教育・研究用マイクロコンピュータネットワーク HOLENET

重松保弘、柴田芳樹、小出真
（九州工業大学情報工学科）

1、まえがき

HOLENET(HDLC Oriented Local Area Experimental Microcomputer Network)は、筆者らによって開発された、ハイレベル伝送手順に基づく実験用マイクロコンピュータネットワークであり、その主な目的は以下に挙げる5点である。

(1) 情報工学科の学生に対して、標準的なネットワークソフトウェア技法を教育し、実習させるための設備を提供する。

(2) 従来の集中型教育用計算機システムでは困難であった並行プロセスやモニタシステム等に関する実験、実習のための設備を提供する。

(3) ローカルコンピュータネットワークにおける通信規約(Communication Protocol)の研究、実験のための設備を提供する。

(4) コンピュータネットワークに基づく分散処理（機能分散、ファイル分散等）システムの研究、実験のための設備を提供する。

(5) 大学キャンパス内の中型コンピュータやマイクロコンピュータ等をネットワーク化することにより、これらのコンピュータ間でリソースの有効利用をはかる。[1)]

HOLENETは、これらの目的システムの核を構成するシステムであり、これらの目的システムに適応できる柔軟性を持っている。本稿では、HOLENETの構成、ネットワークソフトウェアの開発支援システムについて述べる。

2. HOLENETの構成

HOLENETは、図1に示すように、5台のホスト・マイクロコンピュータシステム（以後、ホストと呼ぶ）と5台のCCP(Communication Control Processor)から構成される。

各ホストは、Z-80マイクロプロセッサー、IPL用ROM(2KB)、RAM(64KB)、DMAコントローラ、インタバルタイマ、及び周辺入出力装置（コンソールディスプレイ、標準フロッピーディスク、ドットプリンタまたはディジーホイールプリンタ、PROMライタ）等で構成される。また、各CCPは、図2に示すように、Z-80マイクロプロセッサー、ROM(2KB)、RAM(64KB)、DMAコントローラ、インタバルタイマ、SDLC/HDLCプロトコルコントローラ、割込コントローラ等で構成される。

HOLENETのシステム構成上の特徴は、ホストとCCPがハードウェア的に相互に独立している点である。すなわち、各ホストは、TSSマルチプロセスモニタ（ディジタルリサーチ社MP/Mシステム[2)]）を基本とするマルチプロセスシステムとして独立に動作し、環状に結合されたCCPはホストと独立に、HOLENETのサブネットとして動作することができる。従って、学生実験等において、いくつかのグループはホストを使ってモニタシステムの作成を行い、これと並行して他のグループはCCPを使って通信制御プログラムの作成を行う、といった実験形態が実現できる。またMP/Mでは、ユーザプロセスの組込みが容易であり、プロセス間通信やプロセ

ス間同期の手段も与えられているので、これらを利用してホスト－ホスト間でプロセス間通信も比較的容易に実現できる。

3. ホスト－CCP間インタフェース

ホストとCCPは、インタフェースレジスタ（1バイト×4）を介して相互に通信を行うことができる。ただし、ホストもCCPもインタフェースレジスタを任意の時点でアクセスできるために、ホストとCCP間の通信方法によっては、アクセスの競合が発生する場合がある。そこで、インタフェースレジスタには図3に示すように相互排斥回路を付加して、アクセスの競合を防いでいる。

相互排斥の制御は、CCPアクセスサイクルかホストアクセスサイクルのいずれかを選択することによって行われる。CCP（またはホスト）アクセスサイクルは、CCP（またはホスト）に対して、インタフェースレジスタのアクセスを許すサイクルである。各状態、及び各状態の入力出力の符号化を表1に示す。

相互排斥制御回路は、8ビット長の水平型マイクロプログラムによって制御される。マイクロプログラムの上位6ビットはオペレーション部であり、下位3ビット（最上位部はオペレーション部と兼用）は飛先番地部である。飛先番地部には現在の状態符号が、またオペレーション部には現在の出力符号が、それぞれ埋められている。また、マイクロ命令を格納するROMの番地は、遷移直前の状態符号と入力符号から決定される。従って、ROMの番地とその内容（マイクロ

