



Notes for EPICS Training at PSI



To register for an EPICS Training have a look at
<http://epics.web.psi.ch/training/>
or send a mail to
elke.zimoch@psi.ch

Table of Contents

Table of Contents	2
An EPICS Dictionary.....	3
Linux Commands.....	4
What is Linux?.....	4
For what do we use Linux in an EPICS Training?	4
A Short List of Useful Commands.....	5
CA Commands.....	6
caget	6
caput.....	6
cado.....	6
cainfo.....	6
camon.....	7
Help and Options	8
PSI Environment Commands.....	9
swit.....	9
bootinfo.....	9
externalLinks.....	10
findrecord.....	10
rmc	10
Naming Convention at SLS	11
General statements	11
Example Record Names at SLS.....	12
Alarms and their Colors	13
NO_ALARM	13
MINOR	13
MAJOR.....	13
INVALID.....	14
not Connected	14
Howto caQtDM at PSI.....	15
How to Develop	15
How to Start Panels.....	15
How to convert medm panels	15
caQtDM on Windows	15
Installation of caQtDM	15
More Documentation	15
Cooler Exercise.....	16
Problem description	16
Records needed	16
The substitution file	17
The template file	17
Develop the exercise further	19
Useful things to know about EPICS	20
Ten really neat Things about EPICS.....	20

An EPICS Dictionary

Many technical terms are used in the EPICS community. For most of them the meaning is not clear for non-specialists. The following table will explain some of the terms.

Channel	This is another term for Process Variable. More specific: it is the term for a Process Variable that is connected to a client program.
Channel Access	Channel Access is the communication protocol used by EPICS to transfer information through the network.
Channel Access Server	Software which implements Process Variables for the use with Channel Access. An IOC is also a Channel Access Server.
Channel Access Client	Software which allows access to Process Variables using Channel Access. For example <code>caQtDM</code> is a Channel Access Client.
EPICS	Stands for Experimental Physics and Industrial Control System .
Input	Inside an EPICS environment, input means that information comes from the hardware to the user (or more technically: from the Channel Access Server to the Channel Access Client). Therefore, all measured values and all read back values are input values.
Input Output Controller (IOC)	An IOC is a computer running <code>iocCore</code> , a software package from the EPICS base release. It defines records to implement Process Variables. An IOC is as well a Channel Access Server as a Channel Access Client. This implies that IOCs can communicate with each other.
Output	Inside an EPICS environment, output means that information goes from the user to the hardware (or more technically: from the Channel Access Client to the Channel Access Server). Therefore, all set values are output values.
Process Variable (PV)	A PV is a piece of data which can be accessed by its unique name. Often the PV name consists of a record name and the name of a record field, connected by a dot. For example <code>AIRDI-PCT:CURRENT.EGU</code> . A PV is the basic data set of the Channel Access protocol.
Record	A Record is a collection of Process Variables which is implemented on an IOC. Dozens of Record Types exist, each with its own set of fields and functionality. The most common types are included in the EPICS base distribution. Documentation on record types and their fields can be found in the Record Reference Manual.
Soft IOC	A computer (most often a PC) which is running <code>iocCore</code> not exclusively. In addition, there might be no hardware attached to the computer. At PSI, VME computers are called IOC and PCs are called SoftIOC if they run EPICS.
VME	VME is not an EPICS term. It stands for VERSA Module Eurocard , which is a computer architecture. Unlike a PC, all cards are on a common bus, called VME bus.

Linux Commands

What is Linux?

- Linux was programmed by Linus Torvalds modeled on UNIX.
- Linux is an UNIX like operating system, which is freely available. Therefore, it is a competitor of Microsoft Windows.
- Linux and many programs running on Linux, are Open Source. This means, they are released under the GNU Public License (GPL), which ensures that free software will stay free. For example, the sources of the programs are available and can be changed by anyone.
- Linux can be installed on all usual PCs. Only very new or exotic hardware may cause trouble finding a driver.

For what do we use Linux in an EPICS Training?

