HET IDE Tutorials

User's Guide

Literature Number: SPNU485C
May 2010–Revised August 2016

Contents

List of Figures

HET IDE Tutorials

This document contains instructions for the HET IDE tutorials. The HET IDE is a free integrated development environment for the programmable high-end timer module. It supports the HET, NHET, and N2HET timers from the TMS470M, TMS570, RM4xx, and RM5xx Hercules™ MCU product lines. The HET IDE includes a code editor, assembler, simulator and Waveform Viewer and editor. The code editor has an 'Insert Instruction' wizard to assist in learning HET assembly syntax, as well as an algorithm library that allows you to simply copy and paste frequently used timing functions into your own custom program. The assembler produces files for simulation and also for linking into your final application. The simulator allows you to debug your program with control and visibility that is not available when running your program on silicon. You can also stream waveforms to and stimulus from an external waveform tool (Synapticad Waveformer Pro) so that the interaction of the timer and the external world can be modeled.

1 Installation

1.1 Installing the HET IDE

The HET IDE can be downloaded from http://www.ti.com/lit/zip/spnc016. Once you download this file, unzip and run the included installer. This tutorial document should be used with HET IDE version 03.05.01 or later.

1.2 Tutorial Installation

The tutorials are automatically installed when the HET IDE is installed. However, you must make a personal copy of the tutorial folder before beginning to work through the tutorials.

The HET IDE installer places the tutorials in the "%PROGRAMDATA%\Texas Instruments\Hercules\HET IDE\<version>\Tutorials" folder. This folder may be hidden, so the installer also places a shortcut to open the folder in windows explorer in the start menu (see Figure 1). The installation folder is read-only and is shared by all users, so make a personal copy of the tutorials folder in your own directory location before continuing. In the following instructions, your copy of the HET IDE tutorials is referred to with the path <your_copy>.

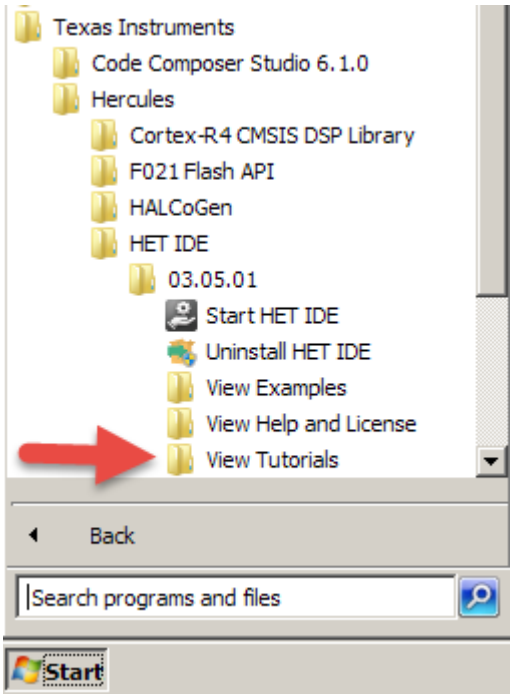


Figure 1. Shortcut to Tutorials Installation (Read Only) Folder

2 Tutorials

It is assumed that you have installed the HET IDE, as described in Section 1.1, and that you are working in your personal copy (<your_copy>) of the tutorials folder as described in Section 1.2

Also, between tutorials you should either close and restart the HET IDE, or use the RESTART button on the HET IDE toolbar to exit an existing simulation before moving on to the next tutorial.

NOTE: Sometimes when changing projects or exiting the program, the Synapticad waveform viewer may present you with the option to "Save Files". It is very easy to miss this dialog and the HET IDE may appear unresponsive while this dialog is open. If the HET IDE appears unresponsive, switch the focus back to the Synapticad application and close this. Open the 'Save Files" dialog (Save None is a good choice for the tutorials). Then, the HET IDE will become responsive again.

NOTE: If you have multiple projects open in the HET IDE, then your project must be made 'the active project' before you can assemble and load the project into the simulator. You can make any project 'the active project'; right click on the project name in the Project Explorer pane of the HET IDE and choose 'Set as Active Project' from the context menu. 'The active project' is always shown in **bold** face.

NOTE: If the *Assemble* or *Assemble and Load* toolbar buttons are greyed out in the HET IDE, but the project is active and there are no simulations currently in progress, it sometimes helps to close the .het source file. Reopening the source file in the editor usually makes these buttons available again.

2.1 Tutorial 1 — Getting Started

2.1.1 Concepts Covered

This Tutorial covers the following topics:

- Opening a project
- Breakpoints
- Run a program
- Cycle count
- Waveform wizard
- View the waveform

2.1.2 Step-By-Step Instructions

1. Double click the "Start HET IDE" icon on the desktop, or from the start menu location shown in Figure 1.
2. Open the Project:
   (a) Click Project → Open Project to open the project.
   (b) Open the *tut1_getting_started.prj* file from <your_copy>\Tutorials\tut1\tut1_overview\tut1_overview.prj
   (c) Confirm that the project 'tut1_overview' is the active project. It should be listed in **bold** face in the Project Explorer pane of the HET IDE.
3. Build the Project:
   (a) Click Debug → Assemble.
   (b) This step compiles the HET file and generates the result in the output window. Make sure that no errors are generated. If there are any errors, correct them and assemble it again.
4. Load an HET program:
   (a) Click Debug → Load HET Program to load the HET program into the RAM.
   (b) You will see the program load into the simulator and Synapticad Waveformer Pro or Waveviewer Free should launch.
   (c) Bring the HET IDE window into focus (in case Synapticad is currently in focus).

5. Configure the signals to be viewed in the waveform using Waveform Wizard of the HET IDE:
   (a) In the HET IDE, click Tools → Waveform Wizard. Your screen should look like Figure 2.

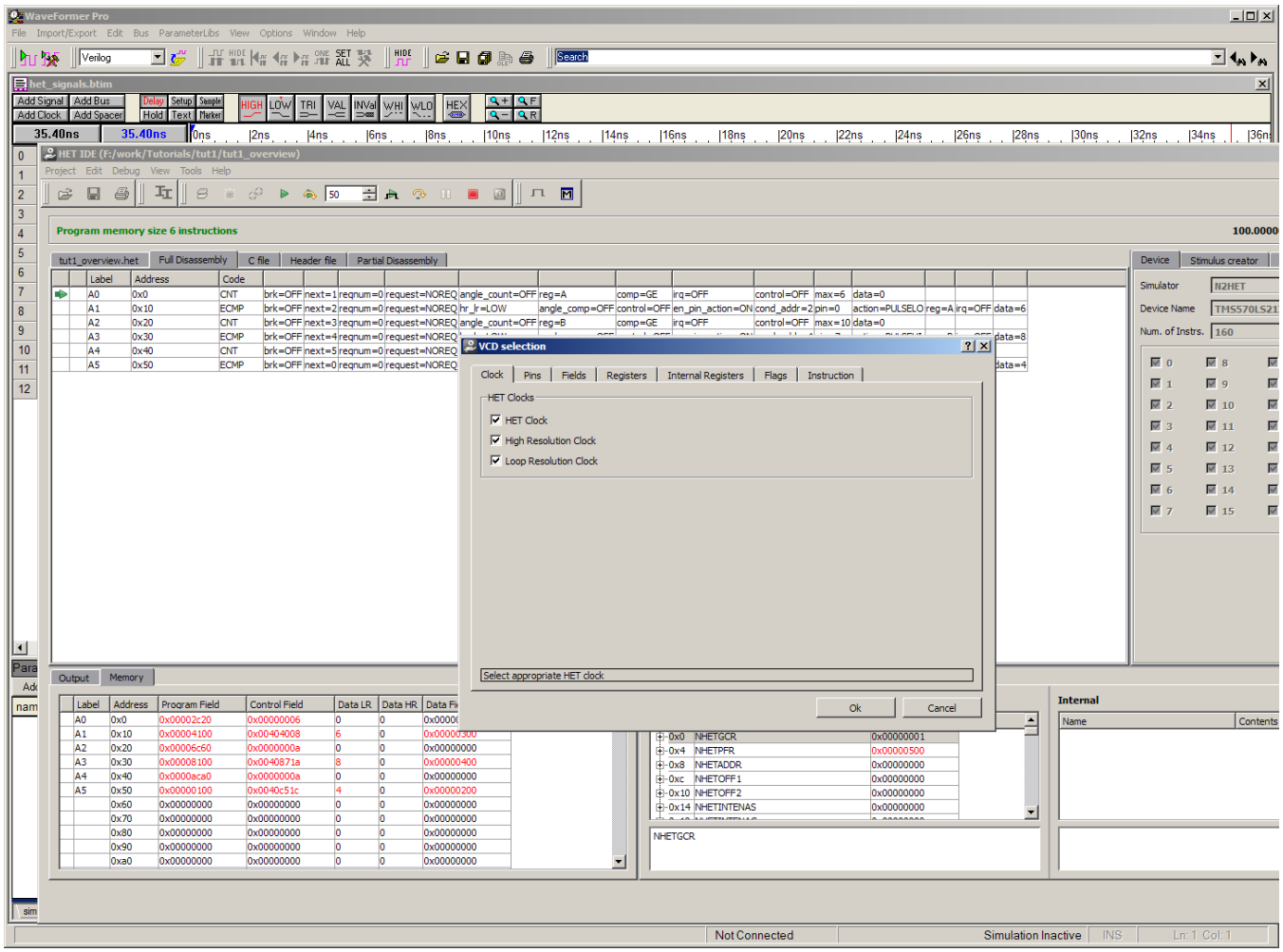


Figure 2. Waveform Wizard

(b) On the Waveform Wizard tab "Clock":
   (i) Select HET Clock .
   (ii) Select High Resolution Clock.
   (iii) Select Loop Resolution Clock.

(c) On the Waveform Wizard tab "Pins" :
   (i) Select HET_0.
   (ii) Select HET_5.
   (iii) Select HET_7.

(d) On the Waveform Wizard tab "Fields":
   (i) Deselect PROGRAM.
   (ii) Deselect CONTROL.
   (iii) Select Data.

(e) On the Waveform Wizard tab "Registers":
   (i) Select NHETADDR.
   (ii) Deselect all other registers.

(f) On the Waveform Wizard tab "Internal Registers":
   (i) Select A,B,T.
   (ii) Deselect all other registers.

(g) On the Waveform Wizard tab "Flags":
   (i) Select Z.
   (ii) Deselect all other flags.

(h) On the Waveform Wizard tab "Instruction":
   (i) Select address 0x0 from the Address drop-down list.
   (ii) Select the PROGRAM, CONTROL, and DATA Instruction Fields.
   (iii) Press the "Add" button. Make sure that you see address 0x0 listed in trace slot 0 ("Sl. No").
   (iv) Repeat steps i - iii, but substitute addresses 0x20 and 0x40 for address 0x0.
   (v) Confirm that you see address 0x20 in slot 1 and address 0x40 in slot 2.

(i) Click OK.

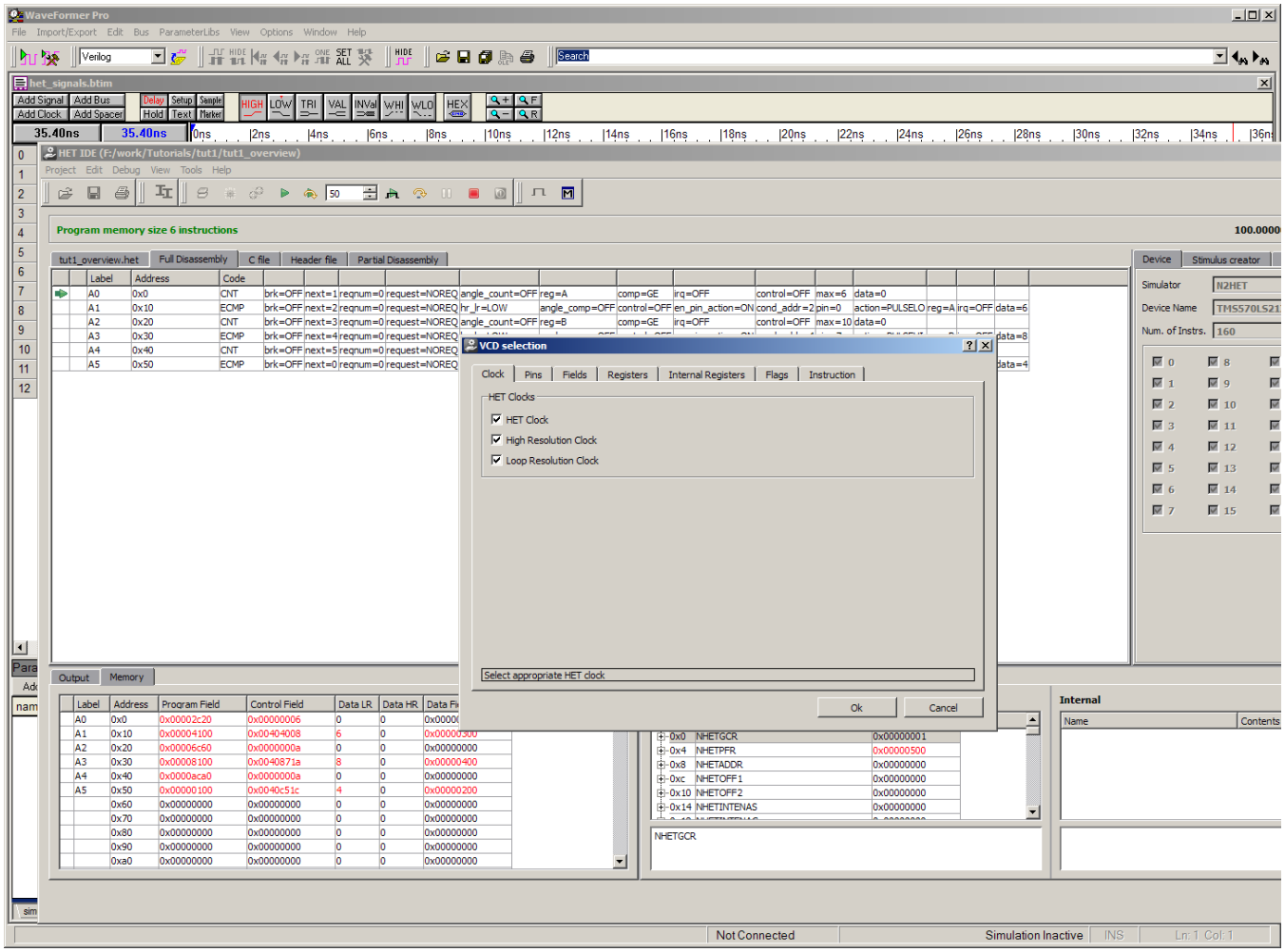


Figure 3. Waveform Wizard

6. Insert Break Points:
   (a) Double click on Labels A1, A3, and A5 to insert break points.
   (b) Confirm that your screen looks like Figure 4.

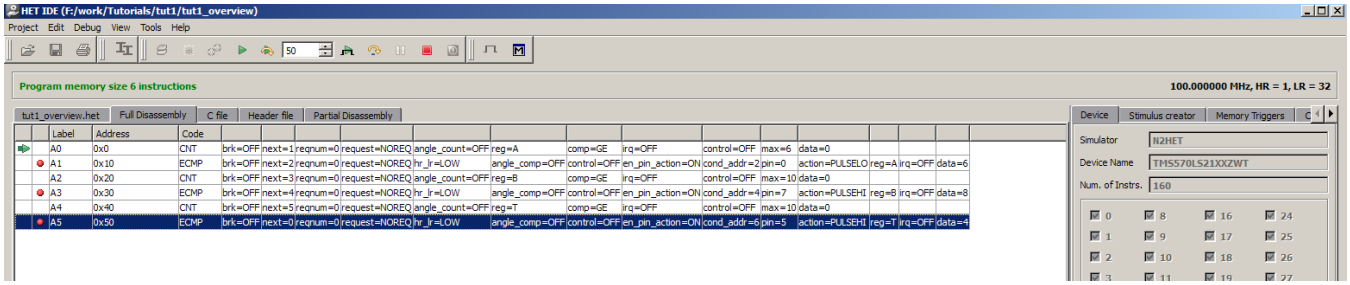
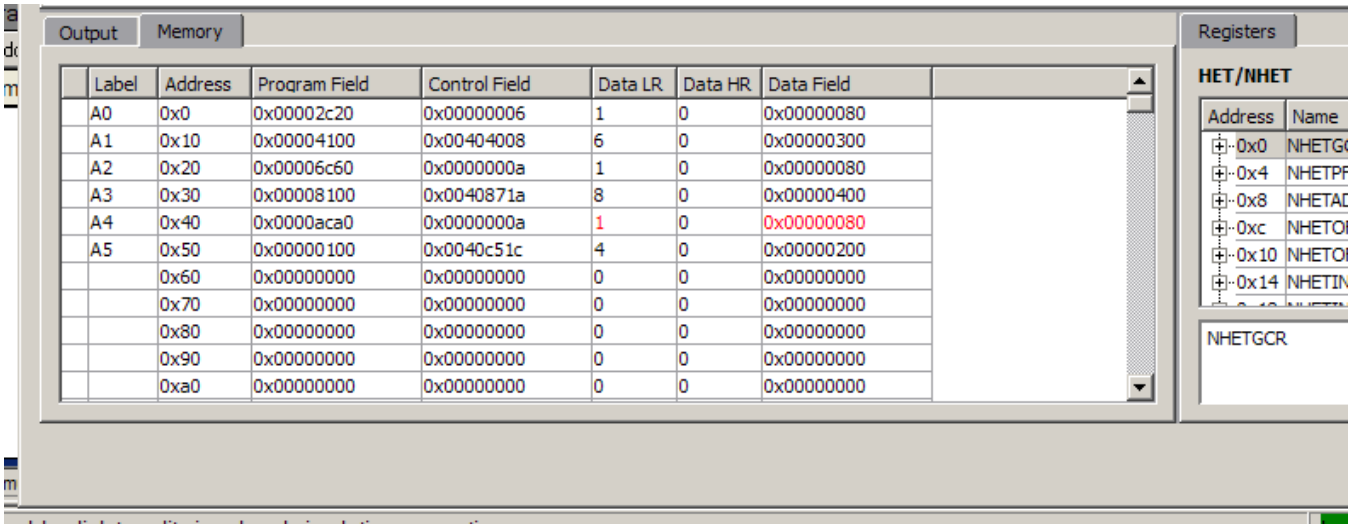


Figure 4. Breakpoints

7. Run the Program with different Run options:
   (a) Click Debug → Run (F5) to run the program. You should stop at the first breakpoint at label A1.
   (b) Run two more times until reaching the breakpoint at label A5.
   (c) Verify that the Data Fields of A0 (address 0x0), A2 (0x20) and A4 (0x40) all have the value 0x00000080 (Data LR = 1, Data HR = 0). You should see a memory window that matches Figure 5.



Label	Address	Program Field	Control Field	Data LR	Data HR	Data Field
A0	0x0	0x00002c20	0x00000006	1	0	0x00000080
A1	0x10	0x00004100	0x00404008	6	0	0x00000300
A2	0x20	0x00006c60	0x0000000a	1	0	0x00000080
A3	0x30	0x00008100	0x0040871a	8	0	0x00000400
A4	0x40	0x0000aca0	0x0000000a	1	0	0x00000080
A5	0x50	0x00000100	0x0040c51c	4	0	0x00000200
	0x60	0x00000000	0x00000000	0	0	0x00000000
	0x70	0x00000000	0x00000000	0	0	0x00000000
	0x80	0x00000000	0x00000000	0	0	0x00000000
	0x90	0x00000000	0x00000000	0	0	0x00000000
	0xa0	0x00000000	0x00000000	0	0	0x00000000

Figure 5. Memory Window After Step 7c

   (d) Run the program an additional three times (press F5 or Debug → Run three more times).
   (e) Remove all the break points by double clicking at the labels. All of the red breakpoint symbols in the second column of the "Full Disassembly" tab should be cleared.
   (f) Click Debug → Step Instruction (or press F11) six times and see that it executes each individual instruction at a time.

(g) Enter 20 in edit box (after Run for Loops icon) in the tool bar (see item 'g' in Figure 6).

(h) Select Debug → Run For Loops from the menu, or Click the Run for Loops button (see item 'h' in Figure 6).

