Freescale Semiconductor, Inc.
Application Notes

Document Number: AN5142
Rev. 0, 06/2015

Features of the FlexTimer Module

1. Introduction

The application note introduces multiple features of the FlexTimer Module (FTM) and provides corresponding code for each feature, along with a waveform or a snapshot of the oscilloscope. While FTM is used in both the Kinetis and Vybrid families, this application note focuses on Kinetis only.

FTM is available for all of the Kinetis K series. FTM is an enhanced version of the Timer/PWM module (TPM) of HCS08 with new features.

FTM can:

- function as the Timer and Pulse Width Modulation (PWM) signal generator,
- generate at most eight PWM signals for each FTM module, and
- generate edge-aligned PWM signals, center-aligned PWM signals, and phase-shift PWM signals.

External/internal fault signals can disable PWM output signal. The PWM module can trigger ADC directly or through the PDB module. FTM can generate four pairs of complementary PWM signal for edge-aligned, center-aligned, and phase shift PWM signals.

FTM also has masking, inverting, and software controlling function for BLDC motor control application. The features described in this application note are applicable for BLDC/ACIM/PMSM/SR motor control or switch mode power supply applications.

Contents

When FTM functions as a timer it can generate fixed cycle time interrupt, generate divided signal, and can count the Encoder signal with 90 degree shift. It has a capture function, which is used to detect and get Hall signal logic in motor control application. The capture function also helps in measuring the duty cycle or the period of external signal.

The code is developed in CodeWarrior for Microcontroller ver10.6 and TWR-K40x256 Tower board.

2. FTM description

Typically, one Kinetis K family processor has two or three FTM modules. FTM with eight channels can generate a PWM signal to control a motor. One of the FTMs can receive Encoder signals, can receive Hall signals, and the third can generate fixed cycle time interrupt.

The FTM0 in general has four pairs or eight channels CH0/CH1, CH2/CH3, CH4/CH5, and CH6/CH7. All the channel pins can output PWM signals and can also be input pins to get capture signals.

The FTM1 has dedicated Phase A and Phase B signal input pins FTM1_QD_PHA/FTM1_QD_PHB, which can accept encoder sensor output signals. The Phase A and Phase B signals have 90 degree phase shift. For example, the Kinetis K40 family has three FTM modules: FTM0, FTM1, and FTM2. FTM1 is a full function module, which has capture function to detect and get Hall signal, generate capture interrupt, and can output PWM signals.

Each FTM supports fault function. The fault signals can be from internal module output for example comparator or external pins. The fault signal can disable PWM signals automatically so that the logic of PWM signal pins can be high or low, which can consequently disable power device in a control system. When the fault signal disappears, the PWM signal can recover automatically or manually.

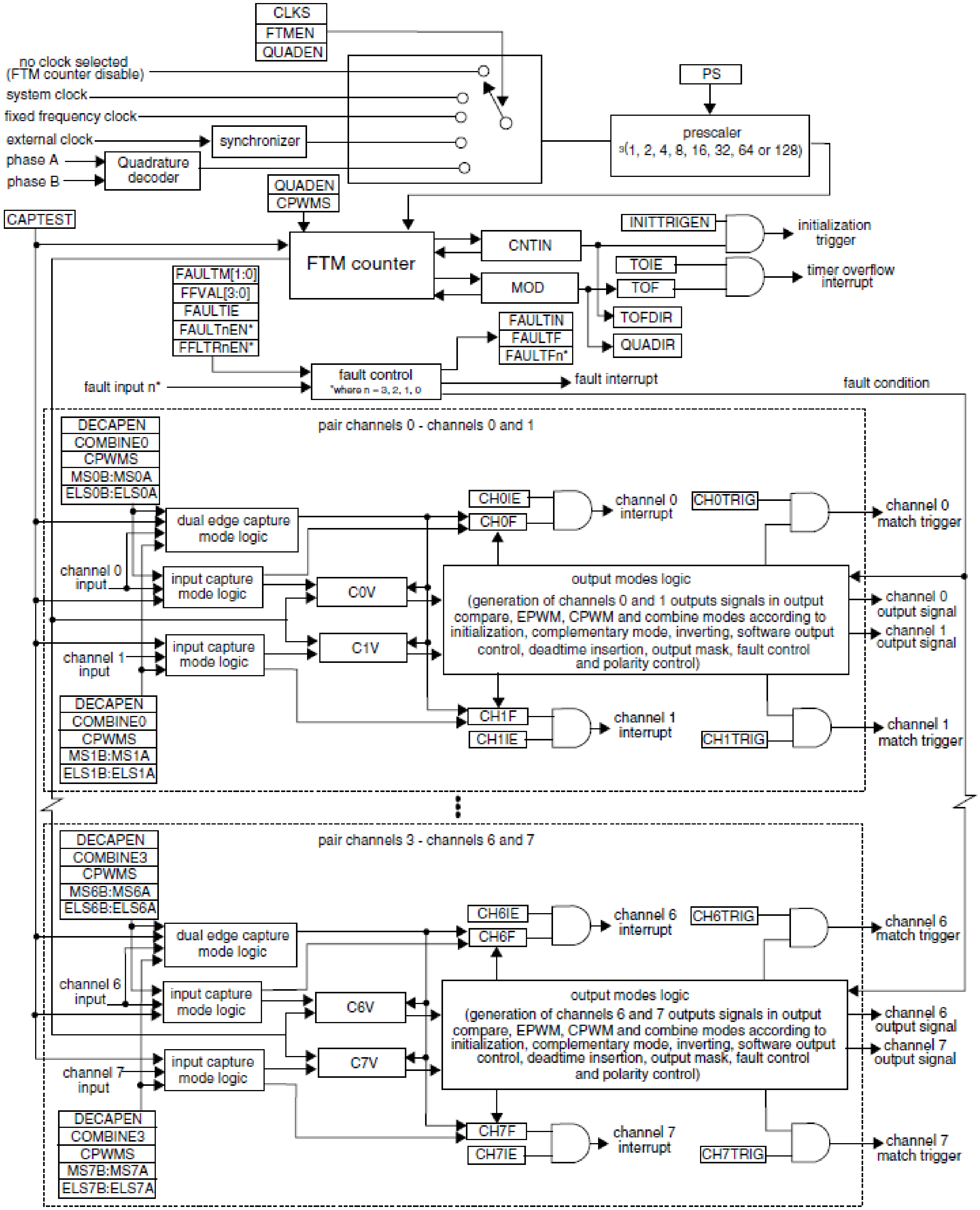

Figure 1. FTM block diagram

Figure 1 is the FTM block diagram. There are several clock sources which can drive the FTM module. Prescaler is the internal clock divider. The FTM counter is reloaded from the FTM_CNTIN register. The FTM counter counts the clock source from FTM_CNTIN to the FTM_MOD register. During the process, the FTM channel pin can change its logic when the FTM counter reaches to FTM_CnV, which is a fundamental mechanism to generate PWM signal.

FTM can be either a timer or PWM signal generator. The FTM_CnSC register determines the modes of the FTM.

Table 1. FTM mode setting and capture edge level selection

DECAPEN	COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
0	0	0	0	1	Input capture	Capture on Rising Edge Only
				10		Capture on Falling Edge Only
				11		Capture on Rising or Falling Edge
			1	1	Output compare	Toggle Output on match
				10		Clear Output on match
				11		Set Output on match
			1X	10	Edge-aligned PWM	High-true pulses (clear Output on match)
				X1		Low-true pulses (set Output on match)
		1	XX	10	Center-aligned PWM	High-true pulses (clear Output on match-up)
				X1		Low-true pulses (set Output on match-up)
	1	0	XX	10	Combine PWM	High-true pulses (set on channel (n) match, and clear on channel (n+1) match)
				X1		Low-true pulses (clear on channel (n) match, and set on channel (n+1) match)
1	0	0	X0		Dual Edge Capture Mode	One-shot capture mode
			X1			Continuous capture mode

3. PWM Features

3.1. Edge-aligned PWM mode

In edge-aligned PWM mode, the FTM counter counts up from the FTM_CNTIN value to the FTM_MOD value. All FTM channels signals align at the edge when the FTM counter changes from the MOD value to the CNTIN value.

The Edge-aligned mode is selected when:

(QUADEN = 0), (DECAPEN = 0), (COMBINE = 0), (CPWMS = 0), and (MSnB = 1)

The edge-aligned PWM period is determined by (MOD − CNTIN + 0x0001) and the pulse width or the duty cycle is determined by (CnV − CNTIN) or (MOD-CNTIN-CnV) dependent on ELSnB:ELSnA bits setting.

Example 1. PWM source code in the edge-aligned mode

```
void PWMOutput_EdgeAlignment(void)
{
SIM_SCGC6|=0x03000000; //enable FTM0 and FTM0 module clock
FTM0_CONF=0xC0; //set up BDM in 11
FTM0_FMS=0x00; //clear the WPEN so that WPDIS is set in FTM0_MODE register
FTM0_MODE|=0x05; //enable write the FTM CnV register
FTM0_MOD=1000;
FTM0_C0SC=0x28; //edge-alignment, PWM initial state is High, becomes low         //after
match
FTM0_C1SC=0x28;
FTM0_COMBINE=0x02; //complementary mode for CH0&CH1 of FTM0
FTM0_COMBINE|=0x10; //dead timer insertion enabled in complementary mode for //CH0&CH1 of
FTM0
FTM0_C1V=500;
FTM0_C0V=500;

FTM0_C2SC=0x28;
FTM0_C3SC=0x28;
FTM0_COMBINE|=0x0200;
FTM0_COMBINE|=0x1000;
FTM0_DEADTIME=0x00;
FTM0_C3V=250;
FTM0_C2V=250;
FTM0_CNTIN=0x00;
FTM0_SC=0x08; //PWM edge_alignment, system clock driving, dividing by 1
}
```

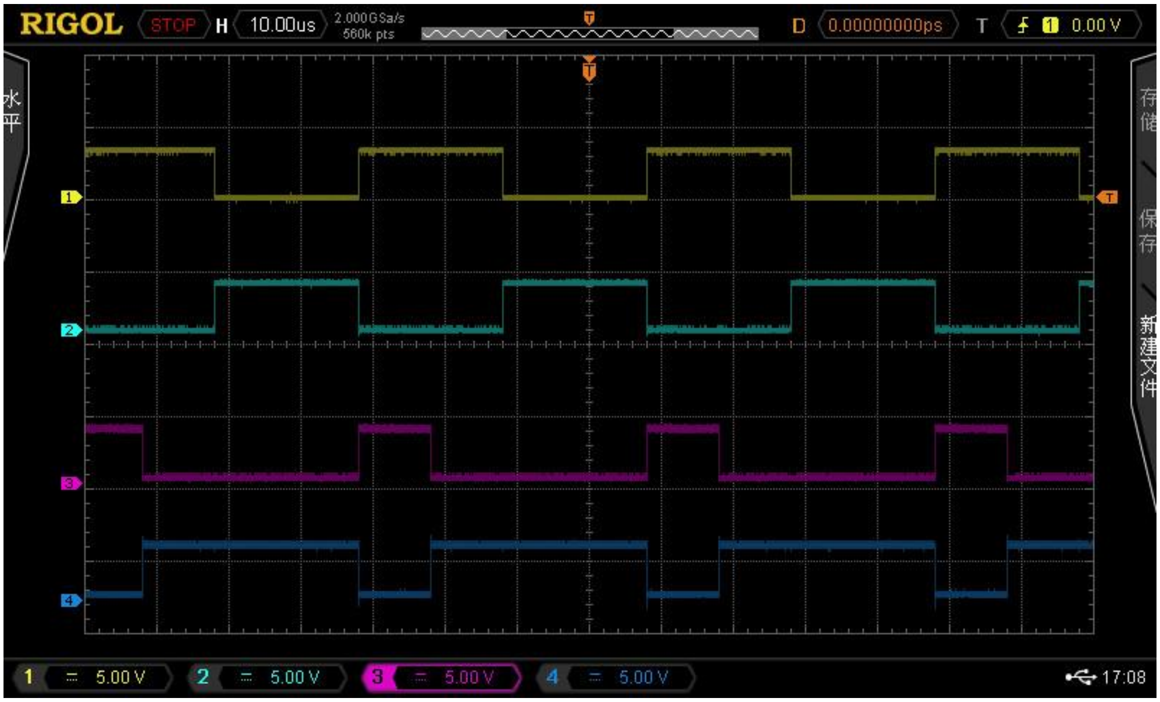

Figure 2. Waveform of the PWM signal with the edge-aligned mode

In Figure 2, the CH1 and CH2 channel signals on the oscilloscope are the FTM0_CH0 and FTM0_CH1 signals. The CH3 and CH4 channels signals on oscilloscope are the FTM0_CH2 and FTM0_CH3 signals. The waveform shows that the FTM0_CH0/FTM_CH1 are complementary signals, the

FTM0_CH2 and FTM0_CH3 are complementary signals, and the rising edge of FTM_CH0 and FTM0_CH2 signals is aligned. The FTM0 works in edge-alignment mode.

