Definition

• What is a timer?
  – A timer is a digital sequential circuit that can count at a precise and programmable frequency
• Built-in timer (like in 68HC11)
  – The frequency is linked to the one of the internal clocks/oscillators
• External timer
  – Larger ranges of frequencies
Memory Elements – Flip Flop

• Introduction
  – Flip Flops are the basic building blocks of digital sequential circuits

• Brief History

R-S Flip Flop

• Logic Symbol
  ![Logic Symbol](image)

• Truth Table

<table>
<thead>
<tr>
<th>Mode of Operation</th>
<th>Input S</th>
<th>Input R</th>
<th>Output Q</th>
<th>Output Q̅</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prohibited</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Prohibited Do not use</td>
</tr>
<tr>
<td>Set</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>For setting Q to 1</td>
</tr>
<tr>
<td>Reset</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Resetting Q to 0</td>
</tr>
<tr>
<td>Hold</td>
<td>0</td>
<td>0</td>
<td>Q</td>
<td>Q̅</td>
<td>Depends Previous State</td>
</tr>
</tbody>
</table>

• Wiring Diagram

![Wiring Diagram](image)

Also called: R-S Latch, Set-Reset FF

• Waveform Diagram

![Waveform Diagram](image)
D Flip-Flop Schematic

D Flip-Flop Schematic
D and J-K Flip Flops

- The D flip-flop avoids the undefined states in the RSFF truth table by reducing the number of inputs.

<table>
<thead>
<tr>
<th>D</th>
<th>C</th>
<th>Q</th>
<th>Q̅</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>no change</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- The JKFF simplifies the RSFF truth table but keeps two inputs. The toggle state is useful in counting circuits.

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Q</th>
<th>Q̅</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>no change</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>p</td>
<td>toggle</td>
</tr>
</tbody>
</table>

Counters

- There are several different ways of categorizing counters:
  - binary-coded decimal (BCD) versus binary,
  - one direction versus up/down and
  - asynchronous ripple-through versus synchronous.

3-bit Ripple Counter

3-bit Synchronous Counter

Timing waveform of 3-bit up counter - divide by 8
You will learn how to:

- prescale the main timer counter
- record the time an event occurs
- take an action at a specific time
- handle counter overflows
- use Real-Time Interrupts
- use the Pulse Accumulator to:
  - count events
  - measure the duration of an event

Usage of timers

- Where can timers be found?
  - EVERYWHERE!
  - Clocks
  - Digital camera
  - Radar
  - Space exploration
  - Anywhere you could think to put one!
  - Anything that could be linked to time
Examples - Input

• Time between two rising edges
  – Radar
• Compute the time between two successive falling edges
  – Track & Field Timing
• Pulse width measurement
  – Time delay

Examples - Output

• Time basis
  – Clocks
  – Rectangle wave generator
    • Use specific registers (see HC11 specific part)
• Creating a time delay
  – 10 ms delay to program an EEPROM
    • 10 ms at 2MHz ⇔ $4E20$ cycles (20,000)
General Description of Main Timer

- Central element: 16-bit free running counter
  
  TCNT
  
<table>
<thead>
<tr>
<th>Bit 15</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>Bit 8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$100E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  - At reset counter starts from $0000 and counts up continuously
  - When $FFFF is reached, counter rolls over to $0000
  - May be read at any time using a double-byte instruction like LDD or LDX
  - Cannot be written or reset during operation

Timer – Prescaler

- Allows 4 clocking rates of the timer counter
  - E-Clock rate divided by: 1, 4, 8, 16
- At reset the default prescale factor is 1
- Must be set during the first 64 E-Clock cycles after reset
Changing the Prescaler

- Trade-off between timer resolution and timer range

<table>
<thead>
<tr>
<th>Prescale Factor</th>
<th>Resolution (one count)</th>
<th>Range (Overflow)</th>
<th>PR1</th>
<th>PR0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>500 ns</td>
<td>32.77 ms</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2 µs</td>
<td>131.1 ms</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>4 µs</td>
<td>262.1 ms</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>8 µs</td>
<td>524.3 ms</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Timer – Overflow

- Timer Overflow
  - Timer overflow flag (TOF) status bit is set each time the counter rolls over from $FFFF to $0000
  - TOF status bit can generate an automatic interrupt request by setting the timer overflow interrupt (TOI) enable bit
  - Registers associated: TMSK2,TFLG2
Timer – Flags

