F3RP61 を用いた EPICS 講習会
実習資料

Revision	1.1
Status	RELEASE
Author	KEKB 制御グループ
Last modification	2016 年 10 月 17 日
Created	2016 年 9 月 6 日

Creation	*Date*	*Author*	*Section*	*Modification*
0.0	2015/03/23	廣瀬 雅哉	All	実習書作成

Revision	*Date*	*Author*	*Section*	*Modification*
0.1	2016/09/06	浅野 和哉	All	「実習環境の解説」、 「実習 1」～「実習 3」の変更 「実習 4」～「実習 6」の削除
0.2	2016/09/26	浅野 和哉	3,4,6	開発環境と用語の説明の追加 実習手順の変更 新規実習の追加
0.3	2016/09/29	浅野 和哉	6	新規実習の追加
1.0	2016/10/12	浅野 和哉	All	全体の見直し
1.1	2016/10/17	浅野 和哉	All	全体の見直し

目次

1 EPICS とは

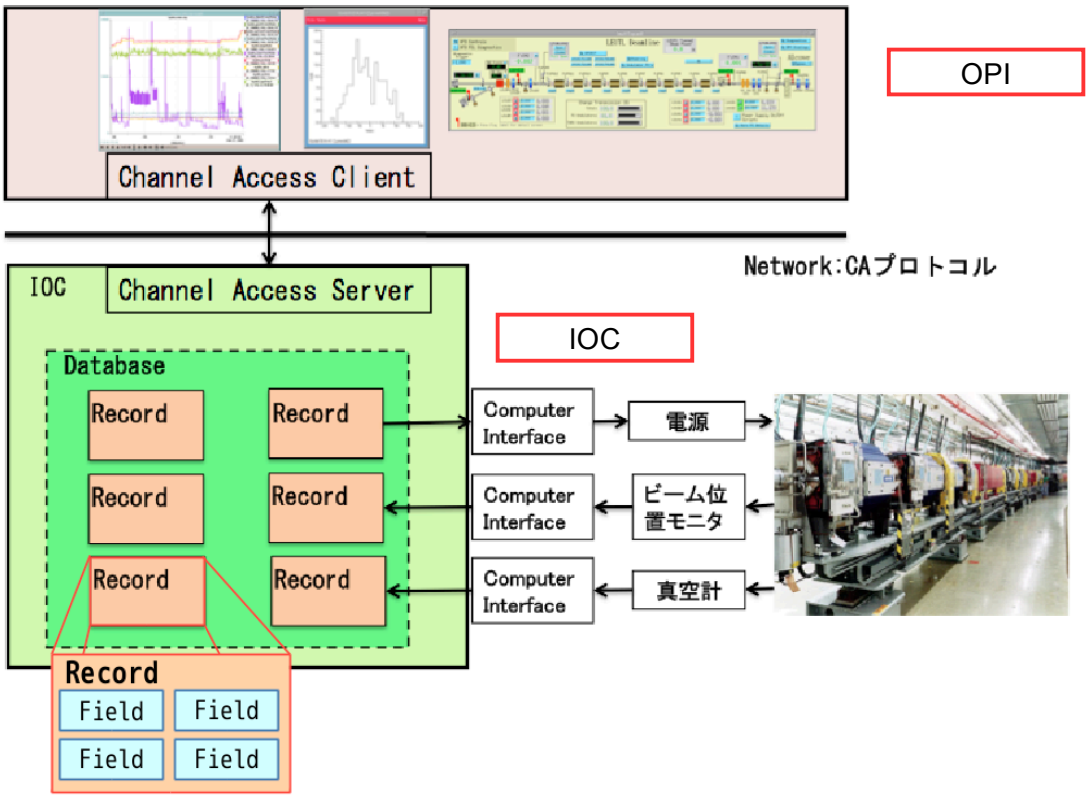
Experimental Physics and Industrial Control System (EPICS)は加速器等の大型実験装置の制御システムを開発･実装するためのソフトウェアです。EPICS の基本的要素は Input / Output Controller(IOC) 、 Operator Interface (OPI) 、 Channel Access(CA)の 3 つです。

IOC は入出力制御計算機であり、EPICS の中で中心的な役割を果たすランタイム･データベースを保持しています。ランタイム･データベースはデバイスを制御するために必要なレコードから構成され、そのレコードの一つ一つは多数のフィールドから構成されています(典型的なフィールドは値を保持する VAL フィールドでありフィールド名を省略した場合は VAL フィールドであると解釈されます)[図 1]。

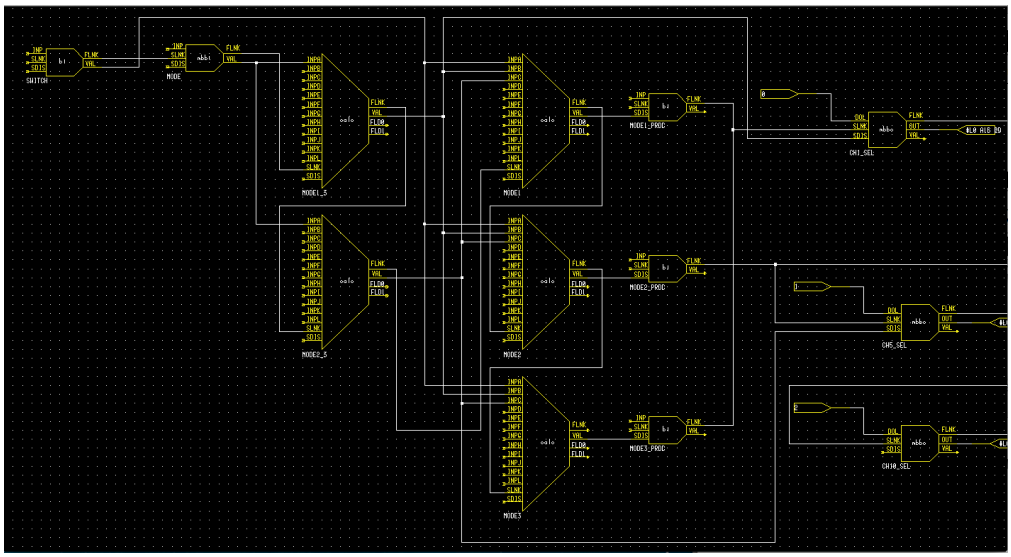
OPI は制御画面、アーカイバ、アラームハンドラなどの上位のソフトウェアを実行する計算機を指します。

