LHD制御用実時間計測システムに関する研究

総合研究大学院大学博士論文

1999年6月

刈谷丈治

目次

1 序論

LHD(Large Helical Device)[1]の建設と超伝導コイルの短尺導体試験などが進行していく中、LHD 完成時に運用する計測システムのあるべき姿を目指した核融合科学研究所の共同研究、「ワークステーションを用いたデータ収集・解析・制御システムの研究」が編成された。筆者は平成5年よりソフトウエア・エンジニアリングの立場から参加し、計測システムのプロトタイプの研究開発をおこなってきた。

核融合による発電方式の研究は、核分裂による原子力発電方式が有効な核廃棄物処理対策を打ち出せず、将来性が危ぶまれている現在、核廃棄物処理が容易[2]という利点から、重要性を増している。そのためにも LHD の建設を成功させ、十分な研究ができる環境を整えることが重要である。

LHD のような大規模システムには、数々の広い分野にまたがった装置の研究開発から、それらを運用するための計算機システムなどが必要であり、各方面の専門家を結集して構築しなければならない。最近の大規模システムには計算機の利用が不可欠であり、計算機がシステム統合の基盤となるため、計算機をうまく使いこなすことが、システム運用の鍵となることが多い。その意味で、専門家の中でも、計算機あるいは情報処理に関する専門家の関与は是非とも欠かせない。

このような大規模システムの開発を行うには、システム全体を見通し、どのような構成にするか、何を基本的な道具にするかなどの基本設計が重要になる。このときに指針を与える学問をソフトウエア・エンジニアリングと言い、さまざまな開発手法が提唱されているが、その有効性については明確になっていない。その理由として、失敗したプロジェクトは表面に現れにくいこと、同一プロジェクトに対して、個々のソフトウエア開発手法を適用した場合と適用しない場合の両方で開発するということがなく、直接比較ができないこと、その手法が正しく適用されたかどうかは適用した個人に依存することなどが考えられる。このため、個々の手法の有効性を主張するには、成功例に使用された手法という評価を積み上げていくしかない。成功例として数えられるには、開発対象がある程度の規模であり、かつ有用なシステムでなければならない[3,4,5,6]。

共同研究では、2000 項目に達するデータをリアルタイムに表示するシステムを設計することが目標として掲げられていたので、構築目標である表示システムは、ソフトウエア・エンジニアリングにおける手法を適用することが必要な、十分に大規模であるシステムである。本研究に参加することは、ソフトウエア・エンジニアリングの有効性を確認する貴重な機会であり、重要な研究である核融合研究に役立てるという、二重の目的がある。

尚、巻末に本論文で使用している用語について、簡単な解説をしてある。

2 データ処理技術の現状と大型実験装置に求められる課題

2.1 データ処理の現状

以下にプラズマ核融合実験に利用されているデータ処理について簡単にレビューを行う。プラズマ核融合実験用のデータ処理はバッチ処理だけだったので、リアルタイムデータ収集の例も調べた。

2.1.1 Alcator C-Mod のデータ処理

Alcator C-Mod におけるデータ処理 (http://www.pfc.mit.edu/mdsplus/ps/cmod-daq.ps) は、MDSplus という、実験管理、実験制御、データ収集、データ解析など、実験を総合管理するソフトウエアにより実施されている。MDSplus は、MIT Plasma Science and Fusion, Los Alamos National Laboratory と Istituto Gas Ionizzati が共同で開発したシステムである。

Alcator C-Mod の中心となる計測装置は CAMAC モジュールであり、350 を超える CAMAC モジュールが約 50 の CAMAC クレートに装備され、4 系統のハイウェイにより 3 台のサーバ計算機に接続されている。これらの計測装置により 1600 種を超える信号を計測する。CAMAC モジュールのうち 200 台は ADC であり、数 KHz から数 MHz のサンプリング速度でデータを変換する。サーバー計算機のほかに約 40 台のワークステーションがあり、サーバ計算機が保存したデータを取り出し、表示や解析をおこなう。

Alcator C-Mod は 10 分から 15 分周期でパルス運転をおこなうトカマク装置であり、データ収集のシーケンスは次のようになる。パルス期間中は計測したデータを装置内メモリに格納する。数秒間のパルスが終了した後、重要なデータは数秒以内に、のこりの全データも 5 分以内に収集され解析のために分配される。パルス運転の準備のために2, 3分かかるので、残りの時間の間に解析を終え、次の実験のために設定をおこなう。

1 回のパルス運転で収集されるデータ量は 80MB を超える。実験装置が建設されたときと比較し、データ量は 8 倍にもなったが、ネットワーク、ワークステーション、ディスク装置などを増設、高性能化することにより、対処している。

MDSplus は Alcator C-Mod のようなパルス運転に適したバッチ処理システムであり、米国の大学や研究所 20 ヶ所以上で広く利用されていて、一種の標準ソフトになりつつある。

2.1.2 JT-60 のデータ処理

JT-60 は日本原子力研究所の臨界プラズマ実験装置で、放電時間15秒以下、放電周期15～30分のトカマク装置である。青柳[7]はそのデータ処理設備を開発している。この処理設備の処理方式は、大型汎用計算機にデータを集めてショット間に処理をおこなうバッチ処理方式である。この大型汎用計算機をショット間計算機と呼ぶ。データは主に3つのルート、CAMAC、大容量記憶システム(Transient Mass Data Storage System, TMDS)とワークステーションで収集される。ショットごとのデータ量は、CAMAC データが約 25MB、TMDS データが約 300MB、ワークステーション・データは 1MB 以下である。

CAMAC クレートはいくつかのグループに分かれ、グループごとにシリアル・ハイウェイで CAMAC インターフェース制御装置(CAMAC Interface Control Unit, CICU)と接続している。CICU はショット間計算機に接続しているデータ収集 LAN と接続している。CAMAC データの収集はこの CICU が直接おこなうため、300～400KB/s の転送速度を達成している。

TMDS データは最大 200K サンプル/秒で 60 チャネルのデータを 300MB まで記録できる。

ワークステーションは、データ収集というより計測データを分散処理するためのもので、ショット間計算機上での利用方法と同じインターフェースでワークステーションからショット間計算機上のデータを使用できるようになっている。

2.1.3 REPUTE-1 のデータ処理

REPUTE-1 は東京大学システム量子工学専攻に設置されている逆転磁場ピンチ型高温プラズマ実験装置である。森川ら[8]はそのデータ収集システムを開発している。システムはミニコンによるスタンドアロン型システムであったが、実験形態の変化に対応して一新し、現在は CAMAC システム、VXI システムおよび UNIX ワークステーションに大別される小システムを、LAN で統合したものになっている。

CAMAC システムではシリアル・ハイウェイを使わず、直接制御用計算機（ボード型ワークステーション）と接続している。CAMAC システムで収集したデータをサーバ・コンピュータ上のファイルに書き込むには、

CAMAC data highway ->

CAMAC パラレル・バス・クレート・コントローラ(PCC) ->

VME バス・インターフェース ->

制御用計算機 ->

光イーサネット ->

サーバ・コンピュータ

という経路をたどる。この経路上でもっとも性能が悪い場所は PCC と VME バス・インターフェースとの間の最大転送速度 1.14MB/s である。ファイル書き出し速度の測定値では、ほぼこの最大転送速度がえられている。CAMAC 自体がバスを中心とする独立した装置であるため、REPUTE-1 の制御用計算機など他の計算機と接続するには、CAMAC のバスと通信するための制御装置を必要とする。

VXI システムでは、スロット 0 にボード型計算機を置いて各モジュールを制御するものと、スロット 0 に MXI(Multisystem eXtension Interface)ボードを置いて外部の制御用計算機で制御するものがある。後者の場合、23MB/s という高速のデータ転送速度がえられる。

2.1.4 TRIAM-1M のデータ処理

TRIAM-1M は九州大学応用力学研究所の超伝導トロイダル磁場実験装置である。上瀧ら[9]はそのデータ処理システムを開発している 。システムでは、7台の CAMAC クレートを使用し、1放電あたり1～2MB のデータを収集している。

TRIAM-1M の放電持続時間が伸び、CAMAC の記憶容量では不足してきたため、次に示すように CAMAC を使用したデータ処理法を改良し、トカマク・プラズマの連続運転に適用できるようにした[10]。

従来の数秒から数十秒のパルス運転のプラズマを対象としていたデータ処理では、放電と共にデータを CAMAC モジュールに集積し、放電終了後データを計算機に取り込んで解析を行う方式であった。この方式に対して、プラズマ持続時間の伸長に対処し、最終目標である連続運転に適用する場合に、次の問題があった。

(1)CAMAC モジュールでは、放電時間伸長に見合うメモリを用意できない。

(2)放電終了を待って解析をおこなっては、長時間あるいは連続運転中にフィードバックすることができない。

この欠点を解決するため、図 2.1 で示すように ADC 以降の機器を2系統用意し、一定期間毎に切り替えるサイクル処理を導入した。メモリ容量 8192 ワードの CAMAC モジュールを用い、サンプリング・タイムが 5ms では 40 秒、サンプリング・タイムが 10ms では

80 秒ごとに2系統を切り替え、データの集積と解析を並行しておこなうことで長時間連続処理を行っている。この方式で2時間におよぶ長時間放電の記録もおこなった[9]。

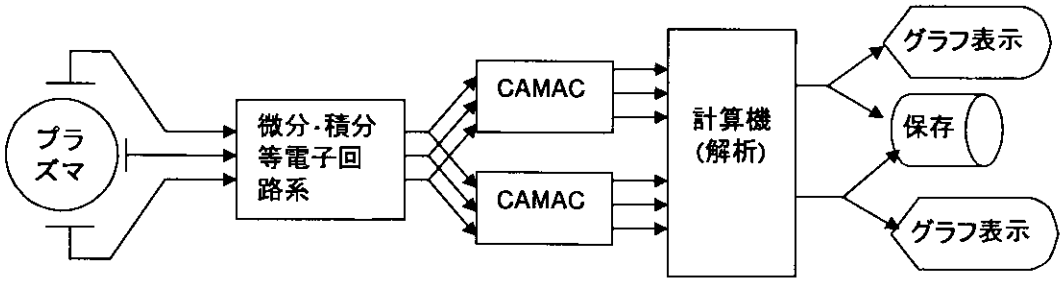


図 2.1 サイクル式データ処理

2.1.5 リアルタイムデータ収集の例

安ら[11]は、1991 年から 1992 年にかけて、高エネルギー物理学研究所(KEK)において UNIX ワークステーションを使ったリアルタイムデータ収集システムを開発した。開発したシステムは、KEK、東工大、東大原子核研究所、カミオカンデなどで使用されている。

従来高エネルギー物理学実験でのデータ収集システムでは、リアルタイム OS ではないという理由で、標準 UNIX を OS とするワークステーションを用いてこなかった。しかし CPU が高速化してきたため、デバイス・ドライバの実装上の工夫をすることにより、事実上リアルタイム処理が可能であることを示した。図 2.2 は開発されたシステムの構成であり、主要要素間のデータ転送能力を示してある。

CAMAC デバイス・ドライバおよびライブラリを開発して測定した性能のなかから、SPARCstation/IPX で測定した CAMAC シングル・データ転送と CAMAC ブロック転送のデータを表 2.1 に示す。ブロック転送は DMA 転送で

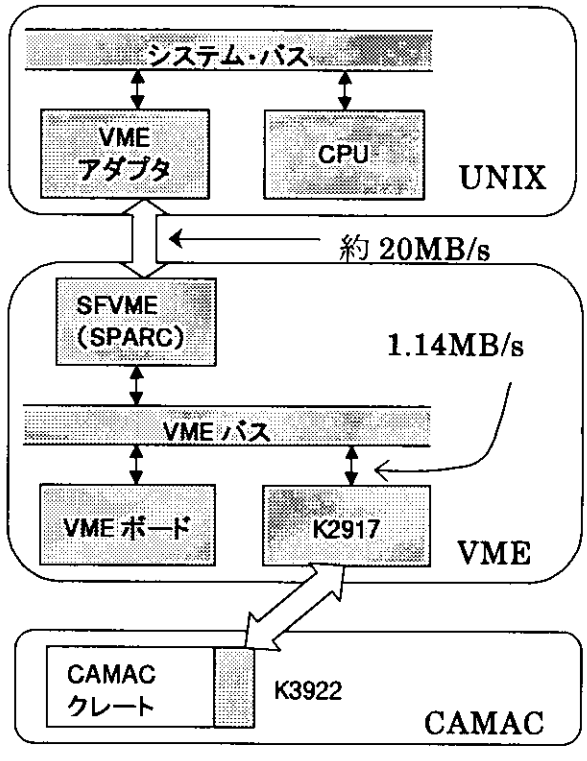


図 2.2 UNIX データ収集システム

あるから、ハードウエアの性能によってほぼ制限される。これに対してシングル・データ転送はプログラムにより制御される PIO であるから、CPU の性能に依存し、かつすべての CPU 時間を転送に費やすことはできないので上限の値となる。なお PIO の転送速度は、比較のため PIO 実行時間から計算した。

方式	転送速度
PIO Read(CAMACW)	15 KB/s
Block Read(CDMAW)	939 KB/s

表 2.1 CAMAC 関数による転送速度

データ収集システムの性能は、事象発生率と、事象ごとに取得するデータ長により、どれだけのデータを収集できたかにより評価する。安ら[11]は結果をデータ収集効率で表している。全データを収集するために必要な最低データ転送速度を分母に、収集できたデータ量を転送速度に換算した数値を分母として表 2.2 にまとめた。単位は byte/s である。表からは 256KB/s が限界であると思われる。

このシステムは UNIX を利用したリアルタイム・データ収集を可能にしたものだが、のちに述べる LHD に求められている速度を満足していない。この方式のボトルネックは VME インターフェースや CAMAC プログラムにあるため、UNIX が稼動する計算機を最新の高性能計算機に取り替えても改善されない。

データ長 byte	事象発生率			
	10Hz	100Hz	400Hz	750Hz
20	198/200	1880/2K	5920/8K	7200/15K
200	1980/2K	18.8K/20K	51.2K/80K	60K/150K
2000	19.8K/20K	170K/200K	216K/800K	240K/1500K
4000	39.6K/40K	240K/400K	256K/1600K	240K/3000K

表 2.2 転送速度

2.2 大型実験装置に求められる課題

前節では大型実験装置のデータ収集について、主にデータ量、転送速度、方式の面から概観した。

それぞれのシステムにおいて、建設時に比べ、多量のデータ、より高速の転送速度が必要になったための対策を施しており、構成要素の高性能化により達成した例、新しい方式を導入する例、あらたな仕組みを考案した例などがある。方式としてはほとんどがバッチ処理であるが、TRIAM-1M のように計測と保存を並行しておこなうもの、UNIX ワークステーションを使ってリアルタイム・データ収集をおこなおうとするものなど、リアルタイム処理への方向が見られる。これら既存の大型実験装置が改善策を施してきた方向、および LHD 計測システムに要請される機能を考えると、大型実験装置に求められる課題は次のような点にまとめられる。

1) 大容量高速転送の必要性

2) リアルタイム処理系とバッチ処理系の両方の必要性

3) 分散処理の必要性

実験装置の大型化に伴い、計測対象が増加し、また計測機器の高速化によりデータ量は増加する一方である。また計測機器単独ではこのデータを処理できず、データ処理は計算機でおこなうのが普通である。このため、計測機器と計算機との間での、大容量高速転送が必要である。

LHD では準定常運転をおこなうため放電時間が長い。またトカマク型装置でも放電時間が伸長してきている。このため、運転パラメータなどを放電中に変更することが必要となり、リアルタイムにデータを収集し、フィードバックをおこなうシステムが必要となる。現在バッチ処理でおこなっていることも、リアルタイム処理に移行する必要性が出てくるであろう。一方フィードバックの対象とならない高速観測データは依然としてバッチ処理しなければならない。このように、リアルタイム処理とバッチ処理の両処理方式が必要である。

計測対象が増加し、リアルタイム処理が必要となると、決められた時間内に処理を終えるために負荷分散をおこなうことが必然となる。分散処理に移行するとそのための通信が必要となるので、ますます大容量高速転送が必要となる。データ流通の観点から、計測装置からデータ処理用計算機まで、全体を統合した分散処理システムとして設計することが必要である。

3 LHD 実験での要求とシステム分析

本章では、LHD 実験での計測対象と特性から、どのような計測方法が必要とされるかを考察し、目的の計測システムを構築するための開発方針を明確にし、その必要性を説明する。次にシステムの規模とシステムに必要とされる性能について詳細な分析をおこなう。システム分析の結果必要となったマルチキャスト通信方式について実験をおこない、その結果利用可能との結論を得たことを説明する。この結果によりシステム構築は可能となり、システムの基本設計をおこなった。基本設計により必要となった構成要素について、システムの中での位置づけと簡単な機能説明をおこなう。

3.1 LHD 実験で要求される機能

LHD(Large Helical Device) はその名のとおり大規模な装置であり、必要な制御・データ処理システムも[1]のⅢ.4 章で解説されているように、大規模かつ複雑なものとなる。この制御・データ処理の対象の一部として、LHD 本体に取り付けられる約 3000 点の計測素子[1]がある。大別すると、低温系、電源系、真空排気系などの LHD 運転をおこなうための計測システムと、プラズマ実験をおこなうための計測システムに分けられる。

本研究では長時間計測対象として、まず低温系を選択し、それについての要求仕様を検討することから研究を開始した。理由は、LHD のもっとも大きな特徴である長時間放電を実現するために、超伝導コイルが使われているからである。さらに、これはリアルタイム処理の必要性とともに、短時間高速でデータ収集をおこなうための機能も備えている必要があり、本研究での計測方式を必要とする対象だからである。

制御をおこなうには、まず必要なデータを取得しなければならない。取得したデータを時系列データとして波形表示することで、取得できたことの確認をおこなうこととした。本研究では、約 3000 点の計測素子の中から、表示対象を 2000 点のアナログ信号に限定して検討をおこなった。アナログ信号に限定したのは、サンプリング速度による変化を見るためである。

LHD 実験は定常的長時間放電実験となり、トカマクに代表される従来の装置での実験に比べ、実験時間が桁違いに長くなる。このため実験中に継続的に制御をおこなわなければならないので、実時間制御となり、取得したデータを実時間表示しなければならない。LHD の制御では、超伝導コイルの電流や液体ヘリウムの流量の制御が中心となる。これらの対象は慣性が大きいため、たとえば緊急減磁でも 20 秒かかる[1]。このため 100ms 以内にフィードバックがおこなえれば良いと考え、実時間性の目標を 100ms とした。100ms 以内に計測データを表示モニターに表示するとして、この時間を、入力値の

変化が表示モニターに出力されるまでの時間として、レスポンス・タイムと呼ぶことにする。レスポンス・タイムが 100ms なので、表示周期も 100ms 間隔、すなわち10Hz とすることが妥当であると考えた。

以上をまとめると、本研究では、「2000 点のアナログ信号を 10Hz で、レスポンス・タイム 100ms 以内に表示すること」を中心となる要求仕様とした。

LHD で計測の対象となるのは、このように低速の信号だけではない。村岡ら[13]によれば、プラズマ電流については 10KHz、その他プラズマの位置、形状、ベータなどについても 1KHz での測定が必要とされている。またクエンチ発生時の電流や音なども高速サンプリングを必要とする。高速でリアルタイム処理ができない場合はバッチ処理をすることになる。システム内に複数の処理方式が混在するため、処理方式による違いを明らかにしておく必要がある。

リアルタイム処理がどのような場合に適用できるかを考察する。

低速のリアルタイム処理として、100ms のレスポンス・タイムに合わせると ADC(Analog to Digital Converter)のサンプリング・タイムを 100ms(10Hz)とする。ADC の生データは 2 バイト(16 ビット)で表現できるので、2000 チャネルのデータならば、40KB/s (=2000 チャネル×2 バイト×10Hz) のデータを ADC から表示モニターに送れれば良い。10Mbps のイーサネットを使って送信するとして、約 4%程度の通信量となるので、この通信は容易におこなえると思われる。標準 UNIX でのプロセス・スケジューリングの基本時間が 10ms であるため、100ms 単位の処理であれば1データずつ処理できる。このようにリアルタイム処理が可能になるためには、サンプリング速度の制限とデータ通信速度の制限の両方を満足する必要がある。

上の制限を満たさないためにリアルタイム処理できないデータは、バッチ処理で対処することになる。

プロセス・スケジューリングの基本時間からは、100Hz を超えるデータは1データずつ処理できない。このデータを 10Hz で表示しようとすると、1KHz のデータなら 100 サンプルを、10KHz のデータなら 1000 サンプルのデータをまとめて表示モニターに送る方式になる。この場合、10KHz のデータが 10 項目あると、それだけで 200KB/s の通信速度が必要となり、通信できないことも予想される。リアルタイムに送信できない場合は、いったん高速保存できるところ、ADC の内部や ADC を接続している計算機のローカル・ディスクに保存し、送信できる状況になったら取り出すというバッチ処理で対処しなければならない。このように時間精度がサンプリング速度より遅い問題はブロック単位での処理で回避できても、通信量の問題があるのでバッチ処理はどうしても必要である。

3.2 開発方針

前節で、LHD 計測システムはリアルタイム処理とバッチ処理に対応することとし、特にこれまでおこなわれていないリアルタイム処理については明確な性能目標を定めた。

システムの開発方針として次の目標を立てた。

LHD の建設は長期にわたるため、その間に発展する技術や製品を取り込んで継続的な性能向上を図るため、極力標準技術を採用する。

ソフトウエア技法のひとつであるプロトタイピングを採用し、まず小規模のプロトタイプ・システムを作成する。ただし、プロトタイプ・システムには、実用システムにまで発展できるように、拡張性と柔軟性を備えるように考慮する。

ソフトウエアとしては大規模になるので、プロトタイピングを始め、各種のソフトウエア技法を適用することが必要である。これまでに開発されたシステムで、同様な構成、目的や機能のものと比較し、そのシステムで使用されたソフトウエア技法を適用する。

以下にそれぞれの項目について詳述する。またデータ処理方式についての考察を進め、リアルタイム間欠処理とよぶ、特にイベント処理に有効な方式について説明をおこなう。

3.2.1 標準技術の採用により継続的な性能向上を図る

LHD の建設は 10 年以上にわたる。これは超伝導機器など、開発に長いリードタイムを必要とするシステムが多いためである。しかし、計算機の進展は一般にこのような機器に比べてきわめて急速に発達するので、大きな実験装置ができあがったときには、計算機システムが陳腐化していることが今までに知られていた。したがって、常に最新の計算機を使用できるようにするため、可能な限り標準的なハードウエア、ソフトウエアを使用することを基本方針とした。最新の高性能の装置と置換することが、性能を強化するための最も容易な手段になるからである。

多量のソフトウエアを作成する必要があるため、長期にわたりその資産が活かされる必要がある。少なくとも設計が活かされるようにしなければならない。近年ではハードウエアよりもソフトウエアの方により費用がかかるため、継続して使用できる、あるいは改造して使用できるソフトウエアは、経費面からシステム構築を支えるものであり、このためにも標準技術の採用が重要になる。

更に、これによって開発開始段階から運用に至るまでの間、同一のソフトウエアを使用できれば、基礎実験段階での結果から、システムが大規模になったときの動作を推測できるため、問題点があれば運用開始前に解決しておくことができる。

これらの理由から標準技術として、イーサネットによるネットワークと、UNIX ワークステーションを採用することとした。オペレーティング・システムとしてはリアルタイム OS という選択枝も考えられたが、

- CPU が高速になるとリアルタイム OS でなくても十分なレスポンス・タイムが期待できる。
- UNIX でもリアルタイム・プロセス用のスケジューリングが行える。
- LHD の定常運転時の制御に必要な時間精度としては、UNIX の基本時間単位である 10ms で十分である。
- プログラム開発環境は標準 UNIX の方が優れている。
- 機器が使えるかどうかは OS に依存し、最新の機器はなかなかリアルタイム OS では使用できない。

などが更に追加される理由である。以上により UNIX を採用した。

このような考え方は、例えば、青柳[7]は、時代遅れのものは製造を中止しているため増設することはできなくなり、「メーカの独自仕様を用いると拡張性に支障をきたすという、典型的な実例を示すという結果となっている。」と表明し、標準技術を採用しなかった場合の問題点を指摘している。また、高価なリアルタイム OS よりも、高速 CPU で稼動する汎用 OS を使用することも提案されている。これは計測分野において標準的な計算機の技術が適用可能な時代になってきたことを示している。

システム基盤の選択には必要な時間精度が大きく影響する。必要な時間精度が 10ms 以上なら標準 UNIX を、10ms 以下 100 μ s 以上ならリアルタイム OS を、それ以下ならハードウエア・ロジックを第一候補として考えるのが適当であろう。

3.2.2 拡張性と柔軟性により実用システムを目指す

2000 チャネルの ADC からのデータをすべて表示するシステムを考えると、ADC を接続する計算機や表示モニターとなる計算機の数はかなり多くなると考えられる。このような大規模なシステムを作成するときに使われるソフトウエア・エンジニアリングの手法として、プロトタイピング[17]を採用した。プロトタイピングとは、最終目標であるシステムの骨格部分を作成し、その骨格を基本として全システムを構築することが可能であると判断されて

から、残りを作成する、あるいは作成したプロトタイプを廃棄して全体を再構築するという方法である。

プロトタイプ・システムは少数チャネルのデータを対象とするので、これを実用システムへと発展させるには、システムに拡張性があり、拡張することにより性能などの問題が生じない方式であることが必要である。

拡張性は分散処理方式により達成することが普通であるが、分散した計算機間の通信が必要となるため、通信量の問題が発生する。この問題を克服して拡張性が保証できるかどうかが 3.4 節のテーマである。

次にどのような柔軟性が必要かを検討する。

LHD での観測対象は、超伝導コイルに関連する部分だけでも、温度、圧力、流量、磁場、歪み、液面高、電圧など、多種にわたる。これらの測定対象に対する表示プログラムはそれぞれ異なるものと予想される。また運用に入り、測定データの組み合わせに対して二次的に計算した値を表示したり、より複雑な組み合わせデータを表示することが予想される。これはプロトタイプ・システムを発展させる場合に、同種のデータを多く配置するという量的な拡張だけではなく、異なる表示プログラムを容易に導入、置換でき、柔軟にシステム構築できるようにする必要性を意味する。

柔軟性についてより具体的に説明する。小規模の実験では、測定と表示が専用装置でおこなわれる図 3.1 のような場合が多い。しかし対象とするデータが多くなりデータ源が複数になると、次のような要求が発生する。

- 単純に、測定--表示の組が追加される。
- 同一データを複数の表示方法で表示する。
- 複数のデータを組み合わせたデータを表示する。
- データを変換して表示する。

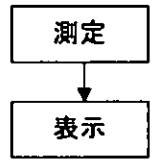


図 3.1

これらの場合を加えたものが図 3.2 である。

ここで導入した、データを複製して配る、複数のデータを統合する、データを変換するという機能、分配・統合・変換の機能をさらに組み合わせる要求が発生することも考えられる。LHD での実際の計測にありそうな組み合わせの例を図 3.3 に示す。

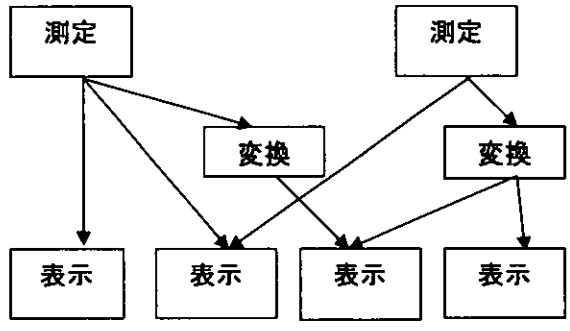


図 3.2 複雑なシステム

液体ヘリウムの温度を超伝導コイルの入口と出口で測定し、流量の測定もおこなう。入口から入った液体ヘリウムが出口に達するまで約 30 分かかるので、温度変化を計測するには、30分前の入口温度が必要となる。複数データに対して変換をおこなう場合、必要なデータが同一データ・パケットに含まれていない場合は、変換の前に統合が必要であることは言うまでもない。実際には図 3.3 よりももっと複雑な組み合わせが必要となることも予想される。

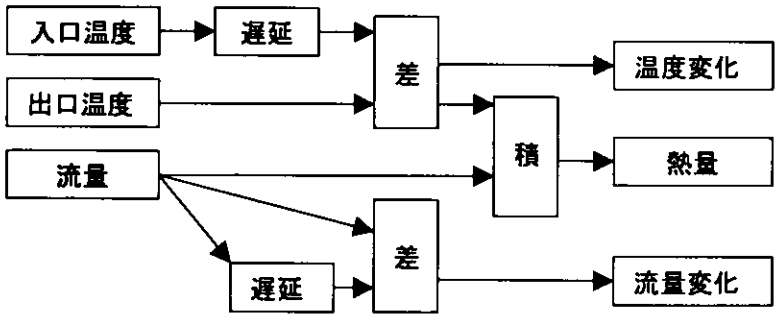


図 3.3 表示システムの例

このように多様なデータ処理の組み合わせが要求されることが予想されるので、柔軟性は非常に重要であり、本研究で作成するプロトタイプ・システムは、実用システムへの発展のために、量的な拡張性と質的な柔軟性をもたなければならない。

3.2.3 ソフトウエア技法を適用する

大規模システムのソフトウエアの開発であるからソフトウエア技法を採用し、明確な設計をおこなう必要がある。一般的に知られている技法を適用するだけでなく、過去の成功事例におけるソフトウエア構築方法に習うことも有効である。LHD 表示システムの目的と、3.3.1 節の分析によりリアルタイム分散処理になるということを考え、次に述べる理由により、著者が開発した SPMS (Statistical Package for Medical Science)[18]と REWMOL (Real World MOdeling Language)[19]の開発で採用したソフトウエア技法を採用することが有効であると考えた。

LHD 表示システムにおいて多数のデータを自由自在に組み合わせて処理をおこなうということは、SPMS の目的と同じである。SPMS はバッチ処理方式の統計パッケージで、既存の統計パッケージが統計手法中心であったのに対し、統計手法を適用するまでのデータの変換、統合などの処理を柔軟におこなうために開発された。データ処理を柔軟におこなうために、データ管理とデータ処理を分離し、データ処理部分を自由に付け加えられるようにしてある。LHD 表示システムの構造を SPMS の構造と対応させて、水

平分割および垂直分割により設計すれば、柔軟に表示方法を付け加えられるシステムになることが期待される。

水平分割とは、機能を同じレベルの複数の機能に分け、それぞれの機能間の関連をなくすことにより、それぞれの機能を独立して開発できるようにするソフトウエア技法である。垂直分割とは、機能を上位と下位に分け、それぞれを独立な機能として設計することである。下位の機能は通常ライブラリという形で提供される。

LHD 表示システムはリアルタイム・システムであり、同時にたくさんのプロセスがデータを通信しながら協調して動作すると考えられる。この状況は REWMOL が記述対象とした実世界の状況と同じである。プロセス間の相互作用は、セマフォーなど、OS のプロセス間通信機能を使って実行するが、プロセスが多数になるとデッドロックになる危険性が増す。プロセス間通信を使わずに、相手プロセスを見るだけで正しい状態を知ることができるならば、デッドロックになる危険性がないので、いつでも自由に実行できる。このように可能なかぎり「相手を見る」だけでよい相互作用を使ってプロセス間の協調をおこなうのがよい。REWMOL の応用例として、この考えに基づいて3桁×2桁の虫食い算を解いた。この問題を解く過程において、100 を超えるプロセスが生成されて相互作用をおこなった[20]。

LHD 表示システムでも、可能な限り「相手を見る」だけで相互作用をするように設計することにより、多数のプロセスを安全に実行できる。

このように、通常はプロセス間通信機能を使う必要がある場合に、問題固有の性質を利用してプロセス間通信機能を使わずにすませるソフトウエア技法には、まだ名称がついていない。

3.2.4 リアルタイム間欠処理

全データをリアルタイム処理することがデータ転送速度上不可能な場合に、平均値や変化の有無などの要約データを計算し、10Hz のリアルタイム表示の対象とすることが望ましい。このためには高速に連続してデータを取得できることが必要である。

LHD の超伝導コイルによる磁場や、高出力発信器の影響を避けるため、ADC は本体から離れたところに設置され、センサーと ADC の間に絶縁アンプを使用する[14,15,16]。この場合測定精度は絶縁アンプの精度で決まり、8 ビット程度の分解能となる。これは、絶縁アンプは通常のリニアアンプに比べて S/N 比が劣るからである、。測定精度を改善する一つの方法は、サンプリング速度を早くし、平均値を取る方法である。これはもっとも典型的な要約データの一種である。

このほかの要約データとしては、一定レベルを超える場合や、一定レベル以上の変化を検出した場合にイベント・データとして通知することなどが考えられる。

要約データを表示し、なんらかのイベントの発生を検出したら、一定期間過去に溯って必要な部分だけ高速データを保存することができれば、大幅に保存データ量を少なくできる。保存対象時間が短かいならば、保存した高速データを即時に送信することも可能となり、イベントが発生したときに、その前後のデータをリアルタイムに見ることができる。

このような処理方式は、リアルタイム処理とバッチ処理を融合した方式であり、リアルタイム間欠処理と呼ぶことにする。

リアルタイム間欠処理をおこなうには AD 変換装置に次の機能が必要となる。全データを読み出すことができないサンプリング速度であっても、イベント検出に必要なデータだけを読み出すことができ、イベント発生前後のデータを蓄積できる大容量のバッファがあること。バッファ容量が小さい場合にイベント検出時のデータ取得を保証するためには、イベント検出時に AD 変換を停止し、データ取得後 AD 変換を再開する方式もある。この場合はイベントの検出をおこなわない不感時間が発生する。

これらのことを考えると、ADC でのデータ取得では少なくとも 1KHz のサンプリング速度の全データを 1 データずつ処理するリアルタイム処理、高速の多数データを一括処理するバッチ処理、さらに 100KHz でのリアルタイム間欠処理に対応できるようにすべきである。

3.3 システム分析

本節では、まずシステム構築のための分析として、システムを構成する計算機の台数を推定する。つぎにデータ通信に必要な性能分析を詳細におこなうため、通信方法ごとに必要となる通信速度を計算する。つぎに、レスポンス・タイムの構成成分を分析し、個々の処理における要求性能を明確化する。最後に計測システムに必須の要素である AD 変換装置と表示プログラムに対する要求内容をまとめる。

