General Purpose Timer Module

TM4C123GH GPTM

Dr. Munesh Singh

Indian Institute of Information Technology Design and Manufacturing
Kancheepuram, Chennai, Tamil Nadu

February 21, 2020

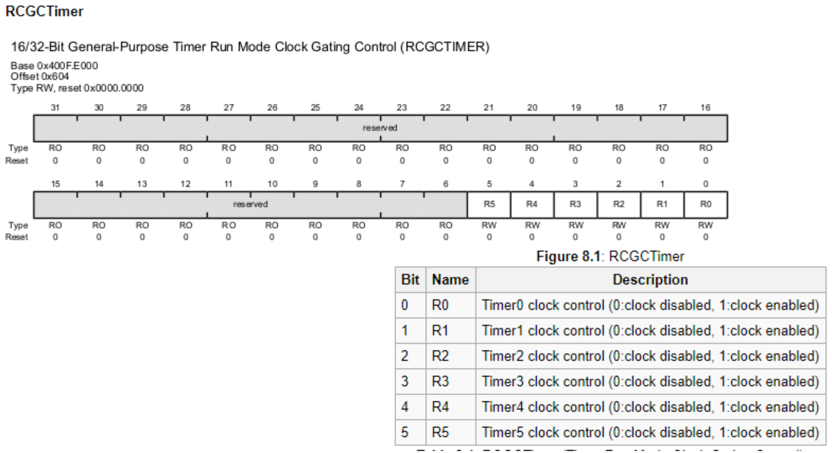
GPIO Timers

- General-Purpose Timer Module (GPTM) contains six 16/32-bit GPTM blocks and six 32/64-bit Wide GPTM blocks
- Each 16/32-bit GPTM block can provide two 16-bit (half-width) timers/counters that are referred to as Timer A and Timer B
- All timers have interrupt controls and separate interrupt vectors as well as separate interrupt handlers.
- GPIO Timer operates in different modes
    - One short/periodic mode
    - Real time clock timer mode
    - Input edge count mode
    - Input edge time mode
    - PWM mode
    - Wait for trigger mode

- The 16/32-bit timer blocks are designated as Timer 0, Timer 1, ..., and Timer 5.
- The following shows the base addresses for the 16/32-bit Timer blocks:
  - 16/32-bit Timer 0 base: 0x4003.0000
  - 16/32-bit Timer 1 base: 0x4003.1000
  - 16/32-bit Timer 2 base: 0x4003.2000
  - 16/32-bit Timer 3 base: 0x4003.3000
  - 16/32-bit Timer 4 base: 0x4003.4000
  - 16/32-bit Timer 5 base: 0x4003.5000
- Each of the Timer blocks contains two timers (Timer A, Timer B)
- TimerA and TimerB each contains a counter
- When the clock is fed to them, they keep counting up/down.

Configure the Timer

- This is done with the RCGCTimer register.



RCGCTimer

16/32-Bit General-Purpose Timer Run Mode Clock Gating Control (RCGCTIMER)

Base 0x400F.E000
Offset 0x604
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8.1: RCGCTimer

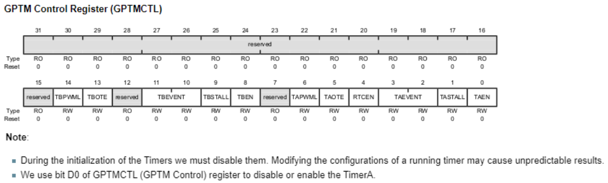
Bit	Name	Description
0	R0	Timer0 clock control (0:clock disabled, 1:clock enabled)
1	R1	Timer1 clock control (0:clock disabled, 1:clock enabled)
2	R2	Timer2 clock control (0:clock disabled, 1:clock enabled)
3	R3	Timer3 clock control (0:clock disabled, 1:clock enabled)
4	R4	Timer4 clock control (0:clock disabled, 1:clock enabled)
5	R5	Timer5 clock control (0:clock disabled, 1:clock enabled)

The General-Purpose Timer Module Registers

- Each GPTM block is composed of different control and status registers.
- These registers can be divided into the following groups:
  - TimerA Control Register group
  - TimerA Status Register group
  - Timers A and B Interrupt and Configuration Register group
  - External Controls group

