

# **TOSHIBA**

## **TX03 Peripheral Driver Usage Example (TMPM372)**

Ver 1  
Sep, 2017

**TOSHIBA ELECTRONIC DEVICES & STORAGE CORPORATION**

## **RESTRICTIONS ON PRODUCT USE**

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

## Index

1	General description .....	1
2	Overview .....	1
3	Build-in hardware usage.....	1
4	Pin Usage .....	4
5	Development Environment.....	6
6	Functional description.....	7
6-1	Operation mode.....	7
6-2	ADC .....	7
6-3	CG .....	8
6-3-1	Power mode change .....	8
6-4	FLASH .....	8
6-5	GPIO.....	10
6-6	OFD .....	10
6-7	SIO/UART.....	10
6-7-1	Retarget.....	10
6-7-2	UART FIFO.....	10
6-7-3	SIO .....	11
6-8	TMRB .....	11
6-8-1	General Timer.....	11
6-8-2	PPG Output .....	11
6-9	VLTD.....	11
6-10	WDT .....	11
6-11	ENC .....	12
6-12	PMD.....	12
6-12-1	Example: Phase Output .....	12
6-13	TRMOSC.....	13
7	Software.....	14
7-1	ADC .....	17
7-1-1	Example: ADC Data Read.....	17
7-2	CG .....	18
7-2-1	Example: Power mode change .....	18
7-3	FLASH .....	21
7-4	GPIO.....	25
7-5	OFD .....	26
7-6	TMRB .....	29
7-6-1	Example: General Timer .....	29
7-6-2	Example: PPG Output .....	31
7-7	SIO/UART.....	34
7-7-1	Example: Retarget.....	34
7-7-2	Example: UART FIFO .....	37
7-7-3	Example: SIO .....	40
7-8	VLTD.....	43

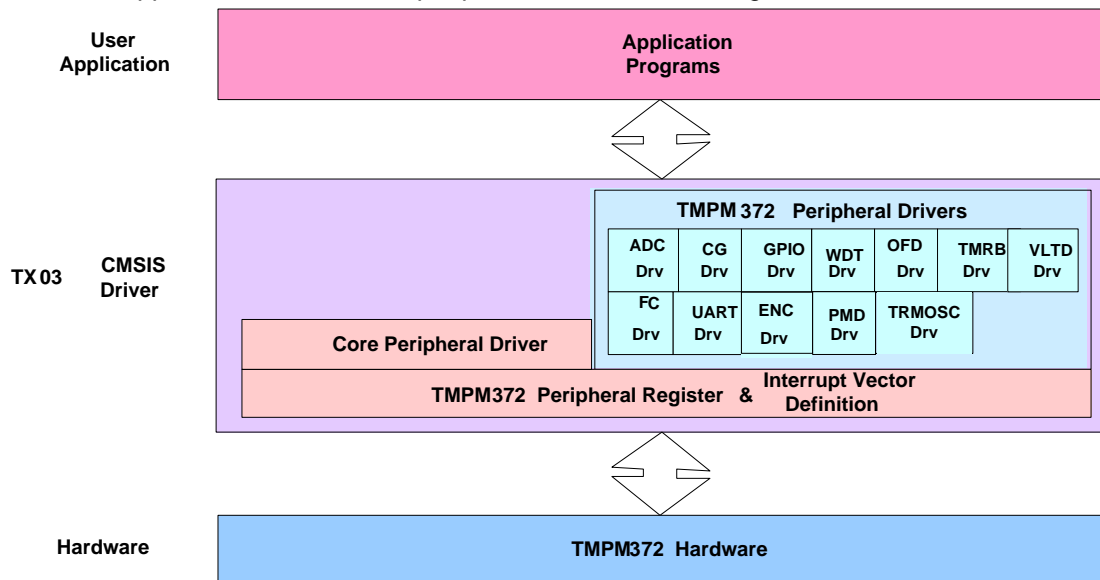
7-8-1	Example: VLTD Reset .....	43
7-9	WDT .....	44
7-10	ENC .....	45
7-10-1	Example: Rolling Detection .....	45
7-11	PMD .....	48
7-11-1	Example: Phase Output .....	48
7-12	TRMOSC .....	51
7-12-1	Example: TRMOSC .....	51

## 1 General description

The example programs described hereafter are specially designed for TOSHIBA TMPM372 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 (8x)
Standby mode	-	Used for CG demo (STOP mode)
SysTick	-	Unused
Watch dog timer ( WDT )	-	Used for WDT
External interrupt ( INT )	INT0	Unused
	INT1	Unused
	INT2	Unused

Hardware	Channel	Use presence, use
	INT3	Unused
	INT4	Unused
	INT5	Unused
	INT6	Used for CG demo to wake up system from stop mode
	INT7	Unused
	INT8	Unused
	INT9	Unused
	INT10	Unused
SIO	SIO0	Used for UART FIFO and SIO demo
	SIO1	Used for UART retarget demo, UART FIFO and SIO demo
	SIO2	Unused
	SIO3	Unused
16-bit timer	TMRB0	Used for TMRB: General Timer
	TMRB1	Unused
	TMRB2	Unused
	TMRB3	Used for TRMOSC: Interval timer Output
	TMRB4	Unused
	TMRB5	Unused
	TMRB6	Used for TMRB: PPG Output Used for TRMOSC: Reference signal input
	TMRB7	Unused
OFD	-	Used for OFD demo
POR	-	Unused
VLTD	-	Used for VLTD demo
Encoder input function	ENC1	Used for ENC Roller demo
Programmable Motor Driver	PMD1	Used for PMD Phase Output demo
12-bit A/D converter	AINB2	Used as POT1(VR1) input
	AINB3	Unused
	AINB4	Unused
	AINB5	Unused
	AINB6	Unused
	AINB7	Unused
	AINB8	Unused
	AINB9	Unused
	AINB10	Unused
	AINB11	Unused
	AINB12	Unused

# TOSHIBA

---

Hardware	Channel	Use presence, use
TRMOSC	-	Used for TRMOSC demo

## 4 Pin Usage

The example programs are tested on TMPM372 evaluation board.

Following is pin usage for example programs TMPM372 evaluation board.

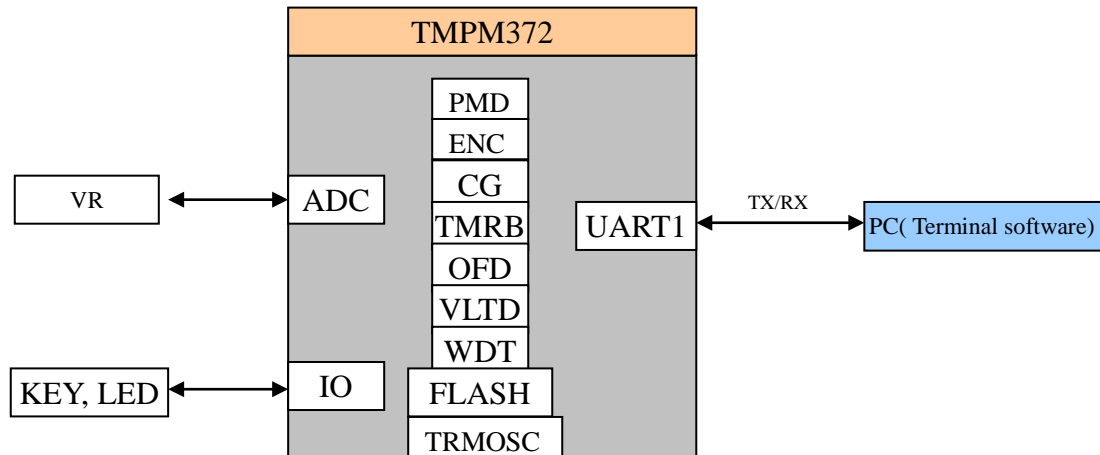
No	Name	Usage
1	SCLK2/CTS2/PD4	Unused
2	TX2/PD5	Unused
3	RX2/PD6	Unused
4	U01/PG0	LED0 / PMD UO1 output
5	X01/PG1	LED1 / PMD XO1 output
6	V01/PG2	LED2 / PMD VO1 output
7	Y01/PG3	LED3 / PMD YO1 output
8	W01/PG4	PMD WO1 output
9	Z01/PG5	PMD ZO1 output
10	EMG1/PG6	PMD EMG1 input
11	OVV1/PG7	Unused
12	DVSS	GND
13	DVDD5	+5V
14	BOOT/TB7IN/PF0	Unused
15	TRACECLK/PB0	Unused
16	TRACEDATA0/PB1	Unused
17	TRACEDATA1/PB2	Unused
18	PB7/TRST	Unused
19	PB6/TDI	Unused
20	PB5/TD0/SWV	Unused
21	PB4/TCK/SWCLK	Unused
22	PB3/TMS/SWDIO	LED4 for PMD demo
23	PE0/TX0	Unused
24	PE1/RX0	Unused
25	PE2/SCLK0/CTS0	Unused
26	PE4/TB2IN/INT5	Unused
27	PF2/ENCA/SCLK3/CTS3	ENCA1
28	PF3/ENCB/TX3	ENCB1
29	PF4/ENCZ/RX3	Unused
30	PF1/TB7OUT	Unused
31	PE5/TB2OUT	Unused
32	PE3/TB4OUT	Unused
33	PA0/INT3/TB0IN	Unused
34	PA1/TB0OUT	Unused

No	Name	Usage
35	PA2/INT4/TB1IN	Unused
36	DVDD5	+5V
37	PM0/X1	Connect 10MHz oscillator
38	DVSS	GND
39	PM1/X2	Connect 10MHz oscillator
40	VOUT15	Unused
41	MODE	Unused
42	RVDD5	+5V, VLTD detected pin (*)
43	VOUT3	Unused
44	RESET	RESET
45	PI3/AINB2	Used as POT1(VR1) input
46	PJ0/AINB3	Unused
47	PJ1/AINB4	Unused
48	PJ2/AINB5	Unused
49	PJ3/AINB6	Unused
50	PJ4/AINB7	Unused
51	PJ5/AINB8	Unused
52	PJ6/AINB9/INTC	SW2/6
53	PJ7/AINB10/INTD	SW3/7
54	PK0/AINB11/INTE	SWS
55	PK1/AINB12/INTF	Unused
56	AVSSB/VREFLB	AGND
57	AVDD5B/VREFHB	A+5V
58	INT7/TB3OUT/PE7	SW1/5, TB3OUT
59	INT6/TB3IN/PE6	SW0/4
60	RX1/TB6IN/PA6	UART: RXD1 (UART1 data receive) TMRB:PPG output TRMOSC: TB6IN
61	TX1/TB6OUT/PA5	TXD1 (UART1 data transfer)
62	SCLK1/CTS1/PA4	Unused
63	INT8/TB4IN/PA7	Unused
64	TB1OUT/PA3	Unused

\* Pins used for VLTD programs on TOSHIBA M372 Evaluation board.

## 5 Development Environment

Following is development environment:



1. Hardware board:

TPM372 evaluation board (Not for SALE)

2. Development tool:

- IAR:
  - 1) J-Link: IAR J-Link-ARM7.0 / J-Link 6.0
  - 2) IDE: IAR Embedded workbench 6.20 version
- KEIL:
  - 1) IDE: KEIL uVision 4.14

**\*Note1:** For ENC&PMD module the development tool version is different

- IAR:
  - 1) J-Link: IAR J-Link-ARM7.0 / J-Link 6.0
  - 2) IDE: IAR Embedded workbench 6.50.1 version
- KEIL:
  - 1) IDE: KEIL uVision 4.60

**\*Note2:** For TRMOSC module the development tool version is different

- IAR:
  - 1) J-Link: IAR J-Link-ARM 7.0 / J-Link 6.0
  - 2) IDE: IAR Embedded workbench 7.10.3 version
- KEIL:
  - 1) IDE: KEIL uVision 5.10

## 6 Functional description

### 6-1 Operation mode

There are three operation modes for TPM372: 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 mode are low power mode.

