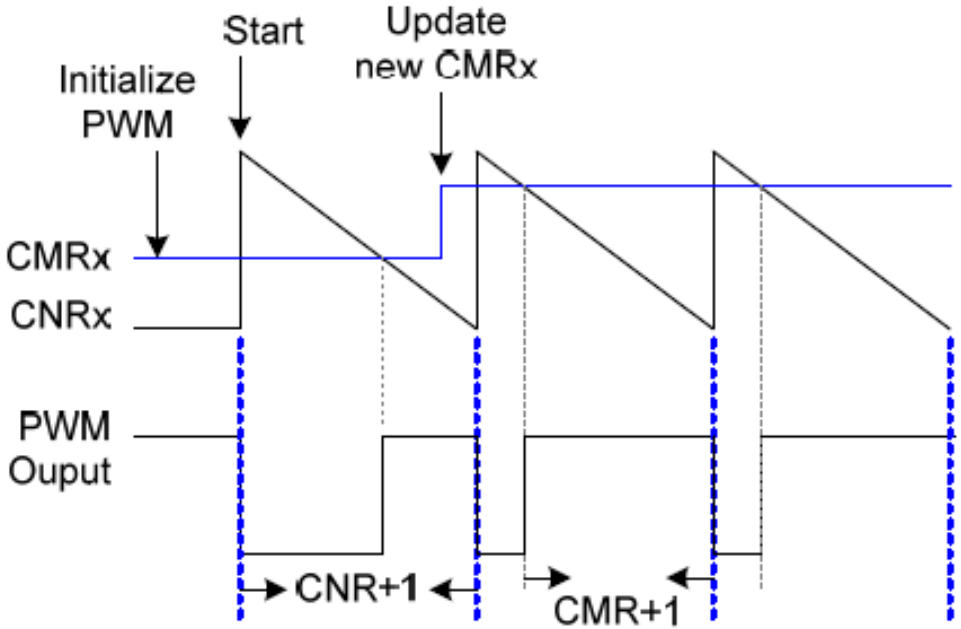
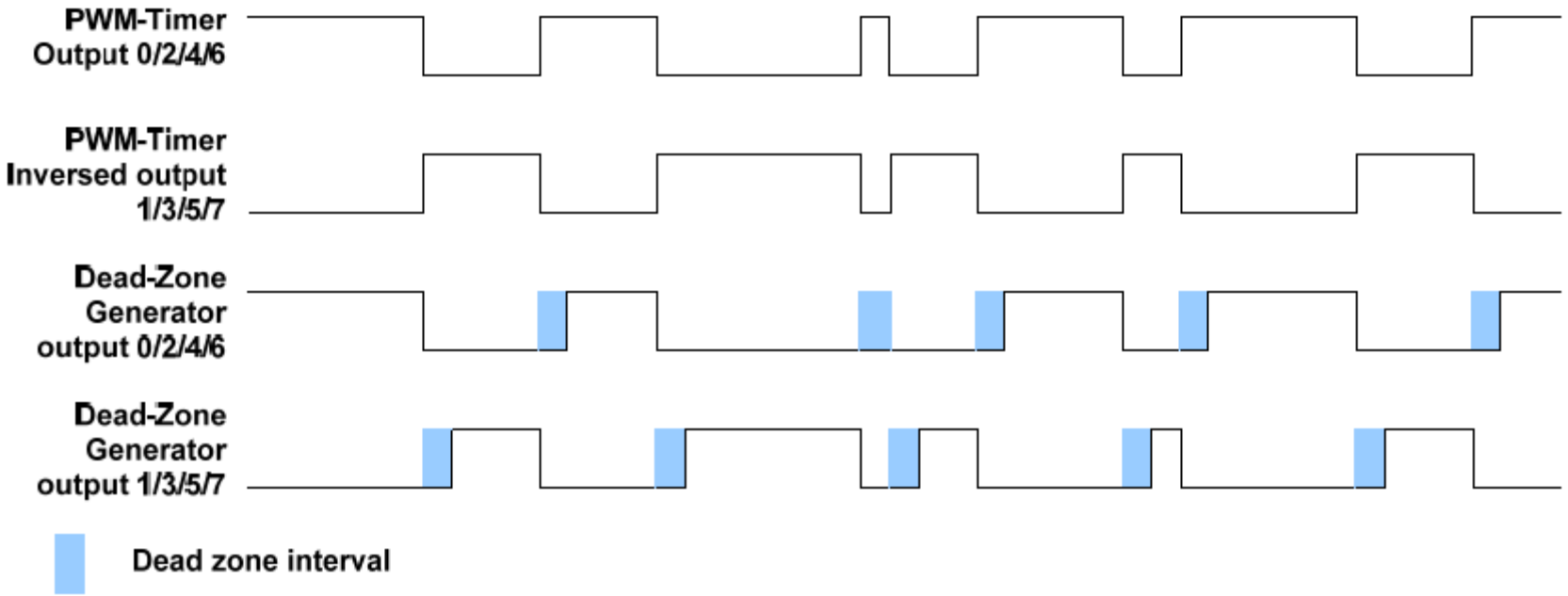
# PWM Generator and Capture Timer (PWM)

- 2 groups of PWM supports total 4 sets of PWM Generators which can be configured as 8 independent PWM outputs, PWM0~PWM7, or as 4 complementary PWM pairs, (PWM0, PWM1), (PWM2, PWM3), (PWM4, PWM5) and (PWM6, PWM7) with 4 programmable dead-zone generators
- The PWM generators can be configured as one-shot mode to produce only one PWM cycle signal or auto-reload mode to output PWM waveform continuously
- PWM Interrupt request synchronized with PWM period



