Raspberry Pi を用いた CAMAC 制御システムの開発

DEVELOPMENT OF CAMAC CONTROL SYSTEM USING RASPBERRY PI

田丸哲也#, A), 内藤孝 B), 坂入崇 C)
Tetsuya Tamaru#, A), Takashi Naito B), Takashi Sakairi C)
A) Kanto Information Service (KIS)
B) High Energy Accelerator Research Organization (KEK)
C) Iwac Service

Abstract

CAMAC is used for accelerator control at KEK's advanced accelerator test facility (ATF), and more than 50 CAMAC crates are connected by serial communication using Lecroy's 5211A optical link. This system was old, built in the 1990s, had problems with the stability of the optical link and often had problems with operation. As an alternative, there is a network crate controller (CC/NET manufactured by Toyo Technica Co., Ltd.) equipped with Linux, but production has already been discontinued. Therefore, focusing on the highly functional, inexpensive and compact Raspberry Pi, we developed a CAMAC control system using the Raspberry Pi and replaced all ATF control (except for some CC/NET). In this presentation, we report the current status and future prospects of ATF control.

1. はじめに

KEK の先端加速器試験施設(ATF)では加速器制御に CAMAC[1]が使われている。CAMAC は古い規格のインターフェースであり ATF 建設当時から他の新しい規格の制御システムが検討されたが、CAMAC は既に多くの種類のモジュールが開発されていることや KEK の他のプロジェクトで使わなくなったモジュールを多数使用出来ることから採用された経緯がある。従ってモジュールによっては 30 年以上前に製作されたものを使用しているため故障も多い。

CAMAC クレートはシリアルハイウェイにより計算機と接続されており、シリアル通信ボード、CAMAC クレートにより構成されている。CAMAC クレート間はバイトシリアル(8bit data+1bit clock)の電気信号ケーブルにより接続されているが、離れた場所にある CAMAC クレートや電気ノイズの多い場所では Lecroy 社製 5211A 光リンクを用いて通信を行っていた。しかし、光リンクの安定性に問題があり、温度が上昇すると通信エラーが発生し運転に支障をきたす事があった。

その解決策として、Linux を搭載したネットワーククレートコントローラー(CC/NET[2])を一部使用しているが、すでに生産中止となっている為、全ての CAMAC クレートに適用する事は難しい。そこで、高機能かつ安価で小型な Raspberry Pi[3]に注目し、Raspberry Pi を用いた CAMAC 制御システムを開発した。昨年の 7 月から順次置き換えられ、今年の 2 月から 3 月にかけて行われたビーム運転では、一部の CC/NET を除き、Raspberry Pi による CAMAC 制御システムに全て置き換えて運転を開始した。

本稿では、更新した CAMAC 制御システムの構成と、運用状況、今後の展望について報告する。

#kan-tama@post.kek.jp

2. 開発

2.1 CAMAC の規格

ATF で使用されているクレートコントローラーは、Serial Highway Interface System ANSI/IEEE Std. 595-1892（シリアルハイウェイ）に準拠しており、バイトシリアルでデータが伝送される。シリアルハイウェイの特徴として、最大 62 台の CAMAC クレートを接続でき、数珠繋ぎの様にループして構成するのが特徴で、主に大規模な構成に向いている。しかし、ループするが故の欠点として、構成内にある SCC や光リンクに異常が発生した場合、全システムがダウンしてしまうという欠点がある[4]。

2.2 開発環境

開発に用いた言語、ライブラリ等を Table 1 に示す。

Table 1: Language and Library Used for Development

Development language	C
EPICS version	3.14.12.5
library	WiringPi,message_queue,shm

CAMAC の制御には Raspberry Pi の GPIO を用いるため、WiringPi を使用した。一度に複数のアクセスで GPIO 制御を割り込まれると、GPIO の I/O が混在し、送受信するデータの順番が崩れてしまう。その為、メッセージキューで排他制御を行う様にした。

3. CAMAC 制御システム

3.1 Serial Driver による CAMAC 制御システム

これまでの CAMAC 制御システムの構成を Fig. 1 に、Raspberry Pi を用いた CAMAC 制御システムの構成を Fig. 2 に示す。

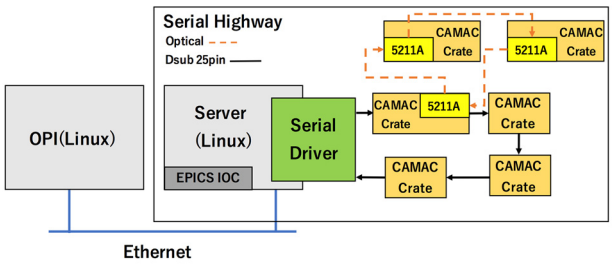

Figure 1: CAMAC control system using Serial Driver.

