TOSHIBA

TX03 Peripheral Driver Usage Example (TMPM370)

Ver 1
Sep, 2017

TOSHIBA ELECTRONIC DEVICES & STORAGE CORPORATION

CMDR-M370UE-01E

RESTRICTIONS ON PRODUCT USE

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

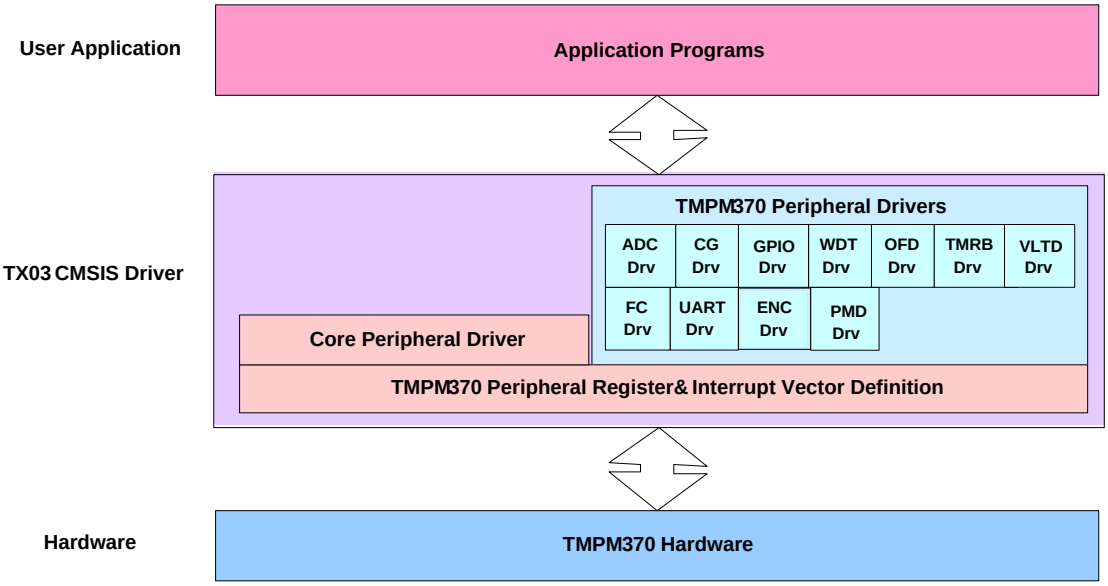
Index

1 General description

The example programs described hereafter are specially designed for TOSHIBA TMPM370 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 peripherial 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	Used for CG demo to wake up system from stop mode
	INT4	Unused
	INT5	Unused
	INT6	Unused
	INT7	Unused
	INT8	Unused
	INT9	Unused
	INT10	Unused
SIO	SIO0	Used for UART retarget demo
	SIO1	Used for UART FIFO and SIO demo
	SIO2	Unused
	SIO3	Unused
16-bit timer	TMRB0	Used for TMRB: General Timer
	TMRB1	Unused
	TMRB2	Unused
	TMRB3	Unused
	TMRB4	Unused
	TMRB5	Unused
	TMRB6	Unused
	TMRB7	Used for TMRB: PPG Output
OFD	-	Used for OFD demo
POR	-	Unused
VLTD	-	Used for VLTD demo
Encoder input function	ENC0	Used for ENC Roller demo
	ENC1	Unused
Programmable Motor Driver	PMD0	Unused
	PMD1	Used for PMD Phase Output demo
12-bit A/D converter	AINA0	Used for ADC data read
	AINA1	Unused
	AINA2	Unused
	AINA3	Unused
	AINA4	Unused
	AINA5	Unused
	AINA6	Unused
	AINA7	Unused
	AINA8	Unused
	AINA9	Unused
	AINA10	Unused
	AINA11	Unused
	AINA12	Unused

Hardware	Channel	Use presence, use
	AINA13	Unused
	AINA14	Unused
	AINB0	Unused
	AINB1	Unused
	AINB2	Unused
	AINB3	Unused
	AINB4	Unused
	AINB5	Unused
	AINB6	Unused
	AINB7	Unused
	AINB8	Unused
	AINB9	Unused
	AINB10	Unused
	AINB11	Unused
	AINB12	Unused
	AINB13	Unused
	AINB14	Unused
	AINB15	Unused
	AINB16	Unused

4 Pin Usage

The example programs are tested on the IAR TMPM370-SK Evaluation Board. Following is pin usage for example programs on IAR TMPM370-SK Evaluation Board(TMPM370FYDFG).

No	Name	Usage
1	CVREFD	Connect with VR4
2	CVREFABC	Connect with VR3
3	DVSS	GND
4	INT3/TB0IN/PA0	INT3/SW1
5	TB0OUT/PA1	SW2
6	INT4/TB1IN/PA2	Unused
7	TB1OUT/PA3	Unused
8	CTS1/SCLK1/PA4	Unused
9	TB6OUT/TXD1/PA5	Unused
10	TB6IN/RXD1/PA6	Unused
11	INT8/TB4IN/PA7	Unused
12	TXD0/PE0	TXD0 (UART0 data transfer)
13	RXD0/PE1	RXD0 (UART0 data receive)
14	CTS0/SCLK0/PE2	Unused
15	TB4OUT/PE3	Unused
16	DVDD5	Unused
17	INT5/TB2IN/PE4	Unused
18	TB2OUT/PE5	Unused
19	INT6/TB3IN/PE6	Unused
20	INT7/TB3OUT/PE7	Unused
21	DVSS	GND
22	INTB/PL0	Unused
23	INTA/PL1	Unused
24	UO0/PC0	LED1
25	XO0/PC1	Unused
26	VO0/PC2	LED2
27	YO0/PC3	Unused
28	WO0/PC4	LED3
29	ZO0/PC5	Unused
30	EMG0/PC6	Unused
31	OVV0/PC7	Unused
32	TB5IN/ENCA0/PD0	ENCA0
33	TB5OUT/ENCB0/PD1	ENCB0

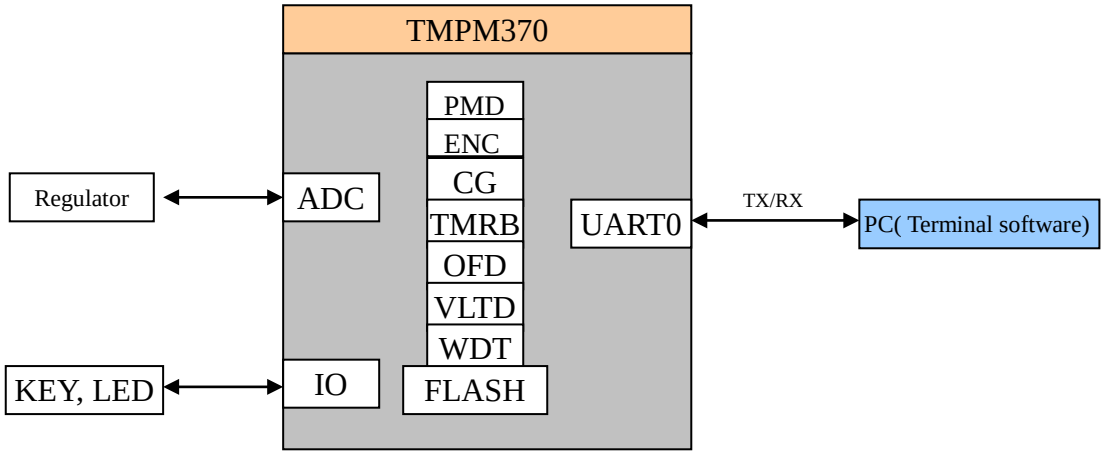
No	Name	Usage
34	ENCZ0/PD2	Unused
35	INT9/PD3	Unused
36	CTS2/SCLK2/PD4	Unused
37	TXD2/PD5	Unused
38	RXD2/PD6	Unused
39	UO1/PG0	PMD UO1 output
40	XO1/PG1	PMD XO1 output
41	VO1/PG2	PMD VO1 output
42	DVDD5	+5V
43	DVSS	GND
44	YO1/PG3	PMD YO1 output
45	WO1/PG4	PMD WO1 output
46	ZO1/PG5	PMD ZO1 output
47	EMG1/PG6	PMD EMG1 input
48	OVV1/PG7	Unused
49	X1	Connect 10MHz oscillator
50	DVSS	GND
51	PM1/X2	Connect 10MHz oscillator
52	PF0/TB7IN/BOOT	Unused
53	PF1/TB7OUT	PPG output
54	PF2/ENCA1/SCLK3	Unused
55	PF3/ENCB1/TXD3	Unused
56	PF4/ENCZ1/RXD3	Unused
57	VOUT15	GND
58	DVSS	GND
59	MODE	GND
60	RVDD5	+5V, VLTD detected pin (*)
61	RESET	RESET
62	VOUT3	GND
63	DVDD5E	+5V
64	DVSS	GND
65	PB0/TRACECLK	Unused
66	PB1/TRACEDATA 0	Unused
67	PB2/TRACEDATA 1	Unused
68	PB3/TMS/SWDIO	Unused
69	PB4/TCK/SWCLK	Unused
70	PB5/TDO/SWV	Unused
71	PB6/TDI	Unused
72	PB7/TRST	Unused
73	PK1/AINB12/INTF	Unused
74	PK0/AINB11/INTE	Unused

