

TX03 Peripheral Driver Usage Example (TMPM3U6)

TOSHIBA ELECTRONIC DEVICE SOLUTIONS CORPORATION

RESTRICTIONS ON PRODUCT USE

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

© 2019 Toshiba Electronic Device Solutions Corporation

Index

1	General description	1
2	Overview	1
3	Build-in hardware usage	1
4	Pin Usage.....	4
5	Development Environment.....	7
6	Functions.....	8
6-1	Operation mode.....	8
6-2	STARTUP	8
6-3	ADC	9
6-4	CG	9
6-4-1	Power mode change	9
6-5	DMAC	10
6-5-1	Memory to peripheral	10
6-6	FLASH	10
6-7	GPIO.....	12
6-8	RMC	12
6-8-1	RMC receiving.....	12
6-9	RTC	13
6-10	SBI.....	13
6-11	SIO/UART.....	14
6-11-1	Retarget.....	14
6-11-2	UART FIFO.....	14
6-11-3	SIO	14
6-12	SSP	14
6-12-1	SSP0 to SSP1 via DMAC.....	14
6-12-2	SSP0 Self Loop Back.....	14
6-13	TMRB	15
6-13-1	General Timer.....	15
6-13-2	PPG Output.....	15
6-14	VLTD.....	15
6-15	WDT	15
6-16	ENC.....	16
6-17	PMD.....	16
6-18	IGBT	17
6-18-1	Single PPG Output.....	17
6-18-2	Double PPG Output.....	18
6-19	OFD.....	18
7	Software	18
7-1	ADC	20
7-1-1	Example: Read AD Value By Software Trigger	20
7-2	CG	22

7-2-1	Example: Power mode change	22
7-3	DMAC	25
7-3-1	Example: Memory to peripheral	25
7-4	FLASH	27
7-5	GPIO	31
7-6	OFD	31
7-7	RMC	34
7-7-1	Example: RMC receiving.....	34
7-8	RTC	36
7-9	SBI.....	38
7-10	SIO/UART.....	41
7-10-1	Example: Retarget.....	41
7-10-2	Example: UART FIFO	45
7-10-3	Example: SIO	48
7-11	SSP	51
7-11-1	Example: SSP0 to SSP1 via DMAC.....	51
7-11-2	Example: SSP0 Self Loop Back.....	53
7-12	TMRB	55
7-12-1	Example: General Timer	55
7-12-2	Example: PPG Output.....	57
7-13	VLTD.....	60
7-14	WDT	61
7-15	ENC.....	62
7-15-1	Example: Rolling Detection	62
7-16	PMD.....	65
7-16-1	Example: Phase Output	65
7-17	IGBT	68
7-17-1	Example: Single PPG Output.....	68
7-17-2	Example: Double PPG Output	70
8	Revision History	72

Arm and Keil are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

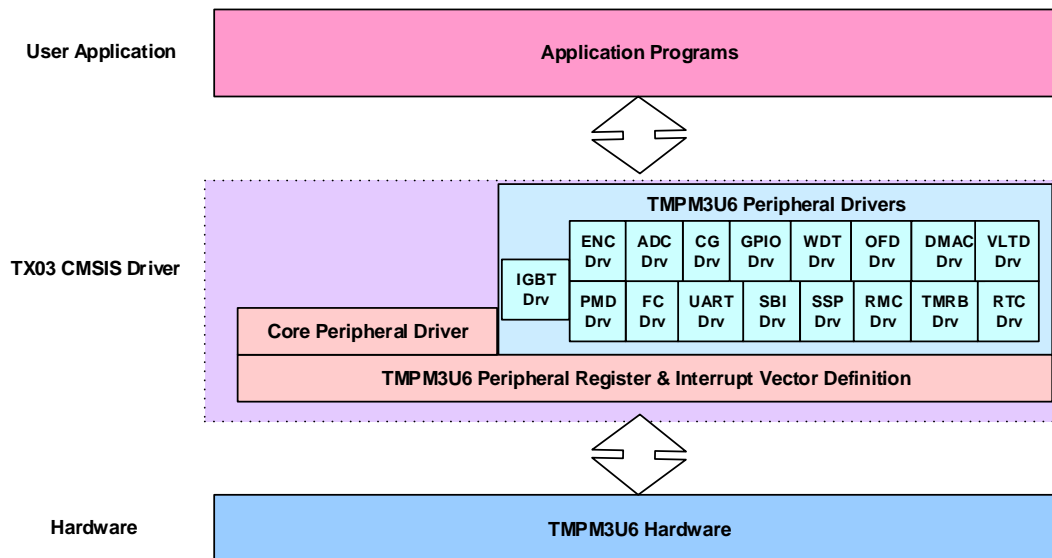
All other company names, product, and service names mentioned herein may be trademarks or registered trademarks of respective companies.

1 General description

The example programs described hereafter are specially designed for TOSHIBA TMPM3U6 MCU. The programs can be executed individually each to show the main MCU functions. Users can utilize some portions of the programs and integrate them into users' own programs to perform customized functions.

2 Overview

User application utilizes TX03 peripheral driver as following.



3 Build-in hardware usage

Hardware	Channel	Use presence, use
CG	Clock Gear	Used for CG demo
	PLL	PLL on (4x)
Standby mode	-	Use SLEEP mode
SysTick	-	Unused
Watch dog timer (WDT)	-	Used for WDT and OFD demo
External interrupt (INT)	INT0	Unused
	INT1	Unused
	INT2	Unused
	INT3	Unused
	INT4	Unused

Hardware	Channel	Use presence, use
	INT5	Unused
	INT6	Unused
	INT7	Unused
	INT8	Unused
	INT9	Unused
	INTA	Unused
	INTB	Unused
	INTC	Unused
	INTD	Unused
	INTE	Unused
	INTF	Unused
DMAC	ch1	Used for DMAC demo
SIO	SIO0	Used for UART & DMAC demo Tx
	SIO1	Used for UART & DMAC demo Rx
	SIO2	Used for UART FIFO demo
	SIO3	Unused
	SIO4	Unused
Synchronous serial port (SSP)	SSP0	Used for SSP transfer demo Tx
	SSP1	Used for SSP transfer demo Rx
Serial bus (SBI)	SBI0	Used for Master Tx
	SBI1	Used for Slave Rx
Remote control signal processor (RMC)	RMC0	Used for RMC signal receive
16-bit timer	TMRB0	Used for TMRB: General Timer
	TMRB1	Unused
	TMRB2	Unused
	TMRB3	Unused
	TMRB4	Unused
	TMRB5	Unused
	TMRB6	Unused
	TMRB7	Used for TMRB: PPG Output
16-bit multi-purpose timer	MPT0	Used for IGBT demo: OnePPG and TwoPPG
	MPT1	Used for PMD demo: Phase Output
	MPT2	Unused
RTC	-	Used for RTC demo
OFD	-	Used for Startup demo
VLTD	-	Used for low voltage detection
Encoder input function	ENC0	Used for ENC Roller demo
	ENC1	Unused
10-bit A/D converter	AIN0	Used for ADC data read
	AIN1	Unused

TX03 Peripheral Driver Usage Example

Hardware	Channel	Use presence, use
	AIN2	Unused
	AIN3	Unused
	AIN4	Unused
	AIN5	Unused
	AIN6	Unused
	AIN7	Unused
	AIN8	Unused
	AIN9	Unused
	AIN10	Unused
	AIN11	Unused
	AIN12	Unused
	AIN13	Unused
	AIN14	Unused
	AIN15	Unused
	AIN16	Unused
	AIN17	Unused
Flash	-	Used for Flash_Userboot

4 Pin Usage

The example programs are tested on the IAR TMPM3U6 Evaluation Board.
Following is pin usage for example programs.

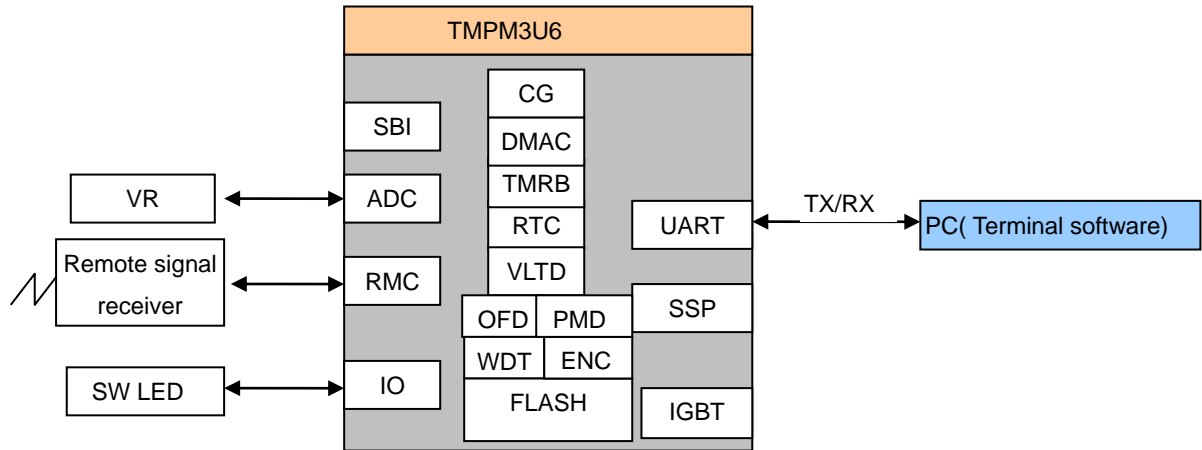
No	Name	Usage
1	PD4, SCLK2, CTS2	Unused
2	PD3, INT9	Unused
3	PD2, ENCZ0, INTD	Unused
4	PD1, ENCB0, TB5OUT	ENCB0
5	PD0, ENCA0, TB5IN, INTC	ENCA0
6	DVSS	GND
7	PC7, MT0IN, RX4	IGBT Trg input
8	PC6, EMG0	IGBT EMG input
9	PC5, ZO0, MTOUT10, MTTB0IN, SCLK4, CTS4	PPG output
10	PC4, WO0, MTOUT00, MTTB0OUT	PPG output
11	PC3, YO0, SP0FSS	SP0FSS
12	PC2, VO0, SP0CLK	SP0CLK
13	PC1, XO0, SP0DI, SCL0 / SI0	SP0DI / SCL0
14	PC0, UO0, SP0DO, SDA0 / SO0	SP0DO / SDA0
15	DVDD5	+5V
16	PP1, XT2	Connect 32.768kHz oscillator
17	PP0, XT1	Connect 32.768kHz oscillator
18	PM1, X2	Connect 10MHz oscillator
19	CVSS	GND
20	PM0, X1	Connect 10MHz oscillator
21	DVSS	GND
22	PG0, UO1, SDA1 / SO1	SW0 / PMD UO1 output / SDA1
23	PG1, XO1, SCL1 / SI1	SW1 / PMD XO1 output / SCL1
24	PG2, VO1, SCK1	SW2 / PMD VO1 output / SCK1
25	PG3, YO1	SW3 / PMD YO1 output
26	PG4, WO1, MTOUT01, MTTB1OUT	LED0 / PMD WO1 output
27	PG5, ZO1, MTOUT11, MTTB1IN	LED1 / PMD ZO1 output
28	PG6, EMG1, GEMG1	LED2 / PMD EMG1 input
29	PG7, MT1IN	LED3
30	PA0, TB0IN, INT3	Unused
31	PA1, TB0OUT, SCOUT	Unused
32	PA2, TB1IN, INT4	Unused
33	PA3, TB1OUT, RXIN	RMC Receive
34	PA4, SCLK1, CTS1	SIO1/

No	Name	Usage
35	PA5, TXD1, TB6OUT	SIO1/ UART1
36	PA6, RXD1, TB6IN	SIO1/ UART1
37	PA7, TB4IN, INT8	Unused
38	PE0, TX0	SIO0/ UART0
39	PE1, RX0	SIO0/ UART0
40	PE2, SCLK0, CTS0	SIO0
41	PE3, TB4OUT	Unused
42	DVDD5	+5V
43	PE4, TB2IN, INT5	Unused
44	PE5, TB2OUT	Unused
45	DVSS	GND
46	PE6, TB3IN, INT6	Unused
47	PE7, TB3OUT, INT7	Unused
48	PN0, SP1DO	SP1DO
49	PN1, SP1DI	SP1DI
50	PN2, SPICLK	SPICLK
51	PN3, SPIFSS	SPIFSS
52	PN4, MTOUT02, MTTB2OUT	Unused
53	PN5, MTOUT12, MTTB2IN	Unused
54	PN6, GEMG2	Unused
55	PN7, MT2IN, INTE	Unused
56	PL2, INTF	Unused
57	DVSS	GND
58	MODE	GND
59	RVDD5	+5V
60	RESET	RESET
61	VOUT3	Unused
62	PH0, AIN0, INT0	AIN0
63	PH1, AIN1, INT1	Unused
64	PH2, AIN2, INT2	Unused
65	PH3, AIN3	Unused
66	PH4, AIN4	Unused
67	PH5, AIN5	Unused
68	PH6, AIN6	Unused
69	PH7, AIN7	Unused
70	PI0, AIN8	Unused
71	PI1, AIN9	Unused
72	PJ0, AIN10	Unused
73	PJ1, AIN11	Unused
74	PJ2, AIN12	Unused
75	PJ3, AIN13	Unused
76	PJ4, AIN14	Unused

No	Name	Usage
77	PJ5, AIN15	Unused
78	PJ6, AIN16, INTA	Unused
79	PJ7, AIN17, INTB	Unused
80	AVSS	GND
81	AVDD5	+5V
82	DVSS	GND
83	DVDD5	+5V
84	PL0, BOOT	Unused
85	FTEST3	Unused
86	PB0, TRACECLK	Unused
87	PB1, TRACEDATA0	Unused
88	PB2, TRACEDATA1	Unused
89	PB3, TMS, SWDIO	TMS
90	PB4, TCK, SWCLK	TCK
91	PB5, TDO, SWV	TDO
92	PB6, TDI	TDI
93	PB7, TRST	TRST
94	PF0, TB7IN	Unused
95	PF1, TB7OUT, ALARM	PPG Output
96	PF2, ENCA1, SCLK3, CTS3	Unused
97	PF3, ENCB1, TX3	Unused
98	PF4, ENCZ1, RX3	Unused
99	PD6, RX2	UART2
100	PD5, TX2	UART2

5 Development Environment

Following is development environment:



1. Hardware board:
TMPM3U6 Evaluation Board.
2. Development tool:
IAR:
IDE: IAR Embedded workbench Version 8.22.2
ARM®:
IDE: ARM KEIL® MDK Version 5.24.2.0

6 Functions

This chapter will focus on the functions demonstrated in driver usage example.

6-1 Operation mode

There are five operation modes for TMPM3U6: NORMAL, SLOW, IDLE, SLEEP and STOP mode.

IDLE, SLEEP and STOP mode are low power mode.

To shift to lower power mode, in system control register STBYCR0<STBY2:0>, select the IDLE, SLEEP or STOP mode, and run the WFI (Wait For Interrupt) command.

➤ **NORMAL mode:**

This mode is to operate the CPU core and the peripheral hardware by using the high-speed clock(fosc1 or fosc2). It is shifted to the NORMAL mode with fosc2(internal high-speed oscillator) after reset.

