TMS470R1x High-End Timer (HET) Reference Guide

Literature Number: SPNU199D
July 2004

Texas Instruments

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 application s 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 form 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

REVISION HISTORY

REVISION	DATE	NOTES
D	7/04	Pages 58, 60: added information about delaying before putting the HET in power-down mode to bit 0 information. Pages 87–153: indicated that bit 21 of the program field of each command is undetermined. Page 83: “U” added to list of abbreviations.
C	4/03	Page 12–13: added section on 64-bit CPU access to Timer RAM.
B	9/02	Converted to a stand-alone book
A	11/00	Pre-dates revision history
*	2/00	Initial version

Contents

Figures

Tables

High-End Timer (HET)

This reference guide provides a general description of the high-end timer (HET). The HET is a software-controlled timer with a dedicated specialized timer micromachine and a set of 21 instructions. The HET micromachine is connected to a port of input/output (I/O) pins.

Topic **Page**

1 Features

The TMS470R1x device family has the following HET features:

- ❏ High-level timer functions such as period and pulse width measurement
- ❏ 32 input/output (I/O) channels for timer functions such as capture, compare, pulse width measurements (PWMs), and general purpose I/O pins
- ❏ 24 HR (HR) hardware channels associated with 24 of the 32 I/O channels
- ❏ User-programmable loop resolution and HR clocks
- ❏ User-programmable micromachine with a reduced instruction set (21 instructions) for build-time functions
- ❏ Multiple 20-bit virtual counters for timers, event counters, and angle counters
- ❏ Dual port RAM with capacity for 64 words of 96 bits (expandable to 256 words, see device-specific datasheet) to optimize the number of cycles per instruction (most instructions require one cycle)
- ❏ Shadow registers for CPU communication
- ❏ Minimal CPU code overhead and required interrupts
- ❏ 35 interrupt sources with two individually programmable levels
- ❏ HR I/Os and coarse resolutions implemented by sub loops for multiple resolution capability
- ❏ Conditional program execution based on pin conditions and compares
- ❏ Software breakpoint capability on each instruction
- ❏ Synchronously operates with the system clock
- ❏ Possibility of using multiple HETs synchronized on the same resolution clock

2 Overview

The HET is a third-generation Texas Instruments (TI) advanced intelligent timer module. The HET module described in this reference guide benefits from experience gained from the TMS370 device family HET design (see section 2.4 on page 5).

This timer module provides sophisticated timing functions for real-time applications such as car engine management. The new HR hardware channels allow greater accuracy for widely used timing functions such as period and pulse measurements, output compare, and PWMs.

The reduced instruction set, based mostly on very generic and comprehensive instructions, has improved the definition and development cycle time of an application and its derivatives. The new HET breakpoint feature, combined with various stop capabilities, makes the TMS470 HET software application easy to debug.

2.1 Timer Module Structure and Execution

The timer consists of a specialized micromachine that operates a reduced instruction set at the same speed as the system clock. Most of the data and the arithmetic and logical unit (ALU) are 25 bits wide. Two 20-bit registers and one 25-bit registers are available to manipulate information such as time, event counts, and angle values. System performance is improved by a wide instruction format (96 bits) that allows the CPU to fetch the instructional operation code and data in one system cycle, thus increasing the speed at which data can be processed. The typical operations performed in the ALU are additions (count), compares, and magnitude compares (higher or same).

Each instruction includes an 8-bit field for specifying the address of the next instruction to be executed. This means that an application program sequence is not controlled by a program counter (PC), but by the actual content of each instruction. This offers greater flexibility in going back and forth in the memory during program execution.

The interface to the host CPU is based on both communication memory and control registers. The communication memory includes timer instructions (program and data). This memory is typically initialized by the CPU after reset before the timer starts execution. Once the timer program is loaded into the memory, the CPU starts the timer execution, and only data parameters can then be read or written into the timer memory. The control registers include bits for selecting timer clock, configuring I/O pins, and controlling the timer module.

The programmer implements timer functions by combining instructions in specific sequences. For instance, a single Count instruction sequence

implements a timer. A PWM would need a two-instruction sequence: Count and Compare. A complex time function may include many instructions in the sequence. The total timer program is a set of time functions executed sequentially, one after the other. Reaching the end, the program must roll to the first function so that it behaves as a loop.

The time for a loop to execute is referred to as a *loop resolution clock cycle*. When the HET rolls over to the first function (i.e., instruction), the timer waits for the resolution clock to restart the execution of the loop to ensure that only one loop is executed for each loop resolution clock.

The execution time of this main loop is the sum of all the cycles required for the instructions that the loop executes. The main loop must be completed within the loop resolution clock. Therefore, you must make sure that the timer functions implemented in the algorithm complete within the resolution clock. Otherwise, the program will execute unpredictably because some instructions will not be executed each time through the loop.

This effect creates a strong link between the accuracy of the timer functions and the number of functions (the number of instructions) the timer can perform. To overcome this limitation, some of the most commonly used instructions can access the HR hardware. This access allows pulse or period measurements, time compare, and PWM output waveforms at the resolution of the HR clock instead of the loop resolution clock.

Updating parameters of compare instructions (time, event, or angle compare values) can be controlled by the timer itself using built-in move instructions. These built-in move instructions actually transfer new data from the CPU into the active compare field of an instruction synchronously with the resolution clock. This synchronization method makes it unnecessary to use interrupts to avoid such problems as incorrect pulse widths.

2.2 Major Advantages

In addition to classic time functions such as input capture or multiple hand-free PWMs, higher-level time functions can be easily implemented in the timer program main loop. Higher-level time functions include angle driven wave forms, angle and time-driven pulses, and input pulse width modulation (PWM) duty cycle measurement.

Because of these high-level functions, data exchanges with the CPU are limited to the fundamental parameters of the application (periods, pulse widths, angle values, etc.); and the real-time constraints for parameter communication are dramatically minimized; for example, few interrupts are required and asynchronous parameter updates are allowed.

The reduced instruction set and simple execution flow control make it simple and easy to develop and modify programs. Simple algorithms can embed all the flow control inside the HET program itself. More complex algorithms can take advantage of the CPU access to the HET RAM. With this, the CPU program can make calculations and can modify the timer program flow by changing the data and control fields of the HET RAM. CPU access to the HET RAM also improves the debug and development of timer programs. The CPU program can stop the HET and view the contents of the program, control, and data fields that reside in the HET RAM.

Finally, the modular structure provides maximum flexibility to address a wide range of applications. The timer resolution can be selected from two cascaded prescalers to adjust the loop resolution and HR clocks. The 32 I/O pins can provide any combination of input, period or pulse capture, and output compare, including 24 HR channels. The standard memory structure allows module configuration from 64 words to 256 words of timer program memory.

2.3 Performance

Most instructions execute in one cycle, but a few take two or three cycles. The average cycle per instruction (CPI) measured on various complex benchmarks is approximately 1.3.

The HET can generate many complex output waveforms without CPU interrupts. Where special algorithms are needed following a specific event (e.g., missing teeth or a short/long input signal), a minimal number of interrupts to the CPU are needed. The minimal interrupts frees the CPU bandwidth to perform other tasks.

2.4 TMS470 HET Compared to TMS370 HET

The TMS470 HET has a similar architecture to the TMS370 HET. However, the following improvements have been made:

- ❏ The instruction field and ALU width have been widened to better meet larger timer requirements.
- ❏ The cycle per instruction (CPI) time has been enhanced to increase overall performance.
- ❏ HR hardware channels have been added to meet the requirement of high accuracy (100ns and below).

The average CPI of 1.3 is a 50% improvement over the 370 HET generation. Using the state-of-the-art TI CMOS ASIC library allows higher operation frequency, which, combined with the optimized CPI, results in an overall performance improvement of 2.5x over the previous generation.

2.5 Instructions Features

The TMS470 HET has the following instructions features:

- ❏ All standard resolution outputs are synchronized to the loop resolution clock to avoid the variability associated with the instruction placement in the HET program.
- ❏ HET uses a RISC-based specialized timer micromachine to carry out a set of 21 instructions.
- ❏ Instructions implemented in a VLIW format (96 bits wide).
- ❏ The HET program execution is self-driven by external or internal events, branching to special routines based on input edges or output compares.

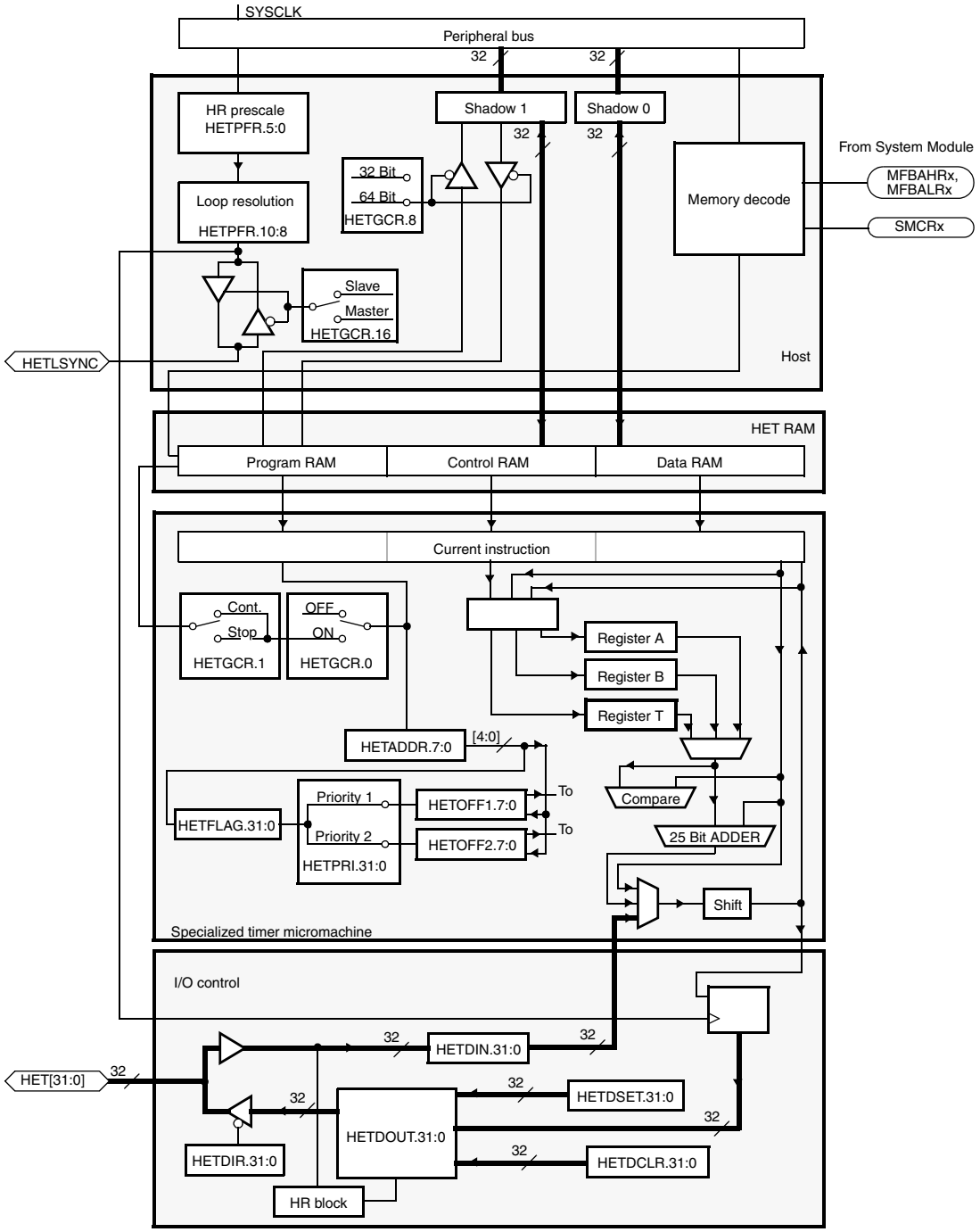
3 Block Diagram

The HET module (see Figure 1) comprises four separate components:

1) Host interface
2) HET RAM
3) Specialized timer micromachine
4) I/O control

The HET is attached to an I/O port of up to 32 pins. Please see the device-specific data sheet for details on the number of HET pins available.

Figure 1. HET Block Diagram



4 HET Functional Description

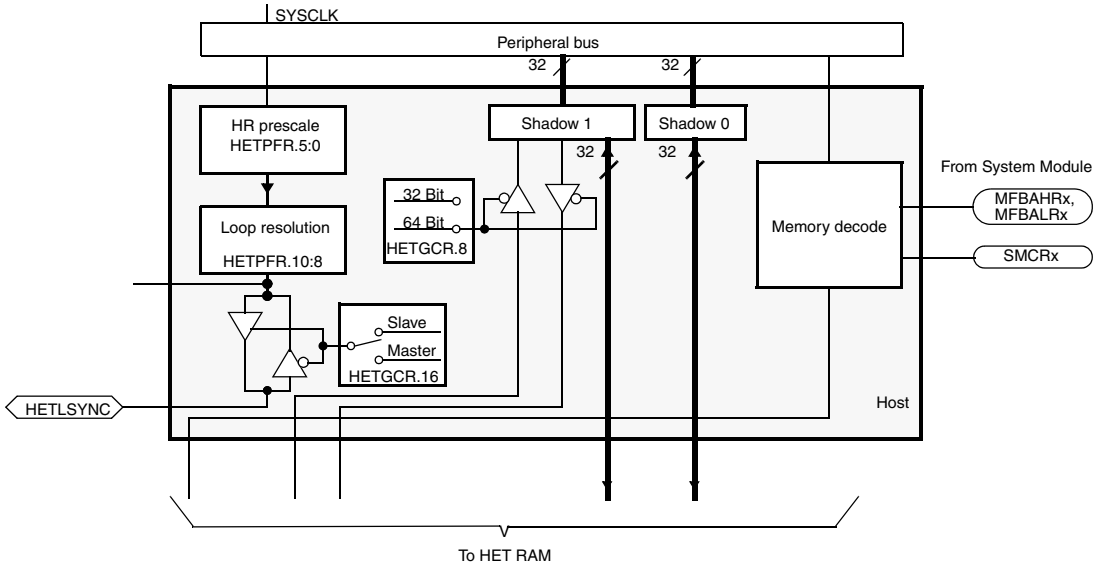
The HET contains RAM into which HET code is loaded. The HET code is run by the specialized timer micromachine. The host interface and I/O control provide an interface to the CPU and external pins respectively.

4.1 Host Interface

The host interface controls all communications between timer-ram and CPU (see Figure 2). It includes following components:

- ❏ An interface from the memory controller in the system module (SMCRx)
- ❏ Two 32-bit shadow registers (shadow register 0 and 1)
- ❏ A bit (HETGCR.8) for controlling how the shadow registers are written/read by the CPU
- ❏ Two user programmable 6-bit(HETPFR.5:0) and 3-bit (HETPFR.10:8) clock pre scalers for HR and loop resolution clocks
- ❏ A bit (HETGCR.16) for controlling whether the HET is configured as a master or a slave
- ❏ An interface from the address manager in the system module (MFBAHRx and MFBALRx)

Figure 2. Host Interface

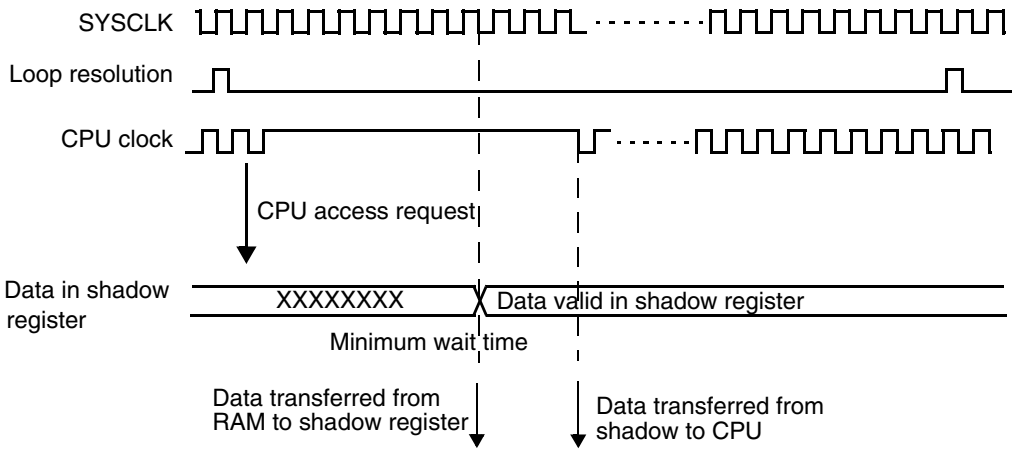


4.1.1 Static Memory Control

The CPU accesses to HET RAM are programmed in the system module through the memory controller. The SMCRx controls the system module in transactions between the system module and the HET.

- ❏ During run-time, the minimum number of wait cycles should be set to seven for the HET RAM in the MMC.
- ❏ During initialization of the HET RAM (loading of the memory before HET is ON), only two wait cycles are needed to access HET RAM to minimize the time spend in the initialization sequence. See Figure 3.

Figure 3. HET RAM Accesses (Example of a 10-Wait Cycle Read Operation)



Each time the CPU performs an access to the HET RAM, one or two cycles of the HET program are dedicated to the CPU access since the CPU and the HET cannot access the HET RAM at the same time. A 32-bit read takes one cycle and a 64-bit takes two cycles. One of the most common uses of a 64-bit read is for the PCNT instruction. The 64-bit read gives you the control field (previous period) and data field (current period) at the same instant in time. The total number of available time slots for the HET must be computed by subtracting the time slots consumed by the CPU from the number of cycles available in a loop resolution. In the worst case of continual CPU access to the HET and seven wait cycles programmed in the system module, the HET loses one time slot (or two cycles in case of auto read/clear of PCNT) out of every eight.

Example:

An example HET configuration allows 24 time slots in a loop resolution (by the prescalers). The Memory controller is programmed with seven wait-state to

the HET. In the case of continuous CPU access, three cycles per resolution will be dedicated to the CPU, leaving 21 time slots really available for the HET application program.

4.1.2 Shadow Registers

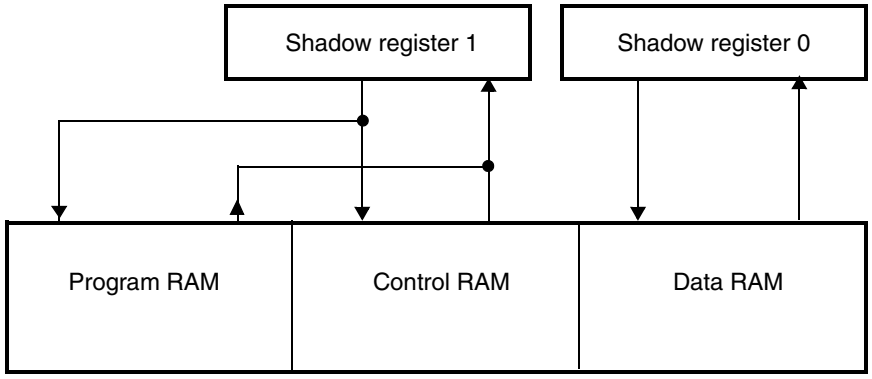
The two shadow registers allow the CPU to access the program, control, or data field in the timer-RAM without interrupting execution of a timer instruction. Shadow register operations are transparent to you. See Figure 4.

Shadow registers allow the timer instructions to access the timer control field and data field 64 bits at a time. To be active, this 64-bit mode must be set in the global configuration register.

The timer-RAM cannot be accessed by byte or half-word (16-bit data).

- ❏ Shadow register 0 is assigned to data field access.
- ❏ Shadow register 1 shares program field and control field accesses.

Figure 4. Shadow Registers with RAM Units



4.1.3 CPU Access to Timer-RAM

- ❏ 32-bit access (HETGCR.16 = 0) (program, control and data field)
  - ■ **Write**: When the CPU executes STR, it first writes 32-bit CPU data into relevant shadow registers. The system module then asserts N-cycles wait time to the CPU (depending on the value in SMCRx). The CPU then resumes its execution. At the end of the current timer instruction, the shadowed data moves into timer selected RAM.

- **Read**: When the CPU executes LDR, the system module first asserts N cycles wait time to the CPU (depending on the value in SMCRx). At the end of the current instruction, the 32-bit timer data are loaded into relevant shadow registers. At the end of the wait time, the shadowed data moves onto the data bus.

❑ 64-bit access (HETGCR.16 = 1)

- **Write**: For a 64-bit write, two STR CPU instructions are required. The control field must be written first, followed by the data field. For the first STR instruction, the CPU writes 32-bit CPU data into shadow register 1. The system module then asserts N cycles wait time to the CPU (depending on the value in SMCRx). For the second STR instruction, the CPU writes 32-bit CPU data into shadow register 0. The timer then asserts N cycles wait time to the CPU. At the end of the current timer instruction, all 64-bit shadowed data moves into timer RAM.
- **Read**: For a 64-bit read, two LDR CPU instructions are executed. The control field must be read first, followed by the data field. For the first LDR instruction, the system module first asserts N cycles wait time to the CPU (depending on the value in SMCRx). At the end of the current timer instruction, the timer loads 32-bit control field into shadow register 1 and the 32-bit data field into shadow register 0 (both in the same cycle). At the end of the wait time, the shadowed data from the control field move onto the data bus. For the second LDR instruction, the timer first asserts N cycles wait time to the CPU (depending on the value in SMCRx). The CPU then moves the 32-bit of shadow register 0 (loaded from the data field at the first LDR) onto the data bus.

Note:

- Interrupts

During the time the 64-bit access bit is set, it is recommended to disable all interrupts.

Background**:**

- Avoid an interrupt routine that causes a delay between the first and the second LDR instruction.
- Disabling the interrupts is a must if an interrupt routine also reads the HET RAM locations.

- DMA (Direct Memory Access) Module:

A situation where the TMS470 CPU is accessing the HET RAM in 64-bit access mode at the same time the DMA is also trying to access the HET RAM can lead to incorrect values for the DMA access and/or the 64-bit CPU access. This situation can be avoided by any of the following methods:

a)The application ensures that DMA accesses to the HET RAM and the ARM7 CPU 64-bit-accesses to the HET RAM do not occur at the same time.

b)The ARM7 CPU temporarily stops the DMA (by setting the DMA_STOP or DMA_HALT bit) prior to doing an ARM7 CPU 64-bit access of the HET RAM. Note that this approach temporarily disables all DMA channels.

The ARM7 CPU temporarily disables only the DMA channel x, which is accessing the HET RAM before performing an ARM7 64-bit access of the HET RAM. This can be done by temporarily clearing the DMENx bit in the DMACCPRn register (for DMA channel x). For TMS470 devices containing a C54x DSP, contradicting modifications of the DMENx bit of the ARM7 and the DSP could be avoided using semaphores in API RAM.

- Automatic read/clear of the HET RAM data field

  The HET provides a feature allowing to automatically clear the data field after a 64-bit read operation of the control and data fields. This feature is implemented via the control bit, which is located in the control field (bit C21). This is a static bit that can be used by any instruction.

  a) Set the control bit in the selected instruction.

  b) Set the 64-bit access bit in the global configuration register.

  c) Perform a 64-bit read access with the CPU.

  d) Loads shadow registers 1 and 0 with the control and data fields of the memory, and then clears the data field [D24:D0] of the instruction. This sequence requires two time slots.

  This feature is mainly used for PCNT instruction. You read a period or pulse value, which is then cleared by the HET itself. This means that you do not need to take a new value as long as the period or pulse data is still cleared.

4.1.4 Memory Selects

The address of the HET RAM in the device memory map is programmable through the MFBAHRx and MFBALRx registers in the system module.The length of the HET RAM and the specific registers that control the position of the RAM are device specific. See the device-specific data sheet for further details.

4.1.5 Emulation Mode

Emulation mode, used by the software debugger, is specified in the global configuration register. When the debugger hits a breakpoint, the CPU sends a suspend signal to the modules. Two modes of operation are provided: suspend and ignore suspend.

- Suspend

  When a suspend is issued, the timer operation stops at the end of the current timer instruction. However, the CPU accesses to the timer RAM or control registers are freely executed.

- Ignore suspend

  The timer RAM ignores the suspend signal and operates real time as normal. Wait states asserted by the timer while the suspend signal is active are executed as normal.

4.1.6 Power-Down

The HET can be put into either local or global low power mode. Global low power mode is asserted by the system (see the system module reference guide; literature number SPNU189) and is not controlled by the HET. During global low power mode, all clocks to the HET are turned off so the module is completely inactive. See Table 1.

Local low power mode is asserted by setting the POWERDOWN (HETGCR [24]) bit; setting this bit stops the clocks to the HET internal logic (state machine), but the HET registers continue to be clocked. In addition, setting the PPWNOVR bit found in the CLKCTRL register of the TMS470 system module disables the clocks to all modules in which the local power-down bit is set. When all the clocks of the HET are powered down, no access from the CPU could be made to the HET registers.

To exit the local power mode, the PPWNOVR bit must then be reset to enable the clocks of the HET register, and the POWERDOWN (HETGCR [24]) bit must be reset to 0 to enable the clocks of the HET internal logic (state machine).

Table 1. Power-Down Mode Summary

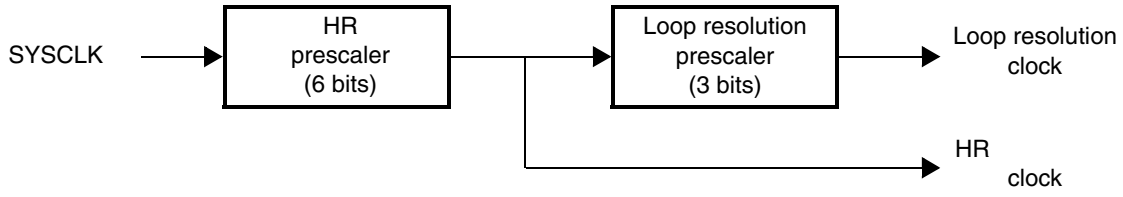
POWERDOWN HETGCR [24]	PPWNOVR CLKCTL	• HET internal clocks •
0	0	All clocks enable
0	1	All clocks enable
1	0	HET register clock enable HET state machine clock disable
1	1	All clocks disable

4.2 Time Base

Two prescalers generate the timer resolution clock for the program loop, and the HR clock for the HR I/O counters. See Figure 5. The prescalers consist of the following:

- A 6-bit prescaler dividing the system clock by a user-defined HR prescale divide rate (hr) stored in the 6-bit HR prescale factor code (with a linear increment of codes). See Table 2.
- A 3-bit prescaler dividing the HR clock by a user-defined loop-resolution prescale divide rate (lr) stored in the 3-bit loop-resolution prescale factor code (with a power of 2 increment of codes). See Table 3.

Figure 5. Prescaler Configuration



The loop resolution (lr) is typically determined by the number of cycles that the HET requires to complete the worst-case application software loop plus CPU accesses. Therefore, you must choose the prescaler numbers so that the number of time slots in any software loop is greater than the number of cycles required in a loop. Typically, you choose the smallest HR prescaler number that will still allow the appropriate number of cycles.

- Time Slots Available (Ts)

  Ts = [HR prescale divide rate (Hr) * Loop Resolution prescale divide rate (Lr)] -1

- HR clock period (HRP)

  *HRP = Hr * SYSCLK period* (Tclk) = Hr / Sysclk

- Loop Resolution clock period (LRP)

  LRP = Loop Resolution Prescale divide rate (LR) * HR clock period (HRP) = Lr * Hr *Tclk = (Hr * Lr)/ Sysclk

Table 2. HR Prescale Factor Codes

HR Prescale Factor Code						HR Prescale Divide Rate
5	**4**	**3**	**2**	**1**	**0**	
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
:	:	:	:	:	:	:
:	:	:	:	:	:	:
1	1	1	1	0	1	62
1	1	1	1	1	0	63
1	1	1	1	1	1	64

Table 3. Loop Resolution Prescale Factor Codes

Loop Resolution Prescale Factor Code				Loop-Resolution Prescale Divide Rate
ordinal	**2**	**1**	**0**	
0	0	0	0	1
1	0	0	1	2
2	0	1	0	4
3	0	1	1	8
4	1	0	0	16
5	1	0	1	32

Loop Resolution Prescale Factor Code				Loop-Resolution Prescale Divide Rate
ordinal	2	1	0	
6	1	1	0	reserved
7	1	1	1	reserved

Note:
When the loop-resolution prescale divide rate is smaller than 32, the non-relevant bits of the HR data fields (non-relevant LSBs) will be one of the following; see also Table 4:

- Written as 0 for HR capture
- Or interpreted as 0 for HR compare

For more information about this, see the descriptions of the ECMP, MCMP, PCNT, PWCNT, and WCAP instructions.

Table 4. Interpretation of the 5-Bit HR Data Field

Loop Resolution Prescale divide rate (Lr)	• Bits of the HR data field					HRP Cycles delay range
	•B[4]	•B[3]	•B[2]	•B[1]	•B[0]	
1	X	X	X	X	X	0
2	**V**	X	X	X	X	0 to 1
4	**V**	**V**	X	X	X	0 to 3
8	**V**	**V**	**V**	X	X	0 to 7
16	**V**	**V**	**V**	**V**	X	0 to 15
32	**V**	**V**	**V**	**V**	**V**	0 to 31
Weight factor as fraction of HR Cycles in one loop	1/2	1/4	1/8	1/16	1/32	

Note: V = Valid bit; X = Non-relevant bit (ignored)

Example:

HETPFR[31:0] register = 0x300 → Lr prescale divide rate = Lr = 8 (→ 8 Hr cycles in one loop).

Assumption: HR data field = 0x14 = 10100b

Lr = 8 → Bits B[1] and B[0] are ignored → Hr delay = 101b = 5 Hr Cycles

or by using the calculation with weight factors:

Hr Delay = Lr * (B[4] * 1/2 + B[3] * 1/4 + B[2] * 1/8 + B[1] * 1/16 + B[0] * 1/32)
= 8 * (1 * 1/2 + 0 * 1/4 + 1 * 1/8)
= 5 Hr cycles

4.2.1 Determining Loop Resolution

As an example, consider an application that requires HR I/Os with a resolution of 62.5 ns, and 8 standard I/Os at 2 μs, and needs 60 time slots for HET application programs.

Considering a system frequency of 32 MHz, the following divide-by rates can be programmed using the following prescaler factor codes:

- Divide-by rate of 2 for the HR clock, giving 62.5 ns (code = 000001 = 0x1):

  HRP = Hr / Sysclk = 2 /32 Mhz = 62.5 ns

- Divide-by rate of 32 for the loop resolution clock, giving 2 μs (code = 101 = 0x5)

  LRP = Lr * HRP = 32 * 62.5 ns = 2 us

- Number of available time slots: Ts = (Hr * Lr) -1 = (2 * 32)-1 = 63

- Hr =2, Lr = 32. Then, the corresponding code inside the HETPFR[31:0] register will be: 0x00000501

If in the example above, the resolution needed moves from 2 μs to 1 μs, then only 31 time slots will be available.

Or, if the number of time slots needed are 90, HRP then moves to 93.75ns.

4.2.2 Multi-HETs Resolution Synchronization

In some applications configured with multi HETs, the resolutions must be synchronized. This is typically the case for configurations that use a unique angle reference shared by several HETs. See Figure 6.

