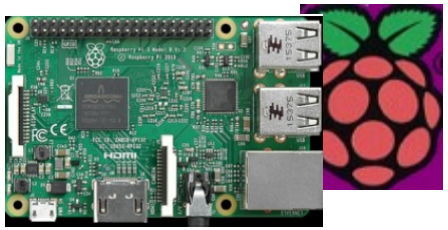
EPICS セミナー @KEK, 2018.11

2018/11/01-02

Kami, KEK/J-PARC

DatabaseとIOC



n.kamikubota

KEK / J-PARC

DatabaseとIOC　…
と言っておきながら

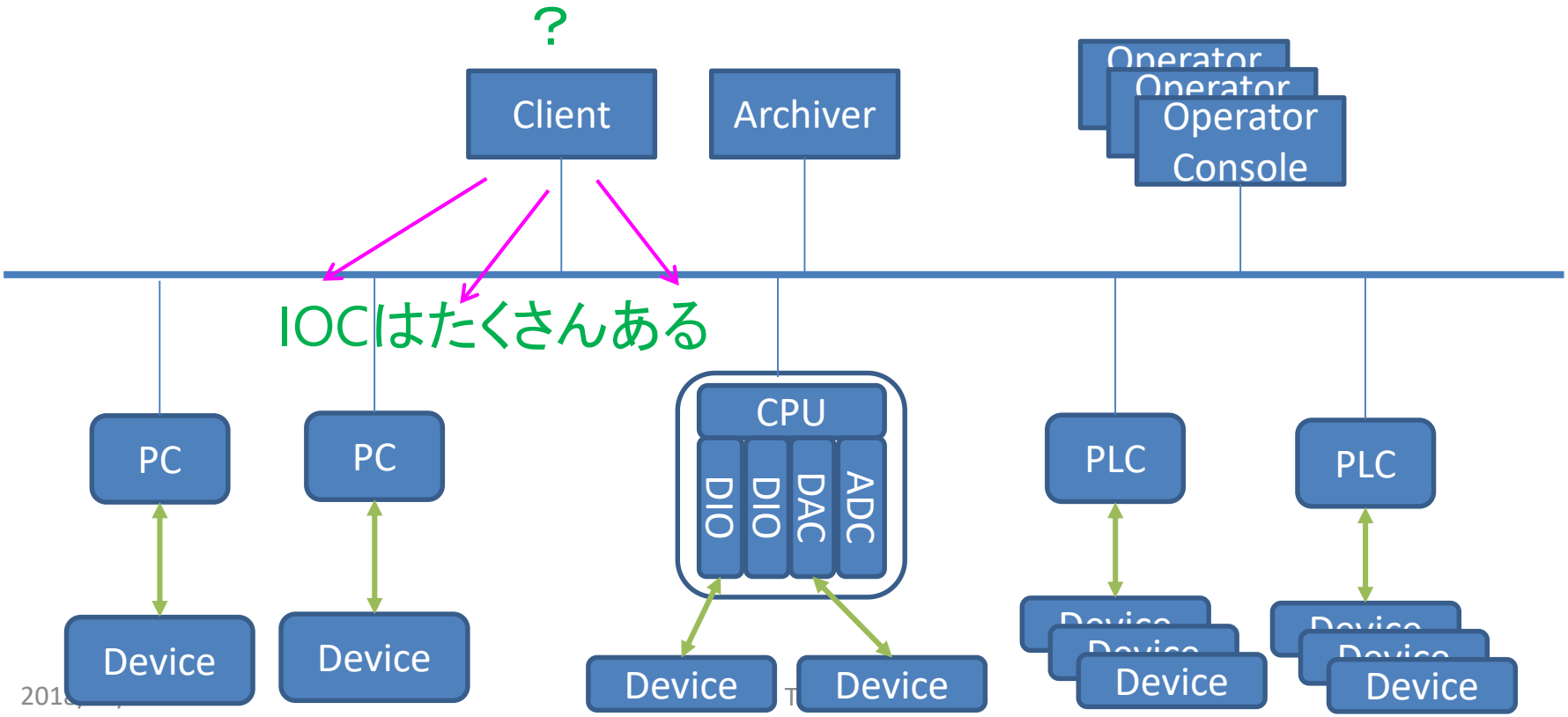
午前の話（EPICS Channel Access Protocol ）の
続きを少し

