EPICSの基本的枠組み

Noboru Yamamoto

H.E. Acc. Res. Org.(KEK) and J-PARC center, Ibaraki, JAPAN

November 2, 2018

1. この文書はEPIC2013に開催されたEPICS講習会に用意された資料をもとに、Latex/Beamer向けにかきなおされました。
2. 今回(2018/11/1)の講習会には不要な部分もありますので、それは飛ばしていきますので、よろしくお願いいたします。

Contents

Todo list

1. What is EPICS
2. EPICS Basics
   - DB:EPICS Rutime Database
   - CA:Channel Access
3. EPICS quick reference
   - 用語集
   - レコード型
   - フィールド名

Contents

Todo list

1. What is EPICS
2. EPICS Basics
   - DB:EPICS Rutime Database
   - CA:Channel Access
3. EPICS quick reference
   - 用語集
   - レコード型
   - フィールド名

1. EPICS の基礎を説明します。
2. EPICS は制御システムを構築するためのソフトウェア　フレームワークです。小規模な応用から、加速器や電波望遠鏡などの大型機器の制御まで、規模に応じたシステムを構築できます。
3. EPICS は基本的に装置の制御を目的に構築されており、高速なデータ収集などでは別のシステムが必要となる場合もあります。

EPICS とは？

EPICS: Experimental Physics and Industrial Control System

- EPICS は計算機による制御システム構築のためのフレームワーク／Toolkit です。EPICS を導入すれば、制御システムが完成する訳ではありません。
- とはいえ、ソフトウェア開発の手間は大きく削減できると期待されます。
  - 制御・測定を行うための基本的なソフトウェアはプログラミング無しで利用可能です。
  - スクリプト言語などからの利用も簡単にできます。
- 現在はフリーソフトウェアとして配布されています。
  - https://epics-controls.org/
- スタートは簡単に、かつ目的に応じた拡張が可能なアーキテクチャを採用しています。

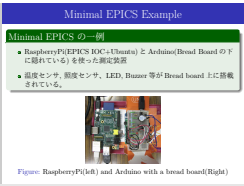
EPICS とは？
EPICS: Experimental Physics and Industrial Control System
EPICS は計算機による制御システム構築のためのフレームワーク／Toolkit です。EPICS を導入すれば、制御システムが完成する訳ではありません。
とはいえ、ソフトウェア開発の手間は大きく削減できると期待されます。
制御・測定を行うための基本的なソフトウェアはプログラミング無しで利用可能です。
スクリプト言語などからの利用も簡単にできます。
現在はフリーソフトウェアとして配布されています。
https://epics-controls.org/
スタートは簡単に、かつ目的に応じた拡張が可能なアーキテクチャを採用しています。

Minimal EPICS Example

Minimal EPICS の一例

- RaspberryPi(EPICS IOC+Ubuntu) と Arduino(Bread Board の下に隠れている) を使った測定装置
- 温度センサ, 照度センサ、LED, Buzzer 等が Bread board 上に搭載されている。

Figure: RaspberryPi(left) and Arduino with a bread board(Right)



1. ここでは、RPi と Arduino を組み合わせた例を示しています。今回の講習会のように RPi の直接デバイスを接続することも可能です
   1.1 Rapberry Pi
   1.2 Arduino: Arduino Poto snap とほぼ互換の部品を接続済み。
2. 大規模な応用としては、世界各地の高エネルギー加速器の制御システムの基盤として使われていることは、みなさんご存知でしょう。

EPICS の ABCD

音楽は Do-Re-Mi.
英語なら、ABC.
EPICS は ABCD からはじまります。

A B C D

縦書きに書いてみて、

A
B
C
D

折りたたんで、

C A

EPICS の ABCD
音楽は Do-Re-Mi.
英語なら、ABC.
EPICS は ABCD からはじまります。
A B C D
縦書きに書いてみて、
A
B
C
D
折りたたんで、
C A

EPICS の ABCD

横に読んでみると、

- CA: Channel Access
- DB: Data Base

この CA と DB が EPICS を理解する上でもっとも重要な二つの言葉です。

EPICS の ABCD
横に読んでみると、
CA: Channel Access
DB: Data Base
この CA と DB が EPICS を理解する上でもっとも重要な二つの言葉です。

“ABCD” を縦書き 2 行に書いて、横二行をそれぞれよむと、CA, DB となります。この二つの言葉が EPICS を理解する上で Key となります。

EPICS の ABCD

CA: Channel Access

- EPICS では全てのデータを Channel として読み書きします。
- Channel はシステムで一意な名前 (Channel 名) で指定します。e.g. caget ET_DEMO:USBTHERM1:MON あるいは ET_DEMO:USBTHERM1:MON.VAL
- EPICS システムで情報を交換するためのネットワークプロトコルです。
- 標準ネットワーク技術 (TCP & UDP/IP) の上に構築されています。

