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, ICn
- A flag bit
- Two edge control bits, EDGnB and EDGnA
- An interrupt mask bit (arm)
- A 16-bit input capture register
Basic Components of Input Capture
Hardware detects capture events

Two or three actions result from a capture event:

1. Current TCNT copied into input capture register.
2. The input capture flag is set.
3. An interrupt is requested if the mask is 1.

The input capture mechanism has many uses:

1. Detect external signals
2. Perform rising edge captures and subtract consecutive captures to obtain the period
3. 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.

<table>
<thead>
<tr>
<th>EDGnB</th>
<th>EDGnA</th>
<th>Active edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Capture on rising</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Capture on falling</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Capture on both rising and falling</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Component</th>
<th>6812</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longest instruction (cycles, μs)</td>
<td>13=3.25μs</td>
</tr>
<tr>
<td>Process the interrupt (cycles, μs)</td>
<td>9=2.25μs</td>
</tr>
<tr>
<td>Execute the handler (cycles, μs)</td>
<td>11=2.75μs</td>
</tr>
<tr>
<td>Max latency (μs)</td>
<td>8.25μs</td>
</tr>
</tbody>
</table>
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++; }
Resolution of a period measurement is the smallest change in period that can be detected.

Resolution of TCNT is from 250ns to 32\(\mu\)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

TCNT:
- DFFF E000 E001
- FFFE FFFF 0000 0001
- 1FFF 2000 2001

- 8192 μs = 16384 cycles

IC1

IC1F

TIC1:
- XXXX
- E000
- 2000

$2000

$E000

Period = $4000
## Period Measurement Resolution

<table>
<thead>
<tr>
<th>Component</th>
<th>6812</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process the interrupt (cycles, $\mu$s)</td>
<td>$9 = 2.25 \mu s$</td>
</tr>
<tr>
<td>Execute the entire handler (cycles, $\mu$s)</td>
<td>$31 = 7.75 \mu s$</td>
</tr>
<tr>
<td>Minimum period (cycles, $\mu$s)</td>
<td>$40 = 10 \mu s$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Period ($\mu$s)</th>
<th>Cycles/interrupt</th>
<th>Time in handler (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>40</td>
<td>100</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>100</td>
<td>40</td>
<td>10</td>
</tr>
<tr>
<td>$P$</td>
<td>40</td>
<td>$1000/P$</td>
</tr>
</tbody>
</table>
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

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

Mode: 0 → 1 → 2

Count: XXXX 0000 0001

TCNT: 3FFF 4000 4001 FFFE FFFF 0000 0001 5FFF 6000 6001

TOF interrupt

IC1: 73728 cycles

IC1F

TIC1: XXXX 4000 6000
TOF Set Just Before IC1F Flag
TOF Set Just After IC1F Flag

Mode 0 → 1 → 2

Count

XXXX 0000 0001 0002

TOF

TOF

TCNT

FFFF FFFE FFFF 0000 0000 0001 1005 1006 1007

69640 = 11008 cycles

IC1

IC1F

IC1F

TIC1

XXXX FFFE 1006
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 }

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
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
    }
}

HW captures the time an event occurs
Can be much more precise than reading a clock register from SW, even in an interrupt handler