TOSHIBA

TX04 Peripheral Driver Usage Example (TMPM46B)

Rev 1.000

Sep, 2017

TOSHIBA ELECTRONIC DEVICES & STORAGE CORPORATION

CMDR-M46BUE-00xE

RESTRICTIONS ON PRODUCT USE

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

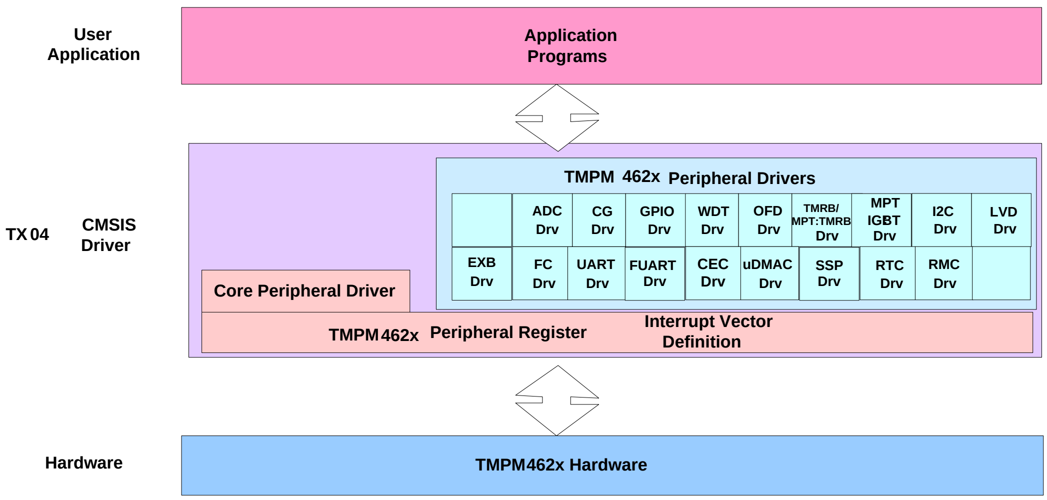
Index

1 General description

The example programs described hereafter are specially designed for TOSHIBA TMPM46B 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 peripheral driver as following.



3 Build-in hardware usage

Hardware	Channel	Use presence, use
CG	Clock Gear	Used for CG demo
	PLL	PLL on (6x)
Standby mode	-	Used for CG demo (IDLE / STOP1 / STOP2 mode)
SysTick	-	Unused
Watch dog timer (WDT)	-	Used for WDT
Interrupt (INT)	INT0	Unused
	INT1	Unused
	INT2	Unused
	INT3	Unused
	INT4	Unused

Hardware	Channel	Use presence, use
	INT5	Unused
	INT6	Unused
	INT7	Unused
	INT8	Unused
	INT9	Unused
	INTA	Unused
	INTB	Unused
	INTC	Unused
	INTD	Unused
	INTE	Unused
	INTF	Used for CG demo to wake up system from stop mode
	INTRTC	Used for RTC demo
	INTRX0	Used for SIO demo
	INTTX0	Used for SIO demo
	INTRX1	Used for SIO demo
	INTTX1	Used for SIO demo
SIO	SIO0	Used for UART retarget demo/ SIO demo/ UART FIFO demo
	SIO1	Used for SIO demo
	SIO2	Unused
	SIO3	Used for UART FIFO demo
16-bit timer	TMRB0	Used for TMRB: General Timer
	TMRB1	Unused
	TMRB2	Unused
	TMRB3	Unused
	TMRB4	Unused
	TMRB5	Unused
	TMRB6	Used for TMRB: PPG Output
	TMRB7	Unused
LVD	-	Used for LVD demo
12-bit A/D converter	AIN0	Used for ADC demo
	AIN1	Unused
	AIN2	Unused
	AIN3	Unused
	AIN4	Unused
	AIN5	Unused
	AIN6	Unused
	AIN7	Unused
uDMAC	-	Used for DMAC demo
Real-time clock	-	Used for RTC demo

Hardware	Channel	Use presence, use
I2C	I2C0	Used for I2C demo: receiver
	I2C1	Unused
	I2C2	Used for I2C demo: transmitter
SSP	SSP0	Used for SSP0 self loopback demo
	SSP1	Unused
	SSP2	Unused
Full UART	FUART0	Used for Full UART demo
	FUART1	Unused
Multi-purpose timer (MPT)	MPT0	Used for IGBT demo: PPG output
	MPT1	Unused
	MPT2	Unused
	MPT3	Unused
External bus interface	-	Used for EXB demo: read&write SRAM
AES	-	Used for AES demo
SHA	-	Used for SHA demo
MLA	-	Used for MLA demo
ESG	-	Used for ESG demo

4 Pin Usage

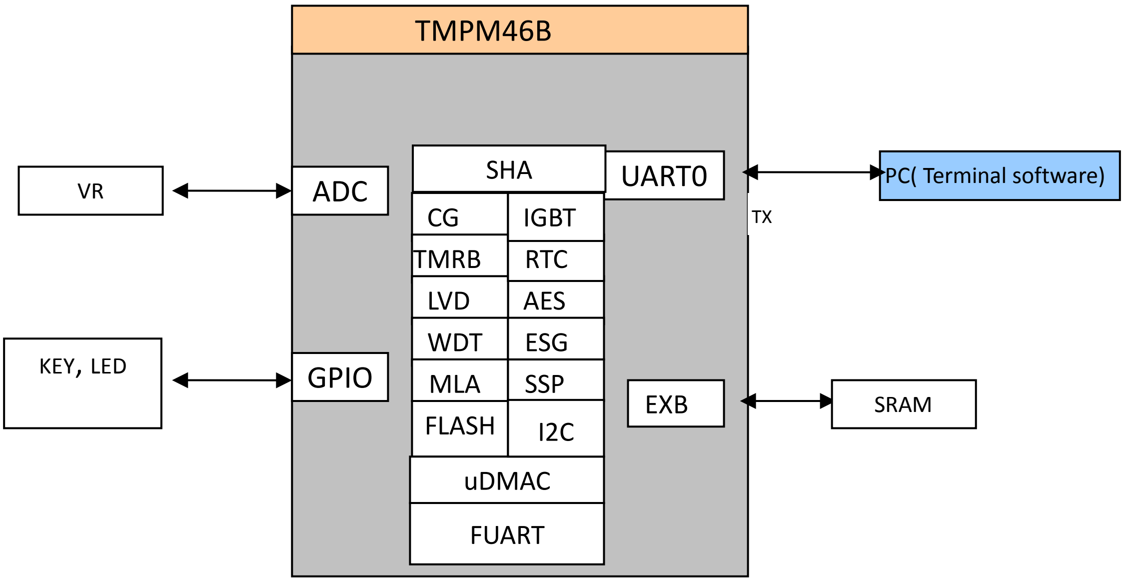
The example programs are tested on TMPM46B evaluation board
Following is pin usage for example programs

Pin No.	Name	Usage
23	PF0	EBIF AD0 / FUART UT0CTS
24	PF1	EBIF AD1 / FUART UT0TXD
25	PF2	EBIF AD2 / FUART UT0RXD
26	PF3	EBIF AD3 / FUART UT0RTS
27	PF4	EBIF AD4
28	PF5	EBIF AD5
29	PF6	EBIF AD6
30	PF7	EBIF AD7
31	PG0	EBIF AD8
32	PG1	EBIF AD9
33	PG2	EBIF AD10
34	PG3	EBIF AD11
35	PG4	EBIF AD12
36	PG5	EBIF AD13
37	PG6	EBIF AD14
38	PG7	EBIF AD15

88	PB0	LED0 / SC3TXD
87	PB1	LED1 / SC3RXD
43	PB2	LED2 / EBIF WR
44	PB3	LED3 / EBIF RD
1	PJ0	SW0/SW4 / AIN0
2	PJ1	SW1/SW5
3	PJ2	SW2/SW6
4	PJ3	SW3/SW7
13	PE0	EBIF A16
14	PE1	EBIF A17 / SC0RXD
15	PE2	EBIF A18 / SC0TXD
16	PE3	EBIF A19 / SC0SCK
17	PE4	EBIF A20 / SC1SCK
18	PE5	EBIF A21 / SC1TXD
19	PE6	EBIF A22 / SC1RXD
20	PE7	EBIF A23
46	PB5	EBIF ALE
51	PH1	EBIF CS1 / I2C2SCL
47	PB6	EBIF BELL
52	PH0	EBIF BELH / I2C2SDA
31	PG0	IGBT MT0IN
34	PG3	IGBT MT0OUT0
33	PG2	IGBT MT0OUT1
32	PG1	IGBT GEMG0
80	PK2	I2C0SDA
79	PK3	I2C0SCL
81	PK1	TB6OUT
59	X1	High frequency resonator connection pin
61	X2	High frequency resonator connection pin
69	XT1	Low frequency resonator connection pin
70	XT2	Low frequency resonator connection pin
62	MODE	MODE pin
58	RESET	Reset signal input pin
47	BOOT	BOOT mode control pin

5 Development Environment

Following is development environment:



1. Development tool:
   - IAR:
     1) J-Link: IAR J-Link-ARM 7.0 / J-Link 6.0
     2) IDE:IAR Embedded workbench 7.30.4 version
   - KEIL:
     1) IDE: KEIL uVision 5.14

6 Functional description

6-1 Operation mode

There are four operation modes for TMPM46B: NORMAL, IDLE, STOP1, STOP2 mode.

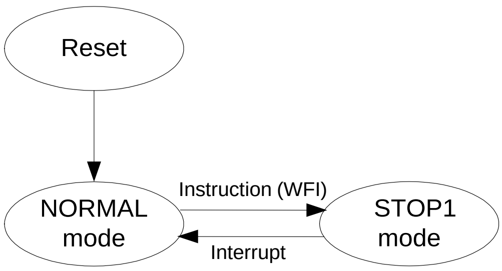
- **NORMAL mode:**

This mode is to operate the CPU core and the peripheral hardware by using the high-speed clock. It is shifted to the NORMAL mode after reset.

IDLE, STOP1, STOP2 mode are low power mode.
To shift to lower power mode, in system control register CGSTBYCR<STBY[2:0]>, select the IDLE or STOP1 or STOP2 mode, and run the WFI (Wait For Interrupt) command.
***Note:** Only STOP1 mode is demonstrated in driver example.

- **STOP1 mode:**

All the internal circuits including the internal oscillator are brought to a stop in STOP1 mode. When releasing STOP1 mode, an internal oscillator begins to operate and the operation mode changes to NORMAL mode.
***Note:** The sample program of STOP1 mode is integrated into CG example.



6-2 ADC

Change the value of VR2, which is connected to AIN0, the voltage on it will be measured and used to change the loop blinking speed of the 4 LEDs (see **Note** below) in board.
The higher the voltage is, the faster the loop blinks.
The driver example step:

1. Software reset ADC.
2. Enable ADC clock supply.
3. Set ADC clock prescale and sample hold time.
4. Select ADC input channel : AIN0
5. Enable ADC repeat mode.
6. Set interrupt mode.
7. Turn VREF on.

8. Start ADC.
9. If ADC is finished, get the result and use it to adjust the LED loop blinking speed.

***Note**: LED0~LED3 connect with PB0~PB3 in TMPM46B MCU board.

6-3 AES

6-3-1 AES ECB mode

This is a simple example to show the procedure of ECB encryption and decryption. AES processor is set to ECB mode. A 384-bit plaintext will be encrypted into a 384-bit encrypted text and then the 384-bit encrypted text will be decrypted into a 384-bit decrypted text. If plaintext equals to decrypted text it means ECB procedure processed successfully, otherwise ECB procedure failed.
The AES processor encrypts/decrypts plaintext or encrypted text in units of 128-bit blocks, therefore the text will be divided in units of 128-bit. Reading/writing data from/to the FIFO is performed by the DMA controller.

6-3-2 AES CBC mode

This is a simple example to show the procedure of CBC encryption and decryption. AES processor is set to CBC mode. A 384-bit plaintext will be encrypted into a 384-bit encrypted text and then the 384-bit encrypted text will be decrypted into a 384-bit decrypted text. If plaintext equals to decrypted text it means CBC procedure processed successfully, otherwise ECB procedure failed.
The AES processor encrypts/decrypts plaintext or encrypted text in units of 128-bit blocks, therefore the text will be divided in units of 128-bit. In CBC mode input the same initial vector both on encryption and decryption. Reading/writing data from/to the FIFO is performed by CPU.

6-3-3 AES CTR mode

This is a simple example to show the procedure of CTR encryption and decryption.
AES processor is set to CTR mode. A 384-bit plaintext will be encrypted into a 384-bit encrypted text and then the 384-bit encrypted text will be decrypted into a 384-bit decrypted text. If plaintext equals to decrypted text it means CTR procedure processed successfully, otherwise CTR procedure failed.
The AES processor encrypts/decrypts plaintext or encrypted text in units of 128-bit blocks, therefore the text will be divided in units of 128-bit. In CTR mode input the same counter value and the encryption key both on encryption and decryption. Reading/writing data from/to the FIFO is performed by CPU.

6-4 CG

6-4-1 Mode switch STOP1

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

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

6-4-2 Mode switch IDLE

Change the CPU operation mode. Only 2 modes are supported, NORMAL and IDLE. Toggle switch to switch the power mode. (When releasing Reset, all SW are off).

Current mode	Action (Switch)	Operation	LED display
NORMAL	Turn off SW1	NORMAL→ IDLE	LED 3 turns off. LED 2 turns on.
IDLE	Turn on SW1 at first, turn on SW0	IDLE →NORMAL	LED 3 turns on. LED 2 turns off.

6-4-3 STOP2

Change the CPU operation mode. Only 2 modes are supported, NORMAL and STOP2. Toggle switch to switch the power mode. (When releasing Reset, all SW are off).

Current mode	Action (Switch)	Operation	LED display
NORMAL	Turn on SW1	NORMAL→ STOP2	LED 0, 3 turns off.
STOP2	Turn off SW1 at first, turn on SW0	STOP2 →NORMAL	LED 0, 3 turns on.

***Note**
LED0~LED3 connect with PB0~PB3 in TMPM46BF10FG MCU board.
SW0 connect with PC1 (Pin No.76) in TMPM46BF10FG MCU board to set external interrupt input (INTF).
SW1 connect with PJ1 (Pin No.2) in TMPM46BF10FG MCU board to control the switch power mode.

6-5 ESG

This is a simple example to show the ESG generates a 512-bit entropy seed.
Set entropy seed latch timing and entropy seed output timing. Get a 512-bit entropy seed using by calculating.

6-6 EXB

This function implements read/write the external SRAM assembled on TMPM46B evaluation board and EXB is set as 16-bit bus width on multiplex bus. The A.C specifications of SRAM (cycles time) used is for specific SRAM chip. The external SRAM is IS61LV6416 with 128Kbyte size. About connection pins please see **Note** below

***Note:**

1. AD[0:15] of TMPM46B evaluation board connect AD[0:15] of IS61LV6416 SRAM;
2. A16 of TMPM46B evaluation board connect A15 of IS61LV6416 SRAM;
3. CS1, WR, RD, ALE, BELL, BELH of TMPM46Bx evaluation board connect CE, WE, OE, ALE, LB, UB of SRAM.

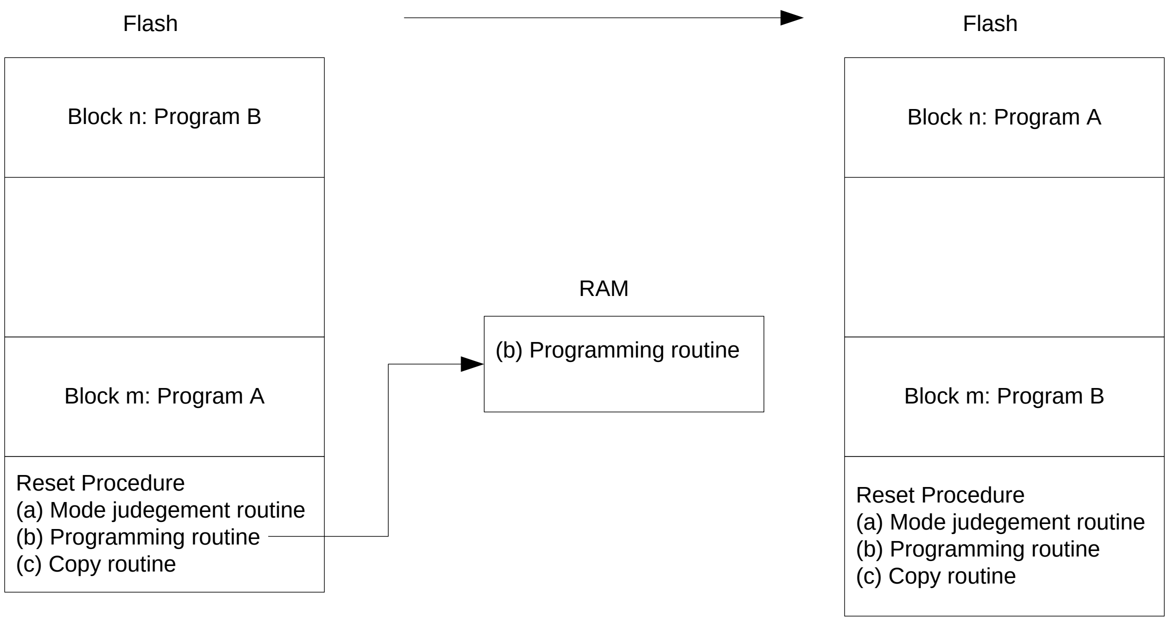
6-7 FLASH

6-7-1 FLASH_UserBoot

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 LED2 Always show,
B: LED1 blinks and LED3 Always show)
By default, the program A will run firstly
—Toggle switch SW4 is used for mode judgment in Reset procedure

SW4 turned on → User boot mode

SW4 turned off → Normal mode



Demo sequence

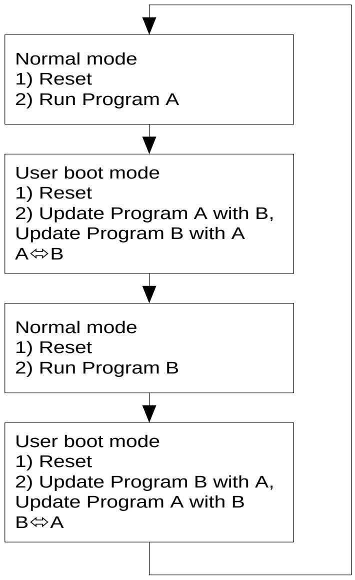
(1) Power on
The demo board runs Reset Procedure.
Then initial program A stored in flash runs.
LED0 blinks and LED2 Always show.

(2) Press RESET button while SW4 is turned on, and release SW4.
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.

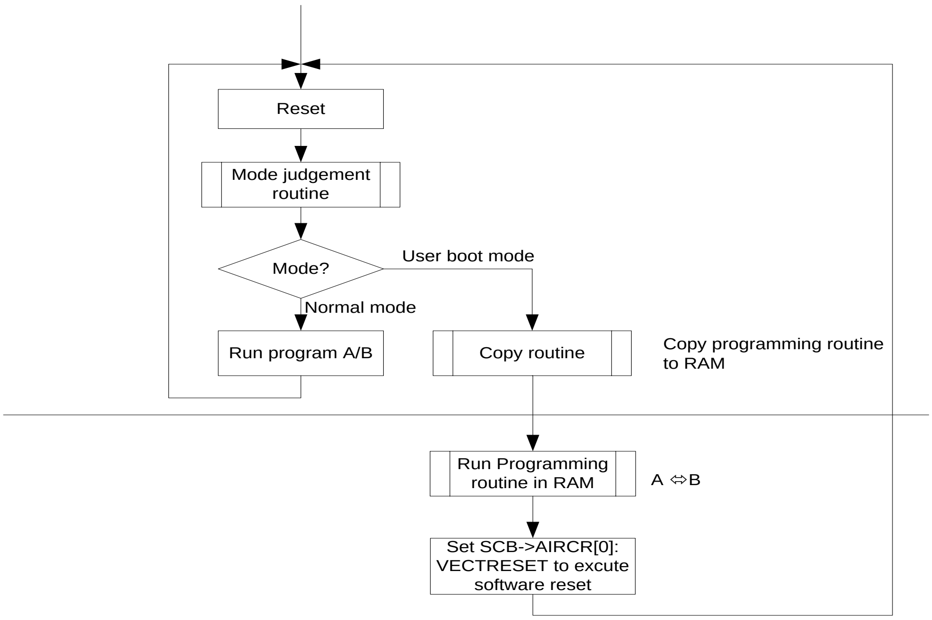
(3) The demo will reset by software automatically.
Then program B stored in flash runs.
LED1 blinks and LED3 Always show.

(4) Press RESET button again while SW4 is turned on, and release SW4.
Same as step (2).

(5) The demo will reset by software automatically.
Then program A stored in flash runs.
LED0 blinks and LED2 Always show.



Demo process flow



6-7-2 FLASH_Swap

This demo is an example which introduces user how to use flash driver and auto swap mode.

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

▪Mode judgment routine: judge to enter User Boot Mode or Normal Mode

▪Swap routine: runs at RAM to set the swap function or release swap function

▪LED flash routine: LED flash runs in Normal mode

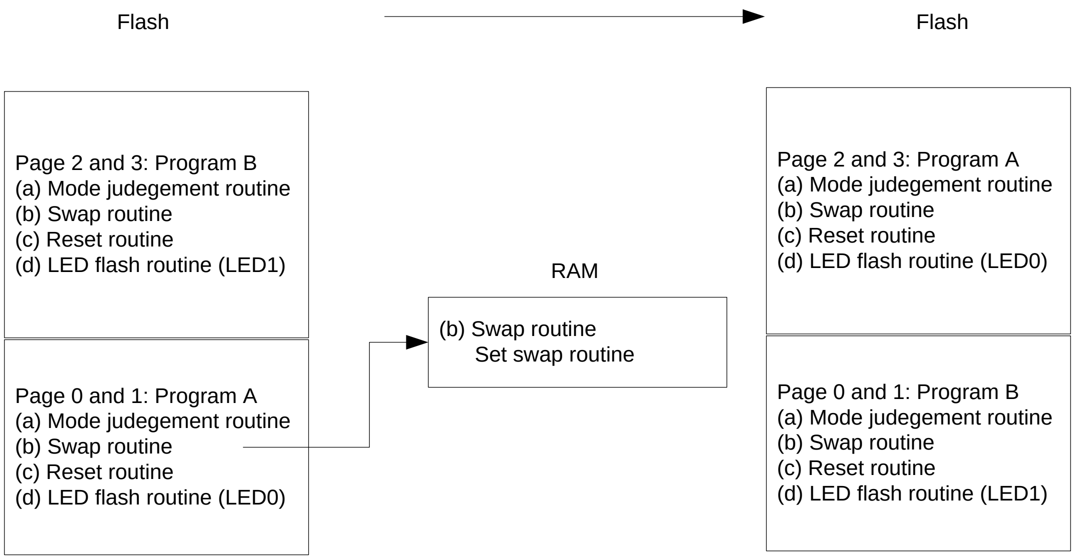
—Program A/B are programmed in flash block in advance.

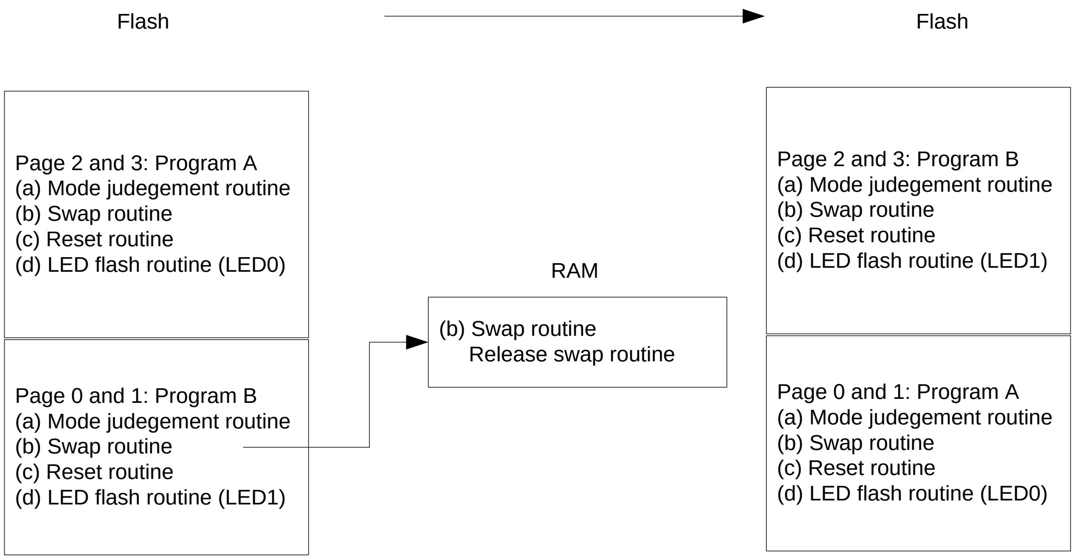
—Program A/B are same except LED flash routine.

—LED0 flash in program A, and LED1 flash in program B.
By default, the program A will run firstly
—Toggle switch SW4 is used for mode judgment.

SW4 turned on → User boot mode

SW4 turned off → Normal mode





Operation procedure:

1) Open IAR EWARM IDE, connect J-Link with PC and TOSHIBA M46B Evaluation Board, and then click menu: file->open->workspace, choose "Flash_Swp_A.eww" to open demo project.

2) Click menu: Project->Options->Debugger->Download, then check "Override

default .board file" option, click the path button to choose file "Flash_Swp_A\IAR\res\Flash_Swp_A_for_M46B.board", then press OK to close Options window.

3) Click menu: Project->Download->Download file, then choose file "Flash_Swp_A\IAR\res\Flash_Swp_A.out", click "open", then the IAR embedded flashloader will download Flash_Swp_A to specified block.

4) Repeat 2) and 3), choose .board file "Flash_Swp_A\IAR\res\Flash_Swp_B_for_M46B.board", and file "Flash_Swp_A\IAR\res\Flash_Swp_B.out" to download Flash_Swp_B to specified block.

5) Click "reset" button on board. You will watch LED0 blinks.

