Australian Government

ANSTO

Motion Software Stack Development for PowerPmacLV controllers in the Australian Synchrotron

Nader Afshar, Ben Baldwinson, Mark Clift, Ross Hogan, Danny Wong and Alan NG – Controls Systems

ICALEPCS 2021 - MOBR05 - Session Date: 18-OCT-21

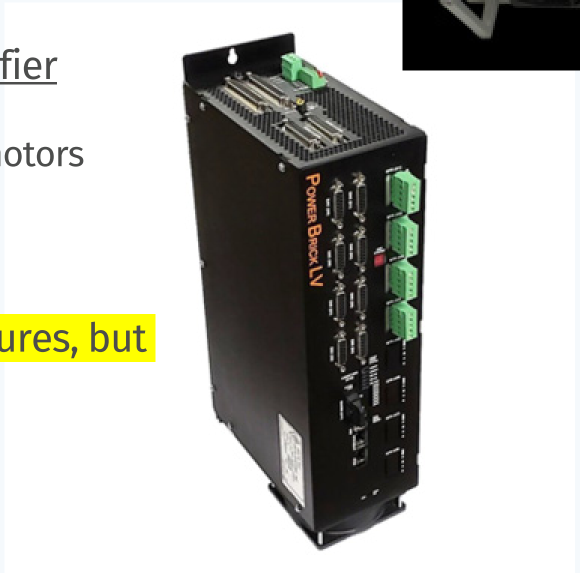
Science. Ingenuity. Sustainability.

Motion software stack for PowerPmac:

- Requirements, challenges and limitations
- Implementation - traditional Motor Record with float64 readback
- Implementation - python based abstraction layer: capmac

The motion controller

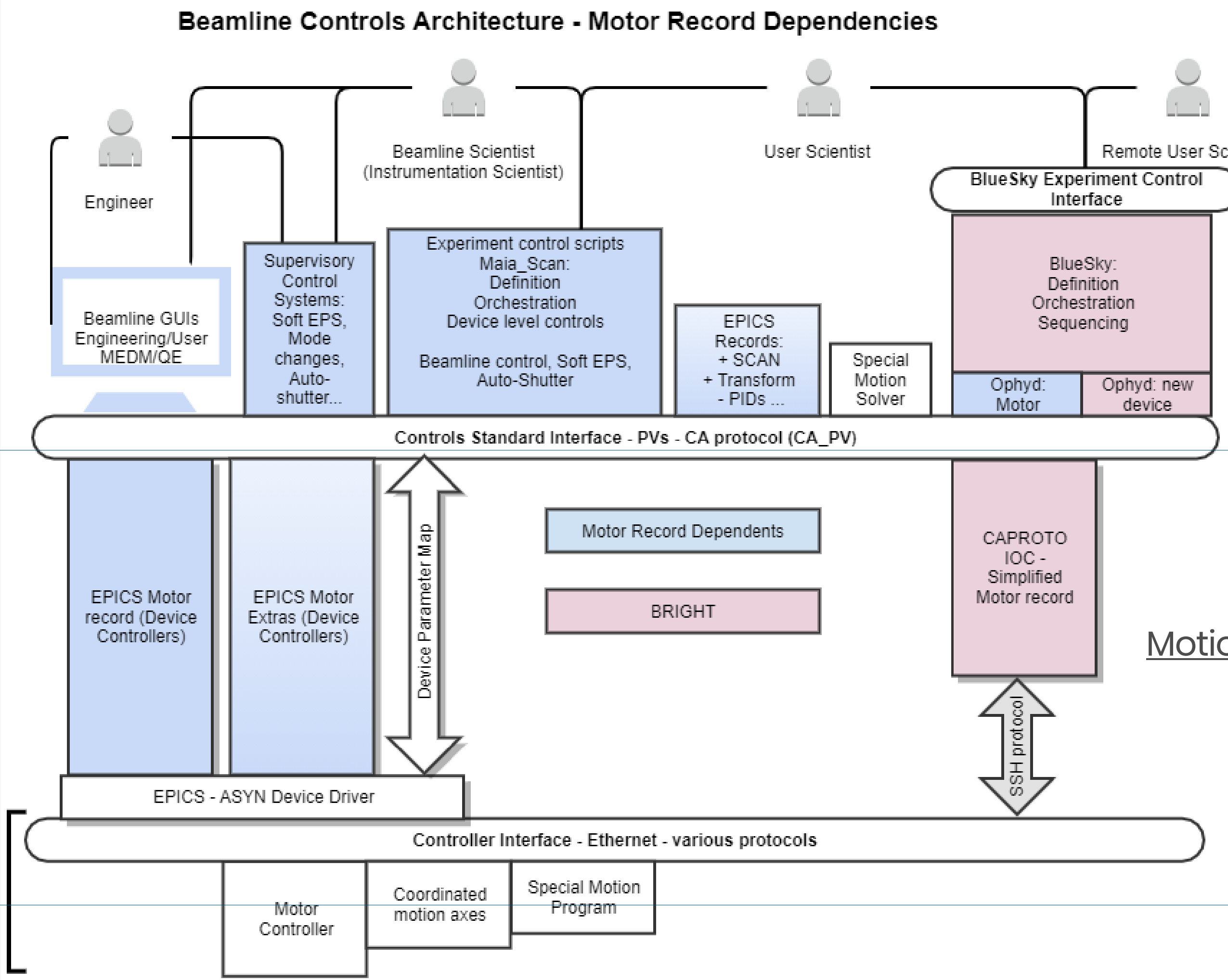
- Australian Synchrotron standardised on Faraday's IMS controller based on Omron's PowerBrickLV
- Advanced and versatile controller platform
- 8 channel integrated 30A 48V (LV) PWM amplifier
    - 2 phase Steppers and 3 phase Brushless DC motors
- Fully standardised hardware
- Challenge: utilise the configurability and features, but keep it maintainable



