「加速器」Vol. 6, No. 4, 2009（321–334）

話　題

EPICS on F3RP61 の開発と応用

小田切　淳一*

Development and Application of EPICS on the F3RP61

Jun-ichi ODAGIRI*

Abstract

A new type of front-end controller was developed for accelerator control systems based on the Experimental Physics and Industrial Control System（EPICS）. The controller comprises a new type of CPU, the F3RP61, running Linux and existing I/O modules for the FA–M3 Programmable Logic Controller（PLC）. By running the EPICS core program on the CPU, it turns into an EPICS Input/Output controller（IOC）to replace the existing front-end control system, which is composed of a VME/VxWorks-based IOC and ordinary PLCs. The consolidation of the front-end control significantly reduces the workload for developing and maintaining the application programs as well as the relevant EPICS driver programs. Tens of F3RP61-based IOCs have been operating well for more than a year in the control systems of accelerators in operation, such as KEKB and J–PARC, demonstrating the stability of EPICS on the F3RP61.

1.　はじめに

ある日，筆者がいつものように出勤すると，机の上に新型の PLC 用 CPU モジュールのカタログが置かれていました．その上には，山本昇さん（元上司）の非常に「個性的」な字で，「この上で ITRON が動くそうです」と記されたメモが付されていました．それを見た筆者は直ちにメッセージの意味を理解しました．

この CPU の上で EPICS を動かせ

この瞬間から長い話が始まりました．言うは易し．行うは難し．実現するまでには紆余曲折がありました．苦しい場面では人（元上司）を恨んだこともありました．また，逆に助けてくれた人（元上司を含む多くの方々）の情けの有難さに涙したこともありました．出だしの掴みは取れたと思いますので，早速，本題に移らせて頂きます．

2.　EPICS とは

導入の短い「はじめに」を抜けると雪国だった，というわけではありませんが，話の展開上，どうしてもここで Experimental Physics and Industrial Control System（EPICS）なる，長い言葉をご説明しなければなりません[1]．なるべく手短に済ませますので，しばし，お付き合いを．

唐突に本音を言えば，ここで EPICS の話などしたくなかったのです．EPICS が嫌われているからです．正確に言えば，制御ソフト全般が嫌われているからです．これまでの長年の暗い思い出が蘇るからです．建設中の加速器の立ち上げが迫ると，“EPICS”と言っただけで，誰もが急な用事を思い出して逃げていきました．時には筆者の顔を見ただけで進路を変える人までいました（避けられた理由が EPICS 以外にもあった可能性については十分に認識しています）．気持ちは分かります．金物が何より優先される加速器業界では，ソフト開発など面倒なだけで評価に繋がらないからです．

愚痴が長くなりましたので，本題に戻ります．EPICS は加速器や大規模実験装置の制御のためのソフトウェアです．無償で Web からダウンロードできますが，ダウンロードしても，いきなり制御はできません．EPICS はソフトウェア部品の集まりであって，ユーザがその部品を使ってアプリケーションを作らなければならないからです．

もともと UNIX 系の世界なので敷居が高いことも事実です．しかし，一旦，覚えてしまえば，その有難さが身にしみて分かります．部品から自動車を組み立てるのと，鉄鉱石の採掘から始めるのと，どちらがお好みですか？　部品の組み立てから始められることの

* 高エネルギー加速器研究機構　KEK, High Energy Accelerator Research Organization
（E-mail: jun-ichi.odagiri@kek.jp）

有難さ，そのために先人が払った血の滲むような努力に思いを馳せるべきです．EPICSは頭を垂れて有難く頂戴すべきものであって，決して忌み嫌うべきものではありません．

さて，主観的な議論は止めにして，上述の部品から組み立てられるアプリケーションが二つに分けられることを指摘します．ひとつは，被制御機器と向き合うInput / Output Controller（IOC）と呼ばれる計算機（VMEやPC）の上で使います[2]．もうひとつは，サーバ計算機の上で使われます．簡単なお絵かき操作でGUIを作れるツールがあります[3]．データ・アーカイブのためのツールもあります[4]．ともかく，EPICSの一番すごいところは，IOC側とサーバ側でチャネル（データの通路）の名前を一致させるだけでサクッと通信ができる[5]．超気持ちいい話ではありませんか．

いずれにせよ，このような機能があるおかげで，IOCが直接機器を制御していた時代には話は単純でした．ところが，最近になって，IOCと被制御機器の間に割り込むものが現われてきました．それがProgrammable Logic Controller（PLC）です．

3.　組込みEPICSの必要性

3.1　PLCは立派な計算機

近年，PLCが加速器制御に多用されていることは周知の事実と思います．PLCはハードウェア相当の堅い制御用コントローラであると信じられ，信頼されています．しかし，実は，その実態が極めて高機能な

ソフトの塊

であることは意外と知られていません．信頼性があるのは，単に開発工程において検証試験と品質管理に十分な投資がなされているからです（この点については後節で補足します）．高機能で高信頼性のPLC．大変に結構な話です．では，それで目出度し，目出度し，で済むかと言えば，そうではありません．

まず，PLCが「ソフトの塊」である以上，使いこなすためには相当の学習コストが発生します．PLCでロジックを組むために使われるラダーのプログラミングは簡単だ，という声を耳にすることがありますが，それはラダーが簡単なのではなく，プログラム自体が簡単だったのではないかと思います．実際，筆者はUNIXのソケットっぽい命令や割込み処理を組み合わせた複雑なラダーを書いたことがありますが，書いた本人ですら，後で読み返すことが困難でした．

また，ラダー・プログラムには字面と実際に行われる処理の間にギャップがあるように思えます．実際，このためにハマったケースを何度か経験しました．機器制御プログラミングにおいては，字面とハードウェアのレジスタへのI/Oの間に挟雑物がないことが重要です．そうでなければ精緻なプログラミングはできません（その意味ではアセンブラで書くのがベスト，ということになりますが，そこは程度の問題です）．そのようなレベルでラダー・プログラミングを極めている人はごく少数に過ぎないのではないかと思います．

学習コストの高さで悪評名立たるEPICSとソフトの塊のPLCの組合せ．これは最悪です．せめてどちらか一方にして欲しい．人生に苦しみは一つで十分です．

3.2　犯人探しの難しさ

第二の問題は不具合発生時の原因究明です．EPICSをベースにした制御システムでは，PLCはIOCの下にぶら下がることになります．両者はイーサネットを介して通信します．IOCとPLCの双方のプログラミングを終えて，いざ試験．よくあることですが，思った通りに動かない．

しかし，試験段階で不具合が見つかるのは幸運なケースです．不幸なケースは加速器のコミッショニングの最中，いざビームが回らん，と盛り上がったところで不具合が発覚した場合です．施設長や主幹までもが中央制御室に勢揃いの緊迫した状況で，「あらら…　」となった場面をご想像下さい．しかも，マーフィーの法則によって，このような場面は賞与の査定時期に起こりがちです．

さて，ここで問題になるのはこれです．

IOCとPLC，悪いのはどちら？

宗教上の理由からでしょうか？　EPICSのプログラマはPLCを触らない，PLCのプログラマはEPICSには近づかない，という，はっきりとした傾向があります．多くの場合，まず別人がプログラムを作成したと考えて良いでしょう．こういった状況で不具合が起きたとき，人間は誰でもまず相手を疑います．瑕疵担保責任のなすり合いが続き，職場内に無用な軋轢と拭い難い不信感を生じる可能性すらあります．このような状況は決して好ましいことではありません．

3.3　EPICSドライバの実装コスト

問題はこれで尽きてはいません．そもそもIOCとPLCの通信のドライバはどうするのか，という問題が残っています．

たかがソケット通信じゃないか

と思われる読者がいらっしゃったとすれば，誠に残念です．たしかに単なるお試しプログラムであればその通りですが，1ミリ秒程度の時間を要する遅いネットワーク通信をサポートするEPICSのドライバには

