



JAERI-Tech

JP0550128

2005-022



原研ERL-FELのための制御・データベースシステムの開発

2005年3月

菊澤 信宏

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは、日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合せは、日本原子力研究所研究情報部研究情報課（〒319-1195 茨城県那珂郡東海村）あて、
お申し越しください。なお、このほかに財団法人原子力弘済会資料センター（〒319-1195 茨城県那珂郡
東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division,
Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-
gun, Ibaraki-ken 319-1195, Japan.

原研 ERL-FEL のための制御・データベースシステムの開発

日本原子力研究所関西研究所光量子科学研究中心

菊澤 信宏

(2005 年 1 月 31 日受理)

原研 ERL-FEL ではパーソナルコンピュータ (PC) ベースのネットワーク分散処理型制御系を独自開発し、運転を行ってきた。その制御系は PC として日本電気株式会社(NEC)製 PC9801 シリーズで構成されたイーサネット LAN 上に構築された。しかしながら、PC は使用開始からすでに 10 年以上が過ぎて PC の規格が変わってしまったために修理や交換部品の入手が難しく、今後の機能の維持が困難となっていた。さらに、オペレーティングシステム (OS) のメーカーサポートの打ち切りにより修正プログラムが提供されなくなったため、多数の PC をネットワークで接続して使用することはネットワークセキュリティ上の問題があった。このため、従来の PC ベースのネットワーク分散処理型制御系の信頼性や安全性を向上させるために、高い信頼性が要求される産業用機器や家電などに多くの導入実績を持つ μ ITRON を OS として組み込んだコントローラを開発した。このローカルコントローラの開発とあわせて、将来のコンソール用計算機の更新が容易なように、多くの処理系で高い互換性を持つ Java 言語により制御用プログラムを新規に開発した。これらによって、高い信頼性と互換性が確保されたため、長期間の連続運転を前提とした信頼性の高い制御系の開発に成功した。

Development of Control and Data Processing System for JAERI ERL-FEL

Nobuhiro KIKUZAWA

Advanced Photon Research Center
(Tokai Site)

Kansai Research Establishment
Japan Atomic Energy Research Institute
Tokai-mura, Naka-gun, Ibaraki-Ken

(Received January 31, 2005)

A personal computer (PC) based distributed control system has been developed for Free Electron Laser (FEL) at Japan Atomic Energy Research Institute (JAERI) and operated since 1992. The control system was implemented on Ethernet LAN of PCs, Nippon Electric Company (NEC Corp.) PC-9800 series 32 bit personal computers. It became troublesome to maintain the control system, because many application programs did not work on outdated hardware interfaces and operating system. Furthermore, since security updates of the operating system (OS) were no longer provided, the problem was in network security when many PCs were connected with the LAN. We have to solve these problems and to improve the reliability and the safety of the control system, an ITRON-based controller was developed. In Japan, the ITRON is very popular and embedded in many products such as industrial instruments or household appliances that are demanded of its high reliability. When the local controller was installed, a new control program was developed by Java language which had high compatibility on many platforms so that replacement of the computer for consoles might become easy in future. High reliability and interchangeability have been successfully realized by them, and the control system made long continuous operation possible.

Keywords: FEL, Accelerator Control System, ITRON, Java, CORBA, Database System

目次

1.はじめに	1
2.原研 ERL-FEL 用旧制御系	3
2. 1 原研 ERL-FEL 装置の概要	3
2. 2 旧制御系(FELOWs)の概要	3
2. 3 FELOWs の改造	4
2. 4 FELOWs の問題点	4
3.新制御系	9
3. 1 新制御系の概要	9
3. 2 ローカルコントローラ	10
3. 3 制御用プログラム	10
3. 4 データベースシステム	12
3. 5 インターネット接続	13
3. 6 自動通報システム	13
4.まとめ	23
謝辞	23
参考文献	24
付録 A	25
付録 B	30

Contents

1 .	Introduction	1
2 .	Previous Accelerator Control System for JAERI ERL-FEL	3
2 . 1	Outline of JAERI ERL-FEL	3
2 . 2	Outline of Previous Control System for JAERI-FEL (FELOWS)	3
2 . 3	Improvement of FELOWS	4
2 . 4	Problems of FELOWS	4
3 .	New Control System	9
3 . 1	Outline of New Control System	9
3 . 2	Local Controller	10
3 . 3	Control Program	10
3 . 4	Database System	12
3 . 5	Internet Connection	13
3 . 6	Autoalarm System	13
4 .	Summary	23
	Acknowledgement	23
	References	24
	Appendix A	25
	Appendix B	30

1. はじめに

加速器の制御系の特徴として、広く分散して配置された多数の計測器からの信号を処理して多数の機器をコントロールする必要があるため、ネットワークでつながれた多数の計算機で構成されていることがあげられる。また、加速器の稼動期間は 10 年以上の長期となることが一般的であり、建設当初の試験期間も含めると制御系の稼動期間はさらに長期にわたることがあげられる。これまでの加速器制御系の例として、高エネルギー加速器研究機構の加速器制御の基本構造の変遷を表 1 に示す[1]。この表から、加速器の稼動期間と比べて短い期間で制御系の構成は大きく変化してきたことがわかる。加速器の制御系を設計する際には将来の改造や拡張を考慮しておくことが重要である。

最近では、コストや入手のしやすさなどの点から、これまでワークステーションなどで構成されていた制御系がパーソナルコンピュータ (PC) で置き換えられる例が増えているが、コンピュータネットワークのセキュリティ対策のためのオペレーティングシステム (OS) のアップデートやバージョンアップが頻繁に必要になるなど、多数の PC を管理するための時間や労力、費用も増えている。さらに、OS のバージョンアップによって機器のドライバの互換性が無くなるなどの問題が発生しやすく、動作確認のための作業も必要である。特に PC のハードウェアやソフトウェアの販売期間は短く、PC を更新するときにはそれらの互換性が問題となっている。

さらに、加速器は長時間にわたって安定して動作することが求められるが、そのためには機器を管理するローカルコントローラの信頼性が特に重要である。しかしながら、一般的に PC は 24 時間の連続運転や長期使用を想定して設計されていないため、駆動部品や電気部品レベルでの信頼性や安定性が確保されておらず、ハードウェアの故障が問題となっている。

原研自由電子レーザー用制御系では PC ベースのネットワーク分散処理型制御系を自主開発し、運用してきた。価格や入手のしやすさなどの点から市販品を利用することとし、各機器とのインターフェースに CAMAC (Computer Automated Measurement And Control)[2] を用い、CAMAC を制御するローカルコントローラとして日本電気株式会社(NEC)製 PC-9801 シリーズを使用してきた。しかしながら、PC-9801 シリーズが製造中止となり機能を維持することが困難となったため、PC の更新が必要となった。これまでの運用の経験から、多数の PC を使用する場合の管理コストや低信頼性の問題を解決するため、PC の代わりとなるローカルコントローラとして、ITRON を組み込んだ CAMAC コントローラを開発した。ITRON は高い信頼性が要求される産業用機器や家電などの分野で組み込みシステム用 OS として業界標準として利用されている OS である。例えば、ITRON はネットワーク接続機能などを持たせることができるために携帯電話などにも組み込まれているが、これらは OS やプログラムの修正が必要になることはほとんどなく、長期間にわたって安定して使い続けられている実績がある。これらの実績から判断して、管理コストが不要で、長期間の連続運転を前提とした信頼性の高いローカルコントローラを開発できたと考えられる。

ローカルコントローラを更新する際に、これまでの制御用プログラムを大幅に修正する必要が生じたため、制御プログラムを新規に開発した。ローカルコントローラは TCP/IP による通信機

能を持つため、オペレータが操作するコンソールは TCP/IP による通信機能を持った計算機であれば対応できる。このため、コンソール用計算機を処理速度の速い機種に更新した。制御プログラムの開発には、計算機の処理系に依存しない開発環境を重視し、Java 言語と CORBA (Common Object Request Broker Architecture) によって制御プログラムを開発した。

また、計算機の処理速度が向上したことによりデータベースシステムを導入することが可能となったため、計測データを効率的に保存・参照するために Web ブラウザからデータの検索ができる Web データベースシステムを開発した。これにより、ブラウザ上で簡単に計測データを検索・表示できるようになり、多くの処理系から利用することができるようになった。

本報告では第 2 章で従来の原研 ERL-FEL 用制御系について述べ、第 3 章で今回開発した新制御系について述べる。第 4 章はまとめとする。

表 1：大型加速器制御の基本構造の変遷 [1]

	80 年代	90 年代	00 年代
Console	CUI (text base)	Window (X+Windows)	Web?
基幹部	mini-computer	Unix	Linux?
Network	専用の高速回線	汎用の TCP/IP	CORBA?
Local-Controller	CAMAC	VME	PLC with Ethernet?

2. 原研 ERL-FEL 用旧制御系

2. 1 原研 ERL-FEL 装置の概要

自由電子レーザー(Free Electron Laser : FEL)とは、図 1 に示すように、電子ビームを電子加速器で加速してアンジュレータに導き、アンジュレータの交番磁場によって蛇行運動する電子ビームと光共振器に蓄えられた自発放射光との相互作用によってレーザー光を発生させる装置である。真空中を移動する自由電子を利用するため媒質中の発熱がなく、従来のレーザー光源と比べて強力なレーザー光を発生でき、電子ビームのエネルギーとアンジュレータのパラメータを変化させることによってレーザー光の波長を選択できるという特徴を持っている。原研自由電子レーザーは電子加速器に超伝導リニアックを用いて電子ビームのエネルギーを 17MeV まで加速し、中赤外域のレーザー光を発生する装置である。2000 年には世界最高出力 (2.34kW) でのレーザー発振に成功した[3]。以前はアンジュレータを通過した電子ビームはビームダンプに捨てられていたが、周回軌道を通して電子ビームを超伝導加速空洞に減速位相で再び入射するように改造し、2002 年には電子ビームの持つ運動エネルギーを高周波源として回収するエネルギー回収型リニアック(Energy Recovery Linac : ERL)の開発に成功した[4]。現在の原研 ERL-FEL 装置の概要を図 2 に示す。

2. 2 旧制御系(FELOWs)の概要

これまで使用してきた原研 ERL-FEL 用制御系 (FEL Operators Window System : FELOWs) は当研究グループ内で自主開発された加速器制御系であり、当初の開発の要求として以下のような項目が挙げられた。

- システムの拡張、改造が容易であること
- 特定のハードウェア／ソフトウェアに依存しないこと
- 保守が容易であること

これらの要求から、安価で入手が容易なパーソナルコンピュータ (PC) を利用し、機能の拡張が容易なネットワーク分散処理型制御系を開発した。加速器の建設開始から制御系の開発も行われ、1992 年から運用が開始された[5]。

加速器の建設が進んで制御対象となる機器が増えていくに従って PC の台数も増設され、完成時にはオペレータが操作する 2 台のコンソール用 PC と、ローカルコントローラとして CAMAC クレートに接続された 5 台の PC から構成され、それらが Ethernet によって接続されていた。

図 3 に FELOWs 制御系の構成を示す。CAMAC クレートには AO(Analog Output)、DO(Digital Output)、AI(Analog Input)、DI(Digital Input)などのモジュールを実装し、これらをローカルコントローラによって制御していた。ローカルコントローラは加速器室内の各機器の設置場所に分散して配置され、CAMAC モジュールから各機器の電源を操作できるよう統一されていた。PC はコストや保守性などの点から市販品を利用することとし、当時は日本国内で広く普及していた

NEC 製の PC9801 シリーズを採用した。

FELOWS は Windows3.1 用プログラムとして開発された。Windows3.1 はグラフィカルユーザーインターフェース (GUI) を採用した OS で、プログラムの操作方法が統一できるため、オペレータが操作を習熟する期間が短縮できるという利点を持っていた。プログラムの開発は Borland 社 Turbo Pascal を使用してオブジェクト指向プログラミングで行われ、プログラムの変更や保守が容易になるように設計された。プログラムはグラフィカルに表示するアプリケーション層、各機器の情報を管理するデバイス・サーバ層、各機器との I/O を行うデバイス層の三層に分けられており、機器の変更に対して制御プログラムの少ない変更で対応できるように考慮された。

計算機ネットワークはサーバおよびクライアントの両方の機能を持たせて対等な立場で相互に通信しながら処理を行うピア・ツー・ピア型のネットワークとし、通信プロトコルには Ethernet を採用した。ネットワークの形態は 10Base-2 によるバス型であった。

2. 3 FELOWS の改造

2000 年に原研自由電子レーザー用超伝導リニアックをエネルギー回収型リニアックに改造する際に、マグネットの大幅な増加に対応するため制御用ソフトウェアの変更を行った。まず、PC の OS を Windows3.1 から Windows95 にアップグレードし、ネットワーク接続が OS レベルでサポートされたのでネットワーク接続を Ethernet 接続から IP 接続に変更した。ネットワークの形態も、それまでのバス型では障害箇所の特定が困難だったため、10Base-T によるスター型に変更した。制御プログラム間のコマンドの送受信には通信速度の観点から User Datagram Protocol(UDP)を使用するように改造した。

超伝導加速器の温度モニタにはレコーダなどの GPIB 機器を利用していたが、インターフェースを CAMAC で統一するために CAMAC の GPIB モジュールからアクセスしていた。しかしながら、GPIB モジュールと他の CAMAC モジュールの同時アクセスが発生すると障害が発生するなどの問題があったので、GPIB 機器へのアクセスは Ethernet-GPIB 変換器 (National Instruments 社 GPIB-ENET/100) を用いるように変更した。

