Route the PWM signals to external pad via crossbar

For the MC56F83xxx, it has 2 eFlexPWM modules PWMA and PWMB, each PWM can output at least 8 PWM signals, the two PWM modules can output at least 16 PWM signals.

The MC56F83xxx has three packages, LQFP100, LQFP80 and LQFP64, for small reduced pin package LQFP80 and LQFP64, maybe some PWM signals can not output to external pads because of reduced pin number, in the case, user can output the PWM signal via crossbar.

For example, the following screenshot is the pin assignment for MC56F83xxx, a simple code is developed and tested, it demos that the PWMA_0A/0B are outputted from the XB_OUT8 and XB_OUT9 pins

Pinout

100 LQFP	80 LQFP	64 LQFP	Pin Name	Default	ALT0	ALT1	ALT2	ALT3
53	42	34	GPIOC9	GPIOC9	SCLK0	XB_IN4	TXD0	XB_OUT8
54	43	35	GPIOC10	GPIOC10	MOSI0	XB_IN5	MISO0	XB_OUT9
55	44	36	GPIOF0	GPIOF0	XB_IN6	TB2	SCLK1	
56	45	—	GPIOF10	GPIOF10	TXD2	PWMA_FAULT6	PWMB_FAULT6	XB_OUT10
57	46	—	GPIOF9	GPIOF9	RXD2	PWMA_FAULT7	PWMB_FAULT7	XB_OUT11
58	47	37	GPIOC11	GPIOC11	CANTX	SCL1	TXD1	
59	48	38	GPIOC12	GPIOC12	CANRX	SDA1	RXD1	
60	49	39	GPIOF2	GPIOF2	SCL1	XB_OUT6	MISO1	
61	50	40	GPIOF3	GPIOF3	SDA1	XB_OUT7	MOSI1	
62	51	41	GPIOF4	GPIOF4	TXD1	XB_OUT8	PWMA_0X	PWMA_FAULT6
63	52	42	GPIOF5	GPIOF5	RXD1	XB_OUT9	PWMA_1X	PWMA_FAULT7
64	—	—	GPIOG8	GPIOG8	PWMB_0X	PWMA_0X	TA2	XB_OUT10
65	—	—	GPIOG9	GPIOG9	PWMB_1X	PWMA_1X	TA3	XB_OUT11

Because I have only MC56F83789-EVK board, the example is based on the MC56F83789-EVK board and CodeWarrior for MCU ver11.1 without QuickStart or SDK.

This is the procedure:

1) configure the external pads as XBAR_OUTx

The example use XBAR_OUT8 and XBAR_OUT9, the XBAR_OUT8 is multiplexed with GPIOG6, the XBAR_OUT9 is multiplexed with GPIOG7, the GPIOG6 and GPIOG7 are connected to external pad by hardware connection.
So the GPIOG port gated clock must be enabled so that the GPIOG_PER register can be written.

```
SIM->PCE0|=0x01; //enable  GPIOG port clock

Write the GPIOG_PER so that the pins can be dominated by
peripheral rather than GPIO
GPIOG->PER=0xC0; //GPIOG6/7 are dominated by peripherals
```

Write the SIM->GPSGL to select the XB_OUT8 and XB_OUT9 function

```
SIM->GPSGL&=~(0xF000);
SIM->GPSGL|=0xB000;
```

2) set up crossbar so that the PWM signal can be outputted from XB_OUT8 and XB_OUT9

The XB_OUT8 corresponds to bit0~5, the XB_OUT9 corresponds to bit8~13 of XBARA->SEL4.