The HET provides a synchronization mechanism for multiple timers. The Clk_master/slave (HETGCR.16) configures the HET in master or slave mode (default is slave mode). A HET in master mode provides a signal to synchronize the prescalers of the slave HET. The slave HET synchronizes its loop resolution to the loop resolution signal sent by the master. The slave does not require this signal after it receives the first synchronization signal. However, anytime the slave receives the re-synchronization signal from the master, the slave must re-synchronize.

Stand-alone 470HET:

a) Configure the HR and LR prescaler control registers and all other control registers (direction, interrupts, etc.)

b) Turn on the 470HET and configure it as master. This can be done in one instruction since both control bits are in the global control register.

For multiple HETs:

a) Select one HET as master by writing in the clk_master/slave bit.

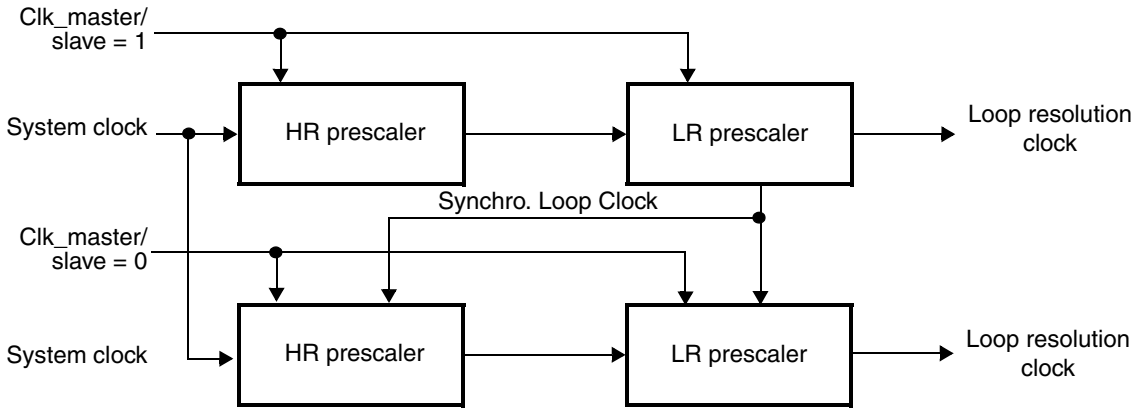
b) Configure the HR and LR prescaler control registers and all other control registers (direction, interrupts, etc.) of the master HET.

c) Configure the HR and LR prescaler control registers and all other control registers (direction, interrupts, etc.) of the slave HET. The loop resolution of the master **must be the same** as the slave's loop resolution.

d) Turn on all slave HETs. They will wait for the synchronization from the master HET to start the program execution.

e) Turn on the master HET.

Figure 6. Multi-HETs Configuration - Synchronization of Resolutions



Note: The Sync

The loop clock signal is two SYSCLK cycles ahead of the resolution clock.

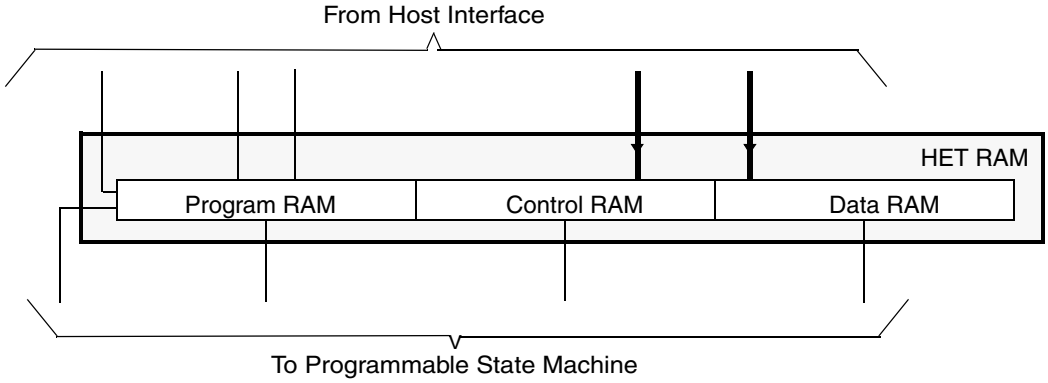
4.3 HET RAM

The timer RAM uses dual port access. This means that one RAM address may be written while another address is read. This is especially useful for the HET architecture since the prefetch is done during the last cycle of each instruction and this RAM gives the capability of writing back data into the current instruction while reading the next instruction. The RAM words are 96-bits wide, which are split into three 32-bit fields (program, control, and data) to fit with the CPU data bus. See Figure 7

Note: RAM

The RAM supports word accesses only.

Figure 7. HET RAM



4.3.1 Memory Map

Table 5 describes the HET memory map.

Table 5. HET Memory Map

Instruction Address	Program Field Address	Control Field Address	Data Field Address	Reserved Address
00h	XX000h	XX004h	XX008h	XX00Ch
01h	XX010h	XX014h	XX018h	XX01Ch
02h	XX020h	XX024h	XX028h	XX02Ch
:	:	:	:	:
:	:	:	:	:
3Fh	XX3F0h	XX3F4h	XX3F8h	XX3FCh
40h	XX400h	XX404h	XX408h	XX40Ch
:	:	:	:	:
FFh	XXFF0h	XXFF4h	XXFF8h	XXFFCh

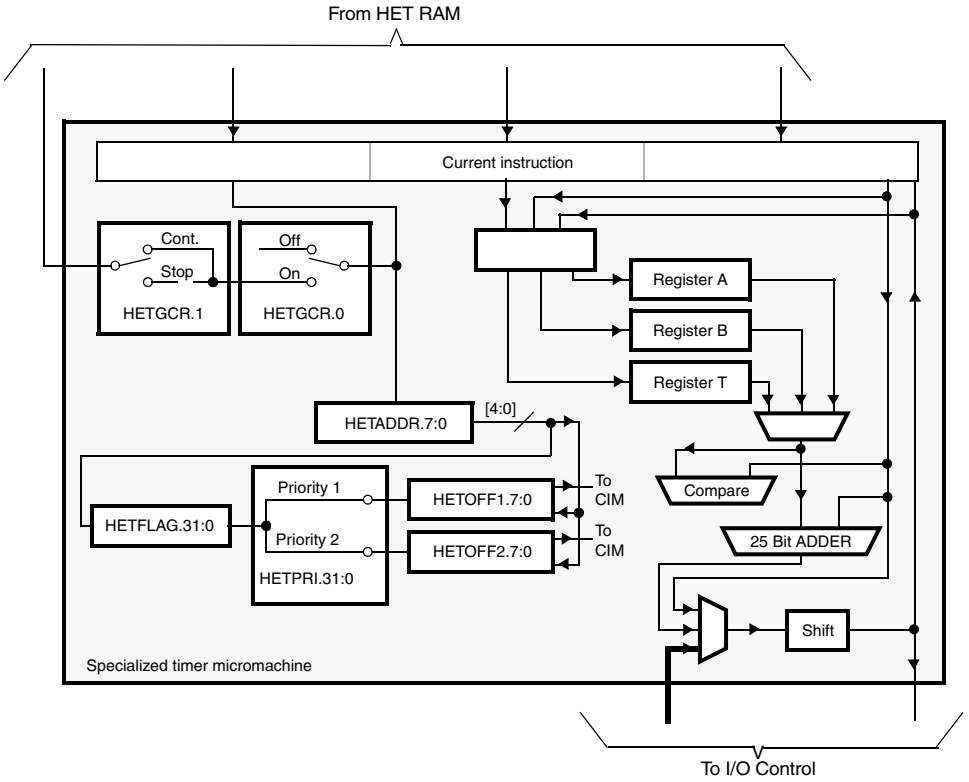
4.4 Specialized Timer Micromachine

The HET has its own instruction set, detailed in Section 6.7. The timer micromachine reads each instruction from the HET RAM. The program and control fields contain the instructions for how the specialized timer micromachine executes the command. For most instructions, the data field stores the information that needs to be manipulated.

The specialized timer micromachine executes the instructions stored in the HET RAM sequentially. The HET program execution is self-driven by external or internal events. This means that input edges or output compares may force the program to branch to special routines using an indexed addressing mode.

Figure 8 shows some of the major operations that the HET can carry out, namely compares, captures, angle functions, additions, and shifts. The HET contains three registers (A, B, and T) used to hold compare or counter values and used by the HET instructions. Data may be taken from the registers or the data field for manipulation; likewise, the data may be returned to the registers or the data field.

Figure 8. Specialized Timer Micromachine



4.4.1 Time Slots and Resolution Loop

Each instruction requires a specific number of cycles or time slots to execute. The resolution specified in the prescaler registers determines the timer accuracy. All input captures, event counts, and output compares are executed once in each resolution loop. HR captures and compares are possible (up to system clock accuracy) on the HR I/O pins. For more information, see section 4.5 on page 28.

4.4.2 Program Loop Time

The program loop time is the sum of all cycles used for instruction execution and CPU accesses. This time may vary from one loop to another depending on the number of CPU accesses or special routine execution.

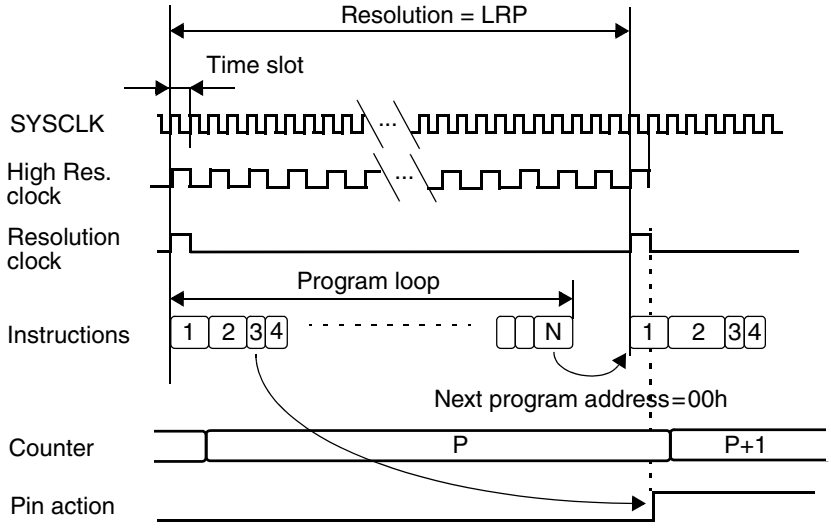
The timer program restarts on every resolution loop. The start address is fixed at RAM **address 00h**. The longest loop in a program must fit within one resolution loop time to guarantee complete accuracy.

The last instruction of a program points to the start address, (next program address= 00h).

If a program loop is shorter than the selected resolution, the program waits until end of resolution before starting the next loop.

The timing diagram (see Figure 9) illustrates the program flow execution.

Figure 9. Program Flow Timings

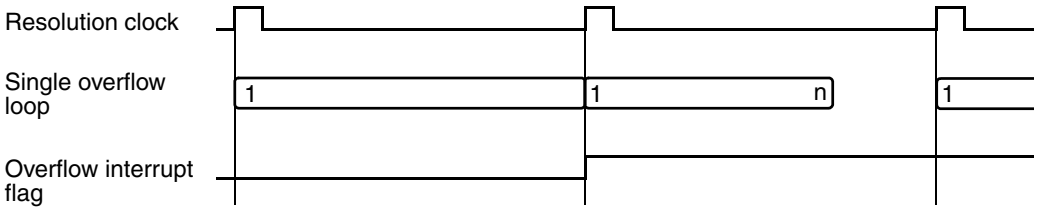


4.4.3 Program Overflow

If the number of time slots used in a program loop exceeds the number available in one resolution, the timer sets the program overflow interrupt flag located in the exceptions interrupt control register. To maintain synchronization of the I/Os, this condition should never occur in a normal operation. During the software debug phase, this flag may be used to debug the programmed loop resolution value. See Figure 10.

In case of program overflow, the HET sequence is started again at address 00h.

Figure 10. Use of the Overflow Interrupt Flag



Note: Limitation on instruction falling on a program overflow

The pin action will not be taken if the instruction that caused the pin action falls on a program overflow. All other instructions such as updating the A, B, and T registers and the RAM will still be performed.

4.4.4 Instruction Execution Sequence

The execution of a HET program begins with the first occurrence of the loop resolution clock, after the HET is turned on. At the first and subsequent occurrences of the loop resolution, the instruction at location address 00h is prefetched. The program execution begins at the occurrence of the loop resolution clock and continues executing the instructions until the program branches to 00h location. The instruction is prefetched at location 00h and execution flag is reset. The HET goes into a wait state until the occurrence of the loop resolution clock and resumes normal execution. If a IDLE state is asserted (i.e., CPU access) at the beginning of the program, (at the occurrence of the loop resolution clock), the execution of the program is postponed by one or two system clock cycles.

Single and multi-cycle instructions prefetch the next instruction in the last cycle of the instruction. SCNT, the only three cycles instruction, fetches the next instruction always in the 3rd cycle, which is the last cycle of the instruction. ECMP, as other single cycle instructions, fetches the next instruction while executing the current instruction. By prefetching the instruction at the next or conditional address, the instruction is latched by the instruction registers and decoded in the same cycle.

The use of a dual-ported RAM permits dual access to the instruction memory. A single read and a single write can be performed in the system cycle. In fact, a read and write to the same location can also be performed in the same system cycle. The write operation is performed, regardless of the prefetching of instruction, in the cycle required by the instruction. Single cycle instruction can perform the following actions in one cycle:

1. Latch the current instruction.

2. Decode and execute the latched instruction.

3. Perform a write instruction to the instruction RAM.

4. Prefetched the instruction at the next or conditional address.

Multi-cycle instructions perform step 1 and 2 in the first cycle. Step 3 is executed in the cycle required by the instruction. Some instruction will perform multiple writes to the RAM, although in different cycles. Step 4 is always performed in the last cycle of the instruction.

The number of instructions executed by a program must be between two loop resolution clocks. The program executes its first instruction in the high phase of the loop resolution clock and stops the execution of the program before the occurrence of the LClock_Last. If the total time of execution for the program exceeds the limit, the program overflow flag is asserted and the execution of the program is stopped. The execution of the program is resumed by prefetching the instruction at location address 00h.

Architecture Restriction

Note:

The size of a HET program should be greater than one instruction, otherwise this instruction will be executed twice.

Note:

A HET program should not use the HWAG value during the last time slot of a resolution loop.

Note:

Any MOVE instruction that modifies a field in the next instruction to be executed will incur a wait cycle.

4.4.5 Multi-Resolution Scheme

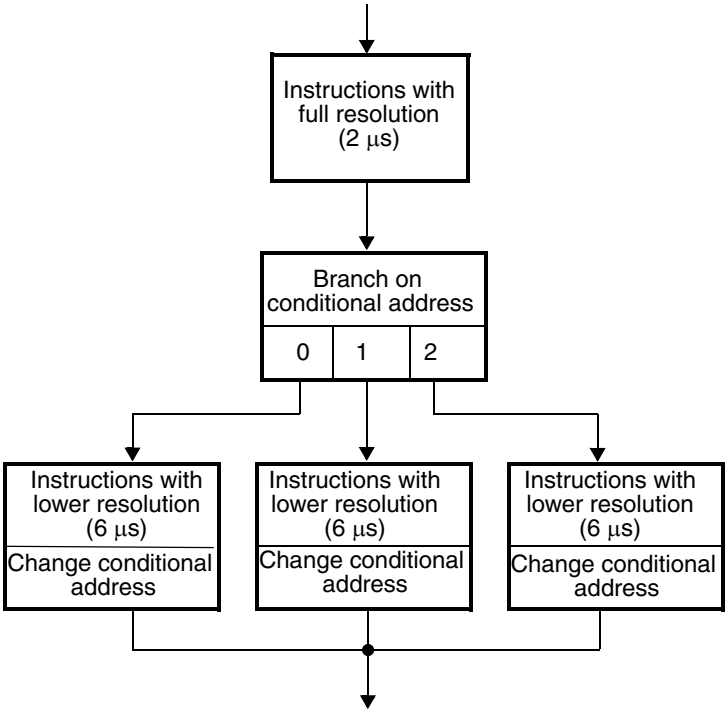
When full resolution is not required, special program sequences with lower resolution can be used to optimize use of time slots.

A lower resolution sequence consists of instructions that are not executed on every resolution loop, but only on every N resolutions. Unconditional Branch instruction and an index sequence, using a MOV64 instruction in each low resolution loop, is required to control this particular program flow. See Figure 11.

Note:

HR instructions must be placed in the main (full resolution) loop to ensure proper operation.

Figure 11. Multi-Resolution Operation Flow

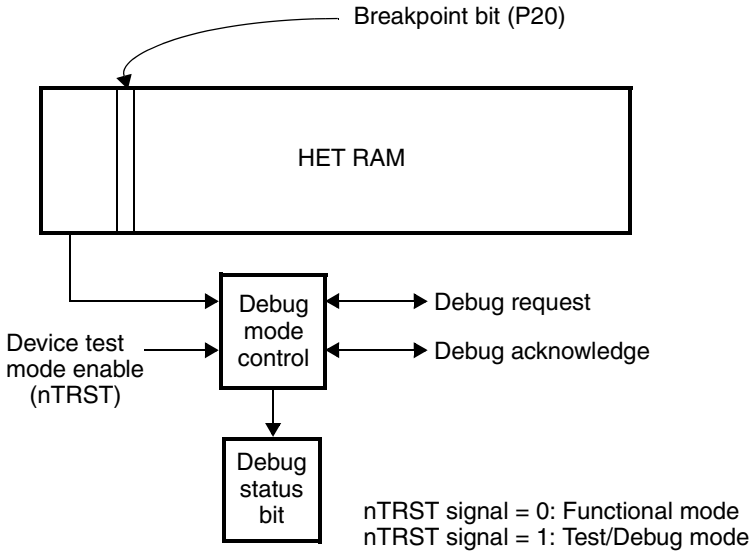


4.4.6 Debug Capability

The HET has built-in debug capability in its architecture that allows you to more easily debug your HET program. See Figure 12.

This debug capability is based on the breakpoint bit that is included as part of each HET instruction (bit P20 of any instruction). This bit can be programmed to a 1 to cause the HET program to freeze after the current instruction is executed. This action will also cause all internal HET state machines to freeze and send a debug request to the CPU. During the HET debug (i.e., HET breakpoint reached), you may still access the contents of all HET control registers, including a register that contains the current HET address to allow you to determine which instruction caused the breakpoint condition. If the Breakpoint bit is programmed to a 0, the instruction will then be executed and resume program operation.

Figure 12. Debug Control Configuration



When the device test mode is enabled and a breakpoint is reached in the HET program, the debug status bit (HETGCR[2]) will be set and the debug request signal will be sent to the CPU. The CPU will wait for the CPU instruction boundary and then send the debug acknowledge signal back to the HET. At this time the debugger may clear the debug status bit by writing a 1 to clear the bit in the HET peripheral frame. This will allow the HET to exit its debug mode when the debug acknowledge signal is deasserted by the CPU.

This debug feature may be used to conditionally freeze the HET operation by branching to a breakpoint instruction when a compare or capture condition exists. For example, an equality compare (ECMP) instruction could be used to branch to an instruction with the breakpoint bit set when the compare is equal. A software capture word (WCAP) instruction could be used to branch to an instruction with the breakpoint bit set when a certain external pin condition exists. When a breakpoint instruction is executed, the HET freezes and the CPU enters debug mode. At this time you may verify the state of the HET RAM or peripheral register and check the current address to determine the source of the breakpoint.

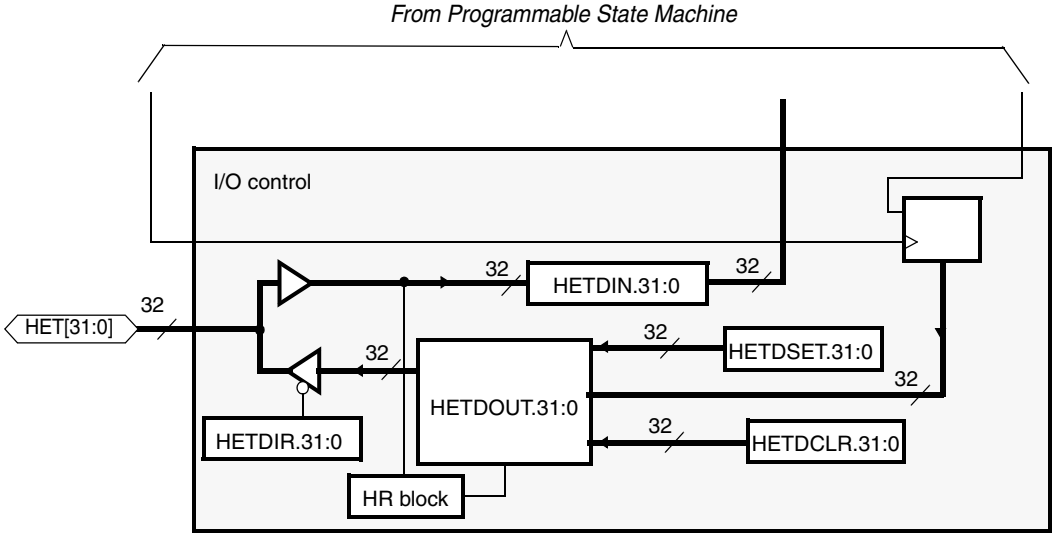
The HET also supports the direct assertion of debug mode from the CPU through the debug acknowledge signal. If this signal is asserted and the ignore suspend bit is not set, the HET then freezes all state machines and allows CPU accesses just as it does during the HET-initiated debug mode described above.

4.5 I/O Control

The HET has up to a 32-bit port. Refer to device specific data sheets for information concerning the number of HETIO available. All the HET pins available are programmable as either inputs or outputs.

These 32 I/Os have an identical structure connected to pins HET[31] to HET[0]. This structure allows any HET instruction to use these I/Os with a loop resolution accuracy. See Figure 13 for an illustration of the I/O control. Among these 32 I/Os, 24 have a special HR structure (pins HET[23] to HET[0]) based on the HR clock.

Figure 13. I/O Control



The general purpose I/Os, pins HET [31] to HET [0], can be used by the CPU as general purpose inputs or outputs using the control register HETDIN for reading and HETDOUT, HETDSET or HETDCLR for writing, depending on the type of action to perform. The HET pins used as general purpose inputs are sampled on each system clock.

Note: Using HETDSET and HETDCLR

The HETDCLR and HETDSET registers can be used to minimize the number of accesses to the peripheral to modify the output register and output pins. Actually, when the application needs to set or to reset n pins without changing the value of the others pins, the first possibility is to read HETDOUT, modify the content (AND, OR, etc.), and write the result into HETDOUT. This solution will take, in the best case, eight Cycles (LDR,AND/OR, STR), and could be interrupted by a function modifying the same register, which will result in a data coherency problem.

Using the HETDSET or HETDCLR registers, the application program must write the mask value (same mask value for the first option) to the register to set or reset the desired pins. This operation will take only three cycles (STR) and is not interruptible.

Example (C program): Set pins using the 2 methods.

```
unsigned int MASK;                  /* Variable that content the bit mask   */
volatile unsigned int *HETDOUT,*HETDSET;       /* Pointer to HET registers */
...
*HETDOUT = *HETDOUT | MASK;                /* Read-modify-write of HETDOUT */
*HETDSET = MASK;                    /* Set the pin without reading HETDOUT */
```

4.5.1 Loop Resolution Structure

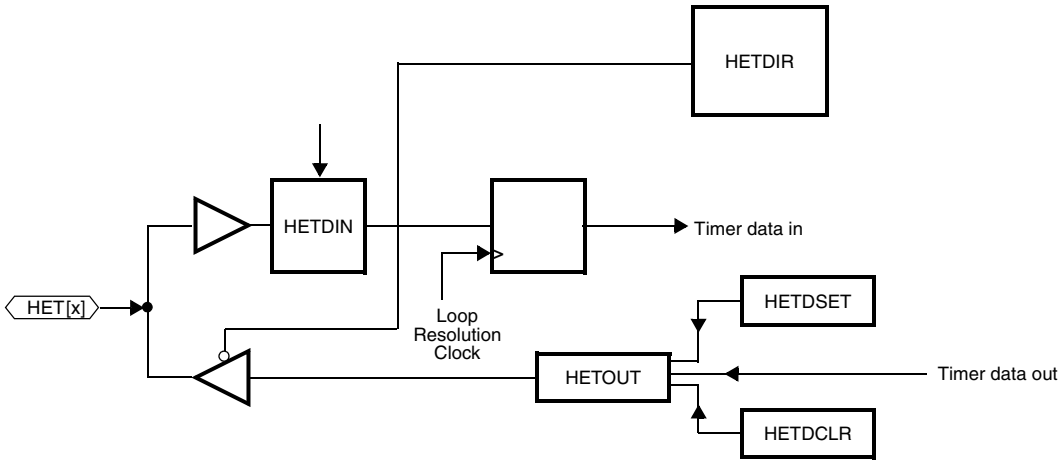
The HET use the pins HET [31] to HET [0] as input and/or output by the way of the instruction set. Actually, each pin could monitor the HET program or could be monitored by the HET program. By using the I/O register of the HET, the CPU is able to interact with the HET program flow.

When an action (set or reset) is taken on a pin by the HET program, the HET will modify the pin at the rising edge of the next resolution clock.

When an event occurs on a HET I/O pin, it is taken into account at the next rising edge of the resolution clock.

The structure of each pin is shown in Figure 14.

Figure 14. HET Loop Resolution Structure for Each Bit



4.5.2 HR Structure

Among the 32 I/Os, 24 have the special HR structure (pins HET [23] to HET [0]) based on the HR clock. See Figure 15. The HR clock frequency is programmed through register HETPFR[5:0]. See section on page 54.

In addition to the standard I/O structure pins, HET [23] through HET [0] have special HR hardware so that these pins can be used as HR input captures (using PCNT or WCAP) or HR output compares (using ECMP, MCMP or PWCNT).

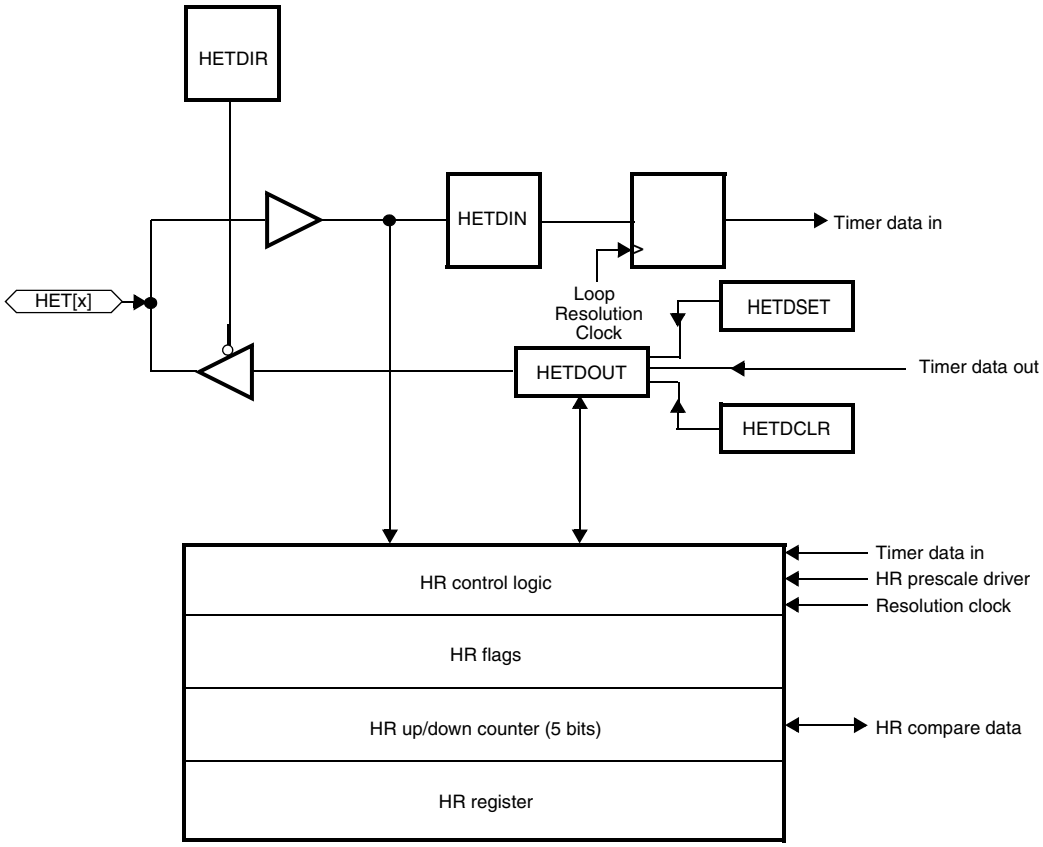
4.5.3 HR Block Diagram

Each time an HR instruction is executed on a given pin, the HR block for that pin is programmed (which HR function to perform and on which edges it should take an action) with the information in the instruction. The HR structure for each pin decodes the pin select field of the instruction and programs its HR structure if it matches.

Note:

Only one HR function is allowed per resolution structure. To enforce this restriction, the architecture shown below is programmed only once per resolution loop. This occurs when the first instruction is executed. You must ensure that only one instruction accesses a given pin in HR mode. The HR structure takes the information from the first instruction and ignores the remaining instructions.

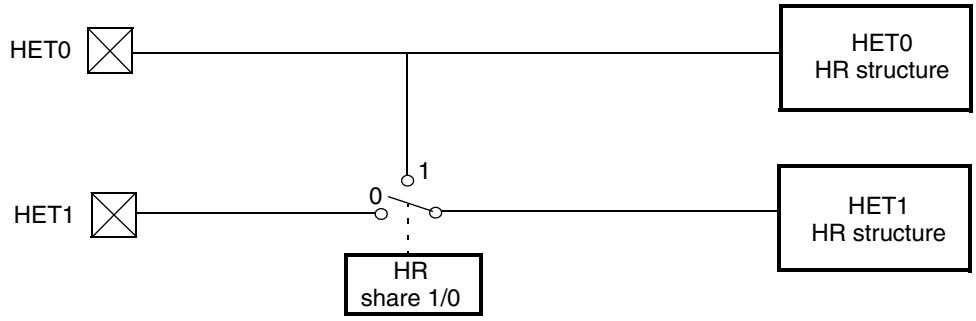
Figure 15. HR I/O Architecture



4.5.4 HR Structures Sharing

The HR SHARE control register bits allow two HR structures to share the same pin **for input capture only**. If these bits are set, the HR structures N and N+1 are connected to pin N. In this structure, pin N+1 remains available for general purpose input/output. See Figure 16.

Figure 16. Example of HR Structure Sharing for HET Pins 0/1



The following program gives an example how the hr share feature(HET0 HR structure and HET [1] HR structure shared) can be used for the PCNT instruction:

```
L00 PCNT { next=L01, type=rise2fall, pin=CC0 }
L01 PCNT { next=L00, type=fall2rise, pin=CC1 }
```

The HET [1] HR structure is also connected to the HET [0] pin. The L00_PCNT data field is able to capture a high pulse and the L01_PCNT captures a low pulse on the **same** pin (HET [0] pin).

