LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note **LIGO-T1300884-v3** 2013/10/29

Advanced LIGO Automation Requirements

Jameson Graef Rollins

Distribution of this document:

LIGO Scientific Collaboration

Draft

California Institute of Technology
LIGO Project, MS 100-36
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project, NW22-295
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
PO Box 159
Richland, WA 99352
Phone (509) 372-8106
Fax (509) 372-8137
E-mail: info@ligo.caltech.edu

LIGO Livingston Observatory
19100 LIGO Lane
Livingston, LA 70754
Phone (225) 686-3100
Fax (225) 686-7189
E-mail: info@ligo.caltech.edu

`http://www.ligo.caltech.edu/`

1 Introduction

The purpose of this document is to outline a set of requirements for automation of the Advanced LIGO detectors. Automation refers generally to any system capable of directing the global state of the detectors with as little human intervention as possible. Full automation necessarily involves coordinating the states of the various interferometer subsystems, including the pre-stabilized laser (PSL), seismic isolation (HEPI, SEI, SUS), length and alignment sensing and control (LSC, ASC, ALS, IMC), and thermal compensation (TCS) systems. The automation system should be able to meet the discrete automation needs of all subsystems in order to direct their actions in a coordinated manor to achieve a desired global state.

While a proposal for an aLIGO automation system has already been presented (and to some extent implemented) under the name *aLIGO Guardian* [1], a formal requirements document for automation was never drafted. Consequently, it has never been clear whether or not the *Guardian* system as proposed would ultimately suite the needs of the project. The purpose of this document is therefore to take a step back and examine the project's needs more carefully, with the goal of effectively guiding further development of the automation system, be it *Guardian* or something else.

Note that "automation" in this context does not include any safety protection systems for system hardware or human operators. System safety should be handled separately, e.g. by the hardware "watchdog" systems [2, 3], and is therefore out of scope of this document.

1.1 Operational modes: observation and commissioning

There are two general modes of operation of the detectors that are relevant when formulating automation requirements: *observation* and *commissioning*.

1.1.1 Observation

Scientific output from the LIGO detectors is only possible when they are operating in their most sensitive configuration, referred to as "science mode". To maximize scientific output of the observatories during observing runs, the detectors should be maintained in this state as much as possible.

The primary goal of the automation system should therefore be to get the detectors from any arbitrary uncontrolled state to this "science mode" state (or as least as close to it as possible), as fast as possible, and without human intervention. Given that the detectors are also susceptible to "lock loss", whereby interferometer control, and thereby science mode, is lost due to unforeseen internal or environmental events, the automation system should also be able to identify when lock loss has occurred and act accordingly to bring the detectors back to science mode.

In this observation mode of the system, automation should be single-minded and robust, with the only goal being to maintain the system in one particular, well-defined configuration for as much time as possible.

1.1.2 Commissioning

Single-mode operation of the automation system would, however, ultimately not serve the needs of the project well. Much of the project's time, and all the hard work, goes into commissioning the detectors to achieve science mode operation in the first place. In fact, one of the primary functions of the commissioning process could be said to be the discovery of the observation-mode automation procedure. The automation system should therefore support the commissioning effort as much as possible.

In contrast to observing, commissioning requires that the automation be flexible and modular. Commissioners should be able to commission separate subsystems/parts/domains of the detectors independently, and bring them together as needed. They should be able to reconfigure the automation system as needed to support various commissioning task (system characterization, noise hunting, controls optimization, calibration, etc.).

1.2 Initial LIGO automation

Initial LIGO automation was handled primarily by two separate programs: the Input Mode Cleaner (IMC) autolocker, and *AutoRUN.pl* (also known as the "autorun gui"). The IMC autolocker maintained lock of the IMC, while AutoRUN.pl was used to set the state of the rest of the interferometer.

Both programs relied on UNIX shell scripts that described sequences of actions necessary to transition the system between various operational states. These shell scripts interacted with the real-time digital control system via EPICS binary IO utilities (*caget*, *caput*, *ezcaswitch*, etc.).