CA はクライアントとなる OPI とサーバーとなる IOC の間、又は IOC 同士でデータにアクセスするための EPICS の通信プロトコルです。またランタイム･データベースのレコード間には、リンクを張ることができます。リンクには INPUT リンク、OUTPUT リンク、FORWARD リンクがあり、データの流れ、プロセスの流れを作り出し、電子回路のようなロジックを組む事ができます[図 2]。

<図 1 ： EPICS 制御システム例>



〈図 2 ： EPICS レコード間のリンクでロジックを実装した例〉



2 実習環境の解説

2.1　実習環境(abcob13.kekb.kek.jp)へのログインと環境変数の設定

本実習では、KEKB 制御ネットワーク上にある abcob13 計算機(172.19.42.65)を実習環境として使用します。このマシンにログインするためには、端末を立ち上げ、以下のコマンドを実行してください。ただし、*user* は自身のユーザ名に置き換えてください。以下の”host$”は端末のシェルプロンプトです。

```
host$ ssh -Y user@abcob13.kekb.kek.jp
```

2.2　シェルの確認

自身の使用しているシェルは、以下のコマンドを実行することにより確認できます。

```
abcob13$ echo $SHELL
```

”/bin/bash”と表示されているのであれば、お使いのシェルは bash です。また、csh の場合は”/bin/csh”と表示され、tcsh の場合は”/bin/tcsh”と表示されます。

2.3　環境変数の設定

EPICS に関連するコマンドを実行するために、PATH と EPICS に関する環境変数の設定が必要になります。以下のコマンドを実行し、実習環境向けの設定をしてください。

【bash の場合】

```
abcob13$ source /cont/epics314/app/ET/ET_Lecture_201610/set_path.bash
```

【csh,tcsh の場合】

```
abcob13$ source /cont/epics314/app/ET/ET_Lecture_201610/set_path.csh
```

上記コマンドは、ssh でログインする度に実行する必要があります。この手順を簡略化したい方は、bash の場合、~/.bashrc をエディタで開き、最終行あたりに上記コマンドを追加してください。csh の場合は~/.cshrc に、tcsh の場合は ~/.tcshrc に追記してください。

本実習では複数の端末を使用することになるので、xterm を使用して 3 つほど端末を立ち上げておいてください。

3 IOC(F3RP61)の開発環境

3.1 F3RP61 とは

横河電機(株)が製造、販売する Linux が搭載された CPU モジュールです。単純なデジタル入出力、アナログ入出力モジュールに加え、位置決め制御、温度監視、高速データ収集などを行うモジュールを使用することができます。

F3RP61 の電源を投入し telnet コマンドでログインしてください。
(F3RP61 の 1 号機の場合は■を 1 に、n 号機の場合は n に置き換えてください。)

```
abcob13$ telnet ioccorp61tmp■
```

ログイン名を聞かれるので、root と入力し、Enter キーを押します。
※パスワードはありません。

```
login: root
```

ログインに成功したら、-bash-3.2#というシェルプロンプトが現れます。

3.2 クロスコンパイル

クロスコンパイルとは、ソースコードを解釈し、開発環境とは別の環境で実行可能なプログラムを生成することです。本実習環境では IOC プログラムのコンパイルをホストマシン abcob13 計算機(linux-x86_64)で行いますが、生成された IOC プログラムは F3RP61 で実行します。

3.3 NFS マウントの確認

Network File System(NFS)マウントとは、外部ストレージをネットワーク経由で OS に認識させ、アクセス可能な状態にする事です。F3RP61 のどのディレクトリがホストマシンのどのディレクトリにマウントされているかを確認してみましょう。F3RP61 にログインした状態で以下のコマンドを実行してください。

```
-bash-3.2# df -h
```

【実行結果】

```
Filesystem                      Size      Used    Avail   Use%    Mounted on
/dev/hda1                       3.7G      257M    3.3G    8%      /
tmpfs                           62M       0       62M     0%      /dev/shm
tmpfs                           4.0M      120K    3.9M    3%      /var
tmpfs                           4.0M      0       4.0M    0%      /tmp
sadnas-cont:/sadnas1a_vol1/proj 8.0T      3.6T    4.5T    45%     /mnt/proj
sadnas-cont:/sadnas1a_vol1/users 8.0T     3.6T    4.5T    45%     /mnt/users
```

1 列目はマウント元、6 列目はマウント先を表しています。

3.4　シンボリックリンクの確認

以下に示すように F3RP61 上では、”/mnt/proj”には”/proj”の、”/mnt/users”には”/users”のシンボリックリンクが張られています。

```
-bash-3.2# ls -l /
```

【実行結果】

```
total 1736
drwxr-xr-x  2 root root    4096 Aug 28  2014 bin
lrwxrwxrwx  1 root root      14 Oct 17  2014 cont -> proj/cont/cont
drwxr-xr-x  4 root root    4096 Sep 26 10:30 dev
drwxr-xr-x 31 root root    4096 Sep 26 10:30 etc
-rwxr-xr-x  1 root root    4608 Aug 28  2014 f3rp6x.dtb
drwxr-xr-x  3 root root    4096 Aug 28  2014 home
drwxr-xr-x  9 root root    4096 Aug 28  2014 lib
drwx------  2 root root   16384 Aug 28  2014 lost+found
drwxr-xr-x  4 root root    4096 Aug 28  2014 mnt
drwxr-xr-x  2 root root    4096 May 31  2010 opt
dr-xr-xr-x 57 root root       0 Sep 26 10:29 proc
lrwxrwxrwx  1 root root       8 Oct 17  2014 proj -> mnt/proj
drwxr-xr-x  3 root root    4096 Oct 25  1903 root
drwxr-xr-x  2 root root    4096 Aug 28  2014 sbin
drwxrwxrwt  3 root root      60 Sep 26 10:30 tmp
-rwxr-xr-x  1 root root 1645828 Aug 28  2014 uImage
lrwxrwxrwx  1 root root       9 Oct 17  2014 users -> mnt/users
drwxr-xr-x  8 root root    4096 Aug 28  2014 usr
drwxrwxrwt  8 root root     160 Sep 26 10:29 var
```

前述の NFS マウントと上記のシンボリックリンクにより、F3RP61 上では”/proj”へのアクセスはホストマシンの”sadnas-cont:/sadnas1a_vol1/proj”へのアクセスに、”/users”へのアクセスはホストマシンの”sadnas-cont:/sadnas1a_vol1/users”へのアクセスになります。

ℹ abcob13 計算機と同様に/cont、/proj、/users へのパスのアクセスが可能となるよう整備しています。

3.5　デモ機のモジュール構成

本実習環境で使用するデモ機の I/O モジュールは以下の通りです。

第１スロット　F3RP61 CPU
第２スロット　アナログ入力モジュール（ADC）
第３スロット　アナログ出力モジュール（DAC）
第４スロット　デジタル出力モジュール（DO）
第５スロット　デジタル入力モジュール（DI）