4.5.5 XOR-shared HR Structure

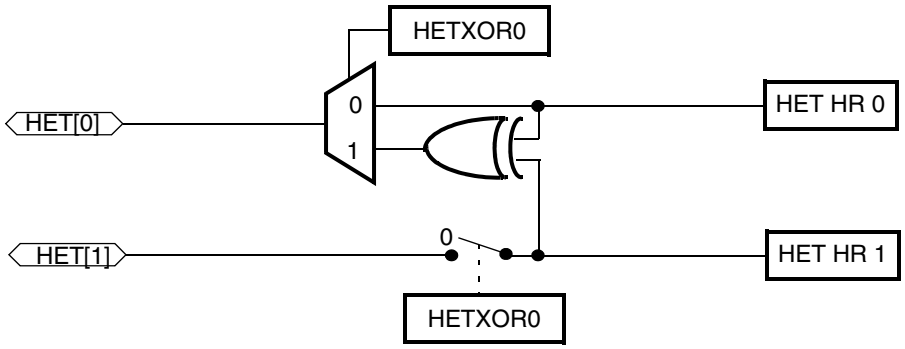
Usually the HET design allows only one HR structure to cause HR edges on a pin configured as output pin. According to section 5.4.5.9 the duration between two HR edges (which both doesn't line up with the LR clock) cannot be smaller than one loop resolution (1 LRP). To eliminate this constraint, the HETXOR register allows a logical XOR of the output signals of two consecutive HR structures. See Figure 17. In this way, it is possible to generate pulses smaller than the loop resolution clock since both edges are now base on the HR clock. This is especially required for symmetrical PWM. See Figure 18.

This apparatus consists of a XOR gate that is connected to the outputs of the HR structure of two consecutive pins. In this structure, pin N+1 remains available for general purpose input/output.

Note: Availability of XOR-shared features

The XOR-share feature is only available for certain TMS470 derivatives. Please Refer to the device specific data sheet.

Figure 17. XOR-shared HR I/O



The following HET program gives an example for **one** channel of the symmetrical PWM. The generated timing is given in Figure 18.

```
MAXC .equ 22
A_   .equ 0  ; HR structure HR0
B_   .equ 1  ; HR structure HR1

CN CNT   { next=EA, reg=A, max=MAXC }

EA ECMP  { next=EB, cond_addr=MA, hr_lr=HIGH, en_pin_action=ON, pin=A_,
           action=PULSELO, reg=A, data=17, hr_data=19 }
MA MOV32 { next=EB, remote=EA, type=IMTOREG&REM, reg=NONE, data=17, hr_data=19 }

EB ECMP  { next=CN, cond_addr=MB, hr_lr=HIGH, en_pin_action=ON, pin=B_,
           action=PULSELO, reg=A, data=5, hr_data=13 }
MB MOV32 { next=CN, remote=EB, type=IMTOREG&REM, reg=NONE, data=5, hr_data=13 }
```

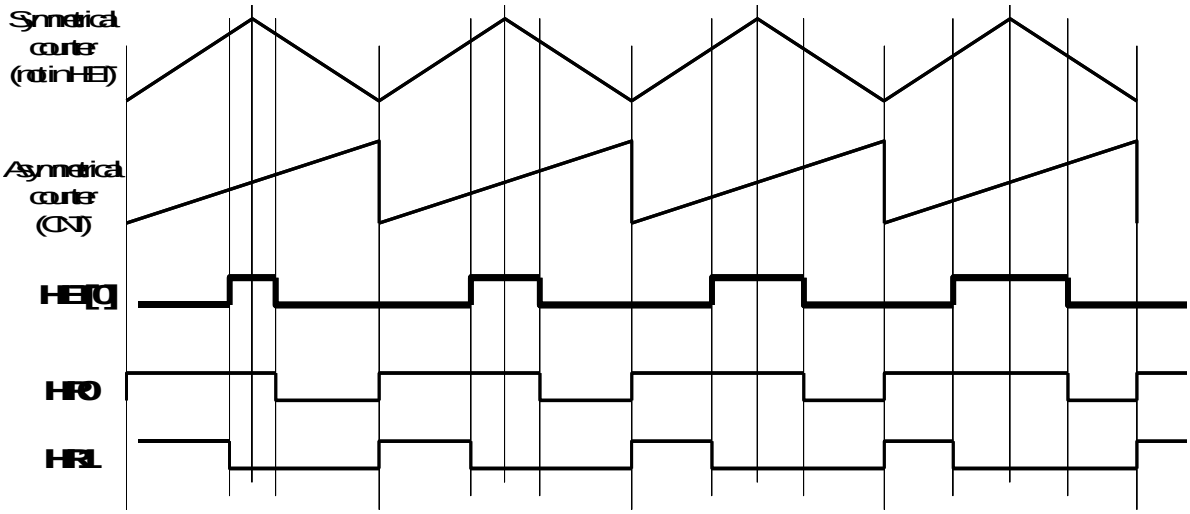
HET Settings and output signal calculation for this example program:

- Pin HET[0] and HET[1] are XOR-shared.
- HETPFR[31:0] register = 0x501 → Lr=32 and Hr=2 → time slots Ts = 64
- PWM period (determined by CNT_max field) = (22+1) * LRP = 736 HRP
- Length of high pulse of (HET[0] XOR HET[1]) =
  LH = (17*LRP+19*HRP)-(5*LRP+13*HRP)
  With lr=32 there is LRP = 32 * HRP, so
  LH = (563-173) * HRP = 390 HRP
- Duty cycle = DC = LH / PWM_period = 390 HRP / (736*HRP) = 53.0%

Figure 18 graphically shows the implementation of the XOR-shared feature. The first 2 waveforms (symmetrical counter and CNT) show a symmetric counter and asymmetric counter. The symmetric counter is shown only to

highlight the axis of symmetry and is not implemented in the HET. The asymmetric counter, which is implemented with a CNT instruction, needs to be set to the period of the symmetric counter. The next two waveforms (HR [0] and HR [1]) show the output of the HR structures, which are the inputs for the XOR gate to create the PWM output on pin HET[0]. Notice that the pulses of signal HET[0] are centered about the axis of symmetry.

Figure 18. Symmetrical PWM with XOR-sharing Output



4.5.6 HR/Low Resolution Bit

Four HR instructions have a dedicated hr_lr bit (HR/low resolution) allowing operation either in HR mode or in standard resolution mode by ignoring the HR field. Those instructions are ECMP, MCMP, PWCNT and WCAP. The hr_lr bit is P(7) (program field bit 7). By default, the hr_lr bit value is 0 which implies HR operation mode. However, setting this bit to one, allows the use of several HR instructions on a single HR pin. Only one instruction will be allowed to operate in HR mode (i.e., bit set to 0), but the others instructions can be used in standard resolution mode (i.e., bit set to 1).

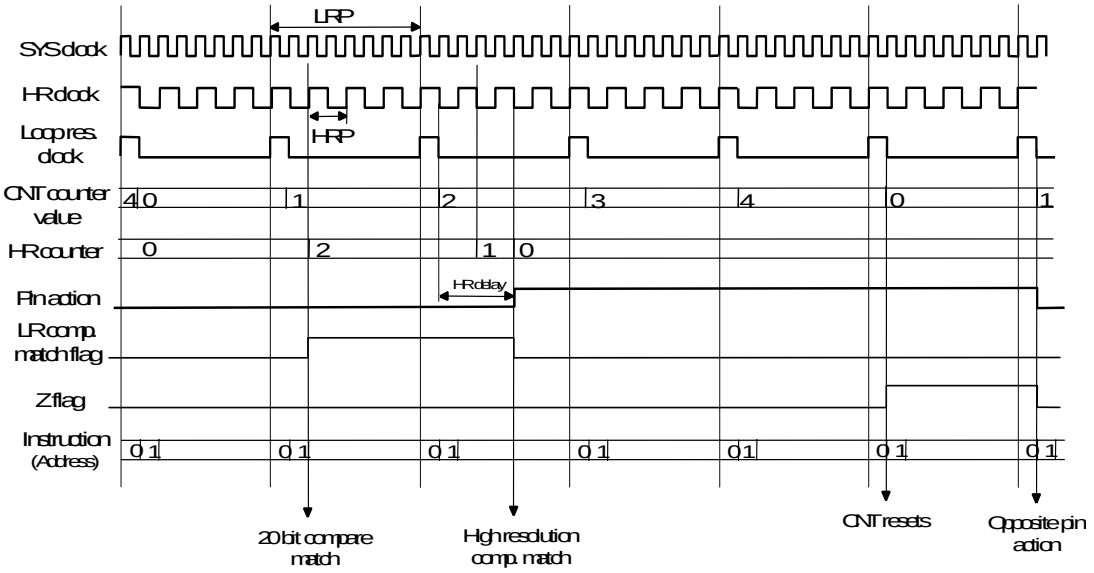
4.5.7 ECMP Execution Example (in HR Mode)

First, is the example of an HR ECMP with a linear prescale of /2 and a loop resolution prescale of /4. With a 32MHz system clock this results in a 62.5ns HR clock and a 0.25us resolution loop clock. This allows eight time slots for the program, which in this case will consist of one CNT and one ECMP instruction. The ECMP instruction is loaded into the HET RAM with a 25-bit compare value, the lower 5 bits representing the HR compare value.

When the 20-bit (low resolution) compare matches, the LR Comp Match flag will be set in the HR hardware for the corresponding pin and the HR compare value will be loaded from the five lower bits of the instruction data field to the HR counter. At the next resolution clock, the HR counter will count down at the HR clock frequency and perform the set action when it reaches zero.

Figure 19 shows where the 20-bit compare value is one and the HR compare value is two.

Figure 19. ECMP Execution Timings




```
HETPFR[31:0] register = 0x201 → lr=4 and hr=2 → time slots ts = 7
L00   CNT  {next=01h, reg=A, irq=OFF, max = 4 }
L01   ECMP {next=00h, en_pin_action=ON, cond_addr= 00h, pin=CC0, action=PULSEHI,
reg=A, irq=OFF, data= 1, hr_data = 0x10 }
; 20 bit compare value is 1 and the HR compare value is 2
```

Note: ECMP Opposite Actions

ECMP Opposite actions are always synchronized to the loop resolution clock.

Capture/Compare pins can be selected individually with instructions ECMP, SCMP, MCMP, PWCNT, MOVE64, DADM64, RADM64. A pin already defined as an output compare can be selected as an internal input in the instructions CCNT, WCAP, BR, and SHFT.

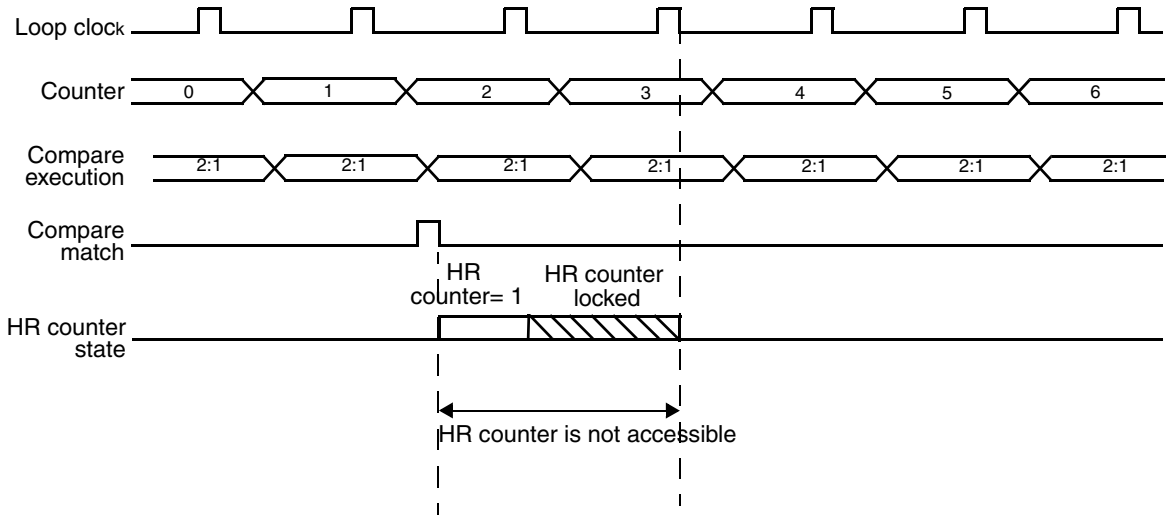
4.5.8 MCMP Execution Example

The MCMP instruction can also be used in HR mode. In this case operation is exactly the same as for the ECMP instruction except that the 20-bit low resolution is now the result of a magnitude compare rather than an equality compare. Otherwise operation is the same as for the MCMP 20-bit match setting the HR flag and loading the HR counter. The HR counter starts to decrement after the next resolution clock and performs the specified pin action. See Figure 20.

4.5.9 Limitation on ECMP and MCMP Operation in HR Mode

There is some limitation on the ECMP and MCMP operation in the HR mode, which Figure 20 illustrates.

Figure 20. ECMP Limitation Timing Diagram



ECMP or MCMP compares its own Data Field value with a register value which is loaded by a CNT instruction. The compare is a 20-bit comparison, even if ECMP or MCMP has an HR value. When this 20-bit comparison matches, and if the HR mode is used, the HR compare value is loaded in the dedicated HR counter. The pin action is supposed to happen in the next loop resolution cycle. The HR counter is locked during all this time, as shown in Figure 20.

In other words, when a compare matches, 2 loop resolution cycles must pass before a new compare (in HR mode) can occur.

Note:

If it is necessary to have a more accurate pulse time, using the HR XOR-shared features with 2 HR pin work around this limitation.

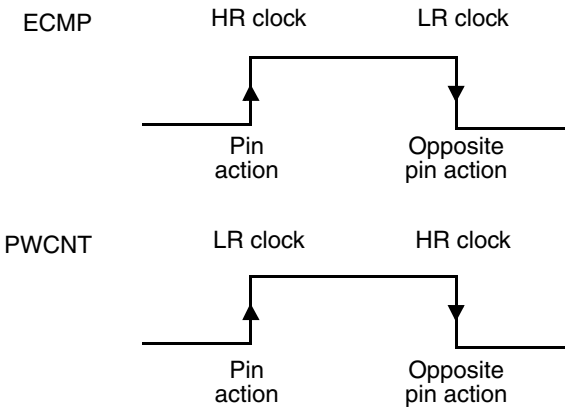
Note:

For the devices in X05 silicon technology, the limitation has been reduced to 1 loop resolution cycle.

4.5.10 PWCNT Execution Example (in HR Mode)

The PWCNT instruction may also be used in HR mode to generate pulse outputs with HR width. The PWCNT instruction operates conversely to the ECMP instruction. See Figure 21. For PWCNT, the opposite pin action is synchronous with the HR clock and for ECMP the pin action is synchronous with the HR clock. The PWCNT pin action is synchronous with the loop resolution clock.

Figure 21. High/Low Resolution Modes for ECMP and PWCNT



4.5.11 PCNT Execution Example (in HR Mode)

In conjunction with HR I/O pins, PCNT is able to capture an HR measurement of the high/low pulse time or periods of the input. As shown in Figure 21, at the first marker the input goes HIGH and the HR counter immediately begins to count. The counter increments and rolls over until the falling edge, where it captures the counter value into the HR capture register (the second marker). The PCNT instruction begins counting when the Synchronized Input

signal goes HIGH and captures both the 20-bit data field and the HR capture register into RAM when the synchronized input falls (third marker).

Note:

The HR capture value written into RAM is shifted appropriately depending on the loop resolution prescale divide rate (Lr).

Figure 22 shows what happens when the capture edge arrives *after* the HR counter overflows. This causes the incremented value to be captured by the PCNT instruction.

Figure 22. PCNT Instruction Timing (With Capture Edge After HR Counter Overflow)

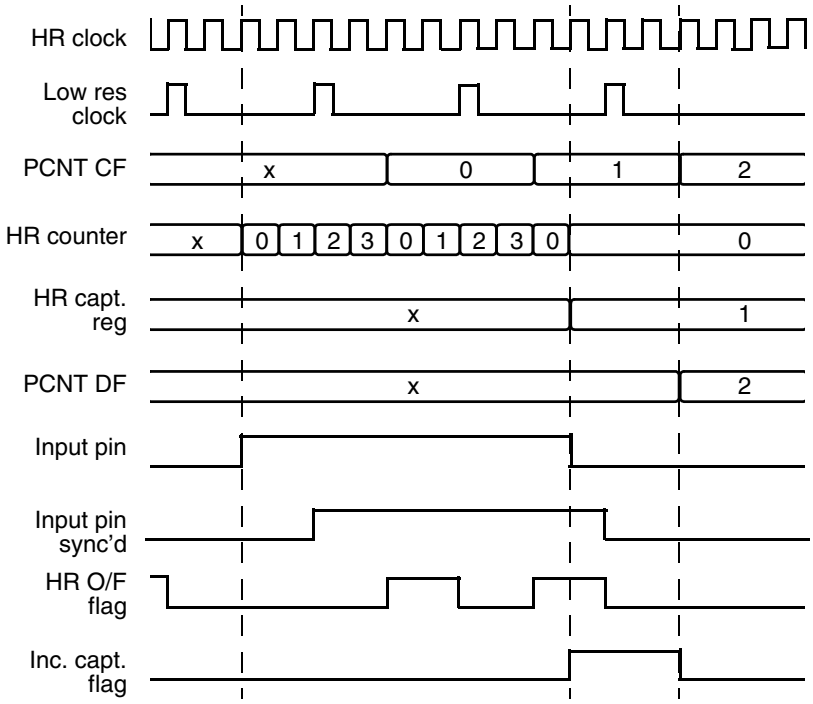
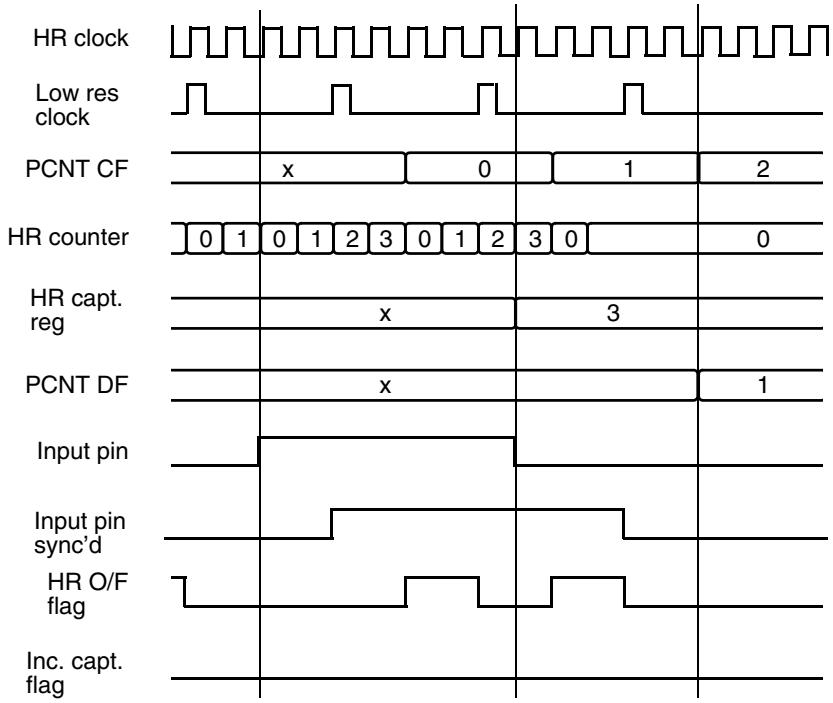


Figure 23 shows what happens when the capture edge arrives *before* the HR counter overflows. This causes the non-incremented value to be captured by the PCNT instruction.

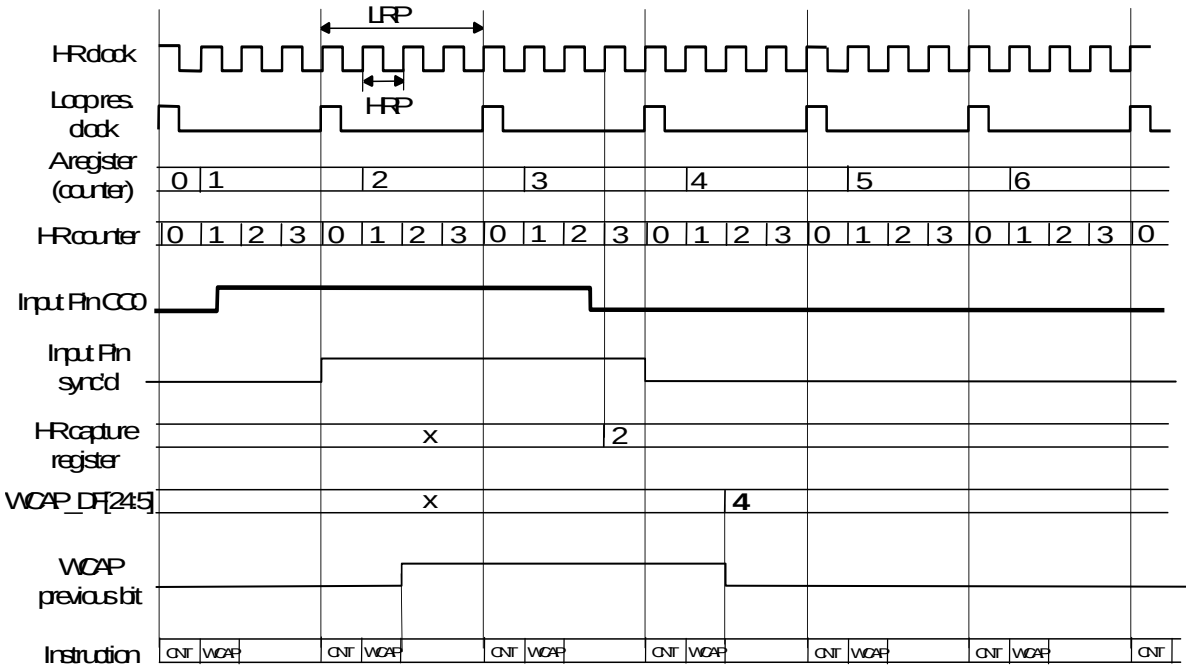
Figure 23. PCNT Instruction Timing (With Capture Edge Before HR Counter Overflow)



4.5.12 WCAP Execution Example (in HR Mode)

The WCAP instruction can be implemented for HR I/O. In this case the HR counter is always enabled (using the Count Always flag in the HR structure) and is synchronized with the resolution loop. When the specified edge is detected, the current value of the HR counter is captured in the HR capture register and written into the RAM after the next WCAP execution. The WCAP instruction effectively time stamps the a free run timer saved in a register (for example, register A shown in Figure 24).

Figure 24. WCAP Instruction Timing




```
HETPFR_register = 0x0200 → lr = 4, hr = 1, ts = 3

L00 CNT  {reg=A, max=0fffffh}
L01 WCAP {next=L00, cond_addr=L00, hr_lr=high, reg=A, event= FALL, pin=CC0,
          data=0}
```

In the timing of the example, the WCAP is configured to capture the counter when a **falling** edge occurs. The WCAP data field (WCAP_DF) is complete in the loop succeeding the loop, in which the edge occurred. In the figure example, the current value of the counter (4) is captured to WCAP_DF[24:5] and the value of the HR capture register (2) is transferred to the valid bits (according the lr prescaler) of WCAP_DF[4:0]. Therefore, in the example 0x090 is captured to WCAP_DF[24:0].

4.6 Interrupts and Exceptions

HET interrupts are generated by any instruction that has an interrupt enable bit in its instruction format. When the interrupt condition in an instruction is true and the interrupt enable bit of that instruction is set, an interrupt flag is then set in the HETFLG register. The address code for this flag is determined by the five LSBs of the current timer program address.

Note:

Enabling or disabling the interrupt generation of certain instructions (BR, ECNT, SHFT, WCAP) when the HET is turned on could falsify the operation of these instructions. This effect is due to the possible modification of the PRV bit by the HET during the read/modify/write of the CPU to modify the control field.

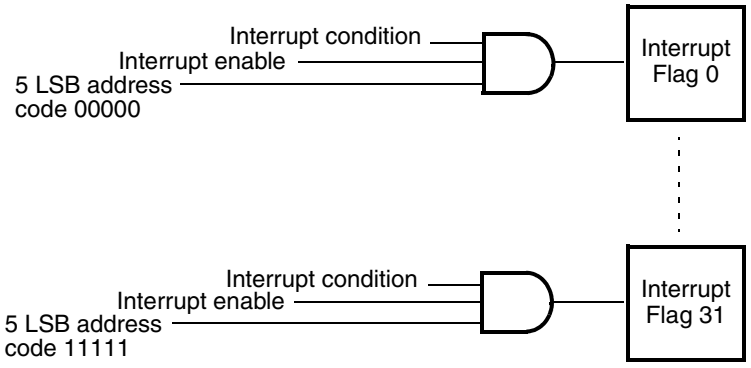
Table 6. Interrupt Sources and Corresponding Offset Values in Registers HETOFFx

Source No.	Offset Value
no interrupt	0
Instruction 0, 32, 64...	1
Instruction 1, 33, 65...	2
:	:
Instruction 31, 63, 95...	32
Program Overflow	33
APCNT underflow:	34
APCNT overflow	35

To execute the interrupt service routine, the main CPU must first determine which source created the interrupt request. This is done by reading the offset register (HETOFFx) which gives the number of the source. Reading the offset register will automatically clear the source flag which created the request. However, if you decide not to use the offset register to check the interrupt source, the flag should be cleared once the interrupt has been serviced.

The instructions capable of generating interrupts are listed in Table 30, “Interrupt Capable Instructions,” on page 81.

Figure 25. Interrupt Servicing



Each interrupt source is associated with a priority level (level 1 or level 2). When multiple interrupts with the same priority level occur during the same loop resolution, the lowest flag bit is serviced first. Two interrupt requests are generated by the HET for the central interrupt module (CIM). Figure 25 shows how the interrupts are serviced.

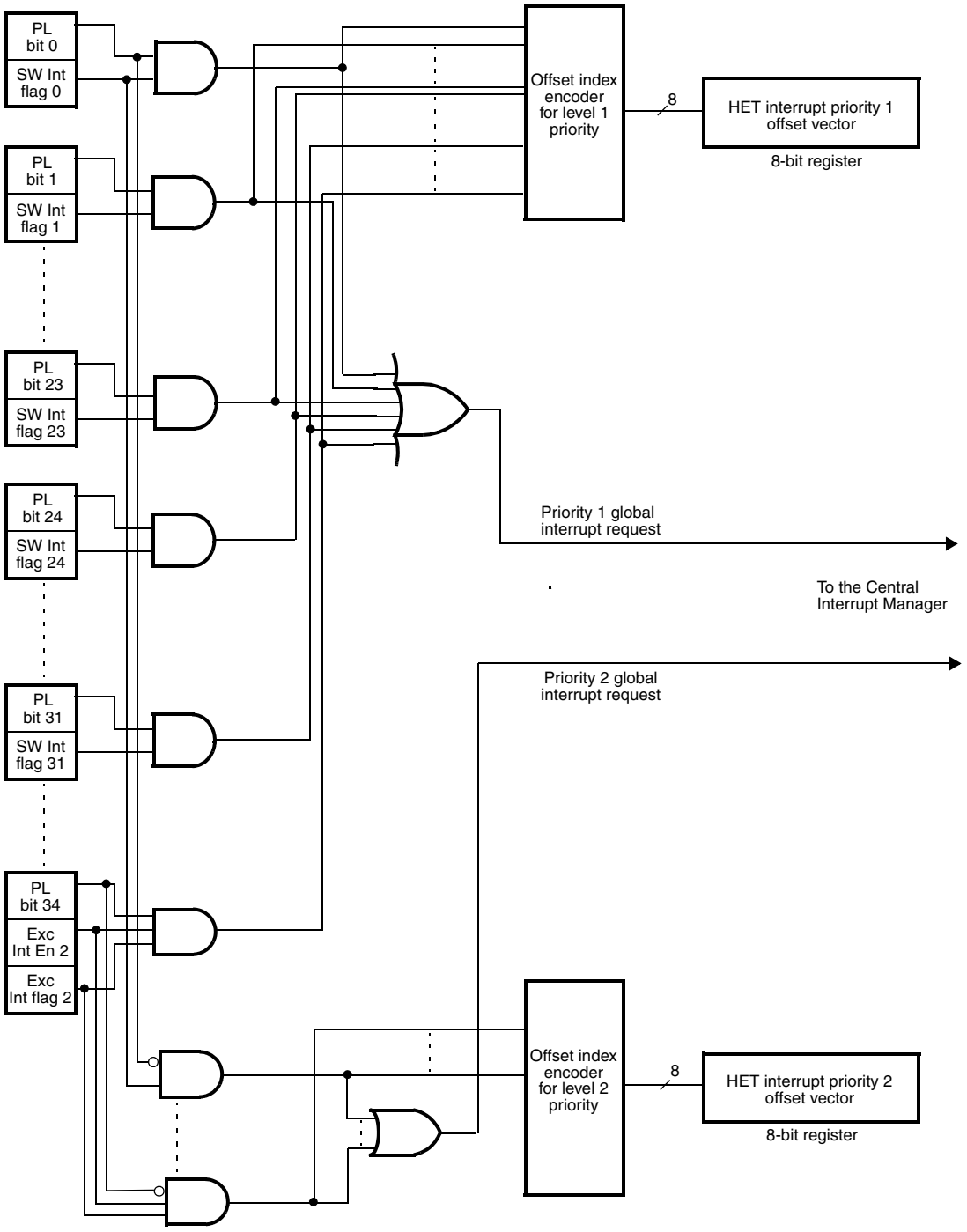
The HET can generate three types of exception from the following:

- ❏ Program overflow
- ❏ APCNT underflow (see section 5.1.2 on page 48)
- ❏ APCNT overflow (see section 5.1.3 on page 48)

Hardware Priority Scheme:

If two or more software interrupts are pending on the same priority level, the offset value will show the one with the highest priority. The interrupt with the highest priority is the one with the lower offset value. This scheme is hard-wired in the offset encoder. See Figure 26.

Figure 26. Interrupt Flag/Priority Level Architecture



5 Angle Functions

Engine management systems require an angle-referenced time base to synchronize signals to the engine toothed wheel. The HET has two methods to provide such a time base:

- ❏ A software angle counter for low-end engine systems. The reference is created by the HET using three dedicated instructions with fractional angle steps equal to /8, /16, /32, /64.
- ❏ An interface to a hardware angle counter for mid- and high-end engine systems. This provides a connection to an external hardware angle generator with angle steps up to /512. The interface allows the hardware angle counter to be embedded inside the HET for future configurations.

5.1 Software Angle Generator

The HET provides three specialized count instructions to generate an angle referenced time base synchronized to an external reference signal (the toothed wheel signal) that defines angular reference points.

The time base is used to generate fractional angle steps between the reference points. The step width K (= 8, 16, 32, or 64) programmed by you defines the angle accuracy of the time base. These fractional steps are then accumulated in an angle counter to form the absolute angle value.

