cup b_{ij}$$

このようにオペレーションを定義すれば，マトリックスM^kは，

$$M^k = \overbrace{M * M * \cdots\cdots * M}^{k\text{ヶ}}$$

と表わすことができる。

1. M^nの証明

コネクション・マトリックスMのn乗に相等するM^nの各要素が，2つのIMP間にnホップで行く道があるか否かを示していることを証明する。

証明　数学的帰納法により，証明する。

(i) $n=2$のとき,

$C=M*M$の要素c_{ik}を考えてみる。c_{ik}は,

$$c_{ik}=\bigcup_{j}(m_{ij}\cap m_{jk})$$

そこで, $c_{ik}=1$であれば, 論理積の項には少なくとも1つ値1のものがある。

それを$m_{ij_0}\cap m_{j_0k}$とすれば, $m_{ij_0}=m_{j_0k}=1$である。

すなわち, Mの定義からIMP iからIMP j_0へ行け, さらにIMP j_0からIMP k へ行けることを示す。従ってIMP iからIMP kへ2ホップで行けることになる。

また, $c_{ik}=0$であれば, 論理積の項の全ては0である。

これは, IMP iからIMP kへ2ホップで行けないことを示す。

もし2ホップで行けるとすれば, あるIMP j_1が存在して, IMP iからIMP j_1へ行け, さらにIMP j_1からIMP kへ行けるはずである。すなわちMの定義より$m_{ij_1}=1$, $m_{j_1k}=1$ でなければならない。これは, $c_{ik}=0$に矛盾する。

(ii) $n=m_0$のとき, M^{m_0}があるIMP iからIMP jへm_0ホップで行けるか否かを示す行列であることが証明されたとして, $n=m_0+1$のとき, M^{m_0+1}があるIMP iからIMP kへm_0+1ホップで行けるか否かを示していることを証明する。

$M^{m_0}=A$とおく。すると仮定により, Aの要素a_{ij}は,

$$a_{ij}=\begin{cases}1 & \cdots\cdots m_0\text{ホップでIMP iからIMP jへ行ける}\\ 0 & \cdots\cdots m_0\text{ホップでIMP iからIMP jへ行けない}\end{cases}$$

である。

そこで, $C=M^{m_0+1}=A*M$の要素c_{ik}は,

$$c_{ik}=\bigcup_{j}(a_{ij}\cap m_{jk})$$

そこで, $c_{ik}=1$であれば, 論理積の項には少なくとも1つ値1のものがある。

それを$a_{ij_2}\cap m_{j_2k}$とすれば, $a_{ij_2}=m_{j_2k}=1$である。

すなわちM^{m_0}, Mより, IMP iからm_0ホップでIMP j_2へ, さらにIMP j_2からIMPkへ1ホップで行けることを示す。従ってIMP iからIMP kへm_0+1ホップで行けることを示す。

また, $c_{ik}=0$であれば, 論理積の項の全ては0である。

これは, m_0+1ホップでIMP iからIMP kへ行けないことを示す。

もし, m_0+1ホップで行けるとすると, IMP iからm_0ホップであるIMP jへ行け, さらにIMP jからIMP kへ行けるはずである。すなわち, M^{m_0}, Mより$a_{ij}=1$, $m_{jk}=1$でなければならない。これは, $c_{ik}=0$に矛盾する。

2. マトリックス演算の簡略化の証明

さて，マトリックスM^kはkホップで行けるパスがあるか否かを示すことが明確になった。また，マトリックスM^2とMの各対応要素の論理和をとったマトリックス

$$M_2 = M^2 \mid M$$

の各要素は，2つのIMP間に2ホップ以内で到達する経路があるか否かを示す。

同様に，M^3，M^4，……，M^{16}を計算し，すべてのマトリックスの論理和

$$M_{16} = M^{16} \mid M^{15} \mid \cdots\cdots \mid M^2 \mid M \qquad \cdots\cdots (1)$$

をとれば，M_{16}の各要素$m16_{ij}$は，IMP iからIMP jに16ホップ以内で到達する経路があるか否かを示すこととなる。JIPNETにおけるIMPの最大数は16個であるから式(1)で求めたマトリックスに0の要素があれば該当するIMP iとIMP jとは互いにディスコネクション状態であることがわかる。

しかしながら(1)式の計算は少なくとも16回のマトリックス積とその論理和の計算を行なわねばならず，これを簡略化する計算法としてJIPNETに於ては，

$$\begin{aligned} M_2 &= M^2 \mid M \\ M_4 &= M_2 * M_2 \\ M_8 &= M_4 * M_4 \\ M_{16} &= M_8 * M_8 \end{aligned} \qquad \cdots\cdots (2)$$

