aLIGO Guardian

Jameson Graef Rollins

LIGO Laboratory
California Institute of Technology

EPICS Collaboration Meeting
2015 ICALEPCS
Melbourne, Victoria, Australia
October 17, 2015

Introduction

Advanced LIGO has developed a novel automation platform called **Guardian**.

The objective of Guardian is to provide:

- a framework for complete, robust automation of the LIGO interferometers and all subsystems.
- an interface that facilitates the unique commissioning process.
- useful diagnostics and coherent tracking of the full state of the instrument to aid detector characterization.

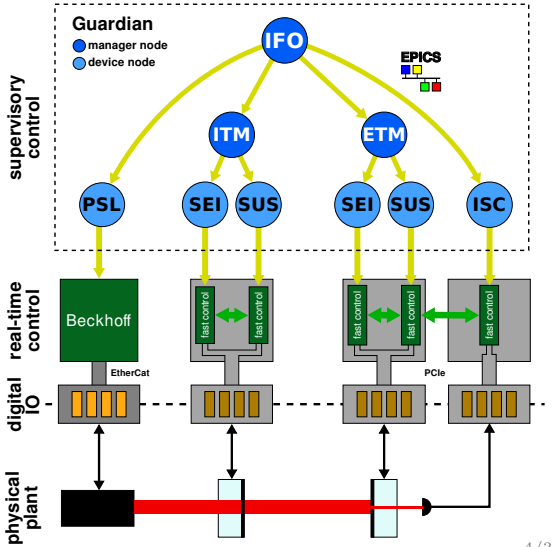
Guardian is now mature, fully deployed at both sites, and in full control of both detectors.

Conceptual Overview

Guardian design concept

Guardian is a **hierarchical, distributed, state machine**.

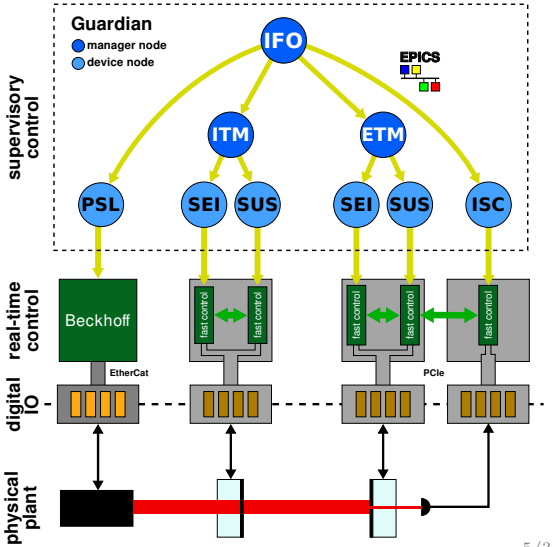
Individual automaton **nodes** oversee well defined sub-domains of the interferometer.



Guardian design concept

A hierarchy of nodes control the full interferometer.

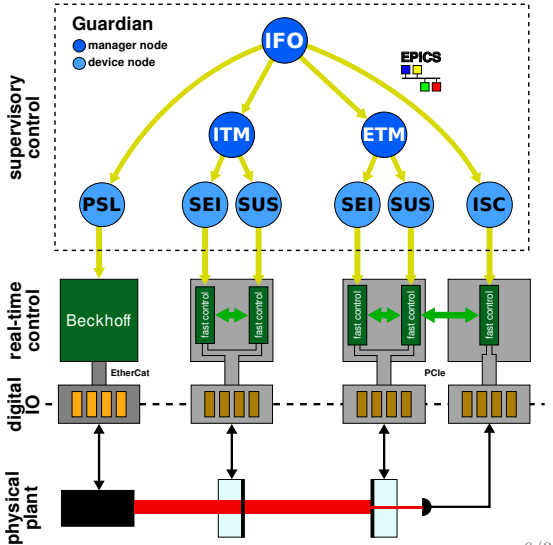
Upper-level **manager** nodes control lower-level **subordinate** nodes, with **device** nodes talking directly to front end hardware.



