マイクロコンピュータ　60－5
とワークステーション
（１９９０．５．８）

高性能リアルタイム・カーネル：ＶＲＴＸ３２

深沢　圭志
エム・シー・エム・ジャパン株式会社

米国ＲＥＡＤＹ　ＳＹＳＴＥＭＳ社（カリフォルニア州）の開発した機器組み込み用高性能リアルタイム・カーネル：ＶＲＴＸ３２とその応用プログラム開発環境（ＶＲＴＸ　ＶＥＬＯＣＩＴＹおよびＣＡＲＤｔｏｏｌｓ）について述べる。ＶＲＴＸ３２は、応用プログラム部、カーネル部、デバイス・ドライバ部が完全独立・非依存となるソフトウェア・コンポーネント手法を採用、各部の開発性・移植性が向上する。さらにシステム変数のデータ構成やサーチ手法の改良により、システム負荷（タスク数、キュー数等）に非依存でフラットな高パフォーマンス（システムコールのオーバーヘッド、割り込み待ち時間、タスクスイッチ時間）を得ることができた。

HIGH PERFORMANCE REAL-TIME KERNEL: VRTX32

Keishi Fukasawa
MCM JAPAN Ltd.
Syuukaen Bldg., 2-11-1 Komazawa, Setagaya-ku, Tokyo 154, Japan

The high performance real-time kernel for enbedded applications: VRTX32 and its integrated development environment (VRTX Velocity and CARDtools), developped by READY SYSTEMS Co. in California, USA, is discussed. VRTX32 is using "software component" approach with a set of independent layers like application program layer, kernel layer, and device driver layer, so that this allows users to have better portability and productivity. Besides, new design of data structure and search algorythm gives fast and consistent performance (system call's overhead, interrupt latency, and rescheduling time) against system loads such as number of tasks and queues.

1　VRTX32とは

1981年5月に設立された米国レディ・システムズ社（ハンター&レディ社として設立、1987年1月にレディ・システムズ社に改名）によって設計・開発された高性能組み込み用リアルタイムO.S：VRTX32について述べる。

VRTX32は、比較的単純な応用プログラムだけでなく、大規模なリアルタイムOSを必要とする種々の応用プログラムに対応するための多様なユーティリティ機能（ファイルI／O管理、ネットワーク間通信等）をサポートしている。それらの機能を必要に応じて組み合わせることで、例えば、密結合型多重処理システムからUNIX、VMSを中心としたTCP／IPベース・ネットワーク環境まで対応する構成が作れる。

2　VRTX32の主機能

2．1　システム構造

VRTX32は、応用プログラム開発時にリアルタイム処理機能（スケジューラ、タスク間同期・通信、メモリ管理、割り込み処理等）を提供している。図1に示すようにVRTX32は、応用プログラムとハードウェアの間に位置し、完全なモジュール構成をとっている。このモジュール化によってVRTX32は、ハードウェアでは困難な機能を提供だけでなく、周辺デバイス等のハードウェアに固有の特長を応用プログラムから見る必要なくできる利点（インフォメーション・ハイディング）がある。

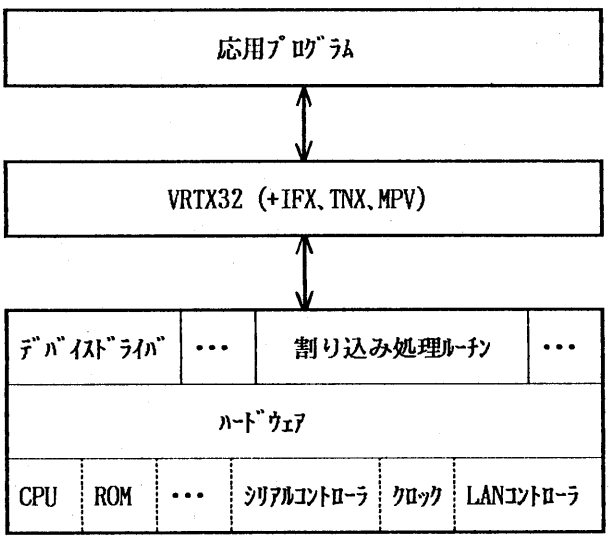

図1　VRTX32システム構成図

2．2　マルチタスク機能

VRTX32の持つ機能のうち主要なものとしてマルチタスク機能があげられる。通常のプログラムでプロシージャに相当するものがリアルタイム・プログラムではタスクとなる。タスクは他のユニットから切り離して考えられる構造単位であり、同時性の単位でもある。マルチタスク・システムは複雑なタスクを平行して処理しなければならない。VRTX32は完全なマルチタスク機能の他にも動的メモリ管理機能、実時間時計、キャラクタI／O、割り込み処理をサポートする。