6) Hold the SW4turned on, then click "reset" button on board, you will watch LED2 light. (It indicates the swap is done)

7) Release SW4 You will watch LED1 blinks.

8) Hold the SW4 turned on, then click "reset" button on board, you will watch LED2 light. (It indicates the swap is release)

9) Release SW4, You will watch LED0 blinks.

***Note:**

<1> You can repeat step 6) to step 9) to display the demo again.
<2> You can create new Flash_Swp_A.out and Flash_Swp_B.out in Release mode of each project. (Due to the file size limited, the file can only be created in Release mode)
<3> LED0~LED2 connect with PB0~PB2 in TMPM46BF10FG MCU board.
<4> SW4 connect with PJ0 in TMPM46BF10FG MCU board.

6-8 FUART

In this program, both Full UART Transmit and Receive FIFO are enabled.
This program sends 64 different data value from Full UART channel 0 UT0TXD pin and receives data from UT0RXD 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 UT0RXD receives. Then the program will compare all the received data with transmitted data.
If received data are same with transmitted data, UART displays “SAME”.
If received data are different with transmitted data, UART displays “DIFFERENT”.

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

The first time:

Enable UT0RTS and UT0CTS hardware flow control, the received data are same with transmitted data.

The second time:

Disable UT0RTS and UT0CTS hardware flow control, the received data are different with transmitted data.

***Note:**

Connect UT0TXD pin with UT0RXD pin on TMPM46BF10 MCU board.
Connect UT0RTS pin with UT0CTS pin on TMPM46BF10 MCU board.
SW0 connect with PJ0 in TMPM46B MCU board.
Connect LED0~3 with PB0~3 on TMPM46B MCU board.

6-9 GPIO

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

***Note**:

LED0~LED3 connect with PB0~PB3 in TMPM46B MCU board.
SW0~SW3 connect with PJ0~PJ3 in TMPM46B MCU board.

6-10 I2C

This function intends to support the I2C bus slave mode.
Connect two I2C buses (I2C0 & I2C2) on evaluation board.
One works as I2C master, and another works as I2C slave.

Uses I2C interrupt to handle I2C bus read/write.

Address of I2C slave: Any.

I2C Slave (I2C0) receives “TOSHIBA” from Master (I2C2) and UART prints it.

I2C2 (master) and I2C0 (slave) demo start,

K3:I2C2 to I2C0

When SW7 is pressed, the string "TOSHIBA" will be sent from I2C2 (master) to I2C0 (slave).

```
Write Over
K1: Show I2C0
```

When SW4 is pressed, the string "TOSHIBA" that got from the I2C0 (slave) received buffer will be printed.

```
TOSHIBA
I2C2 to I2C0 OK
```

***Note:**
SW4 connect with PJ0 in TMPM46B MCU board.
SW7 connect with PJ3 in TMPM46B MCU board.
Connect pin PK2 and PH0 to use I2CxSDA.
Connect pin PK3 and PH1 to use I2CxSCL.

6-11 IGBT

6-11-1 Single PPG Output

This example program demonstrates how to use external trigger to control the PPG output by TMPM46B IGBT driver and EMG function. The MPT channel 0 (IGBT0) is used and the PPG wave is generated from MT0OUT0.

Pin assignment:

Function	Pin Name	Pin No.	PORT
Trigger input	MT0IN	31	PG0
PPG output	MT0OUT0	34	PG3
EMG input	GEMG0	32	PG1

Demo procedure:
1. Use a wire to connect port GEMG0 with DVDD to pull it high.
2. Connect MT0OUT0 to oscilloscope.
3. Power on, LED0 on and no waveform is observed from MT0OUT0.
4. Touch MT0IN with GND to produce a falling edge and keep it to low.
5. The IGBT timer starts to run because of the falling edge and at the same time PPG waveform can be observed from MT0OUT0. PPG cycle is 50us with duty 50%.
6. Connect GEMG0 with GND, then PPG output stops immediately and LED1 lights on.

7. Connect GEMG0 with DVDD again, and EMG state cancels with LED1 off, then repeat step 4 to observe the waveform.

6-11-2 Double PPG Output

This example program demonstrates how to use command start to control two PPG output by TMPM46Bx IGBT driver and EMG function. The MPT channel 0 (IGBT0) is used and two PPG waves are generated from MT0OUT0 and MT0OUT1.

Pin assignment:

Function	Pin Name	Pin No.	PORT
PPG output 0	MT0OUT0	34	PG3
PPG output 1	MT0OUT1	33	PG2
EMG input	GEMG0	32	PG1

Demo procedure:

1 Use a wire to connect port GEMG0 with DVDD to pull it high.
2 Connect MT0OUT0 and MT0OUT1 to oscilloscope.
3 Switch SW7 to OFF.
4 Power on, LED0 on and no waveforms are observed from MT0OUT0 or MT0OUT1.
5 Switch SW7 to ON.
6 The IGBT timer starts to run and at the same time PPG waveform can be observed from MT0OUT0 and MT0OUT1. Both PPG waveforms' cycle is 50us with duty 40 %(high level: 20us) and MT0OUT1 delays MT0OUT0 with 25us.
7 If switch SW7 to OFF, PPG waveform disappears.
8 Connect GEMG0 with GND, then PPG output stops immediately and LED1 lights on.
9 Connect GEMG0 with DVDD again, and EMG state cancels with LED1 off, then repeat step 5 to observe the waveform.

***Note**:
LED0 connect with PB0 in TMPM46BF10FG MCU board.
LED1 connect with PB1 in TMPM46BF10FG MCU board.

6-12 LVD

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

6-13 MLA

6-13-1 MLA Montgomery multiplication mode

This is a simple example to show the procedure of Montgomery multiplication mode.
MLA processor is set to Montgomery multiplication mode. Sets calculation input data to calculate the results through hardware. In addition to this, in this mode need to set the divisor and montgomery parameter.
Three groups calculation input data “OriginaAData, OriginaBData”, “TrueAData, TrueBData” and “FalseAData, FalseBData” will be calculated into “OriginaData”, “TrueData” and “FalseData” separately. The same calculation input data will generate same calculation results, and the different calculation input data will generate different calculation results. So the right result should be “OriginaData” = “TrueData” ≠ “FalseData”.

6-13-2 MLA Multiple length addition mode

This is a simple example to show the procedure of multiple length addition mode.
MLA processor is set to multiple length addition mode. Sets calculation input data to calculate the results through hardware.
Three groups calculation input data “OriginaAData, OriginaBData”, “TrueAData, TrueBData” and “FalseAData, FalseBData” will be calculated into “OriginaData”, “TrueData” and “FalseData” separately. The same calculation input data will generate same calculation results, and the different calculation input data will generate different calculation results. So the right result should be “OriginaData” = “TrueData” ≠ “FalseData”.

6-13-3 MLA Multiple multiplication subtraction mode

This is a simple example to show the procedure of multiple length subtraction mode.
MLA processor is set to multiple length subtraction mode. Sets calculation input data to calculate the results through hardware.
Three groups calculation input data “OriginaAData, OriginaBData”, “TrueAData, TrueBData” and “FalseAData, FalseBData” will be calculated into “OriginaData”, “TrueData” and “FalseData” separately. The same calculation input data will generate same calculation results, and the different calculation input data will generate different calculation results. So the right result should be “OriginaData” = “TrueData” ≠ “FalseData”.

6-14 RTC

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

UART prints:

```
2010/10/22
12:50:55
```

6-15 SHA

This is a simple example to show the SHA generates 256-bit Hash values from message data.

Three 1024-bit messages “OriginalMessage”, “TrueMessage” and “FalseMessage” will be calculated into Hash values “OHashValue”, “THashValue” and “FHashValue” separately. The same messages will generate same Hash values, and the different messages will generate different Hash values. So the right result should be “OHashValue” = “THashValue” ≠ “FHashValue”

Calculations are performed in unit of 512 bits, so 1024 bits message will be divided into 2 units. Message blocks are transferred by CPU, so the SHA core starts calculation by setting ‘1’ to SHASTA<START>.

6-16 SSP

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

UART prints:

```
SSP RX DATA:
xxxxxx
```

***Note:** LED0~LED3 connect with PB0~PB3 in TMPM46B MCU board.

6-17 TMRB

6-17-1 General Timer

This example implements a general timer utilizing MCU timer.

Timer trailingtiming is set to 1ms.

Use this timer for all LED blinking and the period is 1s (500ms ON and 500ms OFF).

6-17-2 PPG Output

Use key SW4 to change PPG waveform. LeadingTiming can be changed as following:

10%, 25%, 50%, 75%, 90% (UART prints)
Power on to enter PPG mode and starts the PPG output.
Change the leadingtiming upon every SW0 pressed.
10% → 25% → 50% → 75% → 90% → 10%

***Note**:
LED0~LED3 connect with PB0~PB3 in TMPM46B MCU board.
SW4 connect with PJ0 in TMPM46B MCU board.

6-18 SIO/UART

6-18-1 UART

This example intends to retarget the stdin and stdout of the C lib.
Stdin and stdout are retargeted to UART. Then application code can use printf() to output to serial port.
***Note**:
This UART channel is limited by hardware, the follow example used UART0.
SW_PORT is SW0, SW0 connect with PJ0 in TMPM46B MCU board.

6-18-2 UART FIFO

In this sample program, UART0 sent the data "TMPM46B1" to UART3 use FIFO, and at same time UART0 receive the data "TMPM46B2" 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 = "TMPM46B2",and RxBuffer1 = "TMPM46B1".
Note:
Connect TX of UART0 (pin15) with RX of UART3 (pin87), connect RX of UART0 (pin14) with TX of UART3 (pin88).

6-18-3 SIO

This demo will show synchronously transfer and receive usage of SIO module in TMPM46B.
It use the channel SIO0, SIO1 and transfer data synchronously between them.(connect TXD0 with RXD1, connect TXD1 with RXD0, connect sclk0 with sclk1)

6-19 µDMAC

This demo will show how to reserve 1K RAM area for control data of each uDMAC unit, and use DMA with software trigger to transfer data from RAM area "src" to "dst".

The driver example step:

1. Initialize uDMAC unit A by setting transfer type as burst type, enable channel, enable channel in mask setting, use primary data, use normal priority.
2. Set primary base address in the head of the reserved 1K RAM area.
3. Configurate the setting data for software trigger and fill it to 'control data' area.
4. Enable DMA unit A after all configuration is finished.
5. If DMAC_BASIC mode is selected, need to trigger it again and again until transfer is finished. If DMAC_AUTOMATIC mode is selected, only need to trigger it once before transfer is finished.

Judge if transfer is finished, then compare the data in destination with source

6-20 WDT

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

The driver example step:

1. Initialize WDT by setting detection time and selecting WDT interrupt as counter overflow operation.
2. There are two demos for WDT and DEMO2 is defined by macro definition.
   DEMO1:
   When timer is overflow, NMI interrupt is generated (LED1 blink once) and then WDT will be cleared.
   DEMO2:
   WDT clear code will be written to the watchdog timer control register, and watch dog timer counter will be cleared.(LED0 blink all the time)

***Note**:
LED0 connect with PB0 in TMPM46B MCU board.
LED1 connect with PB1 in TMPM46B MCU board.

7 Software

This software project creates the sample applications based on TMPM46B evaluation board which will demonstrate the main feature of TMPM46B MCU.

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

The workspace structure and project names are defined as following:

IAR EWARM:

```
├─WDT_NMI
        ├─APP
        │   main.c
        │   tmpm46b_wdt_int.c
        │   tmpm46b_wdt_int.h
        │
        ├─TX04_CMSIS
        │ │ system_TMPM46B.c
        │ │ system_TMPM46B.h
        │ │ TMPM46B.h
        │ │
        │ └─startup
        │       startup_TMPM46B.s
        │
        ├─TX04_Periph_Driver
        │   ├─inc
        │   │   tmpm46b_gpio.h
        │   │   tmpm46b_wdt.h
        │   │   TX04_common.h
        │   │
        │   └─src
        │         tmpm46b_gpio.c
        │         tmpm46b_wdt.c
        │
        └─TMPM46B-EVAL
                led.c
                led.h
                sw.c
                sw.h
```

KEIL MDK:

```
├─WDT_NMI
        ├─APP
        │   main.c
        │   tmpm46b_wdt_int.c
        │   tmpm46b_wdt_int.h
        │
```

```
├─TX04_CMSIS
│     system_TMPM46B.c
│     startup_TMPM46B.s
│
├─TX04_Periph_Driver
│     tmpm46b_gpio.c
│     tmpm46b_wdt.c
│
└─TMPM46B-EVAL
        led.c
        sw.c
```

7-1 ADC

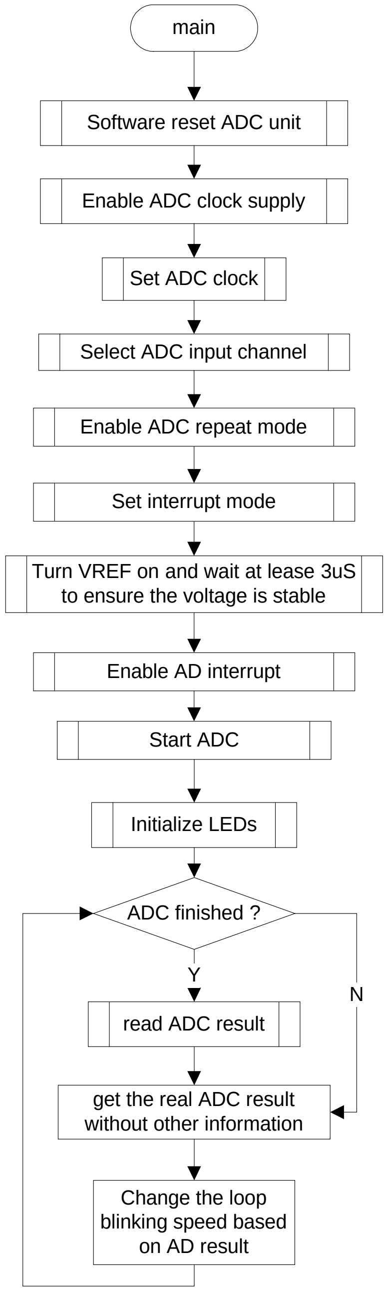
7-1-1 Example: ADC Data Read

This is an example based on the TX04 Peripheral Driver (ADC, CG, GPIO).

The example includes:

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

- **Flowchart:**



- **Code and Explanation for the Example**

At first, reset ADC unit , and enable the clock supply for it, set ADC clock, select channel and enable the repeat mode, set interrupt mode.

```
/* Software reset ADC */
ADC_SWReset(TSB_AD);

/* Enable ADC clock supply */
CG_SetADCClkSupply(ENABLE);

/* Set ADC clock */
ADC_SetClk(TSB_AD,   ADC_CONVERSION_CLK_80,
                ADC_FC_DIVIDE_LEVEL_8);
```

```
/* Select ADC input channel : Channel 0 */
ADC_SetInputChannel(TSB_AD, ADC_AN_00);

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

/* Set interrupt mode */
ADC_SetINTMode(TSB_AD, ADC_INT_CONVERSION_8);
```

Turn VREF on and wait at least 3us to ensure the voltage is stable:

```
ADC_SetVref(TSB_AD, ENABLE);
cnt = 100U;
while(cnt){
        cnt--;
}
```

Enable AD interrupt and start ADC:

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

/* Start ADC */
ADC_Start(TSB_AD);
```

After started AD conversion, wait interrupt flag for ADC conversion finished, then read ADC conversion result and use it to adjust the LEDs loop blinking speed.

```
while (1U) {

        if (fIntADC == 1U) {
                fIntADC = 0U;
                /* Read ADC result when it is finished */
                adResult = ADC_GetConvertResult(TSB_AD, ADC_REG00);

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

        } else {
                /* Do nothing */
        }

        /* use 'timeUp' above to adjust the loop blinking speed */
        if (softT.flag_TimeUp) {
                softT.flag_TimeUp = 0U;
                cnt++;
                if (cnt >= timeUp) {
                        cnt = 0U;
                        idx++;
                        if (idx == 1U) {
                             LED_On(LED0);
                        } else if (idx == 2U) {
                             LED_On(LED1);
                             LED_Off(LED0);
```

```
                } else if (idx == 3U) {
                    LED_On(LED2);
                    LED_Off(LED1);
                } else if (idx == 4U) {
                    LED_On(LED3);
                    LED_Off(LED2);
                } else if (idx == 5U) {
                    idx = 0U;
                    LED_Off(LED3);
                } else {
                    /* Do nothing */
                }
            } else {
                    /* Do nothing */
            }
        } else {
            /* Do nothing */
        }
    }
```

7-2 AES

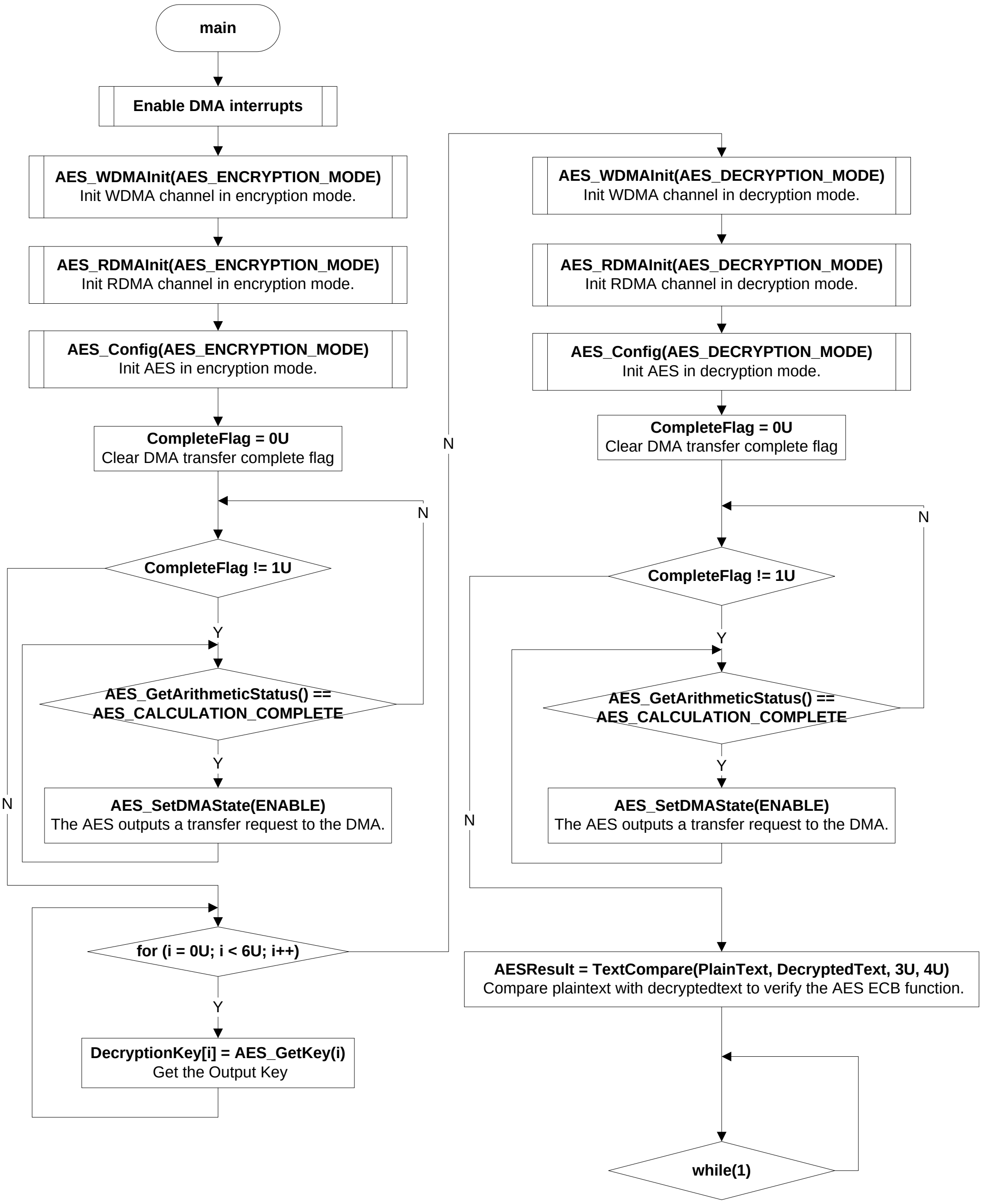
7-2-1 Example: AES ECB mode

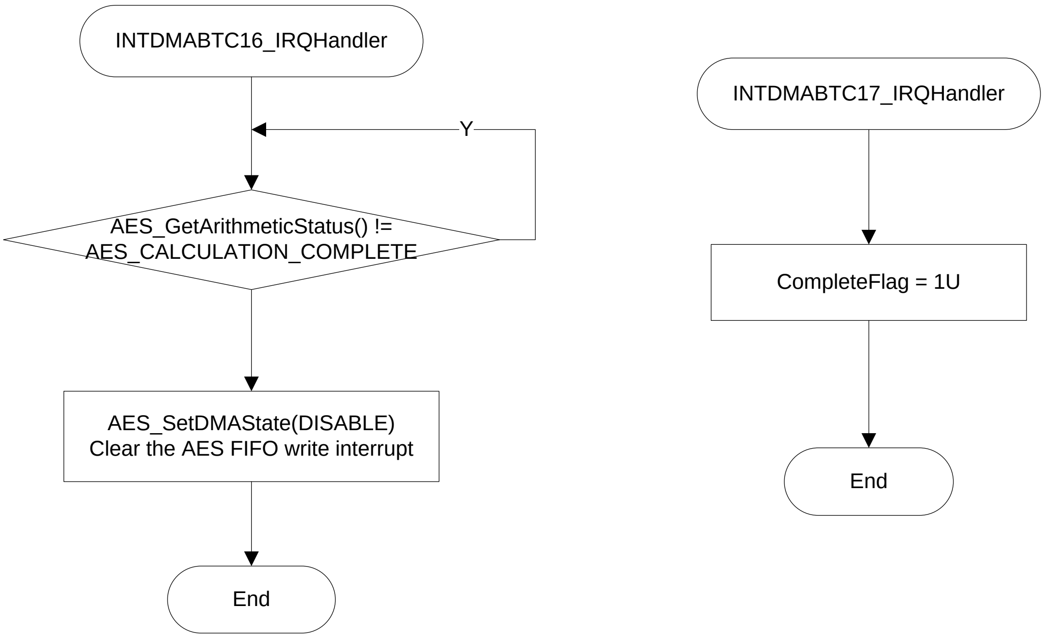
This is a simple example based on the TX04 Peripheral Driver (AES, uDMAC and UART).

The example includes:

1. Initialize uDMAC and AES in ECB encryption mode.
2. AES encrypts plaintext into encrypted text by DMA transfer.
3. Initialize uDMAC and AES in ECB decryption mode.
4. AES decrypts encrypted text into decrypted text by DMA transfer.
5. Compare plaintext with decrypted text to verify the ECB procedure.

- **Flowchart**


main
Enable DMA interrupts
AES_WDMAInit(AES_ENCRYPTION_MODE)
Init WDMA channel in encryption mode.
AES_RDMAInit(AES_ENCRYPTION_MODE)
Init RDMA channel in encryption mode.
AES_Config(AES_ENCRYPTION_MODE)
Init AES in encryption mode.
CompleteFlag = 0U
Clear DMA transfer complete flag
CompleteFlag != 1U
AES_GetArithmeticStatus() ==
AES_CALCULATION_COMPLETE
AES_SetDMAState(ENABLE)
The AES outputs a transfer request to the DMA.
for (i = 0U; i < 6U; i++)
DecryptionKey[i] = AES_GetKey(i)
Get the Output Key
AES_WDMAInit(AES_DECRYPTION_MODE)
Init WDMA channel in decryption mode.
AES_RDMAInit(AES_DECRYPTION_MODE)
Init RDMA channel in decryption mode.
AES_Config(AES_DECRYPTION_MODE)
Init AES in decryption mode.
CompleteFlag = 0U
Clear DMA transfer complete flag
CompleteFlag != 1U
AES_GetArithmeticStatus() ==
AES_CALCULATION_COMPLETE
AES_SetDMAState(ENABLE)
The AES outputs a transfer request to the DMA.
AESResult = TextCompare(PlainText, DecryptedText, 3U, 4U)
Compare plaintext with decryptedtext to verify the AES ECB function.
while(1)
Y
N



- **Code and Explanation for the Example**

Set up DMA interrupts

```
/* enable DMA interrupts */
NVIC_ClearPendingIRQ(INTDMABTC16_IRQn);
NVIC_EnableIRQ(INTDMABTC16_IRQn);
NVIC_ClearPendingIRQ(INTDMABTC15_IRQn);
NVIC_EnableIRQ(INTDMABTC15_IRQn);
NVIC_ClearPendingIRQ(INTDMABTC17_IRQn);
NVIC_EnableIRQ(INTDMABTC17_IRQn);
__enable_irq();
```

Configure WDMA channel to write data to AES processor. Configure RDMA channel to read data from AES processor. Configure AES processor to ECB mode, encryption operation, 192-bit key length and set the key data.

```
/* Initialze WDMA, WDMA will be triggered by DMA transfer interrupt. */
AES_WDMAInit(AES_ENCRYPTION_MODE);
/* Initialzie RDMA, RDMA will be triggered by AES calculation complete. */
AES_RDMAInit(AES_ENCRYPTION_MODE);
/* Initialize AES to encryption mode */
AES_Config(AES_ENCRYPTION_MODE);
```

AES encrypts plaintext to encrypted text in units of 128-bit.

```
CompleteFlag = 0U;
/* Encrypt 3 units plaintext. */
while (CompleteFlag != 1U) {
    while(AES_GetArithmeticStatus() == AES_CALCULATION_COMPLETE) {
        AES_SetDMAState(ENABLE);
    }
}
```

Get the output key. And the output key will be used as the input key in decryption.

```
for (i = 0U; i < 6U; i++) {
```

```
        DecryptionKey[i] = AES_GetKey(i);
    }
```

Configure WDMA channel to write data to AES processor. Configure RDMA channel to read data from AES processor. Configure AES processor to ECB mode, decryption operation, 192-bit key length and set the key data.

