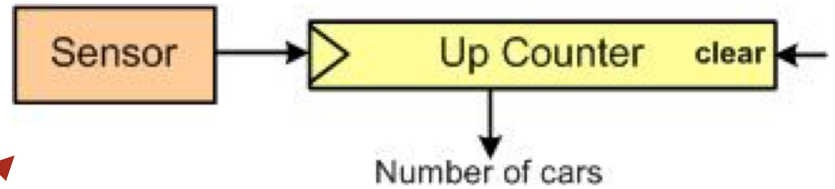
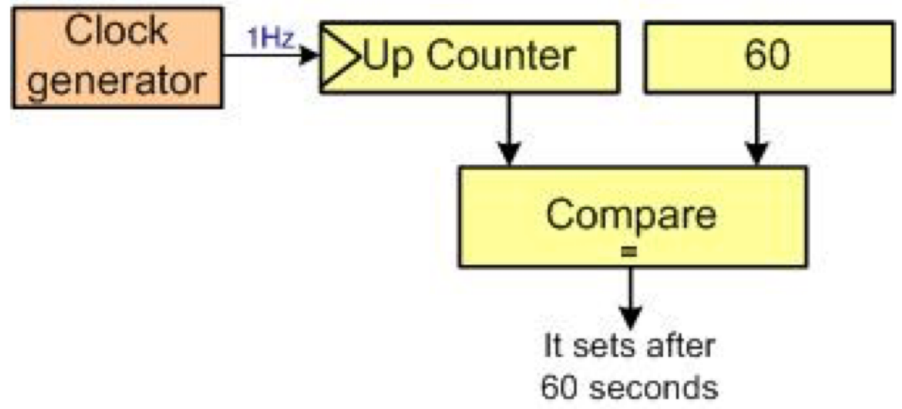
Timer Programming

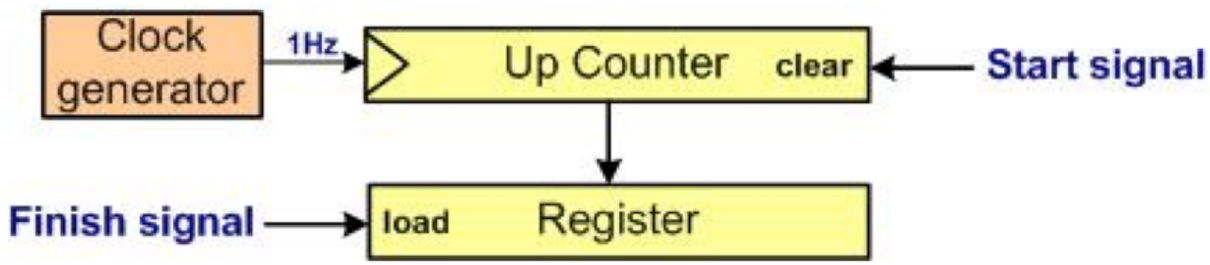
Lesson 05

카운터의 용도

1. 이벤트의 발생 횟수를 카운트
2. 지정된 시간의 지연(delay) 만들기
3. 두 이벤트 간의 시간 차이 측정하기





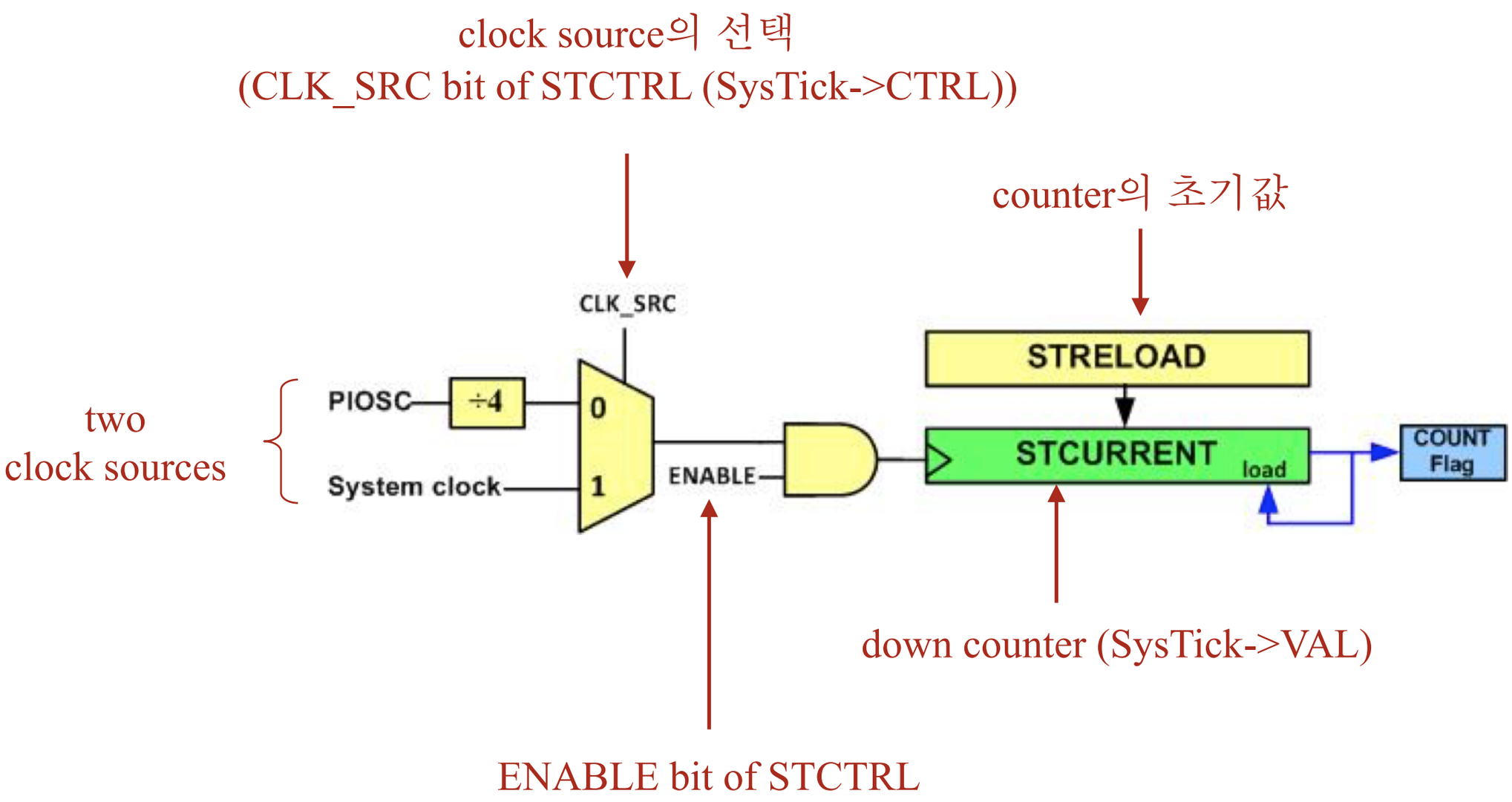


SysTick Timer

System Tick Timer

- **SysTick은 24-bit down counter이다. 시스템 클록 혹은 내부 oscillator에 의해서 driven된다.**
- **초기값 (initial value)에서 0으로 count down된다.**
- **0에 도달하면, 그 다음 클록에 underflow상태가 되고, COUNT flag가 set된다. 그러면 다시 초기값이 load되고 새로 시작한다.**
- **초기값은 0x000000에서 0xFFFFFF 사이의 임의의 값으로 설정할 수 있다.**