# PWM : PWM function

- PWM controller is implemented with Dead Zone generator. They are built for power device protection. This function generates a programmable time gap to delay PWM rising output. User can program PPRx.DZI to determine the Dead Zone interval.
- When PCR.DZEN01 is set, PWM0 and PWM1 perform complementary PWM paired function; the paired PWM period, duty and dead-time are determined by PWM0 timer and Dead-zone generator 0. Similarly, the complementary PWM pairs of (PWM2, PWM3), (PWM4, PWM5) and (PWM6, PWM7) are controlled by PWM2, PWM4 and PWM6 timers and Dead-zone generator 2, 4 and 6, respectively



# PWM : Capture Function

- The alternate feature of the PWM-timer is digital input Capture function (PWM output pin is switched as capture input mode)

- The Capture0 and PWM0 share one timer which is included in PWM0 and the Capture1 and PWM1 share PWM1 timer, and etc. (Therefore user must setup the PWM-timer before enable Capture feature)

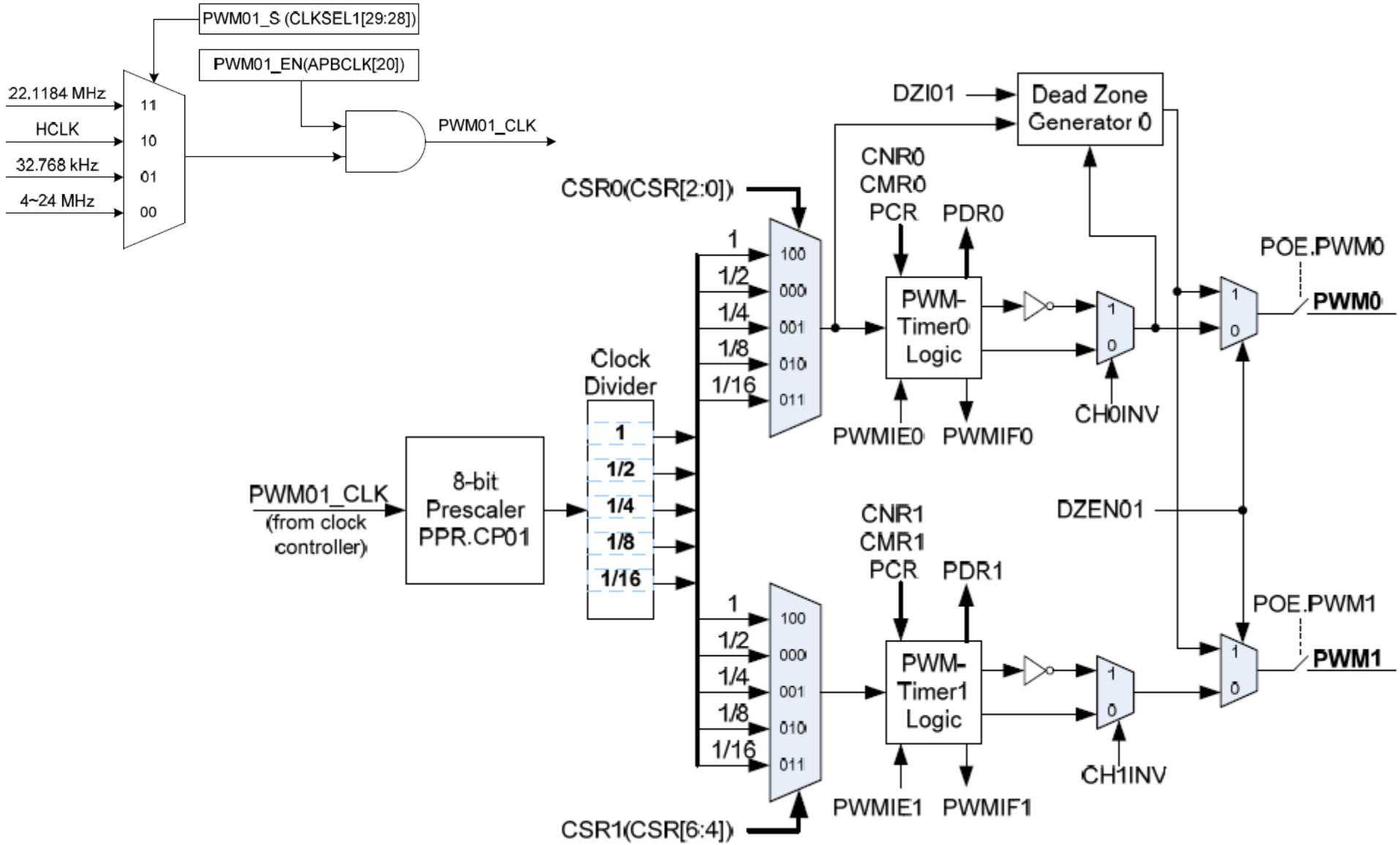
- the capture always latched PWM-counter to Capture Rising Latch Register (CRLR) when input channel has a rising transition and latched PWM-counter to Capture Falling Latch Register (CFLR) when input channel has a falling transition
- Capture channel 0 interrupt is programmable by setting CCR0.CRL_IE0[1] (Rising latch Interrupt enable) and CCR0.CFL_IE0[2]] (Falling latch Interrupt enable) to decide the condition of interrupt occur. Capture channel 1 has the same feature by setting CCR0.CRL_IE1[17] and CCR0.CFL_IE1[18]. And capture channel 2 to channel 3 on each group have the same feature by setting the corresponding control bits in CCR2. For each group, whenever Capture issues Interrupt 0/1/2/3, the PWM counter 0/1/2/3 will be reload at this moment.