• Clearing timer flags
  – Load an accumulator with a mask that has a one in the bit(s)
    corresponding to the flag(s) to be cleared
  – Then write this value to TFLG1 or TFLG2
    • E.g. LDAA #$80, STAA TFLG2 will clear TOF
  – Or use BCLR instruction to clear the flag, the mask should have
    zeros in the bit positions corresponding to the flags to be cleared and
    ones in all other bits. (BCLR read->AND with inversed mask->write
    back)
    • E.g. BCLR TFLG2 #%01111111
• Caution!
  – Don’t use BSET to clear flags
    • because it could inadvertently clear one or more of the other
      flags in the register
    • BSET: read->OR with mask->write back

Input Capture

• Used to measure signal period/frequency
  (capture successive edges with same polarity)

• Measure pulse width (capture successive
  edges with alternate polarity)

• Used as time reference for output
  compare
Input Capture Registers

- There are three timer input pins on Port A (Pins PA0-PA2)
- Each input pin has a corresponding input capture register (16-bits each)
- When an edge is detected at a timer input pin, the current value of the free-running counter is stored in the corresponding input capture register

Input Capture Registers (cont)

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>Bit 8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Bit 0</td>
<td>$1010</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>Bit 8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Bit 0</td>
<td>$1011</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>Bit 8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Bit 0</td>
<td>$1012</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>Bit 8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Bit 0</td>
<td>$1013</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>Bit 8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Bit 0</td>
<td>$1014</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>Bit 8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Bit 0</td>
<td>$1015</td>
</tr>
</tbody>
</table>

- Can be read at any time as a pair of 8-bit registers using instructions like LDD or LDX
- Cannot be written by software
Input Capture Registers (cont)

- Operate independently of each other
- While reading the data in an input capture register, a new input capture to that register will be inhibited for one bus cycle so that the new input capture will not replace the old data before it is read.
- Inhibited capture will be delayed but will not be lost
- Both input captures and output compares are referenced from the same counter, so software latencies do not affect the accuracy to time delay

Timer Input Capture 4/Output Compare 5 Register

Use T14/O5 as either an input capture register or an output compare register, depending on the function chosen for the PA3 pin. To enable it as an input capture pin, set the I4/O5 bit in the pulse accumulator control register (PACTL) to logic level 1. To use it as an output compare register, set the I4/O5 bit to a logic level 0.
Input Edge-Detection Logic

- Used to select which edge of an input is detected

<table>
<thead>
<tr>
<th>Configuration</th>
<th>EDGxB</th>
<th>EDGxA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture Disabled</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Capture on Rising Edge Only</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Capture on Falling Edge Only</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Capture on Any Edge</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Interrupt Generation Logic

- Input capture status flags are automatically set to one each time a selected edge is detected

- Input capture interrupt enable bits
Measuring Long/Short Periods

• Long Periods:
  – Use software to keep track of counter overflows in an 8-bit register
  – Creates a 24-bit counter
  – Stored time values as 3-byte numbers
  – (see Ref. Manual 10.5)

• Short Periods:
  – Measure as short as one timer count periods by connecting one signal to two IC pins.

Other Uses of Input Capture Pins

• Can be used as general purpose input pins when the timer functions are not needed

• Can serve as flexible interrupt input pins
  – Have some advantages over the IRQ pin
  – See Ref. Manual section 10.5.7
Output Compare

- Used for outputting waveforms to control actuators or is used to generate time delays for I/O functions

- Accomplished by comparing the contents of the free-running counter with the compare register. When a match occurs, an output is generated

Output Compare: Basic Concept

- **16 Bit Register Stores a Number**
  - 5 possible Registers to store this number:

<table>
<thead>
<tr>
<th>Register Name</th>
<th>TOC1</th>
<th>TOC2</th>
<th>TOC3</th>
<th>TOC4</th>
<th>TOC5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>$1016 and $1017</td>
<td>$1018 and $1019</td>
<td>$101A and $101B</td>
<td>$101C and $101D</td>
<td>$101E and $101F</td>
</tr>
</tbody>
</table>

- **Comparator checks number against Free Running Counter (TCNT Register)**
  - Really 5 comparators, one for each register
  - This is done in hardware, no processor time used

- **When Counter matches TOCx Register, it triggers an event**
What “Event” is triggered?