3.3.1 システム規模の推定と基本性能の評価

最終的な目標である 2000 点のデータをレスポンス・タイム 100ms で表示するためのシステムの規模を推定した。

使用する計算機をサン・マイクロシステムズ社の SPARCstation 2[21]と仮定した。これは汎用 UNIX が稼動する計算機で、実際にシステム開発に使用した計算機である。表示システムが実用化されてシステムを運用するときには、より高性能のモデルを使用できると期待できるので、ひかえめの性能評価となる。

基本要素は ADC のデータを取り込む収集用計算機とデータを表示する表示用計算機、およびデータを計算機間で送信するネットワークである。

3.3.1.1. 計算機台数の推定

まず表示用計算機の必要台数を推定した。

表示方法として最新データのみを表示するなら多数のデータを表示できるが、経過を見るためグラフとして表示するなら、1 画面に同時に表示できるデータは 16 から 32 程度であろう。画面切り替えを許すと多数のグラフを表示できるが、切り替えに時間を要するので、レスポンス・タイムを 100ms とする意味が失われてしまう。

表示用計算機があまり多数になると設置上の問題や、運転要員数の問題が発生するので、表示用計算機 1 台で表示するデータ数を 64 とし、表示用計算機は 32 台必要であると推定する。単純な推定では均質な表示方法を想定したが、高速なレスポンス・タイムを必要としないデータがあれば切り替え方式で良いので必要台数は減少する。一方、同一データに対して表示方法が異なる表示をおこなうと必要台数は増加する。また高速データに対するイベント発生時の表示も必要であろう。このように台数の増減要因があるので 32 台を目安とした。

次に収集用計算機の必要台数を推定した。

AD 変換素子を並列に並べてチャネル数を大きくすることはできるが、データ取得時には順次処理になってしまうので、チャネル数を大きくすると取得可能なサンプリング速度は低下する。サンプリング速度を低速データでは 1KHz[1]、高速データでは 100KHz を目標としてデータ量を計算すると、1KHz ・ 512 チャネルでは 1MB/s、100KHz ・ 32 チャネルでは 640KB/s となる。この場合は、1MB/s 以上高速にデータを取得できれば連続的に全データを取得できることになる。このデータ取得速度は、SPARCstation 2 の入出力バスである SBus[22]が 32 ビット幅で 20MHz であることから実現可能と思われる。計算機 1 台に最大 512 チャネルの ADC を接続するとして、低速データ 2000 チャネルと高速データ 32 チャネルを接続するには収集用計算機を 5 台必要であるとした。

合計すると少なくとも 37 台の計算機が必要となり、収集用と表示用に機能分散し、それぞれの用途に複数の計算機を使用する形となるため、必然的に分散システムとなる。

3.3.1.2. 通信速度の検討

次に収集用計算機から表示用計算機へのデータ通信速度を検討した。

3.1 節では 2000 チャネル 10Hz なのでデータ通信速度は 40KB/s と見積もったが、これは 1 台の収集用計算機に 2000 チャネル全部が接続され、1 台の表示用計算機で全データを表示するという、現実にはない場合である。低速データだけ考え、4 台の収集用計算機ごとにグループ化し、それぞれが 512 チャネルのデータを 10Hz で計測し、取得したデータをそれぞれ 8 台の表示用計算機に送ると、328KB/s (=512 チャネル×2 バイト×10Hz×4 台×8 台)となる。グループ化せず、32 台のどの表示用計算機でもすべてのデータを受けるようにすると、システムとしての柔軟性は向上するが必要な通信速度は 1311KB/s (=512 チャネル×2 バイト×10Hz×4 台×32 台)となり、10Mbps のイーサネットの物理的限界を超える。この他にも、物理量に変換した値を送ると、データ値の表現に4バイト必要となるので通信速度は 2 倍になるなど、通信速度に対する要求は大きくなる一方である。

計算機間の通信方式として最も標準的な技術である 10Mbps のイーサネットを使用したいが、このように 1 対 1 の通信方式で通信をおこなうと、すべての通信量の総和となる通信速度が必要となり、ネットワークの性能上限によって必要な通信速度が得られなくなる恐れがある。

---

[1] ここで低速データを1KHz としたのは、3.2.4 節で述べた平均化処理をおこなうことを想定しているからであり、平均化処理後は 10Hz ないし 100Hz となる。

1 対 1 通信には、通信速度の問題以外に柔軟性に乏しいという問題もある。表示用計算機を増設すると、新たに増設された計算機への通信をおこなうように設定しなければならず、通信量の増大、収集用計算機の負荷の増大を招き、増設前の正常動作を保証できない。

ユニキャスト通信方式と違い、次節で説明するマルチキャスト通信方式を使用すると、受信する計算機が何台でも通信量は変わらない。今の場合、41KB/s (=512 チャネル×2 バイト×10Hz×4 台)の通信量で、すべての表示用計算機が全データを受信でき、表示用計算機を増設しても通信量は変わらず、収集用計算機に対して何の操作も必要ない。しかし次節で説明する欠点、データ転送が保証されないという問題がある。このため、良いネットワーク環境、すわなち NFS などによるバースト的な通信がなく、リアルタイム・データの通信だけで占有できるネットワークで使用することに限定すれば、信頼性が十分である可能性があるので、どの程度の通信量と通信品質が得られるかを調べる必要があった。マルチキャスト通信方式を使った通信量の実験結果は次節で示す。実験結果によると、リアルタイム通信以外の通信はなく、リアルタイム・スケジューリング・クラスで実行するという良い条件では、800KB/s の通信量でもデータ欠損を生じなかった。これは必要な通信量 41KB/s の約 20 倍であり、マルチキャスト通信方式の採用は十分実用になると判断された。

3.3.1.3. レスポンス・タイムの検討

次に 100ms のレスポンス・タイムが達成できるかどうか検討した。

レスポンス・タイムには、図 3.4 のように複数の要素が関係する。(1)AD 変換の時間、(2)ADC から収集用計算機内に取り込む時間、(3)収集用計算機から表示用計算機に送信する時間、(4)表示用計算機で表示するための時間の総和がレスポンス・タイムとなる。

図 3.4 レスポンス・タイムの構成要素

プログラム間のデータ受け渡しの方法をポーリング方式とすると、収集用計算機では AD 変換終了との同期をとる(2)の処理のために、最悪の場合 UNIX の基本時間単位である 10ms を必要とし、表示用計算機ではデータ受信との同期をとるために最悪 10ms を必要とする。通信に必要な時間(3)は次節で説明するデータ転送時間推定実験により約 5ms と見れば良いことが分かった。これで ADC から 512 チャネルのデータを取り出

す時間(1)と、データを画面に表示する処理(4)に要する時間の和が75ms以内ならば、レスポンス・タイムとして100msを達成できる。これらがAD変換装置の設計とグラフ表示プログラムの設計に課せられる要求となった。

3.3.2 AD変換装置の機能と性能に対する要求

LHD実験で要求される機能と開発方針により、AD変換装置には多チャンネルでリアルタイム処理とバッチ処理の両方に対処できることが要請された。

リアルタイム処理では、AD変換されたデータがすぐにAD変換を停止せずに取り出せることが重要であり、また取り出し速度がAD変換速度より高速である必要がある。このデータ取り出しが高速であれば、リアルタイムに取り出して主記憶に保存し、必要なときに処理することでバッチ処理にも同時に対応できる。そうすると処理するデータ量が増えたときには主記憶を増やせば良く、TRIAM-1M[9]の場合のように装置メモリの不足に悩まされることはない。

一方、従来型のバッチ処理方式プログラムに対応するためには、装置内のメモリにデータを保存する必要がある。

LHDでは、プラズマを発生・加熱するため大電力・高電圧のRF、NBIを利用するので、高精度の計測をおこなうためにはできるだけ早くセンサーの信号を増幅する必要がある[14,15]。このためセンサーの近くにアンプを設置するが、AD変換装置の設置場所とは電気的環境が違うため絶縁アンプを必要とする。絶縁アンプは精度が悪いため、高速サンプリングを行い、移動平均により精度を上げる方法を採ろうとすると、リアルタイム性能を強化しなければならない。またクエンチ検出のためのAE測定では100KHzでデータ処理したいという希望もある。

これらの点を考慮して、AD変換装置に対しては下記の機能と性能を目標とした。

但し、チャネル数が多いと装置内部での遅延が大きくなり、実装上の問題が起きるので、過大な要求を避けている。

- AD変換速度は100KHz以上。
- 最大128チャネル以上でできるだけ多数。
- 16チャネル100KHzのデータをAD変換しながら同時に取り出せること。
- 128チャネル10KHzのデータをAD変換しながら同時に取り出せること。
- AD変換後、可能な限り直ちに(最大100μs以内に)取り出せること。
- 外部クロックも使用できること。

- データ取り出しが間に合わないこと(オーバーフロー)を検出できること。
- 16 チャネル 100KHz のデータを 10 秒間保存できること。
- AD 変換の開始・停止ができること。

3.3.3 表示プログラム作成に対する要求

ここでいう要求は、ひとつの表示プログラム作成に対する要求ではなく、LHD 運用システムにおける表示プログラム作成全体に対する要求である。この要求を満たす表示プログラムを作成することは、運用システムにおける表示プログラムの作成を容易とし、システムの柔軟性を高めることとなる。

リアルタイム表示という条件のもとでの要求を次のようにまとめ、この機能を基本とし多様な表示方式の基本型となるよう考慮した。

- 3.3.1 節の通信および同期に要する時間の見積もりにより、合計 100ms のレスポンス・タイムを実現するため、データ取得後 50ms 以内に表示すること。
- 物理量で表示すること。ただし物理量への変換に要する時間は、計算方法を規定できないので、50ms の時間に含めない。
- 収集用計算機は複数なので、複数のデータ源からのデータを表示できること。
- 制御用計算機の持つシステムの状態などを受け取ることができること。
- 他の表示用計算機の結果を受け取ることができること。
- サンプリング速度の異なるデータ源に対処できること。
- リアルタイムに表示できない高速データを、間欠的にあるいは表示要求があったとき表示するなど、多様な表示方法に対して考慮していること。
- 変化を見るため、データを時系列として扱えること。

3.4 マルチキャスト通信方式の評価

通信量の問題を克服するため、リアルタイム処理での通信方式としてマルチキャスト通信方式を利用することになったので、利用可能性を実験により評価する。

3.4.1 マルチキャスト通信方式の利点と欠点

3.4.1.1. マルチキャスト通信方式の利点

すべての表示用計算機に同一データを分配すれば、各表示用計算機がそれぞれ自由に対象とするデータを選択して表示できる。そうすると、表示対象となるデータが多くても、必要に応じて表示用計算機を増設すれば良いので拡張性があることになる。しかしデータを分配するときにユニキャスト通信方式では同一データを全受信計算機に逐次的に送信することになり、分配する計算機の数だけ時間がかかり、通信量も増大する。

バス型ネットワークであるイーサネットにはマルチキャストという同報通信のための通信方式があり、この方式で送信すると同一データをイーサネット上のどの計算機でも受信できる。図 3.5 にユニキャスト通信方式とマルチキャスト通信方式の違いを図示する。

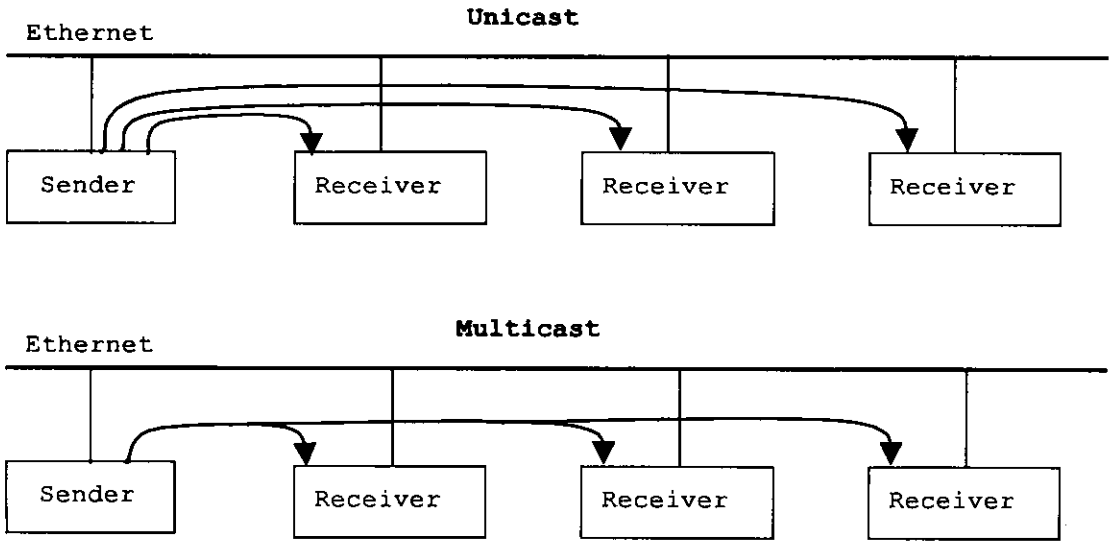


図 3.5 ユニキャストとマルチキャスト

図 3.5 から分かるように、ユニキャスト通信方式で送信する場合は、受信計算機ごとに順番に送信しなければならないのに対して、マルチキャスト通信方式で送信する場合は、1回の送信で全受信計算機にデータを分配できる。すなわち、受信計算機を増設し

ても、まったく送信負荷の増加も受信遅延時間の増加もなくデータ分配をおこなえるようになる。受信計算機の増設時に送信側はまったく何もしなくてよく、送信側と受信側がまったく独立していることはシステムの柔軟性を高めることとなる。

3.4.1.2. マルチキャスト機能の実現

SunOSのバージョン4.xでは、OSとして、マルチキャスト機能をサポートしていない。しかし、物理装置であるイーサネットにはマルチキャストの機能はあるので、イーサネットを直接使用するデバイス・ドライバ(正確には streams)を作成した。マルチキャストの機能はその後 IP の機能として標準化され、SunOS バージョン 5 でサポートされるようになった。本研究で作成したプログラムの名前に mcs_という文字が付いているプログラムは SunOS4.x 用のイーサネットのマルチキャスト機能を使用するプログラムであり、mcip_という文字が付いているプログラムは SunOS5.x 用の IP マルチキャスト機能を使用するプログラムである。

なお、イーサネットのマルチキャスト機能では同一セグメント内の計算機にしかデータを送信できないが、IP マルチキャスト(Internet Protocol Multicasting)に対応しているルータを使用すれば、セグメントを越えてマルチキャストでデータを送信できる。

3.4.1.3. マルチキャスト通信方式の欠点

通常1対1通信に用いられる TCP と異なり、マルチキャスト通信方式では受信確認がおこなわれず、データ通達は保証されない。マルチキャストを使う方式でもデータ喪失の場合に再送要求をして信頼性を上げる reliable multicast と呼ばれる方法もあるが、リアルタイム表示での利用では、一定時間間隔に送信する場合に、転送レートをどの程度上げられるかを問題としており、高速時には喪失パケットが再送される前に次のデータが送られてくるため、再送を待っていてはリアルタイムの意味がなくなる。周期的なリアルタイム送信がおこなわれているときに再送が発生すると、通信量や通信タイミングがランダムに変動することになり、全体が不安定になりかねない。またデータの再送は、送受信計算機の負荷を増大させるので、過負荷の状態で再送要求が起きるとさらに負荷を増大させることになり、ますます機能障害を悪化させる原因となる。このため、リアルタイム表示にマルチキャストを使用するときは、正しく通信量を把握し、データ喪失が起きないように注意するとともに、万一データ喪失が起きた場合に、データ喪失を検出し、1パケットの喪失の場合は補間によりデータを補うなど、動作を継続するために必要な措置を取る。連続して喪失した場合は、性能などの問題があるので通知し、場合によっては停止するなどの措置を取る。

3.4.2 マルチキャストでの通信量を求める実験

3.4.2.1. ネットワーク性能試験での共通事項

まず実験条件として、通信レート、データ・サイズ、使用した計算機およびネットワークについて説明する。

本論文ではリアルタイム・データは 10Hz で送信されることになっている。しかし 3.3.1.1 節で収集用計算機は 5 台と見積もったので、それぞれからリアルタイム・データを受信するなら 50Hz のデータを受信することにほぼ等しくなる。そこで性能評価の実験としては 100Hz を超える高速データについておこない、通信量、通信品質の目安をえることとした。UNIX のスケジューリング時間単位である 10ms 以下の非常に短い時間を対象とするため、プログラムはリアルタイム・スケジューリング・クラスで実行することを基本とした。同じプログラムをタイム・シェアリング・スケジューリング・クラスで実行すると、オペレーティング・システムのタスクの方が優先されるため、テスト・プログラムの実行時刻のゆらぎが大きくなり、信頼できる値をえられない。

実験中に実験データをファイルに出力すると、この出力に必要な CPU の負荷やデバイスからの割り込みなど、時間のゆらぎを大きくする要因となるため、実験中のデータはメモリ上に貯え、実験終了後にファイルに出力した。また、この実験データの保存に必要なメモリ量が実験により大きく変わると、スワップが発生するなどオペレーティング・システムのメモリ管理による影響があると予想されるため、データはメモリ上に度数分布表として貯え、少ないメモリで済むようにした。

1－4バイト	パケット長		クラス	ID
5－8バイト	秒単位の時刻			
9－12バイト	秒以下 μs 単位の時刻			
13－16バイト	予備	予備	予備	連番
17－20バイト	CH1		CH2	
21－24バイト	CH3		CH4	
1037－1040バイト	CH4		CH512	

図 3.6 512チャネルのデータ・パケットの構造

この実験を含め通信に関する実験では、プロトタイプ・システムで実際に使用するデータを用いて実験するほうがより信頼性のある結果を得られるため、送信データとしてすべて図 3.6 に示す、512 チャネルの ADC データと同じ大きさのデータを使用している。データはパケット・ヘッダを含めて 1040 バイトである。データ・パケットの各フィールドの意味については、4.1 節で説明する。

通信量について議論するときは、利用者のデータだけでなく、通信制御のために付加される情報も加えた物理レベルのデータ長[25]を使わなければならない。今の場合、1040 バイトのデータを送信するための物理的なデータ長は、データ 1040 バイト＋UDP ヘッダ 8 バイト＋IP ヘッダ 24 バイト＋ether ヘッダ・トレイラ 26 バイトの合計 1098 バイトである。512 チャネルは 2 のべき乗のチャネル数としては、イーサネットの最大長 1526 バイトに収まる最大のチャネル数である。図 3.6 に示すデータ・パケットがユーザの扱うデータで、このパケットの前後に通信のためのヘッダやトレイラが付いて、物理レベルの通信が扱うパケットとなる。

なお、イーサネットのマルチキャストではなく、IP マルチキャストを使用する場合、データ・パケットをもっと長くできるが、長い IP パケットは複数のイーサネット・パケットに分割して送信される。アプリケーションでの 1 回の送信と、イーサネット上での1回の送信との対応が付かなくなり、動作状態に不確定要素を持ち込むことになる。このため実験ではイーサネットの 1 パケットに収まるデータ長を用いており、実際の運用にあたっても、同様に長さ制限をすることが望ましい。

実験は 10base5のイーサネット上に SPARCstation を接続しておこなった。実験に使用した計算機とネットワークの緒元を表 3.1 と表 3.2 に示す。

表 3.1 使用した計算機

計算機機種	SPARCstation 2	SPARCstation/IPX
CPU クロック周波数	40MHz	40MHz
CPU 処理速度	28.5MIPS	28.5MIPS
主記憶容量	32MB	32MB

表 3.2 使用したネットワーク

イーサネットの規格	10Base5
ケーブル長	20m

次に 100Hz 以上で送信するために必要な特別な処理方式について説明する。100 パケット/秒を超える頻度でデータ・パケットを送信するには 10ms より短いタイミングが必要なため、標準 UNIX が提供するタイマー制御では送信タイミングを作れない。そこで次のパケット送信時間になるまで、ループを実行して待つ、いわゆる busy loop でタイミングを作成しなければならない。busy loop にするとCPU 負荷は100%となる。送信をCPU 負荷 100%のリアルタイム・スケジューリング・クラスで実行し、同時に受信しようとすると、送信だけしかおこなわれずパケットを受信できなくなるため、この場合はタイム・シェアリング・スケジューリング・クラスで送信することとした。

送信は次のコマンドで実行する。

```
mcip_busy_snd_rnd -c512 -p1024 -a20000 -b20000 -n10000 -u200 -o10000 -h100
```

コマンド mcip_busy_snd_rnd の名前は、IP マルチキャストを使い、busy loop によりタイミングを作成し、送信時間間隔はランダムに生成するということを意味する。UNIX では、オプションが少ない場合にこのように実行時に必要なオプションをコマンドの後ろにコマンド引数の形式で与えるのが標準であり、マイナス記号につづく1文字でオプションの種類をこの文字に続いてオプションの値を記述することになっている。指定内容が多い場合はファイルで指定し、コマンド引数ではそのファイル名を指定する。

オプションの -c512 は 512 チャネルのデータを送ることを意味し、ヘッダを含めデータ・パケット長は 1040 バイトとなる。オプション -p1024 は、このデータ・パケットを識別するマルチキャストのポート番号である。オプション -a20000 -b20000 は、ランダムに生成する時間間隔として最小 20000 μ 秒最大 20000 μ 秒を指定しているので、この場合は等時間間隔に毎秒 50 パケット送信する。オプション -n10000 は 10000 個のデータ・パケットを送信することを表す。オプション -u200 -o10000 -h100 は、10000 μ 秒から始め 200 μ 秒単位に 100 段階の度数分布表として測定データを蓄積することを表している。以下このようなコマンドに付随した条件はそれぞれによって値は変わるが同じ意味である。

プログラムは長くなるので、骨格の部分だけをその機能の説明と共に下記に示す。

```
v_next_interval = (v_higher_interval - v_lower_interval) * drand48() +
v_lower_interval;
// 上下限内の一様乱数により時間間隔を設定する
x_usec_add(&v_next_time, v_next_interval);
// 次の送信時刻を求める
x_usec_wait_until(&v_next_time);
// 次の送信時刻まで、busy loop で待つ
```

```
x_packet_send(c_lost_command_data, g_sequence);
// データパケットを送信する
v_send_time = *x_lost_sendtime_ptr(g_packet);
// 送信時刻を得る
v_sec = v_send_time.tv_sec - v_prev_send_time.tv_sec;
v_usec = v_sec * 1000000 + v_send_time.tv_usec - v_prev_send_time.tv_usec;
// 送信時間間隔を計算する
v_class = x_hist(v_usec);
// 度数分布表に加える
v_prev_send_time = v_send_time;
// 送信時刻を更新する
g_sequence ++;
// データ喪失検査のためのシーケンス番号を更新する
```

x_packet_send 関数内では、

```
x_lost_set_sequence(g_packet, v_value);
// シーケンス番号をデータとして記入する
gettimeofday(x_lost_sendtime_ptr(g_packet), NULL);
// 現時刻を送信時刻として記入する
x_mcip_write(g_mcip, (char *) &g_packet, g_packet_size);
// データパケットを送信する
```

のようにシーケンス番号と時刻を記入して送信している。

3.4.2.2. ネットワークの通信容量推定の実験

通信容量とは毎秒何パケットのデータまで受信できるかということで、どの程度の高速現象までネットワークによる分散処理で対処可能かを調べるために実施した。

通信容量を求める実験は図 3.7のように3台の計算機で実施した。計算機として SPARCstation IPX と SPARCstation 2 を 2 台使用した。SPARCstation IPX と SPARCstation 2 は同じ性能である。

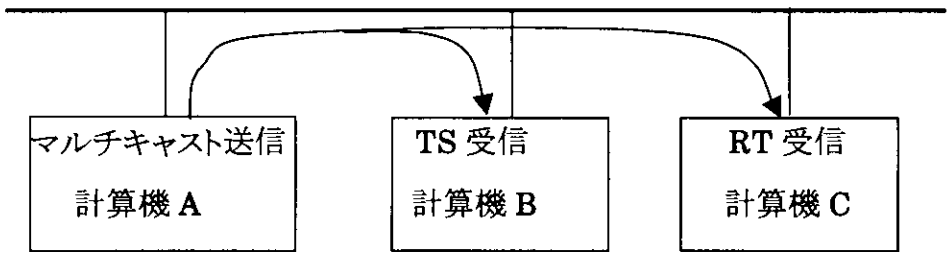


図 3.7 通信容量試験

まずスケジューリング・クラスによる違いを調べるための実験をおこなった。この実験では、計算機 A から 512 チャネルの ADC データに相当する 1040 バイトのデータ・パケットをマルチキャストで送信し、計算機 B と計算機 C で同時に受信した。受信プログラムは同一だが、計算機 B ではタイム・シェアリング・スケジューリング・クラス、計算機 C ではリアルタイム・スケジューリング・クラスで実行し、データの欠損と、データの受信状態を調べた。

この実験から、後で実験結果で示すように、リアルタイム・スケジューリング・クラスではデータ欠損を発生しないのに対し、タイム・シェアリング・スケジューリング・クラスではデータ欠損が発生することが分かった。データ欠損が発生しては通信容量限度を調べる実験には適さないので、通信容量限度を調べる実験ではリアルタイム・スケジューリング・クラスのみを用いた。

送信のテスト・プログラムは、マルチキャストの試験で共通に使用する **mcip_busy_snd_rnd** である。

受信状態の安定性を受信時刻のゆらぎで評価することとした。送受信時刻差を計算するには、両計算機の時刻があっていなければならない。異なる計算機の時刻を合わせようとすると、時刻同期処理による時刻変動が大きいため、異なる計算機の時刻を 10μs の精度まで合わせることはできない。そこで、受信したデータ・パケット間の時間間隔である受信インターバルのゆらぎを調べることにより受信状態の安定性を評価することとした。受信インターバルの計測は次のようにして測定した。

受信は次のコマンドで実行する。

```
mcip_interval -p1024 -u125 -o0 -h30
```

リアルタイム・スケジューリング・クラスで実行する場合は

```
priocntl -e -c RT mcip_interval -p1024 -u125 -o0 -h30
```

となる。**priocntl -e -c RT** は、その後のコマンドをリアルタイム・スケジューリング・クラスで実行することを表す。

コマンド **mcip_interval** は、パケットを読み込んではシーケンスが連続か不連続かによりパケットの欠損の有無を調べ、欠損がなければ受信インターバルを記録し、欠損があれば欠損数を積算する。この処理の骨格は次のとおりである。

```
v_length = x_mcip_read(g_mcip, (char *) &v_packet, sizeof(v_packet));
// パケットを読み込む。
v_sequence = x_lost_sequnce(v_packet);
// パケットのシーケンス番号を取り出す。
```

```
gettimeofday(&v_receive, NULL);
// 受信時刻を得る。
if (v_sequence == (g_sequense_s + 1)) {
    // シーケンス番号が連続しているのでデータ欠損はない。
    v_sec = v_receive.tv_sec - g_start.tv_sec;
    v_usec = v_sec * 1000000 + v_receive.tv_usec - g_start.tv_usec;
    // 前の受信時刻との差を計算する。
    x_hist(v_usec);
    // 受信時間インターバルを度数分布表に加える。
} else {
    g_lost_1st += (v_sequence - g_sequense_s - 1);
    // 欠損パケット数を積算する。
    g_lost_times++;
    // 欠損回数を積算する。
    printf("L  lost 1st packet %d¥n", g_lost_1st);
    // 欠損パケット数を出力する。
}
g_start = v_receive;
// 現受信時刻を前受信時刻として設定する。
g_sequense_s = v_sequence;
// 現シーケンス番号を前シーケンス番号として設定する。
```

スケジューリング・クラスの違いが受信性能に影響すると考え、スケジューリング・クラスの違う2台の計算機での受信状態を測定し比較した。

送信レートをいろいろ変え、10000 パケットを送信して、受信状態を調べた結果、リアルタイム・スケジューリング・クラスではデータ欠損をまったく生じなかった。一方タイム・シェアリング・スケジューリング・クラスではデータ欠損が発生した。この欠損の状況を調べると、欠損する原因が推定できると思われる。データ欠損状況を表3.3に示す。

表 3.3 は、データ欠損を生じたときの欠損パケット数を、毎回のデータ欠損について記録したものである。パケットの送信レートを表の各行のように定め、合計10000パケットを送信する。パケットの欠損が発生した回数が表の列である。1回の欠損で連続して欠損したパケット数が表の値である。欠損回数はランダムであり、1回の欠損で連続したパケットを欠損しているところに特徴がある。

表 3.3 タイム・シェアリング・スケジューリング・クラスで受信したときのデータ欠損ごとの欠損パケット数

	欠損発生番号									
送信パケット数/秒	1	2	3	4	5	6	7	8	9	10
800	29									
500	25									
400	13									
250	9	8								
250	8	4	8	7	8	5	8	15	16	8
	3	8	2	8	3	8	4	8	9	

欠損パケット数を送信パケットのレートで割ると、欠損パケットの占める時間がわかる、表 3.3 で計算すると、最長時間は 250 パケット/秒で送信し、16 パケットを欠損した場合で、時間は 64ms になる。タイムシェアリング・スケジューリング・クラスでは、この程度の時間処理をおこなえないことがあるということである。

リアルタイム・スケジューリング・クラスを使えばデータ欠損を生じないことが分かったので、リアルタイム・スケジューリング・クラスではどの程度の速度まで使えるかを調べるため、受信パケット・インターバルの分布を 10 μ s 秒単位で調べた。結果を表 3.4に 20 μ s 区間の度数分布表としてまとめた。この実験では 100000 パケット送信したが、この実験でもデータ欠損は発生しなかった。合計が 99999 となっているのは、計測しているのは送受信パケット間の時間間隔であり、間隔はパケット数より 1 少なくなるからである。送信インターバルの広がりそのものは転送レートによらず約 3ms に達しており、転送時間の広がりを 3ms 程度見込んでおく必要がわかる。

転送レートが大きくなると受信による負荷が大きくなることが、受信インターバルの広がりが大きくなることに現れると考えた。ゆらぎを受信インターバルの標準偏差で評価することとし、このゆらぎと、イーサネットの理論最大通信量 10Mbps に対する通信量の比率をネットワーク負荷と考え、このふたつを送信インターバルの関数として図 3.8 に示す。

表 3.4 RT スケジューリング・クラスでのインターバルの分布

インターバル μs	送信レート									
	1000p/s		800p/s		667p/s		500p/s		400p/s	
	送信	受信	送信	受信	送信	受信	送信	受信	送信	受信
400		151								
440	1	824		15		2				
480	0	120	2	2		0				
520	0	1	0	1		0				
560	0	0	0	0		0				
600	0	0	0	0		0				
640	0	0	1	0		0				
680	3	33	0	1		0				
720	1	684	0	2		0				
760	0	7152	1	29		1				
800	0	6986	0	56		2		1		
840	0	1027	0	14		0		0		
880	0	1885	1	3		0		0		
920	0	3581	0	35		1		0		
960	49991	18365	1	232		2		0		
1000	49994	30638	51	640		1		0		
1040	4	6883	31	1846		6		0		
1080	1	7423	3161	3922		9		0		
1120	0	9377	870	8010		14		0		
1160	0	2705	39409	37710		52		0		
1200	3	424	52384	35042		74		1		
1240	1	492	28	4689	1	117		1		
1280		144	3256	3052	46	256		0		
1320		15	717	3090	58	478		2		
1360		1	40	877	1	2183		1		
1400		1	42	261	4898	5173		2		
1440		5	1	256	100	7398		4		
1480		38	0	100	79893	52083		1		
1520		64	0	28	9999	21005		2		
1560		116	0	17	3381	4989		3		
1600		336	0	16	1516	3366		9		
1640		233	0	16	35	1802		6		
1680		84	1	4	66	444		7		1
1720		43	0	3	2	245		29		0
1760		49	0	1	2	136		85		2
1800		28	0	0	1	74		60		1

	1000p/s		800p/s		667p/s		500p/s		400p/s	
μs	送信	受信	送信	受信	送信	受信	送信	受信	送信	受信
1840		25	0	0		39		483		1
1880		7	0	0		23	1	2474		0
1920		10	1	1		8	59	5506		0
1960		12	0	0		3	41227	42668		1
2000		7	0	1		1	58652	39546		4
2040		9	0	2		1	60	5429		5
2080		7	0	3		1		2189		5
2120		3	0	7		4		1181		12
2160		1	0	2		1		123		24
2200		0	0	2		2		58	1	34
2240		0	1	0		0		56	0	54
2280		2		1		1		26	0	82
2320		0		1		0		10	0	143
2360		0		1		0		8	1	1173
2400		0		0		0		8	0	3766
2440		0		0		0		4	11338	17346
2480		0		0		0		1	73709	55301
2520		1		1		0		1	14949	16016
2560		1		0		0		1	0	3442
2600		3		0		0		3	0	1919
2640		2		0		0		4	0	386
2680		0		0		1		2	0	75
2720		0		0		0		2	0	77
2760		0		0		1		0	0	55
2800		0		0				0	1	26
2840		0		1				0		22
2880		0						1		6
2920		0						0		4
2960		0						0		3
3000		0						0		1
3040		0						0		1
3080		0						0		2
3120		0						0		4
3160		1						0		0
3200								1		2
3240										1
3280										1
3320										0
3360										1
合計	99999	99999	99999	99999	99999	99999	99999	99999	99999	99999

