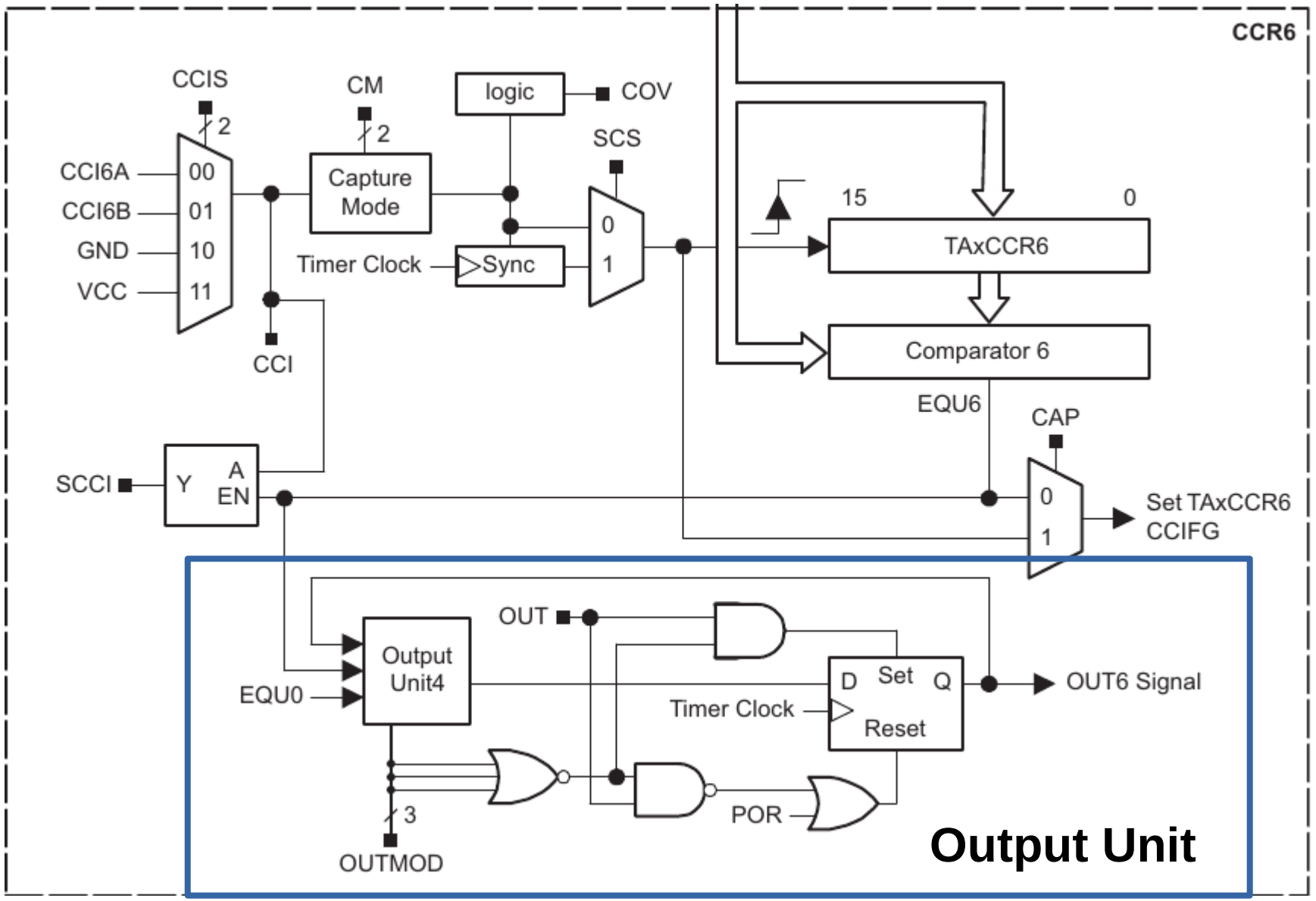
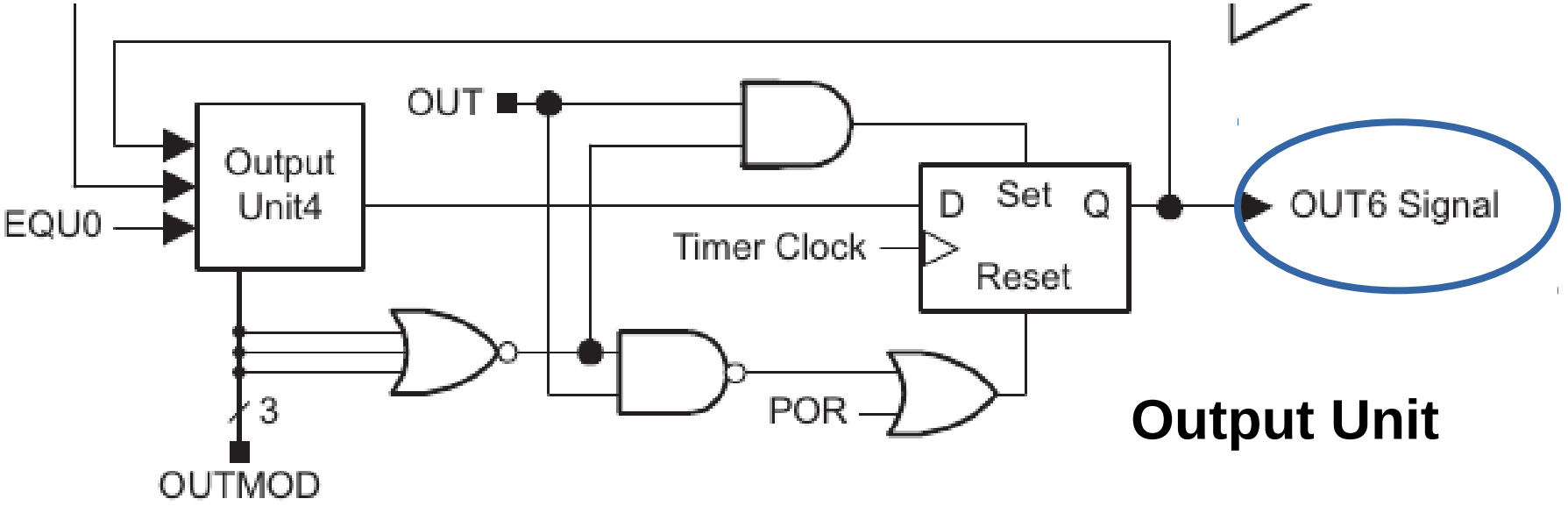
Module 7 – Pulse Width Modulation

Timer_A Compare Capture Register – used to generate PWM



This is from the Reference Manual, example has 6 CCR's, but our chip has 3 CCR's for Timer_A3

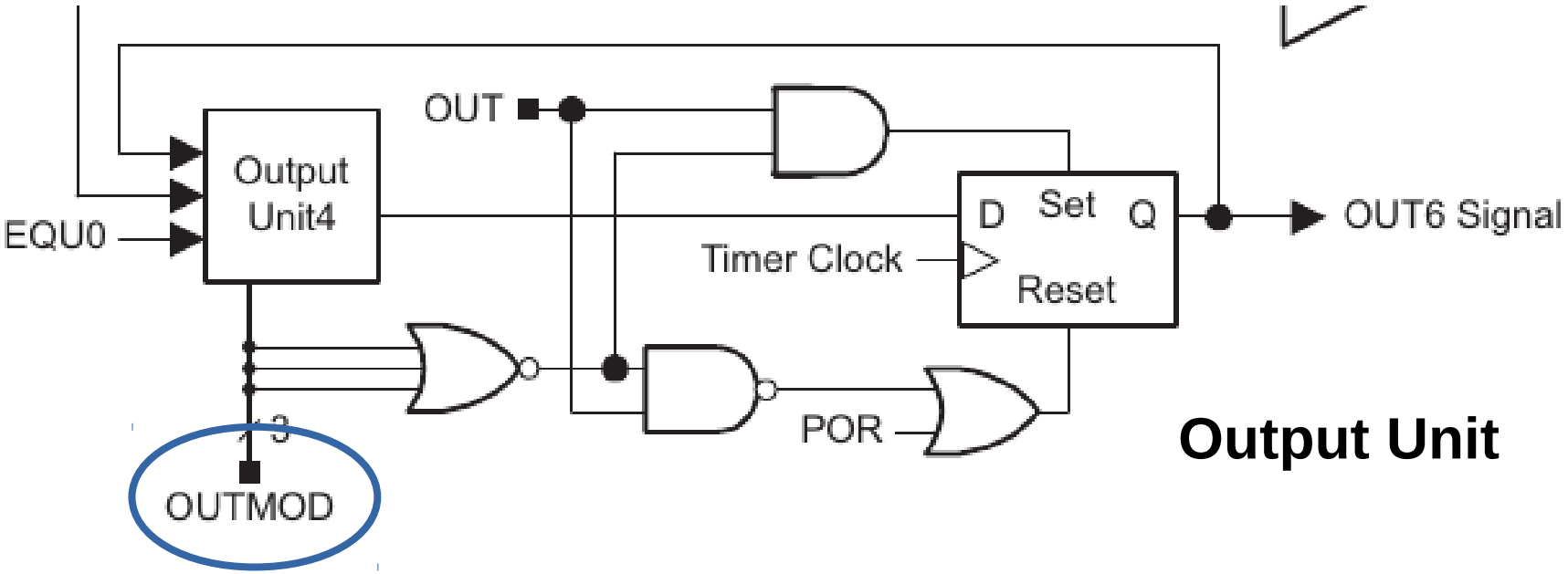


Output Modes

As you may have already seen, each CCR register has its own associated pin. For CCR1 on Timer0 this pin would be named “TA0.1”. Depending upon which mode you put the CCR into; this pin can be used as an input (for Capture) or an output (for either Capture or Compare).

When the pin is used as an output, its value is determined by the OUT bit-field in its control register. The exact details for this are TA0.1 = TA0CCTL1.OUT. (Sometimes you’ll just see this OUT bit abbreviated as OUT1.)

