計算機復合体 MICS-II の設計思想と構成†

大森健児†† 小池誠彦††
山崎竹視†† 大宮哲夫††

MICS-II は，従来一体となっていたシステムコントロール部分とプログラム実行部分を，ハードウェア的にもソフトウェア的にも分離することによって，利用者のイメージに即したシステム構成上でマルチプロセッサシステムに係わる諸問題の研究を可能とする，新しい設計思想に基づく計算機復合体である．

ハードウェア構成は，ユーザプログラムを実行するユーザモジュールと，システムコントロールを行うコントロールモジュールを対にしたプロセッサモジュールを構成の単位とし，これらをメモリバス，I/O バス，コントロールバスで接続することによって実現した．

システムコントロールは，各コントロールモジュールに分散され，バイト長，発生分布の異なるメッセージ転送は，信頼性・応答性に優れるエレメントスイッチング方式によって行われる．

また，メインメモリ獲得時に生じるデッドロックの防止と，マルチプロセッサシステムにおける効果を研究するために，仮想メモリ方式を開発した．

MICS-II は，現在6台のプロセッサモジュール，64k ワードのメインメモリ，12台の入出力装置で 1977 年5月より稼動している．

1. 序　論

計算機技術は，新しい素子の発明によって，数々の変遷を経てきた．今日，また，LSI 技術の急激な進歩によってさらに新たな変革を受けようとしている．

その顕著な現われを計算機復合体にみることができる．計算機復合体は，複数のプロセッサによって構成された計算機システムであり，従来の計算機システムに対して，(1)性能価格化に優れる，(2)信頼性に勝る，(3)拡張性に富む，(4)コンポーネントの量産化に適する，などの利点を有している．

計算機復合体の思想自体はさほど新しいものではないが，本格的な幕明けは 1970 年代初期である．すなわち，マイクロプロセッサ，高集積メモリ，ワンボードコンピュータが世の中に出現したころに始まる．我我のプロジェクト Microprocessor Technology For Future Computer System (MICS) もまた，1973 年に始まった．

当時，カリフォルニア大学から Prime[1] が，カーネギメロン大学から C.mmp[2] が，BBN 社から，Pluribus[3] が相次いで発表された．

MICS プロジェクトは，その中にあって，多用途のマルチユーザサービスを目的として研究開発を開始した．その後，数々のハードウェア／ソフトウェアの修正を加え，1977 年5月より，最大性能 3 MIPS で MICS-II が実験機として稼動を開始した．

2. 設計思想

MICS-II の目的は，利用者にマルチプロセッサシステムを構成するのに必要な道具を提供することによって，利用者のイメージに即したシステム構成上でマルチプロセッサシステムに係わる諸問題の研究を可能とすることにある．

マルチプロセッサシステムの実用化の立ち後れは，オペレーティングシステム，利用技術，言語などの未成熟にある．そこで，これら技術を発展させるために各方面からの研究が必要である．しかし，単一プロセッサシステムを用いての研究は，ハードウェア構成，オペレーティングシステムなどからの制約が強すぎるため，①相互に関連しながら走行するプロセッサ間でのデッドロック問題を検証することができない，②排他制御機構の動作を確認できない，③マルチプロセッサシステムの OS の問題点を完全に把握することができない，などの問題を有している．そのため，広範囲な研究を行うためにはマルチプロセッサ構成のとれる実験システムが必要である．しかし，この場合においても利用者にソフトウェア上の道具を提供するだけでは，①実験システムの構成上の自由度を高くすること

† Design Philosophy and Architecture for a Multi Processor System MICS-II by KENJI OHMORI, NOBUHIKO KOIKE, TAKEMI YAMAZAKI, and TETSUO OHMIYA (Central Research Laboratories, Nippon Electric Co., Ltd.).