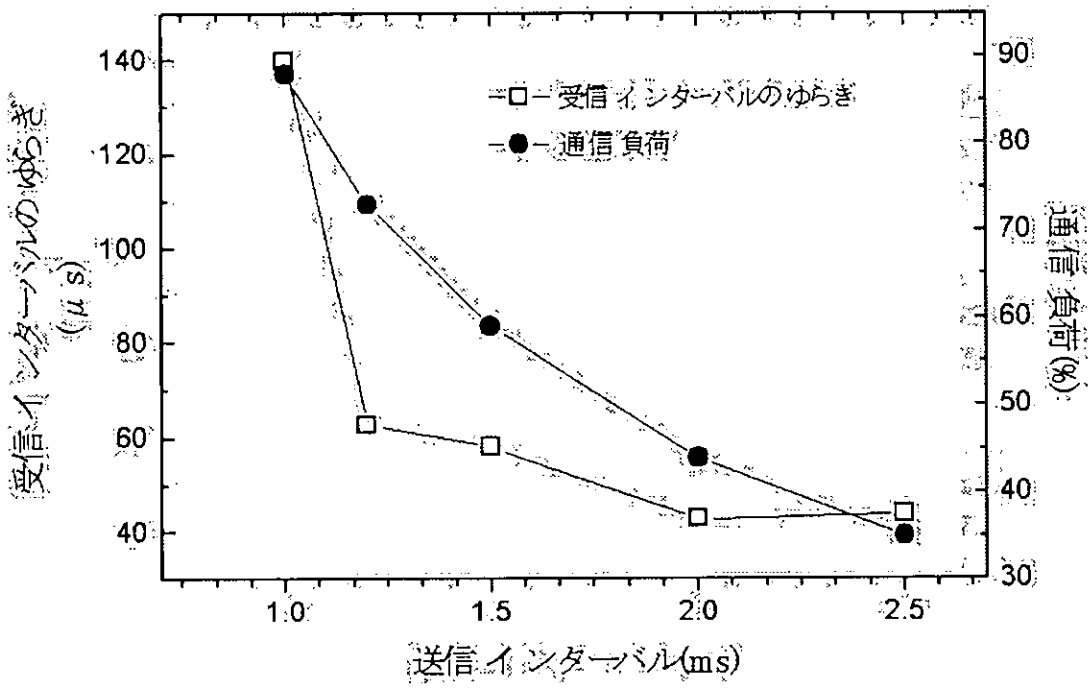


図 3.8 リアルタイム・スケジューリング・クラスでの受信状態

3.4.2.3. 実験結果の分析

データ欠損の原因について分析する。ネットワーク上の伝送エラーのようなランダム現象が原因であるなら、1回のエラーで欠損するパケット数は 1 であるはずであり、表 3.3 のように連続して欠損することはない。また欠損の状況がスケジューリング・クラスに依存していることからも、原因はネットワーク側ではなく、計算機側にあることは明らかである。スケジューリング・クラスによる違いを考えると、タイム・シェアリング・クラスで実行した場合、OS のプロセスが実行中のために待たされている間にデータ・パケットが連続して到着しているからであると考えられる。

次にリアルタイム・スケジューリング・クラスでの受信時間間隔のゆらぎについて分析する。図 3.8 では、送信インターバルが 1.0ms のときに急激にゆらぎが大きくなっており、受信状態が不安定であることを示している。表3．4 の度数分布表では、受信インターバルは多峰性になっており、その程度は送信インターバルが短いほど顕著である。中央の山と隣の山との間隔は、800p/s では 400 μ s なのに対し、1000p/s では 600 μ s と大きく離れ、かつ頻度も非常に大きくなっている。

データ・パケット長が 1098 バイトなので、ネットワーク上での時間にすると 878 μ s となる。データ・パケットの後尾と次のデータ・パケットの先頭との間の平均時間は、800p/s では 372 μ s、1000p/s では 122 μ s となる。送信インターバルが 400p/s のデータから、

ゆらぎが 360 μs を超えるデータは 0.026%、ゆらぎが 120 μs を超えるデータは 0.667% である。1000p/s の場合に、データの受信処理が終了する前に次のデータの受信による割り込みが発生する確率が、800p/s の場合に比べ 25 倍も大きい。この場合に受信処理は大きく遅らされて中央から離れた山を形成することになり、この受信状態の大きなゆらぎは安定した受信状態の継続を妨げ、波及効果を与える。1000p/s の場合、中央から 400 μs 以上長い受信インターバルのデータは 1.11%になっている。0.667%のデータが原因となり全体で 1.11%のデータに影響したことになる。これが 1000p/s でゆらぎが急激に大きくなる原因であると推定される。

3.4.3 データ転送時間推定のための実験

図 3.4 で示したレスポンス・タイムの成分(3)を推定するために、データ転送時間を推定するための実験をおこなった。受信したデータを格納するための計算機上での処理に必要な時間は、ネットワーク上でデータを転送するための時間より短いと考えられるので、レスポンス・タイムの成分(3)の主要部分は、ネットワーク上でデータ・パケットを受け渡すための転送時間である。

3.4.3.1. 実験方法

データ転送時間を図 3.9 の構成で測定した。

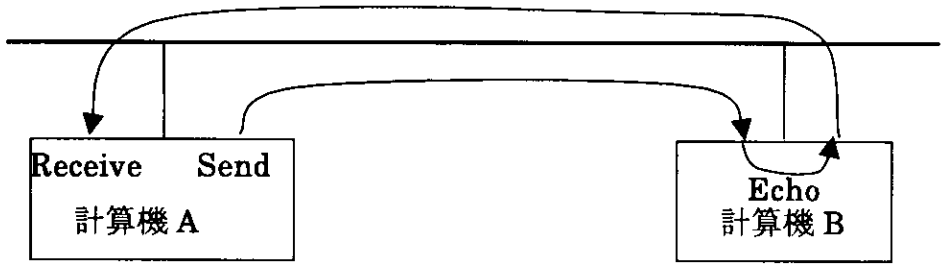


図 3.9 転送時間の測定

計算機 A として SPARCstation/IPX を、計算機 B として SPARCstation 2 を使用した。送信計算機 A から送信時刻をデータとして記入したデータ・パケットをマルチキャストで送信し、エコー計算機 B でそのデータ・パケットを受信し、ただちに別のデータ・パケットとして送信する。計算機 A がこのデータ・パケットを受信し、受信時刻と送信時刻の差を計算すると、ネットワークを A から B へ、B から A へと2回転送する時間が得られるので、この半分の時間が今評価しようとしているネットワークによる転送時間となる。

計算機 A では、3.4.2.1 節で説明した共通の送信コマンド **mcip_busy_snd_rnd** でおこなう。

計算機 B では、つぎのようにエコー処理をおこなう。

エコーは次のコマンドで実行する。

```
priocntl -e -c RT mcip_echo_pkt -p1024 -a1025
```

priocntl -e -c RT は、その後のコマンドをリアルタイム・スケジューリング・クラスで実行することを表す。エコーを行うコマンドは **mcip_echo_pkt** で、オプション **-p1024 -a1025** は、ポート番号 **1024** で受信したパケットをポート番号 **1025** のパケットとしてマルチキャストで送信することを表す。

エコー処理はデータ・パケットを受信すると、ただちに別のポートで送信する。下記にプログラムの骨格を示す。

```
v_length = x_mcip_read(g_mcip, (char *) &g_packet, sizeof(g_packet));
// データパケットを受信する。
x_mcip_write(g_mcip_a, (char *) &g_packet, v_length);
// 別のポートにデータパケットを送信する。
```

計算機 A では、つぎのように送受信時間差の算出をおこなう。

受信は次のコマンドで実行する。

```
priocntl -e -c RT mcip_tra_time -p1025 -u100 -o2000 -h100
```

オプションの意味は送信およびエコーと同じなので省略する。

mcip_tra_time はパケットの受信時刻とパケットに記入されている送信時刻の差を計算する。度数分布表の指定は、100 μ 秒刻みで 2000 μ 秒から 100 段階、12000 μ 秒までとなっている。

プログラムは、データ・パケットを受信し、受信時刻とデータ・パケットに記録されている送信時刻の差を計算する。下記にプログラムの骨格を示す。

```
v_length = x_mcip_read(g_mcip, (char *) &v_packet, sizeof(v_packet));
// データパケットを受信する。
gettimeofday(&v_receive, NULL);
// 受信時刻を求める。
p_send_time = x_lost_sendtime_ptr(v_packet);
// データから送信時刻を取り出す。
```

```
v_sec = v_receive.tv_sec - p_send_time->tv_sec;
v_usec = v_sec * 1000000 + v_receive.tv_usec - p_send_time->tv_usec;
// 送受信の時間差を計算する。
x_hist(v_usec);
// 度数分布表に加える。
```

3.4.3.2. 実験結果と分析

3.4.2.1 節で説明した送信コマンド **mcip_busy_snd_rnd** の-a、-b オプションで指定する送信時間間隔を、20000、10000、8000、5000、4000、2860 と変えて 10000 パケットを送信し、送信インターバルと送受信時間差(ラウンド・トリップ・タイム)を計測した。この送信時間間隔は、送信レートでは、それぞれ 50 パケット/秒、100 パケット/秒、125 パケット/秒、200 パケット/秒、250 パケット/秒、350 パケット/秒に相当する。計算機 B がエコーするので、ネットワーク上の全通信量はこの2倍の、100 パケット/秒、200 パケット/秒、250 パケット/秒、400 パケット/秒、500 パケット/秒、700 パケット/秒となる。

通信量について議論するときは、前節で述べたように物理レベルのデータ長を使わなければならない。今の場合、1040 バイトのデータを送信するための物理的なデータ長は 1098 バイトとなるので、イーサネットの媒体としての最大値である 10Mbps を 100%とすると、実験の送信データ量は、それぞれ最大値の、8.8%、18%、22%、35%、44%、61%に相当する。

転送時間は送受信時間差の半分であるから、送受信時間差を調べれば良い。送受信時間差の表を表 3.5に示す。送受信時間差の測定条件が適切かどうかを調べるために、同時に送信インターバルを測定し、一定間隔に送信できる定常状態になっているかどうかを調べた。その測定結果の度数分布表を表 3.6に示す。上下限以内では時間間隔を等間隔になるようにしているが、データ数の少ない一部の区間は標準の間隔より広く取っており、その時は範囲を表す記号を「~~」と表している。上記プログラムの説明には記述していないが、この場合もシーケンス番号の検査を行っており、250 パケット/秒までデータ欠損は発生していない。

表 3.5 送受信時間差(単位はμs)の分布
p/s はパケット/秒のこと

	送信レート					
	50p/s	100p/s	125p/s	200p/s	250p/s	350p/s
喪失パケット数	0	0	0	0	0	9
送受信時間差						
4000μs 以下	0	0	0	0	0	0
4000~4200μs	7218	599	794	1845	480	2
4200~4400μs	2611	9159	8926	7090	4367	95
4400~4600μs	128	130	173	793	318	2477
4600~4800μs	41	58	35	106	41	2775
4800~5000μs	1	8	5	59	22	369
5000~5200μs	1	2	0	6	8	132
5200~5400μs	0	8	4	3	2902	38
5400~5600μs	0	2	8	3	1616	9
5600~5800μs	0	6	5	11	123	22
5800~6000μs	0	2	1	5	28	29
6000~6200μs	0	4	1	2	9	39
6200~6400μs	0	3	0	2	2	42
6400~6600μs	0	1	1	4	3	1229
6600~6800μs	0	1	3	2	2	1976
6800~7000μs	0	2	2	0	3	305
7000~7200μs	0	2	1	0	4	265
7200~7400μs	0	4	0	0	6	68
7400~7600μs	0	6	2	0	5	15
7600~7800μs	0	1	0	0	2	6
7800~8000μs	0	2	0	0	0	0
8000~~12000 μs	0	0	32	35	20	40
12000μs 以上	0	0	7	34	39	58
合計	10000	10000	10000	10000	10000	9991

表 3.6 送信インターバル(インターバルの単位はμs)

インターバル(μs)	パケット数	インターバル(μs)	パケット数
50 パケット／秒		100 パケット／秒	
10000 以下	49	5000 以下	40
10000~~19800	45	5000~~9800	20
19800~20000	3909	9800~10000	3839
20000~20200	5951	10000~10200	6073
20200~~30000	14	10200~~25000	15
30000 以上	31	25000 以上	12
125 パケット／秒		200 パケット／秒	
2000 以下	45	1000 以下	56
2000~~7800	25	1000~~4400	16
7800~8000	3790	4400~4600	8
8000~8200	6116	4600~4800	87
8200~~20000	14	4800~5000	3759
20000 以上	9	5000~5200	5981
		5200~5400	52
		5400~5600	5
		5600~~20000	28
		20000 以上	7
250 パケット／秒		350 パケット／秒	
1000 以下	68	400 以下	0
1000~~2600	28	400~600	28
2600~2800	27	600~800	234
2800~3000	88	800~1000	678
3000~3200	123	1000~1200	66
3200~3400	1990	1200~~2400	58
3400~3600	324	2400~2600	38
3600~3800	2	2600~2800	166
3800~4000	1800	2800~3000	7685
4000~4200	2880	3000~3200	62
4200~4400	1	3200~~4400	56
4400~4600	426	4400~4600	16
4600~4800	2051	4600~4800	277
4800~5000	127	4800~5000	575
5000~5200	32	5000~5200	22
5200~~20000	27	5200~~10000	30
20000 以上	5	10000 以上	8

転送時間の最小値を調べるには、なるべく低負荷で実験をおこなう必要がある。表3.5 の 50 パケット/秒のデータから、送信したパケットを折り返して受信するまでの時間であるラウンド・トリップ・タイムは約 4ms であることが分かる。従ってデータをネットワーク経由で送信するときの遅延時間(転送時間)はこの半分の 2ms と考えて良い。

物理レベルのパケット長 1096 バイトのデータは、10Mbps のケーブル上で 877 μ s の時間を占めるので、ケーブル上での物理的な遅延は 877 μ s であり、残りの約 1100 μ s は、回路および計算機処理での遅延時間となる。ネットワークを 100Mbps のイーサネットに変更すれば、物理的な遅延時間は 1/10 の約 90 μ s になり、計算機の性能が向上すれば、回路および計算機処理での遅延時間も減少することが期待できる。

送信レートが 250 パケット/秒場合は、送信時間間隔が 4ms であり、ラウンド・トリップ・タイムと一致してしまう。つまり送信開始と受信終了が同時となる。送信インターバルを見ると、この衝突のために処理を約 700 μ s 遅らされ、送信インターバルが正常時の 4000 μ s から 4700 μ s になっている。送信時間間隔は絶対時刻でおこなっているため、700 μ s 遅れて送信すると、次の送信は 3300 μ s 後に開始するため、この送信インターバルの所にもピークが出現している。

送信レートが 350 パケット/秒の場合は、送信インターバルはラウンド・トリップ・タイムより約 1200 μ s 短く、受信より次の送信が先におこなわれ衝突する確率は減少するので、平均インターバル時間以外の頻度は少なくなっているが、全体にゆらぎが大きくなっており、負荷が高いことをうかがわせる。送信が受信より遅れる場合は、受信完了までの待ち時間が長くなっている。

送信レートが低い場合でも送信インターバルのゆらぎが大きい。50 パケット/秒の場合ですら本来のインターバルである 20ms よりも 10ms 以上長くなっている場合がある。この実験での送信はタイムシェアリング・スケジューリング・クラスであり、オペレーティング・システムのプロセスが稼動したために予定時刻に実行できなかったためと考えられる。リアルタイム性を保証するには、低負荷で実行する必要がある。

転送レートが高くなると転送時間のゆらぎが大きくなるため、通信負荷と計算機の性能に注意しなければならないが、100 パケット/秒 の場合で平均的には転送時間を 3ms と見て良いだろう。厳密に最長転送時間を決定することはできないが、共有短期記憶機構(Shared short Term Memory, SSTM 後の説で詳説する)により平均的な転送時間の最悪値が分かっていればよい。そこで転送時間としては 350 パケット/秒でも約 99%のパケットが含まれる 5ms と見積もれば良いと思われる。この 5ms のうち、最短転送時間が 2ms なので、受信処理遅れのゆらぎは 3ms である。

3.4.4 評価結果

イーサネットでマルチキャスト通信方式を採用することにより十分なデータ通信量を得られることが分かった。

単一セグメントの10Base5のイーサネット上で、通信をデータ転送だけに限定し、SPARCstation2 上の受信プログラムをリアルタイム・スケジューリング・クラスで実行すると、800 パケット/秒でデータを転送できた。物理データ長は 1096 バイトだが、測定値データ部分は 1024 バイトなので、データ通信量としては 800KB/s となる。これは 3.3.1.2 節の分析により必要とされた通信量の 20 倍にも達する。この値は安ら[11]の VME バスを使用するシステムの理論限界値 1.14MB/s の 70%であり、実システムで実現した限界値 256KB/s と比べると約 3 倍ほど大きな値になっている。このことはデータ転送用のバスとして、これまで使われてきた VME バスの代わりにイーサネットを使える可能性を示している。イーサネットに接続できる計算機の台数と VME バスのスロット数21を比較すれば、イーサネットのほうがはるかに拡張性に優れている。

低速リアルタイム用としては十分な通信速度であることが分かった。

512 チャネルの ADC データを通信するために必要な転送時間は、最短転送時間 2ms と受信処理遅れのゆらぎ 3ms を合わせ、平均として 5ms と考えれば十分であることが分かった。これが 3.3.1.3 節のレスポンス・タイムの成分(3)の時間であり、低速リアルタイム用としての要求性能である 100ms の一部となる。

実験の結果、イーサネットをデータバスとして使用するときの問題点も明確になった。

実験の結果データ欠損の原因は OS のプロセスにより受信プログラムが実行を妨害されたためであり、同程度以上の優先度を持つプロセスが同時に実行されていれば、やはりデータ欠損を生じる原因となる。従ってデータ欠損を生じないためには、

1)プログラムをリアルタイム・スケジュール・クラスで実行する

2)マルチ・プロセッサの計算機を使用する

などの性能改善により対処する必要がある。広域ネットワークでのデータ欠損はルータの性能が不十分かネットワーク自体の通信容量が少ないからであり、通信方式そのものがデータ欠損の原因ではない。

さらにもしデータ欠損が発生した場合は、送信データに付けた連番の連続性により容易に調べられるので、受信プログラムは必ずデータ欠損監視をおこない、検出時には負荷の再調整やより高性能の計算機への変更をおこなえば良い。

また、多くのプロセスをリアルタイム・スケジューリング・クラスにすると、そのプロセス間での競合によりデータ欠損を生じることは明らかなので、受信処理など、他に回避手段のないプロセスのみリアルタイム・スケジューリング・クラスとし、他のプロセスでは大容量バッファを用いて、処理時間の変動が大きくなっても可能な限りデータ欠損を生じないようにすべきである。

なお図 3.8 で送信インターバルが短い場合のように安定性を損なう原因は OS のプロセスが必要とする時間であり、計算機の性能が向上すると、この時間は減少する。したがって将来はさらに大容量の通信が可能となり、転送時間も減少する。

3.5 基本設計

これまでおこなってきた分析結果をもとに実際にプロトタイプ・システムを作成するための基本設計をおこなう。システムの基本要素は AD 変換装置と表示プログラムであったが、表示プログラムの分析により、データ伝送系と呼ぶサブシステムを分離すると、システム全体の柔軟性が向上することがわかった。AD 変換装置、データ伝送系、表示プログラムについて基本設計をおこない。最後に開発したプロトタイプ・システムについて概説する。

3.5.1 AD 変換装置の基本設計

多くの AD 変換装置が低速リアルタイム用 CAMAC モジュールとして販売されている。CAMAC モジュールとなっている高速 AD 変換装置はトランジェント・レコーダであり、リアルタイムにデータを取り出すことはできない。リアルタイム用の AD 変換装置は記憶容量が少なく、バッチ処理ができない。このため同一データに対してリアルタイム処理とバッチ処理を同時に行うなど、3.3.2 節の要求を満たすには、新型の AD 変換装置を製作する必要があった。

基本設計では AD 変換装置の論理的な構造を定める。

- AD 変換の精度は 10～16 ビットなので、1 データは 2 バイトとする。
- 時刻差による補正を不要にするため、同時サンプリングとする。
- 従来型バッチ処理プログラムに対応するため、装置内に 32MB 以上のメモリを内蔵する。
- AD 変換とデータ取り出しを同時に行うため、図 3.10 のように内蔵メモリはリング・バッファとし、AD 変換済みのデータがどこまでか分かるようにするため write index register を設ける。

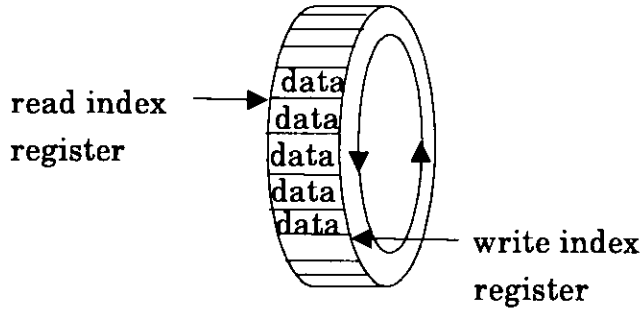


図 3.10 リング・バッファ構造の内蔵メモリ

- データ取り出しが間に合わないことを検出するため、どこまでデータを取り出したかを read index register でプログラムから装置に通知し、装置は AD 変換データを格納するメモリが無くなったらプログラムに通知する。

- データ取り出しを可能な限り高速化するため、AD 変換装置は SBus に直結し、memory mapped I/O により内蔵メモリを主記憶と同等に扱えるようにする。

このような構造の AD 変換装置にすると、リアルタイム処理では次の手順を繰り返してデータ取得をおこなえる。

- 装置の write index register を読み、まだ取り出していないデータがあるかどうか調べる。
- まだ取り出していないデータがあれば、アドレスを計算してそのデータを取り出す。
- 取り込んだデータの次のアドレスを装置の read index register に書き込む。

単一のデータを読み出す手順は上記のとおりであるが、複数チャネルの場合、全チャネルのデータを連続して取り出せば良いので、データ取得を続けるあいだは装置レジスタへの入出力は必要なく、処理は高速となる。SBus は 32 ビット幅であり、内蔵メモリを 1 回アクセスすると 2 チャネルのデータを取り出せるので、16 チャネル 100KHz のデータを取り出すには内蔵メモリを毎秒 800K 回読み込めば良い。SBus のクロックは 20MHz なので、要求を満たすことは可能と考えた。

3.5.2 データ伝送系の分離と基本設計

表示プログラムに対する要求のほとんどは柔軟性に対する要求であるため、明確な概念を形成することにより、柔軟性による変化を管理可能とした。要求を分類すると次のようになる。

- 表示方法に関するもの
- 表示性能に関するもの
- データの組み合わせに関するもの
- データの表現に関するもの
- 同期および時系列に関するもの

このように分類すると、要求されている柔軟性の多くは「表示」という機能に対するものではなくデータそのものの扱いに対するものであることが分かる。そこで機能分割をおこない、データをデータ源から表示用計算機まで伝送し、その間で必要なデータの取捨選択、統合、時間スケールの調整などをおこなう機能の集まりをデータ伝送系として分

離・独立し、表示用計算機上の表示プログラムは基本的にひとつのデータ集合を受け取って表示するものとした。

データ伝送系に要求される機能では、物理量への変換などをおこなう機能以外は決まりきった機能になり、扱っているデータの内容とは独立になるため、あらかじめ必要な機能を提供する処理プログラム、あるいは処理ライブラリを作成しておくことができる。

このデータ伝送系に対して表示用計算機上で稼動する表示プログラムおよび表示機能の集まりを表示系と呼ぶ。このように機能分割をすると、両系間のインターフェースを明確に定めることが重要となる。表示系機能の中心となる表示用プログラムの基本設計は本節から分離して次の節で説明する。

データ伝送系はアプリケーションに依存しない独立サブシステムとして作成しなければならない。したがってデータ伝送系で扱うデータの表現について、あらかじめ決めておく必要がある。データ伝送系ではデータを図 3.6 のように形式が決まっているパケットという単位で扱う。パケットには少なくともパケットの種類を識別する番号、時刻、データの値が含まれる。このパケットをそのままデータ転送用にも使用するため、データ欠損検出のためのシーケンス番号も含む。

アプリケーションが必要とするデータは、現在値だけではない。時系列処理あるいは時間スケール変換のためには過去データを利用できる必要がある。また受信したデータを処理するための同期をとる必要がある。このために表示用計算機では受信したパケットのうち、必要な範囲までの過去データを保存する必要がある。これらの目的のために短期記憶機構を設計した。

「受信したパケットを短期記憶から取り出す」ことは「AD 変換したデータを内蔵メモリから取り出す」こととほぼ同じであり、AD 変換装置の場合と同様にリング・バッファを基本として短期記憶機構を設計できる。インターフェースを全く同じにできると、表示プログラムでは AD 変換装置のデータを表示しているのか、受信パケットを表示しているのかを区別する必要がなくなり、同一のプログラムを使えることになる。AD 変換装置の設計と短期記憶機構の設計では、可能な限り同じ使い方ができるように工夫した。

同一表示用計算機内で同一パケットを扱う複数のプロセスが稼動している状況は容易に想像できる。したがって短期記憶機構は複数のプロセスに対して同じデータを供給する必要がある。短期記憶機構にデータ共有機能を組み込み、共有短期記憶機構 (Shared Short Term Memory, SSTM) とすることが必要である。SSTM が最終的にはこのシステム全体が有機的に動作するための基本技術のひとつとなった。

表示プログラムの開発を容易にするためには、SSTM から得られるデータは物理量であることが望ましい。値の一貫性を保つためには、データ発生源で物理量に変換して

から送信するのが一番良いが、ADC 生データは2バイトなのに対し、物理量は 4 バイトを必要とするので、データ通信量が 2 倍となることと、表示用計算機で変換するなら表示に必要なデータだけを変換すればよいため、ここでは SSTM から表示プログラムにデータを渡す際に物理量に変換するものとした。このように SSTM に物理量への変換機能を組み込んだものを拡張 SSTM と呼ぶ。ただし通常は区別しない。異なる表示用計算機上で同じデータを扱う場合があるため、同一の物理量への変換がおこなわれるようにするための一貫性維持の仕組みが必要となった。

以上のようにデータ伝送系とのインターフェースとして、データ表現をデータ・パケット、アクセス機構として SSTM を使うことに決めたことにより、データ伝送系が提供する機能を次のように決めた。それぞれの機能は、ライブラリまたは実行プログラムとして提供される。

- データ・パケット送受信ライブラリ
- 共用短期記憶機構ライブラリ
- SSTM 上のデータ・パケットを自動的に送信するプログラム
- 受信したデータ・パケットを自動的に SSTM に格納するプログラム
- データ・パケット統合プログラム

このほか必要に応じて作成したプログラムはデータ・パケットの内容に依存しないため、いろいろなところで使用できる。

3.5.3 表示プログラムの基本設計

前節の設計でデータ伝送系を分離したので、表示プログラムはひとつの SSTM からデータを取り出して表示すれば良いことになった。SSTM を使えば過去データも直接アクセスできるため、1軸はデータ項目、2軸は時間の2次元配列として与えられたデータを表示すると考えて良いので、表示プログラムの設計では、純粋に内部処理に傾注すれば良い。

表示プログラムに対しては各種の表示方法に柔軟に対処できることが要請される。したがって表示プログラムのためのライブラリを提供して新規表示プログラムの開発を容易にした。

表示プログラムは使用するウィンドウ・システムに依存するという問題がある。ウィンドウ・システムの標準が Motif になるか Open Window になるかが決まっていなかったため、垂直機能分割設計をおこなってウィンドウ・システム選択の影響を小さくした。すなわ

ち、ウィンドウ・システムに共通する下位ライブラリである Xlib にのみ依存する表示プログラムの下位ライブラリと、ウィンドウ・システムに依存する表示プログラムの上位ライブラリに分割し、下位ライブラリを共通に使用できるようにした。(このような考慮を払って基本設計をおこなったが、LHD 制御データ処理装置での表示は WWW と Java というまったく新しい技術を使用している。)

標準的な表示プログラムとしてスクロール・グラフを作成することとした。これは受信したデータを直ちに表示するもので、レスポンス・タイムの評価のために必要であった。同時に 8 データを、切り替えにより 64 以上のデータを表示できるものとした。

3.6 開発したプロトタイプ・システムの概説

基本設計に基づいて開発したプロトタイプ・システムの構成要素を図 3.11 に示す。

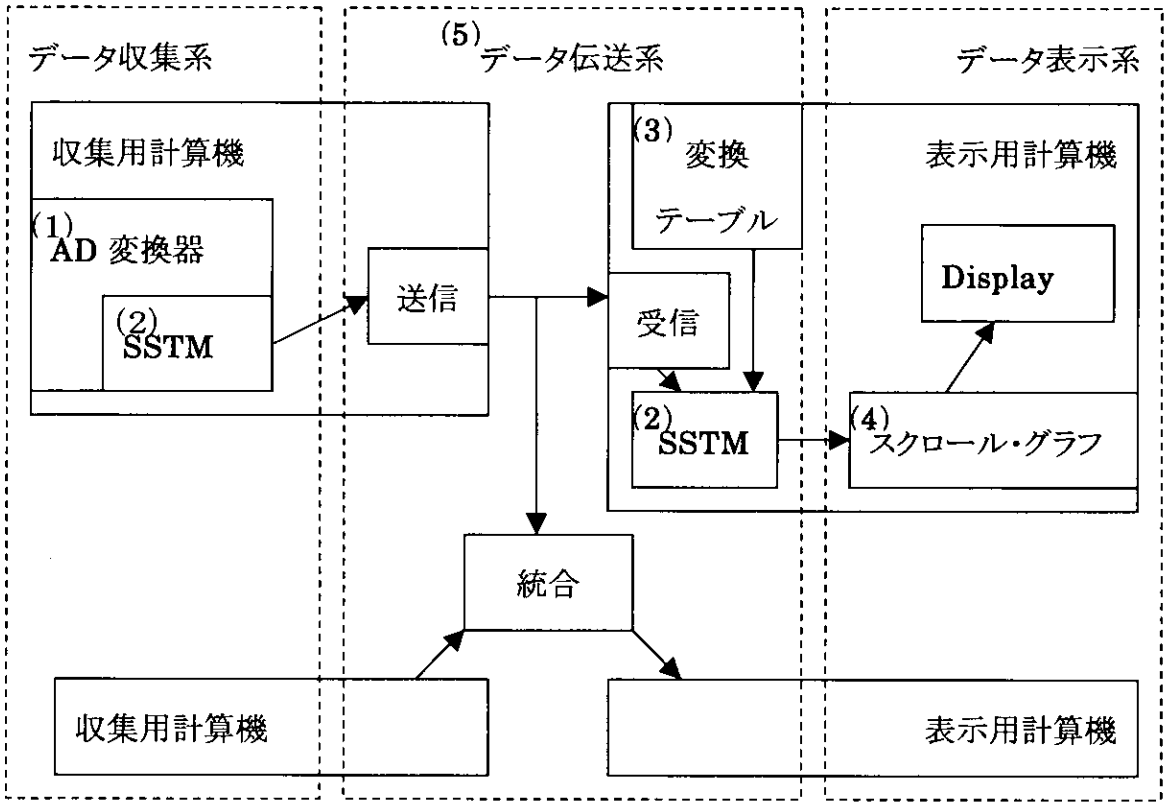


図 3.11 プロトタイプ・システムの構成要素

この図はプロトタイプ・システムでの重要な構成要素を示すものであり、番号をつけた要素について説明する。これらの要素間での処理がデータ伝送系のサービスであり、実現したサービスについては 4.5 節で説明する。

データ収集系、データ伝送系、データ表示系とは、それぞれデータ収集機能、データ伝送機能、データ表示機能に関わる装置、プログラム、データ・バッファ全体を意味する。データ伝送系は、データ表示系などの外部の系での利用には、各系のプログラムに組み込んで使用するライブラリを、データ伝送系内部での利用には、独立して稼動するアプリケーション・プログラムを提供する。データ収集系の機能は収集用計算機、データ表示系の機能は表示用計算機により実行される。データ伝送系の機能は、収集用計算機でも表

示用計算機でも実行される。図の統合機能は、どちらの計算機で実行してもよく、また統合専用の計算機を充ててもよい。

データ収集機能のひとつであるアナログ・データ収集機能は、AD 変換器を接続した収集用計算機において実行される。データ表示機能のひとつであるスクロール・グラフ表示は、Display を接続した表示用計算機において実行される。

データ伝送機能の役割は、データ収集機能により得られたデータをデータ表示系に渡すことである。もっとも簡単な場合は、データ収集用計算機上での送信機能とデータ表示用計算機上での受信機能だけが利用される。通常は SSTM と変換テーブルにより、物理量をデータ表示系に提供する。複数データ源のデータを統合する場合はデータ統合機能を利用する。データ統合をどの計算機上で実行するかは、システムの負荷を調べて決めることになる。

プロトタイプ・システムの設計目標である 100ms のレスポンス・タイムとは、図の AD 変換器で変換終了したデータが、図の Display に表示されるまでの時間である。

このシステムの典型的な利用方法は次のようなものである。

AD 変換装置を接続している収集用計算機では、定期的にリアルタイム・データをデータ・パケットの形式にまとめ、マルチキャスト通信方式で送信すると同時に、ファイルに保存する。表示用計算機では受信したデータ・パケットを SSTM に格納する。データ受信を待っているスクロール・グラフ・プログラムは直ちに波形として表示する。

この利用方法を出発点として、運用規模の表示システムにするには、収集用計算機、表示用計算機、必要であればデータ変換をおこなう計算機を多数ネットワーク上に配置する。収集用計算機が収集したデータを一次データ・パケットとしてネットワーク上へ送信する。一次データ・パケットを加工して新たなデータ・パケットを作成する必要がある場合は、ネットワーク上のデータ伝送系機能を実行するアプリケーションがデータ・パケットの統合・変換をおこなって多種の要求されたデータ・パケットを生成して、ふたたびネットワーク上に送信する。これを二次データ・パケットと呼ぶ。送信はすべてマルチキャスト通信方式で送信されるので、ネットワークに接続された任意数台の表示用計算機によって受信でき、リアルタイム表示をおこなう。必要なら制御用計算機の持つシステムの状態や、表示用計算機の出力する要約データやイベント・データを一次データと統合することもできる。

次に作成したシステムの主な要素について簡単に説明する。表題につけた番号は図 3.11 でそれぞれの要素に付けてある番号である。

(1)新型 AD 変換装置

SPARCstation の SBus に直結する装置で、拡張ボードだけからなる 8 チャネルの小型 AD 変換装置 AD70700 と、外付け筐体に AD 変換機を装備し、32 チャネル単位に 256 チャネルまで増設できる大型 AD 変換装置 AD71000 を開発した。共に 100KHz までの変換速度を持ち、AD71000 は最大 32MB まで内蔵メモリを増設できる。ワークステーション内の主記憶上に構成された共用短期記憶機構と AD 変換装置内の短期記憶機構をデータの読み出しに関してまったく同等に扱えるようにするため、主記憶のメモリ・マップ上に AD 変換装置の内蔵メモリを配置し、複数のプロセスが同時に内蔵メモリからデータを読み出せるようにして、AD 変換装置を共用化した。物理的な装置としてはひとつしかないので、start/stop などの制御がおこなえるのはひとつのプロセスからだけである。

