Document Number:AN4560
Rev. 0, 8/2012

PWM Synchronization Using Kinetis Flextimers

by: Xianhu Gao
Automotive and Industrial Solutions Group

Contents

1 FlexTimer introduction

The FlexTimer (FTM) on Kinetis MCU is built upon a very simple timer, HCS08 Timer PWM Module (TPM), used for many years on Freescale 8-bit microcontrollers. But the FTM extends the functionality on input capture, output compare, and especially the generation of PWM signals to meet the demands of motor control, digital lighting solutions, and power conversion. However, it can be backward compatible with TPM by configuring the FTMx_MODE register.

The FTM module is powerful and flexible when used to generate PWM signals required in some applications. The users can get the desired control signals by changing the registers FTMx_MOD, FTMx_CNTIN, FTMx_CnV, FTMx_OUTMASK, FTMx_INVCTRL, and FTMx_SWOCTRL. But any writes to these registers will be latched in the write buffer first because of the hardware structure. Therefore, updating these registers requires a lot of attention.

This application note covers the whole details about the FTM synchronization, including the Legacy and Enhanced PWM Synchronization mode. At the end, example code is given showing both modes in both software and hardware trigger ways.

2 FlexTimer synchronization and registers concerned

The FTM module offers PWM synchronization mechanism which provides an opportunity to update the MOD, CNTIN, CnV, OUTMASK, INVCTRL and SWOCTRL registers with their buffered value and force the FTM counter to the CNTIN register value.

NOTE

- It is expected that the PWM synchronization be used only in Combine mode.
- The Legacy PWM Synchronization (SYNCONF[SYNCMODE] = 0) is a subset of the Enhanced PWM Synchronization (SYNCONF[SYNCMODE] = 1). Thus, it is expected that only the Enhanced PWM Synchronization be used.

The control registers of the FTM module, associated with the PWM synchronization are as below:

- FTMx_MODE, in which the FTMEN and PWMSYNC fields are concerned.
  Generally when using PWM function in Combine mode, FTMEN must be set as 1, or it is in TPM Compatible mode and can only offer basic PWM functions. In TPM mode (FTMEN = 0), the CNTIN, MOD, and CnV registers are updated simply.
  - CNTIN register is updated at the next system clock after CNTIN was written.
  - MOD register is updated after it is written and the FTM counter changes from MOD to CNTIN or MOD –1 according to SC[CPWMS].
  - CnV register is updated after it is written and the FTM counter changes from MOD to CNTIN (EPWM mode) or MOD –1 (CPWM mode). In Output Compare mode, the CnV register is updated on the next FTM counter change after it is written.

  PWMSYNC selects which triggers can be used by MOD, CnV, OUTMASK, and FTM counter synchronization,and configures the synchronization only when SYNCONF[SYNCMODE] = 0, or in Legacy mode which is not recommended in Kinetis.

- FTMx_SYNC
  It selects the software or hardware trigger source, load point and synchronization mode to OUTMASK and FTM counter.

  NOTE

  - The software trigger (SWSYNC field) and hardware triggers (TRIG0, TRIG1, and TRIG2 bits) have a potential conflict if used together when SYNCONF[SYNCMODE] = 0. It is recommended using only hardware or software triggers, but not both at the same time, otherwise unpredictable behavior is likely to happen.
  - The selection of the the maximum and the minimum loading point enabled by CNTMAX and CNTMIN fields, is intended to provide the update of MOD, CNTIN, and CnV registers across all enabled channels simultaneously. The use of the loading point selection together with SYNCONF[SYNCMODE] = 0 and hardware trigger selection (TRIG0, TRIG1, or TRIG2 bits) is likely to result in unpredictable behavior.

- FTMx_COMBINE, in which the SYNCEN and COMBINE fields are concerned.

  The recommended usage is in Combine mode. So, COMBINE must be set. SYNCEN enables the synchronization function to registers C(n)V and C(n+1)V.

- FTMx_SYNCONF

  This register selects the PWM synchronization configuration, SWOCTRL, INVCTRL and CNTIN registers synchronization, if FTM clears the TRIGj bit (where j = 0, 1, 2) when the hardware trigger j is detected.

- FTMx_PWMLOAD

This register enables the loading of the MOD, CNTIN, C(n)V, and C(n+1)V registers with the values of their write buffers when the FTM counter changes from the MOD register value to its next value or when a channel (j) match occurs. A match occurs for the channel (j) when FTM counter = C(j)V.

3 Synchronization principle

The Kinetis MCU offers two kinds of trigger source for synchronization:

- Software trigger: Software trigger source is SYNC[SWSYNC].
- Hardware trigger: Hardware trigger can be selected from CMPx output, PDB trigger output, FTM_FLT pin, or SYNC[TRIG0], SYNC[TRIG1], and SYNC[TRIG2] fields. The hardware trigger source varies for different devices, and can be checked from device chip configuration.

3.1 Hardware trigger

Hardware trigger signal inputs of the FTM module are enabled when SYNC[TRIGn] = 1, where n = 0, 1 or 2, corresponding to each one of the input signals, respectively. The hardware trigger input n is synchronized by the system clock. The PWM synchronization with hardware trigger is initiated when a rising edge is detected at the enabled hardware trigger inputs.

- If SYNCONF[HWTRIGMODE] = 0, SYNC[TRIGn] is cleared when 0 is written to it, or when the trigger n event is detected.
- If SYNCONF[HWTRIGMODE] = 1, SYNC[TRIGn] is cleared only when 0 is written to it.

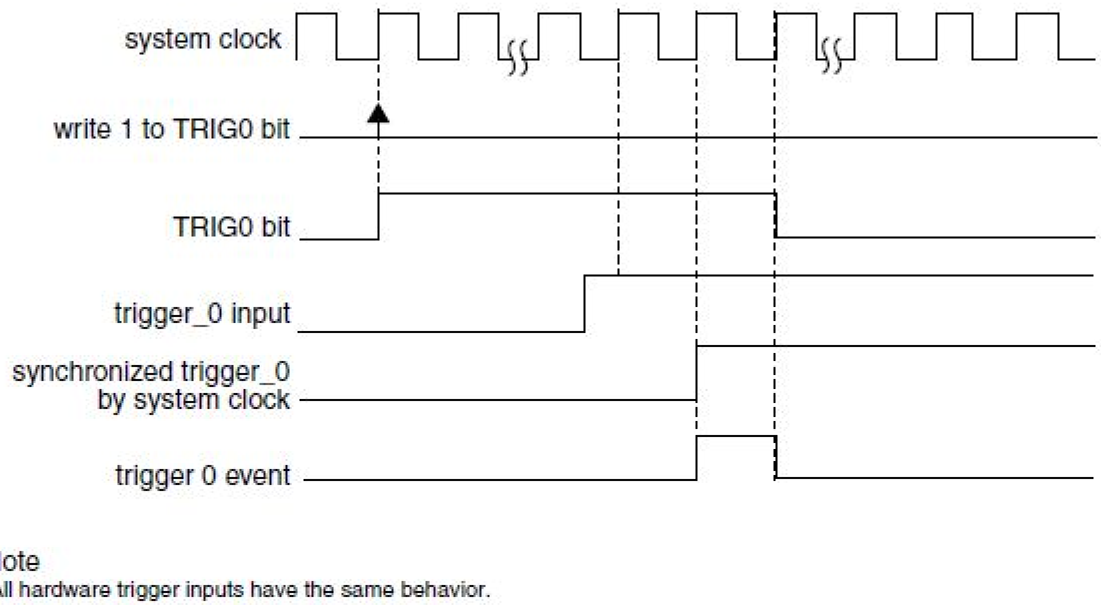

Figure 1. Hardware trigger event with SYNCONF[HWTRIGMODE] = 0

NOTE

It is expected that SYNCONF[HWTRIGMODE] be 1 only with enhanced PWM synchronization when SYNCONF[SYNCMODE] = 1.

3.2 Software trigger

A software trigger event occurs when 1 is written to SYNC[SWSYNC]. SYNC[SWSYNC] is cleared when 0 is written to it, or when the PWM synchronization initiated by the software event, is completed.

- In Legacy PWM Synchronization mode (when SYNCONF[SYNCMODE] = 0)
  - When MODE[PWMSYNC] = 1, or MODE[PWMSYNC] = 0 and SYNC[REINIT] = 0, SYNC[SWSYNC] is cleared at the next selected loading point (See Boundary cycle and loading points) after the software trigger event has occurred.
  - When MODE[PWMSYNC] = 0 and SYNC[REINIT] = 1, SYNC[SWSYNC] is cleared when the software trigger event occurs.
- In Enhanced PWM Synchronization mode (when SYNCONF[SYNCMODE] = 1),
  - When SYNCONF[SWRSTCNT] = 0, SYNC[SWSYNC] is cleared at the next selected loading point after that the software trigger event occurred.
  - When SYNCONF[SWRSTCNT] = 1, SYNC[SWSYNC] is cleared when the software trigger event occurs.

