PF ビームライン・インターロックシステムとその集中管理システムの開発

小菅 隆
高エネルギー加速器研究機構
技術部測定器第五課
〒305-0801 茨城県つくば市大穂 1-1

概要

ビームライン・インターロックシステムは放射光研究施設のビームライン毎に設置され、放射線安全、ビームライン真空の保持、ビームライン構成要素の保護を目的とし高い信頼性の元に動作してきたが、様々な技術開発等を行うことにより信頼性を保持したままでの低コスト化及び省力化に成功した。

また、集中管理システムはこれらビームライン・インターロックシステムを統合的に監視・制御するシステムであり、これまでビームラインにおけるトラブルへの対応を迅速に行うための技術の導入を行ってきたが、新たに分散化システムを導入することで、高い保守性を有するシステムを構築することが出来た。

1. はじめに

放射光研究施設（以下 PF）には 2001 年 9 月現在、2.5GeV リングに 21 本、6.5GeV-AR リングに 3 本のビームラインが設置され様々な放射光利用実験が行われている。それぞれのビームラインは 1〜4 本のブランチビームライン（branch beamline）に分岐され、リングからのシンクロトロン放射光は各ブランチビームラインの実験ステーションまで導かれる。ビームラインは真空に保たれており、真空封止のためのゲートバルブや真空度を監視する真空ゲージなどが取り付けられている。また、X 線を利用するビームラインなどにはハッチと呼ばれる放射線防護用の区画が設置されており、実験装置等が収納されている。ビームライン・インターロックシステム（以下 BLIS）[1-3]は PF の全てのビームラインに設置され、これらゲートバルブ、真空ゲージ、実験ハッチなどに加え放射光ビームの出射・停止を司るビームシャターやセンサーなどを統合的に監視制御しており、高い信頼性を保持している。

PF ではビームライン性能の向上を目指した改造が頻繁に行われる。そのため BLIS は高い信頼性を維持すると共に非常に柔軟なシステムでなければならない。また、ビームラインの改造は放射光リング運転停止期間中の限られた期間に行われるため、BLIS 新設及び改造に際しては作業の省力化が求められる。これらと同時に低コスト化も大きな課題である。

これまで、PLC（Programmable Logic Controller、シーケンサ）を積極的に利用することで BLIS の柔軟性を確保することは可能となっていた。また、PLC のプログラムをビームライン毎に書き換える部分とビームラインに共通の部分とに分割することにより、プログラミングに対する省力化もある程度行うことが出来た。しかし、システムの構成上、これ以上の省力化及び低価格化を行うためには限界があった。

そして、集中管理システム[4-6]はこれら BLIS を統合的に監視・制御するシステムである。各 BLIS からの運転状態を示す信号は光ファイバーを用いて集中管理システムに入力される。また、集中管理システムからのビームライン使用の許可信号等は同じく光ファイバーを用いて各 BLIS に出力される。これまで集中管理システムは安定して動作してきたが、2.5GeV リングの集中管理システムにおいて老朽化等の問題が発生した。また、同システムにおいては運転中に保守作業を行えない等の作業上の問題点も指摘された。

以上、幾つかの問題点を克服すべく検討を重ねた結果、BLIS についてはシステム構成の大幅な見直し及び省配線システムの導入、プログラミングの省力化を図るためのプログラム自動生成システムの開発[7]を行うこととした。また、集中管理システムについては加速器制御用に開発されたシステムである COACK（Component Oriented Accelerator Control Kernel）[8-10]が非常に柔軟なシステムであり、集中管理システムにも有効であることに着目、COACK をシステムの中心に導入することにより、システムの分散化を行うこととした。

2. BLIS

BLIS はビームライン毎に設置されており、接続されたビームシャッター、ゲートバルブ、真空ゲージ、ハッチなどのビームラインコンポーネントを統合的に管理している（図 1参照）。それぞれの BLIS は独立して動作しており、構成は以下のとおりである。

- メインラック：機器の制御を行うための PLC を内蔵するとともに保守などを行うためのコントロールパネル（メインコントローラ）も持つ。
- ステーションコントローラ：実験者が直接操作を行い、放射光ビームの出射及び停止、ゲートバルブの開閉等を行うためのコントロールパネル。
- ハッチ退出制御盤：ハッチ内に取り付けられ、ハッチを閉とする際の定められた退出手順を行うために使用される。
- ハッチ状態表示盤：ハッチ外側に取り付けられ、ハッチの開閉状態及びビームシャッターの

開閉ステータスを表示する。

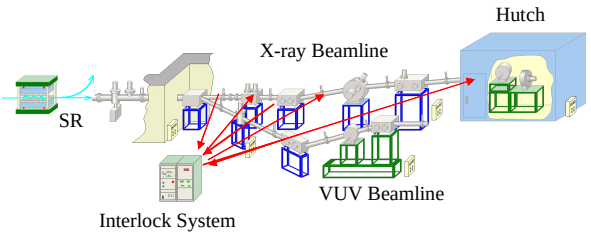

図1: ビームラインと BLIS

また、BLIS において接続された全てのビームラインコンポーネントの制御はメインラック内に内蔵された OMRON 製の PLC によって行われる。この事により BLIS は様々なビームライン構成に柔軟に対応する事が出来る。これまで、BLIS は導入当初から定期的な動作試験及び保守作業等により、特に大きなトラブルも無く安定して動作してきた。

2.1 実験者の放射線被爆からの防護

BLIS が動作する上で最も重要なロジックが放射線安全の為のロジックである。X 線を利用するビームラインにおいて実験者は実験試料の交換や調整の為に実験ハッチに立ち入らなければならない。放射光利用実験は不特定多数の実験者により昼夜を問わず行われるため、放射線安全には細心の注意をはらう必要がある。BLIS が備える放射線安全の為のロジックは次の通りである。