(2)共有短期記憶機構(SSTM)

SSTM は、非同期に動作するプロセス間でのデータ授受のインターフェースであり、図のように入力と出力の仲立ちをしている。AD 変換器内の SSTM では、AD 変換素子が入力となる。

(3)変換テーブル

変換テーブルは ADC の生データから物理量に変換する方法を定義したテーブルであり、SSTM の初期化時に変換方法をデータ・パケットの各要素と対応づけることにより、プログラムに対して物理量へ変換するサービスを提供する。変換テーブルはテーブル定義言語により定義されたファイルから生成される。変換方法として線形変換とスプライン補間[23,24]が定義されている。すべての計算機で同一の定義ファイルを参照することにより一貫性を保持している。

(4)スクロール・グラフ

データをスクロールする波形として Display に表示するプログラムで、SSTM からデータ・パケットを取得し、直ちに表示をおこなう。1 画面に 8 データの波形を表示でき、画面切り替えにより 64 データを表示できる。表示画面の大きさなどは起動時の定義ファイルで設定する。自動レンジ、固定レンジの指定を動的におこなえる。

(5)データ伝送系

データ伝送系の実体はデータ伝送にかかわる機能を提供するサービスの集合体である。データ伝送機能には、SSTM からデータ・パケットを取り出して送信する。受信したデータ・パケットを SSTM に書き込む。複数のデータ・パケットを統合してひとつのパケットにする。SSTM 内のデータ・パケットから間引いたデータ・パケットを作るなど、たくさん

の機能がある。これらの機能は、小さな独立したアプリケーション・プログラムにより実現されるもの、データ収集系やデータ表示系のプログラムが呼び出すライブラリにより機能を提供するものなどがある。

4 プロトタイプ・システムの構築

前節の基本設計に基づいてプロトタイプ・システムを構築した。図 4.1 にプロトタイプ・システムの構成要素およびそれらの関連を示した。この図は、実システムの構成図ではなく、システムを構成するプログラムなど構成要素の関連を示すための図であり、実際のシステムでは同じ構成要素が何度も現われたり、図には示していない接続がおこなわれたりする。3.6 節で概説した構成要素について、その実現方法や機能について、本章で詳細に説明する。図 4.1 中にある構成要素間を結ぶ線に付けた番号は、データ伝送系のサービス機能および関連する機能であり、4.5 節で詳細に説明する。

なお、伝送系の機能である統合は、収集用計算機、表示用計算機あるいは統合専用の計算機のどれでおこなっても良いので、図 4.1 では稼動する計算機を明示していない。

本システムは多数の計算機が共同作業を行う分散処理システムであり、図 4.1 から分かるようにデータ伝送系に含まれるさまざまなアプリケーション・プログラムやライブラリの機能によりデータの流れを統合している。図 4.1 のように多数のところで、データ授受がおこなわれるので、データ受け渡し方法を統一したインターフェースとして確立する必要がある。インターフェースを、データ表現は 4.1 節で説明するパケット構造とし、アクセス方式を 4.2 節で述べる SSTM とすることにより統一した。

4.1 データ・パケットによるデータ表現の統一

測定データの分配・統合において、データ長だけ分かれば良い場合もあるが、測定値以外の管理情報を必要とすることがある。これはイーサネット・パケットにはイーサネット・ヘッダ、IP パケットには IP ヘッダが付けられるように[25]、パケットに自己記述情報を付けることにより、end to end すなわち、データ発生源とデータ利用先での処理以外の処理を自動化するものである。たとえば、パケット長(packet_size)が分かれば、データを自動的に保存することができる。連番(sequence)が分かれば、データ欠損を検出することができる。

LHD 表示システムにおいての end to end とは、AD 変換装置を接続した収集用計算機と表示用計算機の関係であり、中間にある伝送系をこれら両端の計算機上の機能から独立させるためには、LHD 表示システムにおけるデータ・パケットを定義しなければならない。

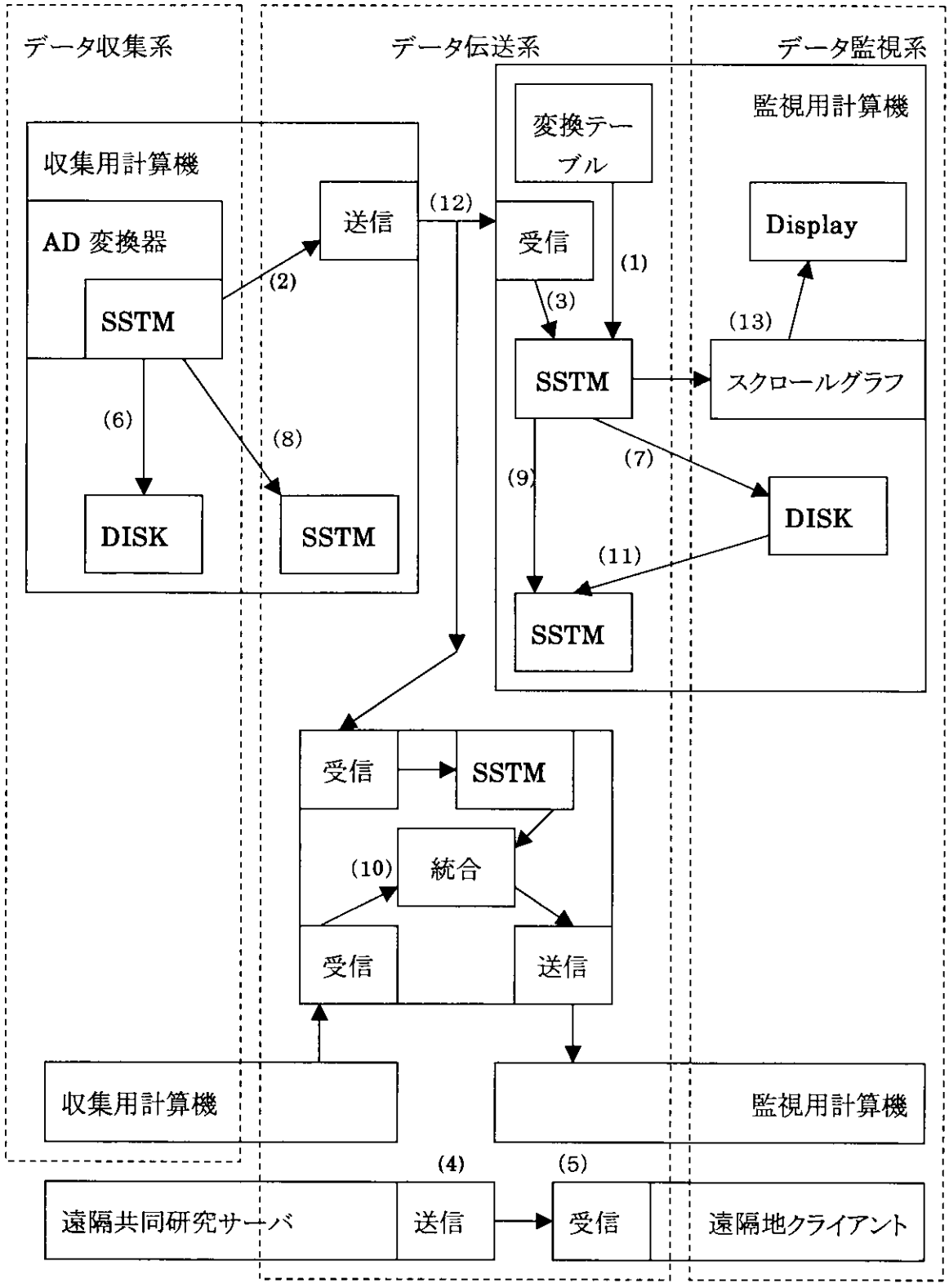


図 4.1 プロトタイプ・システムのソフトウエア構成要素

データ・パケットはヘッダ部と値部から構成され、プロトタイプ・システムにおいては、ヘッダはリスト 4.1 のように定義されている。この定義を図示したものが図 3.6 である。

```
#define C_STM_PACKET_HEADER          ¥
    unsigned short byte_size;¥
    unsigned char  packet_class;¥
    unsigned char  packet_id;¥
    int            time_sec;¥
    int            time_usec;¥
    unsigned char  stm_packet_option_1; ¥
    unsigned char  stm_packet_option_2; ¥
    unsigned char  stm_packet_option_3; ¥
    unsigned char  sequence;

typedef struct s_stm_packet_header {
    C_STM_PACKET_HEADER
}   t_stm_packet_header;

#define c_stm_packet_class_adc       0
#define c_stm_packet_class_vto       2
```

リスト 4.1 データ・パケットのヘッダ

byte_size フィールドにはパケット長が入る。packet_class フィールドにはパケット・クラスが、packet_id フィールドにはパケット識別番号が入る。パケット・クラスはデータ表現の分類とし、現在は ADC 生データのクラスと物理値に変換されたデータのクラスが定義されている。パケット識別番号は同じデータ表現でのデータ内容の種類を番号で表す。パケット・クラスとパケット識別番号でパケットを分類する。パケットの時刻は秒単位の値とμ秒単位の値で表現する。μ秒単位の値は、リアルタイム表示では必要としなくても、高速データも同じ表現で保存しアプリケーション・プログラムを共通にするために定義してある。sequence フィールドはパケット欠損を検出するためのシーケンス番号である。stm_packet_option_1 などのオプション・フィールドは、データ値以外の情報を伝えるためにアプリケーション・プログラムで使用して良い。

データ・パケットの値部は、このヘッダの下に生データ(c_stm_packet_class_adc)の場合は short 型の配列、物理量(c_stm_packet_class_vto)の場合は float 型の配列として記述される。

このようにデータ・パケットを定義しておくと、1パケットを受信してパケット・クラスとパケット識別番号を調べ、データベースにデータ内容の定義を問い合わせることにより、表示を自動化におこなえるようになる。

システム内のすべてのアプリケーションがこのデータ・パケット定義に基づいて作成されるため、この定義を変更することは重大な影響を与える。本方式により運用システムを構築するときは、このデータ・パケット定義については、将来の利用方法まで考慮して十分な検討をしなければならない。

4.2 共用短期記憶機構(SSTM)

SSTM は非同期のプロセス間でデータ・パケットを受け渡す機構であり、おもにネットワークからの受信に使用されるが、計算機内でのプロセス間インターフェースにも使用される。データ・パケットでデータを受け取るアプリケーションのインターフェースを SSTM に限定することにより、データ・パケットへのアクセス方法を統一できる。

SSTM は計算機内でのデータ・パケットの分配や物理量の提供もおこない、アプリケーションの作成と利用を容易にする。

4.2.1 SSTM によるアクセス方法の統一

本システムは多数の計算機がネットワーク上で協調する分散処理システムであり、各計算機上のアプリケーションはそれぞれ独立して非同期に動作している。これらのアプリケーションが容易に協調できるようにすることが、システムの柔軟性にとって重要な要素である。各計算機上では計算機上のアプリケーションと非同期にネットワークからのデータ受信がおこなわれる。このためになんらかのバッファが必要となる。このバッファとしてこの節で述べる SSTM を用いることにより、アプリケーション間でのデータの受け渡しとネットワークを経由しての受信のインターフェースを統一した。このようにすると、アプリケーションにとってデータが計算機内で作成されたかネットワークから得られたかの区別がなくなり、アプリケーション側ではネットワークの存在を考慮する必要がなくなる。図 4.1 あるいは図 5.1 に示したように、受信データが直ちに処理されない場合は、かならず SSTM に蓄積している。

4.2.2 SSTM によるデータの分配

マルチキャストを使うとネットワークを介した計算機へのデータ分配を、負荷と遅延の増大無しにおこなえることが分かった。しかしこの方法だけだと各計算機が同一のデータに対して単一の機能を実行することとなり、計算機の台数が増大し、ネットワーク上の通信量も非常に大きくなり、ネットワーク自体の通信容量という限界にぶつかることとなる。計算機の性能は増大しており、一台の計算機で複数の処理をおこなうのが普通であるので、計算機内でのデータの分配も考えなければならない。

計算機内でのデータ流通を標準化し、データ分配を行うための機構として SSTM を使用する。SSTM は図 4.2 に示すように基本的にはリング・バッファであり、データ・パケットを保存の単位とする。SSTM は目的に応じて必要な期間のデータ・パケットを保持する大きさに作成する。SSTM の実体は、OS のサービス機能の一種である共用メモリを使用して作成する。

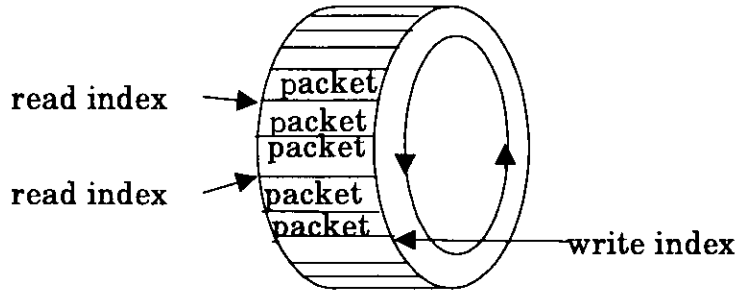


図 4.2 共用短期記憶機構

SSTM の目的は、一定期間の過去データを使用可能にすることと、複数のプロセスから同時にデータを取得可能にすることである。たとえば、LHD のリアルタイム表示で、超伝導コイルへの入り口での液体ヘリウムの温度を、コイルからの出口での温度と同時に表示して比較するためには、入り口から出口までに移動する時間だけの過去データを保持する必要がある。このようにデータを時系列データとして扱える必要がある。

書き込みプロセスはデータ・パケットを書き込んでから、バッファ上での現在の書き込み位置を示す write index を進める。周期の分かっている周期的データ・パケットであるなら、時間差から index の差を計算できるので、write index からの相対アドレスにより過去データ・パケットを参照できる。読み出しプロセスはそれぞれの読み出し位置である read index を持ち、バッファ内を自由に参照できる。複数の読み出しプロセスを許すため、SSTM では、バッファと write index は共用メモリ内に置かれている。これに対して、read index は各読み出しプロセスの固有のメモリ空間に置かれる。図 4.2 では read index がふたつあるので、ふたつのプロセスが同時に SSTM を使っている状態を表わしている。

なお、周期的データ・パケットでない場合は、データ・パケットに時刻が記入されているので、この時刻の値を調べて目的の時刻のデータ・パケットを得る。

このように共通リソースに対して書き込みプロセスと読み出しプロセスがある場合は、reader writer problem として有名であり、通常はセマフォーなどを使ったプロセス間同期機構を使用する。しかしプロセス間同期機構はオペレーティング・システムが提供する負荷の重い処理であり、LHD 表示システムのように 1 データごとの高速処理で実行するには不適切な機構である。したがって、SSTM では制御に必要な共通リソースを write index のみとし、読み出しプロセスは write index を読みとるだけとし、SPARCstation

でアトミックな命令であることが保証されている[26]1ワードの read/write だけを使うことにより、プロセス間同期機構を使わないように設計されている。

書き込みプロセスが複数の場合は、書き込むデータ・パケットと write index の整合性を保証しなければならないため、プロセス間同期機構を必要とする。書き込みプロセス1が write index を読み込み、1進めて書き込もうとする前に書き込みプロセス2 が write index を読み込むと、両方のプロセスが同一 index のところにデータ・パケットを書き込むことになる。このように複数のプロセスが共通リソースの更新をおこなう場合は、プロセス間で同期をとる必要がある。SSTM では、読み込みプロセスは write index を読み込むだけであり、かつ write index は単調に変化するだけなので、読み込んだ write index の正しさ(write index が示すデータ・パケットが存在すること)は保証されている。

リング・バッファ自体は、たとえば RS232C 通信ドライバで文字単位から行単位への変更や速度調整のために用いられているが、SSTM の目的である過去データへのランダム・アクセスや複数プロセスへのデータ提供の例はなく、ここでの試みは新しいものである。

SSTM によるデータ提供機能を使うことにより、アプリケーション・プログラムは、それぞれの必要に応じて自由にそれぞれのタイミングで過去データを含めたデータを処理することが可能になった。一時的に負荷が大きくなり処理遅れを発生することがあっても、短期記憶が十分な大きさであれば平均的には処理をおこなうことができるので、ピーク性能ではなく平均性能で計算機を選択できる利点がある。

さらに、SSTM を使用するアプリケーションは、それぞれの read index と SSTM が保持する write index との差を調べることにより、平均処理遅れ、最大処理遅れなどの性能測定をおこなうことができ、アプリケーションの要求する計算機の性能を正確に評価できる。処理遅れが徐々に増加する場合は、性能が絶対的に不足していることを意味する。大きな遅延のないリアルタイム処理をおこなうには、最大処理遅れを調べ、その原因となる一時的に高負荷となる処理を別の計算機に移すか、マルチ・プロセッサとするなどの性能向上策を施すことが必要である。

SSTM の仕組みは説明したように単純なので、SSTM に格納されるデータ型はデータ・パケットにかぎらず、どのような型でも対応できる。データが可変長型の場合は、データ本体を格納する領域を別に確保し、その領域へ格納したデータへのポインタを SSTM への書き込みデータとすれば良い。

固定長データを扱う SSTM は次のコマンドで作成する。

```
stm_create -f <filename> -t <type> -s <data size> -n <noof data>
```

・**<filename>**は共用メモリのキーとなるパス名。

・<type>は r または p で示すデータの種類で data size の意味に影響する。
・r は raw データを表し、このときの data size はデータのバイト数、
・p は packet データを表し、このときの data size はデータのチャネル数。
・<noof data>は短期記憶に保管できるサンプル数。

raw データは、AD 変換装置から出力する short int 配列データのためのもので、packet データは、前節で定義したデータ・パケットのためのものである。このように、AD 変換装置のデータを直接扱ったり、ネットワーク上を流れるデータ・パケットを扱うことが容易になるようにしている。このコマンドが正常におこなわれると共用メモリを表すパス名のファイルが作成されるので、書き込みまたは読み出しのためにオープンして使用する。オープンする順序にも、書き込み読み出しをおこなう順序にも何も制限はない。書き込めるなら書き込んで良いし、読み出せるなら読み出して良い。存在しないデータをアクセスしてはならないということだけが制限である。

4.2.3 SSTM による物理量の提供

前節で説明した SSTM は AD 変換装置からの生データをネットワーク間で通信するときのバッファとして使用したり、計算機内で生データを分配するときに使用される。表示用アプリケーションとのインターフェースでは、生データではなく物理量で扱う方が望ましい。物理量は単精度浮動小数点数で表現すれば良いので固定長データであり、固定長用 SSTM を利用できる。そこで SSTM を拡張して、ADC の生データから物理量へ変換し、物理量を提供する拡張 SSTM を作成した。図 4.1 の(1)で示すように拡張 SSTM は、データ取得時に物理量への変換テーブルを使用する機能を有する SSTM である。

もともと物理量が入っているデータ・パケットならば、それを保存して提供するには SSTM そのもので良い。「生データから物理量への変換」をおこなうために、いろいろな仕組みが必要となる。この仕組みは複雑であるが、利用者は欲しい物理量が入っているチャネル番号(またはデータ名)を知っていれば良いだけで、SSTM としての使い方はまったく同じである。利用者は、いつどのようにして物理量に変換されるかを知る必要は無い。このように SSTM と拡張 SSTM の見かけ上の違いはわずかなので、特別な場合以外は区別せず SSTM と言う。

拡張 SSTM は次のコマンドで作成する。

```
stmx_create -c <config file>
```

物理量への変換に必要な情報は多量なので、変換情報を記述したコンフィグレーション・ファイルを引数で指定する。使用例をリスト 4.2 に示す。

```
stm "second.stm" 32 channel 1000 data;
#
load table ;
#
# ivcoil data
include </opt/JK/etc/stm/ivcoil.channel>
```

リスト 4.2 拡張共用短期記憶のコンフィグレーション・ファイル例

このコンフィグレーション・ファイルでは、32 チャネルのデータを 1000 単位時間にわたり保持する拡張短期記憶を作成すること、物理量への変換をおこなうため変換テーブルを使用すること、変換テーブルの詳細情報は/opt/JK/etc/stm/ivcoil.channel ファイルから取得することを示している。include 文は共通情報をひとつのファイルとし、複数のコンフィグレーション・ファイルから同一ファイルを参照することにより一貫性を保証するための仕掛けである。/opt/JK/etc/stm/ivcoil.channel ファイルの内容をリスト 4.3 に示す。

リスト 4.3 の中で使用されている各種の変換関数は、どのアプリケーション・プログラムでも使用できる共通のリソースとして、リスト 4.4、リスト 4.5 のように変換テーブルとして定義されている

リスト 4.3 の中の

```
M14-CH16 at 6 xCernox correction C14458;
```

という定義により、M14-CH16 という名称のデータの物理量は、入力となる ADC 生データ用 SSTM のチャネル 6 のデータに対して、変換テーブル上の関数 xCernox を適用して抵抗値に変換し、その結果に変換テーブル上のスプライン補間関数 C14458 を適用して温度に変換して取得するということを表している。xCernox 関数は変換テーブル定義の中で

```
amp xCernox        (32768, 65536) "" -> (0.0, 0.10) Mohm ;
```

と定義されており、定電流源およびアンプのゲイン設定が、ADC の値 32768 が 0.0Mohm、ADC の値 65536 が 0.10Mohm になるように線型変換をおこなうことを表している。C14458 は変換テーブル定義の中で

```
spline C14458     Mohm -> K      include <CGR/C14458.tbl> ;
```

と定義されており、リスト 4.5 に示すセンサーの較正表から作成されたデータによりスプライン補間をおこなう。最終的に得られる温度のデータ名は M14-CH16 であり、アプリケーション・プログラムはこの名前を指定するだけで物理量を得られる。

以上のように物理量への変換をおこなう SSTM では、データを保持する機能だけでなく、変換演算をおこなう処理が重要となる。ここでは、この変換テーブル自体を SSTM とは独立に定義し、このテーブル自体を共用メモリ内に配置してどのプログラムからでも使用できるようにしてある。

```
#
M14-CH4 at 1 xCernox correction C14598;
# 10uA * ohm * 2 = V, ohm(at 5V) = 5/2/10uA = 250000
M14-CH6 at 2 xRuO2 correction RuO2;
M14-CH8 at 3 xCernox correction Cernox;
M14-CH10 at 4 xCernox correction Cernox;
M14-CH12 at 5 xCernox correction Cernox;
M14-CH16 at 6 xCernox correction C14458;
M14-CH20 at 7 xCernox correction C14626;
M14-CH24 at 8 xCernox correction C14647;
M15-CH4 at 9 xCernox correction C14455;
M15-CH6 at 10 xRuO2 correction RuO2;
M15-CH10 at 11 xCernox correction C14639;
M15-CH12 at 12 xRuO2 correction RuO2;
M15-CH16 at 13 xCernox correction C14252;
M15-CH20 at 14 xCernox correction C14396;
M15-CH24 at 15 xCernox correction C14401;
M16-CH4 at 16 xCernox correction C14399;
# 1kG = .859mV * 50 5V = 5/50/.859 * 1000 = 116.41kG
M17-CH22 at 17 xHall_1;
# 1kG = .849mV * 50   5V = 5/50/.849 * 1000 = 117.79kG
M17-CH24 at 18 xHall_2;
# 1kG = .745mV * 50 5V = 5/50/.745 * 1000 = 134.23kG
M18-CH2 at 19 xHall_3;
# 1kG = .738mV * 50 5V = 5/50/.738 * 1000 = 135.5kG
M18-CH4 at 20 xHall_4;
# ampx2
PI123 at 21 xPSI correction P94779B;
PI133A at 22 xPSI correction P94779A;
CH23 at 23 (32768,65536)->(0, 5) V;
PI133B at 24 xPSI correction P94779A;
CH25 at 25 (32768,65536)->(0, 5) V;
CH26 at 26 (32768,65536)->(0, 5) V;
CH27 at 27 (32768,65536)->(0, 5) V;
CH28 at 28 (32768,65536)->(0, 5) V;
CH29 at 29 (32768,65536)->(0, 5) V;
CH30 at 30 (32768,65536)->(0, 5) V;
CH31 at 31 (32768,65536)->(0, 5) V;
CH32 at 32 (32768,65536)->(0, 5) V;
```

リスト 4.3 **/opt/JK/etc/stm/ivcoil.channel** ファイルで
各チャネルに割り当てられた物理量の定義

```
dir "/opt/JK/etc/tbl" ;
#
# spline
#
spline Cernox       Mohm -> K          include <Cernox.tbl> ;
spline RuO2         ohm -> K include <RuO2.tbl> ;
spline C14252       Mohm -> K          include <CGR/C14252.tbl> ;
spline C14396       Mohm -> K          include <CGR/C14396.tbl> ;
spline C14399       Mohm -> K          include <CGR/C14399.tbl> ;
spline C14401       Mohm -> K          include <CGR/C14401.tbl> ;
spline C14455       Mohm -> K          include <CGR/C14455.tbl> ;
spline C14458       Mohm -> K          include <CGR/C14458.tbl> ;
spline C14598       Mohm -> K          include <CGR/C14598.tbl> ;
spline C14626       Mohm -> K          include <CGR/C14626.tbl> ;
spline C14639       Mohm -> K          include <CGR/C14639.tbl> ;
spline C14647       Mohm -> K          include <CGR/C14647.tbl> ;
spline P94779A      V -> PSI  include <CSPV/P94779A.tbl> ;
spline P94779B      V -> PSI  include <CSPV/P94779B.tbl> ;
#
# current source
#
amp xCernox         (32768, 65536) "" -> (0.0, 0.10) Mohm ;
amp xRuO2           (32768, 65536) "" -> (0.0, 250000) ohm ;
#
# hall
#
# 1kG = .859mV * 50 5V = 5/50/.859 * 1000 = 116.41kG
amp xHall_1         (32768, 65536) "" -> (0.0, 116.41) kG ;
# 1kG = .849mV * 50   5V = 5/50/.849 * 1000 = 117.79kG
amp xHall_2         (32768, 65536) "" -> (0.0, 117.79) kG ;
# 1kG = .745mV * 50 5V = 5/50/.745 * 1000 = 134.23kG
amp xHall_3         (32768, 65536) "" -> (0.0, 134.23) kG ;
# 1kG = .738mV * 50 5V = 5/50/.738 * 1000 = 135.5kG
amp xHall_4         (32768, 65536) "" -> (0.0, 135.5) kG ;
#
# pressure
#
amp xPSI (32768, 65536) "" -> (0.0, 2.5) V ;
amp x_1             (32768, 65536) "" -> (0.0, 5.0) V ;
amp x_2             (32768, 65536) "" -> (0.0, 2.5) V ;
amp x_5             (32768, 65536) "" -> (0.0, 1.0) V ;
amp x_10 (32768, 65536) "" -> (0.0, 0.5) V ;
amp x_20 (32768, 65536) "" -> (0.0, 0.25) V ;
(途中省略)
amp x_100mV         (32768, 65536) "" -> (0.0, 50.0) mV ;
amp x_200mV         (32768, 65536) "" -> (0.0, 25.0) mV ;
amp x_500mV         (32768, 65536) "" -> (0.0, 10.0) mV ;
amp x_1000mV        (32768, 65536) "" -> (0.0, 5.0) mV ;
```

リスト 4.4 コイル冷却表示に使用した変換テーブル

```
# C14458
(0.000218,   44.833)
(0.000229,   41.810)
(0.000242,   38.803)
(0.000258,   35.801)
(0.000278,   32.769)
(0.000295,   30.691)
(0.000312,   28.855)
(0.000329,   27.219)
(0.000348,   25.693)
(0.000370,   24.169)
(0.000396,   22.668)
(0.000432,   20.966)
(0.000457,   19.952)
(0.000485,   18.976)
(0.000517,   18.024)
(0.000554,   17.065)
(0.000597,   16.118)
(0.000651,   15.146)
(0.000716,   14.182)
(0.000798,   13.195)
(0.000905,   12.204)
(0.001048,   11.198)
(0.001248,   10.179)
(0.001594,    8.995)
(0.002060,    7.995)
(0.002844,    6.988)
(0.003934,    6.179)
(0.005600,    5.472)
(0.007616,    4.966)
(0.010090,    4.570)
(0.013517,    4.214)
(0.016324,    4.009)
(0.019901,    3.813)
(0.024750,    3.612)
```

リスト 4.5

センサー C14458 に対する変換テーブル

4.3 新型 AD 変換装置の開発

4.3.1 基本的な考え方

通常 ADC というとアナログ・データをデジタル・データに変換する AD 変換素子を含む装置全体を指すのが普通であるが、本節では、AD 変換素子を ADC と表記し、ADC を中心とする、計算機に接続してデータを取り込む装置を AD 変換装置と表記して区別する。

長時間運転に伴い、これまでバッチ処理をしていたデータに対してもリアルタイム処理することが要求されたので、リアルタイム性能を向上することがもっとも重要である。データ取り出しが高速(短時間にかつ多量に取り出せること)であれば、リアルタイムに取り出して主記憶に保存し、必要なときに処理することでバッチ処理に対応できる。

リアルタイム処理の高速化にも限度があることは明らかでなので、プラズマ計測やクエンチ時などの高速短期間データ収集などの用途には、データを蓄積しておき、あとで取り出すバッチ方式も可能にする。

ADC から計算機までの経路やデータ移動方式が複雑だと、データの移動に時間がかかり、性能上のボトルネックになる恐れがあるので、データ取り出しを高速化するためには、可能な限り単純な構成としなければならない。このために、PLA で作成した制御回路を計算機の入出力バスに直接接続する方式とした。データの移動はハードウエア論理回路の動作だけでおこなえる。図 4.3 に外付け新型 AD 変換装置の構造を示す。

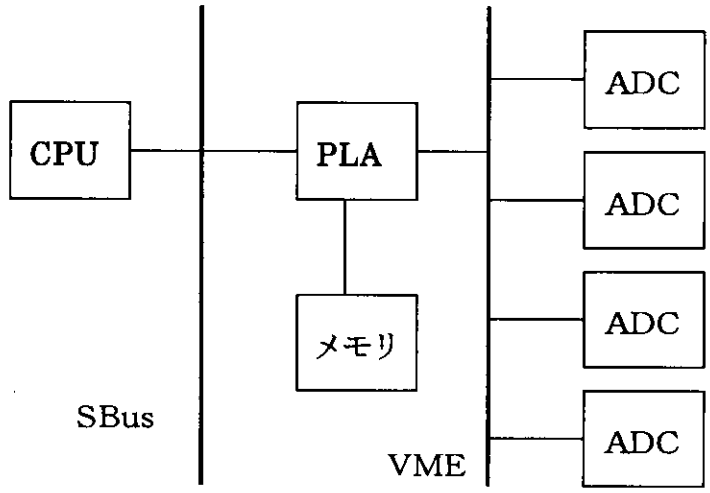


図 4.3 外付け新型AD変換装置の構造

ADC で AD 変換が終了すると、直ちに VME バスを介してメモリに送られる。アプリケーションがデータを読み込むときは、PLA に命令を発行し、メモリから CPU に転送するだけになる。

CAMAC に装備する ADC モジュールの場合、CAMAC のバスを制御する CAMAC 命令を実行するソフトウエアを実行して ADC モジュールからのデータ移動をしなければ

ならない。ソフトウエア制御が必要だと、表2.1に示した安らの[11]PIOでの性能のように、どうしても低速になってしまう。

4.3.2 リアルタイム性能向上のための構造

データ取り出し時間を短くするには、AD 変換が終了したらすぐにデータを取り出さなければならないので、可能な限り小さな単位で AD 変換装置から計算機にデータを取り込めるようにする必要がある。これまでの連続的にデータを取得できる AD 変換装置はダブル・バッファ方式である。ダブル・バッファ方式は図 2.1 のようにバッファを2面持ち、別々のバッファで書き込みと読み出しをおこなうことにより並行処理するものである。書き込みプロセスと読み出しプロセスとの同期を取ってバッファを切り替えなければならないため、ひんぱんに切り替えをおこなえず、大きな単位で切り替えをおこなっている。このようにこれまでのダブル・バッファ方式の AD 変換装置では、AD 変換装置から取り込むデータの大きさは任意にならず、大きな単位でデータを扱うため、AD 変換の終了待ちとデータの取り込みに時間がかかり、リアルタイムでのデータの取り込みはできない。そこで最小単位の1データでもデータを取り込めるように設計した。

小単位のデータにアクセスするために、図 4.4 のように AD 変換後のデータを貯える内蔵メモリをリング・バッファとし、バッファ内の任意のデータに対してアドレスを直接指定して取り出せる方式とした。さらに、後の節で説明するように、同一データに対してリアルタイム処理とバッチ処理をおこなうという要求を満たすため、複数のアプリケーションが同時に内蔵メモリから読み出せるようにした。図 4.4 で read index がふたつあるのは、ふたつのプロセスが読み出しをおこなっている状態を表している。

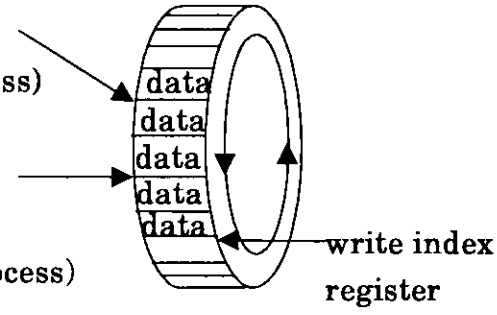


図 4.4 内蔵メモリの構造

通常 AD 変換装置をアクセスするプロセスはひとつなのに、内蔵メモリの構造をこのようにして、機能的には SSTM と同じ構造としたのは、SSTM と同じに見せかけることにより、SSTM と同様にデータ分配を容易にすることと、ソフトウエアを SSTM 用と AD 変換装置用で共通化するためである。この新型 AD 変換装置を共用短期記憶方式 AD 変換装置と呼ぶ。共用短期記憶方式 AD 変換装置では装置内に大容量メモリを持ち、ADC が変換を終えると直ちにこのメモリに書き込み write index を進める。アプリケーションはこの write index をモニタし、未読データがあれば何時でも読み出すことができる

