# HET Integrated Development Environment

# User's Guide

Literature Number: SPNU483A
November 2011

# Contents

# List of Figures



# List of Tables

# 1 Introduction

## 1.1 Purpose

This document describes the usage of HET IDE. The HET IDE is an Integrated Development Environment (IDE) intended to simplify the usage of the High End Timer simulator.

## 1.2 Scope

This document gives a detailed explanation of features of the GUI and different options available for the user.

## 1.3 SynaptiCAD's WaveFormer Pro and WaveViewer Software

The HET IDE launches either SynaptiCAD's WaveFormer Pro or WaveViewer, depending on your installed license, as the waveform viewer that captures simulation results. This document describes how to obtain a license for a full version of WaveFormer Pro and some of the features for working with the viewers.

## 1.4 Glossary

| | |
|---|---|
| **Algorithm** | A section of code stored in a database for easy insertion |
| **Algorithm Library** | Collection of Algorithms |
| **Complex Breakpoints** | Breaks/Halts the simulation when a specified action happens on a pin or when the memory contents matches with specified values |
| **GUI** | Graphical User Interface |
| **HET** | High End Timer |
| **NHET** | Next generation High End Timer |
| **N2HET** | Latest generation of HET |
| **IDE** | Integrated Development Environment. Tool where you can write program, compile, and see the output |
| **Memory Trigger** | Change a particular location in the memory when certain conditions (time, cycles, pin action, or interrupts) are met. |
| **SynaptiCAD** | The company that makes the waveform viewer and timing diagram editor used by HET IDE. |
| **VCD** | Value Change Dump File. Output file containing the waveform |
| **WaveFormer Pro** | SynaptiCAD's timing diagram editor that can generate stimulus vectors for the HET simulator and displays waveform results sent from the simulator. You must obtain a license from SynaptiCAD to use this feature. The first 90 days are free. |
| **WaveViewer** | SynaptiCAD's waveform viewer that can display streamed waveform results from the HET Simulator. If no WaveFormer Pro license is detected, then this is the default waveform viewer. |

# 2 Installation

Please refer to the Release Notes in the Installation folder for the installation and uninstallation procedures.

## 2.1 Installing the License for SynaptiCAD's WaveFormer Pro

The SynaptiCAD waveform viewers should be installed automatically during the installation of the HET_IDE software, but you will need to obtain a license directly from SynaptiCAD to enable the WaveFormer Pro features. To obtain the license:

1. Go to http://www.syncad.com/waveformer_het.htm
2. Fill out the form and SynaptiCAD will email you a license.

As a user of the HET IDE Software, you can request a free 90-day license of WaveFormer Pro with GigaWave feature. With this license, you will be able to draw stimulus vectors for HET simulations and view the streamed waveform results from the simulator. If you do not obtain the license, then your stimulus vectors will have to be created in separate tool, saved as a VCD file, and loaded into the HET Simulator separately. After the 90-day evaluation period or if no license is installed, the HET_Simulator will use SynaptiCAD's WaveViewer as a waveform viewer for displaying the results of the simulator.

During your 90-day license, you may use the full set of WaveFormer's features in the rest of your projects. WaveFormer Pro is a professional timing diagram editor that lets you draw and analyze timing of your circuit. WaveFormer Pro enables you to automatically determine critical paths, verify timing margins, adjust for reconvergent fanout effects, and perform "what if" analysis to determine optimum clock speed. WaveFormer Pro also lets you specify and analyze system timing and perform Boolean level simulation without the need for schematics or simulation models. The tool can also export your timing diagram as a Verilog, VHDL, SPICE or gate-level simulator simulator model. WaveFormer Pro also has the ability to import and annotate simulation and logic analyzer data, for publication-quality design documentation.

If you would like a permanent copy of WaveFormer Pro, please contact SynaptiCAD at sales@syncad.com, www.syncad.com, or call by phone 540-953-3390.

# 3 Overview of HET IDE

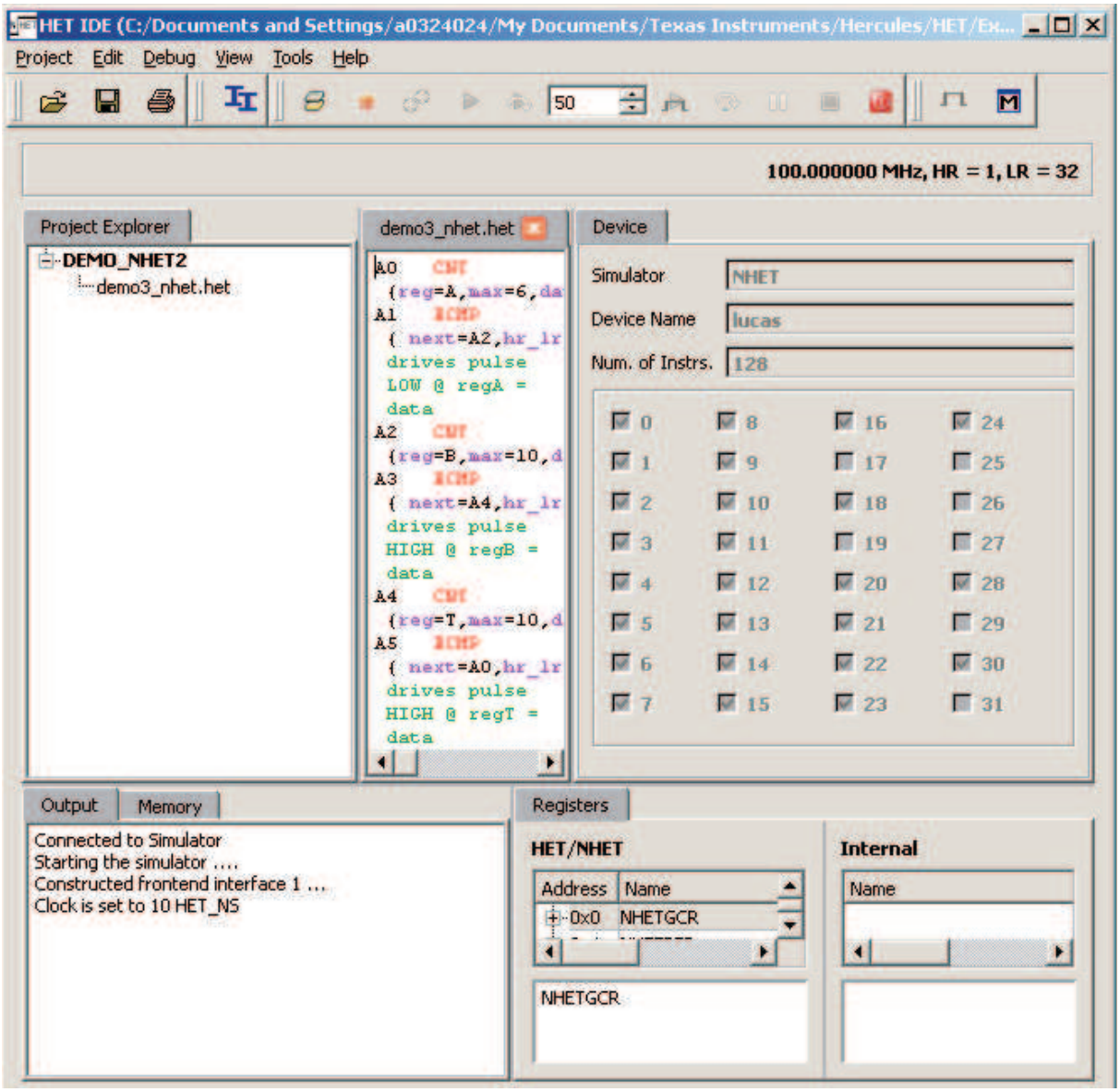
This chapter gives a brief introduction of HET IDE, its modes of operation and lists the features available in the HET IDE.

The latest information on HET IDE can be found on the official HET IDE wiki page.

## 3.1 Introduction

The HET IDE is an Integrated Development Environment (IDE) intended to simplify the usage of the High End Timer Simulator. It provides advanced debug features such as Complex Breakpoints or Memory Triggers which are helpful to debug large applications.

**Figure 1. View of HET IDE in Edit Mode**



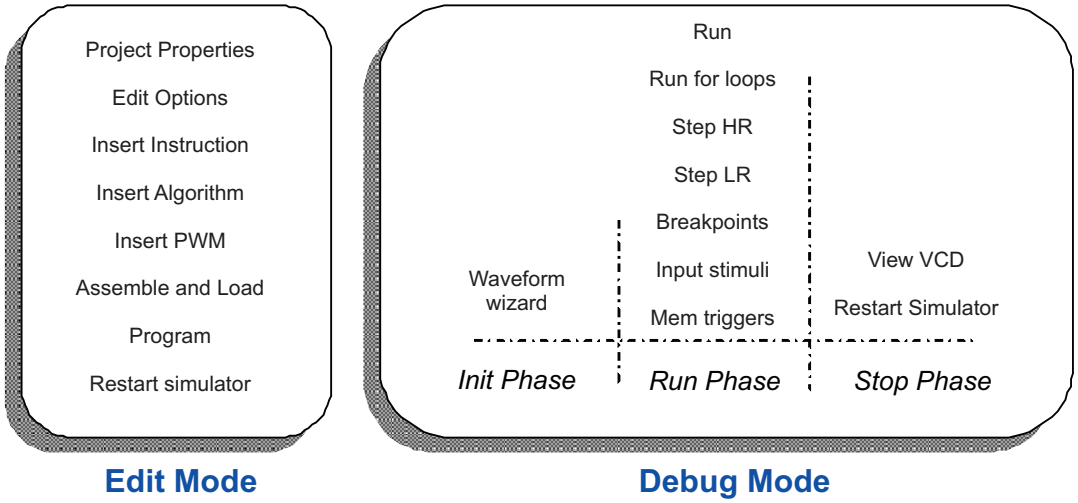

## 3.2 Modes of Operation

The HET IDE simulator has two modes:

1. Edit mode
2. Debug mode - which is further divided into three phases:
   (a) Initialization phase
   (b) Run phase
   (c) Stop phase

Depending on the mode, certain options are enabled, and some are disabled. Figure 2 below lists the options enabled to a user in edit mode and in debug mode.

