Two eFlexPWM module synchronization

The doc is written for the ticket: https://community.nxp.com/thread/440967

The documentation talks about how to synchronize two or more eFlexPWM modules on the same processor or different processor. There are applications which require more PWM channels for example some SR(switch reluctance with multiple phases) motor control application or dual motor control application using one processor or more processor. The eFlexPWM modules design itself allow multiple eFlexPWM modules to synchronize. The eFlexPWM synchronization means that all PWM channels are coherent, in other words, all PWM signal seems that they are from the same timer.

Two or more eFlexPWM module synchronization must meet two conditions, one is that all the eFlexPWM module use the same clock source, another is that external signal can reset the PWM generator counter. The first condition can be met easily because the eFlexPWM modules can be configured to use the same clock source. For the second condition, pls refer to the following figure, there are multiple signal can reset(init) the 16 bits counter, they are Local Sync, Master Reload, Master Sync and PWM_EXT_SYNC, yes, we can use the PWM_EXT_SYNC signal to synchronize different eFlexPWM module on the same processor or the eFlexPWM module on another processor.

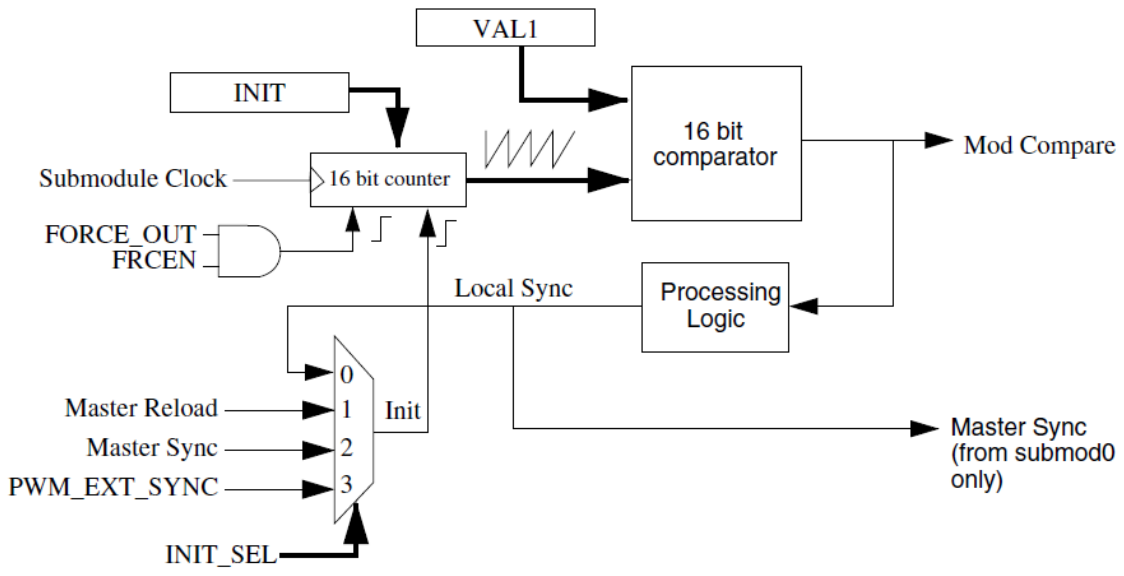


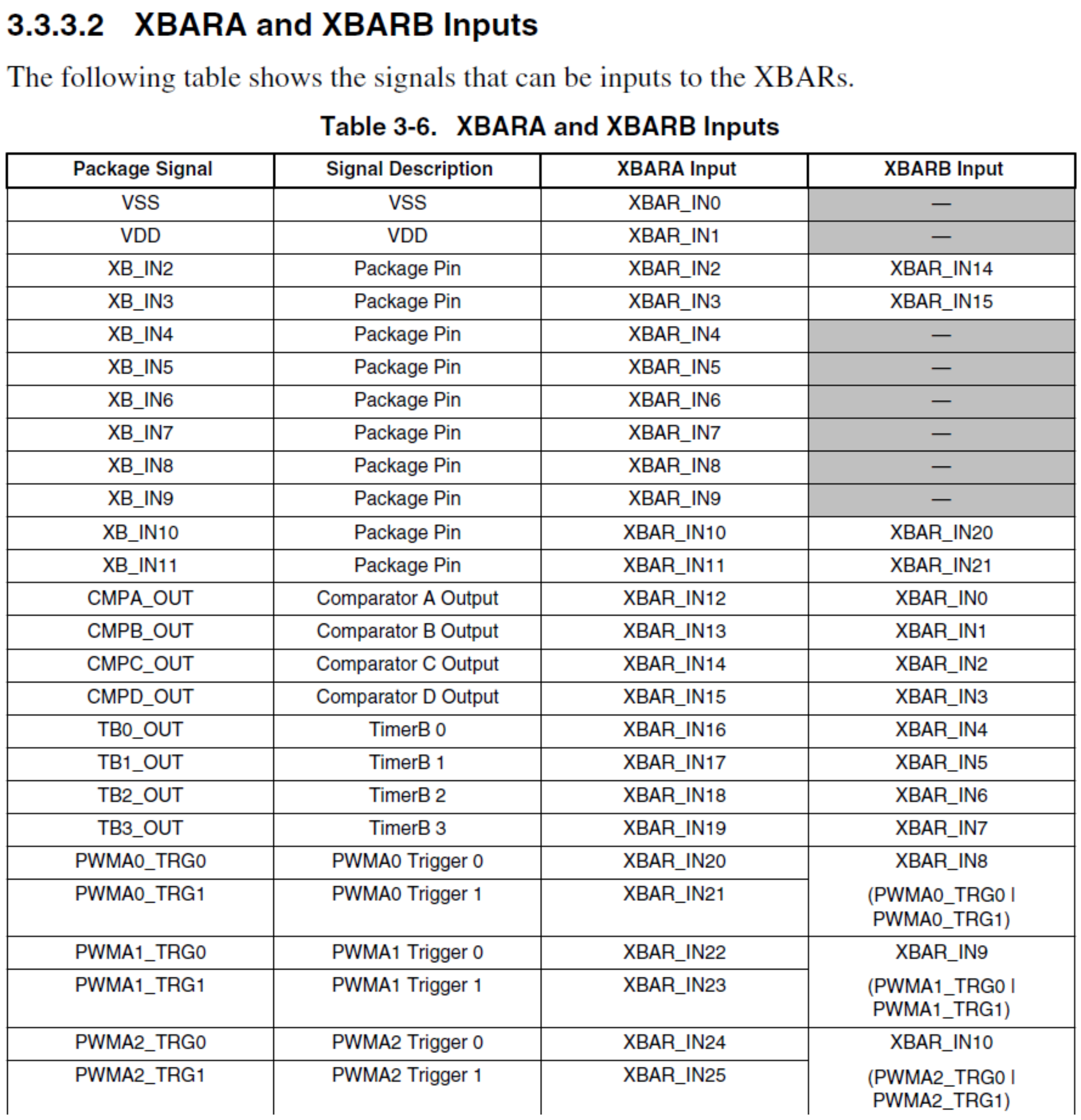
Figure 28-252. Submodule Timer Synchronization

For the Kinetis KV4x, KV5x and MC56F82xxx and MC56F84xxx, there is a crossbar, multiple signal can drive the PWM_EXT_SYNC signal.

This is the crossbar figure of MC56F84789.

Table 3-8. XBARA Outputs (continued)

XBARA Output	Signal	Signal Description
XBAR_OUT51	TA2_IN	Timer A2
XBAR_OUT52	TA3_IN	Timer A3
XBAR_OUT53	PWMB0_EXT_SYNC	PWMB0 Ext Synch
XBAR_OUT54	PWMB1_EXT_SYNC	PWMB1 Ext Synch
XBAR_OUT55	PWMB2_EXT_SYNC	PWMB2 Ext Synch
XBAR_OUT56	PWMB3_EXT_SYNC	PWMB3 Ext Synch
XBAR_OUT57	PWMB_FORCE	PWMB Force
XBAR_OUT58	EWM_IN	External Watchdog Monitor



3.3.3.2 XBARA and XBARB Inputs

The following table shows the signals that can be inputs to the XBARs.