The AutoRUN.pl program included a graphical interface from which an operator could select a desired interferometer state ("Acquire", "Detect", "CM", etc.). Once selected, it would run the necessary scripts to transition the system from the currently identified state to the requested state. It was also capable of watching for lock loss, in which case it would kill any running "up" scripts and run "down" scripts that would attempt to recover the initial state so that lock could be reacquired.

This setup worked for initial LIGO, but there were problems and limitations:

minimal state verification The scripts ran "blind", with little to no verification that the appropriate initial conditions for transitions were actually met.

unreliable IO All communication with the digital control system was done through EPICS stand-alone binary utilities. This type of interface is slow and unreliable as connections to the channel access servers are continually being created and destroyed. There was also no verification that EPICS read/write calls had actually succeeded before the transitions continued.

asynchronous The scripts were executed asynchronously, which meant that the state of the system at any point in time was not well defined.

monolithic A single system description was encoded into the AutoRUN.pl. This meant that it was inflexible and could not be easily reconfigured.

unstructured Scripts were not well managed or documented which left the system difficult to maintain.

As much as possible, these issues should be improved or rectified in any automation solution developed for aLIGO: system state should be understood and validated before action is taken, IO should be robust, the system should be modular, well structured, etc. In general, the increased complexity of aLIGO and its subsystems means that a loose bundle of scripts strung together with a single, monolithic program like AutoRUN.pl will not be sufficient to meet the needs of the project.

The iLIGO use of scripts in particular may also turn out to be problematic in the context of robust automation. Initial LIGO relied heavily on scripts, both as commissioning tools and ultimately as part of the emergent automation solution. This is partially because of their ease of use (scripts record transition recipes in a manor that is easy for humans to parse, they are easily executed from the command line, etc.) but also partially because no unified automation system was available. But problems arise when scripts are utilized in an automation environment, as will be discussed later in this document.

1.3 Terminology

There are a couple of terms and concepts surrounding automation that are worth illuminating before proceeding, as they may show up in the discussions below.

1.3.1 process variables: readbacks and parameters

A **process variable** is the general name for any piece of data in a digital control system. They are divided roughly into two types. **Readbacks** are variables that provide a reading of some signal, either internal to the digital system or read from an external sensor. Readbacks are read only. **Settings** are writable variables used for controlling some aspect of the system. Settings are typically used to control parameters of functions that transform data flowing through feedback controllers (gains, additive offsets, active filters, etc.), or specifying the state of an automation unit.

1.3.2 plant vs. controller state

For the case of LIGO automation, it is important to highlight a distinction between the state of the interferometer plant, which is reflected in some set of readback values, and the state of the digital controller, which is described by the values of all controller settings in aggregate. Most automated mechanical systems rely on synchronicity between the state of the plant and that of the controller, and in most circumstances lack of synchronicity would be considered a failure.

For LIGO, though, the inherent instability of the interferometers at the core of the detectors means that there is necessarily a distinction between the state of physical plant and that of the controller. For instance, when locking an optical cavity, parameters of the controller are set to facilitate lock acquisition, but they don't necessarily change once lock is acquired. In other words, the same controller state can be associated with two very different plant states, without there necessarily being a failure.

2 System description

2.1 Existing control and automation capabilities

There are two different real-time control systems currently in use in aLIGO: a “fast”, digital control system, hereafter referred to as the **RTS** (“Real-Time System”), and slower, programmable logic controllers provided by **Beckhoff Automation**. Control parameters for both the RTS and Backhoff systems are exposed via **EPICS** process variables (PVs). The EPICS PVs are accessible through both human-machine and programmatic interfaces. Figure 2.1 is an simplified overview block diagram of the interferometer control structure.

