Application Report
SPNA225–June 2015

PWM Generation and Input Capture Using HALCoGen N2HET Module

Vineeth Thomas Alex

ABSTRACT

This application report describes the steps to generate PWM and Input capture using N2HET black-box driver provided by HALCoGen for the Hercules-based MCU's. The content also covers both hardware and software timing restrictions of the N2HET module with the black-box driver configuration.

Contents

List of Figures

List of Tables

1 Introduction

The N2HET module provides timing functions for real-time applications like engine management, motor control and voltage regulation. It allows the generation of high precision PWMs on 32 programmable pins and can be used for input capture utilities such as period and pulse measurements, edge counting and so forth. Hardware Abstraction Layer Code Generator (HALCoGen) provides an easy-to-use software layer for the N2HET to implement these features for entry-level and veteran developers. However, there are certain restrictions on this usage. The whole spectrum of possible Period Times/Pulse Widths is not available to be generated. Similarly, the accuracy of measured input parameters is dependent on various conditions. The following sections describe the functioning of the HALCoGen N2HET black-box driver in detail and deal with the limitations (both hardware and software).

2 HALCoGen N2HET Driver

The N2HET module is a programmable timer with a RISC-based controller. HALCoGen provides the option to use the N2HET driver in two configurations: White Box and Black Box.

In the White Box configuration, you can provide your own N2HET program to be loaded to the N2HET memory. In the Black Box configuration, you can customize the standard N2HET program provided by HALCoGen using the graphical user interface (GUI).

2.1 Steps to Configure Black Box Driver in HALCoGen

Once a new project is created in HALCoGen, these steps can be followed to enable and configure the HALCoGen Black Box driver.

1. Enable HETx driver in the “Driver Enable” tab.
2. Select HETx pins in the “PINMUX” tab.
3. Under the HETx module, make sure the “Enable Advanced Mode/Disable BlackBox Driver” check box is unselected.
4. Eight PWMs can be configured with different time periods and duty cycles in the “PWM 0-7” tab and pins can be assigned to each PWM. Interrupts can be assigned in the “Pwm Interrupts” tab.
5. Edge counting can be configured in the “Edge 0-7” tab and pins can be assigned to each edge capture channel. Interrupts can be assigned to trigger on edges in the “Edge Interrupts” tab.
6. Input signal measurement can be configured in the “Cap 0-7” tab and pins can be assigned to each channel.
7. Pins can be configured for direction, pull features and HR sharing features in the “Pin“ tabs.

3 N2HET Instructions Used in HALCoGen

3.1 For PWM Generation

These are the instructions that are used for the generation of a PWM with variable duty cycle and period.

Table 1. PWM Generation Instructions

Instruction	Description
PWCNT	This instruction controls the active time (based on duty and period), the polarity of the PWM and the output pin to be used.
DJZ	This instruction controls the period time.
MOV64 (for duty cycle update)	This instruction is used to update the PWCNT instruction after every period (the DJZ count runs out). The instruction is modified when the PWM duty cycle value is to be modified.
MOV64 (for period)	This instruction is used to update the DJZ instruction after every period (the DJZ count runs out). This instruction is modified when the PWM period is to be modified.

3.2 For Input Capture

The following instructions are used for the Input capture capability of N2HET module in HALCoGen.

Table 2. Input Capture Instructions

Instruction	Description
ECNT	This instruction is used for edge capture feature. The type of event (rising or falling edge) to be captured and the input pin to be used are the parameters. ECNT also gives the count of events that happened.
PCNT (for duty cycle)	This instruction is used to measure the active time for a captured signal. The polarity of the input signal and the input pin are the parameters for this instruction
PCNT (for period)	This instruction is used to measure the period time for a captured signal. The polarity of the input signal and the input pin are the parameters for this instruction.
WCAP	This instruction is used for getting the time-stamp for a particular event (rising or falling edge).

