TOSHIBA

TX03 Peripheral Driver Usage Example (TMPM311)

Rev 1.000
Sep, 2017

TOSHIBA ELECTRONIC DEVICES & STORAGE CORPORATION

CMDR-M311UE-00xE

RESTRICTIONS ON PRODUCT USE

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

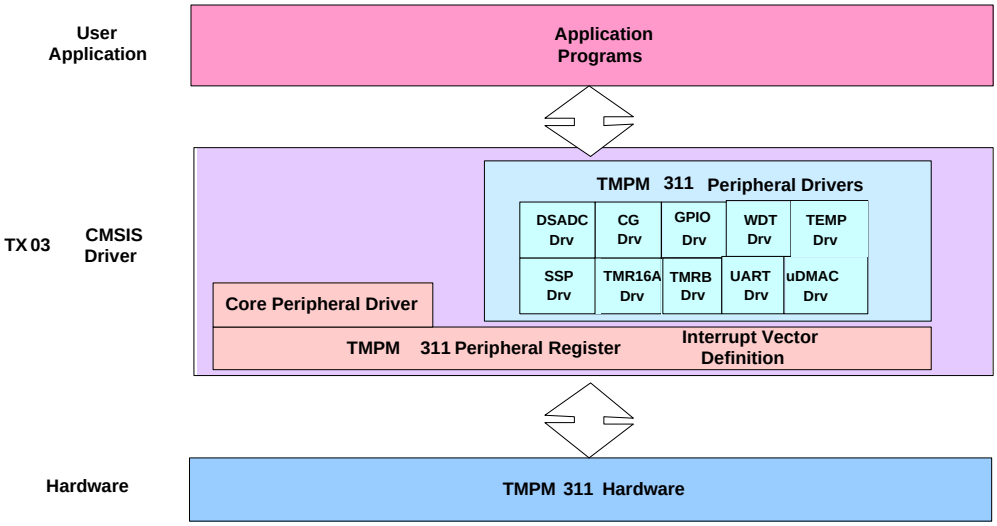
Index

1 General description

The example programs described hereafter are specially designed for TOSHIBA TMPM311CHDUG MCU. The programs can be executed individually each to show the main MCU functions. Users can utilize some portions of the programs and integrate them into users' own programs to perform customized functions.

2 Overview

User application utilizes TX03 peripheral driver as following.



3 Build-in hardware usage

Hardware	Channel	Use presence, use
CG	Clock Gear	Used for CG demo
SysTick	-	Unused
Watch dog timer (WDT)	WDT0	Used for WDT demo
Interrupt (INT)	INT0	Unused
	INT1	Unused
SIO	SIO0	Used for UART retarget demo/ SIO demo/ UART FIFO demo
16-bit Timer / Event Counters	TMRB0	Used for TMRB: General Timer, PPG Output
	TMRB1	Unused
	TMRB2	Unused

Hardware	Channel	Use presence, use
	TMRB3	Unused
16-bit timerA	TMR16A0	Used for TMR16A: General Timer
DSADC	UnitA	Used for DSADC demo
	UnitB	Used for DSADC demo
	UnitC	Used for DSADC demo
	UnitD	Used for DSADC demo
uDMAC	-	Used for DMAC demo
SSP	SSP0	Used for SSP0 self loopback demo
TEMP	-	Used for TEMP demo

4 Pin Usage

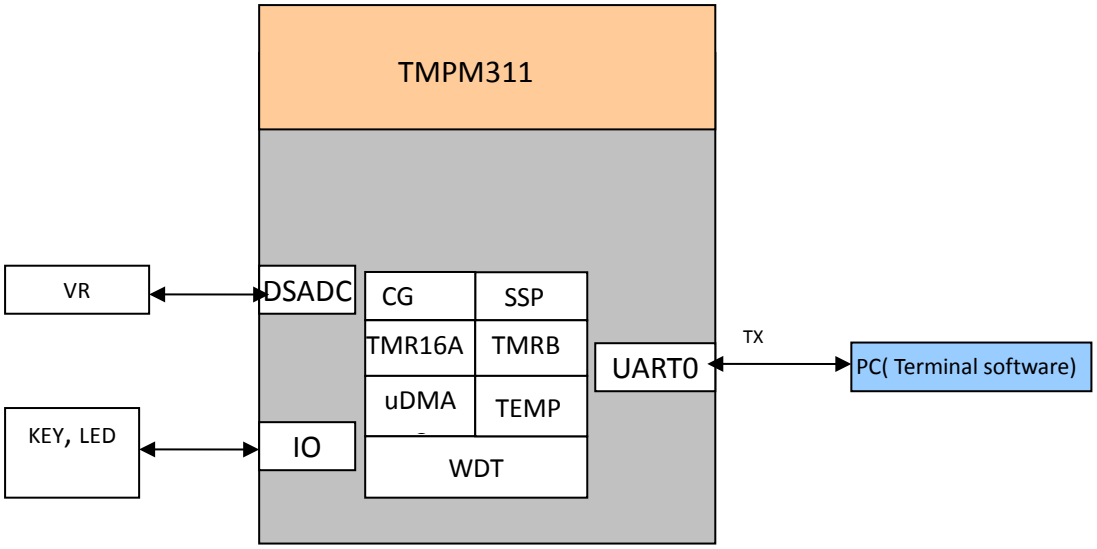
The example programs are tested on TMPM311CHDUG evaluation board.
Following is pin usage for example programs

Pin No.	Name	Usage
1	PA0	SP0CLK
2	PA1	SP0DO
3	PA2	SP0DI
4	PA3	SP0FSS
10	PA4	SC0CLK
11	PA5	SC0RXD
12	PA6	SC0TXD
14	PB0	LED0
15	PB1	TB0OUT / LED1
16	PB2	LED2
17	PB3	LED3
18	PB4	SW0
19	PB5	SW1
20	PB6	SW2
21	PB7	SW3
22	AGNDREFA	Connect DSAINA to Gnd for demos of DSADC
23	DAINA+/ DAINA-	DSAINA voltage input for demos of DSADC
25	VREFINA	Connect to pin '+' of a 1uF capacitor for demos of DSADC
26	AGNDREFB	Connect DSAINB to Gnd for demos of DSADC
27	DAINB+/ DAINB-	DSAINB voltage input for demos of DSADC
29	VREFINB	Connect to pin '+' of a 1uF capacitor for demos of DSADC
30	AGNDREFC	Connect DSAINC to Gnd for demos of DSADC

31	DAINC+/ DAINC-	DSAINC voltage input for demos of DSADC
33	VREFINC	Connect to pin '+' of a 1uF capacitor for demos of DSADC
34	AGNDREFD	Connect DSAIND to Gnd for demos of DSADC and TEMP
35	DAIND+/ DAIND-	DSAIND voltage input for demos of DSADC
37	VREFIND	Connect to pin '+' of a 1uF capacitor for demos of DSADC and TEMP
38	DSRVDD3	Connect to Vcc for demos of DSADC and TEMP
39	SRVDD	Connect to Vcc for demos of DSADC and TEMP
40	DSRVSS	Connect to Gnd for demos of DSADC and TEMP
6	PD0	High frequency resonator connection pin
8	PD1	High frequency resonator connection pin
41	MODE	MODE pin
43	RESET	Reset signal input pin

5 Development Environment

Following is development environment:



1. Hardware board:
   TMPM311CHDUG evaluation board

2. Development tool:
   - IAR:
     1) J-Link: IAR J-Link-ARM 7.0 / J-Link 6.0
     2) IDE: IAR Embedded workbench 7.10.3 version
   - KEIL:
     1) IDE:KEIL uVision 5.10

6 Functional description

6-1 Operation mode

There is only one operation mode for TMPM311: NORMAL 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.

6-2 DSADC

This is a simple example to show DSADC operation.

It reads four units DSADC value by UART communication. It runs on board "PM-EBD007-068A" for TMPM311CHDUG.