- ハッチ閉で初めてビームシャッターを開と出来る。
- ハッチ開となったら、ビームシャッターが閉まる。

この意味でハッチが閉まる事は大切なことである。PF では単にハッチの扉を閉じただけでシステムがハッチの閉を認識するのでは、実験者などがハッチ内に取り残された場合などを考慮すると安全であるとしていない。BLIS では以下のようなハッチ退出手順を設け、その手順が正しく遂行された場合にのみ、ハッチが閉であると認識する事にしている。

1. ハッチ内に誰も残されていない事を確認。
2. 退室制御盤の退室ボタンを押す。
3. 退室制御盤のブザーが鳴り、回転灯が回りだす。
4. ハッチのドアを閉める。
5. ハッチ状態表示盤の退室中ランプが点滅から常点灯に変わったら、ドアのキーを左に回しハッチを閉める。
6. ハッチが閉となる。

2.2 ビームライン真空の保持

ビームラインは真空に保たれているが、保守や実験試料の交換作業がビームラインの一部を真空リークさせた上で行われることがある。この時、ゲートバルブ開閉等の操作手順の誤りはビームライン真空へ大きな影響を与えるだけでなく、リングの運転自体にも影響を及ぼしかねない。また、放射光利用実験中に何らかのトラブルにより発生する突発的な真空リークなどにも対応する必要がある。BLIS では真空のインターロックとして以下のようなロジックを採用している。

- 真空度が悪い場合、ゲートバルブを開しない
- 真空が悪化したらゲートバルブを閉とする

2.3 ビームラインコンポーネントの保護

放射光は非常に強力な光であり、ゲートバルブなどに直接照射されるとゲートバルブ自体が破損する可能性がある。また、ビームシャッターなどの冷却水の停止はビームラインに深刻なトラブルを引き起こす可能性がある。BLIS はこれらビームラインコンポーネントを放射光ビームによる損傷から保護するためのロジックを備えている。

- ゲートバルブが開でければビームシャッターを開しない
- ゲートバルブが開で無くなったらビームシャッターを閉とする

3. BLIS 構築における低コスト化

BLIS は様々なビームライン構成に対し柔軟に対応出来るが、システムのコスト削減が課題となっていた。今回、BLIS の構成を根本から見直し（図 2参照）タッチパネルや省配線システムを導入することで BLIS の信頼性を確保したまま、約 40%のコストダウンに成功した。

3.1 タッチパネルの導入

これまで BLIS において各ビームラインコンポーネントの状態を示す表示パネルには、LED 表示及び彫刻あるいはシルク印刷によるグラフィックパネルを使用していた。しかし、この方法はビームラインの改造などの度にパネル部分の改造が必要な為、ビームライン改造時のコストに大きな影響を与えていた。

以前には高価であった液晶タッチパネルも最近になって低価格化が進み、BLIS メインラック内のコントロールパネルやステーションコントローラに利用してもコストを抑える事が可能となった。新しいシステムでは、安価になった液晶タッチパネルを導入することで状態表示用 LED 及びゲートバルブ開閉スイッチ等への配線を減らし BLIS 全体のコストを抑える事に成功した。また、新システムではビームライン改造時に発生するコストも大幅に削減されている。

3.2 省配線システム

これまで、BLIS において各ビームラインコンポーネントからの配線は全て BLIS のメインラックに集

中していた。一般的に BLIS の制御するビームラインコンポーネントの数は 70 程度であり、BLIS のコントロールパネルからの配線も含めると BLIS メインラック内に直接接続される配線の数は膨大なものであった。省配線システムを導入することで配線に関するコストを削減することが出来る。新システムでは Device Net を採用することで BLIS 機器間の大幅な配線数削減に成功している。

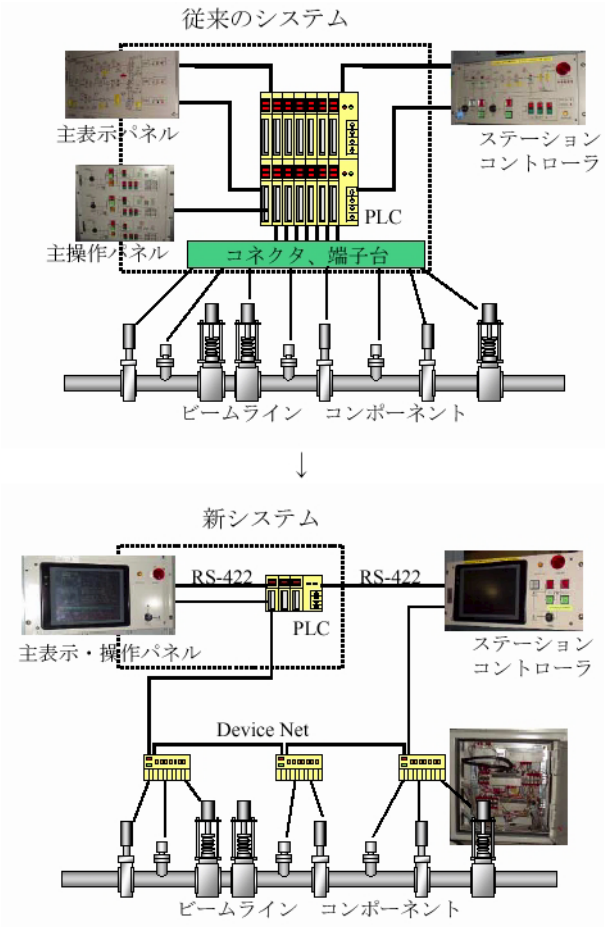

図2: 旧システムと新システム

4. BLIS プログラムの自動生成による省力化

BLIS に接続されるビームラインコンポーネントは全て PLC によって制御されるため、BLIS において PLC のプログラミングは非常に重要である。システムの完成に際しても、それぞれのロジックが正しく動作するかどうかの綿密なデバッグ作業が行われる。

