Development of J-PARC PPS Monitoring System for J-PARC Accelerator Control System

Hiroyuki Nemoto[1,A)], Masatoshi Adachi[B)], Masahiko Tanaka[B)]
Kikuo Kudo[C)], Yasunori Takeuchi[C)], Noboru Yamamoto[C)]
[A)] ACMOS INC.
2713-7 Muramatsu, Tokai-mura, Naka-gun, Ibaraki, 319-1112
[B)] Mitsubishi Electric System & Service Co., Ltd.
2-8-8 Umezono, Tsukuba, Ibaraki, 305-0045
[C)] J-PARC Center, KEK and JAEA, 2-4 Shirakata Shirane, Tokai-mura, Ibaraki, Japan, 319-1195

Abstract

J-PARC PPS (Personnel Protection System) is an independent and isolated system from the accelerator control system, in order to avoid unexpected influences caused by the control system. However, monitoring and recording PPS status are inevitable for efficient accelerator operations. We have developed one-way links from the PPS to the control system. Monitoring and recording PPS status are now possible at the control system without losing independency of the PPS.

J-PARC加速器制御システムからのJ-PARC加速器PPSシステム状態監視用モニタの開発

１．はじめに

PPS (Personnel Protection System：人的保護系)は、放射線被曝等の危険から人の安全を確保するためのシステムである。

J-PARC加速器でも、PPSは安全性を確保するため、J-PARC加速器制御システムとは独立し単独で動作が可能なシステムとして構築されている。今回、PPSの独立性を損なう事無く制御システムからのモニタを可能とするために、機器・ソフトウェアの整備を行った。PPSからの信号読出しのため、加速器制御システムに読出し専用のPLCを設置し、アプリケーションを開発した。このシステムと利用法について報告する。

２．PPS状態監視用モニタ

2.1 PPS状態監視用モニタ構成

J-PARC PPSは横河電機株式会社製のPLC (FA-M3[2])を用いて構成されている[1]。総ての入出力はPLCモジュール経由で行われる。一方、加速器制御システムは、標準的なTCP/IPネットワークとEPICS（Experimental Physics and Industrial Control System）[3]ツールキットを用いて構成されている。この二つの異なるシステムを接続するために、PPS側と制御システム側にそれぞれ専用のPLCモジュールを用意し、PPS側PLCのディジタル出力モジュールからの信号を制御システム側PLCのディジタル入力モジュールで読み出す事とした（図１参照）。

PPS側からは信号出力のみ、制御システム側からは信号読み出しのみであり、信号は片一方通信しか行えない。また、PLC間は信号線で結ばれているため、ネットワーク上からの不用意な操作は行えない。これらにより、PPS本体に制御システムが影響を与えることが無い事を担保している。

実際のPPSは、安全性を確保するため２重系となっている。このため制御システム側も２系統の読出し系を用意した。また、制御システム側PLCはJ-PARC 中央制御棟のPPSラック内に設置して、信号配線の取合いを簡単にしている。

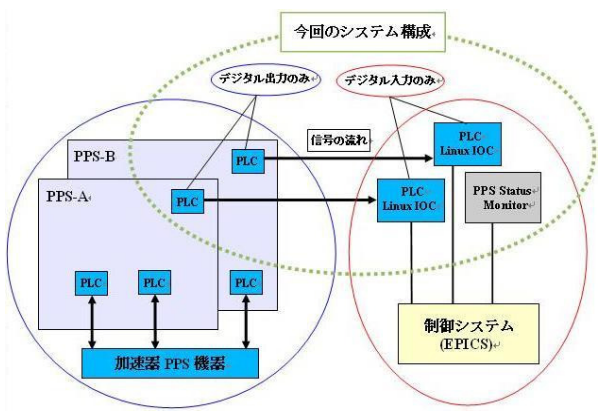

図１：状態監視用モニタ構成

[1] E-mail: hiroyuki@post.j-parc.jp

2.2 PLC構成

PPS側で制御システムに信号を送りだすために、既存のPLCシステムに出力モジュールを追加して、かつラダープログラムを修正した。既存のPPS側の変更は、ごく小規模なものですんだ。

