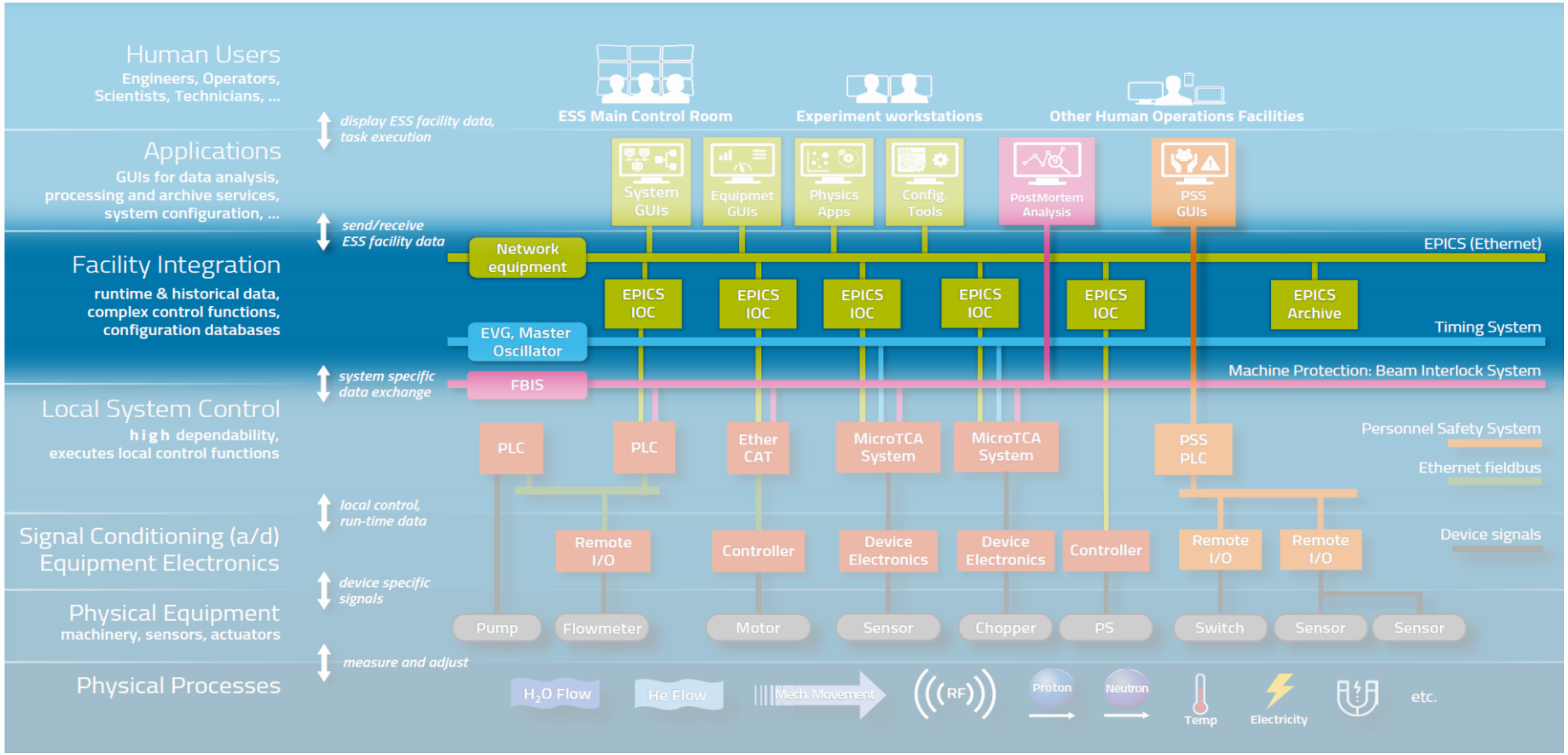
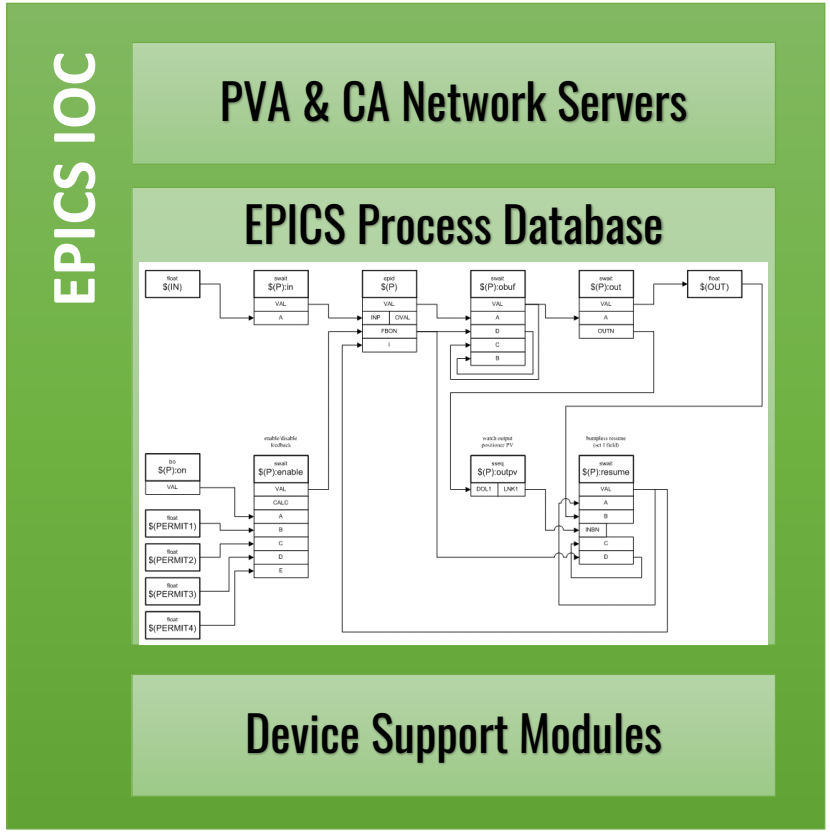
Technology Focus:
Integrating COTS Devices with EPICS