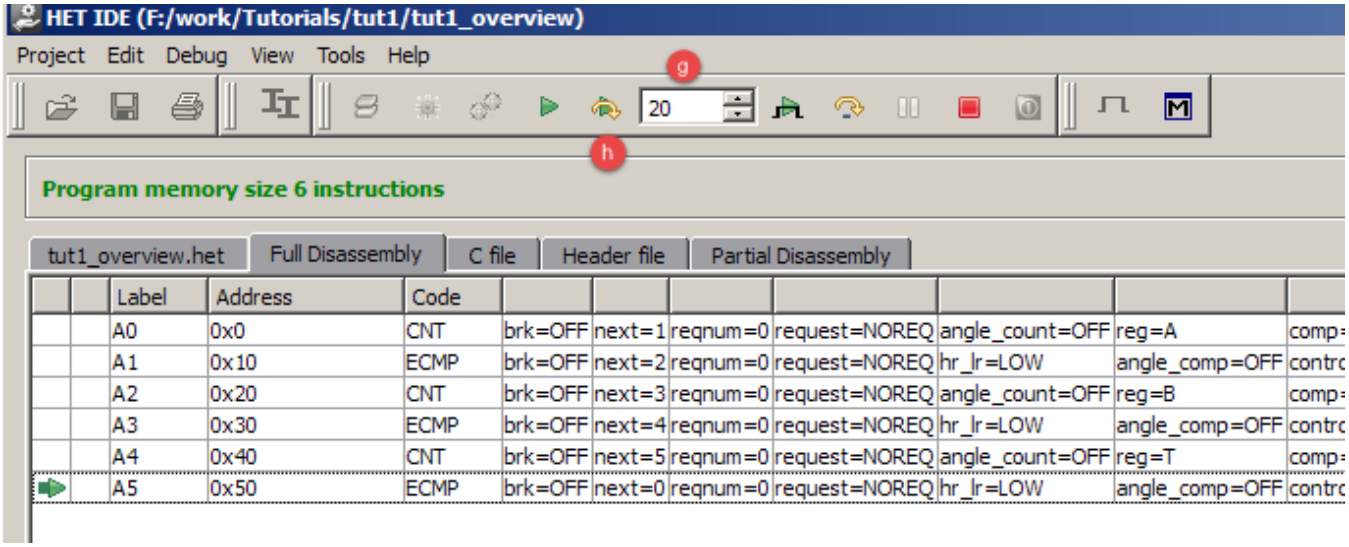


Figure 6. Controls for 'Run For Loops'

(i) Check cycle count at the bottom of the GUI window. It should read 736 cycles (from an initial value of 101 cycles).

8. Stopping Execution:

(a) Click Debug → Pause to pause the execution (if enabled).

(b) Click Debug → Stop to stop the execution or press the stop button.

9. View the Waveform:
   (a) Switch to the Synapticad Waveformer Pro or Waveviewer Free Application.
   (b) Press the Zoom-Full Button (Magnifier with "F" beside it). Your screen should look like Figure 7.

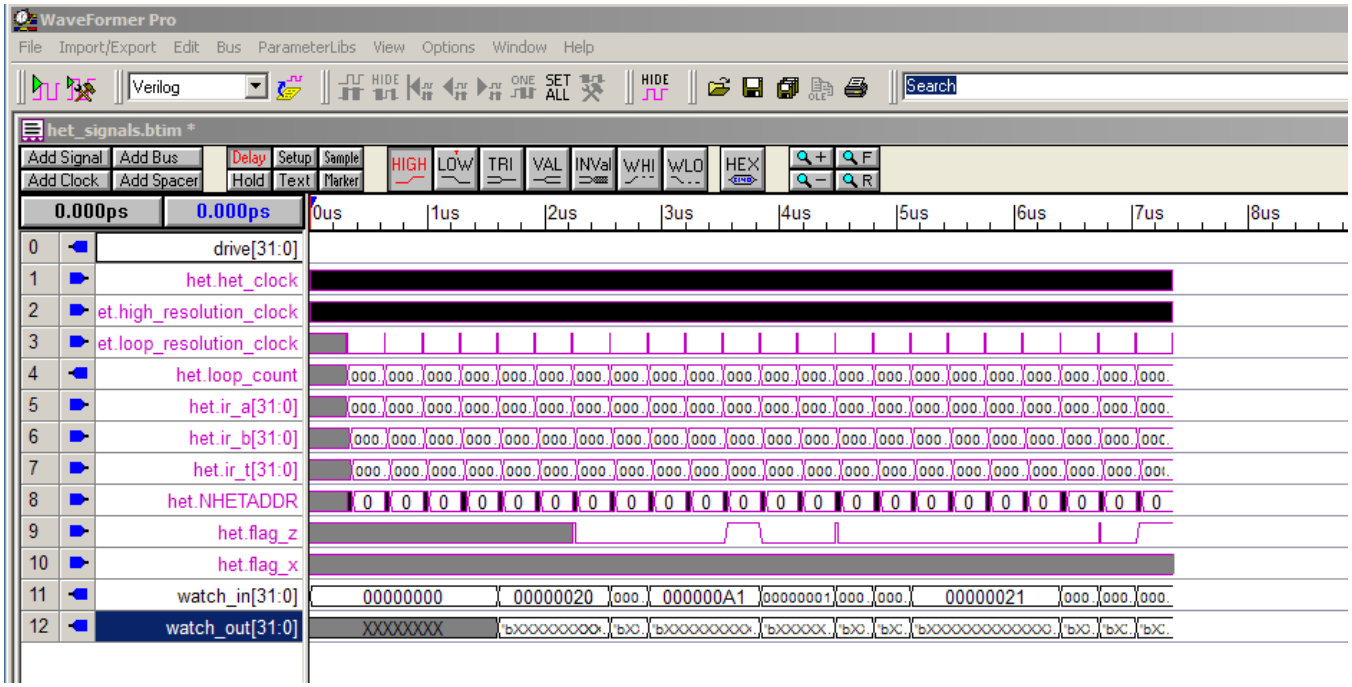


Figure 7. Waveform at Completion of Tutorial 1

2.2 Tutorial 2 — Input Stimuli

This tutorial covers the following features:

- Creating repetitive input stimulus using the built-in Stimulus Creator
- Using the Insert Instruction wizard to add instructions to a program.

1. Open the HET IDE.
2. Create a new project, called my_tut2. From the HET IDE file menu, select Project → New Project. When the New Project dialog appears (see Figure 8), change the project name to 'my_tut2' and make sure the project path is pointing to a folder where you can read or write.

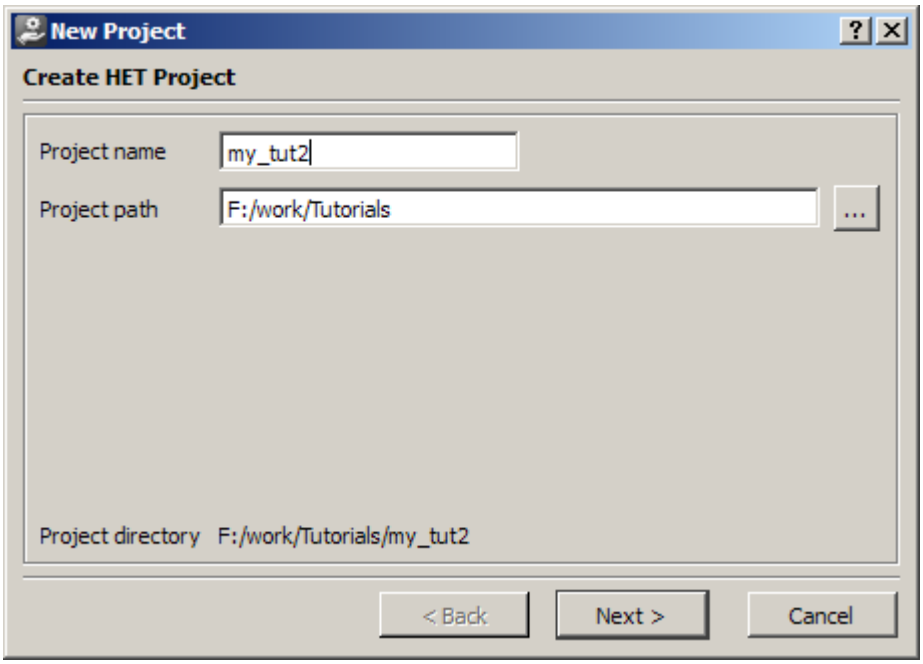


Figure 8. New Project Dialog - Initial Screen

3. Press the Next button to bring up a selection of HET devices. For this project, select the TMS570LS31XXZWT device (see Figure 9).

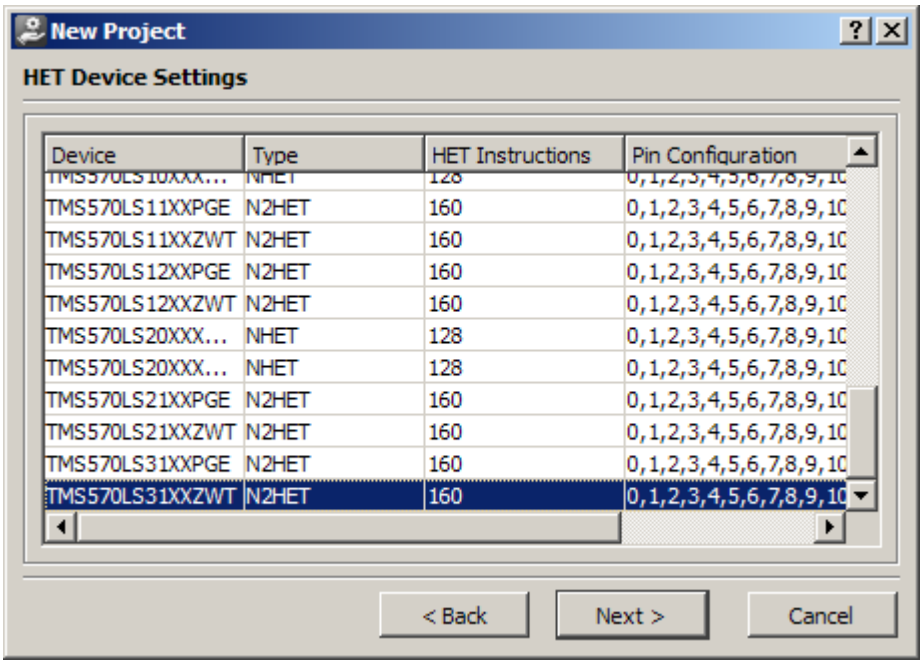


Figure 9. New Project - Device Selection

4. Press the Next button again to bring up the Clock Frequency dialog. Set the clock frequency to 90 MHz (see Figure 10).

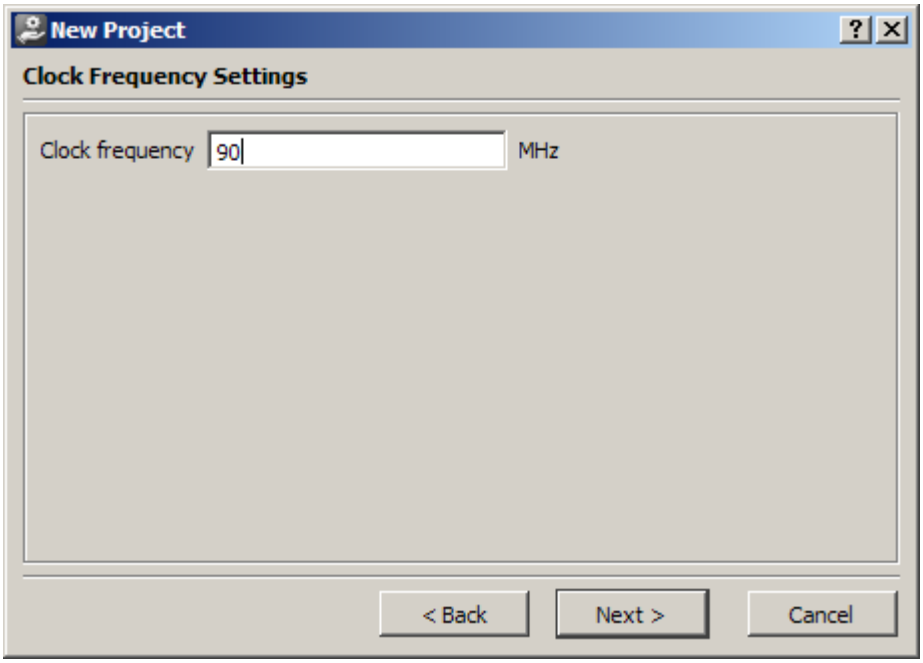


Figure 10. New Project - Clock Frequency

5. Press Next and your project will be created with an empty HET assembly source file named my_tut2.het. To begin adding code, press the Insert Instruction toolbar button (see Figure 11).

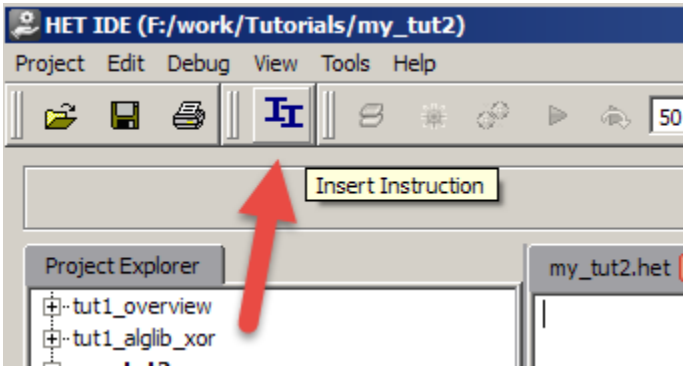


Figure 11. Insert Instruction Tool

6. Add an input capture instruction - PCNT. Select PCNT from the list of instructions that appears (see Figure 12) and press Insert.

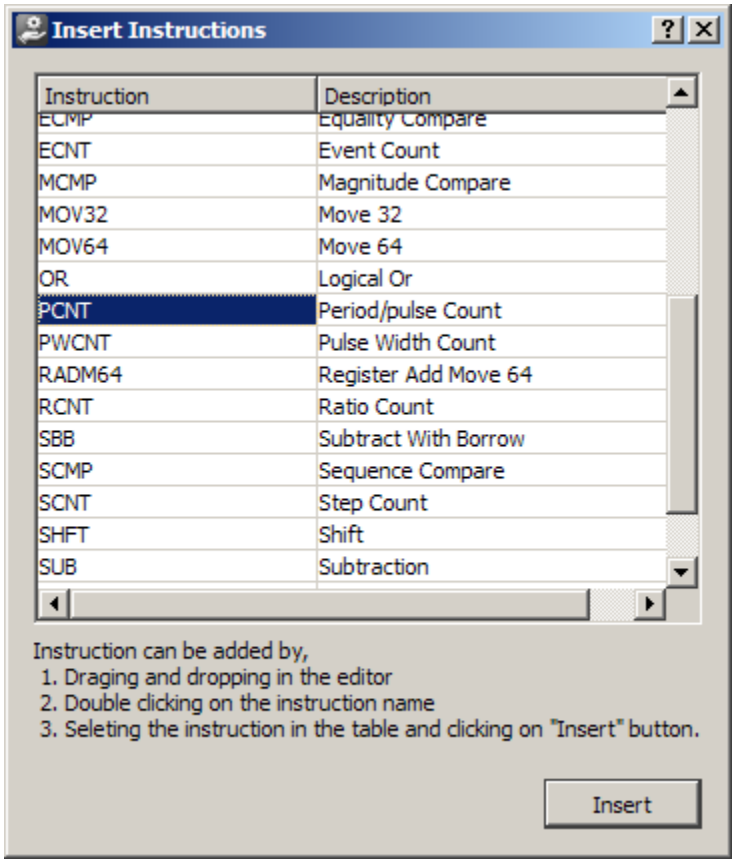


Figure 12. Insert Instruction Tool - Instruction List

7. After inserting the PCNT instruction, a wizard appears to help you fill out the required (red) and optional parameters for PCNT. In this case, the time between the rising and falling edges on pin 1 is captured. Therefore, select 'type' as 'RISE2FALL' and 'pin' as '1'. Create a label for this instruction so that other instructions can reference it, (label the instruction P01). Put a default value in the data field by setting period and data both to 0. Finally, after this instruction is finished, branch back to the beginning of the program. Even though the instruction has not been coded yet, label it C00; enter 'C00' in the 'next' field. All HET programs must branch back to the first instruction (at address 0) before the end of the loop resolution period, or an instruction overflow will occur. Your completed PCNT wizard will look like Figure 13. Note that many of the optional fields are left blank. This is ok, as the assembler assumes default values for all unspecified optional fields.

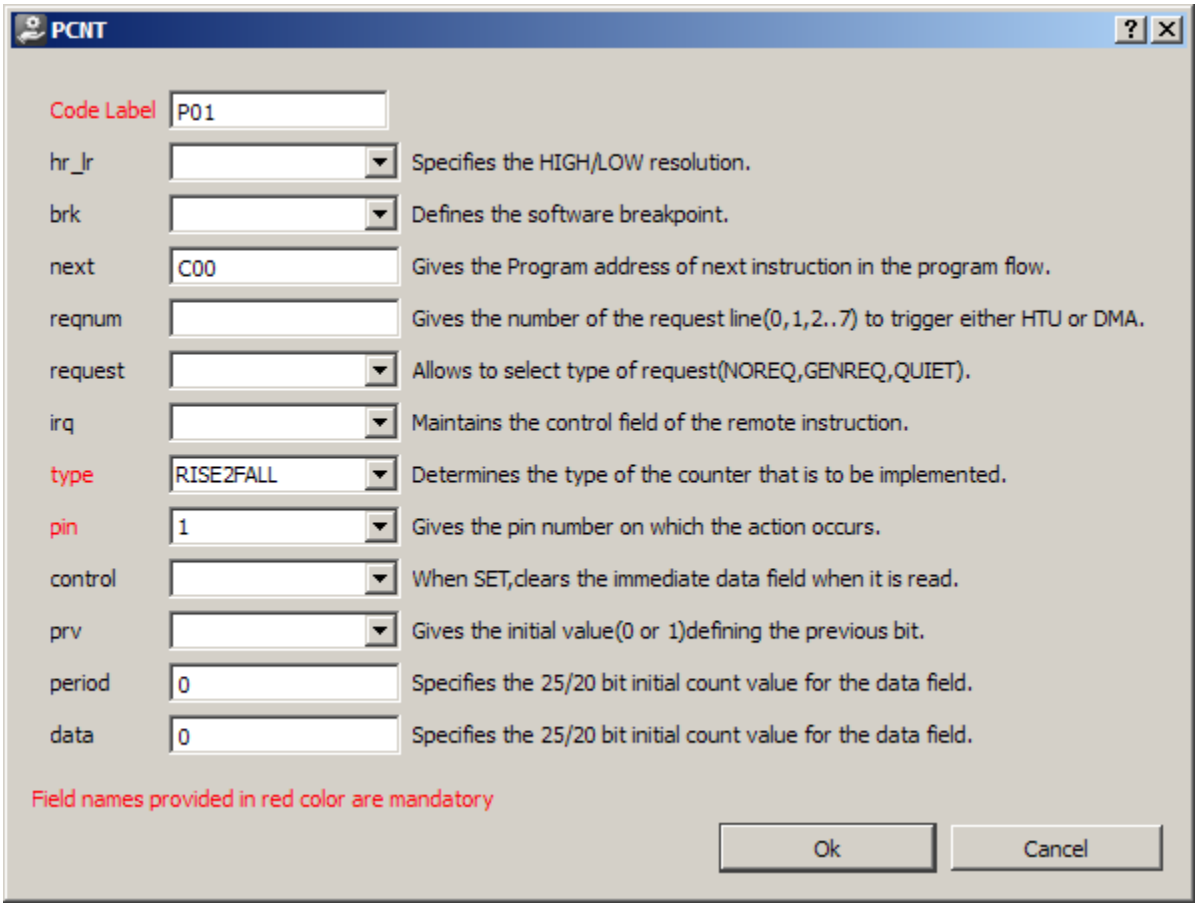


Figure 13. Completed PCNT Wizard, Required Fields in Red (optional fields may be left blank)

8. Press OK on the PCNT wizard. Note that an error message appears (see Figure 14). After every instruction insertion through the wizard, the HET IDE tries to assemble your program to make sure that the instruction you just inserted is free from errors. In this case, the error is expected. The PCNT instruction was told to branch to the instruction labeled "C00' next, but that instruction was not yet entered so the assembly will fail. This is OK so select "Yes".