No	Name	Usage
75	PJ7/AINB10/INTD	Unused
76	PJ6/AINB9/INTC	Unused
77	PJ5/AINB8	Unused
78	PJ4/AINB7	Unused
79	PJ3/AINB6	Unused
80	PJ2/AINB5	Unused
81	PJ1/AINB4	Unused
82	PJ0/AINB3	Unused
83	AVSSB/VREFLB	AGND
84	AVDD5B/VREFHB	A+5V
85	PI3/AINA11/AINB2	Unused
86	PI2/AINA10/AINB1	Unused
87	PI1/AINA9/AINB0	Unused
88	AVSSA/VREFLA	AGND
89	AVDD5A/VREFHA	A+5V
90	PI0/AINA8	Unused
91	PH7/AINA7	Unused
92	PH6/AINA6	Unused
93	PH5/AINA5	Unused
94	PH4/AINA4	Unused
95	PH3/AINA3	Unused
96	PH2/AINA2/INT2	Unused
97	PH1/AINA1/INT1	Unused
98	PH0/AINA0/INT0	Analog input, connect with VR1
99	AMPVSS	AGND
100	AMPVDD5	A+5V

* Pins used for VLTD programs on TOSHIBA M370 Evaluation board.

5 Development Environment

Following is development environment:



1. Hardware board:

   IAR TMPM370-SK Evaluation Board

2. Development tool:
   - IAR:
     1) J-Link: IAR J-Link-ARM 7.0 / J-Link 6.0
     2) IDE: IAR Embedded workbench 5.5 version
   - KEIL:
     1) U-Link: Realview ULINK2
     2) IDE: KEIL uVision 4.14

***Note:** For ENC&PMD 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 6.50.1 version
- KEIL:
  1) IDE: KEIL uVision 4.60

6 Functional description

6-1 Operation mode

There are three operation modes for TMPM370: 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.

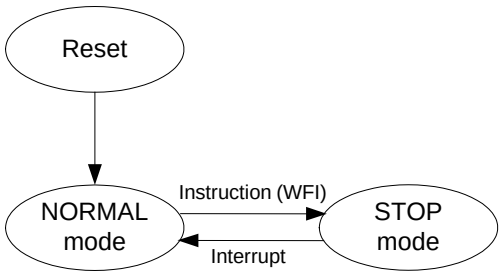
IDLE and STOP mode are low power mode.
To shift to lower power mode, in system control register CGSTBYCR<STBY2:0>, select the IDLE, or STOP mode, and run the WFI (Wait For Interrupt) command.

Note: Only STOP mode is demostrated 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(INT0 to INTF) can release the SLEEP 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 PH0/AINA0. The voltage on it will be measeured and print to stdout. As stdout is retargeted to UART, connect UART0 with a PC and the AD result can be displayed on terminal software.

6-3 CG

6-3-1 Power mode change

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

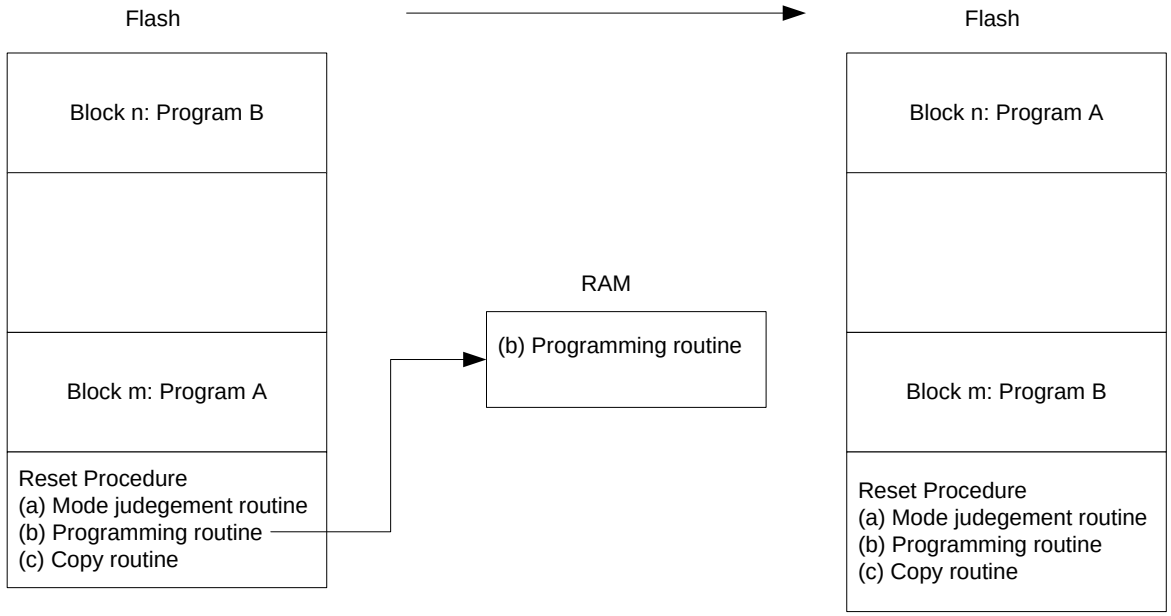
Current mode	Action (Key)	Operation	LED display
NORMAL	Press SW2	NORMAL→ STOP	LED off
STOP	Press SW1	STOP →NORMAL	LED on

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

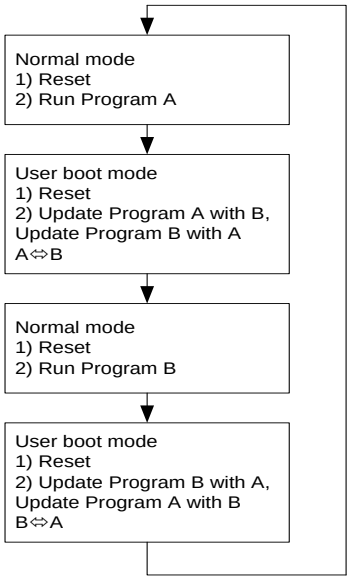


Demo sequence

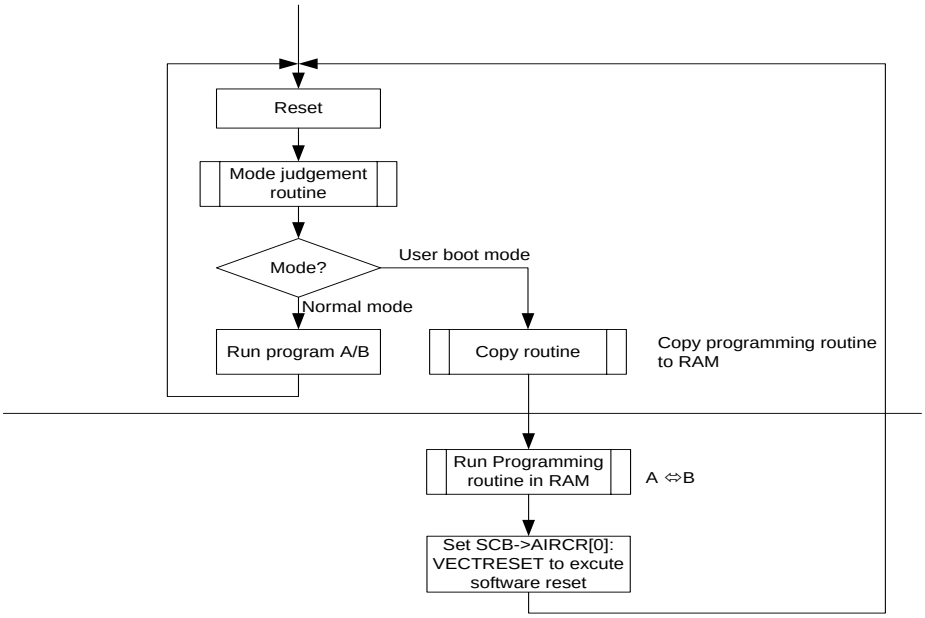
(1) Power on
The demo board runs Reset Procedure.
Then initial program A stored in flash runs.
LED1 blinks.

(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 configures GPIO to LED, then turn on the LED, or turn off the LED.

6-6 OFD

The example demonstrates how to initialize the OFD function.

