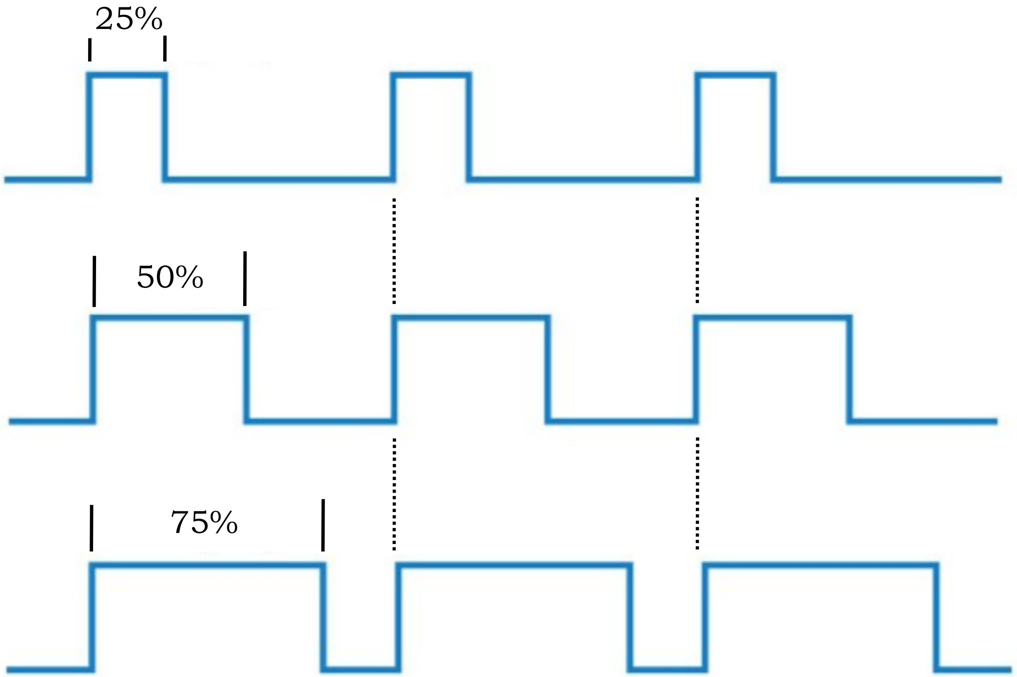
Timer in PWM Mode

16/32 bits, **Count Down**, 8-bit Pre-scaler (Timer Extension)

PWM Signals

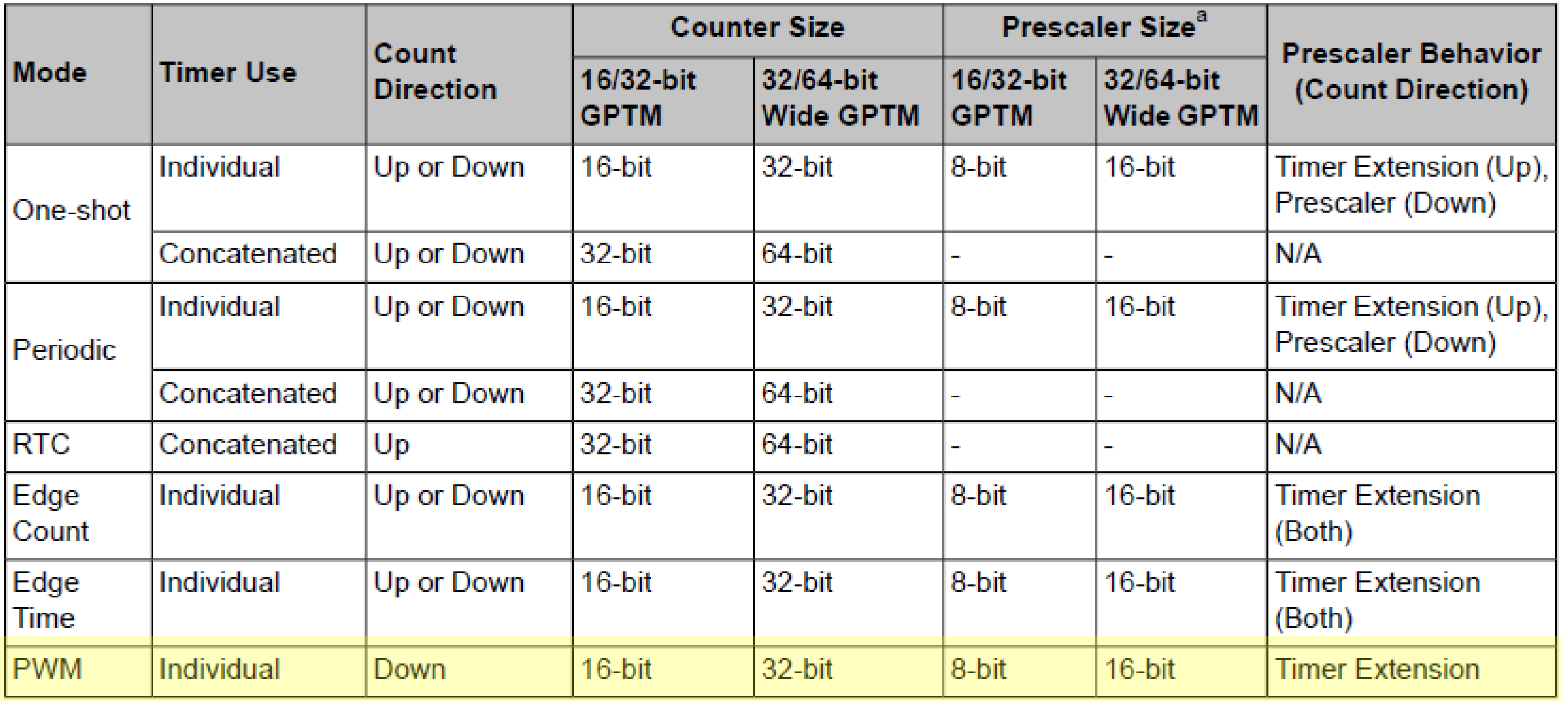
- 25% duty cycle
- 50% duty cycle
- 75% duty cycle