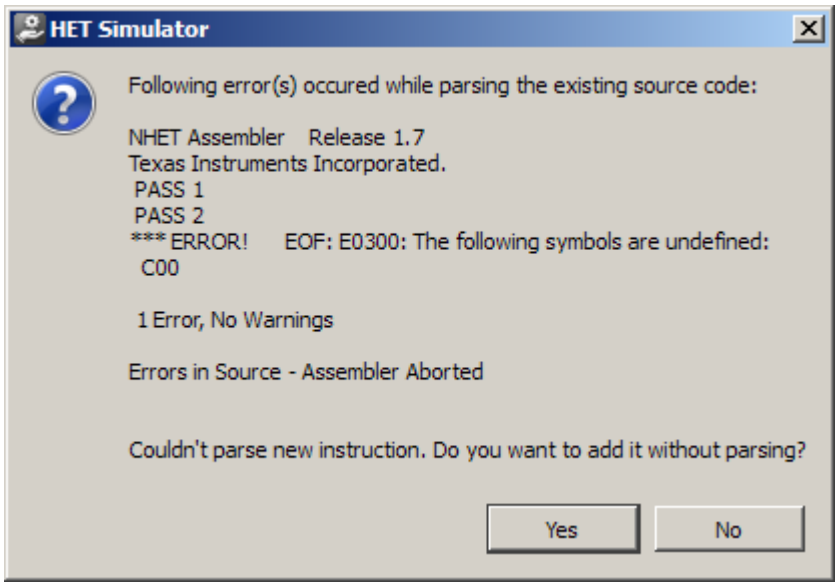


Figure 14. Error Because C00 is Not Yet Defined - This is Normal

9. Now, create the instruction for label C00. Use the Insert Instruction tool again, but select CNT from the instruction list instead of PCNT. The CNT instruction wizard should appear.

10. Populate the CNT instruction wizard (see Figure 15). The label should be C00. The next field could be blank, in which case the assembler will choose the next instruction in the program. In this case, it is explicitly specified as P01. Note that the next instruction is not *required* to be sequential and can be any instruction. We want the counter to count between 0, 1, 2, .. 10 and then roll over to begin at 0 again. So, a max value of 10 and an initial data value of 0 is placed in the wizard. Also, the count should be placed in register A. When you've completed populating the CNT instruction wizard, press OK.

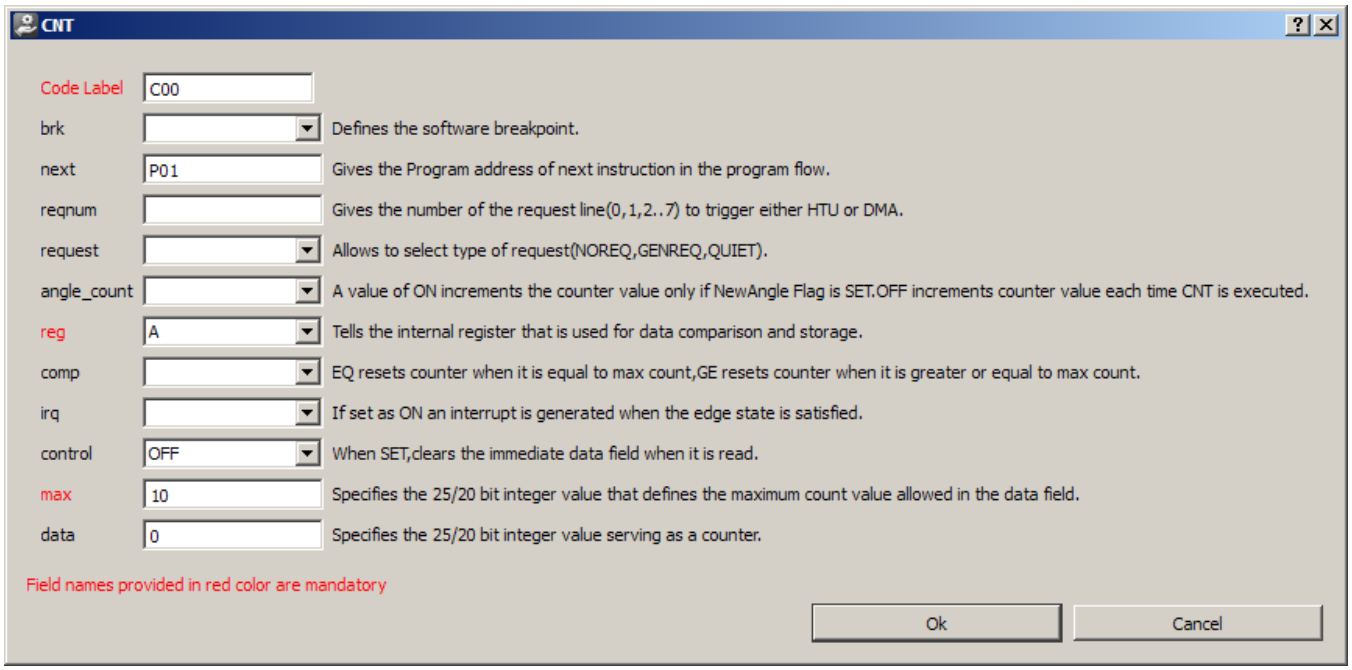


Figure 15. Completed CNT Instruction Wizard

11. You should now have a complete program consisting of two instructions (see Figure 16). Confirm that C00 is the first instruction in the program, if not swap C00 and P01 using the editor. This program periodically counts between 0...10 and uses this counter as timebase to measure the interval between rising and falling edges of pin 1.

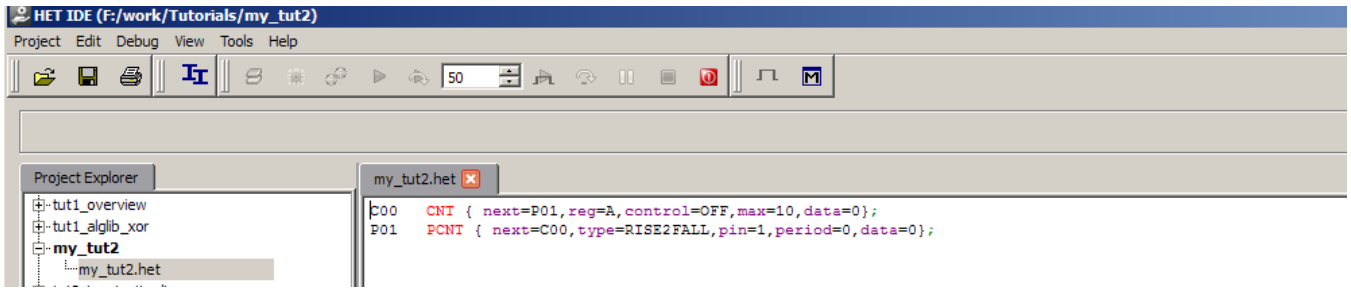


Figure 16. Complete Program for Tutorial 2

12. Assemble and Load your new program into the HET IDE's simulator.

13. To test a program that performs an input capture function, stimulus on the capture pin must be provided during simulation. There are several ways to provide stimulus; in this tutorial, the built-in Stimulus Creator is used. When the simulator starts, find the Stimulus Creator tab in the upper right hand pane of the HET IDE window (see Figure 17).

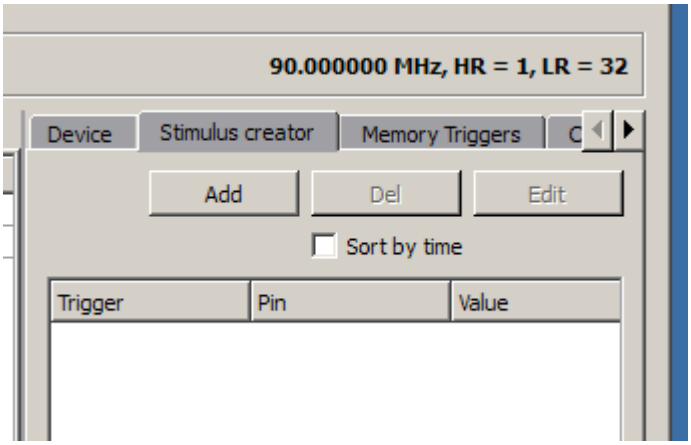


Figure 17. Stimulus Creator

14. Press the "Add" button to add input stimulus. A Pin Trigger to drive pin 1 high has been set up, starting at 250 ns into the simulation and repeating every 100 ns. To accomplish this, configure the pin trigger dialog (see Figure 18). Pay close attention to the units; the time unit defaults to 'ps' so be sure to change it to 'ns' in both places. When you have configured this trigger, press OK.

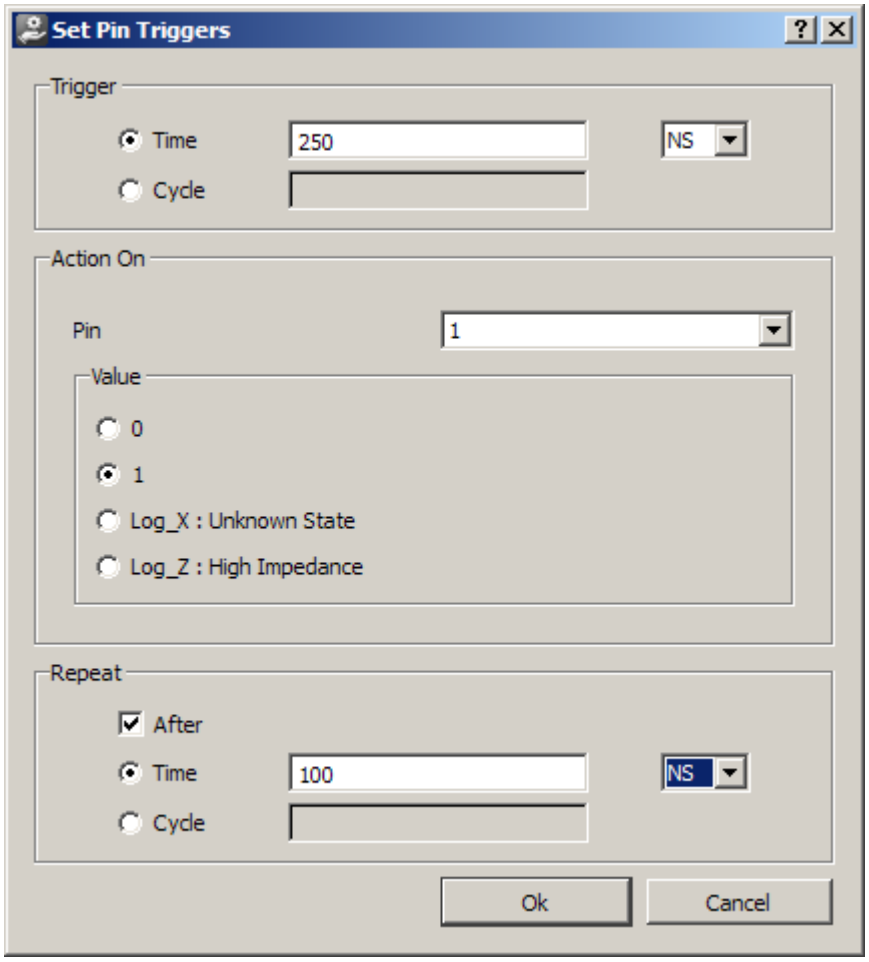


Figure 18. Pin Trigger to Drive Pin 1 High Periodically

15. Press "Add" a second time, to add a second pin trigger. Configure the trigger to drive pin 1 low again, starting at 300 ns into the simulation and repeating every 100 ns. The completed dialog for this trigger is shown in Figure 19

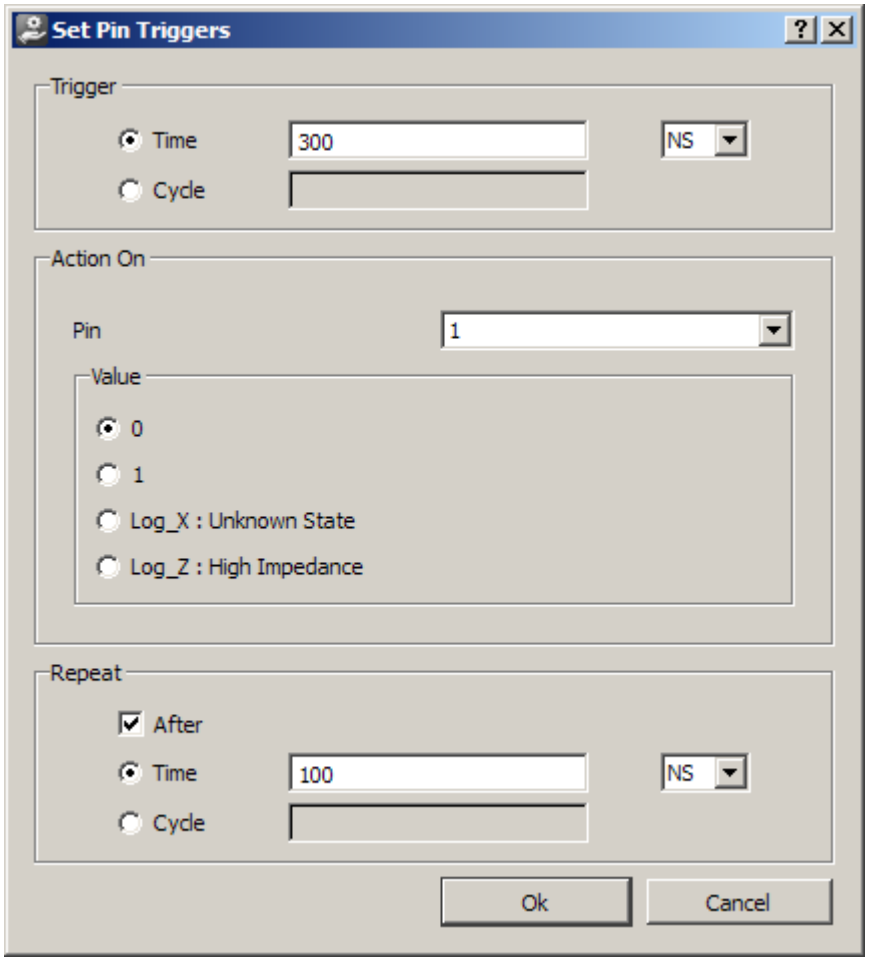


Figure 19. Pin Trigger to Drive Pin 1 Low Periodically

16. At this point, your stimulus creator should look like Figure 20. The combination of both pin triggers with repetitive periods, is to drive pin 1 high at 250 ns, low at 300 ns, and then continually repeat. This creates a toggling input stimulus on pin 1 with a period of 100 ns.

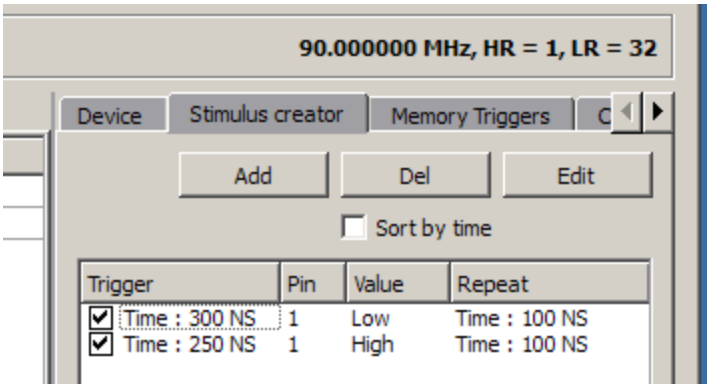


Figure 20. Completed Stimulus Creator Showing Both Triggers

17. Run the simulation for 50 loops, then switch to the Waveform Viewer and zoom so that the full simulation result is visible. You should see a waveform similar to Figure 21.

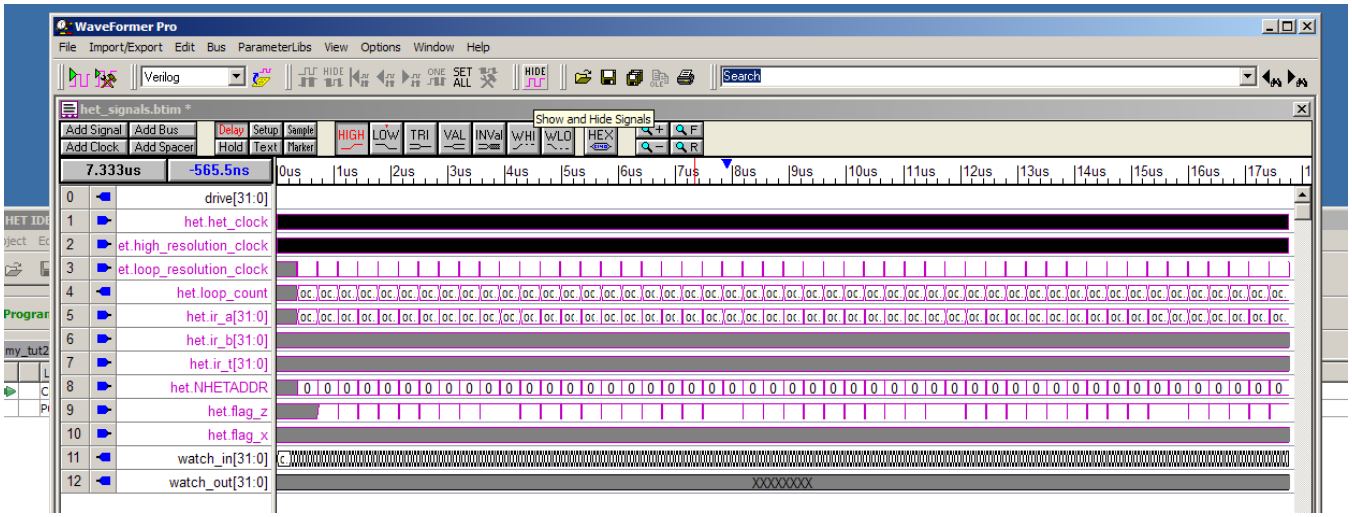


Figure 21. Result of Simulation

18. Zoom in so that you can see transitions on the watch_in[31:0] bus. You will see that Pin 1 is toggling every 100 ns, just as it was configured to do through the stimulus creator. (The value of the bus toggles between 0x02 and 0x00). A later tutorial explains how to expand the bus and show the signals individually.

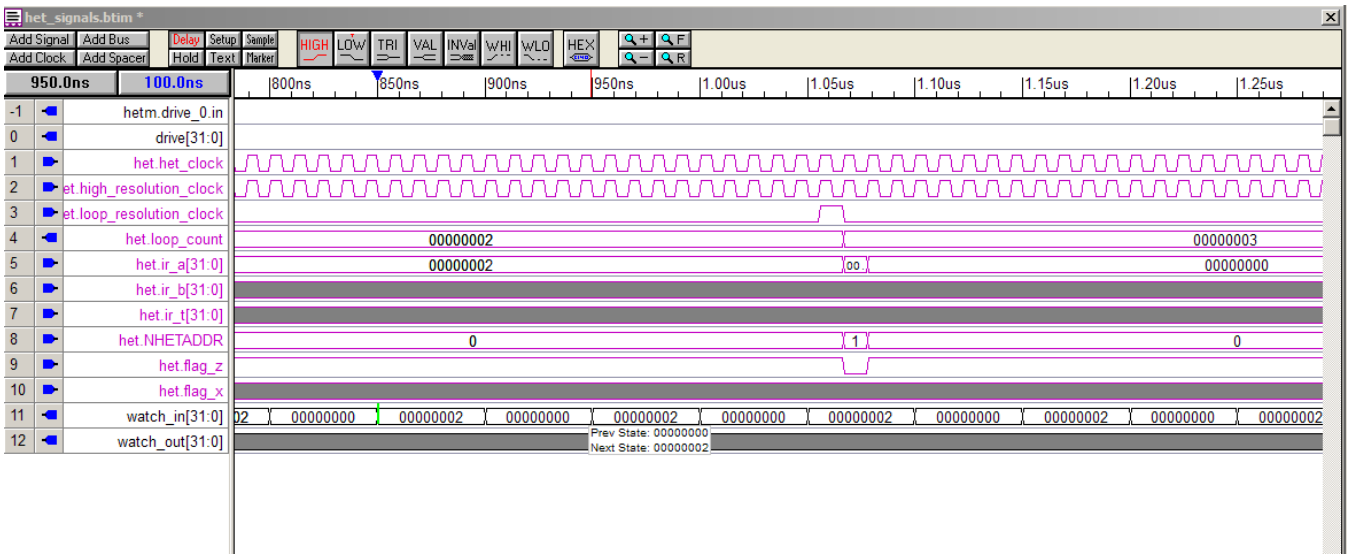


Figure 22. Zoom in of Simulation Result Waveform to Inspect Stimulus on watch_in[31:0]

2.3 Tutorial 3 — Changing Device and Clock Configuration

This tutorial covers the following features:

- Changing Device configuration
- Clock configuration

1. Open the HET GUI.
2. Open the Project:
   (a) Click Project → Open Project to open the project.
   (b) Open the *tut3_config.prj* file from <your_copy>/Tutorials/tut3_config.
3. Change Device Configuration:
   (a) Right click on tut3_config in Project explorer and select project properties (see Figure 23).

