TOSHIBA

TX03 ペリフェラルドライバ使用例 (TMPM311)

第 1.000 版

2017 年 9 月

東芝デバイス＆ストレージ株式会社

CMDR-M311UE-00xJ

本製品取り扱い上のお願い

- ソフトウエア使用権許諾契約書の同意無しに使用しないで下さい。

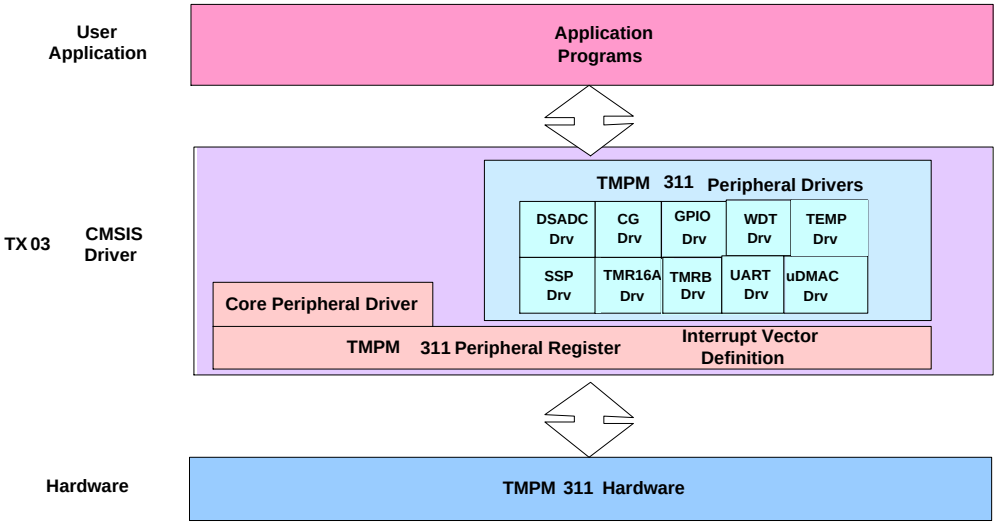
目次

1 はしがき

本サンプルプログラムは、東芝製マイコンTMPM311用です。各サンプルプログラムは、主なMCU内蔵機能を単独で実行するように出来ています。サンプルプログラム内の一部を取り出して再利用することで、必要な機能を動作させることができます。

2 概要

TX03ペリフェラルドライバを下記のように使用します。



3 使用する機能

機能	チャネル	使用/未使用
CG	クロックギア	CG サンプル
SysTick	-	未使用
ウォッチドッグタイマ (WDT)	WDT0	Used for WDT demo
外部割込み (INT)	INT0	未使用
	INT1	未使用
シリアルチャネル (SIO/UART)	SIO0	SIO/UART サンプル: リターゲット(UART), SIO サンプル, UART FIFO サンプル
16 ビットタイマ / イベントカウンタ (TMRB)	TMRB0	TMRB サンプル: 汎用タイマ, PPG 出力
	TMRB1	未使用
	TMRB2	未使用
	TMRB3	未使用
16 ビットタイマ A	TMR16A0	TMR16A サンプル: 汎用タイマ

機能	チャネル	使用/未使用
DSADC	UnitA	DSADC サンプル
	UnitB	DSADC サンプル
	UnitC	DSADC サンプル
	UnitD	DSADC サンプル
uDMAC	-	DMAC サンプル
SSP	SSP0	SSP0 サンプル(セルフループバック)
TEMP	-	TEMP サンプル

4 端子用途

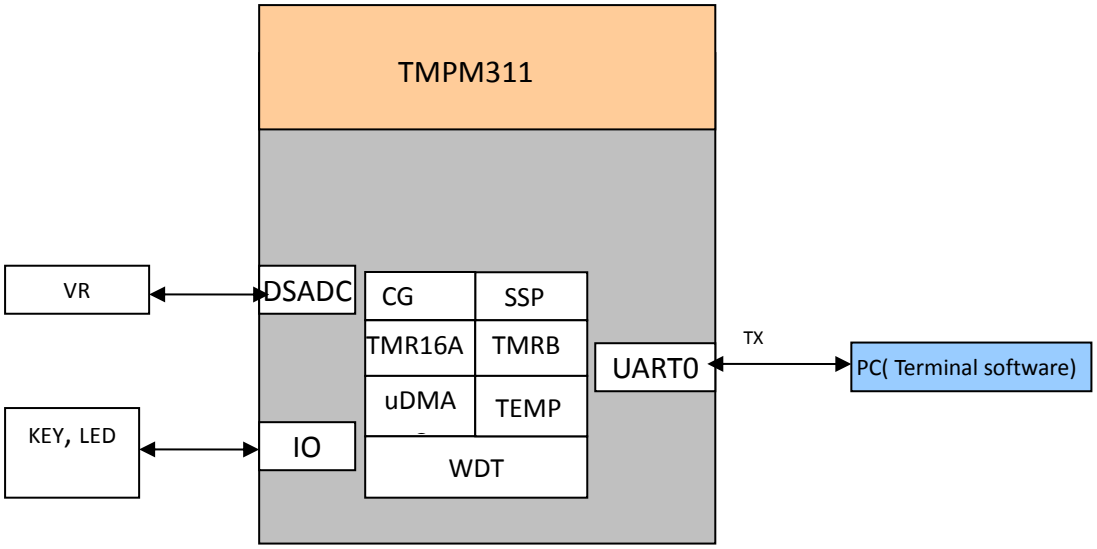
本サンプルプログラムは、開発環境に記載された評価ボードを用いてテストされています。以下に、端子用途を説明します。

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	24 ビットΔΣ型 AD 用 GND
23	DAINA+/ DAINA-	アナログ入力
25	VREFINA	24 ビットΔΣ型 AD 用電源
26	AGNDREFB	24 ビットΔΣ型 AD 用 GND
27	DAINB+/ DAINB-	アナログ入力
29	VREFINB	24 ビットΔΣ型 AD 用電源
30	AGNDREFC	24 ビットΔΣ型 AD 用 GND
31	DAINC+/ DAINC-	アナログ入力
33	VREFINC	アナログ入力
34	AGNDREFD	24 ビットΔΣ型 AD 用 GND
35	DAIND+/ DAIND-	アナログ入力

37	VREFIND	アナログ入力
38	DSRVDD3	アンプ用電源
39	SRVDD	基準電圧回路用電源
40	DSRVSS	基準電圧回路用 GND
6	PD0	高速発振子接続
8	PD1	高速発振子接続
41	MODE	MODE 端子: Low レベルに固定
43	RESET	リセット信号入力

5 開発環境

下記に開発環境の構成を示します。



1. ハードウェア:
   TMPM311CHDUG 用評価ボード (非売品)

2. 開発ツール:
   - 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 機能

6-1 動作モード選択

本デバイスの動作モードは NORMAL モードのみです。
NORMAL モード:
CPU コアおよび周辺ハードウェアを高速クロックで動作させるモードです。

6-2 DSADC

DSADC と UART のペリフェラルドライバを使用し、4 つの DSADC の変換結果を UART に出力するサンプルプログラムです。

補足:
端子接続
1. 電圧入力
    (a) pin 23 (DAINA+) / pin 24 (DAINA-) 電圧 (-0.375V < 入力電圧 < +0.375V)
    (b) pin 27 (DAINB+) / pin 28 (DAINB-) 電圧 (-0.375V < 入力電圧 < +0.375V)
    (c) pin 31 (DAINC+) / pin 32 (DAINC-) 電圧 (-0.375V < 入力電圧 < +0.375V)
    (d) pin 35 (DAIND+) / pin 36 (DAIND-) 電圧 (-0.375V < 入力電圧 < +0.375V)
2. pin 22 (AGNDREFA), pin 26 (AGNDREFB), pin 30 (AGNDREFC), pin34 (AGNDREFD) を GND に接続します。
3. VREFINx と AGNDREFx の間に 1μF のコンデンサを接続します。
    (a) 1uF のコンデンサの'+'と pin 25 (VREFINA) を接続し、コンデンサの'-'と pin 22 (AGNDREFA)を接続します。
    (b) 1uF のコンデンサの'+'と pin 29 (VREFINB) を接続し、コンデンサの'-'と pin 26 (AGNDREFB) を接続します。
    (c) 1uF のコンデンサの'+'と pin 33 (VREFINC) を接続し、コンデンサの'-'と pin 30 (AGNDREFC) を接続します。
    (d) 1uF のコンデンサの'+'と pin 37 (VREFIND) を接続し、コンデンサの'-'と pin 34 (AGNDREFD) を接続します。
4. pin 38 (DSRVDD3) と pin 39 (SRVDD)を Vcc に接続します。
5. pin 40 (DSRVSS) を GND に接続します。