ので、計算機の速度が速いなら、1回の AD 変換で得られる 1 チャンク(チャネル数分のデータのかたまり)のデータ毎に読み出して処理できる。このように ADC とアプリケーションの間には、データを受け渡すためのバッファ・メモリがあるだけなので、CAMAC などの外部機器に装備する AD 変換装置と異なり、ADC からアプリケーションまでの経路での遅延は物理的な伝送遅延だけとなる。したがってデータ取得に要する時間は、原理的には計算機命令の時間であるμ秒のオーダーとなるので、リアルタイム性能としてはこれ以上望めないレベルと考えられる。実質的にはアプリケーションがデータを取得するサイクルの周期がリアルタイム性能となり、OS のスケジューリングの単位である 10ms が現実的な性能となる。

短期記憶や write index を保持するレジスタは AD 変換装置内にある。UNIX のようなユーザ・レベルとカーネル・レベルが別れている計算機でこのような装置内のメモリをアクセスするためには、通常IO命令でデバイス・ドライバを呼び出すという方法を使う。IO命令を使うとカーネル機能を呼び出し、メモリ空間をユーザ空間からカーネル空間に切り替えるなどの操作が必要となり、わずか1データを取り出すために多くの命令を実行することが必要となり、重たい操作となる。これまでバッチ処理がおこなわれていたのは、このように小容量データを扱うコストが大きいということが理由のひとつであったと考えられる。

SunOS の場合はデバイス上のメモリをユーザ空間にマッピングする memory mapped I/O 機能があり、IO 命令を使わずに主記憶上のデータの read/write 命令でアクセスすることができる。共用短期記憶方式 AD 変換装置ではこの機能を使っているため、利用者が単純にデータを読み出すだけなら、SSTM と全く同じ使用方法となる。

AD 変換装置へのアクセスが SSTM と全く同じであることから、他に便利な利用法が現れる。ランダム・アクセスが可能なので、必要なデータだけを読み出すことができる。1チャンク毎に処理するのでは間に合わない場合は、一定量のデータが貯まってから一括処理することができる。このような利用方法ではリアルタイム処理とバッチ処理の区別は無い。ただ処理対象の大きさを目的と性能を考えて適切に決めれば良いだけである。短期記憶が大容量であり、過去データをアクセスできるため、イベント発生時にだけデータを取り込むというリアルタイム間欠処理も可能となる。このように、新型 AD 変換装置は、リアルタイム処理、バッチ処理およびリアルタイム間欠処理のいずれの方式でも使用でき、しかも同時におこなえる。

作成した共用短期記憶方式 AD 変換装置の性能は、次のとおりである。ADC の最大サンプリング・クロックは 100KHz で、同時サンプリング方式である。短期記憶からのデータ読み出しは1データ(2バイト)毎に SBus の 6 サイクルを必要とする。SBus のバス・クロックは SPARCstation 2 では 20MHz のため、短期記憶からの読み出し速度は 6.7MB/s (=2byte/(50ns*6)) となる。SBus の最大バス・クロックは25MHzなので[22]、

SPARCstation20 などの高性能のワークステーションを使用すると 8.3MB/s (=2byte/(40ns*6))まで性能が上がる。

上記の速度は理論最大値であり、使用する計算機の速度によって変わる。SPARCstation 2 で実測すると、短期記憶からアプリケーションにデータを読み込む速度は約 4MB/s であった。リアルタイム処理のため、AD 変換中にデータを取り込むと、短期記憶への書き込みと短期記憶からの読み出しが競合するので、2～2.5MB/s が AD 変換装置としての実質的なリアルタイム性能であった。

4.3.3 高速データでの利用方法

アプリケーション・プログラムの読み出し速度が ADC からの書き込み速度より高速であるならば、アプリケーション・プログラムは任意のデータ単位で読み出せるので、リアルタイム処理とバッチ処理の違いはないことは前節で説明した。したがって問題となるのは、ADC からの書き込み速度の方が高速の場合である。じっさい、リアルタイムでの読み出し速度の実効値を 2MB/s とすると、32 チャネルの製品では 31KHz 以上の高速データは読み出しが間に合わないことになる。

読み出しが間に合わない場合は、短期記憶が一杯になったら、もっと正確に言うと、write index register の値が read index register の値を追い越そうとしたらオーバー・フロー割り込みを発生させて AD 変換を停止し、蓄えられたデータを読み出すというバッチ方式で対処する。ADC を外付けにする拡張型 AD 変換装置では、短期記憶容量が最大 32MB なので、16 チャンネル 100KHz の場合で、最長 10 秒間のデータを蓄えることができる。

前節で述べたリアルタイム間欠処理方式では、ハードウエアあるいはソフトウエアによりイベントを検出し、イベント検出後一定のデータが蓄積された時点で AD 変換を停止し、イベント前後のデータを収集し、処理し終えたら再び AD 変換を開始してイベント待ちをするという使い方が自然な方法である。この方式はイベントごとのデータを一括処理するという意味ではバッチ処理だが、連続して発生するイベントを処理できるという意味ではリアルタイム処理となる。AD 変換速度が高速でなく、AD 変換を続けたままイベント発生時のデータを収集できる場合は、イベント待ちとイベント時のデータ収集処理を平行しておこなうことができる。

4.3.4 AD 変換装置によるデータ分配

ここではアプリケーション・プログラム間で共用する短期記憶を提供するしくみとしての AD 変換装置について詳細に説明する。

SSTM と同じ構造にすることにより、AD 変換装置で直接複数のプロセスにデータを分配できるようにした。しかし AD 変換装置には SSTM と異なる点がある。読み出しプロセスはオーバーフローが発生した場合の措置を AD 変換装置に伝えたり、割り込みが発生した時の write index の値を受け取ったりしなければならない。これらのデバイス・メモリ上にはない情報はカーネル内のデバイス・ドライバが保持している。このデバイス・ドライバが保持し、各プロセス間で共有することが必要なデータを利用しやすくするために、カーネル空間に置かれているデバイス・ドライバのデータ領域をユーザ空間にマッピングした。この結果、複数のプロセスがそれぞれ自分用の AD 変換装置があるかのようにプログラムすれば良くなり、ADC のデータを扱うプロセスを柔軟に組み込むことができ、共用可能な AD 変換装置となった。

このような工夫により複数プロセス間で共用する AD 変換装置としたが、ADC の実体は同一であり、変換周波数は同一、start/stop の制御も同一である。従って AD 変換装置を使うプロセスのうちひとつだけがマスター・プロセスとして AD 変換装置の物理的制御をおこなえ、他のプロセスはスレーブ・プロセスとしてデータの読み出しと割り込みなどの発生の通知を受けるだけとした。図 4.4 の read index のうち、ハードウエアのレジスタとしての read index はマスター・プロセスの read index だけであり、ハードウエアによるオーバーフローの検出はこの read index register についてのみおこなわれる。スレーブ・プロセスの read index は、スレーブ・プロセス固有のもので、スレーブ・プロセスのメモリ空間内に置かれている。

4.3.5 新型 AD 変換装置の製品化

以上の設計のもとに、筆者が計算機と AD 変換装置との間のインターフェースを決め、AD 変換装置を使用するためのデバイス・ドライバを作成した。FMC 社の芳村氏が回路設計と製造をおこない製品化した。

AD 変換装置は、計算機に内蔵する 12 ビット分解能で 8 チャネルの製品 AD70700 型(図 4.5)と、外付け筐体を使用する 16 ビット分解能で 32 チャネルから 256 チャネルまで拡張できる製品 AD71000 型(図 4.6)が作成された。AD70700 型の短期記憶メモリは 8MB、AD71000 型では最大 32MB であり、市販の AD 変

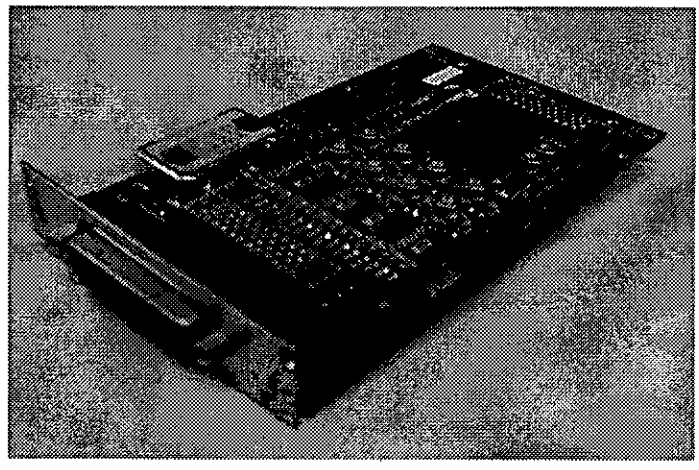


図 4.5 SBus 用内蔵 AD 変換装置

換装置としては抜きんでて大量のメモリを実装し、高速データのバッチ処理を可能としている。第5章での実験や、1997年に開発されたLHD制御データ処理装置ではAD71000型が使用されている。(LHD制御データ処理装置でのAD変換装置のデバイス・ドライバは日本サン・マイクロシステムズ社が作成した。)AD71000型のカタログ仕様を表4.1に示す。

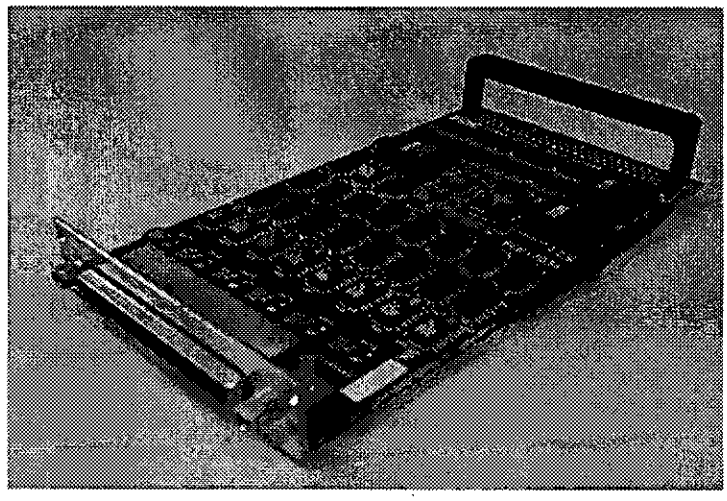


図 4.6 SBus用外付AD変換装置

表4.1 AD71000のカタログ仕様

A/D分解能	16ビット(14ビット精度)
トータルA/D変換速度	32チャンネル:3.2MHz、16チャンネル:1.6MHz
A/D変換方式	逐次比較型
サンプリング速度	10μs～2μs×65535(16ビット)
入力チャンネル	16、32チャンネル(シングルエンド型)
入力電圧	±10V、±5V、0～5V、
入力インピーダンス	10MΩ、15pF
最大入力電圧	±30V連続
スタートトリガー仕様	TTL立ち上がりまたは立ち下がり
信号入力接栓	BNCコネクタまたはアンフェノール24ピンコネクタ
ワークステーション	SPARCstation 2,..., SPARCstation 20
OS	SunOS4.1.X、Solaris2.3
インターフェース	SBus
転送速度	8MB/s(SBusスレーブ)
内蔵バッファメモリ	16MBまたは32MB
拡張	8ユニットまで

4.3.6 今後の改良点

多様な利用形態に対応でき、転送速度も高速である AD 変換装置を開発できた。さらに性能を向上する方法や、LHD での設置条件などに対応し、可用性を向上させるための考察をおこなう。

入出力バスとして PCI バスが普及してきたので、PCI バス用の AD 変換装置とする。PCI バス用だと、SPARCstation と DOS/V 互換パソコンの両方で使用できる。バスの速度が速いため、SBus 用よりも高速になる。

LHD ではセンサーが電磁気的悪環境にあるため、ADC もこの環境で使用できるものにしたい。そのためには外付け装置からのデータ転送を、光ファイバーを用いる方式とする。

リアルタイム読み出しの実効値を 2MB/s とすると、128 チャネルのデータをリアルタイムに読み込めるのは 7.8KHz までである。多チャンネルの場合は1回のデータ転送量が多くなるので、DMA(Direct Memory Access)を使うと PIO を使う方式より高速になる可能性があるので、DMA 方式を実現する。

AD 変換装置から計算機にデータを転送し、つねに計算機がネットワークにデータを転送するなら、AD 変換装置から直接ネットワークにデータを転送する。

4.4 リアルタイム表示

各種のリアルタイム表示プログラムを作成する基となるようにスクロール・グラフ表示プログラムを作成した。特に表示を高速化するための工夫をおこなっている。

4.4.1 スクロール・グラフ表示プログラムの基本設計

多数の測定値をリアルタイムに把握するため、柔軟に表示対象、表示方法を選択できるグラフ表示が必要である。グラフ表示のように GUI(Graphical User Interface)を扱う部分は開発に大きな労力を必要とするので、可能ならば市販の製品を使用するべきである。市販製品あるいはフリーウエアとして、DaDisp、GnuPlot、PvWave、XRT/graph、S、LabVIEW を調査したが、リアルタイム表示をおこなえるものは少なく、できても不十分な性能でしかなかったので、独自開発することにした。プログラムはかなり大量となるので、ここでは内容の説明はしない。以下に設計思想と結果を述べる。

GUI を含むプログラム開発には労力を要するので、モジュール化により再利用できる部分を大きくしたい。そこで全体を垂直分割し、3階層で作成した。ウィンドウを図 4.7のように各部に分け、最下層のライブラリは X Window のライブラリである Xlib の関数を直接使用してスクロール・グラフ表示に必要な基本的な機能を提供する。このライブラリは Xlib にのみ依存するので、上位ライブラリを OpenWindow 用に開発しても Motif 用に開発してもこのライブラリを使用できる。このライブラリのなかで代表的な機能を表 4.2 に示す。

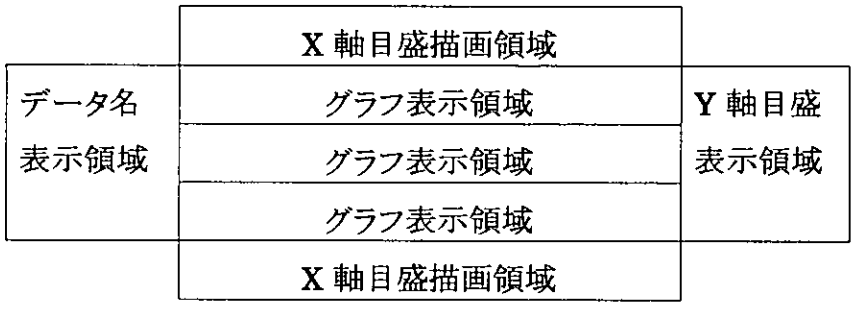


図 4.7 描画領域区分

表 4.2 スクロール・グラフ最下層ライブラリの機能

関数名	機能
x_scg_new	グラフ数を与えて scg 制御データを作成する
x_scg_set_window	Xlib のウインドウとその使用枠を設定する
x_scg_set_data_size	表示データ数と追加表示領域を設定する
x_scg_set_min_max	グラフの最小値と最大値を設定する
x_scg_setup_ok	設定の不足を補い、設定値をチェックする
x_scg_set_yaxis_width	グラフと y 軸の幅を設定する
x_scg_set_xaxis_height	グラフと x 軸の高さを設定する
x_scg_set_rescale	x 軸の目盛属性を設定する
x_scg_set_xaxis_tick_len	x 軸の目盛の長さを設定する
x_scg_set_yaxis_tick_len	y 軸の目盛の長さを設定する
x_scg_set_xaxis_tick_index	x 軸の主目盛と副目盛を表示するデータ番号を設定する
x_scg_set_xaxis_label_offset	x 軸のラベルの表示位置を設定する
x_scg_set_yaxis_label_offset	y 軸のラベルの表示位置を設定する
x_scg_set_unit	y 軸の単位を設定する
x_scg_set_graph_foreground	グラフの表示色を設定する
x_scg_set_xaxis_foreground	x 軸の表示色を設定する
x_scg_set_yaxis_foreground	y 軸の表示色を設定する
x_scg_all_clear	全グラフをクリアし、軸と目盛のみ表示する
x_scg_clear	グラフをクリアし、軸と目盛のみ表示する
x_scg_get_scroll	グラフをスクロールする
x_scg_get_x	データ番号から x 方向の絶対表示位置を取得する
x_scg_get_y	グラフのデータ値から y 方向の絶対表示位置を取得する
x_scg_setup_range	レンジを再計算し、目盛を設定し直す
x_scg_plot	最終ポイントから新ポイントへのグラフを書く
x_scg_point_begin	複数点プロットの開始を指示する
x_scg_point	複数点プロットの位置を与える
x_scg_point_list	複数点プロットをおこなう
x_scg_draw_axis	x, y 軸を書く
x_scg_draw_xaxis	x 軸を書く
x_scg_draw_x_major_all	全グラフの x 軸主目盛を書く
x_scg_draw_x_major	指定したグラフの x 軸主目盛を書く
x_scg_draw_x_minor_all	全グラフの x 軸副目盛を書く
x_scg_draw_x_minor	指定したグラフの x 軸副目盛を書く

表 4.2 の機能は上位のグラフ・ライブラリ開発者のための機能であり、利用者が使用する利用者ライブラリはさらにこの上のレベルで作成されている。画面の配置や描画の制御などを行う利用者ライブラリが中間層のライブラリである。同一画面を使用し、扱うデータの単位が違うなどの応用も考えられる。このような場合にライブラリではなくプログラムそのものを再利用するのが最上位の階層となる。

利用者ライブラリは最下層ライブラリの機能を使ってさらにまとまった単位の描画機能を提供する。またスクロール・グラフ表示以外への応用も考え、データ定義入力などの機能も提供している。この利用者ライブラリの上に作成するアプリケーションとしてスクロール・グラフ表示の他に次のようなものが予想される。

- デマンド・グラフ表示
  スクロール・グラフで高速データを表示すると、グラフの動きに目が追いつかない。また更に高速になると計算機の表示が追いつかない。このような高速データを表示するには、表示要求があったとき、あるいは周期的に1画面分だけ表示する。スクロール・グラフ表示とは、表示するデータの取得方法だけしか違わないので容易に作成できる。
- ファインダー
  4.6 節で説明する事後解析で使用する review は、いろいろな用途に使用できるが、表示用途としてはもっと操作性を良くする必要がある。ファインダーでは、指定したファイルの範囲全体を1画面に表示することにより、対象範囲をすばやく絞れるようにする。またこのときに、サンプリングした値だけでなく、1点に簡約する区間の最大値・最小値も一緒に表示すると、対象範囲を見つけやすくなる。この機能は review がおこなっているファイル上の範囲指定の機能をスクロール・グラフの機能の中に取り込んで作成できる。

4.4.2 スクロール・グラフの使用方法

スクロール・グラフは

```
scg21 ivcoil.scg ivcoil.stm &
```

のように起動する。ivcoil.stm は 4.2 節で説明した SSTM のファイル名であり、ivcoil.scg はリスト 4.6 に示すグラフ表示方法の定義ファイル名である。

```
graph size (400,80);
show data (400,10);
change rate (25,10,400);
time at (180,60);
polling (200,30);
initial view 1;
#
include   <ivcoil.channel> ;
#
view 1
M14-CH4 AA (-0.0001, 0),
M14-CH6 AA (-0.0001, 0),
M14-CH8 AA (-0.0001, 0),
M14-CH10 AA (-0.0001, 0),
M14-CH12 AA (-0.0001, 0),
M14-CH16 AA (-0.0001, 0),
M14-CH20 AA (-0.0001, 0),
M14-CH24 AA (-0.0001, 0);

view 2
M15-CH4 AA (-0.0001, 0),
M15-CH6 AA (-0.0001, 0),
M15-CH10 AA (-0.0001, 0),
M15-CH12 AA (-0.0001, 0),
M15-CH16 AA (-0.0001, 0),
M15-CH20 AA (-0.0001, 0),
M15-CH24 AA (-0.0001, 0),
M16-CH4 AA (-0.0001, 0);

view 3
M17-CH22 AA (-0.0001, 0),
M17-CH24 AA (-0.0001, 0),
M18-CH2 AA (-0.0001, 0),
M18-CH4 AA (-0.0001, 0),
PI123 AA (-0.0001, 0),
PI133A AA (-0.0001, 0),
CH23 AA (-0.0001, 0),
PI133B AA (-0.0001, 0);

右に続く
```

```
続き

途中省略

view 6
CH41 AA (-0.0001, 0),
CH42 AA (-0.0001, 0),
CH43 AA (-0.0001, 0),
CH44 AA (-0.0001, 0),
CH45 AA (-0.0001, 0),
CH46 AA (-0.0001, 0),
CH47 AA (-0.0001, 0),
CH48 AA (-0.0001, 0);

view 7
CH49 AA (-0.0001, 0),
CH50 AA (-0.0001, 0),
CH51 AA (-0.0001, 0),
CH52 AA (-0.0001, 0),
CH53 AA (-0.0001, 0),
CH54 AA (-0.0001, 0),
CH55 AA (-0.0001, 0),
CH56 AA (-0.0001, 0);

view 8
CH57 AA (-0.0001, 0),
CH58 AA (-0.0001, 0),
CH59 AA (-0.0001, 0),
CH60 AA (-0.0001, 0),
CH61 AA (-0.0001, 0),
CH62 AA (-0.0001, 0),
CH63 AA (-0.0001, 0),
CH64 AA (-0.0001, 0);
```

リスト **4.6** グラフ表示定義

リスト 4.6 の定義について説明する。「graph size (400,80);」は、ひとつのグラフが占める表示領域の大きさが 400×80 ピクセルであり、「show data (400,10);」は1画面に 400 データ、つまり 1 データを 1 ピクセルで表示し、10 データ毎にスクロールすることを指示する。画面の大きさはすべてのデータについて共通である。「change rate(25,10,400)」はオート・レンジのパラメータの設定である。表示しているレンジを超えるデータが入ると直ちにレンジは拡大する。縮小は 400 点のデータについてレンジを計算し、データのレンジが表示しているレンジより 25%以上小さければ表示するレンジを 10%縮小することを指示している。一度に最大限に縮小するとすぐに拡大しなければならない可能性が高くなるので、徐々に縮小する。「time at (180,60);」は、180 データごとに時刻を表示し、60 データごとに目盛だけを表示することを指示する。「polling (200,30)」は、200ms ごとにポーリングをおこない、1 回の処理で最大 30 点表示することを指示する。「include <ivcoil.channel>」は、リスト 4.3 のチャネル割り当て定義ファイルを読み込む。

スクロール・グラフは1画面で 8 チャネルのデータを表示でき、この1画面を view と呼ぶ単位として複数の view を定義できる。「view 1」以下の 8 行が第 1 画面に表示される 8 データであり、「M14-CH4 AA (-0.0001, 0)」は、データ名が M14-CH4 で上下限共にオート・レンジで上下限の初期値が (-0.0001, 0) であることを表わしている。同様に第 8 画面まで定義している。

この定義で表示した画面で view1 を表示し、設定変更をしようとしているところを図 4.8 に示す。左上の CTL ボタンを押すと、上部に表示されている CONTROL ポップアップ・ウィンドウが表示される。この VIEW NO 数値フィールドに画面番号を入力して OK ボタンを押すと、指定した画面に切り替わる。view はいくつでも定義でき、同一データを何回でも指定できるので、いろいろな組み合わせの画面を定義しておくと、データの値を比較しやすい。各グラフのデータ名ラベルの上に付けられた番号ボタンを押すと、下部に表示されている VIEW ポップアップ・ウィンドウが表示される。この画面でグラフごとのレンジの指定や色の指定ができる。

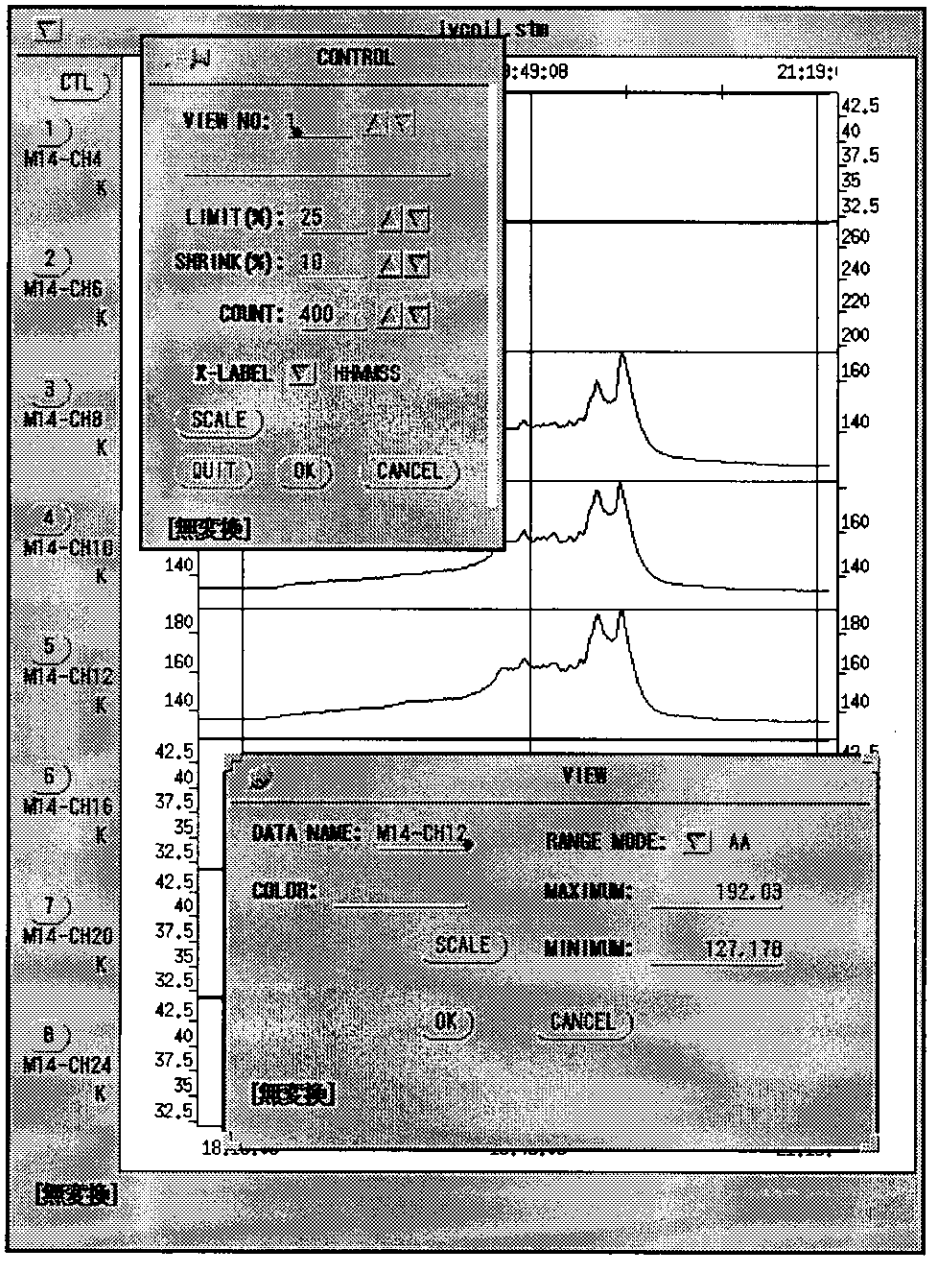


図 4.8 スクロール・グラフ

4.4.3 高速表示の方法と結果

リアルタイム表示をおこなうため、受信したデータを直ちに描画しなければならない。このときに1データの受信ごとにグラフをスクロールすると、計算負荷が高く表示しきれなくなる。このためにグラフ右端に追加領域という短い領域を設け、受信したデータのグラフはこの領域に書き加える。グラフがこの領域の右端に達したら、この領域の幅だけスクロールする。

リスト 4.6 の定義「show data(400,10)」は、400 点のグラフ領域のうち、10 点がこの追加領域であることを指定している。これで 10 データを受信するごとにスクロールするので、1 データの受信ごとにグラフをスクロールする場合に比較して、スクロールの頻度が 1/10 になる。

追加領域が少ないと負荷軽減につながらない。一方追加領域が大きいとスクロール量が大きくなり、スクロールが滑らかに見えない。状況に合わせて追加領域の大きさを決める必要がある。

この方式により表示性能がどの程度になっているかを調べるため、ADC のサンプリング速度を変えて、目視により正しく表示されているかどうかを調べた。サンプリング速度が 100Hz を超えると、波形を目視により把握することがむずかしかった。サンプリング速度を100Hz、追加領域を 10 点とすると、スクロールは毎秒 10 回発生するので、データの発生は 100Hz であるが、波形は 100ms の間静止しているため容易に認識できた。スクロールする場合も 10 点の移動なので、スクロール前後の対応関係も見やすかった。

100Hz のデータを切れ目なく連続して表示できたので、1チャンクのデータ表示に必要な計算時間は最大 10ms であるとして良い。

4.5 データ伝送系

4.5.1 データ伝送系の役割

データ伝送系の機能は、AD 変換装置、SSTM、ネットワーク、ファイルなど、データ・パケットの扱いが定められているものの間でデータ・パケットを送受信するなど、データの内容に無関係な処理のサービスを提供することである。このようにデータ伝送系の役割を共通サービスに限定することにより、あらかじめサービスを用意しておくことができる。データの内容に依存する処理は、それぞれのデータを扱う利用者あるいはサブ・システムの役割とする。図 4.1 にはこのデータ伝送系の主なサービスが番号で示されている。サービスは伝送系と他の系との間ではライブラリとして、伝送系の内部ではアプリケーションとして提供される。

それぞれの計算機、あるいは計算機上のアプリケーションは、このサービスを組み合わせて必要なデータを取得できるので、アプリケーション・プログラムの作成時には、システム全体についてはもちろん、データ送信元に関する知識も必要としない。

この伝送系を確立することにより、表示プログラムなどのアプリケーションを作成することと、システムを構築することを独立させることができる。伝送系が提供するサービスはそれぞれ独立しているので、必要に応じて新しいサービスを提供すれば良い。本節ではプロトタイプ・システムを動作させるために作成した、アプリケーションとして提供するサービスの中から代表的なものについて説明する[2]。

4.5.2 データの送受信

SSTM からの送信

```
stm_to_mcip -f<SSTM 名> -p<ポート番号>
```

図 4.1 の(2)のサービスである。SSTM からデータ・パケットを読み出して<ポート番号>の IP マルチキャストで送信する。SSTM に送信するデータがなければ待ち合わせするので、時間分解能は 10msec となる。

---

[2] AD 変換装置からのデータ取得やファイルへの保存などは、伝送系の範囲ではないがプロトタイプ・システムに必要なため作成した。これらはデータ内容によらない独立アプリケーションとして提供される。

SSTM への受信

```
mcip_to_stm -f<SSTM 名> -p<ポート番号>
```

図 4.1 の(3)のサービスである。<ポート番号>の IP マルチキャスト・パケットを受信して SSTM に書き込む。常に受信待ちをしているので、受信後直ちに書き込むことができる。

ユニキャスト・データ送信

```
stm_to_tcp -f <SSTM 名> -a <計算機名> -p<ポート番号>

stm_to_udp -f <SSTM 名> -a <計算機名> -p<ポート番号>
```

図 4.1 の(4)のサービスである。SSTM のデータ・パケットを、TCP または UDP を使って遠隔監視用の計算機に送信する。

ユニキャスト・データ受信

```
tcp_to_stm -f <SSTM 名> -P<ポート番号>

udp_to_stm -f <SSTM 名> -P<ポート番号>
```

図 4.1 の(5)のサービスである。TCP または UDP で受信したデータ・パケットを SSTM に書き込む。

4.5.3 データの保存

ADC と SSTM のデータ取得方法ほぼ同じなので、プログラムもほとんど同じとなる。ADC データの保存は図 4.1 の(6)、SSTM のデータ保存は図 4.1 の(7)のサービスである。

ADC データの固定長データ保存

```
adc_to_disk -f<ファイル名> -b<レコード長> -n<レコード数> -v<ADC デバイス>
```

<ADC デバイス>から指定した<レコード長>の ADC データを<レコード数>回指定した<ファイル名>のファイルに保存する。レコード長の単位はバイトではなく、チャンク数なので、レコードのバイト長はチャネル数により変わる。

チャネル数が 128 ならば、レコード長を 10 とすると、2560 バイト(=128 チャネル×2 バイト×10 レコード)がレコードのバイト長となる。高速データをバッチ処理のために保存するときに使用する。

ADC データの指定長保存

wadc_pre_stop

tadc_pre_stop

ADC データの保存を会話的におこなう。PRE TRIGGER で保存開始命令以前に AD 変換したデータを保存する量を指定する。保存終了も会話により指示する。テキスト・ベースの tadc_pre_stop とウィンドウ・ベースの wadc_pre_stop を作成した。wadc_pre_stop の設定画面を図 4.9 に示す。START ボタンが押されると、その直前のデータを PRE TRIGGER で指定したレコード数だけ取り込み、その後 STOP ボタンが押されるまでデータ取得を続ける。この画面で ADC は ADC のデバイス名、DIRECTORY はデータ保存場所、MODE はマニュアルか自動の切替えであり、File は保存ファイル名の指定である。CHANNEL はチャンネル数、BLOCK SIZE はサンプル数、CLOCK はサンプリング周波数である。

