TOSHIBA

TX03 Peripheral Driver Usage Example (TMPM381/383)

Rev 0.102

Sep, 2017

TOSHIBA ELECTRONIC DEVICES & STORAGE CORPORATION

CMDR-M381UE-00xE

RESTRICTIONS ON PRODUCT USE

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

Index

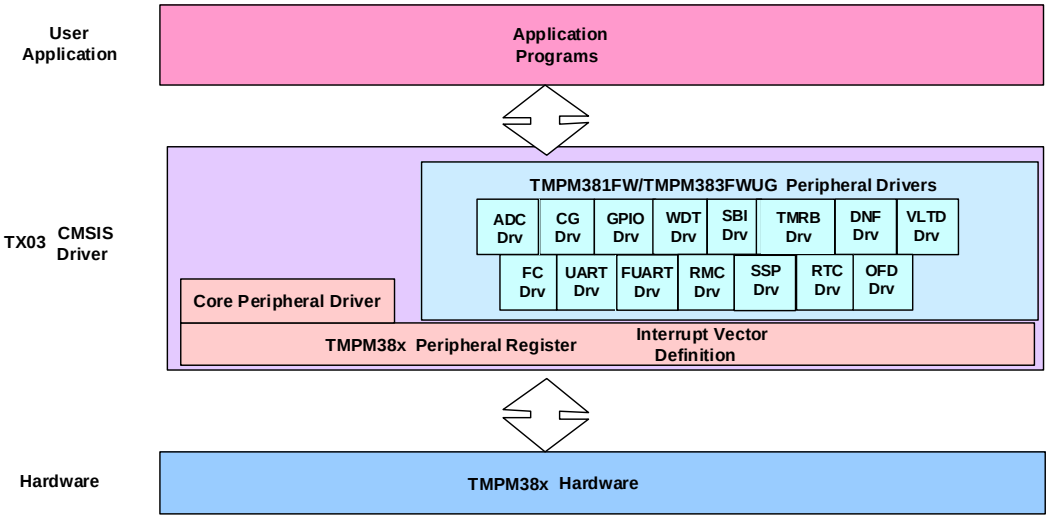
1 General description

The example programs described hereafter are specially designed for TOSHIBA TMPM38x(see Note below) 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.

Note: TMPM38x stands for TMPM381/383 hereafter.

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 (4x)
Standby mode	-	Used for CG demo (STOP mode)
Watch dog timer (WDT)	-	Used for WDT demo
External interrupt (INT)	INT0	Used for CG power mode change demo
	INT3	Used for DNF demo
	INTRTC	Used for RTC demo
	INTSBI0, INTSBI1	Used for SBI slave(master) demo / SIO demo
	INTRX0	Used for UART SIO / FIFO demo

Hardware	Channel	Use presence, use
	INTRX1	Used for UART SIO / FIFO demo
	INTTX0	Used for UART SIO / Retarget / FIFO demo
	INTTX1	Used for UART SIO / FIFO demo
SIO	SIO0	Used for UART retarget demo / SIO demo / UART FIFO demo
	SIO1	Used for SIO demo
16-bit timer	TMRB0	Used for TMRB: General Timer
	TMRB6	Used for TMRB: PPG Output
12-bit A/D converter	AIN0	Used for ADC data read demo
Real-time clock	-	Used for RTC demo
SBI	SBI0	Used for Slave / Master demo
SSP	SSP0	Used for SSP self loopback demo
Full UART	-	Used for Loopback demo
RMC	-	Used for RMC receiving demo
DNF	-	Used for DNF demo
OFD	-	Used for OFD demo
VLTD	-	Used for low voltage detection
Flash	-	Used for FC demo

4 Pin Usage

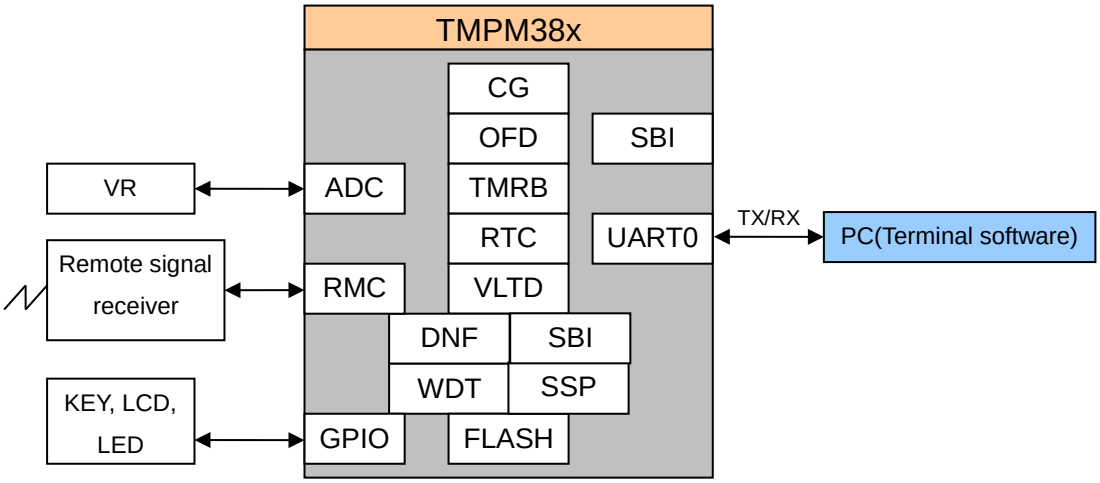
The example programs are tested on TMPM38x evaluation board.
Following is pin usage for example programs.

Pin No.	Name	Usage
1	PH0	ADC AIN0 / CG INT0
2	PF1	DNF TB7OUT
3	PA0	DNF INT3/ LED0
4	PE0	UART SC0TXD (M383)
5	PE1	UART SC0RXD (M383)
6	PC6	FUART UT0TXD50A
7	PC7	FUART UT0RXD50
8	PC5	FUART UT0TXD50B
9	PA5	TMRB TB6OUT UART TXD1 (M381) UART TXD1 (M383)
10	PA6	UART RXD1 (M381)

Pin No.	Name	Usage
		UART RXD1 (M383)
11	PE2	UART SCLK0
12	PA4	UART SCLK1
13	PA1	LED1
14	PA2	LED2
15	PA3	LED3
16	PH4	SW0/SW4
17	PH5	SW1/SW5
20	PH6	SW2/SW6
21	PH7	SW3/SW7
22	X1	High frequency resonator connection pin
23	X2	High frequency resonator connection pin
24	XT1	Low frequency resonator connection pin
25	XT2	Low frequency resonator connection pin
26	MODE	MODE pin
27	RESET	Reset signal input pin
28	BOOT	BOOT mode control pin

5 Development Environment

Following is development environment:



1. Hardware board:
   TMPM38x evaluation board (Not for sale)
   Starget kit for M381

2. Development tool:
   - IAR:

1) J-Link: IAR J-Link-ARM 7.0/ J-Link 6.0
2) IDE:IAR Embed Workbench 7.30.4 version

- KEIL:

1) IDE:KEIL uVision MDK 5.14

6 Functions

6-1 Operation mode

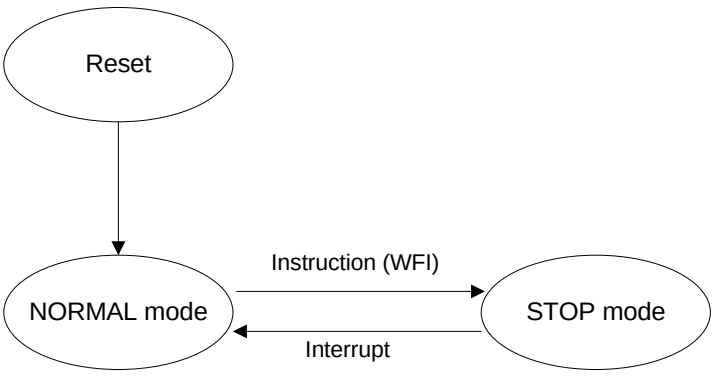
There are five operation modes for TMPM38x: NORMAL, SLOW, IDLE, SLEEP and STOP mode.
The NORMAL mode and the SLOW mode use the high-speed and low-speed clocks for the system clock respectively.
Note: Only NORMAL mode is demonstrated in the driver example.

- **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. The low-speed clock can also be oscillated.
IDLE, SLEEP, STOP mode are low power comsumption modes. To shift to the low power consumption mode, specify the mode in the system control register CGSTBYCR<STBY[2:0]> and execute the WFI (Wait For Interrupt) instruction.
Note: Only STOP mode is demonstrated in driver example.

- **STOP mode:**

Except some peripheral circuits, all the internal circuits including the internal oscillator are brought to a stop in STOP mode. When releasing STOP mode, the operation mode changes to the operation mode before entering STOP mode.
Note: The sample program of STOP mode and NORMAL mode is integrated into CG example.



6-2 ADC

Change the value of VR1(POT1), which is connected to *AIN0*. The voltage on it will be measured and used to change the loop blinking speed of the 4 LEDs (see **NOTE** below) in board.
The higher the voltage is, the faster the loop blinking speed.

NOTE:
LED0~LED3 connect with PA0~PA3 in TMPM38x MCU board.

6-3 CG

6-3-1 Power mode change

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

Current mode	**Action (Switch)**	**Operation**	**LED display**
NORMAL	Turn on SW4	NORMAL→ STOP	All LED turns off.
STOP	Turn off SW4 Turn on SW0(see **NOTE**).	STOP →NORMAL	All LED turns on.

Note:
LED0~3 connect with PA0~3 in TMPM38x MCU board.
SW0 connect with PH0 in TMPM38x MCU board to set external interrupt input (INT0).
SW4 connect with PH4 in TMPM38x MCU board to control the switch power mode

6-4 DNF

This is a simple example to show the procedure of DNF (Digital Noise Filter Circuit).
DNF can eliminate noise of input signals from external interrupt pins at the certain range.
TMRB can operate in 16-bit PPG output mode and output square waves with specified frequency and duty.
In this demo, DNF detect pulse that generated by PPG. If DNF detect noise pulse, interrupt request will be generated.

***Note:**
Connect PA0 with PF1 in TMPM38x MCU board.
PA0 is configured pin (INT3) of DNF.
PF1 is configured pin (TB7OUT) of TMRB.

6-5 FLASH

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

Demo sequence

(1) Power on
The demo board runs Reset Procedure.
Then program will start at block0.
If SW0 is turned off, LED1 blings quickly.

(2) If SW0 is turned on 1st, Programming routine is copied to RAM by Copy routine.
Erase block1.
Write data to block1.
Compare the data read from the block1 with the written data, if they are equal, LED1 blings slowly and LED2 always show.

(3) Same as step (1).

NOTE:
LED0~LED3 connect with PA0~PA3 in TMPM38x MCU board.
SW0 connect with PH4 in TMPM38x MCU board.

6-6 FUART

6-6-1 LoopBack

In this program, both Full UART Transmit and Receive FIFO are enabled.
This program sends 64 different data value from Full UART channel 0 TXD0 pin and receives data from RXD0 pin. When toggle switch SW0 is turned on, this program starts to read data from Receive FIFO. The program doesn’t stop reading data until Receive FIFO is empty. After Toggle switch SW0 is turned off and turned on several times, the program finished reading all the data that RXD0 receives. Then the program will compare all the

received data with transmitted data.
If received data are same with transmitted data, UART0 prints “RX TX SAME”.
If received data are different with transmitted data, UART0 prints “RX TX DIFF”.
This program can be run for two times to see the hardware flow control function.
NOTE:
Connect UT0TXD50A (PC6) pin with UT0RXD50 (PC7) pin on TMPM38x MCU board.

6-7 GPIO

This is a simple application based on the Peripheral Driver (GPIO), use GPIO API functions to configure LEDs(see **Note** below) and switchs(see **Note** below), turn on the LEDs(see **Note** below), or turn off the LEDs(see **Note** below).

***Note**:
LED0~LED3 connect with PA0~PA3 in TMPM38x MCU board.
SW0~SW3 connect with PH4~PH7 in TMPM38x MCU board.

6-8 OFD

This is a simple application based on the OFD driver, use OFD API functions to generate a reset for micro if the external oscillation of high frequency for CPU clock exceeds the detection frequency range.

6-9 RMC

6-9-1 RMC receiving

This module intends to catch the remote signal data and decode it. Decoded data will be displayed on terminal on IAR. Custom code or address code and command code will be displayed in Hex format.

Display format: TMPM38x RMC Demo
RMC_6: XX XX
RMC_6: YY YY

Format	Terminal soft display
1	RMC_1：
2	RMC_2：
3	RMC_3：
4	RMC_4：
5	RMC_5：

6	RMC_6：

6-10 RTC

Use the internal RTC to display the time and date on LCD.
Display the time and date on LCD and update the date and time.
Initial setting: **2015/06/12 13:54:00**, 24hours format.
Update interval: 1 second.
LCD prints:

```
2015/06/12
13:54:00
```

6-11 SBI

6-11-1 SBI Slave

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

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

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

```
K1: SBI SLAVE RECV
```

When SW4 (see **NOTE**) is pressed, the string “TOSHIBA” that got from the SBI (slave) received buffer will be printed.

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

NOTE:
SW4 connect with PH4 in TMPM38x MCU board.

