임베디드시스템 기초(#514115)

#5. Timer A

한림대학교
전자공학과 이선우

한림대학교
HALLYM UNIVERSITY

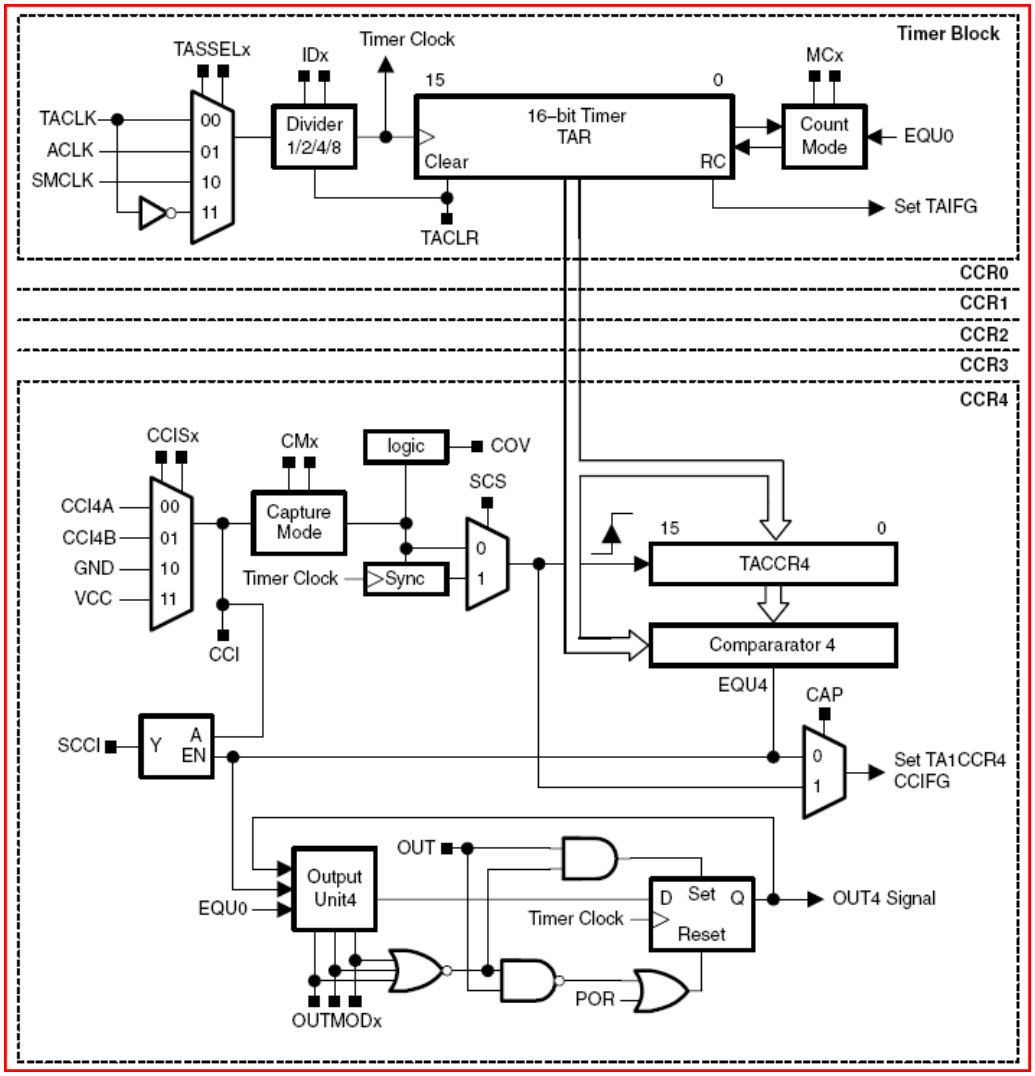
MSP430x4xx 타이머 종류

- MSP430x4xx series는 다음과 같은 3종의 타이머 내장
  - Basic Timer1
    - Two independent, cascadable 8-bit timers
    - Selectable clock source
    - Interrupt capability
    - LCD control signal generation
  - Timer A, Timer B
    - 16-bit timer/counter with 3 or 5 capture/compare registers
    - Multiple capture/compares, PWM outputs, interval timing

Features of Timer A

- 4개의 동작 모드를 가지는 비동기 16-bit timer/counter
- 클럭 소스 선택/설정 가능
- 칩에 따라 3개 혹은 5개의 capture/compare regs. 가짐
  - **의미: 변하는 카운터 값과 비교하여 동작 하는 제어 장치가 3/5개 있음을 뜻함.**
  - MSP430FG4618: Timer_A3 (3개의 CCRs이 있음)
- PWM 기능을 가지는 설정 가능한 출력 기능
  - Pin을 통해 H/L 출력 가능
- 비동기 입/출력 레칭(latching)
- Timer_A 관련 인터럽트 발생 소스에 대한 빠른 해석을 위한 Interrupt vector reg.(TAIV)를 가짐.

Block diagram



▪ MSP430xG461x의 경우 CCR0~CCR2까지 3개의 제어장치만 존재

➔ TAR 값을 이용하여 별도 3개의 TACCR0/1/2를 이용

Related control registers

Table 15–3. Timer_A3 Registers

Register	Short Form	Register Type	Address	Initial State
Timer_A control Timer0_A3 Control	TACTL/ TA0CTL	Read/write	0160h	Reset with POR
Timer_A counter Timer0_A3 counter	TAR/ TA0R	Read/write	0170h	Reset with POR
Timer_A capture/compare control 0 Timer0_A3 capture/compare control 0	TACCTL0/ TA0CCTL	Read/write	0162h	Reset with POR
Timer_A capture/compare 0 Timer0_A3 capture/compare 0	TACCR0/ TA0CCR0	Read/write	0172h	Reset with POR
Timer_A capture/compare control 1 Timer0_A3 capture/compare control 1	TACCTL1/ TA0CCTL1	Read/write		Reset with POR
Timer_A capture/compare 1 Timer0_A3 capture/compare 1	TACCR1/ TA0CCR1	Read/write	0174h	Reset with POR
Timer_A capture/compare control 2 Timer0_A3 capture/compare control 2	TACCTL2/ TA0CCTL2	Read/write	0166h	Reset with POR
Timer_A capture/compare 2 Timer0_A3 capture/compare 2	TACCR2/ TA0CCR2	Read/write	0176h	Reset with POR
Timer_A interrupt vector Timer0_A3 interrupt vector	TAIV/ TA0IV	Read only	012Eh	Reset with POR