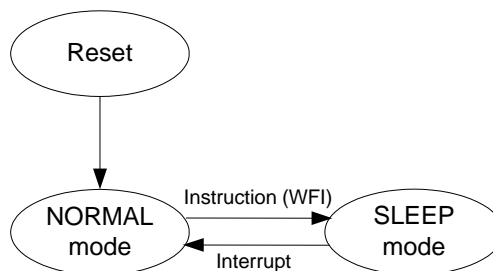
Note: Only SLEEP mode is demonstrated in driver example.

➤ **SLEEP mode:**

The internal low-speed oscillator (fs), real time clock and RMC can operate. By releasing the SLEEP mode, the device returns to the preceding mode of the SLEEP mode and starts operation.

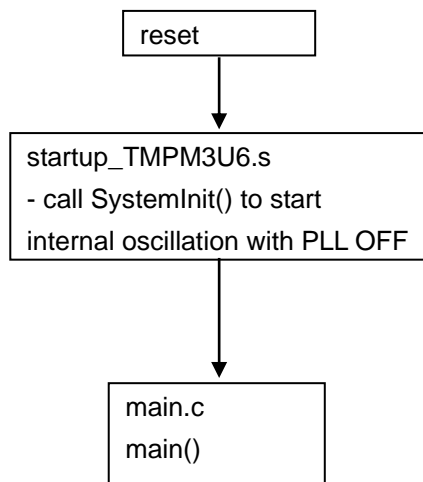
RTC interrupt, RMC interrupt and the extern interrupt can release the SLEEP mode.

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



6-2 STARTUP

Start up is running between reset and main function.



In main(), we demo the WDT and OFD functions, TMPM3U6 can be operated by both internal OSC and external OSC, however OFD circuit can detect only external OSC clock frequency using internal OSC clock.

When you use M3U6 OFD function, It is necessary to select "use the internal oscillation with PLL OFF" in "System Init", then start external oscillation and use the OFD drivers to perform the Oscillation Frequency Detector function.

6-3 ADC

There is a "slide volume" potentiometer connected to PN0/AIN0, Displays the measured AD conversion value in terminal.

6-4 CG

6-4-1 Power mode change

Change the CPU operation mode. Only 2 modes are supported, NORMAL and SLEEP. Press remote controller power key to switch the power mode. And also maybe set the peripheral power according to power mode.

<u>Current Mode</u>	<u>Action (remote key)</u>	<u>Operation</u>	<u>Terminal display</u>
NORMAL	Press [9], [CH DOWN], [VOL DOWN]	NORMAL → SLEEP	Now, Going to Sleep
SLEEP	Press [7], [CH UP], [VOL UP]	SLEEP → NORMAL	Wakeup from Sleep

6-5 DMAC

6-5-1 Memory to peripheral

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 TMPM3U6 Evaluation Board must be modified: Connect the TX of UART0 and RX of UART1 via wire.

6-6 FLASH

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

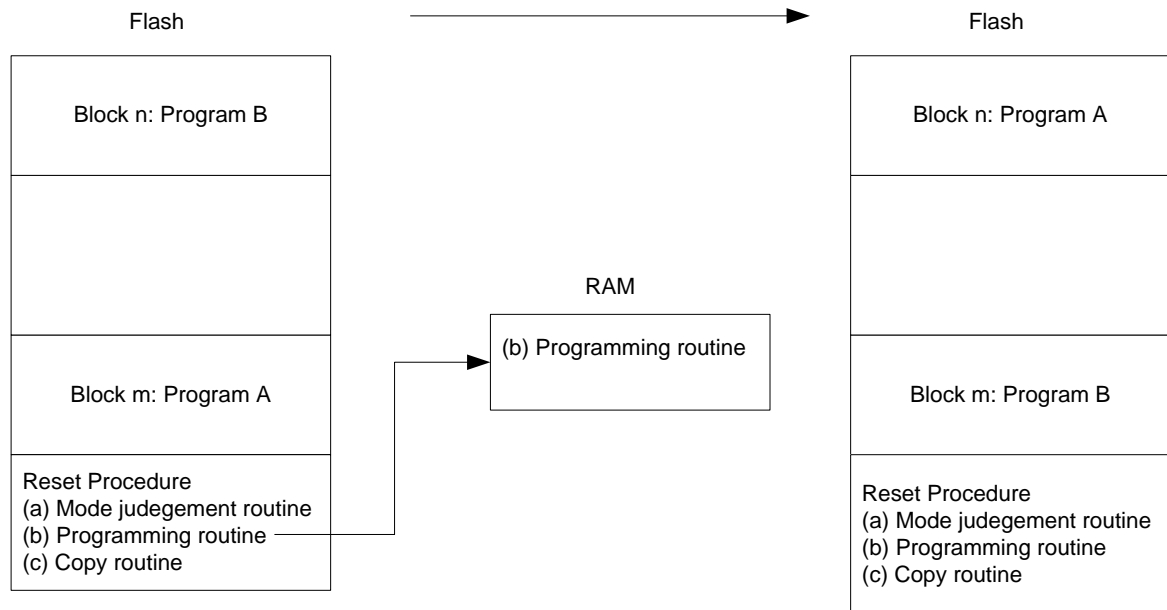
The demo runs in Normal mode and User Boot Mode of Single chip mode.

- Reset procedure: includes Mode judgment, Programming routine(flash APIs), Copy routine
- Mode judgment routine: judge to enter User Boot Mode or Normal Mode
- Copy routine: copy programming routine(flash APIs) from flash to RAM
- Programming routine: runs at RAM and swap Program A and Program B code in flash
- Program A/B (Reset procedure) are programmed in flash block in advance.
- Program A/B(A: LED0,2blink, B: LED1,3 blink)

By default, the program A will run firstly

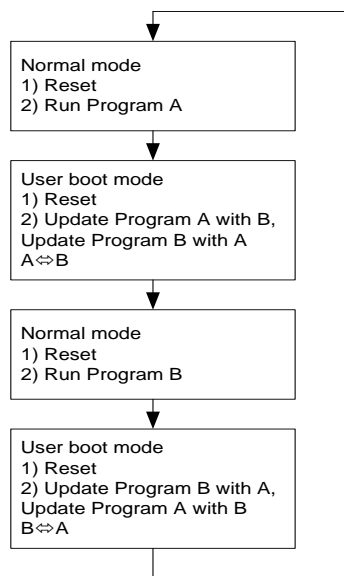
- Push switch1(SW0) is used for mode judgement in Reset procedure

Push switch0(SW0)-high position	Normal mode
Push switch0(SW0)-low position	User boot mode

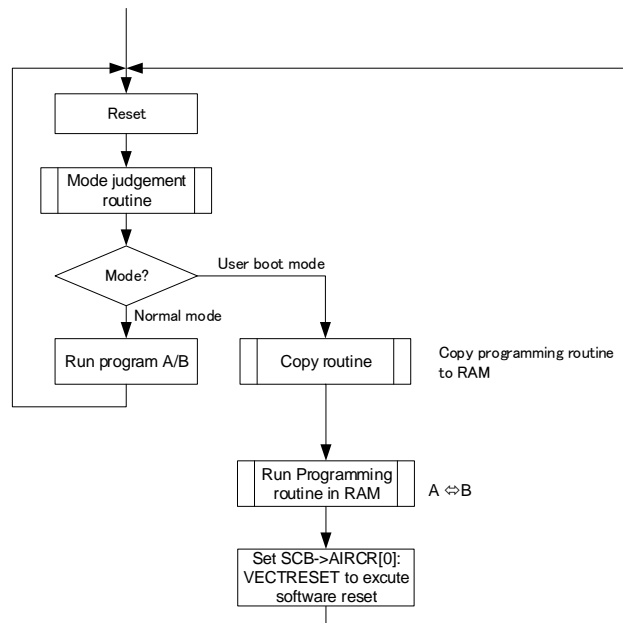


Demo sequence

- (1) Power on
Change the switch "SW0" to high position, then power on or press RESET button.
ProgramA will run at first.
LED 0,2 blink.
- (2) Change the switch "SW0" to low position, then press RESET button, LED will display 0->1->2->3.
- (3) After light LED3, change the switch "SW0" to high position, and system will reset automatically to run ProgramB.
LED 1,3 blink.



• Demo process flow



NOTE:

LED0~LED3 connect with PG4~PG7 in TMPM3U6 Evaluation board.

SW0 connect with PG0 in TMPM3U6 Evaluation board.

6-7 GPIO

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

6-8 RMC

6-8-1 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: TMPM3U6 RMC Demo

RMC_1: XX XX

RMC_1: YY YY

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

Following additional hardware connection should be added on the IAR TMPM3U6 Evaluation Board.

1: Connects a remote control receiver to PA3 on the IAR TMPM3U6 Evaluation Board.

2: The power 3V supply to the remote control receiver.

6-9 RTC

Use the internal RTC to display the time and date on terminal software.

Display the time and date on terminal software and update the date and time.

Initial setting: 2019/06/21 14:59:00, 24-hour format.

Update interval: 1 second.

Display example:

2019/6/21
14:59:00

6-10 SBI

This function intends to support the I2C bus slave mode.

Connect two I2C buses (SBI0 & SBI1) on TMPM3U6 evaluation board.

One is working as I2C master, and the other is working as I2C slave.

Use SBI interrupt to handle I2C bus read/write.

Address of I2C slave: 0xB0.

I2C Slave (SBI1) receives "TOSHIBA" from I2C Master (SBI0).

Received results can be checked by RAM variable gl2CrxData[] in debugger.

Note: In order to run this sample software, use TMPM3U6 evaluation board which must be modified.

Connect the pin PC1(SCL0) and pin PG1(SCL1).

Connect the pin PC0(SDA0) and pin PG0(SDA1).

6-11 SIO/UART

6-11-1 Retarget

This function intends to retarget the stdin and stdout of the C lib.
Stdin and stdout are retargeted to UART2. Then application code can use printf() to output data from serial port.

6-11-2 UART FIFO

In this sample program, UART1 sent the data "TMPM3U61" to UART2 use FIFO, and at same time UART1 receive the data "TMPM3U62" which send from UART2 and also use FIFO.
Set one breakpoint before the function ResetIdx(),when program stop in the breakpoint you can read the RxBuffer = "TMPM3U62",and RxBuffer1 = "TMPM3U61".

6-11-3 SIO

This demo will show synchronously transfer and receive usage of SIO module in TMPM3U6
It use the channel SIO0, SIO1 and transfer data synchronously between them.(connect TXD0 with RXD1, connect TXD1 with RXD0, connect sclk0 with sclk1)

6-12 SSP

6-12-1 SSP0 to SSP1 via DMAC

For M3U6, there are 2 SSP channels, SSP0 is set as SPI Mast while SSP1 is set as SPI Slave, then connect correspond pins, use DMA to transmit/receive infinite self increased data.

Note: TOSHIBA TMPM3U6 evaluation board must be modified as below:

	SSP0 pins	Note	SSP1 pins
1	PC3/SP0FSS	Connect PC3 to PN3	PN3/SP1FSS
2	PC2/SP0CLK	Connect PC2 to PN2	PN2/SP1CLK
3	PC1/SP0DI	Connect PC1 to PN0	PN1/SP1DI
4	PC0/SP0DO	Connect PC0 to PN1	PN0/SP1DO

* SSP0 to SSP1 via DMAC demo should run on TOSHIBA TMPM3U6 Evaluation Board.

6-12-2 SSP0 Self Loop Back

Infinite data is sent and checked if the received data is OK or not. And if transmission is set to lowest speed mode(lowest bit rate), what happens will be checked.

Note: The SD card socket must be left nothing inside.

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. Leading Timing can be changed as following:

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

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

Change the leading timing every switch changing:

10% → 25% → 50% → 75% → 90% → 10%

6-14 VLTD

This application implements power supply voltage status detecting by VLTD.

If the voltage is lower than the detection voltage, turn on LED1.

If the voltage is higher than the detection voltage, turn off LED1.

6-15 WDT

The watchdog timer cannot be used in the STOP mode, SLEEP mode and SLOW mode where high-speed frequency clock is stopped. Before transition to these modes, the watchdog timer should be disabled.

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.

DEMO1:

If timer is overflow then WDT will be cleared, and NMI interrupt is generated.

DEMO2:

WDT clear code will be written to the watchdog timer control register, and watch dog timer counter will be cleared.

6-16 ENC

This example detects the rolling of mouse wheel by TMPM3U6 ENC driver.

Demo procedure:

1. Use a wire to connect the pinA (see diagram 1) of mouse wheel encoder with the PD0 (ENCA0) .
2. Use a wire to connect the pinB (see diagram 1) of mouse wheel encoder with the PD1 (ENCB0).
3. Connect the USB of mouse with the PC, the Com (see diagram 1) of mouse wheel encoder will connect with 5V.
4. After running the program, LED1 will blink.
5. While rolling the mouse wheel forward, LED1 will blink faster and faster. When the blink speed reaches to the maximum, it will back to the minimum speed.
6. While rolling the mouse wheel backward, LED1 will blink slower and slower. When the blink speed reaches to the minimum, it will back to the maximum speed.

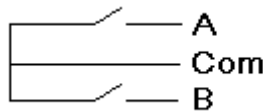


Diagram 1

6-17 PMD

This example demonstrates the phase outputs in PMD module.

Demo procedure:

1. Open the project and choose one demo to define: DEMO_U_PHASE or DEMO_3_PHASE. And then run the demo.
2. Use a wire to connect the EMG1 (PG6) with the DVSS (GND) to pull it low. (LED3 will light on) It means that PMD is in EMG protection.
3. Use a wire to connect the EMG1 (PG6) with the DVDD5 (TP2) to pull it high. (LED3 will light off) It means that PMD is in normal operation.
4. Connect UO1 (PG0) to oscilloscope to observe the waveform.
5. Connect XO1 (PG1) to oscilloscope to observe the waveform.
6. Connect VO1 (PG2) to oscilloscope to observe the waveform.
7. Connect YO1 (PG3) to oscilloscope to observe the waveform.
8. Connect WO1 (PG4) to oscilloscope to observe the waveform.
9. Connect ZO1 (PG5) to oscilloscope to observe the waveform.

Phenomenon:

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

The waveform duty cycle of UO1 is 25%, the upper output is 5V, and the period is 820us.XO1 is the inverting of UO1.

VO1 and WO1 are same with UO1, and YO1 and ZO1 are same with XO1

If the DEMO_3_PHASE is defined, the phase outputs are different.

The waveform duty cycle of UO1 is 25%, the upper output is 5V, and the period is 820us.XO1 is the inverting of UO1.

The waveform duty cycle of VO1 is 50%, the upper output is 5V, and the period is 820us.YO1 is the inverting of VO1.

The waveform duty cycle of WO1 is 75%, the upper output is 5V, and the period is 820us.ZO1 is the inverting of WO1.

6-18 IGBT

6-18-1 Single PPG Output

This example program demonstrates how to use external trigger to control the PPG output by TMPM3U6 IGBT driver and EMG function. The MPT channel 0 (IGBT0) is used and the PPG wave is generated from MTOUT00.