2．3　高速性

VRTX32は豊富な機能だけでなく、高速な応答性能をシステムコールのオーバヘッド、割り込み応答時間、リスケジュール時間の全てで実現している。これにはシステム・データ構成、スケジューリング・アルゴリズム、検索アルゴリズム等の改良によるところが大きい。表1に性能表の抜粋を示す。尚、レディ・システムズ社では全てのシステムコールに関して同様のデータを用意している。

2．4　性能の一定性

VRTX32の最も大きな特徴は、性能特性が一定に確立されていることである。通常、リアルタイム・システムのタスク数は、10～100の間を推移しているが、VRTX32の場合には、そのタスク数に依存せず均一の性能を発揮するよう設計されている。

VRTX32機能	ｵｰﾊﾞﾍｯﾄﾞ (μs)
ｼｽﾃﾑｺｰﾙ	
ﾀｽｸ 生成	60
ﾒﾓﾘﾌﾞﾛｯｸの獲得	27
ﾒｰﾙﾎﾞｯｸｽへの送信	23
ﾒｰﾙﾎﾞｯｸｽからの受信	19
ｼｽﾃﾑ動作	
割り込み応答時間	10
ﾘｽｹｼﾞｭｰﾙ時間	15
条件：MC68020ﾌﾟﾛｾｯｻ(25 MHz) 1ｳｪｲﾄ・ｽﾃｰﾄ ﾀｽｸ 数、ｷｭｰ 数等： 2~64	

表１　ＶＲＴＸ３２性能表[1)]

図２に示すようＶＲＴＸ３２の性能は、システム変数（タスク数、キュー数、メールボックス数、割り込み）に非依存になっている。したがって、タスクのスケジューリング遅延は、実質上一定に保たれる。これは２段階ハッシュ・テーブルを用いたサーチアルゴリズムの採用によるもので、例えばタスクのインサーション作業も常に一定時間内に終了する。ＶＲＴＸ３２のこの決定型アルゴリズムによりスケジューリング時間が予め予想できるわけで、高信頼性が要求される応用プログラムの作成において大きな利点となると思われる。つまり不定応答型ＯＳでは、ワーストケースにおけるＯＳの応答時間の評価が多くの場合、非常に困難であるからである。

３　ソフトウェア・コンポーネント思想

ＶＲＴＸ３２は標準的ハードウェア・コンポーネントのように扱え、かつ多様なハードウェアやソフトウェアの環境に何の変更もなくビルディング・ブロックのように組み込めることを特長とする。その結果として、高度な柔軟性と一般的なプログラム開発アプローチでは得られない簡易さを提供している。以下に代表的な特徴を述べる。

(1)　広範なマイクロプロセッサのサポート

ＶＲＴＸ３２は１６ビットから３２ビットまでの多くの汎用マイクロプロセッサをサポートしている。

ﾓﾄﾛｰﾗ 社　：　680x0ｼﾘｰｽﾞ、
88000 ('90Q2予)
ｲﾝﾃﾙ社　：　80x86ｼﾘｰｽﾞ、80386 、
80960Kx ('90Q2予)
AMD 社　：　29000
NS社　：　32000ｼﾘｰｽﾞ
TRON仕様ﾁｯﾌﾟ：
Gmicroｼﾘｰｽﾞ(日立、富士通、三菱)
TX-1 (東芝)
MIL-STD-1750A

同一ユーザ・インターフェースを持ちながら様々なマイクロプロセッサをサポートするＶＲＴＸ３２は、プロジェクトによって異なるマイクロプロセッサを採用しなければならない状況下におけるＯＳの標準化に有効であると思われる。

