General Purpose Timer Module (GPTM)

Part 1: Periodic Mode

Applications of Timers

- In embedded applications, *time* can be represented by the *count* of a Timer, based on a system clock.
- Timers are an integral part of embedded systems & programming.
- Many tasks or applications would be difficult to implement without use of Timers.
- Timers in embedded systems are used for:
  - Task scheduling (through periodic interrupts).
  - Pulse period or width measurements.
  - Event counting.
  - Elapsed time determination between events.
  - Waveform generation.
  - Frequency & duty cycle measurement of periodic signals.
  - ….

SysTick Timer vs General-Purpose Timers

- SysTick Timer is part of the ARM-Core definition. Almost all ARM Cortex-M processors have a SysTick timer.
- Because SysTick is part of the ARM Core, it is usually used by the OS for task scheduling.
- General-Purpose Timers (or just Timers) are implemented as part of a microcontroller's peripherals.
  - It is specific to the microcontroller/manufacturer and not part of the ARM core.
- SysTick Timer is usually simpler in design and therefore easier to program.
- General-Purpose Timers are more flexible, have more features and have I/O pins to interface externally.

TM4C123G Timers

- In the Tiva TM4C123G, the Timers are called **General-Purpose Timer Modules** (GPTM).
- There are total of:
  - Six 16/32-bit GPTM modules, &
  - Six 32/64-bit GPTM modules.

For CS397, we will focus on the 16/32 bit GPTM modes.

TM4C123G Timers A & B

- Total of **six** 16/32 bit Timers: **Timer 0** to **Timer 5**.
- Each 16/32-bit Timer block contains TWO Timers: **Timer A** & **Timer B**.
- Each Timer A & B contains its own Counter and can work independently.

Timer	Up/Down Counter
16/32-Bit Timer 0	Timer A
	Timer B
16/32-Bit Timer 1	Timer A
	Timer B
16/32-Bit Timer 2	Timer A
	Timer B
16/32-Bit Timer 3	Timer A
	Timer B
16/32-Bit Timer 4	Timer A
	Timer B
16/32-Bit Timer 5	Timer A
	Timer B

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spms376e.pdf, p706)

TM4C123G Timers A & B

- Each **16/32-bit** GPTM is made up of TWO 16-bit modules (A and B), called **Timer A** & **Timer B**.
- We have the following:
  - **Timer 0A** & **Timer 0B** (*Base address: 0x4003.0000)*
  - **Timer 1A** & **Timer 1B** (*Base address: 0x4003.1000)*
  - **Timer 2A** & **Timer 2B** (*Base address: 0x4003.2000)*
  - **Timer 3A** & **Timer 3B** (*Base address: 0x4003.3000)*
  - **Timer 4A** & **Timer 4B** (*Base address: 0x4003.4000)*
  - **Timer 5A** & **Timer 5B** (*Base address: 0x4003.5000)*

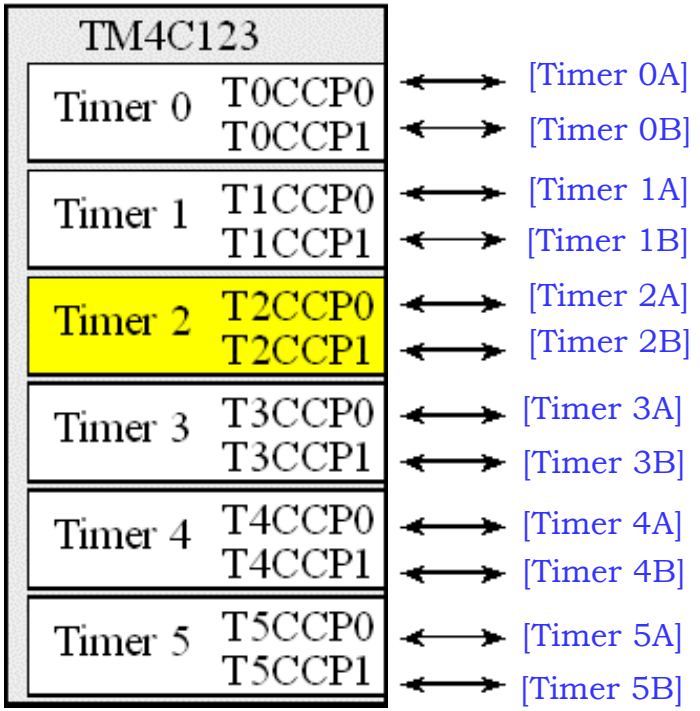
- Timer A & Timer B can be programmed to operate independently of each other:
  - They can work as **two 16-bit Timers**, or, together as **one 32-bit Timer**.

TM4C123G Timer Pins

16/32-bit GPTM modules:

- Each Timer has an associated bi-directional pin:
  - T0CCP0 for Timer 0A
  - T0CCP1 for Timer 0B.
  - T1CCP0 for Timer 1A
  - T1CCP1 for Timer 1B.
  - T2CCP0 for Timer 2A
  - T2CCP1 for Timer 2B.

  ..... *and so on.*



CCP is short for '**C**apture **C**ompare **P**WM'.

TM4C123G Timers (32/64-bits)

- Similarly, each **32-/64-bit** modules is made up of TWO 32-bit modules (A and B), called **Timer A** & **Timer B**.
  - Wide Timer 0A & Wide Timer 0B
    - *Base address: 0x4003.6000*
  - Wide Timer 1A & Wide Timer 1B
    - *Base address: 0x4003.7000*
  - Wide Timer 2A & Wide Timer 2B
    - *Base address: 0x4004.C000*
  - Wide Timer 3A & Wide Timer 3B
    - *Base address: 0x4004.D000*
  - Wide Timer 4A & Wide Timer 4B
    - *Base address: 0x4004.E000*
  - Wide Timer 5A & Wide Timer 5B
    - *Base address: 0x4004.F000*

TM4C123G Timers A & B Signals

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
T0CCP0	1 28	PB6 (7) PF0 (7)	I/O	TTL	16/32-Bit Timer 0 Capture/Compare/PWM 0.
T0CCP1	4 29	PB7 (7) PF1 (7)	I/O	TTL	16/32-Bit Timer 0 Capture/Compare/PWM 1.
T1CCP0	30 58	PF2 (7) PB4 (7)	I/O	TTL	16/32-Bit Timer 1 Capture/Compare/PWM 0.
T1CCP1	31 57	PF3 (7) PB5 (7)	I/O	TTL	16/32-Bit Timer 1 Capture/Compare/PWM 1.
T2CCP0	5 45	PF4 (7) PB0 (7)	I/O	TTL	16/32-Bit Timer 2 Capture/Compare/PWM 0.
T2CCP1	46	PB1 (7)	I/O	TTL	16/32-Bit Timer 2 Capture/Compare/PWM 1.
T3CCP0	47	PB2 (7)	I/O	TTL	16/32-Bit Timer 3 Capture/Compare/PWM 0.
T3CCP1	48	PB3 (7)	I/O	TTL	16/32-Bit Timer 3 Capture/Compare/PWM 1.
T4CCP0	52	PC0 (7)	I/O	TTL	16/32-Bit Timer 4 Capture/Compare/PWM 0.
T4CCP1	51	PC1 (7)	I/O	TTL	16/32-Bit Timer 4 Capture/Compare/PWM 1.
T5CCP0	50	PC2 (7)	I/O	TTL	16/32-Bit Timer 5 Capture/Compare/PWM 0.
T5CCP1	49	PC3 (7)	I/O	TTL	16/32-Bit Timer 5 Capture/Compare/PWM 1.

GPIO pin
(alternate function)

PMC*n*
(in GPIOPCTL register)

All Timer CCP pins are multiplexed with GPIO pins.

TM4C123G Timer Modes

- Timers operates in one of the following four modes:
  - **Periodic mode** / One-shot mode :
    - Timer continues counting UP/DOWN after each timeout (Periodic);
    - Stops when Time-out event reached (One-shot).
  - Input Edge Count or Edge Time:
    - counts UP/DOWN with 16-bit pre-scalar.
  - **PWM Mode**:
    - Timer used with pre-scalar, counts DOWN only.
  - RTC mode:
    - Timer A & B are concatenated, counter counts UP at 32.768 KHz, reloads after terminal count is reached.

TM4C123G Timers

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

Note: **Pre-scaler** is only available when the Timers are used **INDIVIDUALLY** – *not available when Timers are concatenated.*

Use this table as a reference for the Timer modes.
This table will be provided during Quiz/Exam.

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

GPTM Timer Registers

Not all registers are listed in the slides

GPTM Timer Registers

- Each GPTM module has its own set of **Control** & **Status** registers.
- Some registers are denoted by A & B and they both work in the same manner.
- Timer Registers can be grouped by functions:
  - **Timer Control/Configuration.**
  - **Timer Count Status.**
  - **Timer Interrupt Control/Status.**

(… more details in following slide …)

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spmu376e.pdf, p726)

Timer Registers *(grouped by functions)*

- **Timer Control/Configuration Registers:**
  - GPTM Configuration (**GPTMCFG**)
  - GPTM Control (**GPTMCTL**)
  - GPTM Mode (**GPTMTAMR** –*Timer A*, **GPTMTBMR** –*Timer B*)
  - GPTM Interval Load (**GPTMTAILR** –*Timer A*, **GPTMTBILR** –*Timer B*)
  - GPTM Match (**GPTMTAMATCHR** –*Timer A*, **GPTMTBMATCHR** –*Timer B*)
  - GPTM Prescale (**GPTMTAPR** –*Timer A*, **GPTMTBPR** – *Timer B*)
  - GPTM Prescale Match (**GPTMTAPMR** -*Timer A*, **GPTMTBPMR** -*Timer B*)
  - GPTM Prescale Snapshot (**GPTMTAPS** -*Timer A*, **GPTMTBPS** -*Timer B*)
- **Timer Count Status Registers:**
  - GPTM Timer Register (**GPTMTAR** –*Timer A*, **GPTMTBR** –*Timer B*)
  - GPTM Timer Value Register (**GPTMTAV** –*Timer A*, **GPTMTBV** –*Timer B*)
  - GPTM Prescale Value Register (**GPTMTAPV** –*Timer A*, **GPTMTBPV** –*Timer B*)
- **Timer Interrupt Control/Status Registers:**
  - GPTM Interrupt Mask (**GPTMIMR**)
  - GPTM Masked Interrupt Status (**GPTMMIS**)
  - GPTM Interrupt Clear (**GPTMICR**)

