Administrivia

- 6780 project proposal feedback.
- Midterm online course evaluation.
- Midterm study packet.
Basic Principles 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

![Diagram of input capture module components](image-url)
Basic Principles of Input Capture (cont)

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:
- Arm the flag bit so that an interrupt is requested on the active edge of an external signal.
- 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.

<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>
Setting the TFLG1 Register

- Care must be taken when clearing the TFLG1 register.
- The following works:
  \[
  \text{TFLG1} = 0x01; \quad \text{ldy} \quad \#\$1000
  \]
  \[
  \text{ldaa} \quad \#\$01
  \]
  \[
  \text{staa} \quad \$23,Y
  \]
- The following does not:
  \[
  \text{TFLG1} \leftarrow 0x01; \quad \text{ldx} \quad \#\$1000
  \]
  \[
  \text{bset} \quad \$23,X,\$01
  \]

Real Time Interrupt Using an Input Capture

<table>
<thead>
<tr>
<th>Component</th>
<th>6812</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longest instruction (cycles, $\mu$s)</td>
<td>13=3.25$\mu$s</td>
</tr>
<tr>
<td>Process the interrupt (cycles, $\mu$s)</td>
<td>9=2.25$\mu$s</td>
</tr>
<tr>
<td>Execute the handler (cycles, $\mu$s)</td>
<td>11=2.75$\mu$s</td>
</tr>
<tr>
<td>Max latency ($\mu$s)</td>
<td>8.25$\mu$s</td>
</tr>
</tbody>
</table>
Periodic Interrupt Using Input Capture

```c
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 32µs (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 Example

TCNT

\[ \text{DFFFFE000E001} \quad \text{FFFFFFFF0000001} \quad \text{1FFFF20002001} \]

- 8192 µs = 16384 cycles

\[ \text{TIC1} \quad \text{IC1} \quad \text{IC1F} \quad \text{IC1F} \]

\[ \text{XXXX} \quad \text{E000} \quad \text{2000} \]

\[ \begin{align*}
$2000 \\
- $E000 \\
& \text{Period} = \frac{-$E000}{4000}
\end{align*} \]
### 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

```c
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;
}
```

32-bit Period Measurement

- Every time TCNT register overflows from $FFFFFF$ 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
Initialization for 32-Bit Period Measurement

```c
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
    TCL4 = (TCTL4&0xF3)|0x04; // rising
    TFLG2 = 0x80;           // Clear TOF
    Mode = 0;               // searching for first
    asm cli }
```
Input Capture ISR for Period Measurement

```c
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

```c
void interrupt 16 TOhandler(void){
    TFLG2 = 0x80; // ack
    Count++;
    if(Count==65535){ // 35 minutes
        MsPeriod=LsPeriod=65535;
        TIE=0x00; TSCR2=0x00; // Disarm
        Mode = 2; // done
    }
}
```
Measure Resistance Using Pulse Width

\[ T(\text{sec}) = 0.45 \times (R_1 + R) \times C \] where \( R_1, R \) are in \( \Omega \) and \( C \) is in \( F \)

Gadfly Pulse-Width Measurement

```c
void Init(void){
    DDRB |= 0x80; // PB7 is output
    TIOS &=~0x04; // clear bit 2
    DDRT &=~0x04; // PT2 is input capture
    TSCR1 =0x80;  // enable
    TSCR2 =0x01;  // 500 ns clock
    TIE = 0x00;} // no interrupts
```
```c
unsigned short Measure(void) {
    unsigned short Rising;
    TCTL4 = (TCTL4&0xCF)|0x10; // Rising
    TFLG1 = 0x04; // clear C2F
    PORTB|=~0x80;
    PORTB|= 0x80; // rising edge on PB7
    while(TFLG1&0x04==0){}; // wait for rise
    Rising = TC2; // TCNT at rising edge
    TFLG1 = 0x04; // clear C2F
    TCTL4 = (TCTL4&0xCF)|0x20; // Falling
    while(TFLG1&0x04==0){}; // wait for fall
    return(TC2-Rising-1000); }
```

Interrupt-Driven Pulse-Width Measurement

![Diagram of pulse width measurement](image)
Pulse-Width Measurement Using Interrupts

unsigned short PW;       // units of 500 ns
unsigned short Rising;   // TCNT at rising
unsigned char Done;      // Set each falling

void Init(void) {
    asm sei            // make atomic
    TIOS &=~0x02;      // clear bit 1
    DDRT &=~0x02;      // PT1 is input capture
    TSCR1 =0x80;       // enable
    TSCR2 =0x01;       // 500 ns clock
    TCTL4|=0x0C;       // Both edges IC1
    TIE |= 0x02;       // arm IC1
    TFLG1 = 0x02;      // clear C1F
    Done = 0;
    asm cli
}

void interrupt 9 TC1handler(void){
    if(PTT&0x02){       // PT1=1 if rising
        Rising = TC1;   // Setup for next
    } else{
        PW = TC1-Rising; // measurement
        Done = 0xFF;
    }
    TFLG1 = 0x02;       // ack, clear C1F
}
unsigned short PW;     // units of 500 ns
unsigned char Done;    // Set each falling
void Init(void) {
    asm sei            // make atomic
    TIOS &="0x06;      // clear bits 2,1
    DDRT &="0x06;      // PT2,PT1 input captures
    TSCR1 = 0x80;      // enable
    TSCR2 = 0x01;      // 500 ns clock
    TCTL4 =(TCTL4&0xCF)|0x10; // IC2 Rise
    TCTL4 =(TCTL4&0xF3)|0x08; // IC1 Fall
    Done = 0;         // set on the falling edge
    TIE |= 0x02;      // arm IC1, not IC2
    TFLG1 = 0x02;     // clear C1F
    asm cli
}
void interrupt 9 TIC1handler(void){
    TFLG1 = 0x02;  // ack C1F
    PW = TC1-TC2;  // from rise to fall
    Done = 0xFF;
}