TOSHIBA

TX04 Peripheral Driver Usage Example (TMPM440)

Ver 1

Sep, 2017

TOSHIBA ELECTRONIC DEVICES & STORAGE CORPORATION

CMDR-M440UE-01E

RESTRICTIONS ON PRODUCT USE

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

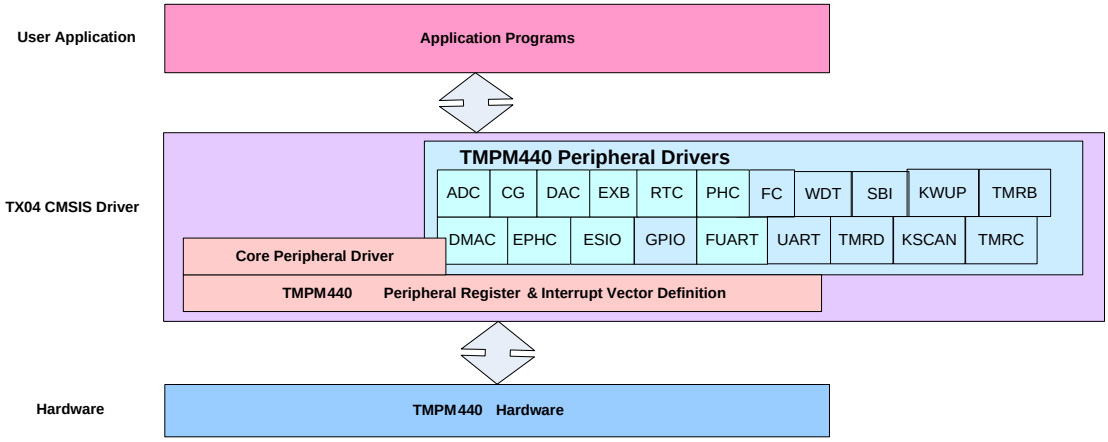
Index

1 General description

The example programs described hereafter are specially designed for TOSHIBA TMPM440 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 TX04 peripherial driver as following.



3 Build-in hardware usage

Hardware	Channel	Use presence, use
ADC	Unit A	Used for ADC demo
	Unit B	Not used
	Unit C	Not used
CG	-	Used for CG demo
Standby mode	-	Use SLEEP mode
External interrupt	Other external intterrup	Not used
SIO	SIO0	Used for UART retarget demo, Used for DMAC_UART demo, Used for SIO demo
	SIO1	Used for DMAC_UART demo, Used for SIO demo
	Others channel	Not used
16-bit timerB	TMRB0	Used for TMRB: General Timer,
	TMRB12	Used for TMRB: PPG output
	Others TMRB	Not used
WDT	WDT	Used for WDT demo
PHC	2 channel	PHC0 is used for Demo
SBI	SIB0	Used for SBI SLAVE demo
DMAC	DMACA	Used for DMAC_UART demo
	Others units	Not used
PSC	-	Used for PSC repeat proc demo

4 Pin Usage

The example programs are tested on TOSHIBA TMPM440 Evaluation Board (the chip mounted is M440FEXBG) . Following is pin usage for the example programs.

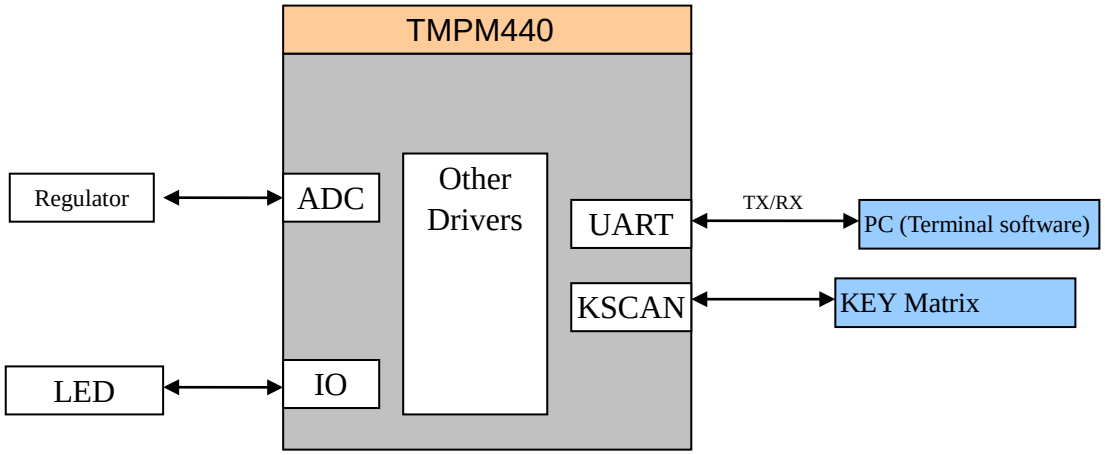
Pin No.	Name	Usage
A7	PAD0	SW0
C8	PAD1	SW1
B7	PAD2	SW2
A6	PAD3	SW3
G7	PAD4	SW4
C6	PAD5	SW5
B7	PAD6	SW6
A5	PAD7	SW7
M19	PT0	LED0
M18	PT1	LED1
N20	PT2	LED2
K14	PT3	LED3
N19	PT4	LED4
L15	PT5	LED5
N18	PT6	LED6
P20	PT7	LED7
L3	PU2	PHC Demo output 0
M6	PU3	PHC Demo output 1
L1	PU4	PHC0IN0
M2	PU5	PHC0IN1
L19	PY4	TD0OUT0
L18	PY5	TD1OUT0
K3	DAAOUT	DAAOUT
W17	PA0	EBIF Data BUS D0
Y18	PA1	EBIF Data BUS D1
V18	PA2	EBIF Data BUS D2
U18	PA3	EBIF Data BUS D3
W19	PA4	EBIF Data BUS D4
Y19	PA5	EBIF Data BUS D5
W20	PA6	EBIF Data BUS D6
W18	PA7	EBIF Data BUS D7
V14	PB0	EBIF Data BUS D8
W14	PB1	EBIF Data BUS D9
P13	PB2	EBIF Data BUS D10
R14	PB3	EBIF Data BUS D11
E20	PH4	UART0 TX
F20	PH5	UART0 RX
A14	PK4	UART1 TX
A13	PK5	UART1 RX
G20	PH6	SIO SCLK0
A15	PK6	SIO SCLK1

Pin No.	Name	Usage
B15	PW4	Tmrb12 output
Y15	PB4	EBIF Data BUS D12
V15	PB5	EBIF Data BUS D13
W15	PB6	EBIF Data BUS D14
P14	PB7	EBIF Data BUS D15
V12	PC0	EBIF ADDR BUS A0
W12	PC1	EBIF ADDR BUS A1
R12	PC2	EBIF ADDR BUS A2
Y13	PC3	EBIF ADDR BUS A3
V13	PC4	EBIF ADDR BUS A4
W13	PC5	EBIF ADDR BUS A5
R13	PC6	EBIF ADDR BUS A6
Y14	PC7	EBIF ADDR BUS A7
R18	PD0	EBIF ADDR BUS A8
N15	PD1	EBIF ADDR BUS A9
T19	PD2	EBIF ADDR BUS A10
U20	PD3	EBIF ADDR BUS A11
T18	PD4	EBIF ADDR BUS A12
U19	PD5	EBIF ADDR BUS A13
V20	PD6	EBIF ADDR BUS A14
V19	PD7	EBIF ADDR BUS A15
L14	PE0	EBIF ADDR BUS A16
P19	PE1	EBIF ADDR BUS A17
M15	PE2	EBIF ADDR BUS A18
R20	PE3	EBIF ADDR BUS A19
P18	PE4	EBIF ADDR BUS A20
M14	PE5	EBIF ADDR BUS A21
R19	PE6	EBIF ADDR BUS A22
T20	PE7	EBIF ADDR BUS A23
N14	PF0	EBIF CTL BUS RD
Y16	PF1	EBIF CTL BUS WR
V16	PF2	EBIF CTL BUS BELL
R15	PF3	EBIF CTL BUS BELH
W16	PF5	EBIF CTL BUS CS0
H19	PR2	EPHC Demo output 0
H15	PR3	EPHC Demo output 1
K20	PR6	EPHC0IN0
K19	PR7	EPHC0IN1
J19	PR4	SBI SLAVE DEMO SCL0
J18	PR5	SBI SLAVE DEMO SDA0

Pin No.	Name	Usage
C3	PAA0	AINA0
H18	PR0	FUART TXD6
H20	PR1	FUART RXD6
H19	PR2	FUART CTS6
H15	PR3	FUART RTS6

5 Development Environment

Following is development environment:



1. Hardware board:
   TOSHIBA M440 Evaluation Board.

2. Development tool:
   - IAR:
     1) J-Link: IAR J-Link-7.0/8.0
     2) IDE:IAR Embed Workbench for ARM version 6.40.2
   - KEIL:
     1) u-Link: Realview ULINK2
     2) IDE:KEIL uVision MDK version 4.54

***Note:** For PSC module the development tool version is different

- IAR:
  1) J-Link: IAR J-Link-ARM 7.0 / J-Link 6.0
  2) IDE:IAR Embedded workbench 6.50.1 version
- KEIL:
  1) IDE: KEIL uVision 4.60

6 Functions

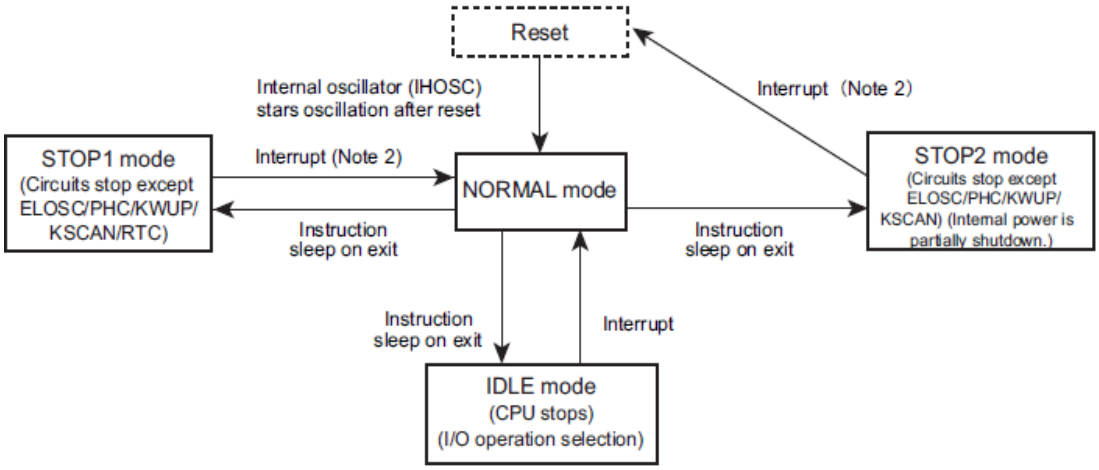
This chapter will focus on the functions demonstrated in driver usage example.

Operation mode

There are 4 operation modes for TMPM440: NORMAL, IDLE, STOP1 and STOP2 mode.
IDLE and STOP mode are low power mode.

- **NORMAL mode:**

This mode is to operate the CPU core and the peripheral hardware by using the high-speed clock.
The below figure shows a mode transition diagram.



6-1 ADC

ADC Data Read

There is a potentiometer that connected to PAA1/AIN01. The voltage on it will be measured and output the change voltage value. As the change of the voltage shows in the LED display. While the output voltage become higher, the more LED is on.

6-2 CG

NORMAL <-> STOP1 mode change

This is a simple demo to change the CPU operation mode. Only 2 modes(NORMAL and STOP1) are supported .

Current mode	Action (Key)	Operation	LED display
NORMAL	Key1 is pressed(low active)	NORMAL→ STOP1	LED1,2,3,4 off
STOP1	Key2 is pressed(low active)	STOP1 →NORMAL	LED1,2,3,4 on

Note:
Key1 is connected to pin Y12 (PP6, inner pull up, active low, used to enter STOP1 mode)
Key2 is connected to pin R11 (PP7, inner pull up, active low, used as interrupt source INT0 to release STOP1 mode)

NORMAL <-> STOP2 mode change

Change the CPU operation mode between NORMAL mode and STOP2 mode.

Current mode	Action (Key)	Operation	LED display
NORMAL	Trun off SW0	NORMAL→ STOP2	LED7 is OFF.
STOP2	Trun on SW0 at first,Trun on Exernal Interrupt SW*.	STOP2 →NORMAL	LED 7 is ON.

Note: External interrupt SW is KEY matrix.

NORMAL <-> IDLE mode change

Change the CPU operation mode between NORMAL mode and IDLE mode.

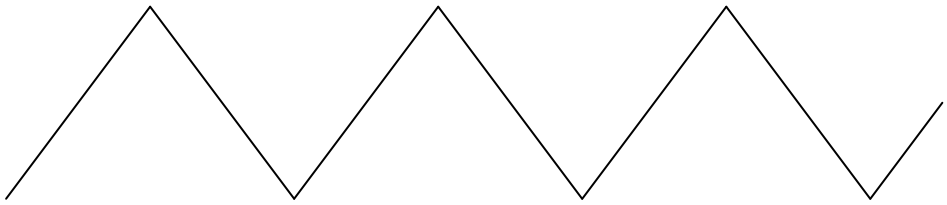
Current mode	Action (Key)	Operation	LED display
NORMAL	Trun off SW0	NORMAL→ IDLEWFI	LED7 is OFF.
IDLE	Trun on SW0 at first,Trun on Exernal Interrupt SW*.	IDLE →NORMAL	LED 7 is ON.

Note: External interrupt SW is KEY matrix.

6-3 DAC

Sawtooth waveform output

Change the output code of DAx from 0 to 1023 at the fixed interval; when it reaches 1023,change the output code of DAx from 1023 to 0 at the fixed interval. So the waveform outputted from the DAx pin will be as the following:



6-4 DMAC

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 TMPM440-EVAL evaluation board must be modified:

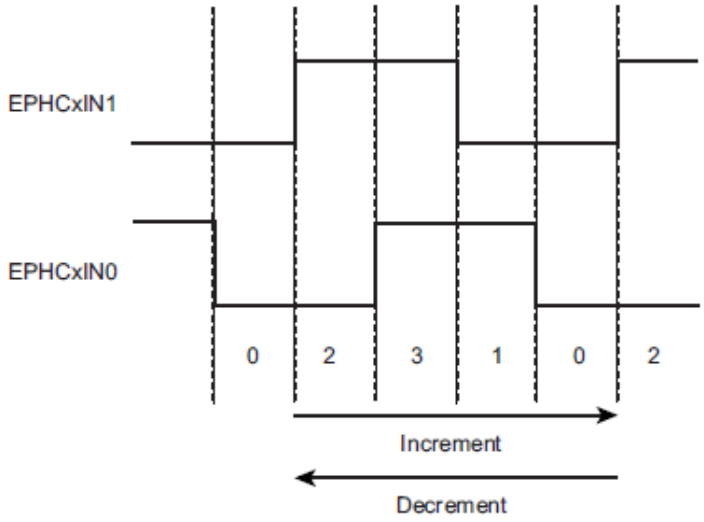
Connect the TX of UART0(E20，PH4) and RX of UART1(A13,PK5) via wire

6-5 EPHC

EPHC_Counter

This function intends to use the Quadruple mode of EPHC.

Using GPIO to simulate two-phase input as EPHC0IN0 and EPHC0IN1 input.
The input waveform like following graphic EPHCxCNT<MA2DIR>="1" (Negative direction):



+

The up-and-down counter is increment or decrement when the state transition of the two-phase pulse changes once.

Counter should get 40 times for increment and 40 times for decrement.

Counter results can be checked by RAM variable count_result[] in debugger.
count_result[0] = 0x7FFE,
count_result[1] = 0x7FFD,
count_result[2] = 0x7FFC
…
count_result[38] = 0x7FD8,
count_result[39] = 0x7FD7,
count_result[40] = 0x7FD8,
count_result[41] = 0x7FD9,
count_result[42] = 0x7FDA
…
count_result[79] = 0x7FFF.

Note: In order to run this sample software, use TOSHIBA TMPM440 Eval board which must be modified.

EPHC pins:
Pin PR6 (EPHC0IN0)
Pin PR7 (EPHC0IN1)

Connect the pin PR6 (pin k20 EPHC0IN0) and pin PR2 (pin H19 PortR output).
Connect the pin PR7 (pin k10 EPHC0IN1) and pin PR3 (pin H15 PortR output).

6-6 ESIO

Basic transfer

This demo will show basic usage of ESIO module in M440.
It use single transfer on ESIO0, (connect its Tx with Rx). The LEDs will blink to show the data is being received. User can define BITRATE_MIN to see the blink effect.

Note: Connect pin A8(PJ0, EXIO_Tx0) to pin B8(PJ4, ESIO_Rx0),

6-7 EXB

SRAM Read/Write

This function implements read/write the external SRAM assembled on TMPM440 evaluation board and EXB is set as 16-bit bus width on separate bus. The A.C specifications of SRAM (cycles time) used is for specific SRAM chip. The external SRAM is IS62WV51216BLL with 1Mbyte size.

***Note:**

1. AD[0:15] of TMPM440 evaluation board connect AD[0:15] of IS62WV51216BLL SRAM;
2. A16 of TMPM440 evaluation board connect A15 of IS62WV51216BLL SRAM;
3. CS0,WE,OE,ALE,BELLn, BELHn of TMPM440 evaluation board connect CE,WE,OE,ALE, LB,UB of SRAM.

6-8 FC

FC SWAP

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

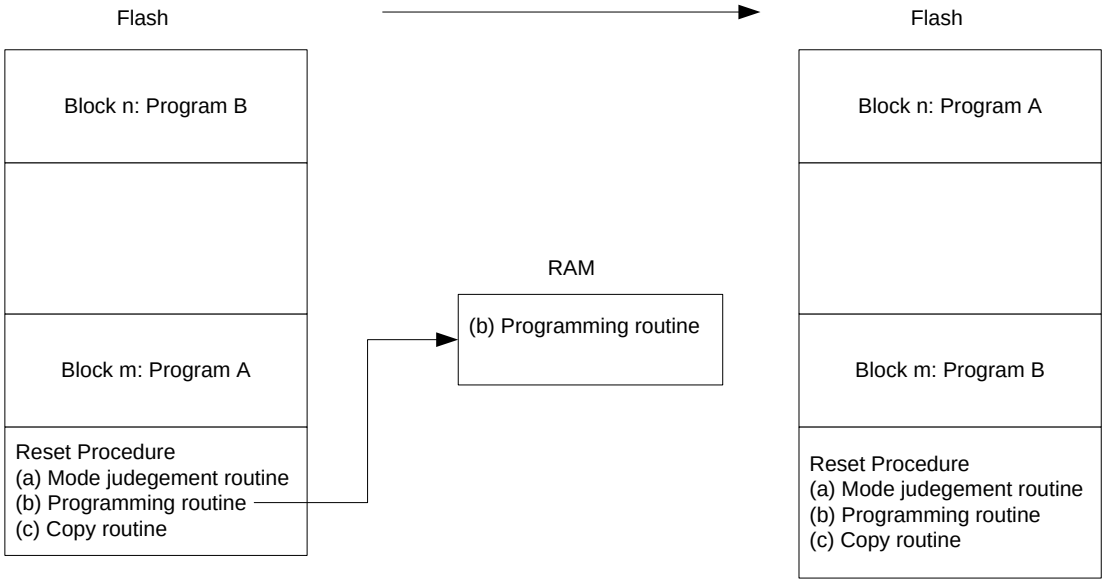
－Program A/B（A: LED 0 is on, B: LED 1 is on)

By default, the program A will run firstly

－Button SW0 is used for mode judgment in Reset procedure

SW0 is released　　Normal mode

SW0 is pressed　　User boot mode



Demo sequence

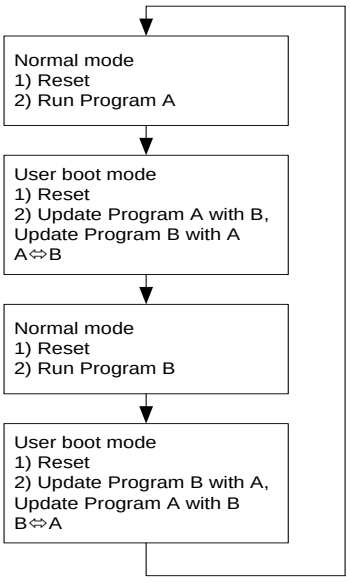
(1) Power on

When SW0 is released, power on or press RESET button. ProgramA will run at first. LED 0 is blink.

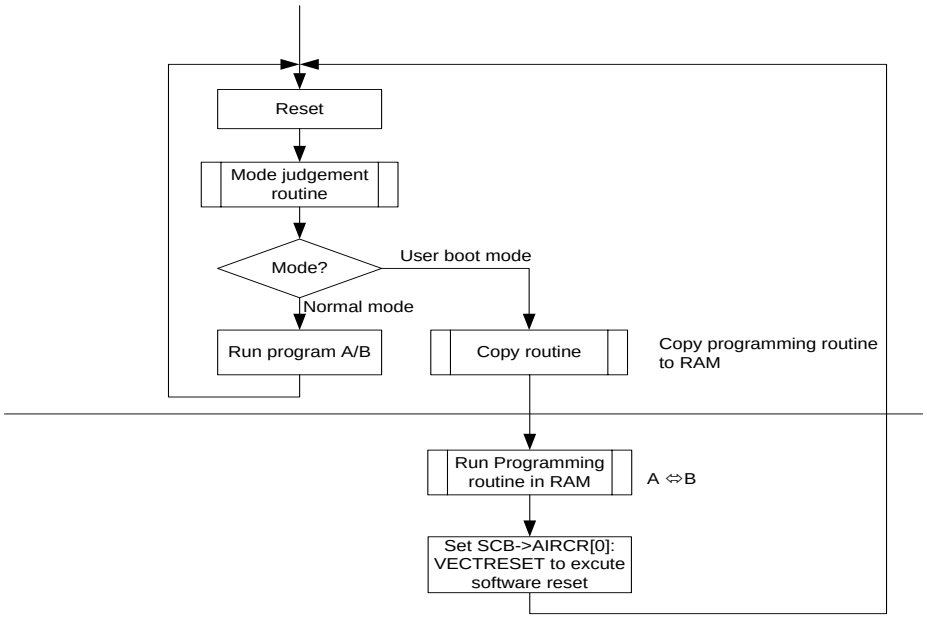
(2) Pressing SW0, then press RESET button.