# PWM : Capture Function

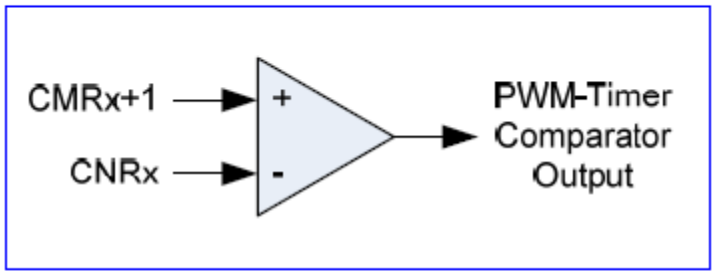
- The maximum captured frequency that PWM can capture is confined by the capture interrupt latency.
- When capture interrupt occurred, software will do at least three steps:
    - Read PIIR to get interrupt source
    - Read CRLRx/CFLRx(x=0~3) to get capture value
    - and finally write 1 to clear PIIR to zero.
- If interrupt latency will take time T0 to finish, the capture signal mustn't transition during this interval (T0). In this case, the maximum capture frequency will be 1/T0.

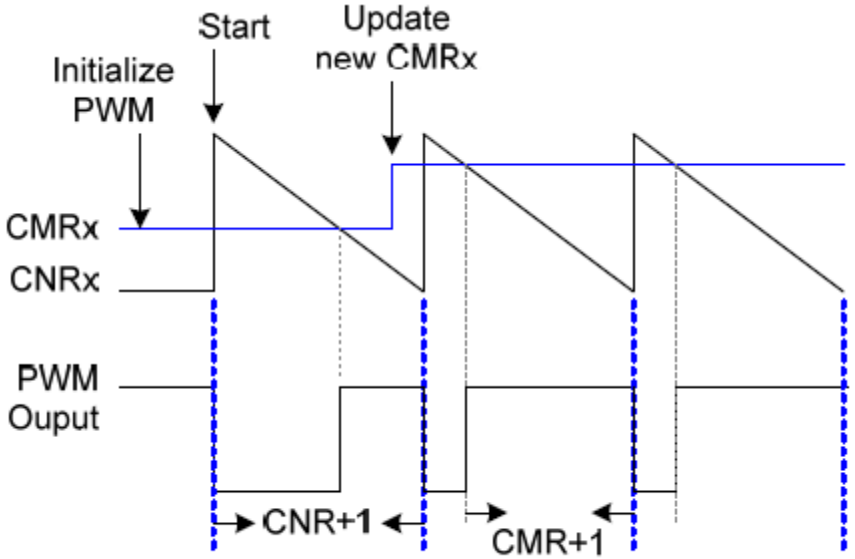
For example:
HCLK = 50 MHz, PWM_CLK = 25 MHz, Interrupt latency is 900 ns
So the maximum capture frequency will is 1/900ns ≈ 1000 kHz

# PWM : Block Diagram



# PWM : PWM-Timer Operation



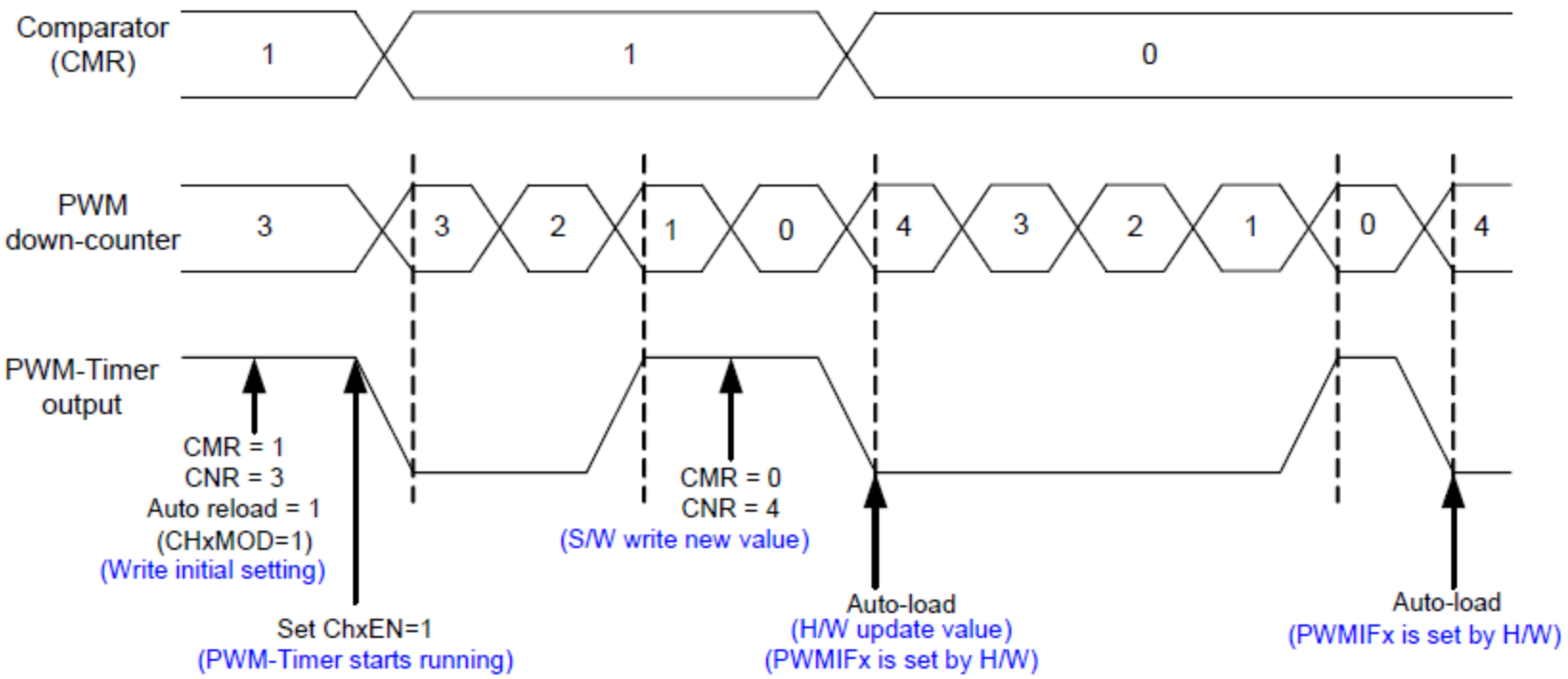




Note: x= 0~3.

