Clock Tree Tool

This document describes how to install and use Clock Tree Tool (CTT).

Topic **Page**

Trademarks

Windows is a registered trademark of Microsoft.
Java is a trademark of Oracle.
All other trademarks are the property of their respective owners.

1 CTT Overview

The Clock Tree Tool is a Java™ based stand-alone application. This is an interactive clock tree configuration software for the device. The CTT allows the user to:

- Visualize the device clock tree
- Interact with clock tree elements and view the effect on clock-tree configuration registers
- Interact with the clock-tree configuration registers and view the effect on the device clock tree
- View a trace of all the device registers affected by the user interactions with clock tree

The advantage of the tool is that the user can visualize the device clock tree state on power-on reset and then customize the configuration of the clock tree for the specific use-case and identify the device register settings associated to that configuration.

Being an interactive visual tool, the CTT gives the user a global view of the device clock tree architecture and allows determining the exact register settings to obtain the specific configuration.

2 CTT System Requirements

- Java JRE 1.8 or higher (can be downloaded from https://java.com/en/download/).
- Has been tested for Microsoft Windows ® 10.
- The ideal screen resolution is 1920x1080 (4k displays are also supported, however some renderings in the CTT view can appear small due to the high resolution).

3 CTT Running Specifics

The start-up sequence of the CTT consists of reading an entire clock tree description database files. This would normally take from 10 seconds to several minutes depending on the specific features set of the device.

Similarly, the *Refresh View* function (see Section 6.4, *CTT Menu Commands Description*) that updates the *MAIN VIEW* (see Section 6, *CTT Views Description*), covers the entire clock tree of the device and takes as well e from 10 seconds to several minutes depending on the specific features set of the device.

4 CTT Running Linux Requirements

CAUTION

Before running CTT package, Linux user should perform the following steps:

- In the console, go to CTT install folder.
- Run the following command (see (1))

(1)
```
java -
cp jGraphLib/lib/*:<name_of_CTT_package>.jar:. org.ti.clockTreeTool.simulation.ClockTreeTool
```

5 CTT Installation/Uninstallation

5.1 CTT Installation

CAUTION

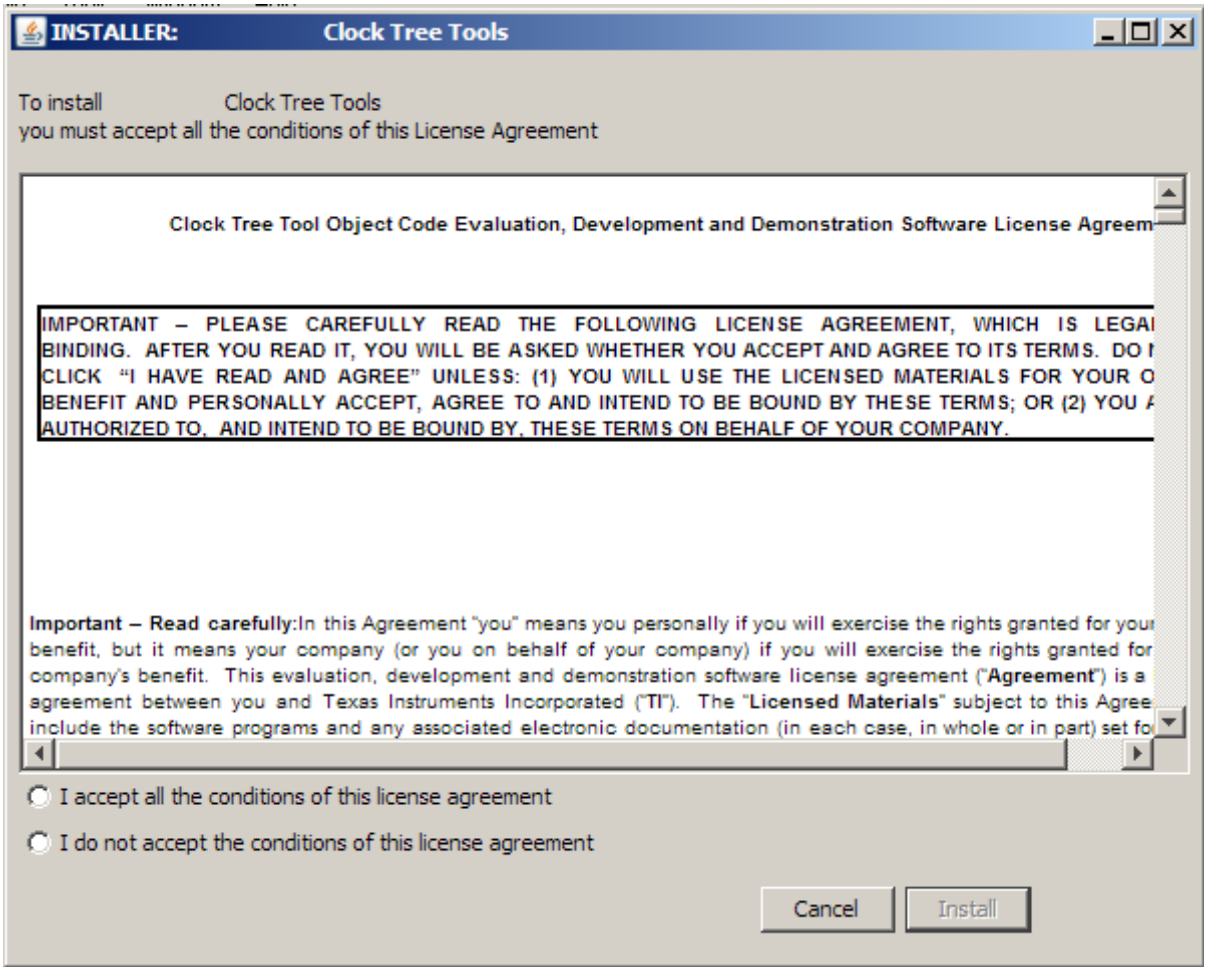
Java™ Runtime Environment, Standard Edition (v1.8 or higher) must be installed before Clock Tree Tool is run.

The CTT installation procedure is composed of several steps described in Section 5.1.1 through Section 5.1.3.

5.1.1 CTT Installation: Step 1

To install the Clock Tree Tool double click (java -jar in terminal for Linux users) on the "Installer-CTT-xxxx" file. The installer will execute and display the *License Agreement Window - View 1* shown in Figure 1. The user must accept the conditions of the license in order to proceed with the installation of the CTT.

Figure 1. CTT Installation: License Agreement Window - View 1



5.1.2 CTT Installation: Step 2

When the conditions of the license agreement are accepted, the *Install* button is enabled, see Figure 2. Click on the *Install* button to proceed to the *Destination Directory Selection Window*, see Figure 3. It allows the user to identify the installation directory of the Clock Tree Tool. Once the directory is selected click the *Select* button to start the installation.

Figure 2. CTT Installation: License Agreement Window - View 2

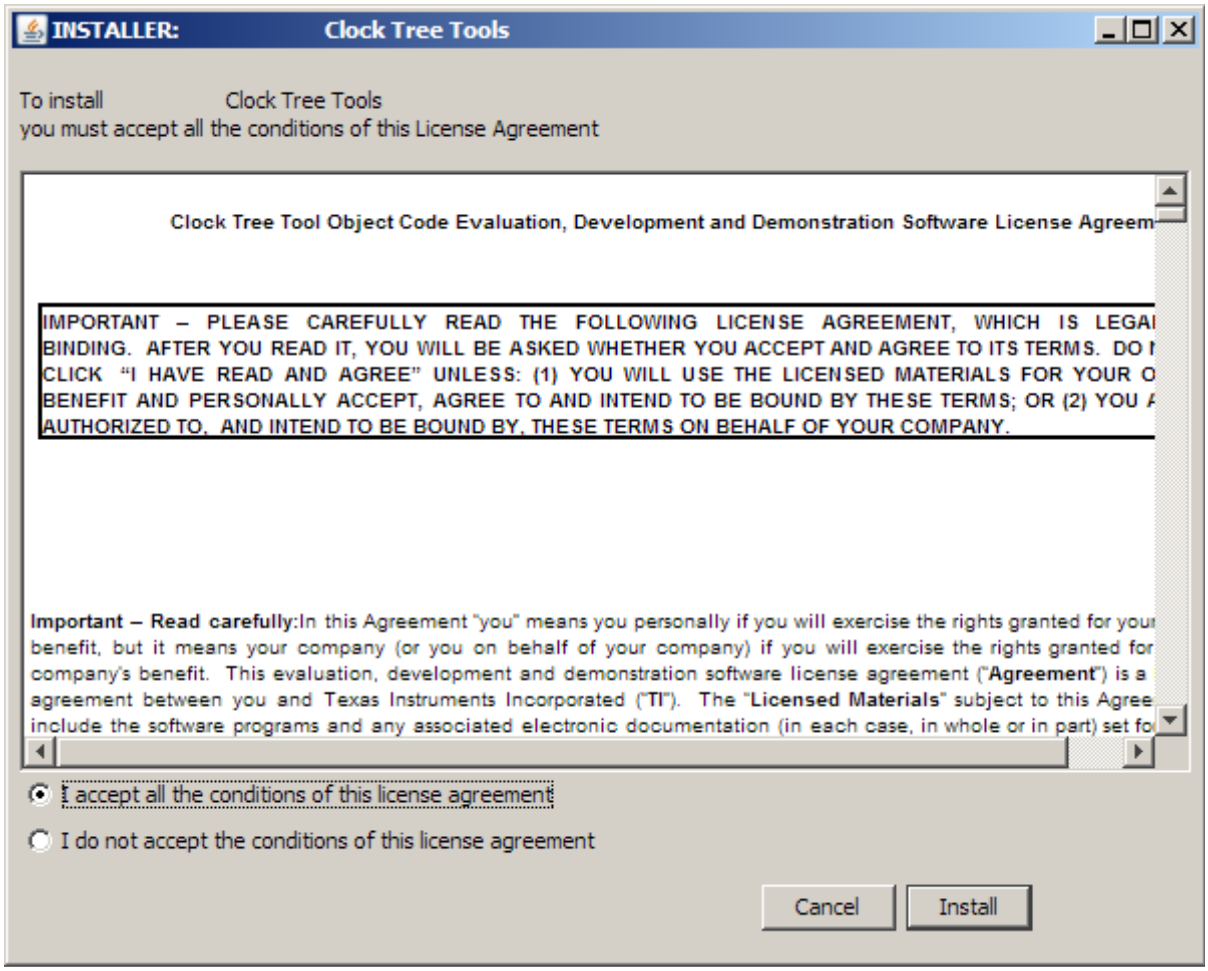
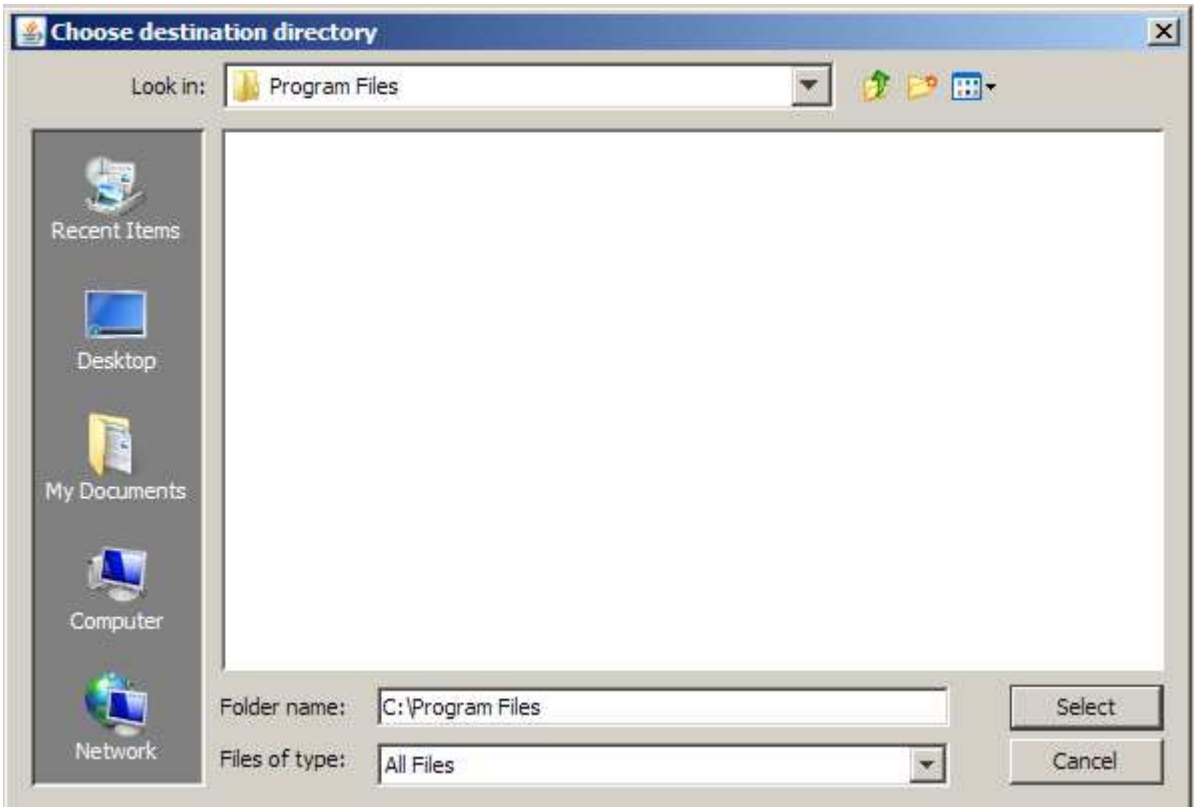


