LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY

LIGO Laboratory / LIGO Scientific Collaboration

LIGO-T1300958 Advanced LIGO 2/17/2013

Front-end Control State Implementation

Stefan, Ballmer, Chris Wipf, Daniel Sigg

Distribution of this document:
LIGO Scientific Collaboration

This is an internal working note
of the LIGO Laboratory.

California Institute of Technology
LIGO Project – MS 18-34
1200 E. California Blvd.
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
185 Albany St
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
P.O. Box 159
Richland WA 99352
Phone 509-372-8106
Fax 509-372-8137

LIGO Livingston Observatory
P.O. Box 940
Livingston, LA 70754
Phone 225-686-3100
Fax 225-686-7189

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

Table of Contents

1 Introduction

The purpose of this document is to define a workflow which reduced the exposed complexity of the LIGO front-end models. It is not a state machine or an active controls system, but tries to simplify the exposed control complexity of a front-end system by defining a number of state control variables which in turn define the state of a large number of EPICS parameters.

Looking at an auto-centering servo we typically have two quadrant detectors with four input filters each, followed by sum, pitch and yaw calculation. These values are then processed by two 2x2 input matrices, 4 servo filter bands, two 2x2 output matrix and actuation filters. This exposes a vast array of EPICS control channels to the user. Most of them have fixed values and never change, some of them may change in a scripted way and a few are adjustable.

From a user point of view most of this complexity is unwanted. The possible control states are only a few. In the simplest case, it just consists of servo on or servo off. All of the filter banks typically have fixed states, so do the input and output matrices. There may be an adjustable threshold to determine the minimum light power on a quadrant detector. This is then used to determine when to operate the auto-centering servo.

The idea proposed here is to define a small set of state control variables using a configuration file. The configuration file will list the available states for each control variable. It will further define whether a value is fixed or adjustable. If it is a fixed value, it will define it. Since we are dealing with a configuration file which can be loaded at run-time, values are not hard coded in the front-end model. Changes are no more difficult than loading a new filter. We will have the same version and configuration control as with “foton” filter files, i.e. the files will get archived whenever they get loaded to the front-end.

Channels which are controlled by a state control variable can no longer be changed by the user directly. Using EPICS, they become effectively read-only channels. This poses a problem for commissioning, since being able to change a state on-the-fly and run a test is an important diagnostics tool. We therefore recommend that each control state has two predefined states: Default and Manual. The default state also serves as a safe state and is the default state after booting. Additional states such as Auto-center and Hold can then be added by the user.

To facilitate large front-end systems which contain multiple subtasks, we also propose to implement sub-state control variables. An ISC system may contain an auto-centering part, and ALS part and an LSC part. Each part can then define one or more sub-state control variables. Meaning, the auto-centering servo can have its own state control variable which is independent from the other parts.

The guardian control software now has a much easier task. No longer does it have to watch a large number of channels just to make sure they are in the right state. Active controls is also simplified. Instead of setting numerous control values synchronously to instigate a state change, only a few control variables have to be set. The guardian control software still has the task to recognize error conditions, deduced lock states, inform the user and put the system into the desired state.

Summarizing, this document proposes to implement front-end control states to significantly reduce the complexity of tracking, saving and referencing the interferometer status. This will provide an effective status system, implemented directly in the front-end systems whenever possible.

2 Front-end operation

- The front-end to EPICS-server channel interface will include an additional check against the configuration file. This check could be implemented on either front-end or EPICS server side. An implementation as part of the EPICS server side has the advantage that the same code could be used for controlling both front-end and Beckhoff applications.
- The desired state is commanded via additional EPICS channels specified in the configuration file (one or more to command the main state(s), additional ones to command every sub-state.
- If the commanded main model state specifies a value for a given EPICS channel, the value from the configuration file will be used in the front-end, and fed back to EPICS as a read-only variable.
- If the commanded main model state specifies a sub-state for a given EPICS channel, the value corresponding to the commanded sub-state will be used, and fed back to EPICS as a read-only variable.
- If the commanded main model state specifies manual operation for a given EPICS channel, the channel value provided by EPICS will be used.
- The configuration file can be read from disk upon request or on boot-up (like the current foton file). On boot-up the initialization values will be loaded.
- A re-initialization can be commanded separately.
- Upon loading the configuration file, a copy of the file is archived.
- The main and sub-state can be commanded through dedicated EPICS variables specidfied in the configuration file. An independent state read-back variable (with the EPICS post-fix "_READBACK") is also provided.
- [An error bit indicating manual operation?]

