Application Note

Using the Timer of the Z8 Encore! ® and Z8 Encore! XP® Family of Microcontrollers

AN029102-0810

Abstract

This application note describes a method for using the timer (on-chip peripheral) of Z8 Encore!® and Z8 Encore! XP® family of microcontrollers. This document contains sample codes (in C and assembly language) that illustrate the method to initialize timers and its associated general purpose input/output (GPIO) for the Z8 Encore! and Z8 Encore! XP® devices. This document also highlights the Z8 Encore! and Z8 Encore! XP® family timer functionality, and provides a set of routines to use the timer in different operating modes such as the one-shot, continuous, counter, comparator counter, PWM single and dual output, capture, capture restart, compare, gated, and capture/compare modes.

- ***Note1:*** *For ease of documentation and discussion, the term "Z8 Encore!" or "Z8 Encore! devices" will be used to refer to both Z8 Encore! and Z8 Encore XP devices from this point forward.*
- ***Note2:*** *The source code (AN0291-SC02.zip) associated with this application note has been tested with ZDS II – version 4.11.0, and is available for download from www.zilog.com.*

Overview of the Timer Peripheral in the Z8 Encore! Devices

The Z8 Encore! devices contain up to four 16-bit reloadable timers, which can be used for timing, event counting, or generation of pulse-width modulated (PWM) signals. Figure 1 below displays the architecture of the timer peripheral.

The features of the Z8 Encore! Timer includes the following:

– 16-bit reloadable counter
– Programmable prescalar with prescale values from 1 to 128
– PWM output generation
– Capture and compare capability
– External input pin for timer input, clock gating, or capture signal. External input pin signal frequency is limited to a maximum of one-fourth the system clock frequency.
– Timer output pin
– Timer interrupt

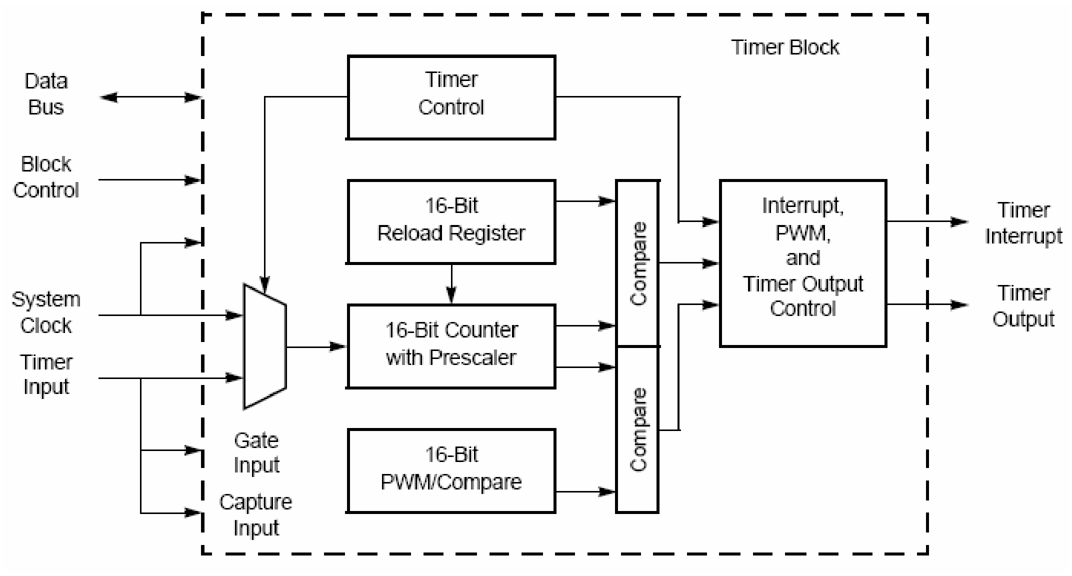

Figure 1 Architecture of the Z8 Encore! Timer

The different Z8 Encore! devices have slight differences on the operating modes offered per device to support different application requirements. The timer operating modes of the different Z8 Encore! devices are as follows:

Table 1 Z8 Encore! Timer Operating Modes

Operating Mode	F0830	F083A	F082A	F0823	F0822	F1680	F64xx
One Shot	✓	✓	✓	✓	✓	✓	✓
Triggered One Shot	-	-	-	-	-	✓	-
Continuous	✓	✓	✓	✓	✓	✓	✓
Counter	✓	✓	✓	✓	✓	✓	✓
Comparator Counter	✓	✓	✓	✓	-	✓	-
PWM Single Output	✓	✓	✓	✓	✓	✓	✓
PWM Dual Output	✓	✓	✓	✓	-	✓	-
Capture	✓	✓	✓	✓	✓	✓	✓
Capture Restart	✓	✓	✓	✓	-	✓	-
Compare	✓	✓	✓	✓	✓	✓	✓
Gated	✓	✓	✓	✓	✓	✓	✓
Capture/Compare	✓	✓	✓	✓	✓	✓	✓
Demodulation	-	-	-	-	-	✓	-

Z8 Encore! Timer Register Description

The Z8 Encore! Timer registers are briefly described in this section.

1. Timer Control Registers (TxCTL)

The Timer Control Register selects the timer operating mode, prescale value, timer interrupt configuration, and PWM delay configuration. This register also defines the polarity of the timer input pin, and enables/disables the timer. It also includes a read-only `INPCAP` bit to identify if the most recent timer interrupt is caused by an input capture event.

Table 2 Timer Control Register 0 (TxCTL0)

BITS	7	6	5	4	3	2	1	0
F082A	TMODEHI	TICONFIG		Rsv	PWMD			INPCAP
F0823	TMODEHI	TICONFIG		Rsv	PWMD			INPCAP
F0822	Rsv			CSC	Rsv			
F1680	TMODE[3]	TICONFIG		CSC	PWMD			INPCAP
F64xx	Rsv			CSC	Rsv			

TMODEHI / TMODE[3] (Timer Mode High Bit) – This bit along with the TMODE field in TxCTL1 register determines the operating mode of the timer. This is the most significant bit of the timer mode selection value.

TICONFIG (Timer Interrupt Configuration) – This field configures the timer interrupt definition.

Rsv (Reserved) – Must always be 0.

PWMD (PWM Delay Value) – This field is a programmable delay to control the number of system clock cycles delay before the timer output and the timer output complement are forced to their active state.

Table 3 Timer Control Register 1 (TxCTL1)

BITS	7	6	5	4	3	2	1	0
F082A	TEN	TPOL	PRES			TMODE		
F0823	TEN	TPOL	PRES			TMODE		
F0822	TEN	TPOL	PRES			TMODE		
F1680	TEN	TPOL	PRES			TMODE		
F64xx	TEN	TPOL	PRES			TMODE		

TEN (Timer Enable) – Enables/disables the timer.

TPOL (Timer Input/Output Polarity) – Operation of this bit is a function of the current operating mode of the timer. Generally, this bit selects the polarity of the timer input/output pins.

PRES (Prescale Value) – The timer input clock is divided by 2^{PRES}, where PRES can be set from 0 to 7. The prescalar is reset each time the timer is disabled.

TMODE (Timer Mode) – Defines the timer operating mode.

Table 4 Timer Control Register 2 (TxCTL2)

BITS	7	6	5	4	3	2	1	0
F082A	N/A							
F0823	N/A							
F0822	N/A							
F1680	Rsv		PWM0UE	TPOLHI	Rsv			TCLKS
F64xx	N/A							

PWM0UE (PWM0 Update Enable) – This bit determines whether writes to the PWM0 High and Low Byte Registers are buffered or not.

TPOLHI (Timer Input/Output Polarity High Bit) – This bit determines if timer count is triggered and captured on both edges of the input signal.

TCLKS (Timer Clock Source) – This bit determines if timer clock source is the system clock or the peripheral clock.

Rsv (Reserved) – Must always be 0.

2. Timer High and Low Byte Registers (TxH, TxL)

The Timer High and Low Byte Registers contain the current 16-bit timer count value.

Table 5 Timer High Byte Register (TxH)

BITS	7	6	5	4	3	2	1	0
FIELD	TH							
RESET	0							

Table 6 Timer Low Byte Register (TxL)

BITS	7	6	5	4	3	2	1	0
FIELD	TL							
RESET	0							1

3. Timer Reload High and Low Byte Registers (TxRH, TxRL)

The Timer Reload High and Low Byte Registers stores the16-bit reload value, which is the maximum count value that initiates a timer reload to `0001h`. In COMPARE mode, these registers form the 16-bit compare value.

Table 7 Timer Reload High Byte Registers (TxRH)

BITS	7	6	5	4	3	2	1	0
FIELD	TRH							
RESET	1							

Table 8 Timer Reload Low Byte Register (TxL)

BITS	7	6	5	4	3	2	1	0
FIELD	TRL							
RESET	1							

4. Timer PWM High and Low Byte Registers (TxPWMH, TxPWML)

The Timer PWM High and Low Byte Registers are used for pulse-width modulator (PWM) operations. These two bytes form a 16-bit value that is compared to the current 16-bit timer count. When a match occurs, the PWM output changes state. These registers also store the 16-bit captured timer value when operating in CAPTURE or CAPTURE/COMPARE modes.