†† 日本電気(株)中央研究所

ができない，②システムオーバヘッドを伴うため精度の高いデータを収集できない，などの理由により，本質的な解決とならない．そのため，MICS-IIにおいては，プロセッサを含めたハードウェアそのものを利用者に提供することを考えた．

2.1 システム設計

利用者にハードウェアそのものを提供するという考えは，具体的には次のことを意味する．

その一つは，利用者に特権命令を含めたすべてのハードウェア機構の使用を許すことである．MICS-IIは，実験システムであるため，OS の開発，アプリケーションプログラムの開発を行う利用者にとってこれは必要な条件である．

他の一つは，利用者の希望する種類のプロセッサを提供することである．この実現には2つの方式が考えられる．その一つは汎用性に富むマイクロプログラム制御のプロセッサを開発することである．他の一つは多種類のプロセッサを接続できるハードウェアインタフェースを開発することである．前者はあるプロセッサが故障を起こしたとき他のプロセッサにその処理を移しての再実行が可能であるという利点を有しているが，現時点ではこの種のプロセッサを開発するのに相当な時間を要する．

そこで，MICS-IIでは前者を最終目標に後者を当面の目標にした．このような自由度の高いシステムを従来の計算機システムの方式——すなわちシステム管理プログラムとユーザプログラムの実行を同一プロセッサ上で行う方式——で実現することは，次の理由により妥当でない．

イ．利用者の使用したいプロセッサが変化するためシステム管理プログラムをその都度適合させる必要がある．

ロ．システム管理プログラムの下でユーザプログラムを実行させた場合に使用可能なハードウェア機能が制限される．

ハ．ユーザプログラムがシステム管理プログラムの領域に干渉する危険がある．

そこで，MICS-IIではシステム管理部分とプログラム実行部分の分離を図り，それらを別々のプロセッサで行わせることとした．

2.2 システム管理部分

マルチプロセッサ構成によって実現されるプログラム実行部分の実験システムとしての能力は，システム管理部分の構成に大きく影響される．システム管理部分にはプログラム実行部分に不必要なオーバヘッドを与えることなく実験システムとしての充分なサポートを提供することが要求される．これを単一プロセッサシステムで実現することは，次の問題点を有している．

イ．プログラム実行用プロセッサの追加がシステムオーバヘッドの増加を招かないようにするため最終システム構成を想定して性能の高いプロセッサを最初から用意する必要がある．

ロ．システム管理用のプロセッサの故障がシステム全体の故障につながりフォールトトレラント構成をとりにくい．

一方，システム管理部分の機能を，プログラム実行部分のプロセッサにのみ関連するローカル処理と，システム全体に関連するメモリ管理，I/O 管理，ジョブ管理などのグローバル処理にわけて分析したところ，ステップ数で前者が2k ワード，後者が5k ワード，走行時間の比率で前者が90% 以上，後者が数 % であることが判明した．

プログラム実行用のプロセッサごとにそのローカル処理を専門に行うプロセッサを設け，また，グローバル処理を集中して行う方法は，㋑ローカル処理はマイクロプロセッサで充分に実行できる．㋺グローバル処理はプログラム実行用のプロセッサでユーザプログラムの一種として実行できるため経済的であると同時に，①処理量の変化に応じたモジュールの追加削除が容易である．②単一プロセッサシステムに対して集中している部分が少なく，グローバル処理を行うプロセッサの交換が可能であるため信頼性に優れたシステム構成がとれる，などの利点を有している．そこでMICS-II では，ユーザプログラムを実行するユーザプロセッサ (UP) とローカル処理を行うマイクロプロセッサ (μP) の対を並べた形を基本構成とすることにした．

2.3 プログラム実行部分

利用者に自由度の高い使用容易な実験システムを提供するためには，プログラム実行部分に対して次の基本機能をサポートする必要がある．

イ．利用者の希望する種類のプロセッサの提供

ロ．異なるプロセッサ上に実装されているプログラム間での情報交換手段

ハ．異なるプロセッサ上に実装されているプログラム間でのアクセス領域保護

ニ．試作ハードウェア装置の接続の容易さ

ホ．ハードウェアリソースの有効利用

利用者の希望するプロセッサをUPとして使用できるようにするためには，それをUPとして接続できることが必要である．この接続条件を同期インタフェースとするとプロセッサの種類ごとに相当な回路規模をもつインタフェースモジュールが必要である．そこで接続インタフェースは，接続しようとするプロセッサの固有のインタフェースが条件に合わない場合でも簡単な変更で対処でき，しかもUPの動作速度に依存して処理のなされる非同期なメモリアクセス，I/Oアクセスによるものとした．