***Note**:

Pins connection:

1. Voltage input
    (a) Connect pin 23 (DAINA+) / pin 24 (DAINA-) to voltage (-0.375V < voltage input < +0.375V)
    (b) Connect pin 27 (DAINB+) / pin 28 (DAINB-) to voltage (-0.375V < voltage input < +0.375V)
    (c) Connect pin 31 (DAINC+) / pin 32 (DAINC-) to voltage (-0.375V < voltage input < +0.375V)
    (d) Connect pin 35 (DAIND+) / pin 36 (DAIND-) to voltage (-0.375V < voltage input < +0.375V)
2. Connect pin 22 (AGNDREFA), pin 26 (AGNDREFB), pin 30 (AGNDREFC) and pin34 (AGNDREFD) to Gnd.
3. Connect a 1μF capacitor to between VREFINx and AGNDREFx.
    (a) Connect pin 25 (VREFINA) to pin '+' of a 1uF capacitor, the '-' pin of capacitor is connected to pin 22 (AGNDREFA).
    (b) Connect pin 29 (VREFINB) to pin '+' of a 1uF capacitor, the '-' pin of capacitor is connected to pin 26 (AGNDREFB).
    (c) Connect pin 33 (VREFINC) to pin '+' of a 1uF capacitor, the '-' pin of capacitor is connected to pin 30 (AGNDREFC).
    (d) Connect pin 37 (VREFIND) to pin '+' of a 1uF capacitor, the '-' pin of capacitor is connected to pin 34 (AGNDREFD).
4. Connect pin 38 (DSRVDD3) and pin 39 (SRVDD) to Vcc.

5. Connect pin 40 (DSRVSS) to Gnd.

6-3 GPIO

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

6-4 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
```

6-5 TEMP

This is a simple example to show the MCU measures a relative temperature using a temperature sensor.
A temperature sensor outputs a voltage based on the reference voltage circuit (BGR) according to temperatures. The output voltage is input to unit D of DSADC. With AD conversion, a corresponding digital value to temperatures is obtained.
It reads DSADC unit D value by UART communication. It runs on board "PM-EBD007-068A" for TMPM311CHDUG.

***Note**:
Pins connection:
1. Voltage input
    (a) Internal Analog Input from temperature sensor
2. Connect pin34 (AGNDREFD) to Gnd.
3. Connect a 1μF capacitor to between VREFINx and AGNDREFx.
    (a) Connect pin 37 (VREFIND) to pin '+' of a 1uF capacitor, the '-' pin of capacitor is connected to pin 34 (AGNDREFD).
4. Connect pin 38 (DSRVDD3) and pin 39 (SRVDD) to Vcc.
5. Connect pin 40 (DSRVSS) to Gnd.

6-6 TMR16A

Set a value of T16A0RG<RG>, then start count-up, if the counter value matches with a value of T16A0RG<RG>, the counter will be cleared to "0x0000" and continued to count-up and the same time a match detection interrupt INTT16A0 will be output.

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

6-7 TMRB

6-7-1 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).

6-7-2 PPG Output

Use key SW0 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%

6-8 SIO/UART

6-8-1 UART_Retarget

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.
***Note1**:
This UART channel is limited by hardware, the follow example used UART0.

6-8-2 UART FIFO

In this sample Demo, Master UART0 sent the data "TMPM3111" to Slave UART0 use FIFO, and at same time Master UART0 receive the data "TMPM3112" which send from Slave UART0 and also use FIFO.
Set one breakpoint before the function ResetIdx(),when program stop in the breakpoint you can read the RxBuffer = "TMPM3112" and can read the RxBuffer1 = "TMPM3111" (connect Master SC0TXD with Slave SC0RXD, connect Master SC0RXD with Slave SC0TXD, connect Master SC0SCK with Slave SC0SCK)
***Note2**:
In this demo, slave applications must run first, and then run the host applications.

6-8-3 SIO

This demo will show synchronously transfer and receive usage of SIO module.
It use the channel Master SIO0, Slave SIO0 and transfer data synchronously between them (connect Master SC0TXD with Slave SC0RXD, connect Master SC0RXD with Slave SC0TXD, connect Master SC0SCK with Slave SC0SCK)

***Note3**:
In this demo, slave applications must run first, and then run the host applications.

6-9 uDMAC

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.

6-10 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)

7 Software

This software project creates the sample applications based on TMPM311CHDUG evaluation board which will demonstrate the main feature of TMPM311CHDUG 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
        │   tmpm311_wdt_int.c
        │   tmpm311_wdt_int.h
        │
        ├─TX03_CMSIS
        │ │ system_TMPM311.c
        │ │ system_TMPM311.h
        │ │ TMPM311.h
        │ │
        │ └─startup
                startup_TMPM311.s
        │
        ├─TX03_Periph_Driver
        │   ├─inc
        │   │   tmpm311_cg.h
        │   │   tmpm311_gpio.h
```

```
│   │   tmpm311_wdt.h
│   │   TX03_common.h
│   │
│   └─src
│         tmpm311_cg.c
│         tmpm311_gpio.c
│         tmpm311_wdt.c
│
└─TMPM311-EVAL
        led.c
        led.h
        sw.c
        sw.h
        common_uart.c
        common_uart.h
```

KEIL MDK:

```
├─WDT_NMI
    ├─APP
    │   main.c
    │   tmpm311_wdt_int.c
    │
    ├─TX03_CMSIS
    │     system_TMPM311.c
    │     startup_TMPM311.s
    │
    ├─TX03_Periph_Driver
    │     tmpm311_cg.c
    │     tmpm311_gpio.c
    │     tmpm311_wdt.c
    │
    └─TMPM311-EVAL
          led.c
          sw.c
          common_uart.c
```

7-1 DSADC

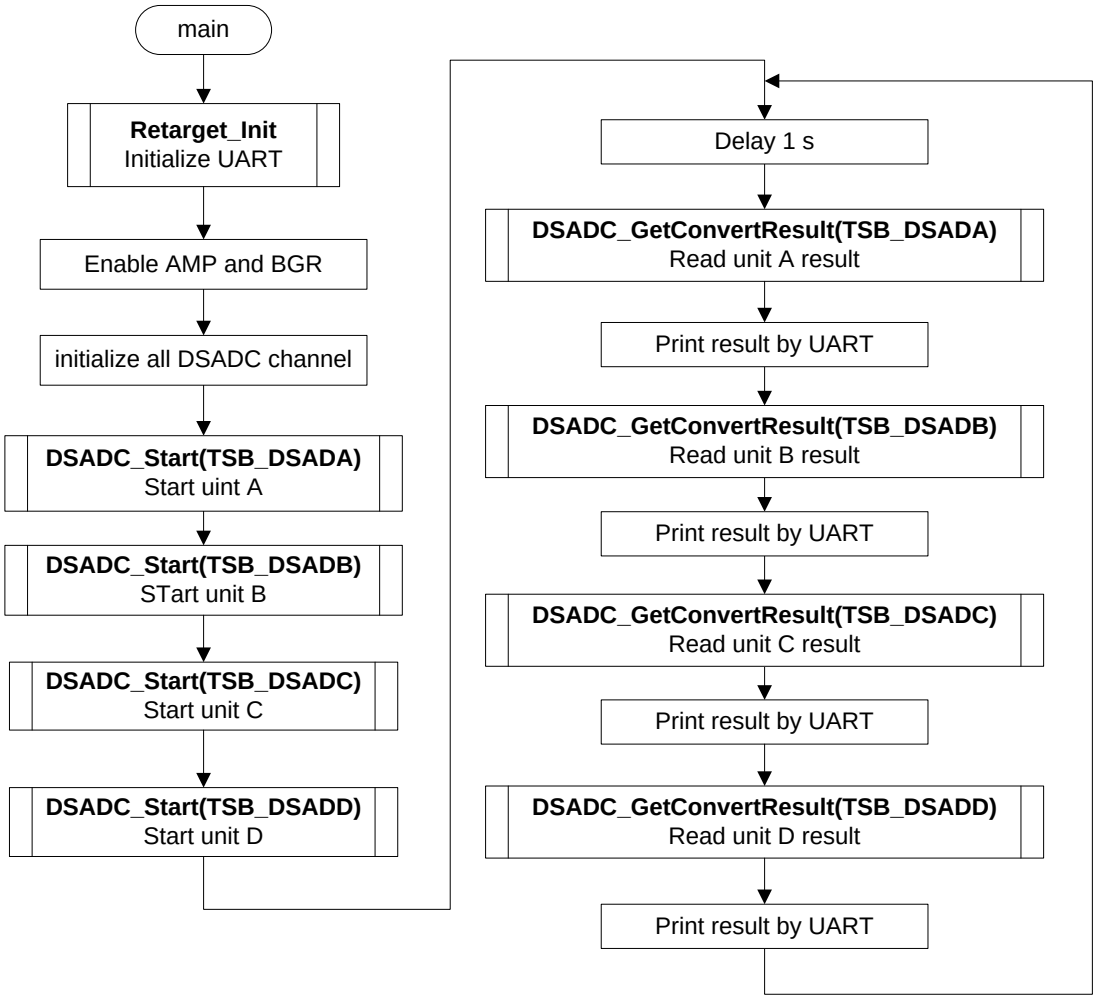
7-1-1 Example: DSADC Data Read

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

The example includes:

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

- **Flowchart:**



- **Code and Explanation for the Example**

At first, initialize UART.

```
    Retarget_Init();
```

Then enable AMP and BGR and wait 1 ms to secure the stabilization time.

```
    TMRB_SetRunState(TSB_TB0, TMRB_RUN);

     /* Enable AMP and BGR */
    TEMP_SetBGRState(ENABLE);
    TEMP_SetAMPState(ENABLE);

    while (f1ms != 1U) {
    };
```

Initialize all DSADC channel.

```
    for (i = 0U; i < (sizeof(t_DSADx) >> 2U); ++i) { /* initialize all DSADC channel */
        DSADCx = t_DSADx[i];
        DSADC_SWReset(DSADCx);

        InitStruct.Clk = DSADC_FC_DIVIDE_LEVEL_1;
        InitStruct.BiasEn = ENABLE;
        InitStruct.ModulatorEn = ENABLE;
        InitStruct.SyncMode = DSADC_SYNC_MODE; /* DSADC_SYNC_MODE
*/
        InitStruct.HardwareFactor = DSADC_HARDWARE_TRIGGER_EXT;
        InitStruct.HardwareEn = DISABLE;

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

Get the convert result and print the value by UART.

```
    while (1U) {
        if (f1s == 1U) {
            f1s = 0U;
            result = DSADC_GetConvertResult(TSB_DSADA);
            printf("DSADA: %d\r\n", result);

            result = DSADC_GetConvertResult(TSB_DSADB);
            printf("DSADB: %d\r\n", result);

            result = DSADC_GetConvertResult(TSB_DSADC);
            printf("DSADC: %d\r\n", result);

            result = DSADC_GetConvertResult(TSB_DSADD);
            printf("DSADD: %d\r\n", result);
        }
    }
```

7-2 GPIO

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

7-2-1 Example: GPIO Data Read

This is a simple example based on the TX03 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_PB, GPIO_BIT_4 | GPIO_BIT_5 | GPIO_BIT_6 |
                        GPIO_BIT_7);
```

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

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

```
if (GPIO_ReadDataBit(GPIO_PB, GPIO_BIT_4) == 1U) {
                tmp = 1U;
} else {
               /*Do nothing */
}
```

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

```
Uint8_t tmp;
tmp = GPIO_ReadData(GPIO_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);
```

7-3 SSP

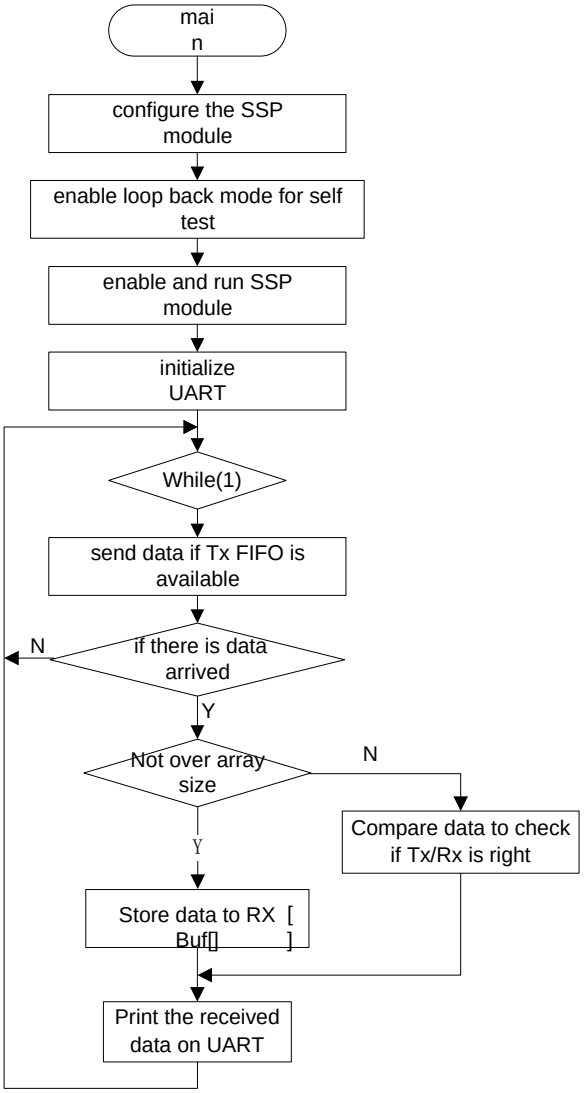
7-3-1 Example: SSP0 Self Loop Back

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

The example includes:

1. Configuration and initialization SSP module channel0
2. Enable its loop back mode to do self Tx/Rx

- **Flowchart**



- **Code and Explanation for the Example**

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

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

	uint16_t receive = 0U;
	char SSP_UART_Data[16];

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

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

```
	/* default is to run at maximum bit rate */
	initSSP.PreScale = 2U;
	initSSP.ClkRate = 1U;
	/* define BITRATE_MIN to run at minimum bit rate */
	/*	BitRate = fSYS / (PreScale x (1 + ClkRate)) */
#ifdef BITRATE_MIN
	initSSP.PreScale = 254U;
	initSSP.ClkRate = 255U;
#endif
	initSSP.ClkPolarity = SSP_POLARITY_LOW;
	initSSP.ClkPhase = SSP_PHASE_FIRST_EDGE;
	initSSP.DataSize = 16U;
	initSSP.Mode = SSP_MASTER;
	SSP_Init(TSB_SSP0, &initSSP);

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

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

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

	while (1) {

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

		/* check if there is data arrived */
		fifoState = SSP_GetFIFOState(TSB_SSP0, SSP_RX);
		if ((fifoState == SSP_FIFO_FULL) || (fifoState == SSP_FIFO_NORMAL)) {
			receive = SSP_GetRxData(TSB_SSP0);
			if (cntRx < MAX_BUFSIZE) {
				Rx_Buf[cntRx] = receive;
				cntRx++;
			} else {
				/* Place a break point here to check if receive data is right. */
				/* Success Criteria:	*/
				/* Every data transmited from Tx_Buf is received in Rx_Buf. */
				/* When the line "#define BITRATE_MIN" is commented, the SSP
```

```
                    is run in maxium */
                /* bit rate, so we can find there is enough time to transmit date
                   from 1 to    */
                /* MAX_BUFSIZE one by one. but if we uncomment that line, SSP
                     is run in          */
                /*    minimum bit rate, we will find that receive data can't catch
                     "datTx++",     */
                /*    in this so slow bit rate, when the Tx FIFO is available, the
                     cntTx has       */
                /*    been increased so much.    */
                __NOP();
                result = Buffercompare(Tx_Buf, Rx_Buf, MAX_BUFSIZE);
                if (result    == NOT_SAME)
                {
                      LED_On(LED0);
                      LED_On(LED1);
                }else
                {
                      LED_On(LED2);
                      LED_On(LED3);
                }
            }

        } else {
              /* Do nothing */
        }

        sprintf((char *)SSP_UART_Data, "%d", receive);

        common_uart_disp("SSP RX DATA:");
        common_uart_disp("\n");
        common_uart_disp(SSP_UART_Data);
        common_uart_disp("\n");

    }
}
```