6-11-2 SBI Master

This function intends to support the I2C bus master mode.
Connect two I2C buses (SBI & I2C) on evaluation board.
Work as I2C master.
Use SBI interrupt to handle I2C bus write.
Address of I2C: Any.
I2C Master (SBI) sends “TOSHIBA” to Slave (SBI).
SBI (master) demo start,
When SW7 (see **NOTE**) is pressed, the string “TOSHIBA” will be sent from SBI (master) to SBI (slave).

NOTE:
SW7 connect with PH7 in TMPM38x MCU board.

6-12 SSP

Data is sent and checked if the received data is the same as the send one the led 2 (see **Note** below) and led 3 will light on, if not same the led 0 and led 1 will light on, and the receive data will be printed on UART as below

UART prints

SSP RX DATA: xxxxxx

***Note:** LED0~LED3 connect with PA0~PA3 in TMPM38x MCU board.

6-13 TMRB

6-13-1 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).
.

6-13-2 PPG Output

Use Toggle Switch 1 to change PPG waveform. Connect PA5/TB6OUT to oscilloscope.

LeadingTiming can be changed as following:
10%, 25%, 50%, 75%, 90%
Power on to enter PPG mode and starts the PPG output.
Change the leading timing every switch changing:
10% ->25%--> 50%-->75% --> 90%-->10%

6-14 SIO/UART

6-14-1 SIO

This demo will show synchronously transfer and receive usage of SIO module in TMPM38x (see Note below). It use the channel SIO0, SIO1 and transfer data synchronously between them.(Connect TXD0 with RXD1, connect TXD1 with RXD0, connect sclk0 with sclk1.)

6-14-2 Retarget

This function 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 data from serial port.

6-14-3 UART FIFO

In this sample program, UART0 sent the data "TMPM38x1" to UART1 use FIFO, and at same time UART0 receive the data "TMPM38x2" 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 = "TMPM38x2",and RxBuffer1 = "TMPM38x1".

6-15 VLTD

6-15-1 VLTD Reset

This application implements power supply voltage status detecting by VLTD.
When the power supply voltage is lower than the detection voltage, the MCU reset automatically be hardware and PH0 is set to high level by software.

6-16 WDT

This is a simple example to show the procedure of Watchdog timer (WDT). The WDT is for detecting malfunctions (runaways) of the CPU caused by noises or other disturbances and remedying them to return the CPU to normal operation.
If the WDT detects a runaway, it generates an INTWDT interrupt or reset. The WDT cannot be used in the STOP, SLEEP and SLOW mode while high-speed frequency clock is stopped.
WDT is enabled after reset.
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.

7 Software

This software project creates the sample applications based on TMPM38x evaluation board witch will demonstrate the main feature of TMPM38x MCU.

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

For example, the workspace structure and project name of module WDT are defined as following:

IAR EWARM:

```
├─WDT_NMI
        ├─APP
        │   main.c
        │   tmpm38x_wdt_int.c
        │   tmpm38x_wdt_int.h
        │
        ├─TX03_CMSIS
        │   │   system_TMPM38x.c
        │   │   system_TMPM38x.h
        │   │   TMPM38x.h
```

```
│   └─startup
│         startup_TMPM38x.s
│
├─TX03_Periph_Driver
│   ├─inc
│   │   tmpm38x_gpio.h
│   │   tmpm38x_wdt.h
│   │   tmpm38x_cg.h
│   │   tx03_common.h
│   │
│   └─src
│         tmpm381_gpio.c
│         tmpm381_wdt.c
│         tmpm381_cg.c
│
└─TMPM38x-EVAL
          led.c
          led.h
```

KEIL MDK:

```
├─WDT_NMI
    ├─APP
    │   main.c
    │   tmpm38x_wdt_int.c
    │   tmpm38x_wdt_int.h
    │
    ├─TX03_CMSIS
    │    system_TMPM38x.c
    │    startup_TMPM381.s
    │
    ├─TX03_Periph_Driver
    │    tmpm381_gpio.c
    │    tmpm381_wdt.c
    │    tmpm381_cg.c
    │
    └─TMPM381-EVAL
         led.c
```

NOTE: tmpm38x can be tmpm381/tmpm383.

Project files for the starter kit are stored in the ***-SK folder.

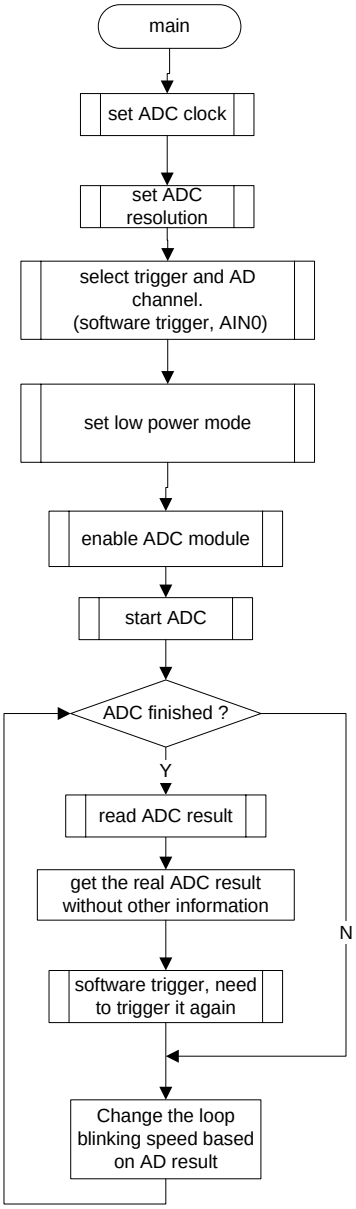
7-1 ADC

7-1-1 Example: ADC Data Read

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

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

- **Flowchart:**



- **Code and Explanation for the Example**

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

```
    /* 1. set ADC clock */
    ADC_SetClk(TSB_AD, ADC_HOLD_FIX, ADC_FC_DIVIDE_LEVEL_NONE);

    /* 2. Set ADC resolution */
    ADC_SetResolution(TSB_AD, ADC_12BITS );

    /* 3. select trigger and AD channel, this time we use software trigger, */
    /*    the VR1 is connected to AIN0, remember to input with macro TRG_ENABLE() */
    ADC_SetSWTrg(TSB_AD, ADC_REG0, TRG_ENABLE(ADC_AIN0));

    /* 4. to let ADC module work, we must disable low power mode */
    ADC_SetLowPowerMode(TSB_AD, DISABLE);

    /* 5. enable ADC module */
    ADC_Enable(TSB_AD);
```

Then start AD conversion:

```
    ADC_Start(TSB_AD, ADC_TRG_SW);
```

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

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

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

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

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

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

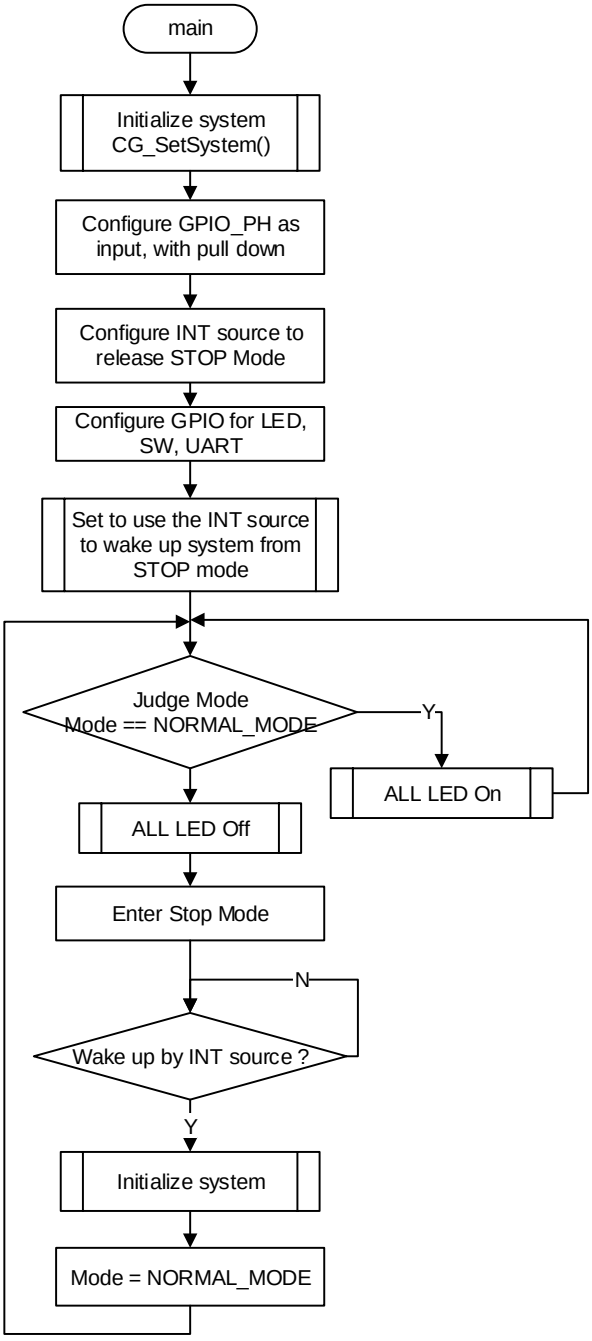
7-2 CG

7-2-1 Example: Power mode change

This is a simple example based on the TX03 Peripheral Driver (CG and GPIO).
The example includes:

1. Basic setup operation of CG
2. How to switch between normal mode and slop mode
3. How to enable multiple clock circuit

- **Flowchart**



- **Code and Explanation for the Example**

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

```
void CG_SetSystem(void)
{
    /* Set fgear = fc/2 */
    CG_SetFgearLevel(CG_DIVIDE_2);
    /* Set fperiph to fgear */
    CG_SetPhiT0Src(CG_PHIT0_SRC_FGEAR);
    CG_SetPhiT0Level(CG_DIVIDE_4);
    /* select external oscillator */
```

```
    CG_SetFoscSrc(CG_FOSC_EHOSC);
    CG_SetPortM(CG_PORTM_AS_HOSC);
    /* Enable high-speed oscillator */
    CG_SetFosc(CG_FOSC_EHOSC, 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();
}
```

Configure SW, LED and INT source pins

Configure the I/O pins for INT source, LED, SW and UART.

```
        /* Initialize system */
        CG_SetSystem();

        /* Configure GPIO for wakeup interrupt INT0 */
        GPIO_SetInput(GPIO_PH, GPIO_BIT_0);      /* Set PH0 to input */
        GPIO_EnableFuncReg(GPIO_PH, GPIO_FUNC_REG_1, GPIO_BIT_0);
/* Set PH0 for INT0 */
        GPIO_SetPullDown(GPIO_PH, GPIO_BIT_0, ENABLE);

        /* Configure GPIO for LED */
        LED_Init();

        /* Configure GPIO for SW */
        SW_Init();

        /* Configure GPIO for UART */
        hardware_init(UART_RETARGET);
```

Configure external interrupt to wake up system

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

```
    /* Disable interrupts */
    __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_RISING, 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.

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

```
}
```

Enable multiple clock circuit

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

```
Result CG_EnableClkMulCircuit(void)
{
	Result retval = ERROR;
	WorkState st = BUSY;
	retval = CG_SetPLL(ENABLE);
	if (retval == SUCCESS) {
		/* Set warm up time */
		CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC_EXT_HIGH,
CG_WUODR_PLL);
		CG_StartWarmUp();

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

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

	return retval;
}
```

7-3 DNF

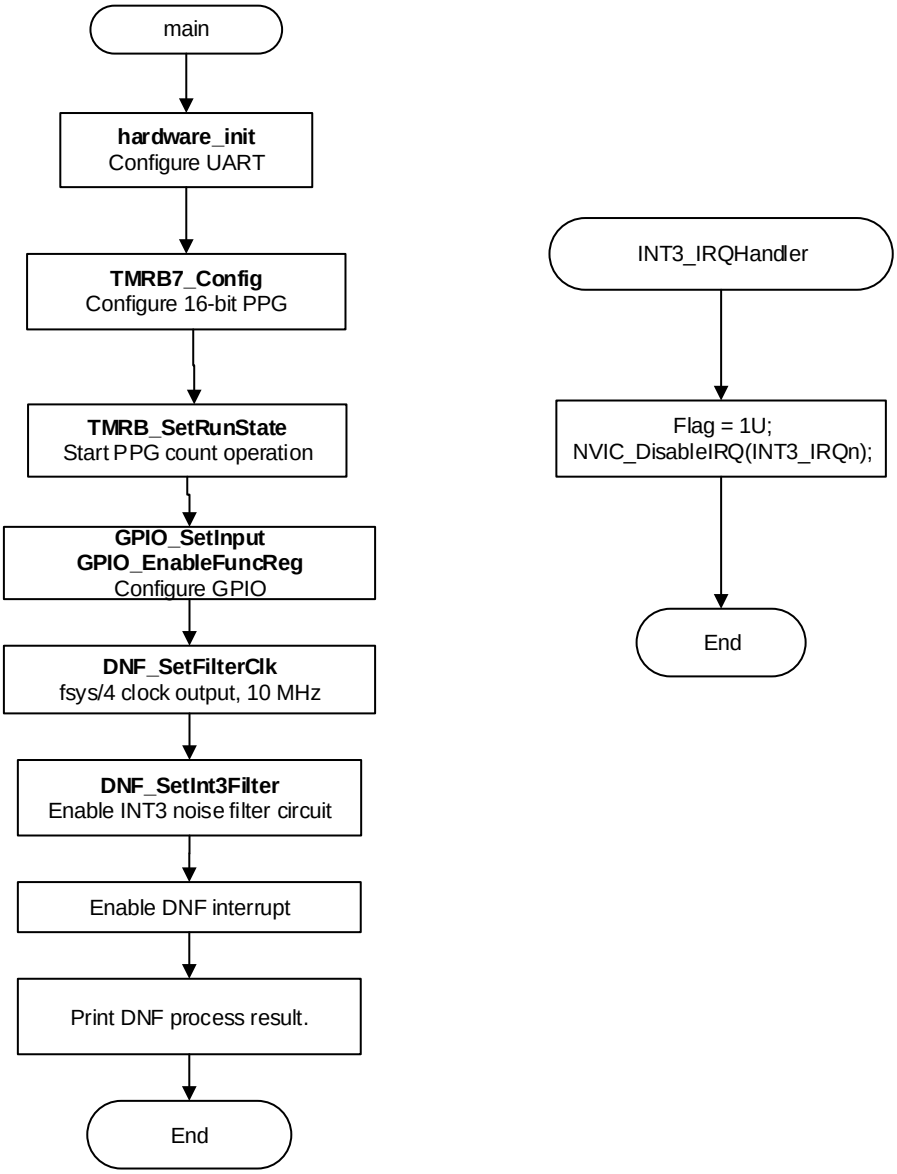
7-3-1 Example: DNF

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

The example includes:

1. Configure DNF and TMRB.
2. DNF detect pulse from external interrupt pin (INT3).
3. UART print DNF process result.

- **Flowchart**



- **Code and Explanation for the Example**

Configure UART to print character string. Configure TMRB to PPG output mode to generate square waves with specified frequency and duty. And then, start PPG count operation.

