

---

# *Timers and Event Counters*



# In These Notes . . .

---

We learn the basics of the Timer/Counter peripheral

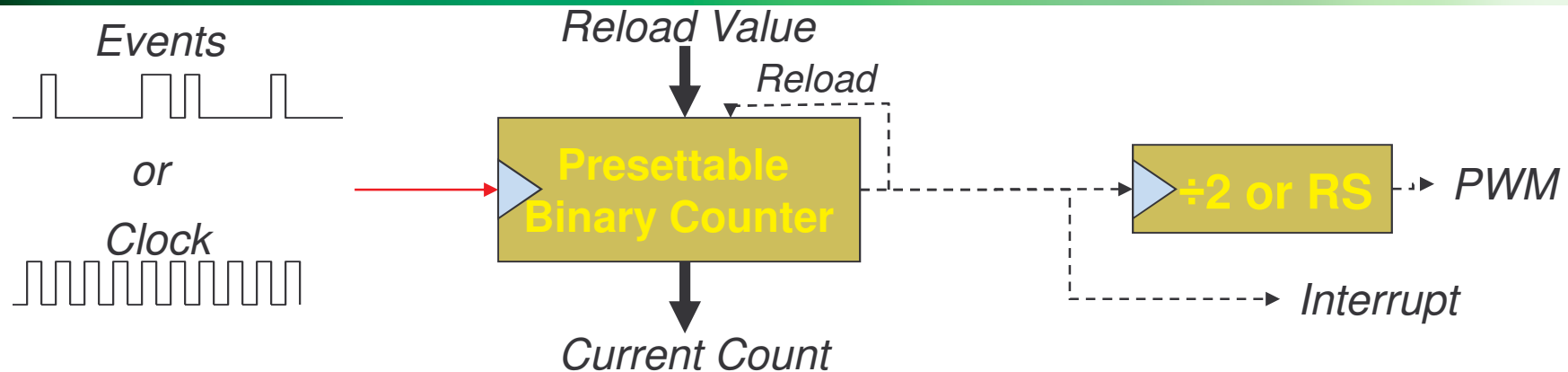
- Called *timers* by Renesas

We examine how to set up the timers for different operation:

- Timer mode
- Event counting mode
- Pulse Width Modulation (PWM) mode
- One-shot timer mode

We then examine how to use a microcontroller in these modes

# Timer/Counter Introduction

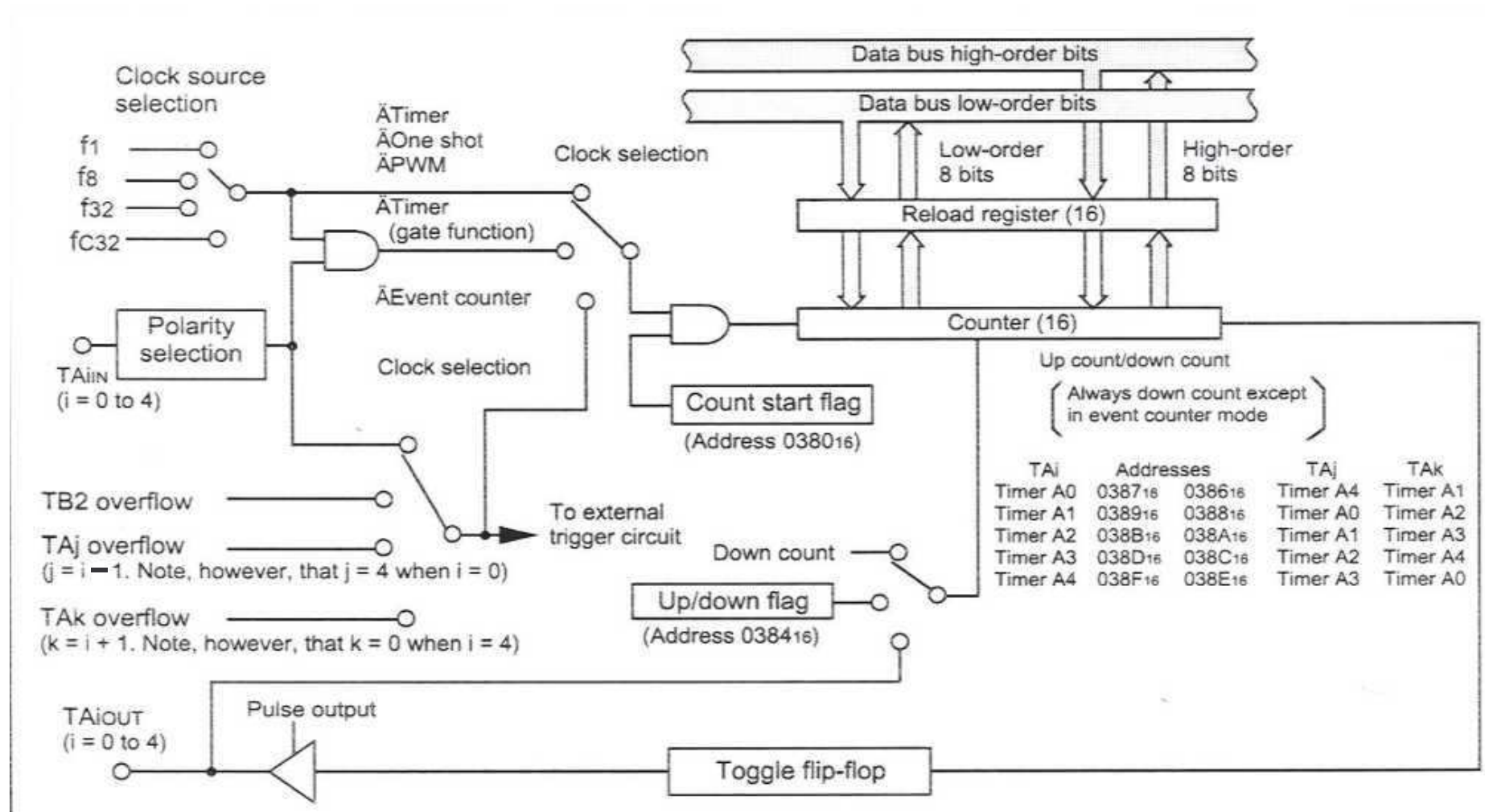


Common peripheral for microcontrollers

Based on pre-settable binary counter, enhanced with configurability

- Count value can be read and written by MCU
- Count direction can often be set to up or down
- Counter's clock source can be selected
  - **Counter mode:** count **pulses** which indicate **events** (e.g. odometer pulses)
  - **Timer mode:** clock source is periodic, so counter value is proportional to **elapsed time** (e.g. stopwatch)
- Counter's overflow/underflow action can be selected
  - Generate interrupt
  - Reload counter with special value and continue counting
  - Toggle hardware output signal
  - Stop!