これまで、出来る限りバグの少ないプログラムを作成する工夫として、プログラムをビームラインの構成毎に変更が必要な部分とそうでない部分に分割、ブロック化し、BLIS を新設する際にはこの変更の必要の無い部分をそのまま流用する方法をとっていた。また、プログラムのブロック化は開発時間においてもある程度有効であった。なお、プログラムの再利用を可能とするために BLIS のシステム構成は統一されていた。

しかし、省配線システムの導入する上で大きな問題が発生した。省配線システムを利用した BLIS ではビームライン構成に伴い、接続されるビームラインコンポーネントの I/O アドレスが大きく変化するのである。これまでのブロック化の方法でも些細な I/O アドレスの変更ミスがバグの原因の大半を占めていた事を考慮すると、バグの発生する確率は大きくなる事が予想された。

これらの問題を解決する方法として考え出されたのが「BLIS プログラムの自動生成」である。BLIS プログラム自動生成システム（以下自動生成システム）では、取り決めに従ってビームラインの構成を入力しておく事で、BLIS のプログラムを自動的に作成する事が可能である。本システムの開発により大半を占めていた I/O アドレスの変更ミスによるバグが皆無となった。また、BLIS プログラム作成に要する時間も、ブロック化によるプログラミングが「約 4 日間」であったのに対して「約 3 時間程度」と大幅な短縮に成功している。

4.1 PLC とプログラミングの概要

PLC はマイクロコンピューターをベースとしたシステムで、一般的にリレーや電磁弁などを直接駆動可能なパラレル出力ポートやアイソレートされたパラレル入力ポートを持っている。また、小規模なものから大規模なシステム向けのものまで種類は様々で、ADC 入力、モーションコントロール機能等を付加する事が可能な機種もある。現在、BLIS ではパラレル入力及びパラレル出力のみを使用している。

これらの入出力の制御は PLC 用のプログラムを作成することによって行えるが、一般的には PC 上で動作するサポートソフト等を使用して、プログラムの作成からプログラムの転送、テストまでを行う。

4.1.1 ラダー図とニーモニック

PLC のプログラミングはラダー図を作成することから始まる。ラダー図では左右に電源の+と-をそれぞれ示す線を配し、その間に接点やリレーソレノイドを示す記号を配置していく。たとえば図3に示すリレーロジックをラダー図で表現すると図4のようになる。この例では「inputA」と名づけられたアドレス 00001番の入力ポートと「inputB」と名づけられた 00002番の入力ポートの両方が ON となったとき初めて「outputC」と名づけられた00010番の出力ポートが ON となる。また、「inputA」か「inputB」のどちらかが ON となると、「outputD」と名づけられた00011番の出力ポートが ON となる。このラダー図は直接サポートソフト上で入力する事が可能である。

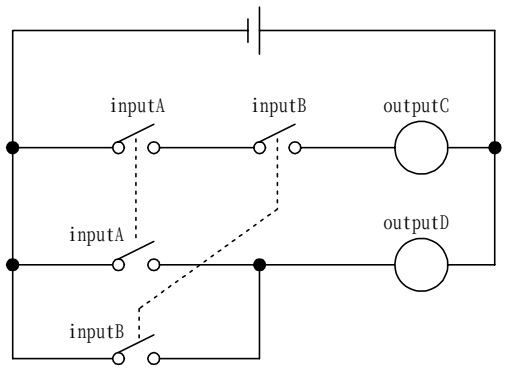

図3: リレーロジック回路例

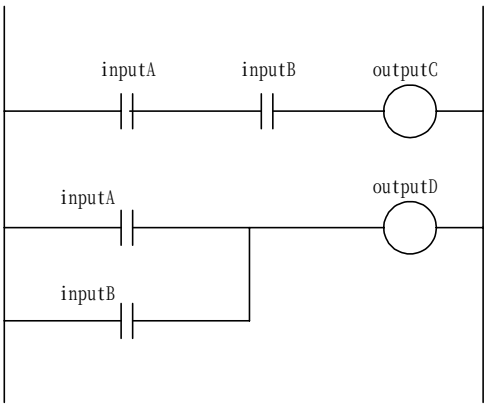

図4: ラダー図例

次に PLC のプログラムのもう一つの表現としてはニーモニックがあり、図 4のラダー図をニーモニックに直すと以下のようになる。

```
LD    00001
AND 00002
OUT 00010
LD    00001
OR    00002
OUT 00011
```

プログラミングコンソールを使い直接 PLC にプログラムを入力する際やサポートソフトでニーモニック入力を行う際には、このようなニーモニックを直接入力することとなる。

自動生成システムではこのニーモニックを直接作成する。

4.1.2 BLIS におけるプログラム

BLIS の PLC プログラミングでは、可能な限り基本的なコマンドを使用するようにしている。この事で PLC に機種変更の必要が生じても大きな問題は発生しない。

図 5は BLIS プログラムの抜粋でゲートバルブを制御する部分である。この部分ではゲートバルブが開いているか閉じているかの判断、開閉に時間がかかりすぎた場合等の以上の検知を行っている。