Ralph Lange, ITER Organization

Context: EPICS IOCs in the Integration Layer



Context: Device Support – Connecting Records to Hardware



- EPICS clients connect using the *pvAccess* and *Channel Access* network protocols
- Configurable interconnected objects (*records*) provide flexibility and functionality
- Records that represent I/O channels use the *Device Support* layer to connect to hardware

Choices are: Standards or Flexibility

One of the EPICS principles: **Configuration over Coding**

Two examples:

1 – Use a rich high-level communication standard

- Rich standard covers everything you would ever need
- On-the-wire protocol: standard
- Shifts integration effort to the device

2 – Use a low-level communication abstraction with a flexible configuration

- Transport of messages is handled by fixed (coded) parts
- Content of messages is handled by a configurable engine
- On-the-wire protocol: device specific
- Shifts integration effort to the IOC (message engine configuration)

StreamDevice

EPICS

1 – OPC UA: Interfacing SCADA to Controllers (PLCs)

Industrial standard (2006) to interface controllers

- Covers data, alarms, events, historical data, remote methods
- Symbolic addressing
- Browsable

Based on OPC Classic (Microsoft, 1996), adding

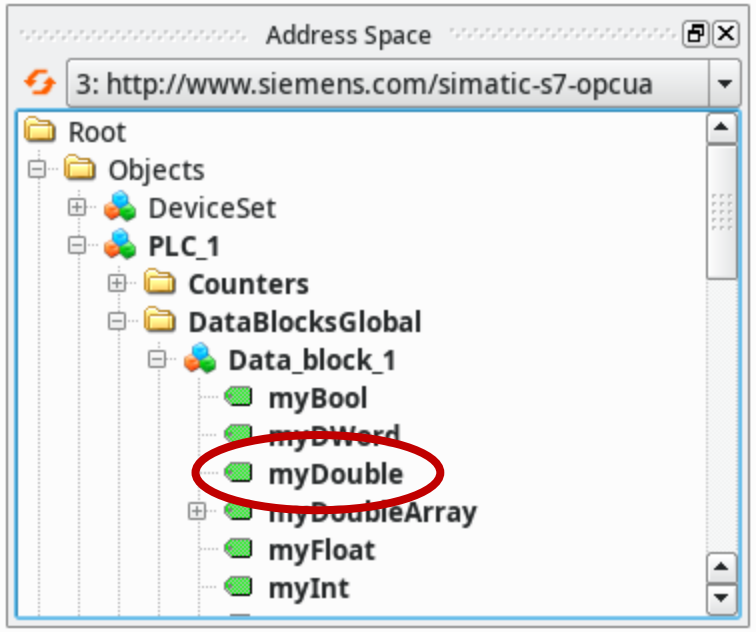
- Portability
- WAN Support (TCP instead of DCOM)
- Safety/security (certificate-based authentication and encrypted connection)
- Information modeling (user-defined structures)