**Figure 2. Modes of Simulator and Commands Available**



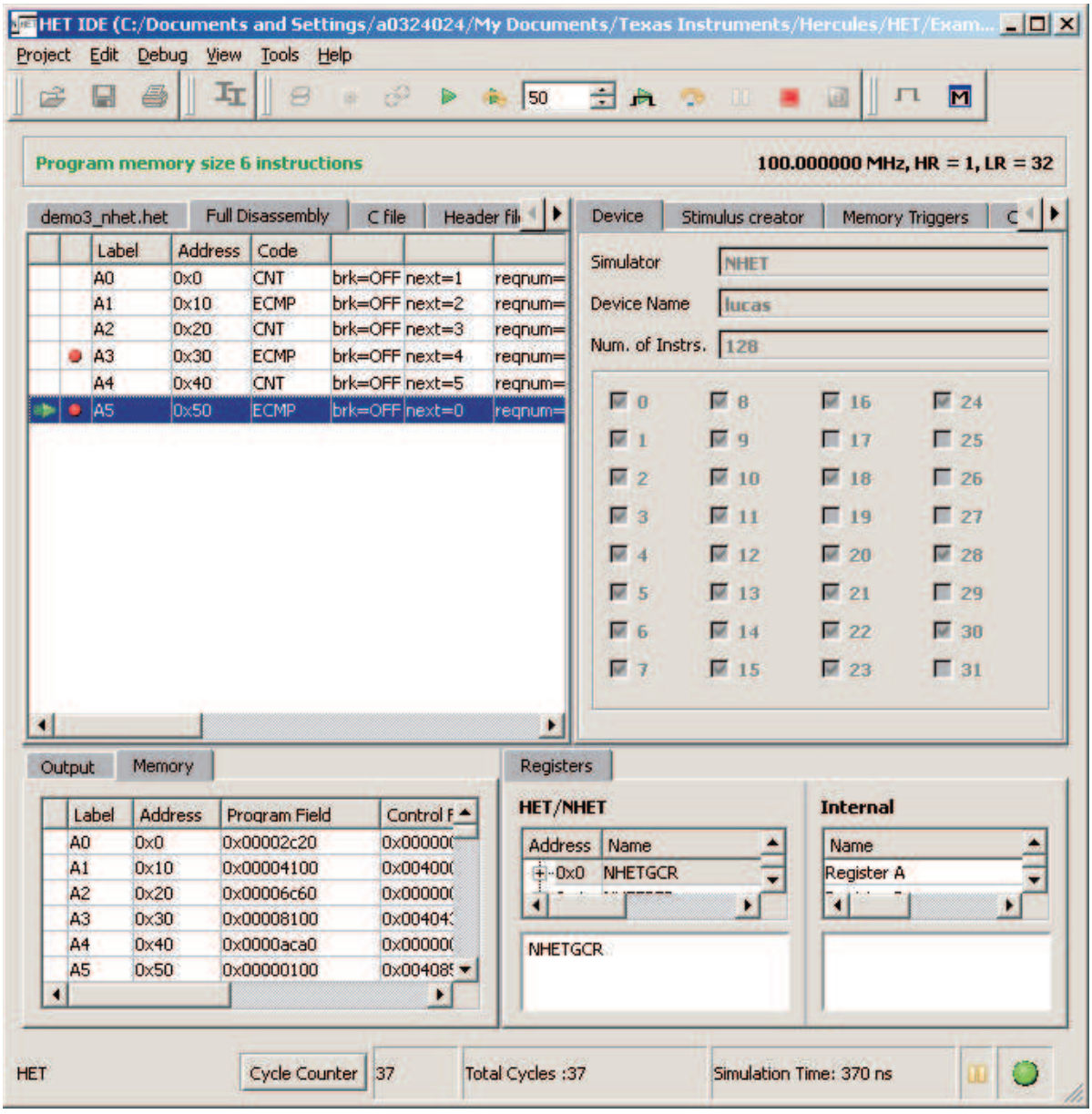

## 3.3 Features

Following are the important features available in the HET IDE:

1. Configure the HET device either by selecting from the existing devices or create a new custom device
2. Create and manage projects
3. Add and modify instructions easily using Drag and Drop and double-click features
4. Save frequently used code in Algorithm Library
5. Generate PWM by specifying Frequency and Duty Cycle
6. Drive Input Stimuli on HET pins
7. Set Complex Break Points on pin actions, data field
8. Modify memory at certain point / on pin action using Memory Triggers
9. Select the signals to be traced in VCD using WaveForm Wizard/SynaptiCad wizard
10. See the changes of pins, registers, data, etc simultaneously during simulation in SynaptiCAD (Dynamic Streaming)
11. Edit an instruction by double-click
12. Compile-on-fly (compile an instruction immediately after adding)
13. Dump memory into files

Figure 3 gives the look of HET IDE in debug mode. Memory and register windows are displayed in the bottom. Input stimuli, memory triggers, and complex breakpoints are placed on the right. Disassembly is shown in the main area.

**Figure 3. Debug View of HET IDE**



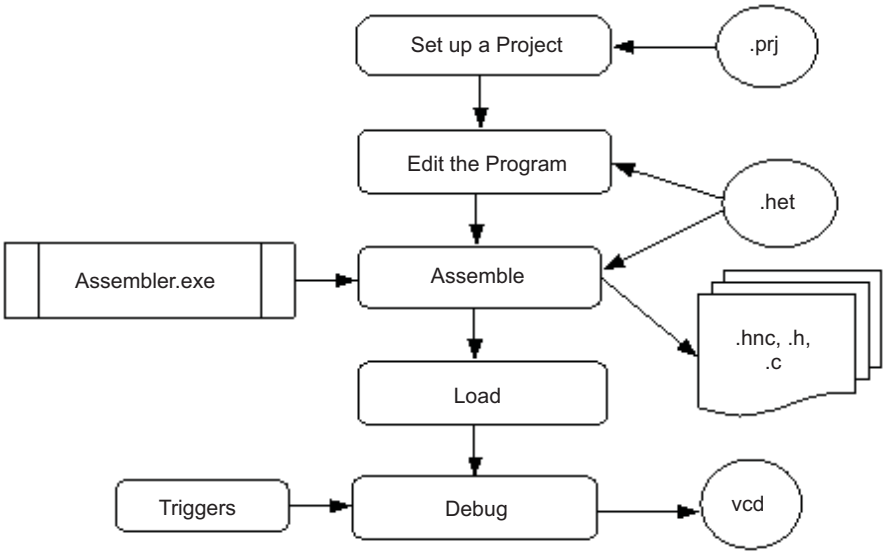

# 4 Programmer's View

This section describes the usage of the HET IDE.

## *4.1 Flowchart*

This flowchart shows all the steps involved from the creation of a project to the generation of an output VCD file by using HET IDE. Subsequent sections describe each of these steps.

**Figure 4. Flowchart Showing the Flow of HET IDE**




The HET IDE maintains the files in forms of projects. Each project will save the HET file, its settings (clock, device, pin configuration, etc), debug features, and output selection.

A new project can be created using the Project Wizard, or an existing project can be opened. In this project, the HET file can be edited using the Drag and Drop feature for inserting instructions. The user can then save, assemble, load and view the results in the VCD file.

# 5 Setup

## *5.1 Device Configuration & Selection*

The HET IDE provides options such as:

- Creating a new device configuration
- Modifying the existing device and changing the device for a project

The HET IDE supports two types of configurations – HET and NHET (22 instructions). The default devices and user devices are listed in the left side of the Device Configuration Window. **User cannot edit the "Default" devices but can edit 'User' devices.**

**Refer to Section 6.1 and Section 6.2 for more details on creating a new device configuration and modifying an existing device configuration.**

### 5.1.1 Project Creation

A project can be created by using the Project Wizard (**Project → New Project**). Please refer to Section 6.3 for how to create a new project from scratch using Project Wizard. The project will be created with the default register settings.

Refer to Appendix B for more details.

**NOTE:**

- Default HET and NHET register files are provided with the installation for user.
- Two files, het_signal.btim and het_watch_only_signals.btim, will be created in the project.

### 5.1.2 Project Properties

Project settings such as device type, clock frequency, stimulus VCD file and pin configurations can be modified using **Project → Project Properties**. A different .het file can also be set using **Project → Set HET file** option. Refer to Section 6.6 for more details.

**NOTE:** When closing a project, all properties will be saved along with the project.

## *5.2 Edit*

A program can be added in the editor window using any of the three options:

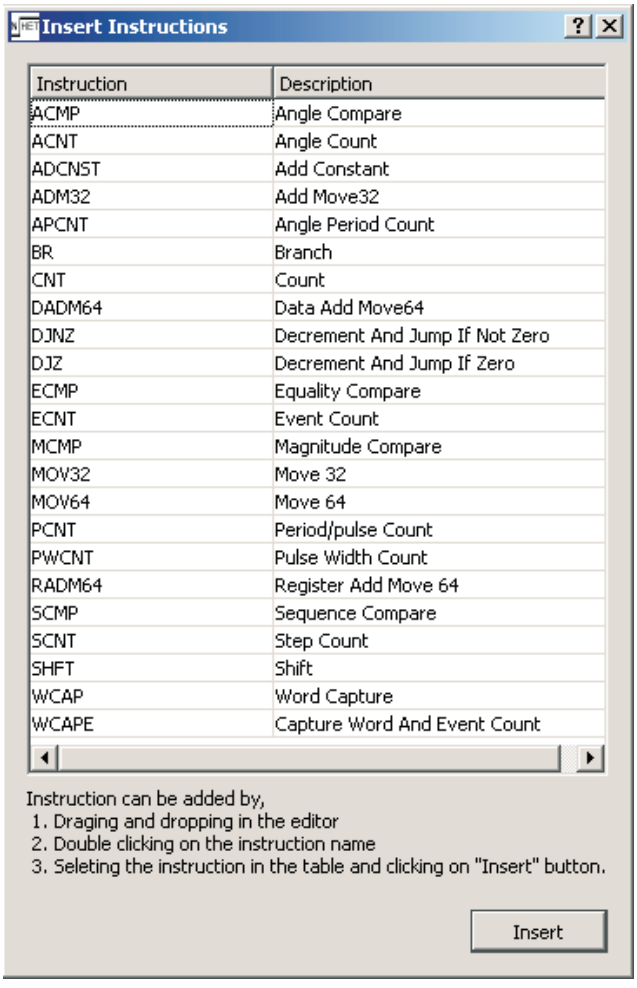
1. Insert Instruction
2. Insert Algorithm
3. Insert PWM

### 5.2.1 Insert Instruction

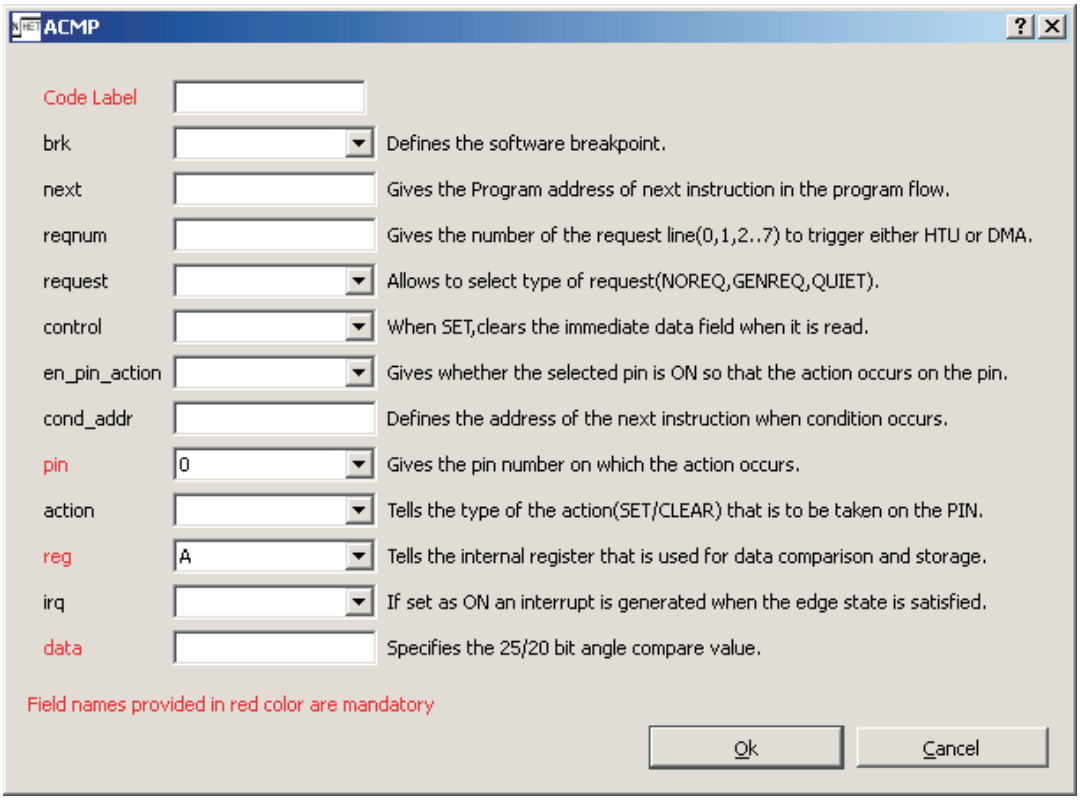
Instructions can be added to the HET file using the drag and drop feature. This can be done easily by clicking the "**Insert Instruction**" icon on the tool bar.

