OMAP5910/5912 Applications Processor Timers Reference Guide

Literature Number: SPRU891
February 2005

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

Preface

Read This First

About This Manual

This document describes the timers present in the OMAP5910/5912 devices.

Related Documentation From Texas Instruments

Documentation that describes the OMAP5910/5912 devices, related peripherals, and other technical collateral, is available in the OMAP5910 Product Folder on TI’s website: www.ti.com/omap5910, and in the OMAP5912 Product Folder on TI’s website: www.ti.com/omap5912.

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

Contents

Figures

Tables

This page is intentionally left blank.

Timers

Both the OMAP5912 and OMAP5910 application processors have three MPU private timers, three DSP private timers, one MPU watchdog timer, one DSP watchdog timer, and one 32-KHz operating system (OS) timer. The OMAP5912 processor also has one reserved 32-KHz watchdog timer, one 32-KHz synchronization timer, and eight general-purpose (GP) timers.

The MPU and DSP private timers are used for general housekeeping functions inside OMAP5910/5912.

The watchdog timers reset or generate an interrupt to MPU or DSP when they reach timeout. They can be used to detect user programs stuck in an infinite loop, loss of program control, or a runaway condition. In addition, they have general housekeeping capabilities.

The 32-KHz OS timer is able to generate periodical interrupts to the OS. This is used to keep track of the current time to control the operation of the device drivers, and also for OS scheduling purposes. If the OMAP chip is in deep sleep mode, it can also be used to wake up the system.

Using the reserved 32-KHz watchdog timer is not supported. It needs to be disabled upon reset.

The 32-KHz synchronization timer is a simple counter which can be used by the 32-KHz input clock to enable synchronization between devices, but only when the OMAP5912 is used in conjunction with another component (e.g., a modem) having the same clock input.

Each GP timer has the housekeeping capability. In addition, its internal compare logic allows interrupt events on the programmable counter matching value. If it is clocked by the 32-KHz input, it can be used to wake up the chip from deep sleep. GP timers 1, 2, and 3 can also be configured to provide a programmable pulse-width modulation (PWM) output on the dedicated output pin.

1 Introduction

Figure 1 shows all the timers in the OMAP5912. The diagram also applies to the OMAP5910, except that OMAP5910 does not have the reserved 32-KHz watchdog timer, the synchronization timer, or the GP timers.

There are three MPU private timers and an MPU watchdog timer. These are MPU private peripherals and only accessed by the MPU via the MPU private peripheral bus.

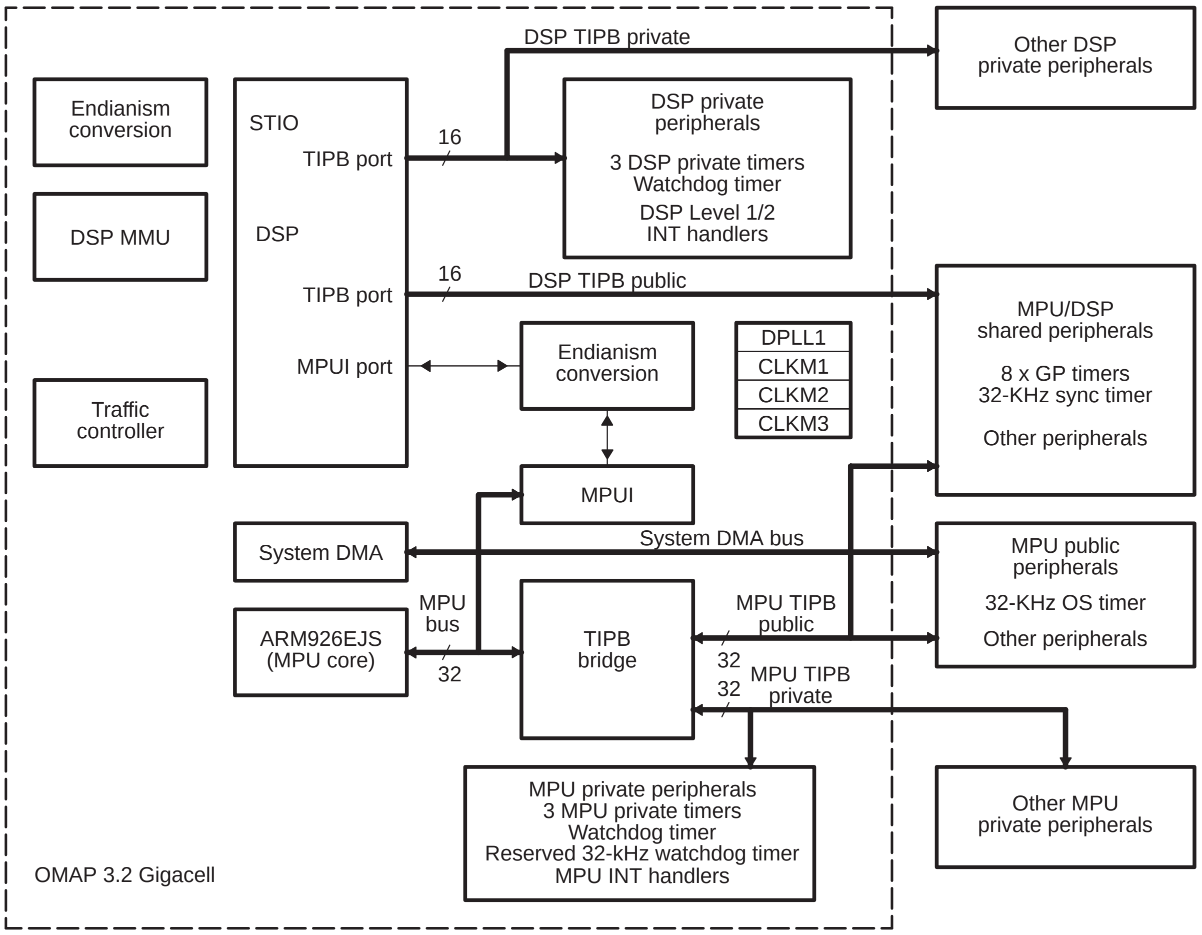
There are three DSP private timers and a DSP watchdog timer. These are DSP private peripherals and are accessed by the DSP via the DSP private peripheral bus.

Eight general-purpose (GP) timers and a 32-KHz synchronization timer are MPU and DSP shared peripherals. They are accessed by both processors via the MPU and DSP public peripheral buses.

The 32-KHz operating system (OS) timer is an MPU public peripheral. It is accessed by the MPU via the MPU public peripheral bus and the system DMA via the system DMA bus.

The system clock (12 MHz, 13 MHz, or 19.2 MHz) is turned off in deep sleep mode, but the 32-KHz clock remains active. As it is active, the 32-KHz OS timer can be used to wake up the chip from deep sleep. If a GP timer is configured to be clocked by the 32-KHz signal, it can also wake up the chip from deep sleep. The MPU watchdog timer also has the wakeup capability, as it resets both the MPU and DSP upon timeout. The DSP watchdog timer only resets the DSP when its counter expires.

Figure 1. OMAP5912 Timers



2 MPU and DSP Private Timers

This section presents the purpose and features of the MPU and DSP private timers in the OMAP5912. This information also applies to the OMAP5910.

There are three MPU private timers and three DSP private timers inside the OMAP5912 that are typically used for general housekeeping functions.

2.1 Introduction to the MPU and DSP Private Timers

2.1.1 Features

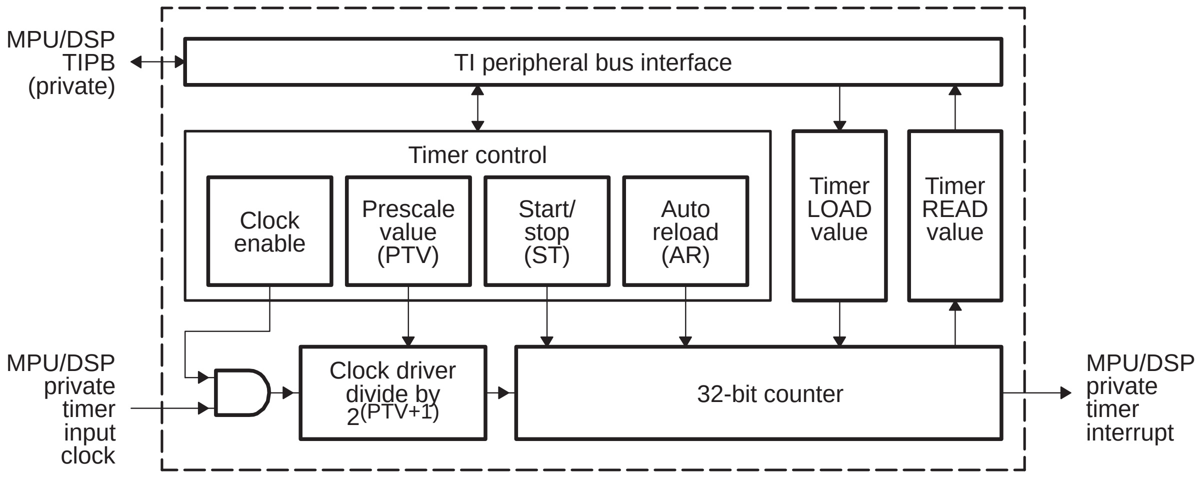
The features of the MPU/DSP private timers are as follows:

- ❑ 32-bit down count
- ❑ Interrupt to MPU/DSP when the timer expires
- ❑ Programmable timer period
- ❑ Auto-reload mode and one-shot mode
- ❑ On-the-fly read capability

2.1.2 Functional Block Diagram

Figure 2 shows the MPU/DSP private timer block diagram. The functionality of the MPU and DSP private timers is identical except that the MPU timers interface to the 32-bit MPU private peripheral bus, and the DSP timers interface to the 16-bit DSP private peripheral bus.

Figure 2. MPU/DSP Private Timer Block Diagram



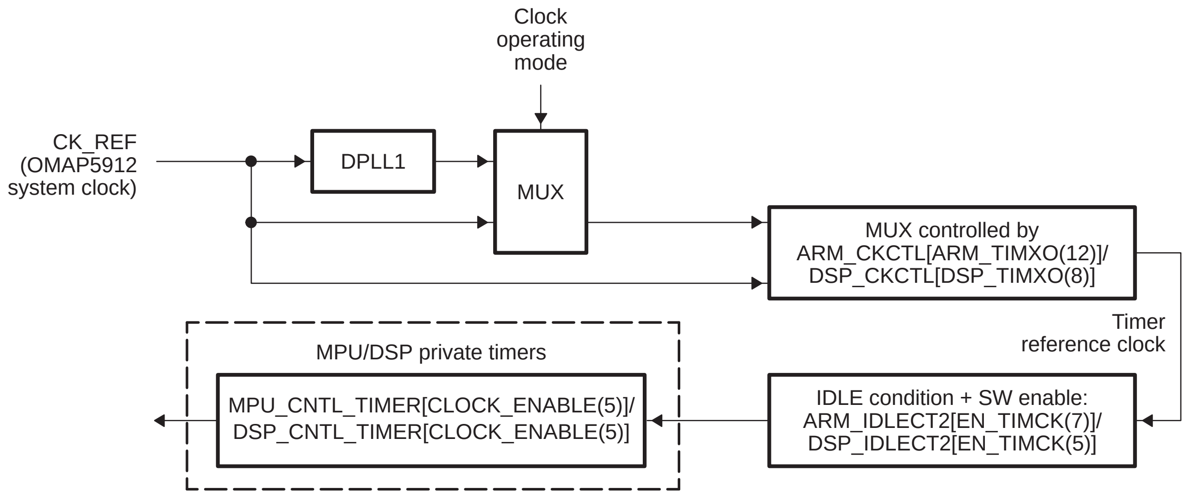
2.2 Common Architecture and Operations

2.2.1 Clock Control

Figure 3 shows the clocking for the MPU and DSP private timers.

CK_REF is the 12-, 13- or 19.2-MHz OMAP system clock (12-, 13- or 19.2-MHz oscillator, or from external clock input).

Figure 3. Clock Input to MPU/DSP Private Timers


Clock operating mode
CK_REF (OMAP5912 system clock)
DPLL1
MUX
MUX controlled by ARM_CKCTL[ARM_TIMXO(12)]/ DSP_CKCTL[DSP_TIMXO(8)]
Timer reference clock
MPU/DSP private timers
MPU_CNTL_TIMER[CLOCK_ENABLE(5)]/ DSP_CNTL_TIMER[CLOCK_ENABLE(5)]
IDLE condition + SW enable: ARM_IDLECT2[EN_TIMCK(7)]/ DSP_IDLECT2[EN_TIMCK(5)]

2.2.1.1 Configuration of the Input Reference Clock

The timer reference clock for the MPU/DSP private timer module is controlled by the OMAP5912 clock generation and reset module. By configuring the ARM_/DSP_TIMXO bit in the ARM_/DSP_CKCTL register of the clock module, you can select from two possible clock sources:

1) ARM_/DSP_TIMXO = 0: Select the OMAP5912 system clock as the timer reference clock for the MPU/DSP private timers.

2) ARM_/DSP_TIMXO = 1: Select the output from the DPLL1 module as the timer reference clock for the MPU/DSP private timers.

The default selection is the output from DPLL1. When operating from DPLL1, the private timer should be stopped before programming a change to the DPLL1 frequency divisor. For detailed information on programming DPLL1, see the Clock Generation and Reset Management section in the *OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide* (SPRU749).

The timer reference clocks can be stopped or activated by configuring the EN_TIMCK bit in the ARM_/DSP_IDLECT2 register.

- ❑ EN_TIMCK = 0: The timer reference clock is stopped.
- ❑ EN_TIMCK = 1: The timer reference clock is activated and can be stopped by configuring the IDLTIM_ARM/_DSP bit in the ARM_/DSP_IDLECTL1 register.
  - ■ IDLTIM_ARM/_DSP = 0: The timer reference clock remains active when the MPU/DSP enters the idle mode.
  - ■ IDLTIM_ARM/_DSP = 1: The timer reference clock is stopped in conjunction with the MPU/DSP clock when the MPU/DSP enters the idle mode.

2.2.1.2 Input Clock Enable

The MPU/DSP private timer reference clock can be disabled with the CLOCK_ENABLE bit in the MPU_/DSP_CNTL_TIMER register.

2.2.2 Interrupt Period

The MPU/DSP private timer generates an interrupt to the MPU/DSP when the counter passes 0. The interrupt period is defined by:

- ❑ The frequency of the timer reference clock for the MPU/DSP private timer (see section 2.2.1).
- ❑ The value of the prescaler bit field (PTV) in the MPU_/DSP_CNTL_TIMER register. The input reference clock to the timer is divided by $2^{(PTV+1)}$. The PTV value is in the range from 0 to 7. Table 1 lists some valid PTV values and the corresponding clock divisor.
- ❑ The values of the load registers: MPU_LOAD_TIMER and DSP_LOAD_TIMER_HI/_LO.

Table 1. Prescaler Value and Clock Divisor for MPU/DSP Private Timer

PTV Bit Field in MPU/DSP Private Timer Control Register	Input Clock Divisor
PTV, (PTV = 0...7)	$2^{(PTV+1)}$
000	2
111	256

The following equations are used to determine the timer interrupt period.

For the MPU private timer:

$$T_{MPU\_Timeout} = T_{MPU\_ref\_clk} \times (\texttt{<MPU\_LOAD\_TIMER>+1}) \times 2^{(PTV+1)}$$

For the DSP private timer:

$$T_{DSP\_Timeout} = T_{DSP\_ref\_clk} \times (<DSP\_LOAD\_TIMER\_HI, LO>+1) \times 2^{(PTV+1)}$$

where $T_{MPU\_ref\_clk}$ and $T_{DSP\_ref\_clk}$ are the periods of the input reference clocks for the MPU and DSP private timers.

Based on these equations, Table 2 calculates examples for various MPU private timer interrupt periods for a hypothetical input reference clock frequency of 100 MHz ($T_{MPU\_ref\_clk}$ = 10 ns).

Table 2. Example of MPU Private Timer Interrupt Periods

MPU_LOAD_TIMER	PTV = 000	PTV = 111
0x0 (granularity)	20 ns†	2.56 µs
0xFFFF FFFF (max period)	85.9 s	10995 s (3 hr 3 min 15 s)

† Although the timer can be programmed to generate 20ns periodical interrupts, this is not recommended, as the TIPB bus and the ARM interrupt handling mechanism are not fast enough to handle those interrupts.