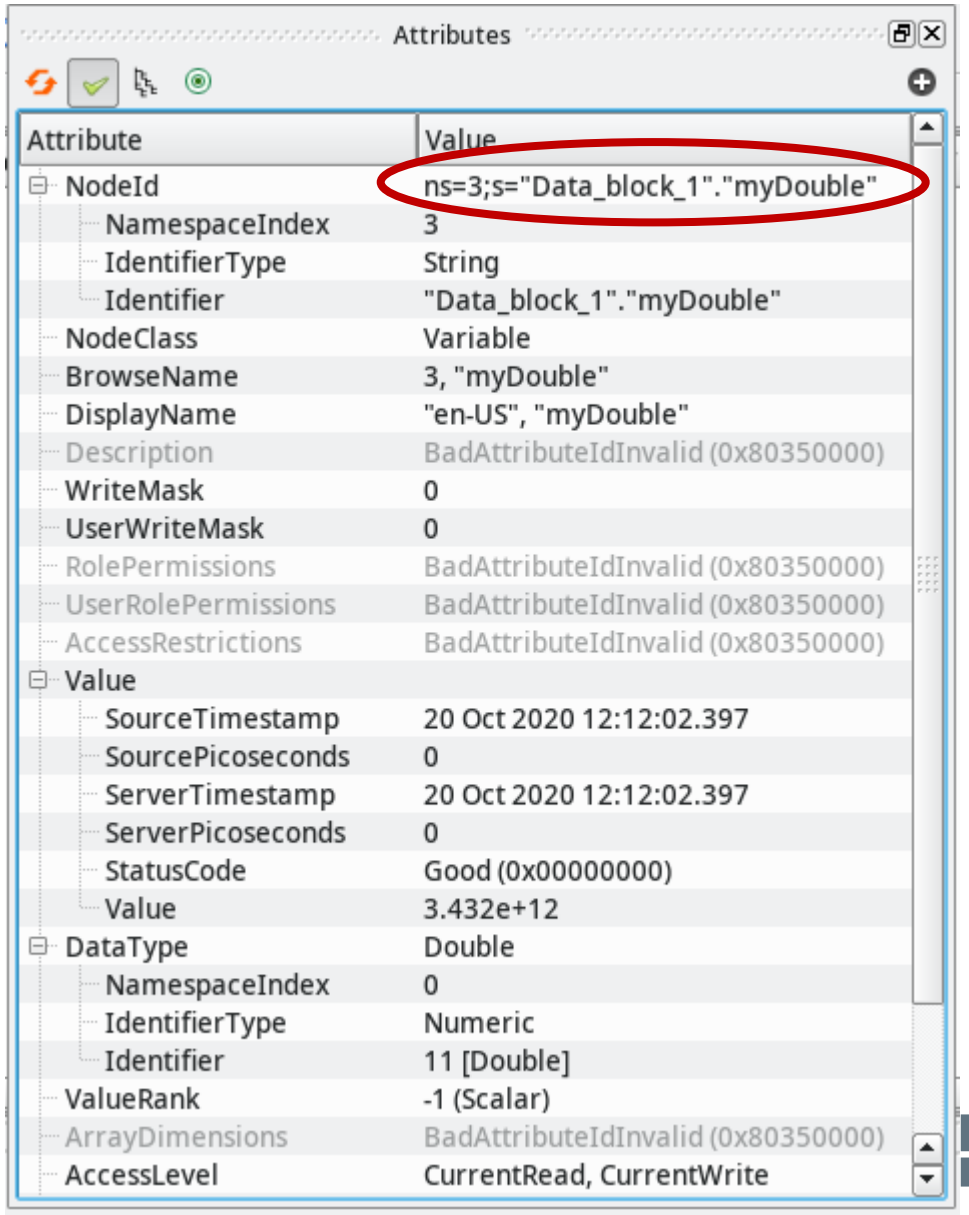
Gaining momentum as universal integration standard

Some controllers (e.g., Siemens S7-1500 series PLCs) offer an embedded OPC UA server