- PWM frequency = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; where xy, could be 01, 23, 45 or 67, depends on selected PWM channel.
- Duty ratio = (CMR+1)/(CNR+1)
- CMR >= CNR: PWM output is always high
- CMR < CNR: PWM low width= (CNR-CMR) unit
- PWM high width = (CMR+1) unit
- CMR = 0: PWM low width = (CNR) unit; PWM high width = 1 unit

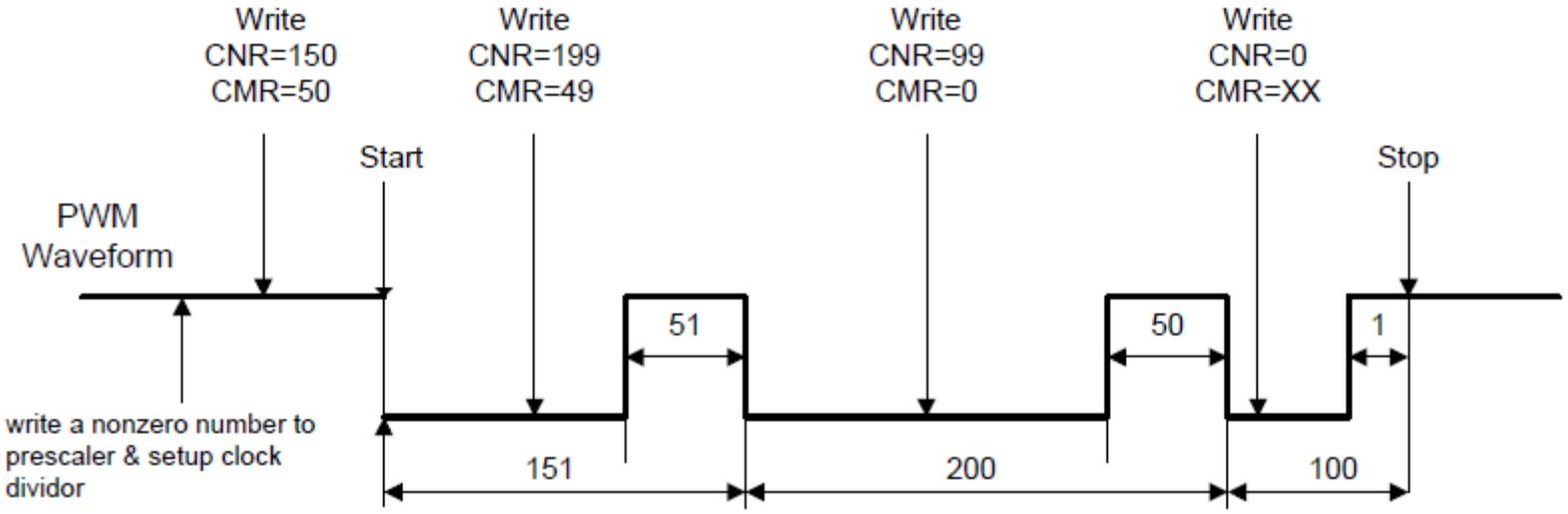
Unit = one PWM clock cycle.

# PWM : PWM-Timer Operation Timing



# PWM : PWM-Timer Operation

- PWM Timers have double buffering function the reload value is updated at the start of next period without affecting current timer operation.
- The PWM counter value can be written into CNRx and current PWM counter value can be read from PDRx.
- PWM0 will operate at one-shot mode if CH0MOD bit is set to 0, and operate at auto-reload mode if CH0MOD bit is set to 1.
- It is recommend that switch PWM0 operating mode before set CH0EN bit to 1 to enable PWM0 counter start running because the content of CNR0 and CMR0 will be cleared to zero to reset the PWM0 period and duty setting when PWM0 operating mode is changed.



# PWM : Timer Start Procedure

1. Setup clock source divider select register (CSR)
2. Setup prescaler (PPR)
3. Setup inverter on/off, dead zone generator on/off, auto-reload/one-shot mode and Stop PWM-timer (PCR)
4. Setup comparator register (CMR) for setting PWM duty.
5. Setup PWM down-counter register (CNR) for setting PWM period.
6. Setup interrupt enable register (PIER) (option)
7. Setup corresponding GPIO pins as PWM function (enable POE and disable CAPENR) for the corresponding PWM channel.
8. Enable PWM timer start running (Set CHxEN = 1 in PCR)

The value of CNR0 will reload to PWM0 counter when it down count reaches zero. If CNR0 is set to zero, PWM0 counter will be held. PWM1~PWM7 performs the same function as PWM0.