Table 3-6. XBARA and XBARB Inputs

Package Signal	Signal Description	XBARA Input	XBARB Input
VSS	VSS	XBAR_IN0	—
VDD	VDD	XBAR_IN1	—
XB_IN2	Package Pin	XBAR_IN2	XBAR_IN14
XB_IN3	Package Pin	XBAR_IN3	XBAR_IN15
XB_IN4	Package Pin	XBAR_IN4	—
XB_IN5	Package Pin	XBAR_IN5	—
XB_IN6	Package Pin	XBAR_IN6	—
XB_IN7	Package Pin	XBAR_IN7	—
XB_IN8	Package Pin	XBAR_IN8	—
XB_IN9	Package Pin	XBAR_IN9	—
XB_IN10	Package Pin	XBAR_IN10	XBAR_IN20
XB_IN11	Package Pin	XBAR_IN11	XBAR_IN21
CMPA_OUT	Comparator A Output	XBAR_IN12	XBAR_IN0
CMPB_OUT	Comparator B Output	XBAR_IN13	XBAR_IN1
CMPC_OUT	Comparator C Output	XBAR_IN14	XBAR_IN2
CMPD_OUT	Comparator D Output	XBAR_IN15	XBAR_IN3
TB0_OUT	TimerB 0	XBAR_IN16	XBAR_IN4
TB1_OUT	TimerB 1	XBAR_IN17	XBAR_IN5
TB2_OUT	TimerB 2	XBAR_IN18	XBAR_IN6
TB3_OUT	TimerB 3	XBAR_IN19	XBAR_IN7
PWMA0_TRG0	PWMA0 Trigger 0	XBAR_IN20	XBAR_IN8
PWMA0_TRG1	PWMA0 Trigger 1	XBAR_IN21	(PWMA0_TRG0 | PWMA0_TRG1)
PWMA1_TRG0	PWMA1 Trigger 0	XBAR_IN22	XBAR_IN9
PWMA1_TRG1	PWMA1 Trigger 1	XBAR_IN23	(PWMA1_TRG0 | PWMA1_TRG1)
PWMA2_TRG0	PWMA2 Trigger 0	XBAR_IN24	XBAR_IN10
PWMA2_TRG1	PWMA2 Trigger 1	XBAR_IN25	(PWMA2_TRG0 | PWMA2_TRG1)

The PWMB0_EXT_SYNC input signal is the synchronization signal PWM_EXT_SYNC for SM0 of eFlexPWMB module, which can reset PWMB sub-module0 counter, The PWMB1_EXT_SYNC input signal can reset the PWMB sub-module1,….

The MC56F84789 has two eFlexPWM modules: eFlexPWMA and eFlexPWMB, the two module are independent. Assume that you want to use eFlexPWMA to synchronize eFlexPWMB, for eFlexPWMA module itself, you can use master sync to synchronize the other sub-modules of eFlexPWMA module(SM1/SM2/SM3 of eFlexPWM). The eFlexPWMA module can generate the master sync signal to reset all the sub-modules of eFlexPWMB(SM0/SM1/SM2/SM3 of eFlexPWMB). The master sync signal can be from PWMA0_TRG1 if you set the bit1 of PWMA_SM0TCTRL register. The master sync means that when the PWM counter matches with the PWMA_SM0VAL1, the signal will be generated.

In detail, this is the register configuration:

	INIT_SEL bits:
PWMA_SM0CTRL2:	3b'000
PWMA_SM1CTRL2:	3b'010
PWMA_SM2CTRL2:	3b'010
PWMB_SM0CTRL2:	3b'011
PWMB_SM1CTRL2:	3b'011
PWMB_SM2CTRL2:	3b'011

That the INIT_SEL bits is set as 3b'000 for PWMA_SM0CTRL2 means that SM0 of PWMA reset itself when the PWM counter matches with PWMA_SM0_VAL1.

That the INIT_SEL bits is set as 3b'010 for PWMA_SM1CTRL2 and PWMA_SM2CTRL2 means that synchronization signal from SM0 of PWMA reset counter of PWMA_SM1 and PWMA_SM2 when the PWM counter matches with PWMA_SM0_VAL1.

