TOSHIBA

TX03 Peripheral Driver Usage Example (TMPM361/362/363/364)

Ver 1

Sep, 2017

TOSHIBA ELECTRONIC DEVICES & STORAGE CORPORATION

CMDR-M364UE-01E

RESTRICTIONS ON PRODUCT USE

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

Index

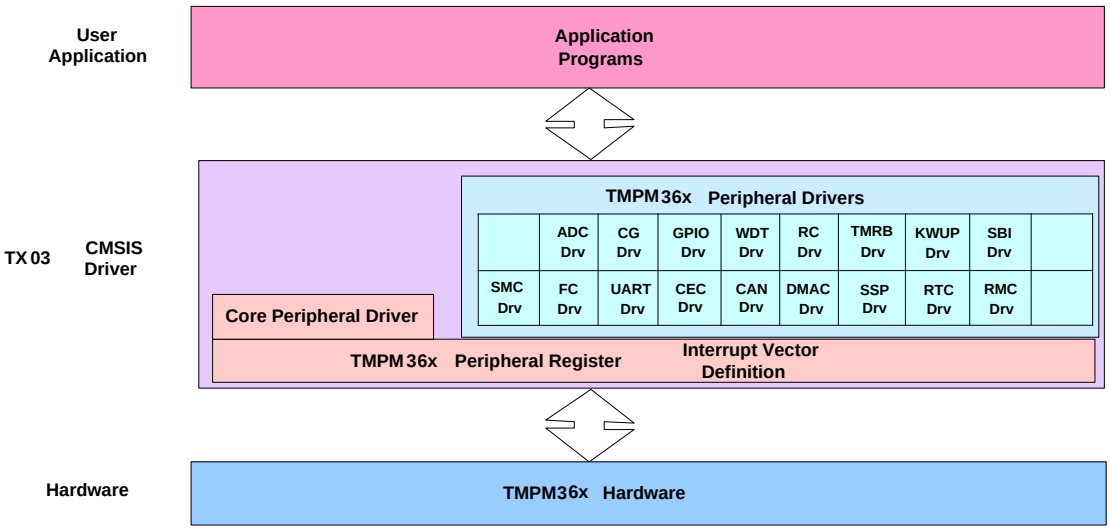
1 General description

The example programs described hereafter are specially designed for TOSHIBA TMPM36x (see **Note** below) 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.

***Note**: The "TMPM36x" in this document can be TMPM361/362/363/364.

2 Overview

User application utilizes TX03 peripheral driver as following.



3 Build-in hardware usage

Hardware	Channel	Use presence, use
CG	Clock Gear	Used for CG demo
	PLL	PLL on (4x)
Standby mode	-	Used for CG demo (SLEEP mode)
SysTick	-	Unused
Watch dog timer (WDT)	-	Used for WDT
External interrupt (INT)	INTTB0	Used for TMRB demo
	INTTX11	Used for SIO/UART demo
	INTCANRX	Used for CAN demo
	INTCECRX	Used for CEC demo

	INTCECTX	Used for CEC demo
	INTDMACTC0	Used for DMAC demo
	INTRMCRX0	Used for RMC demo
	INTRTC	Used for RTC demo
	INTSBI2	Used for SBI demo
	Other	Unused
SIO/UART	SIO/UART11	Used for UART demo
	Other	Unused
CAN	-	Used for CAN demo
CEC	-	Used for CEC demo
16-bit timer	TMRB0	Used for TMRB demo
	Other	Unused
12-bit A/D converter	AIN0	Used as ADC data read
	Other	Unused
DMAC	-	Used for DMAC demo
Real-time clock	-	Used for RTC demo
SBI	SBI2	Used for SBI demo: transmitter
	Other	Unused
SSP	SSP0	Used for SSP0 self-loopback demo
RMC	RMC0	Used for RMC demo: receive
	Other	Unused
KWUP	-	Used for KWUP demo
RC	-	Used for RC demo
RMC	-	Used for SMC demo

4 Pin Usage

The example programs are tested on TMPM36x evaluation board.
Following is pin usage for example programs.

No				Name	Usage
M361	M362	M363	M364		
	1		1	PK6, AIN14	Unused
	2		2	PK7, AIN15	Unused
1	3	1	3	AVSS	AD converter GND
2	4	2	4	VREFH	Supplying the AD converter with a reference power supply
3	5	3	5	RESETn	Reset input
4	6	4	6	MODE	Unused
5	7	5	7	PL0, SDA0, TB0OUT	Unused
6	8	6	8	PL1, SCL0, TB1OUT	Unused

7	9	7	9	PL2, SCK0, TB2OUT	Unused
8	10	8	10	PL3, INT0, TB3OUT	Unused
9	11	9	11	PL4, TXD1, TB4OUT	Unused
10	12	10	12	PL5, RXD1, TB5OUT	Unused
11	13	11	13	PL6, SCLK1, TB6OUT, CTS1n	Unused
12	14	12	14	PL7, INT1, TB7OUT	Unused
13	15	13	15	DVSS	GND
14	16	14	16	PM0, SCLK2, TB1IN0, CTS2n	Unused
15	17	15	17	PM1, TXD2, TB1IN1	Unused
16	18	16	18	PM2, RXD2, ALARMn	Unused
17	19	17	19	PM3, INT2, TB3OUT	Unused
18	20	18	20	PM4, SCLK3, CTS3n	LED1
19	21	19	21	PM5, TXD3	LED2
20	22	20	22	PM6, RXD3	LED3
21	23	21	23	PM7, INT3	LED4
22	24	22	24	PN0, TXD4	Unused
23	25	23	25	PN1, RXD4	Unused
24	26	24	26	PN2, SCLK4, TB2IN0, CTS4n	Unused
25	27	25	27	PN3, INT4, TB2IN1, RMIN0	RMC RX
	28		28	PN4, TXD5	Unused
	29		29	PN5, RXD5	Unused
	30		30	PN6, SCLK5, TBFIN0, CTS5n	Unused
	31		31	PN7, INT8, TBFIN1, RMIN1	Unused
	32		32	PO0, TXD6, TB8OUT	Unused
	33		33	PO1, RXD6, TB9OUT	Unused
	34		34	PO2, SCLK6, TBAOUT, CTS6n	Unused
	35		35	PO3, INT9, TBBOUT	Unused
	36		36	PO4, TXD7, TBCOUT	Unused
	37		37	PO5, RXD7, TBDOUT	Unused
	38		38	PO6, SCLK7, TBEOUT, CTS7n	Unused
	39		39	PO7, INTA, TBFOUT	Unused
26	40	26	40	PP0, CS2n	Unused
27	41	27	41	PP1	Unused
28	42	28	42	PP2, BLS0n, SPDO	SPP SPDO
29	43	29	43	PP3, BLS1n, SPDI	SPP SPDI
30	44	30	44	PP4, WEn, SPCLK	SPP SPCLK
31	45	31	45	PP5, OEn, SPFSS	SPP SPFSS
32	46	32	46	PP6, ALEn	Unused
33	47	33	47	DVDD3B	+3.3V
34	48	34	48	DVSS	GND
35	49	35	49	PA0, D0, AD0	Unused
36	50	36	50	PA1, D1, AD1	Unused

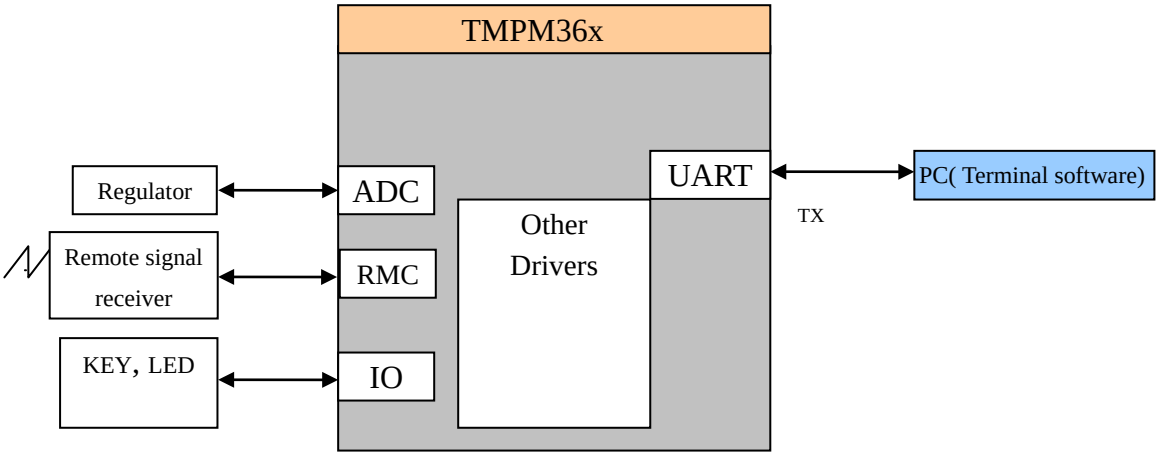
37	51	37	51	PA2, D2, AD2	Unused
38	52	38	52	PA3, D3, AD3	Unused
39	53	39	53	PA4, D4, AD4	Unused
40	54	40	54	PA5, D5, AD5	Unused
41	55	41	55	PA6, D6, AD6	Unused
42	56	42	56	PA7, D7, AD7	Unused
43	57	43	57	PB0, D8, AD8	Unused
44	58	44	58	PB1, D9, AD9	Unused
45	59	45	59	PB2, D10, AD10	Unused
46	60	46	60	PB3, D11, AD11	Unused
47	61	47	61	PB4, D12, AD12	Unused
48	62	48	62	PB5, D13, AD13	Unused
49	63	49	63	PB6, D14, AD14	Unused
50	64	50	64	PB7, D15, AD15	Unused
	65		65	PC0, A1, TXD8	Unused
	66		66	PC1, A2, RXD8	Unused
	67		67	PC2, A3, SCLK8, CTS8n	Unused
	68		68	PC3, A4	Unused
	69		69	PC4, A5, TXD9	Unused
	70		70	PC5, A6, RXD9	Unused
	71		71	PC6, A7, SCLK9, CTS9n	Unused
	72		72	PC7, A8	Unused
	73		73	PD0, A9, TXD10	Unused
	74		74	PD1, A10, RXD10	Unused
	75		75	PD2, A11, SCLK10, CTS10n	Unused
	76		76	PD3, A12	Unused
	77		77	PD4, A13, TXD11	UART TX
	78		78	PD5, A14, RXD11	UART RX
	79		79	PD6, A15, SCLK11, CTS11n	Unused
	80		80	PD7, A16, INTB	Unused
51	81	51	81	PE0, A17, TB5IN0	Unused
52	82	52	82	PE1, A18, TB5IN1	Unused
53	83	53	83	PE2, A19, TB6IN0	Unused
54	84	54	84	PE3, A20, TB6IN1	Unused
55	85	55	85	PE4, A21, TXD0, CTXD	CAN TX
56	86	56	86	PE5, A22, RXD0, CRXD	CAN RX
57	87	57	87	PE6, A23, SCLK0, CTS0n	Unused
58	88	58	88	PE7, INT5, SCOUT	Unused
59	89	59	89	DVDD3B	+3.3V
60	90	60	90	DVSS	GND
61	91	61	91	SWDIO	Unused
62	92	62	92	SWCLK	Unused

63	93	63	93	PF0, TRACECLK	Unused
64	94	64	94	PF1, TRACEDATA0, SWV	Unused
65	95	65	95	PF2, TRACEDATA1	Unused
66	96	66	96	PF3, TRACEDATA2	Unused
67	97	67	97	PF4, TRACEDATA3	Unused
68	98	68	98	PG0, SDA1, TB7IN0	Unused
69	99	69	99	PG1, SCL1, TB7IN1	Unused
70	100	70	100	PG2, SCK1, CS0n	Unused
71	101	71	101	PG3, INT6, CS1n	Unused
72	102	72	102	PG4, SDA2, TB9IN0	SBI master TX
73	103	73	103	PG5, SCL2, TB9IN1	SBI master CLK
74	104	74	104	PG6, SCK2, USBPON, CS3n	Unused
75	105	75	105	PG7, INT7, USBOC, WDTOUTn	Unused
	106		106	PH0, SDA3, TBAIN0	Unused
	107		107	PH1, SCL3, TBAIN1	Unused
	108		108	PH2, SCK3, TBBIN0	Unused
	109		109	PH3, INTC, TBBIN1	Unused
	110		110	PH4, SDA4, TBDIN0	Unused
	111		111	PH5, SCL4, TBDIN1	Unused
	112		112	PH6, SCK4, TBEIN0	Unused
	113		113	PH7, INTD, TBEIN1	Unused
76	114	76	114	RVDD3	+3.3V
77	115	77	115	XT1	Connect to low-speed oscillator
78	116	78	116	XT2	Connect to low-speed oscillator
79	117	79	117	DVDD3A	+3.3V
80	118	80	118	X1	Connect to high-speed oscillator
81	119	81	119	DVSS	GND
82	120	82	120	X2	Connect to high-speed oscillator
83	121	83	121	DVDD3B	+3.3V
84	122	84	122	DVSS	GND
85	123	85	123	D+	Unused
86	124	86	124	D-	Unused
87	125	87	125	NMIn	Unused
88	126	88	126	TEST1	Unused
89	127	89	127	TEST2	Unused
90	128	90	128	PI0, BOOTn	Unused
91	129	91	129	PI1, CEC	CEC
92	130	92	130	AVDD3	Supplying AD converter with

					power supply
93	131	93	131	PJ0, AIN0	ADC IN
94	132	94	132	PJ1, AIN1	Unused
95	133	95	133	PJ2, AIN2	Unused
96	134	96	134	PJ3, AIN3, ADTRGn	Unused
97	135	97	135	PJ4, AIN4, KWUP0	KEY1
98	136	98	136	PJ5, AIN5, KWUP1	KEY2
99	137	99	137	PJ6, AIN6, KWUP2	KEY3
100	138	100	138	PJ7, AIN7, KWUP3	KEY4
	139		139	PK0, AIN8	Unused
	140		140	PK1, AIN9	Unused
	141		141	PK2, AIN10	Unused
	142		142	PK3, AIN11	Unused
	143		143	PK4, AIN12	Unused
	144		144	PK5, AIN13	Unused