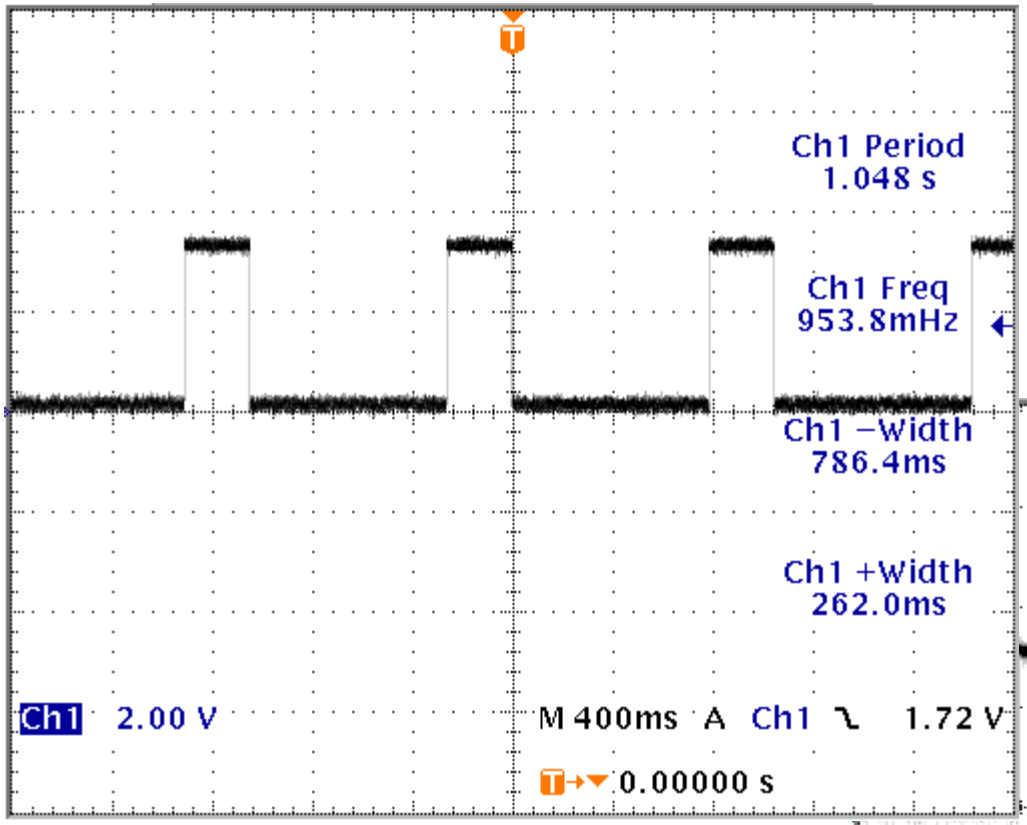
# PWM : Program Example

```
void InitPWM1(void)
{
  /* Step 1. GPIO initial */
  SYS->GPAMFP.PWM1_AD14=1;

  /* Step 2. Enable and Select PWM clock source*/
  SYSCLK->APBCLK.PWM01_EN = 1;  // Enable PWM clock
  SYSCLK->CLKSEL1.PWM01_S = 0;  // Select 12Mhz for PWM clock source

  PWMA->PPR.CP01=11;  //Prescaler 0~255, Setting 0 to stop output clock
  PWMA->CSR.CSR1=3;   //clock divider->0:/2, 1:/4, 2:/8, 3:/16, 4:/1

  /* Step 3. Select PWM Operation mode */
  //PWM1
  PWMA->PCR.CH1MOD=1; //0:One-shot mode, 1:Auto-load mode
  // CNR and CMR will be auto-cleared after setting CH0MOD from 0 to 1.
  // PWM frequency = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)];
  PWMA->CNR1=0xFFFF;
  PWMA->CMR1=0x3FFF;

  PWMA->PCR.CH1INV=0; //Inverter->0:off, 1:on
  PWMA->PCR.CH1EN=1;  //PWM function->0:Disable, 1:Enable
  PWMA->POE.PWM1=1;   //Output to pin->0:Diasble, 1:Enable
}
```

