Framework for Testing and Operation of the ATLAS Level-1 MUCTPI and CTP

R. Spiwoks [a], D. Berge [a], N. Ellis [a], P. Farthouat [a], S. Haas [a],
J. Lundberg [a], S. Maettig [a, b], A. Messina [a], T. Pauly [a], D. Sherman [a]

[a] CERN, 1211 Geneva 23, Switzerland
[b] University of Hamburg, 20146 Hamburg, Germany

Ralf.Spiwoks@cern.ch

Abstract

The ATLAS Level-1 Muon-to-Central-Trigger-Processor Interface (MUCTPI) receives information on muon candidates from the muon trigger sectors and sends multiplicity values to the Central Trigger Processor (CTP). The CTP receives the multiplicity values from the MUCTPI and combines them with information from the calorimeter trigger and other triggers of the experiment and makes the final Level-1 decision. The MUCTPI and CTP are housed in two 9U VME64x crates and are made of nine different types of custom designed modules. This paper will present the framework which is used for debugging, commissioning and operation of all modules of the MUCTPI and CTP.

Testing of the modules has been considered right from design. Most types of modules contain diagnostic memories at the input of the module which can be used to capture incoming data or to inject data into the module. Testing of the modules can be achieved by capturing data at input of a down-stream module, by reading out data from a monitoring buffer, or by reading out monitoring counters.

A layered software framework using C++ has been developed for configuring and controlling all modules and for testing them independently or grouped into complete sub-systems. The lowest level uses the ATLAS VME library and driver. At the next higher level, a compiler translates a description of the VME registers from XML to C++ code. This code together with existing code for some components, e.g. HPTDC, DELAY25, and JTAG, is combined to the low-level library of the module. A menu program provides access to all methods of the module low-level library. Generators create data for the test memories. Simulators calculate expected results. Generators, simulators and the low-level library are combined to a suite of test programs which cover the full functionality of the MUCTPI and CTP. The low-level library is also used by the control and monitoring programs which integrate the sub-systems into the ATLAS experiment control and monitoring framework.

I. Introduction

The ATLAS experiment at the Large Hadron Collider (LHC) at CERN uses a three-level trigger system. The Level-1 trigger [1] is a synchronous system operating at the bunch crossing (BC) frequency of 40.08 MHz of the LHC. It uses information on clusters and global energy in the calorimeters and on tracks found in the dedicated muon trigger detectors. An overview of the ATLAS Level-1 trigger is shown in Figure 1. The Level-1 central trigger consists of the Muon-to-Central-Trigger-Processor Interface (MUCTPI), the Central Trigger Processor (CTP), and the Timing, Trigger and Control (TTC) partitions.

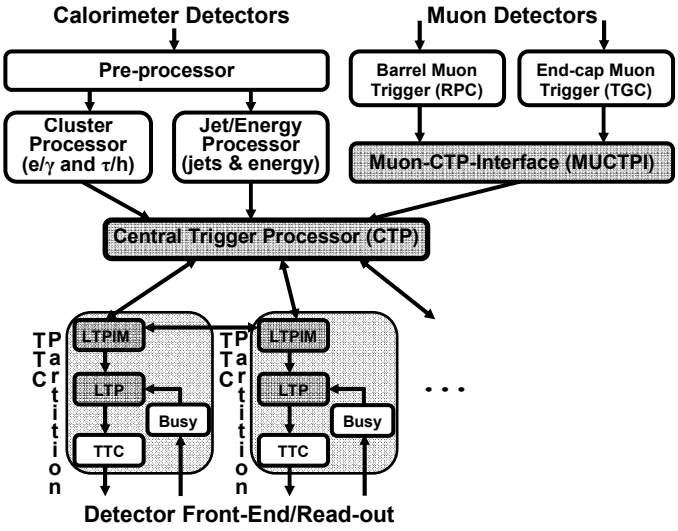

Figure 1: Overview of the ATLAS Level-1 Trigger

The MUCTPI [2] combines trigger information from the two dedicated muon trigger detectors, the Resistive Plate Chambers (RPC) in the barrel and the Thin-Gap Chambers (TGC) in the end-cap region. The CTP [3] forms the Level-1 trigger decision (accept or reject) for every BC, and distributes it to the TTC partitions. It also receives timing signals from the LHC and fans them out to the TTC partitions. The TTC partitions perform the distribution of the timing, trigger and control signals to all sub-detector front-end electronics. In the ATLAS experiment there are about 40 TTC partitions. For a full overview see [4].

