

CS/ECE 5780/6780: Embedded System Design

John Regehr

Lecture 13: Input Capture

Basic Ideas of Input Capture

Triggers interrupts on rising or falling edges of external signals.
Can also measure the period or pulse width of TTL-level signals.

Each input capture module has:

- An external input pin, IC_n

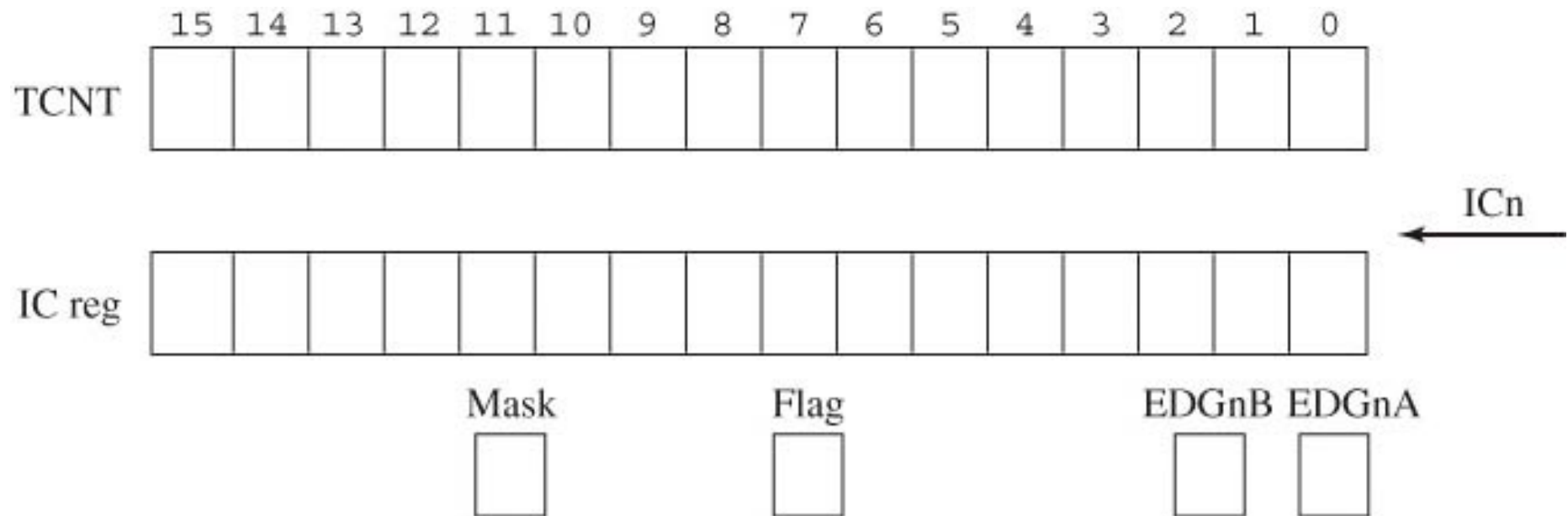
- A flag bit

- Two edge control bits, EDG_nB and EDG_nA

- An interrupt mask bit (arm)

- A 16-bit input capture register

Basic Components of Input Capture



Basic Ideas of Input Capture (cont)

Hardware detects capture events

Two or three actions result from a capture event:

- Current TCNT copied into input capture register.

- The input capture flag is set.

- An interrupt is requested if the mask is 1.

The input capture mechanism has many uses:

- Detect external signals

- Perform rising edge captures and subtract consecutive captures to obtain the period

- Perform a rising edge capture, then a falling edge capture, and subtract to obtain the pulse width

Control Bits and Flags

Input captures are on port T (i.e., **PTT**).

Set pin to input capture mode by setting bit to 0 in **TIOS**.

Input capture registers are **TC0**, ..., **TC7**.

Arm interrupts using **TIE**.

Flags are found in **TFLG1**.

Set edge to trigger on using **TCTL3** and **TCTL4**.

EDGnB	EDGnA	Active edge
0	0	None
0	1	Capture on rising
1	0	Capture on falling
1	1	Capture on both rising and falling

Setting the TFLG1 Register

Care must be taken when clearing the TFLG1 register.