3 Control file format

General remarks:

- The file lists every EPICS channel used by the front-end model, with filter-module-related channels and matrix-related channels grouped together.
- The IFO name will be omitted in the channel list, allowing copying files between interferometers.
- For editing a dedicated GUI (like foton) will be written.
- The file will include tables to define the main state(s) and sub-states.
- In a main state table, a initialization values have to be specified. Channels not listed in any initialization list will default to zero, and will be read-only in any state other than 0 (off).
- State 0 (off) and state 1 (default) always exist. An arbitrary number of additional states can be defined.

- In the main state table, for each state and for each EPICS channel one of the following has to be specified: 1) a value, or 2) manual operation through EPICS, or 3) a pointer to the relevant sub-state table.
- An arbitrary number of sub-state tables can be provided. Unlike the main state tables, they do not have to list all channels in an initialization list.
- In the sub-state table, for each sub-state and for each involved EPICS channel one of the following has to be specified: 1) a value, or 2) manual operation through EPICS.
- For binary channels “manual” mode can be commanded for every bit individually. This is done via a manual mask.
- In addition to the default state and the user-defined states, a off or manual state is implicitly defined. It's state number is zero (0). For it, all channels default to “manual”.

Example:

LSC-MASTERSTATE

Channel name	INIT	0: OFF	1: DEFAULT	2: RUN
LSC-DARM_GAIN	1	manual	2	3
LSC-DARM_SW1S	0xFF	manual	0x33 (Mask: 0xF3)	0x33 (Mask: 0xF3)
LSC-CARM_GAIN	0	manual	manual	manual
LSC-MICH_GAIN	0	manual	See INIT (0)	Sub state LSC-GAINSTEPPING

LSC-GAINSTEPPING

Channel name	0: OFF	1: DEFAULT	2: STEP A	3: STEP B
LSC-MICH_GAIN	manual	See default above	1	2

3.1 Example

We envision using an xml file format. The example tables above would look as below:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--This is a configuration file of the LIGO project!-->
<ControlStateDef xmlns:p="https://dcc.ligo.org/LIGO-T1300958/public" Target="lsc">
        <Assign Name="LSC-REFL_A_RF45_I_GAIN">1.2</Assign>
        <Table Name="LSC-MASTERSTATE" Type="main" Location="internal">
                <Assign Name="LSC-DARM_GAIN">1</Assign>
                <Assign Name="LSC-DARM_SW1S">0xF3</Assign>
                <Assign Name="LSC-CARM_GAIN">0</Assign>
                <Assign Name="LSC-MICH_GAIN">0</Assign>

                <State Number="0" Name="OFF" Ramp="0"/>

                <State Number="1" Name="DEFAULT">
                        <Assign Name="LSC-DARM_GAIN" Type="val" >2</Assign>
                        <Assign Name="LSC-DARM_SW1S" Type="val" Mask="0xF3">
                                0x33
                        </Assign>
                        <Assign Name="LSC-CARM_GAIN" Type="man"/>
                </State>

                <State Number="2" Name="RUN">
                        <Assign Name="LSC-DARM_GAIN" Type="val" RAMP="3.0">3</Assign>
                        <Assign Name="LSC-DARM_SW1S" Type="val"
                                Mask="0xF3">0x33</Assign>
                        <Assign Name="LSC-CARM_GAIN" Type="man"/>
                        <Assign Name="LSC-MICH_GAIN" Type="sub">
                                LSC-GAINSTEPPING
                        </Assign>
                </State>
        </Table>
        <Table Name="LSC-GAINSTEPPING" Type="sub" Location="internal">
                <State Number="0" Name="OFF"/>
                <State Number="1" Name="DEFAULT"/>
                <State Number="2" Name="STEP A" RAMP="1.0">
                        <Assign Name="LSC-MICH_GAIN" Type="val">1</Assign>
                </State>
                <State Number="3" Name="STEP B" RAMP="1.0">
                        <Assign Name="LSC-MICH_GAIN" Type="val">2</Assign>
                </State>
        </Table>
</ControlStateDef>
```

Specifically, the xml file has the structure outlined below.

3.2 Header

The XML header is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--This is a configuration file of the LIGO project!-->
```

3.3 ControlStateDef

This tag describes the top level container.

Attribute Target: Contains the real-time model name for which the file specifies states, without the interferometer name. Examples: lsc, asc, susbs

Attribute xmlns:p: Specifies XML namespace. The name space points to the current version of this document. It should be set to https://dcc.ligo.org/LIGO-T1300958/public.