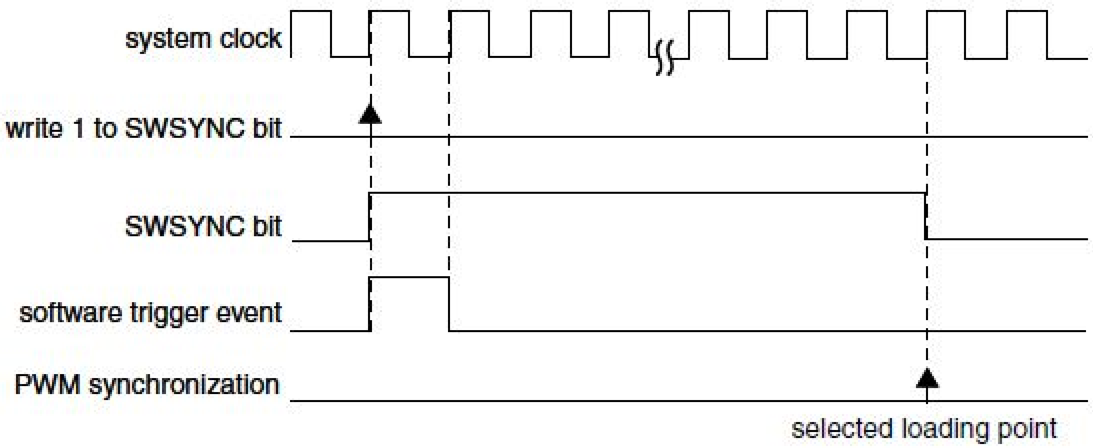

Figure 2. Software trigger event

3.3 Legacy PWM synchronization

Legacy mode is selected when SYNCONF[SYNCMODE] = 0. However, it is expected that the registers are synchronized only by the Enhanced PWM Synchronization.

3.3.1 MOD register synchronization

The MOD register synchronization updates the MOD register with its buffer value. This synchronization is enabled if MODE[FTMEN] = 1.

The MOD register synchronization can be done either by the Enhanced PWM Synchronization (SYNCONF[SYNCMODE] = 1) or the Legacy PWM Synchronization (SYNCONF[SYNCMODE] = 0). However, it is expected that the MOD register be synchronized only by the Enhanced PWM Synchronization.

In the case of enhanced PWM synchronization, the MOD register synchronization depends on SYNCONF[SWWRBUF], SYNCONF[SWRSTCNT], SYNCONF[HWWRBUF], and SYNCONF[HWRSTCNT], according to this flowchart:

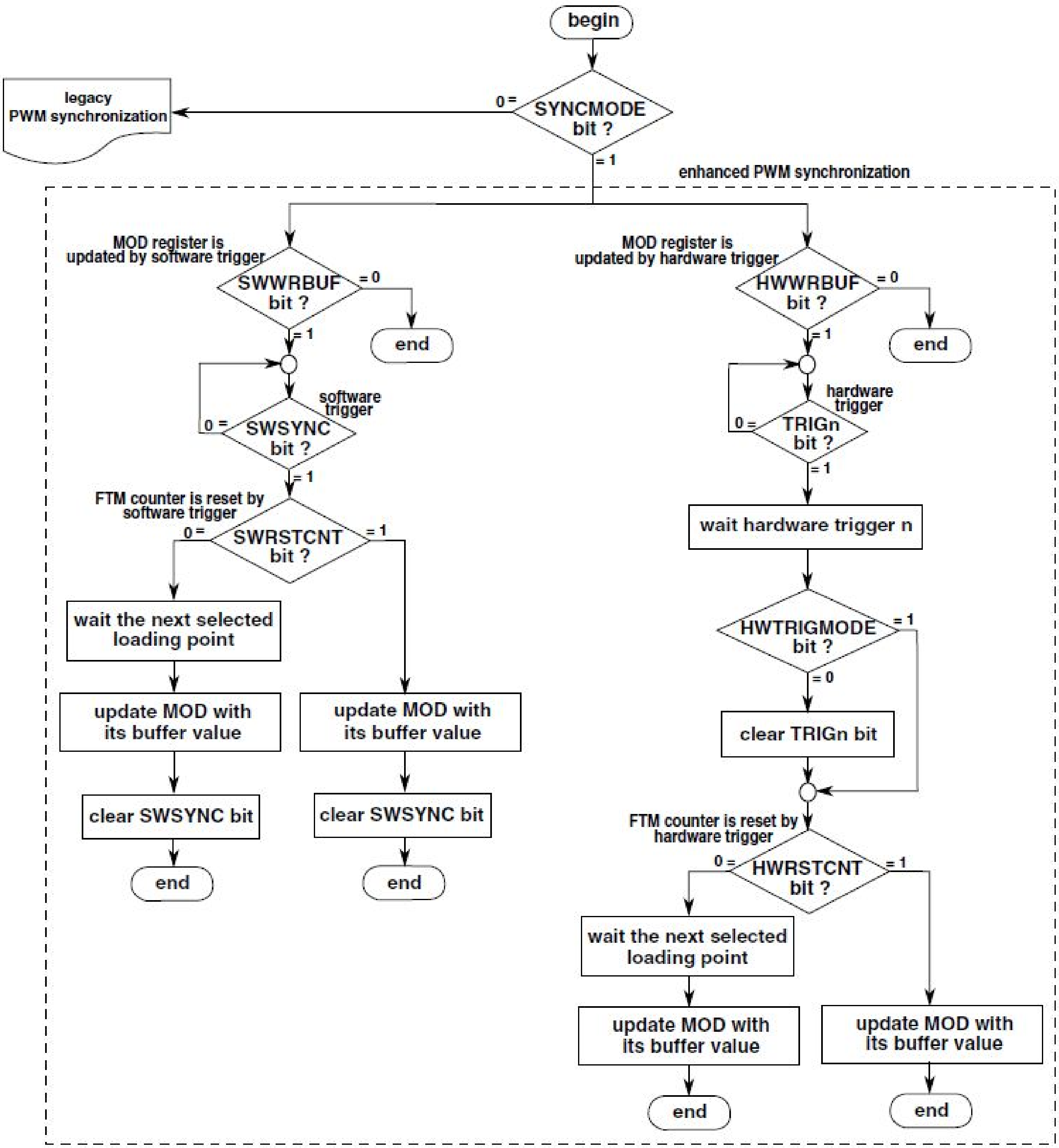

Figure 3. MOD register synchronization flowchart

3.3.2 CNTIN register synchronization

The CNTIN register synchronization can be done only by the Enhanced PWM Synchronization when SYNCONF[SYNCMODE] = 1.

3.3.3 C(n)V and C(n+1)V register synchronization

The synchronization mechanism is the same as the MOD register synchronization.

However, it is expected that the C(n)V and C(n+1)V registers be synchronized only by the Enhanced PWM Synchronization (when SYNCONF[SYNCMODE] = 1).

3.3.4 OUTMASK register synchronization

The OUTMASK register can be updated at each rising-edge of the system clock, when SYNCONF[SYNCHOM] = 0, or by the Legacy PWM synchronization, when SYNC[SYNCHOM] = 1 and SYNCONF[SYNCMODE] = 0. However, it is expected that the OUTMASK register be synchronized only by the enhanced PWM synchronization.

In the case of Legacy PWM Synchronization, the OUTMASK register synchronization depends on MODE[PWMSYNC] according to the following description.

If SYNCONF[SYNCMODE] = 0, SYNC[SYNCHOM] = 1, and SYNC[PWMSYNC] = 0, then this synchronization is done on the next enabled trigger event.

- If the trigger event was a software trigger, then SYNC[SWSYNC] is cleared on the next selected loading point. See Figure 4.
- If the trigger event was a hardware trigger, then SYNC[TRIGn] is cleared. See Figure 5.

  Examples with software and hardware triggers follow.

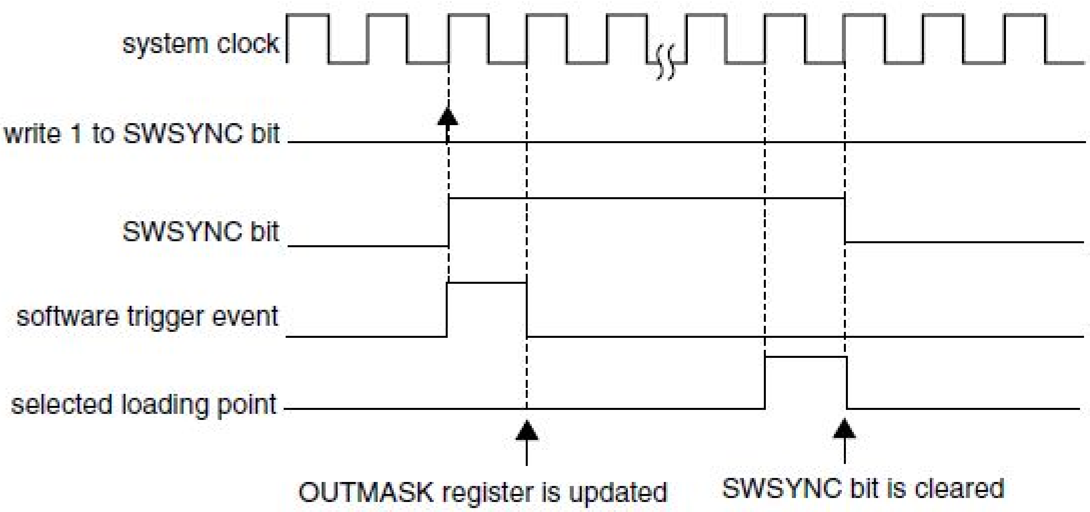

Figure 4. OUTMASK Synchronization with SYNCONF[SYNCMODE] = 0, SYNC[SYNCHOM] = 1, MODE[PWMSYNC] = 0 and software trigger was used