Besides routing the CCR OUT signal to a pin, it can also be used by other MSP430 peripherals. For example, on some devices the A/D converter could be triggered by the timer directly.

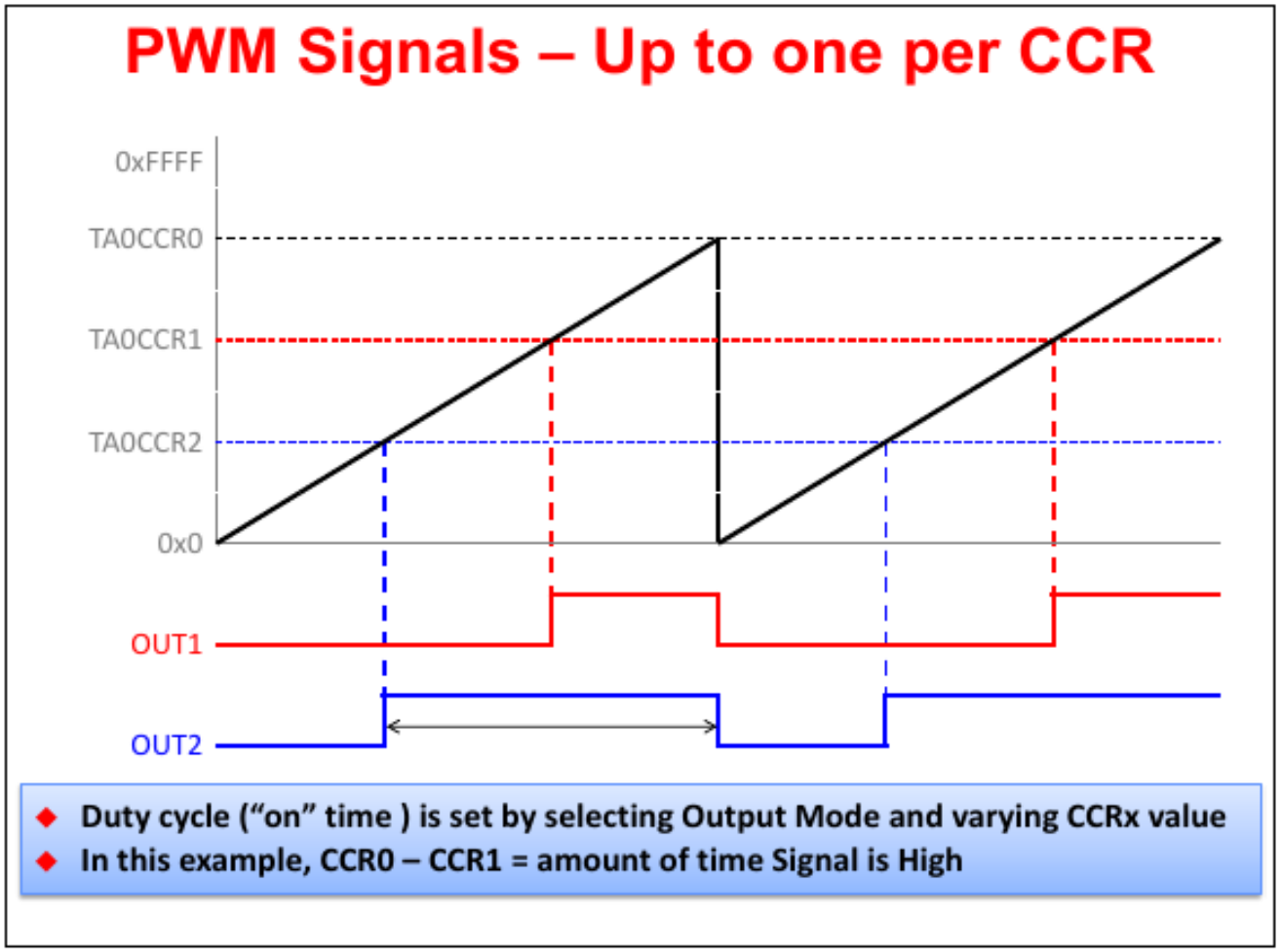


So, what is the value of OUT for any given CCR register?

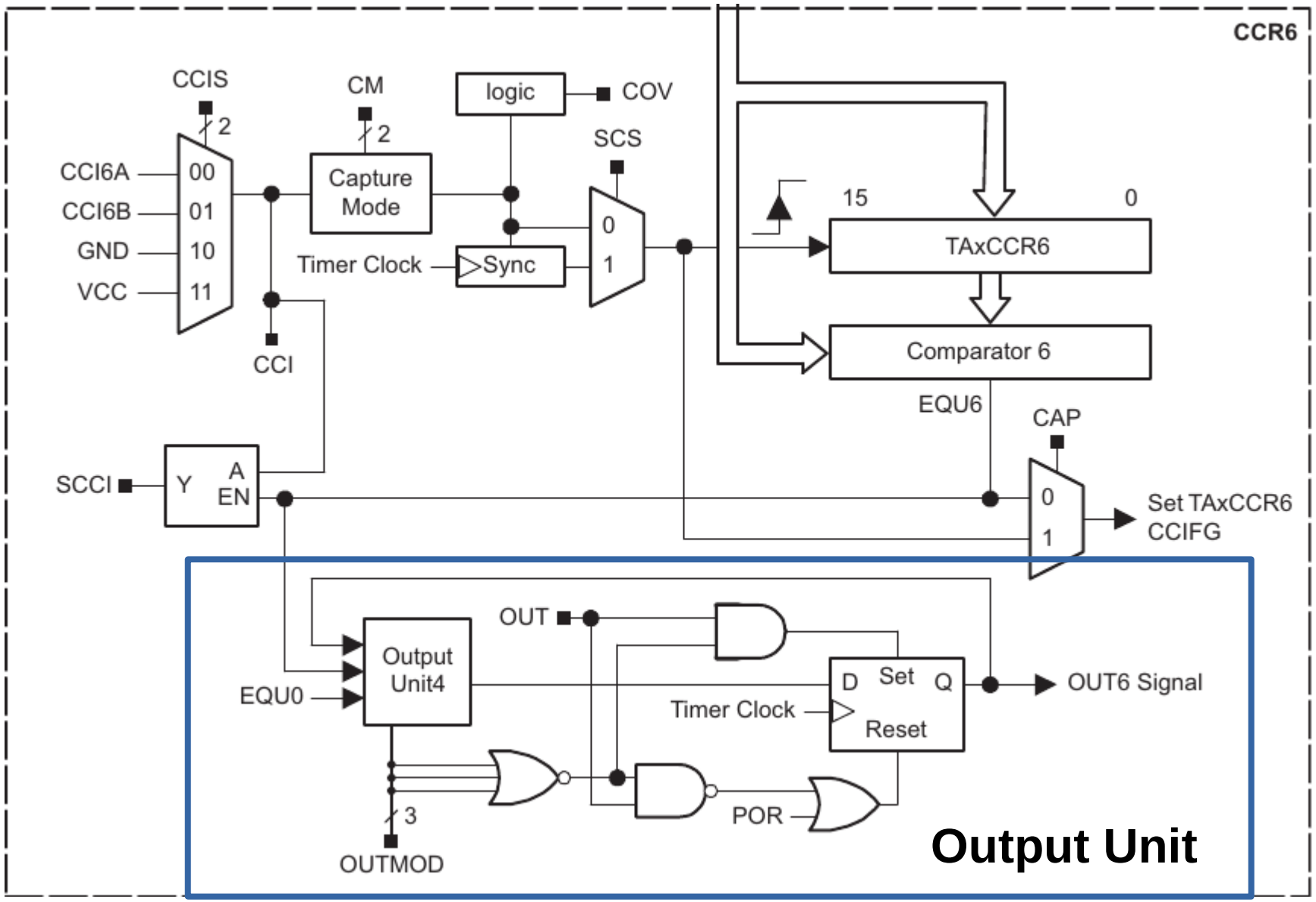
The value of OUT is determined by the OutputMode, as we discussed earlier. (Each CCR control register has its own OUTMOD bit-field). This setting tells the OUT bit how to react as each compare or capture occurs. As previously stated, there are 8 different OutputMode choices.

For example, setting OUTMOD = 0 mean it's not changed by the timer's hardware. That is, it's under software control. You can set OUT to whatever you like by writing to it in the CCRx control register.

In the case of the MSP430, any timer can generate a PWM waveform by configuring the CCR registers appropriately. In fact, if you are using a Timer_A5, you could output 4 or 5 different PWM waveforms.



Timer_A Compare Capture Register – used to generate PWM



12.2.5.1 Output Modes

The output modes are defined by the OUTMOD bits and are described in Table 12-2. The OUTn signal is changed with the rising edge of the timer clock for all modes except mode 0. Output modes 2, 3, 6, and 7 are not useful for output unit 0 because EQUn = EQU0.

Table 12-2. Output Modes

OUTMOD	Mode	Description
000	Output	The output signal OUTn is defined by the OUT bit. The OUTn signal updates immediately when OUT is updated.
001	Set	The output is set when the timer *counts* to the TAxCCRn value. It remains set until a reset of the timer, or until another output mode is selected and affects the output.
010	Toggle/Reset	The output is toggled when the timer *counts* to the TAxCCRn value. It is reset when the timer *counts* to the TAxCCR0 value.
011	Set/Reset	The output is set when the timer *counts* to the TAxCCRn value. It is reset when the timer *counts* to the TAxCCR0 value.
100	Toggle	The output is toggled when the timer *counts* to the TAxCCRn value. The output period is double the timer period.
101	Reset	The output is reset when the timer *counts* to the TAxCCRn value. It remains reset until another output mode is selected and affects the output.
110	Toggle/Set	The output is toggled when the timer *counts* to the TAxCCRn value. It is set when the timer *counts* to the TAxCCR0 value.
111	Reset/Set	The output is reset when the timer *counts* to the TAxCCRn value. It is set when the timer *counts* to the TAxCCR0 value.