```
    /* Initialze WDMA, WDMA will be triggered by DMA transfer interrupt. */
    AES_WDMAInit(AES_DECRYPTION_MODE);
    /* Initialzie RDMA, RDMA will be triggered by AES calculation complete. */
    AES_RDMAInit(AES_DECRYPTION_MODE);
    /* Initialize AES to encryption mode */
    AES_Config(AES_DECRYPTION_MODE);
```

AES decrypts encrypted text to decrypted text in units of 128-bit.

```
    CompleteFlag = 0U;
    /* Decrypt the 3 units encryted text. */
    while (CompleteFlag != 1U) {
        while(AES_GetArithmeticStatus() == AES_CALCULATION_COMPLETE )
{
            AES_SetDMAState(ENABLE);
        }
    }
```

Compare plaintext with decryptedtext to verify the AES ECB mode. If AESResult = SUCCESS, it means that the AES ECB mode procedure has passed verification.

```
    /* Compare plaintext with decrypted text to verify the AES ECB function. */
    AESResult = TextCompare(PlainText, DecryptedText, 3U, 4U);
    if (AESResult == SUCCESS) {
        common_uart_disp("AES ECB mode processed successfully !\n");
    } else {
        common_uart_disp("AES ECB mode processed with error !\n");
    }
```

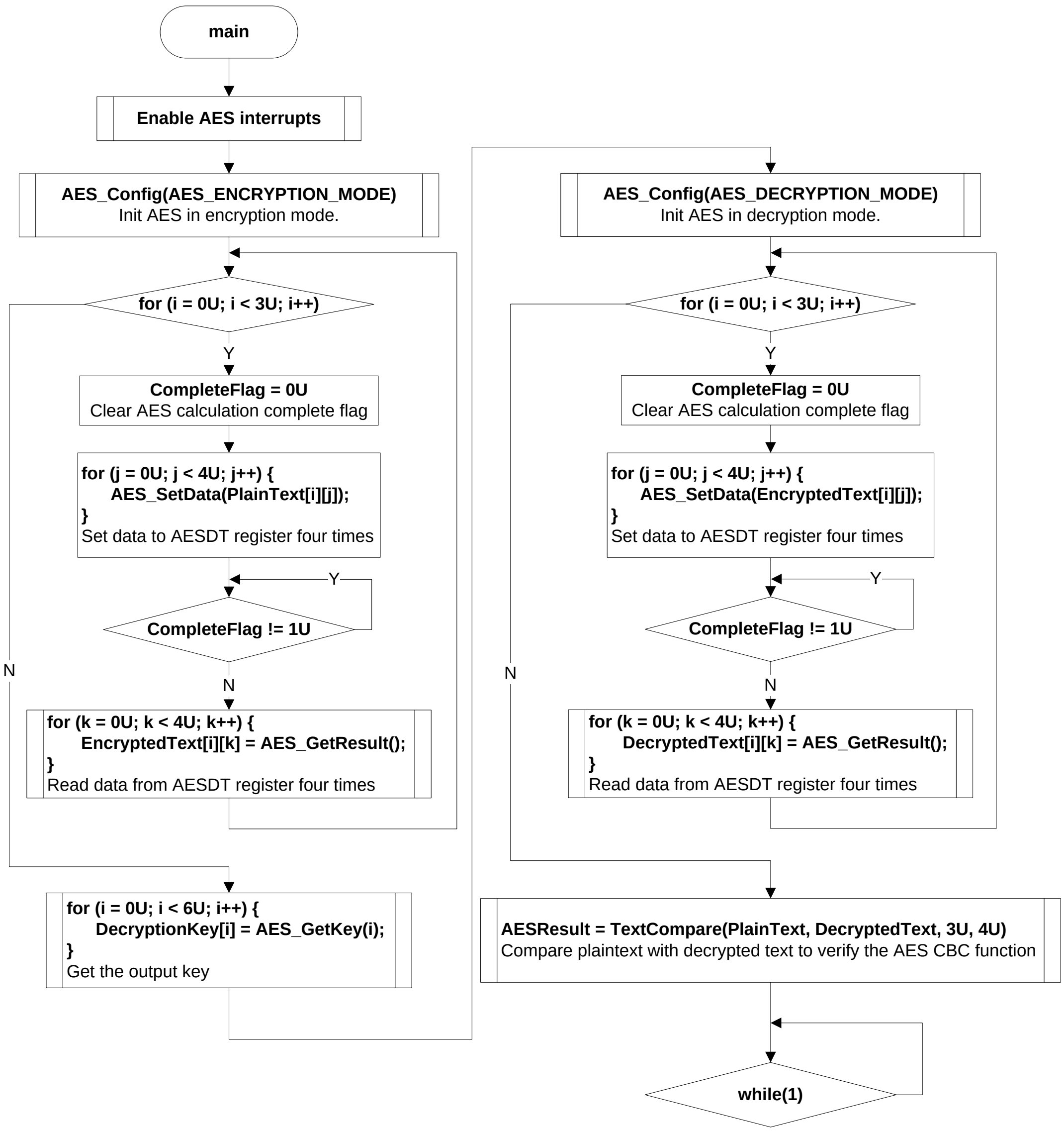
7-2-2 Example: AES CBC mode

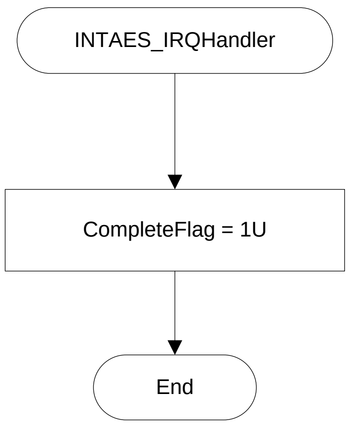
This is a simple example based on the TX04 Peripheral Driver (AES and UART).

The example includes:

1. Initialize AES in CBC encryption mode.
2. AES encrypts plaintext into encrypted text by CPU transfer.
3. Initialize AES in CBC decryption mode.
4. AES decrypts encrypted text into decrypted text by CPU transfer.
5. Compare plaintext with decrypted text to verify the CBC procedure.

- **Flowchart**





- **Code and Explanation for the Example**

Set up AES interrupts

```
/* enable AES calculation completion interrupt */
NVIC_ClearPendingIRQ(INTAES_IRQn);
```

```
        NVIC_EnableIRQ(INTAES_IRQn);
        __enable_irq();
```

Configure AES processor to CBC mode, encryption operation and 192-bit key length; set the key data and input initial vector.

```
        /* Initialize AES to encryption mode */
        AES_Config(AES_ENCRYPTION_MODE);
```

AES encrypts plaintext to encrypted text in units of 128-bit.

```
    /* Encrypt 3 units plaintext. */
    for (i = 0U; i < 3U; i++) {
          CompleteFlag = 0U;
        /* If 128-bit data is stored in the AESDT register, calculation automatically
starts */
        for (j = 0U; j < 4U; j++) {
            AES_SetData(PlainText[i][j]);
        }
        /* Wait for calculation completion */
        while (CompleteFlag != 1U) {
            /* Do nothing */
        }
        /*The CPU reads the value from AESODT<ODT[31:0]>(four times) */
        for (k = 0U; k < 4U; k++) {
            EncryptedText[i][k] = AES_GetResult();
        }
    }
```

Get the output key. And the output key will be used as the input key in decryption.

```
    for (i = 0U; i < 6U; i++) {
        DecryptionKey[i] = AES_GetKey(i);
    }
```

Configure AES processor to CBC mode, decryption operation and 192-bit key length; set the key data and input initial vector.

```
    /* Initialize AES to encryption mode */
    AES_Config(AES_DECRYPTION_MODE);
```

AES decrypts encrypted text to decrypted text in units of 128-bit.

```
    /* Decrypt the 3 units encryted text. */
    for (i = 0U; i < 3U; i++) {
        CompleteFlag = 0U;
        /* If 128-bit data is stored in the AESDT register, calculation automatically
starts */
        for (j = 0U; j < 4U; j++) {
            AES_SetData(EncryptedText[i][j]);
        }
        /* Wait for calculation completion */
        while (CompleteFlag != 1U) {
            /* Do nothing */
        }
        /*The CPU reads the value from AESODT<ODT[31:0]>(four times) */
        for (k = 0U; k < 4U; k++) {
            DecryptedText[i][k] = AES_GetResult();
```

```
        }
    }
```

Compare plaintext with decryptedtext to verify the AES CBC mode. If AESResult = SUCCESS, it means that the AES CBC mode procedure has passed verification.

```
    /* Compare plaintext with decrypted text to verify the AES CBC function. */
    AESResult = TextCompare(PlainText, DecryptedText, 3U, 4U);
    if (AESResult == SUCCESS) {
        common_uart_disp("AES CBC mode processed successfully !\n");
    } else {
        common_uart_disp("AES CBC mode processed with error !\n");
    }
```

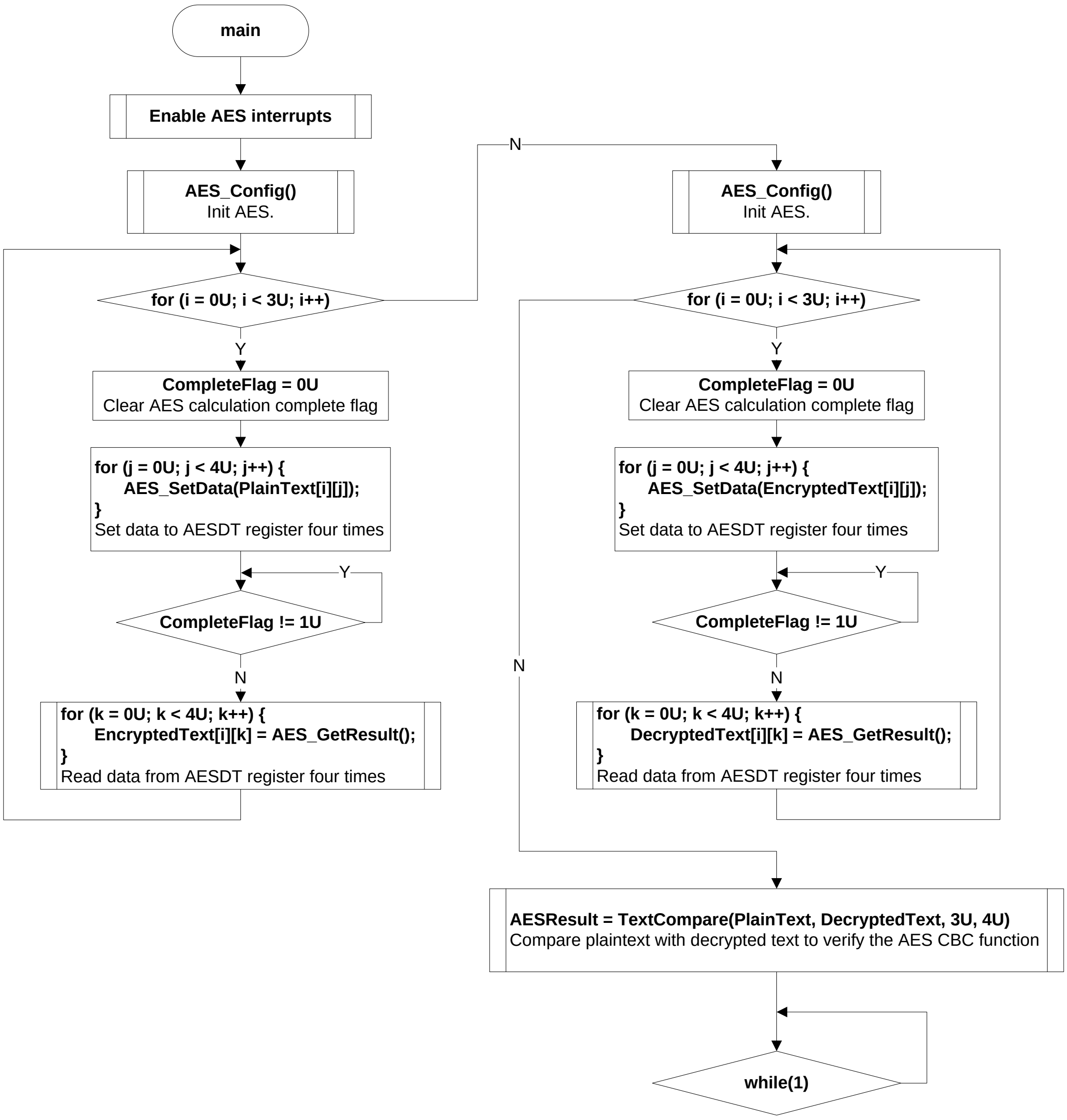
7-2-3 Example: AES CTR mode

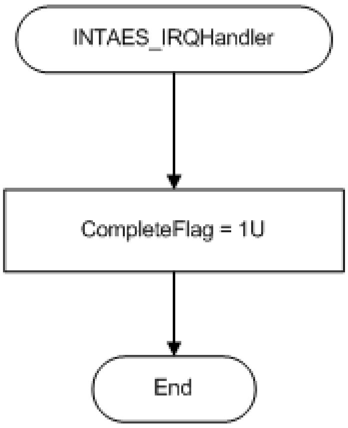
This is a simple example based on the TX04 Peripheral Driver (AES and UART).

The example includes:

1. Initialize AES in CTR mode.
2. AES encrypts plaintext into encrypted text by CPU transfer.
3. Initialize AES in CTR mode.
4. AES decrypts encrypted text into decrypted text by CPU transfer.
5. Compare plaintext with decrypted text to verify the CTR procedure.

- **Flowchart**





- **Code and Explanation for the Example**

Set up AES interrupts

```
/* enable AES calculation completion interrupt */
NVIC_ClearPendingIRQ(INTAES_IRQn);
```

```
NVIC_EnableIRQ(INTAES_IRQn);
__enable_irq();
```

Configure AES processor to CTR mode and 192-bit key length; set the key data; input counter value and clear AESMOD<OP> to “0”.

```
/* Initialize AES to encryption mode */
AES_Config();
```

AES encrypts plaintext to encrypted text in units of 128-bit.

```
/* Encrypt 3 units plaintext. */
for (i = 0U; i < 3U; i++) {
        CompleteFlag = 0U;
    /* If 128-bit data is stored in the AESDT register, calculation automatically
starts */
    for (j = 0U; j < 4U; j++) {
        AES_SetData(PlainText[i][j]);
    }
    /* Wait for calculation completion */
    while (CompleteFlag != 1U) {
        /* Do nothing */
    }
    /*The CPU reads the value from AESODT<ODT[31:0]>(four times) */
    for (k = 0U; k < 4U; k++) {
        EncryptedText[i][k] = AES_GetResult();
    }
}
```

Configure AES processor to CTR mode and 192-bit key length; set the key data; input counter value and clear AESMOD<OP> to “0”.

```
/* Initialize AES to encryption mode */
AES_Config();
```

AES decrypts encrypted text to decrypted text in units of 128-bit.

```
/* Decrypt the 3 units encryted text. */
for (i = 0U; i < 3U; i++) {
    CompleteFlag = 0U;
    /* If 128-bit data is stored in the AESDT register, calculation automatically
starts */
    for (j = 0U; j < 4U; j++) {
        AES_SetData(EncryptedText[i][j]);
    }
    /* Wait for calculation completion */
    while (CompleteFlag != 1U) {
        /* Do nothing */
    }
    /*The CPU reads the value from AESODT<ODT[31:0]>(four times) */
    for (k = 0U; k < 4U; k++) {
        DecryptedText[i][k] = AES_GetResult();
    }
}
```

Compare plaintext with decrypted text to verify the AES CTR mode. If AESResult = SUCCESS, it means that the AES CTR mode procedure has passed verification.

```
/* Compare plaintext with decryptedtext to verify the AES CTR function. */
AESResult = TextCompare(PlainText, DecryptedText, 3U, 4U);
if (AESResult == SUCCESS) {
      common_uart_disp("AES CTR mode processed successfully !\n");
} else {
      common_uart_disp("AES CTR mode processed with error !\n");
}
```

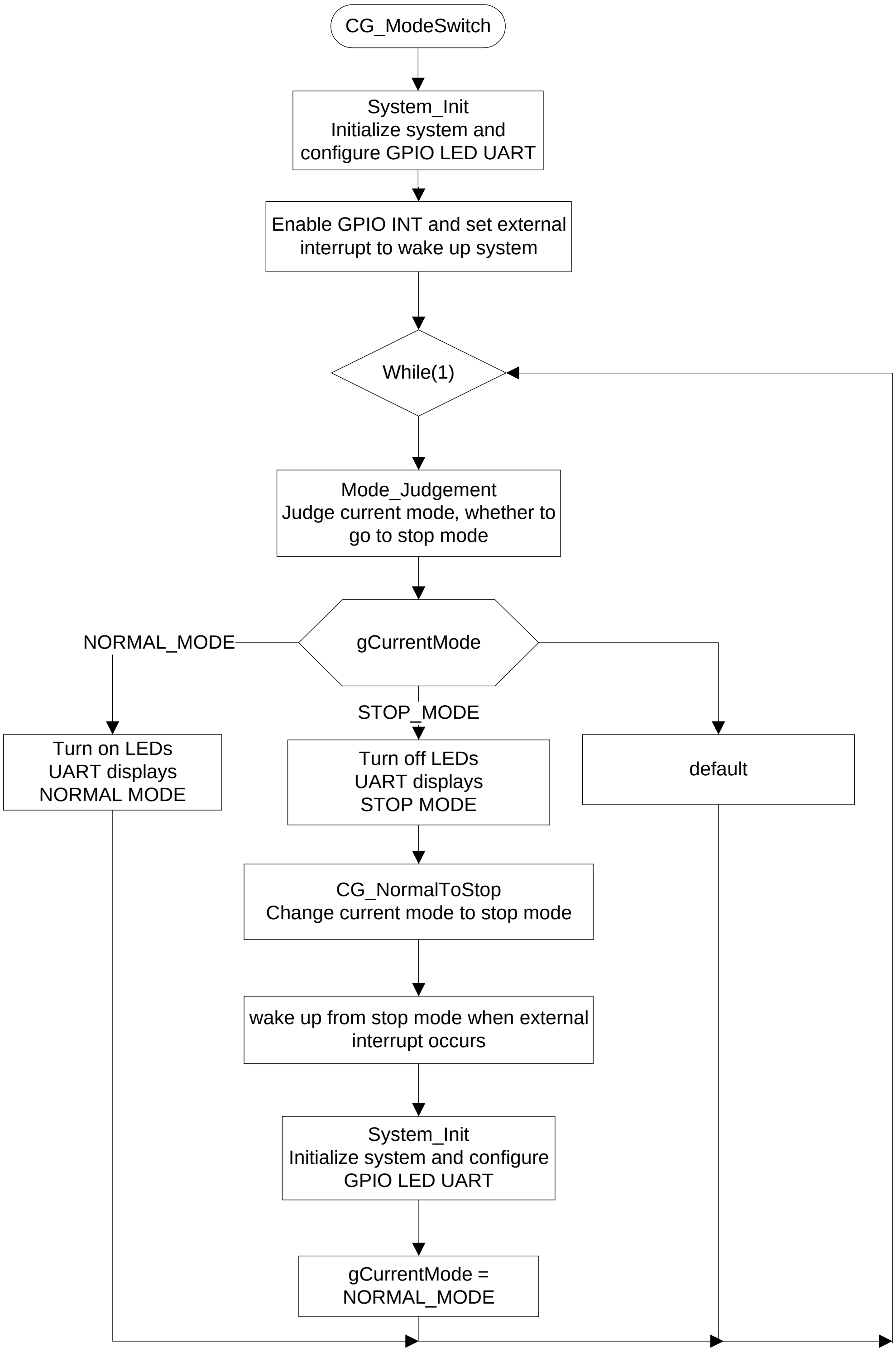
7-3 CG

7-3-1 Example: Mode switch STOP1

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

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

- **Flowchart**



- **Code and Explanation for the Example**

The following simple example is based on TX04 Peripheral Driver (CG), which will switch between normal mode and stop mode.

Normal setup for CG (after reset)

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

Example code is as the following:

```
if (CG_GetFoscSrc()==CG_FOSC_OSC_INT){
    /* Switch over from IHOSC to EHOSC*/
    switchFromIHOSCtoEHOSC();
}
/* 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);
/* Set ΦT0 = fc/4 */
CG_SetPhiT0Level(CG_DIVIDE_4);
/* Set low power consumption mode stop1 */
CG_SetSTBYMode(CG_STBY_MODE_STOP1);
```

Configure external interrupt to wake up system

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

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

Setup to enter STOP mode

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

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

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.

```
WorkState st = BUSY;
CG_SetPLL(DISABLE);
CG_SetFPLLValue(CG_16M_MUL_5_FPLL);
retval = CG_SetPLL(ENABLE);
if (retval == SUCCESS) {
    /* Set warm up time */
    CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC_EXT_HIGH,
                           CG_WUODR_PLL);
    CG_StartWarmUp();
    do {
        st = CG_GetWarmUpState();
    } while (st != DONE);

    retval = CG_SetFcSrc(CG_FC_SRC_FPLL);
```

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

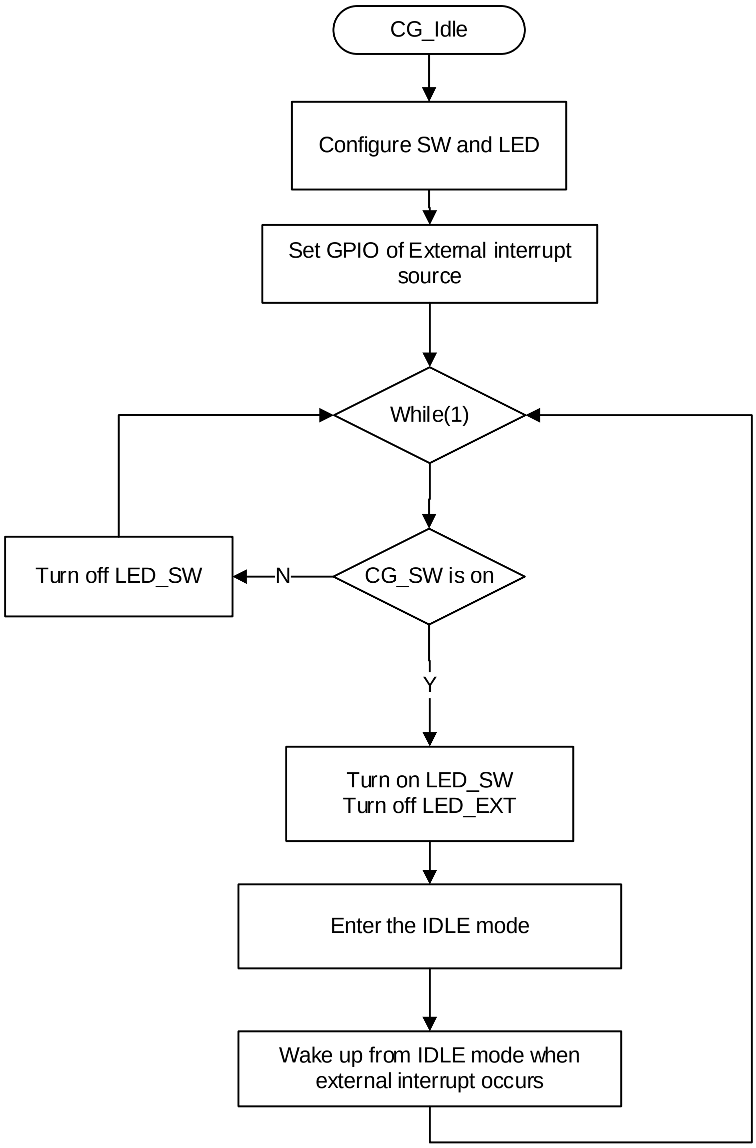
7-3-2 Example: Mode switch IDLE

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

The example includes:

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

- **Flowchart**



- **Code and Explanation for the Example**

The following simple example is based on TX04 Peripheral Driver (CG), which will switch between normal mode and IDLE mode.

Configure external interrupt to wake up system
Configure IO port for external interrupt INTF. Clear interrupt pending request, then enable INTF.

```
    GPIO_ExtIntSrc();
    CG_ClearINTReq(CG_ExtINTSrc);
    NVIC_ClearPendingIRQ(ExtINTSrc_IRQn);
    NVIC_EnableIRQ(ExtINTSrc_IRQn);
```

Setup to enter IDLE mode
Prepare for entering IDLE mode. Setup the INT source (INTF) to wake up system, use __WFI() instruction to enter idle mode.

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

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

    __DSB();
    __WFI();

    /* INT release */
    CG_SetSTBYReleaseINTSrc(CG_ExtINTSrc,
CG_INT_ACTIVE_STATE_RISING, DISABLE);
```

Control mode switch between normal mode and idle mode
Read a status of LED_EXT in while() loop after turn on LED_EXT.
When the status of CG_SW is OFF, turn off LED_SW. When the status of CG_SW is ON, enter IDLE mode after turn on LED_SW.

```
    LED_On(LED_EXT);
    while (1U) {

        if (SW_Get(CG_SW) == 0U) {
            LED_On(LED_SW);

            /* LED indicator is off before enter IDLE */
            LED_Off(LED_EXT);

            enter_IDLE();

        } else {
            LED_Off(LED_SW);
        }
    }
```

7-3-3 Example: Mode witch STOP2

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

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**

The following simple example is based on TX04 Peripheral Driver (CG), which will switch between normal mode and stop mode.