2.2.3 Power Management

Before the MPU/DSP subsystem can enter any of the low power states, the MPU/DSP timer reference clock must be stopped in the following two ways:

- ❑ Set the EN_TIMCK bit in the MPU_/DSP_IDLECT2 register to 0 to force the clock to stop.
- ❑ Leave the EN_TIMCK bit in the MPU_/DSP_IDLECT2 register as 1, but stop all the MPU/DSP private timers by configuring the ST bit in the MPU_/DSP_CNTL_TIMER register and set the IDLTIM_ARM/_DSP bit to 1. Setting the IDLTIM_ARM/_DSP bit to 1 causes the input reference clock to be stopped in conjunction with MPU/DSP clock when the idle mode is entered.

For details on OMAP5912 power management, see the *OMAP5912 Multimedia Processor Power Management Reference Guide* (SPRU753).

2.2.4 Reset Considerations

Upon reset, the MPU/DSP timer module is in the following state:

- ❑ The input reference clock is from DPLL1 (ARM_CKCTL[ARM_TIMXO]/ DSP_CKCTL[DSP_TIMXO] = 1).
- ❑ The input reference clock is stopped (EN_TIMCK bit is 0 in the ARM_/DSP_IDLECT2 register).

- ❑ The input clock is not enabled inside timer module (CLOCK_ENABLE bit is 0 in the MPU_/DSP_CNTL_TIMER register).
- ❑ The timer is in one-shot mode.
- ❑ The timer is stopped (ST bit is 0 in the MPU_/DSP_CNTL_TIMER register).

2.2.5 Interrupt Support

An interrupt is generated to the MPU/DSP when an MPU/DSP private timer decrements to 0. Table 3 shows the interrupt number used by each MPU/DSP private timer.

Table 3. MPU/DSP Private Timer Interrupt Number

Timer	Interrupt Number
MPU private timer 1	MPU level 1 interrupt #26
MPU private timer 2	MPU level 1 interrupt #30
MPU private timer 3	MPU level 1 interrupt #16
DSP private timer 1	DSP level 1 interrupt #23
DSP private timer 2	DSP level 1 interrupt #22
DSP private timer 3	DSP level 1 interrupt #8

All interrupts are falling edge sensitive.

For details about OMAP5912 interrupt handling, see the *OMAP5912 Multimedia Processor Interrupts Reference Guide* (SPRU757).

2.2.6 One-Shot Mode versus Auto-Reload Mode

An MPU/DSP private timer works in one-shot mode if the AR(1) bit in the MPU_/DSP_CNTL_TIMER register is 0. In this mode, when the timer counter expires an interrupt is generated and the timer is stopped. The ST(0) bit is automatically reset by the internal logic.

The MPU/DSP private timer works in auto-reload mode if the AR(1) bit is 1. In this mode, when the timer counter expires an interrupt is generated and the new count value is loaded from the MPU_LOAD_TIMER register or the DSP_LOAD_TIMER_HI/_LO register. The timer continues counting down from the new loaded count value.

2.2.7 Initialization

The following procedure is an example of MPU/DSP private timer initialization.

1) Configure the timer input reference clock.
   a) Program the ARM_/DSP_TIMXO bit in the ARM_/DSP_CKCTL register to select the reference for the timer clock (1: DPLL1 is selected; 0: OMAP5912 system clock is selected). If DPLL1 is selected, the DPLL1 needs to be programmed.
   b) Set the EN_TIM bit in the ARM_/DSP_IDLECT2 register to activate the timer reference clock.
2) Configure the interrupt controller module. For the MPU private timer:
   a) Enable the MPU global interrupt.
   b) Program the corresponding level 1 interrupt priority_level (ILR) register for the timer interrupt being used. For the MPU private timers 1, 2, and 3, it is ILR26, ILR30, and ILR16 respectively.
      i) Program the PRIORITY field to set the interrupt priority (0 highest; 31 lowest).
      ii) Set the SENS_LEVEL bit to 0 as falling edge sensitive.
      iii) Set the FIQ bit to 1 to set the interrupt as a FIQ.
   c) Prepare the FIQ ISR and place its address to the appropriate entry of the interrupt vector (entry 7 in the vector).
   d) Prepare the timer ISR.
   e) Enable the specific timer interrupt by clearing the corresponding bit in the MPU interrupt level 1 mask register.
3) Configure the interrupt controller module. For the DSP private timer:
   a) Enable the DSP global interrupt.
   b) Prepare the timer ISR and put its address in the appropriate entry of the interrupt vector (based on the interrupt number being used).
   c) Enable the specific timer interrupt by clearing the corresponding bit in the DSP interrupt level 1 mask register.

2.2.8 Common Operations

2.2.8.1 *Start/Stop an MPU/DSP Private Timer*

Setting the ST bit in the MPU_/DSP_CNTL_TIMER register starts the MPU/DSP private timer. At start, the timer loads the values in the MPU_LOAD_TIMER register or the DSP_LOAD_TIMER_HI and DSP_LOAD_TIMER_LO registers into the counter.

Clearing the ST bit stops the MPU/DSP private timer. When stopped, the timer keeps the current value.

Note:

Only the ST and EN_TIMCLK bits in the MPU_/DSP_CNTL_TIMER register can be written while the timer is running.

Undefined results occur if following configurations are changed while the timer is running:

- ❑ Prescaler (PTV) and auto-reload (AR) bits in the MPU_/DSP_CNTL_TIMER register
- ❑ MPU_LOAD_TIMER register
- ❑ DSP_LOAD_TIMER_HI/_LO registers

2.2.8.2 *Read Timer Count Value*

The timer value can be read from the MPU_/DSP_READ_TIMER register either on-the-fly or after the timer is stopped.

For the MPU, the MPU_READ_TIMER is a 32-bit register and can be read directly.

However, the DSP peripheral bus only has a width of 16 bits. To correctly read the value of the DSP private timers, the upper 16 bits must be read prior to the lower 16 bits. When the upper read occurs, the lower 16 bits are simultaneously stored in a temporary register. The contents of this temporary register are provided by an access to the lower 16 bits.

2.2.9 Emulation Considerations

The FREE bit in the MPU_/DSP_CNTL_TIMER register defines the behavior of the MPU/DSP private timer when the debugger halts its parent processor.

- ❑ FREE = 1: Keep running while the debugger halts its processor.
- ❑ FREE = 0: Stop the timer while the debugger halts its processor (even if the ST bit in the timer control register is set).

2.2.10 Pseudo Code Example

The pseudo code below shows how to use an MPU/DSP private timer.

1) Initialization.
2) Program the timer to the desired interrupt period and mode.
    a) Program the MPU_/DSP_LOAD_TIMER register to set the load value.
    b) Program the PTV field in the MPU_/DSP_CNTL_TIMER register to set the clock divider value (PTV = 0 ... 7).
    c) Program the AR bit in the MPU_/DSP_CNTL_TIMER register. (1: auto-reload mode; 0: one-shot mode)
    d) Program the FREE bit in the MPU_/DSP_CNTL_TIMER register. (1: run free during emulation stop; 0: stop during emulation stop)
    e) Load the desired counter value to the MPU_LOAD_TIMER register or DSP_LOAD_TIMER_HI/_LO register.
    f) Set the CLOCK_ENABLE bit in the MPU_/DSP_CNTL_TIMER register to enable the input clock.
3) Start the timer by setting the ST bit in the MPU_CNTL_TIMER register and wait until an interrupt happens.
4) When an interrupt happens, control jumps depending on the timer:
    a) For the MPU private timer, control jumps to the FIQ ISR, which should:
        i) Read the interrupt level 1 source register for FIQ to get the interrupt number, which should be the timer interrupt number.
        ii) Call the timer ISR to process the interrupt.
        iii) Set the NEW_FIQ_ARG bit in the interrupt level 1 control register to acknowledge the interrupt, which will allow new FIQ generation.
    b) For the DSP private timer, control jumps the timer ISR in which the interrupt is processed.

2.3 MPU Private Timer Registers

Table 4 lists the register base addresses of the MPU private timers. Table 5 lists the registers. All the registers are 32 bits. Table 6 through Table 8 are the descriptions for each register.

Table 4. Register Base Addresses of MPU Private Timers

MPU Private Timer	MPU Byte Address
Timer 1	0xFFFE: C500
Timer 2	0xFFFE: C600
Timer 3	0xFFFE: C700

Table 5. MPU Private Timer Registers

MPU Byte Base Address = 0xFFFE C500, 0xFFFE C600, 0xFFFE C700			
Name	**Description**	**R/W**	**Offset**
MPU_CNTL_TIMER	MPU control timer	R/W	0x00
MPU_LOAD_TIMER	MPU load timer	W	0x04
MPU_READ_TIMER	MPU read timer	R	0x08

Table 6. MPU Control Timer Register (MPU_CNTL_TIMER)

MPU Byte Base Address = 0xFFFE C500, 0xFFFE C600, 0xFFFE C700, Byte Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:7	RESERVED	Reserved.		0x0000000
6	FREE	Defines the behavior of the MPU private timer when the debugger halts its parent processor. 0: Stop counting even if ST =1. 1: Keep counting if ST=1.	R/W	0
5	CLOCK_ENABLE	Enable input reference clock to the MPU private timer module 0: Disable. 1: Enable.	R/W	0
4:2	PTV	Prescale timer input reference clock Clock divisor = $2^{(PTV+1)}$	R/W	000

Table 6. MPU Control Timer Register (MPU_CNTL_TIMER) (Continued)

MPU Byte Base Address = 0xFFFE C500, 0xFFFE C600, 0xFFFE C700, Byte Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	AR	0: One-shot mode. 1: Auto-reload mode.	R/W	0
0	ST	0: Stop timer value decrement. 1: Start timer value decrement. In one-shot mode (AR = 0), this bit is cleared automatically when the timer value expires.	R/W	0

Table 7. MPU Load Timer Register Value (MPU_LOAD_TIMER)

MPU Byte Base Address = 0xFFFE C500, 0xFFFE C600, 0xFFFE C700, Byte Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	MPU_LOAD_TIMER	This value is loaded when the timer expires or when it starts.	W	Undefined

Table 8. MPU Read Timer Register Value (MPU_READ_TIMER)

MPU Byte Base Address = 0xFFFE C500, 0xFFFE C600, 0xFFFE C700, Byte Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	MPU_READ_TIMER	Value of the timer.	R	Undefined

2.4 DSP Private Timer Registers

Table 9 lists the register base addresses of the DSP private timers. Table 10 lists the registers. Table 11 through Table 15 are the descriptions for each register.

All the registers are 16 bits except the DSP_CNTL_TIMER register, which is 32 bits.

Table 9. Base Addresses of DSP Private Timer Registers

DSP Private Timer	DSP Word Address (I/O Space)
Timer 1	0x2800
Timer 2	0x2C00
Timer 3	0x3000

Table 10. DSP Private Timer Registers

Name	Description	R/W	DSP Word Offset
DSP_CNTL_TIMER	DSP control timer	R/W	0x00
DSP_LOAD_TIMER_LO	DSP load timer value, 1/2 LSW	W	0x03
DSP_LOAD_TIMER_HI	DSP load timer value, 1/2 MSW	W	0x02
DSP_READ_TIMER_LO	DSP read timer value, 1/2 LSW	R	0x05
DSP_READ_TIMER_HI	DSP read timer value, 1/2 MSW	R	0x04

Table 11. DSP Control Timer Register (DSP_CNTL_TIMER)

DSP Word Base Address = 0x2800, 0x2C00, 0x3000, Word Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:7	RESERVED	Reserved.		0x000
6	FREE	Defines the behavior of the DSP private timer when the debugger halts its parent processor.	R/W	0
		0: Stop counting even if ST =1.		
		1: Keep counting if ST=1.		
5	CLOCK_ENABLE	Enables input reference clock to the DSP OS timer module	R/W	0
		0: Disable.		
		1: Enable.		
4:2	PTV	Prescale timer input reference clock	R/W	000
		Clock divisor = $2^{(PTV+1)}$		
1	AR	0: One-shot mode.	R/W	0
		1: Auto-reload mode.		
0	ST	0: Stop timer value decrement.	R/W	0
		1: Start timer value decrement.		
		In one-shot mode (AR = 0), this bit is cleared automatically when the timer expires.		

Table 12. DSP Load Timer Register Low (DSP_LOAD_TIMER_LO)

DSP Word Base Address = 0x2800, 0x2C00, 0x3000, Word Offset = 0x03				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSP_LOAD_TIMER_LO	This value is loaded when the timer expires or when it starts.	W	Undefined

Table 13. DSP Load Timer Register High (DSP_LOAD_TIMER_HI)

DSP Word Base Address = 0x2800, 0x2C00, 0x3000, Word Offset = 0x02				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSP_LOAD_TIMER_HI	This value is loaded when the timer expires or when it starts.	W	Undefined

Table 14. DSP Read Timer Register Low (DSP_READ_TIMER_LO)

DSP Word Base Address = 0x2800, 0x2C00, 0x3000, Word Offset = 0x05				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSP_READ_TIMER_LO	Value of the timer bits (15:0): To read a correct value for DSP private timer, the upper 16 bits (from DSP_READ_TIMER_HI) must read prior to reading this register.	R	Undefined

Table 15. DSP Read Timer Register High (DSP_READ_TIMER_HI)

DSP Word Base Address = 0x2800, 0x2C00, 0x3000, Word Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSP_READ_TIMER_HI	Value of the timer bits (31:16): To read correct value for DSP private timer, this register must be read first, followed by lower 16 bits of DSP_READ_TIMER_LO.	R	Undefined

3 Watchdog Timers

3.1 Introduction

There are three watchdog timers inside the OMAP5912: one MPU watchdog timer, one DSP watchdog timer, and one reserved 32-KHz watchdog timer.

Using the 32-KHz watchdog is not supported. Upon system reset, it needs to be disabled before it expires (in approximately 19s) via the following procedure:

```
(*(volatile int*)0xFFFEB048) = 0xAAAA;
// wait until the write operation is completed
while ( ((*(volatile int *)0xFFFEB034) & 0x10) == 0x10 );
(*(volatile int*)0xFFFEB048) = 0x5555;
// wait until the write operation is completed
while ( ((*(volatile int *)0xFFFEB034) & 0x10) == 0x10 );
```

This section presents the purpose and features of the MPU and DSP watchdog timers in the OMAP5912. All the information also applies to the OMAP5910.

3.1.1 Purpose

There are two useful watchdog timers inside the OMAP5912. One is in the MPU subsystem (controlled by the MPU), and the other is in the DSP subsystem (controlled by the DSP). Each watchdog timer can be configured either as a watchdog timer or a housekeeping timer.

When the timer is configured as a watchdog timer, the software must periodically write to the count register before the counter expires. If the counter expires, the MPU watchdog timer resets the OMAP3.2 gigacell except the DPLL1, while the DSP watchdog timer resets the DSP only. The watchdog timers can be used to detect user programs that are stuck in an infinite loop, have a loss of program control, or are in a runaway condition.

When used as a housekeeping timer, it is 16-bit counter configurable either to auto-reload mode or one-shot mode with on-the-fly read capability. It generates an interrupt to the MPU/DSP when the count expires.

3.1.2 Features

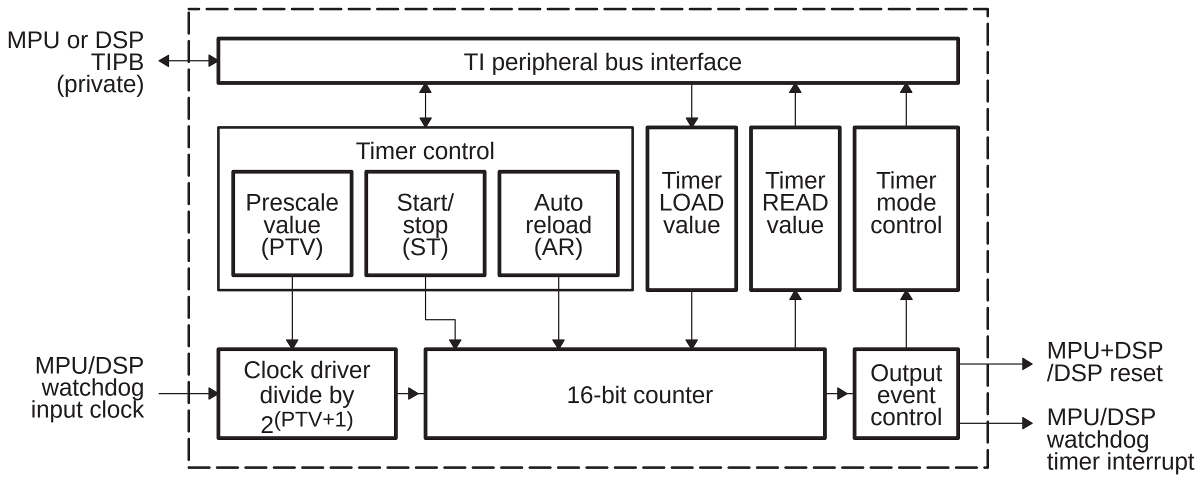
The following are features of the MPU/DSP watchdog timers:

- 16 bit count down
- Possesses on-the-fly read capability
- Fixed low input clock frequency (system_clock/14)
- Programmable reset or interrupt period
- If configured as a watchdog timer, the MPU watchdog timer resets both the MPU and DSP when the counter passes 0, while the DSP watchdog timer resets the DSP only
- If configured as a housekeeping timer:
  - Supports both auto-reload and one-shot mode
  - Generates an interrupt to MPU/DSP when counter passes 0

3.1.3 Functional Block Diagram

Both watchdog timers have identical functionality, except that the MPU timer interfaces to the 32-bit MPU private peripheral bus, and the DSP timer interfaces to the 16-bit DSP private peripheral bus.