Table 9 Timer PWM High Byte Register

BITS	7	6	5	4	3	2	1	0
FIELD	PWMH							
RESET	0							

Table 10 Timer PWM Low Byte Register

BITS	7	6	5	4	3	2	1	0
FIELD	PWML							
RESET	0							

Code Preparation and Setup

The preprocessor directives used to enable or disable the timer, to switch the timer between different operating modes, and to change the prescalar are as follows:

```
#define TIMER0_ISR_ENABLE        // Remove if interrupt is not used
#define TIMER0_OUTPUTPIN_ENABLE  // Remove if timer output is not used
//#define TIMER0_CAPTURE_MODE    // Uncomment if using Capture, Capture
                                 // Restart, or Capture Compare mode

#define RELOAD                   (0x1201)
#define PRESCALE                 (PRESCALAR_4)
#define STARTCOUNT               (0x0001)
#define COUNTER                  (0x0005)    // timer counter value

//////////////////////////////////////////////////////////////////////////
// Different Operating modes of the TIMER
#define ENABLE_TIMER             (0x80)
#define DISABLE_TIMER            (0x7F)

#define ONE_SHOT_MODE            (0x00)
#define CONTINUOUS_MODE          (0x01)
#define COUNTER_MODE             (0x02)
#define PWM_MODE                 (0x03)
#define CAPTURE_MODE             (0x04)
#define COMPARE_MODE             (0x05)
#define GATED_MODE               (0x06)
#define CAPTURE_COMPARE_MODE     (0x07)
// Additional Timer Modes Not Applicable to F64xx and F0822
#define PWM_DUAL_OUTPUT_MODE     (0x08)
#define CAPTURE_RESTART_MODE     (0x09)
#define COMPARATOR_COUNTER_MODE  (0x0A)

//////////////////////////////////////////////////////////////////////////
// Prescalar Values
#define PRESCALAR_1              (0x00)            // Divide by 1
#define PRESCALAR_2              (0x08)            // Divide by 2
#define PRESCALAR_4              (0x10)            // Divide by 4
#define PRESCALAR_8              (0x18)            // Divide by 8
#define PRESCALAR_16             (0x20)            // Divide by 16
#define PRESCALAR_32             (0x28)            // Divide by 32
#define PRESCALAR_64             (0x30)            // Divide by 64
#define PRESCALAR_128            (0x38)            // Divide by 128

//////////////////////////////////////////////////////////////////////////
// Directives for various Timer Registers
#define PWM_HIGH                 (0x00)
#define PWM_LOW                  (0x04)

#define POLARITY_HIGH            (0x40)
```

```
#define POLARITY_LOW             (0x00)

//////////////////////////////////////////////////////////////////////////
// Directives for Timer Interrupt Configurations
// Not applicable to F64xx and F0822 devices!
#define TICONFIG_RELOAD_COMPARE_INPUT  (0x00)
#define TICONFIG_CAPTURE_DEASSERT      (0x40)
#define TICONFIG_RELOAD_COMPARE        (0x60)

//////////////////////////////////////////////////////////////////////////
// To setup port output for timer0, use the macro defined below:
#define Enable_Timer0_OutputPin        (PAAF |= 0x02)

//////////////////////////////////////////////////////////////////////////
// To setup port input for timer0, use the macro defined below:
#define Enable_Timer0_InputPin         (PAAF |= 0x01)

//////////////////////////////////////////////////////////////////////////
// To change the interrupt priority of the timer, use the routines
// illustrated below:
void TMR0_Priority_Low(void)
{
   IRQ0ENH &= ~0x20;
   IRQ0ENL |= 0x20;
}

void TMR0_Priority_Nominal(void)
{
   IRQ0ENH |= 0x20;
   IRQ0ENL &= ~0x20;
}

void TMR0_Priority_High(void)
{
   IRQ0ENH |= 0x20;
   IRQ0ENL |= 0x20;
}
```

➢ ***Note:*** *Ensure that changing Timer0 priorities does not change the priorities of other interrupt resources.*

The source code provided along with this document is designed to meet the requirements for all Z8 Encore! family of microcontrollers. To switch between the different devices, a checking is done according to the CPU family selected under the project settings. This condition checking routine can be found on several places in the source code, as in the example below:

```
………<code here> …………
#if   defined(_Z8ENCORE_XP_F082A_SERIES) ||
      defined(_Z8ENCORE_XP_F0823_SERIES) ||
      defined(_Z8ENCORE_F1680)
T0CTL0 = TICONFIG_RELOAD_COMPARE;      // Set Timer to interrupt on
                                       // reload/compare events only
#endif
………<more code here> …………
```

Initialization Routines for the Different Timer Operating Modes

This section describes several ready-to-use C routines for the different operating modes of the timer. These routines can be used in general for the Z8 Encore! family of microcontrollers with slight modifications done on the code to fit user requirements. For assembly routines, please refer to the appendix section.

The Z8 Encore! Timers are 16-bit up-counter timers. Minimum timeout delay is set by loading the value `0001h` into the Timer Reload High and Low Byte Registers and setting the prescale value to 1. Maximum timeout delay is set by loading the value `0000h` into the Timer Reload High and Low Byte Registers and setting the prescale value to 128. If the timer reaches `FFFFh`, the timer rolls over to `0000h` and continues counting.

For all the Timer modes of operation, the following are chosen by default:

- Device: Z8 Encore! 28-pin controller
- Timer interrupt priority is high
- Timer clock source is the system clock
- Timer output polarity is low
- Timer counts up to 1msec, if necessary

The various operating modes of the Timer are explained below.

➢ ***Note:** The timer clock source can be configured to be either the system clock or an external peripheral clock for the Z8 Encore! F1680 device only. For the remainder of the Z8 Encore! devices, timer clock source is always the system clock.*

1. ONE-SHOT Mode

In ONE-SHOT mode, the timer counts up to the 16-bit reload value stored in the Timer Reload High and Low Byte Registers. Upon reaching the reload value, the timer generates an interrupt and the count value in the Timer High and Low Byte Registers are reset to `0001h`. The timer automatically disables and stops counting. Also, if the timer output alternate function is enabled, the Timer Output pin will change its state for one system clock cycle.

The figure below illustrates the output generated when timer in running under ONE-SHOT mode. A short pulse is generated just before the timer is started, then the timer is automatically disabled after 1msec.

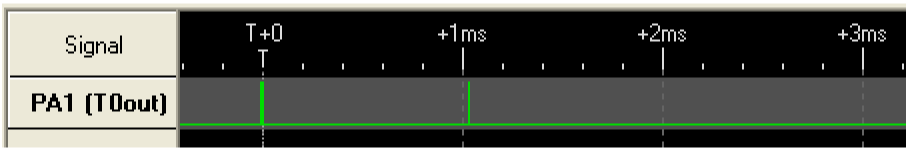

Figure 2 Output Generated in ONE-SHOT Mode

The timer period is given by the following equation:

$$\text{ONE-SHOT Mode Time-Out Period (s)} = \frac{(\text{Reload Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

For example, the reload value for a timeout period of 1 msec is calculated as follows:

Reload value = ((1 msec * system clock frequency) / prescalar) + start value
Reload value = ((0.001 * 18432000) / 4) + 1 = 0x1201

The Timer0 Reload High Register (`T0RH`) must be loaded with a value of `12h` and the Timer0 Reload Low Register (`T0RL`) must be loaded with a value of `01h`.

The code given below illustrates how to configure the timer in ONE-SHOT mode. The output generated by this code is shown in Figure 2 above.

```
//////////////////////////////////////////////////////////////////////////
void TMR0_INIT_XP_OneShot(void)
{
   DI();

   T0CTL1 = ONE_SHOT_MODE | PRESCALE;  // Disable timer, ONE-SHOT mode,
                                       // prescale = 4
   T0H = (STARTCOUNT >> 8);            // Set starting count value = 0001h
   T0L = (STARTCOUNT & 0x00FF);
   T0RH = (RELOAD >> 8);               // Set reload value = 1201h (1msec)
   T0RL = (RELOAD & 0x00FF);

   #ifdef TIMER0_ISR_ENABLE
   TMR0_Priority_High();               // Set timer interrupt priority
   #endif // TIMER0_ISR_ENABLE

   #ifdef TIMER0_OUTPUTPIN_ENABLE
   Enable_Timer0_OutputPin;            // Configure Port A for
                                       // alternate function operation
   #endif // TIMER0_OUTPUTPIN_ENABLE

   T0CTL1 |= ENABLE_TIMER;             // Enable timer

   EI();
}
//////////////////////////////////////////////////////////////////////////
```

2. CONTINUOUS Mode

In CONTINUOUS mode, the timer counts up to the 16-bit reload value stored in the Timer Reload High and Low Byte Registers. Upon reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte Registers are reset to `0001h`, and counting resumes. Also, if the timer output alternate function is enabled, the Timer Output pin will change its state.

The figure below illustrates state of the timer output pin and the interrupt generated when running under CONTINUOUS mode. The timer continuously counts at 1msec interval.