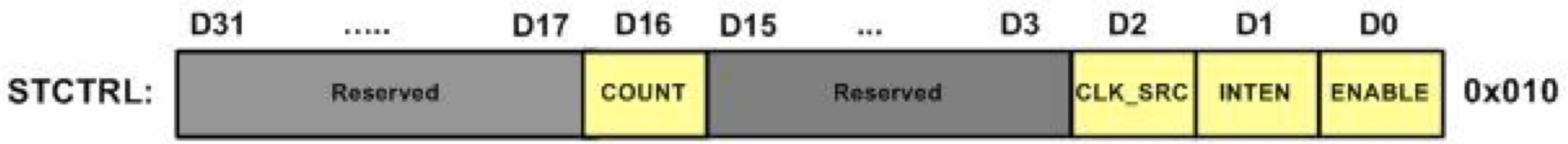
System Tick Timer



SysTick Registers

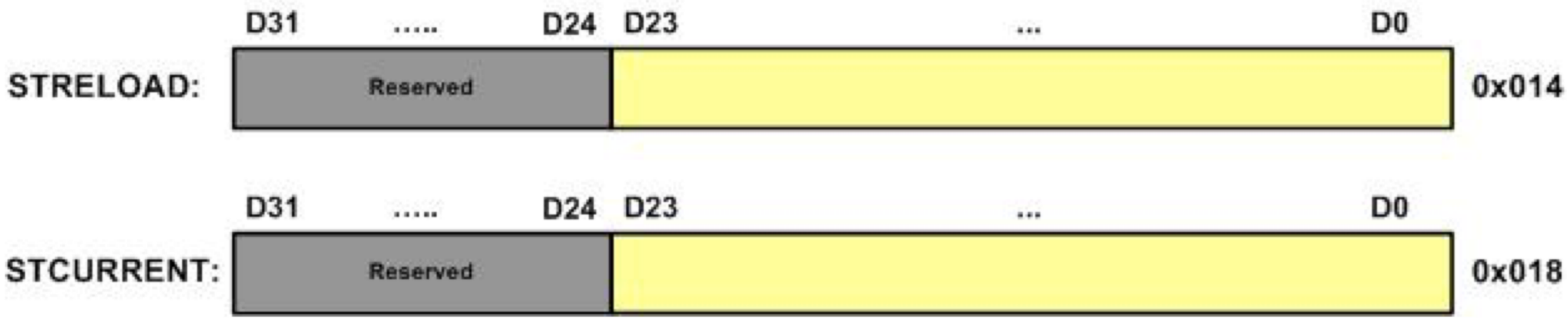
- **3 registers**
  - SysTick Control and Status Register (STCTRL)
  - SysTick Reload Value Register (STRELOAD)
  - SysTick Current Value Register (STCURRENT)

SysTick Control and Status Register (STCTRL)



bit	Name	Description
0	ENABLE	Enable (0: the counter is disabled, 1: enables SysTick to begin counting down)
1	INTEN	Interrupt Enable 0: Interrupt generation is disabled, 1: when SysTick counts to 0 an interrupt is generated
2	CLK_SRC	Clock Source 0: Precision internal oscillator (PIOSC) divided by 4 1: System clock
16	COUNT	Count Flag 0: the SysTick has not counted down to zero since the last time this bit was read 1: the SysTick has counted down to zero *Note: this flag is cleared by reading the STRCTL or writing to CTCURRENT register.*

STRELOAD and STCURRENT



Monitoring STCURRENT on LEDs

```
/* This program let the SysTick counter run freely and dumps the counter values to the tri-
color LEDs continuously. The counter value is shifted 20 places to the right so that the
changes of LEDs will be slow enough to be visible.
*/

#include “TM4C123GH6PM.h”

int main (void)
{
    int x;
    /* enable clock to GPIOF at clock gating control register */
    SYSCTL->RCGCGPIO |= 0x20;
    /* enable the GPIO pins for the LED (PF3, 2 1) as output */
    GPIOF->DIR = 0x0E;
    /* enable the GPIO pins for digital function */
    GPIOF->DEN = 0x0E;

    /* Configure SysTick */
    SysTick->LOAD = 0xFFFFFF;   /* reload reg. with max value */
    SysTick->CTRL = 5;          /* enable it, no interrupt, use system clock */

    while (1)
    {
        x = SysTick->VAL;   /* read current value of down counter */
        x = x >> 20;        /* shift right to slow down the rate */
        GPIOF->DATA = x;    /* dump it to the LEDs */
    }
}
```

Toggle PF1 LED using SysTick Counter

```
/* Toggle PF1 LED every 1 second using the SysTick Counter */
/* The system clock is running at 16 MHz. 1sec / 62.5ns = 16,000,000 for RELOAD
   register. The program then polls the flag and toggles the red LED PORTF1
   every time COUNT flag is set.
   The COUNT flag is cleared when the STCTRL register is read. */

#include "TM4C123GH6PM.h"

int main (void)
{
    /* enable clock to GPIOF at clock gating control register */
    SYSCTL->RCGCGPIO |= 0x20;
    /* enable the GPIO pins for the LED (PF3, 2 1) as output */
    GPIOF->DIR = 0x0E;
    /* enable the GPIO pins for digital function */
    GPIOF->DEN = 0x0E;

    /* Configure SysTick */
    SysTick->LOAD = 16000000 - 1; /* reload with number of clocks per second */
    SysTick->CTRL = 5;     /* enable it, no interrupt, use system clock */

    while (1)
    {
        if (SysTick->CTRL & 0x10000) /* if COUNT (D16 of CTRL reg.) flag is set */
            GPIOF->DATA ^= 2;    /* toggle PORTF1 red LED */
    }
}
```

Making Delays using SysTick

```
/* This program sets up the SysTick to set the COUNT flag at 2 Hz.
   The system clock is running at 16 MHz.
   0.5sec / 62.5ns = 8,000,000 for RELOAD register since 1 / 0.5sec = 2Hz
   The program then polls the flag and counts up a software counter every time COUNT flag is set.
   The  counter value is written to the tri-color LEDs.
   The COUNT flag (D16 of CTRL reg.)is cleared when the STCTRL register is read. */


#include “TM4C123GH6PM.h”

int main (void)
{
    int myCount  = 0;

    /* enable clock to GPIOF at clock gating control register */
    SYSCTL->RCGCGPIO |= 0x20;
    /* enable the GPIO pins for the LED (PF3, PF2, and PF1) as output */
    GPIOF->DIR = 0x0E;
    /* enable the GPIO pins for digital function */
    GPIOF->DEN = 0x0E;

    /* Configure SysTick */
    SysTick->LOAD = 8000000-1;   /* reload with number of clocks per half second */
    SysTick->CTRL = 5;    /* enable it, no interrupt, use system clock */

    while (1)
    {
        if (SysTick->CTRL & 0x10000)   /* if COUNT flag (D16 of CTRL reg.) is set */
        {
            myCount++;
            GPIOF->DATA = myCount;    /* write the count to LEDs */
        }
    }
}
```

Generating Delays using TI Timers

Timer Block

- **General Purpose Timer Module (GPTM)이라고 불리는 12개의 타이머 블록(timer block)들을 제공**
  - 6개는 16/32-bit 타이머 블록, 나머지 6개는 32/64-bit 타이머 블록
  - 하나의 16/32 비트 타이머 블록은 1개의 32비트 타이머 혹은 2개의 16비트 타이머로 사용 가능
  - 각각의 타이머 블록은 2개의 타이머로 구성됨: Timer A와 Timer B.