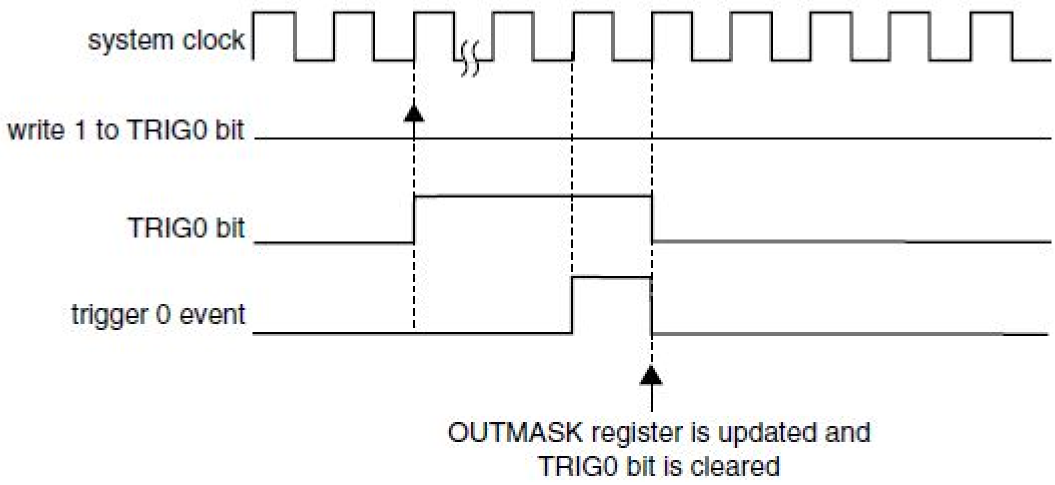

Figure 5. OUTMASK synchronization with SYNCONF[SYNCMODE] = 0, SYNCONF[HWTRIGMODE] = 0, SYNC[SYNCHOM] = 1, MODE[PWMSYNC] = 0,1, and a hardware trigger was used

3.3.5 INVCTRL register synchronization

The INVCTRL register synchronization updates the INVCTRL register with its buffer value.

The INVCTRL register can be updated at each rising-edge of the system clock, when SYNCONF[INVC] = 0, or by the Enhanced PWM Synchronization mode, when SYNCONF[INVC] = 1 and SYNCONF[SYNCMODE] = 1, according to the flowchart shown in Figure 6.

In the case of enhanced PWM synchronization, the INVCTRL register synchronization depends on SYNCONF[SWINVC] and SYNCONF[HWINVC].

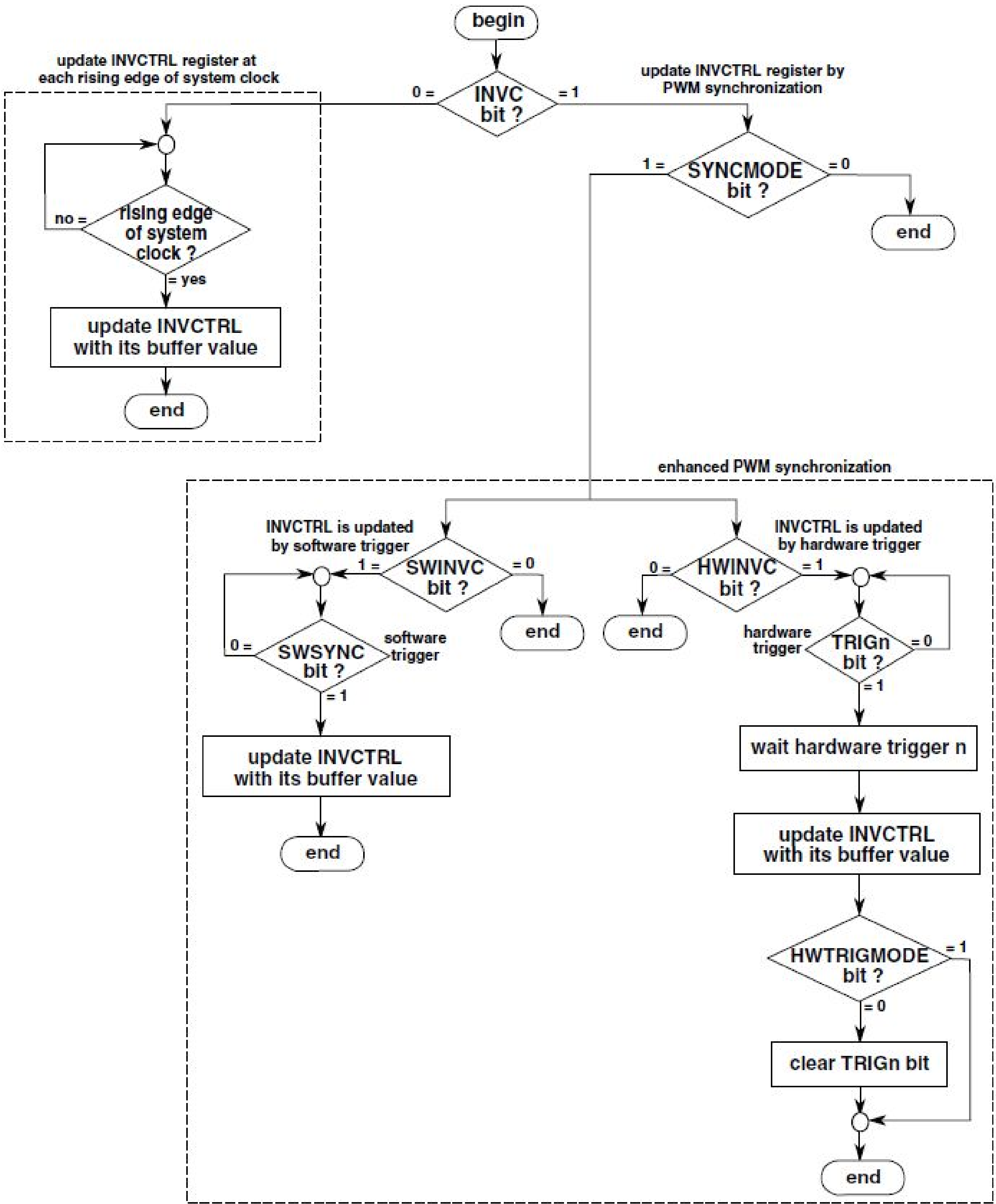

Figure 6. INVCTRL register synchronization flowchart

3.3.6 SWOCTRL register synchronization

The SWOCTRL register can be updated:

- At each rising-edge of the system clock when SYNCONF[SWOC] = 0, or
- By the Enhanced PWM Synchronization when SYNCONF[SWOC] = 1 and SYNCONF[SYNCMODE] = 1

The Legacy mode is not supported.

3.3.7 FTM counter synchronization

In the case of Legacy PWM synchronization, the FTM counter synchronization depends on SYNC[REINIT] and MODE[PWMSYNC] fields, according to the following description.

If SYNCONF[SYNCMODE] = 0, SYNC[REINIT] = 1, and MODE[PWMSYNC] = 0, then this synchronization is done on the next enabled trigger event.

- If the trigger event was a software trigger, then SYNC[SWSYNC] is cleared according to the example shown in Figure 7.
- If the trigger event was a hardware trigger then SYNC[TRIGn] field is cleared according to hardware trigger. See the example shown in Figure 8.

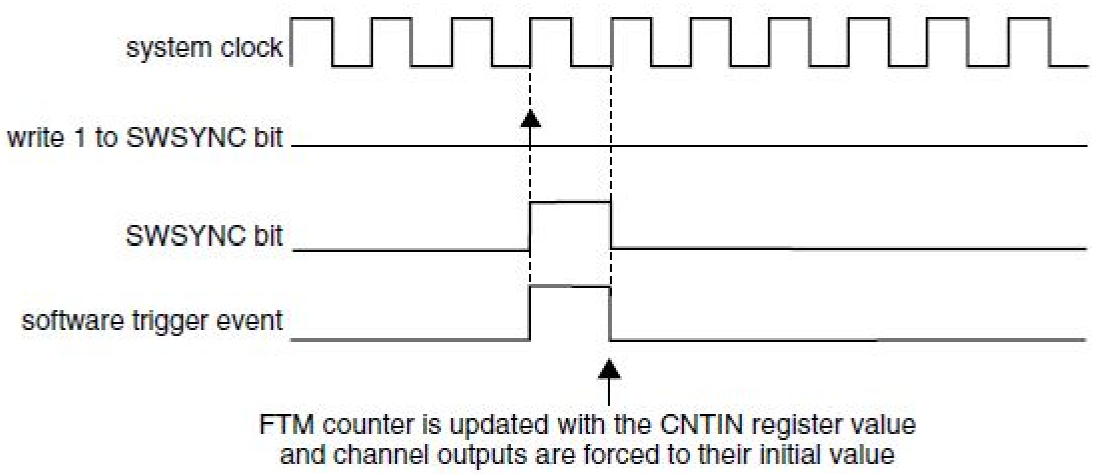

Figure 7. FTM counter synchronization with SYNCONF[SYNCMODE] = 0, SYNC[REINIT] = 1, MODE[PWMSYNC] = 0, and software trigger was used

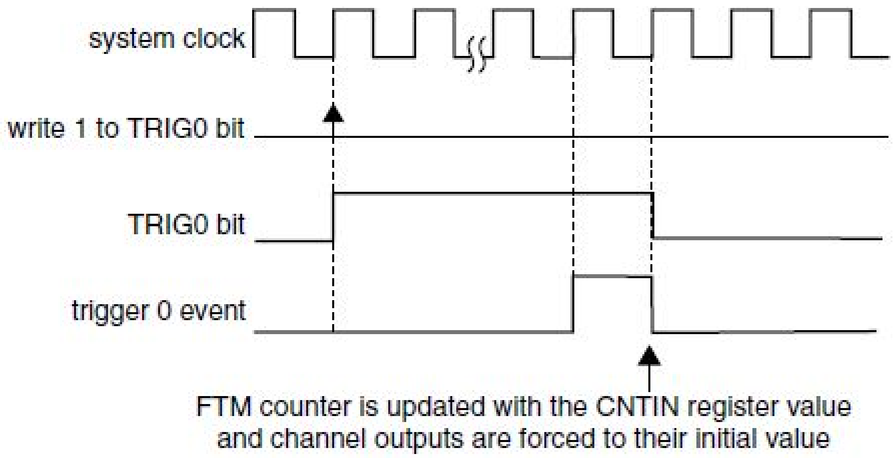

Figure 8. FTM counter synchronization with SYNCONF[SYNCMODE] = 0, SYNCONF[HWTRIGMODE] = 0, SYNC[REINIT] = 1, MODE[PWMSYNC] = 0, 1, and hardware trigger was used

3.4 Enhanced PWM synchronization

Enhanced mode is selected when SYNCONF[SYNCMODE] = 1. This synchronization mode is recommended.

