EtherCAT デバイスへの EPICS 実装手法の検討

STUDY ON EPICS IMPLEMENTATION METHOD FOR EtherCAT DEVICES

内山暁仁#, 込山美咲
Akito Uchiyama #, Misaki Komiyama
RIKEN Nishina Center

Abstract

In recent years, TCP/IP-based protocols such as SCPI have been convenient and widely used in device communication. TCP/IP-based devices are frequently used in the RIBF control system constructed using EPICS, and communication between them and the EPICS IOC is realized by asynchronous socket communication via AsynDriver, NetDev, and Stream Device. On the other hand, there are often cases where the socket is not reconnected between the EPICS IOC and the device after the socket communication is cut due to an unexpected power outage due to a power failure of the device or network switch. In this case, the EPICS IOC may need to be restarted, hindering accelerator operation. In order to improve the reliability of these communications, we examined the introduction of EtherCAT, a field network using Ethernet in the physical layer. An EtherCAT-based system consists of a master and slaves. This prototype system implements the EPICS IOC on the same CPU module as the EtherCAT master. The EtherCAT master communicates with multiple EtherCAT slaves. The EPICS IOC has device support created using a Python-based API to interface to the EtherCAT master.

1. はじめに

RIBF 加速器施設の制御システムは EPICS (Experimental Physics and Industrial Control System)ベースで構築されている。 2002 年旧施設で運用当初、VME CPU にインストールされた EPICS IOC（Input/Output Controller）はデバイス間のインターフェースとして VME バスを利用しており、ドライバ経由でデバイスにアクセスしていた[1]。その後、EPICS R3.14 シリーズの導入により、それらインターフェースにバス接続だけでなく、IP プロトコルの利用が進んだ[2]。現在 RIBF 制御系では自社開発のデバイスである N-DIM、また MELSEC, OMRON, FA-M3 といった各社 PLC 等が多用されており、これらは Ethernet で接続され NetDev[3], AsynDriver[4, 5]を用いたデバイスサポートで運用されている。デジタルマルチメータといった測定機器はプレゼンテーション層のプロトコルとして SCPI (Standard Commands for Programmable Instruments) と呼ばれる ASCII ベースの通信コマンドが用いられているケースが多く、これらは Stream Device[6]を用いたデバイスサポートで運用されている。

OSI 参照モデルの物理層とデータリンク層のプロトコルである Ethernet は非常に便利で扱いやすく、現在多くのメーカーの機器で標準装備されている。一方で、これらデバイスと IOC 間は通常、非同期なソケット接続で通信させるが、停電明けやデバイスの不具合による再起動、予期せぬスイッチの電源断といった後に、しばしばソケット接続が戻らず、IOC を再起動させることによって再接続をさせているケースがある。これらが円滑な加速器オペレーションの妨げにもなっている。また Ethernet のプロトコルとしての利便性を生かしたまま、IP ネットワークで実現できなかった速いレスポンスが必要なアプリケーションへの対応も今後必要になってくる、と予想される。これらを解決する手段として Ethernet ベースのフィールドネットワークの一つである EtherCAT について RIBF 制御系への実装手法の検討をした。

a-uchi@riken.jp

2. デバイス監視の従来手法

2.1 死活監視

ハートビートがデバイスの死活監視として実装される事がある。例えば、デバイス側に 1 秒毎に 1 と 0 に値が変化するレジスタを組み込み、それを EPICS 側で定期的に読みだし、値に変化がなかった場合は何らかの不具合が起きていると判定する仕組みである。これを用いる事により、上位からソケット通信の健全性を判断する事はできると考えられる。一方でデバイス側にその様な機能を実装できるとは限らず、その様な仕組みがない場合、IOC とデバイス間のソケット通信の健全性を判断する事は簡単ではない。RIBF では、SCAN しているアナログ値にノイズ等の変化も全くなく、ある時間同じ値がずっと読み出されるといった状況証拠を基に IOC の再起動を試みるといった運用をするケースもある。

2.2 EPICS 管理システム

RIBF 制御系では、EPICS 管理システムが運用されている[7]。これは IOC がどの様な PV を持つか、または、どのデバイスと接続しているか等を管理するシステムである。仕組みとしては、EPICS スタートアップスクリプトを基にデータベースファイルをパースし、record 名や構成されている field、そしてアクセスするデバイス名をリレーショナルデータベースに格納し、ウェブアプリケーションを用いて検索等に利用している。