That the INIT_SEL bits is set as 3b'011 for PWMB_SM0CTRL2, PWMB_SM1CTRL2 and PWMA_SM2CTRL2 means that synchronization signal from external source reset counter of PWMB_SM0 ,PWMB_SM1 and of PWMB_SM2.

For inter-processor eFlexPWM synchronization, this is hardware connection. Firstly, output the PWMA_TRIG1 signal to an output pad via crossbar for one processor, for example anyone of XBAR_OUT4 to XBAR_OUT11, connect the signal to anyone of XB_IN2 to XB_IN11 for another processor via wire, it is okay.

2. crossbar configuration.

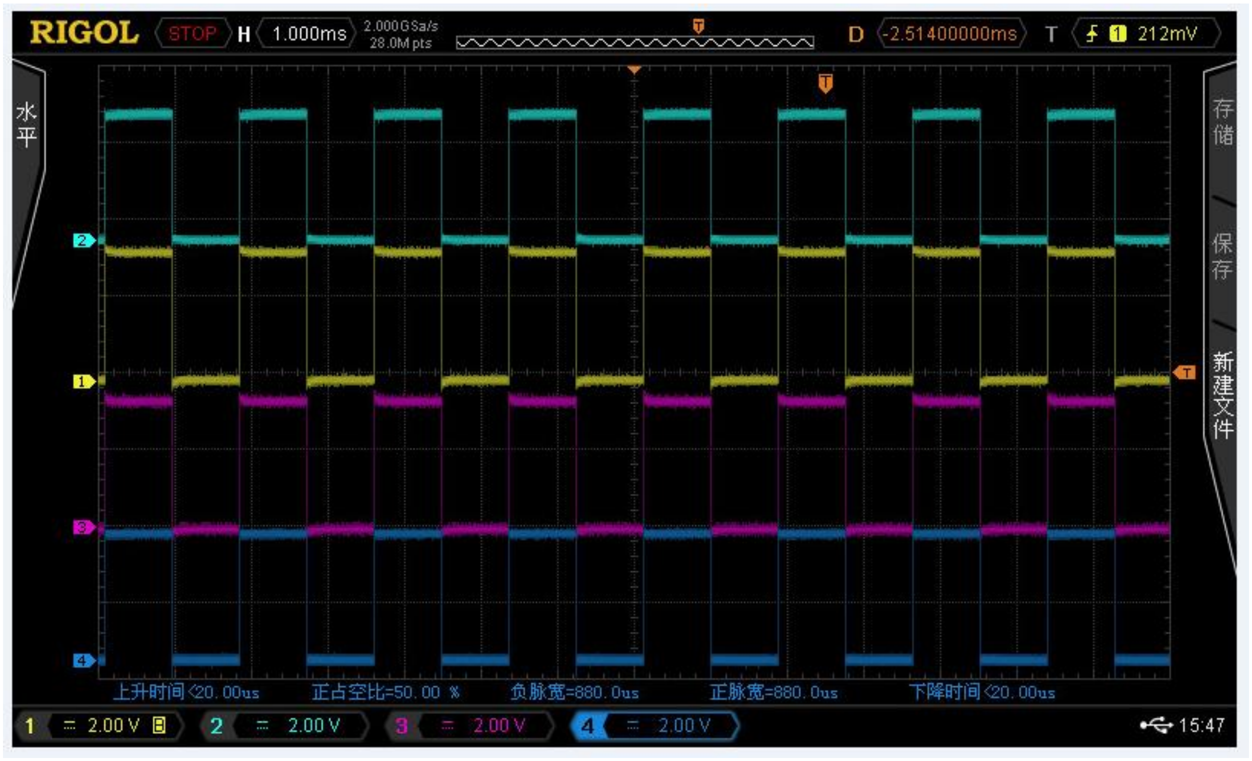
The PWMA0_TRIG1 signal is routed to PWMB0_EXT_SYNC, PWMB1_EXT_SYNC, PWMB2_EXT_SYNC via crossbar, this is the crossbar configuration:

```
XBARA_SEL26|=21<<8; //XBAR_OUT53

XBARA_SEL27|=21; //XBAR_OUT54

XBARA_SEL27|=21<<8; //XBAR_OUT55
```

Waveform of PWMA and PWMB:



In the above waveform of oscilloscope, the channel1 is PWMA_0A(GPIOE1 pin), the channel2 is PWMA_1A(GPIOE3), the channel3 is PWMB_0A(GPIOG3), the channel4 is PWMB_1A(GPIOG1 pin).

