Basic Principles of Input Capture

- *Input capture* can be used to measure the period or pulse width of TTL-level signals.
- Can also trigger interrupts on rising or falling transitions of external 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 Principles of Input Capture (cont)

- 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. Arm the flag bit so that an interrupt is requested on the active edge of an external signal.
  2. Perform two rising edge captures and subtract the measurement to get the period.
  3. Perform a rising edge capture, then a falling edge capture, and subtract to get pulse width.
Input Capture Interface on 68HC11

<table>
<thead>
<tr>
<th>TCNT</th>
<th>TIC1</th>
<th>TIC2</th>
<th>TIC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

6811 input capture

Control Bits and Flags

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

TFLG1

<table>
<thead>
<tr>
<th>TFLG1</th>
<th>TCF</th>
<th>ICIF</th>
<th>ICIE</th>
<th>PA2</th>
<th>PA1</th>
<th>PA0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Setting the TFLG1 Register

- Care must be taken when clearing the TFLG1 register.
- The following works:
  - `ldy #$1000 TFLG1 = 0x01;`
  - `ldaa #$01`
  - `staa $23,Y`
- The following does not:
  - `ldx #$1000 TFLG1 |= 0x01;`
  - `bset $23,X,$01`
### Real Time Interrupt Using an Input Capture

<table>
<thead>
<tr>
<th>Component</th>
<th>6811</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longest instruction (cycles, μs)</td>
<td>41=20.5μs</td>
</tr>
<tr>
<td>Process the interrupt (cycles, μs)</td>
<td>14=17μs</td>
</tr>
<tr>
<td>Execute the handler (cycles, μs)</td>
<td>28=14μs</td>
</tr>
<tr>
<td>Max latency (μs)</td>
<td>41.5μs</td>
</tr>
</tbody>
</table>

### Periodic Interrupt Using Input Capture

```c
unsigned int TIME; // incremented
void Init(void){
    asm("sei"); // make atomic
    TCTL2 = (TCTL2&0xFC)|0x01;
    TMSK1|=0x01; // Arm IC3
    TIME=0; // initially clear
    asm("cli");
    #pragma interrupt_handler IC3Han
    void IC3Han(void){
        if(TFLG1|=0x01){asm("swi");
            TFLG1=0x01; // acknowledge
        }
    }
}
```

### Init for Periodic Interrupt Using Input Capture

```c
TIME rmb 2 ;every 1 ms
Init sei ;make atomic
ldaa TCTL2 ;0ld value
anda #$FC ;Clear EDG3B=0
oraa #$01 ;EDG3BA =01
staa TCTL2 ;on rise of PA0
ldaa TMSK1 ;0ld value
oraa #$01 ;IC3I=1
staa TMSK1 ;Arm IC3F
ldd #0
std TIME ;init global
ldaa #$01 ;clear IC3F
staa TFLG1
cli ;enable
rts
```

### Periodic Interrupt Using Input Capture

```c
IC3Han ldaa TFLG1 ;is XX XXXX1 [4]
anda #$01 [2]
bne ClkHan [3]
swi
ClkHan ldaa #$01 ;clear IC3F [2]
staa TFLG1 ;Acknowledge [4]
ldx TIME [5]
inx [3]
stx TIME [5]
rti
org $FFEA
fdb IC3Han
```
Period Measurement

- Resolution of a period measurement is the smallest change in period that can be detected.
  - Resolution of TCNT is from 500ns to 4μs.
- 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.

Resolution of TCNT is from 500ns to 4μs.

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

<table>
<thead>
<tr>
<th>Component</th>
<th>6811</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process the interrupt (cycles,μs)</td>
<td>14=7μs</td>
</tr>
<tr>
<td>Execute the entire handler (cycles,μs)</td>
<td>50=25μs</td>
</tr>
<tr>
<td>Minimum period (cycles,μs)</td>
<td>64=32μs</td>
</tr>
<tr>
<td>Period (μs)</td>
<td>Cycles/interrupt</td>
</tr>
<tr>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>320</td>
<td>64</td>
</tr>
<tr>
<td>P</td>
<td>64</td>
</tr>
</tbody>
</table>

Initialization for Period Measurement in C

```c
unsigned int Period; // units of 500 ns
unsigned int First; // TCNT first edge
unsigned char Done; // Set each rising

void Ritual(void){
    asm("sei"); // make atomic
    TCTL2 = (TCTL2&0xCF)|0x10; // rising
    First = TCNT; // first will be wrong
    Done=0; // set on subsequent
    TFLG1 = 0x04; // Clear IC1F
    TMSK1 |= 0x04; // Arm IC1
    asm("cli");}
```
Initialization for Period Measurement

Period rmb 2 ;units 500 ns
First rmb 2 ;TCNT at first edge
Done rmb 1 ;set each rising
Init sei ;make atomic
ldaa TCTL2 ;Old value
anda #$CF ;Clear EDG1B=0
oraa #$10 ;EDG1BA =01
staa TCTL2 ;on rise of PA2
1dd TCNT
std First ;init global
clr Done

ISR for Period Measurement (cont)

ldaa #$04 ;clear IC1F
staa TFLG1
ldaa TMSK1 ;Old value
oraa #$04 ;IC1I=1
staa TMSK1 ;Arm IC1F
cli ;enable
rts

ISR for Period Measurement

IC1Han ld aa  #$01 ;clear IC3F [2]
staa TFLG1 ;Acknowledg [4]
1dd TIC1 [5]
subd First [6]
std Period [5]
1dd TIC1 [5]
std First [5]
ldaa #$FF ;set flag [2]
staa Done [4]
rti [12]
org $FFEE
fdb IC1Han