制御システム側のPLCシステムは、新規追加である。PLCのコントローラには、通常のシーケンスCPU（ラダーを載せる）でなく、Linux OSが動作する新しいCPUモジュール(F3RP61)[2]を採用した。

CPUモジュールの内部Flashには、Linuxの上にEPICSをインストールした。その結果、このCPUモジュールのEthernetインタフェースを通して、制御端末のEPICSアプリケーションがPLCと直接通信できるようになった。F3RP61のEPICS化の詳細については、[4]を参照されたい。

図２に制御システム側PLC（２系統のうち１式）の構成を示す、２式ともに同様の構成である。

図２：制御システム側のPLC構成

３．制御アプリケーション

3.1 全体像

制御アプリケーションによる監視を実現するために、(1) EPICSデータベースの作成、(2) PPS状態のGUI画面表示、のそれぞれを説明する。

3.2 EPICSデータベース

EPICSデータベース（信号定義）は、PPSシステムの信号表のエクセル表を基に作成した。２系統で合計392チャネル分のレコード（EPICS環境での信号単位をレコードと呼ぶ)が必要になるが、これを手作業で作成するのは効率が悪い。そこで、Pythonスクリプトを用いてレコードを構築した。

元のエクセル表には「信号の名称、チャネル名、信号の説明、信号On・Off時の状態(Major/Minor)、信号On・Off時の状態を表す文字列」などが含まれている。　このエクセル表をPythonスクリプトで処理して、EPICSデータベースが作成される。本アプリケーションではデータの読込みのみなので、全レコードがInput型で構成されている。

図３に作成したデータベース（一部）の例を示す。PPSシステムは２重系であるが、データベース定義の一部分をマクロ表記（図３では“number”と”id”）して、定義ファイルを共通にした。PLCのCPUモジュールが起動するときに、マクロが実際の文字列に変換されてEPICSレコードが生成される。

EPICSデータベースを自動起動シェルスクリプトを処理する際、その中でprocServ（Process Server with Telnet Console and Log Access）[5]を使用している。CPUモジュールでサポートされているネットワーク接続方法はtelnetのみである。procServでEPICSデータベースを起動すれば、自動起動したあと特定のポートでtelnet接続することでレコードの運用状態を確認することができる。また、レコードがなんらかの異常で停止しても、procServには自動的に再起動させる機能がある。

```
record(bi,"test $(number):$(id)Testrecords ")
{
        field(DTYP,   "F3RP61")
        field(INP,    "@U0,S5,X1")
        field(ZNAM,   "Alert")
        field(ONAM,   "Normal")
        field(ZSV,    "MAJOR")
        field(SCAN,   "1 second")
}
```

図３：EPICSデータベースの例

3.3 GUI画面

GUIはEPICSの標準ツールであるMEDM（Motif Editor and Display Manager）を用いて作成した。

PPS側からの出力信号を制御システム側で読み込み表示することが目的であるため、PPSに対して画面から一切の操作を加えることはできない。

図４に記すメイン画面では、PLC2台分の392チャネル全数を同時に表示する。表示項目は、「チャネル番号、信号名、アラーム状態（二台分）、Value値画面呼び出しボタン」である。これらがPLCモジュールのSLOT番号別に配置されている。信号名やアラーム状態のseverity（アラームの深刻度レベル）は、PPSシステムから提供されたエクセルファイルに基づいて設定されている。画面右上にはPLC/IOCのTIME STAMPが表示されている。（図５(A)参照）

Value値画面呼び出しボタンで表示される画面（図５(B)）では、アラーム色だけではなく文字として状態表示することで、操作に慣れた作業者だけでなく、一般の作業者でも状態を把握しやすくしている。

