

網向きプロセス間通信制御プロセッサの構成

和賀井フミ子 和田哲郎 小森 育 田中英彦 元岡 達
(東京大学 工学部)

第1章 序

計算機網の構成法に対する研究は各所で行なわれているが、当研究室に於いても、以前から、プロセス間通信機構を主軸とする計算機網TECNETと、網向きのOS(NOS)を開発してきた。^(1,2,3)

プロセス間のインタラクションは、メッセージを交換することにより行なわれる。その際、プロセスは相手プロセスの存在ホストを意識することなく通信できるという特徴を持つ。

現在は、網向きのOSの核機能をハードウェアでサポートするサブシステムについて考察し、実装を行なっているので、これについて報告する。

第2章

オペレーティング
システムのハード
ウェア化について

2-1

OSの機能分散と核の機能の分離

分散処理の一方式として、機能分散型計算機を考えることができる。これは、特定機能に専用化されたサブシステムを組合せて構成されるものである。この方法では、OSを機能別に専用マシンの上に分散する方式になっている。

例えば、当研究室においてはシステム全体を、機能サブシステム、システム管理サブシステム、データサブシステム、I/Oサブシステムの4つに分割した機能分散型計算機を構成し、種々の検討を行なっている。^(4,5)

ここでは、この考えを更に進めて、OSの基本となる部分、つまり核の機能分散について考察する。

2-2

核の機能と分離

分散処理向きのOSの核の機能とし

て必要と考えられるものは、一般に、マルチプログラミング、割込処理、仮想記憶、等の基本的な機能の他に、プロセスに対するサービス機能として、

1) プロセス間通信(同一システム内及び他システム間) 2) プロセス間の同期 3) プロセスの生成・消滅 4) プロセスに対する保護 5) プロセスの核利用インタフェスの提供、等がある。

プロセスには、核プロセス、システムプロセス、ユーザプロセスの3種類がある。

プロセス間のやりとりの道具としては、マクロが幾つか用意されている。

ここでは、上記のサービス機能の分離について考える。すなわち、プロセスにとっては重要だが、必ずしもホスト内で処理する必要のないこれらの機能を専用のハードウェアとともにホストから取り出し、ホスト外で処理するようにする。

このことにより、機能分散構造のサブシステム間インタフェスが明確になり、各サブシステムの制御プログラムが簡略化され、効率も上がる、と思われる。又、このような特定機能を専用ハードウェアとして設けたことで、これらの間のインタフェスが明確になれば、異種計算機に於いても同一の機能がそのまま使える、という利点が生ずる。

そこで、機能分離を幾つか試み、処理のし易さ、インタフェス、オーバーヘッド等の面から比較する。

2-2-1

分離に当たって考慮すべき事

1) 分散処理向きのプロセス間通信は他システム内のプロセスにも適用する必要があるので、できるだけホストから出した方がよい。

2) プロセスの保護機能は、プロセスの用いるマクロの引数に関して、セキュリティのチェック（通信や同期の可否）や値のバインディング（ユーザ使用論理値からシステム処理値へ）を行なうので、通信や同期機能と同じかホスト寄りに置く。

3) プロセスの同期管理は、プロセス管理テーブルを用いて、プロセスの状態遷移の管理を行なう部分である。

wait状態やready状態の切換は一部は他プロセスによるが、プロセス間通信によって行なわれることも多いので、通信機構と同じ所に置くと良い。

run プロセスの切換は、デイスパッチャによって行なわれるが、この為の機構（レジスタやアドレス回避領域や割込）は、ホスト内に必要である。

同期管理をホストから出した場合、run プロセスの切換の必要がある時だけホスト外から、割込等によってデイスパッチャに指示し、それ以外のプロセスの遷移は、ホスト外でプロセス管理テーブルのみを操作するだけでよいので、ホスト内でのオーバーヘッドはなくなる。

4) プロセスの生成・消滅については同期管理と同様な事が言える。すなわち、プログラムのロード等の機構はホストに置く必要があるが、その指示やプロセス管理テーブルの操作機能は、必ずしもホストに置く必要はない。