NOTE

In Figure 2, CH1 is Yellow channel, CH2 is Cyan channel, CH3 is pink channel, and CH4 is blue channel.

3.2. Center-aligned PWM mode

In center-aligned PWM mode, the FTM counter counts up from FTM_CNTIN to FTM_MOD and then counts down from FTM_MOD to FTM_CTNIN. All FTM channels signals align at the point when the FTM counter reaches up to FTM_MOD value.

The center-aligned mode is selected when (QUADEN = 0), (DECAPEN = 0), (COMBINE= 0), and (CPWMS = 1).

Example 2. The PWM source code in the center-aligned mode

```
void PWMOutput_CenterAlignment(void)
{
SIM_SCGC6|=0x03000000; //enable FTM0 and FTM0 module clock
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
FTM0_CONF=0xC0; //set up BDM in 11
FTM0_FMS=0x00; //clear the WPEN so that WPDIS is set in FTM0_MODE reg
FTM0_MODE|=0x05; //enable write the FTM CnV register
FTM0_MOD=1000;
FTM0_C0SC=0x28; ////center-alignment, PWM begins with High
FTM0_C1SC=0x28; //PWM waveform is high-low-high
FTM0_COMBINE=0x02; //complementary mode for CH0&CH1 of FTM0
FTM0_COMBINE|=0x10; // dead timer insertion enabled in complementary mode for //CH0&CH1 of
FTM0
FTM0_DEADTIME=0x1F; //dead time is 16 system clock cycles
FTM0_C1V=500;
FTM0_C0V=500;
FTM0_CNTIN=0x00;

FTM0_C2SC=0x28;
FTM0_C3SC=0x28;
FTM0_COMBINE|=0x0200;
FTM0_COMBINE|=0x1000;
FTM0_C3V=250;
FTM0_C2V=250;
FTM0_SC=0x68;
}
```

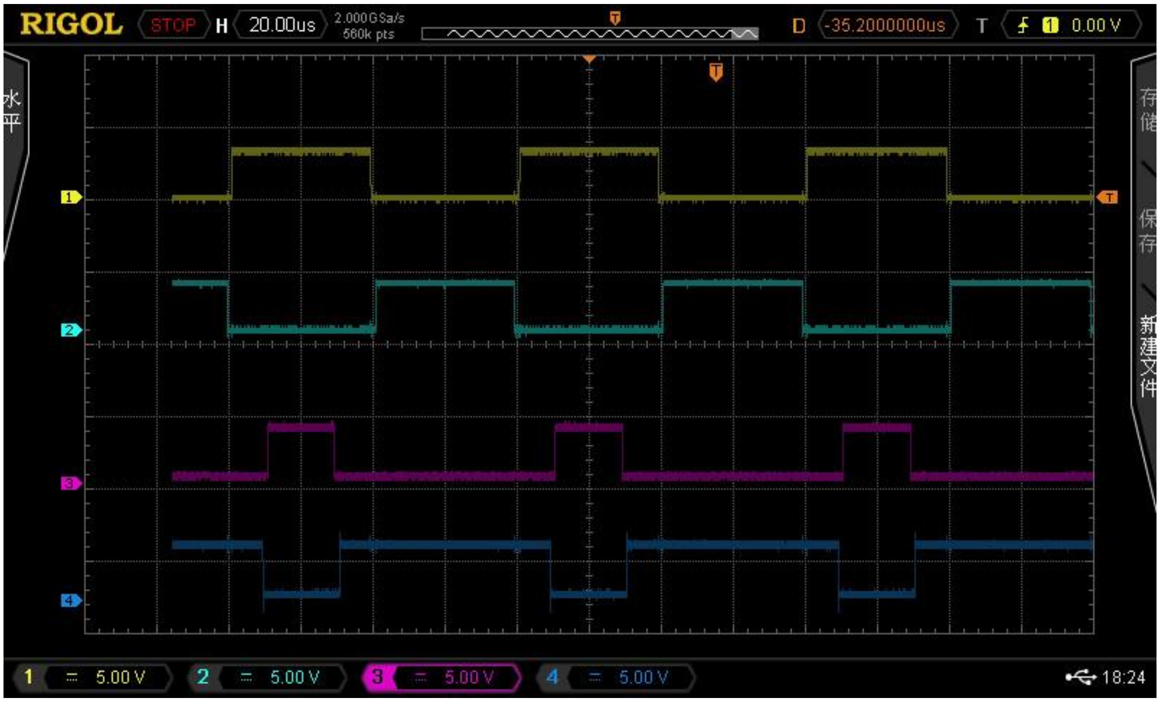

Figure 3. Waveform of the PWM signals with the center-alignment mode

In Figure 3, CH1 and CH2 signals on the oscilloscope are the FTM0_CH0 and FTM0_CH1 signals. The CH3 and CH4 signals on the oscilloscope are the FTM0_CH2 and FTM0_CH3. The waveform shows that the FTM0_CH0/FTM_CH1 are complementary signals. The FTM0_CH2 and FTM0_CH3 are complementary signals. The center of FTM_CH0 and FTM0_CH2 signals are aligned, specifically, the center of Low logic of both FTM_CH0 and FTM0_CH2 signals are aligned due to the fact that FTM signal waveform mode is High-Low-High. The FTM0 works in the center-alignment mode, while a dead-time is inserted, which means that there is a low time between low edge of FTM0_CH0 and rising edge of FTM_CH1.

3.3. Complementary PWM signal

The complementary PWM means that two FTM channels comprise a pair. The FTM_CHn+1 channel signal and FTM_CHn channel signal are not independent. The FTM_CHn+1 channel signal is inverter of FTM_CHn channel signal if the dead time is not inserted. Both the edge-aligned and center-aligned PWM modes support complementary PWM signals.

The complementary PWM signals are widely used in motor control and switch mode power supply application.

If the COMPx=1 in FTM_COMBINE register, the corresponding channels works in complementary mode irrespective of whether the FTM works in the edge-alignment mode or the center-alignment mode. If the DTEx=1, a dead time is inserted for the pair of PWM signals.

See Figure 2 and Figure 3 for the complementary PWM signals.

3.4. Phase-shift PWM signal (Combined PWM signals or Asymmetric PWM signal)

The Phase-shift PWM signal is also referred to as the combined PWM signals or the asymmetric PWM signal. The PWM channel signals are aligned in either edge-aligned or center-aligned but the FTM can also generate phase shift PWM signals. Consider, there are two complementary pairs PWM: PWM0/PWM1 and PWM2/PWM3. The PWM0/PWM1 are complementary PWM signal pair, PWM2/PWM3 are another complementary PWM signal pair. The PWM0 and PWM2 can have a programmable phase shift with the same duty cycle. The phase shift PWM signals are widely used in phase-shift full bridge DC/DC converter and the current reconstruction of sinusoidal-type motor FOC control with one serial shunt resistor.

The combined mode is selected when (FTMEN = 1), (QUADEN = 0), (DECAPEN = 0), (COMBINE = 1), (CPWMS = 0), and (COMP=1).

The combined PWM mode supports only the edge-aligned (CPWMS = 0) and complementary modes (COMP=1) but does not support center-aligned mode. In complementary mode, the FTM_CHn+1 channel PWM signal is inverter of FTM_CHn channel PWM signal, so the FTM_Cn+1V register is not used, when (COMBINE = 1), and (CPWMS = 0) and (COMP=1), the FTM counter counts from CNTIN to MOD value. Consider that the COMBINE bit is set, and the FTM_CnV register value is less than that of FTM_Cn+1V register, when the FTM counter reaches up to FTM_CnV value, the output logic of FTM_CHn pin changes from 0 to 1, or from 1 to 0. When the FTM counter reaches up to FTM_Cn+1V value, the output logic of FTM_CHn changes again. In this way, the output logic of FTM_CHn pin is changed twice in one PWM cycle, so you can generate a complicated PWM signal. We can adjust the FTM_CnV/FTM_Cn+1V registers value of FTM_CHn and the FTM_Cn+2V/FTM_Cn+3V register value of FTM_CHn+2 pair so that the FTM_CHn and FTM_CHn+2 PWM signal can have a phase shift but with the same duty cycle.

The following code can generate phase shift PWM signals.

Example 3. Code to generate phase shift PWM signals

```
void PWMPhaseShift(void)
{
SIM_SCGC6|=0x03000000; //enable FTM0 and FTM0 module clock
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
FTM0_SC=0x00;
FTM0_CONF=0xC0; //set up BDM in 11
FTM0_FMS=0x00; //clear the WPEN so that WPDIS is set in FTM0_MODE reg
FTM0_MODE|=0x05; //enable write the FTM CnV register
FTM0_MOD=1000;
FTM0_C0SC=0x28; //High-Low_high for combined and complementary mode
FTM0_C1SC=0x28;
FTM0_C2SC=0x28;
FTM0_C3SC=0x28;
FTM0_COMBINE=0x0303; //complementary and combined mode for CH0&CH1, CH2&CH3
FTM0_COMBINE|=0x1010; // dead timer insertion enabled in complementary mode //for CH0&CH1 of
FTM0
FTM0_DEADTIME=0x1F; //deattime is 31 system clock cycles
FTM0_C1V=750;
FTM0_C0V=250;
FTM0_CNTIN=0x00;
FTM0_C2V=500;
FTM0_C3V=1000;
FTM0_SC=0x08; //PWM edge_alignment, system clock driving, dividing by 1
}
```

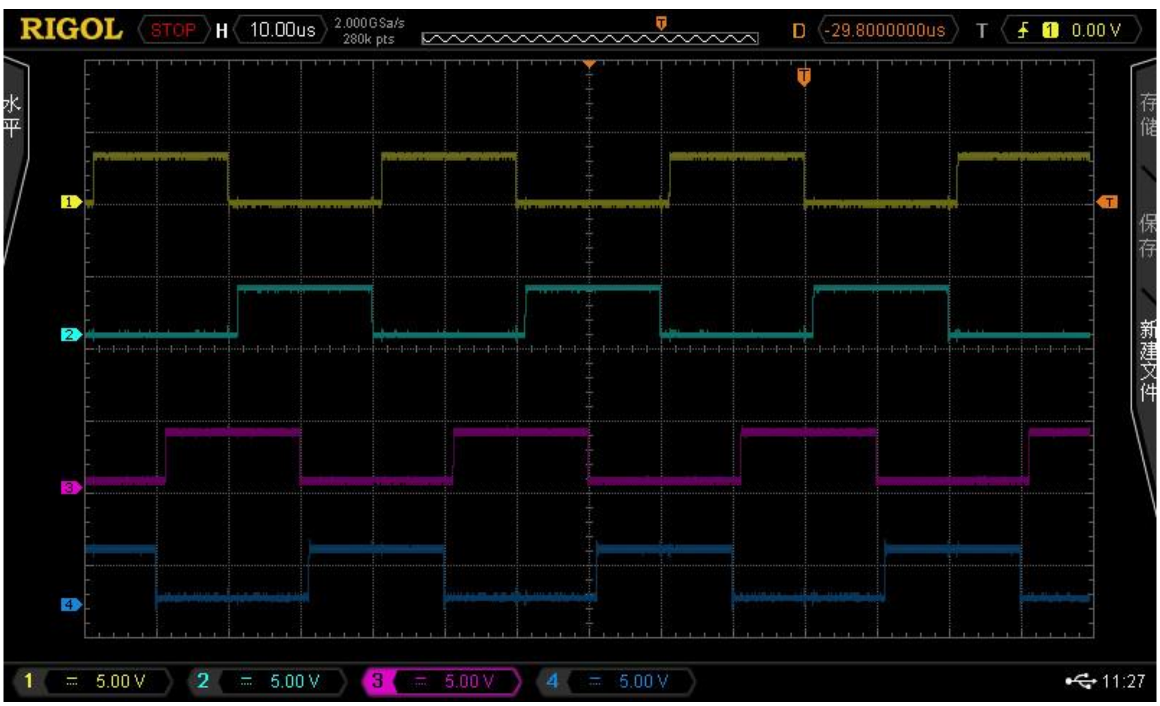

Figure 4. Phase-shift PWM signal in the combined and complementary modes

In Figure 4, the CH1 and CH2 channels on oscilloscope are FTM0_CH0 and FTM0_CH1 signals. The CH3 and CH4 channels on oscilloscope are FTM0_CH2 and FTM0_CH3 signals. The waveform shows that the FTM0_CH0/FTM_CH1 are complementary signals. The FTM0_CH2 and FTM0_CH3 are complementary signals with dead time inserted. The combined mode cannot work with the center-alignment mode. From the waveform, FTM0_CH0 and FTM0_CH2 have 90 degree phase shift.