DB: runtime Data Base

- 全ての HW 制御はこの Runtime Database を経由します。
- DB = Records + Links
  - Database は入力、出力、演算などの機能をもつ “レコード ” から構成されます。
  - Database 中のレコードはデータの流れや、動作の流れを指示するリンクが存在します。

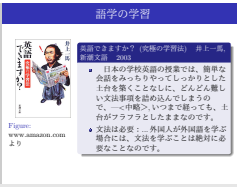
1. この例では、ET_DEMO:USBTHERM1:MON[.VAL] がチャンネル名です
2. この様に、チャンネル名の一般形は、<Record 名>.<フィールド名>となっています。但し、フィールド名が VAL の場合には、チャンネル名から.VAL の部分を省略しても良いことになっています。つまり　チャンネル名として <レコード名>　の形も許されますが、これは <レコード名>.VAL の省略形であるということになります。
3. Process Variable という呼び名は、Record/channel との呼称が確定する以前にこれらの概念につけられた呼び名です。Record/Field/Channel を区別して使うのが適切です。一方 EPICS の将来バージョン（？）である EPICS 7 (EPICS 4 + EPICS 4) では、　CA プロトコルを発展させた PVAccess プロトコルが定義されています。この新しいプロ

語学の学習

Figure:
www.amazon.com
より

英語できますか？ (究極の学習法)　井上一馬, 新潮文語　2003

- 日本の学校英語の授業では、簡単な会話をみっちりやってしっかりとした土台を築くことなしに、どんどん難しい文法事項を詰め込んでしまうので、—<中略>, いつまで経っても、土台がフラフラとしたままなのです。
- 文法は必要：... 外国人が外国語を学ぶ場合には、文法を学ぶことは絶対に必要なことなのです。



EPICS を現場で応用しようとするのは、ネイティブスピーカーではない人が英語を学ぶのに似ていて、実地での経験と EPICS の考え方に関する知識の双方が必要だと思いわれます。ここでは、英語の文法ほどではありませんが、EPICS の仕組みについてお話したいとおもいます。実際のデータベースの作成などで、疑問に思われたことのいくつかでも解決すれば良いのですが。

DB: Runtime Database

DB: Runtime Database

DB: Runtime Database

DB: Runtime Database

DB: Runtime Database

DB: Runtime Database

DB: Runtime Database

DB: Runtime Database
DB: Runtime Database
DB: Runtime Database
DB: Runtime Database
DB: Runtime Database
DB: Runtime Database
DB: Runtime Database
DB: Runtime Database

DB

DB=Records + Links

DB の基本要素

NO RDB EPICS DB は Relational Dabase ではありません。
EPICS DB は on Memory のデータベースです。

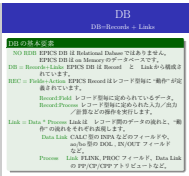
DB = Records+Links EPICS DB は Record と Link から構成されています。

REC = Fields+Action EPICS Record はレコード型毎に “動作” が定義されています。

- **Record:Field** レコード型毎に定められているデータ。
- **Record:Process** レコード型毎に定められた入力／出力／計算などの操作を実行します。

Link = Data * Process Link は レコード間のデータの流れと、“動作” の流れをそれぞれ表現します。

- **Data Link** CALC 型の INPA などのフィールドや、ao/bo 型の DOL , IN/OUT フィールドなど。
- **Process Link** FLINK, PROC フィールド、Data Link の PP/CP/CPP アトリビュートなど。



Record 型

Record type= Data fields + Process(Action, Function)

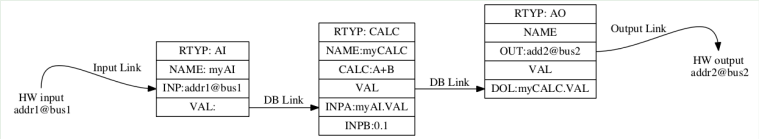
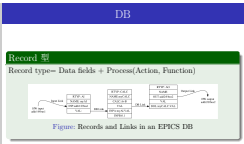


Figure: Records and Links in an EPICS DB



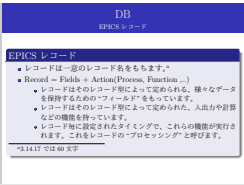
EPICS DB の一例。HW からデータを読み込み、その値を元に、HW に値を設定するという動作を表しています。

1. ここには三つのレコードがあります。それぞれ、AI 型、CALC 型、AO 型のレコードです。
2. AI 型レコードは HW からデータを読み出すという機能を持ちます。読みだした値は VAL フィールドに収められます。
3. AO 型レコードは HW にデータを書き出すという機能を持ちます。DOL フィールドで指定されたレコードの値を出力します。出力した値は VAL フィールドに収められます。
4. 入出力先は INP/OUT フィールドにしていします。HW に出力する際は、HW のバス／デバイスのアドレスをそれぞれの HW が指定するやり方で指定します。入出力先をチャンネルとすることもできます。
5. CALC 型レコードは、CALC フィールドに指定された計算