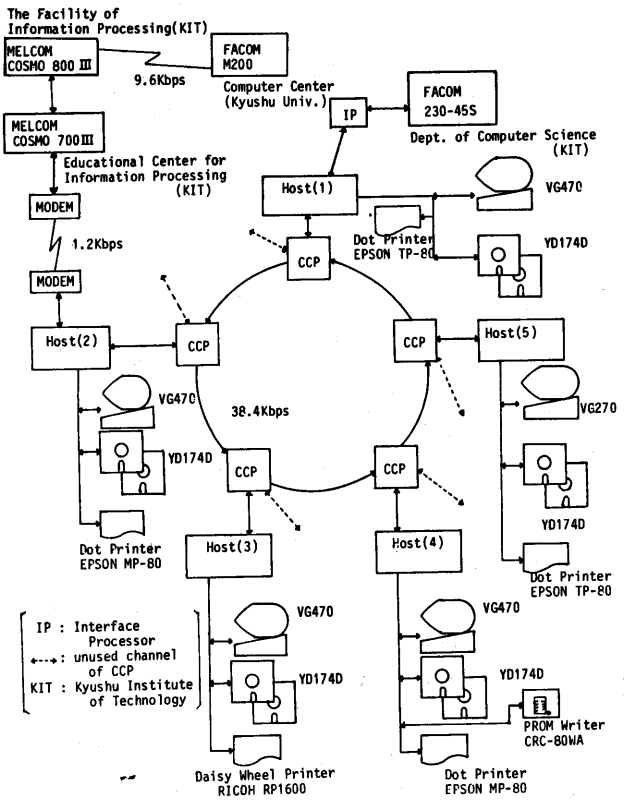


図1　HOLENETのシステム構成

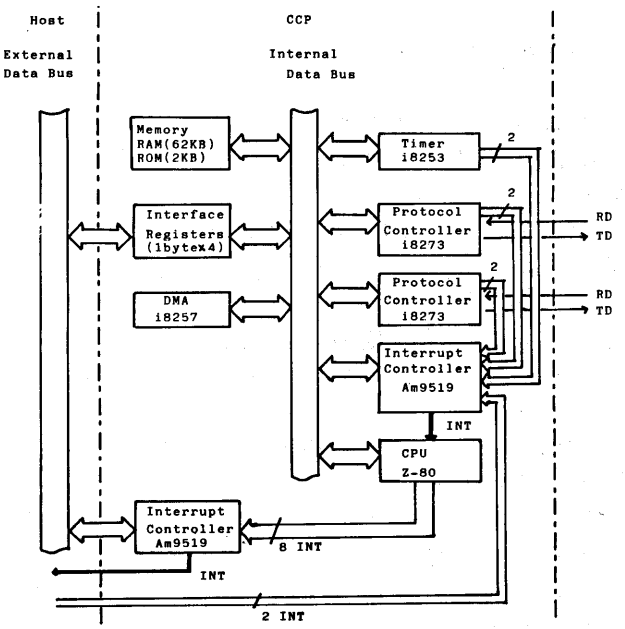


図2　CCPのブロックダイヤグラム

命令）は次のようになる。

［address］　　［imstructiom］

imput || old address　；　output of curremt state || currem state code

ただし、||記号は、結合記号である。

相互排斥を行うマイクロプログラムを図5に示す。

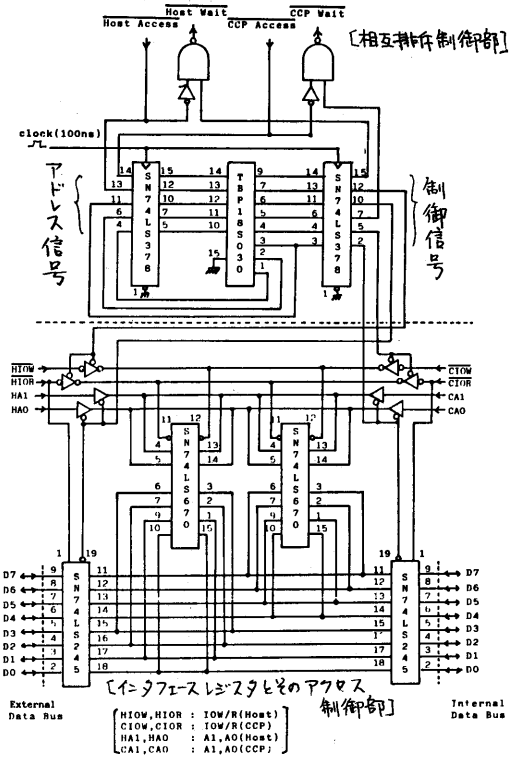


図3 インタフェースレジスタとその相互排斥回路

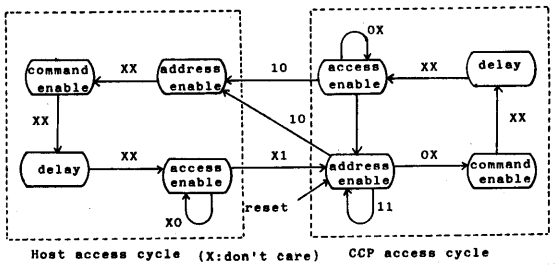


図4 インタフェースレジスタのアクセスサイクル状態遷移図

表1　図4の状態、及び入力出力の符号化

(1) 状態と出力

	state	state code	output
CCP access cycle	address enable	000	111110
	command enable	001	111100
	delay	010	111100
	access enable	011	111000
Host access cycle	address enable	100	110111
	command enable	101	100111
	delay	110	100111
	access enable	111	000111

(2) 入力

input	access
00	by CCP and Host
01	by CCP
10	by Host
11	---

以下の章では、4個のインタフェースレジスタを、各々便宜上、下記の名前で呼ぶ

HCR : Host Commamd Register
HDR : Host Data Register
CCR : CCP Commamd Register
CDR : CCP Data Register

```
*******************************
* Interface Registers         *
*  Mutual Exclusion Program   *
*******************************
States Address      Contents
      <43 210>     <765432 10>
[000]  11 000       111110 00
       0X 000       111100 01