コールバックという仕掛けが必要です．コールバック自体はそれほど難しいことではありません．まず電話をかける．一旦切って相手からの呼び出しを待つ．大したことはなさそうです．本当にそうでしょうか？

問題は，このコールバック処理と，タイムアウト処理（相手からの応答がなくて，しびれを切らせてキャンセルする場合の処理）が競合した場合に生じます．こういった時間差がある入出力の例外処理では必ず間の悪いタイミングが生じます．

誰かが5万円で車を売りたい，とメーリングリストに売り注文を投げたとします．別の誰かが，買いたい，と応じて返信しました．しかし，返信が売り手のメールボックスに届く前に，売り手は気が変わって取消しのメールを送信した．どちらのメールも未だどこかのルーターで処理中です．さて，この先，どのような順序でメールが着信すると契約が成立するのでしょうか？　答は忘れてしまいましたので民法典をご参照ください．民法では隔地者間での契約成立の条件を細かく規定しています．これは，通信用のドライバを作成する際，問題となり得る微妙なタイミングとして，どのようなケースを予め考慮しておくべきか，を検討する上で大いに参考になります[6]．

いずれにせよ，このような面倒な問題はすべての通信に共通する話です．新しいPLCとの通信をサポートするたびに民法典を参照していたのではたまりませんので，この面倒な部分は共通のドライバとして一つだけ作ることにして，機種のプロトコルに依存する処理（こちらは単なる文字列操作）と分離し，これらに上述のドライバを共有させることにすれば，容易に多機種のPLCとの通信を実現できるようになります．

このような設計方針に基づき，KEKのEPICS横断化グループは，理化学研究所仁科加速器研究センター（以下，理研仁科センター）の制御チームと共同でNetDevというライブラリを作成しました[7]．その成果について筆者がこれまでに著した論文の中では，「頭を使ったから多機種を容易にサポートできるようになった」と主張してきましたが，これについては，多少，大本営発表的な面もあります．もちろん嘘ではありませんが，「容易に」とは言っても，まるっきり機種ごとに独立に実装した場合に比べて，の話に過ぎません．先に述べたような通信系ドライバに共通の問題に加えて，TCP接続の場合，通信路の不具合で接続が切れた後の自動復旧処理などにも注意が必要ですので，それはそれなりに大変な仕事でした．安定して使えるようになったのは，理研仁科センターの込山美咲さんがRIBFでの運用に向けて忍耐強く地道な評価試験を続けられた努力の賜物であると思います．

結局，工夫したから容易になったとはいえ，NetDevの開発期間は相当な長期に亘りました．その間に筆者を含めて開発に携わった人に支払われた給与の総額は相当なものになるはずです．

4.　μITRONへのEPICSの移植の試み

前節で述べたことを要約すると，IOCの下にPLCをぶら下げることの問題点は以下の三つです．

1）　ソフトの塊と化したPLCのラダー・プログラミングとその開発環境の学習コスト
2）　複数のコントローラにまたがったロジックのデバッグの難しさ
3）　IOCとPLC間の通信に必要なEPICSドライバの開発コスト

これらは全て，元を正せばIOCとPLCという二つのフロントエンド計算機を使うことに起因します．

近年，技術の進歩によりPLC用のCPUとして「RTOSモジュール」とか「C言語コントローラ」と呼ばれるOSを搭載したモジュールが市場で調達可能になりました．

これらのCPUを採用し，その上でEPICSを実行する，つまり，PLC自身をIOCとすることで上の三つの問題がすべて雲散霧消することは明らかです（図1）．

実際，当時すでに，フロントエンド・コントローラの上でEPICSを直接実行する「組込みEPICS技術」が国内外の複数の研究機関で検討され，また実際に開発，運用されてもいました[8]．その意味でアイデアとしての新奇性は乏しかったと言えるかもしれません．しかし，汎用性（豊富なI/Oモジュール）と高い信頼性を持ち，すでに加速器制御で多用されているPLCに目を付けた点においては，我々が初めてだっ

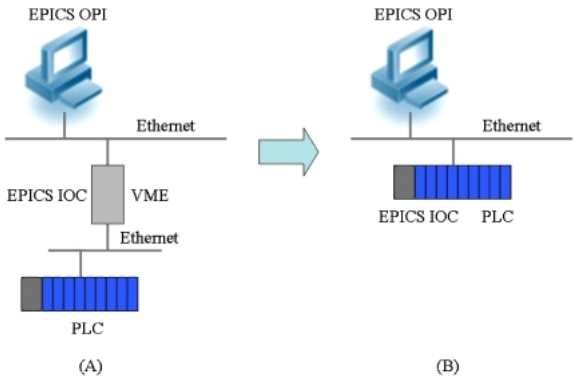

図1　PLCのIOC化

たのではないかと自負しています．

このようなCPUのOSの候補として，VxWorks, μITRONなど，いくつかの選択肢がありました．素直に考えればEPICSがその開発時からIOCのOSとして採用してきたVxWorksを選ぶことが順当であったと思います．しかし，当時の日本製のフロントエンド・コントローラの多くがμITRONを採用していたことを考慮し，敢えて困難を承知の上でμITRONを選択しました，というのはあくまで山本さんのお考えで，筆者は全く別の動機を持っていました．

当時，筆者はNHKで放映されていた「プロジェクトX」の影響を強く受けていました．毎週，中島みゆきの「地上の星」に心を震わせ，ラストでは涙を流していた中年男が，どうして外国製のOSなど選べましょう？　ここはひとつ，日本発の技術で世界に一発かましてやろう．そう思うのは極めて自然な成り行きでした（現在の筆者には，もうそんな元気はありません．とにかく楽な道を選びたい．今はそれだけしか考えていません）．

EPICSをμITRONに移植する仕事は，当時，総合研究大学院大学に在籍していたJiang Geyangさんの学位論文のテーマとなり，筆者はその技術指導を行いました[9]．しかし，実際に始めてみると，当初予想したよりもかなり困難な仕事であることが分かりました．

技術上の問題で最大のネックとなったのは組込み向けアプリケーション開発環境のC++のサポートの弱さでした．必要なライブラリがない．意味不明のエラー・メッセージがバラバラと出る．C++の“++”が「小さな親切，余計なお世話」に見える筆者には手も足もでませんでした．結局，山本さん，EPICS開発者の主要メンバの一人であるJeffrey O. Hillさんの助力を得て，何とかEPICSの移植を完了し，2005年のJiangさんの学位取得に結実しました[10]．

しかし，その過程では他にも色々と苦労がありました．まず，バージョン問題です．μITRONは基本的にはカーネルだけです．実際にEPICSを動作させるためにはBSD互換のソケット・ライブラリなどの他のソフトウェア・パッケージと組み合わせる必要があります．ところが，このバージョンのカーネルにはこのバージョンのコンパイラが必要だ．でもこちらのパッケージには，云々．バージョンの整合性の確保が大きな問題でした．さらに，それぞれのソフトウェアが頻繁にバージョンアップを行うという状況にあっては，「解なし」という状況を避け続けることが非常に難しいことが予想されました．

また，商用ソフトウェアを組み合わせる場合の著作権に関わる問題，ソフトウェアを提供するベンダーとの守秘義務契約による制約が非常に煩わしかったことも負担でした．副次的な効果として，筆者は著作権法にかなり明るくなりました．

本来であればJiangさんが行った仕事を筆者が引き継ぎ，実際の加速器制御に応用するはずでしたが，上述の理由により，他の仕事との両立は不可能であると判断せざるを得ませんでした．

5.　Linuxのメリット

μITRONを諦めた後，自然に考えればVxWorksに戻る，というのが採るべき選択肢であったかもしれません．しかし，結果的には，その道は選びませんでした．J-PARC主リング制御で全面的に採用されたLinuxでのプログラミングに馴染み，その楽さと効率に魅せられてしまったからです[11]．