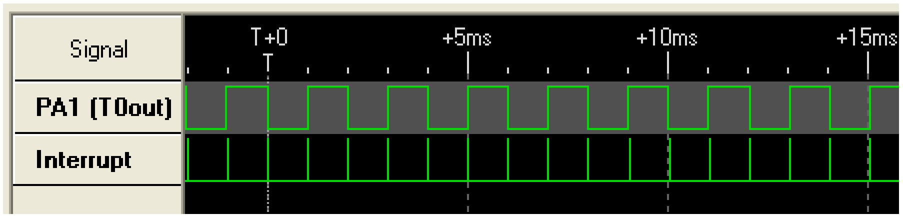

Figure 3 Output Generated in CONTINUOUS Mode

The timer period is given by the following equation:

$$\text{CONTINUOUS Mode Time-Out Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

For example, the reload value for a timeout period of 1 msec is calculated as follows:

Reload value = (1 msec * system clock frequency) / prescalar
Reload value = (0.001 * 18432000) / 4 = 0x1200

The code below illustrates how to configure the timer in COTINUOUS mode. The output generated by this code is shown in Figure 3.

```
//////////////////////////////////////////////////////////////////////////
void TMR0_INIT_XP_Continuous(void)
{
   DI();

   T0CTL1 = CONTINUOUS_MODE | PRESCALE;  // Disable timer
                                        // CONTINUOUS mode, prescale = 4
   T0H = (STARTCOUNT >> 8);             // Set starting count value = 0001h
   T0L = (STARTCOUNT & 0x00FF);
   T0RH = (RELOAD >> 8);                // Set reload value = 1201h (1msec)
   T0RL = (RELOAD & 0x00FF);

   #ifdef TIMER0_ISR_ENABLE
   TMR0_Priority_High();                // Set timer interrupt priority
   #endif // TIMER0_ISR_ENABLE

   #ifdef TIMER0_OUTPUTPIN_ENABLE
   Enable_Timer0_OutputPin;             // Configure Port A for
                                        // alternate function operation
   #endif // TIMER0_OUTPUTPIN_ENABLE

   T0CTL1 |= ENABLE_TIMER;              // Enable timer

   EI();
}
//////////////////////////////////////////////////////////////////////////
```

3. COUNTER Mode

In COUNTER mode, the timer counts the number of input transitions from the timer input pin. The `TPOL` bit selects whether the count occurs at the rising or falling edge of the timer input signal. Upon reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte Registers are reset to `0001h`, and counting resumes. Also, if the timer output alternate function is enabled, the Timer Output pin will change its state. In COUNTER mode, the prescalar is disabled.

The figure below illustrates state of the timer output pin and the interrupt generated when running under COUNTER mode. The timer counts input transitions (up to 5 counts, in this example) then generates an interrupt and/or toggles timer output pin.

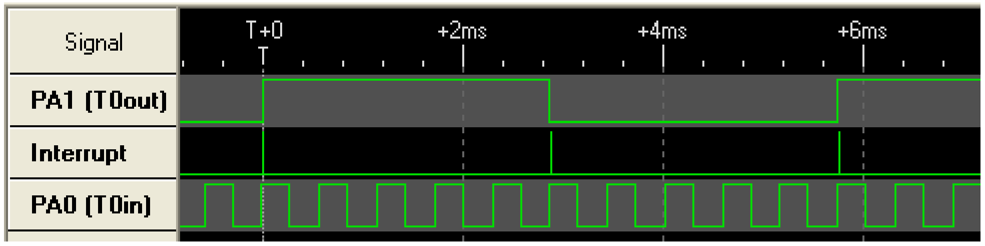

Figure 4 Output Generated in COUNTER Mode

The number of Timer Input transitions since the timer started is given by the following equation:

$$\text{COUNTER Mode Timer Input Transitions} = \text{Current Count Value} - \text{Start Value}$$

The code below illustrates how to configure the timer in COUNTER mode. The output generated by this code is shown in Figure 4 above.

```
////////////////////////////////////////////////////////////////////////////
void TMR0_INIT_XP_Counter(void)
{
   DI();

   T0CTL1 = COUNTER_MODE;                  // Disable timer, COUNTER mode
   T0H = (STARTCOUNT >> 8);                // Set starting count value = 0001h
   T0L = (STARTCOUNT & 0x00FF);
   T0RH = (COUNTER >> 8);                  // Set reload value = 0005h
   T0RL = (COUNTER & 0x00FF);

   #ifdef TIMER0_ISR_ENABLE
   TMR0_Priority_High();                   // Set timer interrupt priority
   #endif // TIMER0_ISR_ENABLE

   #ifdef TIMER0_OUTPUTPIN_ENABLE
   Enable_Timer0_OutputPin;                // Configure Port A for
                                           // alternate function operation
   #endif // TIMER0_OUTPUTPIN_ENABLE

   Enable_Timer0_InputPin;                 // Enable timer input pin

   T0CTL1 |= ENABLE_TIMER;                 // Enable timer
   EI();
}
///////////////////////////////////////////////////////////////////////////
```

4. COMPARATOR COUNTER Mode

In COMPARATOR COUNTER mode, the timer counts the number of input transitions from the analog comparator output. The `TPOL` bit selects whether the count occurs at the rising or falling edge of the comparator output signal. Upon reaching the reload value, the timer generates an interrupt, timer count value in the Timer High and Low Byte Registers are reset to `0001h`, and counting resumes. Also, if the timer output alternate function is enabled, the Timer Output pin will change its state.

The figure below illustrates the state of the timer output pin and the interrupt generated when running under COMPARATOR COUNTER mode. Cout reflects the comparator output state. The timer simply acts as a counter for the comparator output.

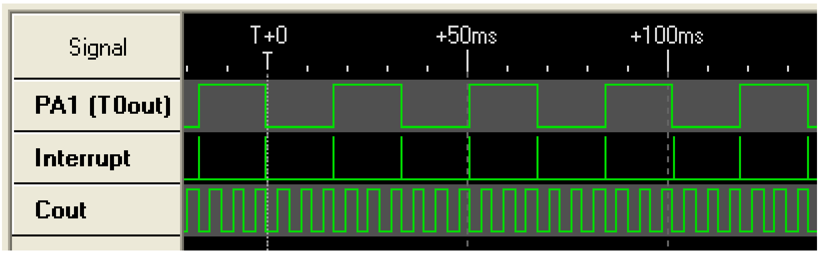

Figure 5 Output Generated in COMPARATOR COUNTER Mode

The number of comparator output transitions since the timer started is given by the following equation:

$$\text{Comparator Output Transitions} = \text{Current Count Value} - \text{Start Value}$$

The code below illustrates how to configure the timer in COMPARATOR COUNTER mode. Make sure that comparator is initialized properly. The output generated by this code is illustrated in Figure 5.

```
///////////////////////////////////////////////////////////////////////////
void TMR0_INIT_XP_ComparatorCounter(void)
{
   DI();

   // Comparator0 must be initialized prior to timer initialization.

   T0CTL1 = COMPARATOR_COUNTER_MODE & 0x07;  // Disable timer
                                    // COMPARATOR COUNTER mode
   T0CTL0 = (COMPARATOR_COUNTER_MODE << 4) & 0x80;
   T0H = (STARTCOUNT >> 8);          // Set starting count value = 0001h
   T0L = (STARTCOUNT & 0x00FF);
   T0RH = (COUNTER >> 8);            // Set reload value = 0005h
   T0RL = (COUNTER & 0x00FF);
   #ifdef TIMER0_ISR_ENABLE
   TMR0_Priority_High();            // Set timer interrupt priority
   #endif // TIMER0_ISR_ENABLE

   #ifdef TIMER0_OUTPUTPIN_ENABLE
   Enable_Timer0_OutputPin;         // Configure Port A for
                                    // alternate function operation
   #endif // TIMER0_OUTPUTPIN_ENABLE

   T0CTL1 |= ENABLE_TIMER;          // Enable timer
   EI();
}
//////////////////////////////////////////////////////////////////////////
```

➢ ***Note:*** *Comparator Counter mode is not supported in Z8F0822 and Z8F64xx devices.*

5. PWM SINGLE OUTPUT Mode

In PWM SINGLE OUTPUT mode, the timer outputs a pulse-width modulator (PWM) signal through a dedicated GPIO port pin. The timer first counts up to the 16-bit PWM match value stored in Timer PWM High and Low Byte Registers. Upon reaching the PWM match value, Timer Output toggles. The timer continues counting until it reaches the reload value stored in Timer Reload High and Low Byte Registers. Upon reaching the timer reload value, the timer generates an interrupt, the Timer Output toggles, the count value in the Timer High and Low Byte Registers are reset to `0001h`, and counting resumes.

The initial state of the Timer Output pin is determined by the `TPOL` bit. If TPOL is set to 1, the Timer Output signal begins as high (1) and transitions to low (0) when PWM match value is reached, and then returns to a high (1) state when reload occurs.

On the other hand, if TPOL is set to 0, the Timer Output signal begins as low (0) and transitions to high (1) when PWM match value is reached, and then returns to a low (0) state when reload occurs.

The figure below illustrates the output generated when running under PWM Single Output mode. An output high time ratio of ~70% is set for this example.