Release SW0 when LED2 is ON.

(3) After exchange of successful, system will reset automatically to run DemoB. LED1 is blink.



・**Demo process flow**



When the demo entering user boot mode, LED will display as below:

LED2 is on
(User boot mode)

6-9 FUART

LoopBack

In this program, both Full UART Transmit and Receive FIFO are enabled.
This program sends 64 different data value from Full UART channel 6 TXD6 pin and receives data from RXD6 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 RXD6 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.
The first time:
Enable RTS and CTS hardware flow control, the received data are same with transmitted data.
The second time:
Disable RTS and CTS hardware flow control, the received data are different with transmitted data.

NOTE:

Connect TX6 (H18) pin with RX6 (H20) pin on TMPM440 MCU board.

Connect RTS6 (H15) pin with CTS6 (H19) pin on TMPM440 MCU board.

6-10 GPIO

GPIO for LED

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

6-11 KSCAN

KSCAN_Demo

This is a simple application based on the Peripheral Driver (KSCAN, GPIO).
The LED0 to LED7 is ON when the key0 to K7 is pressed one by one.

6-12 KWUP

Low power mode release

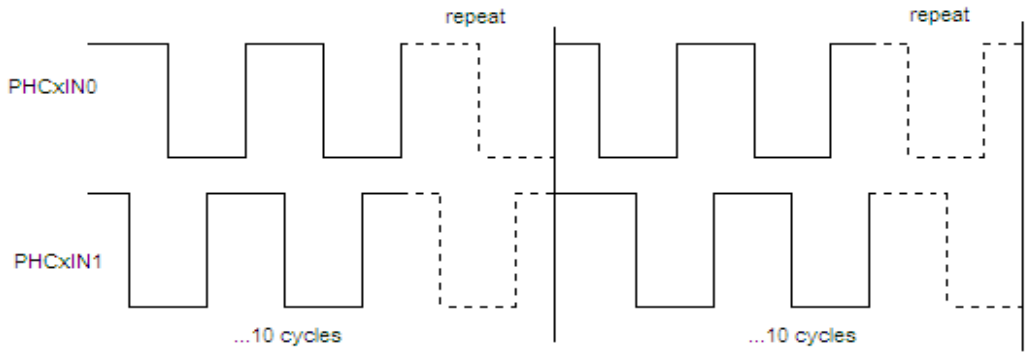
This function implements press KEY1 (PAD1) CPU enter low power mode,LED is shutdown,then press KEY0(PAD0) to release low power mode.

6-13 PHC

PHC_Counter

This function intends to use the Quadruple mode of PHC.

Using GPIO to simulate two-phase input as PHC0IN0 and PHC0IN1 input.
The input waveform like following graphic:



The up-and-down counter is increment or decrement when the state transition of the

two-phase pulse changes once.

Counter should get 40 times for increment and 40 times for decrement.

Counter results can be checked by RAM variable count_result[] in debugger.
count_result[0] = 0x7FFE,
count_result[1] = 0x7FFD,
count_result[2] = 0x7FFC
…
count_result[38] = 0x7FD8,
count_result[39] = 0x7FD7,
count_result[40] = 0x7FD8,
count_result[41] = 0x7FD9,
count_result[42] = 0x7FDA
…
count_result[79] = 0x7FFF.

Note: In order to run this sample software, use TOSHIBA TMPM440 Eval board which must be modified.

PHC pins:
Pin PU4 (PHC0IN0)
Pin PU5 (PHC0IN1)

Connect the pin PU4 (PHC0IN0) and pin PU2 (PortU-Pin2 output).
Connect the pin PU5 (PHC0IN1) and pin PU3 (PortU-Pin3 output).

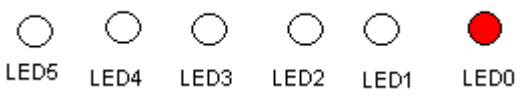
6-14 RTC

RTC_Basic

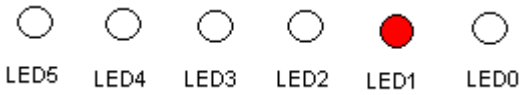
Use the internal RTC to display the second on LEDs.

Display the second of the time as binary on LEDs.

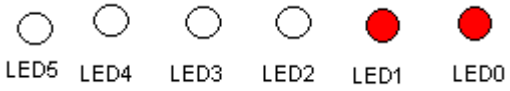
LEDs displays example:

1s: 

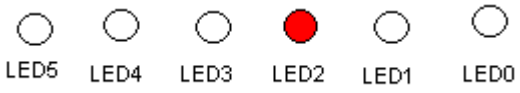
2s:



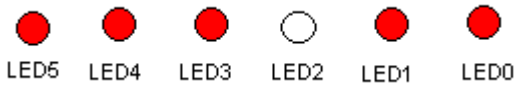
3s:



4s:



…….



59s (hex: 0x3B = 0b00111011)

6-15 SBI

SBI Slave

This function intends to support the I2C bus master and slave mode.
Connect two I2C buses on the two evaluation boards, one evaluation board works as master and another evaluation board works as slave.

Uses SBI interrupt to handle I2C bus read/write.

Address of I2C slave: 0xB0.
I2C Slave (SBI0) of board one receives “TOSHIBA” from I2C Master (SBI0) of the other board.
Received results can be checked by RAM variable gI2CrxData[] in debugger.

Note: In order to send data between the two boards, connect the ports below.
Connect the two pins PR4 of M440 Evaluation board (SCL).
Connect the two pins PR5 of M440 Evaluation board (SDA).
Connect two ground pins of the boards to make the two boards have the same ground.

6-16 TMRB

General Timer

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

PPG Output

Use Toggle Switch 1 to change PPG waveform. Leadingtiming can be changed as following:
10%, 25%, 50%, 75%, 90%
Power on to enter PPG mode and starts the PPG output.
Change the leadingtiming every switch changing:
10% ->25%--> 50%-->75% --> 90%-->10%

6-17 TMRC

Up Counter

This timer is a 32-bit binary counter, A counter value can be captured into capture register by software. Every time "1" is written to TCTBTRUN<TBTCAP>, a counter value at this point is captured into TCTBTCP register. If up counter is overflowed, an overflow interrupt INTTCTBT will occur and the counter value will be cleared to "0" then start up counting.

6-18 TMRD

Interval timer

This function implements a general timer by using TMRD0 16-bit interval timer.
Timer trailingtiming is set to 500us.
Use this timer for all LEDs blinking and the period is 1s (500ms ON and 500ms OFF).

Interlock PPG output

This function implements interlock PPG output by using TMRD0.
PPG cycle is set to 500us, and leadingtiming is 50%.(250us H level, 250us L level).
Phase A is set to fast than phase B, and the phase shift (delay) (θ) is 120 degrees. The phase A0 and B0 can be observed by oscilloscope from PY4/TD0OUT0 and PY6/TD1OUT0.

6-19 SIO/UART

Retarget

This function intends to retarget the stdin and stdout of the C lib.
Stdin and stdout are retarget to UART0. Then application code can use printf() and getchar() to output to serial port or input from serial port.

UART FIFO Demo

In this sample program, UART0 sent the data "TMPM4401" to UART3 use FIFO, and at same time UART0 receive the data "TMPM4402" which send from UART3 and also use FIFO.

Set one breakpoint before the function ResetIdx(),when program stop in the breakpoint you can read the RxBuffer = "TMPM4402",and RxBuffer1 = "TMPM4401".

SIO Demo

This demo will show synchronously transfer and receive usage of SIO module in TMPM440.
It use the channel SIO0, SIO1 and transfer data synchronously between them.(connect TXD0 with RXD1, connect TXD1 with RXD0, connect sclk0 with sclk1).

Note: Connect pin G20(PH6, SIO0_SCLK) with pin A15(PK6, SIO1_SCLK).
Connect pin E20(PH4, SIO0_TXD0) with pin A13(PK5, SIO1_RXD1).
Connect pin A14(PK4, SIO1_TXD1) with pin F20(PH5, SIO0_RXD0).

6-20 WDT

WDT Demo

The watchdog timer begins operation immediately after a reset is released. If not using the watchdog timer, it should be disabled. The watchdog timer can not be used at the high-speed frequency clock is stopped. Before transition to below operation modes, the watchdog timer should be disabled. In IDLE mode, its operation depends on WDMOD<I2WDT> setting.

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 disabled.
   DEMO2:
   WDT clear code will be written to the watchdog timer control register, and watch dog timer counter will be cleared.

6-21 PSC

PSC Repeat Proc

The code and data of PSC are transmitted after a reset start using DMAC.
The transmitted PSC program is repeatedly executed using TMRB channel 9.

7 Software

This software project creates the sample applications based on TOSHIBA TMPM440 Evaluation Board which will demonstrate the main feature of TMPM440FEXBG.

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:

```
\---TMPM440
    +---Libraries
    |   +---TX04_CMSIS
    |   |   |   system_TMPM440.c
    |   |   |   system_TMPM440.h
    |   |   |   TMPM440.h
    |   |   \---startup
    |   |       +---arm
    |   |       |       startup_TMPM440.s
    |   |       \---iar
    |   |               startup_TMPM440.s
    |   \---TX04_Periph_Driver
    |       +---inc
    |       |       tmpm440_adc.h
    |       |       tmpm440_cg.h
    |       |       tmpm440_dac.h
    |       |       tmpm440_dmac.h
    |       |       tmpm440_ephc.h
    |       |       tmpm440_esio.h
    |       |       tmpm440_exb.h
    |       |       tmpm440_fc.h
    |       |       tmpm440_fuart.h
    |       |       tmpm440_gpio.h
    |       |       tmpm440_kscan.h
    |       |       tmpm440_kwup.h
    |       |       tmpm440_phc.h
    |       |       tmpm440_rtc.h
    |       |       tmpm440_sbi.h
    |       |       tmpm440_tmrb.h
    |       |       tmpm440_tmrc.h
    |       |       tmpm440_tmrd.h
    |       |       tmpm440_uart.h
```

```
|       |       tmpm440_wdt.h
|       |       tx04_common.h
|       \---src
|       |       tmpm440_adc.c
|       |       tmpm440_cg.c
|       |       tmpm440_dac.c
|       |       tmpm440_dmac.c
|       |       tmpm440_ephc.c
|       |       tmpm440_esio.c
|       |       tmpm440_exb.c
|       |       tmpm440_fc.c
|       |       tmpm440_fuart.c
|       |       tmpm440_gpio.c
|       |       tmpm440_kscan.c
|       |       tmpm440_kwup.c
|       |       tmpm440_phc.c
|       |       tmpm440_rtc.c
|       |       tmpm440_sbi.c
|       |       tmpm440_tmrb.c
|       |       tmpm440_tmrc.c
|       |       tmpm440_tmrd.c
|       |       tmpm440_uart.c
|       |       tmpm440_wdt.c
\---Project
    +---Examples                //only 1 examples listed here
    |   +---ADC
    |   |   \---ADC_Data_Read
    |   |       +---App
    |   |       |       main.c
    |   |       +---IAR
    |   |       |       ADC_Data_Read.ewd
    |   |       |       ADC_Data_Read.ewp
    |   |       |       ADC_Data_Read.eww
    |   |       \---KEIL
    |   |               ADC_Data_Read.uvopt
    |   |               ADC_Data_Read.uvproj
    |   \---Workspace
    |       +---IAR
    |       |       Examples_for_M440_Driver.eww
    |       \---KEIL
    |               Examples_for_M440_Driver.uvmpw
    \---Template
        +---IAR
```

```
|           TMPM440FEXBG_Flash.icf
\---KEIL
            tmpm440.sct
```

7-1 ADC

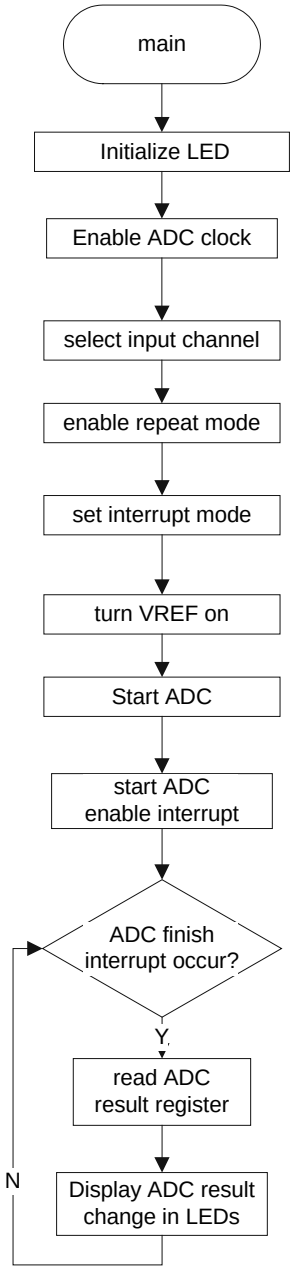
Example: ADC Data Read

This is a simple example based on the TX04 Peripheral Driver (ADC).

The example includes:

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

- **Flowchart:**



- **Code and Explanation for the Example**

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

```
/* Enable ADC clock supply */
CG_SetPLL1ForADC(ENABLE);

/* set ADC clock */
ADC_SetClk(TSB_ADA,
           ADC_CONVERSION_CLK_80,
           ADC_FC_DIVIDE_LEVEL_8);

/* select ADC input channel */
ADC_SetInputChannel(TSB_ADA, ADC_AN_01);
```

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

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

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

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

Then start ADC and enable INTAD:

```
    ADC_Start(TSB_ADA);             /* Start ADC in unit A */

    /* enable ADA interrupt */
    NVIC_EnableIRQ(INTADA_IRQn);
```

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

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

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

```
    ADC_Result result = ADC_GetConvertResult(TSB_ADA, ADC_REG_00);
```

7-2 CG

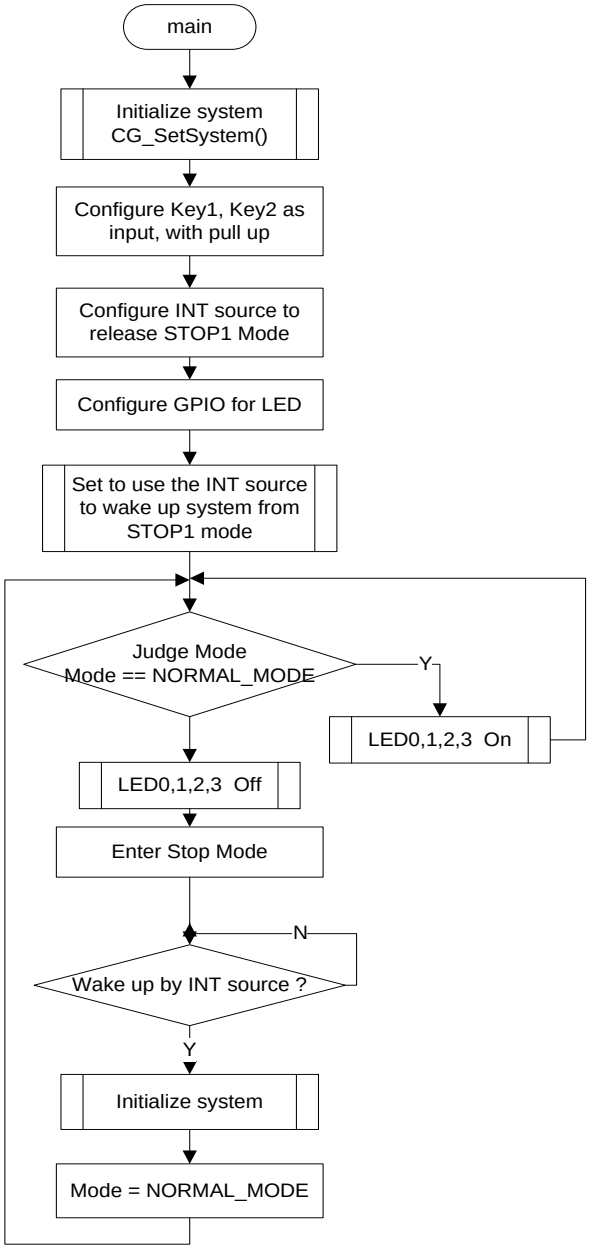
7-2-1 Example: NORMAL <-> STOP1 mode change

This is a simple example based on the TX04 Peripheral Driver (CG, GPIO).

The example includes:

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

- **Flowchart**



- **Code and Explanation for the Example**

Normal setup for CG (after reset)

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

```
void CG_SetSystem(void)
{
    if (CG_GetFoscSrc()==CG_FOSC_OSC_INT){
        /* Switch over from IHOSC to EHOSC*/
        switchFromIHOSCtoEHOSC();
    }

    /* Set up pll and wait for pll to warm up, set fc source to fpll */
    CG_EnableClkMulCircuit();

    /* Set fgear = fc/2 */
    CG_SetFgearLevel(CG_DIVIDE_2);

    /* Set fperiph to fgear */
    CG_SetPhiT0Src(CG_PHIT0_SRC_FGEAR);
    CG_SetPhiT0Level(CG_DIVIDE_64);

    /* Set low power consumption mode stop */
    CG_SetSTBYMode(CG_STBY_MODE_STOP1);

    /* Set pin status in stop mode to "active" */
    CG_SetPinStateInStop1Mode(ENABLE);
}
```

Configure Keys, LED pins, INT source pins

Configure the I/O pins for Key, LED and INT source

```
        /* Configure Key1(PP6), Key2(PP7) as input, with pull up */
        TSB_PP->IE |= 0x03U << 6U ;
        TSB_PP->PUP |= 0x03U << 6U ;

        /* Configure INT0 function on pin PP7(Key2) */
        TSB_PP->FR3 |= 0x01U << 7U;

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

Configure external interrupt to wake up system

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

```
    CG_ClearINTReq(CG_INT_SRC_0);

    /* Set to use INT0, pull up and low active) to wake up system from STOP mode */
    CG_SetSTBYReleaseINTSrc(CG_INT_SRC_0,
                            CG_INT_ACTIVE_STATE_FALLING, ENABLE);

    NVIC_ClearPendingIRQ(INT0_IRQn);
    NVIC_EnableIRQ(INT0_IRQn);
```

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,CG_WUODR_EXT);
	/* Enter stop mode */
	__WFI_wait();
}
```

Eight __NOP() instruction are added after __WFI() instruction because the status of interrupt might be disable.

```
void __WFI_wait()
{
	__WFI();
	__NOP();
	__NOP();
	__NOP();
	__NOP();
	__NOP();
	__NOP();
	__NOP();
	__NOP();
}
```

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,
					CG_WUODR_PLL);
		CG_StartWarmUp();

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

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

	return retval;
}
```

7-2-2 Example: NORMAL <-> STOP2 mode change

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

The example includes:

1. Basic setup operation of CG
2. How to switch between normal mode and stop2 mode

- **Code and Explanation for the Example**

Configure the I/O pins for Key, LED and KSCAN.

```
SW_Init();
LED_Init();

configGPIO_KSCAN(ENABLE, DISABLE);    /* Set KSCAN's GPIO */
```

When a reset factor is not STOP2 mode release, configure KSCAN and configure interrupt to release standby mode. Then start KSCAN. In this sample, low frequency oscillation is used as a KSCAN source clock.

```
configCG_FS(WARMTIME);

KSCAN_SetSCLK(KSCAN_KSCL_FS);      /*select fs for ksclk */

configKSCAN();                /* KSCAN initial */

CG_ClearINTReq(CG_INT_SRC_INTKSCAN);
NVIC_ClearPendingIRQ(INTKSCAN_IRQn);

NVIC_EnableIRQ(INTKSCAN_IRQn);

KSCAN_SetINTReq(ENABLE);

startKSCAN(ENABLE);
```

When a reset factor is STOP2 mode release, configure interrupt to release standby mode. Then disable port keep function.

```
CG_SetSTBYReleaseINTSrc(CG_INT_SRC_INTKSCAN,
CG_INT_ACTIVE_STATE_RISING, DISABLE);

NVIC_EnableIRQ(INTKSCAN_IRQn);

CG_SetPortKeepInStop2Mode(DISABLE);
```

Read a status of SW0 in while() loop after turn on LED7.
When the status of SW0 is OFF, do nothing. When the status of SW0 is ON, enter STO2 mode after turn off LED7.

```
LED_On(LED7);
while (1U) {
    if (SW_Get(SW0) == 0U) {
        LED_Off(LED7);          /* LED7 is off before enter stop2 */

        enterSTOP2();
    }
```

```
    }
```

Enable KSCAN interrupt as a factor of STOP2 mode release.

```
    CG_SetSTBYReleaseINTSrc(CG_INT_SRC_INTKSCAN,
CG_INT_ACTIVE_STATE_RISING, ENABLE);
```

Prepare to enter STOP2 mode.
Select STOP2 mode as a standby mode. Enable internal oscillation after PLL is OFF. Then enable port keep function.