The first counter APCNT incremented on each resolution clock measures the periods P(n) of the external signal. The second counter SCNT counts by step K up to the previous period value P(n-1), measured by APCNT, and then recycles. The resulting period of SCNT is the fraction P(n-1) / K. The third counter ACNT accumulates the fractions generated by SCNT.

A HET timer program can only have one angle generator. Figure 27 illustrates the basic operation of APCNT, SCNT, and ACNT.

Due to stepping, the final count of SCNT does not usually exactly match the target value P(n-1). Figure 28 illustrates how SCNT compensates for this feature by starting each cycle with the remainder (final count - target) of the previous cycle.

Figure 27. Operation of HET Count Instructions

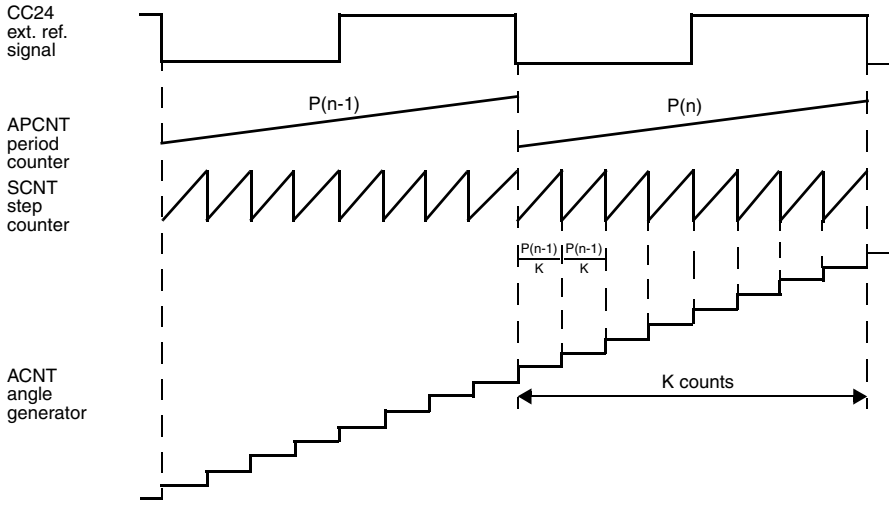
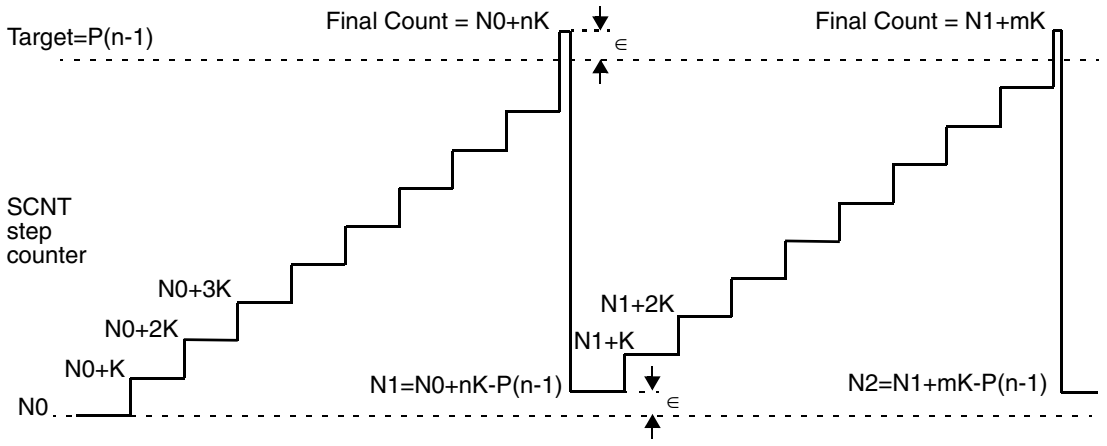
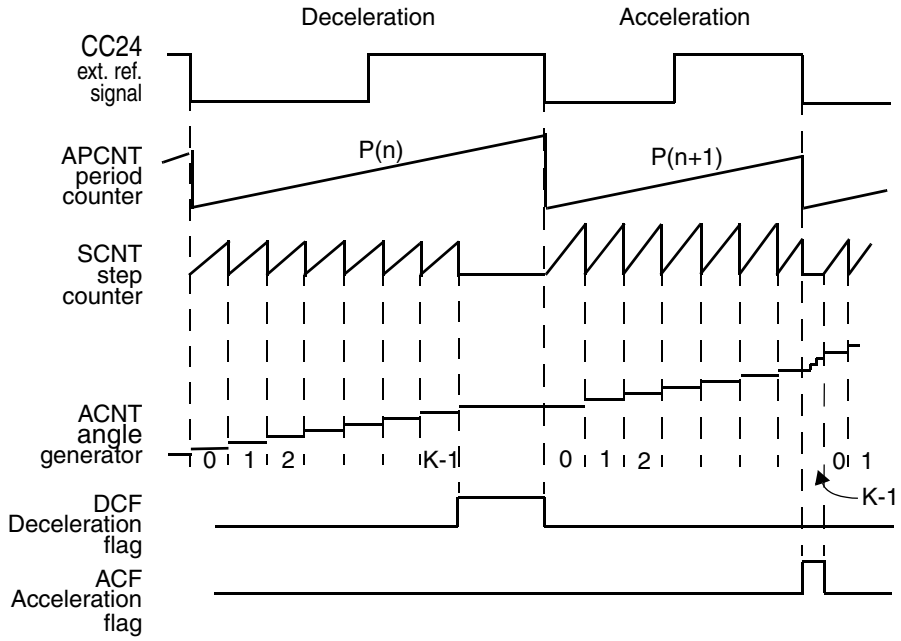


Figure 28. SCNT Count Operation



ACNT detects period variations of the external signal measured by APCNT and compensates related counting errors. A period increase is flagged in the deceleration flag. A period decrease is flagged in the acceleration flag. If no variation is flagged, ACNT increments the counter value each time SCNT reaches its target. If acceleration is detected, ACNT increments the counter value on each timer resolution (fast mode). If deceleration is detected, ACNT is stopped. Figure 29 illustrates how the compensations for acceleration and deceleration operate.

Figure 29. ACNT Period Variation Compensations



5.1.1 Singularities

Singularities (gaps, in this case, from missing teeth in a toothed wheel) in the external reference signal can be masked. The start and end of singularities are defined by gap start and gap end values specified in SCNT and ACNT. When ACNT reaches gap start or gap end, it sets/resets the gap flag.

While the gap flag is set, new periods of the external reference signal are ignored for angle computation. SCNT uses the last period measured by APCNT just before gap start.

Figure 30 and Figure 31 illustrate the behavior of the angle generator during a gap after a deceleration or acceleration of the HET.

Figure 30. HET Timings Associated with the Gap Flag (ACNT Deceleration)7

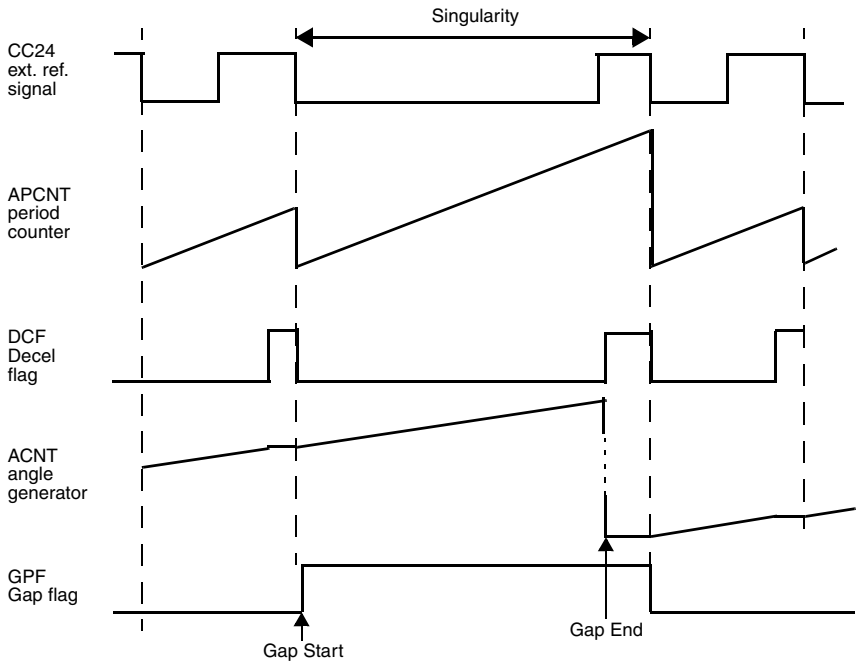
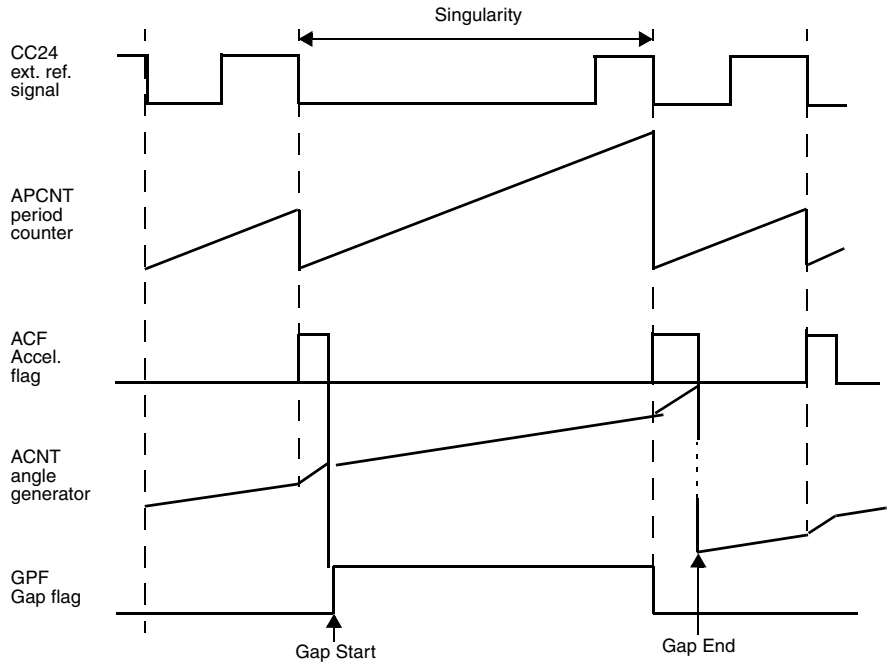


Figure 31. HET Timings Associated with the Gap Flag (ACNT Acceleration)



5.1.2 APCNT Underflow

The fastest valid external signal APCNT can accept must satisfy the following condition:

$$\text{Step Width K} \quad < \quad \frac{\text{Period Min.}}{\text{Resolution (LRP)}}$$

This condition fixes the maximum possible step width once the minimum period and the resolution of an application are specified.

If a period value accidentally falls below the minimum allowed, APCNT stops the capture of these periods and sets the APCNT underflow interrupt flag located in the exceptions interrupt control register. In such a situation, SCNT and ACNT continue to be executed using the last valid period captured by APCNT.

5.1.3 APCNT Overflow

The slowest valid external signal APCNT can measure must satisfy the following condition:

$$\frac{\text{Period Max}}{\text{Resolution}} \quad < \quad 1048575$$

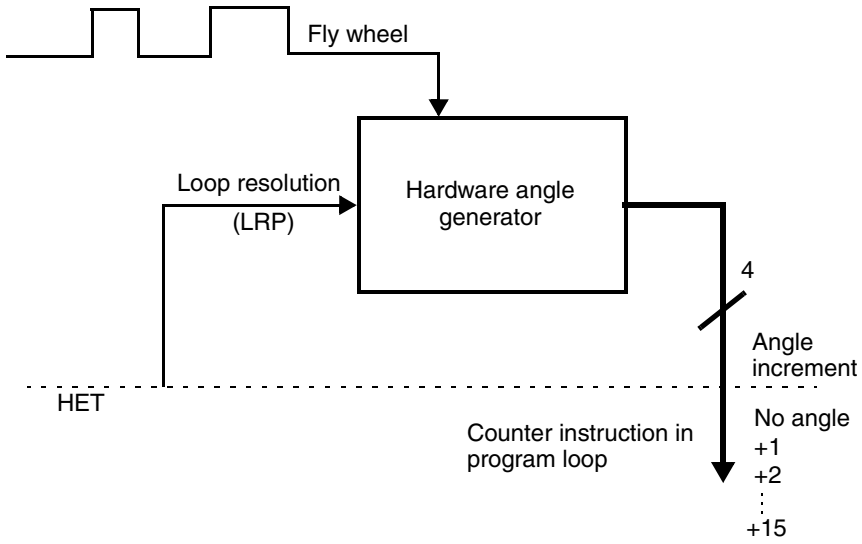
When this limit is reached (APCNT Count equals all 1's), APCNT stays at a maximum count (stops counting). APCNT remains in this position until the next specified capture edge is detected on the selected pin and sets the APCNT overflow interrupt flag located in the exceptions interrupt control register. In this situation, SCNT and ACNT continue to be executed using the maximum APCNT period count.

5.2 Interface to a Hardware Angle Generator

The interface to a hardware angle generator (HWAG) handles the following information:

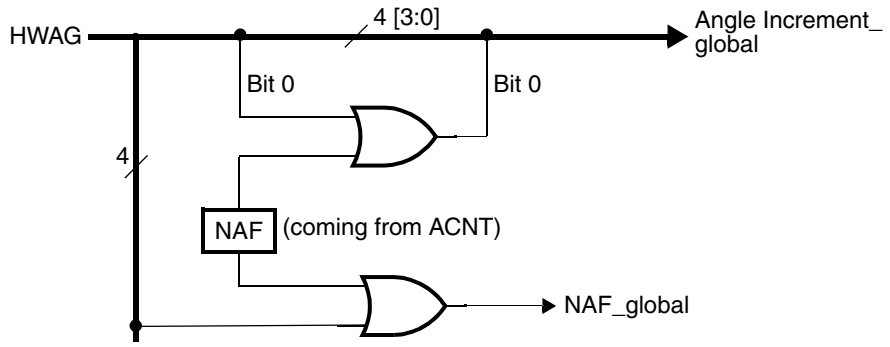
- The HET resolution clock (LRP) is provided to the HWAG to calculate the new angle increments according to the speed of the engine toothed wheel during the resolution (LRP). See Figure 32.
- A 4-bit bus provides the angle increment that gives the incremental value of angle-related counters since the last resolution clock. The normal way to increment these counters is to add 1 to the counters, but the angle increment bits are allowed to increment by 2, 3, 4, and so forth to be consistent with the loop resolution (allow multiple angle counter increments inside a program loop).

Figure 32. Hardware Angle Generator Interface



The CNT in angle mode uses either angle increment or new angle flag (coming from the SW angle generator) to increment the virtual timer. The logic design comprises two OR gates as shown in Figure 33.

Figure 33. Angle Increment and NAF Count Logic



When the angle mode is set, the CNT instruction has a different mode of operation. The CNT data field increments on each loop resolution by the value specified by angle increment, as shown in the following example:

+0	When angle increment	= 0
+1	When angle increment	= 1
+2	When angle increment	= 2
+3	When angle increment	= 3

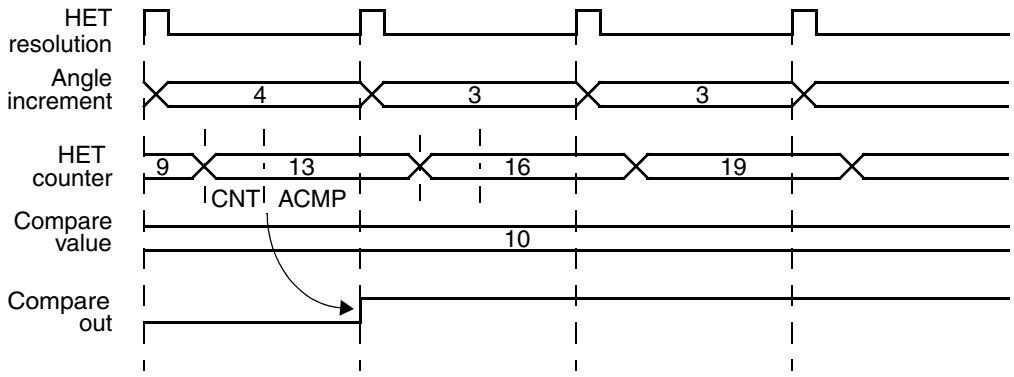
When the HWAG is disconnected, the angle increment global comes from the SWAG and is set depending on the new angle flag.

When NAF = 0, then angle increment global = 0
When NAF = 1, then angle increment global = 1

Since the angle value could be increased by 2, 3, 4, etc., the compare value could be in between the old angle value and the new angle value of the HET angle counter (where new angle value equals old angle value, plus angle increment). To perform an angle compare that guarantees not to miss a compare value, the HET provides the ACMP instruction. See Figure 34.

The ACMP instruction is more than an equal compare. It performs an in-between comparison operation (old value < compare value ≤ new angle value) to match the toothed wheel position.

Figure 34. ACMP Operation Example



With the ACMP instruction, the compare that is performed will be:

$9 < 10 \leq 13$

The comparison performed does the following operations in parallel:

ACMP value > HET angle counter — angle increment

ACMP value ≤ HET angle counter

The angle increment modification performed by the HWAG will be synchronous to an engine toothed wheel edge. For more information about the hardware angle counter, refer to the *Hardware Angle Generator User' Guide (SPNU203)*.

6 HET Control Registers

Table 7 presents the summary of the HET control registers.

Table 7. Control Registers Summary

Offset Address†	Mnemonic	Name	Description	Page
0x00	HETGCR	Global Configuration Register	Controls how the HET interfaces with the CPU and multiple HETs.	57
0x04	HETPFR	Prescale Factor Register	Controls the number of available time slots in a HET program loop.	61
0x08	HETADDR	HET Current Address Register	Contains the current HET program address.	62
0x0C	HETOFF1	Offset Level 1 Register	Provides the offset that represents the pending high priority interrupt.	64
0x10	HETOFF2	Offset Level 2 Register	Provides the offset that represents the pending low priority interrupt.	66
0x14	HETEXC1	Exception Control Register 1	Enable and priority for APCNT overflow, APCNT underflow, and program overflow exceptions.	67
0x18	HETEXC2	Exception Control Register 2	Contains APCNT overflow, APCNT underflow, and program overflow flags.	68
0x1C	HETPRY	Interrupt Priority Register	Contains interrupt priority based on line of HET code or exception.	69
0x20	HETFLG	Interrupt Flag Register	Contains interrupt flag based on the line of HET code.	70
0x24		Reserved		
0x28		Reserved		
0x2C	HETHRSH	HR Share Control Register	Allows you to share adjacent HR pins	71
0x30	HETXOR‡	HR XOR control register	Allow you to XOR adjacent HR pins	72
0x34	HETDIR	Data Direction Register	Configures the corresponding pin as an input or an output.	73
0x38	HETDIN	Input Data Register	Reflects the current value on the pin.	73
0x3C	HETDOUT	Output Data Register	Specifies output value to the pins (when configured as an output).	74
0x40	HETDSET	Set Data Register	Sets the bits in the HETDOUT register.	74
0x44	HETDCLR	Clear Data Register	Clears the bits in the HETDOUT register.	75

A) The physical address (base + offset) of these registers is device specific. See the device specific data sheet to verify the register base address.

B) The HETXOR register is only available for certain TMS470 derivatives. Refer to the device specific data sheet.

Accesses to the peripheral registers can be done in byte, half-word, or word format.

For the registers having a read/clear, read/set structure, the update is as follows:

- ❏ Writing a 1 performs the action (set or clear depending on the structure).
- ❏ Writing a 0 has no effect (no change for the bit).

Table 8. Read/Write Conventions

Action	Description
R	Read
W	Write
C	Clear
S	Set
n	Value after reset
RWP	Read in all modes, write in privilege mode only

Note:

In Table 35, the shaded cells have indeterminate read values, and writes have no effect.

6.1 HET Register Summary

Figure 35 summarizes all the HET registers.

Figure 35. HET Registers

Offset Address Register	31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x00 HETGCR	Reserved							Power Down	Reserved							CLK Master/ Slave
	Reserved							64-Bit Access	Reserved					Debug Status Flag	Ignore Susp.	Turn On/Off
0x04 HETPFR	Reserved															
	Reserved							Loop Resolution Prescale Factor Code			Reserved			HR Prescale Factor Code		
0x08 HETADDR	Reserved															
	Reserved										HETADDR.7:0					
0x0C HETOFF1	Reserved															
	Reserved										OFFSET1.7:0					
0x10 HETOFF2	Reserved															
	Reserved										OFFSET2.7:0					

Figure 35. HET Registers (Continued)

Offset Address Register	31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x14 HETEXC1	Reserved							APCNT OVRFL ENA	Reserved							APCNT UNRFL ENA
	Reserved							PRGM OVRFL ENA	Reserved					APCNT OVRFL PRY	APCNT UNRFL PRY	PRGM OVRFL PRY
0x18 HETEXC2	Reserved															
														APCNT OVRFL FLAG	APCNT UNRFL FLAG	PRGM OVRFL FLAG
0x1C HETPRY	HETPRY.31:16															
	HETPRY.15:0															
0x20 HETFLG	HETFLAG.31:16															
	HETFLAG.15:0															
0x2C HETHRSH	Reserved															
	Reserved		HR SHARE 23/22	HR SHARE 21/20	HR SHARE 19/18	HR SHARE 17/16	HR SHARE 15/14	HR SHARE 13/12	HR SHARE 11/10	HR SHARE 9/8	HR SHARE 7/6	HR SHARE 5/4	HR SHARE 3/2	HR SHARE 1/0		

Figure 35. HET Registers (Continued)

Offset Address Register	31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x30 HETXOR	Reserved															
	Reserved				XOR SHARE 23/22	XOR SHARE 21/20	XOR SHARE 19/18	XOR SHARE 17/16	XOR SHARE 15/14	XOR SHARE 13/12	XOR SHARE 11/10	XOR SHARE 9/8	XOR SHARE 7/6	XOR SHARE 5/4	XOR SHARE 3/2	XOR SHARE 1/0
0x34 HETDIR	HETDIR.31:16															
	HETDIR.15:0															
0x38 HETDIN	HETDIN.31:16															
	HETDIN.15:0															
0x3C HETDOUT	HETDOUT.31:16															
	HETDOUT.15:0															
0x40 HETDSET	HETDSET.31:16															
	HETDSET.15:0															

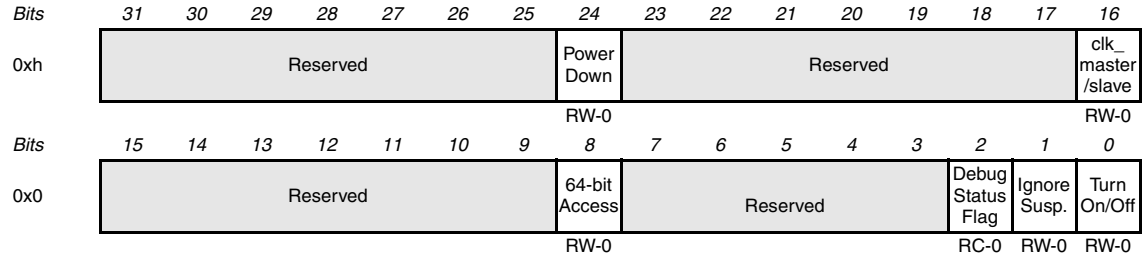
Figure 35. HET Registers (Continued)

Offset Address Register	31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x44 HETDCLR	HETDCLR.31:16															
	HETDCLR.15:0															

6.2 Global Configuration Register (HETGCR)

Figure 36 and Table 9 Ilustrate the HETGCR register.

Figure 36. Global Configuration Register (HETGCR)



RW = Read/Write, RC = Read/Clear, -*n* = value after reset

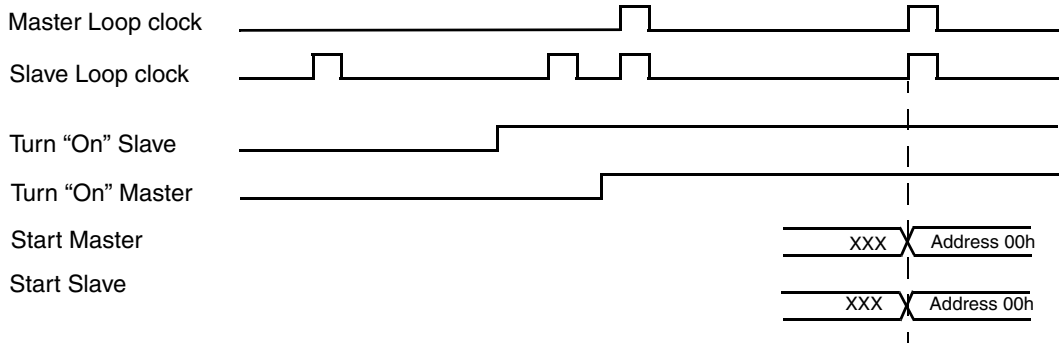
Table 9. Global Configuration Register (HETGCR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		
24	Power-Down		HET power-down mode bit. When Power-Down = 1, the HET internal clocks are stopped. This leaves the module in a static state in which it consumes the lowest possible current. To release the power-down mode, this bit must be written to 0; however, to enable the clock (stopped in power down), which allows writing to the register, the signal PPWNOVR must be set to a high level. After setting Turn-off, you must delay until the end of the timer program loop before putting the HET in power-down mode. This can be done by waiting until the HET PROGRAM ADDRESS becomes zero. Any operation mode (read/write):
		0	HET clock is running
		1	HET clock is stopped (power-down mode)
23–17	Reserved		

16	clk_master/ slave		This bit is used to synchronize multi-HETs. If set (HET is master), the HET outputs a signal to synchronize the prescalers of the slave HET. By default, this bit is reset, which means a slave configuration. Any operation mode (read/write):
		0	HET is configured as a slave.
		1	HET is configured as a master.
			Note: This bit must be set to 1 for single-HET configuration.
15–9	Reserved		
8	64-bit Access		Any operation mode (read/write):
		0	Timer RAM is accessed 32 bits at a time
		1	Timer RAM is accessed 64 bits at a time
7–3	Reserved		
2	Debug Status Flag		This flag is set when the device test mode is set and a breakpoint is reached in the HET program. A debug request signal is asserted to the main CPU. Any operation mode (read):
		0	No HET instruction with a breakpoint has been reached since the flag was last cleared.
		1	A HET instruction with a breakpoint has been reached since the flag was last cleared.
			Any operation mode (write):
		0	No effect.
		1	Clears the bit.
1	Ignore Suspend		When ignore suspend = 0, the timer operation is stopped on suspend (the current timer instruction is completed). Timer RAM can be freely accessed during suspend. When set to 1, the suspend is ignored and the HET continues operating. Any operation mode (read/write):
		0	HET stops when a software breakpoint is encountered.
		1	HET ignores software breakpoints.

0	Turn On/Off		When turn on/off = 0, the timer program stops executing. Turn-off is automatically delayed until the current timer program loop is completed. After enabling Turn-off, you must delay until the end of the timer program loop before putting the HET in power-down mode. This can be done by waiting until the HET PROGRAM ADDRESS becomes zero.Turn-off does not affect the content of the timer RAM, ALU registers, or control registers. Turn-off resets all flags. See Figure 37. Any operation mode (read/write): 0 HET is OFF. 1 HET is ON. Turn-off does not affect the state of the pins. You must set/reset the timer pins when they are turned off, or re-initialize the timer RAM and control registers before a reset. After a CPU reset, the timer is turned off by default. When turn off/on equals 1, timer program execution starts synchronously to the Loop clock. In case of multiple HETs configuration, the slave HETs are waiting for the loop clock to come from the master before starting execution. Then, the timer address points automatically address 00h (corresponding to program start).

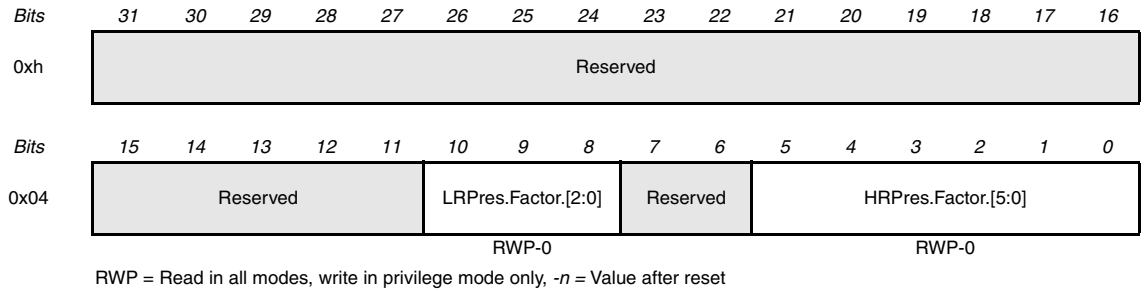
Figure 37. Multiple HETs Turn-On Sequence



6.3 Prescale Factor Register (HETPFR)

Figure 38 and Table 13 present the HETPFR.

Figure 38. Prescale Factor Register (HETPFR)



RWP = Read in all modes, write in privilege mode only, *-n* = Value after reset

Table 10. Prescale Factor Register (HETPFR) Field Descriptions

Bit	Name	Value	Description
31–11	Reserved		
10–8	Loop Resolution Pre-scale Factor Code		The binary code programmed in the register gives the loop resolution (LR). The LR pre-scale factor code and the HR pre-scale factor code define the number of time slots available. See Table 11 for the loop resolution encoding format, calculated by: *Time Slots Available = HR Prescale Divide rate X Loop Resolution Prescale Divide rate* The results apply to any operation mode (read/write).
8–6	Reserved		
5–0	HR Prescale Factor Code		The binary code programmed in the register gives the HR. See Table 12 for the HR encoding format. The results apply to any operation mode (read/write).

Table 11. Loop Resolution Encoding Format

Loop Resolution Prescale Factor Code			Loop-Res Prescale Divide Rate
2	**1**	**0**	
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	Reserved
1	1	1	Reserved

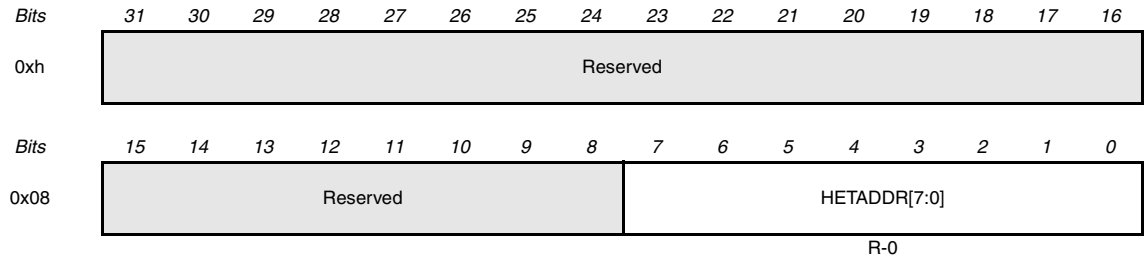
Table 12. HR Encoding Format

HR Prescale Factor code						HR Prescale Divide Rate
5	**4**	**3**	**2**	**1**	**0**	
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
0	0	0	0	1	1	4
0	0	0	1	0	0	5
:	:	:	:	:	:	:
:	:	:	:	:	:	:
1	1	1	1	0	1	62
1	1	1	1	1	0	63
1	1	1	1	1	1	64

6.4 HET Current Address Register (HETADDR)

Figure 39 and Table 13 present the register for HETADDR.