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spmu376e.pdf, p726)

Timer Register Map *(1/2)*

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	RW	0x0000.0000	GPTM Configuration	727
0x004	GPTMTAMR	RW	0x0000.0000	GPTM Timer A Mode	729
0x008	GPTMTBMR	RW	0x0000.0000	GPTM Timer B Mode	733
0x00C	GPTMCTL	RW	0x0000.0000	GPTM Control	737
0x010	GPTMSYNC	RW	0x0000.0000	GPTM Synchronize	741
0x018	GPTMIMR	RW	0x0000.0000	GPTM Interrupt Mask	745
0x01C	GPTMRIS	RO	0x0000.0000	GPTM Raw Interrupt Status	748
0x020	GPTMMIS	RO	0x0000.0000	GPTM Masked Interrupt Status	751
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear	754
0x028	GPTMTAILR	RW	0xFFFF.FFFF	GPTM Timer A Interval Load	756
0x02C	GPTMTBILR	RW	-	GPTM Timer B Interval Load	757
0x030	GPTMTAMATCHR	RW	0xFFFF.FFFF	GPTM Timer A Match	758
0x034	GPTMTBMATCHR	RW	-	GPTM Timer B Match	759
0x038	GPTMTAPR	RW	0x0000.0000	GPTM Timer A Prescale	760
0x03C	GPTMTBPR	RW	0x0000.0000	GPTM Timer B Prescale	761
0x040	GPTMTAPMR	RW	0x0000.0000	GPTM TimerA Prescale Match	762
0x044	GPTMTBPMR	RW	0x0000.0000	GPTM TimerB Prescale Match	763

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spmu376e.pdf, p726)

Timer Register Map *(2/2)*

Offset	Name	Type	Reset	Description	See page
0x048	GPTMTAR	RO	0xFFFF.FFFF	GPTM Timer A	764
0x04C	GPTMTBR	RO	-	GPTM Timer B	765
0x050	GPTMTAV	RW	0xFFFF.FFFF	GPTM Timer A Value	766
0x054	GPTMTBV	RW	-	GPTM Timer B Value	767
0x058	GPTMRTCPD	RO	0x0000.7FFF	GPTM RTC Predivide	768
0x05C	GPTMTAPS	RO	0x0000.0000	GPTM Timer A Prescale Snapshot	769
0x060	GPTMTBPS	RO	0x0000.0000	GPTM Timer B Prescale Snapshot	770
0x064	GPTMTAPV	RO	0x0000.0000	GPTM Timer A Prescale Value	771
0x068	GPTMTBPV	RO	0x0000.0000	GPTM Timer B Prescale Value	772
0xFC0	GPTMPP	RO	0x0000.0000	GPTM Peripheral Properties	773

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spmu376e.pdf, p726)

Timer Data Structure *[**file:** TM4C123GH6PM7.h]*

0x4003.0000	CFG
0x4003.0004	TAMR
0x4003.0008	TBMR
0x4003.000C	CTL
0x4003.0010	SYNC
0x4003.0014	RESERVED
0x4003.0018	IMR
0x4003.001C	RIS
0x4003.0020	MIS
0x4003.0024	ICR
0x4003.0028	TAILR
0x4003.002C	TBILR
0x4003.0030	TAMATCHR
0x4003.0034	TBMATCHR
0x4003.0038	TAPR
0x4003.003C	TBPR
0x4003.0040	TAPMR
0x4003.0044	TBPMR
0x4003.0048	TAR
0x4003.004C	TBR
0x4003.0050	TAV
0x4003.0054	TBV
0x4003.0058	RTCPD
0x4003.005C	TAPS
0x4003.0060	TBPS
0x4003.0064	TAPV
0x4003.0068	TBPV
	RESERVED
0x4003.0FC0	PP

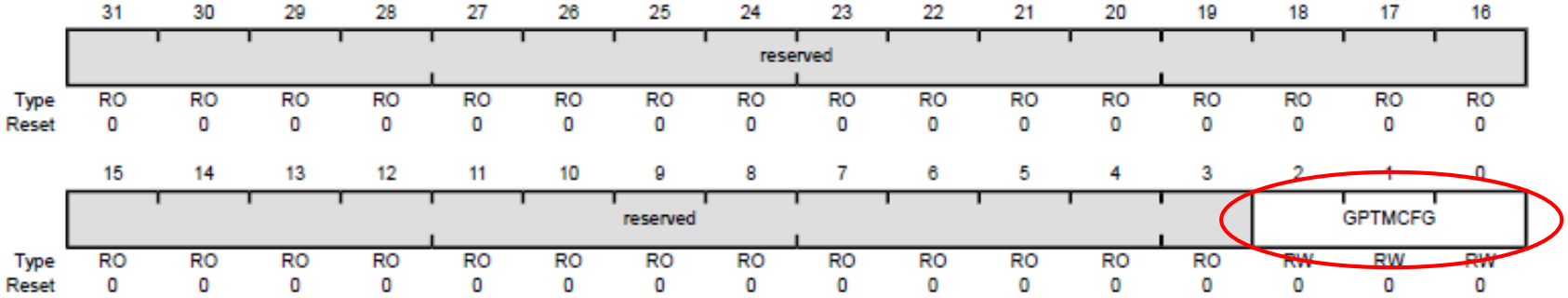
```
typedef struct {             /* TIMER0 Structure      */
  __IO uint32_t  CFG;        /* GPTM Configuration    */
  __IO uint32_t  TAMR;       /* GPTM Timer A Mode     */
  __IO uint32_t  TBMR;       /* GPTM Timer B Mode     */
  __IO uint32_t  CTL;        /* GPTM Control          */
  __IO uint32_t  SYNC;       /* GPTM Synchronize      */
  __I  uint32_t  RESERVED;
  __IO uint32_t  IMR;        /* GPTM Interrupt Mask   */
  __IO uint32_t  RIS;        /* GPTM Raw Interrupt Status      */
  __IO uint32_t  MIS;        /* GPTM Masked Interrupt Status   */
  __O  uint32_t  ICR;        /* GPTM Interrupt Clear           */
  __IO uint32_t  TAILR;      /* GPTM Timer A Interval Load     */
  __IO uint32_t  TBILR;      /* GPTM Timer B Interval Load     */
  __IO uint32_t  TAMATCHR;   /* GPTM Timer A Match             */
  __IO uint32_t  TBMATCHR;   /* GPTM Timer B Match             */
  __IO uint32_t  TAPR;       /* GPTM Timer A Prescale          */
  __IO uint32_t  TBPR;       /* GPTM Timer B Prescale          */
  __IO uint32_t  TAPMR;      /* GPTM TimerA Prescale Match     */
  __IO uint32_t  TBPMR;      /* GPTM TimerB Prescale Match     */
  __IO uint32_t  TAR;        /* GPTM Timer A                   */
  __IO uint32_t  TBR;        /* GPTM Timer B                   */
  __IO uint32_t  TAV;        /* GPTM Timer A Value             */
  __IO uint32_t  TBV;        /* GPTM Timer B Value             */
  __IO uint32_t  RTCPD;      /* GPTM RTC Predivide             */
  __IO uint32_t  TAPS;       /* GPTM Timer A Prescale Snapshot */
  __IO uint32_t  TBPS;       /* GPTM Timer B Prescale Snapshot */
  __IO uint32_t  TAPV;       /* GPTM Timer A Prescale Value    */
  __IO uint32_t  TBPV;       /* GPTM Timer B Prescale Value    */
  __I  uint32_t  RESERVED1[981];
  __IO uint32_t  PP;         /* GPTM Peripheral Properties     */
} TIMER0_Type;
```

Timer Configuration Registers

GPTMCFG, GPTMCTL, GPTMTnMR

Configuration Register (GPTMCFG)

Type, Reset Value: R/W, 0x0000.0000



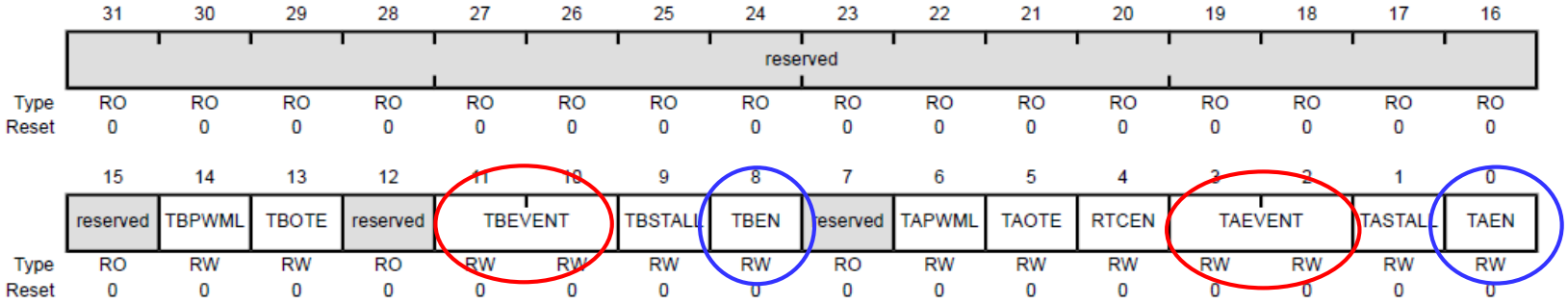
- Register configures the global operation of the GPTM module.
- It sets the GPTM to be:
  - 32- or 64-bit mode (concatenated timers), or
  - 16- or 32-bit mode (individual, split timers), or
  - Real-Timer-Clock (RTC).
- GPTMCFG bits definition:

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

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spmu376e.pdf, p727)

Control Register (GPTMCTL)

Type, Reset Value: R/W, 0x0000.0000



- Register is used with the **GPTMCFG & GMTMTnMR** to set the timer configuration & enable other timer features. GPTMCTL for both timers A & B.
- Output Trigger Enable (TAOTE, TBOTE) can be used to initiate transfers on the ADC module.