Timer block을 제어하는 레지스터. 즉 **TAR 동작 제어**

Rising edge clock signal 입력에 따라 값 증가/감소

Compare/Capture 제어 블록#0 제어 레지스터

Compare/Capture 관련 데이터 레지스터 (16bit). 동작설정에 따라 TAR과 비교/TAR값 캡쳐된 결과 저장

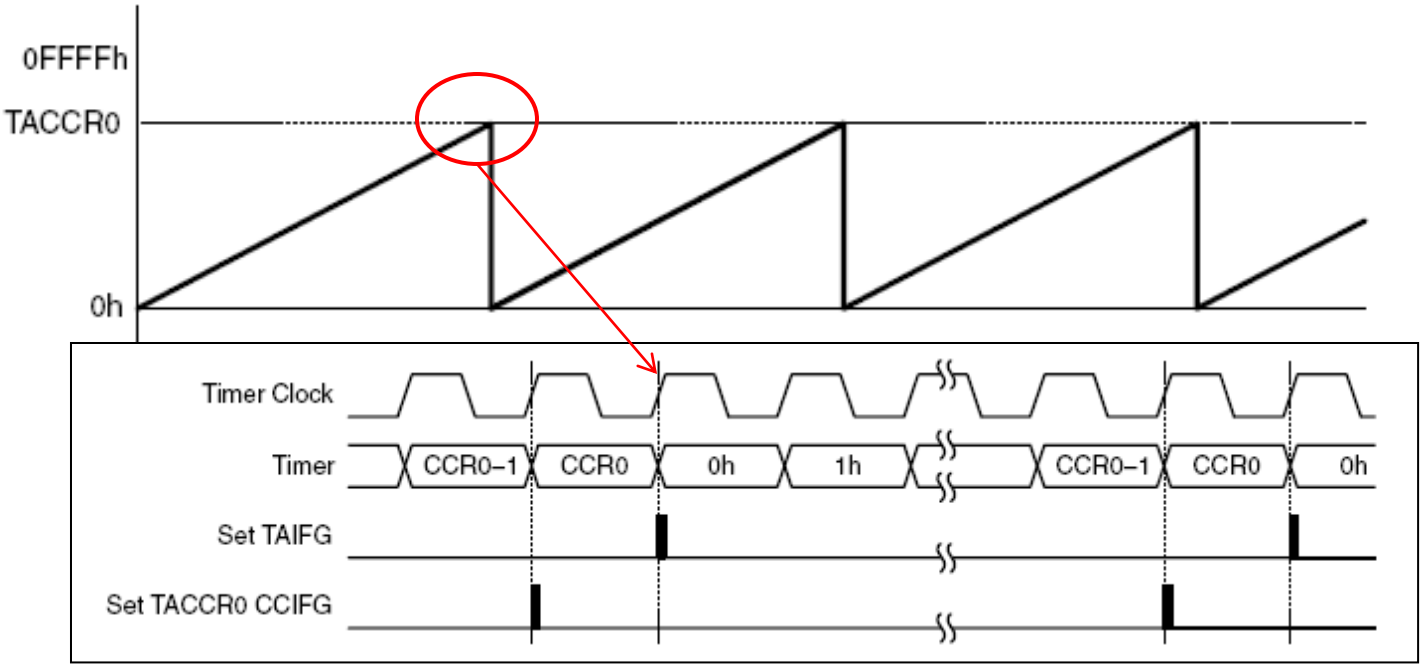
- TACTL,TA0CTL의 차이점: 일부 장치(430x415/417,xW42x)는 동일한 TimerA를 2개 가짐. 430x461x 장치는 1개의 TA를 가지므로 TA만 사용함.
- 2개 Timer_A를 가지는 장치는 TA0/1을 구별하기 위해 TA0CTRL/TA1CTL로 다른 이름의 레지스터를 가짐. (표15-4 참조)

동작 방식

- Timer block
  - 일반적인 16비트 up/down 카운터로 동작
  - 클럭 소스: 3개 중 하나 (ACLK, SMCLK, TACLK)
    - TACLK: 외부에서 공급하는 클럭 신호 (x461x 장치:P1.5)
  - Divider 제공: /2,/4,/8 중 하나 선택 가능 (Idx bits)
  - Reset 기능: TACLR bit=1로 TAR, Idx, MCx clear. Automatic reset.
  - 4개 Mode
    - MCx bit(in TACTL reg.)로 설정
      - 00: Stop → Timer_A 정지, 저전력 위해 사용하지 않을 때 사용
    - MCx>0: Running
      - 01:Up, 10:continuous, 11:Up/down

UP MODE

- 작동방식: TAR 값이 증가하다 CCR0 reg.에 써있는 값 (>0)이 되면 인터럽트를 발생시킴
  - 정확하게는 다른 종류의 2개 인터럽트 (CCIFG



Up mode 이용하여 특정 인터벌 만들기

```
#include <msp430xG46x.h>

#pragma vector=TIMERA0_VECTOR
__interrupt void timer_handler(void)
{
  P1OUT ^= 0x01;  //toggle P1.0
}

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD

  P1DIR |= 0x01;

  //setup Timer_A
  TACCTL0 = 0x0010;  // 0000 0000 0001 0000
  TACCR0 = 13180; //13180 usec
  TACTL = 0x0210  ; //0000 0010 0001 0000
  __enable_interrupt();
  __low_power_mode_0();
}
```

요구사항: 13.18msec 의 인터벌마다 P1.0 출력 toggle시켜 클럭 신호 발생

15	14	13	12	11	10	9	8
CMx		CCISx		SCS	SCCI	Unused	CAP
rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	r	r0	rw–(0)

7	6	5	4	3	2	1	0
OUTMODx			CCIE	CCI	OUT	COV	CCIFG
rw–(0)	rw–(0)	rw–(0)	rw–(0)	r	rw–(0)	rw–(0)	rw–(0)

15	14	13	12	11	10	9	8
Unused						TASSELx	
rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)

7	6	5	4	3	2	1	0
IDx		MCx		Unused	TACLR	TAIE	TAIFG
rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	w–(0)	rw–(0)	rw–(0)

Timer_A3 관련 Interrupts

- 2개의 인터럽트 소스
  - TACCR0 CCIFG0 (IAR C에선 TIMERA0_VECTOR)
  - CCIFG1, CCIFG2, TAIFG는 모두 하나의 인터럽트로 처리됨. (P1/2와 동일) (TIMERA1_VECTOR)

ADC12	ADC12IFG (see Notes 1 and 2)	Maskable	0FFEEh	23
Timer_A3	TACCR0 CCIFG0 (see Note 2)	Maskable	0FFECh	22
Timer_A3	TACCR1 CCIFG1 and TACCR2 CCIFG2, TAIFG (see Notes 1 and 2)	Maskable	0FFEAh	21
I/O Port P1 (Eight Flags)	P1IFG.0 to P1IFG.7 (see Notes 1 and 2)	Maskable	0FFE8h	20