TimerA Control Register Group

- Eight registers are used to configure and control the operations of the Timer A & Timer B:
  - GPTM Configuration Register (GPTMCFG)
  - GPTM Control Register (GPTMCTL)
  - GPTM Timer A Mode Register (GPTMTAMR)
  - GPTM Timer A Interval Load Register (GPTMTAILR)
  - GPTM Timer A Match Register (GPTMTAMATCHR)
  - GPTM Timer A Prescale Register (GPTMTAPR)
  - GPTM Timer A Prescale Match Register (GPTMTAPMR)
  - GPTM Timer A Prescale Snapshot Register (GPTMTAPS)



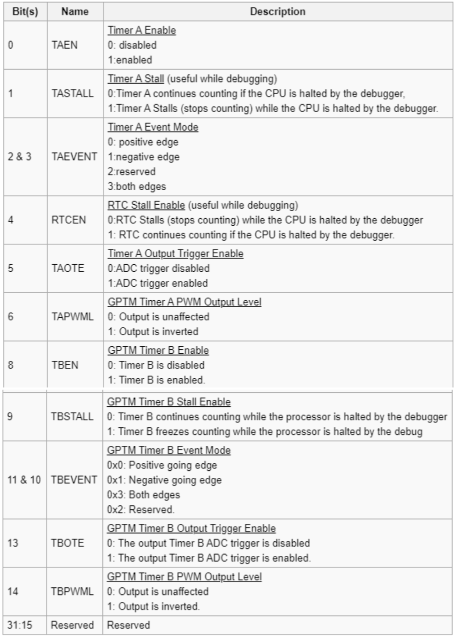
GPTM Control Register (GPTMCTL)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	TBPWML	TBOTE	reserved	TBEVENT		TBSTALL	TBEN	reserved	TAPWML	TAOTE	RTCEN	TAEVENT		TASTALL	TAEN
Type	RO	RW	RW	RO	RW	RW	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note:

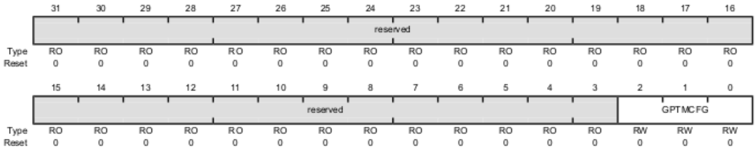
- During the initialization of the Timers we must disable them. Modifying the configurations of a running timer may cause unpredictable results.
- We use bit D0 of GPTMCTL (GPTM Control) register to disable or enable the TimerA.



Bit(s)	Name	Description
0	TAEN	Timer A Enable 0: disabled 1:enabled
1	TASTALL	Timer A Stall (useful while debugging) 0:Timer A continues counting if the CPU is halted by the debugger, 1:Timer A Stalls (stops counting) while the CPU is halted by the debugger.
2 & 3	TAEVENT	Timer A Event Mode 0: positive edge 1:negative edge 2:reserved 3:both edges
4	RTCEN	RTC Stall Enable (useful while debugging) 0:RTC Stalls (stops counting) while the CPU is halted by the debugger 1: RTC continues counting if the CPU is halted by the debugger.
5	TAOTE	Timer A Output Trigger Enable 0:ADC trigger disabled 1:ADC trigger enabled
6	TAPWML	GPTM Timer A PWM Output Level 0: Output is unaffected 1: Output is inverted
8	TBEN	GPTM Timer B Enable 0: Timer B is disabled 1: Timer B is enabled.
9	TBSTALL	GPTM Timer B Stall Enable 0: Timer B continues counting while the processor is halted by the debugger 1: Timer B freezes counting while the processor is halted by the debug
11 & 10	TBEVENT	GPTM Timer B Event Mode 0x0: Positive going edge 0x1: Negative going edge 0x3: Both edges 0x2: Reserved.
13	TBOTE	GPTM Timer B Output Trigger Enable 0: The output Timer B ADC trigger is disabled 1: The output Timer B ADC trigger is enabled.
14	TBPWML	GPTM Timer B PWM Output Level 0: Output is unaffected 1: Output is inverted.
31:15	Reserved	Reserved

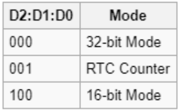
GPTM Configuration Register (GPTMCFG)

- To configure TimerA as 16- or 32-bit, we must use GPTMCFG (GPTM Configuration) register
- To use the 16-bit option we need to have D2:D1:D0=0x4