Pin assignment:

Function	Pin Name	Pin No.
Trigger input	MT0IN	7
PPG output 0	MTOUT00	10
EMG input	GEMG0	8

Demo procedure:

1. Use a wire to connect port GEMG0 with DVDD to pull it high.
2. Connect MTOUT00 to oscilloscope.
3. Power on, LED0 on and no waveform is observed from MTOUT00.
4. Connect MT0IN to the SW1 of the evaluation board.
5. Push the SW1.
6. The IGBT timer starts to run because of the falling edge of MT0IN and at the same time PPG waveform can be observed from MTOUT00. PPG cycle is 50us with duty 50%.
7. Connect GEMG0 with GND, then PPG output stops immediately and LED0 lights on.
8. Connect GEMG0 with DVDD again, and EMG state cancels with LED0 off, then repeat step 4 to observe the waveform.

6-18-2 Double PPG Output

This example program demonstrates how to use command start to control two PPG output by TMPM3U6 IGBT driver and EMG function. The MPT channel 0 (IGBT0) is used and two PPG waves are generated from MTOUT00 and MTOUT10.

Pin assignment:

Function	Pin Name	Pin No.
PPG output 0	MTOUT00	10
PPG output 1	MTOUT10	9
EMG input	GEMG0	8

Demo procedure:

1. Use a wire to connect port GEMG0 with DVDD to pull it high.
2. Connect MTOUT00 and MTOUT10 to oscilloscope.
3. Switch SW7 to OFF.
4. Power on, LED0 on and no waveforms are observed from MTOUT00 or MTOUT10.
5. Switch SW1 to ON.
6. The IGBT timer starts to run and at the same time PPG waveform can be observed from MTOUT00 and MTOUT10. Both PPG waveforms' cycle is 50us with duty 40 % (high level: 20us) and MTOUT10 delays MTOUT00 with 25us.
7. If switch SW1 to OFF, PPG waveform disappears.
8. Connect GEMG0 with GND, then PPG output stops immediately and LED0 lights on.
9. Connect GEMG0 with DVDD again, and EMG state cancels with LED0 off, then repeat step 5 to observe the waveform.

6-19 OFD

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

When the clock not exceeds the OFD detection frequency range, LED0 will be blinking. When the clock exceeds the OFD detection frequency range, OFD will generate a reset for I/O. Then software will be reset. The OFD reset flag will be set. If the flag is detected, OFD will be disabled and LED1 will be blinking.

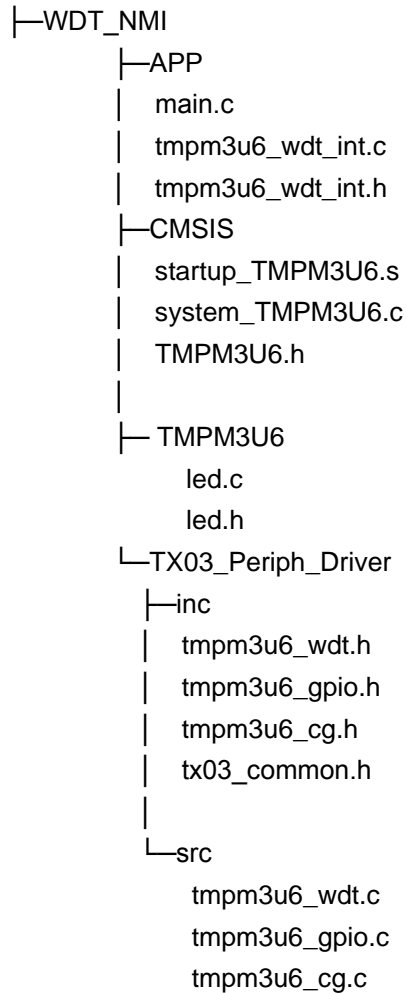
7 Software

This software project creates the sample applications based on IAR TMPM3U6 Evaluation Board which will demonstrate the main feature of TMPM3U6 MCU.

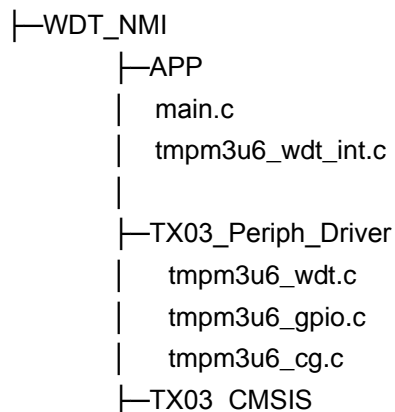
Use current driver software and IAR EWARM or KEIL MDK to implement the sample applications. Create an independent project for each feature.

The workspace structure and project names are defined as following:

IAR EWARM:



KEIL MDK:




```

| system_TMPM3U6.c
| startup_TMPM3U6.s
|
└─ TMPM3U6
    led.c
    
```

7-1 ADC

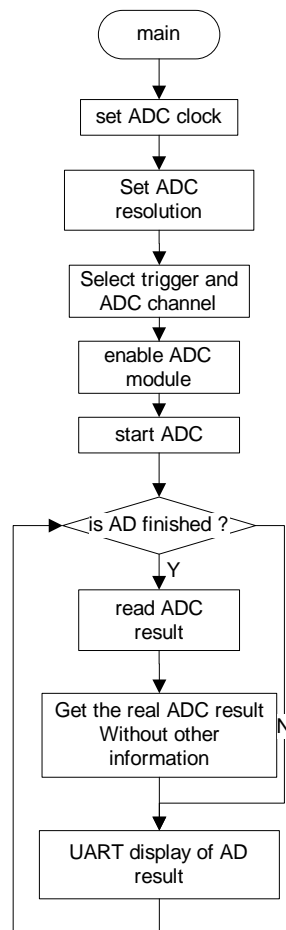
7-1-1 Example: Read AD Value By Software Trigger

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

The example includes:

1. ADC configuration and initialization
2. Read AD result of AIN0 by software trigger

• Flowchart



- **Code and Explanation for the Example**

At first, set ADC clock, select software trigger with ADC_REG0 and AIN0, disable low power mode, and enable ADC module.

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

/* 2. 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(ADC_REG0, TRG_ENABLE(ADC_AIN0));

/* 3. to let ADC module work, we must disable low power mode */
ADC_SetLowPowerMode(DISABLE);

/* 4. enable ADC module */
ADC_Enable();
```

Then start ADC:

```
/* 5. now start ADC */
ADC_Start(ADC_TRG_SW);
```

After ADC start, check AD conversion finished or not, When the AD conversion flag is set, the AD conversion result is acquired and displayed on the terminal

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

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

        /* get the real ADC result without other information */
        myResult = result.Bit.ADResult;

        /* software trigger, need to trig it again */
        ADC_Start(ADC_TRG_SW);
    } else {
        /* Do nothing */
    }

    if(fdisplay == 1U){
        fdisplay = 0U;
        printf("¥nAIN0 > 0x%03x",myResult);
    } else{
        /* Do nothing */
    }
}
```

After storing AD conversion result in "myResult" variable, it is displayed on the terminal

7-2 CG

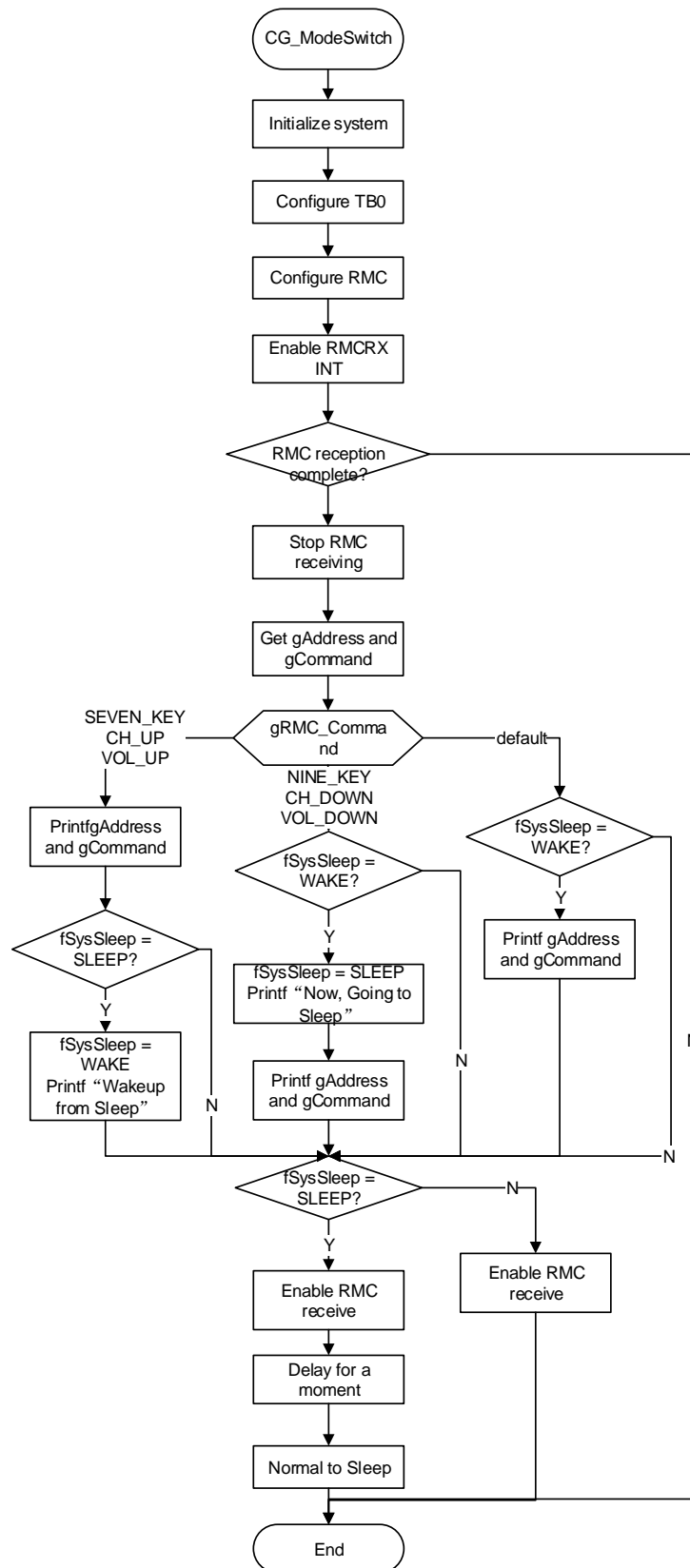
7-2-1 Example: Power mode change

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

The example includes:

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

Normal setup for CG (after reset)

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

Example code is as the following:

```
/* Set fgear = fc */
CG_SetFgearLevel(CG_DIVIDE_1);
/* Set fperiph to fgear */
CG_SetPhiT0Src(CG_PHIT0_SRC_FGEAR);
CG_SetPhiT0Level(CG_DIVIDE_1);
/* Set SCOUT source */
CG_SetSCOUTSrc(CG_SCOUT_SRC_PHIT0);
/* Set port M as HOSC */
CG_SetPortM(CG_PORTM_AS_HOSC);
/* Set fosc source */
CG_SetFoscSrc(CG_FOSC_OSC1);
/* Enable low-speed oscillator */
CG_SetFs(ENABLE);
/* Set low power consumption mode sleep */
CG_SetSTBYMode(CG_STBY_MODE_SLEEP);
/* Set high-speed and low-speed oscillator to be enabled after releasing stop mode */
*/
CG_SetExitStopModeFosc(ENABLE);
CG_SetExitStopModeFs(ENABLE);
/* Set pin status in stop mode to "active" */
CG_SetPinStateInStopMode(ENABLE);
/* Set up pll and wait 200us for pll to warm up , set fc source to fpll */
CG_EnableClkMulCircuit();
/* Set system clock to fgear */
CG_SetFsysSrc(CG_FSYS_SRC_FGEAR);
```

Setup to enter sleep mode

Set CG module: slow ->sleep mode.

First set TMRB and RMC, and enable RMC interrupt, and clear interrupt request.

```
CG_ClearINTReq(CG_INT_SRC_RMC_RX);
CG_SetSTBYReleaseINTSrc(CG_INT_SRC_RMC_RX,
                        CG_INT_ACTIVE_STATE_RISING, ENABLE);
NVIC_ClearPendingIRQ(INTRMCRX_IRQn);
NVIC_EnableIRQ(INTRMCRX_IRQn);
```

Prepare to enter sleep mode. First disable some modules except CPU, RTC, I/O and RMC if it has been enabled. Then set system clock to fs and set warm up time, wait for warm up is complete. Set RMC interrupt to wake up system. Finally use __WFI() instruction enter sleep mode.

```
/* Set CG module: Normal ->Sleep mode */
/* Add code here to disable modules (except CPU,RTC,I/O,CEC and RMC) if it has been enabled... */
/* Then set system clock to fs */
FunctionalState fs_st;
fs_st = CG_GetFsState();
if (fs_st == DISABLE) {
    CG_SetFs(ENABLE);
    CG_SetWarmUpTime(CG_WARM_UP_SRC_XT1, CG_WUODR_EXT);
    /* Start warm up */
```

```
CG_StartWarmUp();
/* Check whether warm up ends or not */
while (CG_GetWarmUpState() == BUSY);
} else {
    /* Do nothing */
}

CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC1, CG_WUODR_EXT);
/* Set RMC wake up system */
CG_SetSTBYReleaseINTSrc(CG_INT_SRC_RMC_RX,
CG_INT_ACTIVE_STATE_RISING, ENABLE);
/* Enter sleep mode */
__WFI();
```

Enable multiple clock circuit

Set PLL and set the source of fc. First enable PLL, then set warm up time and wait warm up time is complete. Finally set fPLL for fc source.

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

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

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

7-3 DMAC

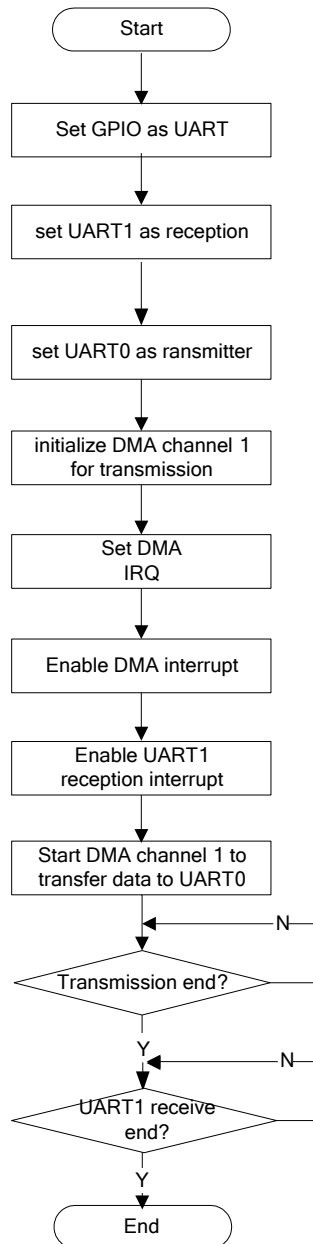
7-3-1 Example: Memory to peripheral

This is a simple example based on the TX03 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 TX03 Peripheral Driver (DMAC), which will transfer data from memory to UART0.