Note: For a PWM signal, period of signal do not change when duty cycle changes.

Timer PWM Mode

Table 11-3. General-Purpose Timer Capabilities



Mode	Timer Use	Count Direction	Counter Size		Prescaler Size[a]		Prescaler Behavior (Count Direction)
			16/32-bit GPTM	32/64-bit Wide GPTM	16/32-bit GPTM	32/64-bit Wide GPTM	
One-shot	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Up), Prescaler (Down)
	Concatenated	Up or Down	32-bit	64-bit	-	-	N/A
Periodic	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Up), Prescaler (Down)
	Concatenated	Up or Down	32-bit	64-bit	-	-	N/A
RTC	Concatenated	Up	32-bit	64-bit	-	-	N/A
Edge Count	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Both)
Edge Time	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Both)
PWM	Individual	Down	16-bit	32-bit	8-bit	16-bit	Timer Extension

a. The prescaler is only available when the timers are used individually

PWM mode: Count **DOWN** only

Prescalar: **MSB** bits of Timer.

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spmu376e.pdf, p708)

Timer PWM Mode – *Timer Operation*

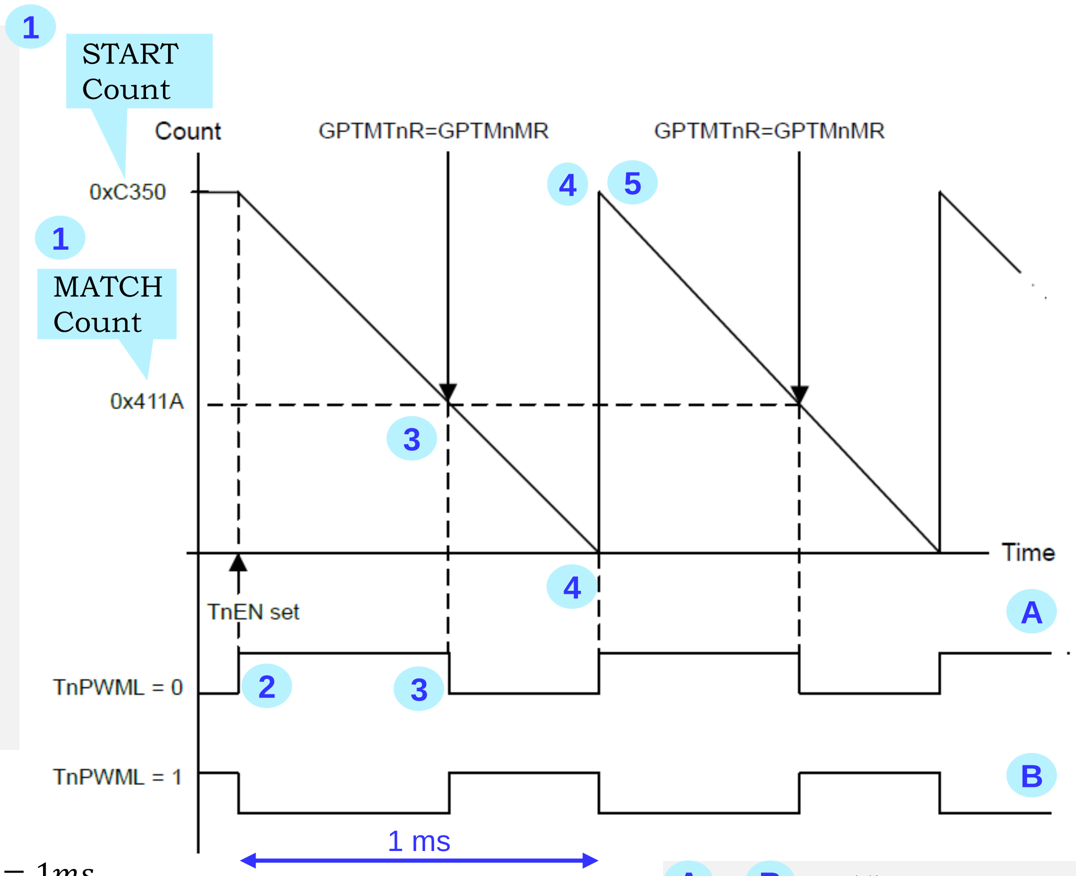
1. Initialize Timer Start value (ILR) to 0xC350; Match value (MATCHR) = 0x411A.
2. Timer output goes to ‘1’.
3. Timer counts down on every clock cycle. When Timer counts down to Match value, output toggles.
4. Timer continues to count down to ‘0’ & then reloads to Start (ILR) value.
5. Cycle repeats.