       10 000       110111 00
[001]  XX 001       111100 10
[010]  XX 010       111000 11
[011]  0X 011       111000 11
       11 011       111110 00
       10 011       110111 00
[100]  XX 100       100111 01
[101]  XX 101       100111 10
[110]  XX 110       000111 11
[111]  X0 111       000111 11
       X1 111       111110 00
```

```
*******************************
*       ROM Contents          *
*******************************
Add.  Data      Add.  Data
00000 11110001  10000 11011100
00001 11110010  10001 11110010
00010 11100011  10010 11100011
00011 11100011  10011 11011100
00100 10011101  10100 10011101
00101 10011110  10101 10011110
00110 00011111  10110 00011111
00111 00011111  10111 00011111

01000 11110001  11000 11111000
01001 11110010  11001 11110010
01010 11100011  11010 11100011
01011 11100011  11011 11111000
01100 10011101  11100 10011101
01101 10011110  11101 10011110
01110 00011111  11110 00011111
01111 11111000  11111 11111000
```

図5 インタフェースレジスタ相互排斥マイクロプログラム

4. IPL機能

本章では、ホストからCCPへ通信制御プログラムを転送するIPL機能について述べる。HOLENETには2種類のIPL機能が用意されている。すなわち、ローカルIPL機能とリモートIPL機能である。前者では、CCPに接続されたホストからIPLが行われる。また、後者では、一次局CCPから二次局CCPへ通信制御プログラムを転送することによって、二次局CCPへのIPLが実行される。この場合、一次局