(資料 by obina)

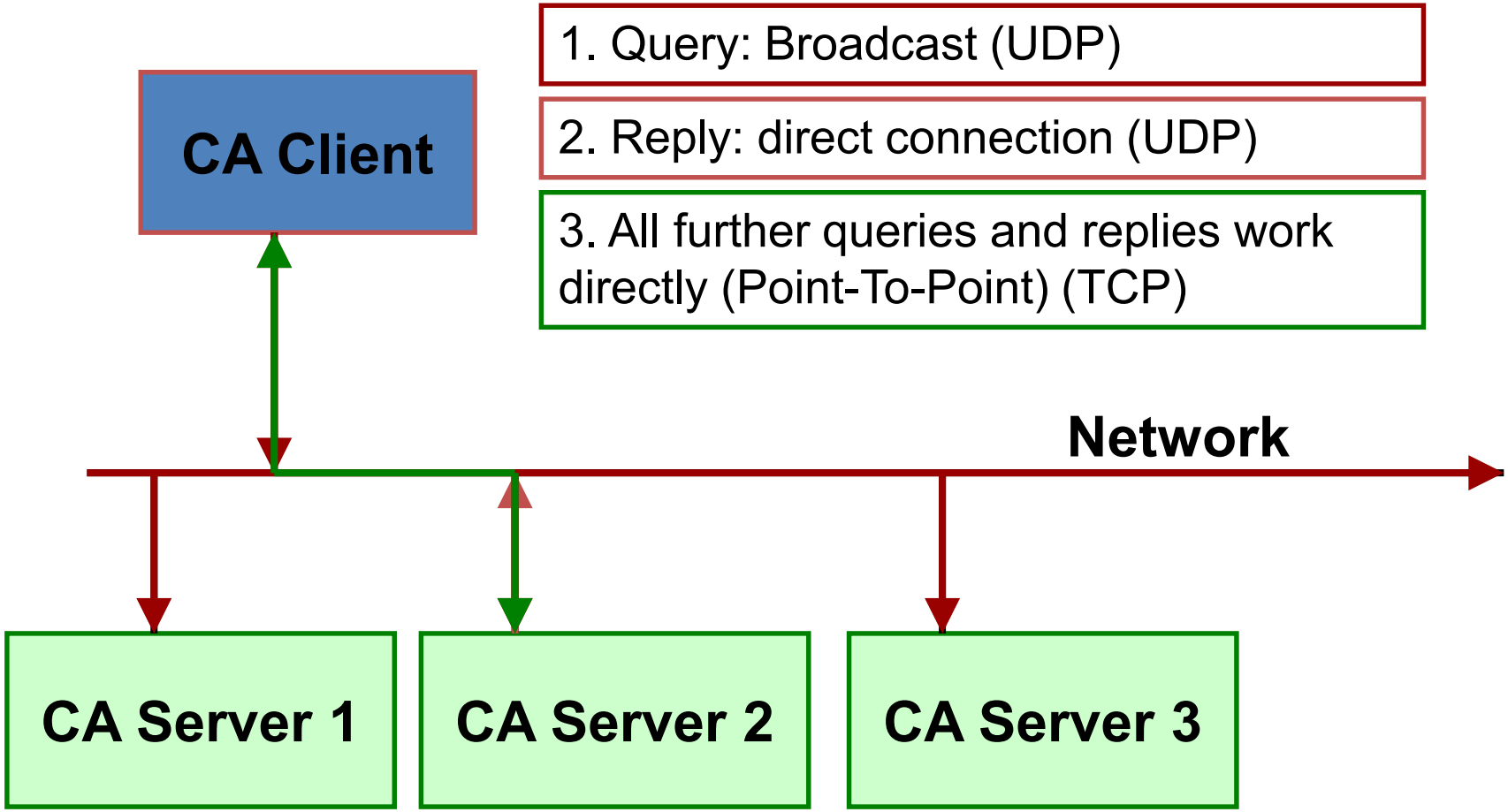
も少し Channel Access Protocol

EPICSでは、どのようにして「レコード名だけ」で対象となるIOCを見つけてデータをとってくるのか？

ネットワーク内にIOCはたくさん存在しており、どれが何のレコードを持っているかは分からない。



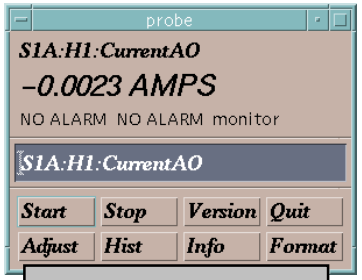
Some Details of Channel Access



Default UDP ports: 5064 and 5065
Default TCP ports: 5064 and 5065

Defined in environment variables
Need to be free in firewall!