Configure the I/O pins for Key, LED:

```
    SW_Init();
    LED_Init();

    GPIO_ExtIntSrc();
```

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

```
    /* after release standby mode */
    if (is_StanbyRelease()) {
        LED_On(LED2);
        CG_SetSTBYReleaseINTSrc(CG_ExtINTSrc,
                CG_INT_ACTIVE_STATE_RISING, DISABLE);

        NVIC_EnableIRQ(ExtINTSrc_IRQn);

        CG_SetPortKeepInStop2Mode(DISABLE);
```

When a reset factor is not STOP2 mode release, enable interrupt to release standby mode.

```
} else {
        CG_ClearINTReq(CG_ExtINTSrc);
        NVIC_ClearPendingIRQ(ExtINTSrc_IRQn);
        NVIC_EnableIRQ(ExtINTSrc_IRQn);
    }
```

Read a status of LED_EXT in while() loop after turn on LED_EXT.
When the status of CG_SW is OFF, turn on LED_SW. When the status of CG_SW is ON, enter STO2 mode after turn off LED_EXT.

```
    LED_On(LED_EXT);
    while (1U) {
        if (SW_Get(CG_SW) == 1) {

            LED_Off(LED_ALL);     /* LED   is off before enter stop2 */
            enter_STOP2();

        } else {
            LED_On(LED_SW);
        }
    }
}
```

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

```
void config_STOP2(void)
{
    volatile WorkState st = BUSY;
```

```
    volatile uint32_t wuef = 0U;

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

    TSB_CG->PLLSEL &= PLLON_CLEAR;
    TSB_CG->PLLSEL &= PLLSEL_CLEAR;
    while (CG_GetFcSrc() != CG_FC_SRC_FOSC) {
    };                                  /* Confirm */

    /* When IHOSC is disable, enable IHOSC */
    if (CG_GetFoscState(CG_FOSC_OSC_INT) == DISABLE) {
        /* Enable IHOSC */
        CG_SetFosc(CG_FOSC_OSC_INT, ENABLE);

        /* Wait until IHOSC become stable */
        CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC_EXT_HIGH,
OSCCR_WUPT_EXT);
        CG_StartWarmUp();
        wuef = TSB_CG->OSCCR & 0x00008000U;
        while (wuef) {              /* Warm-up */
            wuef = TSB_CG->OSCCR & 0x00008000U;
        }

        /* Set IHOSC as fosc */
        CG_SetFoscSrc(CG_FOSC_OSC_INT);

        /* Wait until fosc become IHOSC */
        while (CG_GetFoscSrc() != CG_FOSC_OSC_INT) {
        };
    }

    CG_SetPortKeepInStop2Mode(ENABLE);
}
```

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

```
    __DSB();
    __WFI();
```

7-4 ESG

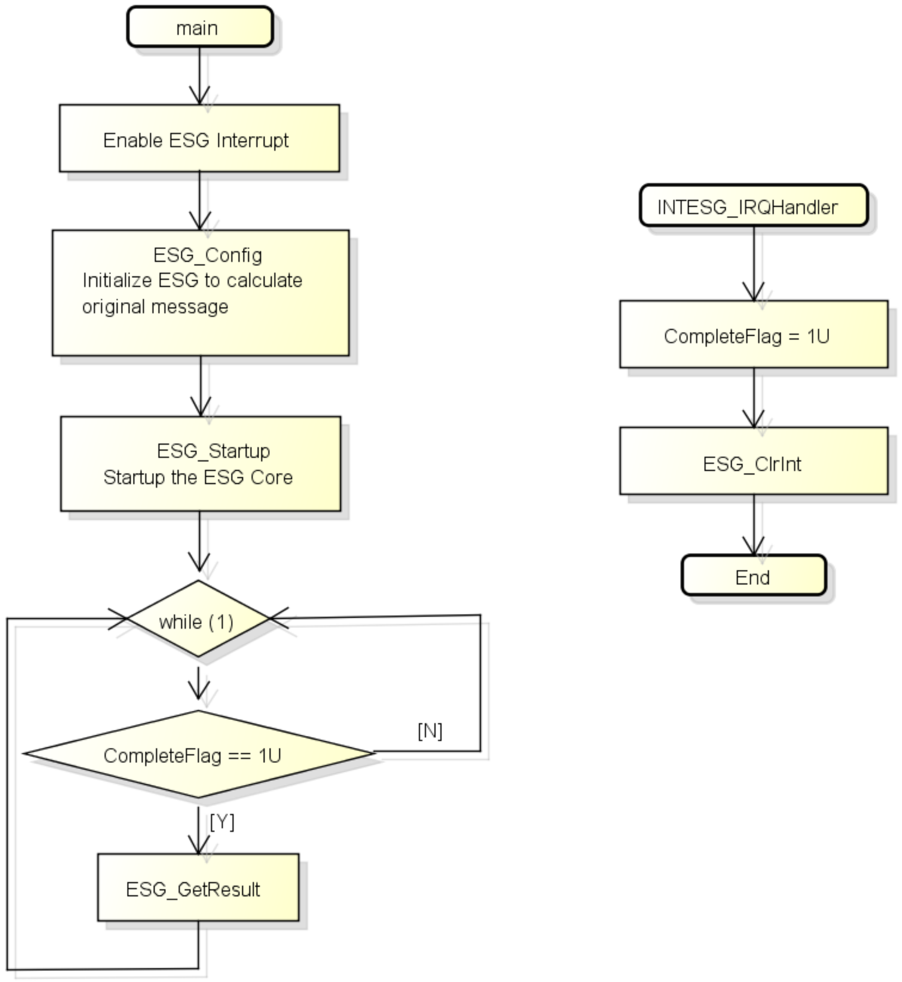
7-4-1 Example: ESG

This is a simple example based on the TX04 Peripheral Driver (ESG and UART).

The example includes:

1. Initialize ESG to calculate original message.
2. Start calculation and get the calculation result.

- **Flowchart**



- **Code and Explanation for the Example**

Set up ESG interrupt

```
/* enable ESG calculation completion interrupt */
NVIC_ClearPendingIRQ(INTESG_IRQn);
NVIC_EnableIRQ(INTESG_IRQn);
__enable_irq();
```

Configure the basic ESG configuration..

```
void ESG_Config(ESG_LatchTiming Value, uint16_t Fintming)
{
    /* Check the parameters */
    assert_param(IS_ESG_LATCH_TIMING(Value));

    /* Confirm the ESG core stops */
    if (ESG_GetCalculationStatus() == ESG_CALCULATION_COMPLETE) {
      /* Confirm no interrupt generation */
      if (ESG_GetIntStatus() == DISABLE) {
        /* Set the latch timing. */
        ESG_SetLatchTiming(Value);
        latchtiming = ESG_GetLatchTiming();
        if (Fintming >= 512U * (latchtiming + 1U) + 3U) {
        /* Set the output timing. */
        ESG_SetFintiming(Fintming);
```

```
            } else {
                common_uart_disp("Fintming value input error !\n");
            }
        } else {
            /* Do nothing */
        }
    } else {
        /* Do nothing */
    }
}
```

Initialize ESG to calculate original message, Start up the ESG core.

```
ESG_Config(ESG_LATCH_TIMING_1, 560);
ESG_Startup();
```

When the ESG core ends the operation, it outputs an interrupt. Then an entropy seed can be read..

```
if ((CompleteFlag == 1U)) {
        CompleteFlag = 0;
        /* Get the calculation result */
        ESG_GetResult(Data);
        for(i = 0U; i < 16U; i++) {
#ifdef DEBUG
            printf("0x%08x \r\n", Data[i]);
#endif
            sprintf((char *)buffer, "%lu", Data[i]);
            common_uart_disp(buffer);
            common_uart_disp("\n");
        }
}
```

7-5 EXB

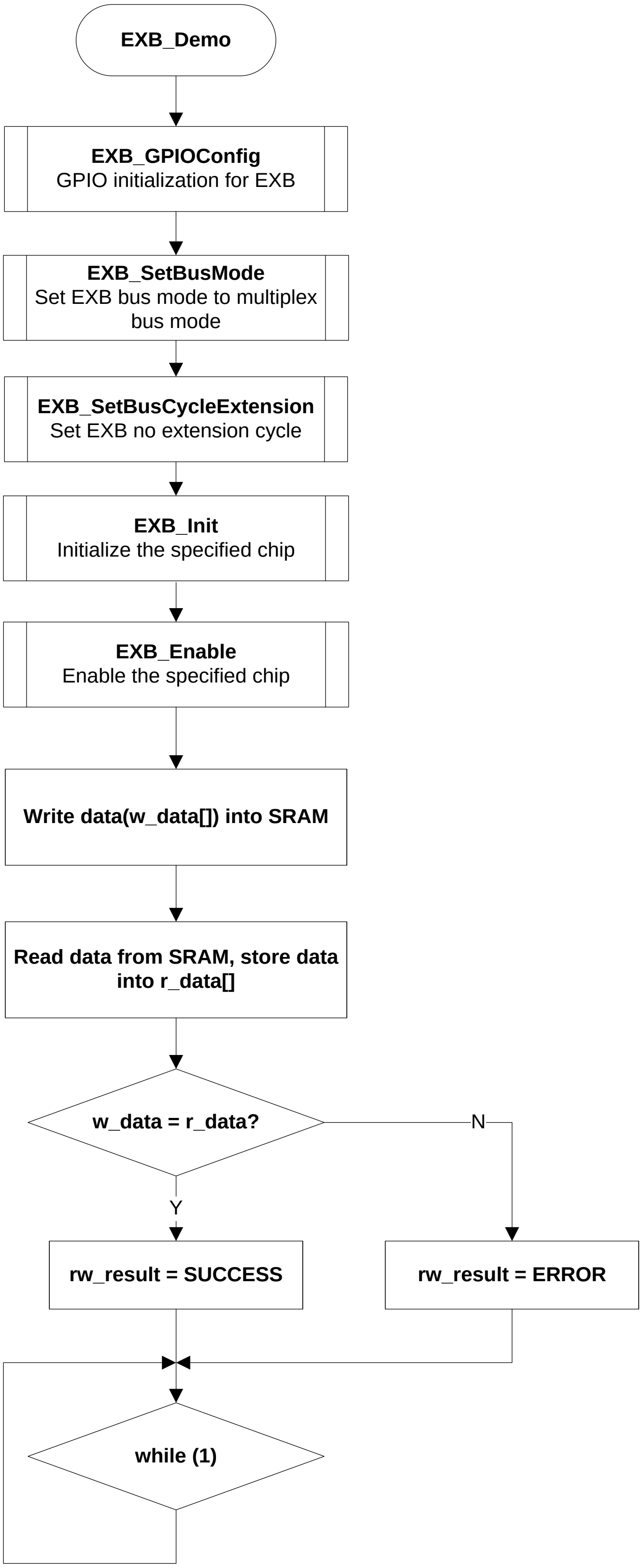
7-5-1 Example: SRAM Read/Write

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

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.

```
    uint8_t chip = EXB_CS1;
    uint8_t BusMode = EXB_BUS_MULTIPLEX;
    uint8_t Cycle = EXB_CYCLE_QUADRUPLE
    rw_result = SUCCESS;

#ifdef SRAM_RW
    uint32_t w_data[TEST_DATA_LEN] = { 0U };
    uint32_t r_data[TEST_DATA_LEN] = { 0U };
    uint16_t rw_cnt = 0U;
    uint32_t *addr = NULL;
    uint16_t i = 0U;
#endif

    EXB_InitTypeDef InitStruct = { 0U };

    /* Configure UART */
    hardware_init(UART_RETARGET);

    InitStruct.AddrSpaceSize = EXB_128K_BYTE;
    InitStruct.StartAddr = 0x00U;
    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_8;
    InitStruct.Cycles.ReadSetupCycle = EXB_CYCLE_2;
    InitStruct.Cycles.WriteSetupCycle = EXB_CYCLE_2;
    InitStruct.Cycles.ALEWaitCycle = EXB_CYCLE_2;
    InitStruct.Cycles.ReadRecoveryCycle = EXB_CYCLE_2;
    InitStruct.Cycles.WriteRecoveryCycle = EXB_CYCLE_2;
    InitStruct.Cycles.ChipSelectRecoveryCycle = EXB_CYCLE_2;
```

Configure GPIO for EXB and EXB, then enable the specified chip. Create data in w_data[] , then write this 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;
    }

    /* write data from w_data[] to SRAM */
    for (i = 0; i < TEST_DATA_LEN ; i++) {
      addr[i] = w_data[i];
```

```
    }

    /* read data from SRAM, store into r_data[] */
    for (i = 0; i < TEST_DATA_LEN ; i++) {
      r_data[i] = addr[i];
    }

    /* check rw_result to see if SRAM write/read is successful or not */
    for (i = 0; i < TEST_DATA_LEN ; i++) {
        if (w_data[i] != r_data[i]) {
            rw_result = ERROR;
            break;
        }
    }

    if (rw_result == SUCCESS) {
        common_uart_disp("SRAM read/write successful \n");
    } else {
        common_uart_disp("SRAM read/write failed \n");
    }

#endif
    while (1) {
      /* Do nothing */
    }
```

7-6 FLASH

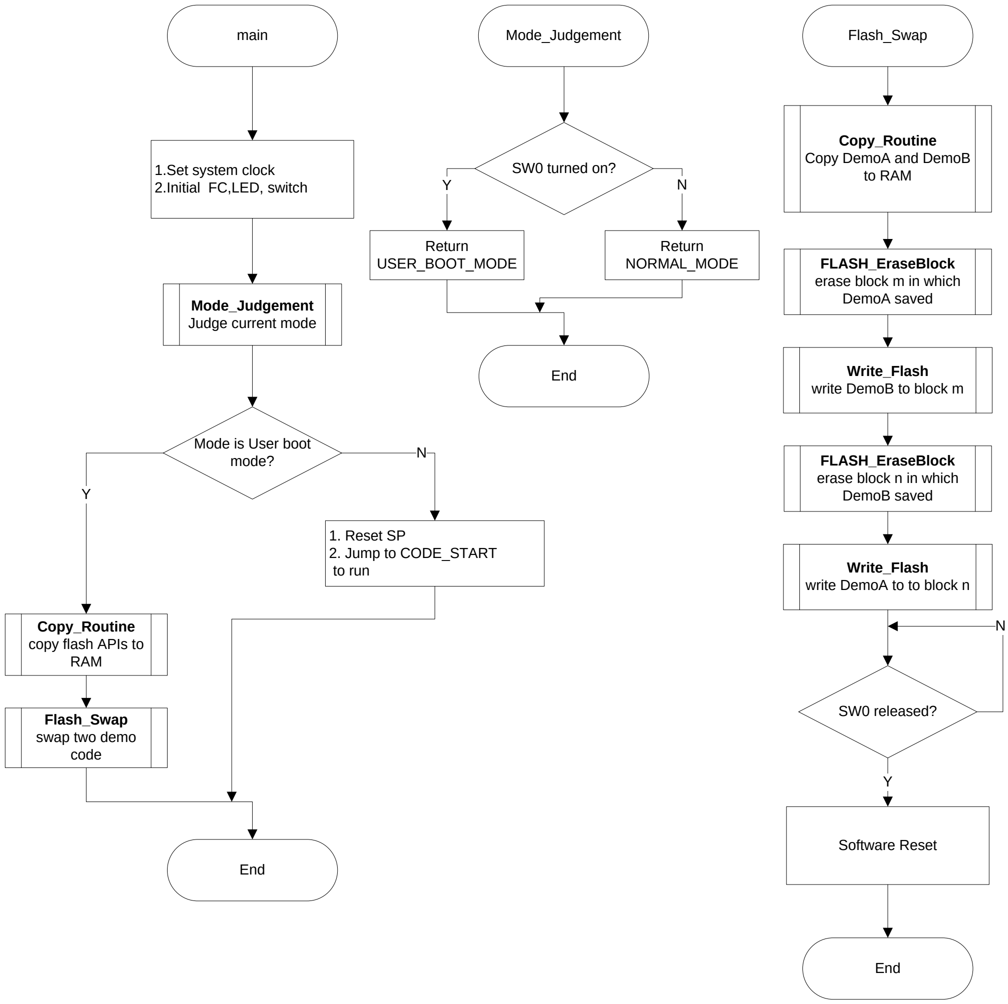
7-6-1 Example: Flash_UserBoot

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

The example includes:

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

- **Flowchart**



- **Code and Explanation for the Example**

At first initialize LED and switch. SW4 (GPIO) is used to judge current mode when reset .

```
    FC_init();
    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(SW4) == 1U) ? USER_BOOT_MODE : NORMAL_MODE;
}
```

If the SW4 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,LED2 Always show).

```
#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 SW4 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 */
      FC_SelectArea(FC_AREA_ALL);
      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;
    }

      FC_SelectArea(FC_AREA_NONE);
```

After SW4 released, it will execute software reset by using NVIC_SystemReset(). Then this demo will run again to enter normal mode, because Demo A and B have been exchanged, the address “CODE_START” has become the start address of Demo B, Demo B will run (LED1 blinks,LED3 Always show).

```

    while (SW_Get(SW4) == 1U) {
    }
    /* software reset */
    NVIC_SystemReset();
```

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

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

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

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

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

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

Function Write_Flash() will call FLASH_WritePage() to write data automatically in 16 bytes. The process of this function is basically same as FLASH_EraseBlock() except the automatic page program command.

```
        *addr1 = (uint32_t) 0x000000AA; /* bus cycle 1 */
        *addr2 = (uint32_t) 0x00000055; /* bus cycle 2 */
        *addr1 = (uint32_t) 0x000000A0; /* bus cycle 3 */
        for (i = 0U; i < PROGRAM_UNIT; i++) {
            *PA = *source;
            source++;
        }
```

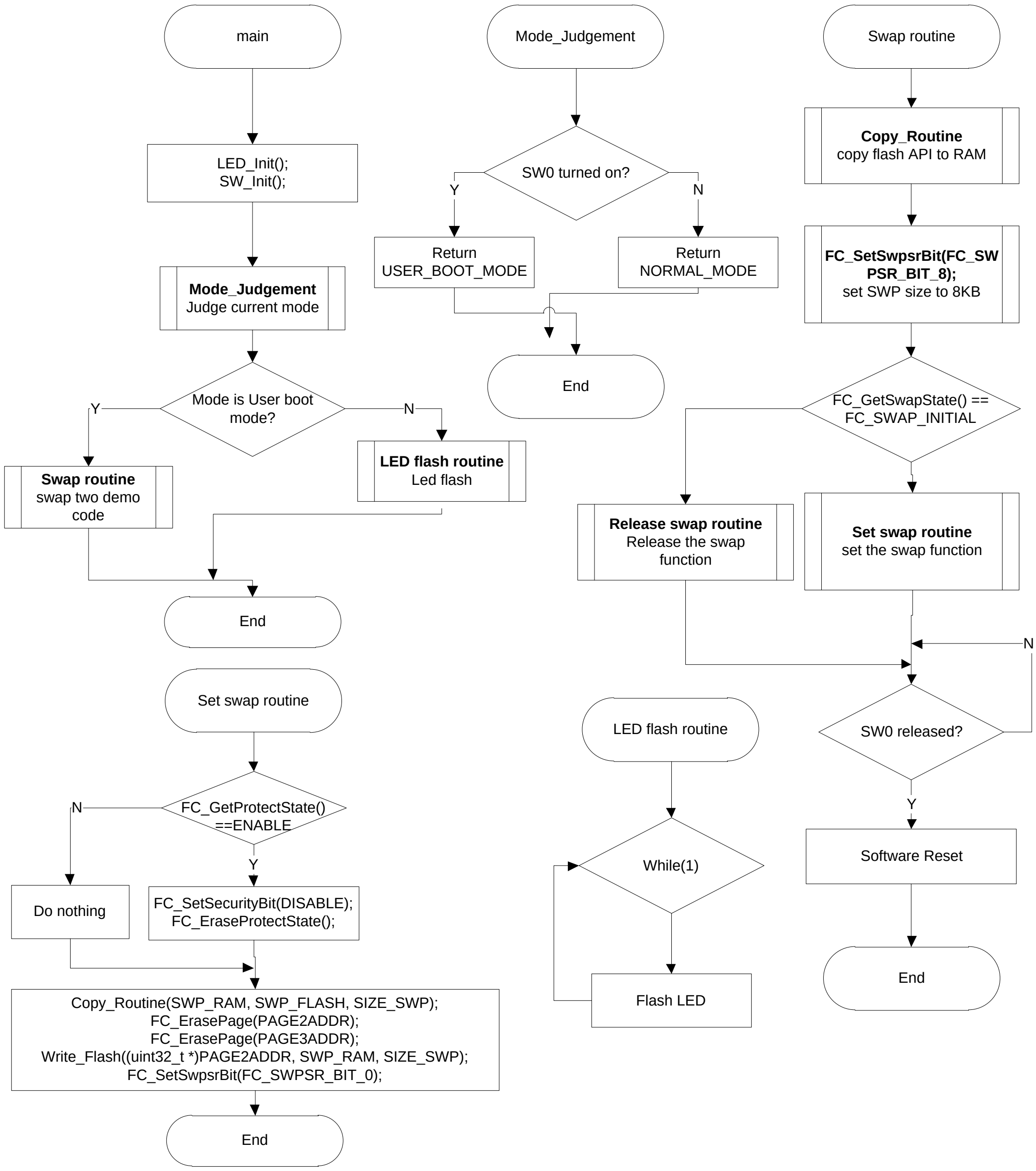
7-6-2 Example: Flash_Swap

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

The example includes:

1. On-board programming method of Flash Memory (Read/Write page, Swap)

- **Flowchart**



- **Code and Explanation for the Example**

At first initialize LED and SW.

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

Then judge the mode.

```
        if (Mode_Judgement() == SWP_BOOT_MODE) {     /* if SW4 is turned on, 
enter SWP boot mode */                    }
```

If the mode is SWP_BOOT_MODE, it will copy flash API to RAM and set SWP size to 8KB.

```
        Copy_Routine(FLASH_API_RAM, FLASH_API_ROM, SIZE_FLASH_API);
/* copy flash API to RAM */
        FC_SetSwpsrBit(FC_SWPSR_BIT_8);                 /* set SWP size to 
8KB*/                }
```

And then judge the swap state

```
        if (FC_GetSwapState() == FC_SWAP_INITIAL) {
```

If the sate of swap is FC_SWAP_INITIAL, it will ensure the state of FC protect is disable. And then set the swap function.

```
        if(FC_GetProtectState() == ENABLE){
                FC_EraseProtectState();
            } else {
                /* Do nothing */
            }
            Copy_Routine(SWP_RAM, SWP_FLASH, SIZE_SWP);    /* copy 
program to RAM */
            FC_ErasePage(PAGE2ADDR);
            FC_ErasePage(PAGE3ADDR);
            Write_Flash((uint32_t *)PAGE2ADDR, SWP_RAM, SIZE_SWP);
/* write program to PAGE2ADDR */
            FC_SetSwpsrBit(FC_SWPSR_BIT_0);      /* enable swp function */
```

If the swap state is not FC_SWAP_INITIAL, it will release the swap function.

```
        } else {
            FC_EraseProtectState();                 /* release swp function */
         }
```

LED2 will light after set or release swap function. And then wait for Key SW4 to release, software will reset at last.

```
                delay(4000000U);
                LED_On(LED2);
                /* wait for Key SW4 to release */
                while(Mode_Judgement() == SWP_BOOT_MODE){
                    /* Do nothing */
                }
                /* software reset */
                NVIC_SystemReset();
```

If the mode is NORMAL_MODE, it will flash the specified LED. (The specified LED is LED0 in program A, and LED1 in program B)

```
        while(1){
            LED_On(LED0);
            delay(4000000U);
            LED_Off(LED0);
            delay(4000000U);
        }
```

7-7 FUART

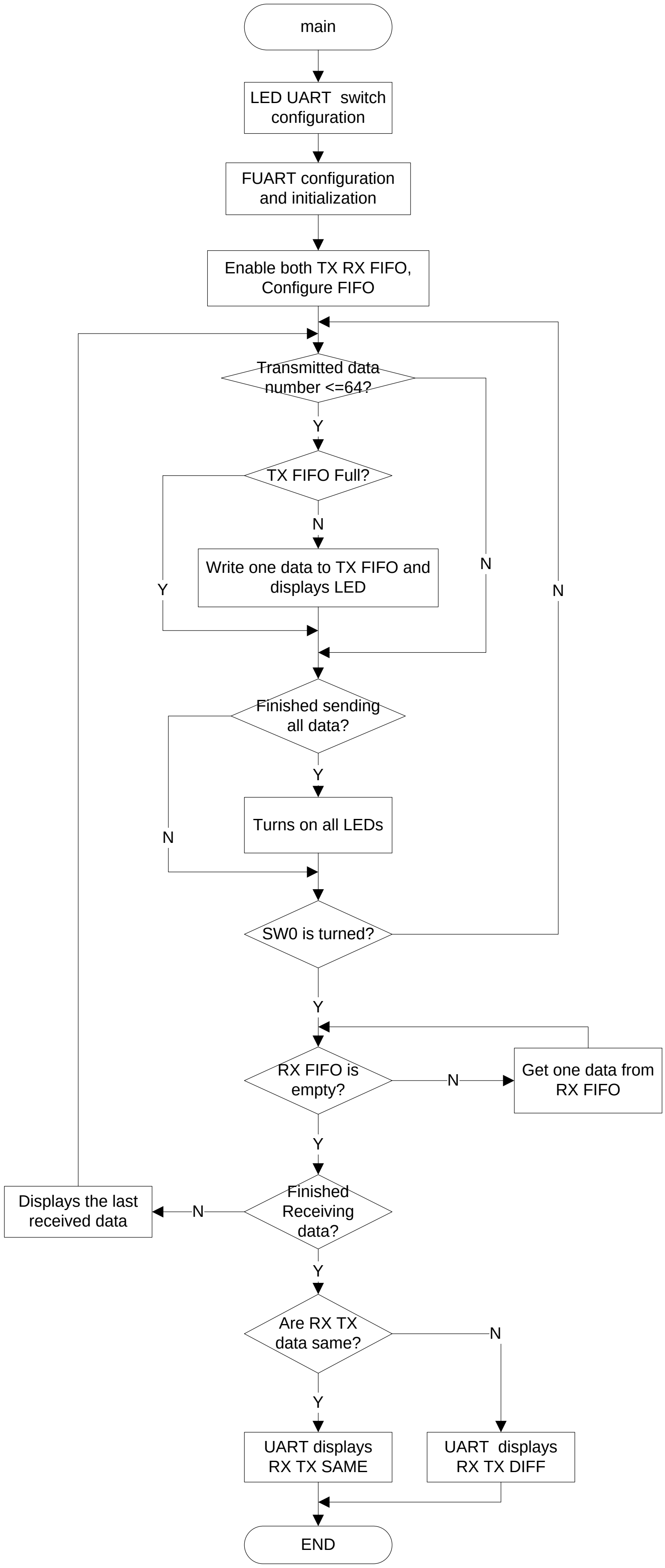
7-7-1 Example: LoopBack

This is a simple example based on the TX04 Peripheral Driver (FUART, GPIO).
This program can be run for two times to see the hardware flow control function.
The first time: Enable UT0RTS and UT0CTS hardware flow control
The second time: Disable UT0RTS and UT0CTS hardware flow control

The example includes:

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

- **Flowchart**


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

- **Code and Explanation for the Example**

Before you run the program, you need to decide whether to use the UT0RTS and UT0CTS flow control. If RUN_NONE_FLOW_CONTROL is undefined, UT0RTS and UT0CTS 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 switch and UART.
Use GPIO peripheral drivers configure GPIO for LED switch and UART.

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

Use GPIO peripheral drivers configure GPIO for FUART0.