```
/* Configure UART */
hardware_init(UART_RETARGET);
/* Configure 16-bit PPG */
TMRB7_Config();
/* Start PPG count operation */
TMRB_SetRunState(TSB_TB7, TMRB_RUN);
/* Configure GPIO to INT3 (PA0) */
GPIO_SetInput(GPIO_PA, GPIO_BIT_0);
GPIO_EnableFuncReg(GPIO_PA, GPIO_FUNC_REG_2, GPIO_BIT_0);
```

Configure DNF noise filter circuit determination input level to 0.6us. And then, enable INT3 noise filter circuit

```
/* fsys/4 clock output, 10 MHz */
DNF_SetFilterClk(DNF_FILTER_CLK_FSYS_4);
/* Enable INT3 noise filter circuit */
DNF_SetInt3Filter(ENABLE);
```

Set up INT3 noise filter interrupt

```
__disable_irq();
NVIC_ClearPendingIRQ(INT3_IRQn);
NVIC_EnableIRQ(INT3_IRQn);
__enable_irq();
```

Print DNF process result.

```
      while(1U) {
#if defined CORRECT_PULSE
            if (Flag == 1U) {
                  common_uart_disp("DNF processed with error !\r\n");
            } else {
                  common_uart_disp("Correct pulse is detected !\r\n");
            }
#elif defined NOISE_PULSE
            if (Flag == 1U) {
                  common_uart_disp("Noise pulse is detected !\r\n");
            } else {
                  common_uart_disp("DNF processed with error !\r\n");
            }
#endif
    }
```

7-4 FLASH

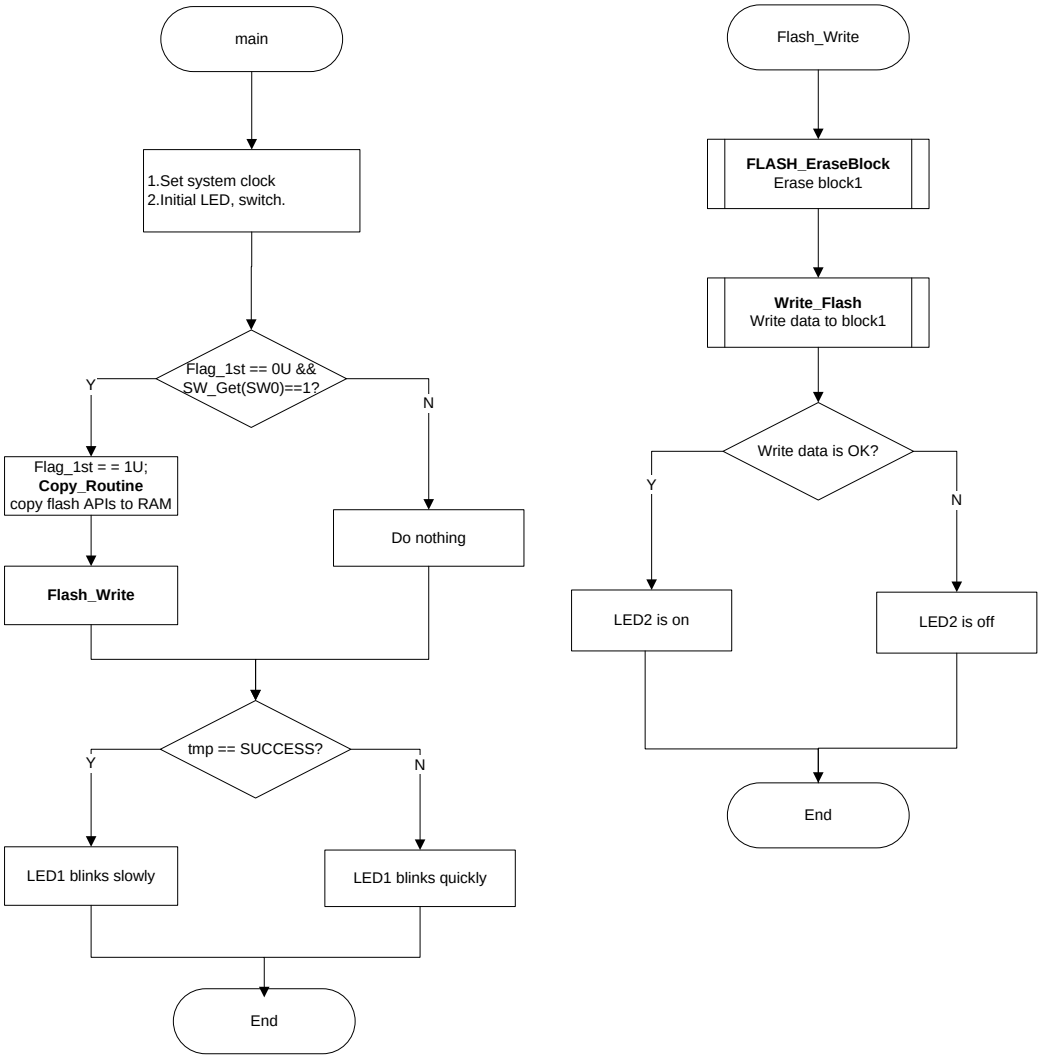
7-4-1 Example: Flash Write

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

The example includes:

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

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

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

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

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

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

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

```
    int i = 0U;
    uint32_t pagedata[FC_PAGE_SIZE];
    Result tmp = ERROR;
    uint32_t buffer[FC_PAGE_SIZE] = {0U};
    if (FC_SUCCESS == FC_EraseBlock(FC_WRITE_ADDR)) {                    /*
erase this block which will be written */
        for (i = 0U; i < FC_PAGE_SIZE; i++) {
            pagedata[i] = 0x12345678U;
        }
        if (FC_WritePage(FC_WRITE_ADDR, pagedata) == FC_SUCCESS)
/* write data to block which was erased */
        {
            Copy_Routine(buffer, (uint32_t*)FC_WRITE_ADDR,
FLASH_PAGE_SIZE);
```

```
                tmp = SUCCESS;
                LED_On(LED2);
                for (i = 0U; i < FC_PAGE_SIZE; i++) {
                    if(buffer[i] != pagedata[i]) {
                        tmp = ERROR;
                        LED_Off(LED2);
                        break;
                    }else {
                       /* Do nothing */
                    }
                }
            }else {
                /* Do nothing */
            }
        } else {
           /* Do nothing */
        }
        return tmp;
```

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

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

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

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

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

```
            while (BUSY == FC_GetBusyState()) {        /* check if FLASH is busy with
overtime counter */
                if (!(counter--)) { /* check overtime */
                    retval = FC_ERROR_OVER_TIME;
```

```
            break;
        } else {
            /* Do nothing */
        }
    }
```

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

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

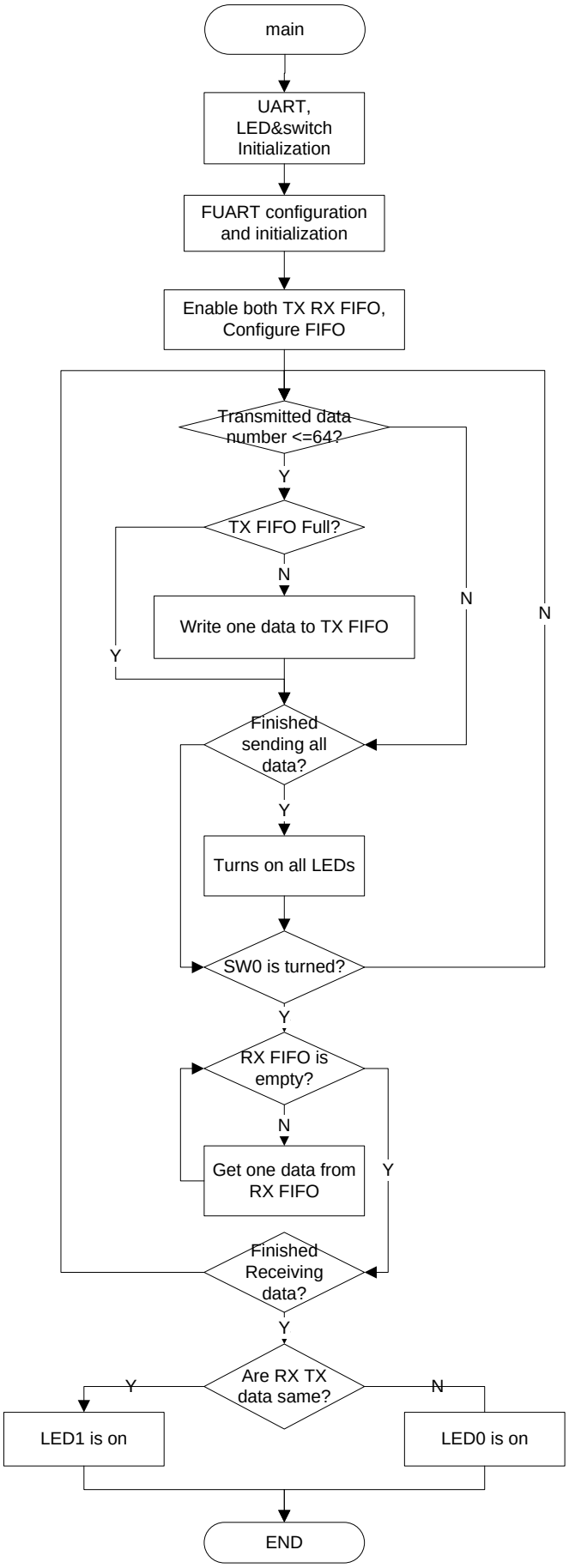
7-5 FUART

7-5-1 Example: LoopBack

This is a simple example based on the TX03 Peripheral Driver (FUART).
This program can be run for two times to see the hardware flow control function.
The example includes:

1. LED, switch Initialization, Full UART configuration and initialization.
2. Full UART Transmit data process.
3. Turn SW0 to receive data.
4. When receving data is finished, compare the received data with transmitted data.

- **Flowchart**


main
UART, LED&switch Initialization
FUART configuration and initialization
Enable both TX RX FIFO, Configure FIFO
Transmitted data number <=64?
Y
TX FIFO Full?
N
Write one data to TX FIFO
Y
N
N
Finished sending all data?
Y
Turns on all LEDs
SW0 is turned?
Y
RX FIFO is empty?
N
Get one data from RX FIFO
Y
Finished Receiving data?
Y
Are RX TX data same?
Y
N
LED1 is on
LED0 is on
END

- **Code and Explanation for the Example**

At first, the program Initializes LED and switch.
Configure GPIO for LED and switch.

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

Configure GPIO for FUART0.

```
/* Configure port PC5 to be UT0TXD50B */
GPIO_SetOutput(GPIO_PC, GPIO_BIT_5);
GPIO_SetPullUp(GPIO_PC, GPIO_BIT_5, ENABLE);
GPIO_SetPullDown(GPIO_PC, GPIO_BIT_5, DISABLE);
GPIO_SetOpenDrain(GPIO_PC, GPIO_BIT_5, DISABLE);
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_5, GPIO_BIT_5);

/* Configure port PC6 to be UT0TXD50A */
GPIO_SetOutput(GPIO_PC, GPIO_BIT_6);
GPIO_SetPullUp(GPIO_PC, GPIO_BIT_6, ENABLE);
GPIO_SetPullDown(GPIO_PC, GPIO_BIT_6, DISABLE);
GPIO_SetOpenDrain(GPIO_PC, GPIO_BIT_6, DISABLE);
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_5, GPIO_BIT_6);

/* Configure port PC7 to be UT0RXD50 */
GPIO_SetInput(GPIO_PC, GPIO_BIT_7);
GPIO_SetPullUp(GPIO_PC, GPIO_BIT_7, ENABLE);
GPIO_SetPullDown(GPIO_PC, GPIO_BIT_7, DISABLE);
GPIO_SetOpenDrain(GPIO_PC, GPIO_BIT_7, DISABLE);
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_5, GPIO_BIT_7);
```

Create a FUART_InitTypeDef structure and fill all the data fields, then Initialize FUART0.

