TTCScope – A scope-based TTC analysis tool

Per Moosavi*

Supervisor: Christian Ohm

PH-ADT-TR, CERN

August 23, 2013

Abstract

This document describes a scope-based analysis tool for the TTC system. The software, `ttcscope`, was designed to sample the encoded TTC signal from a TTCex module with the aid of a LeCroy WaveRunner oscilloscope. From the sampled signal, the bunch crossing clock is recovered, along with the signal contents: level-1 accepts, TTC commands, and trigger types. Two use-cases are addressed: analysis of TTC signals and calibration of TTC crates. The latter includes calibration schemes for two signal phase shifts, one related to level-1 accepts, and the other to TTC commands.

Introduction

In the ATLAS first-level trigger, the facilities through which the Central Trigger Processor (CTP) send instructions to the various subdetectors, are provided by the Trigger, Timing and Control (TTC) system. The subdetectors are organised into 40 TTC partitions. Each partition has one Local Trigger Processor (LTP) and one TTCvi module. The latter assembles the various electrical signals received from the CTP into two channels: one A and one B channel. The first channel carries level-1 accepts (L1As), while the second is used for transmitting addressed (long) and broadcast (short) commands. The relay to the individual readout systems in a partition is handled via a TTCex module. In this module, the A and B channel signals are combined into a multiplexed electrical signal. This is then converted to an optical signal, which is sent at a 2:1 rate with respect to the 40.08 MHz bunch crossing (BC) clock. A total of 10 optical outputs are provided on the front panel of the TTCex module. Using optical tree couplers, the signal is fanned out over an optical network to the readout electronics in the partition. At the receiving end, there are TTCrx chips, which handle the reception of the optical signals. These are ASICs specifically designed to recover the BC clock, separate the A and B channels, and identify L1As and TTC commands [1, 4, 5].

This document gives a brief description of a scope-based analysis tool for the TTC system – [3] is the full-length version. The software, titled `ttcscope`, was written in C++ and is intended to be used together with a LeCroy WaveRunner digital storage oscilloscope (DSO) for the purpose of analysing the encoded TTC signal. To this end, the software essentially replicates the work carried out by the TTCrx. However, the software-based processes are considerably slower than their hardware counter-parts. The software is therefore limited to analysing segments of data, as opposed to real-time data handled by the TTCrx.

*Electronic address: `pmoosavi@kth.se`

TTC SYSTEM

The idea, underlying the development of the software, was to analyse the encoded TTC signal with the aid of an oscilloscope. A signal sequence is received from the TTCex module, either via one of its ENC outputs*, or with the use of an optical probe. From this sequence, the BC clock and the A and B channel contents can be reconstructed. Due to the nature of the setup, only segments of the TTC data stream can be analysed. However, the user has the ability to trigger the scope at the appropriate moment based on external signals, e.g. an L1A, a BGo signal or an inhibit signal from the TTCvi [4].

The signal, sampled from a TTCex module, is encoded using differential Manchester encoding. On top of this, the signal is multiplexed, which in this case means that adjacent bits in the signal belong to different channels. In Figure 1, the "building blocks" of the TTC signal are shown. Their lengths correspond to one BC clock tick (≈ 25.95 ns). The building blocks can be combined to form a TTC signal sequence, with some restrictions: only 11 consecutive L1As are allowed. If simply repeated, the second building block, i.e. a 0 (1) sent over the A (B) channel, corresponds to the idle state. Consequently, using the above restriction, the channels can be identified in this state [1].

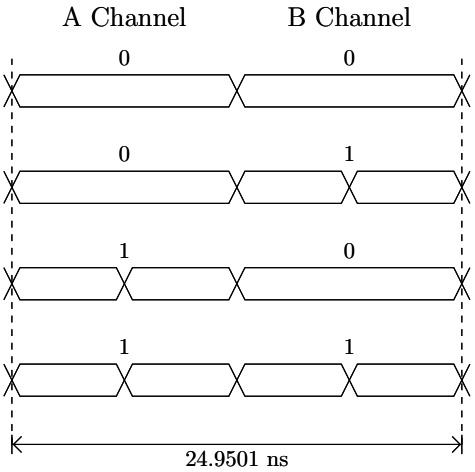

Figure 1: Differential Manchester encoding. The upper two building blocks contain no L1A, while the lower two building blocks contain one L1A each.