リアルタイムOS上では，アプリケーション・プログラムはカーネルと一体化したモジュールとして動作するのが普通です．このため，EPICSのようなマルチタスクの複雑なプログラムは，その一部であっても修正を加えるとカーネルを含めてリブートせざるを得ません（そうでないものもあるかもしれません）．これに対して，Linuxの上でユーザ・プロセスとして走るEPICSの起動と終了はとても速い．“Ctrl＋C”キーでプロセスを殺し，上向き矢印キーで直ちに再スタート．文字通り一瞬です．そのため，ある程度まとめてプログラミングを済ませてからデバッグに移る，という必要はありません．ちょっと書き直しては再コンパイルして動作確認．これを頻繁に繰り返すことができます．いわゆる「パターで刻むプログラミング」です．急がば回れ．安全速度が一番早い．千里の道も一歩から．塵も積もれば山となる．点滴岩をも穿つ．Slow but steady wins the race. これらの格言はプログラミングの世界においてこそ，極めつけの真理です．

Linuxを使うもうひとつの利点は，開発環境を気軽に個人のラップトップPC上にインストールできる，という点です．とりあえず労働問題的な側面を無視すれば自宅でもプログラミングができることが何よりの魅力です．土日はもとより，お盆，正月，いつでもどこでもOK．ベッドの上でもトイレの中でも．（さすがに風呂では止めた方が良いでしょう．もちろん離着陸段階にある旅客機の中では絶対に不可です．）

もちろん筆者にこのような働き方を推奨する意図はありません．また，実際にそのような働き方をしたことを意味するものでもありません．ただ，新しい加速器の建設段階において，制御ソフトウェア開発の進捗

が危機的な状況にある場合には，あくまで緊急避難として「どこでもプログラミング（風呂と離着陸段階にある旅客機を除く）」が可能であることを指摘しているにすぎません．

ともかく，ひとたび Linux 上の EPICS 開発環境の便利さ，心地よさに慣れてしまうと，それを捨てる気にはどうしてもなれませんでした．

6. EPICS on F3RP61 の誕生

6.1 ことの発端

これまでの話を要約すると，次のようになります．IOC と PLC をイーサネットで繋いで使うのは止めて，PLC 自体を IOC にしたい．しかし，一方で Linux の快適さと利便性には捨てがたいものがある．できれば Linux が OS として走る PLC 用の CPU が欲しい．しかし，そんな都合の良いものはこの世にない．なければ作ればよい．でも自分では作れない．誰か作ってくれないか？

駄目でもともと，の発想で，深い考えもなく 2007 年の加速器学会で横河電機株式会社の企業展示ブースに当時同社にご勤務されていた中川博記さんを訪ねました．以前から，中川さんが EPICS の μITRON 上への移植の話に関心を示されていたからです．

何方かとお話しされていたところに半ば割り込むようにして話し掛け，要望をお伝えすると，

いや，実はね，今まさにその話をしていたところ

との答．「えらく調子のいい奴だな．たんに話を合わせてるだけじゃないの？」というのが正直な印象でした．ともかく可能性を追求して下さるというだけでも有難いお話でしたが，その時点では，その半年後に本当に評価機が手元に届くとは思いもよりませんでした．

6.2 EPICS 起動！

時は 2008 年 2 月．J-PARC 主リングの Day-1 まで残すところあと 3 カ月．望むものは目の前にあるのにとても自分で評価試験をする時間はない．いつも頼りにする先輩の秋山篤美さんとて状況は同じ．一方，ベータ版テスターからのフィードバックなしに横河電機で製品化が進むはずはない．何としてでも速やかに評価をしなければならない．さて，どうしたものか．

その時，頭に浮かんだのは理研仁科センターのリニアックでオペレータをされている，住重加速器サービスの内山暁仁さん．彼は日経 Linux の記事を頼りにボード PC の上に組込み Linux を構築した経験がある[12]．EPICS も知っている．彼しかいない．でも，彼だって暇じゃない．忙しくて自分じゃできないから代わりにやって．そんな話がまかり通るくらいなら消費者金融など必要ない．

再び，駄目でもともと，の発想で評価機を携え，理研仁科センターを訪れ，込山さんと内山さんに相談を持ち掛けたところ，予想に反してご快諾を頂けました（この点については「迷惑だったけれど断り切れなかった」という事実を筆者が誤認している可能性があります）．

しかし，その後，筆者が担当するはずであった EPICS のビルドは進みませんでした．迫りくる J-PARC 主リングの Day-1 に向け，大脳負荷 100% の状態で，バグ作りとバグ潰しを繰り返す日々を送っていたからです．その時，突然，中川博記さんからメールが届きました．評価機への Linux カーネルの移植をされた横河電機の亀田喜一さんが EPICS のビルドに成功された，とのこと．EPICS をビルドする Make ファイルはかなり複雑なので，EPICS に経験をお持ちでない方があっさりとビルドされたことに驚きました．しかし，よく考えてみれば，Linux カーネルのビルドに比べれば EPICS のビルドなど，たやすいことだったのでしょう．「これでいける」と確信し，亀田さんから送って頂いた Make ファイルを内山さんに転送したところ，ほどなく内山さんからのメールが届きました．「ひょっとして…」と期待しつつ，メールを開くと

EPICS が動きました

とあります．おまけに IOC プログラムを起動した際のスクリーンショットまで添付されていました（**図 2**）．

ふいに熱いものが込み上げ…

筆者は…　泣いた

（この時，筆者の胸の中で中島みゆきの「地上の星」が反響していたことは言うまでもありません．）

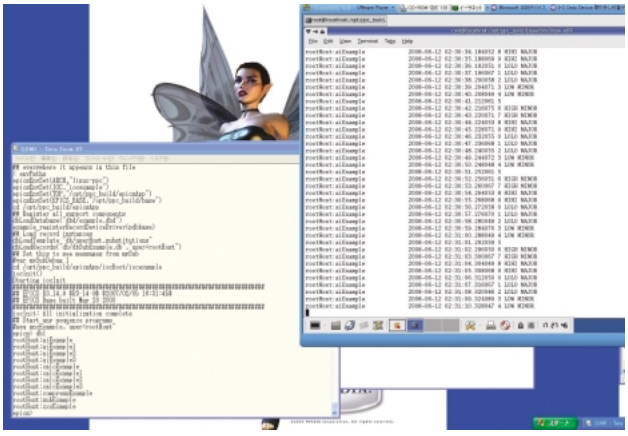

図 2　EPICS on F3RP61 誕生の瞬間

その後，可及的速やかに理研仁科センターに足を運びました．目的はI/Oモジュールにアクセスするための EPICS ドライバの実装です．組込み EPICS が第4節の冒頭に述べた従来方式（IOC のの下に PLC をぶら下げるやり方）の問題点の中，以下の二つを解決することは自明です．

1） ソフトの塊と化した PLC のラダー・プログラミングとその開発環境の学習コスト
2） 複数のコントローラにまたがったロジックのデバッグの難しさ

また，三つめの問題，

3） IOC と PLC 間の通信に必要な EPICS ドライバの開発コスト

については，IOC がイーサネット越しに PLC と通信するための EPICS ドライバに比べて，IOC（＝PLC）の CPU がバス上の I/O モジュールにアクセスするための EPICS ドライバがはるかに簡単になることによって解決されます．PLC バス上での I/O は瞬時に終わるため，コールバックの仕組みは不要だからです．単に CPU モジュールに付属する Linux ドライバが提供するインタフェースをラップするだけで EPICS ドライバを作ることができます．文字通り「どなたでもご自宅で余暇を利用してできる簡単なお仕事」です．そうは言っても，やはり論より証拠．実証するためには客観的なデータが必要です．そこで，一番簡単な EPICS のレコード（Binary Output）からデジタル出力モジュールに出力するための EPICS ドライバを作成し，実装時間を計測することにしました．込山さんがタイムキーパーの役を買って出てくれました．スタートの合図とともに実装開始．モジュール上の LED が点灯した時点で時計をストップ．計測結果はなんと

