femto.py - High Level Application

femto.py version 2.0

Justin May
AD-Diagnostics and Appliations Development, LCLS Timing

Date: 2021-08-17

Under Development

Introduction

The original version of femto.py was written in 2012 for the commissioning of the LCLS-I injector reconfiguration and commissioning of the MEC hutch. It was effectively a hybrid of iddle-ware software and what would later be formalized as a high level application (HLA). The code base grew beyond its initial scope, and had some initial structures that weren't easily scalable to the eventual large number of installed systems.

In 2021, motivated by our experience with photon operations and the need for a new femto.py for LCLS-II and related laser systems, we refactored and expanded the femto.py HLA. This documentation applies to this new version of femto.py and associated systems.

Theory of Operation

The key functions of the HLA fall under the categories of: hardware monitoring and management, rf bucket detection and correction, and phase control. The various topologies and numerologies of the lasers in use at the laboratory mean that logic is not necessarily consistent, and so in some cases multiple inheritance is used. Currently, the instances of this are limited to:

1. Generation 1- and Generation 2-Specific Locker Cores
2. Combined Function Time Interval and Frequency Counter Support (currently only applies to Facet-II)

Modifications to Calibration Function

The original HLA performed a brute force calibration wherein a fixed sample sweep of the range of the oscillator phase shifter was later fit to a sawtooth function, the parameters of said function being the output of the calibration routine. This was never made very clear to the user, particularly regarding the output of the fit function (which considered in arbitrary units was used to indicate the health of the system). It was also a fairly slow system, dependent on the move-confirm speed of the phase shifter in the locker. Some attempt was made early on to improve on the calibration, and provide some clearer feedback to the end user, but this was never finished.

The new generation of the code uses a more structured optimizer. However, due to current limitations in the RF firmware of the locker, calibrations are susceptible to corruption from phase hysteresis in the lockers. The

most stable operation is currently achieved with a manual calibration of the laser.

Modifications to Configuration Control

The original HLA relied on a command line argument addressing hard-coded definitions within the codebase to set PVs and configurations for the given laser it was being run on. This was minimally extensible while trying to maintain a consistent code base across the site. Moreover, when we began to expand the scope of the precision timing systems to other facilities and operational environments, this and a lack of centralized version control made it very difficult to manage and reconcile the various installations, often inheriting conflicts in the configurations (for example, between the FACET sector 10 and sector 20 systems).

To remedy this and make a clearer delineation between operational logic and the data model for an installation, the HLA has been refactored to use configuration files that are stored separately. These can be authored by appropriate personnel for an installation. We've attempted to make them somewhat extensible, but since we are supporting both a legacy code base and newly developed timing systems, there are some parameters that are still somewhat restricted in their format and possible values. Still, removing the definitions for a machine from the main code increases supportability and minimizes the chances for errant code commits to break operational installations.

During development of the new HLA, and specifically the creation of configuration files for all of the installed systems at the lab, unit testing was used to insure that PVs loaded in the new HLA matches against those from the original installations. In this way, we have high confidence that the new code, when run on existing installations, has a high likelihood of drop-in operation. However, this should be tested prior to program delivery.

Source Code Documentation

Where appropriate updated or additional comments have been added to the legacy code to make it more consistently documented. In addition, versions of condensed help files are available from the HLA control panels. This documentation is managed under version control, alongside the source documents from which Sharepoint and Confluence documentation is produced.

{Auto-generated code documentation not yet included}

Technical and Lifecycle Details

This HLA is managed through git version control on the SLAC github repository fs-timing. Revisions, feature requests and bugs should be managed ideally through a combination of git features and SLAC mgmt systems like CATER.

The HLA is written against Python 3.6, but has been tested up to 3.8.12. It does not currently have any package manager configuration, though classes use conditional imports to support multiple installation environments. LCLS-photon installations rely on PCDS Anaconda environments, while accelerator side installations use the configured Python3 environments. The HLA and the associated IOCs that support the precision timing system are not currently deployed against Epics v7, though this should be expected to be completed in 2022.