Instructions can be inserted in three ways. First, select the instruction to be inserted, then

1. Drag and drop onto the editor window and provide the required parameters.
2. Double-click on the instruction to be inserted and provide the required parameters.
3. Select the instruction, then click the "INSERT" button and provide the required parameters.

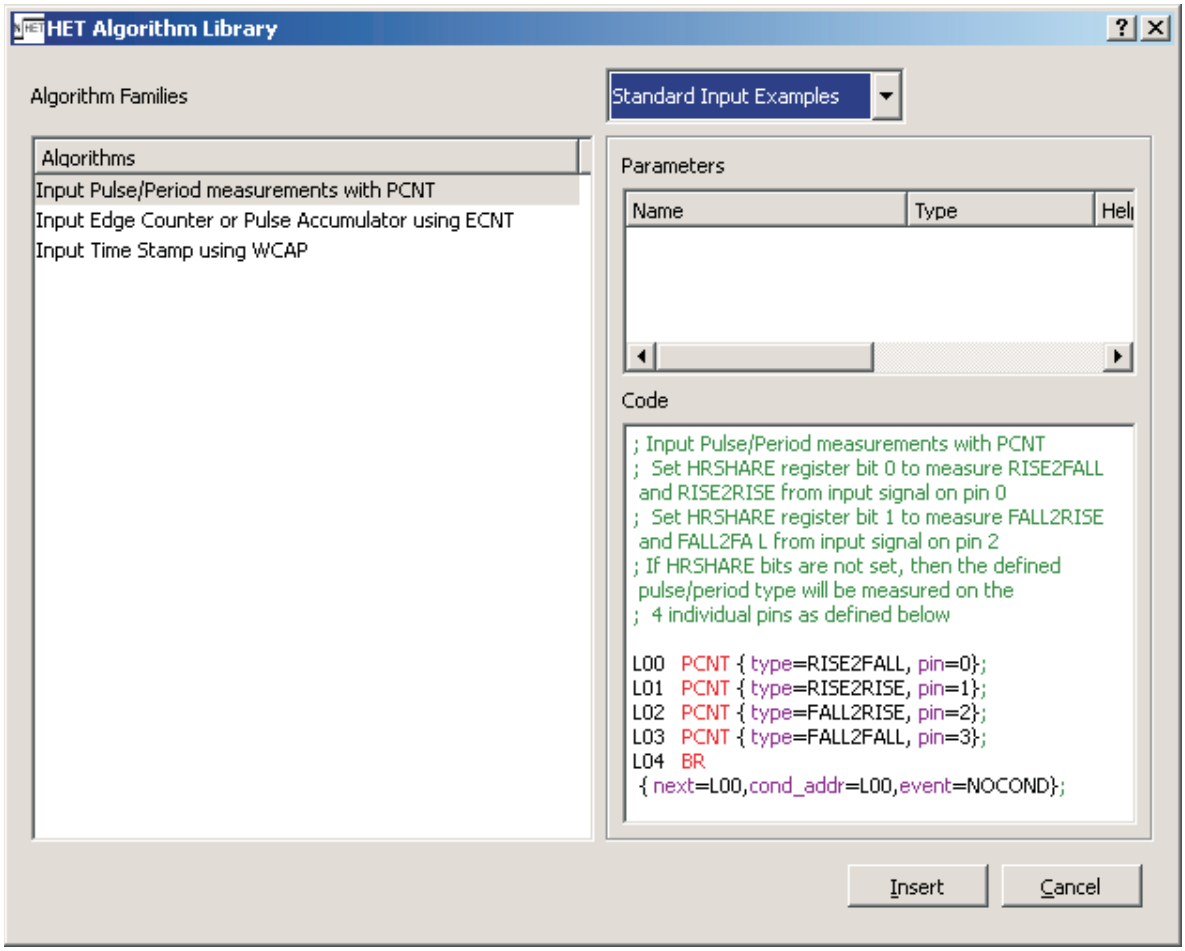


Instructions can be modified by using the "**double-click**" option. Double-click on the instruction to be modified and modify the parameter values.



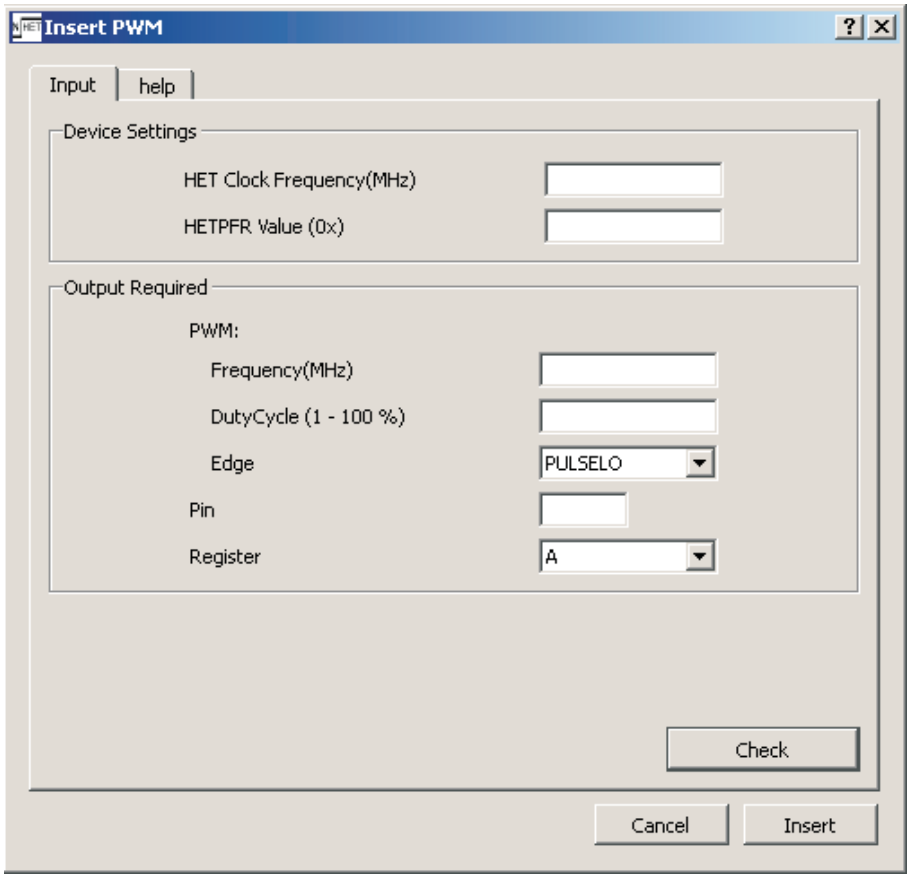
### 5.2.2 Algorithm Library

Frequently used code can be saved as an algorithm, and by using simple clicks, it can be added to the editing file. The blocks are parameterized so that the user can pass any value to those parameters. Refer to the *HET IDE Tutorial Guide* (SPNU485) for more details on how to create and insert an algorithm.



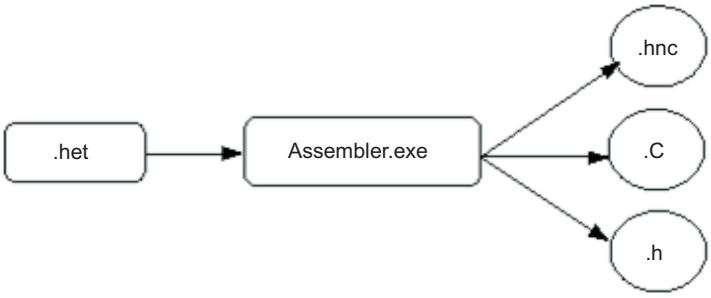
### 5.2.3 Insert PWM

HET IDE provides an option to generate a simple PWM wave on a pin by giving the required values. Use the option **Edit → Insert PWM** and provide the required values to insert the code with specified values.




## 5.3 Assemble

**Figure 5. Assembler Flow**




The Assembler will take the **.het** file as input, compile it, and generate **.hnc file, .c,** and **.h** files as output. Any errors while compiling will be displayed in the output window. The first error in the output window will be highlighted in the editor window.

## 5.4 Load

This step will load the instructions into the HET memory and enter into Debug mode by opening a disassembly window showing the Label, Address, and Code of each instruction. The user can either insert or remove breakpoints. This step enables the Run, Run for Loops, Step Instruction, and Run to End of Loop options.

## 5.5 Debug

This allows the user to generate stimulus on the input pins using the Stimulus creator option. A VCD file is generated to view all the changes on signals. Refer to Section 8 for more about how to create Stimulus as well as different advanced debug features available.

## 5.6 File Formats

The NHET/HET Assembler accepts both ***.het** and ***.C** file types as inputs and generates a ***.hnc** file as output. These files (*.het or *.C) can be set as HET files to a project using **Project** → **Set HET file**.

### 5.6.1 Format of C Files

The C file must follow the following syntax:

1. The array of hex code should start with HET_MEMORY.
2. Each 32-bit value should start with '0x' and should be the first word in the line.
3. Each 32-bit should start in a new line.

```
HET_MEMORY const HET_INIT0_PST[1] =
{
    /*P00*/
    {
        0x00002E00,
        0x00000000,
        0x00000000,
        0x00000000
    }
};
```

**NOTE:** If there is no code available, the **Label** and **Code** columns will be empty in the disassembly window.

## 5.7 Run a Program

Refer to "**Tutorial 1** – **Getting Started**" in the *HET IDE Tutorial Guide* (SPNU485) to get an idea on how to run a program.