Figure 3. CTT Installation: Destination Directory Selection Window

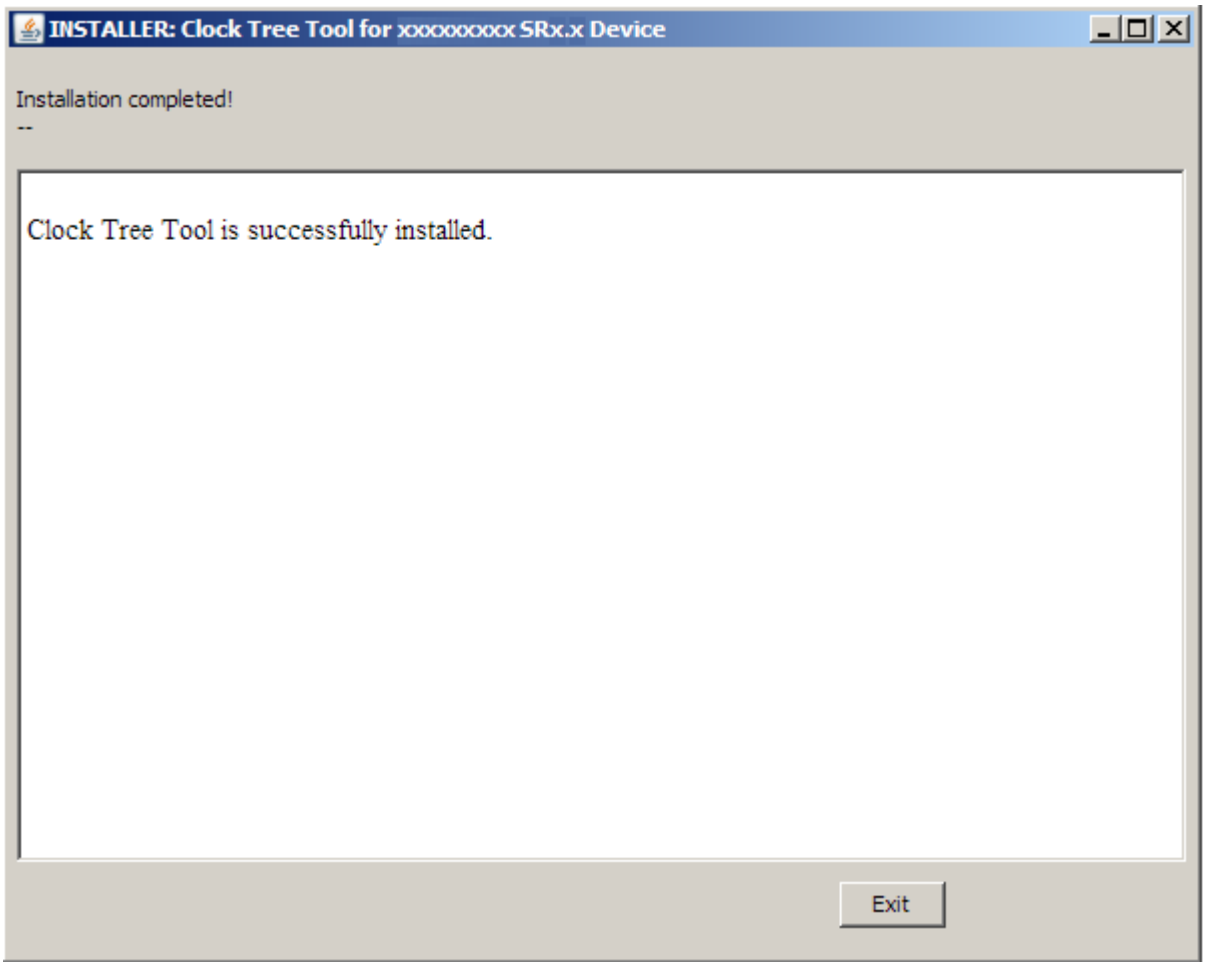


5.1.3 CTT Installation: Step 3

When the installation is finished the message "Installation completed." is displayed, see Figure 4. Click on *OK* button to proceed to the last window - *Exit Window*, see Figure 5. There, click *Exit* button to complete the CTT installation.

Figure 4. CTT Installation: Complete Message Window

Figure 5. CTT Installation: Exit Window



5.2 CTT Uninstalation

The Clock Tree Tool does not leave application traces of its installation in the OS applications, features, and registry. So to uninstall it, simply navigate to the installation folder and delete it.

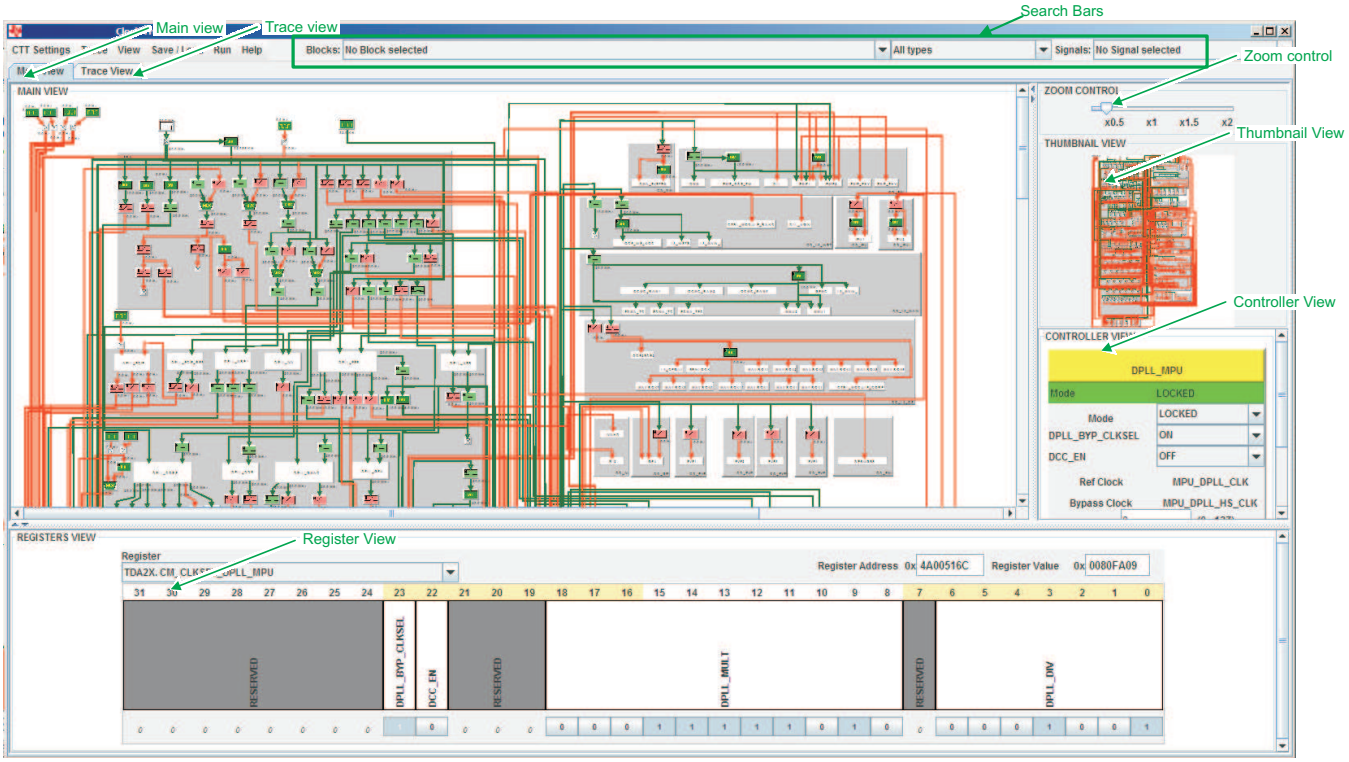
6 CTT Graphical User Interface (GUI) Description

6.1 CTT Views Description

The CTT GUI is composed of 5 sub-views (see Figure 6):

- *MAIN VIEW*
- *THUMBNAIL VIEW*
- *CONTROLLER VIEW*
- *REGISTERS VIEW*
- *Trace View*

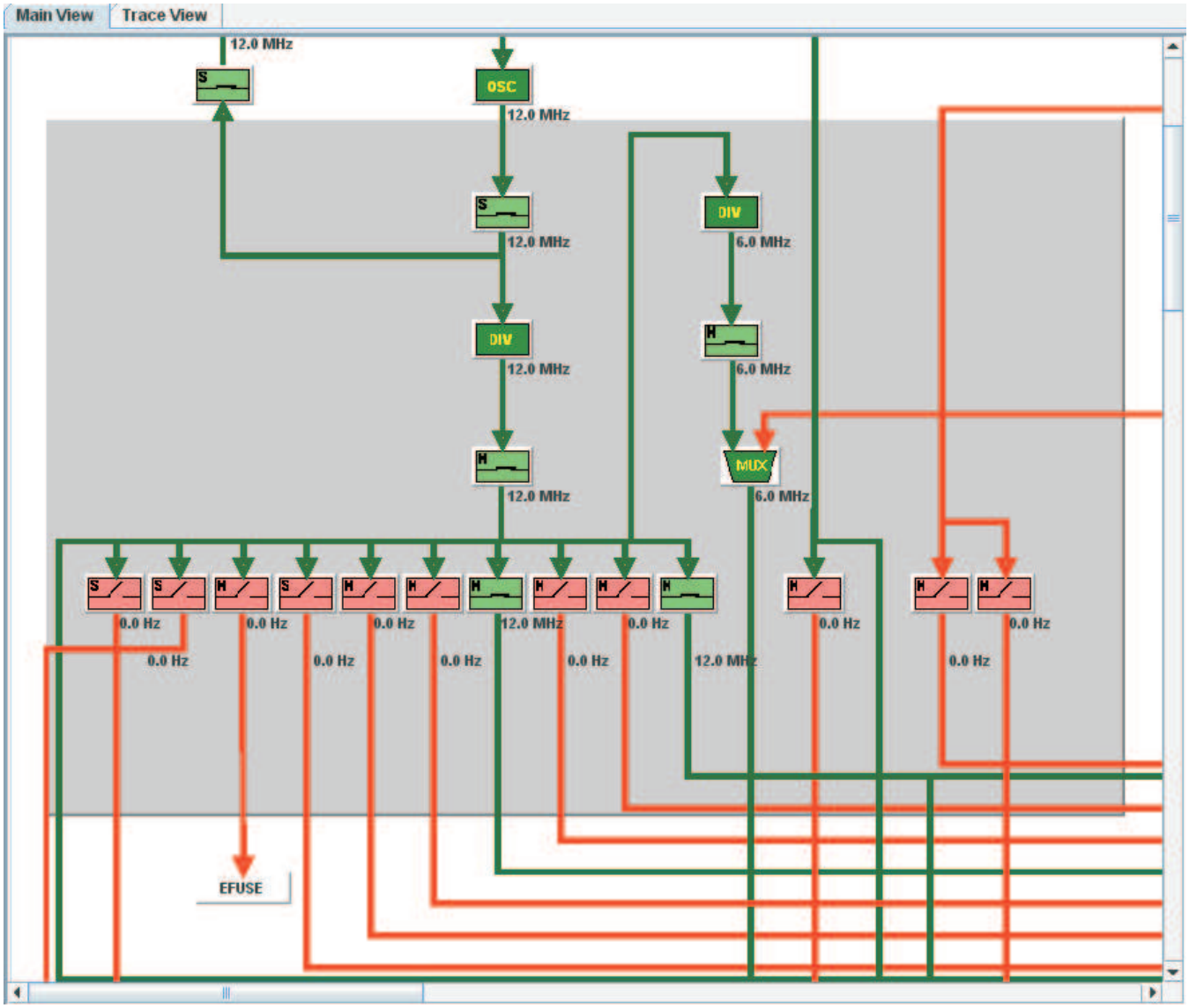
Figure 6. CTT Views



6.1.1 CTT MAIN VIEW

The *MAIN VIEW* (see an example in Figure 7) presents a focused view of the device clock tree particular section.