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

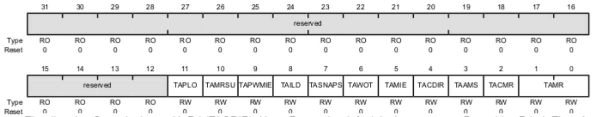
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													GPTMCFG		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



D2:D1:D0	Mode
000	32-bit Mode
001	RTC Counter
100	16-bit Mode

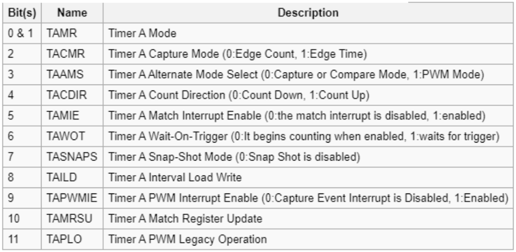
TimerA Mode selection register (GPTMTAMR)

- The mode selection such as periodic, count up/down selection for TimerA is done with GPTM TimerA Mode (GPTMTAMR) register.



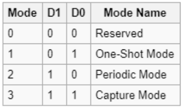
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
reserved
Type RO RO RO RO RO RO RO RO RO RO RO RO RO RO RO RO
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
reserved TAPLO TAMRSU TAPWMIE TAILD TASNAPS TAWOT TAMIE TACDIR TAAMS TACMR TAMR
Type RO RO RO RO RW RW RW RW RW RW RW RW RW RW RW RW
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

The direction Count is done with D4 (TACDIR). Upon Reset, the default is down counter. By making D4=1, TimerA counts up.



Bit(s)	Name	Description
0 & 1	TAMR	Timer A Mode
2	TACMR	Timer A Capture Mode (0:Edge Count, 1:Edge Time)
3	TAAMS	Timer A Alternate Mode Select (0:Capture or Compare Mode, 1:PWM Mode)
4	TACDIR	Timer A Count Direction (0:Count Down, 1:Count Up)
5	TAMIE	Timer A Match Interrupt Enable (0:the match interrupt is disabled, 1:enabled)
6	TAWOT	Timer A Wait-On-Trigger (0:It begins counting when enabled, 1:waits for trigger)
7	TASNAPS	Timer A Snap-Shot Mode (0:Snap Shot is disabled)
8	TAILD	Timer A Interval Load Write
9	TAPWMIE	Timer A PWM Interrupt Enable (0:Capture Event Interrupt is Disabled, 1:Enabled)
10	TAMRSU	Timer A Match Register Update
11	TAPLO	Timer A PWM Legacy Operation

Table 8.4: GPTMTAMR (GPTM Timer A Mode)

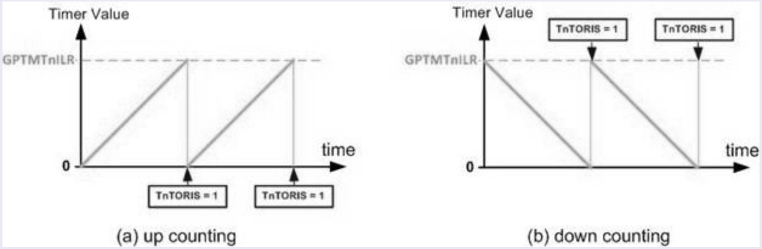


Mode	D1	D0	Mode Name
0	0	0	Reserved
1	0	1	One-Shot Mode
2	1	0	Periodic Mode
3	1	1	Capture Mode

Table 8.5: TAMR Bits of GPTMTAMR

GPTM Timer n Interval Load (GPTMTnILR)

- When the timer is counting down (the timer counts down if the TACDIR bit of the GPTMTAMR register is 0)
- The timer counter begins counting from GPTMTnILR and goes down until it reaches zero.
- the timer counter is reloaded with the value from GPTMTnILR and the TnTORIS flag of the GPTMRIS register is set.