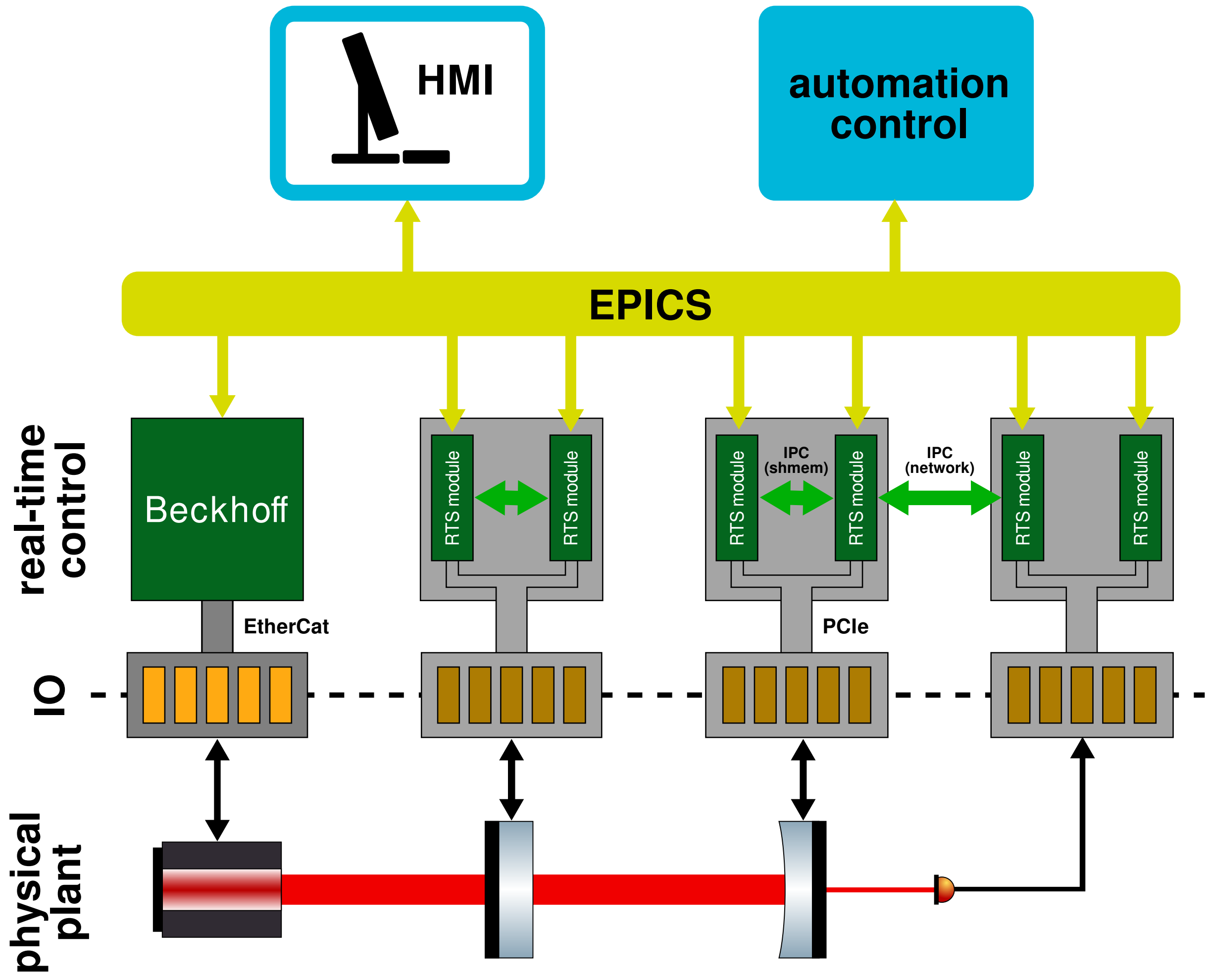

Figure 1: Overview of interferometer control structure. The real-time control system (RTS and Beckhoff) interfaces with the physical plant (i.e. interferometer) via the input/output (IO) layer. Process variables of the real-time control are exposed via EPICS to human-machine interfaces (HMI) and automation control.

