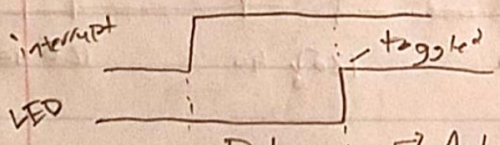
10/10

1) Board datasheet
LEDs P1.0 (red)
P1.6 (Green)

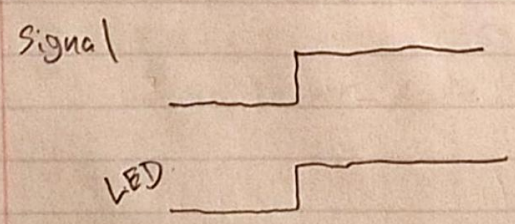
2) Chip datasheet Pg(49)
PinOut P1.6/TA0.1 Timer #0 Channel 1
P1.0 → ✗ no timer

P 49 ↳ Divert the Pin
P1DIR = 1
P1SEL1 = 1
P1SEL2 = 0

Timer Output
toggle Via interrupt



Delay → Activate Clock, CPU, Save PC, SR
find & launch ISR



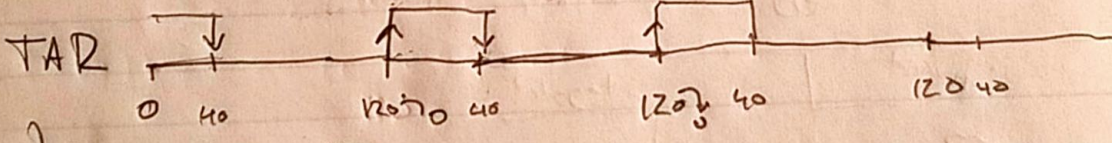
Timer output using Channel 1
3: set/reset
7: reset/set

Channel 1
(TAR = TACCR1)

rollback to 0
(UP, continuous)

PWM to do LED brightness

Frequency = 100Hz (0.01 sec)
VLO: 12kHz → 120 cycles ← Period
TACCR1 = 40 ← (0-120) more brightness
Channel 1
→ reset/set



duty cycle = $\frac{40}{120}$

Eg Timer 0_A, UPmode, VLO 12kHz, reset/set (#7)
TACCR1 [0-120] range = 20 initially
Timer 1_A, upmode (1 sec interval w/ interrupt) (12000 cycles)
=7 Cycle TACCR1 (20-40-60-80)

```
#define LED P1.6
// Divert Pin to TA0.1
P1DIR |= LED;
P1SEL1 |= LED;
P1SEL2 &= ~(LED);

//Timer #0
TACCR1 = 120;
TACTL = TASSEL_1|ID_0|MC_1|TACLR;
//output mode #7 on Channel 1
TACCR1 = 20;                                  } PMW
TACCTL1 |= OUTMOD_7; //.set/reset
// Timer #1
TA1CCR0 = (12000-1);
TA1CCTL0 |= CCIE;
TA1CCTL0 &= CCIFG;
TA1CTL = TASSEL_1|ID_0|MC_1|TACLR;
__low_power_mode_3();
```

ISR on Next Page

```
Void TA1_A0_ISR(){
    if(TACCR1 >= 80)
        TACCR1 = 20;
    else
        TACCR1 += 20;
    // hardware Clears Flag
}
```

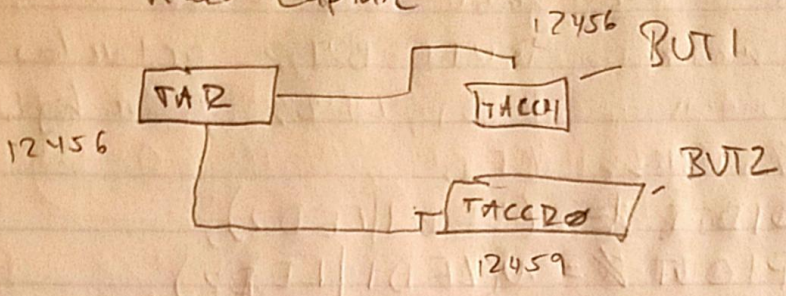
TV Show Quiz game

Port 1 { mask: BUT1 BUT2 active low
mask: LED1, LED2 active high

```
//initialize LED
P1DIR |= (LED1|LED2);
P1OUT &= ~(LED1|LED2);
// Buttons
P1DIR &= ~(BUT1|BUT2);
P1REN |= (          );
P1IES |= (          );
P1IE  |= (          );
P1IFG &= ~(         );
__low_power_mode_4(); __enable_interrupt();
                      for(;;){}
```

```
Void Port1_ISR(){
    //Detect button 1:
    if (P1IFG & BUT1) != 0){
        P1OUT |= LED1;
        P1IFG &= ~BUT1 }
    if((P1IFG & BUT2) != 0){
        P1OUT |= LED2;
        P1IFG &= ~BUT2;
    }
}
```

Timer Capture



low-power mode overriding feature
- Available in some Advanced MSP430 chips
- prevent unintentional shutoff of clock needed.
- If LPM shutsdown a clock and a module requests it, this clock will activate as long as the module uses it.

9 18

LCD

LCDM0 - LCDM7 : address array

```
// test
uint_8t * ptr = & LCDM0;
int i;
for(i = 0; i < 8; i++){
    ptr[i] = 0xFF;
}
```

Display 16 Bit assigned up to 65535 → 5 digits

```
void display_Uint16 (uint16_t n){
    Uint_8t *ptr = & LCDM0
    do{
        digit = n % 10;
        Ptr[i] = digit
        n = n / 10;
        i++;
    } while (n != 0);
    while (i <= 7){
        PTr[i] = 0;
        i++;
    }
```