TOSHIBA

TX00 Peripheral Driver Usage Example

(TMPM037)

Ver 2

May, 2018

TOSHIBA ELECTRONIC DEVICES & STORAGE CORPORATION

CMDR-M037UE-01E

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 3
5 Development Environment....5
6 Functional description 6
6-1 Operation mode 6
6-2 ADC....6
6-3 CG....7
6-3-1 STOP1....7
6-3-2 IDLE....7
6-4 DMAC 7
6-5 FLASH 8
6-6 GPIO 10
6-7 I2C 10
6-7-1 I2C Slave 10
6-7-2 I2C Master 11
6-8 LVD 11
6-9 TMR16A....11
6-10 TMRB....12
6-10-1 General Timer 12
6-10-2 PPG Output....12
6-11 SIO/UART....12
6-11-1 Retarget 12
6-11-2 UART FIFO....12
6-11-3 SIO....13
6-12 WDT....13
7 Software....13
7-1 ADC....15
7-1-1 Example: ADC Data Read....15
7-2 CG....17
7-2-1 Example: NORMAL <-> STOP1 mode change....17
7-2-2 Example: NORMAL <-> IDLE mode change....20
7-3 DMAC 21
7-3-1 Example: DMA transfer from Memory to peripheral....21
7-4 FLASH 24
7-4-1 Example: Flash_UserBoot 24
7-5 GPIO 28
7-5-1 Example: GPIO Data Read....28
7-6 I2C 29
7-6-1 Example: I2C Slave....29
7-6-2 Example: I2C Master 33
7-7 LVD 37

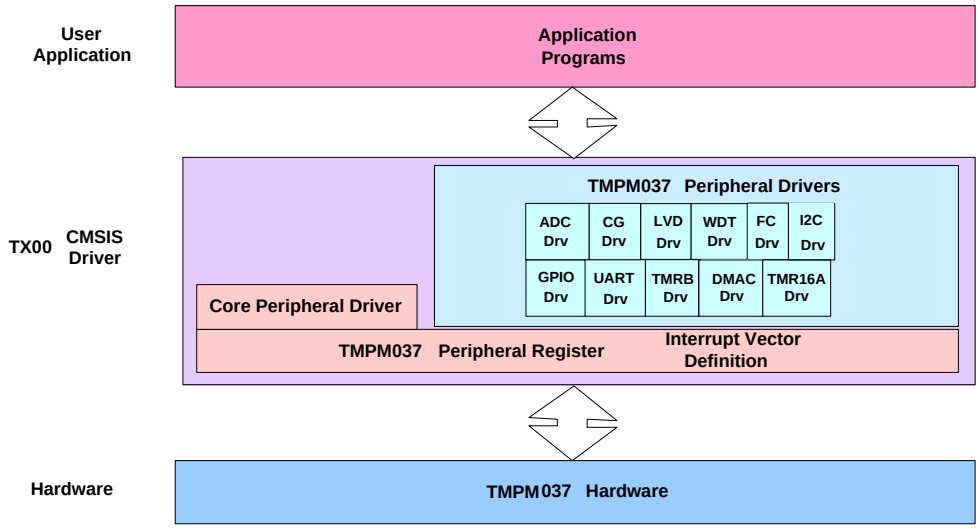
7-7-1 Example: LVD 37
7-8 TMR16A 38
7-8-1 Example: General Timer 38
7-9 TMRB 40
7-9-1 Example: General Timer 40
7-9-2 Example: PPG Output 42
7-10 SIO/UART 46
7-10-1 Example: Retarget 46
7-10-2 Example: UART FIFO 49
7-10-3 Example: SIO 53
7-11 WDT 57
7-11-1 Example:WDT 57

1 General description

The example programs described hereafter are specially designed for TMPM037FWUG 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 TX00 peripheral driver as following.



3 Build-in hardware usage

Hardware	Channel	Use presence, use
CG	Clock Gear	Used for CG demo
	PLL	PLL on (2x)
Standby mode	-	Used for CG demo (IDLE mode, STOP1 mode)
SysTick	-	Unused
Watch dog timer (WDT)	-	Used for WDT
Interrupt (INT)	INT0	Used for CG demo to wake up system from stop mode
	INT1	Unused
	INT2	Unused
	INT3	Unused

Hardware	Channel	Use presence, use
	INT4	Unused
	INT5	Unused
SIO	SIO0	Used for UART retarget demo/ SIO demo/ UART FIFO demo
	SIO1	Used for SIO demo
	SIO2	Used for UART Retarget demo
	SIO3	Used for UART FIFO demo
	SIO4	Unused
16-bit timerB	TMRB0	Used for TMRB: General Timer, PPG Output
	TMRB1	Unused
	TMRB2	Unused
	TMRB3	Unused
	TMRB4	Unused
	TMRB5	Unused
	TMRB6	Unused
	TMRB7	Unused
16-bit timerA	TMR16A0	Used for TMR16A: General Timer
	TMR16A1	Not used
LVD	-	Used for LVD demo
10-bit A/D converter	AIN0	Used for ADC demo
	AIN1	Unused
	AIN2	Unused
	AIN3	Unused
	AIN4	Unused
	AIN5	Unused
	AIN6	Unused
	AIN7	Unused
DMAC	-	Used for DMAC demo
I2C	I2C0	Used for I2C demo: receiver, transmitter

4 Pin Usage

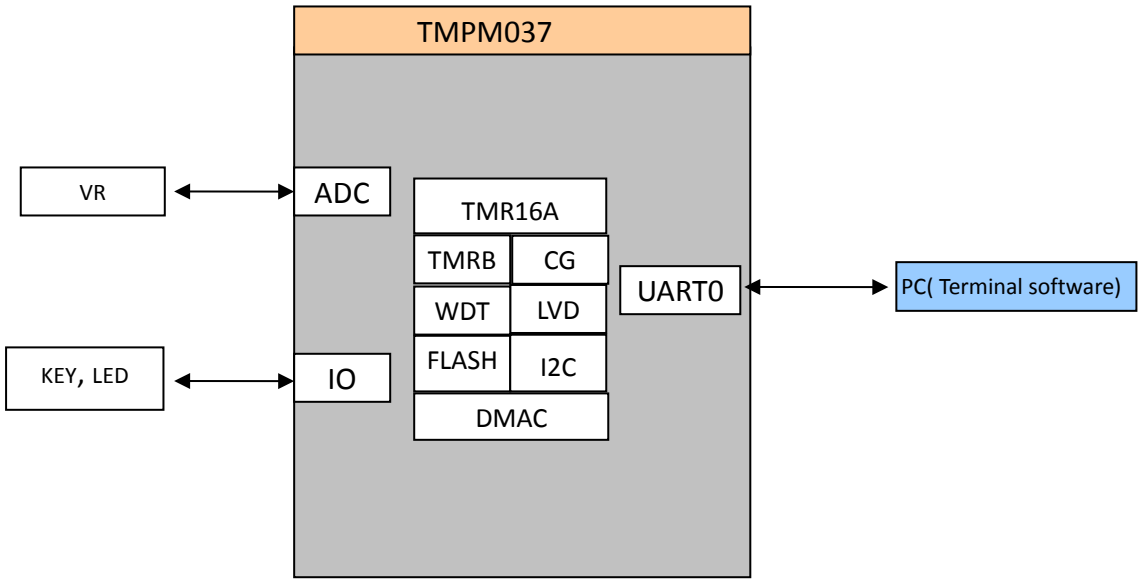
The example programs are tested on TMPM037FWUG evaluation board
Following is pin usage for example programs

Pin No.	Name	Usage
1	PD0	Unused
2	PD1	SC0SCLK
3	PD2	SC0RXD
4	PD3	SC0TXD
5	PD4	Unused
6	PD5	Unused
7	AVDD3	Power supply pin for the analog circuit.
8	AVSS	GND pin for the analog circuit.
9	PA0	AIN0
10	PA1	Unused
11	PA2	Unused
12	PA3	Unused
13	PA4	Unused
14	PA5	Unused
15	PA6	Unused
16	PA7	Unused
17	PF7	SW3
18	PF6	SW2
19	PF5	SW1
20	PF4	SW0
21	PF3	SC3TXD
22	PF2	SC3RXD
23	PF1	Unused
24	PF0	Unused
25	PC5	LED3
26	PC4	LED2
27	PC3	TB0OUT/LED1
28	PC2	LED0
29	DVSS	GND pin for the digital circuit.
30	DVDD3	Power supply pin for the digital circuit.
31	PC1	I2C0SDA
32	PC0	I2C0SCL
33	PG7	Unused
34	PG6	Unused
35	PG5	Unused

36	PG4	Unused
37	PG3	Unused
38	PG2	SC1TXD
39	PG1	SC1RXD
40	PG0	SC1SCLK
41	PE0	DipSW0
42	PE1	DipSW1
43	PE2	Unused
44	PE3	SC2RXD/USB_RXD
45	PE4	SC2TXD/USB_TXD
46	PE5	Unused
47	PE6	Unused
48	PE7	Unused
49	PB7	Unused
50	PB6	Unused
51	PB5	INT0
52	PB4	Unused
53	PB3	Unused
54	PB2	Unused
55	PB1	Unused
56	PB0	BOOT
57	DVDD3	Power supply pin for the digital circuit.
58	X1	Connected to a high-speed oscillator.
59	X2	Connected to a high-speed oscillator.
60	DVSS	GND pin for the digital circuit.
61	REGOUT	Pin connected with the capacitor for the regulator
62	RESET	Reset input pin
63	MODE	MODE pin MODE pin must be connected to GND.
64	RVDD3	Power supply pin for the regulator.

5 Development Environment

Following is development environment:



1. Hardware board:
   TMPM037FWUG evaluation board

2. Development tool:
   - IAR:
     1) IDE: IAR Embedded workbench 7.8.0.3.12146 version
   - KEIL:
     1) IDE: KEIL uVision 5.24.2.0

   - ICE:
     CMSIS-DAP(on-board)

6 Functional description

6-1 Operation mode

There are three operation modes for TMPM037FWUG: NORMAL, IDLE, STOP1 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.

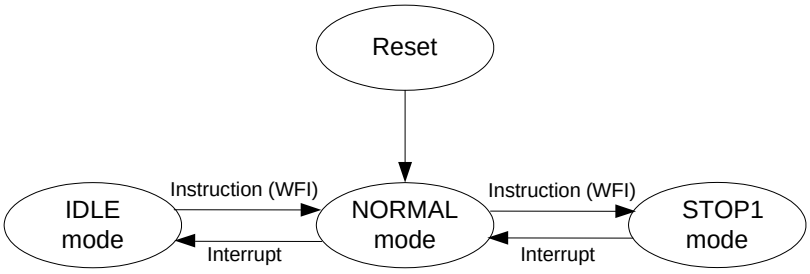
IDLE and STOP1 modes are low power mode.
To shift to lower power mode, in system control register CGSTBYCR<STBY[2:0]>, select the IDLE or STOP1 mode, and run the WFI (Wait For Interrupt) command.
***Note:** Only STOP1 mode is demonstrated in driver example.

- **STOP1 mode:**

All the internal circuits including the internal oscillator are brought to a stop in STOP1 mode. And the internal oscillator activates the clock after releasing the STOP1 mode then transit to the Normal mode. The STOP1 mode enables to select the pin status by setting the port register.
***Note:** The sample program of STOP1 mode is integrated into CG example.

- **IDLE mode:**

Only the CPU is stopped in this mode. Each peripheral function has one bit in its control register for enabling or disabling operation in the IDLE mode. When the IDLE mode is entered, peripheral functions for which operation in the IDLE mode is disabled stop operation and hold the state at that time.



6-2 ADC

There is a potentiometer that connected to PA0/AIN0. The voltage on it will be measured 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 STOP1

Change the CPU operation mode. Only 2 modes are supported, NORMAL and STOP1. Toggle switch to switch the power mode.

Current mode	Action (Switch)	Operation	LED display
NORMAL	Turn on SW1	NORMAL→ STOP1	UART prints “NORMAL MODE” →”STOP MODE” LED 0,1,2,3 turns off.
STOP1	Turn off SW1 at first, Turn on SW0	STOP1 →NORMAL	UART prints “STOP MODE” →” NORMAL MODE” LED 0,1,2,3 turns on.

6-3-2 IDLE

Change the CPU operation mode between normal mode and Idle mode.