We use Linux PCs as terminals, to develop EPICS databases, to create graphical user interfaces, and to show what is going on at the VME computers. The Linux operating system is used in the SLS control rooms and at the beamlines as well. Therefore it is the environment which you will encounter if you use EPICS at the SLS.

As we are using only a few commands, it is not necessary that you are familiar with Linux. The needed commands are listed in table 1. All other programs will be explained and are not standard Linux commands.

We will use a shell to insert commands which are then executed directly. Most of the commands are abbreviations (like in UNIX). Sometimes, you may think that the Linux developers are unable to remember commands with more than five characters. The most common commands are shorter.

The shell is case sensitive. All commands and EPICS names have to be entered the same way as shown. In general commands are in lower case. At the SLS, most EPICS names are in upper case.

A Short List of Useful Commands

- cd** ”change **d**irectory” When this command is entered without argument, it will change the directory to the home directory of the user. If a directory name is entered after the command, it will change to this directory. To go one directory up use ”cd ..”.
- Example: `cd /G/TRAINING/T1`
- cp** ”**c**opy” Copies a file to another destination or copy it with another name. The first argument is the original, the second is the new destination or name.
- Example: `cp MTEST-VME-T1 startup.script test`
- This command makes a copy from the original file MTEST-VME-T1 startup.script and names the copy test.
- ll** ”long list” This is a short cut of the command ”`ls -l`”, which lists the contents of a directory in a detailed way. The command needs no arguments. A line of the output looks like
- ```
-rw-rw-r-- 1 zimoch e sls 686 Mar 30 08:29 startup.script
```
- The first ten characters show the file access permissions (and if it is a file or a directory), then the owner of the file and the group the owner belongs to are displayed. The number 686 in the example line is the file size in bytes. Then the date of the last change is shown and finally the name of the file.
- pwd**          ”**p**rint **w**orking **d**irectory” Shows the full name of the current directory. The command takes no argument.
- rm**           ”**r**emove” This will delete the file given as argument. Attention: The file is gone after this command! There is no trashcan where you can get it back if you accidentally deleted it. Think twice! Example: `rm test`
- nedit &**       One of the many editors available on Linux. It is easy to use and therefore will be used in the EPICS Training. If you give a file name after the command, this file will be opened in the editor. Otherwise, if you do not specify a file, a blank editor is opened and you can choose a file with the menu File (and then Open....).
- Example: `nedit G TRAINING T1 COOLER.template &`

## CA Commands

The following commands can be used in the command line of a shell. On a Linux computer in the SLS network, you can get and set values of EPICS records:

```
caget <RecordName>
caput <RecordName> <Value>
cado <RecordName>
cainfo <RecordName>
camon <RecordName>
```

Due to channel access gateways, records values are read-only if you connect to them from the office net. Additionally, you can write beamline records only if you are in the beamline network. These rules are applied for security reasons.

### **caget**

**caget** - read a record value. The formatted value is printed with the unit appended. In case of an alarm condition, severity and status are appended as well.

```
zimoche@pc4859[/work] ~ > caget ARIDI-PCT:CURRENT
ARIDI-PCT:CURRENT 301 mA
zimoche@pc4859[/work] ~ >
```

### **caput**

**caput** - sets a record to a given value. The value should have the correct type (number, characters, ...) and should be given without unit. The command will replay the readback value of the records after the write has finished. This is an easy possibility to check if your command was successful.

```
zimoche@pc4859[/work] ~ > caput V13BL-OP-SH1:SIZE-SET 5
V13BL-OP-SH1:SIZE-SET 5.000 mm
zimoche@pc4859[/work] ~ >
```

### **cado**

**cado** - sets a record to one. This can be used for easy start sequence, fanout or subroutine records. The command will not print any output to the command line.

```
zimoche@pc4859[/work] ~ > cado V13BL-OP-SH1:STOP
zimoche@pc4859[/work] ~ >
```