7-4 TEMP

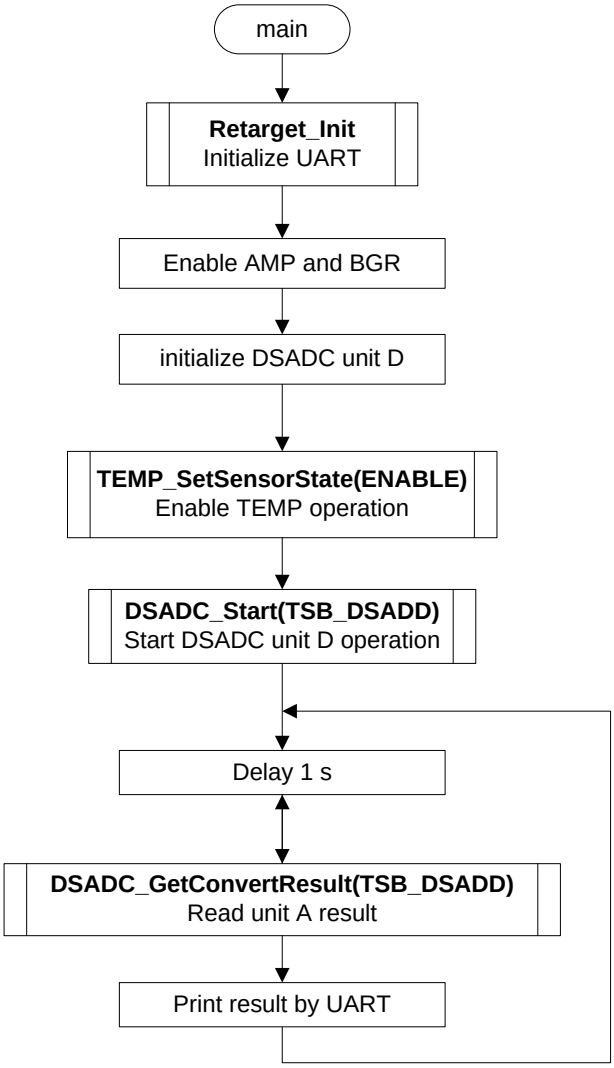
7-4-1 Example: TEMP

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

The example includes:

1. Initialize TEMP and DSADC.
2. DSADC convert TEMP internal analog input to digital value.
3. UART prints digital value

- **Flowchart:**



- **Code and Explanation for the Example**

At first, initialize UART.

```
    Retarget_Init();
```

Then enable AMP and BGR and wait 1 ms to secure the stabilization time.

```
    TMRB_SetRunState(TSB_TB0, TMRB_RUN);

     /* Enable AMP and BGR */
    TEMP_SetBGRState(ENABLE);
    TEMP_SetAMPState(ENABLE);
    /* wait 1 ms to secure the stabilization time */
    while (f1ms != 1U) {
    };
```

Initialize DSADC unit D.

```
    DSADC_SWReset(TSB_DSADD);

    InitStruct.Clk = DSADC_FC_DIVIDE_LEVEL_1;
```

```
    InitStruct.BiasEn = ENABLE;
    InitStruct.ModulatorEn = ENABLE;
InitStruct.SyncMode = DSADC_A_SYNC_MODE;    /* DSADC_A_SYNC_MODE; */
    InitStruct.HardwareFactor = DSADC_HARDWARE_TRIGGER_EXT;
    InitStruct.HardwareEn = DISABLE;

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

Get the convert result and print the value by UART.

```
    while (1U) {
        if (f1s == 1U) {
            f1s = 0U;
            result = DSADC_GetConvertResult(TSB_DSADD);
            printf("DSADD: %d\r\n", result);
        }
    }
```

7-5 TMR16A

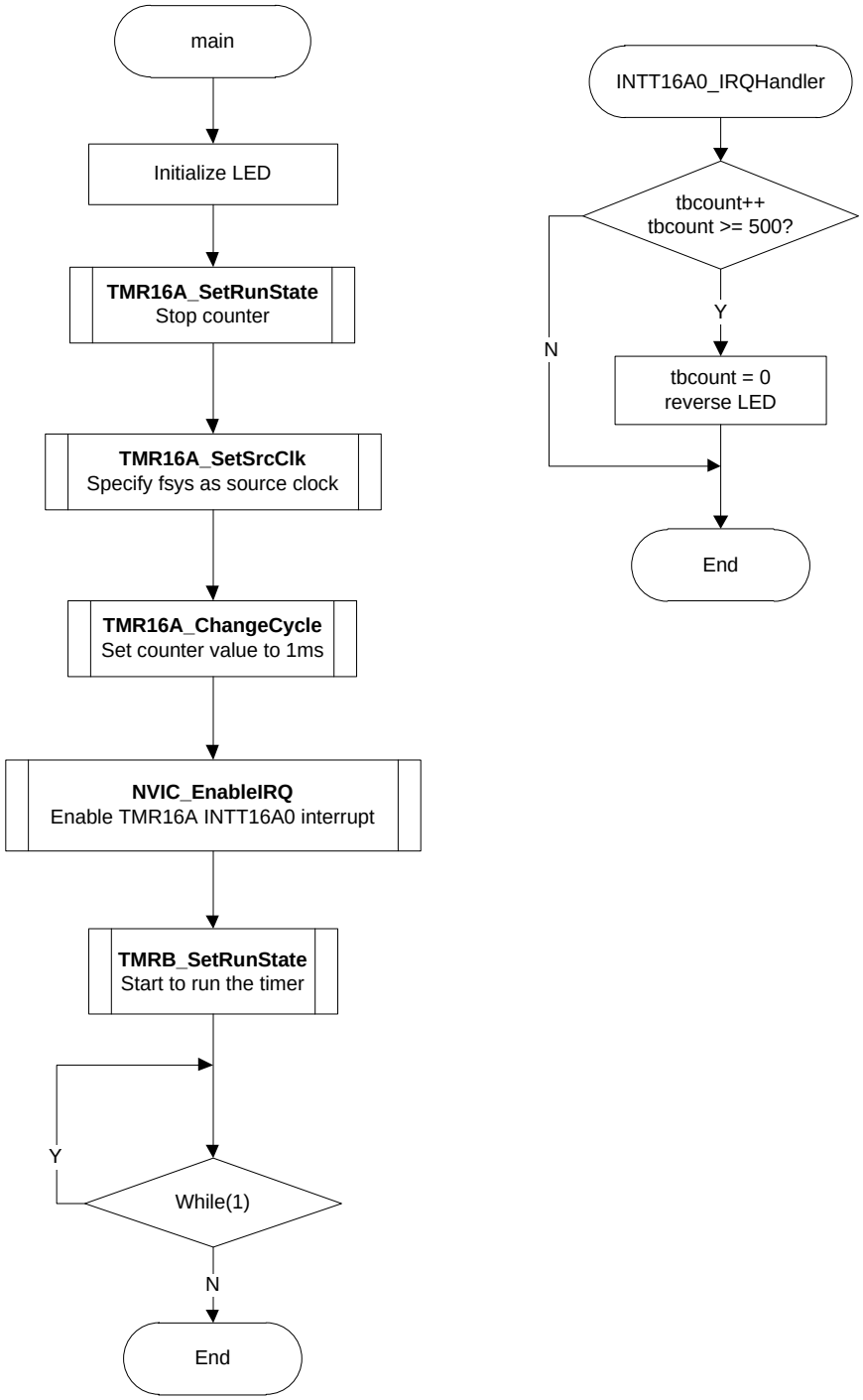
7-5-1 Example: General Timer

This is a simple example based on the TX03 Peripheral Driver (TMR16A, GPIO).

The example includes:

1. TMR16A setting
2. General Timer for 1ms

- **Flowchart**



- **Code and Explanation for the Example**

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

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

First stops the counter. This demo will set cycle to 1ms, macro TMR16A_1MS equals 0x1A40U, because ftmra = fphiT0 = fsys = fc = 8MHz, Ttmra = 1/8us, 1000*8 = 8000 = 0x1A40 (Please see CG part for more detail information about the clock setting)

Then run the counter.