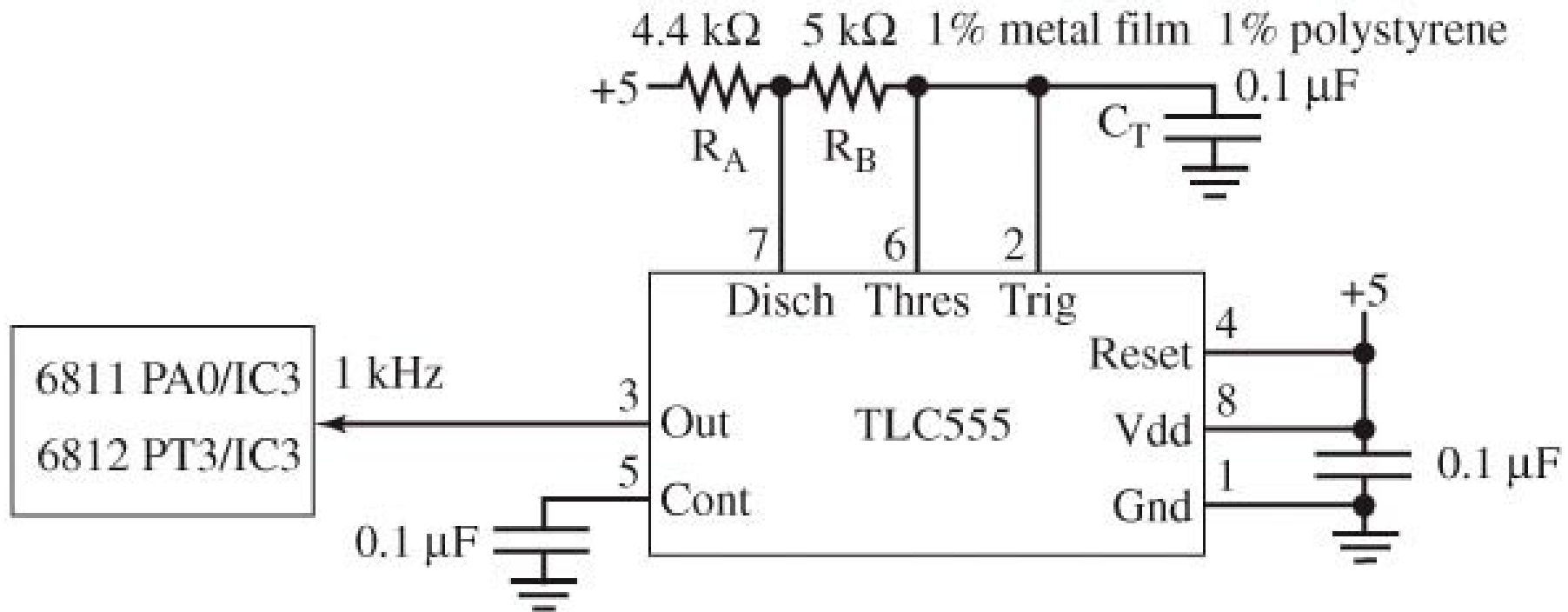
The following works:

```
TFLG1 = 0x01;    ldy  #$1000
                  ldaa  #$01
                  staa  $23,Y
```

The following does not:

```
TFLG1 |= 0x01;   ldx  #$1000
                  bset  $23,X,$01
```

Real Time Interrupt Using an Input Capture



Component	6812
Longest instruction (cycles, μs)	13 = 3.25 μs
Process the interrupt (cycles, μs)	9 = 2.25 μs
Execute the handler (cycles, μs)	11 = 2.75 μs
Max latency (μs)	8.25 μs

Periodic Interrupt Using Input Capture

```
unsigned short Time;           // incremented
void Init(void){
    asm sei                     // make atomic
    TIOS &=~0x08;              // PT3 input capture
    DDRT &=~0x08;              // PT3 is input
    TSCR1 = 0x80;              // enable TCNT
    TSCR2 = 0x01;              // 500ns clock
    TCTL4 = (TCTL4&0x3F)|0x40;
    TIE |= 0x08;               // Arm IC3, rising
    TFLG1 = 0x08;              // initially clear
    Time = 0;
    asm cli }
void interrupt 11 IC3Han(void){
    TFLG1 = 0x08;               // acknowledge
    Time++; }
}
```

Period Measurement

Resolution of a period measurement is the smallest change in period that can be detected.

Resolution of TCNT is from 250ns to 32 μ s (4 MHz E Clock).