CCPへは、そのホストからローカルIPLを行う。

4.1 IPL用コマンド

IPLを行うためにホストからCCPへ送られるコマンド、パラメータ及びその機能を表2に示す。

表2 IPL用コマンド、パラメータ及び機能

command	parameters	function
IPL	load address, program	load the program on CCP(s)
RUN	start address	start the program at the start address
SRIPL	---	Set remote-IPL mode
RRIPL	---	Reset remote-IPL mode

〔コマンド送信〕 ホストからCCPへコマンド送信を行う時、ホストはコマンドをHCRに書込んだ後、CCPに対してコマンド割込を発生させる。CCPはコマンド割込を受付ると、HCRからコマンドを読出し、受諾すればACKレスポンスをCCRに書込んだ後、ホストに対してコマンド割込を発生させる。

〔パラメータ送信〕 コマンドがパラメータを持つ場合、CCPはデータ割込を受付けると、パラメータをHDRから読み出した後、これをCDRに書込むことでエコーバックを行い、ホストに対してデータ割込を発生させる。

4.2 ローカルIPL手順

ホストのローカルIPLプログラムは、CCPにIPLコマンドを送った後、コンソールより指定された実行形式プログラムファイルをフロッピーディスクから読出してCCPに送出する。その後、RUNコマンドをCCPに送り、プログラムの実行開始を指示する。ローカルIPLの手順を図6に示す。

```
Host            Interface      CCP
                  Register

IPL command     --->[HCR]
command-int.    ------------>
                    [HCR]---> IPL command
                    [CCR]<--- ACK response
                <------------ command-int.
response        <---[CCR]
address(high)   --->[HDR]
data-int.       ------------>
                    [HDR]---> H
                    [CDR]<--- H
                <------------ data-int.
address(low)    --->[HDR]
data-int.       ------------>
                    [HDR]---> L
                    [CDR]<--- L
                <------------ data-int.
  / data        --->[HDR]
r | data-int.   ------------>
e |                 [HDR]---> data
p |                           data --> (HL)
e |                           data <-- (HL)
a |                           increment HL
t |                 [CDR]<--- data
  \             <------------ data-int.
RUN command     --->[HCR]
command-int.    ------------>
                    [HCR]---> RUN command
                    [CCR]<--- ACK response
                <------------ command-int.
response        <---[CCR]
address(high)   --->[HDR]
data-int.       ------------>
                    [HDR]---> H
                    [CDR]<--- H
                <------------ data-int.
address(low)    --->[HDR]
data-int.       ------------>
                    [HDR]---> L
                    [CDR]<--- L
                <------------ data-int.
                              jump to (HL)
```

図6 ローカルIPL手順

4.3 リモートIPL手順

リモートIPL手順における標準的なコマンド列は次のようになる。

SRIPL IPL RUN RRIPL IPL RUN

まずSRIPLコマンドが一次局CCPによって受諾されると、一次局CCPはリモートIPLモードになり、以後IPL及びRUNの各コマンドは、二次局へのリモートIPLを指示するコマンドと解釈される。またこのとき一次局CCPは、全二次局CCPにSIM(Set Initial Mode)コマンドフレーム[4]を送出し、全二次局CCPを初期状態に設定する。リモートIPLモードにおいてIPLコマンドを受取った一次局CCPは、続いてホストから送られてくる実行形式プログラムを128バイト単位でブロック化し、UI(Unnumbered Information)コマンドフレーム[4]を構成し、全二次局CCPに対して送出する。リモートIPLモードにおいてRUNコマンドを受取った一次局CCPは、IPLされたプログラムの実行を指示するUIコマンドフレームを全二次局CCPに送出する。

RRIPLコマンドの受信によって一次局CCPはローカルIPLモードに戻り、続くIPLコマンドで一次局CCPの通信制御プログラムを受取り、RUNコマンドでその実行を開始する。

