

Full EPICS database for
MicroBooNE Control System,
with alarms/warnings/status displays and
list of what else is needed

Glenn Horton-Smith
MicroBooNE DAQ meeting
2012-04-10

What's in this talk

- Status displays, alarms, and warnings
- PV naming convention, DB structure
- Summary spreadsheets
- Subsystems needing IOCs to be written
- Information needed for each subsystem

Subsystem status index display

The screenshot shows a window titled 'display_AlarmSum.opi' with a 'Subsystem status index' table. The table has 6 columns and 3 rows of subsystems. Each cell contains a status label and a button. The 'RackTemp' button is highlighted with a red border, and its status 'MAJOR' is also highlighted. Below the table, there is a field 'AlarmHandler HeartBeat dt' with the value '4'. A yellow callout box on the right contains three bullet points explaining the status indicators and the heart beat field.

Subsystem	Status	Subsystem	Status	Subsystem	Status
ArPurity	NO_ALARM	CalibrationSt...	NO_ALARM	CrateRails	NO_ALARM
HVC	NO_ALARM	ODH	NO_ALARM	OnDetectorP...	NO_ALARM
SEBStatus	NO_ALARM	TPCBias	NO_ALARM	TPCDrift	NO_ALARM
				TriggerStatus	NO_ALARM

AlarmHandler HeartBeat dt 4

- Status of each subsystem shown above its name.
- Pressing button switches display to detailed subsystem status.
- Time since last heart beat seen, gets yellow or red border if too big, used to verify Alarm Handler is running.

Detailed subsystem status for CrateRails

- Master alarm sum and button to go back to index. (Here it shows a severity of "major" due to RackTemp.)

display_CrateRails.opi

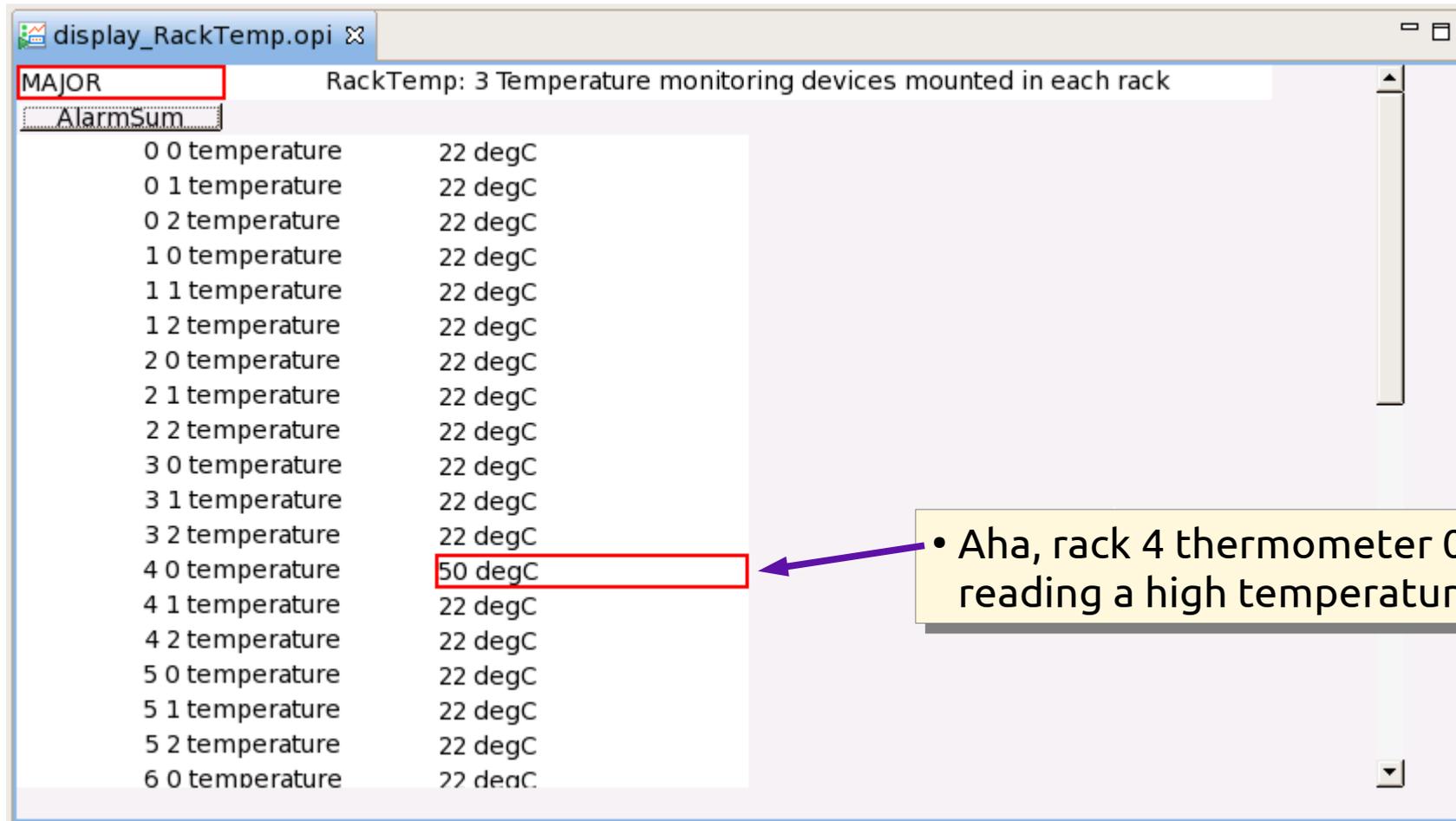
MAJOR CrateRails: Voltage, current, state, 2 units/TPC rack, 1 unit/PMT/Trig rack (PMT)

