aLIGO Guardian Status Update

Jameson Graef Rollins

aLIGO SYS meeting
September 9, 2014

LIGO-G1400891

Overview

This is an overview of Guardian deployment status, and of the primary remaining tasks.

Outline:

- Guardian core deployment status
- Usercode deployment status
- Remaining tasks
- Proposal for improved SUS alignment offset handling

Relevant other documents:

- Guardian Progress and Remaining Tasks (T1400461)
- Guardian Overview and User Manual (G1400016)
- Guardian awiki

Guardian core deployment status

Guardian core

The core Guardian code base is composed to two packages:

guardian
: Core system graph and state execution system and libraries, command line interface, helper utilities, MEDM control interface, supervision interface, etc. Interface and behavior has stabilized; adding new features as needed.
LLO: v1063, LHO: r1083

ezca/cdsutils
: Ezca customized EPICS client library and other useful CDS interaction utilities (servo, step, avg, etc.).
Adding/improving functions as needed.
LLO: r274, LHO: v322

Both packages installed at both sites and all supporting facilities.

Site infrastructure

Guardian node supervision infrastructure is in place at both sites on dedicated machines. `guardctrl` interface is used for controlling nodes (creating, starting, stopping, listing, etc.), as well as viewing node logs.

```
jameson.rollins@opsportal:~ 0$ guardctrl list
node         s k m vers  state                                message
----         - - - ----  -----                                -------
HPI_BS       * * M 1060  READY
HPI_ETMX     *   P 1060  WATCHDOG_TRIPPED_FULL_SHUTDOWN       WATCHDOG TRIP: FULL_SHUTDOWN (4)
HPI_ETMY     * * M 1060  READY
HPI_ITMX     * * M 1060  READY
HPI_ITMY     * * M 1060  READY
IAS_INPUT    *   E 1060  INIT
IAS_MICH     *   E 1060  INIT
IAS_PRC      *   E 1060  INIT
IAS_SRC      *   E 1060  INIT
IAS_XARM     *   E 1060  IDLE
IAS_YARM     *   E 1060  IDLE
IFO_ALIGN    *   E 1060  IDLE
IFO_DRMI     *   E 1060  DOWN
IFO_IMC      * * E 1060  LOCKED
IFO_LOCK     *   E 1063  IDLE
IFO_OMC      *   E 1060  DOWN
ISI_BS_ST1   * * M 1060  HIGH_ISOLATED
ISI_BS_ST2   * * M 1060  DAMPED
ISI_ETMX_ST1 *   P 1060  WATCHDOG_TRIPPED_FULL_SHUTDOWN       WATCHDOG TRIP: FULL_SHUTDOWN (4)
ISI_ETMX_ST2 *   P 1060  WATCHDOG_TRIPPED_FULL_SHUTDOWN       WATCHDOG TRIP: FULL_SHUTDOWN (4)
```

Recent core improvement: state tracking

Recently improved state and status tracking ability for all nodes:

- persistent state indices: numeric index for states can be manually specified, and remains persistent over time.
- `NOMINAL` state definitions indicate the expected state of the node during observation.
- `OK` status bit for each node indicates overall node status (`MODE` = EXEC, `STATE` = `NOMINAL`, `REQUEST` = `NOMINAL`, `STATUS` = DONE, `ERROR` = False, etc.)

Recent core improvement: state tracking

Recently improved state and status tracking ability for all nodes:

- persistent state indices: numeric index for states can be manually specified, and remains persistent over time.
- `NOMINAL` state definitions indicate the expected state of the node during observation.
- `OK` status bit for each node indicates overall node status (`MODE` = EXEC, `STATE` = `NOMINAL`, `REQUEST` = `NOMINAL`, `STATUS` = DONE, `ERROR` = False, etc.)

Usercode deployment status

Usercode deployment

Deployment of "usercode", i.e. actual automation logic, written by subsystem leads and commissioners (and Jamie), is progressing nicely:

- SUS and SEI subsystems essentially complete at both sites (with caveats)
- IMC auto-lockers working at both sites
- Full IFO locking node under development at LLO, soon at LHO
- DRMI, OMC, ALS locking nodes under development at LHO
- prototype of initial alignment system at LHO

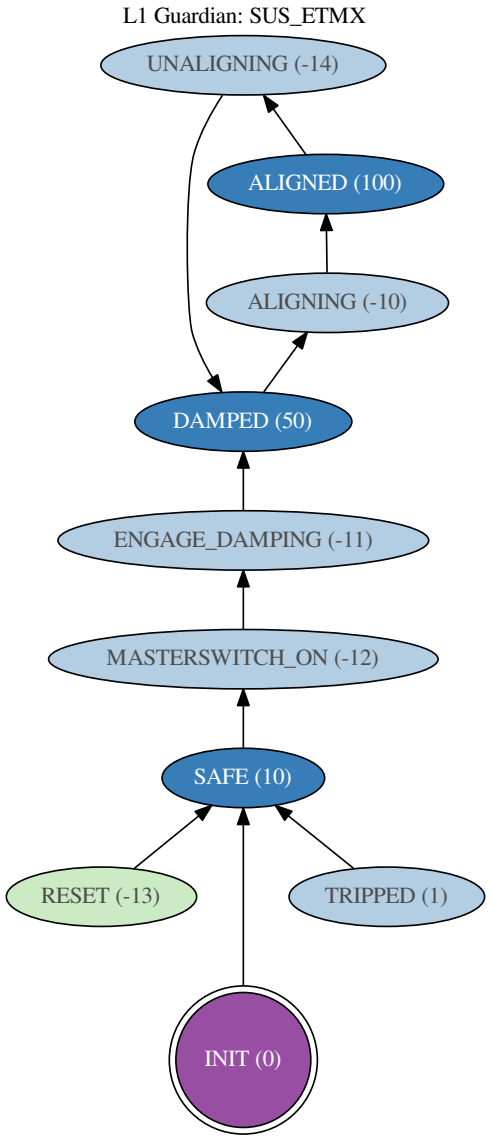
SUS and SEI systems are well parameterized, with ability to customize individual systems as needed.