TAEVENT: Timer A Event Mode
TBEVENT: Timer B Event Mode
- Positive Edge : 0x00
- Negative Edge : 0x01
- Both Edge : 0x03

(Used in Input Capture Modes) – not for CS397 module

TAEN: Timer A Enable
TBEN: Timer B Enable
- Timer Disabled : 0
- Timer Enabled : 1

(See also next slide …)

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

Control Register (GPTMCTL)

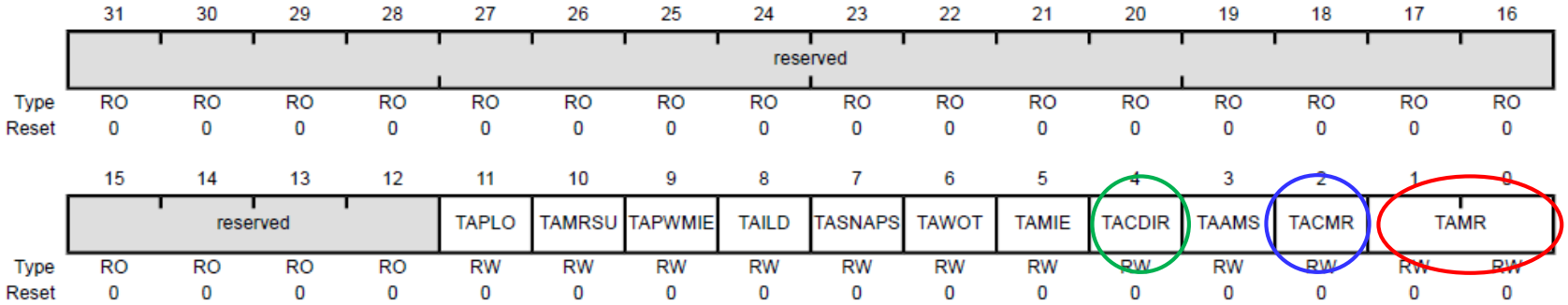
Timer A

Timer B

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

Timer A Mode Register (GPTMTAMR)

Type, Reset Value: R/W, 0x0000.0000



Register configures the GPTM based on the configuration selected in the **GPTMCFG** register

Timer Mode Selection:

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

Timer Capture Mode: *(not used in CS397)*

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

Timer Count Direction:

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

For Timer B, Mode register is **GPTMTBMR** and has the same bit definitions.

(See also next slide …)

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spmu376e.pdf, p729)

Timer A Mode Register (GPTMTAMR)

Timer A bits definition

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

Enabling a GPTM Module

RCGCTIMER, PRTIMER

Enabling a Timer

- Before a Timer can be initialized and to be able to access its registers, we need to enable the Clock signal to the module & check that its registers are ready to be accessed.
  - Use **RCGCTIMER** register to enable the clock to the Timer module.
  - Use **PRTIMER** register to check if Timer registers are ready to be accessed.
- Both the **RCGCTIMER** & **PRTIMER** registers are part of the set of System Control registers.

System Control Registers

Table 5-7. System Control Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xA04	PRTIMER	RO	0x0000.0000	16/32-Bit General-Purpose Timer Peripheral Ready	404
0xA08	PRGPIO	RO	0x0000.0000	General-Purpose Input/Output Peripheral Ready	406
0xA0C	PRDMA	RO	0x0000.0000	Micro Direct Memory Access Peripheral Ready	408
0xA14	PRHIB	RO	0x0000.0001	Hibernation Peripheral Ready	409
0x55C	SRWTIMER	RW	0x0000.0000	32/64-Bit Wide General-Purpose Timer Software Reset	335
0x600	RCGCWD	RW	0x0000.0000	Watchdog Timer Run Mode Clock Gating Control	337
0x604	RCGCTIMER	RW	0x0000.0000	16/32-Bit General-Purpose Timer Run Mode Clock Gating Control	338

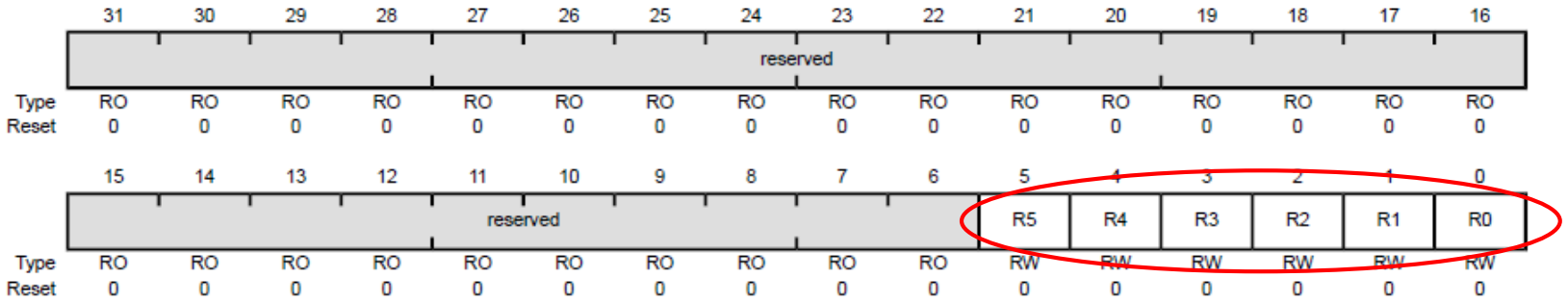
- Use **RCGCTIMER** register to enable the clock to the Timer. Clock needs to be enabled before we can use a Timer.
- Use **PRTIMER** register to check if a Timer registers are ready to be accessed.

System Control Registers have base address of 0x400F.E000

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spms376e.pdf, p338, p404)

Run Mode Clock Gating Register (RCGCTIMER)

Type, Reset Value: R/W, 0x0000.0000



When bit is enabled, Timer is provided a clock & accesses to Timer registers are allowed.

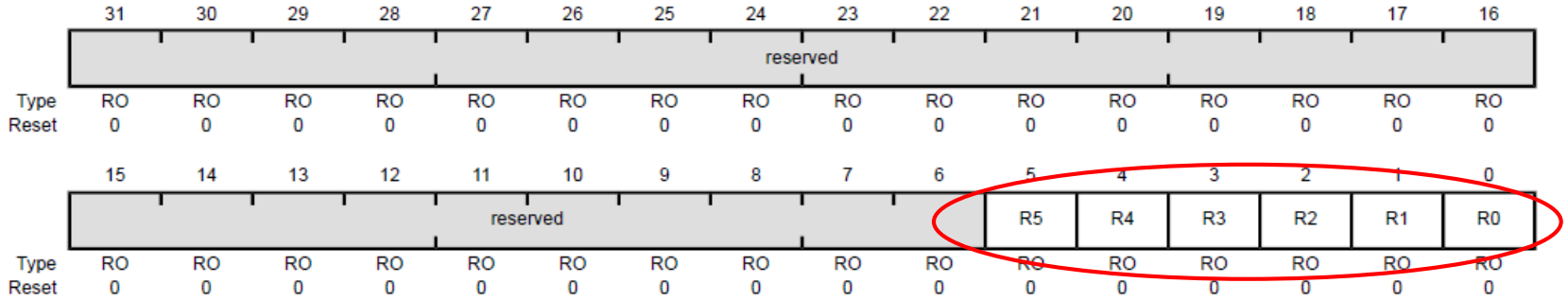
When disabled, the clock is disabled to save power & accesses to Timer registers generate a bus fault.

- **R0** = '1': Enable & provide clock to Timer 0.
- **R1** = '1': Enable & provide clock to Timer 1.
- **R2** = '1': Enable & provide clock to Timer 2.
- **R3** =' 1': Enable & provide clock to Timer 3.
- **R4** = '1': Enable & provide clock to Timer 4.
- **R5** = '1': Enable & provide clock to Timer 5.
- A '0' will disable the corresponding Timer module.

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spms376e.pdf, p338)

General-Purpose Timer Ready Register (PRTIMER)

Type, Reset Value: RO, 0x0000.0000



- Check this register bits to be sure Timer module is ready.
  - **Bit = '1' => Timer is ready to be assessed.**
  - Check upon a power reset, run mode clocking or change in power status.
- **R0** denotes Timer 0.
- **R1** denotes Timer 1.
- **R2** denotes Timer 2.
- **R3** denotes Timer 3.
- **R4** denotes Timer 4.
- **R5** denotes Timer 5.

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spms376e.pdf, p404)

Programming Examples

```
/** bit definitions in TM4C123GH6PM7.h  **/
#define SYSCTL_RCGCTIMER_R5   0x00000020  // Timer 5 Run Mode Clock Gating Ctrl
#define SYSCTL_RCGCTIMER_R4   0x00000010  // Timer 4 Run Mode Clock Gating Ctrl
#define SYSCTL_RCGCTIMER_R3   0x00000008  // Timer 3 Run Mode Clock Gating Ctrl
#define SYSCTL_RCGCTIMER_R2   0x00000004  // Timer 2 Run Mode Clock Gating Ctrl
#define SYSCTL_RCGCTIMER_R1   0x00000002  // Timer 1 Run Mode Clock Gating Ctrl
#define SYSCTL_RCGCTIMER_R0   0x00000001  // Timer 0 Run Mode Clock Gating Ctrl

#define SYSCTL_PRTIMER_R5     0x00000020  // Timer 5 Peripheral Ready
#define SYSCTL_PRTIMER_R4     0x00000010  // Timer 4 Peripheral Ready
#define SYSCTL_PRTIMER_R3     0x00000008  // Timer 3 Peripheral Ready
#define SYSCTL_PRTIMER_R2     0x00000004  // Timer 2 Peripheral Ready
#define SYSCTL_PRTIMER_R1     0x00000002  // Timer 1 Peripheral Ready
#define SYSCTL_PRTIMER_R0     0x00000001  // Timer 0 Peripheral Ready
```

```
/** enable timer 0 clock                **/
SYSCTL->RCGCTIMER |= SYSCTL_RCGCTIMER_R0; /* see pg338 of spms376e,pdf */
/** enable timer 1 & 2 clock  **/
SYSCTL->RCGCTIMER |= SYSCTL_RCGCTIMER_R1 | SYSCTL_RCGCTIMER_R2;

/** Check if Timer 0 & 1 is ready through reading PRTIMER register bits **/
while( 0 == (SYSCTL->PRTIMER & SYSCTL_PRTIMER_R0) );  // Timer 0
while( 0 == (SYSCTL->PRTIMER & SYSCTL_PRTIMER_R1) );  // Timer 1
```

Periodic Timer Mode

GPTMT*n*MR, GPTMT*n*ILR

Periodic Timer 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): **Pre-scaler** is only available when the Timers are used **INDIVIDUALLY**
– *not available when Timers are concatenated.*