Channel Access Commands



Channel Access Client

CA Client

CA Server

Channel Access Server

Process Variables:

S1A:H1:CurrentAO

S1:P

S1:P

S1:G1:vacuu

"connection request" or "search request"

"get" or "caget"

"put" or "caput"

"set a monitor"

Who has a PV named "S1A:H1:CurrentAO"?

I do.

What is its value?

25.5 AMPS

Change its value to 30.5

OK, it is now 30.5

"put complete"

30.5 is too high. It is now set to the maximum value of 27.5.

You are not authorized to change this value

Notify me when the value changes

It is now 20.5 AMPS

It is now 10.5 AMPS

It is now 0.0023 AMPS

"post an event" or "post a monitor"

For all fuctionality see
http://www.aps.anl.gov/epics/docs/ca.php
for EPICS R3.14 Channel Access Reference Manual

DatabaseとIOCの話

の
始まり

What is “database” ?

- EPICSでよく使われる用語
  - OPI, IOC, CA(Channel Access), PV(~レコード), で .. Database !

- レコード（or PV=Process Variable）とは？
  - 名前がついた信号の一つ一つを指す
  - 読んだり(caget)、書いたり(caput)、する単位
- Databaseとは？
  - よその講習会の説明では
    「Database = Records + Fields + Links」
  - いわゆる Relational Database (MySql, Oracle, PostGresql, ..)ではない
  - Databaseは、IOCに実体があるレコードの集合体である
    - と言われてもよくわからないんじゃ・・ まあ 次々っ

What is “database”？（続き）

- レコードを定義してみる
  - 文法

```
record(record_type, record_name) {
  field(field_name, “value”)
    …
  #commnet
}
```

実装の時、普通は、
「**hoge.db**」というfile名にする

Field定義は複数行あることが普通
指定が無ければDefault値になる

  - 例（PLC型IOC用の、Digital Input用のひな形db）

(bi型) **di.db**　1bit input の定義

1行目：**bi型で指定の名前のレコードを作るぜ！**

```
record(bi,"$(head):DI:SLOT$(slot):CH$(ch)")
{
        field(DTYP,"F3RP61")
        field(INP,"@U0,S$(slot),X$(ch)")
        field(SCAN,"$(scan)")
        field(ONAM,"On")
        field(ZNAM,"Off")
}
```

DTYPは、横河PLC”F3RP61”を指定
INPは、PLCの信号場所の指定

ONAM, ZNAMは、値が0ne=1及び
Zero=0の時、cagetで返す文字列

SL1
CPU
(RP61)

SL2
(DO)

SL3
(DI)

KEKでよく使われる
横河PLC型IOC

マクロ指定：
- $(head)など**マクロは、実装時に別の文字列に置き換えられる**
- $(slot)や$(ch)は、hardware moduleに関連付けられる
- $(scan)は、繰り返し読み出す周期を指定する（0.5秒など）

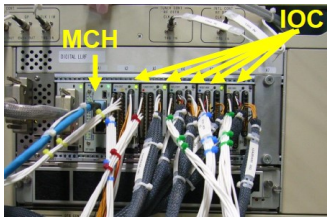
What is IOC ?

- IOC = Input / Output Controller
  - 昔(90年代)は “VME-bus computer running VxWorks” だけ
  - 今は “anything in which PVs are running”
    - (bus+module 型) VME, PLC(Yokogawa), cPCI, NI PXI/cRIO, xTCA, …
    - (FPGA組込み型) Xilinx Zynq(FPGA+Arm), Altera/Intel, NI cRIO, Suzaku, ..
    - (その他) ＝＞次ページにて‥

VME

PLC

uTCA



cRIO

NI LabViewは、EPICS CA のoptionがあります

■ ■ ■

- IOCのOS (Operating Systems)は
  - 昔(90年代)は Real-time OS “VxWorks” だけ
    - VME+VxWの縛りが消えたのは2004年ver.3.13->3.14になった時
  - 今は Linux, Window, VxWorks, RTEMS, ..