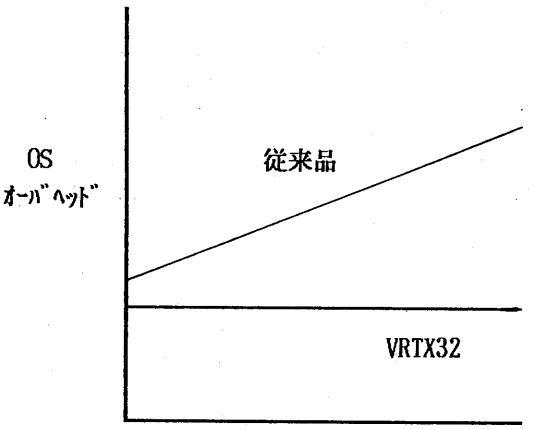

図２　ＶＲＴＸ３２性能特性[2)]

(2)　ハードウェア環境に対する非依存性

ＶＲＴＸ３２は、その設計段階においてハード

ウェア環境に対する仮定を最小限度しか設けていないため、割り込み、クロック、メモリ管理、システム・バス、Ｉ／Ｏデバイスに関してＯＳの移植性に制限が加わることはない。つまり、ＯＳ自体は何の変更の必要もなく周辺ハードウェアの初期化ルーチンおよびデバイスドライバのみの変更・追加のみでどのようなマイクロプロセッサ・ボードへも移植可能となる。

(3) ポジション非依存

ＶＲＴＸ３２は、約８ＫＢと非常にコンパクトなコード・サイズとなっており、完全にポジション非依存である。プログラム・カウンタやベースアドレスから相対アドレスで割り付けるため任意のアドレス空間に割り付け可能となっている。システム初期化時に参照すべきアドレスはポインタとしてコンフィギュレーション・テーブルに格納される。コンフィギュレーション・テーブルには他のユーティリティのアドレスを指示するコンポーネントベクタ・テーブル（ＣＶＴ）を指すポインタも含んでいる。図３にソフトウェア・コンポーネント・インターフェースを示す。

(4) 開発ツール非依存

ＶＲＴＸ３２の機能は、プロシージャやファンクションコールとしてでなく「ソフトウェア割り込み」インストラクションによって起動される。したがって、ＶＲＴＸ３２と応用プログラムをリンクする必要はなく、特定のリンカーに依存することもないわけである。応用プログラムを高級言語で開発する場合にも簡単なライブラリ・ルーチンでＶＲＴＸ３２とインターフェースがとれるため、言語、コンパイラ、開発システム等に依存せずＶＲＴＸ３２を使った応用プログラムの開発が可能である。

(5) 拡張性

ＯＳ機能を拡張するためにいくつかのユーザ定義ルーチンを指定できるようになっている。まずＣＶＴを通して起動されるユーザ定義システムコールを作成できる。応用プログラム固有のシステムコールを簡単に付加することが可能である。