Figure 39. HET Current Address Register (HETADDR)



R = Read, *-n* = Value after reset

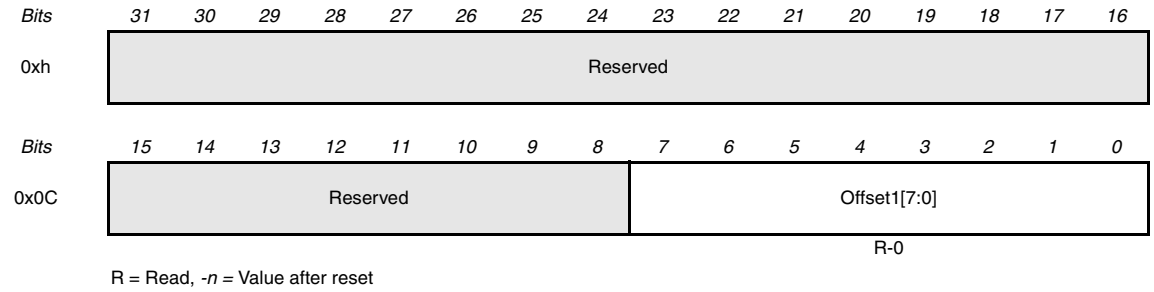
Table 13. HET Current Address Register (HETADDR) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		
7–0	HETADDR[7:0]		Any operation mode (read): Current HET program address register. Any operation mode (write): Writes have no effect.

6.5 Offset Index Priority Level 1 Register (HETOFF1)

Figure 40 and Table 14 present the HETOFF1 register.

Figure 40. Offset Index Priority Level 1 Register (HETOFF1)



R = Read, -*n* = Value after reset

Table 14. Offset Index Priority Level 1 Register (HETOFF1) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		
7–0	Offset[7:0]		HETOFF1[7:0] indexes the currently pending high-priority interrupt. Offset values and sources are listed below and the interrupt encoding format is presented in Table 15. Any operation mode (write): Writes have no effect Any operation mode (read): Read of these bits determines the pending HET interrupt; the corresponding flag (in the HETFLG) is also cleared. Emulation mode (read): Read of these bits determine the pending HET interrupt but the corresponding flag is not cleared.

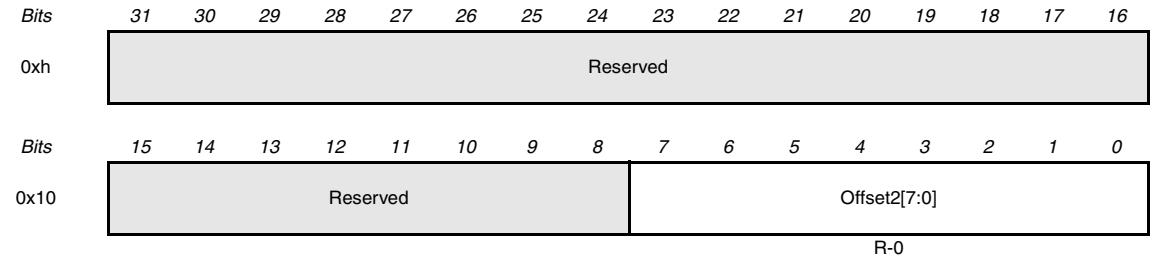
Table 15. Interrupt Offset Encoding Format

Source No.	Offset Value
no interrupt	0
Instruction 0, 32, 64...	1
Instruction 1, 33, 65...	2
:	:
Instruction 31, 63, 95...	32
Program Overflow	33
APCNT underflow:	34
APCNT overflow	35

6.6 Offset Index Priority Level 2 Register (HETOFF2)

Figure 41 and Table 16 describe the HETOFF2 register.

Figure 41. Offset Index Priority Level 2 Register (HETOFF2)



R = Read, -*n* = Value after reset

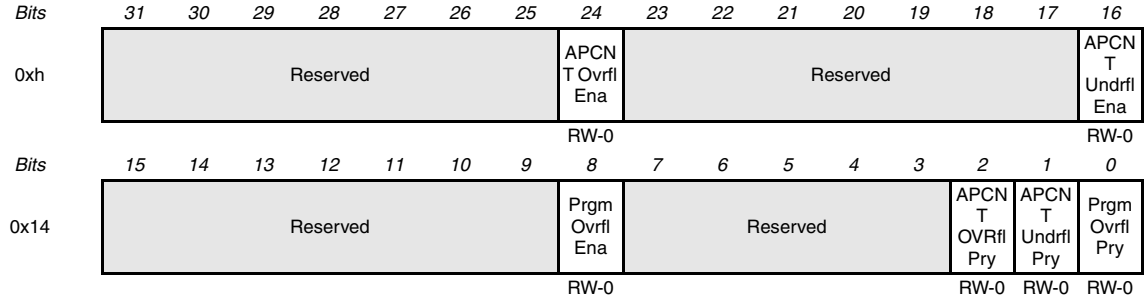
Table 16. Offset Index Priority Level 2 Register (HETOFF2) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		
7–0	Offset2[7:0]		HETOFF2[7:0] indexes the currently pending low-priority interrupt. Offset values and sources are listed in Table 35 on page 54. Any operation mode (write): Writes have no effect Any operation mode (read): Read of these bits determines the pending HET interrupt; the corresponding flag (in the HETFLG) is also cleared. Emulation mode (read): Read of these bits determine the pending HET interrupt, but the corresponding flag is not cleared.

6.7 Exception Control Register 1 (HETEXC1)

Figure 42 and Table 17 describe the HETEXC1.

Figure 42. Exception Control Register (HETEXC1)



RW = Read/Write, *-n* = Value after reset

Table 17. Exception Control Register (HETEXC1) Field Descriptions

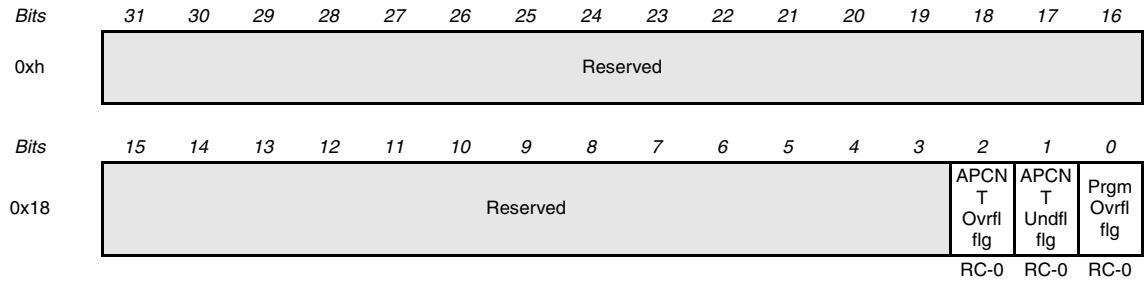
Bit	Name	Value	Description
31–25	Reserved		
24	APCNT Ovrfl Ena		APCNT Overflow Enable
			Any operation mode (read/write):
		0	APCNT overflow exception is not enabled.
		1	Enables the APCNT overflow exception.
23–17	Reserved		
16	APCNT Undrfl Ena		APCNT Underflow Enable
			Any operation mode (read/write):
		0	APCNT underflow exception is not enabled.
		1	APCNT underflow exception is enabled.
15–9	Reserved		
8	Prgm Ovrfl Ena		Program Overflow Enable
			Any operation mode (read/write):
		0	The program overflow exception is not enabled.
		1	The program overflow exception is enabled.

7–3	Reserved		
2–0	Exception Priority Level bits Register bits		Used to select the priority of any of the three potential exception sources. Any operation mode (read/write):
		0	Exception priority level 2
		1	Exception priority level 1

6.8 Exception Control Register 2 (HETEXC2)

Figure 43 and Table 18 describe the HETEXC2.

Figure 43. Exception Control Register 2 (HETEXC2)



RC = Read/Clear, *-n* = Value after reset

Table 18. Exception Control Register 2 (HETEXC2) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved		
2	APCNT Ovrfl flg		APCNT Overflow Flag
			Any operation mode (read):
		0	Exception has occurred since the flag was cleared.
		1	Exception has not occurred since the flag was cleared.
			Any operation mode (write):
		0	No effect.
		1	The bit is cleared.
1	APCNT Undrfl flg		APCNT Underflow Flag
			Any operation mode (read):
		0	Exception has occurred since the flag was cleared.
		1	Exception has not occurred since the flag was cleared.
			Any operation mode (write):
		0	No effect.
		1	Clears the bit.

0	Prgrm Overfl flg		Program Overflow Flag
			Any operation mode (read):
		0	Exception has occurred since the flag was cleared.
		1	Exception has not occurred since the flag was cleared.
			Any operation mode (write):
		0	No effect.
		1	The bit is cleared.

6.9 Interrupt Priority Register (HETPRY)

Figure 44 and Table 19 describe the HETPRY register.

Figure 44. Interrupt Priority Register (HETPRY)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xh	HETPRY[31:16]															
	RWP-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	HETPRY[15:0															
	RWP-0															

RWP = Read in all modes, write in privilege mode only, *-n* = Value after reset

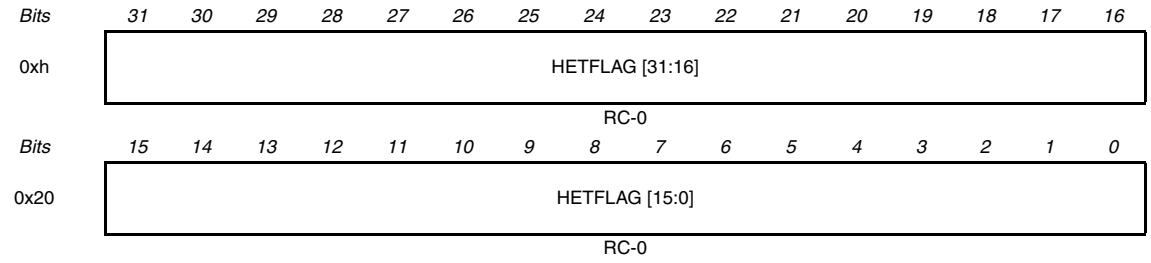
Table 19. Offset Index Priority Level 2 Register (HETOFF2) Field Descriptions

Bit	Name	Value	Description
31–0	HETPRY[31:0]		HET Priority Level bits. Used to select the priority of any of the 32 potential interrupt sources coming from the instructions. Any operation mode (read/write):
		0	Indicates software priority level 2.
		1	Indicates software priority level 1.

6.10 HET Interrupt Flag Register (HETFLG)

Figure 45 and Table 20 describe the HETFLG.

Figure 45. HET Interrupt Flag Register (HETFLG)



RC = Read/Clear, *-n* = Value after reset

Table 20. HET Interrupt Flag Register (HETFLG) Field Descriptions

Bit	Name	Value	Description
31–0	HET-FLAG[31:0]		Interrupt flag register bits These bits are set when an interrupt condition has occurred and when the interrupt enable bit (in the instruction) is set. The flag is also set by the five LSBs of the instruction address that generated the interrupt. To clear the flag, these bits must be set to 1 or the reading of the corresponding HETOFFx register will automatically clear the flag. Any operation mode (read):
		0	No HET instruction with an interrupt has been reached since the flag was last cleared.
		1	A HET instruction with an interrupt has been reached since the flag was last cleared.
			Any operation mode (write):
		0	No effect.
		1	The bit is cleared.

6.11 HR Share Control Register (HETHRSH)

Figure 46 and Table 21 describe the HETHRSH register.

Figure 46. HR Share Control Register (HETHRSH)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xh	Reserved															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	Reserved				HR Share 23/22	HR Share 21/20	HR Share 19/18	HR Share 17/16	HR Share 15/14	HR Share 13/12	HR Share 11/10	HR Share 9/8	HR Share 7/6	HR Share 5/4	HR Share 3/2	HR Share 1/0
					RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

RW = Read/Write, *-n* = Value after reset

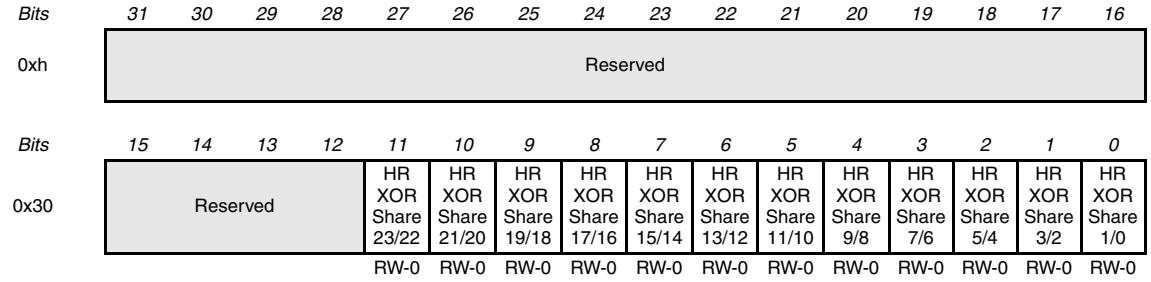
Table 21. HR Share Control Register (HETHRSH) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved		
11–0	HR Share[23:0]		HR Share Bits. Enable the share of the same pin for two HR structures. For example, if bit HR share 1/0 is set, the pin HET0 will then be connected to both HR structures 0 and 1. Any operation mode (read/write):
		0	Pins are not shared.
		1	Pins are shared. If HR share bits are used, pins not connected to HR structures can be accessed as general inputs/outputs

6.12 HR XOR-share Control Register (HETXOR)

Figure 47 and Table 22 describe the HETXOR register.

Figure 47. HR XOR-Share Control Register (HETXOR)



RW = Read/Write, *-n* = Value after reset

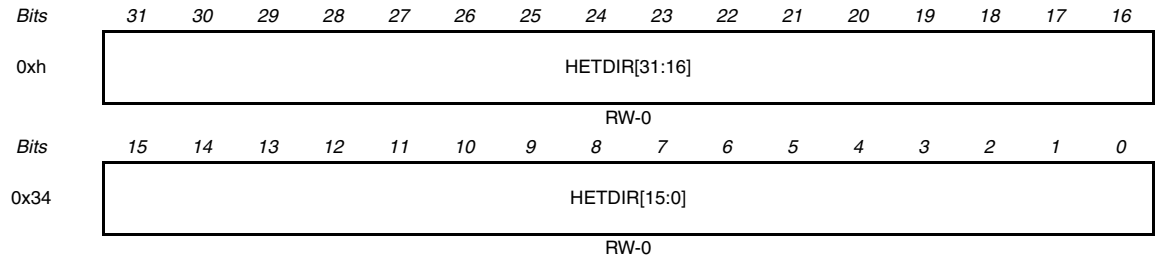
Table 22. HR XOR-Share Control Register (HETXOR) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved		
11–0	HR XOR-Share[23:0]		HR XOR-Share Bits. Enable the XOR-share of the same pin for two HR structures. For example, if bit HR XOR-share 1/0 is set, the pin HET0 will then be commanded by a logical XOR of both HR structures 0 and 1. Any operation mode (read/write): 0 Pins are not XOR-shared. 1 Pins are XOR-shared. If HR share bits are used, pins not connected to HR structures can be accessed as general inputs/outputs **Note: The HETXOR register is only available for certain TMS470 derivatives. Refer to the device specific data sheet.**

6.13 HET Direction Register (HETDIR)

Figure 48 describes the HETDIR register.

Figure 48. HET Direction Register (HETDIR)



RW = Read/Write, -*n* = Value after reset

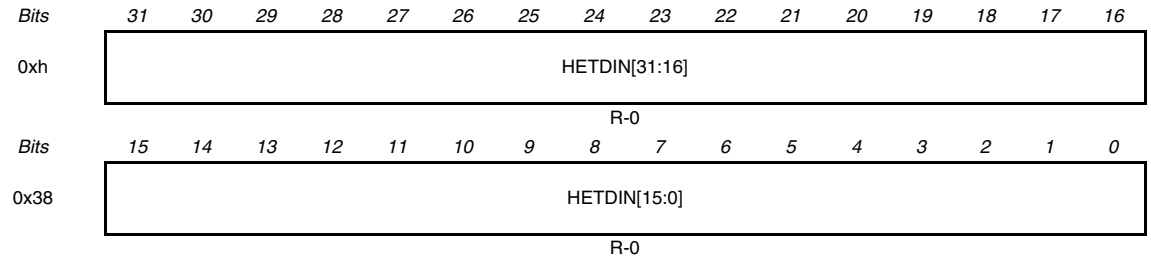
Table 23. HET Direction Register (HETDIR) Field Descriptions

Bit	Name	Value	Description
31–0	HET DIR[31:0]		Input/Output Direction bits. Used to select the direction of the HET pins. Any operation mode (read/write):
		0	HET pin configured as input.
		1	HET pin configured as output.

6.14 HET Data Input Register (HETDIN)

Figure 49 and Figure 24 describe the HETDIN register.

Figure 49. HET Data Input Register (HETDIN)



R = Read, -*n* = Value after reset

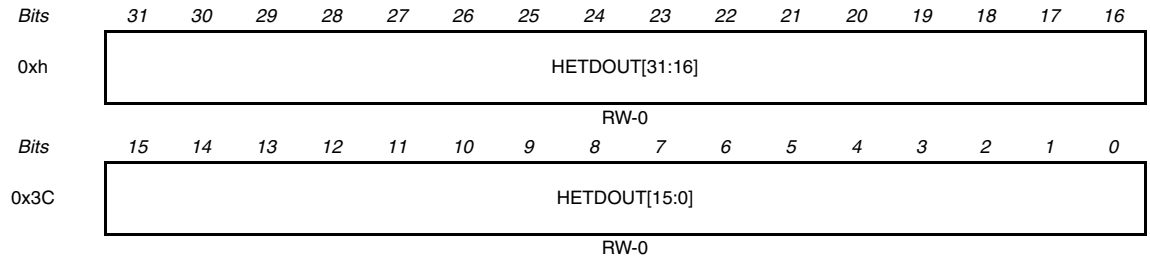
Table 24. HET Data Input Register (HETDIN) Field Descriptions

Bit	Name	Value	Description
31–0	HET DIN[31:0]		HET Data Input Register bits Any operation mode (read):
		0	Logic low.
		1	Logic high.
			Any operation mode (write): Writes to these bits have no effect.

6.15 HET Data Output Register (R-Write) (HETDOUT)

Figure 50 and Table 25 describe the HETDOUT register.

Figure 50. HET Data Output Register (R-Write) (HETDOUT)



RW = Read/Write, *-n* = Value after reset

Table 25. HET Data Output Register (R-Write) (HETDOUT) Field Descriptions

Bit	Name	Value	Description
31–0	HET DOU[31:0]		HET Data Output Register bits Used to set/reset the HET pins. A read to this register gives the value of the corresponding HETDOUT register. Any operation mode (read/write):
		0	Logic low.
		1	Logic high.

6.16 HET Data Set Register (R-Set) (HETDSET)

Figure 51 and Table 26 describe the HETDSET register.

Figure 51. Het Data Set Register (R-Set) (HETDSET)

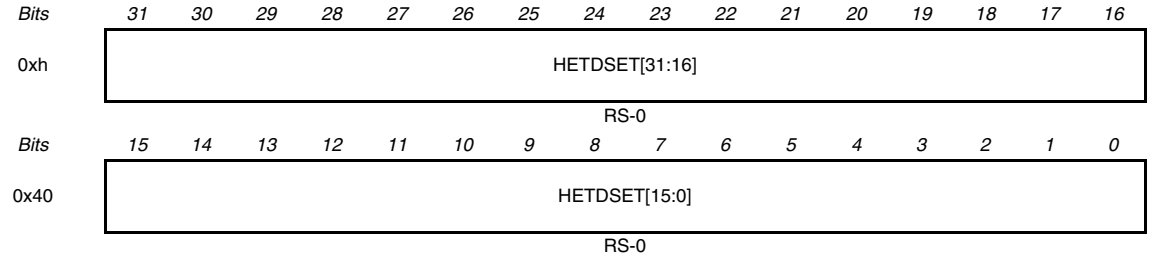


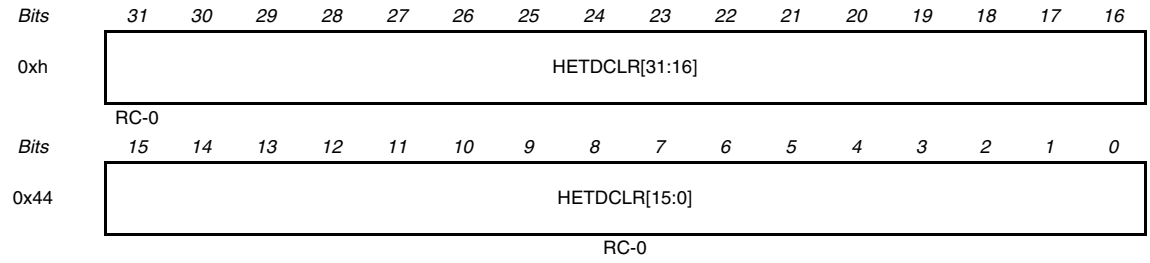
Table 26. Het Data Set Register (R-Set) (HETDSET) Field Descriptions

Bit	Name	Value	Description
31–0	HET DSET[31:0]		HET Data Set Register bits Used to set the HET pins to the HIGH level. A read to this register gives the value of the corresponding HETDSET register. Any operation mode (read): HETDSET reflects the contents of HETDOUT.
		0	Logic low.
		1	Logic high.
			Any operation mode (write):
		0	Leaves the corresponding bit in HETDOUT unchanged.
		1	Sets the corresponding bit in HETDOUT to 1.

6.17 HET Data Clear Register (R-Clear) (HETDCLR)

Figure 52 and Table 27 describe the HETDCLR register.

Figure 52. HET Data Clear Register (R-Clear) (HETDCLR)



RC = Read/Clear, -*n* = Value after reset

Table 27. HET Data Clear Register (R-Clear) (HETDCLR) Field Descriptions

Bit	Name	Value	Description
31–0	HET DCLR[31:0]		HET Data Clear Register bits Used to clear the HET pins to 0. A read to this register gives the value of the corresponding HETDCLR register. Any operation mode (read): HETDCLR reflects the contents of HETDOUT.
		0	Logic low.
		1	Logic high.
			Any operation mode (write):
		0	Leaves the corresponding bit in HETDOUT unchanged.
		1	Sets the corresponding bit in HETDOUT to 1.

7 Instruction Set

The following sections discuss the HET instruction set.

7.1 Instruction Summary

Table 28 presents a list of the instructions in the HET instruction set. The pages following describe each instruction in detail.

Table 28. Instruction Summary

Abbreviation	Instruction Name	Opcode	Sub-Opcode	Cycles	Pages
ACMP	Angle Compare	Ch	-	1	5-83
ACNT	Angle Count	9h	-	2	5-86
ADCNST	Add Constant	5h	-	2	5-90
ADM32	Add Move 32	4h	C5=1	1 or 2	5-92
APCNT	Angle Period Count	Eh	-	1 or 2	5-97
BR	Branch	Dh	-	1	5-101
CNT	Count	6h	-	1 or 2	5-103
DADM64	Data Add Move 64	2h	-	2	5-107
DJZ	Decrement and Jump if -zero	Ah	P[6-5]=10	1	5-110
ECMP	Equality Compare	0h	C[6-5]=00	1	5-112
ECNT	Event Count	Ah	P[6-5]=01	1	5-115
MCMP	Magnitude Compare	0h	C[6-5]=1X	1	5-118
MOV32	Move 32	4h	C5=0	1 or 2	5-122
MOV64	Move 64	1h	-	1	5-126
PCNT	Period/Pulse Count	7h	-	1	5-129
PWCNT	Pulse Width Count	Ah	P[6-5]=11	1	5-134
RADM64	Register Add Move 64	3h	-	1	5-137
SCMP	Sequence Compare	0h	C[6-5]=01	1	5-140
SCNT	Step Count	Ah	P[6-5]=00	3	5-144
SHFT	Shift	Fh	-	1	5-148
WCAP	Software Capture Word	Bh	-	1	5-152

Table 29. FLAGS Generated by Instruction

	Flag Name	Set / Reset by	Used by
Z	Z flag	CNT, APCNT, PCNT, ACNT, SCNT, SHFT	ACMP, ECMP, SCMP, MCMP, ACNT, BR, SHFT
X	X flag	ACMP	SCMP
SWF 0-1	Step width flags	SCNT	ACNT
NAF	New Angle flag	ACNT	NAF_global
NAF_global	New Angle flag (global)	*HWAG* or NAF	CNT, ECNT, BR
ACF	Acceleration flag	ACNT	SCNT, ACNT
DCF	Deceleration flag	ACNT	SCNT, ACNT
GPF	Gap flag	ACNT	APCNT, ACNT

The instructions capable of generating SW interrupts are listed in Table 30.

Table 30. Interrupt Capable Instructions

Interrupt Capable Instructions	Non Interrupt Capable Instructions
ACMP	ADCNST
ECMP	ADM32
SCMP	DADM32
MCMP	MOV32
CNT	MOV64
ECNT	RADM64
ACNT	SCNT
APCNT	
PWCNT	
PCNT	
DJZ	
WCAP	
SHFT	
BR	

7.2 Abbreviations

Abbreviations marked with a star (*) are available only on specific instructions.

BRK Defines the software breakpoint for the device software debugger.

Default: OFF.

Location: Program field [20]

Next Defines the program address of the next instruction in the program flow. This value may be a label or an 8-bit unsigned integer.

Default: Current instruction + 1.

Location: Program field [19:12]

Remote* Determines the 8-bit address of the remote address for the instruction.

Default: Current instruction + 1.

Location: Program field [7:0]

Control Determines whether the immediate data field [24:0] is cleared when it is read. When the bit is not set, reads do not clear the immediate data field.

Default: OFF.

Location: Control field [21]

En_pin_action* Determines whether the selected pin is ON so that the action occurs on the chosen pin.

Default: OFF.

Location: Control field [20]

Cond_addr* Conditional address: (Optional) Defines the address of the next instruction when the condition occurs.

Default: Current address + 1.

Location: Control field [19:12]

Pin* Pin Select: Selects the pin on which the action occurs.
Enter the pin number.[A]
Default: pin 0.
Location: Control field [11:7] except PCNT,

U Reading a bit marked with U will return an indeterminate value.

[A]The format CC{pin number} is also supported.

7.3 Encoding Formats and Bits

Table 31 through Table 36 provide information about encoding formats and bits.

Table 31. PIN Encoding Format

MSB				LSB	Pin Select
0	0	0	0	0	Selects HET0
0	0	0	0	1	Selects HET1
(each pin may be selected by writing its number in binary)					
1	1	1	1	0	Selects HET30
1	1	1	1	1	Selects HET31

Reg* Register Select: Selects the register for data comparison and storage.
Default: No register (None).
Location: Control field [2:1] except CNT

Table 32. Register Bit Field Encoding Format

Register[A]	C[2]	C[1]
A	0	0
B	0	1
T	1	0
None	1	1

[A]The register bits field could be placed either in the Program field (CNT), or in the control field (all others' instructions use register field).

Action* (2 Action Option) Either sets or clears the pin

Default: Clear.

Location: Control Field [4]

Table 33. PIN Action Bit Field (2 options)

Action	C[4]
Clear	0
Set	1

Action* (4 Action Option) Either sets, clears, pulse high or pulse low on the pin. Pulse high occurs when the pin is set on the compare and toggles at the overflow.

Default: Clear.

Location: Control Field [4:3]

Table 34. PIN Action Bit Field (4 options)

Action	C[4][A]	C[3][B]
Clear	0	0
Set	1	0
Pulse Low	0	1
Pulse High	1	1

[A]Bit C[4] is also called enable pin action.
[B]C[3] is also called opposite pin action.

hr_lr* Specifies HIGH/LOW data resolution. If the hr_lr field is HIGH, the instruction implements the hr_data field (when the action is carried out on an HR pin). If the hr_lr field is LOW, the hr_data field is ignored.

Default: HIGH.
Location: Program Field [7]

Table 35. High-Low Resolution Bit Field

hr_lr	Prog. field [7]
LOW	1
HIGH	0

prv* Specifies the initial value defining the previous pin-level bit for the first edge detect performed by the instruction. The edge detect is performed by comparing the current pin value to the value stored in the previous pin-level bit. A value of ON sets the previous pin-level bit to 1. A value of OFF sets the initial value of the previous (prv) bit to 0. After the initial comparison, the value of the prv bit is set or reset by the system.

Default: OFF.

Location: Control Field [20]

cntl_val* Available for the DADM64, MOV64, and RADM64, this bits field allows you to specify the replacement value for the remote control field.

comp_mode* Specifies the compare mode. This field is used with the 64-bit move instructions. The field ensures that the sub-opcodes are moved correctly.

Default: ECMP.

Location: Control Field [6:5]

Table 36. Comp_mode Bit Field

comp_mode	C[6]	C[5]
ECMP	0	0
SCMP	0	
MCMP	1	
ACMP	1	

7.4 Instruction Description

The following sections provide information for individual instructions.

7.4.1 ACMP (Angle Compare)

Syntax

ACMP {
[brk={OFF | ON}]
[next={*label* | *8-bit unsigned integer***}]**
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={*label* | *8-bit unsigned integer***}]**
pin={pin number**}**
[action={CLEAR | SET}]
reg={A | B | T | NONE}
[irq ={OFF | ON}]
data={*20-bit unsigned integer***}**
}

Opcode Ch, [P11:P8]

Table 37. ACMP Program Field (P31:P0)

31–22	21	20	19–12	11–8	7–0
0	Res	BRK	Next program address	Opcode	Reserved
10	1-U	1	8	4	8

Table 38. ACMP Control Field (C31:C0)

31–22	21	20	19–12	11–7	6	5	4	3	2–1	0
0	Con-trol	En. pin ac-tion	Conditional ad-dress	Pin select	Res.	Cout prv	Pin ac-tion	Res.	Register select	Int. ena.
10	1	1	8	5	1	1	1	1	1	1

Table 39. ACMP Data Field (D31:D0)

31–25	24–5	4–0
0	Data	Reserved
7	20	5

Cycles One

Register modified Selected register (A, B, or T)

Description

The purpose of the comparison is to assert pin action when the angle compare value lies between the old counter value and the new counter value (held in the selected register). Since the angle increment varies from one loop resolution clock to another, an exact equality test cannot be applied. Instead, the following inequality is used to determine the occurrence of a match:

Old counter value < Angle compare value ≤ New counter value

This is done by performing following comparisons:

Selected register value minus angle increment < angle compare value
Angle compare value ≤ selected register value

register Register B is recommended for typical applications with ACMP.

irq Specifies whether or not an interrupt is generated. Specifying ON generates an interrupt when the edge state is satisfied and the gap flag is set. Specifying OFF prevents an interrupt from being generated.

Default: OFF.

data Specifies the 20-bit angle compare value.