### **cainfo**

**cainfo** - shows a collection of record field values. The fields depend on the record type, but some basic information is always given. For example, you can see on which IOC the record is running. Additionally, the command can be used to check the possible values

for an enumeration field. For example you can use the command on the SCAN field of a record.

```
zimoche@pc4859[/work] ~ > cainfo ARIDI-PCT:CURRENT
ARIDI-PCT:CURRENT:
NAME ARIDI-PCT:CURRENT
VAL 300.96249157
EGU mA
SIZE 1
SEVR NO_ALARM
STAT NO_ALARM
TIME {07/23/04 13:21:18.695144729}
TYPE DBF_DOUBLE
HOPR 500
LOPR 0
DRVH 500
DRVL 0
HIHI 0
HI 0
LOLO 0
LO 0
PREC 0
ACCESS R
IOC slscag02.psi.ch
RTYP ai
DESC ""
zimoche@pc4859[/work] ~ >
```

### **camon**

**camon** - sets a monitor to a record. As soon as the value of the record changes, the new value will be displayed. You can cancel this command by <CTRL><C>.

```
zimoche@pc4859[/work] ~ > camon ARIDI-PCT:CURRENT
ARIDI-PCT:CURRENT 301 mA
ARIDI-PCT:CURRENT 300 mA
ARIDI-PCT:CURRENT 301 mA
ARIDI-PCT:CURRENT 300 mA
ARIDI-PCT:CURRENT 301 mA
ARIDI-PCT:CURRENT 300 mA
ARIDI-PCT:CURRENT 301 mA
...
```

## **Help and Options**

If you type one of the CA-Commands without an argument, you will get an Online Help like this:

```
zimoche@pc4859[/work] ~ > caget
usage: caget [flags] <channel> [<channel> ...]
cagets [flags] <channel> [<channel> ...]
caput [flags] <channel> <value> [<channel> <value> ...]
caputq [flags] <channel> <value> [<channel> <value> ...]
cainfo [flags] <channel> [<channel> ...]
camon [flags] <channel> [<channel> ...]
cado [flags] <channel> [<channel> ...]
cawait [flags] <channel> '<condition>' [<channel>
'<condition>'...]
caget reads and formats values from channels (arrays too)
cagets writes 1 to .PROC and reads after processing has
finished
caput writes, waits until processing finishes and reads
back
caputq writes but does not wait for processing
cainfo reads additional information
camon starts monitors (terminate with CTRL-C)
cado writes 1 but does not wait for processing
cawait waits until any condition ('>4.3', '!3...5', etc)
matches
accepted flags:
-date add record execution date
-localdate add host date
-time add record execution time
-localtime add host time
-noname don't add channel name
-nounit don't add units
-stat always add severity and status
-nostat never add severity and status
-hex show integer values as hex
-prec <digits> override the PREC field
-timeout <sec> timeout cawait after <sec> seconds
-version print version and exit
-help print this help text and exit
zimoche@pc4859[/work] ~ >
```



## PSI Environment Commands

On the Linux PCs at the SLS, there are some useful scripts available. They are used like commands, i.e. they are typed into the command line of a terminal. Like all Linux commands, they are case sensitive. The command can have options, separated by blanks. The programs listed here are all installed in the /prod and /work directory. So they can be used where at least one of these directories is available. This is true for the machine network, all beamlines and the offices in the SLS building (WSLA). If you are somewhere else, try to login to slslc03 which is the main development computer of the SLS.

### **swit**

Installs IOC Applications from the working directory or from the CVS repository to \$INSTBASE. You need this command to install the boot directory of your IOC. To use swit you have to be in the directory where the substitution and template files are located which you want to install. For the EPICS Training, this is the directory ~/G/TRAINING/T1 for the first Training IOC.

A short description of available options and some examples can be obtained by using the -h option: `swit -h`

Example:

```
swit -V -ioc MTEST-VME-T1
```

Find more information on the web page

<http://city.psi.ch/city/index.php/swit/index.php>

### **bootinfo**

Gives you information about the last boot process of your IOC. For example the time and date and the EPICS version loaded. It is useful to check if the situation is still the same or simply to look what happened during the last night.