Timer Block

- **6개의 16/32-bit 타이머 블록은 Timer0, Timer1,…,Timer5로 지칭됨**
  - 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
- **6개의 32/64-bit 타이머 블록은 WTimer0, WTimer1,…,WTimer5로 지칭됨**

Enabling Clock

각 비트가 Timer0 ~ Timer5에 해당

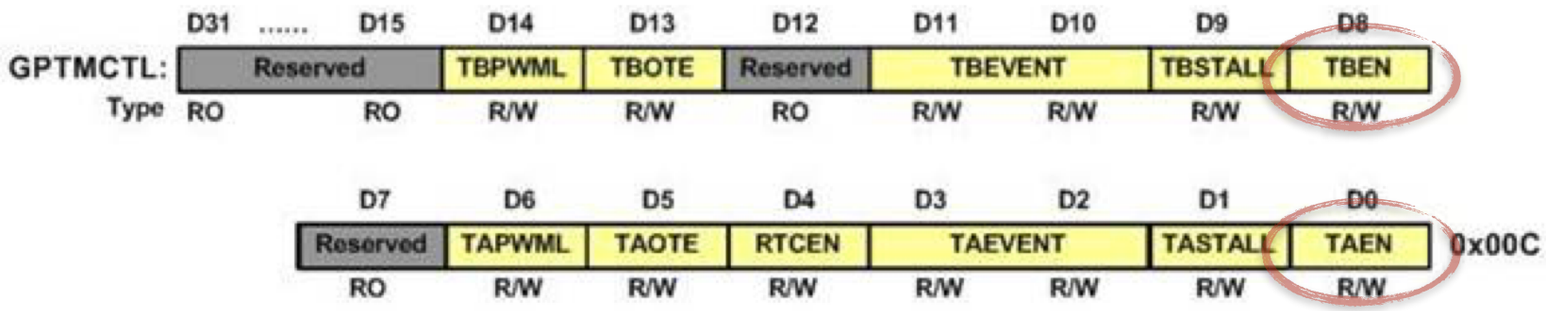
	D31	.....	D6	D5	D4	D3	D2	D1	D0	
RCGCTIMER:	Reserved			R5	R4	R3	R2	R1	R0	0x604
Type	RO		RO	R/W	R/W	R/W	R/W	R/W	R/W	

bit	Name	Description
0	R0	Timer 0 clock control (0: clock disabled, 1: clock enabled)
1	R1	Timer 1 clock control (0: clock disabled, 1: clock enabled)
2	R2	Timer 2 clock control (0: clock disabled, 1: clock enabled)
3	R3	Timer 3 clock enable (0: clock disabled, 1: clock enabled)
4	R4	Timer 4 clock enable (0: clock disabled, 1: clock enabled)
5	R5	Timer 5 clock enable (0: clock disabled, 1: clock enabled)

```
SYSCTL->RCGCTIMER |= 1;      /* enable clock to Timer Block 0 */
```

GPTM Control Register

- **GPTMCTL의 D0, D8비트를 이용하여 Timer A, B를 enable/disable함**
  - Timer의 설정을 초기화하는 동안 Timer를 disable해야 함.



bit	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. While tracing the code, when you pause on a line the timer stops counting if the bit is set. This facilitates us to see exactly what really happens.
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	Timer A Wait-on-Trigger 0: It begins counting when the timer is enabled, 1: It waits for the input to be triggered.

Note: Timer B is configured using bits 8 to 14 in the same way.

```
TIMER0->CTL = 0; /* disable Timer before initialization */
```

16 혹은 32-bit 설정

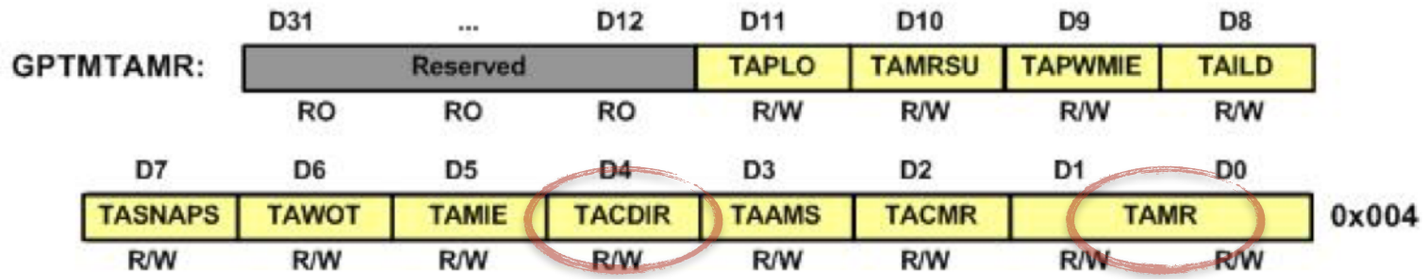
- **GPTM Configuration Register의 D2, D1, D0를 이용해서 설정**

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

```
TIMER0->CFG = 0x04;          /* 16-bit option */
```

TimerA Mode Selection

- **GPTMTAMR 레지스터를 이용해 Timer A의 모드를 선택**



모드 선택 →

Up or Down 선택 →

bit	Name	Description
0 & 1	TAMR	Timer A Mode (See Table 5-1)
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

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 5-1: TAMR Bits of GPTMTAMR

```
/* one-shot mode and down-counter */
TIMER0->TAMR = 0x01;
```

GPTM Timer n Interval Load (GPTMTnILR): Periodic Mode

GPTM Timer n Interval Load Register

GPTMRIS 레지스터의 TnTORIS flag가 set된다.