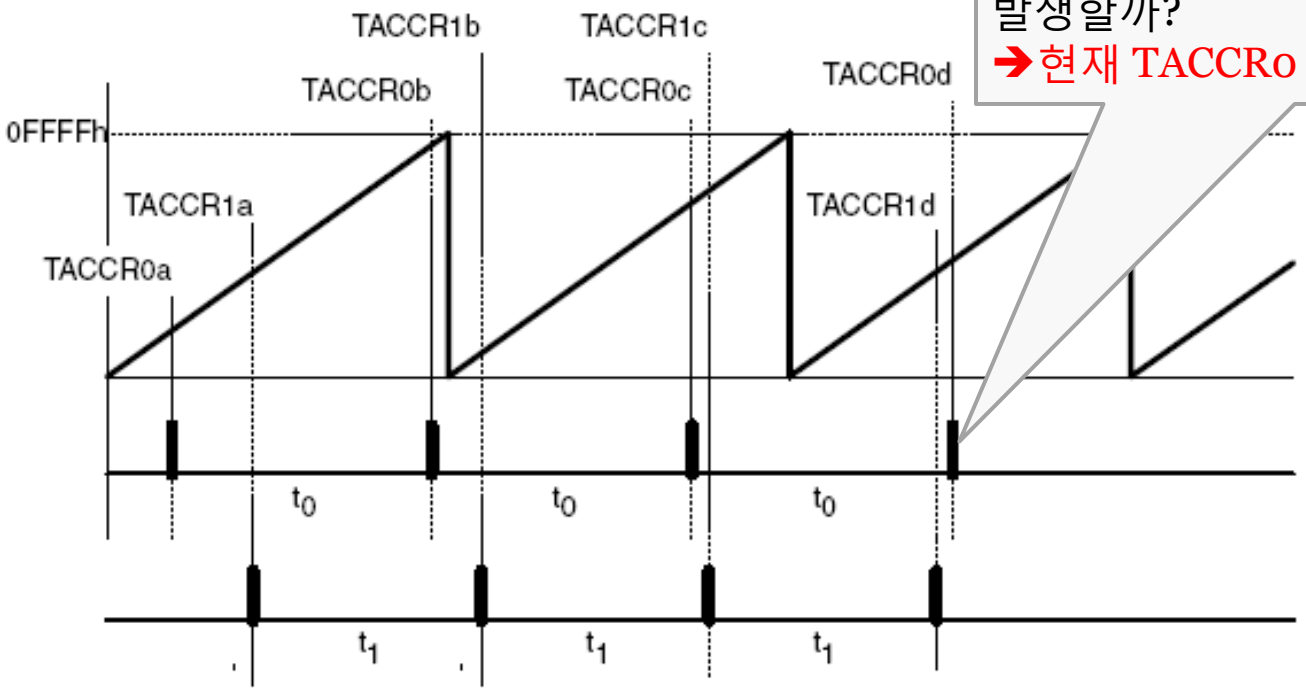
  - CCIFG1/2 및 TAIFG는 순서대로 우선순위를 가지며 ISR에서 어떤 소스인가를 보다 빠르게 파악할 수 있도록 TAIV reg.를 제공함.

TAIV Contents	Interrupt Source	Interrupt Flag	Interrupt Priority
00h	No interrupt pending	–	
02h	Capture/compare 1	TACCR1 CCIFG	Highest
04h	Capture/compare 2	TACCR2 CCIFG	
06h	Capture/compare 3†	TACCR3 CCIFG	
08h	Capture/compare 4†	TACCR4 CCIFG	
0Ah	Timer overflow	TAIFG	
0Ch	Reserved	–	
0Eh	Reserved	–	Lowest

† Timer1_A5 only

Continuous Mode의 이용 방법

- 하나의 TAR(free-running counter)과 여러 개의 CCRx에 저장된 값과의 비교를 이용하여 복수개의 인터벌 발생이 가능



CCR0 ISR에서 어떤 작업을 해야 동일한 시간간격 후에 다시 인터럽트가 발생할까?
➔현재 TACCR0 += TAR+INTERVAL

복수 인터벌 만들기 (연속모드 이용법)

```
#include <msp430xG46x.h>
#define CCR1INTVAL 8000          //8msec
#define CCR2INTVAL 3000          //3msec

 #pragma vector=TIMERA1_VECTOR
__interrupt void ccr_handler(void)
{
switch (TAIV)
  {
    case 10: break;  //TAIFG not used
    case 2:  //use TACCR1
      P1OUT ^= 0x01;  //toggle P1.0
     TACCR1 += CCR1INTVAL;
      break;

    case 4: //use TACCR2
      P1OUT ^= 0x02;  //toggle P1.1
     TACCR2 += CCR2INTVAL;
      break;

  }
}
```