II. The MUCTPI

The MUCTPI [2] receives the muon candidates from all 208 trigger sectors, calculates multiplicities for six programmable p_T thresholds and sends the results to the CTP. It resolves cases where a single muon traverses more than one sector and thus avoids double counting. The MUCTPI sends summary information to the Level-2 trigger and to the data acquisition (DAQ). It identifies, in particular, regions of interest (RoI) for the Level-2 trigger processing. The MUCTPI can also take snapshots of the incoming sector data for diagnostics and accumulate rates of incoming muon candidates for monitoring.

The MUCTPI is implemented as a single-crate 9U VME system with three different types of modules and a dedicated active backplane as shown in Figure 2.

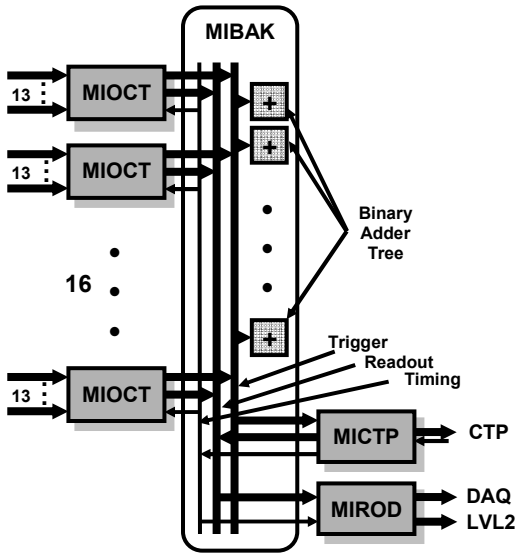

Figure 2: Overview of the MUCTPI

The octant module (MIOCT) receives the muon candidates from the trigger sector logic and resolves overlaps. The active backplane (MIBAK) performs the multiplicity summing, the readout transfer and the timing signal distribution. The CTP interface module (MICTP) receives timing and trigger signals from the CTP and sends multiplicities to the CTP. The readout driver module (MIROD) sends summary information to the Level-2 trigger and the DAQ.

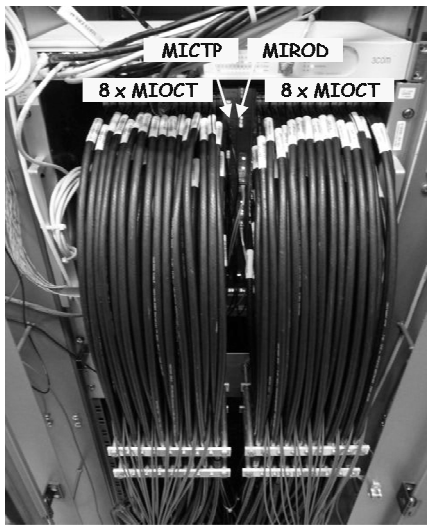

Figure 3: The MUCTPI in ATLAS

A prototype of the MUCTPI was installed in the experiment in 2005. It provided almost full functionality and missed only some flexibility in the overlap handling. The MUCTPI has been upgraded incrementally to the final system. Figure 3 shows the setup in the experiment with 16 MIOCTs, the MIROD and the MICTP. The MICTP is currently the last prototype module which, although it provides full functionality, will soon be replaced by a new MICTP. The new MICTP is based on a more recent FPGA allowing all logic to be in a single device. It also uses the same PCB as the MIROD. This is useful for providing spares to the MUCTPI. Another complete and another partial MUCTPI are available in the laboratory as spares as well as for firmware modification and software development.

III. THE CTP

The CTP [3] receives, synchronizes and aligns trigger inputs from calorimeter and muon triggers, and others. It generates the Level-1 Accept (L1A) according to a programmable trigger menu. The CTP has, in addition, the following functionality: it generates a trigger-type word accompanying every L1A; it generates preventive dead time in order to prevent front-end buffers from overflowing; it generates summary information for the Level-2 trigger and the DAQ; it generates a precise time stamp using GPS with a relative precision of 5 ns and an expected absolute precision of 25 ns after calibration; it generates other timing signals like the Event Counter Reset (ECR). The CTP can measure the timing of the trigger inputs which is very important during commissioning. It can take snapshots of the incoming trigger inputs for diagnostics and accumulate rates of incoming trigger inputs and internally generated trigger combinations for monitoring.