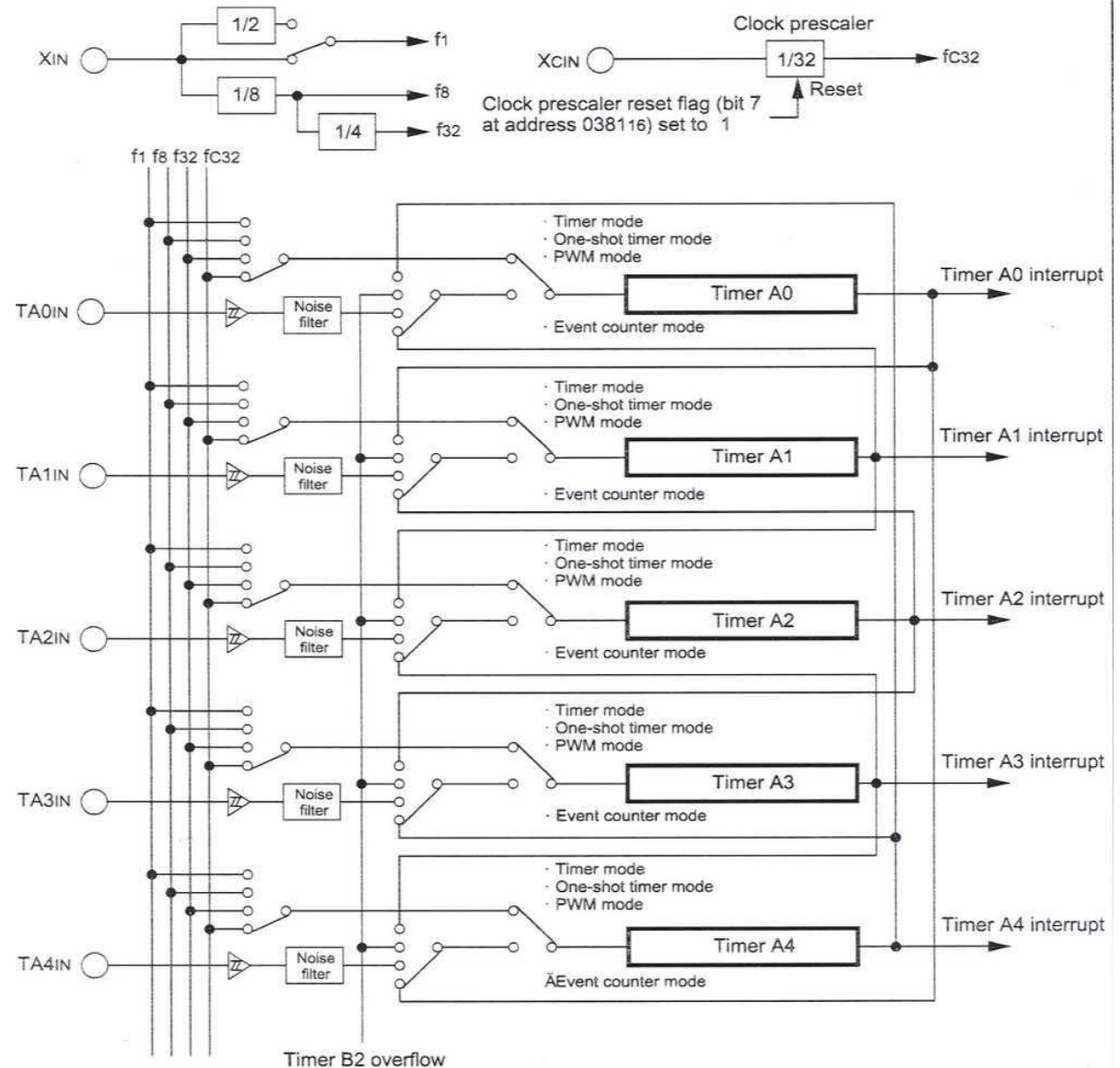
# Timer A Block Diagram



# High-Level Timer A Block Diagram

Timer A devices will be the most frequently used

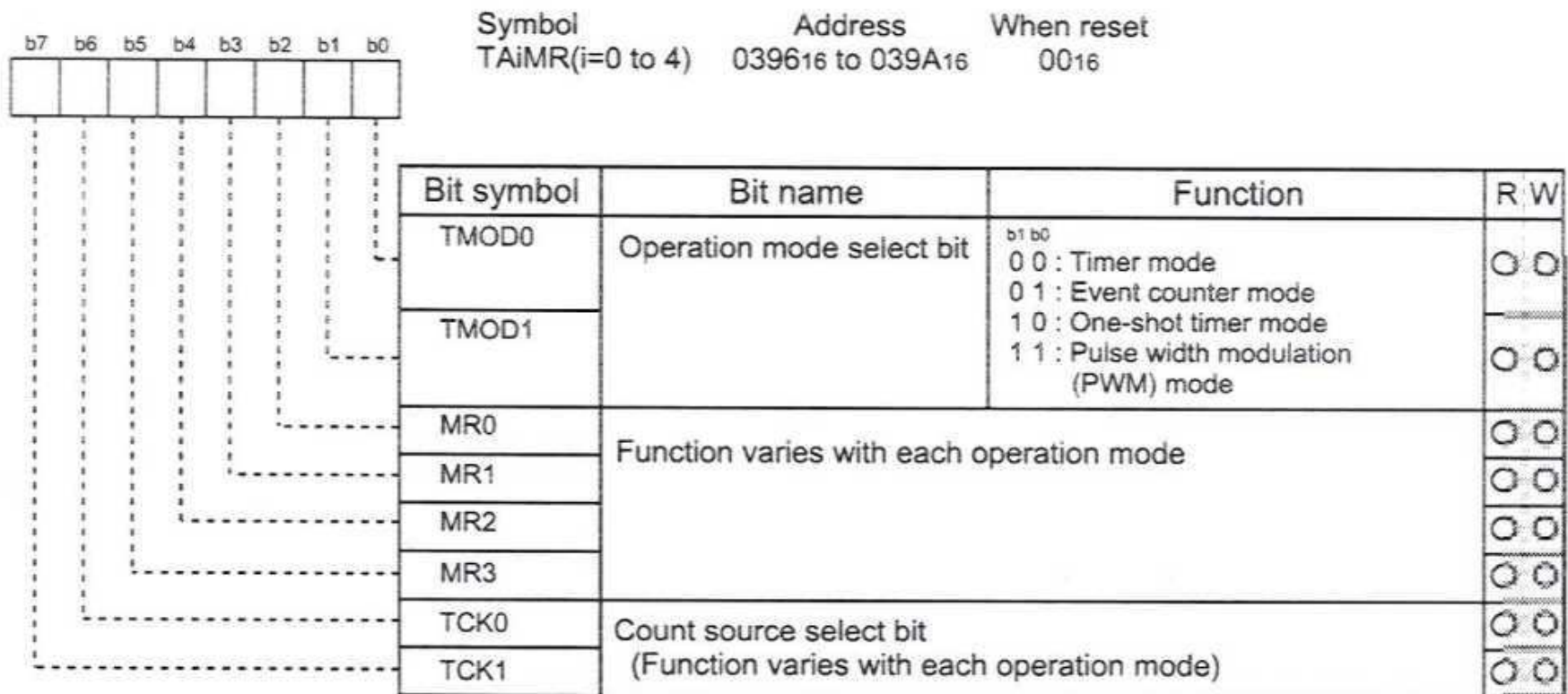
Flexible – can be cascaded to create larger timers (i.e. 32 bits long)



# Timer A Mode Register

To use the timer, you must set up how you wish to use it (i.e. via TA0MR). After that, the mode register has different settings depending on bits 1 and 0.

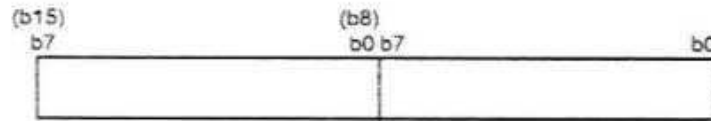
Timer Ai mode register





# Timer A “Data” Register

Timer Ai register (Note 1)



Symbol	Address	When reset
TA0	0387 <sub>16</sub> ,0386 <sub>16</sub>	Indeterminate
TA1	0389 <sub>16</sub> ,0388 <sub>16</sub>	Indeterminate
TA2	038B <sub>16</sub> ,038A <sub>16</sub>	Indeterminate
TA3	038D <sub>16</sub> ,038C <sub>16</sub>	Indeterminate
TA4	038F <sub>16</sub> ,038E <sub>16</sub>	Indeterminate