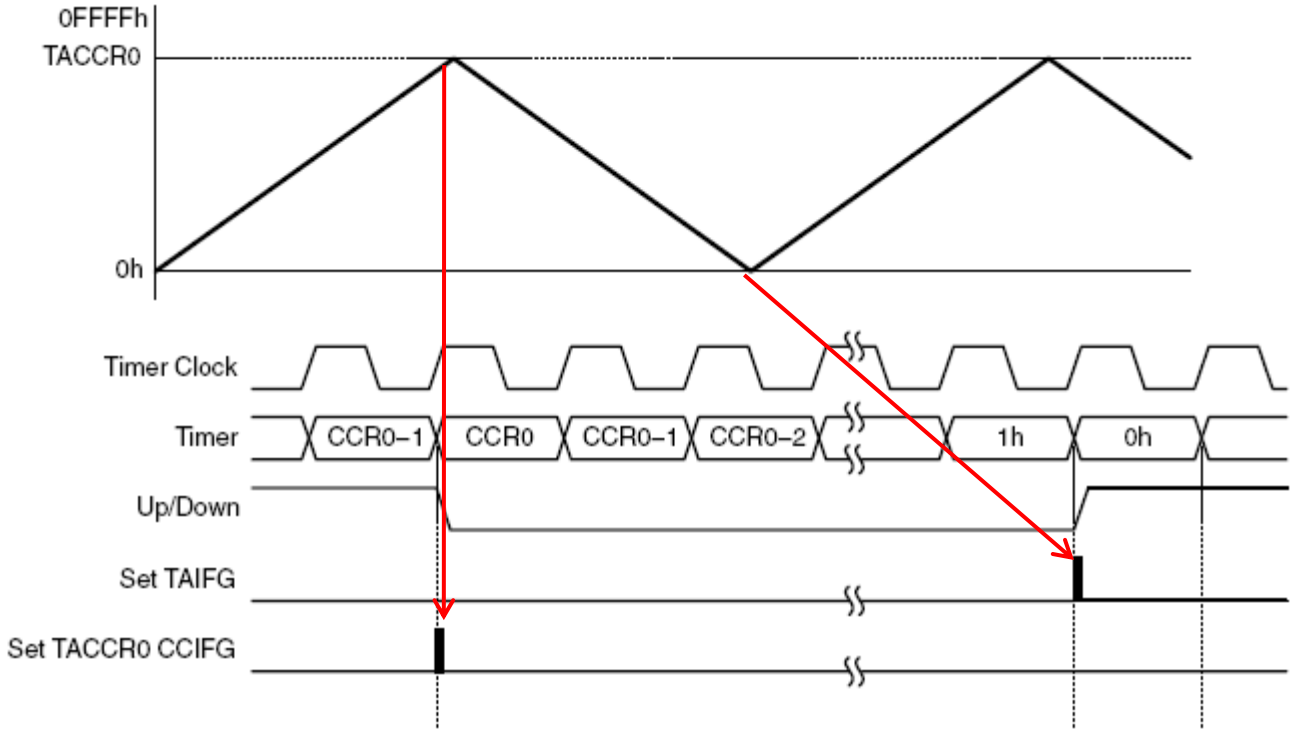
현재의 CCR2 값에 원하는 인터벌의 크기를 더하여 다음 CCR2값을 갱신함.
Ex. 현재 TAR=3000이 상태에서 TACCR2=6000이 되어 3000usec 후에 CCR2 IRQ 발생!

요구사항: CCR1 이용 8msec
인터벌마다 P1.0 출력 toggle,
CCR2이용 3msec마다 P1.1 toggle 함

```
void main(void)
{
 WDTCTL = WDTPW + WDTHOLD

 P1DIR |= 0x03;

 //setup Timer_A
 TACCTL1 = 0x0010;  // CCR1 Interrupt Enable
 TACCR1 = 8000; //8ms
 TACCTL2 = 0x0010;  // CCR2 Interrupt Enable
 TACCR2 = 3000; //8ms
 TACTL = 0x0220  ; //SSEL=SMCLK,
    MCx=continuous
 __enable_interrupt();
 __low_power_mode_0();
}
```

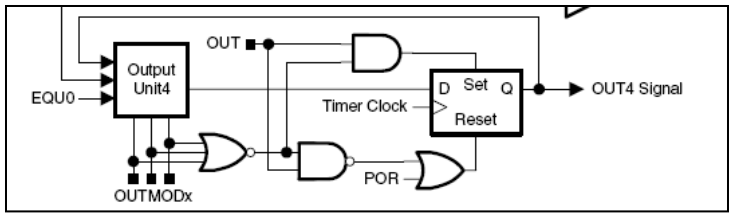
Up/down mode

- Square pulse를 만들고자 할 때 이용
- TAR이 증가 후 감소함



TA3의 입출력 핀 & Output Mode

- MSP430FG461x의 timer_A3의 입출력 핀 사양
  - 입력: P1.5(TACLK), P1.0(CCI0A), P1.2(CCI1A), P2.0(CCI2A)
  - 출력: P1.0(TA0), P1.2(TA1), P2.0(TA2)
- 각 CC block은 Output Unit이라는 장치를 가지며, 이를 통해 연결된 핀의 값을 주어진 설정에 따라 set/clear하여 원하는 신호를 만들어 낼 수 있다.



Timer_A3 Signal Connections

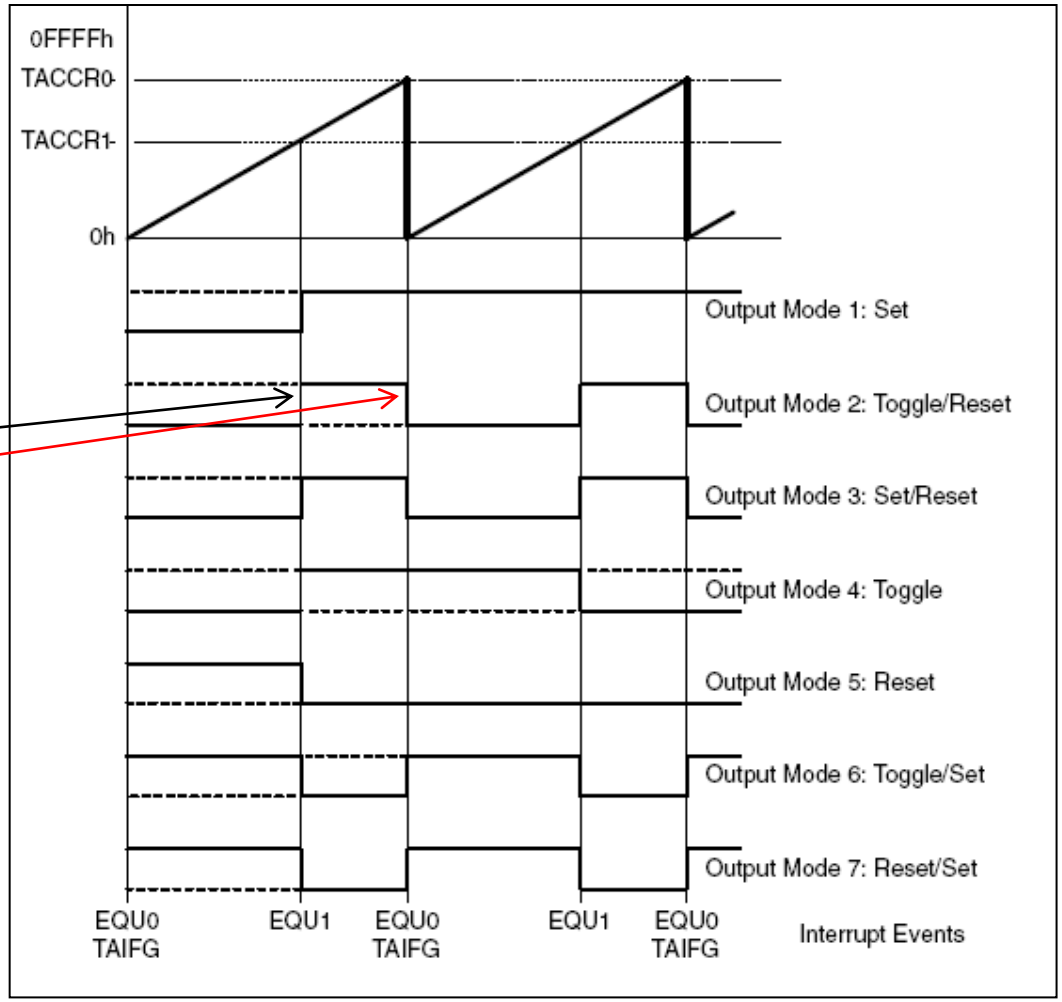
Input Pin Number PZ/ZQW	Device Input Signal	Module Input Name	Module Block	Module Output Signal	Output Pin Number PZ/ZQW
82/B9 - P1.5	TACLK	TACLK	Timer	NA	
	ACLK	ACLK			
	SMCLK	SMCLK			
82/B9 - P1.5	TACLK	INCLK			
87/A7 - P1.0	TA0	CCI0A	CCR0	TA0	87/A7 - P1.0
86/E7 - P1.1	TA0	CCI0B			
	DV_{SS}	GND			
	DV_{CC}	V_{CC}			
85/D7 - P1.2	TA1	CCI1A	CCR1	TA1	85/D7 - P1.2
	CAOUT (internal)	CCI1B			ADC12 (internal)
	DV_{SS}	GND			
	DV_{CC}	V_{CC}			
79/A10 - P2.0	TA2	CCI2A	CCR2	TA2	79/A10 - P2.0
	ACLK (internal)	CCI2B			
	DV_{SS}	GND			
	DV_{CC}	V_{CC}			