1時間と56分

でした（「なんだよ，それくらいのこと，私なら30分でやるけどね」と思われた読者がいらっしゃったとすれば，筆者はあっさり同意します．その日はちょっと体調が悪かったのです）．ともかく，従来方式が抱える問題点のすべてが解決されることが見事に証明されました[13]．

6.3 EPICS シーケンサの試験

I/O モジュールへのアクセスに成功した後，直ちに EPICS シーケンサの動作確認を行いました．組込み EPICS では，従来型 CPU の上で走るラダー・プログラムを EPICS シーケンサで置き換えることになるため，これが安定に動作することを確認することは極めて重要だからです．EPICS シーケンサは State Notation Language（SNL）と呼ばれる C 言語に似た独自言語で記述されます[14]．その特徴は，その名が示す通り state（状態）という概念に基づくことです．また，機器制御には欠かせないイベント待ちを容易に記述する構文を持つことも特記すべき点です．これらの特徴により，EPICS シーケンサを使うことによって，可読性の高いシーケンス制御プログラムを書くことができます．

安定性を確認するため，昼夜を問わない試験を継続しました．（就寝前にプログラムを起動し，翌朝，動いていることを確認しただけです．）その結果，実際の加速器制御システムで試験運用を始めるに足る安定性を有することを確信することができました．ひとたび枯れてしまえば，開発コストに終わりの見えるドライバと異なり，アプリケーションの開発には終わりがありません．その意味において，EPICSシーケンサを使うことによる開発効率の向上は極めて重要なのです．

6.4 コードネーム，「ガンダム」

これは使える，との感触は得たものの，さすがに製品化が決まっていないものを実際の加速器制御システムに採用できるはずなどありません．

この時，すでに J-PARC 主リングの Day-1 は過ぎていましたが，次にはビーム取り出しラインの立ち上げが迫っていました．なんとかそこで使いたい．というより，使わなければ間に合わない．ともかく製品化をプッシュしたい．しかし相手は営利企業．市場見通しのないままで製品化などできるはずがない．さりとて製品化されていない評価機など売り込めない．鶏が先か卵が先か．無限ループじゃ埒が明かない．今度ばかりは，駄目でもともと，では済まないことを承知の上で，中川博記さんの

なんとかします

の一言を信じ，「絶対に大丈夫です」と太鼓判を押して各方面に売り込むことにしました．

しかし，ここで問題がありました．製品化後の製品名は知らなかったことです．名無しの権兵衛を売り込むわけにもいかないので，勝手に「ガンダム」と呼ぶことにしました（べつに「アトム」でも「やまと」でも何でも良かったのですが，一番強そうなものを選びました）．

ともかく，2008年8月26日，晴れて F3RP61 の製品リリースのアナウンスがあったときはときには，本当に，ほっとしました[15]（「あの時，もし横河さんに納期倒しをされていたらコンクリ詰めにされて東海の海に沈められていたかもしれない」と思うと，今で

図3　第5回日本加速器学会年会・第33回リニアック技術研究会（広島）における関係者の集合写真

Linux kernel 2.6.24.3

Model Name : **F3RP61**

32bit 533MHz PowerPC

128MB RAM

64MB Flash ROM

図4　F3RP61（LinuxをOSとして採用したPLC CPU）

もぞっとします）．

以上のような経緯で，KEKではいまだにF3RP61（実はe-RT3 2.0という別名もある）よりも「ガンダム」の方が通りの良い名として使われています（筆者はいまだにKEKの職員から，「ガンダムを扱っている代理店の連絡先を教えて下さい」というメールを頂くことがあります）．

図3に，筆者が同僚から

お前，横河に名刺作って貰えよ

と嫌味を言われていた時期に開かれた加速器学会での関係者の集合写真を，**図4**にF3RP61（旧姓「ガンダム」，別名e-RT3 2.0）の写真を示します．

7.　加速器制御への応用

市場での調達が可能となれば，当然，実システムでの評価に進むことになります[16]．万一，滑った場合にも後戻りできるような適当な応用例を求めてKEKB制御グループの古川和朗さんに相談に伺いました．すると，うまい具合にF3RP61で改善を期待できる事例が二つあることが分かりました．

ひとつはパルスQ電磁石電源の制御で，もうひとつは可動ビームマスクの位置決め制御でした．

前者については，電源制御のPLCのラダーを筆者自身が実装した経緯があるため，願ったり，叶ったり，のお話です．後者についても，筆者自身がJ-PARCの遅い取り出しビームラインの制御で，セプタム電磁石群の位置決め制御にF3RP61を使うことを検討していたので，うまくすれば，経験を共有することができます．

パルスQ電磁石電源制御については，三川勝彦さんが中心となり，三増俊弘さんと筆者が協力する，という形で行いました[17,18]．出力電圧を，ある閾値より上か下か，また，上げるか，下げるか，でランピング方法を変えるロジックが必要とされたため，EPICSシーケンサの可読性の高さを実証する良い事例となりました．

