資　　料

PDP-11/20 デュアルシステムにおけるデバイスシェアリングシステムの製作†

本田　直人††　葛山　善基††　藤井　護††　都倉　信樹††

Abstract

A PDP-11 dual system with an interprocessor communication channel (DA11-D) has been installed at Osaka University. Development work started on a device-sharing system, in which a user can use all devices connected with another CPU as if they were of its own.

This paper describes the design and implementation of a device-sharing system which is consistent with a manufacturer-supplied disk operating system, and discusses some of the problems encountered.

This system began the operation in December 1972 and is now used successfully.

1. はじめに

近年，複数の CPU（中央処理装置）を持ついわゆるマルチ・コンピュータ・システムが増加している．それにともないミニコンのネットワークに関する研究も各地で盛んに行われるようになった[1]．大阪大学基礎工学部情報工学科には，2台のミニコン PDP-11/20 とこれらを結ぶインタフェース装置（DA11）から成る図1に示すようなデュアルシステムがある．これらA・B両システムは，導入時にはそれぞれ DOS(Disk Operating System)のもとで独立に動作できるだけで，Aシステムからは高速紙テープリーダなどを，またBシステムからはラインプリンタなどを使用できなかった．もし，このようなデバイス（入出力装置）を A・B両システムから使えば，各システムにデバイスを増設したのと同じ効果を得ることができる．そこで既存の DOS を改造して，コンソールとテレタイプを除くすべてのデバイスを両システムから使えるようなデバイスシェアリングシステムを開発した．以下，2章では PDP-11 の概要を紹介し，3章でこのシステムの作製の基本方針を，4章ではここで用いたデータ転送

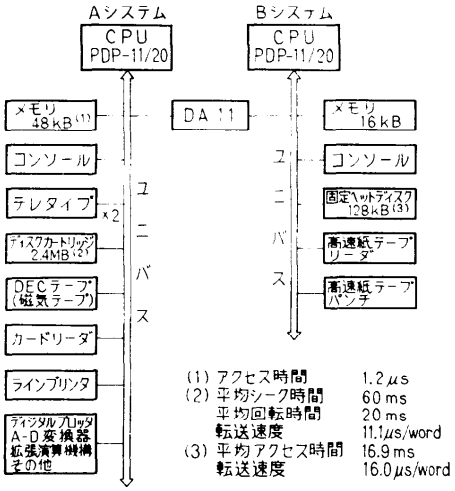

図 1　デュアルシステムの構成
Fig. 1　The dual system configuration

の方法を，5章ではロック・アンロックの問題を，また6章ではシステム全体のデバイスを管理するデバイス管理ルーチンについて述べる．

2. 既存システムの概要

2.1 ハードウェア[2]

○ユニバスによる共通母線方式を用いている．

○各デバイスの制御装置内のレジスタは，メモリと

† A device-sharing system for a PDP-11/20 dual system, by Naoto HONDA, Yoshiki KATSUYAMA, Mamoru FUJII and Nobuki TOKURA (Faculty of Engineering Science, Osaka University)

†† 大阪大学基礎工学部情報工学科

同様に番地付けられており，すべての命令は，メモリに対してと同様にこれらのレジスタに対して適用できる．

○デバイスは，それぞれ固有のメモリ・ロケーション（割込みベクトル）を指すハードウェアのポインタを持っている．このメモリ・ロケーションにデバイスに対応する割込み処理ルーチンの開始番地を設定しておくと，デバイスからの割込みで，機械的に対応する割込み処理ルーチンを選択する．したがって，どのデバイスからの割込みかをソフト的に解析する必要はない．