Output Unit 이용법: UP Mode

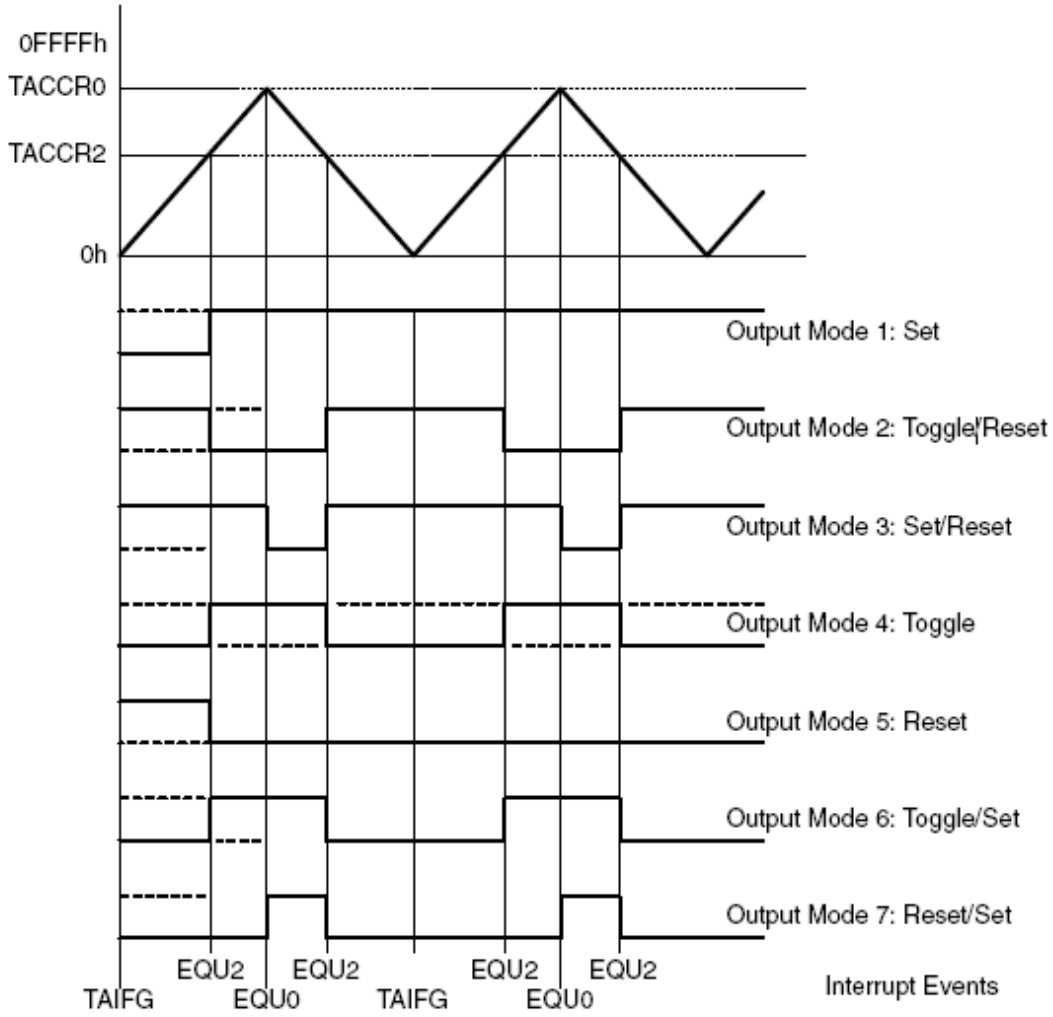
15	14	13	12	11	10	9	8
CMx		CCISx		SCS	SCCI	Unused	CAP
rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	r	r0	rw–(0)

7	6	5	4	3	2	1	0
OUTMODx			CCIE	CCI	OUT	COV	CCIFG
rw–(0)	rw–(0)	rw–(0)	rw–(0)	r	rw–(0)	rw–(0)	rw–(0)

- 7개 모드
  - Set
  - Toggle/Reset
  - Set/Reset
  - Toggle
  - Reset
  - Toggle/Set
  - Reset/Set



Output example: up/down mode



Example codes: Output unit이용 클럭신호발생

```
/*
 *  Just use OUTPUT mode, even not use IRQ.
 *  Use UP mode, CCR0
 *
 *  Result in P1.0/TA0 = 32768/200 Hz  signal.
 */
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD

  P1DIR |= 0x01;
  P1SEL |= 0x01; //set P1.0 to TA0

  //setup Timer_A
  TACCTL0 = 0x0080;  // OUTMODE=4 (Toggle)
  TACCR0 = 100-1;   //32768Hz/100
TACTL = 0x0110  ; //SSEL=ACLK,  MCx=Up
  __low_power_mode_3();
}
```