2-2-2 機能の分離法

以上の事に基づいて、下表のように5つのタイプに分けた。

これらのうち、比較的良好なのは、HIS4 と H4S1 である。

前者は、プロセスの諸サービス機能とプロセスが分離されている。後者はインタフェースハードウェアの面からは作り易い。

2-3

機能の分離によって生ずる利点

- 1) ホストのOSの負担が軽減される。
- 2) 特定機能の専用ハードウェアによる効率向上
- 3) ホストから取り出したことにより異種計算機にも、諸特殊ハードウェアの接続が容易になった。
- 4) 機能の階層化が明白になる。

第3章 NOS

3-1 NOSの現状

網向きOS・NOS I版は実装がほぼ終了し、ローカル通信、リモート通信ファイルシステム等の評価が行なわれている。(6)

第I版の構成は、H5S0タイプであり次節で述べるように幾つか改善する必要がある。第II版については、この点を考慮して、H4S1タイプを採用し、現在実装中である。

タイプ	ホスト内	外	特 徴
H5S0	IS, CAP PSN, C&D IPC	PT CCB	すべての処理をホストで行なうのでオーバーヘッドが大きい。プログラムモジュール間の境が不明確になりやすい。
H4S1	IS, CAP PSN C&D	PT CCB, IPC	プロセス間通信処理を専用のマシンで行なうのでホストの負担は軽減される。
H3S2	IS CAP C&D	PT CCB, PSN IPC	プロセス間通信に関してはH4S1と同様である。PTが外にあるので生成・消滅の処理に不便が生ずる。
H2S3	IS CAP	PT C&D PSN IPC	プロセス間通信に関してはH4S1と同様である。プロセスの生成・消滅と同期管理が同じ場所にあるのでPTの操作は便利になる。
HIS4	IS	PT CCB, CAP C&D, PSN IPC	H2S3と同様の特徴をもつ。保護機能もホストから出したことにより、ホストでは割込等の必要最低限の機構を除いて、純粋なプロセス空間となる。

注

IPC: プロセス間通信
PSN: プロセスの同期管理
C&D: プロセス生成・消滅管理
CAP: プロセスの保護
IS: 核利用インタフェース
ポート
PT: プロセス管理テーブル
CCB: 通信制御テーブル

3-2

NOS第I版のソフトウェア構成上の検討——実装法に関して

1) OSのすべての機能をホストに組み込んだ為プログラムが膨大になり、ミニコン(OKITAC4300C)ではメモリの増設が必要になったこと。(12kwのメモリを3回の増設、36kwにした)

特にプロセス間通信のプログラムではバッファ領域がかなりの部分(25kw)を占め、相対的にユーザ領域が小さくなっている。

2) OSをアセンブラやμコード等の低レベル言語で書いたもので、理解しにくく、デバッグにも非常に時間がかかったこと。又OS作成中に新たな機能が要求されると、その度追加していったのでプログラムがいびつになり、更に理解しにくくなっている。

3-3 NOS第II版の実装方針

前節の検討に基づいて、NOS II版を以下の方針で作成している。

3-2の1) に対しては、OSの中で比較的下層に置くことができ、専用マシンの設計やインタフェースが容易であるプロセス間通信機構をホストから取り出す方式(H4S1)を採った。

3-2の2) に対しては、OS記述向き的高级言語で書くことにより改善を画った。具体的には、OS設計段階では、Pascalを用いて書き下し(フローチャートは使用せず)、実装には、PL/Mを用いて記述した。又、プロセスレベルでは、C-Pascalを発展させた言語の開発を行なっている。(7)

第4章

プロセス間通信 サブシステムの 設計方針

4-1 プロセス間通信の方式

4-1-1 ユーザの立場から

プロセスとのおしはメッセージ単位で通信を行なうことができる。すなわちプロセスはSEND, RECEIVE等の通信用

マクロを用いて通信要求を出す。

送信要求を出したプロセスと、受信要求を出したプロセスの相手がお互いに一致すれば、そこで通信契約が成立し、メッセージの転送が行なわれる。この際プロセスは、相手プロセスの存在ホストを意識せずに通信できる。

4-1-2 システムの立場から

通信マクロの処理では、プロセスの提出した引数を論理値から、処理値に変換したり、通信の資格を調べたりする。(通信の capability) そこで通信可、ならば、通信相手の存在ホスト別に処理される。