We can see that **Duty Cycle** of PWM signal can be change by the **Match value**.

Assume 50MHz system bus:

$$PWM\ Period = (0xC350 - 0x0) \times \frac{1}{50MHz} = 1ms$$

$$Duty\ Cycle = \frac{(0xC350 - 0x411A)}{0xC350} = 66.7\%$$



A & B are Timer Outputs. (PWML=0 is default) – bit in CTL reg.

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spmu376e.pdf, p717)

PWM Mode Operation *(Assume Timer A)*

- Timer can be configured as a 24-bit **DOWN** counter.
  - 16 bits (**TAILR**) + 8 bits (**TAPR** - pre-scaler).
- **TAILR** & **TAPR** store the START value of the count-down counter.
- When **TAEN** bit is set, Timer is enabled & it begins to count DOWN until it reaches 0x00.
- In the next counter cycle, it reloads the counter value from **TAILR** & **TAPR** & count down continues.
- Timer Output & Match Event:
  - At start of count down, when Timer counter value equals **TAILR** & **TAPR,** Timer output is ‘1’, and goes to ‘0’ when Timer values equals values in **TnMATCHR** & **TnPMR** regs.

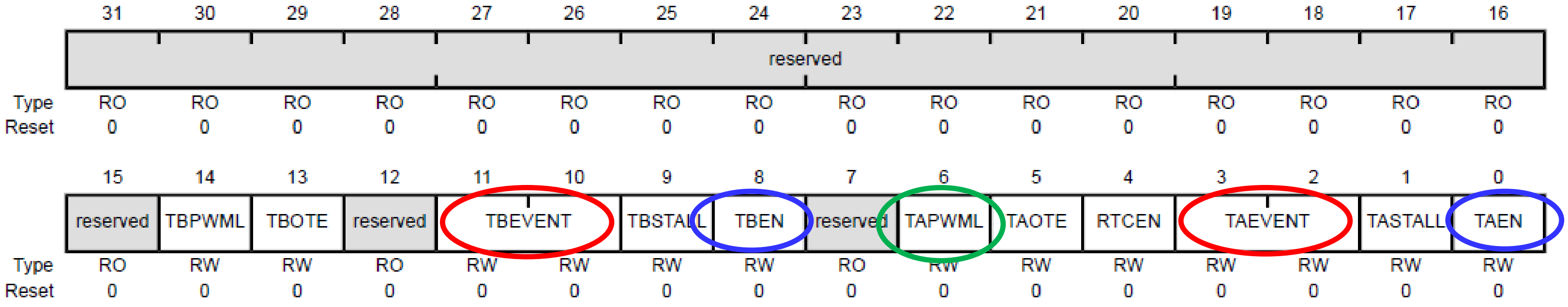
- Timer PWM output can be inverted through **TAPWML** bit (**CTL** reg).

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spms376e.pdf, p708)

Setting Timer to PWM Mode:

Control Register (GPTMCTL)

Type, Reset Value: R/W, 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	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

- Register is used with the **GPTMCFG & GMTMTnMR** to set the timer configuration & enable other timer features.

TAEVENT: Timer A Event Mode
TBEVENT: Timer B Event Mode

- Positive Edge : 0x00
- Negative Edge : 0x01
- Both Edges : 0x03

TAEN: Timer A Enable
TBEN: Timer B Enable

- Timer Disabled : 0
- **Timer Enabled : 1**

PWM Signal Polarity:

- **TAPWML = 1 or 0 : Inverts PWM signal.**

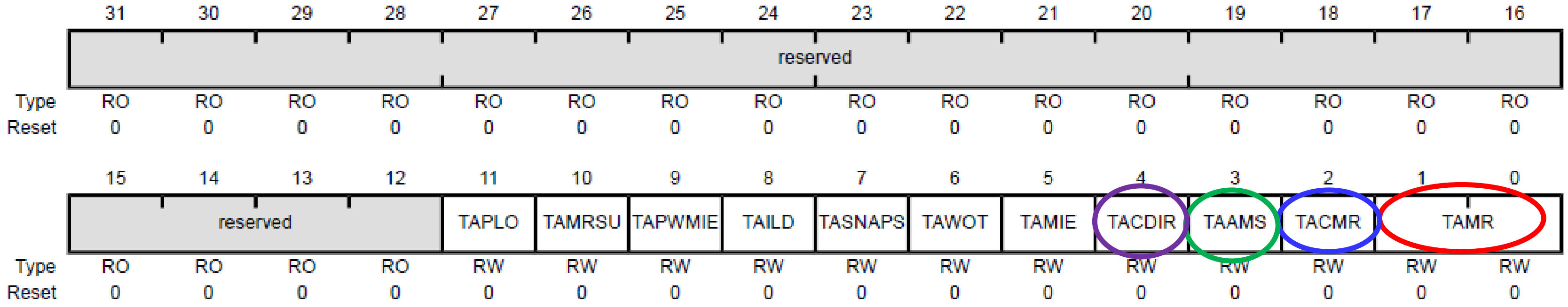
PWM Mode: Set register to values in **BOLD**

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spmu376e.pdf, p737)

Setting Timer to PWM Mode:

Timer A Mode Register (GPTMTAMR)

Type, Reset Value: R/W, 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				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

Timer Mode:

- TAMR = 0x1 : One-Shot
- **TAMR = 0x2 : Periodic Timer**
- TAMR = 0x3 : Capture Mode

Timer Count Direction: (Count Down for PWM)

- **TACDIR = 0** : Count Down (*default*)
- **TACDIR – 1** : Count Up

PWM Interrupt: *(not used in SEM2306)*

- TAAPWMIE = 1 : Enables PWM Interrupt

Timer Capture Mode:

- **TACMR = 0 : Edge-Count Mode**
- TACMR = 1 : Edge-Time Mode

Timer Alternate Mode Select:

- **TAAMS = 1 : Enable PWM mode**
- TAAMS = 0 : Capture or Compare mode

PWM Mode: Set to values in **BOLD**

Timer PWM Registers

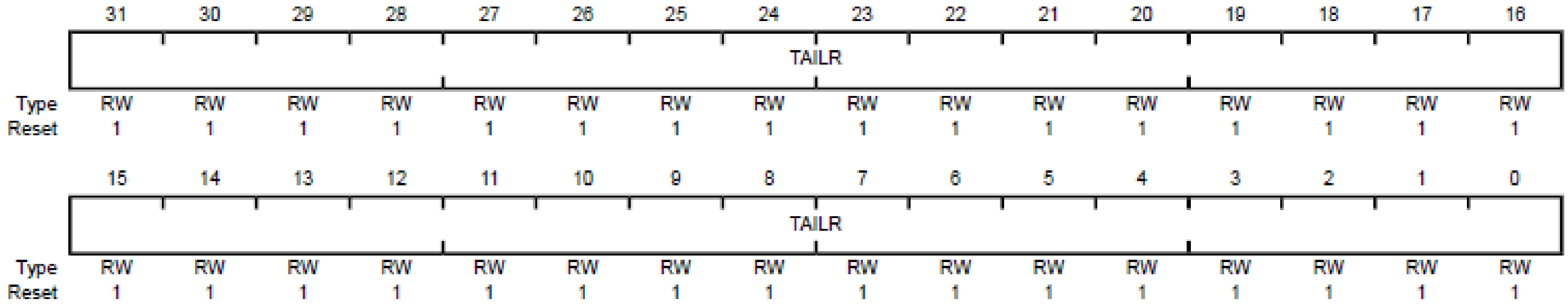
PWM PERIOD: GPTMT*n*ILR, GPTM*n*PR
PWM DUTY-CYCLE: GPTMT*n*MATCHR, GPTMT*n*PMR

PWM Mode Registers

- Four registers that will determine the PWM output waveform:
  - **Start of PWM count** is loaded in:
    - T*n*ILR (Interval Load Register)
    - T*n*PR (Pre-scale Register). – *if used*
  - **Duty cycle of PWM** signal determined by the Match registers:
    - T*n*MATCHR (Match Register)
    - T*n*PMR (Pre-scale Match Register). – *if used.*

Timer A Interval Load Register (GPTMTAILR) *– from Lecture Notes on Timer, Part 1*

Type, Reset Value: R/W, 0xFFFF.FFFF



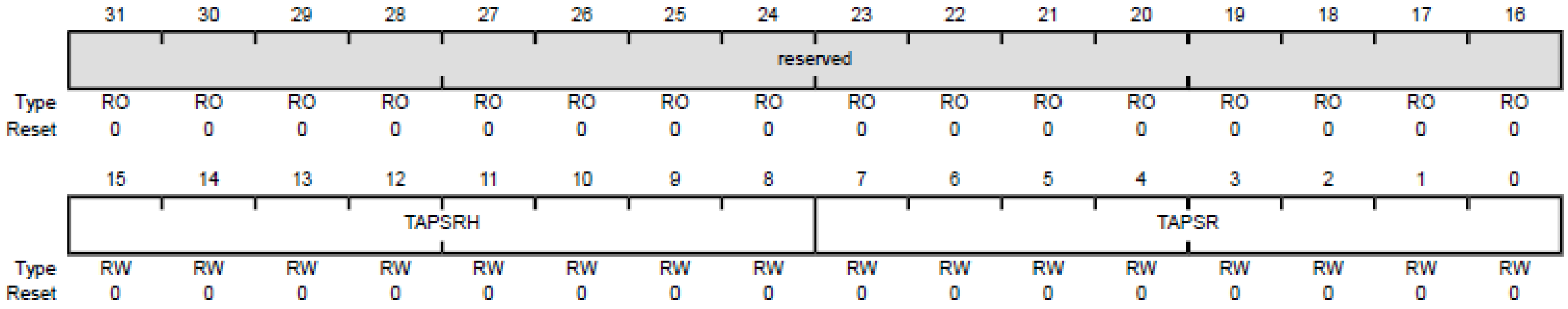
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TAILR															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

- **TAILR** = 32-bit load value.
- When the timer is counting **DOWN**, this register is used to load the **starting count value** into the Timer.
- When the Timer is counting **UP**, this register sets the **upper bound** for the timeout event.