5 Development Environment

Following is development environment:



1. Hardware board:
   TMPM364 evaluation board

2. Development tool:
   - IAR:
     1) J-Link: IAR J-Link-ARM 7.0 / J-Link 6.0

2) IDE: IAR Embedded workbench 5.5 version

- KEIL:
  1) U-Link: RealView ULINK2
  2) IDE: KEIL uVision 4.10

6 Functional description

6-1 Operation mode

There are four operation modes for TMPM36Bx: NORMAL, IDLE, STOP1, STOP2 mode.
There are five operation modes for TMPM36x: NORMAL, SLOW, IDLE2/1, SLEEP and STOP 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.

- **SLOW mode:**

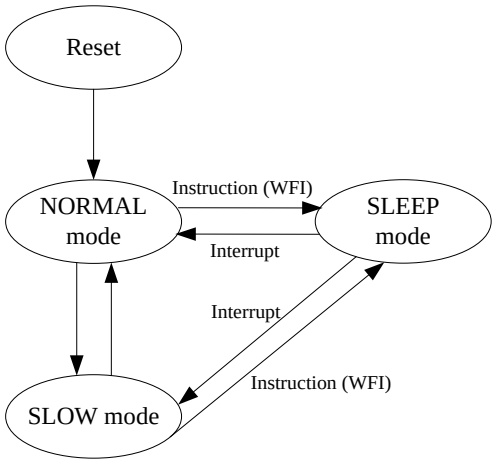
  This mode is to operate the CPU core and the peripheral hardware by using the low-speed clock with high-speed clock stopped. The SLOW mode reduces power consumption compared to the NORMAL mode. This mode allows only the following peripheral functions to operate: I/O ports, real-time clock (RTC), CEC and remote control signal preprocessor (RMC).

- **SLEEP mode:**

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

IDLE2/1, SLEEP and STOP modes are low power mode.
To shift to lower power mode, in system control register CGSTBYCR<STBY[2:0]>, select the IDLE2/1 or SLEEP or STOP mode, and run the WFI (Wait For Interrupt) command.

Note: Only SLEEP mode is demonstrated in driver example.



6-2 ADC

This is a potentiometer that connected to PJ0/AIN0. The voltage on it will be measured.

6-3 CAN

This example demonstrates the test loop back mode of CAN APIs, such as mailbox message setting, mailbox configuration, message transmission/receiving, mode change (test loop back mode) status getting and usage for interrupt.

Note: This CAN demo could run at KEIL TMPM363 and TMPM364 evaluation board, but it was only tested on TMPM364 evaluation board.

Demo procedure:

1. Use a wire to connect pin PE4 with PE5 (pin No.85 and 86 on evaluation board).
2. Power on, LED PM4~PM7 turn off.
3. Press button PJ4, enable CAN test loop back mode and ADC module.
4. Wait ADC to sample potentiometer AIN0 and read out the sampling value.
5. Copy the value into mailbox0 (transmission mailbox), and enable mailbox10 (receiving mailbox).
6. After mailbox10 receiving the value, displays the high 4 bits of the value on LED PM4~PM7.
7. Repeat step 4, the LED PM4~PM7 will change with rotating the potentiometer AIN0.

6-4 CEC

The process is described as follow:

1. The target board receives the CEC message from DVD player
2. Print the CEC message data on the Terminal I/O of IAR Embedded Workbench
3. When the target board receives the “Active_Source” (CEC command) from DVD player (When DVD player turn on or DVD play a source, the command will be sent), the target board send “Standby” command to DVD player.
4. DVD player should be change to standby mode.

6-5 CG

This example demonstrates how to switch from NORMAL mode to SLEEP mode.

6-6 DMAC

This example implements how to use DMA with software trigger to transfer data from RAM area "SRC_BUFFER" to "DST_BUFFER".

The driver example step:

1 Initialize DMAC by setting transfer type as burst type, enable channel.
2 Configure the setting data for software trigger and fill it to 'control data' area.
3 Enable DMAC after all configurations is finished.
4 Judge if transfer is finished, then compare the data in destination with source.

6-7 FLASH

This example demonstrates the feature of flash APIs such as the FC register write and read operation.

6-8 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.

6-9 KWUP

This example implements press KEY1 CPU enter low power mode, LED is turn off, then press KEY0 to release low power mode.

6-10 RC

This example demonstrates the feature of ram control APIs such as the RC register write and read operation.

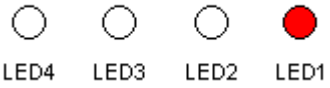
6-11 RMC

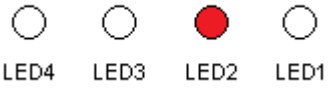
This example intends to catch the remote signal data and decode it.
TOSHIBA format or RC5 format remote control device is available in this application; user can change the definition on header file to select one remote control format. In this application, if the M36x demo board receives the TOSHIBA format or RC5 format remote control signal, then it will print the received data on the Terminal I/O of IAR Embedded Workbench, just as ”RMC DATA: 0x2fd807f ”(TOSHIBA format).

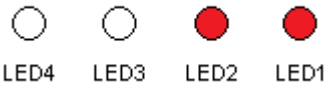
6-12 RTC

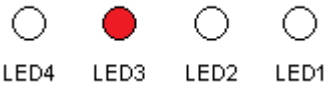
The internal RTC is displayed to display the second on LEDs.
The second of the time as binary is displayed on LEDs.

LEDs displays example:

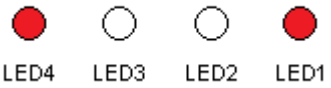
1s:


2s:


3s:


4s:


…….

59s:


6-13 SBI&I2C

This example intends to support the I2C bus slave mode.
Use SBI/I2C interrupt to handle I2C bus read/write.

I2C Slave (I2C) write 2bytes from SBI2 to EEPROM (Address is 0xA0), and I2C Slave (I2C) reads 2bytes data from EEPROM to SBI2.

Received results can be checked by RAM variable gI2C_RxData[ ] in debugger.

Note: In order to run this sample software, connect SBI2 on the evaluation board and EEPROM on M330 IAR board.

6-14 SMC

This example demonstrates procedure of initialize SRAM.

6-15 SSP

This example demonstrates the test loop back mode of SSP APIs.
Infinite data is sent and checked if the received data is OK or not. And if transmission is set to lowest speed mode (lowest bit rate), what happens will be checked.

6-16 TMRB

This example demonstrates self-loop back procedure. This function implements a general timer utilizing MCU timer.
Timer trailing timing is set to 1ms.

6-17 SIO/UART

6-17-1 UART transfer demo

This example demonstrates transfer procedure.

6-17-2 UART FIFO

In this sample program, UART0 sent the data "TMPM3611" to UART1 use FIFO, and at same time UART0 receive the data "TMPM3612" which send from UART1 and also use FIFO.
Set one breakpoint before the function ResetIdx(),when program stop in the breakpoint you can read the RxBuffer = "TMPM3612",and RxBuffer1 = "TMPM3611".

6-17-3 SIO

This demo will show synchronously transfer and receive usage of SIO module in TMPM361. It use the channel SIO0, SIO1 and transfer data synchronously between them. (Connect TXD0 with RXD1, connect TXD1 with RXD0, connect sclk0 with sclk1)

6-18 WDT

This example implements how to setup the WDT API to generate NMI interrupt.

7 Functional description

This software project creates the sample applications based on KEIL MCBTMPM360 Evaluation Board and TOSHIBA TMPM364 Evaluation Board which will demonstrate the main feature of TMPM36xMCU.

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:

```
\---TMPM36x
    +---Libraries
    |   +---TX03_CMSIS
    |   |   |   system_TMPM36x.c
    |   |   |   system_TMPM36x.h
    |   |   |   TMPM36x.h
    |   |   \---startup
    |   |       +---arm
    |   |       |       startup_TMPM36x.s
    |   |       \---iar
    |   |               startup_TMPM36x.s
    |   \---TX03_Periph_Driver
    |       +---inc
    |       |       tmpm36x_adc.h
    |       |           :
    |       |       tmpm36x_wdt.h
    |       |       tx03_common.h
    |       \---src
    |               tmpm36x_adc.c
    |                   :
    |               tmpm365_wdt.c
    \---Project
        +---Examples            //only 1 examples listed here
        |   +---ADC
        |   |   \---ADC_Data_Read
        |   |       +---App
        |   |       |       main.c
        |   |       +---IAR
        |   |       |       ADC_Data_Read.ewd
        |   |       |       ADC_Data_Read.ewp
        |   |       |       ADC_Data_Read.eww
```

```
|   |         \---KEIL
|   |                   ADC_Data_Read.uvopt
|   |                   ADC_Data_Read.uvproj
|   \---Workspace
|         +---IAR
|         |         Examples_for_M36x_Driver.eww
|         \---KEIL
|                   Examples_for_M36x_Driver.uvmpw
\---Template
      +---IAR
      |         TMPM36x_flash.icf
      \---KEIL
                tmpm36x.sct
```

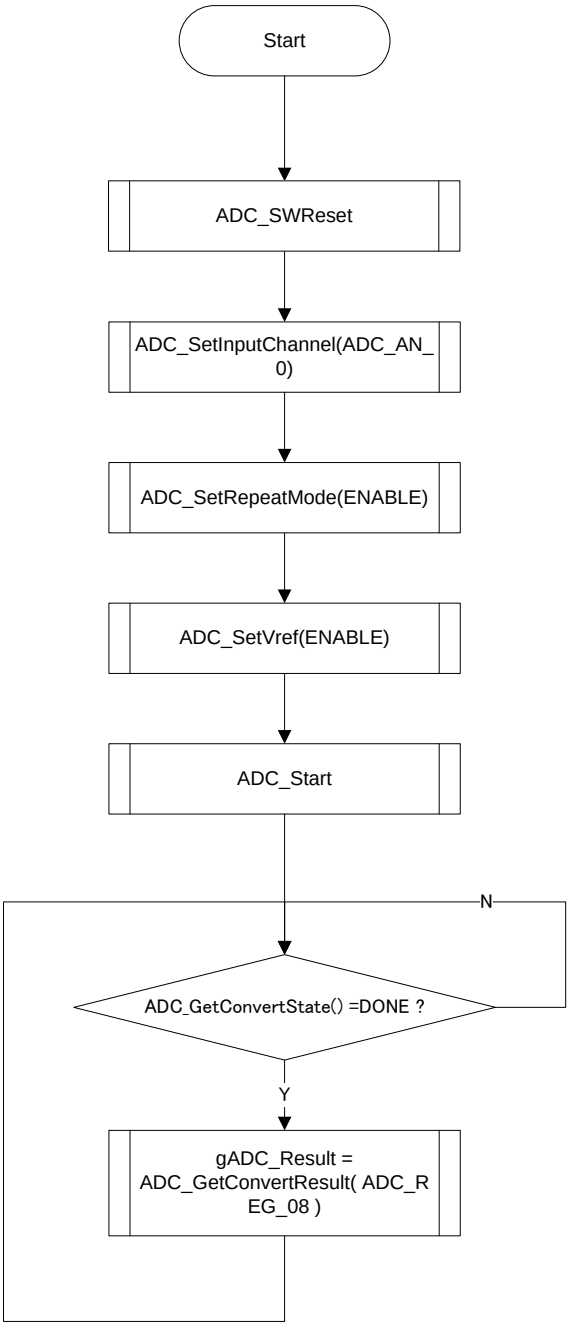
7-1 ADC

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

The example includes:

1. ADC configuration and initialization
2. Start ADC in fixed-channel repeat mode and read AD result of AIN0

- **Flowchart**



- **Code and Explanation for the Example**

At first, initial ADC setting for AN0, repeat mode, ADC Vref ON. For example,

```
ADC_SetInputChannel(ADC_AN_0);
ADC_SetRepeatMode(ENABLE);
ADC_SetVref(ENABLE);
```

Then start ADC.

```
ADC_Start();
```

After ADC start, check ADC convert end or not, if ADC is finished, the read the ADC result data.

```
gADC_state = ADC_GetConvertState();
if (DONE == gADC_state) {
      gADC_state = BUSY;
      gADC_Result = ADC_GetConvertResult(ADC_REG_08);
}
```

7-2 CAN

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

The example includes:

1. CAN mailbox configuration, bit timing setting.
2. CAN mode: suspend, configuration, normal and test loop back mode.
3. CAN transmission/receiving method.
4. CAN module status, interrupt usage.