の4回のマトリックス演算で済ませている。

以下に(2)で(1)を置き換える事が可能である旨の証明を行なう。

こゝで，式(2)を一般化して扱う。

マトリックスMを次の条件をみたすn次正方行列とする。

(i) 対角要素は0で，その他の要素は0または1である。

(ii) 対称行列である。

マトリックスM^k，M_2，$M_2{}^k$を次のように定義する。

$$M^k = \overbrace{M * M * \cdots\cdots * M}^{k\text{ケ}} = M^{k-1} * M$$

$$M_2 = M^2 \mid M$$

$$M_2{}^k = M_2{}^{k-1} * M_2{}^{k-1} \qquad (k \geqq 2)$$

こゝで以下の式が成立する事を証明する。

$$M_2{}^k = M^{2^k} \mid M^{2^k-1} \mid \cdots\cdots \mid M^2 \mid M \qquad \cdots\cdots (3)$$

証明は以下の(Ⅰ)，(Ⅱ)，(Ⅲ)のステップにわけて行う。

(I) 式(3)の証明の準備1. （オペレーション′の導入）

(A) オペレーション′の定義

非負の正方行列Aに対して，A′を次のように定義する。

$$A'\text{の要素}\quad a'_{ij} = \begin{cases} 1 & (a_{ij} > 0) \\ 0 & (a_{ij} = 0) \end{cases}$$

なお，constantも1次の特別な行列とみなす。

(B) 行列演算におけるオペレーション′に関する定理とその証明

非負行列A，Bに対して，次の定理1～4が成立する。

定理1. $(A')' = A'$

定義より明らか

定理2. $(nA)' = A' \quad (n > 0,\ \text{constant})$

定義より明らか

定理3. $(A+B)' = (A'+B')' = (A'+B)' = (A+B')'$

まず，$(A+B)' = (A'+B')'$について証明する。

A，Bの要素 a_{ij}，b_{ij}（$\geqq 0$）に対して，

$P = (a_{ij} + b_{ij})'$

$Q = (a_{ij}' + b_{ij}')'$

とおく。P，Qの値は0または1である。

$P = 0$とすれば，$a_{ij} = b_{ij} = 0$となる。 $\therefore Q = 0$

$P = 1$とすれば，a_{ij}，b_{ij}のうち少なくとも一方は正である。 $\therefore Q = 1$

$\therefore \quad (A+B)' = (A'+B')'$

なお，$(A'+B)' = ((A')'+B')' = (A'+B')' = (A+B)'$

$(A+B')$も同様である。

定理4. $(A \cdot B)' = (A' \cdot B')' = (A' \cdot B)' = (A \cdot B')'$

まず，$(A \cdot B)' = (A' \cdot B')'$について証明する。$C = A \cdot B$の各要素 c_{ik}は

$$c_{ik} = \sum_j a_{ij}\, b_{jk}$$

と表わすことができる。

いまこの両辺にオペレーション′を作用させる。

$c_{ik}' = (a_{i1} b_{1k} + a_{i2} b_{2k} + \cdots\cdots + a_{ij} b_{jk} + \cdots\cdots)'$

各項 $a_{ij} b_{jk}$を一次の特別の行列とみなせば，定理3より，

$c_{ik}' = ((a_{i1} b_{1k})' + (a_{i2} b_{2k})' + \cdots\cdots + (a_{ij} b_{jk})' + \cdots\cdots)'$

各項$(a_{ij} b_{jk})' = (a_{ij})'(b_{jk})'$は明らかであるから，

$$c_{ik}' = (a_{i1}' b_{1k}' + a_{i2}' b_{2k}' + \cdots\cdots + a_{ij}' b_{jk}' + \cdots\cdots)'$$

$$= (\sum_{j} (a_{ij}' b_{jk}'))'$$

これは，$(A \cdot B)'$の各要素c_{ik}'が，$(A' \cdot B')'$の要素に等しいことを示す。

$$\therefore \quad (A \cdot B)' = (A' \cdot B')'$$

なお，$(A' \cdot B)' = ((A')' \cdot B')' = (A' \cdot B')' = (A \cdot B)'$

$(A \cdot B')'$も同様である。

(C) 行列の演算におけるオペレーション＊と′の関係

要素が0と1から成る行列をA，Bとすれば，次の定理が成立する。

定理5. $A * B = (A \cdot B)'$

証明

$C = A * B$の要素c_{ik}は，次のように定義された。

$$c_{ik} = \bigcup_{j} (a_{ij} \cap b_{jk})$$

$$= (a_{i1} \cap b_{1k}) \cup (a_{i2} \cap b_{2k}) \cup \cdots\cdots \cup (a_{ij} \cap b_{jk}) \cup \cdots\cdots$$