4 N2HET Instruction Flow Diagram in HALCoGen

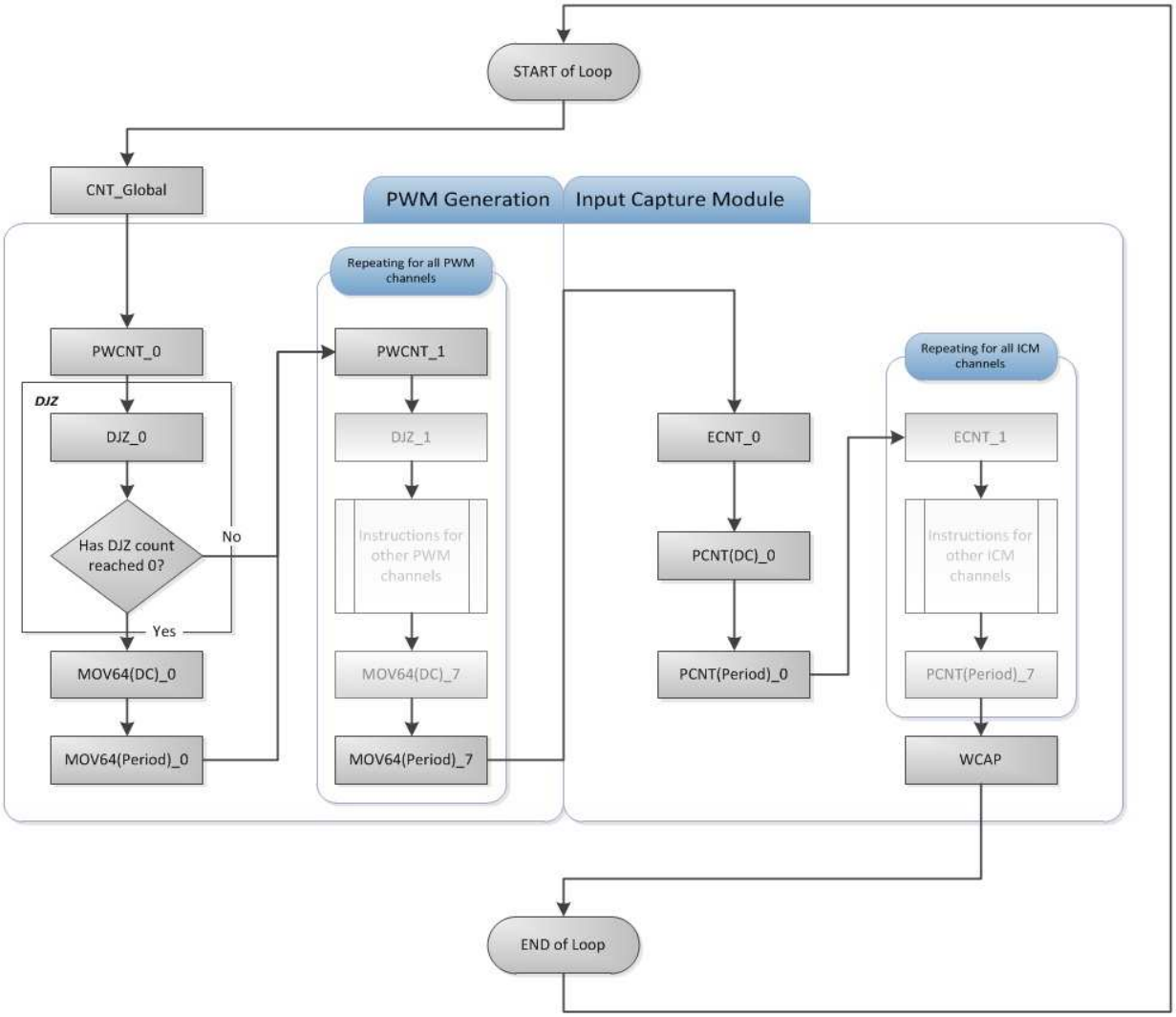

Figure 1. Flow of Control Between N2HET Instructions

The N2HET assembly instruction code for the default configuration of HALCoGen is provided in Appendix A.

5 N2HET Driver as a Hardware Timer

The HALCoGen N2HET black-box driver can be thought of as a hardware timer for ease of understanding. The following block diagram expresses the black-box driver (only PWM generation) in these terms treating instructions fields as traditional hardware timer registers.

The instruction labels correspond to the N2HET assembly code provided in Appendix A.