Timer A Prescale Register (GPTMTAPR) – *from Lecture Notes on Timer, Part 1*

Type, Reset Value: RW, 0x0000.0000

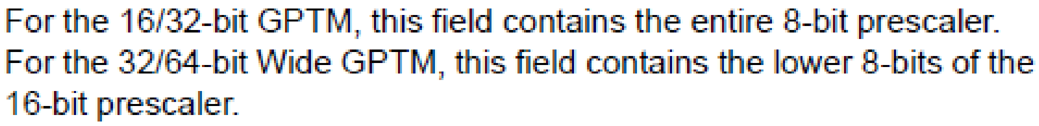


	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	TAPSRH								TAPSR							
Type	RW	RW	RW	RW	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

Register allows software to extend the range of the timers when they are used individually.

Timer count is extended through a **Prescale** value.

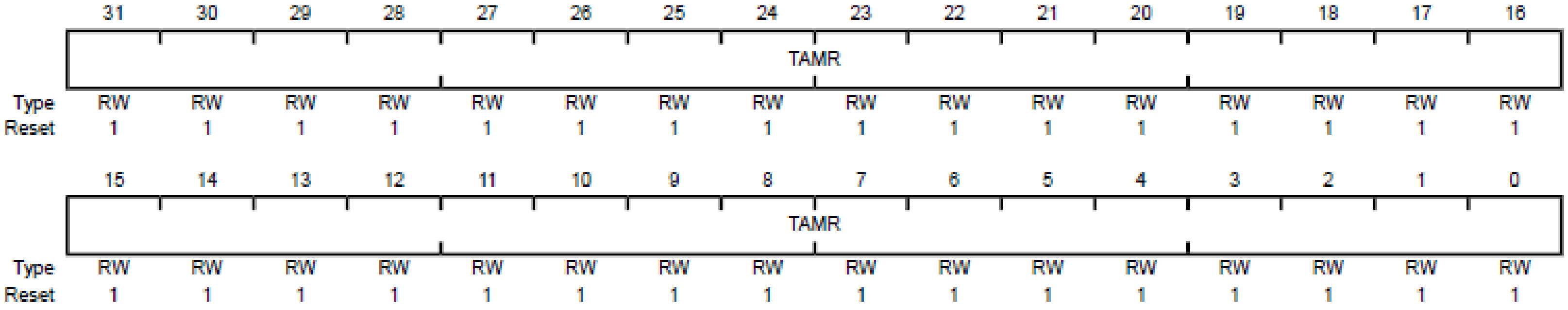
TAPSR: GPTM Timer A Prescale (8-bits)



For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler.
For the 32/64-bit Wide GPTM, this field contains the lower 8-bits of the 16-bit prescaler.

Timer A Match Register (GPTMTAMATCHR)

Type, Reset Value: R/W, 0xFFFF.FFFF



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TAMR															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

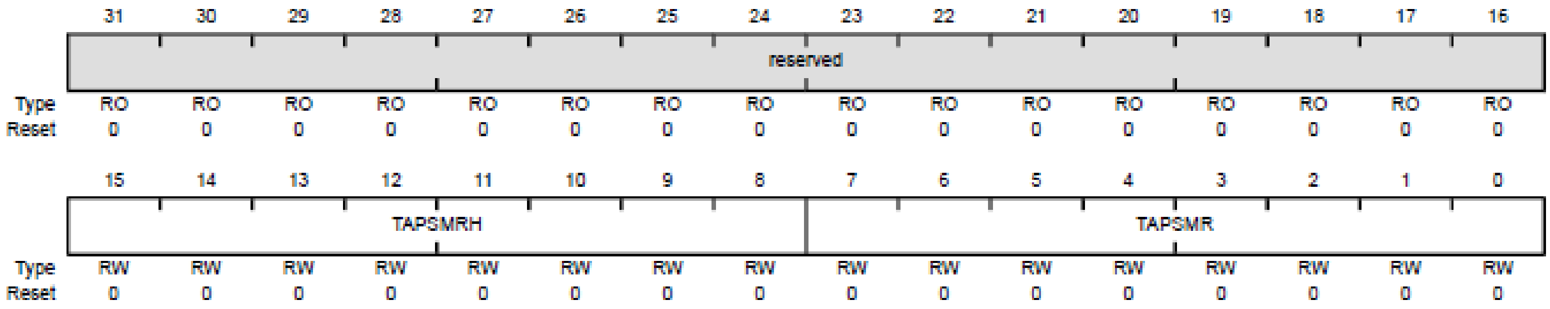
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TAMR															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

This register is loaded with a match value.

Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode

Timer A Prescale Match Register (GPTMTAPMR)

Type, Reset Value: R/W, 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
	TAPSMRH								TAPSMR							
Type	RW	RW	RW	RW	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

- This register allows software to extend the range of the **GPTMTAMATCHR** when the timers are used individually.
- In 16-bit mode, **TAPSMR** contains the 8-bit pre-scaler value.