It connects to an Oracle database and should be independent of the EPICS environment you are in. Several options are possible. To get more documentation use the command with the -h option: `bootinfo -h`

General the syntax of the command is `bootinfo [options] [pattern]`

Where pattern is an IOC name or a part of an IOC name. If you give only a part of the name, information about all systems which include the part are displayed.

Example:

```
bootinfo MTEST-VME-T1
```

## ***externalLinks***

This command is used for a substitution file. All record names which are used in links, but not defined in this file are listed. This is very helpful to find typos in record names. Additionally, the syntax of the whole substitution/template construct is tested. If `externalLinks` fails, the IOC will fail during reboot as well. This helps to find missing brackets and quotes, and wrong field names.

The command has to be used in the directory where the substitution and template files are available. For the EPICS Training this would be the directory `~/G/TRAINING/T1` for the first Training IOC.

Example:

```
externalLinks MTEST-VME-T1 example.subs
```

## ***findrecord***

Sometimes, you may only remember a part of a record name. To find the whole name you can use the `findrecord` script. The command searches an Oracle database that is updated during every IOC reboot. Therefore, you can only find actual names.

The command can be used in any directory and should be independent of the EPICS environment you are in. General the syntax of the command is

```
findrecord [options] [pattern]
```

Where `pattern` is the part of the record name you still remember. You can use wildcards as well: `%` stands for an arbitrary number of characters, `^` stands exactly for one character. Several options are possible. To get more documentation use the command with the `-h` option: `findrecord -h`

Example:

```
findrecord MTRT1
```

## ***rmc***

The abbreviation is for remote minicom. The command opens a direct serial (minicom) connection to the Debug port of the VME IOC. A new window will open and here you can use EPICS and VxWorks commands to do diagnosis of your system. You can connect only to VME IOCs in the same network. Depending on the network you might be asked to enter the `slsop` password or to give the user name and password of a valid afs user.

Example:

```
rmc MTEST-VME-T1
```

## Naming Convention at SLS

EPICS records need unique names. This can only be achieved with a naming convention which is mandatory for all record names. The rules of the convention differ from institute to institute, there is no global agreement. Inside PSI there is a slightly different convention for each facility.

At the SLS, a record name consists of five parts:

[1][2] – [3] – [4] : [5]

1. Kingdom (1 character):  
A=Accelerator, X=Beamline, I=Infrastructure, M=Test
2. Domain (4 characters):  
Domain is a functional region with slightly different structure depending on the kingdom:
  - All accelerators (Linac, Ring, Booster, etc.) share the same subsystems (magnets, vacuum, diagnostics etc.). Thus domain is a cell of a system matrix with the first pair of chars identifying the accelerator (LI, RI, BO...) and the second pair the technical subsystem (MA, VA, DI...)
  - In case of the experiment the identifier consists of two numbers for the sector, a character for the source and another one to distinguish different beamlines on the same light source, e.g. 04SA.
3. Device (Optional):  
Since different systems have different topologies, the 3rd and 4th field of the name string have different meanings.
4. Component:  
Since different systems have different topologies, the 3rd and 4th field of the name string have different meanings.
5. Function (Optional):  
An arbitrary description of the function the record provides.

### **General statements**

- Delimiter: The dash (–) (ASCII 45) was chosen in order to avoid conflicts with software to be used for dealing with names: Dot (.) and colon (: ) would cause conflicts with EPICS, slash (/) with MS-EXCEL, underscore ( ) was found to be ugly, backslash (\) inconvenient to type, others ( , ; ^ > = + ) too unfamiliar.
- No preference for upper or lower case characters is made.
- The total maximum length of the name string is 27 characters incl. delimiters. This is a rule specified by EPICS (up to version 3.14).
- It has proven to be useful to check other record names in the field where you are developing new ones. There are standards which evolve from the use of the naming convention in similar devices. These local standards should be followed.
- If there is a system which should be switched to EPICS, it is a good idea to stick to old names as far as possible, to make the change easier.