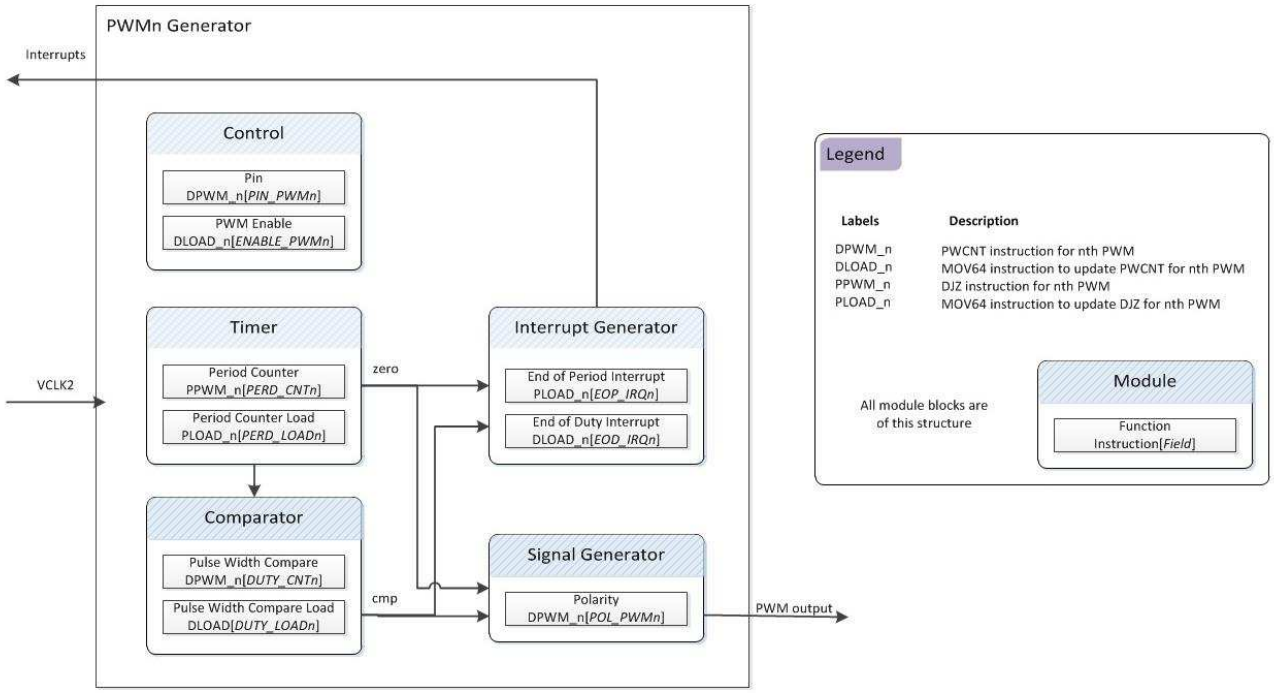

Figure 2. Block Diagram of N2HET Driver as Hardware Timer

The timing controls for the period and duty cycle of the nth PWM is controlled by the PPWM_n and DPWM_n instructions. The values for the instructions are reloaded to the instruction fields at the end of the period by the PLOAD_n and DLOAD_n instructions, respectively. The output pin associated with the PWM and the polarity of the PWM signal is determined by the DPWM_n instruction. The DLOAD_n instruction allows an option to disable the PWM output. The PLOAD_n and DLOAD_n instructions define whether the PWM interrupts are enabled.

NOTE: The HALCoGen APIs use the HET Interrupt Enable Set Register and the HET Interrupt Enable Clear Register to enable and disable N2HET interrupts instead of modifying the above instructions.

6 Timing Limitations

6.1 Basic Assumptions

For the sake of simplicity, assume the following values for useful calculations and explanations.

$VCLK_2$ = 90 MHz

High Resolution Pre-Scale Factor(*hr*) = 1

Loop Resolution Pre-Scale Factor(*lr*) = 128

These assumptions can be extended to the following.

T_{VCLK2} = 1000/90 ns = 11.11ns

High Resolution Period(HRP) = *hr* * T_{VCLK2} = 11.11 ns

Loop Resolution Period(LRP) = *lr* * HRP = 1422.22 ns

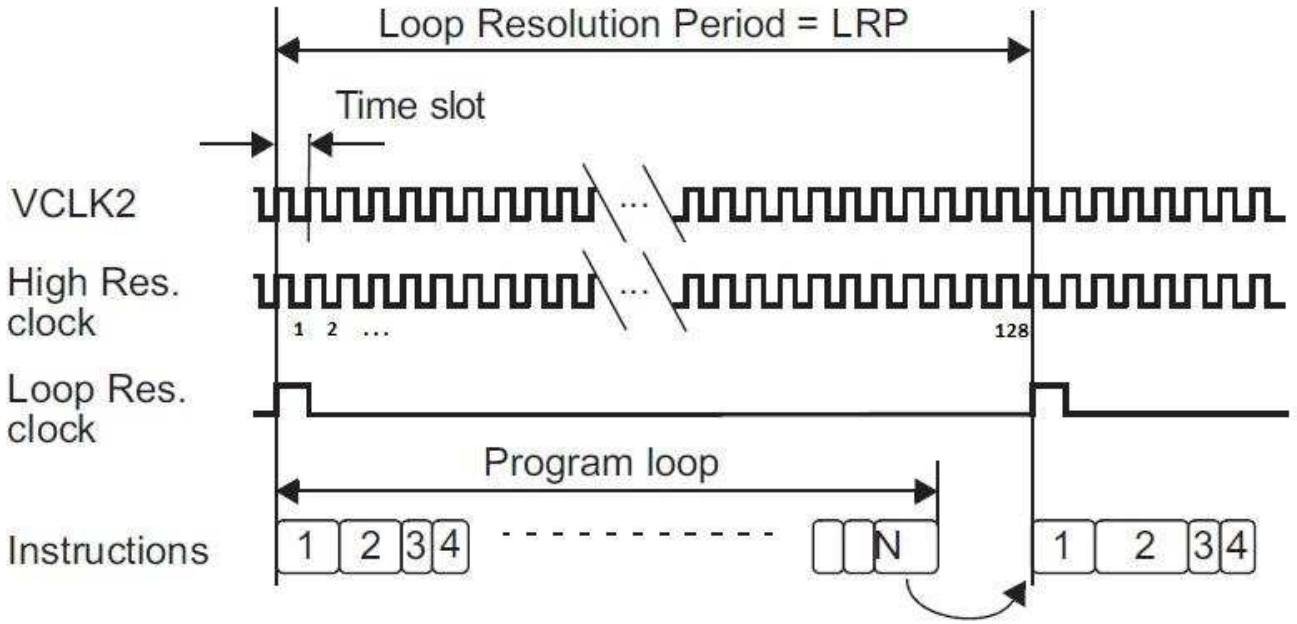

Figure 3. Relation between VCLK2, HRP and LRP

All experiments and measurements are based on TMS570LS3137ZWT MCU.

NOTE: The following limitations on performance are caused mainly because HALCoGen does not use HR instructions. For better performance on producing PWMs, see Appendix B.

6.2 Timing Limitations on PWM Generation

6.2.1 Maximum Period - 0x01FFFFFF Times the LRP

The PWM period is determined by the DJZ instruction, which has a data field of 25 bits that is decremented at every loop. The time period ends when the count strikes 0. Then the MOV64 instruction loads the period value again into the data field and a new period starts. If the maximum value in the data field (0x01FFFFFF) is loaded, the total period will be 0x01FFFFFF times the LRP.