Figure 7. CTT *MAIN VIEW*



The device clock tree is represented as a tree structure composed of "nodes" or "blocks" (that is, the rectangular elements) and the "links" or "signals" (that is, the arrows). The direction of the signal identifies the source and the destination blocks of the signal. A block may be a source block to multiple blocks and may in turn have multiple source blocks connected to it.

The clock tree has following types of blocks:

- Pin
- Pin-Clock Source
- Crystal
- Clock Source
- Oscillator
- Clock Switch (Hardware/Manual/Automatic)
- Divider
- High-Speed Divider
- Multiplexer

- DPLL
- Module
- Delimits

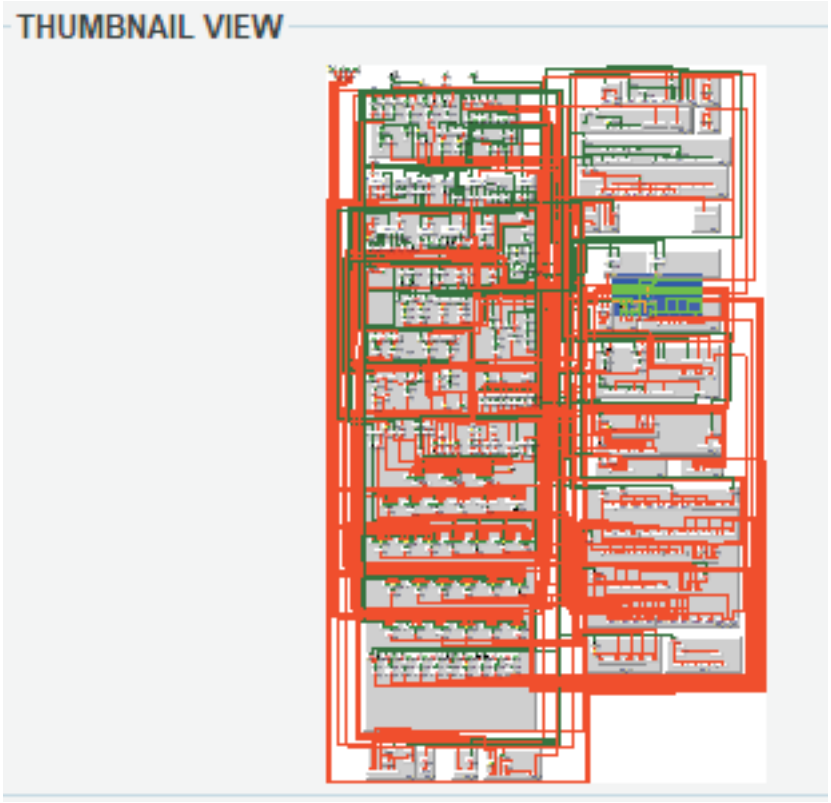
Note: Refer to the device Technical Reference Manual and to Section 7, *CTT Blocks*, for the description of these blocks.

The user can mouse drag or use the scroll bars to move around the view. The view highlights the state of the blocks and the signals visually. For example, the state of a clock switch (Open/Close) is presented by a red open switch or a green close switch symbol. Similarly, the state of a clock signal (Active/Gated) is highlighted by the signal being green or red.

6.1.2 CTT THUMBNAIL VIEW

The *THUMBNAIL VIEW* (see an example in Figure 8) provides a global view of the device clock tree. The *THUMBNAIL VIEW* also highlights the region of the clock tree visible in the *MAIN VIEW* by a bounding rectangle. As the slide bars of the *MAIN VIEW* are displaced the bounding rectangle in the *THUMBNAIL VIEW* also moves accordingly.

Figure 8. CTT *THUMBNAIL VIEW*



6.1.3 CTT CONTROLLER VIEW

The *CONTROLLER VIEW* (see Figure 9 and Figure 10 for examplary views) highlights a signal or a block of the clock tree. The user selects (that is, clicks on) the signal/block in the *MAIN VIEW* and it is highlighted in the *CONTROLLER VIEW*. If a signal is selected, its current frequency is presented (see Figure 9), whereas, if a block is selected, depending on the block type its parameters are presented (see Figure 10).

Figure 9. CTT *CONTROLLER VIEW* of a Signal

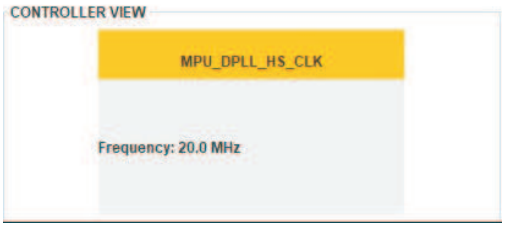
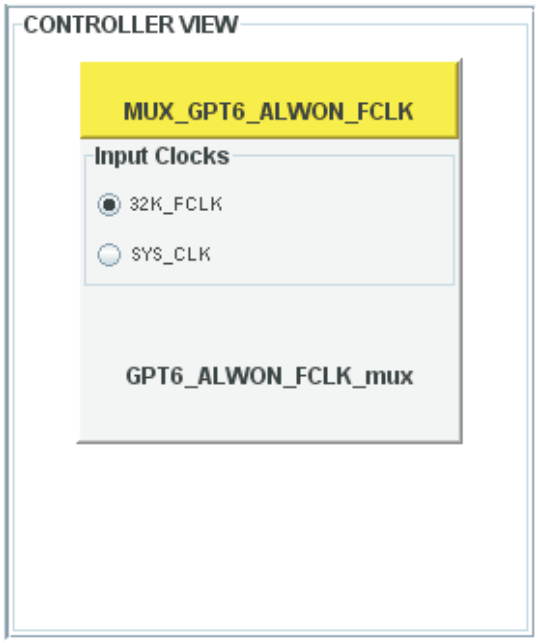


Figure 10. CTT *CONTROLLER VIEW* of a Block



6.1.4 CTT REGISTERS VIEW

The *REGISTERS VIEW* is composed of a *Register* or *Select Register* list box, on the upper left-hand side; *Register Address* and *Register Value* or *Address : Value*, on the upper right-hand side; and *Register Bits* view, on the lower side of the *REGISTERS VIEW*.

There are two types of *REGISTERS VIEW* - new-style format, see Figure 11, and old-style format, see Figure 12.

The *Register* or *Select Register* list box shows the currently selected register.

The *Register Address* and *Register Value* or *Address : Value* of the *REGISTERS VIEW* presents the address and the current hexadecimal value of the register. In the new-style format, the user may type in a different address to select a new register, or can type in new register value to reconfigure the register bitfields.

The *Register Bits* view lists all the bits/bitfields of the selected register (for example, 0 to 31 bits for a 32 bits register). Each bit is identified by the bit number (0 for the Last-Significant Bit (LSB)). In the old-style format, below the bit number is the current value of the bit (1/0). In the new-style format, between the current value of the bit/bitfield is its name.

A toggle button below the bit number of the user configurable (that is read/write) bits allows the user to toggle the bit value. Pressing the button sets the bit value to 1 and in the released state the bit value is 0. There is no button associated to the RESERVED bits of the register (that is, the user cannot modify the states of these bits).

When the user selects a register in *Register* or *Select Register* list box, its contents (that is, bits and value) are highlighted in the *Register Bits* view; and the *Register Address* and *Register Value* or *Address : Value* are also displayed.
When the user changes a parameter of a block in the *CONTROLLER VIEW*, the associated bit/bitfield is updated in the register and the *REGISTERS VIEW* displays the affected register.
When the value of a bit/bitfield of a register changes in the *REGISTERS VIEW*, the *Trace View* captures this change also.

NOTE: When the user changes a parameter of a block which affects bitfields of more than one register, the *REGISTERS VIEW* only shows the last register updated. The *Trace View* shows the complete list of registers affected by this change.

Figure 11. CTT *REGISTERS VIEW* - New-Style Format

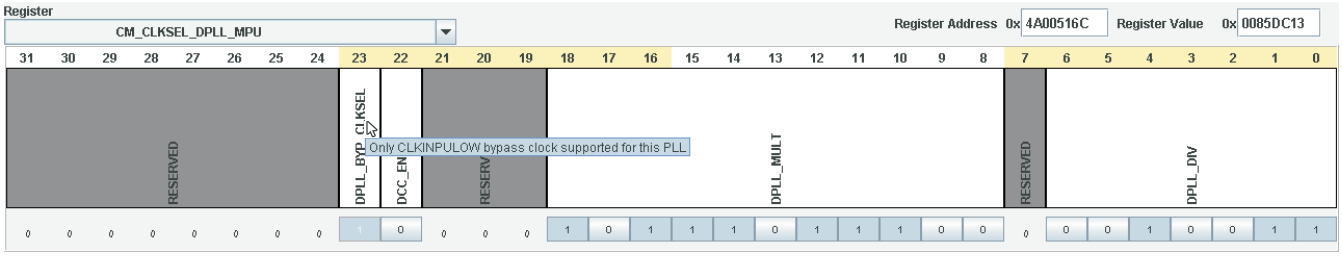
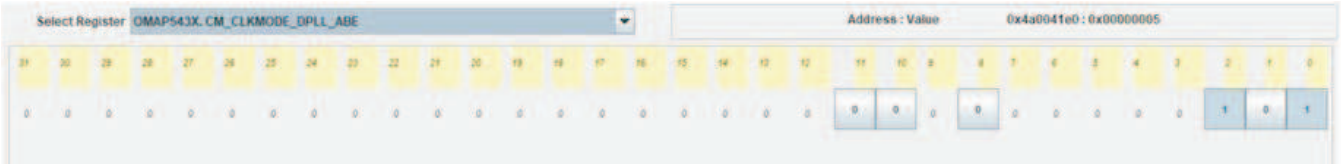


Figure 12. CTT *REGISTERS VIEW* - Old-Style Format



In the new-style format, when user positions the pointer on the name of the bit/bitfield a pop-up displays the name of the corresponding bitfield.

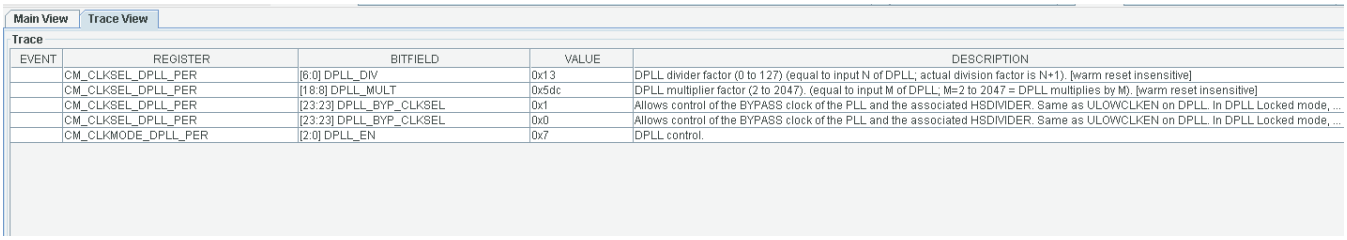
In the old-style format, when user positions the pointer on the number of a register bit a pop-up displays the name of the corresponding bitfield.

6.1.5 CTT Trace View

The *Trace View* (see Figure 13) allows the user keep track on register changes made anywhere from the GUI views.

NOTE: User can reset/clean the trace event log from *Trace->Reset* menu option.

Figure 13. CTT *Trace View*

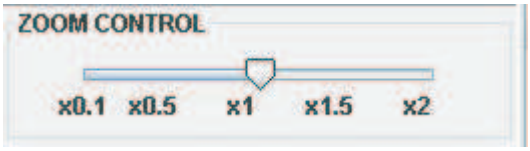


6.2 CTT ZOOM CONTROL

The *ZOOM CONTROL* (see Figure 14) allows the user to change the zoom level of the *MAIN VIEW*. By default the zoom level is set to x1. The user can zoom in by shifting the slider to the right (towards x2) and zoom-out by shifting the slider to the left (towards x0.1).