Execution

```
X = 0;

If (Data field value ≤ Selected register value)
       Cout = 0;

else
       Cout = 1;

If (Z == 0 AND (Selected register value - Angle Inc. < Data
field value) AND Cout == 0)

- or -
      (Z == 1 AND (Cout_prv == 1 OR Cout == 0)))
       {
       X = 1;
```

```
      If (Enable Pin Action == 1)
          Selected Pin = Pin Action AT next loop resolution
          clock;
      If (Interrupt Enable == 1)
          SW interrupt flag = 1;
      Jump to Conditional Address;
      }
else
      Jump to Next Program Address;
Cout_prv = Cout (always executed)
```

Note: Carry-Out Signal (Cout)

Cout is the carry-out signal of the adder.Even if it is not a flag, it is valid all along ACMP instruction execution.

Angle inc. = NAF_global or hardware angle generator 4-bit input.

7.4.2 ACNT (Angle Count)

Syntax

ACNT {
[brk={OFF | ON}]
[next={*label* | *8-bit unsigned integer*}]
edge={RISING | FALLING}
[irq={OFF | ON}]
[control={OFF | ON}]
[prv={OFF | ON}]
gapend=20-bit unsigned integer
[data=20-bit unsigned integer]
}

Opcode **9h, [P11:P8]**

Table 40. ACNT Program Field (P31:P0)

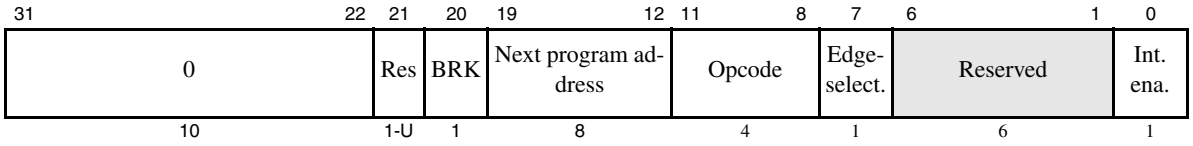


Table 41. ACNT Control Field (C31:C0)

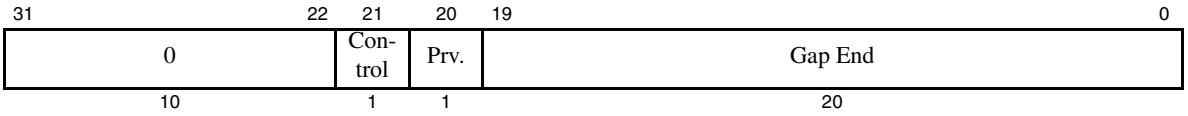
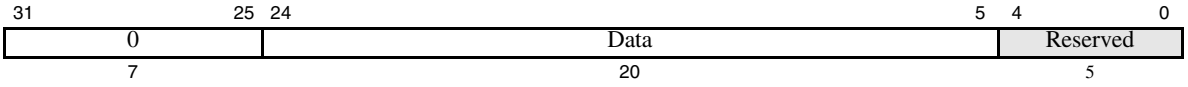


Table 42. ACNT Data Field (D31:D0)



Cycles Two, as follows:

- ❏ First cycle: Angle increment condition and gap end comparison.
- ❏ Second cycle: Gap start comparison.

Register modified Register B (angle value)

Description

This instruction defines a specialized virtual timer used after SCNT and APCNT to generate an angle-referenced time base that is synchronized to an external signal (that is, a toothed wheel signal). ACNT uses pin HET24 exclusively**.** The edge select must be the same as the HET24 edge which was selected in the previous APCNT.

ACNT refers to the same step width selection that the previous SCNT saved in flags SWF0 and SWF1 (see information on SCNT).

ACNT detects period variations of the external signal measured by APCNT and compensates related count errors.

A period increase is flagged in the deceleration flag (DCF). A period decrease is flagged in the acceleration flag (ACF). If no variation is detected, ACNT increments the counter value each time SCNT reaches its target.

If acceleration is detected, ACNT increments the counter value on each timer resolution. If deceleration is detected ACNT does not increment and is thus saturated.

ACNT also specifies the gap end angle value defining the end value of a gap range in ACNT where period measurements in APCNT are temporarily stopped to mask singularities in the external signal. ACNT uses register A containing gap start and register B to store the counter value.

Edge Specifies the edge for the input capture pin (HET[24]).

Action	P7	Edge Select
Rising	1	Detects a rising edge of HET24
Falling	0	Detects a falling edge of HET24

Irq ON generates an interrupt when the edge state is satisfied and the gap flag is set. OFF prevents an interrupt from being generated.

Default: OFF.

gapend Defines the 20-bit end value of a gap range. The start value is defined in the SCNT instruction.

GAPEND = (Step Value * (# of teeth on the toothed wheel + # of missing teeth)) -1

data Specifies the 20-bit initial count value for the data field.

Default: 0.

> **Note: Target Edge Field**
>
> The target edge field represents the three LSBs of data field register in case of step width = 8, four LSBs for step width = 16, five LSBs for step width = 32 and six LSBs for step width = 64.

Execution

```
Increment Condition: ((Z = 1 AND DCF = 0) OR ACF = 1)
Pin Edge Condition: Specified edge detected on HET24
Target Edge Condition: (Target Edge field in data field =
0) AND (Angle Increment condition is true) AND (GPF = 0)
If (Angle Increment Condition) is false
      {
      NAF_global = 0;
      Register B = Data field register;
      }
else
      {
      NAF_global = 1;
      If (Counter value != GapEnd - 1)
           {
           Register B = Data field register + 1;
           Data Field Register = Counter value + 1;
           }
      else
           {
           Register B = 0;
           Data Field Register = 0;
           If (ACF == 0)
             DCF = 1;
           }
      }
Z = 0;
If (Data field register == GapStart)
      {
```

```
        GPF = 1;
        If (Target Edge condition is true)
            {
            ACF = 0;
            If ((specified edge is not detected on pin HET24)
            AND (data field register != 0) AND (ACF == 0))
              DCF = 1;
            }
        If (specified edge is detected on pin HET24)
            {
            DCF = 0;
            If ((data field register != 0) AND (DCF == 0))
              ACF = 1;
            If (GPF == 1)
              {
              GPF = 0;
              Z = 1;
              If (Interrupt Enable == 1)
                SW interrupt flag = 1;
              }
            }
        }
Prv bit = Current HET24 pin level;
```

7.4.3 ADCNST (Add Constant)

Syntax

ADCNST {
[**brk=**{**OFF** | **ON**}]
[**next=**{*label* | *8-bit unsigned integer*}]
[control={OFF | ON}]
remote={*label* | *8-bit unsigned integer*}
min_off=*20-bit unsigned integer*
data=*20-bit unsigned integer*
[**hr_data=***5-bit unsigned integer*]
}

Opcode 5h, [P11:P8]

Table 43. ADCNST Program Field (P31:P0)

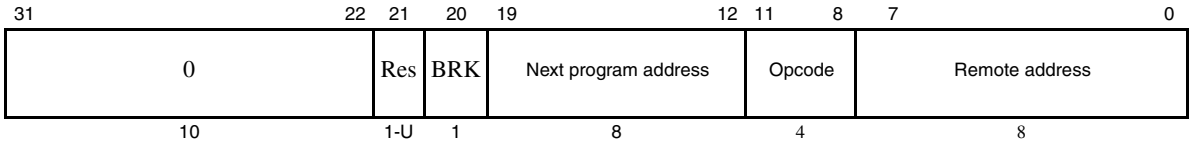


Table 44. ADCNST Control Field (C31:C0)

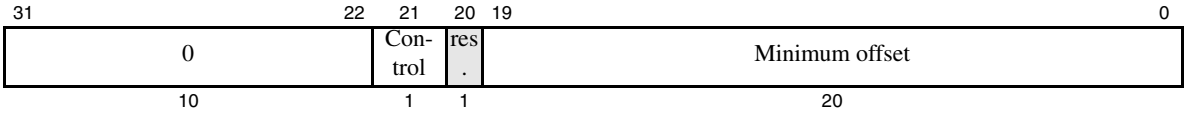
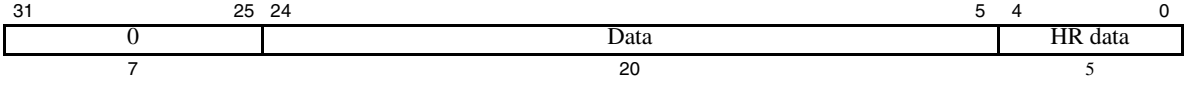


Table 45. ADCNST Data Field (D31:D0)



Cycles Two

Register modified Register T (implicitly)

Description ADCNST is an extension of ADMOV32. ADCNST first checks whether the data field value at the remote address is zero; it then performs different adds and moves on the result. ADCNST is typically used to extend the counter value of PWCNT.

min_off A 20-bit constant value that is added to the data field value if the remote data field is null.

data A 20-bit value that is always added to the remote data field.

Default: 0.

hr_data Five least significant bits of the data addition to the remote data field.

Default: 0.

Table 53 and Table 54 illustrate the behavior of ADCNST if the remote data field is or is not zero.

Figure 53. ADCNST Operation If Remote Data Field[24:5] Is Not Zero

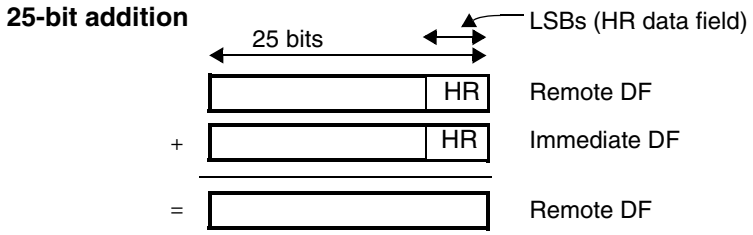
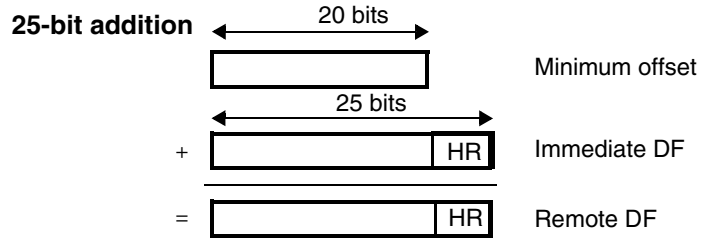


Figure 54. ADCNST Operation if Remote Data Field [24:5] Is Zero



Execution

```
If (Remote Data Field Value [24:5] != 0)
      Remote Data Field = Immediate Data Field + Remote Data
      Field;
else
      Remote Data Field = Immediate Data Field + min. off-
      set(bits C19:C0);
Jump to Next Program Address;
```

7.4.4 ADM32 (Add Move 32)

Syntax

ADM32 {
[brk={OFF | ON}]
[next={*label* **|** *8-bit unsigned integer***}]**
remote={*label* **|** *8-bit unsigned integer***}**
[control={OFF | ON}]
[init={OFF | ON}]
type={IM&REGTOREG | REM&REGTOREG |
IM&REMTOREG | IM&REGTOREM}
reg={A | B | T | NONE}
data=*20-bit unsigned integer*
[hr_data=5*-bit unsigned integer***]**
}

Opcode 4h, [P11:P8];

Sub-Opcode *1h, [C5]*

Table 46. ADM32 Program Field (P31:P0)

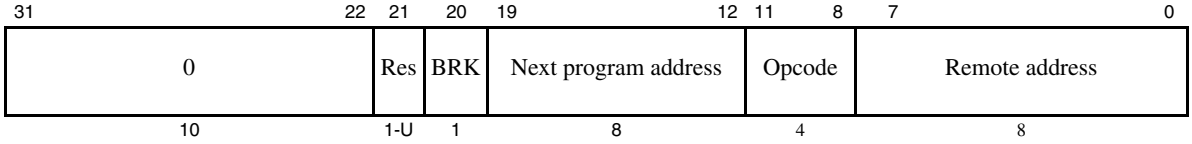


Table 47. ADM32 Control Field (C31:C0)

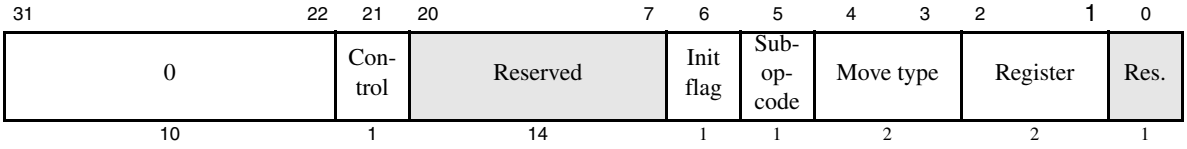
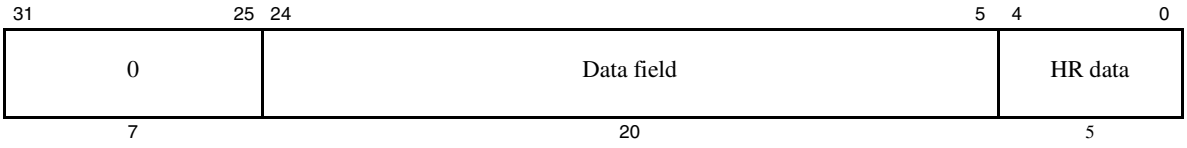


Table 48. ADM32 Data Field (D31:D0)



Cycles One or two cycles (see Table 49)

Register modified Selected register (A, B, or T)

Description This instruction modifies the selected ALU register or data field values at the remote address depending on the move type. The modified value results from adding the immediate or remote data field to the ALU register or the remote data field, depending on the move type. Table description shows the C2 and C1 bit encoding for determining which register is selected.

init (Optional) Determines whether or not system flags are initialized. A value of ON reinitializes the following system flags to these states:

- Acceleration flag (ACF) = 0
- Deceleration flag (DCF) = 1
- Gap flag (GPF) = 0
- New angle flag (NAF) = 0

A value of OFF results in no change to the system flags.

Default: OFF

type Specifies the move type to be executed.

Table 49. Move Types for ADM32

Type	C4	C3	Add			Destination(s)	Cycles
IM&REGTOREG	0	0	Imm. data field	+	Reg. A, B, or T	Register A, B, or T	1
REM&REGTOREG	0	1	Remote data field	+	Reg. A, B, or T	Register A, B, or T	2
IM&REMTOREG	1	0	Imm. data field	+	Remote data field	Register A, B, or T	2
IM&REGTOREM	1	1	Imm. data field	+	Reg. A, B, or T	Remote data field	1

If selected register is T, the operation is a 25-bit Addition/move. If A or B register is selected, it is limited to 20-bit operation since A and B only support 20-bit.

data Specifies the 20-bit integer value for the immediate data field.

hr_data Specifies the 5 least significant bits of the immediate data field.

Default: 0.

Execution

```
switch (C4:C3)
      {
      case 00:
          Selected register = Selected register + Immediate
          Data Field;
      case 01:
          Selected register = Selected register + Remote
          Data Field;
      case 10:
          Selected register = Immediate Data Field + Remote
          Data Field;
      case 11:
          Remote Data Field = Selected register + Immediate
          Data Field;
      }
If (Init Flag == 1)
      {
      ACF = 0;
      DCF = 1;
      GPF = 0;
      NAF = 0;
      }
else
      All flags remain unchanged;
Jump to Next Program Address;
```

Register modified Selected register (A, B or T)

Figure 55 through Figure 58 illustrate the ADM32 operation for various cases.

Figure 55. ADM32 Add and Move Operation for IM&REGTOREM (Case 00)

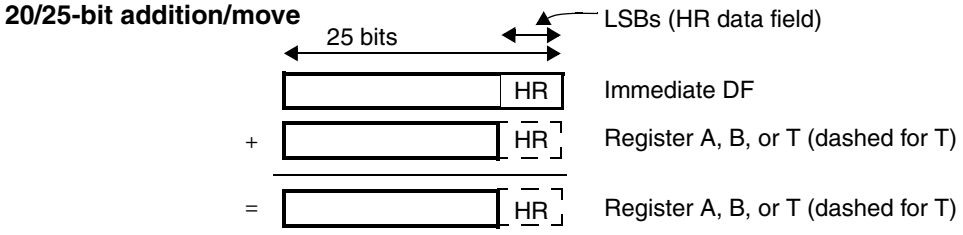


Figure 56. ADM32 Add and Move Operation for REM&REGTOREG (Case 01)

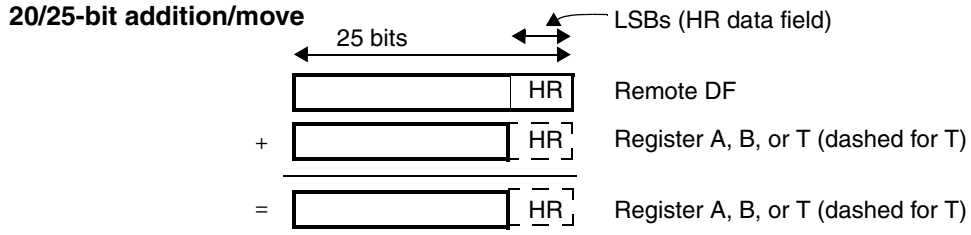


Figure 57. ADM32 Add and Move Operation for IM&REMTOREG (Case 10)

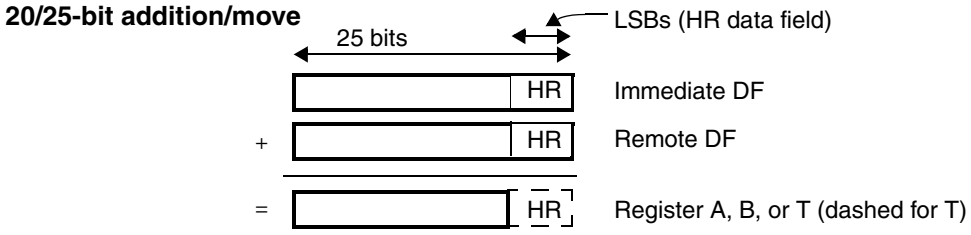
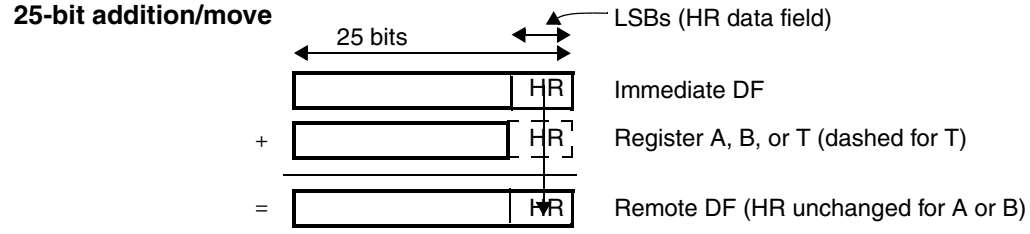


Figure 58. ADM32 Add and Move Operation for IM&REGTOREM (Case 11)



7.4.5 APCNT (Angle Period Count)

Syntax

APCNT {
[brk={OFF| ON}]
[next=*{label | 8-bit unsigned integer}]*
[irq={OFF | ON}]
type={FALL2FALL | RISE2RISE}
[control={OFF | ON}]
prv={OFF | ON}
period=*20-bit unsigned integer*
[data=*20-bit unsigned integer]*
}

Opcode Eh, [P11:P8]

Table 50. APCNT Program Field (P31:P0)

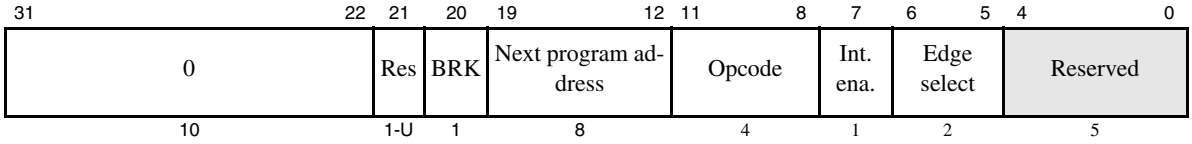


Table 51. APCNT Control Field (C31:C0)

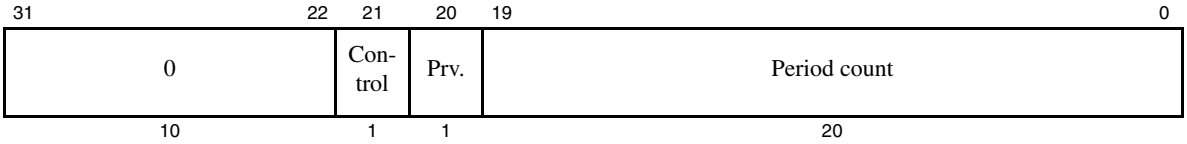
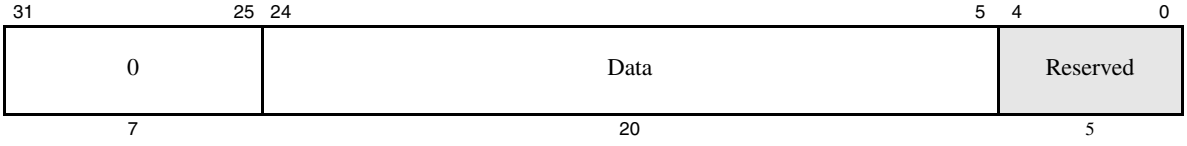


Table 52. APCNT Data Field (D31:D0)



Cycles One or two cycles
One cycle (normal operation) two cycles (edge detected)

- ❏ Cycle 1: edge detected (normal operation)
- ❏ Cycle 2: edge detected and GPF = 1 and underflow condition is true

Register modified Register A and T (implicitly)

Description This instruction is used before SCNT and ACNT to generate an angle-referenced time base synchronized to an external signal (that is, a toothed wheel signal). It is assumed that the pin and edge selections are the same for APCNT and ACNT.

APCNT is restricted to pin HET24. The toothed wheel must then be connected to pin HET24.

APCNT uses the gap flag (GPF) defined by ACNT to start or stop captures in the period count field [C19:C0]. When GPF=1, the previous period value is held in the control field and in register T. When GPF= 0, the current period value is captured in the control field and in register T.

APCNT uses the step width flags (SWF0 and SWF1) defined by SCNT to detect period durations shorter than one step, and then disables capture.

The edge select encoding is shown in Table 53.

irq ON generates an interrupt when the edge state is satisfied. OFF prevents an interrupt from being generated.

Default: OFF.

type Specifies the edge type that triggers the instruction.

Default: Fall2Fall.

Table 53. Edge Select Encoding for APCNT

type	P6	P5	Selected Condition
Fall2Fall	1	0	Falling edge
Rise2Rise	1	1	Rising edge

period Contains the 20-bit count value from the previous APCNT period.

data 20-bit value serving as a counter.

Default: 0.

Execution

```
Z = 0;
If (Data field register != FFFFFh)
      {
      Register A = Data field register + 1;
      Data field register = Data field register + 1;
      }
elseIf (specified edge not detected on HET24)
      {
      Register A = FFFFFh;
      APCNT Ovflw flag = 1;
      }
If (specified edge detected on HET24)
      {
      Z = 1;
      If (Data field register == FFFFFh)
          {
          Register A = FFFFFh;
          Register T = FFFFFh;
          Period count = FFFFFh;
          }
      elseIf (GPF == 0 AND Data Field register ≥ Step width)
          {
          Register A = Data field register + 1;
          Register T = Register A;
          Period count = Register T;
          If (Interrupt Enable == 1)
            SW interrupt flag = 1;
          }
      If (GPF == 1)
          Register T = Period count;
```

```
If (Data Field register < Step width)
                          {
                          Register T = Period count;
                          APCNT Undflw flag = 1;
                          Data field register = 00000h;
                          }
                    }
              else
                    Register T = Period count;
              Prv bit = Current HET24 pin level;
              Jump to Next Program Address;
```

7.4.6 BR (Branch)

Syntax

BR {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
[control={OFF | ON}]
[prv={OFF | ON}]
cond_addr=*{label | 8-bit unsigned integer}*
[pin= {pin number}]
event={NOCOND | FALL | RISE | BOTH | ZERO | NAF | LOW | HIGH}
[irq={OFF | ON}]
}

Opcode Dh, [P11:P8]

Table 54. BR Program Field (P31:P0)

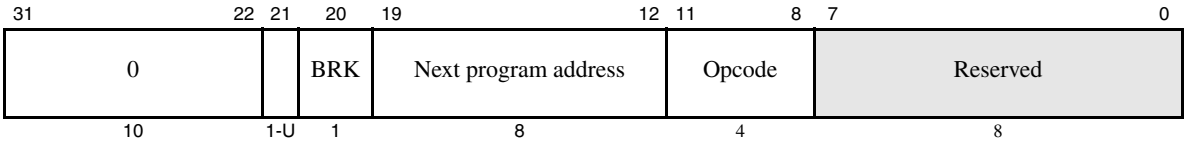


Table 55. BR Control Field (C31:C0)

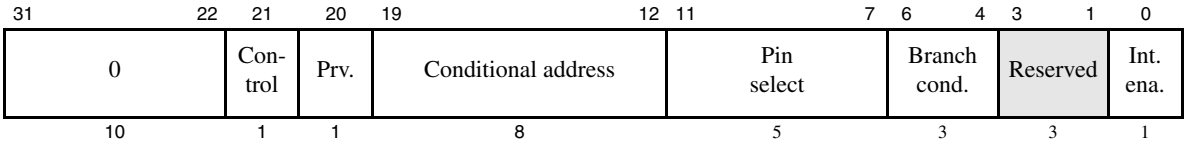
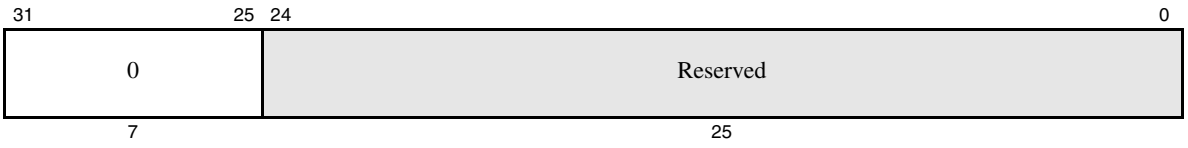


Table 56. BR Data Field (D31:D0)



Cycles One

Register modified None

Description This instruction executes a jump to the conditional address [C19:C12] on a pin or a flag condition, and can be used with all pins.

An edge is detected by comparing the current pin level sampled at loop start to the previous pin level stored in the Prv. (Previous) bit [C20].

event Specifies the event that triggers a jump to the indexed program address.

Default: FALL

Table 57 provides the branch condition encoding.

Table 57. Branch Condition Encoding for BR

event	C6	C5	C4	Branch Condition
NOCOND	0	0	0	Always
FALL	0	0	1	On falling edge on the selected pin
RISE	0	1	0	On rising edge on selected pin
BOTH	0	1	1	On rising or falling edge on selected pin
ZERO	1	0	0	If Zero flag is set
NAF	1	0	1	If NAF_global flag is set
LOW	1	1	0	On LOW level on selected pin
HIGH	1	1	1	On HIGH level on selected pin

irq ON generates an interrupt when the event occurs that triggers the jump. If irq is set to OFF, no interrupt is generated.

Default: OFF.

Execution

```
If (Condition is true)
      {
      If (Interrupt Enable == 1)
          SW interrupt flag = 1;
      Jump to Conditional Address;
      }
else
      Jump to Next Program Address;
Prv. bit = Selected Pin level; (Always executed)
```

7.4.7 CNT (Count)

Syntax

CNT {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
[angle_count={OFF | ON}]
[reg={A | B | T | NONE}]
[irq={OFF | ON}]
[control={OFF | ON}]
max=*20-bit unsigned integer*
[data=*20-bit unsigned integer]*
}

Opcode 6h, [P11:P8]

Table 58. CNT Program Field (P31:P0)

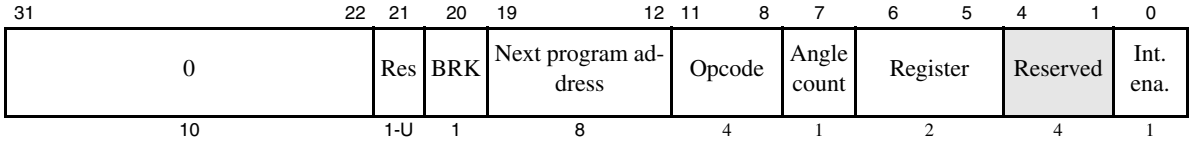


Table 59. CNT Control Field (C31:C0)

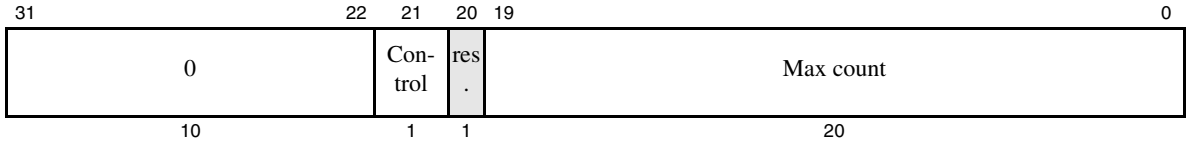
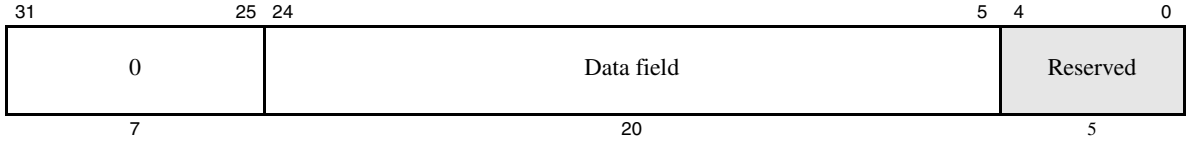


Table 60. CNT Data Field (D31:D0)



Cycles One or two
One cycle (time mode), two cycles (angle mode)

Register modified Selected register (A, B or T)

Description This instruction defines a virtual timer. The counter value stored in the data field [D24:5] is incremented unconditionally on each resolution when in time

mode (angle count bit [P7] = 0). When the count reaches the maximum count specified in the control field, the counter is reset. It takes one cycle in this mode.

In angle mode (angle count bit [P7] = 1), CNT needs data from the hardware angle generator (HWAG) or software angle generator (SWAG) using the logic circuitry shown in *Section 5.5.2, "Interface to a Hardware Angle Generator"*. When in angle count mode, the count instruction is incremented by the angle increment value from the HWAG. This value can be from 0 to 15. For the SWAG, the angle increment value will be 0 or 1. It takes two cycles in this mode.

Table 32, “Register Bit Field Encoding Format,” on page 83 shows the P6 and P5 encoding for selecting the required ALU register.

angle_count Specifies when the counter is incremented. A value of ON causes the counter value to be incremented only if the new angle flag is set (NAF_global = 1). A value of OFF increments the counter each time the assembler encounters the CNT instruction.

Default value for this field is OFF.

irq ON generates an interrupt when the counter overflows to zero. The interrupt is not generated until the data field is reset to zero. If irq is set to OFF, no interrupt is generated.

Default: OFF.

max Specifies the 20-bit integer value that defines the maximum count value allowed in the data field. When the count in the data field is equal to max, the data field is reset to 0 and the Z system flag is set to 1.

data Specifies the 20-bit integer value serving as a counter.

Default: 0.

