Route the ACOMP output and CTimer0 Cap0 input to the same pin via SWM

For the LPC8xx family, the peripheral module input or output signals are not connected to external pads fixedly as most processor does. With SWM module, the peripheral module input or output signals can be routed to any external GPIO pins via software configuration.
There is special requirement that one peripheral output signal functions as the input signal of another peripheral, in the case, the peripheral output signal and the peripheral input signal can be routed to the same pad via SWM so that the input and output signals are connected internally without external cable.
The LPC802 has ACOMP module, CTimer module and SWM module, per customer requirement, the ACOMP output signal and CTimer0 capture0 input signal can be routed to the same GPIO pin without external connection so that the ACOMP output signal can trigger CTimer0 capture0 event internally.

1. ACOMP output signal pin and the CTimer0_Cap0 input signal are routed to the same pad PIO0_13 with SWM

With the following code, the CTimer0_capture0 signal is routed to PIO0_13

```
//using PIO0_13 or the pin 21 of CN3 as the CTimer0 Cap0 pin
void CTimer0_cap0_assignment(void)
{
     CLOCK_EnableClock(kCLOCK_Swm);
     //CTimer0_Cap0 pin is routed to PIO0_13 as input
      SWM_SetMovablePinSelect(SWM0, kSWM_T0_CAP_CHN0, kSWM_PortPin_P0_13);

}
```

With the following code, the ACOMP output signal is routed to PIO0_13

```
//The PIO0_16 (pin 15 of CN3 on LPC802-EVK board) function as CMP_I4
void AcompPinAssignment(void)
{
     CLOCK_EnableClock(kCLOCK_Swm);
     //set the PIOI0_16 pin as ACMP_I4
     SWM_SetFixedPinSelect(SWM0, kSWM_ACMP_INPUT4,true);
          //ACOMP output is routed to PIO0_13
     SWM_SetMovablePinSelect(SWM0, kSWM_ACMP_OUT, kSWM_PortPin_P0_13);

}
```

The project is developed under MCUXpresso IDE and SDK package, run on the LPCXpresso802 board

In the **ctimer_cap0_callback() function, a GPIO toggling function is called, which can toggle a LED since the LED is connected to the GPIO.**
The PIO0_16 (pin 15 of CN3 on LPC802-EVK board) functions as ACMP_I4
Result: connect the pin 15 of CN3 on LPC802-EVK board(**PIO0_16) to the VDD or GND via cable, the LED will toggle, it means that the CTimer capture0 event has happened, the CTimer0_capture0 callback is called.**

2. ACOMP configuration

For the ACOMP module, the positive input source is the DAC output which is set up as 1.67V. The negative input source of the ACOMP is the ACMP_I4, which is multiplexed with PIO0_16 (pin 15 of CN3 on LPC802-EVK board).

```
acomp_config_t CMPconfig;
const acomp_ladder_config_t cmp_ladderConfig=
{
        .ladderValue=0x10, //setput the DAC output 1.67V
        .referenceVoltage=kACOMP_LadderRefVoltagePinVDD
};

//The PIO0_16 (pin 15 of CN3 on LPC802-EVK board) function as ACMP_I4
void acompInit(void)
{
    POWER_DisablePD(kPDRUNCFG_PD_ACMP);
    ACOMP_GetDefaultConfig(&CMPconfig);
    ACOMP_Init(ACOMP,&CMPconfig);
    ACOMP_SetLadderConfig(ACOMP,&cmp_ladderConfig);
         /* Configure ACOMP negative and positive input channels. */
    //set the LADDER_OUT as the positive input, the ACMP_I4 as the CMP
negative input
    ACOMP_SetInputChannel(ACOMP, 0, 4);
}

//The PIO0_16 (pin 15 of CN3 on LPC802-EVK board) function as CMP_I4
void AcompPinAssignment(void)
{
     CLOCK_EnableClock(kCLOCK_Swm);
     //set the PIOI0_16 pin as ACMP_I4
     SWM_SetFixedPinSelect(SWM0, kSWM_ACMP_INPUT4,true);
          //ACOMP output is routed to PIO0_13
     SWM_SetMovablePinSelect(SWM0, kSWM_ACMP_OUT, kSWM_PortPin_P0_13);

}
```

3. CTimer configuration

This is the CTimer module configuration, it function as a Timer, once the capture signal is detected, the capture interrupt is triggered, the callBack function is called.
Here, the `ctimer_callback_table[] is highglighted, this is function pointer table, the element corresponds to item in the Interrupt register.`
`In detail, in the ctimer_callback_table[], the element1 corresponds to the callback of match channel0. The element2 corresponds to the callback of match channel1,…, the element4 corresponds to the callback of capture0 event, the element5 corresponds to the callback of capture1 event.`