○一般的な記憶保護機構はない．

○A・B 両システム間の通信を可能にするインタフェース装置 DA-11 を持つ．DA-11 には次の2種類の通信方式がある[3]．

（1） D. M. A. (Direct Memory Access) 方式

相手の CPU を介さずに，相手の番地付けられたメモリおよびデバイスの制御レジスタに直接アクセスできる方法である．今A側からユニバス上の 18 ビットのバスアドレスラインの上8ビットに，特定のビットパターン Z_A を，下10 ビットに任意のビットパターンXをのせると，DA11 は Z_A を検知して，DA11 中の窓レジスタ W_B の内容 Y_B を用いアドレスを Y_B・X† と変換してB側のアドレスラインにのせ，A側はB側のアドレス Y_B・X へあたかも自分の側のアドレススペースであるがごとくアクセスできて，情報（1バイトまたは2バイト単位）を転送できる．A側もB側も全く対称に作られており，B側からも同じことができる．以下いちいち断らずA側から見て述べる．DA11 はA側からとB側からのデータ転送が同時にできるという意味で全二重の通信路と考えてよい．上のように，Aで Z_A・X というアドレスを発するとB側の Y_B・X というアドレスへ直接アクセスできるから，AからB側の Y_B・0…0 から Y_B・1…1 までの 1024 バイトの領域 C_B をA側の領域のごとくアクセスできる．もちろんB側も C_B に直接アクセスできるから，この部分はAとBにとってB側にある共通領域となる．C_B の位置は窓レジスタ W_B の内容できまる．W_B は通常B側からアクセスされ，A側から値を変えることはできないので，B側で適当な値をセットしてもらう必要がある．一方，B側は DA11 のステータスレジスタの特定のビットを設定することで，A が C_B へアクセスするのを禁止できる．このように，相手側からのアクセスを制御できるという意味でシステム間の一種の記憶保護機構をもっている．

PDP-11 にはチャネル装置はないが，ディスクや DEC テープはブロック転送を行う機能を各制御装置が持っており，転送開始番地として Z_A・X を指定すれば，例えば，Aのディスクから DA11 を経由してBの Y_B・X 番地以降へ CPU A, B をわずらわせずに転送できる．

（2） P. T. (Program Transfer) 方式

これも全二重の方式と考えてよい．A から DA 11 中のデータレジスタ DR_A（2バイト）にデータを送ると，DA 11 はB側に割込みをかける．B側は，DR_A の内容を読むことにより，Aからのデータを受けとる．

2.2 DOS モニタ[4]

○DOS モニタは，ユーザがコンソールを通じてファイル管理プログラム，コンパイラ，実行制御用ルーチンなどと通信を行いながら，インタラクティブにプログラムの開発，実行を行うのに適したモニタであり，マルチプログラムの機能はない．

○ユーザプログラムからの I/O マクロはモニタ中の I/O ルーチンで処理されるが，これは大きく次の2つに分れる．

（i） ロジカル I/O (LIO)：データのブロッキング，フォーマティングを行う．ファイル構造を持つデバイス（後述）では，この他にデバイスのブロック管理も行う．

（ii） フィジカル I/O (PIO)：各デバイスに対して用意されており，デバイスの制御装置を直接操作し，データの転送，割込みの処理を行う．

3. システム作製の基本方針

前節でも述べたように，このモニタはインタラクティブにシステムを使用するのに適したものであるため，通常のバッチモニタのようにジョブあるいはジョブステップの開始時に必要なリソースをすべて確保する方式では，デバイスを不当に長時間にわたり占有することになり，実用上デバイスシェアの効果が著しく低下する．このために，デバイスの確保は必要に応じて動的に行うことが望ましい．しかし，このときファイルの混入（例えば，ラインプリンタの出力用紙に両ユーザのファイルが交互に出力されるような事態）は避けなければならない．これらを考慮して，ここでは次のような基本方針をたてた．

† Y_B と X の連接 (concatenation) を意味する．

(i) 既存のモニタの変更をできるだけ少くする.

(ii) 既存のシステムプログラムやユーザプログラムが新しいモニタのもとでそのまま使えること.

(iii) ユーザから見て,自分の使用したいデバイスがいずれの側にあるにせよ,その使用手続きが同じであること,したがってユーザがプログラムを書く際,いずれのシステムで走らせるかを考慮する必要はない.

(iv) デバイスの占有時間をできるだけ短くすること.

4. データ転送

以上のような現有システムの特徴および基本方針から,データ転送は次のような方法で行うことにした.

使用するデバイスがどちら側のものであっても,使用する側の PIO が直接(デバイスが相手側であれば,DA 11 を通して D. M. A. 方式で)デバイスを制御する.相手側のデバイスの場合,相手側の CPU がこの I/O 操作に対して補助することは,使用する側が直接デバイスを制御できるように DA 11 中の窓レジスタを設定することと,そのデバイスからの割込みを相手側(そのデバイスを使用している側)に伝達することだけである.したがって,相手側デバイスに対するデータ転送は図2のようになる.