```
    TMR16A_SetRunState(TSB_T16A0, TMR16A_STOP);   /* Counter stops*/
    TMR16A_SetSrcClk(TSB_T16A0, TMR16A_SYSCK);    /* Set source clock to system clock */
    TMR16A_ChangeCycle(TSB_T16A0, TMR16A_1MS);    /* Set counter value to 1ms*/
    NVIC_EnableIRQ(INTT16A0_IRQn);
    TMR16A_SetRunState(TSB_T16A0, TMR16A_RUN);
```

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

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

7-6 TMRB

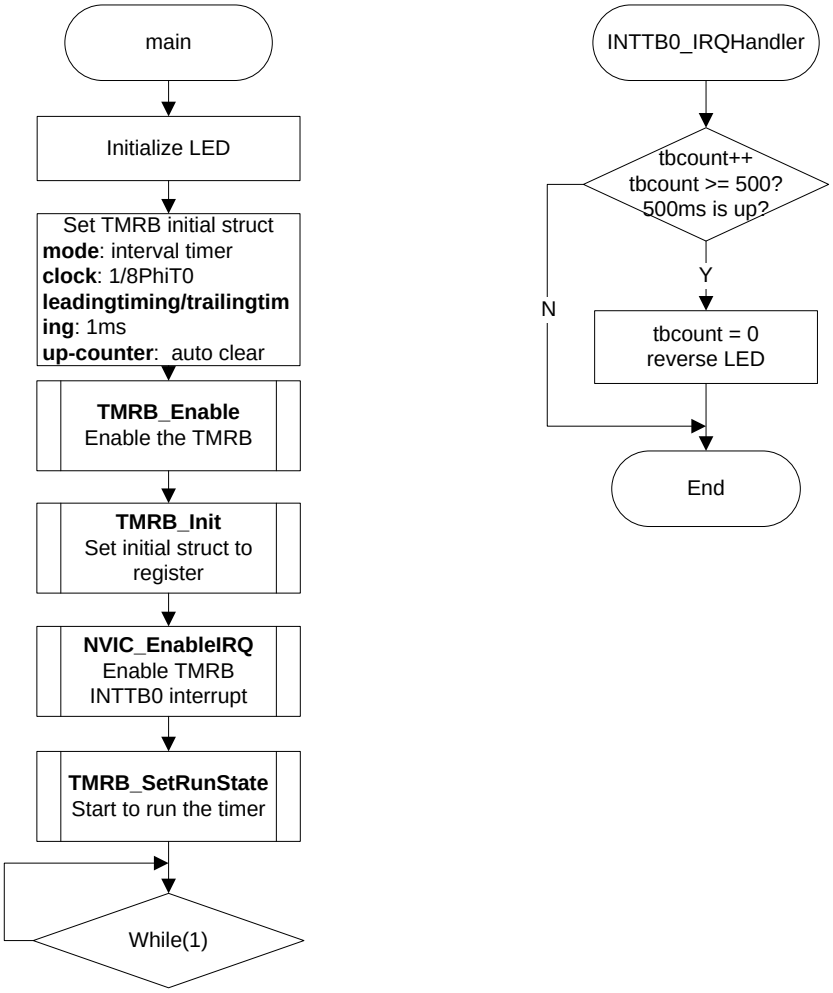
7-6-1 Example: General Timer

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

The example includes:

1. TMRB0 initialization
2. General Timer for 1ms

- **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_CG0->CLKCR >> 3U) & 7U));
    T0 = T0/((Div[ClkDiv])*1000000U);

    return(Tmrb_Require_us * T0);
}
```

7-6-2 Example: PPG Output

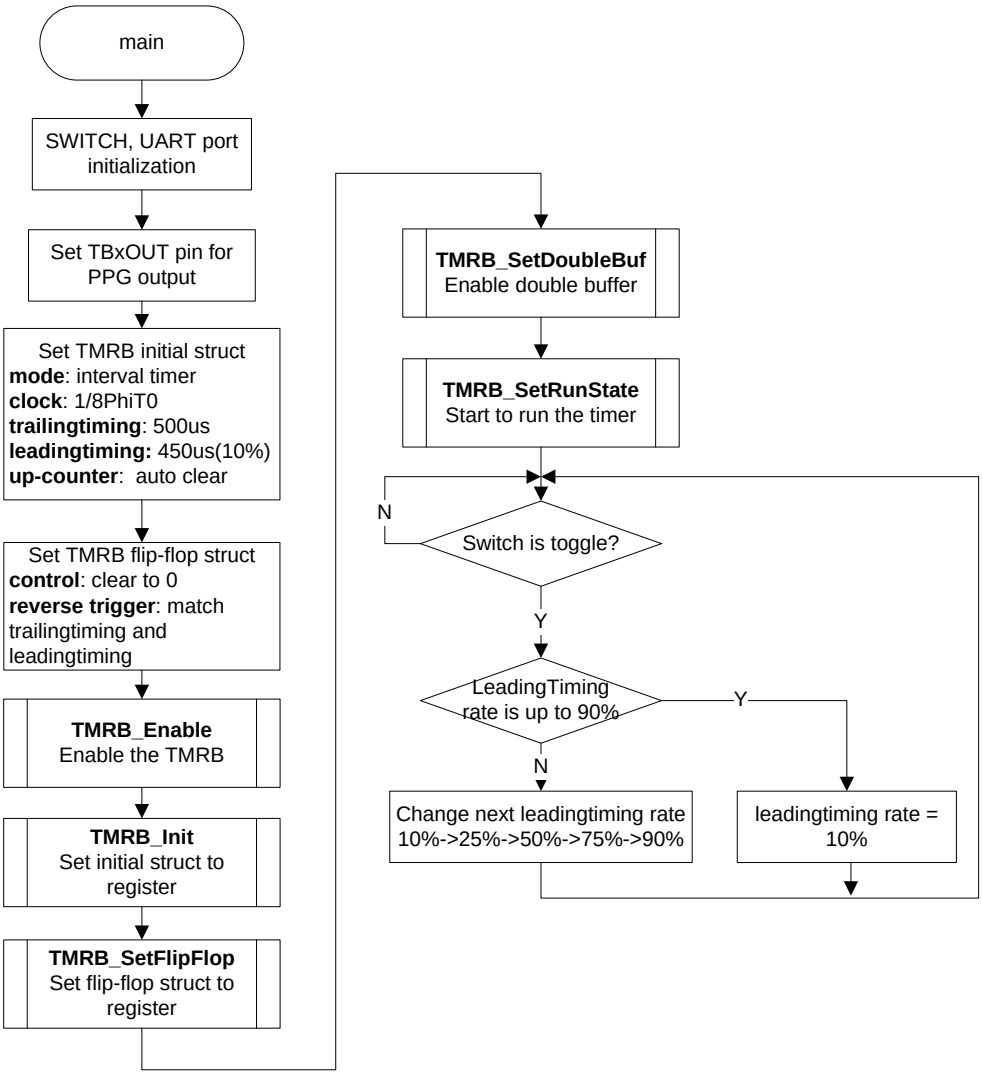
This is a simple example based on the TX03 Peripheral Driver (TMRB, GPIO, UART).

The example includes:

1. TMRB0 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 PB1 as TB0OUT 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};
    uint32_t LeadingTiming[5U] = {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 PB1 as TB0OUT for PPG output */
    GPIO_SetOutput(GPIO_PB, GPIO_BIT_1);
    GPIO_EnableFuncReg(GPIO_PB, GPIO_FUNC_REG_1, 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_TB0);
    TMRB_Init(TSB_TB0, &m_tmrb);
    TMRB_SetFlipFlop(TSB_TB0, &PPGFFInital);
    /* enable double buffer */
    TMRB_SetDoubleBuf(TSB_TB0,ENABLE, TMRB_WRITE_REG_SEPARATE);
    TMRB_SetRunState(TSB_TB0, TMRB_RUN);
```

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