次に、ＴＣＢ拡張ルーチンを起動させるためのフック（ＨＯＯＫ）を３種類用意している。これらフックを利用することで浮動小数点デバイスやＭＭＵをサポートできる。

４　ＶＲＴＸ３２システムコール

４．１　タスク管理

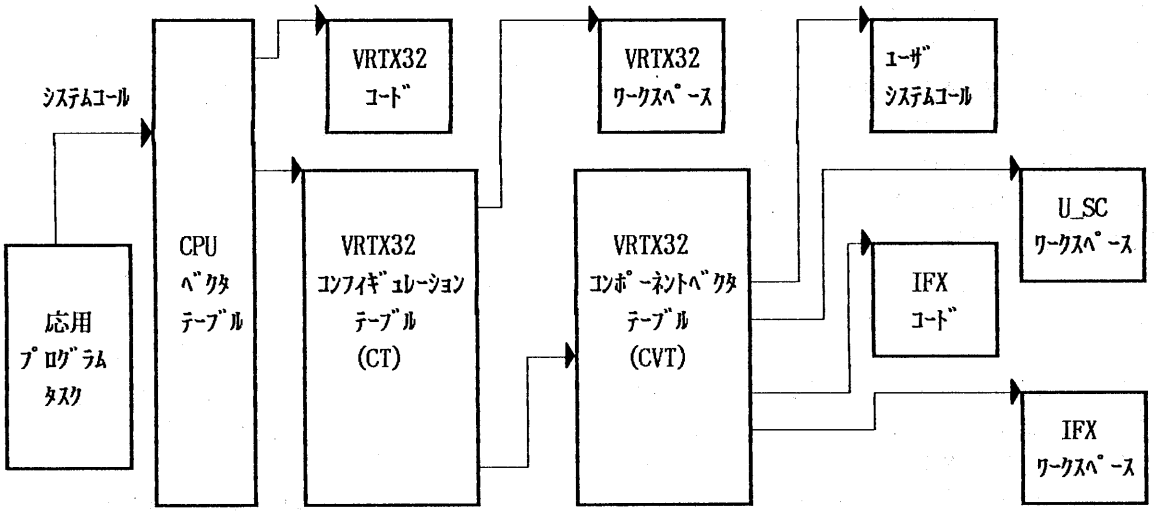

図３　ソフトウェア・コンポーネント・インターフェース

ＶＲＴＸ３２はマルチタスク用スケジューラに加えて、タスクの生成、削除、強制待ち、再起動などのシステムコールを提供している。ＶＲＴＸ３２ではリアルタイム事象に対して最高速な応答性を保証するために先取り優先順位型のスケジューリング手法を採用した。

ＶＲＴＸ３２スケジューラはタスクの認識番号（タスクＩＤ）と優先度（プライオリティ）によってタスクの管理を行なう。タスクＩＤとプライオリティは、タスク生成時に指定される。タスクＩＤはタスクを選択的に管理するために必要であり、各タスクには固有のタスクＩＤが設定される。タスクＩＤは１～２５５の数字で現わされるため２５５のタスクの個別管理が可能で、かつタスクＩＤの０は個別管理をする必要のない複数のタスク（例えばサーバ・タスク）に割り付け可能である。ＶＲＴＸ３２ではプライオリティよってタスク間の相対的実行緊急度が決定されるが、システムコールにより動的にプライオリティを変更することもできる。このプライオリティには２５６段階あり、０が最も高く２５５が最も低いプライオリティとなっている。同じプライオリティの場合にはタイムスライス・オプションを指定できるが、指定のない場合にはタスクが実行可能状態（レディ状態）になった順序通りにスケジュール順位が決定される。図４にタスク状態の遷移図を示す。

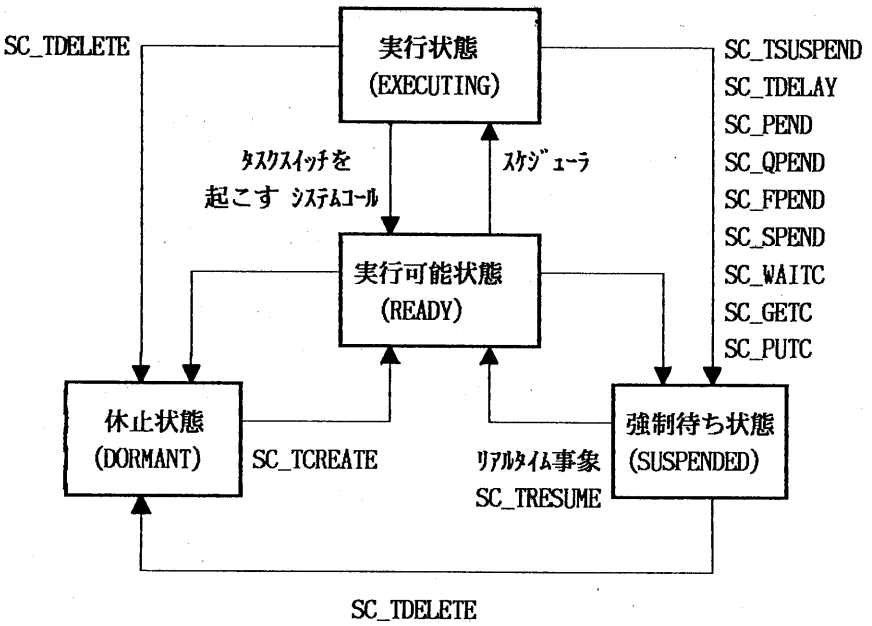

図４　タスクの状態遷移

４．２　タスク間同期・通信

ＶＲＴＸ３２はリアルタイム用マルチタスクシステムが必要とする通信および同期の効率的手段を提供する。タスク間や割り込み処理ルーチン（ＩＳＲ）との排他制御、通信、同期をサポートするためにメールボックス、キュー、セマフォ、イベントフラグを用意している。