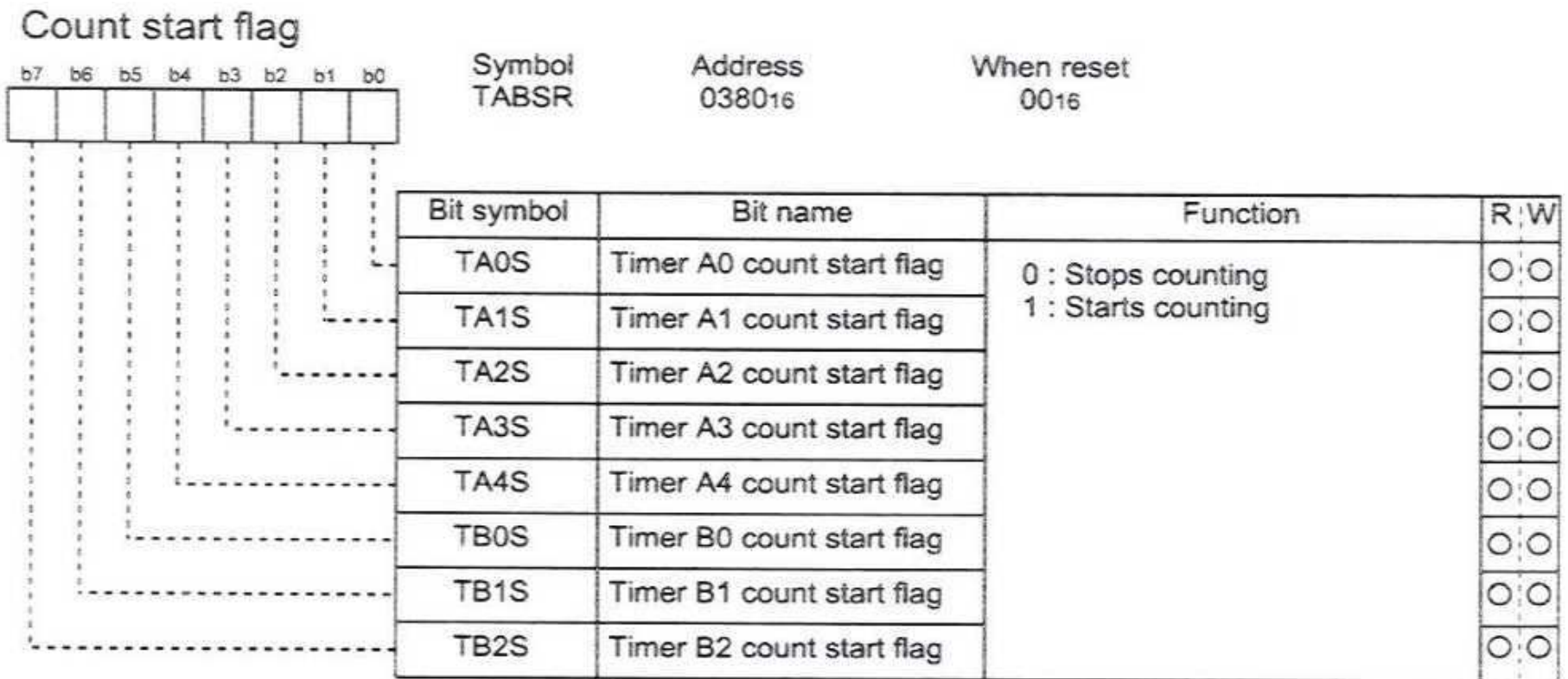
Function	Values that can be set	R	W
· Timer mode Counts an internal count source	0000 <sub>16</sub> to FFFF <sub>16</sub>	○	○
· Event counter mode Counts pulses from an external source or timer overflow	0000 <sub>16</sub> to FFFF <sub>16</sub>	○	○
· One-shot timer mode Counts a one shot width	0000 <sub>16</sub> to FFFF <sub>16</sub> (Note 2,6)	×	○
· Pulse width modulation mode (16-bit PWM) Functions as a 16-bit pulse width modulator	0000 <sub>16</sub> to FFFE <sub>16</sub> (Note 3,4,6)	×	○
· Pulse width modulation mode (8-bit PWM) Timer low-order address functions as an 8-bit prescaler and high-order address functions as an 8-bit pulse width modulator	00 <sub>16</sub> to FE <sub>16</sub> (High-order address) 00 <sub>16</sub> to FF <sub>16</sub> (Low-order address) (Note 3,5,6)	×	○

