TOSHIBA

TX00 Peripheral Driver Usage Example (TMPM061)

Ver 1

Sep, 2017

TOSHIBA ELECTRONIC DEVICES & STORAGE CORPORATION

CMDR-M061UE-01E

RESTRICTIONS ON PRODUCT USE

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

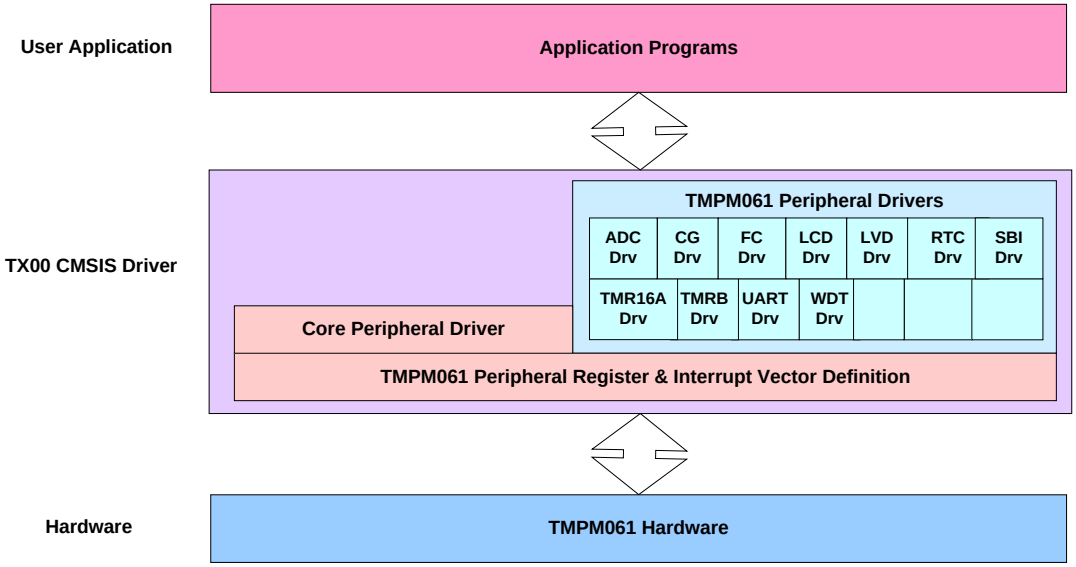
Index

1 General description

The example programs described hereafter are specially designed for TOSHIBA TMPM061 MCU. The programs can be executed individually each to show the main MCU functions. Users can utilize some portions of the programs and integrate them into users' own programs to perform customized functions.

2 Overview

User application utilizes TX00 peripherial driver as following.



3 Build-in hardware usage

Hardware	Channel	Use presence, use
CG	Clock Gear	Used for CG demo
Standby mode	-	Use SLEEP mode
External interrupt	INT0	Used for call CG wake up.
	INT1	Not used
	INT2	Not used
	INT3	Not used
SIO	SIO0	Used for UART retarget demo
	SIO1	Not used
	SIO2	Not used

Hardware	Channel	Use presence, use
	SIO3	Used for UART retarget and SIO demo
16-bit timerB	TMRB0	Used for TMRB: General Timer, PPG Output
	TMRB1	Not used
16-bit timerA	TMR16A0	Used for TMR16A: General Timer
	TMR16A1	Not used
	TMR16A2	Not used
	TMR16A3	Not used
	TMR16A4	Not used
	TMR16A5	Not used
	TMR16A6	Not used
RTC	RTC	Used for RTC demo
ADC	AIN0	Used for ADC demo
	AIN1	Not used
DSADC	DSAIN0	Used for DSADC demo
	DSAIN1	Used for DSADC demo
	DSAIN2	Used for DSADC demo
SBI	SBI0	Used for Master Tx/Slave Rx
WDT	WDT0	Used for WDT demo
LCD	LCD	Used to drive a customer’s LCD
LVD	LVD	Used for LVD demo

4 Pin Usage

The example programs are tested on TOSHIBA TMPM061 Evaluation Board.
Following is pin usage for example programs on TOSHIBA TMPM061 Evaluation Board.

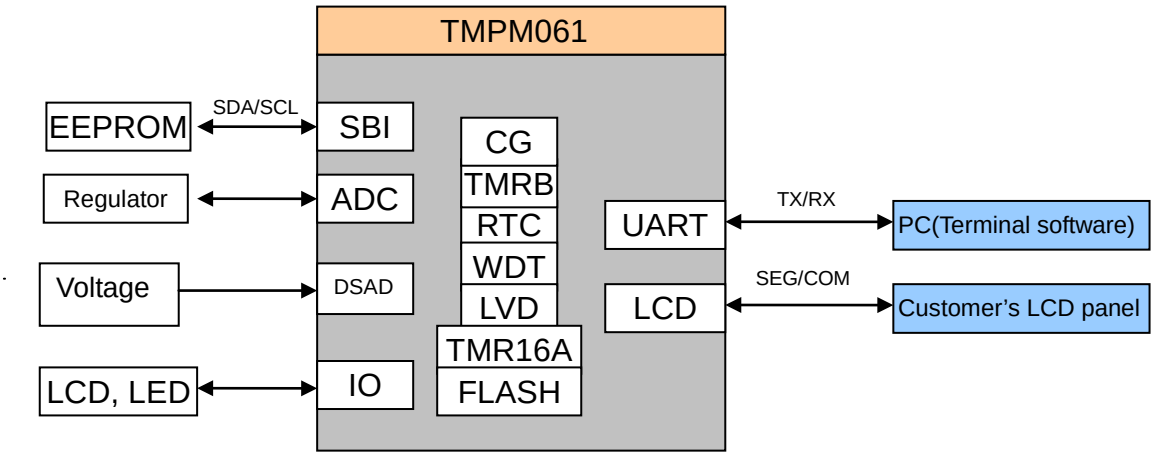
No	Name	Usage
1	VREFH	Supplying the 10bit AD converter with a reference power supply. Connect to DVDD3
2	AVDD3	Supplying the 10bit AD converter with a power supply. Connect to DVDD3
3	MODE	MODE pin. Connected to Gnd.
4	XT1	Not used
5	XT2	Not used
6	#RESET	Reset input pin, With a pull-up and a noise filter. Low active
7	X1	Connected to a high-speed oscillator. High-speed clock input pin.
8	DVSS	Gnd pin
9	X2	Connected to a high-speed oscillator.
10	DVDD3	Power supply pin
11	INT2/TB0OUT/PG0	Set PG0 as TB0OUT for PPG output
12	PH0/IROUT0/TXD0/	UART0 TX / LED0/SIO
13	PH1/RXD0	UART0 RX/ LED1/SIO
14	PH2/SCLK0/#CTS0/T16A0OUT	LED2/SCLK0
15	PH3/TXD1/IROUT1/#DBGEN0	Not used
16	PH4/RXD1	Not used
17	PH5/SCLK1/#CTS1/T16A1OUT	Not used
18	PI0/TXD2/IROUT2/#DBGEN1	Not used
19	PI1/RXD2	Not used
20	PI2/SCLK2/#CTS2/T16A2OUT	Not used
21	PI3/TB0IN	Not used
22	COUT	1.5V output
23	RVSS	GND pin
24	RVDD3	Power supply pin
25	PI4/TXD30	TXD3
26	PI5/RXD30	RXD3
27	PI6/SCLK30/#CTS30/T16A5OUT	SCLK3
28	PJ0/SDA0/SO0/#BOOT	SDA0 in SBI module
29	PJ1/SCL0/SI0	SCL0 in SBI module

No	Name	Usage
30	PJ2/SCK0/INT1	Not used
31	PJ3/RTCOUT	Not used
32	PJ4/T16A3OUT/SCOUT	Not used
33	PJ5/TB1IN/XTCLKIN	Not used
34	PE7/SEG39/SWDIO0	SEG39
35	PE6/SEG38/SWCLK0	SEG38
36	PE5/SEG37/TXD31	SEG37
37	PE4/SEG36/RXD31	SEG36/SWS
38	PE3/SEG35/SCLK31/#CTS31	SEG35/SW37
39	PE2/SEG34/T16A6OUT	SEG34/SW26
40	PE1/SEG33/SWDIO1	SEG33/SW15
41	PE0/SEG32/SWCLK1	SEG32/SW04
42	PD7/SEG31	SEG31/LCDDB7
43	PD6/SEG30	SEG30/LCDDB6
44	PD5/SEG29	SEG29/LCDDB5
45	PD4/SEG28	SEG28/LCDDB4
46	PD3/SEG27	SEG27/LCDDB3
47	PD2/SEG26	SEG26/LCDDB2
48	PD1/SEG25	SEG25/LCDDB1
49	PD0/SEG24	SEG24/LCDDB0
50	PC7/SEG23	SEG23
51	PC6/SEG22	SEG22
52	PC5/SEG21	SEG21
53	PC4/SEG20	SEG20
54	PC3/SEG19	SEG19
55	PC2/SEG18	SEG18/LDRS
56	PC1/SEG17	SEG17/LCDRW
57	PC0/SEG16	SEG16/LCDE
58	PB7/SEG15	SEG15
59	PB6/SEG14	SEG14
60	PB5/SEG13	SEG13
61	PB4/SEG12	SEG12
62	PB3/SEG11	SEG11
63	PB2/SEG10	SEG10
64	PB1/SEG9	SEG9
65	PB0/SEG8	SEG8
66	PA7/SEG7	SEG7
67	PA6/SEG6	SEG6
68	PA5/SEG5	SEG5
69	PA4/SEG4	SEG4
70	PA3/SEG3	SEG3

No	Name	Usage
71	PA2/SEG2	SEG2
72	PA1/SEG1	SEG1
73	PA0/SEG0	SEG0
74	DVDD3	Power supply pin
75	DVSS	GND pin
76	COM0	LCD common0 output pin
77	COM1	LCD common1 output pin
78	COM2	LCD common2 output pin
79	COM3	LCD common3 output pin
80	PK0/INT3/LV1	Not used
81	PK1/TB1OUT/LV2	Not used
82	VLC	LCD power supply pin
83	AGNDREF0	24bit ΔΣAD converter: GND pin. Connect to Gnd.
84	DAIN0+	24bit ΔΣAD converter analog input
85	DAIN0-	24bit ΔΣAD converter analog input
86	VREFIN0	Connected to DVDD3
87	AGNDREF1	Connected to Gnd
88	DAIN1+	24bit ΔΣAD converter analog input
89	DAIN1-	24bit ΔΣAD converter analog input
90	VREFIN1	Connected to DVDD3
91	AGNDREF2	Connected to Gnd
92	DAIN2+	24bit ΔΣAD converter analog input
93	DAIN2-	24bit ΔΣAD converter analog input
94	VREFIN2	Connected to DVDD3
95	DSRVSS	Connected to Gnd
96	DSRVDD3	Connected to DVDD3
97	SRVDD	Connected to DVDD3
98	PF0/AIN0	AIN0 for ADC module
99	PF1/AIN1/INT0	INT0 for CG wakeup
100	AVSS	10bit AD converter: GND pin, Connect to Gnd

5 Development Environment

Following is development evironment:



1. Hardware board:
   - TOSHIBA TMPM061 Evaluation Board.
2. Development tool:
   - IAR:
     1) J-Link: IAR J-Link-7.0
     2) IDE: IAR Embed Workbench for ARM version 6.40
   - KEIL:
     1) u-Link: Realview ULINK2
     2) IDE: KEIL uVision MDK version 4.21

6 Functions

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

Operation mode

There are five operation modes for TMPM061: NORMAL, SLOW, IDLE, SLEEP and STOP mode.
IDLE, SLEEP and STOP mode are low power mode.
To shift to lower power mode, in system control register STBYCR0<STBY2:0>, select the IDLE, SLEEP or STOP mode, and run the WFI (Wait For Interrupt) command.

- **NORMAL mode:**

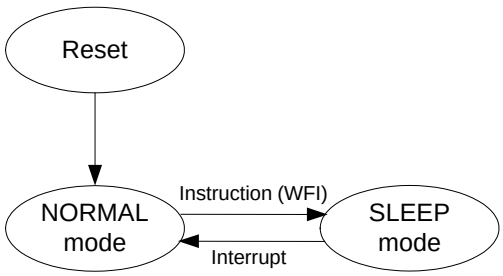
This mode is to operate the CPU core and the peripheral hardware by using the high-speed clock(fosc1 or fosc2). It is shifted to the NORMAL mode with fosc2(internal high-speed oscillator) after reset.