These output modes combine with what ‘count’ Mode is being used

The OUTn signal is changed when the timer reaches the TAxCCRn and TAxCCR0 values, depending on the output mode. An example is shown in Figure 12-13 using TAxCCR0 and TAxCCR1.

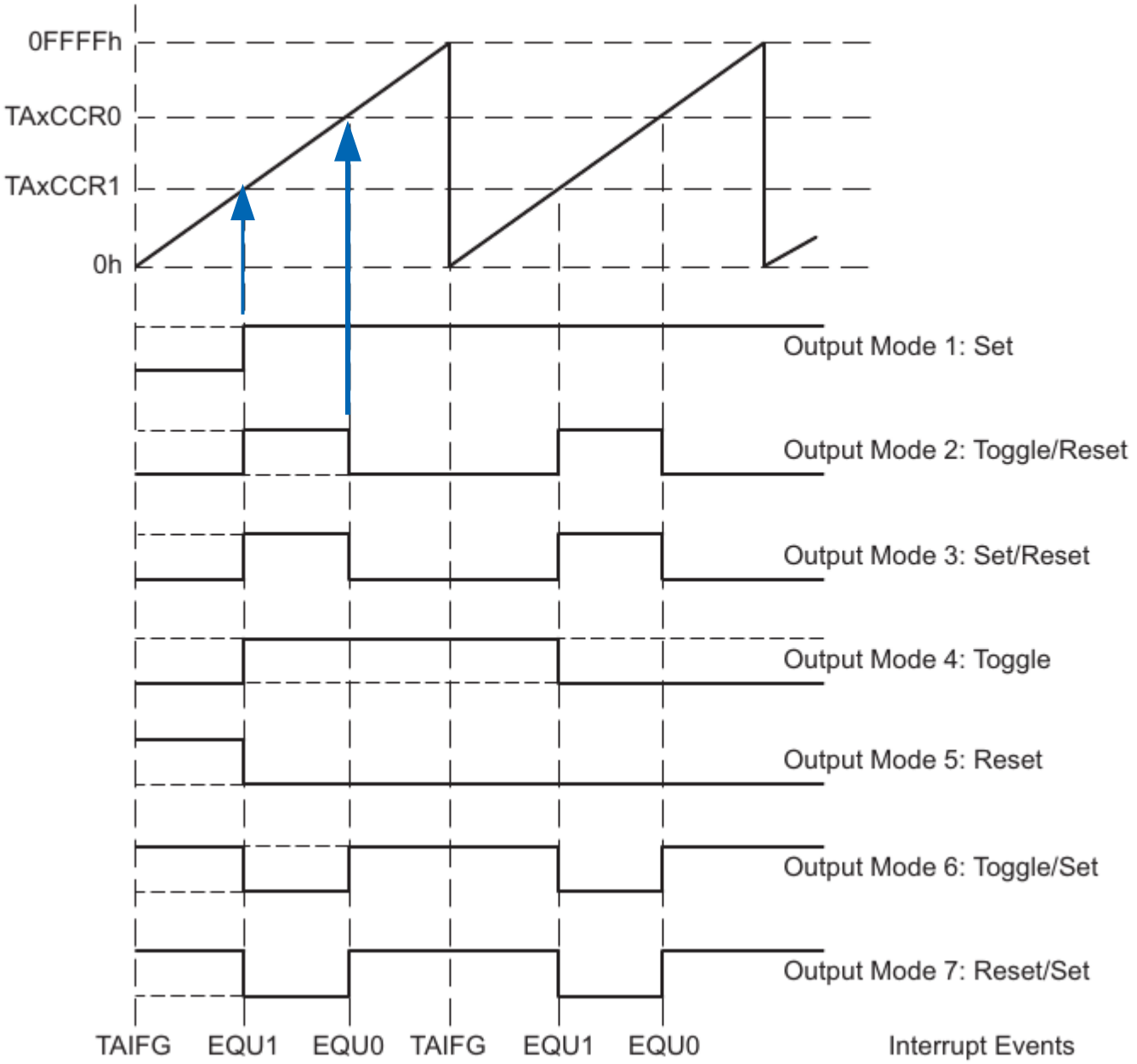


Figure 12-13. Output Example – Timer in Continuous Mode

The OUTn signal is changed when the timer *counts* up to the TAxCCRn value and rolls from TAxCCR0 to zero, depending on the output mode. Figure 12-12 shows an example using TAxCCR0 and TAxCCR1.

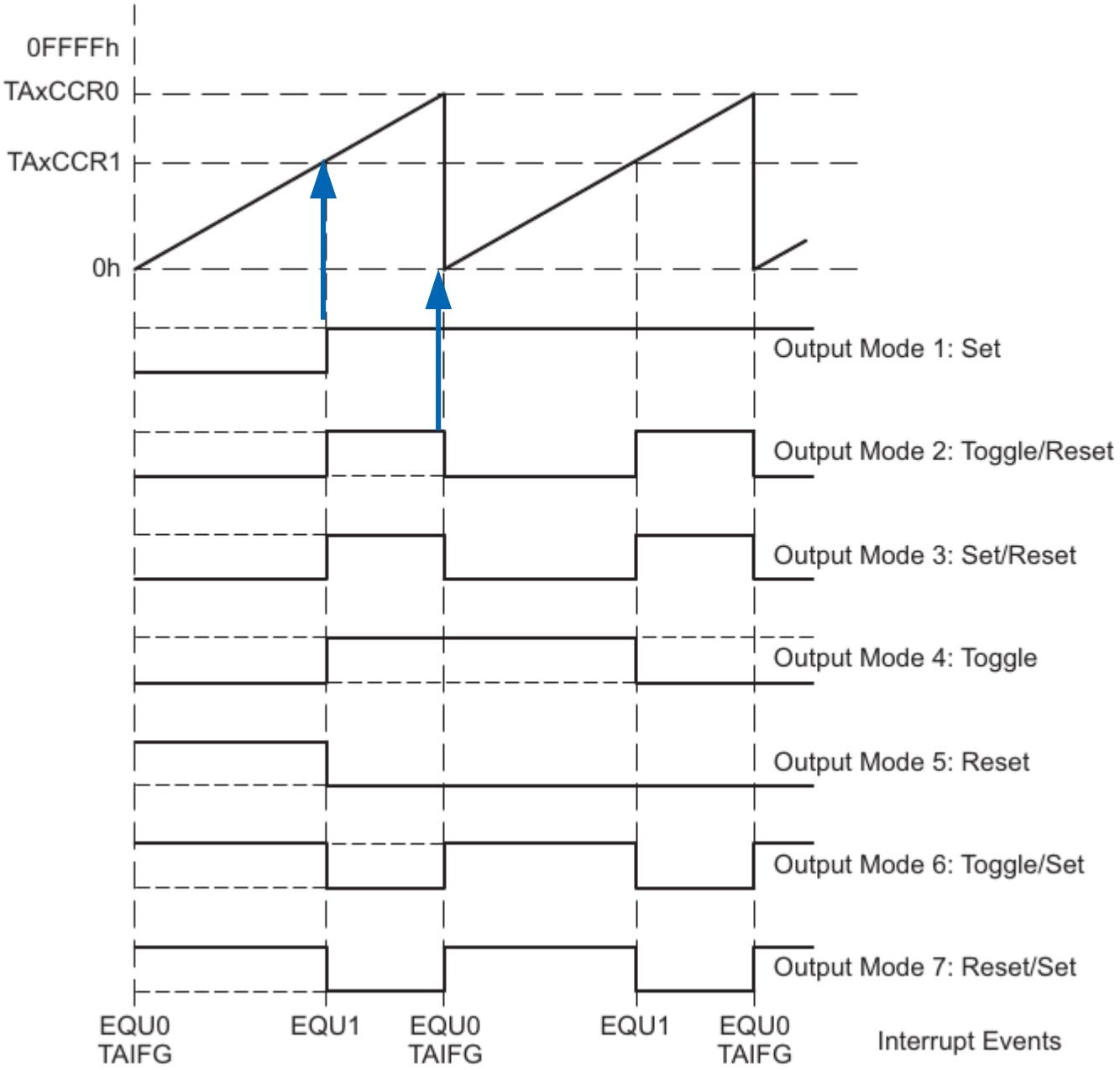


Figure 12-12. Output Example – Timer in Up Mode

The Up/Down mode supports applications that require dead times between output signals (see Section 12.2.5). For example, to avoid overload conditions, two outputs driving an H-bridge must never be in a high state simultaneously. In the example shown in Figure 12-9, the t_{dead} is:

$t_{dead} = t_{timer} \times (TAxCCR1 - TAxCCR2)$

Where:

t_{dead} = Time during which both outputs need to be inactive
t_{timer} = Cycle time of the timer clock
TAxCCRn = Content of capture/compare register n