3.5. Divider

FTM can be a clock divider. When FTM uses an external clock signal or an internal clock as a clock source, the FTM can divide the source clock and output the divider of the source clock signal.

Configuration: DECAPEN=0, Combine=0, CPWMS=0, MSnB:MSnA=01, ELSnB:ELSnA=01.

The FTM_CNT starts from FTM_CNTIN value to FTM_MOD value, when the FTM_CNT reaches up to the FTM_CnV register value, the corresponding FTM_CHn pin output signal toggles, so the FTM channel FTM_CHn output signal cycle=2*(FTM_MOD-FTM_CNTIN)*(tick clock cycle).

There is only one FTM_CNT for each FTM therefore if you use multiple channels to output divided clock signal, all channel output signals should have the same cycle time, but should have different phase shift. The FTM_CnV register can control the phase shift.

Example 4. Source code of the divider feature

```
void FTM_Divider(void)
{

SIM_SCGC6|=0x03000000; //enable FTM0 and FTM0 module clock
FTM0_CONF=0xC0; //set up BDM in 11
FTM0_MODE|=0x05; //enable write the FTM CnV register
FTM0_MOD=1000;
FTM0_C0SC=0x14; //FTM output signal toggle when the FTM counter matches with FTM0_C0V
register
FTM0_C1SC=0x14;
FTM0_C1V=500;
FTM0_C0V=500;
FTM0_CNTIN=0x00;
FTM0_SC=0x08; // system clock driving, dividing by 1
```

3.6. Generating a fixed cycle time interrupt

For the Kinetis K family, FlexTimer (FTM), Periodic Interrupt Timer (PIT), Programmable Delay Block (PDB), and Low Power Timer (LPTMR) modules can generate fixed cycle interrupt.

Each FTM channel can generate interrupt. For example, if one FTM channel FTM_CH0 is used to generate interrupt, you should set the CHIE bit in FTM_C0SC register. When the FTM counter reaches FTM_C0V, the CHF bit in FTM_C0SC register is set and an interrupt is generated. No matter which channel is used, the interrupt cycle time is FTM_MOD*(tick cycle time).

The timer overflow event of FTM can also generate interrupt, you can set the TOIE bit in FTM_SC to enable the FTM overflow interrupt. The interrupt cycle time is FTM_MOD*(tick cycle time).

Each FTM has only one interrupt vector which is shared by all of the FTM interrupt sources within the ISR. Check the flag to identify the interrupt source, then clear the corresponding flag and execute the corresponding code.

Example 5. Source code of generating fixed cycle time interrupt based on FTM

```
void FixedTimerInterrupt(void)
{
//toggle LED E1 by toggling PCT7 pin, set the PCT7 as GPIO output pin mode
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
```

```
PORTC_PCR7=0x100; //PTC7 in GPIO mode
GPIOC_PDDR=0x80; //PTC direction register, PTC7 is in output mode
SIM_SCGC6|=0x03000000; //enable FTM0 and FTM0 module clock
FTM0_CONF=0xC0; //set up BDM in 11
FTM0_MODE|=0x0B; //enable write the FTM CnV register
FTM0_CNTIN=0x00;
FTM0_MOD=1000;
FTM0_SC=0x48; //enable FTM0
asm("cpsie i");
}
void Ftm0_interruptInit(void)
{
NVICIP62=0x30; //set FTM0 interrupt priority
NVICICPR1|=1<<30;   //clear the pending register of interrupt source 69(PIT1)
NVICISER1|=1<<30;  //set the interrupt source of PIT1
}
void FTM0_ISR()
{
       if(FTM0_SC&0x80)
       {
       GPIOC_PTOR=0x80;
       FTM0_SC&=0x7F;
       asm("nop"); //set a break point
       }
```

Finally, you have to change the FTM0_ISR vector in the vector table.

For the FTM0 module, there is only one vector in the vector table, but there are multiple interrupt sources. You can check the FTM_STATUS register to identify the interrupt source and take appropriate action.

3.7. FTM triggering ADC

In motor control and switch mode power supply applications, it is necessary to measure the current at the center of the PWM signal with shunt resistor as a current sensor. At the center of the PWM signal, power devices do not switch. Consequently, the voltage on the shunt resistor is stable.

The FTM0 module has eight channels from CH0 to CH7. Typically, six channels are enough to control a motor, except for the switch reluctance motor. The remaining two channels control the instant to trigger ADC, referred as ADC synchronized with PWM.

The FTM_EXTTRIG register can determine which FTM channel can trigger ADC. The channels CH0/CH1/CH2/CH3/CH4/CH5, and CNTIN register all can trigger ADC. For example, if the CH0 bit is set in FTM_EXTTRIG register, when the FTM_CNT reaches up to FTM_C0V register value, a signal is generated to trigger ADC. If the INITTRIGEN bit is set in FTMx_EXTTRIG register and when the FTM_CNT reaches up to FTM_CNTIN register value, a signal is generated to trigger ADC.

For the Kinetis K40 family, the SIM_SOPT7 register controls the ADC triggering source.

ADC is an important module. ADC can accept differential analog signal and single-ended analog signal, in differential mode. The analog voltage range of ADC is from -3.3 V to +3.3 V. In single-ended mode, the analog voltage range is from GND to 3.3 V.

ADC has the software triggering mode, the hardware triggering mode, and the continuous mode. If the ADCO bit is set in ADC_SC3 register, the ADC is in continuous mode. Once ADC is triggered to start sampling by either the software or hardware triggering mode, the ADC converts immediately after the last conversion is over. For the software triggering, writing the ADC_SC1A will trigger ADC to convert immediately. If the AIEN bit in ADC_SC1A is set, an interrupt of ADC conversion is generated automatically and you can read the sample in ISR.

For hardware triggering, PDB, PIT, FTM, LPTMR, on-chip CMP all can trigger ADC, the hardware triggering source setting is chip by chip, for K40, see the SIM_SOPT7 register.

The following code demonstrates how to trigger ADC with FTM channel.

Example 6. Trigger ADC with FTM channel

```
void FTM_TrigAdc(void)
{
SIM_SCGC6|=0x03000000; //enable FTM0 and FTM0 module clock
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
PORTC_PCR7=0x100; //PTC7 in GPIO mode
GPIOC_PDDR=0x80; //PTC direction register, PTC7 is in output mode
FTM0_SC=0x00;
FTM0_CONF=0xC0; //set up BDM in 11
FTM0_FMS=0x00; //clear the WPEN so that WPDIS is set in FTM0_MODE register
FTM0_MODE|=0x05; //enable write the FTM CnV register
FTM0_MOD=1000;
FTM0_C0SC=0x28; //High_Low_high for center-alignment
FTM0_C1SC=0x28;
FTM0_C2SC=0x28;
FTM0_C3SC=0x28;
FTM0_C4SC=0x28;
FTM0_C5SC=0x28;
FTM0_C6SC=0x28;
FTM0_COMBINE=0x020202; //complementary mode for CH0&CH1, CH2&CH3 of FTM0
FTM0_COMBINE|=0x101010; // dead timer insertion enabled in complementary mode //for CH0&CH1
of FTM0
FTM0_DEADTIME=0x1F; //dead time is 31 system clock cycles
FTM0_CNTIN=0x00;
FTM0_C1V=500; //in complementary, C0V/C2V/C4V are used to control duty cycle.
FTM0_C0V=500; //in complementary mode, C1V/C3V/C5V are not used.
FTM0_C2V=500;
FTM0_C3V=500;
FTM0_C4V=500;
FTM0_C5V=1000;
//triggering ADC at the center point of PWM signal while PWM is set up in //center alignment
mode
FTM0_SC=0x28; //PWM center alignment, system clock driving, dividing by 1
asm("nop");
//set that FTM0_C5V to trigger ADC, in complementary, C5V is not used to //control duty cycle
FTM0_EXTTRIG|=0x08;
}
 unsigned int sample[8];

void AdcInit(void)
{
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
SIM_SCGC3|=0x08000000; //enable ADC1 clock
ADC1_CFG1=0x44; //adc clock is bus clock/2, busclock is 20M, ADC clock is 10MHz
ADC1_CFG2=0x00;
ADC1_SC2=0x40; //hardware triggered,
ADC1_SC3=0x00; //not continuous conversion
ADC1_SC1A=0x54; //interrupt enable and select ADC1_DM1 channel
//set that FTM0 to trigger ADC1
SIM_SOPT7=0x8800; //FTM0 trigger ADC1, alternative triggering, only ADC1_SC1A //conversion is
valid
NVICIP58=0x30; //setting interrupt of ADC1
```

```
NVICICPR1|=1<<26;   //clear the pending register of interrupt source 58
NVICISER1|=1<<26;  //set the interrupt source of ADC1
asm("cpsie i"); //interrupt enable
}
void ADC_ISR(void)
{
       GPIOC_PTOR=0x80; //toggle indicator
       sample[0]=ADC1_RA;
       asm("nop");
       //ADC1_SC1A=0x14; //interrupt enable and select ADC1_DM1 channel
}
```

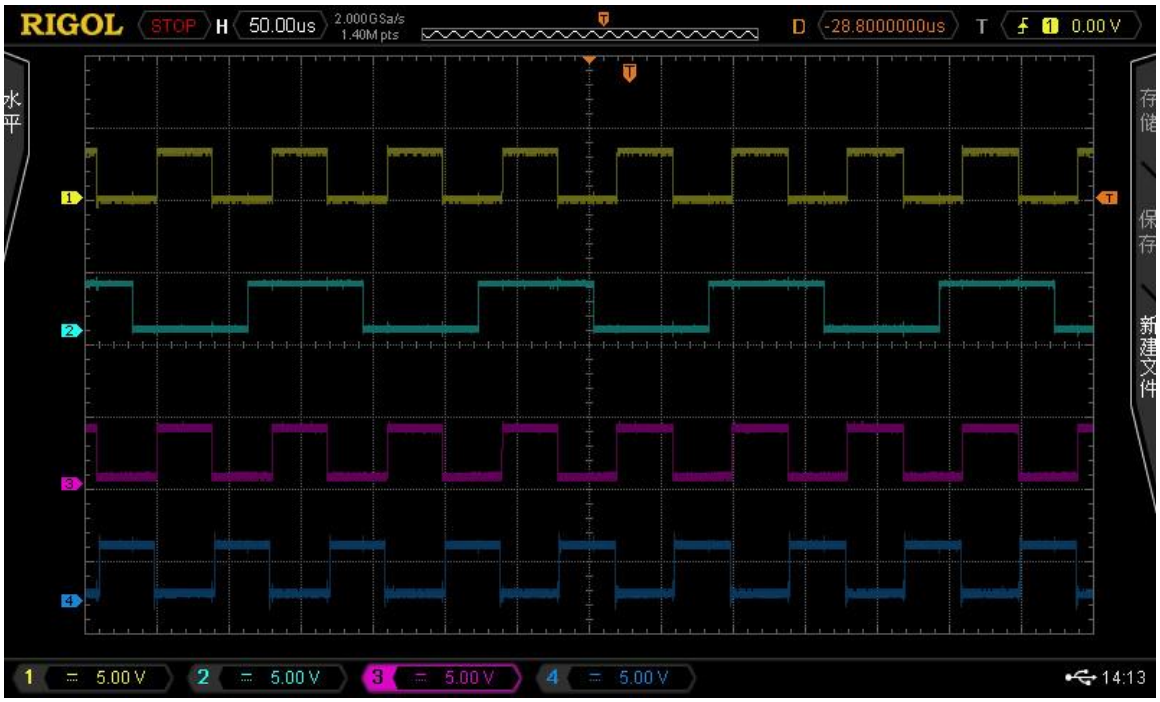

Figure 5. FTM triggering ADC

In Figure 5, the CH1 channel on oscilloscope is the FTM0_CH0 signal, CH2 channel on the oscilloscope is PTC7 pin toggling signal which is toggled in ADC_ISR by software, the CH3 and CH4 channels on oscilloscope are the FTM0_CH2 and FTM0_CH3 signals. The waveform shows that the PTC7 toggling instant (ADC sampling instant) occurs at the center of the Low voltage of FTM0_CH0 signal.