Table 3-5. XBARA Inputs (continued)

Package Signal	Signal Description	XBARA Input
PWMA0_MUX_TRIG0/ PWMB0_OUT_TRIG0 [1]	PWMA or PWMB	XBAR_IN20
PWMA0_MUX_TRIG1/ PWMB0_OUT_TRIG1	PWMA or PWMB	XBAR_IN21

3.3.3.3 XBARA Outputs

Table 3-6. XBARA Outputs

XBARA Output	Signal	Signal Description
XBAR_OUT0	DMA_REQ0	XBAR DMA Request 0
XBAR_OUT1	DMA_REQ1	XBAR DMA Request 1
XBAR_OUT2	DMA_REQ2	XBAR DMA Request 2
XBAR_OUT3	DMA_REQ3	XBAR DMA Request 3
XBAR_OUT4	XB_OUT4	Package Pin
XBAR_OUT5	XB_OUT5	Package Pin
XBAR_OUT6	XB_OUT6	Package Pin
XBAR_OUT7	XB_OUT7	Package Pin
XBAR_OUT8	XB_OUT8	Package Pin
XBAR_OUT9	XB_OUT9	Package Pin
XBAR_OUT10	XB_OUT10	Package Pin
XBAR_OUT11	XB_OUT11	Package Pin

So the code is

```
XBARA->SEL4=(21<<8)|(20);
```

The PWMA0_MUX_TRIG0 signal is PWMA_0A signal, the PWMA0_MUX_TRIG1 is PWMA_0B, in other words, the PWMA_0A signal is A signal for sub-module0 of PWMA module. The PWMA_0B signal is B signal for sub-module0 of PWMA module.

From above table 3-5, the XBAR_IN20 is multiplexed with PWMA0_MUX_TRIG0 and PWMB0_OUT_TRIG0, The XBAR_IN20 is multiplexed with PWMA0_MUX_TRIG1 and PWMB0_OUT_TRIG1. So SIM->PWM_SEL register is used to select the one of the two signals.

SIM_PWM_SEL field descriptions (continued)

Field	Description
10 XBAR_IN30	xbar input 30 0 PWMB1_MUX_TRIG0 1 PWMA1_OUT_TRIG0
9 XBAR_IN29	xbar input 29 0 PWMB0_MUX_TRIG1 1 PWMA0_OUT_TRIG1
8 XBAR_IN28	xbar input 28 0 PWMB0_MUX_TRIG0 1 PWMA0_OUT_TRIG0
7 XBAR_IN27	xbar input 27 0 PWMA3_MUX_TRIG1 1 PWMB3_OUT_TRIG1
6 XBAR_IN26	xbar input 26 0 PWMA3_MUX_TRIG0 1 PWMB3_OUT_TRIG0
5 XBAR_IN25	xbar input 25 0 PWMA2_MUX_TRIG1 1 PWMB2_OUT_TRIG1
4 XBAR_IN24	xbar input 24 0 PWMA2_MUX_TRIG0 1 PWMB2_OUT_TRIG0
3 XBAR_IN23	xbar input 23 0 PWMA1_MUX_TRIG1 1 PWMB1_OUT_TRIG1
2 XBAR_IN22	xbar input 22 0 PWMA1_MUX_TRIG0 1 PWMB1_OUT_TRIG0
1 XBAR_IN21	xbar input 21 0 PWMA0_MUX_TRIG1 1 PWMB0_OUT_TRIG1
0 XBAR_IN20	xbar input 20 0 PWMA0_MUX_TRIG0 1 PWMB0_OUT_TRIG0

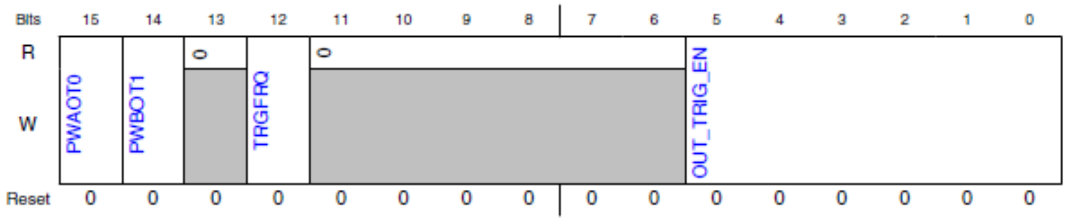
```
SIM->PWM_SEL&=~0x03; //clear XBAR_IN20 and XBAR_IN21 bits
```

3)configure the SMxTCTRL register so that the PWMA_0A and PWMA_0B can be selected.