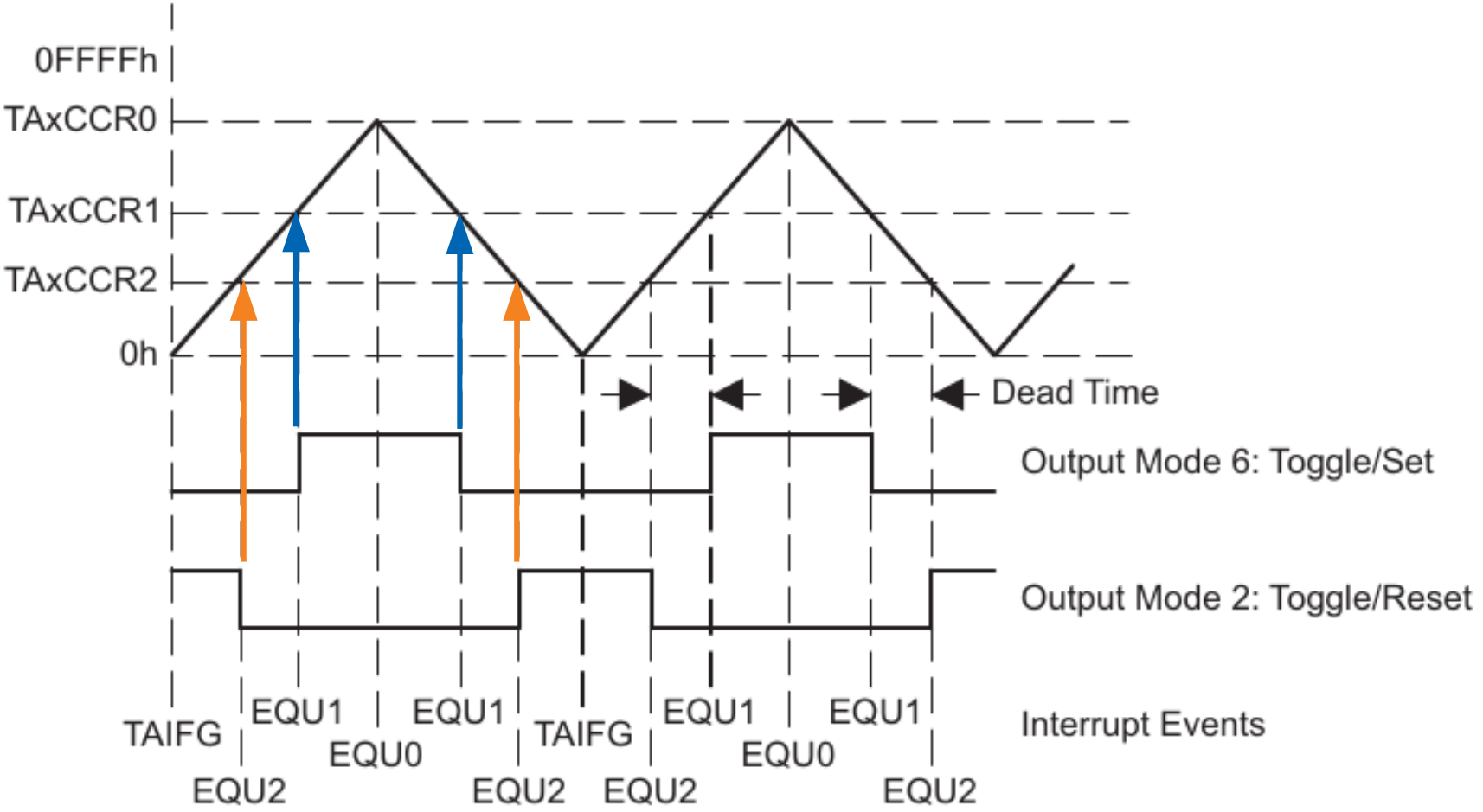


Figure 12-9. Output Unit in Up/Down Mode

The OUTn signal changes when the timer equals TAxCCRn in either count direction and when the timer equals TAxCCR0, depending on the output mode. Figure 12-14 shows an example using TAxCCR0 and TAxCCR2.

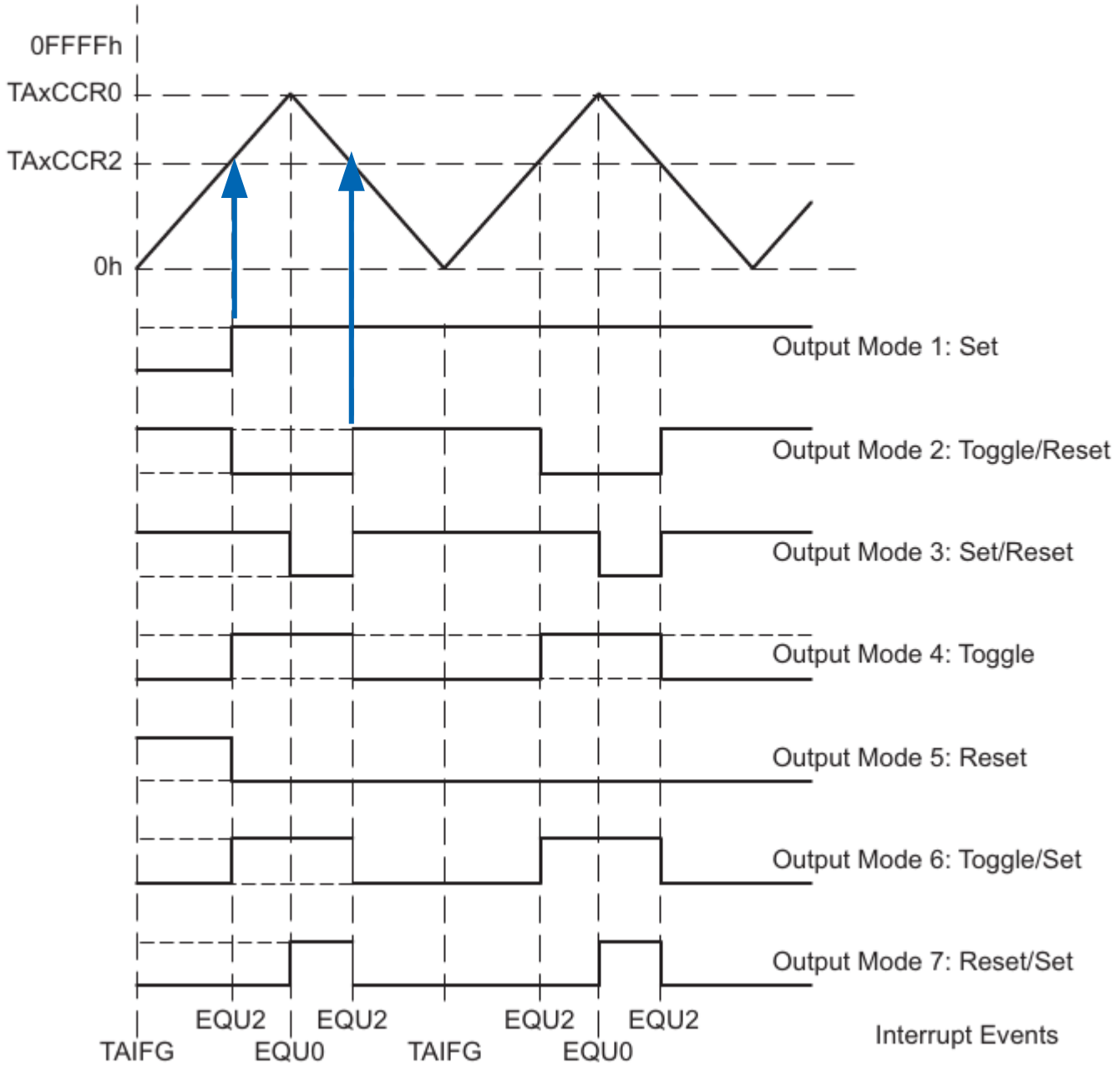


Figure 12-14. Output Example – Timer in Up/Down Mode

Table 12-6. TAxCCTLn Register Description

7-5	OUTMOD	RW	0h	Output mode. Modes 2, 3, 6, and 7 are not useful for TAxCCR0 because EQUx = EQU0. 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Capture/compare input. The selected input signal can be read by this bit.
2	OUT	RW	0h	Output. For OUTMOD = 0, this bit directly controls the state of the output. 0b = Output low 1b = Output high

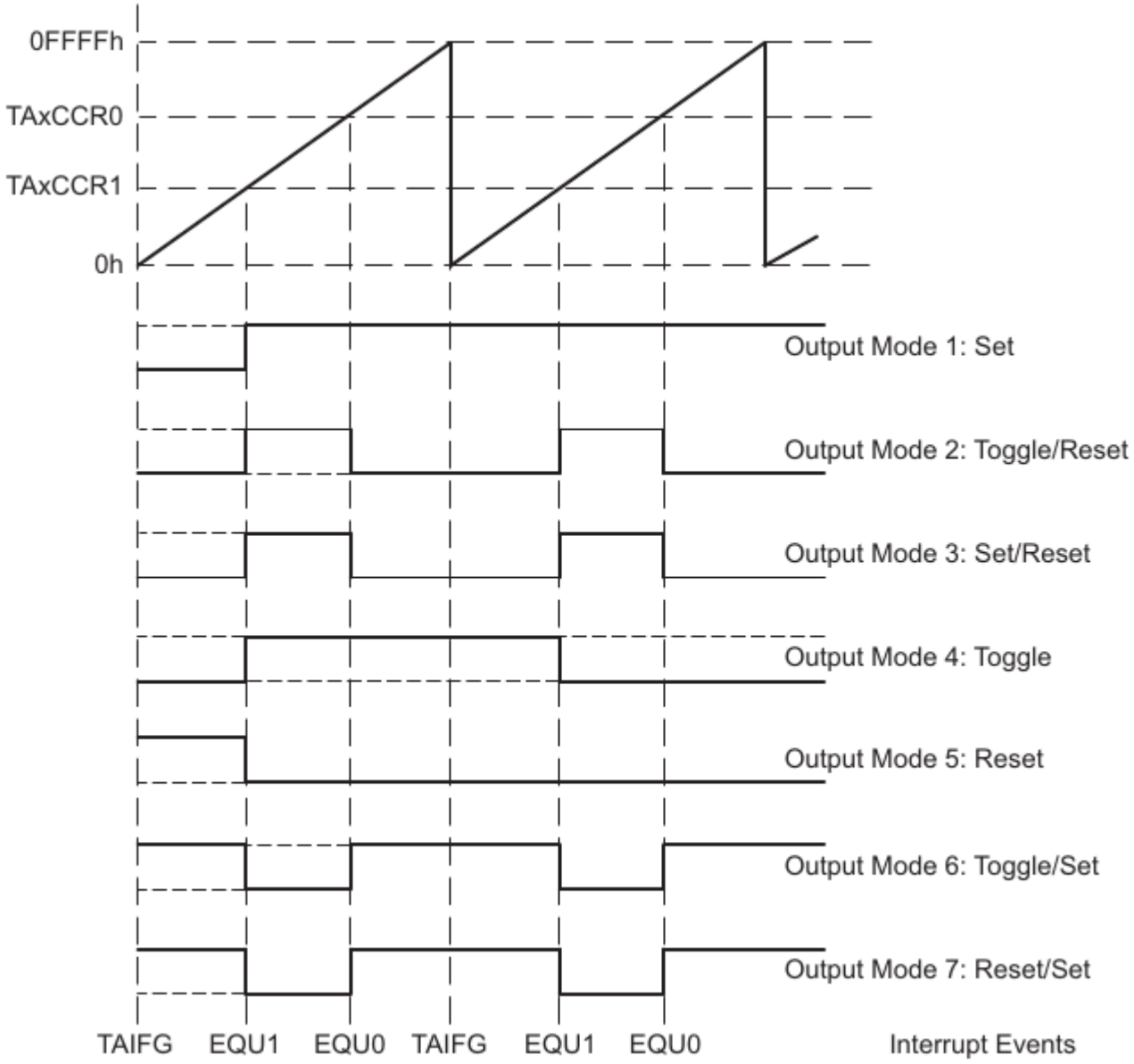


Figure 12-13. Output Example – Timer in Continuous Mode

EQU - The nomenclature used in the MSP430 User's Guide.