SUS fully deployed

SUS was the simplest subsystem to get automated. All suspensions nodes have identical guardian structure, using a common, base SUS.py guardian module and the sustools.py library interface.

Handles full recovery to aligned state from watchdog trips.

Handling of alignments has needed continual refinement. Different techniques preferred: LHO wants separate ALIGNED/MISALIGNED states, LLO does not. See Section 4 for discussion.

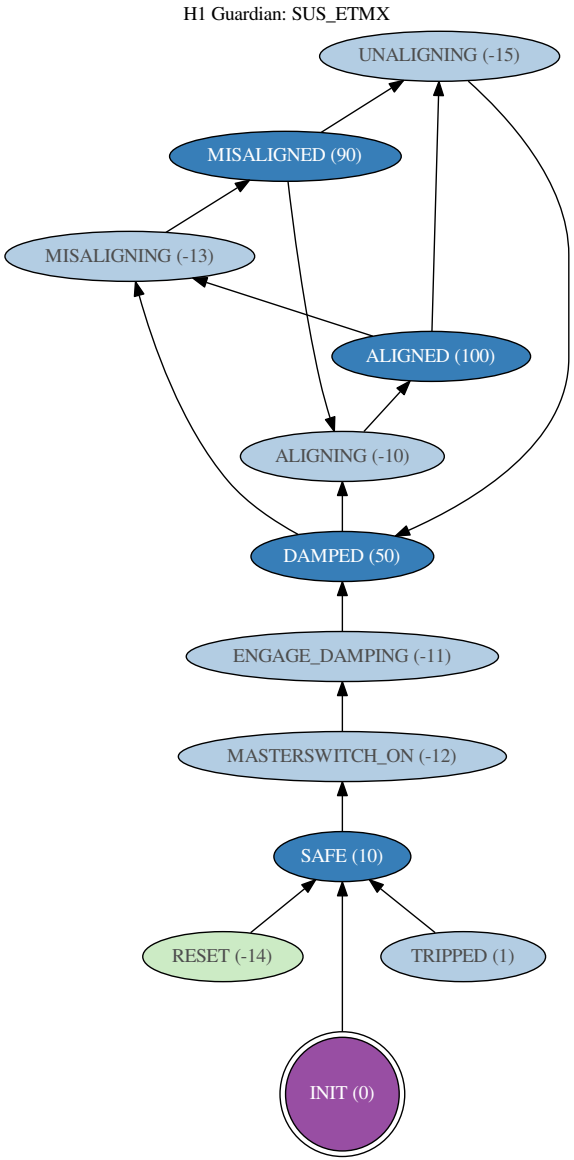


SUS fully deployed

SUS was the simplest subsystem to get automated. All suspensions nodes have identical guardian structure, using a common, base SUS.py guardian module and the sustools.py library interface.

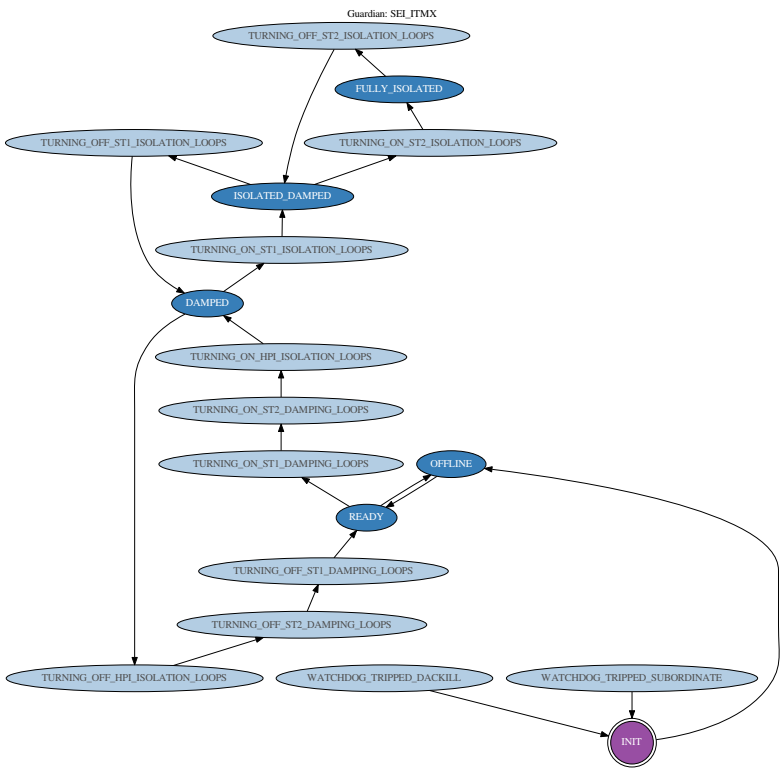
Handles full recovery to aligned state from watchdog trips.

