TX03 Peripheral Driver Usage Example (TMPM3U0FSDMG)

TOSHIBA ELECTRONIC DEVICE SOLUTIONS CORPORATION

RESTRICTIONS ON PRODUCT USE

- DO NOT USE THIS SOFTWARE WITHOUT THE SOFTWARE LISENCE AGREEMENT.

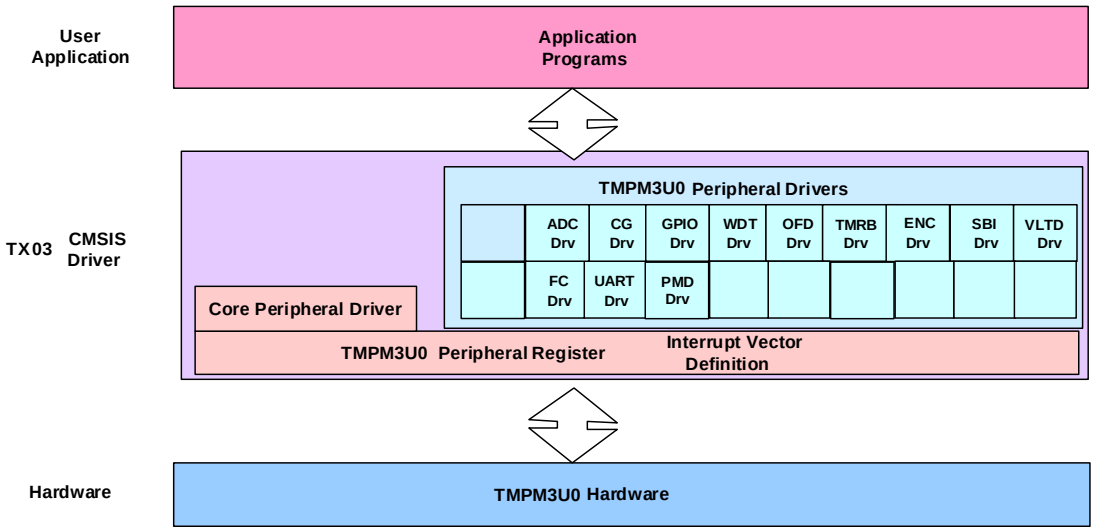
Index

1 General description

The example programs described hereafter are specially designed for TOSHIBA TMPM3U0FSDMG MCU. The programs can be executed individually each to show the main MCU functions. Users can utilize some portions of the programs and integrate them into users' own programs to perform customized functions.

2 Overview

User application utilizes TX03 peripheral driver as following.



3 Build-in hardware usage

Hardware	Channel	Use presence, use
CG	Clock Gear	Used for CG demo
	PLL	PLL on (4x)
Standby mode	-	Used for CG demo (STOP mode)
SysTick	-	Unused
Watch dog timer (WDT)	-	Used for WDT
External interrupt (INT)	INT6	Unused
	INT7	Used for CG demo to wake up system from stop mode
	INTC	Unused

Hardware	Channel	Use presence, use
SIO	SIO0	Used for UART retarget demo , UART_FIFO_Demo, SIO demo
	SIO1	Used for UART_FIFO_Demo
16-bit timer	TMRB0	Used for TMRB: General Timer
	TMRB4	Unused
	TMRB5	Unused
	TMRB7	Used for TMRB: PPG Output
OFD	-	Used for OFD demo
POR	-	Unused
VLTD	-	Used for VLTD demo
12-bit A/D converter	AIN9	Used as ADC data read
	AIN10	Unused
	AIN11	Unused
	AIN12	Unused
SBI	SBI	Used for SBI demo
Encoder input function	ENC0	Used for ENC demo
Programmable Motor Driver	PMD1	Used for PMD Phase Output demo

4 Pin Usage

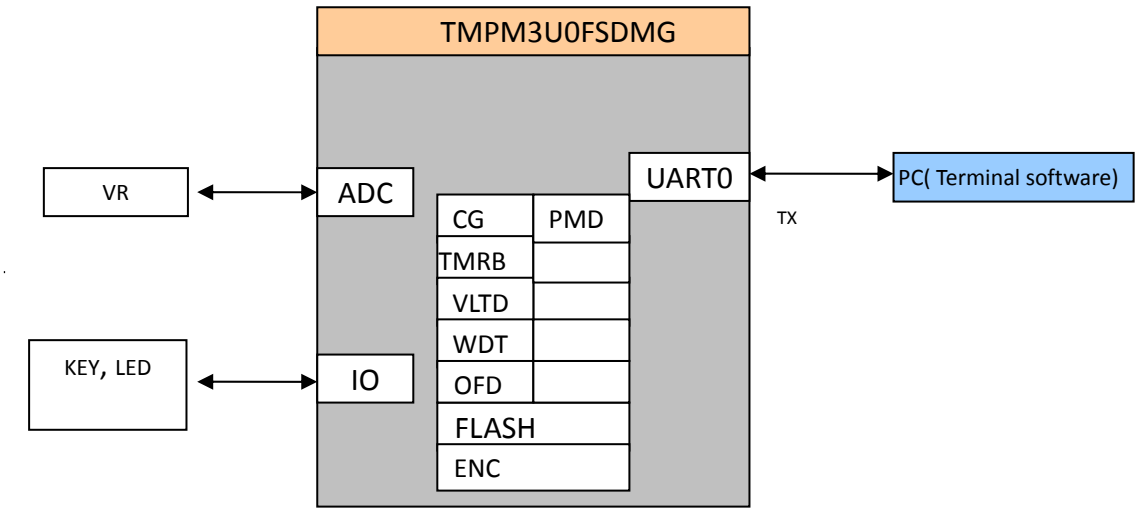
The example programs are tested on TMPM3U0FSDMG evaluation board (TX03-BASEG-A + M3U0FSDMG-XFRM-A + M3U0FSDMG)

Following is pin usage for example programs

No	Name	Usage
1	AVSSB/VREFLB/AMPVSS	Supplying the AD converter with a reference power supply
2	AINB9/PJ6	AINB9 input
3	AINB10/PJ7	SW1/5
4	AINB11/PK0	SW2/6
5	AINB12/PK1	SW3/7
6	AVDD5B/VREFHB/AMPVDD5	Supplying the AD converter with a reference power supply
7	MODE	Unused
8	U0/PG0	LED0/ PMD UO output
9	X0/PG1	LED1/ PMD XO output
10	V0/PG2	LED2/ PMD VO output
11	Y0/PG3	LED3/ PMD YO output
12	W0/PG4	PMD WO output
13	Z0/PG5	PMD ZO output
14	EMG/OVV/PG6	PMD EMG input
15	RESET	Reset input
16	PB3/TMS/SWDIO/(RXD1)	Unused
17	PB4/TCK/SWCLK/(TXD1)	Unused
18	PB5TD0/SWV/SCK0/(SDA0)	SBI Tx/Rx / LED4 for PMD demo
19	PB6/TB7OUT/TDI/SCL0/SI0/RXD1/ INT6	TMRB PPG out / SBI SCL / UART1Rx
20	PF0/TB7IN/BOOT/SDA0/SO0/TXD1/ INTC	UART1Tx
21	PE2/ENCZ/SCLK0/CTS0/INT7/(SCL0)	SW0/4 / SBI SCL / SIO0 SCLK0
22	VOUT3	Unused
23	VINREG5	Unused
24	VOUT15	Unused
25	PM0/X1	Connect to high-speed oscillator
26	DVSSB	GND
27	PM1/X2	Connect to high-speed oscillator
28	DVDD5B	+5.0V
29	PE1/ENCB/RXD0/TB4IN	ENCB input / SIO0 Rx
30	PE0/ENCA/TXD0	ENCA input / SIO0Tx

5 Development Environment

Following is development environment:



1. Hardware board:
   TMPM3U0FSDMG evaluation board (TX03-BASEG-A + M3U0FSDMG-XFRM-A + M3U0FSDMG)

2. Development tool:
   - IAR:
     IDE:IAR Embedded workbench Version 8.22.2
   - ARM®:
     IDE: ARM KEIL® MDK Version 5.24.2.0

6 Functional description

6-1 Operation mode

There are three operation modes for TMPM3U0FSDMG: NORMAL, IDLE, STOP mode.

- **NORMAL mode:**

This mode is to operate the CPU core and the peripheral hardware by using the high-speed clock. It is shifted to the NORMAL mode after reset.

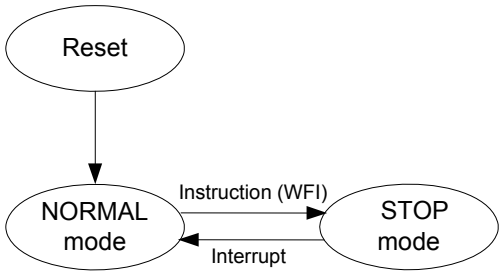
IDLE and STOP modes are low power mode.
To shift to lower power mode, in system control register CGSTBYCR<STBY[2:0]>, select the IDLE or STOP mode, and run the WFI (Wait For Interrupt) command.

Note: Only STOP mode is demonstrated in driver example.

- **STOP mode:**

All the internal circuits including the internal oscillator are brought to a stop in STOP mode.
When releasing STOP mode, an internal oscillator begins to operate and the operation mode changes to NORMAL mode.
The STOP mode enables to select the pin status by setting the CGSTBYCR<DRVE>.

Note: The sample program of STOP mode is integrated into CG example.



6-2 ADC

Change the value of VR1(POT1), which is connected to PJ6/AINB9. The voltage on it will be measured and used to change the loop blinking speed of the 4 LEDs in board.
The higher the voltage is, the faster the loop blinks.

6-3 CG

Change the CPU operation mode. Only 2 modes are supported, NORMAL and STOP. Toggle switch to switch the power mode.

Current mode	Action (Switch)	Operation	LED display
NORMAL	Press down SW1	NORMAL→ STOP	UART prints “NORMAL MODE” →”STOP MODE” LED 0,1,2,3 turns off.
STOP	Press down SW0	STOP →NORMAL	UART prints “STOP MODE” →” NORMAL MODE” LED 0,1,2,3 turns on.

6-4 ENC

This example detects the rolling of mouse wheel by TMPM3U0FSDMG ENC driver.

Demo procedure:

1. Use a wire to connect the pinA(see diagram 1) of mouse wheel encoder with the pin30(ENCA) of TMPM3U0FSDMG MCU board.
2. Use a wire to connect the pinB(see diagram 1) of mouse wheel encoder with the pin29(ENCB) of TMPM3U0FSDMG MCU board.
3. Connect the USB of mouse with the PC, the Com(see diagram 1) of mouse wheel encoder will connect with 5V.
4. After running the program,LED1 will blink.
5. While rolling the mouse wheel forward, LED1 will blink faster and faster. When the blink speed reach to the maximum, it will back to the minimum speed.
6. While rolling the mouse wheel backward, LED1 will blink slower and slower.When the blink speed reach to the minimum, it will back to the maximum speed.

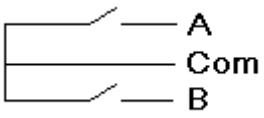


diagram 1

6-5 FLASH

This example demonstrates the feature of flash APIs such as erase and write operation.
—Reset procedure: includes programming routine(flash APIs), copy routine
▪Copy routine: copy programming routine(flash APIs) from flash to RAM
▪Programming routine: runs at RAM and write data to block1.

Demo sequence
(1) Power on
TMPM3U0FSDMG MCU board runs Reset Procedure.
Then program will start at block0.
If SW0 is turned off, LED1 blinks quickly.
(2) If SW0 is turned on 1st, Programming routine is copied to RAM by Copy routine.
Erase block1.
Write data to block1.
Compare the data read from the block1 with the written data, if they are equal, LED1 blinks slowly and LED2 always show.
If they are not equal or writing is unsuccessfully, LED2 will light off and LED1 will blink quickly.
(3) Same as step (1).

6-6 GPIO

This is a simple application based on the Peripheral Driver (GPIO), use GPIO API functions to configure LED and switch, turn on the LED, or turn off the LED.

6-7 VLTD

This application implements power supply voltage status detecting by VLTD.
When the power supply voltage is lower than the detection voltage, the MCU reset automatically by hardware and PG0 is set to high level by software.

6-8 OFD

This is a simple application based on the Peripheral Driver (OFD).

When the clock exceeds the OFD detection frequency range, OFD will generate a reset for I/O. Then software will be reset. The OFD reset flag will be set. If the flag is detected, OFD will be disabled and LED2 will be blinking.

6-9 SBI

6-9-1 SBI Slave

This sbi mode intends to support the I2C bus slave mode.
Connect two I2C buses (SBI & I2C) on evaluation board.
Work as I2C slave.

Use SBI interrupt to handle I2C bus read.
If the received address matches its slave address specified at SBII2CAR or is equal to the general-call address, the SBI pulls the SDA line to the "Low" level during the ninth clock and outputs an acknowledge signal.

I2C Slave (SBI) receives “TOSHIBA” from Master (SBI) and UART prints it.

SBI (slave) demo start,

```
K1: SBI SLAVE RECV
```

When SW1 is pressed, the string “TOSHIBA” that got from the SBI (slave) received buffer will be printed.

```
TOSHIBA
MASTER SBI to SLAVE SBI OK
```

6-9-2 SBI Master

This function intends to support the I2C bus master mode.
Connect two I2C buses (SBI & I2C) on evaluation board.
Work as I2C master.

Use SBI interrupt to handle I2C bus write.
Address of I2C: Any.
I2C Master (SBI) sends “TOSHIBA” to Slave (SBI).

SBI (master) demo start,
When SW3 is pressed, the string “TOSHIBA” will be sent from SBI (master) to SBI (slave).

6-10 TMRB

6-10-1 General Timer

This example implements a general timer utilizing MCU timer.
Trailing timing is set to 1ms.
Use this timer for all LED blinking and the period is 1s (500ms ON and 500ms OFF).

6-10-2 PPG Output

Use key SW0 to change PPG waveform. Leading timing can be changed as following:
10%, 25%, 50%, 75%, 90% (UART prints)
Power on to enter PPG mode and starts the PPG output.
Change the leading timing upon every SW0 pressed.
10% → 25% → 50% → 75% → 90% → 10%

6-11 UART

6-11-1 Retarget

This example intends to retarget the stdin and stdout of the C lib.
stdin and stdout are retargeted to UART. Then application code can use printf() to output to serial port.

6-11-2 UART FIFO

In this sample program, UART0 sent the data "TMPM3U01" to UART1 use FIFO, and at same time UART0 receive the data "TMPM3U02" which send from UART1 and also use FIFO.
Set one breakpoint before the function ResetIdx(),when program stop in the breakpoint you can read the RxBuffer = "TMPM3U02",and RxBuffer1 = "TMPM3U01".

Connect TX0 with RX1 and connect RX0 with TX1.