Figure 4. Watchdog Timer Block Diagram



3.1.4 Supported Modes

A watchdog timer can be configured as either a watchdog timer or a housekeeping timer.

Both the MPU and DSP watchdog timers are configured for watchdog mode after reset. For specific watchdog mode information, see section 3.3.

Both MPU and DSP watchdog timers can also be configured to work as 16-bit housekeeping timers, supporting both auto-reload and one-shot mode with on-the-fly read capability. For specific housekeeping mode information, see section 3.4.

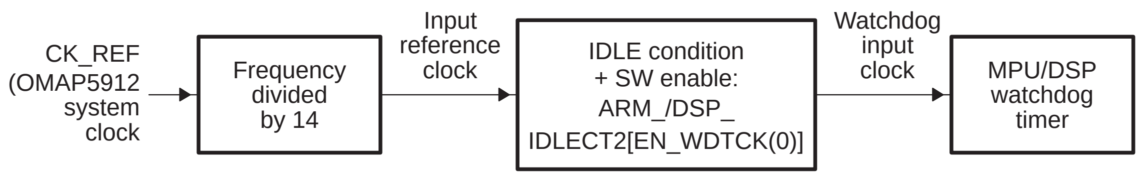
3.2 Common Architecture and Operations

3.2.1 Clock Control

Figure 5 shows the clocking for the MPU and DSP watchdog timers.

CK_REF is the 12-, 13- or 19.2-MHz system clock (12-, 13- or 19.2-MHz oscillator, or from external clock input).

Figure 5. Clock Input to Watchdog Timers


CK_REF (OMAP5912 system clock)
Frequency divided by 14
Input reference clock
IDLE condition + SW enable: ARM_/DSP_ IDLECT2[EN_WDTCK(0)]
Watchdog input clock
MPU/DSP watchdog timer

3.2.1.1 Configuration of the Input Reference Clock for Watchdog Timers

The input reference clocks for the watchdog timer modules are derived from OMAP5912 system clock, divided by 14. For example, if the system clock is 12 MHz, the watchdog input reference clock will be about 0.86 MHz.

If the MPU/DSP watchdog timer is configured for watchdog mode, the input reference clock is always active, regardless of the EN_WDTCK bit value in the MPU_/DSP_IDLECT2 register.

If the watchdog timer is configured for housekeeping mode, the input clock can be stopped or activated by configuring the EN_WDTCK bit in the MPU_/DSP_IDLECT2 register.

- ❑ EN_WDTCK = 0: The input reference clock is stopped.
- ❑ EN_WDTCK = 1: The input reference clock is activated and can be stopped by configuring the IDLWDT_ARM/_DSP bit in the ARM_/DSP_IDLECTL1 register.
  - ■ IDLWDT_ARM/_DSP = 0: The input reference clock remains active when the MPU/DSP enters the idle mode.
  - ■ IDLWDT_ARM/_DSP = 1: The input reference clock is stopped in conjunction with the MPU/DSP clock when the MPU/DSP enters the idle mode.

3.2.2 Timeout Period

The timeout period of an MPU/DSP watchdog timer is defined by:

- ❑ The frequency of the input reference clock, which is the OMAP system clock frequency divided by 14.
- ❑ The value of the prescaler bit field (PTV) in the MPU_/DSP_WDT_CNTL register. The input clock frequency is further divided by $2^{(PTV+1)}$. When the watchdog timer is in watchdog mode, the PTV value is fixed to 7. When the watchdog is configured as a housekeeping timer, PTV is in the range 0 to 7.
- ❑ The value of the load register, MPU_/DSP_WDT_LOAD.

The following equation is used to determine the timeout period.

$$T_{WDT\_Timeout} = T_{WDT\_ref\_clk} \times (\text{<MPU\_/DSP\_WDT\_LOAD>}+1) \times 2^{(PTV+1)}$$

where $T_{WDT\_ref\_clk}$ is the input reference clock period of the watchdog timer.

Table 16 shows the characteristics of the watchdog timer for different MPU_/DSP_WDT_LOAD values.

Table 16. Watchdog Timer Characteristics

OMAP System Clock	WDT_ref_clk	MPU_/DSP_WDT_LOAD	$T_{WDT\_Timeout}$(PTV=7)
12 MHz	1167 ns	0x0001	597.34 µs
12 MHz	1167 ns	0xFFFF	19.57 s

3.2.3 Power Management

Before the MPU/DSP subsystem enters any of the low power states, the corresponding input reference clock to the watchdog timer must be stopped, as it implies that the watchdog timer must be configured for housekeeping mode.

There are two ways to stop the input reference clock when the MPU/DSP watchdog timer is in housekeeping mode:

- ❏ Set the EN_WDTCK bit in the MPU_/DSP_IDLECT2 register to 0 to force the clock to stop.
- ❏ Set the EN_WDTCK bit in the MPU_/DSP_IDLECT2 register to 1, but stop the MPU/DSP watchdog timer by setting the ST bit in the MPU_/DSP_WDT_CNTL register, and program the IDLWDT_ARM/_DSP bit to 1. This causes the input reference clock to be stopped automatically in conjunction with the MPU/DSP clock when the idle mode is entered.

For details on OMAP5912 power management, see the *OMAP5912 Multimedia Processor Power Management Reference Guide* (SPRU753).

3.2.4 Reset Considerations

Both the MPU and DSP watchdog timers are configured for watchdog mode after powerup. The default value of the MPU_/DSP_WDT_LOAD register is 0xFFFF.

When the MPU watchdog expires, it resets the OMAP3.2 gigacell, except for the DPLL1. When the DSP watchdog expires, it resets only the DSP.

If the OMAP system clock is 12 MHz, the reset occurs in approximately 19 seconds upon powerup, unless the default setting is changed.

3.2.5 Interrupt Support

If the watchdog is configured as a housekeeping timer, the timer can be configured to generate an interrupt when the counter expires.

MPU watchdog timer uses MPU level 1 interrupt #27. DSP watchdog timer uses DSP level 1 interrupt #13. Both interrupts are falling edge sensitive.

For details about OMAP5912 interrupt handling, see the *OMAP5912 Multimedia Processor Interrupts Reference Guide* (SPRU757).

3.2.6 Common Operations

3.2.6.1 Read Timer Values

The timer values can be read from the respective MPU or DSP read timer register (MPU_/DSP_WDT_READ) either on-the-fly or after the timer is stopped.

3.2.7 Emulation Considerations

The FREE bit in the MPU_/DSP_WDT_CNTL_TIMER register defines the MPU/DSP watchdog behavior in housekeeping mode when the debugger halts its parent processor.

❑ FREE = 1: Keep running while the debugger halts its processor.

❑ FREE = 0: Stop the timer while the debugger halts its processor, even if the ST bit in the timer control register is set.

If the MPU/DSP watchdog timer is configured for watchdog mode, it keeps counting until it expires, regardless of this bit.

3.3 Watchdog Mode Operations

3.3.1 Initialization

On power up, the MPU/DSP watchdog timer defaults to watchdog mode.

When the timer is in housekeeping mode, it can be switched back to watchdog mode by setting the WATCHDOG bit in the MPU_/DSP_WDT_TIMER_MODE register. In this case, the value loaded into MPU_/DSP_WDT_LOAD register is set to the maximum value (0xFFFF), as on power up.

Note:

In watchdog mode, the MPU_/DSP_WDT_CNTL register must not be used. The watchdog timer cannot be stopped by clearing the ST bit. The prescale (PTV) value is fixed to 7 regardless of the PTV field value. Auto-reload and one-shot do not apply because, if the counter expires, the processor is reset and the watchdog registers are reinitialized.

3.3.2 Programming the Watchdog Timer in Watchdog Mode

When used as a watchdog timer, it is the responsibility of the software to periodically write to the MPU_/DSP_WDT_LOAD register before the counter expires. The newly loaded value must be different from the previous value, otherwise the write will not be taken into account. Due to internal sequencing, three timer clock periods must pass before a new value can be written into the register. If the OMAP system clock (CLK_REF) is 12 MHz, the duration of three timer clock periods is approximately 3.5 μs.

3.3.3 Pseudo Code Example

The following procedure shows how to use the watchdog timer in watchdog mode.

1) Arrange a routine which is executed periodically before the WDT expires. For example, this routine can be launched by the MPU/DSP private timers or GP timers.

2) If the watchdog timer is in housekeeping mode, start the watchdog timer by setting the WATCHDOG bit in the MPU_/DSP_WDT_TIMER_MODE register.

3) Wait until an interrupt occurs before the watchdog times out. Inside the ISR, reload the MPU_/DSP_WDT_LOAD register with a different value from the current one.

3.4 Housekeeping Mode Operations

3.4.1 Initialization

On power up, the MPU/DSP watchdog timer defaults to watchdog mode. By writing a predefined sequence to the WATCHDOG_DIS field of the MPU_/DSP_WDT_TIMER_MODE register, the timer can be switched to housekeeping mode. A sequence decode is initialized when 0xF5 is written. Once in this state, if the next write is different from 0xA0, the state machine causes a reset as if the watchdog timer has timed out. The watchdog timer cannot be disabled by simply clearing the WATCHDOG bit in the MPU_/DSP_WDT_TIMER_MODE register.

3.4.2 Programming the Watchdog Timer in Housekeeping Mode

The timer in housekeeping mode is started by setting the ST bit in the control timer register (MPU_/DSP_WDT_CNTL) to 1. It is stopped by resetting this bit to 0. When the timer is stopped, the timer counter keeps the current value.

If the AR bit in the MPU_/DSP_WDT_CNTL_TIMER register is 0, the timer decrements from the loaded value down to 0 and then stops. If the AR bit is 1, the timer continues. A new value from the MPU_/DSP_WDT_LOAD register is loaded into the timer counter when it expires.

Note:

Only the ST bit in the MPU_/DSP_WDT_CNTL register can be written while the timer is running. Undefined results occur if the PTV or AR bits in the MPU_/DSP_WDT_CNTL register are written. Undefined results occur if the MPU_/DSP_WDT_LOAD register is written while the timer is running.

3.4.3 Pseudo Code Example

The following procedure shows how to switch a watchdog timer into housekeeping mode and how to use the timer.

1) Switch the timer to housekeeping mode, if necessary:
    a) Write 0xF5 to the MPU_/DSP_WDT_TIMER_MODE register.
    b) Write 0xA0 to the MPU_/DSP_WDT_TIMER_MODE register.
2) Enable the watchdog timer input clock by setting the EN_WDTCK bit in the ARM_/DSP_IDLECT2 register.
3) Configure the timer:
    a) Program the MPU_/DSP_WDT_LOAD_TIMER register to the desired load value.
    b) Program the PTV field in the MPU_/DSP_WDT_CNTL_TIMER register to set the clock divider value (PTV = 0 ... 7).
    c) Program the AR bit in the MPU_/DSP_WDT_CNTL_TIMER register. (1: auto-reload mode; 0: one-shot mode)
    d) Program the FREE bit in the MPU_/DSP_WDT_CNTL_TIMER register. (1: run free during emulation stop; 0: stop during emulation stop)
4) Configure the MPU interrupt controller module for the MPU watchdog timer:
    a) Enable the MPU global interrupt.
    b) Program the corresponding level 1 interrupt priority_level (ILR) register for the timer interrupt being used. (For the MPU watchdog timer, ILR27.)
        i) Program the PRIORITY field to set the interrupt priority (0 highest; 31 lowest).
        ii) Set the SENS_LEVEL bit to 0 as falling edge sensitive.
        iii) Set the FIQ bit to 1 to set the interrupt as a FIQ.
    c) Prepare the FIQ ISR and place its address the appropriate entry of the interrupt vector (entry 7 in the vector).
    d) Prepare the watchdog timer ISR.
    e) Enable the specific timer interrupt by clearing the corresponding bit in the MPU level 1 interrupt mask register.
5) Configure the DSP interrupt controller module for DSP watchdog timer:
    a) Enable the DSP global interrupt.
    b) Prepare the watchdog timer ISR and put its address in entry 13 of the interrupt vector.
    c) Enable the timer interrupt by clearing the corresponding bit the in the DSP level 1 interrupt mask register.

6) When the interrupt occurs:
   a) For the MPU watchdog timer, control jumps to the FIQ ISR, which will:
      i) Read the level 1 interrupt source register for FIQ to get the interrupt number (which should be interrupt #27).
      ii) Call the timer ISR to process the interrupt.
      iii) Set the NEW_FIQ_ARG bit in the interrupt level 1 control register to acknowledge the interrupt.
   b) For the DSP watchdog timer, control jumps to the watchdog timer ISR in which the interrupt is processed.

3.5 MPU Watchdog Timer Registers

Table 17 lists the MPU watchdog timer registers. All the registers are 16 bits. The base (byte) address is 0xFFFE C800. Table 18 through Table 21 provide the descriptions for each register.

Table 17. MPU Watchdog Timer Registers

Name	Description	R/W	MPU (Byte) Address
MPU_WDT_CNTL_TIMER	MPU watchdog control timer	R/W	0xFFFE C800
MPU_WDT_LOAD_TIMER	MPU watchdog load timer	W	0xFFFE C804
MPU_WDT_READ_TIMER	MPU watchdog read timer value	R	0xFFFE C804
MPU_WDT_TIMER_MODE	MPU watchdog timer mode	R/W	0xFFFE C808

Table 18. MPU Watchdog Control Timer Register (MPU_WDT_CNTL_TIMER)

MPU Byte Address = 0xFFFE C800, Byte Offset 0x00				
Bits	**Name**	**Description**	**R/W**	**Reset Value**
15-12	RESERVED	Reserved.	R/W	
11-9	PTV	Prescale clock timer value Clock divisor = $2^{(PTV+1)}$ if the timer is in house-keeping mode. Clock divisor is fixed 256 if the timer is in watch-dog mode.	R/W	0
8	AR	Auto-reload/One-shot timer when in housekeep-ing mode. 0: One-shot timer. 1: Auto-reload timer.	R/W	0

Table 18. MPU Watchdog Control Timer Register (MPU_WDT_CNTL_TIMER) (Continued)

Bits	Name	Description	R/W	Reset Value
7	ST	Start/Stop timer when in housekeeping mode 0: Stop the timer. 1: Start the timer.	R/W	0
6-2	RESERVED	Reserved.		
1	FREE	It defines the behavior of the MPU watchdog timer in housekeeping mode when the debugger halts its parent processor. 0: Stop counting even if ST =1. 1: Keep counting if ST=1. The MPU watchdog timer keeps counting in watchdog mode, regardless of this bit value.	R/W	1
0	RESERVED	Reserved.		