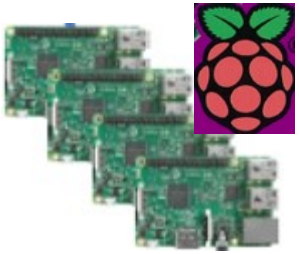
今日の実習では 3.15を使用

What is IOC？(続き)

- 今日の実習で使うIOC
  - (Soft IOC) RealなI/Oが無く、ソフトウェアだけのIOC
    - Real I/Oが無いのに役に立つ？＝＞役に立ちます・・
    - “softIOC” command で、今日は仮想電源のお試し
  - (Raspberry Pi) GPIOなどで何かReal I/Oがつながる
    - ご存じ超小型PC EPICS IOCになります
    - “makeBaseApp” command で、今日はLチカのお試し（GPIO制御）

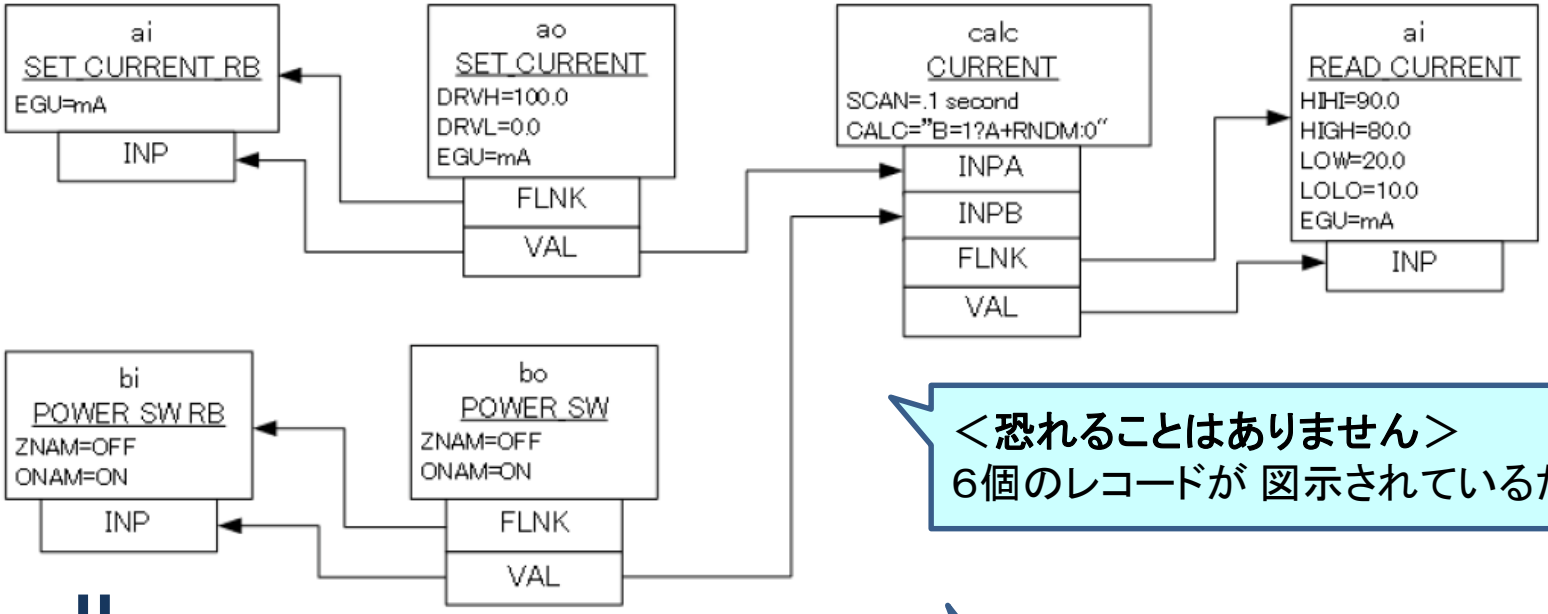
Soft

RPi



再び What is “database”？ （再び）

- bi 1個より 面倒な（普通な？）レコードを定義：仮想電源



＜恐れることはありません＞
6個のレコードが 図示されているだけ

||

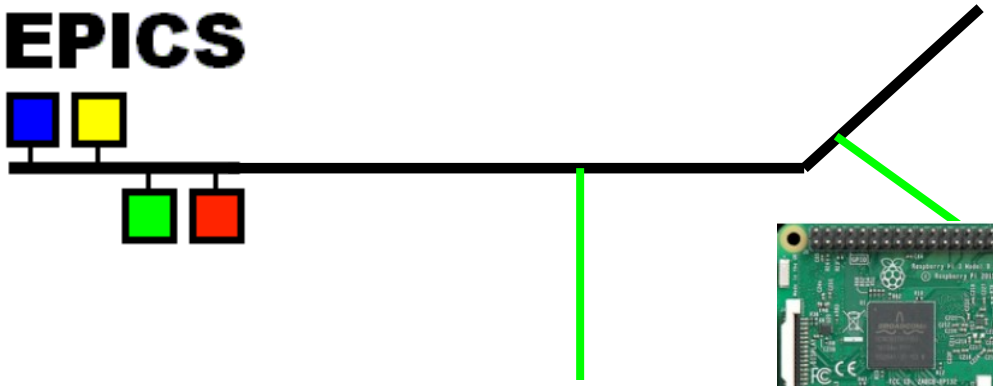
```
record(bi, "$(head):PS_SIM:POWER_SW_RB")
{
  field(DESC, "Read back the power switch")