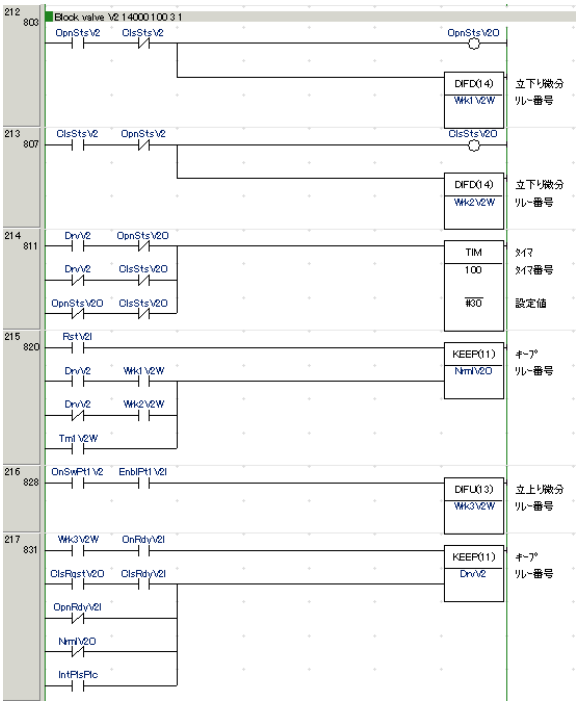

図5: BLIS プログラムの抜粋

これらのロジックは BLIS に接続されるビームラインコンポーネント毎に記述されるが、同様なビームラインコンポーネントならば、サポートソフト上でその部分をコピーし入出力ポート番号、或いは入出力名を書き換える事で毎回入力を行う必要は無い。しかし、実際の BLIS プログラムは膨大で、ニーモニック表示した際の一つの命令を一行とすると、数千行にも及ぶ。昨今、サポートソフトの性能は向上してきたが、「入出力ポート番号、或いは入出力名を書き換える」のような手作業では当然の事ながら入力ミスが発生する。更にこの入力ミスによって発生するバグはテストの際にも発見が困難である。

発見が困難なバグの削減及び手作業に掛る労力の軽減が自動生成システム開発に関する大きな動機である。

4.2 自動生成システムの構成

自動生成システムは現在 OMRON 製の PLC のみに対応しており、次のプログラム及びデータファイルからなる。

- blisc：BLIS プログラム自動生成を行うためのメインプログラム。ビームライン構成を示すソースファイルから PLC のニーモニックを生成する。また、信号名のテーブルを生成することもできる。blislib 以下のディレクトリには本プログラムのサブルーチン群が格納される。
- blisnm2cx.pl：blisc により生成された PLC のニーモニックを市販の PLC サポートソフトへ転送するためのプログラム。
- blisio2cx.pl：blisc により生成された信号名のテーブルを市販の PLC サポートソフトへ転送す

るためのプログラム。

- xxx.bl（xxx は任意）：blisc を利用して BLIS プログラム自動生成を行うための blisc ソースファイル。ビームラインの構成を記述する。
- IO アドレス外部参照ファイル：各ビームラインコンポーネントが接続される PLC の I/O アドレステーブルは blisc ソースファイルに直接記述する事も可能であるが、外部ファイルとして別途用意することも出来る。ファイルの名前は任意である。
- xxx.nm（xxx の部分はソースファイルと同じ）：blisc の起動によって生成される PLC のニーモニック。
- xxx.tbl（xxx の部分はソースファイルと同じ）：blisio2cx.pl プログラムによって市販の PLC サポートソフトへ送られる I/O アドレスのテーブル。blisc を-t オプション指定して起動する事により出力される。

blisc 及び blisnm2cx.pl、blisio2cx.pl はそれぞれ Perl で記述されており、あらかじめ Perl5 がシステムにインストールされている必要がある。blisc は Perl の動作する様々なオペレーティングシステム（以下 OS）上で使用可能である。また、blisnm2cx.pl、blisio2cx.pl は Perl(Active Perl)のインストールされた Windows（Windows98、2000 etc.）上で動作する。

4.3 自動生成システムの動作

自動生成システムでの最初の手順は BLIS の構成を示す blisc ソースファイルを作成する事である。また、必要に応じて I/O アドレス外部参照ファイルを作成する。次に blisc を使用してニーモニックを作成する。また、同様に blisc を使用して I/O アドレスのテーブルを出力する。

その後、サポートソフト（OMRON CXprogramer）を起動、blisio2cx.pl、blisnm2cx.pl を使用しサポートソフトに I/O アドレステープル及びニーモニックを転送する。最後にサポートソフトでプログラムチェックを行った後、実際に PLC にプログラムを転送する。

なお、デバッグを行い手直しが必要になった場合はこの手順を繰り返す。但し、I/O アドレスに変更がなければ I/O アドレステープルの blisc による出力及び blisio2cx.pl を使用しての転送は必要ない。

自動生成システムの使用に関する手順を図6に示す。

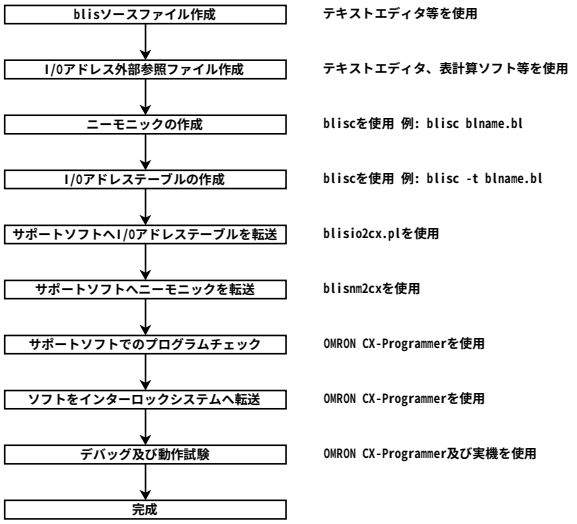

図6: BLIS プログラム自動生成の手順

4.4 ソースファイルの作成

blisc によって使用されるソースファイルはテキストファイルであり、一般的なテキストエディタを用いて作成する事が可能である。ソースファイル中に記述する項目は、宣言文やコンポーネントの作成を blisc に指示するためのコマンドであり、構成は次の通りである。