To shift to lower power mode, in system control register CGSTBYCR<STBY2:0>, select the IDLE, or STOP mode, and run the WFI (Wait For Interrupt) command.

**Note:** Only STOP mode is demonstrated in driver example.

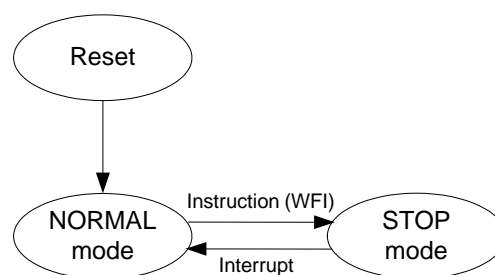
➤ **STOP mode:**

All the internal circuits including the internal oscillator are brought to a stop in STOP mode.

By releasing the STOP mode, the device returns to the preceding mode of the STOP mode and starts operation.

The STOP mode enables to select the pin status by setting the CGSTBYCR<DRVE>. Only external interrupt can release the STOP mode.

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



### 6-2 ADC

Change the value of VR1, which is connected to PI3/AINB2. 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 LEDs loop blink.

## 6-3 CG

### 6-3-1 Power mode change

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

Current mode	Action (Key)	Operation	LED display
NORMAL	Press SW1	NORMAL → STOP	LED off
STOP	Press SW0	STOP → NORMAL	LED on

## 6-4 FLASH

This example demonstrates the feature of flash APIs such as erase and write operation.

The demo runs in Normal mode and User Boot Mode of Single chip mode.

—Reset procedure: includes Mode judgment, Programming routine(flash APIs), Copy routine

- Mode judgment routine: judge to enter User Boot Mode or Normal Mode
- Copy routine: copy programming routine(flash APIs) from flash to RAM
- Programming routine: runs at RAM and swap Program A and Program B code in flash

—Program A/B (Reset procedure) are programmed in flash block in advance.

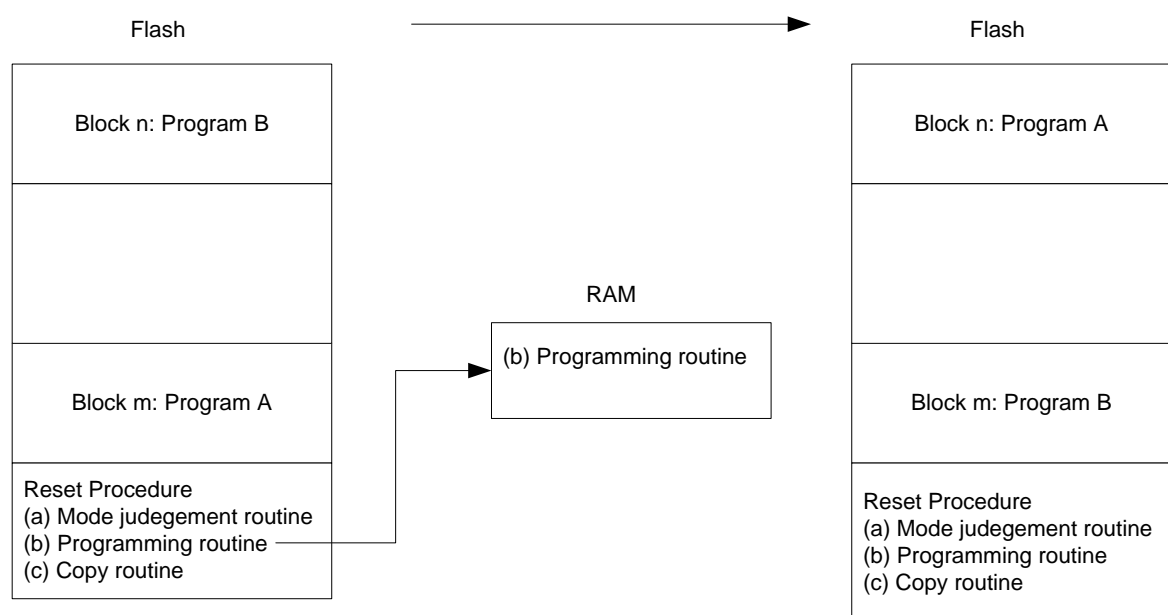
—Program A/B(A: LED1 blinks, B: LED2 blinks)

By default, the program A will run firstly

—KEY SW1 is used for mode judgment in Reset procedure

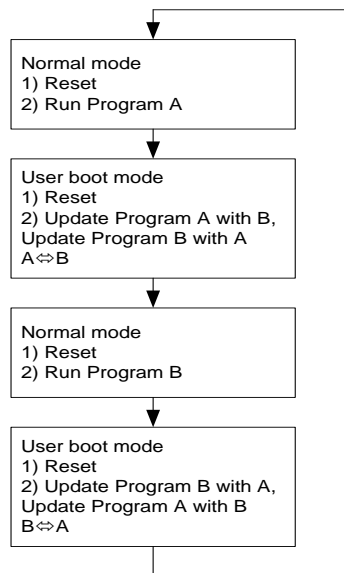
KEY SW1 pressed → User boot mode

KEY SW1 un-pressed → Normal mode

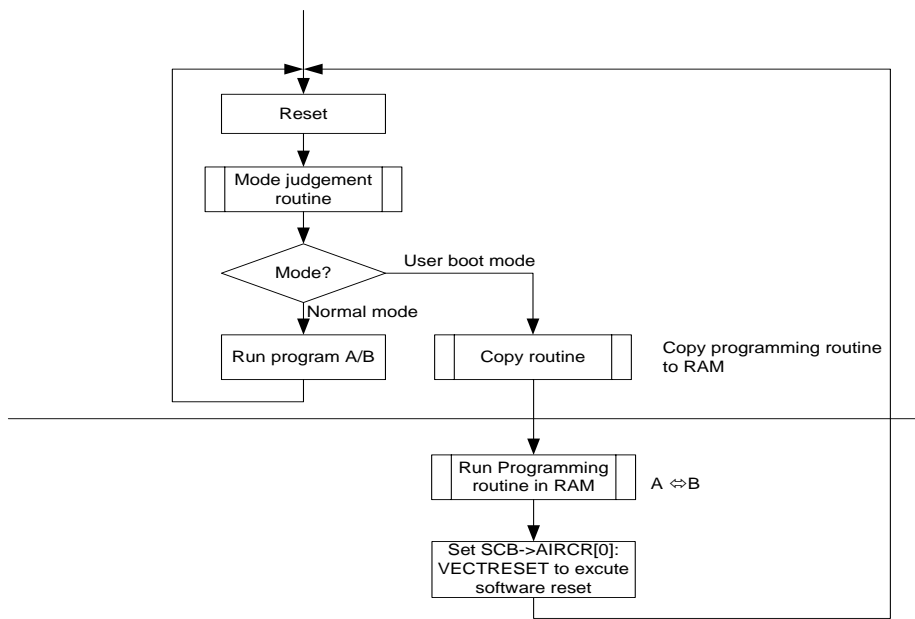


## Demo sequence

- (1) Power on  
The demo board runs Reset Procedure.  
Then initial program A stored in flash runs.  
LED1 blinks.
- (2) Press RESET button while pressing KEY SW1 button, and release KEY SW1 until LED3 turns on.  
The demo board runs Reset Procedure: Programming routine is copied to RAM by Copy routine.  
Then run Programming routine to swap program A and B in flash.
- (3) After a while, LED3 turns off, it indicates that the swap is finished.  
Then the demo will reset by software automatically.  
Then program B stored in flash runs.  
LED2 blinks.



## • Demo process flow



## 6-5 GPIO

This example uses GPIO to control LED and switch on board.

## 6-6 OFD

This example shows how to initialize the OFD. LED will show out the status of the system clock.

## 6-7 SIO/UART

### 6-7-1 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.

### 6-7-2 UART FIFO

In this sample program, UART0 sent the data "TMPM3721" to UART1 use FIFO, and at same time UART0 receive the data "TMPM3722" 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 = "TMPM3722", and RxBuffer1 = "TMPM3721".

## 6-7-3 SIO

This demo will show synchronously transfer and receive usage of SIO module in TPM372. It uses the channel SIO0, SIO1 and transfers data synchronously between them. (connect TXD0 with RXD1, connect TXD1 with RXD0, connect sclK0 with sclK1)

## 6-8 TMRB

### 6-8-1 General Timer

This function implements a general timer utilizing MCU timer.

Timer trailing timing is set to 1ms.

Use this timer for LED blinking and the period is 1s (500ms ON and 500ms OFF).

### 6-8-2 PPG Output

Use switch 1 to change PPG waveform. Leading timing can be changed as following:

10%, 25%, 50%, 75%, 90%

Power on to enter PPG mode and starts the PPG output.

Change the leading timing every switch changing:

10% --> 25% --> 50% --> 75% --> 90% --> 10%

## 6-9 VLTD

This example implements power supply voltage status detecting by VLTD.

When the power supply voltage is lower than the detection voltage, the MCU resets automatically by hardware and PG0 will be high level.

## 6-10 WDT

The watchdog timer cannot be used in the STOP mode where 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 in which DEMO2 is switched 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.

## 6-11 ENC

This example detects the rolling of mouse wheel by TMPM372 ENC driver.

### Demo procedure:

1. Use a wire to connect the pinA (see diagram 1) of mouse wheel encoder with the pin27 (ENCA1) of TMPM372FWUG MCU board.
2. Use a wire to connect the pinB (see diagram 1) of mouse wheel encoder with the pin28 (ENCB1) of TMPM372FWUG 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 reaches 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 reaches to the minimum, it will back to the maximum speed.

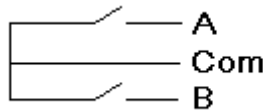


Diagram 1

## 6-12 PMD

### 6-12-1 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. (LED4 will light on)
2. Use a wire to connect the EMG1 (pin 10) with the DVDD5 (pin 13) to pull it high. (LED4 will light off)
3. Connect UO1 (pin 4) to oscilloscope to observe the waveform.
4. Connect XO1 (pin 5) to oscilloscope to observe the waveform.
5. Connect VO1 (pin 6) to oscilloscope to observe the waveform.
6. Connect YO1 (pin 7) to oscilloscope to observe the waveform.
7. Connect WO1 (pin 8) to oscilloscope to observe the waveform.

8. Connect ZO1 (pin 9) 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.

## 6-13 TRMOSC

Using the internal oscillation adjustment function (TRMOSC) to adjust the internal oscillator.

