

EUROPEAN SPALLATION SOURCE



DAQ Architecture for Instruments at the European Spallation Source

PRESENTED BY TOBIAS RICHTER - EXPERIMENT CONTROL AND DATA CURATION GROUP



2022-06



ess

ESS Controls and Readout Architecture

Based on EPICS – Part of the Ancient Revelations of ESS



Low Level Control and Engineering Interfaces



Three types of interfaces:

User

The average facility user

Expert

An expert user, also often the instrument scientist

Engineering

ESS support staff with special needs outside of running an experiment

We aim to cater for users and experts in NICOS.

The ideal way to support engineering tasks for a specific device depends on many factors.

Main Menu Enabled		Timing Setttings		Network Setttings RO1 Sta		s and Config RO1	
hannel settings —	Input Impedance	Range	Readback	Offset	Threshold		Buffer Overflow Count
	50 Ohms	▼ +/-5V ▼	+/-5V 50Ohm	0.050 V	0.5500 V -5	5.0000 V	0
2	50 Ohms	▼ +/-50mV ▼	+/-50mV 500hm	0.000 V	0.0000 V -C	0.0500 V	0
3	50 Ohms	▼ +/-50mV ▼	+/-50mV 500hm	0.000 V	0.0000 V -0	0.0500 V	0
4	50 Ohms	▼ +/-50mV ▼	+/-50mV 500hm	0.000 V	0.0000 V -0	0.0500 V	0
ak Detection Paran Triggering Rising Hold Falling Hold Samples Before Samples After	Peak Detect v 1 1 5 1	Peak Detect 1 1 5 1		Ι			



Experiment Control Programme

NICOS

Integrated high level interface and device abstraction to low level EPICS controls.

- Python CLI
- Qt user interface
- Extensible and customizable

NOT an integral part of the data path.

- Interacts with devices via EPICS
- Controls file writing via Kafka and sends data there for recording
- Can visualise live data from Kafka

000	NCOS - guest at 172.30.242.32:1301				
	Instrument: Unknown Experiment: Unknown				
Experiment	>> + New Cmd + Rur	Devices			
Samples		Enter search expression		*	
	Output Scan Plot Detector Image Script Status	Name	Value	Status	
Instrument interaction	m [08:05:31] 1/1 0.00 2.001 2.001 271000.000	 Collimator Cont 	cuide1		
Script Editor	[08:05:31] [08:05:31] Finished at: 2020-04-01 06:05:31	▼ ⊕ cryo			
History	[08:05:31] [08:06:02] >>> [ouest 2020-04-01 06:06:02] scan(x, 0, 1, 10, t=5)	T S	2.001 K 2.001 K		
Logs	[08:06:03]	* * & detector			
	[08:06:03] Started at: 2020-01 06:06:03	ar_flow	11.29 l/min 0.000 V	idle idle	Networked Instrument COntrol Sy
Finish Experiment	[08:06:03] Sear Humber: 4 [08:06:03] Filename:00000004.dat	vacdet	5.06e-04 mbar	idle	FRM II
	[08:06:03] [08:06:03] # x T Ts det	•••• • • • • • • • • • • • • • • • • •	on		
	[08:06:03] mm K K evts [08:06:03]	ecurrent	59.879 mA	idle	Forschungs-Neutronenguel
	[08:06:03] det : Starting counting for 5 seconds [08:06:09] 1/10 0.00 2.001 2.001 270000.000	o lightshutter	4.58 MW	idle	r orsonangs Neutronenquei
	[08:06:10] det : Starting counting for 5 seconds [08:06:16] 2/10 1.00 2.001 2.001 270000.000				Heinz Majer-Leibnitz
	[08:06:17] det : Starting counting for 5 seconds [08:06:23] 3/10 2.00 2.001 2.001 270000.000	det beart_beat	- evis	Connected	
	[08:06:23] det : Starting counting for 5 seconds [08:06:29] 4/10 3:00:2:001 2:001 2:0000:000	▼ ⊕ monitors	0.44	141-	
	[08:06:30] det : Starting counting for 5 seconds	© m2	0 cts	idle	PAUL SCHERRER INSTITU
	[08:06:37] det : Starting counting for 5 seconds	o m2pos	in .	1414	
	[08:06:42] 6/10 5:00 2:001 2:001 2:000 000 [08:06:43] det : Starting counting for 5 seconds		Ucts	icie	
	[08:06:49] 7/10 6.00 2.001 2.001 270000.000 [08:06:50] det : Starting counting for 5 seconds	A P	-0 MPa	idle, below user IL.	
	[08:06:56] 8/10 7.00 2.001 2.001 269000.000 [08:06:57] det : Starting counting for 5 seconds	 o samplepot o potgate 	closed		
	[08:07:02] 9/10 8.00 2.001 2.001 269000.000 [08:07:03] det : Starting counting for 5 seconds	o vacpot	1.07e-03 mbar	idle	
	[08:07:09] 10/10 9.00 2.001 2.001 270000.000 [08:07:09]	gammashutter	open		
	[08:07:09] Finished at: 2020-04-01 06:07:09	o monolithshutter	open		NICOS - guest at localhost:1301
	[09:36:18] >>> [guest 2020-04-02 07:36:18] count(t=5)	S sits	0.00 x 0.00 mm	idle	
	[09:36:26] det - 3 det ag contrag to 5 decidas [09:36:26] det = 270000.000	• • system			
	[09:36:26] det : Starting counting for 5 seconds	v 🔮 Exp			
	[09:36:32] >>> [guest 2020+04-02 07:36:32] count(t=5)	lastpoint	0 4		perature Center Mark
	[09:36-32] det : Starting counting for 5 seconds [09:36-38] count: det = 270000.000	KafkaForwarder		idle	
		Sample		Listening	
		Detector image			
		History		20000	
		Loos		10000	
		Logs		01	
				0	
		Finish Experiment		192	
	The second s				
	No. SARA				
Jan Jaco				160	
- TH				128	
~					
				00	
State of State of State				90	
A STATE					
				64	
1					
No Com					
				32	
	test menus and test				
C 11				0	L 40 04 8
8- SUI					5° 14 39°
100	Hard Money				