```
    /* Configure port PF1 to be UT0TXD */
    GPIO_SetOutput(GPIO_PF, GPIO_BIT_1);
    GPIO_EnableFuncReg(GPIO_PF, GPIO_FUNC_REG_3, GPIO_BIT_1);

    /* Configure port PF2 to be UT0RXD */
    GPIO_SetInput(GPIO_PF, GPIO_BIT_2);
    GPIO_EnableFuncReg(GPIO_PF, GPIO_FUNC_REG_3, GPIO_BIT_2);

    /* Configure port PF0 to be UT0CTS */
    GPIO_SetInput(GPIO_PF, GPIO_BIT_0);
    GPIO_EnableFuncReg(GPIO_PF, GPIO_FUNC_REG_3, GPIO_BIT_0);

    /* Configure port PF3 to be UT0RTS */
    GPIO_SetOutput(GPIO_PF, GPIO_BIT_3);
    GPIO_EnableFuncReg(GPIO_PF, GPIO_FUNC_REG_3, 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(64U==cntTx){
                    LED_On(LED_ALL);        /* sending data is finished */
                }
        }
}
```

Each time when Key SW0 is turned, the program starts to read data from Receive FIFO. If some data are got, the program doesn’t stop reading data until the FIFO is empty, and UART will display the last received data this time. If no any data can be got when SW0 is turned, UART displays “RX FINISH” the program starts to compare the received data with transmitted data. If received data are same with transmitted data, UART displays “RX TX SAME”. If received data are different with transmitted data, UART displays “RX TX DIFF”.

```
SW0_this = SW_Get(SW0);
rxlast = rxthis;
rxthis = cntRx;

if (rxlast != rxthis) {        /* there are some data that has been received    */
        common_uart_disp("LAST RX DATA:");
        rxnum[0] = ('0' + receive/10U);
        rxnum[1] = ('0' + receive%10U);
        common_uart_disp(rxnum);     /* dispaly the last received data */
} else {                    /* receiving data is finished */
        common_uart_disp("RX FINISH");
        result = Buffercompare(Tx_Buf, Rx_Buf, MAX_BUFSIZE);
        if (result == SAME) {
                /* received data are same with trnsmitted data */
                /* UT0RTS and UT0CTS flow control has worked normally */
                        common_uart_disp("RX TX SAME");
                        while(1){}
        } else {
                /* received data are different with trnsmitted data */
                /* UT0RTS and UT0CTS flow control doesn't work */
                        common_uart_disp("RX TX DIFF");
                        while(1){}
        }
}
```

***Note:**

LED0~LED3 connect with PB0~PB3 in TMPM46B MCU board.

SW0 connect with PJ0 in TMPM46B MCU board.

7-8 GPIO

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

7-8-1 Example: GPIO Data Read

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

- **Code and Explanation for the Example**

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

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

    GPIO_SetInput(GPIO_PJ,   GPIO_BIT_0 | GPIO_BIT_1 | GPIO_BIT_2 |
GPIO_BIT_3);
```

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

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

```
    if (GPIO_ReadDataBit(GPIO_PJ, GPIO_BIT_0) == 1U) {
                tmp = 1U;
    } else {
               /*Do nothing */
    }
```

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

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

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

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

7-9 I2C

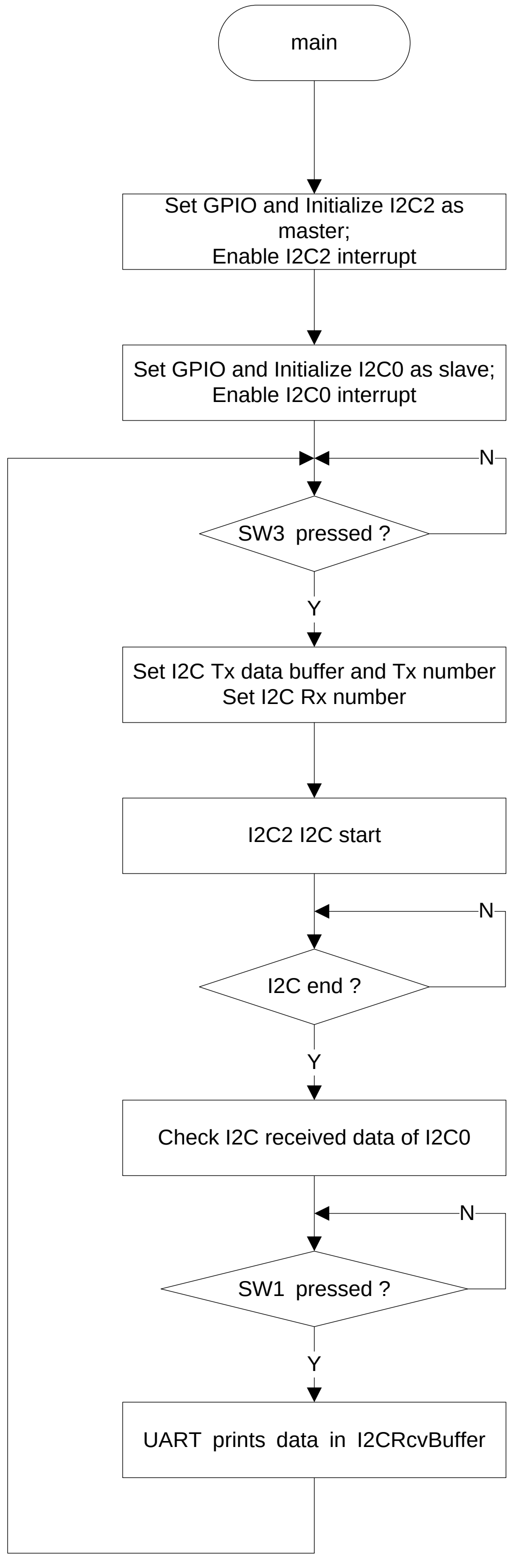
7-9-1 Example: I2C Slave

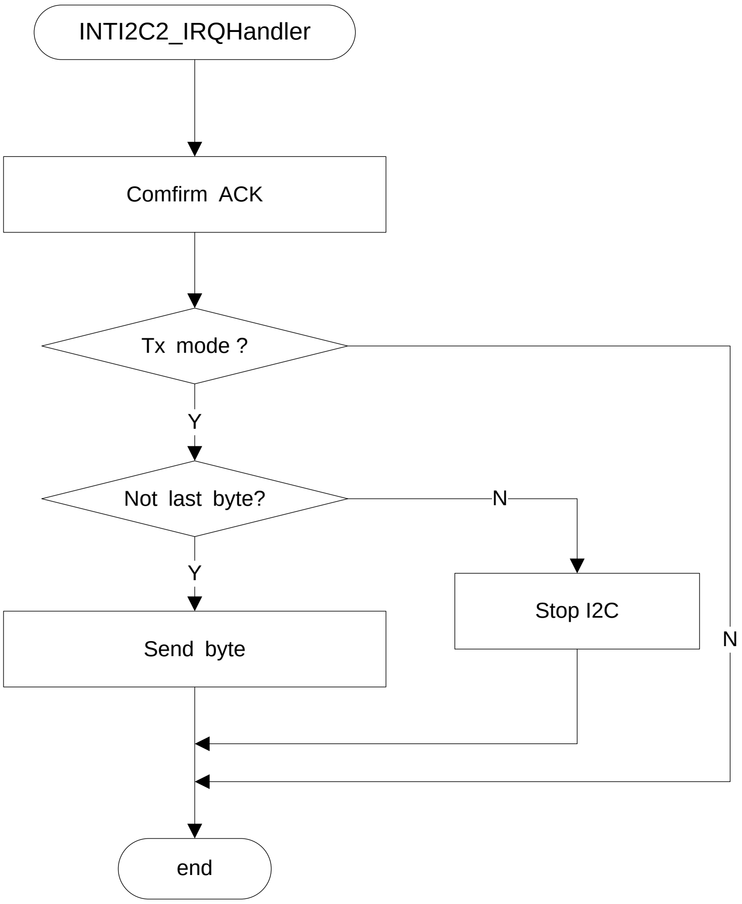
This is a simple example based on the TX04 Peripheral Driver (I2C, GPIO).

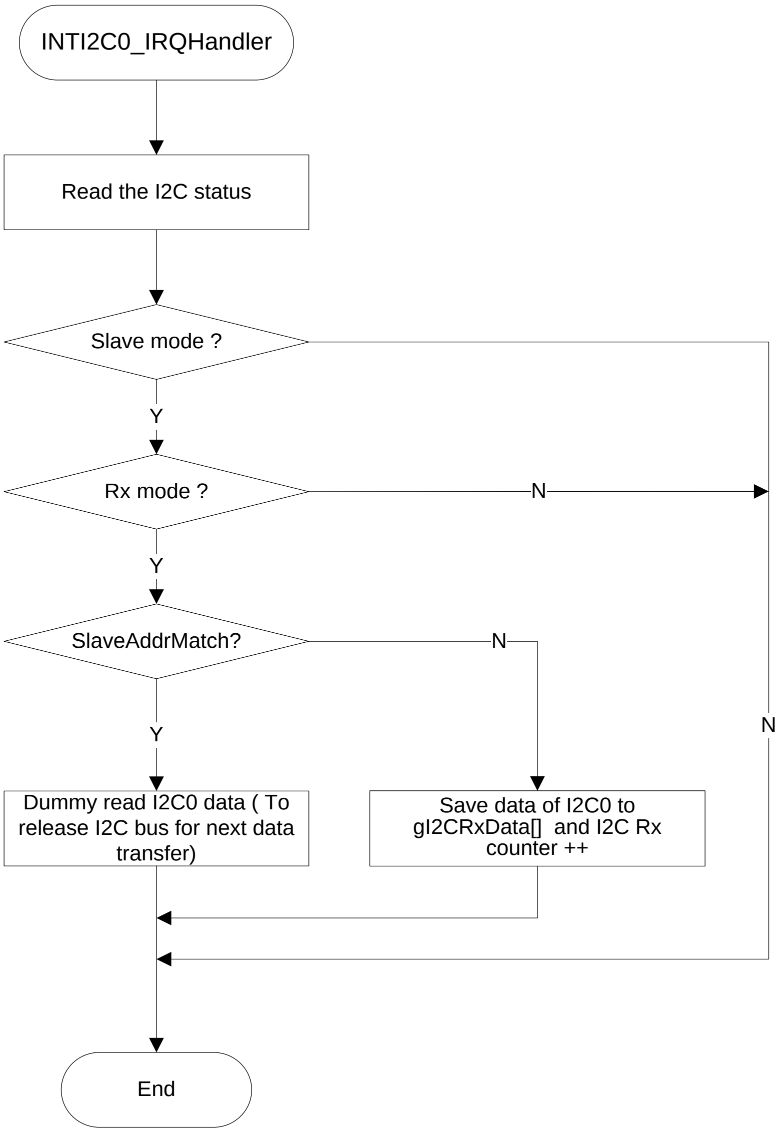
The example includes:

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

- **Flowchart**


main
Set GPIO and Initialize I2C2 as master;
Enable I2C2 interrupt
Set GPIO and Initialize I2C0 as slave;
Enable I2C0 interrupt
SW3 pressed ?
N
Y
Set I2C Tx data buffer and Tx number
Set I2C Rx number
I2C2 I2C start
I2C end ?
N
Y
Check I2C received data of I2C0
SW1 pressed ?
N
Y
UART prints data in I2CRcvBuffer


INTI2C2_IRQHandler
Comfirm ACK
Tx mode ?
Y
Not last byte?
N
Y
Stop I2C
N
Send byte
end



- **Code and Explanation for the Example**

At first, configure GPIO for I2C2 I2C mode.

```
GPIO_EnableFuncReg(GPIO_PH, GPIO_FUNC_REG_4, GPIO_BIT_0);
GPIO_EnableFuncReg(GPIO_PH, GPIO_FUNC_REG_4, GPIO_BIT_1);
GPIO_SetOutputEnableReg(GPIO_PH, GPIO_BIT_0, ENABLE);
GPIO_SetOutputEnableReg(GPIO_PH, GPIO_BIT_1, ENABLE);
GPIO_SetInputEnableReg(GPIO_PH, GPIO_BIT_0, ENABLE);
GPIO_SetInputEnableReg(GPIO_PH, GPIO_BIT_1, ENABLE);
GPIO_SetOpenDrain(GPIO_PH, GPIO_BIT_0, ENABLE);
GPIO_SetOpenDrain(GPIO_PH, GPIO_BIT_1, ENABLE);
```

Configure GPIO for I2C0 I2C mode.

```
GPIO_EnableFuncReg(GPIO_PK, GPIO_FUNC_REG_3, GPIO_BIT_2);
GPIO_EnableFuncReg(GPIO_PK, GPIO_FUNC_REG_3, GPIO_BIT_3);
GPIO_SetOutputEnableReg(GPIO_PK, GPIO_BIT_2, ENABLE);
```

```
        GPIO_SetOutputEnableReg(GPIO_PK, GPIO_BIT_3, ENABLE);
        GPIO_SetInputEnableReg(GPIO_PK, GPIO_BIT_2, ENABLE);
        GPIO_SetInputEnableReg(GPIO_PK, GPIO_BIT_3, ENABLE);
        GPIO_SetOpenDrain(GPIO_PK, GPIO_BIT_2, ENABLE);
        GPIO_SetOpenDrain(GPIO_PK, GPIO_BIT_3, ENABLE);
```

Then enable, initialize and configure I2C2 channel and enable INTI2C2.

```
    myI2C.I2CSelfAddr = SELF_ADDR;
    myI2C.I2CDataLen = I2C_DATA_LEN_8;
    myI2C.I2CACKState = ENABLE;
    myI2C.I2CClkDiv = I2C_SCK_CLK_DIV_32;
    myI2C.PrescalerClkDiv = I2C_PRESCALER_DIV_12;
    I2C_SWReset(TSB_I2C2);
    I2C_Init(TSB_I2C2, &myI2C);
    NVIC_EnableIRQ(INTI2C2_IRQn);
    I2C_SetINTReq(TSB_I2C2,ENABLE);
```

Initialize and configure I2C0 channel and enable INTI2C0.

```
    myI2C.I2CSelfAddr = SLAVE_ADDR;
    myI2C.I2CDataLen = I2C_DATA_LEN_8;
    myI2C.I2CACKState = ENABLE;
    myI2C.I2CClkDiv = I2C_SCK_CLK_DIV_32;
    myI2C.PrescalerClkDiv = I2C_PRESCALER_DIV_12;
    I2C_SWReset(TSB_I2C0);
    I2C_Init(TSB_I2C0, &myI2C);
    NVIC_EnableIRQ(INTI2C0_IRQn);
    I2C_SetINTReq(TSB_I2C0,ENABLE);
```

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

```
            /* Initialize TRx buffer and Tx length */
        case MODE_I2C_INITIAL:
            gI2CTxDataLen = 8U;
            gI2CTxData[0] = gI2CTxDataLen;
            gI2CTxData[1] = 'T';
            gI2CTxData[2] = 'O';
            gI2CTxData[3] = 'S';
            gI2CTxData[4] = 'H';
            gI2CTxData[5] = 'I';
            gI2CTxData[6] = 'B';
            gI2CTxData[7] = 'A';
            gI2CWCnt = 0U;
            for (gICnt = 0U; gICnt < 8U; gICnt++) {
                gI2CRxData[gICnt] = 0U;
            }
            gI2CMode = MODE_I2C_START;
            break;
```

Check if the I2C bus is free or not, I2C_SetSendData() is used to set data "SLAVE_ADDR".and direction is "I2C_SEND" to I2C data buffer; then I2C_GenerateStart(TSB_I2C2) is used to to start I2C process.

```
            /* Check I2C bus state and start TRx */
        case MODE_I2C_START:
            i2c_state = I2C_GetState(TSB_I2C2);
            if (!i2c_state.Bit.BusState) {
```

```
            I2C_SetSendData(TSB_I2C2, SLAVE_ADDR | I2C_SEND);
            I2C_GenerateStart(TSB_I2C2);
            gI2CMode = MODE_I2C_TRX;
        } else {
            /* Do nothing */
        }
        break;
```

The data transfer is handled in INTI2C2.

The data receive is handled in INTI2C0.

In INTI2C2 function, I2C master send process is handled according to I2C bus state. During I2C master sending, I2C_SetSendData() is used to send next data, I2C_GenerateStop() is used to stop I2C when I2C process is finished.

```
void INTI2C2_IRQHandler(void)
{
    TSB_I2C_TypeDef *I2Cx;
    I2C_State I2c_sr;

    I2Cx = TSB_I2C2;
    I2c_sr = I2C_GetState(I2Cx);

    if (i2c_sr.Bit.MasterSlave) {           /* Master mode */
        if (i2c_sr.Bit.TRx) {     /* Tx mode */
            if (i2c_sr.Bit.LastRxBit) { /* LRB=1: the receiver requires no further
data. */
                I2C_GenerateStop(I2Cx);
            } else {                /* LRB=0: the receiver requires further data. */
                if (gI2CWCnt <= gI2CTxDataLen) {
                    I2C_SetSendData(I2Cx, gI2CTxData[gI2CWCnt]);            /*
Send next data */
                    gI2CWCnt++;
                } else {            /* I2C data send finished. */
                    I2C_GenerateStop(I2Cx);    /* Stop I2C */
                }
            }
        } else {                    /* Rx Mode */
            /* Do nothing */
        }
    } else {                        /* Slave mode */
        /* Do nothing */
    }
}
```

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

```
void INTI2C0_IRQHandler(void)
{
    uint32_t tmp = 0U;
    TSB_I2C_TypeDef *I2Cy;
    I2C_State i2c0_sr;

    I2Cy = TSB_I2C0;
    i2c0_sr = I2C_GetState(I2Cy);
```

```
    if (!i2c0_sr.Bit.MasterSlave) {         /* Slave mode */
        if (!i2c0_sr.Bit.TRx) { /* Rx Mode */
            if (i2c0_sr.Bit.SlaveAddrMatch) {
                /* First read is dummy read for Slave address recognize */
                tmp = I2C_GetReceiveData(I2Cy);
                gI2CRCnt = 0U;
            } else {
                /* Read I2C received data and save to I2C_RxData buffer */
                tmp = I2C_GetReceiveData(I2Cy);
                gI2CRxData[gI2CRCnt] = tmp;
                gI2CRCnt++;
            }
        } else {                        /* Tx Mode */
            /* Do nothing */
        }
    } else {                            /* Master mode */
        /* Do nothing */
    }
}
```

7-10 IGBT

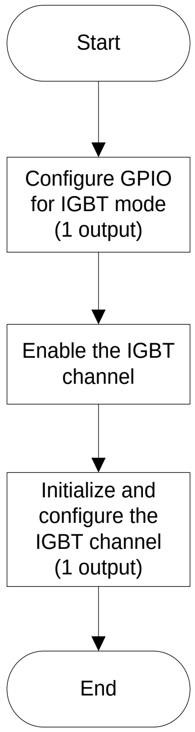
7-10-1 Example: Single PPG Output

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

The example includes:

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

- **Flowchart**



- **Code and Explanation for the Example**

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

```
IGBT_InitTypeDef myIGBT;
/* IGBT trigger start: falling edge start and active level is "Low" */
myIGBT.StartMode = IGBT_FALLING_TRG_START;
/* IGBT operation: continuous operation */
myIGBT.OperationMode = IGBT_CONTINUOUS_OUTPUT;
/* IGBT stopping status: initial output satus and counter */
myIGBT.CntStopState = IGBT_OUTPUT_INACTIVE;
/* Trigger edge accept mode: Don't accept trigger during active level */
myIGBT.ActiveAcceptTrg = DISABLE;
/* Interrupt cycle: Every one cycle */
myIGBT.INTPeriod = IGBT_INT_PERIOD_1;
/* For M46B: fperiph = fc = 16MHz*4 = 64MHz, T0 = fperiph = 64MHz, figbt = T0/2 = 32MHz */
myIGBT.ClkDiv = IGBT_CLK_DIV_2;
/* MT0OUT0 initial state is Low, and active level is High */
myIGBT.Output0Init = IGBT_OUTPUT_HIGH_ACTIVE;
/* Disable MT0OUT1 output */
myIGBT.Output1Init = IGBT_OUTPUT_DISABLE;
/* Trigger input noise elimination time: 240/fsys */
myIGBT.TrgDenoiseDiv = IGBT_DENOISE_DIV_240;
myIGBT.Output0ActiveTiming = 1U;
```

```
    myIGBT.Output0InactiveTiming = 1U + IGBT_PPG_PERIOD_50US / 2;
    /* Period: 50us */
    myIGBT.Period = IGBT_ PPG _PERIOD_50US;
    /* The polarity of MTOUT0x at EMG protection: High-impedance */
    myIGBT.EMGFunction = IGBT_EMG_OUTPUT_HIZ;
    /* EMG input noise elimination time: 240/fsys */
    myIGBT.EMGDenoiseDiv = IGBT_DENOISE_DIV_240;
```

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

```
    /* Enable IGBT and EMG interrupt */
    IGBT_Enable(IGBT0);
    NVIC_EnableIRQ(INTMTEMG0_IRQn);

    /* If the timer is still running, wait until it stops */
    do {
        counter_state = IGBT_GetCntState(IGBT0);
    } while (counter_state == BUSY);
    /* Cancel the EMG protection state */
    do {
        cancel_result = IGBT_CancelEMGState(IGBT0);
    } while (cancel_result == ERROR);
```

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

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

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

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

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

If EMG protection level on GEMG is active, the EMG interrupt occurs. Turn on the LED1 and stop the timer.

```
  void INTMTEMG0_IRQHandler(void)
  {
      /* If EMG protection, turn on the LED and stop the timer */
      LED_On(LED1);
      IGBT_SetSWRunState(IGBT0, IGBT_STOP);
  }
```

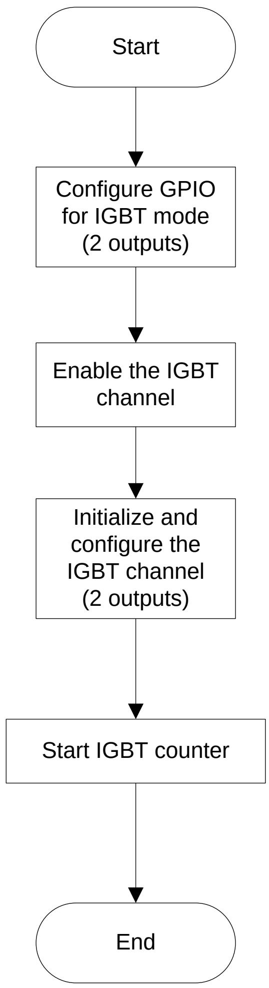
7-10-2 Example: Double PPG Output

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

The example includes:

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

- **Flowchart**



- **Code and Explanation for the Example**

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

```
IGBT_InitTypeDef myIGBT;
/* IGBT trigger start: command start */
```

```
    myIGBT.StartMode = IGBT_CMD_START;
    /* IGBT operation: continuous operation */
    myIGBT.OperationMode = IGBT_CONTINUOUS_OUTPUT;
    /* IGBT stopping status: initial output satus and counter */
    myIGBT.CntStopState = IGBT_OUTPUT_INACTIVE;
    /* Trigger edge accept mode: Don't accept trigger during active level */
    myIGBT.ActiveAcceptTrg = DISABLE;
    /* Interrupt cycle: Every one cycle */
    myIGBT.INTPeriod = IGBT_INT_PERIOD_1;
    /* For M46B: fperiph = fc = 16MHz*4 = 64MHz, T0 = fperiph = 64MHz, figbt = T0/2 =
32MHz */
    myIGBT.ClkDiv = IGBT_CLK_DIV_2;
    /* MT0OUT0 initial state is Low, and active level is High */
    myIGBT.Output0Init = IGBT_OUTPUT_HIGH_ACTIVE;
    /* MT0OUT1 initial state is Low, and active level is High */
    myIGBT.Output1Init = IGBT_OUTPUT_HIGH_ACTIVE;
    /* Trigger input noise elimination time: no use */
    myIGBT.TrgDenoiseDiv = IGBT_NO_DENOISE;
    myIGBT.Output0ActiveTiming = 1U;
    myIGBT.Output0InactiveTiming = 1U + IGBT_ACTIVE_PERIOD_20US;
    myIGBT.Output1ActiveTiming = myIGBT.Output0InactiveTiming +
                                    IGBT_DEAD_TIME_5US;
    myIGBT.Output1InactiveTiming = myIGBT.Output1ActiveTiming +
                                    IGBT_ACTIVE_PERIOD_20US;
    /* Period: 50us */
    myIGBT.Period = IGBT_PPG_PERIOD_50US;
    /* The polarity of MTOUT0x at EMG protection: High-impedance */
    myIGBT.EMGFunction = IGBT_EMG_OUTPUT_HIZ;
    /* EMG input noise elimination time: 240/fsys */
    myIGBT.EMGDenoiseDiv = IGBT_DENOISE_DIV_240;
```

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

```
    /* Enable IGBT and EMG interrupt */
    IGBT_Enable(IGBT0);
    NVIC_EnableIRQ(INTMTEMG0_IRQn);

    /* If the timer is still running, wait until it stops */
    do {
        counter_state = IGBT_GetCntState(IGBT0);
    } while (counter_state == BUSY);
    /* Cancel the EMG protection state */
    do {
        cancel_result = IGBT_CancelEMGState(IGBT0);
```

```
    } while (cancel_result == ERROR);
```

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

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

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

```
/* If switch is ON, start to run IGBT timer */
if (SWITCH_ON) {
    IGBT_SetSWRunState(IGBT0, IGBT_RUN);
} else {
  /* Do nothing */
}
```

If EMG protection level on GEMG is active, the EMG interrupt occurs. Turn on the LED1 and stop the timer.

```
void INTMTEMG0_IRQHandler(void)
{
        /* If EMG protection, turn on the LED and stop the timer */
        LED_On(LED1);
        IGBT_SetSWRunState(IGBT0, IGBT_STOP);
}
```

7-11 LVD

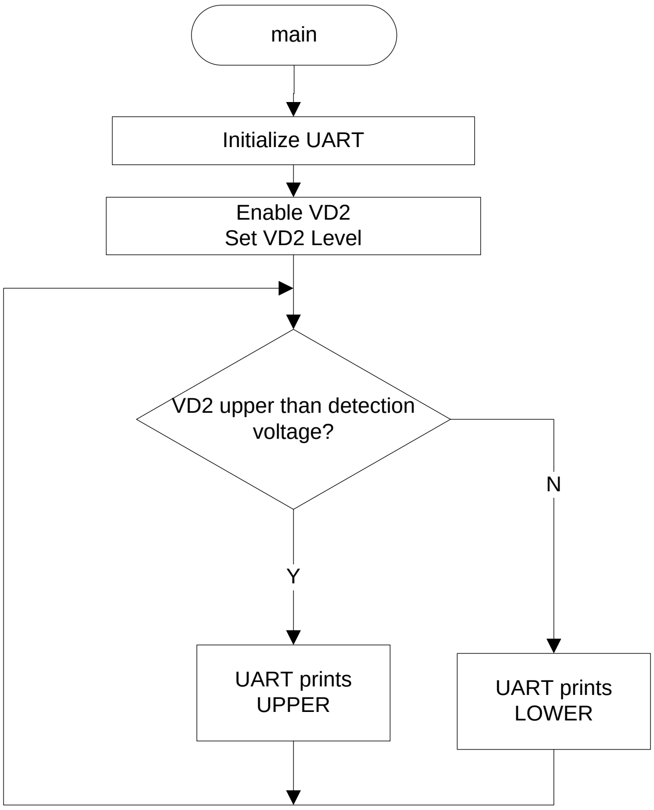
7-11-1 Example: LVD

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

The example includes:

1. LVD configuration
2. LVD status monitoring

- **Flowchart**



- **Code and Explanation for the Example**

Initialize the system first. Disabe WDT explicitly to prevent confusion. After that, initialize UART.