The combination of the RTS and Beckhoff covers most of LIGO's controls needs. What is currently missing from the picture is any supervising automation control structure that would fill in the blue box in Figure 2.1. Currently nothing other than the iLIGO-style shell scripts currently being written by commissioners exists here. Filling in this gap is largely the purpose of this document.

Both the RTS and Beckhoff systems are, as generic computing platforms, capable of providing some amount of automation either for themselves or for the system as a whole. The question, though, is whether it is convenient or prudent to implement and program automation logic into them directly, and/or use either as a supervising automation controller, or if something more is needed. In either case, understanding their capabilities is necessary in the broader context and is what will be discussed in this section.

2.1.1 Real-time system

The interferometer is maintained at its linear operating point by the RTS fast digital control system [5]. The RTS consists of a distributed set of real-time controller **modules**. Sensor data is collected via analog-to-digital converters (ADCs) and passed to the RTS modules for processing. Control signals generated by the RTS modules are sent to digital-to-analog converters (DACs) to actuate on the physical plant. The RTS modules can pass signals between each other via inter-process communication (IPC) channels. RTS modules operate with sample rates from 2-64 kHz.

RTS modules are compiled from C code derived from graphical block diagram descriptions of signal flow (drawn in MATLAB Simulink) by a Real-time Code Generator (RCG). The diagrams represent the flow of data through the controller in a single clock cycle, with over-cycle calculations delaying subsequent calculations (soft real-time). The RCG recognizes signal conditioning elements, as well as logical control elements such as switches, matrices, logic gates, etc. Arbitrary functions of input and output arrays can be provided via user-defined C functions.

Many, but not all, of the RTS settings can be controlled internally (see e.g. the cdsFiltCtrl part [6]). It is also possible for RTS modules to control external systems via EPICS, although it is currently inefficient and network-intensive to do so. Both of these limitations could be fixed with some amount of work.

It would be quite cumbersome to program automation logic with Simulink block diagrams, but it could be done with C code instead, although at the expense of accessibility and "user friendliness". In either case compilation is required, which would slow down the development cycle, an important consideration during commissioning when the automation code would likely be modified frequently.

Both of these restrictions mean that it would be difficult to handle all necessary automation tasks within the RTS. All of this said, a lot of work has gone in to moving some useful automation features into the RTS. For instance, cavity locking time has been decreased by triggering feedback based on power buildup in the cavities. Pursuit of this type of internal automation *assistance* should certainly continue to prove valuable.

It is worth noting that a proposal was made to handle all automation via the RTS by extending it to allow communication of slow settings with other modules (and potentially Beckhoff) over dedicated shared memory connections [7]. The reaction to this proposal was not particularly enthusiastic, so it was essentially dropped, but it may be worth revisiting when looking for solutions to the requirements discussed in this document.

2.1.2 Slow controls and Beckhoff

Interferometer “slow controls”, e.g. those that operate at slower sample rates (< 128 Hz), are currently handled by the commercial Beckhoff control system, which consist of EtherCAT “fieldbuses” controlled by Beckhoff TwinCAT [9] programmable logic controllers (PLCs). The PLCs are programmed using the IEC 61131-3 [10] programming standard, and are compiled into real-time code executed on Windows PCs.

Beckhoff includes drivers to support EtherCAT control of many different hardware devices. Three distinct EtherCAT networks are used to control components of the PSL, ISC, and TCS subsystems (see section 2.2), including:

- electronic gain and filter switches in anti-aliasing/imaging, whitening/dewhitening, and feedback analog electronics.
- laser tempurature and frequency controllers
- RF source and VCO controllers
- opto-mechanical controllers (picomoters, shutters)
- ring heater drivers

While there is no requirements document as such for the aLIGO slow controls systems, there is an presentation which describes the use of EtherCAT and Beckhoff TwinCAT in aLIGO slow controls [11].

Beckhoff TwinCAT is marketed as an industrial automation system, and could therefore be considered as a possible solution for supervisory automation. One impediment, though, is the lack of native EPICS support. A custom TwinCAT EPICS IOC has recently been written to allow control of the TwinCAT systems directly via EPICS [12], but there is currently no way for Beckhoff systems to read/write EPICS records hosted on other systems.