可動ビームマスク（**図5**）の位置決め制御については，三菱電機システムサービスの中村卓也さんが担当

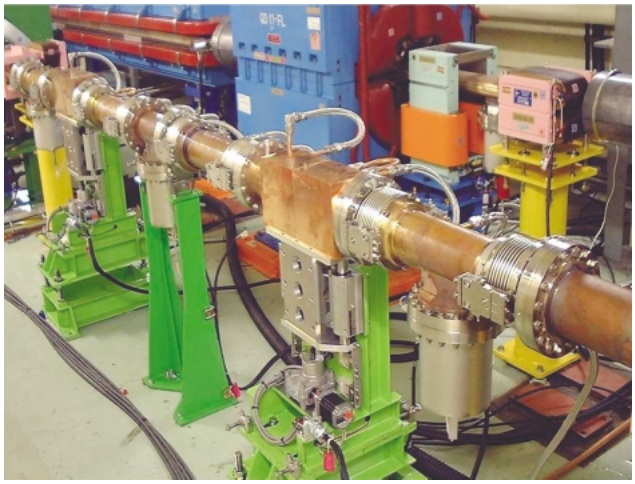

図 5　KEKB 可動ビームマスク

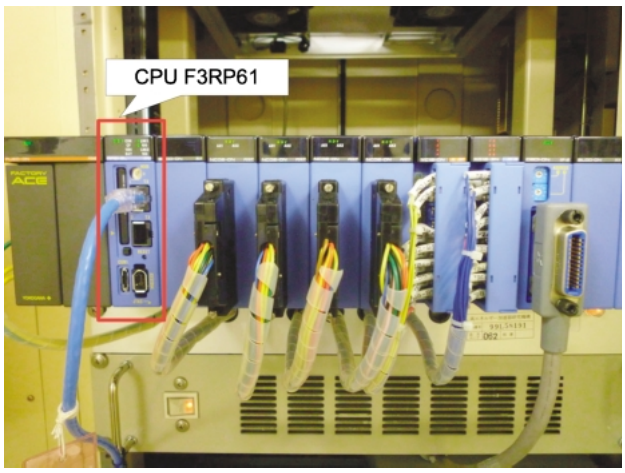

図 7　ビームマスク位置決め制御用 IOC

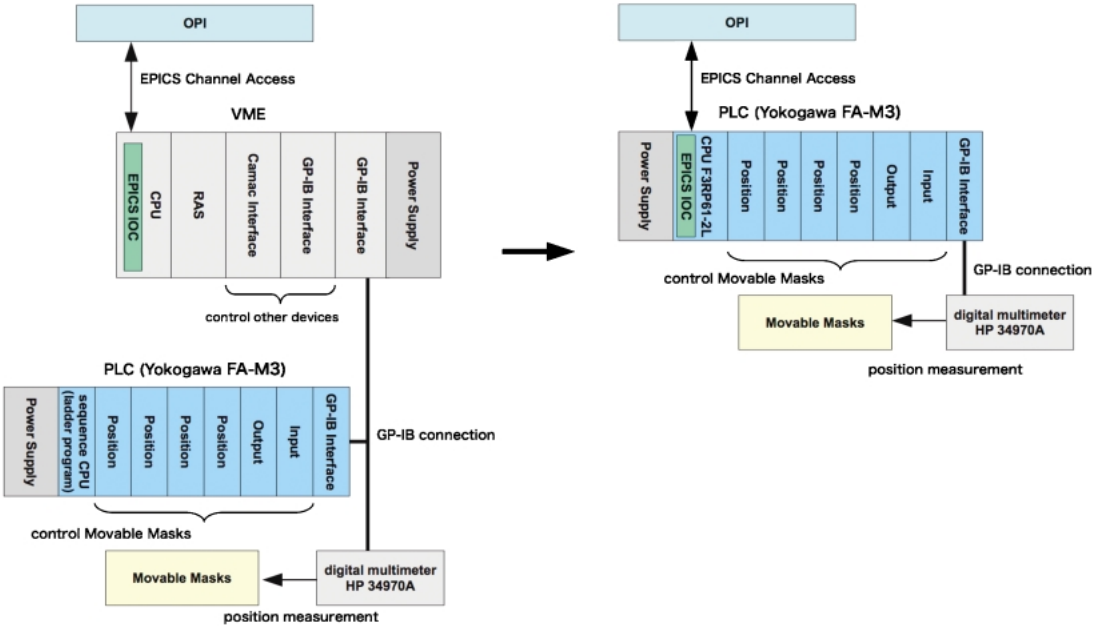

図 6　可動ビームマスク制御システムの簡素化

されました[19,20]．それまで使われてきたシステムでは，VME / VxWorks ベースの IOC と PLC が GPIB で接続されていました．位置決めモジュールを制御するシーケンスはラダーで実装され，上位の VME 計算機（IOC）からはパラメータの設定と読み返しを行っていました．IOC と PLC との間の GPIB 通信の信頼性に問題があったため，PLC ユニット上の F3RP61 を IOC とする方式に改めました．これにより，位置決めモジュールの制御シーケンスを実装していたラダーは EPICS シーケンサに置き換えられました．この応用事例は，システムをすっきりさせることにより信頼性と保守性を向上させた好事例であると思います（図 6，図 7）．また，多軸の制御をたった一つのテンプレートとなるプログラムと，軸ごとのパラメータを記述するパラメータ・ファイルの組合せにより実現したこともソフトウェアの保守性の向上に寄与しています．

KEKB 制御システムでの動作実績により，J-PARC ビーム取り出しラインの制御に自信を持って応用することができました．主な用途は大別して，電源制御，位置決め制御，データ収集，及びインタロック状態のモニタです．J-PARC 主リングからの遅い取り出しビームラインでは，F3RP61 が全面的に採用されました（図 8）．図 9 は KEK の村杉茂さんの手によるメイン操作パネルです．

図8　J-PARC遅い取り出しビームライン（中央部は低磁場セプタム電磁石）

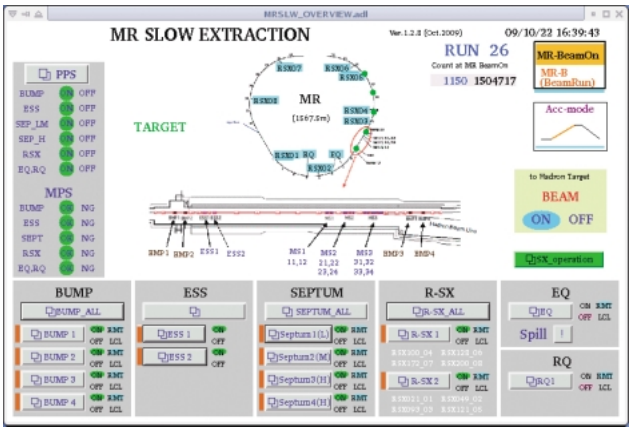

図9　遅い取り出し制御メイン・パネル

遅い取り出しビームラインの電源制御については，静電セプタム電源，低磁場セプタム電磁石電源，中磁場セプタム電磁石電源，高磁場セプタム電磁石電源のすべての制御を関東情報サービスの高木誠さんが一人で担当されました[21,22]（但し，後述するインタロックは除きます）．短期間にこれだけの制御ソフトウェアを一人で完成させることができたのは，高木さんのスキルとEPICS on F3RP61，特にEPICSシーケンサの開発効率の高さの相乗効果です．まさに「鬼に金棒，ツインカムにターボ」です．

位置決め制御に関しては筆者が担当しましたが，既述のKEKBの可動ビームマスクの位置決め制御と内容が被るため，割愛します．

筆者が個人的に最も面白いと思うF3RP61の応用例は，ビームモニタからのデータ収集システムです[23,24]．これについては，関東情報サービスの本橋重信さんに担当して頂きました．もちろんデータ収集といっても，オシロスコープで観測するような生信号ではなく，然るべき処理回路で積分された比較的遅い信号のモニタの話です．J-PARC主リングでは，二種のビームモニタのデータ収集システムにF3RP61が利用されています．一つはビームロスモニタで，もうひとつはFast Current Transformer（FCT）です．主リング全体には300本を超えるガス・チェンバー型ビームロスモニタが配置されています（図10）．ビームロスモニタからの出力信号の積分値を読み出すデータ収集システムは8台のIOC（＝PLC）ユニット（図11）から成り，1ユニットは最大48チャンネルまで扱えます．ビームロスの空間分布のみならず，マシンサイクル中の時間構造も計測する必要があるため，マシンサイクルごとに定周期スキャンを繰り返します．定時性を要求されるプログラミングはC言語で記述されるEPICSドライバで実装されました．

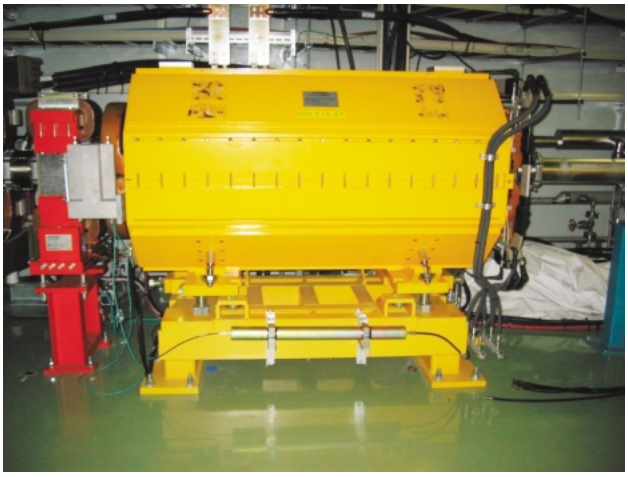

図10　ビームロスモニタ（電磁石架台側面）

FCTの処理回路からの出力を読み出すデータ収集システムは，1チャンネルのみの信号を扱う小規模なシステムです．もちろん，ビームに同期してデータを読む必要があるため，タイミング・システムからの同期信号によって起動されるイベント駆動型のシステムです．同期信号を受け付け，F3RP61に割り込みを掛けるためにFA-M3 PLCの高速デジタル入力モジュールが使用されています（上述のビームロスモニタにおいても，スキャン開始のトリガ信号の入力のために同じ仕組みが使われています）．