情報交換手段をその速度面から大別すると，低速度のネットワーク型と高速度のメモリ共有型の2通りがある．実験システムに対しては，疎結合，密結合，またはその混合のマルチプロセッサ構成が，実験の目的に応じて要求されるのでいずれの手段も実現する必要がある．そこで，ネットワーク型に対しては，その多くはシステム管理部分で行わせることにした．メモリ共有型の実現には，すべてのユーザプロセッサからアクセス可能なメインメモリの存在が必須であるが，メインメモリへのアクセスの方法には，マトリックス型，マルチポート型，共通バス方式などがある．MICS-IIでは拡張性，経済性に優れる共通バス方式を採用した．しかし，この方式は，各プロセッサからメインメモリへのアクセス頻度が高くなったとき，プロセッサの性能低下が問題となる．そこで，各UPに専用のメモリをもたせプログラムおよびデータの分配による性能低下低減に取り組めるようにした．

一つのシステムを複数の利用者が使用している場合には，利用者間の保護が重要である．異なる利用者が同時に同一のユーザプロセッサを利用しないことを前提にすると，利用者間の保護をするためには，同一プロセッサ上に実装されていないプログラム間の共通領域を除く相互の領域保護機能が実現されれば十分である．この実現には，プロセッサの発生するアドレスを実際のメモリ上のアドレスに変換する機構と，アクセス条件の検査機構が必要である．さらに，実際のメモリ容量が利用者の要求しているメモリ容量の総数よりも小となったとき，メモリリソースを要求しあって利用者のプログラム間でデッドロックが生じる．このデッドロック問題を解決し，メモリを効率的に使用するため仮想メモリ方式が必要である．

実験システムとしてのMICS-IIにおいては，各種実験のために，試作ハードウェア装置を接続する必要がある．これら試作ハードウェア装置のほとんどは入出力命令によって実行されるためアクセス頻度は低い．従って，共通バス方式でのプロセッサの性能低下は問題とならないので，試作ハードウェア装置以外の入出力装置も接続できる共通バスを設けハードウェアリソースの効率化を図った．

2.4 マイクロプロセッサ間データ交換

MICS-IIの基本構成は，UPとμPの対を並べることであるが，システム管理が分散化しているためグローバル処理要求時，相互診断時などには，μPで相互のデータ交換が必要である．

データ交換の方法として，ユーザプログラム実行のために設けられたメインメモリを利用する方法があるが，これはシステム管理プログラムの領域をユーザプログラムの干渉から保護できないため妥当でない．

すべてのμPからアクセスできる共通メモリを専用に設けることは，①システム全体の信頼性が共通メモリの信頼性に大きく依存する，②グローバル処理も含めたシステム管理部分の完全な分散処理化により将来高度なフォールトトレラントシステムを指向するときの障害となる，などの理由から妥当でない．

そこで，μP間のデータ交換をメッセージ通信により行うことを考えた．システム管理部分におけるデータ交換の特性は，以下のようである．

イ．要求がバースト的に発生する．

ロ．データ長は平均数十ビットであるが長さは一定でない．

ハ．データ交換に要する時間はシステムオーバヘッドにつながる．

従って，これをコンピュータネットワークで実現するには，①データ長が短いため，パケット全体の符号に対し情報ビットの割合が小さすぎ効率的でない，②パケットの組立て，分解のため不必要な時間を要する，などが問題となった．

そこで，MICS-IIでは，メッセージを数ビットのメッセージエレメントに区切り，これを単位として並列に送ることを考えた．この方法では，①メッセージの分解，組立てが簡単である，②誤り検出にパリティ符号を用いれば，一符号当りの情報量が高いなどの利点を有している．しかし，メッセージ通信中，通信路を2つのμPで専有する方式では，データ長の長いメッセージ転送中，他のμPで緊急な通信要求が発生したときそれに応じることができない．そこで，MICS-IIでは，メッセージエレメントごとに通信路を得ることとし，さらに，通信に緊急度を示すためのレベルをも

