BEAM

CERN Trajectory Measurement System

TmsTesting

Project	CERN-TMS
Date	2024-09-28
Reference	Cern-tms/TmsTesting
Version	2.3.0
Author	Dr Terry Barnaby

Table of Contents

1. Introduction

This document covers the testing of the TMS system. It describes the basic test programs provided, provides an overview of test methods and describes the script based test system used.

There are a few programs that have been developed to aid the testing of the TMS system These include:

- TmsControl: This simple command line application acts as a client to the TMS system and allows most control and data access functions to be carried out.
- TmsControlGui: This GUI application performs the functions of the tmsControl application but with with a GUI interface.
- TmsSigGen: This program allows the generation of simple test signals for the TMS system. The test signals can be written to the PUPE's data test memory or can be used with the TMS Signal generator's AWG board for the generation of real analogue/digital signals.
- TmsStateGen: This program creates some basic State/Phase tables to match the signals generated with the TmsSigGen program.
- TmsTestData: This command line program, allows the complete system to be set into simulated timing and data mode and then data integrity and data performance tests to be performed.

These test applications are available on the TMS system controller but can be ported to other systems.

2. TmsControl test program

The TmsControl application acts as a client to the TMS system and allows most control and data access functions to be carried out. It is a simple command line application that takes the following command line parameters:

Usage: tmsControl [options] <hostname>

Option	*Description*
-ring <ring>	The ring to connect to. Ring should be a value from 1 to 4.
-help	Help on command line parameters
-debug 0x1122	Set Debug mask
-version	Display Version numbers
-events	Show all events
-outFile <fileName>	Send output to given file. Default to data.txt
-kst	Format output file for the kst plotting package.
-init	Initialise TMS system
-test	Test TMS system
-status	Get Status of TMS system
-statistics	Get Statistics from TMS system
-configure <file>	Configure taking channel information from the given file
-getConfiguration	Gets and displays the current configuration
-setControl <file>	Adds or modifies an entry in the TMS State/Phase table database from the given file
-delControl <Spec>	Deletes an entry from the TMS State/Phase table database. Spec: CycleType,Channel
-cycleType <type>	Sets the CycleType for the next cycle
-getCycleInfo	Read information on current cycle
-getData <DataSpec>	Read data from the system using the DataSpec given. DataSpec: <chan,period,startTime,orbit,bunch,numSamples,function>
--setSimulation <timing,data,setNextCycle>	Sets the internal simulation modes. Boolean values.
--getSimulation	Gets the internal simulation modes
-perfPu	Perform a getData performance test direct to the first PU
-perf	Perform a getData performance test

-channel <n>	Channel to use
-captureDiagnostics <CapSpec>	Capture Diagnostics data. CapSpec: <chan,source,clock,startTime,postTriggerDelay,trigAnd,trigStore,trigData,trigMask>
-setTestData <chan,filename>	Set the test input data for the given channel
-setPupeConfig <chan,iAdcClk,iTiming>	Set the PUPE test configuration for the given channel. iAdcClk: Uses the internal ADC 125MHz clock rather than one synchronised to the external 10MHz Clock iTiming: Uses internal hardware/software timing rather than the external timing signals

3. TmsControlGui test program

This GUI application performs the functions of the **tmsControl** application but with with a GUI interface. This is described in the **TmsControlGui** manual. As well as providing control of the TMS system it has the ability to display the data and diagnostics in graphical form.

4. TmsSigGen test program

This program allows the generation of simple test signals for the TMS system. The test signals can be written to the PUPE's data test memory or can be used with the TMS Signal generator AWG board for the generation of real analogue/digital signals.

This program is documented in the manual **TmsSigGen**.

5. TmsStateGen test program

This program creates basic State/Phase table ASCII files to match the signals generated with the **TmsSigGen** program. The program has a simple command line interface:

tmsStateGen [options] <fileName>

Generates a Tms State/Phase Table ASCII file for a particular BEAM type. The resulting information is written to the file named “fileName”.