6-11-3 SIO Demo

This demo will show synchronously transfer and receive usage of SIO module.
It use the channel Master SIO0, Slave SIO0 and transfer data synchronously between them.(connect Master TXD0 with Slave RXD0, connect Master RXD0 with Slave TXD0, connect Master sclk0 with Slave sclk0)

*Note:
In this demo, slave applications must run first, and then run the host applications.

6-12 WDT

The watchdog timer cannot be used in the STOP mode while high-speed frequency clock is stopped.
Watch dog timer is enabled after reset, and is disabled in SystemInit().

The driver example step:

1. Initialize WDT by setting detection time and selecting WDT interrupt as counter overflow operation.
2. There are two demos for WDT and DEMO2 is defined by macro definition.

DEMO1:
When timer is overflow, NMI interrupt is generated and then WDT will be cleared.
DEMO2:
WDT clear code will be written to the watchdog timer control register, and watch dog timer counter will be cleared.

6-13 PMD

6-13-1 Example: Phase Output

This example demonstrates the phase outputs in PMD module.

Demo procedure:

1. Open the project and choose one demo to define: DEMO_U_PHASE or DEMO_3_PHASE. And then run the demo.
2. Use a wire to connect the EMG (pin 14) with the AVSSB (pin 1) to pull it low. (LED4 will light on) It means that PMD is in EMG protection.
3. Use a wire to connect the EMG (pin 14) with the ADVDD5B (pin 6) to pull it high. (LED4 will light off) It means that PMD is in normal operation.
4. Connect UO (pin 13) to oscilloscope to observe the waveform.
5. Connect XO (pin 12) to oscilloscope to observe the waveform.
6. Connect VO (pin 11) to oscilloscope to observe the waveform.
7. ConnectYO (pin 10) to oscilloscope to observe the waveform.
8. Connect WO (pin 9) to oscilloscope to observe the waveform.
9. Connect ZO (pin 8) to oscilloscope to observe the waveform.

Phenomenon:
If the DEMO_U_PHASE is defined, the phase outputs are same.

The waveform duty cycle of UO1 is 25%, the upper output is 5V, and the period is 410us.XO1 is the inverting of UO1.
VO1 and WO1 are same with UO1, and YO1 and ZO1 are same with XO1

If the DEMO_3_PHASE is defined, the phase outputs are different.
The waveform duty cycle of UO1 is 25%, the upper output is 5V, and the period is 410us.XO1 is the inverting of UO1.
The waveform duty cycle of VO1 is 50%, the upper output is 5V, and the period is 410us.YO1 is the inverting of VO1.
The waveform duty cycle of WO1 is 75%, the upper output is 5V, and the period is 410us.ZO1 is the inverting of WO1.

7 Software

This software project creates the sample applications based on TMPM3U0FSDMG evaluation board which will demonstrate the main feature of TMPM3U0FSDMG MCU.

Use current driver software and IAR EWARM or KEIL MDK to implement the sample applications. Create an independent project for each feature.

The workspace structure and project names are defined as following:

IAR EWARM:

```
├─WDT_NMI
        ├─APP
        │   main.c
        │   tmpm3U0_wdt_int.c
        │   tmpm3U0_wdt_int.h
        │
        ├─TX03_CMSIS
        │ │ system_TMPM3U0.c
        │ │ system_TMPM3U0.h
        │ │ TMPM3U0.h
        │ │
        │ └─startup
              startup_TMPM3U0.s
        │
        ├─TX03_Periph_Driver
        │   ├─inc
        │   │   tmpm3U0_cg.h
        │   │   tmpm3U0_gpio.h
        │   │   tmpm3U0_wdt.h
```

```
│   │   tx03_common.h
│   │
│   └─src
│         tmpm3U0_cg.c
│         tmpm3U0_gpio.c
│         tmpm3U0_wdt.c
│
└─TMPM3U0-EVAL
        led.c
        led.h
        sw.c
        sw.h
        common_uart.c
        common_uart.h
```

KEIL MDK:

```
├─WDT_NMI
    ├─APP
    │   main.c
    │   tmpm3U0_wdt_int.c
    │
    ├─TX03_CMSIS
    │    system_TMPM3U0.c
    │    startup_TMPM3U0.s
    │
    ├─TX03_Periph_Driver
    │    tmpm3U0_cg.c
    │    tmpm3U0_gpio.c
    │    tmpm3U0_wdt.c
    │
    └─TMPM3U0-EVAL
         led.c
         sw.c
         common_uart.c
```

7-1 ADC

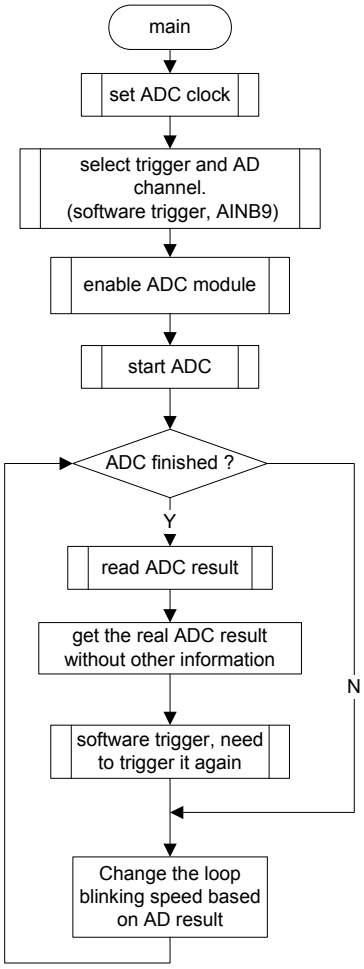
7-1-1 Example: ADC Data Read

This is a simple example based on the TX03 Peripheral Driver (ADC, GPIO).

The example includes:

1. ADC configuration and initialization
2. Start AD conversion by software trigger and read AD result
3. Use AD conversion result to adjust the loop blinking speed.

- **Flowchart**



- **Code and Explanation for the Example**

At first, set ADC clock, select trigger type and AD channel, enable ADC module.

```
    /* 1. set ADC clock */
    ADC_SetClk(TSB_ADB, ADC_HOLD_FIX, ADC_FC_DIVIDE_LEVEL_8);

    /* 2. select trigger and AD channel, this time we use sofeware trigger, */
    /* the VR1 is connected to ADC unit B channel 9, remember to input with macro
TRG_ENABLE() */
    ADC_SetSWTrg(TSB_ADB, ADC_REG0, TRG_ENABLE(ADC_AIN9));

    /* 3. enable ADC module */
    ADC_Enable(TSB_ADB);
```

Then start AD conversion:

```
    ADC_Start(TSB_ADB, ADC_TRG_SW);
```

After started AD conversion, check ADC module state. When AD conversion is finished, start another AD conversion.

```
while (1U) {
    /* check ADC module state */
    adcState = ADC_GetConvertState(TSB_ADB, ADC_TRG_SW);

    if (adcState == DONE) {
        /* read ADC result when it is finished */
        result = ADC_GetConvertResult(TSB_ADB, ADC_REG0);

        /* get the real ADC result without other information */
        /* "/256" is to limit the range of AD value */
        myResult = 16U - result.Bit.ADResult / 256U;

        /* software trigger, need to trigger it again */
        ADC_Start(TSB_ADB, ADC_TRG_SW);
    }
```

After get the AD result in “myResult”, use it to change the loop blinking speed of the 4 LEDs.

7-2 CG

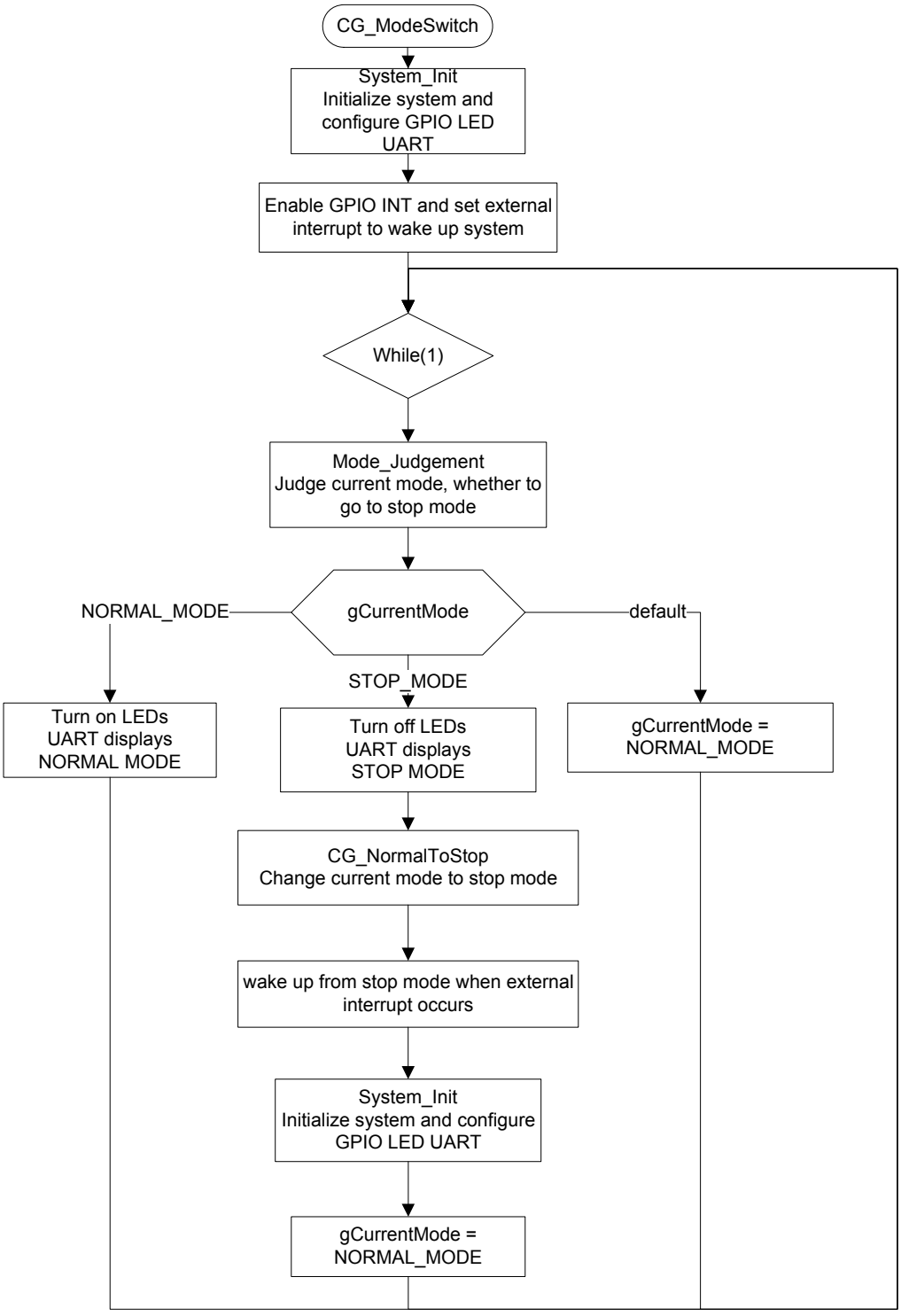
7-2-1 Example: Power Mode Change

This is a simple example based on the TX03 Peripheral Driver (CG, GPIO,UART).

The example includes:

1. Basic setup operation of CG
2. How to switch between normal mode and stop mode

- **Flowchart**



- **Code and Explanation for the Example**

The following simple example is based on TX03 Peripheral Driver (CG), which will switch between normal mode and stop mode.

Normal setup for CG (after reset)

The following code is just an example for setting CG in normal mode. It is supposed that the high-speed oscillator is 10MHz.

Example code is as the following:

```
/* Set fgear = fc/2 */
CG_SetFgearLevel(CG_DIVIDE_2);
/* Set fperiph to fgear */
CG_SetPhiT0Src(CG_PHIT0_SRC_FGEAR);
CG_SetPhiT0Level(CG_DIVIDE_64);
TSB_CG->SYSCR &= 0x10000;
/* Enable high-speed oscillator */
CG_SetFosc(CG_FOSC_OSC1,ENABLE);
/* select external oscillator */
CG_SetFoscSrc(CG_FOSC_OSC1);
/* Set low power consumption mode stop */
CG_SetSTBYMode(CG_STBY_MODE_STOP);
/* Set pin status in stop mode to "active" */
CG_SetPinStateInStopMode(ENABLE);
/* Set up pll and wait for pll to warm up, set fc source to fpll */
CG_EnableClkMulCircuit();
```

Configure external interrupt to wake up system

Confiure external interrupt INT7 to wake up system. Clear interrupt pending request, then enable INT7.

```
__disable_irq();
CG_ClearINTReq(CG_INT_SRC_7);
/* Set external interrupt to wake up system */
CG_SetSTBYReleaseINTSrc(CG_INT_SRC_7,
CG_INT_ACTIVE_STATE_RISING, ENABLE);
NVIC_ClearPendingIRQ(INT7_IRQn);
NVIC_EnableIRQ(INT7_IRQn);
__enable_irq();
```

Setup to enter STOP mode

Prepare for entering stop mode. Set warm up time, use __WFI() instruction to enter stop mode.

```
/* Set CG module: Normal ->Stop mode */
CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC1, CG_WUODR_EXT);
/* Enter stop mode */
__WFI();
```

Enable multiple clock circuit

First set PLL value, and then set warm up time and wait warm up time is completed. Finally set fPLL as fc source.

```
WorkState st = BUSY;
CG_SetPLL(DISABLE);
CG_SetFPLLValue(CG_FPLL_8M_MULTIPLY_5);
retval = CG_SetPLL(ENABLE);
if (retval == SUCCESS) {
    /* Set warm up time */
    CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC1, CG_WUODR_PLL);
    CG_StartWarmUp();

    do {
        st = CG_GetWarmUpState();
```

```
        } while (st != DONE);

        retval = CG_SetFcSrc(CG_FC_SRC_FPLL);
    } else {
        /*Do nothing */
    }
```