In detail, setting the PWAOT0 bit of PWM->SMxTCTRL, the PWMA_0A will be routed to PWMA0_MUX_TRIG0, setting the PWBOT1 bit of PWM->SMxTCTRL, the PWMA_0B will be routed to PWMA0_MUX_TRIG1.

```
PWMA->SM0TCTRL=0xC000;
```

27.3.22.2 Diagram



27.3.22.3 Fields

Field	Function
15 PWAOT0	Mux Output Trigger 0 Source Select This bit selects which signal to bring out on the PWM's PWM_MUX_TRIG0 port. 0b - Route the PWM_OUT_TRIG0 signal to PWM_MUX_TRIG0 port. 1b - Route the PWMA output to the PWM_MUX_TRIG0 port.
14 PWBOT1	Mux Output Trigger 1 Source Select This bit selects which signal to bring out on the PWM's PWM_MUX_TRIG1 port. 0b - Route the PWM_OUT_TRIG1 signal to PWM_MUX_TRIG1 port. 1b - Route the PWMB output to the PWM_MUX_TRIG1 port.
13 —	RESERVED
12 TRGFRQ	Trigger frequency This read/write bit allows control over the frequency of the trigger outputs when using non-zero values of CTRL[LDFQ]. 0b - Trigger outputs are generated during every PWM period even if the PWM is not reloaded every period due to CTRL[LDFQ] being non-zero. 1b - Trigger outputs are generated only during the final PWM period prior to a reload opportunity when the PWM is not reloaded every period due to CTRL[LDFQ] being non-zero.

Table continues on the next page...

Field	Function
11-6 —	RESERVED
5-0 OUT_TRIG_EN	Output Trigger Enables These bits enable the generation of PWM_OUT_TRIG0 and PWM_OUT_TRIG1 outputs based on the counter value matching the value in one or more of the VAL0-5 registers.

Appendix

1)Pin and crossbar configuration

```
//The GPIOG6 multiplexed with XBAR_OUT8
```

```
//GPIOG7 multiplexed with XBAR_OUT9
void PWMAToXB_OUT(void)
{
    SIM->PCE0|=0x01; //enable  GPIOG port clock
    //GPIOG->PER|=0xC0;
    GPIOG->PER=0xC0;
    SIM->GPSGL&=~(0xF000);
    SIM->GPSGL|=0xB000;

    //XBAR_OUT8 pin outputs XB_IN20 signal,XBAR_OUT9 pin outputs
XB_IN21 signal,
    XBARA->SEL4=(21<<8)|(20);
    //XBARA->SEL4=(1<<8)|1;  //for test
    //select PWMA0_MUX_TRIG0 and PWMA0_MUX_TRIG1 outputs
    SIM->PWM_SEL&=~0x03;
    //configure PWMA
    PWMA->SM0TCTRL=0xC000;
    }
```