- I/O テーブルの宣言部
- 操作パネル等の設定及びロジック作成部
- ビームラインコンポーネント作成命令
- ビームライン構成の定義部
- 真空系の定義部
- 動作ロジックの作成命令

なお、ソースファイル中の"#"で始まる行及び改行だけの「空行」は blisc によって無視される。実際のソースファイルの抜粋を以下に示す。

```
# PLC に関する設定
def
  IntPlsPlc  25315
  TstDsplyPlc          10015
  Pls1sPlc   25502
  Pls02sPlc 25501
  EvtOnPlc 25313
  EvtOffPlc 25314
  DvNtErPlc            40114
enddef
:中略
# IO アドレス外部参照ファイルの読込
def bl-1iolist.txt
:中略
#  操作パネル等の設定及びロジック作成
_PTM_OPERATE = 10
make mctl 10100 10
_PTM_OPERATE =11
make bctl BctlA 10300 15 Valve
```

```
:中略
# ステーションコントローラ等の作成
_OpcAPtChannel = 32000
make opc OpcA 11500
:中略
# ビームラインコンポーネントの作成
_BbsACounter = 37
_BbsATimeUp = 1200
make shutter BbsA 12300 35 5 dual
_BbsBCounter = 44
_BbsBTimeUp = 1200
make shutter BbsB 12400 40 5 dual
:中略
# ビームライン構成の定義
def
_BlModeA 5
_BlModeB 5
_BlModeC 1
_WaterMain MskWtr_BbsA1Wtr_BbsB1Wtr
_AirMain   AirMin_BbsA2Air
:中略
# 真空系の定義
vacsection Mbs_Va1_Vb1_Vc1
vacsection Va1_Ga1_Ga2_Ga3_Va2
vacsection Va2_Ga4_Va3
vacsection Va3_Ga5_Va4
vacsection Va4_Ga6
vacsection Vb1_Gb1_Gb2_Gb3_Vb2
vacsection Vb2_Gb4_Vb3
vacsection Vb3_Gb5_Vb4
vacsection Vb4_Gb6
:中略
# 動作ロジックの作成
make mlmctl
make mlbctl A
make mlbctl B
make mlbctl C
make mlabnormaldisplay MAbnrmlDsply Main
make mlabnormaldisplay AAbnrmlDsply A
make mlabnormaldisplay BAbnrmlDsply B
make mlabnormaldisplay CAbnrmlDsply C
:中略
# 終了
make end
```

4.4.1 I/0 テーブル作成

blisc が出力するニーモニックでは直接の I/O アドレスが使用される為、初めに各ビームラインコンポーネントが接続される I/O アドレスの設定が必要となる。ソースファイル中での I/O アドレス定義は以下の通りである。

```
#書式 1
Name = Address

#書式 2
def
Name Address
Name Address
enddef
```

（*Name*:I/O 名、*Address*:I/O ポートアドレス)

また、I/O アドレスの定義に関しては外部ファイルを利用することも出来る。外部ファイルの参照を行う際は以下のようなコマンドを記述する。

include *filename*

（*filename* は任意のテキストファイル名）

このとき参照される外部ファイルは

I/O 名[TAB]I/O アドレス[改行]

のような形式であり、Excel などの表計算ソフトで作成したタブ区切りのテキストファイルがそのまま利用可能である。

4.4.2 操作パネル等の設定及びロジック作成

BLIS にはメインコントローラ、ステーションコントローラなど、ビームラインコンポーネントの状態表示及び開閉を行うための操作盤が存在する。これらのロジックの生成を行うための定義及びコマンドは以下の通りである。

```
#Operate Key 操作時に使用する定数の設定
_PTM_OPERATE = Number
(Number は設定値)

#メインコントローラロジックの作成
make mctl Address Timer
（Address:使用する内部補助リレーのアドレス、Timer:使用するタイマーカウンター番号）

#ブランチコントローラロジックの作成
make bctl Name Address Timer Option
（Name:コントローラ名、Address:使用する内部補助リレーのアドレス、Timer:使用するタイマーカウンター番号、Option:バルブ操作スイッチを幾つ付けるか等のオプション）

#ステーションコントローラのタッチパネル設定
_OpcAPtChannel = Address
（Address:使用する内部補助リレーの先頭アドレス）

#ブランチコントローラロジックの作成
make opc Name Address
（Name:コントローラ名、Address:使用する内部補助リレーの先頭アドレス）
```

4.4.3 ビームラインコンポーネントのロジック作成

ビームラインコンポーネント作成命令を以下の様に記述する事でビームラインコンポーネントに関するロジックを作成する事が出来る。

```
make Component Name Address Option
```

（*Component*:コンポーネント種別、*Name*:コンポーネント名、*Address*:使用する内部補助リレーのアドレス、*Option:* コンポーネントに応じて指定するオプション）

4.4.4 ビームライン構成の分類

PF のビームラインの構成は多種多様であるが、ビームシャッター、ビームストッパー、実験ハッチ等の構成にのみ着目すると 40 種類の組み合わせが存在

する。自動生成システムではこれらの構成に応じたサブルーチンを Call する事で生成されるロジックを切り替えている。なお、各組み合わせについてはそれぞれ番号が割り振られており、以下の宣言文を記述する事でロジックの切り替えが行える。

_BlModeA *Number*
_BlModeB *Number*
_BlModeC *Number*
_BlModeD *Number*
(_BlModeA～D:分岐ビームライン、*Number*:ビームライン構成に対する番号)

たとえば、分岐ライン B の実験ハッチが 1 つで、ダウンストリームシャッターがホワイト仕様になっている場合、構成選択宣言文は以下の通りとなる。
_BlModeB 5