```
    /*Set standby mode as Stop2 */
    CG_SetSTBYMode(CG_STBY_MODE_STOP2);

    TSB_CG->PLLSEL &= 0xfffffffe;
    do {
        tmp = (TSB_CG->PLLSEL & 0x00000001);
    } while (0x00000000 != tmp);

    /* Clock(external) Setup */
    TSB_CG->OSCCR &= OSCCR_WUODR_MASK;
    TSB_CG->OSCCR |= OSCCR_WUODR_EXT;

    if (TSB_CG_OSCCR_XEN2 == 0U) {
        TSB_CG->OSCCR |= 0x00010000;

        TSB_CG->OSCCR |= 0x00080000;

        TSB_CG->OSCCR |= 0x00000001;

        /*wait warm-up finish */
        do {
            tmp = (TSB_CG->OSCCR & 0x00000002);
        } while (0x00000000 != tmp);
    }

    TSB_CG->OSCCR &= 0xfffdffff;

    TSB_CG->STBYCR |= 0x00020000;
```

Finally, enter STOP2 mode by executing __WFI() instruction.

```
    __WFI();
```

7-2-3 Example: NORMAL <-> IDLE mode change

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

The example includes:

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

- **Code and Explanation for the Example**

Configure the I/O pins for Key, LED and KSCAN.

```
SW_Init();
LED_Init();

/* Set KSCAN's GPIO */
configGPIO_KSCAN(ENABLE, DISABLE);
```

Configure KSCAN, then start KSCAN. In this sample, timer output is used as a KSCAN source clock.

```
/* TB19OUT(fPLL:1/1) */
configTMRB_TB19OUT(TMRB19TIME_fPLL_1_1);

KSCAN_SetSCLK(KSCAN_KSCL_TBOUT);

/* KSCAN initial */
configKSCAN();

CG_ClearINTReq(CG_INT_SRC_INTKSCAN);
NVIC_ClearPendingIRQ(INTKSCAN_IRQn);

NVIC_EnableIRQ(INTKSCAN_IRQn);

KSCAN_SetINTReq(ENABLE);

startKSCAN(ENABLE);
```

Read a status of SW0 in while() loop after turn on LED7.
When the status of SW0 is OFF, do nothing. When the status of SW0 is ON, enter IDLE mode after turn off LED7.

```
LED_On(LED7);
while (1U) {
    if (SW_Get(SW0) == 0U) {
        /* LED7 is off before enter IDLE */
        LED_Off(LED7);

        enterIDLE();
    }
}
```

Enable KSCAN interrupt as a factor of IDLE mode release.

```
CG_SetSTBYReleaseINTSrc(CG_INT_SRC_INTKSCAN,
CG_INT_ACTIVE_STATE_RISING, ENABLE);
```

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

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

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

```
__WFI_wait();
```

Eight __NOP() instruction are added after __WFI() instruction because the status of interrupt might be disable.

```
void __WFI_wait()
{
```

```
        __WFI();
        __NOP();
        __NOP();
        __NOP();
        __NOP();
        __NOP();
        __NOP();
        __NOP();
        __NOP();
    }
```

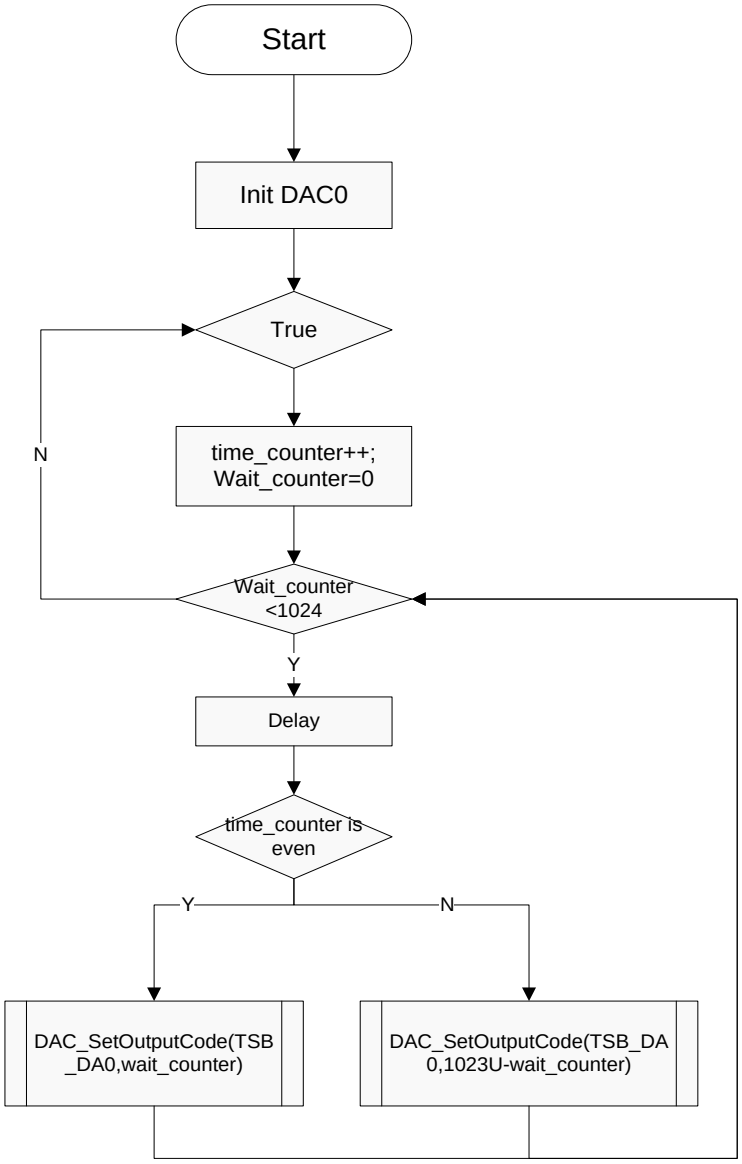
7-3 DAC

Example: Sawtooth waveform output

This is a simple example based on the TX04 Peripheral Driver DAC.

The example includes:
DAC0 sawtooth waveform output

- **Flowchart:**



- **Code and Explanation for the Example**

The following simple example is based on TX04 Peripheral Driver (DAC), which output sawtooth waveform from DA0 pin

Firstly set initial value for DA0 output code and start the operation of it

```
DAC_SetOutputCode(TSB_DA0,0U);
DAC_SetVOutHoldTime(TSB_DAA);
DAC_Start(TSB_DA0);
```

Change the output code at the fixed interval to generate sawtooth waveform.

```
while(1){
        time_counter++;

        for(wait_counter=0U;wait_counter<1024U;wait_counter++){
            for(i=0U;i<3000U;++i){
```

```
            /* Do nothing */
        }
        if( time_counter%2U == 1U){
            DAC_SetOutputCode(TSB_DA0,wait_counter);
        }
        else{
            DAC_SetOutputCode(TSB_DA0,1023U-wait_counter);
        }
    }
}
```

7-4 DMAC

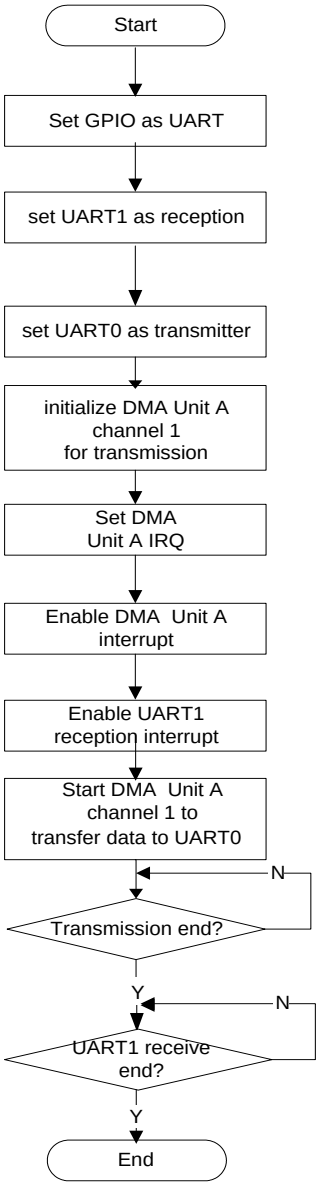
Example: Memory to peripheral

This is a simple example based on the TX04 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 TX04 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);_
UART_SetTxDMAReq(UART0, ENABLE);
```

Configure and initialize DMA Unit A 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.A_TxDstPeriph = DMACA_SIO0_UART0_TX;
DMAC_InitStruct.TxINT = ENABLE;

DMAC_Init(DMAC_UNIT_A, myChannel, &DMAC_InitStruct);
```

Enable DMA Unit A IRQ

```
NVIC_ClearPendingIRQ(INTDMAC0TC_IRQn);
NVIC_EnableIRQ(INTDMAC0TC_IRQn);
```

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

```
DMAC_Enable(DMAC_UNIT_A);
DMAC_SetTxINTConfig(DMAC_UNIT_A, myChannel, DMAC_INT_TX_END, ENABLE);
DMACA_SetSWBurstReq(DMACA_SIO0_UART0_TX);
DMAC_SetDMAChannel(DMAC_UNIT_A, 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(DMAC_UNIT_A);
if (state.Bit.CH1_INTReq == 1U) {
    tmp = DMAC_GetTxINTReq(DMAC_UNIT_A, DMAC_CHANNEL_1);
    if (tmp == DMAC_TX_END_REQ) {
        TxEndFlag = DMAC_GetChannelTxState(DMAC_UNIT_A, DMAC_CHANNEL_1);
        DMAC_ClearTxINTReq(DMAC_UNIT_A, DMAC_CHANNEL_1, DMAC_INT_TX_E
  ND);
    } else {
        /* Do nothing */
    }
```

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

7-5 EPHC

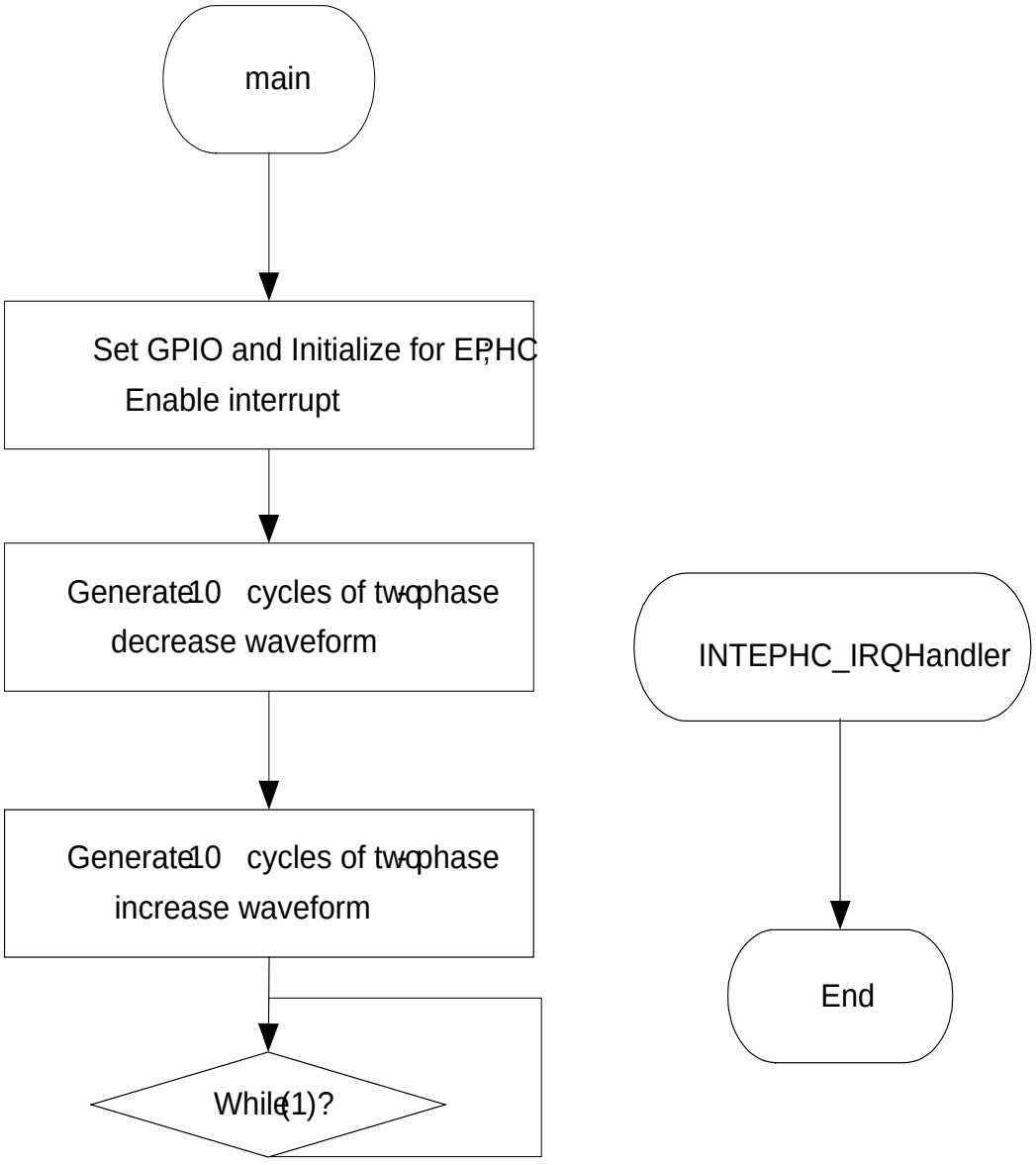
Example: EPHC Demo

This is a simple example based on the TX04 Peripheral Driver (EPHC, GPIO).

The example includes:

1. EPHC up-and-down counter works in quadruple mode

- **Flow Chart**



- **Code and Explanation for the Example**

At first, initialize GPIO Function,

PR6 - EPHC0IN0 function, input

PR7 - EPHC0IN1 function, input

PR2 - GPIO, output

PR3 - GPIO, output

```
#define     GPIO_EPHC0          GPIO_PR
#define     EPHC0IN0_BIT        GPIO_BIT_6
#define     EPHC0IN1_BIT        GPIO_BIT_7
#define     GPIO_EPHCOUT        GPIO_PR
#define     EPHC0IN0_OUT_BIT    GPIO_BIT_2
#define     EPHC0IN1_OUT_BIT    GPIO_BIT_3
```

```
    GPIO_SetInput(GPIO_EPHC0, (EPHC0IN0_BIT | EPHC0IN1_BIT));
    GPIO_SetOutput(GPIO_EPHCOUT, (EPHC0IN0_OUT_BIT | EPHC0IN1_OUT_BIT));
    GPIO_EnableFuncReg(GPIO_EPHC0, GPIO_FUNC_REG_3, (EPHC0IN0_BIT | EPHC0IN1_BIT));
    GPIO_DisableFuncReg(GPIO_EPHCOUT, GPIO_FUNC_REG_2, (EPHC0IN0_OUT_BIT | EPHC0IN0_OUT_BIT));
```

Prepare EPHC initialization structure.

```
EPHC_InitTypeDef InitStruct;

InitStruct.CountDownEdgeSelForP1 = EPHC_CNT_MA1DN_NOCNTDOWN0;
InitStruct.CountUpEdgeSelForP1 = EPHC_CNT_MA1UP_NOCNTUP0;
InitStruct.InputClkSelection = EPHC_CNT_BRCK_FC;
InitStruct.PhaseSelection = EPHC_CNT_PBDIR_POSITIVEPHASE;
InitStruct.ModeSetting = EPHC_CNT_MA12_PULSE_2;
InitStruct.DirectionSetting = EPHC_CNT_MA2DIR_NEGATIVEDIR;
InitStruct.NoiseFilterCtrl = EPHC_CNT_NOISEFILTER_NONE;
InitStruct.CountClearCtrl = EPHC_COUNT_CLR;
```

Enable the EPHC module and set the initialization structure to specified registers. Enable all interrupt, in the end, start the EPHC to run.

```
EPHC_Enable(TSB_EPHC);
EPHC_Init(TSB_EPHC, &InitStruct);
EPHC_DisableInterrupt(TSB_EPHC, EPHC_IE_INT_ALL);
EPHC_EnableInterrupt(TSB_EPHC,EPHC_IE_INT_ALL);
NVIC_EnableIRQ(INTEPHC_IRQn);
EPHC_ASetRunState(TSB_EPHC, EPHC_RUN);
```

Then the main routine will enter “While(1)” ,when counter underflow or overflow occurs the interrupt happens.

```
void INTEPHC_IRQHandler(void)
{
    /* if overflow or underflow interrupt occurred,clear EPHC0ADAT to 0. */
}
```

7-6 ESIO

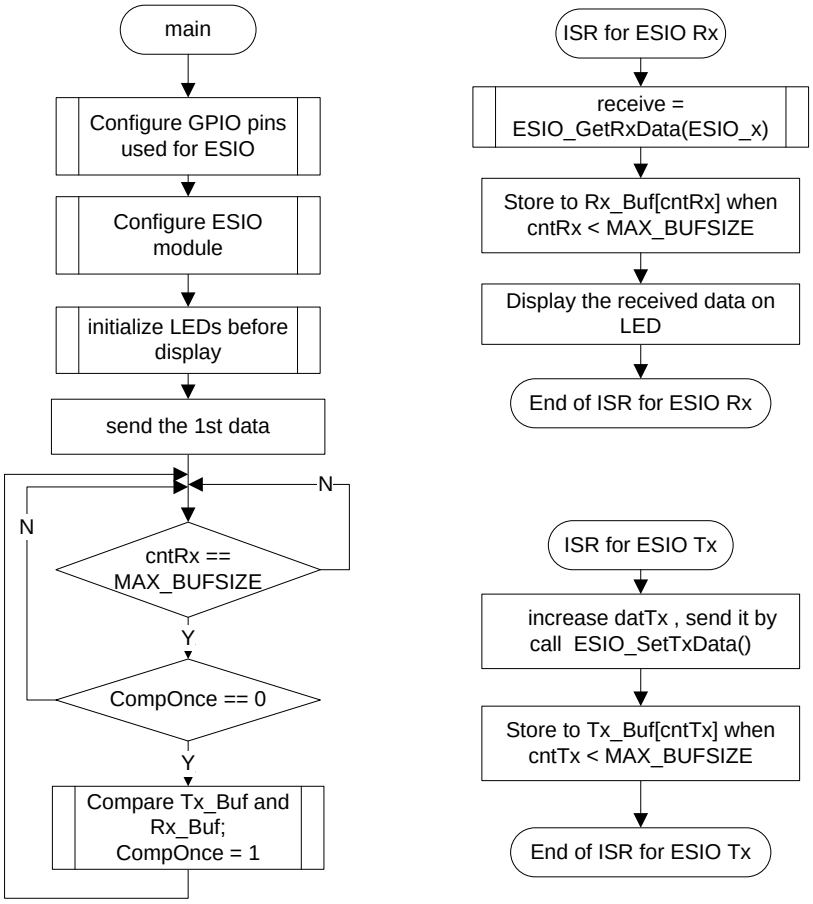
7-6-1 Example: Basic transfer

This is a simple example based on the TX04 Peripheral Driver (ESIO).

The example includes:

1. Basic setup operation of ESIO
2. ESIO interrupt of Tx and Rx

- **Flowchart**



- **Code and Explanation for the Example**

At first, configure the GPIO pins for ESIO by setting the CR, FR1, IE register of proper port. And initialize LEDs port for display.

Then, configure the ESIO module by calling driver functions.

```
void ConfigESIO(void)
{
    /* Single transfer */
    uint8_t transfer_num = 1;
```

```
    /* maximum clock: clk = phiT0/4, divider=1 */
    ESIO_BaudClock clk = ESIO_PHIT0_DIVIDE_4;
    uint8_t divider = 1U;

    ESIO_InitTypeDef init = { 0U };

    ESIO_Enable(ESIO_x);
    /* ESIO_SWReset(ESIO_x); */

    ESIO_SetTxRxCtrl(ESIO_x, ENABLE);
    ESIO_SelectMode(ESIO_x, ESIO_MODE_SIO);
    ESIO_SelectTransferMode(ESIO_x, ESIO_TRMODE_TXRX);

    ESIO_SetTransferNum(ESIO_x, transfer_num);

    ESIO_SetFIFOLevelINT(ESIO_x, ESIO_TX , 0U);
    ESIO_SetFIFOLevelINT(ESIO_x, ESIO_RX , 0U);

    /* enable ESIO interrupt */
    ESIO_SetINT(ESIO_x, ESIO_INT_ALL, ENABLE);

#ifdef BITRATE_MIN
    clk = ESIO_PHIT0_DIVIDE_1024;
    divider = 16U;
#endif
    ESIO_SetBaudRate(ESIO_x, clk, divider );

    init.DataDirection = ESIO_LSB_FIRST;
    init.FrameLength = sizeof(datTx) * 8U;          /* size of datTx in main() */
    init.CycleBetweenFrames = 1U;
    init.NumberOfLINE = 1U;
    init.ClkPolority = ESIO_CLKPOL_LOW;
    init.ShortestIdleCycle = 1U;
    init.CS0ActiveLevel = ESIO_CS_ACTIVE_LOW;
    init.DelayCycleNum_CS_SCLK = 1U;
    init.DelayCycleNum_Negate_CS = 1U;
    init.HorizontalParityCheck = DISABLE;
    /* init.HorizontalParity = ESIO_PARITY_ODD; */
    init.VerticalParityCheck = DISABLE;
    ESIO_Init(ESIO_x, &init);

    NVIC_ClearPendingIRQ(INTE0RX_IRQn );
    NVIC_EnableIRQ(INTE0RX_IRQn);

    NVIC_ClearPendingIRQ(INTE0TX_IRQn);
    NVIC_EnableIRQ(INTE0TX_IRQn);

    NVIC_ClearPendingIRQ(INTE0ERR_IRQn);
    NVIC_EnableIRQ(INTE0ERR_IRQn);

    __enable_irq();


}
```

After that, send the 1st data to trigger the Tx interrupt.

```
    Tx_Buf[0] = datTx;
    ESIO_SetTxData(ESIO_x, datTx);
```

In ISR of Tx, continue sending the rest data.