1 – OPC UA: Integration Workflow

Step 1: Browse the server to find the interesting things





*Free browsing tool: **UaExpert***

1 – OPC UA: Integration Workflow

Step 2: Configure the IOC by connecting to the OPC UA server from the startup script…

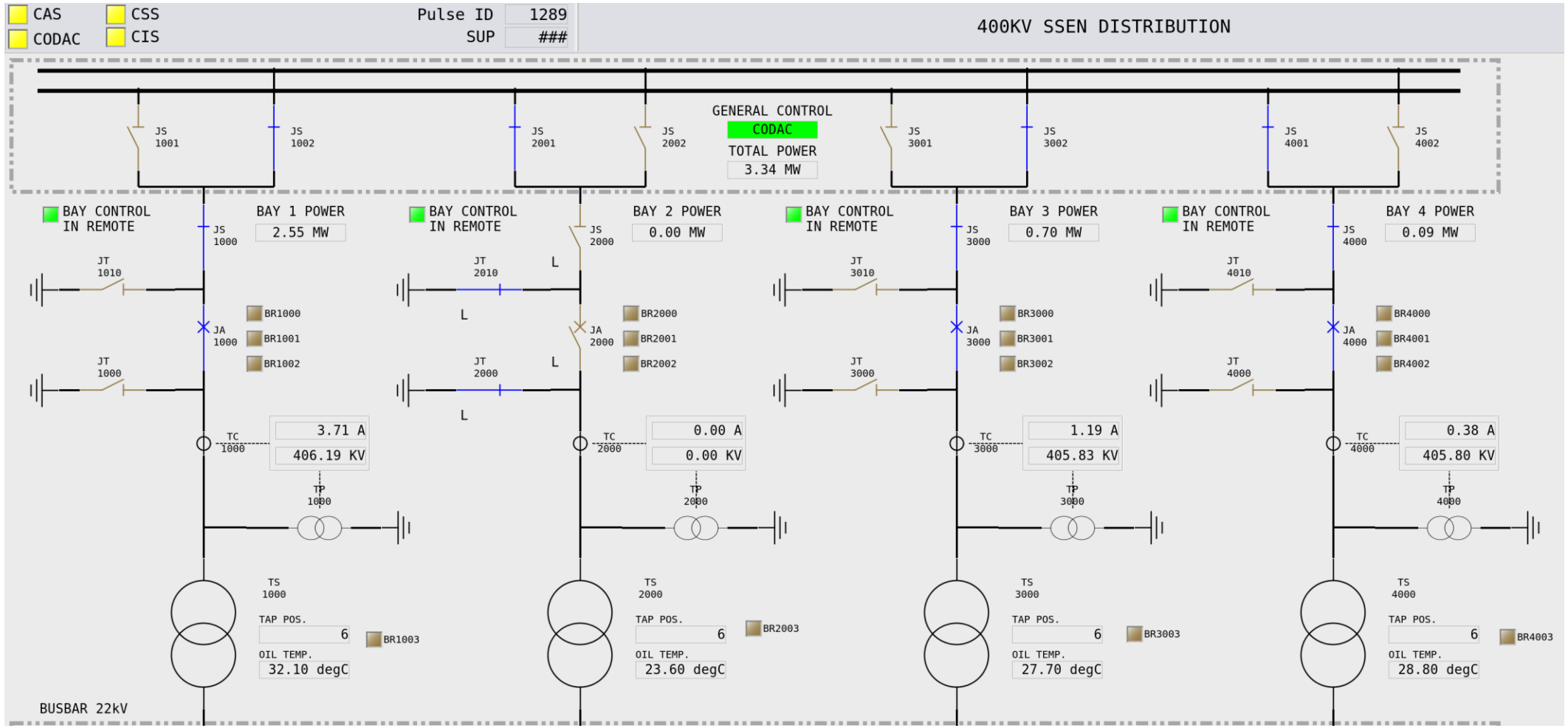
```
opcuaCreateSession PLC1 opc.tcp://plc1.controls.local:4840
opcuaCreateSubscription SUB1 PLC1 200
```

…and adding an EPICS record to your database

```
record(ai, "$(DEVICE):dbl1") {
    field(DTYP, "OPCUA")
    field(INP,  "@SUB1 ns=3;s=\"Data_block_1\".\"myDouble\"")
    field(PREC, "3")
}
```