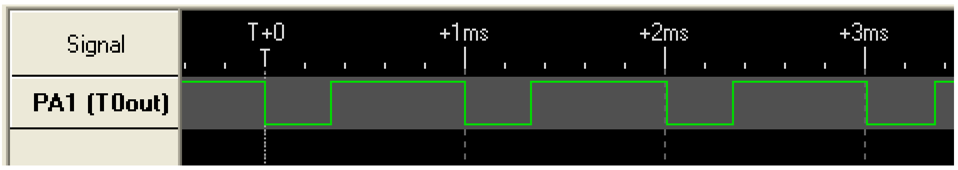

Figure 6 Output Generated in PWM SINGLE OUTPUT Mode

The PWM period is given by the following equation:

$$\text{PWM Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

If TPOL is set to 1, the ratio of the PWM output high time to the total period is given by the following equation:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{PWM Value}}{\text{Reload Value}} \times 100$$

If TPOL is set to 0, the ration of the PWM output high time to the total period is given by the following equation:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{Reload Value} - \text{PWM Value}}{\text{Reload Value}} \times 100$$

The code below illustrates how to configure the timer in PWM SINGLE OUTPUT mode. The output generated by this code is shown in Figure 6.

```
//////////////////////////////////////////////////////////////////////////
void TMR0_INIT_XP_PWM(void)
{
   DI();

   T0CTL1 = PWM_MODE | PRESCALE;        // Disable timer, PWM mode
   T0H = (STARTCOUNT >> 8);             // Set starting count value = 0001h
   T0L = (STARTCOUNT & 0x00FF);
   T0RH = (RELOAD >> 8);                // Set reload value = 1201h (1msec)
   T0RL = (RELOAD & 0x00FF);
   T0PWMH = (RELOAD / 3) >> 8;          // Set PWM value (~70% high time)
   T0PWML = (RELOAD / 3) & 0x00FF;

   #ifdef TIMER0_ISR_ENABLE
   TMR0_Priority_High();                // Set timer interrupt priority
   #endif // TIMER0_ISR_ENABLE

   Enable_Timer0_OutputPin;             // Configure Port A for
                                        // alternate function operation
   T0CTL1 |= ENABLE_TIMER;              // Enable timer

   EI();
}
//////////////////////////////////////////////////////////////////////////
```

➢ ***Note:*** *The PWM mode of the Z8F64xx Series is the same as the PWM Single Output mode of the rest of Z8 Encore! devices.*

6. PWM DUAL OUTPUT Mode

In PWM DUAL OUTPUT mode, the timer outputs a pulse-width modulator (PWM) output signal pair (basic PWM and its complement) thru two GPIO port pins. The timer first counts up to the 16-bit PWM match value stored in the Timer PWM High and Low Byte Registers. When the timer count value matches the PWM value, the timer output toggles. The timer continues counting until it reaches the reload value stored in the Timer Reload High and Low Byte Registers. Upon reaching the reload value, the timer generates an interrupt, the count value in the

Timer High and Low Byte Registers are reset to `0001h`, and counting resumes.

If the `TPOL` bit is set to 1, the timer output signal begins as high and transitions to low when the timer value matches the PWM value. The timer output signal returns to high after the timer reaches the reload value and is reset to `0001h`.

If the `TPOL` bit is set to 0, the timer output signal begins as low and transitions to high when the timer value matches the PWM value. The timer output signal returns to low after the timer reaches the reload value and is reset to `0001h`.

The timer also generates a second PWM output signal, the timer output complement. The timer output complement is the complement of the timer output PWM signal. A programmable deadband delay can be configured to time delay (0 to 128 system clock cycles) PWM output transitions on these two pins from a low to a high state. This ensures a time gap between the deassertion of one PWM output to the assertion of its complement.

The figure below displays the output generated by the timer when running under PWM DUAL OUTPUT mode. The timer output is simply reversed for the timer output complement.

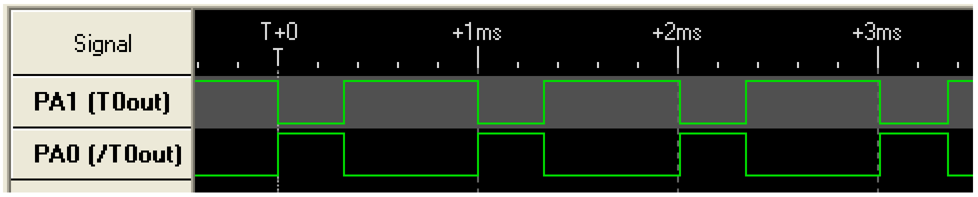

Figure 7 Output Generated in PWM DUAL OUTPUT Mode

The PWM period is represented by the following equation:

$$\text{PWM Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

If an initial starting value other than 0001h is loaded into the Timer High and Low Byte Registers, the ONE-SHOT mode equation determines the first PWM timeout period.

If `TPOL` is set to 0, the ratio of the PWM output high time to the total period is represented by the following equation:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{Reload Value} - \text{PWM Value}}{\text{Reload Value}} \times 100$$

If `TPOL` is set to 1, the ratio of the PWM output high time to the total period is represented by the following equation:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{PWM Value}}{\text{Reload Value}} \times 100$$

The code below illustrates how to configure the timer in PWM DUAL OUTPUT mode. The output generated by this code is shown in Figure 7.

```
///////////////////////////////////////////////////////////////////////////
void TMR0_INIT_XP_PWM_DualOutput(void)
```

```
{
   DI();

   T0CTL0 = (PWM_DUAL_OUTPUT_MODE << 4) & 0x80;// Set PWM Dual Output mode
   T0CTL1 = (PWM_DUAL_OUTPUT_MODE & 0x07) | PRESCALE;    // Disable timer
                                    // PWM Dual Output mode
   T0H = (STARTCOUNT >> 8);         // Set starting count value = 0001h
   T0L = (STARTCOUNT & 0x00FF);
   T0RH = (RELOAD >> 8);            // Set reload value = 1201h (1msec)
   T0RL = (RELOAD & 0x00FF);
   T0PWMH = (RELOAD / 3) >> 8;      // Set PWM value
   T0PWML = (RELOAD / 3) & 0x00FF;  // (approx 30% duty cycle)

   #ifdef TIMER0_ISR_ENABLE
   TMR0_Priority_High();            // Set timer interrupt priority
   #endif // TIMER0_ISR_ENABLE

   Enable_Timer0_InputPin;          // Configure Port A for
   Enable_Timer0_OutputPin;         // alternate function operation

   T0CTL1 |= ENABLE_TIMER;          // Enable timer

   EI();
}
///////////////////////////////////////////////////////////////////////////
```

➢ ***Note:*** *PWM Dual Output mode is not supported in Z8F0822 and Z8F64xx devices.*

7. CAPTURE Mode

In CAPTURE mode, the current timer count value is recorded when the desired timer input signal transition occurs. The TPOL bit determines if the capture occurs on a rising edge or a falling edge of the timer input signal. The capture count value is written to the Timer PWM High and Low Byte Registers. When a capture event occurs, an interrupt is generated and the timer continues counting. Also, an interrupt is generated when the timer reaches the reload value stored in the Timer Reload High and Low Byte Registers.

The figure below illustrates the output generated when the timer is running under CAPTURE mode. A short pulse is generated inside an interrupt service routine, where the capture count value is also read.

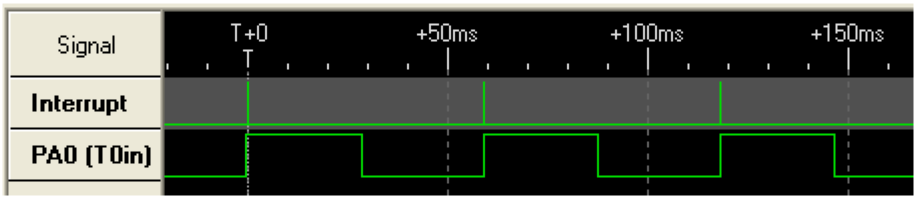

Figure 8 Output Generated in CAPTURE Mode

The elapsed time from timer start to capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time (s)} = \frac{(\text{Capture Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

The code below illustrates how to configure the timer in CAPTURE mode. The output generated in this code is shown in Figure 8 above.

```
///////////////////////////////////////////////////////////////////////////
void TMR0_INIT_XP_Capture(void)
{
   DI();

   T0CTL1 = CAPTURE_MODE | PRESCALE;   // Disable timer, CAPTURE mode
   T0H = (STARTCOUNT >> 8);            // Set starting count value = 0001h
   T0L = (STARTCOUNT & 0x00FF);
   T0RH = (0xFFFF >> 8);               // Set reload value
   T0RL = (0xFFFF & 0x00FF);
   T0PWMH = 0x00;                      // Clear the PWM registers
   T0PWML = 0x00;

   #ifdef TIMER0_ISR_ENABLE
   TMR0_Priority_High();               // Set timer interrupt priority
   #endif // TIMER0_ISR_ENABLE

   Enable_Timer0_InputPin;             // Enable timer input pin

   T0CTL1 |= ENABLE_TIMER;             // Enable timer
   EI();
}
///////////////////////////////////////////////////////////////////////////
```

8. CAPTURE RESTART Mode

In CAPTURE RESTART mode, the current timer count value is recorded when the desired timer input transition occurs. The capture count value is written to the timer PWM High and Low Byte Registers. The TPOL bit determines if the capture occurs on a rising edge or a falling edge of the timer input signal. When the capture event occurs, an interrupt is generated and the count value in the Timer High and Low Byte Registers are reset to 0001h, and counting resumes. The INPCAP bit is set to indicate that the timer interrupt occurred due to an input capture event.

If no capture event occurs, the timer counts up to the 16-bit compare value stored in the Timer Reload High and Low Byte Registers. Upon reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte Registers are reset to 0001h, and counting resumes. The INPCAP bit is cleared to indicate that the timer interrupt is not caused by an input capture.