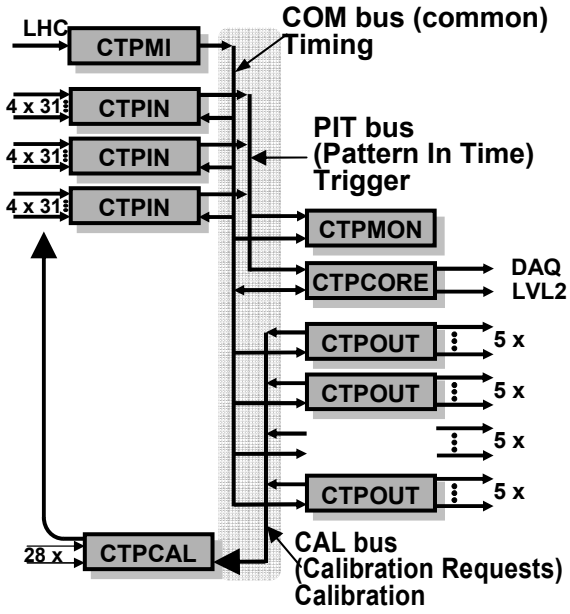

Figure 4: Overview of the CTP

The CTP is implemented as a single-crate 9U VME system with six different types of modules and three dedicated backplanes as shown in Figure 4. The machine interface module (CTPMI) receives timing signals from the LHC. The input module (CTPIN) receives trigger input signals, synchronizes and aligns them, and sends them to the Pattern-In-Time (PIT) backplane using a switch matrix. The monitoring module (CTPMON) performs bunch-by-bunch

monitoring. The core module (CTPCORE) forms the L1A using Look-Up Tables (LUTs) and Content-Addressable Memories (CAMs), and sends summary information to the Level-2 trigger and the DAQ. The output module (CTPOUT) sends timing signals to the TTC partitions and receives calibration requests. The calibration module (CTPCAL) time-multiplexes the calibration requests of the detectors and receives additional front panel inputs. The Pattern-In-Time (PIT) bus transports the synchronized and aligned trigger signals from the CTPINs to the CTPCORE and the CTPMON. The common (COM) bus contains timing signals. The calibration (CAL) bus transports the calibration requests from the CTPOUTs to the CTPCAL.

The final CTP was installed in the experiment in 2006. Figure 5 shows the CTP with the CTPMI, three CTPINs, the CTPMON, the CTPCORE, four CTPOUTs, and the CTPCAL. There is an additional NIM-to-LVDS fan-in module for receiving NIM trigger signals and routing them to one of the CTPINs. Another two complete CTPs are available in the laboratory as spares as well as for firmware modification and software development.

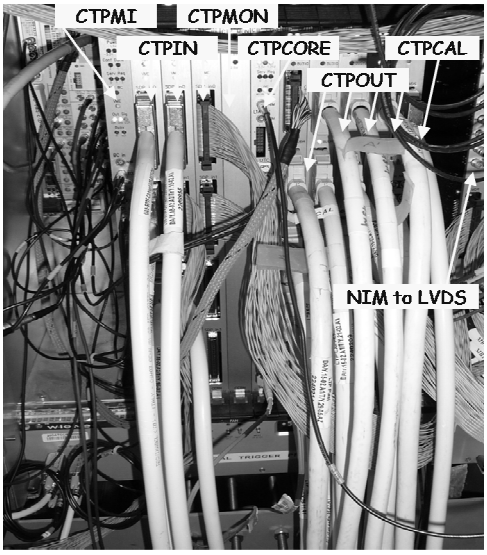

Figure 5: The CTP in ATLAS

IV. Test Framework

A. Principles and Architecture

The problem of the test framework is the considerable number of different types of modules. In the CTP there are six different types, in the MUCTPI three. There are in total enough modules to populate two full systems of each type and a third partially. One of each system is installed in the experiment, the other two in the laboratory. The MUCTPI and CTP are also relatively complex: There is a large number and size of inputs. There are many parameters for configuration and processing. And there are many different use cases, in particular for testing of prototypes which requires a rapid evolution of the firmware and software, for testing of the modules which guarantees the quality of the production, and for operation which provides the integration into the experiment.