1 – OPC UA: Integration Workflow

Step 3: Design your client applications and connect



Other Communication Standards Supported by EPICS

OPC Classic

- Predecessor (subset) of OPC UA; Windows only; local network only

EtherCAT

- Ethernet based fieldbus technology
- Real-time capability (deterministic)

MODBUS

- Often used to integrate remote I/O suited for PLCs directly into an EPICS IOC

BACnet

- Standard used in Building Automation and Control: HVAC, lighting, cooling water, fire detection

2 – StreamDevice: Request/Response Done printf/scanf Style

Abstraction of request/response style communication over streaming transport

- EPICS Device Support (for all standard record types) by Dirk Zimoch / PSI
- Uses the ASYN framework to provides a common „byte stream“ type interface to everything serial: RS-232, RS-485, IEEE-488 (GPIB), TCP, UDP

Run-time protocol engine that handles the content going over the wire

StreamDevice

- ASCII configuration file defines a set of request/response pairs *(protocols)*
- Values are formatted and parsed through printf/scanf type format converters (also supporting hex, raw byte, BCD, different checksums, regular expressions,...)

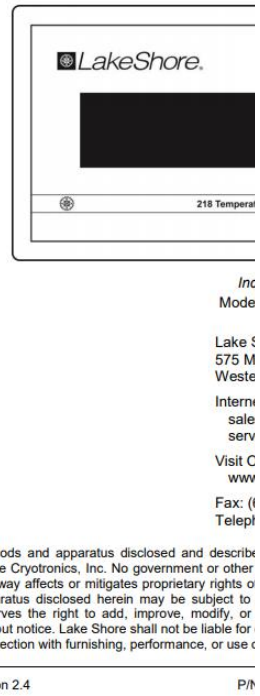
Most common EPICS method to integrate „String over TCP“ devices

- Hundreds of protocol files exist, mostly available „on demand“ through Tech-Talk

EPICS

2 – StreamDevice: Integration Workflow

Step 1: Read the friendly manual



KRDG? Query Kelvin Reading for a Single Input or All Inputs

Input: **KRDG? <input>**

Returned: **<Kelvin value>**. Format: **+nn.nnn[term]**
Or if all inputs are queried:
<Input 1 Kelvin Value>,<Input 2 Kelvin Value>,<Input 3 Kelvin Value>,<Input 4 Kelvin Value>,<Input 5 Kelvin Value>,<Input 6 Kelvin Value>,<Input 7 Kelvin Value>,<Input 8 Kelvin Value>. Format: **+nn.nnn,+nn.nnn,+nn.nnn,+nn.nnn,+nn.nnn,+nn.nnn,+nn.nnn,+nn.nnn**

Remarks: Returns the Kelvin reading for a single input or all inputs. **<input>** specifies which input(s) to query. **0** = all inputs, **1-8** = individual input. **NOTE:** Use **0** (all inputs) when reading two or more inputs at the maximum update rate of 16 rdg/s.

Remote Operation 6-25

2 – StreamDevice: Integration Workflow

Step 2: Add a protocol to the protocol file for your device (e.g., LS218.prot)

```
…
temp_K_1 {
    out "KRDG? 1";
    in "%f";
}
…
```