```
FUART_InitTypeDef myFUART;

myFUART.BaudRate = 300U;
myFUART.DataBits = FUART_DATA_BITS_8;
myFUART.StopBits = FUART_STOP_BITS_1;
myFUART.Parity = FUART_1_PARITY;
myFUART.Mode = FUART_ENABLE_TX | FUART_ENABLE_RX;
myFUART.FlowCtrl = FUART_NONE_FLOW_CTRL;

FUART_Init(FUART0, &myFUART);
```

Enable 50% duty mode for FUART0.

```
FUART_SetPeriodDetection(FUART0, FUART_PERIOD_DETEC_7 );
FUART_SetTxTerminalMode(FUART0, FUART_1_TERMINAL);
FUART_SetStartBitTerminal(FUART0, FUART_TXD50A);
FUART_EnableDuty(FUART0);
```

Use FUART peripheral drivers enable FUART0 and configure FIFO.

```
FUART_Enable(FUART0);
FUART_EnableFIFO(FUART0);
FUART_SetINTFIFOLevel(FUART0,FUART_RX_FIFO_LEVEL_16,
FUART_TX_FIFO_LEVEL_4);
```

Then FUART0 starts to send data, the Full UART will only send 64 differfent data

value, and it will send the data when the transmit FIFO is normal or empty. When each data is sent, the LEDs display the data. If all the data is sent, All LEDs turn on.

```
    if (cntTx < MAX_BUFSIZE) {
        FIFOStatus = FUART_GetStorageStatus(FUART0, FUART_TX);
        if ((FIFOStatus == FUART_STORAGE_EMPTY) || (FIFOStatus ==
FUART_STORAGE_NORMAL)) {
            FUART_SetTxData(FUART0, Tx_Buf[cntTx]);
            LED_TXDataDisplay(Tx_Buf[cntTx]);
            cntTx++;
            if (64U == cntTx) {
                LED_On(LED_ALL);      /* sending data is finished */
            }
        }
    }
```

Each time when Key SW0 is turned, the program starts to read data from Receive FIFO. If some data are got, the program doesn't stop reading data until the FIFO is empty. If no any data can be got when SW0 is turned, it starts to compare the received data with transmitted data. If received data are same with transmitted data, LED1 is on. If received data are different with transmitted data, LED0 is on.

```
    SW0_last = SW0_this;
    SW0_this = SW_Get(SW0);
    SW0_this = !SW0_last;
    if (SW0_last != SW0_this) {        /* turn the switch SW0 */
        while (FUART_STORAGE_EMPTY != FUART_GetStorageStatus(FUART0,
FUART_RX)) {
            receive = FUART_GetRxData(FUART0);
            Rx_Buf[cntRx] = receive;
            cntRx++;
        }

        rxlast = rxthis;
        rxthis = cntRx;

        if (rxlast != rxthis) {        /* there are some data that has been received   */
            LED_Off(LED_ALL);
            LED_On(LED2);
        } else {                    /* receiving data is finished */
            result = Buffercompare(Tx_Buf, Rx_Buf, MAX_BUFSIZE);
            if (result == SAME) {
                /* received data are same with transmitted daa */
                LED_Off(LED_ALL);
                LED_On(LED1);
                while (1) {
                }
            } else {
                /* received data are different with transmitted data */
                LED_Off(LED_ALL);
                LED_On(LED0);
                while (1) {
                }
            }
        }
    }
```

7-6 GPIO

This function configures GPIO to make LEDs and Switches work. Read Switches to control LEDs on and LEDs off.

7-6-1 Example: GPIO Data Read

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

1. GPIO initialization
2. Read data from GPIO
3. Write data to 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 Switch. For example,

```
    GPIO_SetOutput(GPIO_PA, GPIO_BIT_0 | GPIO_BIT_1 | GPIO_BIT_2 |
GPIO_BIT_3);

    GPIO_SetInput(GPIO_PH,   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).
For example, read SW0:

```
    if (GPIO_ReadDataBit(GPIO_PH, 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_PA);
    tmp |= led;
    GPIO_WriteData(GPIO_PA, tmp);
```

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

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

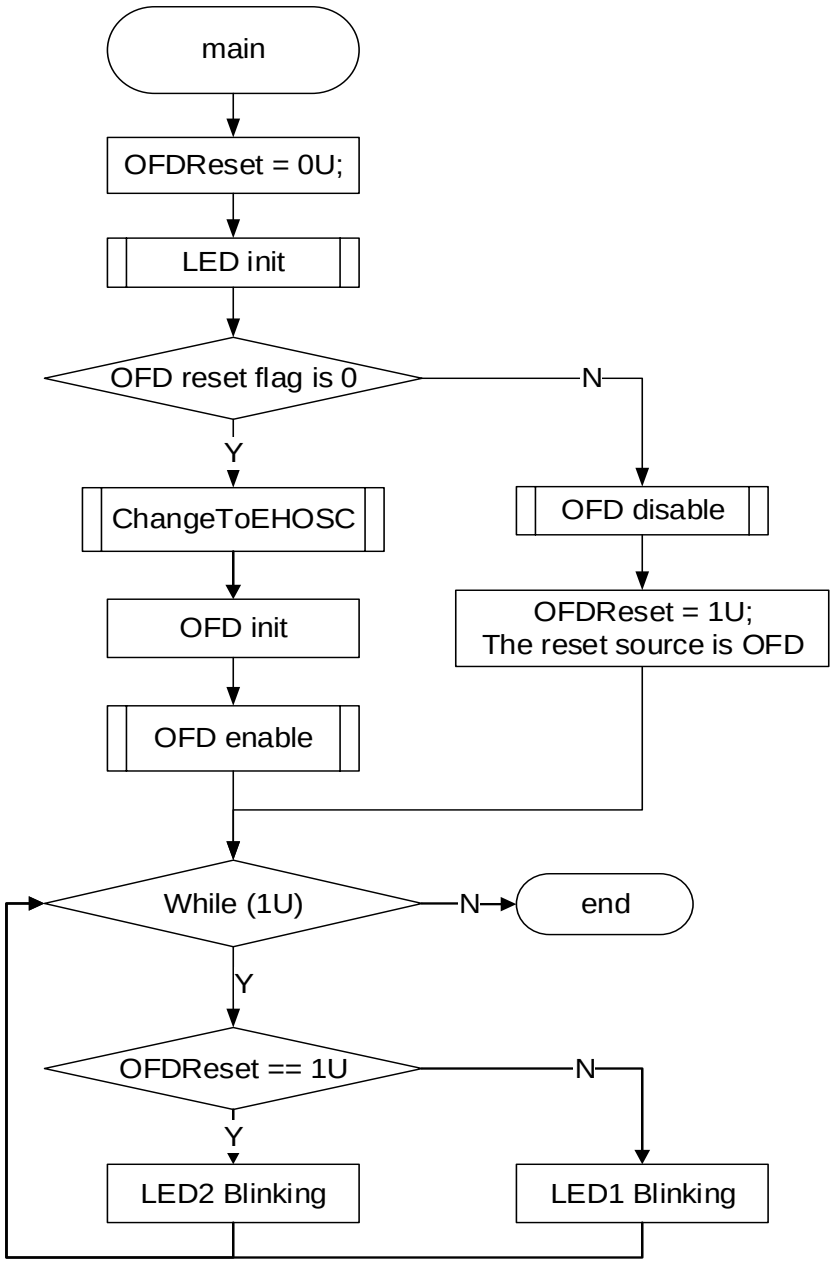
7-7 OFD

7-7-1 Example: 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 check (in CG).
3. OFD enable and disable.

- **Flowchart**



- **Code and Explanation for the Example**

At first, initialize LED.

```
LED_Init();
```

Check the OFD reset flag.

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

In normal reset, for example, when the MCU is reset by power up, the OFD flag will be '0', then the program change to use the external oscillation and initialize OFD to set OFD detect frequency range.

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

```
void ChangeToEHOSC(void)
{
    /* Use the external oscillation */
    TSB_CG->OSCCR &= 0x000FFFFFUL;
    TSB_CG->OSCCR |= 0xFFF00000UL;    /* Set the warm up time WUODR[11:0] */
    TSB_CG_OSCCR_HOSCON = 1U;     /* Set gpio port as X1/X2 for external oscillator */
    TSB_CG_OSCCR_XEN1 = 1U;        /* Enable external oscillator */
    TSB_CG_OSCCR_WUPSEL2 = 1U;
    TSB_CG_OSCCR_WUPSEL1 = 0U;    /* Select warm-up clock */
    TSB_CG_OSCCR_WUEON = 1U;       /* Start warm up */
    while (TSB_CG_OSCCR_WUEF) {
        /* Wait for warm-up finish */
    }
    TSB_CG_OSCCR_OSCSEL = 1U;     /* Use the external oscillation */
    TSB_CG_OSCCR_XEN2 = 1U;        /* Enable internal oscillator for ofd */
}
```

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

```
OFD_SetRegWriteMode(ENABLE);
```

Then set detection frequency.

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

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

LED1 and LED2 indicate the system clock status as below.

LED status	Meaning
LED1 blinking	MCU runs 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 uses the default setting to run under inner oscillator. OFD function is disabled.

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

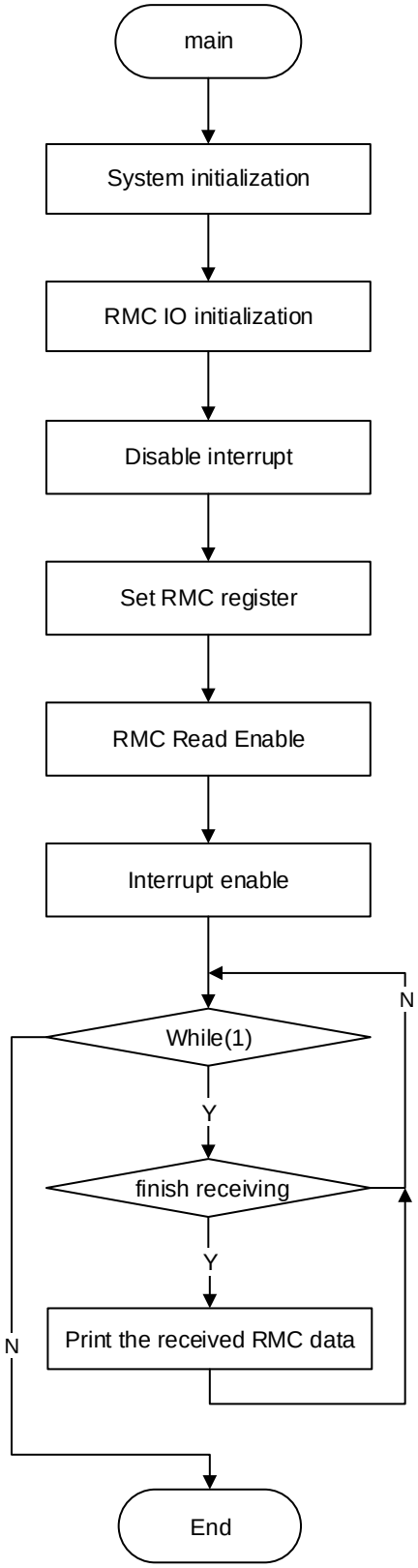
7-8 RMC

7-8-1 Example: RMC receiving

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

The example includes:
1. RMC IO initialization
2. Receiving RMC data

- **Flow Chart**


main
System initialization
RMC IO initialization
Disable interrupt
Set RMC register
RMC Read Enable
Interrupt enable
While(1)
N
Y
finish receiving
Y
Print the received RMC data
N
End

- **Code and Explanation for the Example**

At first, in main(), create a myRMC structure, then fill all the data fields.
For example

```
RMC_InitTypeDef myRMC;

myRMC.LeaderPara.MaxCycle = RMC_MAX_CYCLE;
myRMC.LeaderPara.MinCycle = RMC_MIN_CYCLE;
myRMC.LeaderPara.MaxLowWidth = RMC_MAX_LOW_WIDTH;
myRMC.LeaderPara.MinLowWidth = RMC_MIN_LOW_WIDTH;
myRMC.LeaderPara.LeaderDetectionState = ENABLE;
myRMC.LeaderPara.LeaderINTState = DISABLE;
myRMC.FallingEdgeINTState = DISABLE;
myRMC.SignalRxMethod = RMC_RX_IN_CYCLE_METHOD;
myRMC.LowWidth = RMC_TRG_LOW_WIDTH;
myRMC.MaxDataBitCycle = RMC_TRG_MAX_DATA_BIT_CYCLE;
myRMC.LargerThreshold = RMC_LARGER_THRESHOLD;
myRMC.SmallerThreshold = RMC_SMALLER_THRESHOLD;
myRMC.InputSignalReversedState = DISABLE;
myRMC.NoiseCancellationTime = RMC_NOISE_CANCELLATION_TIME;
```

Then, enable and initialize the RMC channel.

```
RMC_Enable(RMCx);
```

Initial the specified RMC channel with the structure which includes the basic RMC configuration.

```
RMC_Init(RMCx, &myRMC);
```

Enable the reception of the specified RMC0 channel.

```
RMC_SetRxCtrl(RMCx, ENABLE);
```

In interrupt INTRMCRX_IRQHandler():
Get the interrupt factor for the specified RMC0 channel.

```
RMC_INTFactor myRMC_INTFactor;
myRMC_INTFactor = RMC_GetINTFactor(TSB_RMC);
```

Get the leader detection result for the specified RMC0 channel.

```
RMC_LeaderDetection myRMC_LeaderDetection;
myRMC_LeaderDetection = RMC_GetLeader(TSB_RMC);
```

Get the received data from the specified RMC0 channel.