6-3 GPIO

GPIO のペリフェラルドライバを使用し、サンプルプログラムです。LED の ON/OFF を行うサンプルプログラムです。

6-4 SSP

SSPとUARTのペリフェラルドライバを使用し、受信したデータをUARTに表示するサンプルプログラムです。送受信データが同一の場合はLED2とLED3をONし、不一致の場合はLED0とLED1をONします。

UART 表示:

```
SSP RX DATA:
xxxxxx
```

6-5 TEMP

TEMP と UART のペリフェラルドライバを使用し、温度センサと接続された DSADC のユニット D の変換結果を UART に出力するサンプルプログラムです。

補足:
端子接続:
1. 入力電圧
    (a) 温度センサとアナログ入力が内部接続されます。
2. pin34 (AGNDREFD) を GND に接続します。
3. VREFINx と AGNDREFx との間を 1μF コンデンサで接続します。
    (a) 1uF のコンデンサの'+'と pin 37 (VREFIND) を接続し、コンデンサの'-'と pin 34 (AGNDREFD) を接続します。
4. pin 38 (DSRVDD3) と pin 39 (SRVDD) を VCC に接続します。
5. pin 40 (DSRVSS) を GND に接続します。

6-6 TMR16A

TMR16Aのペリフェラルドライバを使用し、汎用タイマを実現するサンプルプログラムです。
タイマの周期は1sです。このタイマを使用し、1s (500msでON、500msでOFF) 間隔でLED点滅を行います。

6-7 TMRB

6-7-1 汎用タイマ

TMRBのペリフェラルドライバを使用し、汎用タイマを実現するサンプルプログラムです。
タイマの周期は1sです。このタイマを使用し、1s (500msでON、500msでOFF) 間隔でLED点

滅を行います。

6-7-2 PPG 出力

TMRBのペリフェラルドライバを使用し、PPG出力を行うサンプルプログラムです。
PPG 波形は、タイマ出力端子をオシロスコープに接続することで確認できます。
SW0 を押すことでデューティーを変更します。
10% → 25% → 50% → 75% → 90% → 10%
電源投入すると PPG モードに入り、PPG 波形出力を開始します。

6-8 SIO/UART

6-8-1 リターゲット(UART)

C 言語の標準入出力ライブラリの標準入力(stdin)、標準出力(stdout)をターゲットのハードウェアに合わせて変更することをリターゲットと呼びます。
標準入力と標準出力を、UARTにリターゲットし、printf()関数を使ってシリアルポートから出力を行います。
補足:
サンプルプログラムでは UART0 を使用します。

6-8-2 UART FIFO

マスタとなる UART0 から"TMPM3111"というデータを FIFO を使用して、スレーブの UART0 へ送信します。また同時にマスタとなる UART0 は"TMPM3112"というデータを FIFO を使用して、スレーブの UART0 から受信します。
ResetIdx() 関数の先頭にブレークポイントを設定すると、RxBuffer=”TMPM3112” 、RxBuffer1="TMPM3111"となると設定したブレークポイントで停止します。
(マスタ側の SC0TXD とスレーブ側の SC0RXD を、マスタ側の SC0RXD とスレーブ側の SC0TXD を接続します。)
補足:
まずホスト側のプログラムを実行し、その後、マスタ側のプログラムを実行してください。

6-8-3 SIO

SIO のペリフェラルドライバを使用し、同期モードでデータ転送を行うサンプルプログラムです。
マスタ側の SIO0 とスレーブ側の SIO0 を接続します。(マスタ側の SC0TXD とスレーブ側の SC0RXD を、マスタ側の SC0RXD とスレーブ側の SC0TXD を, マスタ側の SC0SCK とスレーブ側の SC0SCK を接続します。

補足:

まずホスト側のプログラムを実行し、その後、マスタ側のプログラムを実行してください。

6-9 uDMAC

uDMAC のペリフェラルドライバを使用し、1KB の RAM 領域の中で"src" から "dst"へソフトウェアトリガを利用してデータ転送を行うサンプルプログラムです。

処理手順:

1. uDMAC のユニット A の初期化: 転送タイプはバーストタイプ、チャネルはイネーブル、マスク設定のチャネルはイネーブル、初期データは使用、優先度通常は使用
2. プライマリベースアドレスの設定: 1KB RAM の先頭アドレス
3. ソフトウェアトリガの設定
4. uDMAC のユニット A の動作を開始
5. DMAC_BASIC モードが選択されている場合、転送が完了するまで再度トリガをかけ続けます。DMAC_AUTOMATIC モードが選択されている場合、転送が完了する前に一度トリガをかけます。
6. 転送完了を確認した後、送信元と送信先のデータを比較します。

6-10 WDT

WDT のペリフェラルドライバを使用して WDT 割り込みの発生とウォッチドッグカウンタをクリアするサンプルプログラムです。

処理手順:

1. 検出時間の設定とカウンターオーバーフロー時の動作としての WDT 割り込み設定を行い、WDT を初期化します。
2. 2 つの WDT 用サンプルがあり、DEMO2 はマクロ定義にて切り替えられます。
   DEMO1:
   タイマーオーバーフロー時に NMI 割り込みを発生し、WDT をクリアします。
   DEMO2:
   ウォッチドッグタイマ制御レジスタにクリアコードを書き込み、ウォッチドッグタイマカウンタをクリアします。

7 ソフトウェア

本ソフトウェアは、TMPM311 の主要機能を評価ボードを使用して動作確認するためのサンプルプログラムです。

サンプルプログラムの実装には、最新のドライバーソフト、および IAR EWARM、または KEIL MDK

をご使用ください。機能ごとにプロジェクトを作成してください。

ワークスペース構造とプロジェクト名は、以下の通りです:

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
```

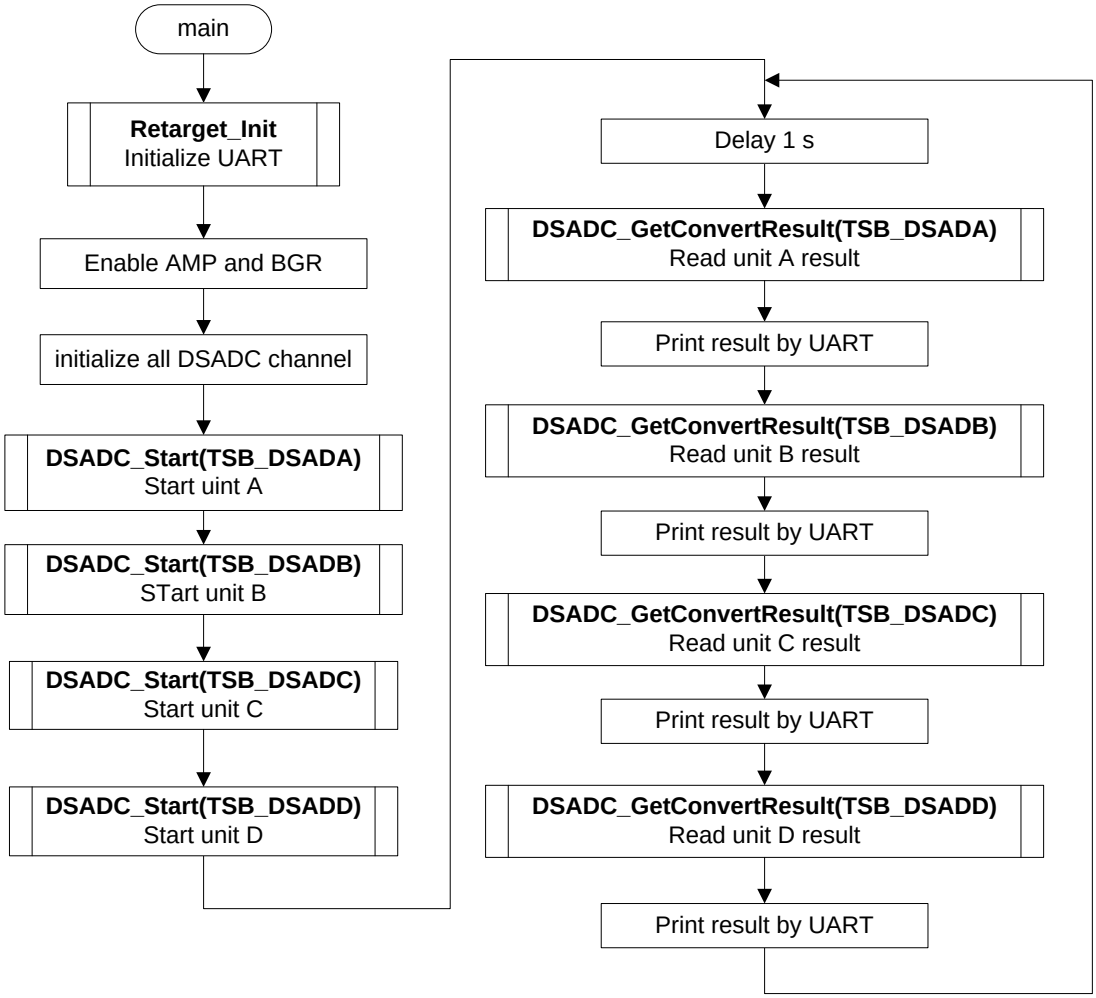
7-1 DSADC

7-1-1 例: DSADC データリード

ペリフェラルドライバ (DSADC, TEMP, UART) を用いたサンプルプログラムです。

このサンプルプログラムでは以下を行います。
1. ソフトウェアトリガにて DSAD 変換を実施
2. DSAD 変換の状態と AD 変換結果のリード

- **フローチャート:**



- **サンプルプログラムのコードと説明**

まず UART の初期化を行います。

```
    Retarget_Init();
```

AMP と BGR を許可状態にし、安定時間として 1ms 待ちます。

```
    TMRB_SetRunState(TSB_TB0, TMRB_RUN);

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

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

DSADC のすべてのチャネルを初期化します。

```
    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);
    }
```

変換結果を取得し、UART に変換結果を出力します。

```
    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

7-2-1 例: GPIO データリード

ペリフェラルドライバ(GPIO)を使用したサンプルプログラムです。

このサンプルプログラムでは以下を行います。

1. GPIO の初期化
2. GPIO へのデータライト
3. GPIO からのデータリード

- **サンプルプログラムのコードと説明**

まず GPIO_SetOutput() 関数を使用して GPIO を LED 用に設定します。次に GPIO_SetInput()関数を使用して GPIO をスイッチ用に設定します。

```
    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);
```

for(; ;)文内でスイッチの状態によって LED の ON/OFF を制御します。

GPIO_ReadDataBit()関数を使用してスイッチ状態をリードします。

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

GPIO_WriteData()関数を使用して LED を ON します。

```
    Uint8_t tmp;
    tmp = GPIO_ReadData(GPIO_PB);
    tmp |= led;
    GPIO_WriteData(GPIO_PB, tmp);
```

GPIO_WriteData()関数を使用して LED を OFF します。

```
    Uint8_t tmp;
    tmp = GPIO_ReadData(GPIO_PB);
    tmp &= ~led;
    GPIO_WriteData(GPIO_PB, tmp);
```

7-3 SSP

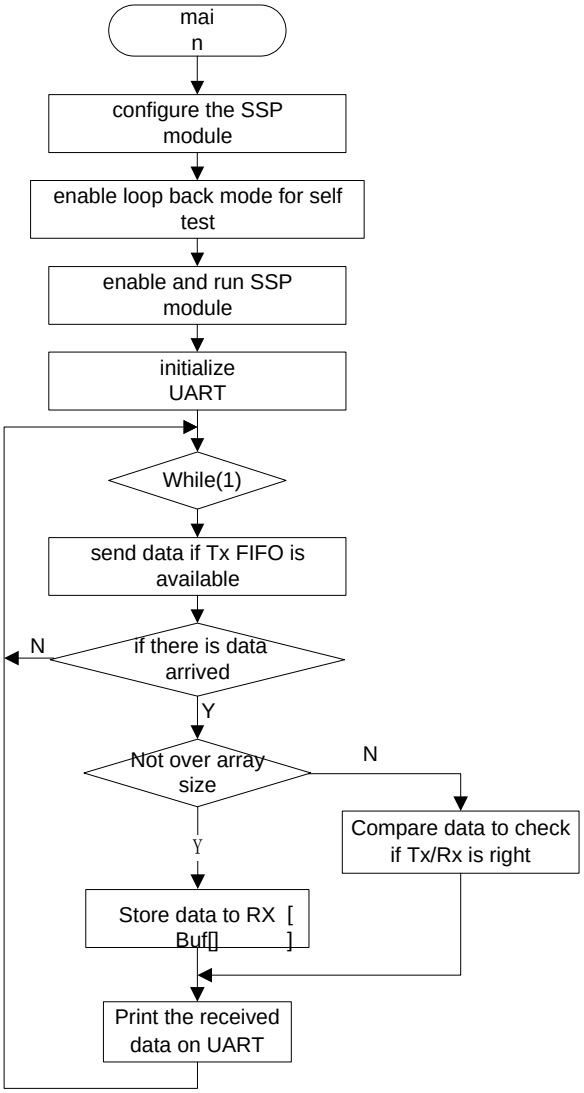
7-3-1 例: SSP0 セルフループバック

ペリフェラルドライバ(SSP, GPIO)を使用したサンプルプログラムです。

このサンプルプログラムでは以下を行います。

1. SSP0 の初期化
2. ループバックモードによるデータ送受信

- **フローチャート:**



- **サンプルプログラムのコードと説明**

まず、SSP0 の初期化を行います。

```
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);
```

ループバックモードに設定します。

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

SSP を許可します。

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

GPIO を LED 用に設定し、UART をリターゲットできるよう設定します。

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

データ送受信を行います。

```
	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 */
          }
```

受信データを UART 経由で表示します。