EPICS 管理システムの機能の一つに、デバイス監視する仕組みがある。これはデバイスのホスト名とポート番号を基にポートスキャンし、ポートがオープンできなかったデバイスに関しては、ICMP で疎通を確認するというシーケンスで監視を行っている。デバイス監視の結果を

示すウェブアプリケーションのスクリーンショットを Fig. 1 に示す。一方で、ポートオープンできる数はデバイスによって決まっており、オープンできなかったとしても、それがデバイスに問題があるとは限らない。また、例え EPICS 管理システム側でデバイスのコネクションをオープンできたとしても、IOC のソケットとは別なコネクションであるため、それをもって IOC とデバイス間の接続に対する健全性を必ず判断できる、というわけではない。

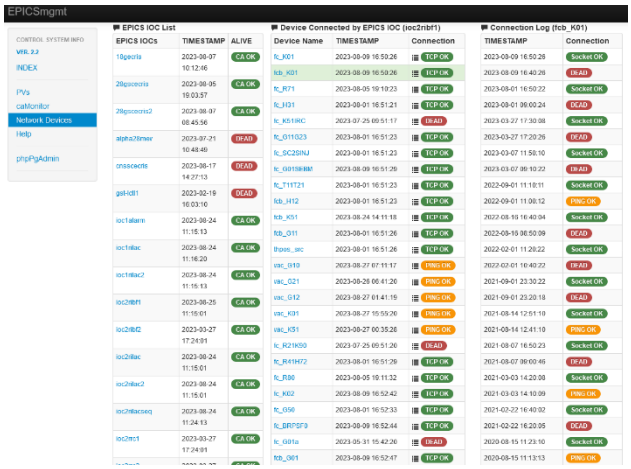

Figure 1: Screenshot of the web application with alive monitoring for network-based devices by EPICS management system.

3. フィールドネットワーク

3.1 信頼性・性能

EPICS Channel Access で取得した値をトリガーとして利用する Soft Machine Protection System は簡便に実装できるという利点から、J-PARC MR では多くの導入例がある [8]。また同様の仕組みは RIBF でも小規模ながら実装されている[9]。一般的に IP プロトコルはバスに比べ遅い通信になるため、レスポンスは遅く、リアルタイム性は低い。IOC がデバイスにアクセスする SCAN 間隔を考慮すると、1 秒以下の反応時間では対応できると考えられる。

一方でインターロックの信頼性は重要であり、恒久的なインターロック信号として使用する時には、より信頼性の高い仕組みにし、安全性を高めるべきである。RIBF では次期計画も検討されており[10]、ビーム強度の大幅な増強に対し、制御システムを柔軟に対応させると共に、信頼性と性能の向上が求められている。上記を実現する手法として Ethernet ベースのフィールドネットワークの導入を検討している。

3.2 EtherNet/IP

EtherCAT 同様、物理層に Ethernet を持つフィールドネットワークである、EtherNet/IP の検討も行っている [11]。RIBF 制御ネットワークは既に計算機室、電源室、そして加速器室内に張り巡らされており、EtherNet/IP は TCP/IP とネットワークを混在する事が可能な仕様になっている。したがって、EtherNet/IP は専用ネットワークが必要なく、通常の IP ネットワークのスイッチにそのまま接続でき、Ethernet ケーブルの敷設を最小限に済ませられる事から、高い利便性があると考えている。

3.3 EtherCAT

一方、EtherCAT は専用ネットワークが必要で、TCP/IP とネットワークを混在する事ができないので、新たにケーブル敷設をする必要がある。しかし、マスタとスレーブ間の通信が汎用プロトコルの中では高速なため、EtherNet/IP より速いレスポンスが実現できると考えられる。また、高いリアルタイム制御ができる点が長所である。今後加速器が高度化され、ビーム強度が増強された時に必要になる、高速なインターロック信号の出力や精度の高い同期制御が、簡便に可能になると考えている。