Current mode	Action (Switch)	Operation	LED display
NORMAL	Turn on SW1	NORMAL→ IDLE	UART prints ”IDLE MODE” LED 0,1,2,3 turns on.
IDLE	Turn off SW1 at first, then turn on SW0	IDLE →NORMAL	UART prints “NORMAL MODE” LED 0,1,2,3 turns off.

6-4 DMAC

This function implements transmission from UART0 to UART1, the string “TOSHIBA” is sent via UART0 to UART1.
The string is transferred from a RAM area to UART0 data register by DMAC, each character of the string is sent via UART0 and received by UART1. After UART1 received the data, it is saved in a character array.

The received value can be checked by RAM variable, the character array in debugger.

***Note:** In order to run this sample software, the IAR TMPM037FWUG Evaluation Board must be modified: Connect the TX of UART0 and RX of UART1 via wire.

6-5 FLASH

This example demonstrates the feature of flash APIs such as erase and writes 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) is programmed in flash block in advance.

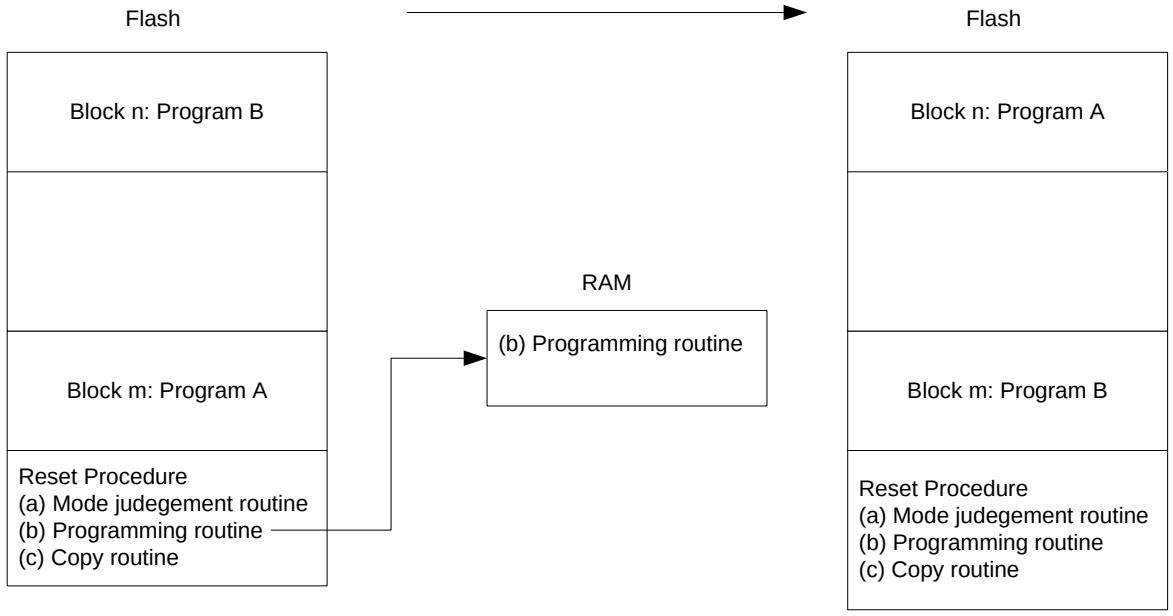
—Program A/B (A: LED0 blinks, B: LED2 blinks)

By default, the program A will run firstly

—Toggle switch SW0 is used for mode judgment in Reset procedure

SW0 turned on → User boot mode

SW0 turned off → Normal mode



Demo sequence

(1) Power on

The demo board runs Reset Procedure.

Then initial program A stored in flash runs.

LED0 blinks.

(2) Press RESET button while SW0 is turned on, and release SW0.

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.

LED3 lights, when finished, LED3 turns off and LED1 lights

(3) The demo will reset by software automatically.

Then program B stored in flash runs.

LED2 blinks.

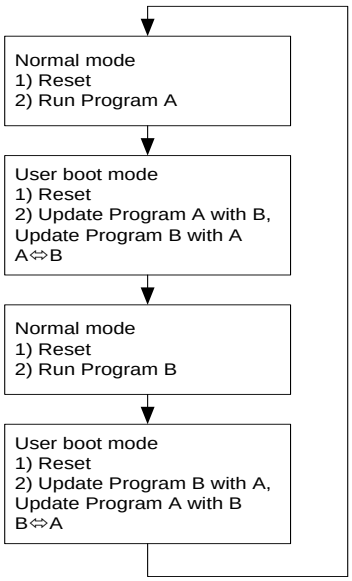
(4) Press RESET button again while SW0 is turned on, and release SW0.

Same as step (2).

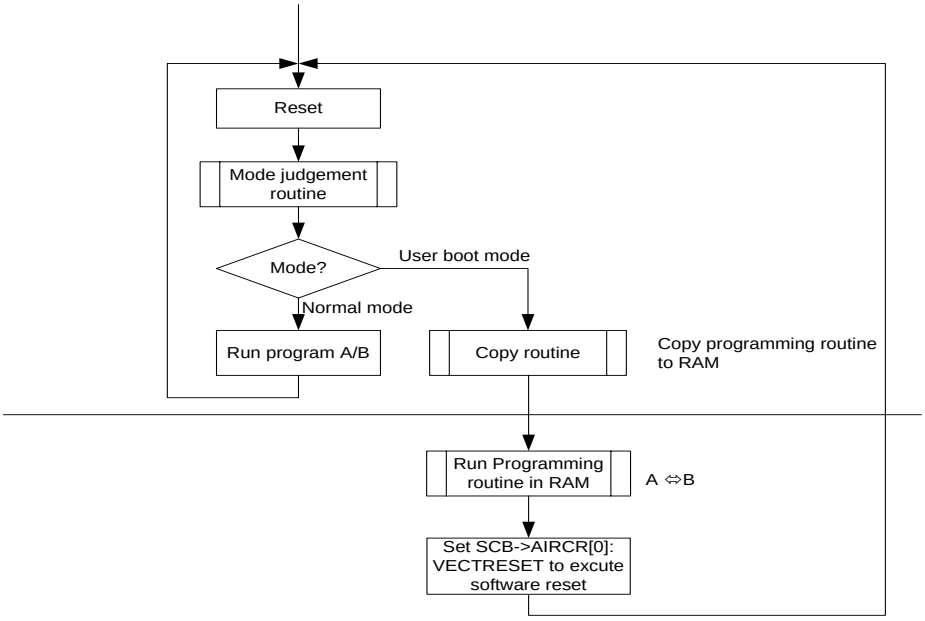
(5) The demo will reset by software automatically.

Then program A stored in flash runs.

LED0 blinks.



Demo process flow



6-6 GPIO

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

6-7 I2C

6-7-1 I2C Slave

This function intends to support the I2C bus slave mode.
Connect two I2C buses on evaluation board.
Work as I2C slave.

Use I2C interrupt to handle I2C bus read.
If the received address matches its slave address specified at I2CAR or is equal to the general-call address, the I2C pulls the SDA line to the "Low" level during the ninth clock and outputs an acknowledge signal.

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

I2C (slave) demo start,

```
K1: I2C SLAVE RECV
```

When SW1 is pressed, the string “TOSHIBA” that got from the I2C (slave) received buffer will be printed.

```
TOSHIBA
MASTER I2C to SLAVE I2C OK
```

6-7-2 I2C Master

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

Use I2C interrupt to handle I2C bus write.

Address of I2C: Any.
I2C Master sends “TOSHIBA” to Slave (I2C).

I2C (master) demo start,
When SW3 is pressed, the string “TOSHIBA” will be sent from I2C (master) to I2C (slave).

6-8 LVD

This application implements power supply voltage status detecting by LVD.
When the power supply voltage is lower than the detection voltage, UART prints “LOWER”.
When the power supply voltage is upper than the detection voltage, UART prints “UPPER”.

6-9 TMR16A

Set a value of T16A0RG<RG>, then start count-up, if the counter value matches with a value of T16A0RG<RG>, the counter will be cleared to "0x0000" and continued to count-up and the same time a match detection interrupt INTT16A0 will be output.

Timer cycle is set to 1ms.Use this timer for LED blinking and the period is 1s (500ms ON

and 500ms OFF).

6-10 TMRB

6-10-1 General Timer

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

6-10-2 PPG Output

Use key SW1 to change PPG waveform. LeadingTiming can be changed as following:
10%, 25%, 50%, 75%, 90%
Power on to enter PPG mode and starts the PPG output.
Change the leadingtiming upon every SW1 pressed.
10% → 25% → 50% → 75% → 90% → 10%

***Note**: PC3 connect with oscilloscope to observe the PPG output.

6-11 SIO/UART

6-11-1 Retarget

This example intends to retarget the stdin and stdout of the C lib.
Stdin and stdout are retargeted to UART. Then application code can use printf() to output to serial port.
***Note**: This UART channel is limited by hardware, and the follow example used UART2.

6-11-2 UART FIFO

In this sample program, UART0 sent the data "TMPM0371" to UART3 use FIFO, and at same time UART0 receive the data "TMPM0372" which send from UART3 and also use FIFO.
Set one breakpoint before the function ResetIdx(),when program stop in the breakpoint you can read the RxBuffer = "TMPM0372",and RxBuffer1 = "TMPM0371".

6-11-3 SIO

This demo will show synchronously transfer and receive usage of SIO module in TMPM037FWUG.
It uses the channel SIO0, SIO1 and transfer data synchronously between them. (Connect TXD0 with RXD1; connect TXD1 with RXD0; connect sclk0 with sclk1)

6-12 WDT

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

The driver example step:

1. Initialize WDT by setting detection time and selecting WDT interrupt as counter overflow operation.
2. There are two demos for WDT and DEMO2 is defined by macro definition.
   DEMO1:
   When timer is overflow, NMI interrupt is generated (LED1 blink once) 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. (LED0 blink all the time)

7 Software

This software project creates the sample applications based on TMPM037FWUG evaluation board which will demonstrate the main feature of TMPM037FWUG 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
        │   tmpm037_wdt_int.c
        │   tmpm037_wdt_int.h
```

```
│
├─TX00_CMSIS
│ │ system_TMPM037.c
│ │ system_TMPM037.h
│ │ TMPM037.h
│ │
│ └─startup
        startup_TMPM037.s
│
├─TX00_Periph_Driver
│   ├─inc
│   │   tmpm037_cg.h
│   │   tmpm037_gpio.h
│   │   tmpm037_wdt.h
│   │   TX00_common.h
│   │
│   └─src
│       tmpm037_cg.c
│       tmpm037_gpio.c
│       tmpm037_wdt.c
│
└─TMPM037-EVAL
        led.c
        led.h
        sw.c
        sw.h
```

KEIL MDK:

```
├─WDT_NMI
    ├─APP
    │   main.c
    │   tmpm037_wdt_int.c
    │
    ├─TX00_CMSIS
    │    system_TMPM037.c
    │    startup_TMPM037.s
    │
    ├─TX00_Periph_Driver
    │    tmpm037_cg.c
    │    tmpm037_gpio.c
    │    tmpm037_wdt.c
    │
    └─TMPM037-EVAL
```

led.c
sw.c

7-1 ADC

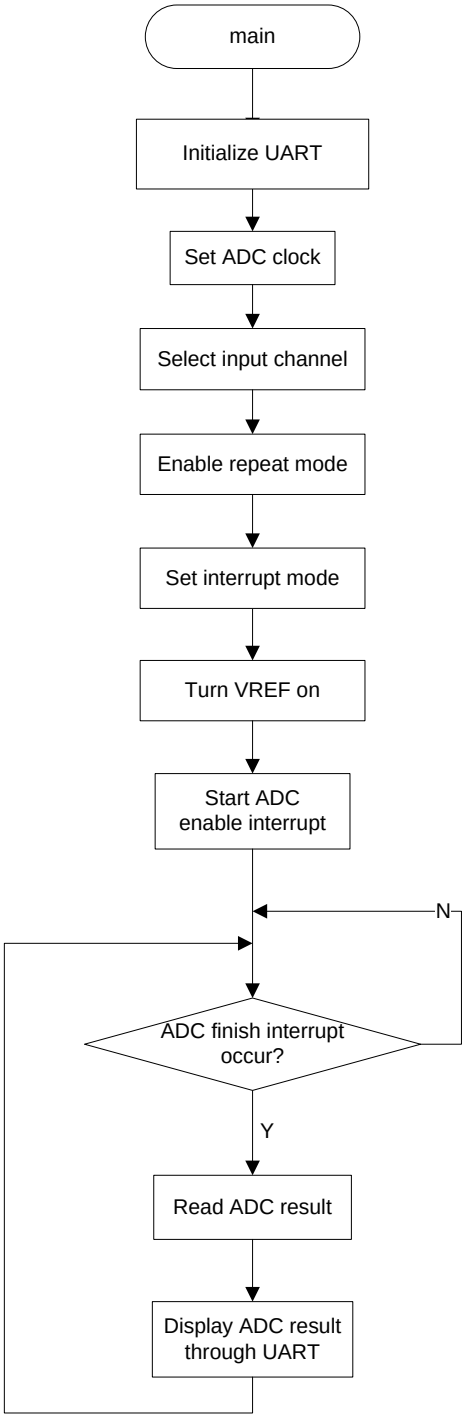
7-1-1 Example: ADC Data Read

This is a simple example based on the TX00 Peripheral Driver (ADC, GPIO and UART).

The example includes:

1. ADC configuration and initialization
2. Start ADC in fixed-channel repeat mode and read AD result of AIN0

- **Flowchart**



- **Code and Explanation for the Example**

At first, set ADC clock, select input channel, enable repeat mode, set interrupt mode and turn VREF on.