インタロック状態のモニタについては，もっぱらKEKの村杉茂さんと日本アドバンストテクノロジーの名倉信明さんが担当されました．インタロックのロジック自体は従来型CPU上で実行されるラダー・プ

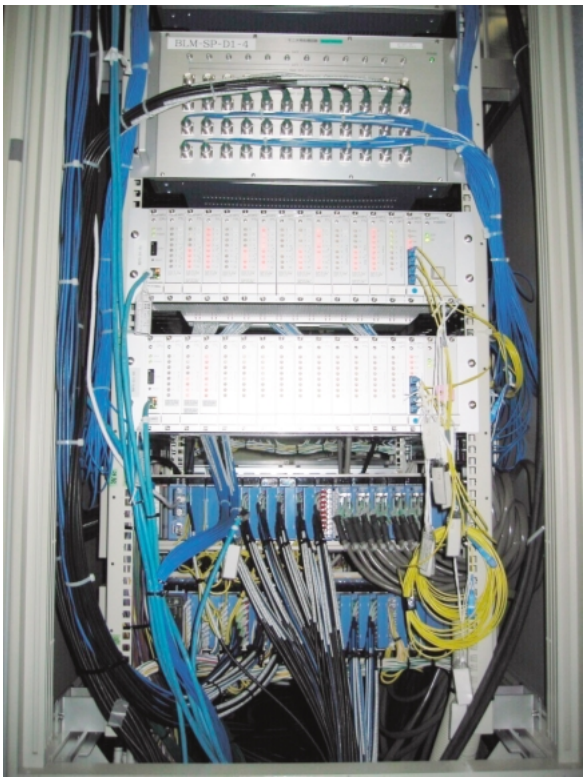

図11　ビームロスモニタ用データ収集システムの処理回路（上段）とIOC（下段）

ログラムで実装し，同一ユニット上のF3RP61が共有メモリを介して従来型CPUからインタロック状態を読み出す，というマルチCPU構成のシステムです．このような構成を採ったのは，もちろん，インタロック・システムには特に高い信頼性が求められるからです．

これまでの実績から見て，筆者個人としては，F3RP61自体は従来型CPUと同等の信頼性を有するのではないか？という印象を持っています．しかし，アプリケーション・プログラムのレベルでは，マルチスレッドを使うEPICSにラダー・プログラムと同等の信頼性を持たせることは困難であろうと考えています．

その理由をご説明するため，ラダー・プログラムをC言語風に記述してみましょう．細かい話は抜きにして，以下のような感じになるでしょう．

```
for (; ;)
  {
    if (a) do_a( );
    if (b) do_b( );
    if (c) do_c( );
    if (d) do_d( );
    if (e) do_e( );
    if (f) do_f( );
        …
  }
```

一見して分かるように，この一つのスレッド（処理の流れ）から成るプログラムには試験のカバレッジ（取り得るコード・パス全体に占める試験済みのコード・パスの割合）が上げ易い，という特徴があります．つまり信頼性が上げ易い，ということです．

その代償は，デバイス・ドライバのようなプログラムにつきものの，複数の外部イベントにより駆動される非同期的な処理を記述すると訳が分からなくなる，という点です．もちろん，やってできないことではありません．実際，マルチタスクOSを使ったアプリケーションであっても，システム関数の先にあるカーネルまで降ってカーネル内のコード・パスを通り，再びアプリケーションに戻るまでの道筋は一本に繋がることに気付きます．マルチタスクは，あくまでもカーネルがユーザに見せている幻想にすぎません．

要するに何が言いたいかといえば，カーネルのコードを頭の中で瞬時に構築できるほどの超人的な頭脳を持っていれば，ラダー・プログラムであってもマルチタスクOSを使ったアプリケーション並みの複雑な処理が記述できる．ところが，現実にはそんな人はいない．だから単純で信頼性のあるプログラムしか書けない．それがラダー・プログラムで実装されたロジックに信頼性がある理由です．

複数の外部イベントにより駆動される複雑な処理であれば，それぞれのイベント毎にタスク（スレッド）に分けた方が，人間にとって圧倒的に分かり易くなります．しかし，その代償として，試験のカバレッジを上げることが難しくなります．頭の中でたった二つのタスクAとタスクBを用意し，コードの仮想試験をしてみましょう．まずタスクAを1ステップ動かしてからタスクBを1ステップ動かした場合．タスクAを2ステップ動かしてからタスクBを1ステップ動かした場合　…　はい，読者の皆様に十分にご納得いただくためには，本学会誌の何年分かの紙面が必要になります（もちろん，「自分は超人的な頭脳を持っ

ている」と勘違いして等価な処理をラダー・プログラムで実装したとすれば，デバッグはさらに困難になるでしょう）．

要は，実装できるロジックは単純なものに限られるが信頼性を上げたい場合はラダー・プログラムを，信頼性を犠牲にしてでも複雑な処理を分かり易く記述したい場合にはマルチタスクOS上で動作するC言語のプログラムを使う，というのが適切な使い分けになるということです．そこで，F3RP61と従来型のラダーCPUを組み合わせれば「一粒で二度おいしい」システムを組むことができる，という結論に達します．

F3RP61の加速器への応用という本題から話がだいぶ外れてしまいました．以上ではKEKにおける応用例について述べましたが，本技術を共同で開発された理研仁科センターにおいても，F3RP61がRIBFの新イオン源制御のためのIOCとして実用に供され，ソフトウェア開発の効率性とF3RP61の安定性について高い評価が得られていると伺っています[25]．

8. リアルタイム拡張

ここまで忍耐強くお読み下さった読者の中には，「Linuxだとリアルタイム応答性が問題になるよね？まさか，都合の悪い話については，すっとぼけるつもりじゃないよね」と思われた方も多いのではないかと思います．もちろん，違います．リアルタイム応答性の問題についても見通しはありました．F3RP61のBSPに含まれるLinuxカーネルはCONFIG_PREEMPTというオプション（これはカーネル2.6の標準的な機能です）を立ててビルドされているため，数ミリ秒のレベルのリアルタイム応答性が期待できるからです．

このような機能のなかった一昔前のLinuxカーネルはリアルタイム応答性が悪いことで有名でした．その原因は，カーネルモードで実行中のプロセスから実行を横取り（preemption）することができなかったためです．

例えば，トイレが一つしかない家でお父さんが入ってしまったとします．お父さんが怖くて，家族は誰も急かすことなどできません．ひたすら，お父さんが用を済ませて出てくるのを待つしかありません．これでは，場合によって，とても具合の悪い結果をもたらします（「お父さん」がプロセス（厳密にはスレッド）で，「トイレ」がカーネルです．もちろん，この家族は皆がお互いに恐れ合っている，という多少，無理のある設定です）．

しかし，バージョン2.6からは状況が大きく変わりました．トイレのドアをノックできるようになったのです．もちろん，ノックされたからといって，いつでも直ちに交代できるとは限りません．どうしても「ちょっと待って」と言わざるを得ない局面もあるでしょう．また，突然の故障でトイレ修理の専門家（割込み処理）に占有されて何人たりともトイレを使えない場合も生じます．そうであったとしても，「ノック禁止」の場合に比べれば，格段にリアルタイム応答性が向上することは明らかです．