4. プロトタイプシステム

4.1 EtherCAT マスタ

一般的に EtherCAT は 1 台のマスタと複数のスレーブから構成される。今回プロトタイプとして実装した EtherCAT マスタは株式会社インターフェースの SuperCD® [12] を用いた(Fig. 2 参照)。仕様を Table 1 に示す。SuperCD®は IoT 向け小型エッジコンピュータであり、OS として Debian 9.3 (stretch) をカスタマイズした Linux (Interface Linux System 8) がインストールされている。インターフェース社の SuperCD®は LabVIEW ベースの IOC として RIBF 制御系で既に実績があることも選定した理由の一つになっている[13]。EtherCAT マスタ機能もソフトウェアとして株式会社インターフェースが提供しており、この Linux 上で走らせて運用される。

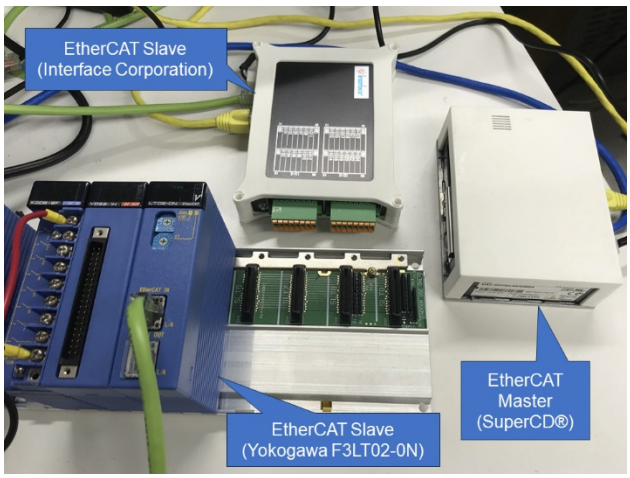

Figure 2: Photograph of the implementation test on the prototype system with one EtherCAT master, two EtherCAT slaves, which are Yokogawa F3LT02-0N, and digital input/output modules provided by Interface corporation.

4.2 EtherCAT スレーブ

EtherCAT スレーブとして、インターフェース社のデジタル入出力(ELS-293133)と横河電機の FA-M3 モジュールである F3LT02-0N を実装した。テスト環境のシステムチャートを Fig. 3 に示す。F3LT02-0N を EtherCAT スレーブとして利用するには、FL-net で接続する時と同様に、

WideField3[14]を用いてシーケンス CPU 側に「FA リンク/FL-net 系統の設定」で PDO 自動リフレッシュにリンクレジスタを割り当てる必要がある。このシステムでは、F3LT02-0N を利用したスレーブからマスタへの送信領域は W0001、マスタからスレーブへのデータ受信領域を W4097 で行った。これらリンクレジスタを介して、EtherCAT マスタは PLC CPU に命令やデータの送受信をする事が可能である。一方で、EtherCAT スレーブ間でのデータのやり取りは、スレーブ 1→マスタ→スレーブ 2 といった具合に、一旦マスタを介する必要がある。

Table 1: SuperCD® Hardware Specifications

CPU	Processer	Intel Atom E3845 1.91 GHz
	Number of cores	4
	Number of threads	4
Memory	4 GB	
Storage	SSD 32 GB	
LAN	2 ports (1000BASE-T)	

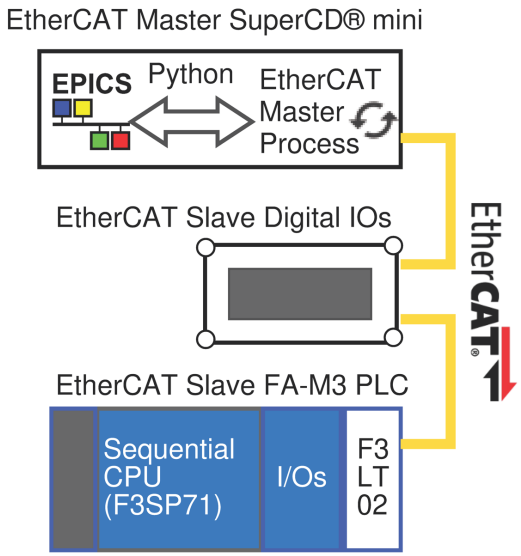

Figure 3: System chart of the prototype system using EtherCAT and EPICS.