## **Example Record Names at SLS**

The examples listed here are chosen randomly.

- **ARIDI-PCT:CURRENT**
  1. Kingdom: A - the record belongs to the accelerator
  2. Domain: RIDI - RI for ring, DI for Diagnose
  3. Device: not used
  4. Component: PCT - device type for current measurement
  5. Function:CURRENT - self explained

The records ARIDI-PCT:CURRENT reads the actual beam current of the accelerator ring from a PCT device.
- **X10SA-ID-GAP:SET**
  1. Kingdom: X - the record belongs to a beamline
  2. Domain: 10SA - name of the beamline (sector ten (10), short straight (S), first beamline (A))
  3. Device: ID - device group is insertion device
  4. Component: GAP - the component is the gap of the insertion device
  5. Function: SET - the function is, to set the value

With the record X10SA-ID-GAP:SET a user can set a value for the gap of the insertion device in sector ten (which is a small straight section), which delivers beam for the first beamline of this section.
- **ALIRF-A2-KLY:COMP**
  1. Kingdom: A - the record belongs to the accelerator
  2. Domain: LIRF - LI for Linac, RF for radio frequency
  3. Device: A2 - there are two RF stations, this belongs to the second one
  4. Component: KLY - the RF device is the klystron
  5. Function:COMP - this compares set and readback value

The record ALIRF-A2-KLY:COMP compares two values of the klystron in RF station A2 in the Linac.

## Alarms and their Colors

An EPICS record is always in one of five "alarm" states:

| Status        | Color             |
|---------------|-------------------|
| NO ALARM      | Green             |
| MINOR         | Yellow            |
| MAJOR         | Red               |
| INVALID       | White             |
| not Connected | White box or Pink |

These states can be monitored for example by the Alarmhandler (CA-Client). Standard caQtDM user interfaces as well display them quite often. To ensure consistency it is important to stick to the same colors as used now.

You should always check the alarm state of a record in your own programs and react accordingly. The record value alone will not be enough to differ between "good" and "bad" values.

Because EPICS already provides the possibility to enter limits for the alarms in the record, you should take advantage from this. If you put the limits into the client programs and do the comparison yourself, all other programs, for example command line ca-commands, will not show the alarms. On the other hand, using the record makes later changes easier, because you only have to change at one point.

### ***NO\_ALARM***

This should be the normal alarm state. It indicated that everything is ok. If this should be displayed, use the color green.

### ***MINOR***

This alarm state indicates that a first limit has been violated. In the record, you can specify an upper and a lower limit: for example for AI and AO record, you have to configure the fields HIGH and LOW (and additionally the fields HSV and LSV have to be set to "MINOR").

A minor alarm is a warning to the operator that something might be a problem. If you display this warning, use yellow.

### ***MAJOR***

A major alarm is a real alarm and indicates an error. In the record, you can specify an upper and a lower limit: for example for AI and AO record you have to configure the fields HIHI and LOLO (and additionally the fields HHSV and LLSV have to be set to "MAJOR").

An operator should react to a major alarm and solve the problem. On displays, an error is shown in red.

## **INVALID**

If a record enters the state **INVALID**, this means that the value of the record is not longer valid. This can happen for example when hardware is broken or not connected (or switched off). On displays, use white to indicate the invalid state.

Directly after reboot all records are set to **INVALID** to indicate that they have never been processed. The following example shows that the record **X10SA-OPBPM3:FIT-H0** has not been processed since the last reboot of the IOC:

```
zimoche@pc4859[/work] ~ > caget X10SA-OP-BPM3:FIT-H0
X10SA-OP-BPM3:FIT-H0 -3082.0000 (SEVR:INVALID STAT:UDF)
```

On the command line, the fields **SEVR** (Severity) and **STAT** (Status) are displayed; **UDF** means undefined. After the record has been processed for the first time (for example when it reads from the hardware), the state changes automatically.