The test framework is based on several principles. The VME interface is the same for all modules. This is true for the hardware whose design is a copy from module to module, for the firmware which is used like an IP block [5], and for the software, i.e. the VME drivers and libraries. The modules were also right from the beginning designed with diagnostic memories which can be used for input to capture data, or for output to inject data into the processing. The modules were also designed with readout facilities and counters. The event-like readout is used for monitoring, and the counters which can be integrating or on a bunch-by-bunch basis allow counting of data or of the BUSY status at several stages of the processing. The entire test framework is based on the common software framework provided by the ATLAS Readout Driver Crate DAQ (RCD) [6]. This framework contains the ATLAS VME driver and library, contains many utilities for bit strings, modules, JTAG chains, menus, and components like the HPTDC and DELAY25 chips. The framework also provides access to the ATLAS TDAQ control system.

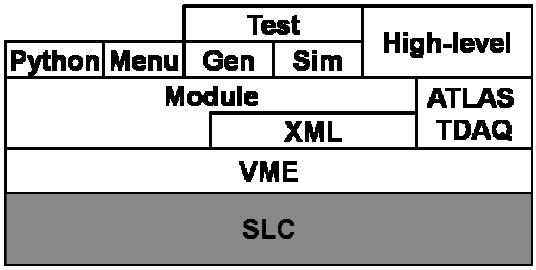

Figure 6: The Framework Architecture

The test framework is organised in several layers, see Figure 6. At the lowest layer is the Scientific Linux CERN (SLC). On top of that is the ATLAS RCD framework with its VME driver and library. On top of that are the module libraries for each type of module. These libraries are partly generated automatically from XML files and partly hand coded. They consist of C++ objects with methods for access of all functionality provided by any module. On top of the module libraries are PYTHON scripts which are used for rapid testing and menu programs which give full access to all functionality. Ancillary objects are generators for providing test data and simulators for behavioural simulation of the modules. On top of these are the test programs which provide a test suite for all modules and systems. On top of the ATLAS TDAQ control software is the high-level software which allows one to configure, control and monitor the modules in the experiment.

B. Low-level Software

The low-level software is based on the module libraries. Part of the module libraries can be generated in an automated way from XML code. The main idea is that a module's VME map is described in an XML file in terms of a module, its blocks, the registers in the blocks, the fields in the registers

and the possible values for each field. An excerpt of such a description can be seen in Figure 7. A (pre-) compiler is then run over the XML file which generates C++ classes for bit string objects for all fields and registers and a C++ class for the module with read and write methods for all registers and fields using the bit strings. These methods can be used in test programs in a very simple and intuitive way. Future extensions foreseen to the tool are to add more detail, e.g. read-only, write-only, read-modify-write functions, and to add support for more complex parts of the VME map like memories and block transfers.

```
<module name="MICTP" type="A32" size="0x00080000" … >
  <block name="Readout">
    <register name="MultiplicityConfig"   addr="0x00000200">
      <field name="RamEnable"             mask="0x00000002">
        <value name="DISABLED"            data="0x00000000"/>
        <value name="ENABLED"             data="0x00000002"/>
    </field>
…
```

Figure 7: Excerpt of a Module's VME Map using XML

The C++ class automatically generated by the XML compiler is augmented by code containing all higher level functions for sequences of operations as needed by the module. Then a menu program is developed from the module library which is based on a text-driven menu and provides access to all methods and thus all registers and fields of the module. The code of the menu program can easily be extended whenever new features are included into the module library. It is foreseen in the future to develop a tool to derive the menu automatically from the module library. There exists a menu program for each type of module which gives detailed and complete control over the module and which is intended to be used by an Level-1 central trigger expert.