# 6 HET IDE Environment Setup

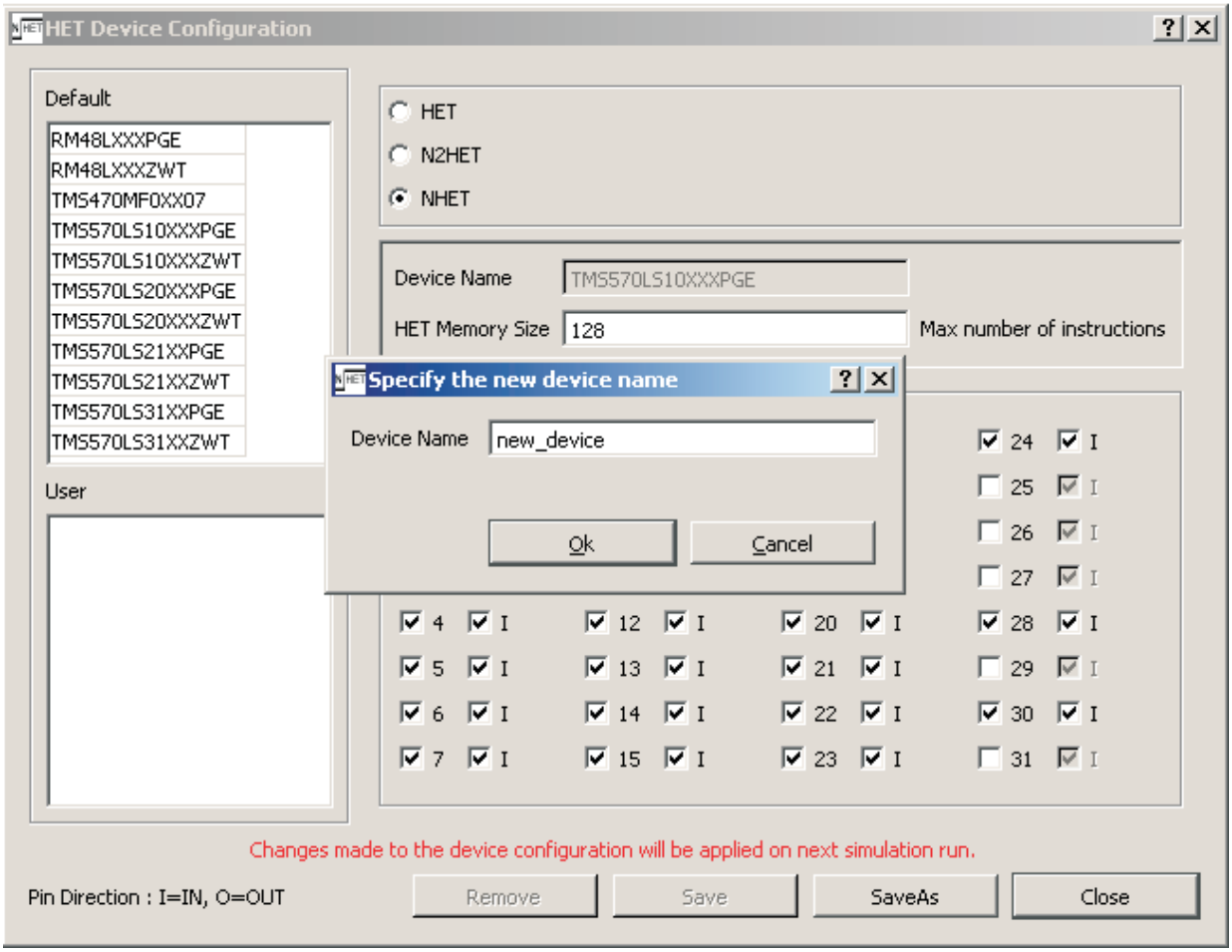
This section gives a detailed description of the features available in the HET IDE.

## 6.1 Create a New Device Configuration

If the required configuration is not present, create a new configuration by following the steps below:

1. Go to Tools → Device Configuration
2. Select an existing device
3. Select the device type (HET/NHET)
4. Click "Save As"
   (a) Provide a new 'Device Name' and click "OK"

**Figure 6. Create a New Device**




5. Select the new device
6. Set the ‘HET Memory Size’
7. Select the required pins and their default direction (I – In, O – Out)
8. Click "Save"

## 6.2 Modify an Existing Device Configuration

HET IDE provides an option to modify an existing device configuration. The user can only modify configurations under “**USER**”. The configurations under “**DEFAULT**” cannot be modified.

Follow the steps below to modify an existing device:

1. Go to Tools → Device Configuration
2. Select the device to modify from the list under “USER”
3. Make the changes
4. Click "Save"

Note that the changes made will either be visible in the next simulation run or be visible immediately upon restarting the simulator.

### 6.3 Create a New Project from Scratch

1. Go to Project → New Project to create a new project
2. Specify the name in the Project Name field and the location of where to store the project in the Project Path field and click "Next"
3. Select the type of the device from the list and click "Next"
4. Specify the clock frequency and click "Next"
5. Specify the HET source file name (new file or existing file) and click "Next"
6. Check whether all the project properties are correct and click "Finish"

### 6.4 Assigning a Different Device to an Existing Project

The user can assign a different device to a project using the following steps:

1. Go to Project → Project Properties
2. Click on “Change Device” and select a new Device
3. Click “OK”
4. Click “OK”

This option is available only if the project is active. For more about how to make a project active, refer to Section 6.5.

### 6.5 Setting a Project as an Active Project

When multiple projects are open, the user can easily switch to another project by using the “**Set As Active Project**” option. Right-click on the project and select the “Set as Active Project” option, or go to **Project → Set As Active Project** to make that project active.

### 6.6 Setting a Different *.het or *.C File to a Project

Every project will have its own HET file. HET IDE provides an option to set different *.het or *.C files to a project, i.e., the user can set other HET or C files to a project according to the requirements.

1. Go to Project → Set HET file
2. Specify the path of the HET or C file
3. Click "OK"

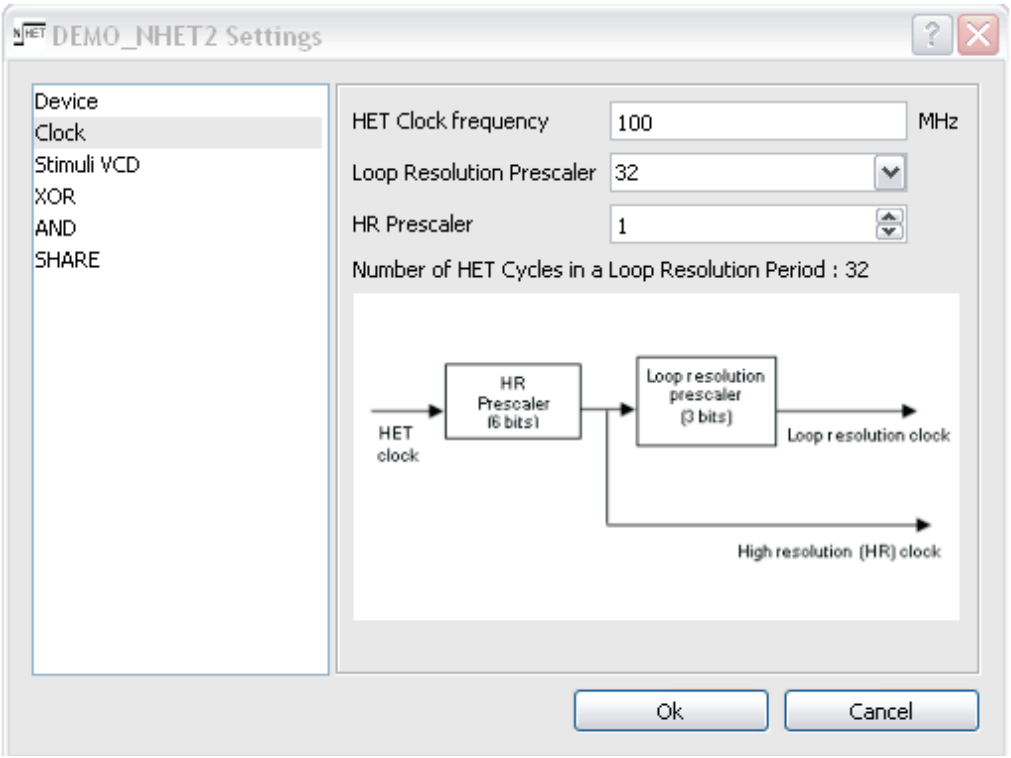
### 6.7 Changing Clock Frequency and PFR Register Value

HET IDE provides an option to modify Clock Frequency and Prescale Factor Register values in a single step.

1. Go to Project → Project Properties
2. Specify the path of the HET or C file
3. Click "OK"
   The Prescale Factor can be changed by modifying the Loop Resolution value and HR value. It automatically updates the NHETPFR/HETPFR register with the new value.
4. Specify the Loop Resolution Prescaler
5. Specify the HR Prescaler
6. Click "OK"

The number of HET cycles present in an LRP is also displayed.

**Figure 7. Clock Configuration Window**




## 6.8 Restart the Simulation

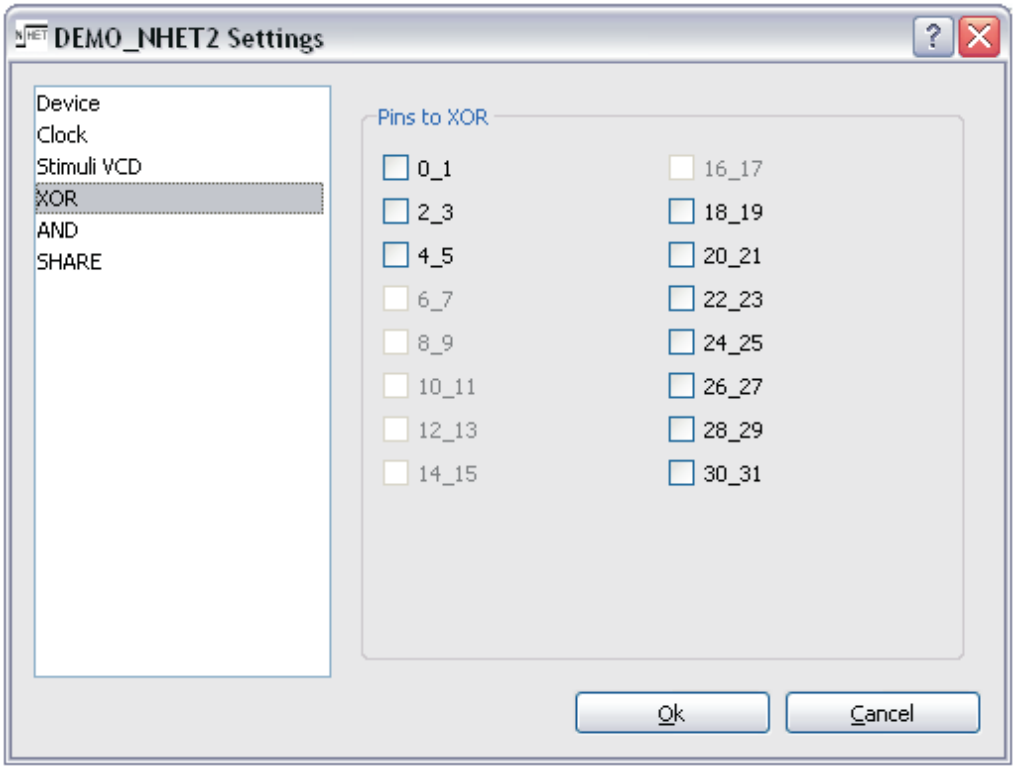
The Simulator can be restarted without closing the application. In Debug mode, stop the simulation and then restart by using **Debug → Restart Simulator**, or use the Restart Simulator button on the toolbar.