Handling of alignments has needed continual refinement. Different techniques preferred: LHO wants separate ALIGNED/MISALIGNED states, LLO does not. See Section 4 for discussion.

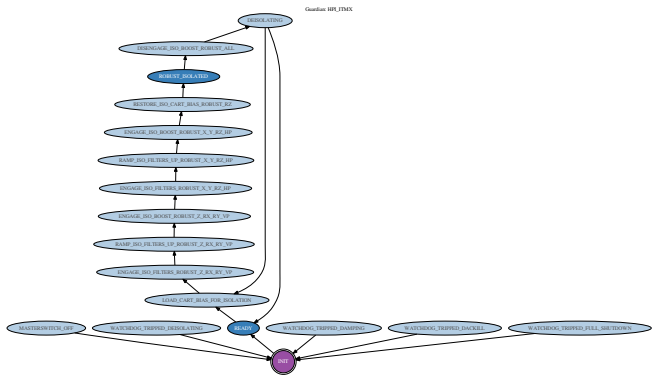


SEI fully deployed

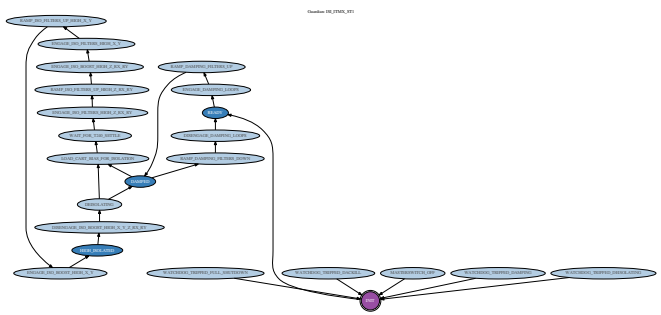
SEI systems fully automated, for both HAMs and BSCs. Automatic watchdog trip recovery to full isolation. “Chamber managers” coordinate isolation between HPI and ISIs nodes (HAM: 3 nodes, BSC: 4 nodes).



SEI_ITMX (manager)