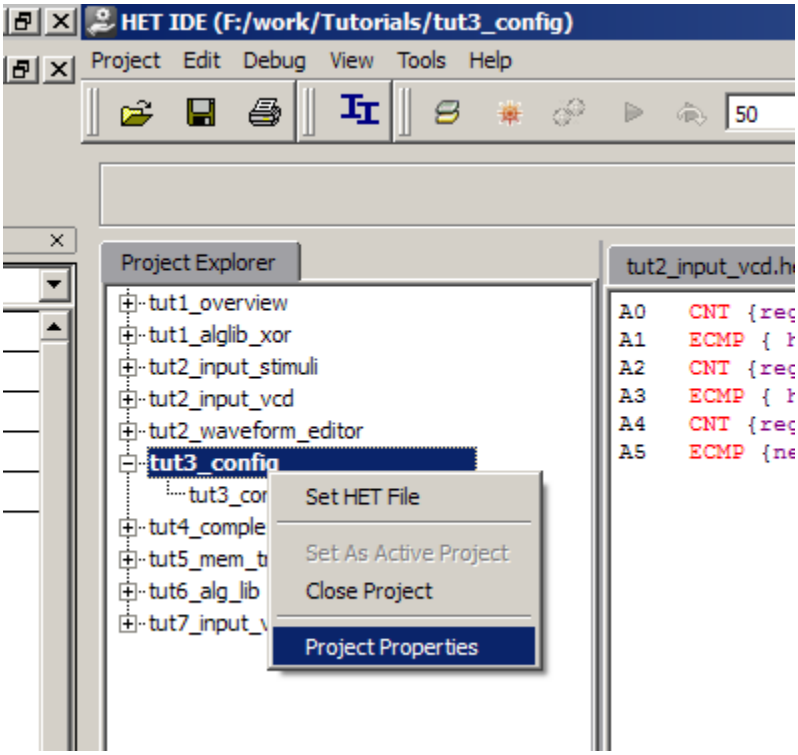


Figure 23. Opening Project Properties Dialog

(b) When the project properties dialog opens, make sure Device is selected in the left pane. Then, click on the "Change Device" button (see Figure 24).

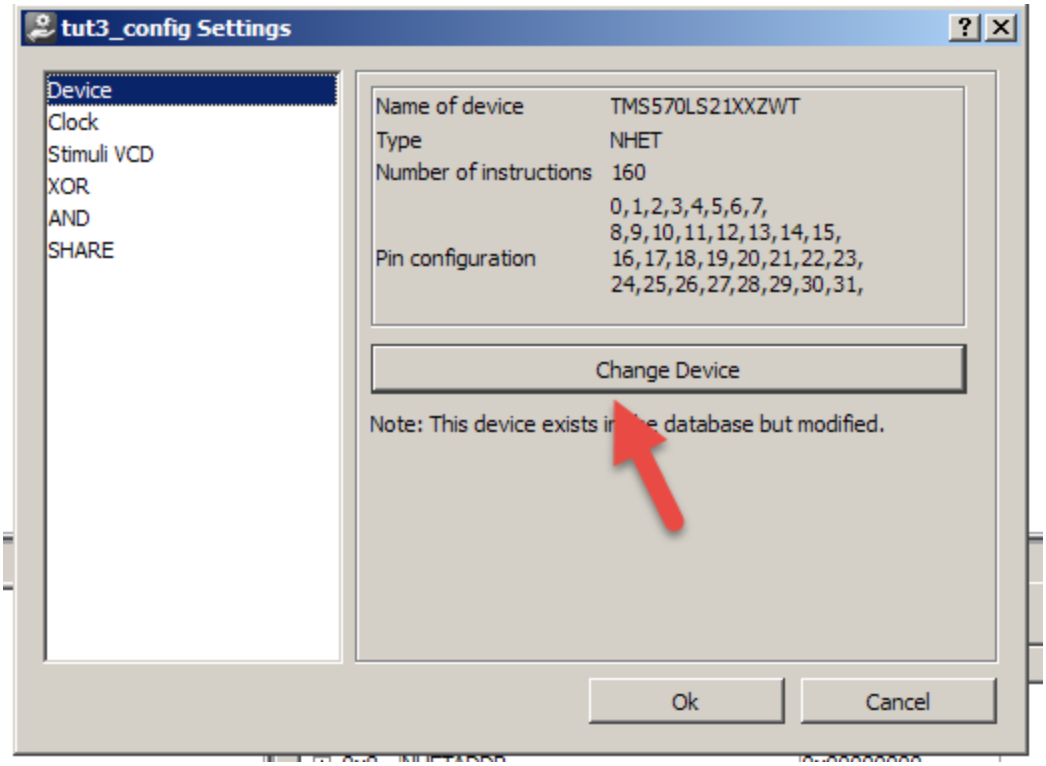


Figure 24. Changing the Device Configuration

(c) A list of supported Hercules devices will appear. Select the RM48LXXXZWT with N2HET and 160 words of instruction memory (see Figure 25). Then press OK.

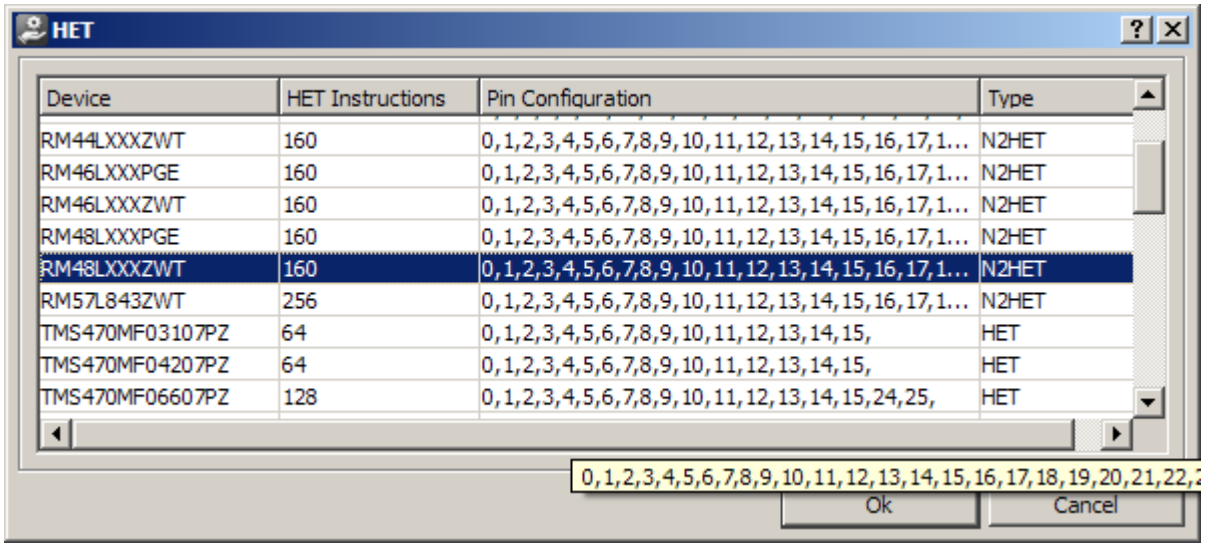


Figure 25. Selecting a Device From Device Configuration List

(d) The project properties dialog still shows the previous device. Press OK to apply the change and close the project properties dialog.

(e) Repeat step 3a, opening the project properties dialog again. This time you should see that your change to the device name has taken effect (see Figure 26).

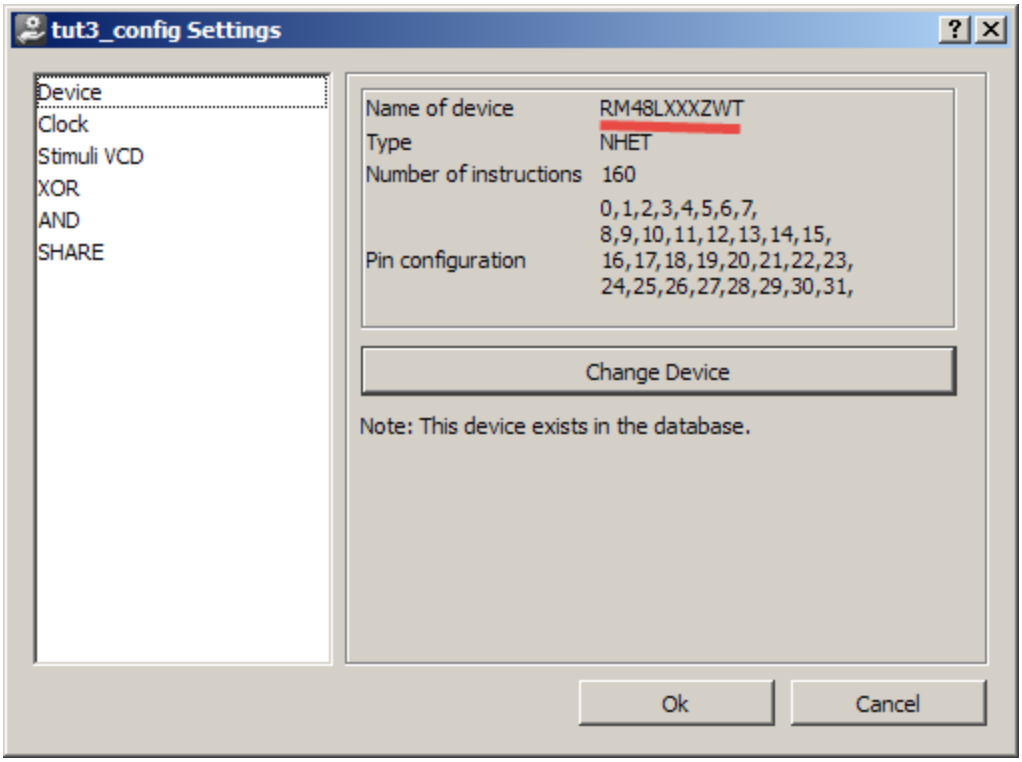


Figure 26. Project Properties Dialog Showing Newly Selected Device

4. Change the Clock Configuration:
   (a) With the project properties still open from step 3e, select "Clock" in the left hand pane.
   (b) Change the HET Clock Frequency to 110 MHz (9.09 ns period) (see Figure 27).
   (c) Change the Loop Resolution Prescale to 16 (see Figure 27). Leave the HR Prescaler at 1. With these settings, the Loop resolution clock period should be 16 * 9.09 ns = 145.5 ns.
   (d) Press the OK button to close the dialog.

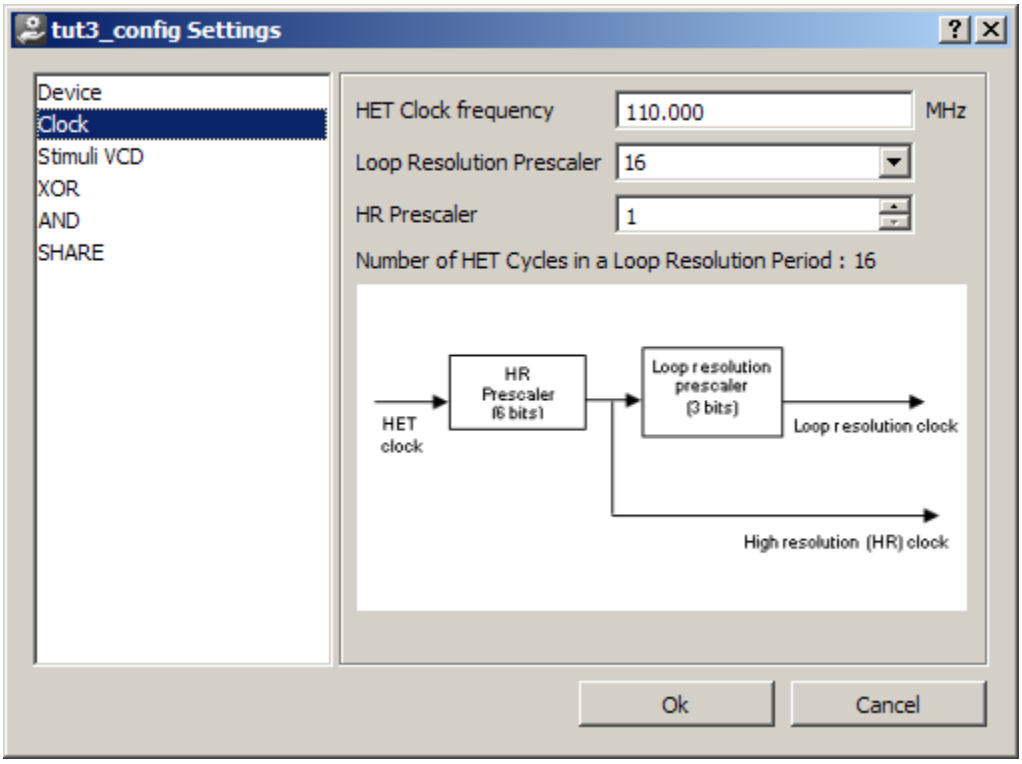


Figure 27. Changing the Clock Configuration to 110 MHz HET Clock and LR Clock Divider to 16

   (e) Press the "Assemble and Load" toolbar button to load the program into the simulator. Then, run the simulation for 20 loops.
   (f) Switch to the Waveformer window to inspect the clock frequencies. You can measure the clock period (1/frequency) in the Waveformer with these steps:
      (i) Left click on one edge of the clock signal in the waveform display so that a green marker appears on the edge.
      (ii) Move the mouse to hover over the next edge of the same direction (rise or fall).
      (iii) Read the delta time from the Waveformer display; it will be listed in blue above the signal list.

(g) Using the technique described in step 4f, verify that the HET clock matches your setting of 110 MHz (9.09 ns period) (see Figure 28).

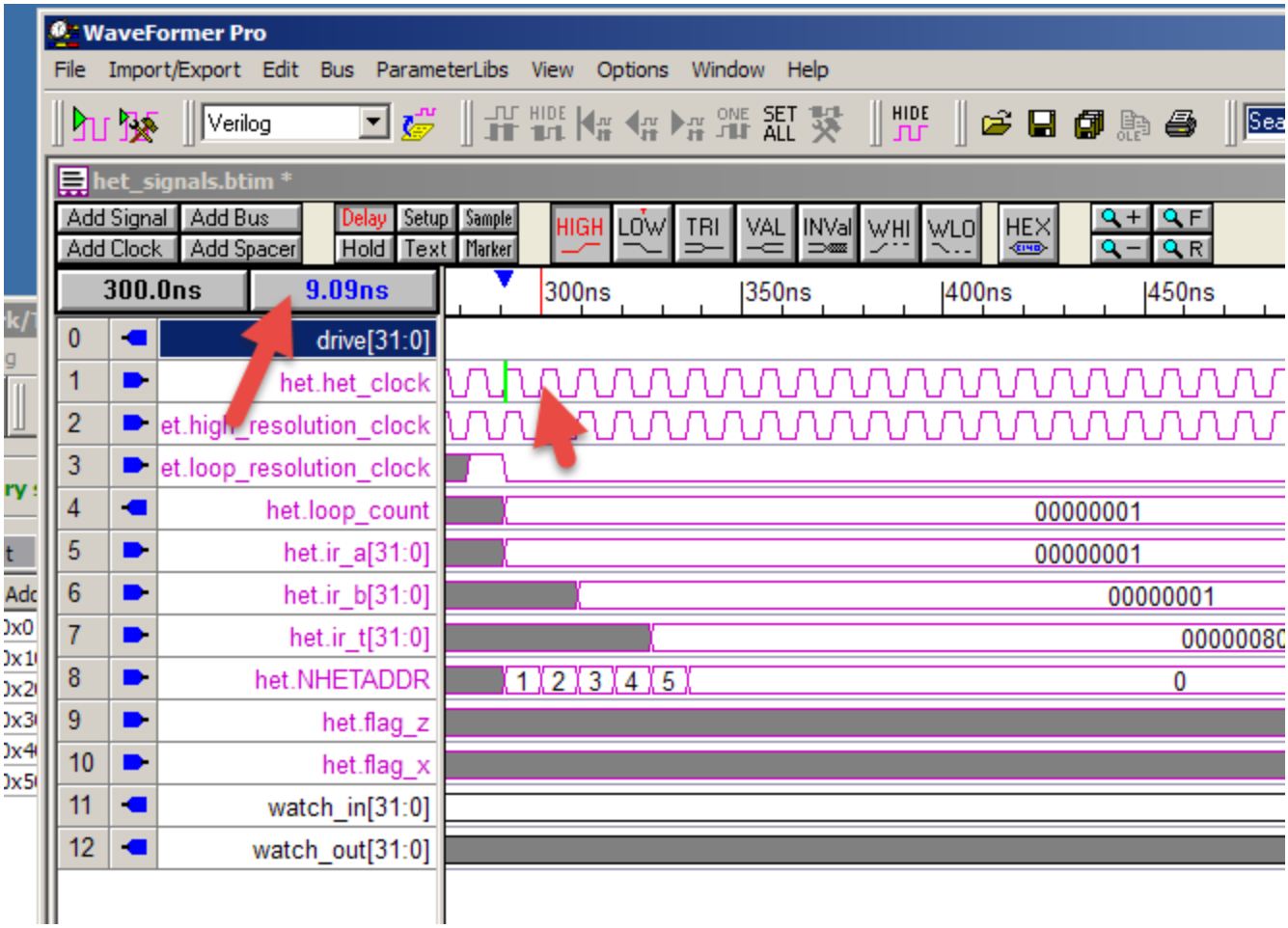


Figure 28. HET Clock Period Measurement

(h) Using the technique described in step 4f, verify that the HET loop resolution clock prescaler is 16 (145.5 ns) by measuring the loop resolution clock period (see Figure 29).