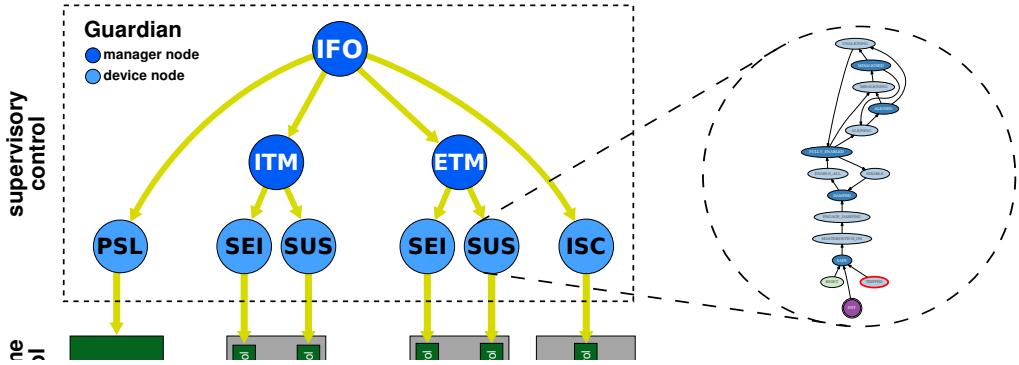
Guardian design concept

All communication is handled by EPICS.
It handles communication:

- between nodes.
- between nodes and the real-time front ends.
- between operator interfaces and the nodes.
- for node status acquisition.



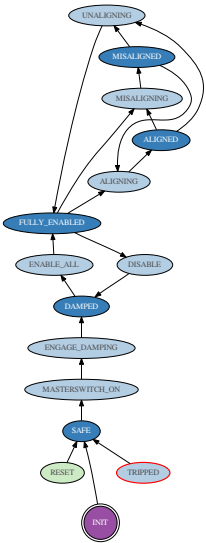
Guardian design concept: nodes



Each node executes a **state graph** for its system.

Guardian design concept: state graphs

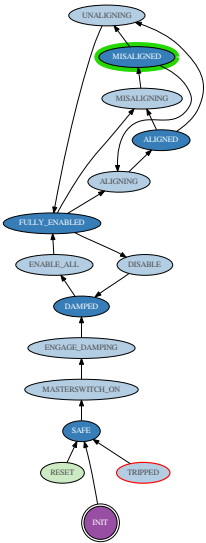
The **state graphs** describe the accessible states of the system, and the allowable transitions between states.



Guardian design concept: state graphs

The **state graphs** describe the accessible states of the system, and the allowable transitions between states.

The node accepts commands in the form of a **state request**.

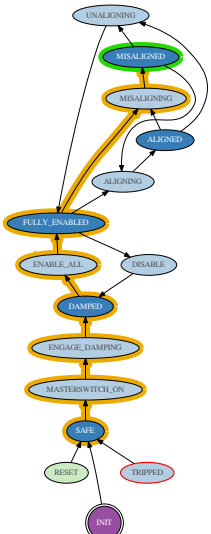


Guardian design concept: state graphs

The **state graphs** describe the accessible states of the system, and the allowable transitions between states.

The node accepts commands in the form of a **state request**.

Guardian then calculates the path from the current state to the requested state, and executes all states in the path in sequence.

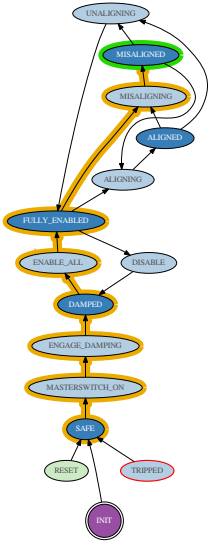


Guardian design concept: state graphs

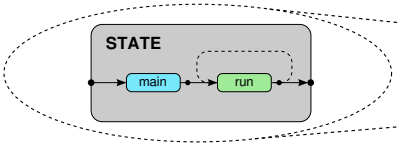
Guardian is a **finite state machine** (FSM): each state is a logically distinct block of code.

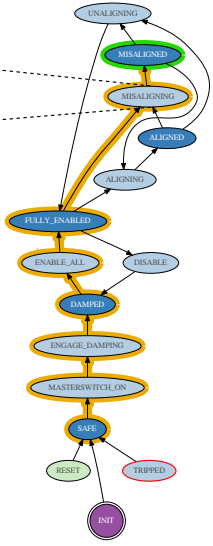
The FSM design forces all persistent process variables to be *external* to the code, in this case stored in `EPICS` records that are fully recorded by the data acquisition system.

The full state of the automation system at any point in time can then be **completely reconstructed** from data on disk.



Guardian design concept: states





Guardian nodes are **soft real-time systems** that employ a timed *run loop* that executes state code.

For each state the *main* method is run once upon entrance, then the *run* method runs continuously to check for exit conditions.