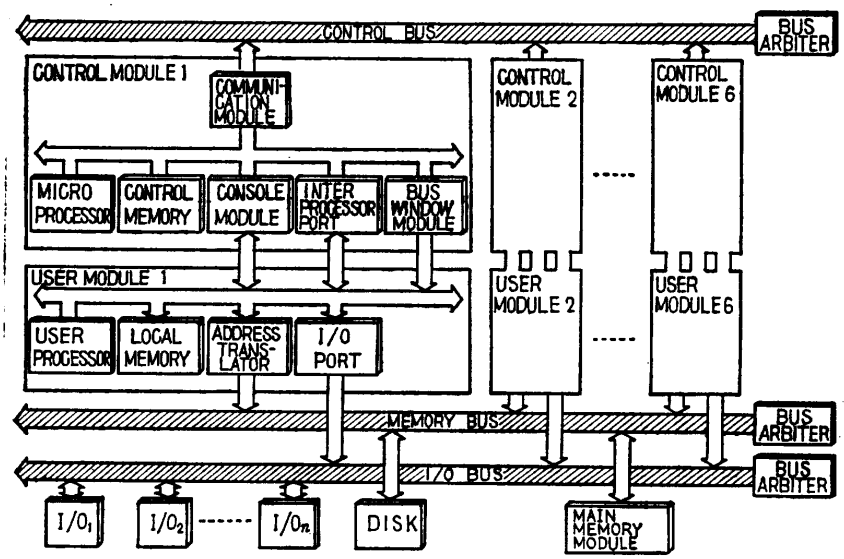

図 1 MICS-II のブロック図
Fig. 1 MICS-II blockdiagram.

たせることとした．さらに，通信可能な相手はリンクを結んでいる μP だけとし，プログラム上，ハードウェア上の間違いから誤って他の μP へアクセスすることのないようにした．

3. 実現手法

ハードウェア構成は**図1**のように μP を核にしたコントロールモジュール (CM) と UP を核にしたユーザモジュール (UM) を対にしたプロセッサモジュール (PM) を実装の単位とし，μP 間の通信路のためのコントロールバス，メインメモリへのアクセスのためのメモリバス，入出力装置，試作ハードウェア装置のための I/O バス上へのそれらの並列接続とした．

コントロールモジュールは，μP，ローカル処理プログラムのためのコントロールメモリ (CMM)，μP 間通信のためのコミュニケーションモジュール (COM)，ネットワーク型の情報交換のためのインタプロセッサポート (IPP) とバスウィンドモジュール (BWM)，UP を制御するためのコンソールモジュール (CON) で構成した．

ユーザモジュールは，UP，ローカルメモリ (LMM)，メインメモリアクセス時にアドレス変換およびアクセス条件検査を行うアドレストランスレータ (AT)，入出力装置アクセス時にデバイスアドレスへの変換を行う I/O ポート (IOP) で構成した．

ローカル処理用のプログラムをプロセッサモジュールマネジャ (PMM)，グローバル処理用のプログラムをシステムリソースマネジャ (SYSM) と名付けた．

次に，メインメモリの動的割当てを行う仮想メモリシステム，μP 間のデータ交換手段であるエレメントスイッチング方式の実現方法について述べる．

3.1 仮想メモリシステム

MICS-II における仮想メモリシステムは，動的なメインメモリの割当ておよびページのスワッピングを行うことによって，マルチプロセッサシステムにおける仮想メモリの効果を研究できるようにするとともに，メインメモリの獲得時に生じるデッドロックを防止する．そこで，メインメモリへのアクセス機構とその割当てを詳述し，このシステムの実現方法を説明する．

3.1.1 アクセス機構

各 UP は独自のアドレスを有す．そのアドレス空間の全部またはその一部は，ローカルメモリ，メインメモリに写像される．メインメモリへの写像関係はそのアクセス条件とともに AT 内のアドレス変換テーブルに**図2**のように記述される．このアドレス変換テーブルによって，UP がメインメモリへのアクセスを行ったとき，プロセッサの発生する論理アドレスがメインメモリ上の物理アドレスに変換される．この機構によって，異なる UP 間に実装されたプログラム間での干渉を防止することができる．また，動的な割当てを許すために，このテーブルは μP によって書換えることができる．そのとき，アクセス条件として，1）メインメモリ未割当て，2）命令コードのみ読出し可，3）すべての読出し可，4）すべてのアクセス可，のいずれかを指定することができる．