Resolution is also the units of measurement.

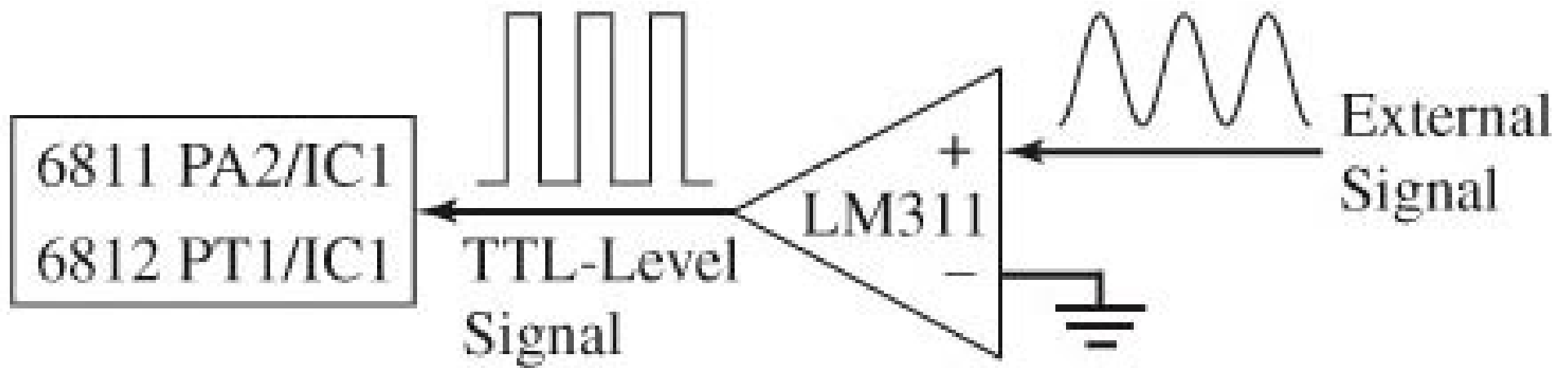
Precision is the number of separate and distinguishable measurements.

Precision of TCNT is 65,536 different periods (16-bit).

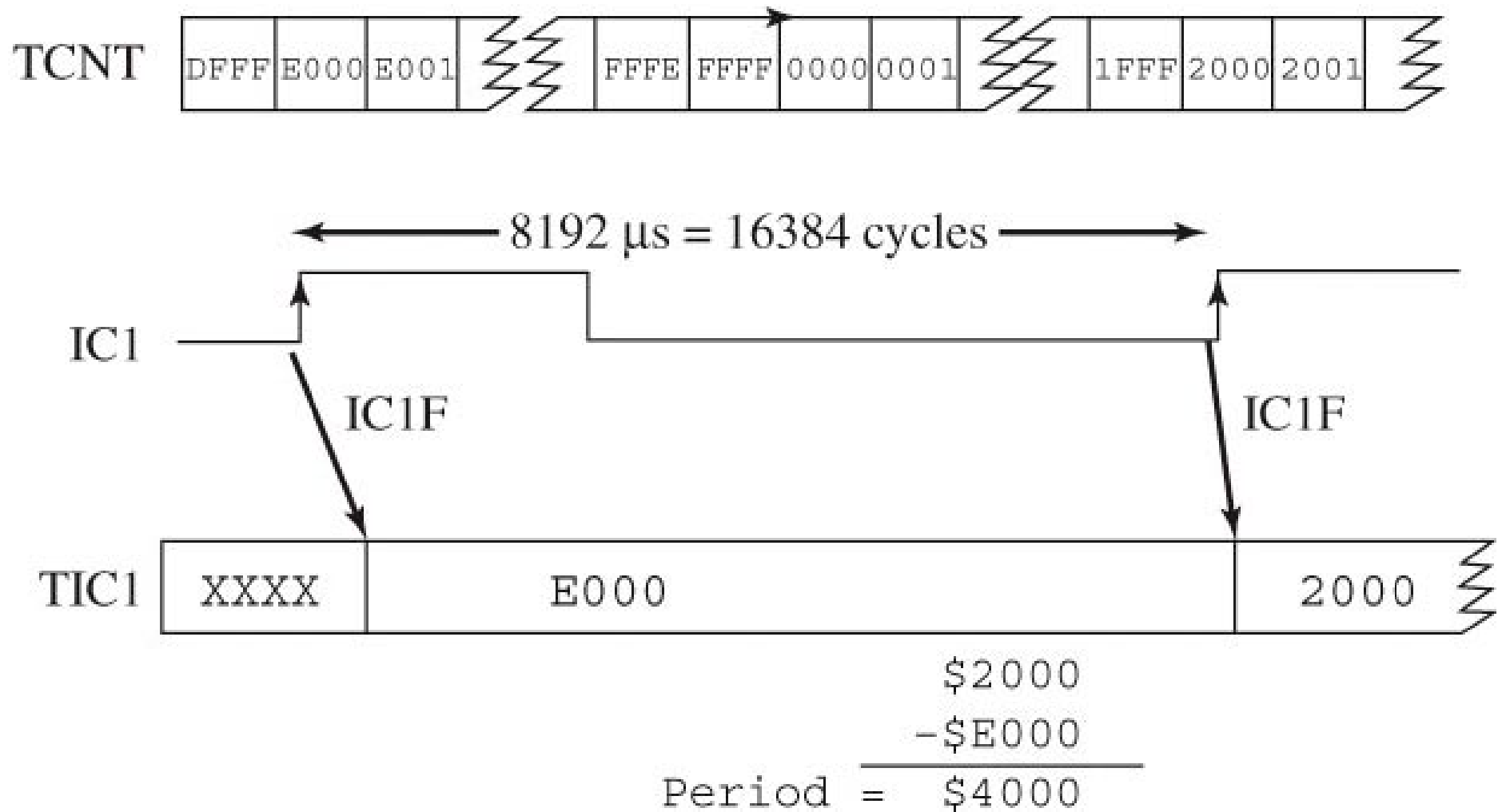
Range is min and max values that can be measured.