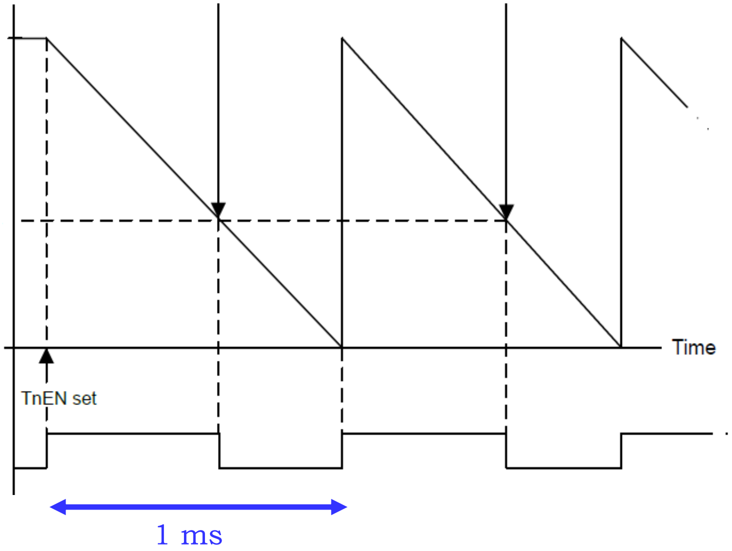
Timer PWM Examples

PWM PERIOD: GPTMT*n*ILR, GPTM*n*PR
PWM DUTY-CYCLE: GPTMT*n*MATCHR, GPTMT*n*PMR

Ex 1: Timer PWM Operation (1KHz, 50%)

If system clock is 80 MHz & we want a PWM output at 1 KHz with duty cycle of 50% :

- At 80MHz bus clock, period = 12.5ns.
- With a 24-bit count DOWN Timer,
  - maximum count = 2^{24}
  - maximum interval = $2^{24} \times 12.5$ns = 209.715 ms.



$$Period\ at\ 1\ KHz = \frac{1}{1000} = 1\ ms$$

$$no\ of\ counts = \frac{1ms}{12.5ns} = 80{,}000 = 0x01.3880$$

Pre-scaler timer extension)

$$duty\ cycle = \frac{(80{,}000 - n)}{80{,}000} = 0.5$$

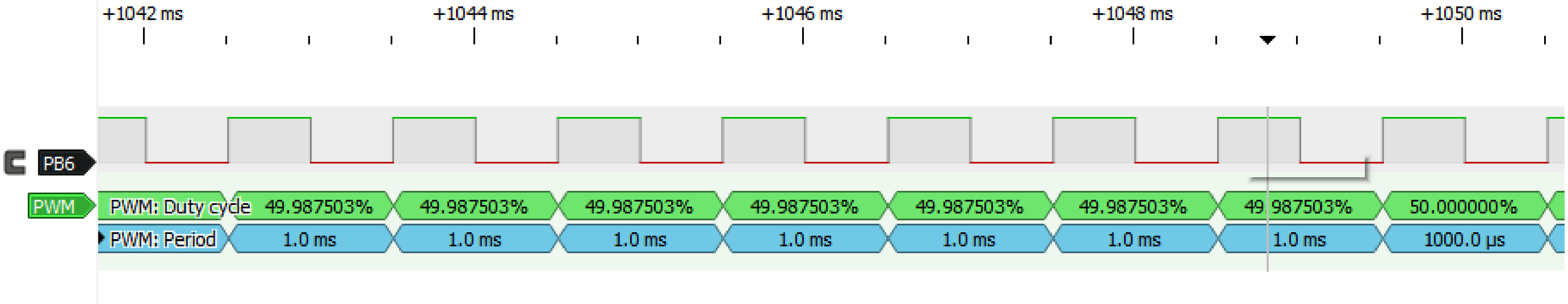
$$n = 40{,}000 = 0x00.9C40$$

Reload (**TAILR**) = 0x0387F
Pre-scaler (**TAPR**) = 0x01
Match (**TAMATCHR**) = 0x09C3F
Pre-scaler Match (**TAPMR**) = 0x00

Ex 1: Timer PWM Operation (1KHz, 50%)

If system clock is 80 MHz & we want a PWM output at 1 KHz with duty cycle of 50% :

Timer PWM output waveform:



Ex 2: Timer PWM Operation (100 Hz, 25%)

If system clock is 80 MHz & we want a PWM output at 100 Hz with duty cycle of 25% :

- At 80MHz bus clock, period = 12.5ns.
- With a 24-bit count down Timer,
  - maximum count = 2^{24}
  - maximum interval = $2^{24} \times 12.5$ns = 209.715 ms.

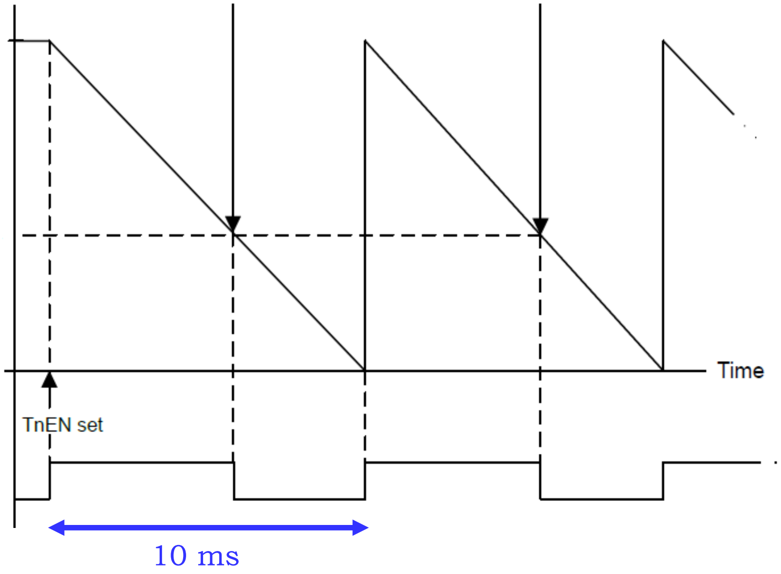
$$Period\ at\ 100\ Hz = \frac{1}{100} = 10\ ms$$