Note 1: Read and write data in 16-bit units.

Note 2: When the timer Ai register is set to 0000<sub>16</sub>, the counter does not operate and the timer Ai interrupt request is not generated. When the pulse is set to output, the pulse does not output from the TAIOUT pin.

# Count Start Register

Once the timer has been loaded with a value, start it counting.





# Counter Mode

Count pulses representing events

Odometer example

- Measure total distance your car has traveled
- Events are wheel rotations, measured with magnetic sensors (dirt-proof!)
- Distance traveled = counter value \* 100.53"
  - Assume 16" tire radius. Tire circumference =  $2\pi r = 100.53$ "
- Will limited range of 16 bit counter be a problem?
  - $100.53" * 2^{16}-1 = 1247.78$  miles
- Yes. So need to extend range in software.
  - Enable overflow interrupt for the timer
  - Create an ISR to count overflows



# TAiMR in Event Counting Mode

Timer Ai mode register  
(When not using two-phase pulse signal processing)

Bit	Symbol	Address	When reset
b17			
b16			
b15	0		
b14			
b13			
b12			
b11	0		
b10	1		

Bit symbol	Bit name	Function	R	W
TMOD0	Operation mode select bit	0 1 : Event counter mode (Note 1)	○	○
TMOD1			○	○
MR0	Pulse output function select bit	0 : Pulse is not output (TAiOUT pin is a normal port pin) 1 : Pulse is output (Note 2) (TAiOUT pin is a pulse output pin)	○	○
MR1	Count polarity select bit (Note 3)	0 : Counts external signal's falling edge 1 : Counts external signal's rising edge	○	○
MR2	Up/down switching cause select bit	0 : Up/down flag's content 1 : TAiOUT pin's input signal (Note 4)	○	○
MR3	0 (Must always be 0 in event counter mode)		○	○
TCK0	Count operation type select bit	0 : Reload type 1 : Free-run type	○	○
TCK1	Invalid when not using two-phase pulse signal processing Can be 0 or 1		○	○

# Up/Down Flag

The default is that the timer will count down.

Up/down flag (Note 1)