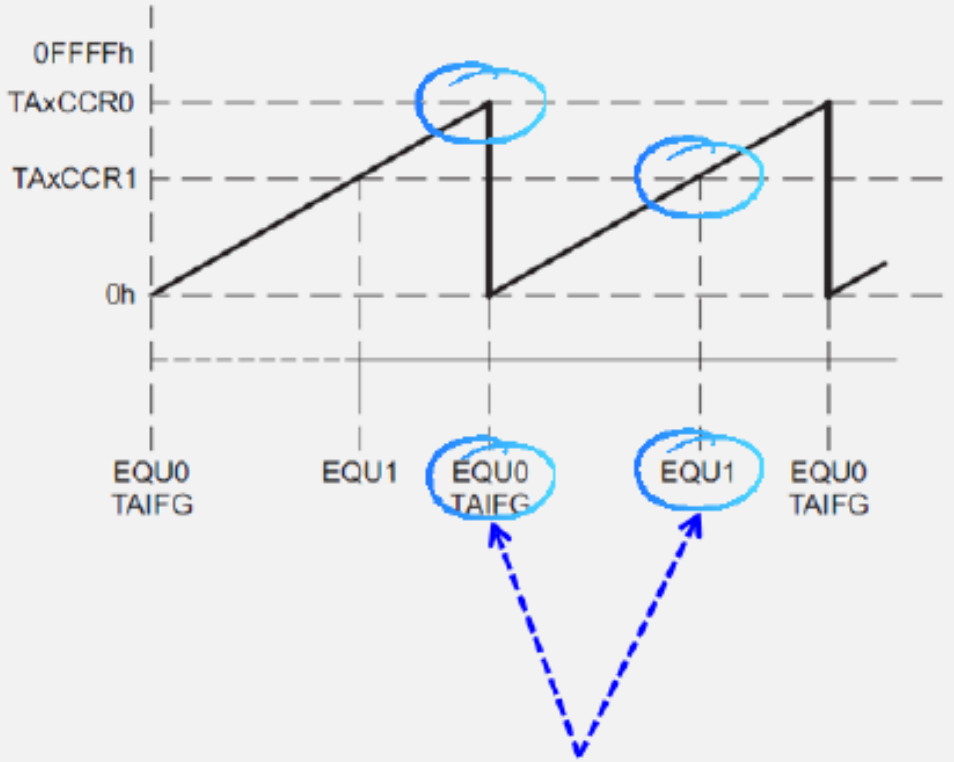
Apparently, there is an EQU (equate) signal inside the timer for each CCR. For example, the equate signal for CCR1 would be called EQU1.

While these EQU values cannot be read directly from any of the timer control registers, the documentation makes use of them to describe when a comparison becomes true.

Therefore, when the timer counter (TAR) becomes equal to a compare register (CCR), the associated EQU signal becomes true.

This can be seen in the following diagram captured from the TIMER_A documentation. Notice how EQU0 becomes true when TAR=CCR0; likewise, EQU1 becomes true when TAR=CCR1.

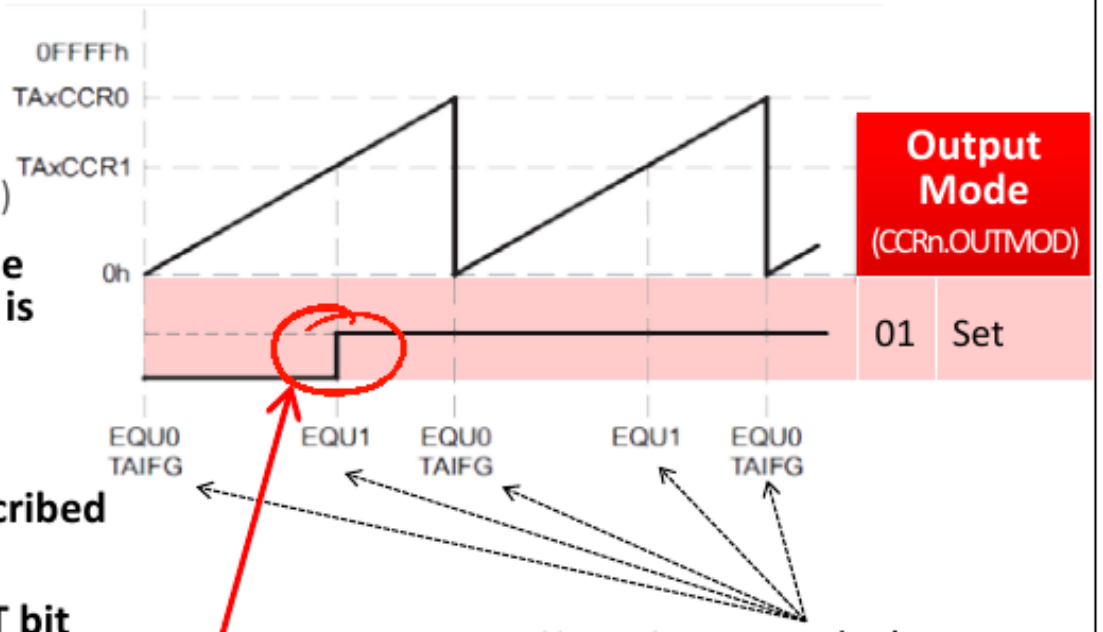
EQU: When TAR = CCR



- Nomenclature used in MSP430 User's Guide
- EQU0 and EQU1 are names for when CCR0 and CCR1 compare events occur (e.g. CCR1 = TAR)
- Similar EQUn events exist for each CCR register
- TAIFG is the generic timer interrupt whenever the count (in TAR) goes to zero

Timer CCR (Compare) Output Mode 01

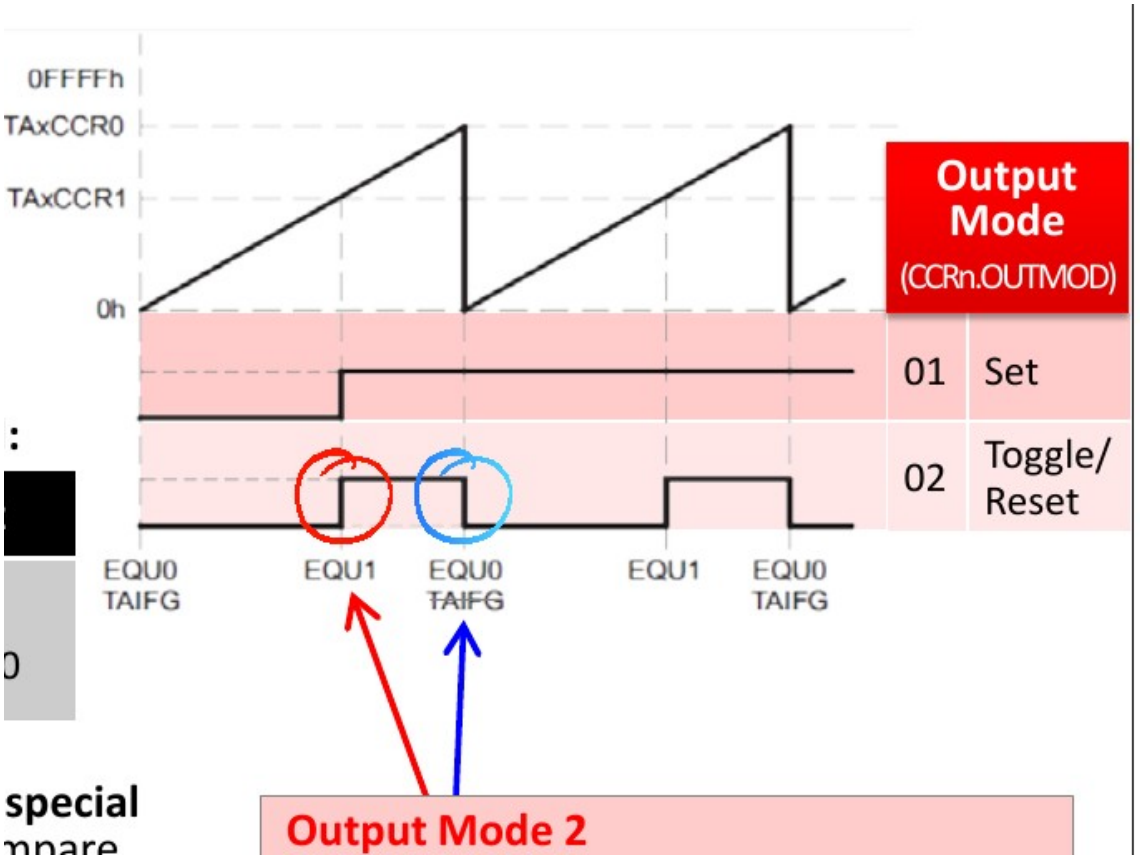
- **Each CCR has it's own signal (e.g. TA0.1)**
  - Input for capture (CCI)
  - Output for compare (OUT)
- **Used as output, the value in register bit CCRn.OUT is routed to TA0.n**
- **Value of OUT is affected by Output Mode (CCRn.OUTMOD) as described over the next few slides**
- **If OUTMOD=0, then OUT bit (and hence the signal) is under software control**



Note: Interrupts don't vary with OUTMOD, only the OUTput signal changes

Output Mode 1

- **OUTMOD = 01 is called "Set"**
- **This means that OUT (e.g. TA0.1) is set on EQU1**
- **That is, whenever TAR=CCR1**



OUTMOD = 2 ("Toggle/Reset" mode)

OutputMode 2 is a bit more interesting than the previous output modes. Notice how this mode is called "Toggle/Reset". Each of these names corresponds to a different event.