$$no\ of\ counts = \frac{10ms}{12.5ns}$$

$$= 800{,}000 = 0x0C.3500$$

$$duty\ cycle = \frac{(800{,}000 - n)}{800{,}000} = 0.25$$

$$n = 600{,}000 = 0x09.27C0$$



Reload (**TAILR**) = 0x034FF

Pre-scaler (**TAPR**) = 0x0C

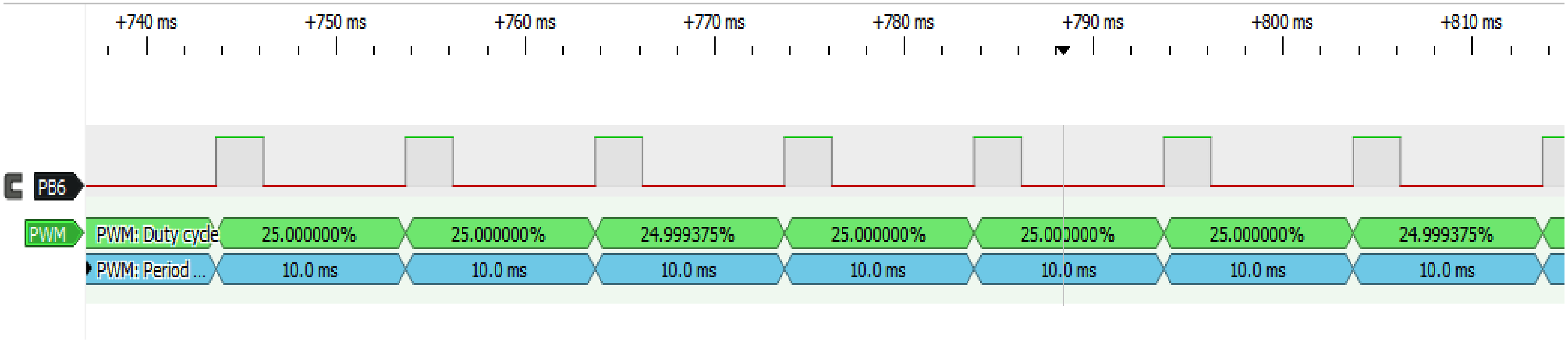
Match (**TAMATCHR**) = 0x027BF

Pre-scaler Match (**TAPMR**) = 0x09

Ex 2: Timer PWM Operation (100 Hz, 25%)

If system clock is 80 MHz & we want a PWM output at 1 KHz with duty cycle of 25% :

Timer PWM output waveform:



Ex 3: Timer PWM Operation (100 KHz, 25%)

If system clock is 80 MHz & we want to get a PWM output at 100 KHz with duty cycle of 25% :

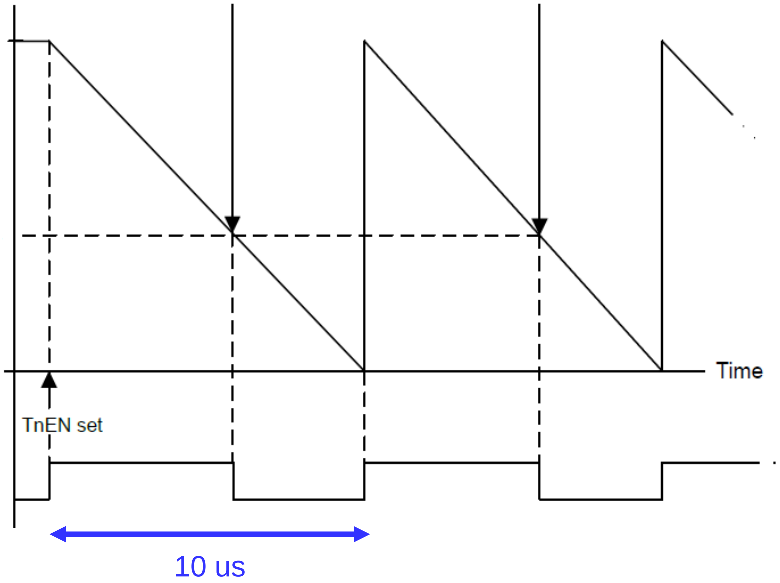
- At 80MHz bus clock, period = 12.5ns.
- With a 24-bit count down Timer,
  - maximum count = 2^{24}
  - maximum interval = $2^{24} \times 12.5$ns = 209.715 ms.

$$Period\ at\ 100\ KHz = \frac{1}{100{,}000} = 10\ us$$

$$no\ of\ counts = \frac{10us}{12.5ns} = 800 = 0x0320$$

$$duty\ cycle = \frac{(800 - n)}{800} = 0.25$$

$$n = 600 = 0x0258$$



No need for pre-scaler!