4.4.5 真空系の定義

ビームライン真空に関するロジックを作成する前には、あらかじめビームライン真空に関連するコンポーネントの構成を定義しておく必要がある。自動生成システムにおいて真空関連コンポーネント構成の記述は、手動バルブを除く全てのゲートバルブを閉じた時に出来るそれぞれの区画内に存在する真空関連機器名を列挙することにより行う。たとえば図7のような構成の場合の記述は以下のようになる。

```
#Section 1
vacsection Valve1_Gauge1_Valve2_Valve4

#Section 2
vacsection Valve2_Gauge2_Valve3

#Section 3
vacsection Valve3_Gauge3
```

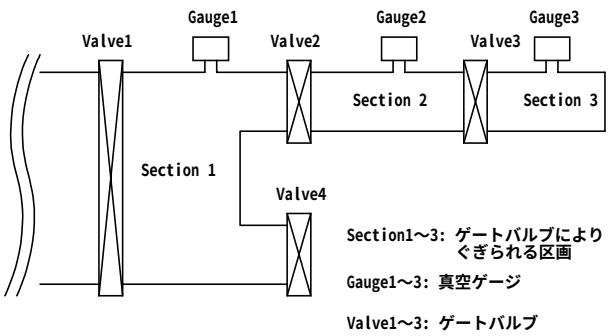

図7: 真空環境の構成例

4.4.6 各種設定の定義

自動生成システムではビームライン構成や真空系定義の他にも各コンポーネントがどの分岐ラインに所属するかなどの定義を行う。

4.4.7 動作ロジックの作成

ビームライン構成の定義や真空計の定義を全て行った後はビームシャッターを開閉するための条件などの各コンポーネントに対するロジック作成を行う。以下にその例を示す。

```
#シャッターA ロジックの作成
make mlshutter A

#ハッチ A ロジックの作成
make mlhutch A
```

4.5 自動生成

ソースファイルの記述が終了後、blisc によってニーモニック及び I/O テーブルの作成を行う。blisc の実行はコマンドラインから「blisc ソースファイル名」、あるいは「perl blisc ソースファイル名」（動作環境によって blisc が直接実行できない場合、以下例題には blisc を直接実行できる場合を使う)と入力することにより行う。

4.5.1 ニーモニックの出力

ニーモニックの作成は以下の様に blisc をオプション無しで起動することにより行う。
blisc *filename*
(*filename*: .bl の拡張子を持ったソースファイル名)
なお、ソースファイルの記述にエラーがある場合、blisc はエラー部分を CRT 上に出力し途中で停止する。エラーなく自動生成が完了した場合、".nm"の拡張子が付いたニーモニックファイルが作成される。

4.5.2 I/0 テーブルの出力

以下のように"-t"オプションを指定して blisc を実行すると、I/O アドレスに対する信号名の情報を持つ I/O テーブルが出力される。
blisc -t *filename*
(*filename* .bl の拡張子を持ったソースファイル名)
作成される I/O テーブルには".tbl"の拡張子が自動的に付加される。

4.6 サポートソフトへのプログラム転送

PLC へのプログラム転送及び実際のデバッグ作業はサポートソフトを使用して行う為、blisc によって作成されたニーモニック及び I/O テーブルをサポートソフトに転送する必要がある。しかし、現行のサポートソフトにはこれらのようなテキストファイルを直接読み込む機能は残念ながら用意されていない。自動生成システムでは Windows のクリップボードを使用し、ニーモニック及び I/O テーブルのデータを直接貼り付ける方法を使用する事でサポートソフトへのデータ転送を行う。

4.6.1 I/0 テーブルの転送

サポートソフトに I/O テーブルを転送しておくとプログラムのラダー表示の際などに、各種 I/O 名が表示されるようになる。これら I/O 名の表示は実際のデバッグ作業を効率化するために不可欠な情報である。

I/O テーブルの転送には blisio2cx.pl を使用する。Windows 上に Perl が正しくインストールされていれば blisio2cx.pl をダブルクリックすることにより、コンソール画面が現れ以下の様にファイル名の入力が求められる。

Please input filename >

ここでファイル名を入力すると I/O テーブルがクリップボードにコピーされるので、その後、サポートソフトの変数テーブル表示画面上で貼り付けを行う。

4.6.2 ニーモニックの転送

サポートソフトにニーモニックを転送するためには blisnm2cx.pl を使用する。手順は I/O テーブルの転送と同様、blisnm2cx.pl を起動するとファイル名の入力が求められるので適切なニーモニックファイルの名前を入力する。その後、ニーモニックはクリップボード上にコピーされる。最後にサポートソフトのプログラム表示モードをニーモニックにし貼り付けを行うと全てのニーモニックがサポートソフト上に転送される。

5. 集中管理システム

全ての BLIS は光ファイバーにより集中管理システムに接続されている（図 8参照）。集中管理システムはこれら BLIS から送られてくるシステムを監視すると共に、各 BLIS に対してビームライン使用の許可などの制御信号を送っている。これまでの集中管理システムの構成は図 9に示す通り、DOS を使用したパーソナルコンピュータ（以下 PC）、CAMAC 及び自作のコントロールパネルより構成されていた。

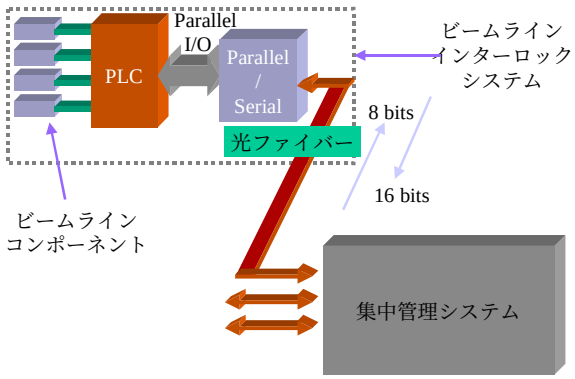

図8: BLIS と集中管理システム

集中管理システムはこれまで特に大きなトラブルも無く安定に動作してきたが、システムのプログラムは単一のシングルタスクのプログラムであった。このため、データベース更新などの些細な作業の為にもシステム全体を停止する必要がある等、保守作業上問題が指摘されていた。今回これらの問題を含め集中管理システムで使用している PC の老朽化及びハードウエアの同等品が入手困難になった等の理由から、COACK を利用した新しい集中管理システムを開発する事となった。