NOTE

In the complementary mode, only the FTM_C0V/C2V/C4V/C6V are used to control the PWM duty. The FTM_C1V/C3V/C5V/C7V registers are not used in this mode. Therefore, you can use one of the registers to control the ADC sampling instant as the example. The example uses the FTM_C5V register to control the ADC sampling instant by the instruction `FTM0_EXTTRIG|=0x08`.

3.8. Single-edge capture mode

The FTM capture mode is used to measure the duty cycle or cycle time of the tested signal, another utilization of capture mode is to detect the edge of external signal and generate interrupt to notify that an external event happens. Sometimes, you need to measure the tested signal duty cycle or cycle time which is low frequency signal compared to the tick, in the case, the tested signal can be connected to the FTM channel pin. For example, FTM_CHx, the FTM can detect the edge of the tested signal, copy the FTM_CNT value to FTM_CnV register automatically, and trigger an interrupt, in the ISR of FTM. You can read the FTM_CnV register value and save it to a variable, when another edge is detected and an interrupt is triggered, in the same ISR of FTM, read the FTM_CnV to another variable. You can figure out the difference of the two variable, if the mode is captured on different edge, the difference is duty cycle. If the mode is captured on same edge, the difference is period time.

Configuration:DECAPEN=COMBINE=CPWMS=0, MSnB:MSnA=00

When ELSnB:ELSnA=01, the mode is Capture on rising edge, when ELSnB:ELSnA=10, the mode is *capture on falling edge*. Both the modes are used to measure the cycle time.

When ELSnB:ELSnA=11, the mode is Capture on rising/falling edge, the mode is used to test duty cycle.

When an edge is detected, the channel flag bit CHF in corresponding FTM_CnSC register is set automatically. If the CHIE bit in the same register is set, an interrupt even will happen.

In BLDC motor control application, Hall sensor is used to detect the position of rotor and compute the velocity so that a speed loop can be established. The Hall sensor is connected to the channel pin of independent FTM (FTM_CHx), the FTM is set up in single-edge capture mode using above setting. FTM can detect both falling/rising edge of Hall signal, and generate a capture interrupt. In the capture ISR, you can read the logic of GPIO pins of 3 Hall signal and then decide to turn on or turn off the PWM signals based on the three Hall logic.

NOTE

The bits in the GPIO port input register GPIOx_PDIR can reflect the pin logic even if the pin is not configured for the GPIO mode, but is configured as peripheral mode.

The following code demonstrates how to configure single capture mode for FTM.

Example 7. Configuring single-capture mode

```
void singleCapture(void)
{
//enable FTM0 clock
SIM_SCGC6|=0x03000000; //enable FTM0 and FTM0 module clock
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
FTM0_SC=0x00;
FTM0_C0SC|=0x04;   //Capture on Rising Edge Only
FTM0_COMBINE=0x00; //clear
//enable capture interrupt
FTM0_C0SC|=0x40;  //enable CH0 interrupt
FTM0_SC|=0x08;
//in ISR of capture interrupt, read the FTM_c0V register to get the capture value
}
Void FTM0_ISR(void)
{
if(FTM0_STATUS&0x01)
```

```
{
        if(FTM0_STATUS&0x01)
        {
                tempPrev=temp;
                temp=FTM0_C0V; //read FTM0 counter register
                diff=temp-tempPrev;
                FTM0_STATUS&=0xFE;
                //read Hall sensor logic
               //The FTM0_CH0 channel is multiplexed with GPIOC1, read GPIOC1 logic
                var0=GPIOC_PDIR;
                GPIOC_PTOR=0x80;
                asm("nop"); //set a break point here
        }
}
```

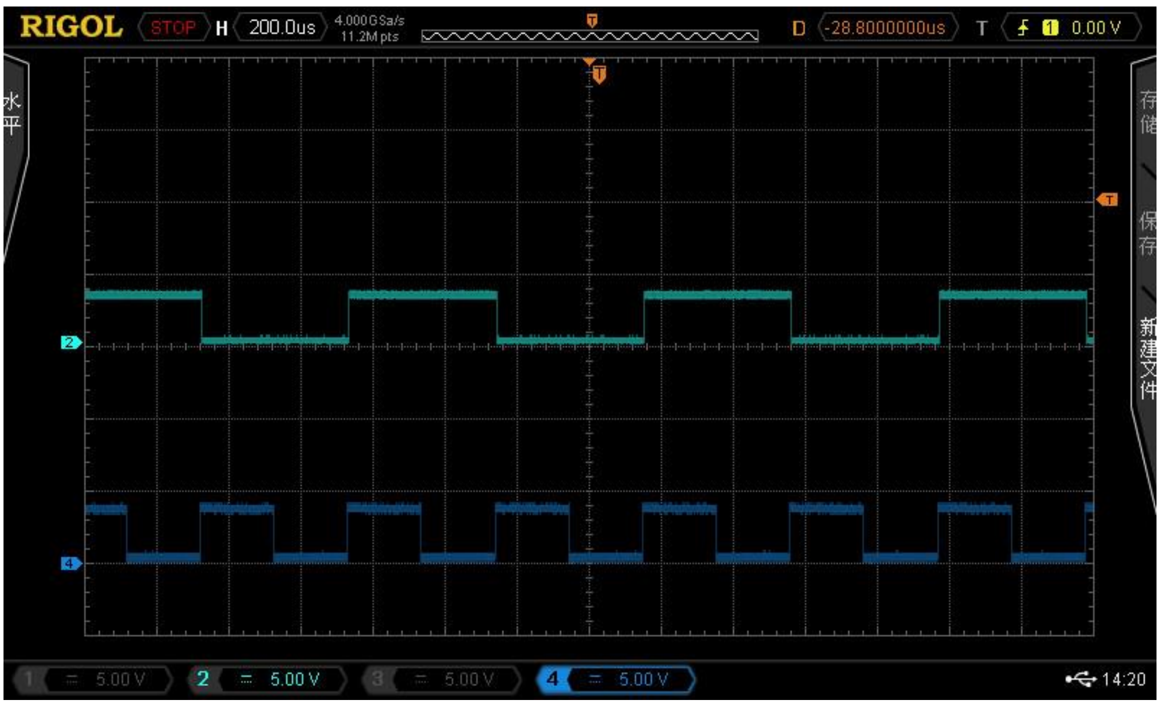

Figure 6. Single-capture mode waveform

In Figure 6, the CH4 channel on oscilloscope is the tested signal, CH2 channel on the oscilloscope is the GPIOC7 signal, which is toggled in the FTM0_ISR. The waveform shows that the rising edge of the captured signal can trigger a capture interrupt.

3.9. Dual-edge capture mode

The mechanism of dual-edge capture mode is similar to the single-edge capture mode. The only difference between the dual-edge capture mode and the single-edge capture mode is that the dual-edge capture mode uses two FTM channels to test the duty cycle time or period of the tested signal. In the dual-edge capture mode, the tested signal must input from even FTM channels. For example, FTM_CH0/FTM_CH2/FTM_CH4/FTM_CH6 input pin, the odd channel signal is ignored.

Figure 7 is a diagram of the dual-edge capture mode.

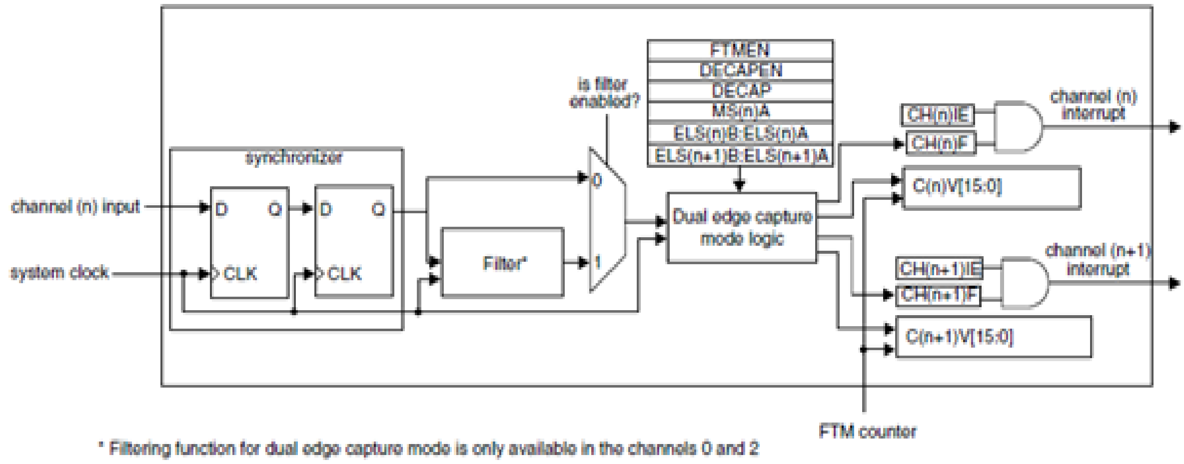

Figure 7. Dual-edge capture mode

3.9.1. Configuration for the dual-edge capture mode.

FTMEN=1, DECAPEN=1, the ELSnB:ELSnA bits select the first edge of the captured signal. The ELSn+1B:ELSn+1A bits select the second edge of the captured signal. MSnA bit select either the one-shot mode or the continuous mode.

The one-shot mode is selected when the MSnA bit is cleared; setting the DECAPx bit launches the capturing process. When the first edge of captured signal is detected, the FTM counter value is copied to FTM_CnV register, while the DECAPx bit keeps *1* state. When the second edge of the captured signal is detected, the FTM counter value is copied into FTM_Cn+1V register, and the DECAPx bit is cleared automatically, CHn+1F bit in FTM_Cn+1SC register is set. If the FTM_CHn+1 channel interrupt is enabled with CHIE bit in FTM_Cn+1SC set, an interrupt occurs, in the ISR of capture interrupt. You can read the FTM_Cn+1V and FTM_CnV register, find the difference, and get the duty cycle or the period.

The continuous dual-edge capture mode is selected when MSnA bit is set. Setting the ENCAPx bit launches the capturing process also. In the continuous dual-edge capture mode, the edge in the FTM channel is captured continuously. When the second edge of the captured signal is detected, CHn+1F bit in FTM_Cn+1SC register is set, if the FTM_CHn+1 channel interrupt is enabled with CHIE bit in FTM_Cn+1SC is set, an interrupt occurs. You can read the FTM_Cn+1V and FTM_CnV register and compute the duty cycle or the period.

You can select the rising or falling edge of the captured signal by the ELSnB:ELSnA bits for the first edge and ELSn+1B:ELSn+1A bits for second edge. If the selected two edges are the same, period is measured. If the selected two edge are different, the duty cycle is measured. If the first edge is rising and the second edge falling, the high pulse time is measured. If the first edge is falling and the second edge rising, the low pulse time is measured.

The following code demonstrates how to set up FTM in dual-edge capture mode.