アナログ出力モジュールとアナログ入力モジュールのチャネル 1 同士はループバックされています。

4 実習１：F3RP61 と EPICS を用いた実習

4.1　IOC プログラム開発環境の作成

ホームディレクトリの下に epics という名のディレクトリを作り、その下に app という名のディレクトリを作成してください。そのために、以下のコマンドを実行してください。
（既に同じ名前のディレクトリがある場合は適宜別名で作成してください。）

```
abcob13$ mkdir -p ~/epics/app
```

~/epics/app に移動し、makeBaseApp.pl コマンドで IOC プログラムの開発環境を作成してください。

```
abcob13$ cd ~/epics/app
abcob13$ makeBaseApp.pl -t ioc demo
abcob13$ makeBaseApp.pl -i -t ioc demo
```

実行すると、この IOC はどのアーキテクチャを使用するのかを聞かれます。
linux-f3rp61-r203 と入力し Enter キーを押下してください。

```
The following target architectures are available in base:
    linux-x86_64
    linux-f3rp61
    vxWorks-mv4100-vw67
    vxWorks-mv5500-vw683
    solaris-sparc-gnu
    vxWorks-mv5500-vw551
    freebsd-x86_64
    linux-f3rp61-r203
What architecture do you want to use? linux-f3rp61-r203
```

その後に、この IOC はどのアプリケーションをブートするのかを聞かれます。今回は１つしかアプリケーションがありませんので、そのまま Enter キーを押すだけです。

その後 make してください。

```
abcob13$ make linux-f3rp61-r203
```

4.2　作成した開発環境の確認

この段階で、~/epics/app ディレクトリの下に、以下に示すディレクトリやファイルが生成されたことを ls コマンドや find コマンドを使用して確認してください。

<ディレクトリ階層>

```
app
   |--bin
   |--configure
        |--RELEASE
   |--dbd
   |--demoApp
        |--Db
             |--Makefile
        |-src
             |--Makefile
   |--iocBoot
       |--iocdemo
             |--st.cmd
```

太文字で表記した箇所はユーザが編集するファイルとそれを収めたディレクトリを表しています。(細文字で表記した箇所はビルド・ツールが自動的に生成するディレクトリであり、通常その下にあるファイルをユーザが編集することはありません。) app/demoApp/Db ディレクトリではランタイム･データベースの作成を、app/demoApp/src ディレクトリではソースコードの作成を行います。

app/dbd ディレクトリ内のファイルはユーザが直接編集するものではありませんがデータベースに関わるあらゆる定義が網羅されていますので辞書代わりに使う事ができます。

4.3　app/configure/RELEASE ファイルのコピーと内容の確認

/cont/epics314/app/ET/ET_Lecture_201610/epics/app/configure ディレクトリに修正済みの RELEASE ファイルを用意していますのでコピーしてください。

```
abcob13$ cd ~/epics/app/configure
abcob13$ mv RELEASE RELEASE.org
abcob13$ cp sepics/app/configure/RELEASE ./
abcob13$ make
```

コピーした RELEASE ファイルには以下の 1 行が追記されています。

```
F3RP61=/proj/epics/R314/R31412/modules/R314123-CSA/f3rp61-1.4.0
```

これは、F3RP61 のデバイスサポートがどこにインストールされているかを EPICS ビルドシステムに教えるために必要です。

4.4　app/demoApp/src/Makefile のコピーと内容の確認

次に/cont/epics314/app/ET/ET_Lecture_201610/epics/app/demoApp/src ディレクトリから Makefile をコピーしてください。

```
abcob13$ cd ~/epics/app/demoApp/src
abcob13$ mv Makefile Makefile.org
abcob13$ cp /cont/epics314/app/ET/ET_Lecture_201610/epics/app/demoApp/src/Makefile ./
abcob13$ make
```

続いて Makefile の内容を確認してください。黄色でハイライトした部分がオリジナルから変更した箇所です。

[Makefile]

```
# Build the IOC application
#PROD_IOC = demo
PROD_IOC_linux-f3rp61-r203 = demo
PROD_IOC += $(PROD_IOC_$(T_A))
-------------------一部省略------------------------------
# demo.dbd will be made up from these files:
demo_DBD += base.dbd
demo_DBD += f3rp61.dbd
-------------------一部省略------------------------------
# Finally link to the EPICS Base libraries
demo_LIBS += f3rp61
PROD_LDLIBS += -lm3
USR_LDFLAGS += -L/proj/epics/f3rp6x/BSP_cdrom/f3rp6x_BSP_R203/yokogawa/library

# NOTE: To build SNL programs, SNCSEQ must be defined
# in the <top>/configure/RELEASE file

ifneq ($(SNCSEQ),)
    SNCFLAGS += +r
    demo_DBD += sncExample1.dbd
    demo_SRCS += sncExample1.stt
    demo_DBD += sncExample2.dbd
    demo_SRCS += sncExample2.stt
    demo_DBD += sncExample3.dbd
    demo_SRCS += sncExample3.stt
    demo_DBD += sncExample4.dbd
    demo_SRCS += sncExample4.stt
    demo_LIBS += seq pv
endif

demo_LIBS += $(EPICS_BASE_IOC_LIBS)
#==========================
```

4.5　ランタイム･データベースのサンプルのコピーと修正

本実習で使用する以下のファイルを/cont/epics314/app/ET/ET_Lecture_201610/epics/app/demoApp/Db からコピーしてください。

Makefile
demo1.db
demo2.db
demo2.substitutions
demo3.db
demo3.substitutions
demo4.db
demo5.db

```
abcob13$ cd ~/epics/app/demoApp/Db
abcob13$ mv Makefile Makefile.org
abcob13$ cp /cont/epics314/app/ET/ET_Lecture_201610/epics/app/demoApp/Db/Makefile ./
abcob13$ cp /cont/epics314/app/ET/ET_Lecture_201610/epics/app/demoApp/Db/demo* ./
```

コピーした demo2.substitutions と demo3.substitutions ファイルをエディタで開き”odagiri”と記述されている箇所を自身のユーザ名に置き換えてください。

[Makefile]

```
# databases, templates, substitutions like this
DB += demo1.db
DB += demo2.db
DB += demo2.substitutions
DB += demo3.db
DB += demo3.substitutions
DB += demo4.db
DB += demo5.db
```