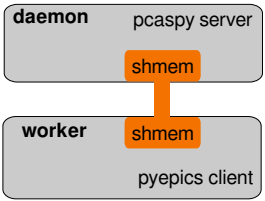
When a state is *complete* guardian transitions to the next state in the path.

Features

Features: architecture and EPICS usage

Guardian uses two great EPICS Python bindings:

- pyepics 3.2.1
- pcaspy 0.4.1



Guardian utilizes a daemon+worker architecture:

- python `multiprocessing` package)
- The **daemon** is the main process that launches and manages the worker process, and uses pcaspy for the control interface. The daemon loads the system state graph, and processes user state requests, and calculates paths.
- The **worker** receives a state/method execution command from the daemon, and executes the user code.
- The daemon can terminate the worker as needed.
- Communication between daemon and worker is done via **shared memory**.

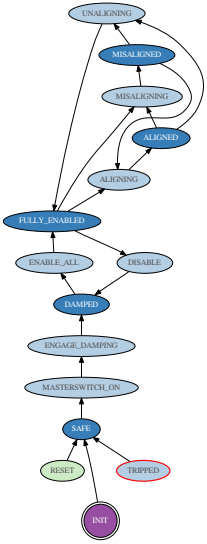
Features: programming and visualization

User code is standard Python

User code, i.e. actual automation logic, is written in standard Python, with no special syntax and full access to all Python functionality. Since logic is broken up by state, the code is very modular and accessible.

Dynamic graph generation

System state graphs can be generated dynamically, providing immediate visual representations of the logic that are useful for development and reference.



Features: user code reloading and archiving

On demand, on the fly user code reloading

User code can be reloaded dynamically without requiring recompilation or affecting current execution. All errors and exceptions are caught and reported to the user via the logs.

Automatic code archiving

All user code is automatically committed to per-node code archives (using `git` SCM) upon load. This provides a **full, permanent record of all running code** at any given point in time.

Features: EPICS control and status interface

Guardian uses EPICS to provide a standardized control interface. This is used for commissioner/operator interaction, intra-node communication and management, and status archiving.

The interface provides **full node state and status reporting**:

- requested state
- currently executing state
- whether the system has arrived at the requested state
- whether the system has reached the ultimate "NOMINAL" operating state
- current operating mode, execution status, internal state, connection and setpoint status, manager information, etc.

Features: EPICS control and status interface

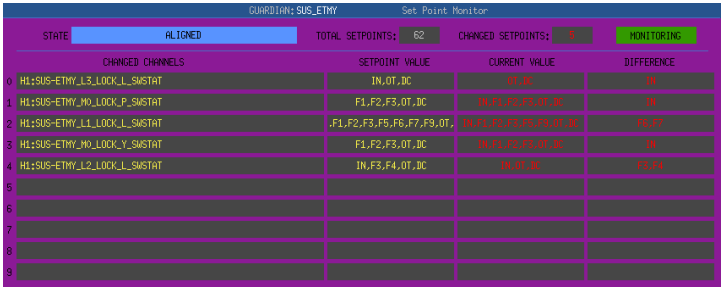
Guardian uses EPICS to provide a standardized control interface. This is used for commissioner/operator interaction, intra-node communication and management, and status archiving.

The interface provides **full node state and status reporting**:

- requested state
- currently executing state
- whether the system has arrived at the requested state
- whether the system has reached the ultimate "NOMINAL" operating state
- current operating mode, execution status, internal state, connection and setpoint status, manager information, etc.

Features: monitoring and diagnostics

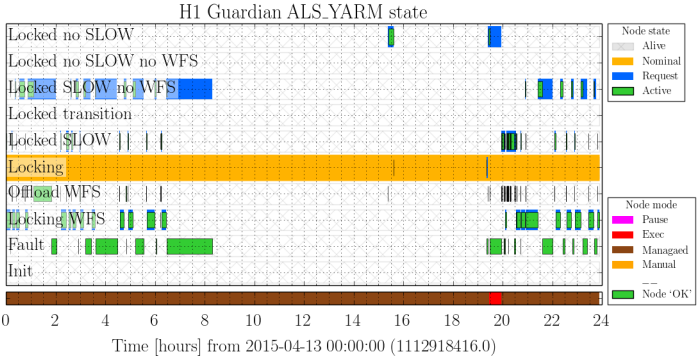
All nodes maintain persistent connections to their front end systems. **Connection status is checked continuously**, and appropriate errors are raised if connections are lost.



Channel access writes are recorded internally, and nodes can **automatically check all set points during execution** and either alarm or fault upon detection of external changes.