Note: Only SLEEP mode is demostrated in driver example.

- **SLEEP mode:**

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

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



6-1 ADC

Change the value of *VR2*, which is connected to *AIN0*(pin98) . The voltage on it will be measured and printed to stdout. As stdout is retargeted to UART0, connect UART0 with a PC and the AD result can be displayed on terminal software.

The UART configuration is: baud rate = 115200, no handshake, 8-bit data, 1-bit stop.

The pin1(VREFH) and pin2(AVDD3) must be connect to pin10 or pin74 (DVDD3).
The pin100(AVSS) must be connected to pin8 or pin75(DVSS).

6-2 CG

6-2-1 Power mode change

Change the CPU operation mode. Only 2 modes are supported, NORMAL and STOP. Press the SW0 to change the power mode from NORMAL mode to STOP mode and LED0 ~ LED4 is turned off, and then Press the SW1 to change the power mode from STOP mode to NORMAL mode and LED0 ~ LED4 is turned on.

* Connect PF1/ PIN 99 (INT0) to PIN4 of PJ24 of TX03 basic board.

Current Mode	Action	Operation
NORMAL	SW0	NORMAL → STOP
STOP	SW1	STOP → NORMAL

6-3 DSADC

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

It reads three channels DSADC value, and output the value by UART communication.

It includes:

1. Software trigger for DSADC.
2. Check the DSADC status and read the result.

NOTE:

Pins connection:

1. Voltage input
   (a) Connect pin 84 (DAIN0+) / pin 85(DAIN0-) to voltage (-0.375V < voltage input < +0.375V)
   (b) Connect pin 88 (DAIN1+) / pin 89(DAIN1-) to voltage (-0.375V < voltage input < +0.375V)
   (c) Connect pin 92 (DAIN0+) / pin 93(DAIN0-) to voltage (-0.375V < voltage input < +0.375V)

2. Connect pin 83(AGNDREF0), pin 87(AGNDREF1), pin 91(AGNDREF2) to Gnd.

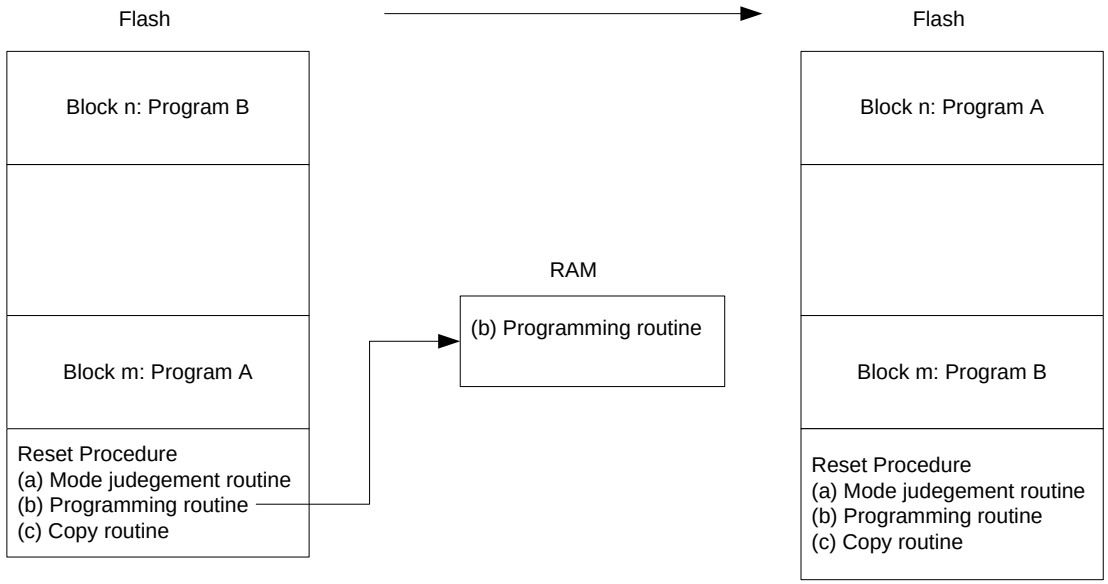
3. Connect pin 86(VREFIN0) to pin '+' of a 1uF capacitor, the '-' pin of capacitor is connected to pin 83(AGNDREF0).
4. Connect pin 90(VREFIN1) to pin '+' of a 1uF capacitor, the '-' pin of capacitor is connected to pin 87(AGNDREF1).
5. Connect pin 94(VREFIN2) to pin '+' of a 1uF capacitor, the '-' pin of capacitor is connected to pin 91(AGNDREF2).
6. Connect pin 96(DSRVDD3) and pin 97(SRVDD) to Vcc.
7. Connect pin 95(DSRVSS) to Gnd.

6-4 FLASH

This example demonstrates the feature of flash APIs such as erase and write operation.
The demo runs in Normal mode and User Boot Mode of Single chip mode.
－Reset procedure: includes Mode judgment, Programming routine(flash APIs), Copy routine
・Mode judgment routine: judge to enter User Boot Mode or Normal Mode
・Copy routine: copy programming routine(flash APIs) from flash to RAM
・Programming routine: runs at RAM and swap Program A and Program B code in flash
－Program A/B (Reset procedure) are programmed in flash block in advance.
－Program A/B(A: LED0 blinks and LCD displays ”DEMO A”,
B: LED1 blinks and LCD displays “DEMO B”)
By default, the program A will run firstly
－Toggle switch SW0 is used for mode judgment in Reset procedure

SW0 turned on → User boot mode

SW0 turned off → Normal mode



Demo sequence

(1) Power on

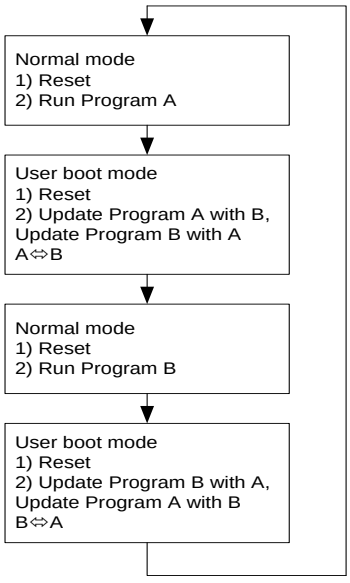
The demo board runs Reset Procedure.
Then initial program A stored in flash runs.
LED0 blinks, and LCD displays “DEMO A”.

(2) Press RESET button while SW0 is turned on, and release SW0 until LCD displays “User Boot Mode”.
The demo board runs Reset Procedure: Programming routine is copied to RAM by Copy routine.
Then run Programming routine to swap program A and B in flash.
During this period, LCD displays information such as “RAM transferring ……”, “FLASH swapping ……” and “Finished Restart ……”.

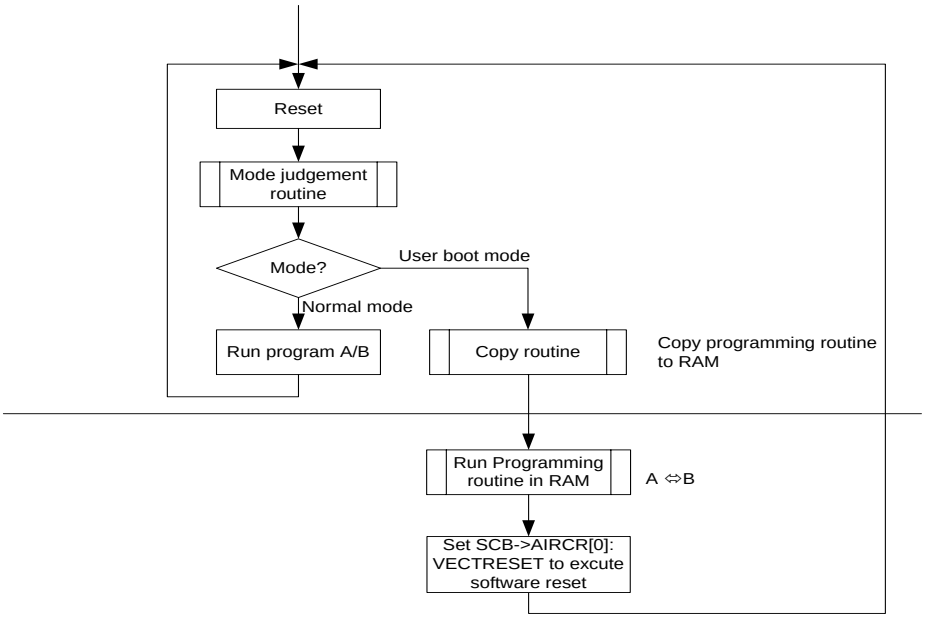
(3) After LCD has displayed “Finished Restart ……”, the demo will reset by software automatically.
Then program B stored in flash runs.
LED1 blinks, and LCD displays “DEMO B”.

(4) Press RESET button again while SW0 is turned on, and release SW0 until LCD display information “User Boot Mode”.
Same as step (2).

(5) After LCD has displayed “Finished Restart ……”, the demo will reset by software automatically.
Then program A stored in flash runs.
LED0 blinks, and LCD displays “DEMO B”.



・**Demo process flow**



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

User Boot Mode

(Reset Procedure)
While RAM transferring or flash programming, LCD displays the current processing.
LCD display sample:

RAM transferring ……

(Copy routine)

FLASH swapping ……

(Programming routine)

Finished Restart ……

(Programming routine)

6-5 LCD

There are two LCD example which will demonstrate how to use M061 LCD driver.

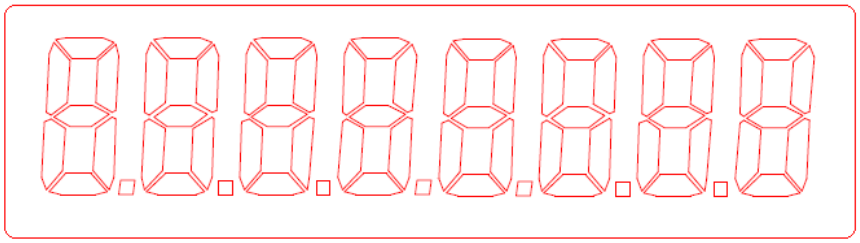
6-5-1 LCD demo1

It use an customer's LCD, the digit in the string '12345678' will be displayed on the digit '8' area one by one in adjustable interval time.

The display order are:

1. display 1 in the right one
2. display 12 in right two digit
3. display 123 in right three digit
4. display 1234 in right four digit
5. display 12345 in right five digit
6. display 123456 in right six digit
7. display 1234567 in right sever digit
8. display 12345678 in all the eight digit
9. goto step 1 above

The simplified LCD panel information is as below (1/4 duty, 1/3 bias):



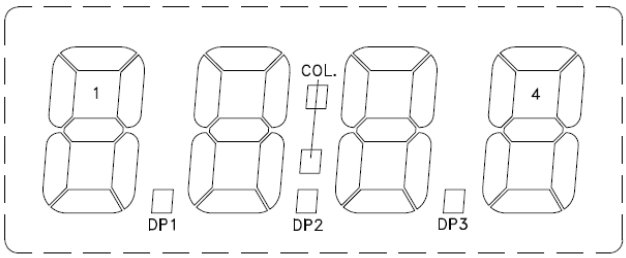
6-5-2 M061-SK LCD demo

This demo use the static LCD on TMPM061-SK board,(type: S5026D with 4 digit '8') the digit in the string ‘ 0123456789’ will be displayed on the LCD one by one in adjustable interval time.

The display order are:

1. display 0
2. display 01
3. display 012
4. display 0123
5. display 123.4
6. display 23.45
7. display 3.456
8. display 4567
9. display 5678
10. display 6789
11. goto step 1 above

The simplified LCD panel information is as below (static LCD):



6-6 LVD

This example implements power supply voltage status detecting by LVD.

When the power supply voltage is normal, LED0 will blink, LED1 is off.
When the power supply voltage is lower than the detection voltage, it will cause an interrupt: INTLVD, in the interrupt service routing, the LED0 will be turned off and LED1 will blink.