OS のバージョンアップにより共有ディレクトリが標準機能として利用できるようになったため、計測データの共有にはファイルの共有を利用した。

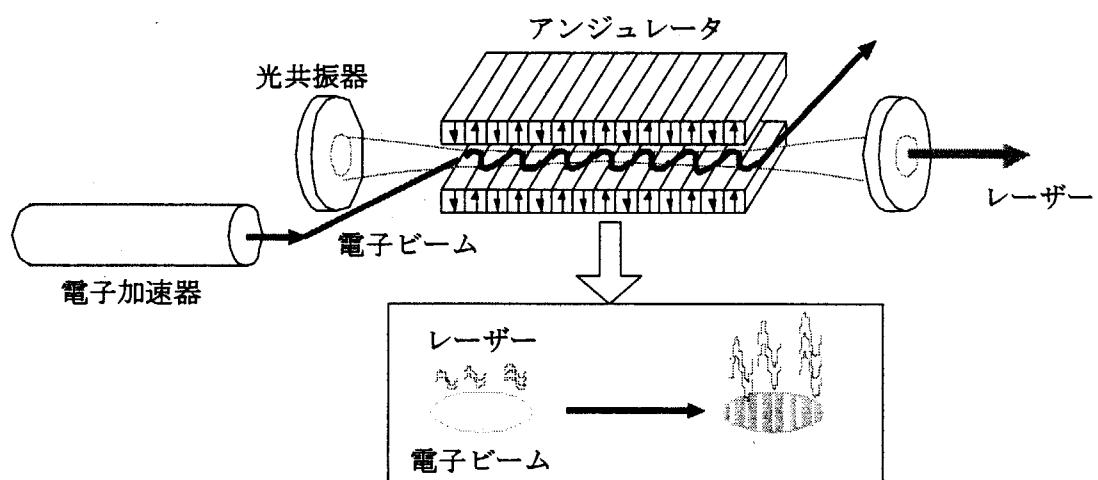
2. 4 FELOWS の問題点

ソフトウェアは加速器の改造にあわせて改良を行ってきたが、計算機などのハードウェアは開発当初からの構成で使用しており、大規模な追加投資は行ってこなかった。このため、PC9801 はハードウェアの仕様が変わってしまい、古い規格の製品は製造中止となつたため、修理や交換部品の入手が問題となっていた。例えば、使用していた CAMAC 用インターフェースボード(東陽テクニカ CC/9801-01)は NEC 製 PC9801 用の古い規格であるが、この製品は製造中止となつたため、入手不可能であった。現在の標準規格である PCI バス規格のインターフェースボード(東

陽テクニカ CC/PCI)に更新するためには、OS を WindowsNT/2000 にバージョンアップする必要があり、そのためのドライバを新規に開発する必要があった。

PC の OS はそれまでの 16bit OS から Windows2000/XP などの 32bit OS になったが、制御用プログラムは Windows3.1 上の 16bit アプリケーションとして開発されたため 32bit アプリケーションに改良するためには基本的な部分から大幅な変更が必要であった。たとえば、Windows3.1 はアプリケーションが自主的に制御を解放するノンプリエンプティブ方式のマルチタスクであり、ひとつのアプリケーションがエラーによって制御を開放しないと OS がハングアップしたようになり制御に支障をきたすが、Windows2000/XP は OS が強制的に複数のアプリケーションを切り替えるプリエンプティブ方式のマルチタスクになり、他のアプリケーションの影響を受けにくいうようになった。しかしながら、よりマルチタスクを意識したプログラムにする必要があり、いくつかの制御プログラムがハードウェアへアクセスする場合には排他処理を行うように改造する必要があった。また、Windows3.1 では同一のメモリ空間で動作するため制御プログラム間でデータの共有が容易であったが、Windows2000/XP では各プログラムが別のメモリ空間で動作するようになつたため、複数のプログラム間でデータを共有する場合には共有メモリを使用するよう改造する必要があった。これらの改造を行うためには制御プログラムの基本的な部分から変更が必要であり、プログラムを新規に開発するのと同様の工程が必要であったため、古い制御用プログラムが動作する Windows95 を使用する必要があり、PC を更新することができなかつた。しかしながら、最近ではセキュリティ上の安全性を維持するために OS のアップデートやバージョンアップなどが必要になつたが、既に Windows95 のメーカーサポートが打ち切られ、修正プログラムが提供されないという問題があつた。

これらの問題から、制御系全体の信頼性を向上させるためには、ハードウェアおよびソフトウェアの更新が必要となつてゐた[6]。



電子と光の相互作用によるレーザーの増幅

図 1：自由電子レーザーの原理

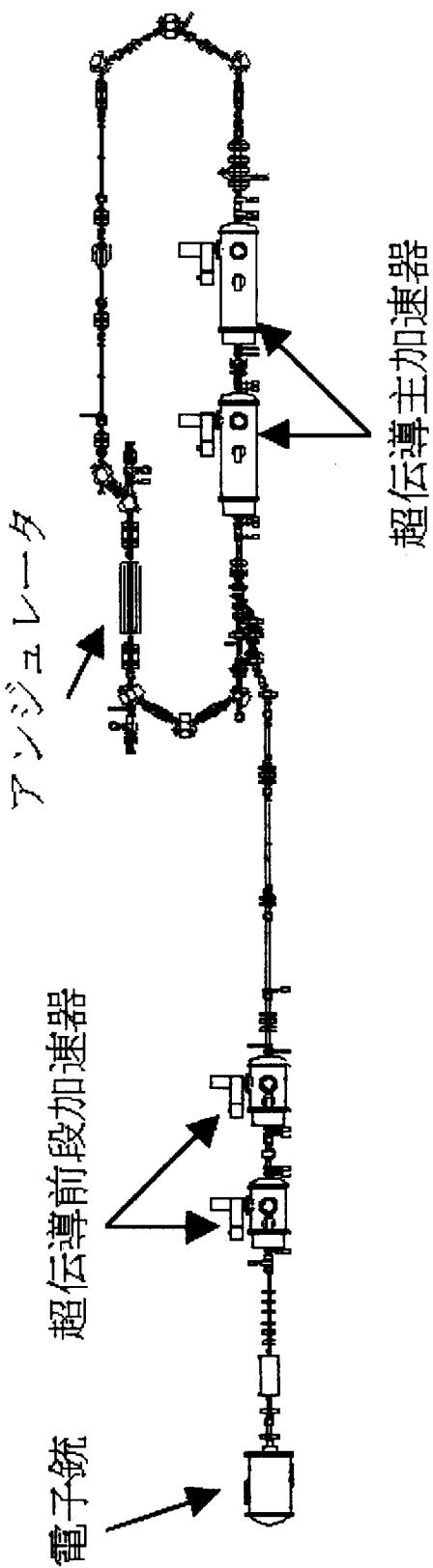


図 2 : JAERI ERL-FEL の構成

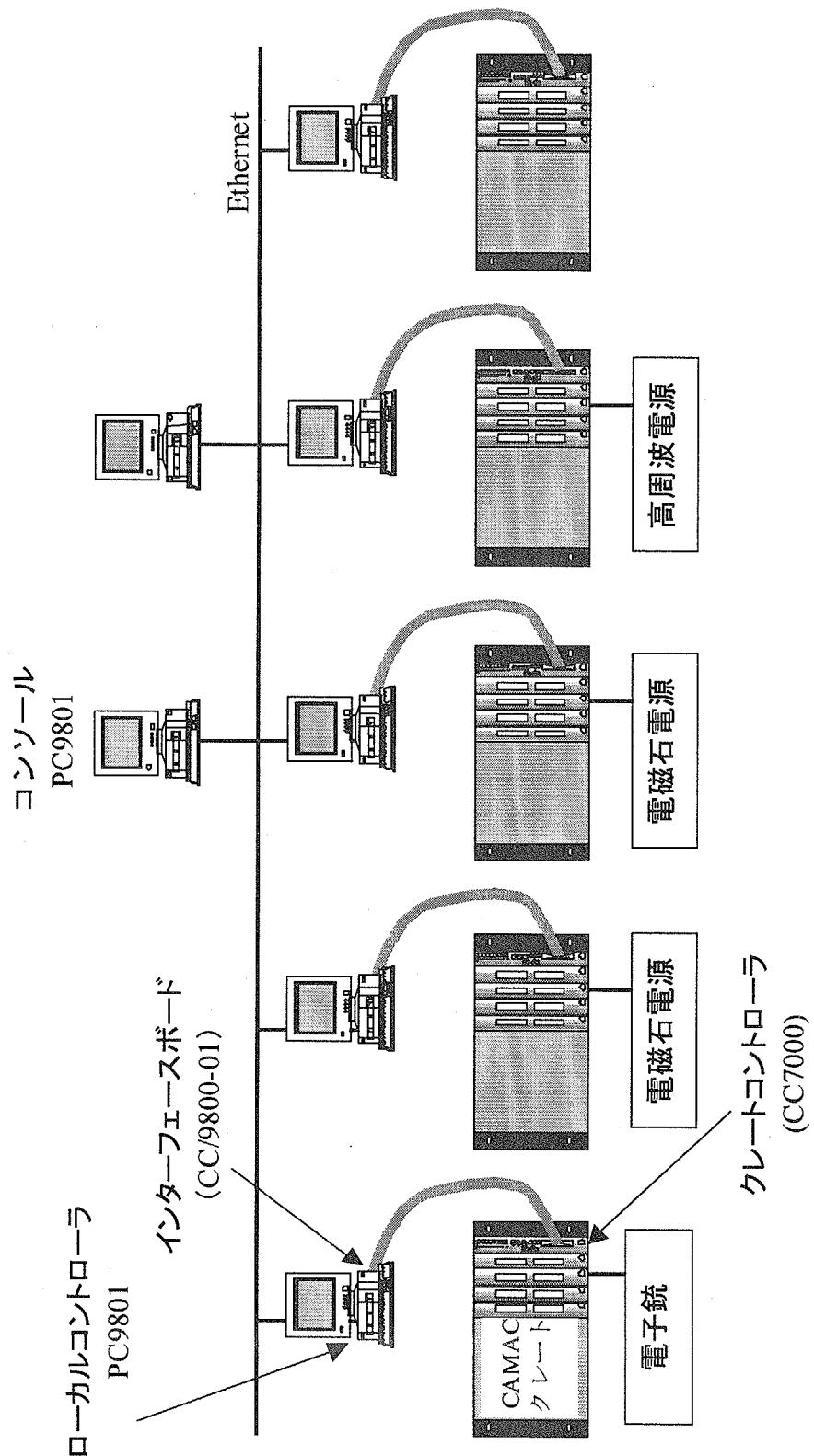


図 3 : FELOWS 制御系の構成

3. 新制御系

3. 1 新制御系の概要

制御系を更新する要求仕様として以下の項目を挙げた。

- 信頼性を向上させること。
- 特殊なハードウェア・ソフトウェアに依存しないこと。
- 将来の拡張や改造が容易であること。
- 機能を維持するための時間とコストを最小限にすること。

これらの条件を満たす選択肢として、以下の 3 方式を検討した。

(a) PCI-CAMAC 方式

CAMAC クレートコントローラと PCI 規格のインターフェースボードを使用し、PCI スロットを搭載した PC などから CAMAC を制御する方式である。PC は C 言語などの高級言語でプログラムを開発できるという利点がある反面、PC の信頼性の低さや管理コストの高さなどの欠点があった。

(b) Ethernet-CAMAC 方式

組込み用 OS を内蔵して通信機能を持った CAMAC クレートコントローラを使用し、TCP/IP によって CAMAC を制御する方式である。OS は ROM やフラッシュ RAM に内蔵されており、HDDなどを持たないため PC よりも故障の確率が低いと期待できた。しかしながら、PC と同様に OS のシャットダウン操作などが必要であり、開発環境が特殊になるという欠点があった。

(c) シーケンサ方式

産業分野で広く用いられているシーケンサを CAMAC の代わりに使用し、シーケンサの通信モジュールを利用して PC から機器を制御する方式である。近年、加速器分野でもシーケンサを利用する例が増えており[7]、ドライバソフトなどの整備も行われている。シーケンサは個々の I/O モジュールを組み合わせて使用されるため I/O モジュールの交換や追加が容易であり、安価である。しかし、現在使用している CAMAC をシーケンサに置き換えた場合、シーケンサから各機器までの配線を新たに設置する必要があり、動作確認までの作業を短期間に行うのは困難であった。

以上の 3 方式を検討した結果、(a)PCI-CAMAC 方式を基本として PC の代わりに μITRON を組み込んだ CAMAC コントローラを開発する方式を採用した。μITRON 上でも C 言語などの高級言語でプログラムを開発できる利点があり、ローカルコントローラとして PC を使用した場合に問題になる管理コストなども大幅に軽減できる。この方式は既存の CAMAC モジュールはそのまま使いながらローカルコントローラのみを更新する方式であるため、CAMAC モジュールから各電源までの既設の配線は変更する必要が無く、しばらく新旧のシステムを併用しながら動作確認を行うことができるため、更新作業のためのダウンタイムを最小限にできる。このコントローラについて以下に述べる。