```
          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 例: TEMP

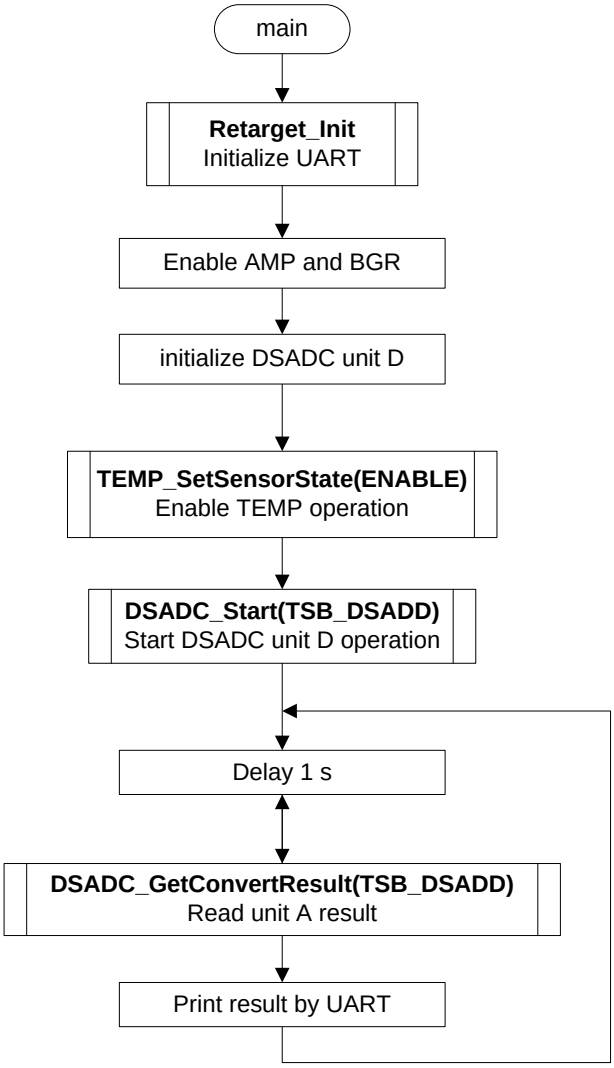
ペリフェラルドライバ (TEMP, DSADC, UART) を使用したサンプルプログラムです。

このサンプルプログラムでは以下を行います。

1. TEMP と DSADC の初期化
2. TEMP で計測された電圧を DSAD で変換します。

3. デジタル値を UART 経由で出力します。

- **フローチャート:**



- **サンプルプログラムのコードと説明**

まず、UART の初期化を行います。

```
Retarget_Init();
```

AMP と BGR を許可し、安定時間として 1 ms 待ちます。

```
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) {
};
```

DSADC のユニット D を初期化します。

```
    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);
```