So, in this case:

```
Max Period  =  0x1FFFFFF  x  LRP
           =  33554431   x  1422.22 ns
           =  47721851 us  (experimental values coincide with this value)
```

6.2.2 Minimum Period - 4 Times the LRP

The LRP is the smallest unit for the N2HET (since HALCoGen does not use HR instructions). So if the period value was equal to one LRP, the duty cycle values possible would be 0% and 100%. Similarly, if the period value was 2 LRPs, then the duty values possible are 0%, 50% and 100%. If the period was 3 LRPs, the duty values possible would be 0%,33%,66% and 100%.

In HALCoGen, the minimum period to 4 LRPs is restricted, so the possible values for the duty cycle are 0% ,25%, 50%, 75% and 100%.

In this case:

```
Min Period  =  4  x  LRP
            =  4  x  1422.22 ns
            =  5.689 us     (experimental values coincide with this value)
```

NOTE: The HALCoGen API *pwmSetSignal()* allows you to generate PWMs with periods 1, 2 and 3 times the LRP. But this will be at the expense of resolution as explained above.

6.2.3 Resolution - 1 LRP

Since the LRP is the smallest unit for the N2HET, all period and duty durations will be multiples of this value. The minimum difference possible between two consecutive values is equal to one LRP. The smaller the LRP means the better the resolution, but also the smaller the number of instructions possible on the N2HET. HALCoGen does not impose any more restrictions on the resolution of generated PWM configuration.

6.3 Timing Limitations on Input Capture Module

6.3.1 Signal Measurement - Max Period - 0x01FFFFFF Times the LRP

In signal measurement, the maximum period that can be measured is similar to the maximum period PWM, which can be produced by the N2HET. The PCNT instruction that is used to measure the PWM pulses has a Data field of 25 bits. If the input PWM has a higher period than this maximum, then the measured period will be equal to the maximum period (no warnings will be issued by HALCoGen). Also in this case, the measured duty cycle will be inaccurate.

```
Max Period  =  0x1FFFFFF  x  LRP
            =  33554431   x  1422.22 ns
            =  47721851 us  (experimental values coincide with this value
```

6.3.2 Signal Measurement - Min Period - 1 LRP

The minimum period that can be measured is equal to 1 LRP of the N2HET. If the period of the input PWM is less than the LRP, the measured period will be zero. The measured duty cycle also follows this rule. This means, for a signal with the minimum period, the duty cycle will either be 100% or 0%.

```
Min Period  =  1  x  LRP
            =  1422.22 ns  (experimental values coincide with this value)
```

6.3.3 Signal Measurement - Resolution - 1 LRP

The measured period will always be a multiple of the LRP. For all input signals (with periods less than the maximum measurable period), the measured period will be the highest multiple of the LRP that is less than the actual period. So, the closest measurable periods will differ by 1 LRP.

```
Min Difference  =  1  x  LRP
                =  1422.22 ns  (experimental values coincide with this value)
```

6.3.4 Edge Counting - Minimum Delay Required Between Successive Edges - 1 LRP

For accurate capture of edges, the N2HET hardware imposes a restriction. The minimum time period between edges should be more than the LRP (there may be additional delay required depending on the device variant) For more information, see the device-specific data sheet). HALCoGen does not enforce anymore constraints on this parameter.

```
Min Delay    =  1  x  LRP
             =  1422.22 ns  (experimental values coincide with this value)
```

6.4 Summary of Timing Limitations

Table 3. Summary

Device	PWM			ICM			
	Initial Generation			Signal Measurement			Edge Counting
	Max Period	Min Period	Resolution	Max Period	Min Period	Resolution	Min Delay Required between edges
General Case	0x01FFFFFF times LRP	4 times LRP	LRP	0x01FFFFFF times LRP	LRP	LRP	LRP
TMS570LS3137ZWT							
with given assumptions	47721851 us	5689 ns	1422 ns	47721851 us	1422 ns	1422 ns	1422 ns

7 References

- TMS570LS31x/21x 16/32-Bit RISC Flash Microcontroller Technical Reference Manual (SPNU499)
- TMS570LS31x4/21x4 16- and 32-Bit RISC Flash Microcontroller Data Sheet (SPNS165)
- HET IDE (NHET Assembler) http://www.ti.com/tool/HET_IDE