[demo1.db] (sncExample1.stt で使用)

```
record(ai, "ET_$(user):LOOPBACK:ADC")
{
        field(SCAN, ".1 second")
        field(DTYP, "F3RP61")
        field(INP, "@U0,S2,A1")
        field(LINR, "LINEAR")
        field(ESLO, "0.0005")
        field(HOPR, "10.0")
        field(LOPR, "-10.0")
#       field(MDEL, "0.1")
}
record(ao, "ET_$(user):LOOPBACK:DAC")
{
        field(DTYP, "F3RP61")
        field(OUT, "@U0,S3,A1")
        field(LINR, "LINEAR")
        field(ESLO, "0.0005")
```

```
        field(HOPR, "10.0")
        field(LOPR, "-10.0")
}
```

[demo2.db] (sncExample1.stt で使用)

```
record(bo, "ET_$(user):OUT_RELAY:$(no)")
{
        field(DTYP, "F3RP61")
        field(OUT, "@U0,S4,Y$(no)")
        field(ONAM, "On")
        field(ZNAM, "Off")
}
```

[demo2.substitutions]

```
file db/demo2.db {
    pattern { user, no }
  { "odagiri", "01" }
-----------省略------------
  { "odagiri", "08" }
}
```

[demo3.db]

```
record(bi, "ET_$(user):IN_RELAY:$(no)")
{
        field(SCAN, ".1 second")
        field(DTYP, "F3RP61")
        field(INP, "@U0,S5,X$(no)")
        field(ONAM, "On")
        field(ZNAM, "Off")
}
```

[demo3.substitutions]

```
file db/demo3.db {
    pattern { user, no }
  { "odagiri", "01" }
 -----------省略------------
  { "odagiri", "32" }
}
```

[demo4.db] (sncExample2.stt で使用)

```
record(ai, "ET_$(user):TARGET_VOLTAGE")
{
}
```

[demo5.db] (sncExample4.stt で使用)

```
record(ai, "ET_$(user):loLimit")
{
}
record(ai, "ET_$(user):hiLimit")
{
}
```

ET_$(user)の ET は、Epics Training の略称であり、名前空間で実運転用の他のレコードと衝突することを避けています(本資料は講習用のため KEKB 用のレコード名の命名規則に完全には則っていません)。正しい命名方法は参考文献[3]を参照してください。

コピーとファイルの修正を終えたら上書き保存し、make してください。ここでの make は app/demoApp/Db から新たに作成された app/db へファイルをコピーするだけです。

```
abcob13$ make
```

app/demoApp/Db は.db ファイルを作成するための工場です。app/db は出庫待ちの完成品を置くための倉庫です。

4.6 st.cmd ファイルのコピーと修正

/cont/epics314/app/ET/ET_Lecture_201610/epics/app/iocBoot/iocdemo ディレクトリに st.cmd ファイルを用意していますのでコピーしてください。

```
abcob13$ cd ~/epics/app/iocBoot/iocdemo
abcob13$ mv st.cmd st.cmd.org
abcob13$ cp /cont/epics314/app/ET/s/epics/app/iocBoot/iocdemo/st.cmd ./
abcob13$ chmod +x st.cmd
```

コピーした後に st.cmd をエディタで開き、”odagiri” と記述されている箇所を自身のユーザ名に置き換えてください。

```
dbLoadRecords("db/demo1.db","user=odagiri")
dbLoadTemplate "db/demo2.substitutions"
dbLoadTemplate "db/demo3.substitutions"
dbLoadRecords("db/demo4.db","user=odagiri")
dbLoadRecords("db/demo5.db","user=odagiri")
```

F3RP61 上で IOC を起動させるため、F3RP61 にログインしている端末で以下のディレクトリへ移動し、./st.cmd で IOC プログラムを実行してください。

```
-bash-3.2# cd /users/user/epics/app/iocBoot/iocdemo
-bash-3.2# ./st.cmd
```

4.7　iocsh コマンドの実行

エラーなく実行されて IOC シェルのプロンプト(epics>)が現れたら以下の IOC シェル・コマンドを試してみてください。ただし、*user* は自身のユーザー名に置き換えてください。

IOC シェル・コマンドの一覧を印字するコマンドとして help コマンドがあります。

```
epics> help
```

help コマンドの使い方は、

```
epics> help <cmd>
```

ここで、<cmd>は help で表示されたコマンド名です。 help コマンドではワイルドカードを使えるので、

```
epics> help db*
```

とすれば、”db”で始まるコマンドの使用方法を見ることができます。

以下に頻繁に使われる 4 つのコマンドを紹介します。

IOC にロードされているレコードの一覧を印字するコマンドは以下の通りです。

```
epics> dbl
```

指定されたレコードのフィールドの値を印字するコマンドは以下の通りです。

```
epics> dbpr ET_user:LOOPBACK:DAC
```

指定されたレコードの VAL フィールドに値 5 を書き込むコマンドは以下の通りです。

```
epics> dbpf ET_user:LOOPBACK:DAC 5
```

指定されたレコードの VAL フィールドから値を読みだすコマンドは以下の通りです。

```
epics> dbgf ET_user:LOOPBACK:ADC
```

epicsThreadShow コマンドを試してみましょう。
IOC プログラムが多数のスレッドからなるプログラムである事が確認できます。

```
epics> epicsThreadShow
```

【実行結果】

```
           NAME             EPICS ID      PTHREAD ID    OSIPR  OSSPRI   STATE
         _main_             0x10047040    0             0      0        OK
         errlog             0x1004c070    1208603824    10     0        OK
         taskwd             0x1004fa90    1208865968    10     0        OK
     timerQueue             0x1005f870    1209390256    70     0        OK
          cbLow             0x100519f8    1210438832    59     0        OK
       cbMedium             0x10051bd0    1211487408    64     0        OK
         cbHigh             0x100a3b60    1212535984    71     0        OK
       dbCaLink             0x100a3e48    1213584560    50     0        OK
f3rp61Seq_mcmd              0x10050410    1213846704    90     0        OK
       scanOnce             0x100a1f60    1214895280    70     0        OK
        scan600             0x100aa690    1215943856    60     0        OK
        scan300             0x100aa808    1216992432    61     0        OK
        scan120             0x100aa980    1218041008    62     0        OK
         scan60             0x100aaaf8    1219089584    63     0        OK
         scan30             0x100aac70    1220138160    64     0        OK
         scan10             0x100aade8    1221186736    65     0        OK
          scan5             0x100aaf60    1222235312    66     0        OK
          scan2             0x100ab0d8    1223283888    67     0        OK
          scan1             0x100ab250    1224332464    68     0        OK
        scan0.5             0x100ab3c8    1225381040    69     0        OK
        scan0.2             0x100ab540    1226429616    70     0        OK
        scan0.1             0x100ab6b8    1227478192    71     0        OK
        CAS-TCP             0x100abb48    1228002480    18     0        OK
     CAS-beacon             0x100abd28    1228264624    17     0        OK
        CAS-UDP             0x100abef8    1228788912    16     0        OK
```