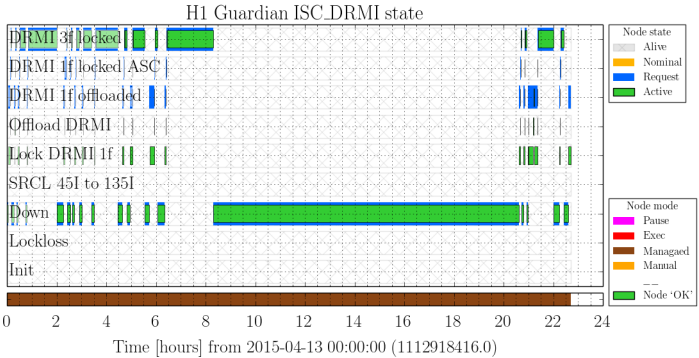
Features: state tracking and reporting

The full state and status of all Guardian nodes are recorded by the data acquisition system.



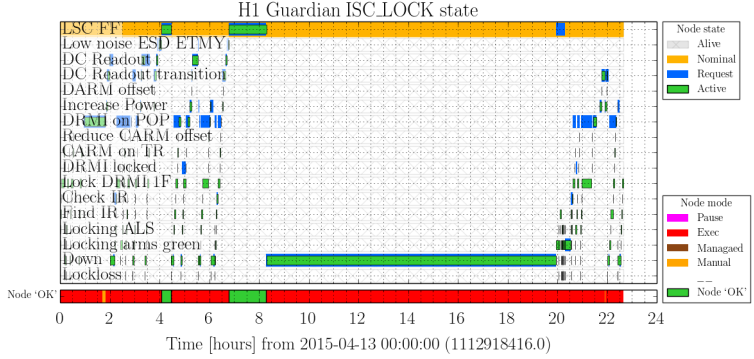
Features: state tracking and reporting

The full state and status of all Guardian nodes are recorded by the data acquisition system.



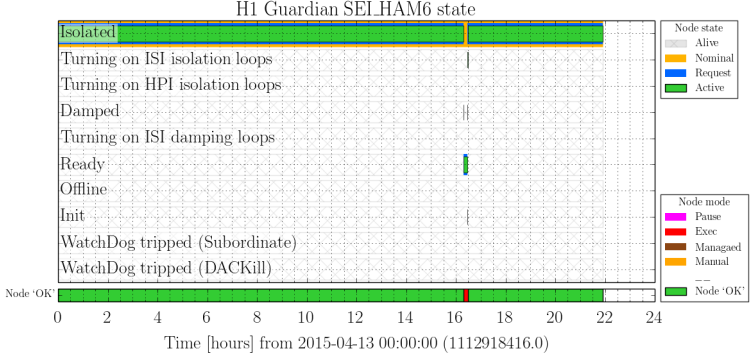
Features: state tracking and reporting

The full state and status of all Guardian nodes are recorded by the data acquisition system.



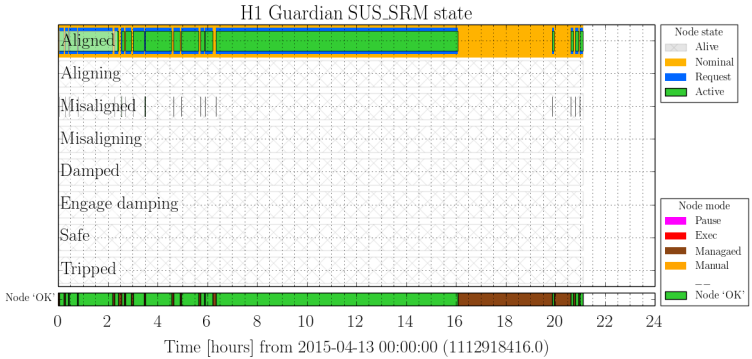
Features: state tracking and reporting

The full state and status of all Guardian nodes are recorded by the data acquisition system.



Features: state tracking and reporting

The full state and status of all Guardian nodes are recorded by the data acquisition system.