ローカル通信(同一システム内)ならば通信契約成立以後、メッセージを送信領域から受信領域にコピーする。

リモート通信(他システム間)ならば通信要求をコマンドの形で他ホストのプロセスに伝え、通信契約成立後はメッセージをパケットに分割して転送する。

コマンドやメッセージパケットが到達したことの確認にACKを返している。

実際のパケット伝送には、伝送制御手順を使用し信頼性を向上するとともに、パケットの転送が無い時は、回線が接続されていることの確認の為に定期的に「遊むパケット」を転送している。

4-2

プロセス間通信に必要な機能

プロセス間通信機構では、主にプロセスが出した通信に関する情報を管理する部分と、実際の転送に関する管理をする部分との2つの機能に分けられる。

制御部には、個々の通信対応に通信制御テーブル(CCB)を置き、通信の状態やモードの管理や、ホストや受信コマンドからの通信情報の読み書き、通信制御の管理としてリモート通信の場合の他ホストへのコマンド作成やメッセージのパケット化の作業、ローカル通信の場合のメッセージの転送処理、

ホストに対して通信状況の通知、等の機能が必要である。

後者には、回線処理機能、すなわちパケットのルーティング、到達確認法（ACKや再送）、中継処理や、回線伝送手順の管理、等が必要である。

このような事から、プロセス間通信用の専用マシンは、制御部としてのCCP (Communication Control Processor) と回線用としてのLCP (Line Control Processor) との二階層として構成した。

尚、CCPとLCPは、OSIのモデルでは、それぞれSession, Transport及びNetwork, Datalink, Physical層に対応すると考えられる。

4-2-1

メッセージの転送法について

1) リモート通信

ホストからメッセージを取り出し、CCPでパケット化してLCPに送る方法は、CCPの負荷が大きいことや、ホスト $\leftrightarrow$ CCP、CCP $\leftrightarrow$ LCPと二回コピーを行なうので遅くなること等から得策ではない。そこでメッセージのパケット化はLCPで行なうことにした。すなわち、メッセージはホストとLCP間で直接受渡すこととし、LCPではCCPからのメッセージのパケット化情報に基づいて、パケットの分解及び合成を行なう。

2) ローカル通信

メッセージはホスト内主記憶での移動で済みパケット化の必要はない。そこでこの場合は、通信制御テーブルの効率的利用の面も考え合わせて、一旦ホストからメッセージを取り出し、CCPを経由して再びホストに書き込むという方式を採用することにした。

4-2-2

リモート通信のコマンド処理

ホストからの通信要求はCCPにおいて解釈し、SRQ (Send Request), RRQ (Receive Request) 等の計算機間コマンドを作成

してLCPに送る。又、他ホストからの受信コマンドはLCPを経由してCCPに通知され、CCPで解釈して必要があればホストに通知する。

4-2-3

CCPで回復可能なエラー処理

制御テーブル等の領域が不足している場合は、自ホストに対してはその旨の通知、他ホストに対してはRejectコマンドを送り返している。

通信のタイムアウトの場合、メッセージの転送は終了しているのに一定時間経てもMAK (Message ACK) が来ない時はCancelコマンドを両プロセスに発行する。又、通信要求を出したが、一定時間以上経っても相手から通信要求が来ない時は、Rejectを発行する。等のことを行なっている。

4-2-4

LCPの処理

LCPでは、パケットの分解、合成及び伝送制御を行なう。伝送制御手順はNOS I版では独自のものを使用していたが、効率等を考慮してHDLCを採用し、非同期平衡モードを簡略化したものを考えた。即ち、フレームとしてはRR, RNR, I, SABM, UAの5種類だけで切断モード、初期モードは省略した。

又、再送については、必ず抜けたパケットから後をすべて送る方式を採用しているので伝送制御の効率は下がるがこうすることにより、パケット番号は常に順番が保たれるので、組み替えをする必要がなくなり、制御プログラム自体も簡単になるのでエラーが無い場合のスループットを上げる効果がある。