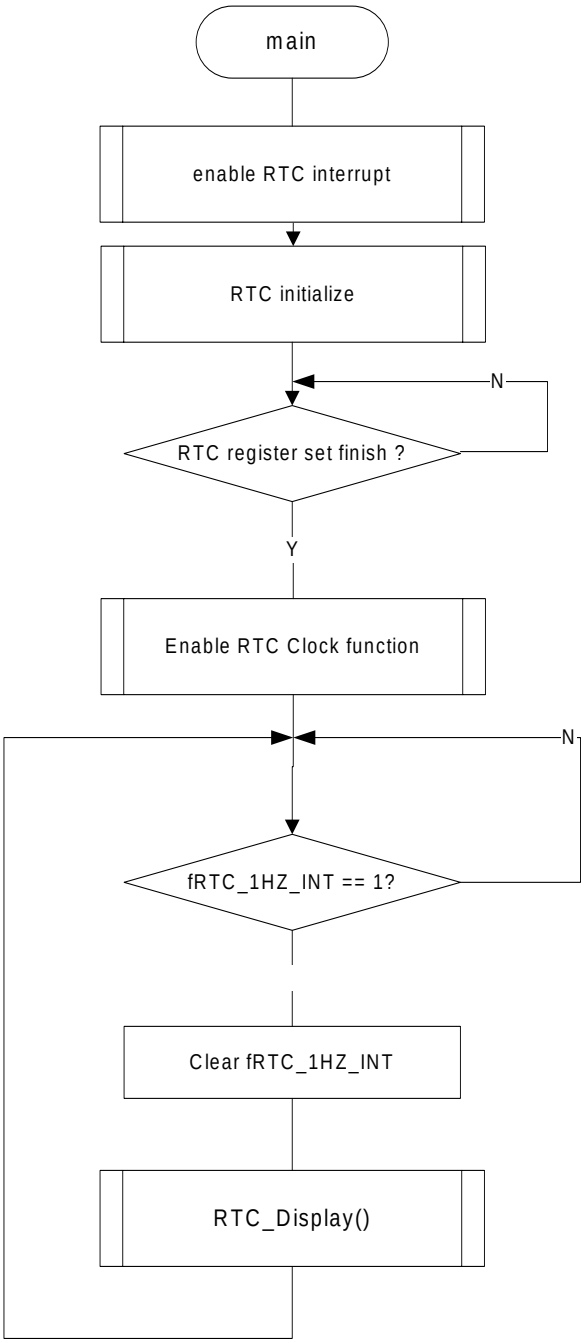
```
RMC_RxDataTypeDef myRMC_RxDataDef;
myRMC_RxDataDef = RMC_GetRxData(TSB_RMC);
```

7-9 RTC

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

The example includes:
1. RTC initialization
2. Get RTC date and time

- **Flow chart:**



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

At first set source(RTC interrupt) to exit sleep mode

```
CG_SetSTBYReleaseINTSrc(CG_INT_SRC_RTC,
                        CG_INT_ACTIVE_STATE_FALLING, ENABLE);
```

Initialize RTC .Create a RTC_DateTypeDef and RTC_TimeTypeDef structure and fill all the data fields. For example, Initial setting: 2015/06/12 13:54:00, 24hours format.

```
Date_Struct.LeapYear = RTC_LEAP_YEAR_3;
Date_Struct.Year = (uint8_t) 15U;
Date_Struct.Month = (uint8_t) 6U;
Date_Struct.Date = (uint8_t) 12U;
```

```
Date_Struct.Day = RTC_FRI;

Time_Struct.HourMode = RTC_24_HOUR_MODE;
Time_Struct.Hour = (uint8_t) 13U;
Time_Struct.Min = (uint8_t) 54U;
Time_Struct.Sec = (uint8_t) 00U;
```

Disable clock and alarm function.

```
RTC_DisableClock();
RTC_DisableAlarm();
```

Reset RTC sec counter, enable clock and alarm function.

```
RTC_ResetClockSec();
RTC_EnableClock();
RTC_EnableAlarm();
```

Set RTC time value, date value,1Hz interrupt and *RTCINT*.

```
RTC_SetTimeValue(&Time_Struct);
RTC_SetDateValue(&Date_Struct);
RTC_SetAlarmOutput(RTC_PULSE_1_HZ);
RTC_SetRTCINT(ENABLE);
```

After the setting above, enable RTC interrupt. Waiting for RTC register set finished , then enable RTC Clock function.

```
NVIC_EnableIRQ(INTRTC_IRQn);
__enable_irq();

/* waiting for RTC register set finish */
while (fRTC_1HZ_INT != 1U) {
      /* Do nothing */
}
fRTC_1HZ_INT = 0U;

/* Enable RTC Clock function */
RTC_EnableClock();
```

In RTC interrupt request function,The RTC interrupt will occurs a interrupt flag every second. Then RTC interrupt request will be cleared.

```
fRTC_1HZ_INT = 1U;
/*clear RTC interrupt request */
CG_ClearINTReq(CG_INT_SRC_RTC);
```

After interrupt flag occus RTC ones place of second value will be sent to LED. Following is shown how to get RTC date and time value.

```
/* Send RTC Date value to LCD */
/* Get RTC Date value */
Year = RTC_GetYear();
Month = RTC_GetMonth();
Date = RTC_GetDate(RTC_CLOCK_MODE);

/* Set LCD display */
/* Dispaly year */
```

```
RTC_Disp_YMD[0] = '2';
RTC_Disp_YMD[1] = '0';
RTC_Disp_YMD[2] = (Year / 10U) + 0x30U;
RTC_Disp_YMD[3] = (Year % 10U) + 0x30U;
RTC_Disp_YMD[4] = '/';
/* Display month */
RTC_Disp_YMD[5] = (Month / 10U) + 0x30U;
RTC_Disp_YMD[6] = (Month % 10U) + 0x30U;
RTC_Disp_YMD[7] = '/';
/* Display date */
RTC_Disp_YMD[8] = (Date / 10U) + 0x30U;
RTC_Disp_YMD[9] = (Date % 10U) + 0x30U;

RTC_Disp_YMD[10] = 0U;
Send_To_LCD(FIRST_LINE, LCD_COMMAND);
Send_LCD_Text(RTC_Disp_YMD);

/* Send RTC Time value to LCD */
/* Get RTC Time value */
Hour = RTC_GetHour(RTC_CLOCK_MODE);
Min = RTC_GetMin(RTC_CLOCK_MODE);
Sec = RTC_GetSec();

/* Display hour */
RTC_Disp_HMS[0] = (Hour / 10U) + 0x30U;
RTC_Disp_HMS[1] = (Hour % 10U) + 0x30U;
RTC_Disp_HMS[2] = ':';
/* Display min */
RTC_Disp_HMS[3] = (Min / 10U) + 0x30U;
RTC_Disp_HMS[4] = (Min % 10U) + 0x30U;
RTC_Disp_HMS[5] = ':';
/* Display sec */
RTC_Disp_HMS[6] = (Sec / 10U) + 0x30U;
RTC_Disp_HMS[7] = (Sec % 10U) + 0x30U;
RTC_Disp_HMS[8] = 0U;

/* LCD Display */
Send_To_LCD(SECOND_LINE, LCD_COMMAND);
Send_LCD_Text(RTC_Disp_HMS);
```

7-10 SBI

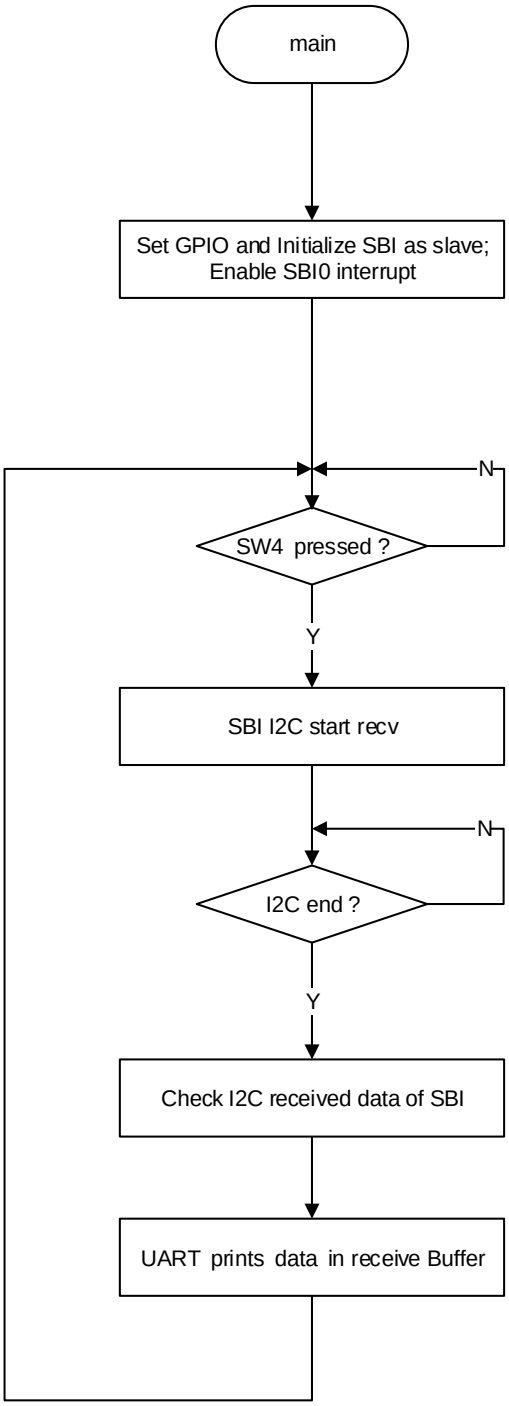
7-10-1 Example: SBI Slave

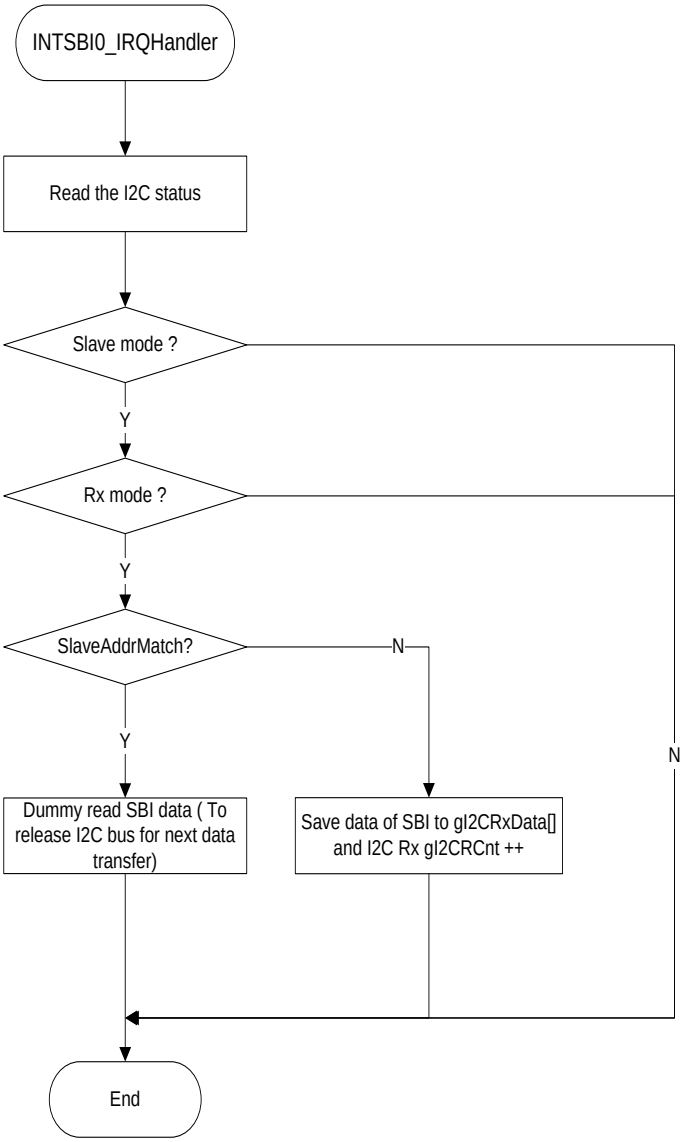
This is a simple example based on the TX03 Peripheral Driver (SBI, GPIO).