```
  /* set ADC clock */
ADC_SetClk(ADC_CONVERSION_42_CLOCK, ADC_FC_DIVIDE_LEVEL_16);

  /* select ADC input channel */
  ADC_SetInputChannel(ADC_AN_0);
```

```
/* Enable ADC repeat mode */
ADC_SetRepeatMode(ENABLE);

/* Set interrupt mode */
ADC_SetINTMode(ADC_INT_CONVERSION_8);

/* Turn VREF on */
ADC_SetVref(ENABLE);

/* Wait at least 3us to ensure the voltage is stable */
Delay(10U);
```

Then start ADC:

```
ADC_Start();

/* enable AD interrupt */
NVIC_EnableIRQ(INTAD_IRQn);
```

After started ADC, wait for INTAD. When INTAD occurs, call ADC_Display() function.

```
while (1) {
    if (fIntADC == 1U) {
        fIntADC = 0U;
        ADC_Display();
    }
}
```

In ADC_Display() function, read corresponding ADREG to get ADC result.

```
ADC_ResultTypeDef ADC_Result = ADC_GetConvertResult(ADC_REG_0);
```

7-2 CG

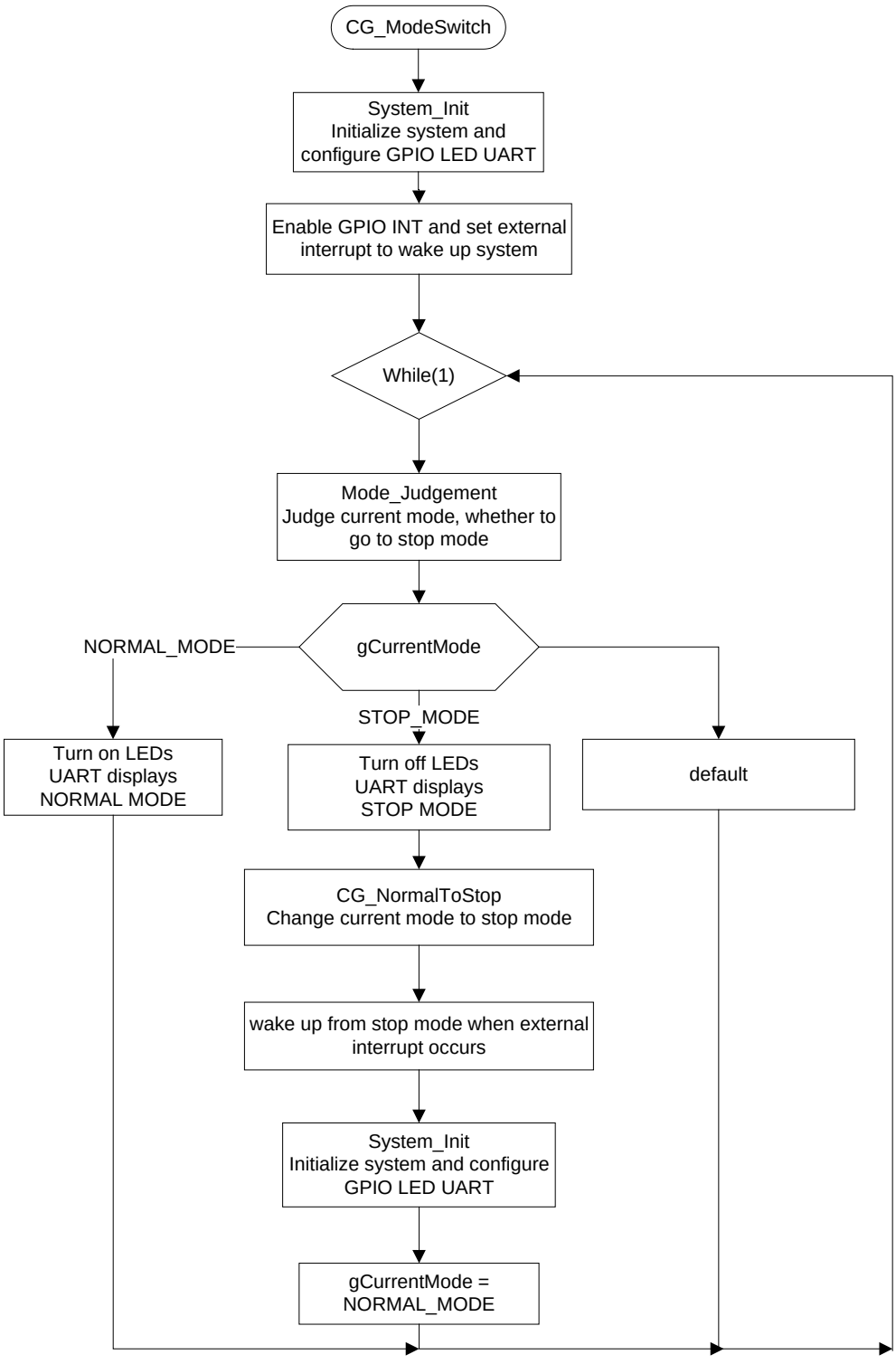
7-2-1 Example: NORMAL <-> STOP1 mode change

This is a simple example based on the TX00 Peripheral Driver (CG, GPIO and UART).

The example includes:

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

- **Flowchart**



- **Code and Explanation for the Example**

The following simple example is based on TX00 Peripheral Driver (CG, GPIO and UART), 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 16MHz.

Example code is as the following:

```
if (CG_GetFoscSrc()==CG_FOSC_OSC_INT){
    /* Switch over from IHOSC to EHOSC*/
    switchFromIHOSCtoEHOSC();
}
/* Disable PLL. PLL cannot be used, since fosc = 16MHz that larger than 10 MHz. */
CG_DisableClkMulCircuit();
/* Set fgear = fc/2 */
CG_SetFgearLevel(CG_DIVIDE_2);
/* Set fperiph to fgear */
CG_SetPhiT0Src(CG_PHIT0_SRC_FGEAR);
/* Set ΦT0 = fc/2 */
CG_SetPhiT0Level(CG_DIVIDE_2);
/* Set low power consumption mode stop1 */
CG_SetSTBYMode(CG_STBY_MODE_STOP1);
```

Configure external interrupt to wake up system

Confiure external interrupt INT0 to wake up system. Clear interrupt pending request, then enable INT0.

```
__disable_irq();
CG_ClearINTReq(CG_INT_SRC_0);
/* Set external interrupt to wake up system */
CG_SetSTBYReleaseINTSrc(CG_INT_SRC_0,
                        CG_INT_ACTIVE_STATE_FALLING, ENABLE);
NVIC_ClearPendingIRQ(INT0_IRQn);
NVIC_EnableIRQ(INT0_IRQn);
__enable_irq();
```

Setup to enter STOP mode

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

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

Disable multiple clock circuit

PLL cannot be used, since fosc = 16MHz that larger than 10 MHz.

```
Result retval = ERROR;
WorkState st = BUSY;
retval = CG_SetFcSrc(CG_FC_SRC_FOSC);
if (retval == SUCCESS) {
    /* Set warm up time */
    CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC_EXT_HIGH,
                                                CG_WUODR_FOSC);
    CG_StartWarmUp();
    do {
        st = CG_GetWarmUpState();
    } while (st != DONE);
    /* fc source is set to fosc, so PLL can be disabled successfully. */
    retval = CG_SetPLL(DISABLE);
} else {
```

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

7-2-2 Example: NORMAL <-> IDLE mode change

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

The example includes:

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

- **Code and Explanation for the Example**

Configure the I/O pins for SW, LED and UART:

```
TMRB_InitTypeDef m_tmrb;

SW_Init();
LED_Init();
hardware_init(UART_RETARGET);
```

The following sets the timing of interrupt generation.

```
m_tmrb.Mode = TMRB_INTERVAL_TIMER;
m_tmrb.ClkDiv = TMRB_CLK_DIV_512;
m_tmrb.TrailingTiming = TMRB_1MS;     /* periodic time is 1ms */
m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;
m_tmrb.LeadingTiming = TMRB_1MS;      /* periodic time is 1ms */
```

In this sample program, the low power consumption mode is released by TMRB interrupt generation. The following is the setting.

```
TMRB_Enable(TSB_TB0);
TMRB_Init(TSB_TB0, &m_tmrb);
TMRB_SetINTMask(TSB_TB0,
                        TMRB_MASK_MATCH_LEADINGTIMING_INT);
TMRB_SetIdleMode(TSB_TB0, ENABLE);

NVIC_ClearPendingIRQ(ExtINTSrc_IRQn);
NVIC_EnableIRQ(INTTMRB0_IRQn);
```

When the status of SW1 is OFF, turn off LED_ALL.
When the status of SW1 is ON, turn on LED_ALL. After stopping and clearing the up counter, enter IDLE mode after turn on LED_ALL.

```
while (1U) {
    if (SW_Get(SW1) == 1U) {
        LED_On(LED_ALL);
        common_uart_disp("IDLE MODE¥r¥n");
        Delay(); /* Delay sometime to let UART print string. */
        TMRB_SetRunState(TSB_TB0, TMRB_STOP);
        TMRB_SetRunState(TSB_TB0, TMRB_RUN);
        enter_IDLE();
        LED_Off(LED_ALL);
        common_uart_disp("NORMAL MODE¥r¥n");
```

```
    } else {
            LED_Off(LED_ALL);
            common_uart_disp("NORMAL MODE¥r¥n");
      }
    }
```

Prepare to enter IDLE mode.
Select IDLE mode as a standby mode.

```
    /*Set standby mode as IDLE */
    CG_SetSTBYMode(CG_STBY_MODE_IDLE);
```

Finally, enter IDLE mode by using __WFI() instruction.