Firstly set UART0 as transmitter and enable it

```

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

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.TxPeriph = DMAC_SIO_0_RTX;
DMAC_InitStruct.TxINT = ENABLE;

DMAC_Init(myChannel, &DMAC_InitStruct);
```

Enable DMA IRQ

```
NVIC_ClearPendingIRQ(INTDMACTC_IRQn);
NVIC_EnableIRQ(INTDMACTC_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_SIO_0_RTX);
DMAC_SetDMACChannel(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_INTRReq == 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 */
}
```

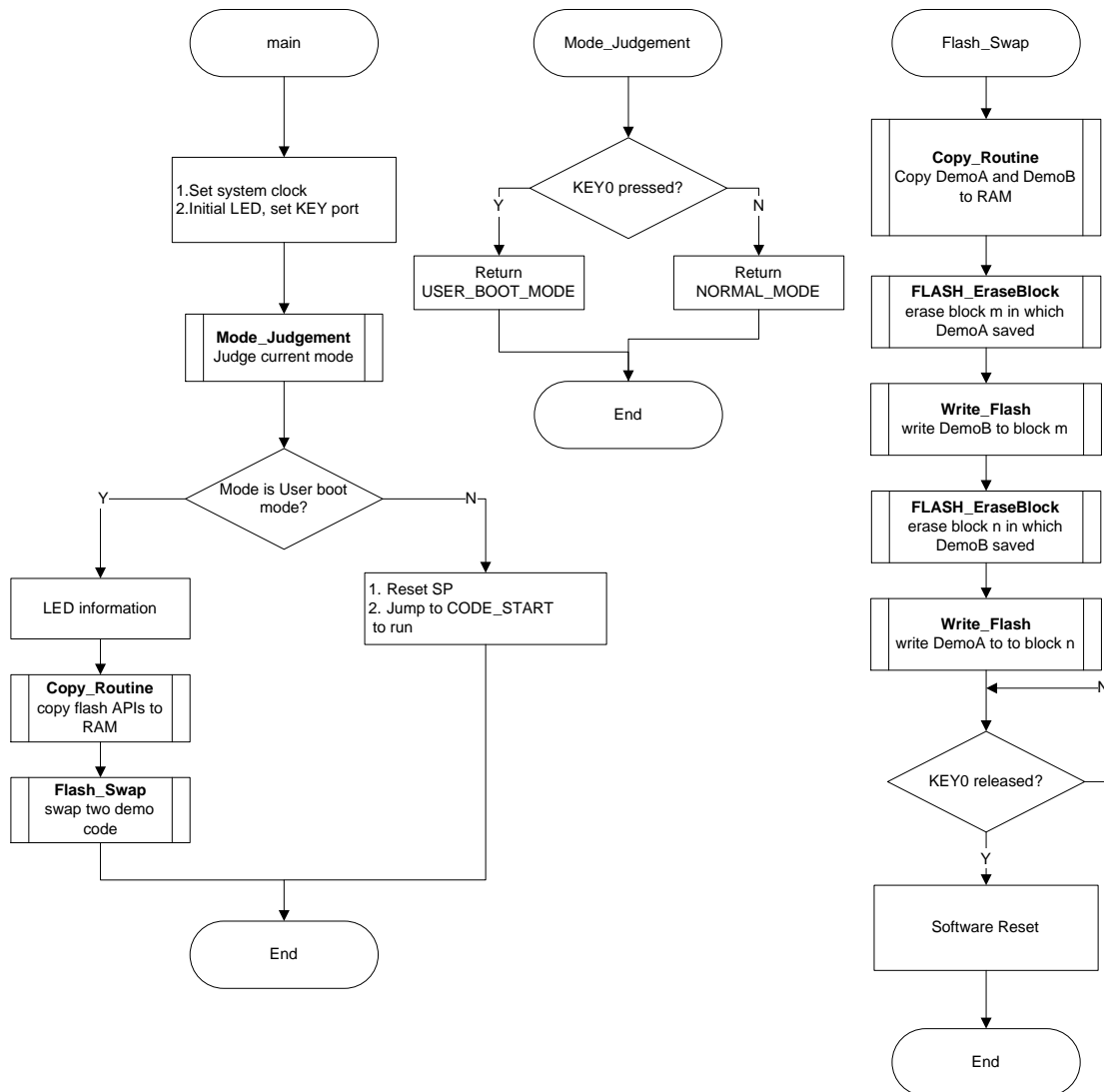
7-4 FLASH

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 configure system clock and initialize SWITCH port and LED. SWITCH port (GPIO) is used to judge current mode when reset and LED will display current operation information.

```

CG_SetPLL(DISABLE);          /* Disable PLL */
CG_SetFcSrc(CG_FC_SRC_FOSC); /* Select fosc */
GPIO_SetInput(GPIO_PD, GPIO_BIT_0); /* set port D to input */
SW_Init();
LEDInit();                   /* LED initialization */
  
```

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 SWITCH is high 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 (LED0,2 blink).

```
#if defined ( __CC_ARM ) /* RealView Compiler */
    ResetSP(); /* reset SP */
#elif defined ( __ICCARM__ ) /* IAR Compiler */
    asm("MOV R0, #0"); /* reset SP */
    asm("LDR SP, [r0]");
#endif

startup = CODE_START;
startup(); /* jump to code start address to run */
```

If the SWITCH is low when reset, the routine will enter user boot mode. LED0 light to indicate this mode. Then the routine will copy APIs for flash operation from address "FLASH_API_ROM" in Flash memory to address "FLASH_API_RAM" in RAM, because Flash memory cannot erase/program itself when runs the code at the same time. LED1 light to indicate RAM transferring.

```
LedOn(LED0);
delay(0xFFFFFU);
LedOff(LED0);
LedOn(LED1);
delay(0xFFFFFU);
Copy_Routine(FLASH_API_RAM, FLASH_API_ROM, SIZE_FLASH_API); /*
copy flash API to RAM */
```

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

In function Flash_Swap(), firstly it will copy Demo A and B from Flash memory to RAM, then call function FC_EraseBlock () and Write_Flash() to erase and program Flash memory. The code of Demo A and B will be exchanged in Flash memory. LED2 light to indicate swapping two demos.

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

LedOff(LED1);
LedOn(LED2);
delay(0xFFFFFU);

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;
    }

```

LED3 light to indicate the swap operation has completed. After SWITCH changed to high, it will execute software reset by using SCB->AIRCRCR register. Then this demo will run again to enter normal mode, because Demo A and B have been exchanged, the address "CODE_START" has become the start address of Demo B, Demo B will run (LED1,3 blink).

```

    LedOff(LED2);
    LedOn(LED3);
    delay(0xFFFFFU);

    while (SW_Get(SW0) == 1U) { /* wait for switch change to high */
    }

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

```

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

```

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

```

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

```

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

```

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

```

    while (BUSY == FC_GetBusyState()) {
        /* 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 FC_WritePage () to write data automatically in one page. The process of this function is basically same as FC_EraseBlock () except the automatic page program command.

```

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

```

```
        source++;
    }
```

7-5 GPIO

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

The example includes:

1. GPIO initialization
2. Write data to GPIO

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

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

```
uint8_t tmp;
GPIO_SetOutput(GPIO_PG, 0XF0U);
tmp = GPIO_ReadData(GPIO_PG);
tmp &= 0x0Fu;
GPIO_WriteData(GPIO_PG,tmp);
```

Then set GPIO to SW.

```
GPIO_SetPullDown(GPIO_PG, GPIO_BIT_0 | GPIO_BIT_1 | GPIO_BIT_2 |
GPIO_BIT_3, ENABLE);
GPIO_SetInput(GPIO_PG, GPIO_BIT_0 | GPIO_BIT_1 | GPIO_BIT_2 |
GPIO_BIT_3);
```

In the While process, do the LED demo: LED on and LED off, and display Second%10U.

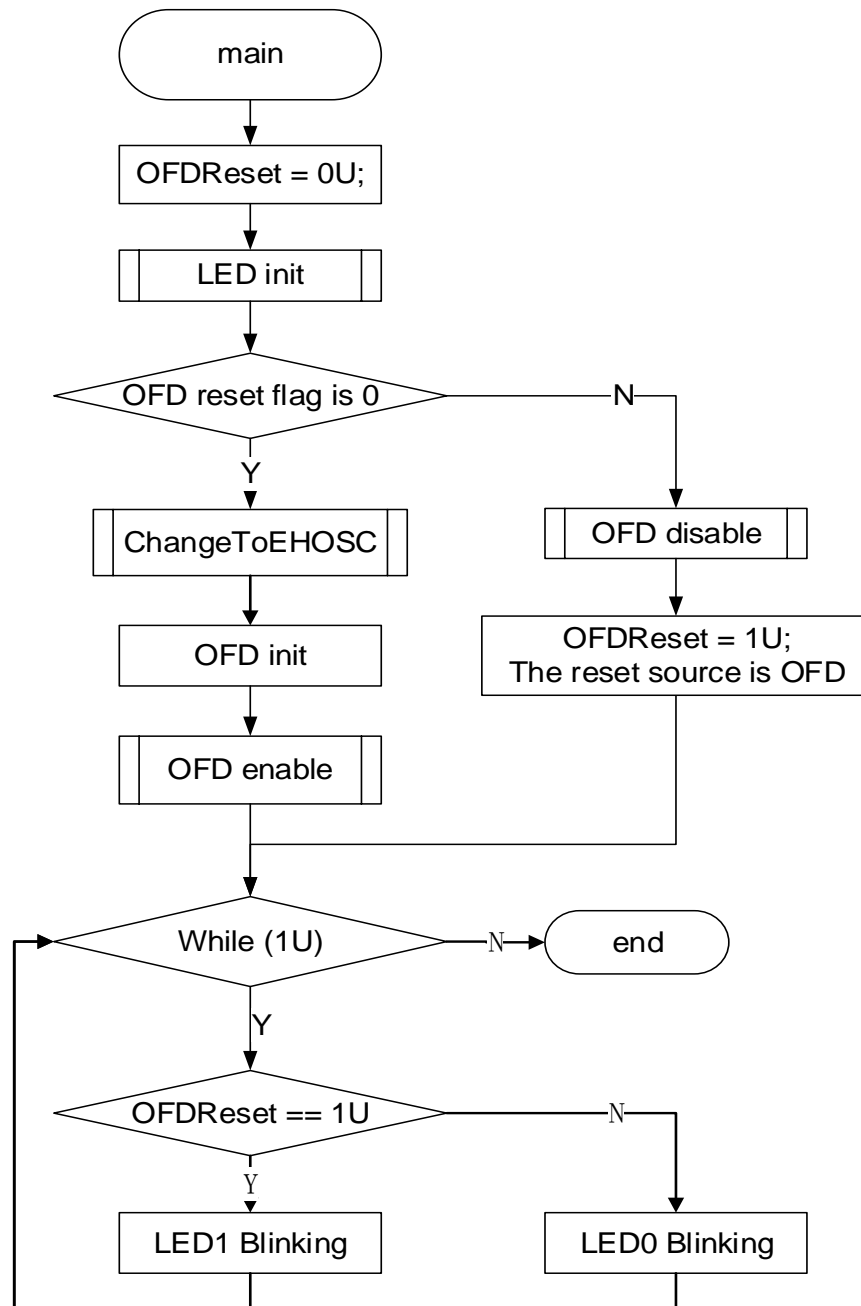
7-6 OFD

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

The example includes:

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

- Flowchart



- **Code and Explanation for the Example**

At first, initialize LED.

```
LED_Init ();
```

Check the OFD reset flag.

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

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

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

```
void ChangeToEHOSC(void)  
{  
    /* Use the external oscillation */  
    TSB_CG->OSCCR &= 0x000FFFFFUL;  
    TSB_CG->OSCCR |= 0xFFFF0000UL; /*Set the warm up time WUODR[13:2] */  
    TSB_CG->OSCCR_HOSCON = 1U; /*Set port as X1/X2 for external oscillator */  
    TSB_CG->OSCCR_XEN1 = 1U; /*Enable external oscillator */  
    TSB_CG->OSCCR_WUPSEL2 = 1U;  
    TSB_CG->OSCCR_WUPSEL1 = 0U; /*Select warm-up clock */  
    TSB_CG->OSCCR_WUEON = 1U; /*Start warm up */  
    while (TSB_CG->OSCCR_WUEF) {} /* Warm-up */  
    TSB_CG->OSCCR_OSCSEL = 1U; /* Use the external oscillation */  
    TSB_CG->OSCCR_XEN2 = 1U; /* Enable internal oscillator for OFD */  
}
```

To configure 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;
```

LED0 and LED1 indicate the system clock status as below.

LED status	Meaning
LED0 blinking	MCU run normally with external oscillator works OK, OFD function is enabled and ready to detect oscillator abnormal status
LED1 blinking	External oscillator failure, causes the OFD reset, MCU use the default setting to run under inner oscillator. OFD function is disabled.

```
while (1U) {
    if (OFDReset == 1U) {
        LED_On(LED1);
        Delay();
        LED_Off(LED1);
        Delay();
    } else {
        LED_On(LED0);
        Delay();
        LED_Off(LED0);
        Delay();
    }
}
```

7-7 RMC

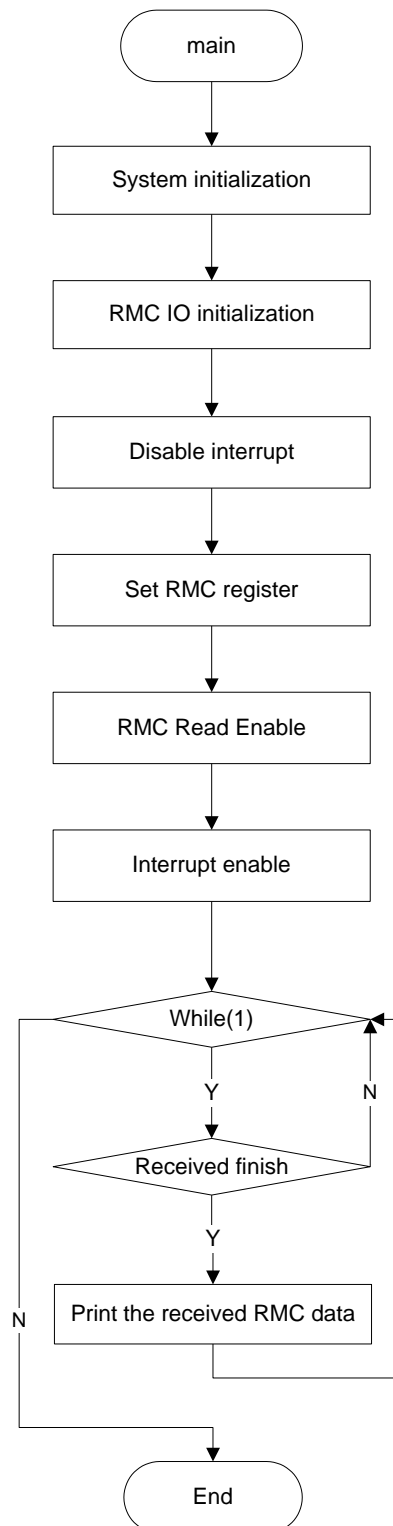
7-7-1 Example: RMC receiving

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



- 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 0.

```
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_RMC0);
```

Get the leader detection result for the specified RMC0 channel.

```
RMC_LeaderDetection myRMC_LeaderDetection;
myRMC_LeaderDetection = RMC_GetLeader(TSB_RMC0);
```

Get the received data from the specified RMC0 channel.