## **not Connected**

Strictly speaking this is not a state of the record at all, but a message from the CA client that it could not connect to the record. This happens if no CA server answers to the broadcast until a timeout. It can happen if you have network problems, the IOC is booted or you misspelled the record name.

C displays this with a white box where the value should be on the window.

Some other programs in the SLS controls system use Pink.

Please make sure that none of your clients show “not connected” records during standard operation. This will cause new broadcasts from the client every time an IOC is booted and this will load the network.

## Howto caQtDM at PSI

caQTDM is installed on all GFA SL6 (Linux) computers by default.

### ***How to Develop***

To start the Qt-Designer use the command  
`caqtdm_designer`

### ***How to Start Panels***

To start a caQtDM panel at PSI use the script  
`caqtdm`

### ***How to convert medm panels***

To convert an medm panel named `xxx.adl` to caQtDM use  
`adl2ui xxx.adl`

This will create a caQtDM file named `xxx.ui`.

## ***caQtDM on Windows***

Installation of caQtDM

- Install the Package using the AIT Software Kiosk
- Now you can start caQtDM (both designer and execution) out of the Start/Programs/caQtDM/caQtDM-Menu

## ***More Documentation***

The main documentation about caQtDM can be found on  
<http://epics.web.psi.ch/software/caqtdm/>

## Cooler Exercise

This exercise was originally developed by John Mclean from Argone National Laboratory for the "Getting started with EPICS" lectures series.

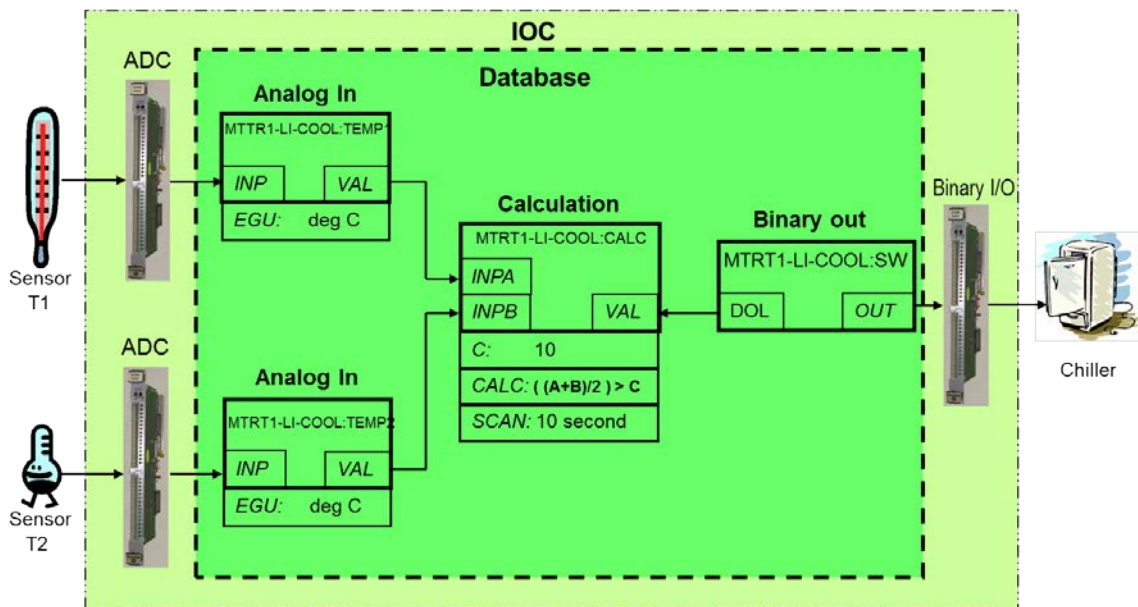
### Problem description

In the LINAC, we have a water chiller that must be turned ON whenever the average temperature of two temperature sensors rises above a set point. The set point is nominally 10 degrees centigrade.

I relay on your imagination for this exercise. The first two potentiometers will be your temperature sensors, to allow you full control over the "temperatures". The cooler will be represented by the first LED.

