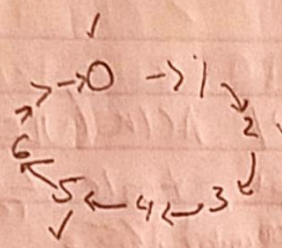
10/8 Timers

Timer_A Advanced

- using multiple channels

TAR - 3-bit overflow



- use continuous mode
  0 → 7 rollback to 0.
  us channel 1 (TACCR1) to get a period of 3 cycle

TACCR1 = 2 + 3 = 5 + 3 = 8 0

even though we have overflow the answer is still correct

0 + 3 = 3 + 3 = 6 + 3 = 1

((6+3) Mod 8 = 1

Channel 1: 20k
Channel 2: 30k

TAR: 0 ← 64k ← 16k ← 36k ← 60 + 20k = 80k ← 64k

TACCR1: 20k, 40k, 60k

TAR: 0 ← 26k ← 56k ← 22k

TACCR2: 30k, 60k

60k + 30k = 90k
90k − 64k = 26k

Ch. 0

TAR = TACCR0 CCIE/CCIFG → TACCTL0

Channel 1

TAR = TACCR1 CCIE/CCIFG → TACCTL1

E.9.

Run timer_A w/ ACLK (VLO. 12kHz)

channel 0 → interrupt, toggle LED each (9000 cycles) .75s red (P1.0)

Channel 1 → Green LED (P1.6)

.3s (3600 cycles)

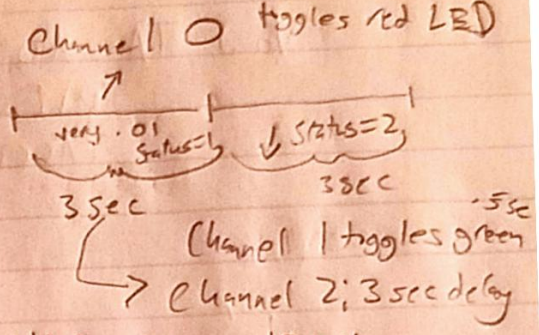
```
#Define Red BIT0
#Define Green BIT6
main...
  P1DIR = (Red | Green);
    //Channel 0
  TACCR0 = (9000-1);
  TACCTL0 |= CCIE;
  TACCTL0 &= ~CCIFG;
    //Channel 1
  TACCR1 = (3600-1);
  TACCTL1 |= CCIE;
  TACCTL1 &= ~CCIFG
    //Clock config
  TACTL = TASSEL_1 | ID_0 | MC_2 | TACLR;
  _low_power_mode_3(); //set GIE
```

∧ ISR

```
Void TA_A0_ISR(){
    P1Out ^= Red;
    TACCR0 += 9000 //schedule next interrupt
      // Flag cleared by hardware
}

Void TA_A1_ISR(){
    // Detect Channel 1
    if(TACCTL1 & CCIFG) != 0){
        P1OUT ^= green;
        TACCR1 += 3000;
        TACCTL1 &= ~CCIFG
```