ISR for Period Measurement in C

#pragma interrupt_handler TIC1handler()
void TIC1handler(void){
  Period=TIC1-First;
  First=TIC1; // Setup for next
  TFLG1=0x04; // ack by clearing IC1F
  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

```c
unsigned int MsPeriod, LsPeriod;
unsigned int First;
unsigned int Count;
unsigned char Mode;

void Ritual(void){
  asm("sei"); // make atomic
  TFLG1 = 0x04; // Clear IC1F
  TMSK1 |= 0x04; // Arm IC1
  TCTL2 = (TCTL2&0x0C)|0x10; // rising
  TFLG2 = 0x80; // Clear TOF
  TMSK2 |= 0x80; // Arm TOF
  Mode=0;
  asm("cli"); }
```

Input Capture ISR for Period Measurement

```c
#pragma interrupt_handler TIC1handler()
void TIC1handler(void){
  if(Mode==0){
    First = TIC1; Count=0; Mode=1;
    if(((TIC1&0x8000)==0)
      &&(TFLG2&0x80)) Count--;
  } else {
    if(((TIC1&0x8000)==0)
      &&(TFLG2&0x80)) Count++;
    MsPeriod=Count; Mode=2;
    LsPeriod=TIC1-First;
    if (TIC1<First) MsPeriod--;
    TMSK1=0x00; TMSK2=0x00; // Disarm
    TFLG1=0x04; } // ack, clear IC1F
```

Timer Overflow ISR for 32-Bit Period Measurement

```c
#pragma interrupt_handler TOhandler()
void TOhandler(void){
  TFLG2=0x80;
  Count++;
  if(Count==65535){ // 35 minutes
    MsPeriod=LsPeriod=65535;
    TMSK1=0x00;
    TMSK2=0x00; // Disarm
    Mode=2; }
```

Measure Resistance Using Pulse Width
Gadfly Pulse-Width Measurement in C

```c
void Init(void){
    TMSK1=0x00; // no interrupts
    unsigned int Measure(void) {
    unsigned int Rising;
    TCTL2=(TCTL2&0xF3)|0x4; // Rising edge
    TFLG1=0x02; // clear IC2F
    PORTB&=0x7F; PORTB|=0x80; // rising edge on PB7
    while(TFLG1&0x02==0){} // wait rise
    Rising=TIC2; // TCNT at rising edge
    TCTL2=(TCTL2&0xF3)|0x8; // Falling edge
    TFLG1=0x02; // clear IC2F
    while(TFLG1&0x02==0){} // wait fall
    return(TIC2-Rising-1000); }
}
```

Initialization for Gadfly Pulse-Width Measurement

```assembly
; B=PB7, Q=PA1/IC2
Init  ldx #$1000 ;I/O registers
     ldax #$00 ;gadfly
     staa $22,X ;TMSK1 IC2I=0
     rts
```

Gadfly Pulse-Width Measurement in Assembly

```assembly
; return Reg D as R in Kohm
Rising equ 0 ;First TCNT
Meas  ldx #$1000 ;I/O registers
     ldax #$04 ;Rising edge
     staa $21,X ;Set TCTL2
     bcrl $23,X,$FD ;IC2F=0
     bcrl $04,X,$80 ;PB7=0
     bset $04,X,$80 ;PB7=1
First  brcl $23,X,$02,First
; Wait for first rising edge
     ldy $12,X ;TCNT at rising
     ldax #$08 ;Falling edge
     staa $21,X ;Set TCTL2
```

Gadfly Pulse-Width Measurement in Assembly (cont)

```assembly
     bcrl $23,X,$FD ;IC2F=0
     pshy ;Save on stack
Second  brcl $23,X,$02,Second
; Wait for next falling edge
     ldd $12,X ;TCNT at falling
     tsy
     subd Rising,Y
; RegD=pulse width 1000 to 2000 cyc
     subd #$1000 ;0<=R<=1000Kohm
     puly
     rts
```
Interrupt-Driven Pulse-Width Measurement

Slide 33

Pulse-Width Measurement Using Interrupts

```c
#define PA2 0x04 // the input signal
#pragma interrupt_handler TIC1handler()
void TIC1handler(void){
  if(PORTA&PA2){ // PA2=1 if rising
    Rising = TIC1; // Setup for next
  }
  PW = TIC1 - Rising; // the measurement
  Done = 0xFF;
  TFLG1 = 0x04; // ack, IC1F=0
}
```

Slide 35

Pulse-Width Measurement Using Interrupts

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

void Ritual(void){
  asm("sei"); // make atomic
  TCTL2 |= 0x30;
  // IC1F set on both rising and falling
  Rising = TCNT; // current TCNT
  Done = 0; // set on falling
  TFLG1 = 0x04; // Clear IC1F
  TMSK1 |= 0x04; // Arm IC1
  asm("cli");}
```

Slide 34

Pulse-Width Measurement Using Two Channels

Slide 36
Pulse-Width Measurement Using Two Channels

```c
unsigned int PW;    // units of 500 ns
unsigned char Done;  // Set each falling

void Ritual(void){
    asm("sei");    // make atomic
    TCTL2=(TCTL2&0xCF)|0x20;
    // falling edges of IC1, TCNT->TIC1
    TCTL2=(TCTL2&0xF3)|0x04;
    // rising edges of IC2, TCNT->TIC2
    Done=0;         // set on the falling edge
    TFLG1=0x04;     // Clear IC1F
    TMSK1|=0x04;    // Arm IC1, not IC2
    asm("cli");}
```

Pulse-Width Measurement Using Two Channels

```c
#pragma interrupt_handler TIC1handler()
void TIC1handler(void){
    PW=TIC1-TIC2;  // time from rise to fall
    Done=0xFF;
    TFLG1=0x04;   // ack by clearing IC1F
```