```
void INTE0TX_IRQHandler(void)
{
      datTx++;
      ESIO_SetTxData(ESIO_x, datTx);
      if (cntTx < MAX_BUFSIZE) {
            Tx_Buf[cntTx] = datTx;
            cntTx++;
      }
}
```

Use ISR of Rx to receive data

```
void INTE0RX_IRQHandler(void)
{
      uint8_t tmp = 0U;

      receive = ESIO_GetRxData(ESIO_x);
      if (cntRx < MAX_BUFSIZE) {
          Rx_Buf[cntRx] = (uint8_t)receive;
          cntRx++;
     }
     /* adjust the loop blinkging speed */
     tmp = (uint8_t) (receive >> 4U);
     LED_Display(tmp);
}
```

7-7 EXB

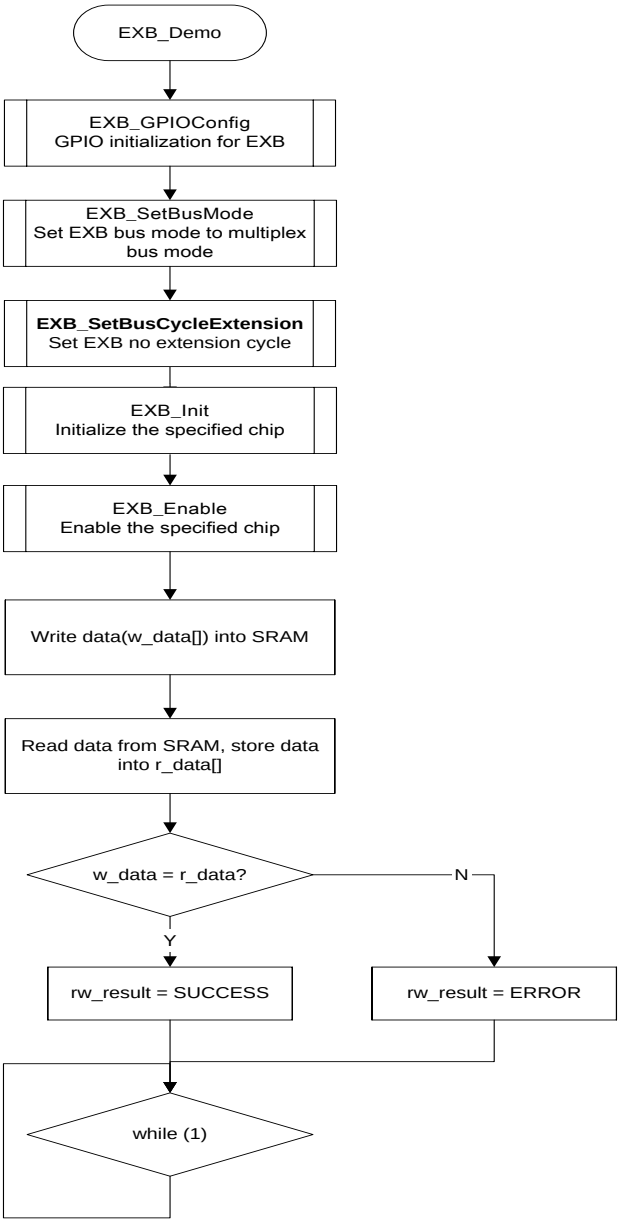
Example: Read/Write SRAM

This is a simple example based on the TX04 Peripheral Driver (EXB, GPIO).

The example includes:

1. Initialization of EXB
2. Read/Write the external SRAM

- **Flowchart:**



- **Code and Explanation for the Example**

Firstly set initial value for EXB.

```
	uint32_t w_data[TEST_DATA_LEN] = { 0U };
	uint32_t r_data[TEST_DATA_LEN] = { 0U };
	uint16_t rw_cnt = 0U;
	uint8_t chip = EXB_CS0;
	uint8_t BusMode = EXB_BUS_SEPARATE;
	uint8_t Cycle = EXB_CYCLE_NONE;

#ifdef SRAM_RW
	uint32_t addr = NULL;
	uint16_t i = 0U;
#endif

	EXB_InitTypeDef InitStruct = { 0U };

	InitStruct.AddrSpaceSize = EXB_1M_BYTE;
	InitStruct.StartAddr = EXB_SRAM_START_ADDR;
	InitStruct.BusWidth = EXB_BUS_WIDTH_BIT_16;
	/* Set cycles time according to AC timing of SRAM datasheet,base clock: EXBCLK(fsys)
*/
	InitStruct.Cycles.InternalWait = EXB_INTERNAL_WAIT_4;
	InitStruct.Cycles.ReadSetupCycle = EXB_CYCLE_1;
	InitStruct.Cycles.WriteSetupCycle = EXB_CYCLE_1;
	InitStruct.Cycles.ALEWaitCycle = EXB_CYCLE_1;
	InitStruct.Cycles.ReadRecoveryCycle = EXB_CYCLE_1;
	InitStruct.Cycles.WriteRecoveryCycle = EXB_CYCLE_1;
	InitStruct.Cycles.ChipSelectRecoveryCycle = EXB_CYCLE_1;
```

Configure GPIO for EXB and EXB, then enable the specified chip. Write w_data[] into SRAM, after reading data from SRAM and store the data into r_data[]. Check rw_result to see whether SRAM write/read is successful.

```
#ifdef SRAM_RW
	EXB_GPIOConfig();
#endif

	EXB_SetBusMode(BusMode);
	EXB_SetBusCycleExtension(Cycle);
	EXB_Init(chip, &InitStruct);
	EXB_Enable(chip);

#ifdef SRAM_RW
	/* SRAM Read/Write demo */
	addr = (uint32_t *) (((uint32_t) InitStruct.StartAddr) | EXB_SRAM_START_ADDR);

	for (i = 0U; i < TEST_DATA_LEN; i++) {
		w_data[i] = i;
	}

	rw_cnt = TEST_DATA_LEN * sizeof(w_data[0]);
	memcpy(addr, w_data, (uint32_t) rw_cnt);
	__DSB();
```

```
	memcpy(r_data, addr, (uint32_t) rw_cnt);

	/* check rw_result to see if SRAM write/read is successful or not */
	if (memcmp(w_data, r_data, (uint32_t) rw_cnt) == 0U) {
		rw_result = SUCCESS;
	} else {
		rw_result = ERROR;
	}
#endif
	while (1) {
		/* Do nothing */
	}
```

7-8 FLASH

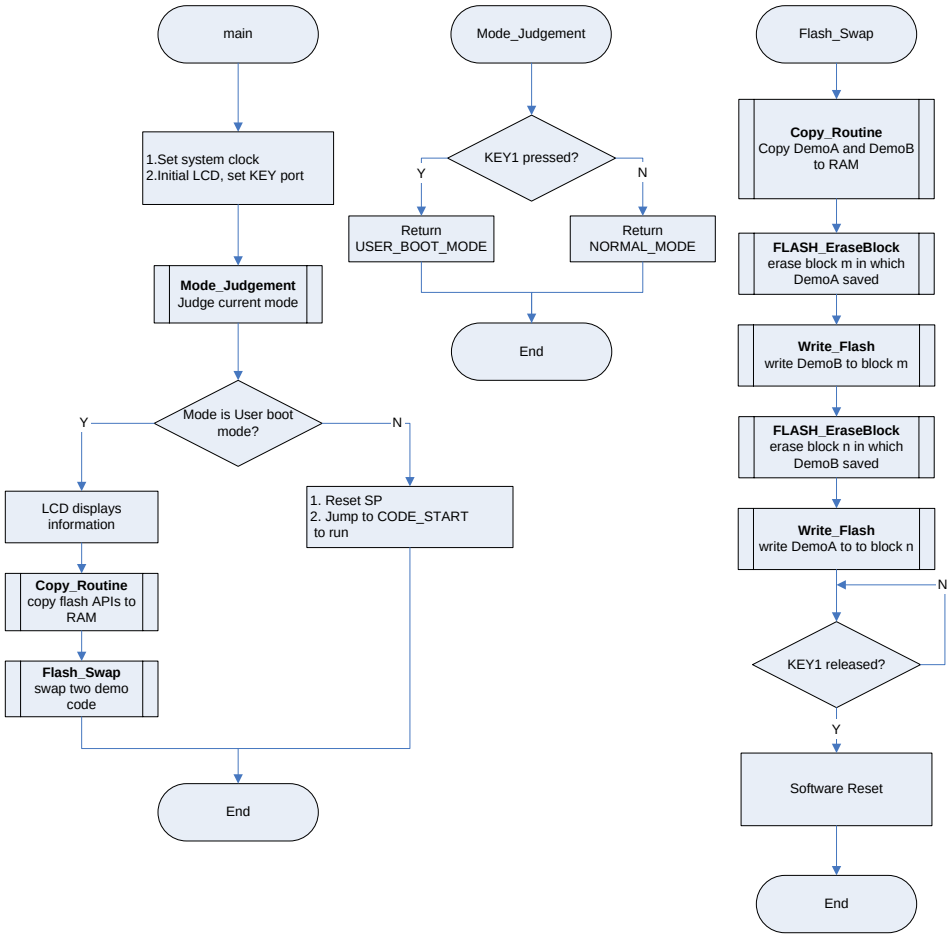
Example: Flash swap

This is a simple example based on the TX04 Peripheral Driver (FLASH).

The example includes:

1. On-board programming method of Flash Memory (Rewrite/Erase)
2. Operation mode: Single chip mode (Normal mode and User boot mode)
3. Use User boot mode to update code in Flash Memory

- **Flowchart**



- **Code and Explanation for the Example**

At first initialize LED and switch. SW0 is used to judge current mode when reset and Flash operation information.

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

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

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

If the SW0 is not turned on when reset, the routine will enter normal mode. Then the routine will reset SP and jump to address “CODE_START” to run. Demo A saved in at address “CODE_START” which belongs to block m, so Demo A will run (LED0 blinks).

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

If the SW0 is turned on when reset, the routine will enter user boot mode. Then the routine will copy APIs for flash operation from address “FLASH_API_ROM” in Flash memory to address “FLASH_API_RAM” in RAM, because Flash memory cannot erase/program itself when runs the code at the same time.

```
        Copy_Routine(FLASH_API_RAM, FLASH_API_ROM, SIZE_FLASH_API);
      /* copy flash API to RAM */
```

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

In function Flash_Swap(), firstly it will copy Demo A and B from Flash memory to RAM, then call function FLASH_EraseBlock() and Write_Flash() to erase and program Flash memory. The code of Demo A and B will be exchanged in Flash memory.

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

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

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

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

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

After SW0 released, it will execute software reset by using SCB->AIRCR register. Then this demo will run again to enter normal mode, because Demo A and B have been exchanged，then Demo B will run (LED1 blinks).

```
  while (SW_Get(SW0) == 1U) {
    }

    NVIC_SystemReset();    /* software reset */
```

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

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

7-9 FUART

Example: LoopBack

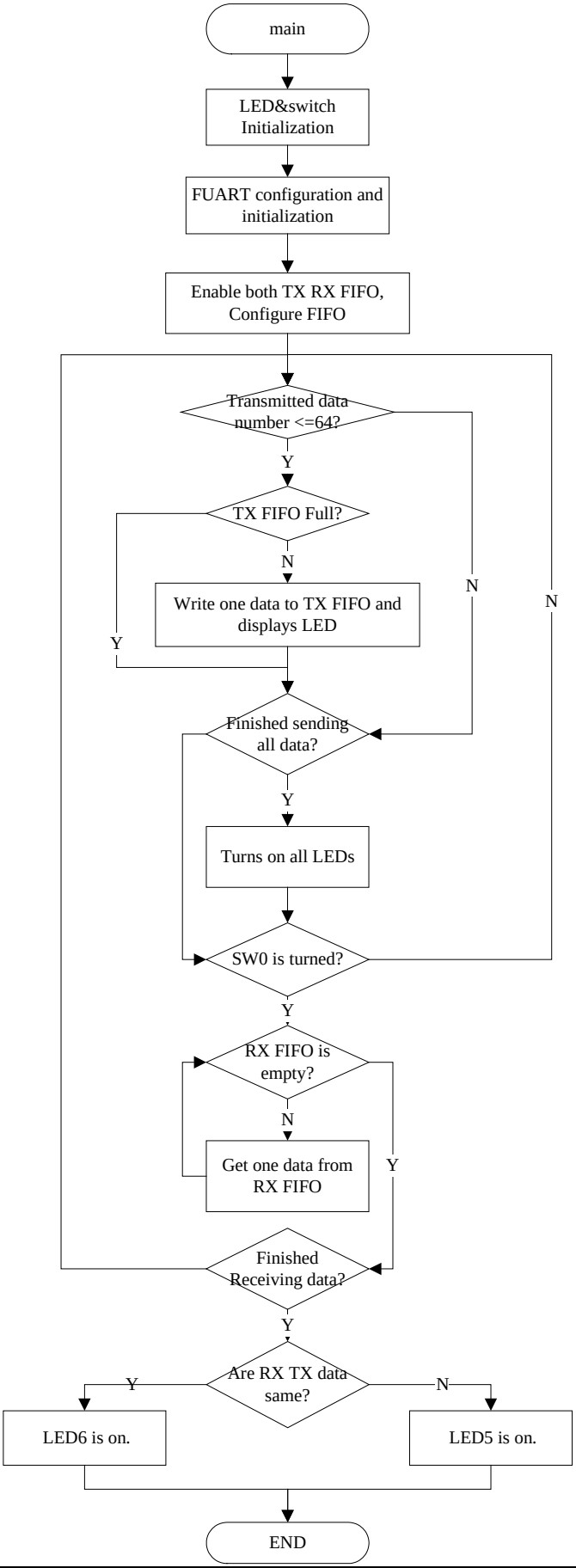
This is a simple example based on the TX04 Peripheral Driver (FUART).

This program can be run for two times to see the hardware flow control function.

The first time: Enable RTS and CTS hardware flow control.
The second time: Disable RTS and CTS hardware flow control.

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
LED&switch Initialization
FUART configuration and initialization
Enable both TX RX FIFO, Configure FIFO
Transmitted data number <=64?
Y
N
TX FIFO Full?
N
Y
Write one data to TX FIFO and displays LED
Finished sending all data?
N
N
Y
Turns on all LEDs
SW0 is turned?
Y
RX FIFO is empty?
N
Y
Get one data from RX FIFO
Finished Receiving data?
Y
Are RX TX data same?
Y
N
LED6 is on.
LED5 is on.
END

- **Code and Explanation for the Example**

Before You run the program You need to decide whether to use the RTS and CTS flow control. If RUN_NONE_FLOW_CONTROL is undefined, RTS and CTS flow control will be enabled in the program. If RUN_NONE_FLOW_CONTROL is defined, there will be no flow control in the program.

```
/* #define RUN_NONE_FLOW_CONTROL */
```

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

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

Configure GPIO for FUART0.

```
    /* Configure port PR0 to be TXD06 */
    TSB_FUART->FR1 |= GPIO_BIT_0;
    TSB_FUART->IE &= ~GPIO_BIT_0;
    TSB_FUART->CR |= GPIO_BIT_0;

    /* Configure port PR1 to be RXD06 */
    TSB_FUART->FR1 |= GPIO_BIT_1;
    TSB_FUART->IE |= GPIO_BIT_1;
    TSB_FUART->CR &= ~GPIO_BIT_1;

    /* Configure port PR2 to be CTS06 */
    TSB_FUART->FR1 |= GPIO_BIT_2;
    TSB_FUART->IE |= GPIO_BIT_2;
    TSB_FUART->CR &= ~GPIO_BIT_2;

    /* Configure port PR3 to be RTS06 */
    TSB_FUART->FR1 |= GPIO_BIT_3;
    TSB_FUART->IE &= ~GPIO_BIT_3;
    TSB_FUART->CR |= GPIO_BIT_3;
```

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;

#ifdef RUN_NONE_FLOW_CONTROL
    myFUART.FlowCtrl = FUART_NONE_FLOW_CTRL;
#else
    myFUART.FlowCtrl = FUART_CTS_FLOW_CTRL | FUART_RTS_FLOW_CTRL;
#endif

    FUART_Init(FUART0, &myFUART);
```

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 (62U == 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, LED6 is on. If received data are different with transmitted data, LED5 is on.

```
    SW0_last = SW0_this;
        SW0_this = SW_Get(SW0);
        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(LED7);
            } else {                    /* receiving data is finished */
                result = Buffercompare(Tx_Buf, Rx_Buf, MAX_BUFSIZE);
                if (result == SAME) {
                    LED_Off(LED_ALL);
                    LED_On(LED6);
                    while (1) {
                    }
                } else {
                    /* received data are different with transmitted data */
                    /* RTS and CTS flow control doesn't work */
                    LED_Off(LED_ALL);
                    LED_On(LED5);
                    while (1) {
                    }
                }
            }
        }
```

7-10 GPIO

Example: GPIO for LED

This is a simple example based on the TX04 Peripheral Driver (GPIO).

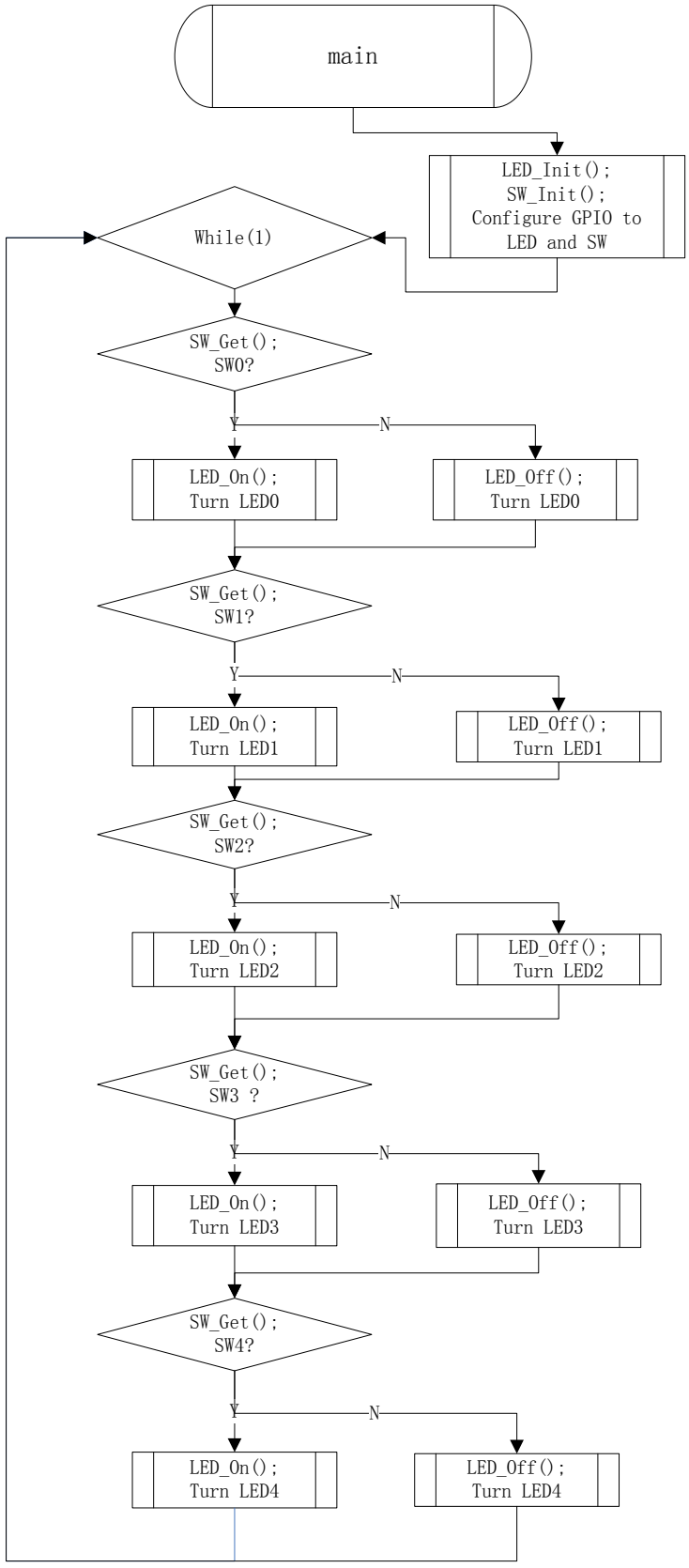
The example includes:

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

Connections of SW, LED, LCD:

Port	Function
PAD0	SW0
PAD1	SW1
PAD2	SW2
PAD3	SW3
PAD4	SW4
PT0	LED0
PT1	LED1
PT2	LED2
PT3	LED3
PT4	LED4

- **Flow chart:**


main
LED_Init();
SW_Init();
Configure GPIO to
LED and SW
While(1)
SW_Get();
SW0?
Y
N
LED_On();
Turn LED0
LED_Off();
Turn LED0
SW_Get();
SW1?
Y
N
LED_On();
Turn LED1
LED_Off();
Turn LED1
SW_Get();
SW2?
Y
N
LED_On();
Turn LED2
LED_Off();
Turn LED2
SW_Get();
SW3 ?
Y
N
LED_On();
Turn LED3
LED_Off();
Turn LED3
SW_Get();
SW4?
Y
N
LED_On();
Turn LED4
LED_Off();
Turn LED4

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

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

```
/* LED0~LED4 */
uint8_t tmp = 0U;
GPIO_SetOutput(LED_DATA_PORT, LED_ALL);
tmp = GPIO_ReadData(LED_DATA_PORT);
tmp &= (~LED_ALL);
GPIO_WriteData(LED_DATA_PORT,tmp);
/* SW0~4 */
GPIO_SetInput(SW_PORT, GPIO_BIT_ALL);
```

In the while(1) process, run the LED demo: Read Switch to control LED on and LED off.

