

Extending the Control Software for Beam Interlock System 2

CERN Summer Student Programme 2022 Final Report

Aleksander Buszydlik
Technische Universiteit Delft
a.j.buszydlik@student.tudelft.nl

Abstract

The Beam Interlock System (BIS) is a network of devices aiming to protect the machinery of the Large Hadron Collider (LHC) complex. The BIS devices are used to calculate the value of the Global Beam Permit, a control signal that, if false, triggers the LHC Beam Dumping System. A new version of the system – BIS2 – with increased flexibility, diagnostic capabilities, and safety guarantees is being developed. It will be running in parallel to BIS and, as it requires a similar Control System, its introduction is an opportunity to renovate the software in place. The new solution adapts the Unified Controls Acquisition and Processing (UCAP) framework which facilitates the integration between different controls subsystems of the accelerators. In July and August of 2022, the new control software has been extended in two ways. First, the existing code was reorganized to improve the reusability of the generators for virtual device definitions. Second, generators for five new functionalities have been implemented in the system.

1 Introduction

The BIS constitutes a crucial part of the machine protection systems for the LHC. Devices of the BIS monitor the accelerator subsystems and allow for the operation only if a list of conditions is met. Otherwise, the Beam Dumping System is triggered to protect the LHC from the beam-induced damage. Subsystems controlled by the BIS devices include, for example, the level of vacuum or the state of the magnets. Values from these subsystems undergo a logical “AND” to calculate the value of the so-called Local Beam Permits which are then used to evaluate the accelerator-wide Global Beam Permit [4]. Additionally, values may be masked (overridden) or completely disabled by the operator which influence the results of the Local and Global Beam Permits [4]. Thus, the controls software for BIS must expose such additional information about the device subsystems to its users.

The new version of BIS is underway, with plans to put it into operation for Run 4 of the LHC [3]. The Controls and Beam Studies for Protection section of the Machine Protec-

tion and Electrical Integrity Group¹ is developing the control software for BIS2. It makes use of the UCAP framework to handle the Acquisition – Transformation – Publishing tasks using generic data converters [1] as presented in Figure 1.

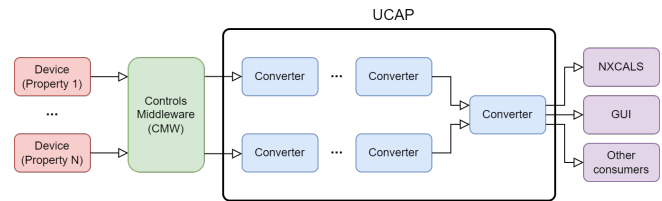


Figure 1: UCAP converters acquire data from CMW devices or other converters, transform it, and republish it to, e.g., NXCALs or BIS.

UCAP was selected for the new system for multiple reasons:

- it is being developed internally at CERN with a goal to be used in various controls systems of the organization;
- it abstracts away many IT operation tasks, allowing to focus on the development of business logic;
- it allows to publish intermediate results of the transformations along with the final outputs;
- the virtual UCAP devices expose the same interface as the real BIS devices registered in CMW;
- the UCAP converters can be reused in different scenarios.

The framework allows for the generation of device definition files which describe virtual UCAP devices – not associated with hardware, as opposed to the actual BIS devices – including a list of available transformations. Such definitions can be loaded into isolated nodes where consumers may subscribe to the data using a Device/Property model [1].

When the interlock devices publish data from the CIBM/CIBX manager boards via the Controls Middleware (CMW) [2], this data is transformed and republished by the virtual UCAP devices. After it has been published, the data from real and virtual devices may be written to NXCALs (the CERN accelerator logging service) [5], consumed in the BIS Graphical User Interface, or used by other clients.

¹<https://mpe-cb.web.cern.ch>

This report summarizes my contributions to the controls software for BIS2. It adheres to the following structure. First, Section 2 describes the efforts taken to reorganize the codebase of the project. Then, Section 3 informs about the new functionalities added to the BIS2 systems. Section 4 outlines the planned next steps in the development of BIS2 software. Finally, Section 5 forms the conclusion to this report.

2 Reorganization of the codebase

Initially, the codebase of BIS2 was organized in a way that similar concerns of the system were made accessible via different virtual devices. In particular, analogous functionalities from different data sources were not made accessible by the same device which resulted in unnecessary overhead as some functionalities always require a specific set of transformations. For instance, we may want to:

1. extract information from the Board Registers;
2. extract corresponding information from the History Buffer which records data over time in a rolling buffer;
3. combine the two sources to produce a single output.

Thus, the architecture of the system was redesigned to organize the existing functionalities and facilitate code reuse. The new setup relies on a `VirtualDeviceDefinitionManager` object which allows the user to register a list of transformations that will be written to the definition file of a virtual device whenever its *generator* is executed.