変換結果を UART 経由で出力します。

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

7-5 TMR16A

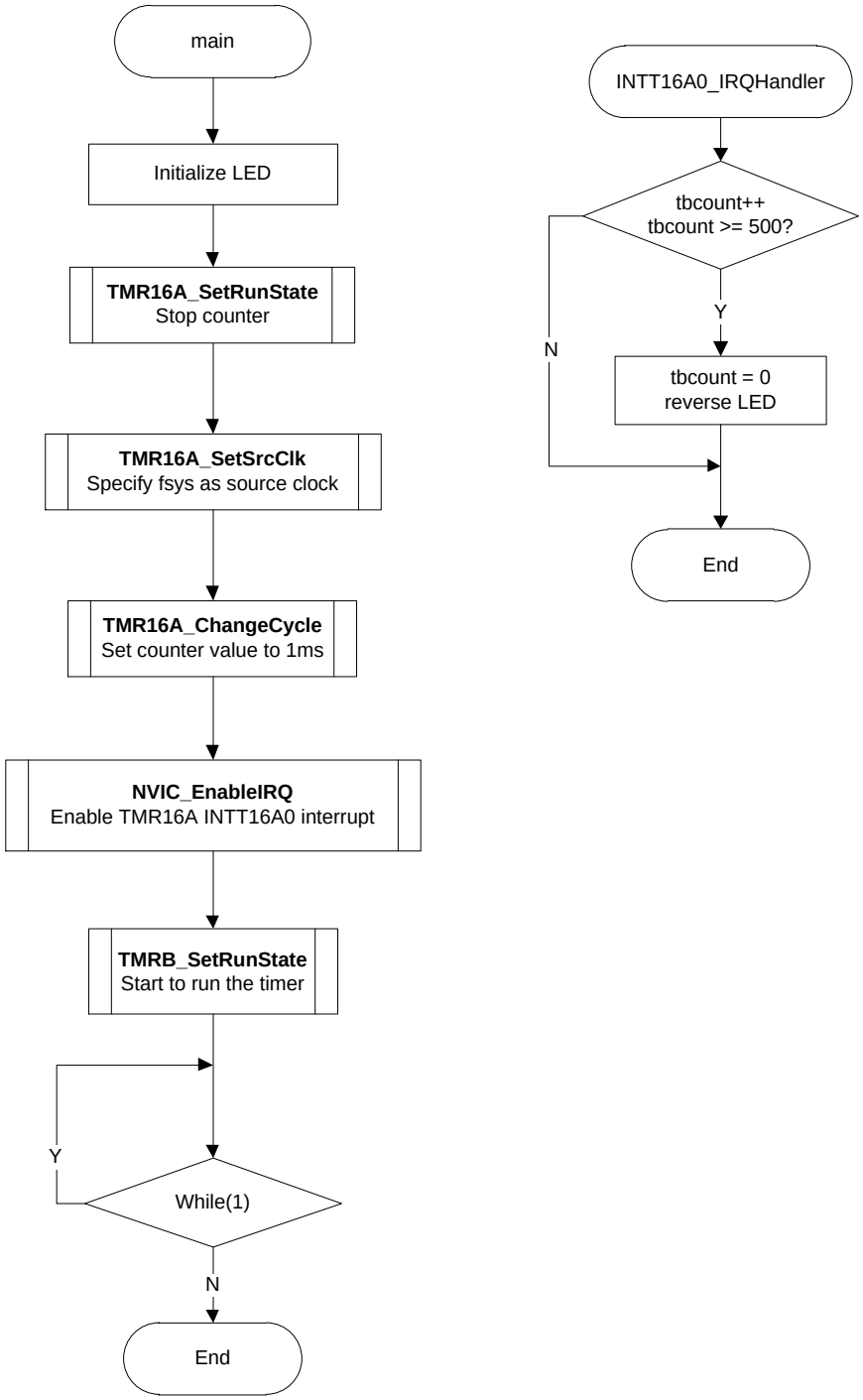
7-5-1 例: 汎用タイマ

ペリフェラルドライバ (TMR16A, GPIO) を使用したサンプルプログラムです。

このサンプルプログラムでは以下を行います。

1. TMR16A の初期化
2. 1ms の汎用タイマ

- **フローチャート:**



- **サンプルプログラムのコードと説明**

まず GPIO を LED 用に設定し、すべての LED を ON します。

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

カウンタを停止し、サイクルを TMR16A_1MS マクロ(0x1A40)を使用して 1ms に設定します。このマクロ値は次のように算出されます。

ftmra = fphiT0 = fsys = fc = 8MHz, Ttmra = 1/8us, 1000*8 = 8000 = 0x1A40

カウンタを開始します。

```
    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);
```

while(1)文にて割り込み発生を待ちます。割り込みハンドラ内でカウントアップし、500ms までカウントすると LED を反転表示させ、カウントを再開します。

```
    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

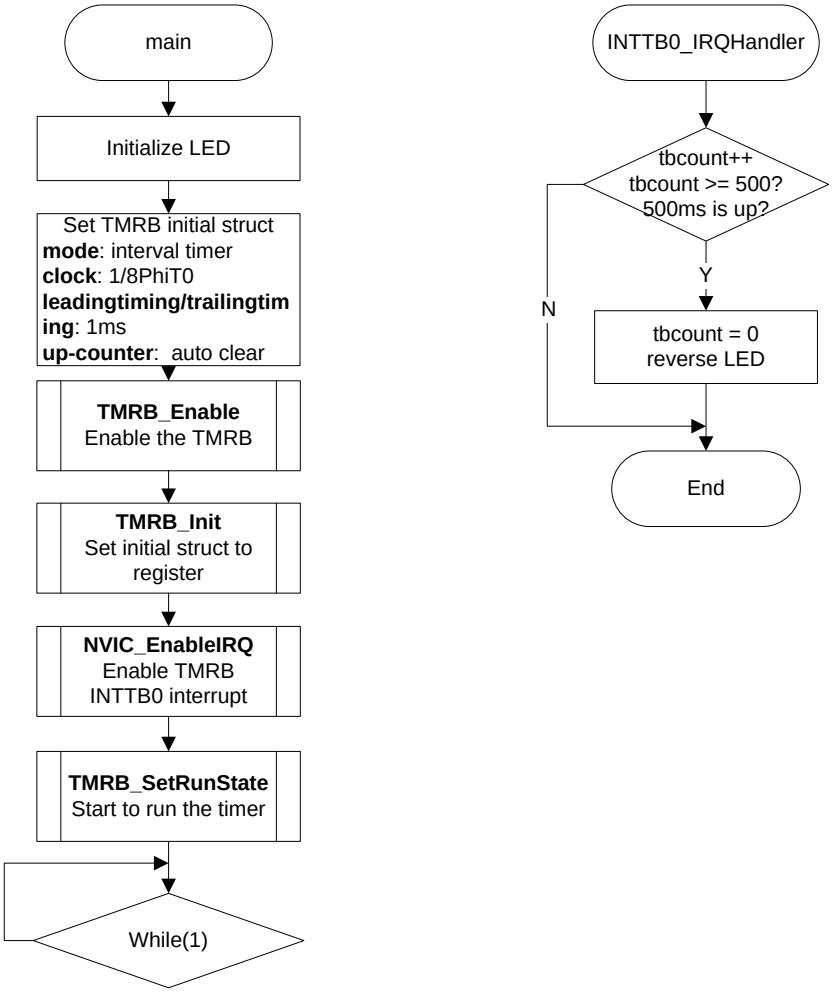
7-6-1 例: 汎用タイマ

ペリフェラルドライバ (TMRB, GPIO) を使用したサンプルプログラムです。

このサンプルプログラムでは以下を行います。

1. TMRB の初期化
2. 1ms の汎用タイマ

- **フローチャート:**



- **サンプルプログラムのコードと説明**

まず GPIO を LED 用に設定し、LED を ON します。

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

TMRB 設定用の構造体を用意し TMRB モード、クロック、アップカウンタクリア方法、周期とデューティを設定します。このサンプルプログラムでは、1ms の周期とデューティを設定します。このカウント値は Tmrb_Calculator 関数で算出します。

```
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);
```

TMRB 動作の許可、および初期化を行います。INTTB0 割り込み(1ms 毎にトリガ)を許可し、最後に TMRB を動作します。

```
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*/
```

while(1)内にて割り込み発生を待ちます。割り込みハンドラ内でカウントアップし、500ms までカウントすると LED を反転させ、カウントを再開します。

```
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 */
}
```

Tmrb_Calculator()関数:

```
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);
}
```

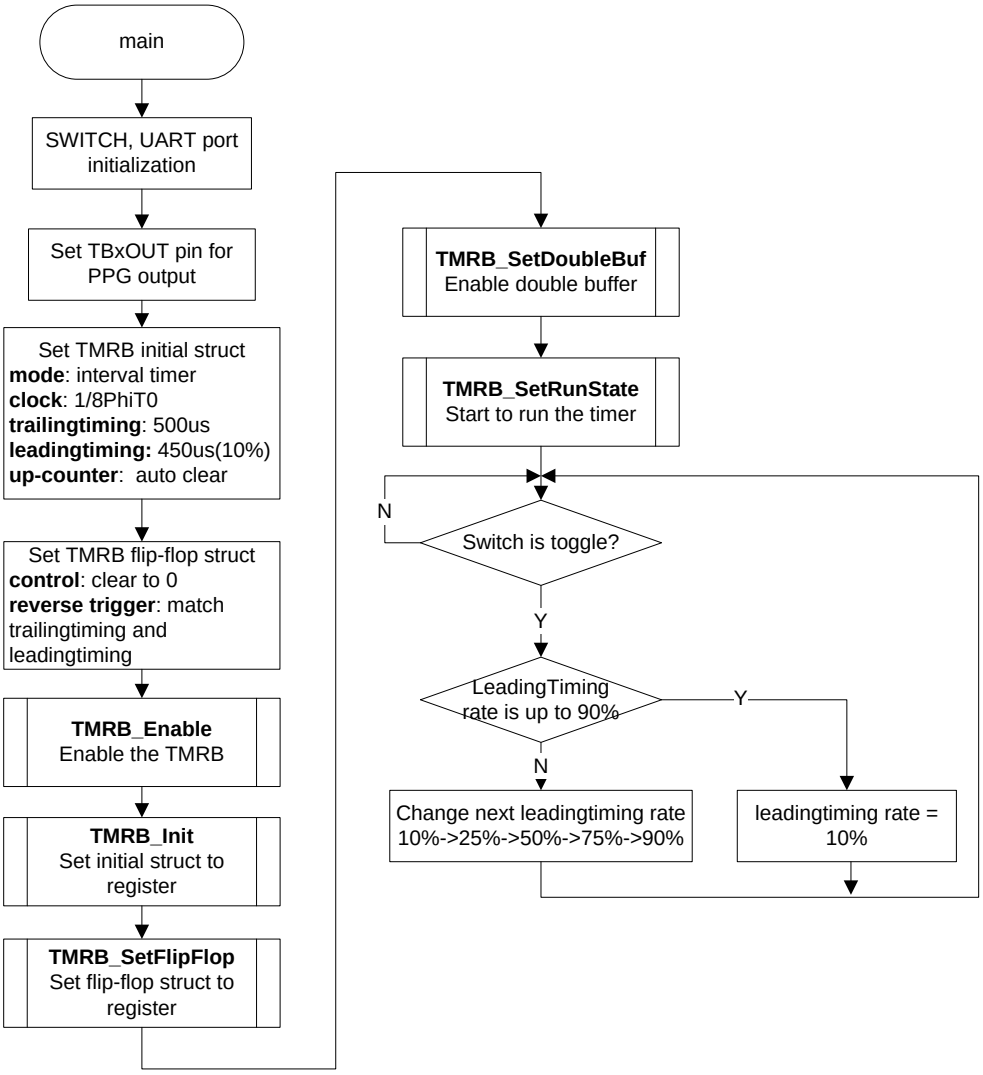
7-6-2 例: PPG 出力

ペリフェラルドライバ (TMRB, GPIO, UART) のプログラムサンプルです。

このサンプルプログラムでは以下を行います。

1. TMRB0 の初期化
2. PPG 動作の設定
3. PPG 波形の調整

- **フローチャート:**



- **サンプルプログラムのコードと説明**

最初に、配列 tgtLeadingTiming、LeadingTimingus、LeadingTiming を初期化し、PPG 出力用に PB1 を TB0OUT に設定します。

```
	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 出力 */
    GPIO_SetOutput(GPIO_PB, GPIO_BIT_1);
    GPIO_EnableFuncReg(GPIO_PB, GPIO_FUNC_REG_1, GPIO_BIT_1);
```

TMRB 初期化構造体を用意し、TMRB モード、クロック、アップカウンタクリア方法周期、デューティを設定します。本サンプルプログラムでは、500μs の周期とデューティを設定します。周期とデューティの設定値は Tmrb_Calculator()関数で算出します。

```
    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% */
```

フリップフロップ初期化構造体、フリップフロップ制御、反転トリガ引数を設定します。反転トリガは、デューティとサイクルと一致するよう設定します。

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

TMRB モジュールを許可し、指定レジスタに初期化構造体、フリップフロップ構造体を設定します。ダブルバッファを許可し、キャプチャ機能を禁止にします。最後に、TMRB を動作させます。

```
    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);
```

スイッチが Low から High になるまで待ち、同時に UART に現在のデューティを表示します。