Turn on LED by using GPIO_WriteDataBit(GPIO_Port GPIO_x, uint8_t Bit_x, uint8_t BitValue).

```
uint32_t tmp;
tmp = GPIO_ReadData(LED_DATA_PORT);
tmp |= led;
GPIO_WriteData(LED_DATA_PORT, tmp);
```

Turn off LED by using GPIO_WriteDataBit(GPIO_Port GPIO_x, uint8_t Bit_x, uint8_t BitValue).

```
uint32_t tmp;
tmp = GPIO_ReadData(LED_DATA_PORT);
tmp &= ~led;
GPIO_WriteData(LED_DATA_PORT, tmp);
```

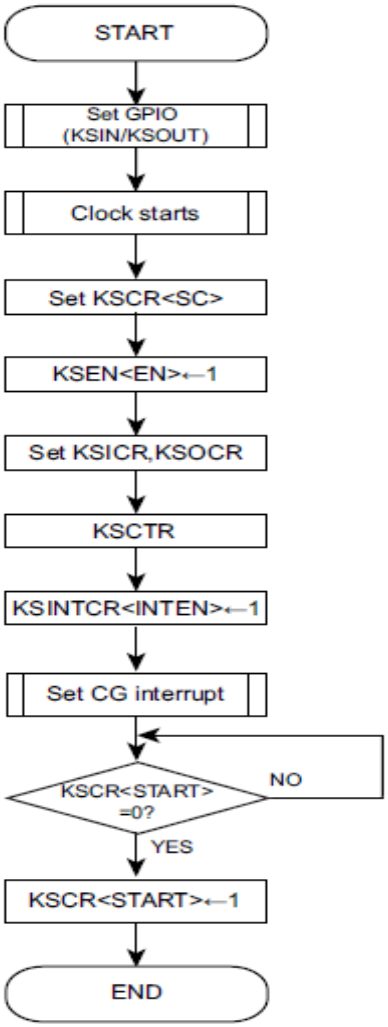
7-11 KSCAN

Example : KSCAN Demo

This is a simple application based on the Peripheral Driver (KSCAN, GPIO, and CG).

The example includes:
1. GPIO initialization
2. KSCAN configuration
3. KSCAN interrupt.

Flow chart:



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

```
LED_Init();
```

```
/*Set clock to FS*/
        CG_SetFs(ENABLE);
        CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC_EXT,WARMTIME);
        do {
            st = CG_GetWarmUpState();
        } while (st != DONE);
/*configure the KSCAN GPIO*/
        KSCAN_GPIOConfig();
/*Clock starts*/
        KSCAN_SetSCLK(KSCAN_KSCL_FS);
        /*Set KSEN(SC)*/

/*Set KSEN<EN>*/
        KSCAN_Enable();

/*Set KSICR, KSOCR*/
        KSCAN_SetInputCtrlMask(KSCAN_CH_0,DISABLE); /*All channels are not masked*/

        KSCAN_SetOutputCtrl((~KSCAN_CH_0),ENABLE);
        KSCAN_SetStrobeOutput(2U);
/*Set KSTCR*/
        KSCAN_SetStrobeWidth(3U);
        KSCAN_SetChatCancelTime(60U);     /*Sets a chattering cancel time*/

        KSCAN_SetConsecutiveMatches(KSCAN_MATCH_2C);
/*KSINTCR<INTEN>, enable KSCAN interrupt*/
        KSCAN_SetINTReq(ENABLE);
/*Set CG interrupt*/
    CG_SetSTBYReleaseINTSrc(CG_INT_SRC_INTKSCAN,CG_INT_ACTIVE_STATE_RISING,ENABLE);
    NVIC_EnableIRQ(INTKSCAN_IRQn);
/*start KSCAN*/
    KSCAN_Start();
```

7-12 KWUP

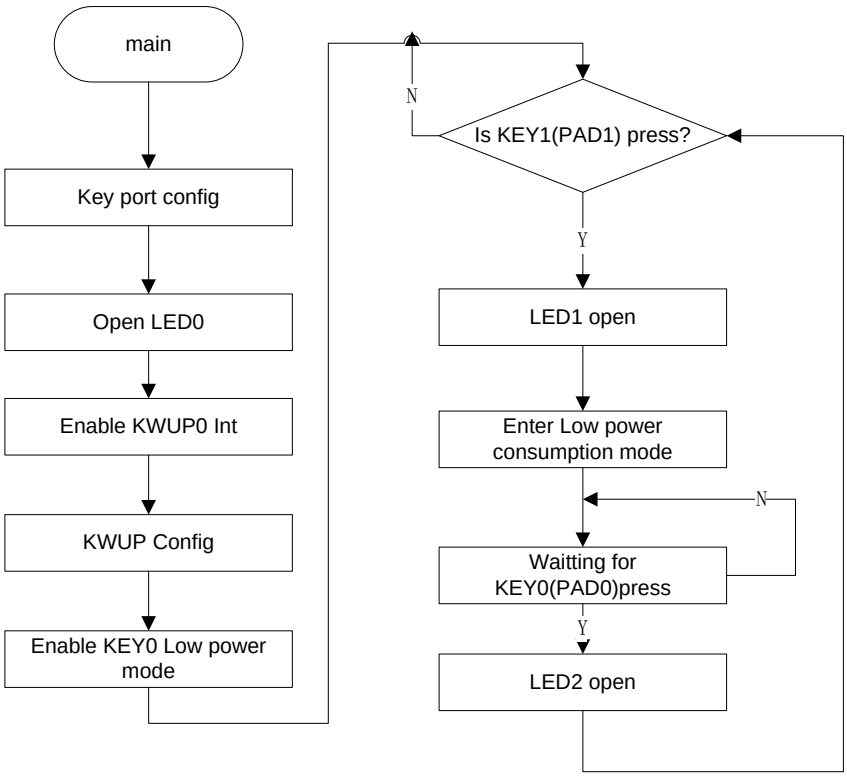
Example: Low power mode release

This is a simple example based on the TX04 Peripheral Driver (KWUP, GPIO).

The example includes:

1. Setting of KWUP
2. Enter and realse Low power mode

- **Flowchart:**



- **Code and Explanation for the Example**

Firstly set initial value for KWUP and Config GPIO for KWUP.

```
KWUP_SettingTypeDef keySetting;
KWUP_PortStatus portStatus;
TSB_KWUP_TypeDef * TSB_KWUPx;

CG_SetSTBYMode(CG_STBY_MODE_STOP1);
/* set PAD0,PAD1 to Key0,1 */
TSB_PAD_FR1_PAD0F1 = 1U;
TSB_PAD_PUP_PAD0UP = 1U;
TSB_PAD_IE_PAD0IE = 1U;
```

```
    TSB_PAD_FR1_PAD1F1 = 1U;
    TSB_PAD_PUP_PAD1UP = 1U;
    TSB_PAD_IE_PAD1IE = 1U;

    LED_Init();
    LED_On(LED0);

    /* enable KWUP interrupt */
    NVIC_ClearPendingIRQ(INTKWUPA_IRQn);
    NVIC_EnableIRQ(INTKWUPA_IRQn);

    keySetting.KeyN = KWUP_INPUT_0;
    keySetting.PullUpCtrl = KWUP_PUP_CTRL_BY_STATIC;
    keySetting.ActiveState = KWUP_ACTIVE_BY_RISING_EDGE;
    keySetting.INTNewState = ENABLE;
    TSB_KWUPx=TSB_KWUPA;
    KWUP_SetConfig(TSB_KWUPx,&keySetting);

    KWUP_SetPullUpConfig(TSB_KWUPx,KWUP_CYCLES_4_FS,
                                    KWUP_CYCLES_256_FS);
```

Press KEY1 (PAD1) , CPU enter low power mode,LED is off, then press KEY0(PAD0) to release low power mode.

```
    /* enable low power mode release interrupt to INTKWUP0 */
    TSB_CG->IMCGF = 0x100000U;
    TSB_CG->IMCGF += 0x010000U;

    do {
        portStatus = KWUP_GetPortStatus(TSB_KWUPx);

        if (portStatus.Bit.Key1 == 0U) {            /* press key(PAD1) */
            LED_On (LED1);
            /* enter low power mode */
            __WFI_wait();

            /* press KEY(PAD0) to release low power mode */

            LED_On(LED2);
        }
    }
    while (1);
```

Eight __NOP() instruction are added after __WFI() instruction because the status of interrupt might be disable.

```
void __WFI_wait()
{
    __WFI();
    __NOP();
    __NOP();
    __NOP();
    __NOP();
    __NOP();
    __NOP();
    __NOP();
    __NOP();
```

```
}
```

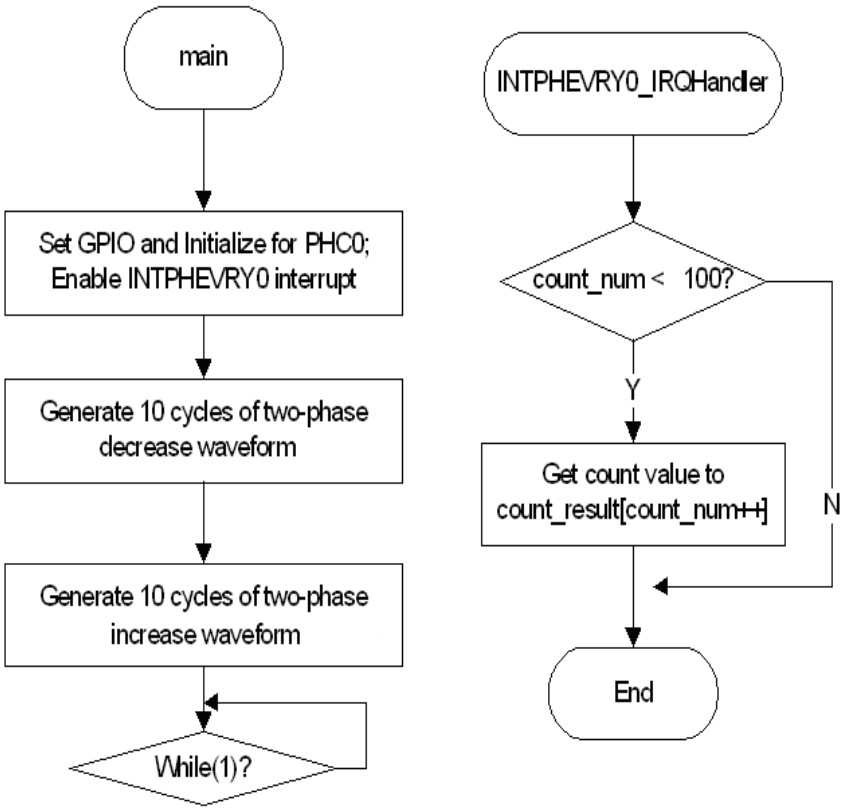
7-13 PHC

Example: PHC Demo

This is a simple example based on the TX04 Peripheral Driver (PHC, GPIO).

The example includes:
PHC up-and-down counter works in quadruple mode

- **Flow Chart**



- **Code and Explanation for the Example**

At first, initialize GPIO Function,
PU4 - PHC0IN0 function, input
PU5 - PHC0IN1 function, input
PU2 - GPIO, output
PU3 - GPIO, output

```
#define     GPIO_PHC0           GPIO_PU
#define     PHC0IN0_BIT         GPIO_BIT_4
```

```
#define     PHC0IN1_BIT          GPIO_BIT_5
#define     GPIO_PHCOUT          GPIO_PU
#define     PHC0IN0_OUT_BIT      GPIO_BIT_2
#define     PHC0IN1_OUT_BIT      GPIO_BIT_3
```

```
    GPIO_SetInput(GPIO_PHC0, (PHC0IN0_BIT | PHC0IN1_BIT));
    GPIO_SetOutput(GPIO_PHCOUT, (PHC0IN0_OUT_BIT | PHC0IN1_OUT_BIT));
    GPIO_EnableFuncReg(GPIO_PHC0,GPIO_FUNC_REG_1,(PHC0IN0_BIT|
PHC0IN1_BIT));
    GPIO_DisableFuncReg(GPIO_PHCOUT,GPIO_FUNC_REG_1, (PHC0IN0_OUT_BIT |
PHC0IN0_OUT_BIT));
```

Prepare PHC initialization structure, fill PHC mode, noise filter controller and couner clear controller. This demo sets PHC quadruple mode, with noise filter and clear the counter.

```
    PHC_InitTypeDef InitStruct;

    InitStruct.Mode = PHC_CR_MODE_4TIMES; /* Quadruple mode */
    InitStruct.NoiseFilterCtrl = PHC_CR_NOISEFILTER_ON; /* Noise Filter on */
    InitStruct.CountClearCtrl = PHC_COUNT_CLR; /* Clear counter */
```

Enable the PHC module and set the initialization structure to specified registers. Enable INTPHEVRY0 interrupt, this interrupt is triggered every counter change, in the end, start the PHC to run.

```
    PHC_Enable(TSB_PHC0); /* Enable PHC Operation */
    PHC_Init(TSB_PHC0, &InitStruct); /* Set InitStruct to PHC */
    PHC_DisableInterrupt(TSB_PHC0, PHC_CR_INT_ALL); /* Disable all interrupt */
    PHC_EnableInterrupt(TSB_PHC0,PHC_CR_INT_EVERY); /* Enable INTPHEVRY0
interrupt */
    NVIC_EnableIRQ(INTPHEVRY0_IRQn);
    PHC_SetRunState(TSB_PHC0, PHC_RUN); /* Run PHC0 */
```

Then the main routine will enter “While(1)” to wait the interrupt happens. In interrupt routine, use a get the count value.

```
    void INTPHEVRY0_IRQHandler(void)
    {
        if(count_num < 100) {              /*Judge if overflow count_result[] or not*/
            count_result[count_num++] = PHC_GetPulseCntValue(TSB_PHC0); /* get
count value*/
        } else {
            /* Do nothing */
        }
    }
```

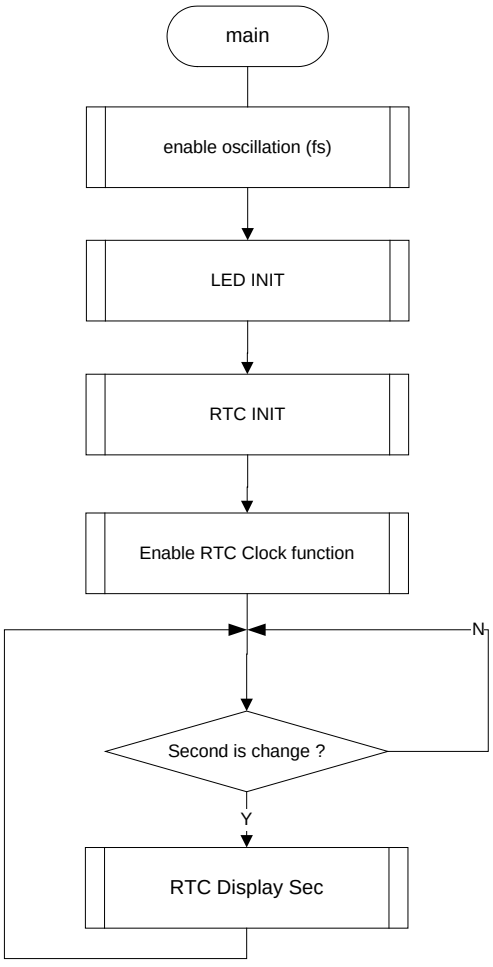
7-14 RTC

Example: RTC Demo

This is a simple example based on the TX04 Peripheral Driver (RTC).

The example includes:
1. RTC initialization
2. Get RTC second

- **Flow chart:**



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

Initialize RTC .Create a RTC_DateTypeDef and RTC_TimeTypeDef structure and fill all the data fields. For example, Initial setting: 2010/10/22 23:59:00, 24hours format.

```
Date_Struct.LeapYear = RTC_LEAP_YEAR_2;
Date_Struct.Year = (uint8_t) 10U;
Date_Struct.Month = (uint8_t) 10U;
Date_Struct.Date = (uint8_t) 22U;
Date_Struct.Day = RTC_FRI;

Time_Struct.HourMode = RTC_24_HOUR_MODE;
Time_Struct.Hour = (uint8_t) 23U;
Time_Struct.Min = (uint8_t) 59U;
Time_Struct.Sec = (uint8_t) 0U;
```

Disable clock and alarm function.

```
RTC_DisableClock();
```

Reset RTC sec counter, disable RTCINT.

```
RTC_ResetClockSec();
RTC_SetRTCINT(DISABLE);
```

Set RTC time and date value

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

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

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

The second value will be sent to LED display in binary.

Following is shown how to get the second value and display it.

```
Sec = RTC_GetSec();
/* Display */
TSB_LED->DATA = Sec
```

7-15 SBI

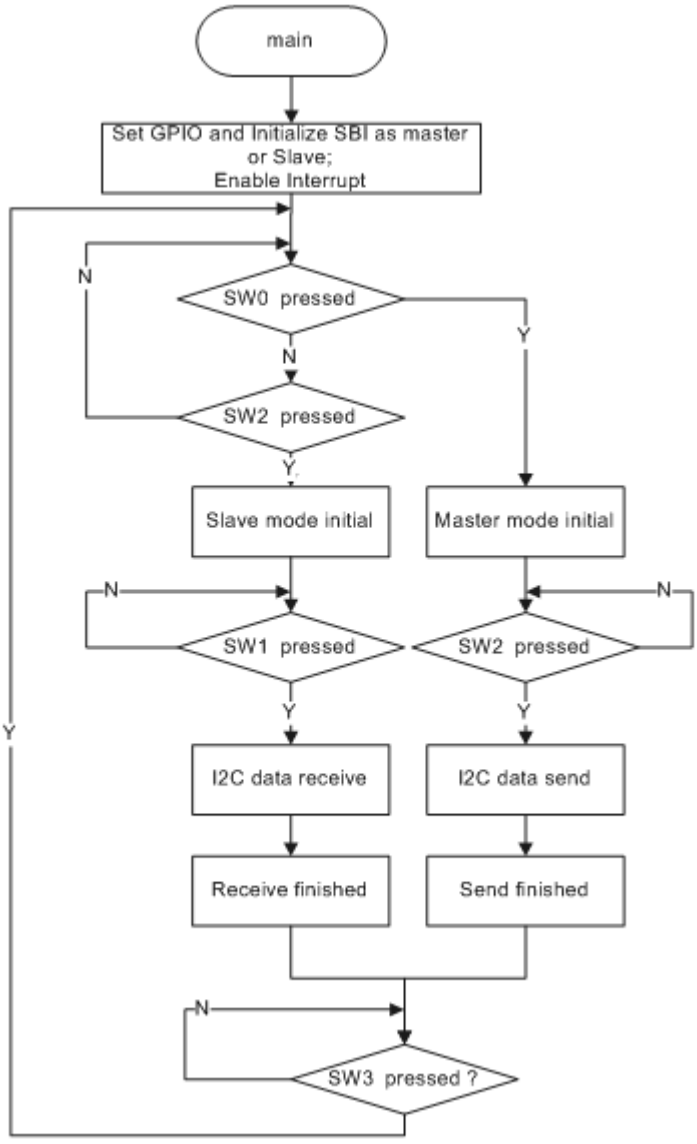
Example: SBI Slave

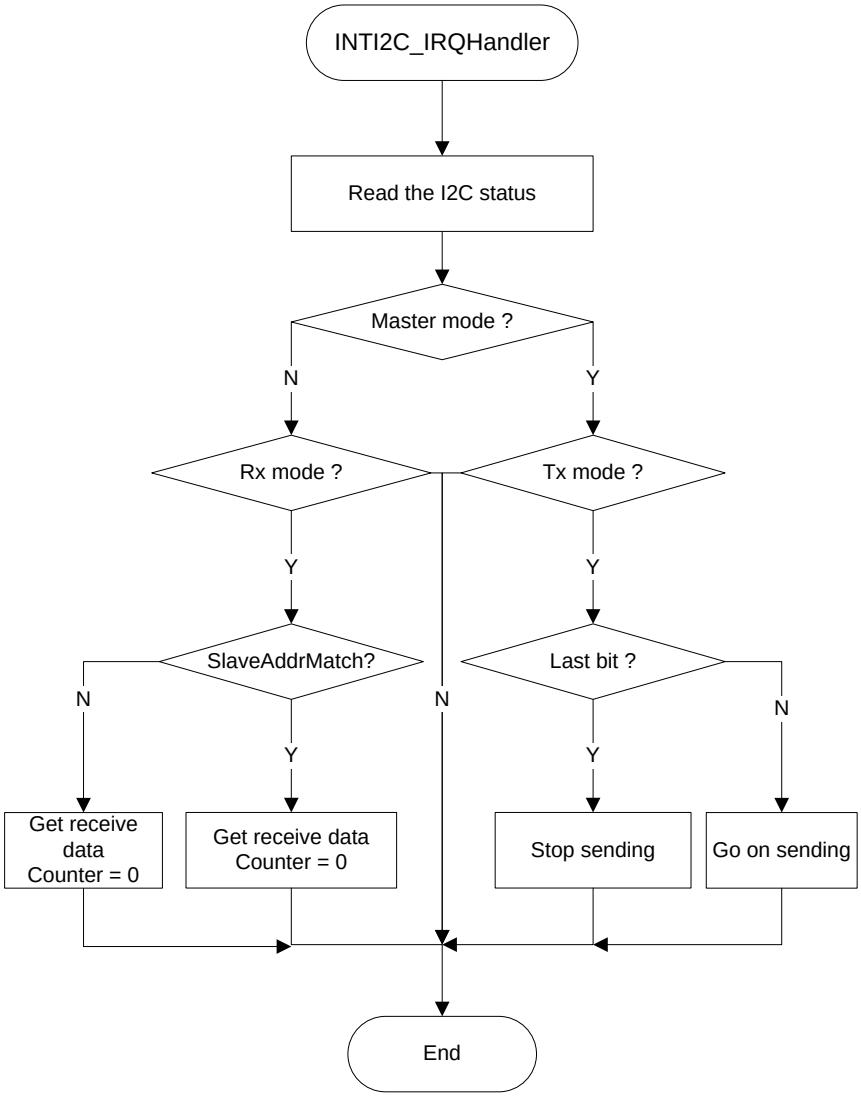
This is a simple example based on the TX04 Peripheral Driver (SBI, GPIO).

