DISTRIBUTED TIMING DIAGNOSTIC APPLICATIONS

Paul Kennerley, Ioan Kozsar, Julian Lewis, Javier Serrano

CERN, Geneva, Switzerland

Abstract

The CERN timing system delivers events to the accelerator complex via a distribution network to receiver modules located around the laboratory. These modules generate pulses for nearby equipment and interrupts for the local host. Despite careful planning, hardware failure and human error can lead to anomalies within the control system. Diagnosing such errors requires a formal description of the logical and topological timing layout. This paper describes the design and implementation of a suite of timing diagnostic software applications which allow users to quickly diagnose and remedy faults within the CERN timing system.

TIMING SYSTEM OVERVIEW

At the heart of the timing system is the Central Beam and Cycle Manager [1] CBCM which may be considered as a scheduler for the CERN accelerator network. The CBCM distributes timing events over a network of cables to timing receiver cards located near end user equipment. On start up, each card is configured with control values which specify actions that should be performed for each accelerator cycle and which events trigger them. Examples of control values are the counter delay, start, clock, mode, output channel, output enable state etc. A counter output may in turn be used to start other counters forming multiple chains of dependencies before finally triggering end user equipment. These counter chains depend on the physical cabling, on static and dynamic configuration data and on the machine cycles being executed. An example of equipment that relies on such a chain is an extraction kicker. For a given accelerator cycle with the beam destination the kicker sends the beam to, the kicker must pulse, but in other cycles with a different destination it does not. In this example, a counter might be loaded and started by a CBCM extraction timing event. Its output may be used to start timing chains in the extraction process of an accelerator and elsewhere. The next counter in the kicker extraction chain may be programmed to count revolution frequency ticks. Its output may in turn start a third counter counting RF ticks, before finally pulsing the extraction kicker. This is typical of the situation found in extraction and injection timing systems.

DIAGNOSING FAULTS

When a fault prevents a timing pulse from being received or causes it to be received at an incorrect time, then diagnosing the fault from the accelerator behaviour is very difficult. Common causes of faults may include; defective cabling, timing receiver hardware failures, incorrect control values, among others. If the machine operators suspect a timing fault they must follow potentially faulty chains in the suspect areas of the controls system verifying each counter. Hence diagnosing a suspected fault requires locating the area within the timing network in which the fault is suspected, running SQL queries against the controls database to discover the physical location of each timing cable and which timing cards are involved in producing the pulse. It is then a matter checking control values for each timing card against the controls database and checking each timing receiver hardware status. Once a fault is detected the implied timing cables and modules can be checked physically.

DIAGNOSTIC SOFTWARE

To reduce the time and effort required for diagnosing faults, a suite of Timing diagnostic applications has been developed. Inspired by the Google Earth application, the first component of the Timing Diagnostic suite is a Java based desktop application which allows users to browse the map of counters by name, channel, timing card, front end computer or by accelerator. It also allows users to select the complete network of timing counters on which a given counter depends.