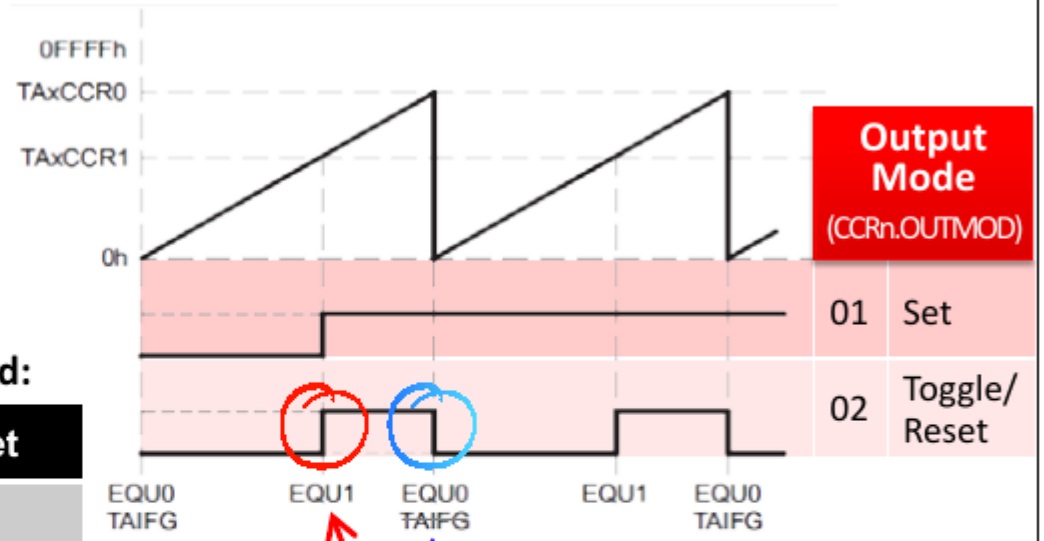
- Toggle - This means that OUT_n should be toggled whenever TAR=CCR_n
- Reset - This implies that OUT=0 (i.e. reset) whenever TAR=CCR0

In other words, when the OutputModes are defined by two names, the first one dictates the value of OUTn whenever the TAR=CCR_n (i.e. whenever EQU_n becomes true). The second name describes what happens to OUT_n whenever TAR=CCR0.

Timer CCR (Compare) Output Mode 02

- **OUT is actually affected by two events:**
  - **EQUn : when TAR=CCRn**
  - **EQU0 : when TAR=CCR0**
- **In other words, the two events are CCRnIFG and CCR0IFG, respectively**
- **Output Mode 02 is called:**

Toggle		Reset
on EQUn		on CCR0



- **As stated earlier, CCR0 is special** It affects all other CCR compare outputs in this same way
- **Note:** In this example, EQU0 and TAIFG happen at the same time; but ~~TAIFG~~ does not affect OUT

Output Mode 2

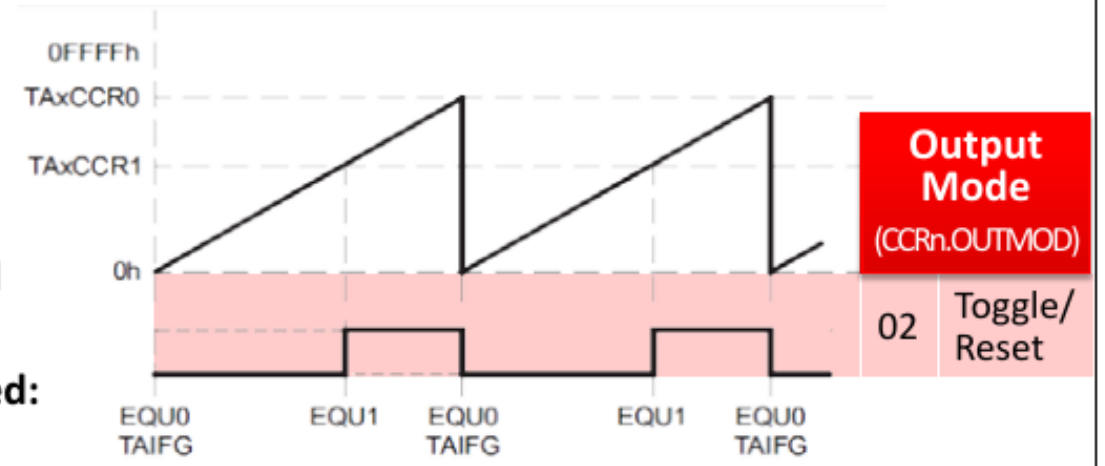
- **OUTMOD = 02 is called "Toggle/Reset"**
- **This means that OUT (e.g. TA0.1) is Toggled upon EQU1**
- **And Reset on EQU0 (i.e. CCR0 match)**

Putting this out on a GPIO pin ...

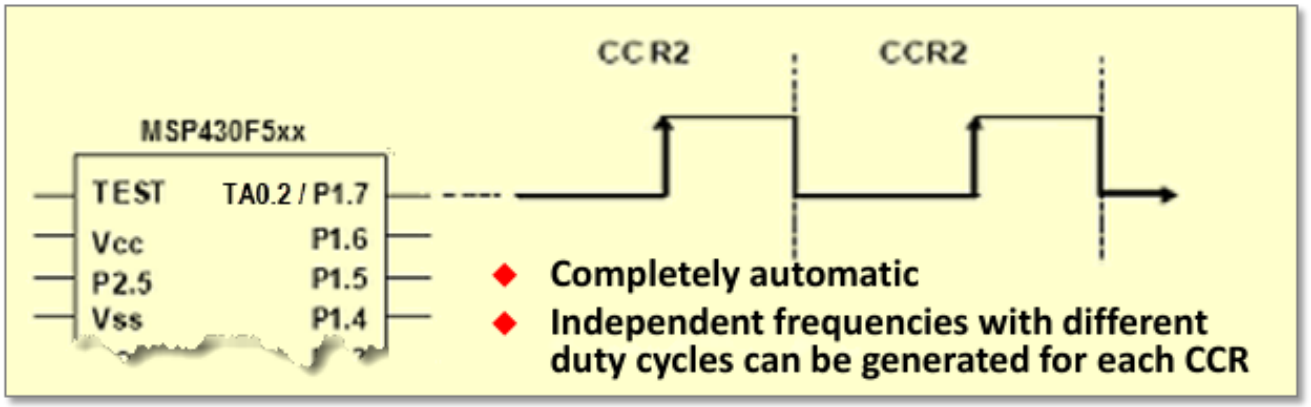
By showing both OUTMOD=1 and OUTMOD=2 in the same diagram, you can see how the value of OUT n can be very different depending upon the OutputMode selected.

Timer CCR (Compare) Output Mode 02

- OUT is actually affected by two events:
  - EQUn : when TAR=CCRn
  - EQU0 : when TAR=CCR0
- In other words, the two events are CCRnIFG and CCR0IFG, respectively
- Output Mode 02 is called:
  Toggle/Reset



Here's an example of routine TA0.2 (i.e. OUT2) to a GPIO pin:



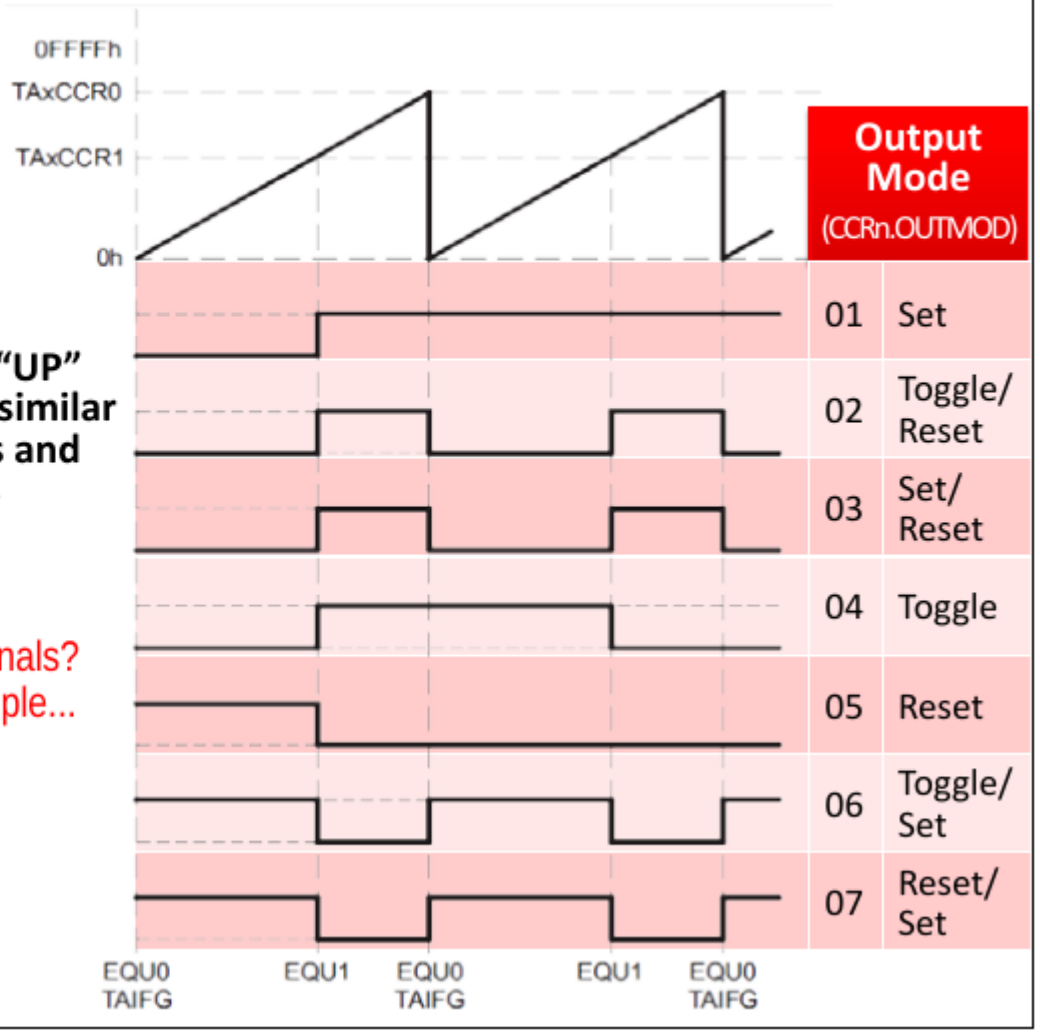
- Completely automatic
- Independent frequencies with different duty cycles can be generated for each CCR