- **Flowchart**

- **Code and Explanation for the Example**

At first initialize all variable used by this demo, include CAN global state, interrupt mask, ADC state, ADC result, CAN bit timing parameters, message for transmission, message for receiving.

```
CAN_GlobalState globalstate;
CAN_INTFactor intmask;
WorkState adc_state = BUSY;
ADC_ResultTypeDef adc_value;
CAN_BitTimingTypeDef bittiming ={ BAUDRATE_PRESCALER,    //Brp = 1023
                                  CAN_TIMING_TSEG1_4TQ, //TSEG1 = 4TQ
                                  CAN_TIMING_TSEG2_4TQ, //TSEG2 = 4TQ
                                  CAN_SINGLE_SAMPLING,    //Single sampling
                                  CAN_TIMING_SJW_1TQ };    //SJW = 1TQ
CAN_MsgTypeDef msgtx = { 0U };
CAN_MsgTypeDef msgrx = { 0U };
```

Then initialize LED, KEY, CAN TX/RX port, ADC on evaluation board.

```
LED_Configuration();
KEY_Configuration();
CAN_PortInit();
ADC_Init();
```

Configure CAN bit timing by using API CAN_ConfigBitTiming(). The baud rate formula is BR = fosc/((TSEG1+TSEG2+3)*(BRP+1)).
For example, fosc = 12MHz, TSEG1 = 6 (7TQ), TSEG2 = 3 (4TQ), BRP = 1, so the baud rate is BR = 12/((6 + 3 + 3) * (1 + 1)) = 500Kbit/s. Note that the bit timing setting shall only be set in configuration mode, after reset, the default mode is configuration mode.

```
/* CAN bit timing setting */
CAN_ConfigBitTiming(&bittiming);
```

Set mailbox0 message ID, data length and configure it as a transmission mailbox.

```
/* CAN Tx mailbox0 setting */
msgtx.MsgID = MSG_ID;
msgtx.MsgDataLen = 1U;          /* data length: 1 byte */
CAN_SetMbxMsg(CAN_MBX_0, &msgtx);
CAN_SetMbxDirection(CAN_MBX_0, CAN_MBX_TX); /* mailbox0: Tx */
```

Set mailbox10 as a receiving mailbox, and the message ID is the same as mailbox0.

```
/* CAN Rx mailbox10 setting */
msgrx.MsgID = msgtx.MsgID;    /* ID of mailbox10 = ID of mailbox0 */
CAN_SetMbxMsg(CAN_MBX_10, &msgrx);
CAN_SetMbxDirection(CAN_MBX_10, CAN_MBX_RX); /* mailbox10: Rx */
```

Configure mailbox transmission order by using API CAN_SetTxOrder(), If the order is by mailbox number, the lower number mailbox will be sent first. In addition, if the order is by ID priority, the higher priority identifier will be sent first.

```
CAN_SetTxOrder(CAN_TX_ORDER_MBX_NUM);/* Order: by mailbox number */
```

Next, prepare to set test loop back mode, and this mode should only be enabled or disabled in suspend mode first. CAN operation mode is changed by API CAN_SetMode(), after setting the mode, the user must wait the specified global state acknowledge bit to set. The global state can be read out by API CAN_GetGlobalState().

```
/* Enter suspend mode */
CAN_SetMode(CAN_SUSPEND_MODE);
while (globalstate.Bit.SuspendModeAck == 0U) {
      globalstate = CAN_GetGlobalState();
}
```

After entering suspend mode, the test loop back mode could be set by API CAN_SetTestMode(). In this mode, CAN can receive its own transmitted message and will generate its own acknowledge bit. No other CAN node is necessary for the operation.

```
/* Set test loop back mode */
CAN_SetTestMode(CAN_TEST_LOOP_BACK_MODE, ENABLE);
```

Then return to normal mode again. CAN is in normal operation mode with enabled test loop back mode. Use API CAN_GetGlobalState() to check and wait the CAN returns normal mode.

```
/* Return normal mode */
CAN_SetMode(CAN_NORMAL_MODE);
while (globalstate.Bit.SuspendModeAck == 1U) {
    globalstate = CAN_GetGlobalState();
}
```

Enable mailbox10 receiving interrupt by using API CAN_SetINTMask(), if the specified mailbox bit is set, the interrupt is enable.

```
intmask.MbxBit.Mbx10 = 1U;
CAN_SetINTMask(CAN_INT_MBX_MASK, &intmask);
NVIC_EnableIRQ(INTCANRX_IRQn);
```

After all initialization setting, wait for PJ4 button in evaluation board being pressed down. If the button has been pressed, enable mailbox0 and 10 by using API CAN_EnableMbx(), and also enable ADC module to sample.

```
/* Wait for PJ4 pressing */
while (!KEY_Get(KEY1)) {
    /* Do nothing */
}
/* Enable mailbox0 and 10 */
CAN_EnableMbx(CAN_MBX_0 | CAN_MBX_10);
/* ADC start to run */
ADC_Start();
```

Checking the ADC module state in a dead loop, if the sampling has finished, read out the sampling value and put 8-bit into message data[0], then use API CAN_SetMbxMsg() to update the data in mailbox0, then use API CAN_SetTxReq() to send the message in mailbox0. Repeat this procedure, and send the ADC sampling value by CAN bus.

```
while (1) {
    adc_state = ADC_GetConvertState();
    if (DONE == adc_state) {            /* ADC convertion finished */
        adc_state = BUSY;
        adc_value = ADC_GetConvertResult(ADC_REG_08);
        msgtx.MsgData[0] = (uint8_t) adc_value.ADCResultValue;

        CAN_SetMbxMsg(CAN_MBX_0, &msgtx);/* Put the result into mailbox0 */
        CAN_SetTxReq(CAN_MBX_0, ENABLE); /* Start to send */
    } else {
        /* Do nothing */
    }
    delay(50000U);
}
```

If mailbox10 received the message sent by mailbox0, the CAN RX interrupt will be trigged. In IRQ INTCANRX_IRQHandler(), check the receiving state by using API CAN_GetMbxTRxState(), if mailbox received a message, the specified bit will be set. Then use API CAN_GetMbxMsg() to read the message and displays the high 4-bit of message data[0] on LED PM4~7.

```
void INTCANRX_IRQHandler(void)
{
  CAN_MbxState mbxstate = { 0U };
  CAN_MsgTypeDef msgrx = { 0U };
  mbxstate = CAN_GetMbxTRxState(CAN_STATE_ID_RX_ACK); /* Rx state */
  if (mbxstate.Bit.Mbx10 == 1U) {        /* If mailbox10 has received a message */
    CAN_GetMbxMsg(CAN_MBX_10, &msgrx);        /* Read mailbox10 */
    LED_Display(msgrx.MsgData[0] >> 4U);/* Get the data, displays on LED*/
                                                /* PM4~7 */
  } else {
    /* Do nothing */
  }
```

In the end, use API CAN_ClearMbxTRxState() to clear mailbox RX state flag and use API CAN_ClearINTFlag() to clear mailbox RX interrupt flag. The corresponding status flags and interrupt flags have to be cleared separately.

```
  CAN_ClearMbxTRxState(CAN_STATE_ID_RX_ACK);   /* Clear the Rx state */
  CAN_ClearINTFlag(CAN_INT_TYPE_RX);   /* Clear the Rx interrupt flag */
}
```

7-3 CEC

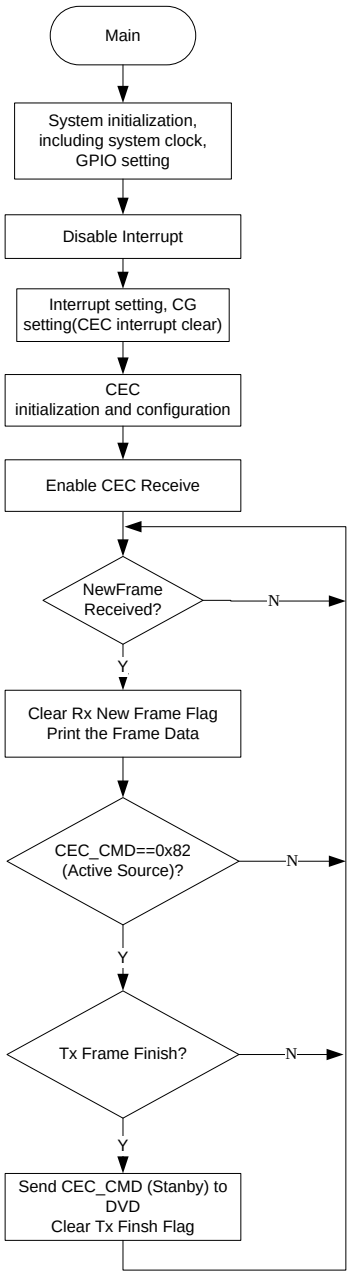
This is a simple example based on the TX03 Peripheral Driver (CEC, GPIO).

The example includes:

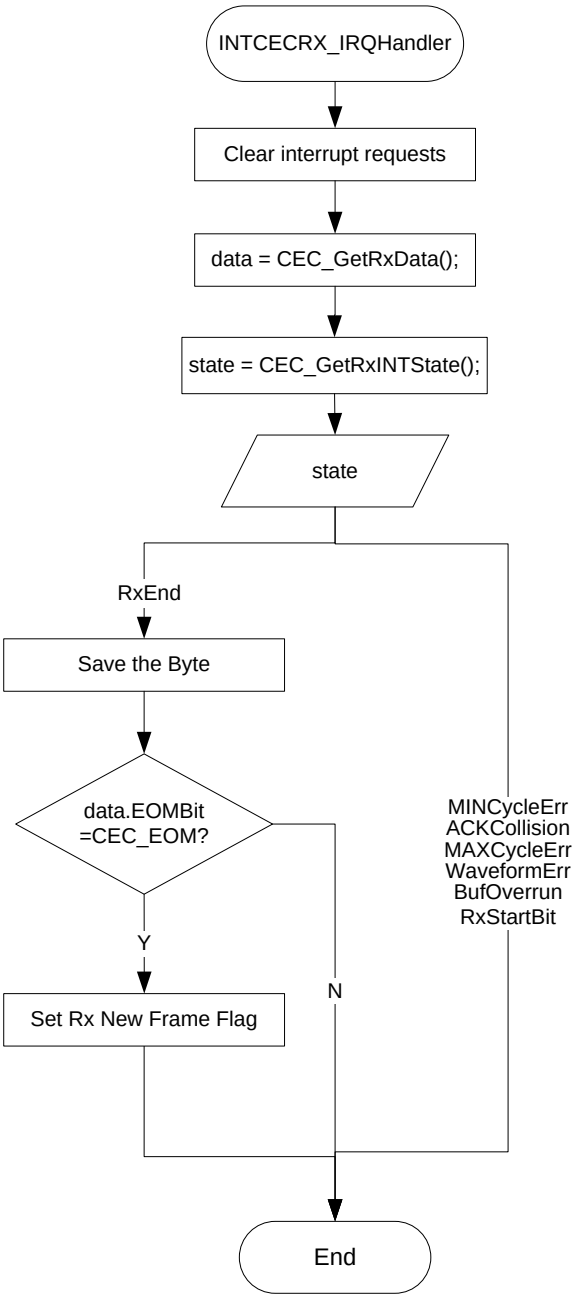
1. CEC configuration.
2. CEC transmission/receiving method.
3. CEC module status, interrupt usage.