EPICS レコード

- レコードは一意のレコード名をもちます。[a]
- Record = Fields + Action(Process, Function ,..)
  - レコードはそのレコード型によって定められる、様々なデータを保持するための “フィールド” をもっています。
  - レコードはそのレコード型によって定められた、入出力や計算などの機能を持っています。
  - レコード毎に設定されたタイミングで、これらの機能が実行されます。これをレコードの “プロセッシング” と呼びます。

[a] 3.14.17 では 60 文字



レコード名はチャンネル名としても有効ですが、全てのチャンネル名がレコード名と言うわけではありません。次に説明するように、レコード名とフィールド名を合わせたものがチャンネル名となります。名前 (NAME), 値 (VAL) など全てのレコードに共通なフィールドもありますが、レコード型特有のフィールドもあります。フィールド名は通常 3 あるは 4 文字の大文字の文字列です. 具体的には、” チャンネルに値を書き出すと、必要があれば適切に変換されて、HW に出力される。” などです。たとえば、AO:analoge output 型は浮動小数点のデータをハードウェアに書き込む機能を持ちます。

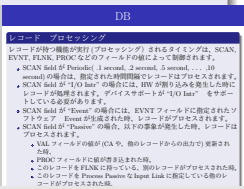
1. Field=プロパティ、Action=メソッドと考えると、Record=Oject と言いたくなりますが、EPICS Record の Action はレコード型に一つだけ。プログラム上は、Record は C++の Object として実装されており、複数のメソッドを持っていますが、EPICS DB で明示的に記述するのは

DB

レコード　プロセッシング

レコードが持つ機能が実行 (プロセッシング) されるタイミングは、SCAN, EVNT, FLNK, PROC などのフィールドの値によって制御されます。

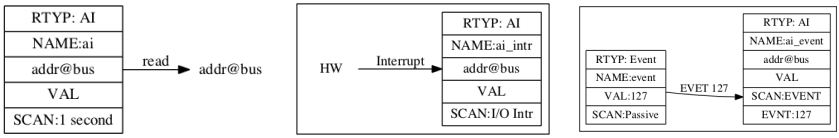
- SCAN field が Periodic(.1 second, .2 second, .5 second, . . . ,10 second) の場合は、指定された時間間隔でレコードはプロセスされます。
- SCAN field が “I/O Intr” の場合には、HW が割り込みを発生した時にレコードが処理されます。デバイスサポートが “I/O Intr” をサポートしている必要があります。
- SCAN field が “Event” の場合には、EVNT フィールドに指定されたソフトウェア Event が生成された時、レコードがプロセスされます。
- SCAN field が “Passive” の場合、以下の事象が発生した時、レコードはプロセスされます。
  - VAL フィールドの値が (CA や、他のレコードからの出力で) 更新された時、
  - PROC フィールドに値が書き込まれた時。
  - このレコードを FLNK に持っている、別のレコードがプロセスされた時。
  - このレコードを Process Passive な Input Link に指定している他のレコードがプロセスされた時。


DB
レコード　プロセッシング
レコードが持つ機能が実行 (プロセッシング) されるタイミングは、SCAN, EVNT, FLNK, PROC などのフィールドの値によって制御されます。
SCAN field が Periodic(.1 second, .2 second, .5 second, . . . ,10 second) の場合は、指定された時間間隔でレコードはプロセスされます。
SCAN field が “I/O Intr” の場合には、HW が割り込みを発生した時にレコードが処理されます。デバイスサポートが “I/O Intr” をサポートしている必要があります。
SCAN field が “Event” の場合には、EVNT フィールドに指定されたソフトウェア Event が生成された時、レコードがプロセスされます。
SCAN field が “Passive” の場合、以下の事象が発生した時、レコードはプロセスされます。
VAL フィールドの値が (CA や、他のレコードからの出力で) 更新された時、
PROC フィールドに値が書き込まれた時。
このレコードを FLNK に持っている、別のレコードがプロセスされた時。
このレコードを Process Passive な Input Link に指定している他のレコードがプロセスされた時。

CPP は SCAN が Passive の場合のみレコードプロセス。CP の場合は Passive の値に関わらずプロセスされます。SCAN=Passive と FLINK, Input/Output Link を使うことで、制御のシークエンスを EPICS DB 内に実装することができます。

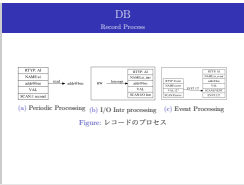
DB

Record Process



(a) Periodic Processing (b) I/O Intr processing (c) Event Processing

Figure: レコードのプロセス



DB

Record Process

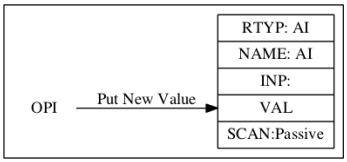


Figure: Passive Processing

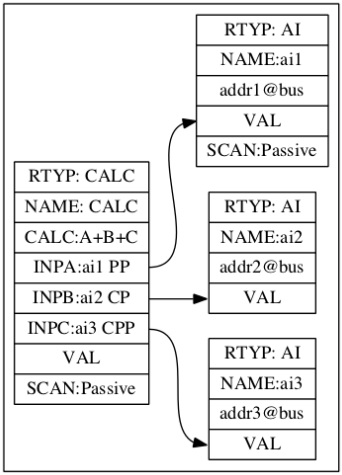
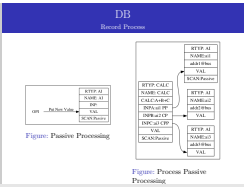