一方，$D = (A \cdot B)'$の要素d_{ik}は，次のようになる。

$$d_{ik} = (\sum_{j} a_{ij} \cdot b_{jk})'$$

$$= (a_{i1} b_{1k} + a_{i2} b_{2k} + \cdots\cdots + a_{ij} b_{jk} + \cdots\cdots)'$$

c_{ik}，d_{ik}の値は0または1である。

いま$c_{ik} = 0$とすると，c_{ik}を構成するどの論理積の項$a_{ij} \cap b_{jk}$も0である。すると，

c_{ik}の項に対応するd_{ik}の項も0になる。　$\therefore d_{ik} = 0$

また，$c_{ik} = 1$とすると，c_{ik}の少なくとも1つの項$a_{ij} \cap b_{jk}$は1である。すると，

d_{ik}の対応する項は1である。　$\therefore d_{ik} = 1$

$$\therefore \quad A * B = (A \cdot B)'$$

さらに，$A * B * C = (A \cdot B \cdot C)'$が成立する。

$$\because \quad A * B * C = ((A * B) \cdot C)' = ((A \cdot B)' \cdot C')' = ((A \cdot B) \cdot C)'$$

(D) 行列の演算におけるオペレーション｜と′の関係

要素が0と1から成る行列をA，Bとすれば，次の定理が成立する。

定理6. $A \mid B = (A + B)'$

証明

$C = A \mid B$の要素，c_{ik}は，次のように定義された。

$$c_{ik} = a_{ik} \cup b_{ik}$$

一方，$D = (A + B)'$の要素d_{ik}は次のようになる。

$$d_{ik} = (a_{ik} + b_{ik})'$$

c_{ik}, d_{ik}の値は，0または1である。

いま,$c_{ik}=1$とすると，直ちに$d_{ik}=1$であることがわかる。

また$c_{ik}=0$とすれば$d_{ik}=0$である。

$\therefore \quad A \mid B = (A+B)'$

さらに，$A \mid B \mid C = (A+B+C)'$が成立する。

$\because \quad A \mid B \mid C = (A \mid B+C)' = ((A+B)'+C)' = (A+B+C)'$

(II) 式(3)の証明の準備2.

マトリックスM^kに関して定理7，定理8が成立する。

定理7. $M^i * M^j = M^j * M^i = M^{i+j}$

証明

(i) $i=1$，$j=2$のとき$M * M^2 = M^2 * M = M^3$を証明する。

$M^2 = M * M = (M \cdot M)'$であるから

$M * M^2 = (M \cdot (M \cdot M)')' = (M \cdot (M \cdot M))' = (M \cdot M \cdot M)'$

$M^2 * M = ((M \cdot M)' \cdot M)' = ((M \cdot M) \cdot M)' = (M \cdot M \cdot M)'$

$\therefore \quad M * M^2 = M^2 * M = M^3$

(ii) $i=1$，$j=m$のとき成立するとして$i=1$，$j=m+1$のときも成立することを証明する。

$$\begin{aligned} M * M^{m+1} &= M * (M^m * M) = (M \cdot (M^m \cdot M)')' \\ &= (M \cdot (M^m \cdot M))' = ((M \cdot M^m) \cdot M)' \\ &= ((M \cdot M^m)' \cdot M)' = (M * M^m) * M \\ &= M^{m+1} * M \end{aligned}$$

$\therefore \quad M * M^{m+1} = M^{m+1} * M = M^{m+2}$

(iii) $i=n$のとき成立するとして$i=n+1$のとき成立する事を証明する。

$$\begin{aligned} M^{n+1} * M^j &= M^n * M * M^j = ((M^n \cdot M)' \cdot M^j)' \\ &= ((M^n \cdot M) \cdot M^j)' = (M^n \cdot (M \cdot M^j))' = (M^n \cdot (M^j \cdot M))' \\ &= ((M^n \cdot M^j) \cdot M)' = (M^n * M^j) * M \\ &= M^j * M^n * M = M^j * M^{n+1} \end{aligned}$$