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 SBI I2C mode.

```
GPIO_EnableFuncReg(GPIO_PR, GPIO_FUNC_REG_1, GPIO_BIT_4);
GPIO_EnableFuncReg(GPIO_PR, GPIO_FUNC_REG_1, GPIO_BIT_5);
GPIO_SetOutputEnableReg(GPIO_PR, GPIO_BIT_4, ENABLE);
GPIO_SetOutputEnableReg(GPIO_PR, GPIO_BIT_5, ENABLE);
GPIO_SetInputEnableReg(GPIO_PR, GPIO_BIT_4, ENABLE);
GPIO_SetInputEnableReg(GPIO_PR, GPIO_BIT_5, ENABLE);
GPIO_SetOpenDrain(GPIO_PR, GPIO_BIT_4, ENABLE);
GPIO_SetOpenDrain(GPIO_PR, GPIO_BIT_5, ENABLE);
/* Pull up for SDA&SCL */
GPIO_SetPullUp(GPIO_PR, GPIO_BIT_4, ENABLE);
GPIO_SetPullUp(GPIO_PR, GPIO_BIT_5, ENABLE);
```

Then enable, initialize and configure SBI Master/Slave mode and enable INTI2C.

```
/* Master mode */
myI2Cm.I2CSelfAddr = SELF_ADDR;
myI2Cm.I2CDataLen = SBI_I2C_DATA_LEN_8;
myI2Cm.I2CACKState = ENABLE;
myI2Cm.I2CClkDiv = SBI_I2C_CLK_DIV_328;
/* Slave mode */
myI2Cs.I2CSelfAddr = SLAVE_ADDR;
myI2Cs.I2CDataLen = SBI_I2C_DATA_LEN_8;
myI2Cs.I2CACKState = ENABLE;
myI2Cs.I2CClkDiv = SBI_I2C_CLK_DIV_328;
gSBIMode = MODE_SBI_I2C_IDLE;

NVIC_EnableIRQ(INTI2C_IRQn);
```

After the above setting, start I2C send.

Initialize master mode. Set SBI Tx buffer and length. And clear Rx buffer.

```
case MODE_SBI_MASTER_INITIAL:
        /* Initialize SBI channel to Master mode */
        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';
        SBI_Enable(TSB_SBI0);
        SBI_SWReset(TSB_SBI0);
        SBI_InitI2C(TSB_SBI0, &myI2Cm);
        gSBIMode = MODE_SBI_I2C_START;
        break;
```

Initialize slave mode. Clear Rx buffer.

```
case MODE_SBI_ SLAVE_INITIAL:
     /* Initialize SBI channel to Slave mode */
     gI2CWCnt = 0U;
     for (gICnt = 0U; gICnt < 8U; gICnt++) {
          gI2CRxData[gICnt] = 0U;
      }
     SBI_Enable(TSB_SBI0);
     SBI_SWReset(TSB_SBI0);
```

```
            SBI_InitI2C(TSB_SBI0, &myI2Cs);
            gSBIMode = MODE_SBI_I2C_ RX;
            break;
```

Check if the I2C bus is free or not, SBI_SetSendData() is used to set data (SLAVE_ADDR).and direction (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_TX;
               LED_On(LED0);
        } else {
              /* Do nothing */
        }
        break;
```

Read gI2CRCnt to check Rx end.and then goto MODE_SBI_SLAVE_FINISHED;

```
case MODE_SBI_I2C_RX:
        LED_On(LED4);
        if (gI2CRCnt > gI2CRxData[0]) {
              gI2CRxDataLen = gI2CRxData[0];
              gI2CRCnt = 0U;
              gSBIMode = MODE_SBI_SLAVE_FINISHED;
         } else {
            /* Do nothing */
         }
         break;
```

Read gI2CWCn to check Tx end.and then goto MODE_SBI_MASTER_FINISHED;

```
case MODE_SBI_I2C_TX:
      LED_On(LED1);
      if (gI2CWCnt > gI2CTxData[0]) {
               gSBIMode = MODE_SBI_MASTER_FINISHED;
      } else {
               /* Do nothing */
      }
      break;
```

The data transfer or receive is handled in INTI2C.

In INTI2C 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.

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

In INTI2C 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.

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

7-16 TMRB

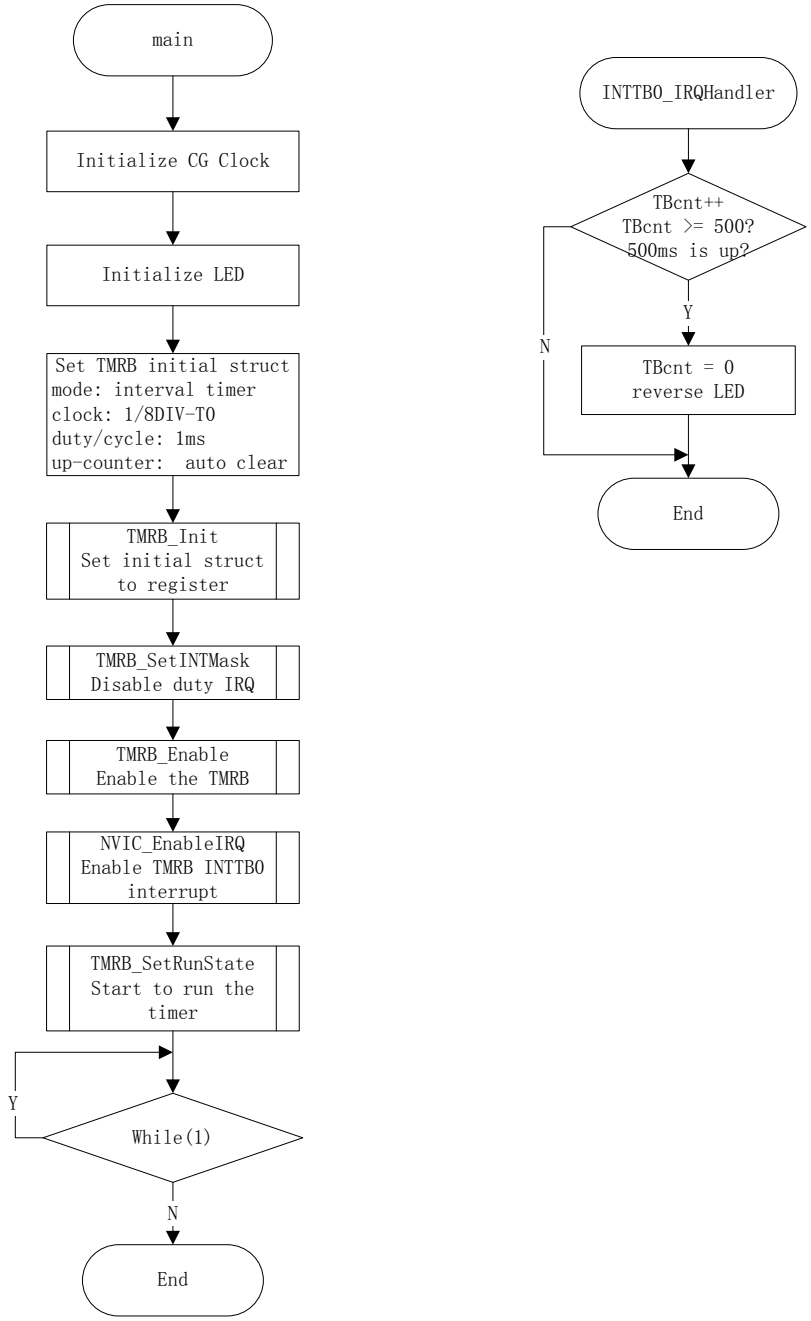
Example: General Timer

This is a simple example based on the TX04 Peripheral Driver (TMRB, GPIO).

The example includes:

1. TMRB0 initialization
2. General Timer for 1ms

- **Flow Chart**



- **Code and Explanation for the Example**

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

```
LED_Init();                          /* LED initialize */
Led_off(LED0 | LED1 | LED2 | LED3);
```

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 0x1770, because fphiT0 = fsys = fc = 10MHz * PLL* = 100MHz, ftmrb = 1/8 fphiT0 = 12MHz, Ttmrb = 0.08us, 1ms/0.08us = 12500 = 0x30D4 (Please see CG part for more detail information about the clock setting)

```
TMRB_InitTypeDef m_tmrb;
m_tmrb.Mode = TMRB_INTERVAL_TIMER;       /* internal timer */
```

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

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

```
TMRB_SetINTMask(TSB_TB0,TMRB_MASK_MATCH_LEADINGTIMING_INT);/*leadi
ngtiming is not used in TMRB_INTERVAL_TIMER mode */
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.

```
tbcount++;
if (tbcount >= 500U) {                                  /* 500ms is up */
      tbcount = 0U;
      /* reverse LED output */
      ledon = (ledon == 0U) ? 1U : 0U;
      if (0U == ledon) {
            Led_Off(LED0);
      } else {
            Led_On(LED0);
      }
} else {
      /* Do nothing */
}
```

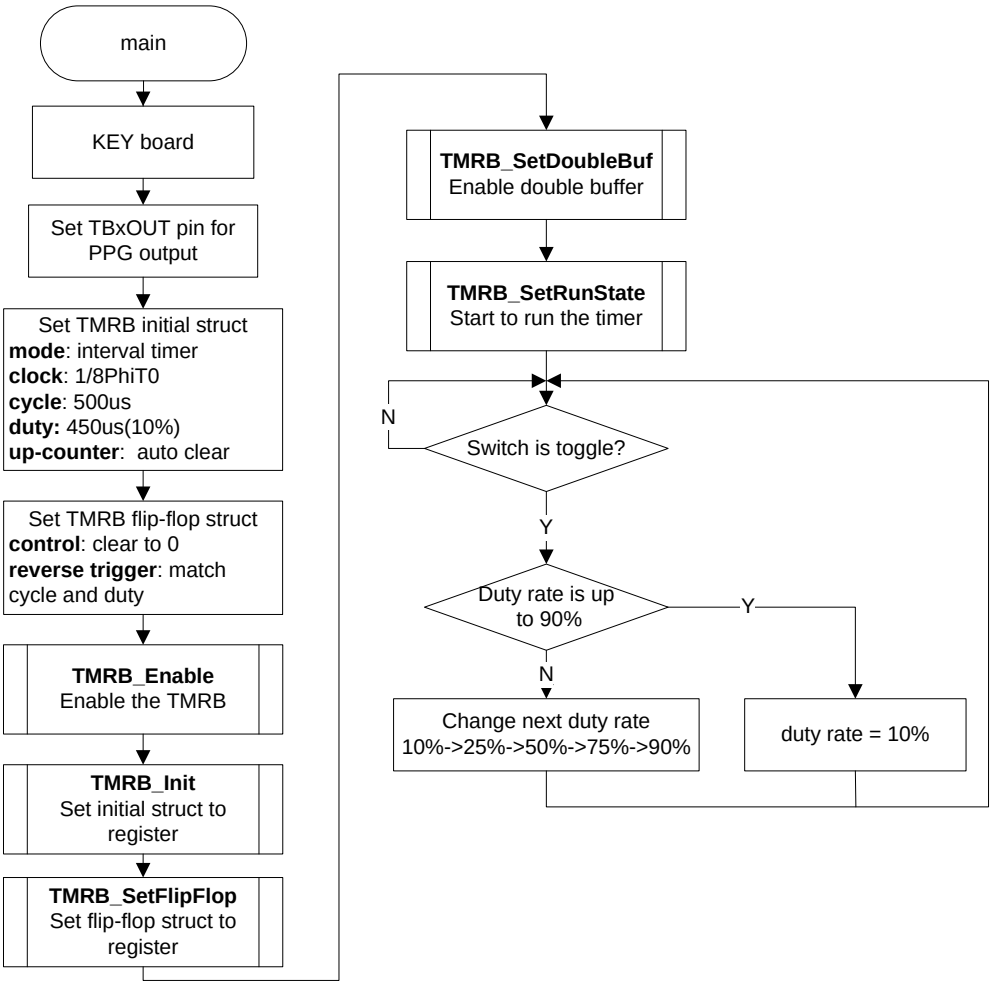
Example: PPG Output

This is a simple example based on the TX04 Peripheral Driver (TMRB, GPIO).

The example includes:

1. TMRB12 initialization
2. PPG process setting and start
3. PPG leadingtiming adjustment

- **Flow Chart**



- **Code and Explanation for the Example**

At first, set PW4 as TB12OUT for PPG output.

```
TSB_PW->CR |= 0x10;
TSB_PW->FR1 |= 0X10;
TSB_PW->IE &= 0;                    /* set KEY port to input */
```

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 TMRB4TIME equals 0x0BB8, because fphiT0 = fsys = fc = 10MHz * PLL * = 100MHz, ftmrb = 1/8 fphiT0 = 12MHz, Ttmrb = 0.08us, 500us/0.08us = 6250 = 0x186A (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 = TMRB4TIME;                  /* trailingtiming is 500us */
m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;     /* up-counter auto clear */
m_tmrb.Leadingtiming  =  Leadingtiming[Rate];                   /*  leadingtiming,
```

```
initial value 10% */
```

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

```
PPGFFInital.FlipflopCtrl = TMRB_FLIPFLOP_CLEAR;
PPGFFInital.FlipflopReverseTrg=TMRB_FLIPFLOP_MATCH_TRAILINGTIMING |
                                TMRB_FLIPFLOP_MATCH_LEADINGTIMING;
```

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

```
TMRB_Enable(TSB_TB12);
TMRB_Init(TSB_TB12, &m_tmrb);
TMRB_SetFlipFlop(TSB_TB12, &PPGFFInital);
TMRB_SetDoubleBuf(TSB_TB12, ENABLE);     /* enable double buffer */
TMRB_SetRunState(TSB_TB12, TMRB_RUN);
```

Wait switch change status, and at the same time.

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

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

```
if (changestatus == 1) {
      Rate++;                   /* change leadingtiming rate */
      if (Rate >= LEADINGTIMINGMAX) {
            Rate = LEADINGTIMINGINIT;
      } else {
            /* Do nothing */
      }
      TMRB_ChangeLeadingtiming(TSB_TB12, Leadingtiming[Rate]);
      changestatus = 0;
} else {
      /* Do nothing */
}
```

The calculation method of leadingtiming value:

TrailingTiming = 500us, ftmrb = 1/8 fphiT0 = 12.5MHz, Ttmrb = 0.08us (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 = 0x15F9U
Leadingtiming = 25%: high-level time is 500*25% = 125us, low-level time is 500-125 = 375us, counter value = 375us/Ttmrb = 0x124FU

Leadingtiming = 50%: high-level time is 500*50% = 250us, low-level time is 500-250 = 250us, counter value = 250us/Ttmrb = 0xC35U

Leadingtiming = 75%: high-level time is 500*75% = 375us, low-level time is 500-375 = 125us, counter value = 125us/Ttmrb = 0x61AU

Leadingtiming = 90%: high-level time is 500*90% = 450us, low-level time is 500-450 = 50us, counter value = 50us/Ttmrb = 0x271U

That is the calculation method of leadingtiming array

```
uint32_t Leadingtiming[5] = { 0x15F9U, 0x124FU, 0Xc35U, 0x61AU, 0x271U};
/* leadingtiming: 10%, 25%, 50%, 75%, 90% */
```

7-17 TMRC

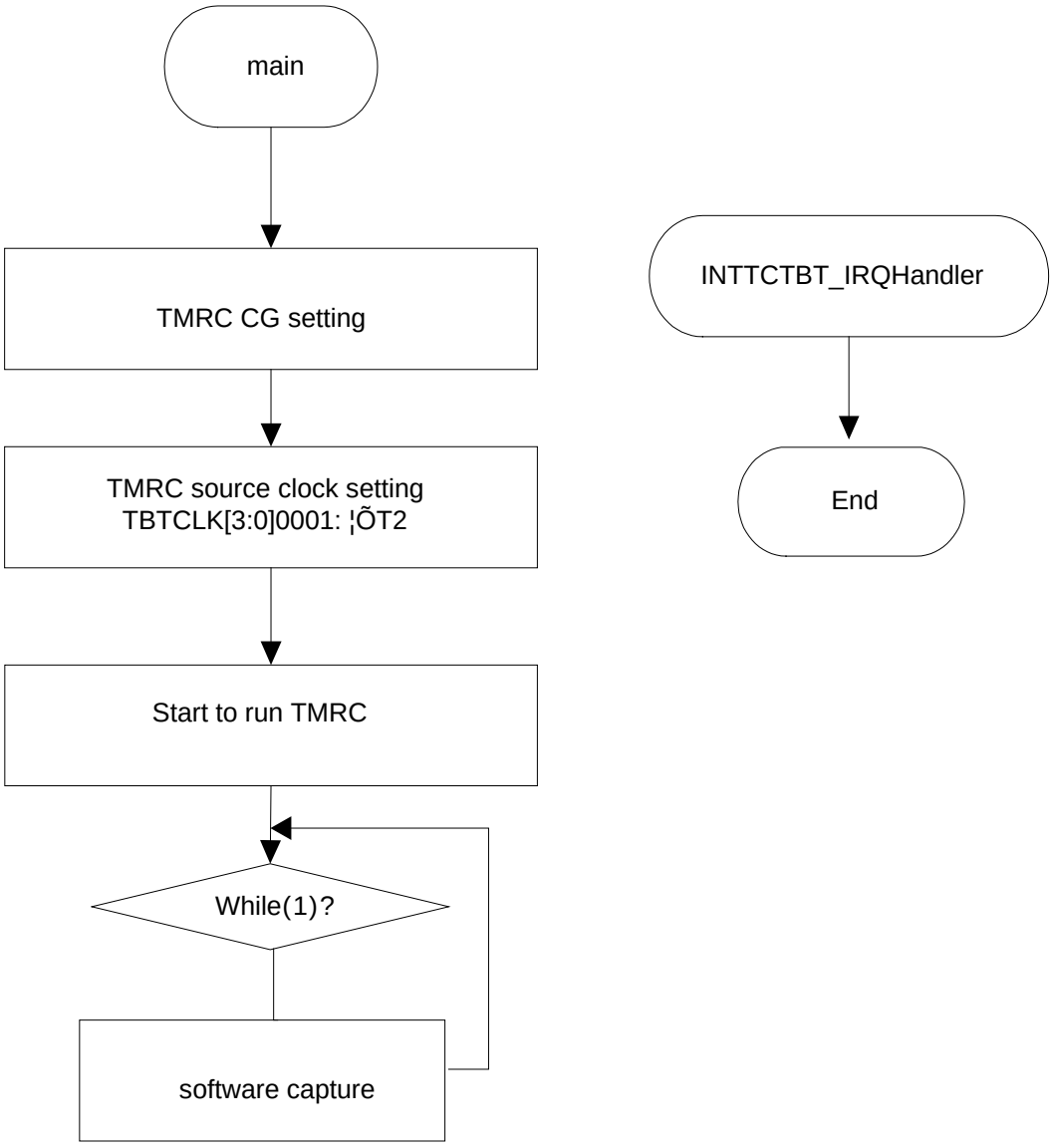
Example: TMRC Counter

This is a simple example based on the TX04 Peripheral Driver (TMRC).

The example includes:

a. TMRC initialization
b. Software capture

- **Flow Chart**



- **Code and Explanation for the Example**

At first, TMRC CG setting.

fosc = 10MHz(external high-speed oscillator)

fpll = 10 * fosc = 100MHz(in this demo, PLL multiplying value is set to 10 multiplying)

fperiph= fc = fpll = 100MHz

φT0(prescaler clock) = fperiph(Prescaler clock selection)=100MHz

φT2(source clock) = φT0/4 = 25MHz

```
CG_SetPhiT0Src(CG_PHIT0_SRC_FC);
CG_SetPhiT0Level(CG_DIVIDE_1);
CG_SetFcPeriphB(FC_B_ALL, ENABLE);
```

Enable the TMRB module ,source clock setting TBTCLK[3:0]0001: φT2*,enable interrupt,and in the end start the TMRC to run.

```
TMRC_Enable(TSB_TC);
TMRC_SetIdleMode(TSB_TC, ENABLE);
TMRC_SetSrcClk(TSB_TC, TMRC_CLK_DIV_4);
NVIC_EnableIRQ(INTTCTBT_IRQn);
TMRC_SetRunState(TSB_TC, TMRC_RUN);
```

Then the main routine will enter “While(1)” ,when counter value overflow. Overflow interrupt INTTCTBT will be occurred and the counter value will be cleared to "0" then start up counting.