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

Enable voltage detection; setting mode and detection level.

```
LVD_EnableVD();
LVD_SetVDLevel(LVD_VDLVL_315);
```

Then a usual while(1) loop.

Check VD status, if VD is upper than the detection voltage, UART displays “UPPER”.

When VD is lower than the detection voltage, UART displays “LOWER”.

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

7-12 MLA

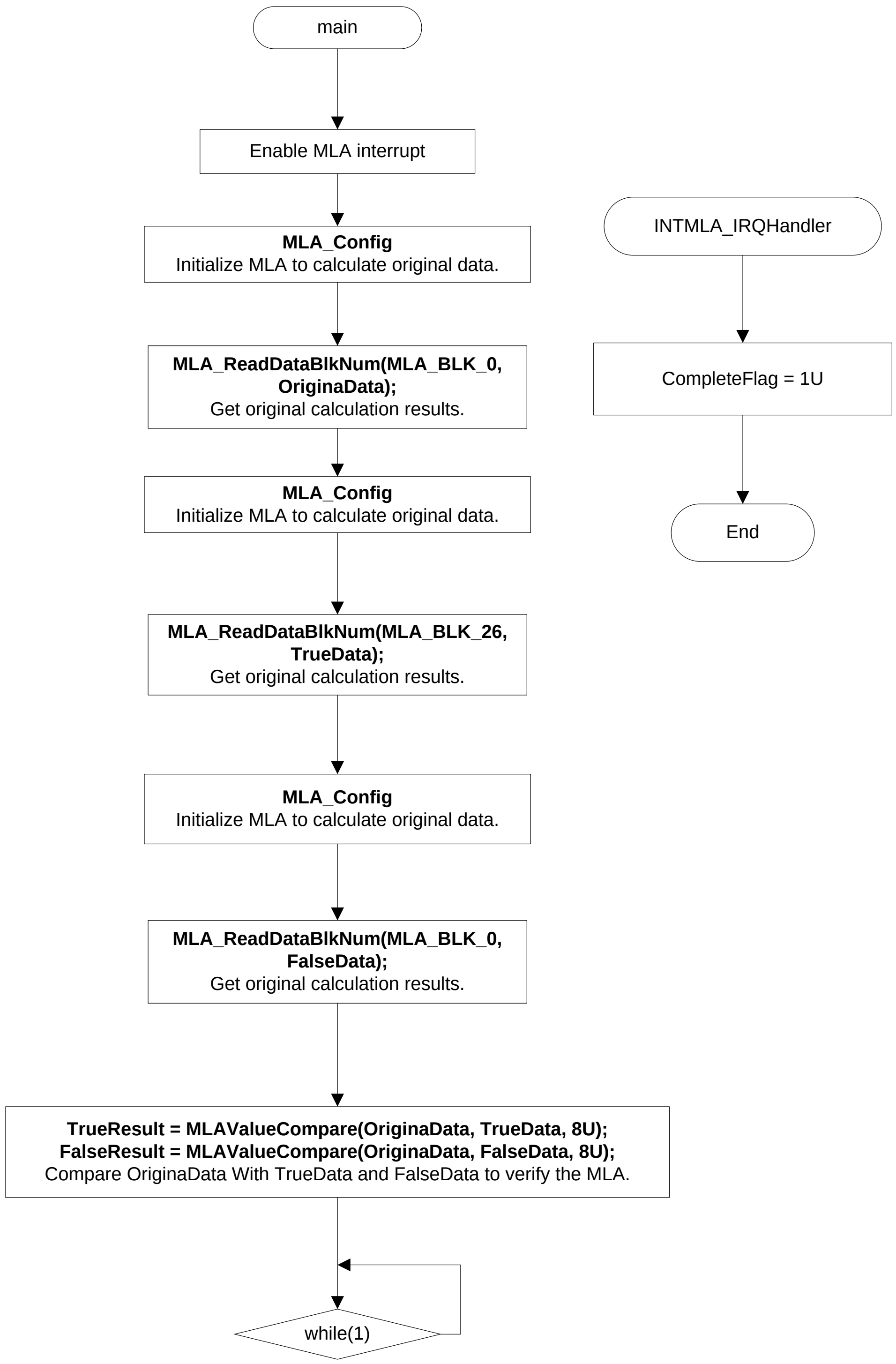
7-12-1 Example: MLA Montgomery multiplication mode

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

The example includes:

1. Montgomery multiplication mode configuration.
2. Get the calculation result process.

- **Flowchart**


main
Enable MLA interrupt
MLA_Config
Initialize MLA to calculate original data.
MLA_ReadDataBlkNum(MLA_BLK_0, OriginaData);
Get original calculation results.
MLA_Config
Initialize MLA to calculate original data.
MLA_ReadDataBlkNum(MLA_BLK_26, TrueData);
Get original calculation results.
MLA_Config
Initialize MLA to calculate original data.
MLA_ReadDataBlkNum(MLA_BLK_0, FalseData);
Get original calculation results.
TrueResult = MLAValueCompare(OriginaData, TrueData, 8U);
FalseResult = MLAValueCompare(OriginaData, FalseData, 8U);
Compare OriginaData With TrueData and FalseData to verify the MLA.
while(1)
INTMLA_IRQHandler
CompleteFlag = 1U
End

- **Code and Explanation for the Example**

Set up MLA interrupts

```
/* enable MLA calculation completion interrupt */
NVIC_ClearPendingIRQ(INTMLA_IRQn);
NVIC_EnableIRQ(INTMLA_IRQn);
  __enable_irq();
```

Set calculation data input to the specified data block, Set Montgomery multiplication divisor factor and montgomery parameter. When Montgomery calculation is performed, MLABLK01 is used to store the divisor.

```
/* Sets calculation input data. */
MLA_WriteDataBlkNum(MLA_BLK_2, OriginaAData);
MLA_WriteDataBlkNum(MLA_BLK_5, OriginaBData);
/* Sets the divisor used to Montgomery calculation */
MLA_WriteDataBlkNum(MLA_BLK_1, DivisorData) ;
/* Sets montgomery parameter */
MLA_SetMontgomeryParameter(0x12345678);
MLA_Config(MLA_BLK_2, MLA_BLK_5, MLA_BLK_0);
```

Initialize MLA calculation mode, Configure MLA processor to Montgomery multiplication mode, block number for the calculation data input register and block number for the calculation result output register.

```
/* Initialize MLA calculation mode */
void MLA_Config(uint8_t ABlkNum, uint8_t BBlkNum, uint8_t WBlkNum)
{
    /* Check the parameters */
    assert_param(IS_MLA_BLK_NUM(ABlkNum));
    assert_param(IS_MLA_BLK_NUM(BBlkNum));
    assert_param(IS_MLA_BLK_NUM(WBlkNum));

    /* Set the data block number */
    MLA_SetADataBlkNum(ABlkNum);
    Delay();
    MLA_SetBDataBlkNum(BBlkNum);
    Delay();
    MLA_SetWDataBlkNum(WBlkNum);
    Delay();

    /* Set the calculation mode */
    MLA_SetCalculationMode(MLA_COM_MODE_MUL);
}
```

Get the calculation result and print it out.

```
        /* Wait for calculation completion */
        while (CompleteFlag != 1U) {
            /* Do nothing */
        }
        /* Get the calculation result */
        Delay();
        MLA_ReadDataBlkNum(MLA_BLK_0, OriginaData);
```

Set calculation data input to the specified data block, Set Montgomery multiplication divisor factor and montgomery parameter. When Montgomery calculation is performed, MLABLK01 is used to store the divisor.

```
/* Sets calculation input data. */
MLA_WriteDataBlkNum(MLA_BLK_20, TrueAData);
MLA_WriteDataBlkNum(MLA_BLK_22, TrueBData);
/* Sets the divisor used to Montgomery calculation */
MLA_WriteDataBlkNum(MLA_BLK_1, DivisorData) ;
/* Sets montgomery parameter */
MLA_SetMontgomeryParameter(0x12345678);
MLA_Config(MLA_BLK_20, MLA_BLK_22, MLA_BLK_26);
```

Initialize MLA calculation mode, Configure MLA processor to Montgomery multiplication mode, block number for the calculation data input register and block number for the calculation result output register.

```
/* Initialize MLA calculation mode */
void MLA_Config(uint8_t ABlkNum, uint8_t BBlkNum, uint8_t WBlkNum)
{
    /* Check the parameters */
    assert_param(IS_MLA_BLK_NUM(ABlkNum));
    assert_param(IS_MLA_BLK_NUM(BBlkNum));
    assert_param(IS_MLA_BLK_NUM(WBlkNum));

    /* Set the data block number */
    MLA_SetADataBlkNum(ABlkNum);
    Delay();
    MLA_SetBDataBlkNum(BBlkNum);
    Delay();
    MLA_SetWDataBlkNum(WBlkNum);
    Delay();

    /* Set the calculation mode */
    MLA_SetCalculationMode(MLA_COM_MODE_MUL);
}
```

Get the calculation result and print it out.

```
        /* Wait for calculation completion */
        while (CompleteFlag != 1U) {
            /* Do nothing */
        }
        /* Get the calculation result */
        Delay();
        MLA_ReadDataBlkNum(MLA_BLK_26, TrueData);
```

Set calculation data input to the specified data block, Set Montgomery multiplication divisor factor and montgomery parameter. When Montgomery calculation is performed, MLABLK01 is used to store the divisor.

```
/* Sets calculation input data. */
MLA_WriteDataBlkNum(MLA_BLK_2, FalseAData);
MLA_WriteDataBlkNum(MLA_BLK_5, FalseBData);
/* Sets the divisor used to Montgomery calculation */
MLA_WriteDataBlkNum(MLA_BLK_1, DivisorData) ;
/* Sets montgomery parameter */
MLA_SetMontgomeryParameter(0x12345678);
```

```
MLA_Config(MLA_BLK_2, MLA_BLK_5, MLA_BLK_0);
```

Initialize MLA calculation mode, Configure MLA processor to Montgomery multiplication mode, block number for the calculation data input register and block number for the calculation result output register.

```
/* Initialize MLA calculation mode */
void MLA_Config(uint8_t ABlkNum, uint8_t BBlkNum, uint8_t WBlkNum)
{
    /* Check the parameters */
    assert_param(IS_MLA_BLK_NUM(ABlkNum));
    assert_param(IS_MLA_BLK_NUM(BBlkNum));
    assert_param(IS_MLA_BLK_NUM(WBlkNum));

    /* Set the data block number */
    MLA_SetADataBlkNum(ABlkNum);
    Delay();
    MLA_SetBDataBlkNum(BBlkNum);
    Delay();
    MLA_SetWDataBlkNum(WBlkNum);
    Delay();

    /* Set the calculation mode */
    MLA_SetCalculationMode(MLA_COM_MODE_MUL);
}
```

Get the calculation result and print it out.

```
        /* Wait for calculation completion */
        while (CompleteFlag != 1U) {
            /* Do nothing */
        }
        /* Get the calculation result */
        Delay();
        MLA_ReadDataBlkNum(MLA_BLK_0, FalseData);
```

Compare OHashvalue with THashValue and FHashValue to verify the SHA. If TrueResult = SUCCESS and FalseResult = ERROR, it means that SHA procedure has passed verification.

```
/* Compare original message Hash value with true message Hash value. */
TrueResult = HashValueCompare(OHashValue, THashValue, 8U);
/* Compare original message Hash value with false message Hash value. */
FalseResult = HashValueCompare(OHashValue, FHashValue, 8U);

if ((TrueResult == SUCCESS) && (FalseResult == ERROR)) {
    common_uart_disp("SHA processed successfully !\n");
} else {
    common_uart_disp("SHA processed with error !\n");
}
```

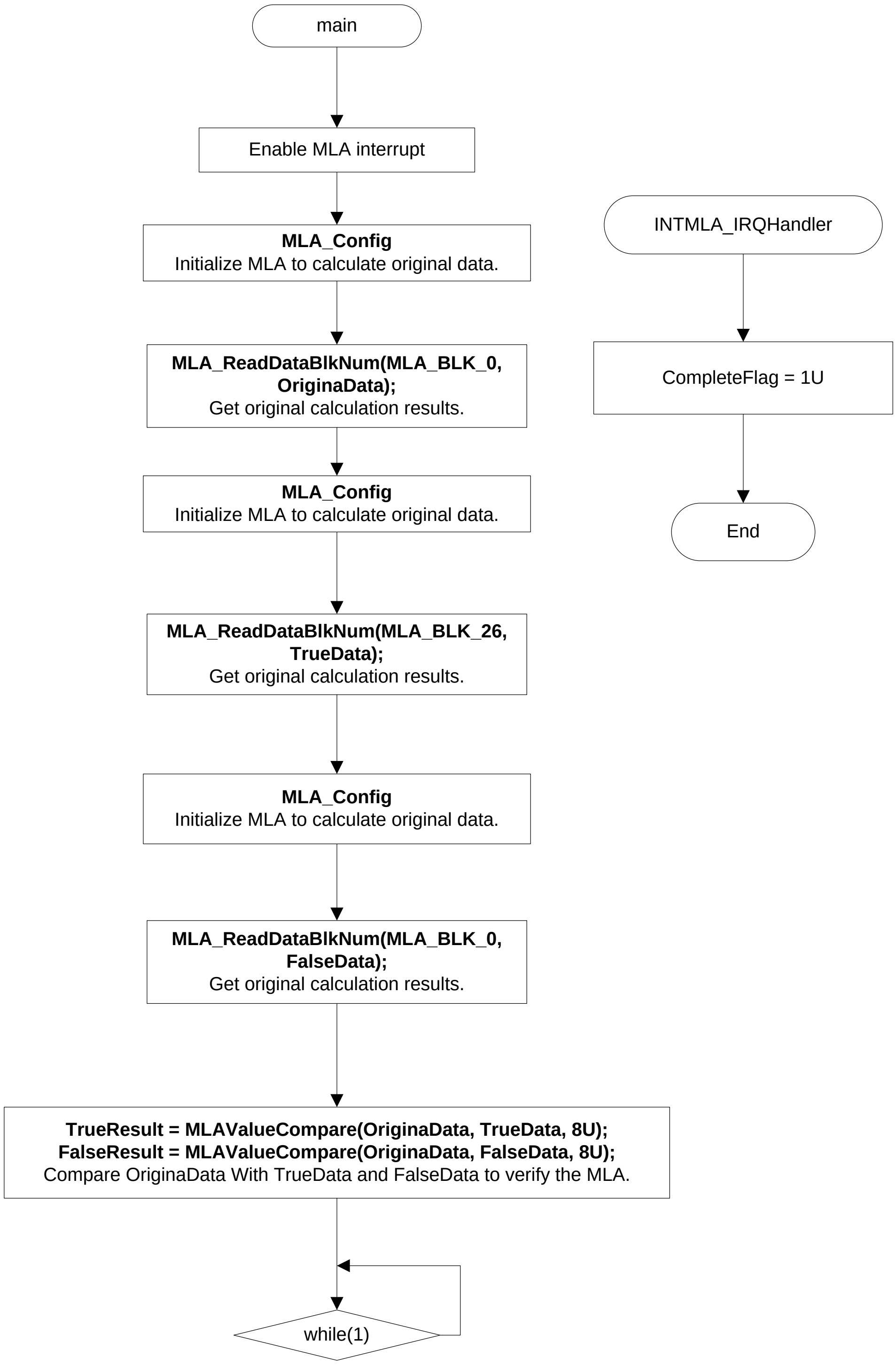
7-12-2 Example: MLA Multiple length addition mode

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

The example includes:

1. Montgomery multiplication mode configuration.
2. Get the calculation result process.

- **Flowchart**


main
Enable MLA interrupt
MLA_Config
Initialize MLA to calculate original data.
MLA_ReadDataBlkNum(MLA_BLK_0, OriginaData);
Get original calculation results.
MLA_Config
Initialize MLA to calculate original data.
MLA_ReadDataBlkNum(MLA_BLK_26, TrueData);
Get original calculation results.
MLA_Config
Initialize MLA to calculate original data.
MLA_ReadDataBlkNum(MLA_BLK_0, FalseData);
Get original calculation results.
TrueResult = MLAValueCompare(OriginaData, TrueData, 8U);
FalseResult = MLAValueCompare(OriginaData, FalseData, 8U);
Compare OriginaData With TrueData and FalseData to verify the MLA.
while(1)
INTMLA_IRQHandler
CompleteFlag = 1U
End

- **Code and Explanation for the Example**

Set up MLA interrupts

```
/* enable MLA calculation completion interrupt */
NVIC_ClearPendingIRQ(INTMLA_IRQn);
NVIC_EnableIRQ(INTMLA_IRQn);
__enable_irq();
```

Set calculation data input to the specified data block.

```
/* Sets calculation input data. */
MLA_WriteDataBlkNum(MLA_BLK_2, OriginaAData);
MLA_WriteDataBlkNum(MLA_BLK_5, OriginaBData);
MLA_Config(MLA_BLK_2, MLA_BLK_5, MLA_BLK_0);
```

Initialize MLA calculation mode, Configure MLA processor to Montgomery multiplication mode, block number for the calculation data input register and block number for the calculation result output register.

```
/* Initialize MLA calculation mode */
void MLA_Config(uint8_t ABlkNum, uint8_t BBlkNum, uint8_t WBlkNum)
{
    /* Check the parameters */
    assert_param(IS_MLA_BLK_NUM(ABlkNum));
    assert_param(IS_MLA_BLK_NUM(BBlkNum));
    assert_param(IS_MLA_BLK_NUM(WBlkNum));

    /* Set the data block number */
    MLA_SetADataBlkNum(ABlkNum);
    Delay();
    MLA_SetBDataBlkNum(BBlkNum);
    Delay();
    MLA_SetWDataBlkNum(WBlkNum);
    Delay();

    /* Set the calculation mode */
    MLA_SetCalculationMode(MLA_COM_MODE_ADD);
}
```

Get the calculation result and print it out.

```
/* Wait for calculation completion */
        while (CompleteFlag != 1U) {
            /* Do nothing */
        }
        /* Get the calculation result */
        MLA_ReadDataBlkNum(MLA_BLK_0, OriginaData);
```

Set calculation data input to the specified data block.

```
/* Sets calculation input data. */
MLA_WriteDataBlkNum(MLA_BLK_20, TrueAData);
MLA_WriteDataBlkNum(MLA_BLK_22, TrueBData);
MLA_Config(MLA_BLK_20, MLA_BLK_22, MLA_BLK_26);
```

Initialize MLA calculation mode, Configure MLA processor to Montgomery multiplication mode, block number for the calculation data input register and block number for the calculation result output register.

```
/* Initialize MLA calculation mode */
void MLA_Config(uint8_t ABlkNum, uint8_t BBlkNum, uint8_t WBlkNum)
{
    /* Check the parameters */
    assert_param(IS_MLA_BLK_NUM(ABlkNum));
    assert_param(IS_MLA_BLK_NUM(BBlkNum));
    assert_param(IS_MLA_BLK_NUM(WBlkNum));

    /* Set the data block number */
    MLA_SetADataBlkNum(ABlkNum);
    Delay();
    MLA_SetBDataBlkNum(BBlkNum);
    Delay();
    MLA_SetWDataBlkNum(WBlkNum);
    Delay();

    /* Set the calculation mode */
    MLA_SetCalculationMode(MLA_COM_MODE_ADD);
}
```

Get the calculation result and print it out.

```
/* Wait for calculation completion */
        while (CompleteFlag != 1U) {
            /* Do nothing */
        }
        /* Get the calculation result */
        MLA_ReadDataBlkNum(MLA_BLK_26, TrueData);
```

Set calculation data input to the specified data block.

```
/* Sets calculation input data. */
MLA_WriteDataBlkNum(MLA_BLK_2, FalseAData);
MLA_WriteDataBlkNum(MLA_BLK_5, FalseBData);
MLA_Config(MLA_BLK_2, MLA_BLK_5, MLA_BLK_0);
```

Initialize MLA calculation mode, Configure MLA processor to Montgomery multiplication mode, block number for the calculation data input register and block number for the calculation result output register.

```
/* Initialize MLA calculation mode */
void MLA_Config(uint8_t ABlkNum, uint8_t BBlkNum, uint8_t WBlkNum)
{
    /* Check the parameters */
    assert_param(IS_MLA_BLK_NUM(ABlkNum));
    assert_param(IS_MLA_BLK_NUM(BBlkNum));
    assert_param(IS_MLA_BLK_NUM(WBlkNum));

    /* Set the data block number */
    MLA_SetADataBlkNum(ABlkNum);
    Delay();
    MLA_SetBDataBlkNum(BBlkNum);
    Delay();
    MLA_SetWDataBlkNum(WBlkNum);
    Delay();
```

```
        /* Set the calculation mode */
        MLA_SetCalculationMode(MLA_COM_MODE_ADD);
    }
```

Get the calculation result and print it out.

```
    /* Wait for calculation completion */
            while (CompleteFlag != 1U) {
                /* Do nothing */
            }
            /* Get the calculation result */
            MLA_ReadDataBlkNum(MLA_BLK_0, FalseData);
```

Compare OHashvalue with THashValue and FHashValue to verify the SHA. If TrueResult = SUCCESS and FalseResult = ERROR, it means that SHA procedure has passed verification.

```
    /* Compare original message Hash value with true message Hash value. */
    TrueResult = HashValueCompare(OHashValue, THashValue, 8U);
    /* Compare original message Hash value with false message Hash value. */
    FalseResult = HashValueCompare(OHashValue, FHashValue, 8U);

    if ((TrueResult == SUCCESS) && (FalseResult == ERROR)) {
        common_uart_disp("SHA processed successfully !\n");
    } else {
        common_uart_disp("SHA processed with error !\n");
    }
```

7-12-3 Example: MLA Multiple length subtraction mode

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

The example includes:

1. Multiple length subtraction mode.
2. Get the calculation result process.

- **Flowchart**

- **Code and Explanation for the Example**

Set up MLA interrupts

```
/* enable MLA calculation completion interrupt */
NVIC_ClearPendingIRQ(INTMLA_IRQn);
NVIC_EnableIRQ(INTMLA_IRQn);
  __enable_irq();
```

Set calculation data input to the specified data block.

```
/* Sets calculation input data. */
MLA_WriteDataBlkNum(MLA_BLK_2, OriginaAData);
MLA_WriteDataBlkNum(MLA_BLK_5, OriginaBData);
MLA_Config(MLA_BLK_2, MLA_BLK_5, MLA_BLK_0);
```

Initialize MLA calculation mode, Configure MLA processor to Multiple length subtraction mode, block number for the calculation data input register and block number for the calculation result output register.

```
/* Initialize MLA calculation mode */
void MLA_Config(uint8_t ABlkNum, uint8_t BBlkNum, uint8_t WBlkNum)
```

```
{
    /* Check the parameters */
    assert_param(IS_MLA_BLK_NUM(ABlkNum));
    assert_param(IS_MLA_BLK_NUM(BBlkNum));
    assert_param(IS_MLA_BLK_NUM(WBlkNum));

    /* Set the data block number */
    MLA_SetADataBlkNum(ABlkNum);
    Delay();
    MLA_SetBDataBlkNum(BBlkNum);
    Delay();
    MLA_SetWDataBlkNum(WBlkNum);
    Delay();

    /* Set the calculation mode */
    MLA_SetCalculationMode(MLA_COM_MODE_SUB);
}
```

Get the calculation result and print it out.

```
/* Wait for calculation completion */
        while (CompleteFlag != 1U) {
            /* Do nothing */
        }
        /* Get the calculation result */
        MLA_ReadDataBlkNum(MLA_BLK_0, OriginaData);
```

Set calculation data input to the specified data block.

```
/* Sets calculation input data. */
MLA_WriteDataBlkNum(MLA_BLK_20, TrueAData);
MLA_WriteDataBlkNum(MLA_BLK_22, TrueBData);
MLA_Config(MLA_BLK_20, MLA_BLK_22, MLA_BLK_26);
```