The elapsed time from timer start to capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time (s)} = \frac{(\text{Capture Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

The code below demonstrates how to configure the timer in CAPTURE RESTART mode.

```
///////////////////////////////////////////////////////////////////////////
void TMR0_INIT_XP_CaptureRestart(void)
{
   DI();

   T0CTL0 = (CAPTURE_RESTART_MODE << 4) & 0x80;    // Disable timer,
                                       // CAPTURE RESTART mode,
   T0CTL1 = (CAPTURE_RESTART_MODE & 0x07) | PRESCALE; // prescale = 4
   T0H = (STARTCOUNT >> 8);            // Set starting count value = 0001h
   T0L = (STARTCOUNT & 0x00FF);
   T0RH = (0xFFFF >> 8);               // Set reload value
   T0RL = (0xFFFF & 0x00FF);
   T0PWMH = 0x00;                      // Clear the PWM registers
   T0PWML = 0x00;

   #ifdef TIMER0_ISR_ENABLE
   TMR0_Priority_High();               // Set timer interrupt priority
   #endif // TIMER0_ISR_ENABLE

   Enable_Timer0_InputPin;             // Enable timer input pin

   T0CTL1 |= ENABLE_TIMER;             // Enable timer
   EI();
}
///////////////////////////////////////////////////////////////////////////
```

➢ ***Note:*** *Capture Restart mode is not supported in Z8F0822 and Z8F64xx devices.*

9. COMPARE Mode

In COMPARE mode, the timer counts up to the 16-bit maximum compare value stored in the Timer Reload High and Low Byte Registers. Upon reaching the compare value, the timer generates an interrupt and counting continues (the timer count value is not reset to `0001h`). If the timer reaches `FFFFh`, the timer rolls over to `0000h` and continues counting.

Also, if the timer output alternate function is enabled, the timer output pin changes state upon compare.

The compare time can be calculated using the following equation:

$$\text{COMPARE Mode Time (s)} = \frac{(\text{Compare Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

The code below illustrates how to configure the timer in COMPARE mode.

```
///////////////////////////////////////////////////////////////////////////
void TMR0_INIT_XP_Compare(void)
```

```
{
   DI();

   T0CTL1 = COMPARE_MODE | PRESCALE;   // Disable timer
                                       // COMPARE mode, prescale = 4
   T0H = (STARTCOUNT >> 8);            // Set starting count value = 0001h
   T0L = (STARTCOUNT & 0x00FF);
   T0RH = (RELOAD >> 8);               // Set compare
   T0RL = (RELOAD & 0x00FF);

   #ifdef TIMER0_ISR_ENABLE
   TMR0_Priority_High();               // Set timer interrupt priority
   #endif // TIMER0_ISR_ENABLE

   #ifdef TIMER0_OUTPUTPIN_ENABLE
   Enable_Timer0_OutputPin;            // Configure Port A for
                                       // alternate function operation
   #endif // TIMER0_OUTPUTPIN_ENABLE

   T0CTL1 |= ENABLE_TIMER;             // Enable timer

   EI();
}
//////////////////////////////////////////////////////////////////////////
```

10. GATED Mode

In GATED mode, the timer counts only when the timer input signal is in its active state, as determined by the `TPOL` bit. When the timer input signal is asserted, the timer counts up to the 16-bit reload value stored in the Timer Reload High and Low Byte Registers. Upon reaching the reload value, the timer generates an interrupt and the count value in the Timer High and Low Byte Registers are reset to `0001h` and counting continues (assuming the timer input signal is still asserted). A timer interrupt is generated when the timer input signal is deasserted or when a reload occurs. To determine if a timer input signal deassertion generated the interrupt, read the associated GPIO input value and compare the value stored in the `TPOL` bit.

Also, if the timer output alternate function is enabled, the timer output pin changes state at timer reset.

The figure below displays the output generated when the timer is configured for GATED mode. In this example, the timer counts only when timer input is in a high state. The timer can also be configured to run only when timer input is in a low state by setting the proper value in the `TPOL` bit.

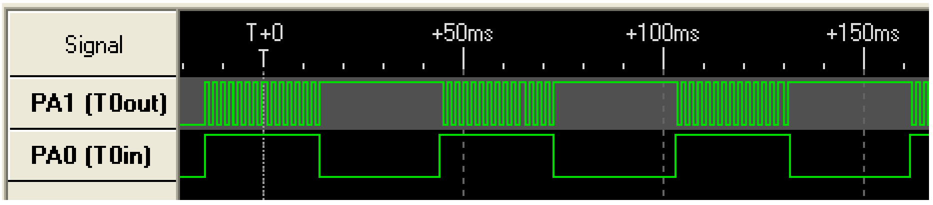

Figure 9 Output Generated in GATED Mode

The code below illustrates how to configure the timer in GATED mode. The output generated by this code is shown in Figure 9 above.

```
////////////////////////////////////////////////////////////////////////////
void TMR0_INIT_XP_Gated(void)
{
   DI();

   T0CTL1 = GATED_MODE | PRESCALE;     // Disable timer
                                       // GATED mode, prescale = 4
   T0H = (STARTCOUNT >> 8);            // Set starting count value = 0001h
   T0L = (STARTCOUNT & 0x00FF);
   T0RH = (RELOAD >> 8);               // et reload value = 1201h (1msec)
   T0RL = (RELOAD & 0x00FF);

   #ifdef TIMER0_ISR_ENABLE
   TMR0_Priority_High();               // Set timer interrupt priority
   #endif // TIMER0_ISR_ENABLE

   #ifdef TIMER0_OUTPUTPIN_ENABLE
   Enable_Timer0_OutputPin;            // Configure Port A for
                                       // alternate function operation
   #endif // TIMER0_OUTPUTPIN_ENABLE

   Enable_Timer0_InputPin;             // Enable timer input pin

   T0CTL1 |= ENABLE_TIMER;             // Enable timer

   EI();
}
////////////////////////////////////////////////////////////////////////////
```

11. CAPTURE/COMPARE Mode

In CAPTURE/COMPARE mode, the timer begins counting on the first timer input transition. The desired transition (rising edge or falling edge) is set by the `TPOL` bit. Every subsequent desired transition of the timer input signal captures the current count value. The capture value is written to the Timer PWM High and Low Byte Registers. When the capture event occurs, an interrupt is generated, the count value in the Timer High and Low Byte Registers are reset to `0001h`, and counting resumes.

If no capture event occurs, the timer counts up to the 16-bit compare value stored in the Timer

Reload High and Low Byte Registers. Upon reaching the compare value, the timer generates an interrupt, the count value in the Timer High and Low Byte Registers are reset to `0001h`, and counting resumes.

In CAPTURE/COMPARE mode, the elapsed time from timer start to capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time (s)} = \frac{(\text{Capture Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

The code below illustrates how to configure the timer for CAPTURE/COMPARE mode:

```
/////////////////////////////////////////////////////////////////////////////
void TMR0_INIT_XP_CaptureCompare(void)
{
   DI();

   T0CTL1 = CAPTURE_COMPARE_MODE | PRESCALE; // Disable timer,
                                 // CAPTURE COMPARE mode, prescale = 4
   T0H = (STARTCOUNT >> 8);            // Set starting count value = 0001h
   T0L = (STARTCOUNT & 0x00FF);
   T0RH = (RELOAD >> 8);               // Set compare value
   T0RL = (RELOAD & 0x00FF);

   #ifdef TIMER0_ISR_ENABLE
   TMR0_Priority_High();               // Set timer interrupt priority
   #endif // TIMER0_ISR_ENABLE

   Enable_Timer0_InputPin;             // Enable timer input pin

   T0CTL1 |= ENABLE_TIMER;             // Enable timer

   EI();
}
/////////////////////////////////////////////////////////////////////////////
```

12. TRIGGERED ONE-SHOT Mode

In TRIGGERED ONE-SHOT mode, the timer is idle until a trigger is received from the timer input pin. The TPOL bit selects whether the trigger occurs on the rising or falling edge of the timer input signal. Following the trigger, the timer counts up to the 16-bit reload value stored I the Timer Reload High and Low Byte Registers. On reaching the reload value, the timer outputs a pulse on the timer output pin, generates an interrupt, and the count value in the Timer High and Low Byte Registers are reset to 0001h. The TPOL bit also sets the polarity of the output pulse.

The figure below illustrates the output generated by the timer when configured to run in TRIGGERED ONE-SHOT mode. A one clock cycle pulse is generated at the timer output pin 1msec after the trigger is detected. The same is performed inside the interrupt routine.