EPICS

2 – StreamDevice: Integration Workflow

Step 3: Configure the IOC by connecting to the device from the startup script…

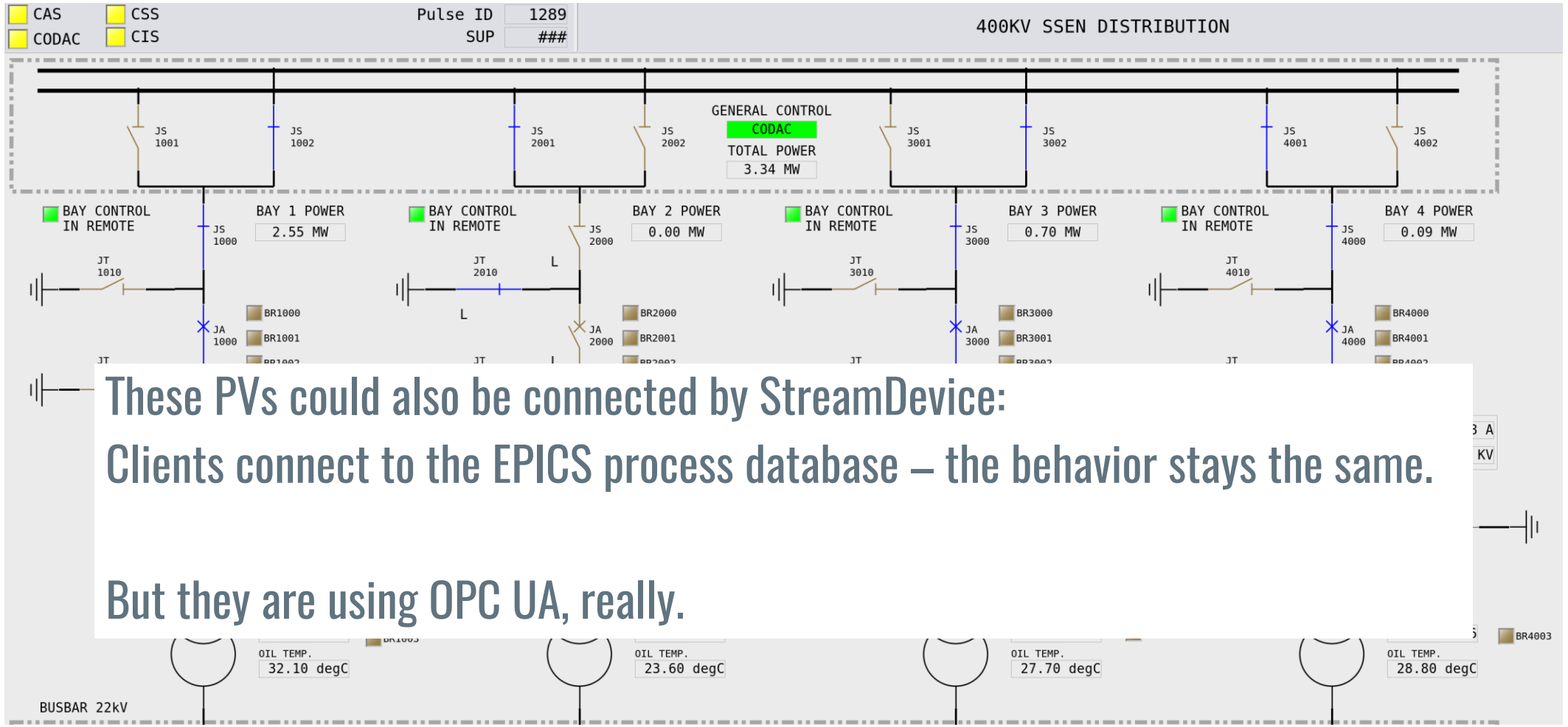
```
drvAsynSerialPortConfigure "LakeShore1", "/dev/ttyS0"
asynSetOption "LakeShore1", 0, "baud", "9600"
asynSetOption "LakeShore1", 0, …
```

…and adding an EPICS record to your database

```
record (ai, "$(DEVICE):TEMP1") {
    field (DTYP, "stream")
    field (INP,  "@LS218.prot temp_K_1 LakeShore1")
    field (EGU,  "K")
    field (PREC, "2")
}
```

2 – StreamDevice: Integration Workflow

Step 4: Design your client applications and connect



These PVs could also be connected by StreamDevice:

Clients connect to the EPICS process database – the behavior stays the same.

But they are using OPC UA, really.

Considerations for Any Specific Device

Rich high-level communication standard (e.g., OPC UA)

- High effort on the device controller
  Development: all adaptation is done on the controller, extensive testing necessary
  Run-time: needs considerable CPU resources on the controller
- One job for all
  Many SCADA systems will be able to integrate the device (not just EPICS)

Low-level byte stream request/response communication (StreamDevice)

- Interface needs to be well-defined, well-documented and stable
  Incompatible changes will break all integrations (not just EPICS)
- Use regular communication patterns and keep the commands stateless
- Optimal vendor-supplied EPICS Integration:
  Protocol file and matching simple test application (names and screens will not be used)
  Device emulator (e.g., Python script) to support automated testing w/o hardware

EPICS

Conclusion

EPICS can integrate COTS devices using generic Device Support modules implementing high levels of standardization and/or flexibility.

Such approaches allow interfacing many different devices by different vendors without coding or recompilation, by configuring the EPICS Process Database appropriately.

Device vendors can ease EPICS integration of their devices by following standards and good practices. Companies providing EPICS consulting can help you.

Thank you for your attention.

EPICS