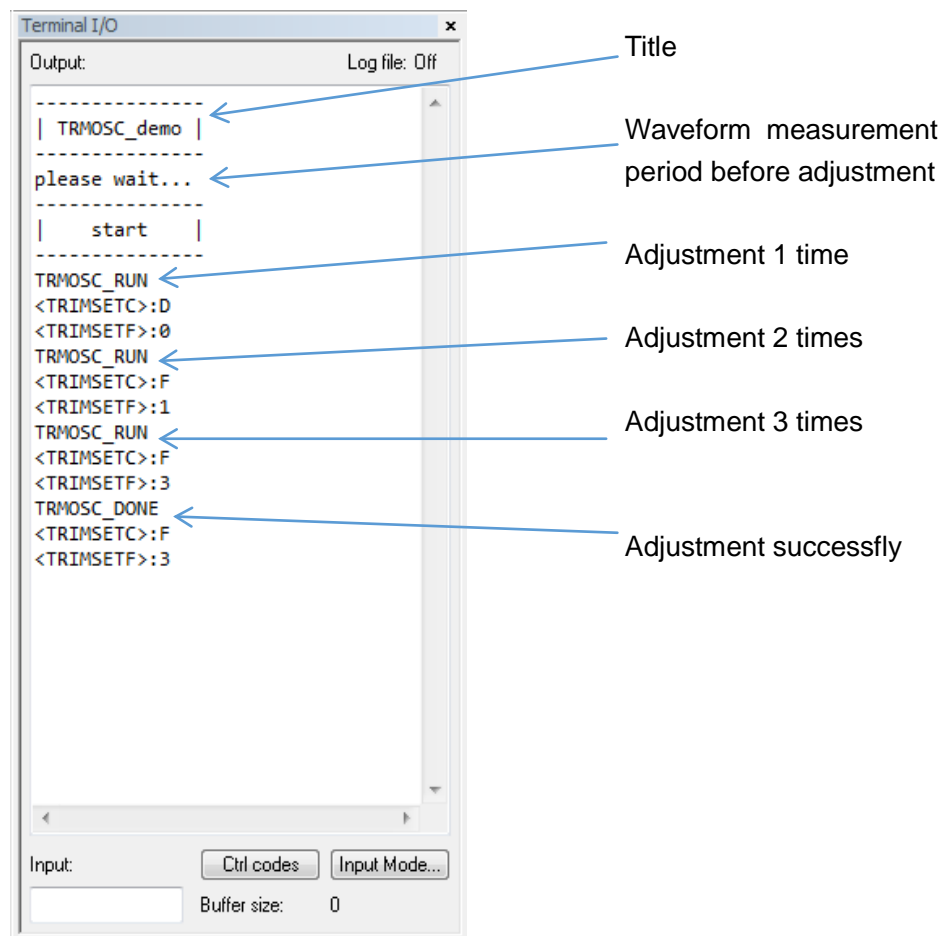
Reference signal is input from the outside to TB6IN.

Parameter signals:

Baudrate	2400bps
Data	8bit
Parity	None
Stop bit	1
Transmit data	0xF0
Frequency	240 Hz

Calculation error, set the adjustment value in the internal oscillation adjustment register.

Use EWARM the terminal I / O functions, output adjustment results.



Monitor the output waveform TB3OUT and confirm.

Output signal specification:

Adjusted number of frequency: 2.375MHz.

Frequency calculation is as follow:

Internal Oscillation	: 9.5MHz
Source clock $\phi$ T1	: $9.5\text{MHz}/2 = 4.75\text{MHz}$
In order to reverse the TB3OUT in 4.75MHz	: $4.75\text{MHz}/2 = 2.375\text{MHz}$

## 7 Software

This software project creates the sample applications based on TMPM372 evaluation board which will demonstrate the main feature of TMPM372 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
        |   tmpm372_wdt_int.c
        |   tmpm372_wdt_int.h
        |
    └─TX03_CMSIS
        |   |   system_TMPM372.c
        |   |   system_TMPM372.h
        |   |   TMPM372.h
        |   |
        |   └─startup
        |       |   startup_TMPM372.s
        |
    └─TX03_Periph_Driver
        |   └─inc
        |       |   tmpm372_cg.h
        |       |   tmpm372_gpio.h
        |       |   tmpm372_wdt.h
        |       |   tx03_common.h
        |       |
        |       └─src
        |           |   tmpm372_cg.c
        |           |   tmpm372_gpio.c
        |           |   tmpm372_wdt.c
        |
    └─TMPM372-EVAL
        |   led.c
        |   led.h
        |   sw.c
        |   sw.h
```

## KEIL MDK:

```
└─WDT_NMI
    └─APP
        |   main.c
        |   tmpm372_wdt_int.c
        |
    └─TX03_CMSIS
        |   |   system_TMPM372.c
        |   |   system_TMPM372.h
        |   |   TMPM372.h
        |   |
```

```
|  └─startup
|      startup_TPM372.s
|
|  └─TX03_Periph_Driver
|      tmpm372_cg.c
|      tmpm372_gpio.c
|      tmpm372_wdt.c
|
|  └─TPM372-EVAL
|      led.c
|      sw.c
```

## 7-1 ADC

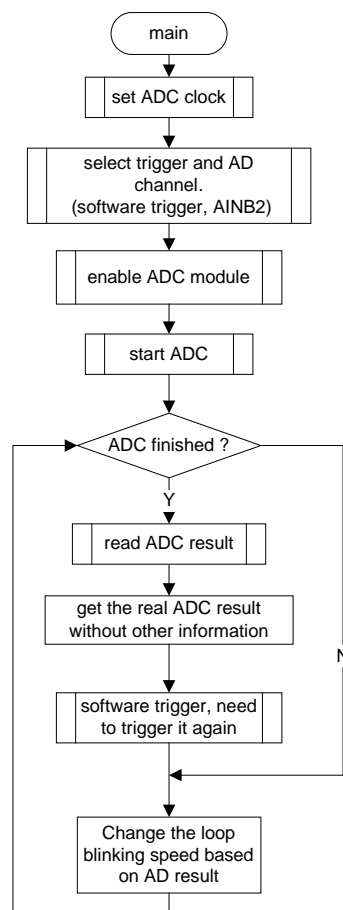
### 7-1-1 Example: ADC Data Read

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

- **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_2);
```

```
/* 2. select trigger and AD channel, this time we use software trigger, */
```

```
/* the VR1 is connected to ADC unit B channel 2, remember to input with macro
TRG_ENABLE() */
ADC_SetSWTrg(TSB_ADB, ADC_REG0, TRG_ENABLE(ADC_AIN2));

/* 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 */
        myResult = result.Bit.ADResult;

        /* 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

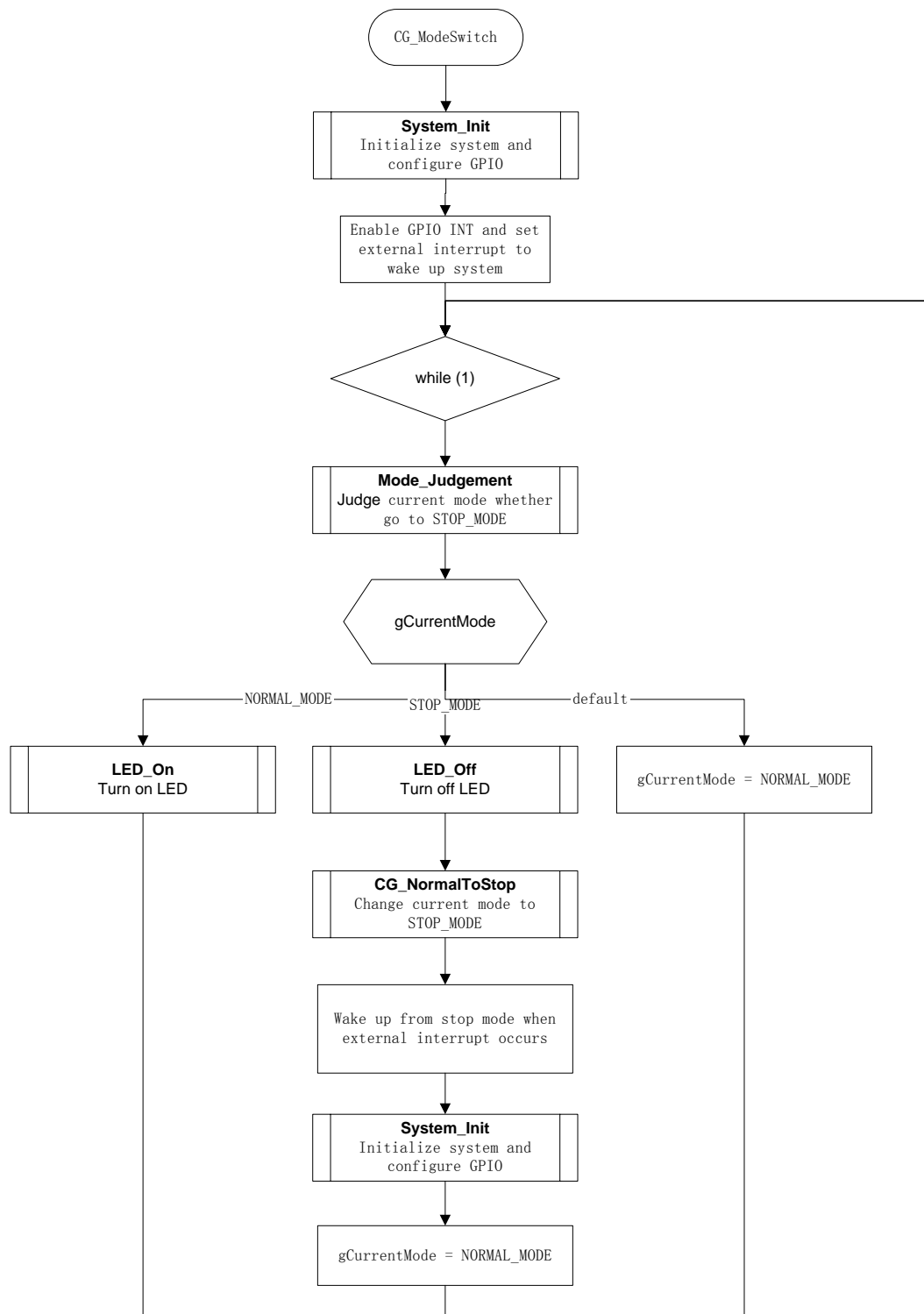
### 7-2-1 Example: Power mode change

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

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, GPIO), 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);
/* select external oscillator */
CG_SetFoscSrc(CG_FOSC_OSC1);
CG_SetPortM(CG_PORTM_AS_HOSC);
/* Enable high-speed oscillator */
CG_SetFosc(CG_FOSC_OSC1, ENABLE);
/* 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();
```

## Setup to enter STOP mode

Prepare for entering stop mode. Set warm up time and external interrupt INT6 to wake up system. Clear interrupt pending request, then enable INT6. Finally use \_\_WFI() instruction to enter stop mode.

```
/* Set CG module: Normal ->Stop mode */
/* Disable interrupts */
__disable_irq();
/* Enable GPIO INT */
CG_ClearINTReq(CG_INT_SRC_6);
/* Set external interrupt to wake up system */
CG_SetSTBYReleaseINTSrc(CG_INT_SRC_6,
                        CG_INT_ACTIVE_STATE_FALLING, ENABLE);
NVIC_ClearPendingIRQ(INT6_IRQn);
NVIC_EnableIRQ(INT6_IRQn);
__enable_irq();

CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC1, CG_WUODR_EXT);
__DSB();
/* Enter stop mode */
__WFI();
```

## Enable multiple clock circuit

Set PLL and set the source of fc.