Execution

```
Z = 0;
If (Angle Count (bit P7 == 1))
      {
      If (NAF_global == 0)
          Jump to Next Program Address;
      else
          {
          If ((Immediate Data Field + Angle Increment) ≥Max
          count)
            {
            Z = 1;
            Selected register = ((Immediate Data Field + An-
            gle Inc.) - Max count);
            Immediate Data Field = ((Immediate Data Field +
            Angle Inc.) - Max count);
            If (Interrupt Enable == 1)
              SW interrupt flag = 1;
            }
          else
            {
            Selected register = Immediate Data Field + Angle
            Increment;
            Immediate Data Field = Immediate Data Field +
            Angle Increment;
            }
          }
      }
```

```
else (Time mode (bit P7 == 0))
      {
      If (Immediate Data Field == Max count)
          {
          Z = 1;
          Selected register = 00000;
          Immediate Data Field = 00000;
          If (Interrupt Enable == 1)
            SW interrupt flag = 1;
          }
      else
          {
          Selected register = Immediate Data Field + 1;
          Immediate Data Field = Immediate Data Field + 1;
          }
      }
Jump to Next Program Address;
```

7.4.8 DADM64 (Data Add Move)

Syntax

DADM64 {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
remote=*{label | 8-bit unsigned integer}*
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr=*{label | 8-bit unsigned integer}]*
[pin=pin number]
comp_mode={ECMP | SCMP | MCMP|ACMP}
[action={CLEAR | SET | PULSELO | PULSEHI}]
reg={A | B | T | NONE}
[irq={OFF | ON}]
data=*20-bit unsigned integer*
[hr_data=5*-bit unsigned integer]*
}

-or-

Syntax

DADM64 {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
remote=*{label | 8-bit unsigned integer}*
[control={OFF | ON}]
cntl_val=*21-bit unsigned integer*
data=*20-bit unsigned integer*
[hr_data=*5-bit unsigned integer]*
}

Opcode 2h, [P11:P8]

Table 61. DADM64 Program Field (P31:P0)

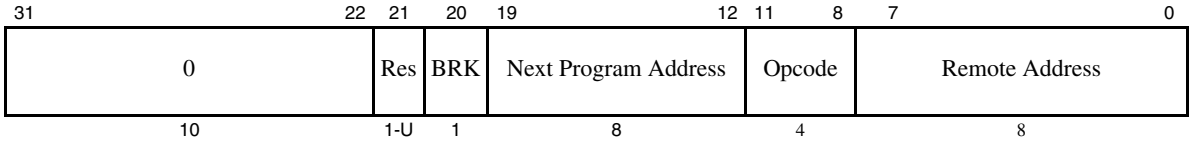


Table 62. DADM64 Control Field (C31:C0)

31 … 22	21	20	19 … 12	11 … 7	6 5	4 … 3	2 … 1	0
0	Con-trol	En-able Pin Action	Conditional Address	Pin Select	Comp. Mode	Action	Register	Int. Ena.
10	1	1	8	5	2	2	2	1

Table 63. DADM64 Data Field (D31:D0)

31 … 25	24 … 5	4 … 0
0	Data Field	High Res. Data Field
7	20	5

Cycles Two

Register modified Register T (implicitly)

Description This instruction modifies the data field and the control field at the remote address. The remote data field value is not just replaced, but is added with the DADM64 data field.

DADM64 has two distinct syntaxes. In the first syntax, bit values may be set by assigning a value to each of the control fields. This syntax is convenient for modifying control fields that are arranged similarly to the format of the DADM64 control field. A second syntax, in which the entire 21-bit control field is specified by the cntl_val field, is convenient when the remote control field is dissimilar to the DADM64 control field. Either syntax may be used, but you must use one or the other but not a combination of syntaxes.

Figure 59 shows the DADM64 add and move operation.

Figure 59. DADM64 Add and Move Operation

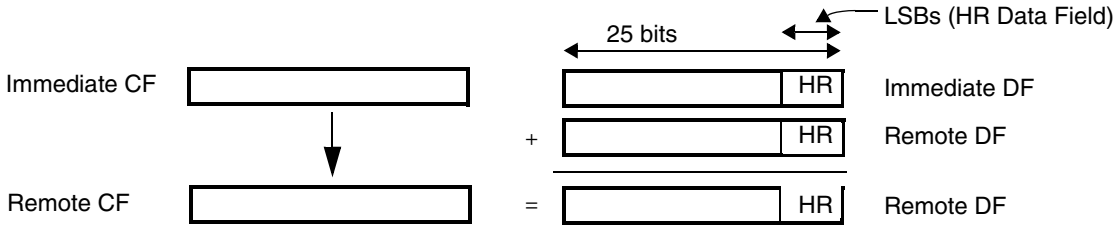


Table 64. DADM64 Control Field Description

Control	maintains the control field for the remote instruction
en_pin_ action	maintains the control field for the remote instruction
cond_ addr	maintains the control field for the remote instruction
pin	maintains the control field for the remote instruction
register	maintains the control field for the remote instruction
action	maintains the control field for the remote instruction
irq	maintains the control field for the remote instruction
data	Specifies the 20-bit initial value for the data field.
hr_data	Five least significant bits of the 25 bit data field. Default: 0
cntl_val	Specifies the 21 least significant bits of the Control field.

Execution

```
Remote Data Field = Remote Data Field + Immediate Data
Field;

Remote Control Field = Immediate Control Field;

Jump to Next Program Address;
```

7.4.9 DJZ (Decrement and Jump if Zero)†

Syntax

DJZ{
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
[control={OFF | ON}]
cond_addr=*{label | 8-bit unsigned integer}*
[reg={A | B | T | NONE}]
[irq={OFF | ON}]
[data=*20-bit unsigned integer]*
}

Opcode Ah, [P11:P8];

Sub-Opcode *Sub-opcode [P6-P5]=10*

Table 65. DJZ Program Field (P31:P0)

31 … 22	21	20	19 … 12	11 … 8	7	6 … 5	4 … 0
0	Res	BRK	Next program address	Opcode	Res.	Sub-opcode	Reserved
10	1-U	1	8	4	1	2	5

Table 66. DJZ Control Field (C31:C0)

31 … 22	21	20	19 … 12	11 … 3	2 … 1	0
0	Control	res.	Conditional address	Reserved	Register	Int. ena.
10	1	1	8	9	2	1

Table 67. DJZ Data Field (D31:D0)

31 … 25	24 … 5	4 … 0
0	Data	Reserved
7	20	5

Cycles One

Register modified selected register (A, B, or T)

† DJNZ is also supported syntax. The functionality of the two instruction names is identical.

Description

This instruction defines a virtual down counter used for delayed execution of certain instructions (to generate minimum on/off times). When DJZ is executed with counter value not zero, the counter value is decremented. If the counter value is zero, the counter remains zero until it is reloaded with a non-zero value. The program flow can be modified when down counter value is zero by using the conditional address.

cond_addr This field is not optional for the DJZ instruction.

irq ON generates an interrupt when the data field reaches zero. No interrupt is generated when the bit is OFF.

Default: OFF.

data Specifies the 20-bit integer value used as a counter. This counter is decremented each time the DJZ instruction is executed until the counter reaches 0.

Default: 0.

Execution

```
If (Data value != 0)
       {
       Selected register = Data field value - 1;
       Down Counter value = Data field value - 1;
       Jump to Next Program Address;
       }
else
       {
       Selected register = 00000h;
       If (Interrupt Enable == 1)
           SW interrupt flag = 1;
       Jump to conditional Address;
       }
```

7.4.10 ECMP (Equality Compare)

Syntax

ECMP {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
[hr_lr={HIGH | LOW}]
[angle_comp={OFF | ON}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr=*{label | 8-bit unsigned integer}]*
pin=pin number
[action={CLEAR | SET | PULSELO | PULSEHI}]
reg={A | B | T | NONE}
[irq={OFF | ON}]
data=*20-bit unsigned integer*
[hr_data=*{5-bit unsigned integer}]*
}

Opcode 0h, [P11:P8];

Sub-Opcode *00h, [C6:C5]*

Table 68. ECMP Program Field (P31:P0)

31 – 22	21	20	19 – 12	11 – 8	7	6	5 – 0
0	Res	BRK	Next program address	Opcode	hr_lr	Angle comp.	Reserved
10	1-U	1	8	4	1	1	6

Table 69. ECMP Control Field (C31:C0)

31 – 22	21	20	19 – 12	11 – 7	6 – 5	4 – 3	2 – 1	0
0	Control	Enable pin action	Conditional address	Pin select	Sub-opcode	Action	Register	Int. ena.
10	1	1	8	5	2	2	2	1

Table 70. ECMP Data Field (D31:D0)

31 – 25	24 – 5	4 – 0
0	Data	HR data
7	20	5

Cycles One

Register modified Register T if selected

Description ECMP can use all pins. This instruction compares a 20-bit data value stored in the data field (D24–D5) to the value stored in the selected ALU register (A, B, or T).

If T-register is selected, and if the 20-bit data field matches, ECMP updates T-register with the 25-bit value (D24-D0).

If an HR pin is chosen (HET23-HET0) and the hr_lr bit is cleared, pin action will occur after a delay from the next resolution clock. If the hr_lr bit is set, the delay is ignored. This delay is programmed in the data field (D4–D0). In the case of a non-HR pin, the pin action will be taken on the next loop resolution clock.

The behavior of the pins is governed by the four action options in bits C4:C3. ECMP uses the zero flag to generate opposite pin action (synchronized to the loop resolution clock).

angle_comp Determines if an angle compare is performed. A value of ON causes the comparison to be performed only if the new angle flag is set (NAF = 1). If OFF is specified, the compare is then performed regardless of the state of the new angle flag.

Default: OFF.

irq Specifies whether or not an interrupt is generated. A value of ON sends an interrupt if register and data field values are equivalent. If OFF is selected, no interrupt is generated.

Default: OFF.

data Specifies the value for the data field. This value is compared with the selected register.

hr_data Specifies the HR pin delay.

Default: 0.

Execution

```
If (Angle Comp. bit == 0 OR (Angle Comp. bit == 1 AND
NAF_global == 1))
    {
      If (Selected register value == Immediate data field
      value)
          {
          If (hr_lr bit == 0)
            {
            If (Enable Pin action == 1)
              Selected Pin = Pin Action AT next loop reso-
              lution clock + HR delay;
            }
          else
            {
            If (Enable Pin action == 1)
              Selected Pin = Pin Action AT next loop reso-
              lution clock;
            }
          If (Interrupt Enable == 1)
            SW interrupt flag = 1;
          If (register T is selected)
            T register = Compare value (25 bit);
          Jump to Conditional Address;
          }
      elseIf (Z == 1 AND Opposite action == 1)
          {
          If (Enable Pin action == 1)
            Selected Pin = opposite Pin Action AT next loop
            resolution clock;
          Jump to Next Program Address;
          }
      else
          Jump to Next Program Address;
      }
If (Angle Comp. bit == 1 AND NAF_global == 0)
      Jump to Next Program Address;
```

7.4.11 ECNT (Event Count)

Syntax

ECNT {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr=*{label | 8-bit unsigned integer}]*
pin= pin number
event={NAF | FALL | RISE |BOTH | ACCUHIGH | ACCULOW}
[reg={A | B | T | NONE}]
[irq={OFF | ON}]
data=*20-bit unsigned integer*
}

Opcode Ah;

Sub-Opcode *01h, [P6-P5]*

Table 71. ECNT Program Field (P31:P0)

31 … 22	21	20	19 … 12	11 … 8	7	6 … 5	4 … 0
0	Res	BRK	Next program address	Opcode	Res.	Sub-opcode	. Reserved
10	1-U	1	8	4	1	2	5

Table 72. ECNT Control Field (C31:C0)

31 … 22	21	20	19 … 12	11 … 7	6 … 4	3	2 … 1	0
0	Control	Prv.	Conditional address	Pin select	Event	Res.	Register	Int. en.
10	1	1	8	5	3	1	2	1

Table 73. ECNT Data Field (D31:D0)

31 … 25	24 … 5	4 … 0
0	Data	Reserved
7	20	5

Cycles One cycle

Register modified None

Description This instruction defines a specialized 20-bit virtual counter used as an event counter or pulse accumulator (see Table 74). The counter value is stored in the data field [D24:D5].

When an event count condition is specified, the counter value is incremented on a pin edge condition or on the NAF_global condition (NAF_global is defined in ACNT).

An edge detect is performed by comparing the current pin level, sampled at loop start, to the previous pin level stored in Prv. bit. The ECNT instruction updates the Prv. bit each time an edge is detected regardless of whether the edge is selected.

This instruction can be used with all pins.

event The event that triggers the counter.

Table 74. Event Encoding Format for ECNT

event	C6	C5	C4	Count Conditions	Mode	Int. Available
NAF	0	0	0	NAF flag is Set	Angle counter	Y
FALL	0	0	1	Falling edge on selected pin	Event counter	Y
RISE	0	1	0	Rising edge on selected pin	Event counter	Y
BOTH	0	1	1	Rising and Falling edge on selected pin	Event counter	Y
ACCUHIGH	1	0	-	while pin is high level	Pulse accumulation	N
ACCULOW	1	1	-	while pin is low level	Pulse accumulation	N

irq ON generates an interrupt when event in counter mode occurs. No interrupt is generated with OFF.

Default: OFF.

data 20-bit integer value serving as a counter.

Default: 0.

Execution

```
If (Event count condition is true according to bits [C6:C4]
(see table 1-17))
{
      Selected register = Immediate Data Field + 1;
      Immediate Data Field = Immediate Data Field + 1;
      If (Interrupt Enable == 1)
          SW interrupt flag = 1;
      Jump to Conditional Address;
      }
else
      Jump to Next Program Address;
Prv. bit = Selected Pin level; (Always executed)
```

7.4.12 MCMP (Magnitude Compare)

Syntax

MCMP {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
[hr_lr={LOW |HIGH}]
[angle_comp={OFF | ON}]
[savesub={OFF | ON}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr=*{label | 4-bit unsigned integer}]*
pin= pin number
order={REG_GE_DATA | DATA_GE_REG}
[action={CLEAR | SET | PULSELO | PULSEHI}]
reg={A | B | T | NONE}
[irq={OFF | ON}]
data=*20-bit unsigned integer*
[hr_data=*5-bit unsigned integer]*
}

Opcode 0h, [P11–P8];

Sub-Opcode *1h, C6*

Table 75. MCMP Program Field (P31:P0)

31–22	21	20	19–12	11–8	7	6	5	4	3–0
0	Res	BRK	Next program address	Opcode	hr_lr	Angle comp.	res.	Save sub.	res.
10	1-U	1	8	4	1	1	1	1	4

Table 76. MCMP Control Field (C31:C0)

31–22	21	20	19–12	11–7	6	5	4–3	2–1	0
0	Con-trol	En-able pin ac-tion	Conditional address	Pin select	Sub-op cod e	Or der	Action	Register	Int. ena.
10	1	1	8	5	1	1	2	2	1

Table 77. MCMP Data Field (D31:D0)

31–25	24–5	4–0
0	Data	HR data
7	20	5

Cycles One

Register modified Register T (implicitly)

Description This instruction compares the magnitude of the 20-bit data value stored in the data field (D24-D5) and the 20-bit value stored in the selected ALU register (A, B, or T). The register select coding for C2 and C1 are shown in Table 32, “Register Bit Field Encoding Format,” on page 83.

Note: The Difference Between Compare Values

The difference between the two data values must not exceed $(2^{19}) - 1$.

If an HR pin is chosen (HET23-HET0) and the hr_lr bit is reset, pin action will occur after a delay from the next resolution clock. If the hr_lr bit is set, the delay is ignored. This delay is programmed in the data field (D4-D0). In the case of a non-HR pin, the pin action will be taken on the next resolution clock.

When the data value matches, an output pin can be set or reset according to the pin action bit (C4). The pin will not change states if the enable pin action bit (C20) is reset.

MCMP uses the zero flag set to generate opposite pin action (synchronized to the loop resolution clock). The save sub bit (P4) provides the option to save the result of a subtraction into register T.

angle_comp Determines whether or not an angle compare is performed. A value of ON causes the comparison to be performed only if the new angle flag is set (NAF = 1). If OFF is specified, the compare is then performed regardless of the state of the new angle flag.

Default: OFF.

savesub When set, the comparison result is saved into the T register (upper 20 bits).

Default: OFF.

order Specifies the order of the operands for the comparison.

Table 78. Magnitude Compare Order for MCMP

Order	C5	Description
REG_GE_DATA	0	Evaluates to true if the register value is greater than or equal to the data field value.
DATA_GE_REG	1	Evaluates to true if the data field value is greater than or equal to the register value.

irq Specifies whether or not an interrupt is generated. A value of ON sends an interrupt if the register and data field values are equivalent. If OFF is selected, no interrupt is generated.

data Specifies the value for the data field. This value is compared with the selected register.

hr_data HR pin display. The default value for an unspecified bit is 0.

Execution

```
If (Angle Comp. bit == 0 OR (Angle Comp. bit == 1 AND
NAF_global == 1))
      {
      If ((C5 == 1 AND (Immediate data value - Selected reg-
      ister) ≥ 0) OR (C5 == 0 AND (Selected register - Imme-
      diate data value) ≥ 0))
          {
          If (hr_lr bit == 0)
            {
            If (Enable Pin action == 1)
              Selected Pin = Pin Action AT next loop reso-
              lution clock + HR delay;
            }
          else
            {
            If (Enable Pin action == 1)
              Selected Pin = Pin Action AT next loop reso-
              lution clock;
            }
```

```
                If (Interrupt Enable == 1)
                  SW interrupt flag = 1;
                If (Save sub. == 1)
                  {
                  If ([C5] == 1)
                    T register = Immediate data value - Selected
                    register;
                  If ([C5] == 0)
                    T register = Selected register - Immediate
                    data value;
                  }
                Jump to Conditional Address;
                }
            elseIf (Z == 1 AND Opposite action == 1)
                {
                If (Enable Pin action == 1)
                  Selected Pin = opposite Pin Action AT next loop
                  resolution clock;
                Jump to Next Program Address;
                }
            else
                Jump to Next Program Address;
            }
      If (Angle Comp. bit == 1 AND NAF_global == 0)
            Jump to Next Program Address;
```

7.4.13 MOV32 (MOVE 32)

Syntax

MOV32 {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
remote=*{label | 8-bit unsigned integer}*
[control={OFF | ON}]
[init={OFF | ON}]
type={IMTOREG | IMTOREG&REM | REGTOREM | REMTOREG}
reg={A | B | T | NONE}
[data=*20-bit unsigned integer]*
[hr_data=*5-bit unsigned integer]*
}

Opcode 4h, [P11:P8];

Sub-Opcode *C5=0*

Table 79. MOV32 Program Field (P31:P0)

31–22	21	20	19–12	11–8	7–0
0	Res	BRK	Next program address	Opcode	Remote address
10	1-U	1	8	4	8

Table 80. MOV32 Control Field (C31:C0)

31–22	21	20–7	6	5	4–3	2–1	0
0	Con-trol	Reserved	Init flag	Sub op-code	Move type	Register	Res.
10	1	14	1	1	2	2	1

Table 81. MOV32 Data Field (D31:D0)

31–25	24–5	4–0
0	Data field	HR data
7	20	5

Cycles One or two cycles

Register modified Selected register (A, B or T)

Description MOV32 replaces the selected ALU register and/or the data field values at the remote address location depending on the move type. Table 32, “Register Bit

Field Encoding Format," on page 83 shows the C2 and C1 bit encoding for selecting the desired register.

Table 82 shows the [C4:C3] bit encoding for selecting the destination for a data move. Figure 60 through Figure 63 illustrate these operations.

If *no register* is selected, the move is not executed, except for configuration C4:C3 = 01, where the remote data field is written with the immediate data field value.

remote Determines the location of the remote address.

Default: Current instruction + 1.

init (Optional) Determines whether or not system flags are initialized. A value of ON reinitializes the following system flags to these states:

- Acceleration flag (ACF) = 0
- Deceleration flag (DCF) = 1
- Gap flag (GPF) = 0
- New angle flag (NAF) = 0

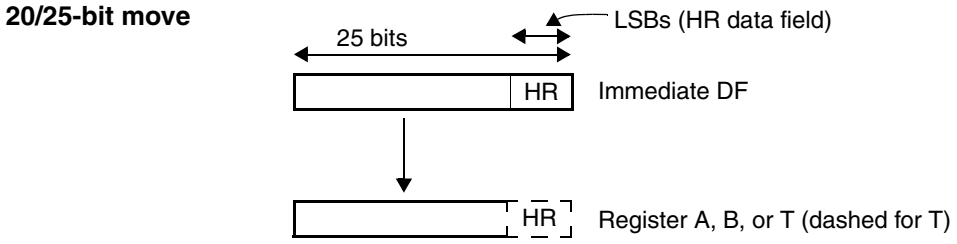
A value of OFF results in no change to the system flags.

type Specifies the move type to be executed.

Table 82. Move Type Encoding Selection

Move Type	C4	C3	Source	Destination(s)	Cycles
IMTOREG	0	0	Immediate data field	Register A, B, or T	1
IMTOREG&REM	0	1	Immediate data field	Remote data field & register A, B, or T	1
REGTOREM	1	0	Register A, B, or T	Remote data field	1
REMTOREG	1	1	Remote data field	Register A, B, or T	2

Figure 60. MOV32 Move Operation for IMTOREG (Case 00)



reg Specifies which register (A, B, T, or NONE) is involved in the move. A register (A, B, or T) must be specified for every move type except IMTOREG&REM. If *NONE* is used with move type IMTOREG&REM, the assembler executes a move from the immediate data field to the remote data field. If *NONE* is used with any other move type, no move is executed.

data Specifies a 20-bit integer value to be written to the remote data field or selected register.

hr_data (Optional) HR pin delay. The default value for an unspecified bit is 0.

Figure 61. MOV32 Move Operation for IMTOREG&REM (Case 01)

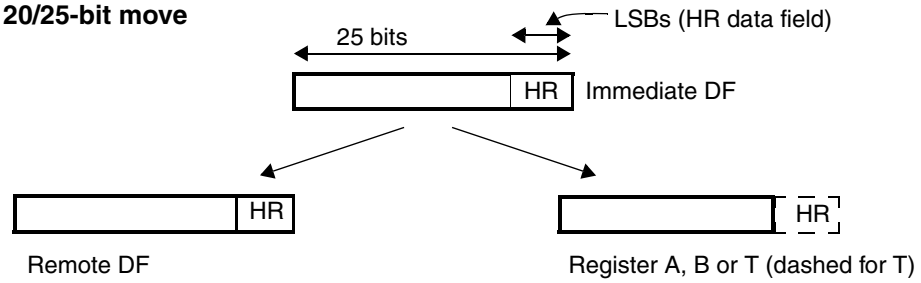


Figure 62. MOV32 Move Operation for REGTOREM (Case 10)

20/25-bit move

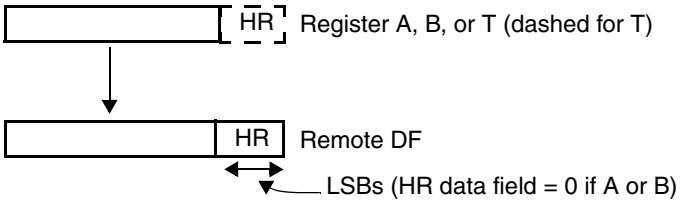
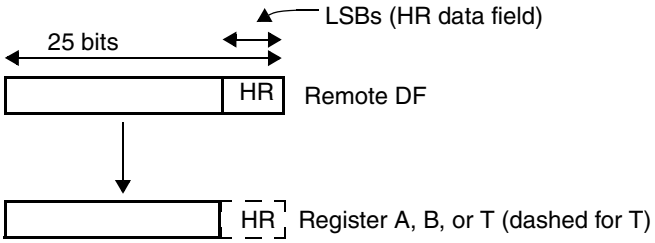


Figure 63. MOV32 Move Operation for REMTOREG (Case 11)

20/25-bit move



Execution

```
switch (C4:C3)
      {
      case 00:
          Selected register = Immediate Data Field;
      case 01:
          Selected register = Immediate Data Field;
          Remote Data Field = Immediate Data Field;
      case 10:
          Remote Data Field = Selected register;
      case 11:
          Selected register = Remote Data Field;
      }

If (Init Flag == 1)
      {
      ACF = 0;
      DCF = 1;
      GPF = 0;
      NAF = 0;
      }

else
      All flags remain unchanged;

Jump to Next Program Address;
```

7.4.14 MOV64 (Data Move)

Syntax

MOV64 {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
remote=*{label | 8-bit unsigned integer}]*
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr=*{label | 4-bit unsigned integer}]*
[pin= pin number]
comp_mode={ECMP | SCMP | MCMP | ACMP}
[action={CLEAR | SET | PULSELO | PULSEHI}]
reg={A | B | T | NONE}
[irq={OFF | ON}]
[data=*20-bit unsigned integer]*
[hr_data=*5-bit unsigned integer]*
}

or

Syntax

MOV64 {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
remote=*{label | 8-bit unsigned integer}]*
[control={OFF | ON}]
[cntl_val=*21-bit unsigned integer]*
[data=*20-bit unsigned integer]*
[hr_data=*5-bit unsigned integer]*

}

Opcode 1h, [P11:P8]

Table 83. MOV64 Program Field (P31:P0)

31 — 22	21	20	19 — 12	11 — 8	7 — 0
0	Res	BRK	Next program address	Opcode	Remote address
10	1-U	1	8	4	8

Table 84. MOV64 Control Field (C31:C0)

31–22	21	20	19–12	11–7	6–5	4–3	2–1	0
0	Control	Enable pin action	Conditional address	Pin select	Comp. mode	Action	Register	Int. ena.
10	1	1	8	5	2	2	2	1

Table 85. MOV64 Data Field (D31:D0)

31–25	24–5	4–0
0	Data	HR data
7	20	5

Cycles One

Register modified None

Description This instruction modifies the data field and the control field at the remote address.

MOV64 has two distinct syntaxes. In the first syntax, bit values may be set by assigning a value to each of the control fields. This syntax is convenient for modifying control fields that are arranged similarly to the format of the MOV64 control field. A second syntax, in which the entire 21-bit control field is specified by the cntl_val field, is convenient when the remote control field is dissimilar to the MOV64 control field. Either syntax may be used, but you must use one or the either but not a combination of syntaxes. See Figure 64.

Figure 64. MOV64 Move Operation

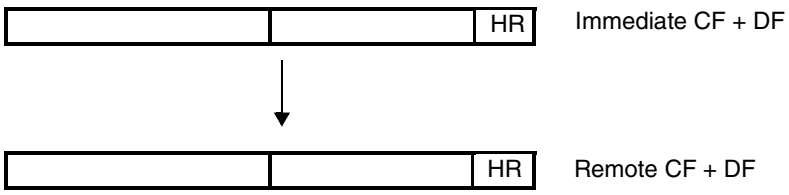


Table 86. MOV64 Control Field Descriptions

Control	Maintains the control field for the remote instruction.
en_pin_action	Maintains the control field for the remote instruction.
cond_addr	Maintains the control field for the remote instruction.
pin	Maintains the control field for the remote instruction.
register	Maintains the control field for the remote instruction.
action	Maintains the control field for the remote instruction.
irq	Maintains the control field for the remote instruction.
data	Specifies the 20-bit initial count value for the data field. If omitted, the field defaults to 0.
hr_data	(Optional) HR pin delay. The default value for an unspecified bit is 0.
comp_mode	Selects the comparison mode type to be used.

Table 87. Comparison Type Encoding Format

comp_mode	C6	C5
ECMP	0	0
SCMP	0	1
MCMP	1	0
ACMP	1	1

Execution

```
Remote Data Field = Immediate Data Field;
Remote Control Field = Immediate control Field;
Jump to Next Program Address;
```

7.4.15 PCNT (Period/Pulse Count)

Syntax

PCNT {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
[irq={OFF | ON}]
type={RISE2FALL | FALL2RISE | FALL2FALL | RISE2RISE}
pin=pin number
[control={OFF | ON}]
[prv={OFF | ON}]
[period=*20-bit unsigned integer]*
[data=*20-bit unsigned integer]*
[hr_data=*5-bit unsigned integer]*
}

Opcode 7h, [P11:P8]

Table 88. PCNT Program Field (P31:P0)

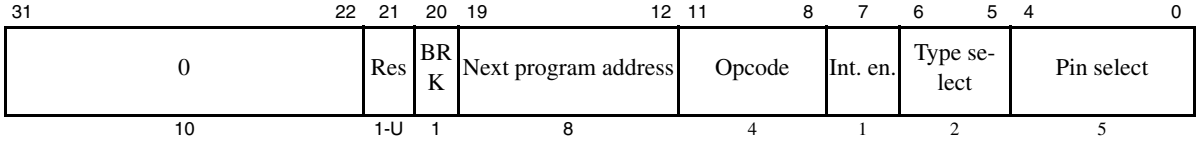


Table 89. PCNT Control Field (C31:C0)

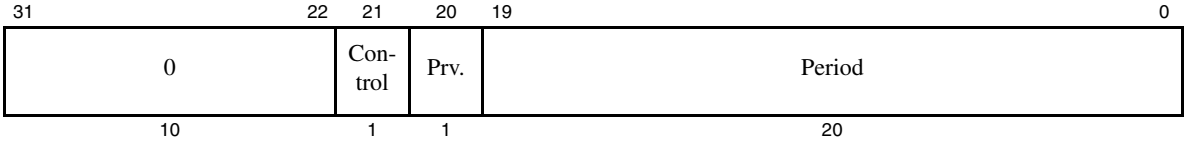
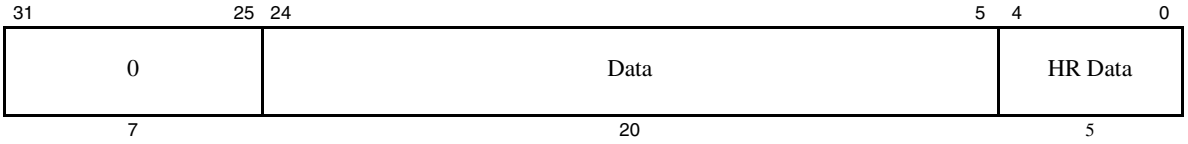


Table 90. PCNT Data Field (D31:D0)



Cycles One

Register modified Register A

Description This instruction detects the edges of the external signal at loop start and measures its period or pulse duration. The counter value stored in the control field [C19:C0] and in the register A is incremented on each timer resolution. If an HR pin is selected (HET23-HET0), then PCNT will use the HR structure on the pin to measure an HR period/pulse count value.

irq (Optional) Specifies whether or not an interrupt is generated. A value of ON sends an interrupt if register and data field values are equivalent. If OFF is selected, no interrupt is generated.

type (Optional) Determines the type of counter that is implemented.

Table 91. Counter Type Encoding Format

	P6	P5	Period/Pulse Select	Reset On	Capture On
FALL2RISE	0	0	Count low pulse duration on selected pin	Falling edge	Rising edge
RISE2FALL	0	1	Count high pulse duration on selected pin	Rising edge	Falling edge
FALL2FALL	1	0	Count period between falling edges on selected pin	Falling edge	Falling edge
RISE2RISE	1	1	Count period between rising edges on selected pin	Rising edge	Rising edge

period Specifies the 20-bit integer value that holds the previous counter value. The previous counter value is also stored in register A.