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

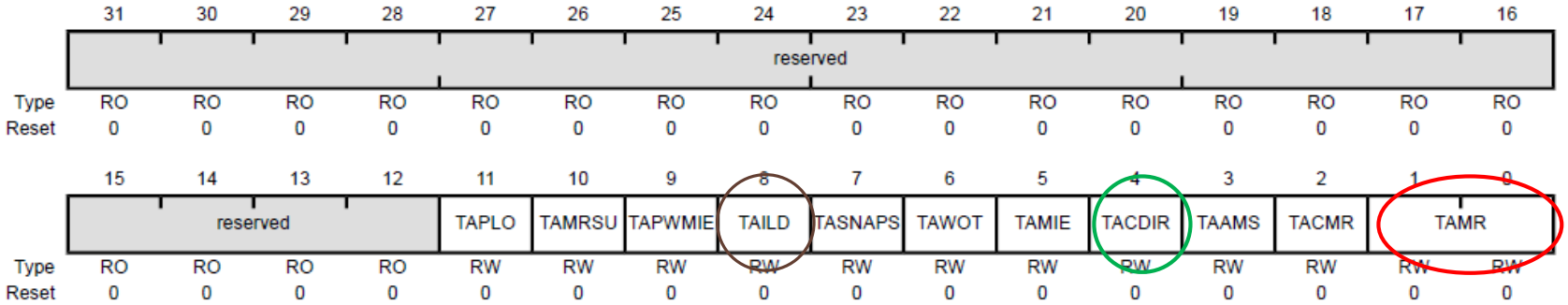
Periodic Timer Mode …

- How do we configure a GPTM Timer to be in **Periodic** mode?
  - Set the Timer Mode (**T*n*MR)** field (2 bits) of the Mode (**T*n*MR**) register; *n* = A or B for Timer A or Timer B.
  - **T*n*MR** = 0x2 for Periodic Timer mode (**TAMR** for Timer A; **TBMR** for Timer B).

- How do we configure the GPTM Timer to count **UP** or **DOWN**?
  - Use the Count Direction (**T*n*CDIR**) bit in the Mode (**T*n*MR**) register.
  - Count UP : **T*n*CDIR** = 1
  - Count DOWN : **T*n*CDIR** = 0.

Timer A Mode Register (GPTMTAMR)

Type, Reset Value: R/W, 0x0000.0000



Timer Count Direction:

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

Timer Mode Selection:

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

TAILD bit:

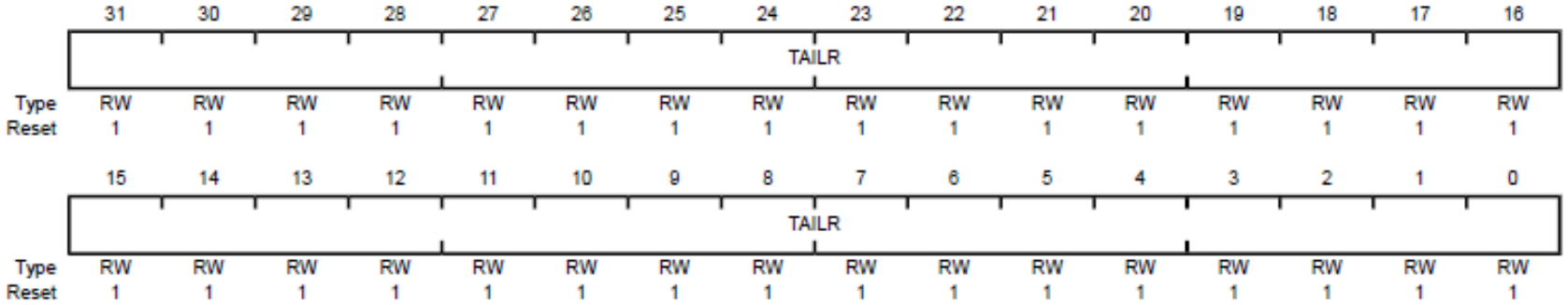
- TAILD = 0 : If program updates TAILR or TAPR while Timer is counting down, new values are loaded at the next clock cycle.
- TAILD = 1: If program updates TAILR or TAPR while Timer is counting down, new values are loaded after the next timeout.

For Timer B, Mode register is **GPTMTBMR** and has the same bit definitions.

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spmu376e.pdf, p729)

Timer A Interval Load Register (GPTMTAILR)

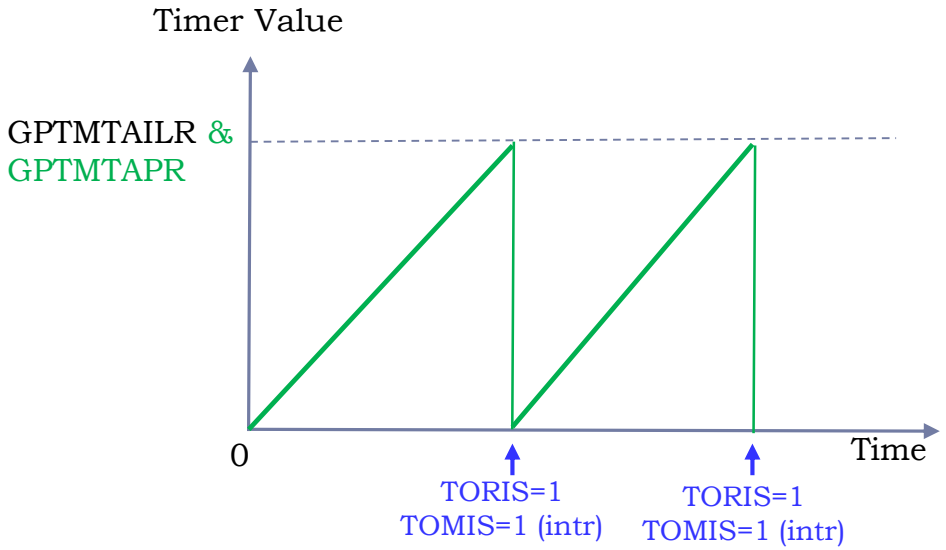
Type, Reset Value: R/W, **0xFFFF.FFFF**



- When the timer is counting **DOWN**, this register is used to load the **starting countdown value** to the Timer.
- When the Timer is counting **UP**, timer starts counting from 0. This register sets the **upper bound** for the timeout event.
- For 16/32-bit mode operation:
  - [16-bit mode]: GPTMTAILR is a 16-bit register; upper 16-bits read as 0s.
  - [32-bit mode]: GPTMTAILR is a 32-bit register; upper 16-bits = GPTMTBILR.
- **TAILR** = 32-bit load value [TBILR:TAILR], or 16-bit mode value [TAILR].

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spms376e.pdf, p756)

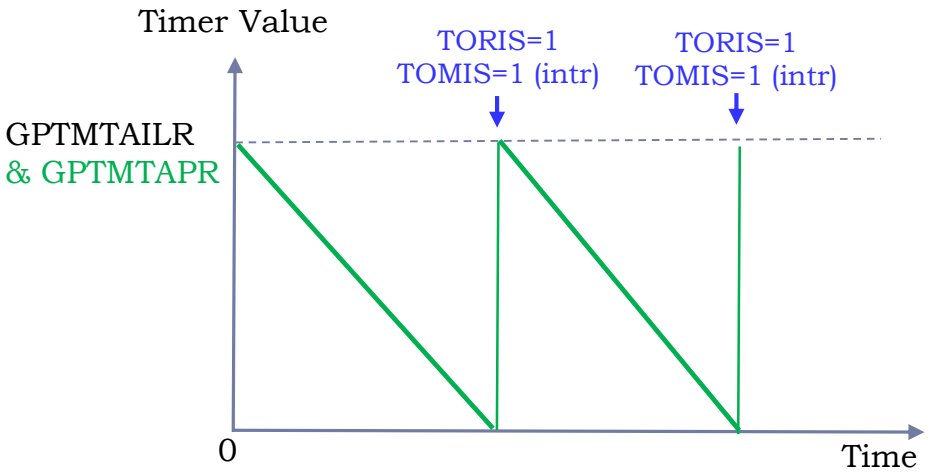
Periodic Mode – Count UP *(Timer A)*



Terminology:
GPTMTAMR: Mode register
GPTMTAILR: Interval Load register
GPTMTAPR: Prescale register
GPTMRIS: Raw Interrupt Status register
GPTMIMR: Interrupt Mask register
GPTMICR: Interrupt Clear register
TORIS: Time Out Raw Interrupt Status
TOMIS: Time Out Mask Interrupt Status

- TACDIR=1 (**GPTMTAMR** register) → Timer counts **UP**.
- Timer counter begins counting from 0 till it reach the value in **GPTMTAILR** & **GPTMAPR** (optional). → ***Time-Out event.***
- Upon Time-Out event, **TATORIS** flag in **GPTMRIS** register is set.
- A Time-Out interrupt is triggered & **TATOMIS** bit is set if interrupt is enabled in the **GPTMIMR**.
  - RIS & MIS status cleared by writing to **GPTMICR** register.
- Timer counter is then reloaded with 0 at the next clock cycle.
- Count UP cycle repeats.

Periodic Mode – Count DOWN *(Timer A)*



Terminology:
GPTMTAMR: Mode register
GPTMTAILR: Interval Load register
GPTMTAPR: Prescale register
GPTMRIS: Raw Interrupt Status register
GPTMIMR: Interrupt Mask register
GPTMICR: Interrupt Clear register
TORIS: Time Out Raw Interrupt Status
TOMIS: Time Out Mask Interrupt Status

- TACDIR=0 (**GPTMTAMR** register) → Timer counts **DOWN**.
- Timer counter begins counting from the value in **GPTMTAILR** & **GPTMTAPR** (optional).
- It counts DOWN till counter value reaches 0 → ***Time-Out event.***
- Upon Time-Out event, **TATORIS** flag in **GPTMRIS** register is set.
- If Time-Out interrupt is enabled in the **GPTMIMR**, an interrupt is triggered & **TATOMIS** bit is set.
  - RIS & MIS register statuses are cleared by writing to the **GPTMICR** register.
- At the next clock cycle, Timer counter is reloaded with value in **GPTMTAILR** register & **GPTMTAPR**.
- Count DOWN cycle starts again at the next clock cycle.

Timer Operation

Take note the following:

- Upon Reset, the **GPTMT*n*ILR** register is initialized to all '1's => value of **0xFFFF.FFFF**.
  - It means the Timer has the highest count value possible upon Reset.
  - Default direction is Count-down.
- Timer counter decrements / increments at the bus clock frequency.