$\therefore \quad M^{n+1} * M^j = M^j * M^{n+1}$

即ち $M^i * M^j = M^j * M^i$ が証明された。

なお(iii)の証明中の展開の一部から

$M^{n+1} * M^j = (M^n * M^j) * M = M^{n+j} * M = M^{n+j+1}$

即ち $M^i * M^j = M^{i+j}$が証明された。

定理8. $M^3 = M^3 \mid M$

(i) 証明の準備（M^2 とMの関係）

M^2 の対角要素は0か1であるが，もし対角要素（i,i）が0であればMの第i行，第i列のすべての要素は0である。

何となれば $M^2 = M * M = (M \cdot M)'$ であるから

$C = M \cdot M$ の要素（i,i）を c_{ii} とすれば，

$$c_{ii} = \sum_j m_{ij} m_{ji}$$

Mは対称行列であるから，$m_{ij} = m_{ji}$ である。

$$\therefore \quad \sum_j (m_{ij})^2 = \sum_j (m_{ji})^2 = 0$$

即ち，$m_{ij} = m_{ji} = 0$ である。

(ii) $M^3 = M^3 \mid M$ の証明

いま，M^2 の対角要素のみを対象とした行列Pを考える。

すなわち，Pの要素 p_{ij} は，

$$p_{ij} = \begin{cases} 0 & (i \neq j) \\ q_{ii} & q_{ii} : M^2 \text{の対角要素}(i,i)\text{の値} \end{cases}$$

従って，行列Pの定義から，

$$M^2 = (M^2 + P)'$$

次に，行列PとMの積 $C = P \cdot M$ について考えてみる。

Pの対角要素以外は全て0であるから，Cの要素 c_{ik} は，

$$c_{ik} = \sum_j p_{ij} m_{jk} = p_{ii} m_{ik}$$

そこで，$p_{ii} = 1$ であれば，$c_{ik} = m_{ik}$ であり，また $p_{ii} = 0$ であれば $c_{ik} = 0$ であるが，このとき，前述したように $m_{ik} = 0$ でもある。

$$\therefore \quad P \cdot M = M$$

この結果を利用して，式 $M^2 = (M^2 + P)'$ の両辺にMをかけ，オペレーション $'$ を作用させる。

$$(M^2 \cdot M)' = ((M^2 + P)' \cdot M)' = (M^2 \cdot M + P \cdot M)'$$

$$\therefore \quad M^3 = (M^3 + M)'$$

$$\therefore \quad M^3 = M^3 \mid M$$

(iii) 式(3)の証明

前述のように，M_2，M_{2^k} を次のように定義する。

$$M_2 = M^2 \mid M$$

$$M_{2^k} = M_{2^{k-1}} * M_{2^{k-1}} \qquad (k \geq 2)$$

こゝで次式が成立する事を証明する。

$$M_{2^k} = M^{2^k} \mid M^{2^{k-1}} \mid M^{2^{k-2}} \mid \cdots\cdots \mid M^2 \mid M$$

証明　数学的帰納法による。

(i)　$k=2$ のとき，$M_4 = M^4 \mid M^3 \mid M^2 \mid M$　を証明する。

$$M_4 = M_2 * M_2 = (M^2 \mid M) * (M^2 \mid M)$$

$$= (M^2+M)' * (M^2+M)' = ((M^2+M)' \cdot (M^2+M)')'$$

$$= ((M^2 \cdot M^2)' + (2M^2 \cdot M)' + (M \cdot M)')'$$

$$= (M^2 * M^2 + M^2 * M + M * M)'$$

$$= (M^4 + M^3 + M^2)'$$

$M^3 = (M^3+M)'$ であるから

$$M_4 = (M^4+M^3+M+M^2)' = (M^4+M^3+M^2+M)'$$

$$\therefore \quad M_4 = M^4 \mid M^3 \mid M^2 \mid M$$

(ii)　$k=\ell$　即ち　$m=2^\ell$ の場合

$$M_m = M_{\frac{m}{2}} * M_{\frac{m}{2}} = M^m \mid M^{m-1} \mid \cdots\cdots\cdots \mid M^2 \mid M$$

が成立するとして，$k=\ell+1$ 即ち $2m=2^{\ell+1}$ のとき成立する事を証明する。

$$M_{2m} = M_m * M_m = ((M^m + M^{m-1} + \cdots\cdots\cdots + M^2 + M)^2)'$$

$$= (M^m \cdot M^m + M^m \cdot M^{m-1} + \cdots\cdots\cdots + M^2 \cdot M + M \cdot M)'$$

$$= (M^{2m} + M^{2m-1} + \cdots\cdots\cdots + M^3 + M^2)'$$

$M^3 = (M^3+M)'$ であるから

$$M_{2m} = (M^{2m} + M^{2m-1} + \cdots\cdots\cdots + M^3 + M^2 + M)'$$

$$\therefore \quad M_{2m} = M^{2m} \mid M^{2m-1} \mid \cdots\cdots\cdots \mid M^2 \mid M$$

以上で(1)の(2)による置き換えが，可能である事が証明された。

昭和50年3月発行

発行所　財団法人　日本情報処理開発センター
東京都港区芝公園3丁目5番8号
機械振興会館内
TEL（434）8211（代表）

印刷所　三協印刷株式会社
東京都渋谷区渋谷3丁目11番11号
TEL（407）7316

49－S001