## 6.9 Using the XOR , AND, SHARE Features

To use the HR Share, XOR-share, and AND-share features, follow the steps below:

1. Go to Project → Project Properties
2. For XOR, select XOR
   For AND, select AND
   For SHARE, select SHARE
3. Click the checkboxes to select the Pins

**Figure 8. XOR/AND/SHARE Configuration Window**




### 6.10 Save and Load Registers

In order to run a program, HET/NHET should be configured properly. The clock ratios, pin directions, and enable bits must be updated.

This is done by using the Save and Load Registers feature. For the first time, the registers can be edited manually in the Register window and saved as a register configuration (*.cfg) using **Tools → Save Registers**. For further runs, this file can be loaded using **Tools → Load Registers**.

Registers can also be saved with a project. When a project is open, the registers that were previously saved will be loaded automatically. The option **Project → Save Registers** is used for this.

### 6.11 Save Memory Dump

The HET IDE provides an option to save the memory into files. This option is available under **Project → Save Memory Dump**.

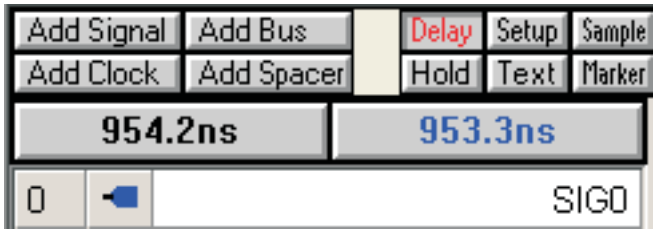
## 7 Synapticad's WaveFormer Pro and WaveViewer

Depending on your installed license, the HET IDE launches either SynaptiCAD's WaveFormer Pro or WaveViewer as the waveform viewer for the simulation results. The instructions for obtaining a WaveFormer Pro license are covered in Section 2.1. This section covers a few of the WaveFormer Pro and WaveViewer features that are most interesting when working with the HET simulator. WaveFormer Pro is a full-timing diagram editor; the complete manual can be found under the **Help > Timing Diagram Editor Help** menu in the WaveFormer window.

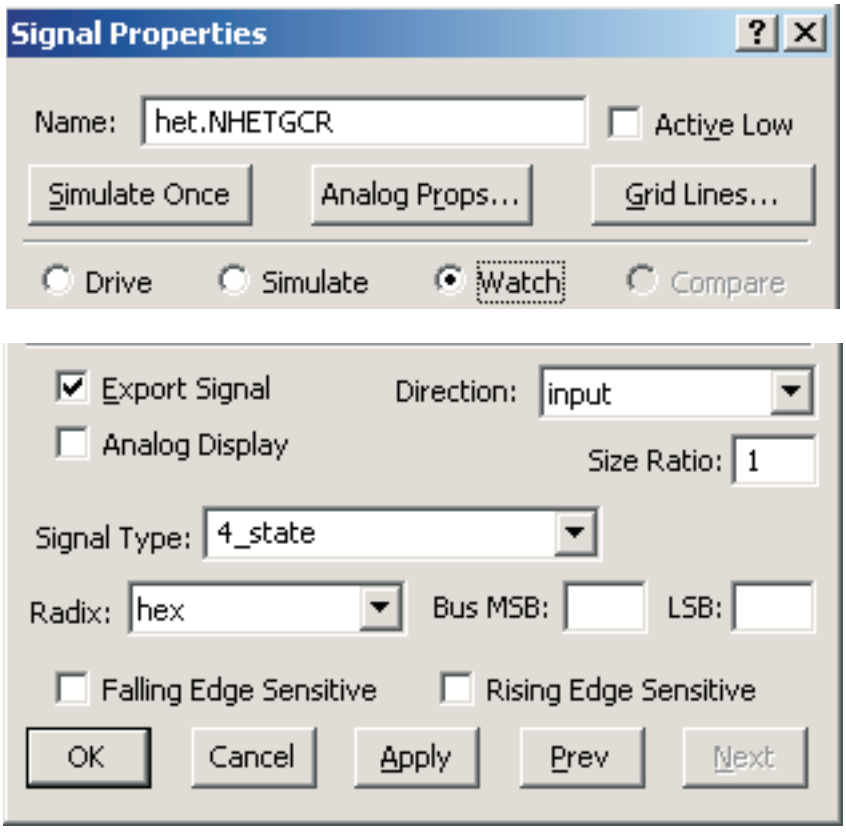
### 7.1 Launching WaveFormer Pro and HET_signals.btim File

The HET simulator will launch WaveFormer Pro loaded with a default waveform file **het_signals.btim**. This file contains the main set of signals and registers that you may want to watch, but more signals and registers can also be watched.

- Inside WaveFormer Pro, click the "Add Signal" button to add a new signal.




- Double-click on the new signal's name to open the *Signal Properties* dialog.
- In the **Name** field, paste the name of the signal to watch and add "***het.***" to the front of the name.
- Set the signal type to ***Watch*** so that the waveform viewer knows to try and hook it up to the simulator.
- Set the **Direction** to ***Input***, because this will be a signal coming from the HET Simulator.
- Set the **MSB** and **Radix** as needed.
- Click "OK" to apply the changes and close the dialog.



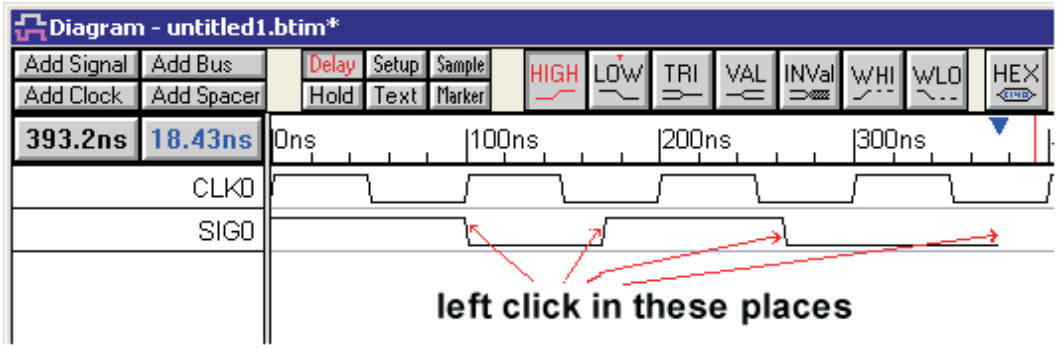

- The next time the simulator is ran, the new signal will display the values generated during simulation.
- To save the signal, choose **File** > **Save Timing Diagram** from the menu, and save the ***het_signals.btim*** file.

## *7.2 Drawing Waveforms*

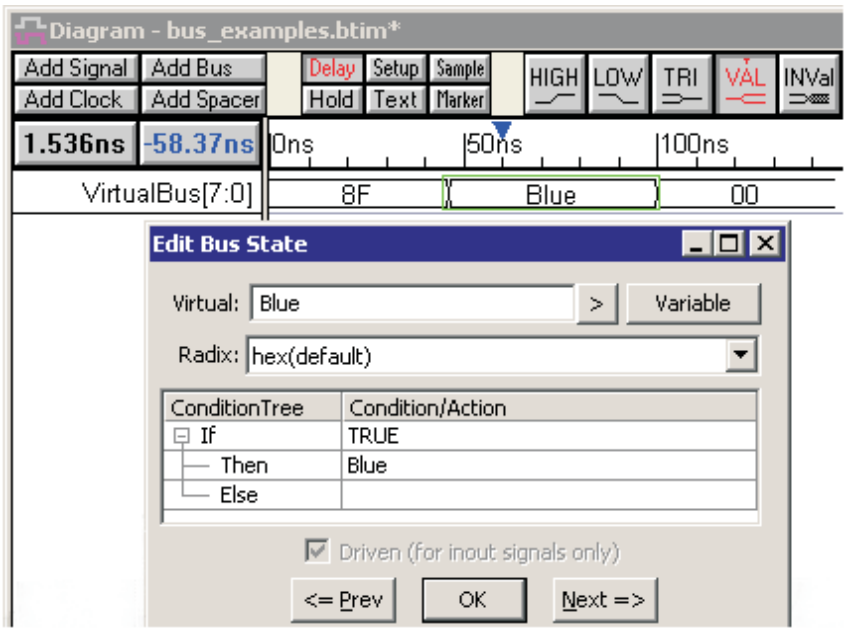
If you have a WaveFormer Pro License, you will be able to draw stimulus on the **output** signals (black signals).

- Look at the waveform buttons. The red button controls the type of waveform that is drawn next. After that, the buttons will toggle to the state with the red T on top. Clicking the buttons will set a different state to be drawn next. This shows a toggling between high and low states.


- In the example below, the mouse is placed on the same row as signal **SIG0** at about **100ns** and then **left-clicked** to draw a high waveform from 0ns to the mouse cursor position. Then, the mouse was moved to **150ns** and **left-clicked** to draw a low segment from the end of the signal to the cursor point. **Chapter 1: Signals and Waveforms** in the Timing Diagram Editor manual covers all of the signal drawing and display features.




- If you draw with the valid bus states, then you can edit the bus segment value by opening the *Edit Bus State* dialog. Either double-click on the segment, or select a segment first, then click the **HEX** button on the button bar.

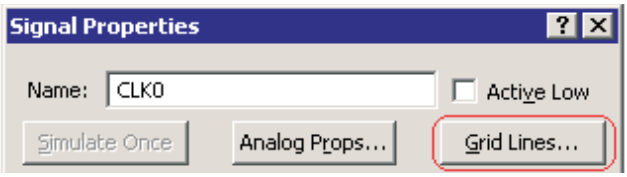



## 7.3 Adding Grid Lines

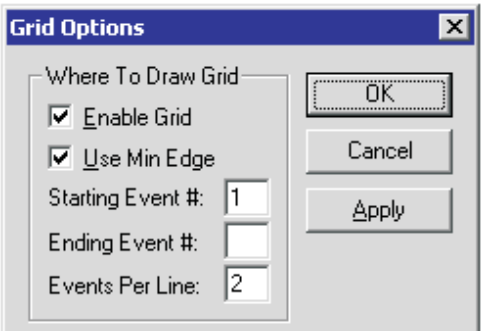
Grid lines are vertical lines drawn from the edges of a clock or on any signal.