HALCoGen N2HET Assembly Code

The following is the N2HET assembly code equivalent of the HALCoGen black-box driver code for the default configuration (with Basic Assumptions). It can be copied to NHET Assembler for better understanding of how the HALCoGen N2HET module works.

```
;/*****************************************************************************************/
; This HET program corresponds to the HALCoGen generated HET code for TMS570LS31xx in default   /
; configuration                                                                               /
;/*****************************************************************************************/
;/----------------------------------------------------------------------------------------/
;/  GUI CONFIGURATION                                                                     /
;/----------------------------------------------------------------------------------------/

; PWM Configuration Parameters
PIN_PWM0         .equ     8
DUTY_LOAD0       .equ    353
PERD_LOAD0       .equ    703
    .asg         "PULSEHI", POL_PWM0
    .asg         "OFF", ENABLE_PWM0
PIN_PWM1         .equ    10
DUTY_LOAD1       .equ    353
PERD_LOAD1       .equ    703
    .asg         "PULSEHI", POL_PWM1
    .asg         "OFF", ENABLE_PWM1
PIN_PWM2         .equ    12
DUTY_LOAD2       .equ    353
PERD_LOAD2       .equ    703
    .asg         "PULSEHI", POL_PWM2
    .asg         "OFF", ENABLE_PWM2
PIN_PWM3         .equ    14
DUTY_LOAD3       .equ    353
PERD_LOAD3       .equ    703
    .asg         "PULSEHI", POL_PWM3
    .asg         "OFF", ENABLE_PWM3
PIN_PWM4         .equ    16
DUTY_LOAD4       .equ    353
PERD_LOAD4       .equ    703
    .asg         "PULSEHI", POL_PWM4
    .asg         "OFF", ENABLE_PWM4
PIN_PWM5         .equ    17
DUTY_LOAD5       .equ    353
PERD_LOAD5       .equ    703
    .asg         "PULSEHI", POL_PWM5
    .asg         "OFF", ENABLE_PWM5
PIN_PWM6         .equ    18
DUTY_LOAD6       .equ    353
PERD_LOAD6       .equ    703
    .asg         "PULSEHI", POL_PWM6
    .asg         "OFF", ENABLE_PWM6
PIN_PWM7         .equ    19
DUTY_LOAD7       .equ    353
PERD_LOAD7       .equ    703
    .asg         "PULSEHI", POL_PWM7
    .asg         "OFF", ENABLE_PWM7

; EDGE Configuration Parameters
```

```
PIN_EDGE0        .equ    9
    .asg         "FALL", POL_EDGE0
PIN_EDGE1        .equ    11
    .asg         "FALL", POL_EDGE1
PIN_EDGE2        .equ    13
    .asg         "FALL", POL_EDGE2
PIN_EDGE3        .equ    15
    .asg         "FALL", POL_EDGE3
PIN_EDGE4        .equ    20
    .asg         "FALL", POL_EDGE4
PIN_EDGE5        .equ    21
    .asg         "FALL", POL_EDGE5
PIN_EDGE6        .equ    22
    .asg         "FALL", POL_EDGE6
PIN_EDGE7        .equ    23
    .asg         "FALL", POL_EDGE7

; CAPTURE Configuration Parameters
PIN_DUTYCAP0     .equ     0
PIN_PERDCAP0     .equ     (PIN_DUTYCAP0+1)
    .asg         "FALL2RISE", POL_DCAP0
    .if $symcmp( POL_DCAP0, "FALL2RISE" )
    .asg         "RISE2RISE", POL_PCAP0
    .else
    .asg         "FALL2FALL", POL_PCAP0
    .endif
PIN_DUTYCAP1      .equ     2
PIN_PERDCAP1      .equ     (PIN_DUTYCAP1+1)
    .asg         "FALL2RISE", POL_DCAP1
    .if $symcmp( POL_DCAP1, "FALL2RISE" )
    .asg         "RISE2RISE", POL_PCAP1
    .else
    .asg         "FALL2FALL", POL_PCAP1
    .endif
PIN_DUTYCAP2      .equ     4
PIN_PERDCAP2      .equ     (PIN_DUTYCAP2+1)
    .asg         "FALL2RISE", POL_DCAP2
    .if $symcmp( POL_DCAP2, "FALL2RISE" )
    .asg         "RISE2RISE", POL_PCAP2
    .else
    .asg         "FALL2FALL", POL_PCAP2
    .endif
PIN_DUTYCAP3      .equ    6
PIN_PERDCAP3      .equ    (PIN_DUTYCAP3+1)
    .asg          "FALL2RISE", POL_DCAP3
    .if $symcmp( POL_DCAP3, "FALL2RISE" )
    .asg          "RISE2RISE", POL_PCAP3
    .else
    .asg          "FALL2FALL", POL_PCAP3
    .endif
PIN_DUTYCAP4      .equ   24
PIN_PERDCAP4      .equ   (PIN_DUTYCAP4+1)
    .asg          "FALL2RISE", POL_DCAP4
    .if $symcmp( POL_DCAP4, "FALL2RISE" )
    .asg          "RISE2RISE", POL_PCAP4
    .else
    .asg          "FALL2FALL", POL_PCAP4
    .endif
PIN_DUTYCAP5      .equ   26
PIN_PERDCAP5      .equ   (PIN_DUTYCAP5+1)
    .asg          "FALL2RISE", POL_DCAP5
    .if $symcmp( POL_DCAP5, "FALL2RISE" )
    .asg          "RISE2RISE", POL_PCAP5
    .else
    .asg          "FALL2FALL", POL_PCAP5
    .endif
```

```
PIN_DUTYCAP6      .equ   28
PIN_PERDCAP6      .equ   (PIN_DUTYCAP6+1)
    .asg          "FALL2RISE", POL_DCAP6
    .if $symcmp( POL_DCAP6, "FALL2RISE" )
    .asg          "RISE2RISE", POL_PCAP6
    .else
    .asg          "FALL2FALL", POL_PCAP6
    .endif
PIN_DUTYCAP7      .equ   30