- **Flowchart**

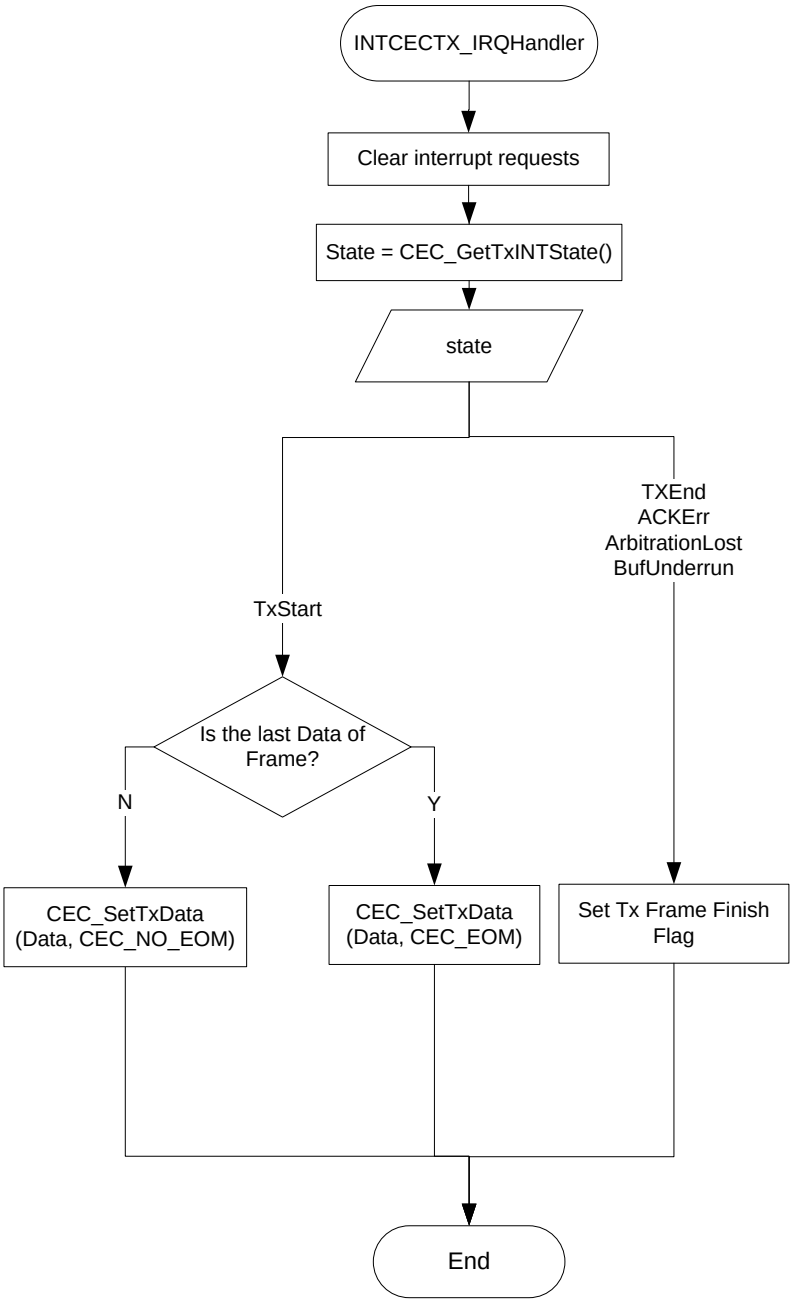
[Main Function]



[CEC Receive Interrupt]



[CEC Transmission Interrupt]



- **Code and Explanation for the Example**

[Main Function]

At first, initialize and configure CEC. For example:

```
CEC_Enable();
CEC_SWReset()
CEC_DefaultConfig()
```

Then Set the logical address and enable reception of CEC. For example:

```
CEC_SetLogicalAddr(CEC_TV);
CEC_SetRxCtrl(ENABLE);
```

After the setting above send the first byte of a frame. For example

```
if(CEC_BROADCAST==Destination){
      CEC_SetTxBroadcast(ENABLE);
}else{
      CEC_SetTxBroadcast(DISABLE);
}
CEC_SetTxData(CEC_Data[0], CEC_NO_EOM);
CEC_StartTx();
```

[CEC Receive Interrupt]

In the interrupt function: INTCECRX_IRQHandler
First get interrupt state:

```
CEC_RxINTState state;
state = CEC_GetRxINTState();
```

Then get Receive data:

```
CEC_DataTypeDef data;
data = CEC_GetRxData();
```

[CEC Transmission Interrupt]

In the interrupt function: INTCECTX_IRQHandler
First get interrupt state:

```
CEC_TxINTState state;
state = CEC_GetTxINTState ();
```

It the next data isn't the last data of a frame then send next data:

```
CEC_SetTxData(CEC_Data[current_num],CEC_NO_EOM);
```

It the next data is the last data of a frame then send last data as follow:

```
CEC_SetTxData(CEC_Data[current_num],CEC_EOM);
```

7-4 CG

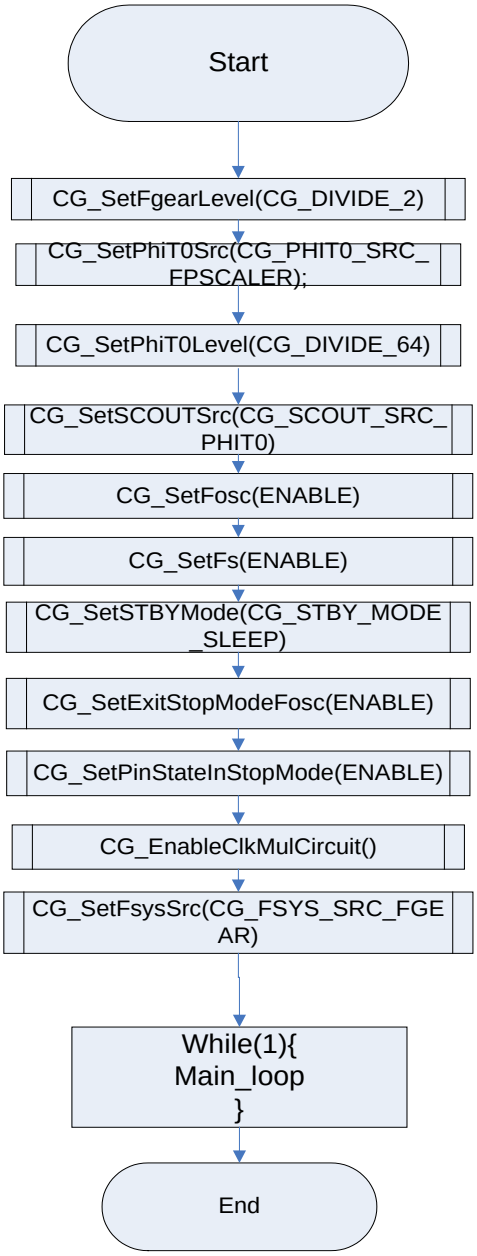
This is a simple example based on the TX03 Peripheral Driver (CG).

The example includes:

5. Basic setup operation of CG
6. How to switch from NORMAL mode to SLEEP mode.

- **Flowchart**

[Main Function]



- **Code and Explanation for the Example**

[Normal setup for CG (after Reset)]

The following codes are just an example for setting CG in Normal mode.
Example code is as the following:

```
/* set fgear = fc/2 */
CG_SetFgearLevel(CG_DIVIDE_2);

#if defined(__TMPM361_CG_H) || defined(__TMPM362_CG_H) ||
  defined(__TMPM363_CG_H) || defined(__TMPM364_CG_H)
CG_SetPhiT0Src
CG_SetPhiT0Src(CG_PHIT0_SRC_FPSCALER);
#endif
CG_SetPhiT0Level(CG_DIVIDE_64);

/* set SCOUT source to ΦT0*/
CG_SetSCOUTSrc(CG_SCOUT_SRC_PHIT0);

/* enable high-speed oscillation*/
CG_SetFosc(ENABLE);

/* enable low-speed oscillation*/
CG_SetFs(ENABLE);

/* set low power consumption mode Sleep*/
CG_SetSTBYMode (CG_STBY_MODE_SLEEP);

/* set high-speed oscillation to be enabled after releasing stop mode*/
CG_SetExitStopModeFosc(ENABLE);

/* set pin status in stop mode to "active"*/
CG_SetPinStateInStopMode(ENABLE);

/* set up pll and wait 200us for pll to warm up , set fc source to fpll*/
CG_EnableClkMulCircuit();   /*this module refer to   Programming Example */

/* set system clock to fgear */
CG_SetFsysSrc(CG_FSYS_SRC_FGEAR);
```

[Sample module CG_ENABLEClkMulCircuit()]

The following codes introduce how to enable multiple clock circuit

```
#define C200US   0x0096U            /* @12M Oscillator */

Result CG_EnableClkMulCircuit(void)
{
    Result retval = ERROR;
    WorkState st = BUSY;

    CG_SetClkMulTimes(CG_MUL_4TIMES);
    retval = CG_SetPLL(ENABLE);

    if (retval == SUCCESS) {
        /*set warm up time to about 200us*/
        CG_SetWarmUpTime(CG_WARM_UP_SRC_X1, C200US);
        CG_StartWarmUp();
```

```
            /*wait warm up to end */
            do {
                  st = CG_GetWarmUpState();
            } while (st != DONE);
            retval = CG_SetFcSrc(CG_FC_SRC_FPLL);

      } else {
            /*Do nothing */
      }
      return retval;
}
```

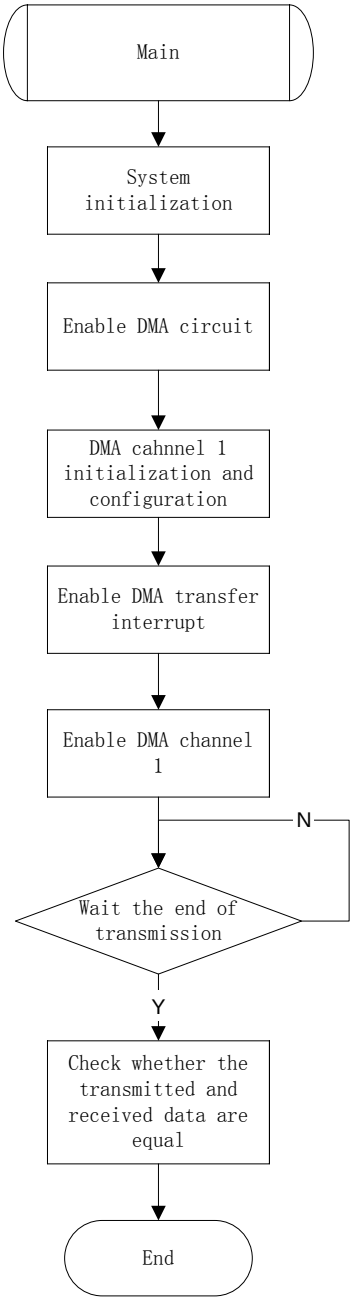
7-5 DMAC

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

The example includes:

1. DMAC initialization.
2. Transfer data from memory to memory via DMAC.

- **Flowchart**



- **Code and Explanation for the Example**

At first, create a DMAC_InitTypeDef structure and fill all the data fields. For example,

```
DMAC_InitTypeDef DMAC_InitStruct;

DMAC_InitStruct.TxDirection = DMAC_MEMORY_TO_MEMORY;
DMAC_InitStruct.SrcAddr = (uint32_t) SRC_Buffer;
DMAC_InitStruct.DstAddr = (uint32_t) DST_Buffer;
DMAC_InitStruct.SrcIncrementState = ENABLE;
DMAC_InitStruct.DstIncrementState = ENABLE;
DMAC_InitStruct.SrcBitWidth = DMAC_WORD;
DMAC_InitStruct.DstBitWidth = DMAC_WORD;
DMAC_InitStruct.SrcBurstSize = DMAC_4_BEATS;
DMAC_InitStruct.DstBurstSize = DMAC_4_BEATS;
DMAC_InitStruct.TxSize = BUFFER_SIZE;
DMAC_InitStruct.TxINT = ENABLE;
```

Then enable DMA circuit, initialize and configure DMA channel. Then enable DMA transfer interrupt and DMA channel.

```
DMAC_Enable();
DMAC_Init(myChannel, &DMAC_InitStruct);
DMAC_TxINTConfig(myChannel, DMAC_INT_TX_END, ENABLE);
DMAC_SetDMAChannel(myChannel, ENABLE);
```

After the setting above, wait for the end of transmission.

```
while (TxEndFlag != DONE) {
        /* Do nothing */
}
```

The flag of transfer end will be set in ISR of DMAC transfer end interrupt.