HPI_ITMX

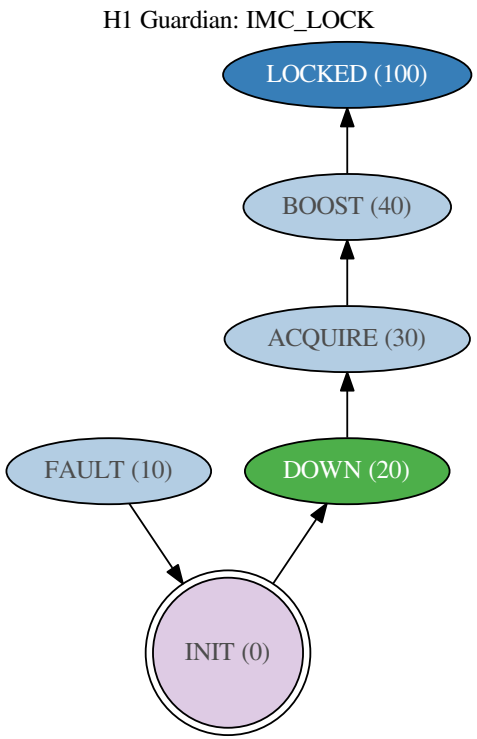


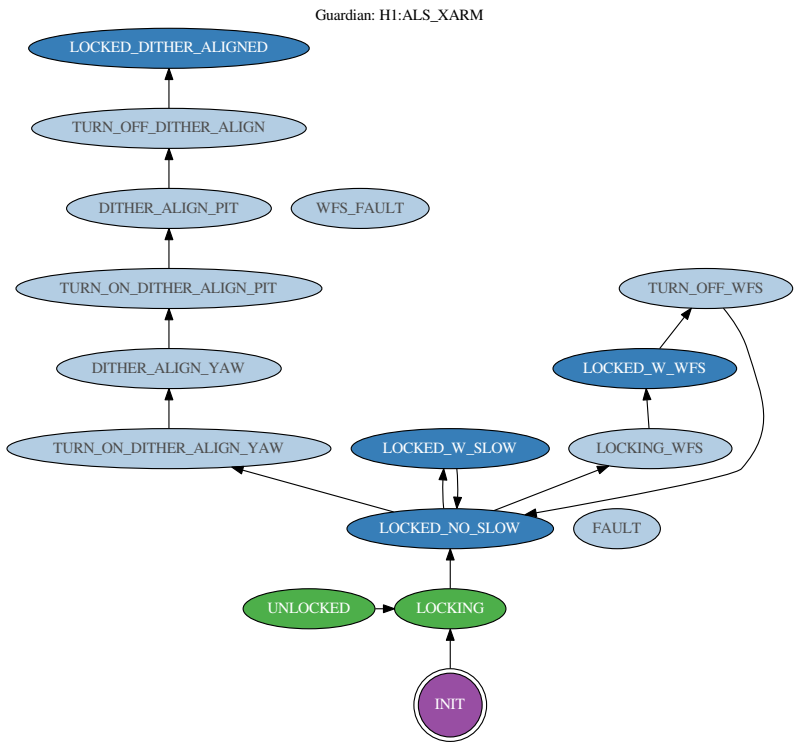
ISI_ITMX_ST{1,2}

IMC and ALS

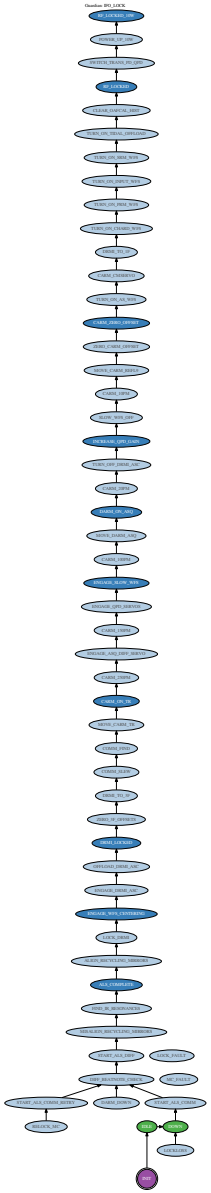
IMC_LOCK IMC auto-locker (first cavity control) deployed at both sites. Some site differences need to be reconciled.

ALS nodes (**XARM**, **YARM**, **COMM**, **DIFF**) used at LHO, but not at LLO.





Full IFO locking



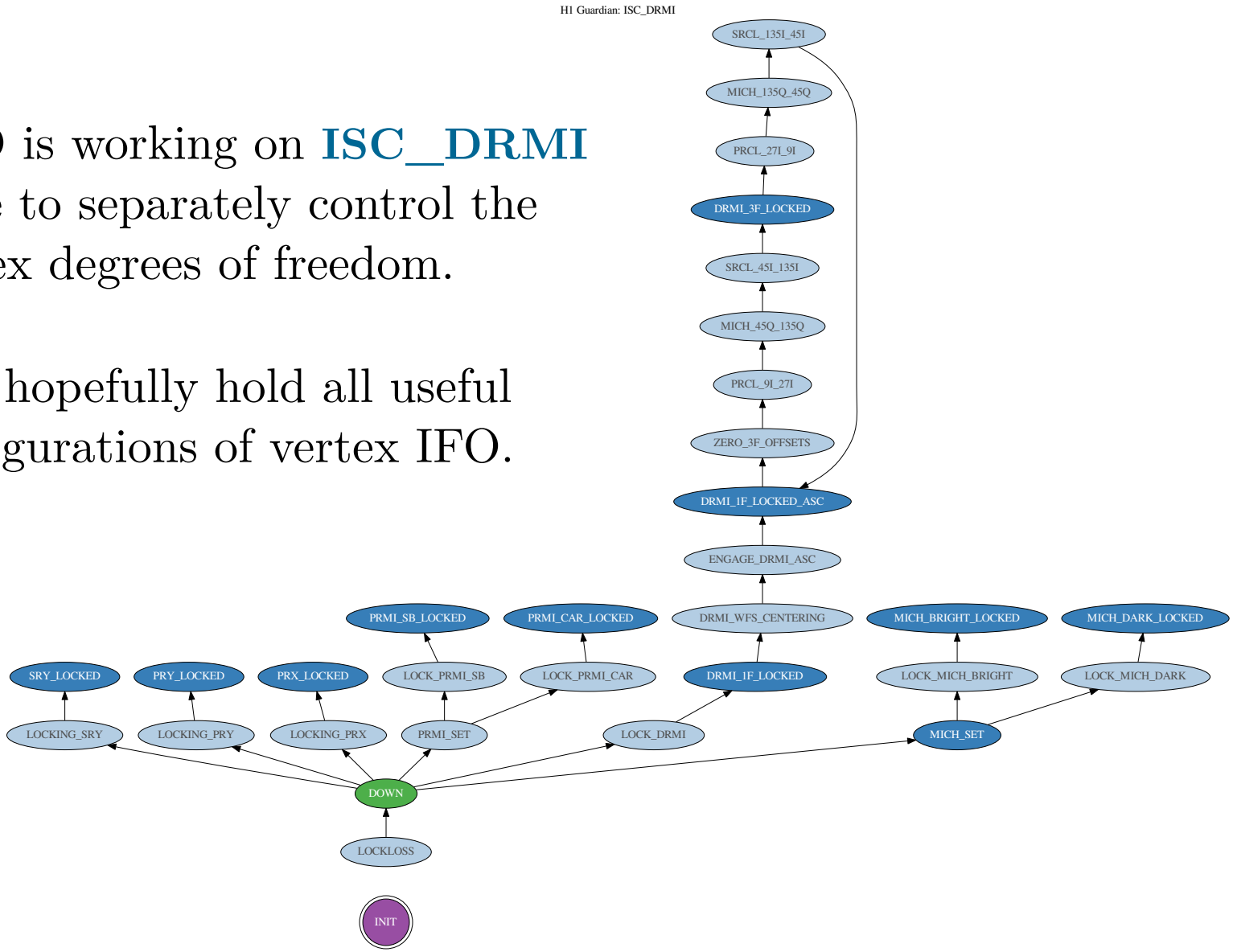
IFO_LOCK: full IFO locking node at LLO. Handles full RF lock: ALS control of the arms, DRMI lock, CARM and DARM transitions, and some amount of ASC integration. No DC readout transition yet.