4.8　Channel Access によるレコードへのアクセス

先ほど作成した IOC 上のレコードに、ホストマシン(abcob13)から caget、caput、camonitor コマンドでアクセスします。別端末で以下のコマンドを実行し、DAC、ADC の設定値を確認してください。ただし、*user* は自身のユーザ名に置き換えてください。

DAC の設定値に 5 を設定してください。

```
abcob13$ caput ET_user:LOOPBACK:DAC 5
```

ADC の設定値を読み出してください。

```
abcob13$ caget ET_user:LOOPBACK:ADC
```

DAC と ADC はループバックしているので caget が返す値は caput した値とほとんど同じ値になります。

ADC の設定値をモニタしてください。

```
abcob13$ camonitor ET_user:LOOPBACK:ADC
```

別の端末で DAC の設定値を 10 に設定してください。

```
abcob13$ caput ET_user:LOOPBACK:DAC 10
```

このとき、camonitor による値がどのように変化するのかを確認してください。
最後に、camonitor コマンドを Ctrl+C により終了させてください。

5 実習 2 ： CSS/BOY による操作画面のコピーと実行

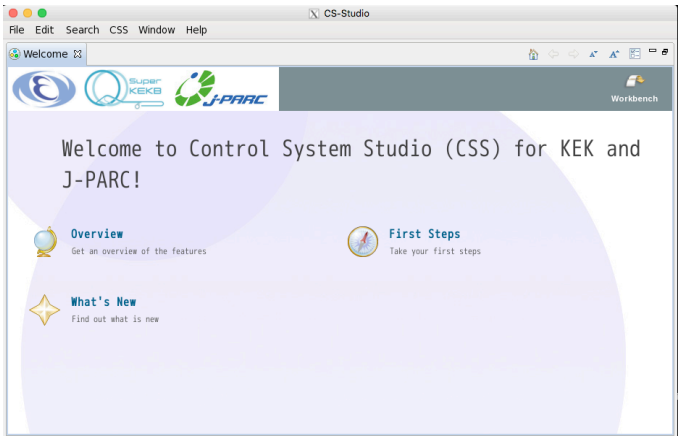
5.1　実習の内容

本実習では、実習 1 で作成した IOC プログラムを操作するためのサンプル画面を実行します。実習 1 で、コマンドで行ったことを GUI により簡単且つ直観的に行えるようにしましょう。

5.2　CSS の起動

別の端末で abcob13 計算機にログインし、以下のコマンドで CSS を起動してください。

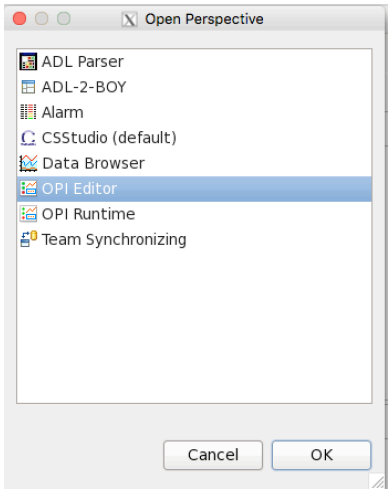
```
abcob13$ css
```



初回起動の場合は Welcome 画面が表示されます。画面右上の Workbench のアイコンを右クリックし、Workbench 画面に切り替えてください。

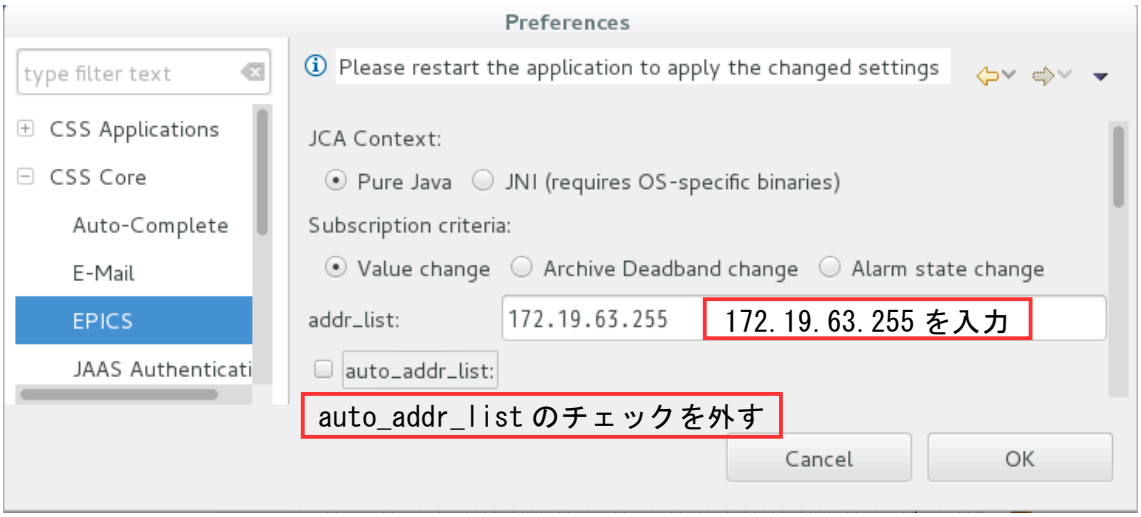
5.3　パースペクティブの切り替え

CSS の左上にあるメニューより Window→Open Perspective→ Other で OPI Editor を選択してください。



5.4　Preferences の設定

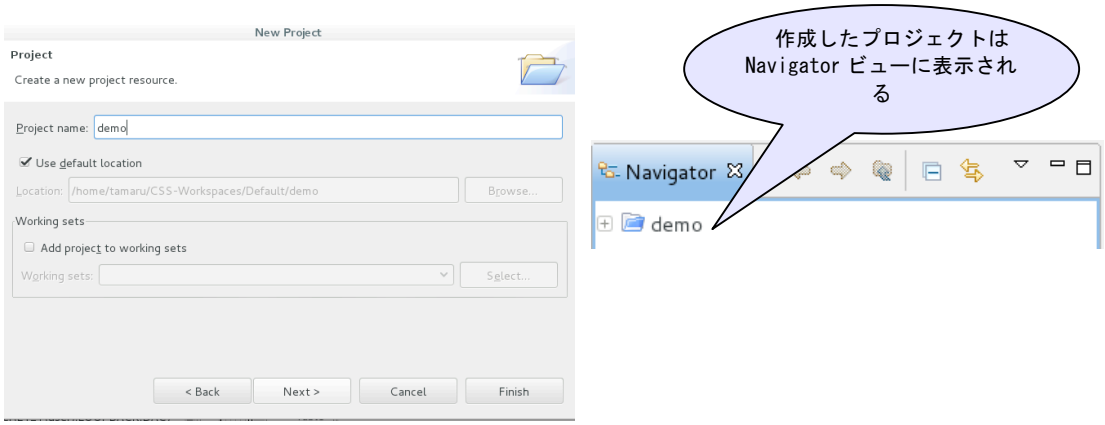
CSS の左上にあるメニューより、作成した IOC プログラム上のレコードにアクセスするため CSS の CA に関する設定を変更します。CSS の左上にあるメニューより Edit→Preferences...→CSSCore→EPICS を選択します。