GPTM Timer n Interval Load (GPTMTnILR)

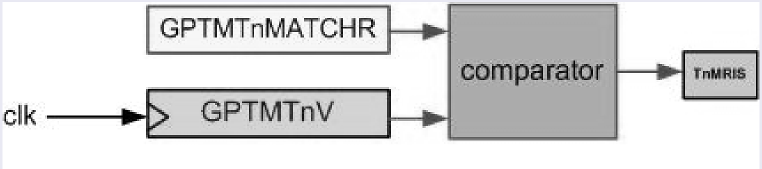
- When the timer is counting up, the timer counter begins counting from 0 and goes up until it reaches the GPTMTnILR value.
- The timer counter is cleared to zero and the TnTORIS flag of the GPTMRIS register is set.
- Upon reset, all bits of the GPTMTnILR register are initialized to 1s which makes the biggest value and has no effect on the timer counting.
- The smaller values for the register leads the timer timeout faster and the TnTORIS flag sets sooner.

Timer A Status Register Group

- Three registers are used to monitor and detect the status of the Timer A:
  - GPTM Timer A Register (GPTMTAR)
  - GPTM Timer A Value Register (GPTMTAV)
  - GPTM Timer A Prescale Value Register (GPTMTAPV).
- GPTMTAV and GPTMTBV are two 16-bit up/down counter registers.
- When a timer is in 16-bit mode, they work as 2 separate counters. When the timer is in 32-bit mode, they are cascaded to form a 32-bit counter.

GPTM TimerA Match Register (GPTMTAMATCHR)

- The GPTM Timer A Match (GPTMTAMATCHR) register can be used to load a match value
- This value can be compared with the current timer value stored in the GPTM Timer A (GPTMTAR) register.
- If both values are equal, the trigger a matching interrupt or set a flag to indicate that a time value matching has occurred.
- The GPTM Timer B has the similar registers and functions.



Note: In the above figure, n is A for TimerA and B for TimerB.
For example GPTMTnV is GPTMTAV in TimerA.

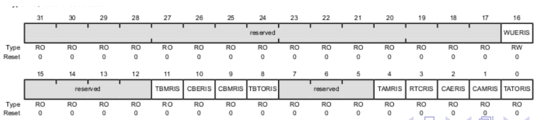
GPTM Timer A Register (GPTMTAR)

GPTMTAR Register

- This register shows the current value of the Timer A counter in all cases except for Input Edge Count and Input Edge Time modes

GPTM Raw Interrupt Status Register (GPTMRIS)

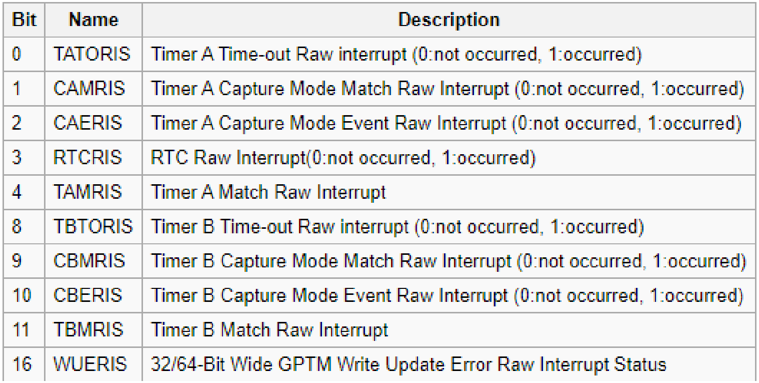
- It monitor and set a raw or internal interrupt if a GPTM-related raw interrupt occurred.
- These bits are set whether or not the interrupt is masked in the GPTMIMR register.
- Only for those bits that have been set on the GPTMIMR register, they can be sent to the NVIC



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															WUERIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				TBMRIS	CBERIS	CBMRIS	TBTORIS	reserved			TAMRIS	RTCRIS	CAERIS	CAMRIS	TATORIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

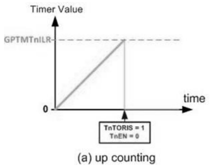
- If a GPTM-related raw interrupt is generated, the corresponding bit on this register is set to 1.
- Each bit can be cleared by writing a 1 to its corresponding bit in GPTMICR register.