6-7 RTC

For EVAL board:

Use the internal RTC to display the time and date on LCD.
Display the time and date on LCD and update the date and time.
Initial setting: 2010/10/22 12:50:55, 24hours format.
Update interval: 1 second.
LCD displays:

2010/10/22 12:50:55

For SK board:

Display the minute and second on LCD board, the intial time is 2010/10/22 12:50:55, 24hours format. Update interval is 1 second.
LCD displays:

6-8 SBI

This function intends to support the I2C bus master and slave mode.
Connect two I2C buses on two M061 evaluation board, one works as master and the other works as slave.
Uses SBI interrupt to handle I2C bus read/write.

I2C Slave board receives “TOSHIBA” from Master board and LCD displays it.
When the program start, LCD displays as below:

Then one board pressed SW0 works as Master, and the other board pressed SW2 works as Slave. The two boards display as below:

Master board Slave board

When SW1 is pressed on two board, the string “TOSHIBA” that got by slave received buffer will be displayed.

Master board Slave board

Note: In order to send data between the two boards, connect the ports below.
Connect the two pins PJ1 of M061 Evaluation board (SCL).
Connect the two pins PJ0 of M061 Evaluation board (SDA).
Connect two ground pins of the boards to make the two boards have the same ground.

6-9 TMR16A

6-9-1 General Timer

Set a value of T16AxRG<RG>, then start count-up, if the counter value matches with a value of T16AxRG<RG>, the counter will be cleared to "0x0000" and continued to count-up and the same time a match detection interrupt INTT16Ax will be output.

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

6-10 TMRB

6-10-1 General Timer

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

6-10-2 PPG Output

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

6-11 UART

6-11-1 Retarget

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

6-11-2 SIO

This demo will show synchronously transfer and receive usage of SIO module in TMPM061
It use the channel SIO0, SIO3 and transfer data synchronously between them.(connect TXD0 with RXD3, connect TXD3 with RXD0, connect sclk0 with sclk3)

6-12 WDT

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

The driver example step:

1. Initialize WDT by setting detection time and selecting WDT interrupt as counter overflow operation.

2. There are two demos for WDT in which DEMO2 is switched by macro definition.

DEMO1:

When timer is overflow, NMI interrupt is generated and then WDT will be disabled.

DEMO2:

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

7 Software

This software project creates the sample applications based on TOSHIBA TMPM061 Evaluation Board which will demonstrate the main feature of TMPM061 MCU.

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

The workspace structure and project names are defined as following:

```
\---M061
    +---Libraries
    |   +---TX00_CMSIS
    |   |   |   system_TMPM061.c
    |   |   |   system_TMPM061.h
    |   |   |   TMPM061.h
    |   |   \---startup
    |   |       +---arm
    |   |       |       startup_TMPM061.s
    |   |       \---iar
    |   |               startup_TMPM061.s
    |   \---TX00_Periph_Driver
    |       +---inc
    |       |       tmpm061_adc.h
    |       |       tmpm061_cg.h
    |       |       tmpm061_dsad.h
    |       |       tmpm061_fc.h
    |       |       tmpm061_lcd.h
    |       |       tmpm061_lvd.h
    |       |       tmpm061_rtc.h
    |       |       tmpm061_sbi.h
    |       |       tmpm061_tmr16a.h
    |       |       tmpm061_tmrb.h
    |       |       tmpm061_uart.h
    |       |       tmpm061_wdt.h
    |       |       tx00_common.h
    |       \---src
    |               tmpm061_adc.c
    |               tmpm061_cg.c
    |               tmpm061_dsad.c
    |               tmpm061_fc.c
    |               tmpm061_lcd.c
```

```
|                   tmpm061_lvd.c
|                   tmpm061_rtc.c
|                   tmpm061_sbi.c
|                   tmpm061_tmr16a.c
|                   tmpm061_tmrb.c
|                   tmpm061_uart.c
|                   tmpm061_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_M061_Driver_IAR.eww
    |       \---KEIL
    |                Examples_for_M061_Driver_Keil.uvmpw
    \---Template
        +---IAR
        |       TMPM061_Flash.icf
        |       TMPM061_RAM.icf
        \---KEIL
                tmpm061.sct
```

7-1 ADC

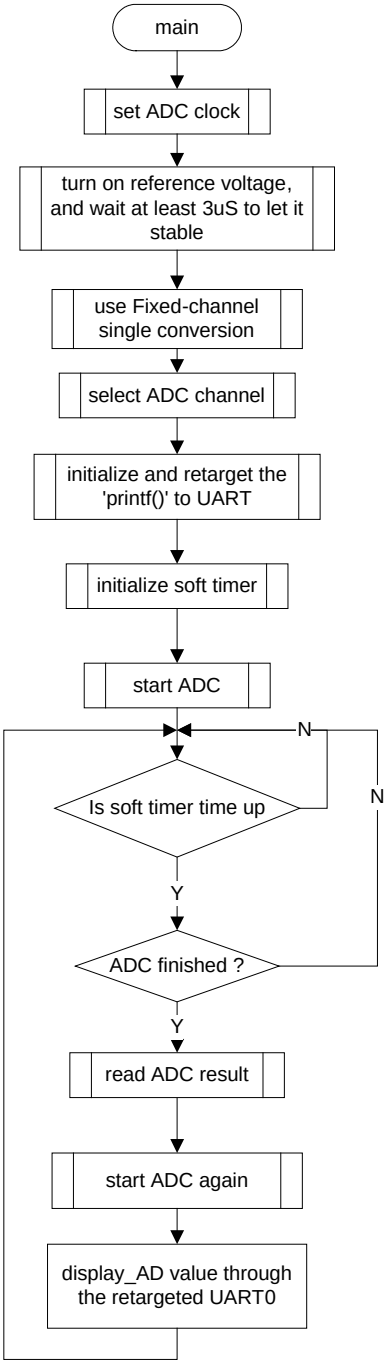
7-1-1 Example: ADC Data Read

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

The example includes:

1. ADC configuration and initialization
2. Start AD conversion by software and read AD result

- **Flowchart:**



- **Code and Explanation for the Example**

At first, set ADC clock, turn on reference voltage,

```
/* 1. set ADC clock */
ADC_SetClk(ADC_CONVERSION_81_CLOCK,
                ADC_FC_DIVIDE_LEVEL_16);

/* 2. turn on reference voltage, and wait at least 3uS to let it stable */
ADC_SetVref(ENABLE);
timeUp = 300U;
while( timeUp ) {
      timeUp -- ;
}
```

Then set to use fixed-channel single conversion, and select AD channel:

```
/* 3. use Fixed-channel single conversion */
ADC_SetScanMode(DISABLE);
ADC_SetRepeatMode(DISABLE);

/* 4. select ADC channel */
ADC_SetInputChannel(ADC_CHANNEL);
```

After that, start ADC:

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

To use the standard “printf()” and soft timer, we need to do some initialize:

```
/* initialize and retarget the 'printf()' to UART */
Retarget_Init();

/* initialize soft timer */
softTimer_Init();
```

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

```
while (1U) {
    softTimer_Run();
    if(softT.flag_TimeUp) {
        softT.flag_TimeUp = 0U;

        /* check ADC module state */
        adcState = ADC_GetConvertState();
        if (adcState == DONE) {
            /* read ADC result when it is finished */
            adResult = ADC_GetConvertResult(ADC_REG_RESULT);
            myResult = (uint32_t)adResult.Bit.ADResult ;
            ADC_Start();
            display_AD(myResult);
        }
    } else {
        /* Do nothing */
    }

}
```

In display_AD() function, use the standard “printf()” to send ADC results to PC:

```
printf("ADC Value is %4d\r\n", myResult);
```

7-2 CG

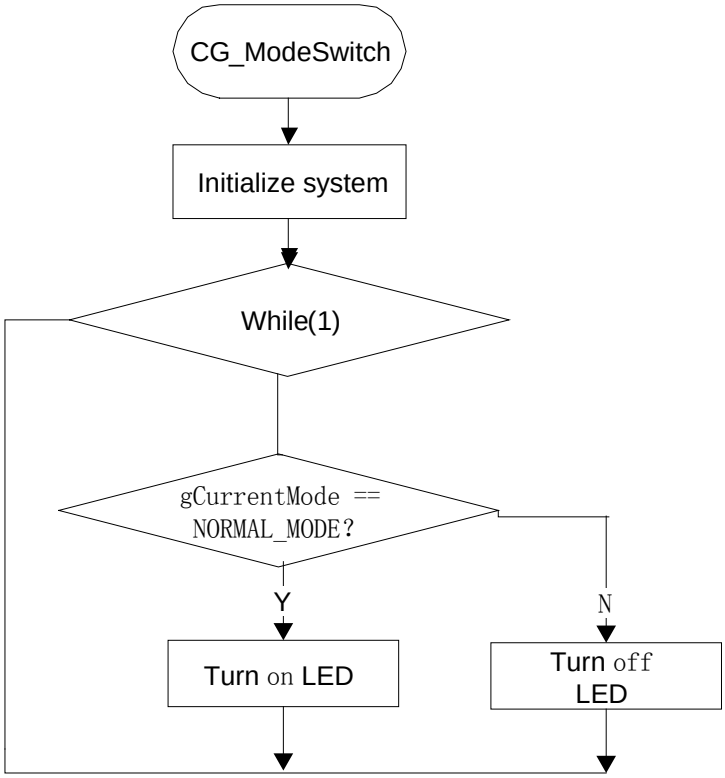
7-2-1 Example: Power mode change

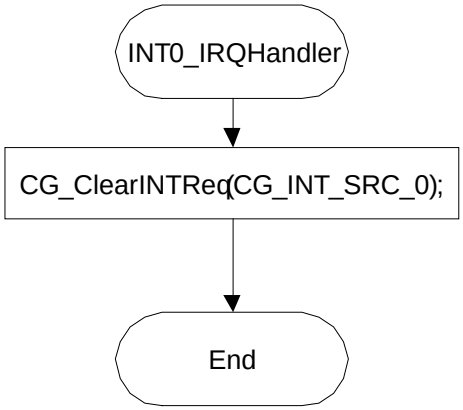
This is a simple example based on the TX00 Peripheral Driver(CG).

The example includes:

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

- **Flowchart**





- **Code and Explanation for the Example**

The following simple example is based on TX00 Peripheral Driver (CG), which will switch between normal mode and sleep mode.

Normal setup for CG (after reset)

The following code is just an example for setting CG in normal mode. It is supposed that the high-speed oscillator is 16MHz.

Example code is as the following:

```
/* Switch over from IHOSC to EHOSC*/
switchFromIHOSCtoEHOSC();
```

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

Enter stop mode

Change CG mode: normal ->stop mode.

Press SW0 to change the mode from normal to stop.

```
/* CG_NormalToStop */
void CG_NormalToStop(void)
{
	/* Set CG module: Normal ->Stop mode */
	CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC2,CG_WUODR_EXT);
	/* Enter stop mode */
	__WFI();
}
```

Prepare to enter normal mode. Disable modules except int0.