7-3 ENC

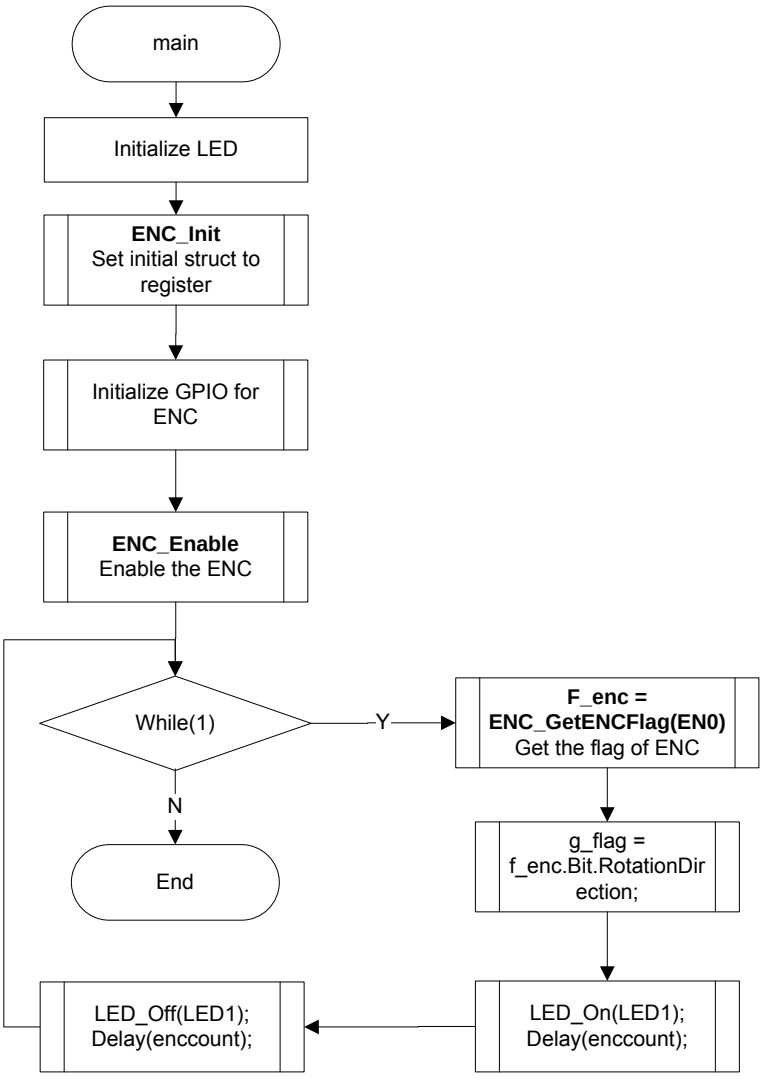
7-3-1 Example: Rolling Detection

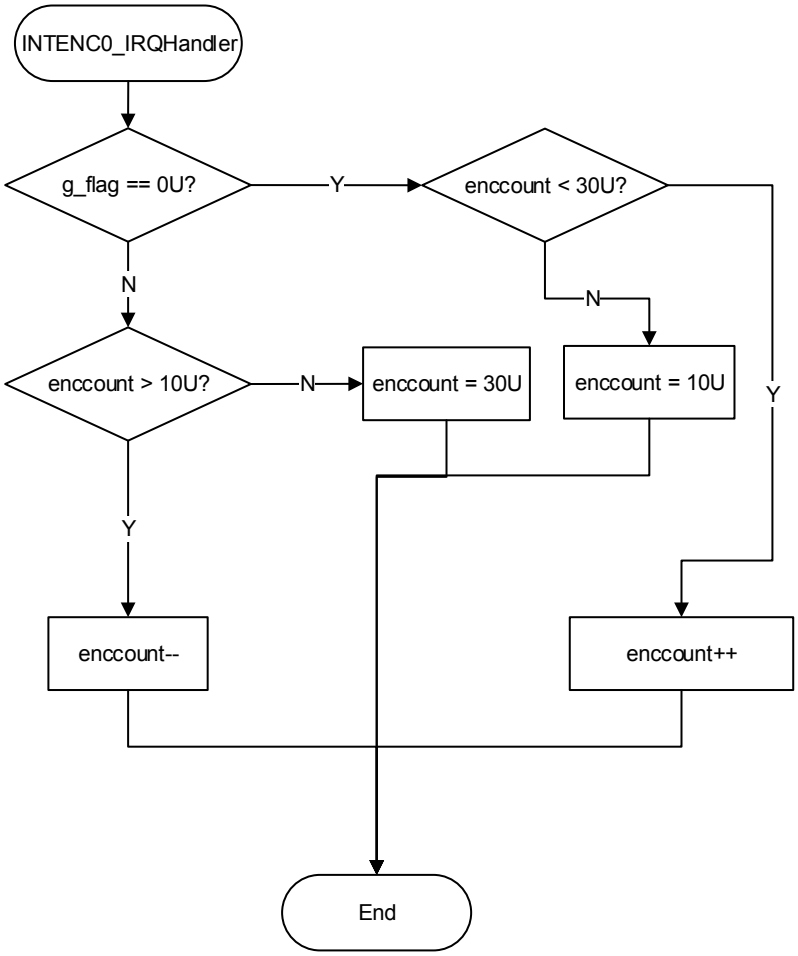
This is a simple example based on the TX03 Peripheral Driver (ENC, GPIO).

The example includes:

1. ENC0 initialization.
2. Rolling detection of mouse wheel.

- **Flowchart**





- **Code and Explanation for the Example**

The following simple example is based on TX03 Peripheral Driver (ENC), which will detect the rolling of mouse wheel.

At first, initialize LEDs.

```
/* LED initialization */
LED_Init();
```

Prepare ENC initialization structure, and initialize the ENC0.

```
/* ENC0 initialization */
m_enc.ModeType = ENC_ENCODER_MODE;
m_enc.PhaseType = ENC_TWO_PHASE;
m_enc.CompareStatus = ENC_COMPARE_DISABLE;
m_enc.ZphaseStatus = ENC_ZPHASE_DISABLE;
m_enc.FilterValue = ENC_FILTER_VALUE31;
m_enc.IntEn = ENC_INTERRUPT_ENABLE;
m_enc.PulseDivFactor = ENC_PULSE_DIV1;

ENC_Init(EN0,&m_enc);
ENC_SetCounterReload(EN0,0xFFFU);
```

Initialize the GPIO for ENC0.Set the PE0 and PE1 as input of ENC0.

```
/* GPIO initialization */
GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_0, ENABLE);/*Set PE0 as
```

```
input */
    GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_3, GPIO_BIT_0);/*Set
PE0 as ENCA */
    GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_1, ENABLE);/*Set PE1 as
input */
    GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_3, GPIO_BIT_1);/*Set
PE1 as ENCB */
```

Enable the ENC0 and ENC0 interrupt.

```
    /* Enable ENC0 */
    ENC_Enable(EN0);
    /* Enable ENC0 interrupt */
    NVIC_EnableIRQ(INTENC0_IRQn);
```

Get the status of rotation direction of ENC0 to indicate the rolling direction of the mouse wheel.Blink the LED1 according to the trailing timing of rolling

```
    /* LED1 blink speed based on the rolling of mouse wheel */
    while(1){
        f_enc = ENC_GetENCFlag(EN0);
        g_flag = f_enc.Bit.RotationDirection;
        LED_On(LED1);
        Delay2(enccount);
        LED_Off(LED1);
        Delay2(enccount);
    }
```

ENC count is the counter of rolling, and it was changed in the interrupt routine.
While rolling the mouse wheel forward, the ENC counter is back to 30 when ENC counter is decreased to 10.and the ENC counter is back to 10 when ENC counter is increased to 30.

```
  if(g_flag == 1U){
        if(enccount < 30U){
            enccount++;
        } else {
            enccount = 10U;
        }
   } else {
        if(enccount > 10U){
            enccount--;
        } else {
            enccount = 30;
        }
    }
```

7-4 FLASH

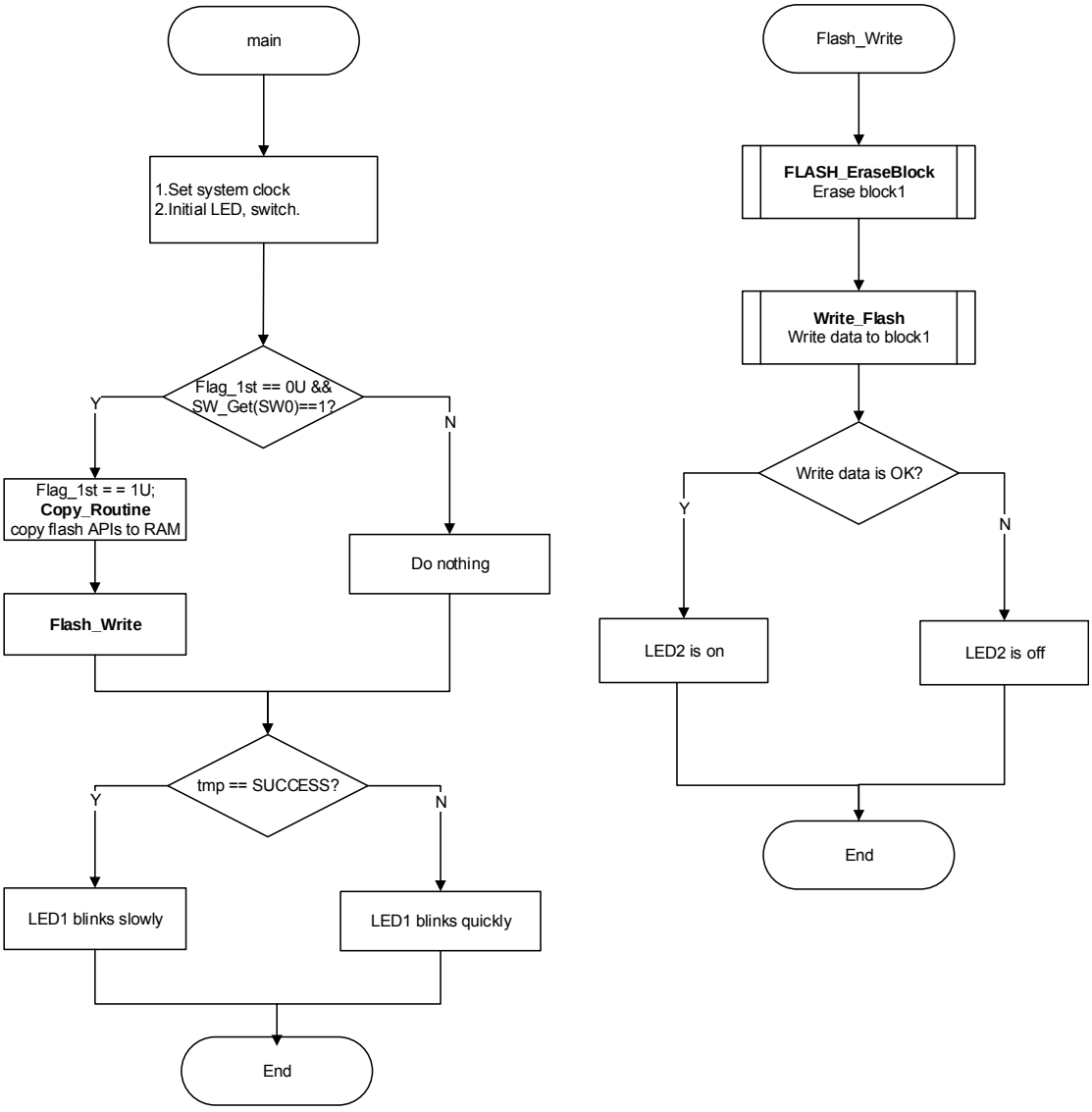
7-4-1 Example: Flash Write

This is a simple example based on the TX03 Peripheral Driver (FLASH, GPIO).

The example includes:

1. On-board programming method of Flash Memory (Rewrite/Erase)

- **Flowchart**



- **Code and Explanation for the Example**

At first initialize LED and switch.

```
LED_Init();
SW_Init();
```

If the SW0 is turned off after reset, the LED1 blinks quickly.
If the SW0 is turned on first time, the routine will copy APIs for flash operation from address “FLASH_API_ROM” in Flash memory to address “FLASH_API_RAM” in RAM, because Flash memory cannot erase/program itself when runs the code at the same time.

```
Copy_Routine(FLASH_API_RAM, FLASH_API_ROM, SIZE_FLASH_API);
```

After copying Flash operation APIs to RAM, it will run function Flash_Write().
If the write operation is OK, LED1 will blink slowly.

```
while (1U) {
        if(Flag_1st == 0U && SW_Get(SW0)) {     /* if SW0 is turned on 1st time*/
            Flag_1st = 1U;
            Copy_Routine(FLASH_API_RAM,                    FLASH_API_ROM,
SIZE_FLASH_API);
            tmp = Flash_Write();
        } else {
            /* Do nothing */
        }
        if(tmp== SUCCESS)
        {
            LED_On(LED1);
            delay(SUCCESS_DELAY);
            LED_Off(LED1);
            delay(SUCCESS_DELAY);
        } else {
            LED_On(LED1);
            delay(ERROR_DELAY);
            LED_Off(LED1);
            delay(ERROR_DELAY);
        }
    }
```

In function Flash_Write(), it will call function FC_EraseBlock() and FC_WritePage () to erase and program flash memory. The data will be written to block1.
If the data read from block1 is same as the written data, LED2 will always show.
If they are not equal, LED2 will light off.

```
    int i = 0U;
    uint32_t pagedata[FC_PAGE_SIZE];
    Result tmp = ERROR;
    uint32_t buffer[FC_PAGE_SIZE] = {0U};
    if (FC_SUCCESS == FC_EraseBlock(FC_WRITE_ADDR)) {                    /*
erase this block which will be written */
        for (i = 0U; i < FC_PAGE_SIZE; i++) {
            pagedata[i] = 0x12345678U;
        }
        if  (FC_WritePage(FC_WRITE_ADDR,  pagedata)  ==  FC_SUCCESS)
/* write data to block which was erased */
        {
            Copy_Routine(buffer,                (uint32_t*)FC_WRITE_ADDR,
FLASH_PAGE_SIZE);
            tmp = SUCCESS;
            LED_On(LED2);
            for (i = 0U; i < FC_PAGE_SIZE; i++) {
                if(buffer[i] != pagedata[i]) {
                    tmp = ERROR;
                    LED_Off(LED2);
                    break;
                }else {
                   /* Do nothing */
                }
            }
        }else {
            /* Do nothing */
        }
    } else {
```

```
        /* Do nothing */
    }
    return tmp;
```

Flash memory operation function FC_EraseBlock () will erase a specified block automatically. The block is specified by parameter “block_addr”. Firstly, this function will check whether the parameter “block_addr” is illegal. Then it will use Flash driver FC_GetBlockProtectState() to check if the specified block is protected.

```
    if (ENABLE == FC_GetBlockProtectState(BlockNum)) {
        retval = FC_ERROR_PROTECTED;
    }
```

If the block is protected, it will return “FC_ERROR_PROTECTED”, otherwise it will send automatic block erase command to erase the block.

```
        *addr1 = (uint32_t) 0x000000AA; /* bus cycle 1 */
        *addr2 = (uint32_t) 0x00000055; /* bus cycle 2 */
        *addr1 = (uint32_t) 0x00000080; /* bus cycle 3 */
        *addr1 = (uint32_t) 0x000000AA; /* bus cycle 4 */
        *addr2 = (uint32_t) 0x00000055; /* bus cycle 5 */
        *BA = (uint32_t) 0x00000030;       /* bus cycle 6 */
```

Then the function will use Flash driver FC_GetBusyState() to monitor when erasing operation finished. At the same time, use a timeout counter to check if the operation is overtime.

```
        while (BUSY == FC_GetBusyState()) {        /* check if FLASH is busy with
overtime counter */
            if (!(counter--)) { /* check overtime */
                retval = FC_ERROR_OVER_TIME;
                break;
            } else {
                /* Do nothing */
            }
        }
```

Function FC_WritePage () is used for write data automatically in one page. The process of this function is basically same as FC_EraseBlock() except the automatic page program command.

```
        *addr1 = (uint32_t) 0x000000AA; /* bus cycle 1 */
        *addr2 = (uint32_t) 0x00000055; /* bus cycle 2 */
        *addr1 = (uint32_t) 0x000000A0; /* bus cycle 3 */
        for (i = 0U; i < FC_PAGE_SIZE; i++) {     /* bus cycle 4~67 */
            *addr3 = *source;
            source++;
        }
```

7-5 GPIO

This function configures GPIO to make LED and Switch work. Read Switch to control LED on and LED off.