4.3 インターフェース

EtherCAT マスタ上で、スレーブへのデータアクセスやマスタ自身の状態取得は全てオブジェクト変数によって取り扱う。例えば F3LT02-0N のリンクレジスタ W0001 が ENI ファイルによって定義されたマッピング"TxPDO-Map00"の場合本システムでは"ID1002_Read-4000"と表される。株式会社インターフェースから提供されている開発環境ではライブラリが用意されており、このオブジェクト変数にライブラリでアクセスする事により、C 言語や Python 等での開発の閾値を下げている。Table 2 に Python を用いて F3LT02-0N のデータ取得するサンプルプログラムを表す。

Table 2: Python Sample Program to Retrieve Data from the EtherCAT Slave using the Object Variable

```
#!/usr/bin/python3
import sys
sys.path.append("/usr/share/interface/CS/samples/python/lib/
")
import ifcsm
ObjectHandle= ifcsm.ifcsmGetObjectHandle("ID1002_Read-
4000")
if ObjectHandle < 0:
    print("ifcsm.GetObjectHandle() failed")
Ret, DI_val = ifcsm.ifcsmRead(ObjectHandle, 0, 1)
if Ret < 0:
    print("ifcsmRead() failed")

print(DI_val)
```

4.4 EPICS デバイスサポート

EtherCAT 用 EPICS デバイスサポートは PyDevice[15]を用いて開発を行った。PyDevice は Python インタプリタ用の EPICS デバイスである。C 言語ベースの Soft IOC から Python 関数を呼び出し、EPICS データベースと接続することができる。PyDevice を用いる事で、EtherCAT プロトタイプシステム用に EPICS デバイスサポート開発の閾値を飛躍的に下げる事ができている。4.3 で示した EtherCAT スレーブである F3LT02-0N のリンクレジスタ W0001 の値を取得する EPICS デバイスサポートの例を Table 3、EPICS データベースの例を Table 4 に示す。

Table 3: Example of Epics Device Support Written in Python. It Saved Under the File Name pydevEtherCAT.py

```
import sys
sys.path.append("/usr/share/interface/CS/samples/python/lib/
")
import ifcsm

def Read(ObjectName):
    handle = EtherCATHandle(ObjectName)
    val = EtherCATRead(handle)
    return val
```

Table 4: Example of EPICS Record Calling the Device Support Using PyDevice From the EPICS Database

```
record(longin,"PyDev:EtherCAT:ID1002_Read") {
  field(SCAN,".1 second")
  field(DTYP,"pydev")
  field(INP,"@pydevEtherCAT.Read('ID1002_Read-
4020:0001')")
}
```

5. 実装結果

SuperCD® で実現できる EtherCAT マスタのプロセスデータ通信周期は最速 100 μsec であり、スレーブ数 2 では実際その周期で動作する事ができた。IOC から Python ベースのデバイスサポート経由で EtherCAT マスタにアクセスし、EtherCAT スレーブへの制御は問題なくできている。EtherCAT マスタと EtherCAT スレーブが接続されているケーブルを抜き、切断状態を意図的に作ると、ネットワークエラーを検出し、それらの情報は上位クライアントで取得が可能である。また、実際にアクセスしている EtherCAT スレーブ数をモニタする事もできている。作成されたテスト用 GUI のスクリーンショットを Fig. 4 に示す。

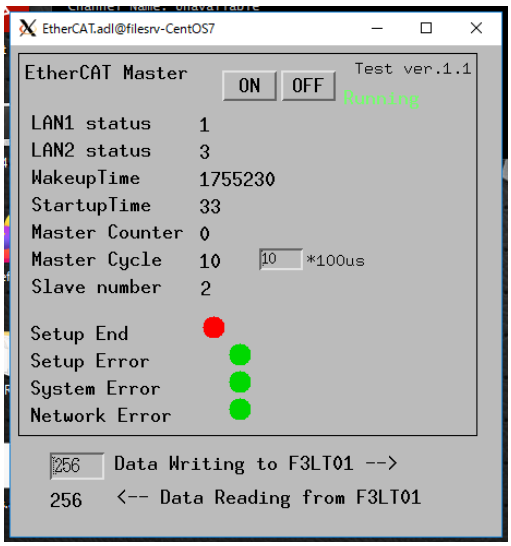

Figure 4: Screenshot of the GUI, which is monitoring the status of the EtherCAT master via EPICS IOC.