### Records needed

Only the analog input records and the binary output record are needed if a script does the calculation. But EPICS provides the calculation record to do such simple computation near the other records. This minimizes the chance that the calculation will be stopped by accident.



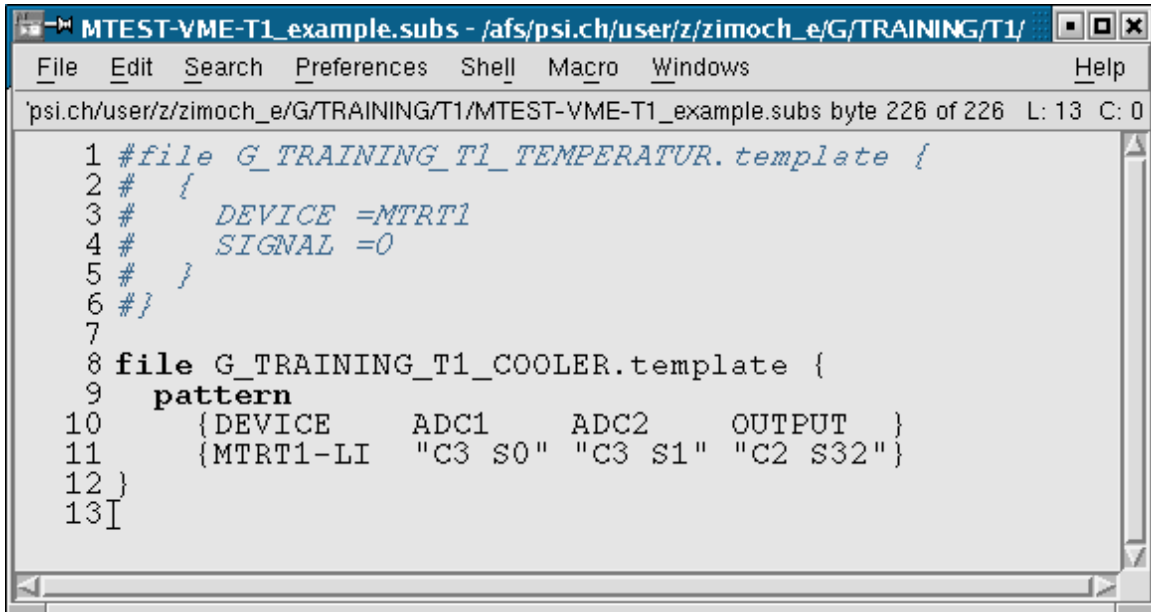
The record names are:

- MTRT1-LI-COOL:TEMP1
- MTRT1-LI-COOL:TEMP2
- MTRT1-LI-COOL:COMPARE
- MTRT1-LI-COOL:SW



## The substitution file

The substitution file is located in the directory ~/G/TRAINING/T1 and is named MTEST-VME-T1\_example.subs



```
1 #file G_TRAINING_T1_TEMPERATUR.template {
2 # {
3 # DEVICE =MTRT1
4 # SIGNAL =0
5 # }
6 #}
7
8 file G_TRAINING_T1_COOLER.template {
9 pattern
10 {DEVICE ADC1 ADC2 OUTPUT }
11 {MTRT1-LI "C3 S0" "C3 S1" "C2 S32" }
12 }
13 |
```

There are two ways to define macros in a substitution file that are equivalent:

1. **file** example.template {  
    {  
        DEVICE = MTRT5  
        SIGNAL = "C3 S1"  
    }  
}
2. **file** example.templeate {  
    **pattern**  
        {DEVICE SIGNAL }  
        {MTRT5 "C3 S1" }  
}

The first way should be used with few macros and few repeated calls of the template. The second method is useful for a large number of macros and a template that is called several times.