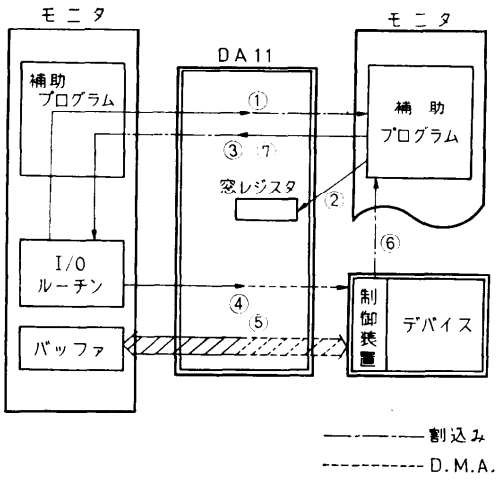

図 2 相手システムのデバイスとのデータ転送
Fig. 2 Data transfer sequence

① 窓レジスタの設定依頼 (P. T. 方式)
② 窓レジスタの設定
③ 窓レジスタ設定完了の通知 (P. T. 方式)
④ デバイス制御レジスタの設定 (D. M. A. 方式)
⑤ データ転送 (D. M. A. 方式)
⑥ デバイスからの割込み
⑦ 割込み伝達 (P. T. 方式)

図2の補助プログラム(後述のデバイス管理ルーチン)のうち,相手側のものはこの I/O 操作に対する前述の補助を,また自分の側のものは I/O ルーチンと相手側の補助プログラムとの間の通信の補助を行う.なお,自分の側のデバイスに対しては,従来どおり上の④⑤⑥の手続きで行われる.ただし,この他にデバイス確保の手続き(後述)も必要である.

5. デバイスのロック・アンロック

モニタ自身からのものを含むすべての I/O 要求は,まず LIO で処理され,LIO から PIO にデバイスの操作要求が出される.前章で述べたようなデータ転送の方法では,デバイスを直接制御する PIO が,各デバイスについて,両システムに1つずつあることになるから†,同時に同じデバイスにアクセスしようとすることも起り得る.またデバイスによっては,交互に同じデバイスにアクセスするとファイルの混入を起す可能性がある.このような事態を避けるためには,例えばAシステムがあるデバイスを使う場合には,安全にデータ転送を行うのに必要な期間Bシステムがこのデバイスを使うことを禁止すればよい.このように,あるデバイスを一定期間他のシステムに使わせないようにすることをそのデバイスをロックするという.ロックしたデバイスに対して,この禁止を解くことをアンロックするという.デバイスシェアリングシステムでは,全システムのデバイスを管理するデバイス管理ルーチン (DEMON——DEvice MONitor——と略す) が用意されており,この DEMON に対して,I/O ルーチンが必要な時点で,必要なデバイスのロック・アンロックの要求を出すことにより,これが実現される.DEMON におけるデバイス管理の手法は6章にゆずるとして,ここでは,I/O ルーチンがどのようにしてロックをかけることにより,ファイルの混入などの問題を解決しているかを述べる.

5.1 ロック方式

各デバイスの I/O ルーチンは,その性格により,次の3つのロック方式のうちのいくつかを組み合わせて用いる.

○転送ロック: PIO がフィジカルな1レコードの

† PIO を1つにすることも可能だが,I/O コマンド体系の大幅な改造が必要となる.

データを転送するために，そのデバイスの制御装置(および必要なら DA 11) を占有するためのロックである．PIO は DEMON に対して，そのデバイスの転送ロック要求を出し，転送許可の応答が来ればロックされたとみなす．このとき，相手システムがすでにこのデバイスをロックしていたならば，この要求は，DEMON 中で待ち行列に入れられ，アンロックされた時点で転送許可の応答が出る．アンロックは，転送ロック解除の要求を DEMON に発することにより行われる．

○占有ロック：LIO がファイルの混入防止のためにかけるロックである．DEMON にそのデバイスの占有ロック要求を出し，占有許可がでればロックがかかったことを意味する．占有したいデバイスがすでに他システムにより占有ロックされている場合には，待ち行列には入らず，占有不能が I/O ルーチンを経てオペレータに伝えられる．