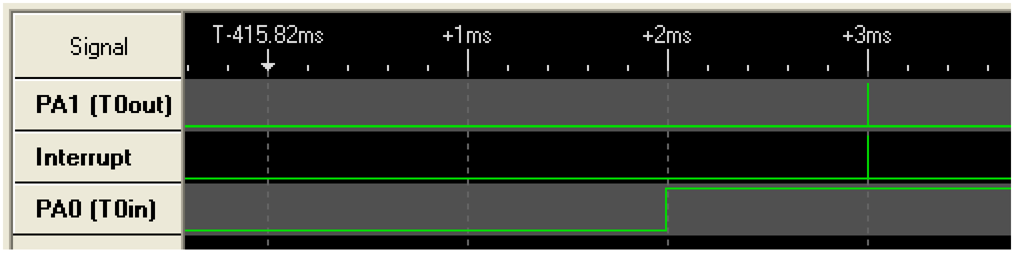

Figure 10 Output Generated in TRIGGERED ONE-SHOT Mode

In TRIGGERED ONE-SHOT mode, the timer clock source is always the system clock. The timer period is always given by the following equation:

$$\text{Triggered ONE-SHOT Mode Time-Out Period (s)} = \frac{(\text{Reload Value - Start Value}) \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

The code below illustrates how to configure the timer for TRIGGERED ONE-SHOT MODE. The output generated by this code is shown in Figure 10.

```
//////////////////////////////////////////////////////////////////////////
void TMR0_INIT_XP_TriggeredOneShot(void)
{
   #ifdef _Z8ENCORE_F1680
   DI();

   T0CTL0 = (TRIGGERED_ONESHOT_MODE << 4) & 0x80;  // Disable timer
                                       // TRIGGERED ONE-SHOT mode
   T0CTL1 = (TRIGGERED_ONESHOT_MODE & 0x07) | PRESCALE;  // prescale = 4
   T0H = (STARTCOUNT >> 8);            // Set starting count value = 0001h
   T0L = (STARTCOUNT & 0x00FF);
   T0RH = (RELOAD >> 8);               // Set reload value = 1201h (1msec)
   T0RL = (RELOAD & 0x00FF);

   #ifdef TIMER0_ISR_ENABLE
   T0CTL0 |= TICONFIG_RELOAD_COMPARE;  // Interrupt on reload event only
   TMR0_Priority_High();               // Set timer interrupt priority
   #endif // TIMER0_ISR_ENABLE

   Enable_Timer0_OutputPin;            // Enable timer output pin
   Enable_Timer0_InputPin;             // Enable timer input pin

   T0CTL1 |= ENABLE_TIMER;             // Enable timer

   EI();
   #endif // _Z8ENCORE_F1680
}
//////////////////////////////////////////////////////////////////////////
```

➢ ***Note:*** *TRIGGERED ONE-SHOT mode is applicable only on F1680.*

13. DEMODULATION Mode

In DEMODULATION mode, the timer begins counting on the first timer input transition. The desired transition (rising or falling edge) is set by the `TPOL` bit. The timer counts up to the 16-bit reload value stored in the Timer Reload High and Low Byte Registers.

Every subsequent desired transition of the timer input signal captures the current count value. The capture value is written to the Timer PWM0 High and Low Byte Registers for rising edges of the timer input signal. For falling edges, the capture count value is written to the Timer PWM1 High and Low Byte Registers. The `TPOL` bit determines if the capture occurs at the rising or falling edge of the timer input signal. If `TPOLHI` is set, capture is done on both the rising and falling edges of the timer input signal.

On capture event, an interrupt is generated and the timer continues counting. The corresponding event flag bit `PWMxEF` is set to indicate if the timer interrupt is due to an input capture event.

If no capture event occurs, the timer counts up to the 16-bit reload value stored in the Timer Reload High and Low Byte Registers. Upon reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte Registers are reset to `0001h`, and counting resumes. The `RTOEF` bit is set to indicate if the timer interrupt is due to a reload event.

The figure below illustrates the output generated when in DEMODULATION mode. This example shows timer interrupts on capture event for both rising and falling edges of the timer input signal.

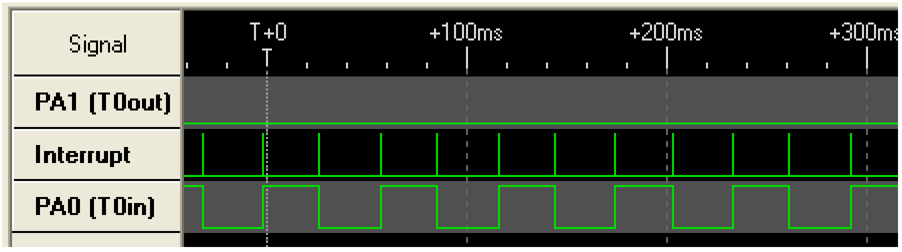

Figure 11 Output Generated in DEMODULATION Mode

In DEMODULATION mode, the elapsed time from timer start to capture event is defined by the following equation:

$$\text{Capture Elapsed Time (s)} = \frac{(\text{Capture Value - Start Value}) \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

The code below illustrates how to configure the timer in DEMODULATION mode. The output generated by this code is shown in Figure 11 above.

```
///////////////////////////////////////////////////////////////////////////
void TMR0_INIT_XP_Demodulation(void)
{
   #ifdef _Z8ENCORE_F1680
   DI();

   T0CTL0 = (DEMODULATION_MODE << 4) & 0x80; // Disable timer
```

```
    T0CTL1 = (DEMODULATION_MODE & 0x07) | PRESCALE; // DEMODULATION mode
                                          // prescale = 4

    T0H = (STARTCOUNT >> 8);            // Set starting count value = 0001h
    T0L = (STARTCOUNT & 0x00FF);
    T0RH = (RELOAD >> 8);               // Set reload value = 1201h (1msec)
    T0RL = (RELOAD & 0x00FF);
    T0PWM0H = 0x00;                     // Clear the timer
    T0PWM0L = 0x00;                     // PWM0 & PWM1 registers
    T0PWM1H = 0x00;
    T0PWM1L = 0x00;

    #ifdef TIMER0_ISR_ENABLE
    T0CTL2 |= TPOLHI;                   // Capture on both edges
    T0CTL0 |= TICONFIG_CAPTURE_DEASSERT;      // Set timer to interrupt
                                        // on capture event only

    TMR0_Priority_High();               // Set timer interrupt priority
    #endif // TIMER0_ISR_ENABLE

    Enable_Timer0_InputPin;             // Enable timer input pin

    T0CTL1 |= ENABLE_TIMER;             // Enable timer

    EI();
    #endif // _Z8ENCORE_F1680
}
///////////////////////////////////////////////////////////////////////////
```

➢ ***Note:** DEMODULATION mode is applicable only on F1680.*

Testing the Z8 Encore! Timer in Different Modes

The different timer routines for the Z8 Encore! XP timers are tested. The figure below illustrates the setup for testing the timer operating modes. A function generator (BK Precision 4040A) is used to provide the required input for the timer. A logic analyzer (Intronix LA1034) is used for capturing the timer output. The routines provided with this document are tested on the following Z8 Encore! XP devices for each operating mode:

- Z8F08A28100KITG + Z8F082ASJ020SG
- Z8F08A28100KITG + Z8F0423SJ005SG
- Z8F08200100KITG + Z8F0822SJ020SC
- Z8F16800128ZCOG + Z8F1680SJ020SG
- Z8F64200100KITG + Z8F6423FT020SG

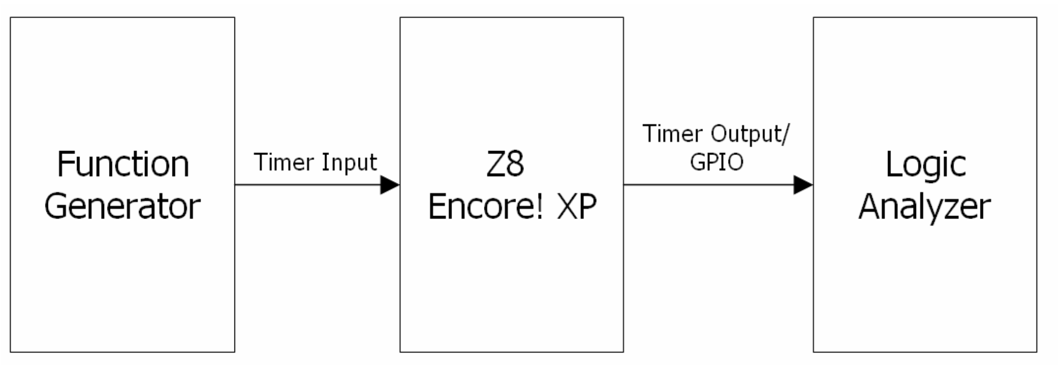

Figure 12 Setup to Test Timer Mode Functionality

➢ ***Caution:*** *When testing for the timers, it is advisable to test directly from the Z8 Encore! device pins. External components attached to the pins affect the state of timer input/output at JP2 pins.*

Table 11 Test Results for the Different Timer Operating Modes

Operating Modes	F0830	F083A	F082A	F0823	F0822	F1680	F64xx
One Shot	passed	passed	passed	passed	passed	passed	passed
Triggered One Shot	--	--	--	--	--	passed	--
Continuous	passed	passed	passed	passed	passed	passed	passed
Counter	passed	passed	passed	passed	passed	passed	passed
Comparator Counter	passed	passed	passed	passed	--	passed	--
PWM Single Output	passed	passed	passed	passed	passed	passed	passed
PWM Dual Output	passed	passed	passed	passed	--	passed	--
Capture	passed	passed	passed	passed	passed	passed	passed
Capture Restart	passed	passed	passed	passed	--	passed	--
Compare	passed	passed	passed	passed	passed	passed	passed
Gated	passed	passed	passed	passed	passed	passed	passed
Capture/Compare	passed	passed	passed	passed	passed	passed	passed
Demodulation	--	--	--	--	--	passed	--

Summary

This application note provides ready-to-use routines on using the timer peripheral of the Z8 Encore! Microcontrollers. It describes how to configure the timer in different modes of operation and use it as per your requirements. Appropriate C preprocessor directives and routines are provided to switch between different operating modes of the timer and to change the interrupt priorities.