worked Instrument COntrol System

Detector Data Acquisition

Time-of-flight event-based

Standardised interface:

Detector "backend master" operates as the interface to the outside world. Talks to frontend electronics via a number of rings.

Provides:

- Timing integration with frontends electronics
- High rate data path direct to DMSC/ECDC event formation
- Interface for configuration via EPICS (slow control)

Beam Monitors (single pixel detectors) use the same infrastructure.





Readout Backend

Front

Bank (

Bank 1

Software for Detectors

Detector

Converter

Event Formation and Slow Control

Event Formation

ESS Detectors run software on commercial off the shelf computers to perform online

processing of detector output. That builds the coincidences, establishes neutron positions and refines the time-stamping. It translates from detector specific readout format to a standardised event serialisation format.

In addition

- Control of the detector electronics, power supplies and other infrastructure (gases, etc)
- Imaging (ODIN) will use cameras (EPICS Area Detector with Kafka plugin) as well as timeof-flight detectors



EFU processed (and corrected/calibrated) data from LoKI tubes,

Instrument Readout Architecture







BrightnESS is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 676548

Apache Kafka

Apache Kafka is a general purpose message broker system. Built for real-time data streaming and to be scalable, faulttolerant and fast.

Scaling happens in clusters across hardware boundaries with individual brokers.

Data is divides in a number of independent topics, with different producers, consumers, retention policy and replication.

We use it as a chronological log that is always on and trigger file writing when needed (file writing can go back in time).



Kafka is agnostic to the payload of the messages.

We use a number of standardised Google Flatbuffer schemas as agreed serialisation interface between producers and receivers. For example:

- Neutron event data
- Chopper timestamps
- EPICS updates (motion, sample environment, etc)
- File writer control (including NeXus structure)

https://github.com/ess-dmsc/streaming-data-types

DAQ: Modular Setup



Central, scalable infrastructure with multiple services

- Controls and DAQ are (almost) completely separated
- File Writing makes no assumptions about control
- Neutron event data is matched to other information (choppers, motion, sample environment) via (asynchronous) timestamps
- Timestamping happens as close to the hardware as appropriate

- Forwarding of EPICS monitor data to Kafka is performed by one dedicated process per instrument
- File Writing is done by a pool of file writers not dedicated to any instrument
- Live histogramming and computing for feedback to the experiment happens via Kafka topics
- Data Processing, Analysis, etc can tap into the live data stream or be triggered by data lifecycle "events" and operate on data residing on disk

Downstream Data Architecture

Data Flow Control and Processing



BrightnESS is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 676548



Data Acquisition Asynchronous and Timestamped

Every data source sends their information independently asynchronously:

- detectors & beam monitors & cameras
- choppers
- motion
- sample environment (temperature, pressure, fields)

Data only updated on change.

Flexible, sparse efficient storage, little hardware support needed, adequate timing synchronisation is critical.

Post processing needed in most cases.



- PTP (Beckhoff: Motion, Sample Environment)
- NTP (Slow Sample Env, e.g. temperature)

How We Write Files

Creating NeXus Templates "With Ease"



· Most recent publication to cite: J. Appl. Cryst. (2015). 48, 301-305 doi:10.1107/S1600576714027575

Conclusion



- Modular setup allows scaling at various levels, if needed
- Interfaces between components defined
- NeXus (data at rest)
- NeXus mapped to JSON (scipp, data catalogue)
- FlatBuffer schemas
- Data Sources can be added or Experiment Control Software be replaced with limited integration effort

- Kafka also enables "live" ("real-time") processing as well as triggering asynchronous operations. Both options are open to users or user code.
- scipp and various data analysis tools are provided and supported by the facility.
- Data Curation is performed by the facility. Policy guarantees long term storage and provides offline processing resources.
- Deployed at ANSTO (DAQ Pipeline) & PSI (Pipeline with NICOS)

Thank you