Example 8. Setting the FTM in dual-edge capture mode

```
void dualCapture(void)
```

```
{
SIM_SCGC6|=0x03000000; //enable FTM0 and FTM0 module clock
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
FTM0_SC=0x00;
FTM0_CNTIN=0x00;
FTM0_MOD=0xFFFF;
FTM0_MODE=0x05;  //set FTMEN bit
FTM0_C0SC|=0x04;   //Capture on Rising Edge Only
FTM0_COMBINE=0x04; //enable dual capture mode
//enable capture interrupt
FTM0_C1SC|=0x04;   //Capture on Rising Edge Only
FTM0_C1SC|=0x40;  //enable CH1 interrupt
FTM0_SC|=0x08;
asm("nop");
FTM0_COMBINE|=0x08; //setting DECAP0 bit to launch the dual capture process
}
void FTM0_ISR(void)
{
if(FTM0_STATUS&0x01)
       {
       //read FTM0 counter register
       tempPrev=FTM0_C0V;
       temp=FTM0_C1V;
       diff=temp-tempPrev;
       FTM0_STATUS&=0xFD;
       //The FTM0 module CH0 channel is multiplexed with GPIOC1, read GPIOC1 logic to get the
       logic of the FTM0_CH0 pin at the instant.
       var0=GPIOC_PDIR;
       GPIOC_PTOR=0x80;
       asm("nop"); //set a break point here
       }

}
```

For the code in Example 8, the tested signal is input from the FTM0_CH0 pin (GPIOC1 pin of K40) and FTM0 works in one-shot mode. Setting the DECAP0 bit in FTM0_COMBINE register launches the capture process. When the second edge of tested signal is captured, the FTM0_ISR is entered (capture interrupt), in the ISR. You have to identify the interrupt source, clear the corresponding bits in status register, and compute the tick number between the two edges.

3.10. Quadrature decoder mode

The Encoder sensor is a position sensor. The quadrature decoder mode of FTM can decode the signals from the Encoder sensor. The Encoder sensor can generate two or four signal. For the absolute Encoder sensor, it can generate four signals: Home, Index, Phase A, Phase B signals. The Home pulse signal is generated by the encoder, when the rotor spins to a known a position. The Index signal can provide the revolution number of the rotor, the Phase A and Phase B can provide the both the position and direction of the rotor. The Phase A and Phase B are two signals with 90 degree shift. For the incremental encoder, it can generate only two signal Phase A and Phase B, the two signals are phase shift with 90 degree, because there is no Home pin, it cannot provide the absolute position.

FTM provides dedicated pin for the Phase A and Phase B signals, but do not provide the Home and Index pins. FTM can only interface with incremental encoder sensor. In general, the FTM0 is applied to generate PWM signal to control a motor, the FTM1 or FTM2 can be applied to interface with Encoder.

For example, the K40 has three FTMs: FTM0, FTM1, FTM2, the FTM1 has FTM1_QD_PHA and FTM1_QD_PHB signal pins and can be used to decode the Encoder sensor signals.

Configuration: FTMEN=1, QUADEN=1

The quadrature decoder mode has two sub modes: count and direction encoding mode and Phase A and Phase B encoding mode. The QUADMODE bit in FTM_QDCTRL register selects the sub-modes.

For the Count and direction encoding mode, there are two signal: Phase A and Phase B, but they are not encoder sensor output signals. The Count and direction encoding mode is helpful for some applications. The Phase B signal controls the counting direction, the Phase A signal controls the counting rate. For example, when the Phase B signal is in high logic, the FTM counter will increase for every rising edge of Phase A signal. When the Phase B signal is in low logic, FTM decreases for every rising edge of Phase A signal. The decrease means that the FTM counter counts tick from FTM_MOD to FTM_CNTIN, then wraps to FTM_MOD, and continues to count.

For the Phase A and Phase B encoding mode, the mode is used to decode the encoder sensor signals exactly. The FTM counter counts each rising/falling edge of both Phase A and Phase B signals, but whether the FTM counter increases or decreases is only dependent on the phase relationship between Phase A and Phase B. When the Phase B lags behind Phase A signal, the FTM counter increases for each rising/falling edge of both Phase A and Phase B signals, while the QUADIR bit in FTM_QDCTRL register reflects the counting direction. When the Phase A lags behind Phase B signal, the FTM counter decreases for each rising/falling edge of both Phase A and Phase B signal.

The following code demonstrates how to set up FTM in quadrature mode.

Example 9. Setting the FTM quadrature mode

```
void quadratureDecoder(void)
{
//using FTM1 as quadrature module, enable FTM0 and FTM1 clock
SIM_SCGC6|=0x03000000; //enable FTM0 and FTM1 module clock
FTM1_SC=0x00; //initialize the FTM1 module as quadrature mode
FTM1_MOD=20;
FTM1_CNTIN=0x00;
FTM1_C0SC=0x00;
FTM1_C1SC=0x00;
FTM1_QDCTRL|=0x01; //enable quadrature mode
FTM1_MODE|=0x05; //enable FTM1 module
//quadrature mode which PhaseA and PhaseB signal have 90 degree shift
FTM1_SC=0x48; //enable counter overflow interrupt
}
void FTM1_ISR(void)
{
GPIOC_PTOR=0x80;
FTM1_SC&=0x7F; //clear overflow flag
asm("nop");
}
void Ftm1_interruptInit(void)
{
NVICIP63=0x30; //set FTM0 interrupt priority
NVICICPR1|=1<<31;   //clear the pending register of interrupt source 69(PIT1)
NVICISER1|=1<<31;  //set the interrupt source of PIT1
}
```

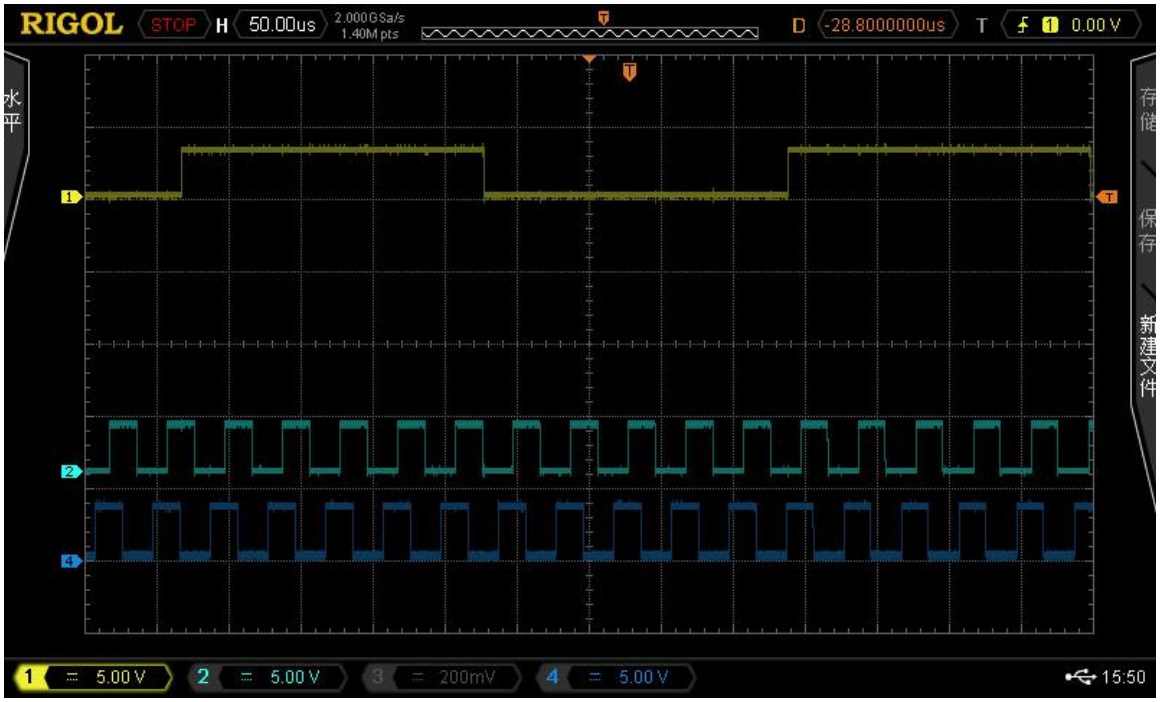

Figure 8. Quadrature counting waveform

In Figure 8, the CH2 and CH4 channels on the oscilloscope are Phase A and Phase B signals from incremental encoder sensor with 90 degree phase shift, which is input from FTM1_QD_PHA (PTB0 of K40) and FTM1_QD_PHB (PTB1 of K40).

The CH1 of the scope is the signal of GPIOC7 pin which is toggled in the FTM1_ISR. You can see that the FTM1 overflow interrupt happens for five pulse of both Phase A and Phase B signals. There are 20 edge of both Phase A and Phase B. In the example, FTM1_MOD=20.

3.11. Updating the FTM registers

In motor control or switch mode power supply applications, the voltage impressed on load is controlled by the PWM duty cycle. Thus, it is necessary to change the FTM_CnV register, change the duty cycle of PWM signal, and then change the voltage impressed on load.

When a value is written to register such as FTM_CnV or FTM_MOD, in some case, the writing just writes the value to Write Buffers of the register. In other case, the writing updates the register immediately and is dependent on the FTM working state and register setting.

For more information, see the *K40P144M100SF2 Reference Manual* (document K40P144M100SF2RM).

3.11.1. Writing the FTM register in the initialization stage

In initialization stage while the FTM is disabled with CLKS=0 in FTM_SC register, writing the FTM_CnV/FTM_MOD register updates the register immediately.

You need to first clear the CLKS=0 in FTM_SC and then set the FTMEN bit in FTM_MODE. The writing to FTM_CNTIN and FTM_CnV, FMT_MOD register takes effect immediately. After all the FTM register are initialized, setting the CLKS bit in FTM_SC register based on the selected clock source will start FTM.

For details see the void `PWMOutput_EdgeAlignment(void) api` function.

3.11.2. Updating the FTM registers with immediate load

To change the PWM duty cycle or the PWM cycle time when the FTM is running, you can set the LOAD bit in FTM_PWMLOAD register to update the FTM_CnV or FTM_MOD. This changes the PWM duty cycle or the PWM cycle time and the loading point can be selected by FTM_PWMLOAD register.

NOTE

The features such as PWM pin masking, software controlling, and PWM pin inverting cannot function with the LOAD bit setting.

You can generate an overflowing interrupt of FTM by setting the TOIE bit. In the interrupt service routine, you can write the the FTM_CnV or FTM_MOD register with new value and then set the LDOK bit in the FTM_PWMLOAD register. The FTM_CnV or FTM_MOD register is updated. You can write FTM_CnV or FTM_MOD register and then set LOAD bit anytime besides in ISR of overflow event.

3.11.2.1. Configuration

The CLKS bits in FTM_SC register cannot be 0, which means FTM should be running. The FTMEN bit in FTM_MODE is set. The PWMSYNC bit in FTM_MODE is cleared, which means that both software and hardware can update the FTM_MOD, FTM_CnV register.

The SYNCMODE bit in FTM_SYNCONF register is set.

IF CNTINC bit in FTM_SYNCONF is set, the CNTIN register is enabled to update. If the SYNCENx bit is set, the corresponding FTM_CnV and FTN_Cn+1V register is enabled to update.

It is unnecessary to set the CHxSEL bits, in ISR of FTM overflow. After you write the FTM_CnV, set the LOAD bit in FTM_PWMLOAD register, and the corresponding FTM registers are updated.

NOTE

The loading event cannot update the FTM_INVCTRL, FTM_OUTMASK, and FTM_SWOCTRL registers.

The following code demonstrates how to set the LOAD bit to change the PWM duty cycle.

