EUROPEAN SPALLATION SOURCE

MotorRecord demystified

Torsten Bögershausen

Example from real live

- Real live challenge:
  When to get of a train ?
- Polling approach:

```
while true {
   if train_has_stopped() {
        get_of_train();
        sleep(1 minute);
}
```

Real live: problems

- Possible problems
  (We assume that you go to the end station)

  a) sleep too long (waste a lots of neutrons)
  b) This works in Germany, but not in Switzerland

Train in Germany

- b1) Germany. What happens ?

  - train comes to station
  - people get off, people get on. You get on.
  - train departs

- Works!

Train in Switzerland

- b2) Switzerland. What happens ?

  - train comes to station
  - people get off, people get on. You get on.
  - train waits until for the time to depart
  - (4-10 minutes)
  - train departs

- You find yourself going out at the same station. Motor speech: done before even started

More sleep

- Next trial, be on the save side:
  Add a sleep() before checking for train stopped.

```
sleep(10 minutes);
while true {
   if train_has_stopped() {
        get_of_train();
        sleep(1 minute);
}
```

- Problem(s) ?
  - When ~~train~~ motor is fast, 10 minutes unnecessary sleep

Last version

- Last trial:

```
if (switzerland)
   sleep(10 minutes)
elsif germany
   (sleep 2 minutes)
else
   raise exception(“not supported”)
while true {
   if train_has_stopped() {
        get_of_train();
        sleep(20 seconds);
}
```

No !

- NoNoNo:
  - We don’t get enough sleep
  - We waste CPU cycles (here)
    and network and CPU cycles on the IOC
  - can we get a call (on the train)
  - a callback (in EPICS)
- Yes!

Motor move

- What does really happen, when I move a motor ?!?
- time: how long does it take ?
  caput -c -w 100: wait for completion.
  Max 100 seconds
  BIFROST...:Mtr: EPICS motorRecord

```
time caput -c -w 100 BIFROST-SpSl1:MC-SlXm:Mtr 11

Old : BIFROST-SpSl1:MC-SlXm:Mtr      10
New : BIFROST-SpSl1:MC-SlXm:Mtr      11

real    0m1.255s
user    0m0.018s
sys     0m0.018s
```

IOC log file

- Warning:
  The following log needs the ess motor, which has a SPAM field.
  Which had been set to max SPAM.

IOC log: caput

- The caput (or pvput) makes it into the IOC

```
2026/04/16 16:29:34.122
[motorRecord.cc:3274
 BIFROST-SpSl1:MC-SlXm:Mtr 00]
 special fieldIdx=VAL value=10 after=0
```

The special() function in the motorRecord is called: The old value of the VAL field is 10.

special(), 1st call

- special(), first call, after=0

```
2026/04/16 16:29:34.122
[motorRecord.cc:3295 BIFROST-
SpSl1:MC-SlXm:Mtr 00] setDmov
old=1 new=0
```

The motorRecord sets to DMOV field to 0. DMOV stands for Done Moving

special(), 2nd call

- Special() second call, after=1

```
2026/04/16 16:29:34.122
[motorRecord.cc:3274 BIFROST-
SpSl1:MC-SlXm:Mtr 00] special
fieldIdx=VAL value=11 after=1
```

The new value of the VAL field is 11.

process()

- # process() is called:
  ```
  16:29:34.122 [mr] process:begin [CAS-client]
  16:29:34.122 [mr] doDVALchangedOrNOTdoneMoving: begin … mip=0x0('')
  ```
  # a decision is needed, based on DVAL and DRBV
  ```
  16:29:34.122 [mr] doRetryOrDone dval=11.000000 … drbv=10.000000
  16:29:34.122 [mr] mipSetBit MOVE(Mo) old=''
  ```

process()

- # record starts movement, absolute

```
16:29:34.122 [mr]
doMoveDialPosition(2178)
mode=Position position=11.000000
frac=1.000000 use_rel=0
val=11.000000 diffDial=1.000000

# we leave the context of channel access task (!)
16:29:34.122 [mr]
process:---------------------- end
```

Motor driver

- Task switch: The motor driver

  # The motor driver (task) is active.
  # axis 0. Units are in picometer

```
16:29:34.122 [smarActMCS2MotorDriver.cpp:510]
MCS2Axis::move(0) position=11000000
relative=0 sensorPresent=1 sensorIsDisabled=0
openLoop=0 minVelocity=0.000000
maxVelocity=1000000.000000
acceleration=10000000.000000
```

  # Not shown: The command that is send to the controller.

  # printout from poller. Bit0 is set (+MOVING)

```
16:29:34.123 [smarActMCS2MotorDriver.cpp:768]
poll(0) oldChanState=0x000e0
chanState=0x000e1 +MOVING
```