Figure: Process Passive Processing



CA : Channel Access

CA : Channel Access

CA : Channel Access

CA : Channel Access

CA : Channel Access

CA : Channel Access

CA : Channel Access

CA:Channel Access

CA : Channel Access
CA : Channel Access
CA : Channel Access
CA : Channel Access
CA : Channel Access
CA : Channel Access
CA : Channel Access

EPICS アーキテクチャ

ハードウェア構成

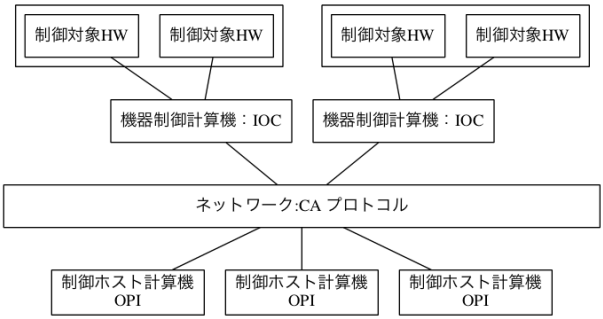
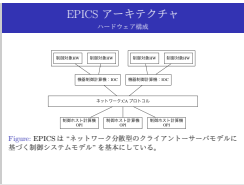


Figure: EPICS は “ネットワーク分散型のクライアントーサーバモデルに基づく制御システムモデル” を基本にしている。



EPICS は分散配置された多数の HW 制御計算機と制御室等に設置された制御ホスト計算機をネットワークで相互に接続した形を基本としています。小規模なシステムの場合には、HW 計算機 (IOC) と制御ホスト計算機 (OPI) が同じ計算機装置である場合もあります。この場合にも、プロセス間の通信は IP によるネットワーク層を経由します。

CA サーバと CA クライアント

- CA はサーバ＆クライアントモデルを採用しています。[a]
  - IOCsh は CA サーバとしての役割を持ちます。
  - IOCsh 以外にも、gateway, portable CAS など の CA サーバが存在します。
  - OPI 上のプログラムは通常、CA クライアントです。
  - IOCsh は CA クライアンとしても動作します。

[a] 説明文書：Channel Access Library Reference manual (https://epics.anl.gov/base/R3-14/8-docs/CAref.html)

Channel Access
CA サーバと CA クライアント
CA はサーバ＆クライアントモデルを採用しています。a
IOCsh は CA サーバとしての役割を持ちます。
IOCsh 以外にも、gateway, portable CAS など の CA サーバが存在します。
OPI 上のプログラムは通常、CA クライアントです。
IOCsh は CA クライアンとしても動作します。
a 説明文書：Channel Access Library Reference manual (https://epics.anl.gov/base/R3-14/8-docs/CAref.html)

EPICS アーキテクチャ

ソフトウエア構成

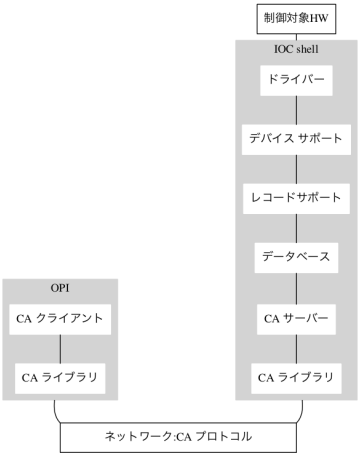
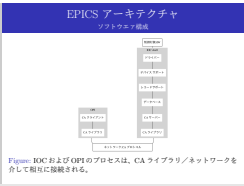


Figure: IOC および OPI のプロセスは、CA ライブラリ／ネットワークを介して相互に接続される。



IOC の中では、制御対象の状態は DB 中のレコードとして表現されます。それぞれのハードウェアの違いは、Device サポートや Driver Ssuport が吸収します。IOC のプロセス間、IOC-OPI プロセス間の通信は全て CA となります。

CA クライアント

- OPI 上で動作する汎用アプリケーション群
- 制御に必要な基本的な機能をすぐに使える。
  - 制御画面作成と実行：EDM, MEDM, CSS/BOY,
  - Data Archiving：Channel Archiver, CSS/Archiver
  - その他の Tools: StripTool, Probe, AlarmHandler, ...
- 様々なプログラム言語で自ら開発することも可能です。
  C/C++, Java, Python, Perl, SAD, ...