PIN_PERDCAP7      .equ   (PIN_DUTYCAP7+1)
    .asg          "FALL2RISE", POL_DCAP7
    .if $symcmp( POL_DCAP7, "FALL2RISE" )
    .asg          "RISE2RISE", POL_PCAP7
    .else
    .asg          "FALL2FALL", POL_PCAP7
    .endif


;/-------------------------------------------------------------------------------------------/
;/  ADDITIONAL CONFIGURATION                                                                 /
;/-------------------------------------------------------------------------------------------/
;The following configuration options are not available in the HALCoGen GUI. The default values
;here are hard-coded in the ;actual HALCoGen black-box driver.

PERD_CNT0        .equ 0
DUTY_CNT0        .equ 0
    .asg         "ON", EOD_IRQ0
    .asg         "ON", EOP_IRQ0
PERD_CNT1        .equ 0
DUTY_CNT1        .equ 0
    .asg         "ON", EOD_IRQ1
    .asg         "ON", EOP_IRQ1
PERD_CNT2        .equ 0
DUTY_CNT         .equ 0
    .asg         "ON", EOD_IRQ2
    .asg         "ON", EOP_IRQ2
PERD_CNT3        .equ 0
DUTY_CNT         .equ 0
    .asg         "ON", EOD_IRQ3
    .asg         "ON", EOP_IRQ3
PERD_CNT4        .equ 0
DUTY_CNT4        .equ 0
    .asg         "ON", EOD_IRQ4
    .asg         "ON", EOP_IRQ4
PERD_CNT5        .equ 0
DUTY_CNT5        .equ 0
    .asg         "ON", EOD_IRQ5
    .asg         "ON", EOP_IRQ5
PERD_CNT6        .equ 0
DUTY_CNT6        .equ 0
    .asg         "ON", EOD_IRQ6
    .asg         "ON", EOP_IRQ6
PERD_CNT7        .equ 0
DUTY_CNT7        .equ 0
    .asg         "ON", EOD_IRQ7
    .asg         "ON", EOP_IRQ7
;/-------------------------------------------------------------------------------------------/
;/  INSTRUCTIONS                                                                             /
;/-------------------------------------------------------------------------------------------/

; The N2HET loop starts at FIRST_INS instruction
; Global CNT instruction
FIRST_INS  CNT { next=DPWM_0,reg=T,comp=EQ,irq=OFF,max=33554431,data=33554431};

; DPWM_0 to PPWM_7 are instructions to produce PWM signals
DPWM_0  PWCNT { next=PPWM_0, hr_lr=LOW, cond_addr=PPWM_0, pin=PIN_PWM0, action=POL_PWM0,
```

```
reg=NONE, irq=OFF, data=DUTY_CNT0, hr_data=0};
PPWM_0  DJZ { next=DPWM_1, cond_addr=DLOAD_0, reg=NONE, irq=OFF, data=PERD_CNT0};
DPWM_1  PWCNT { next=PPWM_1, hr_lr=LOW, cond_addr=PPWM_1, pin=PIN_PWM1, action=POL_PWM1,
reg=NONE, irq=OFF, data=DUTY_CNT1, hr_data=0};
PPWM_1  DJZ { next=DPWM_2, cond_addr=DLOAD_1, reg=NONE, irq=OFF, data=PERD_CNT1};
DPWM_2  PWCNT { next=PPWM_2, hr_lr=LOW, cond_addr=PPWM_2, pin=PIN_PWM2, action=POL_PWM2,
reg=NONE, irq=OFF, data=DUTY_CNT2, hr_data=0};
PPWM_2  DJZ { next=DPWM_3, cond_addr=DLOAD_2, reg=NONE, irq=OFF, data=PERD_CNT2};
DPWM_3  PWCNT { next=PPWM_3, hr_lr=LOW, cond_addr=PPWM_3, pin=PIN_PWM3, action=POL_PWM3,
reg=NONE, irq=OFF, data=DUTY_CNT3, hr_data=0};
PPWM_3  DJZ { next=DPWM_4, cond_addr=DLOAD_3, reg=NONE, irq=OFF, data=PERD_CNT3};
DPWM_4  PWCNT { next=PPWM_4, hr_lr=LOW, cond_addr=PPWM_4, pin=PIN_PWM4, action=POL_PWM4,
reg=NONE, irq=OFF, data=DUTY_CNT4, hr_data=0};
PPWM_4  DJZ { next=DPWM_5, cond_addr=DLOAD_4, reg=NONE, irq=OFF, data=PERD_CNT4};
DPWM_5  PWCNT { next=PPWM_5, hr_lr=LOW, cond_addr=PPWM_5, pin=PIN_PWM5, action=POL_PWM5,
reg=NONE, irq=OFF, data=DUTY_CNT5, hr_data=0};
PPWM_5  DJZ { next=DPWM_6, cond_addr=DLOAD_5, reg=NONE, irq=OFF, data=PERD_CNT5};
DPWM_6  PWCNT { next=PPWM_6, hr_lr=LOW, cond_addr=PPWM_6, pin=PIN_PWM6, action=POL_PWM6,
reg=NONE, irq=OFF, data=DUTY_CNT6, hr_data=0};
PPWM_6  DJZ { next=DPWM_7, cond_addr=DLOAD_6, reg=NONE, irq=OFF, data=PERD_CNT6};
DPWM_7  PWCNT { next=PPWM_7, hr_lr=LOW, cond_addr=PPWM_7, pin=PIN_PWM7, action=POL_PWM7,
reg=NONE, irq=OFF, data=DUTY_CNT7, hr_data=0};
PPWM_7  DJZ { next=EDGE_0, cond_addr=DLOAD_7, reg=NONE, irq=OFF, data=PERD_CNT7};

; EDGE0 to EDGE7 are for Edge capturing and counting
EDGE_0  ECNT { next=EDGE_1, cond_addr=EDGE_1, pin=PIN_EDGE0, event=POL_EDGE0, reg=NONE, irq=ON,
data=0};
EDGE_1  ECNT { next=EDGE_2, cond_addr=EDGE_2, pin=PIN_EDGE1, event=POL_EDGE1, reg=NONE, irq=ON,
data=0};
EDGE_2  ECNT { next=EDGE_3, cond_addr=EDGE_3, pin=PIN_EDGE2, event=POL_EDGE2, reg=NONE, irq=ON,
data=0};
EDGE_3  ECNT { next=EDGE_4, cond_addr=EDGE_4, pin=PIN_EDGE3, event=POL_EDGE3, reg=NONE, irq=ON,
data=0};
EDGE_4  ECNT { next=EDGE_5, cond_addr=EDGE_5, pin=PIN_EDGE4, event=POL_EDGE4, reg=NONE, irq=ON,
data=0};
EDGE_5  ECNT { next=EDGE_6, cond_addr=EDGE_6, pin=PIN_EDGE5, event=POL_EDGE5, reg=NONE, irq=ON,
data=0};
EDGE_6  ECNT { next=EDGE_7, cond_addr=EDGE_7, pin=PIN_EDGE6, event=POL_EDGE6, reg=NONE, irq=ON,
data=0};
EDGE_7  ECNT { next=DUTYCAP_0, cond_addr=DUTYCAP_0, pin=PIN_EDGE7, event=POL_EDGE7, reg=NONE,