```
void INTTCTBT_IRQHandler(void)
{
/* overflow interrupt INTTCTBT occured and the counter value will be
cleared to "0" then start up counting*/

}
```

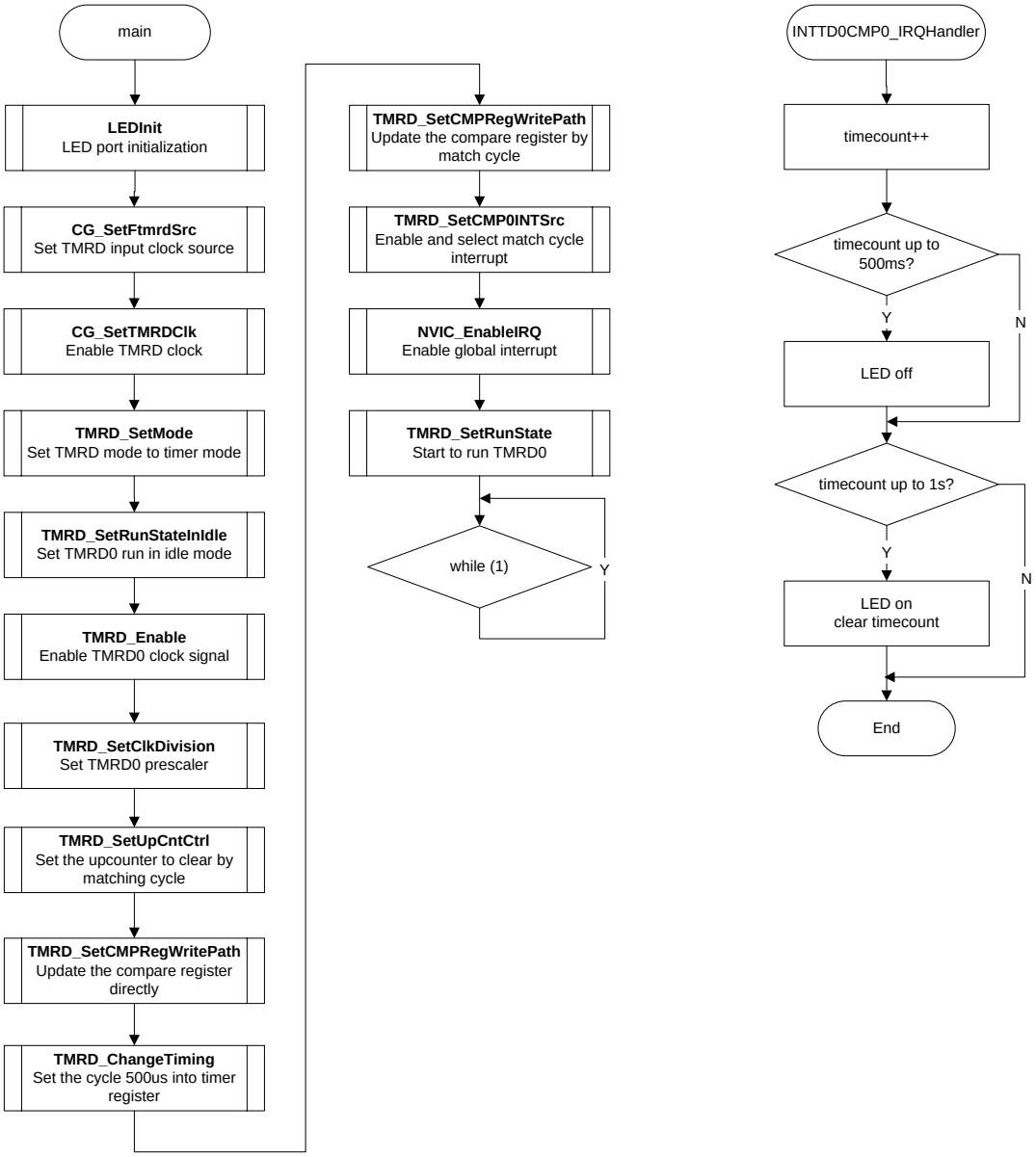
7-18 TMRD

Example: Interval timer

This is a simple example based on the TX04 Peripheral Driver (TMRD, CG).

The example includes:
1.TMRD interval timer mode.
2.TMRD initialization and operation process.
3.TMRD interrupt handling.

- **Flow Chart**



- **Code and Explanation for the Example**

At first, initialize the LED port on evaluation board, and turn on the LED.

```
LED_Init();
```

Configure TMRD clock by using API CG_SetFtmrdSrc() and enable the clock source by using CG_SetTMRDClk().

```
/* TMRD CG setting */
CG_SetFtmrdSrc(CG_TMRD_UNIT_A,
                    CG_FTMRD_SRC_HALF_FC);     /* clock is set to fc/2 */
CG_SetTMRDClk(CG_TMRD_UNIT_A, ENABLE);      /* enable the TMRDACLK */
```

Set TMRD mode by using TMRD_SetMode(). In this demo, both TMRD0 and TMRD1 are set to timer mode.

```
/* TMRD mode setting */
TMRD_SetMode(TSB_TD, TMRD_MODE_BOTH_TMR);
```

Set TMRD0 to run in idle mode by using API TMRD_SetRunStateInIdle().

```
/* TMRD IDLE mode */
TMRD_SetRunStateInIdle(TSB_TD, TMRD_UNIT_CH_0, TMRD_RUN);
```

Then enable TMRD0 clock signal and set the prescaler by using API TMRD_Enable() and TMRD_SetClkDivision().

```
/* TMRD enable clock signal */
TMRD_Enable(TSB_TD, TMRD_UNIT_CH_0);

/* Set the TMRD prescaler */
TMRD_SetClkDivision(TSB_TD, TMRD_UNIT_CH_0, TMRD_CLK_DIV_1);
```

Set the TMRD0A up-counter to clear when match the trailingtiming (auto clear).

```
/* Set the upcounter clear mode */
TMRD_SetUpCntCtrl(TSB_TDA, TMRD_UNIT_CH_0, TMRD_AUTO_CLEAR);
```

After using API TMRD_SetCMPRegWritePath(…, TMRD_CMP_WRITE_DIRECT) to set to update compare register directly, the same data is written to the corresponding compare register when data is written to the timer register by using API TMRD_ChangeTiming().

The calculation method of macro TIME_500US is:
fosc = 10MHz(external high-speed oscillator)
fc = 10 * fosc = 100MHz(in this demo, PLL multiplying value is set to 10 multiplying)
ftmrd = fc / 2 = 50MHz(in this demo, ftmrd = fc/ 2, please see CG_SetFtmrdSrc())
So the value of TIME_500US is 500us*ftmrd = 25000 = 0x61A8

```
/* Update the compare register directly */
TMRD_SetCMPRegWritePath(TSB_TD,
                                    TMRD_UNIT_CH_0,
                                    TMRD_CMP_WRITE_DIRECT);

/* Set the value to timer register */
TMRD_ChangeTiming(TSB_TD, TMRD_TIMING_TD0_TRAILINGTIMING,
                                TIME_500US);                    /* 500us */
```

Next, using API TMRD_SetCMPRegWritePath(…, TMRD_CMP_WRITE_INDIRECT) to enable writing data through the timer register to the compare register when up-counter matching the trailingtiming time.

```
/* Indirct update the compare register */
    TMRD_SetCMPRegWritePath(TSB_TD,
                                    TMRD_UNIT_CH_0,
                                    TMRD_CMP_WRITE_INDIRECT);
```

Then choose the compare 0 interrupt source to match trailingtiming by using API TMRD_SetCMP0INTSrc(), and enable the compare 0 interrupt by using NVIC_EnableIRQ().

```
/* Enable TMRD intterrupt */
TMRD_SetCMP0INTSrc(TSB_TD,
                   TMRD_UNIT_CH_0,
                   TMRD_INT_MATCH_TRAILINGTIMING);

NVIC_EnableIRQ(INTTD0CMP0_IRQn);
```

In the end, start the TMRD0 to run by using TMRD_SetRunState() to run the counter and enter into a dead loop.

```
/* Start to run TMRD */
TMRD_SetRunState(TSB_TD, TMRD_UNIT_CH_0, TMRD_RUN);
```

When 500us is up, the compare 0 interrupt occurs. In IRQ INTTD0CMP0_IRQHandler(), use a variable *timecount* to accumulate. If 500ms is up, turn off the LED. And if 1s is up, turn on the LED again and clear the *timecount* to accumulate from the start.

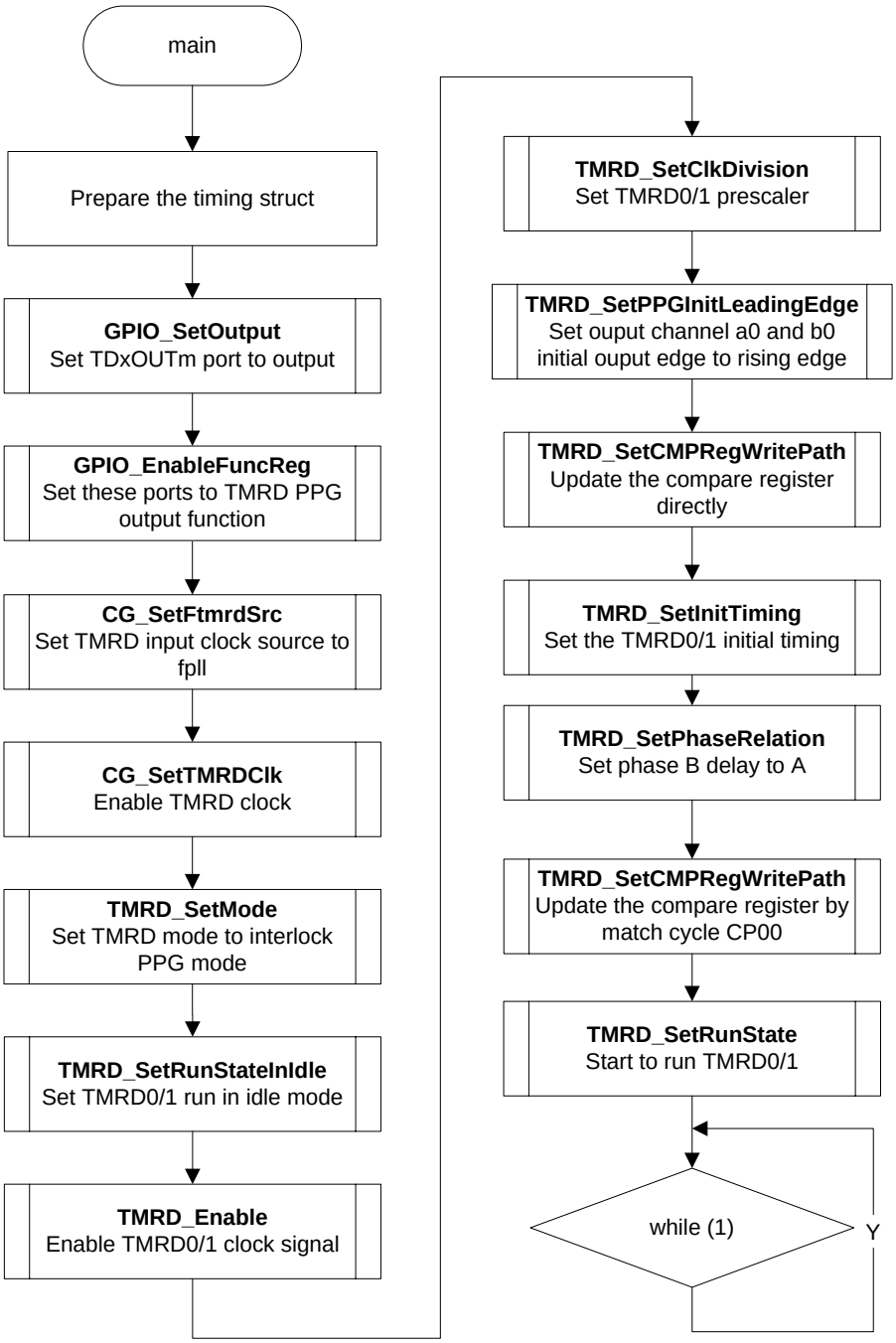
```
void INTTDA0CMP0_IRQHandler(void)
{
    static uint16_t timecount = 0U;

    timecount++;
    if (timecount == 1000U) {     /* 500ms */
        LED_Off(LED_ALL);
    } else if (timecount == 2000U) {       /* 1s */
        timecount = 0U;_
        LED_On(LED_ALL);
    } else {
        /* Do nothing */
    }
}
```

Example: Interlock PPG output

This is a simple example based on the TX04 Peripheral Driver (TMRD, CG, GPIO).

The example includes:
1. TMRD interlock PPG mode.
2. TMRD initialization and operation process.
3. TMRD PPG mode parameters calculation method.

- **Flow Chart**


main
Prepare the timing struct
GPIO_SetOutput
Set TDxOUTm port to output
GPIO_EnableFuncReg
Set these ports to TMRD PPG output function
CG_SetFtmrdSrc
Set TMRD input clock source to fpll
CG_SetTMRDClk
Enable TMRD clock
TMRD_SetMode
Set TMRD mode to interlock PPG mode
TMRD_SetRunStateInIdle
Set TMRD0/1 run in idle mode
TMRD_Enable
Enable TMRD0/1 clock signal
TMRD_SetClkDivision
Set TMRD0/1 prescaler
TMRD_SetPPGInitLeadingEdge
Set ouput channel a0 and b0 initial ouput edge to rising edge
TMRD_SetCMPRegWritePath
Update the compare register directly
TMRD_SetInitTiming
Set the TMRD0/1 initial timing
TMRD_SetPhaseRelation
Set phase B delay to A
TMRD_SetCMPRegWritePath
Update the compare register by match cycle CP00
TMRD_SetRunState
Start to run TMRD0/1
while (1)
Y

- **Code and Explanation for the Example**

At first, initialize the TMRD timing parameters structure. The calculation method is:

TrailingTiming is 500us:
fosc = 10MHz(external high-speed oscillator)
fpll = 10 * fosc = 100MHz(in this demo, PLL multiplying value is set to 10 multiplying)
ftmrd = fpll / 2 = 50MHz(in this demo, ftmrd = fpll / 2, please see CG_SetFtmrdSrc())
So the value is 500us*ftmrd = 25000 = 0x61A8U

Duty is 50%:
Set LeadingTiming0 to 0x0000, then TrailingTiming0 is TrailingTiming/2 = 0x30D4U (250us)
LeadingTiming1 and TrailingTiming1 are not used, so keep their value to 0x0000.

Phase shift is 120 degree:
The phase shift can be calculated by using the formula:

θ = 360 degree*(PhaseShiftTiming/(TrailingTiming+1))

so in this demo, PhaseShiftTiming should be set to TrailingTiming/3 = 0x208DU (167us)

```
#define TIME_CYCLE              ((uint16_t)0x61A8U)     /* 500us */
#define TIME_DUTY               ((uint16_t)0x30D4U)     /* 250us */
#define TIME_PHASE_SHIFT        ((uint16_t)0x208DU)     /* 500/3= 167us */

TMRD_TimingTypeDef timestruct = { 0U };
timestruct.Cycle = TIME_CYCLE;
timestruct.TrailingTiming0 = TIME_DUTY;
timestruct.PhaseShiftTiming = TIME_PHASE_SHIFT;
```

Configure the TDxAOUTm port to TMRD PPG output. In this demo, these ports are port Y 4,5,6,7.

```
/* TD0OUT0, TD0OUT1, TD1OUT0, TD1OUT1 port setting */
GPIO_SetOutput(GPIO_PY,
                GPIO_BIT_4 | GPIO_BIT_5 | GPIO_BIT_6 | GPIO_BIT_7);
GPIO_EnableFuncReg(GPIO_PA,
                        GPIO_FUNC_REG_1,
                        GPIO_BIT_4 | GPIO_BIT_5 | GPIO_BIT_6 | GPIO_BIT_7);
```

Configure TMRD clock by using API CG_SetFtmrdSrc() and enable the clock source by using CG_SetTMRDClk().

```
/* TMRD CG setting */
CG_SetFtmrdSrc(CG_TMRD_UNIT_A,
                CG_FTMRD_SRC_HALF_FC);      /* clock is set to fc/2 */
CG_SetTMRDClk(CG_TMRD_UNIT_A, ENABLE);      /* enable the TMRDACLK */
```

Set TMRD mode by using TMRD_SetMode(). In this demo, use interlock PPG mode.

```
/* TMRD mode setting */
TMRD_SetMode(TSB_TD, TMRD_MODE_INTERLOCK_PPG_3CH);
```

Set TMRD0A/1A to run in idle mode by using API TMRD_SetRunStateInIdle().

```
/* TMRD IDLE mode */
TMRD_SetRunStateInIdle(TSB_TD, TMRD_UNIT_CH_0, TMRD_RUN);
TMRD_SetRunStateInIdle(TSB_TD, TMRD_UNIT_CH_1, TMRD_RUN);
```

Then enable TMRD clock signal and set the prescaler by using API TMRD_Enable() and TMRD_SetClkDivision(). Note that, in interlock PPG mode, the prescaler of TMRD1A is the same as the value setting of TMRD0A. So only setting TMRD0 prescaler is enough.

```
/* TMRD enable clock signal */
TMRD_Enable(TSB_TDA, TMRD_UNIT_CH_0);
```

```
TMRD_Enable(TSB_TDA, TMRD_UNIT_CH_1);
/* Set the TMRD prescaler */
TMRD_SetClkDivision(TSB_TDA, TMRD_UNIT_CH_0, TMRD_CLK_DIV_1);
```

Set the PPG output initial wave edge, both channel a0 and b0 are set to rising edge by using API TMRD_SetPPGInitLeadingEdge().

```
/* Set the PPG output edge */
TMRD_SetPPGInitLeadingEdge(TSB_TD,
                           TMRD_PPG_CHANNEL_A0,
                           TMRD_WAVE_EDGE_RISING);
TMRD_SetPPGInitLeadingEdge(TSB_TD,
                           TMRD_PPG_CHANNEL_B0,
                           TMRD_WAVE_EDGE_RISING);
```

After using API TMRD_SetCMPRegWritePath(…, TMRD_CMP_WRITE_DIRECT) to set to update compare register directly, the same data is written to the corresponding compare register when data is written to the timer register by using API TMRD_SetInitTiming ().The timing structure has been prepared at the beginning of this chapter.

```
/* Direct update the compare register */
TMRD_SetCMPRegWritePath(TSB_TD,
                        TMRD_UNIT_CH_0,
                        TMRD_CMP_WRITE_DIRECT);
TMRD_SetCMPRegWritePath(TSB_TD,
                        TMRD_UNIT_CH_1,
                        TMRD_CMP_WRITE_DIRECT);

/* Set the value to timer register */
TMRD_SetInitTiming(TSB_TD, TMRD_UNIT_CH_0, &timestruct);
TMRD_SetInitTiming(TSB_TD, TMRD_UNIT_CH_1, &timestruct);
```

Then setting the phase relation between phase A and B by using API TMRD_SetPhaseRelation(), and in this demo, phase B is delay to phase A.

```
/* Set the relation of phase A and B */
TMRD_SetPhaseRelation(TSB_TD, TMRD_PHASE_DELAY_OR_SAME);
```

Next, using API TMRD_SetCMPRegWritePath(…, TMRD_CMP_WRITE_INDIRECT) to enable writing data through the timer register to the compare register when up-counter matching the trailingtiming time.

```
/* Indirct update the compare register */
TMRD_SetCMPRegWritePath(TSB_TD,
                        TMRD_UNIT_CH_0,
                        TMRD_CMP_WRITE_INDIRECT);
TMRD_SetCMPRegWritePath(TSB_TD,
                        TMRD_UNIT_CH_1,
                        TMRD_CMP_WRITE_INDIRECT);
```

In the end, start the TMRD0A/1A to run by using TMRD_SetRunState() to run the TMRD0 and enter to a dead loop. In interlock PPG mode, TMRD1A will start to operation in tandem with COUNTER0 of TMRD0A, setting TMRD1A to run by TMRD_SetRunState() becomes invalid, so there is no need to set TMRD1A to run.

```
/* Start to run TMRD */_
TMRD_SetRunState(TSB_TD, TMRD_UNIT_CH_0, TMRD_RUN);
```

7-19 SIO/UART

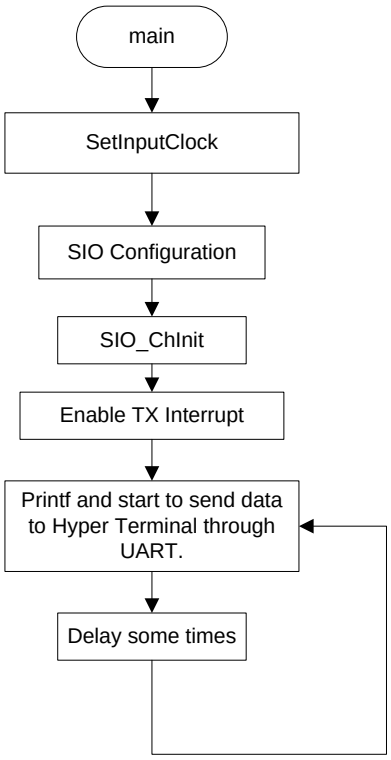
Example: Retarget

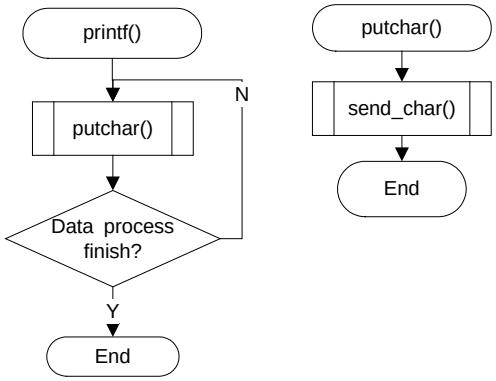
This is a simple example based on the TX04 Peripheral Driver (UART, GPIO).

The example includes:

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

- **Flowchart**





- **Code and Explanation for the Example**

At first configure the GPIO and initialize UART.
Set the inputclock

```
UART_SetInputClock(UART0,0x01);
```

Use GPIO peripheral drivers configure GPIO for UART.