The example includes:

1. SBI configuration
2. SBI slave receive data process

- **Flow Chart**


main
Set GPIO and Initialize SBI as slave;
Enable SBI0 interrupt
N
SW4 pressed ?
Y
SBI I2C start recv
N
I2C end ?
Y
Check I2C received data of SBI
UART prints data in receive Buffer


INTSBI0_IRQHandler
Read the I2C status
Slave mode ?
Y
Rx mode ?
Y
SlaveAddrMatch?
N
Y
N
Dummy read SBI data (To release I2C bus for next data transfer)
Save data of SBI to gI2CRxData[] and I2C Rx gI2CRCnt ++
End

- **Code and Explanation for the Example**

At first, configure GPIO for SBI I2C mode.

```
    GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_3, GPIO_BIT_0 |
GPIO_BIT_1);
    GPIO_SetOutputEnableReg(GPIO_PC, GPIO_BIT_0 | GPIO_BIT_1,
ENABLE);
    GPIO_SetInputEnableReg(GPIO_PC, GPIO_BIT_0 | GPIO_BIT_1, ENABLE);
    GPIO_SetOpenDrain(GPIO_PC, GPIO_BIT_0 | GPIO_BIT_1, ENABLE);
/* Pull up for SDA&SCL */
    GPIO_SetPullUp(GPIO_PC, GPIO_BIT_0 | GPIO_BIT_1, ENABLE);
```

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

```
    myI2C.I2CSelfAddr = SLAVE_ADDR;
    myI2C.I2CDataLen = SBI_I2C_DATA_LEN_8;
    myI2C.I2CACKState = ENABLE;
    myI2C.I2CClkDiv = SBI_I2C_CLK_DIV_328;
    SBI_Enable(TSB_SBI0);
    SBI_SWReset(TSB_SBI0);
    SBI_InitI2C(TSB_SBI0, &myI2C);
    NVIC_EnableIRQ(INTSBI0_IRQn);
```

After the above setting, start I2C wait for receive.

Clear I2C Rx buffer, initialize the SBI Rx buffer and length,

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

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

```
void INTSBI0_IRQHandler(void)
{
    uint32_t tmp = 0U;
    TSB_SBI0_TypeDef *SBIy;
    SBI_I2CState sbi0_sr;

    SBIy = TSB_SBI0;
    sbi0_sr = SBI_GetI2CState(SBIy);
```

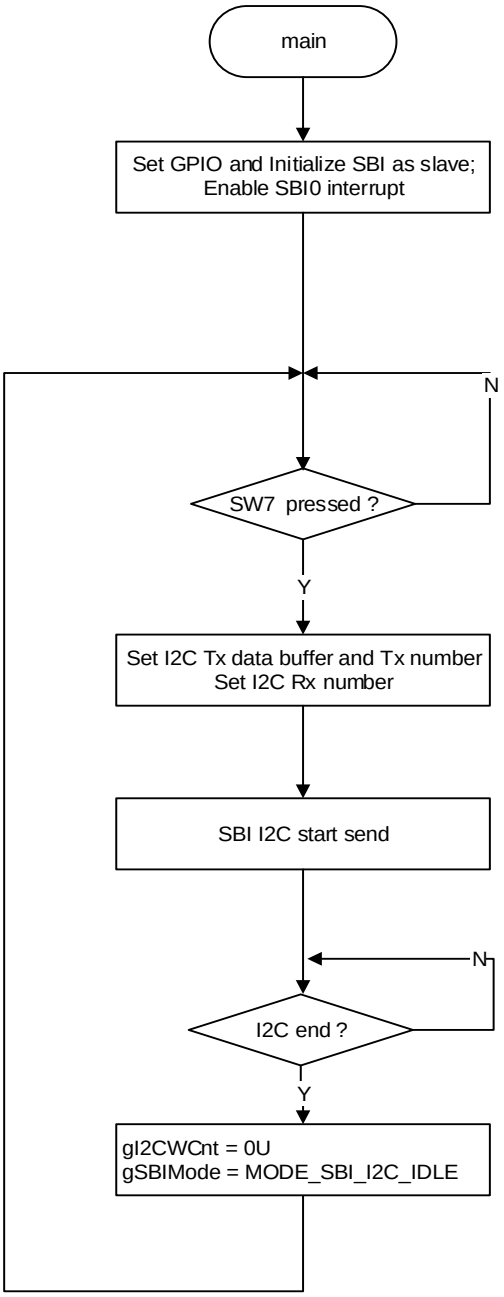
```
    if (!sbi0_sr.Bit.MasterSlave) {        /* Slave mode */
        if (!sbi0_sr.Bit.TRx) { /* Rx Mode */
            if (sbi0_sr.Bit.SlaveAddrMatch) {
                /* First read is dummy read for Slave address recognize */
                tmp = SBI_GetReceiveData(SBIy);
                gI2CRCnt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

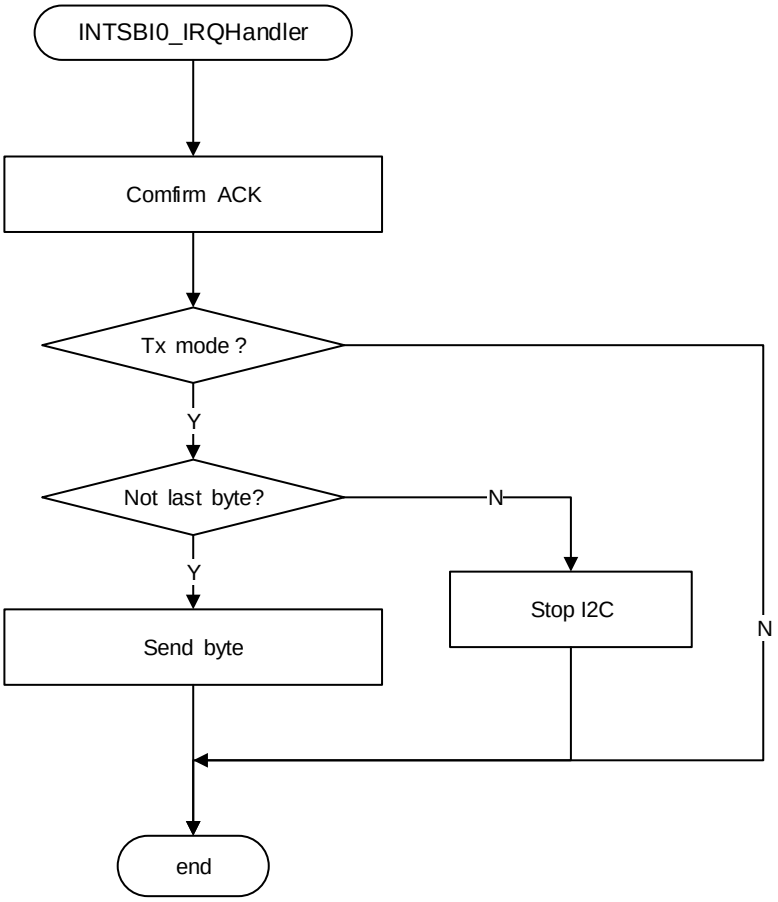
7-10-2 Example: SBI Master

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

1. SBI configuration
2. SBI master send data process

- **Flowchart**


main
Set GPIO and Initialize SBI as slave;
Enable SBI0 interrupt
N
SW7 pressed ?
Y
Set I2C Tx data buffer and Tx number
Set I2C Rx number
SBI I2C start send
N
I2C end ?
Y
gI2CWCnt = 0U
gSBIMode = MODE_SBI_I2C_IDLE



- **Code and Explanation for the Example**

At first, configure GPIO for SBI I2C mode.

```
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_3, GPIO_BIT_0 |
GPIO_BIT_1);
GPIO_SetOutputEnableReg(GPIO_PC, GPIO_BIT_0 | GPIO_BIT_1, ENABLE);
GPIO_SetInputEnableReg(GPIO_PC, GPIO_BIT_0 | GPIO_BIT_1, ENABLE);
GPIO_SetOpenDrain(GPIO_PC, GPIO_BIT_0 | GPIO_BIT_1, ENABLE);
/* Pull up for SDA&SCL */
GPIO_SetPullUp(GPIO_PC, GPIO_BIT_0 | GPIO_BIT_1, ENABLE);
```

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

```
    myI2C.I2CSelfAddr = SELF_ADDR;
    myI2C.I2CDataLen = SBI_I2C_DATA_LEN_8;
    myI2C.I2CACKState = ENABLE;
    myI2C.I2CClkDiv = SBI_I2C_CLK_DIV_328;
    SBI_Enable(TSB_SBI0);
```

```
    SBI_SWReset(TSB_SBI0);
    SBI_InitI2C(TSB_SBI0, &myI2C);
    NVIC_EnableIRQ(INTSBI0_IRQn);
```

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

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

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

Check if the I2C bus is free or not, SBI_SetSendData() is used to set data “SLAVE_ADDR”.and direction is “SBI_I2C_SEND” to SBI data buffer; then SBI_GenerateI2CStart(TSB_SBI0) is used to to start I2C process.

```
        /* Check I2C bus state and start TRx */
    case MODE_SBI_I2C_START:
        i2c_state = SBI_GetI2CState(TSB_SBI0);
        if (!i2c_state.Bit.BusState) {
            SBI_SetSendData(TSB_SBI0, SLAVE_ADDR | SBI_I2C_SEND);
            SBI_GenerateI2CStart(TSB_SBI0);
            gSBIMode = MODE_SBI_I2C_TRX;
        } else {
            /* Do nothing */
        }
        break;
```

The data transfer is handled in INTSBI0.

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

```
void INTSBI0_IRQHandler(void)
{
```

```
    TSB_SBI0_TypeDef *SBIx;
    SBI_I2CState sbi_sr;

    SBIx = TSB_SBI0;
    sbi_sr = SBI_GetI2CState(SBIx);

    if (sbi_sr.Bit.MasterSlave) {          /* Master mode */
        if (sbi_sr.Bit.TRx) {                    /* Tx mode */
            if (sbi_sr.Bit.LastRxBit) { /* LRB=1: the receiver requires no further
data. */
                SBI_GenerateI2CStop(SBIx);
            } else {                /* LRB=0: the receiver requires further data. */
                if (gI2CWCnt <= gI2CTxDataLen) {
                    SBI_SetSendData(SBIx, gI2CTxData[gI2CWCnt]);
                                                              /* Send next data */
                    gI2CWCnt++;
                } else {                                  /* I2C data send finished. */
                    SBI_GenerateI2CStop(SBIx);    /* Stop I2C */
                }
            }
        } else {                    /* Rx Mode */
            /* Do nothing */
        }
    } else {                        /* Slave mode */
        /* Do nothing */
    }
}
```

7-11 SSP

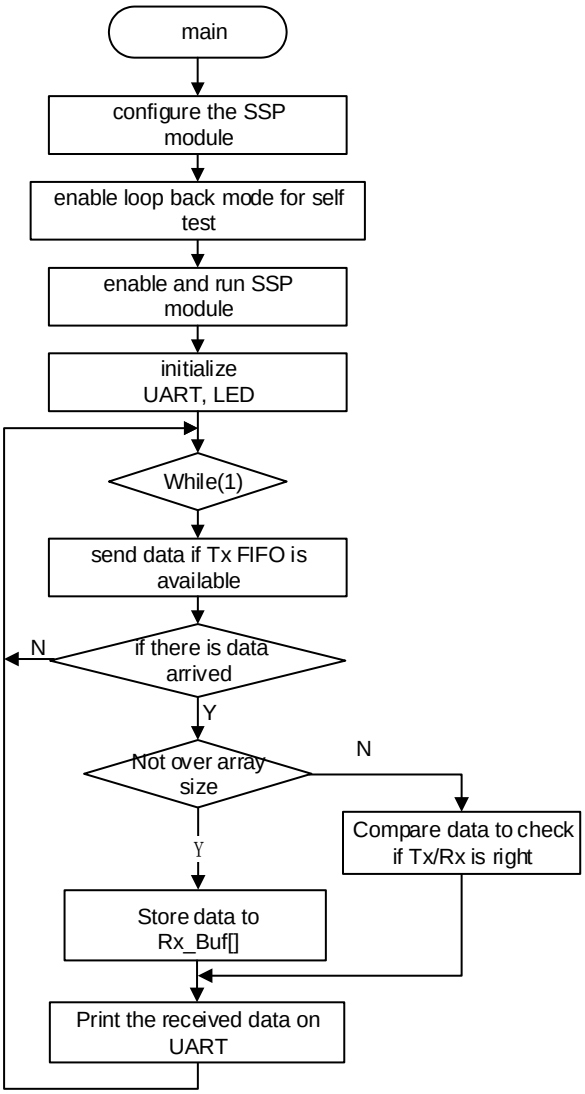
7-11-1 Example: SSP0 Self Loop Back

This is a simple example based on the TX03 Peripheral Driver (SSP, GPIO and UART).
This demo will enable the "Loop back mode" of SSP module to send data and receive it.
User can define BITRATE_MIN to see the effect due to too slow transmit bit rate.
The four LEDs will blink to show the data is being received.

The example includes:

1. Configuration and initialization SSP module channel 0
2. Enable its loop back mode to do self Tx/Rx

- **Flowchart**



- **Code and Explanation for the Example**

Select SPI frame format.

```
    /* Configure the SSP module */
    initSSP.FrameFormat = SSP_FORMAT_SPI;
```

Setting maximum and minimum bit rate.