NOTE: A zoom in/out on mouse scroll and drag to move functionality is also available.

Figure 14. CTT *ZOOM CONTROL*

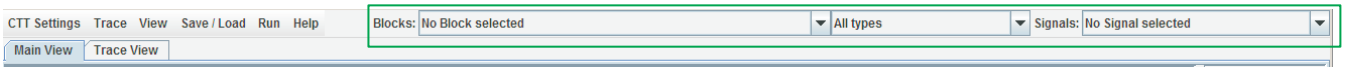


6.3 CTT Search Bars

The CTT *Search Bars* (see Figure 15) allows the user to navigate directly to the desired block or signal within the *MAIN VIEW*. There are 3 bars. The first two are for searching and navigating to a particular block. The third one is for searching signals.

After selecting the desired block or signal, the *MAIN VIEW* will automatically scroll and highlight the selected (from the bars) block or signal.

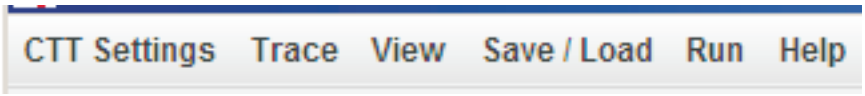
Figure 15. CTT *Search Bars*



6.4 CTT Menu Commands Description

The CTT menu has the following commands (see Figure 16):

Figure 16. CTT Menu



1. *CTT Settings*
   (1.1) *Power-on Reset*
2. *Trace*
   (2.1) *Reset*
3. View
   (3.1) *Hide Others*
   (3.2) *Display All*
   (3.3) *Hide Frequency*
   (3.4) *Display Frequency*
   (3.5) *Refresh View*
   (3.6) *Print View*
4. *Save / Load*
   (4.1) *Save Registers*
   (4.2) *Save Registers (CTT only)*
   (4.3) *Load Registers*
   (4.4) *Save Source Clocks*
   (4.5) *Load Source Clocks*
5. *Run* (not applicable for all CTT packages)
   (5.1) *Frequency Analyzer*

6. *Help*
   (6.1) *About Clock Tree Tool*
   (6.2) *User Manual*
   (6.3) *SRAS License Agreement*

6.4.1 CTT Settings

(1.1) *Power-on Reset*: Triggers a power-on reset for all the configuration registers. All the registers are set to their reset values. As a result, the state of the clock tree is updated and reflects the state after power-on reset. Note: When the CTT starts, the power-on reset is automatically triggered. Hence, the initial clock tree state is that of the device after power-on reset.

6.4.2 CTT Trace

(2.1) *Reset*: Resets (clears) the *Trace View* table.

6.4.3 CTT View

(3.1) *Hide Others*: When a clock signal is selected in the *MAIN VIEW* this command hides all the clocks not associated to the selected clock. A clock is considered associated to another clock if it is directly/indirectly a parent/child of the clock.

(3.2) *Display All*: This command is used to redisplay the entire clock tree from a partial view (as a result of the *Hide Others* command).

(3.3) *Hide Frequency*: This command hides the frequency value of the clock signals in the *MAIN VIEW*.

(3.4) *Display Frequency*: This command displays the frequency values of the clock signals in the *MAIN VIEW*.

(3.5) *Refresh View*: This command refreshes the *MAIN VIEW* representation of the clock tree. It is used if the clock tree representation is not correct and the view needs to be refreshed.

(3.6) *Print View*: When selected, the print option generated an image and saves it in the CTT install directory. For a particular reason a user may want to print the tree onto an image. This image could be helpful if one needs to have it on paper, or just look at it without the need to load the CTT. This may also help when a user want to create a CTT configuration and print it. Then create another one and print it. This way the 2 or more print stamps can be compared and analyzed.

6.4.4 CTT Save / Load

6.4.4.1 CTT Save / Load Registers

The *Save / Load Registers* menus allow the user to either configure the clock-tree configuration registers (used in the CTT) to specific settings given in a file or to write the current values of the registers of the CTT to a file. This way the configuration can be saved, reused, tweaked and shared between team for development and debug.

There are two options for saving registers:

- (4.1) *Save Registers*: Save all registers (Clocking, Control Module, etc.), included in CTT.
- (4.2) *Save Registers (CTT only)*: Save only registers used by CTT.

CAUTION

When using (4.1) *Save Registers* option, be aware that the time to load the .rd1/.rd2 file is considerably long compared to the file generated using (4.2) *Save Registers (CTT only)* option.

There is one option for loading registers:

- (4.3) *Load Registers*: Load specific configuration of the registers from a previously saved file from a

CTT configuration or read-in registers from actual hardware board for debug/reference purposes. In order to do a read-in of registers from actual hardware board the user can use the GEL script in Code Composer Studio, CMM script in Lauterbach, or OMAPconf register print log.

How to perform a read-in of registers from actual hardware board:

The scenario would be to have a running hardware connected and using one of the methods described above, print out a known CTT register dump format. A known format would be:
DeviceName XXXXX_SRX.X
<Register Address> <Register Value>

Then, save the result with the above register format to a **.rd1* file. Load the file into the CTT and the GUI will display the exact hardware clocking configuration at the time the registers were exported.

The other way, to load a configuration into hardware from the CTT, is also possible. User can configure needed register settings from GUI, save registers to a **.rd1* file (go to *Save / Load -> Save Registers* menu or to *Save / Load -> Save Registers (CTT only)* menu) and use this file to load configuration into the connected hardware via the environment used.

6.4.4.1.1 Code Composer Studio (CCS) Memory Dump

The following sequence have to be followed for a **memory dump in CCS**, see Figure 17:

Step 1: In Code Composer Studio load the appropriate target configuration and run it.
Note: For more information on Code Composer Studio set up and usability refer to the Code Composer Studio Help.

Step 2: Wait while the connection to the target is established and the target is initialized.

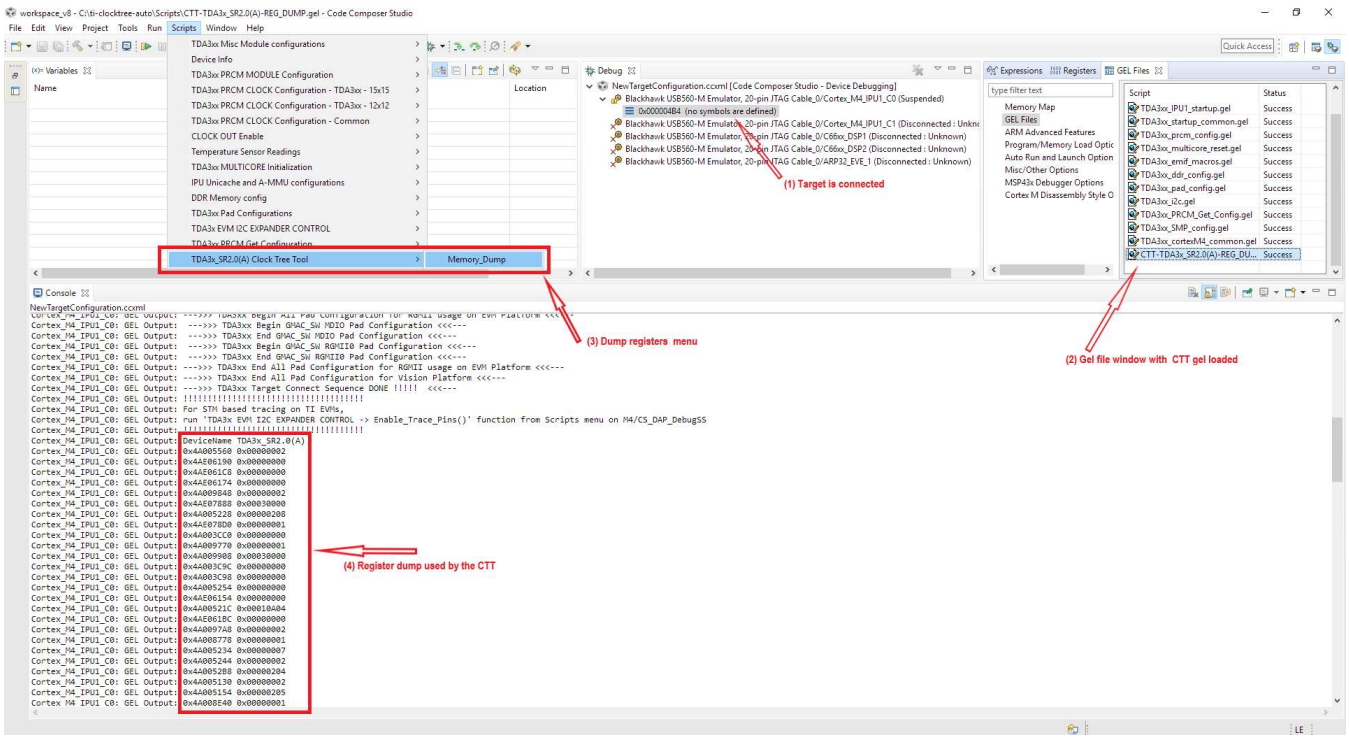
Step 3: Then from the CCS menu go to *Tools -> Gel Files*. Right click in the window that just opened and load CTT-<Device_Name>_<Device_SR>-REG_DUMP.gel script. A menu will appear called *Scripts*.

Step 4: Click on *Scripts -> <Device_Name>_<Device_SR> Clock Tree Tool -> Memory_Dump* .

Step 5: During execution, the GEL script prints the needed CTT registers with values in the console.

NOTE: A set of gel files are provided in the CTT installation folder. See <*CTT isntall dir*>*Scripts*\.

Figure 17. CCS View for Memory Dump



Creating an .rd1 or .rd2 files (CTT compatible files). Note that the following example presents creating an *.rd1* file. Creating *.rd2* file is similar.

1. Copy the output from *Step 5* above into a text file - copy everything from “DeviceName” to the end of the data that the script outputs in the console.
2. In a text file paste all copied data.
   - It should end up with something like:
     CortexA9_0: GEL Output: DeviceName <Device_Name>_<Device_SR>
     CortexA9_0: GEL Output: 0x4A004100 0x00000110
     CortexA9_0: GEL Output: 0x4A004108 0x00000000
     CortexA9_0: GEL Output: 0x4A004110 0x00000000
   - Delete any prefixes (in this example the prefix is “CortexA9_0: GEL Output: ”)
     DeviceName <Device_Name>_<Device_SR>
     0x4A004100 0x00000110
     0x4A004108 0x00000000
     0x4A004110 0x00000000
3. Save the text file as *<filename>.rd1*.
4. Load the created *.rd1* file in the CTT (go to *Save / Load -> Load Registers* menu). The tool will now display a snapshot of the entire system clock configuration of the customer board at the time the gel file was executed.

6.4.4.1.2 OMAPconf Memory Dump under Linux

A console have to be opened wherever is convenient to the user. After the connection to the target is established and the target is initialized. Run the following command:

```
root@am57xx-evm:~# omapconf export ctt am57xx-clocks.rd1
OMAPCONF (rev v1.73 built Tue Sep 26 18:56:57 EDT 2017)

HW Platform:
  Generic DRA72X (Flattened Device Tree)
  DRA72X ES2.0 GP Device (STANDARD performance (1.0GHz))
  TPS65917  ES2.2
```

```
SW Build Details:
  Build:
    Version:  _____                    _____           _         _
  Kernel:
    Version: 4.9.41-rt23-gc038d21a22
    Author: gtbldadm@ubuntu-16
    Toolchain: gcc version 6.2.1 20161016 (Linaro GCC 6.2-2016.11)
    Type: #2 SMP PREEMPT RT
    Date: Tue Sep 26 19:03:11 EDT 2017

Output written to file './am57xx-clocks.rd1'
```

The last line will print a CTT compatable file.

NOTE: A gel and a cmm script files can be found in *<CTT-install-dir>/Scripts/*. For OMAPconf PRCM register dump refer to OMAPconf user guide https://github.com/omapconf/omapconf/wiki.

OMAPconf currently supports TI DRA75x_DRA74x, DRA72x, TDA2x, TDA2Ex, AM572x, AM571x, AM335x, AM437x, OMAP44xx and OMAP54xx devices.