In addition to the menu program for each type of module there also is a generator for generating input data for single-module and full-system tests. The patterns generated include counter-like patterns, walking ones, a toggling pattern, random data, and more complex data with lots of overlapping candidates for the MIOCT testing. Also for each type of module there is a simulator which uses the same configuration as the hardware module and which generates the expected output data from given input data. This can be used for comparison between observed and expected data in tests. The simulation includes, e.g. the overlap handling of the MIOCT modules, the data processing for readout and counters of the MIOCT, MICTP, MIROD, CTPIN, CTPCORE and CTPMON modules, as well as the complete trigger generation in the CTPCORE module.

C. Test Suite

Based on the module libraries, the generators, and simulators there is a suite of tests programs for single-module and full-system tests. The single-module tests usually test register and memory access, and are based on read-write tests. There are also single-module initialisation programs which write a default configuration to the module and which read back the configuration if asked to do so. The more interesting tests concern several modules or full systems. They usually use data from the generator or a file to load into the modules, loop over the data, read back data from readout or counters and compare them to simulation. As an example, the "testCtpReadout" configures the CTP, loads the CTPIN test memories with data which will generate a L1A, starts the trigger generation by enabling the CTPIN test memories and removing the BUSY from the CTPMI, and reads the data from the CTPCORE readout FIFOs and compares them to the expected data from simulation. This tests the full chain from CTPIN memory and switching matrix, the PIT bus, the CTPCORE LUT and CAM processing, as well as the CTP timing.

Other programs in the test suite are concerned with timing alignment which is very important for the Level-1 trigger and the experiment. Using data from the CTPIN test memories or the trigger inputs with a single candidate per orbit of the LHC the CTPIN, CTPMON, and CTPCORE BC Identifier (BCID) values can be aligned with respect to the BCID in the CTPCORE readout by using the BCID offsets at several stages in the processing. Similarly, using data from the MIOCT test memories or the muon sectors sending a test pattern, the MIOCT, MICTP and MIROD BCID values can be aligned with respect to the BCID in the MICTP using the BCID offsets in several stages in the modules as well as the MIOCT muon sector data pipelines. As a consistency check the MIOCT can capture the muon sector data in its test memory and compare the muon sector BCID offsets with the MIOCT BCID.

D. High-level Software

The high-level software provides integration of the MUCTPI and CTP into the experiment by supplying configuration, control, and monitoring to the ATLAS TDAQ control system [7].

The trigger configuration is taken from the ATLAS trigger database which stores the event selection strategy comprising the Level-1 trigger, Level-2 trigger, and Event Filter (Level-3 trigger). The trigger tool is a graphical user interface which allows one to browse and edit all trigger menus in the trigger database. The trigger menu compiler automatically translates the high-level description of the Level-1 trigger menu to all necessary configuration files of the CTP for loading the CTPIN switch matrices and the CTPCORE LUTs and CAMs [8].

In order to be integrated with the ATLAS TDAQ control system each module type needs to be described in a schema in the ATLAS configuration database. Such a schema contains the full configuration data, except for the trigger configuration which comes from the trigger database. The schema also contains provision for describing the flow of data between modules. This allows for an automatic setup of the inputs of the BUSY (including S-Link XOFFs) and MUCTPI sectors. A plug-in for each module type into the standard RCD controller provides the dynamic aspect of control in the sense that during

setup the configuration of each module is read from the configuration database and written into the module using the low-level library. In order to organise the setup in logical sets some plug-ins span several modules, e.g. all active MIOCTs of the MUCTPI, all active CTPINs of the CTP or the BUSY monitoring of the CTP which reads from all CTP modules.

E. Monitoring

The monitoring of the MCUTPI and the CTP is based on the principle of a producer-consumer model: the producer of information is (part of) a plug-in controller which sends data to the ATLAS information service (IS) [9], a network-based information exchange system. The consumer reads data from the IS and analyses and presents the information, usually in the form of a graphical user interface developed using Qt. As an example, the display of the ATLAS MIOCT Monitoring GUI can be seen in Figure 8. It shows the status of all MIOCTs of the MUCTPI.