3. 2 ローカルコントローラ

CAMACとのインターフェースとして、産業用途として多くの実績を持つ μ ITRONをOSとして採用したコントローラ(ニチゾウ電子制御 ND-MCU)をローカルコントローラとして選択した[8]。ローカルコントローラはCPUに日立製RISCプロセッサであるSH4、動作クロック200MHzを採用しており、拡張スロットとして3基のPCIスロットを装備している。リアルタイムOSである μ ITRONは高信頼性が要求される産業機器や家電などの組込みシステムに広く用いられている実績がある。また、HDDなどの駆動部品を用いていないために障害の発生する確率が低く、特別なシャットダウン操作無しに電源をオフできる。 μ ITRONはアップデートやバージョンアップなどが必要になることはほとんど無く、OSとしてWindowsなどを使ったPCと比べて管理コストを大幅に軽減できる。

ローカルコントローラには3基のPCIスロットがあるため、CAMACクレートコントローラ(東陽テクニカCC/7700)用のPCIインターフェースボード(東陽テクニカCC/PCI)をPCIスロットに挿し、パラレルケーブルでクレートコントローラと接続してCAMACを制御する。新しい制御系のハードウェアの構成を図4に示す。実際の運転に必要な機器は電子銃、電磁石電源、RF電源などであり、これまで主にマニュアルで制御していたRF電源もリモートで計算機制御できるように変更を行った。これらの制御のために5台のCAMACクレートが使われているため、5台のローカルコントローラを導入した。ローカルコントローラのPCIスロットにインターフェースボードを追加することによって、将来の拡張性も確保している。

ローカルコントローラからCAMACにアクセスするために、 μ ITRON上で動作するコントロールソフトを開発した。図5にローカルコントローラ内部のソフトウェア構成を示す。ローカルコントローラはEthernetポートを持ちTCP/IPで通信できるため、TCP/IPで送られた制御コマンドはローカルコントローラ内部のコントロールソフトからCAMACドライバを通して実行される。また、AOやDOなどのCAMACモジュールの中には設定値の読み出しができないものがあるため、設定値をメモリ上に保存する機能を設け、制御用PCから設定値を読み出せる機能を設けた。この設定値はバッテリバックアップされたメモリ上の設定値テーブルに保存されるため、ローカルコントローラの起動時に設定値を自動的に復元できるような機能を持たせることができた。この機能により、制御用PCが停止している状態でもローカルコントローラ単体で停電の前後で設定値を自動的に復元することが可能である。図6に電磁石電源操作用ローカルコントローラの接続の写真を示す。

3. 3 制御用プログラム

FELOWSの制御用プログラムはWindows3.1上の16ビットアプリケーションであったが、ローカルコントローラを更新するのにあわせて、制御用プログラムを新規に開発することにした。開発環境を選択する際に、将来の計算機の更新を考慮して制御プログラムが様々な処理系で動作することを重視し、開発言語としてJava言語[9]を選択した。Java言語はオブジェクト指向言語の一つで、機種依存性の非常に少ない言語であり、多くの処理系に対応したプログラムを開発で

きる。この利点から Java 言語は加速器制御の分野でも利用されており、ハードウェアに依存しない部分や、制御速度が必要でない部分は Java 言語で構築する例が増えている。個別の電源の操作画面などの表示には Cosylab 社の製作した Accelerator Beans (Abeans) [10]を利用した。Abeans は加速器の制御用に Java 言語を用いて開発された部品化されたプログラムであり、ドイツの ANKA 放射光施設や理化学研究所の R I ビームファクトリー (RIBF) などでも加速器制御プログラムに使われている[11]。Java 言語により開発したことで複数の開発者の間でプログラムの共有やコードの再利用が容易になり、制御プログラムの開発効率が向上することが期待されている。

制御コマンドの通信には分散オブジェクト技術である Common Object Request Broker Architecture(CORBA)[12, 13]を利用した。CORBA とは Object Management Group (OMG)が規定する分散システム環境でオブジェクト同士がメッセージを交換するための共通仕様であり、CORBA を使うことによって異機種分散環境で OS や開発言語に依存しないクライアント／サーバアプリケーションの開発が容易になる。これらの利点を生かし、加速器制御の分野でも CORBA を使った制御系の開発が行われている[14]。しかしながら、ローカルコントローラは CORBA プロトコルによる通信機能が利用できないため、制御用プログラムから CORBA プロトコルで送られた制御コマンドを TCP/IP に変換する必要があり、そのためのラッパリングソフトを開発した。コンソールの制御用プログラムから送られたコマンドはラッパリングソフトによって CORBA プロトコルから TCP/IP のコマンドに変換され、ローカルコントローラに送られる。このラッパリングソフトは各ローカルコントローラが管理する機器情報の定義ファイルを一括管理しており、起動時に各ローカルコントローラに対して設定値を送信する。電源の数が増えた場合などはこのパラメータ制御ファイルを変更するだけで対応できる。表 2(a)に制御項目および監視項目の一覧を示し、図 7 に電磁石電源制御用プログラムのパネル表示の例を示す。主な制御項目は、電磁石電源、電子銃、RF 電源の制御であり、コンソール用 PC からオペレータが操作したコマンドは CORBA プロトコルでラッパリングソフトに送られ、ラッパリングソフトによって TCP/IP に変換されてローカルコントローラに送られる。パネルに表示された値は、1 秒ごとにローカルコントローラから読み出されて更新されているため、他の PC から操作された場合もリアルタイムに値は更新される。制御系ネットワークに接続された複数の PC から操作できるため、例えば、無線 LAN で接続したノート PC などを使用することにより、実験室内を移動しながら機器の状態監視や操作が可能となり、装置の調整試験などを効率的に行うことができるようになった。主な監視項目は表 2(b)に示すように超伝導加速器の冷却系であり、1 分ごとに計測してデータベースに保存している。表 3 に制御系ソフトウェアの開発に使用したソフトウェアを示す。PC の OS には広く普及している WindowsXP を主に採用した。しかしながら、WindowsXP は OS のアップデートやバージョンアップなどが予定されるため、ソフトウェアの互換性の問題が発生することが予想される。この場合、OS をソースコードまで公開されていてフリーに使用できる Linux に変更することも考慮しているため、使用したソフトウェアは様々なプラットフォームに対応しているものを中心に選択しており、PC の OS を Linux に移行することも容易であると考えられる。

図 10 に制御プログラムの表示例を示す。現在は 2 台の PC をコンソールとして使用しており、

制御卓上に3台のモニタを並べて使用している。

3. 4 データベースシステム

原研自由電子レーザー用超伝導加速器に組み込まれている小型冷凍機は運転員なしの無人運転が可能であるが、故障時には冷凍機の交換作業が必要であり、故障の原因究明や故障の前兆現象を早い段階で発見する技術の開発などが求められていた。このため、冷凍機の温度やヘリウムガス流量などの冷凍機に関するデータのほかに、冷凍機に影響を与える可能性のある室温や冷却水温度、大気圧などの外部環境に関するデータを定期的に計測していた。この結果、大気圧の変動が同軸ケーブル電気長に与える影響の発見 [15]に寄与するなど、これらの外部環境に関する計測データの蓄積が電子ビームの変動の原因究明に有用であることがわかつたため、定期的に計測しているデータ点数も大幅に増加してきた。しかし、データ容量が大きくなるにつれてデータを有効に利用することが困難になってきたため、効率的なデータの参照のためにデータベースシステムを導入した。将来のデータベースシステムの更新なども考慮し、データベースサーバにはソースコードなどが公開されていて様々な処理系に移植されている MySQL Database Server[16]を採用した。

データベースシステムの全体構成を図8に示す。データベースには、運転操作ログ、設定値ログ、計測データなどが保存されている。運転操作ログには、オペレータが行った操作の時刻と内容がラッパリングソフトから自動的にデータベースに保存されるようになっている。すべての操作コマンドはラッパリングソフトを経由するため、どのコンソールから操作を行っても操作内容がデータベースに保存されるようになっている。設定値ログには、オペレータの要求に応じてその時点でのすべての設定値パラメータがデータベースに保存される。計測データには、気温や大気圧、超伝導加速器の温度や冷凍機の冷却能力などの常時計測されているデータが保存されている。

データベースサーバへのアクセスは様々な処理系から可能であるが、処理系にあわせたプログラムが必要になり、使用できる処理系を制限してしまうことになる。このため、PHP(PHP: Hypertext Preprocessor)言語[17]によってWeb上からデータベースにアクセスできるようなシステムを構築した。PHPはスクリプト言語のひとつであり、データベースへのアクセスが容易であるという特徴がある。Webサーバ側で行う処理をスクリプトで記述したことにより、処理の追加や修正も容易である。Webブラウザは現在ではほとんどの処理系で利用できるためWebクライアントの機種に依存せず、どのような処理系からでも計測データの表示が可能になった。データベースから過去のデータを検索してブラウザに表示させることも可能である。一例として、超伝導加速器冷却系のデータ表示の例を図9に示す。この図では、小型4K冷凍機のヘリウムガス流量と冷却能力、超伝導加速器のヘリウム槽内の圧力などの計測データの24時間分をグラフにして表示している。これにより、過去のデータをさかのぼって冷却能力の変動などを確認でき、故障事例の蓄積などにより故障の原因の解明に役立つと考えられる。そのほか、気温、冷却水温、大気圧などの冷却能力に影響を与える可能性のある外部環境のデータも計測しており、それらの日変化や季節変化などの影響を解析するのに役立っている。

3. 5 インターネット接続

現在は高速のインターネットが普及し、大容量のデータをネットワークで共有することも可能になっていることから、制御系ネットワークをファイアーウォールを介して研究所の構内ネットワークと接続した。このことにより、仮想プライベートネットワーク接続により自宅などからインターネットを経由して制御系ネットワークにアクセスすることができるようになり、Web ブラウザによる計測データ表示を利用することによって休日や夜間などでも自宅などから機器の動作確認や故障診断が可能になった。

現在、地理的に遠く離れた場所から実験装置を操作し、計測データを解析する遠隔実験の要求が高まっている。これも、制御系ネットワークをインターネットに接続したことによって、セキュリティ上の配慮は必要であるが、将来は他の研究施設などの遠く離れた場所から実験装置を操作したり測定データを解析したりする遠隔実験なども可能になった。利用者の使用する計算機環境を制限しないためにも、制御用プログラムを Java 言語と CORBA によって構築したことにより遠隔操作のためのプログラムの開発も容易になったため、遠隔実験のための拡張も可能である。

3. 6 自動通報システム

Web ブラウザによるデータ表示により自宅から冷凍機の監視も可能となったが、Web ブラウザによるデータ表示だけでは機器の故障の発見が遅れることになるため、携帯電話を機器の故障時の緊急の連絡通報手段として活用することを考えた。現在では携帯電話などの移動携帯端末から電子メールを送受信できるようになっており、たとえば、計測しているデータから機器の故障が想定される場合には、制御系ネットワークから自動的に電子メールを送信し、管理者の移動携帯端末へ自動通報するシステムを開発した。例として、超伝導加速器用の冷却系の自動通報システムを試験運用している。この冷却系は年間を通じて無人運転を行っており、故障時には早期の対応が必要であるため、1分ごとに冷凍機の冷却能力を監視して冷却能力がゼロにまで下がった場合には管理者に自動的に電子メールで通報している。電子メールは途中のネットワークの障害によって着信までの時間にはばらつきはあるものの、ほぼ1分以内に送信されており、機器の監視の負担を軽減することに寄与している。これによって冷凍機の故障時の液体ヘリウムの総蒸発量を抑えることができ、超伝導加速器の長期間にわたる低温保持に大きく寄与し、このシステムの有用性を確認した。

