Application Report
SPNA069 – May 2005

High End Timer (HET) Getting Started Guide

John Mangino
TMS470 Applications

ABSTRACT

This document serves as a guide to begin developing designs with the High End Timer (HET) advanced intelligent timer module. This timer module provides sophisticated timing functions for real-time applications. The following pages will cover a step-by-step process to program the HET. Starting with the setup for the IAR tools, descriptions of the code for the main program necessary to load the HET and the HET code itself.

Contents

1 Overview 1
2 Code for Actual Use 6
3 References 6

List of Figures

1 HET Coding Sequence 2
2 Batch Method vs. Custom Build Method 4

1 Overview

The HET is a micro machine and programmed with its own instruction set. The HET code is assembled and generates a C program file (xxx.c) and a header file (xxx.h). The C and header files are linked into the main program and then compiled into the main program. The main program upon execution loads the HET RAM and initiates the HET to start running. The main program can modify HET function and parameters by modifying the HET program RAM and the HET control registers. The HET operates independently of the CPU. CPU intervention is only required to modify HET function or read HET data. Block diagram for the coding sequence illustrated in Figure 1.

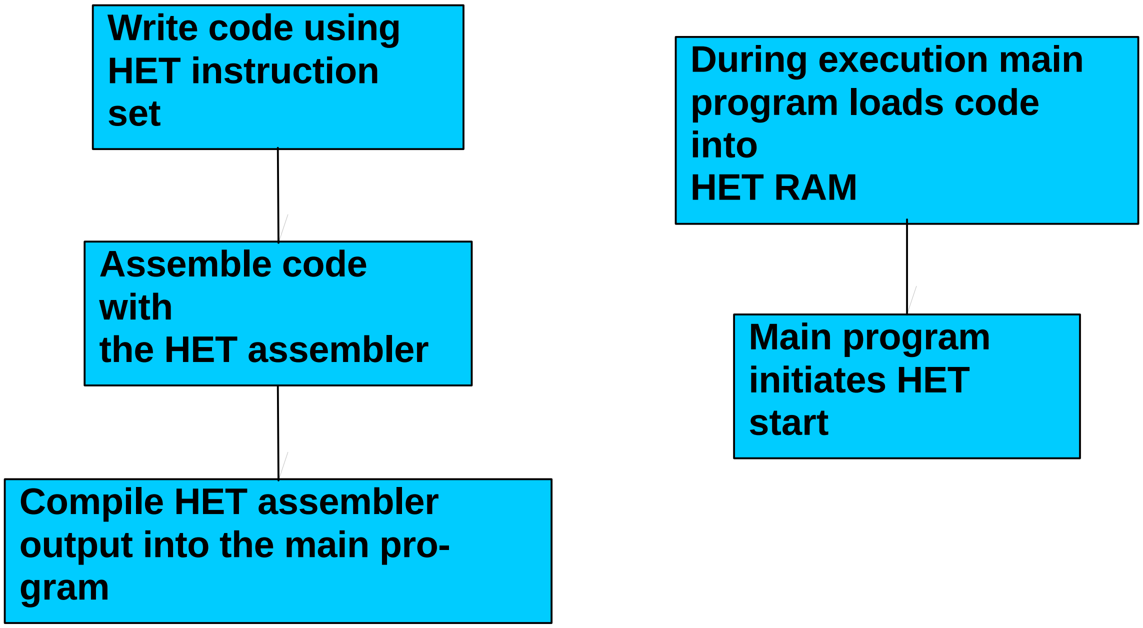

Write code using HET instruction set
Assemble code with the HET assembler
Compile HET assembler output into the main pro-gram
During execution main program loads code into HET RAM
Main program initiates HET start