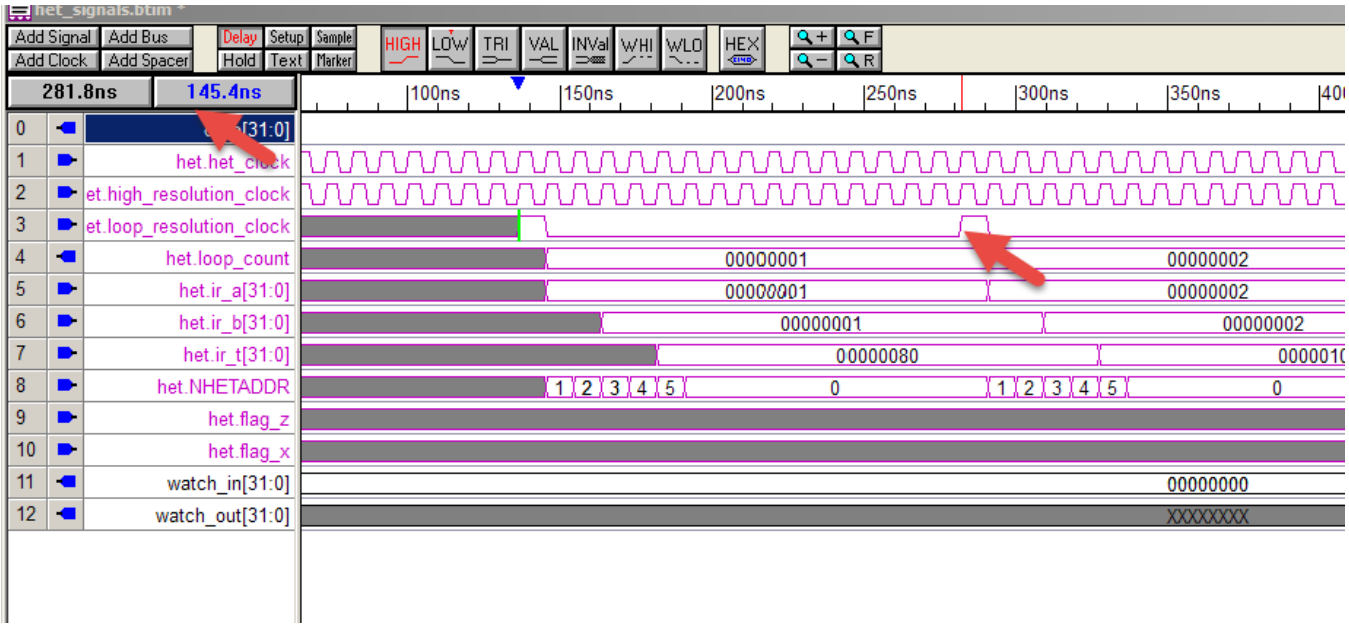


Figure 29. HET Loop Resolution Clock Period Measurement

2.4 Tutorial 4 — Complex Breakpoints

This tutorial covers the following features:

- Complex breakpoints

1. Open the HET GUI.
2. Open the Project:
   (a) Click Project → Open Project to open the project.
   (b) Open the *tut1_overview.prj* file from <your_copy>/Tutorials/tut1_overview folder .

   NOTE: Do not use the project in the tut4_complex_breakpoint folder.
3. Assemble the program (Debug → Assemble).
4. Load the HET program (Debug → Load HET Program).

5. Select the "Complex Breakpoints" tab in the top right pane of the HET IDE (see Figure 30).

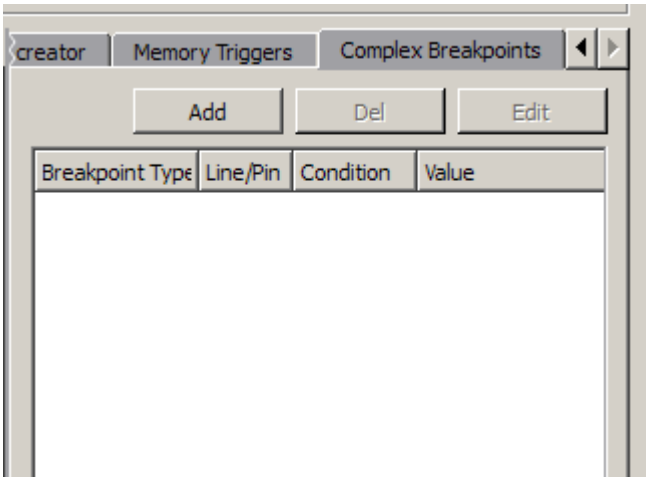


Figure 30. Complex Breakpoints Tab

6. Press the "Add" button to open the Set complex Breakpoints dialog.
7. Select a "Pin" type breakpoint, then choose Pin Number '7' and a Level of '1' (see Figure 31).

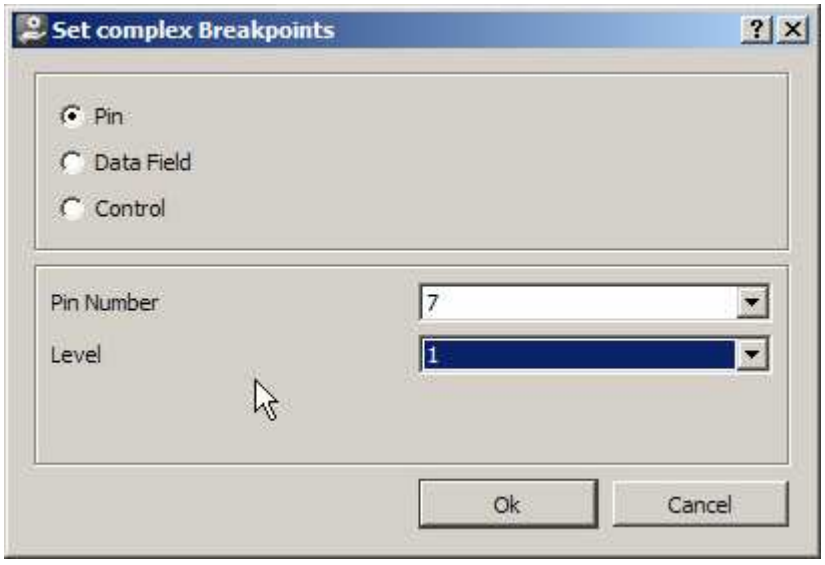


Figure 31. Configuring a Complex Breakpoint to Stop When Pin 7 is High

8. Press 'Ok' and confirm that the breakpoint is displayed in the Complex Breakpoints tab. You should see Breakpoint Type listed as "Pin", Line/Pin as 7, and Value as 1 (see Figure 32).

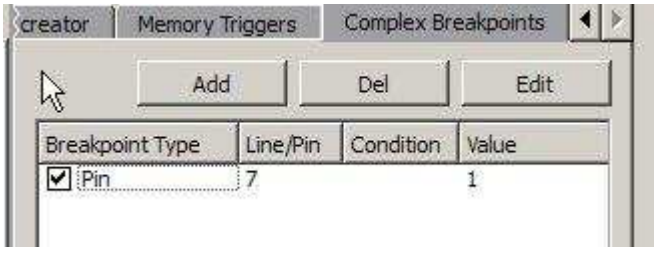


Figure 32. Review Complex Breakpoint Set for Pin 7 High

9. Run the program.

10. The program should stop with a message "Hit a complex breakpoint at pin 7" (see Figure 33).

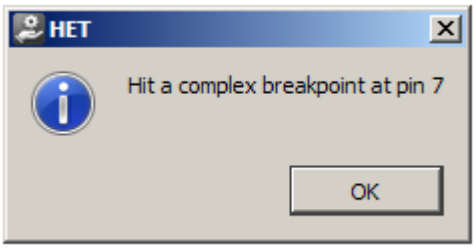


Figure 33. Complex Breakpoint Hit

11. Switch to the Waveform Viewer. Zoom full, then zoom out once, so that the end of the simulation can be seen within the waveform window. Hover the mouse pointer over the very last edge of the watch_in[31:0] bus. You will see that the value has changed from 0x21 to 0xA1 exactly where it was stopped with the breakpoint. The difference between 0x21 and 0xA1 is that bit 7 is set. Bit 7 contains the value for pin 7. This shows that the breakpoint has stopped the HET simulation when bit 7 became 1, as it was configured (see Figure 34).

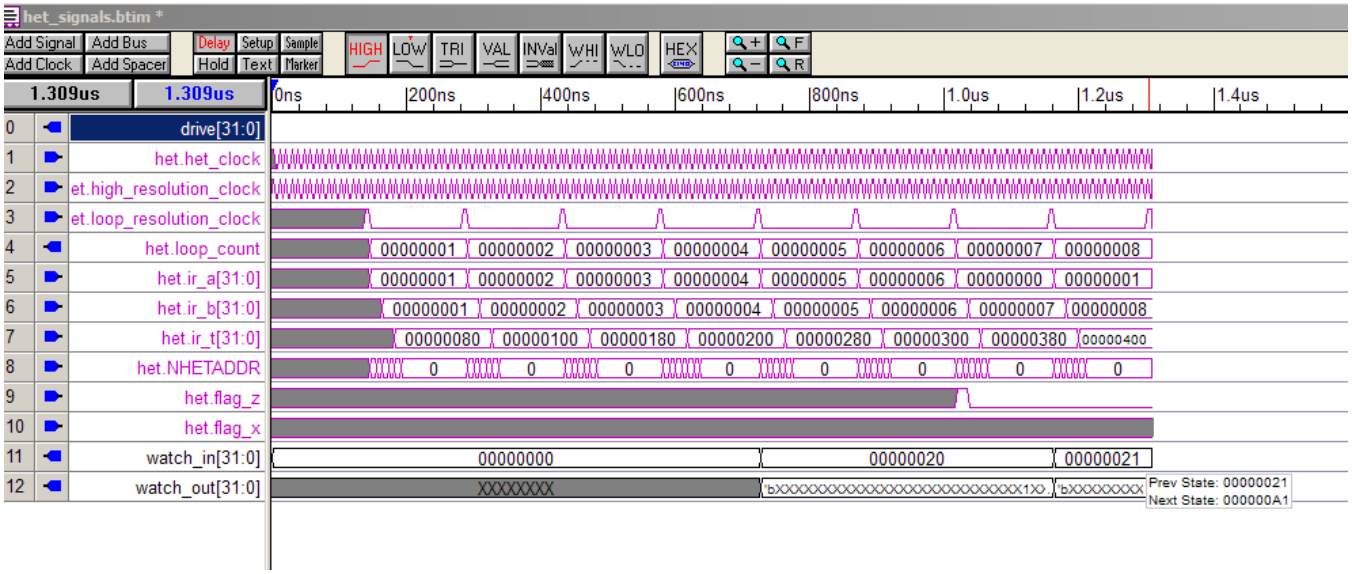


Figure 34. Stopped When Pin 7 Changes to 1 (watch_in[31:0] Changes From 0x21 to 0xA1)

12. Using the "DEL" button in the Complex Breakpoints tab, Delete the complex breakpoint for Pin 7.

2.5 Tutorial 5 — Memory Triggers

This tutorial covers the following features:

- Memory triggers
- Tracing the waveform

1. Open the HET GUI.
2. Open the Project:
   (a) Click Project → Open Project to open the project.
   (b) Open the *tut5_mem_trig.prj* file from <your_copy>/Tutorials/tut5_mem_trig.
   (c) Outside the HET IDE, open the folder <your_copy>/Tutorials/tut5_mem_trig in Windows Explorer
   (d) Delete the *'het_signals.btim'* file; it is outdated. The HET IDE automatically refreshes this file with a clean copy before starting a simulation.
3. Switch back to the HET IDE.
4. Assemble the program (Debug → Assemble).
5. Load the HET program (Debug → Load HET Program).
6. Set Memory Triggers:
   (a) Select the Memory Triggers tab from the top-right pane of the HET IDE window (see Figure 35).

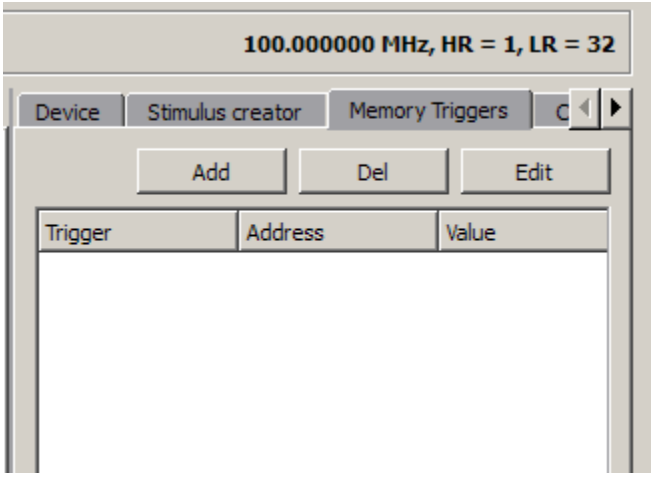


Figure 35. Memory Triggers Tab

(b) Click Add. The Memory Trigger window appears (see Figure 36).

(i) In the Trigger pane, enter value as 2500 and change units to NS by clicking the arrow.

(ii) In the Action pane, enter the address to be modified as 0x40, and enter the value to be written as 13.

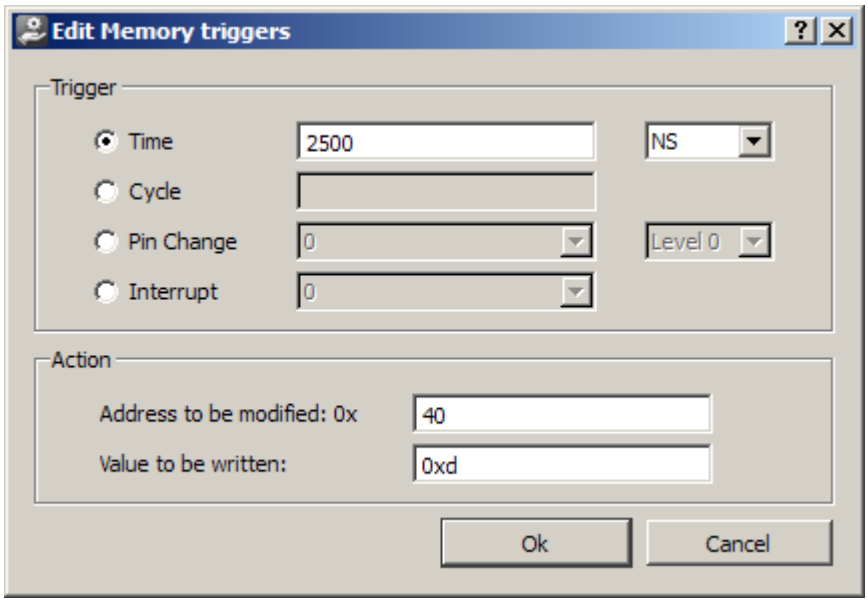


Figure 36. Memory Trigger Editor

(iii) Click OK.

(c) Open Waveform Wizard (see Figure 37).

(i) Go to Tools → Waveform wizard.

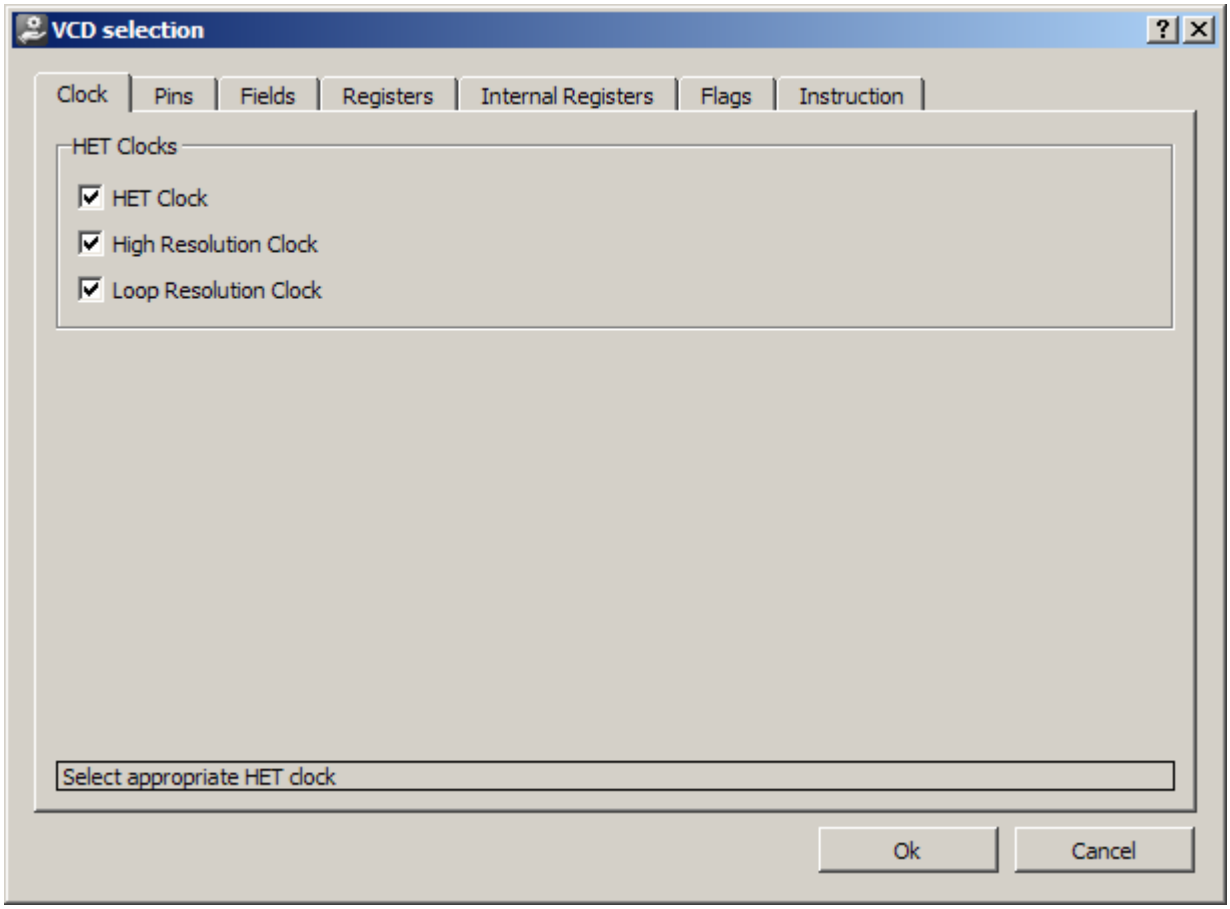


Figure 37. Waveform Wizard

(ii) Go to the Instructions tab. Select 0x40 in the address drop-down list, check Program in instructions fields, and click add (see Figure 38).

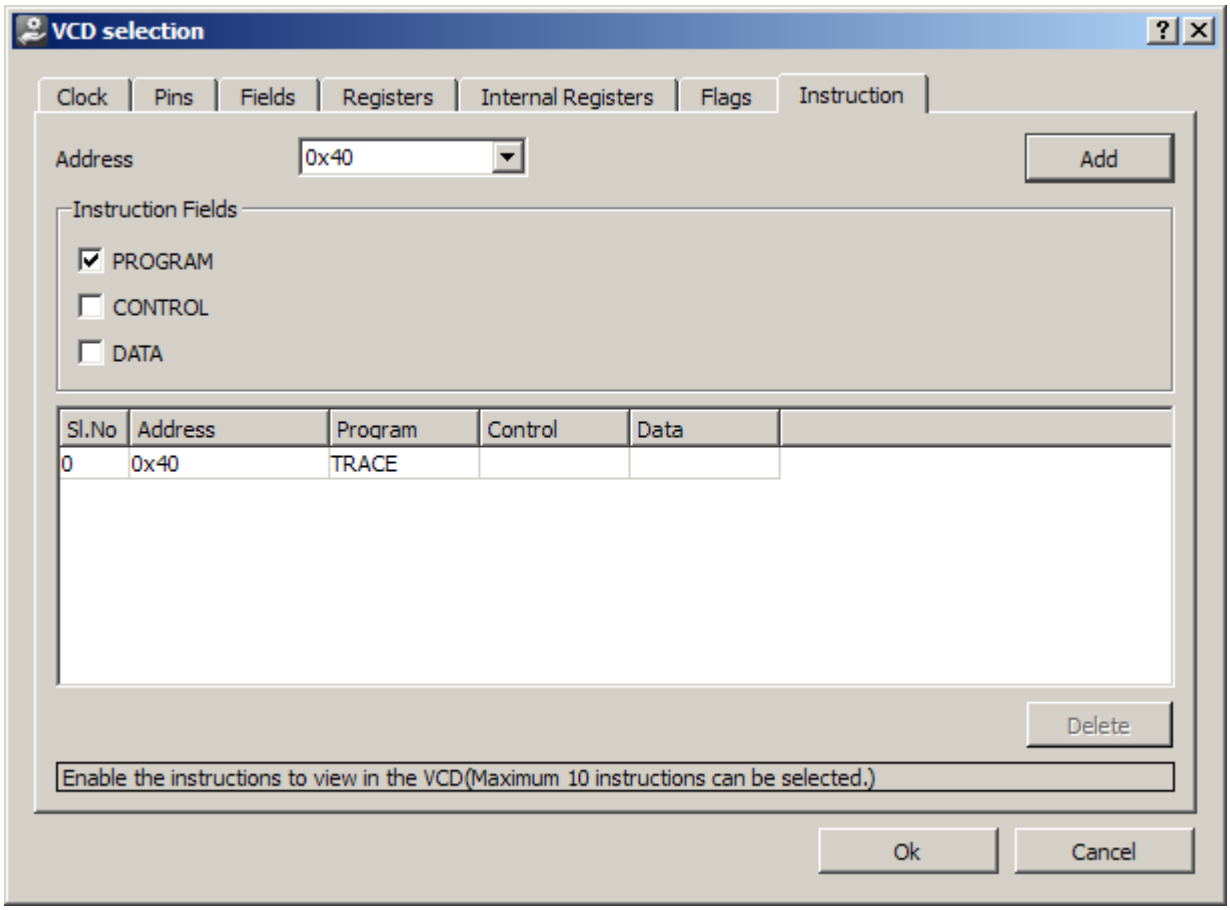


Figure 38. Adding an Instruction Trace to the Waveforms Output by the HET Model

(iii) Make sure that you see Sl. No 0 populated with address 0x40 (see Figure 38), then click on OK.