7-5-1 Example: GPIO Data Read

This is a simple example based on the TX03 Peripheral Driver (GPIO).

The example includes:

1. GPIO initialization
2. Write data to GPIO
3. Read data from GPIO

- **Code and Explanation for the Example**

At first, use GPIO_SetOutput(GPIO_Port GPIO_x, uint8_t Bit_x) to configure GPIO to LED, and GPIO_SetInput(GPIO_Port GPIO_x, uint8_t Bit_x) to configure GPIO to key. For example,

```
GPIO_SetOutput(GPIO_PG, GPIO_BIT_0);
GPIO_SetOutput(GPIO_PG, GPIO_BIT_1);
GPIO_SetOutput(GPIO_PG, GPIO_BIT_2);
GPIO_SetOutput(GPIO_PG, GPIO_BIT_3);
GPIO_SetInput(GPIO_PE, GPIO_BIT_2);
GPIO_SetInput(GPIO_PJ, GPIO_BIT_7);
GPIO_SetInput(GPIO_PK, GPIO_BIT_0);
GPIO_SetInput(GPIO_PK, GPIO_BIT_1);
```

In the for(; ;) process, run the LED demo: Read Switch to control LED on and LED off.

Read Switch by using GPIO_ReadDataBit(GPIO_Port GPIO_x, uint8_t Bit_x).

```
if (GPIO_ReadDataBit(GPIO_PE, GPIO_BIT_2) == 1U) {
        tmp = 1U;
} else {
        /*Do nothing */
}
```

Turn on LED by using GPIO_WriteDataBit(GPIO_Port GPIO_x, uint8_t Bit_x, uint8_t BitValue)

```
uint32_t tmp;
tmp = GPIO_ReadData(GPIO_PG);
tmp |= led;
GPIO_WriteData(GPIO_ PG, tmp);
```

Turn off LED by using GPIO_WriteDataBit(GPIO_Port GPIO_x, uint8_t Bit_x, uint8_t BitValue)

```
uint32_t tmp;
 tmp = GPIO_ReadData(GPIO_PG);
 tmp &= (uint8_t)(~led);
 GPIO_WriteData(GPIO_PG, tmp);
```

7-6 VLTD

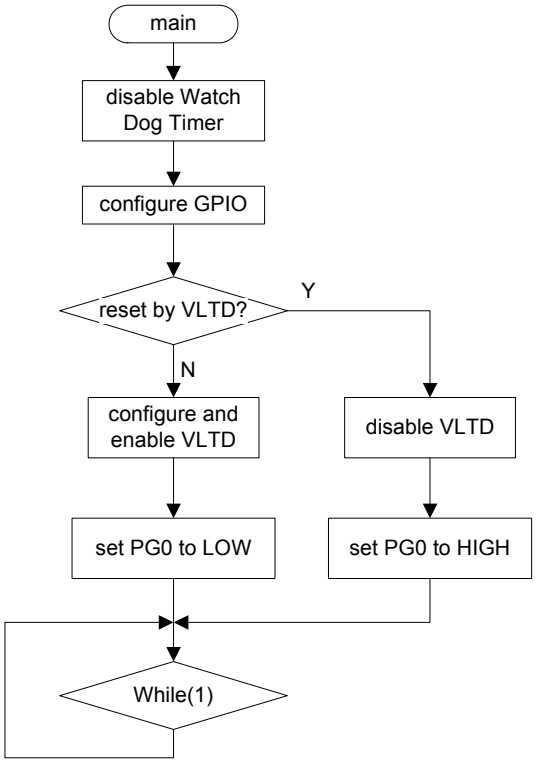
7-6-1 Example: VLTD Reset

This is a simple example based on the TX03 Peripheral Driver (VLTD, CG, GPIO, WDT).

The example includes:

1. VLTD configuration

- **Flowchart:**



- **Code and Explanation for the Example**

At first, disable Watch Dog Timer explicitly. Then set PG0 as output port.

```
WDT_Disable();
GPIO_SetOutput(GPIO_PG, GPIO_BIT_0);
```

Then read VLTD reset flag If it is not reset by VLTD, configure and enable VLTD. Set PG0 to LOW.

```
if (CG_GetResetFlag().Bit.VLTDReset == 0U) {
        VLTD_SetVoltage(VLTD_DETECT_VOLTAGE_46);
        VLTD_Enable();
        GPIO_WriteData(GPIO_PG, 0x00U);
```

Otherwise, disable VLTD and set PG0 to HIGH.

```
} else {
    VLTD_Disable();
    GPIO_WriteData(GPIO_PG, 0x01U);
}
```

7-7 OFD

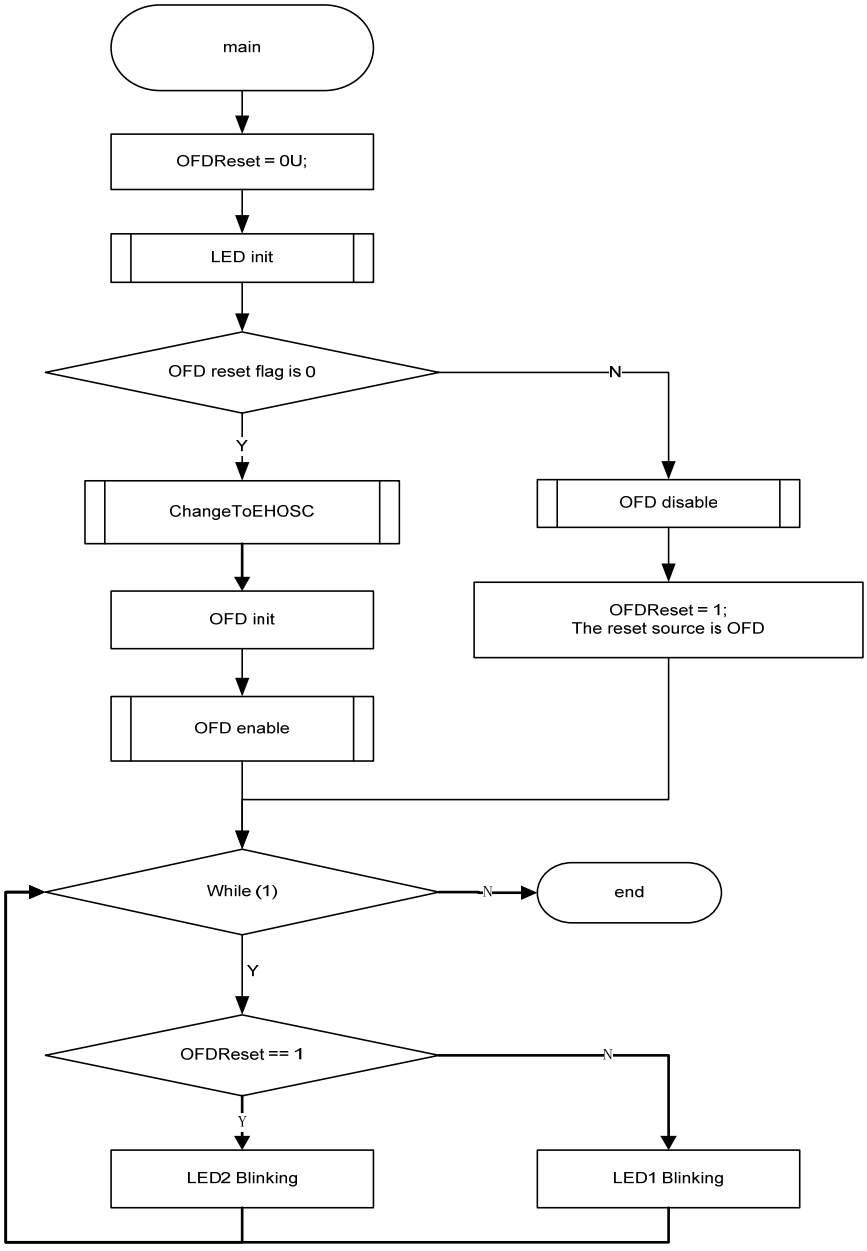
7-7-1 Example: OFD Reset

This is a simple example based on the TX03 Peripheral Driver (OFD, CG, and GPIO).

The example includes:

1. OFD initialization (set OFD detect frequency range)
2. OFD reset flag checking (in CG)
3. OFD enable and disable.

- **Flowchart**



- **Code and Explanation for the Example**

At first, initialize LED.

```
LED_Init ();
```

Check the OFD reset flag.

```
resetFlag = CG_GetResetFlag();
if (resetFlag.Bit.OFDReset == 0U) {
  /* reset source isn't OFD */
}
```

In normal reset, for example, when the MCU is reset by power up, the OFD flag will

be '0', then the program change to use the external oscillation and initialize OFD to set OFD detect frequency range.

Below code is to set up the CG to use the external oscillation.

```
void ChangeToEHOSC(void)
{
    /* Use the external oscillation */
    TSB_CG->SYSCR = 0x00010000UL;
    TSB_CG->OSCCR &= 0x000FFFFFUL;
    TSB_CG->OSCCR |= 0xC3500000UL;          /*Set the warm up time WUODR[13:2] */
    TSB_CG_OSCCR_HOSCON = 1U;      /*Set port as X1/X2 for external oscillator */
    TSB_CG_OSCCR_XEN1 = 1U;        /*Enable external oscillator */
    TSB_CG_OSCCR_WUPSEL2 = 1U;
    TSB_CG_OSCCR_WUPSEL1 = 0U;    /*Select warm-up clock */
    TSB_CG_OSCCR_WUEON = 1U;       /*Start warm up */
    TSB_CG->PLLSEL = (0x0000680FUL<<1U);
    while (TSB_CG_OSCCR_WUEF) {
    }                                   /* Warm-up */
    TSB_CG_OSCCR_OSCSEL = 1U;     /* Use the external oscillation */
    TSB_CG_OSCCR_XEN2 = 1U;
    TSB_CG->OSCCR &= 0x000FFFFFUL;
    TSB_CG->OSCCR |= 0x03E80000UL;
    TSB_CG_OSCCR_PLLON = 1U;       /* PLL enabled */
    TSB_CG->STBYCR = 0x00000103UL;
    TSB_CG_OSCCR_WUEON = 1U;
    while (TSB_CG_OSCCR_WUEF) {
    }                                   /* Warm-up */

    TSB_CG->PLLSEL = ((0x0000680FUL<<1U)|1U);
}
```

To configurate the OFD, set to enable to write its register at first.

```
    OFD_SetRegWriteMode(ENABLE);
```

Then set detection frequency in PLL OFF

```
    OFD_SetDetectionFrequency(OFD_PLL_OFF, 0X25, 0X1D);
```

Then set detection frequency in PLL ON

```
    OFD_SetDetectionFrequency(OFD_PLL_ON,OFD_HIGHER_COUNT, OFD_LOWER_COUNT);
```

If define DEMO2, the abnormal frequency range is set.

```
    OFD_SetDetectionFrequency(OFD_PLL_ON,OFD_HIGHER_COUNT_ABNO
```

```
RMAL, OFD_LOWER_COUNT_ABNORMAL);
```

Enable OFD after initialization.

```
    OFD_Enable();
```

After above setting, OFD will be ready to detect the frequency of external clock. If the clock exceeds the detection frequency range (for example, take the external oscillator out, or make it short, or define DEMO2), OFD will generate a reset signal in the reset source register which is in CG module. Then the MCU will be reset automatically.
If the OFD reset flag is detected, MCU will run under the default inner oscillator, then OFD function will be disabled and the variable OFDReset will be set to 1.

```
    OFD_Disable();
    OFDReset = 1U;
```

LED1 and LED2 indicate the system clock status as below.

LED status	Meaning
LED1 blinking	MCU run normally with external oscillator works OK, OFD function is enabled and ready to detect oscillator abnormal status
LED2 blinking	External oscillator failure, causes the OFD reset, MCU use the default setting to run under inner oscillator. OFD function is disabled.

```
while (1U) {
    if (OFDReset == 1U) {
        LED_On(LED2);
        Delay();
        LED_Off(LED2);
        Delay();
    } else {
        LED_On(LED1);
        Delay();
        LED_Off(LED1);
        Delay();
    }
}
```