Initialize MLA calculation mode, Configure MLA processor to Multiple length subtraction mode, block number for the calculation data input register and block number for the calculation result output register.

```
/* Initialize MLA calculation mode */
void MLA_Config(uint8_t ABlkNum, uint8_t BBlkNum, uint8_t WBlkNum)
{
    /* Check the parameters */
    assert_param(IS_MLA_BLK_NUM(ABlkNum));
    assert_param(IS_MLA_BLK_NUM(BBlkNum));
    assert_param(IS_MLA_BLK_NUM(WBlkNum));

    /* Set the data block number */
    MLA_SetADataBlkNum(ABlkNum);
    Delay();
    MLA_SetBDataBlkNum(BBlkNum);
    Delay();
    MLA_SetWDataBlkNum(WBlkNum);
    Delay();

    /* Set the calculation mode */
    MLA_SetCalculationMode(MLA_COM_MODE_SUB);
}
```

Get the calculation result and print it out.

```
/* Wait for calculation completion */
        while (CompleteFlag != 1U) {
            /* Do nothing */
        }
        /* Get the calculation result */
        MLA_ReadDataBlkNum(MLA_BLK_26, TrueData);
```

Set calculation data input to the specified data block.

```
/* Sets calculation input data. */
MLA_WriteDataBlkNum(MLA_BLK_2, FalseAData);
MLA_WriteDataBlkNum(MLA_BLK_5, FalseBData);
MLA_Config(MLA_BLK_2, MLA_BLK_5, MLA_BLK_0);
```

Initialize MLA calculation mode, Configure MLA processor to Multiple length subtraction mode, block number for the calculation data input register and block number for the calculation result output register.

```
/* Initialize MLA calculation mode */
void MLA_Config(uint8_t ABlkNum, uint8_t BBlkNum, uint8_t WBlkNum)
{
    /* Check the parameters */
    assert_param(IS_MLA_BLK_NUM(ABlkNum));
    assert_param(IS_MLA_BLK_NUM(BBlkNum));
    assert_param(IS_MLA_BLK_NUM(WBlkNum));

    /* Set the data block number */
    MLA_SetADataBlkNum(ABlkNum);
    Delay();
    MLA_SetBDataBlkNum(BBlkNum);
    Delay();
    MLA_SetWDataBlkNum(WBlkNum);
    Delay();

    /* Set the calculation mode */
    MLA_SetCalculationMode(MLA_COM_MODE_SUB);
}
```

Get the calculation result and print it out.

```
/* Wait for calculation completion */
        while (CompleteFlag != 1U) {
            /* Do nothing */
        }
        /* Get the calculation result */
        MLA_ReadDataBlkNum(MLA_BLK_0, FalseData);
```

Compare OriginaData with TrueData and FalseData to verify the MLA. If TrueResult = SUCCESS and FalseResult = ERROR, it means that MLA procedure has passed verification.

```
/* Compare original data calculation results with true data calculation results */
    TrueResult = MLAValueCompare(OriginaData, TrueData, 8U);
/* Compare original data calculation results with false data calculation results */
    FalseResult = MLAValueCompare(OriginaData, FalseData, 8U);
```

```
if ((TrueResult == SUCCESS) && (FalseResult == ERROR)) {
        common_uart_disp("SHA processed successfully !\n");
} else {
        common_uart_disp("SHA processed with error !\n");
}
```

7-13 RTC

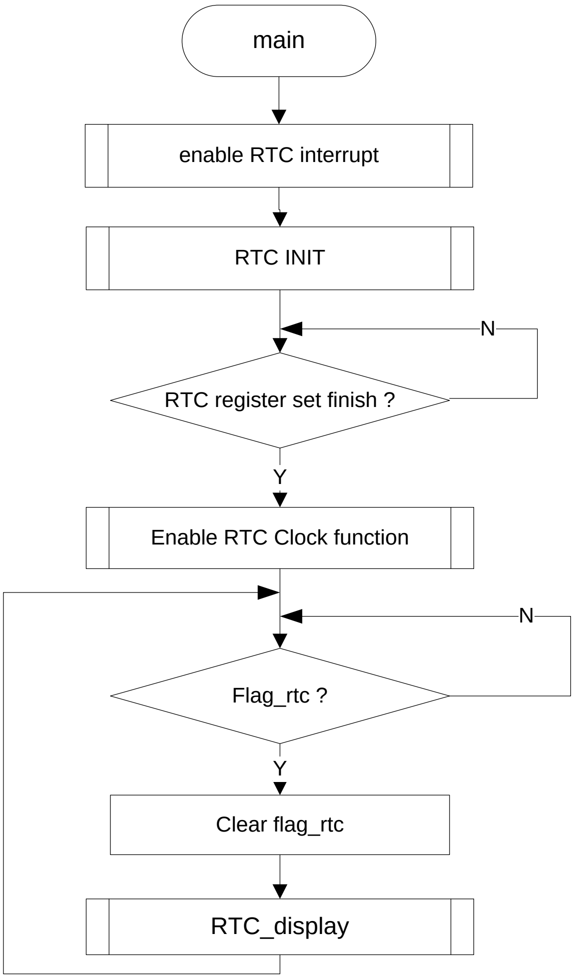
7-13-1 Example: RTC

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

The example includes:

1 RTC initialization
2 Get RTC date and time

- **Flowchart**



- **Code and Explanation for the Example**

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

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

Initialize RTC. Create a RTC_DateTypeDef and RTC_TimeTypeDef structures and fill all the data fields. For example, initial setting: 2010/10/22 12:50:55, 24hours format.

```
RTC_DateTypeDef Date_Struct;
RTC_TimeTypeDef Time_Struct;

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) 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, 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;
/* Clear RTC interrupt request */
CG_ClearINTReq(CG_INT_SRC_RTC);
```

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

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();
```

7-14 SHA

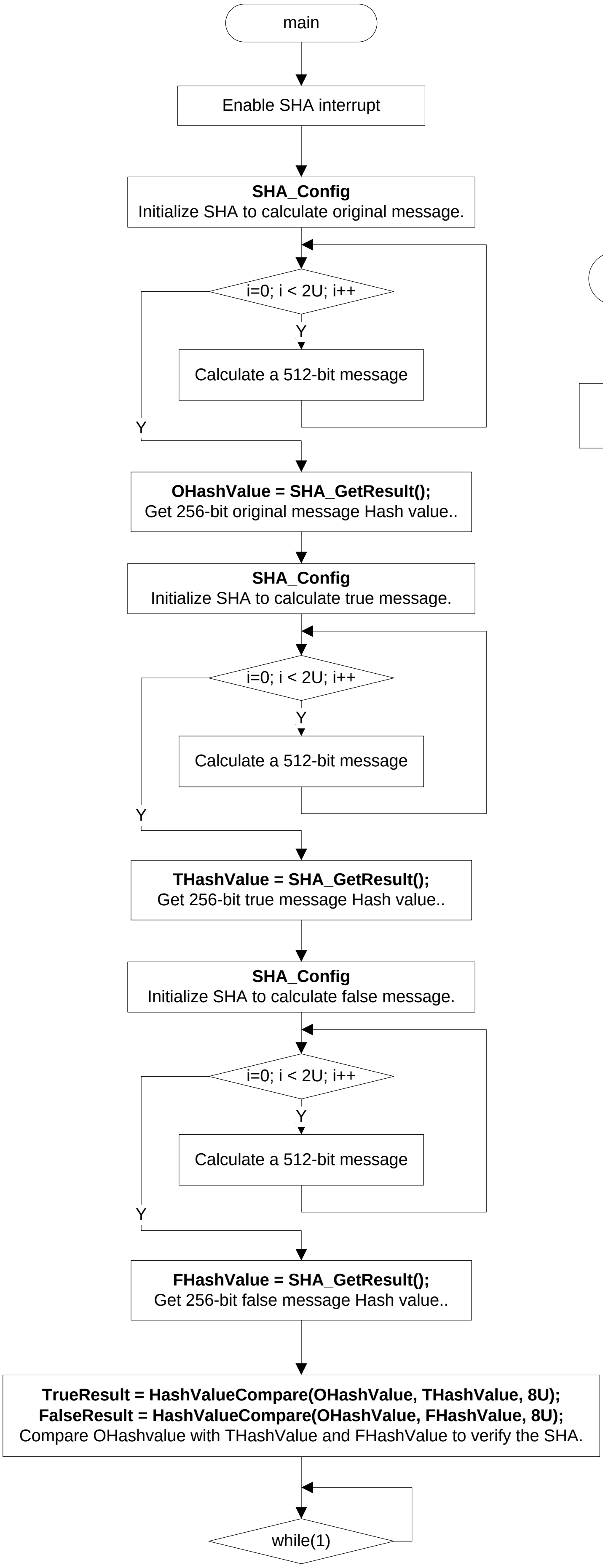
7-14-1 Example: SHA

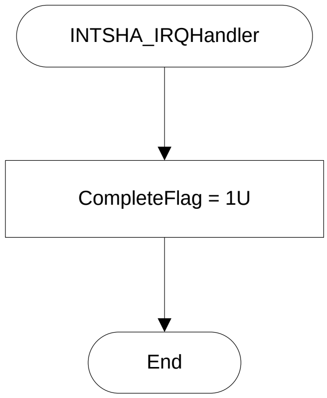
This is a simple example based on the TX04 Peripheral Driver (SHA and UART).

The example includes:

1. Initialize SHA to calculate original message.
2. Start calculation and get the original message Hash value.
3. Initialize SHA to calculate true message.
4. Start calculation and get the true message Hash value.
5. Initialize SHA to calculate false message.
6. Start calculation and get the false message Hash value.
7. Compare OHashvalue with THashValue and FHashValue to verify the SHA.

- **Flowchart**


main
Enable SHA interrupt
SHA_Config
Initialize SHA to calculate original message.
i=0; i < 2U; i++
Y
Calculate a 512-bit message
Y
OHashValue = SHA_GetResult();
Get 256-bit original message Hash value..
SHA_Config
Initialize SHA to calculate true message.
i=0; i < 2U; i++
Y
Calculate a 512-bit message
Y
THashValue = SHA_GetResult();
Get 256-bit true message Hash value..
SHA_Config
Initialize SHA to calculate false message.
i=0; i < 2U; i++
Y
Calculate a 512-bit message
Y
FHashValue = SHA_GetResult();
Get 256-bit false message Hash value..
TrueResult = HashValueCompare(OHashValue, THashValue, 8U);
FalseResult = HashValueCompare(OHashValue, FHashValue, 8U);
Compare OHashvalue with THashValue and FHashValue to verify the SHA.
while(1)


INTSHA_IRQHandler
CompleteFlag = 1U
End

- **Code and Explanation for the Example**

Set up SHA interrupt

```
/* enable SHA calculation completion interrupt */
NVIC_ClearPendingIRQ(INTSHA_IRQn);
NVIC_EnableIRQ(INTSHA_IRQn);
__enable_irq();
```

Configure the basic SHA configuration..

```
void SHA_Config(void)
{
    /* Confirm calculation completion */
    while(SHA_GetCalculationStatus() != SHA_CALCULATION_COMPLETE)
{};

    /* The Hash value specified with the SHAINITx register */
    SHA_SetInitMode(SHA_INIT_VALUE_REG);

    /* An interrupt is output at each calculation */
    SHA_SetCalculationInt(SHA_INT_EACH_CALCULATION);

    /* Set the Hash initial value */
    SHA_SetInitValue(HashInit);

    /* Set the whole message length and unhandled message length */
    SHA_SetMsgLen(MsgLen);
    SHA_SetRmnMsgLen(RmnMsgLen);
}
```

SHA generates 256-bit Hash value from original message.

```
for (i = 0U; i < 2U; i++) {
    CompleteFlag = 0U;
    /* Set a 512-bit message. */
    SHA_SetMessage(OriginalMessage[i]);
    /* Start the SHA processor. */
    SHA_SetRunState(SHA_START);
    /* Wait for calculation completion */
    while (CompleteFlag != 1U) {
        /* Do nothing */
    }
}
```

Get 256-bit original message Hash value.

```
/* Get 256-bit original message Hash value. */
SHA_GetResult(OHashValue);
```

SHA generates 256-bit Hash value from true message.

```
for (i = 0U; i < 2U; i++) {
    CompleteFlag = 0U;
    /* Set a 512-bit message. */
    SHA_SetMessage(TrueMessage[i]);
    /* Start the SHA processor. */
    SHA_SetRunState(SHA_START);
```

```
        /* Wait for calculation completion */
        while (CompleteFlag != 1U) {
            /* Do nothing */
        }
    }
```

Get 256-bit true message Hash value.

```
    /* Get 256-bit true message Hash value. */
    SHA_GetResult(THashValue);
```

SHA generates 256-bit Hash value from false message.

```
    for (i = 0U; i < 2U; i++) {
        CompleteFlag = 0U;
        /* Set a 512-bit message. */
        SHA_SetMessage(FalseMessage[i]);
        /* Start the SHA processor. */
        SHA_SetRunState(SHA_START);
        /* Wait for calculation completion */
        while (CompleteFlag != 1U) {
            /* Do nothing */
        }
    }
```

Get 256-bit false message Hash value.

```
    /* Get 256-bit false message Hash value. */
    SHA_GetResult(FHashValue);
```

Compare OHashvalue with THashValue and FHashValue to verify the SHA. If TrueResult = SUCCESS and FalseResult = ERROR, it means that SHA procedure has passed verification.

```
    /* Compare original message Hash value with true message Hash value. */
    TrueResult = HashValueCompare(OHashValue, THashValue, 8U);
    /* Compare original message Hash value with false message Hash value. */
    FalseResult = HashValueCompare(OHashValue, FHashValue, 8U);

    if ((TrueResult == SUCCESS) && (FalseResult == ERROR)) {
        common_uart_disp("SHA processed successfully !\n");
    } else {
        common_uart_disp("SHA processed with error !\n");
    }
```

7-15 SSP

7-15-1 Example: SSP0 Self Loop Back

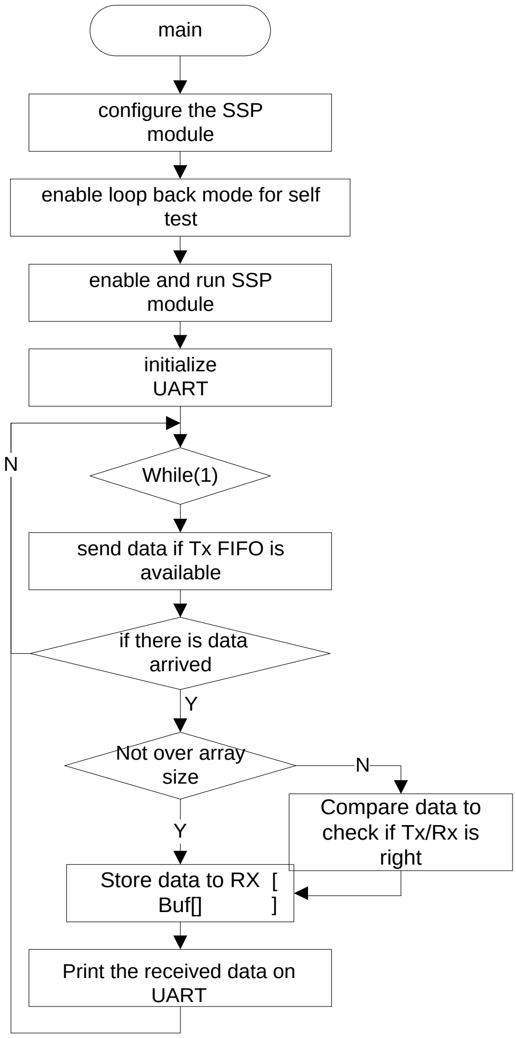
This is a simple example based on the TX04 Peripheral Driver (SSP, GPIO).

The example includes:

1. Configuration and initialization SSP module channel0

2. Enable its loop back mode to do self Tx/Rx

- **Flowchart**



- **Code and Explanation for the Example**

Select SPI frame format.

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

Setting maximum and minimum bit rate.

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

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

```
initSSP.ClkPhase = SSP_PHASE_FIRST_EDGE;
initSSP.DataSize = 16U;
initSSP.Mode = SSP_MASTER;
SSP_Init(TSB_SSP0, &initSSP);
```

Enable Loop back mode and SSP0

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

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

Setting for LED on M46B board

```
/* Initialize LEDs on M46B board before display something */
LED_Init();
hardware_init(UART_RETARGET);
```

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

```
while (1) {

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

    /* Check if there is data arrived */
    fifoState = SSP_GetFIFOState(TSB_SSP0, SSP_RX);
    if ((fifoState == SSP_FIFO_FULL) || (fifoState == SSP_FIFO_NORMAL)) {
        receive = SSP_GetRxData(TSB_SSP0);
        if (cntRx < MAX_BUFSIZE) {
            Rx_Buf[cntRx] = receive;
            cntRx++;
        } else {
            /* Place a break point here to check if receive data is right. */
            /* Success Criteria:       */
            /* Every data transmitted from Tx_Buf is received in Rx_Buf. */
            /* When the line "#define BITRATE_MIN" is commented, the SSP
               is run in maximum */
            /* bit rate, so we can find there is enough time to transmit date
               from 1 to    */
            /* MAX_BUFSIZE one by one. But if we uncomment that line,

               SSP is run in */
```

```
                /* minimum bit rate, we will find that receive data can't catch
                   "datTx++", */
                /* in this so slow bit rate, when the Tx FIFO is available, the
                   cntTx has       */
                /* been increased so much. */
                __NOP();
                result = Buffercompare(Tx_Buf, Rx_Buf, MAX_BUFSIZE);
                if (result   == NOT_SAME)
                {
                    LED_On(LED0);
                    LED_On(LED1);
                } else
                {
                    LED_On(LED2);
                    LED_On(LED3);
                }
            }

        } else {
            /* Do nothing */
        }

        sprintf((char *) SSP_RX_Data, "SSP RX DATA : %d", receive);
        common_uart_disp(SSP_RX _Data);
        common_uart_disp("\n");
    }
```

7-16 TMRB

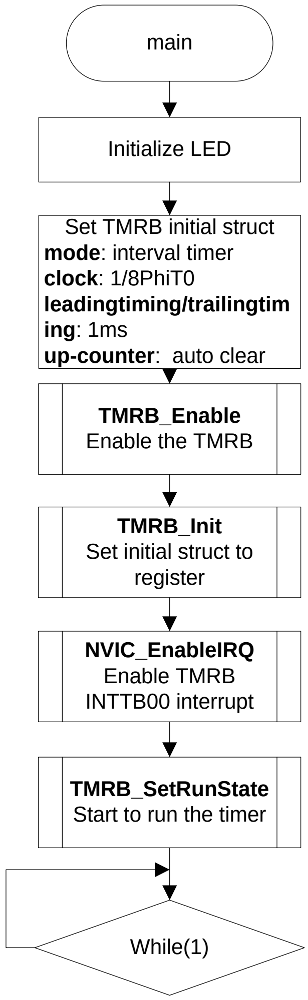
7-16-1 Example: General Timer

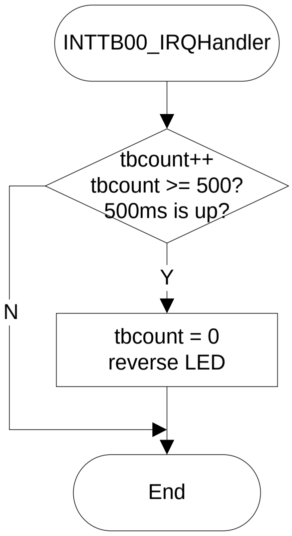
This is a simple example based on the TX04 Peripheral Driver (TMRB, GPIO).

The example includes:

1. TMRB0 initialization
2. General Timer for 1ms

- **Flowchart**





- **Code and Explanation for the Example**

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

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

Prepare TMRB initialization structure, fill TMRB mode, clock, up-counter clear method, trailingtiming and leadingtiming. This demo will set trailingtiming and leadingtiming to 1ms. The value of trailingtiming and the leadingtiming will calculate by the function Tmrb_Calculator.

```
TMRB_InitTypeDef m_tmrb;

m_tmrb.Mode = TMRB_INTERVAL_TIMER;       /* internal timer */
m_tmrb.ClkDiv = TMRB_CLK_DIV_8;          /* 1/8PhiT0 */
/* periodic time is 1ms(require 1000us) */
m_tmrb.TrailingTiming = Tmrb_Calculator(1000U, m_tmrb.ClkDiv);
m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;      /* up-counter auto clear */
```

```
/* periodic time is 1ms(require 1000us) */
m_tmrb.LeadingTiming = Tmrb_Calculator(1000U, m_tmrb.ClkDiv);
```

Enable the TMRB module and set the initialization structure to specified registers. Enable INTTB0 interrupt, this interrupt will be triggered every 1ms, in the end, start the TMRB to run.

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

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

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

The function Tmrb_Calculator :

```
uint16_t Tmrb_Calculator(uint16_t Tmrb_Require_us, uint32_t ClkDiv)
{
    uint32_t T0 = 0U;
    const uint16_t Div[8U] = {1U, 2U, 8U, 32U, 64U, 128U, 256U, 512U};

    SystemCoreClockUpdate();

    T0 = SystemCoreClock / (1U << ((TSB_CG->SYSCR >> 8U) & 7U));
    T0 = T0/((Div[ClkDiv])*1000000U);

    return(Tmrb_Require_us * T0);
}
```

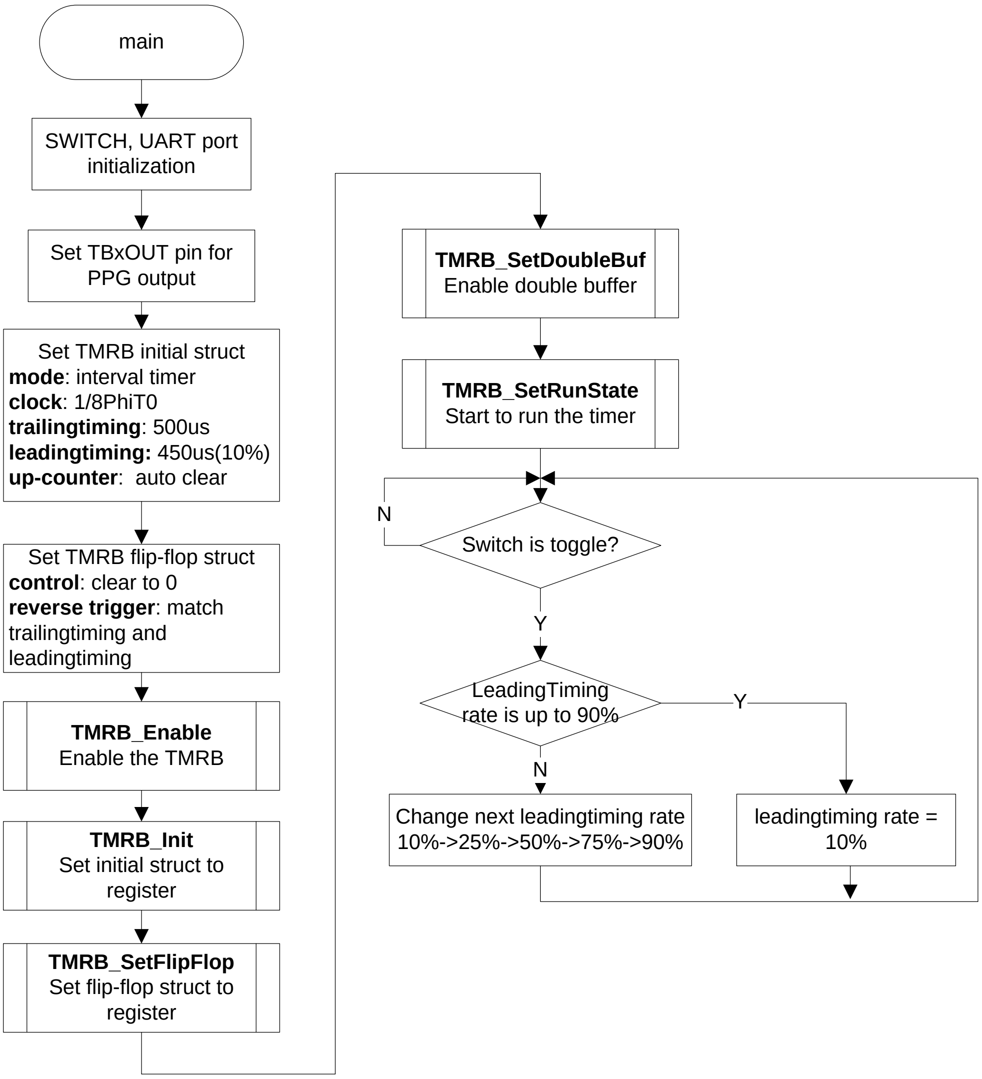
7-16-2 Example: PPG Output

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

The example includes:

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

- ## Flowchart



- ## Code and Explanation for the Example

At first, initialize targeted array tgtLeadingTiming ,LeadingTimingus and LeadingTiming .Then initialze SWITCH and UART channel, set PF1 as TB6OUT for PPG output.

```
    TMRB_InitTypeDef m_tmrb;
    TMRB_FFOutputTypeDef PPGFFInital;
    uint8_t keyvalue;
    uint32_t i = 0U;
    uint32_t tgtLeadingTiming[5U] = { 10U, 25U, 50U, 75U, 90U }; /* leadingtiming: 10%, 25%, 50%, 75%, 90% */
    uint32_t LeadingTimingus[5U] = {0U, 0U, 0U , 0U , 0U};
    uint32_t LeadingTiming[5U] = {0U, 0U, 0U , 0U , 0U};

    /* LeadingTimingus: 50, 125, 250, 375, 450 */
    for (i=0U;i<=4U;i++) {
        LeadingTimingus[i] = tgtLeadingTiming[i] * 5U;
```

```
    }

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

    /* Set PK1 as TB6OUT for PPG output */
    GPIO_SetOutput(GPIO_PK, GPIO_BIT_1);
    GPIO_EnableFuncReg(GPIO_PK, GPIO_FUNC_REG_4, GPIO_BIT_1);
```

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.The value of trailingtiming and the leadingtiming array will calculate by the function Tmrb_Calculator.