Need to eliminate all external script calls, and parameterized the code so it can be shared between sites.

DRMI

LHO is working on **ISC_DRMI** node to separately control the vertex degrees of freedom.

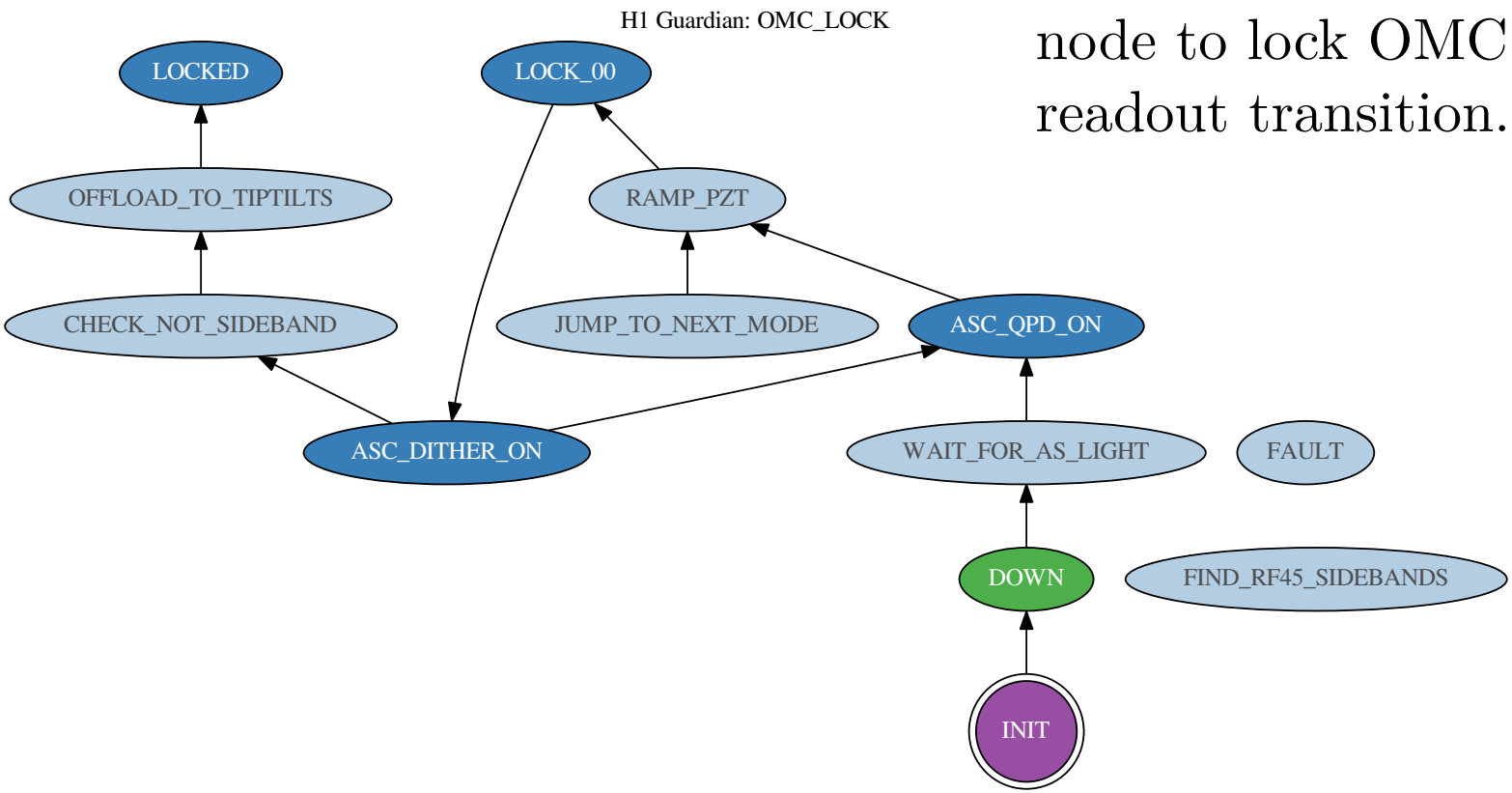
Can hopefully hold all useful configurations of vertex IFO.