7-8 SBI

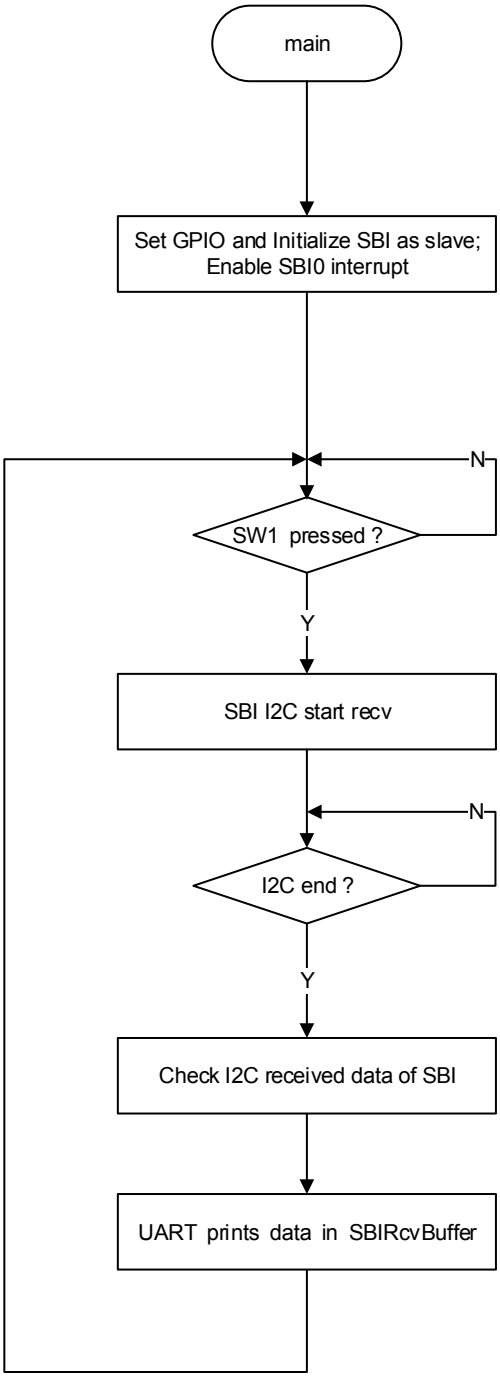
7-8-1 Example: SBI Slave

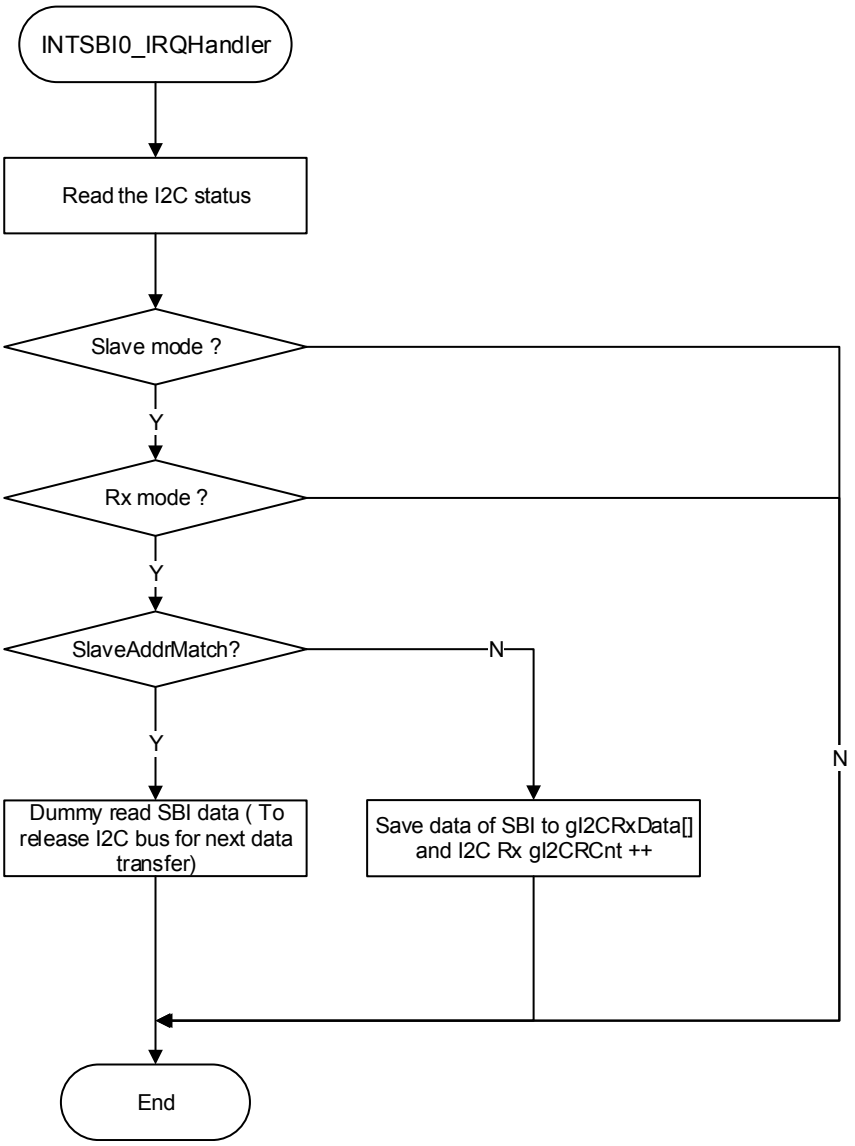
This is a simple example based on the TX03 Peripheral Driver (SBI, GPIO).

The example includes:

1. SBI configuration
2. SBI slave receive data process

- **Flowchart**





- **Code and Explanation for the Example**

At first, configure GPIO for SBI I2C mode.

```
GPIO_EnableFuncReg(GPIO_PB, GPIO_FUNC_REG_5, GPIO_BIT_5);
GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_5, GPIO_BIT_2);
GPIO_SetOutputEnableReg(GPIO_PB, GPIO_BIT_5, ENABLE);
GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_2, ENABLE);
GPIO_SetInputEnableReg(GPIO_PB, GPIO_BIT_5, ENABLE);
GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_2, ENABLE);
GPIO_SetOpenDrain(GPIO_PB, GPIO_BIT_5, ENABLE);
GPIO_SetOpenDrain(GPIO_PE, GPIO_BIT_2, ENABLE);
/* Pull up for SDA&SCL */
GPIO_SetPullUp(GPIO_PB, GPIO_BIT_5, ENABLE);
GPIO_SetPullUp(GPIO_PE, GPIO_BIT_2, ENABLE);
```

Then enable, initialize and configure slave SBI channel and enable INTSBI0.

```
myI2C.I2CSelfAddr = SLAVE_ADDR;
myI2C.I2CDataLen = SBI_I2C_DATA_LEN_8;
myI2C.I2CACKState = ENABLE;
myI2C.I2CClkDiv = SBI_I2C_CLK_DIV_328;
```

```
    SBI_Enable(TSB_SBI);
    SBI_SWReset(TSB_SBI);
    SBI_InitI2C(TSB_SBI, &myI2C);
    NVIC_EnableIRQ(INTSBI0_IRQn);
```

After the above setting, start I2C wait for receive.
Clear I2C Rx buffer, initialize the SBI Rx buffer and length,

```
        /* Initialize TRx buffer and Tx length */
    case MODE_SBI_I2C_INITIAL:
        for (gICnt = 0U; gICnt < 8U; gICnt++) {
            gI2CRxData[gICnt] = 0U;
        }
        gSBIMode = MODE_SBI_I2C_RCV;
        break;
```

The data receive is handled in INTSBI0.

In INTSBI0 function, I2C slave receive process is handled according to I2C bus state, SBI_GetReceiveData() is used to read received data in SBI buffer, I2C stop condition is controlled by master.

```
void INTSBI0_IRQHandler(void)
{
    uint32_t tmp = 0U;
    TSB_SBI_TypeDef *SBIy;
    SBI_I2CState sbi0_sr;

    SBIy = TSB_SBI;
    sbi0_sr = SBI_GetI2CState(SBIy);

    if (!sbi0_sr.Bit.MasterSlave) {        /* Slave mode */
        if (!sbi0_sr.Bit.TRx) { /* Rx Mode */
            if (sbi0_sr.Bit.SlaveAddrMatch) {
                /* First read is dummy read for Slave address recognize */
                tmp = SBI_GetReceiveData(SBIy);
                gI2CRCnt = 0U;
            } else {
                /* Read I2C received data and save to I2C_RxData buffer */
                tmp = SBI_GetReceiveData(SBIy);
                gI2CRxData[gI2CRCnt] = tmp;
                gI2CRCnt++;
            }
        } else {                        /* Tx Mode */
            /* Do nothing */
        }
    } else {                            /* Master mode */
        /* Do nothing */
    }
}
```

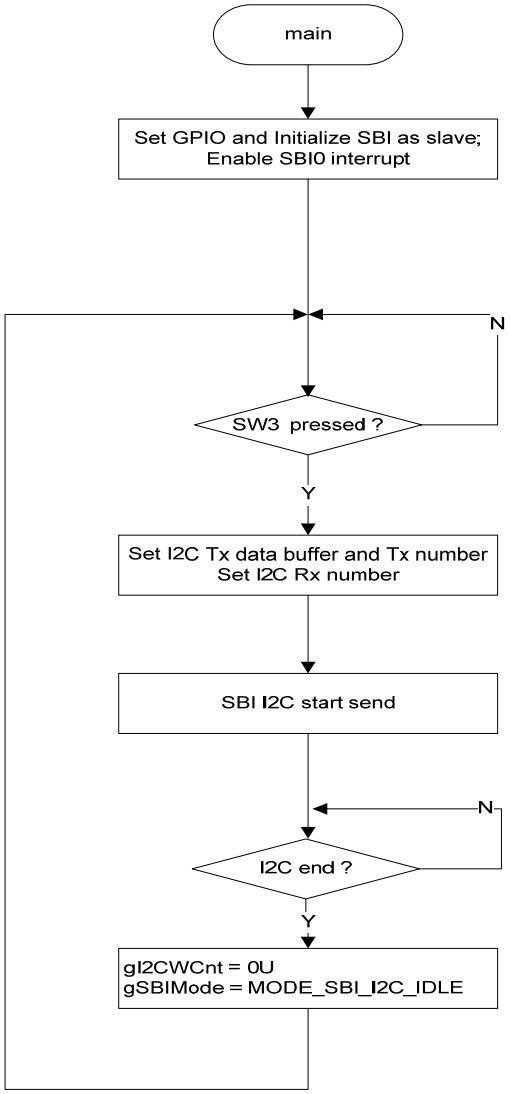
7-8-2 Example: SBI Master

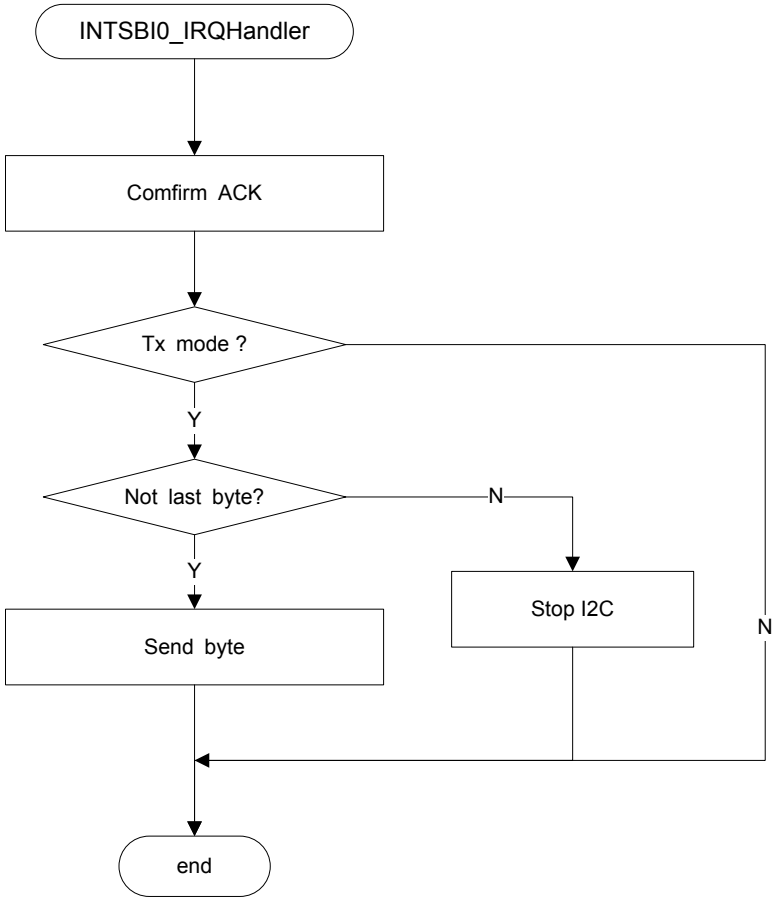
This is a simple example based on the TX03 Peripheral Driver (SBI, GPIO).

The example includes:

1 SBI configuration
2 SBI master send data process

- **Flowchart**





- **Code and Explanation for the Example**

At first, configure GPIO for SBI I2C mode.

```
GPIO_EnableFuncReg(GPIO_PB, GPIO_FUNC_REG_5, GPIO_BIT_5);
GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_5, GPIO_BIT_2);
GPIO_SetOutputEnableReg(GPIO_PB, GPIO_BIT_5, ENABLE);
GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_2, ENABLE);
GPIO_SetInputEnableReg(GPIO_PB, GPIO_BIT_5, ENABLE);
GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_2, ENABLE);
GPIO_SetOpenDrain(GPIO_PB, GPIO_BIT_5, ENABLE);
GPIO_SetOpenDrain(GPIO_PE, GPIO_BIT_2, ENABLE);
/* Pull up for SDA&SCL */
GPIO_SetPullUp(GPIO_PB, GPIO_BIT_5, ENABLE);
GPIO_SetPullUp(GPIO_PE, GPIO_BIT_2, ENABLE);
```

Then enable, initialize and configure slave SBI channel and enable INTSBI0.

```
myI2C.I2CSelfAddr = SELF_ADDR;
myI2C.I2CDataLen = SBI_I2C_DATA_LEN_8;
myI2C.I2CACKState = ENABLE;
myI2C.I2CClkDiv = SBI_I2C_CLK_DIV_328;
SBI_Enable(TSB_SBI);
SBI_SWReset(TSB_SBI);
```

```
    SBI_InitI2C(TSB_SBI, &myI2C);
    NVIC_EnableIRQ(INTSBI0_IRQn);
```

After the above setting, start I2C send.
Clear I2C Rx buffer, initialize the SBI Tx buffer and length, and clear Rx buffer.

```
        /* Initialize TRx buffer and Tx length */
    case MODE_SBI_I2C_INITIAL:
        gI2CTxDataLen = 7U;
        gI2CTxData[0] = gI2CTxDataLen;
        gI2CTxData[1] = 'T';
        gI2CTxData[2] = 'O';
        gI2CTxData[3] = 'S';
        gI2CTxData[4] = 'H';
        gI2CTxData[5] = 'I';
        gI2CTxData[6] = 'B';
        gI2CTxData[7] = 'A';

        gI2CWCnt = 0U;
 gSBIMode = MODE_SBI_I2C_START;
        break;
```

Check if the I2C bus is free or not, SBI_SetSendData() is used to set data “SLAVE_ADDR”.and direction is “SBI_I2C_SEND” to SBI data buffer; then SBI_GenerateI2CStart(TSB_SBI) is used to to start I2C process.

```
        /* Check I2C bus state and start TRx */
    case MODE_SBI_I2C_START:
        i2c_state = SBI_GetI2CState(TSB_SBI);
        if (!i2c_state.Bit.BusState) {
            SBI_SetSendData(TSB_SBI, SLAVE_ADDR | SBI_I2C_SEND);
            SBI_GenerateI2CStart(TSB_SBI);
            gSBIMode = MODE_SBI_I2C_TRX;
        } else {
            /* Do nothing */
        }
        break;
```

The data transfer is handled in INTSBI0.