UP がメインメモリアクセスを行ったとき，アクセス条件が満足されていれば，UP の論理アドレスがメモリ上の物理アドレスに変換され，メモリへのアクセスが行われる．アクセス条件が満足されないときは，アクセスフォールトフラッグが立ち，メモリアクセスを中断したまま μP の介入を待つ．μP は，割込みまたはセンス命令によってそれを検出し，アクセスフォールトを起こしたページとそのアクセス条件を調べにゆく．その条件が，命令コードのみ読出し可または

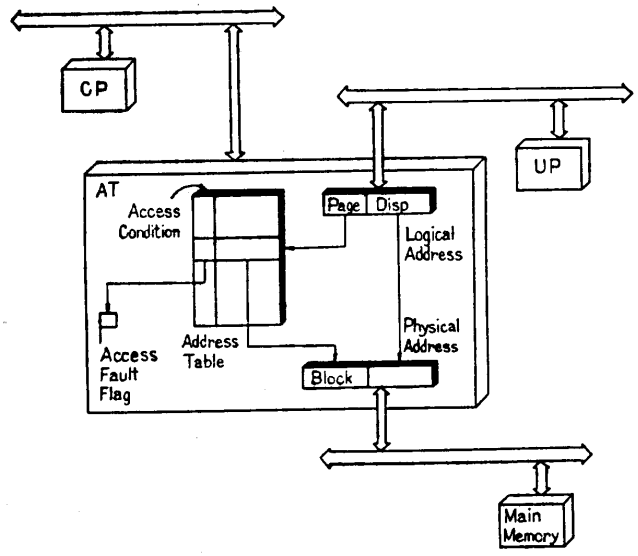

図 2 メモリアクセス機構
Fig. 2 Memory access mechanism.

すべての読出し可のときは，μP は CON を用いて，UP を停止させる．また，その条件がメインメモリ未割当ての場合には，メモリ割当てとページスワップを行った後で，μP は割当てられたメモリブロックアドレスとアクセス条件をアドレス変換テーブル上に書込む．これによって，UP はメモリアクセスを再開する．

3.1.2 メモリ割当て

ユーザプログラムに対するメモリ割当ては，SYSM によって動的に行われるが，メモリ割当ての機会は，UP がメインメモリ未割当てのページへアクセスしたときに起こるページフォールトを PMM が検出したときである．

図3にメモリ割当ての手順を示す．今，SYSM は

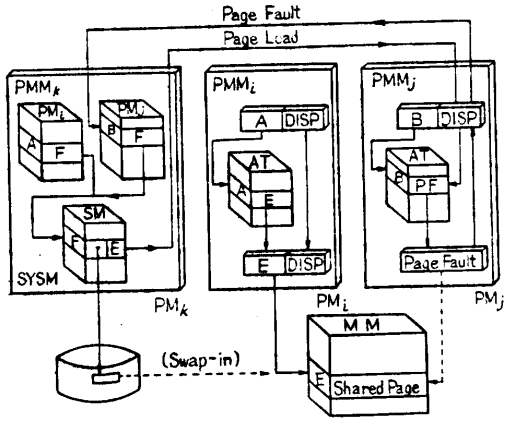

図 3 メモリ割当て機構
Fig. 3 Memory allocation mechanism.

PM$_k$ に実装されているとする．そして，PM$_j$ 内の UP がページBに含まれるアドレスをアクセスしたため，ページフォールトを起こしたと仮定する．PMM$_j$ はこれを検出し，PMM$_k$ に対してページフォールトの通信を発生する．この通信によって，PMM$_k$ は PM$_j$ でページロード要求が発生したことを知る．これは，リソース割当て処理であるために，PMM$_k$ は IPP を介して，SYSM にその旨を伝える．SYSM は，メインメモリ上に割当てられていないブロックがある場合には，その中の一つを割当てブロックとする．

そのようなブロックがない場合には，追出しアルゴリズムによってスワップアウトされるページを得る．そして，SYSM は，スワップアウトされるページとそれを使用している PM の情報を，IPP を介して PMM$_k$ に伝達する．PMM$_k$ は，スワップアウトの対象になっている PM に実装されている PMM に対し，PM 間通信によってスワップアウトされるページの情報を伝達する．通信を受けた PMM は AT 内のアドレス変換テーブルを書換えて，スワップアウトされるページのアクセス条件をメインメモリ未割当てにする．そして，今まで述べた手順とは逆の手順によって，SYSM にアドレス変換テーブル書換え終了の通知をする．そこで，SYSM は実際にスワップアウトを実行し，そして今スワップアウトされたページが実装されていたブロックを割当てブロックとする．