メールボックスは単メッセージ（任意のロングワード）の送受信をサポートすることで通信・同期機能を実現でき、完全な動的特性を持つため生成・削除の操作が省かれる。キューは複数のメッセージを持つ事ができ（メッセージ数はキュー生成時に指定）、タスクのメッセージ送信回数のスピードがメッセージ受信スピードより一時的にでも速くなる場合、有効なバファリング機能となる。キューはプライオリティ順、ＦＩＦＯ順だけでなくメッセージ・ジャミング機能を利用することでＬＩＦＯ順も実現できる。イベントフラグは３２ビットのイベントフラグ・グループとして構成され、事象の発生をビット・シグナルで通知するような場合に有効である。ビット・パターンをみてＯＲ型かＡＮＤ型の同期がとれ、更に複数タスクに同一事象を対応させることも可能である。計数型セマフォはダイクストラの手法を採用し、共有資源の排他制御を行なえる。

４．３　メモリ管理

ＶＲＴＸ３２は自由に使用できるメモリブロック・プールを維持することでメモリの動的割り付けをサポートしているため、タスクは実行時にメモリの割り付け・開放が可能である。リアルタイムシステムで最重要なのは予測可能なシステム動作であり、万が一タスクの要求したサイズのメモリブロックが確保できず遅延を生じることがあればシステムの信頼性は致命的に低下するのは明白である。一般に可変長ブロック割り付け法は全てある条件のもと予測不能な応答時間をもたらすが、ＶＲＴＸ３２では固定長ブロックによるメモリ管理を行なっているため常に一定の応答時間を維持できる。メモリ空間を十分効率的に使用するために、フリープールは動的にパーチション分割される。各パーチションはサイズの異なるブロックを指定可能（ブロックサイズに対する制限はない）になっているので、タスクの要求するメモリ容量に最も近いブロックサイズのパーチションを割り付けることでメモリの浪費を最小限に押さえられる。この固定長ブロックの採用は検索のオーバヘッドを無くし、メモリ・プールのフラグメンテーションも無くす役割を持つ。メモリの効率的使用および高速性能の向上を実現した。図５にパーチションに分割されたメモリプールの例を示す。

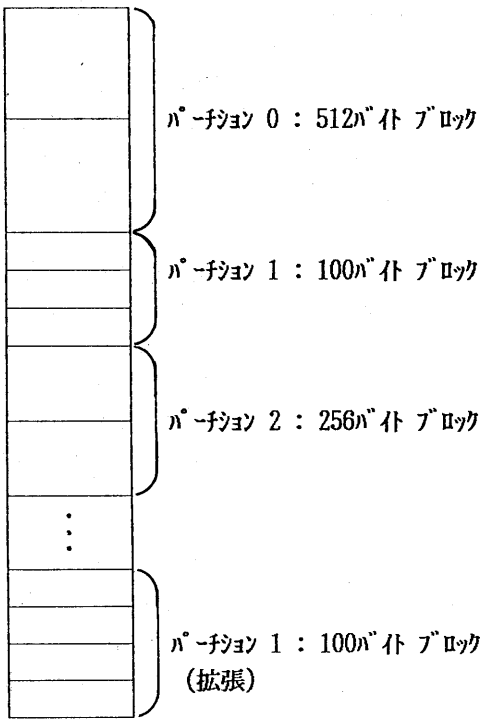

図５　パーチション分割されたメモリプール

４．４　割り込み処理

ＶＲＴＸ３２ベースのシステムでは、ユーザの記述した割り込みハンドラがタスクとデバイスのインターフェースをとっている。ＣＰＵが割り込みを認めるとＣＰＵは即座に制御をそのデバイスに関係する割り込みハンドラに渡し、割り込みハンドラは直接ハードウェアによって、ＯＳのオーバヘッドなしに起動される。同様にＶＲＴＸ３２も割り込みハンドラの処理により制限を受けない。割り込みが発生すると以下の２つの処理を即座に行なわねばならない。

(1) 割り込み処理を開始するためにデバイスからの入力に対し必ず応答しなければならない。タスクには割り込みの発生が通知されねばならない。
(2) 割り込みを受け付けない時間を最短にするため、同一もしくは低い優先度の割り込みはイネーブルにしなければならない。