Symbol	Address	When reset
UDF	0384 <sub>16</sub>	00 <sub>16</sub>

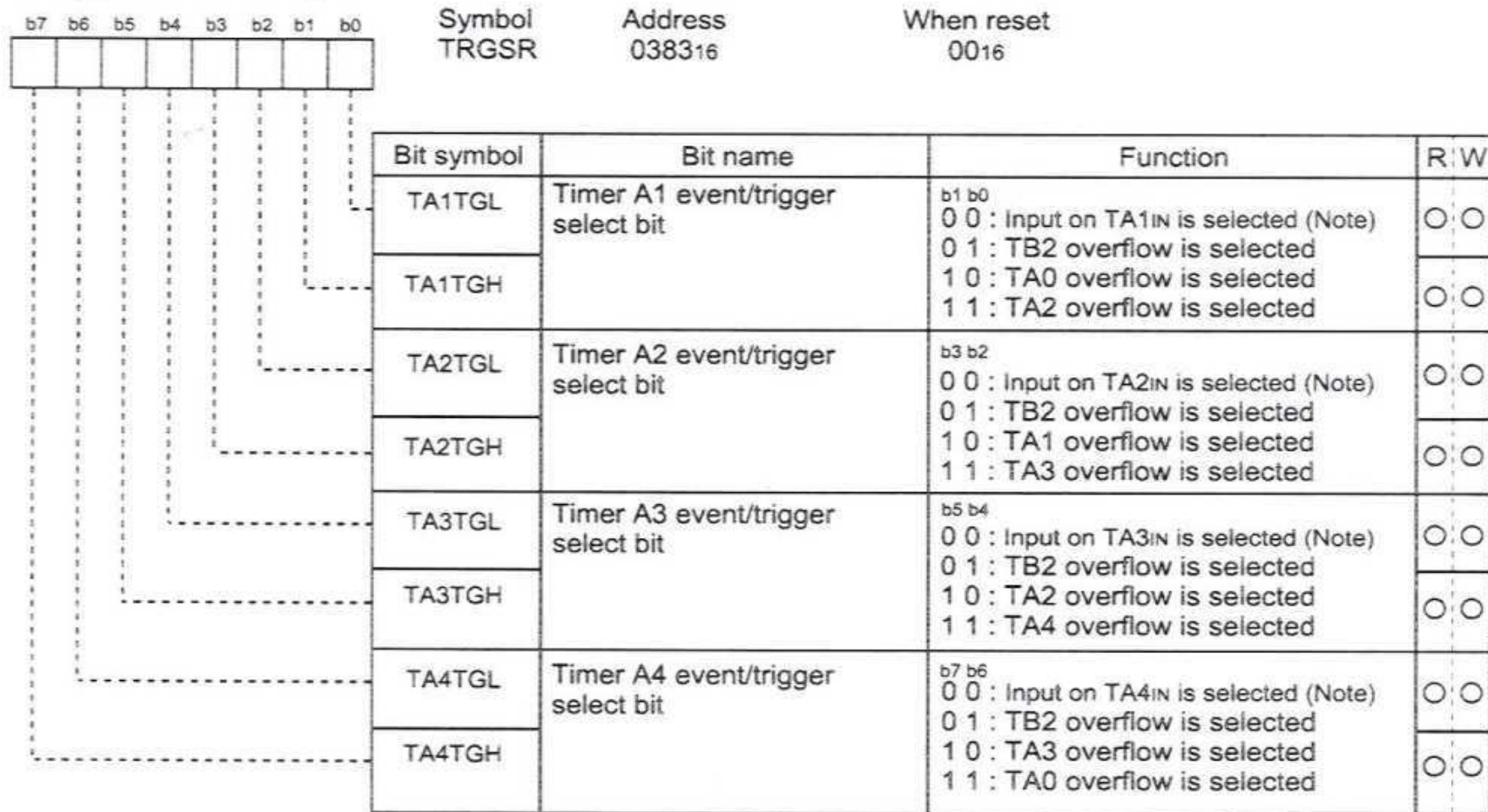
  

Bit symbol	Bit name	F unction	R/W
TA0UD	Timer A0 up/down flag	0 : Down count 1 : Up count  This specification becomes valid when the up/down flag content is selected for up/down switching cause	○ ○
TA1UD	Timer A1 up/down flag		○ ○
TA2UD	Timer A2 up/down flag		○ ○
TA3UD	Timer A3 up/down flag		○ ○
TA4UD	Timer A4 up/down flag		○ ○
TA2P	Timer A2 two-phase pulse signal processing select bit	0 : two-phase pulse signal processing disabled 1 : two-phase pulse signal processing enabled (Note 2)  When not using the two-phase pulse signal processing function, set the select bit to 0	× ○
TA3P	Timer A3 two-phase pulse signal processing select bit		× ○
TA4P	Timer A4 two-phase pulse signal processing select bit		× ○

# Trigger Select Register

You can set the trigger pulse of Timers A1 to A4

Trigger select register



Note: Set the corresponding port direction register to 0 .

# Example – Setting-up Event Mode

```
#define TIME_CONFIG 0x01 /* 00000001 val to load into timer mode reg
    |||||_ TMODE0,TMOD1: EVENT COUNTR MODE
    |||||_ MR0: NO PULSE OUTPUT
    ||||_ MR1: COUNT FALLING EDGES
    |||_ MR2: USE UP/DOWN FLAG
    ||_ MR3: = 0 IN EVENT COUNTER MODE
    |_ TCK0: RELOAD TYPE
    |_ TCK1: BIT NOT USED */

#define CNTR_IPL 0x03 // TA2 priority interrupt level
#define LED p7_2 // LED port on MSV1632 board
#define LED_PORT_DIRECTION pd7_2 // LED port dirn on MSV1632 board

void init() {
    ta2 = 100; //e.g for an automated packaging line, 100 items per case
    // the following procedure for writing an Interrupt Priority Level follows
    // that as described in the M16C data sheets under 'Interrupts'
    _asm (" fclr i"); // turn off interrupts before modifying IPL
    ta2ic |= CNTR_IPL; // use read-modify-write instruction to write IPL
    ta2mr = TIME_CONFIG;
    _asm (" fset i");

    ta2s = 1; //start counting
}
```



# Example – Using Event Mode

```
#pragma INTERRUPT /B TimerA2Int
void TimerA2Int(void) {
    int delaycntr;
    delaycntr = 0;
    count++;      //e.g for an automated packaging line, cnts # of cases
    LED = 1;
    while( delaycntr <0xffff) //software delay for flashing LED
        delaycntr++;
    LED = 0;
}

// initializes variables and LED port. Then does nothing but
// wait for TA2 interrupts.
void main (void) {
    int temp;
    count = 0;
    LED_PORT_DIRECTION = OUTPUT;
    init();
    while (1);
}
```



# Timer Mode – Measure Elapsed Time

Use a fixed-frequency signal  $f_{\text{base}}$  as a time-base

To measure elapsed time – *automatically instead of measuring twiddle (debug) bits on an oscilloscope*

- Clear the timer (or read its value and subtract it out later)
- Let time go by...
- Read timer value (possibly subtract out start time)

Example

```
void Compute_Cosine(float angle){
    unsigned t_start, t_stop, t_cosine;
    t_start = timer_current_count;
    compute,
        calculate,
            approximate,
                interpolate,
                    complete.
    t_stop = timer_current_count;
    t_cosine = t_stop - t_start;
}
```

Gate function

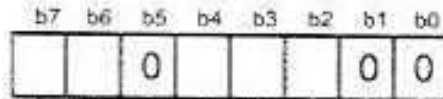
- Can use external signal (applied to TAIIN) to enable/disable counting
- Can select counting during either high or low portion
- Use MR1 and MR2 to configure
- *Convenient way to measure duty cycle*



**WARNING:  
THIS CODE WILL  
NOT COMPILE**

# TAiMR in Timer Mode

Timer Ai mode register



Symbol  
TAiMR(i=0 to 4)      Address      When reset  
0396<sub>16</sub> to 039A<sub>16</sub>      00<sub>16</sub>

Bit symbol	Bit name	Function	R	W
TMOD0	Operation mode select bit	b1 b0 0 0 : Timer mode	○	○
TMOD1			○	○
MR0	Pulse output function select bit	0 : Pulse is not output (TAiOUT pin is a normal port pin) 1 : Pulse is output (Note 1) (TAiOUT pin is a pulse output pin)	○	○
MR1	Gate function select bit	b4 b3 0 X (Note 2): Gate function not available (TAiIN pin is a normal port pin) 1 0 : Timer counts only when TAiIN pin is held L (Note 3) 1 1 : Timer counts only when TAiIN pin is held H (Note 3)	○	○
MR2			○	○
MR3			0 (Must always be 0 in timer mode)	○
TCK0	Count source select bit	b7 b6 0 0 : f1 or f2 0 1 : f8 1 0 : f32 1 1 : fc32	○	○
TCK1			○	○