このように割当てブロックが決定されると，SYSM は，ページフォールトを起こしたページを，二次メモリから割当てブロックへスワップインし，ページロード終了の通知に入る．

図3ではページフォールトを起こしたページ B は PM$_j$ でのページ A と共有されており，それはブロック E にすでにロードされている．従って，SYSM はスワップインすることはなく，ページロード終了の通知に入る．

ページロード終了の通知は，ページロード要求のときの手順とは，逆の手順で PMM$_j$ に通知される．PMM$_j$ は，送られてきたアクセス条件とブロック名を AT のアドレス変換テーブルに書込む．この書込みが終了すると UP は自動的にメモリアクセスを再開する．

3.2 エレメントスイッチング方式

エレメントスイッチング方式は，メッセージエレメ

ントに分解されたメッセージを，リンク確立後メッセージエレメントごとに通信路を確保しながら μP 間でデータを交換する手段である．転送されるデータの種類は，システム管理用のコントロールデータ，相互診断用の確認データ，ユーザプログラム間でのメールなどである．従って，μP 間のデータ交換は PM 間の通信とみなすことができる．ここではその核となる COM の機構を通して，その実現方法を説明する．

3.2.1 メッセージの転送

COM は，図4のように主動的な役割りをするマスタコントローラと従属的な役割をするスレーブコントローラで構成される．マスタコントローラには，データ授受のための MDR レジスタ，転送種別コードのための MFUN レジスタ，到着モジュール名のための MDES レジスタ，出発モジュール名のための MNM レジスタが，また，スレーブコントローラにはデータ授受のための SDR レジスタ，転送種別コードのための SFUN レジスタ，リンク確立条件のための CMD レジスタ，到着モジュール名のための SDR レジスタからなる．そして μP はこれらレジスタを自由に読み書きできる．

モジュール間の通信は，一方の COM のマスタコントローラと他方の COM のスレーブコントローラの間で行われる．転送の方法は，1）MDR から SDR へ，2）SDR から MDR へ，3）MDR から CMD へ，4）CMD から MDR へ，の4種類ある．このうち，前述 1) および 2) は，メッセージエレメント転送用，3) はリンク確立用，4) は故障検出用である．

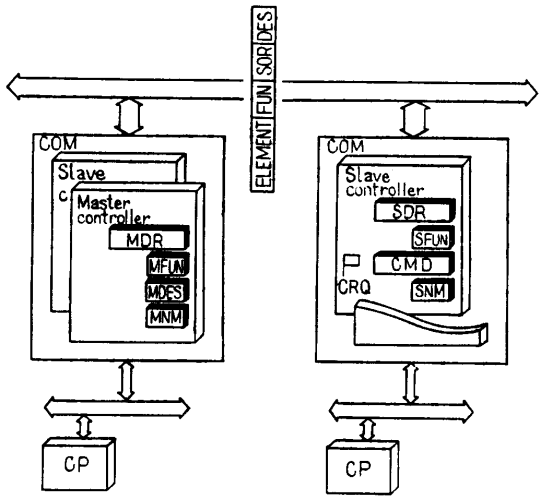

図 4 PM 間通信
Fig. 4 Inter-PM communication.

次にメッセージの送信について述べる．μP はメッセージをメッセージエレメントに分解した後で，MFUN に MDR から SDR へメッセージエレメントを転送せよという転送種別コードを，MDES に到着モジュール名を書込む．そして，最初のメッセージエレメントを MDR に書込み，マスタコントローラに対してスタートをかける．これによって，COM は，コントロールバスを確保し，MDR を ELEMENT に，MFUN を FUN に，MDES を DES に，さらに MNM を SOR に出力する．バス上にデータが出力されると各スレーブコントローラは，DES と SNM，SOR と CMD の下位4ビット（ここにはリンクを確立した時点で出発モジュール名が書込まれている）を比較する．そしてそれらが一致したスレーブコントローラの SDR に，ELEMENT が書込まれる．このようにして一つのメッセージエレメントが転送される．また，何らかの原因で転送が不可能な場合でも一定時間経過すると通信は終了する．通信の成功，不成功は，MDR の状態によって示される．すなわち，MDR が“空”の状態のとき，メッセージエレメントが既に転送できたことを，また“充”の状態のとき，何らかの原因で転送できなかったことを示す．