Reload (**TAILR)** = 0x31F
Pre-scaler (**TAPR**) = 0x00
Match (**TAMATCHR)** = 0x0257
Pre-scaler Match (**TAPMR**) = 0x00

Ex 4: Timer PWM Operation (800 KHz, 33%)

If system clock is 80 MHz & we want to get a PWM output at 800 KHz with duty cycle of 33% :

- At 80MHz bus clock, period = 12.5ns.
- With a 24-bit count down Timer,
  - maximum count = 2^{24}
  - maximum interval = $2^{24} \times 12.5\text{ns}$ = 209.715 ms.

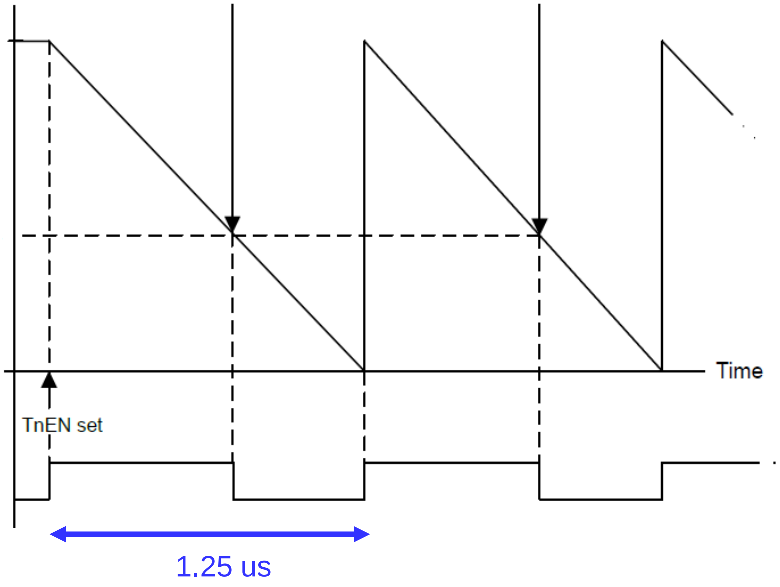
$$Period\ at\ 800\ KHz = \frac{1}{800{,}000} = 1.25\ us$$

$$no\ of\ counts = \frac{1.25us}{12.5ns} = 100 = 0x0064$$

No need for pre-scaler!

$$duty\ cycle = \frac{(100 - n)}{100} = 0.33$$

$$n = 67 = 0x0043$$



Reload (**TAILR**) = 0x0063
Pre-scaler (**TAPR**) = 0x00
Match (**TAMATCHR**) = 0x0042
Pre-scaler Match (**TAPMR**) = 0x00

Timer PWM Mode Initialization

TM4C123GH6PM7

Timer PWM Mode (16-bit) – *Initialization Example*

```
#define PB_T0CCP0   6U        /* PB6  */
void Port_Init( void ){
    /* Enable clocks to peripherals */
    SYSCTL->RCGCTIMER |= SYSCTL_RCGCTIMER_R0;  /* Timer 0     */
    SYSCTL->RCGCGPIO |= SYSCTL_RCGCGPIO_R1     /* GPIO Port B */
    /* Wait for peripherals to be ready */
    while( 0 == (SYSCTL->PRTIMER & SYSCTL_PRTIMER_R1) );
    while( 0 == (SYSCTL->PRGPIO & SYSCTL_PRGPIO_R1) );

    /* setup Timer 0 GPIO pin at PB6 */
    GPIOB->AFSEL |= BIT (PB_T0CCP0);     // enable alternate function
    GPIOB->PCTL |= GPIO_PCTL_PB6_T0CCP0; // enable timer 0 in PB6
    GPIOB->DEN |= BIT (PB_T0CCP0);       // enable PB6
    GPIOB->DIR |= BIT (PB_T0CCP0);       // set as output

    /* setup timer 0 t0 PWM mode        */
    TIMER0->CTL &= ~TIMER_CTL_TAEN;     // disable timer 0 during setup
    TIMER0->CFG |= TIMER_CFG_16_BIT;    // set to 16-bit mode
    /* TAAMS-enable PWM; TAMR-Periodic mode       */
    TIMER0->TAMR |= TIMER_TAMR_TAAMS | TIMER_TAMR_TAMR_PERIOD;
    TIMER0->TAMR &= ~TIMER_TAMR_TACMR;  // TACMR-Edge-Count mode

    TIMER0->TAPR = 0x0; /* prescaler (8 bits)  */
    TIMER0->TAILR = 0x063F;             // TAILR-pulse period
    TIMER0->TAMATCHR = 0x042F;
    TIMER0->TAPMR =  0x0;;
    TIMER0->CTL |= TIMER_CTL_TAEN;      // enable timer 0
}
```

Review Questions

1. Can a Timer configured in PWM mode count UP?
2. Describe how the prescaler is used in Timer PWM mode.
3. Describe how a Timer works in **PWM mode**.
4. A system has a system clock of **80 MHz**. We want to generate a **PWM output** waveform at **Timer 0 (GPIO PB6)** with a frequency of **50 KHz** and with duty cycle of **33%**. What are values we should load to GPTMTAILR, GPTMTAPR, GPTMTAMATCHR and GPTMTAPMR registers
5. A system has a system clock of **50 MHz**. We want to generate a **PWM output** waveform at **Timer 2 (GPIO PB0)** with a frequency of **500 Hz** and with duty cycle of **50%**. What are values we should load to GPTMTAILR, GPTMTAPR, GPTMTAMATCHR and GPTMTAPMR registers?