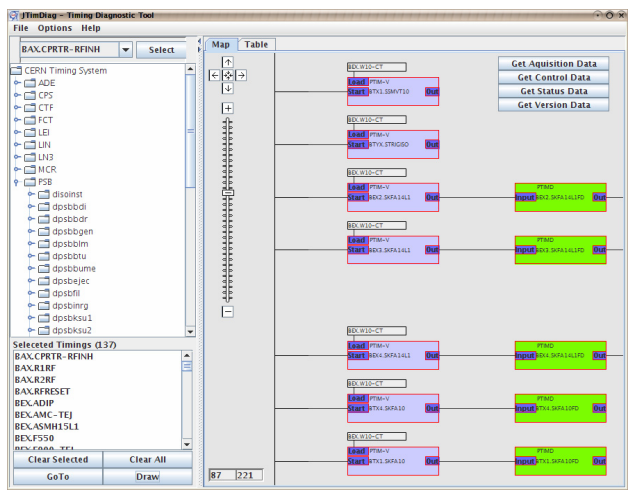


Figure 1: Mapping Tool

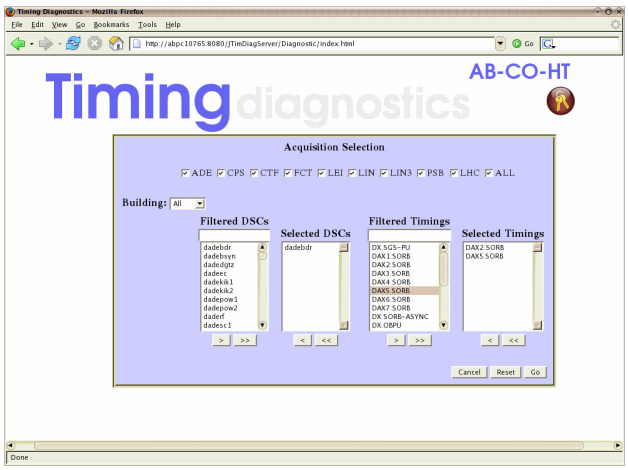
The map displays the physical location of the timing cards on which the counters are located, along with the identification numbers of the connecting cables.

This information allows users to quickly identify the 'route' of dependencies on which an event depends. Once the route has been established, it is possible to determine

which and in what order the timing cables and timing receiver cards should be checked for malfunction.

OBTAINING CONTROL VALUES

Although the mapping software displays the hardware route over which timing signals may pass, it does not show the current control values contained within the timing receiver cards.



The second component in the diagnostic suite is a web application that allows the user to make specific counter selections. The selection interface (shown above) allows timings to be selected from a pool that can be filtered by machine, building, front end computer or regular expression. Changes to the filtering criteria results in an update of the list of selectable timings shown within the selection pool.

The following gives an overview of the type of information that can be retrieved and how it assists in diagnosing faults within the timing system:

- Status information for timing receivers is displayed for past bus errors, phase lock loop errors, General Machine Timing reception errors or self test failed. The timing reception enable/disable state for the card is also displayed. A history of previous errors is available which includes the type of error, how many times it occurred and the time at which it was last detected.
- Control values are obtained for each accelerator cycle and the static parts are compared with configuration data in the controls database. Any discrepancies are highlighted. Dynamic settings such as the current delay and the output enable status are not stored in the database and can not be compared.
- Acquisition values are read for every cycle over the last complete super cycle and include information such the central timing event that loaded the counter, the clock used, and the UTC and cycle times at which the timing counter made an output.
- The version for the host timing library, device driver and each modules VHDL code version.

The web based application also contains online help and a link through which users can download the Timing Diagnostic Mapping application via Java Web Start [6].

DESIGN, ARCHITECTURE & TECHNOLOGY

Mapping Application

The mapping application has been implemented using Java and the embedded version of the open source Derby database which contains the static control values and the logical relationships for each timing counter.

Timing counters may have multiple inputs and multiple outputs. The logical timing map can therefore be considered as a directed graph data structure. Given that there are currently more than 5000 timing counters within the timing system, creating a rule based system for rendering such a data structure would be extremely complex and time consuming. Therefore the task has been simplified. For each child timing counter that has multiple parents, the child timing counter (and its subsequent sub tree) is displayed multiple times, i.e. once for each parent. In effect, this reduces the directed graph into a collection of tree data structures. This greatly reduces the complexity of rendering the data structure as a logical map and is a common technique that is often used to simplify complex electrical diagrams.

The following outlines the algorithm for generating the tree data structures. First, a list of all selected timings is obtained. Two lists are then associated with each timing counter, the first list containing the parent nodes and the second list containing the child nodes. Those nodes without a parent node are considered to be root nodes, each of which is the root of a tree data structure. Note that counters within separate logical maps may be selected resulting in multiple trees being created.