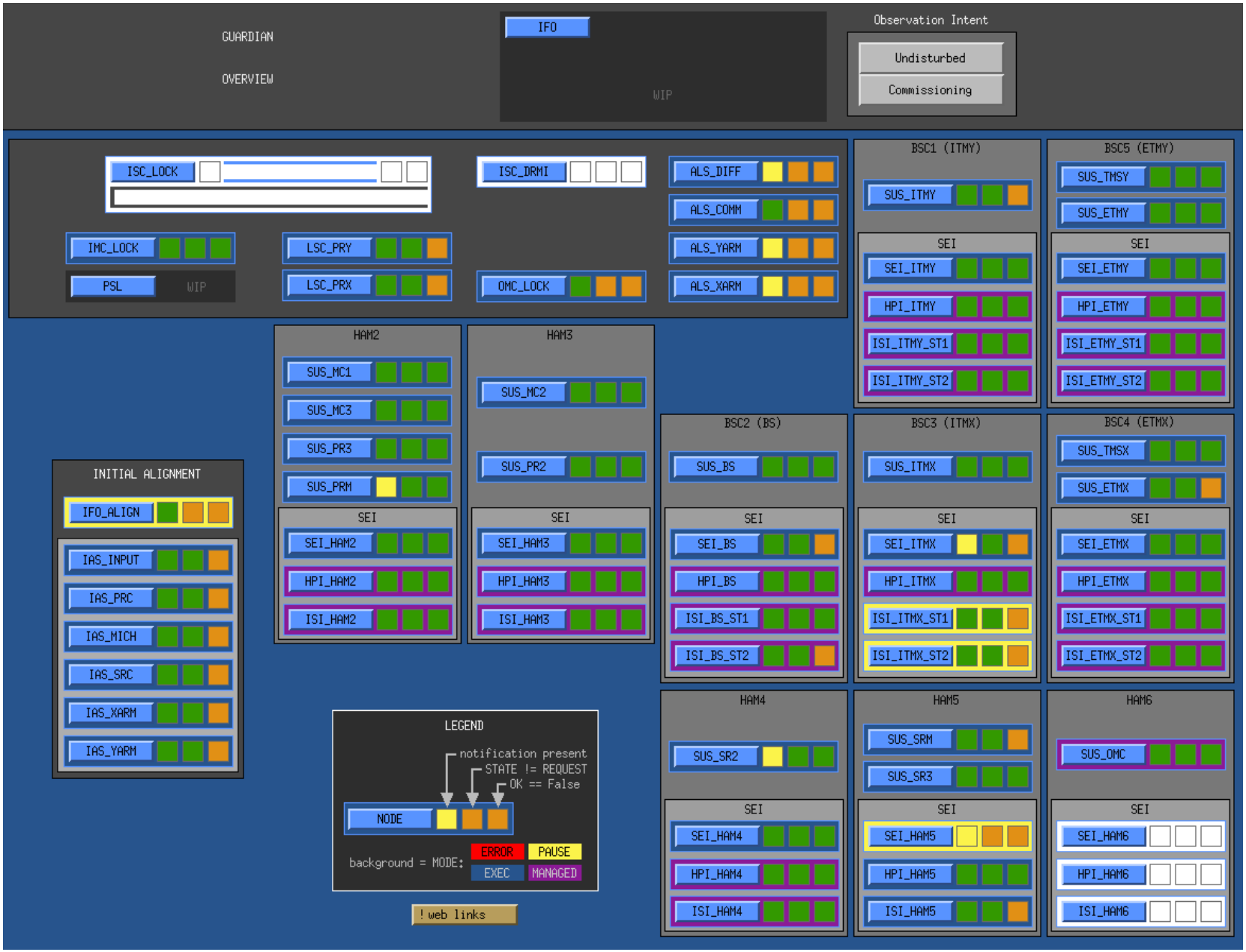
OMC

OMC_LOCK node at LHO based on initial work done at LLO.

Idea is for main LOCK node to use ALS and DRMI nodes to handle initial arm and vertex locks, and OMC node to lock OMC for DC readout transition.



Full system overview



Remaining tasks

Remaining tasks: Project scope

Still a lot left to be done.

The following tasks were (somewhat arbitrarily) put under the scope of aLIGO Project:

- RTS/Beckhoff parameter/settings monitoring: **UNDER DEVELOPMENT**
- node state tracking: **DONE**
- critical bugs: **ONGOING** (bugzilla)
- documentation: **ONGOING**

Parameter/settings monitoring

The current inability to track system parameters, i.e. EPICS settings, has been a known problem. Multiple solutions have been put forth:

EPICS alarms: original guardian concept was to set EPICS alarms for all settings. Unfortunately, EPICS alarms don't support alarming on exact values, so this option is onerous to setup and maintain. Limited testing, never widely deployed.

control state definitions: Sigg et. al. (LIGO-T1300958, LIGO-G1400384), employs external (guardian) processes that continually write values to settings channels. Problematic, since CSD processes overwrite user modifications without user feedback. Tested at LHO.

Parameter/settings monitoring

front-end settings definitions: Bork/Rollins (LIGO-T1400280, LIGO-G1400742), front-end IOC sequencers hold table of setpoint values and raise notifications when monitored settings change. Requires only specifying which channels are to be monitored in the existing safe.snap files. Does not prevent changing settings; only alerts commissioners that settings have changed. Also allows for *much* faster "BURT" restores. Currently in testing. Beckhoff integration still needs to be worked out.

Hope to "down select" a system in the next month or so.

Remaining Operations tasks: ISC

The following tasks, while still necessary for proper operation of the instruments, have been scoped to Operations:

initial alignment
: Commission **IFO_ALIGN** and **IAS** initial alignment nodes at LHO.

better ALS integration
: **ALS** nodes developed at LHO should be integrated with (managed by) the main **ISC_LOCK** node, and deployed at LLO.

Remaining Operations tasks: ISC

DRMI node

ISC_DRMI node being developed at LHO should be integrated with (managed by) the **ISC_LOCK** node, and deployed at LLO.

OMC node and DC readout transition

ISC_OMC node being developed at LHO should be integrated with (managed by) the **ISC_LOCK** node, and deployed at LLO.

Remaining Operations tasks: SUS

SUS optical lever damping

The op-lev damping loops are a bit tricky as they need to be turned on and off depending on the op-lev beams on the QPDs. Some handling of this has been worked on at LLO, but it needs to be refined, and deployed to LHO.