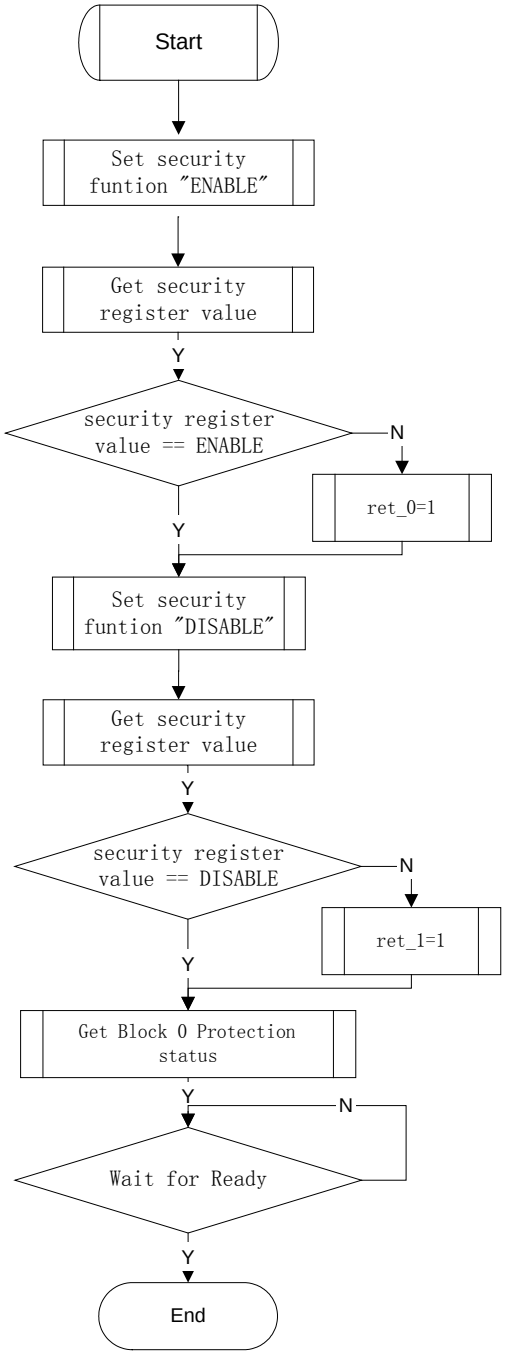
7-6 FLASH

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

The example includes:

1. Read/Write flash registers.

- **Flowchart**



- **Code and Explanation for the Example**

The sample source is shown as below:

```
#include "tmpm36x_fc.h"

int main(void)
{
	FunctionalState tmp_value_sec0, tmp_value_sec1, tmp_value_block;
	uint32_t ret_0 = 0U, ret_1 = 0U,ret_2 = 0U;

	while (1) {
		/* Set security function "ENABLE" */
		FC_SetSecurityBit(ENABLE);

		/* Get security register value */
		tmp_value_sec0 = FC_GetSecurityBit();
		if (tmp_value_sec0 != ENABLE) {
			ret_0 = 1U;
		}

		/* Set security function "DISABLE" */
		FC_SetSecurityBit(DISABLE);

		/* Get security register value */
		tmp_value_sec1 = FC_GetSecurityBit();
		if (tmp_value_sec1 != DISABLE) {
			ret_1 = 1U;
		}

		/* Get Block 0 Protection status */
		tmp_value_block = FC_GetBlockProtectState(FC_BLOCK_0);

		if (tmp_value_block != DISABLE) {
			ret_2 = 1U;
		}
		/* Wait for Ready */
		while (FC_GetBusyState() != DONE) {
		/* Do nothing */
		};
	}
}
```

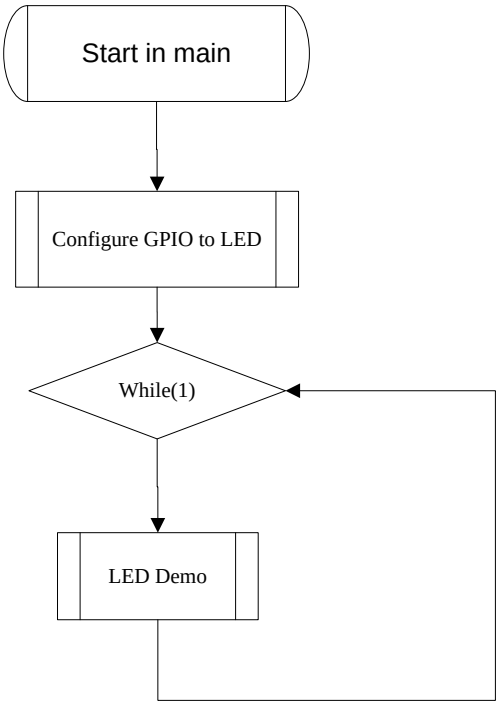
7-7 GPIO

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

The example includes:

1. GPIO initialization.
2. Write data to GPIO.

- **Flowchart**



- **Code and Explanation for the Example**

At first, use GPIO_Init() to configure GPIO to LED. Create a GPIO_InitTypeDef struct, then fill all the data fields. For example,

```
GPIO_InitTypeDef led_io;
led_io.IOMode = GPIO_OUTPUT_MODE;
led_io.PullUp = GPIO_PULLUP_ENABLE;
led_io.OpenDrain = GPIO_OPEN_DRAIN_NONE;
```

Then call GPIO_Init() function to initialize LED.

```
GPIO_Init(GPIO_PM, GPIO_BIT_4, &led_io);
```

In the While process, do the LED demo: LED on and LED off.
Set LED on by Using GPIO_WriteData () to write 1 to GPIO DATA register.

```
GPIO_WriteDataBit(GPIO_PM, GPIO_BIT_4, GPIO_BIT_VALUE_0);
```

Set LED off by Using GPIO_WriteData () to write 1 to GPIO DATA register.

```
GPIO_WriteDataBit(GPIO_PM, GPIO_BIT_4, GPIO_BIT_VALUE_1);
```

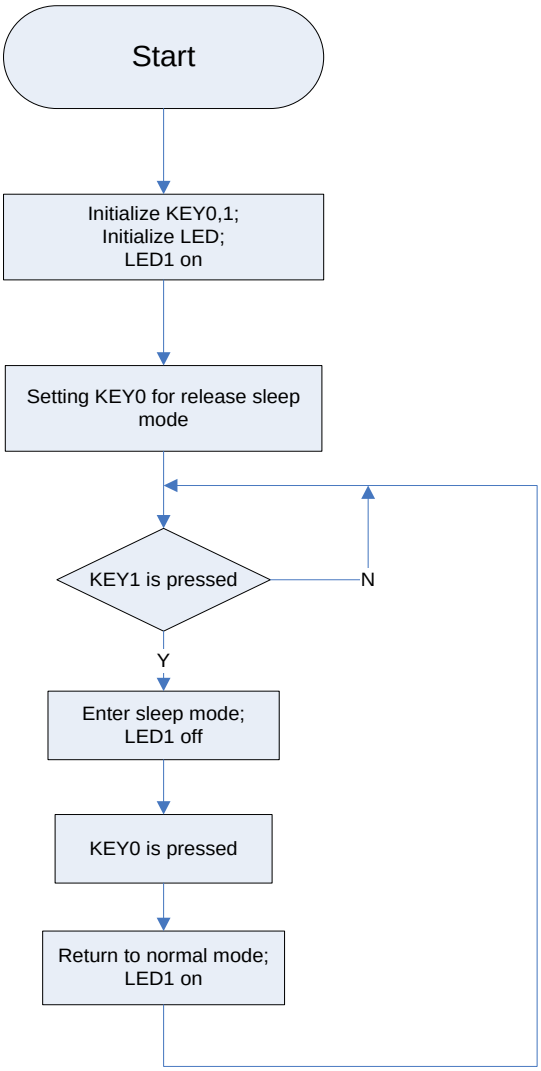
7-8 KWUP

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

The example includes:

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

- **Flowchart**



- **Code and Explanation for the Example**

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

```
    KWUP_SettingTypeDef keySetting;
    KWUP_PortStatus portStatus;
    SystemInit();

    /* set PJ4,PJ5 to Key0,1 */
    TSB_PJ_FR2_PJ4F2 = 1U;
    TSB_PJ_PUP_PJ4UP = 1U;
    TSB_PJ_IE_PJ4IE = 1U;

    TSB_PJ_FR2_PJ5F2 = 1U;
    TSB_PJ_PUP_PJ5UP = 1U;
    TSB_PJ_IE_PJ5IE = 1U;

    /* Set PM4 to Output Port */
    LED_Configuration();
    LED_On(LED1);

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

    keySetting.KeyN = KWUP_INPUT_0;
    keySetting.PullUpCtrl = KWUP_PUP_CTRL_BY_DYNAMIC;
    keySetting.ActiveState = KWUP_ACTIVE_BY_RISING_EDGE;
    keySetting.INTNewState = ENABLE;
    KWUP_SetConfig(&keySetting);

    KWUP_SetPullUpConfig(KWUP_CYCLES_4_FS, KWUP_CYCLES_256_FS);

    KWUP_ClearINTReq();
```

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

```
    /* enable stop mode release interrupt to KWUP */
    TSB_CG->IMCGF = 0x1000U;
    TSB_CG->IMCGF += 0x010U;

    do {
        portStatus = KWUP_GetPortStatus();

        if (portStatus.Bit.Key1 == 0U) {        /* press key(PJ5) */
            LED_Off(LED1);
            /* enter low power mode */
            __WFI();

            /* press KEY(PJ4) to release low power mode */

            LED_On(LED1);
        }
    }
    while (1);
```

7-9 RC

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

The example includes:

1. Read/Write ram control registers.

- **Code and Explanation for the Example**

The sample source is shown as below:

```
#include "tmpm36x_rc.h"

int main(void)
{
	RC_RAMWait tmp_value;
	uint32_t ret_0 = 0U;
	while (1) {
		/* Set the value of RCWAIT register */
		RC_SetRAMWait(RC_RAMWAIT_1);

		/* Get the value of RCWAIT register */
		tmp_value = RC_GetRAMWait();
		if (tmp_value != RC_RAMWAIT_1) {
			ret_0 = 1U;
		}
	}
}
```

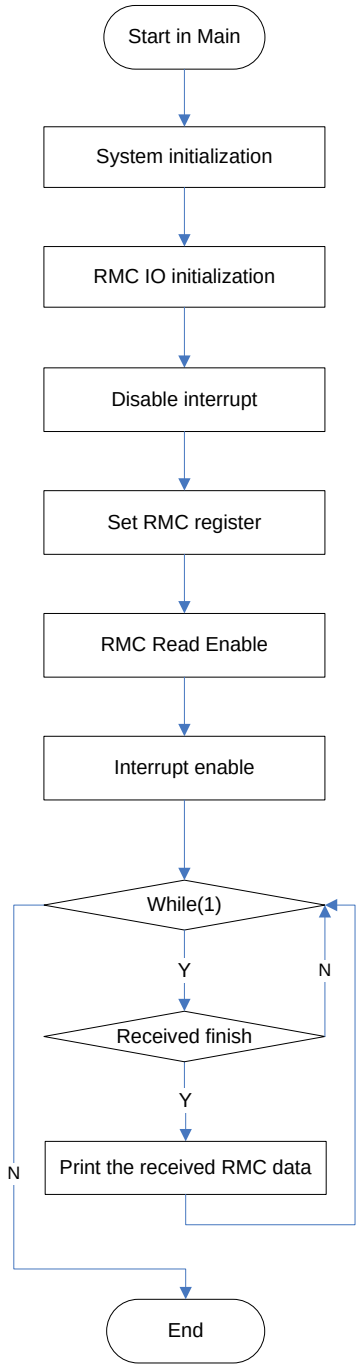
7-10 RMC

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

The example includes:

1. RMC IO initialization.
2. Receive RMC data.

- **Flowchart**



- **Code and Explanation for the Example**

At first, in main(), create a myRMC structure, then fill all the data fields.
For example

```
RMC_InitTypeDef myRMC;

myRMC.LeaderPara.MaxCycle = RMC_MAX_CYCLE;
myRMC.LeaderPara.MinCycle = RMC_MIN_CYCLE;
myRMC.LeaderPara.MaxLowWidth = RMC_MAX_LOW_WIDTH;
myRMC.LeaderPara.MinLowWidth = RMC_MIN_LOW_WIDTH;
myRMC.LeaderPara.LeaderDetectionState = ENABLE;
myRMC.LeaderPara.LeaderINTState = DISABLE;
myRMC.FallingEdgeINTState = DISABLE;
myRMC.SignalRxMethod = RMC_RX_IN_CYCLE_METHOD;
myRMC.LowWidth = RMC_TRG_LOW_WIDTH;
myRMC.MaxDataBitCycle = RMC_TRG_MAX_DATA_BIT_CYCLE;
myRMC.LargerThreshold = RMC_LARGER_THRESHOLD;
myRMC.SmallerThreshold = RMC_SMALLER_THRESHOLD;
myRMC.InputSignalReversedState = DISABLE;
myRMC.NoiseCancellationTime = RMC_NOISE_CANCELLATION_TIME;
```

Then, enable and initialize the RMC channel 0.

```
RMC_Enable(TSB_RMC0);
```

Initial the specified RMC channel with the structure which includes the basic RMC configuration.

```
RMC_Init(TSB_RMC0, &myRMC);
```

Enable the reception of the specified RMC0 channel.

```
RMC_SetRxCtrl(TSB_RMC0, ENABLE);
```

In interrupt INTRMCRX0_IRQHandler ():
Get the interrupt factor for the specified RMC0 channel.

```
RMC_INTFactor myRMC_INTFactor;
myRMC_INTFactor = RMC_GetINTFactor(TSB_RMC0);
```

Get the leader detection result for the specified RMC0 channel.