# PWM : Program Example

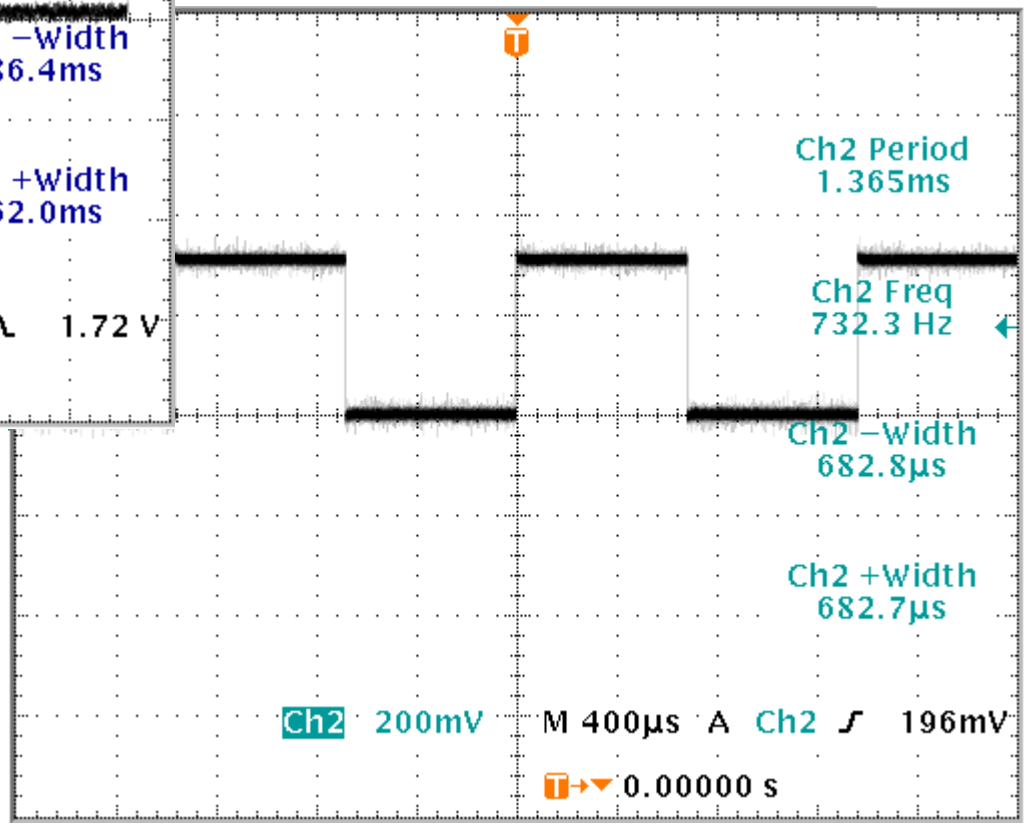
```
void InitPWM2(void)
{
  /* Step 1. GPIO initial */
  SYS->GPAMFP.PWM2_AD15=1;

  /* Step 2. Enable and Select PWM clock source*/
  SYSCLK->APBCLK.PWM23_EN = 1;  // Enable PWM clock
  SYSCLK->CLKSEL1.PWM23_S = 0;  // Select 12Mhz for PWM clock source

  PWMA->PPR.CP23=1;      //Prescaler 0~255, Setting 0 to stop output clock
  PWMA->CSR.CSR2=4;      //clock divider->0:/2, 1:/4, 2:/8, 3:/16, 4:/1

  /* Step 3. Select PWM Operation mode */
  //PWM2
  PWMA->PCR.CH2MOD=1;    //0:One-shot mode, 1:Auto-load mode
  //CNR and CMR will be auto-cleared after setting CH0MOD form 0 to 1.
  // PWM frequency = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)];
  PWMA->CNR2=0x1FFF;
  PWMA->CMR2=0x0FFF;

  PWMA->PCR.CH2INV=0;    //Inverter->0:off, 1:on
  PWMA->PCR.CH2EN=1;     //PWM function->0:Disable, 1:Enable
  PWMA->POE.PWM2=1;      //Output to pin->0:Diasble, 1:Enable
}
```

# PWM : Program Example




Ch1: GPA13=PWM1

Ch2: GPA14=PWM2



# PWM : Re-Start Procedure in Single-shot mode

- As PWM0 operate at one-shot mode, CMR0 and CNR0 should be written first and then set CH0EN bit to 1 to enable PWM0 counter start running. After PWM0 counter down count from CNR0 value to zero, CNR0 and CMR0 will be cleared to zero by hardware and PWM counter will be held. Software need to write new CMR0 and CNR0 value to set next one-shot period and duty. When re-start next one-shot operation, the CMR0 should be written first because PWM0 counter will auto re-start counting when CNR0 is written an non-zero value.