16.6.1 Interrupt Register

The Interrupt Register consists of 4 bits for the match interrupts and 3 bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

Table 206. Interrupt Register (IR, offset 0x000) bit description

Bit	Symbol	Description	Reset Value
0	MR0INT	Interrupt flag for match channel 0.	0
1	MR1INT	Interrupt flag for match channel 1.	0
2	MR2INT	Interrupt flag for match channel 2.	0
3	MR3INT	Interrupt flag for match channel 3.	0
4	CR0INT	Interrupt flag for capture channel 0 event.	0
5	CR1INT	Interrupt flag for capture channel 1 event.	0
6	CR2INT	Interrupt flag for capture channel 2 event.	0
31:7	-	Reserved. Read value is undefined, only zero should be written.	-

```
void ctimer_match0_callback(uint32_t flags)
{
    //GPIO_PortToggle(GPIO, BOARD_LED_BLUE_GPIO_PORT, 1u <<
BOARD_LED_BLUE_GPIO_PIN);
}

void ctimer_cap0_callback(uint32_t flags)
{
    GPIO_PortToggle(GPIO, BOARD_LED_BLUE_GPIO_PORT, 1u <<
BOARD_LED_BLUE_GPIO_PIN);
}

ctimer_callback_t ctimer_callback_table[] = {
        ctimer_match0_callback, NULL, NULL, NULL, ctimer_cap0_callback,
NULL, NULL, NULL};
ctimer_config_t config;
```

```
void cTimerInit_1(void)
{
     CTIMER_GetDefaultConfig(&config);

        CTIMER_Init(CTIMER0, &config);

        /* Configuration 0 */
        matchConfig0.enableCounterReset = true;
        matchConfig0.enableCounterStop  = false;
        matchConfig0.matchValue         = SystemCoreClock / 2;
        matchConfig0.outControl         = kCTIMER_Output_Toggle;
        matchConfig0.outPinInitState    = false;
        //matchConfig0.enableInterrupt    = true;
        matchConfig0.enableInterrupt    = false;
        CTIMER_RegisterCallBack(CTIMER0, &ctimer_callback_table[0],
kCTIMER_MultipleCallback);
        CTIMER_SetupMatch(CTIMER0, kCTIMER_Match_0, &matchConfig0);

CTIMER_SetupCapture(CTIMER0,kCTIMER_Capture_0,kCTIMER_Capture_RiseEdge,true
);
        CTIMER_StartTimer(CTIMER0);
}

void LED_Toggle(void)
{
    GPIO_PortToggle(GPIO, BOARD_LED_BLUE_GPIO_PORT, 1u <<
BOARD_LED_BLUE_GPIO_PIN);
}

//using PIO0_13 or the pin 21 of CN3 as the CTimer0 Cap0 pin
void CTimer0_cap0_assignment(void)
{
     CLOCK_EnableClock(kCLOCK_Swm);
     //CTimer0_Cap0 pin is routed to PIO0_13 as input
      SWM_SetMovablePinSelect(SWM0, kSWM_T0_CAP_CHN0, kSWM_PortPin_P0_13);

}
```