The fact that Beckhoff is exclusively Windows based has also made it somewhat technically and culturally difficult to integrate into the overwhelmingly Linux-based LIGO infrastructure. This may improve, though, as we get more used to them and they become increasingly deployed. EtherCAT, as an “open” standard, is not strictly tied to Beckhoff, so could therefore be utilized by other controller solutions.

2.1.3 EPICS

EPICS [13] is the distributed, network-base, message passing infrastructure that provides the primary interface backbone to the interferometer controls. All control parameters (both settings and readbacks) for both the RTS and Beckhoff systems are exposed as EPICS process variables via user-space input-output controllers (IOCs) processes [12]. The IOCs run with refresh rates of 16 Hz, which limits the overall bandwidth of any EPICS-based controls. EPICS also provides an alarm mechanism that can be used to signal clients that a process variables has deviated from a specified range.

One commonly-cited issue with EPICS is the latency and non-determinism of read/write operations introduced by the TCP/IP network transport layer. ***FIXME: what is the expected skew?*** While less than ideal, it was never actually been shown to be a limiting problem in Initial LIGO, and there are not expected to be any situations in the aLIGO lock acquisition process where it should be an issue either (see section 2.3).

2.2 Subsystems

Each interferometer subsystem has different controls needs, and therefore different automation requirements. In all cases, though, the automation needs of the subsystems involve transitioning them between various well-defined steady state operational modes. In this section we look briefly at the various subsystems with the goal of identifying any specific needs that might place additional requirements on the automation system.

2.2.1 Seismic Isolation (SEI)

The automation needs of the HAM and BSC internal seismic isolation (ISI) platforms involve transitioning them between various levels of damping and isolation. The ISIs are characterized by the long timescales of their mechanical response, which means that the transitions between the various isolation states necessarily involves slow actuation and long wait times.

The automation system should also help the ISI systems recover from fault trips of their watchdog systems which protect them from being over-driven. Watchdog faults should always require human intervention to clear, so the automation system should be able to support states requiring some amount of human interaction.

2.2.2 Suspensions (SUS)

The requirements of the SUS subsystems are somewhat relaxed compared to the other subsystems. State changes of the suspensions, including mis-alignments, activation of boost filters, etc., can be handled as discrete switching of control settings, without needing any kind of special sequencing. The SUS subsystem also employs watchdog systems so requirements similar to those for SEI would be apply here as well.

2.2.3 Interferometer Sensing and Control (ISC)

The automation requirements of the ISC subsystem are certainly the most complex. Since ISC requirements are derived mostly from those of the full interferometer lock acquisition, so defer this discussion until the next section (2.3).

2.2.4 Subsystem Beckhoff usage

The Pre-stabilized Laser (PSL) and Thermal Compensation (TCS) systems, as well as some parts of the ISC (such as the Arm Length Stabilization (ALS)), use dedicated Beckhoff deployments to handle their automation needs. These systems therefore don't have special requirements that would need to be met by an additional supervisory automation system. These systems should, however, be integrate-able into a larger supervisory automation structure. The new EPICS TwinCAT IOC interfaces will be very useful in this regard.

2.3 Lock Acquisition

Certainly the single most important and complicated automation task is that of full interferometer lock acquisition. An analysis of the aLIGO lock acquisition problem can be found in [17], and an outline of the acquisition procedure can be found in [19], section 8. The goal of this section, similar to that of the previous section, is to identify specific aspects of the lock acquisition procedure that might dictate specific additional requirements on the automation system.

There are three basic aspects to locking a simple optical cavity:

1. prepare the system for lock acquisition
2. wait for cavity lock to be achieved
3. activate post-lock boost filters, engage alignment control loops, increase power, etc.

The full aLIGO lock acquisition procedure is essentially this procedure writ large, with the additional complications of the interaction of multiple coupled cavities.