```
RMC_RxDataTypeDef myRMC_RxDataDef;
myRMC_RxDataDef = RMC_GetRxData(TSB_RMC0);
```

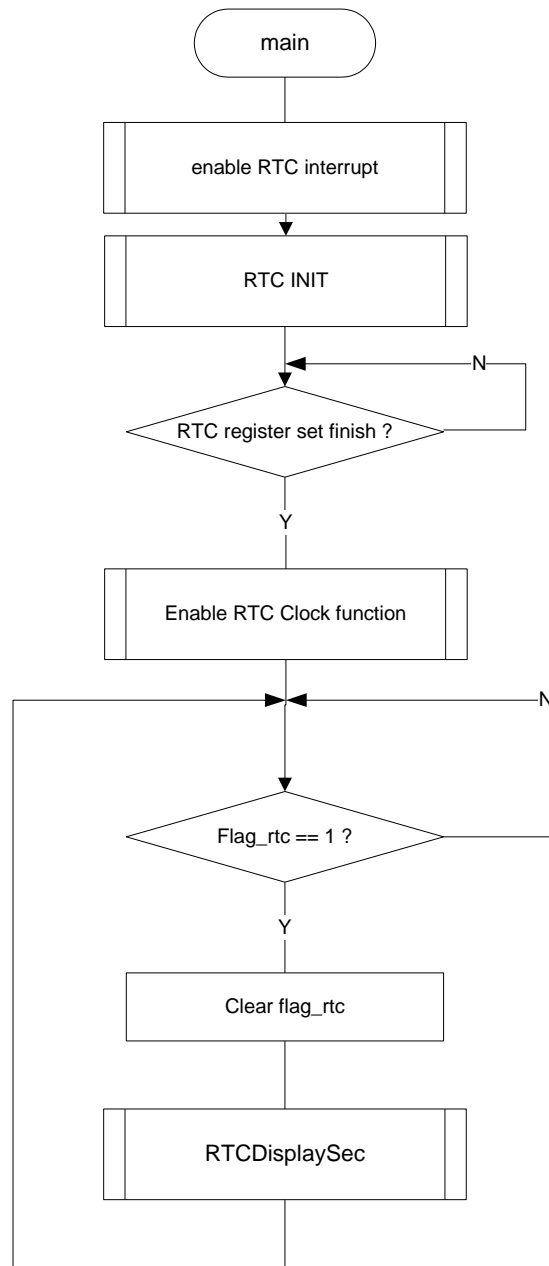
7-8 RTC

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: 2019/6/21 14:59:00, 24hours format.

```
Date_Struct.LeapYear = RTC_LEAP_YEAR_3;
Date_Struct.Year = (uint8_t) 19U;
Date_Struct.Month = (uint8_t) 6U;
```

```
Date_Struct.Date = (uint8_t) 21U;
Date_Struct.Day = RTC_FRI;

Time_Struct.HourMode = RTC_24_HOUR_MODE;
Time_Struct.Hour = (uint8_t) 14U;
Time_Struct.Min = (uint8_t) 59U;
Time_Struct.Sec = (uint8_t) 0U;
```

Disable clock and alarm function.

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

Reset RTC sec counter, enable 1Hz interrupt, enable RTCINT.

```
RTC_ResetClockSec();
RTC_SetAlarmOutput(RTC_PULSE_1_HZ);
RTC_SetRTCINT(ENABLE);
```

Set RTC time and date value

```
RTC_SetTimeValue(&Time_Struct);
RTC_SetDateValue(&Date_Struct);
```

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

```
NVIC_EnableIRQ(INTRTC_IRQn);

/* 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;
CG_ClearINTReq(CG_INT_SRC_RTC);
```

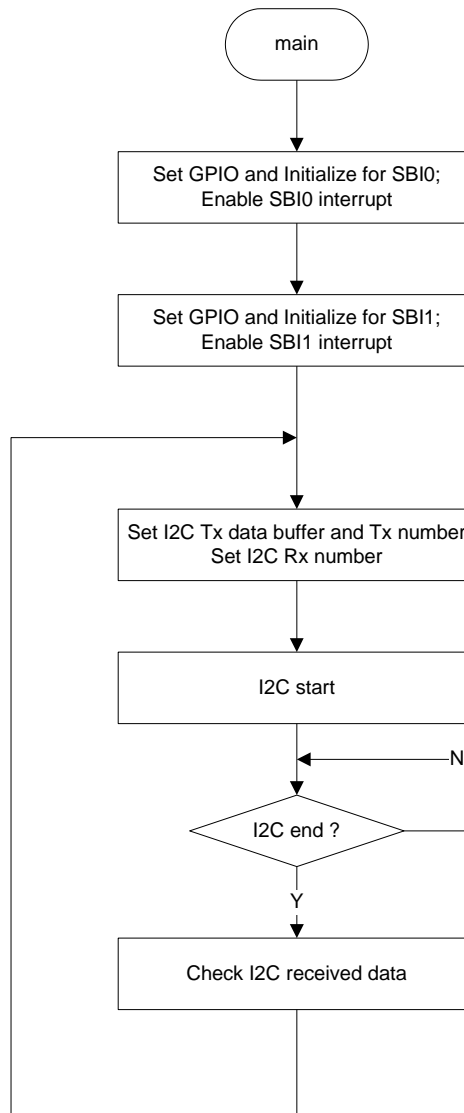
7-9 SBI

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

The example includes:

1. SBI configuration and I2C initialization
2. I2C master send data process
3. I2C slave receive data process

- Flow Chart



- Code and Explanation for the Example

At first, configure GPIO for SBI0 and SBI1 I2C mode.

```
SBI0_IO_Configuration();
SBI1_IO_Configuration();
```

Initialize and configure SBI0 channel and enable INTSBI0.

```
myI2C.I2CSelfAddr = SELF_ADDR;
myI2C.I2CDataLen = SBI_I2C_DATA_LEN_8;
myI2C.I2CACKState = ENABLE;
myI2C.I2CClkDiv = SBI_I2C_CLK_DIV_584;
SBI_Enable(TSB_SBI0);
SBI_SWReset(TSB_SBI0);
SBI_InitI2C(TSB_SBI0, &myI2C);
NVIC_EnableIRQ(INTSBI0_IRQn);
```

Then enable, initialize and configure SBI1 channel and enable INTSBI1.

```
myI2C.I2CSelfAddr = SLAVE_ADDR;
myI2C.I2CDataLen = SBI_I2C_DATA_LEN_8;
```

```
myI2C.I2CACKState = ENABLE;
myI2C.I2CClkDiv = SBI_I2C_CLK_DIV_584;
SBI_Enable(TSB_SBI1);
SBI_SWReset(TSB_SBI1);
SBI_InitI2C(TSB_SBI1, &myI2C);
NVIC_EnableIRQ(INTSBI1_IRQn);
```

After the above setting, start I2C read.

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:
    gl2CTxDatLen = 7U;
    gl2CTxDat[0] = gl2CTxDatLen;
    gl2CTxDat[1] = 'T';
    gl2CTxDat[2] = 'O';
    gl2CTxDat[3] = 'S';
    gl2CTxDat[4] = 'H';
    gl2CTxDat[5] = 'I';
    gl2CTxDat[6] = 'B';
    gl2CTxDat[7] = 'A';
    gl2CWCnt = 0U;
    for (glCnt = 0U; glCnt < 8U; glCnt++) {
        gl2CRxDat[glCnt] = 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_SBI2) is used 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 INTSBI2.

The data receive is handled in INTSBI0.

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

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

    SBIx = TSB_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 (gl2CWCnt <= gl2CTxDataLen) {
            SBI_SetSendData(SBly, gl2CTxData[gl2CWCnt]);
                                                    /* Send next data */
            gl2CWCnt++;
        } else { /* I2C data send finished. */
            SBI_GenerateI2CStop(SBly); /* Stop I2C */
        }
    }
} else { /* Rx Mode */
    /* Do nothing */
}
} else { /* Slave mode */
    /* Do nothing */
}
}
}

```

In INTSBI1 function, I2C slave receive process is handled according to I2C bus state, SBI_GetReceiveData() is used to read received data in SBI buffer, I2C stop condition is controlled by master.

```

void INTSBI1_IRQHandler(void)
{
    uint32_t tmp = 0U;
    TSB_SBI_TypeDef *SBly;
    SBI_I2CState sbi1_sr;

    SBly = TSB_SBI1;
    sbi1_sr = SBI_GetI2CState(SBly);

    if (!sbi1_sr.Bit.MasterSlave) { /* Slave mode */
        if (!sbi1_sr.Bit.TRx) { /* Rx Mode */
            if (sbi1_sr.Bit.SlaveAddrMatch) {
                /* First read is dummy read for Slave address recognize */
                tmp = SBI_GetReceiveData(SBly);
                gl2CRCnt = 0U;
            } else {
                /* Read I2C received data and save to I2C_RxData buffer */
                tmp = SBI_GetReceiveData(SBly);
                gl2CRxData[gl2CRCnt] = tmp;
                gl2CRCnt++;
            }
        } else { /* Tx Mode */
            /* Do nothing */
        }
    } else { /* Master mode */
        /* Do nothing */
    }
}

```

7-10 SIO/UART

7-10-1 Example: Retarget

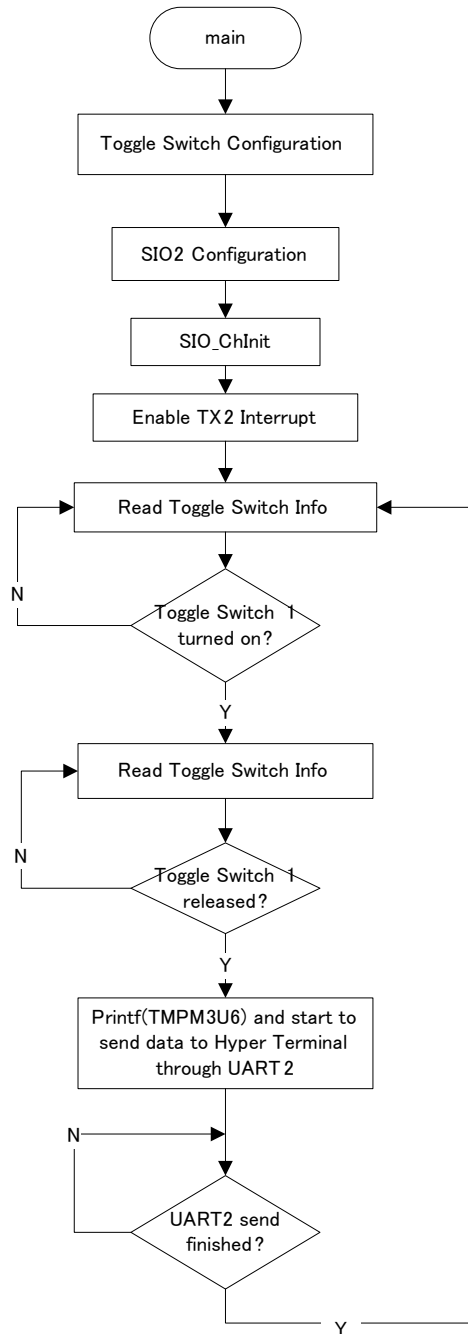
This is a simple example based on the TX03 Peripheral Driver (UART, GPIO).

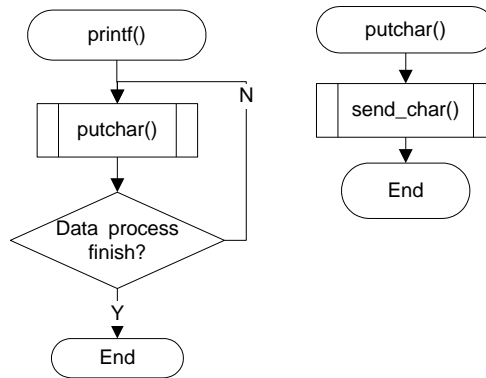
The example includes:

1. UART configuration and initialization.

2. UART TX process.
3. Use UART2 TX interrupt to send data.
4. Retarget printf() to UART2.

• Flowchart





• Code and Explanation for the Example

At first, Configure SW and SIO0, then Initialize UART0.

Use GPIO peripheral drivers configure GPIO for SW.

```
GPIO_SetPullDown(GPIO_PG, GPIO_BIT_0 | GPIO_BIT_1 | GPIO_BIT_2 |
GPIO_BIT_3, ENABLE);
GPIO_SetInput(GPIO_PG, GPIO_BIT_0 | GPIO_BIT_1 | GPIO_BIT_2 | GPIO_BIT_3);
```

Use GPIO peripheral drivers configure GPIO for UART0.

```
/* Set PE0 as SC0TXD */
GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_0, ENABLE);
GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_0);
GPIO_SetPullUp(GPIO_PE, GPIO_BIT_0, ENABLE);
/* Set PE1 as SC0RXD */
GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_1, ENABLE);
GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_1);
GPIO_SetPullUp(GPIO_PE, GPIO_BIT_1, ENABLE);
```

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

```
UART_InitTypeDef myUART;

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

Use UART peripheral drivers enable and initialize UART0.

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

After above setting, enable UART0 TX interrupt.

```
NVIC_EnableIRQ(RETARGET_INT);
```

Then Judge whether Switch 1 has been turned off, if Toggle Switch is turned on, then wait for Switch 0 released:

```
while (wait_SW0) {
    /* Read switch SW0 input */
    SW_info = SW_Get(SW_PORT);
    if (SW_info == 1U) {
        /* wait for TSW1 released */
        do {
            wait_SW0 = 0U;
            SW_info = SW_Get(SW_PORT);
        } while (1);
    }
}
```



```

        } while (SW_info != SWRELEASE);
    } else {
        /* do nothing */
    }
}

```

After Toggle Switch 1 has been released, start to send data by printf() through UART2.

```
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 UART2.

```

#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)
{
    uint8_t tmp = 0U;

    do {
        tmp = gSIOWrIndex;
    } while (gSIORDIndex != tmp);    /* wait for finishing sending */

    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(RETARGET_INT);
    } else {
        /* do nothing */
    }
    return ch;
}

```

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

UART0 TX interrupt routine:

```

void INTTX0_IRQHandler(void)
{
    uint8_t tmp = 0U;

    tmp = gSIOWrIndex;
    if (gSIORDIndex < tmp) {    /* Buffer is not empty */
        UART_SetTxData(UART_RETARGET,
            (uint32_t) gSIOTxBuffer[gSIORDIndex++]);    /* Send data */
        fSIO_INT = SET;        /* SIO INT is enable */
    } else {
        /* Disable SIO0 INT */
        fSIO_INT = CLEAR;
        NVIC_DisableIRQ(RETARGET_INT);
    }
}

```

```

    fSIOTxOK = YES;
}
if (gSIORdIndex >= tmp) { /* Reset buffer index */
    gSIOWrIndex = CLEAR;
    gSIORdIndex = CLEAR;
} else {
    /* Do nothing */
}
}

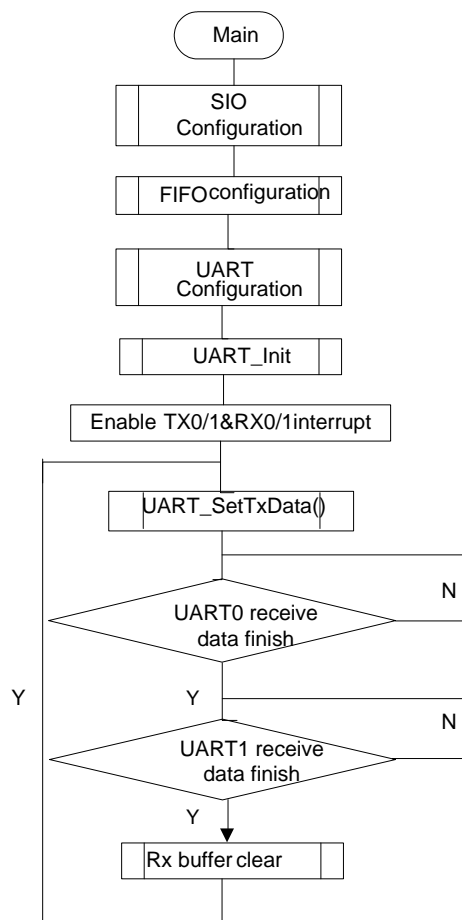
```

7-10-2 Example: UART FIFO

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

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

- Flowchart**



- Code and Explanation for the Example**

At first configure the GPIO and initialize UART.