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

Get the received data from the specified RMC0 channel.

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

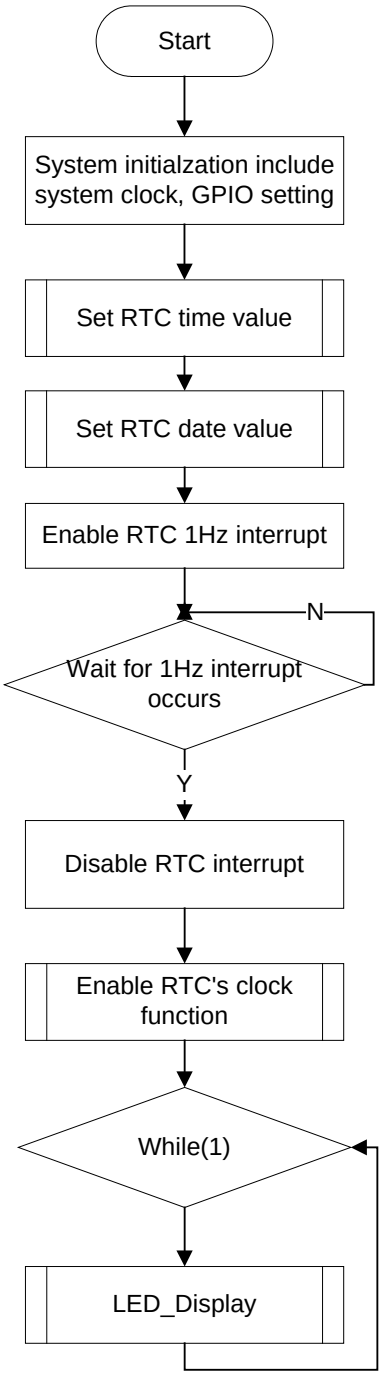
7-11 RTC

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

The example includes:

1. RMC initialization.
2. Get RTC data and time.

- **Flowchart**



- **Code and Explanation for the Example**

At first, create RTC_DateTypeDef struct and RTC_TimeTypeDef struct, and then fill in all the data fields. For example,

```
RTC_DateTypeDef DateStruct;
RTC_TimeTypeDef TimeStruct;
DateStruct.LeapYear = RTC_LEAP_YEAR_2;
DateStruct.Year = (uint8_t)10;
DateStruct.Month = (uint8_t)12;
DateStruct.Date = (uint8_t)31;
DateStruct.Day = RTC_FRI;
TimeStruct.HourMode = RTC_12_HOUR_MODE;
TimeStruct.Hour = (uint8_t)11;
TimeStruct.AmPm = RTC_PM_MODE;
TimeStruct.Min = (uint8_t)59;
TimeStruct.Sec = (uint8_t)50;
```

After the setting above, Set RTC date and time value, and then enable RTC 1HZ interrupt.

```
RTC_SetTimeValue(&TimeStruct);
RTC_SetDateValue(&DateStruct);
__disable_irq();
/* enable RTC interrupt */
NVIC_ClearPendingIRQ(INTRTC_IRQn);
TSB_CG->IMCGF = 0x00000020;
TSB_CG->IMCGF = 0x00000021;
NVIC_EnableIRQ(INTRTC_IRQn);
/* Enable 1Hz interrupt */
RTC_SetAlarmOutput(RTC_PULSE_1_HZ);
/* Enable RTCINT */
RTC_SetRTCINT(ENABLE);
__enable_irq();
```

Then waiting for 1HZ interrupt occurs.

```
/* waiting for RTC register set finish */
while(fRTCSetting_ok != 1);
fRTCSetting_ok = 0;
```

At last, disable RTC interrupt and enable RTC clock function.

```
/* Disable RTCINT */
__disable_irq();
RTC_SetRTCINT(DISABLE);
NVIC_DisableIRQ(INTRTC_IRQn);
__enable_irq();
/* Enable RTC Clock function */
RTC_EnableClock();
```

Use RTC_GetYear() / RTC_GetMonth() / RTC_GetDate() / RTC_GetDay() functions to get the RTC date value.

```
uint8_t Year = 0U;
uint8_t Month = 0U;
uint8_t Date = 0U;
uint8_t Day = 0U;
Year = RTC_GetYear();
Month = RTC_GetMonth();
Date = RTC_GetDate(RTC_CLOCK_MODE);
Day = RTC_GetDay(RTC_CLOCK_MODE);
```

Or call one function as below:

```
RTC_DateTypeDef DateStruct;
RTC_GetDateValue(&DateStruct);
```

Use RTC_GetHourMode() / RTC_GetHour() / RTC_GetMin() / RTC_GetSec() to get the RTC time value.

```
uint8_t HourMode = 0U;
uint8_t Hour = 0U;
uint8_t Min = 0U;
uint8_t Sec = 0U;
HourMode = RTC_GetHourMode();
Hour = RTC_GetHour(RTC_CLOCK_MODE);
Min = RTC_GetMin(RTC_CLOCK_MODE);
Sec = RTC_GetSec();
```

Or call one function as below:

```
RTC_TimeTypeDef TimeStruct;
RTC_GetTimeValue(&TimeStruct);
```

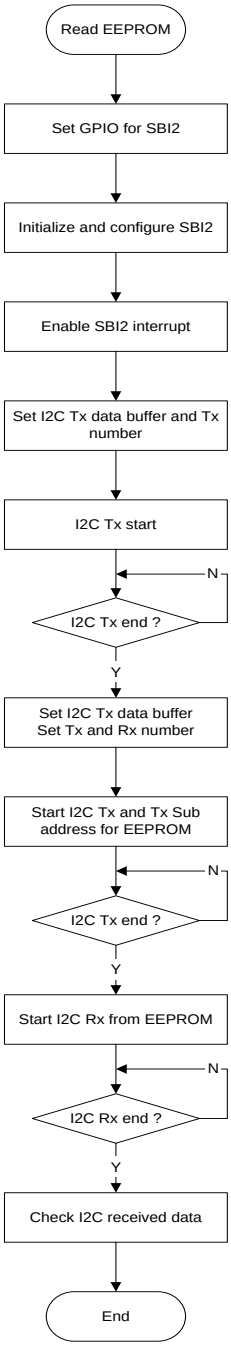
7-12 SBI

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

The example includes:

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

- **Flowchart**



- **Code and Explanation for the Example**

At first, create a SBI_InitI2CTypeDef structure and fill all the data fields. For example,

```
myI2C.I2CSelfAddr = SELFADDR;
myI2C.I2CDataLen = SBI_I2C_DATA_LEN_8;
myI2C.I2CACKState = ENABLE;
myI2C.I2CClkDiv = SBI_I2C_CLK_DIV_328;
```

Then enable, initialize and configure SBI2.

```
SBI_Enable(TSB_SBI2);
```

```
SBI_SWReset(TSB_SBI2);
SBI_InitI2C(TSB_SBI2, &myI2C);
NVIC_EnableIRQ(INTSBI2_IRQn);
```

Following is the procedure of write operation.
Initial I2C Tx data buffer and counter, after checked the I2C bus is free, set slave address and direction of transfer. Then send start condition.

```
do{
    i2c_state = SBI_GetI2CState(TSB_SBI2);
} while (i2c_state.Bit.BusState);;
SBI_SetSendData(TSB_SBI2, SLAVEADDR | SBI_I2C_SEND);
SBI_GenerateI2CStart(TSB_SBI2);
```

Then in i2C interrupt: read the I2C bus state; send the next data, and check I2C data transfer end, stop I2C.

```
SBIx = TSB_SBI2;
sbi_sr = SBI_GetI2CState(SBIx)
……
SBI_SetSendData(SBIx, gI2C_TxData[gI2C_WCnt]);
……
SBI_GenerateI2CStop(SBIx);
```

Now data in I2C Tx buffer has been written into the EEPROM.

Following is the procedure of read operation.
Initial I2C Tx data buffer, Tx counter and Rx counter. After checked the I2C bus is free, set slave address and direction of transfer. Then send start condition.

```
do{
    i2c_state = SBI_GetI2CState(TSB_SBI2);
} while (i2c_state.Bit.BusState);
SBI_SetSendData(TSB_SBI2, SLAVEADDR | SBI_I2C_SEND);
SBI_GenerateI2CStart(TSB_SBI2);
```

Then in i2C interrupt: Read the I2C bus state, send the next data, and check I2C data transfer end, stop I2C (The sub address for EEPROM is sent in this process).

```
SBIx = TSB_SBI2;
sbi_sr = SBI_GetI2CState(SBIx)
……
SBI_SetSendData(SBIx, gI2C_TxData[gI2C_WCnt]);
……
SBI_GenerateI2CStop(SBIx);
```

After The sub address for EEPROM is sent, read the data of EEPROM. The first interrupt after I2C start for send slave address and receive direction, dummy read the I2C data buffer register to release PIN.

```
SBI_SetSendData(TSB_SBI2, SLAVEADDR | SBI_I2C_RECEIVE);
SBI_GenerateI2CStart(TSB_SBI2);
```

(The first interrupt after I2C start for send slave address and receive direction, dummy read the I2C data buffer register to release PIN.)

Then in i2C interrupt: Read the I2C bus state and data buffer, setting for receive the next I2C data(After Rx the data second to last, setting Not generate ACK for next data Rx end; after Rx the last data, setting I2C bit number to 1 for only generate 1 clock for stop condition).

```
SBIx = TSB_SBI2;
sbi_sr = SBI_GetI2CState(SBIx)
……
tmp = SBI_GetReceiveData(SBIx);
……
SBI_SetI2CBitNum(SBIx, SBI_I2C_DATA_LEN_1);
```

```
……
SBI_SetI2CACK(TSB_SBI2,DISABLE);
```

Now data from EEPROM has been read and save to I2C Rx data buffer.

*Note: For I2C data ACK mode is be set to “No-ACK” after receiving data finishes. Please re-set the ACK mode and other parameter for next I2C process.

Since the write address and the read address in EEPROM are the same. So the result of transfer can be check by comparing the data received with the data sent.

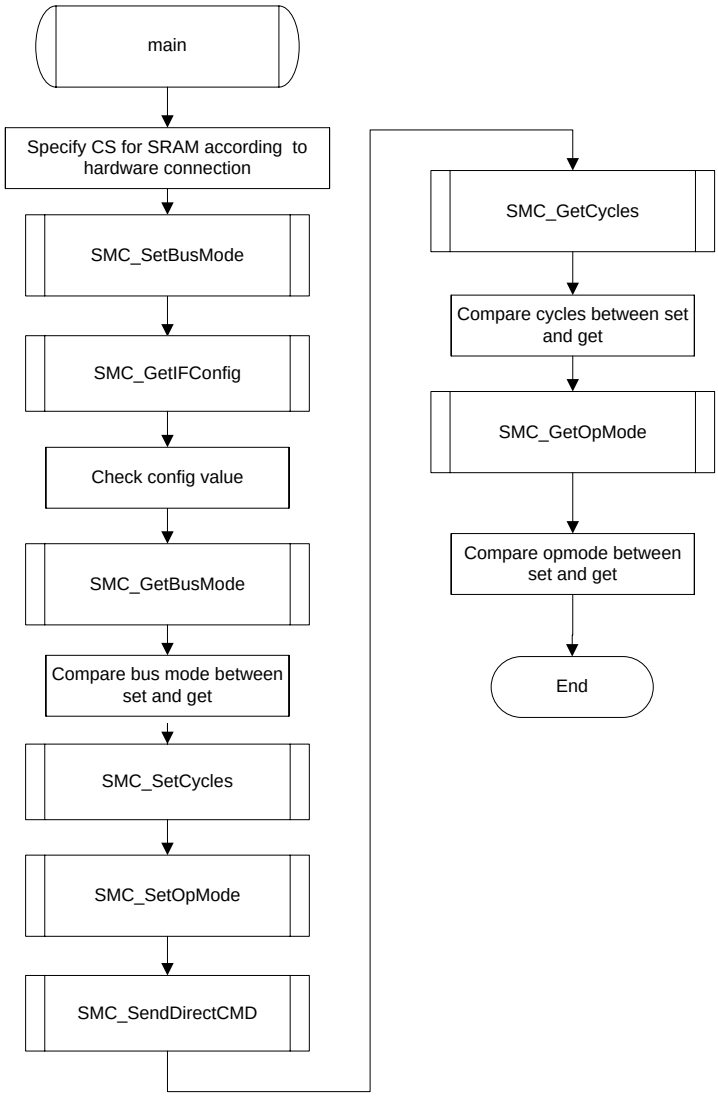
7-13 SMC

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

The example includes:

1. SBI configuration.

- **Flowchart**



- **Code and Explanation for the Example**

Static memory: 16-bit SRAM, and multiplex bus mode.