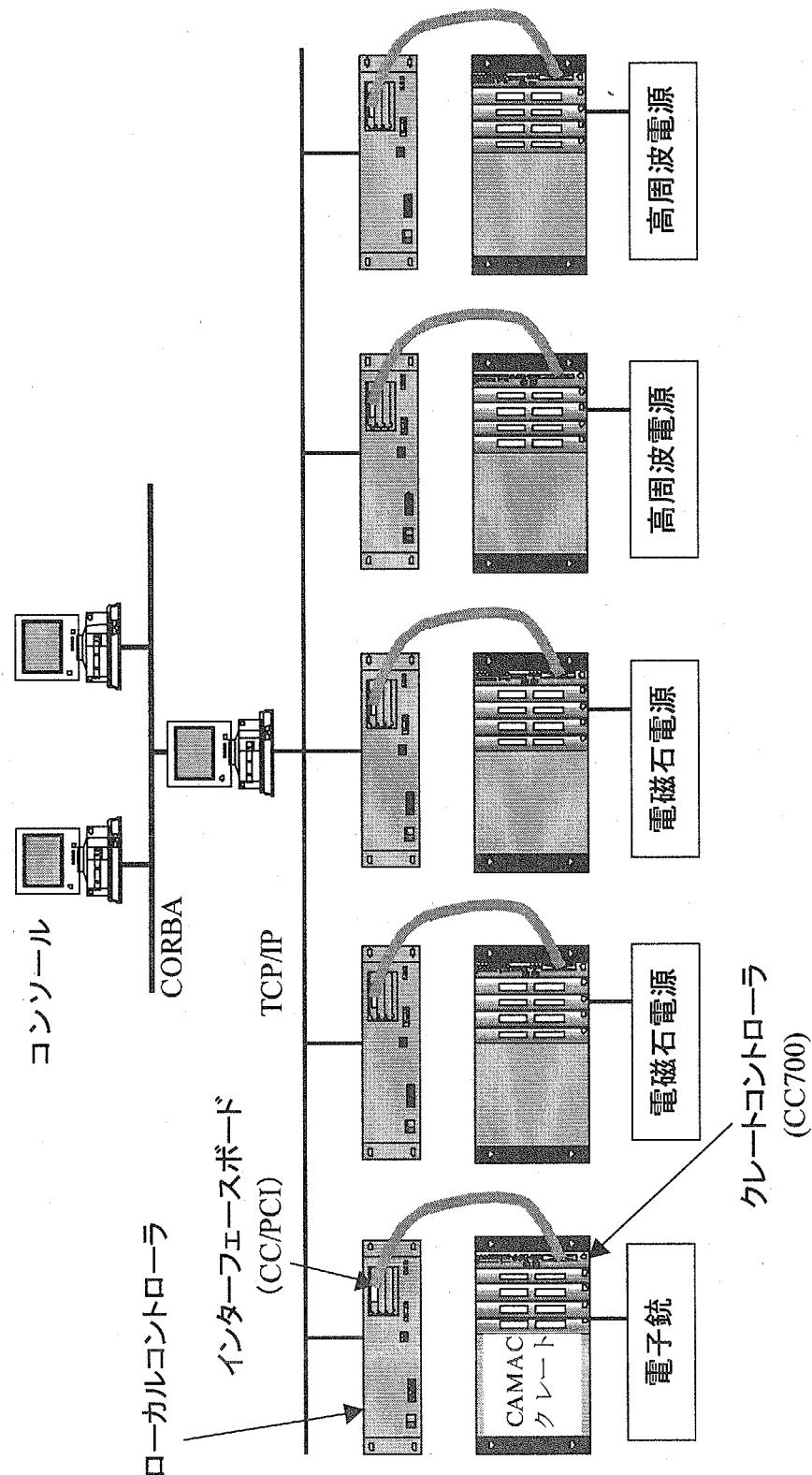


図 4: ノードウェア構成

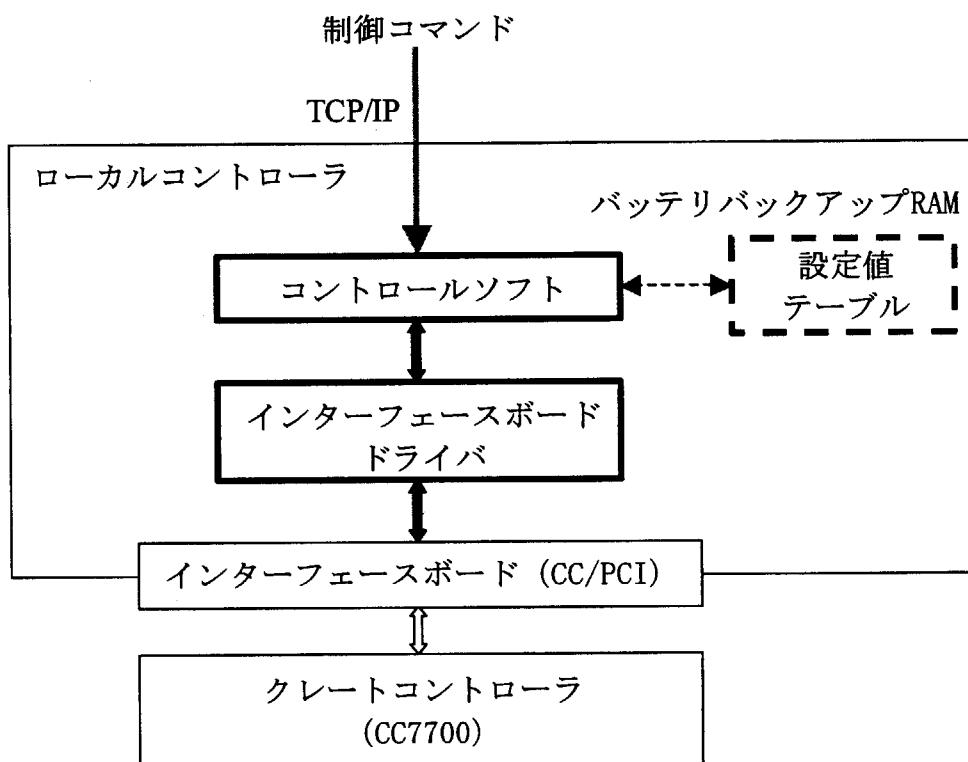


図 5：ローカルコントローラのソフトウェア構成

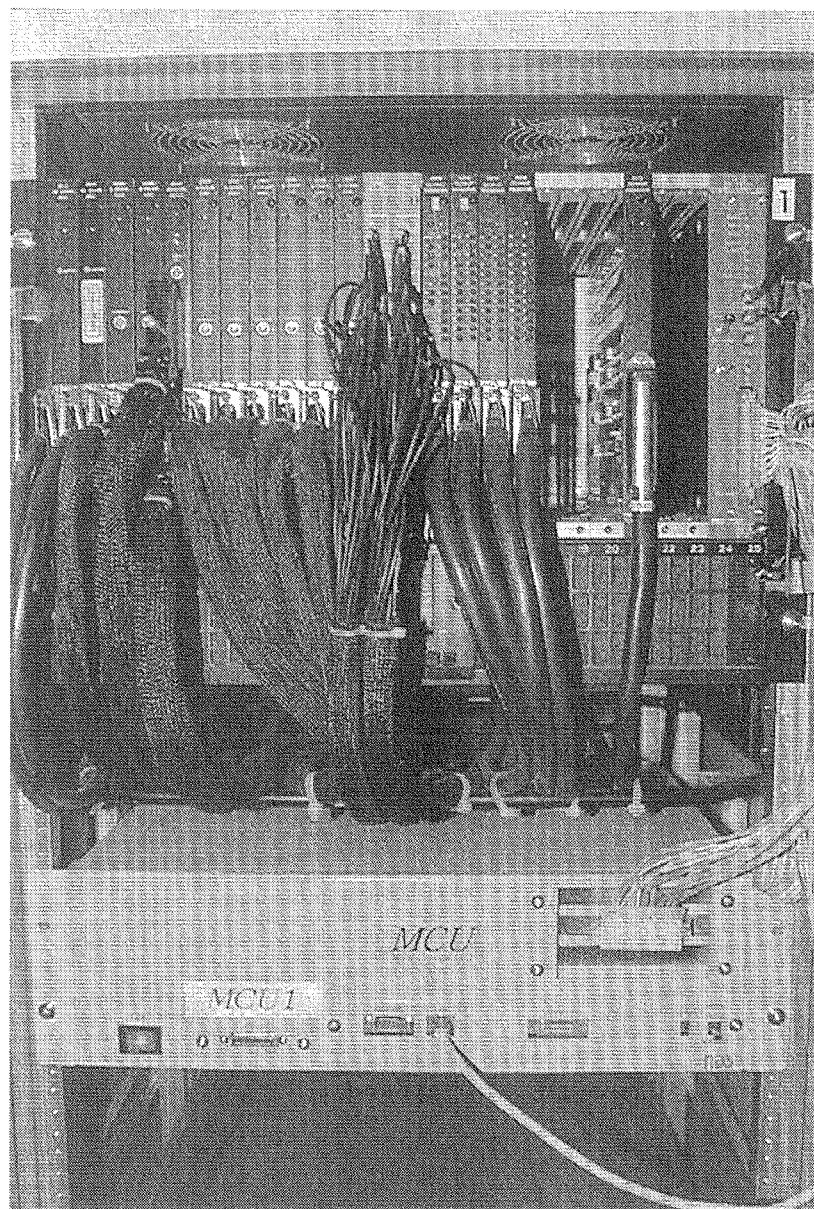


図 6：電磁石電源操作用ローカルコントローラの接続の様子。

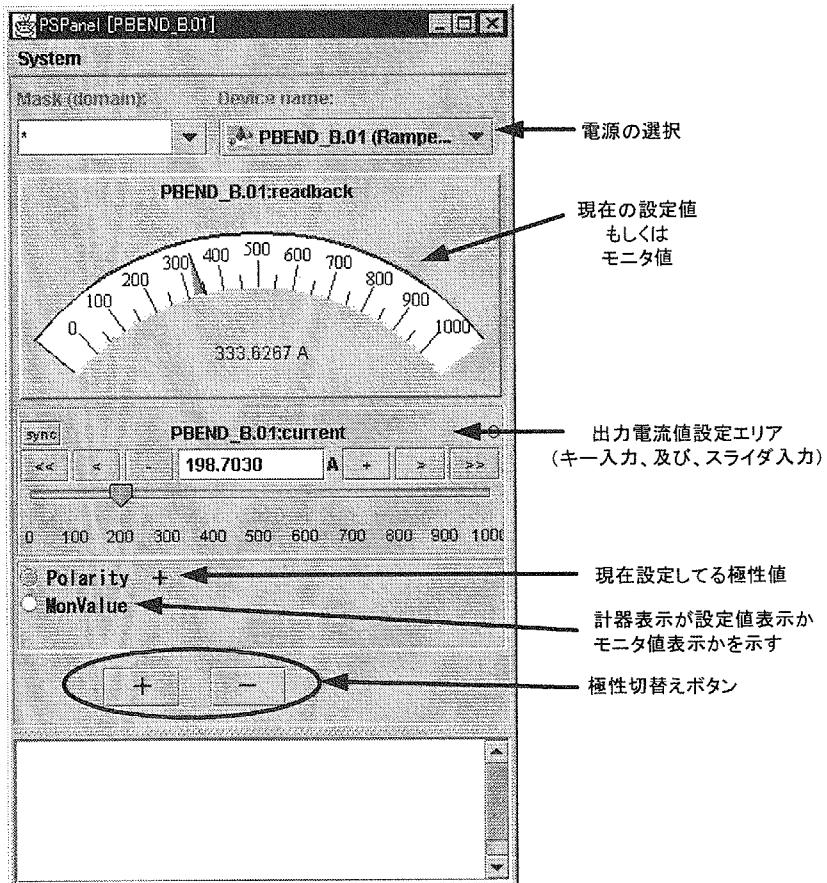


図 7：電磁石電源制御プログラムの表示例

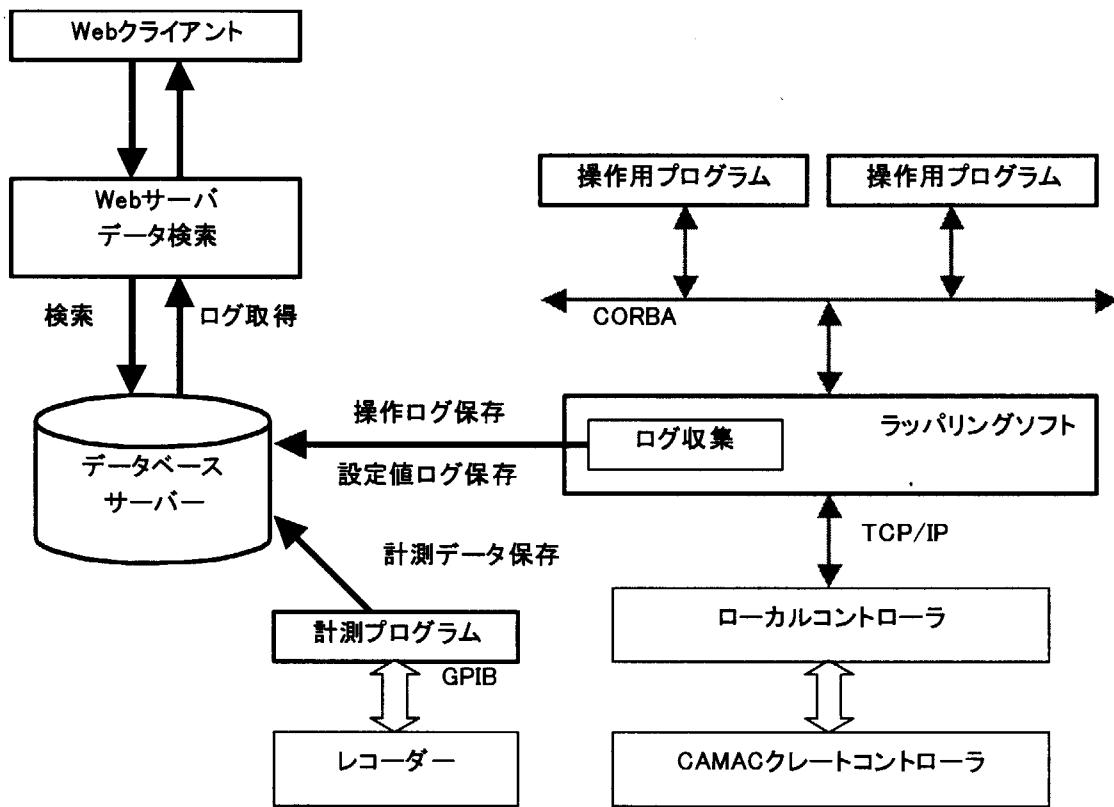


図 8：データベースシステムの全体構成

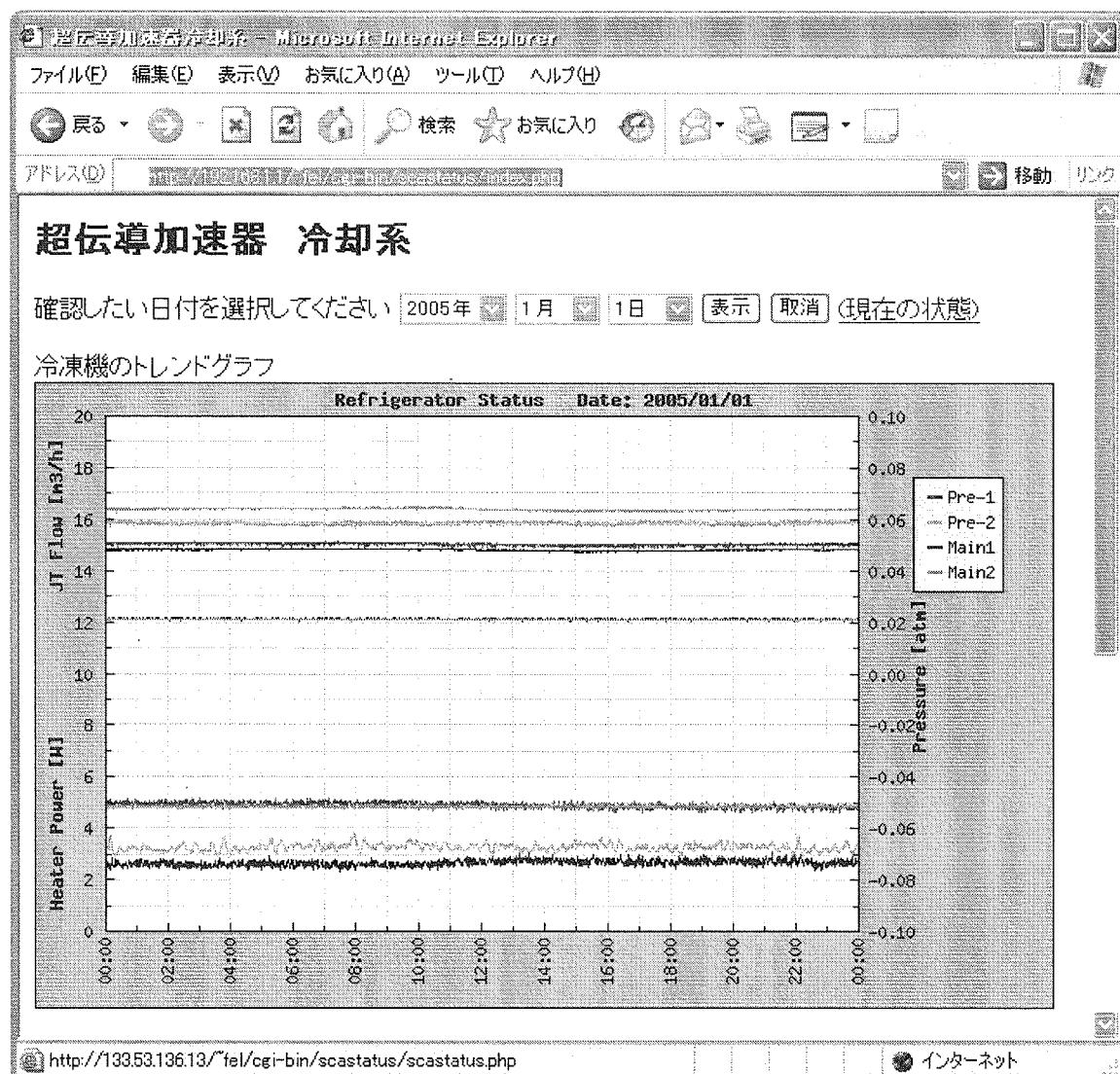


図 9：ブラウザによる超伝導加速器冷却系の計測データ表示例

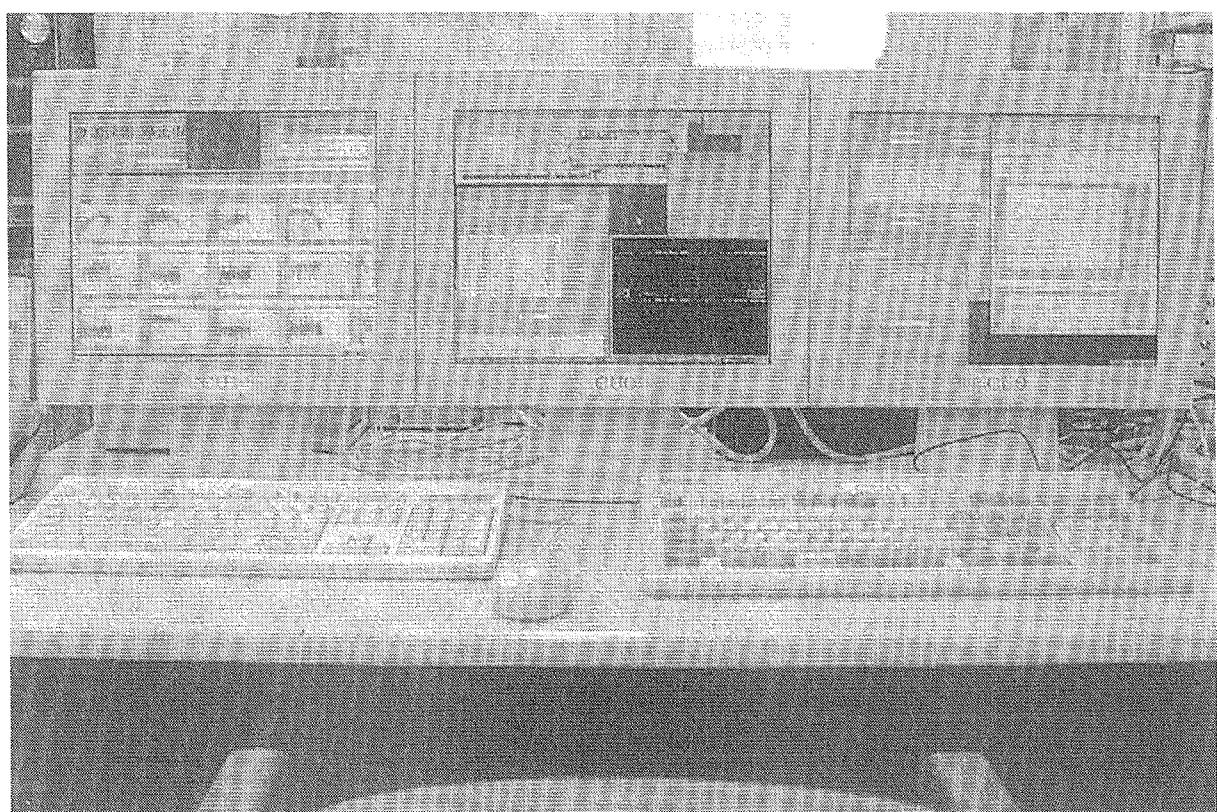


図 10：制御用プログラムの使用例

表 2：制御項目および監視項目一覧

(a) 制御項目一覧

装置名	信号名	信号形態	信号点数
ビーム輸送系電磁石電源	電流値設定	0~10V DC	118 点
	極性切替	無電圧接点	34 点
電子銃	Bias 電圧	0~10V DC	1 点
	Filament 電流	0~10V DC	1 点
	マクロパルス ON/OFF	無電圧接点	1 点
RF 電源ローレベル コントローラ	RF ON/OFF	無電圧接点	5 点
	位相制御 ON/OFF	無電圧接点	5 点
	位相制御	0~10V DC	5 点
	位相制御ゲイン	0~10V DC	5 点
	振幅制御 ON/OFF	無電圧接点	5 点
	振幅制御	0~10V DC	5 点
	振幅制御ゲイン	0~10V DC	5 点
	ループオフセット	0~10V DC	5 点
	チューナー制御 ON/OFF	無電圧接点	5 点
	チューナーオフセット	0~10V DC	5 点
プロファイルモニタ	スクリーン ON/OFF 操作	無電圧接点	30 点
	カメラ切替	無電圧接点	30 点

(b) 監視項目一覧

装置名	信号名	信号形態	信号点数
超伝導加速器	温度	アナログ電圧	24 点
	ヘリウム槽圧力	アナログ電圧	4 点
小型ヘリウムガス冷凍機	ヘリウムガス流量	アナログ電圧	4 点
超伝導加速器 圧力調節器	ヒータ電流	アナログ電圧	4 点
インターロック制御盤	運転表示	無電圧接点	1 点
温度センサー	実験室内温度	熱伝対	3 点
気圧センサー	気圧	アナログ電圧	1 点

表 3：使用したソフトウェア

ローカルコントローラ	OS	μ ITRON
	プログラム言語	C 言語
クライアント PC	OS	WindowsXP
	プログラム言語	Java 言語
	CORBA	JacORB[18]
ラッピングソフト	OS	WindowsXP
	プログラム言語	C++言語
	CORBA	TAO[19]
データベースサーバ	OS	WindowsXP
	データベースサーバ	MySQL
Web サーバ	OS	WindowsXP
	httpd	Apache、Perl、PHP

4. まとめ

原研自由電子レーザー用制御系を更新するために μ ITRON を組み込んだ CAMAC コントローラを開発し、原研 ERL-FEL 用制御・データベースシステムの運用を開始した。この開発により得られた結果を以下にまとめる。

- μ ITRON は組み込みシステム用に開発されたリアルタイム OS であることから現在の PC よりも信頼性が向上し、リアルタイム制御性が向上した。
- 将来にわたって PC を更新する必要が無くなり、現在の CAMAC システムの機能を維持しつづけることができるようになった。
- コンソール用 PC とローカルコントローラを機能分離することができたため、コンソール用 PC を高性能の PC に更新できた。計算機の性能が向上したため、制御用プログラムは Java 言語／CORBA を使用して新規に開発を行った。これにより、将来の PC の更新にも容易に対応できる。
- 電子メールを活用した故障時の自動通報システムによって早期の故障対応が可能となり、超伝導加速器を長期間にわたって低温保持するのに大きく寄与した。
- 計測データの効率的な管理のためにデータベースシステムを導入し、過去の計測データもウェブ上から簡単に参照できるようになった。今後の計測データの蓄積により、機器の故障の原因究明や、電子ビームの変動の原因究明に大きく貢献すると考えられる。

本制御・データベースシステムの開発により、高い信頼性で原研 ERL-FEL 装置を運転し、機器の監視をすることが可能となった。このことは、原研 ERL-FEL 装置の性能向上に役立ち、光源の応用研究に大きく貢献するものである。

謝辞

本研究を行うにあたり、有意義な議論および教示をいただきました自由電子レーザー研究グループの皆様に感謝を申し上げます。

参考文献