```
     __WFI();
```

7-3 DMAC

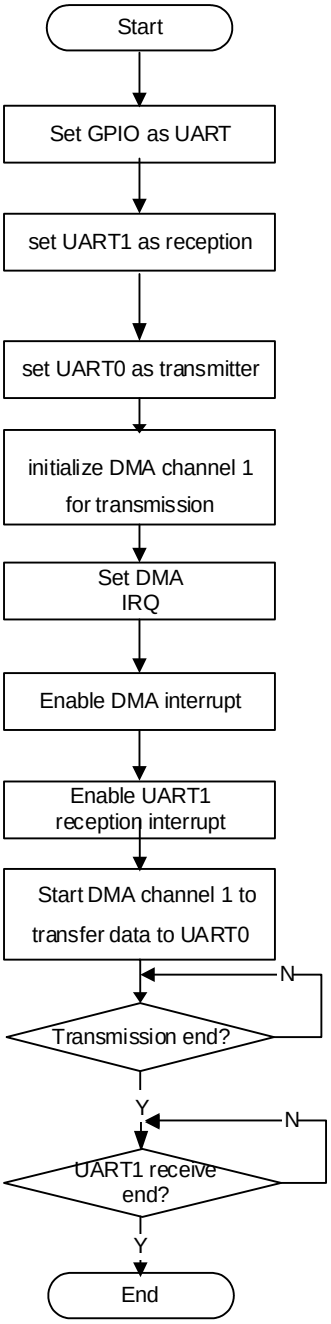
7-3-1 Example: DMA transfer from Memory to peripheral

This is a simple example based on the TX00 Peripheral Driver (DMAC, GPIO, SIO).

The example includes:

1. UART0/1 configuration and DMAC initialization
2. Transfer data from memory to UART0 via DMAC

- **Flowchart:**



- **Code and Explanation for the Example**

The following simple example is based on TX00 Peripheral Driver (DMAC, GPIO, SIO), which will transfer data from memory to UART0.

Firstly set UART0 as transmitter and enable it

```
UART_Rx_Cnt = 0U;
UART_InitStruct.BaudRate = 115200U;
UART_InitStruct.DataBits = UART_DATA_BITS_8;
UART_InitStruct.StopBits = UART_STOP_BITS_1;
UART_InitStruct.Parity = UART_NO_PARITY;
UART_InitStruct.Mode = UART_ENABLE_RX;
```

```
    UART_InitStruct.FlowCtrl = UART_NONE_FLOW_CTRL;
    UART_Enable(UART1);
    UART_Init(UART1, &UART_InitStruct);

    UART_InitStruct.Mode = UART_ENABLE_TX;
    UART_Enable(UART0);
    UART_Init(UART0, &UART_InitStruct);
    UART_SetTxDMAReq(UART0, ENABLE);
```

Configure and initialize DMA channel1 for transmission

```
    DMAC_InitStruct.SrcAddr = (uint32_t) SRC_Buffer;
    DMAC_InitStruct.SrcIncrementState = ENABLE;
    DMAC_InitStruct.SrcBitWidth = DMAC_BYTE;
    DMAC_InitStruct.SrcBurstSize = DMAC_1_BEAT;
    DMAC_InitStruct.DstAddr = (uint32_t) (UART0_BASE_ADDR +
      UART0_BUF_OFFSET);
    DMAC_InitStruct.DstIncrementState = DISABLE;
    DMAC_InitStruct.DstBitWidth = DMAC_BYTE;
    DMAC_InitStruct.DstBurstSize = DMAC_1_BEAT;
    DMAC_InitStruct.TxSize = size;
    DMAC_InitStruct.TxDirection = DMAC_MEMORY_TO_PERIPH;
    DMAC_InitStruct.TxDstPeriph = DMAC_SIO0_UART0_TX;
    DMAC_InitStruct.TxINT = ENABLE;

    DMAC_Init(myChannel, &DMAC_InitStruct);
```

Enable DMA IRQ

```
    NVIC_EnableIRQ(INTDMAC_IRQn);
```

Enable DMA circuit, transfer end interrupt, burst transfer request for UART0 and start channel 1 of DMAC to transfer

```
    DMAC_Enable();
    DMAC_SetTxINTConfig(myChannel, DMAC_INT_TX_END, ENABLE);
    DMAC_SetSWBurstReq(DMAC_SIO0_UART0_TX);
    DMAC_SetDMAChannel(myChannel, ENABLE);
```

Wait the end of DMAC transmission

```
    while (TxEndFlag != DONE) {
        /* Do nothing */
    }
```

Set flag for DMAC transmission end in DMAC ISR

```
    state = DMAC_GetINTReq();
    if (state.Bit.CH1_INTReq == 1U) {
        tmp = DMAC_GetTxINTReq(DMAC_CHANNEL_1);
        if (tmp == DMAC_TX_END_REQ) {
            TxEndFlag  =  DMAC_GetChannelTxState(DMAC_CHANNEL_1);
            DMAC_ClearTxINTReq(DMAC_CHANNEL_1,  DMAC_INT_TX_END);
        } else {
            /* Do nothing */
        }
    } else {
        /* Do nothing */
```

```
}
```

7-4 FLASH

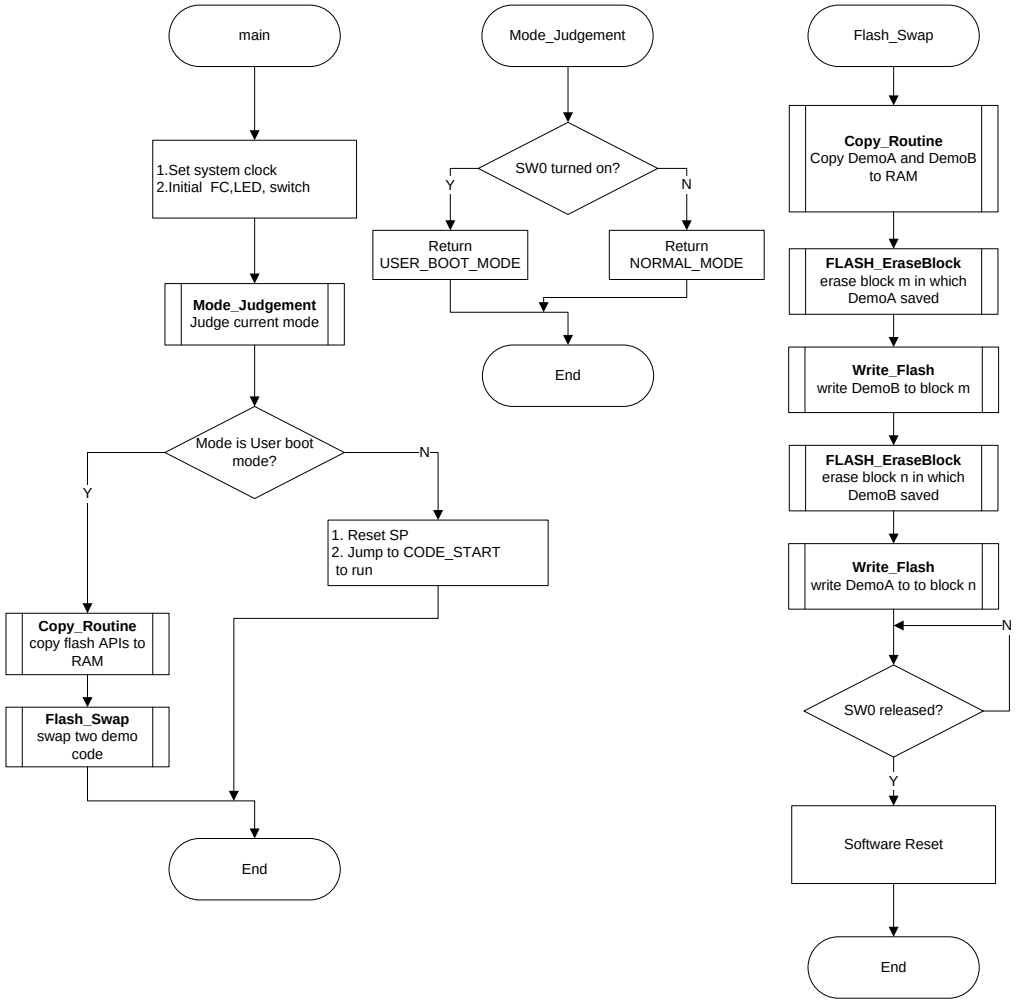
7-4-1 Example: Flash_UserBoot

This is a simple example based on the TX00 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 LED and switch. SW0 (GPIO) is used to judge current mode when reset .

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

Use function Mode_Judgement() to judge which mode will be entered after reset.

```
uint8_t Mode_Judgement(void)
{
    return (SW_Get(SW0) == 1U) ? USER_BOOT_MODE : NORMAL_MODE;
}
```

If the SW0 is not turned on 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 in

at address “CODE_START” which belongs to block m, so Demo A will run (LED0 blinks).

```
#if defined ( __CC_ARM     )          /* RealView Compiler */
            ResetSP();                    /* reset SP */
#elif defined ( __ICCARM__ )       /* IAR Compiler */
            c_stack = 0;
            __set_MSP(*c_stack);    /* reset SP */
#endif
            startup = CODE_START;
            startup();                    /* jump to code start address to run */
```

If the SW0 is turned on when reset, the routine will enter user boot 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.

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

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

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

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

After SW0 released, it will execute software reset by using NVIC_SystemReset(). 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 (LED1 blinks,LED3 Always show).

```
    while (SW_Get(SW0) == 1U) {
    }
    /* software reset */
    NVIC_SystemReset();
```

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

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

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

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

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

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

Function Write_Flash() will call FLASH_WritePage() to write data automatically in 16 bytes. The process of this function is basically same as FLASH_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 < PROGRAM_UNIT; i++) {
    *PA = *source;
    source++;
}
```

7-5 GPIO

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

7-5-1 Example: GPIO Data Read

This is a simple example based on the TX00 Peripheral Driver (GPIO).

The example includes:

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

- **Code and Explanation for the Example**

At first, use GPIO_SetOutput(GPIO_Port GPIO_x, uint8_t Bit_x) to configure GPIO

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

```
    GPIO_SetOutput(GPIO_PC,  GPIO_BIT_2  |  GPIO_BIT_3  |  GPIO_BIT_4  |
GPIO_BIT_5);

    GPIO_SetInput(GPIO_PF,    GPIO_BIT_4  |  GPIO_BIT_5  |  GPIO_BIT_6  |
GPIO_BIT_7);
```

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

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

```
    if (GPIO_ReadDataBit(GPIO_PF, GPIO_BIT_4) == 1U) {
                    tmp = 1U;
    } else {
                  /*Do nothing */
    }
```

Turn on LED by using GPIO_WriteData(GPIO_Port GPIO_x, uint8_t Data)

```
    uint8_t tmp;
    tmp = GPIO_ReadData(GPIO_PC);
    tmp |= led;
    GPIO_WriteData(GPIO_PC, tmp);
```

Turn off LED by using GPIO_WriteData(GPIO_Port GPIO_x, uint8_t Data)