4-2-5

制御手順を用いない転送

ユーザからの通信要求にimmediateの指示があった場合は、SRQ, RRQ等の手続きを踏まずに、いきなり転送を行なうことができる。

4-3 ハードウェア構成

4-3-1

基本アーキテクチャ

基本アーキテクチャとしてマルチマイクロプロセッサを考える。マルチプロセッサにした意味は、論理階層をそのままハードウェアに対応づけることにより、システム全体の階層が明確になり、ソフトウェアも作り易くなるからで、またマイクロプロセッサを使用したのは、価格性能比及び扱い易さからの要請である。構成法の検討の際には、次のような点に留意した。

- i) 論理階層とハードウェアの対応
- ii) 転送ネットワークの回避
- iii) モジュール構成、特にモジュール間接続の単純化
- iv) 全二重、80kbps の回線及び組サポートを要すること。

以上の方針に基づいて、まず論理階層を図4.3.1のようにマッピングした。次に高速回線をサポートするの、回線対

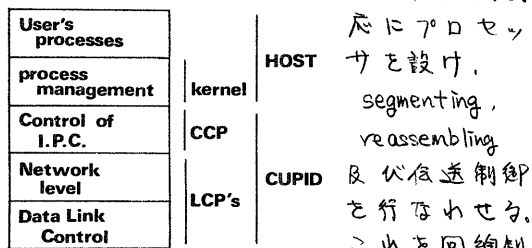


fig. 4.3.1

logical hierarchy of NOS II 御プロセッサ (Line Control Processor: LCP) と呼ぶ。そして LCP's の全体制御及びローカル通信を含めたプロセス間通信の管理を行うプロセッサを通信制御プロセッサ (Communication Control Processor: CCP) と呼ぶ。こうして1つの CCP モジュール、複数の LCP モジュールそして HOST 固有モジュールより構成されたシステムを CUPID (Control Processor of Interprocess communication for Distributed system) と呼ぶ。

4-3-2 結合方式

HOST-CCP-LCP の結合方式は、転送のオーバーヘッドを小さくする上で

非常に重要な問題である。次のような方式を検討した。

まず、HOST-CUPID 間の結合は DMA チャンネルとプログラムチャンネルの二本立にして、前者を、メッセージ本体及び制御情報の転送に用い、後者を、HOST-CUPID 間のコマンド/レスポンスの転送に用いる。

次に、CUPID の内部結合であるが、第1案として CCP-LCP 間をプログラム転送による I/O ポートで接ぐ方式を検討した。しかし、この方式ではオーバヘッドが大きくなり、回線速度に間に合わない可能性がある。そこでメッセージ本体の転送に DMA を使用して高速転送を行う方式が第2案である。しかし、この場合メッセージ本体は一度 CCP を経由して LCP に転送されるので全体のスループットは CCP の処理能力によっておさえられ、その値は回線速度を下回ってしまう。以上のことから明確にコマンドとメッセージ転送ルートを分離し、LCP も直接 HOST の DMA チャンネルに接続するという第3案を検討した。CCP と LCP が8台で9か1のマルチプロセッサがデータ分配に必要になるので HOST 固有モジュールに含ませた。

4-3-3

第3案の詳細検討

この検討は、メッセージが処理されるフローを追うことより始めた(図4.3.2)。

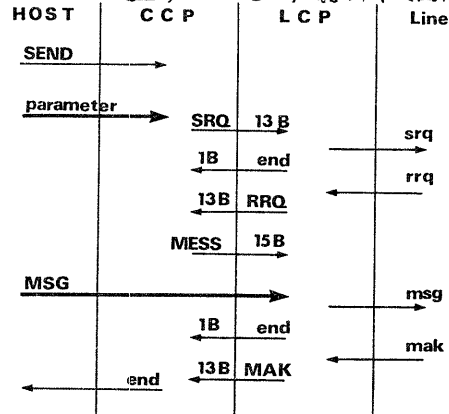


fig. 4.3.2 The flow of msg. processing

4-3-4 最終案

[illegible]

第5章 プロセス間通信
サブシステムの構築

5-1 研究室の計算機

PPS-1 と OKITAC 4300C との間で一対
PPS-1 と FACOM-U300 との間で一対
PPS-1 は主記憶を共有する3台のフ
ロセッサから成るポリフロセッサであ
り、OKITAC と U はミニコンである。