Good measurement systems should detect under and overflows, and when there is no period.

Period Measurement



Period Measurement Example



Period Measurement Resolution

Component	6812
Process the interrupt (cycles, μs)	9 = $2.25 \mu s$
Execute the entire handler (cycles, μs)	31 = $7.75 \mu s$
Minimum period (cycles, μs)	40 = $10 \mu s$

Period (μs)	Cycles/interrupt	Time in handler (%)
10	40	100
20	40	50
100	40	10
P	40	1000/P

Initialization for Period Measurement

```
unsigned short Period;           // 500 ns units
unsigned short First;           // TCNT first edge
unsigned char Done;             // Set each rising
void Init(void){
    asm sei                      // make atomic
    TIOS &=~0x02;               // PT1 input capture
    DDRT &=~0x02;              // PT1 is input
    TSCR1 = 0x80;              // enable TCNT
    TSCR2 = 0x01;              // 500ns clock
    TCTL4 = (TCTL4&0xF3)|0x04; // rising
    First = TCNT;              // first will be wrong
    Done = 0;                  // set on subsequent
    TFLG1 = 0x02;              // Clear C1F
    TIE |= 0x02;               // Arm IC1
    asm cli }
}
```

ISR for Period Measurement

```
void interrupt 9 TC1handler(void){
    Period = TC1-First; // 500ns resolution
    First = TC1;        // Setup for next
    TFLG1 = 0x02;      // ack by clearing C1F
    Done = 0xFF;
}
```