```
    /* Default is to run at maximum bit rate */
    initSSP.PreScale = 2U;
    initSSP.ClkRate = 1U;

    /* Define BITRATE_MIN to run at minimum bit rate */
    /*     BitRate = fSYS / (PreScale x (1 + ClkRate)) */
#ifdef BITRATE_MIN
    initSSP.PreScale = 254U;
    initSSP.ClkRate = 255U;
#endif
    initSSP.ClkPolarity = SSP_POLARITY_LOW;
    initSSP.ClkPhase = SSP_PHASE_FIRST_EDGE;
    initSSP.DataSize = 16U;
```

```
initSSP.Mode = SSP_MASTER;
SSP_Init(TSB_SSP0, &initSSP);
```

Enable Loop back mode and SSP0

```
/* Enable loop back mode for self test */
SSP_SetLoopBackMode(TSB_SSP0, ENABLE);

/* Enable and run SSP module */
SSP_Enable(TSB_SSP0);
```

Setting for LED on M38x board

```
/* Initialize LEDs on M38x board before display something */
LED_Init();
hardware_init(UART_RETARGET);
```

In while loop, data is sent and checked if the received data is the same as the send one the led 2 and led 3 will light on, if not same the led 0 and led 1 will light on, and the receive data will be printed on UART as below.

```
while (1) {

    datTx++;
    /* Send data if Tx FIFO is available */
    fifoState = SSP_GetFIFOState(TSB_SSP0, SSP_TX);
    if ((fifoState == SSP_FIFO_EMPTY) || (fifoState == SSP_FIFO_NORMAL))
    {
        SSP_SetTxData(TSB_SSP0, datTx);
        if (cntTx < MAX_BUFSIZE) {
            Tx_Buf[cntTx] = datTx;
            cntTx++;
        } else {
            /* Do nothing */
        }
    } else {
        /* Do nothing */
    }

    /* Check if there is data arrived */
    fifoState = SSP_GetFIFOState(TSB_SSP0, SSP_RX);
    if ((fifoState == SSP_FIFO_FULL) || (fifoState == SSP_FIFO_NORMAL)) {
        receive = SSP_GetRxData(TSB_SSP0);
        if (cntRx < MAX_BUFSIZE) {
            Rx_Buf[cntRx] = receive;
            cntRx++;
        } else {
            /* Place a break point here to check if receive data is right. */
            /* Success Criteria:       */
            /* Every data transmitted from Tx_Buf is received in Rx_Buf. */
            /* When the line "#define BITRATE_MIN" is commented, the SSP
               is run in maximum */
            /* bit rate, so we can find there is enough time to transmit date
               from 1 to    */
            /* MAX_BUFSIZE one by one. But if we uncomment that line,
               SSP is run in */
            /* minimum bit rate, we will find that receive data can't catch
                "datTx++", */
```

```
                    /* in this so slow bit rate, when the Tx FIFO is available, the
                        cntTx has       */
                    /* been increased so much. */
                    __NOP();
                    result = Buffercompare(Tx_Buf, Rx_Buf, MAX_BUFSIZE);
                    if (result == NOT_SAME)
                    {
                        LED_On(LED0);
                        LED_On(LED1);
                    } else
                    {
                        LED_On(LED2);
                        LED_On(LED3);
                    }
                }

            } else {
                /* Do nothing */
            }
            sprintf((char *) SSP_RX_Data, “SSP RX DATA : %d", receive);
            common_uart_disp (SSP_RX _Data);
            common_uart_disp (“\n”);
    }
```

7-12 TMRB

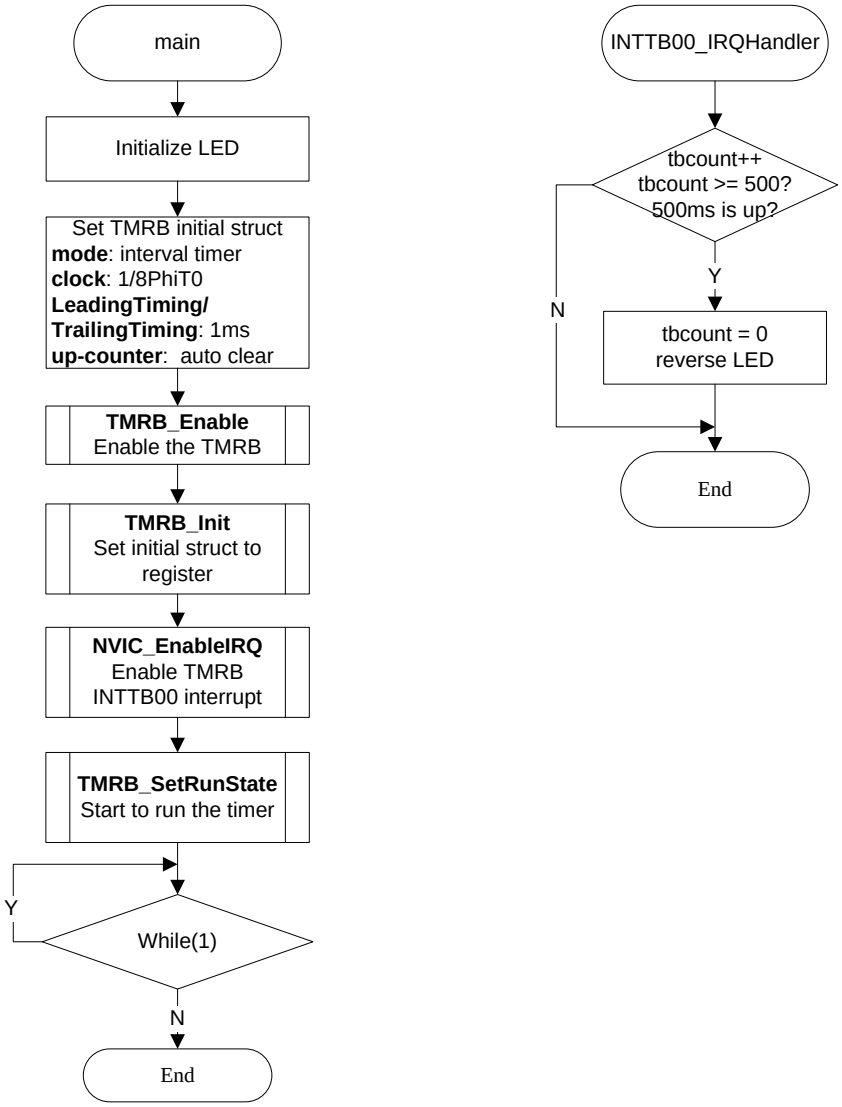
7-12-1 Example: General Timer

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 Eval board and turn on LED.

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

Prepare TMRB initialization structure, fill TMRB mode, clock, up-counter clear method, trailing timing and leading timing. This demo will set trailing timing and leading timing to 1ms, macro TMRB_1MS equals 0x1388, because fphiT0 = fsys = fc = 10MHz * PLL = 40MHz, ftmrb = 1/8 fphiT0 = 5MHz, Ttmrb = 0.2us, 1ms/0.2us = 5000 = 0x 1388 (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 INTTB0 interrupt, this interrupt will be triggered every 1ms, in the end, start the TMRB to run.

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

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

```
static uint16_t tbcount = 0U;
static uint8_t ledon = 1U;
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 */
}
```

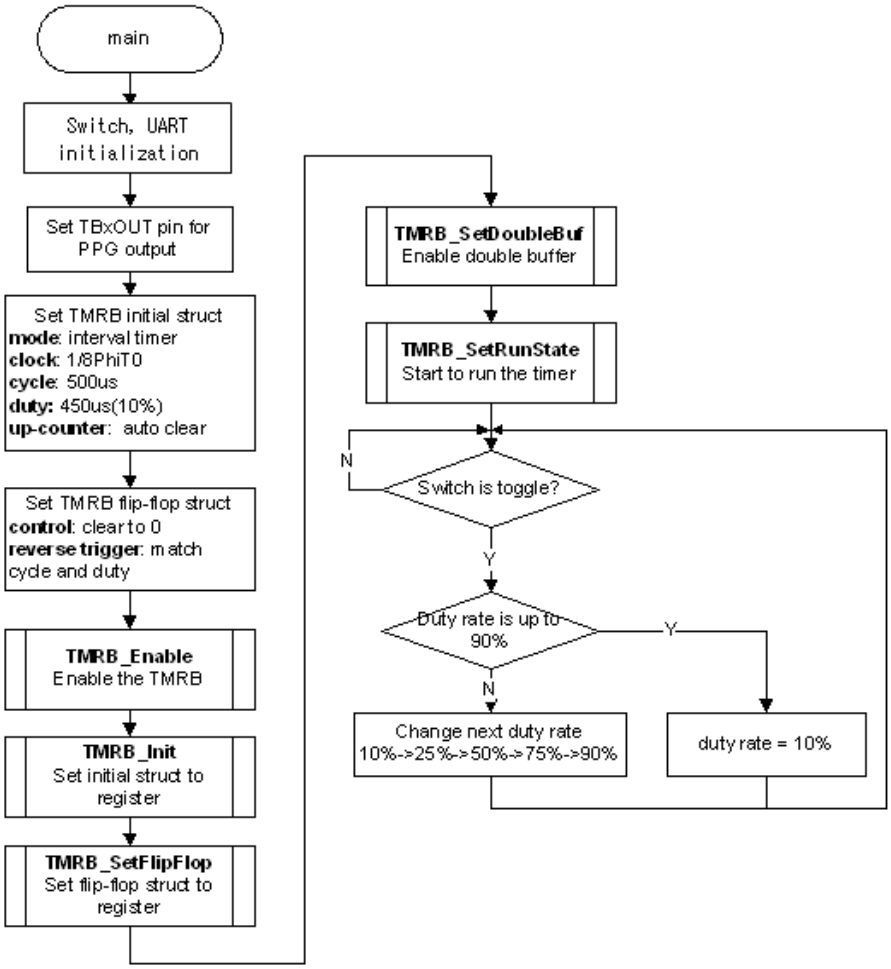
7-12-2 Example: PPG Output

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

The example includes:

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

- **Flow Chart**



- **Code and Explanation for the Example**

At first, initialize SWITCH and UART channel, set PA5 as TB6OUT for PPG output.

```
/* main.c – main function */
/* UART & switch initialization */
SW_Init();
hardware_init(UART_RETARGET);

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

Prepare TMRB initialization structure, and then fill TMRB mode, clock, up-counter clear method, trailing timing and leading timing into it. This demo will set trailing timing to 500us, macro TMRB6TIME equals 0x09C4, because fphiT0 = fsys = fc = 10MHz * PLL = 40MHz, ftmrb = 1/8 fphiT0 = 5MHz, Ttmrb = 0.2us, 500us/0.2us = 2500 = 0x09C4 (Please see CG part for more detail information about the clock setting)

```
/* main.c – main function */
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;
    m_tmrb.TrailingTiming = TMRB6TIME;              /* TrailingTiming is 500us */
    m_tmrb.LeadingTiming = LeadingTiming[Rate]; /* LeadingTiming is 450us */
```

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_TRAILINGTIMIN
G| TMRB_FLIPFLOP_MATCH_LEADINGTIMING;
```

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

```
    TMRB_Enable(TSB_TB6);
    TMRB_Init(TSB_TB6, &m_tmrb);
    TMRB_SetFlipFlop(TSB_TB6, &PPGFFInital);
    /* enable double buffer */
    TMRB_SetDoubleBuf(TSB_TB6,ENABLE, TMRB_WRITE_REG_SEPARATE);
    TMRB_SetRunState(TSB_TB6, TMRB_RUN);
```

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

```
  do {                          /* Handle the condition that start with the SW1 is high */
      keyvalue = SW_Get(SW1);
      LeadingTiming_display();
   } while (GPIO_BIT_VALUE_1 == 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) {    /* change LeadingTiming rate */
        Rate = LEADINGTIMINGINIT;
        } else {
            /* Do nothing */
        }