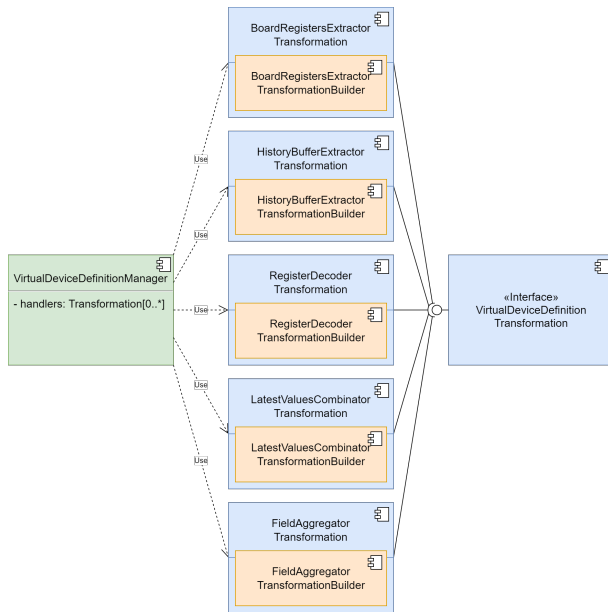


Figure 2: All transformations implement a common interface.

Five transformations are implemented (shown in Figure 2):

- **BoardRegistersExtractorTransformation**, used to extract a specified list of fields from one of the channels (registers) of the Board Registers.
- **HistoryBufferExtractorTransformation**, used to extract events of a certain type from the History Buffer.

- **RegisterDecoderTransformation**, used to decode a string of bits received from an extractor into boolean or numerical values.
- **LatestValuesCombinatorTransformation**, used to combine the values from two different subscriptions by republishing values from one of them whenever available, and values from the other one only when necessary.
- **FieldAggregatorTransformation**, used to republish multiple fields under as single property.

The first three transformations are wrappers around the previously existing converters; the last two make use of new converters implemented as part of my project. All transformations follow the same interface, thus, they can be combined in any number and order to generate definitions for new virtual devices. To that end, it is required to specify a set of required properties following a *builder* design pattern.

3 New virtual device definition generators

The existing codebase was also extended with virtual device definition generators for five functionalities which are outlined in this section. First, the generator for masked inputs is discussed. Then, a similar generator for the information on disabled inputs is presented. Finally, the characteristics of the three generators for “expert information” are explained.

3.1 Masked inputs

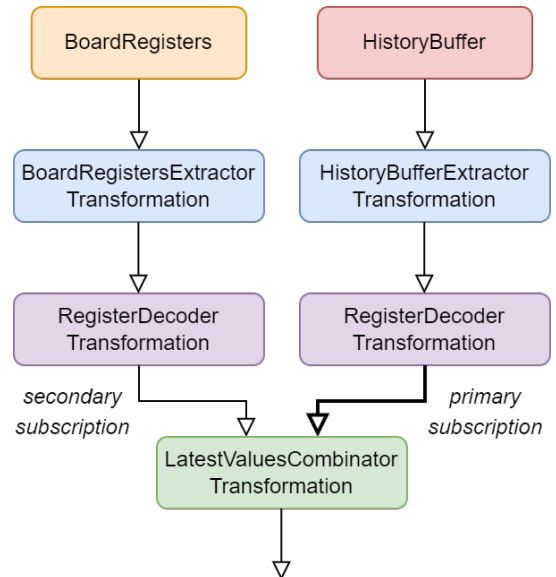


Figure 3: LatestValuesCombinatorTransformation allows to select between two independent data sources.

An operator of the accelerator may decide to mask (override) some of the inputs, for instance, in order to isolate the impact of specific parameters on the system. Information which inputs have been masked is stored in two sources: the Board Registers and the History Buffer. The former is more robust as it is always available, the latter is of higher resolution

as it is more frequently updated. Nonetheless, the History Buffer is only updated when a change occurs in the system, thus, when no changes have occurred it is necessary to read the data from the Board Registers. A virtual device for the information about the masks set in the system should therefore obtain the values from both sources and select which of those should be republished. As presented in Figure 3, we use the `BoardRegistersExtractorTransformation` to obtain the values from the Board Registers and decode them using the `RegisterDecoderTransformation` according to the CIBM technical specification. We use the `HistoryBufferExtractorTransformation` to get the corresponding values from the History Buffer and decode them analogously. Finally, we apply the `LatestValuesCombinatorTransformation`, treating the History Buffer as the primary source that is republished whenever data is available, and the Board Registers as the secondary source that should be republished only when needed.

3.2 Disabled inputs

Some inputs may be entirely disabled in which case their values have no influence on the calculations of the Global Beam Permit. Any of the fourteen inputs for the two redundant matrices A and B may be disabled. The generator for this functionality follows the same logic as shown in Figure 3 where the History Buffer inputs are treated as primary data due to their higher resolution. This also indicates that some generators may share the set of transformations, thus, the common logic has been abstracted away in the source code.