Additional useful tools

Many useful command line and graphical tools are available, to e.g.:

- automatically draw system graphs
- analyze system graphs
- inspect/edit user code
- directly execute code for individual states, or for arbitrary state graph paths
- interactive *guardian shell* environment
- plot guardian status over time

Site infrastructure

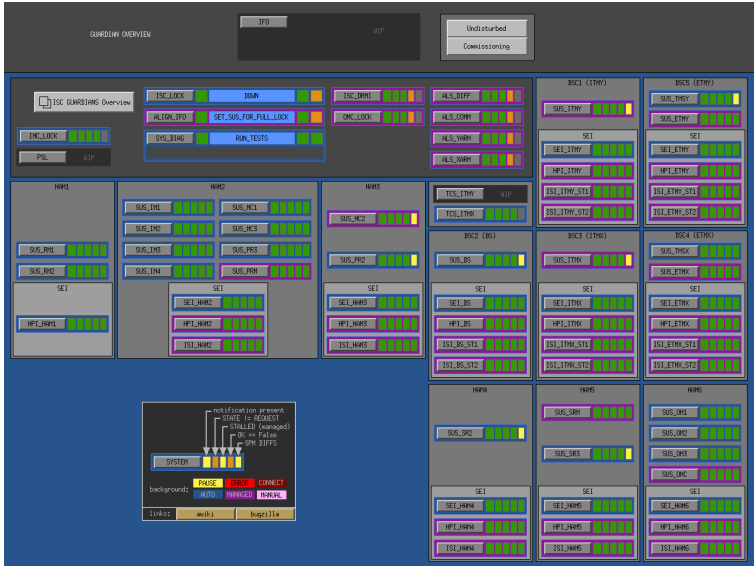
A supervision infrastructure on dedicated hardware at both sites:

- All nodes tightly controlled and monitored
- Verbose, time-stamped logs of all node activity recorded and easily accessible
- Simple interface to create new supervised node instances, start/stop/restart existing nodes
- Node user code archives accessible to all users

```
controls@operator1:~ 0$ guardctrl status
NODE         S VERS EZCA    ARCH_ID OP    E C MODE    OK STATE
----         - ---- ----    ------- ----- - - ----    -- -----
ALIGN_IFO    * 1447  474   78792789 EXEC      MANAGED    SET_SUS_FOR_FULL_LOCK
ALS_COMM     * 1447  474 249288357 EXEC      MANAGED    DOWN
ALS_DIFF     * 1447  474 246471288 EXEC      MANAGED    DOWN
ALS_XARM     * 1447  474 134138931 EXEC      MANAGED    UNLOCKED
ALS_YARM     * 1447  474 176500658 EXEC      MANAGED    UNLOCKED
HPI_BS       * 1447  474   87897949 EXEC      MANAGED  * ROBUST_ISOLATED
```

Interferometer Automation Status

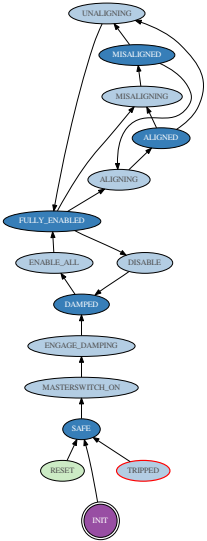
Site overview: over 80 nodes



Subsystems: suspensions (SUS)

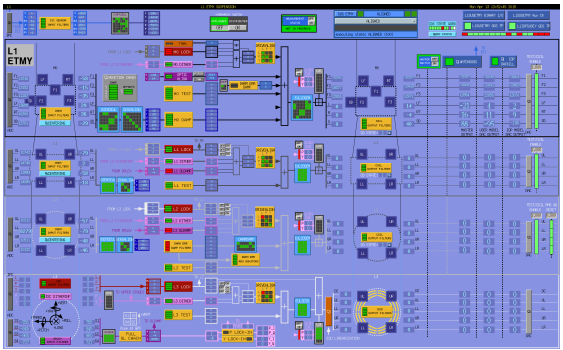
The **suspension** subsystem (SUS) was the first to be automated. All suspensions have similar controller topology, which allows us to use the **same Guardian code for all suspensions**.

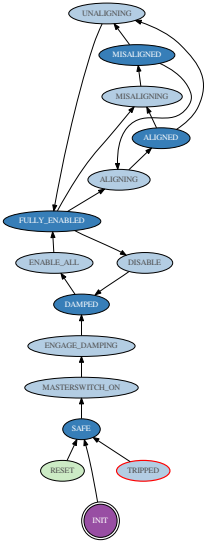
Provides automated recovery to the requested state after operators clear activation of the hardware protection system ("watchdog"), i.e. *one-click* reset.



Subsystems: suspensions (SUS)

The simplified interface provided by Guardian reduces the complexity of the system significantly:



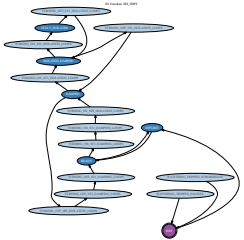


Guardian provides straightforward access to the most common and useful overall states of the system.