```
    do {                            /* wait if switch is Low */
            keyvalue = GPIO_ReadDataBit(KEYPORT, GPIO_BIT_4);
            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_TB0,  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_CG0->CLKCR >> 3U) & 7U));
        T0 = T0/((Div[ClkDiv])*1000000U);

        return(Tmrb_Require_us * T0);
    }
```

7-7 SIO/UART

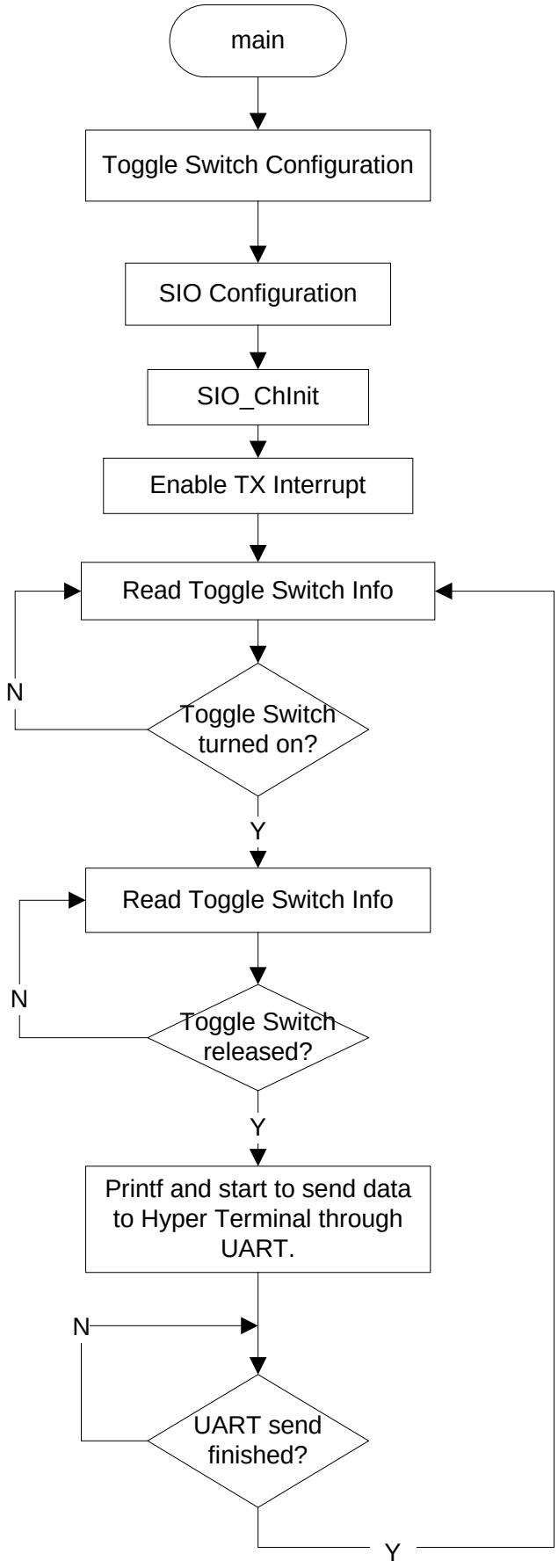
7-7-1 Example: UART_Retarget

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

The example includes:

1. UART configuration and initialization.
2. UART TX process.
3. Use 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_PA, GPIO_BIT_6, ENABLE);
GPIO_SetInputEnableReg(GPIO_PA, GPIO_BIT_6, DISABLE);
GPIO_EnableFuncReg(GPIO_PA, GPIO_FUNC_REG_1, GPIO_BIT_6);
```

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 {
```

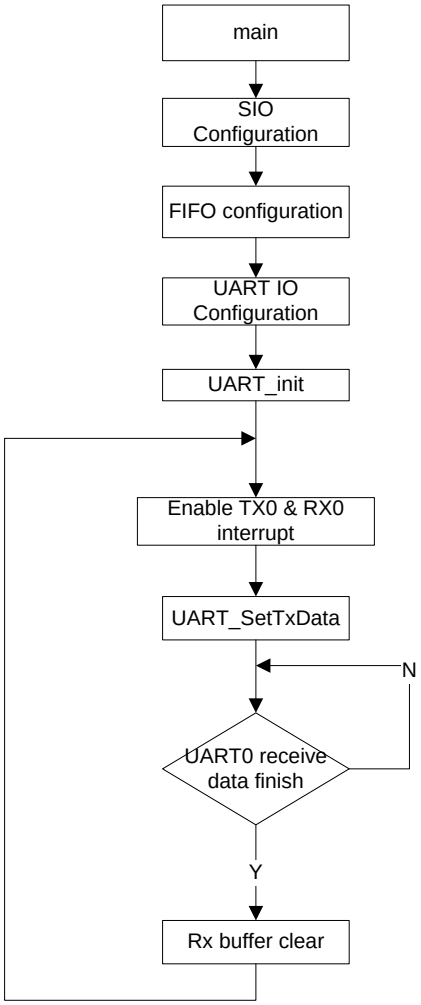
```
    int handle;                         /* Add whatever you need here */
};
FILE __stdout;
FILE __stdin;
int fputc(int ch, FILE * f)
#elif defined ( __ICCARM__ )       /*IAR Compiler */
int putchar(int ch)
#endif
{
    return (send_char(ch));
}

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;
}
```

7-7-2 Example: UART FIFO Master

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

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

- **Flowchart**



- **Code and Explanation for the Example**

At first configure the GPIO and initialize UART.

Use GPIO peripheral drivers configure GPIO for Master UART0

```
void SIO_Configuration(TSB_SC_TypeDef * SCx)
{
        if (SCx == TSB_SC0) {
            TSB_PA->CR |= GPIO_BIT_6;
            TSB_PA->FR1 |= GPIO_BIT_6;
            TSB_PA->FR1 |= GPIO_BIT_5;
            TSB_PA->IE |= GPIO_BIT_5;
        } else {
            /* nothing */
        }
}
```

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

```
	UART_InitTypeDef myUART;

	/* configure SIO0 for reception */
	UART_Enable(UART_0);
	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 Master UART0 channels.

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

FIFO configuration.

```
	UART_RxFIFOByteSel(UART0,UART_RXFIFO_RXFLEVEL);
	UART_TxFIFOINTCtrl(UART0,ENABLE);
	UART_RxFIFOINTCtrl(UART0,ENABLE);
	UART_TRxAutoDisable(UART0,UART_RXTXCNT_AUTODISABLE);
	UART_FIFOConfig(UART0,ENABLE);
	UART_RxFIFOFillLevel(UART0, UART_RXFIFO4B_FLEVLE_4_2B);
	UART_RxFIFOINTSel(UART0,UART_RFIS_REACH_EXCEED_FLEVEL);
	UART_RxFIFOClear(UART0);
	UART_TxFIFOFillLevel(UART0, UART_TXFIFO4B_FLEVLE_0_0B);
	UART_TxFIFOINTSel(UART0,UART_TFIS_REACH_NOREACH_FLEVEL);
	UART_TxFIFOClear(UART0);
```

After above setting, enable UART0 interrupt.

```
	NVIC_EnableIRQ(INTTX0_IRQn);
	NVIC_EnableIRQ(INTRX0_IRQn);
```

The rest process of data flow is finished in ISR of UART0 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);
	}
}
```

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);
```

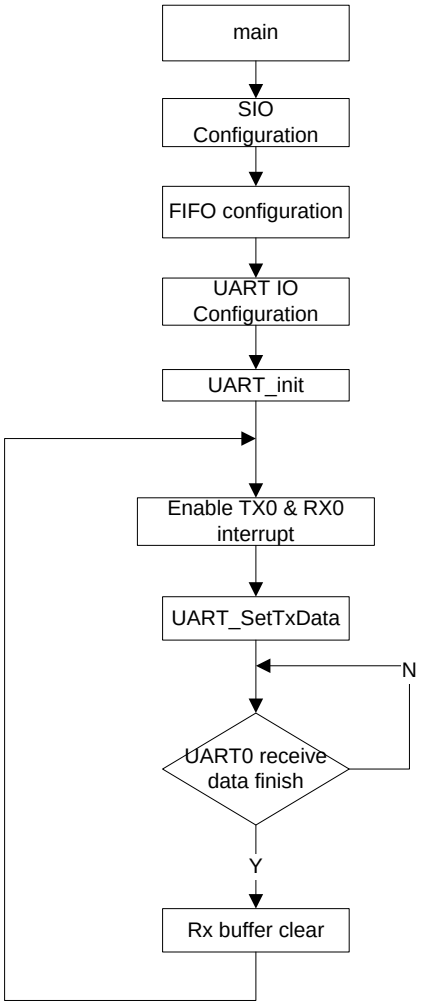
```
    }
}
```

7-7-3 Example: UART FIFO Slave

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

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

- **Flowchart**



- **Code and Explanation for the Example**

At first configure the GPIO and initialize UART.