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 UART0. Then application code can use printf() to output to serial port.

6-7-2 UART FIFO

In this sample program, UART0 sent the data "TMPM3701" to UART1 use FIFO, and at same time UART0 receive the data "TMPM3702" 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 = "TMPM3702",and RxBuffer1 = "TMPM3701".

6-7-3 SIO

This demo will show synchronously transfer and receive usage of SIO module in TMPM370
It use the channel SIO0, SIO1 and transfer 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 switch1 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 key pressed.
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 reset automatically by hardware and PC0 is set to high level by software.

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 TMPM370 ENC driver.

Demo procedure:

1. Use a wire to connect the pinA (see diagram 1) of mouse wheel encoder with the pin32 (ENCA0) of TMPM370FYDFG MCU board.
2. Use a wire to connect the pinB (see diagram 1) of mouse wheel encoder with the pin33 (ENCB0) of TMPM370FYDFG 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.

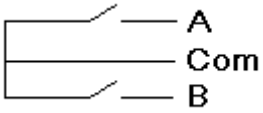


Diagram 1

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.
2. Use a wire to connect the EMG1 (pin 47) with the DVSS (pin 64) to pull it low. (LED1 will light on) It means that PMD is in EMG protection.
3. Use a wire to connect the EMG1 (pin 47) with the DVDD5 (pin 63) to pull it high. (LED1will light off) It means that PMD is in normal operation.
4. Connect UO1 (pin 39) to oscilloscope to observe the waveform.
5. Connect XO1 (pin 40) to oscilloscope to observe the waveform.
6. Connect VO1 (pin 41) to oscilloscope to observe the waveform.
7. ConnectYO1 (pin 44) to oscilloscope to observe the waveform.

8. Connect WO1 (pin 45) to oscilloscope to observe the waveform.
9. Connect ZO1 (pin 46) 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 IAR TMPM370-SK Evaluation Board which will demonstrate the main feature of TMPM370 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
        │   tmpm370_wdt_int.c
        │   tmpm370_wdt_int.h
        │
        ├─TX03_CMSIS
        │   │   system_TMPM370.c
        │   │   system_TMPM370.h
        │   │   TMPM370.h
        │   │
        │   └─startup
        │         startup_TMPM370.s
        │
```

```
        ├─TX03_Periph_Driver
        │   ├─inc
        │   │   tmpm370_cg.h
        │   │   tmpm370_gpio.h
        │   │   tmpm370_wdt.h
        │   │   tx03_common.h
        │   │
        │   └─src
        │       tmpm370_cg.c
        │       tmpm370_gpio.c
        │       tmpm370_wdt.c
        │
        └─TMPM370-SK
                led.c
                led.h
```

KEIL MDK:

```
    ├─WDT_NMI
        ├─APP
        │   main.c
        │   tmpm370_wdt_int.c
        │
        ├─TX03_CMSIS
        │   │   system_TMPM370.c
        │   │   system_TMPM370.h
        │   │   TMPM370.h
        │   │
        │   └─startup
        │       startup_TMPM370.s
        │
        ├─TX03_Periph_Driver
        │    tmpm370_cg.c
        │    tmpm370_gpio.c
        │    tmpm370_wdt.c
        │
        └─TMPM370-SK
             led.c
```

7-1 ADC

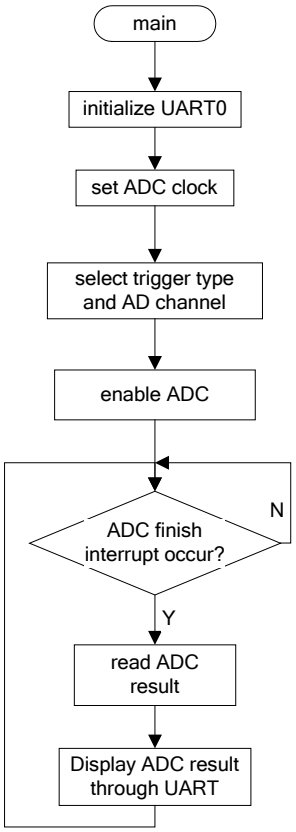
7-1-1 Example: ADC Data Read

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

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.

```
/* set ADC clock */
ADC_SetClk(TSB_ADA, ADC_HOLD_FIX, ADC_FC_DIVIDE_LEVEL_16);

/* select trigger type and AD channel, this time software trigger is used */
/* VR1 is connected to AIN0, remember to input with macro TRG_ENABLE() */
ADC_SetSWTrg(TSB_ADA, ADC_REG0, TRG_ENABLE(ADC_AIN0));

/* enable ADC module */
ADC_Enable(TSB_ADA);
```

Then start AD conversion:

```
ADC_Start(TSB_ADA, ADC_TRG_SW);
```

After started AD conversion, check ADC module state. When AD conversion is finished, call ADC_Display() and start another AD conversion.

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

    If (adcState == DONE) {
        ADC_Display();

        /* Start another AD conversion */
        ADC_Start(TSB_ADA, ADC_TRG_SW);
    }
}
```

In ADC_Display() function, read the corresponding ADREG to get AD conversion result:

```
ADC_Result result = ADC_GetConvertResult(TSB_ADA, ADC_REG0);
```

7-2 CG

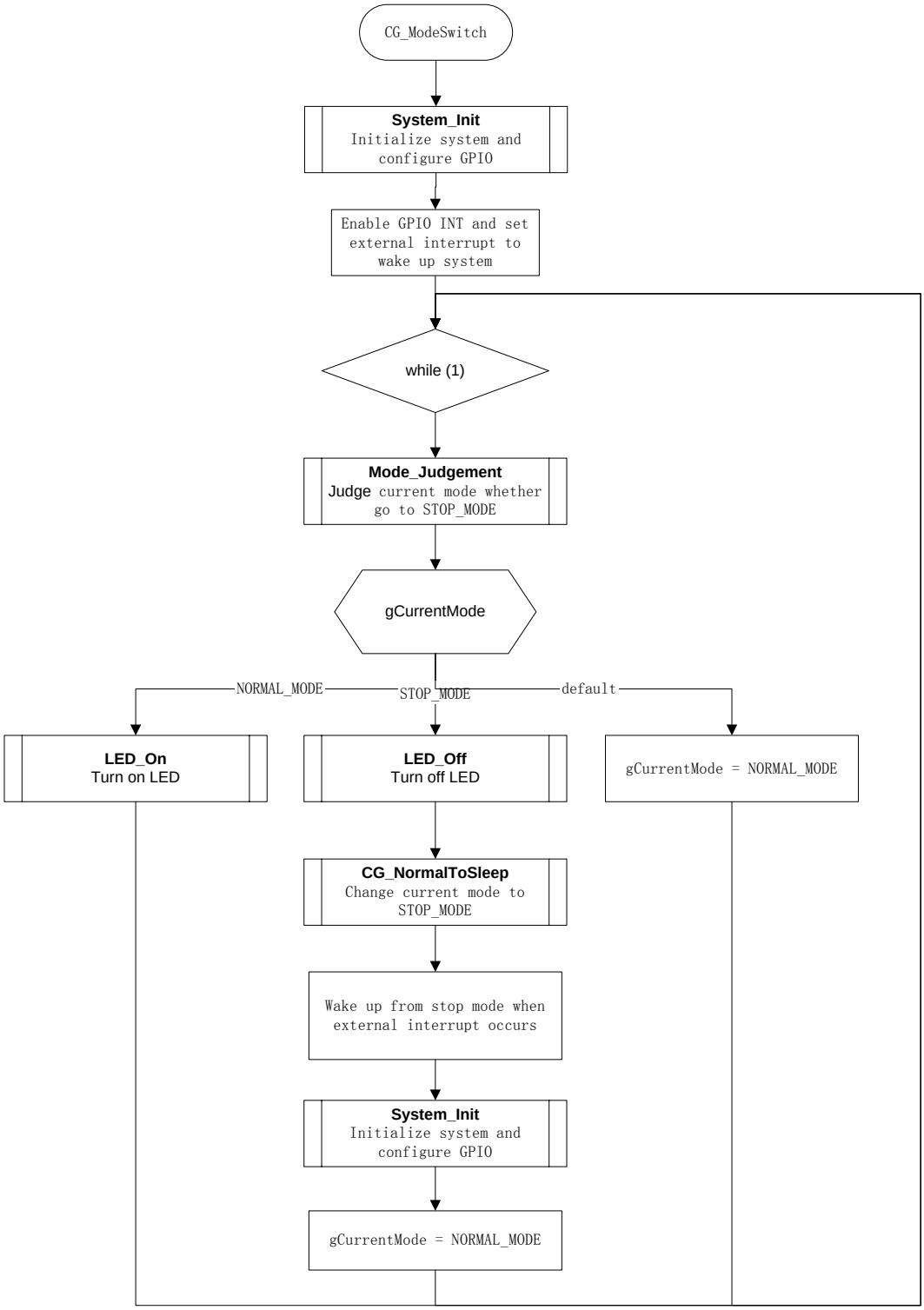
7-2-1 Example: Power mode change

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**

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);
/* Enable high-speed oscillator */
CG_SetFosc(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 INT3 to wake up system. Clear interrupt pending request, then enable INT3. 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_3);
/* Set external interrupt to wake up system */
CG_SetSTBYReleaseINTSrc(CG_INT_SRC_3,
                                    CG_INT_ACTIVE_STATE_FALLING, ENABLE);
NVIC_ClearPendingIRQ(INT3_IRQn);
NVIC_EnableIRQ(INT3_IRQn);
__enable_irq();