○ビットマップロック：**5.3** で述べる．

ここで，占有ロックに対して待ち行列が用意されていないのは，例えば次のような事態（デッドロック）を容易に検出できるようにしたためである．A システムがあるデバイス a を先ず占有し，次に B システムが別のデバイス b を占有する．さらに A システムがデバイス b を，B システムがデバイス a を占有しようとする．もし，占有ロックに対しても待ち行列を用意すると，これらの要求は待ち行列に入るが，両システムともデバイス a，b がそろわないと処理ができないとすれば，デバイス a，b は占有ロックが解除されることはなく永久に処理できない．一方，デバイスを占有できないときはその旨をオペレータに知らせる方式では，オペレータはデバイスがあいたことを確認してから続行指令を発すればよい．したがってデッドロックが生じた場合には，両システムのコンソールに占有不能のメッセージが出力されることになり，デッドロックが検出できる．転送ロックでは，ロックされたデバイスはいつかその処理が終り必ずアンロックされるので，待ち行列を設けても不都合は生じない．

使用雑度の高い DA 11 は，転送ロックのみが許され，占有ロックは許されていない．

各デバイスの I/O ルーチンが，どのロック方式をどのように組み合わせて用いるかは，占有時間の短縮およびファイルの混入の問題を考慮して決定する必要がある．ここでは，デバイスをファイル構造を持つものとそうでないものとに大別して考える．

5.2 ファイル構造を持たないデバイス

ラインプリンタやカードリーダのようなファイル構造を持たないデバイスでは，ファイルの混入を避けるため，ユーザからの OPEN マクロ受付け時にデバイスの占有ロックを行う．引き続く I/O 要求は転送ロックを必要とする．これは，その I/O 処理に DA 11 を必要とする場合，占有ロックできない DA 11 を転送ロックするためである．ユーザからの CLOSE マクロ受付け時に，この占有ロックは解除される．図 3

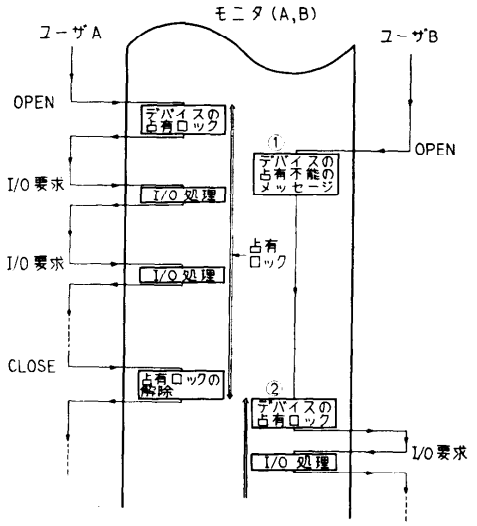

図 3 ラインプリンタやカードリーダのロック・アンロック
Eig. 2 Lock and unlock sequence for non-file structured device

には，両システムからこの種の同じデバイスに同時に I/O 処理を始めようとした場合の流れを示す．(以下の図では，DEMON が全システムのデバイスを管理していることを考慮して，図の簡単のため両モニタを 1 つにまとめて書く．) ①では，OPEN マクロ受付け時に占有ロックできなかったため占有不能をコンソールに出力し，オペレータの指示を待っている．②でオペレータが，そのデバイスがあいたことを確認して続行指令を発すと，処理を続行する．

5.3 ファイル構造を持つデバイス[5]

ディスクや DEC テープのようなファイル構造を持つデバイスにはディレクトリーと呼ばれる登録表があり，これにより複数個のファイルを管理している．ファイルは複数個のブロックより構成されているため，ブロックを資源とみなせる．各ブロックが空いている

かどうかの情報は，そのデバイス中の PBM（パーマネントビットマップ）に集中的に保持されている．PBM は一般に数ページにわかれている．

このようなデバイスからのファイルの読込みは，転送ロックのみで充分である．ただし，入出力とも OPEN 時にそのファイルをロックし CLOSE 時にアンロックする機能は，すでに DOS に含まれているため，書き込み中のファイルを誤って読むことはない．