Timer Interrupts

GPTMIMR, GPTMRIS, GPTMMIS, GPTMICR

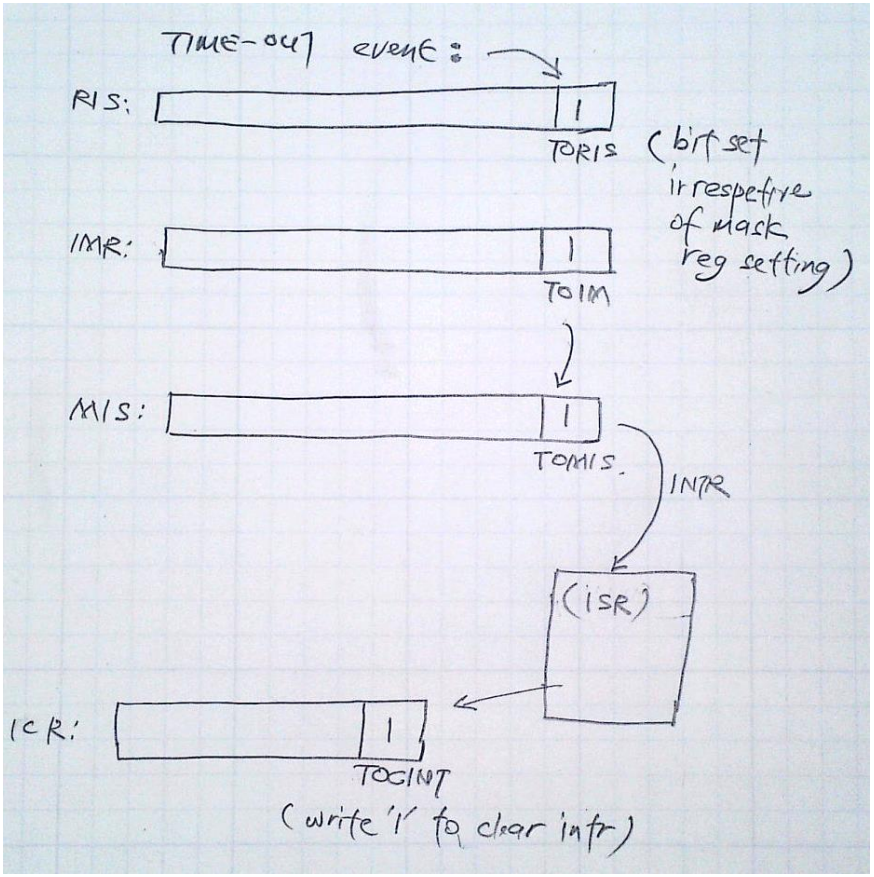
Points to Note about Interrupts

Registers involved in Interrupt Processing:

- IMR: Interrupt Mask register
- MIS: Mask Interrupt Status
- RIS : Raw Interrupt Status register
- ICR: Interrupt Clear register

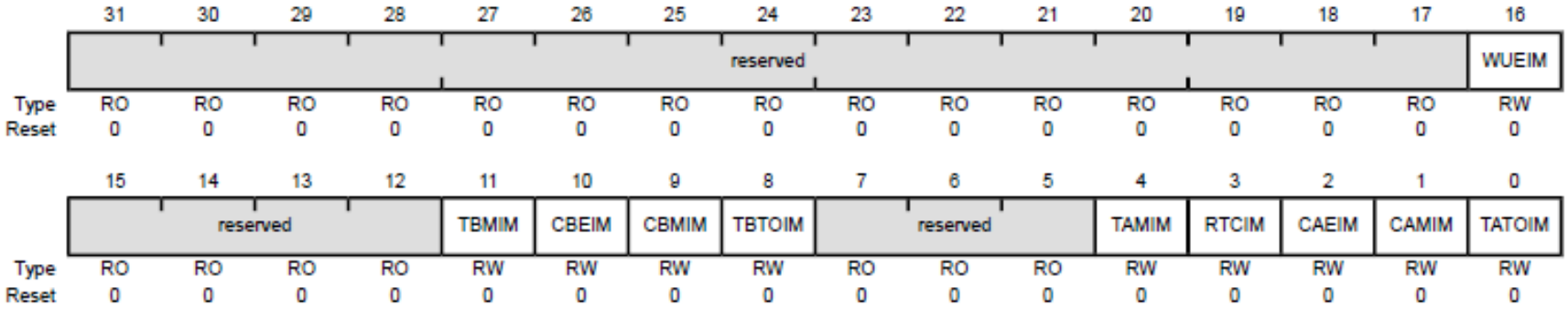
<u>Time-Out Event Interrupt Example:</u>

- If the TOIM (Time-Out Interrupt Mask) in the IMR register is set, Time-Out interrupt is enabled:
  - When a Time-Out event occurs, the TOMIS bit in the MIS register will be set & an interrupt is triggered.
- When a Time-out event occurs, the TORIS bit in the RIS will be set.
  - This is irrespective of whether the Time-out event is masked in the IMR.
- Interrupt needs to be cleared by clearing the corresponding bit in the ICR.
  - For Time-Out event, it will be the TOCINT bit in the ICR register.



Interrupt Mask Register (GPTMIMR)

Type, Reset Value: R/W, 0x0000.0000



[Timer B Mask] [Timer A Mask]

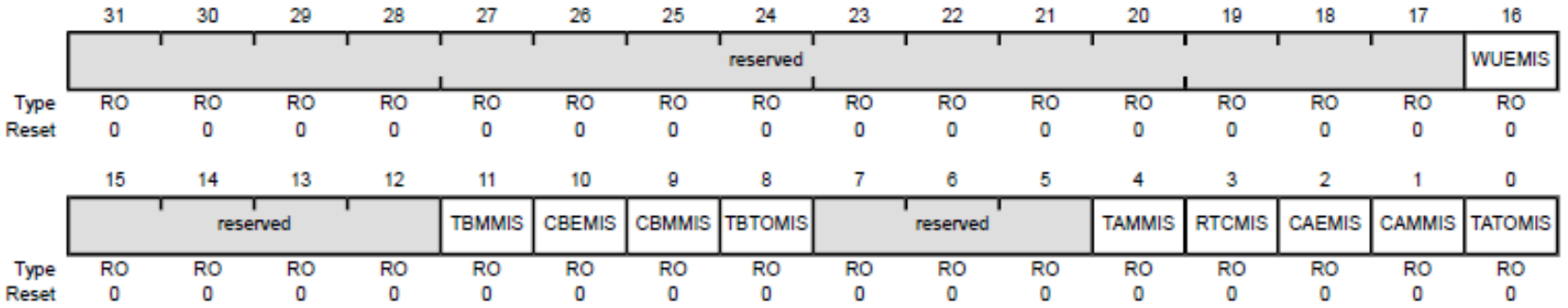
- Register allows software to enable/disable GPTM interrupts.
- Setting a bit enables the corresponding interrupt, while clearing a bit disables it.

- TATOIM, TBTOIM: Timer Time-Out Interrupt Mask
- CAMIM, CBMIM: Timer Capture Mode Match Interrupt Mask
- CAEIM, CBEIM: Timer Capture Mode Event Interrupt Mask
- TAMIM, TBMIM: Timer Match Interrupt Mask
- RTCIM: Real-Time-Clock Interrupt Mask

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spms376e.pdf, p745)

Mask Interrupt Status Register (GPTMMIS)

Type, Reset Value: R0, 0xFFFF.FFFF



[Timer B Intr Status] [Timer A Intr Status]

- Register show the timer interrupts status.
- If an interrupt is unmasked in **GPTMIMR**, & there is an event that causes the interrupt to be asserted, the corresponding bit is set.
- All bits are cleared by writing a 1 to the corresponding bit in **GPTMICR**.

- TATOMIS, TBTOMIS: Timer Time-Out Interrupt status
- CAMMIS, CBMMIS: Timer Capture Mode Match Interrupt status
- CAEMIS, CBEMIS: Timer Capture Mode Event Interrupt status
- TAMMIS, TBMMIS: Timer Match Interrupt status
- RTCMIS: Real-Time-Clock Interrupt status

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spms376e.pdf, p751)

Raw Interrupt Status Register (GPTMRIS)

Type, Reset Value: R0, 0x0000.0000

For 32/64-bit Timer

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

[Timer B Status] [Timer A Status]

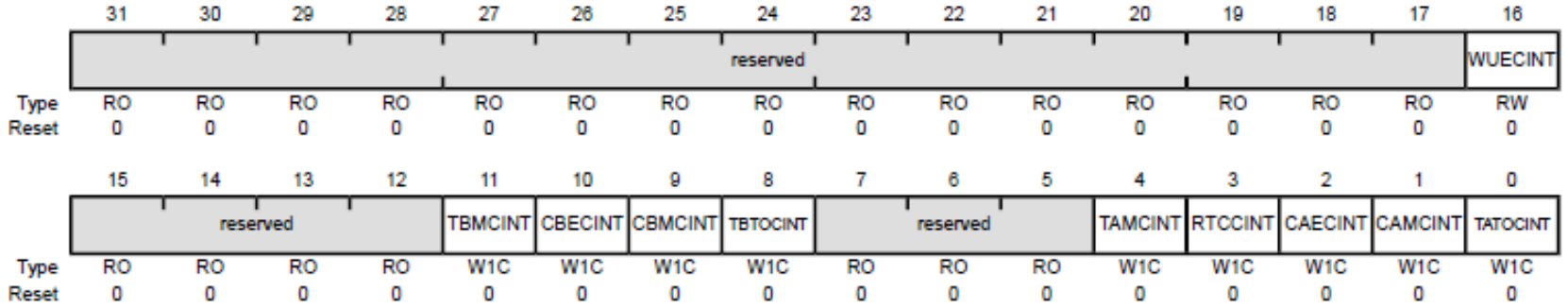
Register shows status of the GPTM's internal interrupts. Status bits are set whether or not the interrupt is masked in **GPTMIMR** register.
Each bit can be cleared by writing a 1 to its corresponding bit in **GPTMICR**.

- TATORIS, TBTORIS: Timer Time-Out RIS status
- CAMRIS, CBMRIS: Timer Capture Mode Match RIS status
- CAERIS, CBERIS: Timer Capture Mode Event RIS status
- TAMRIS, TBMRIS: Timer Match RIS status
- RTCRIS: Real-Time-Clock RIS status

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spms376e.pdf, p748)

Interrupt Clear Register (GPTMICR)

Type, Reset Value: W1C, 0x0000.0000



[Timer B] [Timer A]

- Register is used to clear the interrupt status bits in the **GPTMRIS** & **GPTMMIS** registers.
- Writing a '1' to a bit clears the corresponding bit in the **GPTMRIS** & **GPTMMIS** registers.