In INTSBI0 function, I2C master send process is handled according to I2C bus state. During I2C master sending, SBI_SetSendData() is used to send next data, SBI_GenerateI2CStop() is used to stop I2C when I2C process is finished.

```
void INTSBI0_IRQHandler(void)
{
    TSB_SBI_TypeDef *SBIx;
    SBI_I2CState sbi_sr;

    SBIx = TSB_SBI;
    sbi_sr = SBI_GetI2CState(SBIx);

    if (sbi_sr.Bit.MasterSlave) {           /* Master mode */
        if (sbi_sr.Bit.TRx) {     /* Tx mode */
            if (sbi_sr.Bit.LastRxBit) { /* LRB=1: the receiver requires no further
data. */
                SBI_GenerateI2CStop(SBIx);
            } else {                    /* LRB=0: the receiver requires further data. */
                if (gI2CWCnt <= gI2CTxDataLen) {
```

```
                        SBI_SetSendData(SBIx, gI2CTxData[gI2CWCnt]);            /*
Send next data */
                        gI2CWCnt++;
                    } else {            /* I2C data send finished. */
                        SBI_GenerateI2CStop(SBIx);    /* Stop I2C */
                    }
                }
            } else {                    /* Rx Mode */
                /* Do nothing */
            }
        } else {                        /* Slave mode */
            /* Do nothing */
        }
}
```

7-9 TMRB

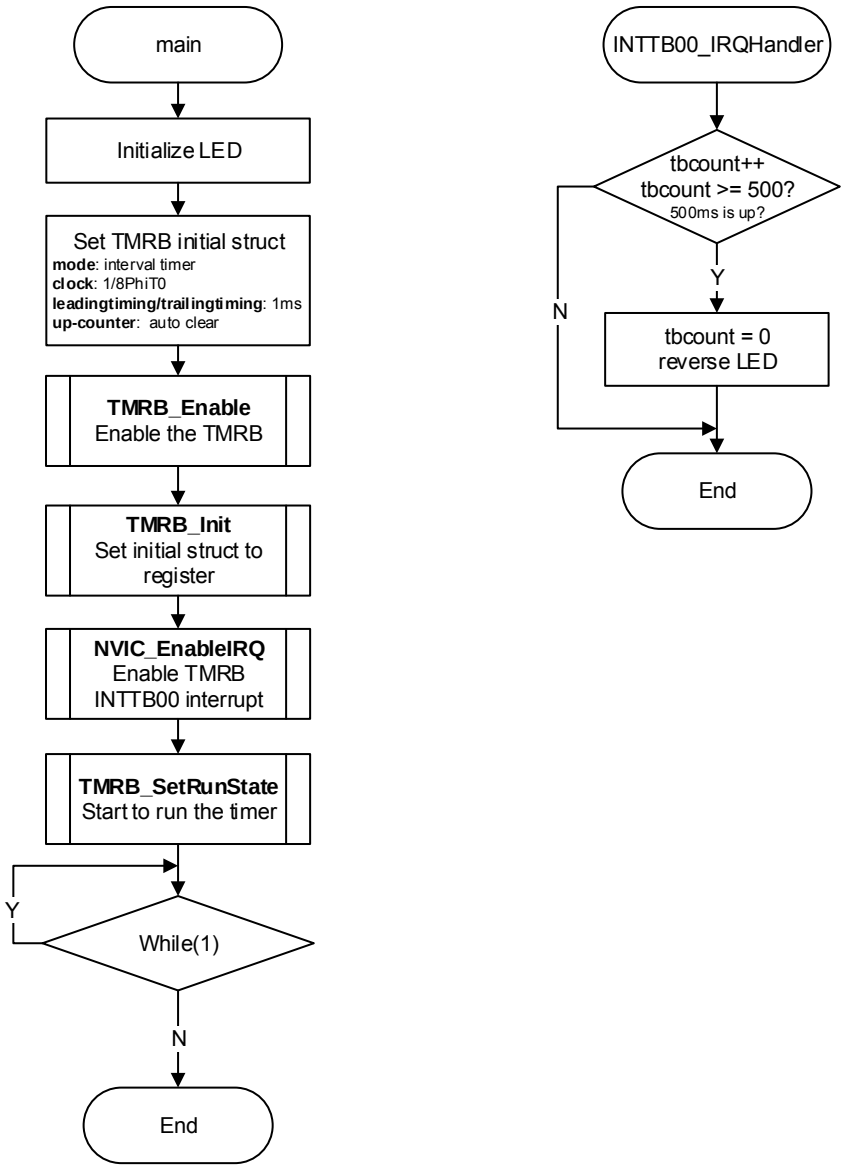
7-9-1 Example: General Timer

This is a simple example based on the TX03 Peripheral Driver (TMRB, GPIO).

The example includes:

1. TMRB0 initialization
2. General Timer for 1ms

- **Flowchart**



- **Code and Explanation for the Example**

At first, initialize LED channel on Eval board and turn on LED.

```
LED_Init();                                    /* LED initialize */
LED_On(LED_ALL);                               /* Turn on LED_ALL */
```

Prepare TMRB initialization structure, fill TMRB mode, clock, up-counter clear method, trailing timing and leading timing. This demo will set trailing timing and leading timing to 1ms. The value of trailing timing and the leading timing will calculate by the function Tmrb_Calculator.

```
TMRB_InitTypeDef m_tmrb;

m_tmrb.Mode = TMRB_INTERVAL_TIMER;      /* internal timer */
m_tmrb.ClkDiv = TMRB_CLK_DIV_8;         /* 1/8PhiT0 */
```

```
/* periodic time is 1ms(require 1000us) */
m_tmrb.TrailingTiming = Tmrb_Calculator(1000U, m_tmrb.ClkDiv);
m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;       /* up-counter auto clear */
/* periodic time is 1ms(require 1000us) */
m_tmrb.LeadingTiming = Tmrb_Calculator(1000U, m_tmrb.ClkDiv);
```

Enable the TMRB module and set the initialization structure to specified registers. Enable INTTB0 interrupt, this interrupt will be triggered every 1ms, in the end, start the TMRB to run.

```
TMRB_Enable(TSB_TB0);                          /* enable the TMRB0 */
TMRB_Init(TSB_TB0, &m_tmrb);                   /* initial the TMRB0 */
NVIC_EnableIRQ(INTTB0_IRQn);                  /* enable INTTB0 interrupt */
TMRB_SetRunState(TSB_TB0, TMRB_RUN);      /* run TMRB0*/
```

Then the main routine will enter “While(1)” to wait the interrupt happens. In interrupt routine, use a counter to count. If 500ms is up, reverse the LED and count again.

```
tbcount++;
if (tbcount >= 500U) {                                  /* 500ms is up */
    tbcount = 0U;
    /* reverse LED output */
    ledon = (ledon == 0U) ? 1U : 0U;
    if (0U == ledon) {
        LED_Off(LED_ALL);
    } else {
        LED_On(LED_ALL);
    }
} else {
    /* Do nothing */
}
```

The function Tmrb_Calculator :

```
uint16_t Tmrb_Calculator(uint16_t Tmrb_Require_us, uint32_t ClkDiv)
{
    uint32_t T0 = 0U;
    const uint16_t Div[8U] = {1U, 2U, 8U, 32U, 64U, 128U, 256U, 512U};

    SystemCoreClockUpdate();

    T0 = SystemCoreClock / (1U << ((TSB_CG->SYSCR >> 8U) & 7U));
    T0 = T0/((Div[ClkDiv])*1000000U);

    return(Tmrb_Require_us * T0);
}
```

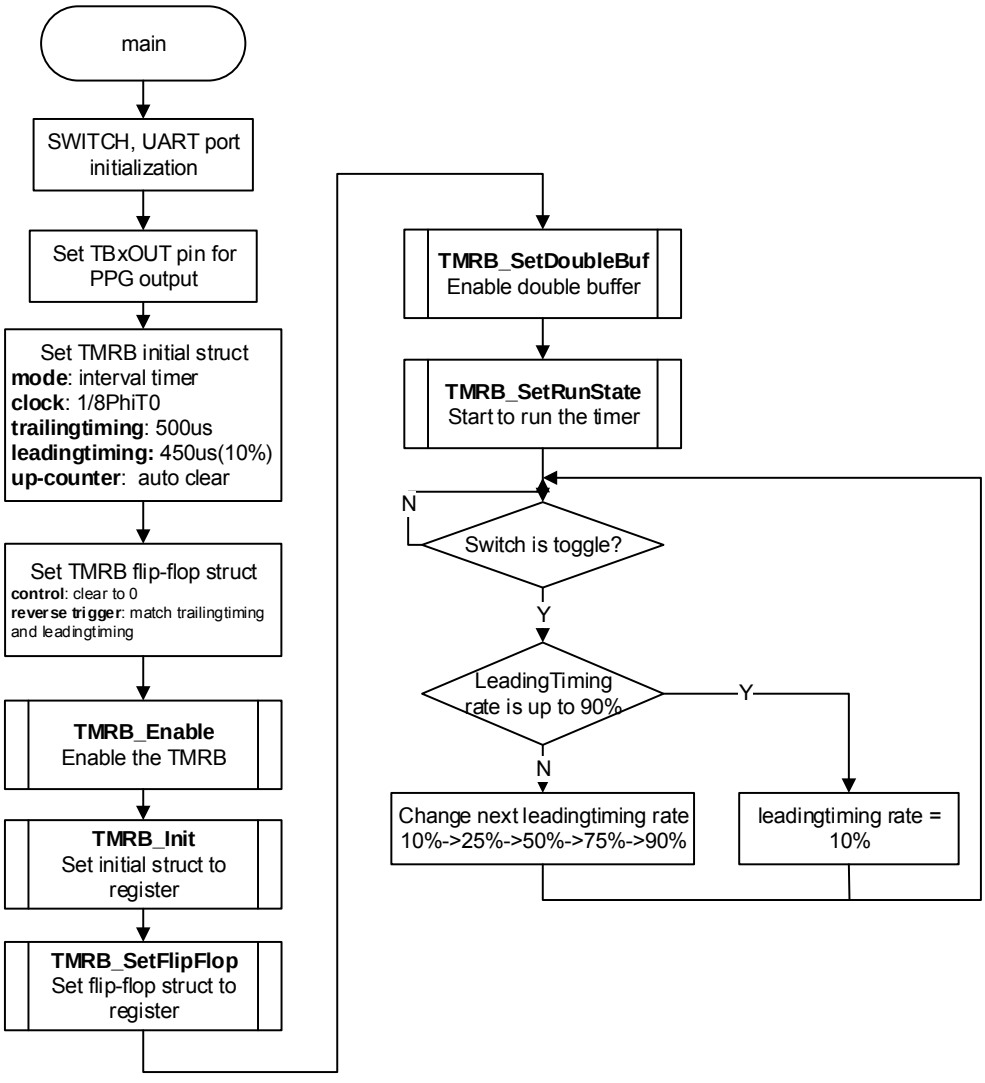
7-9-2 Example: PPG Output

This is a simple example based on the TX03 Peripheral Driver (TMRB, GPIO and UART).

The example includes:

1. TMRB7 initialization
2. PPG process setting and start
3. PPG leading timing adjustment

- **Flowchart**



- **Code and Explanation for the Example**

At first, initialize targeted array tgtLeadingTiming, LeadingTimingus and LeadingTiming. Then initialze SWITCH and UART channel, set PB6 as TB7OUT for

PPG output.

```
    TMRB_InitTypeDef m_tmrb;
    TMRB_FFOutputTypeDef PPGFFInital;
    uint8_t keyvalue;
    uint32_t i = 0U;
    uint32_t tgtLeadingTiming [5U] = { 10U, 25U, 50U, 75U, 90U }; /* leading timing:
10%, 25%, 50%, 75%, 90% */
    uint32_t LeadingTimingus[5U] = {0U};
    uint32_t LeadingTiming[5U] = {0U};

    /* LeadingTimingus: 50, 125, 250, 375, 450 */
    for (i=0U;i<=4U;i++) {
        LeadingTimingus[i] = tgtLeadingTiming[i] * 5U;
    }

    /* UART & switch initialization */
    hardware_init(UART_RETARGET);
    SW_Init();

    /* Set PB6 as TB7OUT for PPG output */
    GPIO_SetOutput(GPIO_PB, GPIO_BIT_6);
    GPIO_EnableFuncReg(GPIO_PB, GPIO_FUNC_REG_3, GPIO_BIT_6);
```

Prepare TMRB initialization structure, and then fill TMRB mode, clock, up-counter clear method, trailing timing and leading timing into it. This demo will set trailing timing to 500us.The value of trailing timing and the leading timing array will calculate by the function Tmrb_Calculator.

```
    TMRB_InitTypeDef m_tmrb;

    m_tmrb.Mode = TMRB_INTERVAL_TIMER;        /* internal timer */
    m_tmrb.ClkDiv = TMRB_CLK_DIV_8;           /* 1/8PhiT0 */
    m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;       /* up-counter auto clear */
    for(i=0U;i<=4U;i++) {
        LeadingTiming[i] = Tmrb_Calculator(LeadingTimingus[i], m_tmrb.ClkDiv);
    }
    m_tmrb.TrailingTiming = Tmrb_Calculator(500U, m_tmrb.ClkDiv);     /* trailing
timing is 500us */
    m_tmrb.LeadingTiming = LeadingTiming[Rate];                        /* leading
timing, initial value 10% */
```

Set flip-flop initialization structure, and fill flip-flop control, reverse trigger parameter into it. Reverse trigger is set to match leading timing and match trailing timing.

```
    PPGFFInital.FlipflopCtrl = TMRB_FLIPFLOP_SET;
    PPGFFInital.FlipflopReverseTrg=TMRB_FLIPFLOP_MATCH_
TRAILINGTIMING | TMRB_FLIPFLOP_MATCH_ LEADINGTIMING;
```

Enable the TMRB module and set the initialization structure and flip-flop structure to specified registers. Enable double buffer, and disable capture function. In the end, start the TMRB to run.

```
TMRB_Enable(TSB_TB7);
TMRB_Init(TSB_TB7, &m_tmrb);
TMRB_SetFlipFlop(TSB_TB7, &PPGFFInital);
/* enable double buffer */
TMRB_SetDoubleBuf(TSB_TB7,ENABLE, TMRB_WRITE_REG_SEPARATE);
TMRB_SetRunState(TSB_TB7, TMRB_RUN);
```

Wait switch from low to high, and at the same time, UART prints current leading timing.

```
do {                            /* wait if switch is Low */
        keyvalue = GPIO_ReadDataBit(KEYPORT, GPIO_BIT_0);
        LeadingTiming_display(); /* display current leading timing */
    } while (GPIO_BIT_VALUE_0 == keyvalue);
```

If the switch is changed to high, change the leading timing according to 10%->25%->50%-> 75%->90%, then from 90% to 10% again.

```
Rate++;
if (Rate >= LEADINGTIMINGMAX) {
    Rate = LEADINGTIMINGINIT;
    } else {
        /* Do nothing */
    }

TMRB_ChangeLeadingTiming(TSB_TB7, LeadingTiming[Rate]);     /* change leading timing rate */
```

The function Tmrb_Calculator :

```
uint16_t Tmrb_Calculator(uint16_t Tmrb_Require_us, uint32_t ClkDiv)
{
    uint32_t T0 = 0U;
    const uint16_t Div[8U] = {1U, 2U, 8U, 32U, 64U, 128U, 256U, 512U};

    SystemCoreClockUpdate();

    T0 = SystemCoreClock / (1U << ((TSB_CG->SYSCR >> 8U) & 7U));
    T0 = T0/((Div[ClkDiv])*1000000U);

    return(Tmrb_Require_us * T0);
}
```

7-10 SIO/UART

7-10-1 Example: Retarget

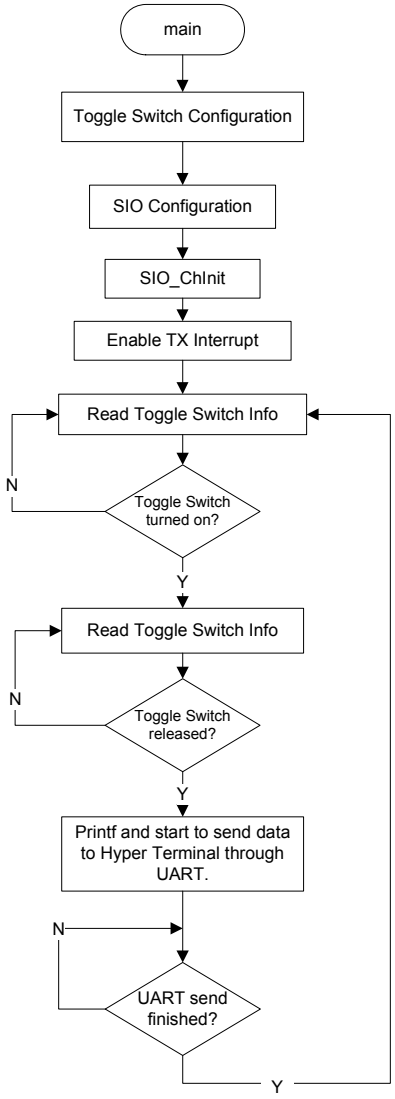
This is a simple example based on the TX03 Peripheral Driver (UART, GPIO).