CG_SetWarmUpTime(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_WUODR_PLL);
      CG_StartWarmUp();

      do {
            st = CG_GetWarmUpState();
      } while (st != DONE);

      retval = CG_SetFcSrc(CG_FC_SRC_FPLL);
} else {
```

```
        /*Do nothing */
    }
```

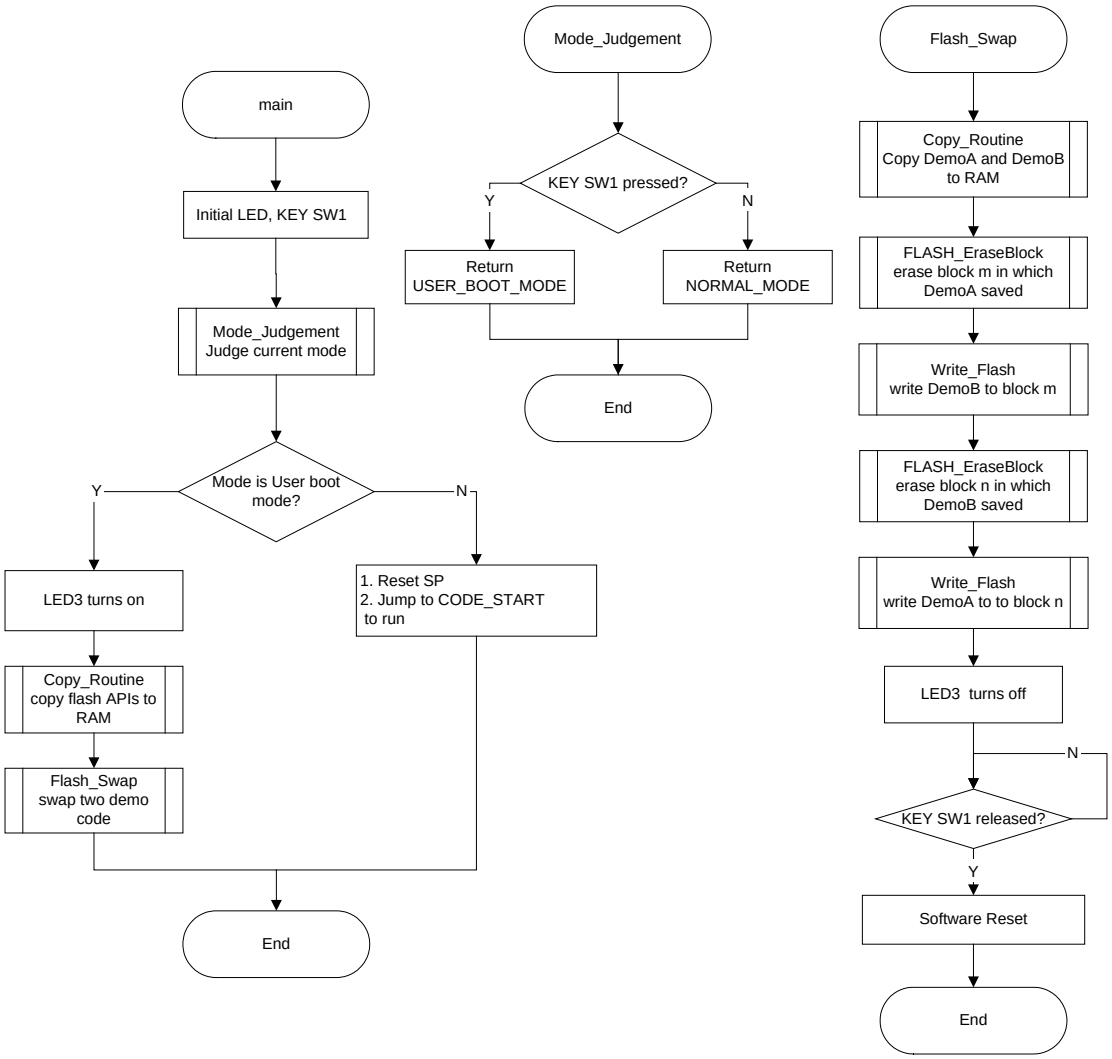
7-3 FLASH

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.

```
    Init_Led();
    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
        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->AIRCR register. Then this demo will run again to enter normal mode, because Demo A and B have been exchanged, the address "CODE_START" has become the start address of Demo B, Demo B will run (LED2 blinks).

```
    delay(0x7FFFFF);
    Led_Off(LED3);                          /* complete and restart */
    delay(0x2FFFFF);

    /* wait for Key SW1 to release*/
    while (SW1_Get() == GPIO_BIT_VALUE_1) {
    }

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

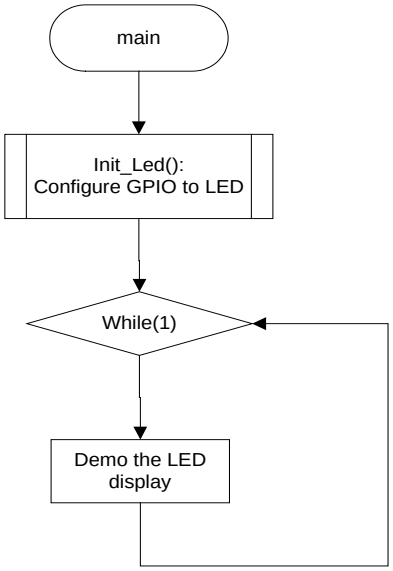
7-4 GPIO

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

The example includes:

1. GPIO initialization
2. Write data to GPIO

- **Flowchart**



- **Code and Explanation for the Example:**

At first, configure GPIO to LED. For example, set the port as output pin, set port data.

```
void Init_Led()
{
	GPIO_SetOutput(LED_DATA_PORT, LED1 | LED2 | LED3);
	GPIO_WriteDataBit(LED_DATA_PORT, LED1, GPIO_BIT_VALUE_0);
	GPIO_WriteDataBit(LED_DATA_PORT, LED2, GPIO_BIT_VALUE_0);
	GPIO_WriteDataBit(LED_DATA_PORT, LED3, GPIO_BIT_VALUE_0);
}
```

In the While process, LED is turned on one by one while others are off.

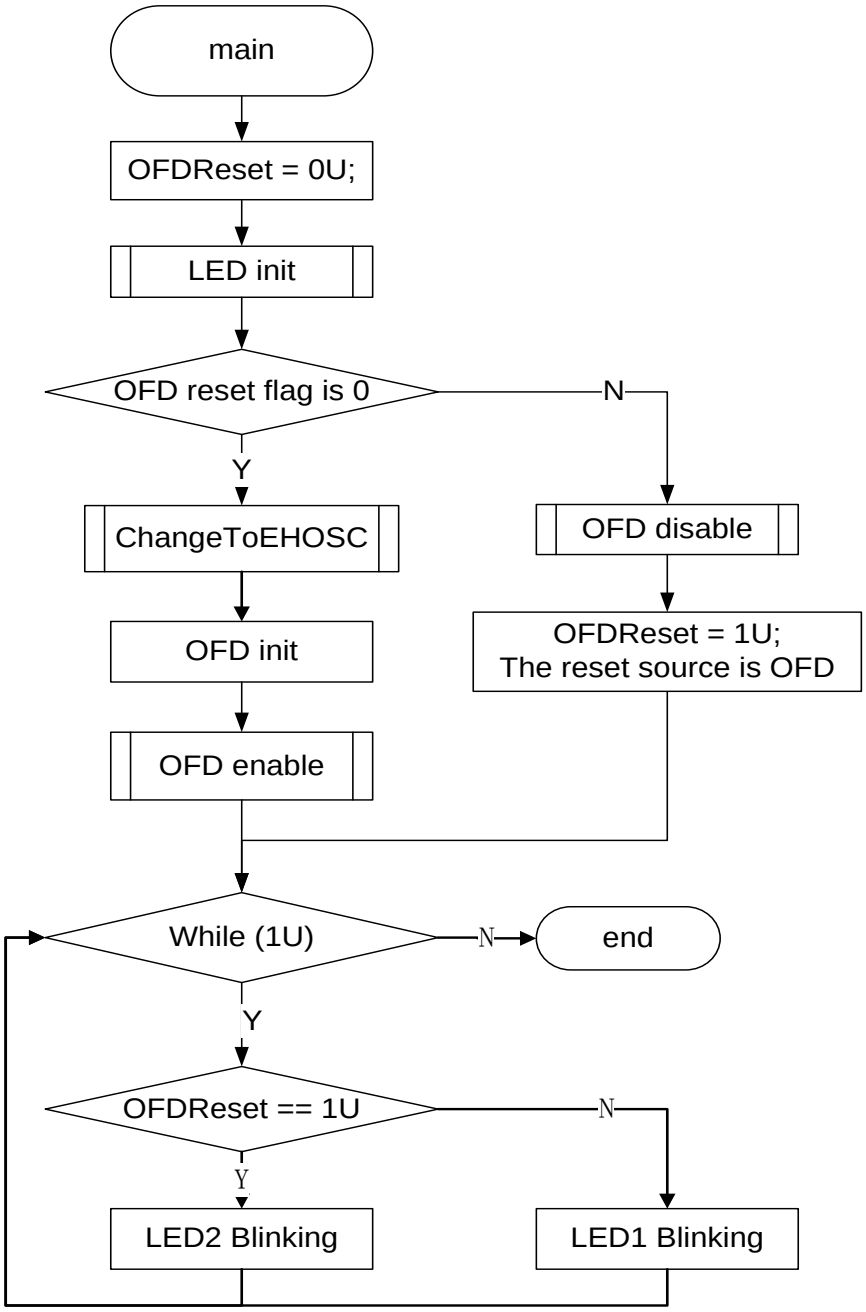
7-5 OFD

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.