Example 10. Setting the LOAD bit to change the PWM duty cycle

```
void FTM0CenterAlign(void)
{
SIM_SCGC6|=0x03000000; //enable FTM0 and FTM0 module clock
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
FTM0_CONF=0xC0; //set up BDM in 11
FTM0_FMS=0x00; //clear the WPEN so that WPDIS is set in FTM0_MODE register
FTM0_MODE|=0x05; //enable write the FTM CnV register
FTM0_MOD=1000;
FTM0_C0SC=0x28; //High_Low_High_ for center-alignment
```

```
FTM0_C1SC=0x28;
FTM0_C2SC=0x28;
FTM0_C3SC=0x28;
FTM0_COMBINE|=0x2020; //enable update the FTM_C0V/FTM0_C1V register
FTM0_COMBINE|=0x0202; //complementary mode for CH0&CH1, CH2&CH3
FTM0_COMBINE|=0x1010; // dead timer insertion enabled in complementary mode //for CH0&CH1 of
FTM0
FTM0_DEADTIME=0x00; //dead time is 0 system clock cycles
FTM0_C1V=500;
FTM0_C0V=500;
FTM0_C2V=500;
FTM0_C3V=500;
FTM0_SYNC|=0x01;
FTM0_SYNCONF|=0x80; //enable enhanced PWM synchronization
FTM0_CNTIN=0x00;
FTM0_SC=0x68; //PWM center_alignment, system clock driving, dividing by 1
       }

void FTM0_ISR()
{
static bool flag;
       if(FTM0_SC&0x80)
       {
              FTM0_SC&=0x7F; //clear TOF bit
              if(flag)
              {
                     FTM0_C0V=750;
                     FTM0_C1V=750;
                     FTM0_C2V=750;
                     FTM0_C3V=750;
              }
              else
              {
                     FTM0_C0V=500;
                     FTM0_C1V=500;
                     FTM0_C2V=500;
                     FTM0_C3V=500;
              }
              flag=!flag;
              FTM0_PWMLOAD|=0x200;
       }
}
```

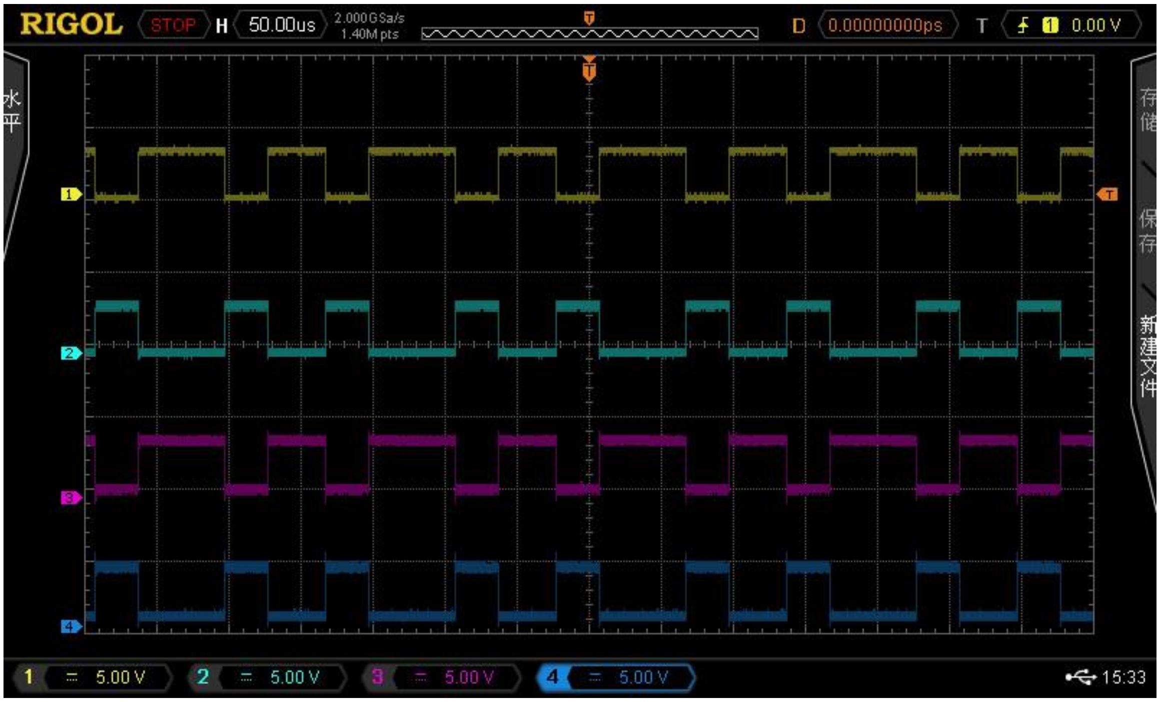

Figure 9. Updating the duty cycle with PWM loading

In Figure 9, the CH1, CH2, CH3, and CH4 channels on the oscilloscope are waveform of the FTM0_CH0, FTM0_CH1, FTM0_CH2 and FTM0_CH3, the duty cycle of the FTM0 channel is 50% or 75% and is changed for every PWM cycle. Setting the LDOK bit enables the FTM register update.

3.11.3. Updating the FTM register with software triggering

With appropriate register configuration, setting the SWSYNC bit in FTM_SYNC register updates the FTM_CnV, FTM_MOD with its buffer value and the process is called as software synchronization. The update can take place immediately or at the next selected loading point: Boundary Cycle and Loading Points. If the SWRSTCNT bit is set, the FTM counter restarts with FTM_CNTIN register value and the FTM_MOD and FTM_CnV registers are updated immediately. If the SWRSTCNT bit is cleared, the FTM counter continues to count normally and the FTM_MOD and FTM_CnV register update at the next loading point. For up counting mode, the FTM counter counts from FTM_CTNIN to FTM_MOD value, then wrap to CNTIN. The loading point occurs at the point when the FTM counter wraps from FTM_MOD to FTM_CNTIN. For up-down counting mode, the FTM counter counts up from CNTIN register value to FTM_MOD value, then counts down from FTM_MOD value to CTNIN value, then repeats. The loading points are enabled if one of CNTMIN or CNTMAX bits in FTM_SYNC are 1. Updating the FTM registers in loading point can guarantee that the PWM output signals have intact cycle.

The FTM registers which the software triggering can update include the FTM_CnV, FTM_MOD, FTM_INVCTRL, FTM_OUTMASK, and FTM_SWOCTRL registers.

The following code demonstrates how to update FTM_CnV and FTM_OUTMASK registers with software triggering.

Example 11. Update FTM_CnV and FTM_OUTMASK registers with software triggering

```
void FTMRegisterUpdate(void)
{
SIM_SCGC6|=0x03000000; //enable FTM0 and FTM1 module clock
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
FTM0_CONF=0xC0; //set up BDM in 11
FTM0_FMS=0x00; //clear the WPEN so that WPDIS is set in FTM1_MODE register
FTM0_MODE|=0x07; //enable write the FTM CnV register
FTM0_MOD=1000;
FTM0_C0SC=0x28; //High_Low_High for center-alignment
FTM0_C1SC=0x28;
FTM0_C2SC=0x28;
FTM0_C3SC=0x28;
FTM0_C4SC=0x28;
FTM0_C5SC=0x28;
FTM0_COMBINE= 0x020202; //complementary mode for CH0&CH1 of FTM1
FTM0_COMBINE|=0x303030; //enable update the CnV and MOD register, dead timer //insertion
enabled
FTM0_DEADTIME=0x0F; //dead time is 32 system clock cycles
FTM0_C1V=750;
FTM0_C0V=250;
FTM0_C3V=750;
FTM0_C2V=250;
FTM0_C5V=750;
FTM0_C4V=250;
FTM0_C7V=750;
FTM0_C6V=250;
FTM0_CNTIN=0x00;
FTM0_SC=0x68; //PWM center_alignment, system clock driving, dividing by 1
FTM0_POL|=0x03; //the masked pin are HIGH logic
FTM0_CONF|=0x03; //four PWM cycle  generate one overflow interrupt
FTM0_SYNCONF|=(FTM_SYNCONF_SWWRBUF_MASK|FTM_SYNCONF_SYNCMODE_MASK); //SET THE //SWWRBUF BIT
AND SWRSTCNT BIT
FTM0_SYNCONF|=FTM_SYNCONF_SWOM_MASK; //enable mask function
FTM0_SYNC|=(FTM_SYNC_SWSYNC_MASK|FTM_SYNC_CNTMIN_MASK);
}
Void FTM0_ISR(void)
{
       if(FTM0_SC&0x80)
             {
              FTM0_SC&=0x7F;  //clear the overflow bit
              if(flag)
              {
                     FTM0_C0V=250;
                     FTM0_C1V=250;
                     FTM0_C2V=250;
                     FTM0_C3V=250;
                     FTM0_C4V=250;
                     FTM0_C5V=250;
                     FTM0_OUTMASK|=0x03; //enable mask
                     if(flagMask)
                     {
                            FTM0_POL|=0x03; //the masked pin are HIGH logic
                     }
                    else
                     {
                            FTM0_POL&=0xFC; //the masked pin are Low logic
                     }
                     flagMask=!flagMask;
              }
              else
              {
                     FTM0_C0V=500;
```

```
            FTM0_C1V=500;
            FTM0_C2V=500;
            FTM0_C3V=500;
            FTM0_C4V=500;
            FTM0_C5V=500;
            FTM0_OUTMASK&=0xFC; //disable mask
        }
        flag=!flag;

        FTM0_SYNC|=FTM_SYNC_SWSYNC_MASK;
    }
```

Figure 10. Updating duty cycle and masking FTM channel pins with software synchronization

In Figure 10, the CH3 and CH4 channels on the oscilloscope are the waveform of FTM0_CH2 and FTM0_CH3 and the duty cycles of both channels are updated for 4 PWM cycles by software synchronization. The CH1 and CH2 channels on the oscilloscope are the waveform of FTM0_CH0 and FTM0_CH1, the interleaving PWM signal and masked signal are outputted for the four PWM cycles. The last two bits of FTM0_POL register are interleaved between 1 and 0, which can change the logic of mask output.

3.11.4. Updating the FTM register with hardware control

This is an example to use Hardware triggering PWM synchronization scheme. For power factor control, the boost layout is used; the critical conduction mode is used as control regulation. During the on-time of the PWM signal, the current flowing through the inductor increases. During the off-time of the PWM signal, the current decreases and the current signal is compared with a threshold via hardware comparator. When the current is less than the threshold, the comparator will output high logic. The rising edge of comparator output signal is required to start the FTM and output PWM signals.

For more information, see the device reference manual for your microcontroller. The triggering sources include on-chip comparator, PDB output, and dedicated FTM hardware triggering pins.

Three hardware trigger signal inputs of FTM are enabled when TRIGn = 1, where *n* = 0, 1, or 2 corresponding to each one of the input signals respectively. The hardware triggering 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.

Configuration: Set the PWMSYNC and FTMEN bits in FTM_MODE register and set the SYMCMODE in FTM_SYNCONF register. Set the HWWRBUF and HWTRIGMODE bits in FTM_SYNCONF register. If you want to have FTM launches a new PWM cycle (FTM counter restarts from FTM_CNTIN register), set the HWRSTCNT bit in the FTM_SYNCONF register.

The following code configures the PTB3 pin as FTM0_FLT0. The FTM0_FLT0 is the hardware triggering source TRIG2. Every rising edge of trigger source2 launches a new PWM cycle.

Example 12. Configuring the PTB3 pin as FTM0_FLT0

```
void FTM0CenterAlign(void)
{
SIM_SCGC6|=0x03000000; //enable FTM1 and FTM0 module clock
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
FTM0_CONF=0xC0; //set up BDM in 11
FTM0_FMS=0x00; //clear the WPEN so that WPDIS is set in FTM0_MODE reg
FTM0_MODE|=0x05; //enable write the FTM CnV register
FTM0_MOD=1000;
FTM0_C0SC=0x28; //High_Low_High for center-alignment
FTM0_C1SC=0x28; //High_Low_High for center-alignment
FTM0_COMBINE|=0x20; //enable update the FTM_C0V/FTM0_C1V register
FTM0_COMBINE|=0x02; //complementary mode for CH0&CH1 of FTM0
FTM0_COMBINE|=0x10; // dead timer insertion enabled in complementary mode for //CH0&CH1 of
FTM0
FTM0_DEADTIME=0x00; //dead time is 0 system clock cycles
FTM0_C1V=500;
FTM0_C0V=500;
FTM0_CNTIN=0x00;
FTM0_SC=0x28; //PWM center_alignment, system clock driving, dividing by 1
}
void hardwareTriggerSeting(void)
{
//enable hardwareTrigger 2,the triggering source is from external pin FTM0_FLT0 pin
SIM_SOPT4&=~(0x01); //FTM0_FLT0 pin is the fault signal, chip by chip
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
PORTB_PCR3=0x600; //PTB3 in FTM mode    :FTM0_FLT0
//enable the TRIG2 signal to trigger FTM0
FTM0_SYNC|=0x40;
FTM0_SYNCONF|=0x30001; //HWWRBUF=1: MOD, CnV all updated by hardware triggering, HWRSTCNT=1:
FTM0 counter count from CNTIN register value
}
```