5-2 ハードウェア

5-2-1 実装法

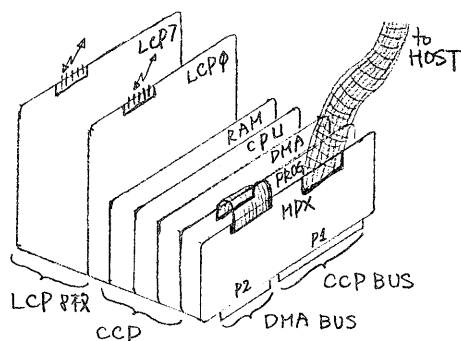


fig. 5.2.1 突裝圖

- 6 -

P1をCCP BUS, P2をDMA BUSとして使用している。なお、CPU及びRAMは市販品を利用しているが、その他はすべて自作で、DMA, PROG, MPXはラッピン、LCP及びマザーボードは、アートワークデザインまでを行って、外注したプリント基板である。

CCP及びLCPのCPUとしては、現時点で最も価格性能比のよいZ-80A(4MHz)を採用した。次に各モジュールの内部構成について述べる。

5-2-2 内部構成

i) HOST固有モジュール

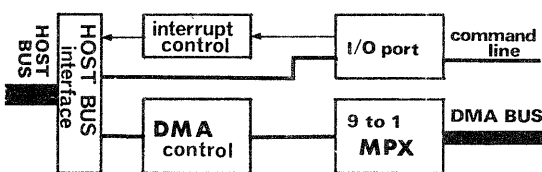


fig.5.2.2

MPX基板にまとめられているこのモジュールは、HOSTとのバスインタフェース、DMA制御、MPXer、コマンド/レスポンス用ポート及び割込み制御の各部より構成されている。MPXerを除いてHOSTに依存した形で実装される。通常TTL 90%程度を要する。HOSTとしてPPS-1, OKITAC 4300c, U-300を考慮しており、U-300についてはこのまま実装する予定であるが、他の2つについては従来のハードウェアの一部が適用できるため、この基板に実装されるのは、DMA BUSへの配線だけとなっている。

ii) CCPモジュール

CCPは、CPU, RAM, PROG, DMAの各基板から構成されている。CCP BUSのマスターはCPUとDMAであり、その制御にシリアルアービターを用いている。

○CPU: 割込み優先度の制御のためにi8259A, 時間監視のためにタイマーi8253を使用している。(LSI 35, SSI 407)

○PROG: コマンドライン用のポートとしてi8255Aを、LCPからの割込みをまとめて制御するためにi8259Aを

スレーブモードで使用している。また、デバッグ用としてモニタROM 4KB及びコンソール用のi8251Aがある。将来CP/M®をのせるためにパワーオンジャンプもついている。(LSI 57, SSI 377)

○DMA: この基板は、DMACを除いてTTLで組んであり、最も高密度実装されている。P.M.M.は4BのDMAのために、メモリバンクを選択するメモリバンクレジスタ、アドレスバッファ、データ用双方向バッファ及び制御回路より構成され、CCPより8KBを4バンクとしてHOSTの主記憶にアタスびをなようになっている。反射機構はソースアドレス、デスティネーションアドレス及びバイトの各カウンタとテンポラリーレジスタ及び制御回路より構成されHOSTのM. & H.のDMAをこなようになっている。