**Three Possibilities:**

- Change the status of one or several Port A pins
- Set a Flag in TFLG Register
- Cause an Interrupt

Changing Port A Pin status with Output Compare

Output Compares 2 to 5:

- Each Output compare controls a SINGLE PIN:

<table>
<thead>
<tr>
<th>Output Compare 2</th>
<th>PA6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Compare 3</td>
<td>PA5</td>
</tr>
<tr>
<td>Output Compare 4</td>
<td>PA4</td>
</tr>
<tr>
<td>Output Compare 5</td>
<td>PA3</td>
</tr>
</tbody>
</table>
Changing Port A Pin status with Output Compare (cont)

Output Compares 2 to 5:
- TCTL1 Register Controls How Each Pin Changes

<table>
<thead>
<tr>
<th>OMx</th>
<th>OLx</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>OCx Does Not Affect Pin (OC1 Still May)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Toggle OCx Pin on Successful Compare</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Clear OCx Pin on Successful Compare</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Set OCx Pin on Successful Compare</td>
</tr>
</tbody>
</table>

Changing Port A Pin status with Output Compare (cont)

Output Compare 1:
- Causes 5 Port A pins to change simultaneously (PA3-PA7)
- Has priority over OC2-OC5

OC1M Register determines which Port A Pins will be Controlled by Output Compare 1

<table>
<thead>
<tr>
<th>PA7</th>
<th>PA6</th>
<th>PA5</th>
<th>PA4</th>
<th>PA3</th>
<th>OC1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>OC1M7</td>
<td>OC1M6</td>
<td>OC1M5</td>
<td>OC1M4</td>
<td>OC1M3</td>
<td>0</td>
</tr>
</tbody>
</table>

OC1D Register sets value to be written to Port A pins selected in OC1M

<table>
<thead>
<tr>
<th>OC1D7</th>
<th>OC1D6</th>
<th>OC1D5</th>
<th>OC1D4</th>
<th>OC1D3</th>
<th>0</th>
<th>0</th>
<th>OC1D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$100C</td>
<td>$100D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Setting a Flag with Output Compare

When Output Compare is successful it sets the corresponding Flag in TFLG1 Control Register:

```
<table>
<thead>
<tr>
<th>OC1F</th>
<th>OC2F</th>
<th>OC3F</th>
<th>OC4F</th>
<th>OC5F</th>
<th>ICF</th>
<th>ICF</th>
<th>ICF</th>
</tr>
</thead>
<tbody>
<tr>
<td>OC1F</td>
<td>OC2F</td>
<td>OC3F</td>
<td>OC4F</td>
<td>OC5F</td>
<td>ICF</td>
<td>ICF</td>
<td>ICF</td>
</tr>
</tbody>
</table>
```

Software must constantly poll TFLG1 register to check for flags

**Output Compare bits are cleared using the methods described earlier**

Causing an Interrupt with Output Compare

Output compare will cause an interrupt when the corresponding bit in TMSK1 is set:

```
<table>
<thead>
<tr>
<th>OC1I</th>
<th>OC2I</th>
<th>OC3I</th>
<th>OC4I</th>
<th>OC5I</th>
<th>ICF</th>
<th>ICF</th>
<th>ICF</th>
</tr>
</thead>
<tbody>
<tr>
<td>OC1I</td>
<td>OC2I</td>
<td>OC3I</td>
<td>OC4I</td>
<td>OC5I</td>
<td>ICF</td>
<td>ICF</td>
<td>ICF</td>
</tr>
</tbody>
</table>
```

TFLG1 $1023

TMSK1 $1022
Forced Output

If you need to change the state of a Port A Pin **BEFORE** output compare occurs

Use Forced Output...software triggers compare to occur and Pin A will change state accordingly

- Write a 1 to force an output compare. Writing a zero will have no effect
- Forced output will not cause the status bits to be set, therefore, no interrupt

Example program (input Capture)
Example program (input Capture)

Example program (input Capture)

REGRAS EQU $1000  *starting address of register block
PVIC1 EQU $00E8  *EVB pseudo vector for ICL
TIC1 EQU $10  *IC1 register
TICL2 EQU $21  *EDG4B,EDG4A,EDG1B,EDG1A,EDG3A
TMSK1 EQU $22  *OCl1...14/05I,IC11,...IC3I
TFLG1 EQU $23  *OCl1F...14/05F,IC1F,...,IC3F

ORG $1040  *s/w mode flag:ff-off,0-1st,1-last edge
ICIMOD RMB 1
ICIDUN RMB 1  *flag:0-not done,1-pulse measured
FSTEDG RMB 2  *time of first edge(16-bits)
PW RMB 2  *pulse width(16-bits)

Example program (input Capture)

Example program (input Capture)

ORG $2000  *prog starts in EVB RAM @ $2000
LDA @57E  *JMP (extended) opcode
STA PVIC1  *EVB’s ICL pseudo vector
LDA $58IC1  *address of ICI service routine
STA PVIC1+1  *finish JMP instruction to ICL routine