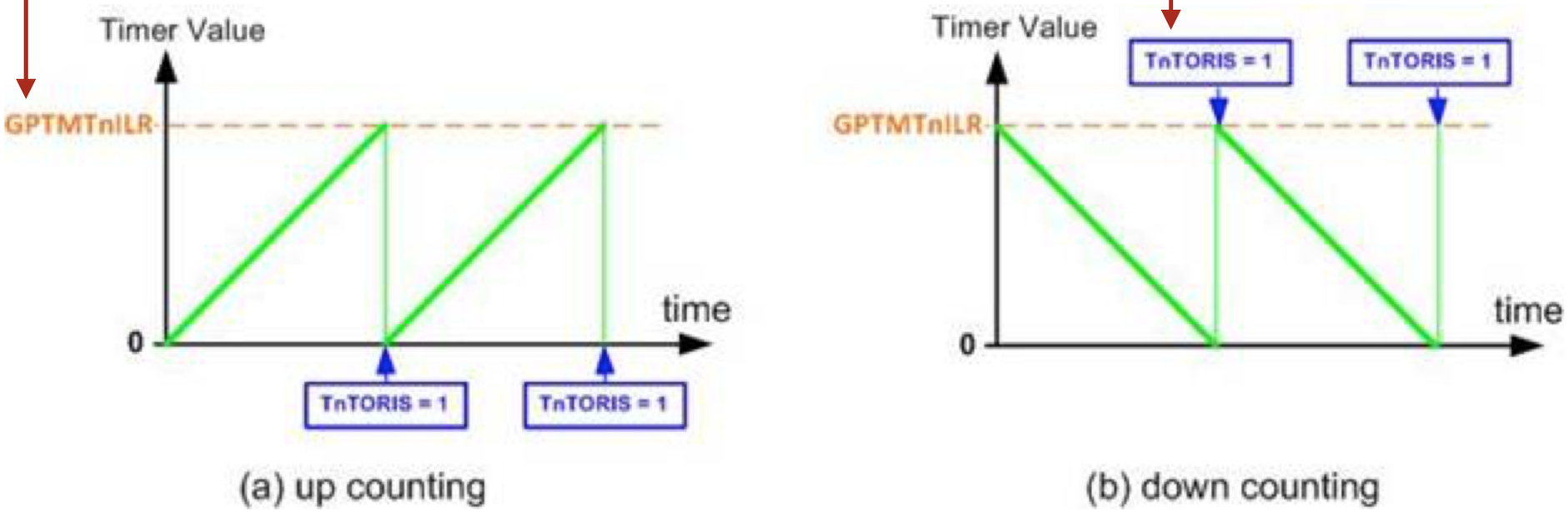


Figure 5-18: The role of GPTMTnR

```
TIMER0->TAILR = 16000*ttime-1;  /* Timer A interval load value register*/
```

One Shot Mode

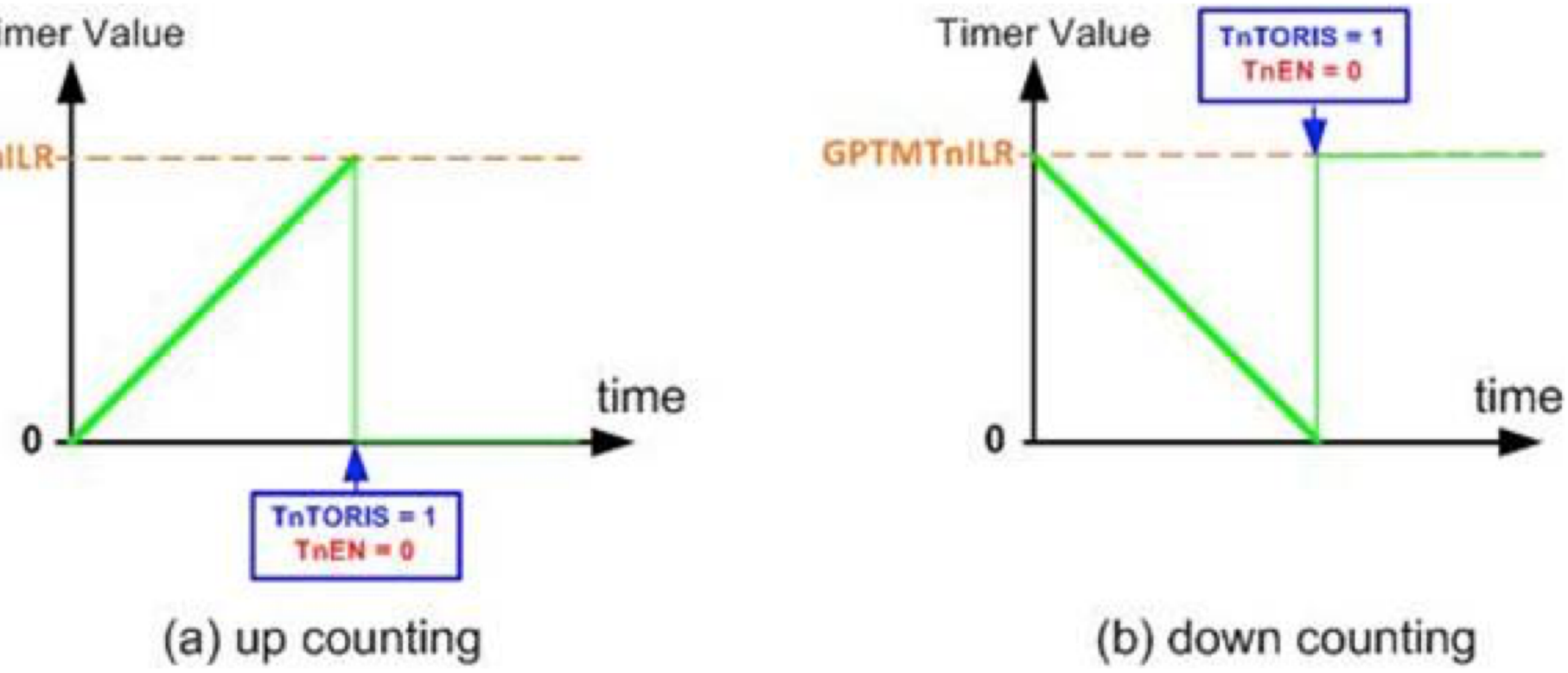
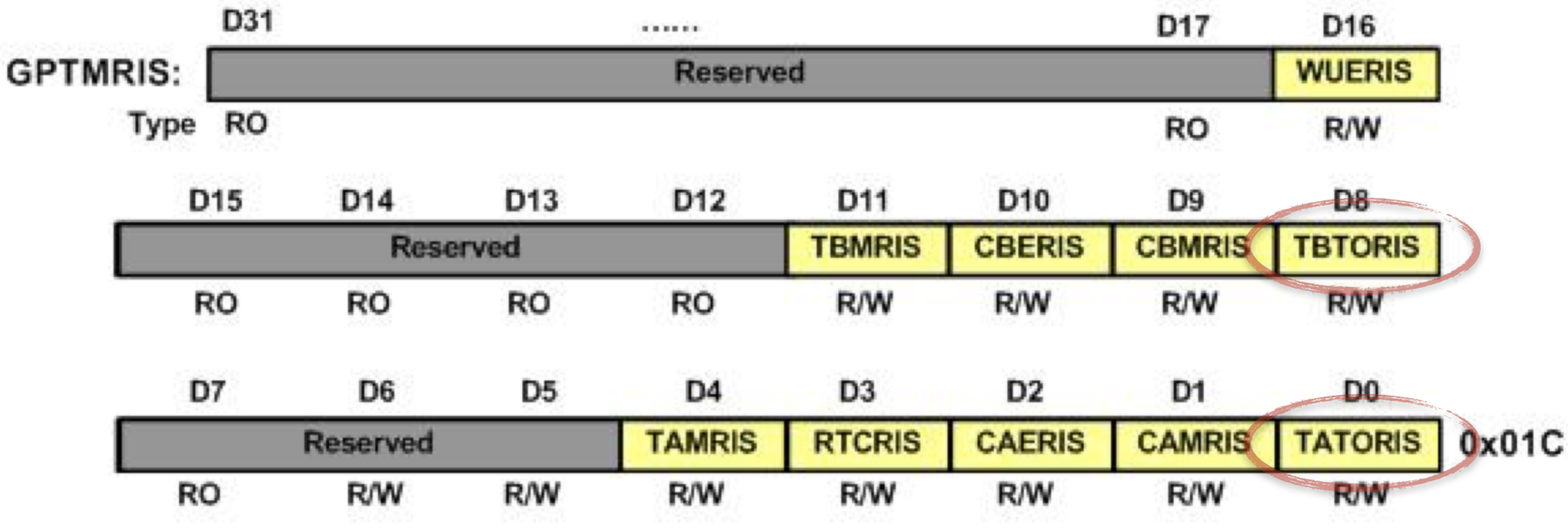