リモートIPL手順における一般的なコマンド列は次のようになる。

SRIPL IPL* [RUN] RRIPL IPL* RUN

ただし　IPL* = 入, IPL, IPL, …

[RUN] = 入, RUN

UIコマンドフレームにおける情報フィールドの内容、リモートIPLにおける二次局CCPの状態遷移図を各々図7、図8に示す。またリモートIPLにおける一次局CCPとそのホスト間の手順を図9に示す。

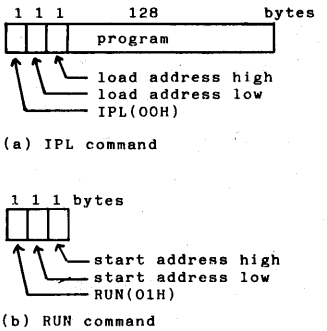
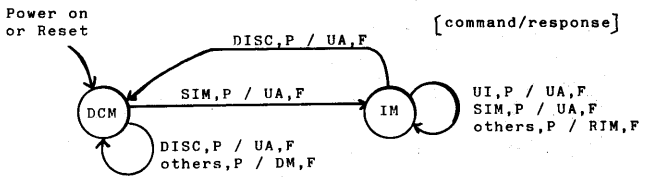


図7 UIコマンドフレームにおける情報フィールドの内容



（記号は文献3),4)に基づく）

図8 リモートIPLにおける二次局CCPの状態遷移図

5、インタフェース・ファンクション

HOLENETは実験用システムであり、これを使用して多くの利用者がネットワークソフトウェアを作成することになる。HOLENET上でネットワークソフトウェアを作成し、デバッグ等を行う場合に発生するいくつかの問題点を以下に考える。

(1) CCPにはコンソールディスプレイ等の周辺入出力装置が接続されていないので、CCP上のプログラムのデバッグは、ホストの周辺入出力装置を使って行わなければならない。

(2) ホストのプロセス（一般には複数のプロセス）とCCPのプログラム間で通信を行うには、イン

Host	Interface Register	CCP
SRIPL command	---> [HCR]	
command-int.	---------- >	
	[HCR] --->	SRIPL command
	[CCR] <---	ACK response
	<----------	command-int.
		execute Act.1
	[CCR] <---	EOC response
	[CDR] <---	end status
	<----------	command-int.
response	<--- [CCR]	
status	<--- [CDR]	
IPL commamd	---> [HCR]	
command-int.	---------- >	
	[HCR] --->	IPL command
	[CCR] <---	ACK response
	<----------	command-int.
response	<--- [CCR]	
address(high)	---> [HDR]	
data-int.	---------- >	
	[HDR] --->	address(high)
	[CDR] <---	address(high)
	<----------	data-int.
address(low)	---> [HDR]	
data-int.	---------- >	
	[HDR] --->	address(low)
	[CDR] <---	address(low)
	<----------	data-int.
/ data	---> [HDR]	
| data-int.	---------- >	
|	[HDR] --->	data
|	[CDR] <---	data
|	<----------	data-int.
|	:	
repeat |	128 bytes	
|	:	
| data	---> [HDR]	
| data-int.	---------- >	
|	[HDR] --->	data
|	[CDR] <---	data
|	<----------	data-int.
|		execute Act.2
|	[CCR] <---	EOC response
|	[CDR] <---	end status
|	<----------	command-int.
| response	<--- [CCR]	
\ status	<--- [CDR]	
RUN command	---> [HCR]	
command-int.	---------- >	
	[HCR] --->	RUN command
	[CCR] <---	ACK response
	<----------	command-int.
response	<--- [CCR]	
address(high)	---> [HDR]	
data-int.	---------- >	
	[HDR] --->	address(high)
	[CDR] <---	address(high)
	<----------	data-int.
address(low)	---> [HDR]	
data-int.	---------- >	
	[HDR] --->	address(low)
	[CDR] <---	address(low)
	<----------	data-int.
		execute Act.3
	[CCR] <---	EOC response
	[CDR] <---	end status
	<----------	command-int.
response	<--- [CCR]	
status	<--- [CDR]	
RRIPL command	----[HCR]	
command-int.	---------- >	
	[HCR] --->	RRIPL command
	[CCR] <---	ACK response
	<----------	command-int.
response	<--- [CCR]	