Use GPIO peripheral drivers configure GPIO for UART1 and UART2.

```
SIO_Configuration(UART1);
SIO_Configuration(UART2);
```

Note) The function SIO_Configuration does not support UART3 / UART4.

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

```
UART_InitTypeDef myUART;

/* configure SIO0 for reception */
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;
```

Use UART peripheral drivers to enable and initialize UART1/2 channels.

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

UART_Enable(UART2);
UART_Init(UART2, &myUART);
```

FIFO configuration.

```
UART_RxFIFOByteSel(UART1, UART_RXFIFO_RXFLEVEL);
UART_RxFIFOByteSel(UART2, UART_RXFIFO_RXFLEVEL);

UART_TxFIFOINTCtrl(UART1, ENABLE);
UART_TxFIFOINTCtrl(UART2, ENABLE);

UART_RxFIFOINTCtrl(UART1, ENABLE);
UART_RxFIFOINTCtrl(UART2, ENABLE);

UART_TRxAutoDisable(UART1, UART_RXTXCNT_AUTODISABLE);
UART_TRxAutoDisable(UART2, UART_RXTXCNT_AUTODISABLE);

UART_FIFOConfig(UART1, ENABLE);
UART_FIFOConfig(UART2, ENABLE);

UART_RxFIFOFillLevel(UART1, UART_RXFIFO4B_FLEVLE_4_2B);
UART_RxFIFOFillLevel(UART2, UART_RXFIFO4B_FLEVLE_4_2B);

UART_RxFIFOINTSel(UART1, UART_RFIS_REACH_EXCEED_FLEVEL);
UART_RxFIFOINTSel(UART2, UART_RFIS_REACH_EXCEED_FLEVEL);

UART_RxFIFOClear(UART1);
UART_RxFIFOClear(UART2);

UART_TxFIFOFillLevel(UART1, UART_TXFIFO4B_FLEVLE_0_0B);
UART_TxFIFOFillLevel(UART2, UART_TXFIFO4B_FLEVLE_0_0B);

UART_TxFIFOINTSel(UART1, UART_TFIS_REACH_NOREACH_FLEVEL);
UART_TxFIFOINTSel(UART2, UART_TFIS_REACH_NOREACH_FLEVEL);

UART_TxFIFOClear(UART1);
UART_TxFIFOClear(UART2);
```

After above setting, enable UART1/2 interrupt.

```
NVIC_EnableIRQ(INTTX1_IRQn);
```

```
NVIC_EnableIRQ(INTRX2_IRQn);  
  
NVIC_EnableIRQ(INTTX2_IRQn);  
NVIC_EnableIRQ(INTRX1_IRQn);
```

The rest process of data flow is finished in ISR of UART1/2 RX and TX interrupt routine
UART1 TX interrupt routine:

```
void INTTX1_IRQHandler(void)  
{  
    volatile UART_Err err;  
  
    if (TxCounter < NumToBeTx) {  
        UART_SetTxData(UART1, TxBuffer[TxCounter++]);  
    } else {  
        err = UART_GetErrState(UART1);  
    }  
}
```

UART2 TX interrupt routine:

```
void INTTX2_IRQHandler(void)  
{  
    volatile UART_Err err;  
  
    if (TxCounter1 < NumToBeTx1) {  
        UART_SetTxData(UART2, TxBuffer1[TxCounter1++]);  
    } else {  
        err = UART_GetErrState(UART2);  
    }  
}
```

UART1 RX interrupt routine:

```
void INTRX1_IRQHandler(void)  
{  
    volatile UART_Err err;  
  
    err = UART_GetErrState(UART1);  
    if (UART_NO_ERR == err) {  
        RxBuffer[RxCounter++] = (uint8_t) UART_GetRxData(UART1);  
    } else {  
        /* do nothing */  
    }  
}
```

UART2 RX interrupt routine:

```
void INTRX2_IRQHandler(void)  
{  
    volatile UART_Err err;  
  
    err = UART_GetErrState(UART2);  
    if (UART_NO_ERR == err) {  
        RxBuffer1[RxCounter1++] = (uint8_t) UART_GetRxData(UART2);  
    } else {  
        /* do nothing */  
    }  
}
```

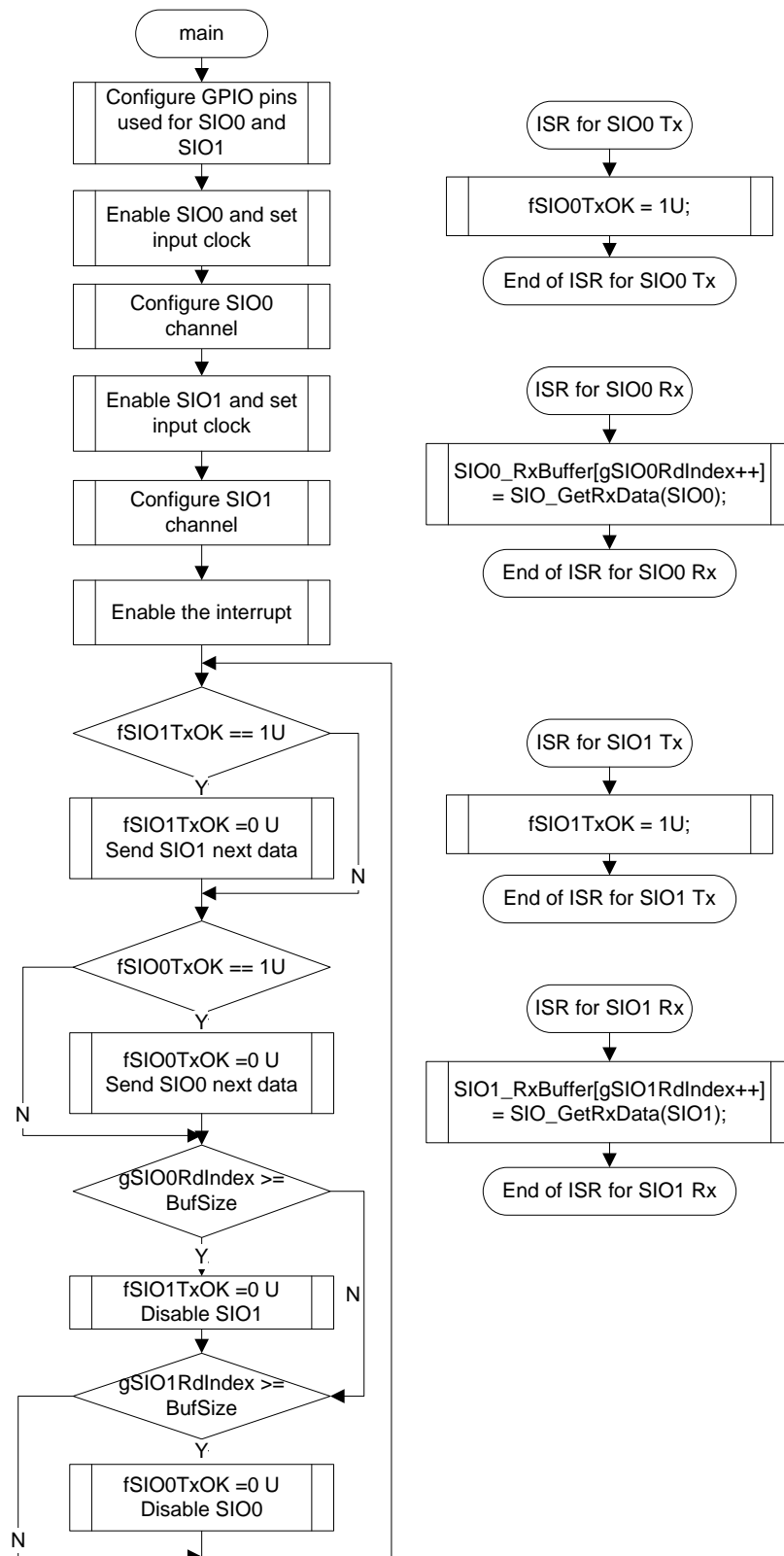
7-10-3 Example: SIO

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

The example includes:

1. Basic setup operation of SIO
2. The data transfer between SIO0 and SIO1
3. SIO interrupt of Tx and Rx

- Flowchart



- **Code and Explanation for the Example**

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

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

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

/*initialize the SIO0 struct */
SIO0_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO0_Init.IntervalTime = SIO_SINT_TIME_SCLK_8;
SIO0_Init.TransferMode = SIO_TRANSFER_FULDPX;
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_FULDPX;
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);
}

```

7-11 SSP

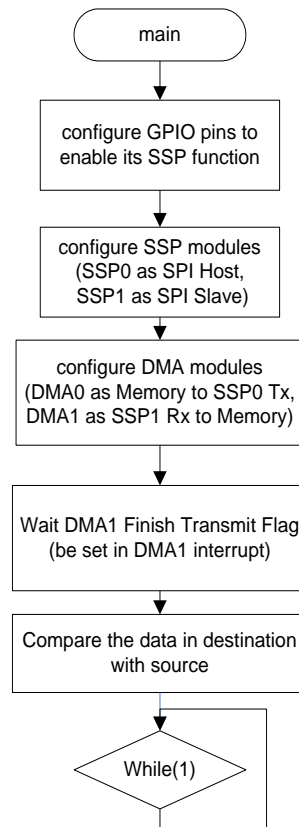
7-11-1 Example: SSP0 to SSP1 via DMAC

This is a simple example based on the TX03 Peripheral Driver (SSP, DMAC, GPIO). It will send some data from SSP0 to SSP1 via DMA.

The example includes:

1. Configuration of SSP0 as SPI Host, and SSP1 as SPI Slave
2. Configuration of DMA0 as Memory to SSP0 Tx, and DMA1 as SSP1 Rx to Memory
3. Check the data transmit flow: Memory → SSP0 Tx → SSP1 Rx → Memory

- Flowchart



- Code and Explanation for the Example

Below is the main.c for the flowchart above.

```

int main(void)
{
    GPIO_SetSSP();
    initSSP();
    InitDMA();

    /* Wait the end of transmission */
    while (TxEndFlag != DONE) {
        /* Do nothing */
    }

    /* now DMA is finished, Set a Break Point here, */
    /* after function Buffercompare() is called, result == SAME */
    result = Buffercompare( SRC_Buffer, DST_Buffer, BUFFER_SIZE);

    while(1) {
        /* do nothing */
    }
}

```

For detail of functions: GPIO_SetSSP(), initSSP(), InitDMA() above, please refer to the comments in the code file.

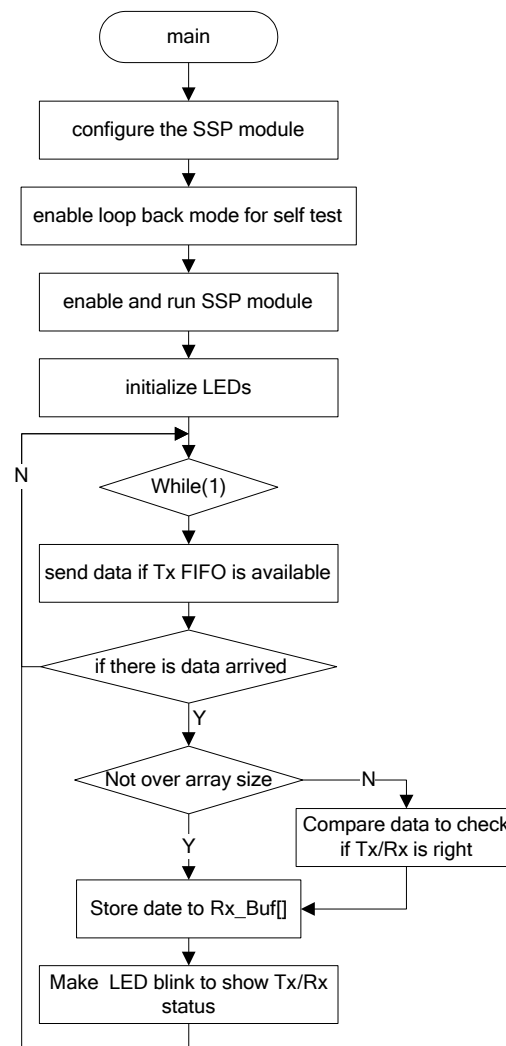
7-11-2 Example: SSP0 Self Loop Back

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

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

```

int main(void)
{
    SSP_InitTypeDef initSSP;
    SSP_FIFOState fifoState;

    uint16_t datTx = 0U;          /* must use 16bit type */
    uint32_t cntTx = 0U;

```

```

uint32_t cntRx = 0U;

uint16_t receive = 0U;
uint16_t Rx_Buf[MAX_BUFSIZE] = { 0U };
uint16_t Tx_Buf[MAX_BUFSIZE] = { 0U };

/* configure the SSP module */
initSSP.FrameFormat = SSP_FORMAT_SPI;

/* 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 for self test */
SSP_SetLoopBackMode(TSB_SSP0, ENABLE);

/* enable and run SSP module */
SSP_Enable(TSB_SSP0);

/* UART & LED initialization */
LEDInit();
hardware_init(UART_RETARGET);

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 maxium */
/* 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);
    LedOn(LED0);
    LedOn(LED1);
} else {
    LedOn(LED2);
    LedOn(LED3);
}
}

} else {
    /* do nothing */
}
printf("SSP RX DATA: %d\r\n", receive);
}
}