Option	***Description***
-t <type>	Specifies the signal type (Beam1 etc)
-T	List all signal types supported
-p	Produce Phase Table ASCII dump file. This is useful to plot the Phase table entries.
-fd <n>	Set pllInitialFrequencyDelay to n ms. Default is 0.
PhaseDelay <n>	Add the given value to all PhaseDelay parameters

PhaseDelayAll <n>	Set the phase delay for all channels to this value
-fref <n>	Set FREF to n Hz. Default is 437000 Hz.
-name <n>	Set the CycleType to the given name. By default the signal type will be used.
-info <n>	Prepend the given string to the info parameter in the file.

6. Testing Overview

The TMS system has three analogue input channels, 8 digital timing inputs and one software cycle number/cycle type event call. In order for the TMS system to work all of these signals must pre present or synthesised by some means. The system provides the ability to simulate some of these signals for test purposes. The following test scenarios give examples of this.

6.1. Testing using on-board data source and software timing

This test method requires no external signals. A test signal is loaded into the test data SDRAM of a PUPE board. The test signal provides the FREF, Sigma, DeltaX and DeltaY input signals. The TMS software simulates the digital timing signals and software next cycle number/type event.

The FREF/Sigma/DeltaX/DeltaY signals data is supplied in a binary file using the format as defined in the PupeFpgaFirmware manual. It is possible to us the TmsSigGen program to generate test signal files and there is a small set of test data files in the /usr/tms/data directory simulating a simple 4 bunch beam with a harmonic number of 8. The *setTestData*() API call is used to load this test data into a PUPE and configure a PUPE channel to use the data as input. Note that only one set of test data can be used on a PUPE board, but this can be supplied to any set of the three PUPE channels. The **TmsControl** and **TmsControlGui** applications can be used to load the test data and set this mode of operation.

The TMS software can generate the timing signals for the TMS system. The *setPupeConfig*() API call can be used to set a module controller to generate software timing signals for all of the PUPE boards in its rack. Each individual PUPE channel can use either the hardware or software generated timing signals.

The main **TmsServer** program has a configuration option, **SimulateNextCycle**, which configures it to automatically call the *setNextCycle*() API call with an incrementing cycle number. This call is also used to synchronise the software timing signal generation of the module controllers together. This allows the full TMS system to be set to use test data and software timing signals without any external signals being used.

The **TmsControl**, **TmsControlGui** and **TmsTestData** applications can be used to set up the above options and examine the system during operation.

The **TmsControlGui** program has the TAB “Pupe Simulation”. If you tick the Timing, SimData and SimNextCycle boxes in the “Overall Simulation” area and click the “Apply” button, this will configure the system to use internal software generated timing for testing.

6.2. Testing using TMS Test Signal Generator

In order to more fully simulate the data and timing signals of a PS machine, we have developed a TMS test signal generator. This uses an arbitrary waveform generator (AWG) card to generate the main TMS Analogue and timing signals. This card produces the Sigma, DeltaX, DeltaY, FREF, SYS_CLOCK, CYCLE_START, INJECTION and HCHANGE signals. The CYCLE_STOP signal is simulated in

software. The setPupeConfig() call is used to enable this mode of operation.

The main **TmsServer** program has a configuration option, **SimulateNextCycle**, which configures it to automatically call the setNextCycle() API call with an incrementing cycle number.

Apart from the software simulating the CYCLE_STOP and setNextCycle() API calls, the system operates as it would if connected to a PS machine. The **TmsSigGen** program is used to set the AWG card to produce an appropriate test waveform. This can consist of a simulated PS machine cycle containing a number of Harmonic changes.

The **TmsControl** and **TmsControlGui** applications can be used to set up the above options and examine the system during operation.

6.3. Using the PS Machine

The full set of analogue data and digital timing signals can be supplied from the PS machine itself. The main TMS analogue and timing signals required include: Sigma, FREF, SYS_CLOCK,CYCLE_START, CYCLE_STOP, INJECTION and HCHANGE signals. The CYCLE_STOP signal can be simulated in software if required. The setPupeConfig() call is used to enable this mode of operation or the "SimulateTiming" parameter in the /etc/tmsPuServer.conf file of a module controller can be set to "0x04".

The main **TmsServer** program has a configuration option, **SimulateNextCycle**, which configures it to automatically call the setNextCycle() API call with an incrementing cycle number.

The **TmsControl** and **TmsControlGui** applications can be used to set up the above options and examine the system during operation.