Table 19. MPU Watchdog Load Timer Register (MPU_WDT_LOAD_TIMER)

MPU Byte Base Address = 0xFFFE C800, Offset = 0x04

Bit	Name	Description	R/W	Reset Value
15-0	MPU_WDT_ LOAD_TIMER	Housekeeping mode: This value is loaded when the timer expires or when it starts. Watchdog mode: Reload the timer with this value.	W	0xFFFF

Table 20. MPU Watchdog Read Timer Register (MPU_WDT_READ_TIMER)

MPU Byte Base Address = 0xFFFE C800, Offset = 0x04

Bit	Name	Description	R/W	Reset Value
15-0	MPU_WDT_ READ_TIMER	Current timer value.	R	0xFFFF

Table 21. MPU Watchdog Timer Mode Register (MPU_WDT_TIMER_MODE)

MPU Byte Base Address = 0xFFFE C800, Offset = 0x08				
Bit	**Name**	**Description**	**R/W**	**Reset Value**
15	WATCHDOG	Write access 1: Switch the timer mode back to watchdog mode. 0: No effect.	W	1
14-8	RESERVED	Reserved.		
7-0	WATCHDOG_DIS	Write access only Writing a predefined sequence (0xF5, followed by 0xA0) in this field switches the timer from watchdog mode to housekeeping mode. After receiving 0xF5, if the second write access is different from 0xA0, the MPU and DSP cores are reset.	W	0

3.6 DSP Watchdog Timer Registers

Table 22 shows the DSP watchdog timer registers. All of the registers are I/O space mapped. Table 23 through Table 26 describe the register bits.

Table 22. DSP Watchdog Timer Registers

Register Name	Description	R/W	DSP (Word) Address (I/O Space)
DSP_WDT_CNTL_TIMER	DSP watchdog control timer	R/W	0x3400
DSP_WDT_LOAD_TIMER	DSP watchdog load timer	W	0x3402
DSP_WDT_READ_TIMER	DSP watchdog read timer value	R	0x3402
DSP_WDT_TIMER_MODE	DSP watchdog timer mode	R/W	0x3404

Table 23. DSP Watchdog Control Timer Register (DSP_WDT_CNTL_TIMER)

DSP Word Base Address = 0x3400, Word Offset = 0x00

Bit	Name	Description	R/W	Reset Value
15-12	Reserved	Reserved.		
11-9	PTV	Prescale clock timer value Clock divisor = $2^{(PTV+1)}$ in housekeeping mode. Clock divisor is fixed 256 in watchdog mode.	R/W	0
8	AR	Auto-reload/One-shot timer when in housekeeping mode 0: One-shot timer. 1: Auto-reload timer.	R/W	0
7	ST	Start/Stop the timer when in housekeeping mode 0: Stop the timer. 1: Start the timer.	R/W	0
6-2		Reserved.		
1	FREE	It defines the behavior of the DSP watchdog timer in housekeeping mode when the debugger halts its parent processor. 0: Stop counting even if ST =1. 1: Keep counting if ST=1. The DSP watchdog timer keeps counting in watchdog mode, regardless of this bit value.	R/W	1
0		Reserved.		

Table 24. DSP Watchdog Load Timer Register (DSP_WDT_LOAD_TIMER)

DSP Word Base Address = 0x3400, Word Offset = 0x02

Bit	Name	Description	R/W	Reset Value
15-0	DSP_WDT_ LOAD_TIMER	Housekeeping mode. This value is loaded when the timer expires or when it starts. Watchdog mode. Reload the timer with this value.	W	0xFFFF

Table 25. DSP Watchdog Read Timer Register (DSP_WDT_READ_TIMER)

DSP Word Base Address = 0x3400, Word Offset = 0x02				
Bit	**Name**	**Description**	**R/W**	**Reset Value**
15-0	DSP_WDT_ READ_TIMER	Read the timer value.	R	0xFFFF

Table 26. DSP Watchdog Timer Mode Register (DSP_WDT_TIMER_MODE)

DSP Word Base Address = 0x3400, Word Offset = 0x04				
Bit	**Name**	**Description**	**R/W**	**Reset Value**
15	WATCHDOG	Write access: 1: Switch back from timer mode to watchdog mode. 0: No effect. Read access (status of timer mode): 0: General housekeeping timer. 1: Watchdog timer.	W	1
14-8	RESERVED	Reserved.		
7-0	WATCHDOG_DIS	Write access only: Writing a predefined sequence (0xF5, followed by 0xA0) in this field switches from watchdog mode to housekeeping mode. After receiving 0xF5, if the second write access is different from 0xA0, the DSP core is reset.	W	1

4 32-KHz OS Timer

4.1 Introduction

This section presents the purpose and features of the 32-KHz OS timer in the OMAP5912. This information also applies to the OMAP5910.

4.1.1 Purpose

The OS on the MPU subsystem requires interrupts at regular time intervals to keep track of the current time to control the operation of the device drivers and also for OS scheduling purposes. The OS time interval can be from 1 ms to 30 ms.

For example, Microsoft Windows CE scheduling has the following requirements:

❑ The periodic interrupt occurs every 1-25 ms

❑ The timer is expected to run in all modes except when suspended

The MPU private timers run with the system clock (CLK_REF) or DPLL1, which may be disabled during the sleep mode. The 32-KHz OS timer runs with 32-KHz clock, and can keep running during the sleep mode. Therefore, it is able to provide the required OS timing interval in all circumstances.

Note:

32-KHz here refers to 32678, not 32000.

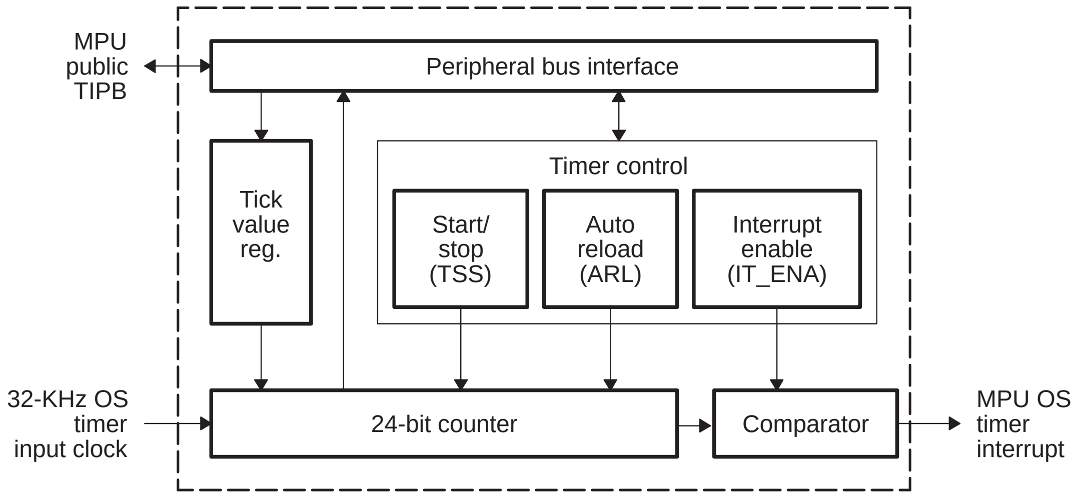
4.1.2 Features

The 32-KHz OS timer is a 24-bit down-counter that generates interrupts to the MPU. The following features are available:

❑ 24-bit down count
❑ Timer reset
❑ Timer start/stop
❑ Interrupt generation as timer counter expires
❑ Support of both auto-reload mode and one-shot mode
❑ On-the-fly register read and write
❑ Interrupt enable/disable
❑ Wake up the OMAP chip from deep sleep mode

4.1.3 Functional Block Diagram

Figure 6. 32-KHz OS Timer Block Diagram



4.2 Common Architecture and Operations

4.2.1 Interrupt Period

The timer interrupt period is defined by the value loaded into the tick value (OS_TIMER_TICK_VAL) register, as shown in the following equation:

```
IRQ_period = (OS_TIMER_TICK_VAL+1)/32768
```

Table 27 shows examples for various OS timer interrupt periods.

Table 27. Timer Interrupt Period According to TVR Value

OS_TIMER_TICK_VAL register	Interrupt Period
0x0000 0001	61 µs
0x0000 028F	19.9 ms
0x0000 07FF	62.5 ms
0x0000 FFFF	2 s
0x00FF FFFF	512 s

Setting the OS_TIMER_TICK_VAL register to 0 is not allowed and leads to unpredictable results.

CAUTION

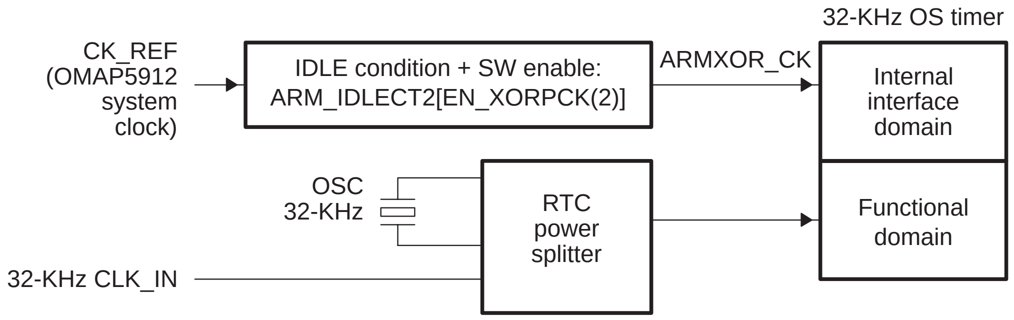
4.2.2 Clock Control

Figure 7 shows the clock input to the OS timer.

The functional domain clock directs the operation of the 32-KHz OS timer. It can be sourced from either the 32-KHz oscillator or an external 32-KHz clock. The clock is active even if the OMAP5912 device is in deep sleep mode.

The interface clock allows the MPU to access the registers of the 32-KHz OS timer. Its source is the external OS and reference peripheral clock (ARMXOR_CK), which is derived from OMAP5912 system clock.

Figure 7. Clock Input to 32-KHz OS Timers


32-KHz OS timer
CK_REF (OMAP5912 system clock)
IDLE condition + SW enable: ARM_IDLECT2[EN_XORPCK(2)]
ARMXOR_CK
Internal interface domain
OSC 32-KHz
RTC power splitter
Functional domain
32-KHz CLK_IN

4.2.2.1 *Internal Interface Clock Enable for the OS Timer*

ARMXOR_CK can be stopped or activated by configuring the EN_XORPCK bit in the ARM_IDLECT2 register.

- EN_XORPCK = 0: ARMXOR_CK is disabled.
- EN_XORPCK = 1: ARMXOR_CK is active and can be disabled by configuring the IDLXOR_ARM bit in the ARM_IDLECTL1 register.
  - IDLXORP_ARM = 0: ARMXOR_CK remains active even if the MPU enters the idle mode.
  - IDLXORP_ARM = 1: ARMXOR_CK is stopped when the MPU enters the idle mode.

4.2.3 Power Management

4.2.3.1 Host Sleep

Before the MPU enters any low power states, ARMXOR_CK (as the internal interface clock for the OS timer) must be stopped in either of the following ways:

- ❏ Setting the EN_XORPCK bit in the ARM_IDLECT2 register to 0 to force the clock to stop.
- ❏ Setting the EN_XORPCK bit in the ARM_IDLECT2 register to 1, but stopping the OS timer by configuring the TSS bit in the OS_TIMER_CNTL register, and other peripherals connected to ARMXOR_CK, in which case ARMXOR_CK will be stopped automatically in conjunction with the MPU clock when the MPU enters the idle mode.

For details on OMAP5912 power management, see the *OMAP5912 Multimedia Processor Power Management Reference Guide* (SPRU753).

4.2.3.2 Wakeup

By enabling its interrupt, the 32-KHz OS timer can wake up the OMAP device from deep sleep mode when the counter passes through 0.

Note:

The 32-KHz OS timer uses its own dedicated interrupt line to wake up the MPU from deep sleep mode, while the GP timers use the shared peripheral wakeup interrupt line to wake up the MPU. For details on the GP timer wakeup capability, see section 6.2.3.

4.2.4 Reset Considerations

After reset, the OS timer is halted and left in auto-reload mode with interrupt generation disabled.

4.2.5 Interrupt Support

If the 32-KHz OS timer interrupt is enabled, an interrupt is generated when its counter decrements to 0. The 32-KHz OS timer uses MPU level 2, interrupt #22. It is falling-edge sensitive.

For details about OMAP5912 interrupt handling, see the *OMAP5912 Multimedia Processor Interrupts Reference Guide* (SPRU757).

4.2.6 Auto-Reload Mode versus One-Shot Mode

The 32-KHz OS timer supports both auto-reload mode and one-shot mode.

The tick value (OS_TIMER_TICK_VAL) register specifies the desired value for the timer to start counting. The tick counter (OS_TIMER_TICK_CNTR) register is loaded with this value and then starts to count down. When the counter reaches 0, the timer generates an interrupt to the MPU interrupt handler.

In auto-reload mode, once the interrupt is back to the high level, the OS_TIMER_TICK_CNTR register is reloaded from the OS_TIMER_TICK_VAL register and starts to count down again.

In one-shot mode, once the interrupt is back to the high level, the timer is automatically stopped (the TSS bit in the OS_TIMER_CNTL register is cleared).

4.2.7 Initialization

The initialization procedure for the OS timer is shown below.

1) Enable the interface clock by setting the EN_XORPCK bit in the ARM_IDLECT2 register.

2) Configure the MPU interrupt controller module:

   a) Enable the MPU global interrupt.

   b) Program the corresponding level 2 interrupt priority level (ILR) register (for the OS timer, it is ILR22).

      i) Program the PRIORITY field to set the interrupt priority (0 highest; 31 lowest).

      ii) Set the SENS_LEVEL bit to 0 as falling edge sensitive.

      iii) Set the FIQ bit to 1 to set the interrupt as a FIQ.

   c) Prepare the FIQ ISR and place its address the appropriate entry of the interrupt vector (entry 7 in the vector).

   d) Prepare the ISR for the OS timer interrupt.

   e) Enable the OS timer interrupt by clearing the corresponding bit in the MPU level 2 interrupt mask register.

4.2.8 Common Operations

4.2.8.1 Read Timer Counter

Setting the TSS bit in the OS_TIMER_CNTL register starts the timer count.

4.2.8.2 Start/Stop

Writing 1 to the TSS bit in the OS_TIMER_CNTL register will launch the timer to start counting.

Resetting the TSS bit causes the counter to be stopped on the next 32-KHz clock cycle. Then the timer counter keeps the current value.

4.2.8.3 Loading/Auto-Reloading

Loading the counter in the timer can be done in two ways:

- Writing a 1 to the TRB bit in the OS_TIMER_CNTL register. The counter will be reloaded on the next 32-KHz clock cycle, whether or not the timer is counting.
- Waiting until the counter reaches 0. It then is reloaded from the OS_TIMER_TICK_VAL register if the auto-reload bit (ARL) is set to 1. Otherwise, the timer is stopped.

4.2.8.4 Peripheral Alignment and Data Width

The TIPB interface data path is 32 bits wide. To keep software compatible with earlier 16-bit timer versions, the access width is taken into account when writing to the registers. This means that writing 8 bits (or 16 bit) sets 24 MSB (or 16 MSB) to 0.

For read accesses, all 32 bits are always output. All unused register bits remain at 0.

4.2.9 Emulation Considerations

The 32-KHz OS timer keeps running even if the debugger halts the MPU.

4.2.10 Pseudo Code Example

The pseudo code below shows how to use the OS timer.

1) Initialize the OS timer.
2) Configure the OS timer:
   a) Enable the OS timer to generate interrupts upon counter expiring by setting the IT_ENA bit in the OS_TIMER_CNTL register.
   b) Program the ARL bit in the OS_TIMER_CNTL register to configure the working mode. (ARL=1: auto-reload mode; ARL=0: one-shot mode)
3) Write the desired counter initial value to the OS_TIMER_TICK_VAL register.
4) Trigger the OS timer to load the initial counter value:
   a) Set the TRB bit in the OS_TIMER_CNTL register.
   b) Poll the TRB bit. When it is cleared, the OS_TIMER_TICK_CNTR register has been loaded with the OS_TIMER_TICK_VAL register.
5) Start the timer by setting the TSS in the OS_TIMER_CNTL register.
6) When the counter reaches 0, an interrupt occurs and control jumps to the FIQ ISR, which will:
   a) Read the MPU level 1 and 2 interrupt source register for FIQ to get the interrupt number (which should be level 2 interrupt #22).
   b) Call the OS timer ISR.
   c) Set the NEW_FIQ_ARG bit in both the level 2 and level 1 interrupt control registers to acknowledge the interrupt.
7) If the OS timer is configured for auto-reload mode, the counter is reloaded from the OS_TIMER_TICK_VAL register on the next cycle and then starts to count down again.
8) If the OS timer is configured for one-shot mode, repeating step 3-5 will allow the OS timer to generate the next interrupt.

4.3 32-KHz OS Timer Registers

Table 28 lists the OS timer registers which are all 32 bits. The base (byte) address is 0xFFFB 9000. Table 29 lists the register access timing constraints. Table 30 through Table 32 provide the descriptions for each register.