3.4.1 MOD register synchronization

In the case of legacy PWM synchronization, the MOD register synchronization depends on MODE[PWMSYNC] and SYNC[REINIT] fields according to the following description.

- If SYNCONF[SYNCMODE] = 0, MODE[PWMSYNC] = 0, and SYNC[REINIT] = 0, or SYNCONF[SYNCMODE] = 0 and MODE[PWMSYNC] = 1, then this synchronization is done on the next selected loading point after an enabled trigger event takes place.
    - If the trigger event was a software trigger, then SYNC[SWSYNC] is cleared on the next selected loading point. See Figure 9.
    - If the trigger event was a hardware trigger, then the trigger enable field, SYNC[TRIGn] is cleared according to Hardware Trigger. See Figure 10.

Examples with software and hardware triggers are shown in the following figures.

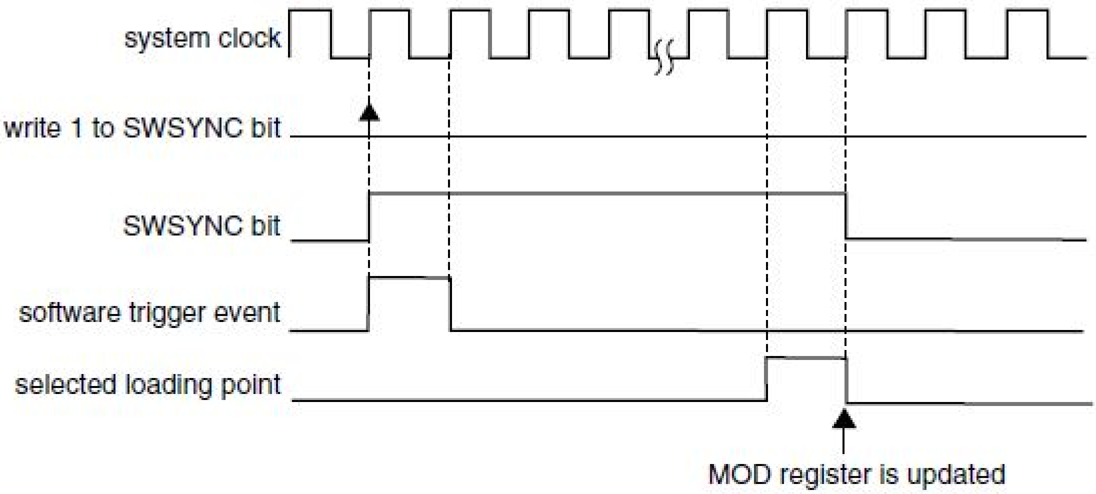

Figure 9. MOD synchronization with SYNCONF[SYNCMODE] = 0, MODE[PWMSYNC] = 0, SYNC[REINIT] = 0, or SYNCONF[SYNCMODE] = 0, MODE[PWMSYNC] = 1, and software trigger was used

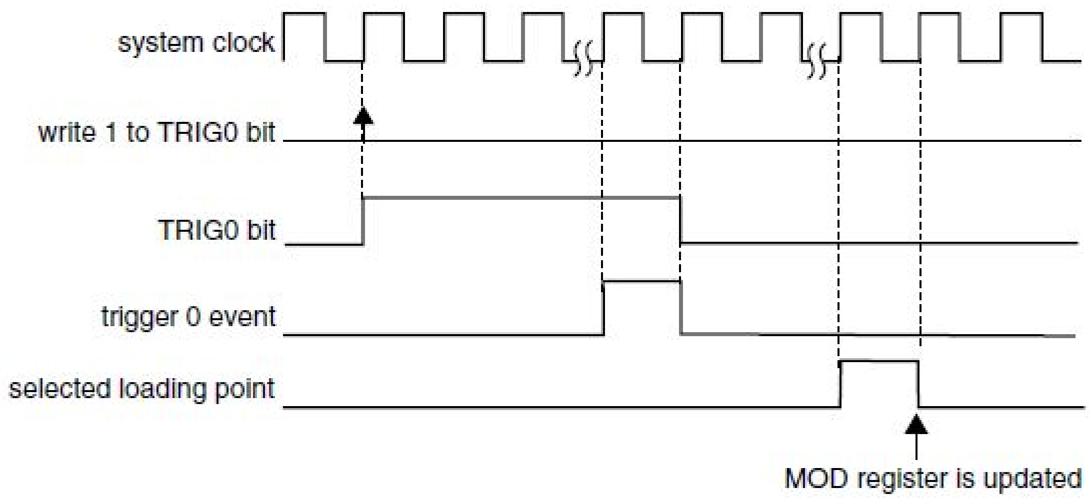

Figure 10. MOD synchronization with SYNCONF[SYNCMODE] = 0, SYNCONF[HWTRIGMODE] = 0, MODE[PWMSYNC] = 0, SYNC[REINIT] = 0, and a hardware trigger was used

If SYNCONF[SYNCMODE] = 0, MODE[PWMSYNC] = 0, and SYNC[REINIT] = 1, then this synchronization is made on the next enabled trigger event.

- If the trigger event was a software trigger, then SYNC[SWSYNC] is cleared according to the example given in Figure 11.
- If the trigger event was a hardware trigger, then SYNC[TRIGn] is cleared according to Hardware Trigger. See Figure 12.

Examples with software and hardware triggers are shown in the following figures.

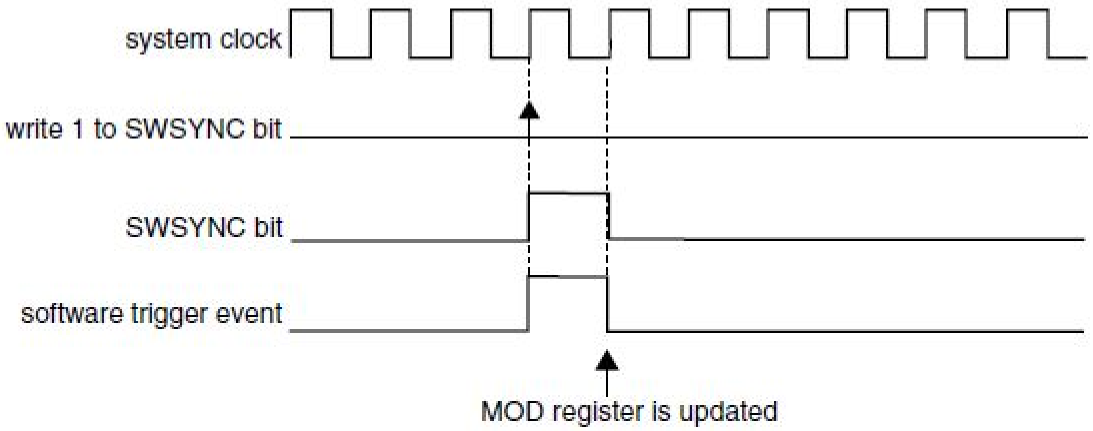

Figure 11. MOD synchronization with SYNCONF[SYNCMODE] = 0, MODE[PWMSYNC] = 0, SYNC[REINIT] = 1, and software trigger was used

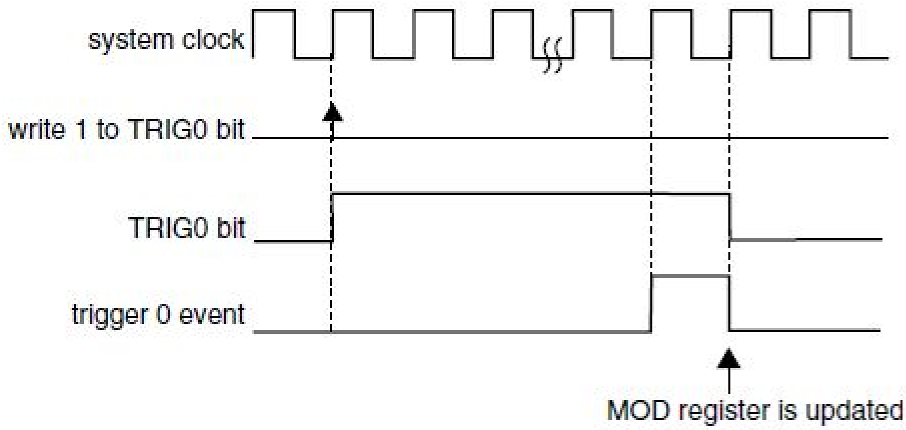

Figure 12. MOD synchronization with SYNCONF[SYNCMODE] = 0, SYNCONF[HWTRIGMODE] = 0, MODE[PWMSYNC] = 0, SYNC[REINIT] = 1, and a hardware trigger was used

3.4.2 CNTIN register synchronization

The CNTIN register synchronization updates the CNTIN register with its buffer value. This synchronization is enabled if MODE[FTMEN] = 1, SYNCONF[SYNCMODE] = 1, and SYNCONF[CNTINC] = 1. The CNTIN register synchronization can be done only by the Enhanced PWM Synchronization (SYNCONF[SYNCMODE] = 1). The synchronization mechanism is the same as the MOD register synchronization done by the enhanced PWM synchronization (See MOD register synchronization).