From the waveform, we can see that all the PWMA and PWMB signals are synchronized.

Conclusion:

The documentation give a solution how to synchronize two independent eFlexPWM modules and give corresponding code. The eFlexPWMA module uses master sync signal to synchronize SM1, SM2, SM3 of eFlexPWMA. The SM0 of eFlexPWMA generate PWMA0_TRIG signal, the signal is used to trigger all sub module of eFlexPWMB via crossbar. With test, the solution is feasible.

Appendix:

```
void PWMB_init(void)
{
	//SM0 initialization

	PWMB_SM0INIT=0xF000;
	PWMB_SM0VAL0=0x0000;
	PWMB_SM0VAL1=0x1000;		//the modulo is 256
	PWMB_SM0VAL2=0xF800;	//75% duty cycle
	PWMB_SM0VAL3=0x800;
	PWMB_SM0VAL4=0xF800;	//25% duty cycle
	PWMB_SM0VAL5=0x800;
	PWMB_SM0CTRL2=0x300; //complementary mode for PWM0A and PWM0B, IP bus
clock,the INIT_SEL should be 11, which means
```

```
	//that the external signal will synchronize the module
	PWMB_SM0CTRL=0x3400; //4 PWM opportunity, PWM clock=Fclk
	PWMB_SM0OCTRL=0x0000; //PWM does not inverter, PWM forced to logic 0 in fault
state
	PWMB_SM0TCTRL=0x0000;
	PWMB_SM0INTEN=0x0000; //disable all interrupt
	PWMB_SM0DISMAP0=0x0000; //Disable fault mask
	PWMB_SM0DISMAP1=0x0000; //Disable fault mask
	PWMB_SM0DTCNT0=0x0000; //dead time is set to 0
	PWMB_SM0DTCNT1=0x0000;
	PWMB_SM0CTRL|=0x04;

	//SM1 module initialization

	PWMB_SM1INIT=0xF000;
	PWMB_SM1VAL0=0x0000;
	PWMB_SM1VAL1=0x1000;      //the modulo is 256
	PWMB_SM1VAL2=0xF800;  //75% duty cycle
	PWMB_SM1VAL3=0x800;
	PWMB_SM1VAL4=0xF800;  //25% duty cycle
	PWMB_SM1VAL5=0x800;
	PWMB_SM1CTRL2=0x300; //complementary mode, IP bus clock, the INIT_SEL should
be 11, which means
	//that the external signal will synchronize the module
	PWMB_SM1CTRL=0x3400; //4 PWM opportunity, PWM clock=Fclk
	PWMB_SM1OCTRL=0x0000; //PWM does not inverter, PWM forced to logic 0
	PWMB_SM1TCTRL=0x0000;
	PWMB_SM1INTEN=0x0000;
	PWMB_SM1DISMAP0=0x0000; //Disable fault mask
	PWMB_SM1DISMAP1=0x0000; //Disable fault mask
	PWMB_SM1DTCNT0=0x0000; //dead time is set to 0
	PWMB_SM1DTCNT1=0x0000;
	PWMB_SM1CAPTCTRLX|=0x40; //PWMA1_X output
	PWMB_OUTEN|=0x0FF0;  //enable PWM output
//	PWMA_FCTRL)=0xF000; //fault logic setting
	PWMB_SM1CTRL|=0x04;

		//test PWM2_X output signal
		//when the SM2 module of PWMA is synchronized by SM0 sync signal, the
PWM1_X/PWM2_X/PWM3_X can output arbitary
		//waveform without any restriction because of PWMA_SMxVAL1 register,
only PWMA_SM0VAL1 register is used to control the
		//PWMA frequency
		PWMB_SM2INIT=0xF000;
		PWMB_SM2VAL0=0x0000;
		PWMB_SM2VAL1=0x1000;      //the modulo is 256
		PWMB_SM2VAL2=0xF800;  //75% duty cycle
		PWMB_SM2VAL3=0x800;