6.4.4.2 CTT Save / Load Source Clocks

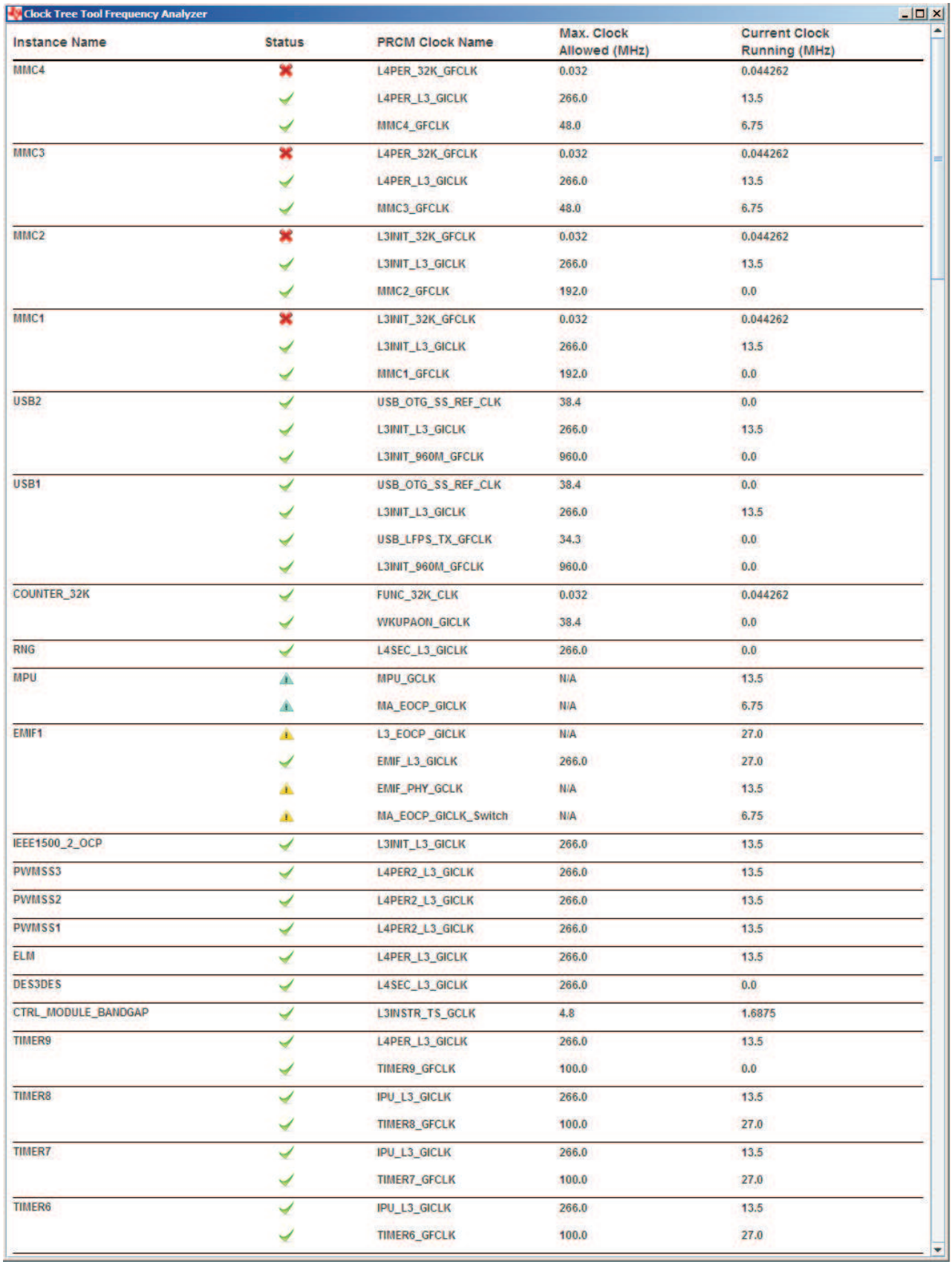
(4.4) *Save Source Clock* and (4.5) *Load Source Clock*: In a given scenario, the user would be using a particular hardware board with different main input source clocks. These source clocks are not directly tied to a register configuration. Therefore, when configured from CTT GUI these source clocks must be saved to be loaded later. For details on clock source configuration from GUI, see Section 7.4.

6.4.5 CTT *Run*

NOTE: CTT *Run* menu is not applicable for all CTT packages.

(5.1) *Frequency Analyzer*: compares the Maximum Supported Frequency table listed in the device-specific Data Manual (DM) versus live CTT configurations and displays a friendly GUI log with proper *error/warning/ok* messages. The image shown in Figure 18 represents an example of a *Frequency Analyzer* view.

Figure 18. CTT *Frequency Analyzer*



Instance Name	Status	PRCM Clock Name	Max. Clock Allowed (MHz)	Current Clock Running (MHz)
MMC4	✗	L4PER_32K_GFCLK	0.032	0.044262
	✓	L4PER_L3_GICLK	266.0	13.5
	✓	MMC4_GFCLK	48.0	6.75
MMC3	✗	L4PER_32K_GFCLK	0.032	0.044262
	✓	L4PER_L3_GICLK	266.0	13.5
	✓	MMC3_GFCLK	48.0	6.75
MMC2	✗	L3INIT_32K_GFCLK	0.032	0.044262
	✓	L3INIT_L3_GICLK	266.0	13.5
	✓	MMC2_GFCLK	192.0	0.0
MMC1	✗	L3INIT_32K_GFCLK	0.032	0.044262
	✓	L3INIT_L3_GICLK	266.0	13.5
	✓	MMC1_GFCLK	192.0	0.0
USB2	✓	USB_OTG_SS_REF_CLK	38.4	0.0
	✓	L3INIT_L3_GICLK	266.0	13.5
	✓	L3INIT_960M_GFCLK	960.0	0.0
USB1	✓	USB_OTG_SS_REF_CLK	38.4	0.0
	✓	L3INIT_L3_GICLK	266.0	13.5
	✓	USB_LFPS_TX_GFCLK	34.3	0.0
	✓	L3INIT_960M_GFCLK	960.0	0.0
COUNTER_32K	✓	FUNC_32K_CLK	0.032	0.044262
	✓	WKUPAON_GICLK	38.4	0.0
RNG	✓	L4SEC_L3_GICLK	266.0	0.0
MPU	!	MPU_GCLK	N/A	13.5
	!	MA_EOCP_GICLK	N/A	6.75
EMIF1	!	L3_EOCP _GICLK	N/A	27.0
	✓	EMIF_L3_GICLK	266.0	27.0
	!	EMIF_PHY_GCLK	N/A	13.5
	!	MA_EOCP_GICLK_Switch	N/A	6.75
IEEE1500_2_OCP	✓	L3INIT_L3_GICLK	266.0	13.5
PWMSS3	✓	L4PER2_L3_GICLK	266.0	13.5
PWMSS2	✓	L4PER2_L3_GICLK	266.0	13.5
PWMSS1	✓	L4PER2_L3_GICLK	266.0	13.5
ELM	✓	L4PER_L3_GICLK	266.0	13.5
DES3DES	✓	L4SEC_L3_GICLK	266.0	0.0
CTRL_MODULE_BANDGAP	✓	L3INSTR_TS_GCLK	4.8	1.6875
TIMER9	✓	L4PER_L3_GICLK	266.0	13.5
	✓	TIMER9_GFCLK	100.0	0.0
TIMER8	✓	IPU_L3_GICLK	266.0	13.5
	✓	TIMER8_GFCLK	100.0	27.0
TIMER7	✓	IPU_L3_GICLK	266.0	13.5
	✓	TIMER7_GFCLK	100.0	27.0
TIMER6	✓	IPU_L3_GICLK	266.0	13.5
	✓	TIMER6_GFCLK	100.0	27.0

All peripheral clock frequencies are being compared to maximum allowed frequencies documented in the device-specific Data Manual and results are being displayed in a log fashioned view. There are several types of results that can be shown:

- **The green check mark** shows that clock is running within max frequency limits.
- **The red cross mark** represents an error which means clock is running faster than the maximum frequency supported by the module.
- **The yellow warning** represents a CTT internal clock that is not documented in the device-specific Data Manual. Therefore that clock cannot be verified.
- **The blue warning** represents a clock that is part/speed grade specific and user must manually check the device-specific Data Manual and determine if clock is running within frequency restrictions.

NOTE: Hovering the message icons will display a pop-up detailed status message.

The *Frequency Analyzer* window has two menu options:

- *File*
- *Options*

File -> Save menu: The current results from the *Frequency Analyzer* can be saved in a .csv file.

Option ->Frequency Rounding On: When selected, clocks calculated in the CTT are rounded to the same digits after decimal point shown in the device-specific Data Manual and then the comparison is performed.

Option ->Frequency Rounding Off: When selected, the clocks calculated in the CTT are not rounded.

6.4.6 CTT Help

(6.1) *About Clock Tree Tool*: Shows the CTT device and model/view versioning.

(6.2) *User Manual*: Opens the user manual document on web - http://www.ti.com/tool/CLOCKTREETOOL#technicaldocuments (For Linux users, if the manual does not load, direct download is available at).

(6.3) *SRAS License Agreement* : Displays the license agreement window.

7 CTT Blocks

This chapter describes the different types of blocks that model the clock tree behavior in the CTT.

NOTE: Any modification of the Block parameters in the *CONTROLLER VIEW* affects the associated register bitfields. The *REGISTERS VIEW* switches to the most recently updated register, while all the bit/bitfield value changes are also added to the *Trace View*.

7.1 CTT Pin Block

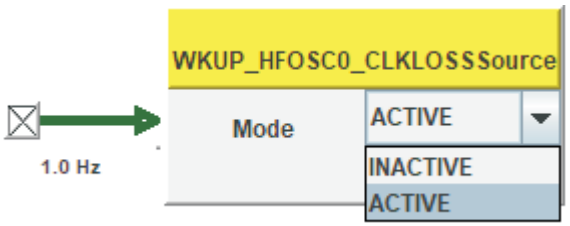
The *Pin* block models a device pin. Figure 19 shows an example of a CTT *Pin* block.

Figure 19. CTT *Pin* Block

7.2 CTT Pin-Clock Source Block

The *Pin-Clock Source* block models a device control bit. Figure 20 shows an example of a CTT *Pin-Clock Source* block. From functional point of view - it can be active or inactive.

Figure 20. CTT *Pin-Clock Source* Block

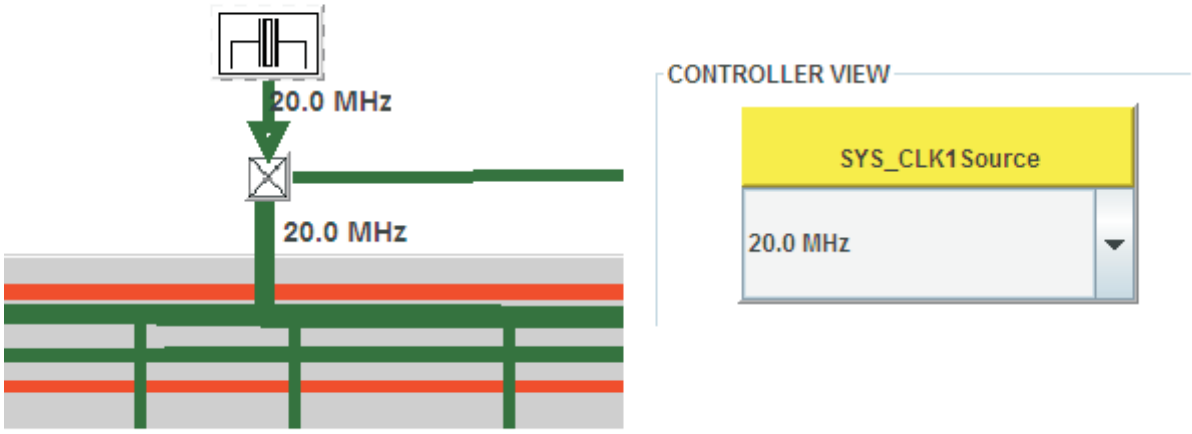


7.3 CTT Crystal Block

The *Crystal* block models an Xtal. In the *CONTROLLER VIEW*, possible frequencies of the *Crystal* can be chosen from a drop-down menu. Figure 21 shows an example of a CTT *Crystal* block.

The currently selected frequency is identified in the *CONTROLLER VIEW* and also in the label next to the *Crystal* block in the *MAIN VIEW*.