Table 28. OS Timer Registers

Name	Description	R/W	MPU (Byte) Address
OS_TIMER_TICK_VAL	Tick value register	R/W	0xFFFB 9000
OS_TIMER_TICK_CNTR	Tick counter register	R	0xFFFB 9004
OS_TIMER_CNTL	Timer control register	R/W	0xFFFB 9008

Table 29. OS Timer Register Access Timing Constraints

Register Name	Read	Write
OS_TIMER_TICK_VAL	Can be read anytime. The value read is the last value written.	Two consecutive writes must be separated by at least one 32-KHz clock period (31.25 µs). Otherwise, the value written is not assured.
OS_TIMER_TICK_CNTR	Reads are resynchronized with the interface clock. This register can be read while the interface clock is active, regardless of the timer functional clock status.	Writing to this register has no effect.
OS_TIMER_CNTL	Can be read anytime. The value read is the last value written.	Two consecutive writes must be separated by at least one 32-KHz clock period (31.25). Otherwise, the value written is not assured.

Table 30. OS Timer Control Register (OS_TIMER_CNTL)

MPU Byte Base Address = 0xFFFB 9000, Byte Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset Value**
31-4	Reserved	Reserved.		
3	ARL	Auto-reload/One-shot mode	R/W	1
		0: One-shot mode. When the counter reaches zero, an interrupt is generated and the timer is stopped.		
		1: Auto-reload mode.		
2	IT_ENA	Interrupt enable/disable	R/W	0
		0: Interrupt is disabled.		
		1: Interrupt is enabled.		
1	TRB	Timer reload bit	R/W	0
		1: Reloads the counter. Once the counter is reloaded, TRB is reset.		
0	TSS	Start/Stop timer	R/W	0
		0: Stop timer.		
		1: Start timer.		
		If one-shot mode is selected (ARL = 0), this bit is automatically cleared when the timer expires.		

Table 31. OS Timer Tick Value Register (OS_TIMER_TICK_VAL)

MPU Byte Base Address = 0xFFFB 9000, Byte Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset Value**
31-24	Reserved	Reserved.	R	0x00
23-0	TICK_VALUE	This value is loaded when the timer expires or when it starts.	R/W	0xFFFFFF

Table 32. OS Timer Tick Counter Register (OS_TIMER_TICK_CNTR)

MPU Byte Base Address = 0xFFFB 9000, Byte Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset Value**
31-24	Reserved	Reserved.	R	0x0
23-0	TICK_COUNTER	Current value of the timer counter.	R	0xFFFFFF

5 32-KHz Synchronization Timer

5.1 Introduction

This section details the purpose and features of the 32-KHz synchronization timer in the OMAP5912. The information does not apply to the OMAP5910 device as it does not have this timer.

5.1.1 Purpose

The 32-KHz synchronization timer is a simple 32-bit counter clocked by the 32-KHz input. Upon waking up from the deep sleep mode, the 32-KHz synchronization timer can be used to determine the sleep duration.

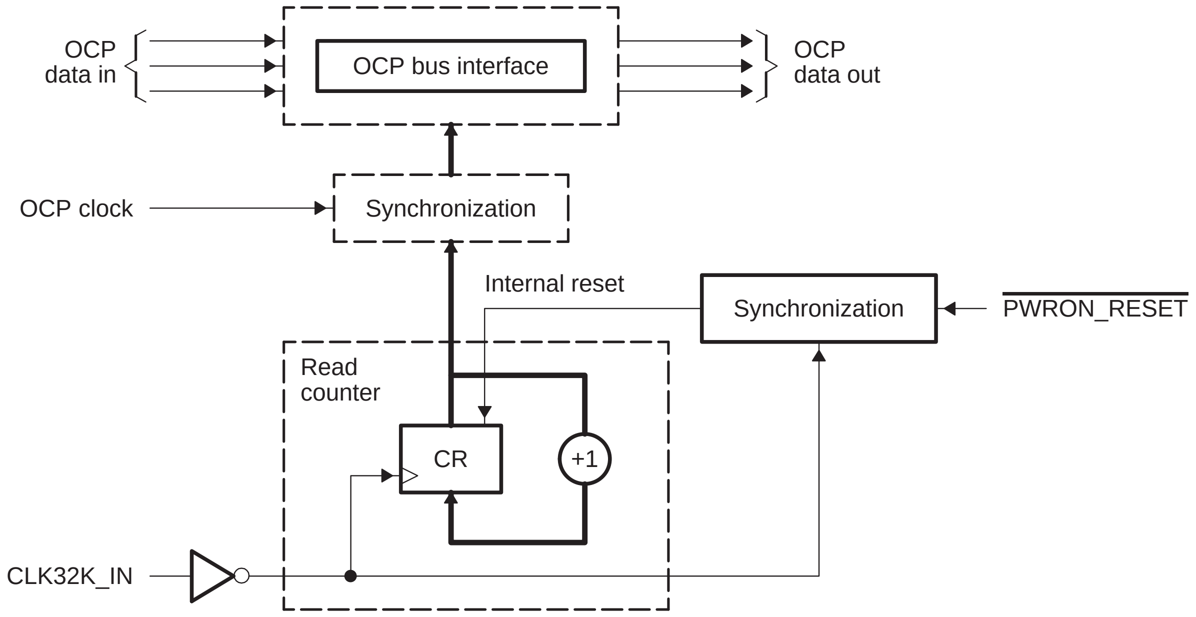
5.1.2 Features

The following features are available with the 32-KHz synchronization timer:

- ❑ 32-bit count up
- ❑ Synchronization with the 32-KHz input clock
- ❑ On-the-fly count value read
- ❑ Dynamically shared by MPU and DSP

5.1.3 Functional Block Diagram

Figure 8. 32-KHz Synchronization TImer Block Diagram



5.2 Common Architecture and Operations

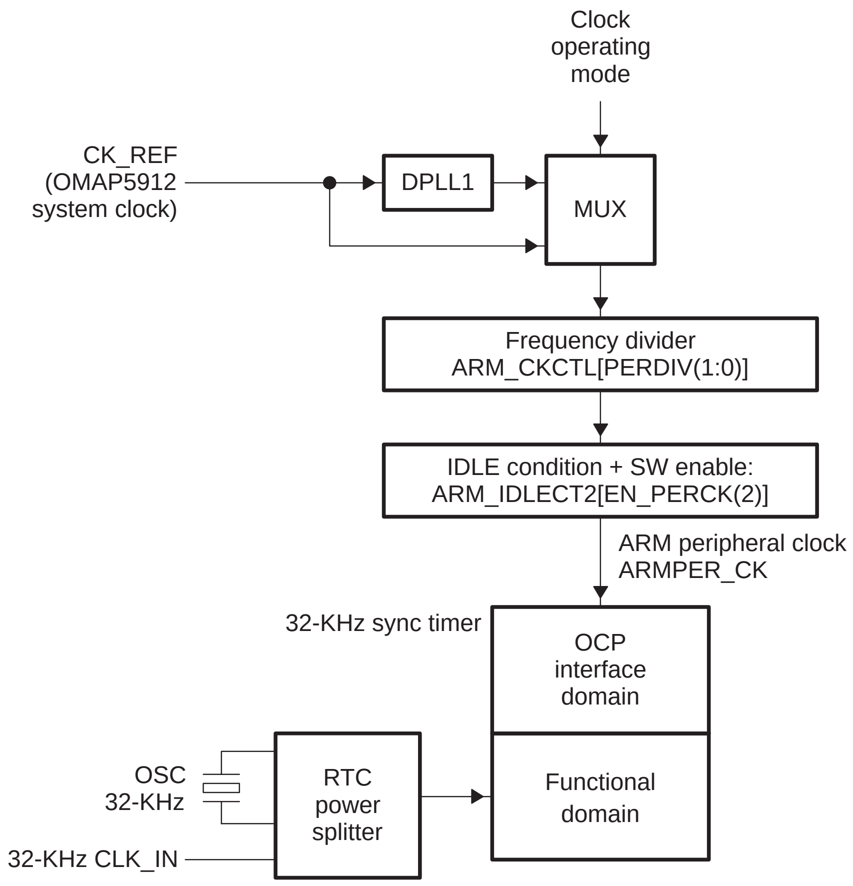
5.2.1 Clock Control

There are two input clocks for the synchronization timer: interface clock and function domain clock, as shown in Figure 9.

The 32-KHz synchronization timer is clocked by the 32-KHz oscillator or an external 32-KHz clock. It can remain active even if the OMAP5912 device is in deep sleep mode. For the external 32-KHz clock requirements, see the *OMAP5912 Multimedia Processor Clocks Reference Guide* (SPRU751).

The clock input source for the internal interface domain is the ARM peripheral clock (ARMPER_CK), which is derived from the OMAP5912 system clock.

Figure 9. Clock Input to the 32-KHz Synchronization Timers


Clock operating mode
CK_REF (OMAP5912 system clock)
DPLL1
MUX
Frequency divider ARM_CKCTL[PERDIV(1:0)]
IDLE condition + SW enable: ARM_IDLECT2[EN_PERCK(2)]
ARM peripheral clock ARMPER_CK
32-KHz sync timer
OCP interface domain
Functional domain
OSC 32-KHz
RTC power splitter
32-KHz CLK_IN

5.2.1.1 Interface Clock Configuration

The internal interface clock for the synchronization timer is the ARM peripheral clock. This clock can be derived from the OMAP5912 system clock or from the DPLL1 clock. The frequency can be further scaled down by configuring the PERDIV bits in the ARM_CKCTL register. The divisor value can be 1, 2, 4 or 8. For detailed information on ARM peripheral clock, see the *OMAP5912 Multimedia Processor Clocks Reference Guide* (SPRU751).

ARMPER_CK can be stopped or activated by configuring the EN_PERCK bit in the ARM_IDLECT2 register.

- EN_PERCK = 0: The ARMPER_CK is stopped.
- EN_PERCK = 1: ARMPER_CK is activated and can be stopped by configuring the IDLLPER_ARM bit in the ARM_IDLECTL1 register.
  - IDLLPER_ARM = 0: ARMPER_CK remains active even if the MPU and TC enter the idle mode.
  - IDLLPER_ARM = 1: ARMPER_CK is stopped in conjunction with the MPU and TC clocks when the idle mode is entered.

5.2.2 Reset Considerations

The synchronization timer module is reset with the external asynchronous powerup reset ($\overline{\text{PWRON\_RESET}}$).

When $\overline{\text{PWRON\_RESET}}$ is released after three 32-KHz clock periods, the counter starts counting up from the reset value of the read counter (32K_SYNC_CNT_CR) register on the falling edge of the 32-KHz clock. The reset value of 32K_SYNC_CNT_CR is 0x03.

5.2.3 Common Operations

5.2.3.1 Read Timer Counter

The counter value can be read from the 32-bit read counter register (32K_SYNC_CNT_CR). For the DSP, it is necessary to perform two 16-bit accesses to read the register. Thus, in the case of the DSP, the LSW should be read first, followed by the MSW.

Internal synchronization logic allows the counter value to be read while the counter is running. The time latency to read a synchronized register is one interface (ARMPER_CK) clock cycle.

5.3 Synchronization Timer Registers

Table 33 lists the synchronization timer registers. All of the registers are 32 bits. The base (byte) address in MPU space is 0xFFFB : C400. The base (word) address in DSP IO space is 0xE200. Table 34 and Table 35 include the descriptions for each register.

Table 33. Synchronization Timer Registers

Name	Description	R/W	MPU Byte Address	DSP Word Address
32K_SYN_CNT_REV	32k synchronization count CID revision identification	R	0xFFFB C400 (MSW) 0xFFFB C402 (LSW)	0xE201 (MSW) 0xE200 (LSW)
32K_SYN_CNT_CR	32k synchronization count read counter	R	0xFFFB C400 (MSW) 0xFFFB C402 (LSW)	0xE209 (MSW) 0xE208 (LSW)

Table 34. Synchronization Timer Identification Register (32K_SYN_CNT_REV)

MPU Byte Base Address = 0xFFFB C400, Byte Offset = 0x02 LSW, 0x00 MSW
DSP Word Base Offset = 2E00, Word Offset = 0x00 LSW, 0x01 MSW

Bit	Name	Function	R/W	Reset
31:8	RESERVED	Reads return 0		0x000000
7:0	CID_REV	Module HW revision number of the current timer module: value set by hardware. Four LSBs of CID_REV indicate a minor revision. Four MSBs of CID_REV indicate a major revision.	R	0x10

Table 35. Synchronization Timer Read Counter Register (32K_SYN_CNT_CR)

MPU Byte Base Address = 0xFFFB C400, Byte Offset = 0x12 LSW, 0x10 MSW
DSP Word Base Offset = 0x2E00, Word Offset = 0x08 LSW, 0x09 MSW

Bit	Name	Function	R/W	Reset
31:16	COUNTER_HI	Value of 32-KHz synchronization timer counter (16-bit MSB).	R	0x0000
15:0	COUNTER_LO	Value of 32-KHz synchronization timer counter (16-bit LSB).	R	0x0003

6 General Purpose Timers

6.1 Introduction

This section presents the purpose and features of the general purpose (GP) timers in the OMAP5912. The GP timers are not available on the OMAP5910.

6.1.1 Purpose

There are eight instances of GP timers in this peripheral module. Each timer instance contains a 32-bit free running upward counter that has housekeeping timer capability. In addition, the built-in compare logic allows an interrupt event on the programmable counter matching value.

The GP timer 1, 2 and 3 can be configured to provide a programmable pulse-width modulation (PWM) output on the corresponding dedicated output pin. The output pin can be programmed to generate one pulse or to toggle when an overflow or a match event occurs.

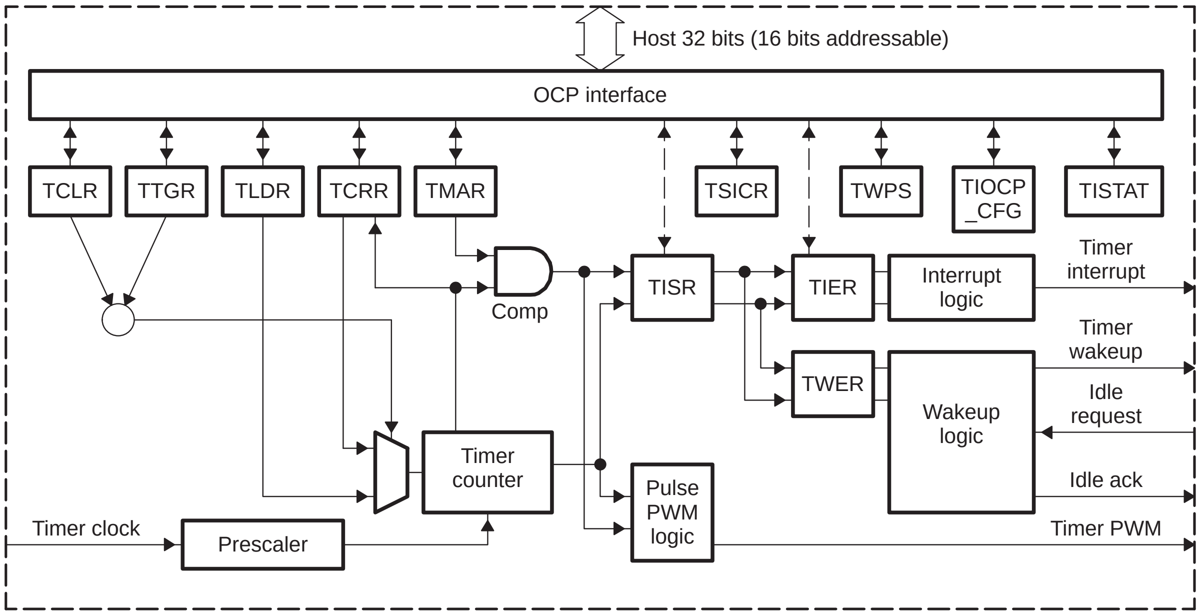
6.1.2 Features

The GP timer has the following features:

- 32-bit count up with compare logic
- Supports both one-shot and auto-reload mode
- Can be started/stopped
- Programmable divider for input clock
- 16-/32-bit addressing
- On-the-fly read/write registers
- Interrupts generated on overflow and compare events
- Interrupt enable/disable
- Wakeup enable/disable
- Register writes performed using either a posted or non-posted methodology
- Dedicated output trigger/PWM signal for GP timer 1, 2, and 3
- Statically shared by the MPU and DSP (Note: The MPU and the DSP must acquire ownership of the timer by software before using the timer. The MPU has default ownership.)