The commands, sent over the B channel, can be of two types: short (16 bits) or long (42 bits); see Figure 2. The former are broadcast commands sent throughout a given partition. The latter are addressed commands, used to send instructions to specific groups of readout electronics. Two types of short commands are of special interest: (i) Bunch Counter Resets (BCRs), and (ii) Event Counter Resets (ECRs). Their names are essentially self-explanatory. Regarding long commands, we will only concern ourselves with one of their tasks, namely the relay of the trigger type associated with each L1A. These are composed of four long commands, with subaddresses functioning as labels. The data content in the first long command contains the 8-bit trigger type, while the other three long commands, taken together, form a 24-bit event or orbit counter [1, 2].

*These are electrical outputs, designed for monitoring the system.

(a) Short command

(b) Long command

Figure 2: Templates for short and long commands. In the hardware, the two types of commands are distinguished by the second leftmost bit. The last part in a command contains a Hamming code checksum (CHCK), which is single-error correcting and double-error detecting.

Software design

The software consists of two seprate parts: (i) waveform acquisition, and (ii) waveform analysis. The software was designed such that these are detached. Moreover, it has minimal dependencies*: the only external dependencies belong to the Boost C++ libraries.

The TTC protocol calls for a generic container class in order to contain the aforementioned messages: L1As, short commands, and long commands. In addition, a trigger type is created whenever four long commands, sent consecutively with the correct subaddresses, are detected. Each message type has its own class, which inherit from the same base class. Consequently, all messages can easily be put into the same container class. In our case, lists are used exclusively for this purpose. For a review of the members included in the message-specific classes, the reader is referred to the source code. Moreover, most of them need no further clarification, apart from those relating to the chronological order in which the messages are received. For instance, the command classes contain the information seen in Figure 2. The only essential addition is a boolean variable informing whether the checksums are consistent or not. On the other hand, each L1A is given a 32-bit pseudo-L1ID, reminiscent of the L1ID given in the hardware. Thus, it consists of a 24-bit event counter and a 8-bit ECR counter, where the former is reset by an ECR.

Throughout the software, individual bits and commands in a TTC signal sequence are given timestamps corresponding to the end of their reception. This has the following implications: (i) Bits are given timestamps corresponding to the end of the waveform segments from which they were recovered. (ii) Commands are given timestamps corresponding to the last received bit in the command. (iii) Trigger types are given timestamps corresponding to the last long command in the trigger type. Furthermore, the timestamps are always given in terms of whole BC clock ticks with respect to the A channel. In addition, a 12-bit BCID is created for each message. This number functions as a timestamp, except that it is reset by a BCR. Since it is impossible to know the BCID before the software has identified the first BCR, it is only actively calculated for messages received after this has occured.

As a first step in the decoding process, the TTC signal is quantised and the BC clock is recovered. This is achieved by identifying the positions of the wavefronts in the signal sequence; cf. Figure 1. The exact timestamps are lost in quantisation process – only a list containing the bits (boolean variables) is retained after this step. Instead, the timestamps

*The rationale behind this was the ambition to build the software on a laptop. The user can thus bring the setup to the TTC crates, installed in USA15, for on-site analysis or calibration.

are calculated based on the positions of the bits in the list. Therefore, it is necessary to "recover" the BC clock concurrently with the quantisation process. This is achieved by calculating a running average for the BC clock period.

From the list containing the signal bits, the A and B channel signals can be separated. This is done using the previously mentioned requirement on the L1As. After the separation, the bits are stored in two lists, one for each channel. In the next step, short and long commands are identified by sweeping over the B channel list. With the commands identified, specifically the BCRs and ECRs, the L1As can be isolated and assigned correct BCIDs and pseudo-L1IDs. Concurrently, the trigger types are identified. The final result is stored in a combined list, ordered with respect to the timestamps.

The software supports two means of waveform acquisition. The first utilises the link to the oscilloscope, while the other makes use of a waveform which has previously been saved to disk. The interaction with the oscilloscope is handled by a scope-specific class. Only the LeCroy WaveRunner range of oscilloscopes is supported. Assuming correct scope settings are used, the user only has to provide an IP address and a channel number. At run-time, the software arms the trigger, predefined in the scope, and waits for the trigger criterion to be fulfilled. When the scope triggers, the waveform from the selected channel is retrieved by the software. It is subsequently passed to the analysis part, described above, in order for the signal to be decoded.