設定後、CSS のメニューより File→RestartCSS を選択し、CSS を再起動します。

5.5　プロジェクトの作成

CSS のメニューより File→New→General→Project を選択します。プロジェクト名には demo と入力し、Finish を押下してください。



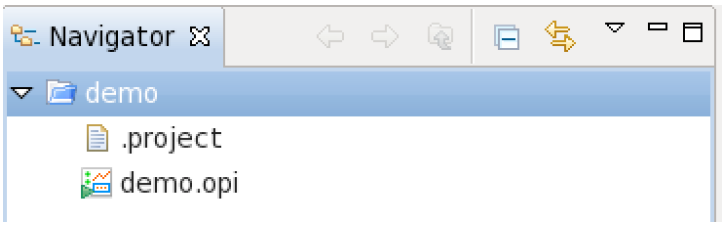
作成したプロジェクトは、Use default location のところに作られます。

5.6　サンプル OPI のコピー

別の端末にて、以下のコマンドを実行し、サンプル OPI ファイルを先ほど作成したプロジェクト内にコピーしてください。

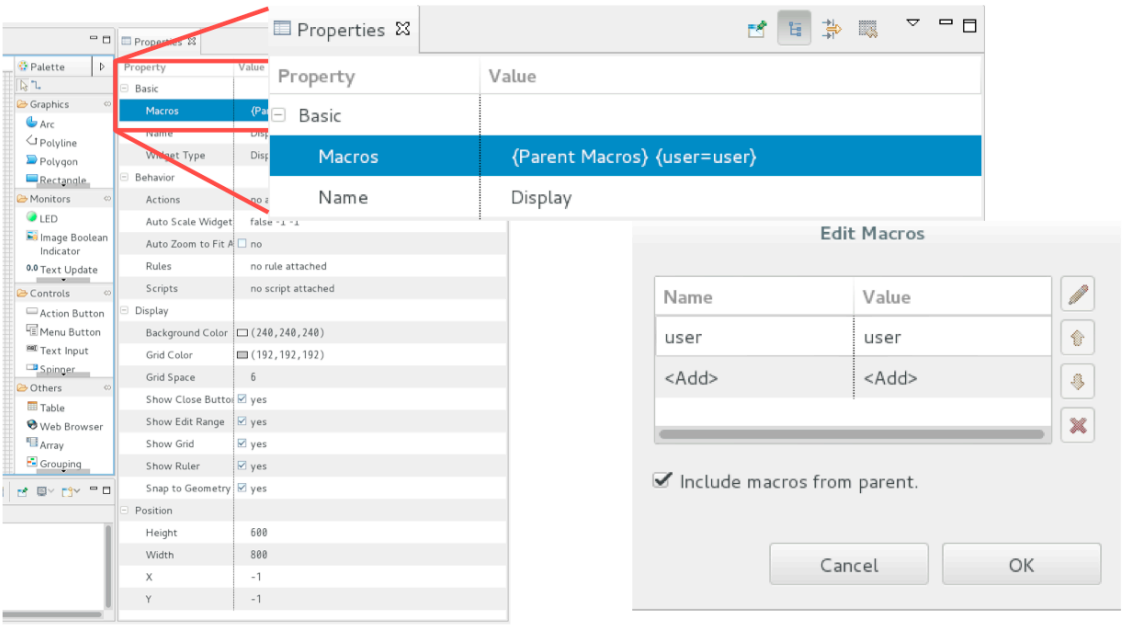
```
abcob13$ cd ~/CSS-Workspaces/Default/demo
abcob13$ cp /cont/epics314/app/ET/ET_Lecture_201610/demo.opi ./
```

コピーし終えたら、Navigator ビューで右クリックコンテキストメニューより Refresh を実行してください。



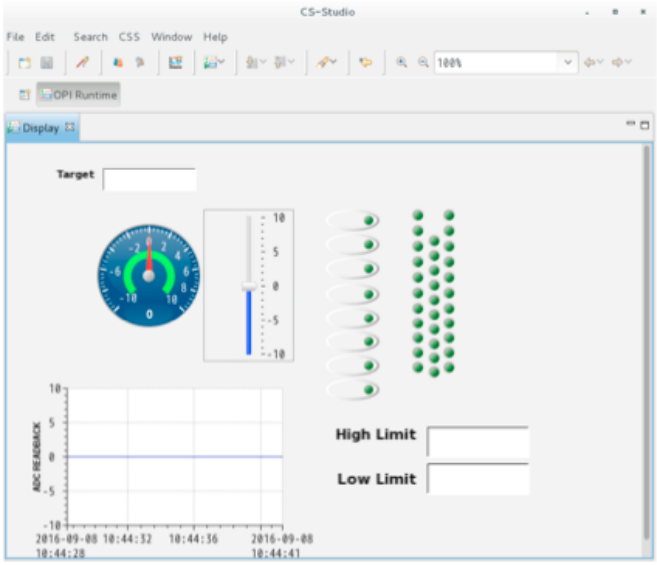
5.7　サンプル OPI の Macro の変更

demo.opi では、ディスプレイの Macro として$(user)が定義されています。この Macro の値は適切に変更する必要があります。Navigator ビューで demo.opi を右クリックし、Open With→OPI Editor を選択します。次に、ディスプレイの Properties ビューより、Basic→Macros プロパティの Value をクリックします。
本実習では、$(user)の値を自身のユーザー名とします。



5.8　サンプル OPI の実行

CSS 画面右上の をクリックし、OPI を実行してください。
Navigator ビューで OPI ファイルを右クリックし、Open With→OPI Runtime を選択しても実行できます。