6.1.3 Functional Block Diagram

Figure 10. GP Timer Block Diagram



6.1.4 Supported Modes

A GP timer supports two major functional modes: housekeeping mode and compare mode.

When used as a housekeeping timer, the timer supports both one-shot and auto-reload mode. An interrupt can be generated upon the counter overflow.

When the internal compare logic is enabled, a GP timer allows interrupt events on the programmable counter matching value. By default, the compare mode is disabled upon reset. For specific information on compare mode, see section 6.3.

The overflow and counter matching interrupts can be enabled at the same time. Upon the interrupt, the interrupt source can be identified by reading the interrupt status register (GPTMR_TISR).

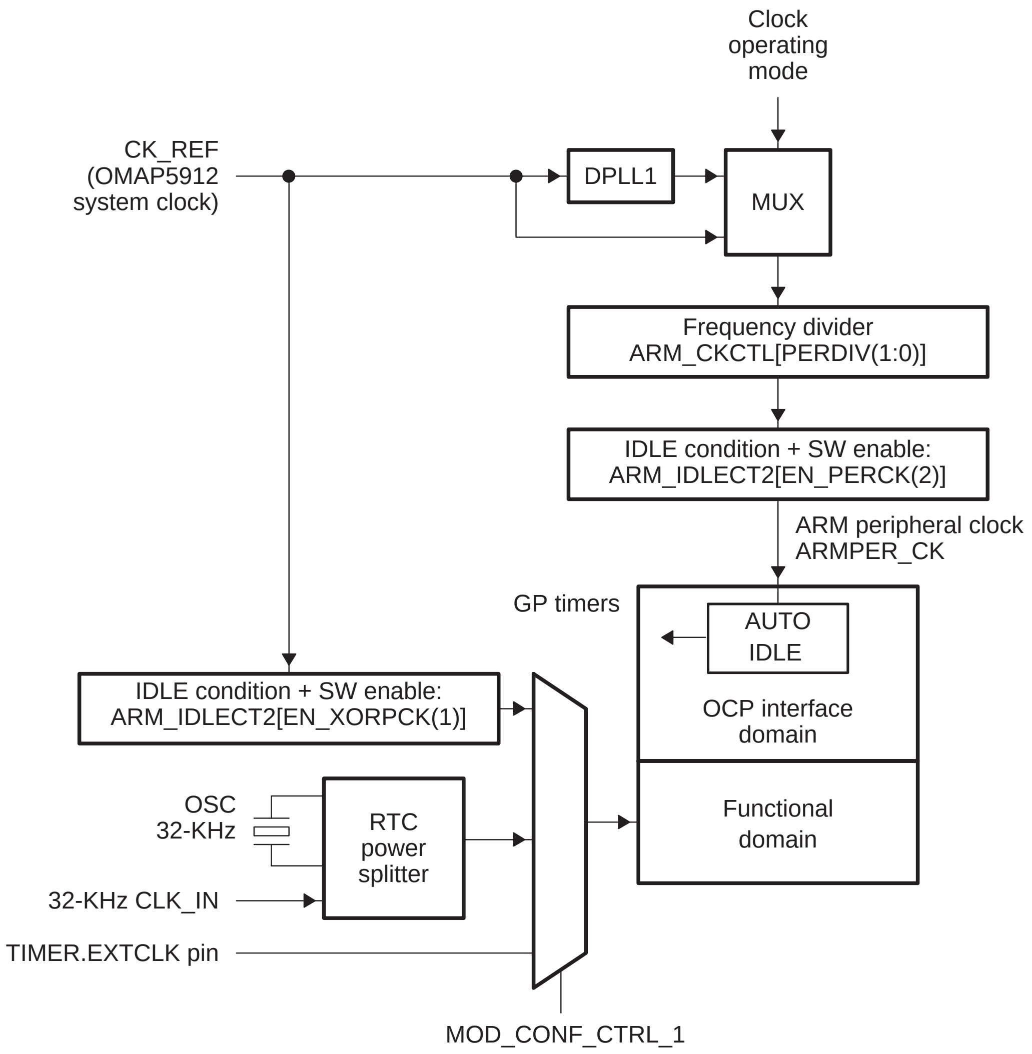
For GP timer 1, 2, and 3, a dedicated output signal can be pulsed or toggled on overflow and match events. This offers timing stamp trigger signal or pulse-width modulation (PWM) signal sources. For detailed information on PWM, see section 6.3.

6.2 Common Architecture and Operations

6.2.1 Clock Control

There are two input clocks for each GP timer, the OCP interface clock and the function domain clock. Figure 11 shows the clock input to GP timers.

Figure 11. Clock Input for GP TImers


Clock operating mode
CK_REF (OMAP5912 system clock)
DPLL1
MUX
Frequency divider ARM_CKCTL[PERDIV(1:0)]
IDLE condition + SW enable: ARM_IDLECT2[EN_PERCK(2)]
ARM peripheral clock ARMPER_CK
GP timers
AUTO IDLE
OCP interface domain
Functional domain
IDLE condition + SW enable: ARM_IDLECT2[EN_XORPCK(1)]
OSC 32-KHz
RTC power splitter
32-KHz CLK_IN
TIMER.EXTCLK pin
MOD_CONF_CTRL_1

6.2.1.1 Configuration of the Input Clock for GP Timers

The OCP interface clock for the GP timer is the ARM peripheral clock (ARMPER_CK), which also serves as the interface clock for the 32-KHz synchronization timer. For detailed information on the ARM peripheral clock, see the *OMAP5912 Multimedia Processor Clocks Reference Guide* (SPRU751).

For each GP timer, there are three possible sources for the function domain clock. The selection is done by configuring the CONF_MOD_GPTIMERx_CLK_SEL_R field in the MOD_CONF_CTRL_1 register (where x=1...8). Table 36 lists all the clock sources and the corresponding value of CONF_MOD_GPTIMERx_CLK_SEL_R. For additional information on MOD_CONF_CTRL_1, see the *OMAP5912 Multimedia Processor Initialization Guide* (SPRU752).

Table 36. Function Domain Clock for the GP Timer

Input Clock	CONF_MOD_GPTIMERx_CLK_SEL_R
ARMXOR_CK	00
32-KHz clock	01
TIMER.EXTCLK pin	10

The default function domain clock is ARMXOR_CK. ARMXOR_CK is the external OS and reference peripheral clock which also serves as the interface domain clock for the 32-KHz OS timer.

6.2.2 Timeout Period

The GP timer is composed of a prescaler stage and a timer counter.

The prescaler stage is clocked with the function domain clock and acts as a clock divider for the timer counter stage to reduce the input function clock frequency. It is enabled when the PRE bit in the timer control register (GPTMR_TCLR) is set. The clock divisor (PS) value can be configured by programming the PTV field in GPTMR_TCLR.

Table 37 lists the PS values based on various PRE and PTV configurations.

Table 37. Clock Divisor Values

PRE	PTV	Divisor (PS)
0	X	1
1	n (n=0...7)	2^{n+1}

The timeout period depends on:

- ❑ Value of the clock divisor (PS)
- ❑ Value loaded into the timer load register (GPTMR_TLDR)

```
timeout period = (0xFFFF FFFF − TLDR + 1) ×
   timer_clock_period × clock divider (PS)
```

Where timer_clock_period = 1/timer_clock_frequency.

For example, Table 38 shows various timeout periods with a 32-KHz timer function clock and where PS = 1.

Table 38. TLDR Value and Corresponding Timeout Period (where PS = 1)

GPTMR_TLDR	Timeout Period
0x0000 0000	37 h
0xFFFF 0000	2 s
0xFFFF FFFF	31.25 µs

6.2.3 Power Management

6.2.3.1 Host Sleep

Before the MPU subsystem enters any the low power state, both the ARMPER_CK and the ARMXOR_CK must be stopped. To stop ARMPER_CK, see section 5.2.1.1 (interface clock configuration for the synchronization timer). To stop ARMXOR_CK, see section 4.2.3 (power management for the OS timer).

6.2.3.2 GP Timer Sleep

When a GP timer is in the idle state, its interface clock is switched off. If the functional clock is from ARMXOR_CK, it is also off. If the functional clock is the 32-KHz clock, the GP timer can keep counting even in the idle state, as long as it is not stopped. As such, the GP timer can be used to wake up the MPU when the MPU subsystem is in deep sleep mode.

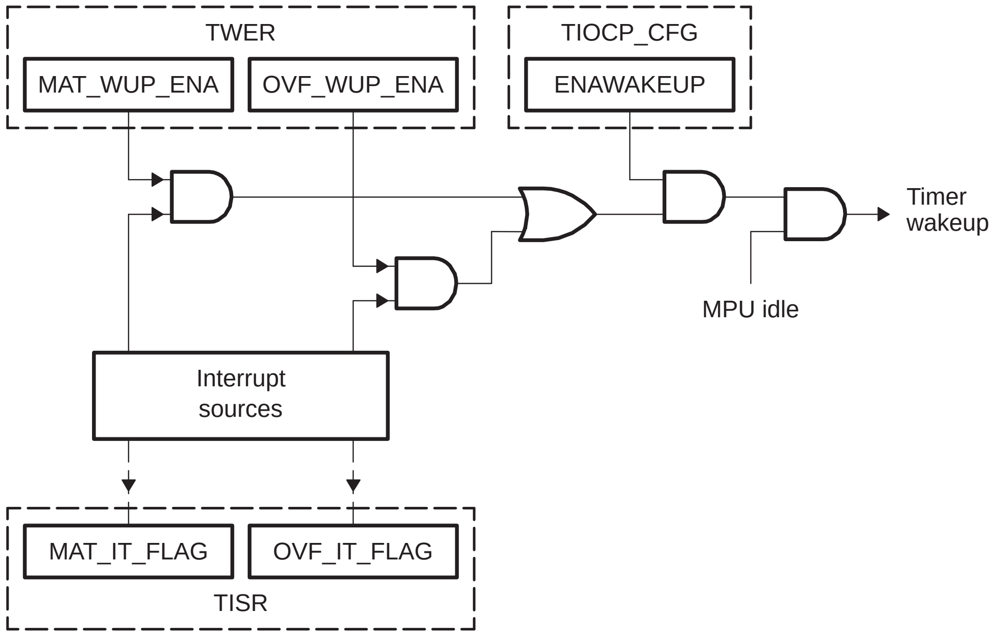
Before the MPU subsystem enters the deep sleep mode, most peripherals must be idle, including all of the GP timers. A GP timer can either enter the idle state or remain active, depending on the value of the IDLEMODE field in the timer OCP configuration register (GPTMR_TIOCP_CFG) as follows:

- ❑ IDLEMODE = No-idle (01), the timer does not go into idle state.
- ❑ IDLEMODE = Force-idle (00), the timer goes into idle state independently of the internal module state.
- ❑ IDLEMODE = Smart-idle (10), the timer behavior is defined by the AUTOIDLE bit in the GPTMR_TIOCP_CFG register. If the bit is 1, the timer goes into the idle state when the timer is not used. If the bit is 0, the timer does not go into the idle state.

6.2.3.3 Wakeup

When the MPU subsystem is in the deep sleep mode, the wakeup capability of the GP timer can be used to wake up the MPU if the 32-KHz clock is selected as its function clock. The wakeup capability can be enabled by configuring the ENAWAKEUP bit in the GPTMR_TIOCP_CFG register. With the wakeup capability enabled, a wakeup request (interrupt) can be sent to the MPU when an overflow or match event happens. Note that a wakeup request can be generated only if the MPU subsystem is in low power mode. Figure 12 shows the wakeup request generation.

Figure 12. Wakeup Request Generation



When the MPU receives a wakeup request issued by the GP timer, the following sequence happens:

1) The interface clock is reactivated if it had been switched off (AUTOIDLE bit in the GPTMR_TIOCP_CFG register was set).
2) The MPU deactivates the idle request signal.
3) The timer deactivates the idle acknowledge signal.
4) The MPU can then read the corresponding bit in the GPTMR_TISR register to find out which interrupt source has triggered the wakeup request.
5) After acknowledging the wakeup request, the MPU resets the status bit and releases the interrupt line by writing a 1 to the corresponding bit in the GPTMR_TISR register.

For details on OMAP5912 power management, see the *OMAP5912 Multimedia Processor Power Management Reference Guide* (SPRU753).

6.2.4 Register Write Mode: Posted versus Non-Posted

All the GP timer registers are 32 bits wide, accessible via OCP interface with 16-bit or 32-bit OCP access (read/write). The 16-bit wide access mode requires two consecutive write operations (16 LSBs followed by 16 MSBs).

The host processor uses the OCP bus protocol to write the GPTMR_TLDR, _TCRR, _TIER, _TISR, _TCLR, _TIOCP_CFG, _TWER, _TTGR, _TSICR and _TMAR registers synchronously with the timer interface clock. When writing to any of these registers (regardless of whether they are 16-bit or 32-bit write), the access mode can be the posted mode, if the following condition is true:

```
frequency(interface_clock) > 4 × frequency(function domain clock)
```

Otherwise the write mode has to be non-posted.

6.2.4.1 Posted Write Mode

This mode is used if the POSTED bit is set to 1 in the timer synchronization interface control register (GPTMR_TSICR). However, before setting this bit, the interface clock frequency must be four times greater than the domain clock frequency.

It uses a posted write scheme for updating any internal register. This means that the write transaction is immediately acknowledged on the OCP interface, although the effective write operation occurs later, because of a resynchronization in the timer clock domain. The advantage is that neither the interconnect nor the CPU that requested the write transaction are stalled. Each register has a status bit that is set if there is a pending write access to the register.

In this mode, it is mandatory that the processor check this status bit prior to any write access. If a write is attempted to a register with a previous access pending, the previous access is discarded without notice. The status bits are bits [4:0] (pending write status bits) in the timer write pending status register (GPTMR_TWPS). These bits are automatically cleared by internal logic when the write to the corresponding register is acknowledged.

The timer module updates the timer counter register (GPTMR_TCRR) value synchronously with the OCP clock. Consequently, any read access to GPTMR_TCRR does not add any resynchronization latency; the current value is always available.

If a write access is pending for a register, reading from this register does not yield a correct result. Software synchronization must be used to avoid incorrect results.

The drawback of this automatic update mechanism is that it assumes a given relationship between the timer interface frequency and the timer clock frequency.

Note:

The GPTMR_TISR and GPTMR_TIER registers operate only with the timer interface clock, so they do not need a write pending status bit.

6.2.4.2 Software Synchronous Non-Post Write

This mode is used if the POSTED bit in the GPTMR_TSICR is set to 0.

It uses a non-posted write scheme for updating any internal register. This means that the write transaction is not acknowledged on the OCP interface until the effective write operation occurs, after the resynchronization in the timer clock domain. The drawback is that both the interconnect and the processor are stalled during this period. The processor cannot serve interrupts, increasing the interrupt latency. An interconnect that includes timeout logic to detect erroneous transactions can generate an unwanted system abort event.

This stall period can be quantified as follows:

$$T_{(stall)} = 3\ \text{interface\_clock\_cycles} + 5 \times \text{function\_domain\_clock\_cycles}$$

The same full resynchronization scheme is used for a read transaction. The same stall period applies. A register read following a write to the same register is always coherent. This mode is functional regardless of the ratio between the OCP interface frequency and the functional clock frequency.

6.2.4.3 Write Mode Selection

Upon reset, the posted mode is enabled by default. The posted mode can be disabled by configuring the POSTED bit to 0 in the GPTMR_TSICR register. The choice between the synchronization modes must consider the frequency ratio and the stall periods that can be supported by the system, without affecting the global performance.

6.2.5 Reset Considerations

Upon a power reset, a GP timer defaults to a one-shot timer but is stopped. Both the overflow and match interrupts are disabled. The prescaler is also disabled. The register access mode is posted.

A GP timer can also be reset by setting the SFT bit in the GPTMR_TSICR register. The effect on the GP timer is the same as a power reset.

As the GP timers are outside of the OMAP3.2 gigacell, an MPU or DSP software reset does not automatically reset them, nor does an MPU/DSP watchdog reset.