AlarmSum

0 0 current	0 A
0 0 state	off
0 0 voltage	0 V
0 1 current	0 A
0 1 state	off
0 1 voltage	0 V
0 2 current	0 A
0 2 state	off
0 2 voltage	0 V
0 3 current	0 A
0 3 state	off
0 3 voltage	0 V
1 0 current	0 A
1 0 state	off
1 0 voltage	0 V
1 1 current	0 A
1 1 state	off
1 1 voltage	0 V
1 2 current	0 A

- Scrollable list of all PV in this subsystem:
 - Crate 0 rail 0 current, state, voltage
 - Crate 0 rail 1 current, state, voltage
 - Crate 1 rail 0 current, state, voltage
 - etc....

Detailed subsystem status for RackTemp



display_RackTemp.opi

MAJOR RackTemp: 3 Temperature monitoring devices mounted in each rack

AlarmSum

0 0 temperature	22 degC
0 1 temperature	22 degC
0 2 temperature	22 degC
1 0 temperature	22 degC
1 1 temperature	22 degC
1 2 temperature	22 degC
2 0 temperature	22 degC
2 1 temperature	22 degC
2 2 temperature	22 degC
3 0 temperature	22 degC
3 1 temperature	22 degC
3 2 temperature	22 degC
4 0 temperature	50 degC
4 1 temperature	22 degC
4 2 temperature	22 degC
5 0 temperature	22 degC
5 1 temperature	22 degC
5 2 temperature	22 degC
6 0 temperature	22 degC

- Aha, rack 4 thermometer 0 is reading a high temperature.

Alarm Handler

The screenshot shows the Alarm Handler software interface. At the top, there is a window titled "Alarm Handler" with a red "MAIN" button. Below this is a menu bar with "File", "Action", "View", and "Setup". The main area is divided into two columns of alarm controls. The left column lists various system parameters with checkboxes and status indicators (R, G). The right column lists temperature sensors with checkboxes and status indicators (R, G). At the bottom, there is a status bar with execution details and control options.

- MAJOR Alarms
- MAIN <----> (0,0,1,0,757)
- ArPurity G <---->
- CalibrationStatus G <---->
- CrateRails G <---->
- Cryo G <---->
- DAQStatus G <---->
- Environment G <---->
- HVC G <---->
- ODH G <---->
- OnDetectorPower G <---->
- PCStatus G <---->
- RackFan G <---->
- RackTemp G <----> (0,0,1,0,41)
- TEST_RackTemp_M3C2/temperature <---->
- TEST_RackTemp_M4C0/temperature <----> <HIHI,MAJOR>
- TEST_RackTemp_M4C1/temperature <---->
- TEST_RackTemp_M4C2/temperature <---->
- TEST_RackTemp_M5C0/temperature <---->
- TEST_RackTemp_M5C1/temperature <---->
- TEST_RackTemp_M5C2/temperature <---->
- TEST_RackTemp_M6C0/temperature <---->

Execution Status: Global Active
Mask <CDATL>: <Cancel,Disable,noAck,noackT,noLog> H=noAck 1hr timer
Group Alarm Counts: (ERROR,INVALID,MAJOR,MINOR,NOALARM)
Channel Alarm Data: <Status,Severity>,<Unack Severity>
Filename: alh/alarm.alh

SilenceOneHour
 SilenceCurrent
Silence Forever: Off
ALH Beep Severity: MINOR

Boot/fiocW

-u:** *compilat
(No files need saving)

- Standalone process pops up this little window and blinks and beeps at you if there is an alarm.
- Pushing the blinking button brings up the main alarm handler window, where you can acknowledge the alarm and optionally do things like flag a channel to be ignored for an hour.

Classic EPICS Alarm Handler (ALH)

- Something beyond the “Index” display is needed to alert us if/when new alarms come in on top of old ones.
- I've configured the tried-and-true ALH.
 - One standalone executable, independent of CSS.
 - Display is Motif-based. Run in common VNC control window accessed by whoever is on shift.
 - Heartbeat variable on CSS-based status display makes sure it is running.
 - It was easy to set up.