3.4.3 C(n)V and C(n+1)V register synchronization

The C(n)V and C(n+1)V registers synchronization updates the C(n)V and C(n+1)V registers with their buffer values.

This synchronization is enabled if MODE[FTMEN] = 1 and COMBINE[SYNCEN] = 1. The synchronization mechanism is the same as the MOD register synchronization (See MOD register synchronization). However, it is expected that the C(n)V and C(n+1)V registers be synchronized only by the Enhanced PWM Synchronization when SYNCONF[SYNCMODE] = 1.

3.4.4 OUTMASK register synchronization

The OUTMASK register synchronization updates the OUTMASK register with its buffer value.

The OUTMASK register can be updated at each rising-edge of the system clock (SYNC[SYNCHOM] = 0) by:

- The Enhanced PWM Synchronization, when SYNC[SYNCHOM] = 1 and SYNCONF[SYNCMODE] = 1, or
- The Legacy PWM Synchronization, when SYNC[SYNCHOM] = 1 and SYNCONF[SYNCMODE] = 0.

However, it is expected that the OUTMASK register be synchronized only by the Enhanced PWM Synchronization.

In the case of Enhanced PWM Synchronization, the OUTMASK register synchronization depends on SYNCONF[SWOM] and SYNCONF[HWOM] fields. See the following flowchart.

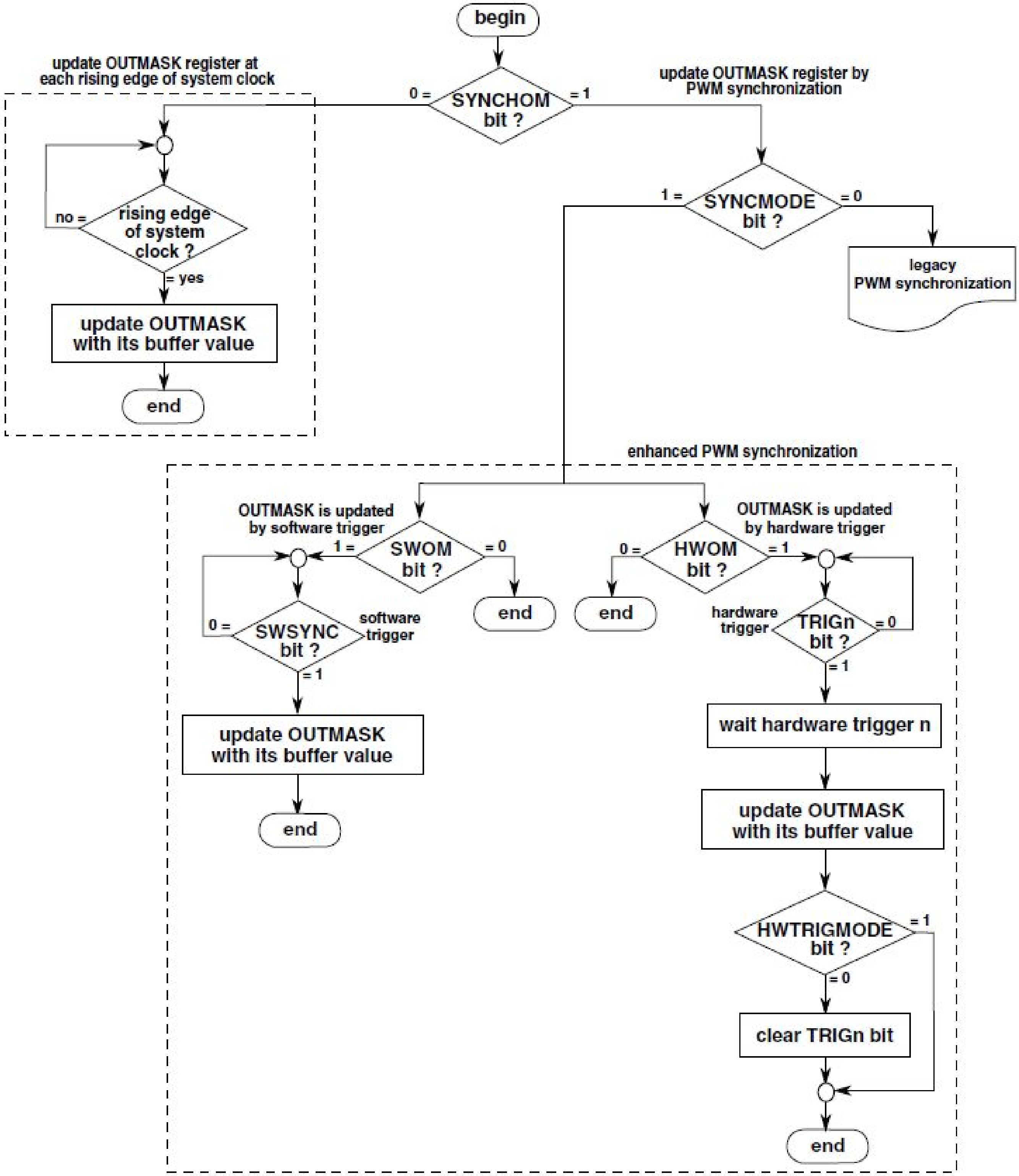

Figure 13. OUTMASK register synchronization flowchart

3.4.5 INVCTRL register synchronization

The INVCTRL register can be updated:

- At each rising-edge of the system clock (SYNCONF[INVC] = 0), or
- By the Enhanced PWM Synchronization when SYNCONF[INVC] = 1 and SYNCONF[SYNCMODE] = 1

The Legacy mode is not supported.

3.4.6 SWOCTRL register synchronization

The SWOCTRL register can be updated:

- at each rising-edge of the system clock when SYNCONF[SWOC] = 0, or
- by the Enhanced PWM Synchronization when SYNCONF[SWOC] = 1 and SYNCONF[SYNCMODE] = 1, according to the flowchart shown in Figure 14.

In the case of enhanced PWM synchronization, the SWOCTRL register synchronization depends on the SYNCONF[SWSOC] and SYNCONF[HWSOC] fields.

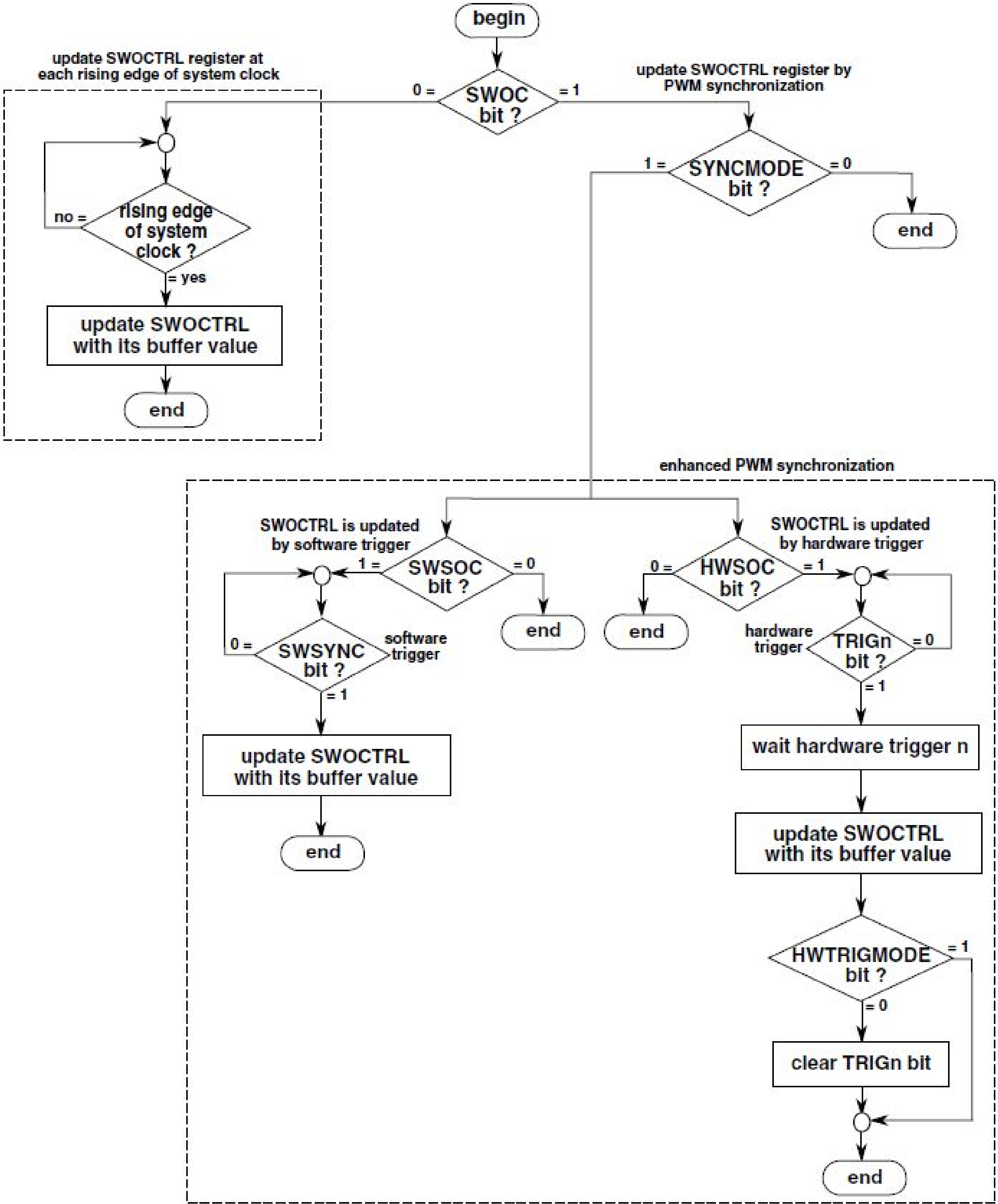

Figure 14. SWOCTRL register synchronization flowchart

3.4.7 FTM counter synchronization

The FTM counter synchronization is a mechanism that allows the FTM to restart the PWM generation at a certain point in the PWM period. All the channels outputs except for those in Output Compare mode, are forced to their initial value, and the FTM counter is forced to its initial counting value defined by the CNTIN register.

