OAK RIDGE National Laboratory

Put- and Get-callback (aka completion) Support, 'Busy' Record

Kay Kasemir

July 2026

U.S. DEPARTMENT OF ENERGY

Basic Automation Example

1. Move motor to some position

Has the motor reached the position?

2. Open shutter

How quickly does the shutter open?

3. Take data for 5 seconds

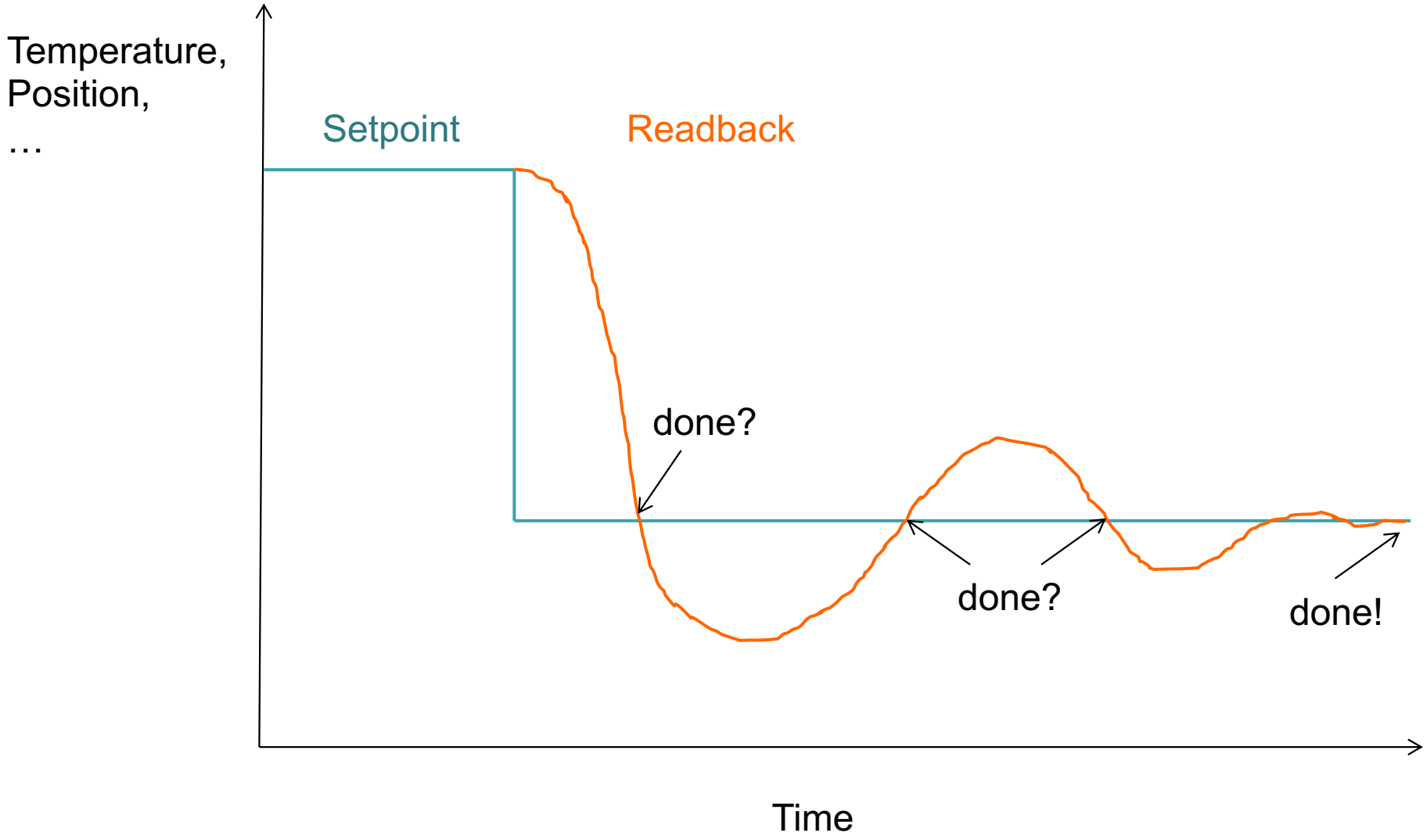
4. Close shutter

What not to do

1. Ask motor to move to position X
2. Wait until motor encoder reports X±0.1mm
3. Ask shutter to open
4. Wait 4 seconds - Fred said that's how long it'll take
5. Take data for 5 seconds
6. Close shutter

Not reliable!

Based on readback, are we there, yet?



IOC has to tell us when it's done!

1. Use CA "put-callback"
2. Record completes callback when it is "done"

Requires record with support for "completion" aka "put-callback"

✓ Motor record: When reaching desired position

? Other record: Check device support manual

! Plain database: Use BUSY record

Does “SomeRecord” support put-callback?