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

7-6 I2C

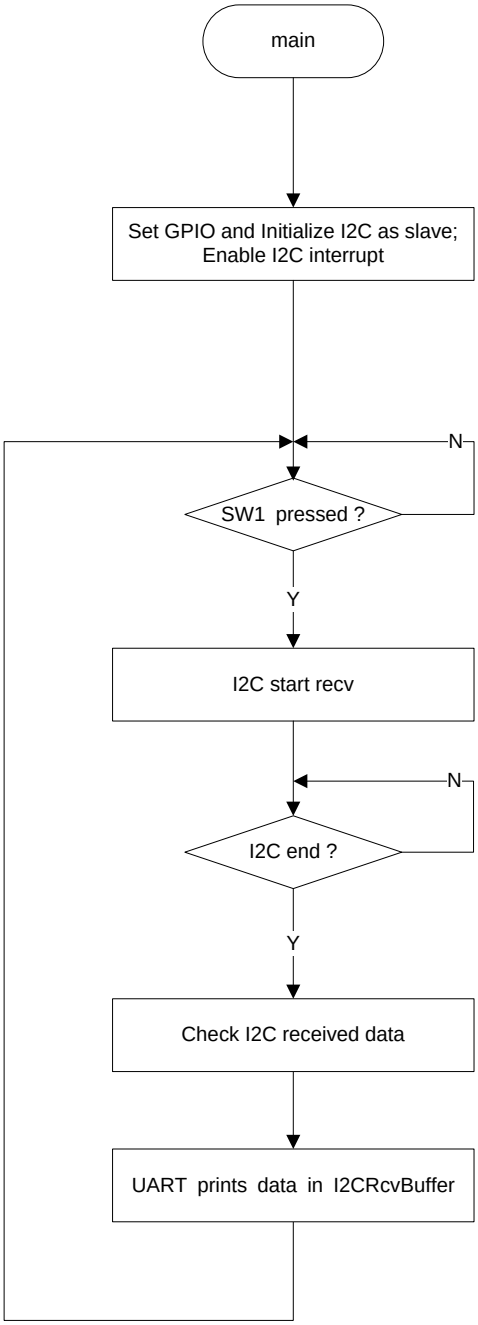
7-6-1 Example: I2C Slave

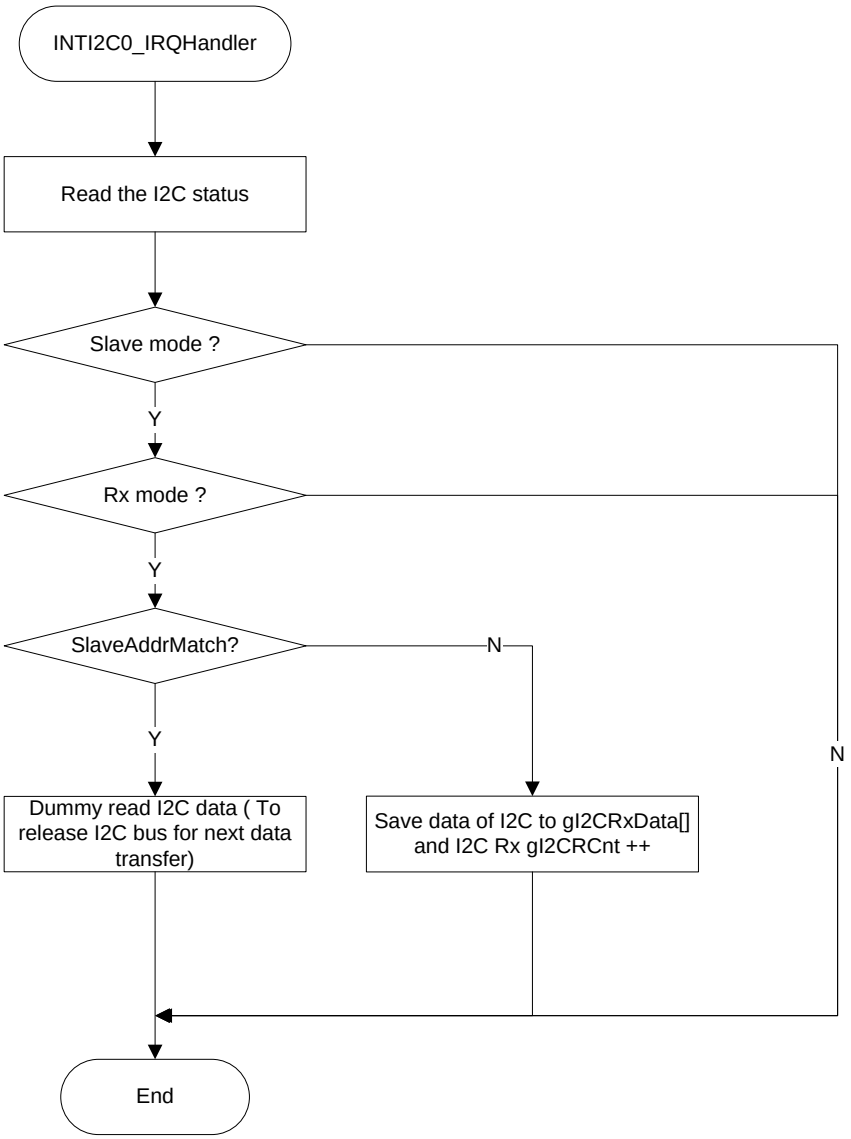
This is a simple example based on the TX00 Peripheral Driver (I2C, GPIO).

The example includes:

1. I2C configuration
2. I2C slave receive data process

- **Flowchart**


main
Set GPIO and Initialize I2C as slave;
Enable I2C interrupt
N
SW1 pressed ?
Y
I2C start recv
N
I2C end ?
Y
Check I2C received data
UART prints data in I2CRcvBuffer



- **Code and Explanation for the Example**

At first, configure GPIO for I2C mode.

```
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_1, GPIO_BIT_0);
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_1, GPIO_BIT_1);
GPIO_SetOutputEnableReg(GPIO_PC, GPIO_BIT_0, ENABLE);
GPIO_SetOutputEnableReg(GPIO_PC, GPIO_BIT_1, ENABLE);
GPIO_SetInputEnableReg(GPIO_PC, GPIO_BIT_0, ENABLE);
GPIO_SetInputEnableReg(GPIO_PC, GPIO_BIT_1, ENABLE);
GPIO_SetOpenDrain(GPIO_PC, GPIO_BIT_0, ENABLE);
GPIO_SetOpenDrain(GPIO_PC, GPIO_BIT_1, ENABLE);
```

Then enable, initialize and configure slave I2C channel and enable INTI2C0.

```
myI2C.I2CSelfAddr = SLAVE_ADDR;
myI2C.I2CDataLen = I2C_DATA_LEN_8;
myI2C.I2CACKState = ENABLE;
```

```
    myI2C.I2CClkDiv = I2C_CLK_DIV_32;
    myI2C.PrescalerClkDiv = I2C_PRESCALER_DIV_12;
    I2C_SWReset();
    I2C_Init(&myI2C);
    NVIC_EnableIRQ(INTI2C0_IRQn);
    I2C_SetINTReq(ENABLE);
```

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

```
    /* Initialize TRx buffer and Tx length */
    case MODE_I2C_INITIAL:
        for (gICnt = 0U; gICnt < 8U; gICnt++) {
            gI2CRxData[gICnt] = 0U;
        }
        gI2CMode = MODE_I2C_RCV;
        break;
```

The data receive is handled in INTI2C0.

In INTI2C0 function, I2C slave receive process is handled according to I2C bus state, I2C_GetReceiveData() is used to read received data in I2C buffer, I2C stop condition is controlled by master.

```
void INTI2C0_IRQHandler(void)
{
    uint32_t tmp = 0U;
    I2C_State i2c0_sr;

    i2c0_sr = I2C_GetState();
    if (!i2c0_sr.Bit.MasterSlave) {        /* Slave mode */
        if (!i2c0_sr.Bit.TRx) { /* Rx Mode */
            if (i2c0_sr.Bit.SlaveAddrMatch) {
                /* First read is dummy read for Slave address recognize */
                tmp = I2C_GetReceiveData();
                gI2CRCnt = 0U;
                tmp = 0U;
                I2C_SetSendData(tmp);
            } else {
                /* Read I2C received data and save to I2C_RxData buffer */
                tmp = I2C_GetReceiveData();
                i2c0_sr = I2C_GetState();
                gI2CRxData[gI2CRCnt] = tmp;
                gI2CRCnt++;
                tmp = 0U;
                I2C_SetSendData(tmp);
```

```
            }
        } else {                    /* Tx Mode */
            /* Do nothing */
        }
    } else {                        /* Master mode */
        /* Do nothing */
    }
}
```

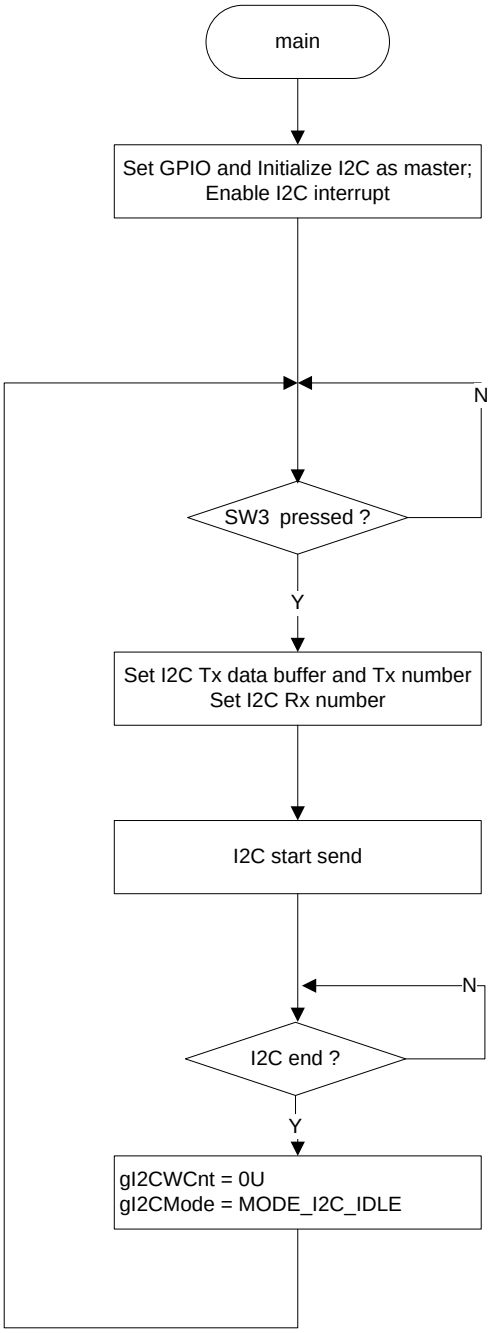
7-6-2 Example: I2C Master

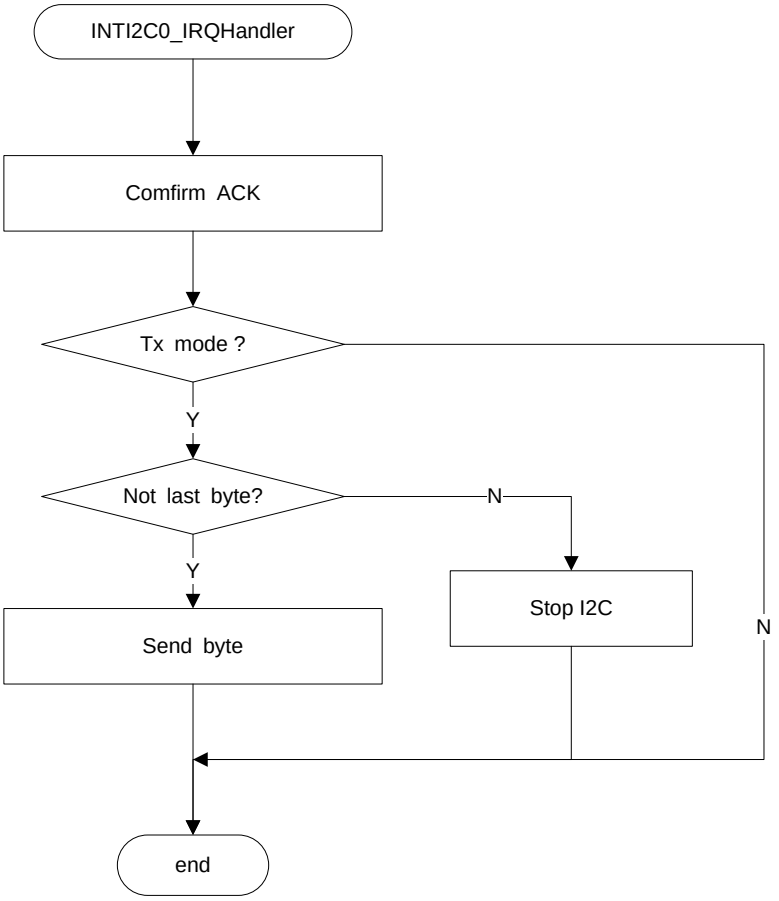
This is a simple example based on the TX00 Peripheral Driver (I2C, GPIO).

The example includes:

1 I2C configuration
2 I2C master send data process

- **Flowchart**


main
Set GPIO and Initialize I2C as master;
Enable I2C interrupt
N
SW3 pressed ?
Y
Set I2C Tx data buffer and Tx number
Set I2C Rx number
I2C start send
N
I2C end ?
Y
gI2CWCnt = 0U
gI2CMode = MODE_I2C_IDLE



- **Code and Explanation for the Example**

At first, configure GPIO for I2C mode.

```
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_1, GPIO_BIT_0);
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_1, GPIO_BIT_1);
GPIO_SetOutputEnableReg(GPIO_PC, GPIO_BIT_0, ENABLE);
GPIO_SetOutputEnableReg(GPIO_PC, GPIO_BIT_1, ENABLE);
GPIO_SetInputEnableReg(GPIO_PC, GPIO_BIT_0, ENABLE);
GPIO_SetInputEnableReg(GPIO_PC, GPIO_BIT_1, ENABLE);
GPIO_SetOpenDrain(GPIO_PC, GPIO_BIT_0, ENABLE);
GPIO_SetOpenDrain(GPIO_PC, GPIO_BIT_1, ENABLE);
```

Then enable, initialize and configure slave I2C channel and enable INTI2C0.

```
myI2C.I2CSelfAddr = SELF_ADDR;
myI2C.I2CDataLen = I2C_DATA_LEN_8;
myI2C.I2CACKState = ENABLE;
myI2C.I2CClkDiv = I2C_SCK_CLK_DIV_32;
```

```
myI2C.PrescalerClkDiv = I2C_PRESCALER_DIV_12;
I2C_SWReset();
I2C_Init(&myI2C);
NVIC_EnableIRQ(INTI2C0_IRQn);
I2C_SetINTReq(ENABLE);
```

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

```
/* Initialize TRx buffer and Tx length */
case MODE_I2C_INITIAL:
    gI2CTxDataLen = 7U;
    gI2CTxData[0] = gI2CTxDataLen;
    gI2CTxData[1] = 'T';
    gI2CTxData[2] = 'O';
    gI2CTxData[3] = 'S';
    gI2CTxData[4] = 'H';
    gI2CTxData[5] = 'I';
    gI2CTxData[6] = 'B';
    gI2CTxData[7] = 'A';

    gI2CWCnt = 0U;
    gI2CMode = MODE_I2C_START;
    break;
```

Check if the I2C bus is free or not, I2C_SetSendData() is used to set data “SLAVE_ADDR”.and direction is “I2C_SEND” to I2C data buffer; then I2C_GenerateStart() is used to to start I2C process.

```
/* Check I2C bus state and start TRx */
case MODE_I2C_START:
    i2c_state = I2C_GetState();
    if (!i2c_state.Bit.BusState) {
        gI2CMode = MODE_I2C_TRX;
        I2C_SetSendData(SLAVE_ADDR | I2C_SEND);
        I2C_GenerateStart();
    } else {
        /* Do nothing */
    }
    break;
```

The data transfer is handled in INTI2C0.