Figure 15 shows the FTM counter synchronization.

NOTE

After the synchronization event has occurred, the channel (n) is set to its initial value and the channel (n+1) is not set to its initial value due to a specific timing of this figure in which the deadtime insertion prevents this channel output from transitioning to 1. If no deadtime insertion is selected, then the channel (n+1) transitions to logical value 1 immediately after the synchronization event has occurred.

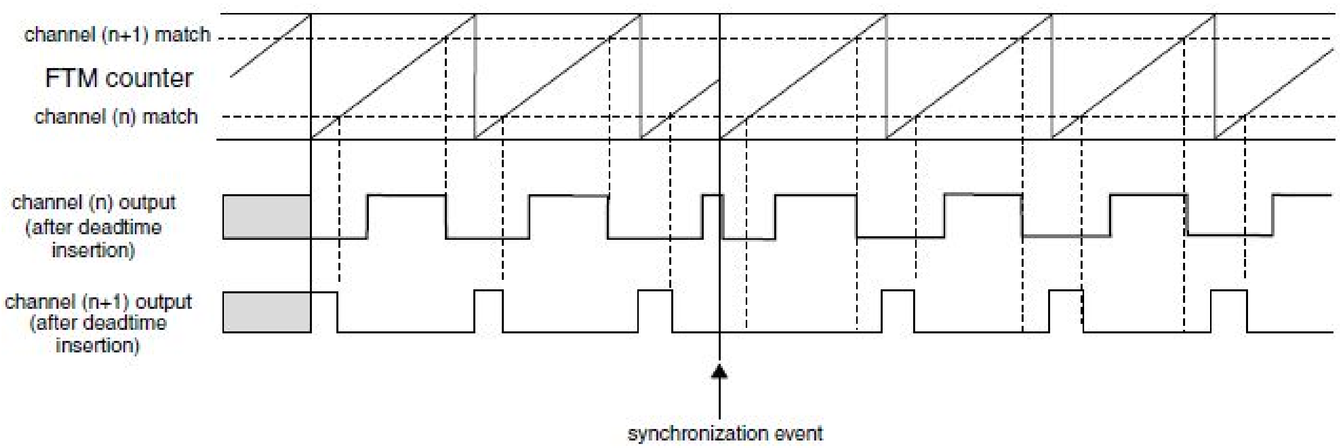

Figure 15. FTM counter synchronization

The FTM counter synchronization can be done by either the Enhanced PWM Synchronization, when SYNCONF[SYNCMODE] = 1, or the Legacy PWM synchronization, when SYNCONF[SYNCMODE] = 0.

However, it is expected that the FTM counter be synchronized only by the enhanced PWM synchronization.

In the case of Enhanced PWM Synchronization, the FTM counter synchronization depends on SYNCONF[SWRSTCNT] and SYNCONF[HWRSTCNT] fields according to the following flowchart.

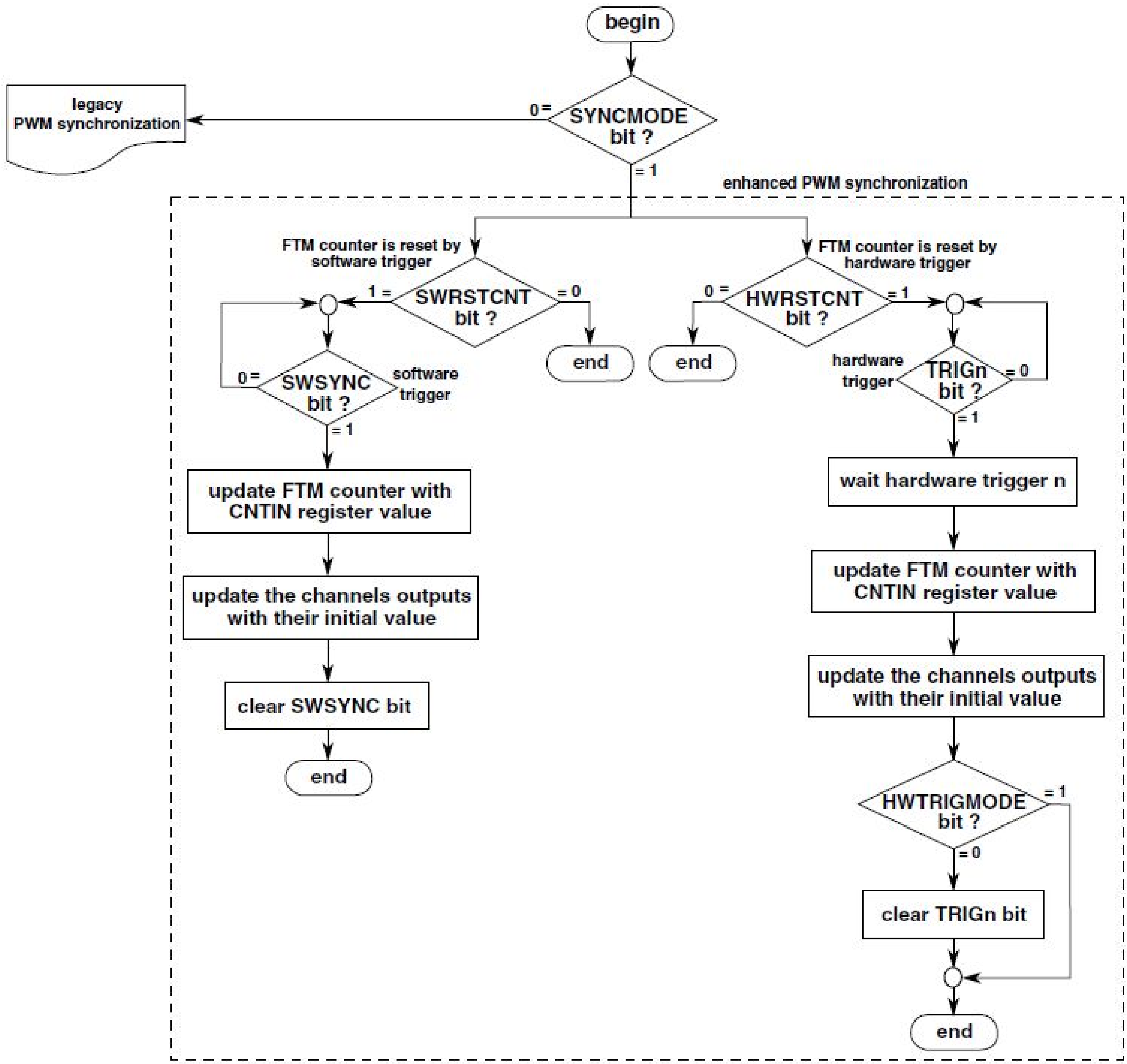

Figure 16. FTM counter synchronization flowchart

3.5 Boundary cycle and loading points

The boundary cycle definition is important for the loading points for the MOD, CNTIN, and C(n)V registers

- In Up-Counting mode, (Up Counting) the boundary cycle is defined as when the counter wraps to its initial value (CNTIN). In this mode, the loading points are enabled if one of the SYNC[CNTMIN] or SYNC[CNTMAX} fields is 1.
- In Up-Down Counting mode (Up-Down Counting), the boundary cycle is defined as when the counter turns from down to up counting and up to down counting. In the up-down counting mode, the loading points are selected by SYNC[CNTMIN] and SYNC[CNTMAX], as indicated in Figure 17.

Figure 17 shows the boundary cycles and the loading points. The loading points are safe places for register updates, thus allowing a smooth transitions in PWM waveform generation.

For both the counting modes, if neither SYNC[CNTMIN] nor SYNC[CNTMAX] is 1, then the boundary cycles are not used as loading points for registers updates. See the register synchronization descriptions in the following sections for details.