これまでの CAMAC 制御システムは Serial Driver を搭載した Linux サーバーから CAMAC クレートへ数珠繋ぎの様に接続し、シリアルハイウェイで構成している。離れた場所にある CAMAC クレートや電気ノイズの多い場所にある CAMAC クレートには光リンクを用いる事で通信を可能にしていた。制御サーバーで Input Output Controller(IOC)を 1 台で管理している為、ソフトウェア等の管理がしやすい反面、ソフトウェアの負荷が集中するため CPU が現在のように高速でない頃は CAMAC アクセスが遅く感じることが多々あった。ソフトウェア等はディスクアレイのサーバーを NFS マウントしている。シリアルハイウェイはリニアック用とダンピングリング用 2-Branch が稼働していた。

しかし、冒頭でも挙げた通り光リンクの通信エラーや SCC の故障などが発生すると、全ての CAMAC が使用出来なくなり復旧に時間がかかってしまう事が度々あった。

3.2 Raspberry Pi による CAMAC 制御システム

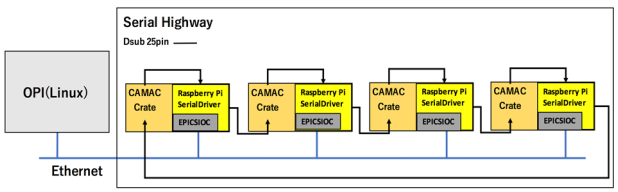

Figure 2: CAMAC control system using Raspberry Pi.

Raspberry Pi 3 Model B とバイトシリアルのインターフェースを組み合わせた Raspberry Pi Serial Driver(RPS)を製作した。RPS とシリアルクレートコントローラー(SCC)で通信を行い、シリアルハイウェイを構成した。これだけで Linux サーバーと Serial Driver の役割を担う。従って、RPS の数だけ CAMAC Branch を作ることが出来る。RPS を複数台用意し、構成台数の少ない Branch（最大で 8 台）を構成する事で、これまでの CAMAC 制御システムの欠点でもある全システムがダウンしてしまう被害を最小限に抑えられる様にした。その為、エラーの発生箇所が分かりやすくなり、以前と比べ素早いトラブル対応が可能となった。ソフトウェア等はディスクアレイのサーバーを NFS マウントして利用しており、1つ1つにソフトウェアを用意する手間を省き、MicroSD カードの寿命を延ばす様にした。万が一 RPS が故障した場合、基本的には RPS を交換してマウント先を変えるだけで対応が可能である。また、何らかの原因で RPS との通信が出来なくなった場合でも RPS 本体の電源を ON/OFF するだけで、制御に必要なプログラム等を自動的にバックグラウンドで実行し復旧出来る様にした。これらにより以前と比べて復旧にかかる時間が削減された。ここまでで改善された点を挙げたが、移行にあたり悪くなった点もいくつか存在する。IOC を分散させた為、どの RPS が何の制御をしているかが分かりづらくなってしまい、管理が煩雑になった。また、ATF ではブロック転送を必要とする CAMAC モジュールを使用していないため、RPS はブロック転送機能を付加していない。

3.3 通信速度

ATF では制御する CAMAC の台数が多い為、通信速度は重視すべき点である。特に加速器のマシンサイクルに同期したデータ処理を行うためには 0.3 秒の間に全ての処理を終える必要がある。リアルタイムではないため時間保証はなされないが、高速性が要求される。その為 CAMAC アクセスを行ってから、Q、及びデータが返されるまでの時間を測定するテストを実施した。

条件として、SCC は Hitachi 製 NA-002、CAMAC アクセスの関数実行前と実行後の時間差を出力するプログラムを C で作成した。構成台数は 1 台で、Function は F0(Read)、F16(Write)で、CAMAC アクセスするモジュールは R/W が可能な 24bit switch register を使用した。これを Read, Write それぞれ 100 回行った時の計測結果を Fig. 3 に示す。

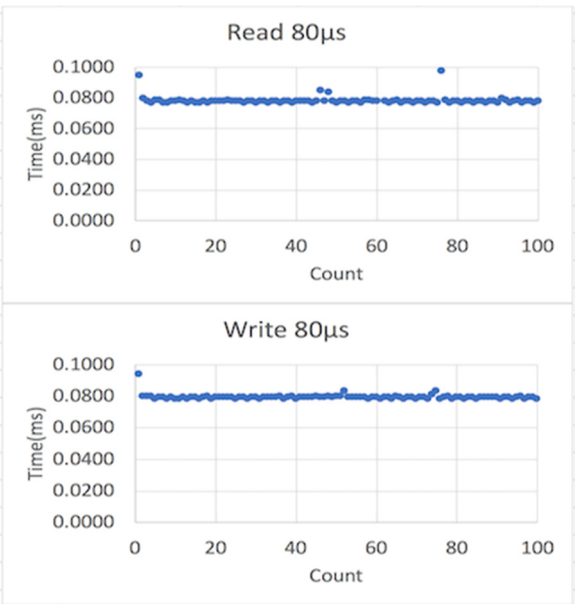

Figure 3: Speed result of read and write by Raspberry Pi.