Motor driver

EUROPEAN SPALLATION SOURCE

-

```
# code is executed inside the MCS2 task
16:29:34.123 [mr]
process:---------------------- begin
[MCS2] mip=0x20(MOVE(Mo))
reason=1,'callbackdata' rmp=10000000
rep=10001000
# The MSTA field is updated
16:29:34.123 [mr] msta.Bits.RA_DONE=0
16:29:34.123 [mr]
process:---------------------- end
```

Callbacks from poller in motor driver

- Motor is moving

```
16:29:34.125 [mr] process:---------------------- begin
[motorPoller] mip=0x20(MOVE(Mo)) reason=1,'callbackdata'
rmp=11000000 rep=10001138
16:29:34.125 [mr] process:---------------------- end

16:29:34.231 [mr] process:---------------------- begin
[motorPoller] mip=0x20(MOVE(Mo)) reason=1,'callbackdata'
rmp=11000000 rep=1010784016:29:34.231
[mr] process:---------------------- end

16:29:35.071 [mr] process:---------------------- begin
[motorPoller] mip=0x20(MOVE(Mo)) reason=1,'callbackdata'
rmp=11000000 rep=10947209
16:29:35.071 [mr] process:---------------------- end
```

Callback when motor is stopped

- Motor has stopped

  # Driver reads not moving from controller

```
16:29:35.176 [smarActMCS2MotorDriver.cpp:768] poll(0)
oldChanState=0x000e1 chanState=0x000e0 -MOVING
```

  # simplified process()

```
16:29:35.177 [mr] process: begin [motorPoller]
mip=0x20(MOVE(Mo)) reason=1,'callbackdata' rmp=11000000
rep=11001000
16:29:35.177 [mr] motor is stopped dval=11.000000
drbv=11.000000 ...
16:29:35.177 [mr] mipSetBit DELAY_REQUEST(Dr)
old=MOVE(Mo) new='Mo Dr'
16:29:35.177 [mr] callbackRequestDelayed() called
```

  16:29:35.177 [mr] process:---------- end

  # waiting for the delay

Delay

- After delay

  # 100 msec later: callback from delay

```
16:29:35.276 [mr] mipClrBit DELAY_REQUEST(Dr)
old='Mo Dr' new=MOVE(Mo)
16:29:35.276 [mr] mipSetBit DELAY_ACK(Da)
old=MOVE(Mo) new='Mo Da'
```

  # process() is called

```
16:29:35.276 [mr] process:begin [scanOnce]
mip=0x820('Mo Da') reason=0,'nothing done'
rmp=11000000 rep=11001000
16:29:35.276 [mr] motor is stopped dval=11.000000
```

After delay

- Callback from delay, part 2

  16:29:35.276 [mr] motor is stopped dval=11.000000 drbv=11.000000 pp=0 udf=0 stat=0 stop=0 pmr->spmg=GO mip=0x820('Mo Da') msta=0x4902
  16:29:35.276 [mr] setDmov MARK old=0 new=1
  # ask the driver for fresh information
  16:29:35.276 [mr] devSupGetInfo
  # Not done yet: set DMOV back to false
  16:29:35.276 [mr] setDmov UNMARK old=1 new=0
  16:29:35.276 [mr] process:----------
  end

End of movement – asked one more time

- New callback

  # process() is called from the driver after devSupGetInfo()

```
16:29:35.277 [mr] process:-- begin [MCS2]
mip=0xc20('Mo Dr Da')
reason='callbackdata' rmp=11000000
rep=11001000
16:29:35.277 [mr] motor is stopped
dval=11.000000 drbv=11.000000 pp=0 udf=0
stat=0 stop=0 pmr->spmg=GO mip=0xc20('Mo
Dr Da') msta=0x4902
...
```

End of movement - done

- Retry needed ?

```
16:29:35.277 [mr] maybeRetry: close
enough commandedDval=11.000000
last.dval=11.000000 dval=11.000000
drbv=11.000000 rdbd=0.010000
diff=0.000000 rcnt=0 pmr->rtry=5
mip=0x20(MOVE(Mo))
```

movement complete. Set DMOV to true, “monitor callback”

```
16:29:35.277 [mr] mipClrBit 'Jf Jr J1 Hf
Hr Mo Ry Lp Mb St Dr Da jS J2 Ex'
old=MOVE(Mo) new=''
16:29:35.277 [mr]
process:---------------------- end
```

conclusion

EUROPEAN SPALLATION SOURCE

- MotorRecord
  - defines a well-known interface
  - has a state machine
    (one command at a time, backlash, retries)
  - caput, pvput, pvxput, pyepics, p4p:
    can wait for finished movement (!)
- The “done” bit (in the driver) is important
- A driver (motor model 3) is easy (™) to write
- Anyone interested in a training ?

This is it

- Thanks for listening