The example includes:

1. UART configuration and initialization.
2. UART TX process.
3. Use UART TX interrupt to send data.
4. Retarget printf() to UART.

This UART channel is limited by hardware, the follow example used UART0.

- **Flowchart**



- **Code and Explanation for the Example**

At first configure the GPIO and initialize UART.
Use GPIO peripheral drivers configure GPIO for UART.

```
    GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_0, ENABLE);
    GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_0, DISABLE);
    GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_0);
```

Create a UART_InitTypeDef structure and fill all the data fields. For example,

```
    UART_InitTypeDef myUART;

    /* configure SIO0 for reception */
    UART_Enable(UART_RETARGET);
    myUART.BaudRate = 115200U;    /* baud rate = 115200 */
    myUART.DataBits = UART_DATA_BITS_8; /* no handshake, 8-bit data, clock
by baud rate generator */
    myUART.StopBits = UART_STOP_BITS_1; /* 1-bit stop, LSB, W-buff enable */
    myUART.Parity = UART_NO_PARITY;
```

```
    myUART.Mode = UART_ENABLE_TX;
    myUART.FlowCtrl = UART_NONE_FLOW_CTRL;
    UART_Init(UART_RETARGET, &myUART);
```

After above setting, enable UART TX interrupt.

```
    NVIC_EnableIRQ(RETARGET_INT)
```

Then start sending data. Here TxBuffer is a character array.

```
    printf("%s\r\n", TxBuffer);
```

The rest process of data flow is finished in ISR of UART0 TX interrupt routine
UART0 TX interrupt routine:

```
void INTTX0_IRQHandler(void)
{
    if (gSIORdIndex < gSIOWrIndex) {       /* buffer is not empty */
        UART_SetTxData(UART_RETARGET,     gSIOTxBuffer[gSIORdIndex++]);
/* send data */
        fSIO_INT = SET;              /* SIO0 INT is enable */
    } else {
        /* disable SIO0 INT */
        fSIO_INT = CLEAR;
        NVIC_DisableIRQ(RETARGET_INT);
        fSIOTxOK = YES;
    }
    if (gSIORdIndex >= gSIOWrIndex) {     /* reset buffer index */
        gSIOWrIndex = CLEAR;
        gSIORdIndex = CLEAR;
    } else {
        /* Do nothing */
    }
}
```

Function printf() will call putchar() for IAR Compiler and fputc() for RealView Compiler to output data to UART.

```
#if defined ( __CC_ARM      )          /* RealView Compiler */
struct __FILE {
    int handle;                        /* Add whatever you need here */
};
FILE __stdout;
FILE __stdin;
int fputc(int ch, FILE * f)
#elif defined ( __ICCARM__ )       /*IAR Compiler */
int putchar(int ch)
#endif
{
    return (send_char(ch));
}

uint8_t send_char(uint8_t ch)
{
    while (gSIORdIndex != gSIOWrIndex) {            /* wait for finishing sending */
        /* Do nothing */
    }
    gSIOTxBuffer[gSIOWrIndex++] = ch;     /* fill TxBuffer */
    if (fSIO_INT == CLEAR) {       /* if SIO INT disable, enable it */
        fSIO_INT = SET;              /* set SIO INT flag */
```

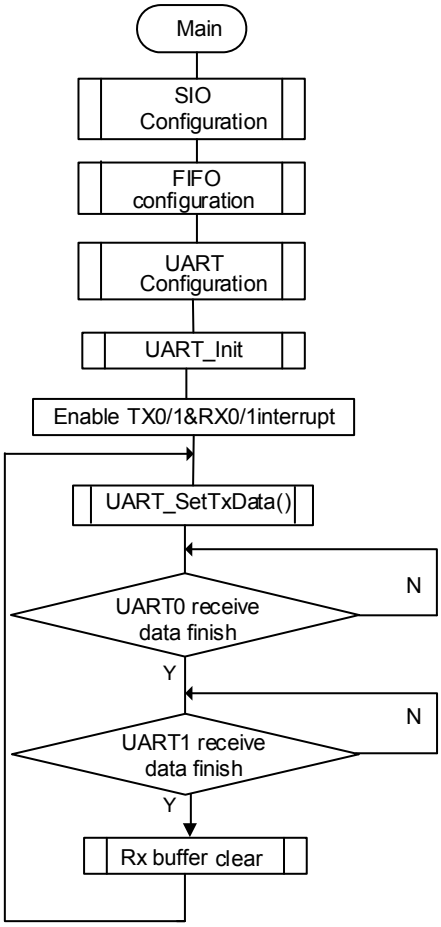
```
            UART_SetTxData(UART_RETARGET, gSIOTxBuffer[gSIORdIndex++]);
            NVIC_EnableIRQ(RETARGET_INT);
        }

        return ch;
}
```

7-10-2 Example: UART FIFO

This is a simple example based on the TX03 Peripheral Driver (UART, GPIO).

The example includes:
1. UART and FIFO configuration and initialization.
2. UART send and receive data use FIFO process.

- **Flowchart**



- **Code and Explanation for the Example**

At first configure the GPIO and initialize UART.

Use GPIO peripheral drivers configure GPIO for UART0 and UART1.

```
void SIO_Configuration(TSB_SC_TypeDef * SCx)
{
	if (SCx == TSB_SC0) {
		TSB_PE->CR |= GPIO_BIT_0;
		TSB_PE->FR1 |= GPIO_BIT_0;
		TSB_PE->FR1 |= GPIO_BIT_1;
		TSB_PE->IE |= GPIO_BIT_1;
	} else if (SCx == TSB_SC1) {
		TSB_PF->CR |= GPIO_BIT_0;
		TSB_PF->FR5 |= GPIO_BIT_0;
		TSB_PB->FR5 |= GPIO_BIT_6;
		TSB_PB->IE |= GPIO_BIT_6;
	}
}
```

Create a UART_InitTypeDef structure and fill all the data fields. For example,

```
	UART_InitTypeDef myUART;

	/* configure SIO0 for reception */
```

```
    UART_Enable(UART_RETARGET);
    myUART.BaudRate = 115200U;   /* baud rate = 115200 */
    myUART.DataBits = UART_DATA_BITS_8; /* no handshake, 8-bit data, clock
by baud rate generator */
    myUART.StopBits = UART_STOP_BITS_1; /* 1-bit stop, LSB,   W-buff enable
*/
    myUART.Parity = UART_NO_PARITY;
    myUART.Mode = UART_ENABLE_TX|UART_ENABLE_RX;
    myUART.FlowCtrl = UART_NONE_FLOW_CTRL;
    UART_Init(UART_RETARGET, &myUART);
```

Use UART peripheral drivers to enable and initialize UART0/1 channels.

```
     UART_Enable(UART0);
     UART_Init(UART0, &myUART);

     UART_Enable(UART1);
     UART_Init(UART1, &myUART);
```

FIFO configuration.

```
    UART_RxFIFOByteSel(UART0,UART_RXFIFO_RXFLEVEL);
    UART_RxFIFOByteSel(UART1,UART_RXFIFO_RXFLEVEL);

    UART_TxFIFOINTCtrl(UART0,ENABLE);
    UART_TxFIFOINTCtrl(UART1,ENABLE);

    UART_RxFIFOINTCtrl(UART0,ENABLE);
    UART_RxFIFOINTCtrl(UART1,ENABLE);

    UART_TRxAutoDisable(UART0,UART_RXTXCNT_AUTODISABLE);
    UART_TRxAutoDisable(UART1,UART_RXTXCNT_AUTODISABLE);

    UART_FIFOConfig(UART0,ENABLE);
    UART_FIFOConfig(UART1,ENABLE);

    UART_RxFIFOFillLevel(UART0, UART_RXFIFO4B_FLEVLE_4_2B);
    UART_RxFIFOFillLevel(UART1, UART_RXFIFO4B_FLEVLE_4_2B);

    UART_RxFIFOINTSel(UART0,UART_RFIS_REACH_EXCEED_FLEVEL);
    UART_RxFIFOINTSel(UART1,UART_RFIS_REACH_EXCEED_FLEVEL);

    UART_RxFIFOClear(UART0);
    UART_RxFIFOClear(UART1);

    UART_TxFIFOFillLevel(UART0, UART_TXFIFO4B_FLEVLE_0_0B);
    UART_TxFIFOFillLevel(UART1, UART_TXFIFO4B_FLEVLE_0_0B);

    UART_TxFIFOINTSel(UART0,UART_TFIS_REACH_NOREACH_FLEVEL);
    UART_TxFIFOINTSel(UART1,UART_TFIS_REACH_NOREACH_FLEVEL);

    UART_TxFIFOClear(UART0);
    UART_TxFIFOClear(UART1);
```

After above setting, enable UART0/1 interrupt.

```
     NVIC_EnableIRQ(INTTX0_IRQn);
     NVIC_EnableIRQ(INTRX1_IRQn);
```

```
        NVIC_EnableIRQ(INTTX1_IRQn);
        NVIC_EnableIRQ(INTRX0_IRQn);
```

The rest process of data flow is finished in ISR of UART0/1 RX and TX interrupt routine

UART0 TX interrupt routine:

```
void INTTX0_IRQHandler(void)
{
    volatile UART_Err err;

    if (TxCounter < NumToBeTx) {
        UART_SetTxData(UART0, TxBuffer[TxCounter++]);
    } else {
        err = UART_GetErrState(UART0);
    }
}
```

UART1 TX interrupt routine:

```
void INTTX1_IRQHandler(void)
{
    volatile UART_Err err;

    if (TxCounter1 < NumToBeTx1) {
        UART_SetTxData(UART1, TxBuffer1[TxCounter1++]);
    } else {
        err = UART_GetErrState(UART1);
    }
}
```

UART0 RX interrupt routine:

```
void INTRX0_IRQHandler(void)
{
    volatile UART_Err err;

    err = UART_GetErrState(UART0);
    if (UART_NO_ERR == err) {
        RxBuffer[RxCounter++] = (uint8_t) UART_GetRxData(UART0);
    }
}
```

UART1 RX interrupt routine:

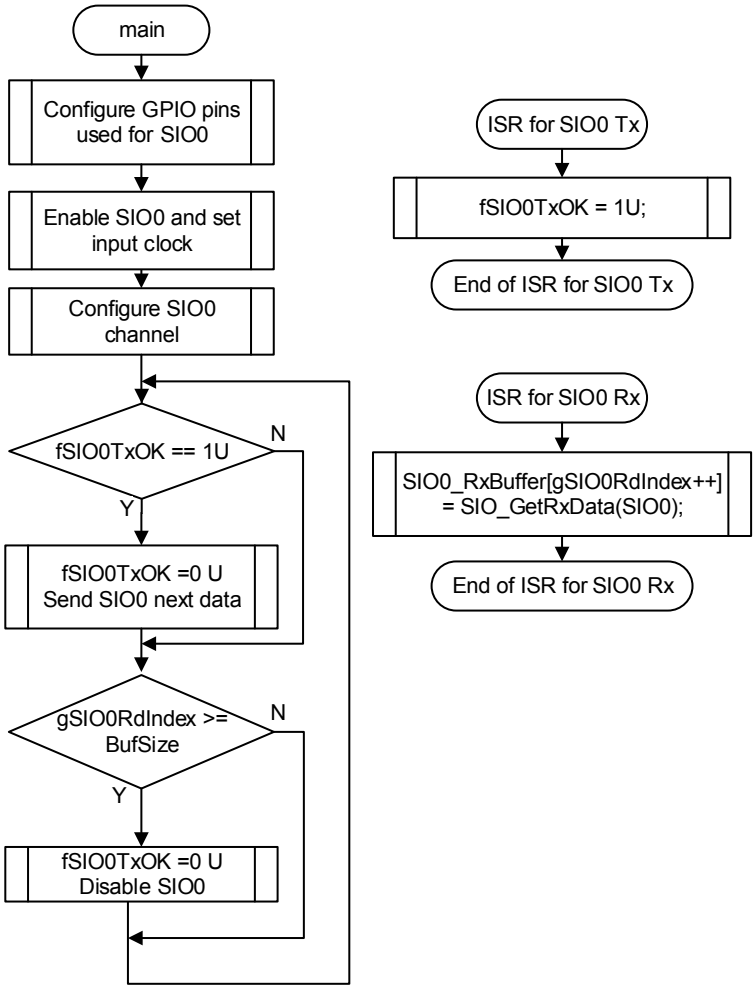
```
void INTRX1_IRQHandler(void)
{
    volatile UART_Err err;

    err = UART_GetErrState(UART1);
    if (UART_NO_ERR == err) {
        RxBuffer1[RxCounter1++] = (uint8_t) UART_GetRxData(UART1);
    }
}
```

7-10-3 Example: SIO Master

This is a simple example based on the TX03 Peripheral Driver (SIO).