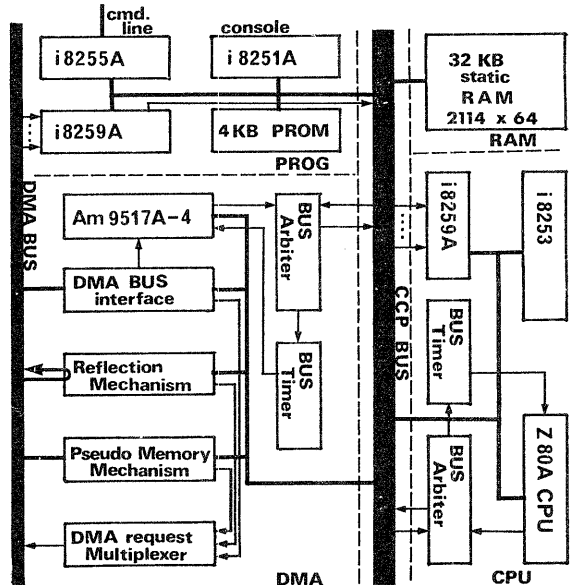


fig.5.2.3 The structure of CCP

DMA BUSインタフェースは、HOST-CCP間をワード単位でDMAが行なえるような制御ロジックである。また、以上の種の機構からDMA req.が出るので、そのMPXをするロジックが付け加えられている。DMACとしては、i8257の改良版であるAm 9517Aを使用し、CH0及び1をCCP内

M. to M. に, CH2 を read, CH3 を write に割り当てている。(LSI 1ヶ, SSI 10ヶ)

◦ RAM: 32KB を 4kbits static RAM 64ヶで構成している。(RAM 64ヶ, SSI 16ヶ)

メモリマップを図5.24に, 割り込み優先度を以下に示しておく。

higher
↑
N.M.I.: Bus time out (10 msec)
INT0: Timer
INT1: LCP φ~7 (φ~7間は rotating)
INT2: HOST (read/write)
INT3: DMAC (end/Bus time out)
lower
INT4: RM

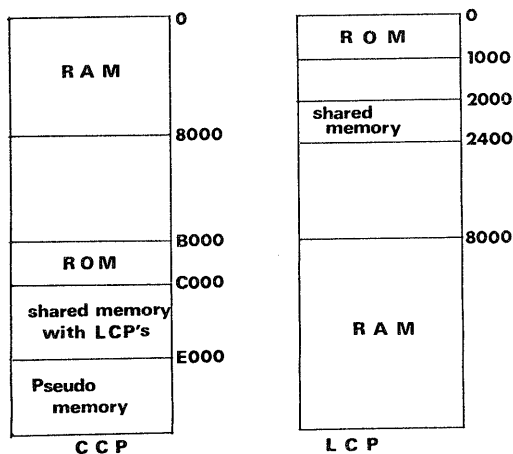


fig.5.24 memory map

iii) LCP モジュール

LCP の構成の主眼は, 80Kbps の全二重 HDLC の回線のサポートである。そのためデータ量の多いメッセージ本体に関しては, 徹底的に DMA を使用し, CPU はデータの転送を一切しないですむようにしてある。

また, 4-3-3 の検討にみるように CCP との共有メモリを LCP 基板上に設けている。(1KB static RAM) 相互排他の方は, 以下の通りである。制御回路は, LCP のアクセス要求と CCP のアクセス要求を LCP のクロックに同期して交互に検査する。アクセス要求が発見されると, メモリサイクルが開始し, として終了すると検査は逆側を調べることになる。競合に対して LCP には walt ライン, CCP には

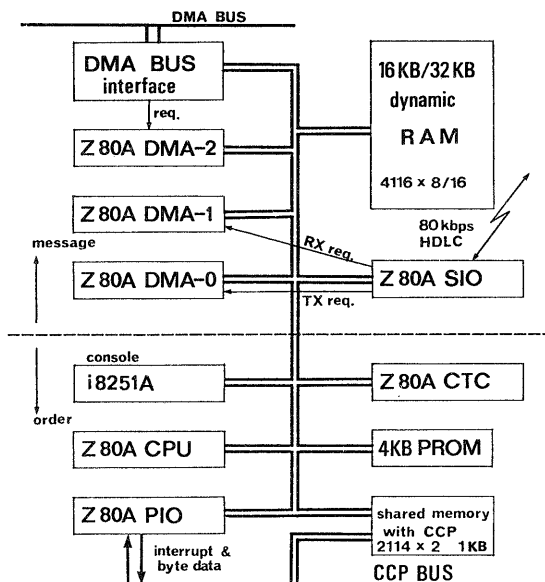


fig.5.25 The structure of LCP

XACK を行っている。

DMA BUS インタフェースは, CCP のそれと全く同じでアドレスカウンタ (16bits) モードレジスタ (R/W, Byte/Word), 入出力用ラッチ, 8bits ↔ 16bits の変換のための順序回路より構成されている。