# Example – Setting-up Timer Mode

```
#define TIME_CONFIG 0x40 /* 01000000 value to load into timer mode register
                        |||||_|_ TMOD0,TMOD1: TIMER MODE SELECTED
                        |||||_|_ MR0:          NO PULSE OUTPUT
                        |||||_|_ MR1,MR2:     GATE FUNCTION NOT SELECTED
                        ||_|_ MR3:           SET TO 0 IN TIMER MODE
                        ||_|_ TCK0,TCK1:     F DIVIDED BY 8 SELECTED */

#define CNTR_IPL 0x03 // TA0 priority interrupt level
#define LED p7_2 // LED4 is connected to p7_2 on the MSV1632/62 board

void init() {
    ta0 = 15000; // 24meg xtal, div by 8, times 15000 counts-> 5msec interrupts.

    // the following procedure for writing an Interrupt Priority Level follows
    // that as described in the M16C data sheets under 'Interrupts'
    // Note: ta0ic is the interrupt control register, memory location x0055
    _asm (" fclr i"); //turn off interrupts before modifying IPL
    ta0ic |= CNTR_IPL; // use read-modify-write instruction to write IPL
    ta0mr = TIME_CONFIG;
    _asm (" fset i");

    ta0s = 1; //start counting
}
```

# Example – Using Timer Mode

```
int time_cnt;
int count;           // Global count value, incremented every second

#pragma INTERRUPT /B TimerA0Int
void TimerA0Int(void) {
    if ((time_cnt += 5) > (1000)) { // = 1 second
        LED ^= 1;                 // toggle LED
        count++;                  // example 1 second "clock"
        time_cnt = 0;
    }
}

// Description: initializes variables and LED port. Then does nothing but
// wait for TA0 interrupts.

void main (void) {
    time_cnt = 0;
    count = 0;
    pd7_2 = 1;
    init();
    while (1); //LED flashing is interrupt driven
}
```

# Extending Time Range in Software

Note: An n-bit counter will overflow every  $2^n/f_{\text{base}}$  seconds

- 16-bit timer and 1 MHz clock ► 65.535 ms range

Need a variable to count overflows

- Using unsigned int, can count  $2^{16}-1$  overflows ►  $2^{32}-1$  ticks
  - Time range:  $2^{32}-1$  ticks at 1 MHz ► 4,294,967.295 ms  $\approx$  1.193 h
- ```
unsigned int overflow_count=0;
```

Need an ISR which runs on overflow

```
#pragma INTERRUPT timer_isr
void timer_isr(void) {
    overflow_count++;
}
```

Configure timer to continue counting upon overflow, don't need reload with a special value

Need a function to merge overflow and timer information

```
unsigned long get_ticks(void)
return (
(unsigned long) overflow_count << 16) +
(unsigned long) ta3;
}
```

Warning: this code will *intermittently* return an incorrect tick count! Shared Data Problem



## Example – Timer & Event Mode

---

The maximum amount of time one can count using two timers is:

- Cascade Timer A0 to Timer A1
- Use the 1/32 clock (bits 7 and 6 of TA0MR set to '10')
- Set both timer values to xFFFF
- Timer A1 output generates an interrupt

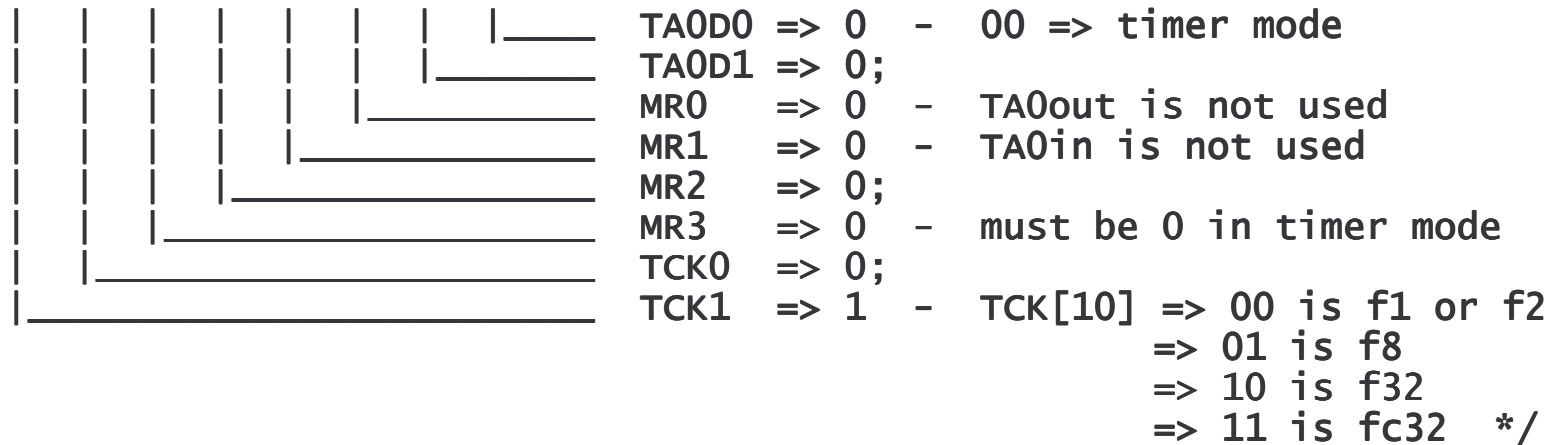
This would create a timer event which would interrupt the CPU every \_\_\_\_\_ seconds (almost \_\_\_ hours!)