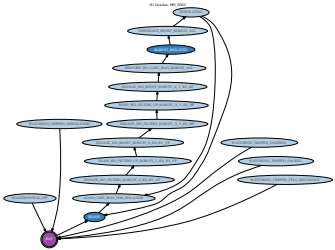
Subsystems: seismic isolation (SEI)

The **seismic isolation** subsystem (SEI) is one of the more complicated subsystems, but also has the most sophisticated user code.

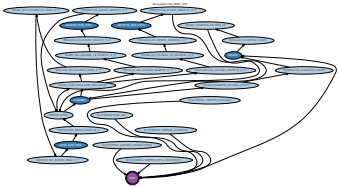
SEI for the "BSC" chambers utilizes four nodes (three for "HAM" chambers), including one *chamber manager*.



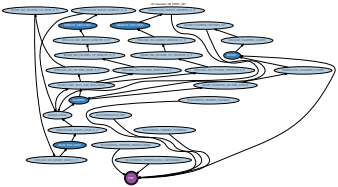
SEI manager



HPI



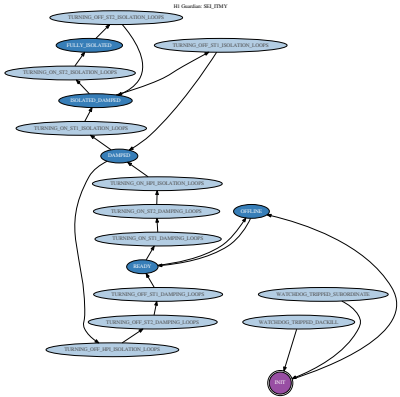
ISI ST1



ISI ST2

Subsystems: seismic isolation (SEI)

The SEI chamber manager simplifies the isolation procedure by coordinating state transitions between the three active stages of the isolation system, as well as by providing a simple interface for the operator/commissioner and the rest of the automation system.

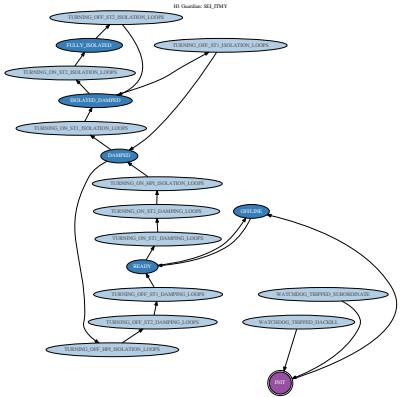


SEI manager

Subsystems: seismic isolation (SEI)

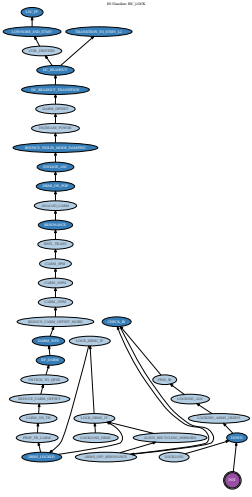
As with SUS, the modularity of Guardian allows us to use a single SEI library to handle all seismic systems, accounting for variations in chamber and configuration.

The SEI Guardians also provide full recovery after watchdog trips.



SEI manager

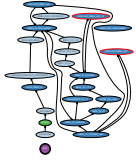
Subsystems: sensing and control (ISC)



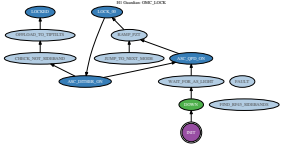
Interferometer sensing and control (ISC) is handled by multiple nodes that independently control:

- the ALS systems (XARM, YARM, COMM, DIFF)
- DRMI degrees of freedom
- OMC lock
- coordinating all subsystems for **full DC readout lock**

H1:ISC_LOCK



H1:ISC_DRMI



H1:OMC_LOCK

Conclusion

Automation of Advanced LIGO with the new Guardian automation platform is working quite well.

Stable, full lock has been achieved at both observatories (H1, L1) with Guardian.

Guardian is also providing the primary readback for the full state of the instruments and all subsystems. This is being actively used for **detector characterization** and **analysis**.

The platform is mostly "feature complete", and we are busy integrating the remaining subsystems into the full automation framework.

Links

- Guardian documentation (LIGO-T1500292)
- Guardian Introduction presentation (LIGO-G1400016)
- Advanced LIGO Automation Requirements (LIGO-T1300884)