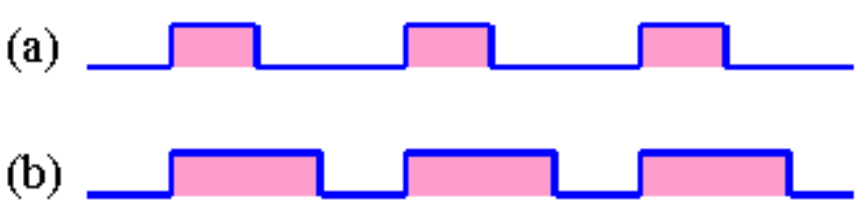
```
/*
 *  Just use OUTPUT mode, even not use IRQ.
 *  Use UP/DOWN mode, CCR0
 *
 *  Result in P1.0/TA0 = SMCLK/1000 Hz  signal.
 */
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD

  P1DIR |= 0x01;
  P1SEL |= 0x01; //set P1.0 to TA0

  //setup Timer_A
  TACCTL0 = 0x0080;  // OUTMODE=4 (Toggle)
  TACCR0 = 250;
  TACTL = 0x0230  ; //SSEL=SMCLK,
     MCx=Up/Down mode
}
```

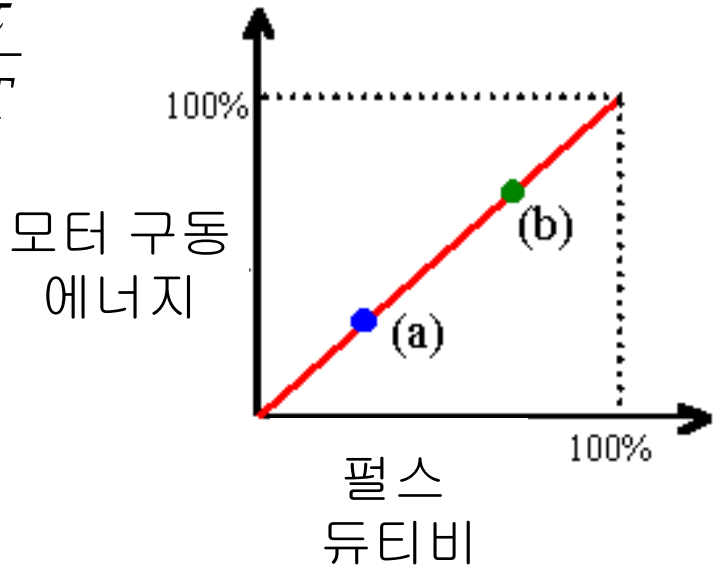
Pulse Width Modulation이란?

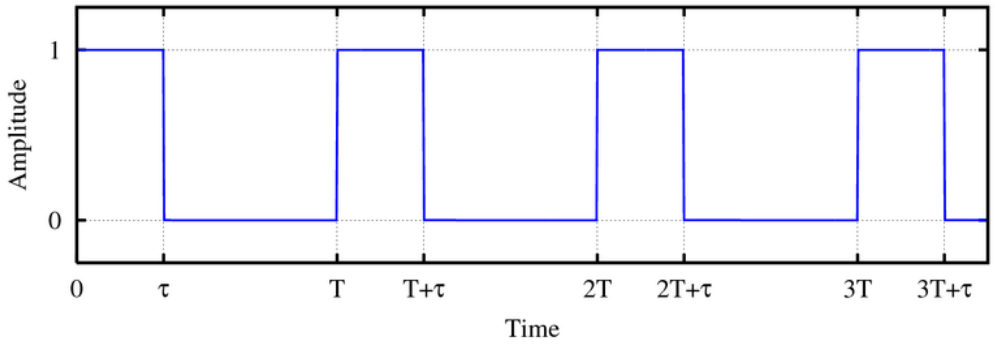
- PWM: Pulse Width Modulation
  - ON/OFF만을 이용하여 중간 크기의 에너지를 전달할 수 있는 효과적인 방법
  - Duty cycle (duty ratio):
    - ON 시간/주기의 비율로 0~100%(혹은 0~1)로 나타냄.

$$D = \frac{\tau}{T}$$

PWM에 의한 모터 속도제어 원리







Example code: PWM 신호 발생

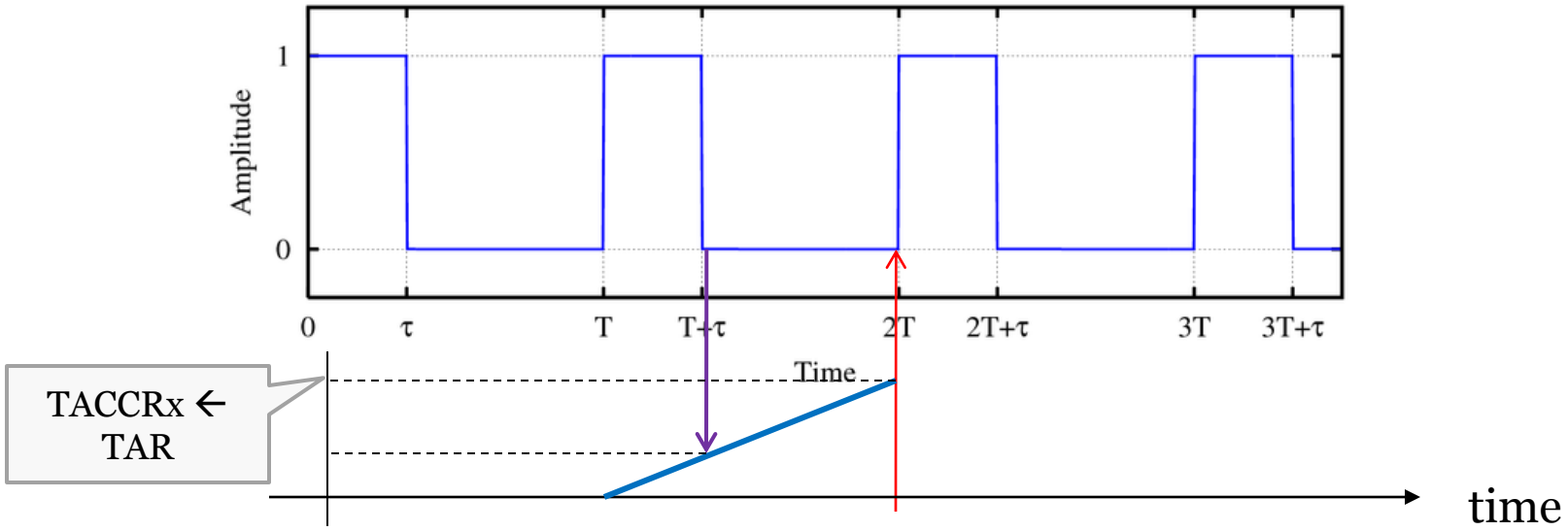
```
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD

  P1DIR |= 0x04;  /P1.2 Output
  P1SEL |= 0x04; //set P1.2 to TA1
  P2DIR |= 0x01;  /P2.0 Output
  P2SEL |= 0x01; //set P2.0 to TA2

  //setup Timer_A
  TACCR0 = 512-1;   //PWM Period
  TACCTL1 = 0x00E0;  // OUTMODE=7 (reset/set)
  TACCR1 = 384;        //duty cycle=75%
  TACCTL2 = 0x00E0;  // OUTMODE=7 (reset/set)
  TACCR2 = 128;        //duty cycle =25%
  TACTL = 0x0110  ; //SSEL=ACLK,  MCx=Up mode
__low_power_mode_3();

}
```

OUTMODE=reset/set 이용
-EQU1에서 RESET: 주기 시작
-CCR0 overflow에서 SET: ON
Timer End
➔Duty cycle=384/512 = 75%

만약 set/reset OUTMODE를 이용한다면??