```
    do {                            /* wait if switch is Low */
            keyvalue = GPIO_ReadDataBit(KEYPORT, GPIO_BIT_4);
            LeadingTiming_display(); /* display current leadingtiming */
      } while (GPIO_BIT_VALUE_0 == keyvalue);
```

スイッチが High になると、下記のようにデューティを設定します。

10%->25%->50%->75%->90%

その後 90% から、また 10% になります。

```
    Rate++;
```

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

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

Tmrb_Calculator 関数:

```
    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

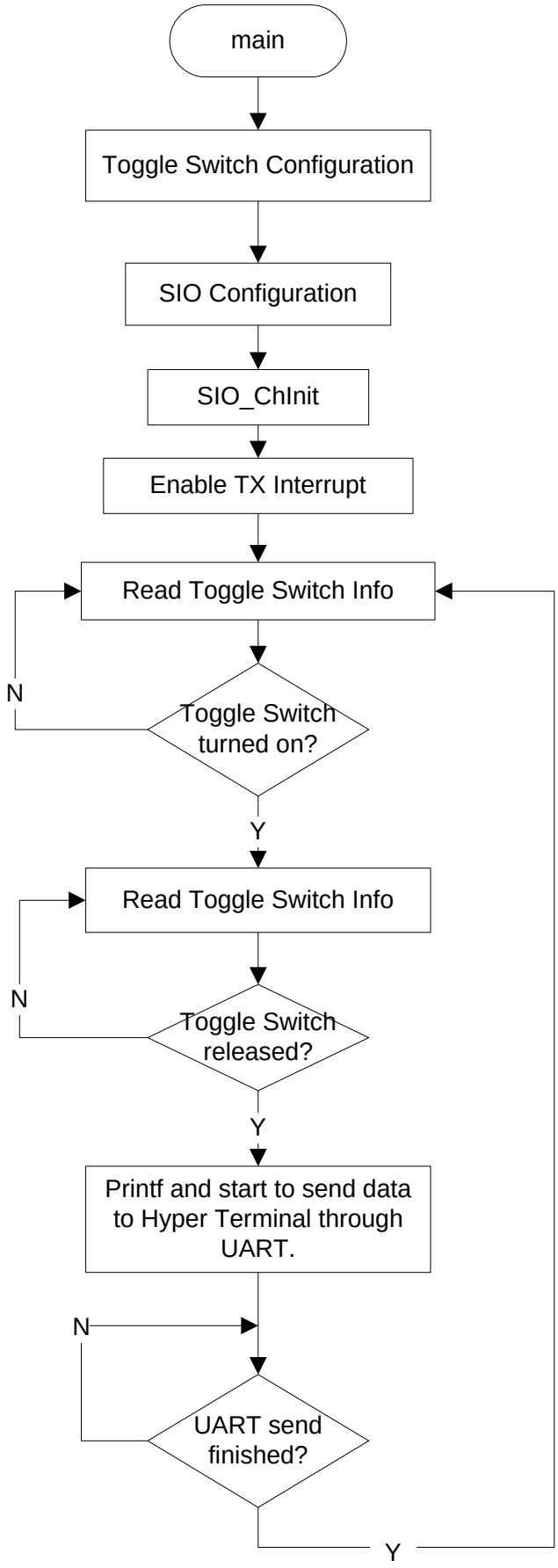
7-7-1 例: リターゲット(UART)

ペリフェラルドライバ (UART, GPIO)を用いたサンプルプログラムです。

このサンプルプログラムでは以下を行います。

1. UART 設定と初期化
2. UART 送信制御
3. データ送信に UART の TX 割り込みを使用
4. UART に printf()関数をリターゲット

- **フローチャート:**


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

- **サンプルプログラムのコードと説明**

まず、GPIO 設定と UART の初期化を行います。
GPIO ペリフェラルドライバを使い、GPIO を UART に設定します。

```
    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);
```

UART_InitTypeDef 構造体を準備し、データを設定します。以下は設定例です。

```
    UART_InitTypeDef myUART;

    /* configure SIO0 for reception */
    UART_Enable(リターゲット(UART));
    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(リターゲット(UART), &myUART);
```

上記設定を行い、その後、UART の送信割り込みを有効にします。

```
    NVIC_EnableIRQ(RETARGET_INT)
```

送信データは TxBuffer に文字列として確認できます。

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

UART0 の送信は UART0 割り込みルーチンで行います。
UART0 の送信割り込みルーチン:

```
void INTTX0_IRQHandler(void)
{
    if (gSIORdIndex < gSIOWrIndex) {       /* buffer is not empty */
        UART_SetTxData(リターゲット(UART),  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 */
    }
}
```

printf()関数は IAR コンパイラの putchar()を、RealView コンパイラの fputc()をコールしてデータ出力を行います。

```
#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(リターゲット(UART), gSIOTxBuffer[gSIORdIndex++]);
            NVIC_EnableIRQ(RETARGET_INT);
      }

      return ch;
}
```

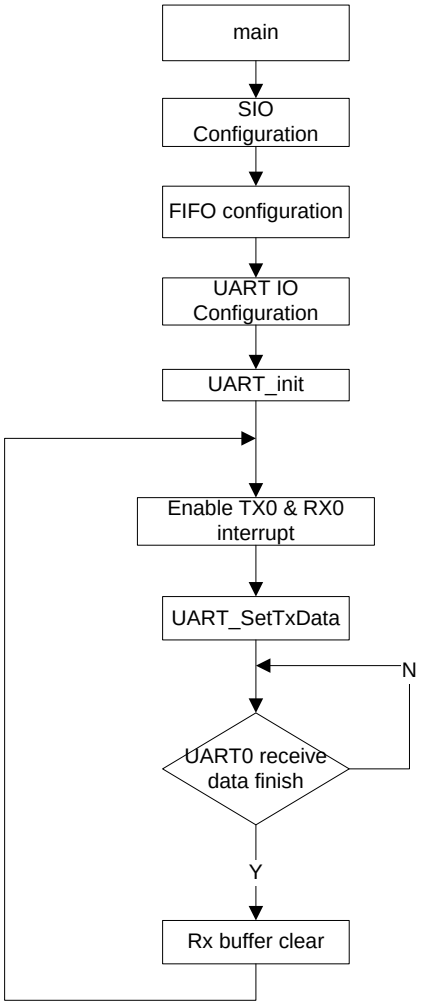
7-7-2 例: UART FIFO (マスタ)

ペリフェラルドライバ(UART, GPIO)を使用したサンプルプログラムです。

この例では以下を行います。

1. UART と FIFO の初期設定
2. FIFO を使用した UART の送受信

- **フローチャート:**



- **サンプルプログラムのコードと説明**

まず GPIO と UART の初期設定を行います。

GPIO をマスタとなる UART0 に設定します。

```
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 */
      }
}
```

UART_InitTypeDef 構造体を準備し、データを設定します。以下は設定例です。

```
    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(リターゲット(UART), &myUART);
```

UART0 の有効化と初期設定を行います。

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

FIFO の初期設定を行います。

```
    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);
```

上記設定を行い、その後、UART0 の割り込みを有効にします。

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

UART0 の送信割り込みルーチン:

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

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

UART0 の受信割り込みルーチン:

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

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

```
}
```

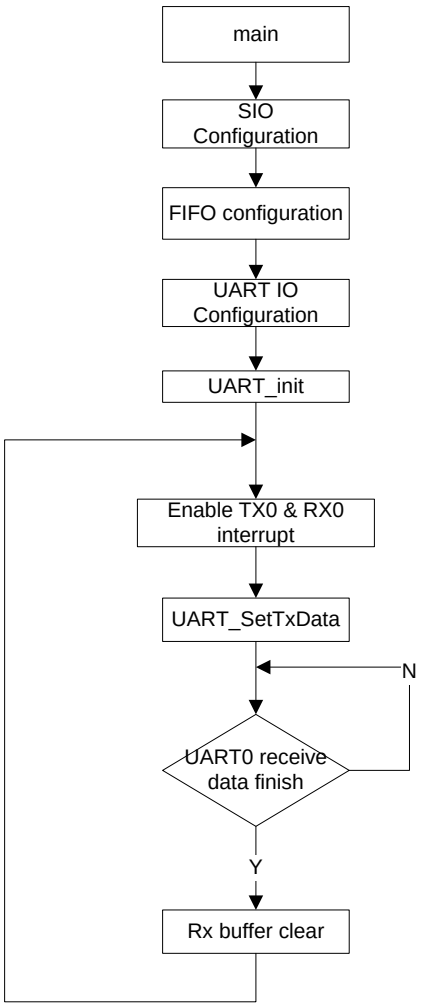
7-7-3 例: UART FIFO (スレーブ)

ペリフェラルドライバ(UART, GPIO)を使用したサンプルプログラムです。

この例では以下を行います。

1. UART と FIFO の初期設定
2. FIFO を使用した UART の送受信

- **フローチャート:**



- **サンプルプログラムのコードと説明**

まず GPIO と UART の初期設定を行います。

GPIO をスレーブとなる UART0 に設定します。

```
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 */
      }
}
```

UART_InitTypeDef 構造体を準備し、データを設定します。以下は設定例です。

```
      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(リターゲット(UART), &myUART);
```

UART0 の有効化と初期設定を行います。

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

FIFO の初期設定を行います。

```
      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);
```

上記設定を行い、その後、UART0 の割り込みを有効にします。

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

UART0 の送信割り込みルーチン:

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

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

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

UART0 の受信割り込みルーチン:

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

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

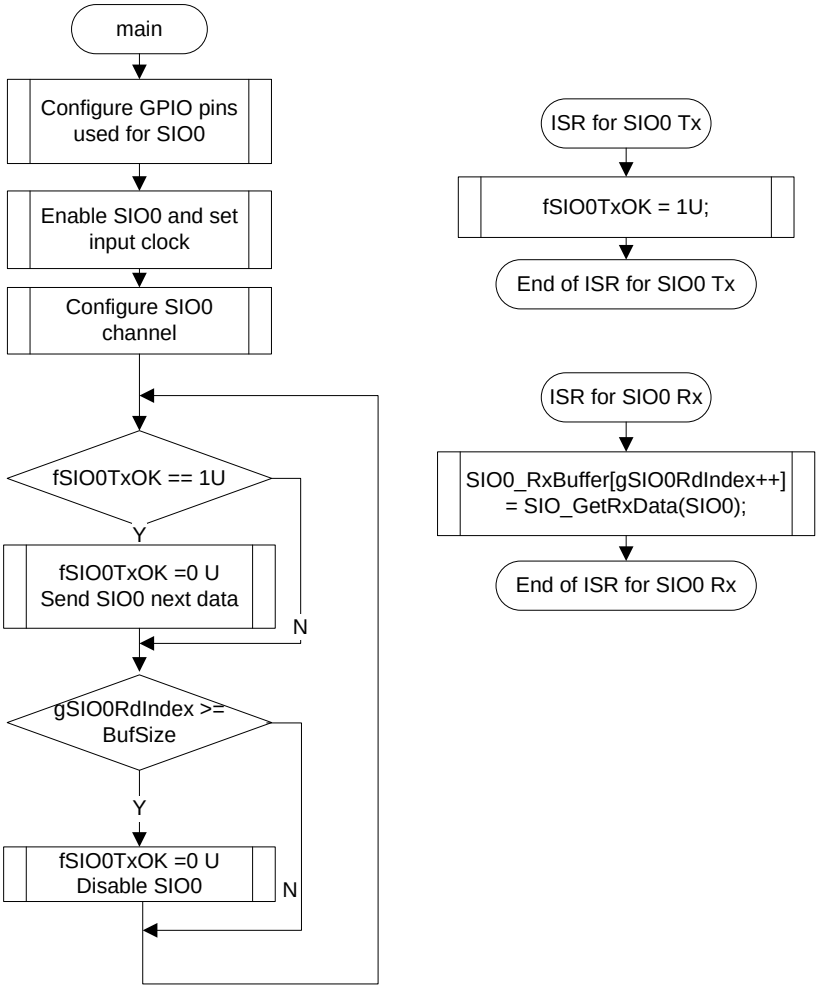
7-7-4 例: SIO (マスタ)

ペリフェラルドライバ(SIO)を使用したサンプルプログラムです。

この例では以下を行います。

1. SIO 動作の基本設定
2. マスタ SIO0 とスレーブ SIO0 間のデータ転送
3. SIO の送受信割り込み

- **フローチャート:**



- **サンプルプログラムのコードと説明**

まず GPIO を SIO に設定します。

その後、SIO0 を有効にし、入力クロックの設定と SIO0 の初期化を行います。

```
/*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);
```

SIO 送受信割り込みを許可します。

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

すべて基本的な設定を行った後、データ送信を行います。

```
    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 */
             }

        }
       }
```

SIO0 送信の割り込みハンドラにおいて、転送完了フラグをセットします。

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

SIO0 受信の割り込みハンドラにおいて、受信バッファからデータを取得します。

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

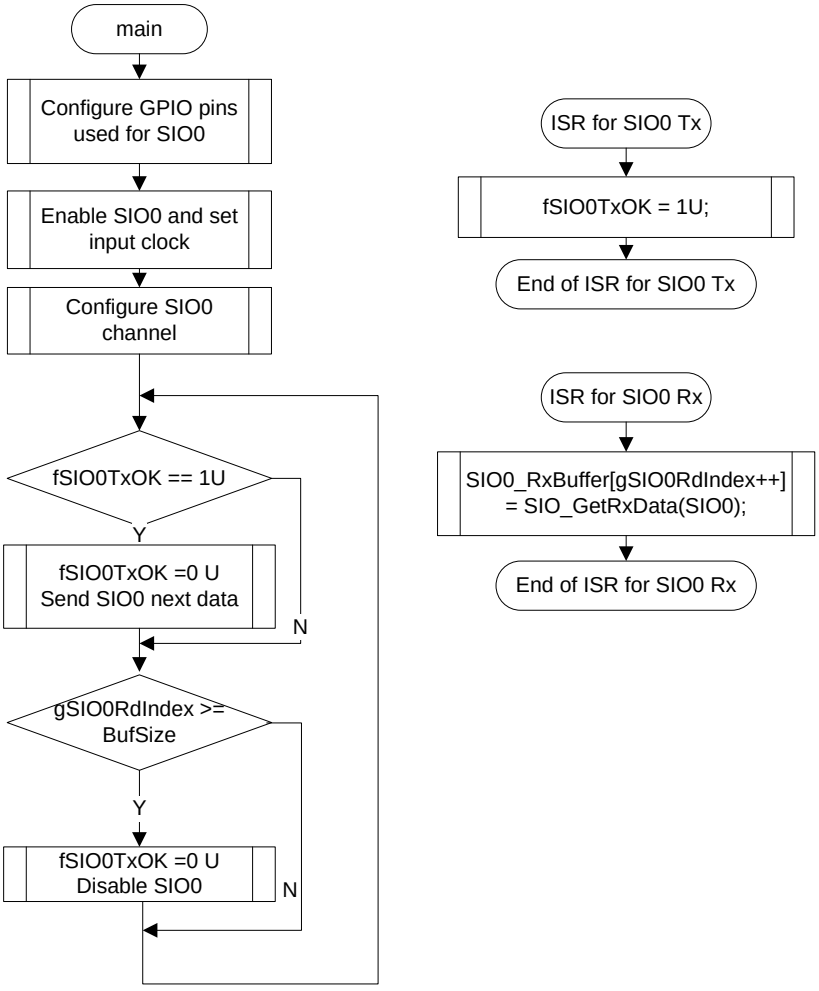
7-7-5 例: SIO (スレーブ)