### 7.3.1 To Draw Grid Lines

- Double-click on the clock or signal name to open the *Signal Properties* dialog, and click the **Grid Lines** button in the top right of the dialog. This opens the *Grid Options* dialog.



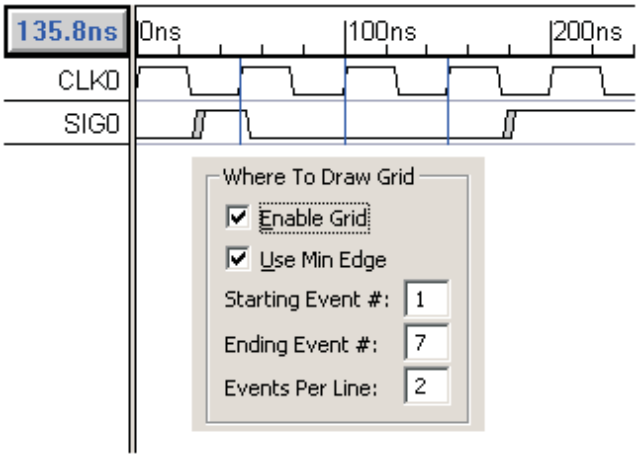

- Check the **Enable Grid** checkbox. This enables the rest of the grid options. If you later decide to turn off the grid, then uncheck this box.




- Fill in the rest of the controls to design a grid pattern. Use the **Apply** button to test your grid settings.

### 7.3.2 Where to Draw Grid Section

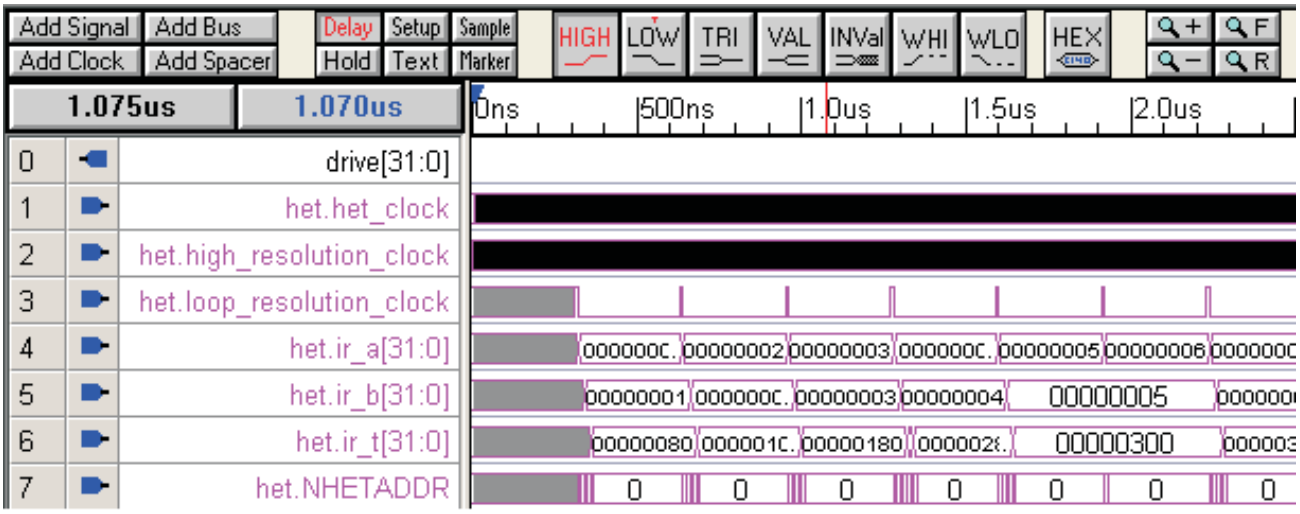
- The **Use min Edge** checkbox determines if grid lines are drawn from the minimum or maximum edge of a clock transition.
- The **Starting Event #** is the event number where the first grid line is to be drawn. The first event in the clock is the 0 event.
- The **Ending Event #** is the event number where the last grid line is to be drawn. If blank, it will draw on all the clock edges.
- The **Events per Line** edit field determines how many clock transitions occur before another grid line is drawn. "1" draws a grid line on every transition, and "2" skips one transition between grid lines, etc.



## 7.4 Bit-slicing a Simulated Signal (e.g. Signal Type Watch)

A simulated signal can display a bit-sliced portion of the signal by changing the MSB and LSB settings. You can also reverse the bits of the slice by reversing the MSB and LSB. Bit-slicing can be done on the original simulated signal without losing data or on a copy of the original signal. The figure below shows two copies of the original signal so that it is easier to compare the bit-slices with the original signal.

- Either run a simulation, or load any previous *.btim file. For this technique to work, the signal must be of type **watch** or **simulate**.
- To see the bit-slice on a different signal than the original signal, copy and paste the signal by selecting the signal name and pressing the **CTRL-C** then the **CTRL-V** keys.
- Double-click on the signal to be sliced and set the MSB and LSB to set the slice range.
- The bits can be reversed by making the MSB smaller than the LSB.



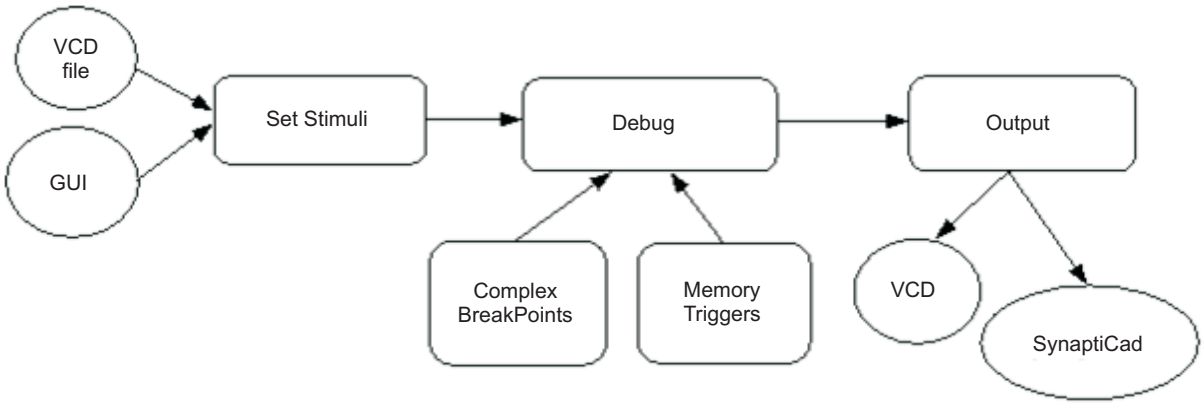

## 7.5 Bit-slicing a Non-simulated Signal (e.g. Signal Type DRIVE)

1. Create a new signal by clicking the **Add Signal** button.
2. Double-click on the new signal to open the *Signal Properties* dialog.
3. Check the **Use Waveform from Library** checkbox. This will display the Library and Signal boxes in that section of the dialog.
4. Choose a Library. The **Signal List** library contains signals in the current diagram. Other library choices will also be available if you have added waveform libraries to your diagram.
5. Choose a **Signal** from the library. This is the signal to be sliced.
6. Set the MSB and LSB to the desired slice. The bits of slice can also be reversed by swapping the MSB/LSB values (e.g. [7:31] instead of [31:7]).

# 8 Debug Features

This section covers the features which make the HET IDE unique. Along with normal debug features, HET IDE comes with powerful utilities to make it complete. It provides various mechanisms to drive inputs onto pins, to break the execution on certain conditions (making it easy to debug lengthy applications), to change memory at certain actions, and to watch signals during run time.

**Figure 9. Features Available in Debug Mode**




## 8.1 Input Stimulus

HET IDE provides options to generate stimulus on input pins. There are three ways to do this depending on the availability of a SynaptiCad License. The table below shows the options that are available with and without a SynaptiCad License.

| Method | Without License | With License |
|---|---|---|
| IDE Stimuli Creator | Yes | Yes |
| VCD file | Yes | Ignored |
| BTIM file (SynaptiCad) | Ignored | Yes |

The following sections explain these three methods.

### 8.1.1 IDE Stimulus Creator

The HET IDE provides an option to change the pin value at a particular time or clock cycle. The user must specify when to take action (cycle count / time), the pin number, and the value to drive. The user can also repeat the stimuli after a period of time (which can be either cycles or fixed time). This option is available in Debug mode only.

Refer to the *HET IDE Tutorial Guide* (SPNU485) for more details.

**NOTE:** The stimulus applied using this feature can be seen on watch pins (hetm.watch_0.in, hetm.watch_1.in and so on).

### 8.1.2 VCD File (WaveViewer)

Traffic can also be generated on the input ports by using the Stimulus VCD file. When driving the stimulus using the VCD file, the ports should be named as “hetm.drive_0.in”, “hetm.drive_1.in” and so on. A piece of sample code is below.

The full list of signal names and scope to be provided are given in Appendix A.

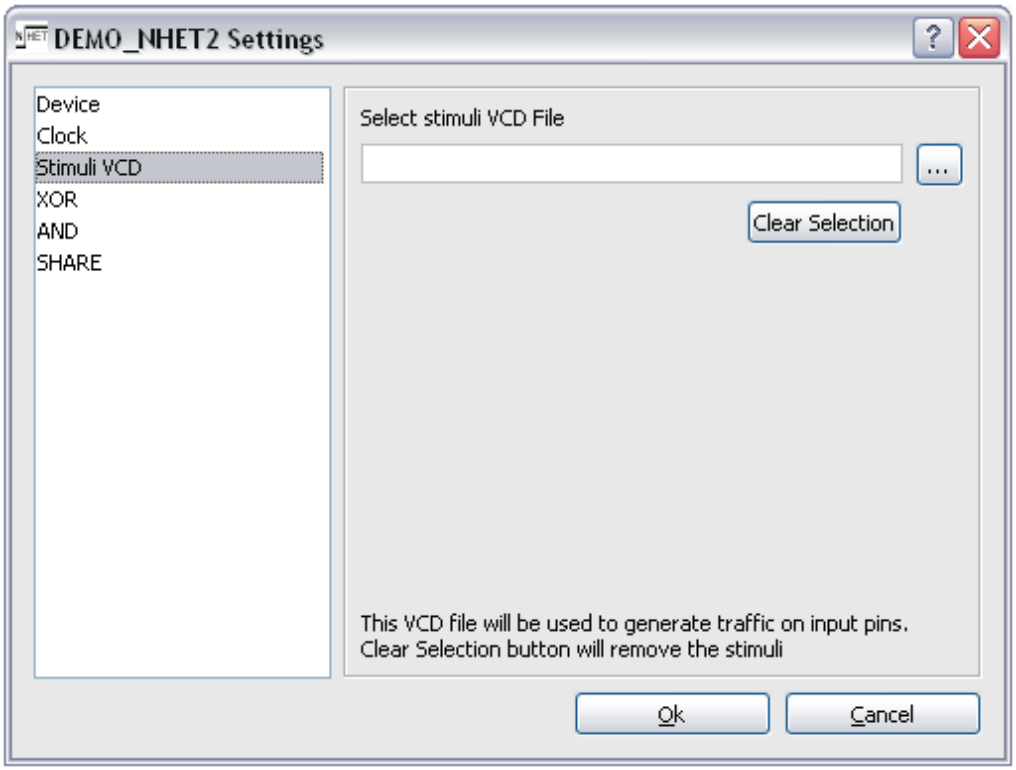
***Sample VCD file header section:***

```
$scope module monitor $end
$scope module synapticad_port_0 $end
$var wire    1 aab out        $end
$upscope $end
$scope module synapticad_port_1 $end
$var wire    1 aad out        $end
$upscope $end
$scope module synapticad_port_2 $end
$var wire    1 aaf out        $end
$upscope $end
$scope module synapticad_port_3 $end
$var wire    1 aah out        $end
$upscope $end
$upscope $end
$enddefinitions $end
```

Go to **Project Properties** → **Stimuli VCD** and provide the path of the Input Stimulus VCD file to generate traffic on input pins and click "OK". This option is available in Edit mode only.