Channel Access
CA クライアント
- OPI 上で動作する汎用アプリケーション群
- 制御に必要な基本的な機能をすぐに使える。
  - 制御画面作成と実行：EDM, MEDM, CSS/BOY,
  - Data Archiving：Channel Archiver, CSS/Archiver
  - その他の Tools: StripTool, Probe, AlarmHandler, ...
- 様々なプログラム言語で自ら開発することも可能です。
  C/C++, Java, Python, Perl, SAD, ...

CA library への API だけでなく、GUI アプリケーションを作成するための Widget なども用意されています。

CA Library の基本操作

search & connect: file の open に相当。指定された Channel をシステム内で探して、接続を確立します。

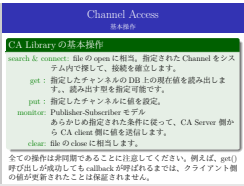
get : 指定したチャンネルの DB 上の現在値を読み出します。、読み出す型を指定可能です。

put : 指定したチャンネルに値を設定。

monitor: Publisher-Subscriber モデル
あらかじめ指定された条件に従って、CA Server 側から CA client 側に値を送信します。

clear: file の close に相当します。

全ての操作は非同期であることに注意してください。例えば、get() 呼び出しが成功しても callback が呼ばれるまでは、クライアント側の値が更新されたことは保証されません。


Channel Access
基本操作
CA Library の基本操作
search & connect: file の open に相当。指定された Channel をシステム内で探して、接続を確立します。
get : 指定したチャンネルの DB 上の現在値を読み出します。、読み出す型を指定可能です。
put : 指定したチャンネルに値を設定。
monitor: Publisher-Subscriber モデル
あらかじめ指定された条件に従って、CA Server 側から CA client 側に値を送信します。
clear: file の close に相当します。
全ての操作は非同期であることに注意してください。例えば、get() 呼び出しが成功しても callback が呼ばれるまでは、クライアント側の値が更新されたことは保証されません。

Unix のファイルシステムでは、open/read/write/close という標準操作がつかわれます。EPICS CA では、search and connect/get/put/monitor/close が基本操作となります。search には UDP broadcast/multicast/unicast が使われます。EPICS_CA_ADDR_LIST および EPICS_CA_AUTO_ADDR_LIST でコントロール get の要求毎に HW からデータを読み出すわけではないことに注意が必要かもしれません。これは、いわゆる subscribe-publisher モデルによる通信と言うことです。モニタ機能を有効に使うことで、効率のよい制御システムとなります。

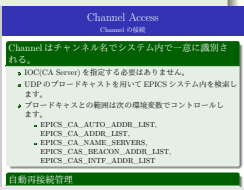
Channel Access

Channel の接続

Channel はチャンネル名でシステム内で一意に識別される。

- IOC(CA Server) を指定する必要はありません。
- UDP のブロードキャストを用いて EPICS システム内を検索します。
- ブロードキャスとの範囲は次の環境変数でコントロールします。
  - EPICS_CA_AUTO_ADDR_LIST, EPICS_CA_ADDR_LIST,
  - EPICS_CA_NAME_SERVERS, EPICS_CAS_BEACON_ADDR_LIST, EPICS_CAS_INTF_ADDR_LIST

自動再接続管理



同じ名前があると、CA クライアントは先に返事を返したチャンネルに接続されます。サブネットを超えて CA で通信する場合には、これらの環境変数を適切に設定するか、CA gateway を導入します。

EPICS CA の search/get/put は非同期

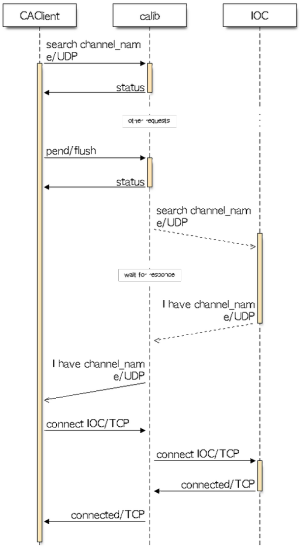
- search/get/put などの関数呼び出し終了は操作の終了を意味しません。
- C/C++のライブラリでは、操作の終了時に callback 関数を呼出す様に設定するのが普通です。
- 複数の呼び出しがキューに入れられます。put 呼び出し等では、適当なところで flush_io 関数などを呼出しておくのが安全です。

Channel Access
非同期入出力
EPICS CA の search/get/put は非同期
search/get/put などの関数呼び出し終了は操作の終了を意味しません。
C/C++のライブラリでは、操作の終了時に callback 関数を呼出す様に設定するのが普通です。
複数の呼び出しがキューに入れられます。put 呼び出し等では、適当なところで flush_io 関数などを呼出しておくのが安全です。

monitor も callback を使います。

CA

CA プロトコル



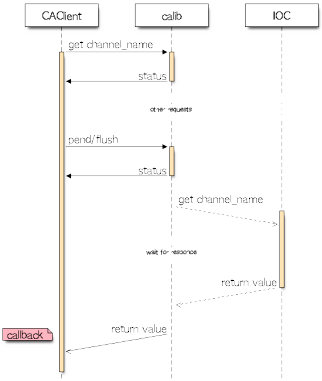
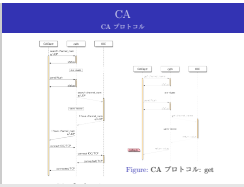
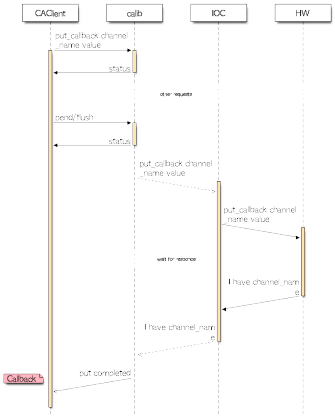