Looking at all the Output Modes…

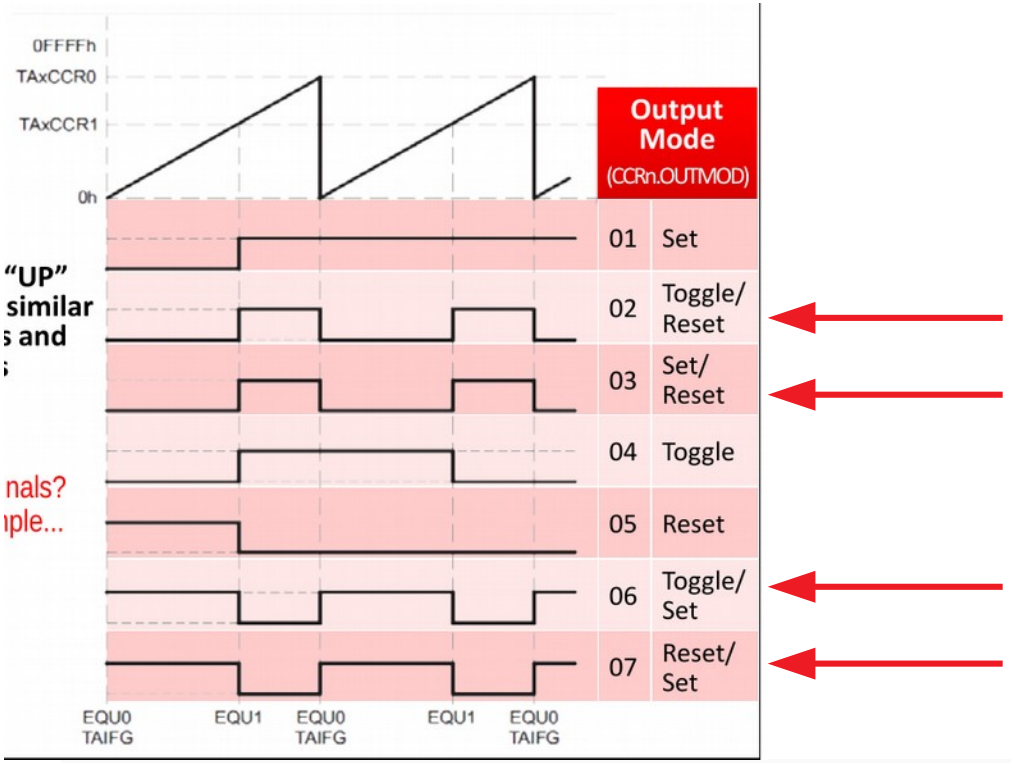
Capture "Output Modes" Summary

- Use different OUTMOD settings to create various signal patterns
- Output modes 2, 3, 6, and 7 are not useful for output unit 0 because EQUn = EQU0
- This summary is for the "UP" mode. User's Guide has similar diagrams for Continuous and UpDown counter modes

Do these look like PWM signals?
Here's a simple PWM example...



Output Mode (CCRn.OUTMOD)	
01	Set
02	Toggle/ Reset
03	Set/ Reset
04	Toggle
05	Reset
06	Toggle/ Set
07	Reset/ Set



Point of Clarification – Only use modes 1, 4, and 5 for CCR0

The second bullet, in the diagram above, states that four of the Output Modes (2, 3, 6, and 7) are not useful when you are working with CCR0.

Why are they not useful?

All four of these OutputModes include two actions:

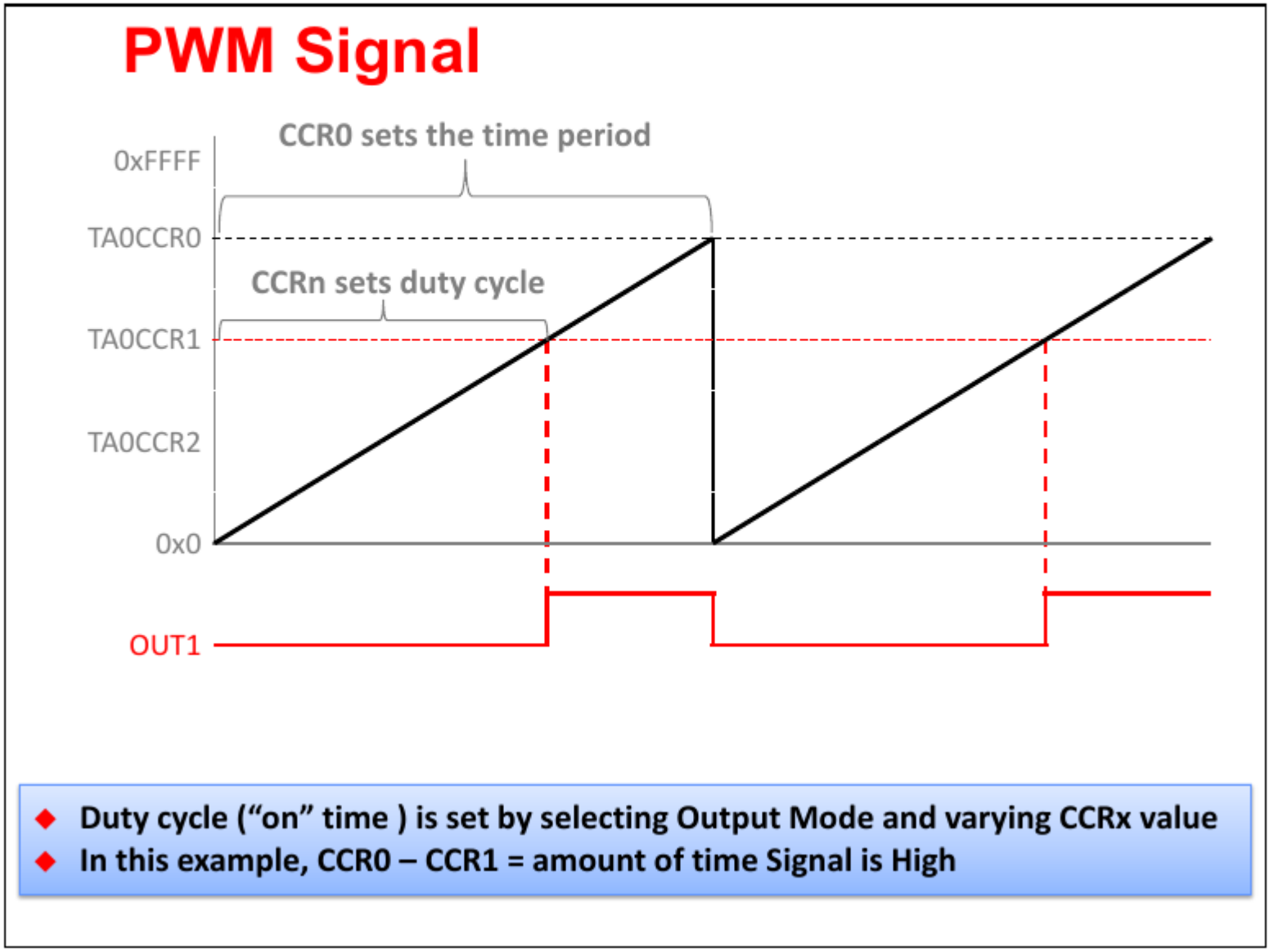
- One action when: CCRn=TAR
- A second action when: CCR0=TAR

In this case, though, CCRn = CCR0. That means these modes could be trying to change OUT0 in two different ways at the same time.

Bottom Line: When using CCR0, only set OUTMOD to 0, 1, 4, or 5.

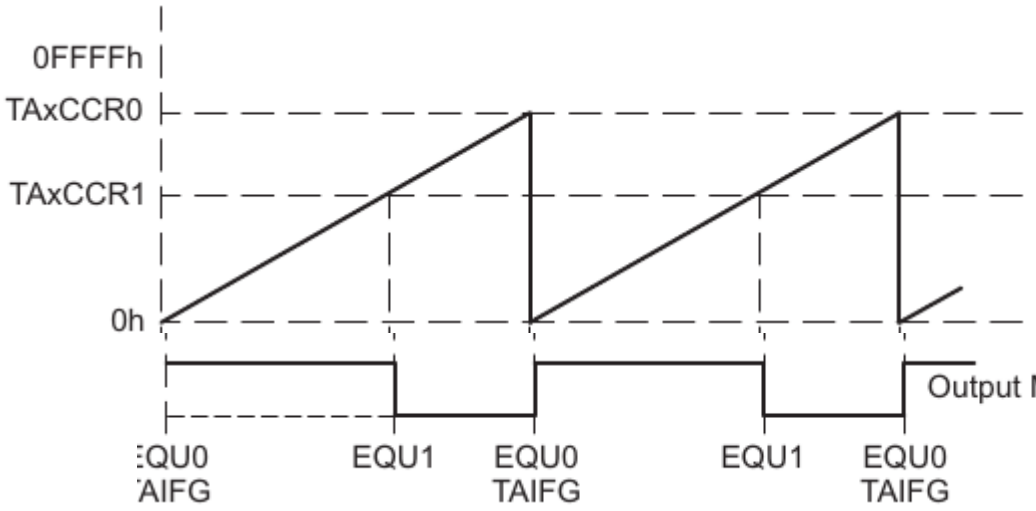
PWM, or pulse-width modulation, is commonly used to control the amount of energy going into a system. For example, by making the pulse widths longer, more energy is supplied to the system.