## The template file

The template file is located in the directory ~/G/TRAINING/T1 and is named G\_TRAINING\_T1\_COOLER.template

```

record (ai, "$(DEVICE)-COOL:TEMP1")
{
 field (DESC, "Sensor T1")
 field (DTYP, "Hy8401")
 field (INP, "#$(ADC1) @")
 field (EGU, "Grad C")
 field (PREC, "1")
 field (LINR, "LINEAR")
 field (EGUF, "100")
 field (EGUL, "-100")
 field (HOPR, "100")
 field (LOPR, "0")
 field (HIGH, "51")
 field (HIHI, "52")
 field (SCAN, ".1 second")
}

record (ai, "$(DEVICE)-COOL:TEMP2")
{
 field (DESC, "Sensor T2")
 field (DTYP, "Hy8401")
 field (INP, "#$(ADC2) @")
 field (EGU, "Grad C")
 field (PREC, "1")
 field (LINR, "LINEAR")
 field (EGUF, "100")
 field (EGUL, "-100")
 field (HOPR, "100")
 field (LOPR, "0")
 field (HIGH, "51")
 field (HIHI, "52")
 field (SCAN, ".1 second")
}

record (calc, "$(DEVICE)-COOL:COMPARE")
{
 field (DESC, "Calculation")
 field (INPA, "$(DEVICE)-COOL:TEMP1")
 field (INPB, "$(DEVICE)-COOL:TEMP2")
 field (INPC, "10")
 field (CALC, "((A+B)/2) > C")
 field (SCAN, "10 second")
 field (FLNK, "$(DEVICE)-COOL:SW")
}

```

```

record (bo, "$(DEVICE)-COOL:SW")
{
 field (DESC, "Switch for Cooler")
 field (DOL, "$(DEVICE)-COOL:COMPARE")
 field (ZNAM, "OFF")
 field (ONAM, "ON")
 field (DTYP, "Dim8001")
 field (OUT, "#$(OUTPUT) @")
 field (OMSL, "closed_loop")
}

```

### ***Develop the exercise further***

The following instructions are suggestions what to do next. It might be useful to develop at least a rough GUI (Graphical User Interface) first to make debugging easier:

- Create a GUI and display:
  - both temperatures with labels
  - the state of the cooler switch (if the cooler is on or off)
  - the alarm states of the temperature sensors – here you might find that you have to change your temperature records first
  - a plot of the temperatures

More things to do that have more to do with records:

- create a record for the average temperature and display the value
- create a record for the temperature limit so that it can be changed (the limit should stay between 10 and 45 °C) and display it
- for some shutdown work there needs to be a way to switch of the control loop and switch the cooler on and off independent from the measured temperatures
- a new device, a heater, should be switched on if the average temperature is below another limit that can range from 0 to 20 °C (make sure the cooler and the heater are never switched on both and have a shutdown switch for the heater as well)  
The new device is using the same driver than the cooler and the output link should be set to "#C2 S33 @"  
Show all of this in your GUI.
- Provide a choice button to change the rate of calculation from once every 10 seconds to other standard intervals (to find out what these standards are use the command "caget MTRT1-LI-COOL:TEMP1.SCAN")
- There is another cooler/heater combination located in the storage ring – duplicate your records and start the names of the new ones with MTRT1-SR

## Useful things to know about EPICS

EPICS is a set of Open Source software tools, libraries and applications developed collaboratively and used worldwide to create distributed soft real-time control systems for scientific instruments such as a particle accelerators, telescopes and other large scientific experiments.

Additional information and international contacts can be found on the web page

<http://www.aps.anl.gov/epics/>

### ***Ten really neat Things about EPICS***

1. It is free. No license fees, no new payment for every upgrade. You can download EPICS free of charge from the web.
2. It is Open Source (i.e. the source code is accessible). Adaptions and changes due to special environment is therefore possible.
3. There are lots of users. It is tested and most bugs are already found.
4. All a client needs to know to access data is a PV name. No single point of failure due to a nameserver and no messing around with fix addresses.
5. You can pick the best tools out there ...
6. ... or build your own.
7. The boring stuff is already done. For example the communication with Channel Access is stable and well tested.
8. There is a lot of expertise available close by.
9. A good contribution becomes internationally known.
10. It does not matter, if you need 10 or 10 million PVs. You can scale EPICS almost freely.