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

```
WorkState st = BUSY;
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 */  
}
```

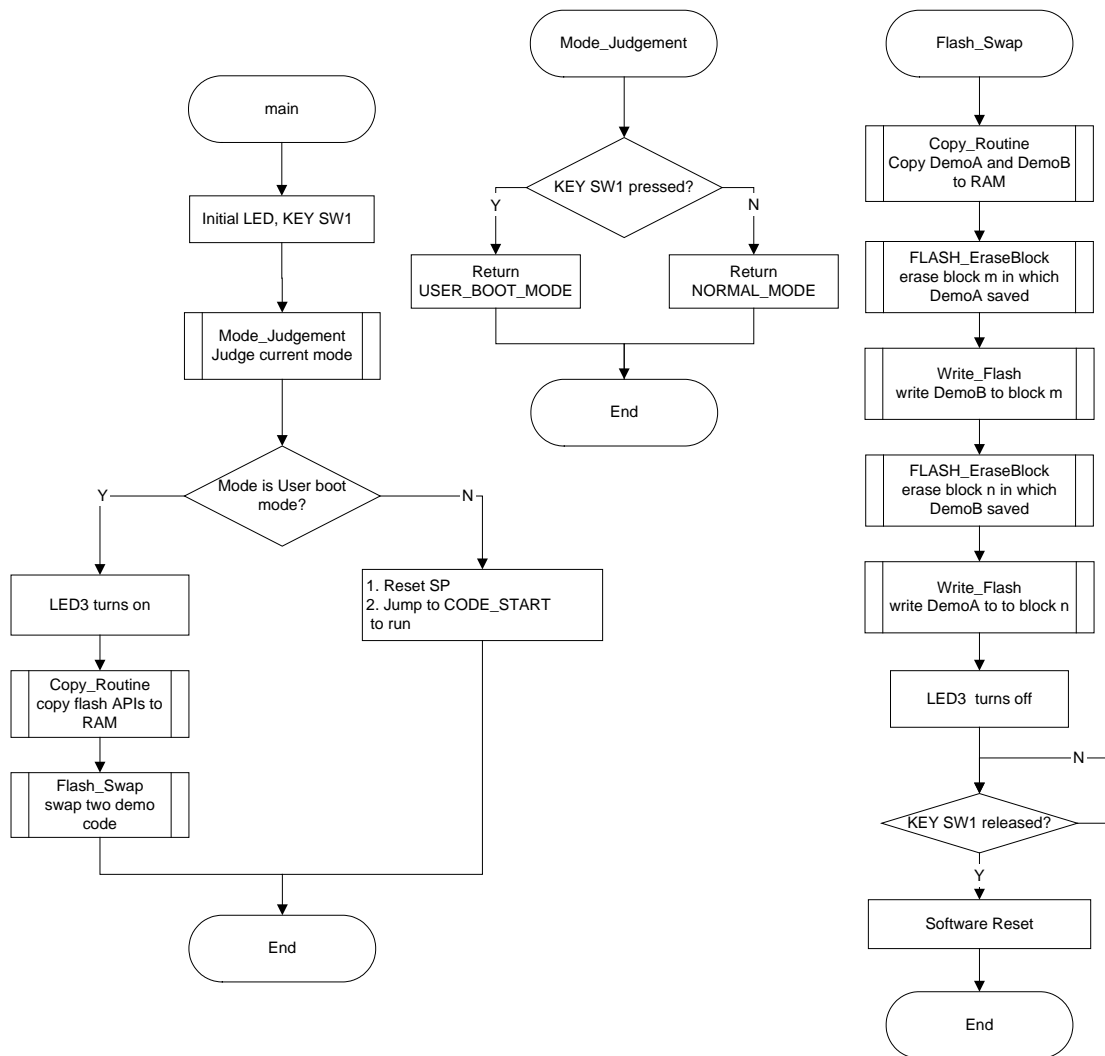
## 7-3 FLASH

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)
2. Operation mode: Single chip mode (Normal mode and User boot mode)
3. Use User boot mode to update code in Flash Memory

- **Flowchart**



- **Code and Explanation for the Example**

At first initialize KEY SW1 and LED. KEY port (GPIO) is used to judge current mode when reset and LED will display current operation information.

```
LED_Init();
SW1_Init();
```

Use function Mode\_Judgement() to judge which mode will be entered after reset.

```
uint8_t Mode_Judgement(void)
{
    return (SW1_Get() == 1U) ? USER_BOOT_MODE : NORMAL_MODE;
}
```

If KEY SW1 is unpressed when reset, the routine will enter normal mode. Then the routine will reset SP and jump to address "CODE\_START" to run. Demo A saved at address "CODE\_START" which belongs to block m, so Demo A will run (LED1 blinks).

```
#if defined ( __CC_ARM )           /* RealView Compiler */
    ResetSP();                     /* reset SP */
#elif defined ( __ICCARM__ )       /* IAR Compiler */
    asm("MOV R0, #0");             /* reset SP */
    asm("LDR SP, [r0]");
#endif
SCB->VTOR = DEMO_START_ADDR; /* redirect vector table */
startup = CODE_START;
startup();                    /* jump to code start address to run */
```

If KEY SW1 is pressed when reset, the routine will enter user boot mode. LED3 will turn on to indicate this mode. Then 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.

```
LED_On(LED3);                    /* enter user boot mode */
Copy_Routine(FLASH_API_RAM, FLASH_API_ROM, SIZE_FLASH_API);
/* copy flash API to RAM */
```

After copying Flash operation APIs to RAM, the routine will jump to function Flash\_Swap(), this function has been copied from Flash memory to RAM by using Copy\_Routine() above.

In function Flash\_Swap(), firstly it will copy Demo A and B from Flash memory to RAM, then call function FC\_EraseBlock () and Write\_Flash() to erase and program Flash memory. The code of Demo A and B will be exchanged in Flash memory.

```
Copy_Routine(DEMO_A_RAM, DEMO_A_FLASH, SIZE_DEMO_A);
/* copy A to RAM */
Copy_Routine(DEMO_B_RAM, DEMO_B_FLASH, SIZE_DEMO_B);
/* copy B to RAM */

/* erase A */
if (FC_SUCCESS == FC_EraseBlock((uint32_t) DEMO_A_FLASH)) {
    /* Do nothing */
} else {
```

```
        return ERROR;
    }

    /* write B to A */
    if (FC_SUCCESS == Write_Flash(DEMO_A_FLASH, DEMO_B_RAM,
SIZE_DEMO_B)) {
        /* Do nothing */
    } else {
        return ERROR;
    }

    /* erase B */
    if (FC_SUCCESS == FC_EraseBlock((uint32_t) DEMO_B_FLASH)) {
        /* Do nothing */
    } else {
        return ERROR;
    }

    /* write A to B */
    if (FC_SUCCESS == Write_Flash(DEMO_B_FLASH, DEMO_A_RAM,
SIZE_DEMO_A)) {
        /* Do nothing */
    } else {
        return ERROR;
    }
}
```

LED3 turns off to indicate the swap operation has completed. After KEY SW1 is released, it will execute software reset by using SCB->AIRCRCR register. Then this demo will run again to enter normal mode, because Demo A and B have been exchanged, Demo B is moved to the position where Demo A was. So Demo B will run (LED2 blinks).

```
delay(0x7FFFFFFF);
LED_Off(LED3);
delay(0x2FFFFFFF);

/* wait for Key SW1 to release */
while (SW1_Get() == 1U) {
}

reg_value = SCB->AIRCRCR; /* software reset */
reg_value = (uint32_t) 0x05FA0001;
SCB->AIRCRCR = reg_value;
```

Flash memory operation function FC\_EraseBlock() will erase a specified block automatically. The block is specified by parameter "BlockAddr". First, this function will check whether the parameter "BlockAddr" 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()) {
    if (!(counter--)) { /* check overtime */
        retval = FC_ERROR_OVER_TIME;
        break;
    } else {
        /* Do nothing */
    }
}
```

Function Write\_Flash() will call FC\_WritePage () to 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++;
}
```

## 7-4 GPIO

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

- 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_6 | GPIO_BIT_7);
GPIO_SetInput(GPIO_PJ, GPIO_BIT_6 | GPIO_BIT_7);
GPIO_SetOutput(GPIO_PK, GPIO_BIT_0);
```

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

Read key by using GPIO\_ReadDataBit(GPIO\_Port GPIO\_x, uint8\_t Bit\_x).

```
if (GPIO_ReadDataBit(GPIO_PE, GPIO_BIT_6) == 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 &= ~led;  
GPIO_WriteData(GPIO_PG, tmp);
```

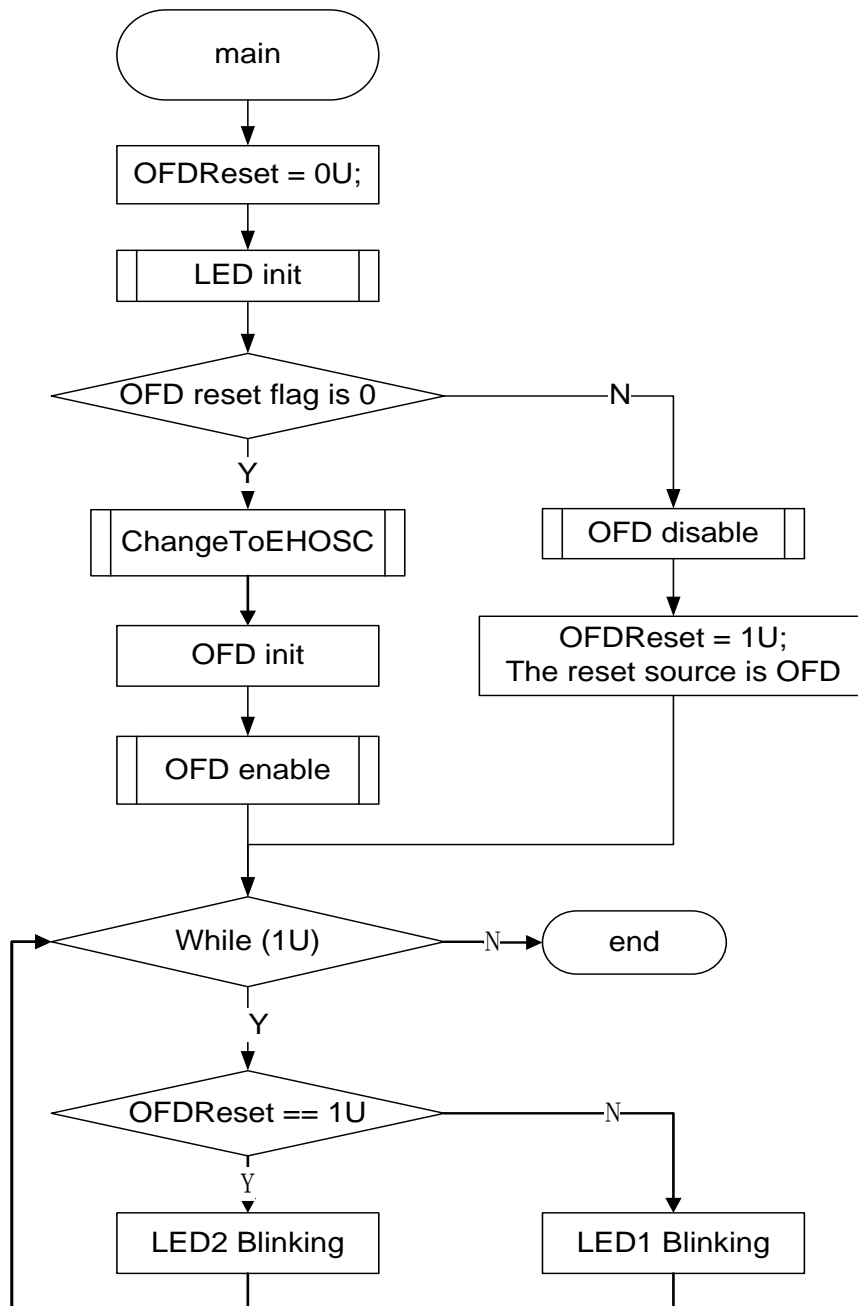
## 7-5 OFD

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 with PLL.