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spms376e.pdf, p754)

GPTM Pre-scaler

16-bit Timers (non-concatenated): T*n*PR, T*n*ILR

Periodic Timer Mode

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	
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

Pre-scaler not used when Timer is concatenated. 32/64 bits Timer

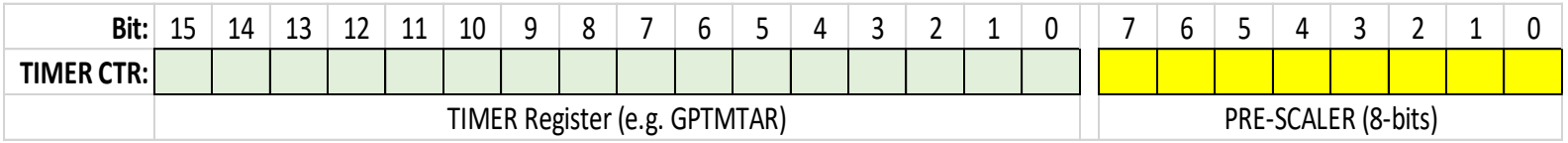
Pre-scalar behaves differently in the various Timer modes.

- **Periodic** / One-Shot Mode:
  - When counting DOWN, the **pre-scaler** acts as a true pre-scaler. It contains the least-significant bits of the count.
  - When counting UP, the **pre-scaler** acts as a timer extension & holds the most-significant bits (MSB) of the count.
- **PWM & Input Edge Count**:
  - **Pre-scaler** acts as a timer extension, regardless of the count direction.

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

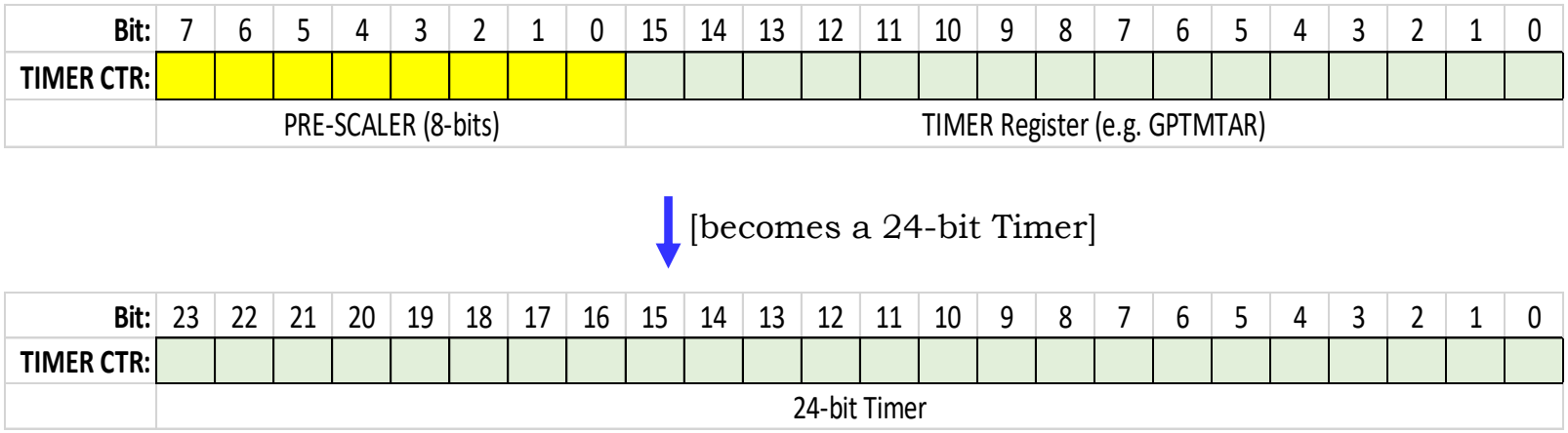
Timer Pre-scaler in Periodic Mode

- Periodic Mode, Count **DOWN**: (TRUE PRE-SCALER)
  - Pre-scaler forms the **LSB** bits; 8-bits width for 16-bit mode Timer.
  - Pre-scaler counts down from ILR value to 0 before Timer counter counts down by 1.
    - It behaves as if there are TWO Timers.
  - Thus, timer counter counts down at frequency of $\left(\frac{SystemClock}{Pre-scaler}\right)$.
    - Timer count slows down! → this is what is meant by **TRUE PRE-SCALER**.



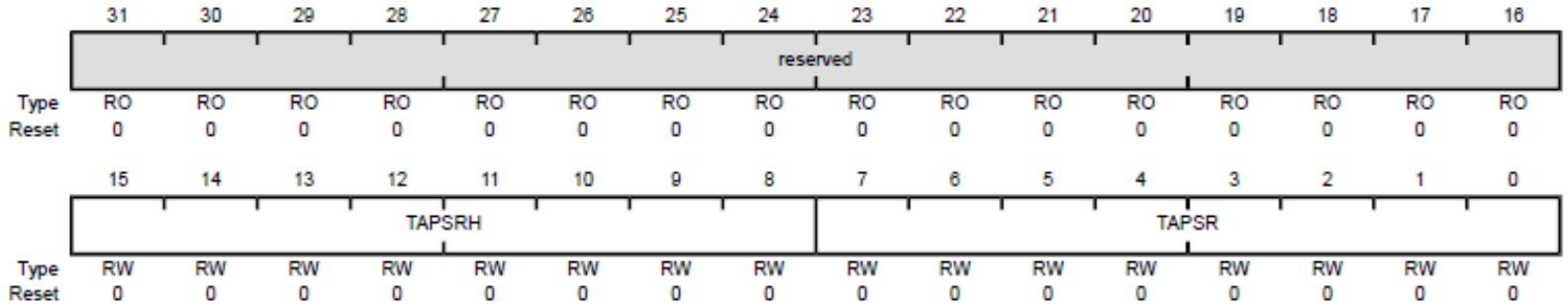
Timer Pre-scaler in Periodic Mode

- Periodic Mode, Count **UP**: (TIMER EXTENSION)
  - Pre-scaler forms the **MSB** bits.
  - Pre-scaler acts as Timer-extension to 24 bits *(16-bits + 8-bits)*.
  - Timer is effectively a 24-bit Timer.
    - It behaves a ONE concatenated 24-bit Timer.
  - Timer counter counts at frequency of (***SystemClock***).



Timer A Pre-scale Register (GPTMTAPR)

Type, Reset Value: RW, 0x0000.0000



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.

TAPSRH: GPTM Timer A Prescale High Byte (8-bits)
(not used in 16/32-bit timer modes)

For the 16/32-bit GPTM, this field is reserved. For the 32/64-bit Wide GPTM, this field contains the upper 8-bits of the 16-bit prescaler.

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spmu376e.pdf, p760)

Pre-Scalar in Timer Operation

In **Periodic Count DOWN Mode**, the Pre-scalar counts DOWN to 0 before the counter value is counted down.

Example: (16-bit Timer A)

- At 80MHz, if Pre-scalar (**TAPR**) = 0xFF, the counter value changes every (12.5×2^8) ns = counter resolution.
- 80MHz / 2^8 = 312,500 Hz = counter frequency with pre-scaler.
- At maximum Preload (**TAILR**) = 0xFFFF (16 bits), we get the largest possible count period/interval.

$$Count\ Interval = \frac{1}{312{,}500} \times 65{,}536 = 209.7152ms$$

2^{16} = 65,536
0xFFFF = 65,535

Periodic Mode:
When counting DOWN, pre-scaler acts as a true pre-scaler. It contains the LSB bits of the count.

Pre-Scalar in Timer Operation

In **Periodic Count UP Mode**, the Timer counter counts with the Pre-scalar value appended as its MSBs.

Example:

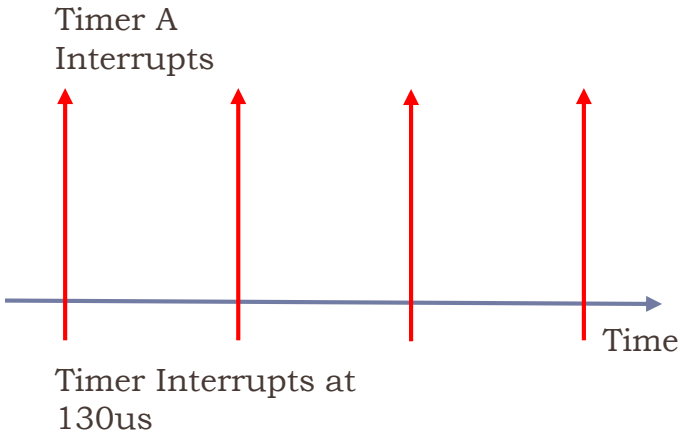
- At 80 MHz bus clock, if Pre-scalar (***TAPR***) = 0x3F and a Preload (***TAILR***) value of 0x55FF (16 bits), what is the time interval for the UP counter?
- At 80 MHz, period = 1/80 MHz = 12.5 ns
- When Timer is counting UP, the pre-scaler acts as a time extension => it becomes the MSB bits.
- Thus, counter load value = 0x3F.55FF (24 bits) = 4,150,783.

$$Interval = (4{,}150{,}783 + 1) \times 12.5ns = 51.88ms$$

Periodic Mode:
When counting UP, pre-scaler acts as Timer extension & holds the MSB bits of the count.

Ex 1: Timer Periodic Mode

- **If we want a Timer A Time-out interrupt to happen at every 130µs:**



- At 80MHz bus clock, period = 12.5ns.
- With a 16-bit count **DOWN** Timer,
  - maximum count = 2^{16}
  - maximum interval = 2^{16} x 12.5ns = 0.8192ms.
- Since 130µs < 0.8192ms, there is no need to make use of a pre-scaler.

$$\text{No of counts} = \frac{130\mu s}{12.5ns} = 10{,}400 = 0x28A0$$

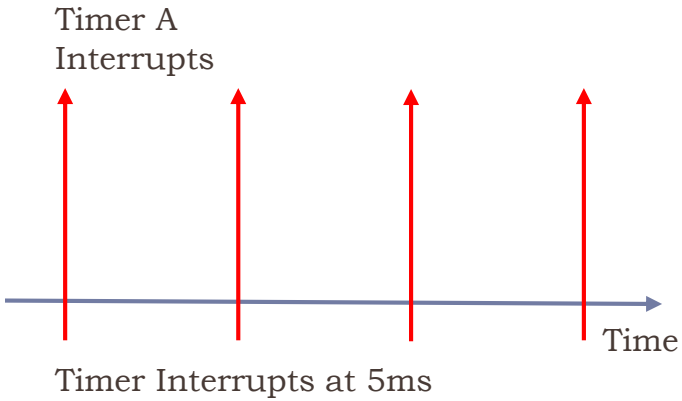
$$Reload\ value\ (\boldsymbol{TAILR}) = 0x28A0 - 1 = 0x289F$$

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

Ex 2: Timer Operation

- **If we want a Timer A Time-out interrupt to happen at every 5ms:**

- At 80MHz bus clock, period = 12.5ns.
- With a 16-bit count **DOWN** Timer,
  - maximum count = 2^{16}
  - maximum interval = 2^{16} x 12.5ns = 0.8192ms.