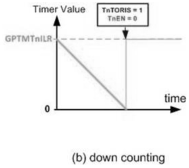
Bit	Name	Description
0	TATORIS	Timer A Time-out Raw interrupt (0:not occurred, 1:occurred)
1	CAMRIS	Timer A Capture Mode Match Raw Interrupt (0:not occurred, 1:occurred)
2	CAERIS	Timer A Capture Mode Event Raw Interrupt (0:not occurred, 1:occurred)
3	RTCRIS	RTC Raw Interrupt(0:not occurred, 1:occurred)
4	TAMRIS	Timer A Match Raw Interrupt
8	TBTORIS	Timer B Time-out Raw interrupt (0:not occurred, 1:occurred)
9	CBMRIS	Timer B Capture Mode Match Raw Interrupt (0:not occurred, 1:occurred)
10	CBERIS	Timer B Capture Mode Event Raw Interrupt (0:not occurred, 1:occurred)
11	TBMRIS	Timer B Match Raw Interrupt
16	WUERIS	32/64-Bit Wide GPTM Write Update Error Raw Interrupt Status

Periodic mode vs. one shot mode

- The timer can be in 4 different modes including the periodic and one shot modes.
- In periodic mode the timer continues counting after each timeout.
- But in one shot mode, the timer stops counting after timeout is reached.
- When it is in up counting and one shot modes, it counts from 0 to GPTMTnILR.
- Goes to zero just once and then the TnEN bit of GPTMCTL is cleared causing the timer to stop



(a) up counting



(b) down counting

- Each timer module has:
  - A clock enable bit, SYSCTL_RCGCTIMER_R
  - A control register, TIMERx_CTL_R
  - A configuration register, TIMERx_CFG_R
  - A mode register, TIMERx_TAMR_R
  - A 32-bit reload register, TIMERx_TAILR_R
  - A resolution (prescale) register, TIMERx_TAPR_R
  - An interrupt clear register, TIMERx_ICR_R
  - An interrupt arm bit, TATOIM, TIMERx_IM_R
  - A flag bit,TATORIS, TIMERx_RIS_R

Setting Up One-Shot/Periodic Timer Mode

(n = A or B and x = 0 to 5)

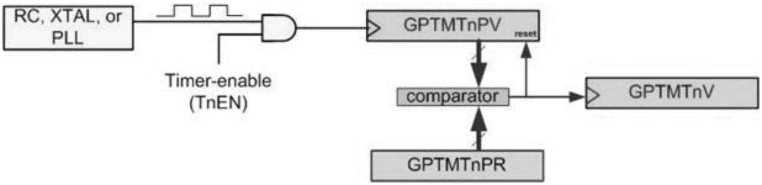
- Disable the selected timer by clearing the TnEN bit in the GPTMCTL register (TIMERx_CTL_R).
- Initialize the GPTMCFG register to set up timer(s) as 16/32-bit timers (TIMERx_CFG_R).
- Configure the TnMR field in the GPTMTnMR (TIMERx_TAMR_R) register by writing
  - 0x1 for one-shot mode.
  - 0x2 for periodic mode.
- Optionally configure the TnSNAPS, TnWOT, TnMTE, and TnCDIR bits in the GPTMTnMR (TIMERx_TAMR_R) register
  - To select whether to capture the value of the free-running timer at timeout
  - Use an external trigger to start counting
  - Configure an additional trigger or interrupt, and count-up/dn.

- Load the start value (time up value) into the GPTMTnILR (TIMERx_TAILR_R).
- GPTMTnPR (TIMERx_TAPR_R, if prescaler is used) registers for the count-down (count-up) operations.
- If interrupts are required, set the appropriate bits in the GPTMIMR (TIMERx_IMR_R) register to enable the selected interrupt source.
- After these initializations and configurations done, set the TnEN bit in the GPTMCTL (TIMERx_CTL_R) register to enable the timer and start counting.
- If no interrupt is used, one can poll the GPTMRIS (TIMERx_RIS_R) register to check the appropriate bits and wait for the time event to occur.

- If an interrupt is used, put appropriate codes inside the interrupt handler to process the interrupt.
- In both cases, the status flags are cleared by writing a 1 to the appropriate bit of the GPTMICR (TIMERx_ICR_R) register.

Prescaler register for Timer A

- The largest time delay we can create with 16 bit timer is about:
  65535 x (1 / 16MHz) = 65535 x 62.5 nsec = 4.096 msec.
- One way to create a longer time delay is to use the prescaler register.
- The prescaler register allows system frequency to be divided by a value between 1 and 256 before it is fed to the TimerA.
- Note the prescaler can yield proper division only when the timer is configured as a down counter.