Use GPIO peripheral drivers configure GPIO for Slave UART0

```
void SIO_Configuration(TSB_SC_TypeDef * SCx)
{
       if (SCx == TSB_SC0) {
            TSB_PA->CR |= GPIO_BIT_6;
            TSB_PA->FR1 |= GPIO_BIT_6;
            TSB_PA->FR1 |= GPIO_BIT_5;
            TSB_PA->IE |= GPIO_BIT_5;
      } else {
            /* nothing */
      }
}
```

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

```
      UART_InitTypeDef myUART;

      /* configure SIO0 for reception */
      UART_Enable(UART_0);
      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 Master UART0 channels.

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

FIFO configuration.

```
      UART_RxFIFOByteSel(UART0,UART_RXFIFO_RXFLEVEL);
      UART_TxFIFOINTCtrl(UART0,ENABLE);
      UART_RxFIFOINTCtrl(UART0,ENABLE);
      UART_TRxAutoDisable(UART0,UART_RXTXCNT_AUTODISABLE);
      UART_FIFOConfig(UART0,ENABLE);
      UART_RxFIFOFillLevel(UART0, UART_RXFIFO4B_FLEVLE_4_2B);
      UART_RxFIFOINTSel(UART0,UART_RFIS_REACH_EXCEED_FLEVEL);
      UART_RxFIFOClear(UART0);
      UART_TxFIFOFillLevel(UART0, UART_TXFIFO4B_FLEVLE_0_0B);
      UART_TxFIFOINTSel(UART0,UART_TFIS_REACH_NOREACH_FLEVEL);
      UART_TxFIFOClear(UART0);
```

After above setting, enable UART0 interrupt.

```
         NVIC_EnableIRQ(INTTX0_IRQn);
         NVIC_EnableIRQ(INTRX0_IRQn);
```

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

UART0 TX interrupt routine:

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

      if (TxCounter1 < NumToBeTx1) {
            UART_SetTxData(UART0, TxBuffer1[TxCounter1++]);
```

```
    } else {
        err = UART_GetErrState(UART0);
    }
}
```

UART0 RX interrupt routine:

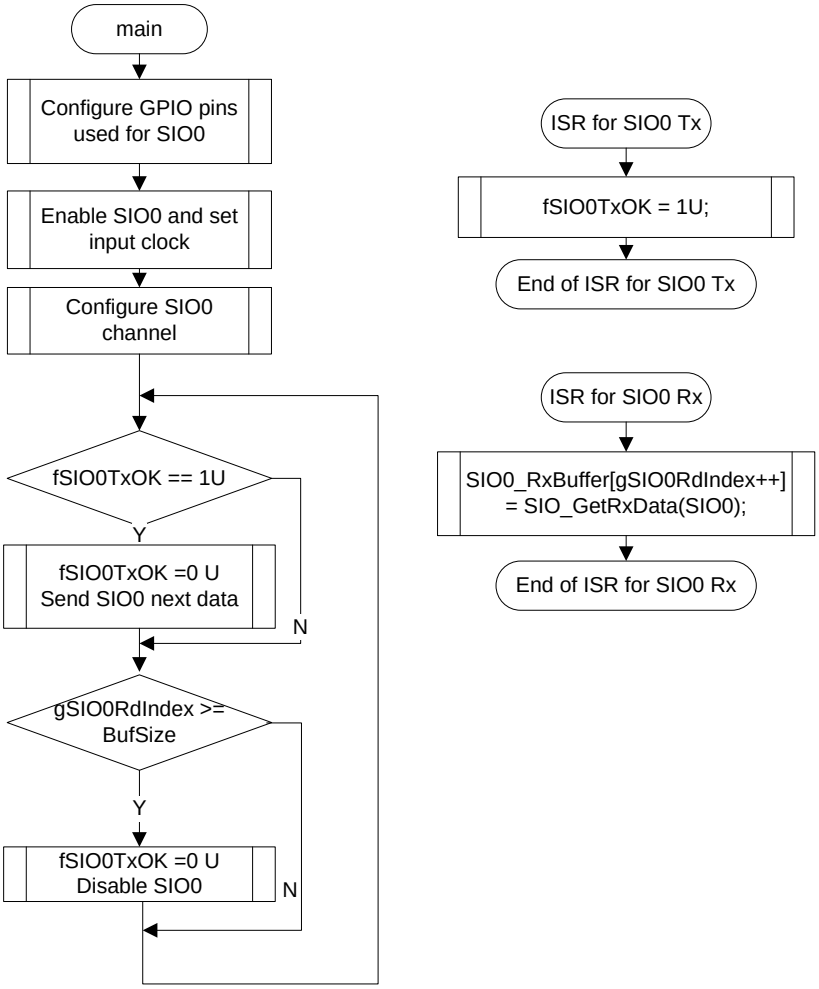
```
void INTRX0_IRQHandler(void)
{
    volatile UART_Err err;

    err = UART_GetErrState(UART0);
    if (UART_NO_ERR == err) {
        RxBuffer1[RxCounter1++] = (uint8_t) UART_GetRxData(UART0);
    }
}
```

7-7-4 Example: SIO Master

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

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

- **Flowchart**



- **Code and Explanation for the Example**

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

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

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

/* Selects input clock for prescaler as PhiT0. */
SIO_SetInputClock(SIO0, SIO_CLOCK_T0);

/*initialize the SIO0 struct */
SIO0_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO0_Init.TIDLE = SIO_TIDLE_HIGH;
SIO0_Init.IntervalTime = SIO_SINT_TIME_SCLK_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 the interrupt of SIO TX, RX.

```
/* Enable SIO0 Channel TX interrupt */
NVIC_EnableIRQ(INTTX0_IRQn);
/* Enable SIO0 Channel RX interrupt */
NVIC_EnableIRQ(INTRX0_IRQn);
```

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

```
while (1) {
        /*SIO0 send data from TXD0 */
          if (fSIO0TxOK == 1U) {
                fSIO0TxOK = 0U;
                SIO_SetTxData(SIO0, SIO0_TxBuffer[gSIO0WrIndex++]);
                if (gSIO0WrIndex == (BufSize+1))
                     gSIO0WrIndex = 0;
          } else {
                /*Do Nothing */
          }

        /*SIO0 receive data end */
          if (gSIO0RdIndex >= BufSize) {
              SIO_Disable(SIO0);
          } else {
                /*Do Nothing */
          }

      }
    }
```

In ISR of SIO0 Tx, set transfer is ok.

```
void INTTX0_IRQHandler (void)
{
    fSIO0TxOK = 1U;
}
```

In ISR of SIO0 Rx, get the receive data from RX buffer

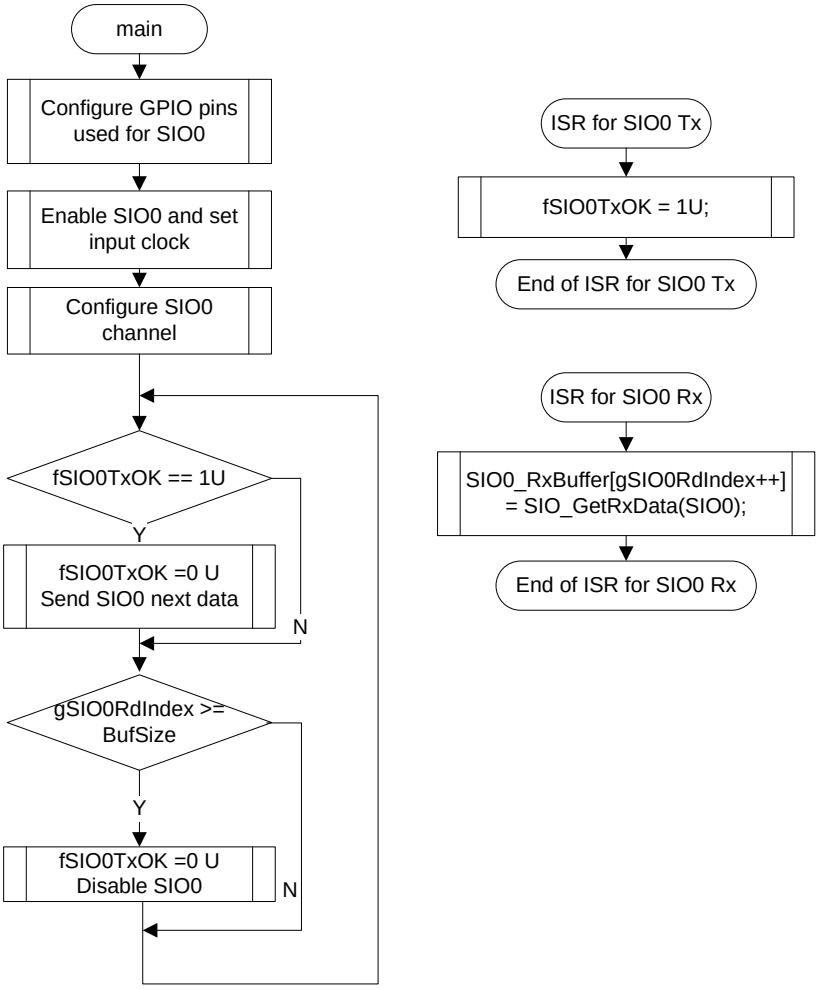
```
void INTRX0_IRQHandler(void)
{
    SIO0_RxBuffer[gSIO0RdIndex++] = SIO_GetRxData(SIO0);
}
```