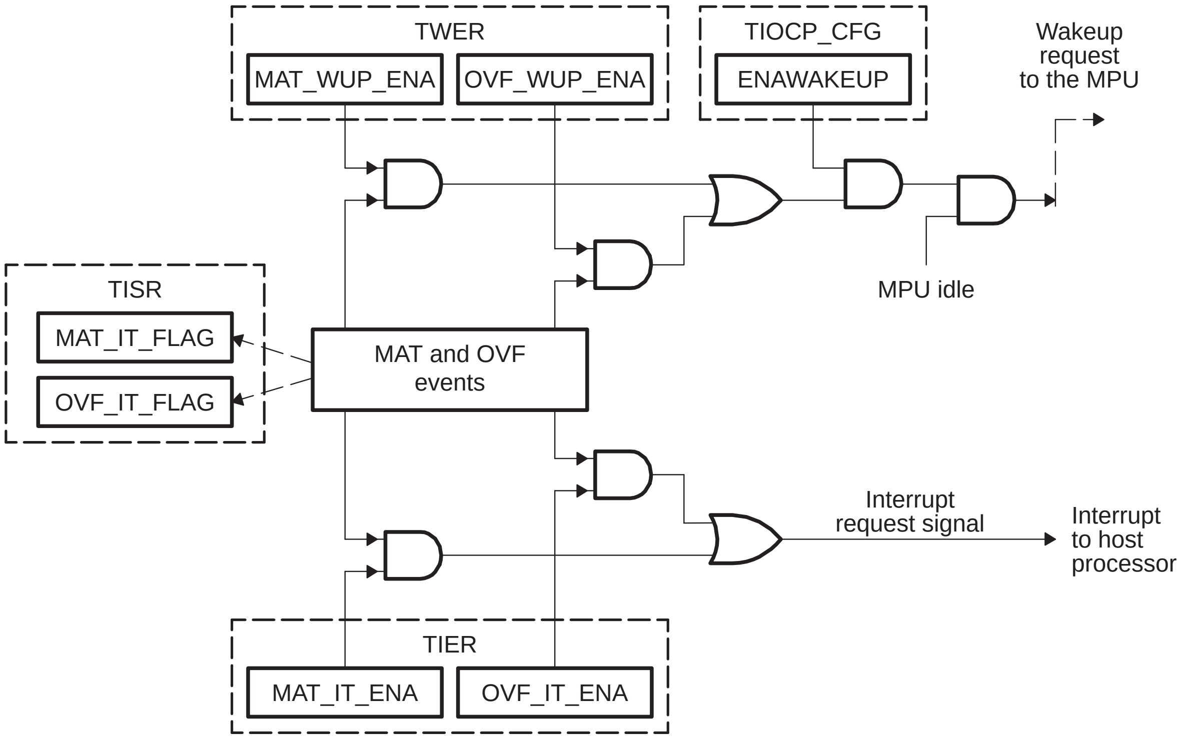
See section 6.2.7 for more information on how to bring a GP out of reset.

6.2.6 Interrupt Support

6.2.6.1 Interrupt Events

Both events are combined into one interrupt request signal and one wakeup signal, as shown in Figure 13. Each event can be independently enabled/disabled with the corresponding bit in the timer interrupt enable register (GPTMR_TIER) for the interrupt and the corresponding bit in the timer wakeup enable register (GPTMR_TWER) for the wakeup.

Figure 13. GP Timer Interrupt Architecture



The internal interrupt line of each GP timer is connected to the host interrupt handler separately. Table 39 lists the interrupt numbers for each GP timer. All the interrupts are level sensitive.

Table 39. GP Timer Interrupt Lines

GP Timer Instance	MPU Interrupt Number	DSP Interrupt Number
GP timer 1	Level 1 interrupt #17	Level 2.1 interrupt #1
GP timer 2	Level 1 interrupt #18	Level 2.1 interrupt #2
GP timer 3	Level 1 interrupt #34	Level 2.1 interrupt #3
GP timer 4	Level 1 interrupt #35	Level 2.1 interrupt #4
GP timer 5	Level 1 interrupt #36	Level 2.1 interrupt #5
GP timer 6	Level 1 interrupt #37	Level 2.1 interrupt #6
GP timer 7	Level 1 interrupt #38	Level 2.1 interrupt #7
GP timer 8	Level 1 interrupt #39	Level 2.1 interrupt #8

The wakeup line of all the GP timers are merged before connecting to the host interrupt handler. The interrupt number is MPU level 2 #46 (peripheral wakeup interrupt). It is falling edge sensitive.

When the interrupt or wakeup event has been issued, the associated interrupt status bit is set in the GPTMR_TISR register. The pending interrupt event is reset when the set status bit is overwritten by a 1.

6.2.7 Initialization

The following initialization procedure needs to be done before a GP timer can be used.

1) If the GP timer is accessed by the DSP:
   a) The MPU needs to program the corresponding static switch register to release the GP timer.
   b) The DSP needs to the program the corresponding static switch register to take the ownership.
2) The MPU configures the interface clock for the GP timer.
   a) Program the PERDIV field in the ARM_CKCTL register to set the clock divisor for the interface clock.
   b) Set the EN_PERCK bit to enable the GP timer interface clock.
3) The MPU selects the function clock for the GP timer by configuring the CONF_MOD_GPTIMERx_CLK_SEL_R field in the MOD_CONF_CTRL_1 register.

4) Configure the interrupt controller module. If the GP timer is accessed by the MPU, the MPU interrupt controller is configured as follows:
   a) Enable the MPU global interrupt.
   b) Program the corresponding level 1 or 2 interrupt priority-level (ILR) register.
      i) Program the PRIORITY field to set the interrupt priority (0 highest; 31 lowest).
      ii) Set the SENS_LEVEL bit to 1 as low level active.
      iii) Set the FIQ bit to 1 to set the interrupt as a FIQ.
   c) Prepare the FIQ ISR and put its address in the appropriate entry of the interrupt vector (entry 7 in the vector).
   d) Prepare the GP timer ISR.
   e) Enable the GP timer interrupt by clearing the corresponding bit in the level 1 or 2 MPU interrupt mask register.
5) Configure the interrupt controller module. If the GP timer is accessed by the DSP, the DSP interrupt controller is configured as following:
   a) Enable the DSP global interrupt.
   b) Program the corresponding DSP level 2.1 interrupt priority-level (ILR) register for the interrupt being used.
      i) Program the PRIORITY field to set the interrupt priority (0 highest; 31 lowest).
      ii) Set the SENS_LEVEL bit to 1 as low level active.
      iii) Set the FIQ bit to 1 to set the interrupt as a FIQ.
   c) Prepare the GP timer ISR and put its address in the appropriate entry of the interrupt vector (based on the interrupt number being used).
   d) Enable the GP timer interrupt by clearing the corresponding bit in the DSP level 2.1 interrupt mask register.

6.2.8 Common Operations

6.2.8.1 Read Timer Counter

The GPTMR_TCRR register is a 32-bit register (16-bit addressable). Therefore, the MPU can either perform one 32-bit access or two 16-bit accesses to it, while the DSP needs to perform two consecutive 16-bit accesses. Because the timer interface clock is completely asynchronous with the timer function clock, some synchronization is done to ensure that the GPTMR_TCRR value is not read while it is being incremented.

In 16-bit mode, the following sequence must be followed to read GPTMR_TCRR properly:

1) Read the lower 16-bits of GPTMR_TCRR. When the lower 16 bits are read, the upper 16 bits of are stored in a temporary register.

2) Read the upper 16 bits of GPTMR_TCRR. During this read, the values of the upper 16 bits that have been stored in the temporary register are read.

6.2.8.2 Update Timer Counter Timer

The GPTMR_TCRR can be updated in three different ways:

- ❑ Write the new counter value directly to GPTMR_TCRR.
- ❑ Write the new counter value to GPTMR_TLDR in auto-reload mode. When the timer overflows, the value in GPTMR_TLDR will be loaded to GPTMR_TCRR automatically.
- ❑ Write the new counter value to GPTMR_TLDR and trigger a timer counter reload by writing anything to timer trigger register (GPTMR_TTGR). When TTGR is written, the value in GPTMR_TLDR is loaded into GPTMR_TCRR. The PTV field in the timer control register (TCLR) is cleared in this operation.

The MPU can access the GPTMR_TCCR, GPTMR_TLDR or GPTMR_TLDR registers by either performing one 32-bit access or two 16-bit accesses to them, while the DSP needs to perform two consecutive 16-bit accesses.

6.2.8.3 Start/Stop a GP Timer

Each GP timer can be started and stopped at any time by configuring the ST bit in the GPTMR_TCLR register. GPTMR_TCRR keeps the current value after the timer is stopped. The timer can be restarted from the old value unless GPTMR_TCRR has been reloaded with a new value.

6.2.9 Emulation Considerations

During emulation mode, the GP timer can either run free or stop counting, depending on the value of the EMUFREE bit in the GPTMR_TIOCP_CFG register.

- ❑ EMUFREE = 1: The timer keeps running in emulation mode and the interrupt is still generated when overflow is reached.
- ❑ EMUFREE = 0: The prescaler and timer are frozen and both resume on exit from emulation mode.

6.2.10 Pseudo Code Example

The following pseudo code shows how to use a GP timer for its housekeeping capability.

1) Initialize to configure the clocks and interrupt module (see section 6.2.7).
2) Set the desired timeout period:
   a) Program the GPTMR_TLDR and GPTMR_TCRR register.
   b) If the prescaler is needed, set the PRE bit and program the PTV field in the TCLR register.
   c) Otherwise, reset the prescaler bit in the GPTMR_TCLR register.
3) Set the AR bit in the GPTMR_TCLR register if auto-reload mode is needed.
4) Enable the overflow interrupt by setting the OVF_IT_ENA bit in the GPTMR_TIER register.
5) Enable or disable the compare capability. To enable the compare logic, see section 6.3. Disable the compare logic as follows:
   a) Reset the CE bit in the GPTMR_TCLR register.
   b) Reset the MAT_IT_ENA bit in the GPTMR_TIER register to disable match event interrupt.
6) Disable the PWM output by programming the TRG field in the GPTMR_TCLR register.
7) Configure the idle mode and wakeup response:
   a) Configure the IDLEMODE, ENWAKEUP and AUTOIDLE fields in the GPTMR_TIOCP_CFG register.
   b) Configure the GPTMR_TWER register.
8) Configure the emulation mode by configuring the EMUFREE bit in the GPTMR_TIOCP_CFG register (1: run free; 0: timer frozen in emulation mode).
9) Select the posted/non-posted register write mode by configuring the POSTED bit in the GPTMR_TSICR register (1: posted; 0: non-posted).
10) Start the timer by setting the ST bit in the GPTMR_TCLR register. An interrupt will occur upon timeout.

11) When an interrupt occurs, control jumps to the FIQ ISR if the GP timer is used by the MPU. The FIQ ISR should:
    a) Read the MPU interrupt level 1 and level 2 source register for FIQ to get the interrupt number (which should be the number corresponding to the GP timer interrupt number).
    b) Call the GP timer ISR, which will:
        i) Read the TISR register to get the interrupt source. Bit OVF_IT_FLAG should be set.
        ii) Write a 1 to the OVF_IT_FLAG bit to clear the interrupt.
        iii) Process the interrupt.
    c) Upon returning from GP timer ISR, the FIQ ISR sets the NEW_FIQ_ARG bit in both the interrupt level 2 and level 1 control registers to acknowledge the interrupt (which allows new interrupt generation).
12) When an interrupt occurs, control jumps to DSP interrupt level 2.1 ISR if the GP timer is used by the DSP, which will:
    a) Read the DSP interrupt level 2.1 source register to get the interrupt number (which should be the GP timer interrupt number).
    b) Call the GP timer interrupt ISR , which will:
        i) Read the TISR register to get the interrupt source. Bit OVF_IT_FLAG should be set.
        ii) Write a 1 to the OVF_IT_FLAG bit to clear the interrupt.
        iii) Process the interrupt.
    c) Upon returning from GP timer ISR, it sets the NEW_FIQ_ARG bit in the DSP level 2.1 interrupt control register to acknowledge the interrupt.
13) If the GP timer is configured for auto-reload mode, GPTMR_TCRR is reloaded from the TLDR register and then starts to count up again. If the GP timer is configured for one-shot mode, reloading GPTMR_TCRR and starting the timer again will allow the next interrupt generation.

6.3 Compare Mode Operations

When the compare enable (CE) bit in GPTMR_TCLR is set to 1, the GPTMR_TCRR register is continuously compared to the value held in the timer match register (GPTMR_TMAR). GPTMR_TMAR value can be loaded at any time. When the GPTMR_TCRR and the GPTMR_TMAR value match, an interrupt event is issued if the MAT_IT_ENA bit is set.

6.3.1 Pseudo Code Example

1) Initialize to configure the clocks and interrupt module (see section 6.2.7).
2) Program GPTMR_TLDR, and GPTMR_TCRR registers. Configure prescaler if needed.
3) Set the AR bit in the TCLR register if auto-reload mode is needed.
4) Enable or disable the overflow interrupt by configuring the OVF_IT_ENA bit in the GPTMR_TIER register.
5) Enable the compare mode:
   a) Set the CE bit in the GPTMR_TCLR register.
   b) Set the MAT_IT_ENA bit in the GPTMR_TIER register to enable match event interrupt.
   c) Load the GPTMR_TMAR register with the desired value to be compared.
6) Disable the PWM output.
7) Configure the idle mode and wakeup response.
8) Configure the emulation mode by configuring the EMUFREE bit in the GPTMR_TIOCP_CFG register.
9) Select the posted/non-posted register write mode.
10) Start the timer.
11) An interrupt will occur when GPTMR_TCRR = GPTMR_TMAR. When an interrupt occurs, control jumps to the FIQ ISR if the GP timer is used by the MPU. The FIQ ISR should:
    a) Read the MPU interrupt level 1 and level 2 source register for FIQ.
    b) Call the GP timer ISR, which will:
       i) Read the GPTMR_TISR register to get the interrupt source. Bit MAT_IT_FLAG should be set.
       ii) Write a 1 to the MAT_IT_FLAG bit to clear the interrupt.
       iii) Process the interrupt.
    c) Set the NEW_FIQ_ARG bit in both the interrupt level 2 and level 1 control register to acknowledge the interrupt.

12) An interrupt will occur when GPTMR_TCRR = GPTMR_TMAR. When an interrupt occurs, control jumps to DSP interrupt level 2.1 ISR if the GP timer is used by the DSP, which should:

    a) Read the DSP interrupt level 2.1 source register to get the interrupt number, which should be the GP timer interrupt number.

    b) Call the GP timer interrupt ISR, which will:

        i) Read the GPTMR_TISR register to get the interrupt source. Bit MAT_IT_FLAG should be set.

        ii) Write a 1 to the MAT_IT_FLAG bit to clear the interrupt.

        iii) Process the interrupt.

    c) Set the NEW_FIQ_ARG bit in the DSP level 2.1 interrupt control register.

6.4 Pulse-Width Modulation

The GP timer 1, 2, and 3 can be configured to provide a programmable pulse-width modulation (PWM) output. The dedicated output pin can be programmed through the TRG and PT bits in GPTMR_TCLR to generate one pulse (one function clock cycle) or to invert the current value (toggle mode) when an overflow or a match event occurs.

The PT bit determines the pulse mode and toggle mode. The TRG bits determine on which register value the timer PWM pin toggles. The PWM pin can toggle when an overflow event occurs, or if an overflow or a match event occurs. When the match event is used, the CE bit in GPTMR_TCLR must be set. For detailed information on TRG and PT, see section 6.5.

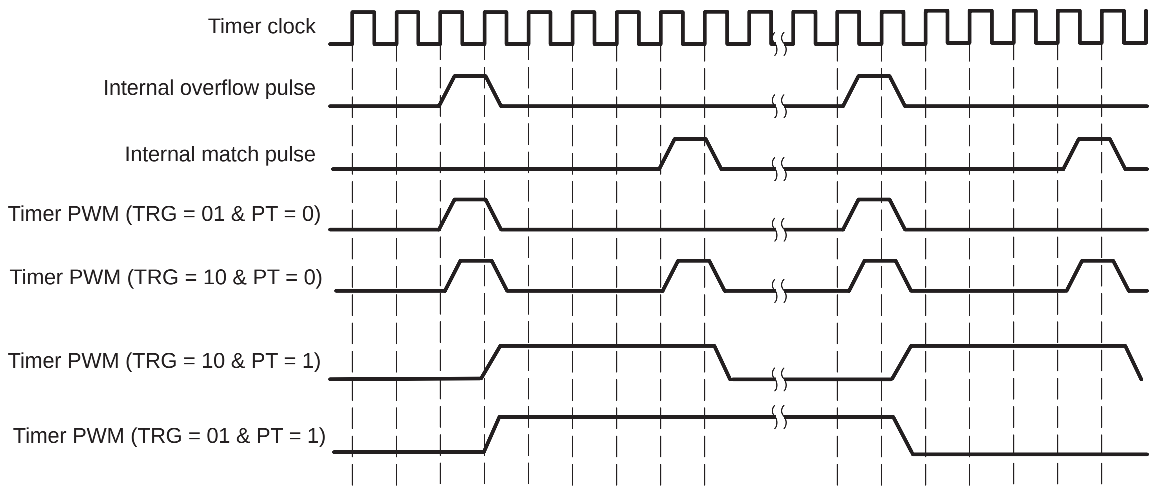
The SCPWM bit in GPTMR_TCLR determines the output pulse polarity. It can only be set or cleared when the counter is stopped or the trigger is off (TRG bits are cleared). This allows the PWM output pin to be set to a known state before modulation starts.