Use-cases

The software can be built using CMT or Make. The latter is intended for external machines, e.g. a laptop. A number of required (optional) inputs must (can) be given at execution. Specifically, a waveform input source must always be provided, either in the form of a scope IP address, or in the form of a path to a waveform file. Operation in connection with an oscilloscope requires correct scope settings. It is crucial that the sampling frequency equals or exceeds 500 MHz. Otherwise, the density of the data sample points will not be sufficient for decoding the TTC signal. The choice of trigger must also be specified by the user. Using the default software settings, the scope waits for a new trigger, stops temporarily, then returns to normal mode after the waveform has been retrieved by the software.

One of the more straightforward use-cases involves monitoring the TTC system for analysis purposes. Given a trigger criterion, the setup can be left to its own devices, waiting for a desired occurrence to take place. In particular, statistical measures, based on the observed L1As and TTC commands, can easily be created and compared with the expected outcome under the given circumstances.

The software can also be used to calibrate the TTC system, in this case two signal phase shifts, which need to be properly adjusted for the modules in a TTC crate. Two calibration schemes are proposed: (i) calibration of the *BC delay* in a TTCvi module, and (ii) calibration of the A channel signal timing.

(i) The phase of the BC clock received by the TTCvi can be altered by turning the *BC delay rotary switch* on the front panel of the TTCvi. This allows for adjusting the phase of the internally-generated A and B channel signals. In turn, this enables cables of different lengths to be used between the TTCvi and the TTCex. If this phase is not

properly adjusted, the TTCex will strobe the TTC signal at the wrong moments, causing its contents to be distorted [2, 4]. The calibration of the BC delay can be done using a scope probe; see [4]. However, `ttcscope` provides the user with one additional tool: since the TTC signal contents are distorted when the wrong settings are used, the majority of commands identified by the software will not be consistent. Hence, by monitoring the outcomes of changing the BC delay, the correct settings for the TTCvi can be discerned.

(ii) In order to minimise the latency for L1As, the TTCvi does not resynchronise the L1As received from an external trigger source, e.g. the CTP or an LTP. Thus, the timing is not altered by turning the BC delay rotary switch on the TTCvi front panel. The phase of the L1As must therefore be calibrated at the external trigger source, or the TTCex will strobe the L1As at the wrong moments [4]. As in the previous case, this phase can be calibrated using a scope probe; see [4]. However, with `ttcscope`, another approach is available. Suppose an LTP functions as our external trigger source, and assume it is set up such that a fixed number of L1As is sent per orbit signal, i.e. for each revolution in the LHC. For certain values of the phase, the number of L1As between two consecutive BCRs will deviate from the preset value. Thus, by monitoring the number of L1As between two consecutive BCRs, a range of allowed values for the phase can be obtained.

Conclusion

A scope-based analysis tool for the TTC system has been developed. The software was written in C++ and can be built on any Linux/UNIX machine using CMT or Make.

Two use-cases have been addressed, one for the specific purpose of calibrating the TTC system. The feasibility of the calibration schemes have yet to be evaluated.

Since there is no dependency on ATLAS-specific software, only minor changes are likely needed in order to configure the software for use with other implementations of the TTC system. Thus, even though the software was designed with ATLAS in mind, other experiments at CERN should be able to use it.

References

[1] Christiansen, J., et al. TTCrx reference manual – a timing, trigger and control receiver ASIC for LHC detectors. RD12 working document, Rev. 3.11, 2005.

[2] Farthouat, P., and Gällnö, P. TTC-VMEbus interface – TTCvi-MkII. RD12 working document, Rev. 1.6, 2000.

[3] Moosavi, P., Ohm, C., and Pauly, T. TTCScope - an analysis tool for the TTC system. CERN-OPEN-2013-020, 2013.

[4] Taylor, B. G. TTC laser transmitter (TTCex, TTCtx, TTCmx) user manual. RD12 working document, Rev. 2.0.

[5] The ATLAS Collaboration. Level-1 trigger technical design report. CERN-LHCC-98-014, 1998.