```
void ChangeToEHOSC(void)
{
    /* Use the external oscillation */
    TSB_CG->OSCCR &= OSCCR_WUODR_MASK;
    TSB_CG->OSCCR |= OSCCR_WUODR_EXT;          /*Set the warm up time
WUODR[13:2]*/
    TSB_CG_OSCCR_HOSCON = 1U;    /*Set gpio 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 */
    while (TSB_CG_OSCCR_WUEF) {} /* Warm-up */
    TSB_CG_OSCCR_OSCSEL = 1U;    /*Use the external oscillation */
    TSB_CG_OSCCR_XEN2 = 1U;      /*Enable internal oscillator for ofd */
    TSB_CG_OSCCR_PLLON = 1U;/**/
    TSB_CG_OSCCR_WUEON = 1U;
    while(TSB_CG_OSCCR_WUEF) {    /* Warm-up          */
        /* Do nothing */
    }
    TSB_CG->PLLSEL = PLLSEL_Val;
    TSB_CG->CKSEL  = CKSEL_Val;
}
```

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

```
OFD_SetRegWriteMode(ENABLE);
```

Then set detection frequency with 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_HIGHER_COUNT_ABNORMAL,
                          OFD_LOWER_COUNT_ABNORMAL);
```

Enable OFD after initialization.

```
OFD_Enable();
```

After above setting, OFD will be ready to detect the frequency of FC. 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();  
    }  
}
```

## 7-6 TMRB

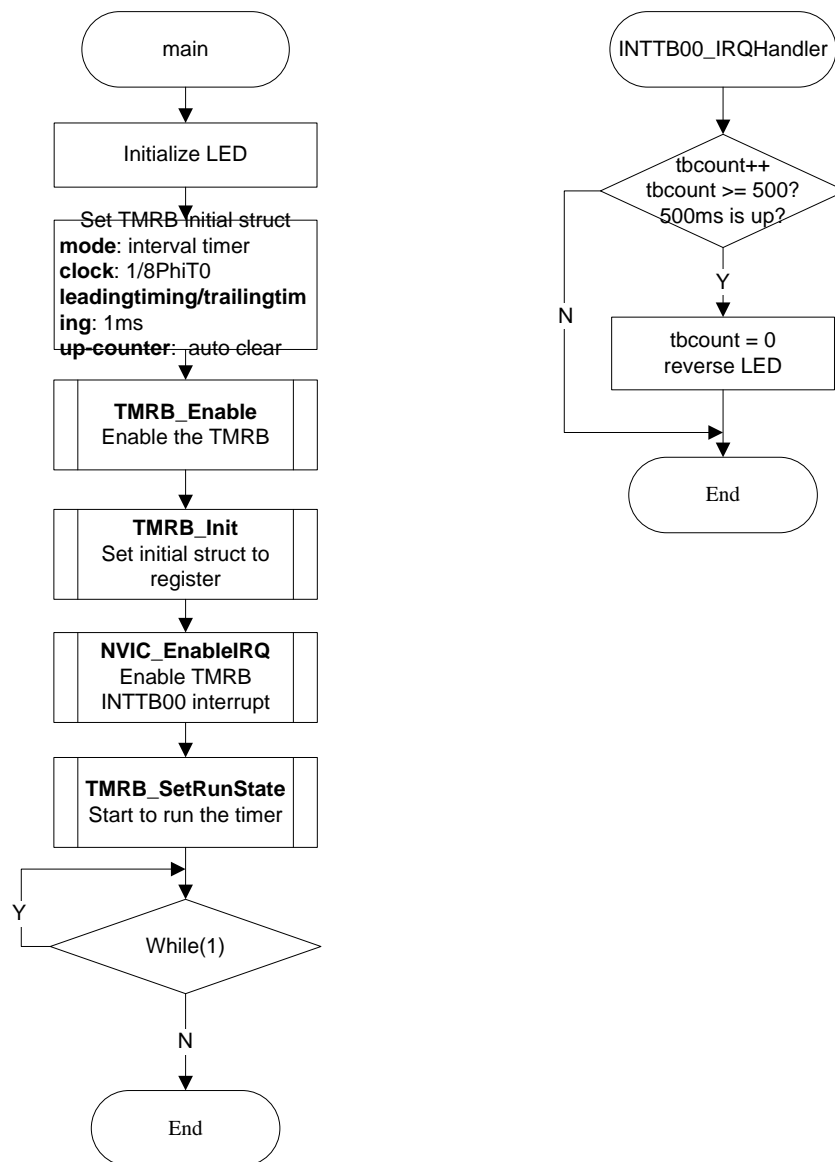
### 7-6-1 Example: General Timer

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

The example includes:

1. TMRB0 initialization
2. General Timer for 1ms

- **Flow Chart**



## • Code and Explanation for the Example

At first, initialize LED on evaluation 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, macro TMRB\_1MS equals 0x2710, because  $f_{\text{phiT0}} = f_{\text{sys}} = f_c = 10\text{MHz} \times \text{PLL} = 80\text{MHz}$ ,  $f_{\text{tmrb}} = 1/8 f_{\text{phiT0}} = 10\text{MHz}$ ,  $T_{\text{tmrb}} = 0.1\mu\text{s}$ ,  $1\text{ms}/0.1\mu\text{s} = 10000 = 0x2710$  (Please see CG part for more detail information about the clock setting)

```
TMRB_InitTypeDef m_tmrb;

m_tmrb.Mode = TMRB_INTERVAL_TIMER; /* internal timer */
m_tmrb.ClkDiv = TMRB_CLK_DIV_8; /* 1/8PhiT0 */
m_tmrb.TrailingTiming = TMRB_1MS; /* periodic time is 1ms */
m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR; /* up-counter auto clear */
m_tmrb.LeadTiming = TMRB_1MS; /* periodic time is 1ms */
```

Enable the TMRB module and set the initialization structure to specified registers. Enable INTTB00 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(INTTB00_IRQn);    /* enable INTTB00 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 */
}
```

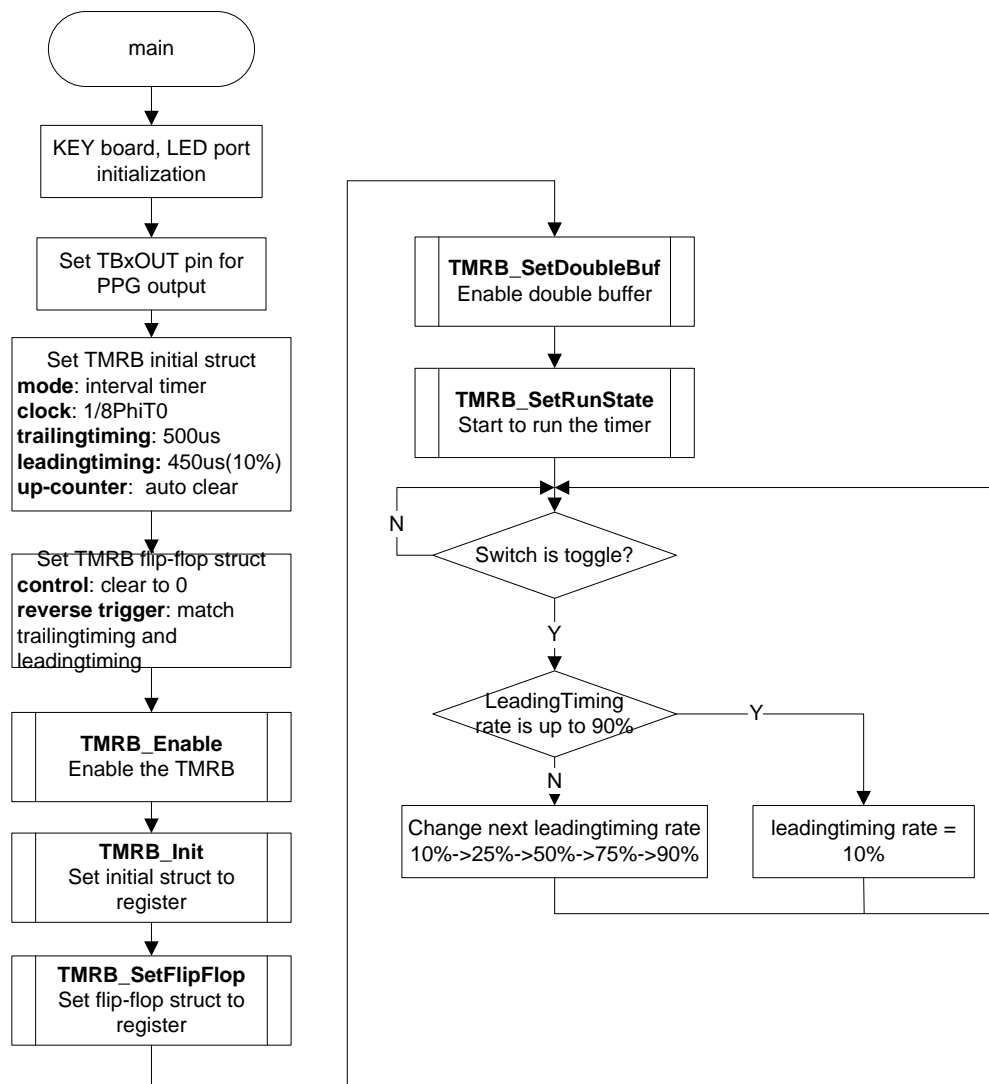
## 7-6-2 Example: PPG Output

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

The example includes:

1. TMRB6 initialization
2. PPG process setting and start
3. PPG leading timing adjustment

- **Flow Chart**



## • Code and Explanation for the Example

At first, initialize KEY and LED, set PA5 as TB6OUT for PPG output.

```

GPIO_SetInput(KEYPORT, GPIO_BIT_6);          /* set KEY port to input */

LED_Init();                                   /* LED initialization*/

/* Set PA5 as TB6OUT for PPG output */
GPIO_SetOutput(GPIO_PA, GPIO_BIT_5);
GPIO_EnableFuncReg(GPIO_PA, GPIO_FUNC_REG_2, GPIO_BIT_5);
  
```

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, macro TMRB6TIME equals 0x09C4, because  $f_{\phi T0} = f_{sys} = f_c = 10\text{MHz} \times \text{PLL} = 80\text{MHz}$ ,  $f_{tmrb} = 1/8 f_{\phi T0} = 10\text{MHz}$ ,  $T_{tmrb} = 0.1\mu\text{s}$ ,  $500\mu\text{s}/0.1\mu\text{s} = 5000 = 0x1388$  (Please see CG part for more detail information about the clock setting)

```

TMRB_InitTypeDef m_tmrb;

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

```
m_tmr.TrailingTiming = TMRB6TIME;          /* trailing timing is 500us */
m_tmr.UpCntCtrl = TMRB_AUTO_CLEAR;         /* up-counter auto clear */
m_tmr.LeadTiming = 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.

```
PPGFFInit.FlipflopCtrl = TMRB_FLIPFLOP_CLEAR;
PPGFFInit.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_TB6);
TMRB_Init(TSB_TB6, &m_tmr);
TMRB_SetFlipFlop(TSB_TB6, &PPGFFInit);
/* enable double buffer */
TMRB_SetDoubleBuf(TSB_TB6,ENABLE, TMRB_WRITE_REG_SEPARATE);
TMRB_SetRunState(TSB_TB6, TMRB_RUN);
```