FIXME: FILL ME IN

3 Requirements

This section intendeds to distill the needs of the detectors described in the previous sections into a set of requirements that can guide the development of an overall automation system. The operational modes of the interferometer as described in the first section, and the needs of the various subsystems described in the second, naturally lead us to a couple of "axiomatic" requirements for automation system which are considered fundamental and ultimately describe the purpose of automation for LIGO. The automation system must:

1. **Support the concept of operational states of the detectors and subsystems that can be requested by operators and maintained indefinitely.**
2. **Bring the system to the operator-requested state from any arbitrary state of the controller and physical plant, without human intervention.**
3. **Identify when the system is not in the requested state and act accordingly to bring the system back to the requested state as quickly as possible.**

4. **Provide modular support for all detector subsystems allowing them to be commissioned and operated independently and then integrated into an overall supervising control structure.**

The subsections below describe these and other requirements in more detail.

3.1 EPICS support

Given how heavily integrated EPICS is into the LIGO controls structure, particularly in the RTS, the automation system should be able to support communication and control via EPICS. It is currently expected that the limitations of EPICS mentioned in section 2.1.3 will not be an issue in automation control ***FIXME: reference? or described in 2.3?***. That said, he would be prudent to consider replacements for EPICS, possibly with systems that provide synchronous control, for future upgrades.

3.2 States and state identification

The concept of an operational *state* of the system is clearly fundamental; any measurement requires keeping the interferometer in a particular configuration (i.e. state) for some extended period of time. This concept of state, as operational terminus, should therefore be built into the automation structure. Furthermore, it should support multiple states, such that commissioners and operators can request a state from the automation system, which should then be able to take the detector or system to that state and maintain it there indefinitely. This should also include recovering to this state as quickly as possible if the state of the system should change unintentionally (e.g. due to lock loss). Supporting this behavior means that the automation system needs to be able to identify when the interferometer is or is not in a particular state so that it can react accordingly. We refer to this generally as *state identification*.

The question then becomes what is necessary to constitute an identifiable state definition for the purpose of automation control. It is certainly necessary that the automation system have access to at least enough process variables to distinguish the various operational states. However, given that the state of a system is ultimately defined by the value of *every* process variable in the system in aggregate, and if the automation system should be capable of responding to arbitrary state changes, then the automation system should be aware of the value of every process variable in its system.

This is particular relevant when talking about the ultimate science mode observational state. If there are strict requirements on the state of the detectors needed for science mode operation, e.g. that we the state of all process variables in the system correspond to known good values, and if we expect the automation system to respond deviations from science mode operation, then again the automation system needs to track all process variables.

Given the immense number of process variables in aLIGO ($\mathcal{O}(10^5)$) it becomes potentially quite onerous to maintain and manage state definitions consisting of the all process variables for a given system. However, the immensity of the task does not mean the issue can be ignored. It is ultimately necessary to track all process variables to assure legitimate science mode operation. Given that the automation system needs this information as well, this tracking should probably be incorporated directly into the automation system.

The EPICS alarm handling system should be very helpful here, as it provides a means to know that process variables are not within tolerances without having to continually check every variable in a separate process. The automation system should then therefore utilize EPICS alarm handling in it state tracking.

3.3 Modularity

Individual detector subsystems or domains need to be commissioned independently before integrating them into the larger structure. This type of commissioning should therefore be supported by the automation system. Similarly, the same automation control structure developed for a particular component should be easily deployable on other similar components, without requiring distinct copies of the automation code.

3.4 Hierarchy

Given that operators/commissioners should be able to request various different configurations of the full interferometer, it follows that there should be a single supervising automation process that ultimately controls the state of the full interferometer. Coupling this with the modularity requirement above means that the single supervising process should be able to control an arbitrary number of modular sub components. This naturally implies that the system should ultimately be hierarchical in nature, such that all control flows down from the single top level supervisor.

Adherence to strict hierarchy, such that each modular component only accepts commands from a single supervising process, should be used to protect against the possibility of multiple systems competing for control of a single component.