7. Switch focus to the Synapticad Waveform Viewer and press the Add Signal button (see Figure 39).

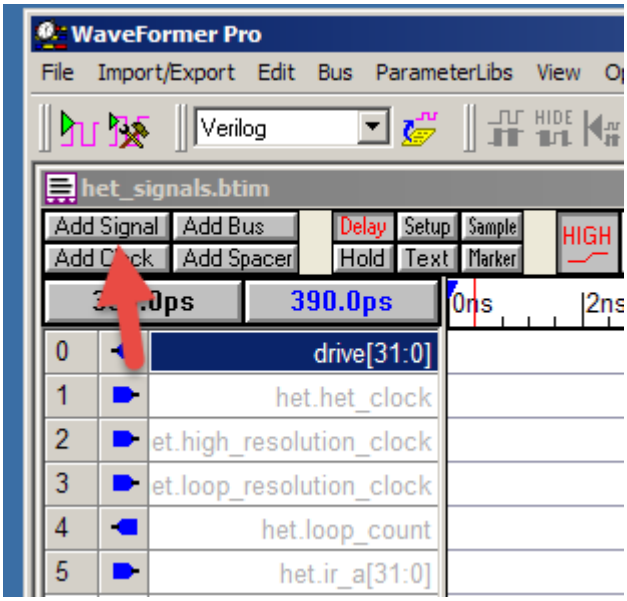


Figure 39. Adding a New Signal to the Waveform Viewer

8. The newly added signal appears at the bottom of the signal list (see Figure 40).

Figure 40. Double Click to Edit the New Signal

9. Double click on the newly added SIG0 to open the Signal Properties Dialog. Then make the changes shown in Figure 41.
   (a) Change the signal name to 'het.mem_0x40_program'. This name must be entered exactly, as it must match the name exported by the HET simulator.
   (b) Change the signal type to Watch.
   (c) Change the signal direction to Input.
   (d) Change the signal to a 32-bit Bus, with MSB=31 and LSB=0.
   (e) Press OK to save the changes.

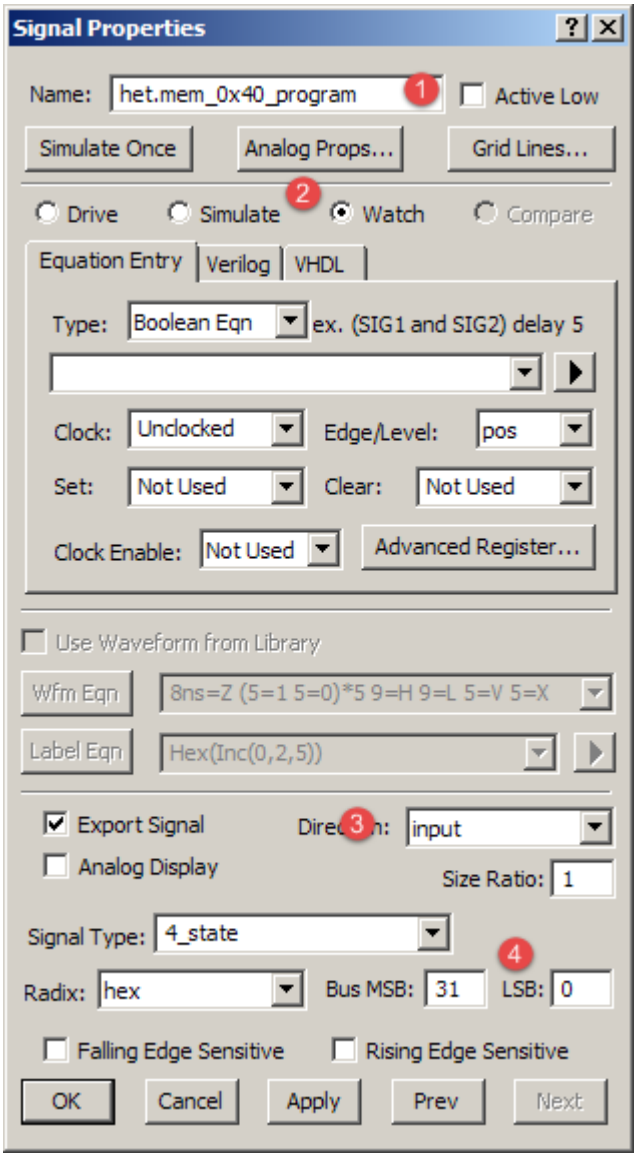


Figure 41. Signal Properties Corresponding to Instruction Trace at Address 0x40

10. Switch focus back to the HET IDE.
11. Run the program for 20 loops.

12. In the Program field of the Memory window at address 0x40, change to 13 (0xD) when time>=2500 ns (see Figure 42).

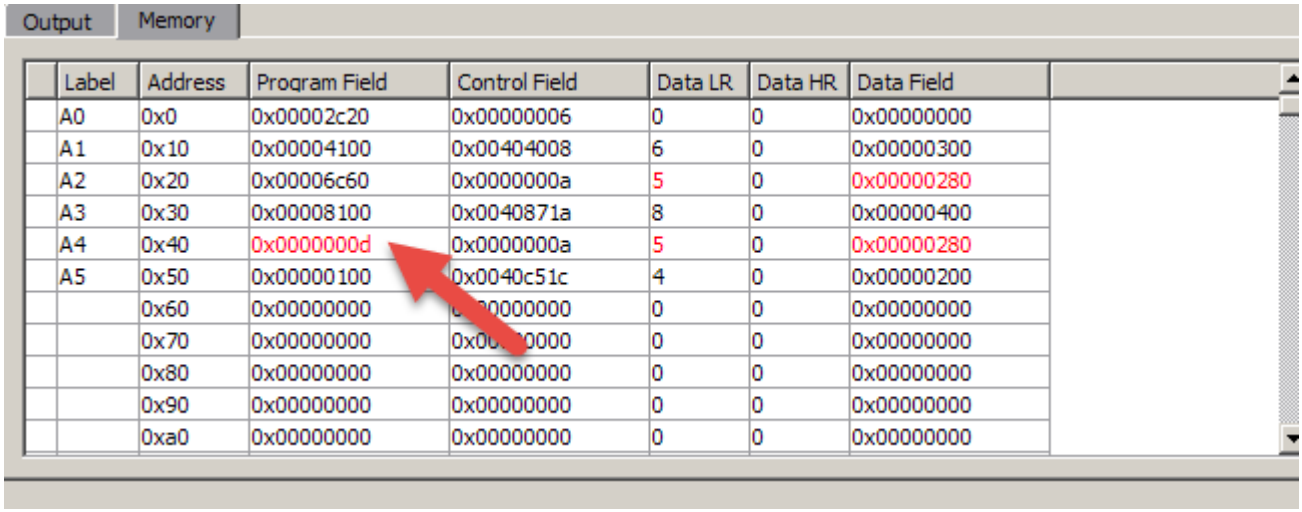


Figure 42. Result of Memory Write (corrupted program field) can be Seen After Simulation in HET IDE Memory Viewer

13. Switch focus to Synpaticad Waveform Viewer to see the resulting waveform with trace.
14. After zooming, your waveform should appear like the one shown in Figure 43. Note that the memory trigger, that you setup to over-write the instruction at address 0x40, can be seen taking effect in the trace of HET.mem_0x40_program[31:0] at 2500 ns (2.5 µs), which matches the trigger time you configured. The value 13 is displayed in hexadecimal format in the waveform listing as 0xD.

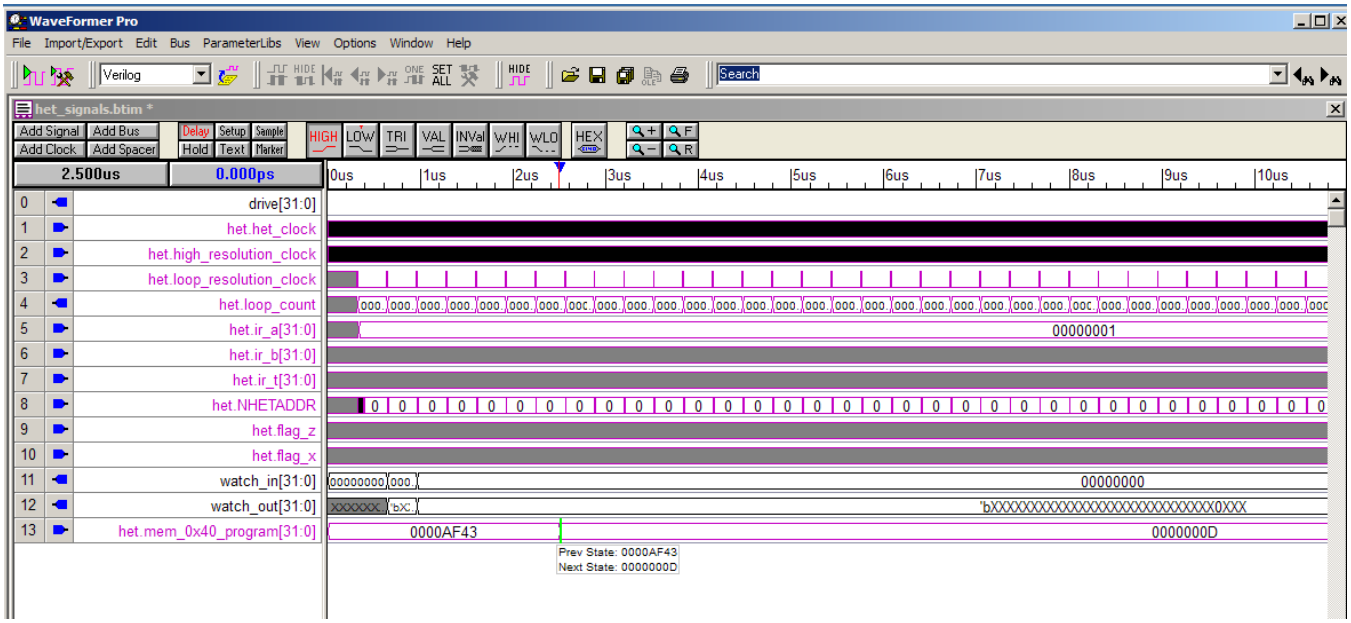


Figure 43. Waveform at Completion of Tutorial 5 (with added instruction trace)

15. A complete list of the signal names that can be exported by the HET simulator and traced with the Synapticad Waveform Viewer can be found in appendices of the *HET IDE User's Guide* (SPNU483). Remember to match the signal names exactly, as the HET simulation model has already been compiled and its port names are fixed.

2.6 Tutorial 6 — Algorithm Library and XOR Configuration

This tutorial covers the following features:

- Inserting Algorithm from Algorithm library
- Selecting Algorithm family
- Parameter values selection
- XOR configuration

1. Open the HET GUI.
2. Open the Project:
   (a) Click Project → Open Project to open the project.
   (b) Open the *tut6_alg_lib.prj* file from <your_copy>/Tutorials/tut6_alg_lib.
   (c) Outside the HET IDE, open the folder <your_copy>/Tutorials/tut6_alg_lib in Windows Explorer
   (d) Delete the *het_signals.btim* file; it is outdated. The HET IDE automatically refreshes this file with a clean copy before starting a simulation.
3. Switch focus back to the HET IDE. You should see that the source file for this tutorial begins with an empty file (except for the initial comment). Functionality is added by using the Algorithm Library.

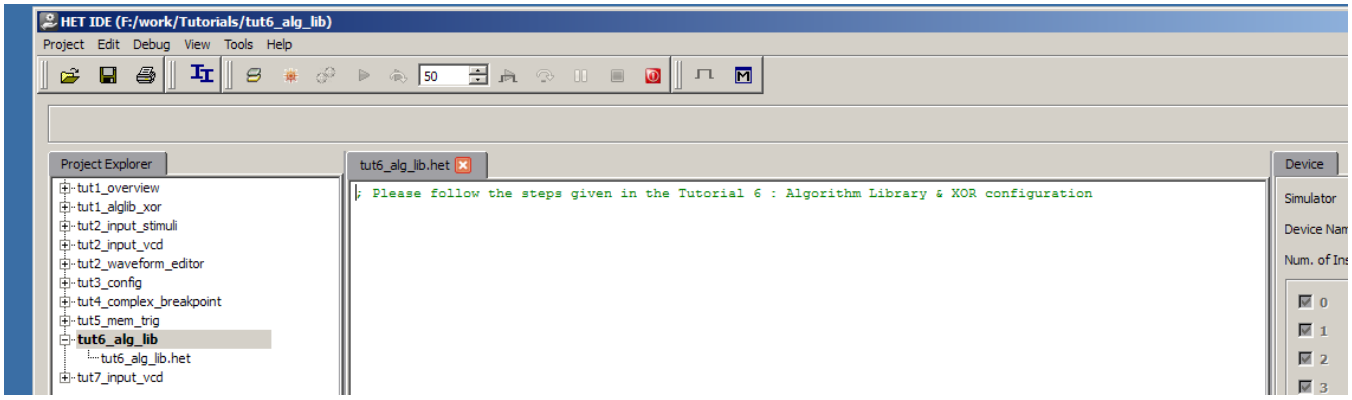


Figure 44. Tutorial 6 Starts With a Blank HET Program

4. Insert new algorithm:
   (a) Click Edit → Insert Algorithm (see Figure 45).

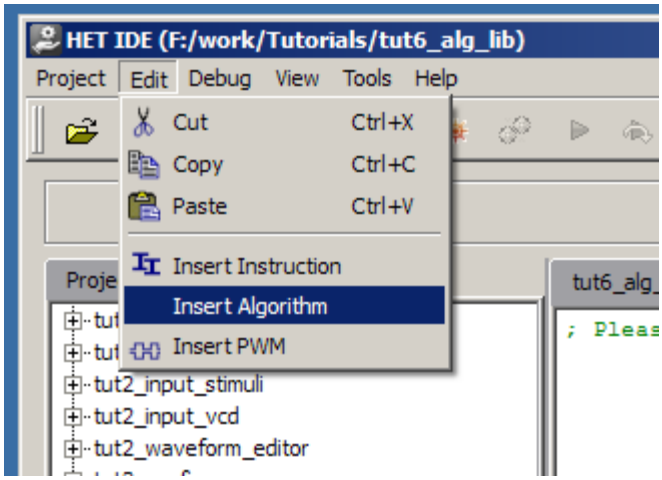


Figure 45. Edit → Insert Algorithm to Open the HET Algorithm Library

(b) Select Standard Output Examples (see Figure 46).

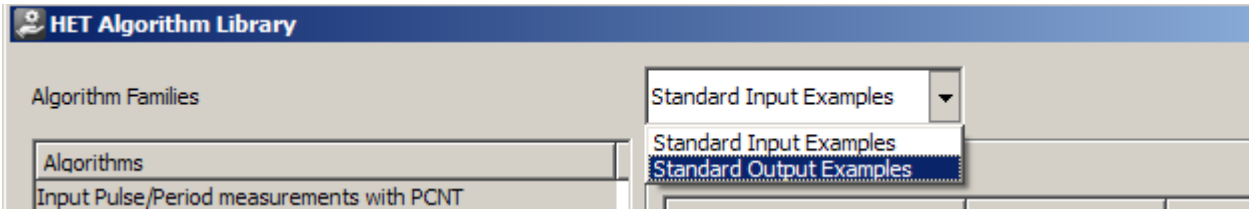


Figure 46. Select Standard Output Examples for PWM Algorithms

(c) Select PWM, three channels with different frequencies, using MCMP (see Figure 47).

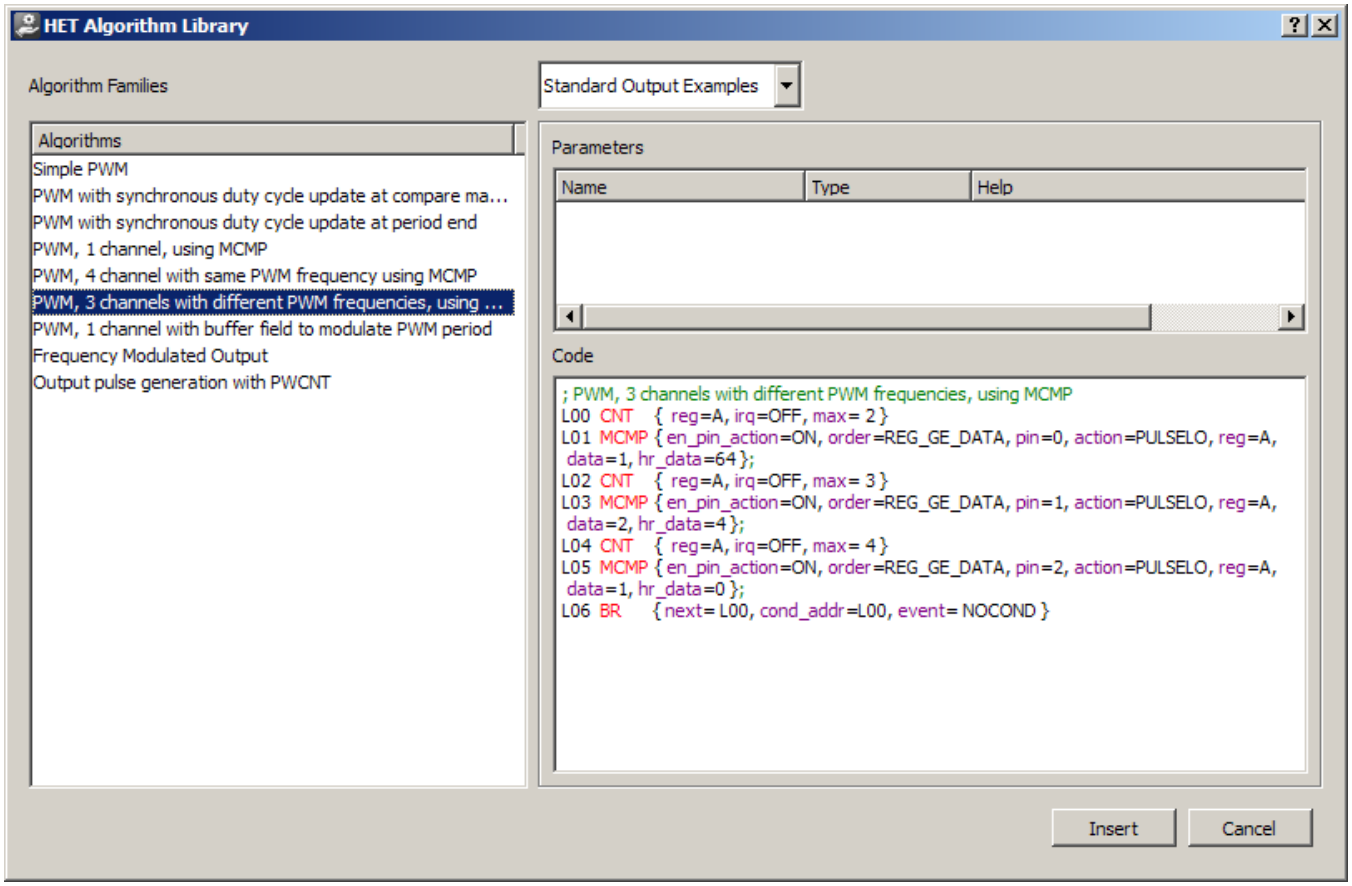


Figure 47. Three Channel PWM Function From the Algorithm Library

(d) Click Insert.

(e) Close the Algorithm Library window.

(f) You should see that the three channel PWM algorithm has been inserted into your HET assembler program (see Figure 48).