Figure 5-19: Counting in One Shot Mode

Raw Interrupt Status Register: GPTMRIS



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

Interrupt Clear Register: GPTMICR

- **GPTMRIS 레지스터의 상태 비트(status bits)를 clear하는 목적으로 사용됨**
- **1을 write하면 GPTMRIS의 대응하는 비트가 clear됨**

```
TIMER0->ICR = 0x1;              /* clear the TimerA timeout flag*/
```

Toggling PF1 LED using One-shot Mode of Timer A of Timer 0

1. enable the clock to Timer0 block
2. disable timer while the configuration is being modified
3. select 16-bit mode
4. select one-shot mode and down-counter mode
5. set interval load register value
6. clear timeout flag
7. enable counter
8. wait for timeout flag to set

Toggling PF1 LED using One-shot Mode of Timer A of Timer 0

```
/* Demonstrates the use of TimerA of Timer0 block to do a delay of the multiple of
milliseconds. Because 16-bit mode is used, it will only work up to 4 ms. */

#include “TM4C123GH6PM.h”

void timer0A_delayMs(int ttime);
void delayMs(int n);

int main (void)
{
    /* enable clock to GPIOF at clock gating control register */
    SYSCTL->RCGCGPIO |= 0x20;
    /* enable the GPIO pins for the LED (PF3, 2 1) as output */
    GPIOF->DIR = 0x0E;
    /* enable the GPIO pins for digital function */
    GPIOF->DEN = 0x0E;

    while(1)
    {
        GPIOF->DATA = 2;          /* turn on red LED */
        timer0A_delayMs(2);       /* Timer A msec delay */
        GPIOF->DATA = 0;          /* turn off red LED */
        delayMs(3);               /* use old delay function */
    }
}
```

Toggling PF1 LED using One-shot Mode of Timer A of Timer 0

```
/* millisecond delay using one-shot mode */
void timer0A_delayMs(int ttime)
{
    SYSCTL->RCGCTIMER |= 1;      /* enable clock to Timer Block 0 */
    TIMER0->CTL = 0;             /* disable Timer before initialization */
    TIMER0->CFG = 0x04;          /* 16-bit option */
    TIMER0->TAMR = 0x01;         /* one-shot mode and down-counter */
    TIMER0->TAILR = 16000*ttime-1;  /* Timer A interval load value register*/
    TIMER0->ICR = 0x1;           /* clear the TimerA timeout flag*/
    TIMER0->CTL |= 0x01;         /* enable Timer A after initialization*/
    while ((TIMER0->RIS & 0x1)==0);  /* wait for TimerA timeout flag to set*/
}

/* delay n milliseconds (16 MHz CPU clock) */
void delayMs(int n)
{
    int i, j;
    for(i = 0 ; i < n; i++)
        for(j = 0; j < 3180; j++)
        {}  /* do nothing for 1 ms */
}
```

Longer Delay: Prescaler Register for Timer A

- 16비트 모드 Timer A는 8-bit prescaler 레지스터를 가짐 (0x00~0xFF)
- down counter일 때만 작동됨

```
TIMER1->TAPR = 250 - 1;      /* TimerA Prescaler 16MHz/250=64000Hz */
```

Toggling PF2 LED every second using Prescaler

```
/* This program incorporates the use of prescaler to create a longer delay. Also the
use of timer has been changed to TimerA of timer Block 1. */

#include “TM4C123GH6PM.h”

void timer1A_delaySec(int ttime);

int main (void)
{
    /* enable clock to GPIOF at clock gating control register */
    SYSCTL->RCGCGPIO |= 0x20;
    /* enable the GPIO pins for the LED (PF3, 2, and 1) as output */
    GPIOF->DIR = 0x0E;
    /* enable the GPIO pins for digital function */
    GPIOF->DEN = 0x0E;

    while(1)
    {
        GPIOF->DATA = 4;          /* turn on blue LED */
        timer1A_delaySec(1);      /* TimerA 1 sec delay */
        GPIOF->DATA = 0;          /* turn off blue LED */
        timer1A_delaySec(1);      /* TimerA 1 sec delay */
    }
}
```

Toggling PF2 LED every second using Prescaler

```
/* multiple of second delays using periodic mode and prescaler*/
void timer1A_delaySec(int ttime)
{
    int i;
    SYSCTL->RCGCTIMER |= 2;      /* enable clock to Timer Block 1 */

    TIMER1->CTL = 0;             /* disable Timer before initialization */
    TIMER1->CFG = 0x04;          /* 16-bit option */
    TIMER1->TAMR = 0x02;         /* periodic mode and down-counter */
    TIMER1->TAILR = 64000 - 1;   /* TimerA interval load value reg */
    TIMER1->TAPR = 250 - 1;      /* TimerA Prescaler 16MHz/250=64000Hz */
    TIMER1->ICR = 0x1;           /* clear the TimerA timeout flag */
    TIMER1->CTL |= 0x01;         /* enable Timer A after initialization */

    for(i = 0; i < ttime; i++)
    {
        while ((TIMER1->RIS & 0x1) == 0);      /* wait for TimerA timeout flag */
        TIMER1->ICR = 0x1;       /* clear the TimerA timeout flag */
    }
}
```

Pin Selection for Timers

Use of timers with the I/O pins

- **5 modes for each timer block**
  1. One-shot / Periodic mode
  2. Real-time clock mode
  3. Input edge-time mode
  4. Input edge-count mode

  } will be covered here
  5. Pulse width modulation (PWM)

Use of I/O pins with the Timer

- **Edge-time과 edge-count 모드에서 각각의 타이머는 1 혹은 2개의 핀과 연결되어 있음**

	TimerA	Pins	TimerB	Pins
Timer0	T0CCP0	PB6 or PF0	T0CCP1	PB7 or PF1
Timer1	T1CCP0	PB4 or PF2	T1CCP1	PB5 or PF3
Timer2	T2CCP0	PB0 or PF4	T2CCP1	PB1
Timer3	T3CCP0	PB2	T3CCP1	PB3
Timer4	T4CCP0	PC0	T4CCP1	PC1
Timer5	T5CCP0	PC2	T5CCP1	PC3

Table 5-2: the pin designation for 16/32-bit Timer block 0 to 5

Selecting alternate function for Timers pin

- **Upon reset, the I/O pins are used as simple I/O.**
- **To use an alternate function, we must set the bit in the AFSEL register to 1 for that pin.**