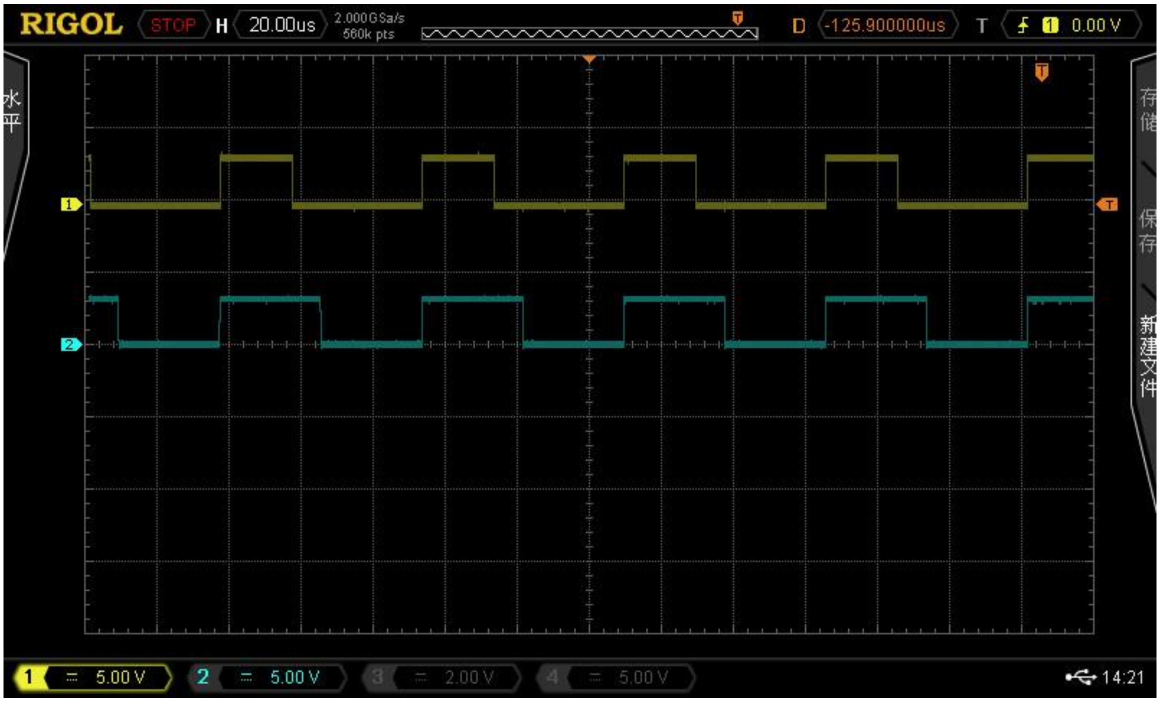

Figure 11. FTM hardware triggering mode

In Figure 11, the CH2 channel on the oscilloscope is triggering signal, the CH1 channel on the oscilloscope is the FTM0_CH0 output signal. You can see that each rising edge of triggering signal starts a new PWM cycle.

3.12. Masking, inverting, and software controlling features

Masking and software controlling features enable the FTM_CHx pin to keep a constant logic rather than a low/high interleaving PWM signal, which can turn on/off power devices directly. The inverting feature exchanges the signals of a pair of FTM channels which are complementary. For example FTM_CH0 and FTM_CH1 are a pair of complementary signals. After the inverting, the FTM_CH0 and FTM_CH1 signals exchange each other, the FTM_CH0 pin outputs the signal which the FTM_CH1 outputs originally, the FTM_CH1 pin outputs the signal which the FTM_CH0 pin outputs originally.

The masking features are used in BLDC motor control application. For BLDC motor control application, three branches are required. This means that six power devices (MOSFET/IGBT) are used, but only two branches turn on. The other branch can turs off anytime. For the branch turning off, no current flows through the turning off branch, and as a result the corresponding MOSFET/IGBT turns off. The masking feature can implement the function to turn off the two power devices. Each FTM channel has a respective bit in the FTM_OUTMASK register, when the FTM_OUTMASK register is written, the value is written into the corresponding register buffer. The FTM pins logic can update immediately after writing the FTM_OUTMASK register or update after software triggering or hardware triggering; it is dependent on the SYNCHOM bit in FTM_SYNC register. When a channel of FTM FTM_CHx is masked, the logic of the FTM_CHx pin becomes the value in safe mode, which is determined by the corresponding bit in FTM_POL register.

Software controlling feature can set/clear the logic of FTM channel pins by setting or clearing a bit in register. In case the FTM channels pin FTM_CHx keeps high or low logic. Setting the CHxOC bit in

FTM_SWOCTRL register means that the corresponding FTM channel FTM_CHx is controlled by software. Clearing the bit means that the FTM channel FTM_CHx is driven by PWM signal or the other source. The CHxOCV bit in FTM_SWOCTRL register controls the logic of the corresponding FTM channel, setting the CHxOCV bit will make the FTM_CHx channel high, clearing the bit will make the pin low. You can update the FTM pins logic immediately after writing the FTM_SWOCTRL register or after software or hardware triggering. It is dependent on the SWOC bit in FTM_SYNCONF register. The update scheme is the same as of the masking function.

The inverting function is mainly used in BLDC motor control application. Assume that you control a DC motor with a H bridge, a pair of signals the FTM_CH0/CH1 is connected to one branch, another pair of signals FTM_CH2/CH3 is connected to another branch, and each pair of signals works in complementary mode. It is required that diagonal power devices turn on or turn off at the same time which means that the FTM_CH0 and FTM_CH3 should have the same timing. If, FTM_CH0 and FTM_CH2 have the same timing, FTM_CH1 and FTM_CH3 have the same timing originally, setting the INVC bit can make a pair of signal inverting. For example, making the FTM_CH2 and FTM_CH3 invert each other. In this way, you can implement FTM_CH0 and FTM_CH3 to have the same timing. Setting the INVxEN bit in FTM_INVCTRL register enables the corresponding FTM channel pair invert.

You can update the inverting of the FTM pin pair immediately after writing the FTM_INVCTRL register or after software or hardware triggering. It is dependent on the INVC bit in FTM_SYNCONF register. The update scheme is the same as of the masking function.

For the masking function code, see “Updating the FTM register with software triggering”.

3.13. Fault signal disabling PWM output signals

Fault function is an important feature for applications as motor control it provides a mechanism to protect the power device and the controlled system. When the PWM signals are used to control a motor or the other instrument, over-temperature, over-voltage, over-current may happen. In such a case, a fault signal can be generated via sensor or the other circuit. The fault signals can disable PWM signals automatically and immediately, while a fault signal can trigger an interrupt so that the core can deal with the situation. For Kinetis family, the PWM fault signals can be from external pads or on-chip peripherals such as comparator. For example, for K40, the FTM fault0 signal is from FTM0_FLT0 pin, it can also be from on-chip comparator CMP0 output.

Each FTM has only one interrupt vector, all the interrupt sources including Fault interrupt, FTM counter overflow interrupt, each channel compare event interrupt, and capture interrupt share the same interrupt vector. Checking the interrupt source and execute corresponding operation is dependent on the firmware in interrupt service routine.

When the fault signal appears, the FTM channel signals are disabled, and kept to a safe logic, which is defined in FTM_POL register. For example, if the POL0=0, when fault signals happen, the FTM_CH0 pin PWM signal is disabled, the FTM_CH0 pin is forced to become low logic. If the POL0=1, when fault signals happen, the FTM_CH0 pin PWM signal is disabled, the FTM_CH0 pin is forced to become high logic.

FTM has multiple channels, but FTM cannot disable specific channels with specific fault signal All fault signals do an OR operation and generate one FAULT signal. Whether the FTM channel can be disabled or not by fault signal is dependent on the FAULTENx bits setting in the FTM_COMBINE register. The fault signal can disable all the FTM channels or only even channels (FTM_CH0/CH2/CH4/CH6) or only odd channels (FTM_CH1/CH3/CH5/CH7). The FAULTM bits in FTM_MODE register determines the configuration.

Regarding the recovering of the PWM signals, there are two modes: Automatic Fault Clearing and Manual Fault Clearing. When the FAULTM=11 in the FTM_MODE register, the Automatic Fault Clearing mode is set up. When the fault signal disappears, the PWM signals will recover in next PWM cycle automatically without any software intervening. When the FAULTM=01 or 10 in FTM_MODE register, the Manual Fault Clearing mode is set up. When fault signal disappears, \after the software clears the FAULTF bit in FTM_MODE register, the PWM signals are recovered in the next PWM cycle from the disabled state.

The following code demonstrates how to set up the FTM registers so that fault signal can disable PWM signal from FTM channel pin.

Example 13. Setting the FTM registers to disable PWM signal by PWM fault signal

```
void faultFunc(void)
{
//the subroutine demonstrate the PWM fault feature of FTM, FTM is set up in
//complementary, edge-alignment, Combined mode.
//the fault signal inputs from FTM0_FLT3(PTB2 pin fro K40)
SIM_SCGC6|=0x03000000; //enable FTM0 and FTM0 module clock
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
FTM0_SC=0x00;
FTM0_CONF=0xC0; //set up BDM in 11
FTM0_FMS=0x00; //clear the WPEN so that WPDIS is set in FTM0_MODE register
FTM0_MODE|=0x05; //enable write the FTM CnV register
FTM0_MOD=1000;
FTM0_C0SC=0x28; //High_Low_High for center-alignment
FTM0_C1SC=0x28;
FTM0_C2SC=0x28;
FTM0_C3SC=0x28;
FTM0_COMBINE|=0x40404040; //all channels from CH0~CH7 are controlled by fault
FTM0_COMBINE|=0x0303; //complementary mode for CH0&CH1, CH2&CH3 of FTM0
FTM0_COMBINE|=0x1010; // dead timer insertion enabled in complementary mode
FTM0_DEADTIME=0x4F; //dead time is 32 system clock cycles
FTM0_C1V=750;
FTM0_C0V=250;
FTM0_CNTIN=0x00;
FTM0_C2V=375;
FTM0_C3V=625;
FTM0_FLTCTRL=0x08;  //enable FTM0_FALT3 pin, which is PTB2
FTM0_MODE|=0x60; //set the FAULTM bits as 11, automatically clearing the fault //bit
//FTM0_MODE|=0x80; //if enable fault interrupt, set the FAULTIE bit, in ISR //of FTM, check
the FAULTF3 flag
FTM0_FLTPOL=0x00; //set High logic as fault signal logic
FTM0_POL=0x00; //cleared,the FTM channel pin is low when disabled by fault, //when set, PWM
pin is high when FTM pin disabled
FTM0_SC=0x08; //PWM edge_alignment, system clock driving, dividing by 1
```

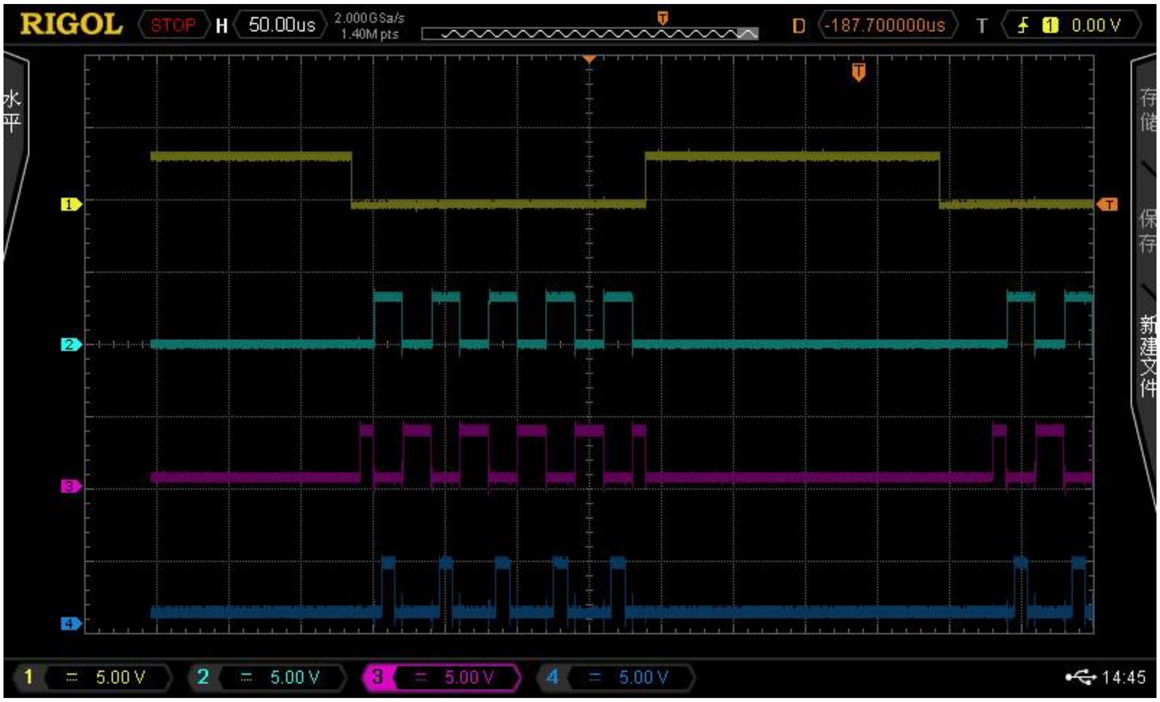

Figure 12. FTM fault function

In Figure 12, the CH1 channel on the oscilloscope is the fault signal FTM0_FLT3, the CH2 and CH3 channels on the oscilloscope are the signals of FTM0_CH0 and FTM0_CH1. The CH4 channel on the scope is the FTM0_CH2 signal.