In INTI2C0 function, I2C master send process is handled according to I2C bus state. During I2C master sending, I2C_SetSendData() is used to send next data, I2C_GenerateStop() is used to stop I2C when I2C process is finished.

```
void INTI2C0_IRQHandler(void)
{
	I2C_State i2c_sr;
	i2c_sr = I2C_GetState();
	if (i2c_sr.Bit.MasterSlave) {		/* Master mode */
		if (i2c_sr.Bit.TRx) {	/* Tx mode */
			if (i2c_sr.Bit.LastRxBit) { /* LRB=1: the receiver requires no further data. */
				I2C_GenerateStop();
			} else {			/* LRB=0: the receiver requires further data. */
				if (gI2CWCnt <= gI2CTxDataLen) {
					I2C_SetSendData(gI2CTxData[gI2CWCnt]);		/* Send next data */
					gI2CWCnt++;
				} else {		/* I2C data send finished. */
					I2C_GenerateStop(); /* Stop I2C */
					gI2C_stop_flag = 1U;
				}
			}
		} else {				/* Rx Mode */
			/* Do nothing */
		}
	} else {					/* master mode */
		/* Do nothing */
	}
}
```

7-7 LVD

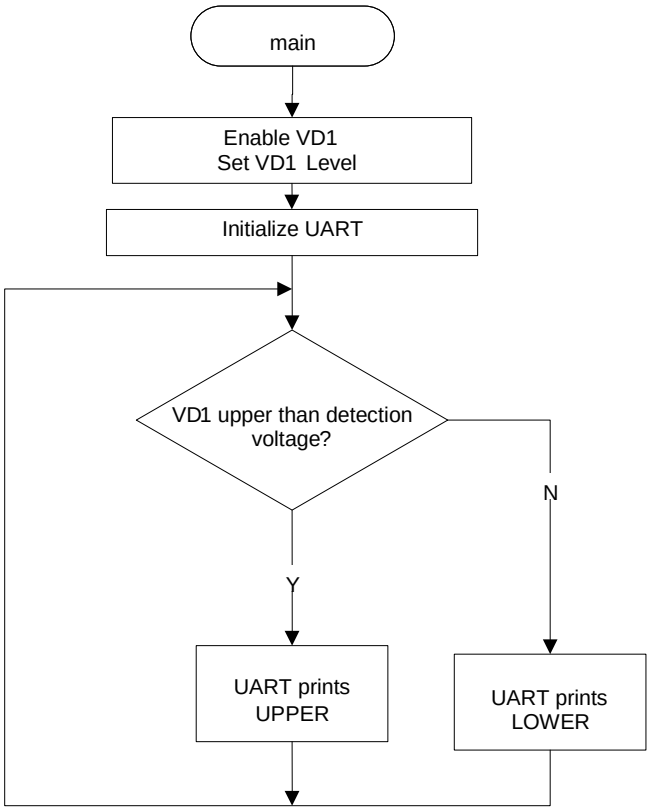
7-7-1 Example: LVD

This is a simple example based on the TX00 Peripheral Driver (LVD, GPIO).

The example includes:

1. LVD configuration
2. LVD status monitoring

- **Flowchart**



- **Code and Explanation for the Example**

Enable voltage detection 1; setting mode and detection level.

```
LVD_EnableVD1();
LVD_SetVD1Level(LVD_VDLVL1_290);
```

Initialize the system first. Disabe WDT explicitly to prevent confusion.

```
hardware_init(UART_RETARGET)                    /* Initialize UART */;
```

Then a usual while(1) loop.

Check VD1 status, if VD1 is upper than the detection voltage, UART displays “UPPER”. When VD1 is lower than the detection voltage, UART displays “LOWER”.

```
while (1) {
    if (LVD_GetVD1Status() == LVD_VD_UPPER) {
        common_uart_disp(“UPPER\n”);
    } else {
        common_uart_disp(“LOWER\n”);
    }
}
```

7-8 TMR16A

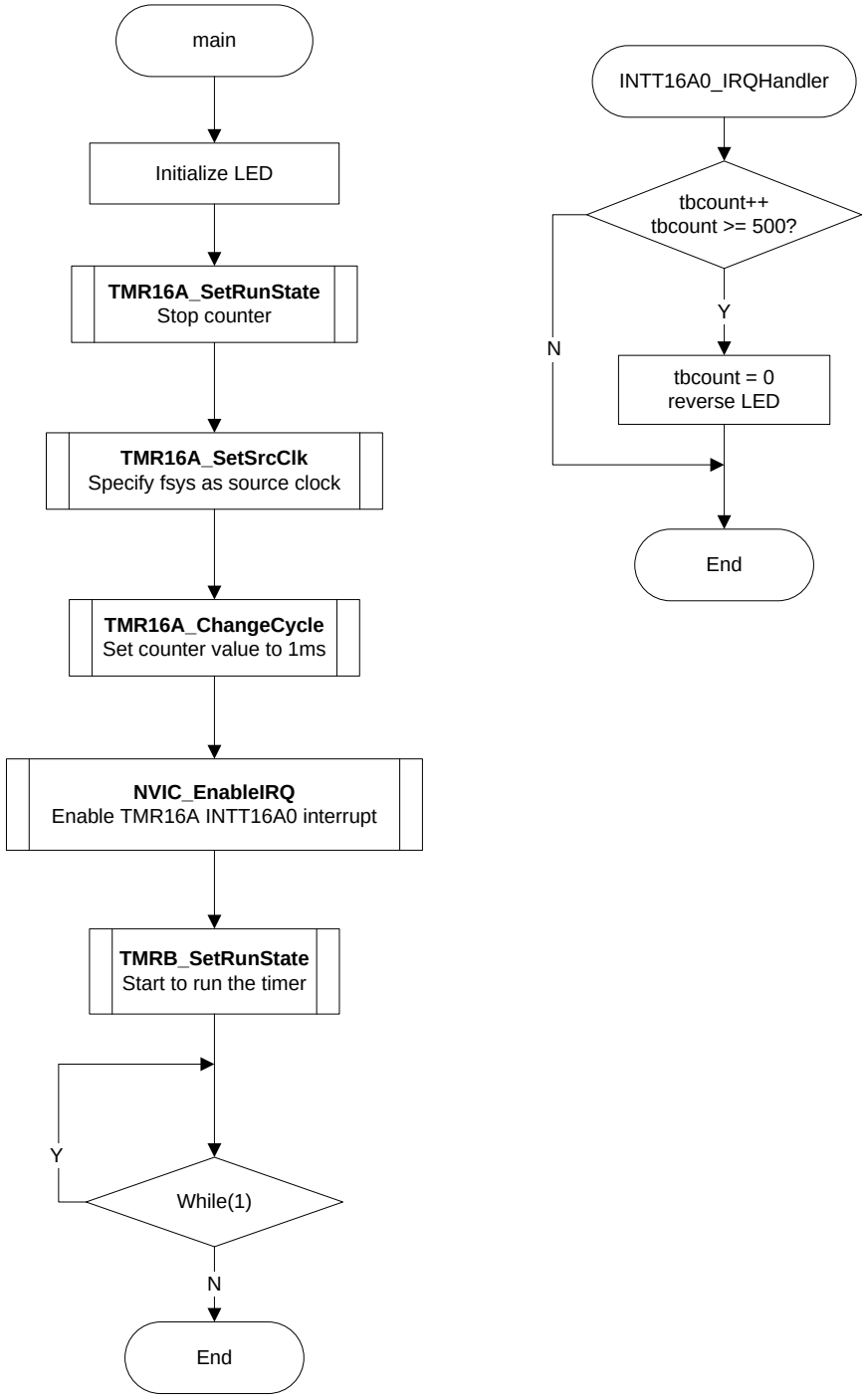
7-8-1 Example: General Timer

This is a simple example based on the TX00 Peripheral Driver (TMR16A, GPIO).

The example includes:

1. TMR16A setting
2. General Timer for 1ms

- **Flowchart**



- **Code and Explanation for the Example**

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

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

```
    LED_On(LED_ALL);                /* Turn on LED_ALL */
```

First stops the counter. This demo will set cycle to 1ms, macro TMR16A_1MS equals 0x4E20U, because ftmrb = fphiT0 = fsys = fc = 20MHz, Ttmrb = 1/20us, 1ms*20 us = 20000 = 0x4E80 (Please see CG part for more detail information about the clock setting)

Then run the counter.

```
    TMR16A_SetRunState(TSB_T16A0, TMR16A_STOP);   /* Counter stops*/
    TMR16A_SetSrcClk(TSB_T16A0, TMR16A_SYSCK);    /* Set source clock to system clock */
    TMR16A_ChangeCycle(TSB_T16A0, TMR16A_1MS);    /* Set counter value to 1ms*/
    NVIC_EnableIRQ(INTT16A0_IRQn);
    TMR16A_SetRunState(TSB_T16A0, TMR16A_RUN);
```

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) {
            LedOff(LED_ALL);
        } else {
            LedOn(LED_ALL);
        }
    } else {
        /* do nothing */
    }
```

7-9 TMRB

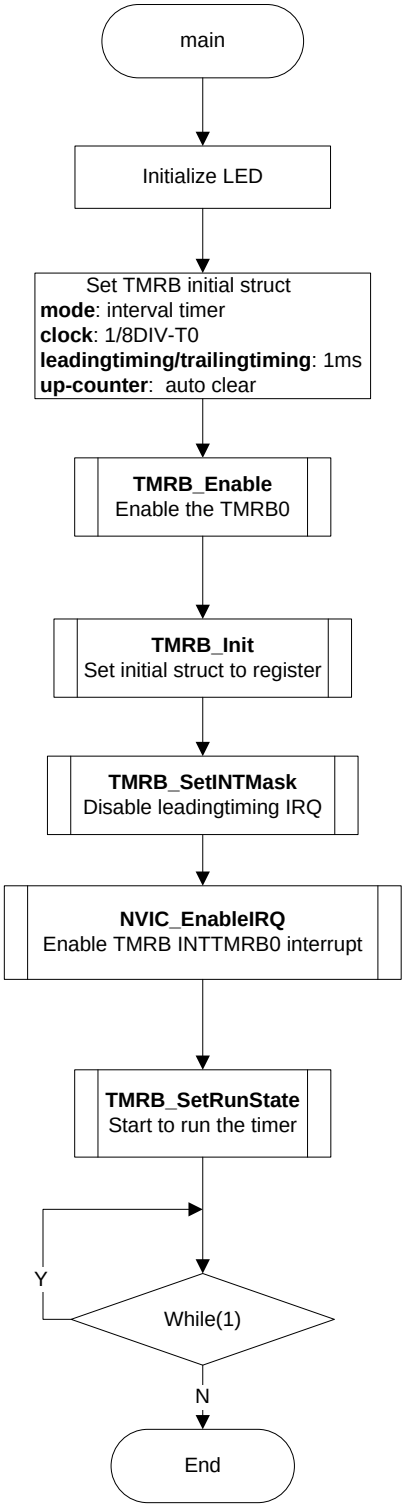
7-9-1 Example: General Timer

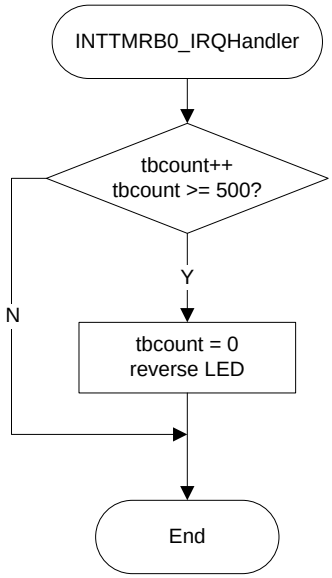
This is a simple example based on the TX00 Peripheral Driver (TMRB, GPIO).

The example includes:

1. TMRB0 initialization
2. General Timer for 1ms

- **Flowchart**





- **Code and Explanation for the Example**

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