メッセージエレメントが転送できない原因として，1）DES と SNM および SOR と CMD の下位4ビットが一致するスレーブコントローラが存在しない，2）SDR が既に“充”の状態であるなどが考えられる．1)はソフトウェアまたはハードウェア上の故障，あるいはリンクの一時切断に起因し，2)は相手側でのメッセージエレメントの未処理に起因するものである．メッセージエレメントが転送できない場合には，ある一定時間繰返し再スタートをかける．それでも不成功の場合には，なんらかの通信不可能な障害が発生したと予想されるので，故障検出復帰ルーチンに入る．また，転送が成功した場合には，次のメッセージエレメントの転送に入る．

メッセージエレメントが転送されると SDR は“空”の状態から“充”の状態に変化する．これにより，スレーブコントローラ側では，新たなメッセージエレメントが転送されたことを知り，メッセージを組立てるために SDR からエレメントを読出す．読出しの結果，SDR は“空”の状態にもどり，次の受信が可能となる．

以上，メッセージエレメントの送信を中心に述べたが，受信は，SDR から MDR にデータが転送される

ことを除けば他は同じである.

3.2.2 リンク確立

PM 間で通信を行うときには，最初にリンク確立を行う必要がある．MICS-II のリンクにはレベルがある．リンクのレベルを設定するのは，マスタコントローラ側であるが，スレーブコントローラ側は，レベルを選ぶ権利がある．CMD の最上位1ビットは，リンク確立中かどうかを示し，続く3ビットは，確立中のレベルあるいはリンク確立の要求レベルを示し，残りの4ビットは，リンクを確立している出発モジュール名を示す.

あるモジュールがリンク確立を行う際は，MDR の下位4ビットに自分のモジュール名を，続く3ビットにレベルを書き，MFUN をリンク確立要求にして通信を行う．相手側が，リンク確立中でなく，要求したレベルが，希望しているそれよりも高いときには，MDR の内容が CMD に書込まれリンクが確立する．それ以外の場合には確立しない．しかし，通信の優先度制御を可能にするために，リンク確立中でも確立中のレベルより高い要求が生じたときには CRQ フラッグが立つ．従って，スレーブコントローラ側では，そのフラッグが立ったことを知って，現在確立中のリンクを一時切断することが可能である．すなわち，CMD の最上位1ビットを未確立にすることによって，今まで確立していたレベルよりも高いレベルのリンク確立を要求することが可能である.

この様子を示したのが**図5**である．今まで COM_1 と COM_3，COM_2 と COM_4 でリンクを確立して通信を行っていたが，PM_2 で PM_3 への緊急を要する通信が生じたと仮定する．この場合，COM_2 側の μP は今までの通信を一時切断するためにマスタコントローラのレジスタ類を退避する．そして，緊急の通信のためレジスタ類をセットしリンク確立要求の通信を起こす．しかし，この要求は，MDR から CMD への転送が不成功のため成立しない．そこで，COM_2 側では，

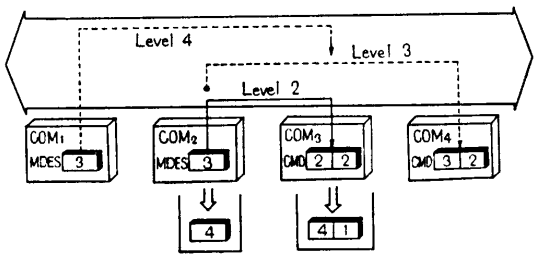

図5 コミュニケーションリンク
Fig. 5 Communication link.

メッセージエレメントの転送不成功の場合と同じように繰返し要求を出す．一方 COM_3 側では，現在リンク確立中のレベル4よりも高いレベル2のリンク確立要求が生じたために CRQ フラッグが立つ．そこで，COM_3 側の μP は，スレーブコントローラのレジスタ類を退避して，CMD の最上位ビットを未確立にする．これによって，次のリンク確立要求のとき MDR の内容が CMD に転送され，COM_2 と COM_3 の間でリンクが確立する．このようにして，今までのリンクは一時切断され，緊急度の高い通信を実行するためのリンクが確立する．この通信が終了したとき，レジスタ類はもとの状態にもどされて再び前のリンクが復活し切断していた通信が再開される.