A matrix is then created, with the ***Y*** dimension equal to the sum of the maximum width of each tree, and the ***X*** equal to the depth of the deepest tree. Each tree is then placed within the matrix, starting with the root node in the left most column. The position of each node within the matrix then represents the position of the corresponding timing counter in the logical map. Creating the logical map is then simply a matter of iterating over the matrix and rendering each counter at the co-ordinates that correspond to the position within the matrix.

Data Acquisition Application

The web based data acquisition application has been designed using a three tier approach.

The top tier, also known as the presentation tier, has been implemented using XHTML, JavaScript and CSS in a style commonly referred to as AJAX [3]. By utilizing the Document Object Model DOM [4] and the XMLHttpRequest [5] interface, the web interface displays the kind of interactive behaviour usually associated with traditional desktop applications.

Once a user has made a selection, the presentation tier organises the selected values into a HTTP POST request

which is then sent asynchronously to the business tier using the XMLHttpRequest interface.

The asynchronous nature of the HTTP POST request is then exploited to allow the client to display a graphical representation showing the progress of the request and even allows the user to cancel it.

The response is then received by the client in an XML format. Using the DOM interface, the client accesses the contents of the returned XML and displays it in tabular form within the client interface. The sorting, hiding and comparing of information is all achieved by manipulating the DOM.

The middle tier, often referred to as the business tier has been implemented using Java servlet technology [2] and resides within an Apache Tomcat servlet engine which acts as both a web server and servlet engine. On receiving a request, the business tier first determines the type of information that is being requested, and then obtains the information from the appropriate component from within the resource tier. This information is then marshalled into an XML format and is returned to the client.

Bottom Tier

The bottom tier, often referred to as the Resource Tier, consists of a lightweight database and a number of daemons, each of which runs on a front end computer.

The database contains a "snapshot" of the configuration data related to the timing system that is contained within the central controls database.

The business tier accesses the database using the JDBC interface and uses the obtained information to satisfy requests for static timing information.

It was decided to create an independent database instead of directly accessing the main controls database due to the fact that the main timing database is not optimized for accessing timing related data and the queries required to obtain such data are both complex and resource intensive. Using a separate database also provides a level of indirection between the diagnostics application and the main controls database. Any changes to the table schema within the central controls database, does not result in changes to the diagnostics application. Instead, the only change required would be to a script that is responsible for data extraction and population.

The daemons running on front end computers are responsible for providing data to satisfy requests concerning dynamic values within the timing system. On receiving a request for dynamic data, the business tier executes an external legacy process, passing details of the data required as parameters.

The legacy process then sends requests via the UDP protocol to each front end computer from which information is required. Each daemon responds with the required information which the legacy application writes to a text file. The legacy application then terminates and the text file is read by the business tier, transformed into an XML format and then sent to the client tier.

CONCLUSION AND FURTHER WORK

The suite of applications introduced here will reduce the time and effort required to diagnose faults within the CERN accelerator timing network.

The project has shown that there are many advantages employing a web based presentation tier. These include platform independence, resolution of deployment problems and it ensures that the latest version of the application is used as the web browser automatically downloads the latest version of the JavaScript, XHTML and related resources each time the application is accessed.

Due to the complexities involved with rendering logical timing maps and the time constraints imposed on the project, it was decided to implement the logical mapping component using Java Swing technology. However, with the recent introduction of the HTML Canvas tag, it is now feasible to implement a pure web based timing diagnostic suite of applications.

CERN network security policy restrictions currently prevent access to the web application component from outside of the CERN network. It is hoped that further research and development into web and network security will allow this as it will permit timing diagnostics to be carried out from anywhere in the world where there is a internet connection

REFERENCES

[1] The Evolution of the CERN SPS Timing System for the LHC Era. ICALEPCS 2003
[2] Servlet container http://java.sun.com/products/servlet
[3] AJAX www.adaptivepath.com/ideas/essays/archives/000385.php
[4] Document Object Model www.w3.org/DOM/
[5] XMLHttpRequest http://www.w3.org/TR/XMLHttpRequest/
[6] JavaWebstart http://java.sun.com/products/javawebstart/