Set bus mode at first according to connection with external memory. For example:

```
uint8_t bus_w = 0xFFU;
bus_w = SMC_BUS_MULTIPLEX;
```

```
SMC_SetBusMode(bus_w);
```

Then read out interface configuration. For example:

```
SMC_GetIFConfig(&config);
```

Then send direct command to static memory after setting cycles and opmode according to SRAM A.C specifications so that these settings are validated. For example:

```
/* Set cycles time according to AC timing of SRAM datasheet, base clock: SMCCLK */
cycles_w.RC_Time = SMC_READ_CYCLE_TIME_5;
cycles_w.WC_Time = SMC_WRITE_CYCLE_TIME_5;
cycles_w.CEOE_Time = SMC_CEOE_DELAY_CYCLE_TIME_1;
cycles_w.WP_Time = SMC_WE_PULSE_CYCLE_TIME_1;
cycles_w.PC_Time = SMC_PAGE_CYCLE_TIME_5;
cycles_w.TR_Time = SMC_TURN_AROUND_CYCLE_TIME_1;
SMC_SetCycles(&cycles_w);

opmode_w.BusWidth = SMC_DATA_BUS_16BIT;
opmode_w.ReadBurstLen = SMC_READ_BURST_4BEAT;
opmode_w.ALE = ENABLE;          /* ALE must be enabled in multiplex bus mode */

/* Make sure SMC_SetCycles and SMC_SetOpMode are performed before
SMC_SendDirectCMD */
cmd.CmdType = SMC_CMD_UPDATEREGS;
cmd.ChipSelect = SMC_CS2;
SMC_SendDirectCMD(&cmd);
```

Then read back cycles and opmode to verify. For example:

```
SMC_GetCycles(chip, &cycles_r);
SMC_GetOpMode(chip, &opmode_r);
```

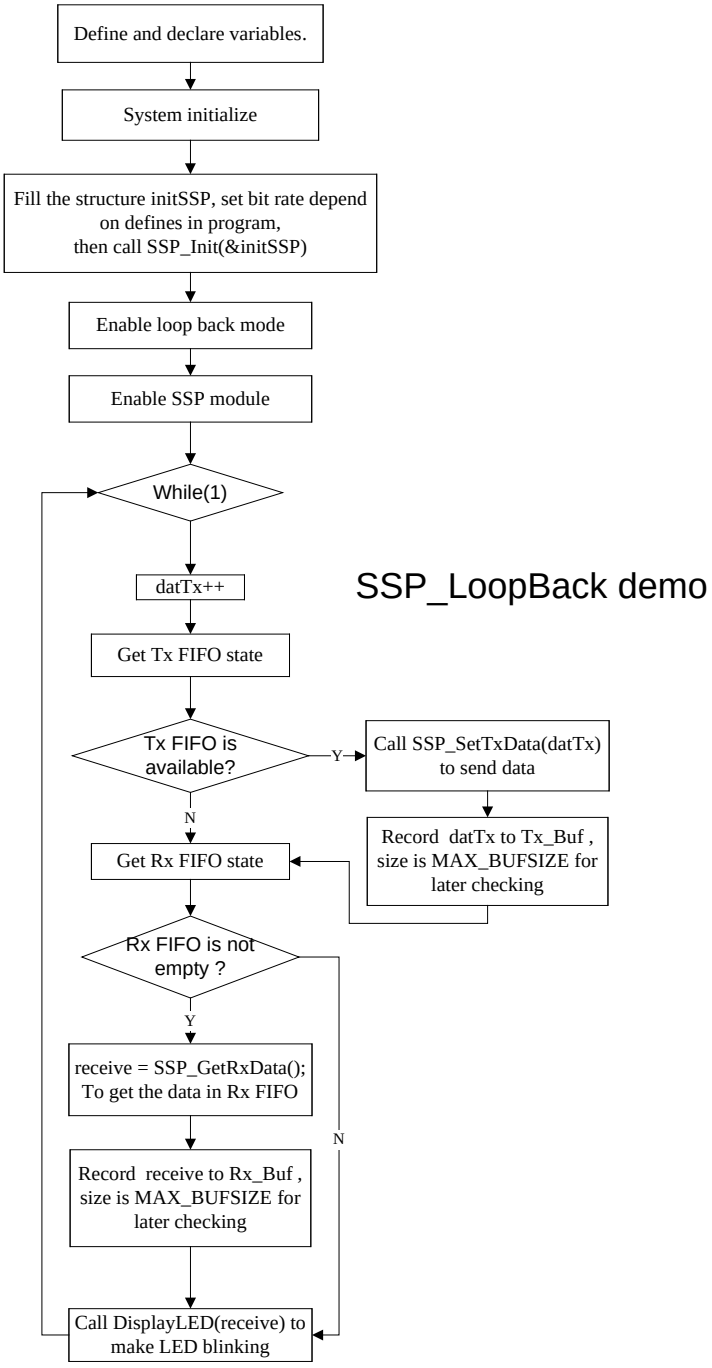
7-14 SSP

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

The example includes:

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

- **Flowchart**



- **Code and Explanation for the Example**

At first, create SSP_InitTypeDef struct and other variables.

```
SSP_InitTypeDef initSSP;

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

uint16_t receive = 0U;
uint16_t Rx_Buf[MAX_BUFSIZE];
uint16_t Tx_Buf[MAX_BUFSIZE];
```

At second, initialize system.

```
SystemInit();

 /* Enable the LED on board to display some info */
LED_Configuration();
```

At third, fill in all the data fields of initSSP. For example,

```
 /* 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;
```

After the setting above, call SSP_Init().

```
SSP_Init(&initSSP);
```

Enable loop back mode for self test .

```
SSP_SetLoopBackMode(ENABLE);
```

Enable SSP module.

```
SSP_Enable();
```

At last, send data when Tx FIFO is available, and receive data when Rx FIFO isn't empty.

```
   while (1) {

        datTx++;

    /* send data if Tx FIFO is available */
        fifoState = SSP_GetFIFOState(SSP_TX);
        if ((fifoState == SSP_FIFO_EMPTY) || (fifoState == SSP_FIFO_NORMAL)) {
```

```
                SSP_SetTxData(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(SSP_RX);
            if ((fifoState == SSP_FIFO_FULL) || (fifoState == SSP_FIFO_NORMAL)) {
                receive = SSP_GetRxData();
                if (cntRx < MAX_BUFSIZE) {
                    Rx_Buf[cntRx] = receive;
                    cntRx++;
                } else {
                   /* Place a break point here to check if receive data is right. */
                    __NOP();
                }

            } else {
                /* do nothing */
            }

            DisplayLED(receive);
        }
```

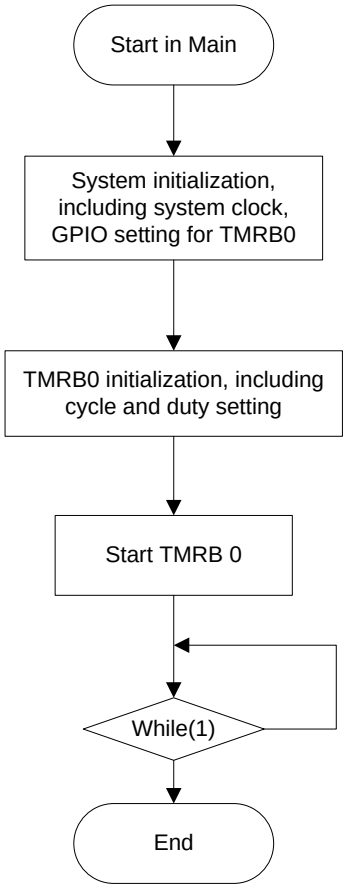
7-15 TMRB

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

The example includes:

1. TMRB initialization.
2. General timer for 1ms.

- **Flowchart**



- **Code and Explanation for the Example**

At first, in main(), create a TMRB_InitTypeDef structure, then fill all the data fields. For example,

```
TMRB_InitTypeDef myTB;
myTB.Mode = TMRB_INTERVAL_TIMER;
myTB.ClkDiv = TMRB_CLK_DIV_8;
myTB.TrailingTiming = TMRB_1ms;          /* Specific value depends on system clock */
myTB.UpCntCtrl = TMRB_AUTO_CLEAR;
myTB.LeadingTiming = TMRB_1ms / 2;      /* Specific value depends on system clock */
```

Then, enable and initialize the TMRB channel 0.

```
TMRB_Enable(TSB_TB0);
TMRB_Init(TSB_TB0, &myTB);
```

At the end of main(), enable INTTB0, and then start the counter of TMRB channel 0.

```
TMRB_SetRunState(TSB_TB0, TMRB_RUN);
```

7-16 SIO/UART

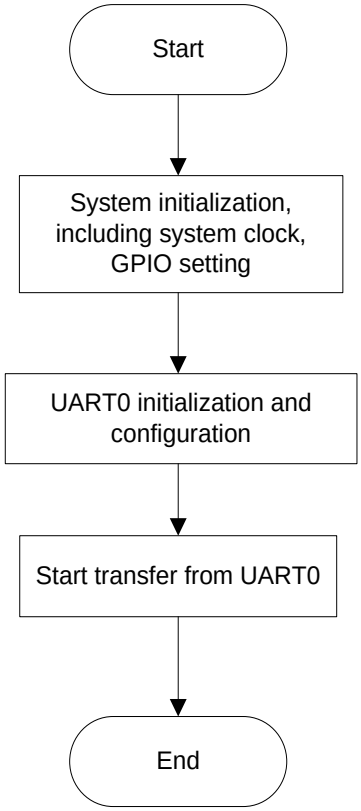
7-16-1 Example: UART transfer demo

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

The example includes:

1. UART configuration and initialization.
2. UART0 TX process.
3. Use UART TX interrupts to send data.

- **Flowchart**



- **Code and Explanation for the Example**

At first, create a UART_InitTypeDef structure and fill all the data fields. For example,

```
UART_InitTypeDef myUART;
myUART.BaudRate = 115200;
myUART.DataBits = UART_DATA_BITS_8;
myUART.StopBits = UART_STOP_BITS_1;
myUART.Parity = UART_NO_PARITY;
myUART.Mode = UART_ENABLE_RX | UART_ENABLE_TX;
myUART.FlowCtrl = UART_NONE_FLOW_CTRL;
```

Then enable, initialize and configure UART channels.

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

After the setting above, enable SC0 TX interrupt. Then start sending data.

```
UART_SetTxData(UART0, TxBuf[0]);
```

Here TxBuf is a character array.

The rest process of data flow can be finished in ISR of SC0 TX interrupt.

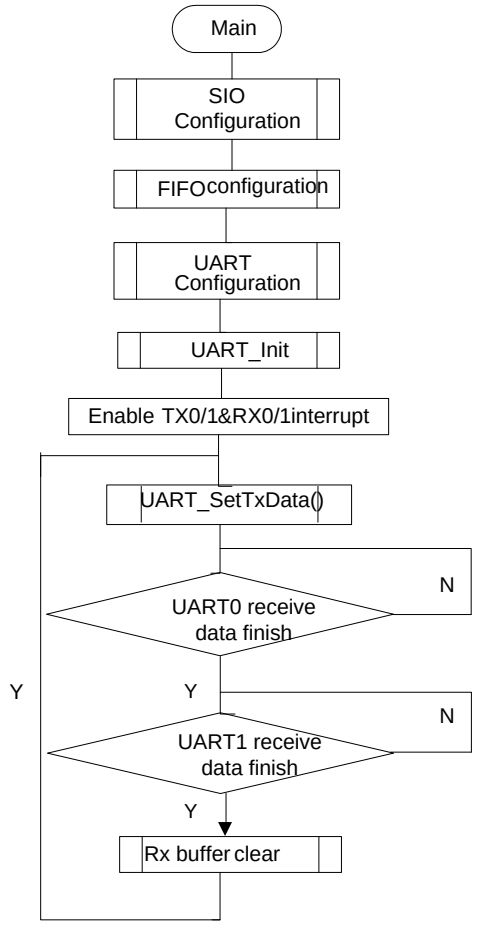
7-16-2 Example: UART FIFO

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 UART0 and UART1.

```
void SIO_Configuration(TSB_SC_TypeDef * SCx)
{
      if (SCx == TSB_SC0) {
            TSB_PE->CR |= GPIO_BIT_4;
            TSB_PE->FR2 |= GPIO_BIT_4;
            TSB_PE->FR2 |= GPIO_BIT_5;
            TSB_PE->IE |= GPIO_BIT_5;
      } else if (SCx == TSB_SC1) {
            TSB_PL->CR |= GPIO_BIT_4;
            TSB_PL->FR1 |= GPIO_BIT_4;
            TSB_PL->FR1 |= GPIO_BIT_5;
            TSB_PL->IE |= GPIO_BIT_5;
      } }
```

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

```
      UART_InitTypeDef myUART;

    /* configure SIO0 for reception */
    UART_Enable(UART_RETARGET);
```

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

Use UART peripheral drivers to enable and initialize UART0/1 channels.