  field(SCAN, "Passive")
  field(INP, "$(head):PS_SIM:POWER_SW")
  field(ZNAM, "OFF")
  field(ONAM, "ON")
}
```

＜恐れることはありません＞
ここは、（さっきと同じ最も簡単な
bi型信号1個の定義

＜だがしかーし＞
・レコード間の矢印の意味は何？
・レコード型”CALC”って何？
・FLNKって何？
・EGUとかDRVHとかHIHIとか??
なお、VAL はそのレコードの値です

Hands-on: DBの基礎

- 実習 : 資料「EPICS Databaseの基礎」by sasaki



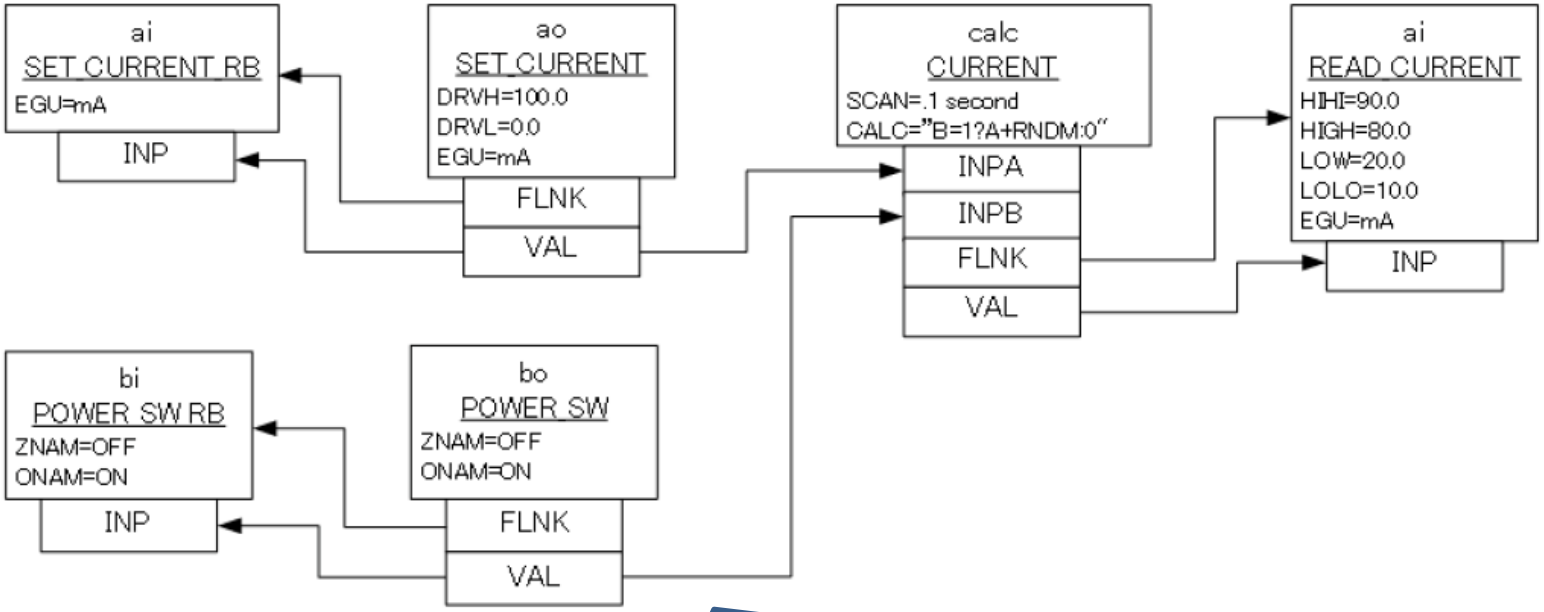
まあ
やってみなはれ

```
$ softIoc -m user=ET_DEMO -d powerSupply.db
Starting iocInit
##############################################
## EPICS R3.15.6
## EPICS Base built Oct 18 2018
##############################################
```