- [1] Kamikubota, et al, Proceedings of the 26th Linear Accelerator Meeting in Japan, Tsukuba, Aug. 1- 3, 2001, p.273-275.
- [2] IEEE, "IEEE Standard Modular Instrumentation and Digital Interface System (CAMAC) (Computer Automated Measurement and Control)", IEEE Std 583-1975.
- [3] N. Nishimori, et al., Nucl. Instr. Meth. A475 (2001) 266-269.
- [4] R. Hajima, et al., Nucl. Instr. and Meth. A507 (2003) 115-119.
- [5] M. Sugimoto, Nucl. Instr. Meth. A331 (1993) 340-345.
- [6] N. Kikuzawa, Proceedings of the 28th Linear Accelerator Meeting in Japan, Tokai, July 30-Aug. 1, 2003, p.437-439
- [7] R. Kato, et al., Proceedings of the 27th Linear Accelerator Meeting in Japan, Kyoto, Aug.7-9, 2002, p365-367.
- [8] N. Kikuzawa, Proceedings of the 14th Symposium on Accelerator Science and Technology, Nov. 11-13, 2003, p269-271.
- [9] C. Horstmann, "Gary Cornell, Core Java 2: Fundamentals", Sun Microsystems Press Java Series.
- [10] G. Tkacik, et al., "Java Beans of Accelerator Devices for Rapid Application Development", PCaPAC99 workshop, Tsukuba,(January 1999).
- [11] T. Tanabe et al., The 13th Symposium on Accelerator Science and Technology, Osaka, (October 2001) p.108.
- [12] S. Vinoski, "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments", IEEE Communications Magazine, Vol. 14, No. 2, February, 1997.
- [13] T. J. Mowbray and W. Ruh, "Inside CORBA: distributed object standards and applications", Addison Wesley Object Technology Series(1998)
- [14] S. Kusano et al., Proceedings of International Conference on Accelerator and Large Experimental Physics Control Systems, Trieste, Italy, 1999, p.535.
- [15] R. Nagai, et al., , Proceedings of the 28th Linear Accelerator Meeting in Japan, Tokai, July 30- Aug. 1, 2003, p.315-317
- [16] M. Widenius and D. Axmark, "MySQL Reference Manual", O'Reilly & Associates, Incorporated (2002).
- [17] Rasmus Lerdorf and Kevin Tatroe, "Programming PHP", O'Reilly & Associates, Mar 2002.
- [18] Gerald Brose, Andreas Vogel, Keith Duddy, "Java Programming with CORBA, 3rd ed.", John Wiley and Sons, New York, January 2001.
- [19] D. C. Schmidt, A. Gokhale, T. Harrison, and G. Parulkar, "A High-Performance Endsystem Architecture for Real-time CORBA," IEEE Communications Magazine, vol. 14, February 1997.