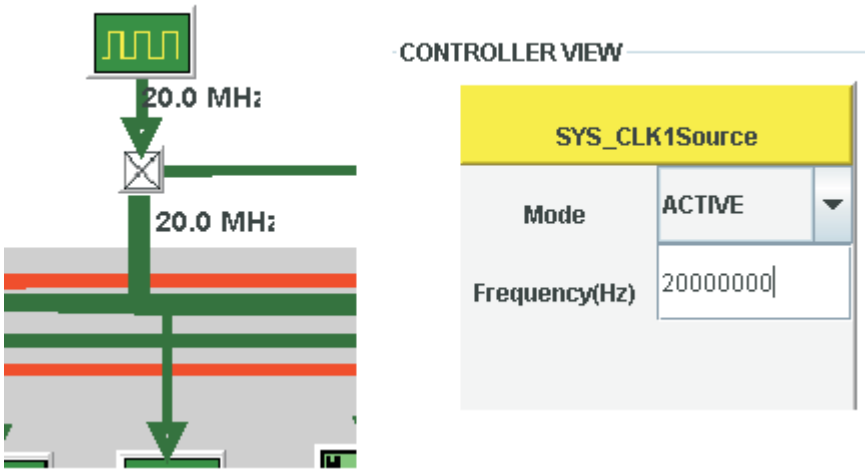
Figure 21. CTT *Crystal* Block



7.4 CTT Clock Source Block

The *Clock Source* block defines the CTT input source clocks. The input clocks are set by entering the value (in Hz) in the text field and changing the state of the block to active as also shown in the example in Figure 22 below.

Figure 22. CTT *Clock Source* Block

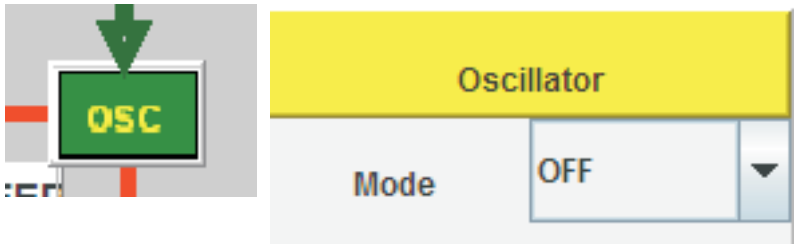


See also Section 6.4.4.2 for saving/loading source clock configuration.

7.5 CTT Oscillator Block

The *Oscillator* block models an oscillator. Its state can be set in the *CONTROLLER VIEW* using the drop-down menu. Figure 23 shows an example of an *Oscillator* block.

Figure 23. CTT *Oscillator* Block



7.6 CTT Clock Switch Block

The *Clock Switch* block allows the clock gating control (that is, enable/disable) within the branches of the clock tree. Essentially, three different types of switches are defined:

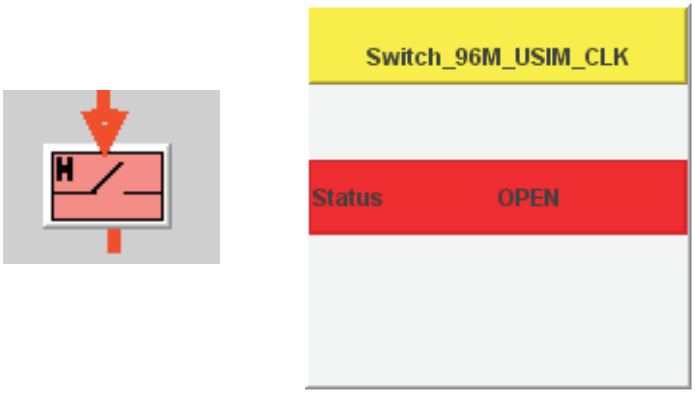
- *Hardware Switch*
- *Manual Switch*
- *Auto Switch*

7.6.1 CTT Hardware Switch Block

Figure 24 shows an example of a *Hardware Switch* block model. This block is controlled by the following hardware gating conditions:

- The derived clock is inactive.
- All modules receiving the derived clock are inactive.
- All switches receiving the derived clock are gated (open).

Figure 24. CTT *Hardware Switch* Block



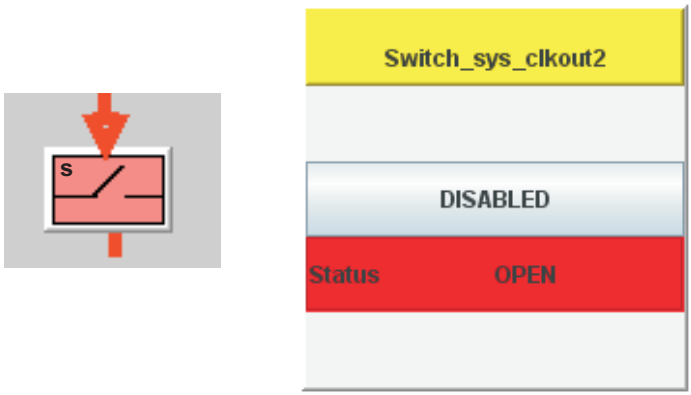
The user has no control over this switch. It is automatically closed when the hardware gating conditions are satisfied.

NOTE: A derived clock is the clock at the output of the switch.

7.6.2 CTT Manual Switch Block

Figure 25 shows an example of a *Manual Switch* block model. This block is software controlled by setting or clearing the enable bits in corresponding registers (generally applicable to module functional clocks). The user can enable or disable the switch using the button in the *CONTROLLER VIEW*.

Figure 25. CTT *Manual Switch* Block



The derived clock from the *Manual Switch* may be connected to multiple modules and can have one or more ENABLE bits associated, to request this clock.

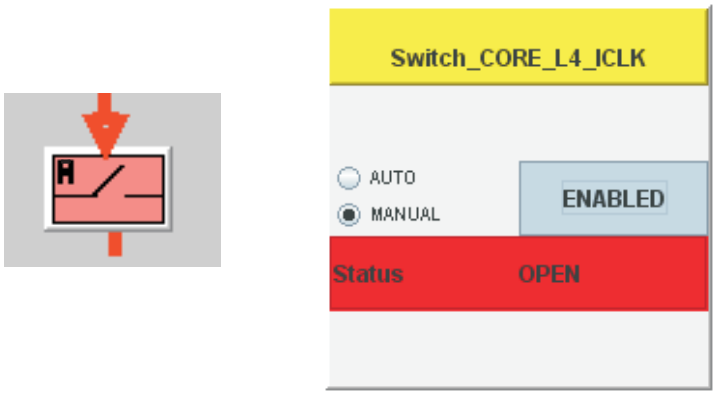
The *Manual Switch* gating condition is:

- All the associated clock ENABLE bits for this clock are cleared to 0.

7.6.3 CTT Auto Switch Block

Figure 26 shows an example of an *Auto Switch* block model. This block is a software or hardware controlled switch. The user can either manually (through software control) enable or disable the derived clock - *MANUAL* mode; or set the switch to *AUTO* mode.

Figure 26. CTT *Auto Switch* Block



The *MANUAL* mode is set by clicking on the *MANUAL* check box (see Figure 26).

In this mode when the push-button on the right side is in ENABLED state, all the associated clock ENABLE bits are set and the switch is closed. Similarly, if the push-button is in DISABLED state, all the clock ENABLE bits are cleared to 0 and the switch is open.

The user can set the switch to *AUTO* mode using the following sequence:

1. Push the push-button to ENABLED state, to set all clock ENABLE bits to 1.
2. Click on the *AUTO* check box to set all the clock AUTO bits to 1.

In the *AUTO* mode the clock is controlled by hardware gating conditions. Hence, whenever the gating conditions are satisfied the clock is automatically disabled; when any of the gating conditions is not satisfied the clock is automatically enabled by the hardware. In this mode no software control of clocks is necessary.

The *AUTO* mode has two clock gating conditions:

- Manual (software)
- Hardware

The manual (software) control clock gating condition is:

- All the associated clock ENABLE bits for this clock are cleared to 0.

The hardware control clock gating conditions are:

- All the associated clock ENABLE bits and clock AUTO bits for this clock are set to 1.
- The derived clock is not requested by any module (that is, the module is inactive).

NOTE: Both the software and hardware clock gating conditions of the *AUTO* mode must be satisfied for the derived clock to be gated automatically.

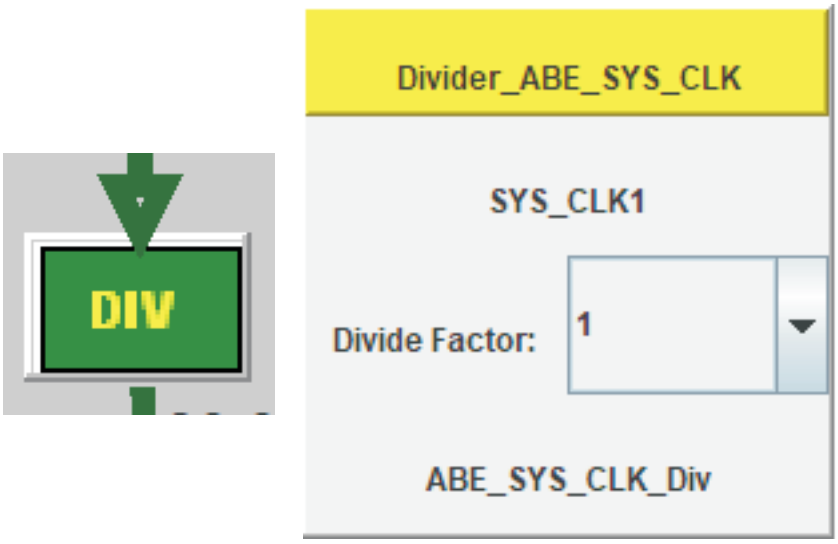
In the *AUTO* mode, the switch will automatically close when any of its gating conditions is not satisfied.

7.7 CTT Divider Block

The *Divider* block performs clock frequency division. Figure 27 shows an example of a *Divider* block. The output clock frequency is the frequency of the input clock divided by the *Divide Factor*, selected by clicking on the associated drop-down list.

NOTE: If the divider has a fixed *Divide Factor* (that is, the software can not change the *Divide Factor*) then the drop-down list contains only one division factor.

Figure 27. CTT *Divider* Block

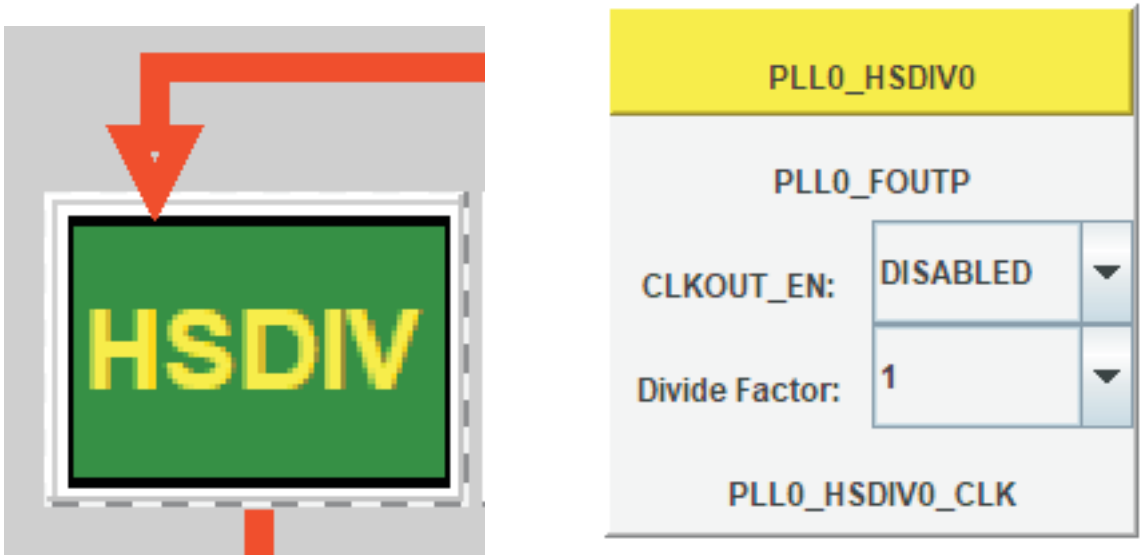


7.8 CTT High Speed Divider Block

The *High Speed Divider* block performs clock frequency division. Figure 28 shows an example of a *High Speed Divider* block. The output clock frequency is the frequency of the input clock divided by the *Divide Factor*, selected by clicking on the associated drop-down list.

NOTE: If the divider is in DISABLED state, set it to ENABLE state, by selecting the mode in the *CLKOUT_EN* drop-down list.