TOP  LDX #REGRAS  *point to register block
     LDA #X00010000
     STA TICL2,X
     LDA #31F
     STA ICIMOD  *FF-ICl off;
     CLR ICIDUN  *signal pulse not done
     BCLR TFLG1,X #FB
     BSET TMSK1,X #04
     CLI  *enable interrupts
     BRA TOP  *sets after pulse done

WAIT LDA ICIDUN  *loop till pulse has been timed
BEQ WAIT  *pulse done, disable interrupts
SEI  *display pulse width, etc.
BRA TOP  *go to top of main & repeat
Example program (input Capture)

*SRIC1-IC1 service routine

SRIC1  LDX  #REGBAS       *point to register b\text{lock}
       INC  ICIMOD       *$FF\rightarrow 0 \text{ 1st edge}, 0\rightarrow 1 \text{ 2nd edge}
       BNE  N01ST       *if not 0, this is trailing edge
       *process leading edge of pulse
       LOD  TIC1,X      *read time of first edge
       STD  FSTEDG      *save till next capture
       *reconfig IC1 for falling edge
       BCLR  TCTL2,X #30 *$EDG1B:EDG1A\rightarrow 0:0
       BSET  TCTL2,X #20 *$EDG1B:EDG1A\rightarrow 1:0
       BRA  O2IIC1      *done processing first edge
       *process trailing edge of pulse
N01ST  LOD  TIC1,X       *get time of trailing edge
       SUBD  FSTEDG      *time of 2nd minus time of 1st
       STD  PW       *store pulse width in hex number cycles.
       BCLR  TCTL2,X #30 *disable IC1
       LDAA  #$1       *IC1DUN
       STA  IC1DUN      *signal pulse measured
       O2IIC1  BCLR  TFLG1,X #$8 *clear IC1F
       RTI       *return from IC1 service

Period Measurement Example

Code

|       | EQU  | $D000               | DEFINE A 2-BYTE LOCATION TO STORE FIRST
|-------|------|---------------------|
| FIRST | EQU  | $D000               | DEFINE A 2-BYTE LOCATION TO STORE PERIOD
| EDGE  |      |                     |
| PERIOD| EQU  | $D002               |                     |
| ORG   | $C000|                     |                     |
| LDX   | #$1000|                    |                     |
| LDAA  | #$10 |                     |                     |
| STAA  | $1021|                     | EDGE DETECTION FOR IC1 SET TO RISING EDGES |
| LDA   | #$104|                     |                     |
| STAA  | $1023|                     | CLEARS ANY OLD FLAGS FROM IC1F |
| LOOP1 | BRCLR| $23,X #04          | LOOP1 LOOP HERE UNTIL FIRST RISING EDGE |
|       |      |                     |                     |
| LDD   | $1010|                     | READ TIME OF FIRST_capture
| STD   | FIRST|                     | STORE FIRST_capture VALUE |
| LDA   | #$04 |                     |                     |
| STAA  | $1023|                     | CLEAR THE IC1F FLAG BEFORE NEXT EDGE |
| LOOP2 | BRCLR| $23,X #04          | LOOP2 LOOP HERE UNTIL NEXT RISING EDGE |
|       |      |                     |                     |
| LDD   | $1010|                     | READ TIME OF SECOND_capture |
| SUBD  | FIRST|                     | FIND THE TIME DIFFERENCE BETWEEN EDGES |
| STD   | PERIOD|                    | STORE THE RESULT AS THE PERIOD |
Real-Time Interrupt

- Generates hardware interrupts at a fixed rate
- Free-running counter cannot be interrupted
- One of four rates - software selected
- One flag - set at the user determined rate
- Flag must be cleared after it is used, especially when using interrupts or a system lock up will occur.

RTI Registers

- TMSK2 $1024
  - Real-Time Interrupt Enable
- TFLG2 $1025
  - Real-Time Interrupt Flag
- PACTL $1026
  - Real-Time Interrupt Rate Selects
Real-Time Interrupt Rate Selects

For 8MHz Crystal Frequency
(2MHz E Clock)

<table>
<thead>
<tr>
<th>RTR1</th>
<th>RTR0</th>
<th>E/2(^{13}) Divided By</th>
<th>Nominal RTI Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4.10 ms</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>8.19 ms</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>16.38 ms</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>8</td>
<td>32.77 ms</td>
</tr>
</tbody>
</table>

RTI Example: Oven Control

- Example: Using a 33 ms RTI interrupt, the slave board should measure the oven temperature and save the value in a global variable. It should turn the heater on if the oven temperature is below the set point, and turn the heater off if the oven temperature is above the set point.
Pulse Accumulator Overview