付録 A

リスト 1: JaeriFEL.idl ファイルの内容

```

module JaeriFEL {
    enum MONI_TIME {
        TIM1MMS, TIM5MS, TIM10MSEC, TIM20MSEC, TIM50MSEC,
        TIM100MSEC, TIM200MSEC, TIM500MSEC,
        TIM1SEC, TIM2SEC, TIM5SEC, TIM10SEC, TIM20SEC, TIM30SEC,
        TIM1MIN, TIM2MIN, TIM5MIN, TIM10MIN, TIM20MIN, TIM30MIN, TIM60MIN
    };

    // description of read-only double channel
    struct RODoubleDesc {
        double dMin;           // minimal value (Add)
        double dMax;           // maximal value (Add)
        double dGraphMin;      // minimal value on graph
                               // (now dMin=dGraphMin)
        double dGraphMax;      // maximal value on graph
                               // (now dMax=dGraphMax)
        double dWarningMin;    // below this warning color is enabled
        double dWarningMax;    // above this warning color is enabled
        double dAlarmMin;      // below this alarm color is enabled
        double dAlarmMax;      // above this alarm color is enabled
        string tUnit;          // units
        string tFormat;         // format of value
        string tDesc;           // description of value
        double dMinStep;        // minimal possible change in value
    };

    // description of read-only double channel
    struct RWDoubleDesc {
        double dMin;           // minimal allowed value
        double dMax;           // maximal allowed value
        double dGraphMin;      // minimal value on graph
        double dGraphMax;      // maximal value on graph
        string tUnit;          // units
        string tFormat;         // format of value
        string tDesc;           // description of value
        double dMinStep;        // minimal possible change in value
    };

    /* Alarm Level ID */
    // Status Indicate
    // NORMAL_LEVEL is GREEN on LED
    // WARNING_LEVEL is YELLOW on LED
    // LALARM_LEVEL is RED on LED
    // HALARM_LEVEL is RED on LED
    // ERROR_LEVEL is RED on LED
    enum ALARM_LEVEL { NORMAL_LEVEL, WARNING_LEVEL, LALARM_LEVEL, HALARM_LEVEL,

```

```

ERROR_LEVEL };
```

```

enumACI_BOOL { ACI_FALSE, ACI_TRUE };
```

```

typedef sequence <string> stringSeq;
```

```

// PS and RF Status in IStat on JaeriFEL system is fixed data
//-----
// PS, RF and Other
// 0bit:Polarity + :NORMAL_LEVEL
// 1bit:MonValuee :NORMAL_LEVEL
// PS, RF and Other
/* Status Information Structure */
typedef struct STRC_FEL_STAT {
    double dCur;           /* Current Data(Physical) */
    double dVol;           /* Voltage Data(Physical) */
    long lCur;             /* Current Data(Binary) */
    long lVol;             /* Voltage Data(Binary) */
    short sOn;              /* ON Status 0:OFF 1:ON */
    short sRamp;            /* Ramp Status */
    short sZero;            /* Current Zero Status */
                           /* 0:Not Zero 1:Zero */
    short sPolarity;        /* Polarity Status (FEL:[0:+ 1:-] -> [0:- 1:+]) */
    short sRemote;          /* Remote Status 0:Remote 1:Local*/
    short sPermission;      /* Permission of the Operation */
                           /* 0Bit=0:Permission 1:Un-Permission */
                           /* 1Bit=0:Use 1:Un-Use */
    long lStat;             /* Status */
                           /* 0bit:Polarity + (0:- 1:+) */
                           /* 1bit:Mon Value (0:Setting Value 1:Monitor value) */
    long lErr;              /* Error Code */
} tFELSTAT;
```

```

// Put Command Status Information Structure
typedef struct STRC_FEL_PUT_STAT {
    double dSet;           /* Setting Value Data(Physical) */
    double dStart;          /* Start Value(Physical:for Future) */
    long lSet;              /* Setting Value Data(Binary) */
    long lStart;            /* Start Value Data(Binary:for Future) */
    double dStep;           /* for Future:Step Time(Unit:msec for Future) */
    double dTime;            /* for Future:Goal Time(Unit:msec for Future) */
} tFELPUTSTAT;
```

```

typedef struct STRC_FEL_RESTORE_INFO {
    string tName;
    string tGroup;
    long lRestore;
} tFELRESTOREINFO;
typedef sequence< tFELRESTOREINFO > felrestoreinfoSeq;
```

```

/* Add for Poller & Logger(2003.10.16:for future) */
```

```

typedef struct STRC_FEL_STAT_INFO {
    string          tName;
    string          tGroup;
    string          tDevType;
    tFELSTAT       tFELStatInfo;
    tFELPUTSTAT   tFELPutStatInfo;
} tFELSTATINFO;
typedef sequence< tFELSTATINFO > felstatinfoSeq;

typedef struct STRC_FEL_COMM_INFO {
    string          tName;
    string          tGroup;
    string          tDevType;
    long           IBit;
} tFELCOMMINFO;
typedef sequence< tFELSTATINFO > felcommdinfoSeq;

interface PCRefServer {
    /* return Group Name List */
    stringSeq      retGetGroupList();

    /* return IOR in Group */
    stringSeq      retGetGroupIOR( in string tGroup );

    /* Add 2003.11.13 for Restoration Window */
    /* put Restoration data */
    short         retPutRestoration( in felrestoreinfoSeq tFELRestore );

    /* get Restoration data */
    felrestoreinfoSeq  retGetRestoration( in string tGroup );

    /* for Maintenance */
    string        retGetMCUIOR();

    /* Store Status log */
    short         retPutDBStore();
};

/*-----*/
/* Device Interface */
/*-----*/
interface Device {
    // GUI sends tMoni when a monitoring time was changed
    // and a sub-window was closed.
    // JaeriFEL is fixed data(always=TIME1SEC)
    attribute  MONI_TIME  tMoni;

    string          retName();
    stringSeq      retGroup();
}

```

```

        stringSeq      retDevType();
};

/*
/* FEL Device Interface */
*/
interface FELDevice : Device {
    // Caution:tCurDesc != tSetDesc
    // Information on actual current
    // GUI sends tCurDesc when a alarm or warning was changed
    // and a sub-window was closed.
    attribute RODoubleDesc  tCurDesc;

    // Information on actual voltage(for Future Plan)
    attribute RODoubleDesc  tVolDesc;

    // Information on setting current
    attribute RWDoubleDesc   tSetDesc;

    /* Command */
    short      retPutPolPlus();
    short      retPutPolMinus();

    short      retPutAmp( in double dAmp );

    /* for Future */
    // short   retPutRamp( in double dAmp, in double dGoal, in double dStep ):

    tFELSTAT retGetStatus();
    tFELPUTSTAT retGetPutStatus();
};

typedef struct STRC_CAMAC_INFO {
    short      sCrate;
    short      sStation;
    short      sSubAddr;
    short      sFunc;
    long       lData;
} tCAMACINFO;
typedef sequence< tCAMACINFO > camacinfoSeq;

/* for Maintenance */
interface          MCU {
    void      MCU_Reset( in short sMCUNo );
    short     MCU_TbIClr( in short sMCUNo );
    short     CAMAC_Init( in short sMCUNo, in short sCrate1, in short sCrate2,
                           in short sCrate3 );

    camacinfoSeq CAMAC_Get( in short sMCUNo, in camacinfoSeq tCAMAC );
};

```

```
short      CAMAC_Put( in short sMCUNo, in camacinfoSeq tCAMAC );

short      CAMAC_Put_LI( in short sMCUNo, in camacinfoSeq tCAMAC );
short      CAMAC_Put_RS( in short sMCUNo, in camacinfoSeq tCAMAC );
short      CAMAC_Put_EI( in short sMCUNo, in camacinfoSeq tCAMAC );
short      CAMAC_Put_I( in short sMCUNo, in camacinfoSeq tCAMAC );
short      CAMAC_Put_Z( in short sMCUNo, in camacinfoSeq tCAMAC );
short      CAMAC_Put_C( in short sMCUNo, in camacinfoSeq tCAMAC );
camacinfoSeq CAMAC_GetBus( in short sMCUNo, in camacinfoSeq tCAMAC );
};

/* Add for Poller & Logger(2003.10.16:for future) */
interface    Data_Logging {
    short     retGetFELStatus( in felstatinfoSeq tFELStatInfo );
    short     retPutFELCommand( in felcomdinfoSeq tFELCommand );
};

};
```

付録 B

リスト 2 : Java 言語による制御プログラムの例

```

/*
 * JaeriFEL Test-Tool */
/* (Client) */

import java.io.*;
import java.util.*;
import java.net.*;
import org.omg.CORBA.*;

public class FELClient
{
    /* Connect PC Ref Server */
    static JaeriFEL.PCRefServer getPCRefServer( org.omg.CORBA.ORB orb, String[] args )
                                                throws org.omg.CORBA.UserException
    {
        org.omg.CORBA.Object objRefServer = null;
        String ior = null;

        try
        {
            String refFile = "../GUI/PCRefServer.ior";
            java.io.BufferedReader iorReader = new java.io.BufferedReader(
                new java.io.FileReader(refFile) );
            ior = iorReader.readLine();
            iorReader.close();
        }
        catch( Exception e )
        {
            e.printStackTrace();
            return( null );
        }
        if( ior == null )
        {
            return( null );
        }

        objRefServer = orb.string_to_object(ior);
        return( JaeriFEL.PCRefServerHelper.narrow( objRefServer ) );
    }

    /* Connect FELDevice */
    static JaeriFEL.FELDevice[]
        getFELDevice( org.omg.CORBA.ORB orb, JaeriFEL.PCRefServer refDevServer )
    {
        String[] strGroups = refDevServer.retGetGroupList();
        ArrayList devList = new ArrayList();
        for( int i = 0; i < strGroups.length ; i++ )
        {
            System.out.println( "Jaeri FEL[Group Name:" + ( i + 1 ) + "] -> " + strGroups[i] );
            String[] strDevices = refDevServer.retGetGroupIOR( strGroups[i] );
            for( int j = 0; j < strDevices.length ; j++ )
            {
                org.omg.CORBA.Object remote = null;
            }
        }
    }
}

```

```

remote = orb.string_to_object( strDevices[j] );
JaeriFEL.FELDevice dev = JaeriFEL.FELDeviceHelper.narrow( remote );
devList.add( dev );
}
}

JaeriFEL.FELDevice[] retVal = new JaeriFEL.FELDevice[devList.size()];
devList.toArray( retVal );
return( retVal );
}

< */
static int run( org.omg.CORBA.ORB orb, String[] args ) throws org.omg.CORBA.UserException
{
boolean bLoop = true;
int keyin = 0;

JaeriFEL.PCRefServer refDevServer = getPCRefServer( orb, args );
if( refDevServer == null )
{
    return( -1 );
}

JaeriFEL.FELDevice[] devs = getFELDevice( orb, refDevServer );

try
{
    while( bLoop )
    {
        System.out.println( "" );
        System.out.println( "-----" );
        System.out.println( "FEL Device -----> 1" );
        System.out.println( "FEL Restoraton -----> 2" );
        System.out.println( "MCU Debug -----> 3" );
        System.out.println( "-----" );
        System.out.println( "          Exit -----> 0" );
        System.out.println( "-----" );
        System.out.print( "Select No. -----> ?" );
        System.out.flush();

        try
        {
            BufferedReader r = new BufferedReader( new InputStreamReader( System.in ), 1 );
            String s = r.readLine();           // 文字列の入力
            keyin = Integer.parseInt(s);     // 整数に変換
        }
        catch( IOException ex )
        {
            ex.printStackTrace();
        }

/* FELDevice */
if( keyin == 1 )
{
    Dev_Ope( devs );
}
/* FELRestoration */
}
}

```

```

else if( keyin == 2 )
{
    Dev_Rst( refDevServer );
}
/* MCU Debug */
else if( keyin == 3 )
{
    MCU_Debug( orb, refDevServer );
}
/* Exit */
else if( keyin == 0 )
{
    System.out.print( "<< Exit !! >>" );
    bLoop = false;
    continue;
}
/* Other */
else
{
    continue;
}
}
}
catch( Exception e )
{
    e.printStackTrace();
}
return 0;
}

static int Dev_Ope( JaeriFEL.FELDevice[] devs ) throws org.omg.CORBA.UserException
{
    boolean bLoop = true;
    int iKeyin = 0;
    int iNo = 0;
    int iNum;
    int iBit = 0;
    double dAmp = 0.0;

    try
    {
        while( bLoop )
        {
            System.out.println( "" );
            System.out.println( "FEL Device List" );
            for( iNum = 0 ; iNum < devs.length ; iNum++ )
            {
                String strName = devs[iNum].retName();
                String[] strGroup = devs[iNum].retGroup();
                System.out.println("Device Name["++(iNum+1)+"]:[ "+strName+":"+strGroup[0]+"]");
            }
            System.out.println( "-----" );
            System.out.println( "      Exit --> 0" );
            System.out.println( "-----" );
            System.out.print( "Select No. --> ?" );
            System.out.flush();
    }
}

```

```

/*********/
/* キー入力処理 */
/*********/
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();      // 文字列の入力
    iKeyin = Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}

if( iKeyin == 0 )
{
    bLoop = false;
    continue;
}
else if( ( 1 > iKeyin ) && ( iKeyin > ( devs.length + 1 ) ) )
{
    continue;
}

System.out.println("Put Amp -----> 1");
System.out.println("Put Pol(+) -----> 2");
System.out.println("Put Pol(-) -----> 3");
System.out.println("Get Status -----> 4");
System.out.println("Get Status(Loop) ---> 5");
System.out.println("Put Pol(+) (Loop) ---> 6");
System.out.println("Put Amp(Loop) -----> 7");
System.out.println("Put Restore(Loop) ---> 8");
System.out.println("Get Restore(Loop)---> 9");
System.out.println("-----");
System.out.println("        Exit ---> 0");
System.out.println("-----");
System.out.print("Select No. ---> ?");
System.out.flush();

try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();      // 文字列の入力
    iNo = Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}

/* Put Amp */
if( iNo == 1 )
{
    System.out.println("Device Name ---> " + devs[iKeyin - 1].retName());
    System.out.print("Input Amp(double) ---> ?");
    System.out.flush();
}