Wait switch from low to high, and at the same time, display current leading timing on LED.

```
do {
    /* wait if switch is Low */
    keyvalue = GPIO_ReadDataBit(KEYPORT, GPIO_BIT_7);
    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_TB6, LeadingTiming[Rate]); /* change leading
timing rate */
```

The calculation method of leading timing value:

Trailing timing = 500us, ftmr = 1/8 fphiT0 = 10MHz, Ttmr = 0.1us (these time parameters will be different if use different CG setting)

Leading timing = 10%: high-level time is 500\*10% = 50us, low-level time is 500-50 = 450us, counter value = 450us/Ttmr = 0x1194

Leading timing = 25%: high-level time is 500\*25% = 125us, low-level time is 500-125 = 375us, counter value = 375us/Ttmr = 0xEA6

Leading timing = 50%: high-level time is 500\*50% = 250us, low-level time is 500-250 =

250us, counter value =  $250\text{us}/T_{\text{tmrb}} = 0x9C4$

Leading timing = 75%: high-level time is  $500 \times 75\% = 375\text{us}$ , low-level time is  $500 - 375 = 125\text{us}$ , counter value =  $125\text{us}/T_{\text{tmrb}} = 0x4E2$

Leading timing = 90%: high-level time is  $500 \times 90\% = 450\text{us}$ , low-level time is  $500 - 450 = 50\text{us}$ , counter value =  $50\text{us}/T_{\text{tmrb}} = 0x1F4$

That is the calculation method of leading timing array

```
uint32_t LeadingTiming[5] = { 0x1194U, 0xEA6U, 0x9C4U, 0x4E2U, 0x1F4U };  
/* leading timing: 10%, 25%, 50%, 75%, 90% */
```

## 7-7 SIO/UART

### 7-7-1 Example: Retarget

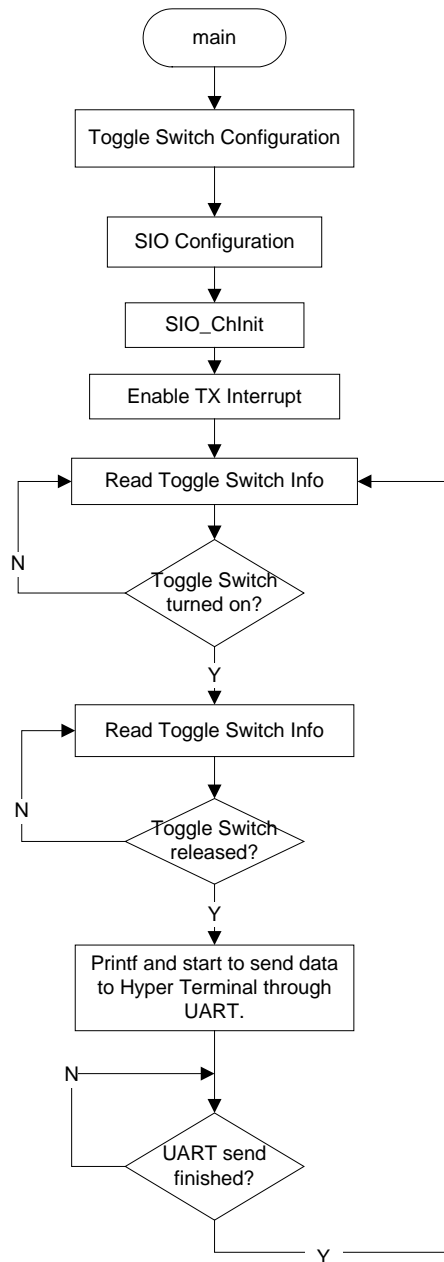
This is a simple example based on the TX03 Peripheral Driver (UART, GPIO).

The example includes:

1. UART configuration and initialization.
2. UART TX process.
3. Use UART0 TX interrupt to send data.
4. Retarget printf() to UART.

This UART channel is limited by hardware, the follow example used UART1.

- 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_PA, GPIO_BIT_5, ENABLE);
GPIO_EnableFuncReg(GPIO_PA, GPIO_FUNC_REG_1, GPIO_BIT_5);
GPIO_EnableFuncReg(GPIO_PA, GPIO_FUNC_REG_1, GPIO_BIT_6);
GPIO_SetInputEnableReg(GPIO_PA, GPIO_BIT_6, ENABLE);
  
```

Create a UART\_InitTypeDef structure and fill all the data fields. For example,

```

UART_InitTypeDef myUART;
  
```

```
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_RX | UART_ENABLE_TX;
myUART.FlowCtrl = UART_NONE_FLOW_CTRL;
```

Use UART peripheral drivers to enable and initialize UART channels.

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

After above setting, enable UART TX interrupt.

```
NVIC_EnableIRQ(INTTX1_IRQn);
```

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

```
UART_SetTxData(UART1, TxBuf[0]);
```

The rest process of data flow is finished in ISR of UART1 TX interrupt routine

UART1 TX interrupt routine:

```
void INTTX1_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(INTTX1_IRQn);
        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(UART0, gSIOTxBuffer[gSIORdIndex++]);
        NVIC_EnableIRQ(INTTX0_IRQn);
    }

    return ch;
}

```

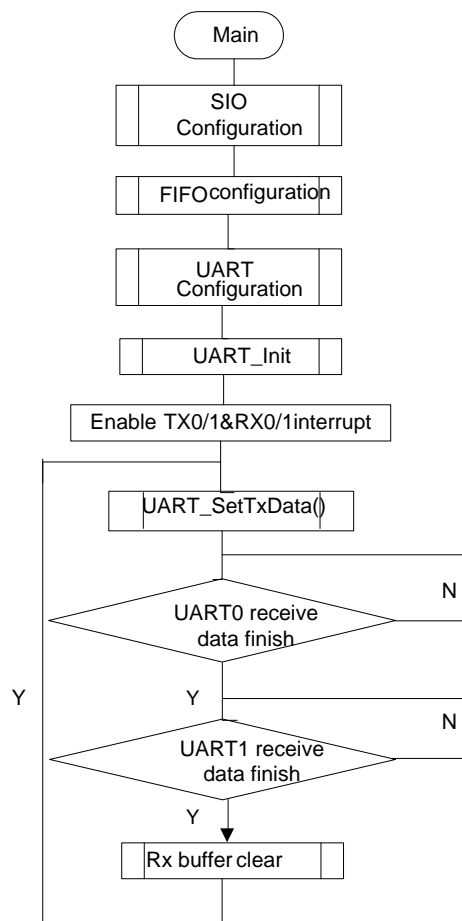
## 7-7-2 Example: UART FIFO

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

The example includes:

UART and FIFO configuration and initialization.  
 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_PA->CR |= GPIO_BIT_5;
        TSB_PA->FR1 |= GPIO_BIT_5;
        TSB_PA->FR1 |= GPIO_BIT_6;
        TSB_PA->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);
    }
}
```

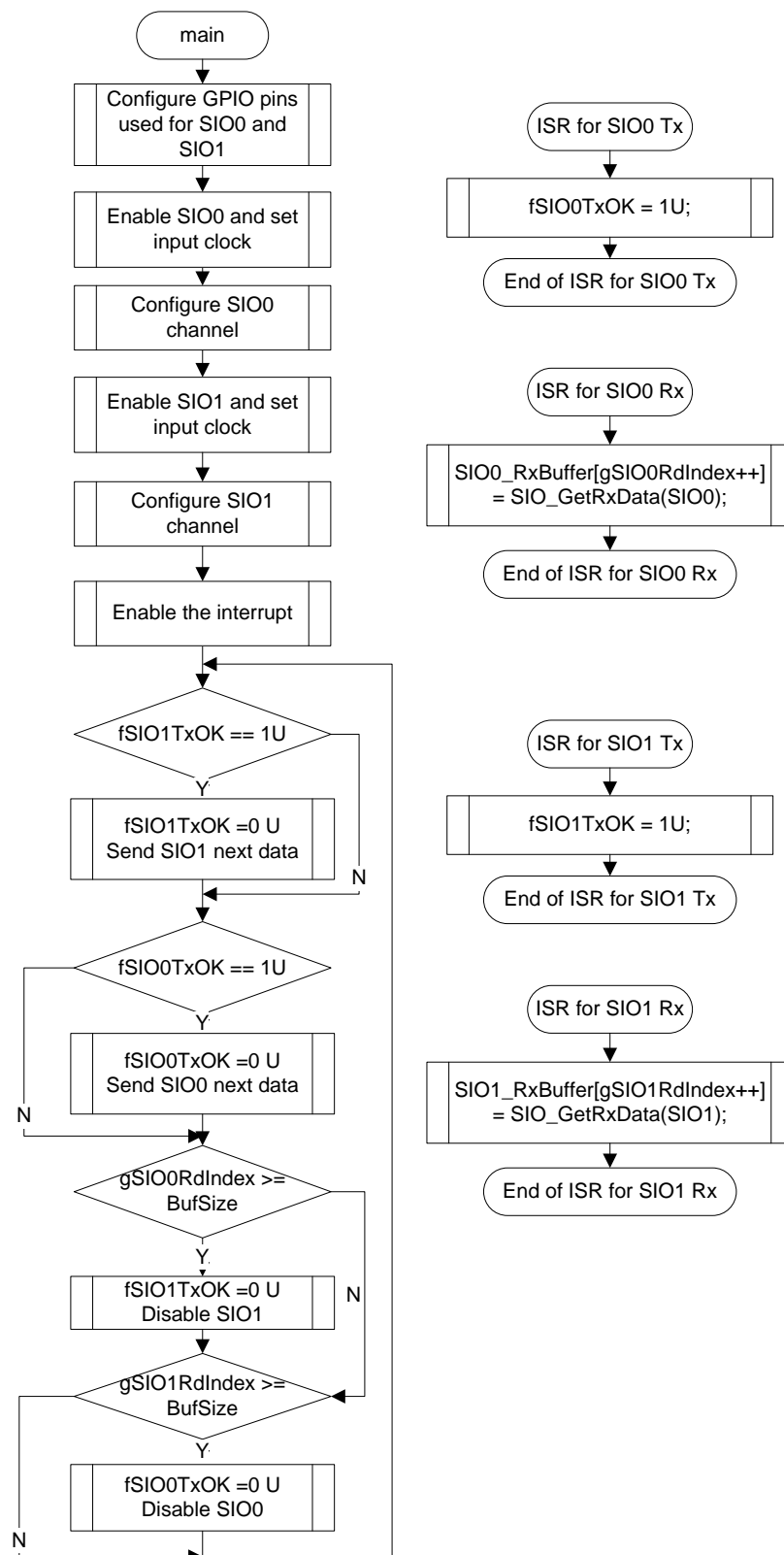
## 7-7-3 Example: SIO

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 SIO0 and SIO1
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);

/*initialize the SIO0 struct */
SIO0_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO0_Init.IntervalTime = SIO_SINT_TIME_SCLK_8;
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_T4;
SIO0_Init.Divider = SIO_BR_DIVIDER_2;
SIO_Init(SIO0, SIO_CLK_BAUDRATE, &SIO0_Init);
```

Enable SIO1 configure the input clock and SIO1 initialization structure.