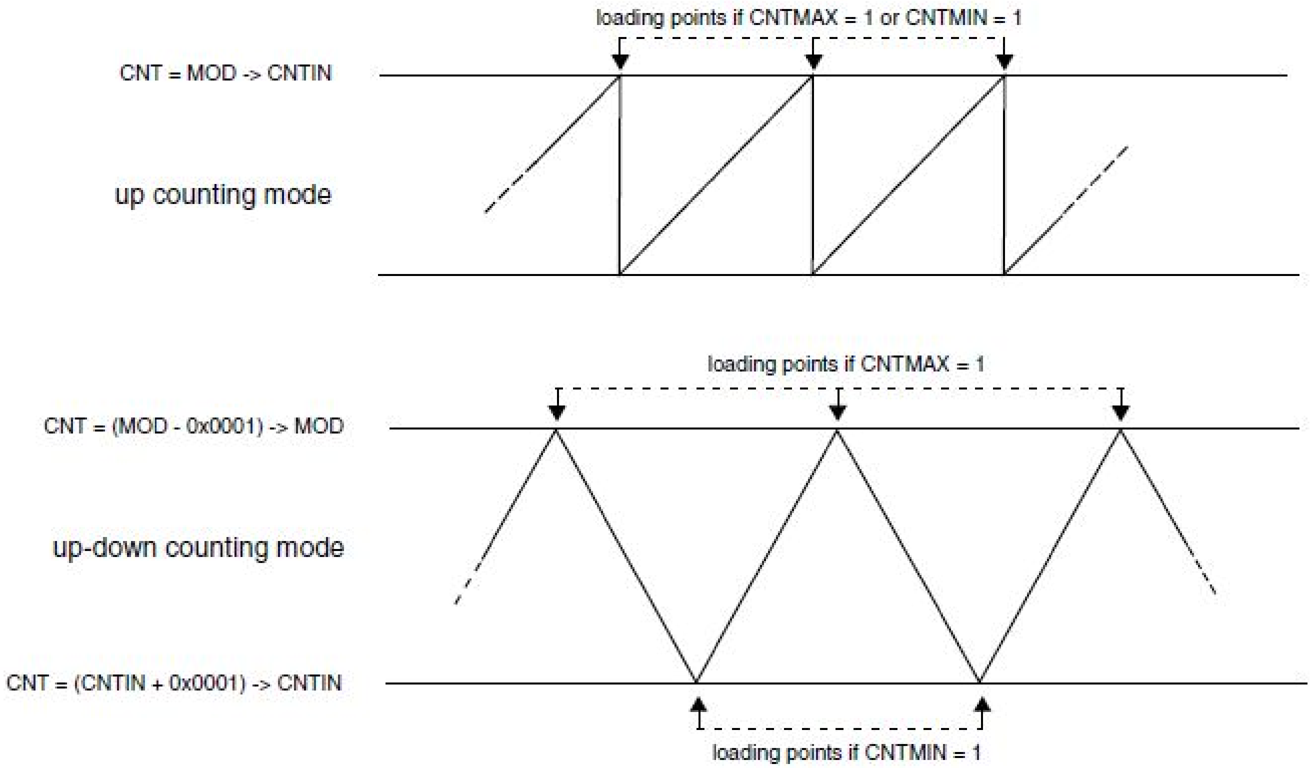

Figure 17. Boundary cycles and loading points

4 Example code

The example code is based on ARM®Cortex™-M4 core KE15 device. Both Legacy and Enhanced PWM synchronization modes are shown and both software and hardware trigger are involved in each mode.

```
/* Trigger source select, enable one macro at one time */

//#define SYNC_TRIGGER_TEST 1 /* Software synchronization */
#define SYNC_TRIGGER_TEST 2   /* Trigger0 synchronization */

/* take FTM0 as example, initial FTM0 registers */

FTM0_MODE    = 0x05;             /* FTM features are        */
FTM0_COMBINE = 0x232323;      /* Combine mode is enabled */

FTM0_C0SC = 0x28;
FTM0_C1SC = 0x28;
FTM0_C2SC = 0x28;
FTM0_C3SC = 0x28;
FTM0_C4SC = 0x28;
FTM0_C5SC = 0x28;
FTM0_MOD  = 999;
FTM0_C0V  =  100;
FTM0_C1V  =  800;
FTM0_C2V  =  100;
FTM0_C3V  =  800;
FTM0_C4V  =  100;
FTM0_C5V  =  800;
```

```
  FTM0_SC   = 0x08;

  #if SYNC_TRIGGER_TEST == 1
    printf("FTM Software synchronization Test-- legacy mode\r\n");
    FTM0_SYNCONF = 0x00000034;
FTM0_SYNC = 0x0C;
  #elif SYNC_TRIGGER_TEST == 2
    printf("FTM TRIG0 synchronization Test-- legacy mode\r\n");
    FTM0_SYNCONF = 0x00000034;
    FTM0_SYNC = 0x1C;
  #endif

  /*  update the FTM0 registers  */
  FTM0_MOD = 500;
  FTM0_C0V = 200;
  FTM0_C1V = 400;
  FTM0_C2V = 200;
  FTM0_C3V = 400;
  FTM0_C4V = 200;
  FTM0_C5V = 400;
  FTM0_OUTMASK = 0x3F;
  FTM0_CNTIN   = 0x30;
  FTM0_INVCTRL = 0x03;
  FTM0_SWOCTRL = 0x3F3F;

  printf("Check the registers still keep old value before synchronization\r\n");
  printf("FTM0_MOD = %d\r\n",FTM0_MOD);
  printf("FTM0_C0V = %d\r\n",FTM0_C0V);
  printf("FTM0_C1V = %d\r\n",FTM0_C1V);
  printf("FTM0_C2V = %d\r\n",FTM0_C2V);
  printf("FTM0_C3V = %d\r\n",FTM0_C3V);
  printf("FTM0_C4V = %d\r\n",FTM0_C4V);
  printf("FTM0_C5V = %d\r\n",FTM0_C5V);
  printf("FTM0_OUTMASK = %x\r\n",FTM0_OUTMASK);
  printf("FTM0_CNTIN = %x\r\n",FTM0_CNTIN);
  printf("FTM0_INVCTRL = %x\r\n",FTM0_INVCTRL);
  printf("FTM0_SWOCTRL = %x\r\n",FTM0_SWOCTRL);
  printf("FTM0_CNT = %x\r\n",FTM0_CNT);

  #if SYNC_TRIGGER_TEST == 1
    FTM0_SYNC = 0x8C;              // software trigger
  #elif SYNC_TRIGGER_TEST == 2
    SIM_SOPT3 &= ~0x00010000;      //before setting, clear first
    asm(nop);
    asm(nop);
    SIM_SOPT3 |= 0x00010000;       //set FTM_SYNCx to generate trigger 0
  #endif

  printf("Check the register value changed after synchronization\r\n");
  printf("FTM0_MOD = %d\r\n",FTM0_MOD);
  printf("FTM0_C0V = %d\r\n",FTM0_C0V);
  printf("FTM0_C1V = %d\r\n",FTM0_C1V);
  printf("FTM0_C2V = %d\r\n",FTM0_C2V);
  printf("FTM0_C3V = %d\r\n",FTM0_C3V);
  printf("FTM0_C4V = %d\r\n",FTM0_C4V);
  printf("FTM0_C5V = %d\r\n",FTM0_C5V);
  printf("FTM0_OUTMASK = %x\r\n",FTM0_OUTMASK);
  printf("FTM0_CNTIN = %x\r\n",FTM0_CNTIN);
  printf("FTM0_INVCTRL = %x\r\n",FTM0_INVCTRL);
  printf("FTM0_SWOCTRL = %x\r\n",FTM0_SWOCTRL);
  printf("FTM0_CNT = %x\r\n",FTM0_CNT);

  #if SYNC_TRIGGER_TEST == 1
    printf("FTM Software synchronization Test-- enhanced mode\r\n");
    FTM0_SYNCONF = 0x00001FB4;      // enhanced mode, software trigger
    FTM0_SYNC = 0x00;
  #elif SYNC_TRIGGER_TEST == 2
    printf("FTM TRIG0 synchronization Test-- enhanced mode\r\n");
FTM0_SYNCONF = 0x001F00B4;      // enhanced mode, hardware trigger 0
    FTM0_SYNC = 0x10;
```

```
  #endif

/*   update the FTM0 registers   */
  FTM0_MOD =  999;
  FTM0_C0V =  100;
  FTM0_C1V =  800;
  FTM0_C2V =  100;
  FTM0_C3V =  800;
  FTM0_C4V =  100;
  FTM0_C5V =  800;
  FTM0_OUTMASK = 0x3F;
  FTM0_CNTIN   = 0x40;
  FTM0_INVCTRL = 0x04;
  FTM0_SWOCTRL = 0x3F3F;

  printf("Check the registers still keep old value before synchronization\r\n");
  printf("FTM0_MOD = %d\r\n",FTM0_MOD);
  printf("FTM0_C0V = %d\r\n",FTM0_C0V);
  printf("FTM0_C1V = %d\r\n",FTM0_C1V);
  printf("FTM0_C2V = %d\r\n",FTM0_C2V);
  printf("FTM0_C3V = %d\r\n",FTM0_C3V);
  printf("FTM0_C4V = %d\r\n",FTM0_C4V);
  printf("FTM0_C5V = %d\r\n",FTM0_C5V);
  printf("FTM0_OUTMASK = %x\r\n",FTM0_OUTMASK);
  printf("FTM0_CNTIN = %x\r\n",FTM0_CNTIN);
  printf("FTM0_INVCTRL = %x\r\n",FTM0_INVCTRL);
  printf("FTM0_SWOCTRL = %x\r\n",FTM0_SWOCTRL);
  printf("FTM0_CNT = %x\r\n",FTM0_CNT);

  #if SYNC_TRIGGER_TEST == 1
    FTM0_SYNC = 0x80;            // generate software trigger
  #elif SYNC_TRIGGER_TEST == 2
    SIM_SOPT3 &= ~0x00010000;    // before setting, clear first
    asm(nop);
    asm(nop);
    SIM_SOPT3 |= 0x00010000;     // set FTM_SYNCx to generate trigger 0
  #endif

  printf("Check the register value changed after synchronization\r\n");
  printf("FTM0_MOD = %d\r\n",FTM0_MOD);
  printf("FTM0_C0V = %d\r\n",FTM0_C0V);
  printf("FTM0_C1V = %d\r\n",FTM0_C1V);
  printf("FTM0_C2V = %d\r\n",FTM0_C2V);
  printf("FTM0_C3V = %d\r\n",FTM0_C3V);
  printf("FTM0_C4V = %d\r\n",FTM0_C4V);
  printf("FTM0_C5V = %d\r\n",FTM0_C5V);
  printf("FTM0_OUTMASK = %x\r\n",FTM0_OUTMASK);
  printf("FTM0_CNTIN = %x\r\n",FTM0_CNTIN);
  printf("FTM0_INVCTRL = %x\r\n",FTM0_INVCTRL);
  printf("FTM0_SWOCTRL = %x\r\n",FTM0_SWOCTRL);
  printf("FTM0_CNT = %x\r\n",FTM0_CNT);
```

Figure 18 and Figure 19 show the execution results of example code with software and hardware triggers respectively.