```

7-12 TMRB

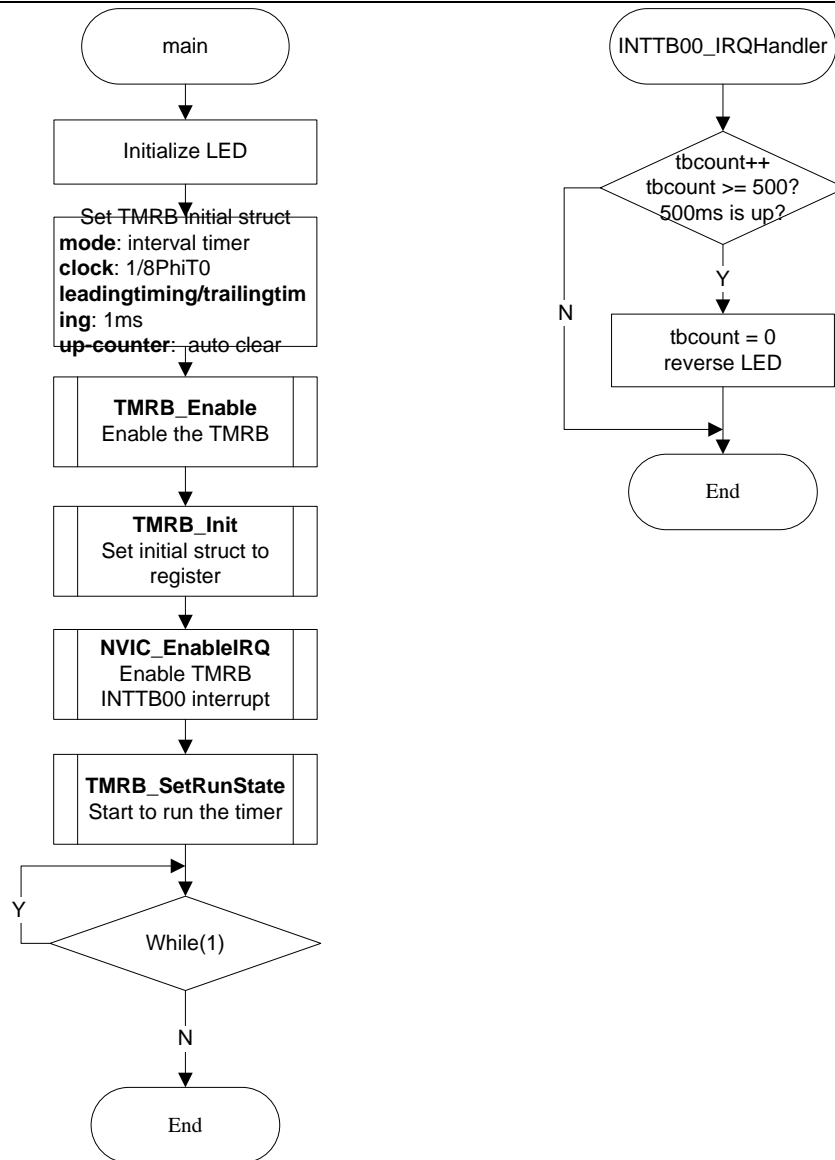
7-12-1 Example: General Timer

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.

```

LEDInit();          /* LED initialize */
LedOn(LED1);        /* Turn on LED1 */
  
```

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 0x1388, because $f_{\phi T0} = f_{sys} = f_c = 10\text{MHz} \times \text{PLL} = 40\text{MHz}$, $f_{tmrb} = 1/8 f_{\phi T0} = 5\text{MHz}$, $T_{tmrb} = 0.2\mu\text{s}$, $1\text{ms}/0.2\mu\text{s} = 5000 = 0x1388$ (Please see CG part for more detail information about the clock setting)

```

TMRB_InitTypeDef m_tmrb;

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

Enable the TMRB module and set the initialization structure to specified registers. Enable INTTB00 interrupt, this interrupt will be triggered every 1ms, in the end, start the TMRB to run.

```
TMRB_Enable(TSB_TB0);           /* enable the TMRB0 */
TMRB_Init(TSB_TB0, &m_tmr);     /* initial the TMRB0 */
NVIC_EnableIRQ(INTTB00_IRQn);   /* enable INTTB00 interrupt */
TMRB_SetRunState(TSB_TB0, TMRB_RUN); /* run TMRB0*/
```

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

```
tbcount++;
if (tbcount >= 500U) {           /* 500ms is up */
    tbcount = 0U;
    /* reverse LED output */
    ledon = (ledon == 0U) ? 1U : 0U;
    if (0U == ledon) {
        LedOff(LED1);
    } else {
        LedOn(LED1);
    }
} else {
    /* do nothing */
}
```

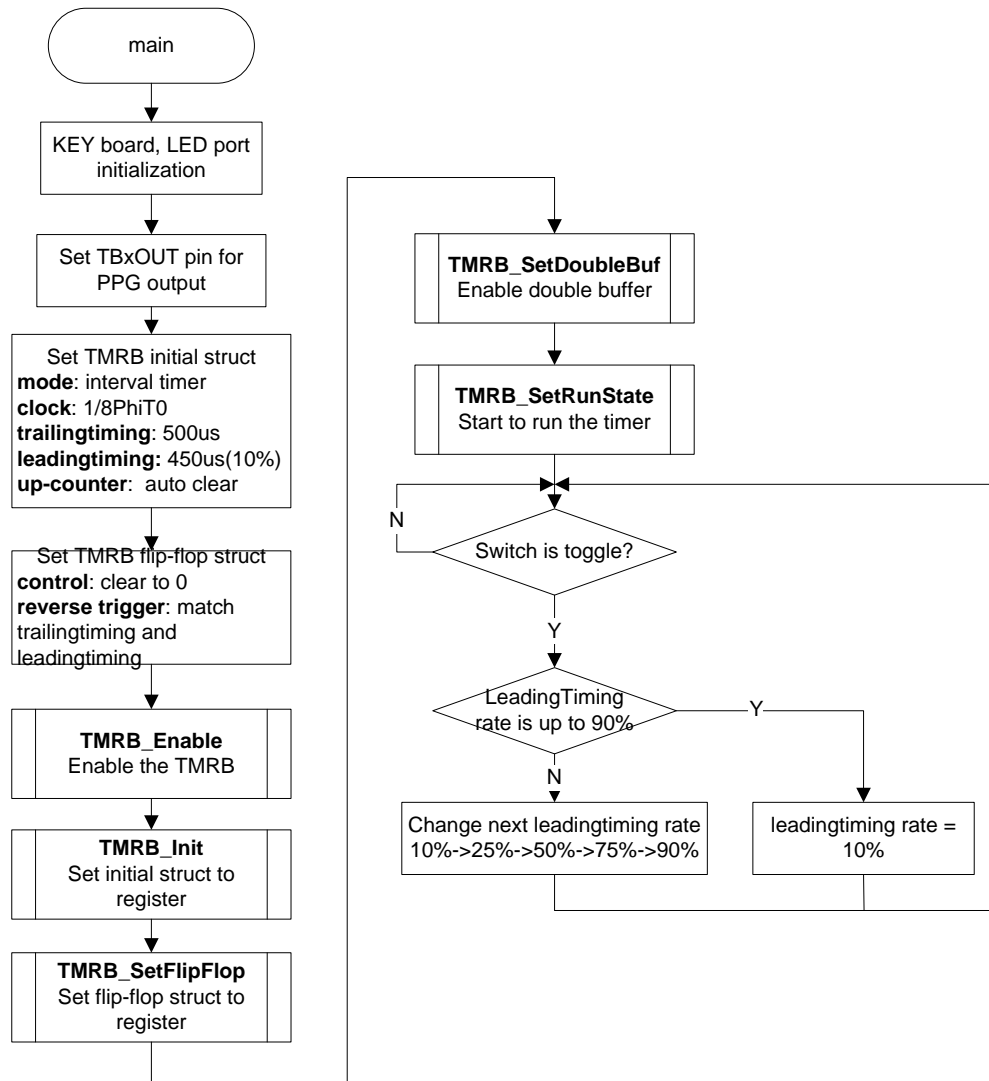
7-12-2 Example: PPG Output

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

The example includes:

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

- **Flow Chart**



• Code and Explanation for the Example

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

```

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

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

Prepare TMRB initialization structure, and then fill TMRB mode, clock, up-counter clear method, trailingtiming and leadingtiming into it. This demo will set trailingtiming to 500us, macro TMRB7TIME equals 0x09C4, because $f_{\text{phiT0}} = f_{\text{sys}} = f_c = 10\text{MHz} \times \text{PLL} = 40\text{MHz}$, $f_{\text{tmrb}} = 1/8 f_{\text{phiT0}} = 5\text{MHz}$, $T_{\text{tmrb}} = 0.2\mu\text{s}$, $500\mu\text{s}/0.2\mu\text{s} = 2500 = 0x09C4$ (Please see CG part for more detail information about the clock setting)

```

TMRB_InitTypeDef m_tmrb;

m_tmrb.Mode = TMRB_INTERVAL_TIMER;
m_tmrb.ClkDiv = TMRB_CLK_DIV_8;
m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;
  
```

```
m_tmrb.TrailingTiming = TMRB7TIME; /* T = 500us */
m_tmrb.LeadTiming = LeadingTiming[Rate];
```

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_TB7);
TMRB_Init(TSB_TB7, &m_tmrb);
TMRB_SetFlipFlop(TSB_TB7, &PPGFFInital);
/* enable double buffer */
TMRB_SetDoubleBuf(TSB_TB7,ENABLE, TMRB_WRITE_REG_SEPARATE);
TMRB_SetRunState(TSB_TB7, TMRB_RUN);
```

Wait switch from low to high, and at the same time, display current leadingtiming on display array.

```
do { /* wait if switch is Low */
    keyvalue = SW_Get(SW1);
    LeadingTiming_display(leadingtiming_num[0]);
} while (GPIO_BIT_VALUE_1 == keyvalue);
```

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

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

TMRB_ChangeLeadingTiming(TSB_TB7, LeadingTiming[Rate]); /* change
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/Tmrb = 0x271

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

That is the calculation method of leadingtiming array

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

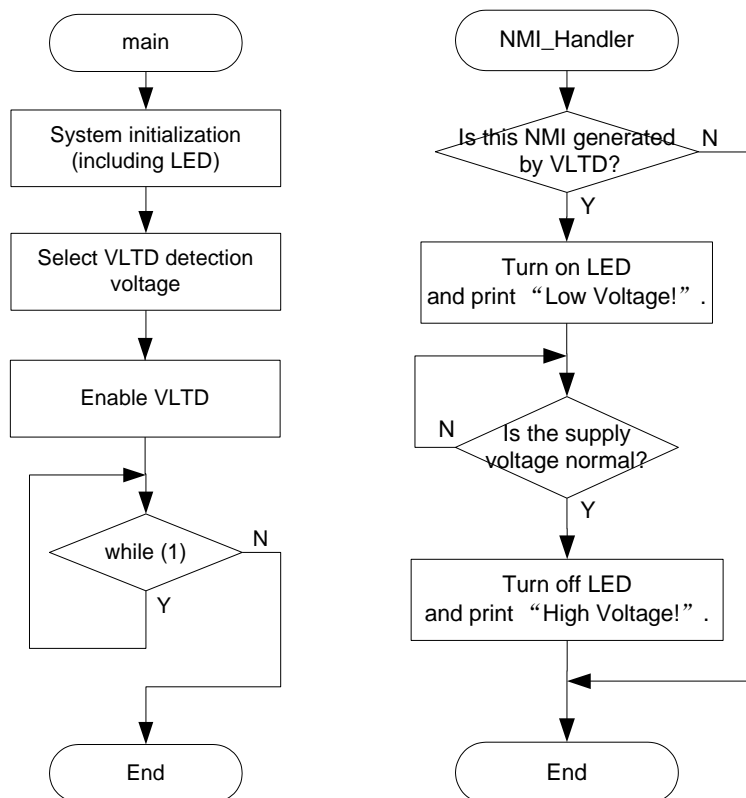
7-13 VLTD

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

The example includes:

1. VLTD configuration
2. VLTD status monitoring

• Flowchart



- **Code and Explanation for the Example**

Main program : [main.c]

In void main(void):

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

```
WDT_Disable ();
LEDInit();
```

Set detection voltage and enable VLTD.

```
VLTD_SetVoltage(VLTD_DETECT_VOLTAGE_46);
VLTD_Enable ();
```

Then a usual while(1) loop.

NMI handler : [tmpm3u6_vltd_int.c]

In void NMI_Handler(void):

Make sure the NMI is generated by VLTD.

```
if (CG_GetNMIFlag().Bit.VoltageDetection == 1U) {
```

Turn on LED.

```
LedOn(LED1);
```

Wait for the power supply voltage to be normal.

```
/* Wait for the power supply voltage to be normal. */
while (VLTD_GetStatus() == 1U) {
    __NOP();
}
```

Turn off LED.

```
LedOff(LED1);
```

At last the NMI_Handler() function returns.

7-14 WDT

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

The example includes:

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

- **Code and Explanation for the Example**

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

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

Initialize WDT and enable it.

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

In DEMO1 Waiting for the NMI interrupt flag.

```
while(1)
```

```
{  
}
```

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

```
if (fIntNMI == 1U) {  
    fIntNMI = 0U;  
    LedOn(LED1);  
    Delay(500U);  
    LedOff(LED1);  
    Delay(500U);  
    WDT_Disable();  
} else {  
    /* Do nothing */  
}
```

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

```
#ifdef DEMO2  
    Delay(500U);  
    WDT_WriteClearCode();  
    LedOn(LED0);  
    Delay(500U);  
    LedOff(LED0);  
#endif
```

7-15 ENC

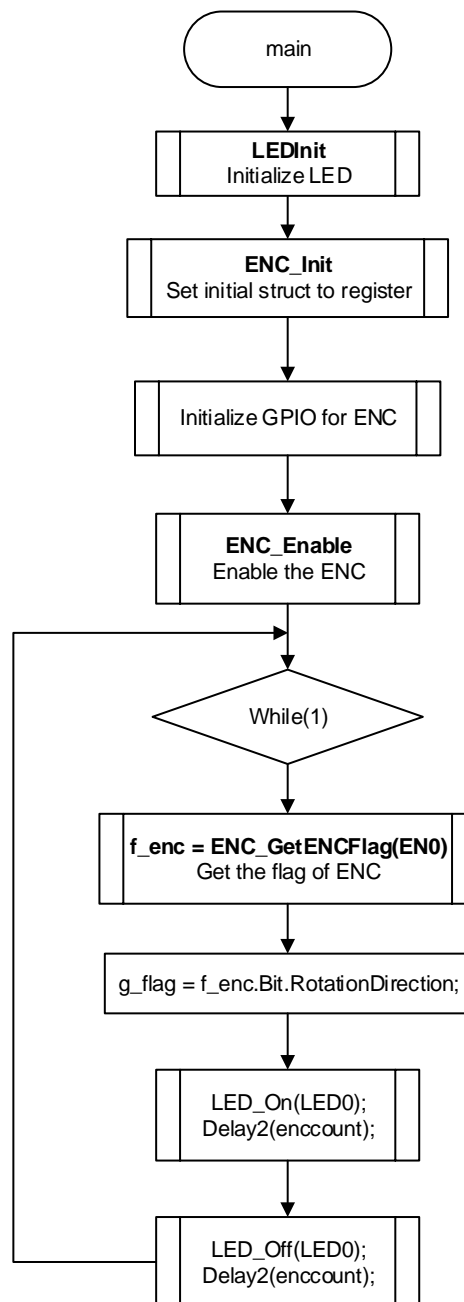
7-15-1 Example: Rolling Detection

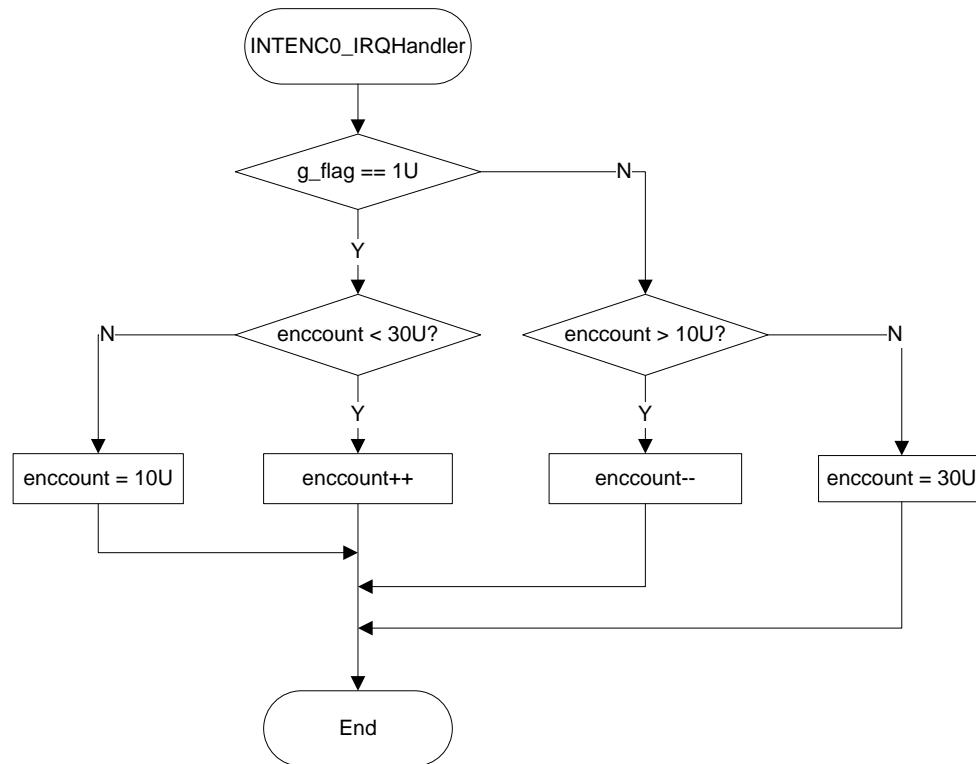
This is a simple example based on the TX03 Peripheral Driver (ENC, GPIO).

The example includes:

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

- Flowchart





• Code and Explanation for the Example

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

At first, initialize LED channel on eval board.

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

Prepare ENC initialization structure, and initialize the ENC0.

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

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

Initialize the GPIO for ENC0. Set the PD0 and PD1 as inputs of ENC0.