The modulation is synchronously stopped when the TRG bits are cleared and overflow occurs. This allows the output pin to be set to a known state when modulation is stopped.

In Figure 14 and Figure 15, the internal overflow pulse is set each time the (0xFFFF FFFFF - GPTMR_TLDR +1) value is reached, and the internal match pulse is set when the counter reaches GPTMR_TMAR register value. In Figure 14, the SCPWM bit in TCLR is set to 0, while in Figure 15 it is set to 1.

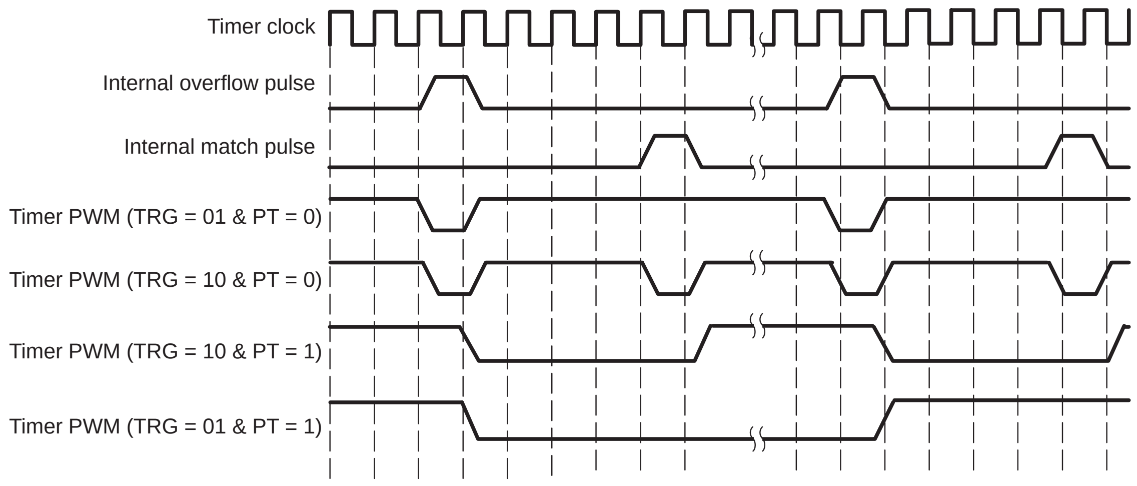
When configuring the GPTMR_TLDR and GPTMR_TMAR registers, the values written must be at least two units smaller than the overflow value (0xFFFFFFFF). If the PWM trigger events are both overflow and match, the difference between the value kept in GPTMR_TMAR and in GPTMR_TLDR must be at least two units.

Figure 14. Timing Diagram of Pulse-Width Modulation With SCPWM Bit = 0


Timer clock
Internal overflow pulse
Internal match pulse
Timer PWM (TRG = 01 & PT = 0)
Timer PWM (TRG = 10 & PT = 0)
Timer PWM (TRG = 10 & PT = 1)
Timer PWM (TRG = 01 & PT = 1)

In Figure 15, TCLR (SCPWM bit) is set to 1.

Figure 15. Timing Diagram of Pulse-Width Modulation With SCPWM Bit = 1


Timer clock
Internal overflow pulse
Internal match pulse
Timer PWM (TRG = 01 & PT = 0)
Timer PWM (TRG = 10 & PT = 0)
Timer PWM (TRG = 10 & PT = 1)
Timer PWM (TRG = 01 & PT = 1)

6.4.1 PWM Output Signal

Table 40 and Table 41 show the PWM output pins and the pin mux control register configurations for GP timers 1, 2, and 3.

Table 40. OMAP5912 ZDY Package: Pin Mux Configuration for the PWM Output Pins

GP Timer	Pin	FUNC_MUX_CNTL_6
1	K17	bits[2:0] = 110
2	K15	bits[5:3] = 100
3	K16	bits[8:6] = 010

Table 41. OMAP5912 ZZG Package: Pin Mux Configuration for the PWM Output Pins

GP Timer	Pin	FUNC_MUX_CNTL_6
1	M18	bits[2:0] = 110
2	L14	bits[5:3] = 100
3	M20	bits[8:6] = 010

6.5 GP Timer Registers

Table 42 lists the base addresses for all GP timers.

Table 42. GP Timer Register Base Address

GP Timer	MPU Domain Byte Address	DSP Word Address (I/O Space)
GP timer 1	0xFFFB 1400	0x8A00
GP timer 2	0xFFFB 1C00	0x8E00
GP timer 3	0xFFFB 2400	0x9200
GP timer 4	0xFFFB 2C00	0x9600
GP timer 5	0xFFFB 3400	0x9A00
GP timer 6	0xFFFB 3C00	0x9E00
GP timer 7	0xFFFB 7400	0xBA00
GP timer 8	0xFFFB D400	0xEA00

Table 43 lists the 32-bit (or 2 x 16-bit) GP timer registers. The registers are accessible in 16-bit mode and use little-endian addressing.

Table 44 through Table 56 contain the register descriptions.

Table 43. GP Timer Registers

Name	Description	R/W	MPU Byte Offset MSW/LSW	DSP Word Offset LSW/ MSW
GPTMR_TIDR	Timer identification	R	0x00/0x02	0x00/0x01
GPTMR_TIOCP_ CFG	Timer OCP configuration	R/W	0x10/0x12	0x08/0x09
GPTMR_TISTAT	Timer system status	R	0x14/0x16	0x0A/0x0B
GPTMR_TISR	Timer status	R/W	0x18/0x1A	0x0C/0x0D
GPTMR_TIER	Timer interrupt enable	R/W	0x1C/0x1E	0x0E/0x0F
GPTMR_TWER	Timer wakeup enable	R/W	0x20/0x22	0x10/0x11
GPTMR_TCLR	Timer control	R/W	0x24/0x26	0x12/0x13
GPTMR_TCRR	Timer counter	R/W	0x28/0x2A	0x14/0x15
GPTMR_TLDR	Timer load	R/W	0x2C/0x2E	0x16/0x17
GPTMR_TTGR	Timer trigger	R/W	0x30/0x32	0x18/0x19
GPTMR_TWPS	Timer write pending status	R	0x34/0x36	0x1A/0x1B
GPTMR_TMAR	Timer match	R/W	0x38/0x3A	0x1C/0x1D
GPTMR_TSICR	Timer synchronization interface control	R/W	0x40/0x42	0x20/0x21

Table 44. Timer Identification Register (GPTMR_TIDR)

MPU Byte Offset = 0x0 (MSW), 0x02 (LSW), DSP Word Offset = 0x0 (LSW), 0x01 (MSW)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:8	RESERVED			0x000000
7:0	TID_REV	Module HW revision number of the current timer module: value set by hardware. Four LSBs of TID_REV indicate a minor revision. Four MSBs of TID_REV indicate a major revision. A reset has no effect on value returned.	R	0x10

Table 45. Timer OCP Configuration Register (GPTMR_TIOCP_CFG)

MPU Byte Offset = 0x10 (MSW), 0x12 (LSW), DSP Word Offset = 0x08 (LSW), 0x09 (MSW)

Bit	Name	Function	R/W	Reset
31:6	RESERVED	Reserved.		0x000000
5	EMUFREE	0: The timer stops in emulation mode. 1: The timer keeps running in emulation mode unless the ST bit in the GPTMR_TCLR register is 0.	R/W	0
4:3	IDLEMODE	Power management, request/acknowledge control 00: Force idle. Put the timer in idle mode unconditionally. 01: No idle. Never put the timer in idle mode. 10: Smart idle. Behavior is defined by the AUTOIDLE bit. 11: Reserved.	R/W	00
2	ENAWAKEUP	Wakeup feature control 0: Wakeup is disabled. 1: Wakeup is enabled.	R/W	0
1	SOFTRESET	Software reset. Set this bit to 1 to reset the timer. The bit is automatically cleared by hardware. It always returns 0 for reads. 0: Normal mode. 1: Reset the OCP and the functional domain.	R/W	0
0	AUTOIDLE	Auto idle configuration 0: Disable auto idle function. 1: Enable auto idle function, which puts the timer in idle mode when there is no access.	R/W	0

This register controls various parameters of the GP timer interface.

Table 46. Timer System Status Register (GPTMR_TISTAT)

MPU Byte Offset = 0x14 (MSW), 0x16 (LSW), DSP Word Offset = 0x0A (LSW), 0x0B (MSW)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:1	RESERVED	Reserved.		0x000000
0	RESETDONE	Internal global reset monitoring. 0: Reset is ongoing. 1: Reset completed.	R	1

This register monitors the internal global reset status. This status bit is set to 1 when all clock domains have been reset. This status can be monitored by the software to check if the module is ready-to-use following a reset (either hardware or software reset).

Table 47. Timer Status Register (GPTMR_TISR)

MPU Byte Offset = 0x18 (MSW), 0x1A (LSW), DSP Word Offset = 0x0C (LSW), 0x0D (MSW)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:2	RESERVED	Reserved.		0x000000
1	OVF_IT_FLAG	0: No overflow interrupt request. 1: Overflow interrupt pending.	R/W	0
0	MAT_IT_FLAG	0: No compare interrupt request. 1: Compare interrupt pending.	R/W	0

The timer status register is used to determine which timer event requested an interrupt. Bit 0 corresponds to the compare result of TCRR and TMAR, and is set when the compare register matches the counter value. Bit 1 corresponds to the TCRR overflow.

If the value is 1, then that timer event is requesting the interrupt. To reset the status bit, a 1 must be written to the appropriate bit. However, an interrupt cannot be generated by writing a 1 to the timer status register bits. If a 0 is written to a bit in the timer status register bits, the value remains unchanged.

Table 48. Timer Interrupt Enable Register (GPTMR_TIER)

MPU Byte Offset = 0x1C (MSW), 0x1E (LSW), DSP Word Offset = 0x0E (LSW), 0x0F (MSW)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:2	RESERVED	Bit 2 must always be set to 0. Writing to bits 31:3 does not have any effect.		0x000000
1	OVF_IT_ENA	Overflow interrupt enable 0: Overflow interrupt is disabled. 1: Overflow interrupt is enabled.	R/W	0
0	MAT_IT_ENA	Compare interrupt enable 0: Compare interrupt is disabled. 1: Compare interrupt is enabled.	R/W	0

The timer interrupt enable register enables certain timer events for generating an interrupt request. Bit 0 determines whether or not the compare register flag MAT_IT_FLAG can generate an interrupt. Bit 1 determines whether or not the overflow counter flag OVF_IT_FLAG can generate an interrupt. Bit 2 must always be written as 0.

Table 49. Timer Wakeup Enable Register (GPTMR_TWER)

MPU Byte Offset = 0x20 (MSW), 0x22 (LSW), DSP Word Offset = 0x10 (LSW), 0x11 (MSW)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:2	RESERVED	Bit 2 must always be set to 0. Writing to bits 31:3 does not have any effects.		0x0000000
1	OVF_WUP_ENA	0: Overflow wakeup generation is disabled. 1: Overflow wakeup generation is enabled.	R/W	0
0	MAT_WUP_ENA	Compare interrupt enable 0: Compare wakeup generation is disabled. 1: Compare wakeup generation is enabled.	R/W	0

The timer wakeup enable register (TWER) masks the expected source of a wakeup event that generates a wakeup request. The TWER is programmed synchronously with the interface clock before any idle mode request coming from the host processor. Bit 2 must always be written as 0.

Table 50. Timer Control Register (GPTMR_TCLR)

MPU Byte Offset = 0x24 (MSW), 0x26 (LSW), DSP Word Offset = 0x12 (LSW), 0x13 (MSW)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:13	RESERVED			0x00000
12	PT	Pulse or toggle mode on timer PWM output pin 0: Pulse. 1: Toggle.	R	0
11:10	TRG	Trigger output mode on timer PWM output pin 00: No trigger. 01: Trigger on overflow. 10: Trigger on overflow and match. 11: Reserved.	R/W	00
9:8	RESERVED	Bit 9:8 must always be set to 00.	R/W	00
7	SCPWM	This bit must be set or cleared while the timer is stopped or the trigger is off. 1: Set the timer PWM output pin and select negative pulse for pulse mode. 0: Clear the timer PWM output pin and select positive pulse for pulse mode.	R/W	0
6	CE	1: Compare mode is enabled. 0: Compare mode is disabled.	R/W	0
5	PRE	Prescaler enable 0: Prescaler is disabled. 1: Prescaler is enabled.	R/W	0
4:2	PTV	Prescale clock timer value Function clock divisor = $2^{(PTV+1)}$ if PRE is enabled.	R/W	000
1	AR	1: Auto-reload timer. 0: One-shot timer.	R/W	0
0	ST	1: Start timer. 0: Stop timer. In one-shot mode (AR = 0), this bit is automatically cleared when the counter overflows.	R/W	0

Table 51. Timer Counter Register (GPTMR_TCRR)

MPU Byte Offset = 0x28 (MSW), 0x2A (LSW), DSP Word Offset = 0x14 (LSW), 0x15 (MSW)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	GPTMR_TCRR	Value of timer counter.	R/W	0x00000000

Table 52. Timer Load Register (GPTMR_TLDR)

MPU Byte Offset = 0x2C (MSW), 0x2E (LSW), DSP Word Offset = 0x16 (LSW), 0x17 (MSW)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	GPTMR_TLDR	Timer counter value loaded on overflow in auto-reload mode or on TTGR write access.	R/W	0x00000000

Table 53. Timer Trigger Register (GPTMR_TTGR)

MPU Byte Offset = 0x30 (MSW), 0x32 (LSW), DSP Word Offset = 0x18 (LSW), 0x19 (MSW)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	GPTMR_TTGR	Writing in the TTGR register, TCRR is loaded from TLDR and prescaler counter is cleared. Reload is done regardless the AR field value of TCLR register.	R/W	0x00000000

Table 54. Timer Write Pending Status Register (GPTMR_TWPS)

MPU Byte Offset = 0x34 (MSW), 0x36 (LSW), DSP Word Offset = 0x1A (LSW), 0x1B (MSW)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:5	RESERVED	Reserved.	R	0x00000 00
4	W_PEND_TMAR	When equal to 1, a write is pending to the TMAR register.	R	0
3	W_PEND_TTGR	When equal to 1, a write is pending to the TTGR register.	R	0
2	W_PEND_TLDR	When equal to 1, a write is pending to the TLDR register.	R	0
1	W_PEND_TCRR	When equal to 1, a write is pending to the TCRR register.	R	0
0	W_PEND_TCLR	When equal to 1, a write is pending to the TCLR register.	R	0

Table 55. Timer Match Register (GPTMR_TMAR)

MPU Byte Offset = 0x38 (MSW), 0x3A (LSW), DSP Word Offset = 0x1C (LSW), 0x1D (MSW)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	COMPARE VALUE	Value to be compared to the timer counter.	R/W	0x00000000

The compare logic consists of a 32-bit wide, read/write data TMAR register and logic to compare the counter current value with the value stored in the TMAR register.

Table 56. Timer Synchronization Interface Control Register (GPTMR_TSICR)

MPU Byte Offset = 0x40 (MSW), 0x42 (LSW), DSP Word Offset = 0x20 (LSW), 0x21 (MSW)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:3	RESERVED	Reserved.	R	0x0000000
2	POSTED	1: Posted mode active (clock ratio needs to fit the interface clock frequency > 4 function domain clock frequency requirement) 0: Posted mode inactive.	R/W	1
1	SFT	This bit resets the all of the functional parts of the module. During reset, it always returns 0. 1: Software reset is enabled. 0: Software reset is disabled.	R/W	0
0	RESERVED	Reserved.	R	0

The timer clock has different sources. Based on the timer clock source selection, the reset value can be overwritten for POSTED to perform non-posted read/write accesses.