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

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

FIFO configuration.

```
    UART_RxFIFOByteSel(UART0,UART_RXFIFO_RXFLEVEL);
    UART_RxFIFOByteSel(UART1,UART_RXFIFO_RXFLEVEL);

    UART_TxFIFOINTCtrl(UART0,ENABLE);
    UART_TxFIFOINTCtrl(UART1,ENABLE);

    UART_RxFIFOINTCtrl(UART0,ENABLE);
    UART_RxFIFOINTCtrl(UART1,ENABLE);

    UART_TRxAutoDisable(UART0,UART_RXTXCNT_AUTODISABLE);
    UART_TRxAutoDisable(UART1,UART_RXTXCNT_AUTODISABLE);

    UART_FIFOConfig(UART0,ENABLE);
    UART_FIFOConfig(UART1,ENABLE);

    UART_RxFIFOFillLevel(UART0, UART_RXFIFO4B_FLEVLE_4_2B);
    UART_RxFIFOFillLevel(UART1, UART_RXFIFO4B_FLEVLE_4_2B);

    UART_RxFIFOINTSel(UART0,UART_RFIS_REACH_EXCEED_FLEVEL);
    UART_RxFIFOINTSel(UART1,UART_RFIS_REACH_EXCEED_FLEVEL);

    UART_RxFIFOClear(UART0);
    UART_RxFIFOClear(UART1);

    UART_TxFIFOFillLevel(UART0, UART_TXFIFO4B_FLEVLE_0_0B);
    UART_TxFIFOFillLevel(UART1, UART_TXFIFO4B_FLEVLE_0_0B);

    UART_TxFIFOINTSel(UART0,UART_TFIS_REACH_NOREACH_FLEVEL);
    UART_TxFIFOINTSel(UART1,UART_TFIS_REACH_NOREACH_FLEVEL);

    UART_TxFIFOClear(UART0);
    UART_TxFIFOClear(UART1);
```

After above setting, enable UART0/1 interrupt.

```
    NVIC_EnableIRQ(INTTX0_IRQn);
    NVIC_EnableIRQ(INTRX1_IRQn);

    NVIC_EnableIRQ(INTTX1_IRQn);
    NVIC_EnableIRQ(INTRX0_IRQn);
```

The rest process of data flow is finished in ISR of UART0/1 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);
	}
}
```

UART1 TX interrupt routine:

```
void INTTX1_IRQHandler(void)
{
	volatile UART_Err err;

	if (TxCounter1 < NumToBeTx1) {
		UART_SetTxData(UART1, TxBuffer1[TxCounter1++]);
	} else {
		err = UART_GetErrState(UART1);
	}
}
```

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);
	}
}
```

UART1 RX interrupt routine:

```
void INTRX1_IRQHandler(void)
{
	volatile UART_Err err;

	err = UART_GetErrState(UART1);
	if (UART_NO_ERR == err) {
		RxBuffer1[RxCounter1++] = (uint8_t) UART_GetRxData(UART1);
	}
}
```

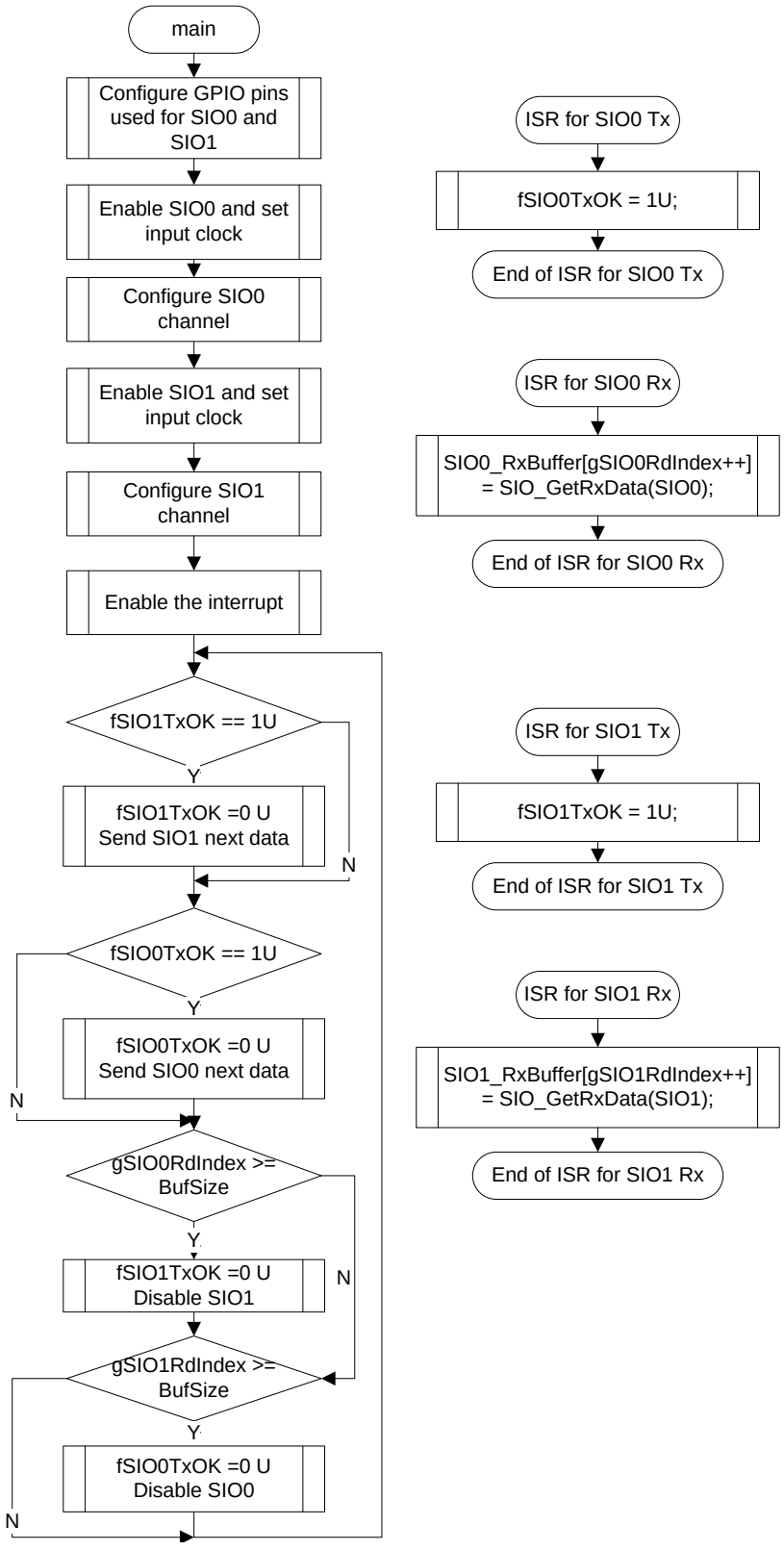
7-16-3 Example: SIO

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

The example includes:
1. Basic setup operation of SIO
2. The data transfer between SIO0 and SIO1

3. SIO interrupt of Tx and Rx

- **Flowchart**



- **Code and Explanation for the Example**

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

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

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

/*initialize the SIO0 struct */
SIO0_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO0_Init.IntervalTime = SIO_SINT_TIME_SCLK_1;
SIO0_Init.TransferMode = SIO_TRANSFER_FULLDPX;
SIO0_Init.TransferDir = SIO_LSB_FRIST;
SIO0_Init.Mode = SIO_ENABLE_TX | SIO_ENABLE_RX;
SIO0_Init.DoubleBuffer = SIO_WBUF_ENABLE;
SIO0_Init.BaudRateClock = SIO_BR_CLOCK_T1;
SIO0_Init.Divider = SIO_BR_DIVIDER_2;
```

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.IntervalTime = SIO_SINT_TIME_SCLK_1;
SIO1_Init.TransferMode = SIO_TRANSFER_FULLDPX;
SIO1_Init.TransferDir = SIO_LSB_FRIST;
SIO1_Init.Mode = SIO_ENABLE_TX | SIO_ENABLE_RX;
SIO1_Init.DoubleBuffer = SIO_WBUF_ENABLE;

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

Enable the interrupt of SIO TX, RX.

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

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

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

```
while (1) {
        /* SIO1 send data from TXD1*/
        if (fSIO1TxOK == 1U) {
            fSIO1TxOK = 0U;
            SIO_SetTxData(SIO1, SIO1_TxBuffer[gSIO1WrIndex++]);
        } else {
            /*Do Nothing */
        }
        /* SIO0 send data from TXD0*/
        if (fSIO0TxOK == 1U) {
            fSIO0TxOK = 0U;
            SIO_SetTxData(SIO0, SIO0_TxBuffer[gSIO0WrIndex++]);
```

```
        } else {
            /*Do Nothing */
        }

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

In ISR of SIO0 Tx, set transfer is ok.

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

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

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

In ISR of SIO1 Tx, set transfer is ok.

```
void INTTX1_IRQHandler(void)
{
    fSIO1TxOK = 1U;
}
```

In ISR of SIO1 Rx, get the receive data from RX buffer.

```
void INTRX1_IRQHandler(void)
{
    SIO0_RxBuffer[gSIO1RdIndex++] = SIO_GetRxData(SIO1);
}
```

7-17 WDT

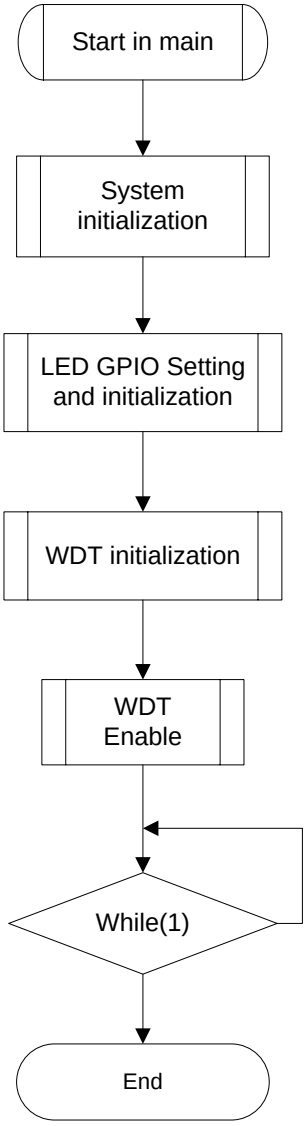
This is a simple example based on the TX03 Peripheral Driver (WDT, GPIO).

The example includes:

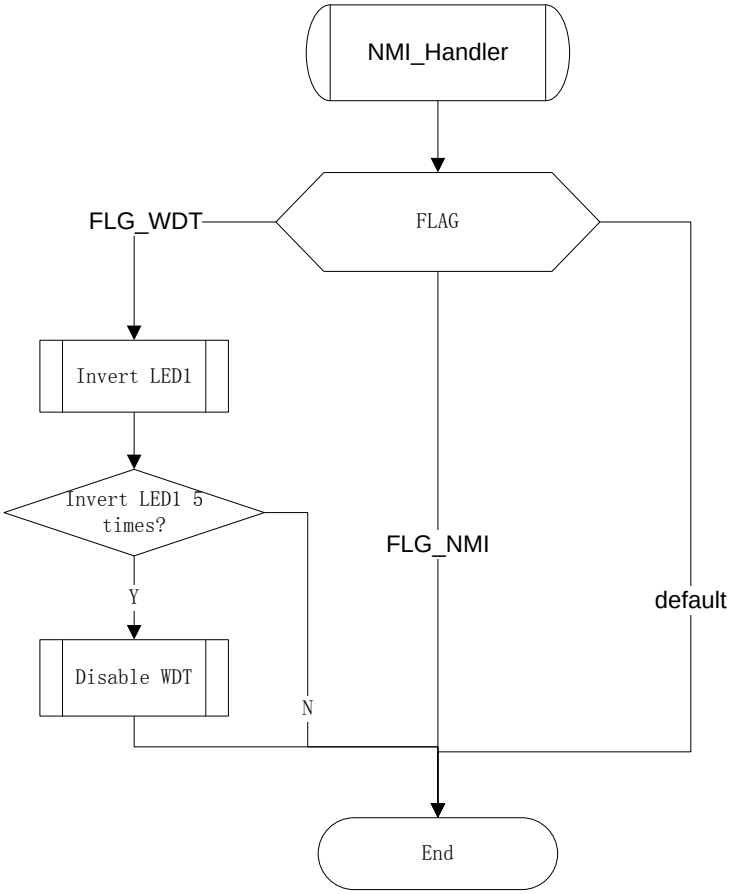
1. Generate a non-makeable interrupt by WDT counter overflow.
2. Invert LED1.
3. Invert LED1 5 times, disable WDT.

- **Flowchart**

[Main Function]



[NMI_Handler Function]



- **Code and Explanation for the Example**

At first, in main(), create a WDT_InitTypeDef struct and fill in all the data fields.

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

Initialize and configure Watch Dog timer then enable it.

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

Wait for the NMI interrupt to occur.

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