7-7-5 Example: SIO Slave

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

The example includes:
1. Basic setup operation of SIO
2. The data transfer between Master SIO0 and Slave SIO0
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.

```
/*configure the SIO0 channel */
SIO_Enable(SIO0);
/* Selects input clock for prescaler as PhiT0. */
SIO_SetInputClock(SIO0, SIO_CLOCK_T0);

/*initialize the SIO0 struct */
SIO0_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO0_Init.TIDLE = SIO_TIDLE_HIGH;
SIO0_Init.TXDEMP = SIO_TXDEMP_HIGH;
SIO0_Init.EHOLDTime = SIO_EHOLD_FC_64;
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;

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

Enable the interrupt of SIO TX, RX.

```
/* Enable SIO0 Channel TX interrupt */
NVIC_EnableIRQ(INTTX0_IRQn);
/* Enable SIO0 Channel RX interrupt */
NVIC_EnableIRQ(INTRX0_IRQn);
```

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

```
while (1) {
        /*SIO0 send data from TXD0 */
         if (fSIO0TxOK == 1U) {
              fSIO0TxOK = 0U;
              SIO_SetTxData(SIO0, SIO0_TxBuffer[gSIO0WrIndex++]);
              if (gSIO0WrIndex == (BufSize+1))
                   gSIO0WrIndex = 0;
         } else {
              /*Do Nothing */
         }

        /*SIO0 receive data end */
         if (gSIO0RdIndex >= BufSize) {
             SIO_Disable(SIO0);
         } else {
              /*Do Nothing */
         }

    }
}
```

In ISR of SIO0 Tx, set transfer is ok.

```
void INTTX0_IRQHandler (void)
{
    fSIO0TxOK = 1U;
}
```

In ISR of SIO0 Rx, get the receive data from RX buffer

```
void INTRX0_IRQHandler(void)
{
    SIO0_RxBuffer[gSIO0RdIndex++] = SIO_GetRxData(SIO0);
}
```

7-8 uDMAC

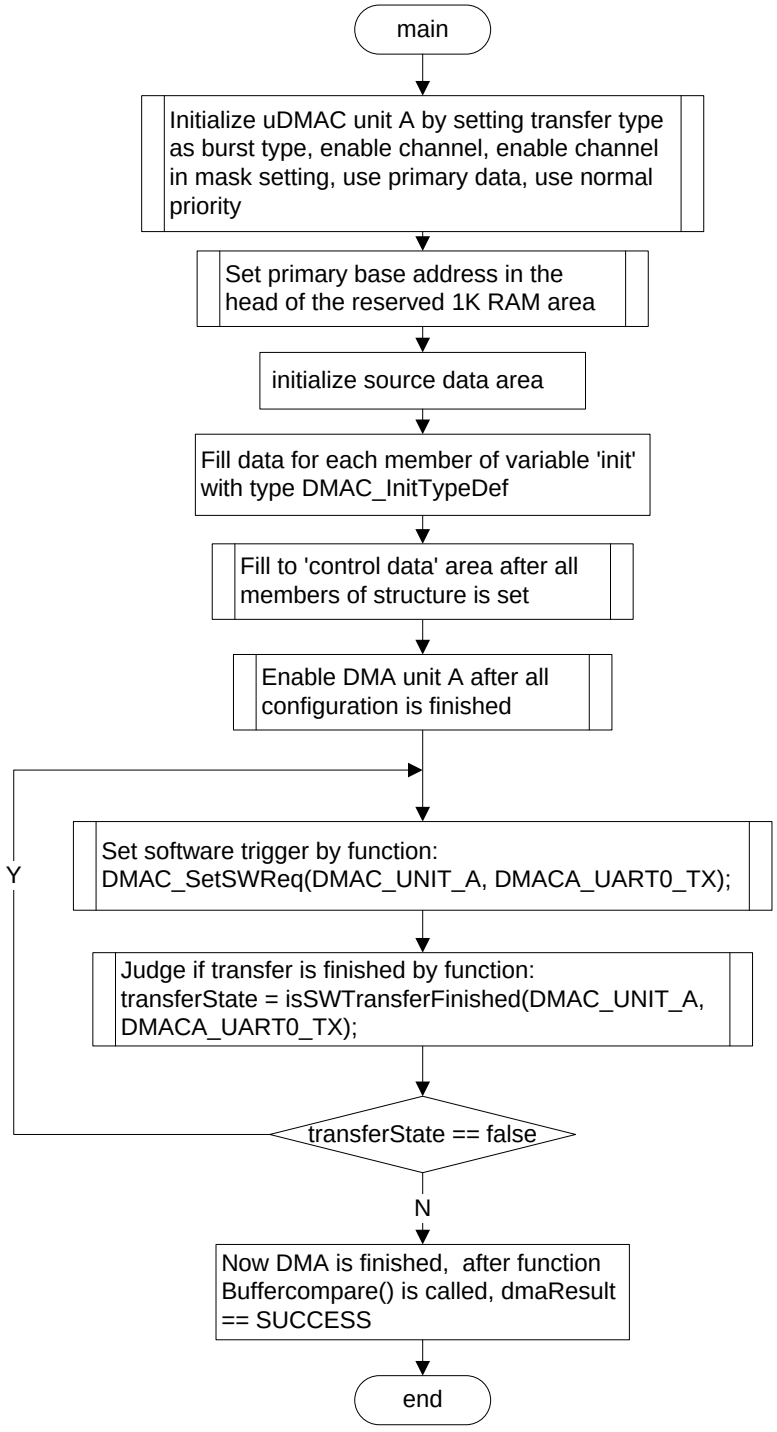
7-8-1 Example: DMA transfer from memory to memory

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

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 */
/* uDMAC_CFG_A are defined in file TMPM311CHDUG_RAM_For_uDMAC.icf */
uint32_t uDMAC_A_Control_Data[256U] @ ".uDMAC_CFG_A" ;

#elif defined ( __CC_ARM    )       /* Keil MDK */
#include <absacc.h>
#define uDMAC_CFG_A        (0x20000400U)
uint32_t uDMAC_A_Control_Data[256U] __at (uDMAC_CFG_A);
#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 "TMPM311CHDUG_RAM_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_UART0_TX, DMAC_BURST);
    DMAC_SetChannel(DMAC_UNIT_A, DMACA_UART0_TX, ENABLE);
    DMAC_SetMask(DMAC_UNIT_A, DMACA_UART0_TX, ENABLE);
    DMAC_SetPrimaryAlt(DMAC_UNIT_A,                    DMACA_UART0_TX,
DMAC_PRIMARY);
    DMAC_SetChannelPriority(DMAC_UNIT_A, DMACA_UART0_TX,
                                        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_UART0_TX, &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_UART0_TX);

  transferState = isSWTransferFinished(DMAC_UNIT_A,
DMACA_UART0_TX);

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

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

```
dmaResult = ERROR;
dmaResult = Buffercompare( src, dst, TX_NUMBERS);
```

7-9 WDT

7-9-1 Example:WDT

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

The example includes:

1. WDT initialization.
2. In DEMO1 WDT isn’t cleared before timer overflow, NMI interrupt is generated.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();
```