```
LED_Init();                                    /* LED initialize */
LED_Off(LED_ALL);                              /* Turn off LED_ALL */
```

Prepare TMRB initialization structure; fill TMRB mode, clock, up-counter clear method, trailingtiming and leadingtiming. This demo will set trailingtiming and leadingtiming to 1ms. Macro TMRB_1MS equals 0x9C4, because fphiT0 = fsys = fc = fosc = 20MHz, ftmrb = 1/8 fphiT0 = 2.5MHz, Ttmrb = 0.4us, 1ms/0.4us = 2500 = 0x9C4 (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 INTTMRB0 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 */
TMRB_SetINTMask(TSB_TB0,
                TMRB_MASK_MATCH_LEADINGTIMING_INT)
NVIC_EnableIRQ(INTTMRB0_IRQn);           /* enable INTTMRB0 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 */
}
```

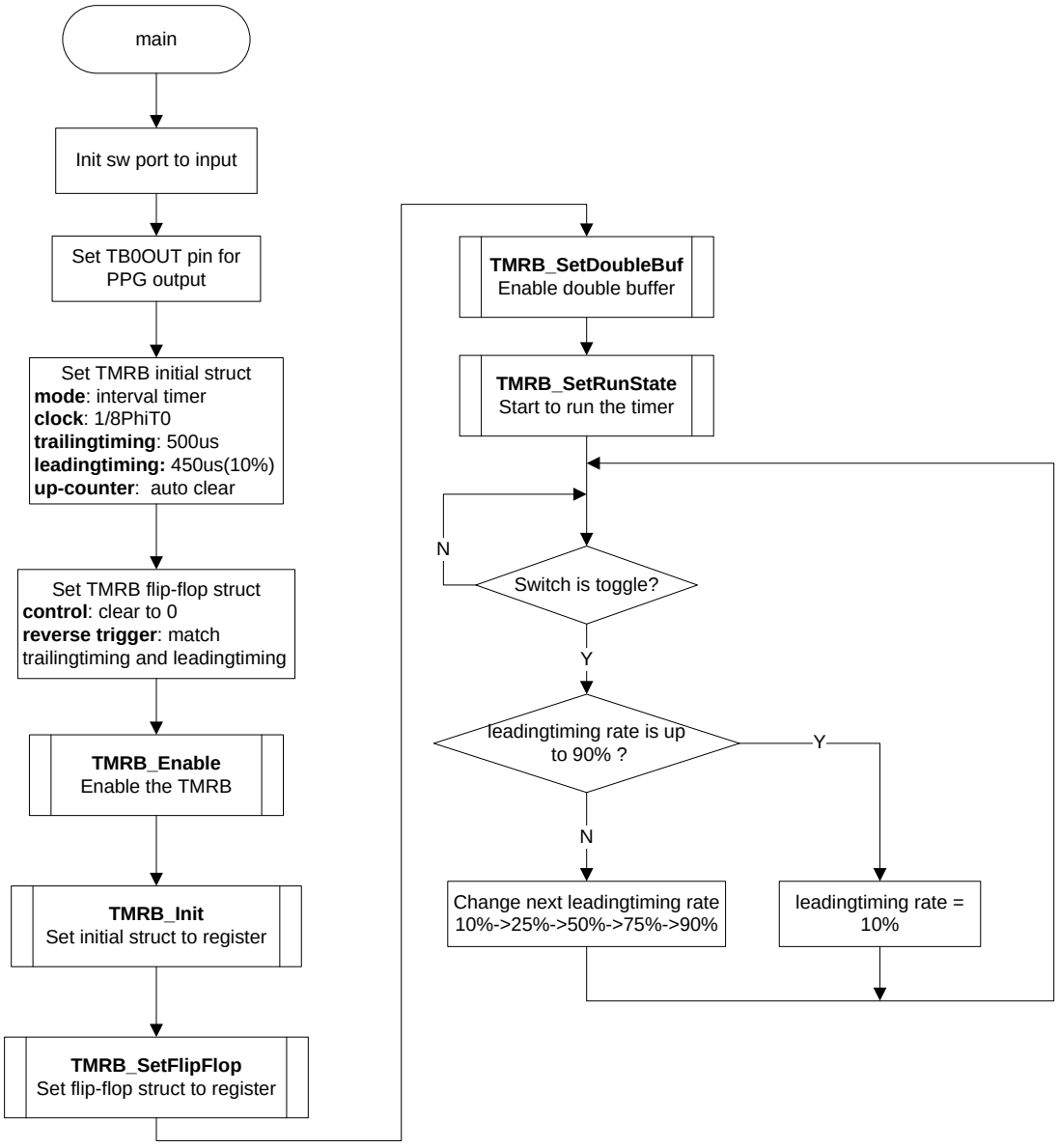
7-9-2 Example: PPG Output

This is a simple example based on the TX00 Peripheral Driver (TMRB, GPIO).

The example includes:

1. TMRB0 initialization
2. PPG process setting and start
3. PPG leadingtiming adjustment

- **Flowchart**



- **Code and Explanation for the Example**

At first, initialize sw port; set PC3 as TB0OUT for PPG output.

```
SW_Init();                          /* Init sw port to input */
/* Set PC3 as TB0OUT for PPG output */
GPIO_SetOutput (GPIO_PC, GPIO_BIT_3);
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_1, GPIO_BIT_3);
```

Prepare TMRB initialization structure, and then fill TMRB mode, clock, up-counter clear method, trailingtiming and leadingtiming into it. This demo will set trailingtiming to 500us, macro TMRB0TIME equals 0x4E2U, because fphiT0 = fsys = fc = fosc = 20MHz, ftmrb = 1/8 fphiT0 = 2.5MHz, Ttmrb = 0.4us, 500us/0.4us = 1250 = 0x4E2U (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.UpCntCtrl = TMRB_AUTO_CLEAR;         /* up-counter auto clear */
    m_tmrb.TrailingTiming = TMRB0TIME;          /* T = 500us */
    m_tmrb.LeadingTiming = LeadingTiming[Rate]; /* leadingtiming, initial
value 10% */
```

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

```
    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_TB0);
    TMRB_Init(TSB_TB0, &m_tmrb);
    TMRB_SetFlipFlop(TSB_TB0, &PPGFFInital);
    TMRB_SetDoubleBuf(TSB_TB0, ENABLE);         /* enable double buffer */
    TMRB_SetRunState(TSB_TB0, TMRB_RUN);
```

Wait switch change status, and at the same time.

```
    keyvalue = SW_Get(SW1);
    if (keyvalue == 1U) {
        delay(0xFFFFU);         /* noise cancel */
        keyvalue = SW_Get(SW1);
        if (keyvalue == 0U) {
            changestatus = 1U;
        } else {
                /* Do nothing */
        }
    } else {
            /* Do nothing */
    }
```

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

```
if (changestatus == 1U) {
        Rate++;                    /* change leadingtiming rate */
        if (Rate >= LEADINGTIMINGMAX) {
            Rate = LEADINGTIMINGINIT;
        } else {
            /* Do nothing */
        }
        TMRB_ChangeLeadingTiming(TSB_TB0, LeadingTiming[Rate]);
        changestatus = 0U;
    } else {
        /* Do nothing */
    }
```

The calculation method of leadingtiming value:
TrailingTiming = 500us, ftmrb = 1/8 fphiT0 = 2.5MHz, Ttmrb = 0.4us (these time parameters will be different if use different CG setting)

LeadingTiming = 10%: high-level time is 500*10% = 50us, low-level time is 500-50 = 450us, counter value = 450us/Ttmrb = 0x465U

LeadingTiming = 25%: high-level time is 500*25% = 125us, low-level time is 500-125 = 375us, counter value = 375us/Ttmrb = 0x3A9U

LeadingTiming = 50%: high-level time is 500*50% = 250us, low-level time is 500-250 = 250us, counter value = 250us/Ttmrb = 0x271U

LeadingTiming = 75%: high-level time is 500*75% = 375us, low-level time is 500-375 = 125us, counter value = 125us/Ttmrb = 0x138U

LeadingTiming = 90%: high-level time is 500*90% = 450us, low-level time is 500-450 = 50us, counter value = 50us/Ttmrb = 0x7DU

That is the calculation method of leadingtiming array

```
uint32_t LeadingTiming[5] = { 0x465U, 0x3A9U, 0x271U, 0x138U, 0x7DU };
/* leadingtiming: 10%, 25%, 50%, 75%, 90% */
```

7-10 SIO/UART

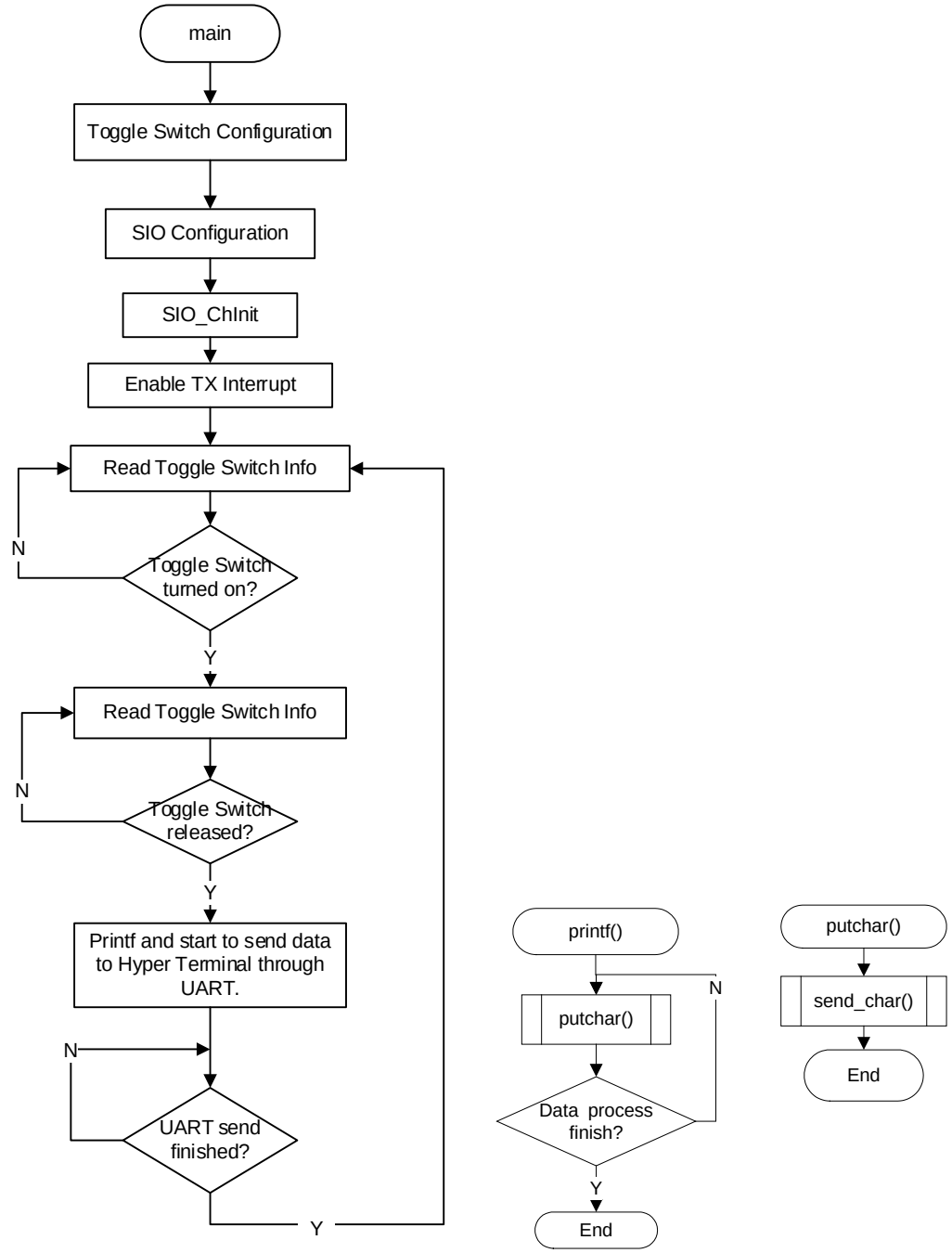
7-10-1 Example: Retarget

This is a simple example based on the TX00 Peripheral Driver (UART, GPIO).

The example includes:

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

- **Flowchart**



- **Code and Explanation for the Example**

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