Figure: CA プロトコル: get



CA

CA プロトコル



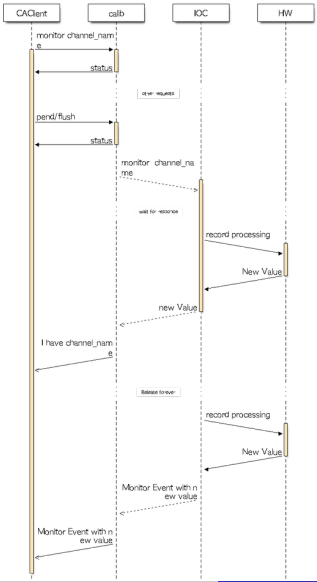
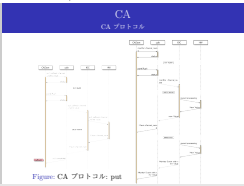


Figure: CA プロトコル: put



Monitor Event

Monitor Event はいつ起きるのか？

- レコードがプロセスされて.VAL が更新され、かつ monitor の条件がみたされた時。
- .STAT や.SEVR　が変化した時
- .ESLO, .EOFF などの特定のフィールドが変化したとき。(レコード型による)

Monitor Event
Monitor Event はいつ起きるのか？
レコードがプロセスされて.VAL が更新され、かつ monitor の条件がみたされた時。
STAT や SEVR　が変化した時
ESLO, .EOFF などの特定のフィールドが変化したとき。(レコード型による)

1. CA で.VAL を変更した時は、Monitor Event が発生しない。

Monitor Event

Monitor Event はいつ起きるのか？

整数型：.VAL など

- VAL の新しい値が古い値 (.MLST) と異なる時。

浮動少数点型データ:.VAL

- デッドバンド (.MDEL, .ADEL) を超えて変化した時。(abs(.VAL - .OVAL) > .MDEL)
- .MDEL は表示用のデッドバンド
- .ADEL はデータアーカイブ用のデッドバンド

CA でモニタを要求する際に、表示 (Monitor) 用かアーカイブ用かを指定できます。全てのフィールドが monitor event を発生するわけではありません。レコードサポートのなかで、post_event API を呼び出した時、monitor event が発生されます。

Channel Access

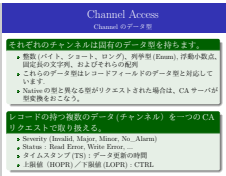
Channel のデータ型

それぞれのチャンネルは固有のデータ型を持ちます。

- 整数 (バイト、ショート、ロング)、列挙型 (Enum), 浮動小数点、固定長の文字列、およびそれらの配列
- これらのデータ型はレコードフィールドのデータ型と対応しています.
- Native の型と異なる型がリクエストされた場合は、CA サーバが型変換をおこなう。

レコードの持つ複数のデータ (チャンネル) を一つの CA リクエストで取り扱える。

- Severity (Invalid, Major, Minor, No_Alarm)
- Status : Read Error, Write Error, ...
- タイムスタンプ (TS) : データ更新の時間
- 上限値 (HOPR) ／下限値 (LOPR) : CTRL



列挙型のフィールドでは、整数の代わりに文字列を使って入出力を行うことができます。

まとめ

EPICS

- CA:Channel Access
  - 非同期動作
  - monitor
- DB: EPICS runtime DataBase
- DB = Records + Links
- Records = Files + Action
- Links = Data * Process

まとめ
EPICS
CA:Channel Access
非同期動作
monitor
DB: EPICS runtime DataBase
DB = Records + Links
Records = Files + Action
Links = Data * Process

Table: EPICS でよく使われる略語のリスト

EPICS	Experimental Physics and Industrial Control Sytesm
IOC	Input Output Controller
OPI	Operator Interface
CA	Channel Access
DB	EPICS DataBase

EPICS quick reference
EPICS jargons
Table: EPICS でよく使われる略語のリスト
EPICS Experimental Physics and Industrial Control Sytesm
IOC Input Output Controller
OPI Operator Interface
CA Channel Access
DB EPICS DataBase

EPICS quick reference

EPICS Records

Table: Basic EPICS records
(all records are part of EPICS base distribution)

AI	Analog input.
AO	Analog output.
BI	Binary input.
BO	Binary output.
LONGIN	Integer input.
LONGOUT	Integer output.
STRINGIN	String input.
STRINGOUT	String output.
WAVEFORM	Array input/output.
SUBARRAY	Used to obtain sub-array from WAVEFORM records.
CALC	Calculation . used for simple calculations.
CALCOUT	Calculation output . same as CALC but with output

EPICS quick reference
EPICS Records
Table: Basic EPICS records
(all records are part of EPICS base distribution)
AI Analog input.
AO Analog output.
BI Binary input.
BO Binary output.
LONGIN Integer input.
LONGOUT Integer output.
STRINGIN String input.
STRINGOUT String output.
WAVEFORM Array input/output.
SUBARRAY Used to obtain sub-array from WAVEFORM records.
CALC Calculation . used for simple calculations.