3.3.1 Table

This tag describes a controlled state.

Attribute Name: Specifies the name of the EPICS channel controlling the states defined in this table.

Attribute Type: The choices are “main” or “sub”. The default is “main”, if omitted. Defines whether the table specifies a main state or a sub state

Attribute Location: The choices are “internal” or “external”. For “internal” the controlling EPICS channel is created in the EPICS server, and it is not accessible by the front-end itself. For “external” the channel has to be defined in the front-end, and thus is accessible by the front-end.

For tables of type *“main”*:

> Every table of type “main” has at least a State 0 (off/manual) and a State 1 (default state). If not listed explicitly, the settings in state 1 correspond to the initialization values.
>
> Multiple tables of type “main” are permissible, but their channel lists as defined in the initialization section must be disjoint.
>
> If a channel is neither listed in the initialization section of a table of type “main” or at the top level, its value is set to 0 and not changeable in any state.

For tables of type “sub”:

> The State 1 of a table of type “sub” is implicitly defined as being equal to the State 1 of the main table pointing to it. However, this implicit definition can be overwritten by explicitly defining a state 1.

Attribute Ramp: Defines the default ramp time in seconds for switching values within the table. If omitted, values are changed immediately. Ramp times are not used during initialization.

3.3.1.1 Initialization

An Assign tag located inside the Table but outside a State tag is treated as an initialization and default value. This is only allowed for a table of type “main”. It can only contain Assign tags of

type “val” or ‘man” (see below). If it is defined as “val”, its value is used both for initialization and as the default value when not specified within a state. When it is defined as “man”, its value is used for initialization but it is left in manual mode when not specified in a state.

The initialization list must define all channels associated with a “main” table.

Every channel name of the target that is neither listed in the initialization list of a “main” table or at the top level, is set to and kept at zero (0).

If a channel is listed, but no value is specified, it will be initialized as zero (0), but will be available in other states.

3.3.1.2 State

This tag describes an individual control state.

Attribute Number: This specifies the number of a state. This corresponds to the selection value of the controlling EPICS channel (specified in the attribute Name). The number zero (0) is reserved for the off/manual state. The number one (1) is reserved for the default state. Negative numbers are not allowed. If all the state numbers are equal or below 15, an EPICS mbbo record will be used. If at least one state value is larger than 15, an EPICS longout record will be used.

Attribute Name: This specifies the name of the state. This will be used to define the EPICS value type for mbbo records. For EPICS longout records the name is ignored.

Channels can be reassigned inside a state. Omitting a channel name implies inheriting the settings from the initialization section.

The State 1 (default state) of a table of type “sub” is implicitly defined as being equal to the State 1 (default state) of the main table pointing to it. However this implicit definition can be overwritten by explicitly defining a state 1.

Channels in State 1 (default state) can have manual entries and Mask fields. However this should be used sparsely, and only where needed. An example might be alignment sliders, whose good settings can change from day to day.

Attribute Ramp: Defines the default ramp time in seconds for switching values within the state. If omitted, values are changed immediately. Ramp times cannot be specified for initialization. Ramp times specified by a state override the one specified by a table.

3.3.2 Assign

The Assign tag is used to assign values to channel names. When defined under ControlStateDef, it denotes a global constant (using type “val”) or a manual value (using type “man”). A manual value is used to initialize the channel, but is then left in manual mode.

When defined inside a table, it denotes an initial and default value for a channel. When defined under a state, it defines the value associated with a state. A channel with its optional bit mask can be defined exactly once under a table. It can be redefined inside multiple state tags of the same table, or in an associated sub state table.

When defined at the top level, it cannot be redefined inside a table. When defined inside a table, it cannot be redefined in another table.

Assign tags with the same channel name but a different bit mask are treated as different entities. If a bit mask is used, it has to match when a channel entity is reassigned inside the different states of a table.

Attribute Name: This specifies the name of a channel, but without the interferometer prefix, e.g., LSC_DARM_GAIN or LSC_DARM_SW2S.

Attribute Type: The choices are either “val”, “man” or “sub”. The default, if omitted, is “val”. Specifies the type of assignment made to the specified channel in the specified state. The keyword “sub”, implies that a sub state Table is defined which in turn will redefine the value.