```
GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_4, ENABLE);
GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_4, DISABLE);
GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_4);
```

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

```
    UART_InitTypeDef myUART;

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

After above setting, enable UART TX interrupt.

```
    NVIC_EnableIRQ(RETARGET_INT)
```

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

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

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

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

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

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

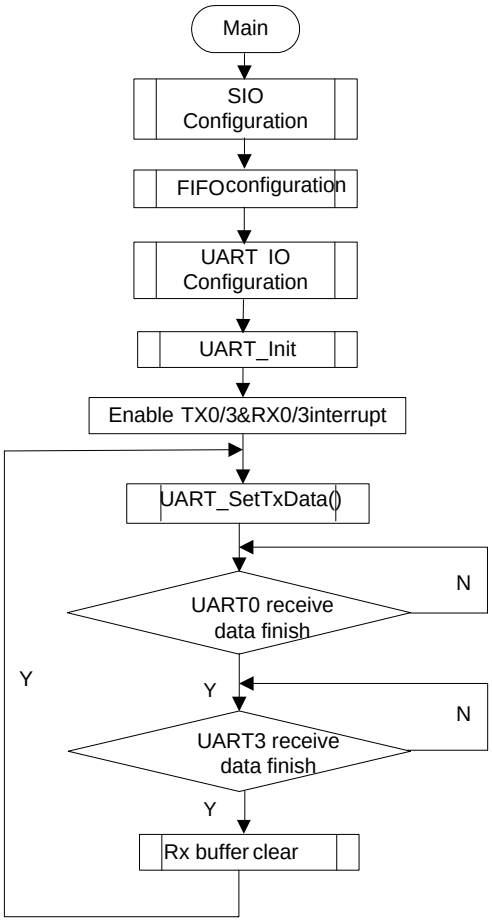
```
}

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

      return ch;
}
```

7-10-2 Example: UART FIFO

This is a simple example based on the TX00 Peripheral Driver (UART, GPIO).

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

- **Flowchart**



- **Code and Explanation for the Example**

At first configure the GPIO and initialize UART.

Use GPIO peripheral drivers configure GPIO for UART0 and UART3.

```
void SIO_Configuration(TSB_SC_TypeDef * SCx)
{
    if (SCx == TSB_SC0) {
        TSB_PD->CR |= GPIO_BIT_3;
        TSB_PD->FR1 |= GPIO_BIT_3;
        TSB_PD->FR1 |= GPIO_BIT_2;
        TSB_PD->IE |= GPIO_BIT_2;
    } else if (SCx == TSB_SC1) {
        TSB_PG->CR |= GPIO_BIT_2;
        TSB_PG->FR1 |= GPIO_BIT_2;
        TSB_PG->FR1 |= GPIO_BIT_1;
        TSB_PG->IE |= GPIO_BIT_1;
    } else if (SCx == TSB_SC2) {
        TSB_PE->CR |= GPIO_BIT_4;
        TSB_PE->FR1 |= GPIO_BIT_4;
        TSB_PE->FR1 |= GPIO_BIT_3;
        TSB_PE->IE |= GPIO_BIT_3;
    } else if (SCx == TSB_SC3) {
        TSB_PF->CR |= GPIO_BIT_3;
        TSB_PF->FR2 |= GPIO_BIT_3;
```

```
            TSB_PF->FR2 |= GPIO_BIT_2;
            TSB_PD->IE |= GPIO_BIT_2;
        }
}
```

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 UART0/3 channels.

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

     UART_Enable(UART3);
     UART_Init(UART3, &myUART);
```

FIFO configuration.

```
    UART_RxFIFOByteSel(UART0,UART_RXFIFO_RXFLEVEL);
    UART_RxFIFOByteSel(UART3,UART_RXFIFO_RXFLEVEL);

    UART_TxFIFOINTCtrl(UART0,ENABLE);
    UART_TxFIFOINTCtrl(UART3,ENABLE);

    UART_RxFIFOINTCtrl(UART0,ENABLE);
    UART_RxFIFOINTCtrl(UART3,ENABLE);

    UART_TRxAutoDisable(UART0,UART_RXTXCNT_AUTODISABLE);
    UART_TRxAutoDisable(UART3,UART_RXTXCNT_AUTODISABLE);

    UART_FIFOConfig(UART0,ENABLE);
    UART_FIFOConfig(UART3,ENABLE);

    UART_RxFIFOFillLevel(UART0, UART_RXFIFO4B_FLEVLE_4_2B);
    UART_RxFIFOFillLevel(UART3, UART_RXFIFO4B_FLEVLE_4_2B);

    UART_RxFIFOINTSel(UART0,UART_RFIS_REACH_EXCEED_FLEVEL);
    UART_RxFIFOINTSel(UART3,UART_RFIS_REACH_EXCEED_FLEVEL);

    UART_RxFIFOClear(UART0);
    UART_RxFIFOClear(UART3);

    UART_TxBufferClear(UART0);
    UART_TxBufferClear(UART3);
```

```
    UART_TxFIFOFillLevel(UART0, UART_TXFIFO4B_FLEVLE_0_0B);
    UART_TxFIFOFillLevel(UART3, UART_TXFIFO4B_FLEVLE_0_0B);

    UART_TxFIFOINTSel(UART0,UART_TFIS_REACH_NOREACH_FLEVEL);
    UART_TxFIFOINTSel(UART3,UART_TFIS_REACH_NOREACH_FLEVEL);

    UART_TxFIFOClear(UART0);
    UART_TxFIFOClear(UART3);
```

After above setting, enable UART0/3 interrupt.

```
    NVIC_EnableIRQ(INTTX0_IRQn);
    NVIC_EnableIRQ(INTRX3_IRQn);

    NVIC_EnableIRQ(INTTX3_IRQn);
    NVIC_EnableIRQ(INTRX0_IRQn);
```

The rest process of data flow is finished in ISR of UART0/3 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);
    }
}
```

UART3 TX interrupt routine:

```
void INTTX3_IRQHandler(void)
{
    volatile UART_Err err;

    if (TxCounter1 < NumToBeTx1) {
        UART_SetTxData(UART3, TxBuffer1[TxCounter1++]);
    } else {
        err = UART_GetErrState(UART3);
    }
}
```

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);
    }
}
```

UART3 RX interrupt routine:

```
void INTRX3_IRQHandler(void)
```

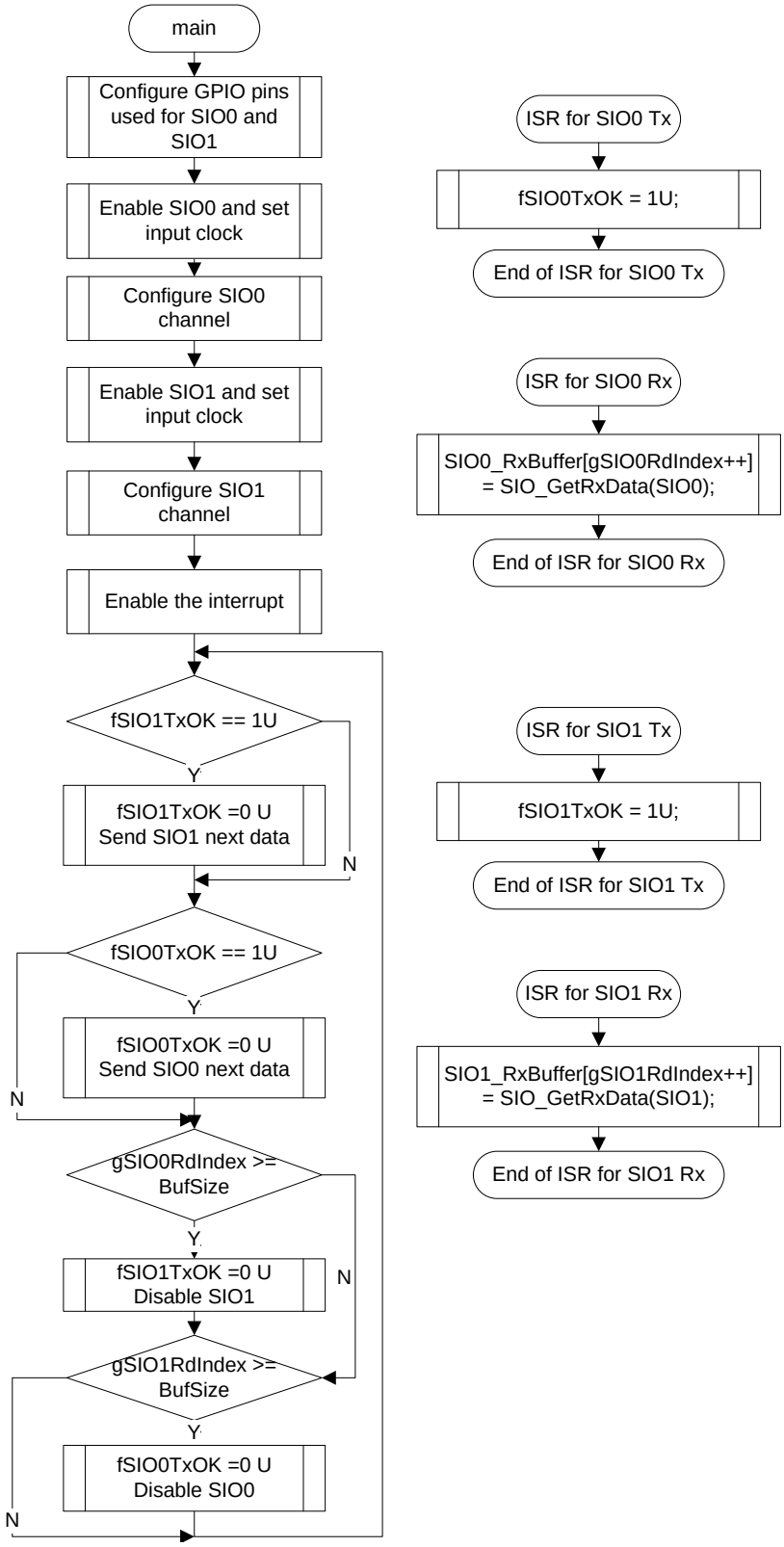
```
{
	volatile UART_Err err;

	err = UART_GetErrState(UART3);
	if (UART_NO_ERR == err) {
		RxBuffer1[RxCounter1++] = (uint8_t) UART_GetRxData(UART3);
	}
}
```

7-10-3 Example: SIO

This is a simple example based on the TX00 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.TIDLE = SIO_TIDLE_HIGH;
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_TS2;
SIO0_Init.Divider = SIO_BR_DIVIDER_2;

SIO_Init(SIO0, SIO_CLK_SCLKOUTPUT, &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.TIDLE = SIO_TIDLE_HIGH;
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;
SIO1_Init.TXDEMP = SIO_TXDEMP_HIGH;
SIO1_Init.EHOLDTime = SIO_EHOLD_FC_64;

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);
}
```

7-11 WDT

7-11-1 Example:WDT

This is a simple example based on the TX00 Peripheral Driver (WDT, GPIO).

The example includes:

1. WDT initialization.
2. In DEMO1 WDT isn't cleared before timer overflow, NMI interrupt is generated. LED1 will blink once
3. In DEMO2 WDT is cleared before timer overflow, the LED0 will blink all the time.

- **Code and Explanation for the Example**

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

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

Initialize WDT and enable it.

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

In DEMO1 Waiting for the NMI interrupt flag.

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

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

```
WDT_Disable();
```

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

```
WDT_WriteClearCode();
```