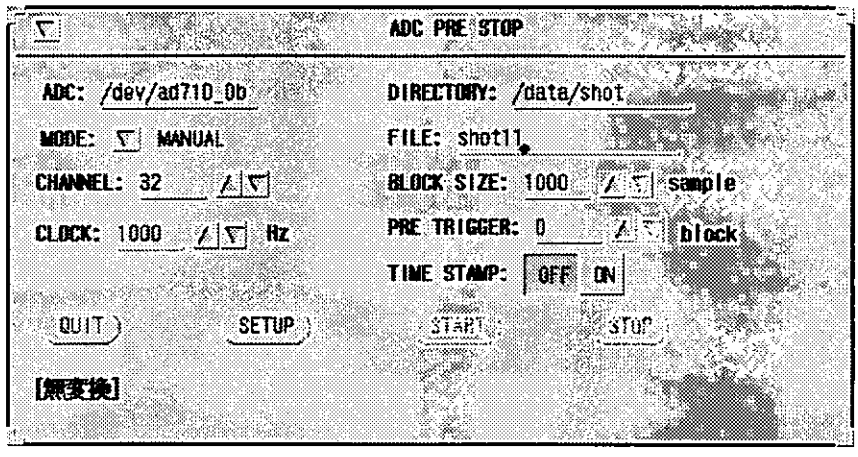


図 4.9 wadc_pre_stop 設定画面

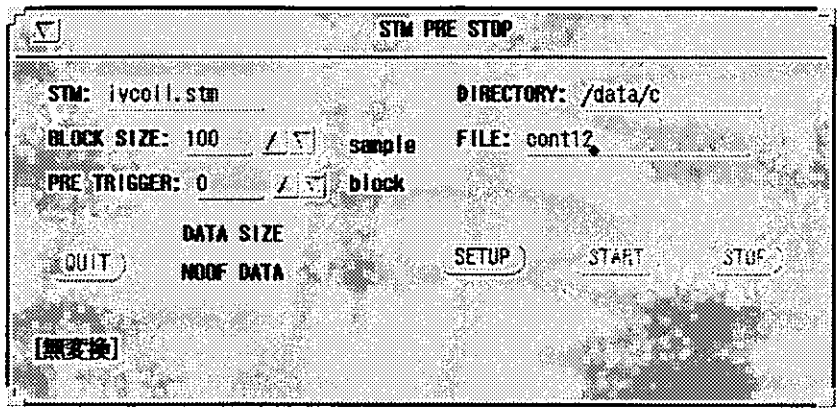


図 4.10 wstm_pre_stop の設定画面

ウィンドウ・ベースの場合は Xwindow を使用するため、収集用ワークステーションの負荷は増大し、通信も多くなるので、高速の計算機が必要となる。

wadc_pre_stop では、ファイル名など必要な事項は画面入力で指定する。図 4.9 の画面は、AD 変換装置のスレーブ・プロセスとして 1KHz のデータを収集するところである。

SSTM データの指定長保存

wstm_pre_stop

AD 変換装置と SSTM はデータ取得方法としては同じなので、wadc_pre_stop に対応するものとして wstm_pre_stop がある。この画面を図 4.10 に示す。利用方法が同じなので、画面も統一してある。STM は SSTM を意味し、ivcoil.stm は共用メモリーを扱うためのファイル名である。

分割ファイル保存

```
stm_to_mfile -f<SSTM名> -d<ファイル名フォーマット> -b<レコード長> -z<レコード数>
```

SSTM のデータをファイルに連続して長期間保存するときに、データがあまりにも大量だと扱いにくいので、一定量のデータを保存するごとにファイルを切り替える。<レコード長>のデータを<レコード数>出力するとファイルを切り替える。ファイル名は<ファイル名フォーマット>中の%s をファイル作成日付 YYMMDD-HHMMSS にして作成する。

4.5.4 時間スケールの変換

平均値によるノイズ除去

```
adc_mean_stm -v/dev/ad710_0c -c<チャネル数> -n<出力間隔> -m<計算データ数> -
f<SSTM名>
```

絶縁アンプ[14,15]のノイズ対策のため、ADC データの平均値を計算して SSTM に書き込む。<出力間隔>ごとに<計算データ数>のデータの平均値を計算する。これらが共に 1000 だとすると、1000 データごとに 1000 データを平均した値を出力するので、結果として得られるデータのサンプリング・レートは 1000 分の 1 になる。時間スケールを変えない移動平均にするには<出力間隔>を 1 にする。このサービスは図 4.1 の(8)に示されている。このように中間的な処理をおこなうときは SSTM を出力先とする。

時間単位の変更

```
stm_cull2 -f <入力SSTM> -t <出力SSTM> -n<間引き率>
```

<入力 SSTM>から入力したデータ・パケットを 1/<間引き率>に間引いて<出力 SSTM>に出力する。定期的データ・パケットならば時間単位が<間引き率>倍に伸びる。このサービスは図 4.1 の(9)に示されている。このように SSTM 間の処理にしておくと、簡単に処理を組み合わせることができる。

4.5.5 データの統合

データが複数の SSTM 上にあると、それぞれの SSTM は独立した時間で制御されているので、複数の SSTM にまたがった処理を記述するには時刻あわせが必要となり、プログラムが難しくなる。そこで、アプリケーションが一緒に扱うと分かっているデータについては、あらかじめデータ伝送系の中で統合したものを供給する。データの統合は図 4.1 の(10)に示されている。図のように待ち合わせが必要なデータは SSTM に格納される。

データ統合に必要なデータとしてネットワークによるデータの分配により供給されるデータを使用すると、データの到着は入力イベントとして通知されるので、ネットワークの通信能力に応じた速度で統合データを供給できる。また統合されたデータをネットワーク上に送信すると、統合後のデータを複数の計算機に分配できることになる。

もっとも簡単に使用できるインタプリタ方式のデータ統合用として、プログラム merge を作成した。merge に統合の内容を定義したファイルを与えるだけで統合できる。リスト 4.7 にその定義ファイルを示す。

```
SYNC ANY ALL;
IN    p0    MCIP 1024 32 ;
IN    p1    MCIP 1025 32 ;
OUT   px    MCIP 1100 32 ;
%%
MOVE px(0) = p1(0);
MOVE px(1) = p0(0);
MOVE px(2) = p1(1);
MOVE px(3) = p0(1);
MOVE px(4) = p1(2);
MOVE px(5) = p0(2);
MOVE 8 px(6) = p1(3);
MOVE 8 px(14) = p0(3);
RESERVE 10 px(22);
```

リスト 4.7 データ統合定義ファイル

先頭行の

```
SYNC ANY ALL;
```

は、同期方式を指定する文で、ANY はどのマルチキャスト入力も統合タイミングとなることを示す。

何かイベントが発生したときに通知される時間的にランダムなデータについて、どれかが変動したことを通知するような場合に使用する。

```
SYNC p1;
```

のように特定の入力を指定した場合は、指定した入力のデータ・パケットを受信したときに統合をおこなって出力する。また

```
SYNC 500msec;
```

のように時間を指定すると、この時間間隔で統合する。

ALL は、すべての入力が更新されたときに出力する指定であり、**ANY** と **ALL** が指定されると最も転送レートの低い入力に同期して出力されることになる。

次の行の **IN**、**OUT** は、入力パケットと出力パケットの指定であり、

```
IN   p0   MCIP 1024 32 ;
```

は、**p0** と名づけた入力は **IP multicast** が入力でそのポート番号は **1024**、データ長は **32** チャネルであることを表している。

MOVE 命令は入力パケット上のどの位置のデータを出力パケットのどの位置に移すかという指定である。**RESERVE** 命令は使用しない出力パケット上のデータを指定するもので、定義忘れを防止するために指定する。

データ項目数が多くなりインタプリタ方式では統合速度が不足するような場合はジェネレータ方式を使う。この方式ではリスト **4.8** のようにデータ移動を指定する部分にプログラムそのものを記述する。データ移動以外の部分はインタプリタ方式のリスト **4.7** と同じである。この定義からデータ統合を行うソース・プログラムを生成し、コンパイル、リンクして統合をおこなう実行形式プログラムを作成する。

```
SYNC ANY ALL;
IN    p0    MCIP 1024 32 ;
IN    p1    MCIP 1025 32 ;
OUT   px    MCIP 1100 32 ;
%%
#include <memory.h>
%%
px(0) = p1(0);
px(1) = p0(0);
px(2) = p1(1);
px(3) = p0(1);
px(4) = p1(2);
px(5) = p0(2);
memcpy(&px(6), &p1(3), 16);
memcpy(&px(6), &p0(3), 16);
```

リスト 4.8 ジェネレータ方式でのデータ統合定義

4.6 再生制御

実験中に保存したデータを処理する事後解析ツールは本プロトタイプ・システムの範囲外であるが、作成したシステムの動作確認に有効なツールとして、オフラインのデータ伝送サービスとも言える再生制御プログラム review を作成した。review は図 4.1 の(11)に示すように、DISK から入力する、リアルタイムではないサービスである。

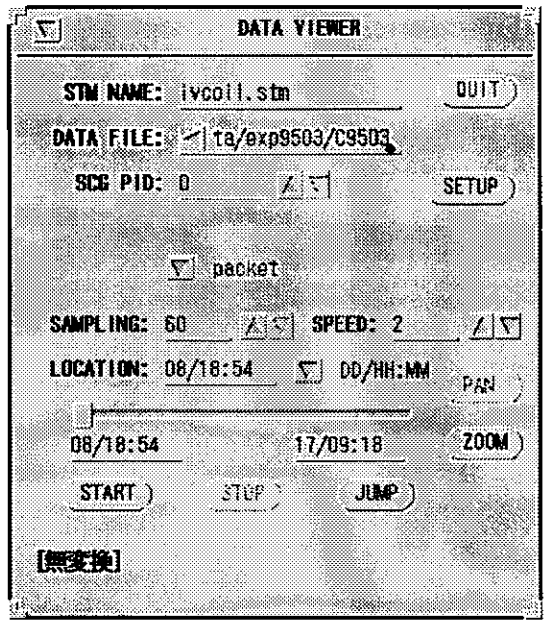


図 4.11 review 初期設定画面

review コマンドを使うと、保存した raw 型データ・ファイル、あるいはマルチキャストされたリアルタイム・データを保存したパケット型データ・ファイルからデータを再生できる。review の起動画面を図 4.11 に示す。ここで、SCG PID はスクロール・グラフのプロセス ID であり、SPEED は送信レートであり、10Hz の整数倍である。他の設定用語は今までの例に準じる。

review の基本機能は、スライダーで指定した範囲のデータを、SAMPLING で指定した間引きをおこないながら、SPEED で指定した速度で、SSTM に出力することである。SPEED は最低速度 10Hz の倍数を指定する。図のように SPEED が 2 なら 20Hz を意味する。初期状態ではファイル全体を出力対象範囲とするが、現在位置あるいは STOP ボタンで停止した位置の周囲で、ボタンを使って対象範囲を広くあるいは狭くできる。また LOCATION で直接位置を指定することもできる。範囲を指定し JUMP ボタンを押すと、指定範囲の先頭からデータ出力を再開する。

事後解析に使用する範囲を選択するために、review コマンドが出力する SSTM のデータを、スクロール・グラフを使って表示する。図 4.12 は、図 4.11 の設定でスクロール表示している途中の画面である。保存されているデータは 32 チャネル、30 秒ごとのデータであり、SAMPLING を 60 に指定しているので、表示上は 1 点が 30 分となり、画面上には約 1 週間、10 日 12 時から 17 日 24 時までの期間のデータが表示されている。データに測定時刻がついているパケット型なので、画面の X 軸に測定時刻が表示されている。

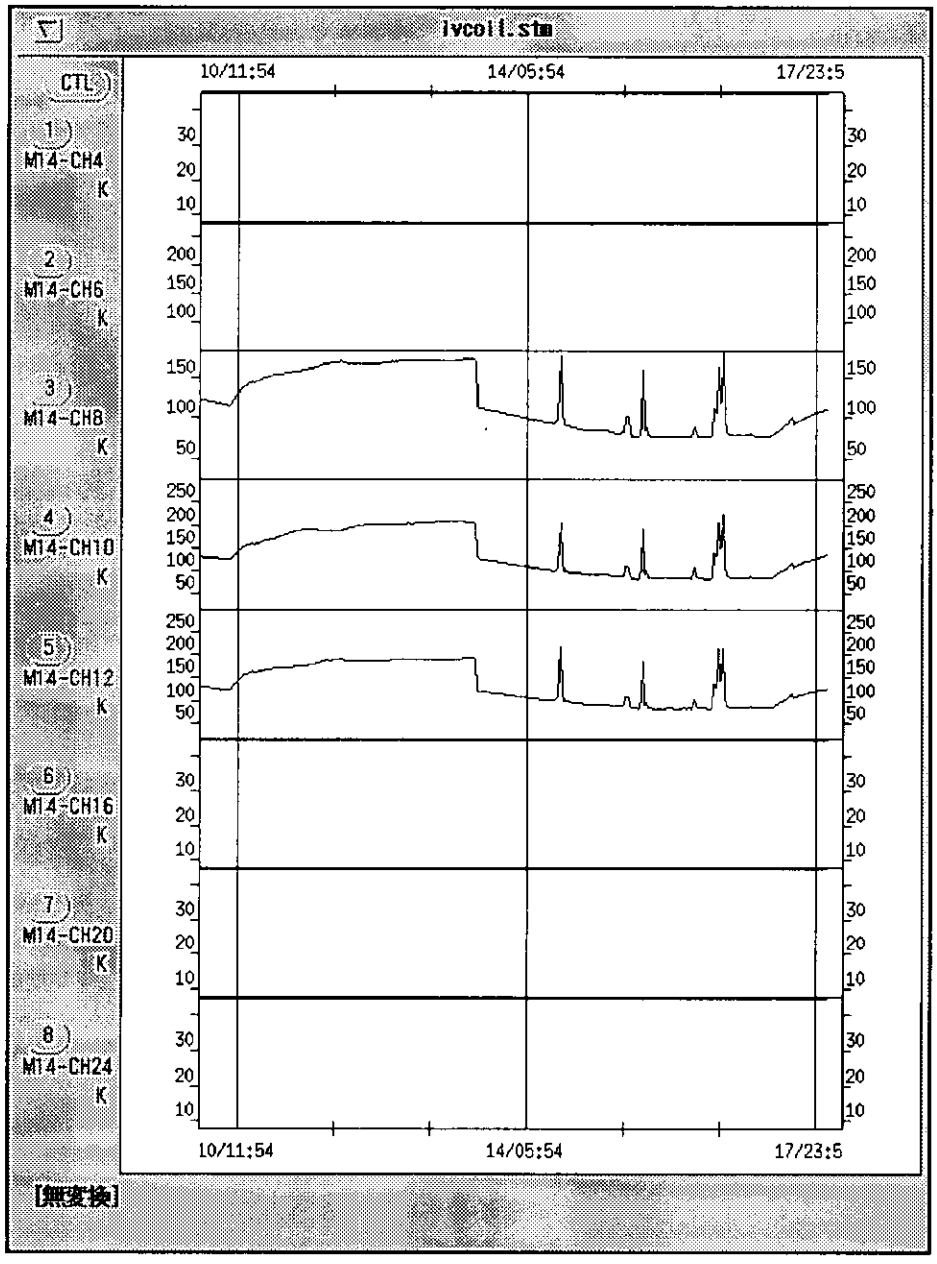


図 4.12 review によるデータ表示

図 4.12 の一番右のピークを観察するために、図 4.13 のように LOCATION を 15 日に設定し、その前後を出力する設定をしている。図 4.12 の 6 倍の分解能で表示するため SAMPLING を 10、すなわち 1 点を 5 分に設定している。この設定で JUMP ボタンを押すと、図 4.14 のように指定したサンプリングでの表示が得られる。拡大したので、見たいピークの構造が見えてきている。この操作を繰り返すことにより、解析対象部分を決定する。

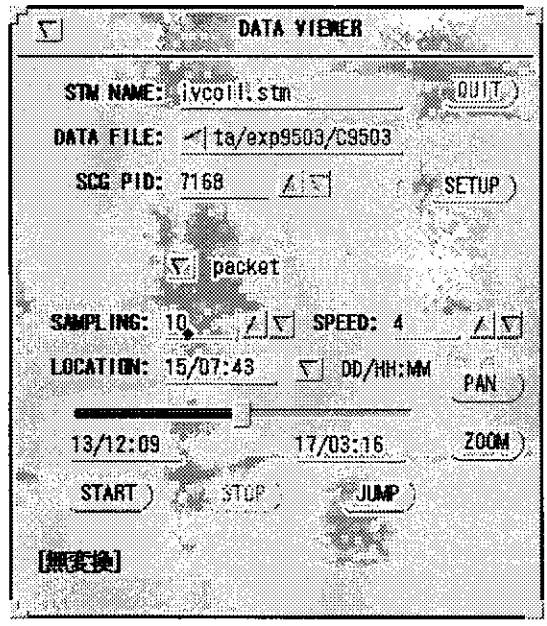


図 4.13 注目部分への位置づけ

review コマンドは選択したデータを定期的に SSTM に出力するだけなので、データ送信コマンドで SSTM のデータをネットワークにマルチキャストすれば、本番の実験を再現できることになり、システムのテストを行える。このときに、再生速度を変更すれば、システムがどの程度の速度までリアルタイムに使用できるかを評価できる。図 4.13 の SPEED はこのための設定で、0.1 秒あたりの出力データ数を設定する。最低 10Hz から計算機の最大性能までの速度でデータを出力できる。

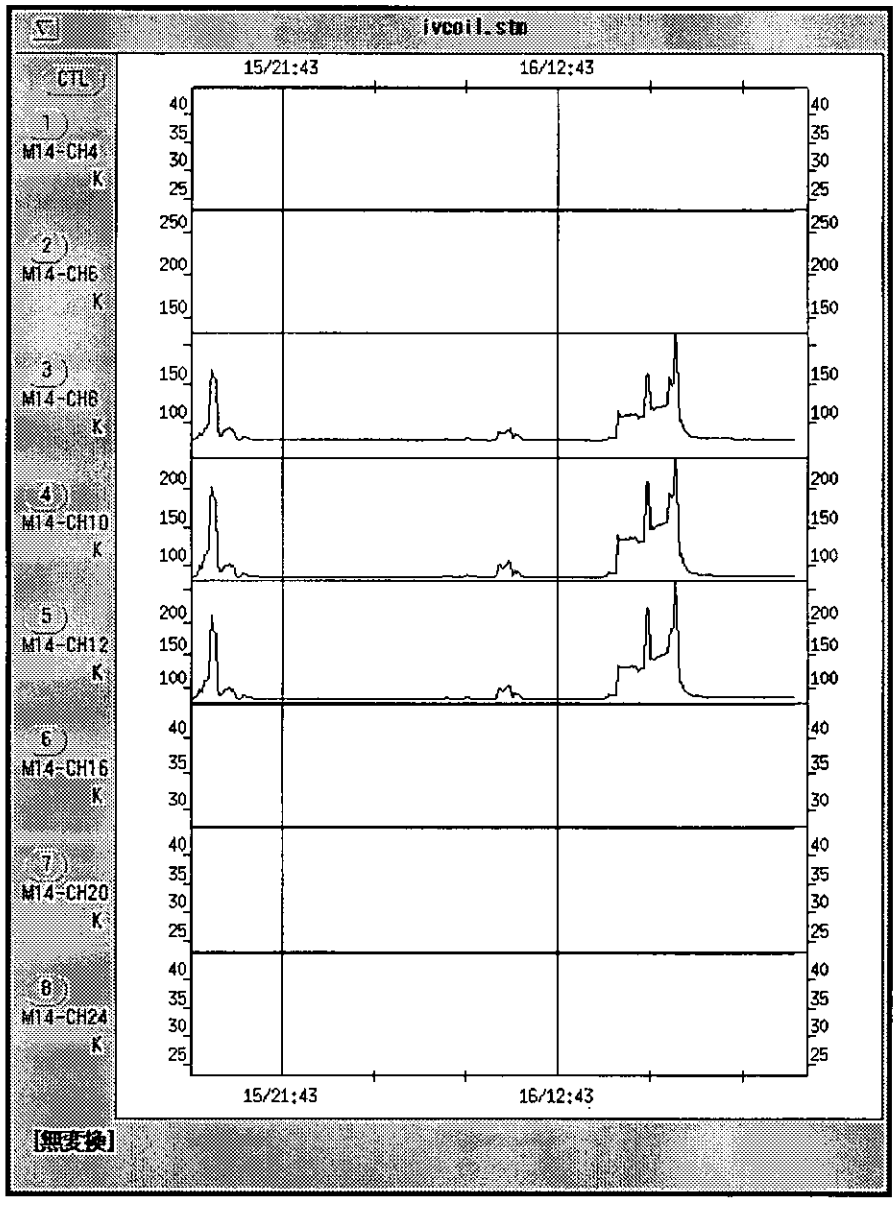

ivcoil.stm
CTL
15/21:43
16/12:43
1
M14-CH4
K
2
M14-CH6
K
3
M14-CH8
K
4
M14-CH10
K
5
M14-CH12
K
6
M14-CH16
K
7
M14-CH20
K
8
M14-CH24
K
[無変換]

図 4.14 拡大表示画面

4.7 プロトタイプ・システムで得られた性能

図 4.1 の経路の中で、リアルタイム性能を評価する経路は、(2)(12)(3)(13)の経路である。図 4.15 にはこの経路の要素だけを示してある。プロトタイプ・システムを使用した試験により図 4.15 の性能が得られた。Display への転送速度が 100chunk/s[3]となっているのは、スクロールするグラフでデータを表示すると、人間の目では 100 サンプルの変化程度が認識できる限界と思われたのでこの速度までしか試験をしていないからであり、本質的な性能限界ではない。同時には 8 データしか表示しないので、100chunk/s は通信速度としては 1.6KB/s(100Hz×8 データ×2 バイト)でしかない。表示では通信速度より描画方法が問題となる。なおこの図の性能は、たとえば ADC から計算機の主記憶間への転送速度というように、データ転送の両端間の最大性能であり、複数の経路を直列に接続した場合は中継点における競合により性能は低下する。

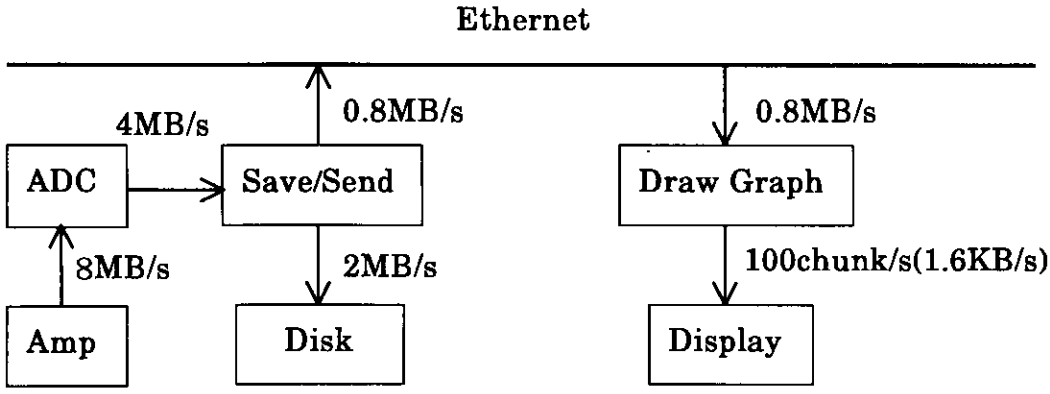


図 4.15 プロトタイプ・システムの性能

図から分かるように、ネットワークがこの時使用したシステムでのボトルネックになっており、転送速度は 0.8MB/s である。LHD 表示システムで必要な通信速度はマルチキャストを使ったために 41KB/s にまで低下しているので、必要な通信速度の約 20 倍の速度である。

---

[3] データ転送速度は通常 MB/s で表すが、データ表示速度は同一時刻のデータの組を単位として表示するので、この組を表す言葉として chunk を使用した。ADC で言えば同一サンプリング・タイムであるデータの組なので sample と呼びたいが、sample は1データを意味することもあるので、異なる言葉を選んだ。

ADC から計算機への転送速度は、主に SBus の速度と方式(PIO か DMA か)で決まってしまうので、計算機の性能を上げてもあまり向上しない。ネットワークの転送速度は、10Mbps のネットワークでは実験の結果 0.8MB/s となっているが、100Mbps あるいは 1000Mbps のイーサネットを使用し、高性能の計算機を使用すれば、ネットワークはボトルネックではなくなり、さらに高速にデータを配布することができる。データ保存では、RAID 技術を使えば 10～20MB/s の速度が得られる。このように標準技術を使える部分は着実に性能が上がっているため性能向上は容易に達成できる。従って最新の製品を使用すると、ADC がボトルネックになる。

レスポンス・タイムの評価をまとめる。100chunk/s のデータを表示できたので、1 チャンクの表示処理に必要な時間は最大 10ms と推定できる。これで

- ADC のデータ変換終了を待ち合わせる時間 10ms(図 4.1 の経路(2))
- ADC のデータを取り出す時間 1ms 以下(図 4.1 の経路(2))
- データをマルチキャストで送信する時間 5ms(図 4.1 の経路(12))
- 表示のために SSTM のデータ受信終了を待つための時間 10ms(図 4.1 の経路(3))
- 表示に要する時間 10ms(図 4.1 の経路(13))

の時間を合わせると、AD 変換開始からそのデータを表示するまでの時間であるレスポンス・タイムとして 40ms 以下を達成できたことになる。スクロールが発生するなど、一時的に計算量が増大したときに遅くなることもあるが、平均的なレスポンス・タイムとしてよい。一時的に処理が遅れても、SSTM で同期をおこなっているので障害の原因にはならない。本研究の目標であるレスポンス・タイム 0.1 秒は達成できた。

4.8 これまでのシステムとの比較

本論文で紹介したこれまでのシステムは、CAMAC をベースにその弱点を制御装置による切り替えや、UNIX ワークステーションへのブリッジの作成などにより克服しようとしたものである。これに対して本システムではデータ伝送系を明確に分離することにより AD 変換装置の役割を「データを伝送系に渡すこと」と単純にし、データの経路を簡単なものとした新型 AD 変換装置を作成した。

CAMAC を使用した場合、表 2.1 から分かるように、リアルタイムでのデータ取得のために最小単位のデータを読み込もうとすると PIO を使うこととなり、1データ毎に CAMAC プログラムを起動しなければならないため 15KB/s と非常に低速になっている。これが複雑さによる不利益である。CAMAC だけで閉じていれば、多数の計測器を制御できる有利なシステムの機能が、より大きなシステムに組み込もうとすると反対に不利な原因となる。

総合性能についての本システムと他システムの比較を表 4.4 に示す。稼動しているシステムはほとんどリアルタイム用システムではないため、リアルタイム転送速度のデータは得られない。リアルタイム処理をおこなっている安らの方式の転送速度と比較する。

表 4.4 総合性能の比較

	処理形態	リアルタイム転送速度
上瀧らの方式など[10]	バッチ処理のみ	—
安らの方式[11]	バッチ・リアルタイム	15KB/s
本方式	主にリアルタイム	800KB/s

次に ADC データを計算機に取り込むデータ転送速度の比較を表 4.5 に示す。本システム以外ではすべてバッチ処理方式である。このデータ転送速度は、AD 変換装置の性能と計算機までのデータ伝送路により決定される。

VXI バスという新型バスの転送速度が非常に高い。このことは標準規格の範囲で性能が向上するとよいが、標準規格を超える性能を得るには新しい方式を採用しなければならないことを示唆している。本システムの転送速度でバッチとは、AD 変換を停止しているときの転送速度であり、リアルタイムとは AD 変換中の転送速度である。バッチ転送速

度は計算機の速度、特にバスの速度に依存し、リアルタイム転送速度はデータ取得方法やチャネル数に依存する。

表 4.5 データ転送速度の比較

	装置あるいはバスの種別	データ転送速度
JT-60[7]	CAMAC	300～400KB/s (バッチ)
REPUTE-1[8]	CAMAC	1.14MB/s (バッチ)
	VXI	23MB/s (バッチ)
安[11]	CAMAC	940KB/s (バッチ)
本システム	SBus	4MB/s (バッチ)
		2～2.5MB/s (リアルタイム)

新型 ADC からのデータ転送速度は、CAMAC を使うこれまでのバッチ処理方式のシステムに比べ、バッチ処理方式ではもちろんのこと、リアルタイム処理方式の転送速度でも、より高速となっている。

5 IV コイル励磁試験への適用

1995 年 11 月に実施された IV コイルの励磁試験において、測定系、伝送系、表示系、保存系からなる分散システムを稼働させた。64 チャネル AD 変換装置を使用し、温度、磁場、歪計、圧力計など、48 項目のデータを対象とした。データ保存の試験はこれに先立ち 1995 年 3 月の冷却実験でもおこなった。

このとき実施した表示システムとしての機能は、

- 長時間にわたる実験の表示およびデータ収集をおこなうこと。
- 定期的間欠的に高速データを収集すること。
- 随時、短時間のあいだ高速データを収集すること。
- 全データについて、短期、長期の表示をおこなうこと。
- リアルタイム表示において、測定値の較正を行った物理量を表示すること。
- 外部にリアルタイム・データを供給すること。

であった。

本章では、実験に使用した計算機と役割をシステム構成図と共に示し、システムが稼動するうえで開発方針や設計した機能がどのように有効に働いているかを説明し、実施した実験内容の詳細を述べる。最後に、この実験で稼動したプロトタイプ・システムを元にして実用システムが作成できることを説明する。

5.1 プロトタイプ・システムの構成

実際の実験はネットワーク管理を容易にするため、リアルタイム・データ用ネットワークおよびバッチ・データ用ネットワークを共用する構成でおこなった。実験の目的は、性能限界を調べることではなく、分散システムによるプロトタイプ・システムの実用性と安定性を示すためだったので、計算機の性能やディスク容量を考慮して、高速データは 1KHz、低速データは 1Hz で取得することとした。またリアルタイムにネットワークを通じて送信できない高速データ保存をおこなうため、収集用計算機でバッチ処理もおこなわなければならず、収集用計算機の負荷を考えての安全策からもリアルタイム機能は低速とした。これは単にリアルタイム処理システムの試験だけではなく、本来の超伝導コイル実験に供する目的があり、超伝導関連の低速データは1Hz で収集すれば良いと言われていたからでもある。

実験システムとして稼動させたプロトタイプ・システムは図 5.1 のとおりである。測定システム(Measurement System)として AD 変換装置を接続する計算機に SPARCstation 20、表示システム(Monitoring System)としてスクロール・グラフ表示用に SPARCstation 2、三次元表示システム(3D Monitoring System)として三次元表示用に SPARCstation 10、保存・解析システム(Storage and Analysis System)として SPARCstation/IPX、遠隔共同研究システム(Remote Participation System)として SPARCstation 2 を充てた。

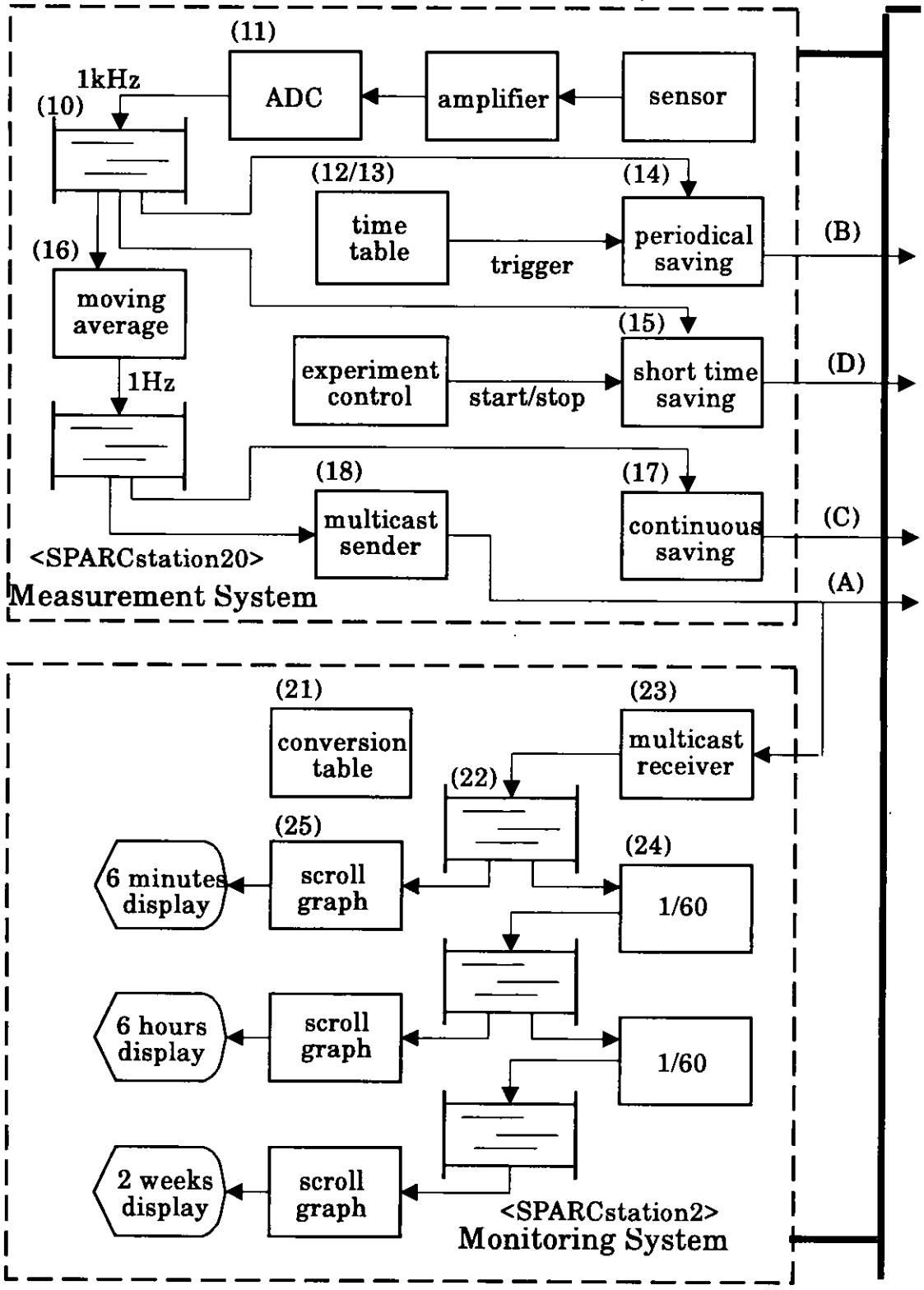
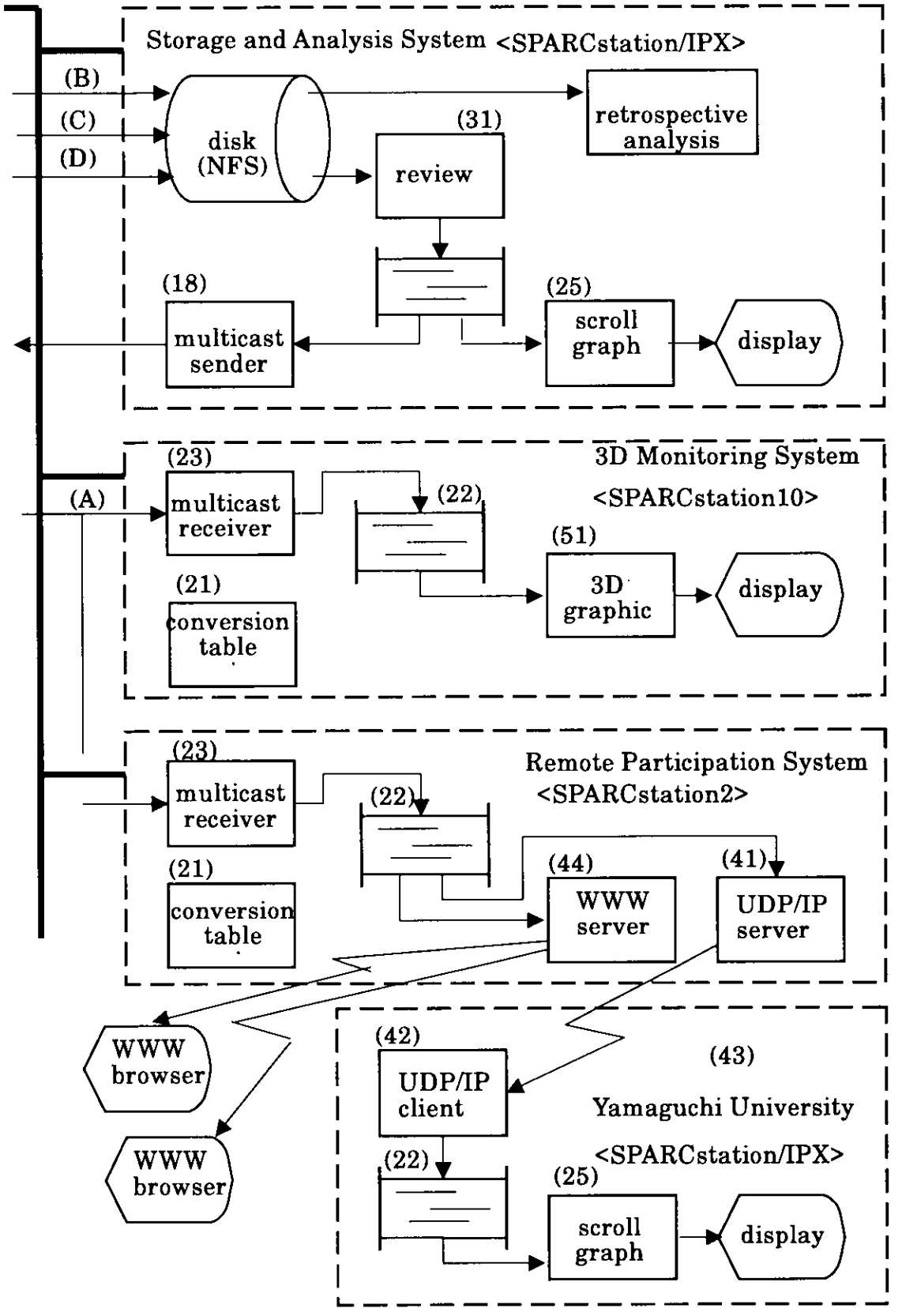
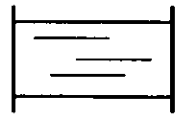