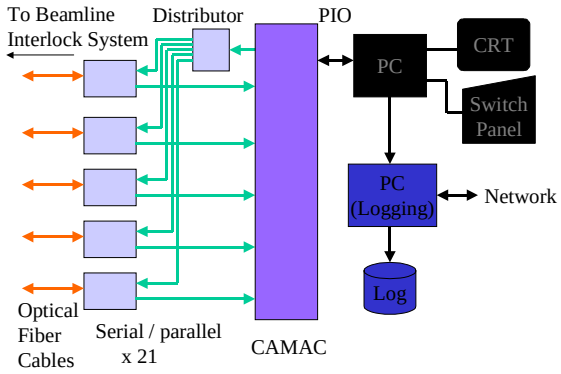

図9: 旧集中管理システムの構成

COACK はもともと加速器の制御用のシステムとして開発が行われてきたが、低価格な PC の利用が可能であること、ネットワークによるシステムの分散化が可能であること、また非常に高い柔軟性を持っている等の理由から、集中管理システムにも応用可能であり、集中管理システム自体の性能も格段に向上させる事が可能である。今回開発した新集中管理システムでは COACK を使用する事で、今までの問題点を全て克服しようとしている。

6. COACK を導入した新集中管理システム

COACK を利用した新集中管理システムの構成を図 10に示す。システムはネットワーク及び複数の PC から構成されている。ここで最も重要な PC は COACK サーバであり仮想的なビームラインのイメージをサーバ内に持っている。またこの他に、システム内には実際に各 BLIS と信号を送受するためのインターフェース用 PC（PLC Interface）、集中管理システムの実際の操作を行うためのオペレータ用 PC（Operators' PC）、集中管理システム用のプログラムを開発するための開発用 PC（Developers' PC）などの PC が接続されている。

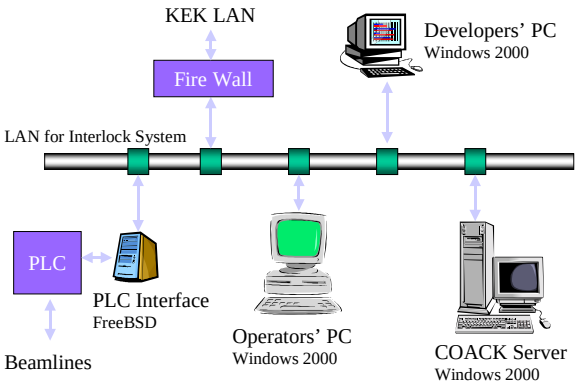

図10: COACK を利用した集中管理システムの構成

6.1 PLC への置き換え

既に述べた通り、旧集中管理システムはビームライン毎に異なった 16 のステータス信号を CAMAC のインプットレジスターを通して監視していた。今回新システムの導入によりこの部分は 2 台の PLC に置き換えられている。

各 PLC からの信号は RS-232C を利用しインターフェース用 PC に送られる。ここで、RS-232C を採用した理由は、一般的な PC は基本的にシリアルポートを有しているのと共に PC 上で動作する殆どの OS がこのシリアルポートをサポートしているからである。

各ビームラインの状況を監視する上で各 BLIS から送られてくる信号の変化はそれほど頻繁には起こらず RS-232C を利用しても十分な転送結果を得ることが出来る。しかし、ビームライントラブルなどの際にはこれらの信号が数ミリ秒程度で変化するが、これらのステータス信号がどの順番で変化したかを知る必要がある。ここでは PLC 内にステータス信号の変化をバッファリングする仕組みを開発した。

ステータスバッファリングのシステムにおいて、各ビームラインから送られてくる 16 ビットの情報が変化したかどうかは PLC の持つ XOR 命令によりチェックされる。もし、ステータスが変わったことが分かると、変化情報は PLC のデータメモリーエリアにスタックされていく。その後、スタックされたデータは非同期に PC により読み込まれる。

6.2 ハードウエア及び OS

新集中管理システムでは複数の PC が利用されている。OS としてはインターフェース用 PC には FreeBSD 4.0 Release を、他の PC には Windows 2000 Professional を使用している。この様にシステムを分散化することにで、各部分の開発や保守の際にシステム全体を停止する必要が無くなった。

6.3 ソフトウエア

旧システムでは以下に示す機能を持つ一つの大きなプログラムが動作していた。

1. 各 BLIS からの入力をチェック（入力チェック）
2. ログデータの保存（ロギング）
3. ビームライン運転状況をモニター画面に表示（ステータス表示）
4. BLIS に使用許可信号を送る（オペレート）

新システムにおいてこれらの機能は 3 台の PC に分散されている。始めに 1 及び 2 に関する機能はインターフェース用 PC に、また、3 及び 4 に関する機能はオペレータ用 PC に移行された。COACK サーバはこれらの機能の中心的なロジック部分を担当している。

なお、オペレータ用 PC で動作するプログラムは Visual Basic を利用することで高機能なグラフィカルユーザインターフェースを備えたものを作成する事が可能であった。実際のオペレータ用 PC 上で動作する集中管理システムアプリケーションプログラムの例を図 11に示す。