- 8-bit Counter
- Incremented by edge on pin
- Used
  - to measure duration of pulse
  - number of events

Key Things to Know

- Can be read or written at any time
- 2 Modes
  - Event Counter
  - Gated Time Accumulation
- PAI Pin: Port A Pin 7
- Registers
Pulse Accumulator

• Associated Registers

<table>
<thead>
<tr>
<th>Register</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMSK2</td>
<td>TOI</td>
<td>RTII</td>
<td>PAOM1</td>
<td>PAI</td>
<td>0</td>
<td>0</td>
<td>PR1</td>
<td>PR0</td>
<td>1024</td>
</tr>
<tr>
<td>TFLG2</td>
<td>TOF</td>
<td>RTIF</td>
<td>PAOVF</td>
<td>PAIF</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1025</td>
</tr>
<tr>
<td>PACI</td>
<td>DDRA7</td>
<td>PAEN</td>
<td>PAMOD</td>
<td>PEDGE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1026</td>
</tr>
<tr>
<td>PACNT</td>
<td>PACTL</td>
<td>DDRA3</td>
<td>I4/O5</td>
<td>RTR1</td>
<td>RTR0</td>
<td>b7</td>
<td>b0</td>
<td></td>
<td>1027</td>
</tr>
</tbody>
</table>

- DDRA7: 0=input, 1=output. (normally configured as input when PA is used)
- PAEN: 0=PA disabled, 1=PA enabled
- PAMOD: 0=event counter, 1=gated time accumulation
- PEDGE
  - event counter: 0=PA responds to falling edges, 1=rising edge
  - gated time: 0=inhibit gate level is zero, 1=inhibit gate level is one
- PAOVI, PAOVF: PA overflow interrupt enable and flag
- PAII, PAIF: PA input edge interrupt enable and flag

Pulse Accumulator

• Event Counting Mode

- Events must be translated into rising/falling edges on PAI to be counted
- PAMOD=0, counts active edge of PAI
- Can cause interrupts after N events – writing N’s 2’s compliment to PACNT
- Can count more than 256 events by tracking the number of overflows.
- Example: a work piece counter on an assembly line can be realized using a light emitter/detector pair.
Event Counting Mode

- **PAMOD=0**
- **Counts Active Edge of PAI pin**

A diagram showing the connection of PA7/PAI/OC1 to an 8-bit counter labeled PACNT.

- **Example:** (PACNT=0;PAEN=1;PEDGE=1)

```
PAI
```

PACNT Value

```
1 2 3 4
```

Pulse Accumulator

- **Gate Time Accumulation Mode**
  - PACNT increments every 64th E-clock cycle when PAI pin is active.
  - PAMOD=1, PEDGE controls the inhibiting PAI pin level
  - Can be used to accumulate the total time the pin was active over a series of pulses
  - A common use is to measure pulse width (easier than using IC)
  - Interrupt function:
    - Overflow interrupt is useful in generating signals longer than the 8-bit counter range
    - PAI edge interrupt is useful for signaling the end of a timing period
Gated Time Accumulation Mode

- PAMOD=1
- Free-running E-clock divided by 64
- Subject to PAI pin being active

Gated Time Example

- PACNT=0; PAEN=1; PEDGE=1
  - PEDGE=1 means “inhibit gate is 1”
  (inhibit counting when PAI is 1)
Pulse Accumulator

- Example: Generate interrupt at specified time
  - Using gated time accumulation (PAMOD=1) to set pulse accumulator to interrupt after 5ms
    - Calculate time for one E/64 cycle
    - Divide delay by time for one E/64 cycle
    - Take 2’s complement and store in PACNT
    - When input goes to active level, counter will increment until overflow

Pulse Width Measurement

- Common use of Gated mode
- Measure duration of single pulses
- Easier than with Input Capture
- Counter is zero before pulse starts
- Pulse time is directly read after rising edge of pulse (need starting and ending count for input capture)
Assembly Code: Initialization to Count Negative Edges

LDAA #BIT54HI
STAA TFLG2 /*Clear previous interrupts*/
LDAA TMSK2
ORAA #BIT54HI /*Enable pulse accumulator interrupts*/
STAA TMSK2
LDAA PACTL
ANDA #BIT7_4LO /*BIT7_4LO = %00001111*/
ORAA #BIT6HI
STAA PACTL /*Select Event, Falling Edge, DDRA7-Input*/

References

1. M68HC11 E Series Technical Data
3. Introduction to Mechatronics and Measurement Systems
   David G. Alciatore, Michael b. Histand
4. Software and Hardware Engineering Motorola M68HC11
   Frederick M. Cady

Questions?