```

```

/* Input Amp */
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 10);
    String s = r.readLine();           // 文字列の入力
    dAmp = Double.parseDouble(s);     // 実数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.println("Put Amp --> " + devs[iKeyin - 1].retPutAmp( dAmp ) );
}

/* Put Pol(+) */
else if( iNo == 2 )
{
    System.out.println("Device Name ---->" + devs[iKeyin - 1].retName() );
    System.out.println("Return Pol(+) -->" + devs[iKeyin - 1].retPutPolPlus() );
}

/* Put Pol(-) */
else if( iNo == 3 )
{
    System.out.println("Device Name ---->" + devs[iKeyin - 1].retName() );
    System.out.println("Return Pol(-) -->" + devs[iKeyin - 1].retPutPolMinus() );
}

/* Get Status */
else if( iNo == 4 )
{
    if( ( 1 <= iKeyin ) && ( iKeyin <= devs.length + 1 ) )
    {
        JaeriFEL.STRC_FEL_STAT      tStat = devs[iKeyin - 1].retGetStatus();
        JaeriFEL.STRC_FEL_PUT_STAT  tPutStat = devs[iKeyin - 1].retGetPutStatus();
        JaeriFEL.R0DoubleDesc       tCur = devs[iKeyin - 1].tCurDesc();
        JaeriFEL.RWDoubleDesc       tSet = devs[iKeyin - 1].tSetDesc();
        System.out.println("Group Name-->" + (devs[iKeyin - 1].retGroup())[0] );
        System.out.println("Device Name-->" + devs[iKeyin - 1].retName() );
        System.out.println("Monitor Value(double) : " + tStat.dCur );
        System.out.println("Monitor Value(long)   : 0x" + Long.toHexString( tStat.lCur ) );
        System.out.println("Min(double)          : " + tCur.dMin );
        System.out.println("Max(double)          : " + tCur.dMax );
        System.out.println("Unit                 : " + tCur.tUnit );
        System.out.println("Format               : " + tCur.tFormat );
        System.out.println("Status(long)         : 0x" + Long.toHexString( tStat.lStat ) );
        System.out.println("Setting Value(double) : " + tPutStat.dSet );
        System.out.println("Setting Value(long)  : 0x" + Long.toHexString(tPutStat.lSet));
        System.out.print("Hit Any Key-->" );
        try
        {
            BufferedReader rr = new BufferedReader(new InputStreamReader(System.in), 3);
            String ss = rr.readLine();           // 文字列の入力
        }
        catch( IOException ex )
        {
            ex.printStackTrace();
        }
        System.out.flush();
    }
}

```

```

}
/* Get Status(Loop) */
else if( iNo == 5 )
{
    if( ( 1 <= iKeyin ) && ( iKeyin <= devs.length + 1 ) )
    {
        while( true )
        {
            JaeriFEL.STRC_FEL_STAT      tStat = devs[iKeyin - 1].retGetStatus();
            JaeriFEL.STRC_FEL_PUT_STAT  tPutStat = devs[iKeyin - 1].retGetPutStatus();
            JaeriFEL.RODoubleDesc       tCur = devs[iKeyin - 1].tCurDesc();
            JaeriFEL.RWDoubleDesc       tSet = devs[iKeyin - 1].tSetDesc();
            System.out.println( "Group Name-->" + (devs[iKeyin - 1].retGroup())[0] );
            System.out.println( "Device Name-->" + devs[iKeyin - 1].retName() );
            System.out.println( "Monitor Value(double) : " + tStat.dCur );
            System.out.println( "Monitor Value(long)   : 0x" + Long.toHexString( tStat.lCur ) );
            System.out.println( "Min(double)           : " + tCur.dMin );
            System.out.println( "Max(double)           : " + tCur.dMax );
            System.out.println( "Unit                  : " + tCur.tUnit );
            System.out.println( "Format                : " + tCur.tFormat );
            System.out.println( "Status(long)          : 0x" + Long.toHexString( tStat.lStat ) );
            System.out.println( "Setting Value(double) : " + tPutStat.dSet );
            System.out.println( "Setting Value(long)   : 0x" + Long.toHexString( tPutStat.lSet ) );
            System.out.println( "Sleep->1sec" );
            try
            {
                Thread.sleep( 100 );
            }
            catch( Exception ex )
            {
                ex.printStackTrace();
            }
            System.out.println( "Wake up!!" );
        }
    }
}
/* Put Pol(+) (Loop) */
else if( iNo == 6 )
{
    if( ( 1 <= iKeyin ) && ( iKeyin <= devs.length + 1 ) )
    {
        while( true )
        {
            System.out.println( "Device Name -->" + devs[iKeyin - 1].retName() );
            System.out.println( "Return Pol(+) -->" + devs[iKeyin - 1].retPutPolPlus() );
            System.out.println( "Sleep->1sec" );
            try
            {
                Thread.sleep( 100 );
            }
            catch( Exception ex )
            {
                ex.printStackTrace();
            }
            System.out.println( "Wake up!!" );
        }
    }
}

```

```

}
else if( iNo == 7 )
{
  if( ( 1 <= iKeyin ) && ( iKeyin <= devs.length + 1 ) )
  {
    while( true )
    {
      System.out.println( "Device Name ---->" + devs[iKeyin - 1].retName() );
      System.out.println( "Return Amp ---->" + devs[iKeyin - 1].retPutAmp( 0.11 ) );
      System.out.println( "Sleep->1sec" );
      try
      {
        Thread.sleep( 100 );
      }
      catch( Exception ex )
      {
        ex.printStackTrace();
      }
      System.out.println( "Wake up!!" );
    }
  }
}
else if( iNo == 8 )
{
  if( ( 1 <= iKeyin ) && ( iKeyin <= devs.length + 1 ) )
  {
    while( true )
    {
      System.out.println("Device Name ----->"+devs[iKeyin-1].retName());
      System.out.println( "Sleep->1sec" );
      try
      {
        Thread.sleep( 100 );
      }
      catch( Exception ex )
      {
        ex.printStackTrace();
      }
      System.out.println( "Wake up!!" );
    }
  }
}
else if( iNo == 9 )
{
  if( ( 1 <= iKeyin ) && ( iKeyin <= devs.length + 1 ) )
  {
    while( true )
    {
      System.out.println( "Device Name ----->" + devs[iKeyin - 1].retName() );
      System.out.println( "Sleep->1sec" );
      try
      {
        Thread.sleep( 100 );
      }
      catch( Exception ex )
      {
        ex.printStackTrace();
      }
    }
  }
}

```

```

        }
        System.out.println( "Wake up!!" );
    }
}
else if( iKeyin == 0 )
{
    bLoop = false;
    continue;
}
}
catch( Exception e )
{
    e.printStackTrace();
}
return( 0 );
}

static int Dev_Rst( JaeriFEL.PCRefServer refDevServer ) throws org.omg.CORBA.UserException
{
    boolean bLoop = true;
    int iKeyin = 0;
    int iNo = 0;
    int iNum;
    int iBit = 0;
    short sRet;
    JaeriFEL.STRG_FEL_RESTORE_INFO[] tRest;

    try
    {
        while( bLoop )
        {
            System.out.println( "" );
            System.out.println( "FEL Group List" );
            String[] strGroups = refDevServer.retGetGroupList();
            for( int i = 0; i < strGroups.length ; i++ )
            {
                System.out.println( "Jaeri FEL[Group Name] --> " + strGroups[i] );
            }
            System.out.println( "-----" );
            System.out.println( "      Exit --> 0" );
            System.out.println( "-----" );
            System.out.print( "Select No. --> ?" );
            System.out.flush();

            /*****
            /* キー入力処理 */
            *****/
            try
            {
                BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
                String s = r.readLine(); // 文字列の入力
                iKeyin = Integer.parseInt(s); // 整数に変換
            }
            catch( IOException ex )
            {

```

```

    ex.printStackTrace();
}
if( iKeyin == 0 )
{
    bLoop = false;
    continue;
}
else if( ( 1 > iKeyin ) && ( iKeyin > ( strGroups.length + 1 ) ) )
{
    continue;
}

System.out.println( "Put Restoration -----> 1" );
System.out.println( "Get Restoration -----> 2" );
System.out.println( "Request Current Status to DB --> 3" );
System.out.println( "-----" );
System.out.println( "                         Exit --> 0" );
System.out.println( "-----" );
System.out.print( "Select No. --> ?" );
System.out.flush();

try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine(); // 文字列の入力
    iNo = Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}

/* Put Restoration */
if( iNo == 1 )
{
}
/* Get Restoration */
else if( iNo == 2 )
{
    tRest = refDevServer.refGetRestoration( strGroups[iKeyin-1] );
    System.out.println( "Restoration Num=" + tRest.length );
}
/* Request Current Status to DB */
else if( iNo == 3 )
{
    System.out.println( "Request Current Status to DB!!" );
    sRet = refDevServer.refPutDBStore();
    System.out.println( "Status=" + sRet );
}
}
}
catch( Exception e )
{
    e.printStackTrace();
}
return( 0 );
}

```

```

static int MCU_Debug( org.omg.CORBA.ORB orb, JaeriFEL.PCRefServer refDevServer )
    throws org.omg.CORBA.UserException
{
    boolean bLoop = true;
    int iKeyin = 0;
    int iMCUNo = 0;
    int iCrate1 = 0;
    int iCrate2 = 0;
    int iCrate3 = 0;
    int iNo = 0;
    int iNum;
    int iBit = 0;

    JaeriFEL.STRG_CAMAC_INFO[] tCamac = new JaeriFEL.STRG_CAMAC_INFO[1];
    tCamac[0] = new JaeriFEL.STRG_CAMAC_INFO();
    JaeriFEL.STRG_CAMAC_INFO[] tBus;
    String strMCU = refDevServer.retGetMCUIOR();
    org.omg.CORBA.Object remote = null;
    remote = orb.string_to_object( strMCU );
    JaeriFEL.MCU dev = JaeriFEL.MCUHelper.narrow( remote );

    try
    {
        while( bLoop )
        {
            System.out.println( "MCU Reset -----> 1" );
            System.out.println( "MCU Table Clear -----> 2" );
            System.out.println( "MCU CAMAC Init -----> 3" );
            System.out.println( "CAMAC Data Get -----> 4" );
            System.out.println( "CAMAC Data Put -----> 5" );
            System.out.println( "CAMAC Put LAM Init(LI) -----> 6" );
            System.out.println( "CAMAC Put Reset(RS) -----> 7" );
            System.out.println( "CAMAC Put Enable Interrupt(EI) -----> 8" );
            System.out.println( "CAMAC Put Inhibit(I) -----> 9" );
            System.out.println( "CAMAC Put Initialize(Z) -----> 10" );
            System.out.println( "CAMAC Put Clear(C) -----> 11" );
            System.out.println( "CAMAC Get Bus Status -----> 12" );
            System.out.println( "-----" );
            System.out.println( "----- Exit --> 0" );
            System.out.println( "-----" );
            System.out.print( "Select No. --> ?" );
            System.out.flush();

            *****/
            /* キー入力処理 */
            *****/
            try
            {
                BufferedReader r = new BufferedReader( new InputStreamReader( System.in ), 3 );
                String s = r.readLine(); // 文字列の入力
                iKeyin = Integer.parseInt( s ); // 整数に変換
            }
            catch( IOException ex )
            {
                ex.printStackTrace();
            }
        }
    }
}

```

```

if( iKeyin == 0 )
{
    bLoop = false;
    continue;
}

/* MCU Reset */
else if( iKeyin == 1 )
{
    System.out.print( "MCU No. -->" );
    System.out.flush();
    try
    {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
        String s = r.readLine();           // 文字列の入力
        iMCUNo = Integer.parseInt(s);     // 整数に変換
    }
    catch( IOException ex )
    {
        ex.printStackTrace();
    }
    dev.MCU_Reset( ( short )iMCUNo );
    System.out.println( "MCU Reset !!");
}

/* MCU Table Clear */
else if( iKeyin == 2 )
{
    System.out.print( "MCU No. -->" );
    System.out.flush();
    try
    {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
        String s = r.readLine();           // 文字列の入力
        iMCUNo = Integer.parseInt(s);     // 整数に変換
    }
    catch( IOException ex )
    {
        ex.printStackTrace();
    }
    System.out.println( "MCU Reset -->" + dev.MCU_TbIClr( ( short )iMCUNo ) );
}

/* CAMAC Init */
else if( iKeyin == 3 )
{
    System.out.print( "MCU No. -->" );
    System.out.flush();
    try
    {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
        String s = r.readLine();           // 文字列の入力
        iMCUNo = Integer.parseInt(s);     // 整数に変換
    }
    catch( IOException ex )
    {
        ex.printStackTrace();
    }
}

