APPLICATION NOTE

PAC52XX Using Watchdog, SysTick and GP Timers

Power Application Controller™

Marc Sousa
Senior Manager, Systems and Firmware

TABLE OF CONTENTS

OVERVIEW

The PAC52XX family of devices supports a variety of different timer resources to enable flexible program design for a variety of consumer and industrial applications.

This document describes the operation of three of the general-purpose timer resources available in the PAC52XX family:

- Watchdog Timer (WDT)
- General Purpose Timer (GP Timer)
- SysTick Timer (SysTick)

See the table below for a feature comparison for each of these timers.

Timer	Clock Sources	Divider	Number of bits	Deep Sleep Wakeup?
WDT	FRCLK, FCLK	/2 to /65536	24	Y
GP Timer	FRCLK	/2 to /65536	24	Y
SysTick	FCLK / 3, HCLK	n/a	24	N

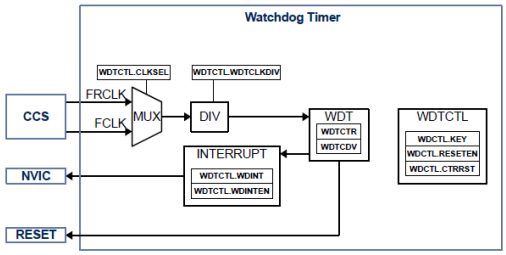
All three of these timer resources can be polled, or can be configured interrupt the ARM Cortex-M0 MCU.

See the sections below for the features and usage for all three of these timers.

WATCHDOG TIMER (WDT)

WDT Overview

The WDT is a 24-bit count-down timer peripheral available in the PAC52XX family of devices. The block diagram for this timer is shown below.



The WDT is a 24-bit count-down timer. This timer may be configured to be clocked either from the FRCLK (input to the PLL) or FCLK (output of the PLL). Because of this, this timer may be used as a wakeup timer when the ARM Cortex-M0 is in deep sleep mode.

The typical use of the Watchdog timer is a low-frequency timer that can reset the device into a safe state if the timer counts down to 0 before being reset. Because of this, the timer divider has a very wide configurable range (from /2 to /65536).

Note that the WDT writes its registers according to its divided input clock, which may be significantly slower than the MCU clock (HCLK). Because of this, after a register write to the WDT, the MCU must check the status of the WDTCTL.WRBUSY bit, before it attempts to write another register. If the user is using the PAC52XX SDK to configure the WDT, this is automatically handled. But if the user is writing to the registers directly, then the user must make sure to check this busy bit in between write operations.

The WDT may be configured to be an interval timer, a watchdog timer or both. See the sections below on each timer mode, and how to write software using the PAC52XX SDK to support these usage cases.

WDT Interval Timer Mode

When configured as an interval timer (WDTCTL.WDINTEN = 1), the WDT counts down from the WDTCDV value (24-bits) to 0. The current timer count is available in the WDTCTR register. To reset the timer, the user writes a pattern to the WDTCTL register, as described in the PAC52XX User Guide. When the timer is reset, the WDTCDV value is assigned to WDTCTR.

If the value of WDTCTR reaches 0 and the WDTCTL.WDTINTEN interrupt enable is set to a 1, the timer sets its interrupt flag (WDTCTL.WDTINT). If the Nested Vectored Interrupt Controller (NVIC) has enabled this interrupt signal, the ARM Cortex-M0 MCU will be interrupt upon this event.

After counting down to 0 the timer will auto-reload the WDTCTR register to the value of WDTCDV and begins counting from that value upon the next clock pulse.

To configure the WDT for FCLK, a clock divider of /64 and interval mode and enable interrupts, the following code may be used:

```
pac5xxx_watchdog_config_clock(WDTCTL_CLKSEL_FRCLK, WDTCTL_PS_DIV64);          // Configure clock input as FRCLK, /64 divider
pac5xxx_watchdog_config(1, 0, countdown_value);                               // Configure interval mode, with CDV

NVIC_EnableIRQ(WDT_IRQn);                                                     // Enable NVIC for the WDT peripheral
_enable_irq();                                                                // Enable global interrupt flag
```

If the user wishes to reset the counter for the WDT, they should call this function

```
pac5xxx_watchdog_reset();
```

When interrupt handler is called (due to WDTCTR counting down to 0), then the WDT IRQ handler is called, as shown below.

```
void WDT_IRQHandler(void)
{

  // WDT interval timer interrupt. Insert code here to handle:

  // TODO

  // OPTIONAL: Disable WDT timer, if we don't need it. If we don't it will auto-reload

  pac5xxx_wdt_config(0, 0, 0);     // interval enable, por enable, count-down value
  NVIC_DisableIRQ(WDT_IRQn);       // Disable NVIC interrupt for WDT

  // Clear interrupt flag to prepare for

  pac5xxx_wdt_clear_if();
}
```

WDT Watchdog Mode

When configured as a watchdog timer (WDTCTL.WDTRSETEN = 1), the WDT counts down from the WDTCDV value (24-bits) to 0. The current timer count is available in the WDTCTR register. To reset the timer, the user needs to write a pattern to the WDTCTL register, as described in the PAC52XX User Guide. The PAC52XX SDK provides a function to reset the WDT. When the timer is reset, the WDTCDV value is assigned to WDTCTR.