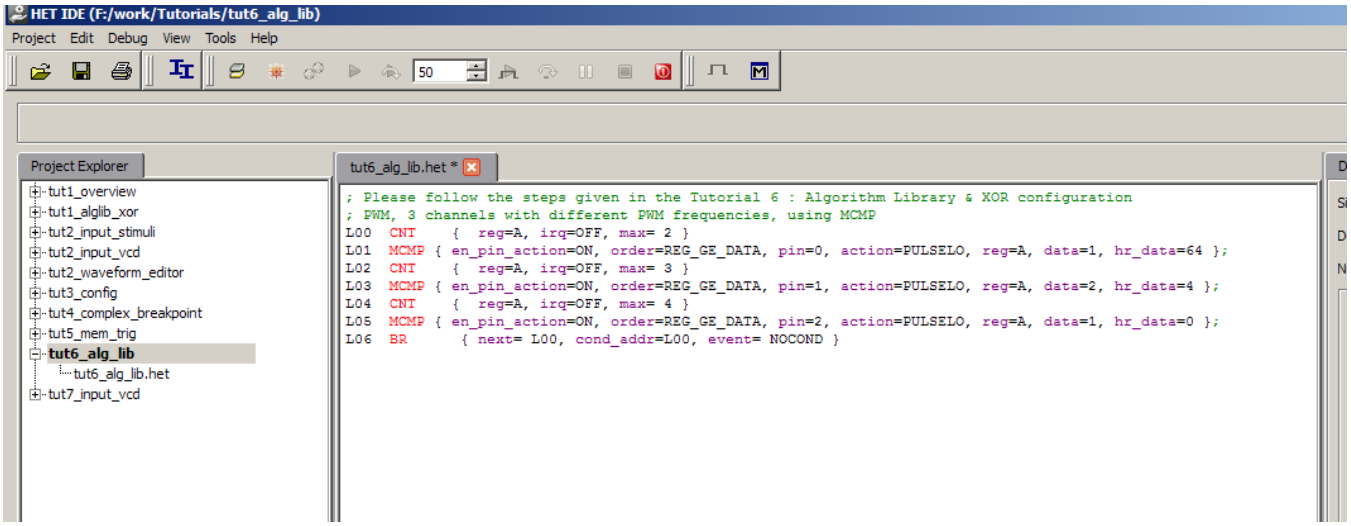


Figure 48. Algorithm Library

(g) Close the HET Algorithm Library wizard.

(h) Save the HET file.

5. Assemble the program.
6. Load the HET program.
7. Run the program for 50 loops.
8. View the Waveform and use the Magnifer + F to zoom to show the full waveform. Your display should look like Figure 49. There is clearly activity on the HET output pins (watch_out[31:0]), but it is difficult to see if the activity is on pins 0,1, and 2 as expected.

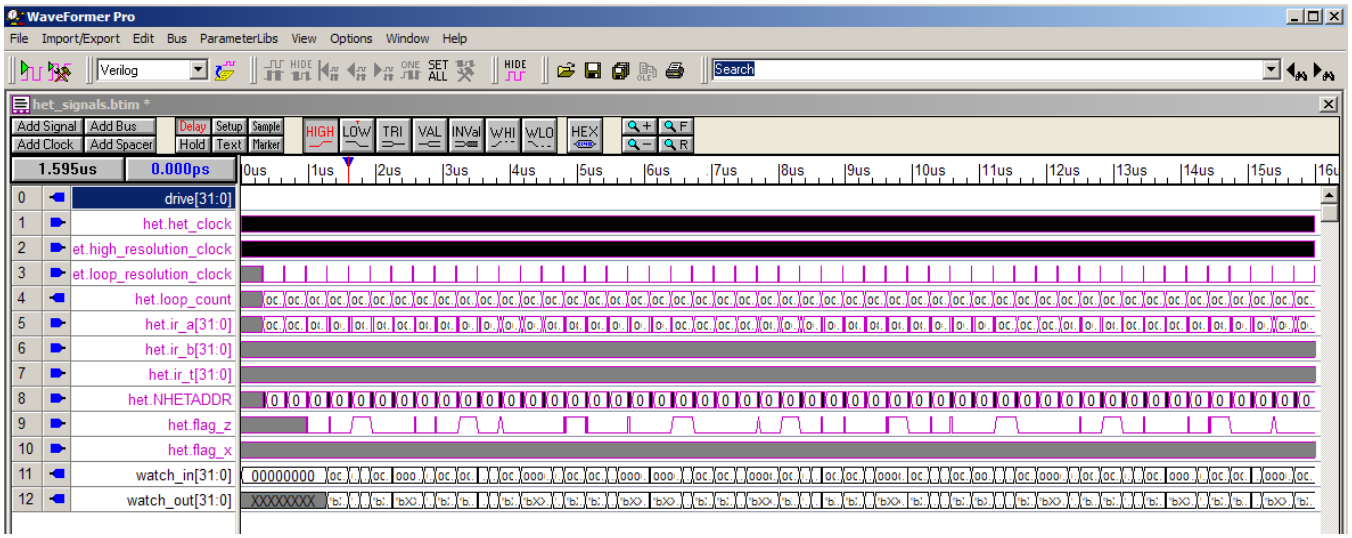


Figure 49. Waveform Without XOR Configuration

9. Expand the 'watch_out[31:0]' bus by right clicking on the signal name and selecting 'Show Bus Member Signals' (see Figure 50).

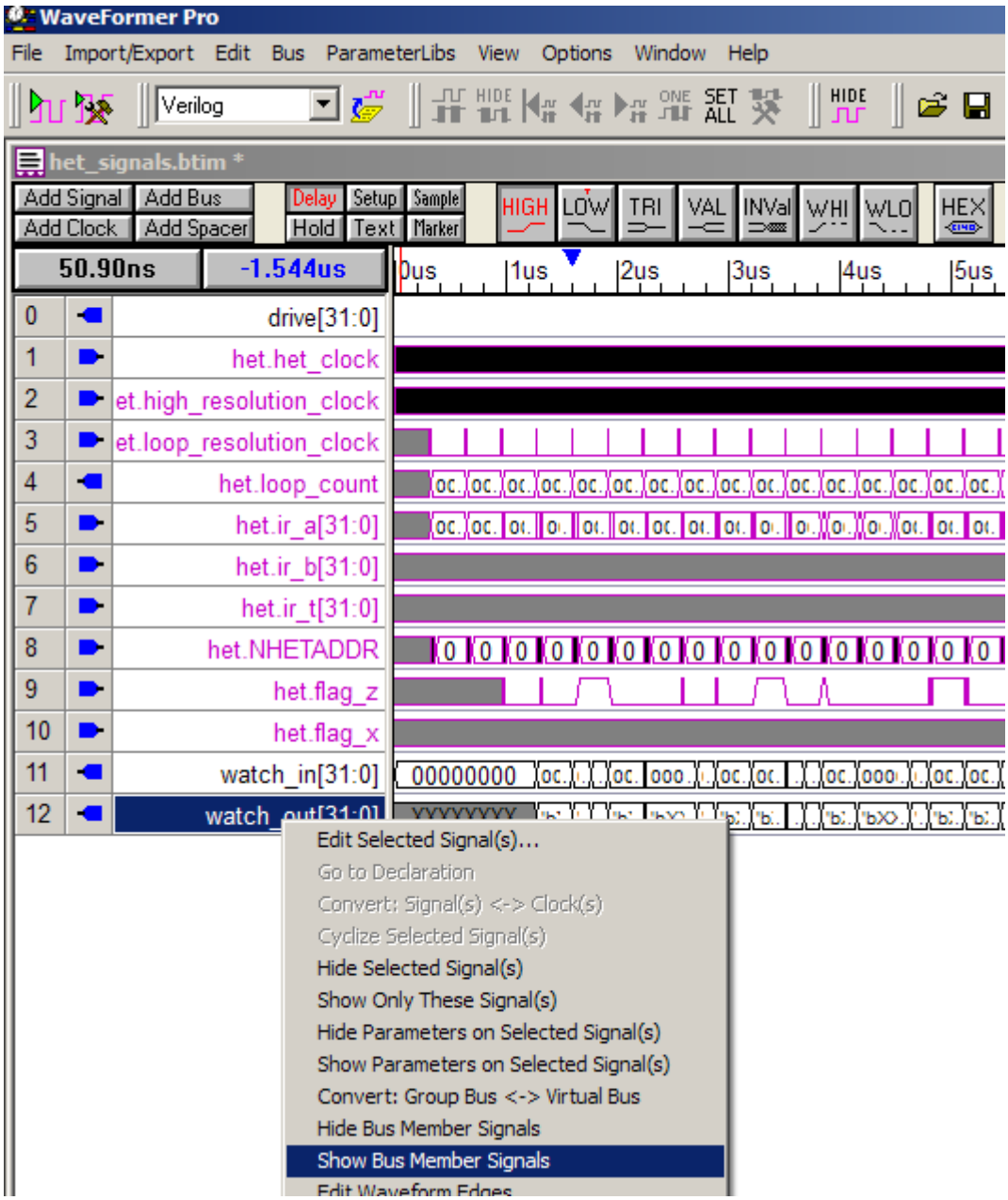


Figure 50. Expanding the watch_out[31:0] Bus to View Individual Signals

10. You should now see the individual signals for HET ports 0, 1, and 2 (see Figure 51).

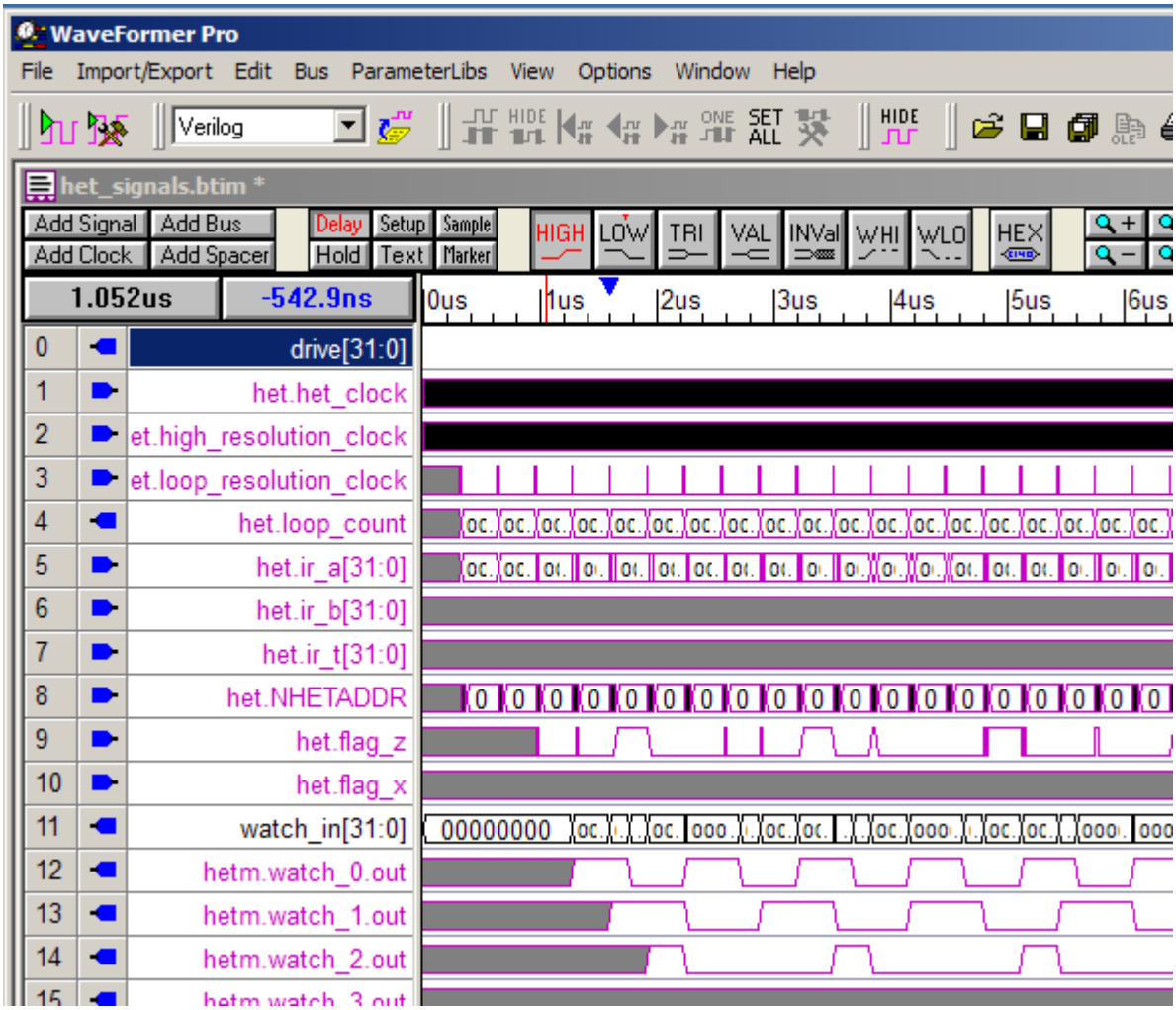


Figure 51. Expanded Waveform Shows Individual Signals for Pins 0,1,2

11. Restart the Simulator.

12. Right click on the project. Right click to open the project properties dialog. Select XOR from the left pane, and check 0_1 in Pins to XOR (see Figure 52). Click OK to apply the change and close the project properties dialog.

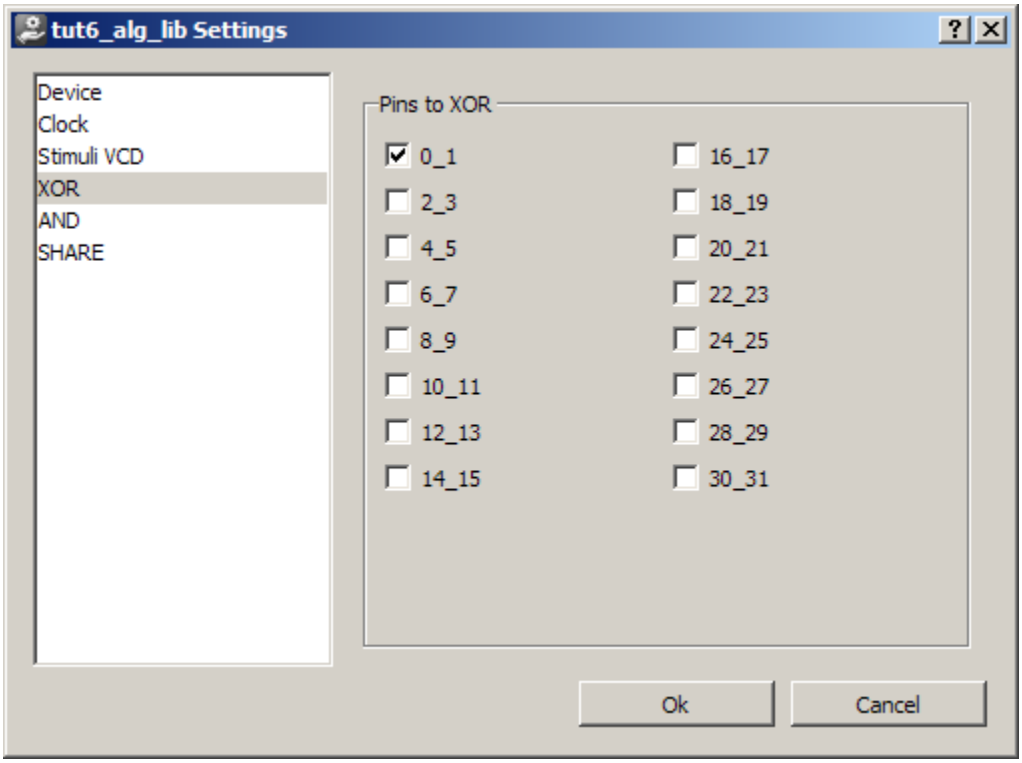


Figure 52. XOR Share Settings

13. Click OK.
14. Assemble the program.
15. Load the HET program.
16. Run the program for 50 loops.
17. Stop the program and switch to the Waveform Viewer to see the results. Your simulation results should match Figure 53. Compare this result to the result shown in Figure 52. Specifically, note how the individual signals from HET pins 0 and 1 from Figure 52 have been combined into a single output on HET pin 0 in Figure 53. This shows how the XOR share function operates. In addition to simply applying the 'XOR' logical function to two different pins, the XOR share feature of the HET allows the same physical pin to be updated twice during a single loop resolution period: once by operating on HET pin 0 and again by operating on HET pin 1. Without XOR (or AND) sharing, a program is restricted to one transition per pin during each loop resolution period.

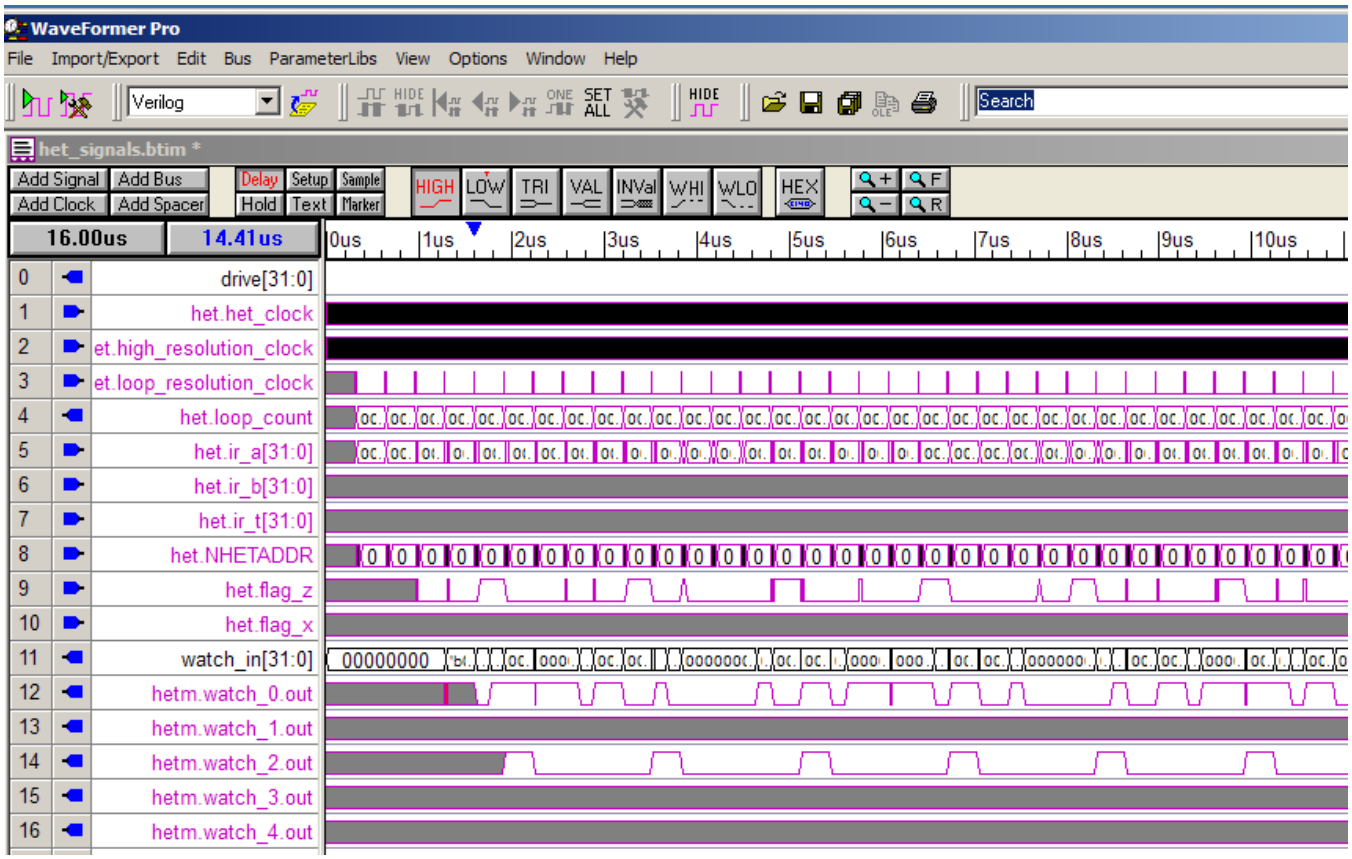


Figure 53. Simulation Results Showing XOR Sharing of Pins 0,1 Output onto Pin 0

2.7 Tutorial 7 — Input VCDs

This tutorial covers the following features:

- Input stimuli from a VCD file

NOTE: To use VCD files as stimulus, the HET IDE must launch the Waveviewer Free application, not Waveformer Pro or any other premium license. If you have acquired a Waveformer Pro (or other premium) license, then you can try Tutorial 8 instead. Alternatively, you can open an instance of Synapticad first from outside the HET IDE and open or create a new file so that you occupy the license file and force the HET IDE to launch the Waveviewer Free application instead. If you work in an environment where there are multiple floating licenses, you may want to change the license server path or license file that Synapticad uses temporarily so that the premium license is not found.