The figure shows that the High logic of fault signal disables the FTM0 channels, and the FTM channels output Low logic. FTM0_POL register determines the logic. The FTM channels can recover automatically after the logic of Fault signal becomes low.

NOTE

The fault feature can only function when the FTM is set up in combined mode for Kinetis family. Currently, it is a disadvantageous. If you wants to use the fault function and use the center-alignment mode of PWM, you can set the FTM module in complementary, edge-alignment, and Combined mode. You can adjust the FTM_CnV register to have the FTM output center-alignment PWM signal so that the fault feature can be used.

3.14. Multiple FTM synchronization

There are multiple FTMs on a chip, but the multiple FTM modules are independent modules. If you need more PWM channels than what one FTM can provide, multiple FTM modules must be used and synchronized. This will enable the PWM to output signals from the different FTM modules and can synchronize with each other.

That two FTM modules synchronize means that the two FTM counters have the same value at any instant. In order to have two FTM synchronize, the first condition is that the two FTM counters count the same clock source. The second condition is that the two FTM counters start counting at the same time. The FTMs of the Kinetis family provides a global time base (GTB) mechanism to synchronize multiple FTMs, the global time base (GTB) is a synchronous signal generated by master FTM, which can have multiple FTM counters start counting synchronously.

The procedure to synchronize two FTMs: set the CLKS=00 in FTM_SC register for both FTM0/FTM1, configure both FTM0 and FTM1, set the GTBEEN=1 in both FTM0_CONF and FTM1-CONF register, enable both FTM0/FTM1 by setting the CLKS bits, at the time, both FTM0/FTM1 are waiting for the GTB signal, and set the GTBOUT=1 in FTM0_CONF. This starts both the FTM0 and FTM1.

The following code sets up both FTM0 and FTM1 such that the two FTM modules can synchronize.

Example 14. Set FTM0 and FTM1 to synchronize

```
void FTM1CenterAlign(void)
{
SIM_SCGC6|=0x03000000; //enable FTM0 and FTM0 module clock
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
FTM1_MODE|=0x05; //enable write the FTM CnV register
FTM1_CONF=0xC0; //set up BDM in 11
FTM1_FMS=0x00; //clear the WPEN so that WPDIS is set in FTM0_MODE reg
FTM1_MOD=1000;
FTM1_C0SC=0x28; // High_Low_High for center-alignment
FTM1_C1SC=0x28;
FTM1_COMBINE=0x22; //complementary mode for CH0&CH1 of FTM1, enable update //CH0/1 of FTM1
FTM1_COMBINE|=0x10; //dead timer insertion enabled for CH0&CH1 of FTM0
FTM1_DEADTIME=0x00; //dead time is 32 system clock cycles
FTM1_C1V=500;
FTM1_C0V=500;
FTM1_CNTIN=0x00;
//FTM1_SC=0x68; //do not start FTM1
}
void FTM0CenterAlign(void)
{
SIM_SCGC6|=0x03000000; //enable FTM0 and FTM0 module clock
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
FTM0_MODE|=0x05; //enable write the FTM CnV register
FTM0_CONF=0xC0; //set up BDM in 11
FTM0_FMS=0x00; //clear the WPEN so that WPDIS is set in FTM0_MODE register
FTM0_MOD=1000;
FTM0_C0SC=0x28; //High_Low_High for center-alignment
FTM0_C1SC=0x28;
FTM0_C2SC=0x28;
FTM0_C3SC=0x28;
FTM0_COMBINE|=0x2020; //enable update the FTM_C0V/FTM0_C1V register
FTM0_COMBINE|=0x0202; //complementary mode for CH0&CH1 of FTM0
FTM0_COMBINE|=0x1010; // dead timer insertion enabled in complementary mode //for CH0&CH1 of
FTM0
FTM0_DEADTIME=0x00; //dead time is 0 system clock cycles
FTM0_C1V=500;
FTM0_C0V=500;
FTM0_C2V=500;
FTM0_C3V=500;
FTM0_CNTIN=0x00;
//FTM0_SC=0x68; //do not start FTM0
}
void FTM0SynFTM1(void)
{
//The FTM0 module generates the GTB signal
//setting GTBEEN bit for both FTM0/FTM1,
FTM0_CONF|=0x200;
FTM1_CONF|=0x200;

//enable both FTM0 and FTM1, FTM0/FTM1 waiting for the GTB signal
```

```
FTM0_SC=0x68;
FTM1_SC=0x28;
//setting the GTBEOUT of FTM0 will start both FTM0/1, commenting the following line, FTM0/1
will not run
FTM0_CONF|=(0x400);
}
void FTM0_ISR()
{
static bool flag;
if(FTM0_SC&0x80)
        {
        FTM0_SC&=0x7F; //clear TOF bit
        if(flag)
        {
                FTM0_C0V=750;
                FTM0_C1V=750;
                FTM0_C2V=750;
                FTM0_C3V=750;
                FTM1_C0V=750;
                FTM1_C1V=750;
        }
        else
        {
                FTM0_C0V=500;
                FTM0_C1V=500;
                FTM0_C2V=500;
                FTM0_C3V=500;
                FTM1_C0V=500;
                FTM1_C1V=500;
        }
        flag=!flag;
        FTM0_PWMLOAD|=0x200;
        FTM1_PWMLOAD|=0x200;
        }

}
```

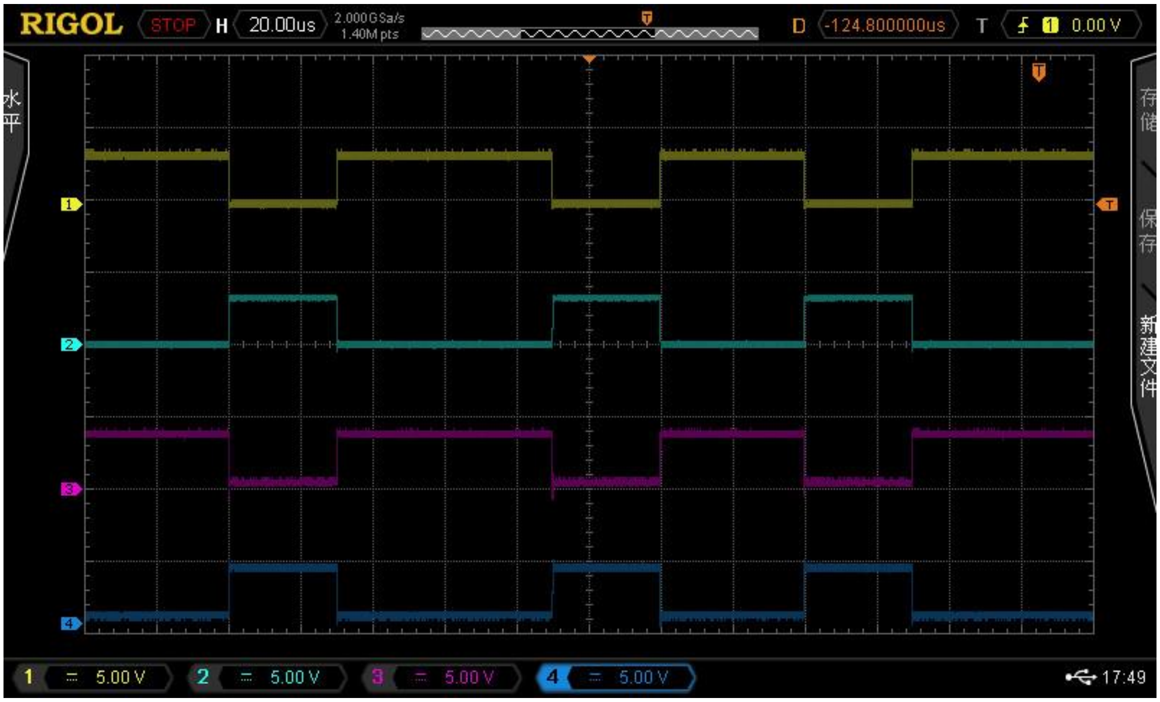

Figure 13. FTM0 and FTM1 synchronization

In Figure 13, the CH1 and CH2 channels on the oscilloscope are FTM0_CH0 and FTM0_CH1 signals, the CH3 and CH4 channels on the oscilloscope are FTM1_CH0 and FTM1_CH1 signals. You can see that both the FTM0 and FTM1 signals are synchronized. It also demonstrates that both FTM0/FTM1 can update the duty cycle.

NOTE

The `void Ftm0_interruptInit(void)` should be called to enable the overflowing interrupt.

3.15. FTM channel initial logic

In initializing stage, the FTM channels pins FTM_CHx is assigned as FTM function by setting the corresponding PORTx_PCRx register, but the FTM has not finished the initializing work by clearing the CLKS bits in FTM_SC register. The FTM_CHx is in tristate or in high impedance state. External pull up/down resistor determines the FTM channel logic, so it is recommended to connect a pull up/down resistor for the FTM_CHx such that the FTM_CHx has a deterministic logic to avoid turning on the power device unexpectedly.

If you want to stop the FTM, clear the CLKS bit, which stops FTM. The FTM_CHx pin will be in tristate, the external pull up/down resistor determines the logic of the FTM channel pins.

3.16. PWM resolution

You can express the PWM resolution in equivalent bits as the general DAC resolution. The PWM resolution can be expressed as resolution bit, the resolution bits = $\log 2^{(PWMModuloRegister)}$.
The PWM output frequency and resolution are not independent. The relationship is:

$2^{ResolutionBits}$ *(PWM output frequency) is PWM driving clock frequency in edge alignment mode or (PWM driving clock frequency)/2 in center-alignment mode.

Appendix A. Pin assignment code

The following code shows the pin assignment code for the Kinetis K40 family.

Example 15. Pin assignment code for Kinetis K40 family

```
void FtmPinassignment(void)
{
SIM_SCGC5=SIM_SCGC5|0x3E00; //enable port A/B/C/D/E clock
PORTC_PCR1=0x400; //PTC1 in FTM mode    :FTM0_CH0:
PORTC_PCR2=0x400; //PTC2 in FTM mode    :FTM0_CH1
PORTC_PCR3=0x400; //PTC3 in FTM mode    :FTM0_CH2
PORTC_PCR4=0x400; //PTC4 in FTM mode    :FTM0_CH3
PORTD_PCR4=0x400; //PTD4 in FTM mode    :FTM0_CH4
PORTD_PCR5=0x400; //PTD5 in FTM mode    :FTM0_CH5

PORTA_PCR1=0x300; //PTA1 in FTM mode    :FTM0_CH6
PORTA_PCR2=0x300; //PTA2 in FTM mode    :FTM0_CH7


PORTB_PCR2=0x600; //configure PTB2 as FTM0_FLT3 as ALT6

PORTC_PCR11=0x100; //PTC11 in GPIO mode and output mode
GPIOC_PDDR=0x800; //PTC direction register, PTC11 is in output mode

PORTC_PCR7=0x100; //PTC7 in GPIO mode and output mode
GPIOC_PDDR=0x80; //PTC direction register, PTC7 is in output mode


PORTA_PCR6=0x100; //PTA6 in GPIO mode and output mode
GPIOA_PDDR=0x40; //PTC direction register, PTA6 is in output mode

PORTB_PCR0=0x600; //PTB0  function as  FTM1_QD_PHA
PORTB_PCR1=0x600; //PTB1 functions as FTM1_QD_PHB
```

NOTE

For more information, see the *K40P144M100SF2 Reference Manual* (document K40P144M100SF2RM).

4. Revision history

Revision number	Date	Substantive changes
0	06/2015	Initial release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN5142
Rev. 0
06/2015