Figure 28. CTT *High Speed Divider* Block



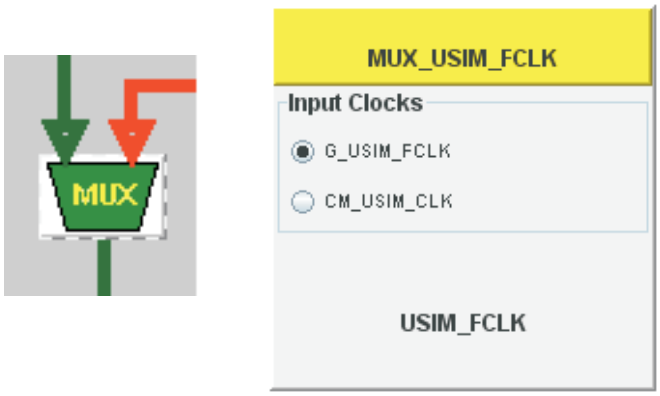
7.9 CTT MUX Block

7.9.1 CTT Basic MUX Block

The *Basic MUX* block is used to perform a selection from multiple source clocks for the derived clock. The user can select the source clock by clicking on the check box corresponding to one of the multiple source clocks in the *CONTROLLER VIEW*. Figure 29 shows an example of a *Basic MUX* block.

The currently selected source clock is identified in the *CONTROLLER VIEW*.

Figure 29. CTT *Basic MUX* Block

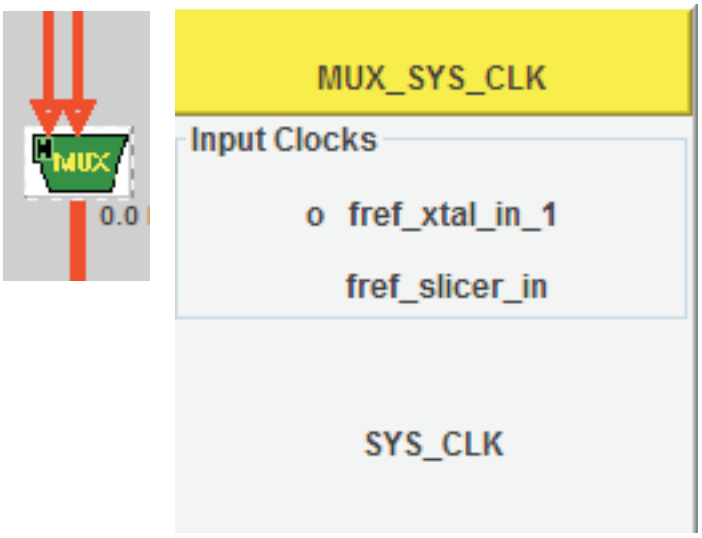


7.9.2 CTT Priority MUX Block

The *Priority MUX* block has predefined priorities of its inputs and the hardware selects the highest priority active clock. Figure 30 shows an example of a *Priority MUX* Block.

The currently selected source clock is identified in the *CONTROLLER VIEW*.

Figure 30. CTT *Priority MUX* Block



7.10 CTT DPLL Block

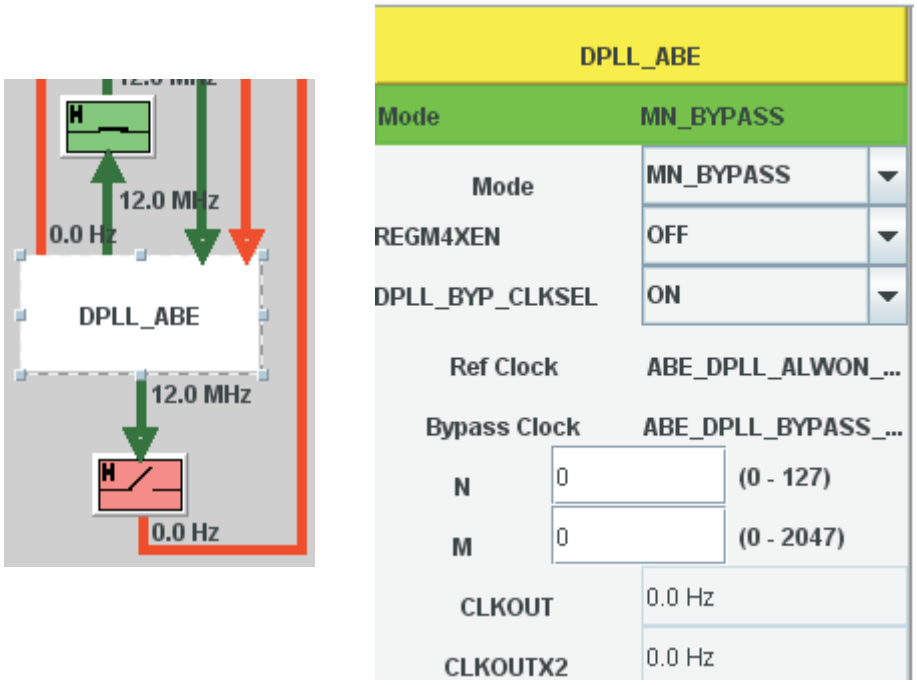
The *DPLL* block receives source clocks and in turn generates the clocks for the device. Refer to the device Technical Reference Manual for details about DPLL functionality.

The following sub-sections present the three basic examples of the *DPLL* blocks.

7.10.1 CTT DPLL Block: First Type

First type of the *DPLL* block is shown in Figure 31.

Figure 31. CTT *DPLL* Block: First Type



The user must follow the below sequence to configure the *DPLL* block of the first type:

1. If the *DPLL* is in *LOCKED* mode, set it to one of the UNLOCKED modes (for example, LOW POWER STOP state), by selecting the mode in the *Mode* drop-down list.
2. Set the *M* and *N* parameters by typing the values in the corresponding edit boxes. NOTE: After entering the value of the parameter in the edit box, ENTER key must be pressed so that the new value is accepted by the tool.
3. Select the output divide factor *M2*, and so forth, by clicking on the associated drop-down list.
4. Switch the *DPLL* to the *LOCKED* mode by clicking on the *Mode* drop-down list and selecting the mode.

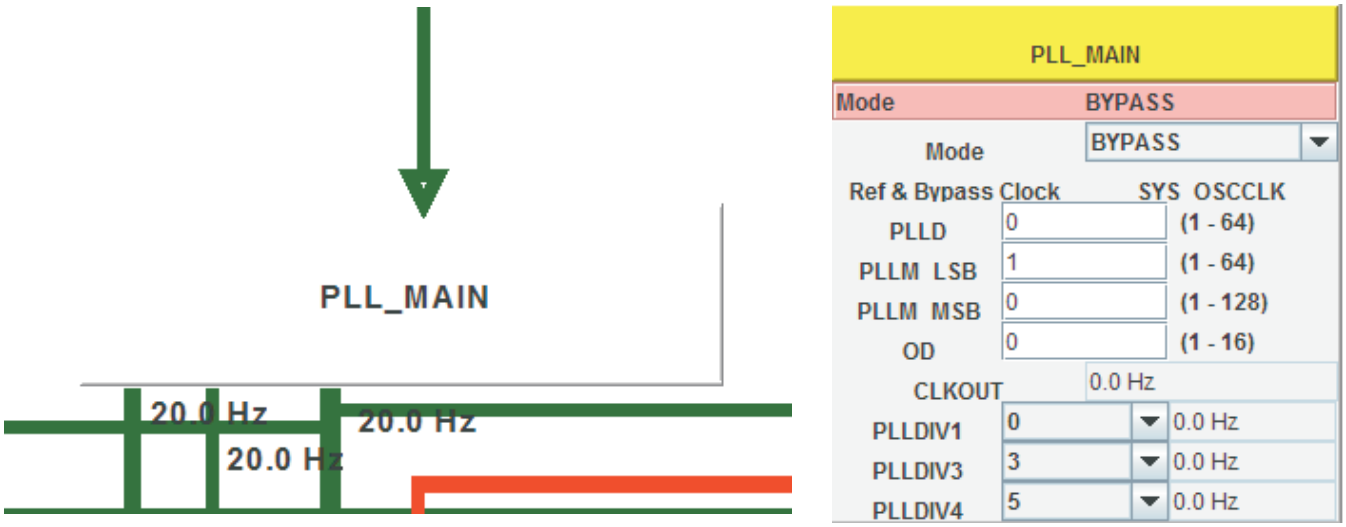
Once the *DPLL* is in *LOCKED* mode the *CLKOUT, CLKOUTX2* and the output clock frequencies (displayed after the output divide factors) will be updated.

DPLLs can also have options from the controller to select bypass clocks, 4xen mode, CLOCKOUTIF, and sd-div modes. For more information about these functionality please refer to, dependent on the device of use, the PRCM Chapter or Clocking Section in the device Technical Reference Manual.

7.10.2 CTT DPLL Block: Second Type

Second type of the *DPLL* block is shown in Figure 32.

Figure 32. CTT *DPLL* Block: Second Type



The user must follow the below sequence to configure the*DPLL* block:

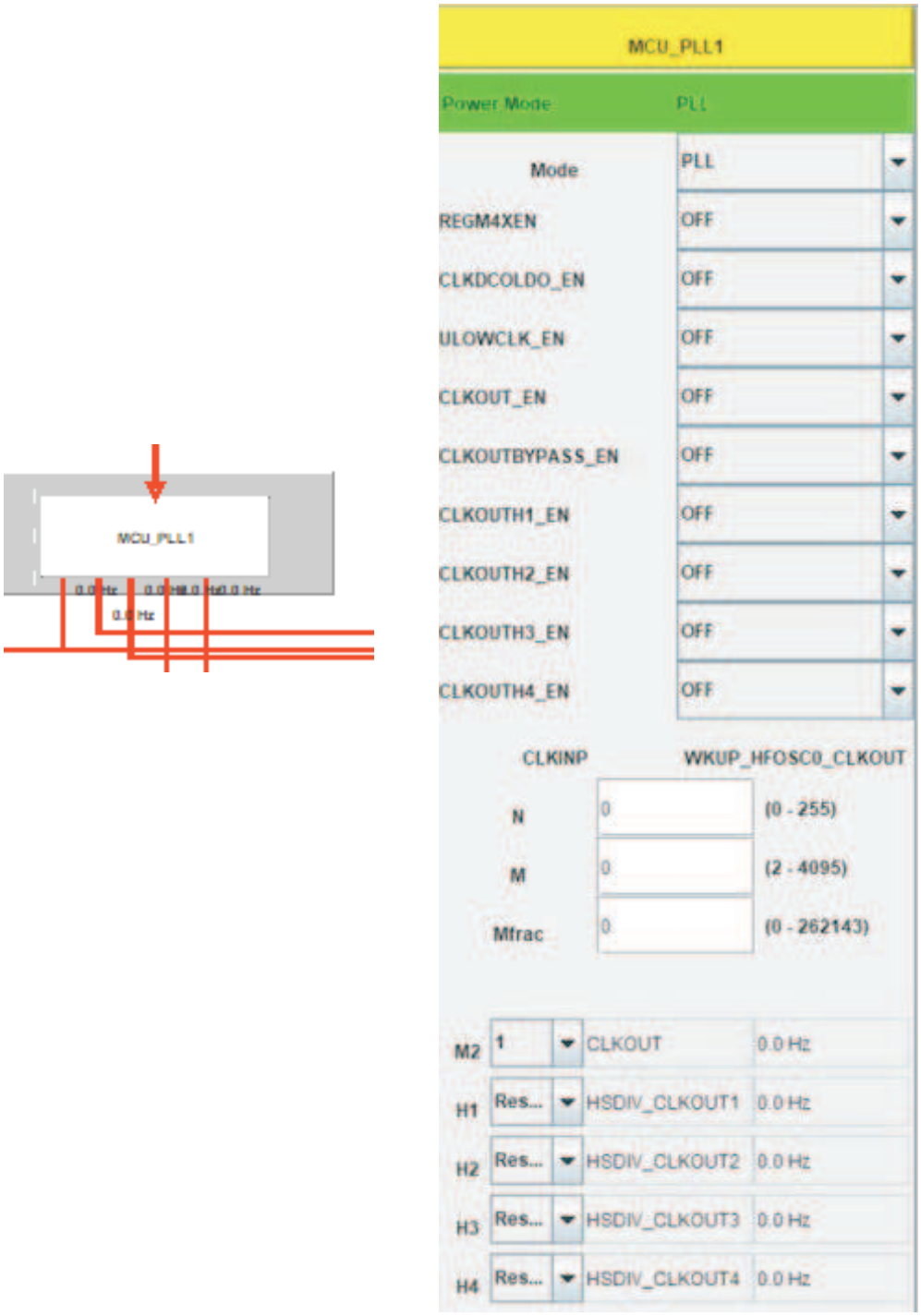
1. Select the mode in the *Mode* drop-down list, for example, *PLL* mode.
2. Set the multipliers and dividers parameters (for example, *PLLM_LSB, PLLD*) by typing the values in the corresponding edit boxes. NOTE: After entering the value of the parameter in the edit box, ENTER key must be pressed so that the new value is accepted by the tool.
3. If available, select the output divide factor *PLLDIV1*, and so forth, by clicking on the associated drop-down list.