一方，データの書き込みに関しては少し事情が異なる．データをこれらのデバイスに書き込む場合には，まず PBM の適当な1ページをメモリに読み込み（これを CBM-コアビットマップという），これを用いて空きブロックを探し，この空きブロックにデータを書き込み同時に CBM を更新する．引き続く出力要求に対してはこの CBM が用いられ，この CBM に空きブロックが無くなれば，これをデバイス上に返し別の PBM を読み込む．CBM は，そのファイルに対する CLOSE マクロを受付けたとき PBM に返される．したがって，両システムから同じデバイスに出力要求を出すと，同じページの PBM を読み込む可能性があり，この場合同じブロックに書き込む事態が起る．このような事態を防ぎ，またロック時間の長い占有ロックをできるだけ避けるため，ここでは転送ロック，占有ロックの他にビットマップロックを用いる．これは PBM をメモリ上に読み込んだ段階でその PBM の一部に使用中であることを示すビットをセットし，他のシステムが PBM のこのページを使うことを禁止することである．したがって，このページの PBM が代表するブロック群を占有ロックしたのと同じ効果を得る．これにより，デバイスの占有ロックを行わなくても，同一ブロックに両システムから書き込むことはなくなる．ただし，PBM を読み込みそのページをロックするまでの間はデバイスの占有ロックを行う．PBM のアンロックは，CBM を PBM に返す際に行われる．このようなロック方式においては，ロックされた PBM を読み込んだ場合には，ロックされていない他の PBM のページを読み込んで使用すればよい．このときの流れを図4に示す．①ではまず PBM の1ページ目を読むが，ロックされているため，2ページ目を読み，これにロックをかけて以後で使用することを示している．

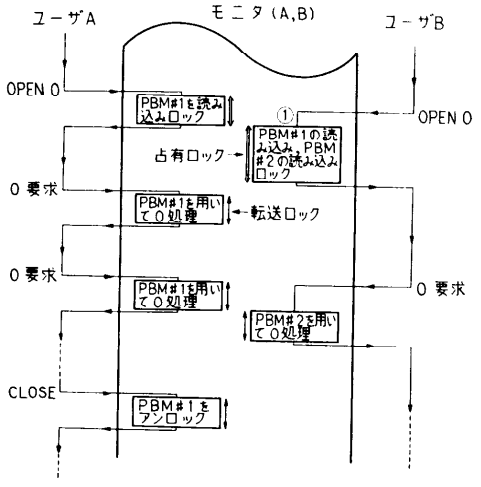

図 4 ディスクのロック・アンロック(出力)
Fig. 4 Lock and unlock sequence for file structured devices

以上のようなロック方式により，このようなデバイスに対するロック時間が非常に短縮される．これは，使用頻度上からも非常に大きな利点である．

6. デバイス管理ルーチン（DEMON）

DEMON は，相手側のデバイス操作に対する4章に述べたような補助および5章で述べたロック・アンロックの要求に対する処理を行い，全デバイスを管理するルーチンである．ただし，システム固有のものとして扱うのが望ましいコンソールや，フィジカル I/O を持たないテレタイプは管理の対象になっておらず，また DA 11 は一種のリソースとして DEMON が管理する．

DEMON は両システムのモニタの常駐部に組み込まれ，それぞれは自分の側のデバイスを管理し，また互いに交信しながら全デバイスを管理する．各システムの DEMON は全く対称に作られており，主従の関係はない．そこで以下特に断らない限りAシステムを中心とし，Bシステムを相手側システムとして説明する．

6.1 DEMON の機能

（1） デバイス管理

5章でも述べたように，モニタ中の LIO や PIO は，デバイスを転送ロック，占有ロックするために，転送ロック要求，転送ロック解除，占有ロック要求，占有ロック解除を必要に応じて DEMON に発する．これらの要求は，そのデバイスがいずれの側にあるにせよ，自分の側の DEMON に対して出され，DEMON は，そのデバイスが自分の側のものであればここで調べ，相手側のデバイスであれば相手側の DEMON

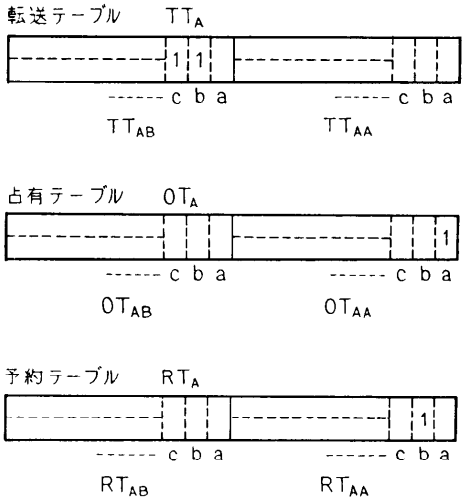

図 5　デバイス管理テーブル
Fig. 5　Device control tables

にロックの可否をたずね，要求を発した I/O ルーチンに返答する．

Aシステムの DEMON は，A側のデバイスを図5に示す3種類のテーブルで管理している．各テーブルは，A側のための領域とB側のための領域にわかれており，各領域のビット位置がA側の各デバイスと1対1に対応している．