EPICS quick reference

Fields of EPICS Records

Table: Common Fields

NAME	Record Name
INP	Input Link
DESC	Description
OUT	Output Link
DTYP	Device Type
DOL	Desired Output Location
PROC	Process Record
OMSL	Output Mode Select
FLNK	Forward Link
EGU	Engineering Units
PACT	Processing Active
HOPR	High Operating Range
STAT	Current Alarm Status
LOPR	Low Operating Range

EPICS quick reference
Fields of EPICS Records
Table: Common Fields
NAME Record Name
INP Input Link
DESC Description
OUT Output Link
DTYP Device Type
DOL Desired Output Location
PROC Process Record
OMSL Output Mode Select
FLNK Forward Link
EGU Engineering Units
PACT Processing Active
HOPR High Operating Range
STAT Current Alarm Status

EPICS quick reference

basic commands

Table: Basic bash-level commands

caget <recName>	Performs CA get. Type “caget .h” for complete help.
caput <recName> <value>	Performs CA put. Type “caput .h” for complete help.
camonitor <recName>	Performs CA monitor. Type “camonitor .h” for complete help.
makeBaseApp.pl	Creates empty or example EPICS application. Type “makeBaseApp.pl .h” for a complete help.
make	Compile and install EPICS application.
make clean uninstall	Uninstall EPICS application

EPICS quick reference
basic commands
Table: Basic bash-level commands
caget <recName> Performs CA get. Type “caget .h” for complete help.
caput <recName> <value> Performs CA put. Type “caput .h” for complete help.
camonitor <recName> Performs CA monitor. Type “camonitor .h” for complete help.
makeBaseApp.pl Creates empty or example EPICS application. Type “makeBaseApp.pl .h” for a complete help.
make Compile and install EPICS application.
make clean uninstall Uninstall EPICS application

EPICS quick reference

iocsh commands

Table: iocsh commands

dbl <record type> <field list> Prints the name of the records in the run time database. If <record type> provided only records of that type are listed. If also <field list> is proved also specified fields are printed. Example "dbl ai,VAL"

dbgrep <pattern> List record names that match pattern.

dbgf <recordName.fieldName> Prints the filed type and value.

dbpf <recordName.fieldName> <value> Set the field of specified record.

dbpr <recordName> <internes level> Prints the filed values. Interest level defines which fields are displayed.

dbtr <recordName> Process record and issues dbpr command.

dbnr Displays number of records by types.

EPICS quick reference
iocsh commands
Table: iocsh commands
dbl <record type> <field list> Prints the name of the records in the run time database. If <record type> provided only records of that type are listed. If also <field list> is proved also specified fields are printed. Example "dbl ai,VAL"
dbgrep <pattern> List record names that match pattern.
dbgf <recordName.fieldName> Prints the filed type and value.
dbpf <recordName.fieldName> <value> Set the field of specified record.
dbpr <recordName> <internes level> Prints the filed values. Interest level defines which fields are displayed.
dbtr <recordName> Process record and issues dbpr command.
dbnr Displays number of records by types.

EPICS 開発環境

makeBaseApp.pl -t <template> <application name>

EPICS のアプリケーション (DB, Device Support, Sequencer program) などを作成するための環境 (directory, Makefile 等）を作成。

makeBaseApp.pl -i -t <template> <ioc name>

上記アプリケーションを使って作成した iocsh を起動する。

EPICS 開発環境
makeBaseApp.pl -t <template> <application name>
EPICS のアプリケーション (DB, Device Support, Sequencer program) などを作成するための環境 (directory, Makefile 等）を作成。
makeBaseApp.pl -i -t <template> <ioc name>
上記アプリケーションを使って作成した iocsh を起動する。

makeBaseApp.pl

EPICS アプリケーション開発に使う標準的なディレクトリ構造や、そのなかのテンプレートとなるファイルを作成する。

- データベース
- OPI
- Device Support/Record Support
- Sequencer プログラム
- Ca クライアント
- EPICS アプリケーションの実行
  - IOC/IOCSH の起動スクリプト等を半自動的に生成してくれる。

makeBaseApp.pl

EPICS アプリケーション開発に使う標準的なディレクトリ構造や、そのなかのテンプレートとなるファイルを作成する。
- データベース
- OPI
- Device Support/Record Support
- Sequencer プログラム
- Ca クライアント
- EPICS アプリケーションの実行
  - IOC/IOCSH の起動スクリプト等を半自動的に生成してくれる。