Once the *DPLL* state is set, the state of *CLKOUT*, and the output clock frequencies (displayed after the output divide factors) will be updated.

7.10.3 CTT DPLL Block: Third Type

Third type of the *DPLL* block is shown in Figure 33.

Figure 33. CTT *DPLL* Block: Third Type



The user must follow the below sequence to configure the *DPLL* block:

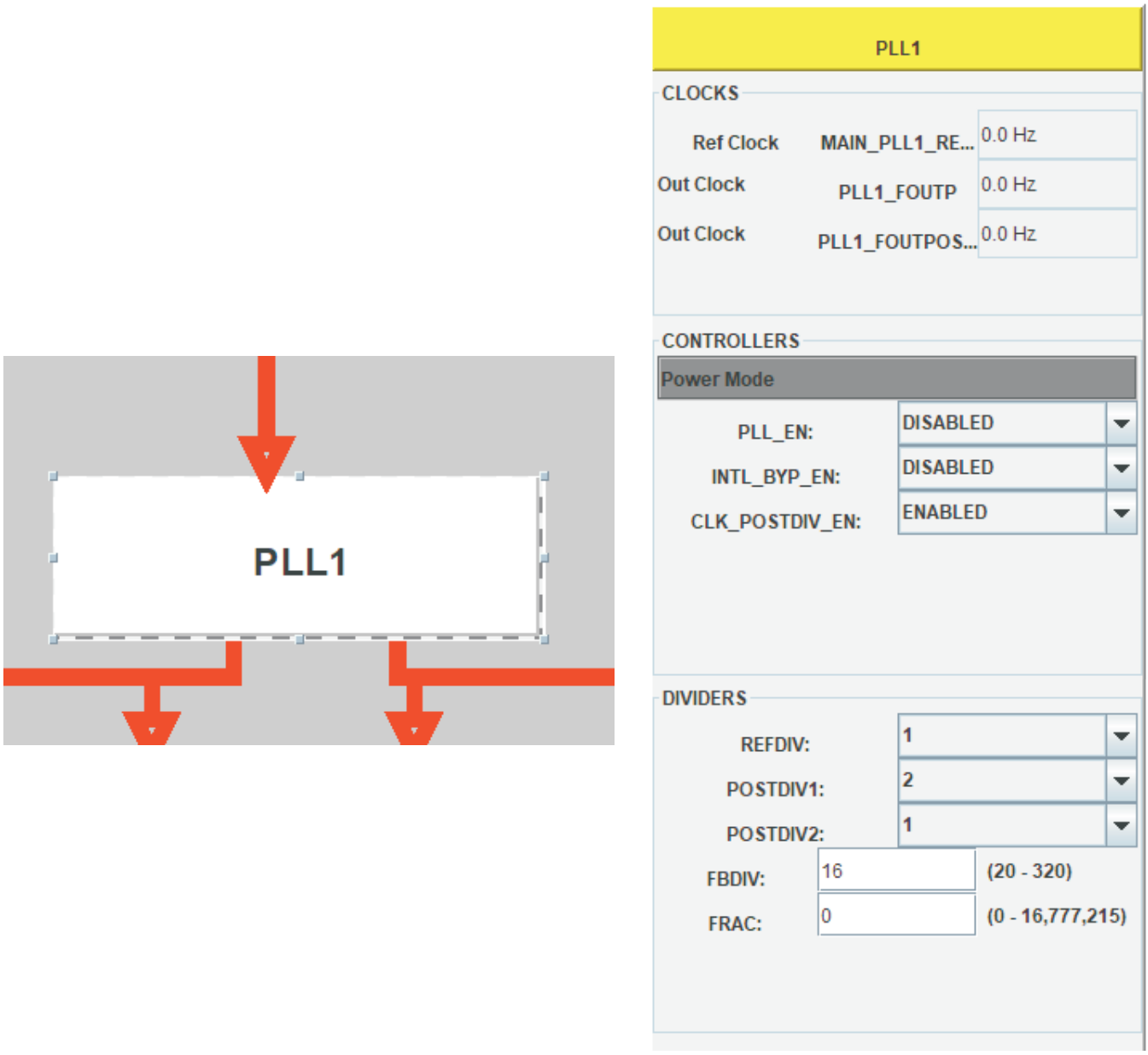
1. Select the mode in the *Mode* drop-down list, for example, *PLL* mode.
2. Set the *M, N* and *Mfrac* parameters by typing the values in the corresponding edit boxes. NOTE: After entering the value of the parameter in the edit box, ENTER key must be pressed so that the new value is accepted by the tool.
3. Enable the clock by changing the state to ON of the <*clock_name*>*_EN* field.
4. Select the output divide factor *M2*, and so forth, by clicking on the associated drop-down list.

Once the *DPLL* is in *PLL* state the *CLKOUT, CLKOUTX2* and the output clock frequencies (displayed after the output divide factors) will be updated.

7.10.4 CTT DPLL Block: Fourth Type

Forth type of the *DPLL* block is shown in Figure 34.

Figure 34. CTT *DPLL* Block: Fourth Type



The user must follow the below sequence to configure the *DPLL* block:

1. If the *DPLL* is in *DISABLED* state, set it to *ENABLE* state, by selecting the mode in the *PLL_EN* drop-down list.
2. Set the *FBDIV* and *FRAC* parameters by typing the values in the corresponding edit boxes. NOTE: After entering the value of the parameter in the edit box, ENTER key must be pressed so that the new value is accepted by the tool.
3. Select the output divide factor *REFDIV* for <pll_name>_FOUTP output clock, by clicking on the associated drop-down list.

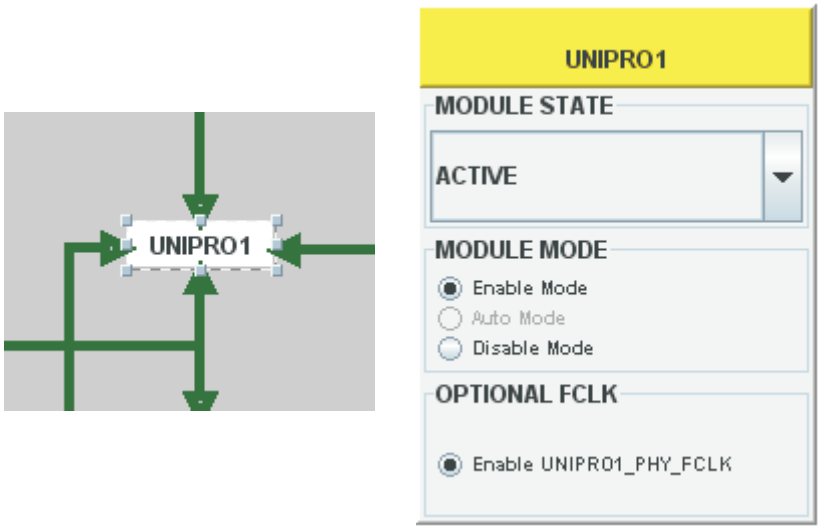
4. Select the output divide factor *POSTDIV1* and *POSTDIV2* for <pll_name>_FOUTPOSTDIV output clock, by clicking on the associated drop-down list. NOTE: To disable *POSTDIV1* and *POSTDIV2* dividers, select *CLK_POSTDIV_EN* to DISABLE and this set <pll_name>_FOUTPOSTDIV equal to <pll_name>_FOUTP output clock.

Once the *DPLL* state is set, the output clock frequencies (displayed after the output divide factors) will be updated.

7.11 CTT Module Block

A *Module* block represents the destination modules, such as I2C, MCSI, McBSP, and so forth. A *Module* receives functional and interface clocks. It may be active or inactive. It can also be in enabled, auto, or disabled mode. *Module* can also have optional functional clocks associated to it. An example of a *Module* is given in Figure 35.

Figure 35. CTT *Module* Block



If a module has only *ACTIVE/IDLE* functionality, the user can switch a module to *ACTIVE* or *IDLE* state and only the *MODULE STATE* drop down menu will be displayed inside the *CONTROLLER VIEW*.

If a module has *MODULE MODE* and *MODULE STATE* functionality, the user must select *MODULE MODE* value first, then the *MODULE STATE* value.

If a *Module* has *OPTIONAL FCLK* functionality, the user may enable optional clocks as well.

NOTE: In the device, there are various combinations of *Module* functionality. A given *Module* can have one or more at the same time.

In basics, the clocks associated to *Module* function as follows:

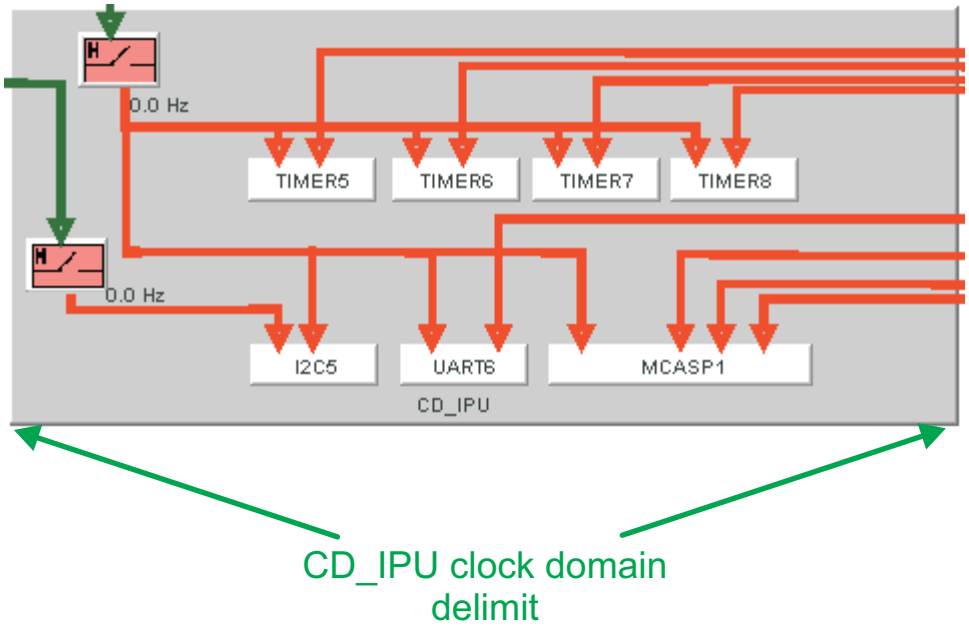
1. Optional functional clock is running whenever the OPTFCLKEN bit is set to 1, and it is not concerned by the *MODULE STATE* (*ACTIVE/IDLE*).
2. Module mode associated clocks are automatically gated if *MODULE MODE* is set to *Disable Mode* and this is the *Module* reaches idle state.
3. When *MODULE MODE* is set to *Enable Mode* functional clock is automatically un-gated. The interface clock is automatically gated/un-gated based on the *Module ACTIVE/IDLE* transition.
4. Module mode associated clocks are automatically gated/un-gated when *MODULE MODE* is set to *Auto Mode* based on the *ACTIVE/IDLE* transition of the *Module*. *Auto Mode* option is available only for modules with interface idle protocol associated clock(s).

For more information about module mode, module state, and optional clocks associated to modules, please refer to, dependent on the device of use, the PRCM chapter or Clocking section in the device Technical Reference Manual.

7.12 CTT Delimits

The *Delimit* block is a highlighted area in the GUI. This block does not have a defined associated controller. The block purpose is to highlight the boundaries of a given clock domain or of a set of instances (for example, *DPLL* blocks) with a similar function. This way the GUI provides better visual interpretation of modules and their clock domain affiliation. Figure 36 shows an example of a *Delimit*.

Figure 36. CTT *Delimit* Block



8 CTT Release Notes

Specific CTT package changes are listed in TI CTT Release Notes.

9 CTT Limitations

For the known limitations of the CTT go to Known Issues.

Revision History

Changes from H Revision (October 2019) to I Revision	Page
• Added CTT High Speed Divider Block section	23
• Added fourth example of PLL block to cover all types in diferent CTT packages	28

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265