ANSTO

Deep dive into the problem space

- Do we still need a *motor abstraction layer*? EPICS Motor Record?
  - yes, Motor Record and/or a streamlined variation of it
- Things we do/don't like about EPICS?
  - Channel Access ✓ / controls logic programming ×
- Min. level of backward compatibility to maintain?
  - at the motor abstraction level (top of the stack) as well as at hardware interfacing
- Controller as part of assembly or as part of the infrastructure?
  - infrastructure, no hardware customisation
- ...



Motion Software Stack

Supporting non-integer readbacks

- ppmac is a floating-point system with no concept of “steps”
- Interactions with motors and coordinate axes make more sense when scaled to engineering units e.g., mm, mrad, deg, revs

Motor Record: scaling

Original model

- `UREV[EGU/rev] = MRES[EGU/Steps] * SREV[Steps/rev]`
- `Dial_readback[EGU] = Controller_readback[Steps] * MRES[EGU/Steps]`
- `VELO[EGU/s] = UREV[EGU/rev] * S[rev/s]`
- `SREV is integer`

MotorFloat64 model

- `UREV[EGU/rev] = MRES[EGU/MotU] * SREV[MotU/rev]`
- `Dial_readback[EGU] = Controller_readback[MotU] * MRES[EGU/MotU]`
- `VELO[EGU/s] = UREV[EGU/rev] * S[rev/s]`
- `SREV is float`

AS2 scaling functionality

- Steps → MotU , Encoder Counts → EncU

This intuitive change is both necessary and sufficient to make Motor Record work in this general case.

» Steps and Counts are abstracted at the controller level by choice.

» Scale factor between MotU and Motor Record EGU is not forced by motor step resolution nor by encoder resolution.