○転送テーブル (TT_A: TT_{AA} および TT_{AB})

A側の各デバイスが，現在どちらのシステムにより転送ロックされているかを示す．図5では，Bシステムが A側のデバイス b, c を転送ロックしている．

○占有テーブル (OT_A: OT_{AA} および OT_{AB})

A側の各デバイスが，現在どちらのシステムにより占有ロックされているかを示す．図5では，Aシステムが A側のデバイス a を占有ロックしている．

○予約テーブル (RT_A: RT_{AA} および RT_{AB})

すでに一方のシステムによって転送あるいは占有ロックされている A側のデバイスに対して，他方のシス

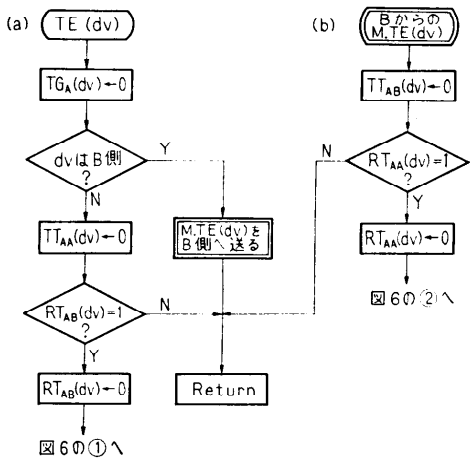

図 7　A側 DEMON の転送ロック解除の処理
Fig. 7　Flow chart of transfer end in DEMON

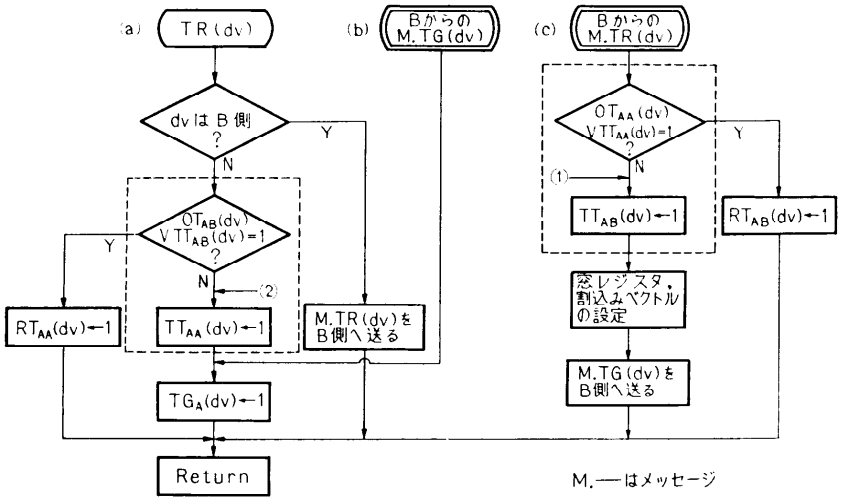

図 6　A側 DEMON の転送ロック要求の処理
Eig. 6　Flow chart of transfer request in DEMON

テムから転送ロック要求が出されたとき，このテーブルがセットされる．図5では，Aシステムからデバイスbの転送ロック要求が出されたが，Bシステムがすでに転送ロックを行なっているため，RT_{AA} の対応するビットがセットされている．

これらのテーブルを用いて，A側の DEMON は，A側のデバイスに対する転送ロック要求，転送ロック解除を図6，図7にそって処理する．ここで，⬭ はA側の I/O ルーチンからの要求に対する処理の入り口であり，⬭ はB側の DEMON からの割込みに対する処理の入り口である．また，破線で囲まれた部分では，優先度を上げることによりすべての割込みを禁止している．

○例 1 AシステムからAシステムのデバイス(1)の転送ロック要求

PIO からのデバイス名（1）をパラメタとする転送ロック要求 TR(1) は，図6（a）の部分に入る．各デバイスがいずれの側にあるのかを示すテーブルにより，このデバイスがA側にあることを知る．そこで，B側がこのデバイスをロックしていないかどうかを調べ，ロックしていなければ $TT_{AA}(1)$ および PIO との通信領域である $TG_A(1)$ をセットし，転送ロック許可を知らせる．PIO は，コントロールが戻ると常に $TG_A(1)$ を見ており，$TG_A(1)$ がセットされればデバイス（1）を使い始める．