Read 1 回の平均値が 79.1 µs、Write 1 回の平均値が 79 µs であった。なお、Read、Write を 1 回の組み合わせとした場合の 1 回の平均値が 157.8 µs であり、

ほぼ Read、Write を合わせた時間となった。1 台のクレート制御専用に開発された CC/NET のシングルアクションは 9 μs [5]程度なので CC/NET の方が速いが、複数台 CAMAC クレートを制御出来るという点、そして低コストであるという点では非常に優れている。

3.4 アラームについて

CAMAC アクセスのエラー原因は複数にわたる。特にビーム運転中にはすぐにオペレーターに知らせ、エラーの原因を把握する必要がある為、アラームを用意した。アラームトリガーと内容を Table 2 に示す。

Table 2: Alarm Trigger and Contents

アラームトリガー	アラーム内容
CAMAC アクセスによるエラー	クレート、ステーションナンバー,アラーム内容を表示
メッセージキューサーバーの停止	Raspberry Pi のホストネーム,アラーム内容を表示
IOC の停止	クレート,アラーム内容を表示

3.5 Raspberry Pi の耐久性

ATF では空調のない場所が多く CAMAC クレートが置いてある場所は熱やノイズによる外的要因でトラブルが発生することが多かった。今回、RPS をインストールするにあたり産業用でない Raspberry Pi 本体と MicroSD カードの耐久性に不安があった。

そこで、熱への耐久性があるかを調べる為テストを実施した。45 □の環境下で、CAMAC アクセスを 1 秒に 1 回行うテストを 1 週間行ったところエラーの発生は無く、その後も正常に動作を続けていた。上記の件から、熱に関しては問題がないことが分かった。

寿命やその他外的要因による故障に関しては一年近く運用しているものもあるが、現時点で故障したケースが発生していない為どれくらい持つのかは不明である。

4. Raspberry Pi Serial Driver (RPS)

RPS をインストールするにあたって、CAMAC クレートの環境によってスペースがない場合があった為、3 種類の形態の RPS を用意した。インストールした RPS の一例を Fig. 4 に示す。

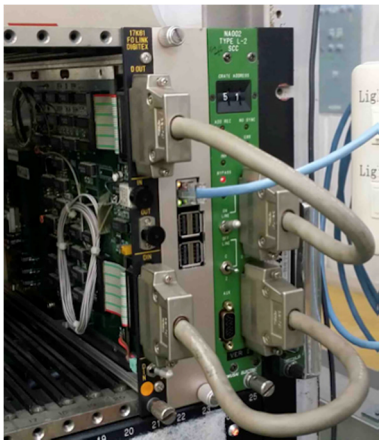

Figure 4: CAMAC controller using optical module.

写真は光モジュールのバイトシリアルのインターフェースを利用して作成した RPS である。電源は CAMAC クレートのバスから取っている。スロットの空きを 3 スロット分必要とする為、スロットの空きが無い CAMAC クレートでは使用出来ない。2 スロットで使える様に改良した RPS や、スロットが全て埋まってしまっている CAMAC クレートに対応が可能な、ラックに取り付けるタイプの RPS を製作した。

5. まとめ

ATF では、現在 CAMAC クレートが 54 台（内 CC/NET での制御が 10 台）、44 台を RPS 25 台で制御している。今後、安定に運用する為に、ハードウェア、ソフトウェアの両面で様々なトラブルに対応出来る様に予備機の用意や、故障した際に短時間で交換出来る様にソフトウェア、マニュアルを整える予定である。

CC/NET も部分的に利用しているが、予備が無く、故障した場合に交換出来ない為、RPS による制御へ置き換える。また、並行し現環境で Raspberry Pi や MicroSD カードの寿命がどれくらい持つかを調べる予定である。

参考文献

[1] Modular Instrumentation and Digital Interface System (CAMAC) ANSI/IEEE Std. 583-1982.
[2] **http://www.tcnland.co.jp/wp/wp-content/uploads/2015/12/CCNET-取扱説明書 Rev_G.pdf**
[3] **https://www.raspberrypi.org**
[4] **http://accwww2.kek.jp/oho/OHOtxt/OHO-1985/txt-1985-B-1.pdf**
[5] **http://www.nucl.phys.titech.ac.jp/presen/data/thesis/b/ay2004/2004B4_Koroku_thesis.pdf**