If the value of WDTCTR ever reaches 0 and the WDTCTL.WDTRESETEN = 1, then the PAC52XX performs a soft reset of the MCU.

If the user wishes to reset the counter for the WDT, they should call this function

```
pac5xxx_watchdog_reset();
```

WDT Interval and Watchdog Timer Mode

The WDT has a special feature that allows a combination of the interval and watchdog timer modes. If both of these modes are enabled at the same time (WDTCTL.INTEN = 1 and WDTCTL.WDTRESETEN = 1), then the timer operates as follows:

- Timer counts down from WDTCDV to 0
- The first time the timer reaches 0, the WDTCTL.INT flag is set (for an interrupt like when the WDT is configured for interval timer mode)
- The timer then auto-reloads the WDTCDV into WDTCTR and continues to count down

- If the timer reaches 0 a *second* time without being reset, then a soft reset is asserted by the PAC52XX

This feature allows this single timer to be used as both an interval timer and watchdog timer.

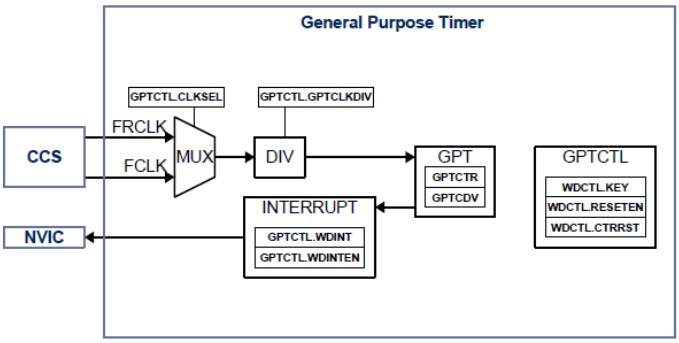
To configure the WDT for both interval and watchdog modes, the user may call the following function:

```
pac5xxx_watchdog_config(1, 1, countdown_value);      // Enable both interval and watchdog modes
```


GENERAL-PURPOSE TIMER (GP TIMER)

GP Timer Overview

The General-Purpose Timer (GP Timer) is a 24-bit count-down timer peripheral available in the PAC52XX family of devices. The block diagram for this timer is shown below.



The GP Timer is a 24-bit count-down timer. This timer may be configured to be clocked either from the FRCLK (input to the PLL) or FCLK (output of the PLL). Because of this, this timer may be used as a wakeup timer when the ARM Cortex-M0 is in deep sleep mode.

Note that the GP Timer writes its registers according to its divided input clock, which may be significantly slower than the MCU clock (HCLK). Because of this, after a register write to the GP Timer, the MCU must check the status of the GPTCTL.WRBUSY bit, before it attempts to write another register. If the user is using the PAC52XX SDK to configure the GP Timer, this is automatically handled. But if the user is writing to the registers directly, then the user must make sure to check this busy bit in between write operations.

The GP Timer may be configured only as an interval timer. See the sections below on how to write software using the PAC52XX SDK to support these usage cases.

GP Timer Interval Mode

When enabled (GPTCTL.GPINTEN = 1), the GP Timer counts down from the GPTCDV value (24-bits) to 0. The current timer count is available in the GPTCTR register. To reset the timer, the user writes a pattern to the GPTCTL register, as described in the PAC52XX User Guide. When the timer is reset, the GPTCDV value is assigned to GPTCTR.

If the value of GPTCTR reaches 0 and the GPTCTL.GPTINTEN interrupt enable is set to a 1, the timer sets its interrupt flag (GPTCTL.GPTINT). If the Nested Vectored Interrupt Controller (NVIC) has enabled this interrupt signal, the ARM Cortex-M0 MCU will be interrupt upon this event.

After counting down to 0 the timer will auto-reload the GPTCTR register to the value of GPTCDV and begins counting from that value upon the next clock pulse.