SUS alignment offsets

Tracking of suspension alignment state is currently difficult and clunky. We need a better way to track/manage alignment offsets, with better guardian integration. See Section 4.

small optic suspensions

No guardians for HSSS suspensions yet.

Remaining Operations tasks: SEI

SEI respond to changing seismic environment

Working on SEI "bland" node to transition blend filters based on current seismic conditions (earthquakes, high microseism, high winds, etc.). Big questions remain, such as should such transitions happen while loosing "science" mode operation.

earthquake early warning/response

An earthquake early warning system is under development. We will hopefully be able to use this to preemptively put the SEI systems into more robust configurations to help locks ride out the earthquakes. See LIGO-G1400811.

Remaining Operations tasks: other

thermal compensation

TCS system has not been automated yet, although work has begun.

pre-stabilized laser

No PSL guardian integration.

Remaining Operations tasks: global

fully integrate all automation code

Move all automation logic into "guardian native" code, eliminating external script calls. External scripts do not use the managed EPICS interface; forking sub-processes is problematic from a process management and complexity standpoint; code is more difficult to debug, maintain, audit, etc.; overall less robust. There's no reason this can't be easily achieved; external scripts mostly only exist because they were written before guardian was deployed.

full management integration

Node management, whereby one node controls one or more other nodes, is currently only used in SEI. The full management hierarchy needs to be commissioned, from top node down to device nodes. The core manager interface will likely need more development to unsure robustness.

Remaining Operations tasks: global

support multiple IFO configurations

Want to be able to easily transition between various IFO configurations, e.g. DRMI only, single arm only, PRMI+arm, etc. Still working out how best to integrate these options into the larger structure.

high power operation

Transition to high power operation, and locking at various IFO thermal states, will be very tricky.

Remaining Operations tasks: global

ODC integration

Need better (any) ODC integration for guardian state reporting.

reconcile site differences

As much as possible, we would like all sites to be running the same automation code. This is achievable, to the extent that the operational configurations of the instruments are commensurate, if the code is well parameterized. Currently many differences exist: IMC lock procedure, loosely commissioned ALS automation at LLO, etc.

Remaining Operations tasks: global

tasks

Better support for specific measurement tasks, such as calibration measurements; related to alternative configurations support above.

various core improvements

An ever-growing list of desired features and minor bugs for the guardian core.

Issues with SUS alignment offsets

Issues with SUS alignment offsets

Tracking of alignment offsets in the suspensions has turned out to be quite tricky and difficult to manage.

Suspension offsets are unlike any other settings: their "correct" values are expected to change frequently over time, as well as during a single global initial alignment.

Currently, offsets are stored in text files in the the USERAPPS repo, but are rarely committed.

Global alignment MEDM screens have buttons that "save" and "restore" to stored offset values, but with different methods used between sites (LLO: save only current; LHO: save "align"/"misalign"))

The problems

There are a couple notable issues with our current offset handling:

- no way to know if current offsets correspond to ALIGNED or MISALIGNED state
- relies on remembering to save offsets to text files that are not well managed. If user forgets to save, alignment can be lost.
- no way to track ALIGNED vs. MISALIGNED offset values over time.

Guardian interaction

Have attempted two different methods for handling offsets in Guardian:

- SUS guardians have both ALIGNED and MISALIGNED states. Moving to the requested state sets the offsets appropriately based on *stored* values.

  Problem: if offsets are not saved, alignment state changes or trip recoveries will cause alignments to be lost.

- SUS guardians have only single ALIGNED state. Offsets handled entirely by global alignment code.

  Problem: no indicator or record of alignment state; no way to locally tell a SUS to go to a known, valid misaligned state;

Proposal to address the issue

ECR E1400107(integration bugzilla 746)

Store ALIGNED and MISALIGNED offset values in separate EPICS records, with switch to switch between them.