3.5 Manual operation and interruptability

It should be trivial to engage and disengage automation control as needed, to override any automation control and to allow for manual operation of any component of the system. All automation control should be fully interruptable, such that operators can easily kill whatever task or process is currently being executed.

3.6 Partitioning

While operating under full supervisory automation, it should be possible to remove subdomains of the interferometer from supervisory control, to allow manual operation and commissioning, without affecting the automation of the remainder of the system. In other words, the supervisory process should be able to be told that it should effectively ignore the state of a sub-domain or component of the system.

3.7 Determinism

States should be defined exclusively by process variables that are visible and recorded. This should ensure that states are well defined, and that transitions between states always behave the same way. There should be no "hidden" variables who affect the behavior of the system without being recorded.

3.8 Performance

The primary measurable figure of merit for the automation system is the amount of time it takes for the automation system to bring the detectors back to science mode after lock loss. From [14]:

> The Lock Acquisition System (LAS) must reliably bring the interferometer to the end of the Acquisition mode within less than 3 minutes from the beginning of the Acquisition sequence.

and the "availability goals" are given as:

> 90% for single interferometer operations; 85% for double coincidence; 75% for triple coincidence.

Given that automation is not directly involved in maintaining lock, and availability is determined partially by how frequently the system falls out of lock, the automation system should nonetheless help achieve these availability goals by recovering from lock loss as fast as possible.

3.9 Robustness

IO (EPICS) failures should always produce errors, and the system should not continue to operate in the face of IO errors.

3.10 Logging

The automation system should log all of its activity, including all EPICS read/write calls. The logs should be easily accessible by the operators.

3.11 Structure and maintenance

The automation code should be well-structured, maintainable, auditable, etc.

The programming language used to should not be overly obscure to so as to require too much special training. It may be desirable to avoid compiled languages in order to keep the development cycle as short as possible.

The management and administration of the automation control processes should not significantly increase the management burden of the site administrators.

3.12 Extensibility

The system should be extensible such that more sophisticated automation control may be implemented in the future as necessary.

References

[1] S. Waldman, M. Evans, "Advanced LIGO System Guardian", LIGO-T1000131

[2] J. Kissel, "The HAM ISI Watchdog", LIGO-T0900011

[3] R. Bork, R. Abbott, "Suspension Watchdog Requirements", LIGO-T1200306

[4] `http://smi.web.cern.ch/smi/`

[5] R. Bork, "AdvLigo CDS Design Overview", LIGO-T0900612

[6] R. Bork, A. Ivanov, "AdvLigo CDS Realtime Code Generator (RCG) Application Developers Guide", LIGO-T080135

[7] J. Rollins, M. Evans, D. Sigg, "Interferometer Automation with Synchronous Control", LIGO-G1200608

[8] `http://www.beckhoff.com/`

[9] `http://www.beckhoff.com/english.asp?twincat/`

[10] `https://en.wikipedia.org/wiki/IEC_61131-3`

[11] D. Sigg, "EtherCAT for Advanced LIGO", LIGO-G1200005

[12] M. Tse, "TwinCAT EPICS IOC Documentation" LIGO-T1300690

[13] `http://www.aps.anl.gov/epics/`

[14] R. Adhikari, S. Balmer, P. Fritschel, "Interferometer Sensing and Control Design Requirements", LIGO-T070236

[15] M. Evans et. al., "Adv. LIGO Arm Length Stabilization Design", LIGO-T0900144

[16] L. Barsotti, M. Evans, "Modeling of Alignment Sensing and Control for Advanced LIGO", LIGO-T0900511

[17] L. Barsotti, M. Evans, "Lock Acquisition Study for Advanced LIGO", LIGO-T1000294

[18] R. Abbott et. al., "Advanced LIGO Length Sensing and Control Final Design", LIGO-T1000298

[19] P. Fritschel, V. Frolov, D. Sigg, "Advanced LIGO Interferometer Integration", LIGO-T1200437

[20] S. Ballmer et. al., "Online Detector Characterization System Overview", LIGO-T1200323