Appendix – Assembly Routines

This section lists down the equivalent assembly routines of the codes discussed in this document.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TMR0_INIT_XP_OneShot:
   DI
   ANDX T0CTL1,#DISABLE_TIMER          ; Disable timer
   ORX T0CTL1, #ONE_SHOT_MODE          ; Set timer to ONE-SHOT mode
   ORX T0CTL1, #PRESCALE               ; Set prescalar
   LDX T0H, #STARTCOUNT_HIGH           ; Load high byte register
   LDX T0L, #STARTCOUNT_LOW            ; Load low byte register
   LDX T0RH, #RELOAD_HIGH              ; Load reload high byte register
   LDX T0RL, #RELOAD_LOW               ; Load reload low byte register

   IFDEF TIMER0_ISR_ENABLE
   TIMER_PRIORITY_HIGH                 ; Set timer interrupt level
   ENDIF ;TIMER0_ISR_ENABLE

   IFDEF TIMER0_OUTPUTPIN_ENABLE
   ENABLE_TIMER0_OUTPUTPIN             ; Enable timer output pin
   ENDIF ;TIMER0_OUTPUTPIN_ENABLE

   ORX T0CTL1, #ENABLE_TIMER           ; Enable timer
   EI

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TMR0_INIT_XP_Continuous:
   DI
   ANDX T0CTL1,#DISABLE_TIMER          ; Disable timer
   ORX T0CTL1, #CONTINUOUS_MODE        ; Set timer to CONTINUOUS mode
   ORX T0CTL1, #PRESCALE               ; Set prescalar
   LDX T0H, #STARTCOUNT_HIGH           ; Load high byte register
   LDX T0L, #STARTCOUNT_LOW            ; Load low byte register
   LDX T0RH, #RELOAD_HIGH              ; Load reload high byte register
   LDX T0RL, #RELOAD_LOW               ; Load reload low byte register

   IFDEF TIMER0_ISR_ENABLE
   TIMER_PRIORITY_HIGH                 ; Set timer interrupt level
   ENDIF ;TIMER0_ISR_ENABLE

   IFDEF TIMER0_OUTPUTPIN_ENABLE
   ENABLE_TIMER0_OUTPUTPIN             ; Enable timer output pin
   ENDIF ;TIMER0_OUTPUTPIN_ENABLE

   ORX T0CTL1, #ENABLE_TIMER           ; Enable timer
   EI

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TMR0_INIT_XP_Counter:
   DI
   ANDX T0CTL1,#DISABLE_TIMER          ; Disable timer
   ORX T0CTL1, #COUNTER_MODE           ; Set timer to COUNTER mode
   LDX T0H, #STARTCOUNT_HIGH           ; Load high byte register
   LDX T0L, #STARTCOUNT_LOW            ; Load low byte register
   LDX T0RH, #COUNTER_HIGH             ; Load counter high byte register
   LDX T0RL, #COUNTER_LOW              ; Load counter low byte register

   IFDEF TIMER0_ISR_ENABLE
   IFNDEF _Z8ENCORE_XP_64XX_F0822
   LDX T0CTL0, #TICONFIG_RELOAD_COMPARE      ; Set timer to interrupt on
                                       ; reload/compare events only
   ENDIF ;_Z8ENCORE_XP_64XX_F0822

   TIMER_PRIORITY_HIGH                 ; Set timer interrupt level
   ENDIF ;TIMER0_ISR_ENABLE

   IFDEF TIMER0_OUTPUTPIN_ENABLE
   ENABLE_TIMER0_OUTPUTPIN             ; Enable timer output pin
   ENDIF

   ENABLE_TIMER0_INPUTPIN              ; Enable timer input pin

   ORX T0CTL1, #ENABLE_TIMER           ; Enable timer
   EI

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TMR0_INIT_XP_ComparatorCounter:
   IFNDEF _Z8ENCORE_XP_64XX_F0822
   DI

   ; Comparator0 must be initialized prior to timer initialization.
   ANDX T0CTL1,#DISABLE_TIMER          ; Disable timer
   ORX T0CTL1,#COMPARATOR_COUNTER_MODE ; Set timer to COMPARATOR COUNTER
   LDX T0CTL0, #TMODE_HIGH
   LDX T0H, #STARTCOUNT_HIGH           ; Load high byte register
   LDX T0L, #STARTCOUNT_LOW            ; Load low byte register
   LDX T0RH, #COUNTER_HIGH             ; Load counter high byte register
   LDX T0RL, #COUNTER_LOW              ; Load counter low byte register

   IFDEF TIMER0_ISR_ENABLE
   IFNDEF _Z8ENCORE_XP_64XX_F0822
   ORX T0CTL0, #TICONFIG_RELOAD_COMPARE; Set timer to interrupt on
                                       ; reload/compare events only
   ENDIF ;_Z8ENCORE_XP_64XX_F0822

   TIMER_PRIORITY_HIGH                 ; Set timer interrupt level
   ENDIF ;TIMER0_ISR_ENABLE

   IFDEF TIMER0_OUTPUTPIN_ENABLE
   ENABLE_TIMER0_OUTPUTPIN             ; Enable timer output pin
   ENDIF ;TIMER0_OUTPUTPIN_ENABLE

   ORX T0CTL1, #ENABLE_TIMER           ; Enable timer
   EI
   ENDIF ;_Z8ENCORE_XP_64XX_F0822

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TMR0_INIT_XP_PWM:
   DI
   ANDX T0CTL1,#DISABLE_TIMER          ; Disable timer
   ORX T0CTL1, #PWM_MODE               ; Set timer to PWM mode
   ORX T0CTL1, #PRESCALE               ; Set prescalar
   LDX T0H, #STARTCOUNT_HIGH           ; Load high byte register
   LDX T0L, #STARTCOUNT_LOW            ; Load low byte register
   LDX T0RH, #RELOAD_HIGH              ; Load reload high byte register
   LDX T0RL, #RELOAD_LOW               ; Load reload low byte register

   IFDEF _Z8ENCORE_F1680
   LDX T0PWM0H,#PWM_HIGH               ; Set PWM high byte register
   LDX T0PWM0L,#PWM_LOW                ; Set PWM low byte register
   ELSE ;_Z8ENCORE_F1680
   LDX T0PWMH, #PWM_HIGH
   LDX T0PWML, #PWM_LOW
   ENDIF ;_Z8ENCORE_F1680

   IFDEF TIMER0_ISR_ENABLE
   IFNDEF _Z8ENCORE_XP_64XX_F0822
   LDX T0CTL0, #TICONFIG_RELOAD_COMPARE; Set timer to interrupt on
                                       ; reload/compare events only
   ENDIF ;_Z8ENCORE_XP_64XX_F0822

   TIMER_PRIORITY_HIGH                 ; Set timer interrupt level
   ENDIF ;TIMER0_ISR_ENABLE

   ENABLE_TIMER0_OUTPUTPIN             ; Enable timer output pin

   ORX T0CTL1, #ENABLE_TIMER           ; Enable timer
   EI

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TMR0_INIT_XP_PWM_DualOutput:
   IFNDEF _Z8ENCORE_XP_64XX_F0822
   DI
   ANDX T0CTL1,#DISABLE_TIMER          ; Disable timer
   ORX T0CTL1, #PWM_DUAL_OUTPUT_MODE   ; Set timer to COMPARATOR COUNTER
   ORX T0CTL0, #TMODE_HIGH
   ORX T0CTL1, #PRESCALE               ; Set prescalar
   LDX T0H, #STARTCOUNT_HIGH           ; Load high byte register
   LDX T0L, #STARTCOUNT_LOW            ; Load low byte register
   LDX T0RH, #RELOAD_HIGH              ; Load reload high byte register
   LDX T0RL, #RELOAD_LOW               ; Load reload low byte register

   IFDEF _Z8ENCORE_F1680
   LDX T0PWM0H,#PWM_HIGH               ; Set PWM high byte register
   LDX T0PWM0L,#PWM_LOW                ; Set PWM low byte register
   ELSE ;_Z8ENCORE_F1680
   LDX T0PWMH, #PWM_HIGH
   LDX T0PWML, #PWM_LOW
   ENDIF ;_Z8ENCORE_F1680

   IFDEF TIMER0_ISR_ENABLE
   TIMER_PRIORITY_HIGH                 ; Set timer interrupt level
   ENDIF ;TIMER0_ISR_ENABLE

   IFDEF TIMER0_OUTPUTPIN_ENABLE
   ENABLE_TIMER0_OUTPUTPIN             ; Enable timer output pin
   ENDIF ;TIMER0_OUTPUTPIN_ENABLE

   ENABLE_TIMER0_OUTPUTPIN             ; Enable timer output pin
   ENABLE_TIMER0_INPUTPIN              ;Enable timer output complement pin

   ORX T0CTL1, #ENABLE_TIMER           ; Enable timer
   EI
   ENDIF ;_Z8ENCORE_XP_64XX_F0822

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TMR0_INIT_XP_Capture:
   DI
   ANDX T0CTL1,#DISABLE_TIMER          ; Disable timer
   ORX T0CTL1, #CAPTURE_MODE           ; Set timer to CAPTURE mode
   ORX T0CTL1, #PRESCALE               ; Set prescalar
   LDX T0H, #STARTCOUNT_HIGH           ; Load high byte register
   LDX T0L, #STARTCOUNT_LOW            ; Load low byte register
   LDX T0RH, #RELOAD_HIGH              ; Load reload high byte register
   LDX T0RL, #RELOAD_LOW               ; Load reload low byte register

   IFDEF _Z8ENCORE_F1680
   LDX T0PWM0H,#%00                    ; Clear PWM high byte register
   LDX T0PWM0L,#%00                    ; Clear PWM low byte register
   ELSE ;_Z8ENCORE_F1680
   LDX T0PWMH, #%00
   LDX T0PWML, #%00
   ENDIF ;_Z8ENCORE_F1680