したがって、割り込みハンドラは通常アセンブラコードで書かれた比較的小さなプログラムとして実装される。

割り込みハンドラをサスペンドさせない多くのＶＲＴＸ３２システムコールが割り込みハンドラから発行でき、タスクとの通信・同期をとるために利用できる。

５　ＶＲＴＸ３２のシステム構成設定と初期化

５．１　システム構成の設定

ＶＲＴＸ３２を実行させる前準備として、システムにその実行環境を知らせるいくつかの設定が

必要になる。例えば、ＶＲＴＸ３２の先頭アドレス、システム・ワークスペースの大きさ、応用プログラム固有のオプション、最大タスク数、フック・ルーチンのアドレス等をＶＲＴＸ３２コンフィギュレーション・テーブルに設定しておく。このシンプルなテーブルはメモリ上のどこにでも置くことが可能になっている。図３で示したようにＶＲＴＸ３２とコンフィギュレーション・テーブルはＣＰＵベクタ・テーブルに登録されており、ＶＲＴＸ３２が初期化されるとき、ベクタ・テーブルを見てコンフィギュレーション・テーブルの位置を獲得する。もうひとつのベクタはＶＲＴＸ３２のエントリ・ポイントを示し、システムコールはこのベクタを用いてＶＲＴＸ３２のファンクションを起動する。これら２つのベクタをＣＰＵベクタ・テーブルのどこに置くかもユーザが自由に指定できるようになっており、ＶＲＴＸ３２が使用するベクタはこれら２つのみで他のベクタは全て応用プログラム用に開放されている。

５．２　システムの初期化

ＣＰＵがリセットされるとＣＰＵはユーザの書いて初期化ルーチンを起動する。初期化の詳細部分は必然的に応用プログラムごとに異なるが、以下に典型的な初期化手順を示す。

(1) メモリ・サブシステムがあれば初期化する。（例えばMMUマッピング・レジスタ）
(2) コンフィギュレーション・テーブルのアドレスとＶＲＴＸ３２のエントリ・ポイントをベクタ・テーブルに登録する。
(3) ＶＲＴＸ＿ＩＮＩＴコールを発行する。（これによりＶＲＴＸ３２はシステム内部変数を初期化してリターンする）
(4) 割り込みコントローラ、ハードウェア・タイマ、シリアルＩ／Ｏコントローラ等を初期化する。
(5) システム初期化時の実行環境に必要なタスク、キュー、メモリ・パーチションなどをＶＲＴＸ３２システムコールを発行して作成する。
(6) ＶＲＴＸ＿ＧＯコールを発行してマルチタスキングを開始する。（レディ状態のタスクをプライオリティ順に実行していく）

６　統合開発環境

６．１　ＶＲＴＸベロシティ

ＶＲＴＸベロシティはリアルタイム組み込みプロセッサ向けに統合化されたＵＮＩＸベースのクロス開発およびランタイム環境である。ＶＲＴＸベロシティ環境内でのターゲット・ボードと開発ＵＮＩＸホストとのインターフェースとしては、ＴＣＰ／ＩＰプロトコル、ＶＭＥバスによるバックプレーンリンク、シリアルリンクをサポートする。図６にＶＲＴＸベロシティ環境構成例を示す。ユーザ・フレンドリなウィンドウ・インターフェースはマルチタスク・システムの直観的理解に有効であり、例えば複数オープンしたウィンドウにタスクを割り付けてデバッグが可能になる。さらにそれらのウィンドウよりＣソースレベル、アセンブラレベル、機械語レベルのリモートデバッグが可能になっている。

ボードレベル・デバッガとしてのＶＲＴＸベロシティはメモリおよびレジスタのモニタ、ブレークポイント、ＶＲＴＸシステムコール機能、開発ホストシステムとの通信、ＶＲＴＸ構造体のモニタ等の重要な機能をサポートしている。この機能はＲＴｓｃｏｐｅによって実現されている。

Ｃソースレベル・デバッガは複数のタスク／ターゲットプロセッサを一度にデバッグできる。それぞれ独立したサブウィンドウをタスクに割り付け、Ｃソースでデバッグ・トレース作業が可能なマルチタスク環境サポートになっている。ユーザはいつでもウィンドウのオープン／クローズができ、ウィンドウがクローズされてもデバッガは活動状態になっている。シングル・ステップ実行、様々なタイミングでのブレークポイント、変数値のモニタ・修正等がウィンドウ上で実行可能である。この機能はＲＴｓｏｕｒｃｅというモジュー