**NOTE:** 'hetm' is the name of the module inside HET IDE which is used to generate stimuli and watch signals on HET pins.

**Figure 10. Set Stimuli VCD File**




Refer to the *HET IDE Tutorial Guide* (SPNU485) for an example.

### 8.1.3 BTIM File (SynaptiCad's WaveFormer)

Input stimuli can be applied to the pins of type **Drive** using the BTIM file. This feature is available only with a SynaptiCad License.

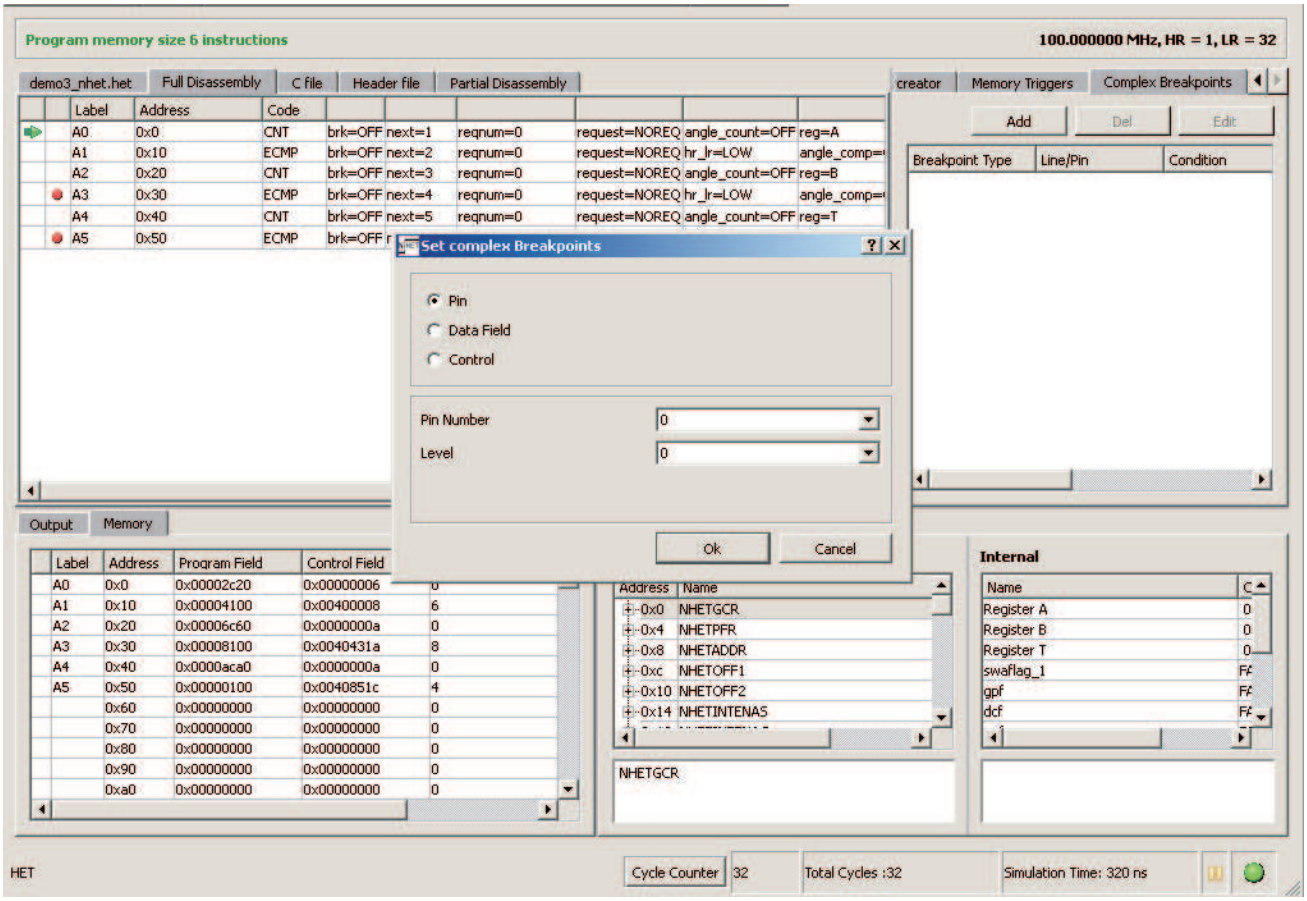
Refer to Section 7.2 for how to draw stimulus on pins.

**NOTE:** The stimulus can always be generated using the Stimulus creator. The options **Stimulus VCD (WaveViewer)** and **BTIM file (WaveFormer)** will be available based on the availability of the SynaptiCad License. WaveViewer is available without a license, whereas WaveFormer is available only with a SynaptiCad license.

**NOTE:** The WaveViewer Free has a limation of being able to process only up to ONE MILLION events (waveform edges).

## 8.2 Advanced Debug Features

**Figure 11. Setting of Complex Breakpoints**




### 8.2.1 Complex Breakpoints

HET IDE provides an option to set breakpoints either on a pin change or on a match of the data field or control field of a line.

During execution, if the data field or control field of a line matches the criteria, the simulation stops at the line. This feature is enabled in Debug mode only.

Refer to the *HET IDE Tutorial Guide* (SPNU485) for more details.

### 8.2.2 Memory Triggers

Sometimes the user might need to modify a particular location of memory when certain conditions occur. This feature helps the user to modify the contents of a particular location in Debug mode. The value at any address (32-bit boundary) can be modified either at a particular time or on a pin action or interrupt event.

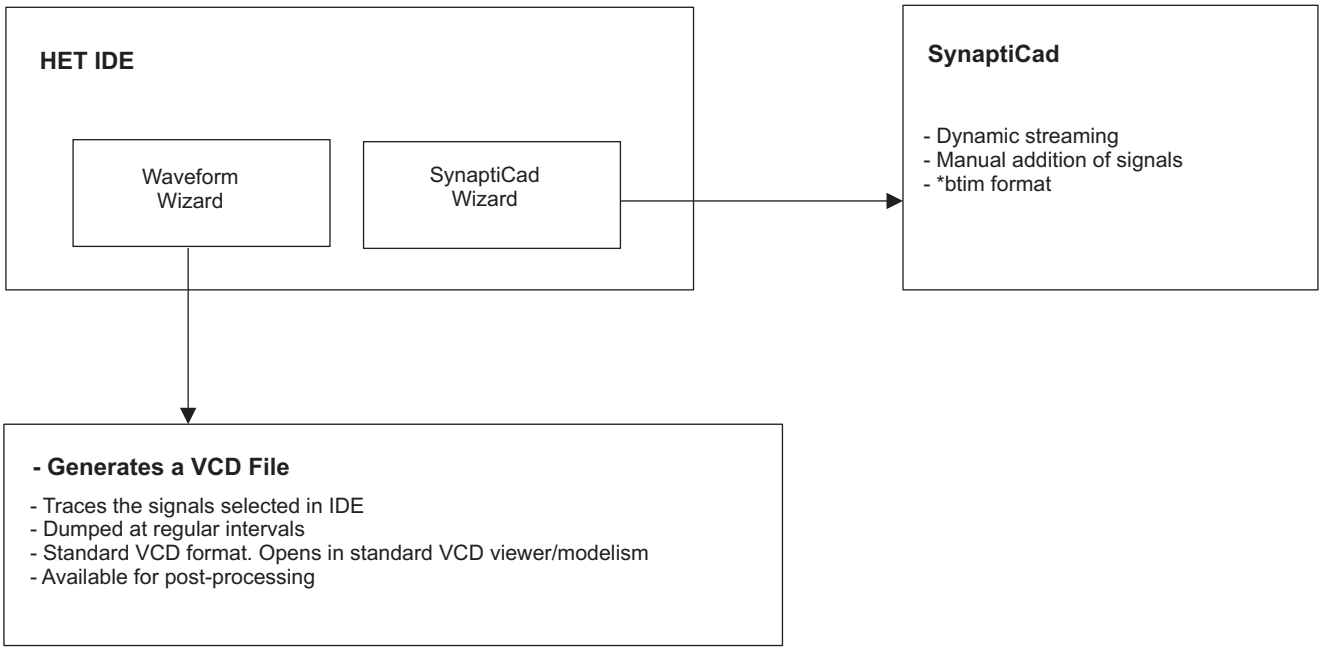
Refer to the *HET IDE Tutorial Guide* (SPNU485) for more details.

**NOTE:** Only one 32-bit word can be written at a time.

## *8.3 Monitor Output*

Outputs of the signals can be monitored in two ways:

1. Dynamic streaming during simulation
2. VCD viewing after simulation




### 8.3.1 VCD Selection

The HET IDE traces the signals and other information into a VCD file. By default, all clocks (HET clock, High Resolution clock, and Low Resolution clock) are traced. However, the user can trace the desired signals (registers, pins, interrupt levels, internal flags and registers, instruction fields, etc.) using the **Waveform Wizard**. Note that this wizard is only useful for VCD dumping.

**NOTE:**

1. If no signals are selected, all three clocks are selected by default.
2. Loop_count signal, which counts the number of the LRPs, is displayed by default.
3. The VCD file is updated at regular intervals of time rather than at every cycle. It may happen that some cycles are missed at the end of the simulation.
4. If clocks are not selected, it is possible that the VCD file is not generated (for small simulation runs). This is the known limitation of the tool.

Refer to the *HET IDE Tutorial Guide* (SPNU485) for more details.

### 8.3.2 Dynamic Streaming

HET IDE provides a feature to view the changes in the values of all signals (clocks, pins, registers, internal registers, etc.) dynamically. The Waveform Viewer window opens when the user clicks the "**Load**" or "**Load and Assemble**" button.

On Load, if you have a SynaptiCad License, *het_signals.btim* file will be opened; without the license, *het_watch_only_signals.btim* file will be opened. This file contains all of the clocks, input pins, output pins, internal registers (a, b, t), internal flags (Z, X), and the NHETADDR register added for dynamic streaming.