		PWMB_SM2VAL4=0xF800;  //25% duty cycle
		PWMB_SM2VAL5=0x800;
		PWMB_SM2CTRL2=0x300; //complementary mode, IP bus clock, the INIT_SEL
should be 11, which means
			//that the external signal will synchronize the module
		PWMB_SM2CTRL=0x3400; //4 PWM opportunity, PWM clock=Fclk
		PWMB_SM2OCTRL=0x0000; //PWM does not inverter, PWM forced to logic 0
		PWMB_SM2TCTRL=0x0000;
		PWMB_SM2INTEN=0x0000;
		PWMB_SM2DISMAP0=0x0000; //Disable fault mask
		PWMB_SM2DISMAP1=0x0000; //Disable fault mask
		PWMB_SM2DTCNT0=0x0000; //dead time is set to 0
		PWMB_SM2DTCNT1=0x0000;
		PWMB_SM2CTRL|=0x04;
		//PWMA global register setting
		PWMB_OUTEN|=0x0FF0;  //enable PWM output
		PWMB_MASK=0x0000;  //disable PWM mask
		PWMB_SWCOUT=0x0000;  //determine dead time logic
		//PWMA_FCTRL)=0xF000;     //fault logic setting
		PWMB_MCTRL|=0x0007; //must use the instruction, otherwise, the counter
will disorder, IPOL is cleared, PWM23 manipulate the duty cycle

	return;
}

void PWMA_init(void)
{
	//SM0 initialization

	PWMA_SM0INIT=0xF000;
	PWMA_SM0VAL0=0x0000;
	PWMA_SM0VAL1=0x1000;      //the modulo is 256
	PWMA_SM0VAL2=0xF800;  //75% duty cycle
	PWMA_SM0VAL3=0x800;
	PWMA_SM0VAL4=0xF800;  //25% duty cycle
	PWMA_SM0VAL5=0x800;
	PWMA_SM0CTRL2=0x0000; //complementary mode for PWM0A and PWM0B, IP bus clock
	PWMA_SM0CTRL=0x3400; //4 PWM opportunity, PWM clock=Fclk
	PWMA_SM0OCTRL=0x0000; //PWM does not inverter, PWM forced to logic 0 in fault
state
	PWMA_SM0TCTRL=0x0002; //generate PWMA0_TRIG1 signal, it will be routed to
trigger PWMB0_EXT_SYNC, PWMB1_EXT_SYNC, PWMB2_EXT_SYNC
	PWMA_SM0INTEN=0x0000; //disable all interrupt
	PWMA_SM0DISMAP0=0x0000; //Disable fault mask
	PWMA_SM0DISMAP1=0x0000; //Disable fault mask
	PWMA_SM0DTCNT0=0x0000; //dead time is set to 0
	PWMA_SM0DTCNT1=0x0000;
	PWMA_SM0CTRL|=0x04;

	//SM1 module initialization

	PWMA_SM1INIT=0xF000;
	PWMA_SM1VAL0=0x0000;
	PWMA_SM1VAL1=0x1000;      //the modulo is 256
	PWMA_SM1VAL2=0xF800;  //75% duty cycle
	PWMA_SM1VAL3=0x800;
	PWMA_SM1VAL4=0xF800;  //25% duty cycle
	PWMA_SM1VAL5=0x800;
	PWMA_SM1CTRL2=0x200; //complementary mode, IP bus clock, the INIT_SEL should
be 10, which means
	//that the SM0 synchronize thw SM1, PWMX_INIT is set as a test
	PWMA_SM1CTRL=0x3400; //4 PWM opportunity, PWM clock=Fclk
	PWMA_SM1OCTRL=0x0000; //PWM does not inverter, PWM forced to logic 0
	PWMA_SM1TCTRL=0x0000;
	PWMA_SM1INTEN=0x0000;
	PWMA_SM1DISMAP0=0x0000; //Disable fault mask
	PWMA_SM1DISMAP1=0x0000; //Disable fault mask
	PWMA_SM1DTCNT0=0x0000; //dead time is set to 0
	PWMA_SM1DTCNT1=0x0000;
	PWMA_SM1CAPTCTRLX|=0x40; //PWMA1_X output
	PWMA_OUTEN|=0x0FF0;  //enable PWM output
//	PWMA_FCTRL)=0xF000; //fault logic setting