The example includes:
1. Basic setup operation of SIO
2. The data transfer between Master SIO0 and Slave SIO0
3. SIO interrupt of Tx and Rx.

- **Flowchart**



- **Code and Explanation for the Example**

At first, configure the GPIO pins for SIO by setting the CR, FR1, IE register of proper port.

Then, enable SIO0 configure the input clock and SIO0 initialization structure.

```
/*Enable the SIO0 channel */
SIO_Enable(SIO0);

/* Selects input clock for prescaler as PhiT0. */
```

```
    SIO_SetInputClock(SIO0, SIO_CLOCK_T0);

    /*initialize the SIO0 struct */
        SIO0_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
        SIO0_Init.TIDLE = SIO_TIDLE_HIGH;
        SIO0_Init.IntervalTime = SIO_SINT_TIME_SCLK_1;
        SIO0_Init.TransferMode = SIO_TRANSFER_FULLDPX;
        SIO0_Init.TransferDir = SIO_LSB_FRIST;
        SIO0_Init.Mode = SIO_ENABLE_TX | SIO_ENABLE_RX;
        SIO0_Init.DoubleBuffer = SIO_WBUF_ENABLE;
        SIO0_Init.BaudRateClock = SIO_BR_CLOCK_TS2;
        SIO0_Init.Divider = SIO_BR_DIVIDER_2;
        SIO_Init(SIO0, SIO_CLK_BAUDRATE, &SIO0_Init);
```

Enable the interrupt of SIO TX, RX.

```
    /* Enable SIO0 Channel TX interrupt */
    NVIC_EnableIRQ(INTTX0_IRQn);
    /* Enable SIO0 Channel RX interrupt */
    NVIC_EnableIRQ(INTRX0_IRQn);
```

After all the basic configure, Enter the data transfer routine .

```
        while (1) {
            /*SIO0 send data from TXD0 */
            if (fSIO0TxOK == 1U) {
                fSIO0TxOK = 0U;
                if (gSIO0WrIndex < BufSize) {
                SIO_SetTxData(SIO0, SIO0_TxBuffer[gSIO0WrIndex++]);
                }
            } else {
                /*Do Nothing */
            }

           /*SIO0 receive data end */
            if (gSIO0RdIndex >= BufSize) {
                SIO_Disable(SIO0);
            } else {
                /*Do Nothing */
            }
        }
```

In ISR of SIO0 Tx, set transfer is ok.

```
void INTTX0_IRQHandler (void)
{
    fSIO0TxOK = 1U;
}
```

In ISR of SIO0 Rx, get the receive data from RX buffer

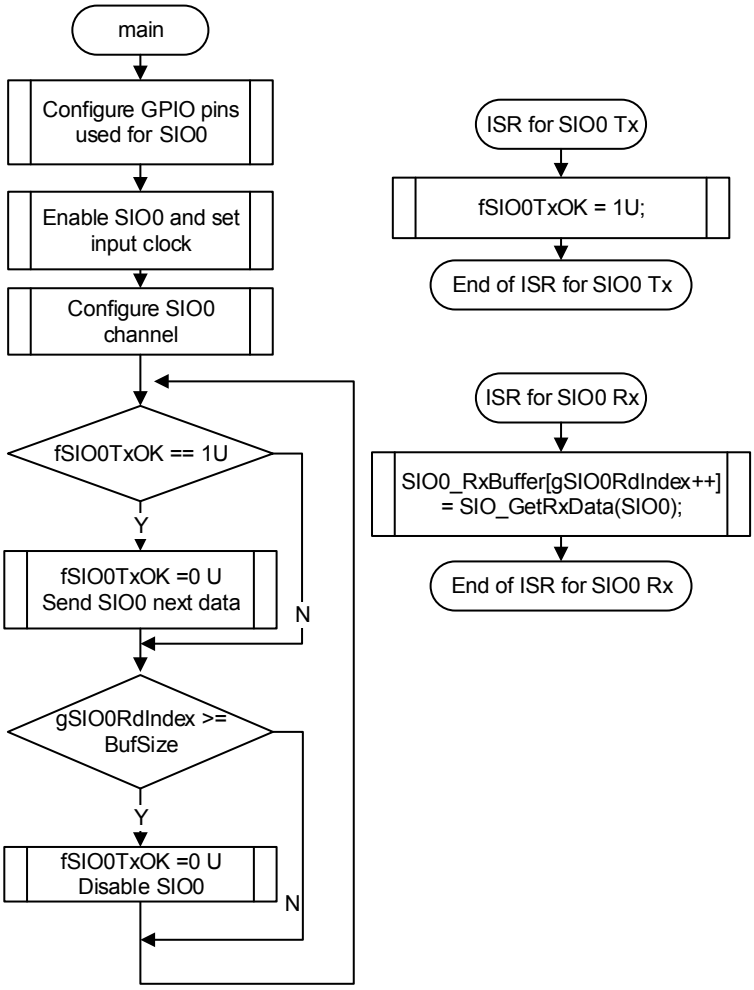
```
void INTRX0_IRQHandler(void)
{
    SIO0_RxBuffer[gSIO0RdIndex++] = SIO_GetRxData(SIO0);
}
```

7-10-4 Example: SIO Slave

This is a simple example based on the TX03 Peripheral Driver (SIO).

The example includes:
1. Basic setup operation of SIO
2. The data transfer between Master SIO0 and Slave SIO0
3. SIO interrupt of Tx and Rx.

- **Flowchart**



- **Code and Explanation for the Example**

At first, configure the GPIO pins for SIO by setting the CR, FR1, IE register of proper port.

Then, enable SIO0 configure the input clock and SIO0 initialization structure.

```
/*configure the SIO0 channel */
SIO_Enable(SIO0);
/* Selects input clock for prescaler as PhiT0. */
SIO_SetInputClock(SIO0, SIO_CLOCK_T0);
```

```
    /*initialize the SIO0 struct */
    SIO0_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
    SIO0_Init.TIDLE = SIO_TIDLE_HIGH;
    SIO0_Init.TransferMode = SIO_TRANSFER_FULLDPX;
    SIO0_Init.TransferDir = SIO_LSB_FRIST;
    SIO0_Init.Mode = SIO_ENABLE_TX | SIO_ENABLE_RX;
    SIO0_Init.DoubleBuffer = SIO_WBUF_ENABLE;
    SIO0_Init.TXDEMP = SIO_TXDEMP_HIGH;
    SIO0_Init.EHOLDTime = SIO_EHOLD_FC_64;
    SIO_Init(SIO0, SIO_CLK_SCLKINPUT, &SIO0_Init);
```

Enable the interrupt of SIO TX, RX.

```
    /* Enable SIO0 Channel TX interrupt */
    NVIC_EnableIRQ(INTTX0_IRQn);
    /* Enable SIO0 Channel RX interrupt */
    NVIC_EnableIRQ(INTRX0_IRQn);
```

After all the basic configure, Enter the data transfer routine .

```
        while (1) {
            /*SIO0 send data from TXD0 */
            if (fSIO0TxOK == 1U) {
                fSIO0TxOK = 0U;
                if (gSIO0WrIndex < BufSize) {
                SIO_SetTxData(SIO0, SIO0_TxBuffer[gSIO0WrIndex++]);
                }
            } else {
                /*Do Nothing */
            }

           /*SIO0 receive data end */
            if (gSIO0RdIndex >= BufSize) {
                SIO_Disable(SIO0);
            } else {
                /*Do Nothing */
            }
        }
```

In ISR of SIO0 Tx, set transfer is ok.

```
void INTTX0_IRQHandler (void)
{
    fSIO0TxOK = 1U;
}
```

In ISR of SIO0 Rx, get the receive data from RX buffer

```
void INTRX0_IRQHandler(void)
{
    SIO0_RxBuffer[gSIO0RdIndex++] = SIO_GetRxData(SIO0);
}
```

7-11 WDT

7-11-1 Example:WDT

This is a simple example based on the TX03 Peripheral Driver (WDT, GPIO).

The example includes:

1. WDT initialization.
2. In DEMO1 WDT isn't cleared before timer overflow, NMI interrupt is generated.
3. In DEMO2 WDT is cleared before timer overflow, the LED0 will blink all the time.

- **Code and Explanation for the Example**

The following code is an example for initializing WDT. Detect time is set to 2^25/fsys, and interrupt will be generated when timer overflow.

```
WDT_InitTypeDef WDT_InitStruct;
WDT_InitStruct.DetectTime = WDT_DETECT_TIME_EXP_25;
WDT_InitStruct.OverflowOutput = WDT_NMIINT;
```

Initialize WDT and enable it.

```
WDT_Init(&WDT_InitStruct);
WDT_Enable();
```

In DEMO1 Waiting for the NMI interrupt flag.

```
while(1)
{
}
```

In DEMO1 When NMI interrupt is occurred the WDT will be disabled and the LED1 blink will be stopped.

```
WDT_Disable();
```

In DEMO2 WDT will be cleared and the LED0 will blink all the time.

```
WDT_WriteClearCode();
```

7-12 PMD

7-12-1 Example: Phase Output

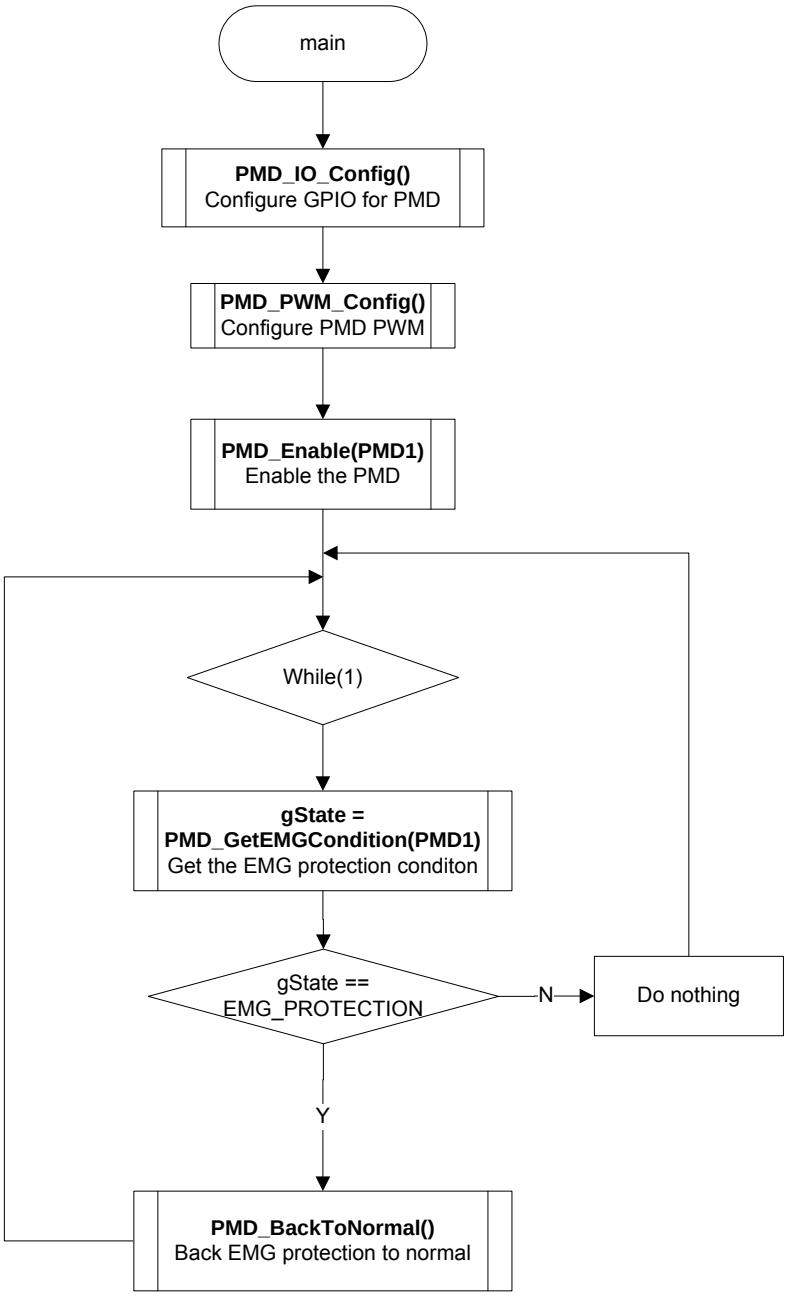
This is a simple example based on the TX03 Peripheral Driver (PMD, GPIO).

The example includes:

1. PMD and GPIO initialization.
2. EMG protection detection.

3. Back EMG protection to normal.

- **Flowchart**



- **Code and Explanation for the Example**

At first, configure the LED and PMD and GPIO.

```
/* LED initialization */
LED4_Init();

/* GPIO configuration*/
PMD_IO_Config();

/* PMD PWM configuration*/
```

```
    PMD_PWM_Config();
```

If the DEMO_U_PHASE is defined, the duty mode is U-phase in common,
If the DEMO_3_PHASE is defined, the duty mode is 3-phase independent.

```
/* PMD1 initialization */
    m_pmd.CycleMode = PMD_PWM_NORMAL_CYCLE;
#ifdef DEMO_U_PHASE
    m_pmd.DutyMode = PMD_DUTY_MODE_U_PHASE;          /* U-phase in
common */
#endif
#ifdef DEMO_3_PHASE
    m_pmd.DutyMode = PMD_DUTY_MODE_3_PHASE;          /* 3-phase
independent */
#endif
    m_pmd.IntTiming = PMD_PWM_INT_TIMING_MINIMUM;
    m_pmd.IntCycle = PMD_PWM_INT_CYCLE_1;
    m_pmd.CarrierMode = PMD_CARRIER_WAVE_MODE_1;     /* PWM mode 1
(center PWM, triangle wave) */
    m_pmd.CycleTiming = 0x3FFFU;

    PMD_Init(PMD1,&m_pmd);
```

Set the different compare value in the 3 phases.
In the duty mode U-phase in common, the outputs are same according to phase U.
In the duty mode 3-phase independent, the outputs are independent according to the compare value.

```
    PMD_SetAllPhaseCompareValue(PMD1,0x0FFFU,0x1FFFU,0x2FFFU);
```

And then enable the PMD.

```
    PMD_Enable(PMD1);
```

Judge the EMG protection condition.If the condition is protection condition, LED4 will light on, and then back EMG protection to normal. If the condition is not protection condition, LED4 will light off.

```
    while(1){
        /* Get the EMG protection condition*/
        gState = PMD_GetEMGCondition(PMD1);
        /* Judge the EMG protection condition */
        if (gState == EMG_PROTECTION) {
            /* LED4 will light on if the condition is protection */
            LED4_On();
            /* Back EMG protection to normal */
            PMD_BackToNormal();
            Delay(1000U);
        } else {
            /* LED4 will light off if the condition is not protection */
            LED4_Off();
        }    }
```

8 Revision History

Revision	Date	Description
1.0	2019-7-17	First release