irq=ON, data=0};

; DUTYCAP_0 to PERDCAP_7 are for Signal Capture and Measurement
DUTYCAP_0  PCNT { next=PERDCAP_0, irq=OFF, type=POL_DCAP0, pin=PIN_DUTYCAP0, period=0, data=0};
PERDCAP_0  PCNT { next=DUTYCAP_1, irq=OFF, type=POL_PCAP0, pin=PIN_PERDCAP0, period=0, data=0};
DUTYCAP_1  PCNT { next=PERDCAP_1, irq=OFF, type=POL_DCAP1, pin=PIN_DUTYCAP1, period=0, data=0};
PERDCAP_1  PCNT { next=DUTYCAP_2, irq=OFF, type=POL_PCAP1, pin=PIN_PERDCAP1, period=0, data=0};
DUTYCAP_2  PCNT { next=PERDCAP_2, irq=OFF, type=POL_DCAP2, pin=PIN_DUTYCAP2, period=0, data=0};
PERDCAP_2  PCNT { next=DUTYCAP_3, irq=OFF, type=POL_PCAP2, pin=PIN_PERDCAP2, period=0, data=0};
DUTYCAP_3  PCNT { next=PERDCAP_3, irq=OFF, type=POL_DCAP3, pin=PIN_DUTYCAP3, period=0, data=0};
PERDCAP_3  PCNT { next=DUTYCAP_4, irq=OFF, type=POL_PCAP3, pin=PIN_PERDCAP3, period=0, data=0};
DUTYCAP_4  PCNT { next=PERDCAP_4, irq=OFF, type=POL_DCAP4, pin=PIN_DUTYCAP4, period=0, data=0};
PERDCAP_4  PCNT { next=DUTYCAP_5, irq=OFF, type=POL_PCAP4, pin=PIN_PERDCAP4, period=0, data=0};
DUTYCAP_5  PCNT { next=PERDCAP_5, irq=OFF, type=POL_DCAP5, pin=PIN_DUTYCAP5, period=0, data=0};
PERDCAP_5  PCNT { next=DUTYCAP_6, irq=OFF, type=POL_PCAP5, pin=PIN_PERDCAP5, period=0, data=0};
DUTYCAP_6  PCNT { next=PERDCAP_6, irq=OFF, type=POL_DCAP6, pin=PIN_DUTYCAP6, period=0, data=0};
PERDCAP_6  PCNT { next=DUTYCAP_7, irq=OFF, type=POL_PCAP6, pin=PIN_PERDCAP6, period=0, data=0};
DUTYCAP_7  PCNT { next=PERDCAP_7, irq=OFF, type=POL_DCAP7, pin=PIN_DUTYCAP7, period=0, data=0};
PERDCAP_7  PCNT { next=TSTMP, irq=OFF, type=POL_PCAP7, pin=PIN_PERDCAP7, period=0, data=0};

; DLOAD_0 to PLOAD_7 are for updating the instructions DPWM_0 to PPWM_7 when the DJZ counts reach
zero(i.e. one time period ends)
DLOAD_0  MOV64 { next=PLOAD_0, remote=DPWM_0, en_pin_action=ENABLE_PWM0, cond_addr=PPWM_0,
pin=PIN_PWM0, comp_mode=ECMP, action=POL_PWM0, reg=NONE,irq=EOD_IRQ0,data=DUTY_LOAD0};
PLOAD_0  MOV64 { next=DPWM_1, remote=PPWM_0, cond_addr=DLOAD_0, comp_mode=ECMP, reg=NONE,
irq=EOP_IRQ0, data=PERD_LOAD0};
```

```
DLOAD_1  MOV64 { next=PLOAD_1, remote=DPWM_1,en_pin_action=ENABLE_PWM1,cond_addr=PPWM_1,
pin=PIN_PWM1, comp_mode=ECMP, action=POL_PWM1, reg=NONE, irq=EOD_IRQ1, data=DUTY_LOAD1};
PLOAD_1  MOV64 { next=DPWM_2, remote=PPWM_1, cond_addr=DLOAD_1, comp_mode=ECMP, reg=NONE,
irq=EOP_IRQ1, data=PERD_LOAD1};
DLOAD_2  MOV64 { next=PLOAD_2, remote=DPWM_2,en_pin_action=ENABLE_PWM2,cond_addr=PPWM_2,
pin=PIN_PWM2, comp_mode=ECMP, action=POL_PWM2, reg=NONE, irq=EOD_IRQ2, data=DUTY_LOAD2};
PLOAD_2  MOV64 { next=DPWM_3, remote=PPWM_2, cond_addr=DLOAD_2, comp_mode=ECMP, reg=NONE,
irq=EOP_IRQ2, data=PERD_LOAD2};
DLOAD_3  MOV64 { next=PLOAD_3, remote=DPWM_3,en_pin_action=ENABLE_PWM3,cond_addr=PPWM_3,
pin=PIN_PWM3, comp_mode=ECMP, action=POL_PWM3, reg=NONE, irq=EOD_IRQ3, data=DUTY_LOAD3};
PLOAD_3  MOV64 { next=DPWM_4, remote=PPWM_3, cond_addr=DLOAD_3, comp_mode=ECMP, reg=NONE,
irq=EOP_IRQ3, data=PERD_LOAD3};
DLOAD_4  MOV64 { next=PLOAD_4, remote=DPWM_4,en_pin_action=ENABLE_PWM4,cond_addr=PPWM_4,
pin=PIN_PWM4, comp_mode=ECMP, action=POL_PWM4, reg=NONE, irq=EOD_IRQ4, data=DUTY_LOAD4};
PLOAD_4  MOV64 { next=DPWM_5, remote=PPWM_4, cond_addr=DLOAD_4, comp_mode=ECMP, reg=NONE,
irq=EOP_IRQ4, data=PERD_LOAD4};
DLOAD_5  MOV64 { next=PLOAD_5, remote=DPWM_5,en_pin_action=ENABLE_PWM5,cond_addr=PPWM_5,
pin=PIN_PWM5, comp_mode=ECMP, action=POL_PWM5, reg=NONE, irq=EOD_IRQ5, data=DUTY_LOAD5};
PLOAD_5  MOV64 { next=DPWM_6, remote=PPWM_5, cond_addr=DLOAD_5, comp_mode=ECMP, reg=NONE,
irq=EOP_IRQ5, data=PERD_LOAD5};
DLOAD_6  MOV64 { next=PLOAD_6, remote=DPWM_6,en_pin_action=ENABLE_PWM6,cond_addr=PPWM_6,
pin=PIN_PWM6, comp_mode=ECMP, action=POL_PWM6, reg=NONE, irq=EOD_IRQ6, data=DUTY_LOAD6};
PLOAD_6  MOV64 { next=DPWM_7, remote=PPWM_6, cond_addr=DLOAD_6, comp_mode=ECMP, reg=NONE,
irq=EOP_IRQ6, data=PERD_LOAD6};
DLOAD_7  MOV64 { next=PLOAD_7, remote=DPWM_7,en_pin_action=ENABLE_PWM7,cond_addr=PPWM_7,
pin=PIN_PWM7, comp_mode=ECMP, action=POL_PWM7, reg=NONE, irq=EOD_IRQ7, data=DUTY_LOAD7};
PLOAD_7  MOV64 { next=EDGE_0, remote=PPWM_7, cond_addr=DLOAD_7, comp_mode=ECMP, reg=NONE,
irq=EOP_IRQ7, data=PERD_LOAD7};

; TSTMP is for time-stamping
TSTMP  WCAP { next=FIRST_INS,cond_addr=FIRST_INS,pin=0,event=NOCOND,reg=T,data=0};
; Instruction TSTMP points back to instruction FIRST_INS and the loop restarts

;/*****************************************************************************************/
;/ END OF FILE                                                                            /
;/*****************************************************************************************/
```