Timer Pin	I/O pin	How to use peripheral function on the pin
T0CCP0	PB6	GPIOB->AFSEL =0x40 (0100 0000 binary)
	PF0	GPIOF->AFSEL =0x01 (0000 0001 binary)
T0CCP1	PB7	GPIOB->AFSEL =0x80 (1000 0000 binary)
	PF1	GPIOF->AFSEL =0x02 (0000 0010 binary)
T1CCP0	PB4	GPIOB->AFSEL =0x10 (0001 0000 binary)
	PF2	GPIOF->AFSEL =0x04 (0000 0100 binary)
T1CCP1	PB5	GPIOB->AFSEL =0x20 (0010 0000 binary)
	PF3	GPIOF->AFSEL =0x08 (0000 1000 binary)
T2CCP0	PB0	GPIOB->AFSEL =0x40 (0000 0001 binary)
	PF4	GPIOF->AFSEL =0x10 (0001 0000 binary)
T2CCP1	PB1	GPIOB->AFSEL =0x02 (0000 0010 binary)
T3CCP0	PB2	GPIOB->AFSEL =0x04 (0000 0100 binary)
T3CCP1	PB3	GPIOB->AFSEL =0x08 (0000 1000 binary)
T4CCP0	PC0	GPIOC->AFSEL =0x01 (0000 0001 binary)
T4CCP1	PC1	GPIOC->AFSEL =0x02 (0000 0010 binary)
T5CCP0	PC2	GPIOC->AFSEL =0x04 (0000 0100 binary)
T5CCP1	PC3	GPIOC->AFSEL =0x08 (0000 1000 binary)
WT0CCP0	PC4	GPIOC->AFSEL =0x10 (0001 0000 binary)
WT0CCP1	PC5	GPIOC->AFSEL =0x20 (0010 0000 binary)
WT1CCP0	PC6	GPIOC->AFSEL =0x40 (0100 0000 binary)
WT1CCP1	PC7	GPIOC->AFSEL =0x80 (1000 0000 binary)
WT2CCP0	PD0	GPIOD->AFSEL =0x01 (0000 0001 binary)
WT2CCP1	PD1	GPIOD->AFSEL =0x02 (0000 0010 binary)
WT3CCP0	PD2	GPIOD->AFSEL =0x04 (0000 0100 binary)
WT3CCP1	PD3	GPIOD->AFSEL =0x04 (0000 1000 binary)
WT4CCP0	PD4	GPIOD->AFSEL =0x10 (0001 0000 binary)
WT4CCP1	PD5	GPIOD->AFSEL =0x20 (0010 0000 binary)
WT5CCP0	PD6	GPIOD->AFSEL =0x40 (0100 0000 binary)
WT5CCP1	PD7	GPIOD->AFSEL =0x80 (1000 0000 binary)

Table 5-3: Timers alternate pin assignment

Timer Pin	I/O Pin	How to select the timer function on the pin
T0CCP0	PB6	GPIOB->PCTL=0x0700 0000
	PF0	GPIOF->PCTL=0x0000 0007
T0CCP1	PB7	GPIOB->PCTL=0x7000 0000
	PF1	GPIOF->PCTL=0x0000 0070
T1CCP0	PB4	GPIOB->PCTL=0x0007 0000
	PF2	GPIOF->PCTL=0x0000 0700
T1CCP1	PB5	GPIOB->PCTL=0x0070 0000
	PF3	GPIOF->PCTL=0x0000 7000
T2CCP0	PB0	GPIOB->PCTL=0x0000 0007
	PF4	GPIOF->PCTL=0x0007 0000
T2CCP1	PB1	GPIOB->PCTL=0x0000 0070
T3CCP0	PB2	GPIOB->PCTL=0x0000 0700
T3CCP1	PB3	GPIOB->PCTL=0x0000 7000
T4CCP0	PC0	GPIOC->PCTL=0x0000 0007
T4CCP1	PC1	GPIOC->PCTL=0x0000 0070
T5CCP0	PC2	GPIOC->PCTL=0x0000 0700
T5CCP1	PC3	GPIOC->PCTL=0x0000 7000
WT0CCP0	PC4	GPIOC->PCTL=0x0007 0000
WT0CCP1	PC5	GPIOC->PCTL=0x0070 0000
WT1CCP0	PC6	GPIOC->PCTL=0x0700 0000
WT1CCP1	PC7	GPIOC->PCTL=0x7000 0000
WT2CCP0	PD0	GPIOD->PCTL=0x0000 0007
WT2CCP1	PD1	GPIOD->PCTL=0x0000 0070
WT3CCP0	PD2	GPIOD->PCTL=0x0000 0700
WT3CCP1	PD3	GPIOD->PCTL=0x0000 7000
WT4CCP0	PD4	GPIOD->PCTL=0x0007 0000
WT4CCP1	PD5	GPIOD->PCTL=0x0070 0000
WT5CCP0	PD6	GPIOD->PCTL=0x0700 0000
WT5CCP1	PD7	GPIOD->PCTL=0x7000 0000

Table 5-4: Timers pin assignment using GPIO_PCTL (Extracted from Table 23-5 of Tiva data sheet)

Using Timers for Input Edge-Time Capturing

Input Edge-Time Mode

- **Input edge-time 모드에서는 특정 I/O핀의 시그널 전환 이벤트를 capture 한다.**
- **이벤트가 발생하면 그 순간의 카운터의 값이 다른 레지스터로 복사되고, 카운팅은 지속된다.**
- **타이머를 input edge time 모드로 설정하려면 GPTMTnMR의 TnMR과 TnCMR 비트를 각각 capture 그리고 edge time 모드로 설정해야 한다. (TnMR=3, TnCMR=1)** **==> see 18 page**