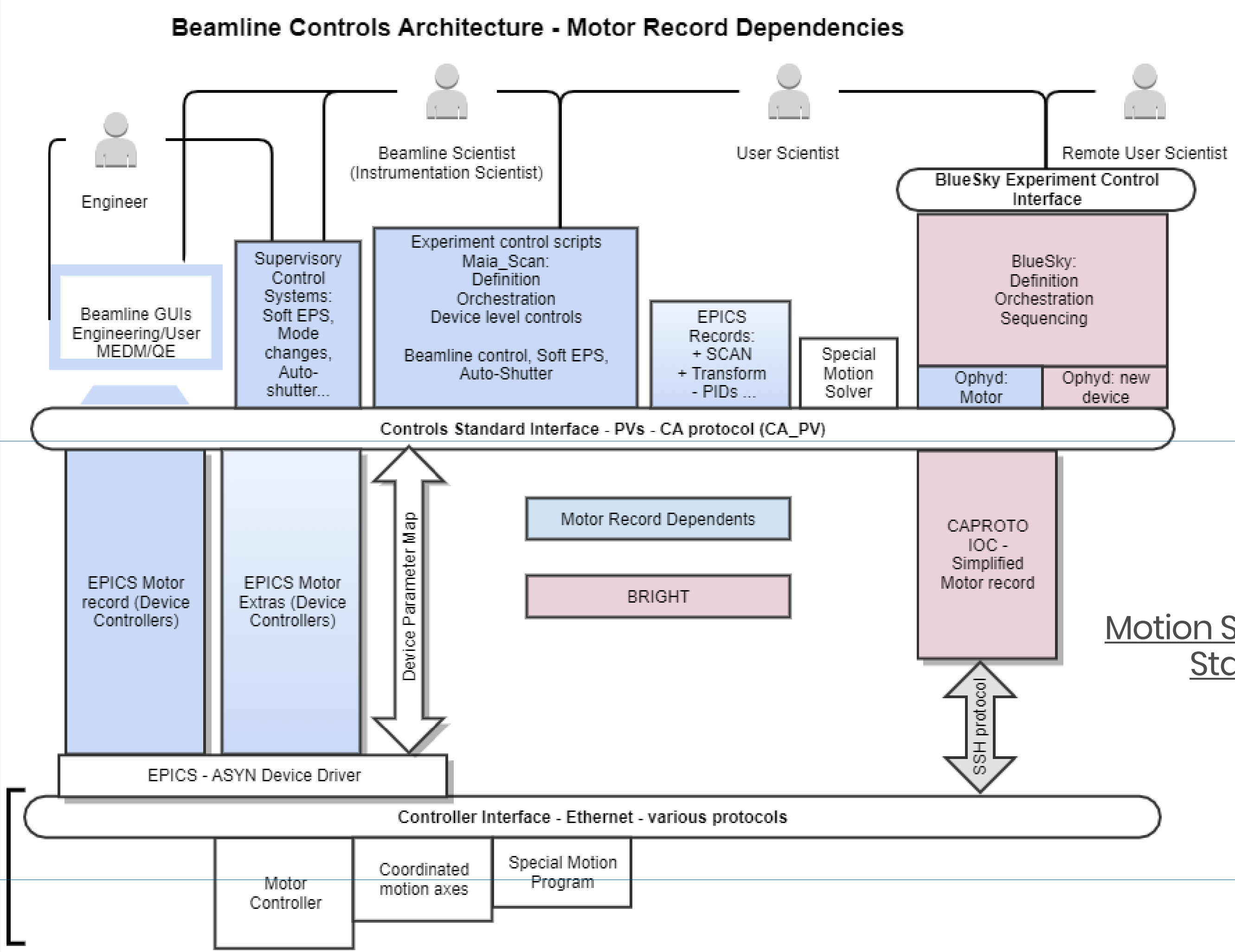
ANSTO

AS2 stack MIM functionality

- Scale and Minimum Incremental Motion (AMIM) can no longer be fused into one field.
  - New PV .AMIM (Axis Minimum Incremental Motion) is introduced
  - .AMIM default value is 1e-9 [EGU], which effectively nullifies the functionality unless needed.

AS2 software stack implementation

- Motor Record R7-2 (08-21-18) [epics-modules/motor: APS BCDA synApps module: motor (github.com)]
- Imported as a baseline for AS in March 2019. https://github.com/dls-controls/pmac
- Fixes and improvements at the DLS ppmac driver level
- New motorFloat64 Motor Record with floating point readback chain
- Adaptations and streamlining of DLS IOC templates
- Developed motion test cases and validation software
  - Capability of rapidly and reliably validating the software/configuration



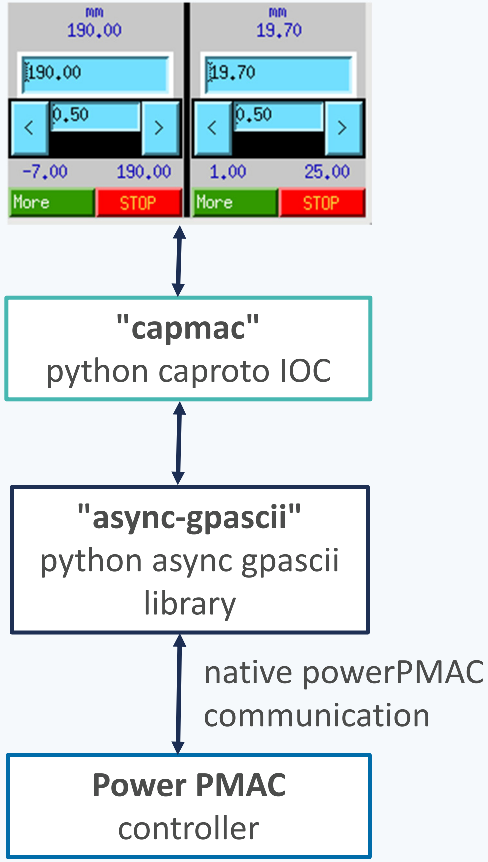
Motion Software Stack

caproto IOC – "capmac"