EOC : End Of Command
Act.1 : SIMコマンドフレームの送信と応答確認
Act.2 : UIコマンドフレーム(IPL)の送信と応答確認
Act.3 : UIコマンドフレーム(RUN)の送信と応答確認

図9. リモートIPLにおける一次局CCPとホスト間の手順

タフェースレジスタの相互排斥、割込を使用したハンドシェイク手法の実現等を含む複雑なインタフェースソフトウェアを作成しなければならない。

(3) ネットワークを介したプロセス間通信を行うとき、あるホストのプロセスが他のホストのプロセスの生成を指示し、通信を行い、通信終了後消滅させる必要がある。これを可能にするためには、CCPのプログラムからホストのプロセスの生成・消滅を指示できることが要求されるが、そのためには複雑なインタフェースソフトウェアを作成しなければならない。

インタフェースファンクションは、これらの問題点を解決し、ネットワークソフトウェアの実現に要する非本質的な労力を軽減するために、HOLENETに用意されたサービス機能である。インタフェースファンクションの一覧を表3に示す。

インタフェースファンクションとしては、高レベルファンクションと低レベルファンクションの2種類が用意されている。前者は、CCPをマスター、ホストをスレーブとするものであり、ホストのコンソールアクセス、キューアクセス、プロセスの生成・消滅、フラグ待ち等の機能(詳細については文献2)を参照のこと)をCCP上のプログラムに提供するものである。(図10) 後者は、ホストをマスター、CCPをスレーブとするものであり、ホストからCCPのメモリ及びレジスタの内容を監視する目的で使用される。(図11)

高レベルファンクションは、CCP上の通信制御プログラム(通常はユーザプログラム)から呼び出される。この時必要なパラメータは、定められたパラメータ形式に従って構成され、レジスタを経由してインタフェースルーチンに渡される。

低レベルファンクションは、コンソールから入力され、インタフェースプロセスによって、CCPのインタフェースルーチンに送られる。インタフェースルーチンは、割込によって駆動され、割込発生直前のレジスタの内容または指定された番地のメモリー内容をインタフェースプロセスに返送する。この結果はコンソールに表示される。

高レベルファンクションの典型的な使用例とその手順を以下に挙げる。

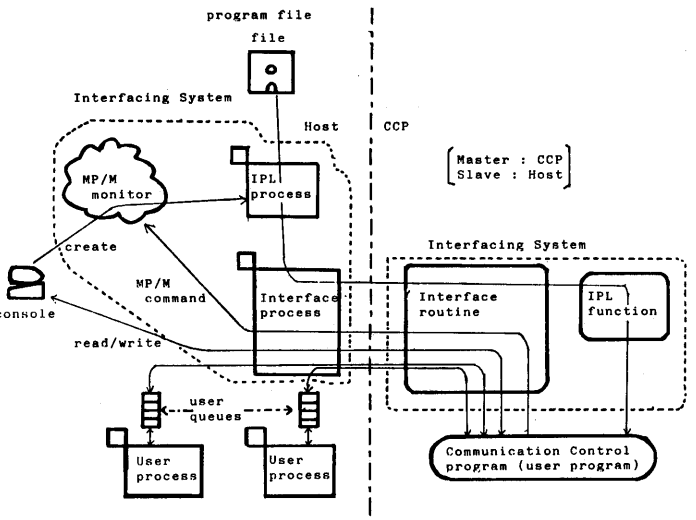


図10 高レベルファンクションシステム

表3 インタフェースファンクション

Higher Level function

1. System Reset	9. Assign Console
2. Open Queue	10. Detach Console
3. Conditional Write Queue	11. Console Input
4. Conditional Read Queue	12. Console Output
5. Send CLI Command	13. Raw Console Input
6. Read Queue	14. Raw Console Output
7. Write Queue	15. Print String
8. Flag Wait	16. Read Console Buffer