Attribute Mask: This specifies a binary mask (up to 32 bits) for the channel. If a mask is used, it has to be repeated in all associated assignments. A channel can be split into multiple entities which have non-overlapping bit masks. Multiple entities of the same channel are for all practical purposes treated like separate channels. Bits of a channel which are defined nowhere, are set and kept at zero. Omitting the Mask attribute implies that no mask is applied, i.e. all bits of the data field are used. With this definition a mask of 0 (no mask) and 0xFFFFFFFF (all bits used) are identical in purpose. A mask can be defined as a decimal, octal or hexadecimal number.

The data field of an Assign tag contains the following:

For Type=“val”, any of the following:

- Floating point number (e.g. 58.1, or 58E0)
- Decimal number (e.g. 58)
- Octal number (e.g. 072)
- Hexadecimal number (e.g 0x3A)
- Boolean value: true or false
- A sting in quotes, indicating an EPICS field enum value (e.g. “inactive” or “off”), or a string value.

For type “man”, the data field is either used as an initialization value or ignored.

For type “sub”, the data field specifies the sub-table channel name which is used to define this channel (no quotes). This is only permitted in a table of type “main”. It is not permitted in state 1 (default state).

Omitting the data field for type “val” is equivalent to specifying 0 in the data field.

Attribute Ramp: Defines the ramp time in seconds for changing to the assigned value. Ramp times only makes sense for value assignments. The value is ramped linearly from the previous one to the newly specified one, when the state is changed. Tables can specify a default ramp time which is used to switch all values used in the table. If a state specified a different ramp time, it gets precedence over the table ramp time. Finally, a ramp time specified as part of an assignment takes precedence over all others.

3.5 Redefinition Rules

The following rules apply for parsing one or more control state definition files:

- Multiple ControlStateDef tags can be defined within a single file, or in multiple files.
- If multiple ControlStateDef tags are defined within a file, they should have a different target. If the same target name is used, the tables, states and assignments of the later definition will be added to the first.
- Generally, it is expected that each table definition has a unique name. If a later table has the same name, its states and assignments are added to the first table. It is treated as a simple reiteration.
- If a table is redefined within the same ControlStateDef tag, it should have the same type and location. If not, the later definition overwrites the previous one.
- If a table is redefined across multiple ControlStateDef tags, only one can have an internal location. They can all have an external location.
- If a table is redefined with a different type, the later definitions are ignored.
- Two state definitions are considered identical, if they are associated with the same table and if they have the same number. If a state is redefined with a different name, the later name is ignored.
- An assignment defines a tag. Two tags are considered identical, if there name and mask are the same. Tags are unique, if they either have different names, or if they have non-overlapping bit masks and identical names.
- Tags which are neither identical nor unique have the same name and overlapping bit masks. They are a mistake and result in undefined behavior.
- Tags are generally reiterated in different states of the same table. This results in different values depending on the state of system. A tag can also be reiterated in a different table, if the first table is of type “main” and the others are of type “sub”.
- If a tag is redefined within the same table and state, the later definition overwrites the previous one.
- A tag which is defined in a table of type “sub”, but is never defined in a table of type “main” is lost and will be ignored.
- However, it is a mistake to redefine a tag across different tables of type “main”. Or, to redefine a tag in a table when it has already be defined as a global constant. When conflicting tag definitions are present, the later definitions will be ignored.

Generally, redefining a table or a state should be avoided. In a few cases, it could make sense to have a global control state definition file and keep site specific variations contained in separate files. Within the same file, it almost never makes sense to redefine a table or a state. Overwriting a previous tag has similar implications. Good practice is to keep the definitions of tables, states and tags within the same group to avoid confusion.

4 GUI for editing the configuration file

- The GUI provides an easily clickable graphical interface for setting the filter modules. (Using “ezcaswitch” logic - not “ezcawrite” logic)
- The GUI provides a formatted matrix interface of system matrices
- The GUI provides a bitmap interface of configuring binary control channel.
- For all other channels the GUI allows entering the value.
- “manual” or “substate” mode has to be command-able on a “per bit”, “per FMx block”, “per ON/OFF switch” or “per value” basis.
- In addition to basic edit functions, the GUI allows copying existing states.

5 Other remarks

- On compilation the front-end code provides the list of channels and updates the existing configuration file.
- On compilation new, previously non-existing channels will be we set to a value 0 (off or disabled). In particular, active user action in the GUI will be need to define a channel as “manual”.
- Upon restart the front-end code will load the initialization values and default to the “default” state.
- Same infrastructure for Front-end and Beckhoff: Beckhoff-EPICS interface will be upgraded to be able to parse the same format of state files for Beckhoff's EPICS channels.