スライダーはアナログ出力モジュール(スロット３)のチャネル 1 の出力電圧を設定するために使用します。ゲージとその下グラフはアナログ入力モジュール(スロット２)のチャネル 1 の入力電圧を表示します。スライダーを動かして動作を確認してください。

スライダーの右隣の押しボタンはデジタル出力モジュール(スロット４)の各チャンネルに対応しています。ボタンを ON / OFF してモジュール上部にある LED が ON / OFF する事を確認してください。

押しボタンの右隣の LED はデジタル入力モジュール(スロット５)の各チャンネルに対応しています。デジタル入力モジュールに挿入されたスイッチを操作して動作を確認してください。

画面左上、右下のテキスト入力窓は次節で扱う SNL プログラムの実習で使用します。

6 実習 3 ： SNL プログラムの実習

6.1　SNL プログラムとは

SNL とは State Notation Language の略で制御のロジックを状態遷移として記述するプログラム言語です。SNL において重要な概念は状態遷移(ある状態で条件の成立を待ち、その条件が成立した場合に、何らかのアクションを実行して別な状態または同じ状態に遷移すること)です。

6.2　実習の内容

本実習では、SNL プログラムの作成方法について 4 つのサンプルプログラム(sncExample1.stt , sncExample2.stt , sncExample3.stt , sncExample4.stt)を用いて学習します。ここでは実習 1 で用意した IOC プログラムの開発環境とランタイム･データベースを使用します。

6.3　app/configure/RELEASE ファイルの修正

4.2.3 で~/epics/app/configure ディレクトリにコピーした RELEASE ファイルをエディタで開き、以下の一行の”#”を削除して有効にしてください。

```
SNCSEQ=/proj/epics/R314/R31412/modules/R314123-CSA/seq-2.1.16
```

これは EPICS シーケンサ(SNL プログラムを扱うためのライブラリ)がどこにインストールされているかを EPICS ビルドシステムに教えるために必要です。

6.4　サンプルプログラムのコピー

サンプルプログラムを /cont/epics314/app/ET/ET_Lecture_201610/epics/app/demoApp/src ディレクトリに用意していますので~/epics/app/demoApp/src ディレクトリ下に”sncExample”で始まる stt ファイルと dbd ファイルをコピーしてください。

```
abcob13$ cd ~/epics/app/demoApp/src
abcob13$ cp /cont/epics314/app/ET/ET_Lecture_201610/epics/app/demoApp/src/sncExample* ./
```

6.4.1 app/demoApp/src/Makefile の確認

[Makefile]

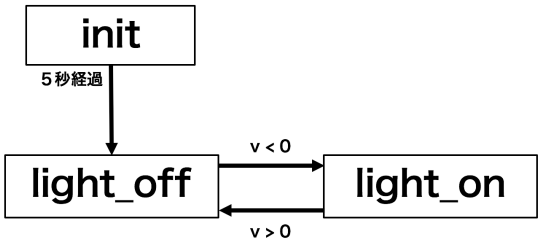
```
---------省略----------------------------------
ifneq ($(SNCSEQ),)
    SNCFLAGS += +r
    demo_DBD += sncExample1.dbd
    demo_SRCS += sncExample1.stt
    demo_DBD += sncExample2.dbd
    demo_SRCS += sncExample2.stt
    demo_DBD += sncExample3.dbd
    demo_SRCS += sncExample3.stt
    demo_DBD += sncExample4.dbd
    demo_SRCS += sncExample4.stt
    demo_LIBS += seq pv
endif
---------省略----------------------------------
```

上記の”SNCFLAGS += +r”は SNL プログラムをリエントラントにする際に必要です。同じ効果を実現するためにソースコードに”+r”を附す方法もあります。

6.4.2 sncExampl1.stt の確認

本プログラムはアナログ入力モジュール(ADC)のチャネル 1 への入力電圧(ET_*user*:LOOPBACK:ADC)をモニタし、負の値の時にデジタル出力モジュール(DO)のチャネル 1(ET_*user*:OUT_RELAY:01)を ON し、正の値の時は同チャネルを OFF するサンプルプログラムです。

[sncExample1.stt の状態遷移図]




```
program sncDemo1
double v;
assign v to "ET_{user}:LOOPBACK:ADC";
monitor v;
short light;
assign light to "ET_{user}:OUT_RELAY:01";

ss ss1 {
 state init{
```

```
        when (delay(5)) {
                printf("sncDemo1: Startup delay over¥n");
        } state light_off
 }

 state light_off {
        when (v < 0) {
                printf("sncDemo1: turning light on¥n");
                light = TRUE;
                pvPut(light);
        } state light_on
 }

 state light_on {
        when (v > 0) {
                printf("sncDemo1: turning light off¥n");
                light = FALSE;
                pvPut(light);
        } state light_off
 }
}
```

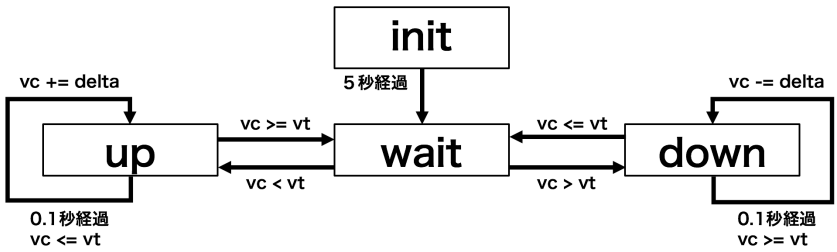
[sncExample1.dbd]

```
registrar(sncDemo1Registrar)
```

6.4.3 sncExampl2.stt の確認

本プログラムはアナログ出力モジュール(DAC)のチャネル 1 からの出力電圧(ET_*user*:LOOPBACK:DAC) と目標電圧値を設定するソフトレコード(ET_*user*:TARGET_VOLTAGE)をモニタし、出力電圧値が目標電圧値に近づくように値を ramp するサンプルプログラムです。

[sncExample2.stt の状態遷移図]




```
program sncDemo2
double vc;
double vt;
double delta;
assign vc to "ET_{user}:LOOPBACK:DAC";
```

```
assign vt to "ET_{user}:TARGET_VOLTAGE";
monitor vt;
monitor vc;

ss ss1 {
 state init {
        when (delay(5)) {
               printf("sncDemo2: Startup delay over¥n");
               vc = 0.0;
               vt = 0.0;
               delta = 0.1;
               pvPut(vc);
               pvPut(vt);
        } state wait
 }

 state wait {
        when (vc < vt) {
               printf("sncDemo2: getting into ramping up¥n");
        } state up
        when (vc > vt) {
               printf("sncDemo2: getting into ramping down¥n");
        } state down
 }

 state up {
        when (vc >= vt) {
               printf("sncDemo2: getting back to wait from up¥n");
        } state wait
        when (delay(0.1)) {
               if (vc + delta > vt) vc = vt;
               else vc += delta;
               pvPut(vc);
               printf("sncDemo2: ramping up...¥n");
        } state up
 }

 state down {
        when (vc <= vt) {
               printf("sncDemo2: getting back to wait from down¥n");
        } state wait
        when (delay(0.1)) {
               if (vc - delta < vt) vc = vt;
               else vc -= delta;
               printf("sncDemo2: ramping down...¥n");
               pvPut(vc);
        } state down
 }
}
```