- Since 5ms > 0.8192ms, we need to make use of a pre-scaler.

$$\frac{5ms}{0.8192ms} = 6.10$$

We choose a pre-scaler division = 7.
Thus, ***TAPR*** = 7 – 1 = 6

$$\frac{5ms}{12.5ns \times 7} = 57{,}143 = 0xDF37$$

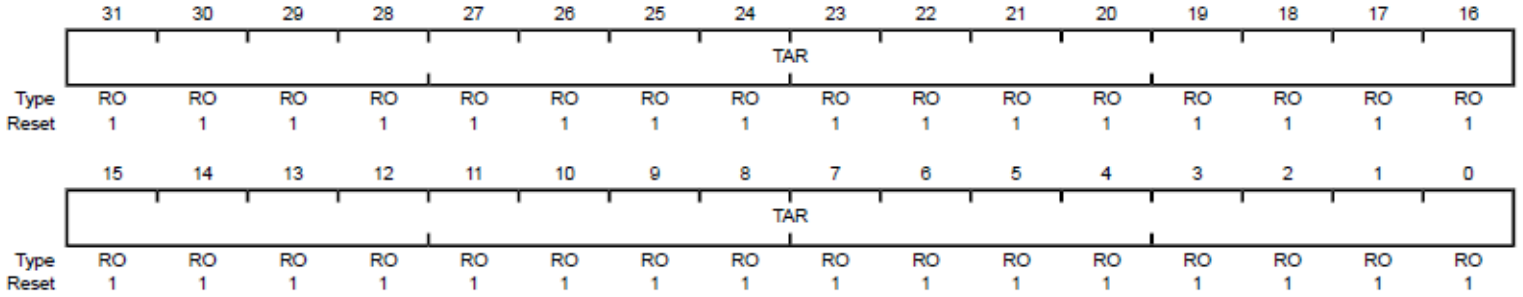
Thus, Reload Value (***TAILR***) = 0xDF37 – 1 = 0xDF36

Timer Match & Interrupt

GPTMT*n*R, GPTMTnV, GPTMT*n*MATCHR, GPTMT*n*PMR

Timer A Register (GPTMTAR)

Type, Reset Value: R0, 0xFFFF.FFFF



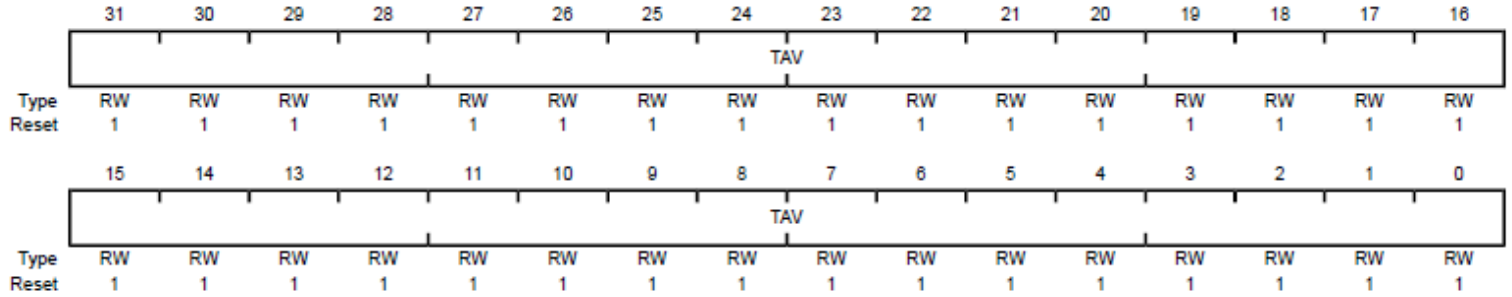
- Register shows **current value** of Timer A counter in all cases (except for Input Edge Count & Time modes).
- 16-bit Periodic & One-shot modes:
  - **Bit (15:0) = counter value**
  - Bit (31:24) = 0
  - To read Pre-scaler – read bit [23:16] of **GPTMTAV** register

- Input Edge Count: Register contains number of edges that have occurred.
- Input Edge Time: Register contains time at which the last edge event took place.
- 16-bit Input Edge Count, Input Edge Time, PWM modes:
  - Bit (15:0) = counter value.
  - Bit (23:16) = prescaler (MSBs)
  - Bit 31:24 = 0).

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spms376e.pdf, p764)

Timer A Value Register (GPTMTAV)

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



- During **READ**, register shows current, free-running value of Timer A in all modes.
- SW can use register value to determine time elapsed between an interrupt & ISR entry when using the snapshot feature in Periodic mode.
- During **WRITE**, value written to register is loaded to **GPTMTAR** register on the next clock cycle.

- 16-bit **Periodic Down** & One-shot modes:
  - **Bit 15:0 = counter value**
  - **Bit 23:16 = true pre-scaler**
  - Bit 31:24 = 0
- 16-bit **Periodic Up**, Input Edge Count & Time, PWM modes:
  - **Bit 15:0 = counter value**
  - **Bit 23:16 = pre-scaler (MSB)**
  - Bit 31:24 = 0
- 32-bit Mode:
  - Bit 31:16 = TBV register value

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spmu376e.pdf, p766)

Timer A Match Register (GPTMTAMATCHR)

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

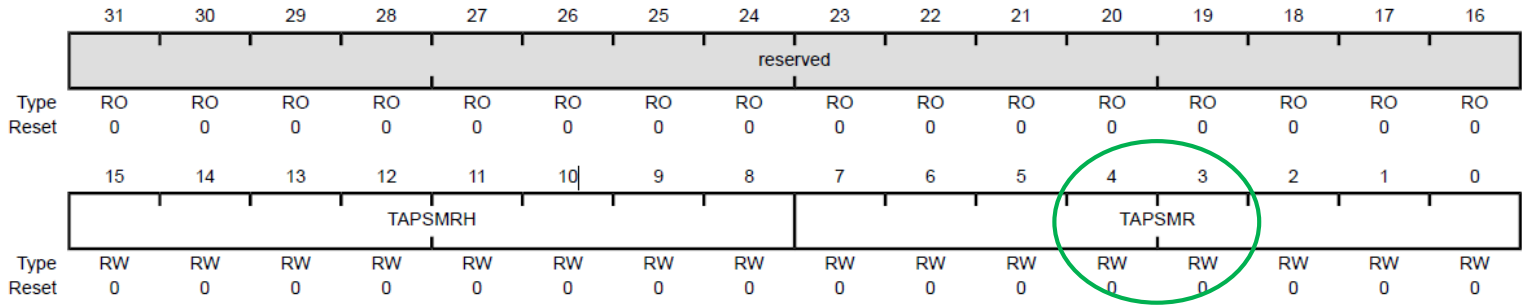
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TAMR															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	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

- **TAMR** = 32 bit Match value.
- In 16-bit mode, only bits 15:0 are used; the upper 16 bits are read as 0s.
- 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.
  - Interrupts enabled through the **TAMIE** (Match interrupt Enable) bit in Mode register.

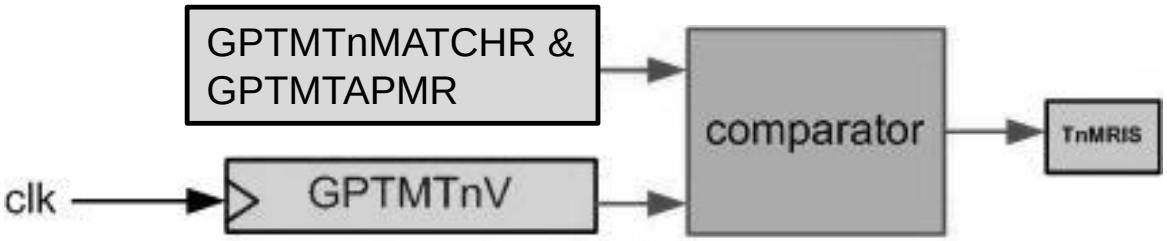
Timer A Pre-scale Match Register (GPTMTAPMR)

Type, Reset Value: R/W, 0x0000.0000



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

How a Timer Match Value Works …



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

- **Match Value** in GPTM Timer A Match (**GPTMTAMATCHR**) & Pre-scale Match (**GPTMTAPMR**) register *(optional, if used)* can be used to compare with **Current Value** stored in the GPTM Timer A (**GPTMTAV**) register to trigger an **interrupt** or set a flag to indicate that a time value matching has occurred if both values are equal.
- When the contents of free-running Timer A counter & Timer A Match & Pre-scale Match registers are equal, the **TAMRIS** Flag goes up indicating there is a match.

Source: Tiva TM4C123GH6PM Microcontroller Data Sheet (spmu376e.pdf, p758)

Periodic Timer Match Interrupt

- How do we enable Match interrupts for a Timer?
  - Set the **T*n*MIE** (Match Interrupt Enable) bit of the Mode (***T*nMR**) register.
- What scenario causes a Timer match interrupt to be generated?
  - Match interrupt is generated when Timer value equals the value loaded to the **T*n*MATCHR** (Timer Match) register & **T*n*PMR** (Prescale Match) register.
  - Interrupt status bits are updated through the **T*n*MIS** (Mask Interrupt Status) & **T*n*RIS** (Raw Interrupt Status) register.

Periodic Timer Mode Initialization

TM4C123GH6PM7 GPTM Module

Periodic Timer Mode – Timer Registers

- **Periodic Timer Mode** is selected by **TACMR** = 0x02 in **TAMR** (Mode) register.
- Count UP or DOWN, selected by **TACDIR** bit**.**
- Timer Reload value is stored in **TAILR** register.
- Pre-scalar is stored in **TAPR** register.
- Timer is enabled or disabled through **TAEN** bit in **CTL** register.
- Timer can be used:
  - Individually *(Timer A or B) – 16 bit mode,* or
  - Concatenated *(Timer (A + B)) – 32 bit mode*.