1. Open the HET GUI.
2. Open the Project:
   (a) Click Project → Open Project to open the project.
   (b) Open the *tut2_input_vcd.prj* file from <your_copy> /Tutorials/tut2_input_vcd. ***(Do not use the files in the tut7_input_vcd folder.)***
3. Selecting the VCD file:
   (a) Right click on the project.
   (b) Go to Project properties.
   (c) Select Stimuli VCD.
   (d) Open the folder to select the VCD file.
   (e) Select the file from <HET installation directory>/tutorials/tut7/inputVCD.vcd.

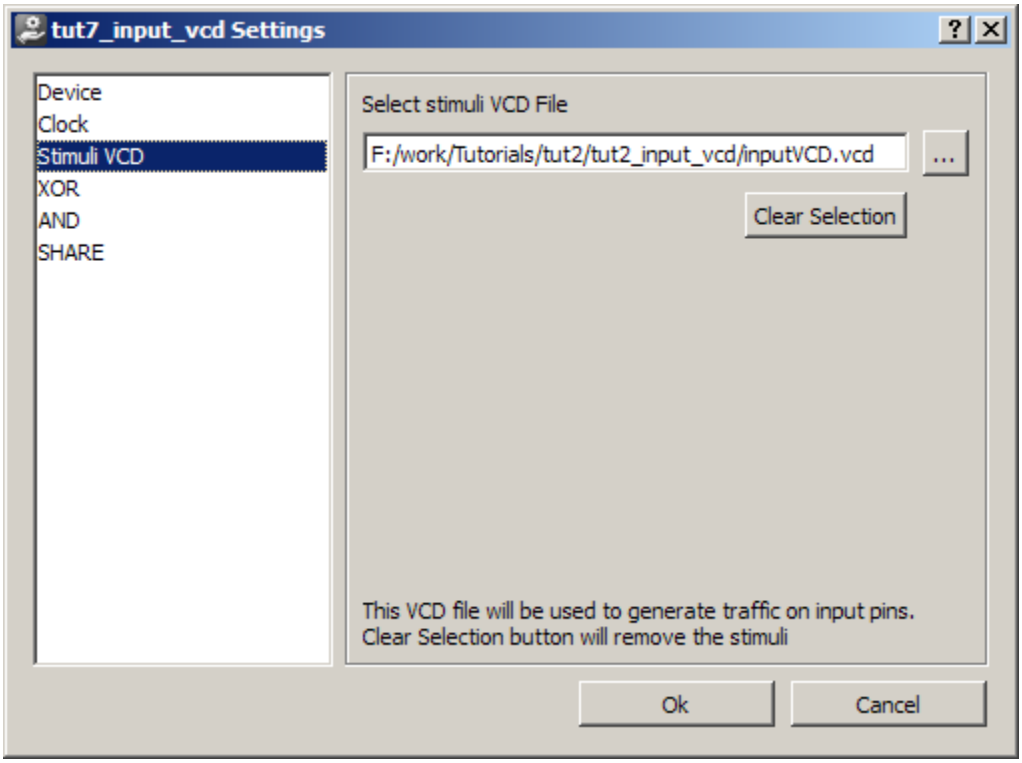


Figure 54. Selecting an Input VCD File From the Project Properties Dialog

   (f) Click on OK.
4. Assemble and Load the program.

5. Run the program for 100 loops.
6. Switch to the Waveform Viewer Free and inspect the results. Your screen (after zooming) should look like Figure 55.

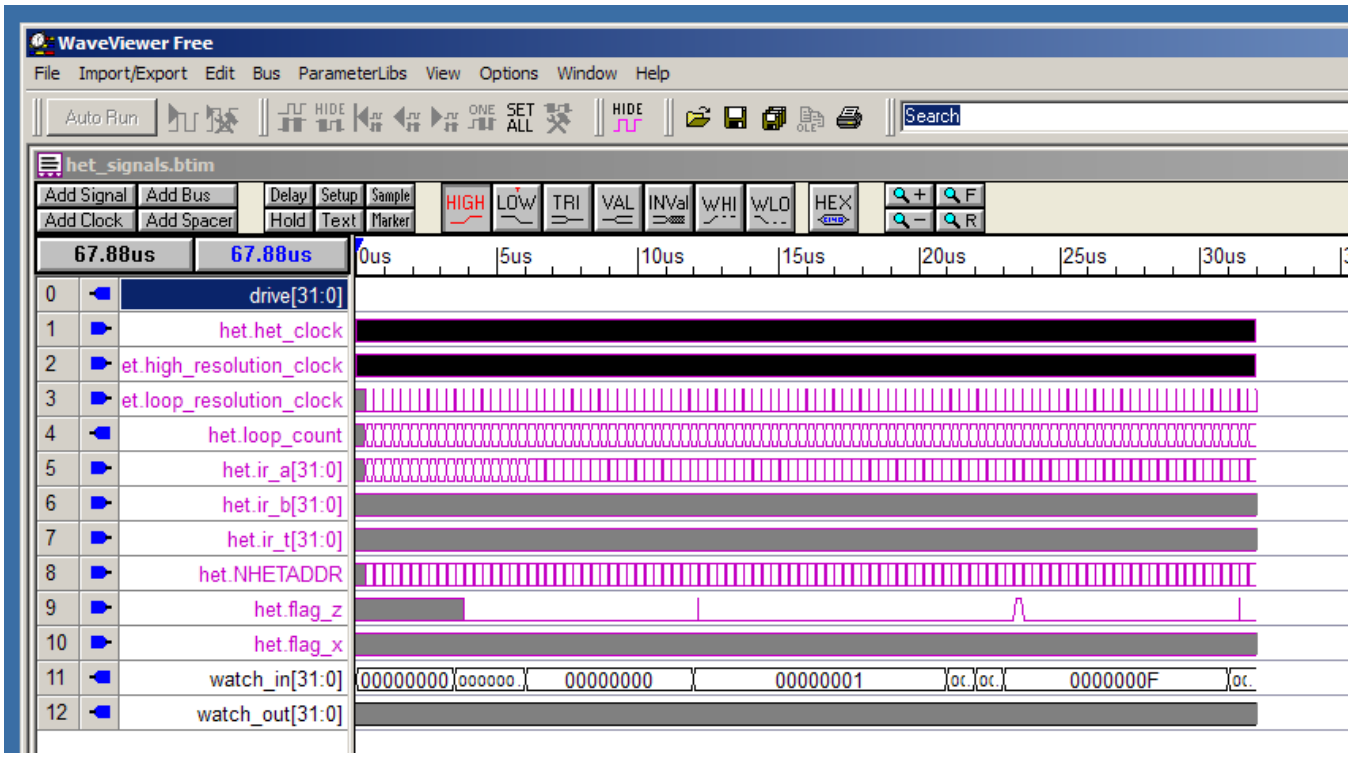


Figure 55. Simulation Results - Stimulus From VCD File Appears in 'watch_in[31:0]' Trace

7. Open the *inputVCD.vcd* file directly in Waveviewer Free and compare the stimulus with what you saw during your simulation. They match, however, note that in the VCD file, the signals that are being driven are shown individually rather than as a bus. Figure 56 shows the VCD file as viewed in Synapticad.

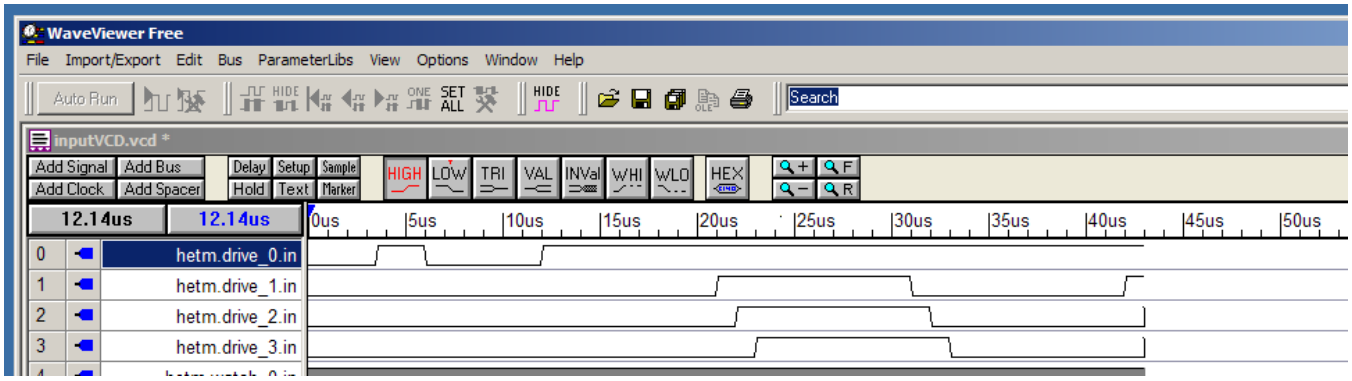


Figure 56. VCD File Used for Stimulus by Tutorial 7

2.8 Tutorial 8 — Input Stimuli From WaveFormer Pro

This tutorial covers the following features:

- Input stimuli VCD in synaptiCAD.with license
- Generating input in WaveFormer Pro

NOTE: For this tutorial, you must be able to access a Waveformer Pro license through the HET IDE. If you only have access to Waveviewer Free, complete Tutorial 7 instead.

1. Open the HET GUI.
2. Open the Project:
   (a) Click Project → Open Project to open the project.
   (b) Open the *tut2_waveform_editor.prj* file from <your_copy> /Tutorials/tut2_waveform_editor. ***(Do not use the files in the tut8_input_waveform_syncad_license folder.)***
3. Assemble and Load the project.
   (a) Open Waveformer Pro. *Make sure that Waveformer Pro is launched (see NOTE above.)*
4. Edit waveforms in the SynaptiCAD WaveFormer:
   (a) Zoom Out to see 100 µs.
   (b) Select the 'drive[31:0]' signal.
   (c) Create a pulses on all pins:
      (i) Click on ~20 µs inside the confines of the drive[31:0] row.
      (ii) Click again at 40-50 µs.
      (iii) Keep clicking at distance of 10 µs. Note that the Waveformer Pro tool automatically alternates the level between high and low as you draw. When complete, your stimulus should look like Figure 57.

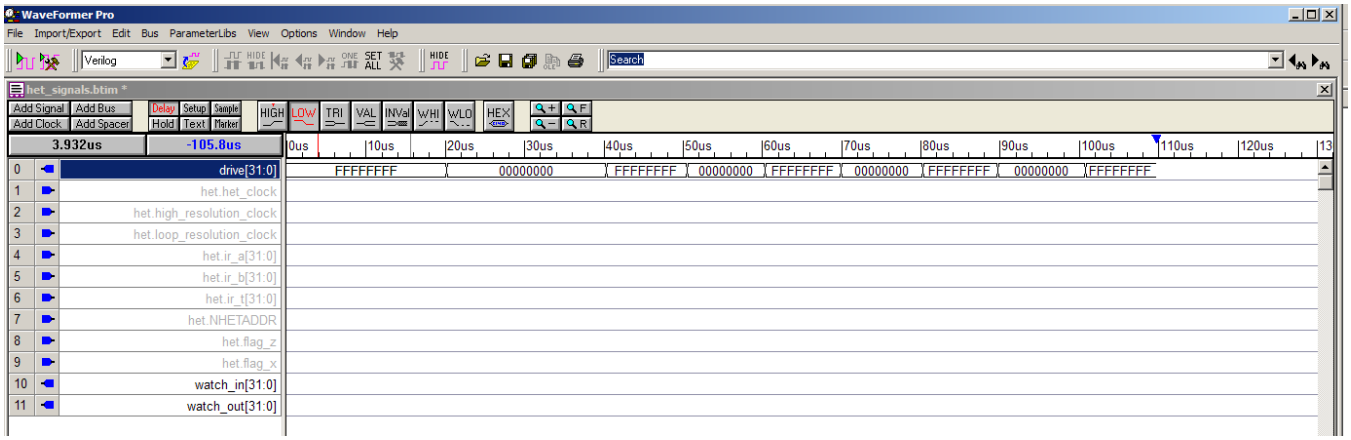


Figure 57. Input Stimulus Drawn Directly in Synapticad Waveformer Pro

   (iv) Save the file
5. Run the simulation for 100 loops.

6. The resulting waveform is shown in Figure 58. Note how the stimulus from the drive[31:0] bus appears on the watch_in[31:0] bus during the simulation run. Also note how the initial state of the drive[31:0] bus is dropped from the simulation. To work around this issue, you can create an additional dummy set of transitions in the drive waveform before the HET program begins executing, as shown in Figure 59. The extra transitions are drawn on a scale of 1 µs so that they are visible in the screen capture; in practice the dummy transitions can be drawn at the 1 ns scale. Figure 60 shows a simulation where the dummy transactions are within 2 ns of time zero so that they do not affect the simulation results because they are complete before the first HET clock.

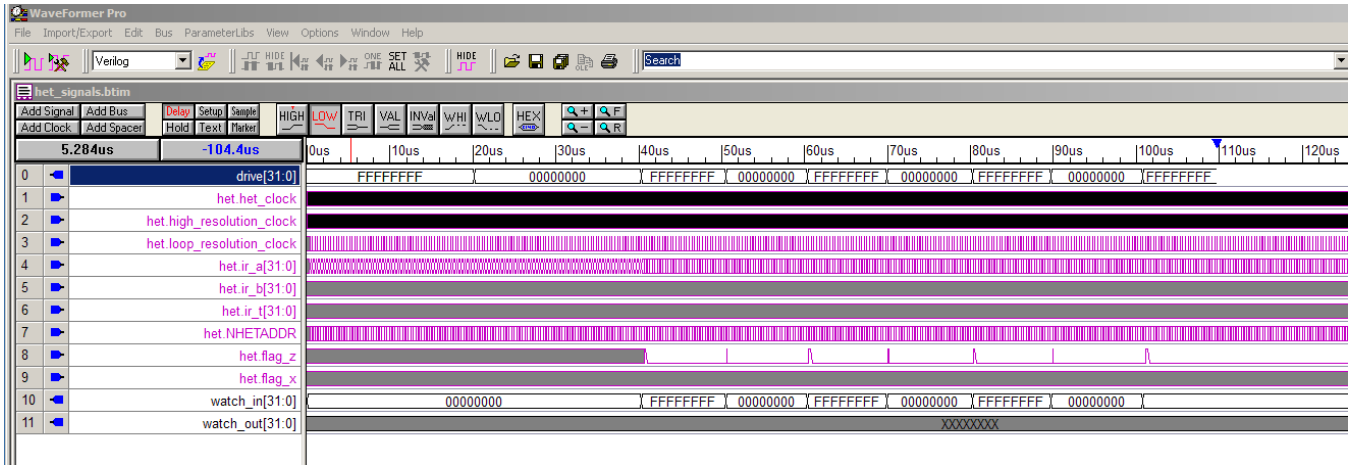


Figure 58. Simulation Waveform From Tutorial 8

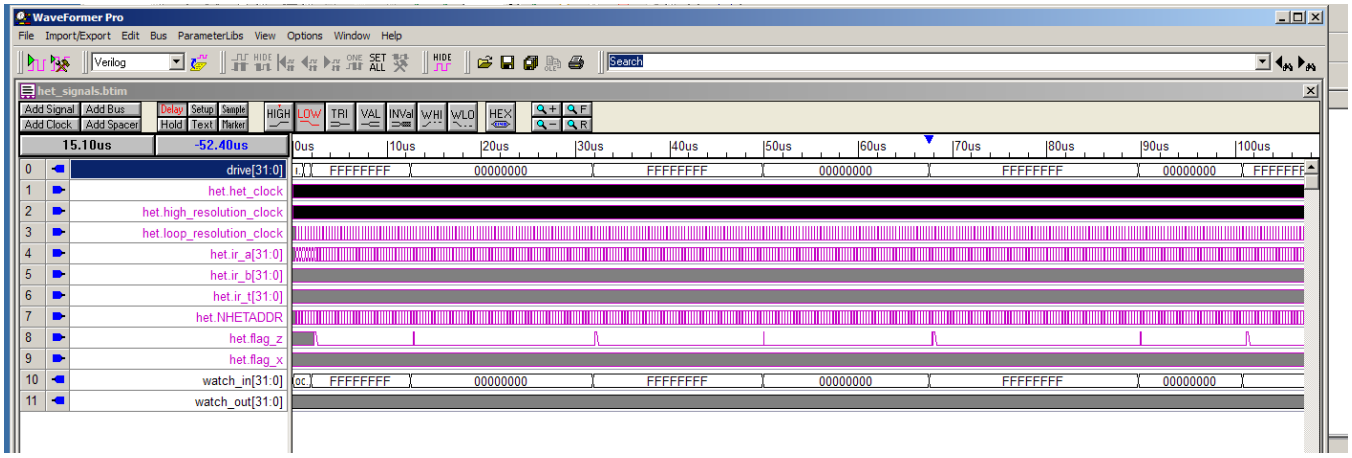


Figure 59. Additional Dummy Stimulus Added to Work Rround Time Zero Issue

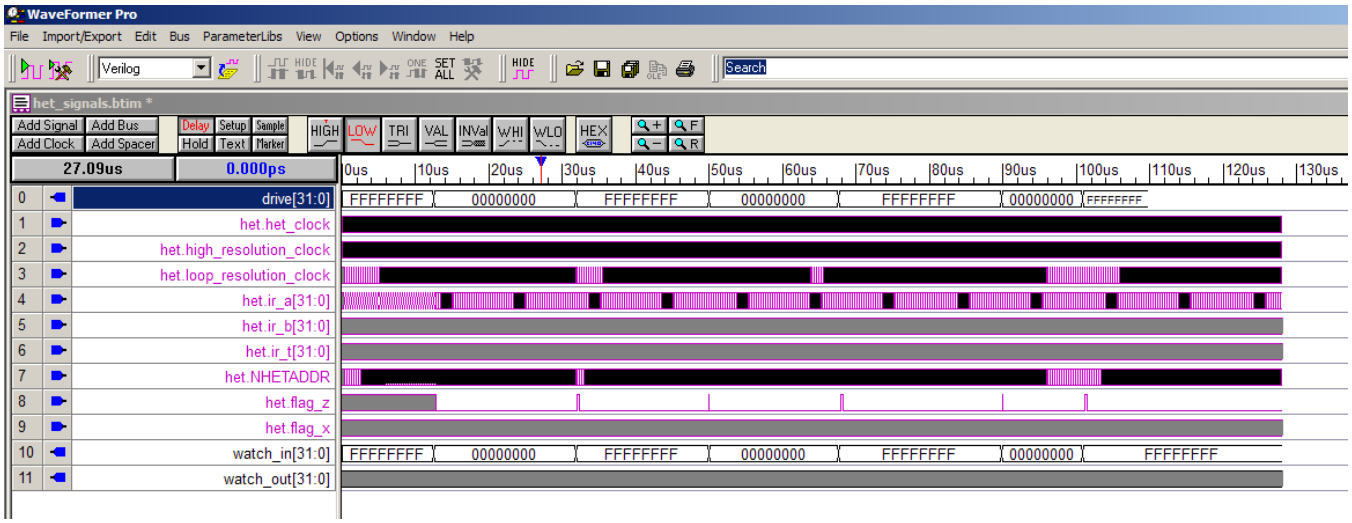


Figure 60. Moving the Dummy Stimulus to the 1 ns Scale Provides Perfect Tracking for Practical Purposes

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from B Revision (October 2015) to C Revision	Page
• Updates were made in Section 2.4.	29

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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