```
Init_Led();
```

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->SYSCR    = SYSCR_Val;
  TSB_CG->OSCCR    = OSCCR_Val;
  TSB_CG->STBYCR = STBYCR_Val;

  if (TSB_CG_OSCCR_PLLON) {     /* If PLL enabled */
    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 configurate 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

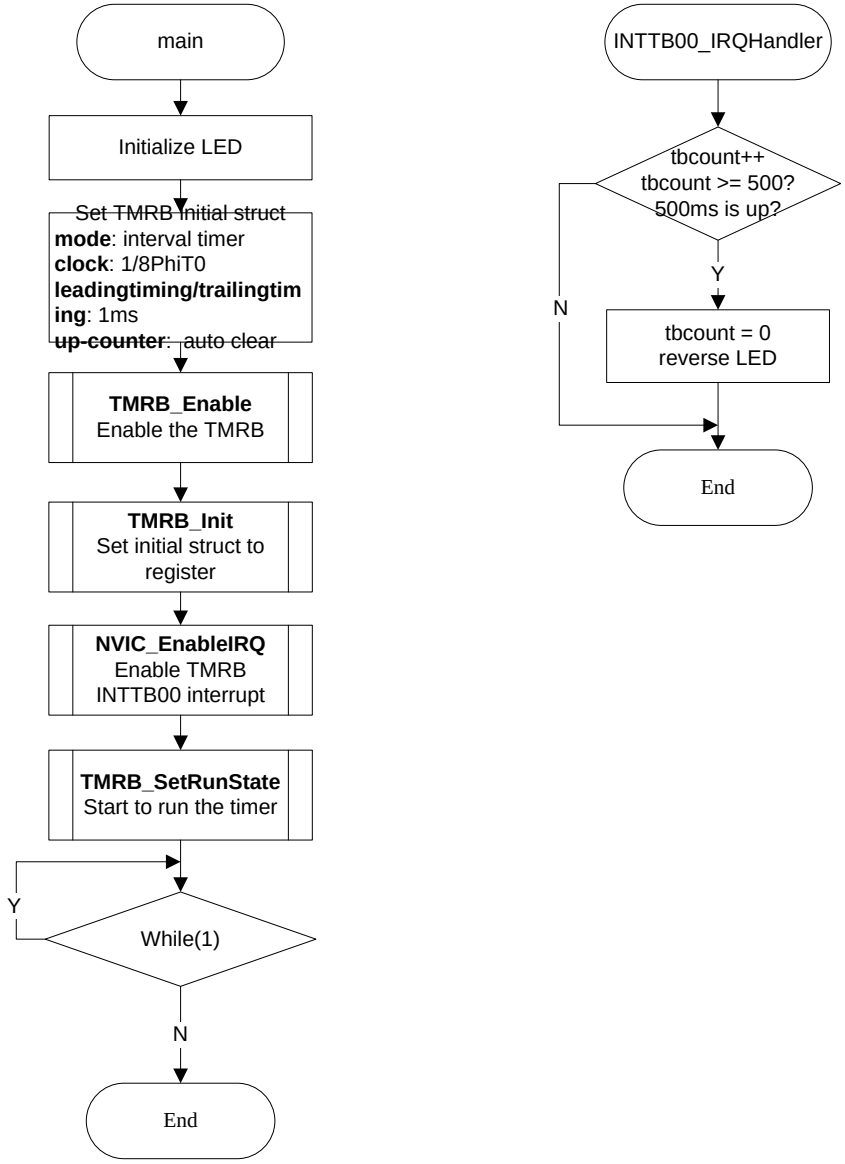
7-6-1 Example: General Timer

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 channel on SK board.

```
Init_Led();                    /* LED initialize */
```

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 0x1388, because fphiT0 = fsys = fc = 10MHz * PLL = 80MHz, ftmrb = 1/8 fphiT0 = 10MHz, Ttmrb = 0.1us, 1ms/0.1us = 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.LeadingTiming = 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(LED1);
    } else {
        Led_On(LED1);
    }
} else {
    /* do nothing */
}
```

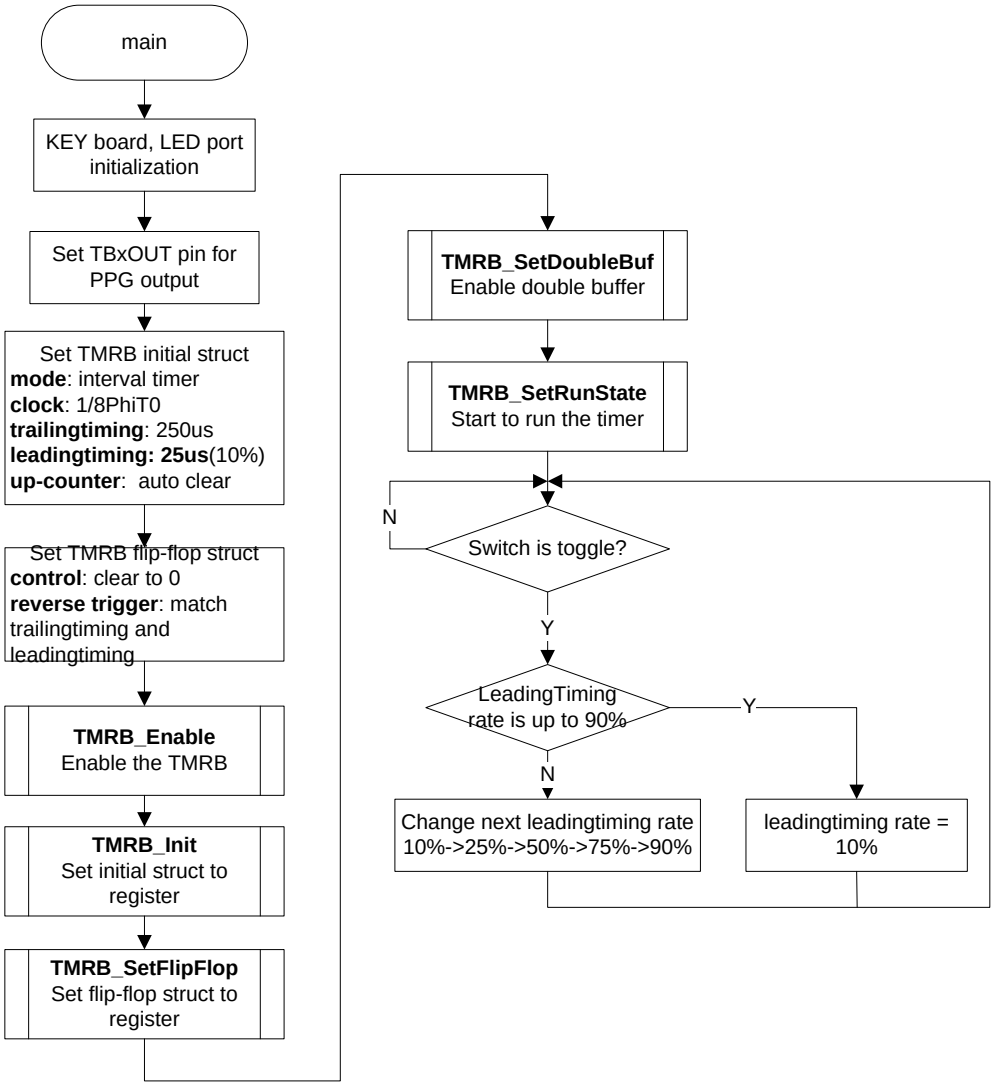
7-6-2 Example: PPG Output

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

The example includes:

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

- **Flow Chart**



- **Code and Explanation for the Example**

At first, initialize KEY board and LED channel, set PF1 as TB7OUT for PPG output.

```
GPIO_SetInput(KEYPORT, GPIO_BIT_0);                    /* set KEY port to input */

Init_Led();                          /* LED initialize */

/* Set PF1 as TB7OUT for PPG output */
GPIO_SetOutput(GPIO_PF, GPIO_BIT_1);
GPIO_EnableFuncReg(GPIO_PF, GPIO_FUNC_REG_1, GPIO_BIT_1);
GPIO_SetPullUp(GPIO_PF, GPIO_BIT_1, ENABLE);          /* Enable pull up */
```

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 250us, macro TMRB7TIME equals 0x09C4, because fphiT0 = fsys = fc = 10MHz * PLL = 80MHz, ftmrb = 1/8 fphiT0 = 10MHz, Ttmrb = 0.1us, 250us/0.1us = 2500 = 0x09C4 (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 = TMRB7TIME;          /* trailing timing is 250us */
m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;         /* up-counter auto clear */
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_CLEAR;
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, display current leadingtiming on 7-seg LED array.

```
do {                            /* wait if switch is Low */
        keyvalue = GPIO_ReadDataBit(KEYPORT, GPIO_BIT_0);
        SegDisplay(leadingtiming_num[Rate]); /* display current leadingtiming */
    } 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 calculation method of leading timing value:
TrailingTiming = 250us, ftmrb = 1/8 fphiT0 = 10MHz, Ttmrb = 0.1us (these time parameters will be different if use different CG setting)

LeadingTiming = 10%: high-level time is 250*10% = 25us, low-level time is 250-25 = 225us, counter value = 225us/Ttmrb = 0x8CA
LeadingTiming = 25%: high-level time is 250*25% = 62.5us, low-level time is 250-62.5 = 187.5us, counter value = 187.5us/Ttmrb = 0x753

LeadingTiming = 50%: high-level time is 250*50% = 125us, low-level time is 250-125 = 125us, counter value = 125us/Ttmrb = 0x4E2

LeadingTiming = 75%: high-level time is 250*75% = 187.5us, low-level time is 250-187.5 = 62.5us, counter value = 62.5us/Ttmrb = 0x271

LeadingTiming = 90%: high-level time is 250*90% = 225us, low-level time is 250-225 = 25us, counter value = 25us/Ttmrb = 0xFA

That is the calculation method of leading timing array:

```
uint32_t LeadingTiming[5] = { 0x8CAU, 0x753U, 0x4E2U, 0x271U, 0xFAU };
/* leadingtiming: 10%, 25%, 50%, 75%, 90% */
```

7-7 SIO/UART

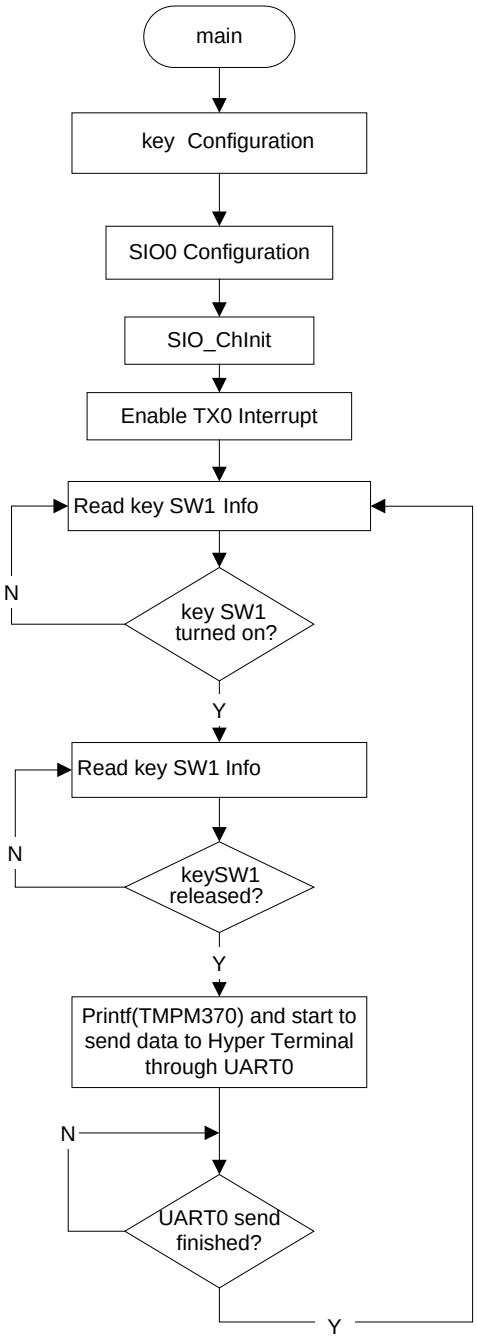
7-7-1 Example: Retarget

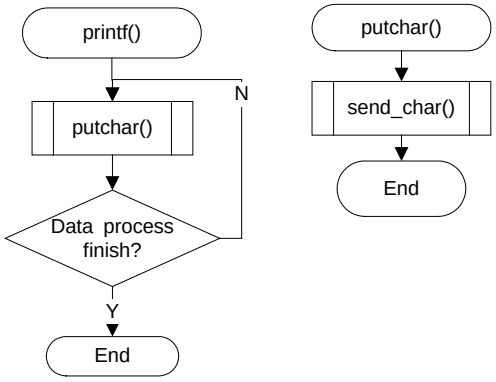
This is a simple example based on the TX03 Peripheral Driver (UART, GPIO).

The example includes:

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

- **Flowchart**


main
key Configuration
SIO0 Configuration
SIO_ChInit
Enable TX0 Interrupt
Read key SW1 Info
N
key SW1 turned on?
Y
Read key SW1 Info
N
keySW1 released?
Y
Printf(TMPM370) and start to send data to Hyper Terminal through UART0
N
UART0 send finished?
Y



- **Code and Explanation for the Example**

At first, Configure Key and SIO0, then Initialize UART0.
Use GPIO peripheral drivers configure GPIO for Key.

```
GPIO_SetPullUp(TSW_PORT, GPIO_BIT_0, ENABLE);
GPIO_SetPullUp(TSW_PORT, GPIO_BIT_1, ENABLE);
GPIO_SetPullUp(TSW_PORT, GPIO_BIT_2, ENABLE);
GPIO_SetInputEnableReg(TSW_PORT, GPIO_BIT_0, ENABLE);
GPIO_SetInputEnableReg(TSW_PORT, GPIO_BIT_1, ENABLE);
GPIO_SetInputEnableReg(TSW_PORT, GPIO_BIT_2, ENABLE);
```

Use GPIO peripheral drivers configure GPIO for UART0.

```
GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_0, ENABLE);
GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_0);
GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_1);
GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_1, 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_TX;
myUART.FlowCtrl = UART_NONE_FLOW_CTRL;
```

Use UART peripheral drivers enable and initialize UART0.

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

After above setting, enable UART0 TX interrupt.

```
NVIC_EnableIRQ(INTTX0_IRQn);
```

Then Read Key info:

```
TSW_info = GPIO_ReadData(TSW_PORT) & TSWBITS;
```

Then Judge whether key SW1 has been turned on, if key SW1 is turned on, then wait for key SW1 released:

```
if (TSW_info == TSW1) {
/* wait for SW1 released */
do {
    wait_TSW1 = 0U;
```

```
        TSW_info = GPIO_ReadData(TSW_PORT) & TSWBITS;
        } while (TSW_info != TSWRELEASE);
    }
```

After key SW1 has been released, start to send data by printf() through UART0.

```
    printf("%s\r\n", TxBuffer);         /* SIO0 send data */
```

Function printf() will call putchar() for IAR Compiler and fputc() for RealView Compiler to output data to UART0.