3.3 Expert information

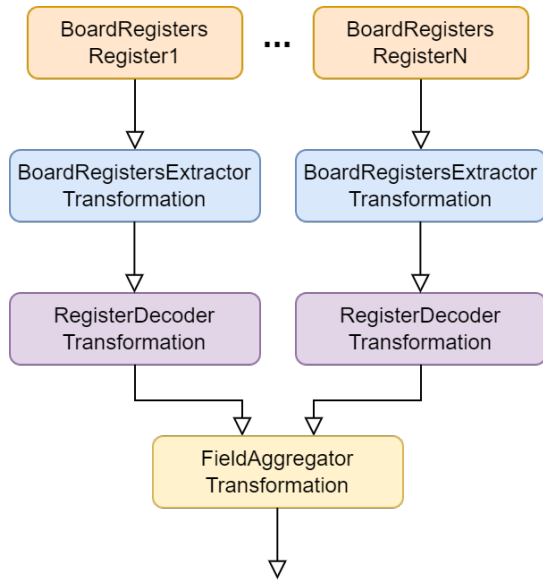


Figure 4: `FieldAggregatorTransformation` can be configured to publish fields from various sources of data under a single property.

This component combines three functionalities relating to the overall operation of the accelerators:

- **STATUS** which is a boolean register informing about the status of an accelerator as measured on a device.
- **FREQ** which is a set of four 32-bit registers describing the beam frequencies for loops A and B on the input and output of a device.
- **BPL.MONITORS** which is a set of 22 32-bit registers describing the spikes of frequency measured on a device.

In the definition generators for these three functionalities we apply a `BoardRegistersExtractorTransformation` followed by a `RegisterDecoderTransformation` on each of the individual registers. Using the latter transformation further allows to rename the fields, making them more understandable for the user. Additionally, for the **FREQ** and **BPL.MONITORS** functionalities we apply a `FieldAggregatorTransformation` (Figure 4) which ensures that instead of multiple properties publishing a single field that would have to be queried separately, a single property aggregating the values of all fields is published.

4 Further work

In the future, the development of BIS2 software should focus on two areas. First, there still exists a number of functionalities present in the current BIS that are missing from its new version. For instance, this includes the overview of matrix equations which are used to calculate the beam permits and decide whether the accelerator is allowed to operate. It can be expected that the inclusion of the missing functionalities will require new converters and their corresponding transformations – current codebase may be used as guidance to implement these. Second, in order to successfully substitute the system-in-place, BIS2 requires a graphical user interface that will provide the operators with an overview of the CIBM/CIBX data. Such GUI is already in development.

5 Conclusion

The current BIS which is part of the machine protection systems in the LHC complex will receive an upgrade to a new version by 2027. BIS2 is in active development and its control software will rely on the UCAP framework for streamlined processing of data from the interlock devices. UCAP provides an interface for creating reusable converters that tackle Acquisition – Transformation – Publishing tasks. These converters are declared in the definitions of virtual devices which, when loaded into a UCAP node, automatically process and republish the data. As part of my Summer Student programme at CERN I worked on improving the existing system in two major ways. First, I refactored the existing codebase to simplify the creation of these definitions, using parameterized Transformations built on top of UCAP converters. These transformations are fully parameterized to support various use cases and can be easily combined to create new virtual devices. Second, I introduced virtual device definition generators for five functionalities: one for combining the information related to masked inputs, one for combining the information related to disabled inputs, and three for aggregating the general information about the operation of the accelerators. These generators required me to develop two new

UCAP converters – the `LatestValuesCombinator` for combining the values published by different subscriptions and the `FieldAggregator` for aggregating the fields published under different properties. Directions of future work include the development of generators for missing functionalities, and the development of a graphical user interface for the BIS2.

Acknowledgements

I would like to extend my gratitude to my supervisor Jonas Barth and the developer of the BIS2 controls software Tobias Skarhed for their engagement with my project and guidance throughout the process. I also want to thank the entire TE-MPE-CB section for the positive atmosphere in the team where I was warmly received. Finally, I am grateful for all of the amazing Summer Student friends I made at CERN.

References

- [1] Lajos Cseppentő and Mark Büttner. “UCAP: A Framework for Accelerator Controls Data Processing @ CERN”. In: *JACoW ICALEPCS2021* (2022), 230–235. 6 p. DOI: 10.18429/JACoW-ICALEPCS2021-MOPV039. URL: <https://cds.cern.ch/record/2809575>.
- [2] Andrzej Dworak et al. “The new CERN Controls Middleware”. In: *Journal of Physics Conference Series* 396 (Dec. 2012), pp. 2017–. DOI: 10.1088/1742-6596/396/1/012017.
- [3] Roland Johnson et al. “The Consolidation of the CERN Beam Interlock System”. In: *JACoW IPAC 21* (2021), 3309–3312. 4 p. DOI: 10.18429/JACoW-IPAC2021-WEPAB282. URL: <http://cds.cern.ch/record/2810349>.
- [4] B Puccio et al. “The CERN Beam Interlock System: Principle and Operational Experience”. In: (2010), 4 p. URL: <https://cds.cern.ch/record/1277643>.
- [5] Jakub Wozniak and Chris Roderick. “NXCALS - Architecture and Challenges of the Next CERN Accelerator Logging Service”. In: (2020), WEPHA163. 5 p. DOI: 10.18429/JACoW-ICALEPCS2019-WEPHA163. URL: <https://cds.cern.ch/record/2778529>.