Input Edge-Time Capturing

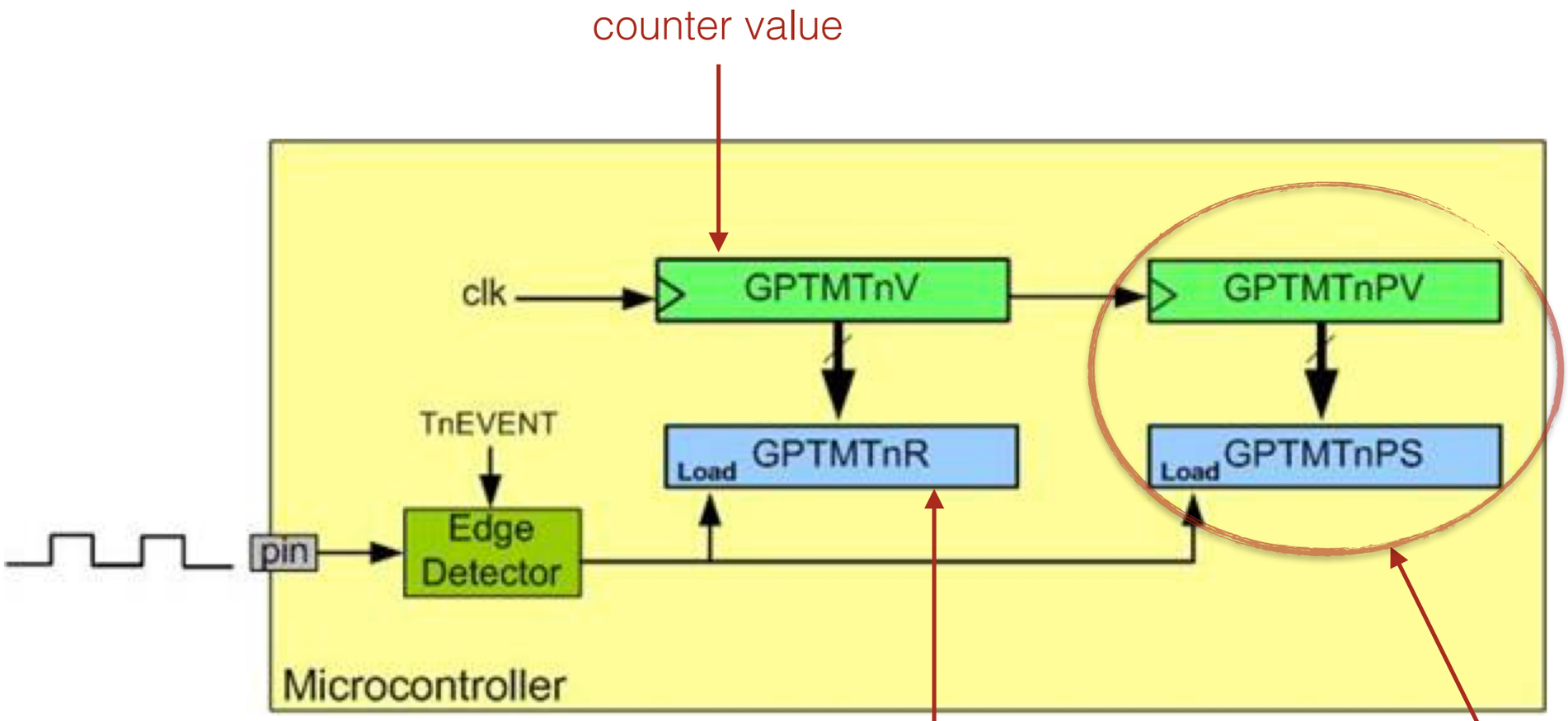


Figure 5-21: Input Edge Time Capturing

64/32-bit 모드에서 사용

event가 발생하면 counter value가
이곳으로 복사됨

Timer counting in input edge time mode

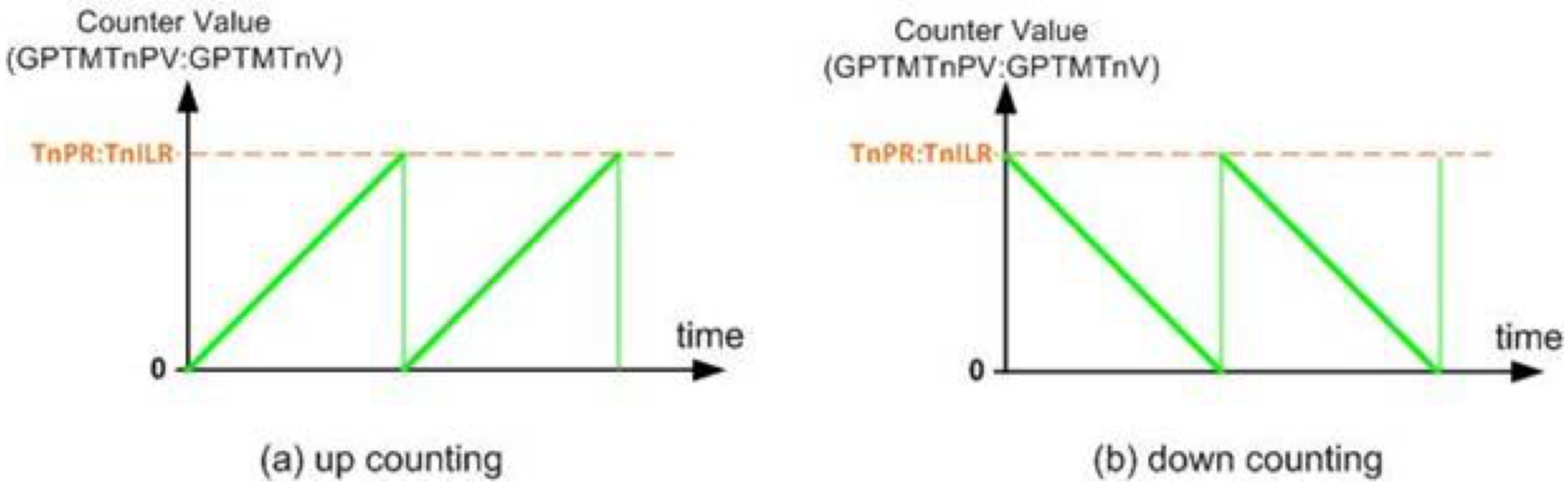


Figure 5-22: Counting in Input Edge-Time Mode

Measuring Period and Pulse Width

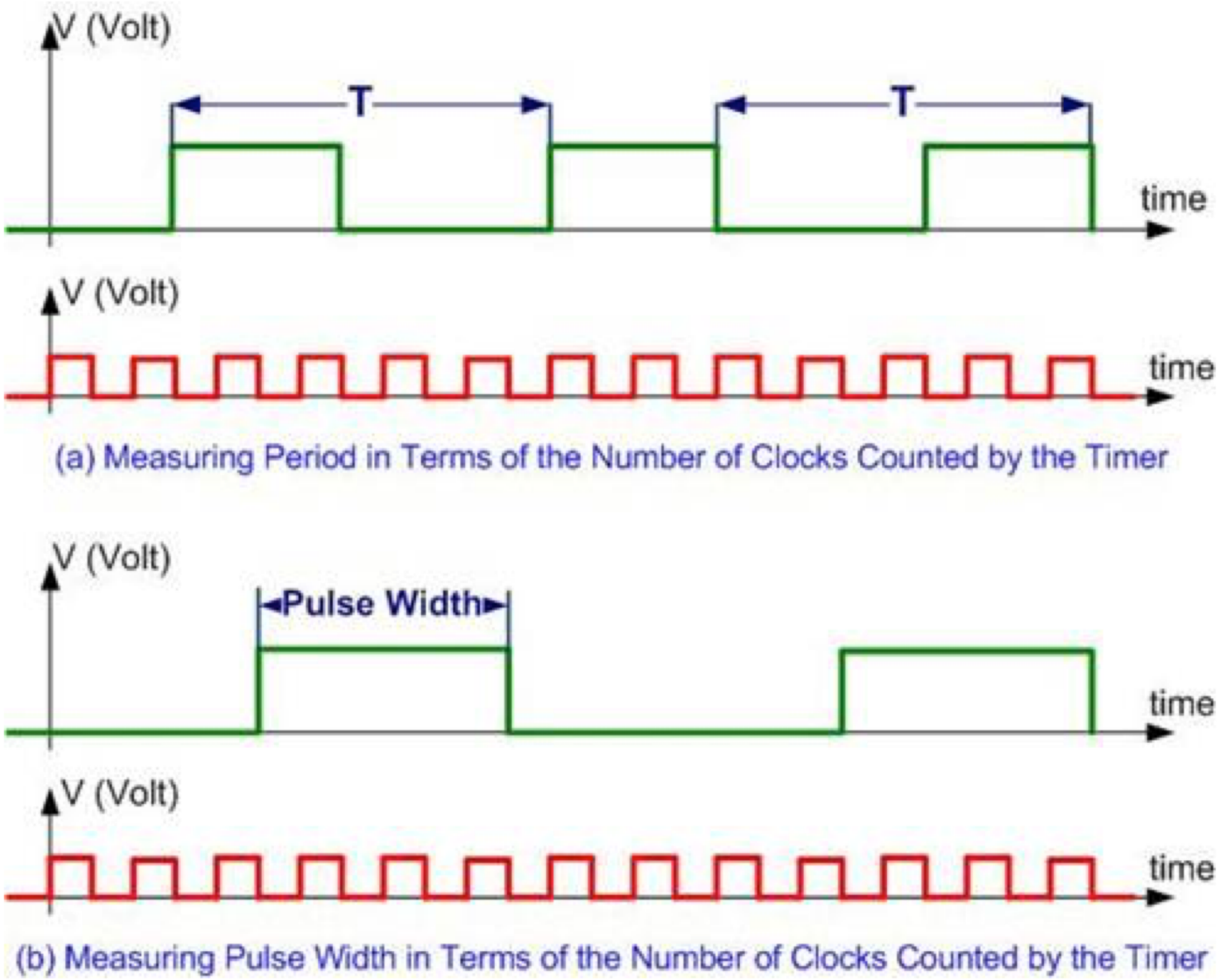


Figure 5-23: Measuring Period and Pulse Width

Edge-Time Capture

```
/* Square wave signal should be fed to PB6 pin. Make sure it is 3.3 to 5V peak-to-
peak. Initialize Timer0A in edge-time mode to capture rising edges. The input pin
of Timer0A is PB6. */

#include “TM4C123GH6PM.h”

void Timer0Capture_init(void)
{
    SYSCTL->RCGCTIMER |= 1;      /* enable clock to Timer Block 0 */
    SYSCTL->RCGCGPIO |= 2;       /* enable clock to PORTB */
    GPIOB->DIR &= ~0x40;         /* make PB6 an input pin */
    GPIOB->DEN |= 0x40;          /* make PB6 as digital pin */
    GPIOB->AFSEL |= 0x40;        /* use PB6 alternate function */
    GPIOB->PCTL &= ~0x0F000000;  /* configure PB6 for T0CCP0 */
    GPIOB->PCTL |= 0x07000000;

    TIMER0->CTL &= ~1;           /* disable timer0A during setup */
    TIMER0->CFG = 4;             /* 16-bit timer mode */
    TIMER0->TAMR = 0x17;         /* up-count, edge-time, capture mode */
    TIMER0->CTL &= ~0x0C;        /* capture the rising edge */
    TIMER0->CTL |= 1;            /* enable timer0A */
}
```

Edge-Time Capture

```
/* This function captures two consecutive rising edges of a periodic signal from
Timer Block 0 Timer A and returns the time difference (the period of the signal).
*/
int Timer0A_periodCapture(void)
{
    int lastEdge, thisEdge;

    /* capture the first rising edge */
    TIMER0->ICR = 4;              /* clear timer0A capture flag */
    while((TIMER0->RIS & 4) == 0) ;    /* wait till captured */
    lastEdge = TIMER0->TAR;       /* save the timestamp */

    /* capture the second rising edge */
    TIMER0->ICR = 4;              /* clear timer0A capture flag */
    while((TIMER0->RIS & 4) == 0) ;    /* wait till captured */
    thisEdge = TIMER0->TAR;       /* save the timestamp */

    return (thisEdge - lastEdge) & 0x00FFFFFF; /* return the time difference */
}
```

Edge-Time Capture

```c
#include "TM4C123GH6PM.h"