32-bit Period Measurement

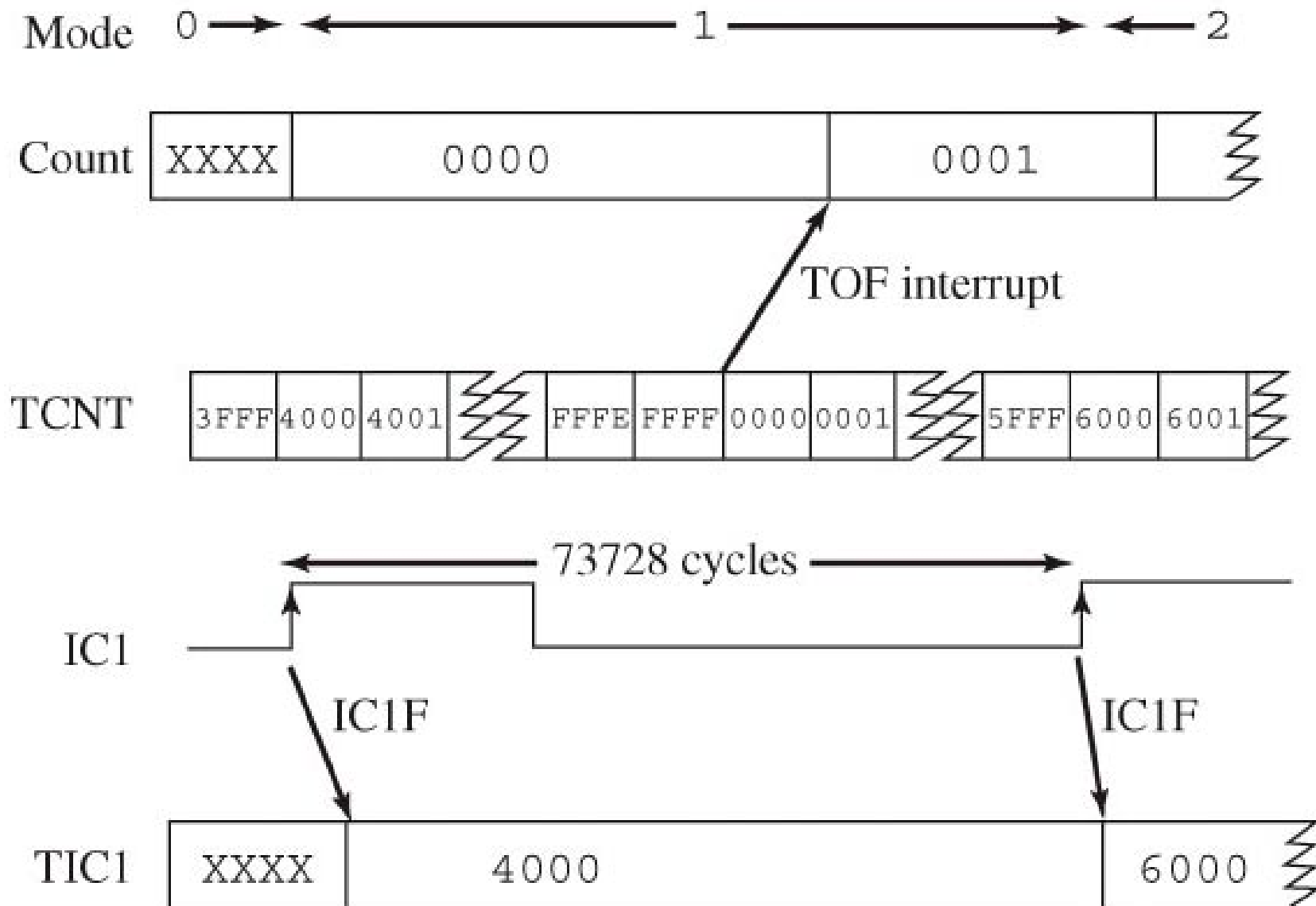
Every time TCNT register overflows from \$FFFF to 0, the TOF flag is set.

Can increase precision to 32-bits by counting the number of TOF flag setting events during one period (Count).