1. Setup comparator register (CMR) for setting PWM duty.

2. Setup PWM down-counter register (CNR) for setting PWM period. After setup CNR, PWM wave will be generated.

# PWM : Stop Procedure

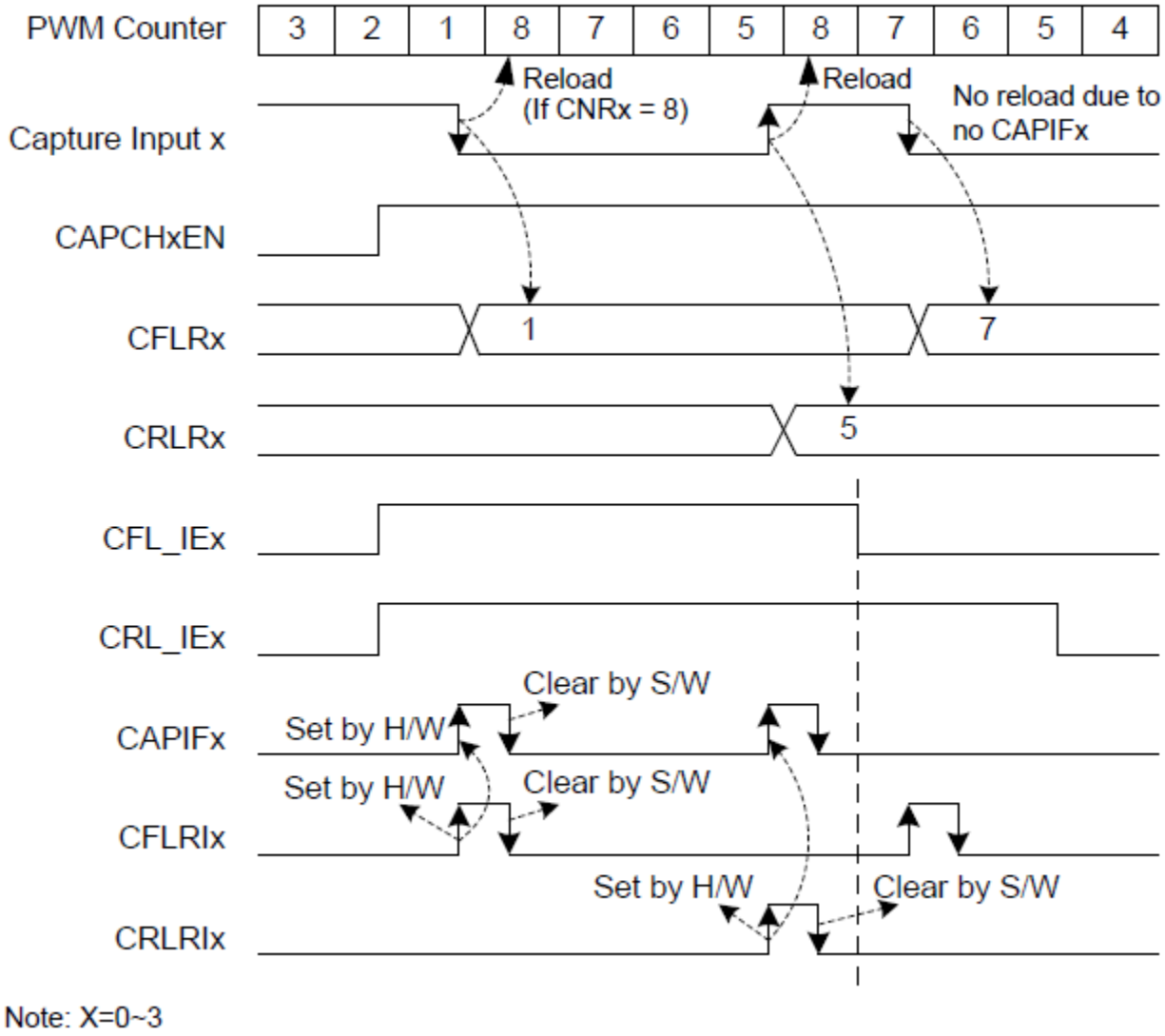
**Method 1:**

Set 16-bit down counter (CNR) as 0, and monitor PDR (current value of 16-bit down-counter). When PDR reaches to 0, disable PWM-Timer (CHxEN in PCR). ***(Recommended)***

**Method 2:**

Set 16-bit down counter (CNR) as 0. When interrupt request happened, disable PWM-Timer (CHxEN in PCR). ***(Recommended)***

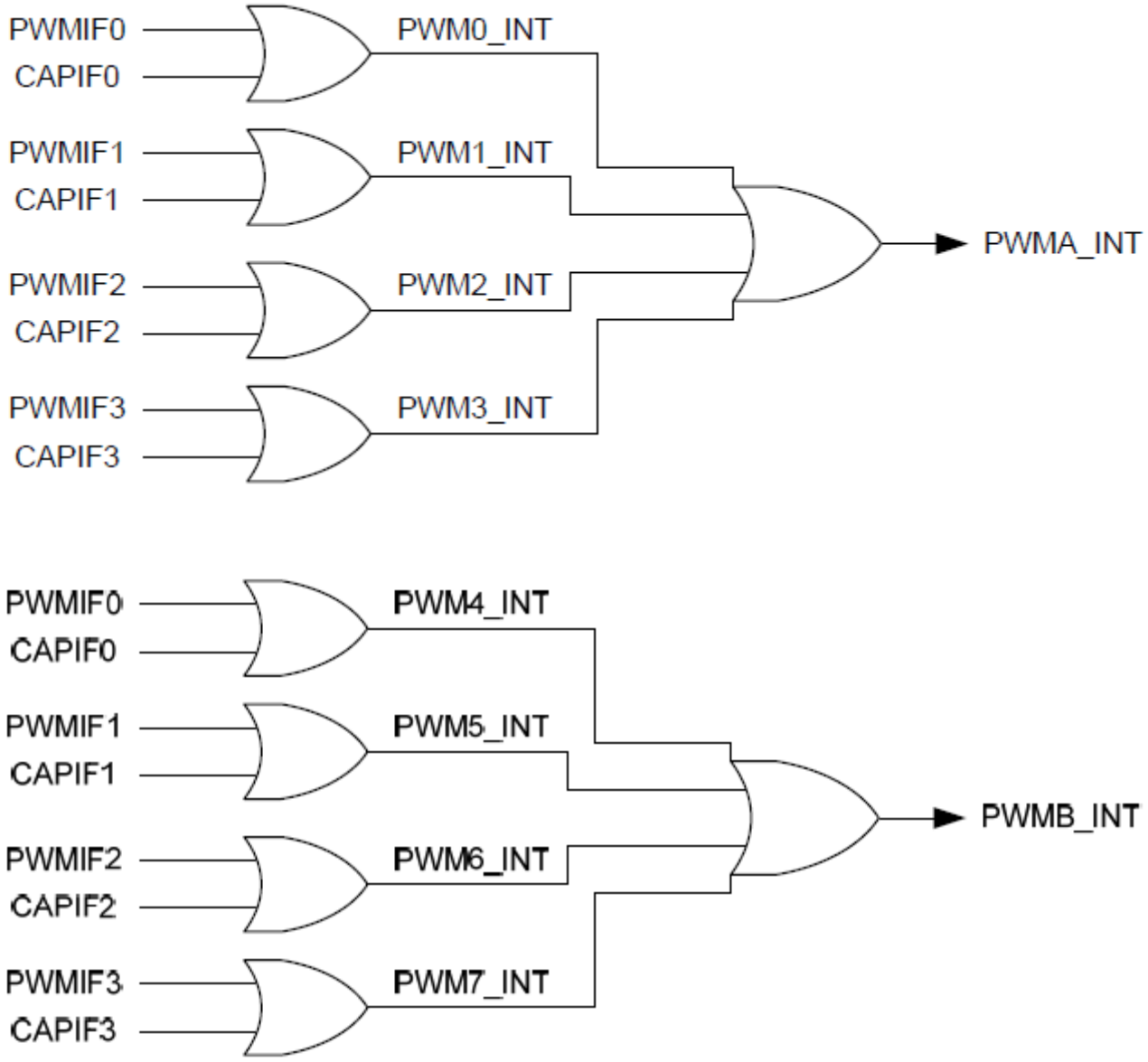
# PWM : Capture Operation




**CNR is 8**:

1. The **PWM counter** will be reloaded with **CNRx** when a capture interrupt flag (**CAPIFx**) is set.
2. The channel low pulse width is (**CNR** + 1 - **CRLR**).
3. The channel high pulse width is (**CNR** + 1 - **CFLR**).