Capture mode

- Capture mode는 **시간 이벤트를 기록**하기 위해 이용됨.
  즉, MCU 내/외부 신호의 변화 시간 간격을 측정할 수 있다.
  - How? **내부 타이머의 값을 특정 시간에 일시 저장하여..**
  - 외부 클럭 신호의 주기 측정 → rising edge 사이 측정
  - 외부 신호의 ON/OFF time 측정 → rising/falling edge 사이 측정

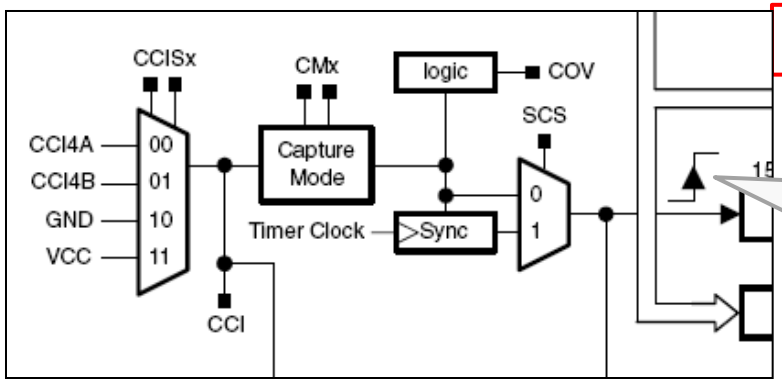


Timer_A3 입력 핀

- MSP430FG461x의 timer_A3의 입출력 핀 사양
  - 입력: P1.5(TACLK), **P1.0(CCI0A), P1.1(CCI0B), P1.2(CCI1A), P2.0(CCI2A)**
- CAP=1
  - CCISx: 측정 대상 선택
  - CMx: input signal의 capture edge 선택→ rising, falling, both

Timer_A3 Signal Connections

Input Pin Number PZ/ZQW	Device Input Signal	Module Input Name	Module Block	Module Output Signal	Output Pin Number PZ/ZQW
82/B9 - P1.5	TACLK	TACLK	Timer	NA	
	ACLK	ACLK			
	SMCLK	SMCLK			
82/B9 - P1.5	TACLK	INCLK			
87/A7 - P1.0	TA0	CCI0A	CCR0	TA0	87/A7 - P1.0
86/E7 - P1.1	TA0	CCI0B			
	DV_{SS}	GND			
	DV_{CC}	V_{CC}			
85/D7 - P1.2	TA1	CCI1A	CCR1	TA1	85/D7 - P1.2
	CAOUT (internal)	CCI1B			ADC12 (internal)
	DV_{SS}	GND			
	DV_{CC}	V_{CC}			
79/A10 - P2.0	TA2	CCI2A	CCR2	TA2	79/A10 - P2.0
	ACLK (internal)	CCI2B			
	DV_{SS}	GND			
	DV_{CC}	V_{CC}			

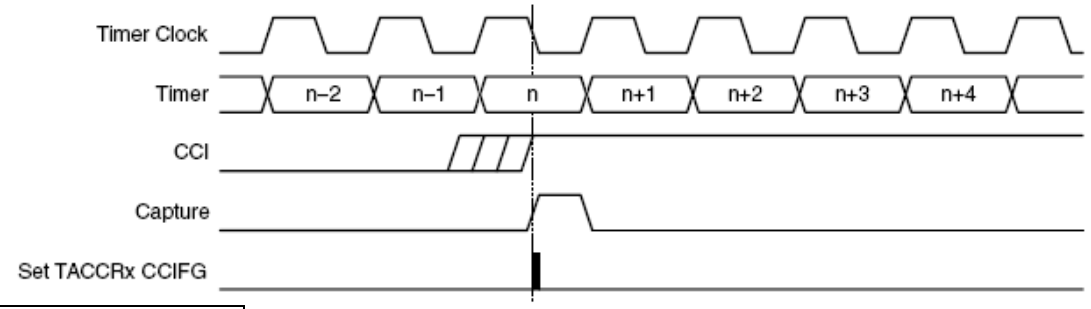


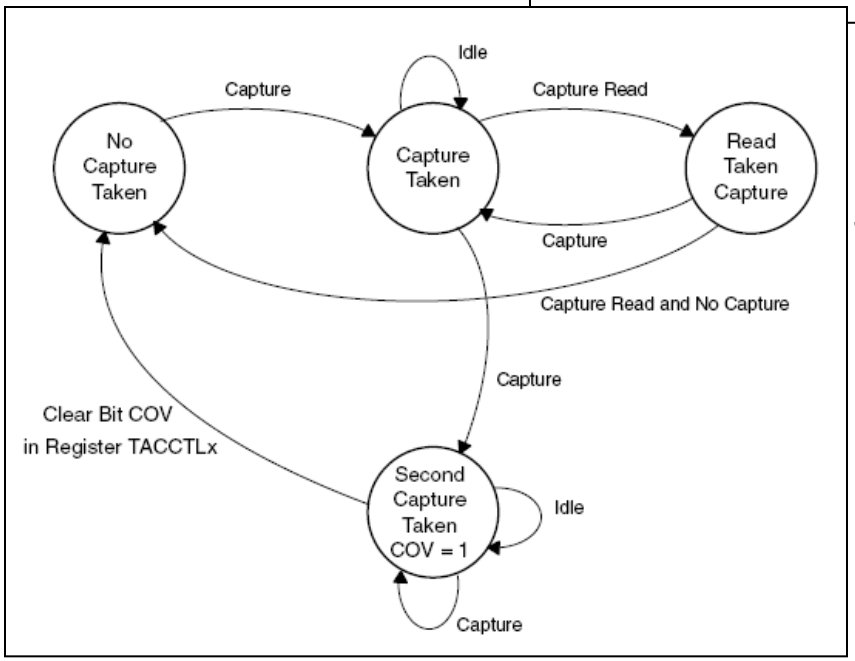
지정된 신호 입력되면(Capture occurs! 캡쳐 ON)
- TAR 값이 TACCRx reg.로 copy
- CCIFGx =1 (IRQ. 발생)

Capture mode timing diagram

SCS: TACCTLx[11]
Synchronize capture source 입력신호 변화를 타이머 클럭과 동기화 시킴.