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

Set external interrupt (int0) interrupt to wake up system. Press the SW1 to change the mode to normal mode.

```
	void INT0_IRQHandler(void)
	{
		CG_ClearINTReq(CG_INT_SRC_0);
	}
```

7-3 DSADC

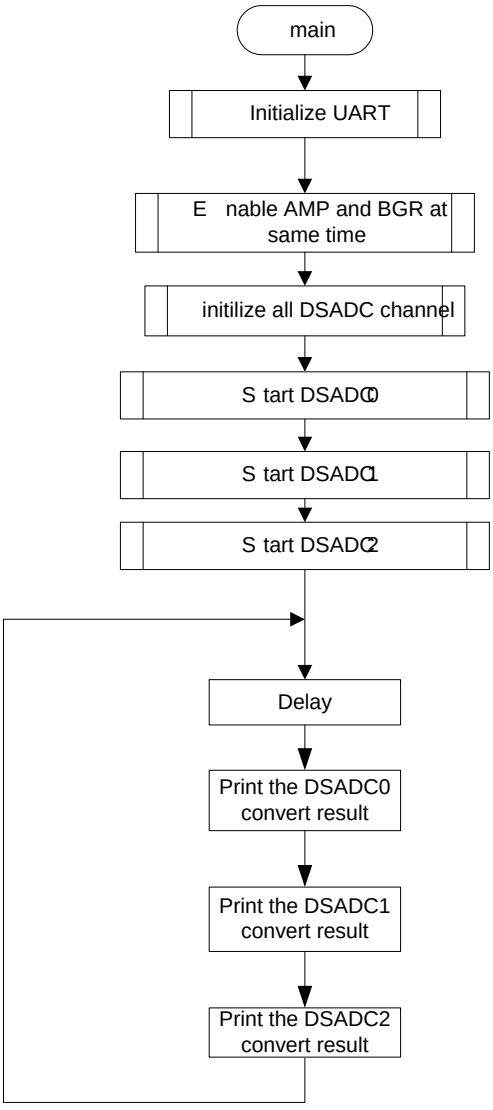
7-3-1 Example: DSADC Data Read

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

The example includes:

1. Software trigger for DSADC.
2. Check the DSADC status and read the result.

- **Flowchart:**



- **Code and Explanation for the Example**

At first, initialize UART.

```
Retarget_Init();
```

Then Enable AMP and BGR at same time.

```
TSB_TEMP->EN = 0x03U;
while(f1ms != 1U){    }; /* delay some time about 1ms*/
```

Initilize all DSADC channel.

```
for(i = 0U; i < (sizeof(t_DSADx)>>2U); ++ i){            /*initilize all DSADC channel*/
        DSADCx = t_DSADx[i];
        DSADC_SWReset(DSADCx);
        DSADC_SetClkSupply(DSADCx,ENABLE);      /*Clock feed to DSADC*/

        InitStruct.Clk = DSADC_FC_DIVIDE_LEVEL_1;
        InitStruct.BiasEn = ENABLE;
        InitStruct.ModulatorEn = ENABLE;
        InitStruct.SyncMode=DSADC_SYNC_MODE;

        InitStruct.Repeatmode = DSADC_REPEAT_MODE;    /*Repeat mode*/
        InitStruct.Amplifier = DSADC_GAIN_8x;
        InitStruct.Offset = 0x03U;
        InitStruct.CorrectEn = ENABLE;
        DSADC_Init(DSADCx,&InitStruct);
    }
```

Get the convert result and print the value by UART.

```
while(1U){
if(f1s == 1U){                  /*every 1s*/
      f1s = 0u;
      result = DSADC_GetConvertResult(TSB_DSAD0);
      printf("DSAD0: %d\r\n",result);

      result = DSADC_GetConvertResult(TSB_DSAD1);
      printf("DSAD1: %d\r\n",result);
      Delay(1000);
      result = DSADC_GetConvertResult(TSB_DSAD2);
      printf("DSAD2: %d\r\n",result);
  }
```

NOTE1: Pins connection:

1. Voltage input
   (a) Connect pin 84 (DAIN0+) / pin 85(DAIN0-) to voltage (-0.375V < voltage input < +0.375V)
   (b) Connect pin 88 (DAIN1+) / pin 89(DAIN1-) to voltage (-0.375V < voltage input < +0.375V)
   (c) Connect pin 92 (DAIN0+) / pin 93(DAIN0-) to voltage (-0.375V < voltage input < +0.375V)
2. Connect pin 83(AGNDREF0), pin 87(AGNDREF1) , pin 91(AGNDREF2) to Gnd.
3. Connect pin 86(VREFIN0) to pin '+' of a 1uF capacitor, the '-' pin of capacitor is connected to pin 83(AGNDREF0).
4. Connect pin 90(VREFIN1) to pin '+' of a 1uF capacitor, the '-' pin of capacitor is

connected to pin 87(AGNDREF1).

5. Connect pin 94(VREFIN2) to pin '+' of a 1uF capacitor, the '-' pin of capacitor is connected to pin 91(AGNDREF2).
6. Connect pin 96(DSRVDD3) and pin 97(SRVDD) to Vcc.
7. Connect pin 95(DSRVSS) to Gnd.

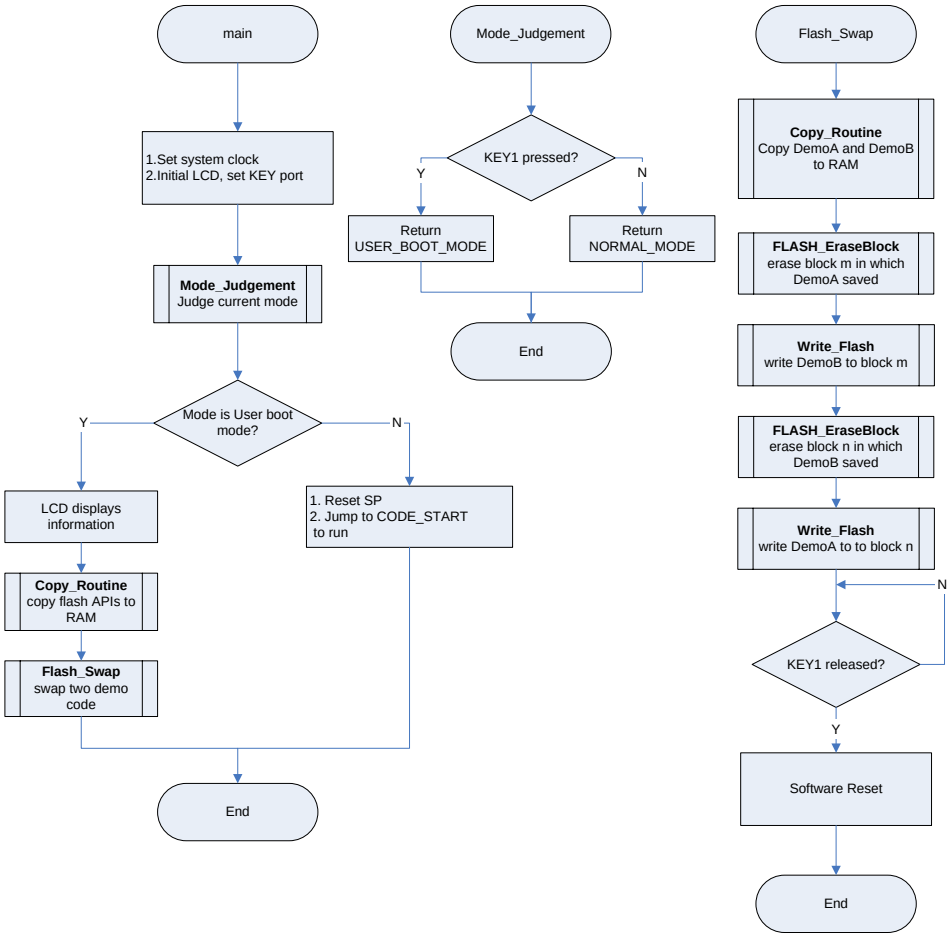
7-4 FLASH

This is a simple example based on the TX00 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 LCD and switch. SW0 (GPIO) is used to judge current mode when reset and LCD will display current running routine and Flash operation information.

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

```
    LCD_Configuration();
```

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. LCD will display information to user. 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.

```
        LCD_Display("User Boot Mode", "");          /* enter user boot mode */
        LCD_Display("RAM transferring", ".....");

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

        LCD_Display("Flash swapping", ".....");
```

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;
    }
```

LCD displays the information to indicate the swap operation has completed. 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, the address “CODE_START” has become the start address of Demo B, Demo B will run (LED1 blinks).

```
    LCD_Display("Finished", "Restart.....");


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

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

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

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

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

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

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

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

Function Write_Flash() will call FLASH_WritePage() to write data automatically in one page. The process of this function is basically same as FLASH_EraseBlock() except the automatic page program command._

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

7-5 LCD

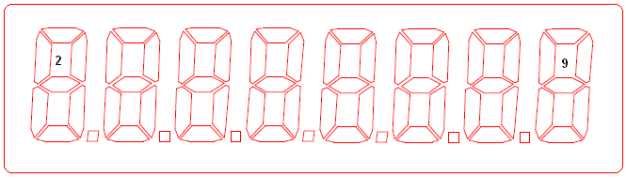
7-5-1 Example: LCD Demo1

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

The example includes:

1. LCD configuration and initialization
2. Display some digit in LCD

The detail schematic drawing inside LCD is as below:



PIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	0
COM1	S15	2D	T1	3D	T2	4D	S1									COM1	/	/	/	
COM2	2E	2C	3E	3C	4E	4C	S2									/	COM2	/	/	
COM3	2G	2B	3G	3B	4G	4B	S3									/	/	COM3	/	
COM4	2F	2A	3F	3A	4F	4A	S4									/	/	/	COM4	

PIN	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
COM1			9D	P7	8D	P6	7D	P5	6D	P4				5D	T3		
COM2			9C	9E	8C	8E	7C	7E	6C	6E				5C	5E		
COM3			9B	9G	8B	8G	7B	7G	6B	6G				5B	5G		
COM4			9A	9F	8A	8F	7A	7F	6A	6F				5A	5F		

Connection table:

LCD pin 36 ----> MCU Seg0 LCD pin 35 ----> MCU Seg1

LCD pin 34 ----> MCU Seg2 LCD pin 33 ----> MCU Seg3

....

LCD pin 19 to 16 ---> MCU Com3 to Com0 …

LCD pin 15 ----> MCU Seg17 LCD pin 14 ----> MCU Seg18

LCD pin 13 ----> MCU Seg19 LCD pin 12 ----> MCU Seg20

...

LCD pin 02 ----> MCU Seg30 LCD pin 01 ----> MCU Seg31

From the information above, we can make the font table as below:

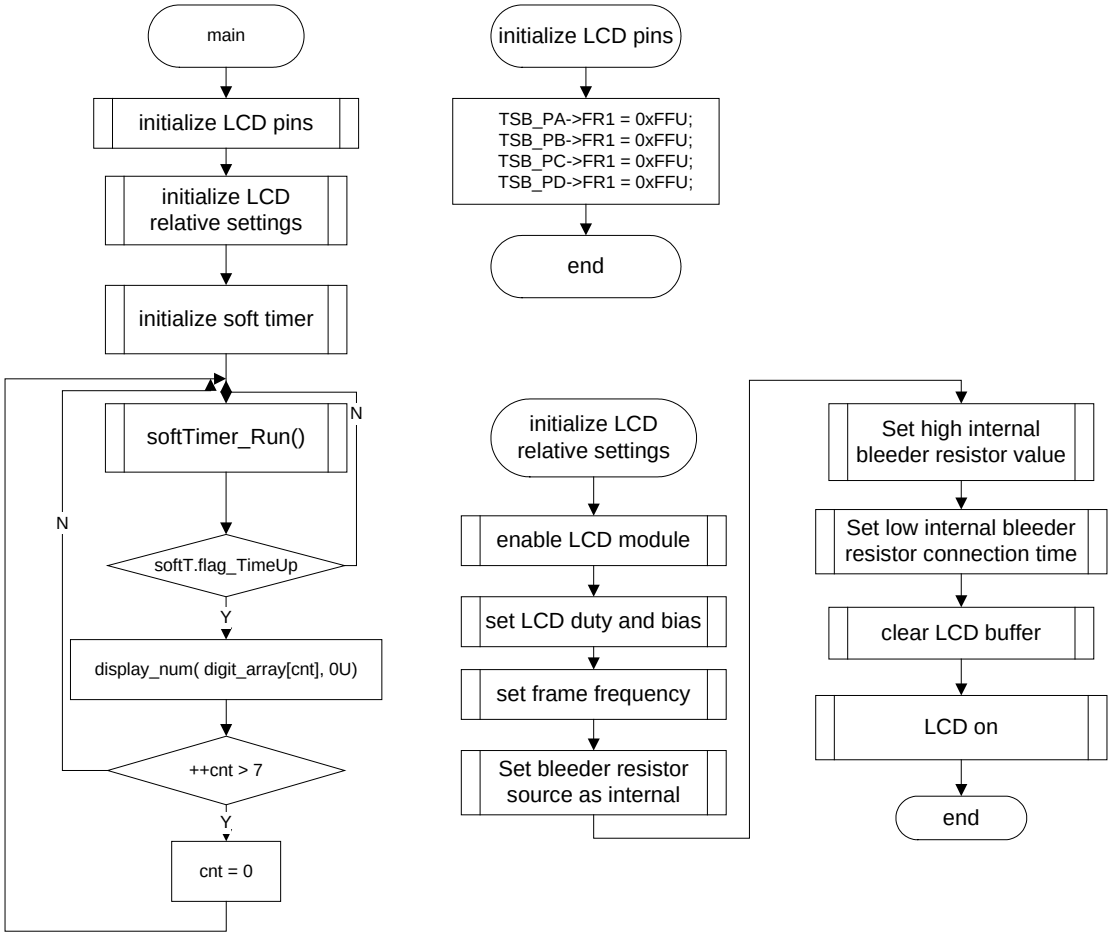
```
const uint8_t number_font_table[] = {
    0xAF,   /* digit '0' */
    0x06,   /* digit '1' */
    0x6D,   /* digit '2' */
    0x4F,   /* digit '3' */
    0xC6,   /* digit '4' */
    0xCB,   /* digit '5' */
```

```
    0xEB,    /* digit '6' */
    0x0E,    /* digit '7' */
    0xEF,    /* digit '8' */
    0xCF,    /* digit '9' */
    0x00     /* ' ' (SPACE) */
};
```