図 5.1 実験システムの構成


Storage and Analysis System <SPARCstation/IPX>
(B)
(C)
(D)
disk (NFS)
(31)
review
retrospective analysis
(18)
multicast sender
(25)
scroll graph
display
3D Monitoring System
<SPARCstation10>
(23)
multicast receiver
(22)
(A)
(51)
3D graphic
display
(21)
conversion table
Remote Participation System
<SPARCstation2>
(23)
multicast receiver
(22)
(21)
conversion table
(44)
WWW server
(41)
UDP/IP server
WWW browser
WWW browser
(42)
UDP/IP client
(43)
Yamaguchi University
<SPARCstation/IPX>
(22)
(25)
scroll graph
display

5.2 実験システムでの開発方針の効果

図 5.1 の実験システムにおいて、開発方針および開発した装置やソフトウエアが有効に機能していることを説明する。図中に表れている右の記号は共有短期記憶機構 SSTM を表している。



5.2.1 ネットワークをデータ・バスとすることによる拡張性

実験システムではそれぞれの計算機の性能があまり高くないため、それぞれが単独の役割を果たすようにした。各計算機の役割は、測定システム(Measurement System)、表示システム(Monitoring System)、保存・解析システム(Storage and Analysis System)、三次元表示システム(3D Monitoring System)および遠隔共同研究システム(Remote Participation System)である。実験期間中に測定システムから通信路(A)でマルチキャストされたデータは、ネットワークをデータ・バスとして表示システム、三次元表示システムおよび遠隔共同研究システムが受信する。バッチ処理で保存するデータには欠損があってはならないため、NFS を使って測定システムから通信路(B)、(C)、(D)のそれぞれの経路で保存・解析システムに送られた。

実験後に実験中の通信を再現するには、保存・解析システムで review を使って保存されているデータを読み込みマルチキャストする。

この実験ではリアルタイム・データを受信する計算機は3台であったが、マルチキャストの性質上、何台あっても問題はないので拡張性のあるシステムである。表示システムでは同一の画面に6分、6時間および2週間のスクロール・グラフを表示したが、ディスプレイの画面が狭いため、同時に見ることはできない。本来はそれぞれの画面ごとに計算機を割り当てるべきものであり、実用時には本論文での推定よりもたくさんの表示システムを置く必要がある。

5.2.2 共用短期記憶機構(SSTM)がもたらす柔軟性

SSTM によるインターフェースの統一は、ネットワークをデータ・バスとして使用することと表裏の関係にある。マルチキャストされたデータを受ける表示システム、三次元表示システムおよび遠隔共同研究システムでは(23)の multicast receiver により受信データを SSTM に格納する。各システム上のアプリケーションは、それぞれのタイミングで SSTM からデータを取得する。

表示システムでは全データを(25)によりスクロール・グラフ表示し、(41)の UDP/IP server も全データを読み込んでインターネットに送信する。これに対して(51)の3次元表示は要求があったときに、要求されたデータ項目の最新データだけを、(44)の WWW サーバは要求があったときに、要求されたデータ項目の最新データまでの1画面に表示できる範囲の過去データを SSTM から取得する。どのアプリケーションでも SSTM からデータを取得するインターフェースは同一であり、(21)の変換テーブルがあれば物理量でデータを取得できる。すなわちアプリケーション・プログラムにとっては SSTM が物理量を出力する AD 変換装置であり、背後の伝送系や計測系についてはまったく知る必要がない。

(51)の三次元表示プログラムは伊藤により、(44)の WWW サーバは奥村により作成された。データ取得方法を通知するために、わずか2回のメイルで十分であった。これは、本システムでの新しいアプリケーション開発の容易性を示し、SSTM による方式がシステム全体としての柔軟性向上に大きな効果を上げていると言える。

multicast receiver は、受信するデータ・パケットの番号と出力する SSTM の名前だけが分かればデータ受信をおこなえる。multicast receiver は複数のシステムで使用されているが、パケットの種類によらず動作するので、プログラムはひとつだけで良い。このように基本的なプログラムが役割の異なる多数の計算機で共通に使用できることは、計算機の増設を容易とし、システムに柔軟性と拡張性をもたせている。

5.2.3 データ分配機能としての共有短期記憶機構

前節で共有短期記憶機構(SSTM)が伝送系との統一インターフェースとなっていることを説明した。SSTM には、計算機内でのデータ分配をおこなう機能もある。測定システム、表示システム、遠隔共同研究システムなどで SSTM から複数の足が出ているものは計算機内でのデータ分配をおこなっていることを表している。遠隔共同研究システムでは(41)UDP/IP server と(44)WWW server が SSTM からデータを取得しており、これらのデータ取得方法は全く異なる。UDP/IP server はリアルタイム・データ通信の一種として全データを受信したら直ちに送信するのに対して、WWW サーバは利用者から表示要求があったときに、指定されたチャネルの一定期間のデータを一括して取得する。UDP/IP サーバは、リアルタイム性を確保するために、受信データ時間間隔を自動認識し、データ待ちポーリング時間を自動調整している。

SSTM からデータを取得するアプリケーションは全く独立に実行しており、さらにアプリケーションは SSTM に対して何の作用も与えない。ましてや SSTM にデータを供給するデータ・フローの上での上位プロセスに対してなんの影響も与えない。つまり、アプリケ

ーションの起動や停止が自由にできるということであり、このことがシステムを柔軟にしている。

5.2.4 リアルタイム処理・バッチ処理の統合

(11)の ADC が SBus 用 AD 変換装置である。(10)の SSTM はこの AD 変換装置に内蔵されているハードウエアとしての共用短期記憶であり、同一 AD 変換装置の1KHz の元データから、移動平均による1Hz リアルタイム・データ作成用、定期的1KHz データ保存用、実験期間中の高速データ保存用のデータを取り出している。

データを処理するときに、一定期間のデータに対して一括処理するならバッチ処理、逐次的に処理するならリアルタイム処理となり、本質的な区別はない。本質的な意味でバッチ処理となるのは、サンプリング速度が速すぎる、あるいはチャネル数が多すぎて AD 変換装置からのデータ取得が間に合わない場合であり、1KHz なら 1000 チャネル、10KHz なら 100 チャネル、100KHz なら 10 チャネルがその限度となる。それ以上のチャンネル数が必要なときは、ADC とワークステーションの組合わせの数を増せば良い。これが分散システムの利点である。

このように処理方式が統合できたのは AD 変換装置のリアルタイム性能が向上しているためであり、AD 変換装置とのインターフェースを簡素化した設計によるものである。10 チャネルの 100KHz データをパケット化して 100Hz のリアルタイム・データとしてマルチキャストすることに何の問題もなく、これまでバッチ処理していた内容をリアルタイム処理に移行することは容易である。

5.2.5 情報の一元化と自己記述性による統一

本システムのような分散処理システムでは、同一情報が複数の場所で使用されるため、複製した情報が独立して修正されることによる矛盾を引き起こさないようにしなければならない。従って、一貫性維持のための仕組みが必要であり、本格的にはデータベースと制御プログラムの導入をおこなうべきである。プロトタイプ・システムではデータベースを使わないため、付録 A に示す工夫を施して、一貫性維持を考慮したシステムとした。

物理量への変換をおこなう(21)の変換テーブルは、テーブルを使用するすべての計算機で共通であり、同一対象を表示する(25)の表示データ定義は各プロセスで共通である。これらの情報は表示システム計算機上の元ファイルを NFS で共有し、複製しないようにしている。記述情報の一元化も定義ファイル記述方式により達成している。(21)の変換テーブルがそれぞれの計算機上に現れているが、これは唯一の元ファイルを読み込んで共有メモリ上に展開した状態を表わしている。

表示システムの中で(25)のグラフ表示プログラムがいろいろな速度に対して使用されている。これらは全く同一であり、個々の表示のために必要な情報も共通である。このように共通に利用できるようになっている理由は、表示プログラムを速度非依存にするとともにデータ・パケットが時刻データを持ち自己記述データとなっているからである。

5.3 実施した実験の内容

図 5.1 中で番号を付けた機能について説明する。データを扱うプロセスはどの順序に起動しても、いつ停止しても障害は発生しない。受信データが無い場合は待ち続けるだけである。このことはシステムの柔軟性を示しているが、一方、これだけでは停止しているという情報が伝わらないため、システム全体の動作状況を監視することの必要性を示唆している。

(11)ADC の起動

ADC 制御マスター・プロセスが ADC を起動し、常時 AD 変換を続けるように駆動する。取得するデータのチャネル数を 64、サンプリング・クロックを 1KHz と設定した。ADC 制御マスター・プロセスは ADC を起動するだけである。ADC は常に 1KHz でデータを生成して内蔵メモリに書き込んでいるので、スレーブ・プロセスを自由に起動・停止でき、起動すると 1KHz、64 チャネルのデータを取得できる。この実験では3つのスレーブ・プロセスが同時に ADC データを取得する。

AD 変換装置内の SSTM によるデータ分配機能が有効に働いて、複数のプロセスが同時にデータを取得できる。またスレーブ・プロセスの起動停止が他に影響を与えないことも重要な点である。次のコマンドで ADC を起動する。引数の意味は前と同じである。

```
adc_run -v/dev/ad710_0a -c64 -u1000 &
```

(12)定期的実行制御での起動通知

定期的な実行の制御には crontabs を使う。毎時 0 分に(13)のコマンドを起動するには、crontab コマンドで次の内容を持つファイルを作成する。crontab コマンドを使わずに直接作成した場合は、reboot しなければ有効にならない。

/etc/cron.d/cron.allow ファイルに使用者 uid を登録し、次の行を

```
0 * * * * /opt/JK/etc/trig/trig_shot.sh
```

crontab /opt/JK/etc/trig/crontab コマンドで登録する。

trig_shot.sh ファイルの内容は下記のようになっており、決められたファイル(ここでは /opt/JK/etc/trig/pid.hour)に登録されているプログラムにシグナルを送る。このよう

にしたのは、利用者が直接 **crontab** を修正しなくて良いようにするためと、利用者のアプリケーション・プログラム起動方法とアプリケーション処理の時刻指定を独立にして、運用の柔軟性を高めるためである。コマンド・リストは以下のようである。

```
#!/bin/sh
PID_FILE=/opt/JK/etc/trig/pid.hour
if [ -f $PID_FILE ]; then
#   echo $0 send signal
    kill -HUP `cat $PID_FILE`
fi
```

(13)定期的実行制御でのアプリケーション起動

固定長データ保存コマンドを定期的に実行するために、

```
rcv_signal -p/opt/JK/etc/trig/pid.hour -c"adc_to_disk -f
/net/ews_sto/ivcoil/S%s -b1000 -n30 -v/dev/ad710_0b" &
```

というコマンドを実行する。この **rcv_signal** というプログラムが自分のプロセス番号を **/opt/JK/etc/trig/pid.hour** ファイルに登録し、(12)の機能により定期的にシグナルを受信する。シグナルを受信すると、引数で指定した

```
adc_to_disk -f/net/ews_sto/ivcoil/S%s -b1000 -n30 -v/dev/ad710_0b
```

という固定長データ保存コマンドを起動する。このとき%s に **YYMMDD-HHMMSS** という形式で起動日時が入るので、ファイルの管理が容易となる。

このようにして、「定期的に起動する」という機能を「データを保存する」アプリケーションから独立させることにより、アプリケーション開発者はいつ起動されるかということについて考えなくてよくなる。これはアプリケーション機能と運用スケジュール機能を垂直分割した例である。

(14)固定長データ保存

高速サンプリングの生データを、毎時 **30** 秒間収集し、それぞれ1ファイルに保存する。 固定長保存コマンドを、毎回ファイル名を指定して次のように起動することにより達成する。**1** 回の起動につき **3840000** バイト(＝**1000Hz**×**64** チャネル×**2** バイト×**30** 秒)のファイルが作成される。 コマンドは、

```
adc_to_disk -f<ファイル名> -b1000 -n30 -v/dev/ad710_0b
```

であり、ファイル名は(13)で説明したように、ショット・データを意味する **S** と作成日時から作成する。

ファイルの実体は保存用計算機上に置かれ、測定用計算機から NFS により書き込みをおこなう。このコマンドの出力するデータファイルには、AD 変換値のみが書き込まれている。これはデータの連結が容易におこなえるようにするためである。データ管理のために、データファイル作成時に、拡張子が.log であるログファイルも一緒に作成する。ログファイルには、先頭データの AD 変換時刻、サンプリング・クロック、チャンネル数などが記入されているので、個々のデータの時刻は計算により求められる。

(15)指定長保存

指定長保存では、通電実験などがおこなわれるときに、マニュアルで開始・終了を指定し、 指定した時間の間のデータを取得する。会話型処理なので、表示用計算機から計測用計算機にログインして使用する。テキスト・ベースの tadc_pre_stop コマンドまたはウィンドウ・ベースの wadc_pre_stop コマンドを使用する。

この実験では直接 AD 変換装置から取得したデータを保存したが、(16)の結果など、SSTM のデータを保存するには、wadc_pre_stop に対応する wstm_pre_stop を使用する。

(16)ノイズ除去

表示および長期保存用に 1KHz のデータから 1Hz のデータを作成する。このときに 1000 サンプルの平均値を取ることによりノイズを除去する。但しこのノイズ除去は、絶縁アンプ[14,15]のノイズ対策には有効であるが、[27]で対象とした低周波成分についてはまったく効果はない。

このノイズ除去は次のコマンド

```
adc_mean_stm -v/dev/ad710_0c -c64 -n1000 -m1000 -f mean.stm &
```

でおこない、平均値は mean.stm という名前の SSTM に、AD 変換時刻とシーケンス番号を加えて格納する。ここで必要となる SSTM は、あらかじめ

```
stmx_create -c mean.def
```

というコマンドで作成しておく。

(17)長期間連続保存

ノイズ除去したデータを mean.stm から取得し、連続して長期間保存する。データが余りにも大量だと扱いにくいので、一定量のデータ毎にファイルを切り替える。次のコマンドで 60 チャンクのデータを1ブロックとし、1440 ブロックを 1 ファイルに書き出す。ファイル名はショット・データと同様に、連続データを意味する文字 C の後にファイル作成日付 YYMMDD-HHMMSS を付けたファイル名とする。約 1 日 1 ファイルとなるが、AD 変換

装置のクロックの精度と計算機のクロックとの精度の違いにより、毎日1秒程度の違いが発生した。次のコマンドで実行する。

```
stm_to_mfile -fmean.stm -d/net/ews_sto/ivcoil/cont/C%s -b60 -z1440
```

(18)リアルタイム送信

次のコマンドで表示用リアルタイム・データの送信をおこなう。マルチキャストなので、複数の計算機で受信できる。ポート番号は 2001 を指定している。長期間保存(17)とリアルタイム送信(18)が同一の SSTM である mean.stm からデータを取得しており、SSTM がデータ分配の役割を果たしている。

```
stm_to_mcip -f mean.stm -p2001 &
```

(21)変換テーブル

表示プログラムが AD 変換データを物理量に変換するために使用する変換テーブルを作成する。較正表やアンプ・ゲインなどの情報は、それぞれの表示プログラムが持つのではなく、4.4 節で説明したように共用テーブルに保管する。

この変換テーブルは次のコマンドで作成する。

```
setenv TBL_FILE ~/tmp/shm/ivcoil.tbl
touch $TBL_FILE
cd /opt/JK/etc/tbl
tbl_load -s 100
include <ivcoil.tbl>
^D
```

全体の定義は、ivcoil.tbl でおこない、補間係数の定義は、C14252.tbl、C14396.tbl、C14399.tbl、C14401.tbl、C14455.tbl、C14458.tbl、C14598.tbl、C14626.tbl、C14639.tbl、C14647.tbl、P94779A.tbl、P94779B.tbl、Cernox.tbl、RuO2.tbl の各ファイルでおこなう。センサーの製造番号を定義ファイル名にしている。定義例はリスト 4.4、リスト 4.5 に示したとおりである。

(22)SSTM の作成

表示用に使用する SSTM をあらかじめ作成しておく。ここでは、秒単位のデータを格納する second.stm、分単位のデータを格納する minute.stm、時間単位のデータを格納する hour.stm を、次のコマンドで作成する。

```
stmx_create -c /opt/JK/etc/stm/second.def
stmx_create -c /opt/JK/etc/stm/minute.def
```

```
stmx_create -c /opt/JK/etc/stm/hour.def
```

second.def ファイルの内容は次のようになっており、

```
stm "second.stm" 32 channel 1000 data;
#
load table ;
#
# ivcoil data
include </opt/JK/etc/stm/ivcoil.channel>
```

SSTM を **second.stm** という名前で作成すること、変換テーブルとして(21)で作成したテーブルを使用すること(テーブル名はデフォールトの環境変数から得ている)、**SSTM** に含まれる各チャネルのデータ名および物理データの属性についてはリスト **4.3** の定義を使うことを指定している。

(23)リアルタイム受信

次のコマンドを使って(18)でリアルタイム送信したパケットを受信する。受信したパケットは、(22)で作成した、秒単位のデータを提供する **SSTM** である **second.stm** に格納する。

```
mcip_to_stm -f second.stm -p2001 &
```

(24)時間単位の変更

長時間表示のために間引きし、データの時間単位を変更する。下記のコマンドで、秒単位のデータを格納する **second.stm** から **1/60** に間引いて分単位のデータを格納する **minute.stm** に格納し、さらにそれを **1/60** に間引いて時間単位のデータを格納する **hour.stm** に格納する。

```
stm_cull2 -f second.stm -t minute.stm -n60 &
stm_cull2 -f minute.stm -t hour.stm -n60 &
```

(25)スクロール・グラフ表示

下記のコマンドで、秒単位データ用**SSTM** である**second.stm**からデータを入力し、**1** 秒 **1** 点のデータを表示用コンフィギュレーション・ファイル である **ivcoil.scg** の定義(リスト **4.3**)によりスクロール表示する。標準では1画面に **400** 点表示するので、**400** 秒間(約 **6** 分間強) のデータを表示できる。

```
setenv SCG_DIR /opt/JK/etc/tbl
scg21 ivcoil.scg second.stm &
```

同様に minute.stm からデータを取得して表示すると、400 分間(約 6 時間強)のデータを表示できる。hour.stm のデータなら、400 時間(約 2 週間強)のデータを表示できる。このように SSTM の大きさとデータ格納方法を選択することにより、必要な過去データを必要な時間精度で格納し、いつでもデータを使用できるようになる。

(31)再生制御

再生制御は事後解析のためのツールであり、主に実験状態のシミュレーションに用いる。図 5.1は、事後にデータをスクロール・グラフで見ることと、シミュレーションのために保存されているデータをネットワークにマルチキャストで送信することを同時におこなっている状態を示している。

(41)ユニキャスト・データ送信

遠隔表示用のサーバは研究室内に置いてあるため、マルチキャスト・データを受信できない。(通常ルータはマルチキャスト・データを通さないように設定してある。)ルータを超えてデータを送信するため、次のコマンドでリアルタイム・データを UDP パケットに乗せて送信した。ysl はサーバの名前である。このデータを受信したサーバは、公開サーバの役割を担っており、研究所外へデータを供給する。ポート番号はデフォールトを使用している。

```
stm_to_udp -f second.stm -a ysl &
```

(42) ユニキャスト・データ受信

次のコマンドで(41)のユニキャスト送信データを受信して SSTM に格納する。

```
udp_to_stm -f second.stm &
```

このように SSTM にデータを格納してしまえば、リアルタイム計測系が外部ネットワーク上に複製されたことになり、本来の計測系に影響を与えることなく、種々の表示方法の試験がおこなえる。

(43)遠隔表示

(41)で使ったユニキャスト通信はインターネットでの通信であるから、どこへでも送信できる。(23)で SSTM に取り込んだリアルタイム・データを(41)のコマンドを使って山口大学の研究室内の計算機へ送信し、(42)のコマンドで受信して SSTM に取り込み、(25)のコマンドでスクロールグラフを表示する。このようにして、実験期間中を通して計測が正しくおこなわれているかどうか、山口大学で表示することができた。

(44)WWW サービス

直接データ・パケットを受信できない共同研究者に対しては、WWW ブラウザだけあればデータの波形を見られるサービスが奥村により作成された[16]。

(51)三次元表示

IV コイル設計に使った CAD データを使って LHD の3次元画像を構築し、その上にセンサーを置く三次元表示用アプリケーションが伊藤により作成された[16]。構築にはAVS というソフトウエア製品を使用した。三次元構造上のセンサーを選択すると、そのセンサーで測定している現在値を表示する。しかしながら、ここで用いられたワークステーションでは、実行速度が大変遅く実用に供されなかった、この論文を書いている時点(1999年 6 月)では、10 倍以上高速のワークステーションが一般的に利用されているので、レスポンス時間も短くなり実用になると思われる。3次元表示のような可視化をおこなうことは、今後のデータ表示方法の一つの方向であろう。

5.4 実験結果と考察

低速データの長時間連続表示および保存、高速データの周期的保存、励磁実験中の高速データ保存を同時に行ったことで、プロトタイプ・システムの基本機能を満たすことができた。期間中に表示プログラムの設定変更をするための再起動をおこなっても問題がなく、システム構成を柔軟に変更できることが示された。

表示用計算機の増設について考察する。実験でのグラフ表示では、1 台の計算機で 3 種の表示時間範囲を表示したが、最高でも 1Hz であるため、負荷は小さいものであった。高速表示が要求されても、人間の目では 100Hz を限度としてよく、負荷が高ければ表示用計算機を増設すればよい。同様に表示データ数が増えることも表示用計算機の設置台数を増やせばよい。

高速データ保存でのサンプリング・レートが 1KHz であり、リアルタイム処理対象外の速度であるため、収集用計算機でバッチ処理方式によるデータ保存処理をおこなった。バッチ処理によるデータ保存や、そのためのプログラム起動などは、計算機負荷を大きく変動させ、リアルタイム性がもっとも重要な収集用計算機の利用法には妨害となる。今後は、受信計算機およびネットワークの性能を向上させ、高速データもネットワークで分配できるようにする必要がある。しかし、もっとも早期に処理ができるのは収集用計算機上であり、この計算機がもっとも高性能である必要がある。データ保存の負荷がリアルタイム送信に影響することを軽減するためには、マルチ・プロセッサとすることが望ましい。

次にシステム構築の容易さについて考察する。収集用計算機からリアルタイム・データを送信し、マルチキャストによってデータ分配を行い、収集用計算機などの計算機上でデータを受信し、共用短期記憶 SSTM に格納するまではデータ分配系の範囲であり、全体を運用する責任者がおこなうことである。従ってアプリケーションとのインターフェースは、SSTM の使用方法ということになる。

三次元表示のように、要求されたときの最新値だけが必要な場合はもっとも簡単な利用方法であり、まず、

```
g_stm = x_stm_open(<filename>, <shm key>);
```

により、あらかじめ作成してある SSTM をオープンし、そのハンドルを g_stm に取得すると、あとは

```
x_stm_get_current_val(g_stm, <channel>, <data address>);
```

というライブラリ関数にチャネル番号を渡すだけで物理量を取得できる。このとき内部では SSTM ライブラリが ADC の生データの値を得て TBL ライブラリを呼び出し物理量に変換するという操作をおこなっている。

このように高度なレベルでのインターフェースを提供しているため、アプリケーションの構築が容易になっている。このことは、表示方法そのものを研究開発していく上で非常に有効なことであり、多種の表示方法が構築されることはシステムの柔軟性に大きく寄与する。

次に情報の正しさについて考察する。アプリケーションが容易に物理量を得ることができた理由は、あらかじめ各計算機ごとに変換テーブルが定義されていたからである。この変換テーブルは、リスト 4.4、リスト 4.5で示すように、センサーの較正表、アンプ・ゲインなど、種々の情報から作成される。現在はこの情報を定義ファイルという形式で保持しているが、その正しさを保証するためには、データベースによる管理が必要となる。

以上をまとめると、プロトタイプ・システムをモデルとして収集用計算機およびネットワークの性能を向上させ、LHD の情報を一元管理し伝送系の運用を自動化すれば、拡張性、柔軟性ともに高度なシステムを構築できると言って良い。

図 5.1 の実験システムで使用されたプログラムの多くは、4章で説明したプロトタイプ・システムでの標準のプログラムであるが、長時間連続保存(17)など、標準のプログラムを実験システム用に改造したものもある。データ伝送系としてどのような機能を提供するかの検討もまだ必要である。

6 LHD 用実システムへの適用と今後の展望

本研究の結果として実用となる LHD 計測システムが作成できること、およびその作成方法があきらかになったので、運用に供される実システムとして制御データ処理装置が作成された。本方式ではリアルタイム・データのバスとしてネットワークを使用することが重要な点のひとつであるため、望ましいネットワーク構成を提案する。本方式は、リアルタイム・データ計測のためのデータ伝送系をアプリケーションから独立させた点が重要な点であり、この方式に沿ってリアルタイム処理方式の標準化がおこなわれることを期待する。

LHD 制御データ処理装置をさらに発展させる方向について議論し、最後に本論文で得られた成果をまとめる。

6.1 LHD 用実システムの実現

本論文の結果を踏まえ、プロトタイプ・システムの中核となる収集用計算機を強化し、情報管理の一元化のためにデータベースを使用した LHD 用実システムとして制御データ処理装置が作成された。

制御データ処理装置は、プロトタイプ・システムを強化すると共に、不足している機能を重点的に補ったシステムであり、東芝と日本サン・マイクロシステムズ社により作成された。制御データ処理装置は、データベース中心で、Java および Web 技術を使ってネットワーク基盤上でのサービスを提供している。提供されているリアルタイム表示サービスを受けてデータを見るだけなら、利用者側には Web ブラウザだけあれば良いようになっている。また管理もブラウザを使っておこなわれる。収集用計算機には合計 768 チャネルの AD 変換装置を接続しており、常時リアルタイムにデータを取得し、保存および提供している。表 6.1 にプロトタイプ・システムと制御データ処理装置との比較を載せる。

表から分かるように、制御データ処理装置は、第 5 章の実験に適用したシステムのうち、利用者側の表示システムを除いた部分を統合している。制御データ処理装置ではデータベースにより、利用者管理を始め LHD の運用に関わる情報をすべて管理する。すなわち、物理的な装置だけにとどまらず、データ項目という抽象的なものにまでタグと言う識別子を付けて管理する。このことは情報の散逸を防ぐと共に、実験状態の情報が唯一に確定していることを保証する。

Web サーバの提供するデータ表示を利用しないで個別の表示方法をおこなう利用者は、収集用計算機から ADC 生データと物理量に変換したデータをマルチキャストにより送信しているので、このデータを受信して表示するプログラムを作成すればよい。このよ

うな表示用計算機を接続してデータを受信しても、制御データ処理装置にまったく影響を与えず、通信量の増加もないことはいままでに述べたとおりである。

表 6.1 プロトタイプ・システムと制御データ処理装置の比較

項目	プロトタイプ・システム	制御データ処理装置
AD 変換装置チャネル数	1KHz*64 チャネル	1Hz*512 チャネル 1KHz*256 チャネル
低速・高速サンプリング	同一 AD 変換装置	独立 AD 変換装置
マルチキャストするデータ	1Hz 生データ	1Hz 生データ＋物理量データ
物理量への変換	表示計算機で	収集計算機で
クエンチ・データ収集（ハードウエア・トリガーによる自動データ収集）	なし	あり
利用者資格	計算機の利用者	データベースにより資格検査
制御方法	直接各計算機で	ブラウザを使ってどこからでも
データ定義ファイル	プレーン・テキストを NFS で配布	データベースにより管理
実験管理	なし	データベースにより管理
実験記録	奥村が別途作成	実験管理に組み込み
3次元処理	伊藤が別途作成	江本が個別に作成した
情報管理	なし	データベースにより人、装置、設定データなど、多数の情報を管理
データ表示	データを受信して各自で	WWW による標準のデータ表示画面
異常検出	なし	上下限検査をして通報
データ処理	なし	対話的にデータ編集をおこなう

制御データ処理装置は、標準のデータ表示と連動し、簡単な異常検出により担当者にポケベルおよびメイルで通知する機構を備えている。今後高度な判定方法が作成されれば、このシステムに組み込むことができる。このように運用に必要な機能を組み込む基礎ができたと言える。

6.2 望ましいネットワーク構成

計測システムのネットワーク構成では、(1)計測システム内での通信のための構成を考えると共に、(2)外部からの不要な通信を受けないようにする必要がある。このためネットワーク構成は図 6.1 のようにするのが良いと考える。

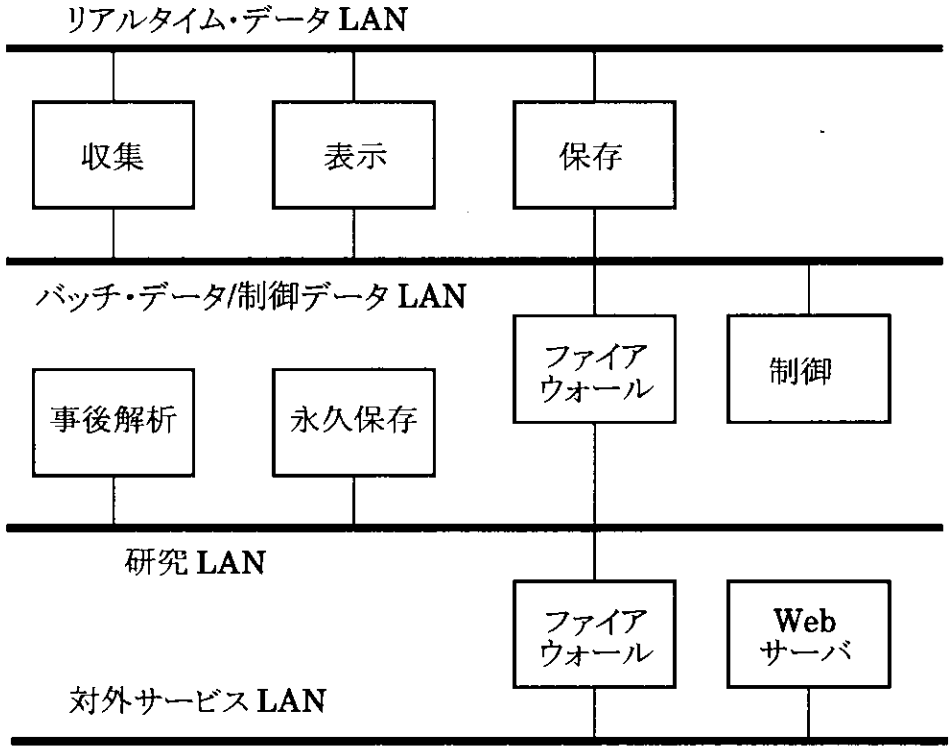


図 6.1 望ましいネットワーク

(1)のために、目的ごとに LAN を設置し通信を分散する。

リアルタイム実験系にためには、リアルタイム・データ用 LAN と保存データをアクセスしたり、表示システムの計算機を制御するためのバッチ・データ/制御データ用 LAN を分離し、リアルタイム・データの通信において衝突をなるべく発生させないようにする。

事後解析のための研究 LAN や対外サービス LAN も分離し、実験に関係のない通信が入り込まないようにする。

(2)のために、それぞれの LAN の間にルータあるいはファイア・ウォールを置いて適切なアクセス制御がおこなえるようにする。外部から内部(図 6.1 の上部)への通信は、外部にある特定の計算機からしかおこなえないようにし、その特定の計算機では少数の管理者が特定の機能しかおこなわないようにする。研究 LAN では、図の永久保存用計算機のみが内部と通信して、事後処理用のデータを保存する。この計算機には利用者用のアカウントは置かず、事後解析用の計算機に NFS で読み取り専用のアクセスのみ許可する。