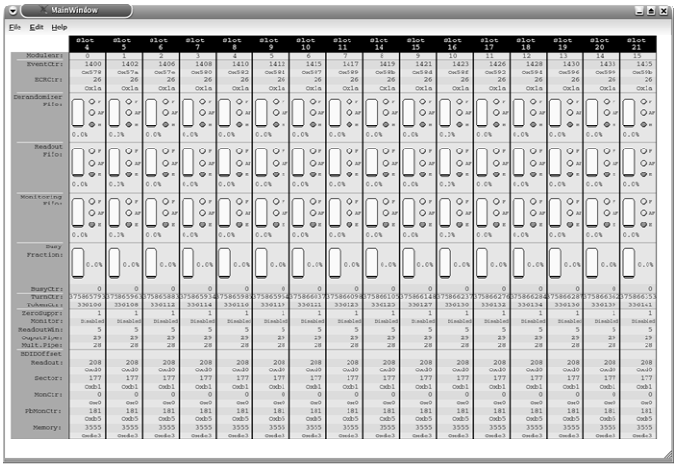

Figure 8: The MIOCT Monitoring GUI

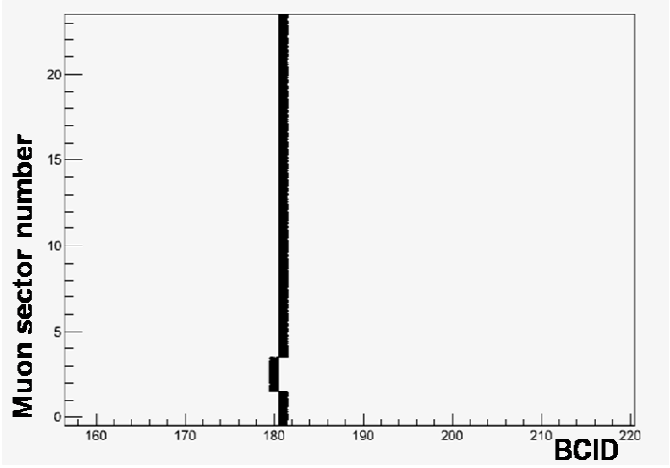

Figure 9: The MIOCT Rate Monitoring

Some regular monitoring tasks are run for data quality monitoring, e.g. the timing-in of the MUCTPI can be checked by reading the per-bunch counters of the MIOCTs and writing the data into histograms. In Figure 9 an example can be seen for a special run where the muon sectors sent a known test pattern. One can clearly see that one of the sectors is wrongly aligned in BCID. This was corrected in the configuration database and used from the next run onwards.

V. SUMMARY

The ATLAS Level-1 MUCTPI and CTP framework for testing and operation covers single-module and full-system testing, provides integration into the experiment, as well as many monitoring facilities. With these the ATLAS Level-1 central trigger is ready for taking data with beam.

VI. REFERENCES

[1] The ATLAS Collaboration, "*The ATLAS Experiment at the CERN Large Hadron Collider*", JINST 3 (2008) S08003.

[2] S. Haas *et al.*, "*The ATLAS Level-1 Muon to Central Trigger Processor Interface*", Topical Workshop on Electronics for Particle Physics, CERN-2007-007, November 2007.

[3] R. Spiwoks *et al.*, "*The ATLAS Level-1 Central Trigger Processor (CTP)*", 11th Workshop on Electronics for LHC and Future Experiments, CERN/LHCC/2005/038 265, November 2005.

[4] S. Ask et al., "*The ATLAS Central Level-1 Trigger Logic and TTC System*", JINST 3 (2008) P08002.

[5] R. Spiwoks, "*The VMEbus Interface of the Central Trigger Processor*", https://edms.cern.ch/document/428910.

[6] S. Gameiro *et al.*, "*The ROD Crate DAQ Software Framework of the ATLAS Data Acquisition System*", IEEE Trans. Nucl. Sci. **53** (2006) 907-911.

[7] The ATLAS Collaboration, "*ATLAS High-level, Trigger, Data Acquisition, Controls*", Technical Design Report, CERN/LHCC/2003-022.

[8] R. Spiwoks *et al.*, "*Configuration of the ATLAS Trigger*", 11th Workshop on Electronics for LHC and Future Experiments, CERN/LHCC/2005/038 269, November 2005.

[9] A. Corsu-Radu et al., "*First-Year Experience from the ATLAS Online Monitoring Framework*", 17th International Conference on Computing in High-Energy and Nuclear Physics", http://cdsweb.cern.ch/record/1181482, March 2009.