# PWM : Interrupt Architecture



# PWM : Capture Start Procedure

1. Setup clock source divider select register (**CSR**)
2. Setup prescaler (**PPR**)
3. Setup channel enabled, rising/falling interrupt enable and input signal inverter on/off (**CCR0**, **CCR2**)
4. Setup auto-reload mode, Edge-aligned type and Stop PWM-timer (**PCR**)
5. Setup PWM down-counter (**CNR**)
6. Enable PWM timer start running (Set **CHxEN** = 1 in **PCR**)
7. Setup corresponding **GPIO** pins as capture function (disable **POE** and enable **CAPENR**) for the corresponding **PWM channel**.

# PWM : Program Example

```
void InitCapture(void)
{
  /* Step 1. GPIO initial */
  SYS->GPAMFP.PWM0_AD13=1;  // System Manager Control Registers

  /* Step 2. Enable and Select PWM clock source*/
  SYSCLK->APBCLK.PWM01_EN = 1;//Enable PWM clock
  SYSCLK->CLKSEL1.PWM01_S = 0;//Select 12Mhz for PWM clock source
  // 0:12MHz, 1:32.768 kHz, 2:HCLK, 3:22.1184 MHz
  PWMA->PPR.CP01=11;  //Prescaler 0~255, Setting 0 to stop output clock
  PWMA->CSR.CSR0=4;   //clock divider->0:/2, 1:/4, 2:/8, 3:/16, 4:/1

  /* Step 3. Select PWM Operation mode */
  PWMA->PCR.CH0MOD=1;     //0:One-shot mode, 1:Auto-load mode
  //CNR and CMR will be auto-cleared after setting CH0MOD form 0 to 1.
  PWMA->CNR0=PWM_CNR;     //Set Reload register
  PWMA->CAPENR=1;         //Enable Capture function pin
  PWMA->CCR0.CAPCH0EN=1;  //Enable Capture function

  /* Step 4. Set PWM Interrupt */
  PWMA->CCR0.CRL_IE0=1;   //Enable Capture rising edge interrupt
  PWMA->CCR0.CFL_IE0=1;   //Enable Capture falling edge interrupt
  PWMA->PIER.PWMIE0=1;    //Enable PWM interrupt for down-counter equal zero.
  NVIC_EnableIRQ(PWMA_IRQn);  //enable PWM inturrupt

  /* Step 5. Enable PWM down counter*/
  PWMA->PCR.CH0EN=1;      //Enable PWM down counter
}
```

# PWM : Program Example

```
void PWMA_IRQHandler(void) // PWM interrupt subroutine
{
  if(PWMA->PIIR.PWMIF0) // if PWM0 down counter reaches zero->bit set
  {
    CaptureCounter++;//Delay (PWM_CNR+1) usec
    if(CaptureCounter==0)
    {
      //Overflow
    }
    PWMA->PIIR.PWMIF0=1;// write '1' to clear this bit to zero
  }
  if(PWMA->CCR0.CAPIF0) // if ch0 Capture interrupt
  {
    if(PWMA->CCR0.CFLRI0)//Calculate High Level
    { // if detect falling transition
      CaptureValue[0]=CaptureCounter*(PWM_CNR+1)+(PWM_CNR-PWMA->CFLR0);//usec
      CaptureCounter=0;
      PWMA->CCR0.CFLRI0=0;// write '0' to clear this bit to zero
    }
    if(PWMA->CCR0.CRLRI0)//Calculate Low Level
    { // if detect rising transition
      CaptureValue[1]=CaptureCounter*(PWM_CNR+1)+(PWM_CNR-PWMA->CRLR0);//usec
      CaptureCounter=0;
      PWMA->CCR0.CRLRI0=0;// write '0' to clear this bit to zero
    }
    PWMA->CCR0.CAPIF0=1;// write '1' to clear this bit to zero
  }
}
```

# PWM : Register Map

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| PWMA_BA = 0x4004_0000 (PWM group A)<br>PWMB_BA = 0x4014_0000 (PWM group B) | | | | |
| PPR | PWMA_BA+0x00 | R/W | PWM Group A Prescaler Register | 0x0000_0000 |
| | PWMB_BA+0x00 | R/W | PWM Group B Prescaler Register | 0x0000_0000 |
| CSR | PWMA_BA+0x04 | R/W | PWM Group A Clock Source Divider Select Register | 0x0000_0000 |
| | PWMB_BA+0x04 | R/W | PWM Group B Clock Source Divider Select Register | 0x0000_0000 |
| PCR | PWMA_BA+0x08 | R/W | PWM Group A Control Register | 0x0000_0000 |
| | PWMB_BA+0x08 | R/W | PWM Group B Control Register | 0x0000_0000 |
| CNR0 | PWMA_BA+0x0C | R/W | PWM Group A Counter Register 0 | 0x0000_0000 |
| | PWMB_BA+0x0C | R/W | PWM Group B Counter Register 0 | 0x0000_0000 |
| CMR0 | PWMA_BA+0x10 | R/W | PWM Group A Comparator Register 0 | 0x0000_0000 |
| | PWMB_BA+0x10 | R/W | PWM Group B Comparator Register 0 | 0x0000_0000 |
| PDR0 | PWMA_BA+0x14 | R | PWM Group A Data Register 0 | 0x0000_0000 |
| | PWMB_BA+0x14 | R | PWM Group B Data Register 0 | 0x0000_0000 |