    TMRB_ChangeLeadingTiming(TSB_TB6, LeadingTiming[Rate]);
```

The calculation method of LeadingTiming value:

- TrailingTiming = 500us, ftmrb = 1/8 fphiT0 = 5MHz, Ttmrb = 0.2us (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 = 0x8CA
- LeadingTiming = 25%: high-level time is 500*25% = 125us, low-level time is 500-125 = 375us, counter value = 375us/Ttmrb = 0x753
- LeadingTiming = 50%: high-level time is 500*50% = 250us, low-level time is 500-250 = 250us, counter value = 250us/Ttmrb = 0x4E2
- LeadingTiming = 75%: high-level time is 500*75% = 375us, low-level time is 500-375 = 125us, counter value = 125us/Ttmrb = 0x271

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

That is the calculation method of LeadingTiming array:

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

7-13 SIO/UART

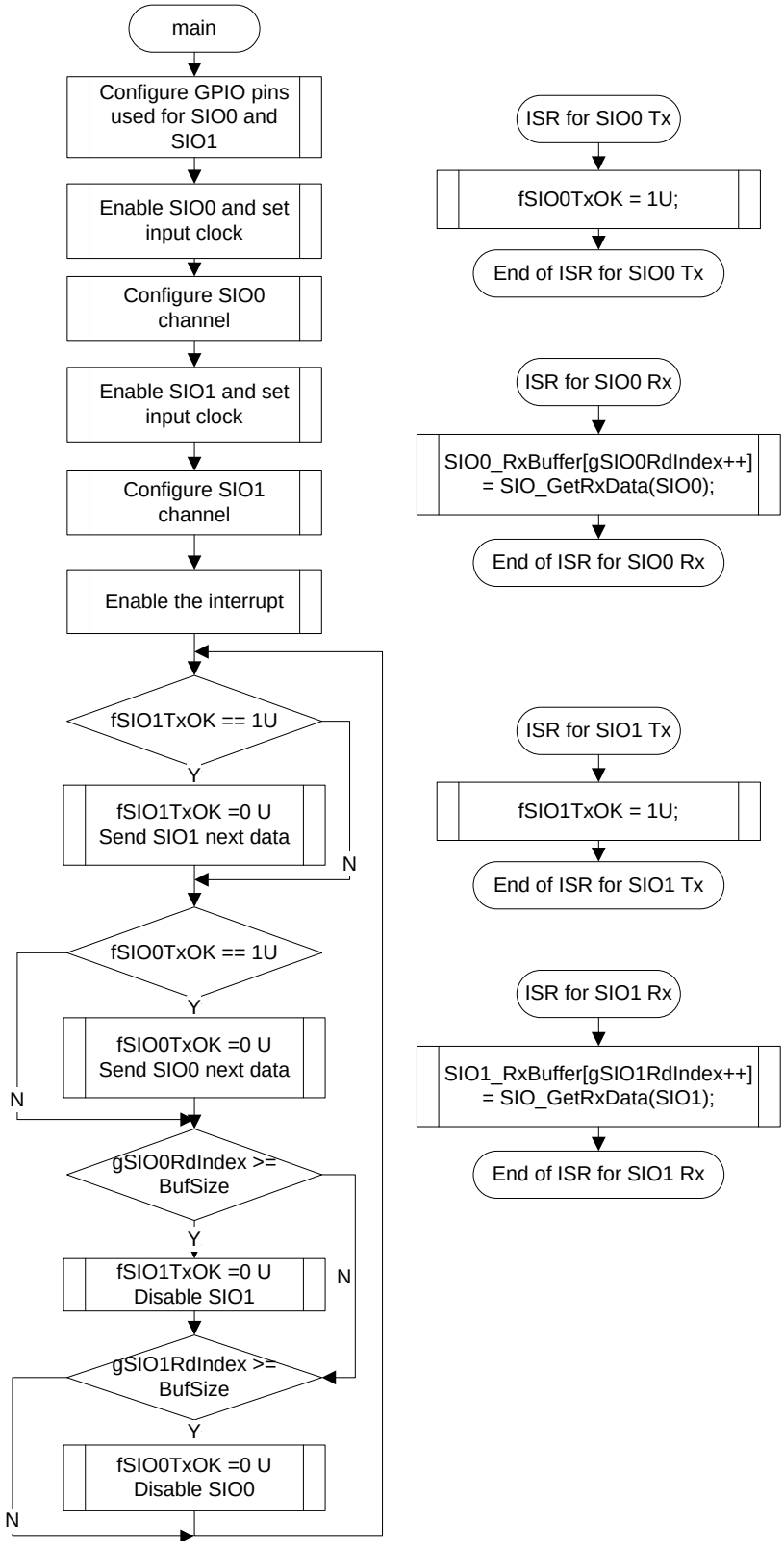
7-13-1 Example: SIO

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.

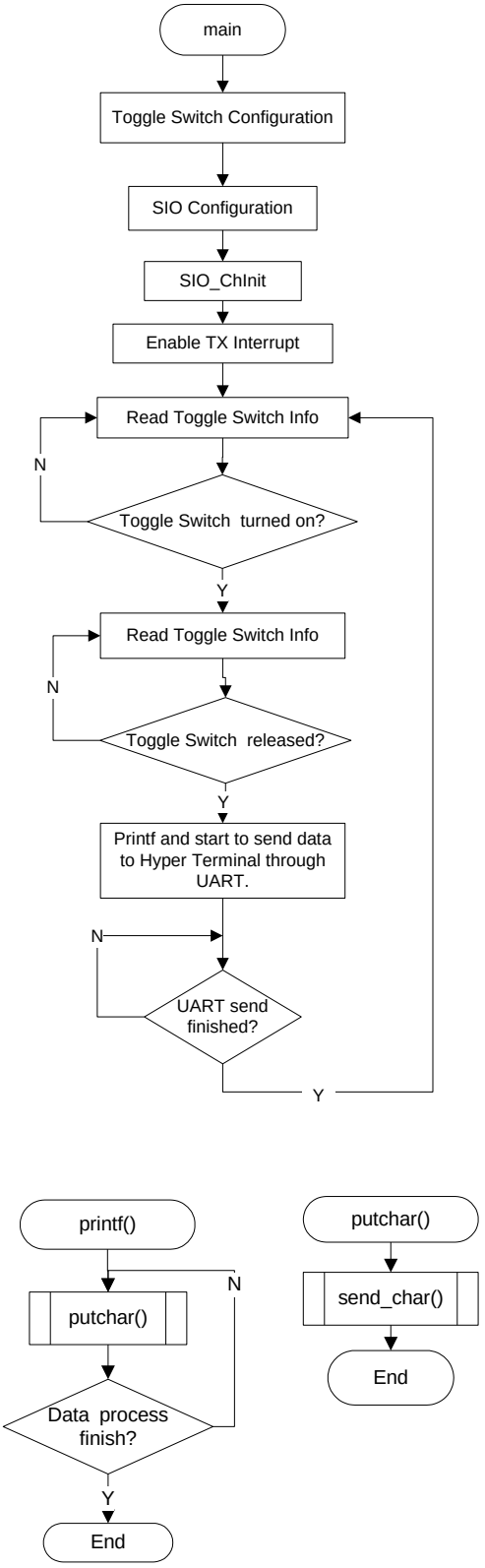
```
void INTRX1_IRQHandler(void)
{
    SIO0_RxBuffer[gSIO1RdIndex++] = SIO_GetRxData(SIO1);
}
```

7-13-2 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 UART1 TX interrupt to send data.
4. Retarget printf() to UART1.

- **Flowchart**



- **Code and Explanation for the Example**

At first, configure toggle switch and SIO0, then initialize UART0.

Use GPIO peripheral drivers configure GPIO for toggle switch.

```
GPIO_SetInput(GPIO_PH, GPIO_BIT_4 | GPIO_BIT_5 | GPIO_BIT_6 |
```

```
GPIO_BIT_7);
    GPIO_SetPullDown(GPIO_PH, GPIO_BIT_4 | GPIO_BIT_5 | GPIO_BIT_6 |
GPIO_BIT_7, ENABLE);
```

Use GPIO peripheral drivers configure GPIO for UART0.

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

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

```
    UART_InitTypeDef myUART;

    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.

```
    SIO_Configuration(UART_RETARGET);
    SIO_ChInit();
```

After above setting, enable UART0 TX interrupt.

```
    NVIC_EnableIRQ(RETARGET_INT)
```

Then Read Toggle Switch info:

```
    SW_info = SW_Get(SW_PORT);
```

Then Judge whether Toggle Switch 0 has been turned on, if Toggle Switch is turned on, then wait for Toggle Switch 0 released:

```
    if (SW_info == 1U) {
        /* wait for switch SW0 released */
        do {
            wait_SW0 = 0U;
            SW_info = SW_Get(SW_PORT);
            delay();
            } while (SW_info != SWRELEASE);
     }
```

After toggle switch 0 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(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 */
    }
}
```

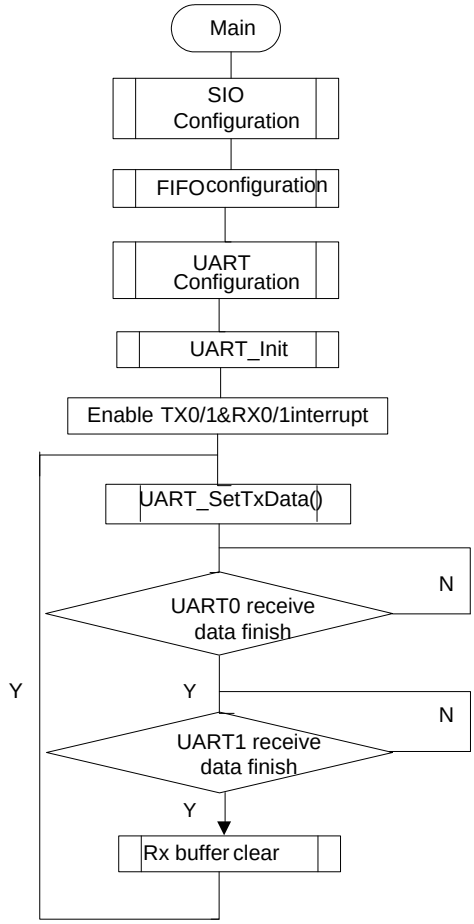
7-13-3 Example: UART FIFO

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

The example includes:

1. UART and FIFO configuration and initialization.
2. UART send and receive data use FIFO process.

- **Flowchart**



- **Code and Explanation for the Example**

At first configure the GPIO and initialize UART.

Use GPIO peripheral drivers configure GPIO for UART0 and UART1.

```
void SIO_Configuration(TSB_SC_TypeDef * SCx)
{
	if (SCx == TSB_SC0) {
		/*SET PE0 AS TXD0 */
		TSB_PE->FR1 |= 1U ;
		TSB_PE->CR |=  1U ;

		/*SET PE1 AS RXD0 */
		TSB_PE->FR1 |= (1U << 1U);
		TSB_PE->IE |= (1U << 1U);
	} else if (SCx == TSB_SC1) {
		/*SET PA5 AS TXD1 */
		TSB_PA->FR1 |= (1U << 5U);
		TSB_PA->CR |= (1U << 5U);

		/*SET PA6 AS RXD1 */
		TSB_PA->FR1 |= (1U << 6U);
		TSB_PA->IE |= (1U << 6U);
	}
}
```

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

```
UART_InitTypeDef myUART;

myUART.BaudRate = 115200U;
myUART.DataBits = UART_DATA_BITS_8;
myUART.StopBits = UART_STOP_BITS_1;
myUART.Parity = UART_NO_PARITY;
myUART.Mode = UART_ENABLE_TX|UART_ENABLE_RX;
myUART.FlowCtrl = UART_NONE_FLOW_CTRL;
```

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);
    }
}
```

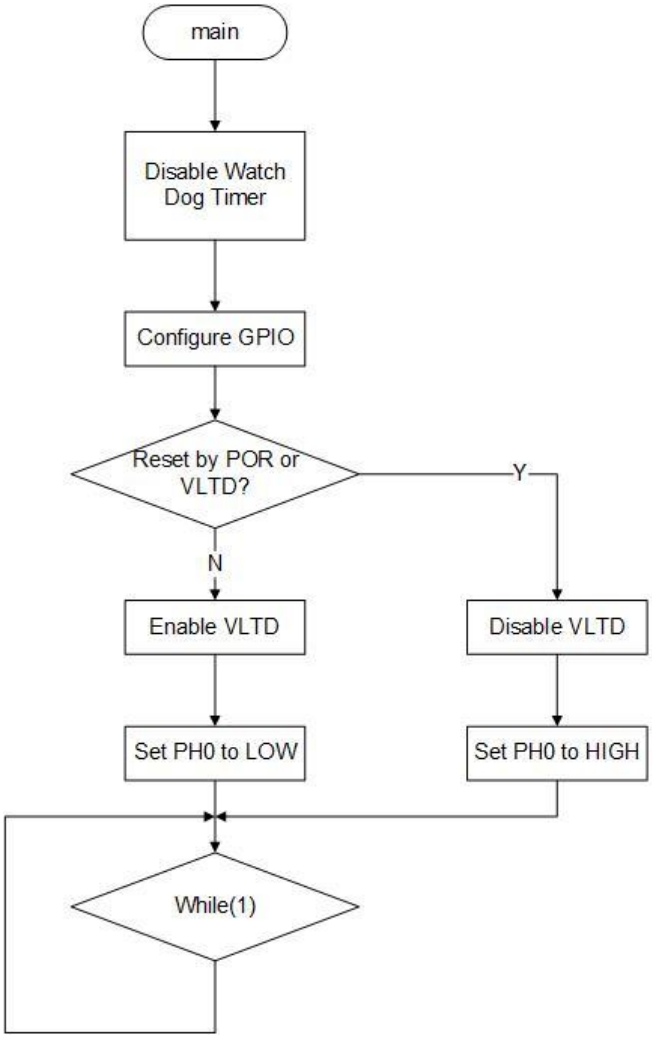
7-14 VLTD

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

The example includes:

1. VLTD configuration

- **Flowchart**



- **Code and Explanation for the Example**

At first, disable Watch Dog Timer explicitly. Then set PH0 as output port.

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

Then read Power On reset flag If it is not reset by VLTD or POR , configure and enable VLTD. Set PH0 to LOW.

```
if (CG_GetResetFlag().Bit._PowerOn == 0U) {
        VLTD_Enable();
        GPIO_WriteData(GPIO_PH, 0x00U);
```

Otherwise, disable VLTD and set PH0 to HIGH.

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

```
}
```

7-15 WDT

7-15-1 Example: WDT

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

1. WDT initialization.
2. In DEMO1 WDT isn't cleared before timer overflow, NMI interrupt is generated.
3. In DEMO2 WDT is cleared before timer overflow, the 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 and enable it.

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

In DEMO1 Waiting for the NMI interrupt flag.

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

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

```
WDT_Disable();
```

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

```
WDT_WriteClearCode();
```