画面作成には数百の信号を表示するが、これだけ多数のアラームや信号名を手入力で行うのは非常に効率が悪い。そこで、EPICSデータベース同様に、Pythonスクリプトを一部使用した。それらの表示位置、表示内容を書き込んだCSVファイルをPythonスクリプトで処理し、MEDMファイルの書式に成形した。

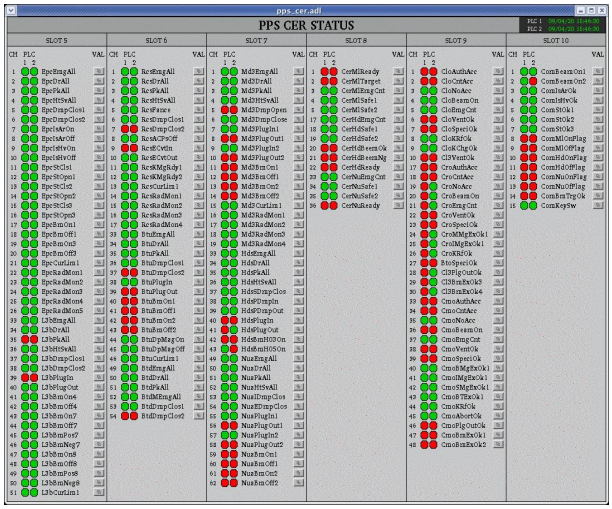

図４：メイン画面

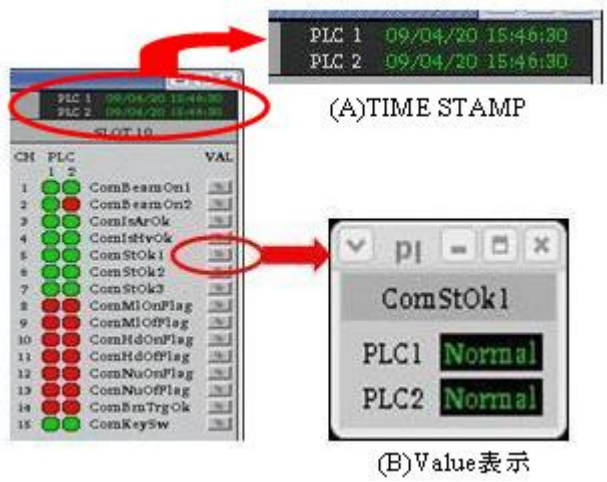

図５：各表示画面

４．まとめ

本システムの開発で、PPSの独立性を損なうことなく、その状態監視が加速器制御システム側で可能となった。また、Linux OSが動作する新しいCPUモジュール(F3RP61)を採用して、EPICSでの経験を生かした効率のよいPLC応用開発ができた。

本システムは現在正常に動作しており、機能的に問題はない。しかし、画面開発においてはスペースの制約があり、幾つか実装を見送った表示がある。

(1) VAL画面の表示内容を、MAIN画面のみで表現できないか、またはアラームとVAL画面を組み合わせて異常時のみ詳細表示できないか。

(2) 信号名を見ただけではどの信号なのか分かりにくい、そのため信号解説の付記を考えたが、スペースがなく断念した

PythonのGUIツールであるTkinterやWxPythonを用いて画面開発を行うことで、今回見送ったこれらの機能が実装可能となることが分かった。これらによる画面の作成は検討に値すると思われる。

謝辞

基本原稿準備の段階で貴重な助言をいただいた、KEK加速器研究施設の上窪田紀彦氏に感謝いたします。

参考文献

[1] F.Hiroki et.al., “Personnel Protection System in J-PARC Linac”, Proceedings of the 1st Annual Meeting of Particle Accelerator Society of Japan, (August 4 - 6, 2004) p.159
[2] YOKOGAWA web site “http://www.yokogawa.co.jp/”
[3] EPICS web site “http://www.aps.anl.gov/epics/”
[4] Junichi Odagiri. “F3RP61を利用したEPICSの加速器制御への応用”, in this meeting
[5] procServ - Information and Download “http://www-csr.bessy.de/control/SoftDist/procServ/”