# What Registers Need to be Set?

```
/*Timer A0 mode register settings
```

```
  b7 b6 b5 b4 b3 b2 b1 b0
```

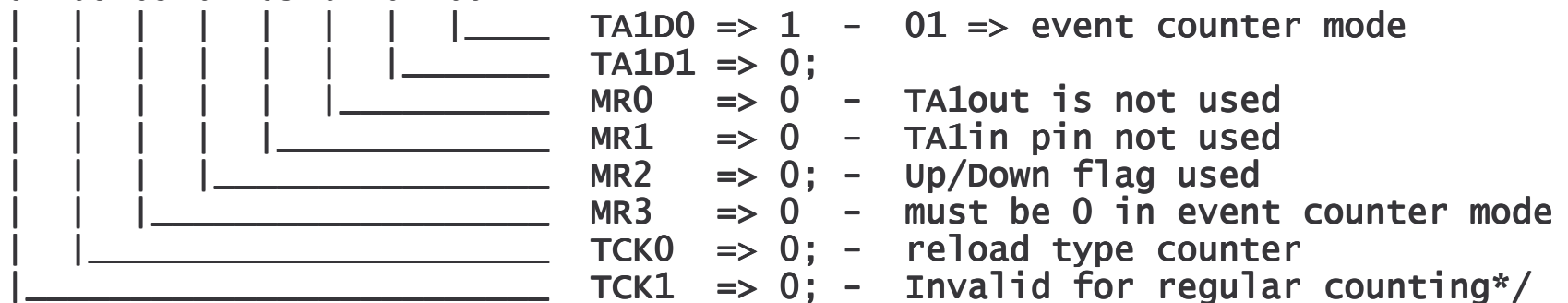


```
TA0MR = x80; // Timer A0 in timer mode with f32 clock
```

```
/******
```

```
- Timer A1 mode register settings
```

```
  b7 b6 b5 b4 b3 b2 b1 b0
```



```
TA1MR = x01; // Timer A1 in event counter mode with reload enable
```

# What Registers Need to be Set?

```
/*Value to initialize TA0 and TA1 to each time at initialization and  
each time overflow occurs
```

```
TAi/(XTAL * fx) = Z ms      - where TAi is initialization value,  
                           XTAL is the clock frequency (20MHz)  
                           fx is the clock multiplier (1,1/8,1/32)  
                           Z is the milliseconds between overflows
```

```
xFFFF = 65536  
65536/(24,000,000 * 1/32) = 87.381 milliseconds
```

```
TA0 overflows every 87.381 ms, TA1 counts the overflows up to 65536,  
then overflows every 5726.6 seconds (about 1.6 hours) */
```

```
TA0    =    xFFFF    // value loaded into TA0 data register  
TA1    =    xFFFF    // value loaded into TA1 data register
```

```
/******
```

```
Count Start Flag -> b0 = 1 - start TA0 counting  
                   b1 = 1 - start TA1 counting  
                   b2-b7 - not applicable*/
```

```
TABSR  =    x03      // start Timers A0 and A1 counting
```

# What Registers Need to be Set?

Timer Up/Down Function Register ->    b0 = 0 - TA0 counting down  
                                          b1 = 0 - TA1 counting down  
                                          b2-b7 - not applicable\*/

```
UDF = x00;        // UDF[10] => 00 makes Timers A0 and A1 count down
```

```
/******
```

```
- Timer Trigger Select Register
```

```
  b7 b6 b5 b4 b3 b2 b1 b0
```

```
          |  |_____  TA1TGL => 0 - TA0 overflow select  
          |_____  TA1TGH => 1; */
```

```
TRGSR =        x02;    // TA0 overflow is selected as trigger
```

```
/******
```

```
  Set TA1 priority level interrupt; */
```

```
TA1IC =        x03;    // set TA1 priority level interrupt
```

# Pulse-Width Modulation

Often need to generate an analog value: motor speed, servo motor position, incandescent light dimmer, adj. switch-mode power supply

Analog circuits have some disadvantage

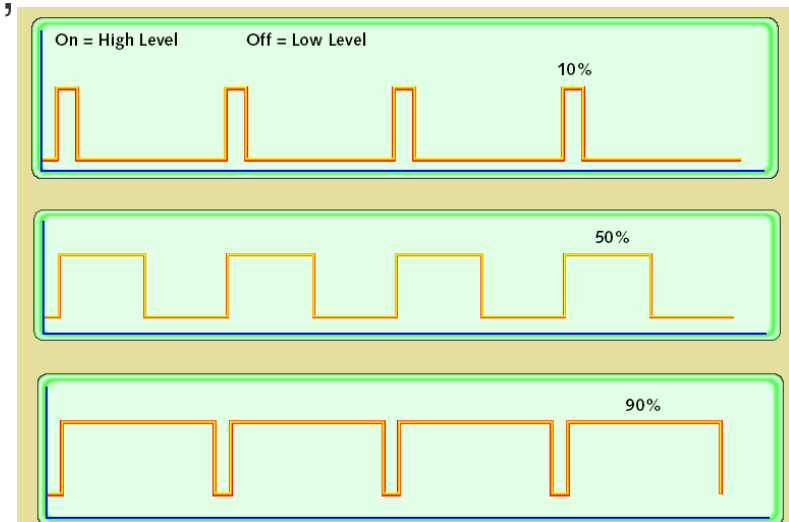
- Generating the voltage accurately – circuits drift with temperature and time
- Analog power amplifiers are inefficient ( $P = E \cdot I$ )
- Analog signals are much more sensitive to noise than digital signals

PWM provides a *digital encoding* of an *analog* value

- Now circuits are digital: more efficient, less sensitive to noise, temperature, aging effects

PWM signal characteristics

- Modulation frequency – how many pulses occur per second (fixed)
- Period –  $1/(\text{modulation frequency})$
- On-time – amount of time that each pulse is on (asserted)
- Duty-cycle –  $\text{on-time} \cdot \text{period}$
- Adjust *on-time* (hence *duty cycle*) to create encoded analog value



# Pulse-Width Modulation

We rely on *low-pass filtering* to “decode” (convert) this high-frequency PWM signal to an analog DC value

- Variable speed motor control: Motor has inertia (will coast if power is turned off)
- Incandescent Lamp Dimmer: Human eye has persistence of vision (retina averages signals above 30 Hz), and filament takes time to cool down and stop emitting visible light
- Adjustable switch-mode power supply: Rely on inductor and capacitor to maintain current and voltage when not drawing current from battery