   IFDEF TIMER0_ISR_ENABLE
   IFNDEF _Z8ENCORE_XP_64XX_F0822
   LDX T0CTL0, #TICONFIG_CAPTURE_DEASSERT    ; Set timer to interrupt on
                                       ; capture events only
   ENDIF ;_Z8ENCORE_XP_64XX_F0822

   TIMER_PRIORITY_HIGH                 ; Set timer interrupt level
   ENDIF ;TIMER0_ISR_ENABLE

   ENABLE_TIMER0_INPUTPIN              ; Enable timer input pin

   ORX T0CTL1, #ENABLE_TIMER           ; Enable timer
   EI

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TMR0_INIT_XP_CaptureRestart:
   ; Not supported in F0822 and F64xx devices!
   IFNDEF _Z8ENCORE_XP_64XX_F0822
   DI
   ANDX T0CTL1,#DISABLE_TIMER          ; Disable timer
   ORX T0CTL1,#CAPTURE_RESTART_MODE    ; Set timer to CAPTURE RESTART mode
   LDX T0CTL0,#TMODE_HIGH
   ORX T0CTL1,#PRESCALE                ; Set prescalar
   LDX T0H, #STARTCOUNT_HIGH           ; Load high byte register
   LDX T0L, #STARTCOUNT_LOW            ; Load low byte register
   LDX T0RH, #%FF                      ; Load reload high byte register
   LDX T0RL,#%FF                       ; Load reload low byte register

   IFDEF _Z8ENCORE_F1680
   LDX T0PWM0H,#%00                    ; Clear PWM high byte register
   LDX T0PWM0L,#%00                    ; Clear PWM low byte register
   ELSE ;_Z8ENCORE_F1680
   LDX T0PWMH,#%00
   LDX T0PWML,#%00
   ENDIF ;_Z8ENCORE_F1680

   IFDEF TIMER0_ISR_ENABLE
   IFNDEF _Z8ENCORE_XP_64XX_F0822
   ORX T0CTL0, #TICONFIG_CAPTURE_DEASSERT    ; Set timer to interrupt
                                       ; on capture events only
   ENDIF ;_Z8ENCORE_XP_64XX_F0822

   TIMER_PRIORITY_HIGH                 ; Set timer interrupt
   ENDIF ;TIMER0_ISR_ENABLE

   ENABLE_TIMER0_INPUTPIN              ; Enable timer input pin

   ORX T0CTL1,#ENABLE_TIMER            ; Enable timer
   EI
   ENDIF ;_Z8ENCORE_XP_64XX_F0822

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TMR0_INIT_XP_Compare:
   DI
   ANDX T0CTL1,#DISABLE_TIMER          ; Disable timer
   ORX T0CTL1, #COMPARE_MODE           ; Set timer to COMPARE mode
   ORX T0CTL1,#PRESCALE                ; Set prescalar
   LDX T0H,#STARTCOUNT_HIGH            ; Load high byte register
   LDX T0L,#STARTCOUNT_LOW             ; Load low byte register
   LDX T0RH,#RELOAD_HIGH               ; Load compare high byte register
   LDX T0RL,#RELOAD_LOW                ; Load compare low byte register

   IFDEF TIMER0_ISR_ENABLE
   IFNDEF _Z8ENCORE_XP_64XX_F0822
   LDX T0CTL0, #TICONFIG_RELOAD_COMPARE      ; Set timer to interrupt on
                                       ; reload/compare events only
   ENDIF ;_Z8ENCORE_XP_64XX_F0822

   TIMER_PRIORITY_HIGH                 ; Set timer interrupt level
   ENDIF ;TIMER0_ISR_ENABLE

   IFDEF TIMER0_OUTPUTPIN_ENABLE
   ENABLE_TIMER0_OUTPUTPIN             ; Enable timer output pin
   ENDIF ;TIMER0_OUTPUTPIN_ENABLE

   ORX T0CTL1, #ENABLE_TIMER            ; Enable timer
   EI

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TMR0_INIT_XP_Gated:
   DI
   ANDX T0CTL1,#DISABLE_TIMER          ; Disable timer
   ORX T0CTL1, #GATED_MODE             ; Set timer to GATED mode
   ORX T0CTL1, #PRESCALE               ; Set prescalar
   LDX T0H,#STARTCOUNT_HIGH            ; Load high byte register
   LDX T0L,#STARTCOUNT_LOW             ; Load low byte register
   LDX T0RH,#RELOAD_HIGH               ; Load reload high byte register
   LDX T0RL,#RELOAD_LOW                ; Load reload low byte register

   IFDEF TIMER0_ISR_ENABLE
   TIMER_PRIORITY_HIGH                 ; Set timer interrupt
   ENDIF ;TIMER0_ISR_ENABLE

   IFDEF TIMER0_OUTPUTPIN_ENABLE
   ENABLE_TIMER0_OUTPUTPIN             ; Enable timer output pin
   ENDIF ;TIMER0_OUTPUTPIN_ENABLE

   ENABLE_TIMER0_INPUTPIN              ; Enable timer input pin

   ORX T0CTL1, #ENABLE_TIMER           ; Enable timer
   EI

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TMR0_INIT_XP_CaptureCompare:
   DI
   ANDX T0CTL1,#DISABLE_TIMER          ; Disable timer
   ORX T0CTL1, #CAPTURE_COMPARE_MODE   ; Set timer to CAPTURE COMPARE mode
   ORX T0CTL1, #PRESCALE               ; Set prescalar
   LDX T0H,#STARTCOUNT_HIGH            ; Load high byte register
   LDX T0L,#STARTCOUNT_LOW             ; Load low byte register
   LDX T0RH,#RELOAD_HIGH               ; Load compare high byte register
   LDX T0RL,#RELOAD_LOW                ; Load compare low byte register

   IFDEF TIMER0_ISR_ENABLE
   IFNDEF _Z8ENCORE_XP_64XX_F0822
   LDX T0CTL0, #TICONFIG_CAPTURE_DEASSERT    ; Set timer to interrupt
                                       ; on capture events only
   ENDIF ;_Z8ENCORE_XP_64XX_F0822

   TIMER_PRIORITY_HIGH                 ; Set timer interrupt level
   ENDIF ;TIMER0_ISR_ENABLE

   ENABLE_TIMER0_INPUTPIN              ; Enable timer input pin
   ORX T0CTL1, #ENABLE_TIMER           ; Enable timer
   EI

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TMR0_INIT_XP_TriggeredOneShot:
   IFDEF _Z8ENCORE_F1680
   ANDX T0CTL1,#DISABLE_TIMER          ; Disable timer
   ORX T0CTL1,#TRIGGERED_ONESHOT_MODE  ; Set timer to TRIGGERED ONE-SHOT
   LDX T0CTL0,#TMODE_HIGH
   ORX T0CTL1,#PRESCALE                ; Set prescalar
   LDX T0H,#STARTCOUNT_HIGH            ; Load high byte register
   LDX T0L,#STARTCOUNT_LOW             ; Load low byte register
   LDX T0RH,#RELOAD_HIGH               ; Load reload high byte register
   LDX T0RL,#RELOAD_LOW                ; Load reload low byte register

   IFDEF TIMER0_ISR_ENABLE
   ORX T0CTL0, #TICONFIG_RELOAD_COMPARE      ; Set timer to interrupt
                                       ; on reload events only
   TIMER_PRIORITY_HIGH                 ; Set timer interrupt level
   ENDIF ;TIMER0_ISR_ENABLE

   ENABLE_TIMER0_OUTPUTPIN             ; Enable timer output pin
   ENABLE_TIMER0_INPUTPIN              ; Enable timer input pin

   ORX T0CTL1,#ENABLE_TIMER            ; Enable timer
   EI
   ENDIF ;_Z8ENCORE_F1680

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TMR0_INIT_XP_Demodulation:
   IFDEF _Z8ENCORE_F1680
   DI
   ANDX T0CTL1,#DISABLE_TIMER          ; Disable timer
   ORX T0CTL1,#DEMODULATION_MODE       ; Set timer to DEMODULATION mode
   LDX T0CTL0,#TMODE_HIGH
   ORX T0CTL1,#PRESCALE                ; Set prescalar
   LDX T0H,#STARTCOUNT_HIGH            ; Load high byte register
   LDX T0L,#STARTCOUNT_LOW             ; Load low byte register
   LDX T0RH,#RELOAD_HIGH               ; Load reload high byte register
   LDX T0RL,#RELOAD_LOW                ; Load reload low byte register
   LDX T0PWM0H,#%00                    ; Clear timer PWM0 & PWM1 registers
   LDX T0PWM0L,#%00
   LDX T0PWM1H,#%00
   LDX T0PWM1L,#%00

   IFDEF TIMER0_ISR_ENABLE
   ORX T0CTL2, #TPOLHI                 ; Capture on both edges
   ORX T0CTL0, #TICONFIG_CAPTURE_DEASSERT    ; Set timer to interrupt
                                       ; on capture events only
   TIMER_PRIORITY_HIGH                 ; Set timer interrupt level
   ENDIF ;TIMER0_ISR_ENABLE

   ENABLE_TIMER0_INPUTPIN              ; Enable timer input pin

   ORX T0CTL1,#ENABLE_TIMER            ; Enable timer
   EI
   ENDIF ;_Z8ENCORE_F1680
```