Other signals can also be added. Refer to Appendix A for the signal names to be given in the SynaptiCad.

**NOTE:** An error window may appear after loading if the Wave Viewer does not find the *het_signal.btim* file in the project folder.
**Solution:** Copy the default *.btim files from the {Installation}\Resources folder into the project folder.

**NOTE:** Instructions are not added in the default *.btim file. To see the dynamic tracing of instructions, add them in the WaveFormer \ WaveViewer and enable the same in the SynaptiCad Wizard.

### 8.3.3 SynaptiCad Wizard

By default, the instructions that are selected in the Waveform Wizard are not traced in the *.btim file. To view these in the *.btim file, enable the instructions in the SynaptiCad wizard and add these signals to the *.btim file. Refer to Appendix A for the names of the instruction signals.

## *8.4 Automation of HET Execution*

HET can be automated with any GUI automation tools – AutoIt or AutoHot Key. A sample for AutoHotKey is provided in the Examples.

# 9 Other Features

This section describes other features of HET IDE.

## *9.1 Send to HALCoGen*

HALCoGen is a tool used by TI to quickly get started writing software. If a user has a HalCoGen project for a device which has HET, then this feature can be used.

By using this feature, the code from the current project will be copied to the selected HET.c file.

# Appendix A Signal Names in SynaptiCAD

In order to add signals in SynaptiCAD, the following names must be used.

**Table 1. Signals**

| Type | Signal Name | Signal ID (NHET/HET) |
|---|---|---|
| Clocks | HET Clock | het.het_clock |
| | High Resolution Clock | het.high_resolution_clock |
| | Loop Resolution Clock | het.loop_resolution_clock |
| Loop Count (displays count of LRP) | Loop Count | het.loop_count |
| Output Pins (Watch Signals) | HET_0 | hetm.watch_0.in |
| | HET_1 | hetm.watch_1.in |
| | HET_2 | hetm.watch_2.in |
| | HET_3 | hetm.watch_3.in |
| | HET_30 | hetm.watch_30.in |
| | HET_31 | hetm.watch_31.in |
| Input Pins (Watch Signals) | HET_0 | hetm.watch_0.out |
| | HET_1 | hetm.watch_1.out |
| | HET_31 | hetm.watch_31.out |
| Fields | Program | het.curr_program_field |
| | Control | het.curr_control_field |
| | Data | het.curr_data_field |
| Internal Registers | A | het.ir_a |
| | B | het.ir_b |
| | T | het.ir_t |
| Flags | Z | het.flag_z |
| | X | het.flag_x |
| | SWT | het.flag_swt |
| | NAF | het.flag_naf |
| | ACT | het.flag_act |
| | DCF | het.flag_dcf |
| | GPF | het.flag_gpf |
| **Instructions**(1) | Instruction fields of address 0x0 added in waveform wizard | het.mem_0x0_data<br>het.mem_0x0_control<br>het.mem_0x0_program |
| | Instruction fields of address 0x20 added in waveform wizard | het.mem_0x20_data<br>het.mem_0x20_control<br>het.mem_0x20_program |
| | General names of instruction fields at any <address> added in waveform wizard | het.mem_<address>_data<br>het.mem_<address>_control<br>het.mem_<address>_program |
| **Registers**(2) | Depends on the NHET/HET configuration | het.NHETGCR / het.HETGCR |
| | | het.NHETPFR / het.HETPFR |
| | | het.NHETOFF1 / het.HETOFF1 |
| | | het.NHETOFF2 / het.HETOFF2 |
| | | het.NHETINTENAS / het.HETEXC1 |
| | | het.NHETINTENAC / het.HETEXC2 |
| | | het.NHETEXC1 / het.HETPRY |
| | | het.NHETEXC2 / het.HETFLG |

(1) User should enable/add the instructions that are to be watched in the waveform wizard.
(2) User must change the names if the device type is changed.

**Table 1. Signals (continued)**

| Type | Signal Name | Signal ID (NHET/HET) |
|---|---|---|
| | | het.NHETPRY / het.HETAND |
| | | het.NHETFLG / het.HETHRSH |
| | | het.NHETAND / het.HETXOR |
| | | het.NHETHRSH |
| | | het.NHETXOR / het.HETDIR |
| | | het.NHETREQENS / het.HETDIN |
| | | het.NHETREQENC / het.HETDOUT |
| | | het.NHETREQDS / het.HETDSET |
| | | het.NHETDIR / het.HETDCLR |
| | | het.NHETDIN / het.HETPDR |
| | | het.NHETDOUT |
| | | het.NHETDSET / het.HETPULDIS |
| | | het.NHETDCLR / het.HETPSL |
| | | het.NHETPDR / het.HETSRS |
| | | het.NHETPULDIS / het.HETLBPSEL |
| | | het.NHETPSL / het.HETLBPDIR |
| | | het.NHETPCR |
| | | het.NHETPAR / het.HETPCR |
| | | het.NHETPPR / het.HETPIEN |
| | | het.NHETSFPRLD / het.HETPIFLG |
| | | het.NHETSFENA / het.HETPAR |
| | | het.NHETLBPSEL |
| | | het.NHETLBPDIR |
| | | het.NHETPINDIS |

# Appendix B Default Register Settings

Default register configuration files for NHET (NHET_Registers_Default.cfg) and HET (HET_Registers_Default.cfg) are provided in the installation.

**Table 2. Default Register Settings for NHET**

| | | |
|---|---|---|
| 0x0 | NHETGCR | 0x00000001 |
| 0x4 | NHETPFR | 0x00000500 |
| 0x4C | NHETDIR | 0xFFFFFFFF |
| 0x54 | NHETDOUT | 0xFFFFFFFF |
| 0x74 | NHETPCR | 0x00000005 |
| 0x90 | NHETLBPDIR | 0x00050000 |

**Table 3. Default Register Settings for HET**

| | | |
|---|---|---|
| 0x0 | HETGCR | 0x00010001 |
| 0x4 | HETPFR | 0x00000500 |
| 0x20 | HETFLG | 0xFFFFFFFF |
| 0x34 | HETDIR | 0xFFFFFFFF |
| 0x3C | HETDOUT | 0xFFFFFFFF |
| 0x74 | HETPAR | 0x00000005 |

# Appendix C PWM Calculations

When generating PWM signals, all of the frequencies cannot be generated. The following logic is used to calculate the frequency that can be generated for the given inputs.

The HET IDE PWM wizard allows a deviation of ±5% from the required frequency.

**Example:**

For given:

HET Clock Frequency (HCLK) = 100 MHz

HETPFR value = 0x201 → lr = 4 and hr = 2

Possible PWM frequencies that can be generated are -

( HCLK/( lr*hr*n) ) where n is an integer – 1, 2, 3,....

= 12.5 MHz, 6.25 MHz, 4.1666 MHz, 3.125 MHz, 2.5 MHz,.....

Calculating time periods:

HRP = 2/100 MHz = 20 ns

LRP = lr * HRP = 4 * 20 ns = 80 ns

lr_frequency = 1/LRP = 12.5 MHz => Generated PWM frequency will be less than this LR_frequency

If required PWM frequency ( $freq_{req}$ ) is 2 MHZ,

max_cnt = (( lr_frequency / pwm_frequency) – 1) = 5

Actual frequency that can be generated for this value of max_cnt is -

$freq_{act}$ = ( lr_frequency / (max_cnt +1)) = 2.08 MHz

Deviation between $freq_{req}$ and $freq_{act}$ is

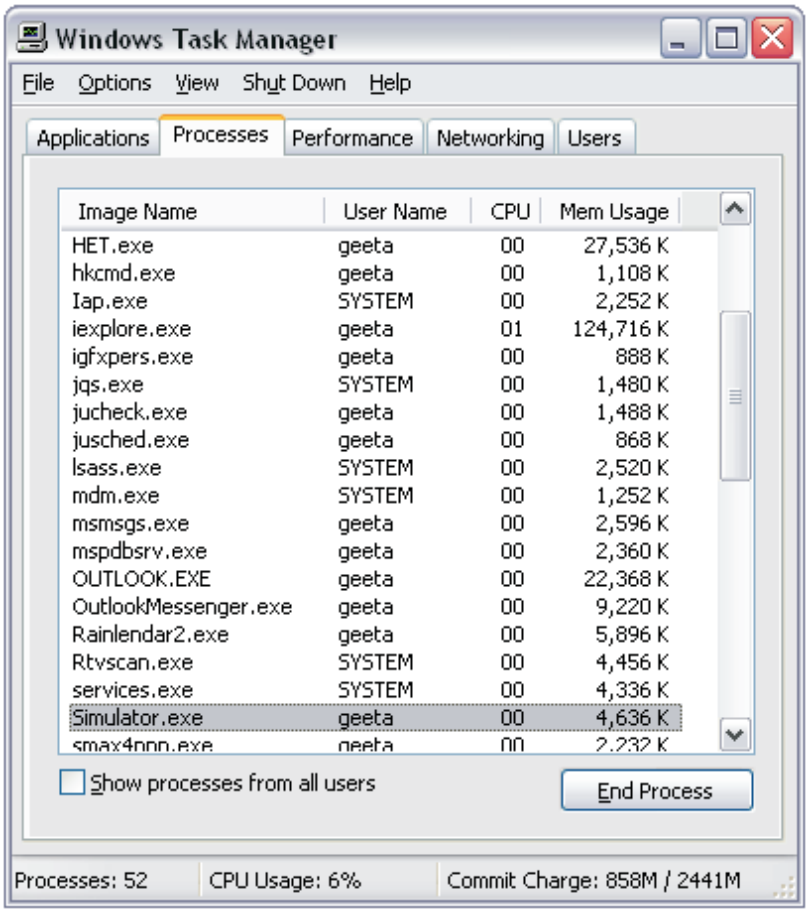
= -4%

Since the deviation is less than -5%, a PWM wave is generated with $freq_{act}$.

If the deviation is above ±5%, the PWM wizard warns the user and shows the nearest possible frequency.

## Appendix D Debugging Hints

1. Make sure that you have configured the device properly (HET/NHET).
   (a) Simulator results are indeterminate if device configuration and code does not match.
2. The HET IDE is a two-process application. In case of unexpected results, both processes must be ended (from the Windows Task Manager).
   (a) HET.exe (Front-end IDE)
   (b) simulator.exe (Back-end Simulator)




3. If the application is not able to start, it may be due to an anti-virus application. Some anti-virus programs or windows firewalls may block the HET application. Please check the status in the anti-virus logs.
4. “SyncadLauncher.dll is not found, reinstalling again may fix the problem” - Make sure that the PATH variable is SET.
5. “Cannot locate waveform viewer” - Make sure that the SYNCAD_HOME environment variable is SET.
6. “bind failed” - Make sure that the port “4242” is available. Use the command “netstat” to check whether this port is occupied or free. If the problem still exists, please send an email to http://e2e.ti.com/support/microcontrollers/default.aspx addressing the issue.
7. When installing on Windows 7 in administrator mode, right-click on the installer and select the “Run as administrator” option for proper installation.

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

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

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