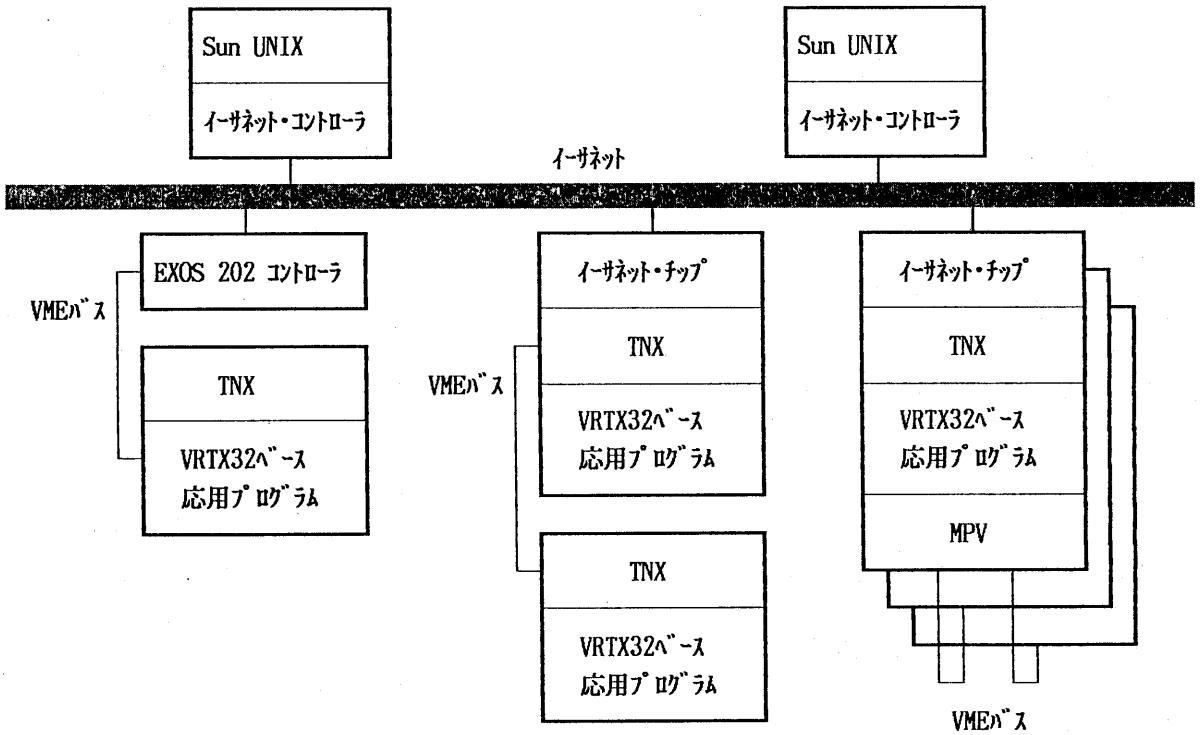

図６　ＶＲＴＸベロシティのＴＣＰ／ＩＰ（ソケット・インターフェース）を利用した構成例

ルを通して行なわれる。

６．２　ＣＡＲＤｔｏｏｌｓ

ＣＡＲＤｔｏｏｌｓは組み込み型リアルタイムシステム開発を支援するために設計された統合ソフトウェア開発支援ツールで、ＶＲＴＸとの統合によりＶＲＴＸベースでの応用システム開発に特に効果をあげる。従来のデータフロー図に加え、タスクマップと呼ばれる詳細レベルの記述からタイミング検証機能までサポートしており、マルチタスク・プログラム開発が持つ固有の複雑性をも記述可能としている。

ＣＡＲＤｔｏｏｌｓはオブジェクト指向型設計手法をサポートしている。構造化設計／構造化解析の他にインフォメーション・ハイディング、データ・タイピング等、例えばＡｄａのパッケージ定義を含むＡｄａ用ＰＤＬ（プログラミング設計言語）をサポートしている。

参考文献

1) VRTX32/680x0 Timing Reference Manual, Ready Systems Corp., Sunnyvale, USA (1987)
2) VRTX32/68020 USER'S GUIDE, Ready Systems Corp., Sunnyvale, USA (1987)