ペリフェラルドライバ(SIO)を使用したサンプルプログラムです。

この例では以下を行います。

1. SIO 動作の基本設定
2. マスタ SIO0 とスレーブ SIO0 間のデータ転送
3. SIO の送受信割り込み

- **フローチャート:**



- **サンプルプログラムのコードと説明**

まず GPIO を SIO に設定します。

その後、SIO0 を有効にし、入力クロックの設定と SIO0 の初期化を行います。

```
/*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);
```

SIO 送受信割り込みを許可します。

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

すべて基本的な設定を行った後、データ送信を行います。

```
  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 */
           }

       }
  }
```

SIO0 送信の割り込みハンドラにおいて、転送完了フラグをセットします。

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

SIO0 受信の割り込みハンドラにおいて、受信バッファからデータを取得します。

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

7-8 uDMAC

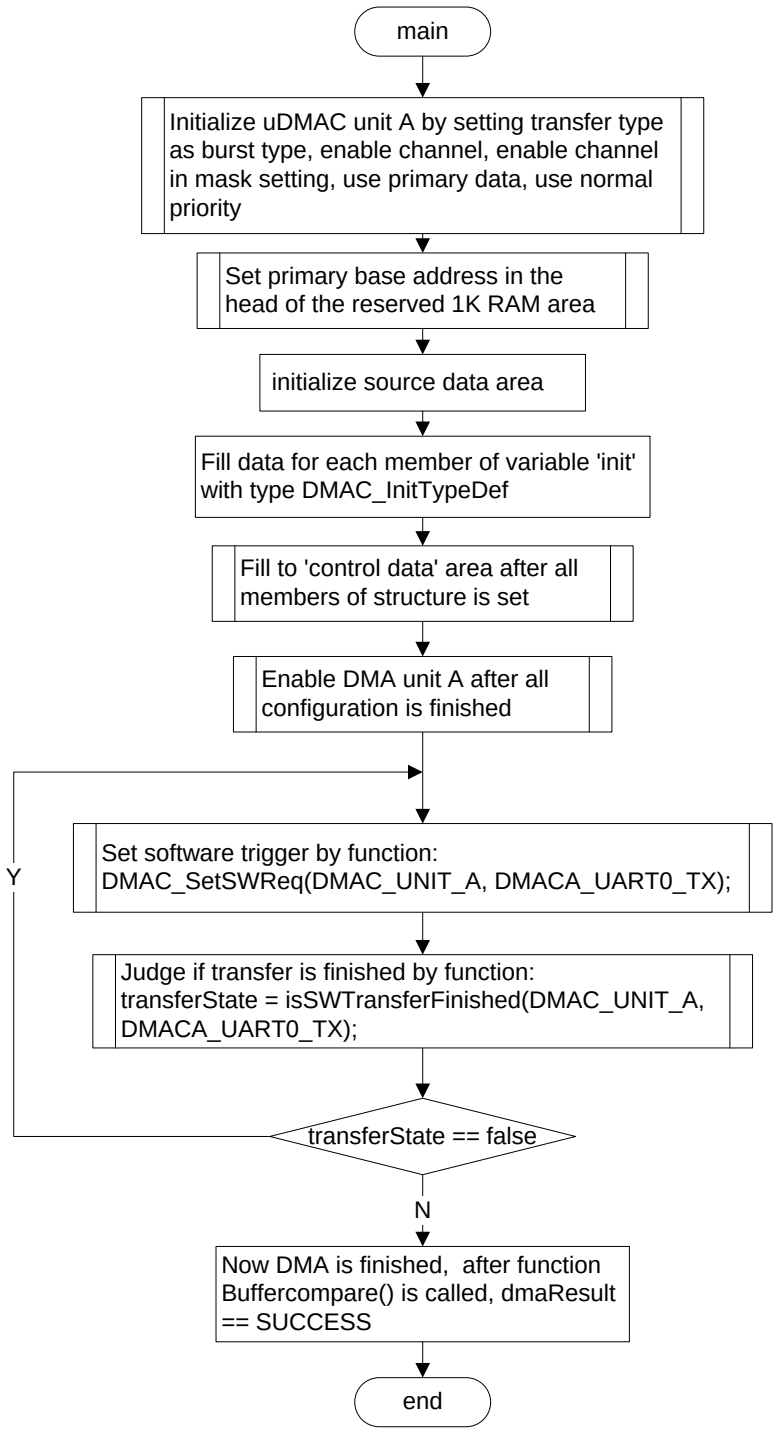
7-8-1 例: メモリ→メモリの DMA 転送

ペリフェラルドライバ(µDMAC) を使用したサンプルプログラムです。

この例では以下を行います。

1. 1KB RAM 領域を µDMA 制御データ用に確保
2. µDMA 設定と初期化
3. ソフトウェアトリガによるメモリ間 DMA 転送開始

- **フローチャート:**



- **サンプルプログラムのコードと説明**

μDMAC は構成データを保存するため、1K バイトの RAM 領域を必要とします。本領域は、ベースアドレスのビット 0～9 を 0 にして、アプリケーションコード中に確保しなければなりません。例えば、0x20000400 は OK ですが、0x20000300 は使用できません。
IAR EWARM と Keil MDK(RealView)における、RAM 領域を確保するための例です。

```
#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
```

Keil MDK では、キーワード ‘__at’が使用されますが、IAR EWARM では十分ではありません。
リンク設定ファイル(.lcf) が、下記内容に変更する必要があります。(UNIT A のみ). 詳細は、
“TMPM311CHDUG_RAM_For_uDMAC.icf”ファイルを参照してください。

```
    /* 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 };
```

RAM を確保した後、転送タイプとしてバーストタイプを設定し、チャネルをイネーブル、マスク
設定のチャネルをイネーブル、初期データを使用、優先度通常を使用してください。

```
    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);
```

確保した1K バイトの RAM 領域の先頭に初期ベースアドレスを設定してください。

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

転送する送信元のデータ領域を初期化してください。

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

“DMAC_InitTypeDef”タイプの変数‘init’のメンバ設定を開始します。
送信元と送信先の最終アドレスを設定します。

```
    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])) ;
```

BASIC あるいは AUTOMATIC モードを選択します。

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

その他のメンバを設定します。

```
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;
```

全ての構成メンバを設定した後、'control data' 領域を入力します。

```
DMAC_FillInitData(DMAC_UNIT_A, DMACA_UART0_TX, &init);
```

全設定が終了した後、DMA ユニット A をイネーブルにします。

```
DMAC_Enable(DMAC_UNIT_A);
```

ソフトウェアトリガを設定します。転送が完了したかどうかを確認します。

```
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 );
```

転送が完了している場合、関数 Buffercompare()を呼び出し dmaResult が SUCCESS であるかを確認してください。

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

7-9 WDT

7-9-1 例: WDT

ペリフェラルドライバ (WDT, GPIO) のプログラムサンプルです。

この例では以下を行います。

1. WDT の初期化

2. DEMO1 では、オーバーフロー前に WDT クリアを行わず、NMI 割り込みを発生させ、LED1 を 1 回点滅します。
3. DEMO2 では、オーバーフロー前に WDT クリアを行い、常時 LED0 を点滅させます。

- **サンプルプログラムのコードと説明**

以下のコードはWDT の初期化の例です。検出時間が 2^25/fsys にされ、オーバーフロー時に NMI 割り込みを発生します。

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

WDT を初期化し、その後 WDT を有効にします。

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

DEMO1 では、NMI 割り込みの発生を待ちます。

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

DEMO1 では、NMI 割り込み発生時に WDT を禁止にし、LED1 を 1 回点滅させます。

```
WDT_Disable();
```

DEMO2 では、WDT クリアを行い、常に LED0 を点滅させます。

```
WDT_WriteClearCode();
```