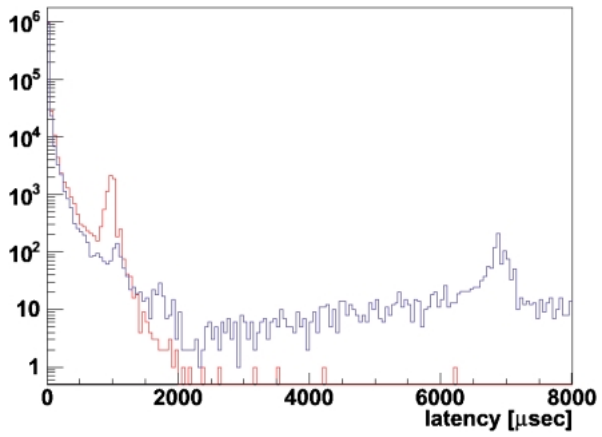

図12 リアルタイム応答性の比較（青：標準カーネル，赤：CONFIG_PREEMPT_DESKTOP適用カーネル）

EPICS on F3RP61の有用性と安定性に評価が確立したころ，KEKの山田秀衛さんから，技術的に大変興味深い話を伺いました．標準カーネル2.6のリアルタイム応答性を，さらに，しかも格段に向上させるパッチがある，とのことでした．実際，山田さんが，CONFIG_PREEMPT_DESKTOPというオプションを有効にしたF3RP61のLinuxカーネルをビルドし，リアルタイム応答性を評価する予備的な試験をされました．その結果を見せて頂きましたが，それは，そのパッチが相当な効果をもたらすことを一目瞭然に示すものでした（**図12**）．

早速，検索サイトで情報を集めたところ，CONFIG_PREEMPT_DESKTOPの上のレベルのオプション，CONFIG_PREEMPT_RTを適用できれば，信じ難いようなリアルタイム応答性が得られることが分かりました[26]．

そこで，横河電機の白津英仁さんにメールを送り，リアルタイム拡張について相談させて下さい，とお願いしました．筆者は，リリースされても間もない製品の機能拡張に関わる要望だったので，当然，否定的な回答（「お客さん，そういった話は1000台買ってか

表 1　定周期実行の遅延時間

レイテンシ範囲	現製品	PREEMPT_RT
	2 ms 6 万回	1 ms 4320 万回
0～	0	1
10～	0	37,382
20～	23,617	7,559,622
30～	26,609	31,478,761
40～	3,341	1,924,410
50～	2,928	1,925,191
60～	1,476	268,123
70～	1,088	5,958
80～	507	546
90～	256	6
100～	84	
110～	28	
120 以上	66	
レイテンシ最大値	1,065 μs	92 μs

らにして下さいよ」）が寄せられるものと予想していました．ところが，驚いたことに，頂いた返信には，なんと

CONFIG_PREEMPT_RT を評価中です

とありました．もうこうなると地上の星どころではありません．「2001 年宇宙の旅」です．筆者の胸中は興奮のるつぼと化しました（「2001 年じゃ過去の話じゃないか」などと野暮な突っ込みを入れる読者がいらっしゃらないことを祈ります）．

果たして，その後，白津さんを通して亀田さんから伺ったリアルタイム応答性に関するプレリミナリーな結果は，検索サイトで得た情報と整合するものでした．その後，横河電機社が公表したデータを同社の許諾を得て**表 1** に掲載します．

表中に示された数値は，hackbench を負荷として実行した状態で，リアルタイム・プロセスの定周期実行の遅延を測定した結果です．一昔前に Linux のリアルタイム拡張に取組み，挫折した経験を持つ筆者から見ると，ちょっと信じがたいような結果です．早く自分の目で見てみたいという思いに駆られました．

願いは 2009 年 10 月の加速器制御の国際会議（ICALEPCS 2009）で叶えられました．実は，横河電機社の企業展示ブースで行ったリアルタイム応答性に関わるデモの内容は，筆者から同社に提案し，筆者

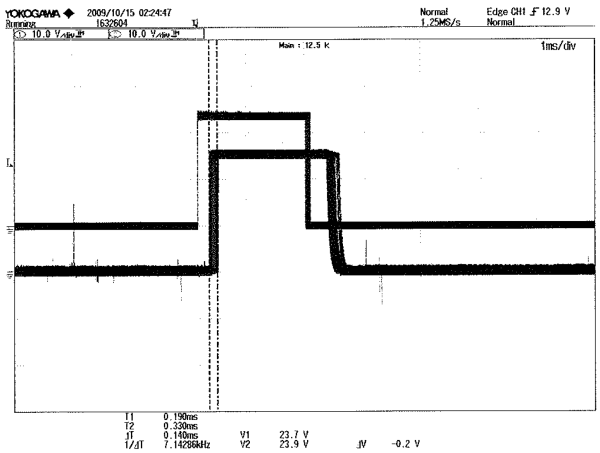

図13　I/O モジュールを使用したジッター測定（上段：入力パルス，下段：出力パルス）

自らが実装したものでした．当然，中身は承知しています．仕掛けは極めて単純です．高速デジタル入力モジュールからパルスを入力し，F3RP61 に割込みを掛ける．すると割込待ちで休眠していたスレッドが起床して直ちにデジタル出力モジュールに出力する．出力パルスのジッターをオシロスコープで観測すればリアルタイム応答性は一目瞭然です．

会議中，同社の企業展示ブースを何度も訪れ，自分の目で確かめました．I/O モジュールの応答までを含むジッターは，確かに 100 マイクロ秒程度に収まっていました．さすがにファイルシステムへのアクセスを生ずる重い CPU 負荷をかけると，ジッターは 150 マイクロ秒ほどに増えましたが，そこで踏みとどまります（**図 13**）．会議期間中，何度も同じ試験を繰り返しました．自分の目でオシロスコープの画面を凝視し続ける中に

これは本物だ

と確信するに至りました．

9.　今後の展開

皆様もご存じのとおり，近年の KEK の高齢化には目覚ましいものがあります．特に KEKB の制御グループにおいては，間もなく 50 の大台に乗る筆者が「最も若い人」です．ですから KEKB 制御グループは，あと 10 年もすれば自然消滅する

絶滅危惧種

です．今後は，ソフトウェア開発においても外注化を進めざるを得ません．これまでは EPICS によるフロントエンド制御に VME/VxWorks を必要としていたため，企業がテストベンチを導入するためのコストがネックとなっていました．しかし，F3RP61/Linux

であれば，導入コストが大幅に軽減されます．また，5節で述べたLinuxの効用とEPICSドライバの簡素化も大変有利に働きます．

このような観点から，現在，SuperKEKBに向けた新しいLLRFシステムのEPICS制御を三菱電機特機システムとともに開発しています．一部のインタロックを除けば完全にEPICSベースのシステムであり，LinuxシステムからEPICSに至るまで同社のエンジニアの出口久城さんに実装をお願いすることになりました．F3RP61から組込みEPICSにのめり込み，いつの間にか話が拡がり，μTCA筐体に挿入される各ボードのIOC化にまで至りました．つまり，μTCAの筐体が一つのIOC-complexになるわけで，大変に画期的なシステムです．今後の進展が本当に楽しみです．

もうひとつの目標は，なんとかEPICS on F3RP61を世界に広めたい，ということです．日本製品を世界に売り込みたいというエコノミック・アニマル的な発想ではありません．これは本当に便利だから是非とも世界中の過重労働に苦しむ同業者に使って欲しい，という純然たる善意からです．

そこで，まずは時差のない東アジアの研究機関にEPICS on F3RP61を紹介しています[27]．余談となりますが，筆者は以前，中国のどこかの研究所の知人に電子メールを送った際，1時間ほどで返信が来て，びっくりしたことがあります．英文で送ったメールに英文で返信が来るのは，どんなに早くても翌日の朝，という反射的な思い込みがあったのでしょう．「ああ，時差がないから当たり前なんだ」と気付いたとき，就業時間帯を共有することの便益を実感しました．電子メールによるコミュニケーションであれば国内と何ら変わりません．新たに導入しようとする研究機関にメールでのサポートができます．東アジアでEPICS on F3RP61のファンが増える．やがてその中から欧米に行く人も出てくる．そこでまたファンが増え，やがて世界に広まる．そのような夢を膨らませると，わくわくします．

10. 結　　論

EPICS on F3RP61の開発により，EPICSベースの制御システムにおけるソフトウェア開発の効率が大幅に向上しました．ソフトウェア開発費のほとんどが人件費であること，日本の人件費が高いこと，の二点を考慮すれば，同技術が納税者負担の軽減に大いに寄与することは明らかです．実際，ソフトウェア開発費が制御システム構築費用のかなりの割合を占めています．