- Pronounced "kah-pee-mac"
- What is it?
  - Python EPICS IOC for powerPMAC controller
  - Based on caproto - a pure python CA library, https://nsls-ii.github.io/caproto/
  - Configurable via yaml file
- Why?
  - Lightweight, can be deployed anywhere
  - Rapid to prototype, no build step required.
  - Good for people familiar with python but not EPICS C++ environment

How does it work?

- Python library "async-gpascii" provides async calls to read/send native commands to the powerPMAC
- "capmac" IOC uses the async library to translate between CA and async calls
- caproto useful for any device if you have a python library



Configuration

- Instance deployment customised via yaml file
- Run command with argument

`>capmac test_ppmac.yml`

- Use your favorite python environment management tool

`>pip install capmac`

```yaml
ppmac_top:
  ip_address: "192.168.0.157"
channels:
  '1':
    pv_name: "TEST_PPMAC:ch1:"
    description: "test axis ch 1"
    direction: 0  # 0= +ve, 1=-ve
    MRES: 1  # [EGU/step]
    velocity: 90 # [deg/s]  "VELO"
    max_velocity: 180  # [deg/s] "VMAX"
    # motor-record settings
    precision: 4 # "PREC"
    engineering_units: "deg" # "EGU"
    soft_limit_plus: 1000 # [deg] "LLM"
    soft_limit_minus: -1000 # [deg] "HLM"
    tweak_value: 100  # [deg] "TWV"
    home_offset: 0 # [deg] "OFF"
```

ANSTO

Code snippet

- Caproto provides an endpoint of a caput for most common records (but not what to do with that endpoint)
- Shown is for a caput to "motor" record ".VELO" PV
- Includes clipping to VMAX and convert user-to-dial
- We pipe the requested velocity to the async function "set_velocity()"
- Set_velocity() sends the native command "motor[x].jogspeed" to the controller
- Response is in-line, no callback
- At this level only a few lines of code required

```
@motor.fields.velocity.putter
async def motor(fields, instance, value):
    slf = fields.parent.group
    vmax = slf.motor.field_inst.max_velocity.value
    if value > vmax:
        velo = vmax
        logger.info(f"VELO clipped to VMAX, {velo}")
    else:
        velo = value
    dvelo = slf.user2dial(velo) / 1000

    await slf.set_velocity(dvelo)
```

```
async def set_velocity(self, velocity):
    send_cmd = f"Motor[{self.chan}].JogSpeed={velocity}"
    logger.info(f"set_velocity: {velocity}, writing {send_cmd}")
    reply = await self.ppmac.send_receive(send_cmd)
    self.check_error(send_cmd, reply)
```

ANSTO

What about motor record functionality?

- Chose not to implement all of the motor record features
- With modern controller many motor record features can be included in the controller itself
- Implementing these PVs allows capmac to operate with bluesky/ophyd - our experimental layer, and provide basic scaling
- Keeps code light and easy to follow

DIR	HLM
ATHM	HLS
VELO	LLM
DVAL	LLS
DRBV	RBV
TWF	DHLM
TWR	DLLM
TWV	DMOV
OFF	VMAX
RHLS	EGU
RLLS	PREC
STOP	MSTA
SYNC	

ANSTO

Testing

- As fast as regular EPICS IOC for this application
- Identified bottleneck in powerPMAC "gpascii" communication; i.e., found that send-many then receive-many is faster than many individual send-receive
- Error messages are common in console; Caproto as well as capmac is still in development and need contributors

ANSTO

Australian Government

ANSTO

Thank you.

Science. Ingenuity. Sustainability.

Terminology

- **EGU:** Position unit at Motor Record- same as IOCUnit, IOCEGU ...
- **MotU:** Position unit reading back from controller, same as ControllerUnit, ControllerEGU, MotorU, MotorUnit
- **EncU:** Position unit reading back from a readback encoder, as in open-loop encoder readback; same as EncoderUnit (float)
- **Step**: Stepper motor step count (integer)
- **Count**: Encoder count, same as encoder tick (integer)
- **rev**: Motor revolution (float)

ANSTO