#include "serial.h"
#include "timer.h"
#include <stdio.h>

void delayMs(int n);

int main(void)
{
    char str[20];
    init_serial();

    printString("Ready\n\r");
    delayMs(500);

    Timer0Capture_init();

    for (int n=0; n<10; n++) {
        delayMs(1000);
        int r = Timer0A_periodCapture();
        sprintf(str, "%d\n\r", r);
        printString(str);
    }
    printString("End.\n\r");
}
```

serial.h와 serial.c가 필요하다.

실습과제

- **Arduino의 PWM을 이용하여 square wave를 생성하라. 생성된 펄스의 주기를 Launchpad를 이용하여 측정하여 serial terminal로 전송하는 프로그램을 작성하라. Arduino 9번 핀의 PWM의 default frequency는 500Hz이다. (16MHz/500Hz ≃ 32000에 가까운 값이 출력되면 정상이다.)**

Using Timers As a Counter

Edge-Counter

- A timer works as a counter when the TAMR bits of the GPTMTnMR are configured to capture mode and the TACMR bit is cleared.
- The timer counts whenever the input pin is triggered.
- Can count the falling edge, rising edge, or both.
- To determine the type, the TnEVENT bits of the GPTMCTL should be initialised.

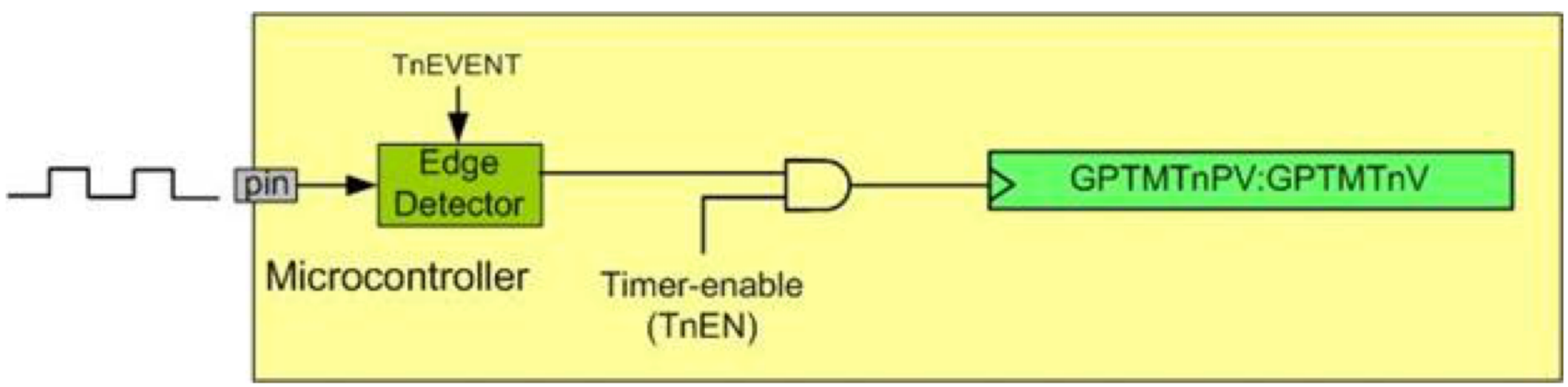


Figure 5-24: Counter Diagram

Input Edge Count Mode

```
/* Initialize Timer2A to capture rising edge in edge-count mode. */
/* The input pin of Timer2A is PF4.*/
void Timer2A_countCapture_init(void)
{
    SYSCTL->RCGCTIMER |= 4;      /* enable clock to Timer Block 2 */
    SYSCTL->RCGCGPIO |= 0x20;    /* enable clock to PORTF */

    GPIOF->DIR &= ~0x10;         /* make PF4 an input pin */
    GPIOF->DEN |= 0x18;          /* make PF4 a digital pin */
    GPIOF->PUR = 0x10;

    GPIOF->AFSEL |= 0x10;          /* enable alternate function on PF4 */
    GPIOF->PCTL &= ~0x000F0000;  /* configure PF4 as T2CCP0 pin */
    GPIOF->PCTL |= 0x00070000;

    TIMER2->CTL &= ~1;           /* disable TIMER2A during setup */
    TIMER2->CFG = 4;             /* configure as 16-bit timer mode */
    TIMER2->TAMR = 0x13;         /* up-count, edge-count, capture mode */
    TIMER2->TAMATCHR = 0xFFFF;   /* set the count limit */
    TIMER2->TAPMR = 0xFF;          /* to 0xFFFFFF with prescaler */
    TIMER2->CTL &= ~0xC;         /* capture the rising edge */
    TIMER2->CTL |= 1;            /* enable timer2A */
}

int Timer2A_countCapture(void)
{
    return TIMER2->TAR;
}
```

실습과제

- **Launchpad의 1번 스위치를 누른 횟수를 카운트하여 serial로 전송하는 프로그램을 작성하라.**