```
TSB_RETG->FR1 |= RXD_BIT | TXD_BIT;
TSB_RETG->CR |= TXD_BIT;
TSB_RETG->IE |= RXD_BIT;
TSB_RETG->PUP |= RXD_BIT | TXD_BIT;
```

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

```
UART_InitTypeDef myUART;

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

After above setting, enable UART TX interrupt.

```
NVIC_EnableIRQ(RETARGET_INT)
```

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

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

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

```
void INTTX0_IRQHandler(void)
{
    if (gSIORdIndex < gSIOWrIndex) {       /* buffer is not empty */
        UART_SetTxData(UART_RETARGET,  gSIOTxBuffer[gSIORdIndex++]);          /* send data */
        fSIO_INT = SET;              /* SIO0 INT is enable */
    } else {
        /* disable SIO0 INT */
        fSIO_INT = CLEAR;
        NVIC_DisableIRQ(RETARGET_INT);
        fSIOTxOK = YES;
```

```
    }
    if (gSIORdIndex >= gSIOWrIndex) {    /* reset buffer index */
        gSIOWrIndex = CLEAR;
        gSIORdIndex = CLEAR;
    } else {
        /* Do nothing */
    }
}
```

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

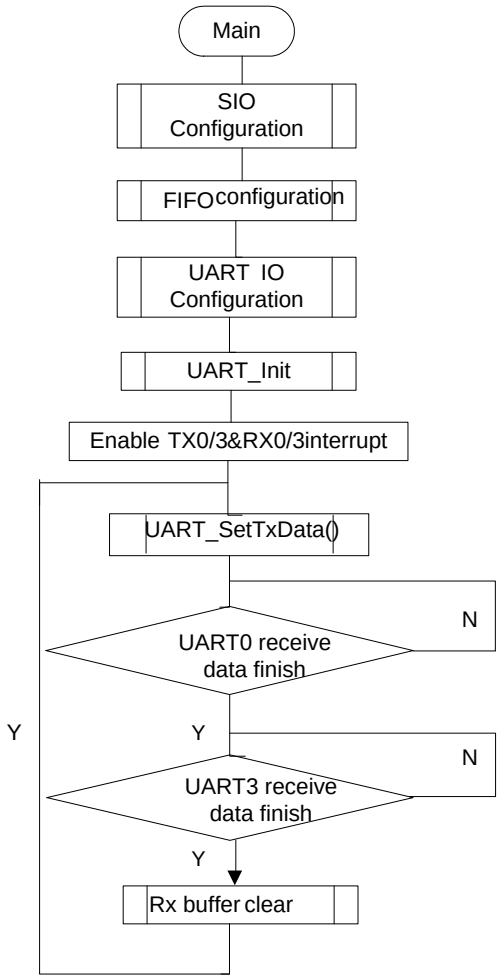
```
#if defined ( __CC_ARM    )          /* RealView Compiler */
struct __FILE {
    int handle;                      /* Add whatever you need here */
};
FILE __stdout;
FILE __stdin;
int fputc(int ch, FILE * f)
#elif defined ( __ICCARM__ )         /*IAR Compiler */
int putchar(int ch)
#endif
{
    return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

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

    return ch;
}
```

Example: UART_FIFO_Demo

This is a simple example based on the TX04 Peripheral Driver (UART, GPIO).

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

- **Flowchart**



- **Code and Explanation for the Example**

At first configure the GPIO and initialize UART.

Use GPIO peripheral drivers configure GPIO for UART0 and UART3.

```
void SIO_Configuration(TSB_SC_TypeDef * SCx)
{
    if (SCx == TSB_SC0) {
        TSB_PH->CR |= GPIO_BIT_4;
        TSB_PH->FR1 |= GPIO_BIT_4;
        TSB_PH->FR1 |= GPIO_BIT_5;
        TSB_PH->IE |= GPIO_BIT_5;
    } else if (SCx == TSB_SC1) {
        TSB_PK->CR |= GPIO_BIT_4;
        TSB_PK->FR1 |= GPIO_BIT_4;
        TSB_PK->FR1 |= GPIO_BIT_5;
        TSB_PK->IE |= GPIO_BIT_5;
    } else if (SCx == TSB_SC2) {
        TSB_PE->CR |= GPIO_BIT_0;
        TSB_PE->FR2 |= GPIO_BIT_0;
        TSB_PE->FR2 |= GPIO_BIT_1;
        TSB_PE->IE |= GPIO_BIT_1;
    } else if (SCx == TSB_SC3) {
        TSB_PM->CR |= GPIO_BIT_4;
```

```
            TSB_PM->FR1 |= GPIO_BIT_4;
            TSB_PM->FR1 |= GPIO_BIT_5;
            TSB_PM->IE |= GPIO_BIT_5;
        }
}
```

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

```
    UART_InitTypeDef myUART;

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

Use UART peripheral drivers to enable and initialize UART0/3 channels.

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

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

FIFO configuration.

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

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

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

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

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

    UART_RxFIFOFillLevel(UART0,UART_FIFO_4B,UART_RXFIFO4B_FLEVLE_4_2B);
    UART_RxFIFOFillLevel(UART3,UART_FIFO_4B,UART_RXFIFO4B_FLEVLE_4_2B);

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

    UART_RxFIFOClear(UART0);
    UART_RxFIFOClear(UART3);

    UART_TxBufferClear(UART0);
    UART_TxBufferClear(UART3);

    UART_TxFIFOFillLevel(UART0,UART_FIFO_4B,UART_TXFIFO4B_FLEVLE_0_0B);
    UART_TxFIFOFillLevel(UART3,UART_FIFO_4B,UART_TXFIFO4B_FLEVLE_0_0B);

    UART_TxFIFOINTSel(UART0,UART_TFIS_REACH_NOREACH_FLEVEL);
```

```
    UART_TxFIFOINTSel(UART3,UART_TFIS_REACH_NOREACH_FLEVEL);

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

After above setting, enable UART0/3 interrupt.

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

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

The rest process of data flow is finished in ISR of UART0/3 RX and TX interrupt routine
UART0 TX interrupt routine:

```
void INTTX0_IRQHandler(void)
{
    volatile UART_Err err;

    if (TxCounter < NumToBeTx) {
        UART_SetTxData(UART0, TxBuffer[TxCounter++]);
    } else {
        err = UART_GetErrState(UART0);
    }
}
```

UART3 TX interrupt routine:

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

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

UART0 RX interrupt routine:

```
void INTRX0_IRQHandler(void)
{
    volatile UART_Err err;

    err = UART_GetErrState(UART0);
    if (UART_NO_ERR == err) {
        RxBuffer[RxCounter++] = (uint8_t) UART_GetRxData(UART0);
    }
```

UART3 RX interrupt routine:

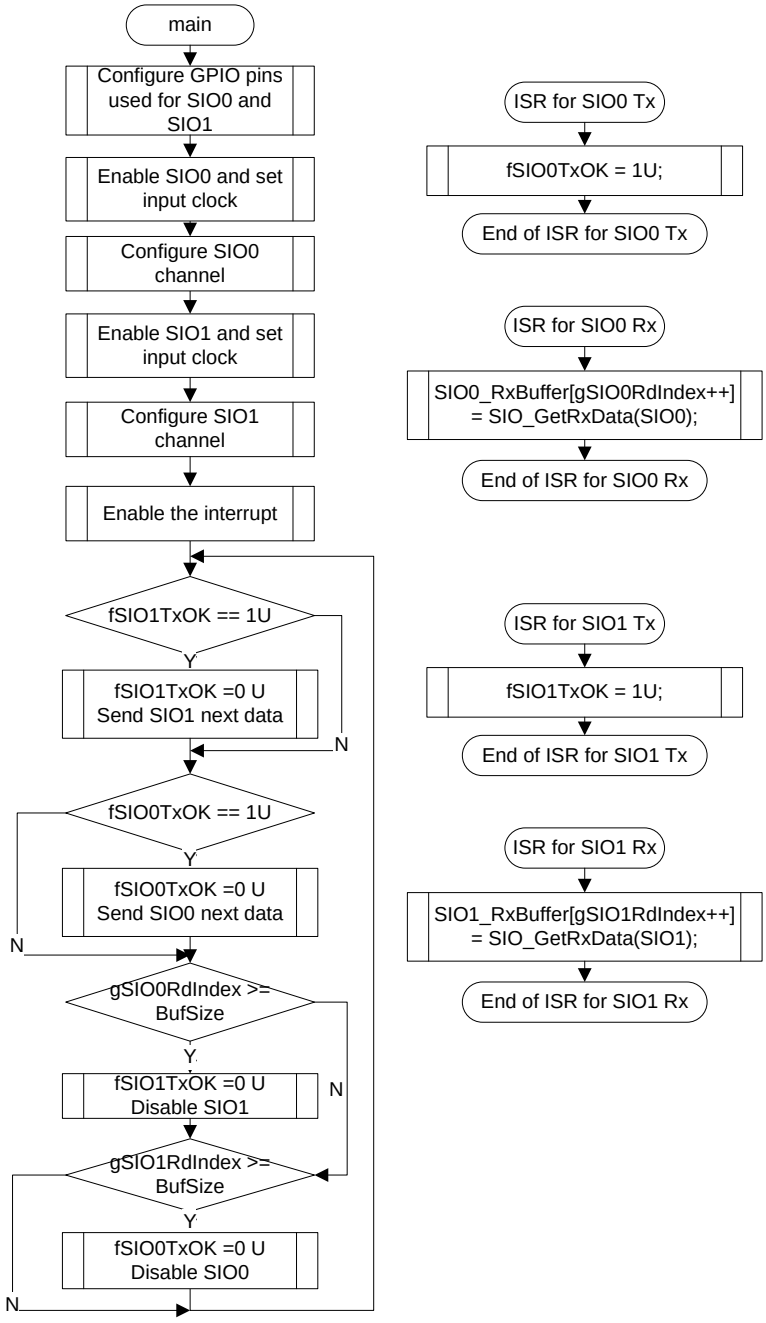
```
void INTRX3_IRQHandler(void)
{
    volatile UART_Err err;

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

Example: SIO Demo

This is a simple example based on the TX04 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);
/* Selects input clock for prescaler as PhiT0. */
SIO_SetInputClock(SIO0, SIO_CLOCK_T0);

/*initialize the SIO0 struct */
SIO0_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO0_Init.TIDLE = SIO_TIDLE_HIGH;
SIO0_Init.IntervalTime = SIO_SINT_TIME_SCLK_1;
SIO0_Init.TransferMode = SIO_TRANSFER_FULLDPX;
SIO0_Init.TransferDir = SIO_LSB_FRIST;
SIO0_Init.Mode = SIO_ENABLE_TX | SIO_ENABLE_RX;
SIO0_Init.DoubleBuffer = SIO_WBUF_ENABLE;
SIO0_Init.BaudRateClock = SIO_BR_CLOCK_TS2;
SIO0_Init.Divider = SIO_BR_DIVIDER_2;

SIO_Init(SIO0, SIO_CLK_BAUDRATE, &SIO0_Init);
```

Enable SIO1 configure the input clock and SIO1 initialization structure.

```
/*Enable the SIO1 channel */
SIO_Enable(SIO1);
/* Selects input clock for prescaler as PhiT0. */
SIO_SetInputClock(SIO1, SIO_CLOCK_T0);

/*initialize the SIO1 struct */
SIO1_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO1_Init.TIDLE = SIO_TIDLE_HIGH;
SIO1_Init.TXDEMP = SIO_TXDEMP_HIGH;
SIO1_Init.EHOLDTime = SIO_EHOLD_FC_64;
SIO1_Init.IntervalTime = SIO_SINT_TIME_SCLK_1;
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);
}
```

7-20 WDT

Example: WDT Demo

This is a simple example based on the TX04 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();
```

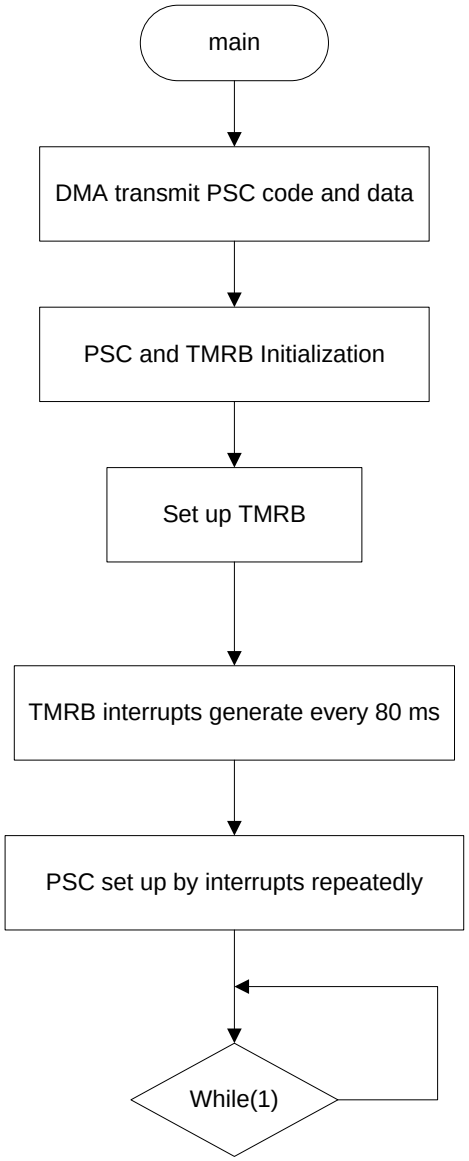
7-21 PSC

Example: PSC Repeat Proc

This is a simple example based on the TX04 Peripheral Driver (PSC, DMAC and TMRB).
This program transmits the code and data of PSC using DMAC.
The TMRB is operated by interrupt of 80 ms cycle.
The program of PSC is executed by interrupt of TMRB channel 9. And then, PSC set up execution repeatedly.

The example includes:
1. The code and data of PSC are linked to the specified ROM area by ICF file.
2. DMAC transmits the code and data from ROM to the PSC code ram and data ram.
3. The TMRB generates interrupts by 80 ms.
4. PSC program is repeatedly executed by TMRB interrupts.

- **Flowchart**



- **Code and Explanation for the Example**

The definition of a link file.

```
      /* PSC settings */
    define symbol __region_PSC_Code_ROM_start__    = 0x000BC000;
    define symbol __region_PSC_Code_ROM_end__      = 0x000BDFFF;
    define symbol __region_PSC_Data_ROM_start__    = 0x000BE000;
    define symbol __region_PSC_Data_ROM_end__      = 0x000BFFFF;

    define symbol __region_PSC_Code_RAM_start__    = 0x40010000;
    define symbol __region_PSC_Code_RAM_end__      = 0x40011FFF;
    define symbol __region_PSC_Data_RAM_start__    = 0x40014000;
    define symbol __region_PSC_Data_RAM_end__      = 0x40015FFF;

    define region ROM_PSC_Code_region = mem:[from __region_PSC_Code_ROM_start__ to __region_PSC_Code_ROM_end__];
    define region RAM_PSC_Code_region = mem:[from __region_PSC_Code_RAM_start__ to __region_PSC_Code_RAM_end__];
    define region ROM_PSC_Data_region = mem:[from __region_PSC_Data_ROM_start__ to __region_PSC_Data_ROM_end__];
```

```
    define region RAM_PSC_Data_region = mem:[from
__region_PSC_Data_RAM_start__ to __region_PSC_Data_RAM_end__];

    initialize manually { section PSC_CodeBinary };
    place in ROM_PSC_Code_region { section PSC_CodeBinary_init };
    place in RAM_PSC_Code_region { section PSC_CodeBinary };

    initialize manually { section PSC_DataBinary };
    place in ROM_PSC_Data_region { section PSC_DataBinary_init };
    place in RAM_PSC_Data_region { section PSC_DataBinary };
```

Setting the DMAC (Memory to Memory)
The code for PSC located at FLASH ROM is transmitted to an address 0x40010000.
Calculation of a CheckSum is also made at the same time.(gPSC_codeChksum)

```
    DMAC_InitTypeDef DMAC_InitStruct;
    DMAC_Channel myChannel = DMAC_CHANNEL_0;
    uint8_t     *pData;

    /* DMA channel3 configuration */
    DMAC_InitStruct.TxINT = ENABLE;
    DMAC_InitStruct.SrcAddr = (uint32_t) PSC_CODEROM;
    DMAC_InitStruct.SrcIncrementState = ENABLE;
    DMAC_InitStruct.SrcBitWidth = DMAC_WORD;
    DMAC_InitStruct.SrcBurstSize = DMAC_1_BEAT;
    DMAC_InitStruct.DstAddr = (uint32_t)PSC_CODEADDRSTA;
    DMAC_InitStruct.DstIncrementState = ENABLE;
    DMAC_InitStruct.DstBitWidth = DMAC_WORD;
    DMAC_InitStruct.DstBurstSize = DMAC_1_BEAT;
    DMAC_InitStruct.TxSize = PSC_CODETRANSSIZE;
    DMAC_InitStruct.TxDirection = DMAC_MEMORY_TO_MEMORY;

    /*Enable DMA circuit */
    DMAC_Enable(DMAC_UNIT_C);
    /* Set DMA endian */
    DMAC_SetTransferEndian(DMAC_UNIT_C, DMAC_LITTLE_ENDIAN);
    /*Initialize DMA channel */
    DMAC_Init(DMAC_UNIT_C, myChannel, &DMAC_InitStruct);
    DMAC_SetTxINTConfig(DMAC_UNIT_C, myChannel, DMAC_INT_TX_END,
ENABLE);

    pData = (uint8_t*)PSC_CODEADDRSTA;
    gPSC_codeChksum = 0x0000;                    /* clear checksum */

    DMAC_SetDMAChannel(DMAC_UNIT_C, myChannel, ENABLE);


    do {
        GetPSC_DestAddr = TSB_DMACC->C0DESTADDR + 0x03;
        if (pData < (uint8_t*)GetPSC_DestAddr) {
            gPSC_codeChksum += *pData;
            pData++;
        }
    }     while     ((DMAC_GetTxINTReq(DMAC_UNIT_C,     myChannel)     !=
DMAC_TX_END_REQ)  ||  (pData  <  (uint8_t*)PSC_CODEADDREND));     /*  Was
calculation of the checksum completed? */
```

```
    DMAC_ClearTxINTReq(DMAC_UNIT_C, myChannel, DMAC_INT_TX_END);
```

The data for PSC located at FLASH ROM is transmitted to an address 0x40014000.
Calculation of a CheckSum is also made at the same time.(gPSC_dataChksum)

```
    / /* DMA channel3 configuration */
    DMAC_InitStruct.TxINT = ENABLE;
    DMAC_InitStruct.SrcAddr = (uint32_t) PSC_DATAROM;
    DMAC_InitStruct.SrcIncrementState = ENABLE;
    DMAC_InitStruct.SrcBitWidth = DMAC_WORD;
    DMAC_InitStruct.SrcBurstSize = DMAC_1_BEAT;
    DMAC_InitStruct.DstAddr = (uint32_t)PSC_DATAADDRSTA;
    DMAC_InitStruct.DstIncrementState = ENABLE;
    DMAC_InitStruct.DstBitWidth = DMAC_WORD;
    DMAC_InitStruct.DstBurstSize = DMAC_1_BEAT;
    DMAC_InitStruct.TxSize = PSC_DATATRANSSIZE;
    DMAC_InitStruct.TxDirection = DMAC_MEMORY_TO_MEMORY;

    /*Enable DMA circuit */
    DMAC_Enable(DMAC_UNIT_C);
    /*Initialize DMA channel */
    DMAC_Init(DMAC_UNIT_C, myChannel, &DMAC_InitStruct);
    DMAC_SetTxINTConfig(DMAC_UNIT_C,  myChannel,  DMAC_INT_TX_END,
ENABLE);

    pData = (uint8_t*)PSC_DATAADDRSTA;
    gPSC_dataChksum = 0x0000;                          /* clear checksum */

    DMAC_SetDMAChannel(DMAC_UNIT_C, myChannel, ENABLE);

    do {
        GetPSC_DestAddr = TSB_DMACC->C0DESTADDR + 0x03;
        if (pData < (uint8_t*)GetPSC_DestAddr) {
            gPSC_dataChksum += *pData;
            pData++;
        }
    }    while    ((DMAC_GetTxINTReq(DMAC_UNIT_C,    myChannel)    !=
DMAC_TX_END_REQ)  ||  (pData  <  (uint8_t*)PSC_DATAADDREND));    /*  Was
calculation of the checksum completed? */

    DMAC_ClearTxINTReq(DMAC_UNIT_C, myChannel, DMAC_INT_TX_END);
```

Setting the TMRB.
Cycle of interrupt is 80 ms.

```
    TMRB_InitTypeDef m_tmrb;

    m_tmrb.Mode = TMRB_INTERVAL_TIMER;
    m_tmrb.ClkDiv = TMRB_CLK_DIV_128;
     /* periodic time is 80 ms */
     m_tmrb.TrailingTiming = SAMPLINGTIME;
     m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;
     /* periodic time is 80 ms */
     m_tmrb.LeadingTiming = SAMPLINGTIME;

    TMRB_Enable(TSB_TB9);
    TMRB_Init(TSB_TB9, &m_tmrb);
    NVIC_EnableIRQ(INTTB09_IRQn);
```

TMRB interrupt function
The function clears the startup event tag caused by TMRB interrupt 9.

```
void INTTB09_IRQHandler(void)
{
    PSC_ClearEventOverrunFlag(PSC_TRIGGER_EVENT_0);
}
```

Setting the PSC
Enable the Repeat execution.

```
PSC_SetRepetProcVectPointer(PSC_STARTADDR);
PSC_SetTriggerEventFUNC(PSC_TRIGGER_EVENT_0, ENABLE);
```

Start of TMRB

```
TMRB_SetRunState(TSB_TB9, TMRB_RUN);
```

The program of PSC
Definition of the data.

```
dsec1 section romdata abs=0x40114000
data1   dd   0x12340000
;
dsec2 section romdata abs=0x40114010
data2   dd   0x00005678
```

Set register A0 to 0x12340000, and set register R0 to 0x00005678. A0 plus R0, and the result is pushed to M0.

```
csec1 section code abs=0x40110000
    mvi     AP0,data1
    ld      A0,(AP0)
    mvi     AP0,data2
    ld      R0,(AP0)
    add     M0
    jmp     lab_csec2nop
```

Set register A0 to 0x12340000, and set register R0 to 0x00005678. A0 minus R0 and the result is pushed to M1.At last, end the PSC program.

```
csec2 section code abs=0x40110100
lab_csec2:
    mvi     AP0,data1
    ld      A0,(AP0)
    mvi     AP0,data2
    ld      R0,(AP0)
    sub     M1
;
    stop
```