もちろん，加速器建設プロジェクト全体でみれば制御に関わる予算の割合そのものが小さいから，ソフトウェア開発の効率化による人件費の削減効果など大した意味はない，という反論もあろうかと思います．しかし，ひとたび買ってしまえば耐用年数を黙って働くハードウェアと異なり，人件費は毎年発生します．「去年は満額きちっと払ったんだから，今年は無給で1年頑張ってみてよ」などと言われて納得する労働者がいるでしょうか？

実を言えば，人件費は二義的な問題です．時は金なり，金は時なり．ソフトウェア開発において人件費を削減できるということは開発時間の短縮を意味します．短時間でのソフトウェア開発．これこそがポイントです．

士農工商犬猫制御

と言われるように何事においても制御には最低優先度が与えられ続け，立ち上げ間際になってデルタ関数的に仕事が降ってくる．この場面でこそ，ソフトウェア開発を加速するEPICS on F3RP61の真価が発揮されます．どう考えても間に合いそうもない案件を間に合わせる．それが本技術開発の本当の意義なのです．加速器が，その生涯に亘って常に性能向上のための改良が続く装置であることを考えれば，ソフトウェア開発の効率化がもたらす恩恵は一時的なものに止まりません．

大阪大学経済研究所の小野善康教授は，不況下での生産効率の改善は失業率の上昇を通じて消費を冷やし，さらに不況を悪化させることを指摘しました[28]．しかし，同教授は同時に，労働の社会的コストが低い不況下にこそ技術開発に投資し，魅力ある商品の開発を通じて消費を刺激すべきである，とも指摘しています．EPICS on F3RP61の開発が日本経済にとってプラスと働くか，マイナスと働くか？　筆者はそれを注視し続けてゆきたいと考えております．

謝辞

本EPICS on F3RP61プロジェクトに対し，一貫したご支援を頂きました，横河電機株式会社の夏井敏樹様に感謝いたします．同氏のご支援なくして本プロジェクトの成功はあり得ませんでした．折に触れてさり気なく同氏から頂いた激励のメッセージに何度も勇気づけられました．

また，製品化後間もなく実績の乏しかった時点において，EPICS on F3RP61の採用をご決断して下さった，高エネルギー加速器研究機構の冨澤正人助教，古

川和朗准教授，小林仁加速器研究施設第一研究系主幹，生出勝宣加速器研究施設長（当時，加速器研究施設第二研究系主幹）に深く感謝いたします．

参考文献

1) 中村達郎，「EPICS 超入門」(http://www-linac.kek.jp/jisshu/EPICSintro.pdf)
2) EPICS: Input/Output Controller Application Developer's Guide (http://icsweb.sns.ornl.gov/kasemir/train_2006/AppDevGuide_3_14_8.pdf)
3) EDM Extensible Display Manager for EPICS (http://www.aps.anl.gov/epics/docs/USPAS2007/lectures/EDM.odp)
4) The EPICS Channel Archiver (http//:www.aps.anl.gov/.../epics gettingstarted/.../The_EPICS_Channel_Archiver.ppt)
5) EPICS R3.14 Channel Access Reference Manual (http://www.aps.anl.gov/epics/base/R3-14/8-docs/CAref.html)
6) 民法第三篇第二章第一節第一款.
7) J. Odagiri, et al., "EPICS Device/Driver Support Modules for Network-based Devices", Proc. of the 3rd Annual Meeting of Particle Accelerator Society of Japan and the 31st Linear Accelerator meeting in Japan, Sendai, Aug. 2–4, 2006.
8) M. Komiyama, et al., "Control System for the RIKEN Accelerator Research Facility and RI-Beam Factory", Proc. of CYCLOTRON 2004, Tokyo, Japan, Oct. 18–22, 2004.
9) Jiang Geyang, "Porting EPICS Core Program onto micro-ITRON/SH4-based Device Controllers", Proc. of International Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC05), Hayama, Japan, March 22–25, 2005.
10) Jiang Geyang, "Development of Embedded EPICS and its Application to Accelerator Control System" (総合研究大学院大学高エネルギー加速器科学研究科後期博士課程　博士論文).
11) G. Shen, et al., "Development of Linux-based IOC with a VME-bus Computer", Proc. of International Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC05), Hayama, Japan, March 22–25, 2005.
12) A. Uchiyama, et al., "Development of Embedded System for Running EPICS IOC by Using Linux and Single Board Computer", Proc. of International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS07), Knoxville, Tennessee, USA, Oct. 15–19, 2007.
13) A. Uchiyama, et al., "Development of Embedded EPICS on F3RP61–2L", Proc. of International Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC08), Ljubljana, Slovenia, Oct. 20–23, 2008.
14) State Notation Language and Sequencer Users' Guide (http://www.slac.stanford.edu/comp/unix/package/epics/sequencer/manual/Manual.pdf)
15) http://www.yokogawa.co.jp/rtos/Products/rtos-prdcpu9-ja.htm
16) J. Odagiri, "Application of EPICS on F3RP61 to Accelerator Control", Proc. of the 6th Annual Meeting of Particle Accelerator Society of Japan, Tokai, Aug 5–7.
17) T. Mimashi, et al., "The Pulsed Quadrupole Magnet for KEKB Low Energy Ring", Proc. of European Particle Accelerator Conference (EPAC08), Genoa, Italy, June 23–27, 2008.
18) M. Mikawa, et al., "Embedded EPICS Controller for KEKB Pulsed Quadrupole Magnet Power Supply", Proc. of International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS09), Kobe, Japan, Oct. 12–16, 2009.
19) Y. Suetsugu, et al., "Experimental Demonstration of Movable Mask with Low Beam Impedance", Proc. of the 5th Annual Meeting of Particle Accelerator Society of Japan and the 33rd Linear Accelerator meeting in Japan, Hiroshima, Japan, Aug. 6–8, 2008.
20) T. Nakamura, et al., "Upgrading the Control System of the Movable Masks for KEKB", Proc. of the 6th Annual Meeting of Particle Accelerator Society of Japan, Tokai, Aug 5–7.
21) M. Tomizawa, et al., "Slow Extraction from J–PARC Main Ring", Proc. of the 6th Annual Meeting of Particle Accelerator Society of Japan, Tokai, Aug 5–7.
22) M.Takagi, et al., "Control of the J–PARC Slow Extraction Line Based on Embedded EPICS", Proc. of the 6th Annual Meeting of Particle Accelerator Society of Japan, Tokai, Aug 5–7.
23) T. Toyama, et al., "Development of Beam Loss Monitoring Using Proportional Counters at J–PARC", Proc. of 42nd ICFA Advanced Beam Dynamics Workshop on High-Intensity, High Brightness Hadron Beams, Nashville, Tennessee, USA, Aug. 25–29, 2008.
24) S. Motohashi, et al., "Improvement of PLC-based Data Acquisition System for Beam Loss Monitors of the J–PARC Main Ring", Proc. of the 6th Annual Meeting of Particle Accelerator Society of Japan, Tokai, Aug 5–7.
25) M. Komiyama, et al., "Upgrading the Control System of RIKEN RI-Beam Factory for New Injector using Embedded EPICS", Proc. of the 6th Annual Meeting of Particle Accelerator Society of Japan, Tokai, Aug 5–7.
26) http://www.celinuxforum.org/CelfPubWiki/JapanTechnicalJamboree13?action = AttachFile&do = get&target=preempt070222celfjambo13.pdf
27) C. H. Kuo, et al., "Design Status of the TPS Control System", Proc. of International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS09), Kobe, Japan, Oct. 12–16, 2009.
28) 小野善康，「景気と経済政策」，岩波新書，576.