○例 2 BシステムからAシステムのデバイス(1)の転送ロック要求

この要求は，B側 DEMON の図6（a）の部分に入る．デバイスはA側のものであるから，A側の DEMON に処理を依頼する (M. TR(1))．これは割込みによってA側の図6（c）に入り，ここでこのデバイスをA側がロックしているかどうかを調べる．ここでは例1によってデバイス（1）がロックされているため，Bのための予約テーブル $RT_{AB}(1)$ をセットして割込みから戻る．B側では処理を依頼した後すぐに PIO に戻るが，B側の通信領域 $TG_B(1)$ がセットされていないため，デバイス（1）を使用できない．

○例 3 AシステムからAシステムのデバイス(1)の転送ロック解除

この要求は図7（a）の部分に入る．通信領域 $TG_A(1)$ をクリアし，A側のデバイスであるから $TT_{AA}(1)$ をクリアする．次に，このデバイスに対するB側の待ちを調べる．例2で $RT_{AB}(1)$ がセットされているため，図6の①へとび，転送テーブル $TT_{AB}(1)$ をセットし，DA 11 の窓レジシタに，BシステムからDA 11 を通してデバイス（1）の制御レジスタに D. M. A. 方式でアクセスできるような値を設定する．同時に，そのデバイスの割込みベクトルに DEMON の割込み伝達ルーチンの入り口の番地を設定する．その後，B側の DEMON にデバイス（1）の転送ロック許可を応答する (M. TG(1))．この応答はB側の DEMON の図6（b）の部分に伝わり，通信領域 $TG_B(1)$ をセットする．これで，例2の要求が許可されたわけで，この段階でBシステムはデバイス（2）を使用できる．

○例 4 BシステムからAシステムのデバイス(1)の転送ロック解除

この要求は，B側 DEMON の図7（a）の部分に入る．まず通信領域 $TG_B(1)$ をクリアし，A側のデバイスであるため，A側へ M. TE(1) を送る．これはA側の図7（b）の部分に入り，$TT_{AB}(1)$ をクリアし，このデバイスに対するA側の待ちを調べる．待ちがなければただちに戻り，待ちがあれば $RT_{AA}(1)$ をクリアし図6②の部分へとび，以下，例1と同様の処理を続ける．

占有ロック要求や占有ロック解除要求も，上の例とほぼ同様な過程で処理される．ただし，占有ロック要求に対する待ち行列が用意されていないことは，前にも述べたとおりである．

（2） その他の DEMON の機能

例3において，デバイス（1）からの割込みはA側にかかるので，これをB側に伝える必要がある．デバイス（1）からの割込みは，割込みベクトルを介して DEMON 中の割込み伝達ルーチンに入り，割込み伝達ルーチンは，デバイス名をパラメタとしてB側へ割込み伝達メッセージを送る．B側の DEMON 中の割込み伝達受理ルーチンが，割込みによってこれを受け，そのデバイスの割込み処理ルーチンにコントロールを移し，割込み処理を任せる．

DEMON には，この他に，リセット命令（そのシステム側のすべてのデバイスの I/O 処理を中断させる命令）の処理の機能もあるが，ここでは述べない．

7. おわりに

DEMON の作成および PIO, LIO の変更は，すべてアセンブリ言語で行なった．DEMON の大きさは，約 300 ステップ 1000 バイトであり，DEMON 以外はオーバレイ部で処理されるため，モニタの常駐部の増加分は約 1000 バイトである．

このシステムは，将来大阪大学大型計算機センタのNEAC 2200/500 のタイムシェアリングシステムと結合される予定であり，これが完成するとデバイスシェアシステムを含めた計算機ネットワークができる．

最後に，このシステム作成に際して，日頃より御指導いただいた嵩忠雄教授ならびに嵩研究室の皆様に深く感謝いたします．

参 考 文 献

1) ミニコンのソフトウェアとネットワーク報告集：情報処理学会プログラミング・シンポジウム (JULY 1972).
2) PDP-11 handbook : Digital Equipment Corp. (1970).
3) DA 11-D INTERPROCESSOR COMMUNICATIONS CHANNEL, Digital Equipment Corp. (FEB. 1972).
4) PDP-11 Disk operating system monitor programer's handbook, Digital Equipment Corp. (1971).
5) PDP-11 Disk operating system functional specification, Digital Equipment Corp. (1970).

(昭和48年6月11日受　付)
(昭和48年7月14日再受付)