IOC by “softIoc” command

再び What is “database”？ （続き）

- 図を理解しよう: 仮想電源



＜おわかりいただけたでしょうか＞

・信号の流れ： (bo or aoの)VALから -> (bi or aiの)INPへ

・処理の流れ： (あるレコードから)FLNKで->(次のレコード)

Q) 通常時はどういう処理が行われるのか？

Q) 処理の開始点はどこか？

・ao, aiのField(DRVH/DRVL, HIHI/HIGH/LOW/LOLO)の意味は？

ちょっと休憩

ここまでは Soft IOC (real I/O無し)でdbの体験
ここからは RPiでreal I/Oを目指そう

再び What is IOC ?

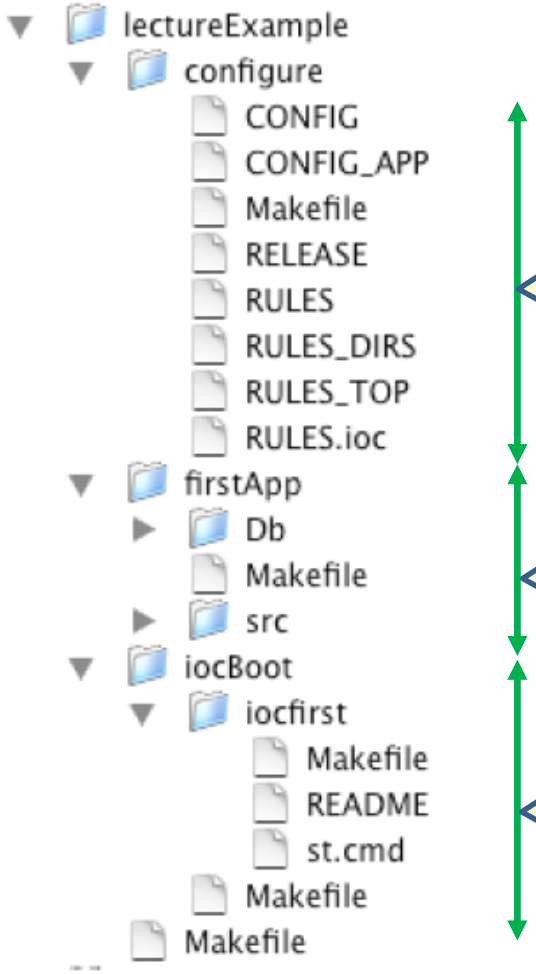
- IOCを作るため
  - “makeBaseApp” command を使います
    - このコマンドはEPICSの核心的部分ですが、最初は理解しにくい ..
    - 実体はperl programで、何を作るかと言えば
      - IOC開発用の Directories -> 次のページ
      - IOCアプリのひな形(application skeleton) (+ sample databases)
  - softIOCと違う点
    - レコードをIOCに「載せる」scriptがある（ないと動かない）
      - defaultではst.cmdという名前 -> 次の次ページ
    - Hardwareを指定する(しない時もある)
      - Field DTYP： PLC型IOCは”F3RP61”、RPi-GPIOは”devgpio”

```
field(DTYP,"F3RP61")
field(INP,"@U0,S$(slot),X$(ch)")
```

```
field(DTYP, "devgpio")
field(OUT,  "@17 H")    # or GPIO17, Active High
```

