ALPHA DATA
BEAM

CERN Trajectory Measurement System

TmsServer

Project	CERN-TMS
Date	2013-02-01
Reference	Cern-tms/TmsServer
Version	2.0.0
Author	Dr Terry Barnaby

Table of Contents

1. Introduction

This document covers the operation of the TmsServer program that runs on the System controller and provides overall control of the TMS System. The TmsServer program runs with real-time process priorities and is normally started at boot time by a system start-up script.

2. Starting the TmsServer program

Normally the TmsServer program is started on the System Controller using the “tmsServer” init file in the directory /etc/init.d. It can be started using the command “service tmsServer start” and stopped with the command “service tmsServer stop”.

The TmsServer program also provides the ability to start the program from the command line with various debug options set. When started from the command line the following command line options are provided:

-c <configurationFile>	Specifies the configuration file to use. This can be used for starting separate servers for each ring in a multi-ring system.
-f	Starts the program in the foreground. By default the program is started as a background task.
-d 0x03	Sets the debug options. The debug options are a bit mask of possible debug enables. These are listed below.

Debug bit values

0x000001	Standard. Provides basic debug print out.
0x000002	Commands. Prints out all of the commands received.
0x000004	Events. Displays all events received.
0x000100	Resource. Resource usage.
0x001000	Threads. Threads usage.

3. Configuration

The TmsServer has a simple ASCII configuration file. By default the file /etc/tmsServer1.conf is used, but the "-c <configFile>" command line option can be given to use a different file.This file can be edited using a conventional text editor. Its contents are as follows:

Parameter	*Default*	*Description*
TmsServer:	ttmssc.tmsnet	The BOAP Servers host name. Normally the System Controllers host name on the TmsNet.
Ring:	1	The ring this server is for. This can be in the range 1-4
Master:	1	A boolean t indicate this server is a master server for the system. This means it will send the Statetable information and the setNextCycle event to all the TmsPuServer processes.
SptDir:	/usr/tms/stateTables	The directory where there State/Phase table library is stored.
SimulateData:	0	If set to 1, the TmsServer will simulate data capture internally. This is useful for debug whith using any TmsPuServer servers and thus any PUPE engine boards.
SimulateNextCycle:	0	If set to 1 the TmsServer will call the setNextCycle() call on each CYCLE_STOP event.
DefaultCycleType:	Beam3	This is the type of cycle that will be set on start-up. It defines which State/Phase tables will be loaded initially.
AdcSysclkSync:	0	Sets the ADC clock to be synchronised with the SYSCLK timing clock
PuServer1:	1	The is the number of the first TmsPuServer
PuServer2:	2	The is the number of the second TmsPuServer
PuServer3:	3	The is the number of the third TmsPuServer
PuServer4:	4	The is the number of the forth TmsPuServer
PickUp*:	1,1,1	This is the logical to physical pick-up table configuration for the ring. It is overwritten on configure() API calls. The values are: ModuleNum, PupeNum and PupeChan.

4. Overall Operation

On startup the TmsServer will initialise its internal state and then publish its BOAP API to the Boap server running on the system who's host name is given in the configuration files "TmsServer" parameter. The BOAP API names have the ring number added to their names so that the TmsServer for each ring can be contacted separately. It will then attempt to connect to and initialise all of the Module Controllers TmsPuServers as defined in the configuration files PuServer? parameters.

It will inform each of the TmsPuServers, it finds, of its local TmsEvent object so that they can send events to the TmsServer.

The TmsServer will now listen for API requests from clients and services them as required. It will also

respond to events from the TmsPuServer programs as needed.

The TmsPuServer programs can detach and re-attach themselves to the TmsServer as required. This allows the system to power up in any sequence and also allows the Module Controllers to be re-booted if required. Any access to a PU channel that is not currently available, will return an appropriate error.

On a multi-ring system multiple TmsServer processes are started, each with its own configuration file named using the "-c <configFile>" option. The configuration files will define the ring that the server is for and all of the PU channels that are part of the ring. The TmsServer will configure the TmsPuServers with the list of channels for its ring. If the "Master:" parameter is set to 1 in the configuration file then this TmsServer will send the StateTable information to all of the rings on initialisation. The master TmsServer is also responsible for sending the setNextCycle event to all of the TmsPuServers servers.

5. Errors

The TmsServer will handle errors in one of two ways.

- The individual API calls will return an errors as appropriate.
- The TmsServer will send an error event to the client.

6. Logging

While in operation all warnings and notices will be written to the systems standard logging daemon. These messages will thus normally appear in the /var/log/messages files.