The relationship between the font data and LCD buffer is listed below:

Drive method	Bits 7/3	Bits 6/2	Bits 5/1	Bits 4/0
1/4 duty	COM3	COM2	COM1	COM0
1/3 duty	-	COM2	COM1	COM0
1/2 duty	-	-	COM1	COM0
Static	-	-	-	COM0

- **Flowchart:**



- **Code and Explanation for the Example**

At first, initialize LCD pins by call lcd_port_init(), which is detailed as below:,

```
void lcd_port_init(void)
{
    /* Port init to SEG and COM */
    TSB_PA->FR1 = 0xFFU;
    TSB_PB->FR1 = 0xFFU;
    TSB_PC->FR1 = 0xFFU;
    TSB_PD->FR1 = 0xFFU;
}
```

Then initialize LCD relative settings by called lcd_configure_init(), which is detailed as below:

```
    /* enable LCD module */
    LCD_Enable();

    /* set LCD duty and bias */
    LCD_SetDutyBias(LCD_DUTY4_BIAS3);              /* 1/4 duty, 1/3 bias */

    /* set frame frequency */
    LCD_SetBaseFreq(LCD_FS_DIVIDE_2_POWER_8); /* F_base = fs / (2^8) */

    /* Set bleeder resistor source as internal */
    LCD_SetBleederSource(LCD_BLEEDER_RESISTOR_INTERNAL);

    /* Set high internal bleeder resistor value */
    LCD_SetInternalBleeder(LCD_BLEEDER_RESISTOR_500K);

    /* Set low internal bleeder resistor connection time */
    LCD_SetLowBleederTime(LCD_2_POWER_6_DEVIDE_BASE_FREQ);

    /* clear LCD buffer */
    lcd_clear();

    /* LCD on */
    LCD_SetDisplay(ENABLE);
```

After that, initialize soft timer

```
softTimer_init();
```

Finally, in the while(1) loop, check the soft timer, if it is time up, call display_num() to display the digit string.

```
    while (1U) {

        softTimer_Run();
        if(softT.flag_TimeUp) {
            softT.flag_TimeUp = 0;

            display_num( digit_array[cnt], 0U);
            if( ++cnt > 7 ) {
                cnt = 0;
            } else {
                /* Do nothing */
            }
```

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

In the function display_num(), the display data will be limited, and do the ‘hide 0 in high position”, then call the core function set_digit() to send the font data to LCD buffer.
for detail code for display_num(), please refer it in the file lcd061.c

The core code for set_digit() is listed as below:

```
    void set_digit(uint8_t digit_loc, uint8_t digit, uint8_t dot)
    {
        LCD_font8 font = { 0U };
        LCD_BufIndex idx = LCD_BUF_SEG14;

        if( digit > SPACE ) {
            digit = SPACE;
        } else {
            /* Do nothing */
        }
        font.All = number_font_table[digit];
        if( dot ){
          font.All |= 0x10U;
        } else {
          /* Do nothing */
        }

        if( digit_loc > MAX_DIGITS - 1U ) {
            digit_loc = MAX_DIGITS - 1U;
        } else {
            /* Do nothing */
        }

        if( digit_loc <= 3U ){
            idx -= (LCD_BufIndex)(digit_loc * 2 );
            LCD_WriteBuf( idx, font.Bit.low);
            LCD_WriteBuf( (LCD_BufIndex)(idx - 1U) , font.Bit.high);
        } else if ( digit_loc == 4U ) {
            LCD_WriteBuf( LCD_BUF_SEG03, font.Bit.low);
            LCD_WriteBuf( LCD_BUF_SEG02, font.Bit.high);
        } else {
            idx = (LCD_BufIndex)(LCD_BUF_SEG2726 + (LCD_BufIndex)(digit_loc -
5U));
            LCD_WriteBuf(idx, font.All);
        }

    }
```

We use the function LCD_WriteBuf() to write the font data to the segment data area, because the connection table for each digit ‘8’ in LCD isn’t in same order, we must process the digit one by one.

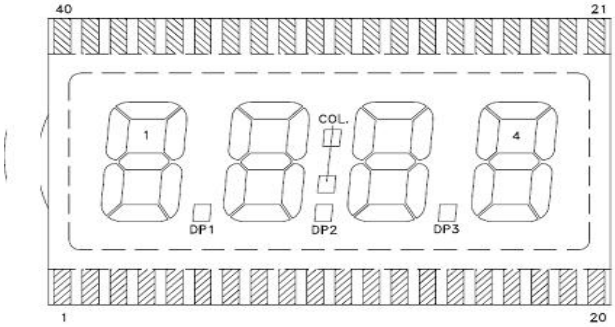
7-5-2 Example: M061-SK LCD

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

The example includes:

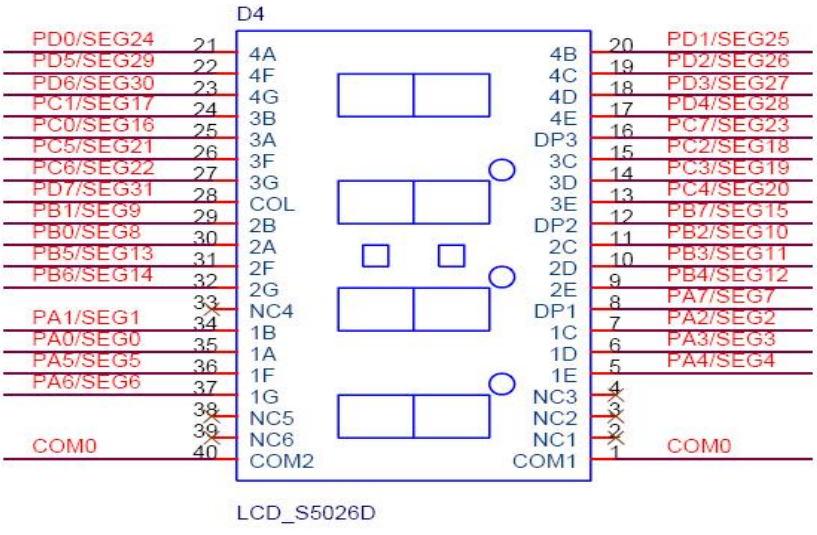
1. LCD configuration and initialization
2. Display some digit in LCD

The detail schematic drawing inside LCD is as below:



PIN	1	2	3	4	5	6	7	8	9	10
COM	COM	/	/	/	1E	1D	1C	DP1	2E	2D
PIN	11	12	13	14	15	16	17	18	19	20
COM	2C	DP2	3E	3D	3C	DP3	4E	4D	4C	4B
PIN	21	22	23	24	25	26	27	28	29	30
COM	4A	4F	4G	3B	3A	3F	3G	COL.	2B	2A
PIN	31	32	33	34	35	36	37	38	39	40
COM	2F	2G	/	1B	1A	1F	1G	/	/	COM

Connection table:



From the information above, we can make the font table as below:

```
const uint8_t number_font_table[] = {
    0x3F,   /* digit '0' */
    0x06,   /* digit '1' */
    0x5B,   /* digit '2' */
```

```
    0x4F,     /* digit '3' */
    0x66,     /* digit '4' */
    0x6D,     /* digit '5' */
    0x7D,     /* digit '6' */
    0x07,     /* digit '7' */
    0x7F,     /* digit '8' */
    0x6F      /* digit '9' */
};
```

The relationship between the font data and LCD buffer is listed below:

Drive method	Bits 7/3	Bits 6/2	Bits 5/1	Bits 4/0
1/4 duty	COM3	COM2	COM1	COM0
1/3 duty	-	COM2	COM1	COM0
1/2 duty	-	-	COM1	COM0
Static	-	-	-	COM0

- **Flowchart:**
  The flowchart for this M061-SK LCD demo is similar with the one in the “Example: LCD Demo1” above. except that here it will call function “display_4digit(&str[idx], dot);” instead to call “display_num(digit_array[cnt], 0U);”

- **Code and Explanation for the Example**

At first, initialize LCD pins by call lcd_port_init(), which is detailed as below:,

```
void lcd_port_init(void)
{
    /* Port init to SEG and COM */
    TSB_PA->FR1 = 0xFFU;
    TSB_PB->FR1 = 0xFFU;
    TSB_PC->FR1 = 0xFFU;
    TSB_PD->FR1 = 0xFFU;
}
```

Then initialize LCD relative settings by called LCD_Init (), which is detailed as below:

```
{
    /* enable low-speed oscillator for LCD */
    if(CG_GetFsState() == DISABLE)
    {
        CG_SetFs(ENABLE);
    }
    lcd_port_init();
    lcd_configure_init();
}
```

Then LCD configure relative settings by called lcd_configure_init(), which is detailed

as below:

```
{
    /* enable LCD module */
    LCD_Enable();

    /* set LCD duty and bias */
    LCD_SetDutyBias(LCD_STATIC);          /* static LCD on M061-SK board */

    /* set frame frequency */
    LCD_SetBaseFreq(LCD_FS_DIVIDE_2_POWER_8);   /* F_base = fs / (2^8) */

    /* Set bleeder resistor source as internal */
    LCD_SetBleederSource(LCD_BLEEDER_RESISTOR_INTERNAL);

    /* Set high internal bleeder resistor value */
    LCD_SetInternalBleeder(LCD_BLEEDER_RESISTOR_500K);

    /* Set low internal bleeder resistor connection time */
    LCD_SetLowBleederTime(LCD_2_POWER_6_DEVIDE_BASE_FREQ);

    /* clear LCD buffer */
    lcd_clear();

    /* LCD on */
    LCD_SetDisplay(ENABLE);
}
```

After that, initialize soft timer

```
softTimer_init();
```

Finally, in the while(1) loop, check the soft timer, if it is time up, call display_4digit() to display the digit string.

```
while (1U) {
    softTimer_Run();
    if(softT.flag_TimeUp) {
        softT.flag_TimeUp = 0;

        if(idx > 3U && idx < 7U){
            dot = 7U - idx;
        } else {
            dot = 0U;
        }
        display_4digit( &str[idx], dot );

        idx++;
        if( idx > 9U ) {
            idx = 0U;
        } else {
            /* Do nothing */
        }
    } else {
        /* Do nothing */
    }
}
```

The code for display_4digit() is listed as below, because for a static LCD, it only use bit0

of LCD buffer, so, we must set the 8 bit data of the font to bit0 of 8 LCD buffer register.

```
void display_4digit(unsigned char *p, uint8_t dot_pos)
{
    uint8_t tmp = 0U, loop = 0U;
    uint8_t idx = 0U, dat = 0U;

    LCD_BufIndex buf_head = LCD_BUF_SEG0100;

    for( ; loop < 4U; loop++) { /* total 4 digit need to display */
        tmp = *(p + loop);
        if( tmp == ' ' ) {          /* is SPACE, hide it */
            tmp = 0U;
        } else {
            tmp = number_font_table [ tmp - '0' ];
            if( dot_pos == loop + 1U) { /* if dot_pos isn't '0', should display dot */
                tmp |= 0x80U;
            } else {
                /* Do nothing */
            }
        }

   /* now 'tmp' store the font data, we will set its 8 bit info to 8 LCD Segment buffer */
        for(idx = 0U; idx < 4U; idx++) {
            dat = 0U;
            if( tmp & 0x01U ) {
                dat |= 0x01U;
            } else {
                /* Do nothing */
            }
            if( tmp & 0x02U ) {
                dat |= 0x10U;
            } else {
                /* Do nothing */
            }
           /* set 2 segment at one time */
            LCD_WriteBuf( (LCD_BufIndex)(buf_head + idx) , dat );
            tmp >>= 2U;
         }
         buf_head += 4U;       /* for the static LCD, one digit '8' need 8 seg, */
                /*    but we process 2 seg at one time, so, only need to add 4 here */
    }
}
```

7-6 LVD

7-6-1 LVD Demo

This example implements power supply voltage status detecting by LVD.