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

Periodic Timer Mode (16-bit) - Initialization

16-bit Timer Initialization: *(assume Timer A)*

1. Disabled Timer during start of initialization (**TAEN** = 0/**CTL** reg).
2. Configure Timer to 16-bit mode (**CFG**=0x04/**CFG** reg)
3. Set to **Periodic mode** (**TAMR**=0x02/**TAMR** reg)
4. Set Count Up/Down direction & reload (**DIR** & **ILD**/**TAMR** reg)
5. Set initial count (**TAILR** reg).
6. Set Pre-scalar (**TAPR** reg).
7. Enable Time-out Interrupt (**TATOIM** = 1/**IMR** reg).
8. Clear Time-out Flag (write to **TATOCINT** bit/**ICR** reg)
9. Enable Timer (**TAEN**=1/**CTL** reg).

Ex: Periodic Timer Mode (16-bit) Initialization (Timer 0)

```
#define PB_T0CCP0 6U        // PB6

/** setup GPIO pins in alternate mode        **/
// other GPIO initialization codes
GPIOB->AFSEL |= BIT(PB_T0CCP0);              /* enable alternate function*/
GPIOB->PCTL &= ~GPIO_PCTL_PB6_M;           /* reset mask for PB6       */
GPIOB->PCTL |= GPIO_PCTL_PB6_T0CCP0;       /* enable timer 0 in PB6    */

/** enable timer clock                        **/
SYSCTL->RCGCTIMER |= SYSCTL_RCGCTIMER_R0; /* enable timer 0 clock     */
while(0 == (SYSCTL->PRTIMER & SYSCTL_PRTIMER_R0)){};  // Timer 0

/** setup Timer 0A in 16-bits operation        **/
TIMER0->CTL &= ~TIMER_CTL_TAEN;     /* 1. disable timer 0 during setup */
TIMER0->CFG |= TIMER_CFG_16_BIT;    /* 2. set to 16-bit mode            */
TIMER0->TAMR |= TIMER_TAMR_TAMR_PERIOD ; /* 3. periodic mode            */
TIMER0->TAMR &= ~TIMER_TAMR_TACDIR; /* 4. count down                    */
TIMER0->TAMR |= TIMER_TAMR_TAILD;   /*    new TAILR/PR values loaded    */
                                    /*    after time-out                */
TIMER0->TAILR = 0xF424;             /* 5. reload value; max = 0xFFFF    */
TIMER0->TAPR = 0xFF;                /* 6. prescaler (8 bits)            */
TIMER0->IMR |= TIMER_IMR_TATOIM;    /* 7. enable timeout intr           */
TIMER0->ICR |= TIMER_ICR_TATOCINT;  /* 8. clear timeout flag            */
TIMER0->CTL |= TIMER_CTL_TAEN;      /* 9. enable timer 0                */
```

Periodic Timer Mode (32-bit) - Initialization

32-bit Timer Initialization: *(assume Timer A)*

1. Disabled Timer during start of initialization (**TAEN** = 0 /**CTL** reg).
2. Configure Timer to 32-bit mode (**CFG**=0x00/**CFG** reg)
3. Set to Periodic mode (**TAMR**=0x02/**TAMR** reg)
4. Set Count Up/Down direction (**DIR** bit/**TAMR** reg)
5. Set initial count (**TAILR** reg).
6. Set Pre-scalar to 0 (**TAPR** reg).
7. Enable Time-out Interrupt (**TATOIM** = 1/**IMR** reg).
8. Clear Time-out Flag (write to **TATOCINT** bit/**ICR** reg)
9. Enable Timer (**TAEN**=1/**CTL** reg).

Note: In 32-bit mode, Timer A and Timer B are concatenated. In this mode, the Pre-scalar is NOT used.

Timer Interrupt Handlers (16/32-bit)

- Interrupt handlers names are pre-defined in file '*startup_TM4C123.s*'.
- Examples:
  - **`TIMER0A_Handler()`** for Timer 0A interrupt handler.
  - **`TIMER1A_Handler()`** for Timer 1A interrupt handler.
  - **`TIMER2A_Handler()`** for Timer 2A interrupt handler.
  - **`TIMER3A_Handler()`** for Timer 3A interrupt handler.

```
DCD     TIMER0A_Handler         ;  19: Timer 0 subtimer A
DCD     TIMER0B_Handler         ;  20: Timer 0 subtimer B
DCD     TIMER1A_Handler         ;  21: Timer 1 subtimer A
DCD     TIMER1B_Handler         ;  22: Timer 1 subtimer B
DCD     TIMER2A_Handler         ;  23: Timer 2 subtimer A
DCD     TIMER2B_Handler         ;  24: Timer 2 subtimer B
DCD     COMP0_Handler           ;  25: Analog Comparator 0
   . . . . . .
   . . . . . .
DCD     GPIOG_Handler           ;  31: GPIO Port G
DCD     GPIOH_Handler           ;  32: GPIO Port H
DCD     UART2_Handler           ;  33: UART2 Rx and Tx
DCD     SSI1_Handler            ;  34: SSI1 Rx and Tx
DCD     TIMER3A_Handler         ;  35: Timer 3 subtimer A
DCD     TIMER3B_Handler         ;  36: Timer 3 subtimer B
```

Timer Interrupt Handlers (32/64-bit)

- Interrupt handlers names are similary defined for the wide Timers.

```
DCD     WTIMER0A_Handler   ;  94: Wide Timer 0 subtimer A
DCD     WTIMER0B_Handler   ;  95: Wide Timer 0 subtimer B
DCD     WTIMER1A_Handler   ;  96: Wide Timer 1 subtimer A
DCD     WTIMER1B_Handler   ;  97: Wide Timer 1 subtimer B
DCD     WTIMER2A_Handler   ;  98: Wide Timer 2 subtimer A
DCD     WTIMER2B_Handler   ;  99: Wide Timer 2 subtimer B
DCD     WTIMER3A_Handler   ; 100: Wide Timer 3 subtimer A
DCD     WTIMER3B_Handler   ; 101: Wide Timer 3 subtimer B
DCD     WTIMER4A_Handler   ; 102: Wide Timer 4 subtimer A
DCD     WTIMER4B_Handler   ; 103: Wide Timer 4 subtimer B
DCD     WTIMER5A_Handler   ; 104: Wide Timer 5 subtimer A
DCD     WTIMER5B_Handler   ; 105: Wide Timer 5 subtimer B
```

Timer Interrupt Handler

Example of Timer 0A ISR:

```
/** ISR in IRQ.c    **/

extern void TIMER0A_IRQHandler( uint32_t Status );

/** Enable interrupt for Timer 0 (in IRQ.c)   **/
void IRQ_Init( void )
{
    NVIC_EnableIRQ( TIMER0A_IRQn );
}

void  TIMER0A_Handler( void )
{
    uint32_t status = TIMER0->RIS;  // or MIS register
    TIMER0A_IRQHandler (status );
}
```

Timer Interrupt Handler

```
/** Interrupt processing function in main.c   **/
/** Function is called from ISR in IRQ.c              **/

extern void TIMER0A_IRQHandler( uint32_t Status )
{
   /* if TRUE, timeout intr has occurred */
   if( 0 != (Status & TIMER_RIS_TATORIS) )
   {
      TIMER0->ICR |= TIMER_ICR_TATOCINT; /* clear intr */
      /** interrupt processing codes  **/
   }
}
```

Defined Interrupt Sources in RIS Register

```
/** Bit fields in the TIMER_O_RIS register.    **/

#define TIMER_RIS_WUERIS   0x00010000  // GPTM Write Update Error Raw
                                       // Interrupt
#define TIMER_RIS_TAMRIS   0x00000010  // GPTM Timer A Match Raw Interrupt
#define TIMER_RIS_RTCRIS   0x00000008  // GPTM RTC Raw Interrupt
#define TIMER_RIS_CAERIS   0x00000004  // GPTM Timer A Capture Mode Event
                                       // Raw Interrupt
#define TIMER_RIS_CAMRIS   0x00000002  // GPTM Timer A Capture Mode Match
                                       // Raw Interrupt
#define TIMER_RIS_TATORIS  0x00000001  // GPTM Timer A Time-Out Raw
                                       // Interrupt
```

(Defined in file 'TM4C123GH6PM7.h'.)

Review Questions

1. How many 16/32-bit Timers are there in the Tiva TM4C123G microcontroller?
2. Which bit and register should be set to program a Timer to count DOWN?
3. Timer 0 has TWO outputs: T0CCP0 and T0CCP1. Why are there two outputs?
4. Which register and bit(s) do we use to enable / disable a Timer?
5. Why do we disable a Timer during initialization?
6. Can a Pre-scalar be used in 32-bit mode for a 16/32-bit GPTM module?
7. Which register is used to enable the clock to a GPTM timer module?
8. Which register and bit(s) do we use to check that the Timer 5 is ready to be accessed?

Review Questions

Periodic Mode

9. Which register and bit(s) is used to set a GPTM to be in Periodic mode?
10. Which register and bit(s) sets the GPTM to count DOWN?
11. For a 16-bit count-DOWN Timer clocked at a bus speed of 40 MHz with a 8-bit pre-scalar, what is the maximum interval between timer interrupts? Show your working.
12. The GPTM Timer modules run at system clock frequency. [TRUE / FALSE?]
13. Describe how Timer 0A operates as a DOWN counter in Periodic mode.
14. Describe how a Timer Match interrupt can be generated.
15. Write the C codes to enable T4CCP0 pin on Timer 4. Consider the GPIO alternate function and control register settings.
16. Write the C code to enable the clock to Timer 4.
17. What is the pre-defined name for the interrupt handler for Timer 5?

Review Questions

Prescaler:

18. At 25MHz bus clock, if Pre-scalar (**TAPR**) = 0x7F and a Preload (**TAILR**) value of 0x33AA (16 bits), what is the time interval that the counter increments?
19. A GPTM is programed to operate in 16-bit, Periodic, count DOWN mode. It is required that a time-out interrupt occur at every 7ms. What are the values we should program to the **TAPR** and **TAILR** registers? Assume 80 MHz bus clock. Show your working.

Please email me at foohoong.fong@digipen.edu if you find any typos or errors in the lecture notes.