Figure 1. HET Coding Sequence

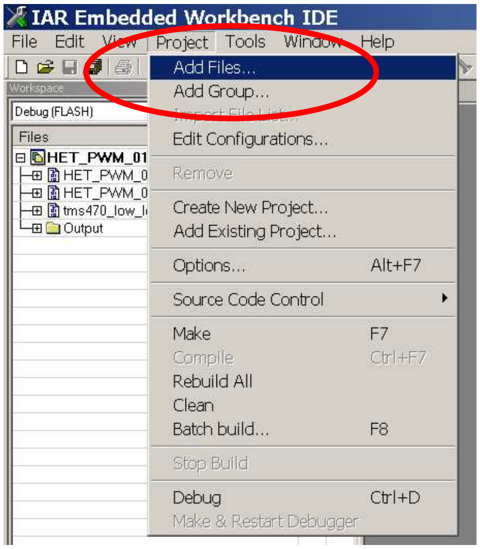
1.1 HET Assembler and IAR Tools Set Up

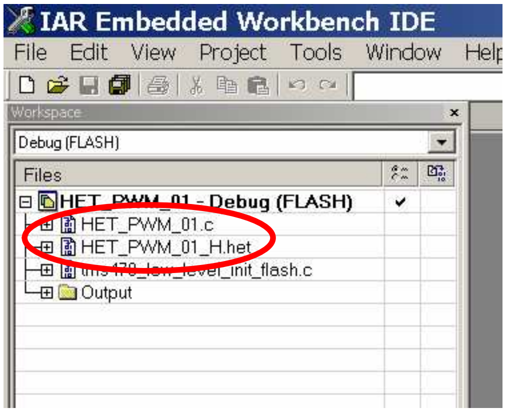
The HET Assembler assembles the HET code into a C program and a header file (xxx.c and xxx.h). There are two ways to assemble the HET program. The first is to use the custom build feature in the IAR tools to automatically assemble the code for you. It will recompile the HET code when it has been modified. The second method requires the programmer to use a batch file or command line to invoke the assembler to create the files manually. The following sections will explain both methods.

1.1.1 IAR Tools and the Custom Build Feature

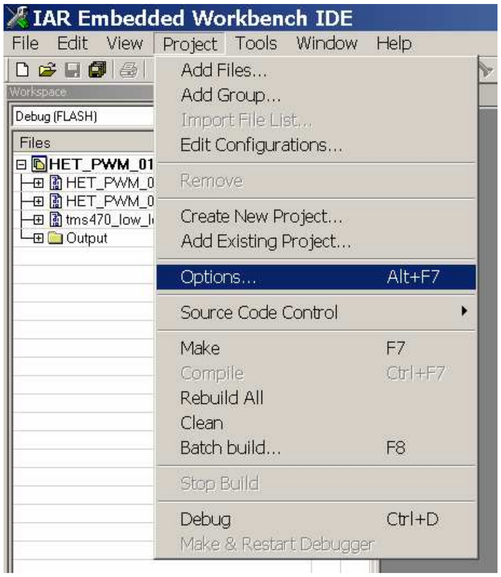
The Custom Build feature uses an alternate tool to generate files for your project. To use this feature:

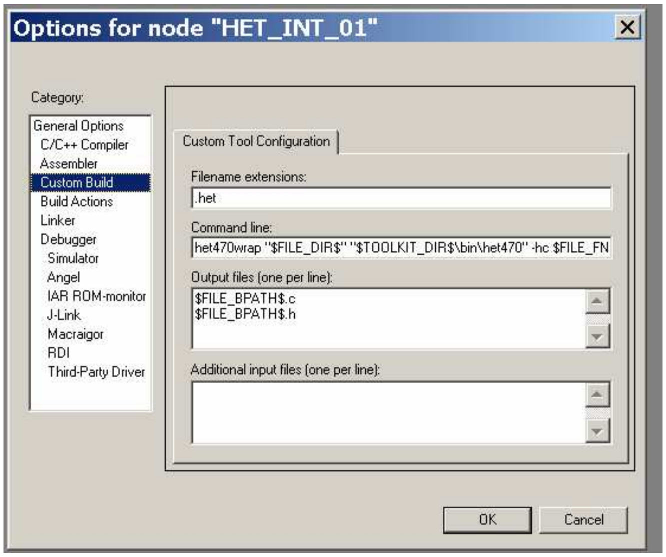
1. Add the HET program file to your project (HET_PWM_01_H.het), This assumes you also added your main program as well (in this case HET_PWM_01.c).




IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
Debug (FLASH)
Files
HET_PWM_01 - Debug (FLASH) ✓
HET_PWM_01.c
HET_PWM_01_H.het
tms470_low_level_init_flash.c
Output

2. Select this file in the workspace window and choose **Project>Options**. Select **Custom Build** from the list of categories.