```

```

}

System.out.print( "Crate1 No. -->?" );
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();          // 文字列の入力
    iCrate1 = Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}

System.out.print( "Crate2 No. -->?" );
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();          // 文字列の入力
    iCrate2 = Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}

System.out.print( "Crate3 No. -->?" );
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();          // 文字列の入力
    iCrate3 = Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}

System.out.println( "MCU Reset -->" +
    dev.CAMAC_Init( ( short )iMCUNo, ( short )iCrate1, ( short )iCrate2, ( short )iCrate3 ) );
}

/* CAMAC Data Get */
else if( iKeyin == 4 )
{
    System.out.print( "MCU No. -->?" );
    System.out.flush();
    try
    {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
        String s = r.readLine();          // 文字列の入力
        iMCUNo = Integer.parseInt(s); // 整数に変換
    }
    catch( IOException ex )
    {
        ex.printStackTrace();
    }

    System.out.print( "Crate No. -->?" );
}

```

```

System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine(); // 文字列の入力
    tCamac[0].sCrat = (short)Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.print("Station No. -->?");
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine(); // 文字列の入力
    tCamac[0].sStation = (short)Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.print("Sub Address No. -->?");
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine(); // 文字列の入力
    tCamac[0].sSubAddr = (short)Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.print("Function No. -->?");
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine(); // 文字列の入力
    tCamac[0].sFunc = (short)Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
tBus = dev.CAMAC_Get((short)iMCUNo, tCamac);
System.out.println("CAMAC Data Get");
System.out.println("Get Data=0x" + Long.toHexString(tBus[0].lData));
}

/* CAMAC Data Put */
else if( iKeyin == 5 )
{
    System.out.print("MCU No. -->?");
    System.out.flush();
}

```

```

try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();           // 文字列の入力
    iMCUNo = Integer.parseInt(s);     // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.print("Crate No. -->? ");
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();           // 文字列の入力
    tCamac[0].sCrate = (short)Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.print("Station No. -->? ");
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();           // 文字列の入力
    tCamac[0].sStation = (short)Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.print("Sub Address No. -->? ");
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();           // 文字列の入力
    tCamac[0].sSubAddr = (short)Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}

System.out.print("Function No. -->? ");
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();           // 文字列の入力
    tCamac[0].sFunc = (short)Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
}

```

```

    ex.printStackTrace();
}
System.out.print( "Put Data(DEC) -->?" );
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();           // 文字列の入力
    tCamac[0].lData = Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.println( "CAMAC Data Put -->" + dev.CAMAC_Put( ( short )iMCUNo, tCamac ) );
}

/* CAMAC LAM Init */
else if( iKeyin == 6 )
{
    System.out.print( "MCU No. -->?" );
    System.out.flush();
    try
    {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
        String s = r.readLine();           // 文字列の入力
        iMCUNo = Integer.parseInt(s);     // 整数に変換
    }
    catch( IOException ex )
    {
        ex.printStackTrace();
    }
    System.out.print( "Crate No. -->?" );
    System.out.flush();
    try
    {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
        String s = r.readLine();           // 文字列の入力
        tCamac[0].sCrate = ( short )Integer.parseInt(s); // 整数に変換
    }
    catch( IOException ex )
    {
        ex.printStackTrace();
    }
    System.out.println( "CAMAC LAM Init -->" + dev.CAMAC_Put_LI( ( short )iMCUNo, tCamac ) );
}

/* CAMAC Reset */
else if( iKeyin == 7 )
{
    System.out.print( "MCU No. -->?" );
    System.out.flush();
    try
    {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
        String s = r.readLine();           // 文字列の入力
        iMCUNo = Integer.parseInt(s);     // 整数に変換
    }
}

```

```

}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.print( "Crate No. -->?" );
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();           // 文字列の入力
    tCamac[0].sCrate = ( short )Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.println( "CAMAC Reset -->" + dev.CAMAC_Put_RS( ( short )iMCUNo, tCamac ) );
}

/* CAMAC Enable Interrupt */
else if( iKeyin == 8 )
{
    System.out.print( "MCU No. -->?" );
    System.out.flush();
    try
    {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
        String s = r.readLine();           // 文字列の入力
        iMCUNo = Integer.parseInt(s);     // 整数に変換
    }
    catch( IOException ex )
    {
        ex.printStackTrace();
    }
    System.out.print( "Crate No. -->?" );
    System.out.flush();
    try
    {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
        String s = r.readLine();           // 文字列の入力
        tCamac[0].sCrate = ( short )Integer.parseInt(s); // 整数に変換
    }
    catch( IOException ex )
    {
        ex.printStackTrace();
    }
    System.out.println("CAMAC Enable Interrupt -->"+
                       dev.CAMAC_Put_EI((short)iMCUNo, tCamac));
}

/* CAMAC Inhibit */
else if( iKeyin == 9 )
{
    System.out.print( "MCU No. -->?" );
    System.out.flush();
    try

```

```

{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();           // 文字列の入力
    iMCUNo = Integer.parseInt(s);     // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.print("Crate No. -->?");
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();           // 文字列の入力
    tCamac[0].sCrate = (short)Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.println("CAMAC Inhibit -->" + dev.CAMAC_Put_I((short)iMCUNo, tCamac));
}

/* CAMAC Initialize */
else if( iKeyin == 10 )
{
    System.out.print("MCU No. -->?");
    System.out.flush();
    try
    {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
        String s = r.readLine();           // 文字列の入力
        iMCUNo = Integer.parseInt(s);     // 整数に変換
    }
    catch( IOException ex )
    {
        ex.printStackTrace();
    }
    System.out.print("Crate No. -->?");
    System.out.flush();
    try
    {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
        String s = r.readLine();           // 文字列の入力
        tCamac[0].sCrate = (short)Integer.parseInt(s); // 整数に変換
    }
    catch( IOException ex )
    {
        ex.printStackTrace();
    }
    System.out.println("CAMAC Initialize -->" + dev.CAMAC_Put_Z((short)iMCUNo, tCamac));
}

/* CAMAC Clear */
else if( iKeyin == 11 )
{

```

```

System.out.print( "MCU No. -->?" );
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();           // 文字列の入力
    iMCUNo = Integer.parseInt(s);     // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.print( "Crate No. -->?" );
System.out.flush();
try
{
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
    String s = r.readLine();           // 文字列の入力
    tCamac[0].sCrate = (short)Integer.parseInt(s); // 整数に変換
}
catch( IOException ex )
{
    ex.printStackTrace();
}
System.out.println( "CAMAC Clear -->" + dev.CAMAC_Put_C( (short)iMCUNo, tCamac ) );
}

/* CAMAC Bus Status */
else if( iKeyin == 12 )
{
    System.out.print( "MCU No. -->?" );
    System.out.flush();
    try
    {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
        String s = r.readLine();           // 文字列の入力
        iMCUNo = Integer.parseInt(s);     // 整数に変換
    }
    catch( IOException ex )
    {
        ex.printStackTrace();
    }
    System.out.print( "Crate No. -->?" );
    System.out.flush();
    try
    {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 3);
        String s = r.readLine();           // 文字列の入力
        tCamac[0].sCrate = (short)Integer.parseInt(s); // 整数に変換
    }
    catch( IOException ex )
    {
        ex.printStackTrace();
    }
    tBus = dev.CAMAC_GetBus( (short)iMCUNo, tCamac );
    System.out.println( "Get Bus Status=0x" + Long.toHexString(tBus[0].lData) );
}
}

```

```
        }
    }
    catch( Exception e )
    {
        e.printStackTrace();
    }
    return( 0 );
}

/*********/
/* Main */
/*********/
public static void main( String args[] )
{
    java.util.Properties props = System.getProperties();
    int status = 0;
    org.omg.CORBA.ORB orb = null;

    try
    {
        orb = org.omg.CORBA.ORB.init(args, null);
        status = run(orb, args);
    }
    catch(Exception ex)
    {
        ex.printStackTrace();
        status = 1;
    }
    System.exit(status);
}
}
```

国際単位系(SI)と換算表

表1 SI基本単位および補助単位

量	名称	記号
長さ	メートル	m
質量	キログラム	kg
時間	秒	s
電流	アンペア	A
熱力学温度	ケルビン	K
物質量	モル	mol
光度	カンデラ	cd
平面角	ラジアン	rad
立体角	ステラジアン	sr

表3 固有の名称をもつSI組立単位

量	名称	記号	他のSI単位による表現
周波数	ヘルツ	Hz	s^{-1}
力	ニュートン	N	$m \cdot kg/s^2$
圧力、応力	パスカル	Pa	N/m^2
エネルギー、仕事、熱量	ジュール	J	N·m
功率、放射束	ワット	W	J/s
電気量、電荷	クーロン	C	$A \cdot s$
電位、電圧、起電力	ボルト	V	W/A
静電容量	ファラード	F	C/V
電気抵抗	オーム	Ω	V/A
コンダクタンス	ジーメンス	S	A/V
磁束	ウェーバ	Wb	$V \cdot s$
磁束密度	テスラ	T	Wb/m^2
インダクタンス	ヘンリイ	H	Wb/A
セルシウス温度	セルシウス度	$^{\circ}C$	
光束度	ルーメン	lm	$cd \cdot sr$
照度	ルクス	lx	lm/m^2
放射能	ベクレル	Bq	s^{-1}
吸収線量	グレイ	Gy	J/kg
線量当量	シーベルト	Sv	J/kg

表2 SIと併用される単位

名称	記号
分、時、日	min, h, d
度、分、秒	°, ', "
リットル	l, L
トントン	t
電子ボルト	eV
原子質量単位	u

$$1 \text{ eV} = 1.60218 \times 10^{-19} \text{ J}$$

$$1 \text{ u} = 1.66054 \times 10^{-27} \text{ kg}$$

表5 SI接頭語

倍数	接頭語	記号
10^{18}	エクサ	E
10^{15}	ペタ	P
10^{12}	テラ	T
10^9	ギガ	G
10^6	メガ	M
10^3	キロ	k
10^2	ヘクト	h
10^1	デカ	da
10^{-1}	デシ	d
10^{-2}	センチ	c
10^{-3}	ミリ	m
10^{-6}	マイクロ	μ
10^{-9}	ナノ	n
10^{-12}	ピコ	p
10^{-15}	フェムト	f
10^{-18}	アト	a

(注)

- 表1～5は「国際単位系」第5版、国際度量衡局1985年刊行による。ただし、1eVおよび1uの値はCODATAの1986年推奨値によった。
- 表4には海里、ノット、アール、ヘクタールも含まれているが日常の単位なのでここでは省略した。
- barは、JISでは流体の圧力を表わす場合に限り表2のカテゴリーに分類されている。
- EC閣僚理事会指令ではbar、barnおよび「血圧の単位」mmHgを表2のカテゴリーに入れている。

換算表

力	N($=10^5$ dyn)	kgf	lbf
	1	0.101972	0.224809
	9.80665	1	2.20462
	4.44822	0.453592	1

$$\text{粘度 } 1 \text{ Pa}\cdot\text{s}(\text{N}\cdot\text{s}/\text{m}^2) = 10 \text{ P(ポアズ)} (\text{g}/(\text{cm}\cdot\text{s}))$$

$$\text{動粘度 } 1 \text{ m}^2/\text{s} = 10^4 \text{ St(ストークス)} (\text{cm}^2/\text{s})$$

圧力	MPa($=10$ bar)	kgf/cm ²	atm	mmHg(Torr)	lbf/in ² (psi)
力	1	10.1972	9.86923	7.50062×10^3	145.038
	0.0980665	1	0.967841	735.559	14.2233
	0.101325	1.03323	1	760	14.6959
	1.33322×10^{-4}	1.35951×10^{-3}	1.31579×10^{-3}	1	1.93368×10^{-2}
	6.89476×10^{-3}	7.03070×10^{-2}	6.80460×10^{-2}	51.7149	1

エネルギー・仕事・熱量	J($=10^7$ erg)	kgf·m		kW·h		cal(計量法)	Btu	ft · lbf	eV	1 cal = 4.18605 J(計量法)	
		1	0.101972	2.77778×10^{-7}	0.238889					$= 4.184 \text{ J (熱化学)}$	
	9.80665		1	2.72407×10^{-6}	2.34270	9.29487×10^{-3}	7.23301	6.12082×10^{19}	$= 4.1855 \text{ J (15}^{\circ}\text{C)}$		
	3.6×10^6	3.67098×10^5	1	8.59999×10^5	3412.13	2.65522×10^6	2.24694×10^{25}	$= 4.1868 \text{ J (国際蒸気表)}$			
	4.18605	0.426858	1.16279×10^{-6}	1	3.96759×10^{-3}	3.08747	2.61272×10^{19}	$= 75 \text{ kgf} \cdot \text{m/s}$	$= 1 \text{ PS (仏馬力)}$		
	1055.06	107.586	2.93072×10^{-4}	252.042	1	778.172	6.58515×10^{21}	$= 735.499 \text{ W}$			
	1.35582	0.138255	3.76616×10^{-7}	0.323890	1.28506×10^{-3}	1	8.46233×10^{18}				
	1.60218×10^{-19}	1.63377×10^{-20}	4.45050×10^{-26}	3.82743×10^{-20}	1.51857×10^{-22}	1.18171×10^{-19}	1				

放射能	Bq	Ci	吸収線量	Gy	rad
	1	2.70270×10^{-11}		1	100
	3.7×10^{16}	1		0.01	1

照 射 線 量	C/kg	R	照 射 線 量	1	3876
	1	2.58×10^{-4}		1	1

線 量 當 量	Sv	rem	線 量 當 量	1	100
	1	0.01		0.01	1

(86年12月26日現在)

原研ER-LFE-Lのための制御・データベースシステムの開発



古紙配合率100%
白色度70%の再生紙を使用しています