- For CPU Freq=16MHz calculate the largest delay size using
  - 16-bit TimerA without prescaler and
  - 16-bit TimerA with prescaler.
- Solution: 1/16MHz = 62.5 nsec is period of clock pulses fed to CPU.
  - 65,536 x 62.5 nsec = 4.096 msec for TimerA 16-bit option .
  - 65,536 x 256 x 62.5ns = 1.0485 second for TimerA 16-bit option with the Prescaler.

Use the periodic mode to create a delay:

```
#include <stdint.h>
#include "tm4c123gh6pm.h"
void timer0A_delayMs(int ttime);
int main (void)
{

  SYSCTL_RCGC2_R |= 0x00000020;   /* enable clock to GPIOF at clock gating control register */

  GPIO_PORTF_DIR_R = 0x0E;       /* enable the GPIO pins for the LED (PF3, 2 1) as output */
  GPIO_PORTF_DEN_R = 0x0E;       /* enable the GPIO pins for digital function */

  while(1) {
    GPIO_PORTF_DATA_R = 2;     /* turn on red LED */
    timer0A_delayMs(500);      /* TimerA 500 msec delay */
    GPIO_PORTF_DATA_R = 0;     /* turn off red LED */
    timer0A_delayMs(500);      /* TimerA 500 msec delay */
  }
}
/* multiple of millisecond delay using periodic mode */
void timer0A_delayMs(int ttime){
 int i;

  SYSCTL_RCGCTIMER_R |= 1;    /* enable clock to Timer Block 0 */

  TIMER0_CTL_R = 0;          /* disable Timer before initialization */
  TIMER0_CFG_R = 0x04;        /* 16-bit option */
  TIMER0_TAMR_R = 0x02;       /* periodic mode and down-counter */
  TIMER0_TAILR_R = 16000 - 1;  /* Timer A interval load value register */
  TIMER0_ICR_R = 0x1;         /* clear the TimerA timeout flag*/
  TIMER0_CTL_R |= 0x01;       /* enable Timer A after initialization */

  for(i = 0; i < ttime; i++) {
    while ((TIMER0_RIS_R & 0x1) == 0)
      ;                    /* wait for TimerA timeout flag */
    TIMER0_ICR_R = 0x1;     /* clear the TimerA timeout flag */
  }
}
```

Use prescaler register for TimerA to create 1-second delay

- Using prescaler of 250 gives us 16MHz/250 = 64000Hz. That means the clock fed to timer A is 1/64000Hz = 15.625 msec.
- To get one second delay, we need 1sec/15.625ms = 64000 for the match register.

Use the Prescale mode to create a delay:

```
#include <stdint.h>
#include "tm4c123gh6pm.h"
void timer1A_delaySec(int ttime);
int main (void)
{
   SYSCTL_RCGC2_R |= 0x00000020;  /* enable clock to GPIOF at clock gating control register */
   GPIO_PORTF_DIR_R = 0x0E;       /* enable the GPIO pins for the LED (PF3, 2, and 1) as output */
   GPIO_PORTF_DEN_R = 0x0E;       /* enable the GPIO pins for digital function */
   while(1) {
      GPIO_PORTF_DATA_R = 4;      /* turn on blue LED */
      timer1A_delaySec(1);        /* TimerA 500 msec delay */
      GPIO_PORTF_DATA_R = 0;      /* turn off blue LED */
      timer1A_delaySec(1);        /* TimerA 500 msec delay */
   }
}
/* multiple of second delays using periodic mode and prescaler*/
void timer1A_delaySec(int ttime)
{ int i;
   SYSCTL_RCGCTIMER_R |= 2;       /* enable clock to Timer Block 1 */
   TIMER1_CTL_R = 0;              /* disable Timer before initialization */
   TIMER1_CFG_R = 0x04;           /* 16-bit option */
   TIMER1_TAMR_R = 0x02;          /* periodic mode and down-counter */
   TIMER1_TAILR_R = 64000 - 1;    /* TimerA interval load value reg */
   TIMER1_TAPR_R = 250 - 1;       /* TimerA Prescaler 16MHz/250=64000Hz */
   TIMER1_ICR_R = 0x1;            /* clear the TimerA timeout flag */
   TIMER1_CTL_R |= 0x01;          /* enable Timer A after initialization */
   for(i = 0; i < ttime; i++) {
      while ((TIMER1_RIS_R & 0x1) == 0)
         ;                   /* wait for TimerA timeout flag */
      TIMER1_ICR_R = 0x1;         /* clear the TimerA timeout flag */
   }
}
```

Thank You