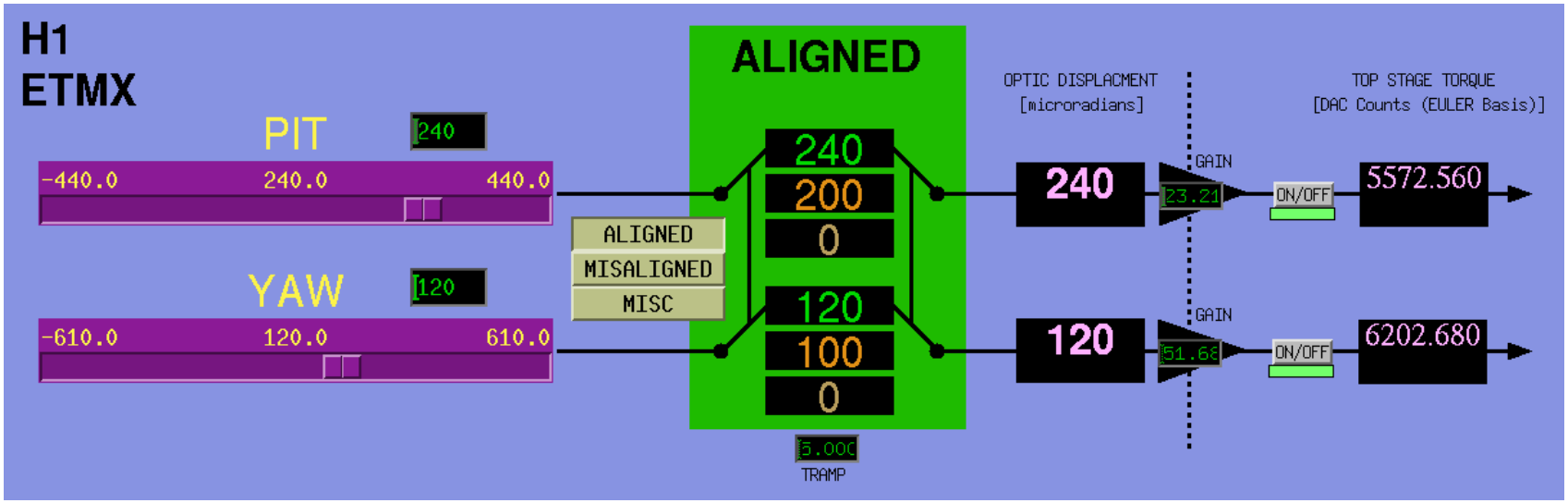
- Setting offsets would be done on a per-state basis.
- Offsets for each state would always be up-to-date, and stored separately in frames. **Would allow for previously unavailable trending of ALIGNED offset values**.
- SUS guardians could have ALIGNED and MISALIGNED states that would simply flip the alignment switch appropriately, with no worry that it would restore to out-of-date values.
- Simply select MISALIGNED to go to known good misalignment value, without having to touch offsets. Trivially restore to last ALIGNED value.

Proposed changes

Changes to be made:

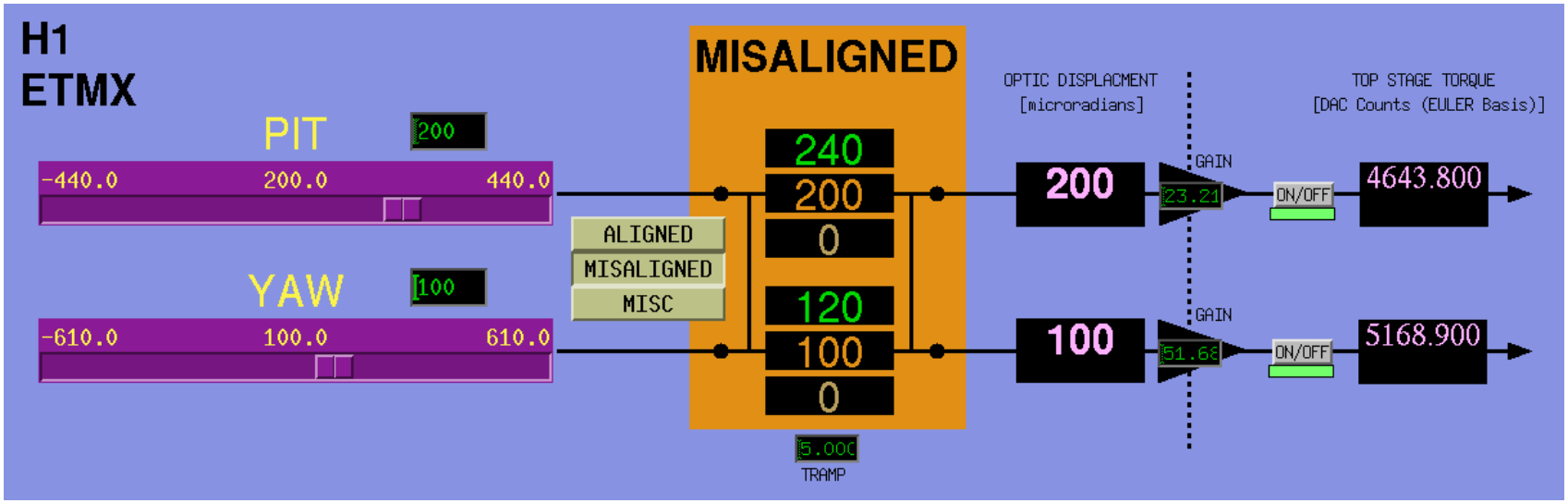
- Break out OPTICALIGN front-end logic into separate common OPTICALIGN library block for all suspensions (should be done anyway), with corresponding common OPTICALIGN MEDM screen.
- Add new PIT/YAW records for alignment states (ALIGNED/MISALIGNED/MISC), with a ganged ramping switch to handle switching PIT and YAW offsets simultaneously.
- Implement switch logic that writes appropriate offset into common OFFSET slider upon switching, and records current slider value into appropriate alignment record.
- Modify all SUS models to point to new OPTICALIGN library block.

MEDM interface



A single channel/slider can be used to control the current offset value, with a switch to select which alignment state the current value corresponds to.

MEDM interface



A single channel/slider can be used to control the current offset value, with a switch to select which alignment state the current value corresponds to.

Discussion

- Saving offsets to disk has one advantage, which is that the last value can easily be recovered after sudden shutdown (power outage, etc.). That said, there's no reason we can't instead make script to recover values from frames/conlog.
- Do we need more than two alignment states? I propose three (ALIGNED, MISALIGNED, MISC), but maybe it would be useful to have more.
- Is there any reason that MISLIGNED values can't be set statically? Do they ever need to be set relative to current alignment values?