4. 結　論

MICS-II の目的は，マルチプロセッサシステムに係わる諸問題の研究を，利用者のイメージに即したマルチプロセッサ上で可能とすることにあった．そこで，システム管理部分とプログラム実行部分の分離が行われたが，これによって，柔軟性に富んだ実験システムを提供することに成功した．すなわち，システム技術の開発期においては，利用者は種々の形態を要求してくるが，実験システムを，ハードウェアの単位で構成可能とすることによって広範囲の要求に答えることができる.

また，システム管理部分とプログラム実行部分の分離によって，ソフトウェアおよびハードウェアの開発を短期間で行わせることに成功した．この開発に要した期間はわずかに2年間であり，研究に携わった人々は，筆者などの他，製作者1名，プログラマ1名であった.

MICS-II は，それ自体マルチプロセッサシステムの雛型とみることができるが，UM と CM の分離は計算機設計技術に大きな影響を与えると思われる．すなわち，ユーザプログラムとシステムプログラムの分離を，モードやレジスタセットの切換えで行っているシステムがいくつか現存するが，保護の強化を一層進める場合には，MICS-II の方式が一つの参考になると思われる.

なお，MICS-II は，1977年5月より，PM 6台，LMM 8kW，メインメモリ 64kW で稼動を開始している.

謝辞 本システムの研究を行うにあたって，常に有意義な助言を与えて下さる日本電気(株)中央研究所周

辺機器研究部木地部長，同所コンピュータシステム部鳳凰部長に深謝します．また，ハードウェアの試作を担当して下さった三野輪氏，OS の作成をして下さった日本タイムシェア(株)出来氏に感謝します．工技院の関係者の方々にもあわせて感謝します．

なお，本研究は通産省工業技術院大型プロジェクト「パターン情報処理システムの研究開発」の一環として行われているもので，MICS-II は対話型複合ターミナルプロセッサの小規模実験セットの通称である．

参考文献

1) Baskin, H. B., Borgerson, B. R., Roberts, R.: PRIME—A modular architecture for terminal-oriented systems, SJCC, pp. 431-437 (1972).
2) Wulf, W. A. and Bell, C. G.: C.mmp—A multimini-processor, FJCC, pp. 765-777 (1972).
3) Ornstein, S. M., Crowther, W. R., Kraley, M. F., Bressler, R. D., Michel, A.: Pluribus—A reliable multiprocessor, NCC, pp. 551-559 (1975).
4) Swan, R. J., Fuller, S. H., Siewiorek, D. P.: Cm*: a Modular, Multi-microprocessor, NCC, pp. 637-644 (1977).
5) Ohmori, K., Koike, N., Nezu, K., Suzuki, S.: MICS—A Multi-Microprocessor System, IFIP, pp. 98-102 (1974).
6) Ohmori, K., Koike, N., Yamazaki, T., Ohmiya, T., Nezu, K.: MICS-II—A Virtual Machine Complex Controlled by Dedicated Microprocessors, COMPCON Spring (1978).
7) 大森健児，小池誠彦，大宮哲夫：MICS マルチマイクロプロセッサシステムについて，情処・計算機アーキテクチャ研資 75-8 (1975).
8) 小池誠彦，大森健児：マルチマイクロプロセッサの制御方式，信学会研資 EC 74-11 (1974).
9) 山崎竹視，大森健児，小池誠彦，大宮哲夫，出来哲史郎：マルチプロセッサシステム MICS-II のシステムコントロール，信学会研資 EC 76-75 (1977).
10) 大森健児，小池誠彦，山崎竹視，大宮哲夫：マルチプロセッサシステム MICS-II による並列処理，信学会研資 EC 77-64 (1978).
11) Ohmori, K., Koike, N., Yamazaki, T., Ohmiya, T., and Nezu, K.: System Management of MICS-II——A Virtual Machine Complex, UJCC, pp. 425-429 (1978).
12) 大森健児，小池誠彦，山崎竹親，大宮哲夫：計算機複合体 MICS-II のシステム評価，情処学論，Vol. 20, No. 2, pp. 130-137 (1979).

(昭和53年3月1日受付)
(昭和53年12月21日採録)