Looking again at the previous example where OUTMOD = 2, we can see that by changing the difference between the values of CCR0 and CCRn we can set the width of OUTn.



```
//******************************************************************************
//  MSP430FR24xx Demo - Timer0_A3, PWM TA0.1-2, Up Mode, DCO SMCLK
//
//  Description: This program generates two PWM outputs on P1.1,P1.2 using
//  Timer0_A configured for up mode. The value in CCR0, 1000-1, defines the PWM
//  period and the values in CCR1 and CCR2 the PWM duty cycles. Using ~1MHz
//  SMCLK as TACLK, the timer period is ~1ms with a 75% duty cycle on P1.1
//  and 25% on P1.2.
//  ACLK = default REFO ~32768Hz, MCLK = SMCLK = default DCODIV ~1MHz
//
//
//           MSP430FR2433
//         ---------------
//     /|\|               |
//      | |               |
//      --|RST            |
//        |               |
//        |     P1.1/TA0.1|--> CCR1 - 75% PWM
//        |     P1.2/TA0.2|--> CCR2 - 25% PWM
//
//
//  Wei Zhao Texas Instruments Inc. Jan 2014
//  Modified for Energia  H Watson 20180714
//  msp430fr243x_ta0_16.c
//******************************************************************************
```

```c
//  msp430fr243x_ta0_16.c
//*****************************************************************************
#include <msp430.h>

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;                 // Stop WDT

    P1DIR |= BIT1 | BIT2;                     // P1.1 and P1.2 output
    P1SEL1 |= BIT1 | BIT2;                    // P1.1 and P1.2 options select

    // Disable the GPIO power-on default high-impedance mode to activate
    // previously configured port settings
    PM5CTL0 &= ~LOCKLPM5;

    TA0CCR0 = 1000-1;                         // PWM Period
    TA0CCTL1 = OUTMOD_7;                      // CCR1 reset/set
    TA0CCR1 = 750;                            // CCR1 PWM duty cycle
    TA0CCTL2 = OUTMOD_7;                      // CCR2 reset/set
    TA0CCR2 = 250;                            // CCR2 PWM duty cycle
    TA0CTL = TASSEL__SMCLK | MC__UP | TACLR;  // SMCLK, up mode, clear TAR

    __bis_SR_register(LPM0_bits);             // Enter LPM0
    __no_operation();                         // For debugger
}
```



P1.0/UCB0STE/TA0CLK/A0/Veref+
P1.1/UCB0CLK/TA0.1/A1
P1.2/UCB0SIMO/UCB0SDA/TA0.2/A2/Veref-
P1.3/UCB0SOMI/UCB0SCL/MCLK/A3
P1.4/UCA0TXD/UCA0SIMO/TA1.2/TCK/A4/VREF+
P1.5/UCA0RXD/UCA0SOMI/TA1.1/TMS/A5
P1.6/UCA0CLK/TA1CLK/TDI/TCLK/A6
P1.7/UCA0STE/SMCLK/TDO/A7

```
45  //******************************************************************************
46  //  MSP430FR243x Demo - Timer1_A3, PWM TA1.1-2, Up Mode, DCO SMCLK
47  //
48  //  Description: This program generates two PWM outputs on P1.4,P1.5 using
49  //  Timer1_A configured for up mode. The value in CCR0, 1000-1, defines the PWM
50  //  period and the values in CCR1 and CCR2 the PWM duty cycles. Using ~1MHz
51  //  SMCLK as TACLK, the timer period is ~1ms with a 75% duty cycle on P1.4
52  //  and 25% on P1.5.
53  //  ACLK = n/a, SMCLK = MCLK = TACLK = 1MHz
54  //
55  //
56  //           MSP430FR2433
57  //         ---------------
58  //     /|\|               |
59  //      | |               |
60  //      --|RST            |
61  //        |               |
62  //        |     P1.5/TA1.1|--> CCR1 - 75% PWM
63  //        |     P1.4/TA1.2|--> CCR2 - 25% PWM
64  //
65  //
66  //  Ling Zhu Texas Instruments Inc. Feb 2015
67  //  msp430fr243x_ta1_16.c
68  //******************************************************************************
```


P1.0/UCB0STE/TA0CLK/A0/Veref+
P1.1/UCB0CLK/TA0.1/A1
P1.2/UCB0SIMO/UCB0SDA/TA0.2/A2/Veref-
P1.3/UCB0SOMI/UCB0SCL/MCLK/A3
P1.4/UCA0TXD/UCA0SIMO/TA1.2/TCK/A4/VREF+
P1.5/UCA0RXD/UCA0SOMI/TA1.1/TMS/A5
P1.6/UCA0CLK/TA1CLK/TDI/TCLK/A6
P1.7/UCA0STE/SMCLK/TDO/A7

```
//  Modified for Energia   H Watson 20180714
//******************************************************************************
#include <msp430.h>

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;                 // Stop WDT

    // Configure GPIO
    P1DIR |= BIT4 + BIT5;
    P1SEL1 |= BIT4 + BIT5;

    // Disable the GPIO power-on default high-impedance mode to activate
    // previously configured port settings
    PM5CTL0 &= ~LOCKLPM5;

    TA1CCR0 = 1000-1;                         // PWM Period
    TA1CCTL1 = OUTMOD_7;                      // CCR1 reset/set
    TA1CCR1 = 750;                            // CCR1 PWM duty cycle
    TA1CCTL2 = OUTMOD_7;                      // CCR2 reset/set
    TA1CCR2 = 250;                            // CCR2 PWM duty cycle
    TA1CTL = TASSEL__SMCLK | MC__UP | TACLR;  // SMCLK, up mode, clear TAR

    __bis_SR_register(LPM0_bits);             // Enter LPM0
    __no_operation();                         // For debugger
}
```

```
45  //******************************************************************************
46  //  MSP430FR243x Demo - Timer0_A3, PWM TA0.1-2, Up/Down Mode, 32kHz ACLK
47  //
48  //  Description: This program generates two PWM outputs on P1.2,3 using
49  //  Timer_A configured for up/down mode. The value in TACCR0, 128, defines the
50  //  PWM period/2 and the values in TACCR1 and TACCR2 the PWM duty cycles. Using
51  //  32kHz ACLK as TACLK, the timer period is 7.8ms with a 75% duty cycle on
52  //  P1.1 and 25% on P1.2. Normal operating mode is LPM3.
53  //  ACLK = TACLK = REFO ~ 32768Hz, MCLK = default DCO ~1.2MHz
54  //  //* External watch crystal on XIN XOUT is required for ACLK *//
55  //
56  //           MSP430FR2433
57  //         ---------------
58  //     /|\|               |
59  //      | |               |
60  //      --|RST            |
61  //        |               |
62  //        |     P1.1/TA0.1|--> CCR1 - 75% PWM
63  //        |     P1.2/TA0.2|--> CCR2 - 25% PWM
64  //
65  //  Ling Zhu Texas Instruments Inc. Aug 2015
66  //  Modified for Energia - H Watson 20180714
67  //   msp430fr243x_ta0_20.c
68  //******************************************************************************
```

```c
//   msp430fr243x_ta0_20.c
//******************************************************************************
#include <msp430.h>

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;                 // Stop WDT
    P1DIR |= BIT1 | BIT2;                     // P1.1 and P1.2 output
    P1SEL1 |= BIT1 | BIT2;                    // P1.1 and P1.2 options select

    // Disable the GPIO power-on default high-impedance mode to activate
    // previously configured port settings
    PM5CTL0 &= ~LOCKLPM5;

    TA0CCR0 = 128;                            // PWM Period/2
    TA0CCTL1 = OUTMOD_6;                      // TACCR1 toggle/set
    TA0CCR1 = 32;                             // TACCR1 PWM duty cycle
    TA0CCTL2 = OUTMOD_6;                      // TACCR2 toggle/set
    TA0CCR2 = 96;                             // TACCR2 PWM duty cycle
    TA0CTL = TASSEL_1 | MC_3;                 // ACLK, up-down mode

    __bis_SR_register(LPM3_bits);             // Enter LPM3

    return 0;
}
```

9-8	TASSEL	RW	0h	Timer_A clock source select 00b = TAxCLK 01b = ACLK 10b = SMCLK 11b = INCLK
5-4	MC	RW	0h	Mode control. Setting MC = 0 when Timer_A is not in use conserves power. 00b = Stop mode: Timer is halted 01b = Up mode: Timer counts up to TAxCCR0 10b = Continuous mode: Timer counts up to 0FFFFh 11b = Up/Down mode: Timer counts up to TAxCCR0 then down to 0000h

```c
//   msp430fr243x_ta0_20.c
//******************************************************************************
#include <msp430.h>

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;                 // Stop WDT
    P1DIR |= BIT1 | BIT2;                     // P1.1 and P1.2 output
    P1SEL1 |= BIT1 | BIT2;                    // P1.1 and P1.2 options select

    // Disable the GPIO power-on default high-impedance mode to activate
    // previously configured port settings
    PM5CTL0 &= ~LOCKLPM5;

    TA0CCR0 = 128;                            // PWM Period/2
    TA0CCTL1 = OUTMOD_6;                      // TACCR1 toggle/set
    TA0CCR1 = 32;                             // TACCR1 PWM duty cycle
    TA0CCTL2 = OUTMOD_6;                      // TACCR2 toggle/set
    TA0CCR2 = 96;                             // TACCR2 PWM duty cycle
    TA0CTL = TASSEL_1 | MC_3;                 // ACLK, up-down mode

    __bis_SR_register(LPM3_bits);             // Enter LPM3

    return 0;
}
```

Figure 12-14. Output Example – Timer in Up/Down Mode