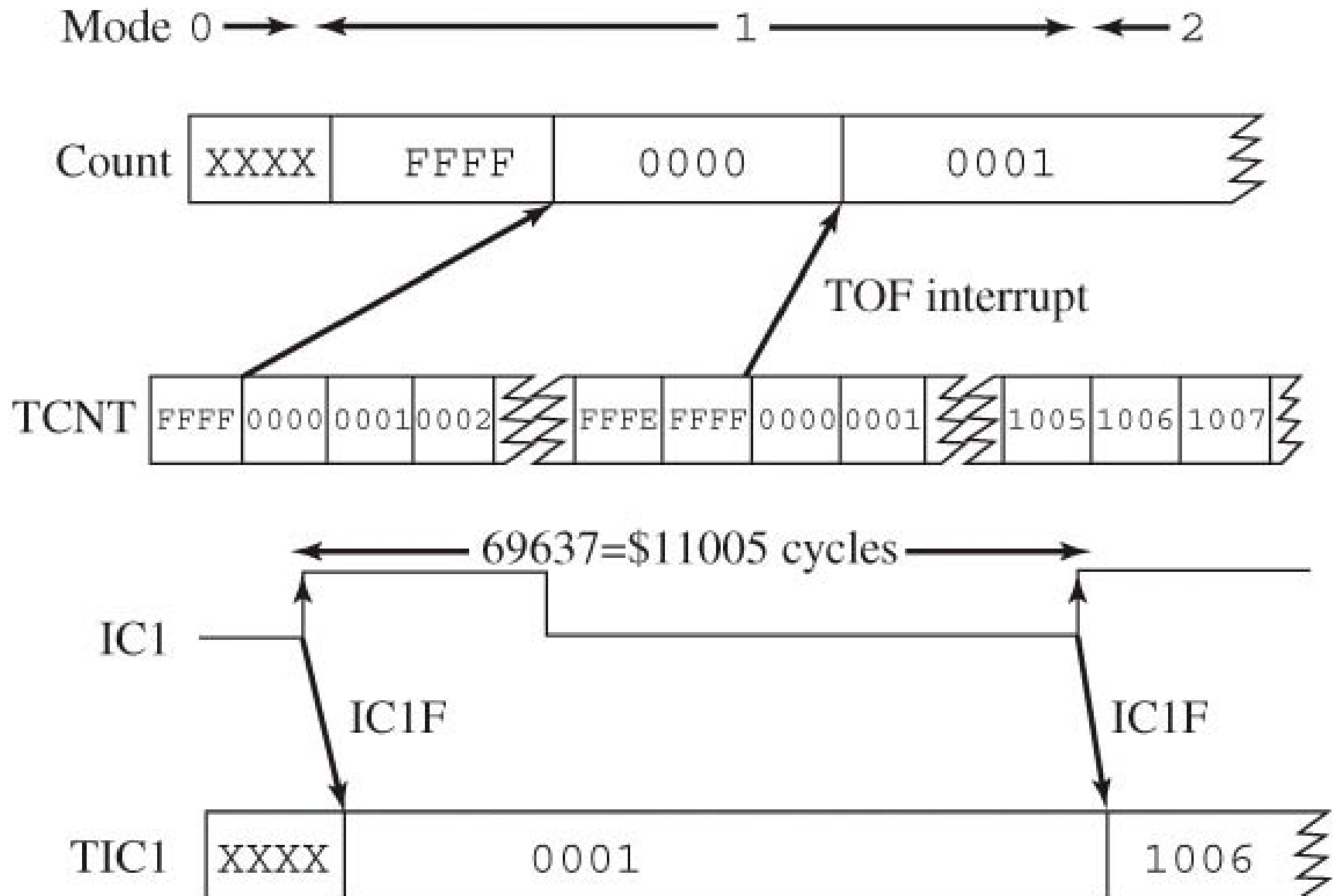
To do this, arm both input capture and timer overflow interrupts.

For each timing measurement, high 16-bits are value of Count, and low 16-bits are value in input capture register.