例) Directory structure created by makeBaseApp

Final <*top*> directory structure



＜恐れることは･･･ありません＞
アプリ名”first”とした時のdir構造

各種環境設定
開発するアプリに応じ、最初に修正する

アプリ”first”の作業エリア
dbファイルなどはここで編集する

IOC “iocfirst”の情報エリア
st.cmd (起動script)はここに作る

再びWhat is “database”？ （続き）

- レコードをiocにのせる(ioc起動script)
  - 例（DigitalOut、DigitalInの信号を定義して動かす）

実装の時、DEFAULTでは、「**st.cmd**」というfile名

(上) **di.db** 1bit input定義

```
record(bi,"$(head):DI:SLOT$(slot):CH$(ch)")
{
        field(DTYP,"F3RP61")
        field(INP,"@U0,S$(slot),X$(ch)")
        field(SCAN,"$(scan)")
        field(ONAM,"On")
        field(ZNAM,"Off")
}
```

(下) **do.db** 1bit output定義

```
record(bo,"$(head):DO:SLOT$(slot):CH$(ch)")
{
        field(DTYP,"F3RP61")
        field(OUT,"@U0,S$(slot),Y$(ch)")
        field(ONAM,"On")
        field(ZNAM,"Off")
}
```

・IOC起動scriptの例 (st.cmd)

slot=2にDOut、slot=3にDin moduleがあって、

各2つ・計4個のレコードを実装する）

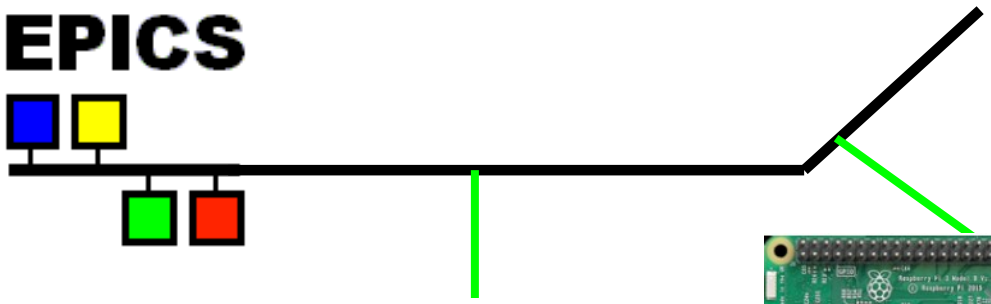
```
## Dec.2012 test by kami
#
## CAUTIION - you are asked to change "your_name"
##            then "./st.cmd"
#
dbLoadRecords("db/do.db", "head=your_name,slot=2,ch=1")
dbLoadRecords("db/do.db", "head=your_name,slot=2,ch=2")
#dbLoadRecords("db/do.db", "head=your_name,slot=2,ch=3")
#dbLoadRecords("db/do.db", "head=your_name,slot=2,ch=4")
dbLoadRecords("db/di.db", "head=your_name,slot=3,ch=1,scan=.5 second")
dbLoadRecords("db/di.db", "head=your_name,slot=3,ch=2,scan=.5 second")
#dbLoadRecords("db/di.db", "head=your_name,slot=4,ch=2,scan=.5 second")
#dbLoadRecords("db/di.db", "head=your_name,slot=5,ch=2,scan=.5 second")
```

- st.cmdでマクロ指定し、各行で1個づつレコードの実体を生成する
- “#”はコメント行

Hands-on: makeBaseApp

- 実習：資料「appl作成：makeBaseApp」by sasaki

最初は理解しにくい、と
言いました・・・
まあ　やればわかるさ
なんくるないさー



```
$ mkdir -p ~/epics/app/work_MyEPICS
$ cd ~/epics/app/work_MyEPICS/
$ makeBaseApp.pl -t example myEpics
$ makeBaseApp.pl -i -t example myEpics
                :
                :
$ chmod +x ./st.cmd
```

IOC by “makeBaseApp” command

再びWhat is “database”？ （続き）

- 実習のioc起動script(st.cmd)を理解しよう

```
#!../../bin/linux-arm/myEpics

< envPaths

cd "${TOP}"

## Register all support components
dbLoadDatabase "dbd/myEpics.dbd"
myEpics_registerRecordDeviceDriver pdbbase

## Load record instances
dbLoadTemplate "db/user.substitutions"
dbLoadRecords "db/dbSubExample.db", "user=kektaro"

cd "${TOP}/iocBoot/${IOC}"
iocInit
```