DMAC としては, Z80A DMA を使用している。SIO の Rx 用, Tx 用に各 1ヶ, そして HOST-LCP 間の転送用に 1ヶ (両方向を切替えて使う) の計 3ヶを使用している。また, HDLC モジュールとする LSI としては, Z80A SIO を使用した。

デバッグ用ユニソールのためのボーレート分周及び低速制御におけるタイマ用に Z80A CTC を, また CCP-LCP 間割り込みとバイト転送のために Z80A PIO を使用した。

メモリマップを図5.24に, 割り込み優先度を以下に示す。

higher
↑
◦ SIO Rx. (receive one character)
◦ SIO Rx. (special receive condition)
◦ DMA Tx. (end) DMA0
◦ DMA Host (end) DMA2
◦ PIO in (order request)
◦ PIO out (response accepted)
lower
※優先度は daisy chain にする。

5-3 ソフトウェアの実装

5-3-1 インタフェース

- 1) ホストから CCP へ (hostparam)
プロセス間通信のユーザからの通信要求に関する引数
- 2) CCP からホストへ (ccpparam)
通信の状態をプロセスに通知する
- 3) CCP から LCP へ (lcpparam)
送信コマンドとメッセージ管理情報
- 4) LCP から CCP へ (lcpparam)
受信コマンドとメッセージ処理済情報

5-3-2 プロトコル

1) ユーザ

実際のシンタックスはプログラミング言語によるが、通信用マクロとして、SEND, RECEIVE, REJECT, CANSEL, RESETを用いている。

マクロの引数は、例えば SEND ならば、通信相手名(論理値)、通信の領域や量やタイプ、リンク等を書き、ポートも用いることができる。⁽⁷⁾

2) CCP レベル

リモート通信の場合の制御コマンドには、SRQ, RRQ, CRQ (Continue ReQuest), REJCT (REJECT), RST (ReSeT), CLOS (=cansel) MAKがあり、相手の CCP とやりとりをして、プロセス間通信の制御を行なう。
(図 4.3.2)

3) LCP レベル

ヘッダとテキストとから成るパケットを送受の単位とし、HDLC を伝送制御手順として用いる。

5-3-3 制御テーブル

CCP には通信制御テーブルがあり、LCP にはメッセージ管理テーブルがある。

5-3-4 プログラム

1) CCP では、主にホストからの情報を解釈し、通信制御テーブルの読み書きや、通信の状態制御を行ない、必要ならば LCP にコマンド等を送り出す部分と、逆に LCP からのコマンド等に基づいて同様な処理をし、必要ならば

```
COMKIND = (SRQ,RRQ,MAK,CRQ,RST,REJCT,CLOS,
            RECANY,RECSELECT,MESS,MESR,MESL,
            LOSRQ,LORRQ,LOCRO,LORST,LOREJCT,
            LOCLS,SENBROAD)
```

```
COMMODE
link : (once,continuous) ;
combuffer : (basic,segmenting) ;
commember : comkind
```

```
IDENTITY
host : hostnumber ;
processid : processnumber ;
portid : portnumber
```

```
AREA
blocklength : blockinteger ;
addr : addinteger
```

```
HOSTPARAM
usermsg : messinteger ;
hostmode : commode ;
hostid : identity ;
hostarea : area ;
selofremoteid : array (.1..5.) of identity
```

```
CCPPARAM
length : integer ;
usermsg : messinteger ;
hostmode : commode ;
hostid : identity ;
remoteid : identity ;
infcode : array (.1..3.) of integer ;
addcode : array (.1..3.) of blockinteger
```

```
LCPPARAM
lcpmode : commode ;
hostid : identity ;
remoteid : identity ;
sysmsgnum : messinteger ;
remotearea : area
```

COMBLOCKTABLE

```
ownnumber : cominteger ;
chainnext : addinteger ;
presentstatus : comstatename ;
usermsg : messinteger ;
hostmode : commode ;
remotemode : commode ;
hostid : identity ;
remoteid : identity ;
sysmsgnum : messinteger ;
transarea : area ;
receivearea : area ;
addition : array (.1..5.) of identity
```