Default: 0.

data 20-bit integer value serving as a counter.

Default: 0.

hr_data HR pin delay.

Default: 0.

If the control bit is set, the period value is automatically cleared after the main CPU performs a 64-bit read access.

If *period-measure* is selected, PCNT captures the counter value into the period/pulse data field [D24:D5] on the selected edge. The HR structure provides HR capture field [D4:D0]. The counter value [C19:C0] is reset on the same edge. The period value is a 25-bit value.

If *pulse-measure* is selected, PCNT captures the counter value into the period/pulse count field [D24:D5] on the selected edge. The HR structure

provides HR capture field [D4:D0]. The counter value [C19:C0] is reset on the next opposite edge. The pulse value is a 25-bit value.

An edge detect is performed by comparing the current pin level, sampled at loop start, to the previous pin level stored in Prv. bit.

When the overflow count (all 1's in the counter value) is reached, PCNT stops counting until the next reset edge is detected. Table 31, "PIN Encoding Format," on page 83 shows the pin encoding for the pin select field [P4:P0].

Execution

```
Z = 0;
If (Period/Pulse select [P6,P5] == 1X)
      {
      If (Period value != FFFFFh)
          {
          Register A = Period value + 1;
          Period value = Period value + 1;
          }
      elseIf (specified edge not detected on selected pin)
          Register A = FFFFFh;
          HR capture value = 11111;
      If (specified edge detected on selected pin)
          {
          Z = 1;
          If (Period value == FFFFFh)
            {
            Register A = FFFFFh;
            Data field = FFFFFh;
            }
          else
            {
            Data field = Period value + 1;
            HR capture value = selected HR counter;
            Period value = 00000h;
            If (Interrupt Enable == 1)
              SW interrupt flag = 1;
```

```
                Jump to Next Program Address;
                }
              }
        }
else /*** Pulse mode ***/
        {
        If (Period value != FFFFFh)
              {
              Register A = Period value + 1;
              Period value = Period value + 1;
              }
        If (specified reset edge is detected on selected pin)
              Period value = 00000h;
        If (specified capture edge is detected on selected
        pin)
              {
              Z = 1;
              If (Period value == FFFFFh)
                {
                Register A = FFFFFh;
                Data field = FFFFFh;
                HR capture value = 11111;
                }
              else
                {
                Data field = Period value + 1;
                If (Interrupt Enable == 1)
                  SW interrupt flag = 1;
                }
              }
        }
        Jump to Next Program Address;
Prv. bit = Selected Pin level;
```

7.4.16 PWCNT (Pulse Width Count)

Syntax

PWCNT {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
[hr_lr={HIGH | LOW}]
[control={OFF | ON}]
[cond_addr=*{label | 8-bit unsigned integer}*
[en_pin_action={OFF | ON}]
pin =pin number
[action={CLEAR | SET | PULSELO | PULSEHI}]
[reg={A | B | T | NONE}]
[irq={OFF | ON}]
data=*20-bit unsigned integer*
[hr_data=*5-bit unsigned integer]*
}

Opcode Ah, [P11:P8];

Sub-Opcode *11h, [P6-P5]*

Table 92. PWCNT Program Field (P31:P0)

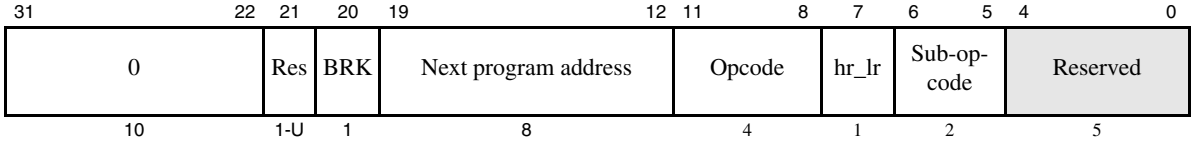


Table 93. PWCNT Control Field (C31:C0)

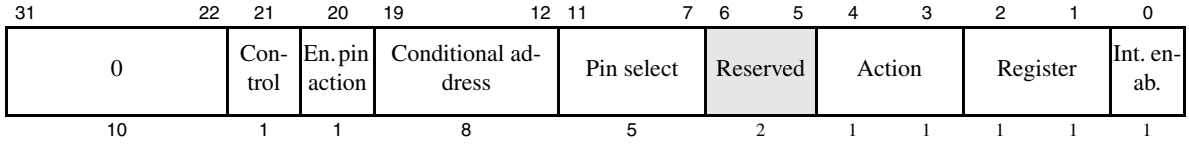
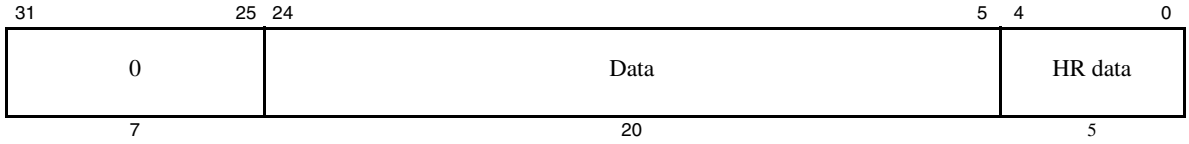


Table 94. PWCNT Data Field (D31:D0)



Cycles One

Register modified Selected register (A, B or T)

Description

This instruction defines a virtual timer used to generate variable length pulses. The counter value stored in the data field is decremented unconditionally on each timer resolution until it reaches zero, and it then stays at zero until it is reloaded with a non-zero value.

The specified pin action is performed as long as the count after count value is decremented is greater than 0. The opposite pin action is performed when the count after decrement just reaches 0.

If an HR pin is chosen (HET23–HET0) and the hr_lr bit is reset, the opposite pin action will be taken after a delay from the next resolution clock. If the hr_lr bit is set, the delay is ignored. This delay is programmed in bits [D4:D0]. In the case of a non-HR pin, the opposite pin action is taken on the next resolution clock.

irq ON generates an interrupt when the data field value reaches 0. No interrupt is generated for OFF

Default: OFF.

data 20-bit integer value serving as a counter.

hr_data HR pin delay.

Default: 0.

Execution

```
If (Data field value == 0)
      {
      Selected register = 0;
      Jump to Next Program Address;
      }
If (Data field value > 1)
      {
      Selected register = Data field value - 1;
      Data field value = Counter value - 1;
      If (Enable Pin action == 1)
          Selected Pin = Pin Action AT next loop resolution
          clock;
      Jump to Next Program Address;
      }
If (Data field value == 1)
      {
      Selected register = 00000h;
      Data field value = 00000h;
      If (Opposite action == 1)
          {
          If (hr_lr bit == 0)
            {
            If (Enable Pin action == 1)
              Selected Pin = Opposite level of Pin Action
              AT next loop resolution clock + HR delay;
            }
          else
            {
            If (Enable Pin action == 1)
              Selected Pin = Opposite level of Pin Action
              AT next loop resolution clock;
            }
          }
      If (Interrupt Enable == 1)
      SW interrupt flag = 1;
      Jump to Conditional Address
      }
```

7.4.17 RADM64 (Register Add Move)

Syntax

RADM64 {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
remote=*{label | 8-bit unsigned integer}*
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr=*{label | 4-bit unsigned integer}]*
[pin= pin number]
comp_mode={ECMP | SCMP | MCMP | ACMP}
[action={CLEAR | SET | PULSELO | PULSEHI}]
reg={A | B | T | NONE}
[irq={OFF | ON}]
data=*20-bit unsigned integer*
[hr_data=*5-bit unsigned integer]*
}

or

Syntax

RADM64 {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
remote=*{label | 8-bit unsigned integer}*
[cntl_val = *21-bit unsigned integer]*
[control={OFF | ON}]
data=*20-bit unsigned integer*
[hr_data=*5-bit unsigned integer*]
}

Opcode

3h, [P11:P8]

Table 95. RADM64 Program Field (P31:P0)

31 … 22	21	20	19 … 12	11 … 8	7 … 0
0	Res	BRK	Next program address	Opcode	Remote
10	1-U	1	8	4	8

Table 96. RADM64 Control Field (C31:C0)

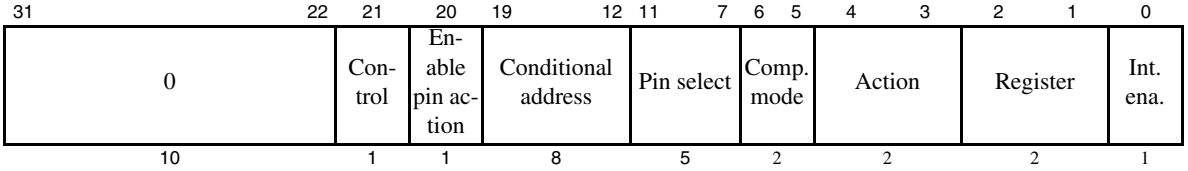
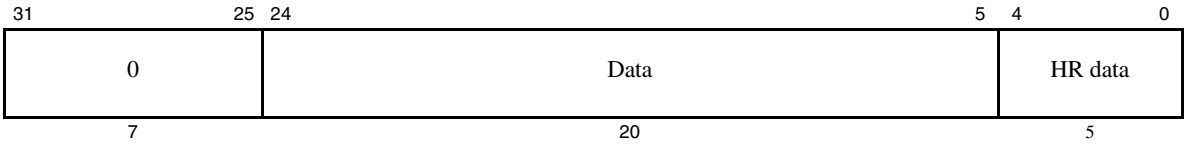


Table 97. RADM64 Data Field (D31:D0)



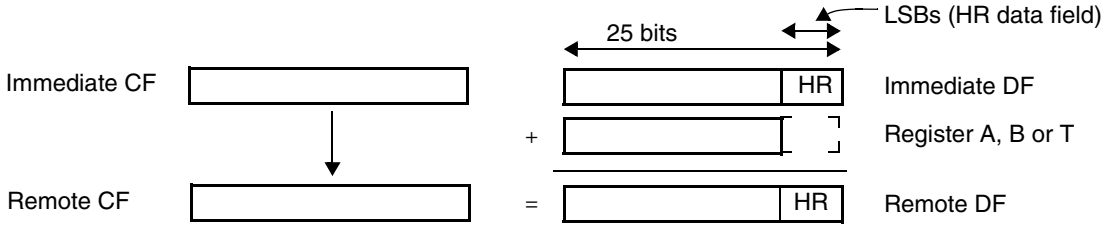
Cycles One

Register modified Selected register (A, B or T)

Description This instruction modifies the data field, the HR data field and the control field at the remote address. The advantage over DADM64 is that It executes one cycle faster. In case of the T-register selected, the addition is a 25-bit addition. The table description shows the bit encoding for determining which ALU register is selected.

RADM64 has two distinct syntaxes. In the first syntax, bit values may be set by assigning a value to each of the control fields. This syntax is convenient for modifying control fields that are arranged similar to the format of the RADM64 control field. A second syntax, in which the entire 21-bit control field is specified by the cntl_val field, is convenient when the remote control field is dissimilar from the RADM64 control field. Either syntax may be used, but you must use one or the either but not a combination of syntaxes. See Figure 65.

Figure 65. RADM64 Add and Move Operation



comp_mode	Selects the comparison mode type to be used.

Table 98. Comparison Type Encoding Format

comp_mode	C6	C5
ECMP	0	0
SCMP	0	1
MCMP	1	0
ACMP	1	1

Control	Maintains the control field for the remote instruction.
en_pin_action	Maintains the control field for the remote instruction.
cond_addr	Maintains the control field for the remote instruction.
pin	Maintains the control field for the remote instruction.
register	Maintains the control field for the remote instruction.
action	Maintains the control field for the remote instruction.
irq	Maintains the control field for the remote instruction.
data	Specifies the 20-bit initial value for the data field. If omitted, the field defaults to 0.
hr_data	Five least significant bits of the 25-bit data field. Default: 0.
cntl_val	Specifies the 21 least significant bits of the Control field.

Execution

```
Remote Data Field = Selected register + Immediate Data
Field (including HR data field);

Remote Control Field = Immediate Control Field;

Jump to Next Program Address;
```

7.4.18 SCMP (Sequence Compare)

Syntax

SCMP {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
cond_addr=*{label | 8-bit unsigned integer}*
pin=pin number
[action={CLEAR | SET}]
[restart={OFF | ON}]
[irq={OFF | ON}]
data=*20-bit unsigned integer*
}

Opcode 0h, [P11–P8];

Sub-Opcode *[C6-C5]=01*

Table 99. SCMP Program Field (P31:P0)

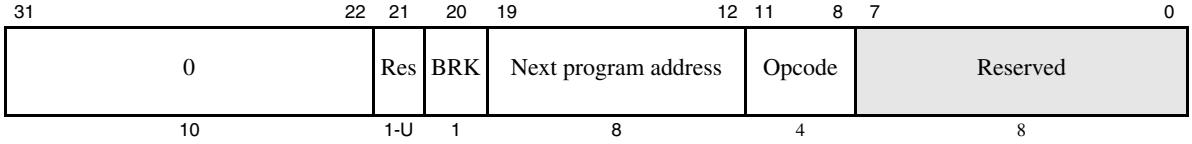


Table 100. SCMP Control Field (C31:C0)

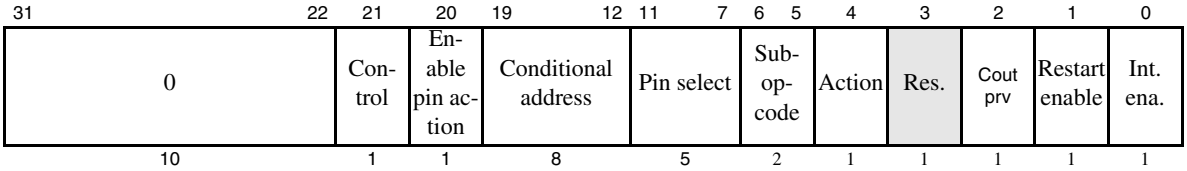
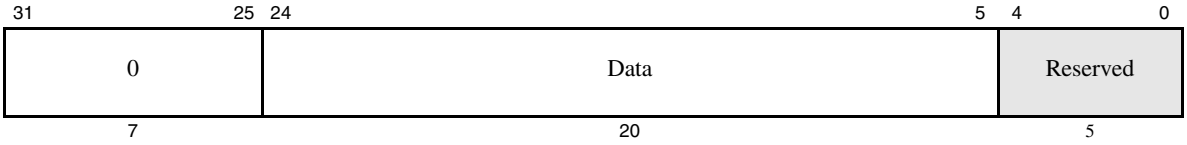


Table 101. SCMP Data Field (D31:D0)



Cycles One

Register modified Register T (implicitly)

Description This instruction alternately performs angle- and time-based operations to generate pulse sequences, using the angle referenced time base. These pulse sequences last for a relative duration using a free running time base. Generally, register B holds the angle values and register A holds the time values. Bit 0 of the conditional address field (C12) specifies whether the instruction is operating in angle or time operation mode.

When the compared values match in angle mode, a pin can be set or reset according to the pin action bit (C4). The pin does not change states if the enable pin action bit (C20) is reset.

The restart enable bit (C1) provides the option to unconditionally restart a sequence using the X-flag bit of ACMP.

restart If restart is set to ON and the X flag = 1, the assembler writes a value of 1 into the immediate index field, writes the value in register A into the immediate data field, and jumps to the next program address. The X flag is set or cleared by the ECMP instruction. If restart is set to OFF, the X flag is ignored; no special action is performed.

Default: OFF.

irq ON generates an interrupt if the register and data field values are equivalent. No interrupt is generated when the field is OFF.

Default: OFF.

data Specifies the 20-bit compare value.

cond_addr Since the LSB of the conditional address is used to select between time mode and angle mode, and since the conditional address is taken only in time mode, the destination for the conditional address must be odd.

Execution

```
If (Data field value ≤ Selected register value)
      Cout = 0;
else
      Cout = 1;
If (Restart Enable == 1 AND X == 1)
      {
      C12 = 1;
      Immediate Data Field = Register A;
      Jump to Next Program Address;
      }
If (Angle Mode is selected (C12 == 0) AND ((Restart En. ==
1 AND X == 0) OR Restart En. == 0))
      {
      If (Z == 0 AND (Register B value - Angle Inc. < Data
      field value) AND Cout == 0) OR

      (Z == 1 AND Cout_prv == 1 OR Cout == 0))

          {
          If (Enable Pin Action == 1)
            Selected Pin = Pin Action;
          If (Interrupt Enable == 1)
            SW interrupt flag = 1;
          Immediate Data Field = Register A;
          C12 = 1; /*** switch to Time Mode ***/
          Jump to Conditional Address;
          }

      else
          Jump to Next Program Address;
      }
If (Time Mode is selected (C12 == 1)) AND ((Restart En. ==
1 AND X == 0) OR Restart En. == 0)
      {
      Register T = Register A - Immediate Data Field;
      (Result of subtract must not exceed (2^19) - 1)
      Jump to Conditional Program Address;
      }
Cout_prv = Cout; (always executed)
```

Example 1. SCMP Example

```
  ; Time Counter in register A
A0CNT{reg=A, max=0x0fffff}
  ; Angle Counter (320 degree) in register B
A1CNT{reg=B, max=0x3C00, irq=OFF, angle_count=ON, da-
ta=0x400}
  ; Angle compare to set pulse on CC23 when match X flag
is set
A2ACMP{next=A3, en_pin_action=ON, cond_addr=A3, pin=CC23,
action=SET, reg=B, irq=OFF, data=0x0220}
  ; Transfer match angle into time, when X is set trans-
fer A into DF, switch in time mode and jump to MCMP else
next ACMP (next loop resolution)
A3SCMP{next=A6, en_pin_action=ON, cond_addr=A5, pin=CC23,
action=SET, restart=ON, irq=OFF, data=0x0230}
  ; restart the sequence compare and jump to next resolu-
tion
A4MOV64{next=A6, remote=A3, en_pin_action=ON, cond_addr=A4,
pin=CC23, comp_mode=SCMP, action=SET, reg=B, data=0x0230}
  ; When T (= Reg. A - SCMP Data field) > compare value
then reset the pin and jump to MOV64 to re-initialize the
SCMP Control and Data field.
A5MCMP{next=A6, en_pin_action=ON, pin=CC23, hr_lr=LOW, ac-
tion=CLEAR, order=REG_GE_DATA, cond_addr=A4, reg=T, da-
ta=0x214, hr_data=0x10}
  ; Branch to first instruction (end of the loop)
A6BR{next=A0, cond_addr=A0, pin=CC25, event=NOCOND}
```

This program generates a pulse always at the same position of a wheel (ACMP instruction) and the pulse lasts for a certain time (value in MCMP instruction) whatever the speed of the wheel.

7.4.19 SCNT (Step Count)

Syntax

SCNT {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
step={8 | 16 | 32 | 64}
[control={OFF | ON}]
gapstart=*20-bit unsigned integer*
[data=*20-bit unsigned integer]*
}

Opcode

Ah, [P11:P8];

Sub-opcode [P6-P5]=00

Table 102. SCNT Program Field (P31:P0)

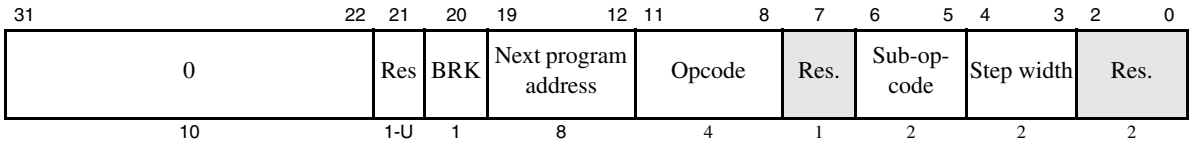


Table 103. SCNT Control Field (C31:C0)

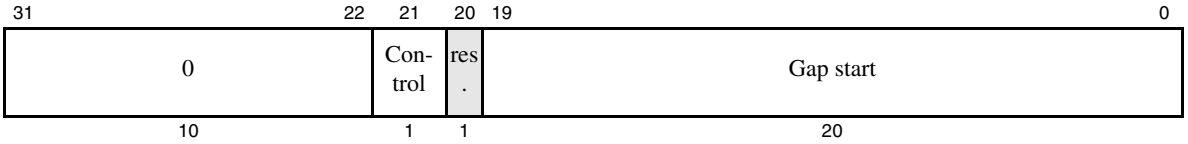
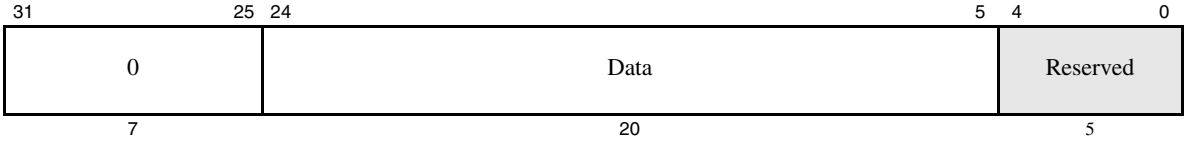


Table 104. SCNT Data Field (D31:D0)



Cycles

Three

Register modified

Register A

Description

This instruction can be used only once in a program and defines a specialized virtual timer used after APCNT and before ACNT to generate an angle-referenced time base synchronized to an external signal (that is, a toothed wheel signal) as defined in APCNT and ACNT. Step width selection bits are saved in two flags, SWF0, and SWF1, to be re-used in ACNT.

SCNT multiplies the frequency of the external signal by a constant *K* defined in the step width field, [P4:P3]. The bit encoding for this field is defined in Table 105:

step Specifies the step increment to be added to the counter value each program resolution. These two bits provide the values for the SWF0 and SWF1 flags. The valid values are listed in the Table 105.

Table 105. Step Width Encoding for SCNT

P4	P3	Step Width (K)
0	0	8
0	1	16
1	0	32
1	1	64

gapstart Defines the gap start angle, which SCNT writes to register A. The gap start value has no effect on the SCNT instruction, but if the ACNT instruction is being used, register A must contain the correct gap start value. For a typical toothed wheel gear:

GAPSTART= (stepwidth x (actual teeth on gear - 1)) + 1.

data Specifies the 20-bit integer value serving as a counter.

Default: 0.

This instruction is incremented by the step value K on each timer resolution up to the previous period value P(n-1) measured by APCNT (stored in register T). The resulting period of SCNT is: $P(n-1)/K$

Due to stepping, the final count of SCNT will not usually exactly match the target p(n-1). SCNT compensates for this error by starting each cycle with the remainder of the previous cycle.

When SCNT reaches the target p(n-1), the zero flag is set as an increment condition for ACNT.

SCNT also specifies a gap start angle, defining the start of a range in ACNT where period measurements in APCNT are temporarily stopped to mask singularities in the external signal.

SCNT uses register A to store the gap start value. Gap start has no effect for SCNT.

Execution

```
SWF1 = P4;
SWF0 = P3;
Z = 0;
If (register T == 00000h)
      Jump to Next Program Address;
else
      {
      If (DCF == 1 OR ACF == 1)
           {
           Data Field register = 00000h;
           Counter value = 00000h;
           }
      If (DCF == 0 AND ACF == 0)
           {
           Data Field register = Data field register + Step
           Width;
           }
      If ((Data Field register - register T) ≥ 0)
           {
           Data field register = Data Field register - reg-
           ister T;
           Z = 1;
           }
      }
Register A = Gap start value;
Jump to Next Program Address;
```

7.4.20 SHFT (Shift)

Syntax

SHFT {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
smode={OR0 | OL0 | OR1 | OL1 | ORZ | OLZ | IRM | ILL | IRZ | ILZ}
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr=*{label | 8-bit unsigned integer}]*
cond={UNC | FALL | RISE}
pin=pin number
[reg={A | B | T | NONE}]
[irq={OFF | ON}]
data=*{20-bit unsigned integer}*
}

Opcode Fh, [P11:P8]

Table 106. SHFT Program Field (P31:P0)

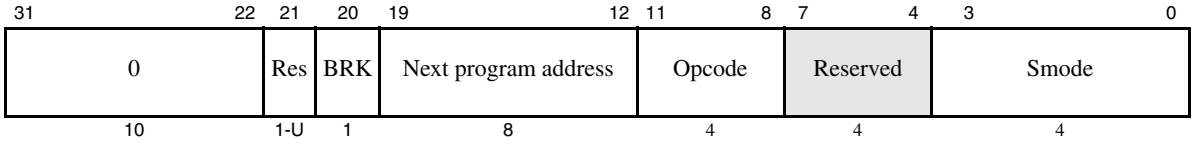


Table 107. SHFT Control Field (C31:C0)

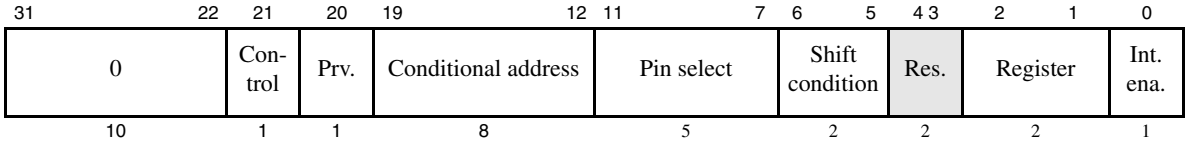
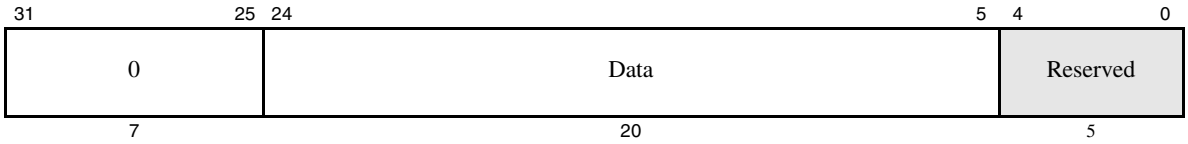


Table 108. SHFT Data Field (D31:D0)



Cycles One

Register modified Selected register (A, B or T)

Description This instruction may only be used on pins CC24-CC31.

This instruction shift the data field of the Instruction. HET pins can be used for data in or data out. SHFT includes parameters to select the shift direction (in, out, left, right), shift condition (shift on a defined clock edge on HET31 or shift always), register for data storage (A, B, or T), and the data pin.

smode Shift mode.

Table 109. SHIFT MODE Encoding Format

smode	P3	P2	P1	P0	Operation	
OR0	0	0	0	0	Shift Out / Right	LSB 1st on HETx / 0 into MSB
OL0	0	0	0	1	Shift Out / Left	MSB 1st on HETx / 0 into LSB
OR1	0	0	1	0	Shift Out / Right	LSB 1st on HETx / 1 into MSB
OL1	0	0	1	1	Shift Out / Left	MSB 1st on HETx / 1 into LSB
ORZ	0	1	0	0	Shift Out / Right	LSB 1st on HETx / Z into MSB
OLZ	0	1	0	1	Shift Out / Left	MSB 1st on HETx / Z into LSB
IRM	1	0	0	0	Shift In / Right	HETx into MSB
ILL	1	0	0	1	Shift In / Left	HETx into LSB
IRZ	1	0	1	0	Shift In / Right	HETx in MSB / LSB into Z
ILZ	1	0	1	1	Shift In / Left	HETx in LSB / MSB into Z

cond Specifies the shift condition.

Table 110. SHIFT Condition Encoding

C6	C5	Shift condition
0	X	Always
1	0	Rising edge of HET31
1	1	Falling edge of HET31

irq	ON generates an interrupt if the Z flag is set. A value of OFF does not generate an interrupt. Default: OFF.
data	Specifies the 20-bit value for the data field.

Execution

```
If (SHIFT condition == 0X)
OR (SHIFT condition == 10 AND CC31 rising edge)
OR (SHIFT condition == 11 AND CC31 falling edge)
    {
    Shift Immediate Data Field by one bit according to bits
    [P3:P0]; (see Table 109)
    Immediate Data Field = Result of the shift;
    Selected register = Result of the shift;

    If ((Immediate Data Field == all 0's AND [P3:P0] == 000X)
    OR (Immediate Data Field == all 1's AND [P3:P0] == 001X))
          {
          Z = 1;
          }
    else if ([P3:P0] == 1010)
          {
          Z = LSB of the Immediate Data Field;
          }
    else if ([P3:P0] == 1011)
          {
          Z = MSB of the Immediate Data Field;
          }
    else
          {
          Z = 0;
          }

    If ((Immediate Data Field == all 0's OR
        (Immediate Data Field == all 1's)
        AND (Interrupt Enable == 1))
          {
          SW interrupt flag = 1;
          Jump to Conditional Address;
          }
    else
          {
          Jump to Next Program Address;
          }
    }
Prv. bit = CC31 Pin level; (Always executed)
Jump to Next Program Address;
```

Note:

The immediate data field evaluates all 0s or all 1s and is performed before the shift operation.

7.4.21 WCAP (Software Capture Word)

Syntax

WCAP {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
[hr_lr={HIGH | LOW}]
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr=*{label | 8-bit unsigned integer}]*
pin=pin number
event={NOCOND | FALL | RISE | BOTH}
reg ={A | B | T | NONE}
[irq={OFF | ON}]
data=*20-bit unsigned integer*
[hr_data=*5-bit unsigned integer]*
}

Opcode Bh, [P11:P8]

Table 111. WCAP Program Field (P31:P0)

31–22	21	20	19–12	11–8	7	6–0
0	Res	BRK	Next program address	Opcode	hr_lr	Reserved
10	1-U	1	8	4	1	7

Table 112. WCAP Control Field (C31:C0)

31–22	21	20	19–12	11–7	6–5	4–3	2–1	0
0	Con-trol	Prv.	Conditional ad-dress	Pin select	Capture condition	Reserved	Register	Int. ena.
10	1	1	8	5	2	2	2	1

Table 113. WCAP Data Field (D31:D0)

31–25	24–5	4–0
0	Data	HR data
7	20	5

Cycles One

Register modified None

Description

This instruction captures the selected register into the data field if the specified capture condition is true on the selected pin. This instruction can be used with all pins.

If an HR pin is selected (HET23-HET0) and the hr_lr bit is reset, the WCAP instruction will capture an HR time stamp into the data field on the selected edge condition. If the hr_lr bit is set, the HR capture is ignored. Edge detection is performed by comparing the current pin level sampled at loop start to the previous pin level stored in the Prv. (Previous) bit [C20].

This instruction updates the Prv. bit each time an edge is detected regardless of whether the edge is selected.

event Specifies the event that triggers the capture.

Table 114. Event Encoding Format for WCAP

C6	C5	Capture Condition
0	0	always
0	1	Capture on falling edge
1	0	Capture on rising edge
1	1	Capture on rising and falling edges

irq ON generates an interrupt when the capture condition is met. No interrupt is generated for OFF
Default: OFF.

data Specifies the 20-bit integer value to be written to the data field or selected register.

hr_data HR capture value.
Default: 0.

Execution

```
If (Specified Capture Condition is true on Selected Pin
OR Unconditional capture is selected) {
    Immediate Data Field = Selected register value;
    If (hr_lr bit == 0)
          Capture the HR value in Immediate HR Data Field;
    If (Interrupt Enable == 1)
          SW interrupt flag = 1;
    Jump to Conditional Address;
    }
else
    Jump to Next Program Address;
Prv. bit = Selected Pin level;
```