[sncExample2.dbd]

```
registrar(sncDemo2Registrar)
```

6.4.4 sncExampl3.stt の確認

本プログラムは SNL をエスケープし C 言語のコードを挿入するサンプルプログラムです。sncExample2.stt に現在時刻を取得する C 言語のコードを挿入しています。状態遷移図については sncExample2.stt に同じです。

```
program sncDemo3
double vc;
double vt;
double delta;
assign vc to "ET_{user}:LOOPBACK:DAC";
assign vt to "ET_{user}:TARGET_VOLTAGE";
monitor vt;
monitor vc;

%% #include <sys/time.h>

ss ss1 {
   state init {
        when (delay(3)) {
               printf("sncDemo3: Startup delay over¥n");
               vc = 0.0;
               vt = 0.0;
               delta = 0.1;
               pvPut(vc);
               pvPut(vt);
        } state wait
    }

    state wait {
        when (vc < vt) {
               printf("sncDemo3: getting into ramping up¥n");
        } state up
        when (vc > vt) {
               printf("sncDemo3: getting into ramping down¥n");
        } state down
    }

    state up {
        when (vc >= vt) {
                      printf("sncDemo3: getting back to wait from up¥n");

        } state wait
        when (delay(0.1)) {
               if (vc + delta > vt) vc = vt;
               else vc += delta;
               pvPut(vc);
                      printf("sncDemo3: ramping up...¥n");
               %{
                      struct timeval tv;
                      gettimeofday(&tv, NULL);
               }%
```

```
                printf("%ld %06lu¥n", tv.tv_sec, tv.tv_usec);
        } state up
    }

    state down {
        when (vc <= vt) {
                        printf("sncDemo3: getting back to wait from down¥n");

        } state wait
        when (delay(0.1)) {
                if (vc - delta < vt) vc = vt;
                else vc -= delta;
                pvPut(vc);
                        printf("sncDemo3: ramping down...¥n");
                %{
                        struct timeval tv;
                        gettimeofday(&tv, NULL);
                }%
                printf("%ld %06lu¥n", tv.tv_sec, tv.tv_usec);
        } state down
    }
}
```

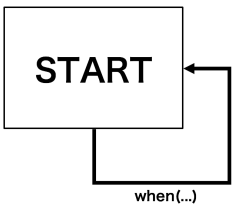
[sncExample3.dbd]

```
registrar(sncDemo3Registrar)
```

6.4.5 sncExampl4.stt の確認

本プログラムはイベントフラグ(evflag)を用いたサンプルプログラムです。時間に余裕がある場合は試してみましょう(ソースコードと実行結果から制御ロジックを読み取ってください)。

[sncExample4.stt 状態遷移図]




```
program sncDemo4

double loLimit;
assign loLimit to "ET_{user}:loLimit";
monitor loLimit;
evflag loFlag;
sync loLimit loFlag;
```

```
double hiLimit;
assign hiLimit to "ET_{user}:hiLimit";
monitor hiLimit;
evflag hiFlag;
sync hiLimit hiFlag;

ss limit {
   state START{
         when ( efTestAndClear( loFlag ) && loLimit > hiLimit ) {
         hiLimit = loLimit;
                pvPut ( hiLimit );
                printf("sncDemo4: Changing hiLimit to %.3lf¥n", hiLimit);
         } state START

         when ( efTestAndClear( hiFlag ) && hiLimit < loLimit ) {
         loLimit = hiLimit;
                pvPut ( loLimit );
                printf("sncDemo4: Changing loLimit to %.3lf¥n", loLimit);
         } state START
   }
}
```

[sncExample4.dbd]

```
registrar(sncDemo4Registrar)
```

6.5 IOC プログラムのビルド

~/epics/app/ディレクトリに移動し make してください。

```
abcob13$ cd ~/epics/app/
abcob13$ make clean
abcob13$ make
```

6.6 IOC プログラムの実行

~/epics/app/iocBoot/iocdemo ディレクトリにある st.cmd をエディタで開き、”#seq sncDemo...”と記述されている行のうち、実行したい SNL プログラムの”#”を削除し有効にしてください。*また、user* は自身のユーザ名に変更し、IOC プログラムを再起動してください。

```
seq sncDemo1, "user=user"
#seq sncDemo2, "user=user"
#seq sncDemo3, "user=user"
#seq sncDemo4, "user=user"
```

iocsh 上に現れるメッセージと CSS/BOY の画面およびソースコードから正しく動作していることを確認してください。

7 付録

ここでは、SuperKEKB における EPICS IOC プログラムの開発環境に関する情報をまとめます。

EPICS version

R3.14.12.3(compactSA 対応版)

EPICS base

/proj/epics/R314/R31412/base-3.14.12.3-CSA

EPICS module

/proj/epics/R314/R31412/modules/R314123-CSA

EPICS アプリケーションのベースディレクトリ

/cont/epics314/app/KEKB

- グループディレクトリ(例:BT、MG、RF、VA、CO 等)
  - 用途別サブディレクトリ(各グループのポリシーによる)

8 参考文献

[1] EPICS Input/Output Controller Application Developer’s Guide:
http://www.aps.anl.gov/epics/base/R3-14/12-docs/AppDevGuide/
[2] EPICS Record Reference Manual:
http://www.aps.anl.gov/epics/wiki/index.php/RRM_3-14
[3] KEKB 制御ネットワークにおけるレコード命名規則:
http://kekb-co-web.kek.jp/top_contents/RecordName.htm
[4] CSS 初心者向け講習会:
http://www-linac.kek.jp/cont/epics/css/CSS_beginners_20120208.pdf
http://www-linac.kek.jp/cont/epics/css/CSS_beginners_20120215.pdf
[5] State Notation Language and Sequencer:
http://www-csr.bessy.de/control/SoftDist/sequencer/
[6] EPICS Device and Driver Support for Yokogawa's F3RP61:
http://www-linac.kek.jp/cont/epics/f3rp61/