Filename extensions: .het

Command line: het470wrap "$FILE_DIR$" "$TOOLKIT_DIR$\bin\het470" -hc $FILE_FN

Output files (one per line):
$FILE_BPATH$.c
$FILE_BPATH$.h

Additional input files (one per line):

3. In the Filename extensions field, type the **filename extension**. Remember to specify the leading period (.het).
4. In the Command line field, type the command line for executing the external tool, for example:

```
het470wrap "$FILE_DIR$" "$TOOLKIT_DIR$\bin\het470" -hc $FILE_FNAME$ $FILE_BNAME$
```

5. In the Output files field, describe the output files that are relevant for the build. In this example, the tool Flex would generate two files—one source file and one header file. The text in the Output files text box for these two files would look like this:

```
$FILE_BPATH$.c
$FILE_BPATH$.h
```

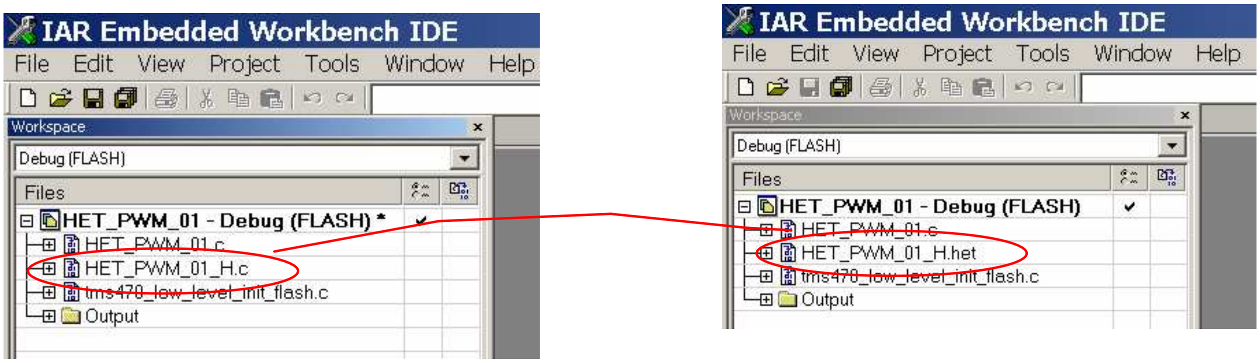
6. Click OK
7. To build your application, choose **Project>Make**.

1.1.2 HET Assembler Using the Command Line or Batch File

The HET assembler can be invoked directly using the command line or batch files. For the following batch file example, the HET program is called HET_PWM_01_H.het and the batch file is named HET_ PWM _01_H.bat. Place the HET_ PWM _01_H.bat and HET_ PWM _01_H.het in the same directory as the assembler. Run the HET_ PWM _01_H.bat file. It will generate the HET_ PWM _01_H.c and HET_ PWM _01_H.h files and place them in the same directory. At this point, you must add the HET_ PWM _01_H.c file to your program and then run the make function. Follow the add file and make example as previously described.

HET_PWM _01_H.bat example:

```
; HET_ PWM _01_H.bat
het470 -hc32 -n0 HET_ PWM _01_H.het .\ HET_ PWM _01_H
pause
```


IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
Debug (FLASH)
Files
HET_PWM_01 - Debug (FLASH) *
HET_PWM_01.c
HET_PWM_01_H.c
tms470_low_level_init_flash.c
Output
IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
Debug (FLASH)
Files
HET_PWM_01 - Debug (FLASH)
HET_PWM_01.c
HET_PWM_01_H.het
tms470_low_level_init_flash.c
Output

Batch Method VS Custom Build Method

Figure 2. Batch Method vs. Custom Build Method

1.1.3 Sample of HET Program and the Assembler Generated Files

1.1.3.1 Sample HET Program "HET_PWM_01_H.het"

```
; HET_PWM_01_H.het
L00: CNT  {next=L01, reg=A, irq=OFF, max=0x214A, data=0}
;  PWM (50% duty cycle)
L01: ECMP {next=L02, reg=A, hr_lr=LOW, en_pin_action=ON, pin=CC0, action=PULSEHI, irq=Off,
data=0x10A5}
;  PWM (25% duty cycle)
L02: ECMP {next=L03, reg=A, hr_lr=LOW, en_pin_action=ON, pin=CC2, action=PULSEHI, irq=Off,
data=0x852}
L03: BR      {next=L00, cond_addr=L00, event=NOCOND}
```