研究 LAN でおこなう事後解析がリアルタイム・データを必要とする場合は、永久保存用計算機のほか、リアルタイム・データ保存用計算機のミラー・サーバを設置する。この場合も、ミラー・サーバのサービスは、NFS や Web などを使って間接的にしか受けられないようにする。

対外サービス LAN では、図の Web サーバのみが内部と通信できる。この Web サーバは、保存用計算機から外部に提供するデータのみを受信し、Web を介してのみアクセスを許可する。多くのデータを外部に提供するなら、この Web サーバをミラー・サーバとし、研究 LAN ではこのミラー・サーバを使うこととしても良い。この場合、研究 LAN はこのミラー・サーバの内側にあるので、どの事後解析用計算機でもアクセスできる。

これまで述べたアクセス制限は、ルータの機能だけで実現できる。内部と通信可能な計算機であっても、telnet は許可しないというように、許される通信プロトコルを制限するためにはファイア・ウォールを設置する。

6.3 リアルタイム計測方式の標準化

TRIAM-1M の例に見られるようにトカマク型装置においても放電持続時間が伸長しているため、トカマク型装置のプラズマ制御に対してもリアルタイム処理が必要になると思われる。各種の装置においてシステム構築を容易に柔軟におこなうには、リアルタイム計測方式の標準化をする必要がある。

リアルタイム計測方式を標準化するためには、4.1節で説明したデータ・パケットの定義と、4.2節で説明した、通信およびアプリケーション間のインターフェースの定義が重要である。後者に対しては共用短期記憶機構を提案した。通信方法として本論文ではマルチキャストを使用したが、データ伝送系はインターフェースによりアプリケーションと切り離されているので、通信方法の選択は自由である。リフレクティブ・メモリでも CAMAC

データ・ハイウェイでも、必要な性能を満たすものならなんでもよい。この自由度はこのシステムの高い柔軟性のひとつでもある。

計測方式を標準化することは実験計測に大きな寄与をする。CAMAC による計測制御装置の標準化はその一例である。本研究では、データ伝送系の機能を確立することにより、リアルタイム計測方式を標準化できることを示したものである。標準化ができれば国際協力により大型実験装置の表示システムを構築できるなどの効果が期待できる。以上の理由により、本方式が今後プラズマ核融合実験でのリアルタイム処理における標準となることを大きな希望としたい。

.リアルタイム計測方式の標準化はデータ取得を容易にするために寄与するが、計測では取得したデータを処理するアルゴリズムが重要であり、データ処理方法がリアルタイム処理に変わると、これまでバッチ処理用に開発されたソフトウエアをリアルタイム処理用に変換する必要がでてくる。たとえば、MDSplus には膨大なソフトウエアの蓄積がある。バッチ処理用のプログラムは、入力データをファイルから短期記憶内の時系列データに切り替えることにより、リアルタイム処理用に変換できる。次に、処理アルゴリズムを一括処理からインクリメンタル処理に変換することで、レスポンス・タイムを短くすることができる。このようにしてバッチ処理用に作られたトカマク型装置の制御システムをリアルタイム化できると考えられる。

6.4 LHD 制御計測システムの今後の課題

現在制御データ処理装置が安定して運用されているが、制御データ処理装置を含め LHD 計測システムの今後の課題について述べる。

(1)対象とするデータを拡大する。

現在は温度、圧力などのアナログ信号だけを扱っているが、伝送系は信号の種類を問わないため、デジタル信号や LHD 運転状態のような制御データもネットワーク上で分配できる。可能な限り本方式でデータを配送することにより、総合的な計測が可能となる。

(2)プラズマ実験用に改善する。

現在運用中の制御データ処理装置では安全な運転をおこなうための LHD の監視に重点があるため、データベースがプラズマ実験用には不向きである。したがって、今後長時間放電がおこなわれるプラズマ実験用には、データベースの構造を変える必要があろう。

(3)動的判断方法を導入する。

制御データ処理装置の拡張性を生かして、さまざまな監視方法の開発を進めることが重要である。現在は静的な判断基準による監視をおこなっているが、LHD の状態により変化する動的な判断をする必要があるであろう。この LHD の状態はデータ・パケット内のデータとして配送される。

(4)S/N 比を改善する。

AD 変換装置としては、絶縁アンプによる信号精度の劣化を防ぐため、シールド・ボックス内に設置できるものを開発する必要がある。

(5)高性能計算機の適用限界を評価する。

マルチ・プロセッサの計算機を用いるとタイムシェアリング・スケジューリング・クラスでもデータ欠損を生じない、あるいは問題ない程度まで少なくなると思われる。システム構築を容易にするためには、この評価をしておく必要がある。

(6)リアルタイム性能を向上させる。

制御データ処理装置を運用を開始してから 1 年半以上もたち、その間に計算機等の進歩も著しく、今後もリアルタイム系なども含めてより高速のシステムが構築できる段階になってきている。前項での評価をおこなえば、AD 変換装置のサンプリング速度を10 KHz とし、リアルタイム・データ取得周期を 10Hz とできる評価が得られるであろう。

(5)ネットワーク構成や分散処理時の性能を評価する。

現在は1Hz という低速リアルタイム表示なので問題を生じていないが、高速リアルタイム表示でデータ統合、データ変換をおこなうためには、ネットワークをバッチ用ネットワーク、リアルタイム用ネットワークに分離し、データ送信計算機が複数台の場合での通信速度および通信容量を調べる必要がある。ただし、データ統合とデータ変換をマルチ・プロセッサでおこない、ネットワークは表示用計算機へのデータ分配だけに使用するのであれば、この問題はない。

7 まとめ

本研究は、核融合科学研究所の共同研究「ワークステーションを用いたデータ収集・解析・制御システムの研究」に、物理学、電子工学、ソフトウエア工学などの専門家を結集して開始された。

LHD は準定常運転での長時間放電に特徴があり、制御用データをリアルタイムに計測することが不可避である。これに対して従来の核融合装置のデータ処理システムはすべてバッチ処理であった。また、CAMAC モジュールからリアルタイムでデータを取得するシステムは、低速で LHD が必要とするデータ転送量を達成していなかった。このため本研究では、LHD 制御用実時間計測システムを構築できることを実証することとした。

LHD にはリアルタイム処理できない高速データや、イベント発生時の記録としてのデータもあり、従来のバッチ処理も必要となる。LHD の制御とは、超伝導コイルを制御することであり、時定数が比較的に大きい。このため、目標とする性能を、「2000 点のアナログ・データを 10Hz の周期で 100ms 以内に表示でき、バッチ処理にも対応できる」こととした。このように、LHD のような大型実験装置の計測システムは、(1)大容量高速転送が必要で、(2)リアルタイム処理とバッチ処理の両方が必要な、(3)分散処理システムにならざるをえない。

この目標を達成するため、次の3つの方針を立てた。(1)LHD の建設は長期にわたるため、標準技術を採用して、最新技術や装置を導入することにより、継続的な性能向上を図る。(2)研究で作成するプロトタイプ・システムが実用規模でも稼動することを保証するため、拡張性と柔軟性のある構成とする。(3)ソフトウエアの規模も大きくなるため、ソフトウエア技法を適用して、継続的に利用できるシステムとなるように開発する。これらの方針により、標準 UNIX 上にイーサネットをネットワークとする分散システムとしてシステムを構築することとした。

システム分析の結果、(1)ネットワークをデータ・バスとして使用できることを確認すること、(2)新型 AD 変換装置を開発すること、(3)データを柔軟に扱えるようにするため、データ収集系とデータ表示系の間にデータ伝送系を設置することにより、利用者に対するインタフェースを簡易化することが必要となった。

ネットワークをデータ・バスとして使用するために、システムの規模から通信量を推定した結果、ユニキャスト通信方式では、柔軟性の高い構成にすると通信量が大きくなりすぎ、構成変更により通信量が変動するという問題があった。これらの問題点はマルチキャスト通信方式では解決されるが、マルチキャスト通信方式を計測に適用した例は知られていない。このため、マルチキャスト通信方式の通信量の限界と転送に要する時間を調べる

実験をおこなった。実験は 512 チャネルのアナログ・データに相当するデータ長のパケットを用い、計算機を SPARCstation2、ネットワークを10Base5のイーサネットでおこなった。

この実験の結果、マルチキャスト通信方式でデータを欠損する原因は、計算機の性能であり、(イ)リアルタイム・スケジューリング・クラスで実行すれば欠損しないこと、(ロ) 800KB/s の通信量を達成できること、(ハ)データ・パケットの転送時間は最短が 2ms であり、全体としては 5ms とすれば十分であることが分かった。

リアルタイムでのデータ取得を可能とするため、新型 AD 変換装置を設計し、メーカーに委託して製品化した。リアルタイム性能を高めるため、計算機のバスに直結し、装置内のバッファ・メモリにランダム・アクセスができるようにした。バッチ処理も可能とするため、このバッファ・メモリは 32MB と大容量にした。これらの結果、特定のチャネルを監視し、イベント発生時にその前後のデータを取得するリアルタイム間欠処理という方式も可能となった。

従来の CAMAC モジュールである AD 変換装置を使うシステムでは、バッチ処理方式でのデータ転送速度は 1.14MB/s である。リアルタイム処理方式では 15KB/s にすぎない。新型 AD 変換装置では、バッチ処理データ転送速度は4MB/s であり、リアルタイム処理データ転送速度では 2MB/s を達成している。このため、従来バッチ処理するしかなかったデータ処理をリアルタイム処理できるようにもなっている。

マルチキャスト通信方式と新型 AD 変換装置により、AD 変換装置からデータ表示までのネットワークを含めた総合性能では、10Mbps のイーサネットを使って 800KB/s の通信速度が得られ、データ伝送系を介したネットワーク上のデータ転送時間は 26ms にすぎないことがあきらかになった。800KB/s の通信速度は、10Hz、40000 点のアナログ・データに対応する。これで目標とするシステムの性能を達成できることが実証された。

データ伝送系はデータ・パケットの送信、受信、変換、統合をおこなうサービスの集合体であり、データ伝送系を使うことにより、計測システムを柔軟に構成できる。収集用計算機では、AD 変換装置から取得したデータをパケットとしてデータ伝送系に渡す。このときに、どこで受信するかを知る必要はない。表示用計算機はデータ伝送系からデータ・パケットを取得して表示する。このときに、どこで AD 変換したかを知る必要がない。収集用計算機も、表示用計算機も、それぞれが扱うデータ・パケットが何かを知っているだけで良い。実行する周期が両者で違っていても良い。

このようにデータ伝送系は異なるプロセス間でのデータ授受を仲介するため、同期とバッファの機能が必要である。データ伝送系の中核となる機構として共用短期記憶機構(Shared Short Term Memory, SSTM)を開発した。SSTM は、共用メモリ上にリング・

バッファを基本として作成し、複数のプロセスから、過去データも含め、バッファ内のパケットをランダムにアクセスできる。SSTM には AD 変換装置の値を物理量に変換する機能を持たせたので、表示用計算機にとっては、あたかも物理量を出力し、必要なら過去のデータも取得できる AD 変換装置が自分の計算機に付属しているかのように見える。アプリケーション・プログラムにとって SSTM が唯一のインターフェースとなるため、開発が容易となった。

作成したプロトタイプ・システムを IV コイル励磁試験で 2 ヶ月間にわたり稼動させ、リアルタイム表示、リアルタイム・データ保存、定期的短時間高速データ保存、実験時高速データ保存を同時におこなった。アプリケーション開発の容易さを調べるため、スクロール・グラフによるデータ表示のほかに、Web サーバによる波形データ表示、LHD の3次元構造上でのデータ値表示、インターネット上へのリアルタイム・データ送信をおこない、SSTM によりアプリケーション・プログラムの開発が容易であること、データ伝送系により、これらのアプリケーションを計測システム内に組み込むことが容易であることが確認された。

プロトタイプ・システムが安定して稼動し、中核となる3つの事項、(1)マルチキャスト通信方式を使ってネットワークをデータ・バスとすることによる拡張性、(2)新型 AD 変換装置によるリアルタイム処理とバッチ処理の同時実行能力、(3)SSTM を中核とするデータ伝送系のサービスによる柔軟性を備え、実用システムを構築できることが実証された。

この結果を受けて LHD 用実システム(制御データ処理装置)が作成され、すでに 1 年半以上も連続して運用されている。また今後の方向として、ネットワークを整備すること、LHD 計測システムの機能をさらに高めることが期待される。

開発したデータ伝送系は、データの内容とは関係のない処理をサービスとして提供するものであり、大型実験装置での実時間計測システムを構築する標準要素となりうるものである。データ・パケットと SSTM を中心として実時間計測システムを標準化すれば、国際協力などでの開発に大きな効果を期待できる。

本論文で開発した、性能試験のためのプログラム、プロトタイプ・システムを構成するプログラム、AD 変換装置のデバイス・ドライバなどのプログラムについては、CDROM で提供する用意があり、種々の研究の現場で広く活用され、客観的評価を受けることを期待する。

参考文献

[1] 大型ヘリカル装置(LHD)計画、プラズマ・核融合学会誌第 74 巻増刊(1998)

[2] 室賀健夫:プラズマ・核融合学会誌第 74 巻第 8 号(1998)、pp.802-807

[3] NIKKEI ELECTRONICS BOKS:「欠陥ゼロのソフトウエア開発」(日経BP、東京、1995)

[4] J.サンダース、E.カラン:「ソフトウエア品質向上のすすめ」(凸版、東京、1996)

[5] Robert Dunn:「ソフトウエアの欠陥除去技術」(日経 BP、東京、1985)

[6] William J. Brown, Raphael C. Malveau 他:「アンチパターン」(ソフトバンク、東京、1999)

[7] 青柳哲雄:プラズマ・核融合学会誌第 72 巻第 12 号(1996)、pp.1370-1375

[8] 森川惇二:プラズマ・核融合学会誌第 73 巻第 2 号(1997)、pp.168-173

[9] 上瀧恵里子、伊藤智之:プラズマ・核融合学会誌第 73 巻第 3 号(1997)、pp.330-334

[10] 上瀧恵里子、伊藤智之:プラズマ・核融合学会誌第 69 巻第 8 号(1993)、pp957-962

[11] 安芳次、竹内康雄:日本物理学会誌 Vol.48, No.11, 1993、pp886-893

[12] 中西秀哉、小嶋護、実験データの収集、プラズマ・核融合学会誌第 72 巻第 10 号(1996)、pp.1062-1072

[13] 村岡克紀、笹尾真実子、長島章、プラズマ・核融合学会誌第 72 巻(1996)、pp525

[14] 山口作太郎他:第 51 回 1994 年春季低温工学・超電導学会、p226

[15] 山口作太郎他:第 52 回 1994 年秋季低温工学・超電導学会、p264

[16] 山口作太郎、庄司主、三戸利行、山崎耕造、刈谷丈治、奥村晴彦、江本雅彦、寺町康昌、大須賀鬨雄:プラズマ・核融合学会誌第 73 巻第 3 号(1997)、pp.335-342

[17] 「ソフトウエア・エンジニアリング」(フジテクノシステム、東京、1993)

[18] 丹後俊郎、刈谷丈治、倉科周介、神沼二真、医用電子と生体工学, Vol.18,No.2, pp30-35(1980)

[19] 刈谷丈治、石原好宏、電子通信学会論文誌 Vol.J68-D,No.2, pp114-121(1985)

[20] 刈谷丈治、山口大学工学部研究報告第 37 巻第 1 号、pp95-102(1986)

[21] 日本サン・マイクロシステムズ株式会社、「SPARCstation 2 製品仕様書」

[22] Sun Microsystems, Inc., 「SBus Specification B.0」

[23] 市田浩三、吉本富士市、スプライン関数とその応用、教育出版(1979)

[24] 奥村晴彦、C 言語による最新アルゴリズム事典、技術評論社(1991)

[25] 上谷晃弘、「ローカルエリアネットワーク-イーサネット概説-」、丸善(1989)

[26] 相越克久、田中長光訳、SPARC アーキテクチャ・マニュアル、凸版(1992)

[27] H. Okumura et al.: Fusion Technology 1994: Proceedings of the 18th Symposium on Fusion Technology, Karlsruhe, Germany, 22-26 August 1994. Elsevier Science, 1995. pp.835-838.

用語集

ADC

Analog to Digital Converter、アナログ信号をデジタル化する素子または装置。

busy loop

休止することなく繰り返して実行すること。休止しないため CPU 利用率は100%となってしまう。

CAMAC

Computer Automated Measurement and Control の略。CAMAC dataway と呼ぶバスにモジュール構造の計測器を挿入することにより、統合された計測システムを構成する。

CDROM

Compact Disk Read Only Memory の略。音楽用のコンパクト・ディスクをデータの読み出し専用の記録媒体として使ったもの。

crontabs

指定した起動時刻にプログラムを起動するために UNIX が使用するテーブル。

DADiSP

DSP Development Corporation(http://www.dadisp.com/)の開発したデータ解析ソフトウエア。

DMA

direct memory access の略。デバイスとメモリ、あるいはデバイスとデバイスの間で CPU の介在なしに直接データ転送する方式。

Ethernet

ゼロックス社で開発されたのち標準化され、LAN においてもっとも普及している通信方式。通常は IEEE802.3 CSMA/CD 標準のこと。現在は通信速度として 10Mbps、100Mbps、1Gbps がある。

GB

giga byte の略。データ量を表す単位。1000MB または 1024MB。

GnuPlot

Thomas Williams と Colin Kelley を中心に開発された、自由に利用できる会話型プロット・プログラム。

(http://www.cm.cf.ac.uk/Latex/Gnuplot/gnuplot.html)

GUI

Graphical User Interface の略。利用者との対話をボタンなどの図形要素を用いておこなう方式。

KB

kilo byte の略。データ量を表す単位。1000 バイトまたは 1024 バイト。

IP

Internet Protocol の略。信頼性のないネットワーク間のパケット通信方式。

IP multicast

IP の上に構築された同報通信を行うためのプロトコル。

LabVIEW

NATIONAL INSTRUMENTS(http://www.natinst.com/) 社の製品。装置の制御や収集したデータの表示をするプログラムを作成するためのツール。

LAN

Local Area Network の略。建物内、キャンパス内など、狭い範囲をカバーする通信網

MB

mega byte の略。データ量を表す単位。1000KB または 1024KB。

Mbps

mega bit per second の略。1 秒間に転送するビット数を表す単位。毎秒 100 万ビット。

MDSplus

Los Alamos National Lab の ZTH グループと CNR(Padua, Italy)の RFX グループにより共同開発されたデータ収集システム。(http://www.psfc.mit.edu/mdsplus/mdsplus.html)

memory mapped I/O

周辺装置に対して主記憶アドレスの一部を割り当て、周辺装置のアクセスを主記憶の read/write と同じ命令で行う方式。

Motif

OSF(Open Software Foundation)という UNIX 標準化団体が策定したウィンドウ管理システムの一種。現在は Open Group (http://www.opengroup.org/)が引きついでいる。

NFS

Network File System の略。他の計算機上のファイルをあたかも自計算機上にあるかのように見せかける技術。データ更新に伴い、ネットワーク通信が発生する。

Open window

サン・マイクロシステムズ社が策定したウィンドウ管理システムの一種。

OS

Operating System の略。計算機内でのプログラム実行の管理、ハードウエア制御を行うソフトウエア

PC

Personal Computer の略。パソコンとも言う。本来は個人用の計算機だったが、現在は業務用にも使用されている。

PCI bus

Peripheral Component Interconnect bus の略。PC の入出力バスとして PCISIG(http://www.pcisig.com/)が推奨するバス。最近は PC だけでなく UNIX ワークステーションにも採用されている。

PIO

programmed input output の略。CPU の命令により入出力を実行するので DMA より低速になる。

PLA

Programmable Logic Array の略。プログラムにより論理回路を構成できる素子。

PV-WAVE

Visual Numerics, Inc(http://www.vni.com/) の開発した可視化およびデータ解析ツール。

RAID

Redundant Arrays of Inexpensive Disks の略。安価なディスクを並列に用い、安全性や高速性を高めた単一のディスクに見せかける技術。

reboot

計算機を再起動すること。

RS-232-C

米国電子工業会規格である、低速、近距離用の通信方式。ほとんどの PC に標準装備されている。最近は規格を超える速度や距離でも使用されている。

SBus

サンマイクロシステムズ社の SPARC 計算機のシステム・バス兼入出力バス。高性能計算機は入出力バスとしてのみ使用される。

SPARC

Scalable Processor ARChitecture の略。サンマイクロシステムズ社の計算機で使われている CPU の命令体系。

SSTM

Shared Short Term Memory の略。複数プロセス間のデータ授受方式として、本論文で推奨する方式。保存期間を任意に指定できるところに特徴がある。

TCP

Transmission Control Protocol の略。永続的接続を確立し、受信確認をおこなう信頼性のある通信方式。

TTL

transistor-transistor logic の略。IC (集積回路) の論理回路方式の一種。

UDP

User Datagram Protocol の略。永続的接続を確立しないで送信し、受信確認を行わない、信頼性のない通信方式。

uid

UNIX で利用者を管理するための番号。

UNIX

ミニコンで使用できる軽いマルチユーザ用タイムシェアリング OS として、ベル研究所で開発された OS。

VME

IEEE 1014-1987,IEC 821 で標準化された計測器用バス

VXI

VMEbus eXtension for Instrumentation の略。VME バスを機能拡張したもの。

WS

Work Station の略。ワークステーション。パーソナル・コンピュータに対して技術用高機能計算機を意味していたが、今はその区別はあいまいになっている。

WWW

World Wide Web の略。CERN で開発された http というプロトコルにより、インターネットに接続されている計算機間で柔軟に情報を交換できる仕組み。現在は World Wide Web Consortium が標準化をおこなっている。(http://www.w3.org/)

Xlib

X Window で動作するプログラムを書くときに使用するライブラリ。

X Window

マサチューセッツ工科大学が開発したディスプレイやキーボードなどの利用法を標準化し、アプリケーションに移植性を持たせるためのシステム。

XRT/graph

KL Group Inc. (http://www/klgroup.com/)が開発したグラフ作成ツール。

アトミック命令

命令を実行している間に他の CPU により中断されないことが保証されている命令。

イーサネット　-> Ethernet

インタプリタ方式

命令を読み込み、逐次解釈しながら実行する方式。使うのは簡単だが性能が悪い。

ウィンドウ・システム

ウィンドウの外見や操作方法の規定および、規定に従ってウィンドウを管理するプログラム。

カーネル

OS の中で特権状態で実行する中核となる部分。

カーネル機能

OS の機能の中で特権状態で実行される機能。利用者のプロセスの切り替え、メモリ空間のマッピングの変更などを行う。

カーネル空間

特権状態にあるカーネルが利用できるメモリ空間で、全メモリとなる。これに対して利用者のプロセスから利用できるユーザ空間は、そのプロセスに割り当てられたメモリだけであり、異なるユーザ空間は相互に使用できないようになっている。

クライアント

サーバのサービスを受けて分担する業務を行う計算機。

クレート

CAMAC モジュールを挿入する電源付きの筐体。

サーバ

計算機がサーバとクライアントに別れ、協調して業務を行うシステムにおいて、クライアント計算機に対して種々のサービスを提供する役目の計算機

ジェネレータ方式

命令からプログラムを作成し、コンパイルして実行する方式。手順が面倒だが性能は良い。

シグナル

UNIX においてプロセス間で信号を伝える方式。

垂直分割

機能を上位と下位に分け、それぞれを独立な機能として設計すること。

水平分割

機能を同じレベルの複数の機能に分け、それぞれの機能間の関連をなくすことにより、簡単なシステムとなるように設計すること。

スケジューリング・クラス

スケジューラが実行可能状態になっているタスクから実行するタスクを選択するときのランク。通常は OS より優先度の低いタイム・シェアリング・スケジューリング・クラスで実行される。リアルタイム・スケジューリング・クラスは OS より優先度が高い。

セグメント

ルーターで区切られた LAN の一部。

セマフォー

複数のプロセス間で同期をとるための方法のひとつ

タイムシェアリング・スケジューリング・クラス

通常のプロセスが実行するレベル。OS のプロセスより優先順位が低い。

タスク

OS が CPU を割り当てる対象。

ダブル・バッファ方式

バッファを2面持ち、一方へ書き込んでいる間に他方から読み出しを行い、書き込みと読み出しの双方が終了したらバッファを切り替える方式。

チャンク

新型 ADC でのデータ・サイズを表すための単位。同時に AD 変換されたチャネル数個のデータの集まり。

特権状態

利用者プログラムでは実行できない命令を実行できる状態。

ハイウェイ

CAMAC クレート間、CAMAC クレートと外部装置を接続する高速データ伝送装置

ファイア・ウォール

ネットワークを通じての不正な侵入や妨害を防ぐために、外部ネットワークとの接点に設置する装置。

プロセス

OS がアプリケーションの実行を管理する単位。実行状態のアプリケーション。

ポート番号

IP 通信において、接続を管理するための番号。

ポーリング方式

処理を必要とする条件が発生したことを、繰り返し命令を実行して調べる方法。

マルチキャスト

同じデータを限定された複数の対象に送信する方式。対象を限定しないブロードキャストとは区別される。

メイン関数

アプリケーション・プログラム起動時に最初に呼び出される関数。

ユーザ空間

利用者のプロセスに割り当てられたメモリ。利用者のプロセスから利用できるメモリは、そのプロセスに割り当てられたメモリだけであり、異なるユーザ空間は相互に使用できないようになっている。

リアルタイム・スケジューリング・クラス

リアルタイム処理を必要とするプロセスが実行するレベル。OS のプロセスより優先順位が高い。

リング・バッファ

ダブル・バッファがバッファを2面持つのに対し、多数のバッファを持ち、巡回して割り当てる方式。

謝辞

本研究の場を提供していただいた核融合研究所の本島修先生に感謝いたします。実験の合間に激励の言葉をかけていただいたことが活力となりました。

この研究は本文で述べたように各種専門家の協力がなければ成立しませんでした。核融合研究所の山口作太郎氏は、システム全体を管理し実験の遂行はほとんど氏の力によるものです。松阪大学の奥村晴彦氏には、Web、クライアント・サーバなどのインターネット技術や、計算アルゴリズムなどで支援していただきました。職業能力開発大学校の寺町康昌氏、高エネルギー加速器研究機構の大須賀鬨雄氏は計測機器の設計、設定をしていただきました。日本サン・マイクロシステムズの江本雅彦氏(現在は核融合研究所)には、計算機システムのセットアップをしていただきました。FMC 株式会社の芳村幸治氏にはAD 変換装置の製作をしていただきました。

本研究でのシステム開発および性能試験に使用した計算機は、日本サン・マイクロシステムズの研究開発支援により借用したものです。同社および担当の安光正則氏に感謝いたします。

A アプリケーション開発支援

プロトタイプ・システム作成の課程でアプリケーション・プログラムの開発を支援する仕組みをいろいろと作成した。その中から、コマンド引数の扱いを容易にし、一貫した意味で使用するための支援と、定義情報の一貫性を維持するために、定義情報ファイルを共用する仕組みについて説明する。

1. コマンド解析支援ライブラリ

リスト 4.3、リスト 4.4、リスト 4.5、リスト 4.6 などに見られるような多量の設定情報を必要とする場合は、記述文法を定義し、コンパイラ・コンパイラを使ってコマンド解析プログラムを作成する。ここでは UNIX の標準コマンドである yacc と lex を使用した。このような言語開発環境が整備されていることは、第 3 章で述べた UNIX の優れた点のひとつである。

計算機言語とアプリケーション用言語を比べると大きな違いがある。アプリケーション用言語では再帰的な構造がないため制御は簡単になるが、キーワードが多量になり文法の大きさとしては大きくなる。このような特徴に注目してコマンド解析支援ライブラリ sys_yy パッケージを構築した。

(1) 定型処理が多いため、これをライブラリあるいはヘッダ・ファイルで提供する。

(2) キーワードの定義を一元化する。

(3) ユーザ・データ定義の一元化のため、include 機能を語彙解析レベルで提供する。

(2)と(3)の一元化とは、一個所で定義すれば、その他のところで必要な情報は自動生成されるか参照するだけでよくなるということであり、一貫性維持のための方式である。

1.1 キーワード定義の一元化

リスト A.1 に拡張短期記憶定義コマンド解析プログラム(ファイル名 stm/apl_yac.my)から本節の説明に必要な一部だけを示す。キーワードは、yacc には数値を表す token リテラル・シンボルとして、lex にはキーワード文字列と token リテラル・シンボルの対応関係として定義される必要がある。図 D.1 の

```
K_token(load, LOAD)
```

という文が、キーワード文字列 "load" がリテラル LOAD であるという定義である。

```
%{
途中省略
%token    Identifier Integer String Real

K_token(load, LOAD)
K_token(ch, CH)
K_token(table, TABLE)

%%
all
          : name load_table statements
          ;
load_table
          : LOAD TABLE ";" { x_apl_load_table(); }
          ;
省略
```

リスト A.1 拡張短期記憶定義コマンド言語文法の一部

拡張子が .my であるファイルにこの定義が記述されていると、拡張子が .y である yacc 用ファイルが自動生成され、その中にトークン定義として

```
%token LOAD
```

という文が生成され、さらに apl_keyword.h ファイルが自動生成され、そのキーワード定義構造体の初期値として

```
"load", LOAD,
```

という文が生成される。リテラル LOAD の値は yacc が生成する apl_yac.h の中で定義される。

1.2 ユーザ・データの一元化

リスト 4.4、リスト 4.5 で定義されている情報はアンプ・ゲインの設定、センサー定電流源の電流値やセンサーの較正表などである。これらの情報をアプリケーションの定義ファイルに直接記入しなければならないなら、複数の監視計算機で使用している共通の情報に対して修正が必要な場合に、整合性を保つことが非常に難しくなる。

このユーザ定義情報を一元管理できるようにするために、「include 構文」という機構を作成した。リスト 4.3 の include </opt/JK/etc/stm/ivcoil.channel>という句は、この位置に＜＞で示すファイルの内容が入ることを指示している。リスト 4.4 の include <CGR/C14252.tbl> も同じである。この機構により「情報を定義する」ことはファイルを作成することであり、「情報を参照する」ことは include 文で引用することとな

って明確に区別され、情報は一個所にのみ書けばよいので、ユーザ・データの一貫性が保証されることになる。

この include 構文は UNIX のプログラム開発環境として提供される m4 マクロ・プロセッサの include マクロに習ったもので、定義ファイルを必要とするアプリケーションの開発を容易にするように設計した。アプリケーション開発者は定義ファイルの文法を定義し、その解釈をするプログラムを作成しなければならない。この include 句を言語として解析するためには、どこに現れるかを文法的に決定しなければならない。しかし include という言葉は、データ定義のための言葉ではなく、概念レベルの異なる、定義ファイル内のどこに現れても良い言葉である。そこで include 句は文法レベルではなく、語彙解析レベルでの機能として提供することで、アプリケーション開発者は何も意識せずに include 句を使えるようにした。

アプリケーション開発者はコマンド語彙解析用 lex プログラムにリスト A.2 の記述を加え、提供する sys_yy ライブラリをリンクするだけで良い。include した定義ファイルの中で他のファイルを include してよいのは当然である。

```
%START          Nest NestEnd
%%
include{space}*[<]{space}*/{nest}+{space}*[>]      { BEGIN Nest; }
<Nest>{nest}+/{space}*[>]     { strcpy(g_nest_filename, yytext); BEGIN NestEnd; }
<NestEnd>{space}*[>]          { BEGIN 0;   x_sys_yy_nest(g_nest_filename); }
```

リスト A.2 include 機構導入のための lex 定義

リスト 4.5 に示したセンサーの変換テーブル内のエントリが、

(0.000218,　44.833)

のように、不要に思える括弧やカンマを付けて記述しているのは、このファイルがただの較正表テキスト・ファイルではなく、ユーザ定義ファイルの文法に合致する記述にしているからである。

2. コマンド引数処理ライブラリ

5 章で実験中に使用するコマンドの説明をした。コマンドの多くは、

```
stm_to_udp -f second.stm -a ysl
```

のように引数でいろいろな事項を伝える必要がある。コマンド引数はアプリケーション中のいろいろなところで必要とする、実行時に設定される情報を渡す機構であり、メイン関数が文字列配列として受け取ることになっている。

この説明には、好ましくない性質がある。すなわち、コマンド引数はグローバルな情報として与え、メイン関数が受け取るが、その意味を知っていて必要とするのは個々の機能を実現しているプログラムの部分である。コマンド引数は文字列で与えるが、個々の機能はそれぞれの内部データ表現で扱う。このようにタイミングとデータ表現に不一致があるのでこれを解決することが必要である。またコマンド処理はどのアプリケーションでも行わなければならないため、コマンド引数ライブラリ **arg** を作成してアプリケーション開発を容易にした。

上記コマンド **stm_to_udp** の場合は、まず

```
x_argd_init();
x_argd_define('A', c_arg_type_inaddr, (int) c_mcip_default_mcip_addr);
x_argd_define('P', c_arg_type_int, c_mcip_default_udp_port);
x_argd_define('N', c_arg_type_int, 1);
x_argd_define('F', c_arg_type_string, (int) "");
x_argd_define('S', c_arg_type_int, 0);
```

のように使用する引数、**A**、**P**、**N**、**F**、**S** についてそのデータ型とデフォールト値を定義しておく。メイン関数で

```
x_argd_read(v_argc, v_argv, "NPSX", "AF")
```

として、**A** と **F** が必須引数であること、**N**、**P**、**S**、**X** はオプション引数であることを指定してコマンドで与えられたパラメータを取り込む。引数を必要とするそれぞれのところで、

```
v_name.sin_addr.s_addr = (int) x_argd_get_inaddr_int('A');
```

のようにすれば、指定した引数の値が内部データ形式で得られる。ユーザが引数を与えたかどうかを調べる関数もある。

この **arg** ライブラリを用いることにより、コマンドがサポートする引数を属性とデフォールト値を含めて宣言的に定義できる。またそれぞれの機能に適した内部データ型でデータを取得できる。上の例では **A** 引数の値はネットワーク・アドレスであり、ユーザが IP アドレスで指定してもドメイン名で指定しても、プログラムの中では **sin_addr** 構造体で扱えるデータ型で取得できる。

このようにコマンド引数の処理が標準化されているので、異なるコマンドで使われるおなじ意味の引数を同じ文字で表すように統一することが容易になる。