What about CSS Alarm Table/Tree/Server (BEAST)

- There is a CSS-based Alarm system (“Beast Ever Alarm System Toolkit”):
 - It has more “moving parts” and “plumbing” (i.e., components and interconnections beyond basic EPICS), and seems significantly more complicated to configure and run.
 - I also found it off-putting that its manual announces “The system described in here is not used in a production environment at this time.” The manual is dated 2010, and may be out of date, but if so then I find that to be off-putting too.
- ALH meets the basic requirements for alarms and is ready to use now. With more work, it could be replaced by a CSS-based AlarmServer plus AlarmTable in display, but I don't see a requirement to do so.

Importance of PV naming convention

(PV stands for “process variable”)

- All the displays, alarm panels, and the EPICS database containing all known slow control channels (758 at last count) were generated from two spreadsheets.
 - The spreadsheets are based on the slow control and monitoring system summary table in the TDR.
- This was possible to do efficiently because of a systematic naming pattern for the PVs:

detector_subsystem_M#C#/variable

PV naming convention

(PV stands for “process variable”)

detector_subsystem_M#C#/variable

- *detector* is used to separate MicroBooNE from PAB, software development tests, etc.
- *subsystem* is “CrateRails”, “RackTemp”, etc.
- Two numbers are used for “module” (or unit, crate, etc) and “channel”.
- *variable* can be “voltage”, “current”, etc.
- **Examples:** PAB_HVC_M0C0/VOLT
TEST_RackTemp_M4C0/temperature

Summary spreadsheets

uB-control-channels-primary.csv

- Defines the names, types, descriptions, and number of units and channels for each subsystem.

uB-control-channels-pvar.csv

- Defines the variables in each channel for a given type, and optionally information such as units of measure (EGU), warning and alarm limits (LOLO, LOW, HIGH, HIHI), and other EPICS information.

These spreadsheets are deliberately simple.

3 sample rows from uB-control-channels-primary.csv

subsystem	Primary type	Short Description	# units	# ch/unit	# var/ch	Long Description
OnDetectorPower	RdOnlyPwrSuppMPOD	POWER LVPS	6	8	3	Voltage, current, state of power supply rails for on-detector electronics
CrateRails	RdOnlyPwrSuppWienerPL508	Rack LVPS - rails	10	4	3	Voltage, current, state, 2 units/TPC rack, 1 unit/PMT/Trig rack (PMT Lambda via VME)
RackTemp	DS1624	Rack temp.	14	3	1	3 Temperature monitoring devices mounted in each rack

See latest version of the full file in [EPICS/make_db/uB-control-channels-primary.csv](#) in the [git master repository](#).

[<https://cdcvs.fnal.gov/redmine/projects/ubooneDAQ/repository/revisions/master/show>]

One example primary type PV definition in `uB-control-channels-pvar.csv`

Environment	ai	temperature	EGU=degC	LOLO=10	LOW=16	HIGH=32	HIHI=40
	ai	humidity	EGU=%	LOLO=8	LOW=20	HIGH=70	HIHI=80
	ai	pressure	EGU=mbar				
	ai	linevoltage	EGU=Vrms	LOLO=104.4	LOW=114	HIGH=126	HIHI=127.2

Note it is possible, but not required, to use this file to define properties for every PV in a subsystem, such as “EGU” (engineering units) or alarm and warning levels. Individual channel values can be set separately. This file is not designed for specifying individual channel values.

See full file in [EPICS/make_db/uB-control-channels-pvar.csv](https://cdcvs.fnal.gov/redmine/projects/ubooneDAQ/repository/revisions/master/show) in the [git master repository](https://cdcvs.fnal.gov/redmine/projects/ubooneDAQ/repository/revisions/master/show). [<https://cdcvs.fnal.gov/redmine/projects/ubooneDAQ/repository/revisions/master/show>]

Complete list of subsystems and primaries

• ArPurity	• ArPurity
• CalibrationStatus	• CalibrationStatus
• Cryo	• Cryo
• DAQStatus	• DAQStatus
• RackTemp	• DS1624
• Environment	• Environment
• HVC (PMTs)	• HVBiRa
• RackFan	• kTeVfanPack
• ODH	• ODH
• PCStatus	• PCStatus
• TPCDrift	• RdOnlyPwrSuppGlassman
• OnDetectorPower	• RdOnlyPwrSuppMPOD
• TPCBias	• RdOnlyPwrSuppWienerPL508
• CrateRails	• SEBStatus
• SEBStatus	• TriggerStatus
• TriggerStatus	

(16 subsystems so far) (15 primary device types)

Subsystems needing IOCs to be written

- RdOnlyPwrSuppGlassman
- RdOnlyPwrSuppMPOD
- RdOnlyPwrSuppWienerPL508
- DS1624
- kTeVfanPack
- Environment
- ArPurity
- CalibrationStatus
- Cryo
- ODH
- SEBStatus
- TriggerStatus

Basically every one except the BiRa PMT HV controller and the DAQStatus and PCStatus “virtual” IOCs.

More information needed for each subsystem

- (0) Contact person for each subsystem.
- (1) Confirmation that the list of control system variables for each subsystem is complete, or corrections as needed.
- (2) Ranges for “yellow” and “red” warnings, and “delta values” for triggering update of display and archive, for each variable.
- (3) Who is going to write the IOC for each subsystem, if different from contact.