Eg



.1sec x 12kHz
= 1200 cycyles Ch0

.5sec x 12kHz
= 6000 cycles Ch1

3 x 12kHz
= 36000 cycles Ch2

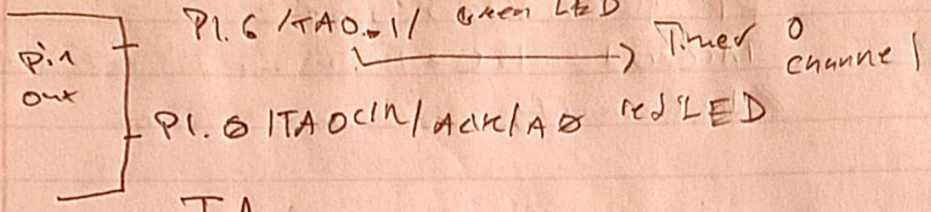
```
#define redLED BIT0  int status = 1;
#define green BIT6
: main()
    P1DIR |= (red | green);
        // channel 0
    TACCR0 = 1200 - 1
    TACCTL0 |= CCIE;
    TACCTL0 &= ~(CCIFG)
        // ch 2
    TACCR2 = 36000-1
    TACCTL2 |= CCIE
    TACCTL2 &= ~(CCIFG);

TACTL = TASSEL_1 | ID_0 | MC_2 | TACLR
_low_power_mode_3();
```

```
Void TA_A0_ISR(){
    P1Out ^= Red;
    TACCR0 += 1200
    //HW clears Flag
}

Void TA_A1_ISR(){
    //Detect 1
    if(TACCTL1 & CCIFG != 0){
        P1out ^= Green;
        TACCR1 += 6000;
        TACCTL1 &= ~CCIFG
    }
    if((TACCTL2 & CCIFG) != 0){
        if(status == 1){ //stop channel 0
            TACCTL0 &= ~CCIE;
            P1OUT &= ~Red; (TA0R)
            TACCR1 = TAR + 6000;
            TACCTL1 |= CCIE;
            TACCTL1 &= ~(CCIFG
            status = 2;
        }
(end of code outside IF)  if(status == 2){
//renew ch 2                TACCTL1 &= ~CCIE;
TACCR2 += 36000; P1out &= ~Green Led;
TACCTL &= ~(CCIFG);       TACCR0 = TA0R + 1200;
                          TACCTL0 |= CCIE;
                          TACCTL0 &= ~(CCIFG);
                          status = 1; }
                      }
```

Timer Output
Pin is multi-use



TAx.y
↓
Timer x Channel y

P1.6 as default

look in datasheet Tables at end Pg 49

Port 1 ---------- ↓ 8 bit
P1SEL0
P1SEL1
P1DIR

8 uses Per Pin

P1DIR = 1, P1SEL1 = 1, P1SEL2 = 0;

// Divert Pin to TA0.1
P1DIR |= BIT6;
P1SEL1 |= BIT6;
P1SEL2 &= ~(BIT6);

Patterns

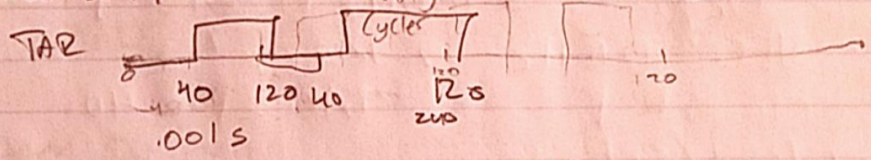
0: None
1: set
3: set/reset
4 Toggle
6: Toggle/set
7: reset/set

Eg. Channel 1 | up or continuous
set/reset
Channel 1 rollback to 0.
(TAR = TACCR1)

Generate a PWM on TA0.1
100Hz VLO 12kHz

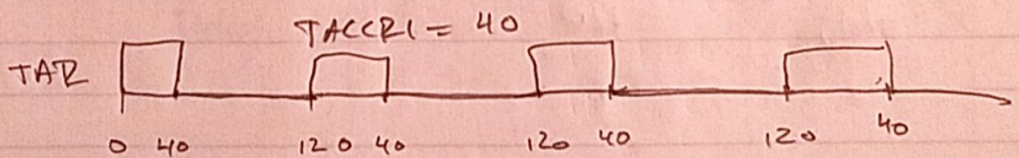
100Hz | s | 120 cycles
12000kHz
↓
UP mode

use set/reset 75% duty cycle TACCR1 = 40



$\frac{80}{120}$ = duty cycle = 66%

use reset/set



Duty cycle $\frac{40}{120}$ = 33%

```
//Divert pins (3 lines b4)
//timer
TACCR0 = 120;
TACTL = TASSEL_1 | ID_0 | MC_1 |
```

```
//Channel
TACCR1 = 40;
TACCTL1 = OUTMOD_7
//set/reset
```