1.1.3.2 Sample Assembler Output Header File "HET_PWM_01_H.h"

```
#define HET_L00_0    (e_HETPROGRAM0_UN.Program0_ST.L00_0)
#define HET_L01_0    (e_HETPROGRAM0_UN.Program0_ST.L01_0)
#define HET_L02_0    (e_HETPROGRAM0_UN.Program0_ST.L02_0)
#define HET_L03_0    (e_HETPROGRAM0_UN.Program0_ST.L03_0)

typedef union
{
     HET_MEMORY    Memory0_PST[4];
    struct
    {
        CNT_INSTRUCTION L00_0;
        ECMP_INSTRUCTION L01_0;
        ECMP_INSTRUCTION L02_0;
        BR_INSTRUCTION L03_0;
    } Program0_ST;
} HETPROGRAM0_UN;
extern volatile HETPROGRAM0_UN e_HETPROGRAM0_UN;
extern const HET_MEMORY HET_INIT0_PST[4];
```

1.1.3.3 Sample Assembler Output C File "HET_PWM_01_H.c"

```
#include "std_het.h"

HET_MEMORY const HET_INIT0_PST[4] =

{
     /* L00_0 */
    {
        0x00001600,
        0x0000214A,
        0x00000000,
        0x00000000

},
     /* L01_0 */
    {
        0x00002080,
        0x00102018,
        0x000214A0,
        0x00000000

},
     /* L02_0 */
    {
        0x00003080,
        0x00103118,
        0x00010A40,
        0x00000000

},
     /* L03_0 */
    {
        0x00000D00,
        0x00000000,
        0x00000000,
        0x00000000
    }
};
```

1.2 HET Setup In the Main Program

This section describes the setup in the main program for the HET. The main program copies the HET code to HET RAM, initializes the HET registers, and enables the HET.

1.2.1 Copying the HET Code to HET RAM

The HET program xxx.het is assembled into a xxx.c file, which is added to the project with the main program. The IAR make function compiles the HET C file with the main program. A routine within the main program must load the HET code into HET RAM. The main program contains the address of the HET RAM for the copy program to use.

```
__no_init volatile HETPROGRAM0_UN e_HETPROGRAM0_UN @ 0x00800000;

void MemCopy32(unsigned long *dst, unsigned long *src, int bytes)
{
  for (int I = 0; I < (bytes + 3) / 4; I++)
    *dst++ = *src++;
}
```

1.2.2 Set Up Of the HET Registers

The set-up for the HET includes configuring the HET registers. These registers control HET functions and set the Loop Resolution (LR) and High Resolution (HR) prescale values. The following is an example of the HET setup.

- The peripheral enable must be set for the HET to function.
- The HETGCR is typically set to master mode (in the case of multiple HETs a choice between master and slave is available), ignore suspend disables the software break points, and the HET enable bit start the HET.
- The HETPFR contains the fields for the HR and LR prescale values.
- The HETDIR and HETDOUT control the GIO and data on the HET pins.

1.2.2.1 HET Setup in the Main Program Example

```
PCR = CLKDIV_1;                                                      // ICLK = SYSCLK/4
PCR |= PENABLE;                                                      // enable peripherals

HETGCR = CLK_MASTER + IGNORE_SUSPEND;  //  HET Master Mode, Ignore SW BP

// copy HET instructions to HET ram

MemCopy32((void *) &e_HETPROGRAM0_UN, (void *) HET_INIT0_PST, sizeof(HET_INIT0_PST));

HETPFR = 0x0000052b;                                             // Set PFR register

HETDIR  = 0xFFFFFFFF;                                            // Set all HET as GIO outputs
HETDOUT = 0xFFFFFFFF;                                         // Flash all leds off and on and off

HETGCR |= ON;                                                           // Start HET
```

2 Code for Actual Use

Refer to the TMS470 website for code examples. http://www.ti.com/tms470

3 References

1. TMS470R1x High-End Timer (HET) Reference Guide (SPNU199)

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 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.

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.

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

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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