Lower Level function

1. Display Registers	2. Display Memory

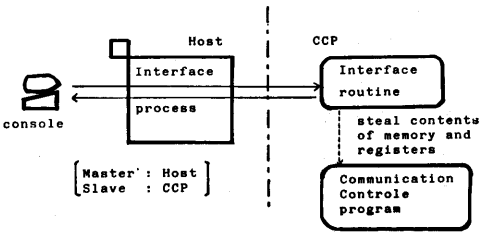


図11 低レベルファンクションシステム

(1) コンソールのアクセス

(i) コンソールの使用権を獲得する。(Assign Console)

(ii) コンソールをアクセスする。

(Console Input/Output, Print String, Read Console Buffer)

(iii) コンソール権を手放す。(Detach Console)

＊ただし、Raw Console Input/Output はコンソール権を持たなくても使用可能である。

(2) ユーザプロセスとキューを介して通信する。

(i) キューをオープンする。(Open Queue)

(ii) キューをアクセスする。([Conditional] Read/Write Queue)

(3) CCP へ新しい通信制御プログラムをロードする。

(i) IPL プロセスの生成を指示する。(Send CLI Command)

＊ Send CLI Command でもって、ホストのプロセスの生成・消滅を指示する。

より詳細な、ホスト側のインタフェースファンクション処理システムを図12に示す。

ファンクションプロセスは、CCP からのコマンド割込によって高レベルファンクションコードを受取り、Hキューに書込んだ後 S_0 に対して V 操作を行う。(図13)

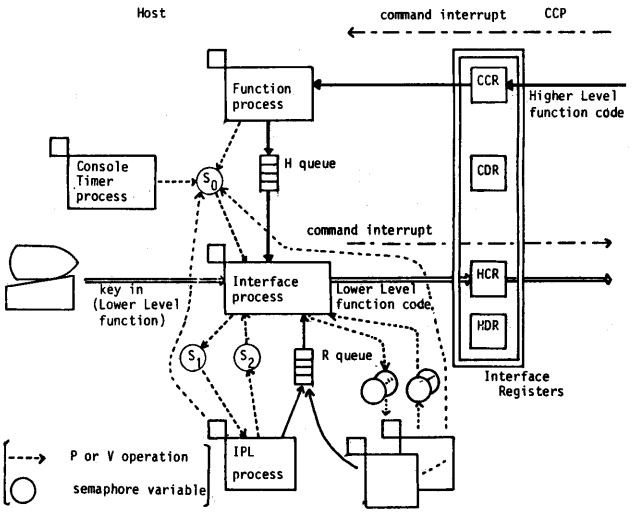


図12 インタフェースファンクション処理システム

インタフェースプロセスには、コンソールモードと非コンソールモードとあり、CCP からのコンソール権獲得要求によってコンソール権を獲得するとコンソールモードになる。コンソールモードになると、コンソールタイマプロセスは、定期的に S_0 に対して V 操作を行って、インタフェースプロセスに低レベルファンクションの入力の検査を、要求する。

IPL プロセスは、IPL 時にインタフェースレジスタを直接使用するために、インタフェースレジスタを管理するインタフェースプロセスとの間に競合が起る。従って、次の順序でインタフェースレジスタの相互排斥を行って、CCP への IPL を行う。

(1) IPL プロセスは、コンソールまたは CCP より生成されると、Rキューに IPL(開始通知)コマンドを書込み S_0 に対して V 操作を行う。(2) インタフェースプロセスは、そのコマンドを受諾すると、Hキューを空にした後、IPL 許可セマフォ S_1 に V 操作を行いブロック状態になる。(3) IPL プロセスは、インタフェースレジスタを介して IPL コマンドを CCP へ送り、IPL を行う。(4) IPL 終了後、IPL プロセスは IPL 終了セマ