When the power supply voltage is normal, LED0 will blink, LED1 is off.
When the power supply voltage is lower than the detection voltage, it will cause an interrupt: INTLVD, the LED0 will be turned off and LED1 will blink.

Test steps:

1. set the power supply voltage large than 3.1V, then power up, LED0 will blink.
2. adjust power supply voltage down to 2.9V, LED1 will blink and LED0 is off.
3. adjust power supply voltage back to 3.1V, LED0 will blink and LED1 is off.

Note: be care for that the power supply must be connected to M061 power pin directly, don't use any voltage regulator. Connect as below:

power supply '+' to M061 pin10 or 74
power supply '-' to M061 pin8 or 75

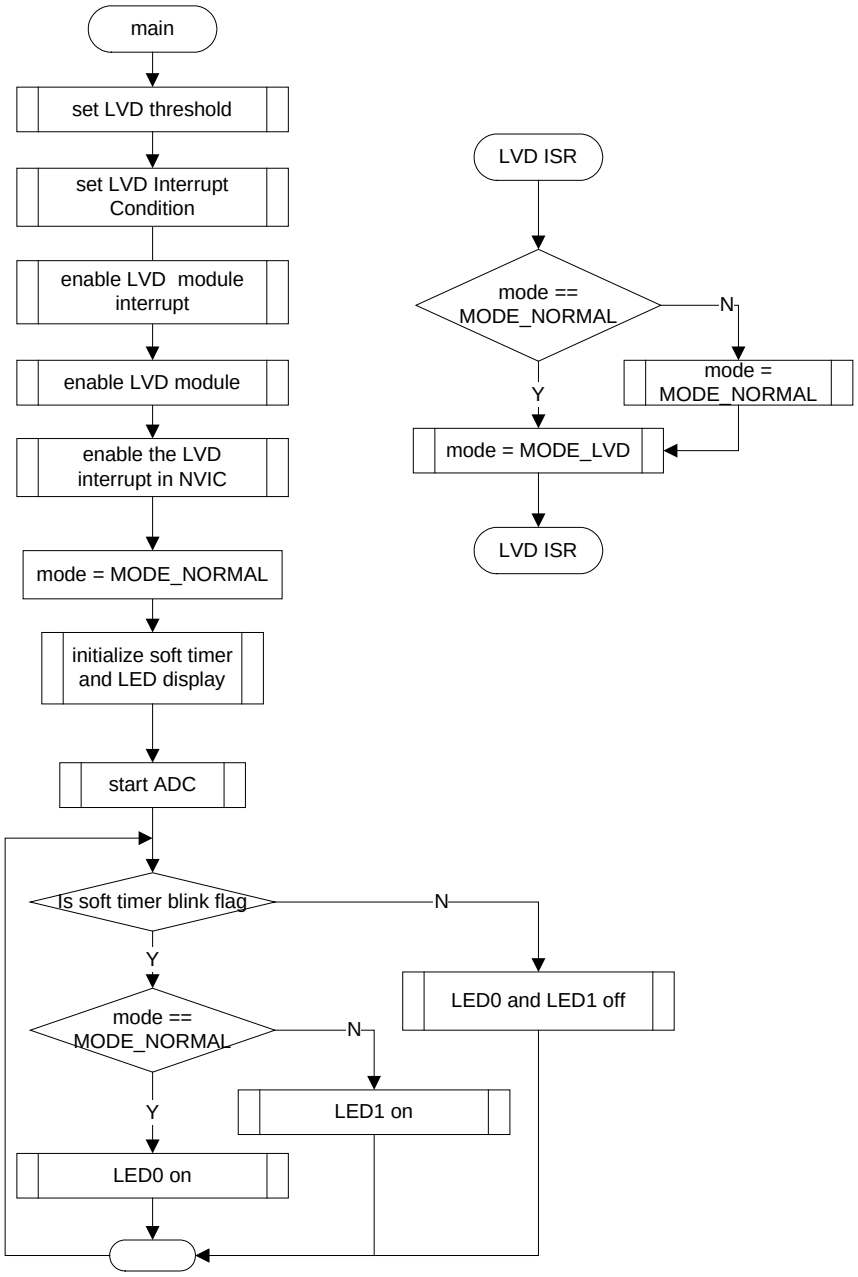
7-6-2 Example: LVD Demo

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

The example includes:

1. LVD configuration and initialization
2. LVD interrupt

- **Flowchart:**



- **Code and Explanation for the Example**

At first, set LVD threshold as 3.00 +/- 0.2V, and set LVD Interrupt Condition: both side,

```
/* 1. set LVD threshold as 3.00 +/- 0.2V */
LVD_SetVoltage(LVD_DETECT_VOLTAGE_300);

/* 2. set LVD Interrupt Condition: both side */
LVD_SetINTCondition(LVD_INTSEL_LOWER_UPPER);
```

Then enable LVD interrupt and enable LVD module:

```
/* 3. enable LVD module interrupt */
LVD_SetINTOutput(ENABLE);

/* 4. enable LVD module */
LVD_Enable();
```

Dont' forget to enable the LVD interrupt in NVIC :

```
NVIC_EnableIRQ(INTLVD_IRQn);
```

After that, initialize LED display and soft timer

```
/* when power is up, mode is 'normal' */
mode = MODE_NORMAL;

/* initialize LEDs on board before display something */
LED_Init();

/* initialize soft timer */
softTimer_Init();
```

To use the standard “printf()” and soft timer, we need to do some initialize:

```
/* initialize and retarget the 'printf()' to UART */
Retarget_Init();

/* initialize soft timer */
softTimer_Init();
```

In the main loop, if the blink flag is set, LED0 or LED1 is light depend on current mode is normal mode or LVD mode. otherwise, both LEDs are off.

```
while (1U) {

    softTimer_Run();
    if(softT.flag_Blink) {
        if( mode == MODE_NORMAL ) {
            LED_On(LED0);
        } else if ( mode == MODE_LVD ) {
            LED_On(LED1);
        } else {
            /* Do nothing */
        }
    } else {
        LED_Off(LED0);
        LED_Off(LED1);
    }

}
```

below is the ISR for LVD. If the power supply is down from normal to below 2.9V(set from part 1 above), it will cause the INTLVDs:

```
void INTLVD_IRQHandler(void)
{
    /* the first time enter this routine, power supply is lower than threshold */
    if( mode == MODE_NORMAL ) {
        mode = MODE_LVD;
    } else if ( mode == MODE_LVD ) { /* the second time enter thsi routine, power
supply is rising and higher than threshold */
        mode = MODE_NORMAL;
    } else {
        /* Do nothing */
    }
```

```
}
```

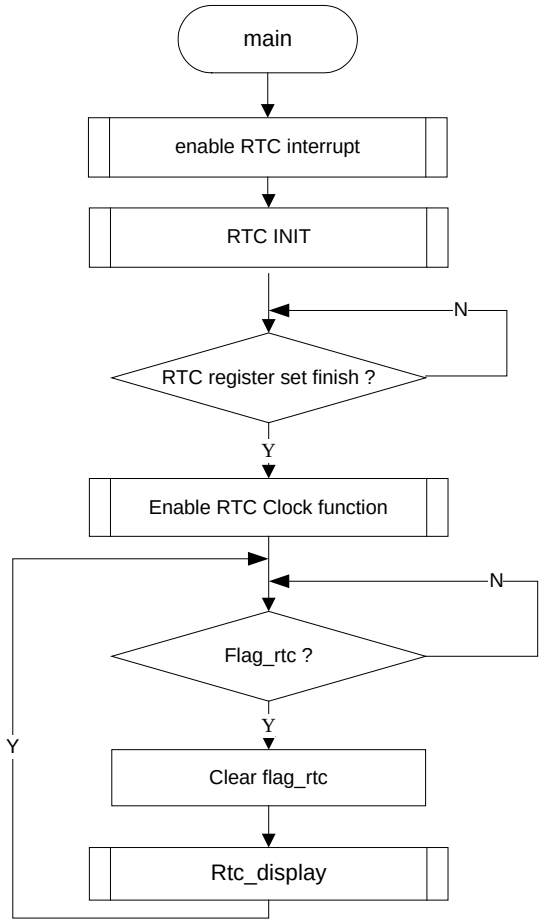
7-7 RTC

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

The example includes:

1. RTC initialization
2. Get RTC date and time

- **Flow chart:**



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

At first, initialize RTC. Create RTC_DateTypeDef and RTC_TimeTypeDef structure and fill all the data fields. For example, Initial setting: 2010/10/22 12:50:55, 24hours format.

```
Date_Struct.LeapYear = RTC_LEAP_YEAR_3;
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) 12U;
Time_Struct.Min = (uint8_t) 50U;
Time_Struct.Sec = (uint8_t) 55U;
```

Disable clock and alarm function.

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

Reset RTC second counter, enable 1Hz interrupt, and enable RTCINT.

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

Set RTC time and date value

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

After the setting above, enable RTC interrupt. Wait for 1second for RTC set finished, and then enable RTC Clock function.

```
NVIC_EnableIRQ(INTRTC_IRQn);
RTC_EnableClock();
```

In RTC ISR routine, the RTC interrupt will happen every second. Then RTC interrupt request will be cleared.

```
fRTC_1HZ_INT = 1U;
CG_ClearINTReq(CG_INT_SRC_RTC);
```

After interrupt, RTC date and time value will be sent to LCD.

Following is shown how to get RTC date and time value.

```
Year = RTC_GetYear();
Month = RTC_GetMonth();
Date = RTC_GetDate(RTC_CLOCK_MODE);
Hour = RTC_GetHour(RTC_CLOCK_MODE);
Min = RTC_GetMin(RTC_CLOCK_MODE);
Sec = RTC_GetSec();
```

7-8 SBI

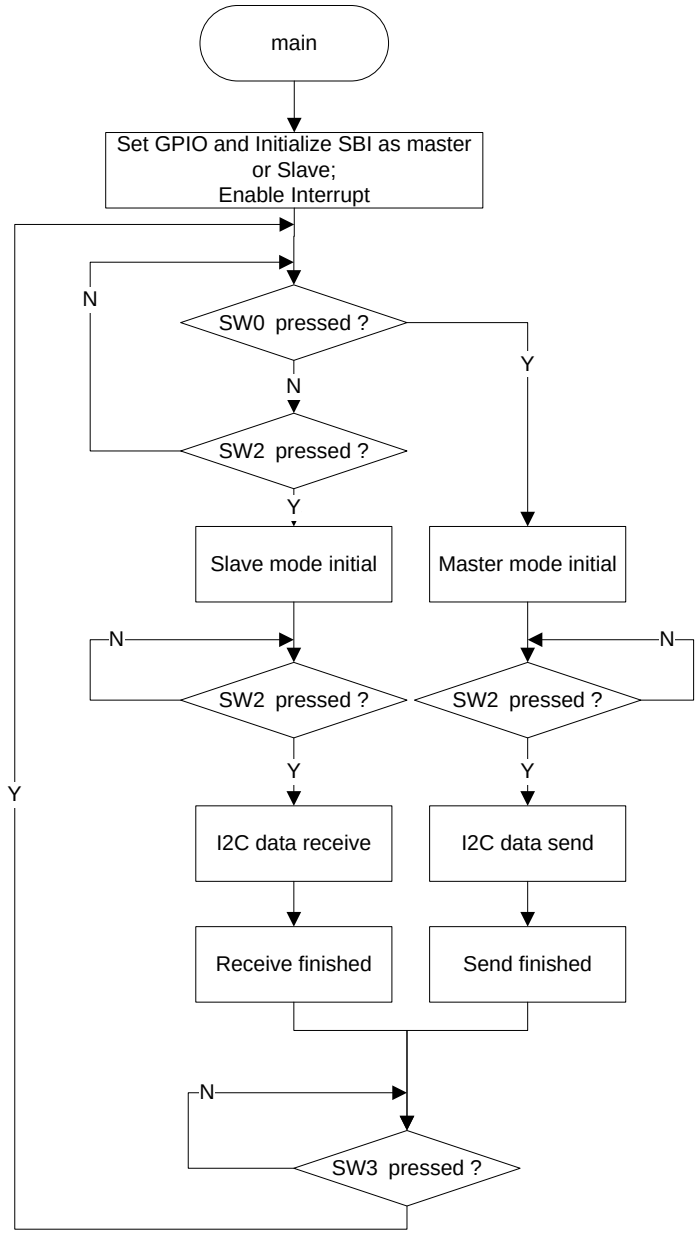
7-8-1 Example: SBI Slave

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

The example includes:

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

- **Flowchart**



- **Code and Explanation for the Example**

At first, configure GPIO for SBI I2C mode.

```
TSB_PJ->FR1 |= 0x03U;
TSB_PJ->CR |= 0x03U;
TSB_PJ->IE |= 0x03U;
TSB_PJ->OD |= 0x03U;
TSB_PJ->PUP |= 0x03U;
```

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;

    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_SBI);
        SBI_SWReset(TSB_SBI);
        SBI_InitI2C(TSB_SBI, &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_SBI);
        SBI_SWReset(TSB_SBI);
        SBI_InitI2C(TSB_SBI, &myI2Cs);
        gSBIMode = MODE_SBI_I2C_ TRX;
          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_SBI) is used to to start I2C process.

```
        /* Check I2C bus state and start TRx */
    case MODE_SBI_I2C_START:
        i2c_state = SBI_GetI2CState(TSB_SBI);
        if (!i2c_state.Bit.BusState) {
            SBI_SetSendData(TSB_SBI, SLAVE_ADDR | SBI_I2C_SEND);
            SBI_GenerateI2CStart(TSB_SBI);
            gSBIMode = MODE_SBI_I2C_TRX;
        } 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 */
        }
    }
```

7-9 TMR16A

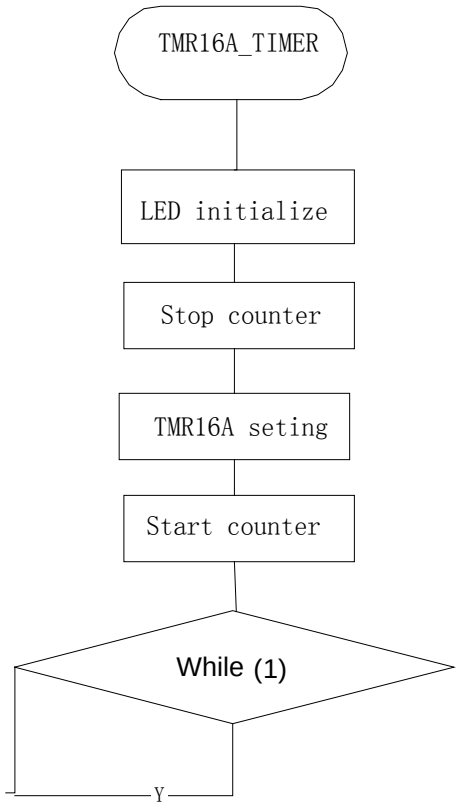
7-9-1 Example: General Timer

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

The example includes:

1. TMR16A setting
2. General Timer 1ms

- **Flow Chart**



- **Code and Explanation for the Example**

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

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

First stops the counter. This demo will set cycle to 1ms, macro TMR16A_1MS equals 0x3E80U, because ftmrb = fphiT0 = fsys = fc = 16MHz,, Ttmrb = 1/16us, 1ms*16 us =

16000 = 0x3E80 (Please see CG part for more detail information about the clock setting)
Then run the counter.

```
        TMR16A_SetRunState(TSB_T16A0, TMR16A_STOP);    /* counter stops*/
        TMR16A_SetSrcClk(TSB_T16A0, TMR16A_SYSCK);     /* set source clock is
system clock */
        TMR16A_ChangeCycle(TSB_T16A0,  TMR16A_1MS);      /*  Set  a  counter
value is 1ms*/
        /* Before starting counter operation, set "0x0000" to T16AxCP<CP> to clear the
counter*/
        TMR16A_ClrCaptureValue(TSB_T16A0);
        NVIC_EnableIRQ(INTT16A0_IRQn);
        TMR16A_SetRunState(TSB_T16A0, TMR16A_RUN);
```

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

```
    tbcount++;
    if (tbcount >= 500U) {                              /* 500ms is up */
        tbcount = 0U;
        /* reverse LED output */
        ledon = (ledon == 0U) ? 1U : 0U;
        if (0U == ledon) {
            LedOff(LED11);
        } else {
            LedOn(LED11);
        }
    } else {
        /* do nothing */
    }
```

7-10 TMRB

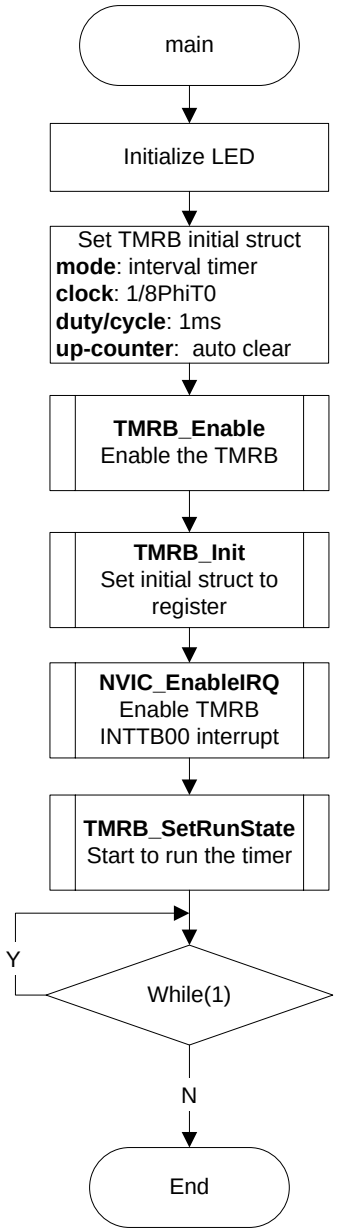
7-10-1 Example: General Timer

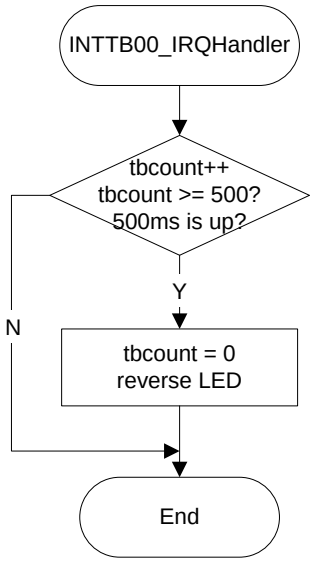
This is a simple example based on the TX00 Peripheral Driver (TMRB).

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 board and turn on LED.

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

Prepare TMRB initialization structure, fill TMRB mode, clock, up-counter clear method, trailing timing and leading timing. This demo will set trailing timing and leading timing to 1ms, macro TMRB_1MS equals 0x7d0, because fphiT0 = fsys = fc = 16MHz, ftmrb = 1/8 fphiT0 = 2MHz, Ttmrb = 0.5us, 1ms/0.5us = 2000 = 0x7d0 (Please see CG part for more detail information about the clock setting)

```
TMRB_InitTypeDef m_tmrb;

m_tmrb.Mode = TMRB_INTERVAL_TIMER;      /* internal timer */
m_tmrb.ClkDiv = TMRB_CLK_DIV_8;         /* 1/8PhiT0 */
```

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

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

```
TMRB_Enable(TSB_TB0);                              /* enable the TMRB0 */
TMRB_Init(TSB_TB0, &m_tmrb);                     /* initial the TMRB0 */
NVIC_EnableIRQ(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) {
            LedOff(LED11);
      } else {
            LedOn(LED11);
      }
} else {
      /* do nothing */
}
```

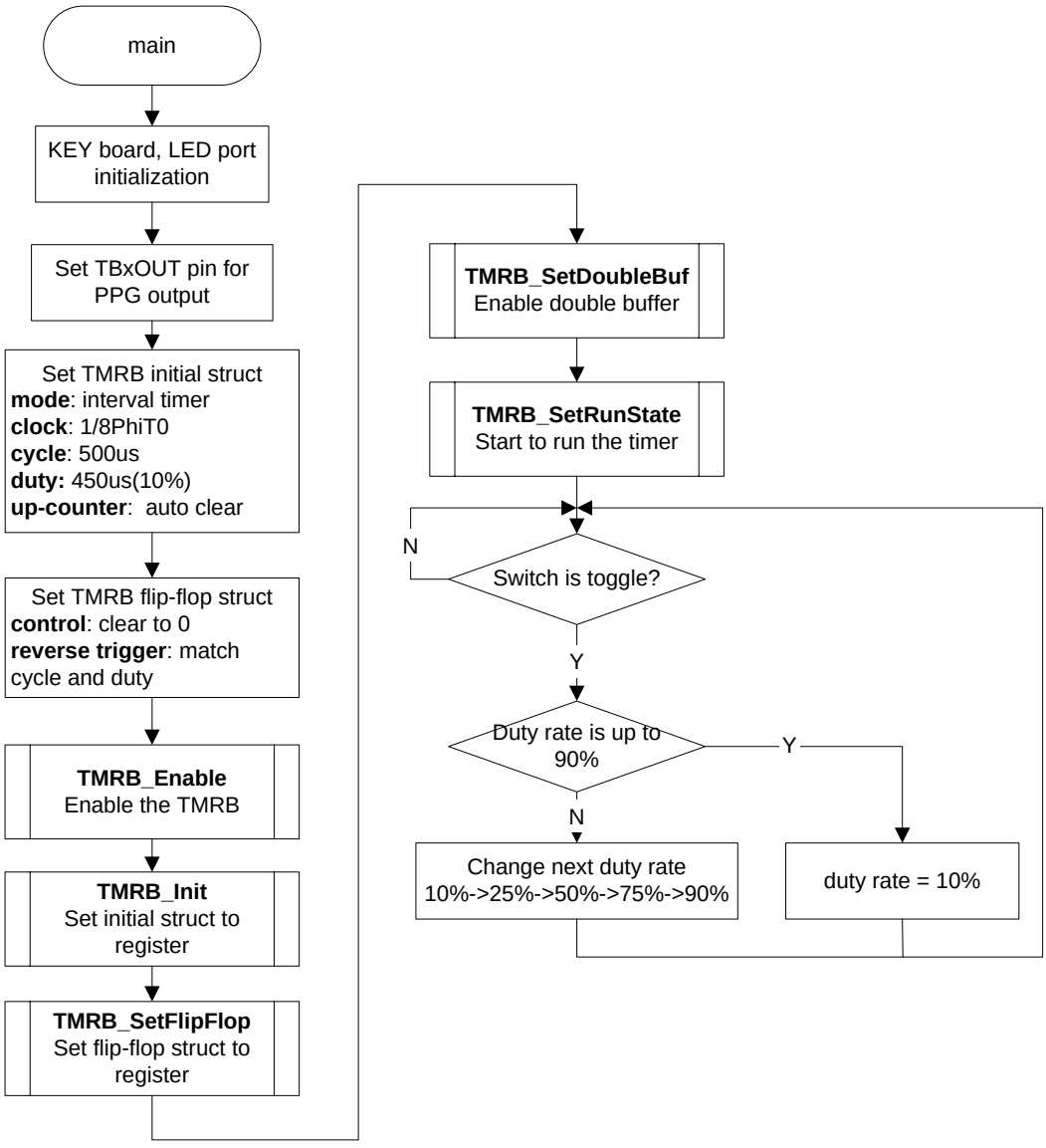
7-10-2 Example: PPG Output

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

The example includes:

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

- **Flow Chart**



- **Code and Explanation for the Example**

At first, set PG0 as TB0OUT for PPG output.

```
TSB_PG->CR |= 0x01;
TSB_PG->FR1 |= 0x01;
```

Prepare TMRB initialization structure, and then fill TMRB mode, clock, up-counter clear method, trailing timing and leading timing into it. This demo will set trailing timing to 500us, macro TMRB0TIME equals 0x03e8, because fphiT0 = fsys = fc = 16MHz, ftmrb = 1/8 fphiT0 = 2MHz, Ttmrb = 0.5us, 500us/0.5us = 1000 = 0x03e8 (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 = TMRB0TIME;        /* trailing timing is 500us */
m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;       /* up-counter auto clear */
m_tmrb.LeadingTiming = LeadingTiming[Rate]; /* leading timing, initial value 10% */
```

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

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

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

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

Wait switch from low to high.

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

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

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

TMRB_ChangeLeadingTiming(TSB_TB0, LeadingTiming[Rate]); /* change
leading timing rate */
```