Many Timer/Counter peripherals can also generate PWM signal

- For example
  - Count up
  - **Set** PWM output when counter value reaches 0
  - **Reset** PWM output when counter value reaches n
  - When counter reaches 11.111, overflow and continue counting

Read Michael Barr’s *Embedded Systems Programming* article “Pulse Width Modulation” for more PWM information

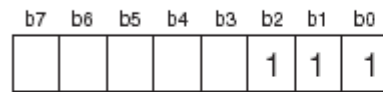
# Timer A PWM Description

| Item                                | Specification                                                                                                                                                                                                                                                                                                                                       |
|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Count source                        | f1, f2, f8, f32, fC32                                                                                                                                                                                                                                                                                                                               |
| Count operation                     | <ul style="list-style-type: none"> <li>The timer counts down (operating as an 8-bit or a 16-bit pulse width modulator)</li> <li>The timer reloads a new count at a rising edge of PWM pulse and continues counting</li> <li>The timer is not affected by a trigger that occurs when counting</li> </ul>                                             |
| 16-bit PWM                          | High level width $n / f_i$ $n$ : Set value<br><ul style="list-style-type: none"> <li>Cycle time <math>(2^{16}-1) / f_i</math> fixed</li> </ul>                                                                                                                                                                                                      |
| 8-bit PWM                           | <ul style="list-style-type: none"> <li>High level width <math>\times n (m+1) / f_i</math> <math>n</math>: values set to timer Ai register's high-order address</li> <li>Cycle time <math>(2^8-1) \times (m+1) / f_i</math> <math>m</math>: values set to timer Ai register's low-order address</li> </ul>                                           |
| Count start condition               | <ul style="list-style-type: none"> <li>External trigger is input</li> <li>The timer overflows</li> <li>The count start flag is set (= 1)</li> </ul>                                                                                                                                                                                                 |
| Count stop condition                | <ul style="list-style-type: none"> <li>The count start flag is reset (= 0)</li> </ul>                                                                                                                                                                                                                                                               |
| Interrupt request generation timing | PWM pulse goes "L"                                                                                                                                                                                                                                                                                                                                  |
| TAiIN pin function                  | Programmable I/O port or trigger input                                                                                                                                                                                                                                                                                                              |
| TAiOUT pin function                 | Pulse output                                                                                                                                                                                                                                                                                                                                        |
| Read from timer                     | When timer Ai register is read, it indicates an indeterminate value                                                                                                                                                                                                                                                                                 |
| Write to timer                      | <ul style="list-style-type: none"> <li>When counting stopped<br/>When a value is written to timer Ai register, it is written to both reload register and counter</li> <li>When counting in progress<br/>When a value is written to timer Ai register, it is written to only reload register (Transferred to counter at next reload time)</li> </ul> |



# Timer A Mode Register for PWM

Timer Ai mode register



Symbol  
TAiMR(i=0 to 4)

Address  
0396<sub>16</sub> to 039A<sub>16</sub>

When reset  
00<sub>16</sub>

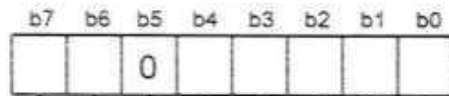
| Bit symbol | Bit name                             | Function                                                                                                  | R/W |
|------------|--------------------------------------|-----------------------------------------------------------------------------------------------------------|-----|
| TMOD0      | Operation mode select bit            | b1 b0<br>1 1 : PWM mode                                                                                   | 0   |
| TMOD1      |                                      |                                                                                                           | 0   |
| MR0        | 1 (Must always be 1 in PWM mode)     |                                                                                                           | 0   |
| MR1        | External trigger select bit (Note 1) | 0: Falling edge of TAiN pin's input signal (Note 2)<br>1: Rising edge of TAiN pin's input signal (Note 2) | 0   |
| MR2        | Trigger select bit                   | 0: Count start flag is valid<br>1: Selected by event/trigger select bits                                  | 0   |
| MR3        | 16/8-bit PWM mode select bit         | 0: Functions as a 16-bit pulse width modulator<br>1: Functions as an 8-bit pulse width modulator          | 0   |
| TCK0       | Count source select bit              | b7 b6<br>0 0 : f1 or f2<br>0 1 : f8<br>1 0 : f32<br>1 1 : fC32                                            | 0   |
| TCK1       |                                      |                                                                                                           | 0   |

Note 1: Valid only when the TAiN pin is selected by the event/trigger select bit (addresses 0382<sub>16</sub> and 0383<sub>16</sub>). If timer overflow is selected, this bit can be 1 or 0.

Note 2: Set the corresponding port direction register to 0.

# One-Shot Flag

One-shot start flag



Symbol  
ONSF

Address  
0382<sub>16</sub>

When reset  
00<sub>16</sub>

| Bit symbol   | Bit name                          | Function                                                                                                                                               | R | W |
|--------------|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|---|---|
| TA0OS        | Timer A0 one-shot start flag      | 1 : Timer start<br>When read, the value is 0                                                                                                           | ○ | ○ |
| TA1OS        | Timer A1 one-shot start flag      |                                                                                                                                                        | ○ | ○ |
| TA2OS        | Timer A2 one-shot start flag      |                                                                                                                                                        | ○ | ○ |
| TA3OS        | Timer A3 one-shot start flag      |                                                                                                                                                        | ○ | ○ |
| TA4OS        | Timer A4 one-shot start flag      |                                                                                                                                                        | ○ | ○ |
| Reserved bit |                                   | Must always be set to 0                                                                                                                                | ○ | ○ |
| TA0TGL       | Timer A0 event/trigger select bit | b7 b6<br>0 0 : Input on TA0IN is selected (Note)<br>0 1 : TB2 overflow is selected<br>1 0 : TA4 overflow is selected<br>1 1 : TA1 overflow is selected | ○ | ○ |
| TA0TGH       |                                   |                                                                                                                                                        | ○ | ○ |

Note: Set the corresponding port direction register to 0 .