To configure the GP TImer for FCLK, a clock divider of /64 enable interrupts, the following code may be used[1]:

```
pac5xxx_rtc_config_clock(RTCCTL_CLKSEL_FRCLK, RTCCTL_PS_DIV64);          // Configure clock input as FRCLK, /64 divider
pac5xxx_rtc_config(1, 0, countdown_value);                               // Configure interval mode, with CDV

NVIC_EnableIRQ(RTC_IRQn);                                                // Enable NVIC for the GP Timer peripheral
_enable_irq();                                                           // Enable global interrupt flag
```

If the user wishes to reset the counter for the GP Timer, they should call this function

```
pac5xxx_rtc_reset();
```

When interrupt handler is called (due to GPTCTR counting down to 0), then the GP Timer IRQ handler is called, as shown below.

```
void RTC_IRQHandler(void)
{

  // WDT interval timer interrupt. Insert code here to handle:

  // TODO

  // OPTIONAL: Disable GP Timer, if we don't need it. If we don't it will auto-reload

  pac5xxx_rtc_config(0, 0, 0);     // interval enable, por enable, count-down value
  NVIC_DisableIRQ(RTC_IRQn);       // Disable NVIC interrupt for GP Timer

  // Clear interrupt flag to prepare for

  pac5xxx_rtc_clear_if();
}
```

[1] Note that the PAC52XX SDK uses the prefix "rtc" for the GP Timer.

SysTick Timer (SysTick)

SysTick Overview

The SysTick Timer is a 24-bit count-down timer available in the ARM Cortex-M0 in the PAC52XX family of devices.

This timer may be configured to be clocked either from the FCLK /3 (output of the PLL divided by 3) or HCLK by setting the SYST_CSR register field “CLKSOURCE”. Because this timer has clock sources that are not gated during deep sleep mode, this timer may NOT be used as a wakeup timer when the ARM Cortex-M0 is in deep sleep mode.

The SysTick timer may be configured to generate an interrupt to the ARM Cortex-M0 MCU when it counts down to 0, or it may be sampled.

For more information on the SysTick timer, see the PAC52XX User Guide or the ARM Cortex-M0 Specification.

SysTick Configuration

The SysTick timer is a part of the ARM Cortex-M0 core, so it is configured via the ARM Cortex System Control Block (SCB). To configure the SysTick time for FCLK /3 (default is FCLK /3), the following code may be used:

```
SysTick->LOAD = countdown_time;           // Set the reload value
SysTick->VAL = 0;                         // Set the current value, so next clock is VAL = LOAD

SysTick->CTRL |= SysTick_CTRL_ENABLE_Msk;  // Enable the timer
```

SysTick Interrupts

The user may reset the SysTick timer (and prevent it from counting down to 0) by writing the VAL register as follows:

```
SysTick->VAL = countdown_time;            // Set the current value, so next clock is VAL = LOAD
```

In order to generate an interrupt when the timer counts down to 0, the user should enable the NVIC interrupt for SysTick and the global interrupt flag as follows:

```
SysTick->LOAD = countdown_time;           // Set the reload value
SysTick->VAL = 0;                         // Set the current value, so next clock is VAL = LOAD
NVIC_IRQEnable(SysTick_IRQn);             // Enable the SysTick exception in the NVIC
__enable_irq();                           // Enable global interrupt flag

SysTick->CTRL |= SysTick_CTRL_ENABLE_Msk;  // Enable the timer
```

When the interrupt is received, override the following function:

```
void SysTick_IRQHandler(void)
{
```

```
  // TODO
  ;

}
```


ABOUT ACTIVE-SEMI

Active-Semi, Inc. headquartered in Dallas, TX is a leading innovative semiconductor company with proven power management, analog and mixed-signal products for end-applications that require power conversion (AC/DC, DC/DC, DC/AC, PFC, etc.), motor drivers and control and LED drivers and control along with ARM microcontroller for system development.

Active-Semi's latest family of Power Application Controller (PAC)™ ICs offer high-level of integration with 32-bit ARM Cortex M0, along with configurable power management peripherals, configurable analog front-end with high-precision, high-speed data converters, single-ended and differential PGAs, integrated low-voltage and high-voltage gate drives. PAC IC offers unprecedented flexibility and ease in the systems design of various end-applications such as Wireless Power Transmitters, Motor drives, UPS, Solar Inverters and LED lighting, etc. that require a microcontroller, power conversion, analog sensing, high-voltage gate drives, open-drain outputs, analog & digital general purpose IO, as well as support for wired and wireless communication. More information and samples can be obtained from http://www.active-semi.com or by emailing marketing@active-semi.com

Active-Semi shipped its 1 Billionth IC in 2012, and has over 120 in patents awarded and pending approval.