```
/* GPIO initialization */
GPIO_SetInputEnableReg(GPIO_PD, GPIO_BIT_0, ENABLE); /*Set PD0 as
input */
GPIO_EnableFuncReg(GPIO_PD, GPIO_FUNC_REG_1, GPIO_BIT_0); /*Set
PD0 as ENCA0 */
GPIO_SetInputEnableReg(GPIO_PD, GPIO_BIT_1, ENABLE); /*Set PD1 as
input */
GPIO_EnableFuncReg(GPIO_PD, GPIO_FUNC_REG_1, GPIO_BIT_1); /*Set
PD1 as ENCB0 */
```

Enable the ENC0 and ENC0 interrupt.

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

Get the status of rotation direction of ENC0 to indicate the rolling direction of the mouse wheel. Blink the LED0 according to the cycle of rolling

```
/* LED0 blink speed based on the rolling of mouse wheel */
while(1){
    f_enc = ENC_GetENCFlag(EN0);
    g_flag = f_enc.Bit.RotationDirection;
    LedOn(LED0);
    Delay2(enccount);
    LedOff(LED0);
    Delay2(enccount);
}
```

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

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

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

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

7-16 PMD

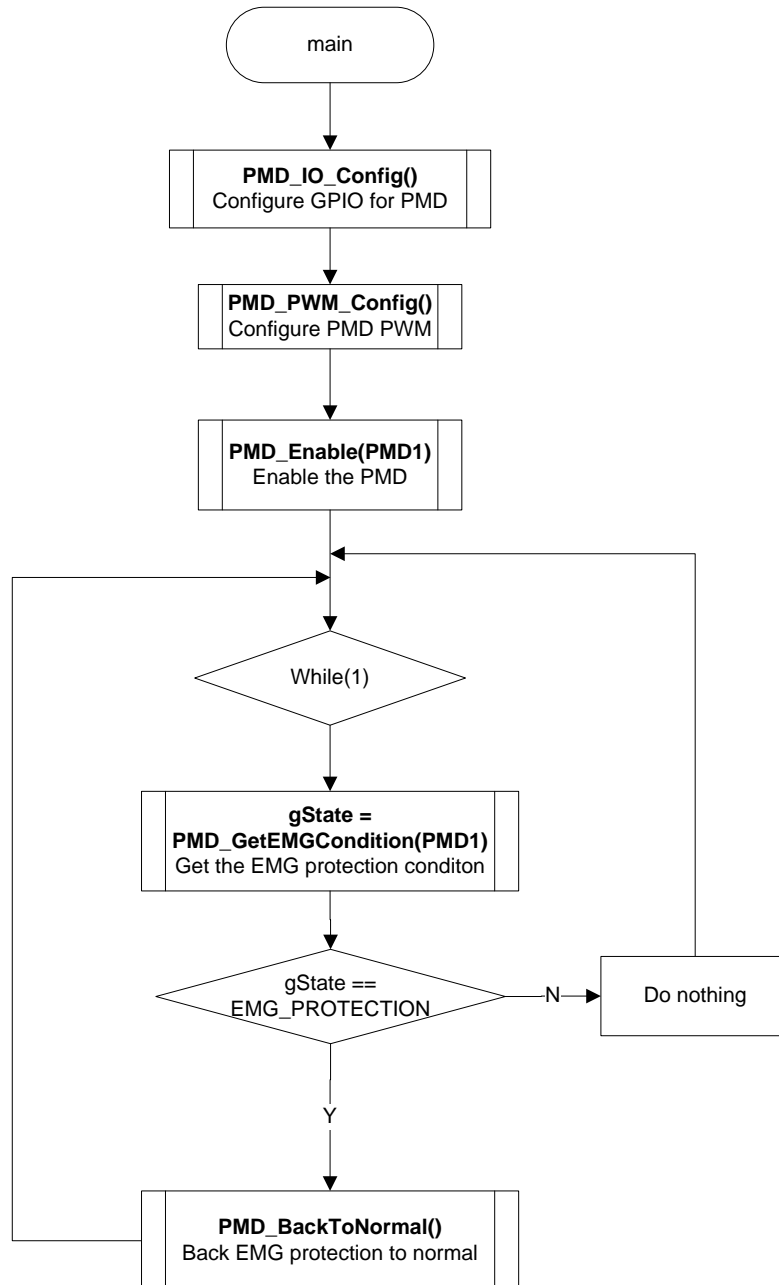
7-16-1 Example: Phase Output

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

The example includes:

1. PMD and GPIO initialization.
2. EMG protection detection.
3. Back EMG protection to normal.

- Flowchart



• Code and Explanation for the Example

At first, configure the LED and PMD and GPIO.

```

/* LED initialization */
LEDInit();

/* GPIO configuration*/
PMD_IO_Config();

/* PMD PWM configuration*/
PMD_PWM_Config();
  
```

If the DEMO_U_PHASE is defined, the duty mode is U-phase in common,

If the DEMO_3_PHASE is defined, the duty mode is 3-phase independent.

```

PMD_InitTypeDef m_pmd;
  
```

```

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

PMD_Init(PMD1, &m_pmd);

```

Set the different compare value in the 3 phases.

In the duty mode U-phase in common, the outputs are same according to phase U.

In the duty mode 3-phase independent, the outputs are independent according to the compare value.

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

And then enable the PMD.

```
PMD_Enable(PMD1);
```

Judge the EMG protection condition.If the condition is in protection condition, LED3 will light on, and then back EMG protection to normal. If the condition is not in protection condition, LED3 will light off.

```

while(1){
    /* Get the EMG protection condition*/
    gState = PMD_GetEMGCondition(PMD1);
    /* Judge the EMG protection condition */
    if (gState == EMG_PROTECTION) {
        /* LED3 will light on if the condition is in protection */
        LedOn(LED3);
        /* Back EMG protection to normal */
        PMD_BackToNormal();
        Delay(1000U);
    } else {
        /* LED3 will light off if the condition is not in protection */
        LedOff(LED3);
    }
}

```


7-17 IGBT

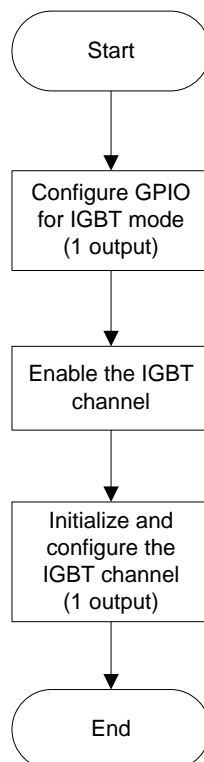
7-17-1 Example: Single PPG Output

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

The example includes:

1. IGBT timer's configuration and initialization (one PPG output).
2. IGBT timer's EMG protection function.
3. The usage of IGBT time's status and EMG interrupt.

- **Flowchart**



- **Code and Explanation for the Example**

After the LED configuration, create IGBT_InitTypeDef structure and fill all the data fields. For example,

```

myIGBT.StartMode = IGBT_FALLING_TRG_START;
myIGBT.OperationMode = IGBT_CONTINUOUS_OUTPUT;
myIGBT.CntStopState = IGBT_OUTPUT_INACTIVE;
myIGBT.ActiveAcceptTrg = DISABLE;
myIGBT.INTPeriod = IGBT_INT_PERIOD_1;
myIGBT.ClkDiv = IGBT_CLK_DIV_1;
  
```

```
myIGBT.Output0Init = IGBT_OUTPUT_HIGH_ACTIVE;
myIGBT.Output1Init = IGBT_OUTPUT_DISABLE;
myIGBT.TrgDenoiseDiv = IGBT_DENOISE_DIV_64;
myIGBT.Output0ActiveTiming = 1U;
myIGBT.Output0InactiveTiming = 1U + IGBT_PPG_PERIOD_50US / 2;
myIGBT.Period = IGBT_PPG_PERIOD_50US;
myIGBT.EMGFunction = IGBT_EMG_OUTPUT_HIZ;
myIGBT.EMGDenoiseDiv = IGBT_DENOISE_DIV_16;

IGBT_IO_Configuration(IGBT0, &myIGBT);
```

Then enable the IGBT channel and EMG protection interrupt, and cancel EMG state before the operation of initialization and configuration.

```
IGBT_Enable(IGBT0);
NVIC_EnableIRQ(INTMTEMG0_IRQn);
NVIC_EnableIRQ(INTMTTB01_IRQn);

do {
    counter_state = IGBT_GetCntState(IGBT0);
} while (counter_state == BUSY);
do {
    cancel_result = IGBT_CancelEMGState(IGBT0);
} while (cancel_result == ERROR);
```

If IGBT_CancelEMGState() returns SUCCESS, call IGBT_Init(), the initialization of IGBT can be complete.

```
IGBT_Init(IGBT0, &myIGBT);
```

Before accepting the start trigger, the software start command must be issued at first.

```
IGBT_SetSWRunState(IGBT0, IGBT_RUN);
```

After this, the corresponding port will output PPG wave once the start trigger is detected.

If EMG protection level on GEMG is active, the EMG interrupt occurs. Turn on the LED0 and stop the timer.

```
void INTMTEMG0_IRQHandler(void)
{
    LedOn(LED0);
    IGBT_SetSWRunState(IGBT0, IGBT_STOP);
}
```

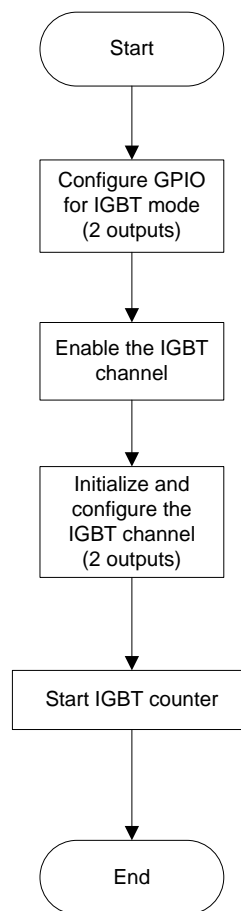
7-17-2 Example: Double PPG Output

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

The example includes:

1. GBT timer's configuration and initialization (two PPG output).
2. IGBT timer's EMG protection function.
3. The usage of IGBT time's status and EMG interrupt.

- **Flowchart**



- **Code and Explanation for the Example**

After the LED and switch port configuration, create IGBT_InitTypeDef structure and fill all the data fields. For example,

```

myIGBT.StartMode = IGBT_CMD_START;
myIGBT.OperationMode = IGBT_CONTINUOUS_OUTPUT;
myIGBT.CntStopState = IGBT_OUTPUT_INACTIVE;
myIGBT.ActiveAcceptTrg = DISABLE;
myIGBT.INTPeriod = IGBT_INT_PERIOD_1;
myIGBT.ClkDiv = IGBT_CLK_DIV_1;
  
```

```
myIGBT.Output0Init = IGBT_OUTPUT_HIGH_ACTIVE;
myIGBT.Output1Init = IGBT_OUTPUT_HIGH_ACTIVE;
myIGBT.TrgDenoiseDiv = IGBT_NO_DENOISE;
myIGBT.Output0ActiveTiming = 1U;
myIGBT.Output0InactiveTiming = 1U + IGBT_ACTIVE_PERIOD_20US;
myIGBT.Output1ActiveTiming = myIGBT.Output0InactiveTiming
                             + IGBT_DEAD_TIME_5US;
myIGBT.Output1InactiveTiming = myIGBT.Output1ActiveTiming
                              + IGBT_ACTIVE_PERIOD_20US;

myIGBT.Period = IGBT_PPG_PERIOD_50US;
myIGBT.EMGFunction = IGBT_EMG_OUTPUT_HIZ;
myIGBT.EMGDenoiseDiv = IGBT_DENOISE_DIV_16;

IGBT_IO_Configuration(IGBT0, &myIGBT);
```

Then enable the IGBT channel and EMG protection interrupt, and cancel EMG state before the operation of initialization and configuration.

```
IGBT_Enable(IGBT0);
NVIC_EnableIRQ(INTMTEMG0_IRQn);
do {
    counter_state = IGBT_GetCntState(IGBT0);
} while (counter_state == BUSY);
do {
    cancel_result = IGBT_CancelEMGState(IGBT0);
} while (cancel_result == ERROR);
```

If IGBT_CancelEMGState() returns SUCCESS, call IGBT_Init(), the initialization and configuration of IGBT can be complete.

```
IGBT_Init(IGBT0, &myIGBT);
```

If switch is ON, send the start command to run.

```
if (SW_Get(SW1) == 1U) {
    IGBT_SetSWRunState(IGBT0, IGBT_RUN);
} else {
    /* Do nothing */
}
```

If EMG protection level on GEMG is active, the EMG interrupt occurs. Turn on the LED0 and stop the timer.

```
void INTMTEMG0_IRQHandler(void)
{
    LedOn(LED0);
    IGBT_SetSWRunState(IGBT0, IGBT_STOP);
}
```

8 Revision History

Revision	Date	Description
1.0	2019-8-23	First release