```
    #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;
    }
```

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

UART0 TX interrupt routine:

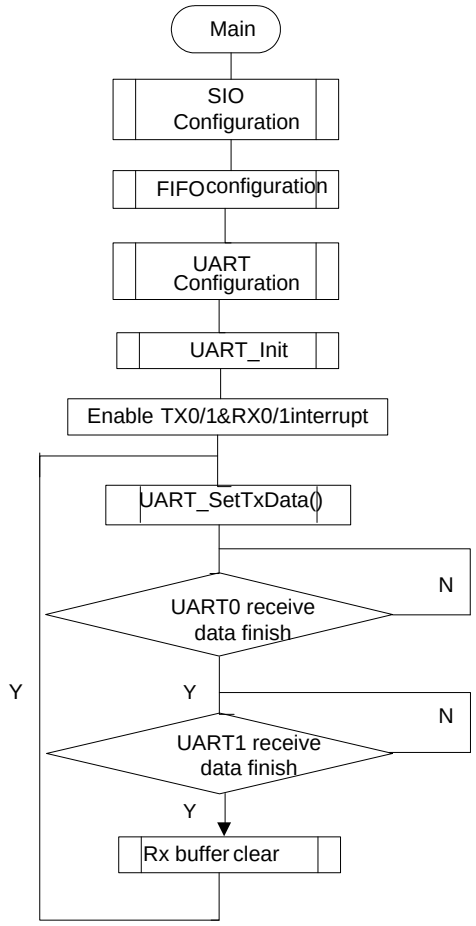
```
    void INTTX0_IRQHandler(void)
    {
        if (gSIORdIndex < gSIOWrIndex) {       /* buffer is not empty */
            UART_SetTxData(UART0, gSIOTxBuffer[gSIORdIndex++]);/* send data */
            fSIO_INT = SET;              /* SIO0 INT is enable */
        } else {
            /* disable SIO0 INT */
            fSIO_INT = CLEAR;
            NVIC_DisableIRQ(INTTX0_IRQn);
            fSIOTxOK = YES;
        }
        if (gSIORdIndex >= gSIOWrIndex) {     /* reset buffer index */
            gSIOWrIndex = CLEAR;
            gSIORdIndex = CLEAR;
        } else {
            /* do nothing */
        }
    }
```

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:

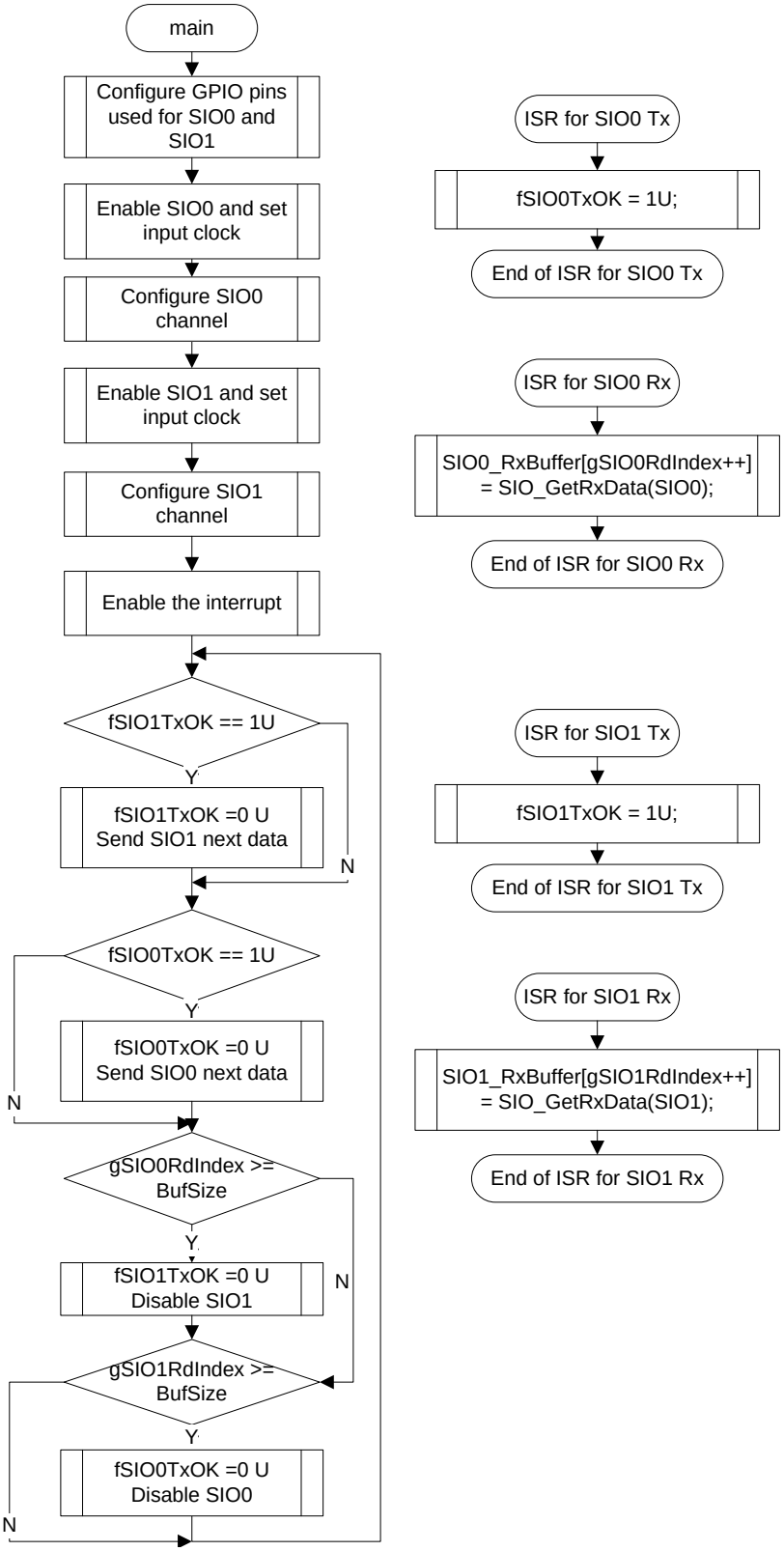
```
void INTRX1_IRQHandler(void)
{
    volatile UART_Err err;

    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

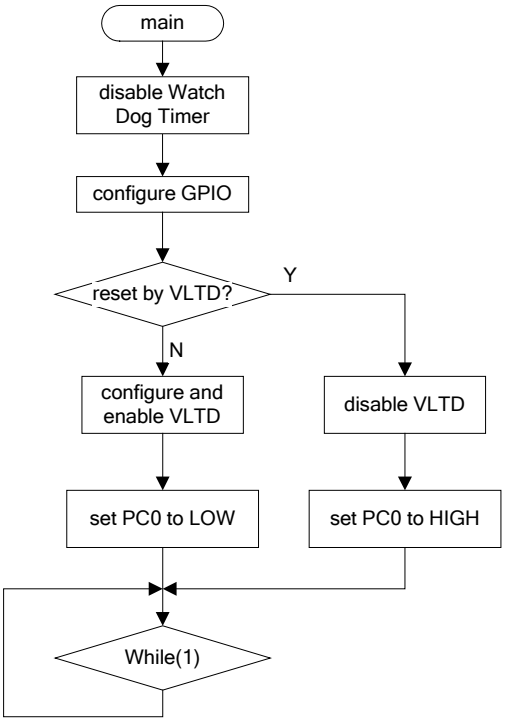
7-8-1 Example: VLTD Reset

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 PC0 as output port.

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

Then read VLTD reset flag If it is not reset by VLTD, configure and enable VLTD. Set PC0 to LOW.

```
if (CG_GetResetFlag().Bit.VLTDReset == 0U) {
        VLTD_SetVoltage(VLTD_DETECT_VOLTAGE_46);
        VLTD_Enable();
        GPIO_WriteData(GPIO_PC, 0x00);
```

Otherwise, disable VLTD and set PC0 to HIGH.