The calculation method of leading timing value:
TrailingTiming = 500us, ftmrb = 1/8 fphiT0 = 2MHz, Ttmrb = 0.5us (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 = 0x384
LeadingTiming = 25%: high-level time is 500*25% = 125us, low-level time is 500-125 =

375us, counter value = 375us/Ttmrb = 0x2EE

LeadingTiming = 50%: high-level time is 500*50% = 250us, low-level time is 500-250 = 250us, counter value = 250us/Ttmrb = 0x1F4

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

LeadingTiming = 90%: high-level time is 500*90% = 450us, low-level time is 500-450 = 50us, counter value = 50us/Ttmrb = 0x64

That is the calculation method of leading timing array

```
uint32_t LeadingTiming[5] = { 0x384U, 0x2EEU, 0x1F4U, 0xFAU, 0x64U };
/* leading timing: 10%, 25%, 50%, 75%, 90% */
```

7-11 SIO

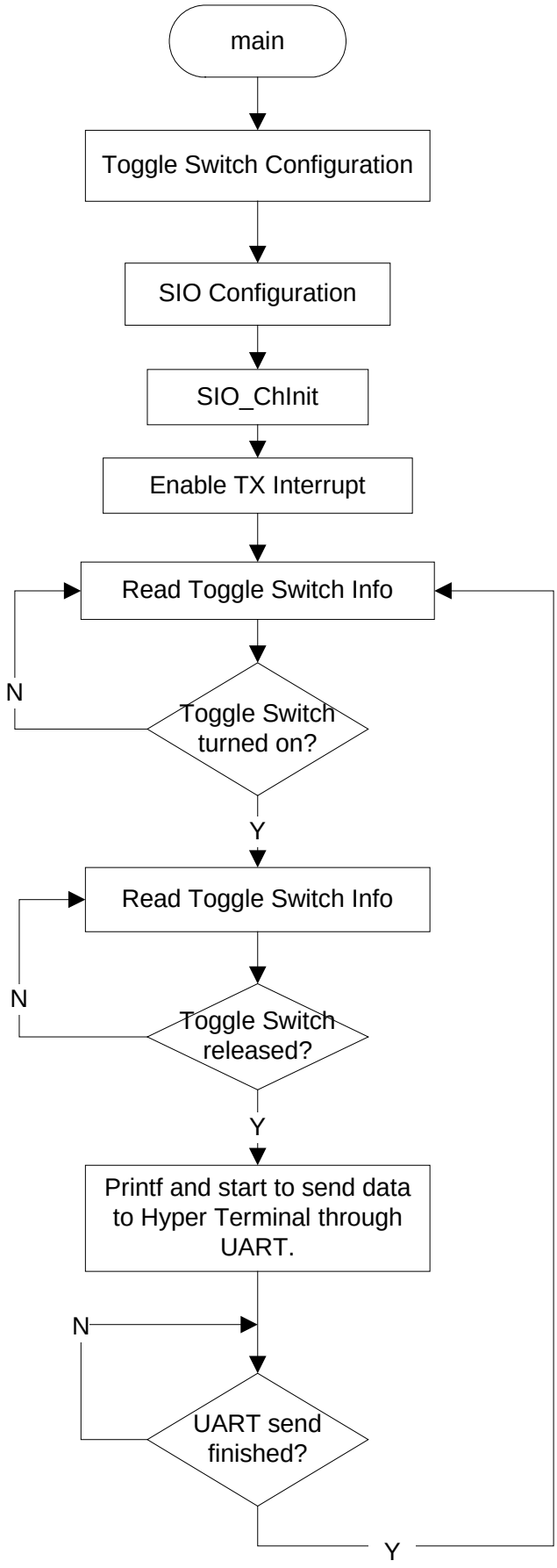
7-11-1 Example: Retarget

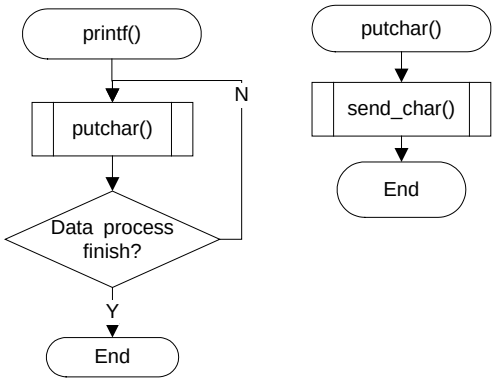
This is a simple example based on the TX00 Peripheral Driver (UART).

The example includes:

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

- **Flowchart**


main
Toggle Switch Configuration
SIO Configuration
SIO_ChInit
Enable TX Interrupt
Read Toggle Switch Info
N
Toggle Switch turned on?
Y
Read Toggle Switch Info
N
Toggle Switch released?
Y
Printf and start to send data to Hyper Terminal through UART.
N
UART send finished?
Y



- **Code and Explanation for the Example**

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

```
TSB_PH->CR |= GPIO_BIT_0;
TSB_PH->FR1 |= GPIO_BIT_0;
TSB_PH->FR1 |= GPIO_BIT_1;
TSB_PH->IE |= GPIO_BIT_1;
```

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;              /* SIO1 INT is enable */
    } else {
        /* disable SIO1 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;
}
```

- **Code and Explanation for the Example**

Below is the main.c for the flowchart above.

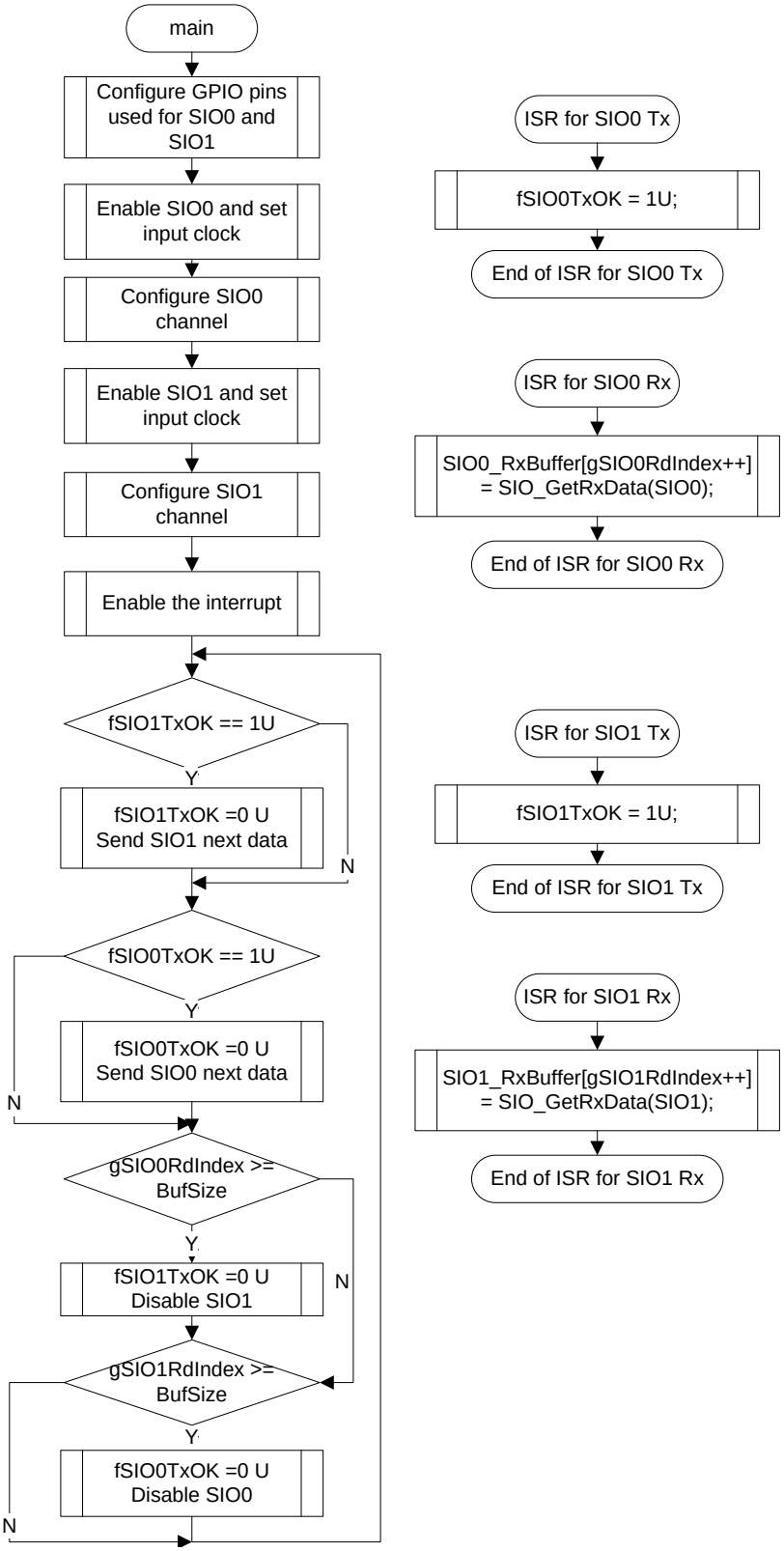
```
int main(void)
{
      GPIO_SetSSP();
      initSSP();
      InitDMA();

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

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

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

7-11-2 Example: SIO

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

The example includes:
1. Basic setup operation of SIO
2. The data transfer between SIO0 and SIO3
3. SIO interrupt of Tx and Rx

- **Flowchart**



- **Code and Explanation for the Example**

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

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

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

/*initialize the SIO0 struct */
SIO0_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO0_Init.IntervalTime = SIO_SINT_TIME_SCLK_8;
SIO0_Init.TransferMode = SIO_TRANSFER_FULLDPX;
SIO0_Init.TransferDir = SIO_LSB_FRIST;
SIO0_Init.Mode = SIO_ENABLE_TX | SIO_ENABLE_RX;
SIO0_Init.DoubleBuffer = SIO_WBUF_ENABLE;
SIO0_Init.BaudRateClock = SIO_BR_CLOCK_T4;
SIO0_Init.Divider = SIO_BR_DIVIDER_2;
SIO_Init(SIO0, SIO_CLK_BAUDRATE, &SIO0_Init);
```

Enable SIO3 configure the input clock and SIO3 initialization structure.

```
/*Enable the SIO3 channel */
SIO_Enable(SIO3);

/*initialize the SIO3 struct */
SIO3_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO3_Init.TransferMode = SIO_TRANSFER_FULLDPX;
SIO3_Init.TransferDir = SIO_LSB_FRIST;
SIO3_Init.Mode = SIO_ENABLE_TX | SIO_ENABLE_RX;
SIO3_Init.DoubleBuffer = SIO_WBUF_ENABLE;

SIO_Init(SIO3, SIO_CLK_SCLKINPUT, &SIO3_Init);
```

Enable the interrupt of SIO TX, RX.

```
/* Enable SIO0 Channel TX interrupt */
NVIC_EnableIRQ(INTTX0_IRQn);
/* Enable SIO3 Channel RX interrupt */
NVIC_EnableIRQ(INTRX3_IRQn);

/* Enable SIO3 Channel TX interrupt */
NVIC_EnableIRQ(INTTX3_IRQn);
/* Enable SIO0 Channel RX interrupt */
NVIC_EnableIRQ(INTRX0_IRQn);
```

After all the basic configure, Enter the data transfer routine .

```
while (1) {
        /* SIO3 send data from TXD3*/
        if (fSIO3TxOK == 1U) {
            fSIO3TxOK = 0U;
            SIO_SetTxData(SIO3, SIO3_TxBuffer[gSIO3WrIndex++]);
        } 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) {
```

```
            fSIO3TxOK = 0U;
            SIO_Disable(SIO3);
        } else {
            /*Do Nothing */
        }
        /*SIO3 receive data end */
        if (gSIO3RdIndex >= 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 SIO3Tx, set transfer is ok.

```
void INTTX3_IRQHandler(void)
{
    fSIO3TxOK = 1U;
}
```

In ISR of SIO3Rx, get the receive data from RX buffer.

```
void INTRX3_IRQHandler(void)
{
    SIO0_RxBuffer[gSIO3RdIndex++] = SIO_GetRxData(SIO3);
}
```

7-12 WDT

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

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();
```