Figure 15–10. Capture Signal (SCS=1)





Capture Overflow 문제

- 캡쳐 발생 후 일시 저장된 값을 읽지 않은 상태에서 다시 캡쳐ON 되어 기존 값을 잃어버리는 경우➔ overflow라고 함.
- 이 경우 HW가 자동적으로 찾아 COV=1로 설정함.
- COV clearing은 반드시 SW에서 실행되야 함.

Example code: 외부 신호 주기 측정

```
Int prev_cnt, Period;
int main( void )
{
  WDTCTL = WDTPW + WDTHOLD;

  //setup TA0 output
  P2SEL |= 0x01;  //P2.0 --> CCI2A
  P2DIR &= ~0x01;
  P2DIR |= 0x08; //P2.3 ouput

  //setup TACTL0 for clock generation
  TACCTL0 =  CCIE;
  TACCR0 = 3000;  //3ms
  //setup TTACTL2 for capturing
  TACCTL2 =0x4910;
  TACCR2 = 0;
  TACTL = 0x0220 ;  //continuous mode, SMCLK

  prev_cnt = 0;
    __enable_interrupt();
  __low_power_mode_0();
  return 0;
}

#pragma vector=TIMERA0_VECTOR
__interrupt void ta0_isr(void)
{
  P2OUT ^= BIT3;
  TACCR0 += INTVL1;
}

#pragma vector=TIMERA1_VECTOR
__interrupt void ta_isr(void)
{
  switch(TAIV) {
  case 2: break;
  case 10: break;
  case 4:
    Period = TACCR2 -prev_cnt;
    prev_cnt = TACCR2;

  }
}
```

Capture mode 이용

- IRQ 발생 시점: P2.3 clock rising edge
- TAR은 연속모드이므로 클럭 주기 측정을 위해서는 사이값을 계산해야함.

Timer_A3 관련 Interrupts

- 2개의 인터럽트 소스
  - TACCR0 CCIFG0 (IAR C에선 TIMERA0_VECTOR)
  - CCIFG1, CCIFG2, TAIFG는 모두 하나의 인터럽트로 처리됨. (P1/2와 동일) (TIMERA1_VECTOR)

ADC12	ADC12IFG (see Notes 1 and 2)	Maskable	0FFEEh	23
Timer_A3	TACCR0 CCIFG0 (see Note 2)	Maskable	0FFECh	22
Timer_A3	TACCR1 CCIFG1 and TACCR2 CCIFG2, TAIFG (see Notes 1 and 2)	Maskable	0FFEAh	21
I/O Port P1 (Eight Flags)	P1IFG.0 to P1IFG.7 (see Notes 1 and 2)	Maskable	0FFE8h	20

  - CCIFG1/2 및 TAIFG는 순서대로 우선순위를 가지며 ISR에서 어떤 소스인가를 보다 빠르게 파악할 수 있도록 TAIV reg.를 제공함.

TAIV Contents	Interrupt Source	Interrupt Flag	Interrupt Priority
00h	No interrupt pending	–	
02h	Capture/compare 1	TACCR1 CCIFG	Highest
04h	Capture/compare 2	TACCR2 CCIFG	
06h	Capture/compare 3†	TACCR3 CCIFG	
08h	Capture/compare 4†	TACCR4 CCIFG	
0Ah	Timer overflow	TAIFG	
0Ch	Reserved	–	
0Eh	Reserved	–	Lowest

† Timer1_A5 only

지금까지 배운 IRQ. sources

Table 3. Interrupt Sources, Flags, and Vectors of MSP430xG461x Configurations

INTERRUPT SOURCE	INTERRUPT FLAG	SYSTEM INTERRUPT	WORD ADDRESS	PRIORITY
Power-Up External Reset Watchdog Flash Memory	WDTIFG KEYV (see Note 1 and 5)	Reset	0FFFEh	31, highest
NMI Oscillator Fault Flash Memory Access Violation	NMIIFG (see Notes 1 and 3) OFIFG (see Notes 1 and 3) ACCVIFG (see Notes 1, 2, and 5)	(Non)maskable (Non)maskable (Non)maskable	0FFFCh	30
Timer_B7	TBCCR0 CCIFG0 (see Note 2)	Maskable	0FFFAh	29
Timer_B7	TBCCR1 CCIFG1 ... TBCCR6 CCIFG6, TBIFG (see Notes 1 and 2)	Maskable	0FFF8h	28
Comparator_A	CAIFG	Maskable	0FFF6h	27
Watchdog Timer+	WDTIFG	Maskable	0FFF4h	26
USCI_A0/USCI_B0 Receive	UCA0RXIFG, UCB0RXIFG (see Note 1)	Maskable	0FFF2h	25
USCI_A0/USCI_B0 Transmit	UCA0TXIFG, UCB0TXIFG (see Note 1)	Maskable	0FFF0h	24
ADC12	ADC12IFG (see Notes 1 and 2)	Maskable	0FFEEh	23
Timer_A3	TACCR0 CCIFG0 (see Note 2)	Maskable	0FFECh	22
Timer_A3	TACCR1 CCIFG1 and TACCR2 CCIFG2, TAIFG (see Notes 1 and 2)	Maskable	0FFEAh	21
I/O Port P1 (Eight Flags)	P1IFG.0 to P1IFG.7 (see Notes 1 and 2)	Maskable	0FFE8h	20
USART1 Receive	URXIFG1	Maskable	0FFE6h	19
USART1 Transmit	UTXIFG1	Maskable	0FFE4h	18
I/O Port P2 (Eight Flags)	P2IFG.0 to P2IFG.7 (see Notes 1 and 2)	Maskable	0FFE2h	17
Basic Timer1/RTC	BTIFG	Maskable	0FFE0h	16
DMA	DMA0IFG, DMA1IFG, DMA2IFG (see Notes 1 and 2)	Maskable	0FFDEh	15
DAC12	DAC12.0IFG, DAC12.1IFG (see Notes 1 and 2)	Maskable	0FFDCh	14
Reserved	Reserved (see Note 4)		0FFDAh	13
			...	...
			0FFC0h	0, lowest

NOTES: 1. Multiple source flags

FG4618 user's manual p.13