4. Main function

```
//Purpose: demo it is okay to route the ACMP output signal and CTimer0_cap0
input signal to the same pin
//via SWM without any external connection.
//LPCXpresso802 board version A and MCUXpresso ver11.x tools.
//Result: connect the ACMP_I4 pin to GND or VDD, the CTimer0 capture ISR
can be entered.

 //I use the on-chip DAC output as the ACMP positive input, use
ACMP_I4(PIO0_16pin, 15 of CN3 on LPCXpresso802 board)
//as the ACMP negative input. I route the CMP_OUT and CTimer0_CAP0 to the
same pin PIO0_13,
//I connect the ACMP_IN4 to VDD or GND, I can see that CTimer0 capture
event can be triggered.
/**
 * @file    LPC802_Project.c
 * @brief   Application entry point.
 */
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC802.h"
#include "fsl_debug_console.h"
#include "fsl_ctimer.h"
#include "fsl_acomp.h"
#include "fsl_swm.h"
#include "fsl_acomp.h"
#include "fsl_power.h"

#define DEMO_ACOMP_POSITIVE_INPUT 0U /* Voltage ladder output. */
#define DEMO_ACOMP_NEGATIVE_INPUT 4U /* ACMP_I4. *

/* TODO: insert other include files here. */
void cTimerInit_1(void);
void LEDInit(void);
void CTimer0_cap0_assignment(void);
void acompInit(void);
void ctimer_cap0_callback(uint32_t flags);
void AcompPinAssignment(void);
/* TODO: insert other definitions and declarations here. */
static ctimer_match_config_t matchConfig0;
/*
 * @brief   Application entry point.
```

```
 */
int main(void) {

    /* Init board hardware. */
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
    /* Init FSL debug console. */
    BOARD_InitDebugConsole();
#endif
    BOARD_InitLEDsPins();
    PRINTF("Hello World\n");
    cTimerInit_1();
    CTimer0_cap0_assignment();
    AcompPinAssignment();
    acompInit();
    /* Force the counter to be placed into memory. */
    volatile static int i = 0 ;
    /* Enter an infinite loop, just incrementing a counter. */
    while(1) {
        i++ ;
        /* 'Dummy' NOP to allow source level single stepping of
            tight while() loop */
        __asm volatile ("nop");
    }
    return 0 ;
}

void BOARD_InitLEDsPins(void)
{
    /* Enables clock for IOCON.: enable */
    CLOCK_EnableClock(kCLOCK_Iocon);
    /* Enables the clock for the GPIO0 module */
    CLOCK_EnableClock(kCLOCK_Gpio0);

    gpio_pin_config_t LED_BLUE_config = {
        .pinDirection = kGPIO_DigitalOutput,
        .outputLogic = 1U,
    };
    /* Initialize GPIO functionality on pin PIO0_8 (pin 14)  */
    GPIO_PinInit(BOARD_INITLEDSPINS_LED_BLUE_GPIO,
BOARD_INITLEDSPINS_LED_BLUE_PORT, BOARD_INITLEDSPINS_LED_BLUE_PIN,
&LED_BLUE_config);
```

```
    gpio_pin_config_t LED_RED_config = {
        .pinDirection = kGPIO_DigitalOutput,
        .outputLogic = 1U,
    };
    /* Initialize GPIO functionality on pin PIO0_9 (pin 13)  */
    GPIO_PinInit(BOARD_INITLEDSPINS_LED_RED_GPIO,
BOARD_INITLEDSPINS_LED_RED_PORT, BOARD_INITLEDSPINS_LED_RED_PIN,
&LED_RED_config);

    gpio_pin_config_t LED_GREEN_config = {
        .pinDirection = kGPIO_DigitalOutput,
        .outputLogic = 1U,
    };
    /* Initialize GPIO functionality on pin PIO0_12 (pin 4)  */
    GPIO_PinInit(BOARD_INITLEDSPINS_LED_GREEN_GPIO,
BOARD_INITLEDSPINS_LED_GREEN_PORT, BOARD_INITLEDSPINS_LED_GREEN_PIN,
&LED_GREEN_config);

    const uint32_t LED_GREEN = (/* No addition pin function */
                                IOCON_PIO_MODE_INACT |
                                /* Enable hysteresis */
                                IOCON_PIO_HYS_EN |
                                /* Input not invert */
                                IOCON_PIO_INV_DI |
                                /* Disables Open-drain function */
                                IOCON_PIO_OD_DI);
    /* PIO0 PIN12 (coords: 4) is configured as GPIO, PIO0, 12. */
    IOCON_PinMuxSet(IOCON, IOCON_INDEX_PIO0_12, LED_GREEN);

    const uint32_t LED_BLUE = (/* No addition pin function */
                               IOCON_PIO_MODE_INACT |
                               /* Enable hysteresis */
                               IOCON_PIO_HYS_EN |
                               /* Input not invert */
                               IOCON_PIO_INV_DI |
                               /* Disables Open-drain function */
                               IOCON_PIO_OD_DI);
    /* PIO0 PIN8 (coords: 14) is configured as GPIO, PIO0, 8. */
    IOCON_PinMuxSet(IOCON, IOCON_INDEX_PIO0_8, LED_BLUE);

    const uint32_t LED_RED = (/* No addition pin function */
                              IOCON_PIO_MODE_INACT |
                              /* Enable hysteresis */
```

```
                            IOCON_PIO_HYS_EN |
                            /* Input not invert */
                            IOCON_PIO_INV_DI |
                            /* Disables Open-drain function */
                            IOCON_PIO_OD_DI);
    /* PIO0 PIN9 (coords: 13) is configured as GPIO, PIO0, 9. */
    IOCON_PinMuxSet(IOCON, IOCON_INDEX_PIO0_9, LED_RED);
}
```