```
/*Enable the SIO1 channel */
SIO_Enable(SIO1);

/*initialize the SIO1 struct */
SIO1_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO1_Init.TransferMode = SIO_TRANSFER_FULLDPX;
SIO1_Init.TransferDir = SIO_LSB_FRIST;
SIO1_Init.Mode = SIO_ENABLE_TX | SIO_ENABLE_RX;
SIO1_Init.DoubleBuffer = SIO_WBUF_ENABLE;

SIO_Init(SIO1, SIO_CLK_SCLKINPUT, &SIO1_Init);
```

Enable the interrupt of SIO TX, RX.

```
/* Enable SIO0 Channel TX interrupt */
NVIC_EnableIRQ(INTTX0_IRQn);
/* Enable SIO1 Channel RX interrupt */
NVIC_EnableIRQ(INTRX1_IRQn);

/* Enable SIO1 Channel TX interrupt */
NVIC_EnableIRQ(INTTX1_IRQn);
/* Enable SIO0 Channel RX interrupt */
NVIC_EnableIRQ(INTRX0_IRQn);
```

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

```
while (1) {
    /* SIO1 send data from TXD1*/
    if (fSIO1TxOK == 1U) {
        fSIO1TxOK = 0U;
        SIO_SetTxData(SIO1, SIO1_TxBuffer[gSIO1WrIndex++]);
    } else {
        /*Do Nothing */
    }
    /* SIO0 send data from TXD0*/
    if (fSIO0TxOK == 1U) {
        fSIO0TxOK = 0U;
        SIO_SetTxData(SIO0, SIO0_TxBuffer[gSIO0WrIndex++]);
    } else {
        /*Do Nothing */
    }
}

/*SIO0 receive data end */
if (gSIO0RdIndex >= BufSize) {
```

```
fSIO1TxOK = 0U;
SIO_Disable(SIO1);
} else {
    /*Do Nothing */
}
/*SIO1 receive data end */
if (gSIO1RdIndex >= BufSize) {
    fSIO0TxOK = 0U;
    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);
}
```

In ISR of SIO1 Tx, set transfer is ok.

```
void INTTX1_IRQHandler(void)
{
    fSIO1TxOK = 1U;
}
```

In ISR of SIO1 Rx, get the receive data from RX buffer.

```
void INTRX1_IRQHandler(void)
{
    SIO0_RxBuffer[gSIO1RdIndex++] = SIO_GetRxData(SIO1);
}
```

## 7-8 VLTD

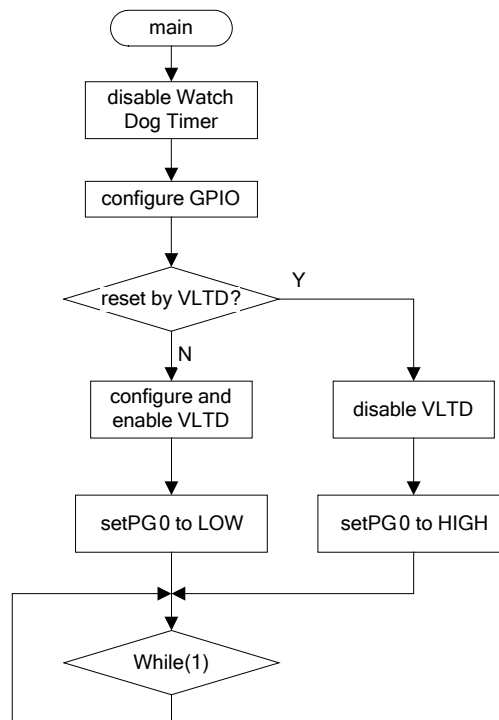
### 7-8-1 Example: VLTD Reset

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, 0x00);
}
```

Otherwise, disable VLTD and set PG0 to HIGH.

```
} else {
    VLTD_Disable();
    GPIO_WriteData(GPIO_PG, 0x01);
}
```

## 7-9 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 LED 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  $2^{25}/f_{sys}$ , 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 LED blink will be stopped.

```
WDT_Disable();
```

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

```
WDT_WriteClearCode();
```

## 7-10 ENC

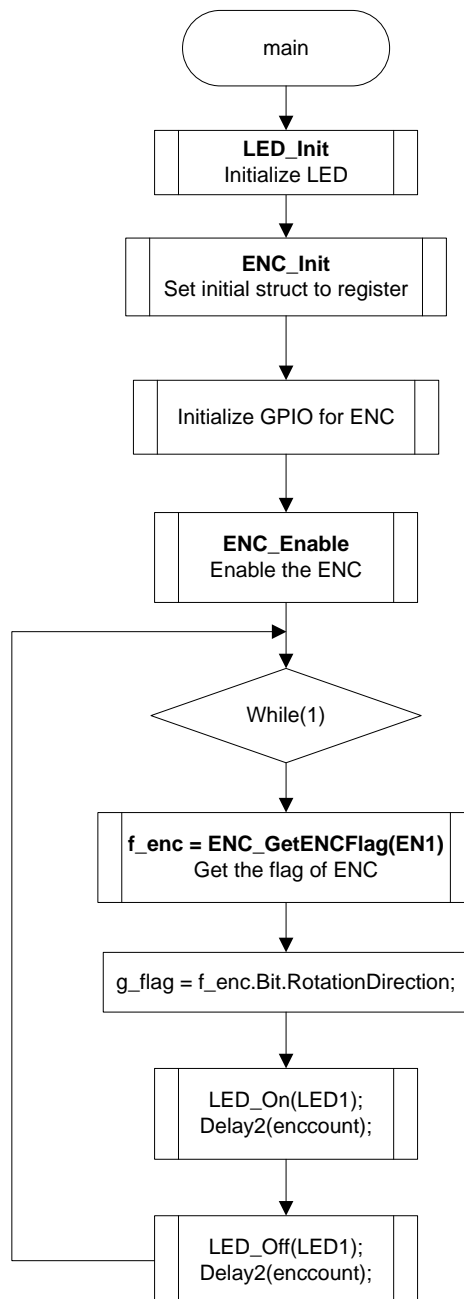
### 7-10-1 Example: Rolling Detection

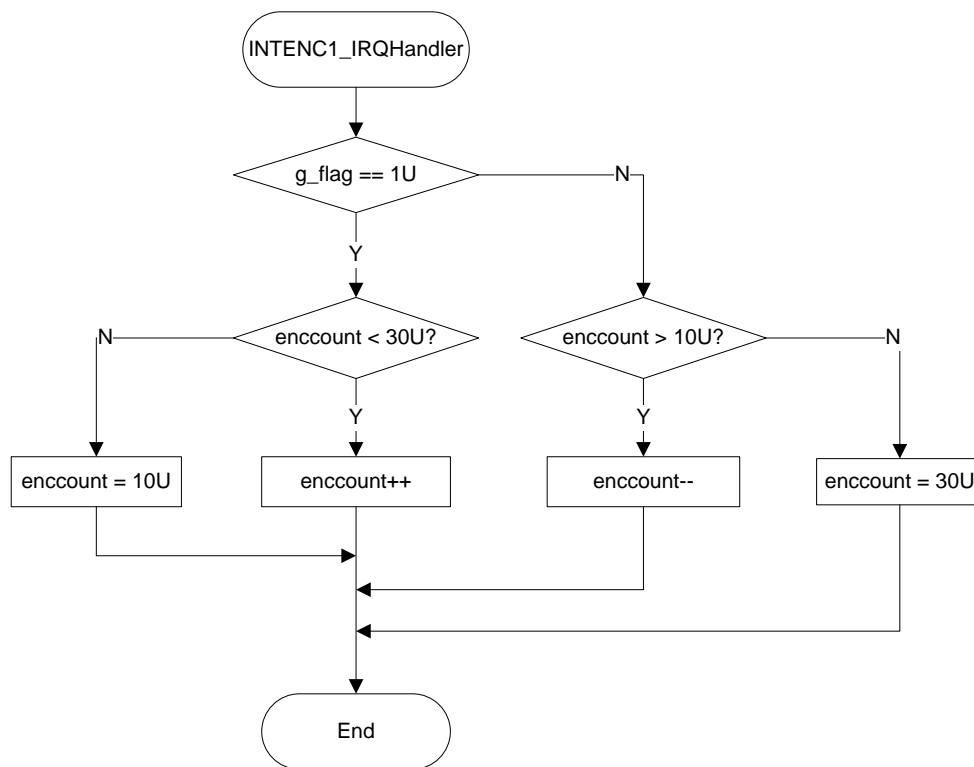
This is a simple example based on the TX03 Peripheral Driver (ENC, GPIO).

The example includes:

1. ENC1 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 LED channel on eval board.

```

/* LED initialization */
LED_Init();

```

Prepare ENC initialization structure, and initialize the ENC1.

```

/* ENC1 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(EN1,&m_enc);
ENC_SetCounterReload(EN1,0xFFFU);

```

Initialize the GPIO for ENC1. Set the PF2 and PF3 as inputs of ENC1.

```

/* GPIO initialization */
GPIO_SetInputEnableReg(GPIO_PF, GPIO_BIT_2, ENABLE); /*Set PF2 as
input */
GPIO_EnableFuncReg(GPIO_PF, GPIO_FUNC_REG_1, GPIO_BIT_2); /*Set
PF2 as ENCA1 */
GPIO_SetInputEnableReg(GPIO_PF, GPIO_BIT_3, ENABLE); /*Set PF3 as
input */
GPIO_EnableFuncReg(GPIO_PF, GPIO_FUNC_REG_1, GPIO_BIT_3); /*Set

```

PF3 as ENCB1 \*/

Enable the ENC1 and ENC1 interrupt.

```
/* Enable ENC1 */
ENC_Enable(EN1);
/* Enable ENC1 interrupt */
NVIC_EnableIRQ(INTENC1_IRQn);
```

Get the status of rotation direction of ENC1 to indicate the rolling direction of the mouse wheel. Blink the LED1 according to the cycle of rolling

```
/* LED1 blink speed based on the rolling of mouse wheel */
while(1){
    f_enc = ENC_GetENCFlag(EN1);
    g_flag = f_enc.Bit.RotationDirection;
    LED_On(LED1);
    Delay2(enccount);
    LED_Off(LED1);
    Delay2(enccount);
}
```

Enccount is the count of rolling, and it changed in the interrupt routine.

While rolling the mouse wheel forward, enccount decreased. When enccount reaches to 10, it will back to 30.

While rolling the mouse wheel backward, enccount added. When enccount reaches to 30, it will back to 10.