6. まとめと今後

物理層としての Ethernet の有用性を生かしたまま、デバイスの信頼性と性能向上をさせる目的で、Ethernet ベースのフィールドネットワークに着目した。それらフィールドネットワークの一つである EtherCAT へ EPICS の実装手法の検討を行った。株式会社インターフェースから提供される EtehrCAT マスタは、データアクセスをオブジェクト名で行うことができ、また Python インターフェースが用意されている事から、PyDevice を用いて簡便にデバイスサポートを開発する事ができた。

一方で EtherCAT スレーブ間のデータのやり取りは EtherCAT マスタを介する事になるため、EPICS sequencer[16]や EPICS データベースといった上位層でこれを行うと、プロセスデータ通信周期の性能を生かし切れない可能性がある。SuperCD® は 4 コア CPU を持っており、1 コアは EtherCAT マスタのプロセスで占有されている。近い将来その内の 1 コアを使ったソフトウェア PLC の提供が計画されている事から、EtherCAT、ソフトウェア PLC、EPICS を組み合わせたシステム評価を行う予定である。

謝辞

システム初期導入で株式会社インターフェースに支援いただきました。

参考文献

[1] M. Komiyama *et al.*, “EPICS を用いた理研加速器研究施設制御系の現状”, Proc. 14th Symposium on Accelerator Science and Technology, Tsukuba, Japan, Nov. 2003, pp. 272-274.
[2] M. Komiyama *et al.*, “Status of Control System for RIKEN RI-Beam Factory”, Proc. ICALEPCS’07, Oak Ridge, TN, USA, Oct. 2007, pp. 187-189.
[3] https://github.com/shuei/netDev
[4] https://github.com/epics-modules/asyn
[5] A. Uchiyama *et al.*, “Development of EPICS Device Support for PIC-Based I/O Card with a Network Interface”, Proc. 5th Annual Meeting of Particle Accelerator Society of Japan and the 33rd Linear Accelerator Meeting in Japan, Higashihiroshima, Japan, Aug. 2008, pp. 454-456.
[6] https://github.com/paulscherrerinstitute/StreamDevice
[7] A. Uchiyama *et al.*, “EPICS PV Management and Method for RIBF Control System”, Proc. ICALEPCS 2015, Melbourne, Australia, Oct. 2015, pp. 769-771.
[8] K. Sato *et al.*, “Development and Operation of the EPICS-based Soft-MPS in J-PARC MR”, Proc. 16th Annual Meeting of Particle Accelerator Society of Japan (PASJ2019), Kyoto, Japan, Jul-Aug. 2019, pp. 279-282.
[9] M. Komiyama, M. Fujimaki, N. Fukunishi, K. Kumagai, A. Uchiyama, and T. Nakamura, “Recent Updates on the RIKEN RI Beam Factory Control System”, Proc. HIAT’15, Yokohama, Japan, Sep. 2015, pp. 104-106.
[10] H. Imao *et al.*, “The Present Status and Future Plan with Charge Stripper RING at RIKEN RIBF”, Proc. IPAC’22, Bangkok, Thailand, Jun. 2022, pp. 796-801.
[11] A. Uchiyama *et al.*, “Evaluation of PLC-based Ethernet/IP Communication for Upgrade of Electromagnet Power Supply Control at RIBF”, Proc. CYC2022, Beijing, China, Dec. 2022, paper TUBO04.
[12] http://www.interface.jp/super/
[13] R. Koyama *et al.*, “Upgrade of Gas Stripper Control System for System Integration at RIBF”, Proc. 16th Annual Meeting of Particle Accelerator Society of Japan, Kyoto, Japan, Jul. – Aug. 2019, pp. 865-868.
[14] https://www.yokogawa.co.jp/solutions/products-and-services/control/control-devices/programmable-logic-controller/plc-software/plc-programming-tool/
[15] https://github.com/klemenv/PyDevice
[16] https://github.com/ISISComputingGroup/EPICS-seq