7. Using the TmsTestData application

The TmsTestData application is a simple command line application that can be used to set the TMS system into simulated timing and data mode as well as perform data integrity and performance tests. Its operation is controlled using command line arguments. The following command line arguments are supported:

-help	Help on command line parameters
-simdata	Set the system to use simulated timing signals and test data. This just sets the master PUPE boards to use software driven timing signals. All the other boards run of the timing bus as normal. All boards are set to use the beam3-437000-8.psd simulated data signal. The TMS Server is set to use the SimBeam3 cycle type.
-info	Prints information on the processing cycle.
-numSamples <n>	The total number of samples to read in a singe getData() call.
-test <testName>	The Test to be performed: single - Fetch all bunches from a single channel. all - Fetch all bunches all channels.
-check	Check the data for validity.
-cont	Continuous.

-quiet	Only print errors.
-channel <n>	The channel number to use with the “single” test.

8. Accessing the TMS nodes

The TMS system has a System Controller (SC) and a number of Module Controllers (MC). Each of these computer systems run the Linux operating system and can be accessed by loging into them over the network or by direct monitor/keyboard access.

The System Controller has a fully fledged Linux operating system installed. The system can be accessed in one of a few main ways:

- Secure shell access using the SSH protocol. As well as a command line interface, X-Windows GUI applications can be run over this interface. See the maintenance manual for default user id's and passwords.
- Console access. By attaching a monitor/keyboard and mouse the system can be accessed directly. See the maintenance manual for default user id's and passwords.

The Module Controller has a minimal Linux system installed and is also on a private network with the SC. The Module Controllers can be accessed in one of a few ways:

- Telnet access using the TELNET protocol from the system controller. This allows you to log into the SC using the SSH protocol and then telnet into an individual module controller. The module controllers host names will be tmsmod1.tmsnet, tmsmod2.tmsnet etc. You can login as the user root with no password.
- Console access. By attaching a monitor/keyboard and mouse the system can be accessed directly.

The module controller has a few basic Linux command line utilities and is based on the Busybox system.

9. Cycle Parameters State/Phase table modifications/Additions

Modifications to and/or adding new PUPE state/phase table configurations can be achieved in one of a few ways.

- The ***setControlInfo***() API call can be used to overwrite or add a state/phase tables to the system.
- The **tmsControl** or **tmsControlGui** application can be used to overwrite or add a state/phase tables from an ASCII file.
- The **tmsControlGui** application can be used to create or edit the state/phase tables directly.
- You can copy the state/phase table file to the /usr/tms/stateTable directory on the system controller and call the *init*() API call.

To update the State/Phase table using the **tmsControlGui** application perform the following:

1. Login to the TMS system as the user “tms”.
2. Run the **tmsControlGui** application.
3. Click on the “Cycle Params” tab.
4. Load a set of “Cycle Params” data using the “Load from TMS”, Cycle Type list and the “Load from TMS” button.

5. You can edit the basic values in the entry boxes on this screen.
6. The updated “Cycle Params” can be uploaded to the TMS system using the “Upload” button, or saved to a file using the “Save to file” button.

Note that the Cycle Parameters will be stored on the TMS system under the CycleType name as given in the CycleParams file. These parameters will be loaded into the FPGA for the next TMS processing cycle that is of this cycle type. The TMS's current cycle type can be changed using the “Cycle Type” entry box in the “Pupe Simulation” dialog.

10. Loading a new FPGA bit-file

The FPGA bit file will normally be provided as an RPM software package. To install a new FPGA bit file copy the RPM file onto the TMS system and run the following command:

“rpm -U <tms-fpga*.rpm>”

where “<tms-fpga*.rpm>” should be the name of the RPM package file.

To get the TMS system to use this bitfile you should initialise the TMS system. This can be done using the “Initialise TMS” button in the **tmsControlGui** applications “Pupe Simulation” TAB. You can also use the “tmsRestart” shell command to restart the TMS system completely.

You can also install the FPGA bit file manually. To do this it should be copied to the /usr/tms/fpga directory and the symbolic link “tms-fpga.bit” should point to the file. This will be loaded when the TMS system is initialised.

11. Script Test System

The script based test system uses the **tmsControl** application to perform some basic system tests.