```
    TMRB_InitTypeDef m_tmrb;

    m_tmrb.Mode = TMRB_INTERVAL_TIMER;        /* internal timer */
    m_tmrb.ClkDiv = TMRB_CLK_DIV_8;                        /* 1/8PhiT0 */
    m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;        /* up-counter auto clear */
    for(i=0U;i<=4U;i++) {
        LeadingTiming[i] = Tmrb_Calculator(LeadingTimingus[i], m_tmrb.ClkDiv);
    }
    m_tmrb.TrailingTiming  =  Tmrb_Calculator(500U,  m_tmrb.ClkDiv);        /*
trailingtiming is 500us */
    m_tmrb.LeadingTiming  =  LeadingTiming[Rate];                            /*
leadingtiming, initial value 10% */
```

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

```
    PPGFFInital.FlipflopCtrl = TMRB_FLIPFLOP_SET;
    PPGFFInital.FlipflopReverseTrg=TMRB_FLIPFLOP_MATCH_TRAILING|
    TMRB_FLIPFLOP_MATCH_LEADING;
```

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

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

Wait switch from low to high, and at the same time, UART prints current leadingtiming.

```
    do {                              /* wait if switch is Low */
            keyvalue = GPIO_ReadDataBit(KEYPORT, GPIO_BIT_0);
            LeadingTiming_display(); /* display current leadingtiming */
       } while (GPIO_BIT_VALUE_0 == keyvalue);
```

If the switch is changed to high, change the leadingtiming according to

10%->25%->50%->75%->90%, then from 90% to 10% again.

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

    TMRB_ChangeLeadingTiming(TSB_TB6, LeadingTiming[Rate]);    /* change
leadingtiming rate */
```

The function Tmrb_Calculator :

```
    uint16_t Tmrb_Calculator(uint16_t Tmrb_Require_us, uint32_t ClkDiv)
    {
        uint32_t T0 = 0U;
        const uint16_t Div[8U] = {1U, 2U, 8U, 32U, 64U, 128U, 256U, 512U};

        SystemCoreClockUpdate();

        T0 = SystemCoreClock / (1U << ((TSB_CG->SYSCR >> 8U) & 7U));
        T0 = T0/((Div[ClkDiv])*1000000U);

        return(Tmrb_Require_us * T0);
    }
```

7-17 SIO/UART

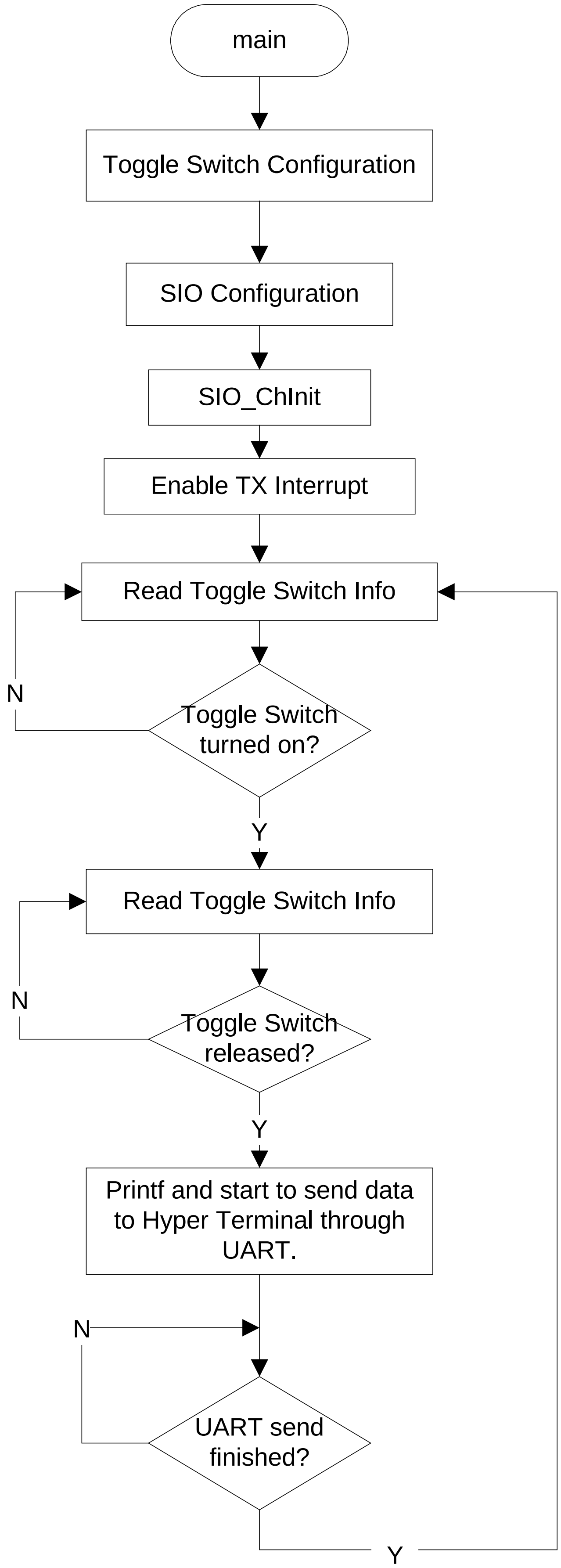
7-17-1 Example: UART

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 UART TX interrupt to send data.
4. Retarget printf() to UART.

- **Flowchart**


main
Toggle Switch Configuration
SIO Configuration
SIO_ChInit
Enable TX Interrupt
Read Toggle Switch Info
Toggle Switch turned on?
N
Y
Read Toggle Switch Info
Toggle Switch released?
N
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.

```
    GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_2, ENABLE);
    GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_2, DISABLE);
    GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_2);
```

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

```
    UART_InitTypeDef myUART;

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

After above setting, enable UART TX interrupt.

```
    NVIC_EnableIRQ(RETARGET_INT)
```

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

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

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

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

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

```
#if defined ( __CC_ARM     )          /* RealView Compiler */
struct __FILE {
    int handle;                       /* Add whatever you need here */
```

```
};
FILE __stdout;
FILE __stdin;
int fputc(int ch, FILE * f)
#elif defined ( __ICCARM__ )       /*IAR Compiler */
int putchar(int ch)
#endif
{
    return (send_char(ch));
}

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

    return ch;
}
```

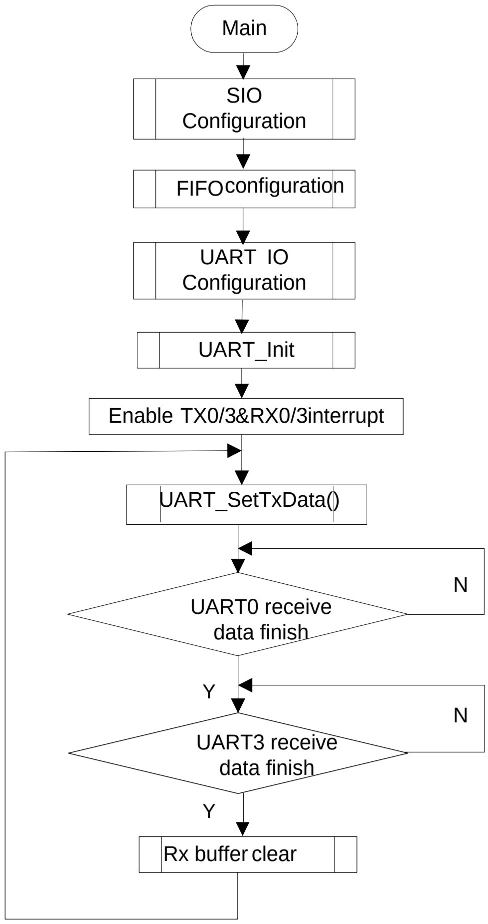
7-17-2 Example: UART FIFO

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) {
		GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_2, ENABLE);
		GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_2, DISABLE);
		GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_2);
		GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_1, DISABLE);
		GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_1, ENABLE);
		GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_1);
	} else if (SCx == TSB_SC1) {
		GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_5, ENABLE);
		GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_5, DISABLE);
		GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_5);
		GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_6, DISABLE);
		GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_6, ENABLE);
		GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_6);
	} else if (SCx == TSB_SC2) {
		GPIO_SetOutputEnableReg(GPIO_PL, GPIO_BIT_2, ENABLE);
		GPIO_SetInputEnableReg(GPIO_PL, GPIO_BIT_2, DISABLE);
		GPIO_EnableFuncReg(GPIO_PL, GPIO_FUNC_REG_5, GPIO_BIT_2);
```

```
            GPIO_SetOutputEnableReg(GPIO_PL, GPIO_BIT_1, DISABLE);
            GPIO_SetInputEnableReg(GPIO_PL, GPIO_BIT_1, ENABLE);
            GPIO_EnableFuncReg(GPIO_PL, GPIO_FUNC_REG_5, GPIO_BIT_1);
      } else if (SCx == TSB_SC3) {
            GPIO_SetOutputEnableReg(GPIO_PB, GPIO_BIT_0, ENABLE);
            GPIO_SetInputEnableReg(GPIO_PB, GPIO_BIT_0, DISABLE);
            GPIO_EnableFuncReg(GPIO_PB, GPIO_FUNC_REG_3, GPIO_BIT_0);
            GPIO_SetOutputEnableReg(GPIO_PB, GPIO_BIT_1, DISABLE);
            GPIO_SetInputEnableReg(GPIO_PB, GPIO_BIT_1, ENABLE);
            GPIO_EnableFuncReg(GPIO_PB, GPIO_FUNC_REG_3, 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|UART_ENABLE_RX;
      myUART.FlowCtrl = UART_NONE_FLOW_CTRL;
      UART_Init(UART_RETARGET, &myUART);
```

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

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

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

FIFO configuration.

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

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

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

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

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

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

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

```
    UART_RxFIFOClear(UART0);
    UART_RxFIFOClear(UART3);

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

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

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

After above setting, enable UART0/3 interrupt.

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

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

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

UART0 TX interrupt routine:

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

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

UART3 TX interrupt routine:

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

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

UART0 RX interrupt routine:

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

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

UART3 RX interrupt routine:

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

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

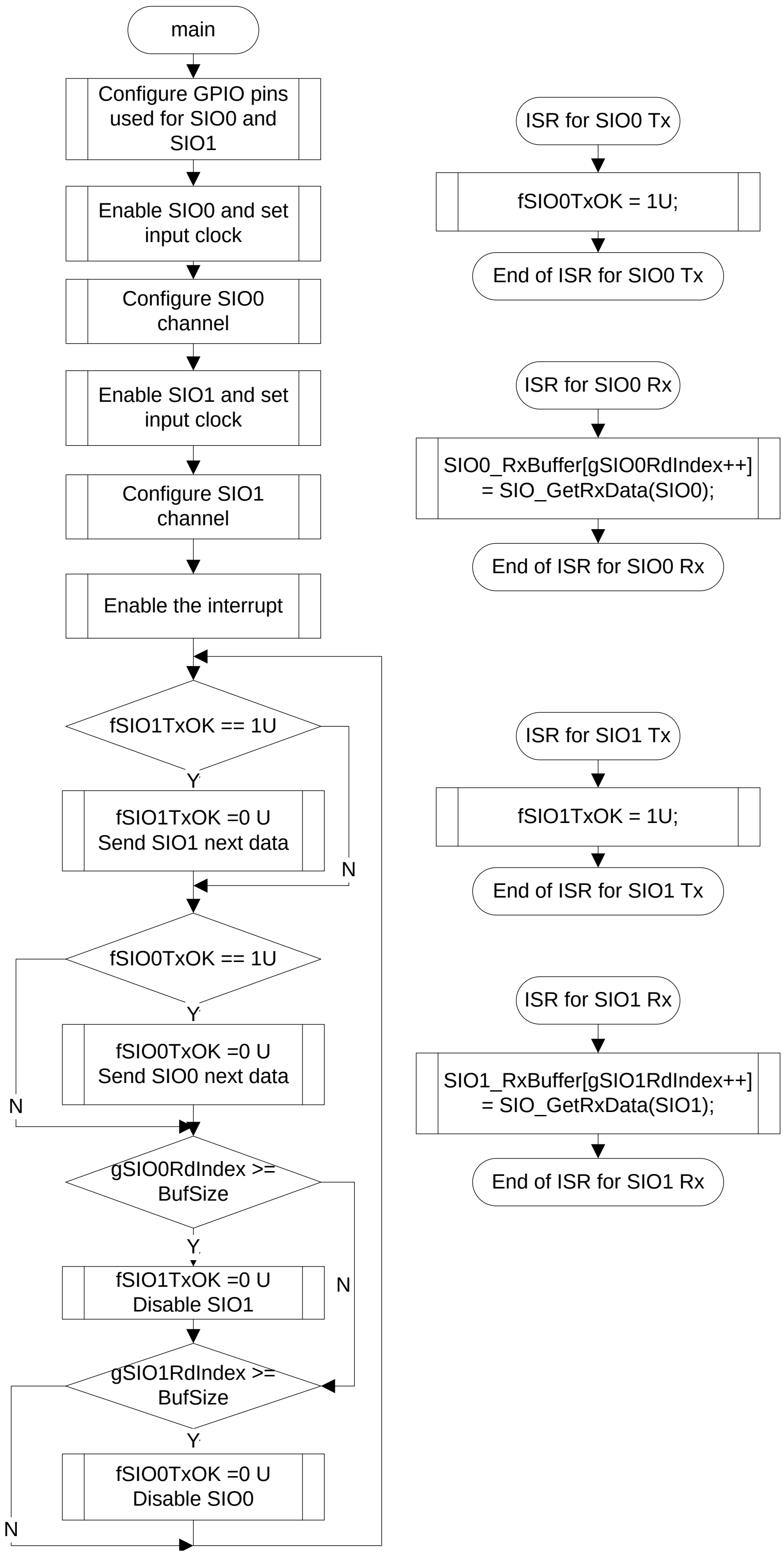
7-17-3 Example: SIO

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);

/*initialize the SIO0 struct */
SIO0_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO0_Init.TIDLE = SIO_TIDLE_HIGH;
SIO0_Init.IntervalTime = SIO_SINT_TIME_SCLK_8;
SIO0_Init.TransferMode = SIO_TRANSFER_FULLDPX;
SIO0_Init.TransferDir = SIO_LSB_FRIST;
SIO0_Init.Mode = SIO_ENABLE_TX | SIO_ENABLE_RX;
SIO0_Init.DoubleBuffer = SIO_WBUF_ENABLE;
SIO0_Init.BaudRateClock = SIO_BR_CLOCK_TS2;
SIO0_Init.Divider = SIO_BR_DIVIDER_2;

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

Enable SIO1 configure the input clock and SIO1 initialization structure.

```
/*Enable the SIO1 channel */
SIO_Enable(SIO1);

/*initialize the SIO1 struct */
SIO1_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO1_Init.TIDLE = SIO_TIDLE_HIGH;
SIO1_Init.TransferMode = SIO_TRANSFER_FULLDPX;
SIO1_Init.TransferDir = SIO_LSB_FRIST;
SIO1_Init.Mode = SIO_ENABLE_TX | SIO_ENABLE_RX;
SIO1_Init.DoubleBuffer = SIO_WBUF_ENABLE;
SIO1_Init.TXDEMP = SIO_TXDEMP_HIGH;
SIO1_Init.EHOLDTime = SIO_EHOLD_FC_64;

SIO_Init(SIO1, SIO_CLK_SCLKINPUT, &SIO1_Init);
```

Enable the interrupt of SIO TX, RX.

```
/* Enable SIO0 Channel TX interrupt */
NVIC_EnableIRQ(INTTX0_IRQn);
/* Enable SIO1 Channel RX interrupt */
NVIC_EnableIRQ(INTRX1_IRQn);

/* Enable SIO1 Channel TX interrupt */
NVIC_EnableIRQ(INTTX1_IRQn);
/* Enable SIO0 Channel RX interrupt */
NVIC_EnableIRQ(INTRX0_IRQn);
```

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

```
while (1) {
        /* SIO1 send data from TXD1*/
        if (fSIO1TxOK == 1U) {
            fSIO1TxOK = 0U;
            SIO_SetTxData(SIO1, SIO1_TxBuffer[gSIO1WrIndex++]);
        } else {
            /*Do Nothing */
        }
        /* SIO0 send data from TXD0*/
        if (fSIO0TxOK == 1U) {
            fSIO0TxOK = 0U;
```

```
                SIO_SetTxData(SIO0, SIO0_TxBuffer[gSIO0WrIndex++]);
            } else {
                /*Do Nothing */
            }

            /*SIO0 receive data end */
            if (gSIO0RdIndex >= BufSize) {
                fSIO1TxOK = 0U;
                SIO_Disable(SIO1);
            } else {
                /*Do Nothing */
            }
            /*SIO1 receive data end */
            if (gSIO1RdIndex >= BufSize) {
                fSIO0TxOK = 0U;
                SIO_Disable(SIO0);
            } else {
                /*Do Nothing */
            }

            /* Print receive buffer */
            if ((gSIO0RdIndex == BufSize) && (gSIO1RdIndex == BufSize)) {
    #ifdef DEBUG
                printf((char *)SIO0_RxBuffer);
                printf((char *)SIO1_RxBuffer);
    #endif
        }
```

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);
}
```

7-18 uDMAC

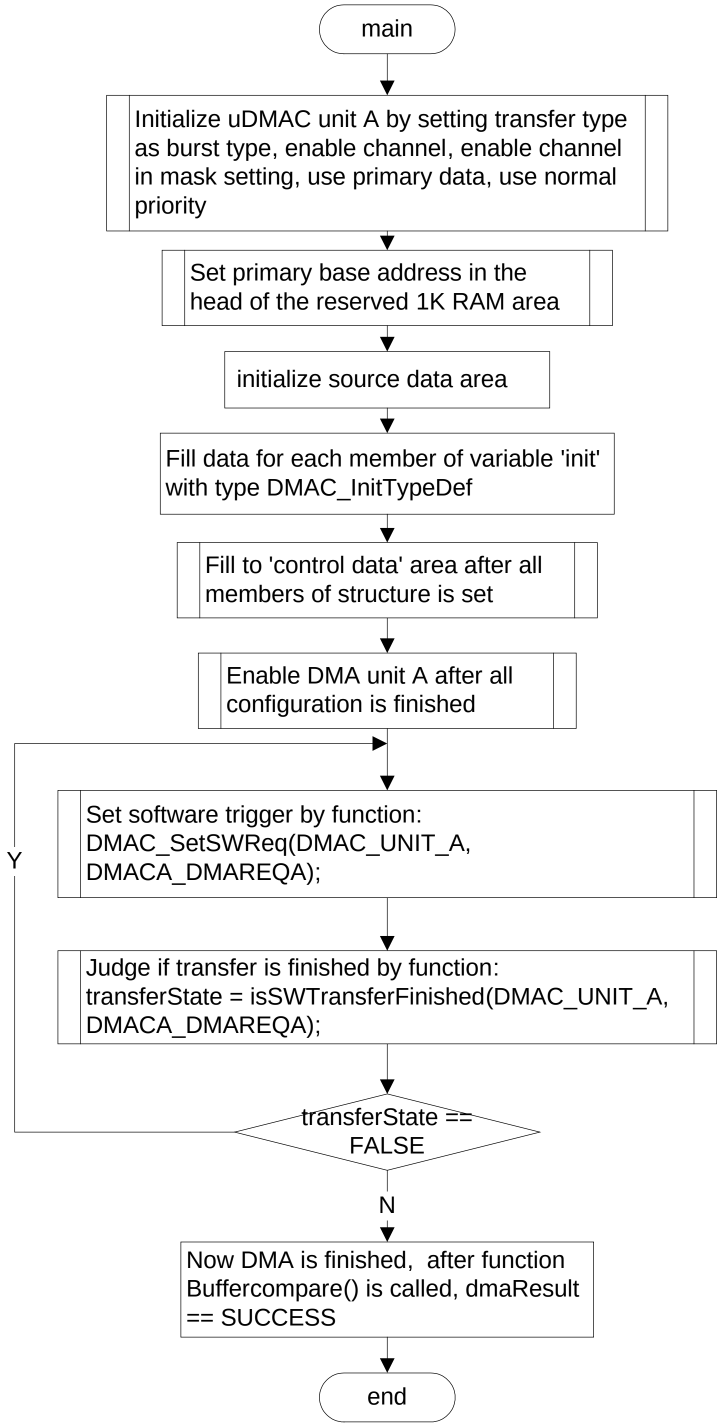
7-18-1 Example: DMA transfer from memory to memory

This is a simple example based on the TX04 Peripheral Driver (uDMAC, GPIO).
***Note:** LED0~LED3 connect with PB0~PB3 in TMPM46B MCU board.

The example includes:

1. Reserve 1K RAM area for uDMAC control data.
2. uDMAC configuration and initialization.
3. Start DMA transfer between memory by software trigger.

- **Flowchart:**



- **Code and Explanation for the Example**

Since the uDMAC needs 1K bytes RAM area to save its configuration data, this area must be reserved in the application code, with bit0 to bit9 of its base address be 0, for example, 0x20000400 is OK but 0x20000300 cann't be used.

The example to reserve RAM area in IAR EWARM and Keil MDK(RealView) is showed as below:

```
#if defined (   __ICCARM__   )      /* IAR EWARM */
/* For TMPM46BF10 uDMAC_CFG_A/B are defined in file
```

```
TMPM46BF10_Flash_For_uDMAC.icf */
uint32_t uDMAC_A_Control_Data[256U] @ ".uDMAC_CFG_A" ;
uint32_t uDMAC_B_Control_Data[256U] @ ".uDMAC_CFG_B" ;

#elif defined ( __CC_ARM   )       /* Keil MDK */
#include <absacc.h>
#define uDMAC_CFG_A         (0x20000400U)
#define uDMAC_CFG_B         (uDMAC_CFG_A + 0x400U)
uint32_t uDMAC_A_Control_Data[256U] __at (uDMAC_CFG_A);
uint32_t uDMAC_B_Control_Data[256U] __at (uDMAC_CFG_B);
#endif
```

For Keil MDK, the keyword '__at' is used and is enough but for IAR EWARM, the link configuration file (.lcf) file must be modified with content below(for UNIT A only). For more detail, please read file "TMPM46B_Flash_For_uDMAC.icf".

```
    /* Reserve 1K RAM for uDMAC configuration */
    define symbol uDMAC_RAM_START_A = 0x20000400;
    define symbol uDMAC_RAM_END_A     = 0x200007FF;

    define region uDMAC_CFG_RAM_A   = mem:[from uDMAC_RAM_START_A
to uDMAC_RAM_END_A];

    place in   uDMAC_CFG_RAM_A     { readwrite section .uDMAC_CFG_A };
```

After reserved the RAM, set transfer type as burst type, enable channel, enable channel in mask setting, use primary data, use normal priority.

```
        DMACA_SetTransferType(DMACA_DMAREQA, DMAC_BURST);
        DMAC_SetChannel(DMAC_UNIT_A, DMACA_DMAREQA, ENABLE);
        DMAC_SetMask(DMAC_UNIT_A, DMACA_DMAREQA, ENABLE);
        DMAC_SetPrimaryAlt(DMAC_UNIT_A,                DMACA_DMAREQA,
DMAC_PRIMARY);
        DMAC_SetChannelPriority(DMAC_UNIT_A, DMACA_DMAREQA,
                                DMAC_PRIOTIRY_NORMAL);
```

Then set primary base address in the head of the reserved 1K RAM area:

```
        DMAC_SetPrimaryBaseAddr(DMAC_UNIT_A,
                                (uint32_t)&uDMAC_A_Control_Data);
```

Initialize the source data area to be transfered

```
        for(idx = 0U; idx < TX_NUMBERS; idx ++ ) {
            src[idx] = idx;
        }
```

Now start setting the members of variable 'init' which is "DMAC_InitTypeDef" type.
Set the end address of source and destination

```
        tmpAddr = (uint32_t)&src;
        init.SrcEndPointer = tmpAddr + ((TX_NUMBERS - 1U) * sizeof(src[0U]));
        tmpAddr = (uint32_t)&dst;
        init.DstEndPointer = tmpAddr + ((TX_NUMBERS - 1U) * sizeof(dst[0U])) ;
```

Select use BASIC or AUTOMATIC mode

```
#if defined(DMA_DEMOMODE_BASIC )
init.Mode = DMAC_BASIC;
#elif defined(DMA_DEMOMODE_AUTOMATIC)
init.Mode = DMAC_AUTOMATIC;
#endif
```

Set other memebers

```
init.NextUseBurst = DMAC_NEXT_NOT_USE_BURST;
init.TxNum = TX_NUMBERS;
init.ArbitrationMoment = DMAC_AFTER_32_TX;

/* now both src and dst are use uint16_t type which is 2bytes long */
init.SrcWidth = DMAC_HALF_WORD;
init.SrcInc = DMAC_INC_2B;
init.DstWidth = DMAC_HALF_WORD;
init.DstInc = DMAC_INC_2B;
```

Fill to 'control data' area after all members of structure is set

```
DMAC_FillInitData(DMAC_UNIT_A, DMACA_DMAREQA, &init);
```

Enable DMA unit A after all configuration is finished

```
DMAC_Enable(DMAC_UNIT_A);
```

Set software trigger, and judge if transfer is finished.

```
do{
    /* Because of "init.Mode = DMAC_BASIC" above, here need to trigger it until transfer is finished, */
    /* If DMAC_AUTOMATIC is used, only need to trigger it once */
    DMAC_SetSWReq(DMAC_UNIT_A, DMACA_DMAREQA);

    transferState = isSWTransferFinished(DMAC_UNIT_A, DMACA_DMAREQA);

}while ( transferState == false );
```

When transfer is finished, call function Buffercompare() to judge if dmaResult is SUCCESS.

```
dmaResult = ERROR;
dmaResult = Buffercompare( src, dst, TX_NUMBERS);
if ( dmaResult   == SUCCESS ) {
    LED_On(LED0);
    LED_On(LED1);
} else {
    LED_On(LED2);
    LED_On(LED3);
}
```

7-19 WDT

7-19-1 Example:WDT

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. LED1 will blink once
3. In DEMO2 WDT is cleared before timer overflow, the LED0 will blink all the time.

- **Code and Explanation for the Example**

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

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

Initialize WDT and enable it.

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

In DEMO1 Waiting for the NMI interrupt flag.

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

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

```
WDT_Disable();
```

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

```
WDT_WriteClearCode();
```

***Note**:
LED0 connect with PB0 in TMPM46B MCU board.
LED1 connect with PB1 in TMPM46B MCU board.