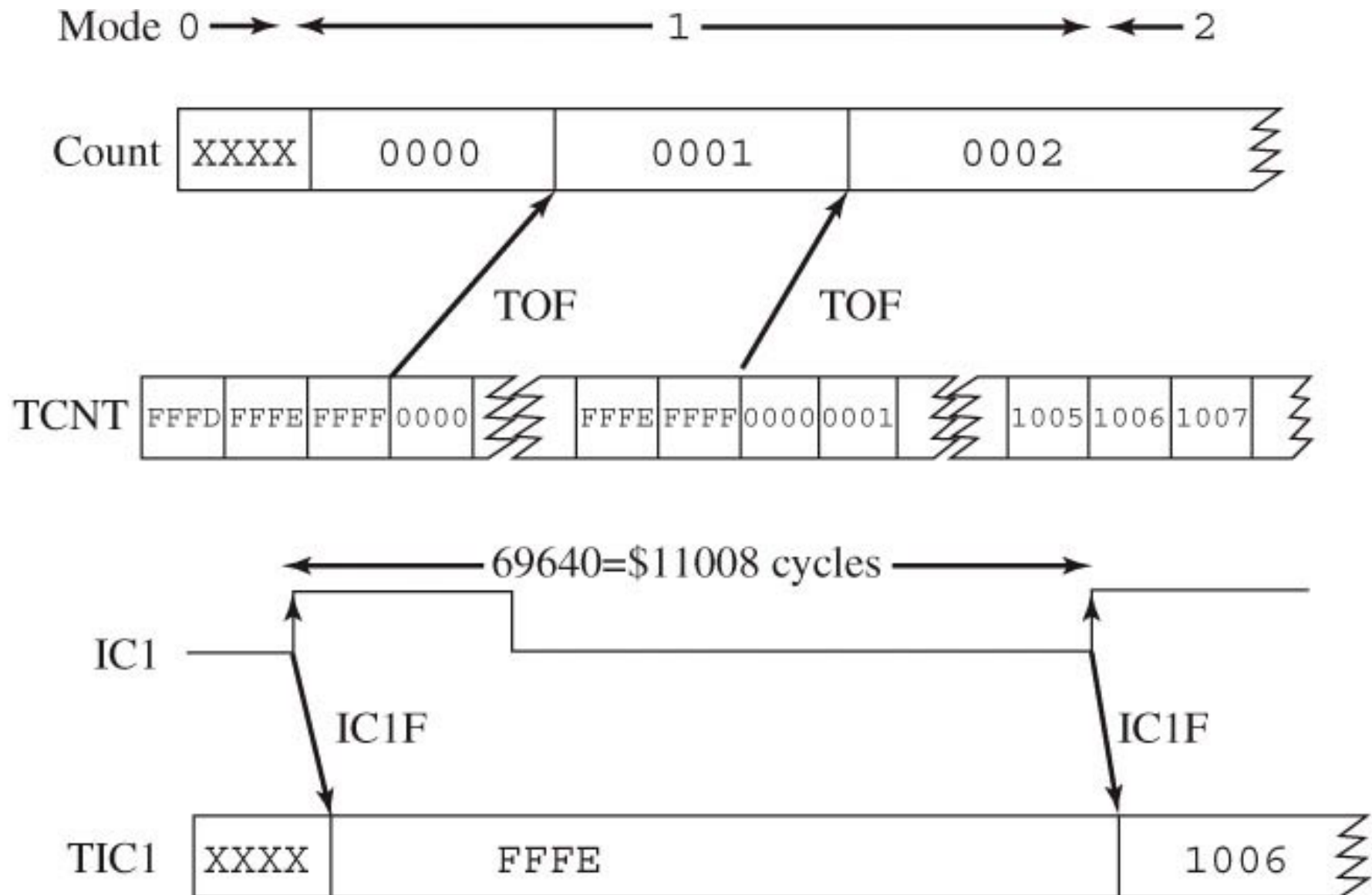
Simple Illustration of 32-bit Period Measurement



TOF Set Just Before IC1F Flag



TOF Set Just After IC1F Flag



Initialization for 32-Bit Period Measurement

```
unsigned short MsPeriod, LsPeriod;
unsigned short First;
unsigned short Count;
unsigned char Mode;
void Init(void){
    asm sei // make atomic
    TIOS &=~0x02; // PT1 input capture
    DDRT &=~0x02; // PT1 is input
    TSCR2 = 0x81 ; // Arm, TOF 30.517Hz
    TSCR1 = 0x80; // enable counter
    TFLG1 = 0x02; // Clear C1F
    TIE |= 0x02; // Arm IC1, C1I=1
    TCTL4 = (TCTL4&0xF3)|0x04; // rising
    TFLG2 = 0x80; // Clear TOF
    Mode = 0; // searching for first
    asm cli }
```

Input Capture ISR for Period Measurement

```
void interrupt 9 TIC1handler(void){
    if(Mode==0){                // first edge
        First = TC1; Count=0;
        Mode=1;
        if(((TC1&0x8000)==0)&&(TFLG2&0x80)) Count--;
    } else {                    // second edge
        if(((TC1&0x8000)==0)&&(TFLG2&0x80)) Count++;
        Mode = 2;              // measurement done
        MsPeriod = Count;
        LsPeriod = TC1-First;
        if(TC1<First){
            MsPeriod--;        // borrow
        }
        TIE=0x00; TSCR2=0x00; } // Disarm
    }
    TFLG1 = 0x02; }           // ack, clear C1F
```

Timer Overflow ISR for Period Measurement

```
void interrupt 16 T0handler(void){
    TFLG2 = 0x80; // ack
    Count++;
    if(Count==65535){ // 35 minutes
        MsPeriod=LsPeriod=65535;
        TIE=0x00; TSCR2=0x00; // Disarm
        Mode = 2; // done
    }
}
```

Input Capture Summary

HW captures the time an event occurs

Can be much more precise than reading a clock register from SW, even in an interrupt handler