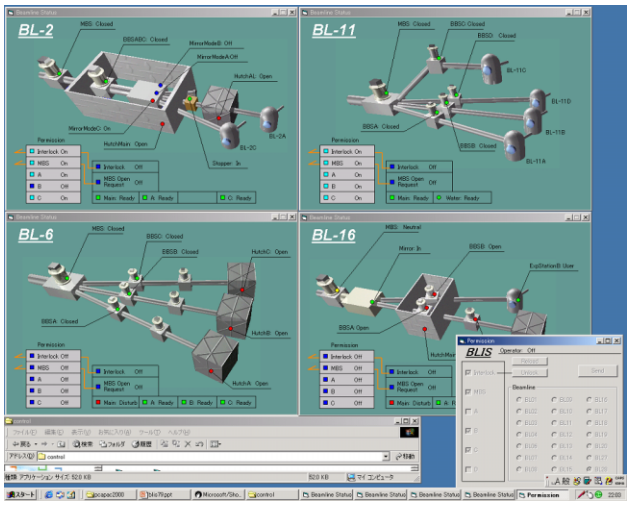

図11: 集中管理システム画面

6.4 non-Windows システム用 COACK インターフェース

COACK は Windows 2000 上で動作するシステムであり、COACK サーバを含めオペレータ用 PC などでは Windows 2000 が使用されている。しかし、集中管理システムの構成上、インターフェース用 PC には FreeBSD が使用されており、COACK に対し non-Windows システム用の何らかのインターフェースを開発する必要があった。ここでは COACK の持つ TCP/IP Socket 通信の機能を利用してインターフェースプログラムの開発を行った。このインターフェースプログラムから COACK に対するメッセージの授受は XML（Extensible Markup Language）によって行われる。また、インターフェースプログラムの核

の部分は Perl を用いて記述されており、様々な OS 上で動作する事が可能である。

なお、今回開発したインターフェースプログラムの核の部分については COACK 以外のシステムにも応用可能であり STARS（Simple Transmission and Retrieval System）[11]として別途開発を進めた。現在、STARS は PF におけるビームラインの一部の計測システムやカードリーダを利用した入退室システムに導入されている。

7. 集中管理システムにおける COACK の有用性

今回、実際に COACK を集中管理システムに導入することで以下のような有用性が確認された。

初めに挙げられる有用性としては、COACK の導入によりシステムの核となる部分についての開発は一切行う必要がないと言うことである。既にシステムの中心的部分は COACK に含まれており、実際に開発する部分は末端の GUI（Graphical User Interface）アプリケーションや装置のドライバー等だけで良いのである。実際にシステムの開発時間は短縮され、旧システムを開発したときに要した開発期間は約 6 ヶ月であったのに対し、新システムにおいては僅か 2 ヶ月で開発作業を終了してしまった。

次に挙げられる有用性は、GUI アプリケーションプログラムの開発や保守に際してシステム全体を停止させる必要が無いということである。実際に今回システムの運転を開始した後に GUI アプリケーションのバグフィックスの必要が発生したが、システム全体を停止させる事無く集中管理システムの運転自体に影響を与える事は無かった。

まとめ

BLIS についてはタッチパネルや省配線システムの導入により、高い信頼性を保持したままでの大幅なコストダウンに成功した。また、BLIS プログラム自動生成システムの開発により、これまでと比較にならない程 BLIS プログラム作成に要する時間が短縮されたと共にバグの発生も大幅に減少した。

そして、集中管理システムにおいては新しく開発された COACK の導入により、非常に保守性の高い、安定したシステムを構築することが出来た。

ここで述べた BLIS と集中管理システムに関する技術開発は、斉藤裕樹氏及び伊藤健二氏と、COACK 開発とシステムの応用については濁川和幸氏、片桐広明氏、白川明広氏、木代純逸氏、阿部勇氏、黒川眞一氏、武藤正勝氏（東北大）、柴崎義信氏（東北大）、小平純一氏（核融合研）、小川英樹氏（核融合研）、小嶋護氏（核融合研）、井上知幸氏（核融合研）、横田光弘氏（核融合研）、塚田究氏（核融合研）らと共同で行ったことを付記しておく。

参考文献

[1] T.Kosuge,Y.Saito and K.Ito Beam Line Interlock System in the Experimental Hall (BLIS) KEK Internal, 90-20 (1990)

[2] 小菅隆、佐藤能雅、伊藤健二 放射光実験施設におけるビームライン・インターロックシステムとその集中管理 プラズマ核融合技術研究会報告 (1984) 242

[3] 小菅隆、佐藤能雅 放射光ビームラインの真空保持及び放射線安全用インターロック系 名古屋大学プラズマ研究所 技術研究会報告 (1988) 206

[4] 小菅隆、斉藤裕樹、伊藤健二 放射光ビームライン・インターロック集中管理システム 核融合科学研究所技術研究会報告 (1991) 172

[5] 斉藤裕樹、小菅隆、伊藤健二 LAN を用いたインターロックシステムの監視 核融合科学研究所技術研究会報告 (1994) 228

[6] 小菅隆、斉藤裕樹、伊藤健二 放射光ビームライン・インターロックシステムとネットワーク KEK Proceedings 95-14 (1996) 23

[7] 小菅隆、齋藤裕樹、伊藤健二 ビームライン・インターロックシステムにおけるシーケンサープログラム自動生成の試み 技術研究会報告 1996・東京分科会 (1997) 59

[8] I. Abe, et al., "COACK-II PROJECT ON ACCELERATOR CONTROL KERNEL DEVELOPMENT", ICALEPCS'99, Trieste, 1999

[9] T. Kosuge, et al., "COACK APPLICATION FOR THE BEAMLINE INTERLOCK SYSTEM AT THE PHOTON FACTORY", PCaPAC2000, Hamburg, 2000

[10] I. Abe, et al., "Recent status on COACK project", PCaPAC2000, Hamburg, 2000

[11] 小菅 隆、斉藤裕樹、伊藤健二 計測・制御用簡易メッセージ配信システムの開発 東北大学技術研究会報告 (2001) 220