EPICS アプリケーションの開発では、定型的に設定すべきことがいくつかある。それぞれの要素毎に適切な環境変数等を設定。ディレクトリ構造も仮定することで、効率化をはかる。これにより、コンパイラ／リンクオプションの指定等が単純化される。

makeBaseApp.pl

makeBaseApp.pl を実行する前に、アプリケーション開発用ディレクトリを作成する。

```
mkdir ~/EpicsADE
cd ~/EpicsADE
```

Top ディレクトリ内で次の二つのコマンドを実行する。

```
makeBaseApp.pl -t example mySample
makeBaseApp.pl -t example -i mySample
```

利用可能なテンプレートの一覧は

```
makeBaseApp.pl -l
```

で表示される。

makeBaseApp.pl
makeBaseApp.pl を実行する前に、アプリケーション開発用ディレクトリを作成する。
mkdir ~/EpicsADE
cd ~/EpicsADE
Top ディレクトリ内で次の二つのコマンドを実行する。
makeBaseApp.pl -t example mySample
makeBaseApp.pl -t example -i mySample
利用可能なテンプレートの一覧は
makeBaseApp.pl -l
で表示される。

example は makeBaseApp が持っているテンプレートの名前。
mySample はユーザがきめるアプリケーションの名前。

makeBaseApp.pl
ディレクトリ構造

```
<TOP>/
    Makefile
    configure/ 設定ファイル等。通常は変更の必要なし
        RELEASE EPICS Base のインストール先等を設定
        ...
    SampleApp
        src/    C/C++/SNL 等のソースコード
            Makefile
            SampleMain.cpp
            ...
        Db/     DB, templates & substitutions ファイル
            ...
<TOP>/iocBoot
    Makefile
    iocSample/
```


makeBaseApp.pl
ディレクトリ構造
<TOP>/
Makefile
configure/ 設定ファイル等。通常は変更の必要なし
RELEASE EPICS Base のインストール先等を設定
...
SampleApp
src/ C/C++/SNL 等のソースコード
Makefile
SampleMain.cpp
...
Db/ DB, templates & substitutions ファイル
...
<TOP>/iocBoot
Makefile
iocSample/

maekBaseApp.pl で作成されるディレクトリ構造

makeBaseApp.pl

ターゲットアーキテクチャの設定

Top ディレクトリで makeBaseApp.pl を使って初めてアプリケーションディレクトリを作成したら、

```
cd ~/EpicsADE
make EPICS_HOST_ARCH=linux-x86  install
```

を一度だけ実行する.
EPICS でサポートされているホストには、

linux-x86 intel x86 上の linux
darwin-ppcx86 MacOSX の Universal binary
win32-x86 Windows 上の MS C++を使った開発
linux-x86_64 64bits Linux
win32-x86-cygwi cygwin 環境を Windows で使う場合。
などがある。

makeBaseApp.pl
ターゲットアーキテクチャの設定
Top ディレクトリで makeBaseApp.pl を使って初めてアプリケーションディレクトリを作成したら、
cd ~/EpicsADE
make EPICS_HOST_ARCH=linux-x86 install
を一度だけ実行する.
EPICS でサポートされているホストには、
linux-x86 intel x86 上の linux
darwin-ppcx86 MacOSX の Universal binary
win32-x86 Windows 上の MS C++を使った開発
linux-x86_64 64bits Linux
win32-x86-cygwi cygwin 環境を Windows で使う場合。
などがある。

EPICS_HOST_ARCH 環境変数は、アプリケーションを開発するホストの OS と CPU アーキテクチャを示す文字列。

EPICS databse のロード

dbLoadDatabase ”O.Common/streamApp.dbd”

1. レコード型の定義、
2. デバイスサポート／ドライバサポートの登録

dbLoadRecords(”./devArduinoDebug.db”,”P=noboruHost,R=0,TTY=TTY,ascan=.2 second”)

1. レコードのデータベースへの登録／初期設定
2. マクロ (例：”P=noboruHost,R=0,TTY=TTY,ascan=.2 second”) によるレコード登録のパラメータ化を実現
3. 多数の繰り返しには、dbLoadTemplate も使える。
4. dbLoadTemplate ”db/userHost.substitutions”

EPICS databse のロード
dbLoadDatabase ”O.Common/streamApp.dbd”
レコード型の定義、
デバイスサポート／ドライバサポートの登録
dbLoadRecords(”./devArduinoDebug.db”,”P=noboruHost,R=0,TTY=TTY,ascan=.2 second”)
レコードのデータベースへの登録／初期設定
マクロ (例：”P=noboruHost,R=0,TTY=TTY,ascan=.2 second”) によるレコード登録のパラメータ化を実現
多数の繰り返しには、dbLoadTemplate も使える。
dbLoadTemplate ”db/userHost.substitutions”