2)PWM configuration

```
//The PWM works in center-alignment, complementary mode, and 900HZ,
the bus clock is 50mHz
//50MHz/(0.9K*2)=27777, which is 0x6C81, the complementary is 0x937F
//all the sub-modules work independently, INIT_SEL=00

void PWMA_init(void)
{
    //SM0 initialization
    //pin assignment for PWMA signal
    SIM->PCE0|=0x04; //enable GPIOE clock, GPIOE0~GPIOE7 are
PWMA_SM0~SM3
    GPIOE->PER|=0xFF;

    //enable PWMA clock
    SIM->PCE3|=0xF0; //enable all PWMA sub-modules clock
    OCCS->DIVBY|=0x01<<6; //set PWM_DIV2
    SIM->PCR&=~(0x01<<10);
    PWMA->SM0INIT=0x937F;
    PWMA->SM0VAL0=0x0000;
    PWMA->SM0VAL1=0x6C81;
    PWMA->SM0VAL2=0xC9BF;
    PWMA->SM0VAL3=0x3640;
```

```
    PWMA->SM0VAL4=0xC9BF;
    PWMA->SM0VAL5=0x3640;
    PWMA->SM0CTRL2=0x2000; //independent mode for PWM0A and PWM0B, IP
bus clock, local synchronize
    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=0x0000;
    PWMA->SM0TCTRL=0xC000;
    PWMA->SM0INTEN=0x0000; //disable all interrupt
    PWMA->SM0DISMAP0=0x0000; //Disable fault mask
    PWMA->SM0DISMAP1=0x0000; //Disable fault mask
    PWMA->SM0DTCNT0=100; //dead time is set to 100, based on 50Mhz
Bus clock, 2 us is 100
    PWMA->SM0DTCNT1=100;

    //SM1 module initialization
    PWMA->SM1INIT=0x937F;
    PWMA->SM1VAL0=0x0000;
    PWMA->SM1VAL1=0x6C81;
    PWMA->SM1VAL2=0xC9BF;
    PWMA->SM1VAL3=0x3640;
    PWMA->SM1VAL4=0xC9BF;
    PWMA->SM1VAL5=0x3640;
    PWMA->SM1CTRL2=0x2000; //independent mode, IP bus clock, the
INIT->SEL should be 00, local synchronize
    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=100; //dead time is set to 100, dead time is 2 us
    PWMA->SM1DTCNT1=100;

    //SM2 initialization
    PWMA->SM2INIT=0x937F;
    PWMA->SM2VAL0=0x0000;
    PWMA->SM2VAL1=0x6C81;
    PWMA->SM2VAL2=0xC9BF;
    PWMA->SM2VAL3=0x3640;
    PWMA->SM2VAL4=0xC9BF;
    PWMA->SM2VAL5=0x3640;
```

```
    PWMA->SM2CTRL2=0x2000; //independent mode, SM0 Clock, the
INIT->SEL should be 00, local synchronization
    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=100; //dead time is set to 0
    PWMA->SM2DTCNT1=100;

    //sub-module3 initialization
    PWMA->SM3INIT=0x937F;
    PWMA->SM3VAL0=0x0000;
    PWMA->SM3VAL1=0x6C81;
    PWMA->SM3VAL2=0xC9BF;
    PWMA->SM3VAL3=0x3640;
    PWMA->SM3VAL4=0xC9BF;
    PWMA->SM3VAL5=0x3640;
    PWMA->SM3CTRL2=0x2000; //independent mode, IP bus clock, the
INIT->SEL should be 00, local synchronization
    PWMA->SM3CTRL=0x3400; //4 PWM opportunity, PWM clock=Fclk
    PWMA->SM3OCTRL=0x0000; //PWM does not inverter, PWM forced to
logic 0
    PWMA->SM3TCTRL=0x0000;
    PWMA->SM3INTEN=0x0000;
    PWMA->SM3DISMAP0=0x0000; //Disable fault mask
    PWMA->SM3DISMAP1=0x0000; //Disable fault mask
    PWMA->SM3DTCNT0=100; //dead time is set to 0
    PWMA->SM3DTCNT1=100;

    PWMA->MASK=0x0000;  //disable PWM  mask
    PWMA->SWCOUT=0x0000;  //determine dead time logic
    PWMA->MCTRL|=0x000F; //must use the instruction, otherwise, the
counter will disorder, IPOL is cleared, PWM23 manipulate the duty
cycle

    return;
}

void PWMEnable(void)
{
    //PWMA global register setting
```

```
    PWMA->OUTEN|=0x0FF0;  //enable PWM output

    //  PWMA->FCTRL)=0xF000; //fault logic setting
    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
//test hardware button on the tower board
}
```