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

7-9 WDT

This is a simple example based on the TX03 Peripheral Driver (WDT, CG, 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/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, then enable WDT.

```
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

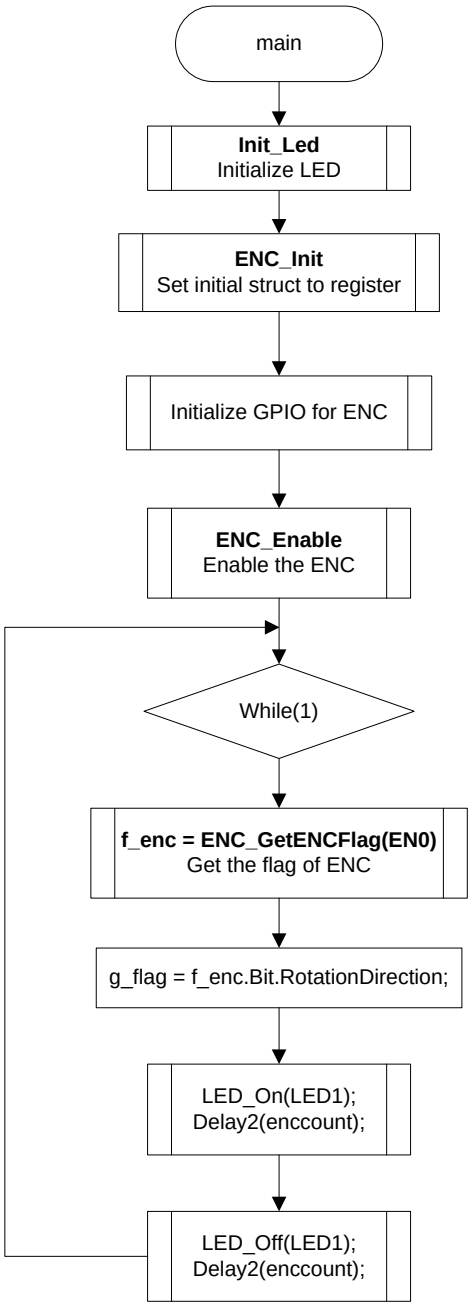
7-10-1 Example: Rolling Detection

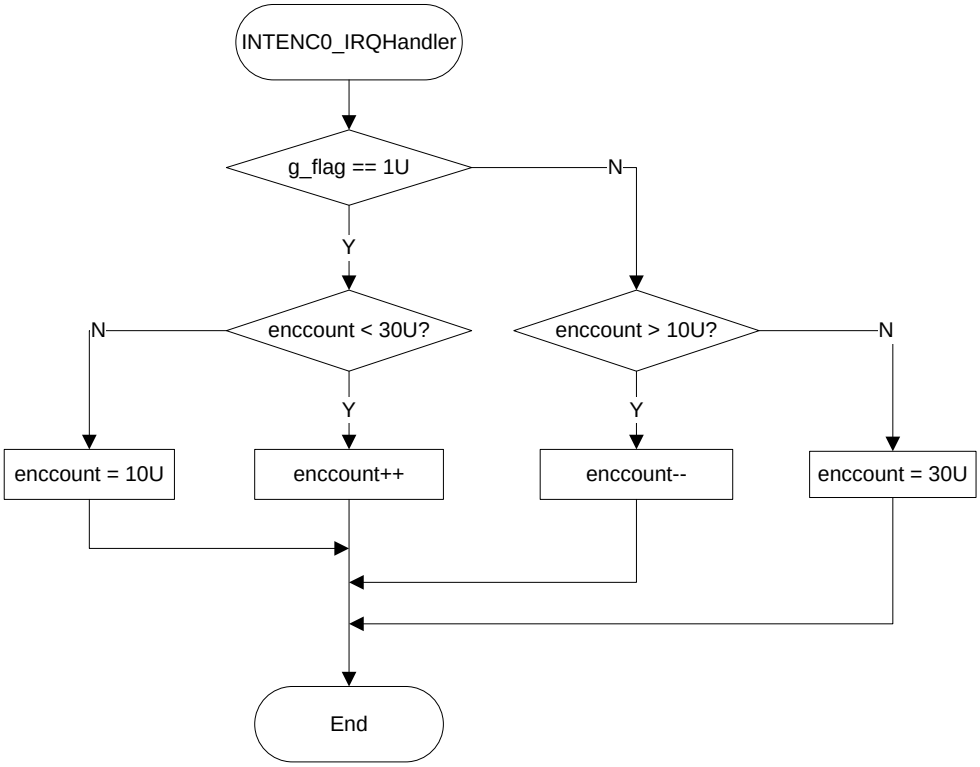
This is a simple example based on the TX03 Peripheral Driver (ENC, GPIO).

The example includes:

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

- **Flowchart**


main
Init_Led
Initialize LED
ENC_Init
Set initial struct to register
Initialize GPIO for ENC
ENC_Enable
Enable the ENC
While(1)
f_enc = ENC_GetENCFlag(EN0)
Get the flag of ENC
g_flag = f_enc.Bit.RotationDirection;
LED_On(LED1);
Delay2(enccount);
LED_Off(LED1);
Delay2(enccount);



- **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 */
  Init_Led();
```

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 PD0 and PD1 as inputs of ENC0.

```
/* GPIO initialization */
GPIO_SetInputEnableReg(GPIO_PD, GPIO_BIT_0, ENABLE);/*Set PD0 as
input */
GPIO_EnableFuncReg(GPIO_PD, GPIO_FUNC_REG_1, GPIO_BIT_0);/*Set
PD0 as ENCA0 */
GPIO_SetInputEnableReg(GPIO_PD, GPIO_BIT_1, ENABLE);/*Set PD1 as
input */
GPIO_EnableFuncReg(GPIO_PD, GPIO_FUNC_REG_1, GPIO_BIT_1);/*Set
```

```
PD1 as ENCB0 */
```

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 cycle 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);
}
```

Enccount is the count of rolling, and it changed in the interupt 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

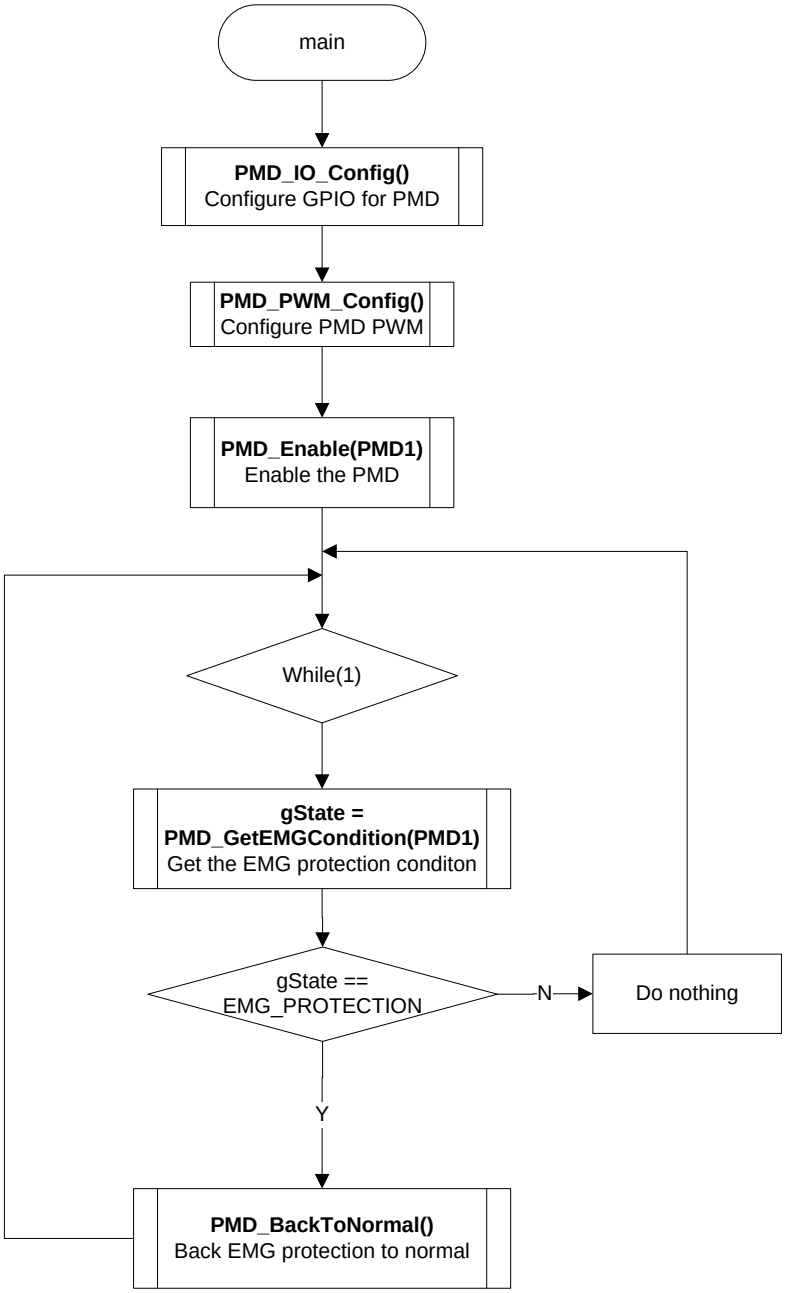
7-11-1 Example: Phase Output

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 */
Init_Led();

/* 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, LED1 will light on, and then back EMG protection to normal. If the condition is not protection condition, LED1 will light off.

```
    while(1){
        /* Get the EMG protection condition*/
        gState = PMD_GetEMGCondition(PMD1);
        /* Judge the EMG protection condition */
        if (gState == EMG_PROTECTION) {
            /* LED1 will light on if the condition is protection */
            Led_On(LED1);
            /* Back EMG protection to normal */
            PMD_BackToNormal();
            Delay(1000U);
        } else {
            /* LED1 will light off if the condition is not protection */
            Led_Off(LED1);
        }    }
```