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

## 7-11 PMD

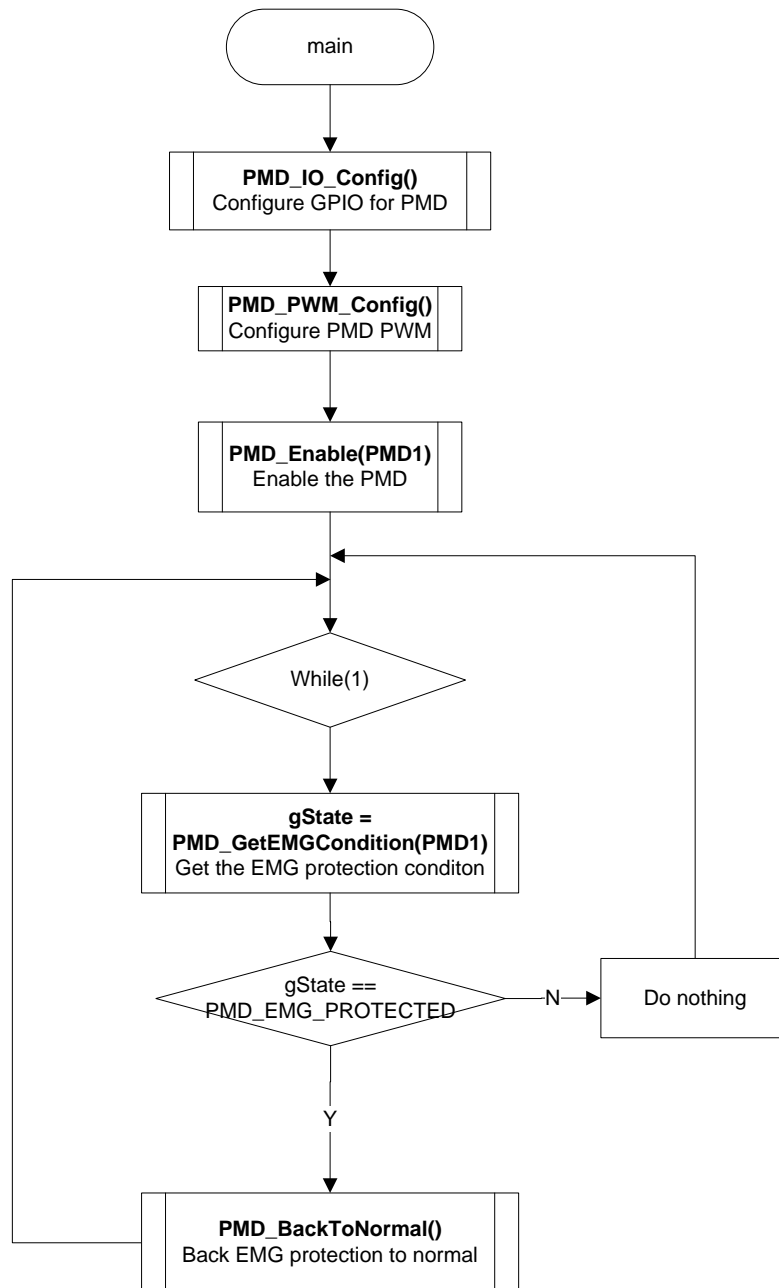
### 7-11-1 Example: Phase Output

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 = 0x3FFU;

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(PMD1,0x0FFU,0x1FFU,0x2FFU);
```

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();
    }
}
```

## 7-12 TRMOSC

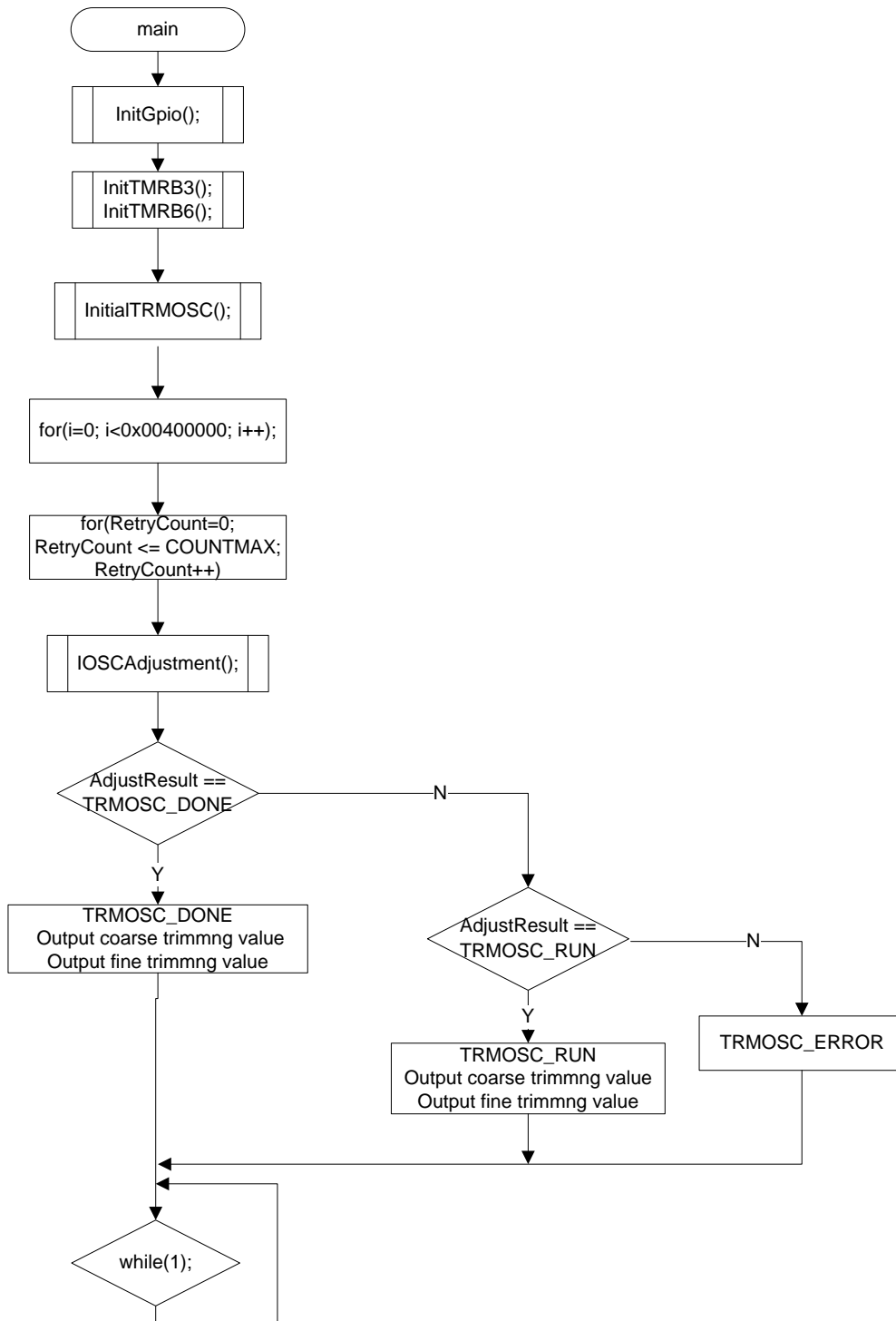
### 7-12-1 Example: TRMOSC

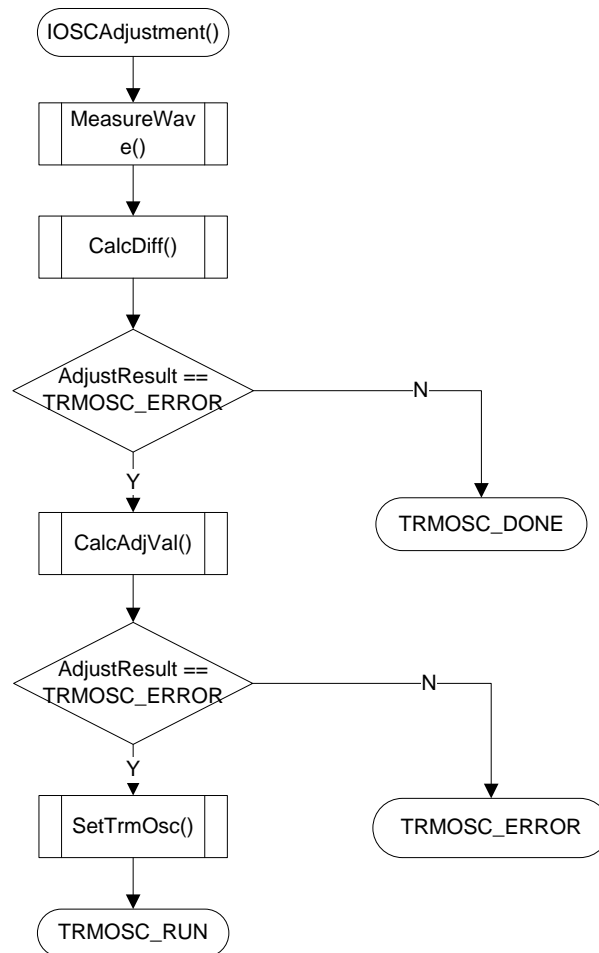
This is a simple example based on the TX03 Peripheral Driver (GPIO, CG, TMRB, TRMOSC).

The example includes:

1. Configuration and initialization TMRB
2. Adjust the internal oscillator

- **Flowchart**





## • Code and Explanation for the Example

Configure GPIO for TMRB.

```

/* TB6IN function */
GPIO_EnableFuncReg(GPIO_PA, GPIO_FUNC_REG_2, GPIO_BIT_6);
GPIO_SetInputEnableReg(GPIO_PA, GPIO_BIT_6, ENABLE);
/* TB3OUT function */
GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_7);
GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_1, ENABLE);

```

Initialize and configure TRMB channel and enable TRMB.

```

/* TB6IN function */
TMRB_InitTypeDef m_tmr;

m_tmr.Mode = TMRB_INTERVAL_TIMER;
m_tmr.ClkDiv = TMRB_CLK_DIV_2;
m_tmr.TrailingTiming = TMRB6COUNT;
m_tmr.UpCntCtrl = TMRB_FREE_RUN;
m_tmr.LeadTiming = TMRB6COUNT;

TMRB_Enable(TSB_TB6);
TMRB_Init(TSB_TB6, &m_tmr);
TMRB_SetCaptureTiming(TSB_TB6, TMRB_CAPTURE_IN_RISING);
TMRB_SetExtStartTrg(TSB_TB6, ENABLE, TMRB_TRG_EDGE_FALLING);
TMRB_SetRunState(TSB_TB6, TMRB_STOP);

```

```
/* TB3OUT function */
TMRB_InitTypeDef m_tmr;
TMRB_FFOutputTypeDef m_tmr_ff;

m_tmr.Mode = TMRB_INTERVAL_TIMER;
m_tmr.ClkDiv = TMRB_CLK_DIV_2;
m_tmr.TrailingTiming = TMRB3COUNT;
m_tmr.UpCntCtrl = TMRB_AUTO_CLEAR;
m_tmr.LeadingTiming = TMRB3COUNT;

m_tmr_ff.FlipflopCtrl = TMRB_FLIPFLOP_CLEAR;
m_tmr_ff.FlipflopReverseTrg =
TMRB_FLIPFLOP_MATCH_LEADINGTIMING;

TMRB_Enable(TSB_TB3);
TMRB_Init(TSB_TB3, &m_tmr);
TMRB_SetFlipFlop(TSB_TB3, &m_tmr_ff);
TMRB_SetRunState(TSB_TB3, TMRB_RUN);
```

## Adjustment of internal OSC

```
MeasureWave(&loscAdjust);          /* Measurement Waveform */
retval = CalcDiff(&loscAdjust);      /* Calculation of difference */
if(retval == TRMOSC_ERROR) {
    retval = CalcAdjVal(&loscAdjust); /* Calculation of adjustment value */
    if(retval == TRMOSC_RUN) {
        AdjustData.TrimmingControl = TRMOSC_CONTROL_ENABLE;
        AdjustData.CoarseTrimmingValue = loscAdjust.CoarseTrim;
        AdjustData.FineTrimmingValue = loscAdjust.FineTrim;
        SetTrmOsc(&AdjustData);      /* Set TRMOSC register */
    }
    else {
        /* do nothing */
    }
}
else {
    /* TRMOSC_DONE */
    /* do nothing */
}
```