・../../bin/linux-arm/myEpicsは、IOCのprogramの実態です
・envPaths, myEpics.dbd, user.substitutions などは、editorで中身を見ることが出来ます

例えば $TOP、$IOC など、ここで定義されます

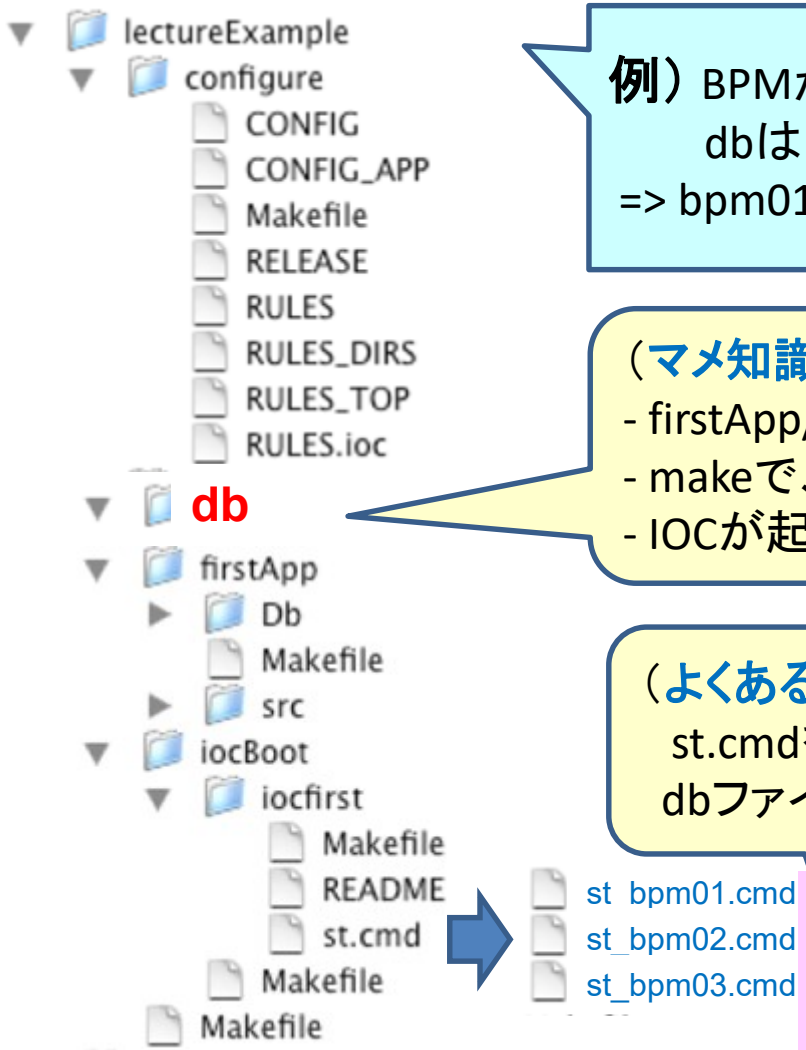
dbdなどこの2行は、このapp用に自動生成され、IOC運用に必要な定義の集合です

この2行で、dbファイルを元にレコードを実装します
user.substitutionsは中身を見てね

myEpicsApp、行くであります　ケロケロ (‘◇’)ゞ

再びWhat is “database”？ （続き）

- 複数のiocがある時（とDirectory構造を再考）



例） BPMが144個あって、IOC1台当たり16個、IOCは計8台
dbは1種類で、144個分はマクロ差し替えで対応
=> bpm01App ～ bpm08App と、同じものをIOC数作るか？

（**マメ知識**） IOC運用時には、db/ が出来ている
- firstApp/Db/ は、db開発時の作業場所
- makeで、作業場所のdbはこのdb/にコピーされる
- IOCが起動する時は、このdb/ のdbファイルを使う

（**よくある策**）
st.cmdを、IOCの数だけ用意する（IOCは同じdirを共有）
dbファイルは同じものを共有

makeBaseAppで作られるdirectory構造は
ややこしくみえるが
運用時に役に立つ構造になっている

これでDatabaseとIOCの話は終わり

最後にまとめページ　。。。

まとめ）DatabaseとIOC

- IOC, CA server, PV, Database, .. その関係