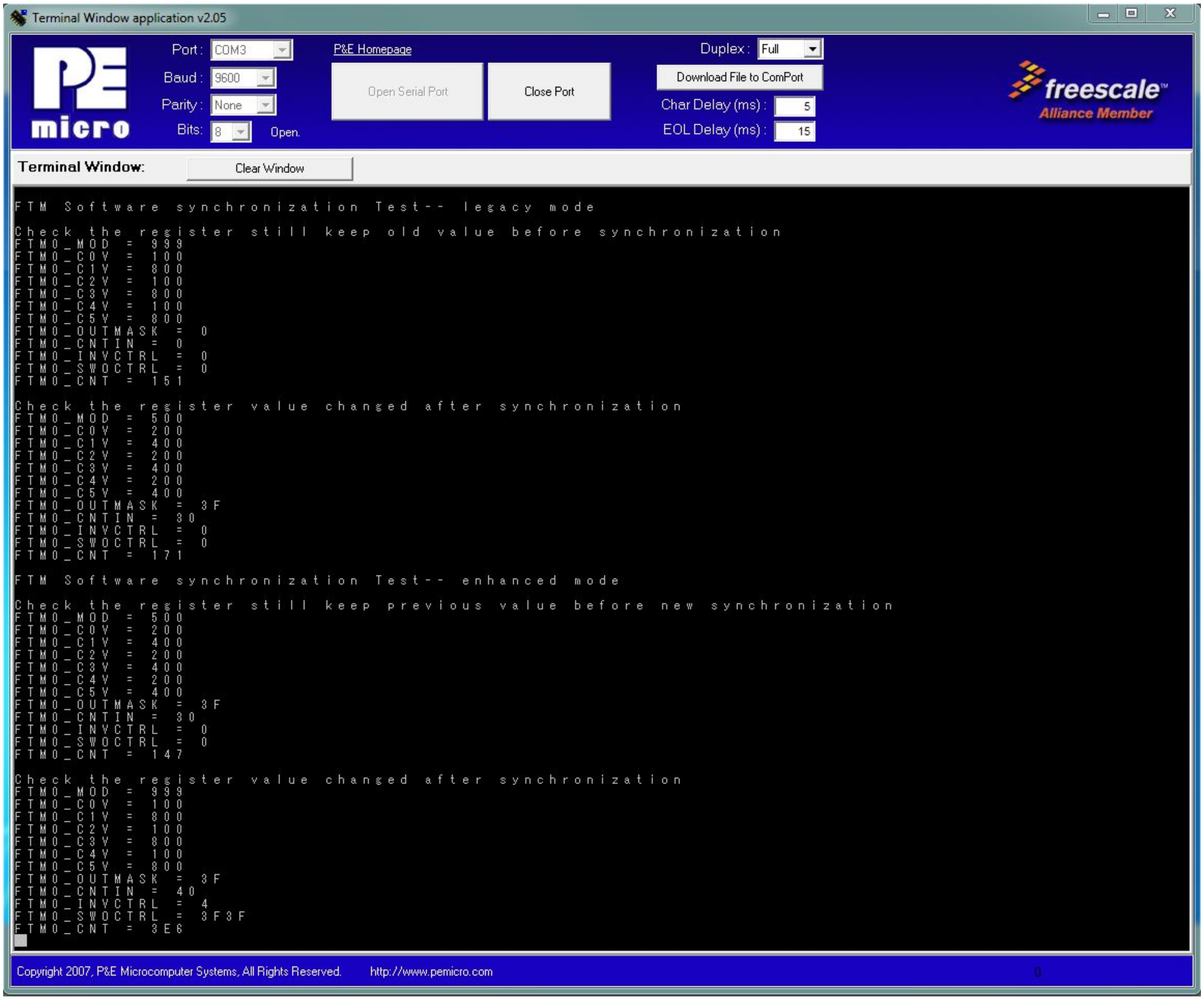

Figure 18. Software Trigger

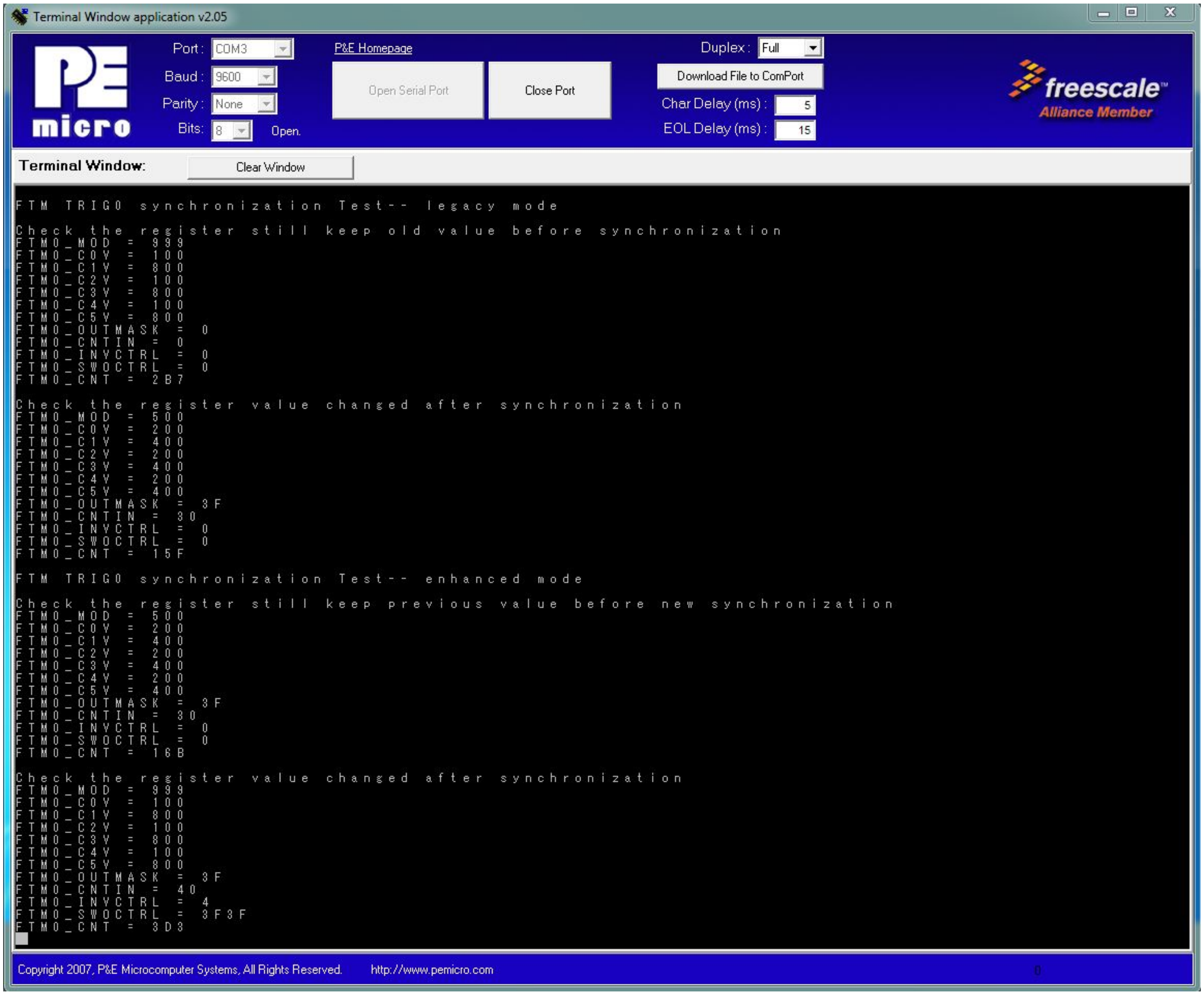

Figure 19. Hardware trigger–trigger 0

5 Conclusion

There are too many ways for FTM synchronization as described in this application note, which include Legacy mode, Enhanced mode and both modes include software and hardware trigger. The choice of the FTM synchronization method depends on target applications.

The enhanced PWM synchronization mode is recommended for motor control and power conversion applications.

6 References

- K60 Sub-Family Reference Manual, available at http://www.freescale.com

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Document Number: AN4560
Rev. 0, 8/2012