	PWMA_SM1CTRL|=0x04;

		//test PWM2_X output signal
		//when the SM2 module of PWMA is synchronized by SM0 sync signal, the
PWM1_X/PWM2_X/PWM3_X can output arbitary
		//waveform without any restriction because of PWMA_SMxVAL1 register,
only PWMA_SM0VAL1 register is used to control the
		//PWMA frequency
		PWMA_SM2INIT=0xF000;
		PWMA_SM2VAL0=0x0000;
		PWMA_SM2VAL1=0x1000;	    //the modulo is 256
		PWMA_SM2VAL2=0xF800;  //75% duty cycle
		PWMA_SM2VAL3=0x800;
		PWMA_SM2VAL4=0xF800;  //25% duty cycle
		PWMA_SM2VAL5=0x800;
		PWMA_SM2CTRL2=0x200; //complementary mode, IP bus clock, the INIT_SEL
should be 10, which means
			//that the SM0 synchronize the SM2 module
		PWMA_SM2CTRL=0x3400; //4 PWM opportunity, PWM clock=Fclk
		PWMA_SM2OCTRL=0x0000; //PWM does not inverter, PWM forced to logic 0
		PWMA_SM2TCTRL=0x0000;
		PWMA_SM2INTEN=0x0000;
		PWMA_SM2DISMAP0=0x0000; //Disable fault mask
		PWMA_SM2DISMAP1=0x0000; //Disable fault mask
		PWMA_SM2DTCNT0=0x0000; //dead time is set to 0
		PWMA_SM2DTCNT1=0x0000;
		PWMA_SM2CTRL|=0x04;
		//PWMA global register setting
		PWMA_OUTEN|=0x0FF0;  //enable PWM output
		PWMA_MASK=0x0000;  //disable PWM mask
		PWMA_SWCOUT=0x0000;  //determine dead time logic
		//	PWMA_FCTRL)=0xF000; //fault logic setting
		PWMA_MCTRL|=0x0007; //must use the instruction, otherwise, the counter
will disorder, IPOL is cleared, PWM23 manipulate the duty cycle

	return;
}

void FORCE_ENABLE(void)
{
	PWMB_SM0CTRL2|=0x80; //FORCE enable is set
	PWMB_SM0CTRL2&=0xFFC7;
	//set PWMA_SM0 in independent mode
	PWMB_SM0CTRL2|=0x2000; //set PWM in independent mode
	PWMB_DTSRCSEL|=0x0A;
```

```
}

void corssbarSetting(void)
{
	//set crossbar so that the PWMA_SM0 master sync can synchronize the PWMB sub 
module
	XBARA_SEL26|=21<<8; //XBAR_OUT53
	XBARA_SEL27|=21; //XBAR_OUT54
	XBARA_SEL27|=21<<8; //XBAR_OUT55

}


void CLOCK_init(void)
{
	SIM_PCE3|=0x00FF;  //enable PWMA and PWMB all channels
	SIM_PCE0|=0xFF7F;  //enable all Timer and GPIO A/B/C/D/E/G/F
	SIM_PCE2|=0x180; //enable SAR ADC clock and Cyclic ADC clock
	return;
}


int main(void)
{

	DisableWatchdog;
	CLOCK_init();
	GPIOEG_init();
	PWMB_init();
	delay();
	PWMA_init();
	corssbarSetting();
	//PWM_ISR_SETTING();
	PWMB_MCTRL|=0x0100;  //enable the PWM module
	PWMB_MCTRL|=0x0200;  //enable the PWM module
	PWMB_MCTRL|=0x0400;  //enable the PWM module
	PWMB_MCTRL|=0x0800;  //enable the PWM module

	PWMA_MCTRL|=0x0100;  //enable the PWM module
	PWMA_MCTRL|=0x0200;  //enable the PWM module
	PWMA_MCTRL|=0x0400;  //enable the PWM module
	PWMA_MCTRL|=0x0800;  //enable the PWM module
	//asm(bfclr #300,sr); //enable interrupt
	for(;;)
	{}
	return(0);

}
```