- Hard to tell from outside
  - No? “put-callback” will return right away like a plain “put”
  - Yes? “put-callback” will return when record is done, which could be after 0.01 sec (fast shutter) or hours (cryostat cooling down)…
- Device support manual tells you that put-callback is supported?
  - Good!
    Motor record, some temperature controller device support, …
- BUSY record allows you to create record logic that supports put-callback

Internals: EPICS record PACT field

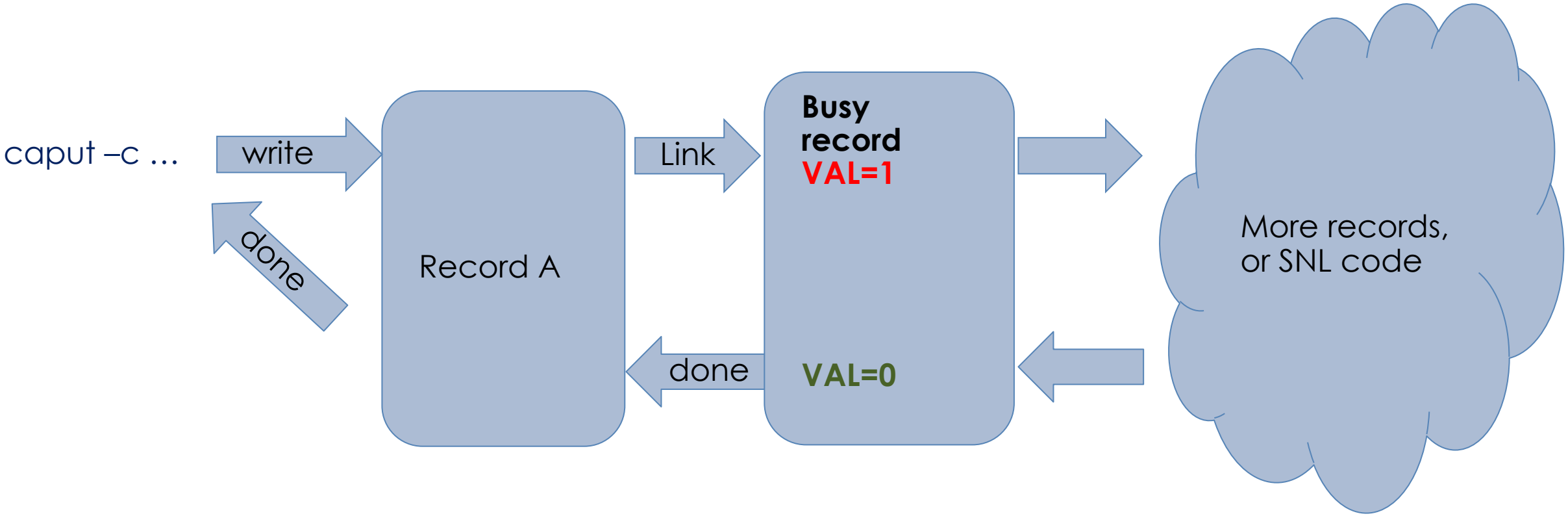
- PACT is binary flag that is 1 (ON) when record is processing
- CA put-callback is mechanism to use PACT information…
- …and not return until PACT is 0 (OFF)

caput -c -w *timeout PV* *value*

caput –c –w 20 SomeMotorRecord 5.7

BUSY Record to support Put-Callback

- VAL = 1 "Busy", VAL=0 "Done"
- Blocks processing (PACT=1) while busy



Typical Use

1. Writing to some record, which has FLNK or OUT links to a busy record, sets busy.VAL = 1
2. This causes writer to that first record to block in put-callback
3. Something else resets busy.VAL = 0
   a) Other database logic
   b) Sequencer code
   c) asyn device support
   d) …
4. Put-callback completes

Monitor, active Get, Get-Callback

- Typical client should use 'monitor' to efficiently receive the most recent value
  - PyEPICS `caget()` does this by default
- But consider this:
  - We monitor "SomePV", last received value was 10
  - `caput(SomePV,11)`

  We should *real soon* receive an update with value 11,
  but until then PyEPICS `caget("SomePV")` will return 10

  → Sometimes you need to force an active, round-trip "caget",
  not rely on the next monitor.
  In PyEPICS: `caget("SomePV", use_monitor=False)`
- There is also "get callback" that will cause record to process, wait for completion, then return the current value

Put-Callback
(and sometimes round-trip get or get-callback)

.. Is essential for robust automation

Check if your device support offers callback/completion

If not, use BUSY record to create database logic that supports put-callback