Sample N2HET Program

The limitations described in this document are derived from the fact that HALCoGen does not exploit HR instructions. The following is a sample N2HET program which uses HR instructions and goes beyond these limitations.

```
; Sample NHET Program
; CNT instruction controls period of the PWM
L00  CNT { next=L01, reg=A, irq=OFF, max=0};
; ECMP instruction controls duty time
L01  ECMP { next=L00, hr_lr=HIGH, en_pin_action=ON, cond_addr=L00, pin=0, action=PULSEHI, reg=A,
irq=OFF, data=0, hr_data=0x20};
```

This program produces a PWM with period of 1 LRP and Duty cycle of 25% (1 HRP) on pin N2HET[0]. The Basic Assumptions are redefined as follows:

$VCLK_2$ = 90 MHz

High Resolution Pre-Scale Factor(*hr*) = 1

Loop Resolution Pre-Scale Factor(*lr*) = 4

T_{VCLK2} = 1000/90 ns = 11.11 ns

High Resolution Period(HRP) = *hr* * T_{VCLK2} = 11.11 ns

Loop Resolution Period(LRP) = lr * HRP = 44.44 ns

```
PWM Period  =   1  x  LRP
            =   44.44 ns
PWM Duty    =   1  x  HRP
            =   11.11 ns
```

Note that the rising time and falling time for the signal are significant.

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