MESSAGETABLE

```
hostmode : commode ;
remotemode : commode ;
hostid : identity ;
remoteid : identity ;
sysmsgnum : messinteger ;
messarea : area ;
nummesp : array (.1..pnum.) of imep
```

PACKET

```
pabmode : commode ;
hostid : identity ;
remoteid : identity ;
sysmsgnum : messinteger ;
msglength : blockinteger ;
packnum : integer ;
textlength : integer ;
textcontents : array (.1..maxlength.) of char
```

```
IMEP
meptarea : area ;
packetnum : integer ;
done : boolean
```

ホストに知らせる部分 R、から成る。
他に割込処理などのルーチンがある。

この部分は PL/M で記述され、T で 800 行、R で 500 行、コンパイルすると 12.6 kB 程度になる。

2) LCP では CCP からのオーガを解釈し、メッセージ制御テーブルへの書き込みや、計算機間コマンド及び、メッセージのパケット化を行なう部分①と、受信したパケットをメッセージ化してホストに転送したり、計算機間コマンドの場合には、CCP に知らせる部分②から成るメッセージ制御プログラムをバックグラウンドで走らせ、伝送制御手順を実行する HDLC プログラムを割込処理に含ませている。

この部分は、現在コーディング中であるが、コード効率が、スルーポイントに非常に影響するので、アセンブラで記述している。

5-4 OS 記述言語の比較

NOS I 版は、アセンブラやムコードで書かれているが、第 II 版は前述(3-2)のデバッグ効率の点などを考慮して、まず、Pascal を用いて論理構造を記述し、次いでそれを PL/M に変換する、という方法を採用した。これらの言語の使用経験から、言語の比較をすると、以下の通りである。

書き易さ Pascal が最も良い。
array や record タイプの要素がすべて identifier で記述でき、ローケーションを全く気にしなくてよい。これに比べて PL/M は array(n) の形で指定する必要がある。

if-then-else や case 文を使用することにより場合分けの漏れが防げる。

Pascal では状態遷移図を見ながら即プログラムに書き下すことができる。

デバッグ アセンブラでは命令と番地とが一对一に対応しているので動作の場所が直接的にわかるが、PL/M では、プログラムのソースの場所と、

実際の番地との変換表を見ながら、アセンブラのモニタコードを解釈しなければならない点でめんどうである。しかし一つのルートについてデバッグできると、他のルートについては、同様の考え方が流用できる点では楽になる。

記述日数 第 I 版のアセンブラの場合は、設計しながら記述していったのでそのまま比較できないが、約 1 年かかり、Pascal は状態遷移図を見ながら約 2 か月、PL/M は Pascal から変換して約 1.5 か月かかった。

第 6 章 現状及び評価

現在、ハードウェアはほぼ完成しており、HOST に接続してデバッグを行なっている。また、ソフトウェアは CCP がデバッグ途中、LCP がコーディング中である。

実装と平行して、スルーポイントの推定を行なっているが、リモート通信の場合には、回線速度で、ローカル通信の場合には、DMA BUS のスルーポイントをそれぞれおさえられるが、ローカル、リモートをみれば、単純にはならず、CCP のオーバーヘッドによって 20% 程下がり、約 340 kB/sec 程度であろうと思われる。

参考文献

- 1) 田中・元岡「研究用電子計算機網 TECNET」
信学研資 EC 73-57
- 2) 和賀井・田中・元岡「網向きオペレーティングシステム
についての一考察」信学研資 EC 77-43
- 3) H.TANAKA & T.MOTO-OKA: Network Oriented
Operating System of TECNET
J. of the fac of Eng. Univ of Tokyo (B)
Vol. XXXV No.4 1980
- 4) 正井・田中・元岡「機能分散型計算機における
入出力サブシステムの試作」信学研資 EC 77-46
- 5) 吉田・田中・元岡「機能分散型計算機における
データサブシステム」分散処理システム研究会 5-4, 1980
- 6) 本田公男「分散ファイルシステムにおける重複データ管理方式」
東大工学部 修論 1980
- 7) 大和・田中・元岡「分散処理システム記述言語の
設計とそのカーネルの製作」情処 56 年前期全国大会
2E-7