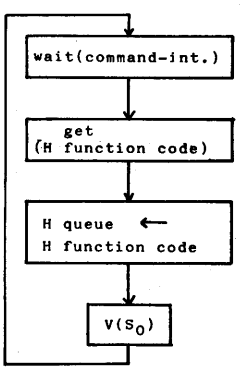


図13 ファンクションプロセスの処理手順

フォS2に対してV操作を行いインタフェースプロセスのブロック状態を解く。

IPLプロセスとインタフェースプロセスの処理手順を、各々図14,図15に示す。また、インタフェースプロセスの処理手順においてRコマンドが、IPL(開始通知)コマンドの場合の処理手順を図16に示す。

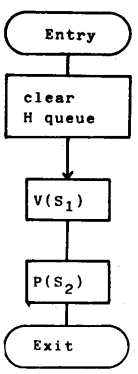


図16 図14においてRコマンドがIPLコマンドの場合の処理手順

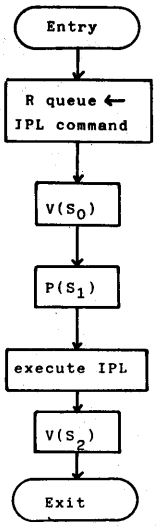


図15 IPLプロセスの処理手順

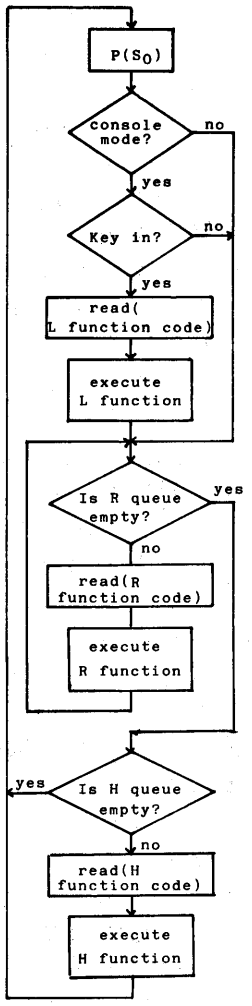


図14 インタフェースプロセスの処理手順

6. あとがき

HOLENETは、自立した通信サービス機能を持つサブネットとホスト計算機群から構成される、ネットワークの教育・研究用モデルシステムを目標に開発された。通信機能をCCPに、データ処理機能をホストに各々分担させ、リモートIPL機能を付加することにより、この目標はほぼ達成されたと思われる。また、CCPの通信制御用LSIにはSDLC/HDLCプロトコルコントローラを用いているので、HOLENET上でHDLC手順に基づくデータリンクレベル通信プロトコルの実験、実習が可能である。

ホストのモニタとしてMP/Mマルチプロセスモニタを採用し、MP/Mの下で動作するインタフェースファンクションを用意したことにより、高度なネットワークソフトウェア(プロセス間通信、ファイル転送等)を短期間に開発し、実験、実習を行うことが容易になった。なお、インタフェースプロセス、IPLプロセスの大きさは各々4KB,3KBであり、CCPのインタフェースルーチンとIPLファンクションは、共に約1KBである。現在、簡易通信プロトコルを用いたいくつかの応用プログラム(ファイル転送、ジョブ転送、ネットワークゲーム等)が、HOLNETよび動作している。

謝辞　本研究に御協力いただいた磯森行教授、安在弘幸助教授、及び石田博明、一ノ宮弘の両氏に感謝します。

参考文献

1) 重松、小出："マルチプロセスモニタに基づくインテリジェント端末システム"，情処学会マイクロコンピュータ研究会(1982)
2) Digital Research："MP/M User's Guide"
3) "Data Communication－High Level Data Link Control Procedure", ISO/TC 97/SC 6
4) 日本電信電話公社："DCNAデータリンクレベルプロトコル", DCNA PS020-1980
5) "Recommendation X.25", CCITT
6) Tamenbaum, A. S. "Network Protocols," Computing Surveys Vol. 13, No.4(Dec.1981)