TOSHIBA

TX03 ペリフェラルドライバ使用例 (TMPM372)

第一版

2017 年 9 月

東芝デバイス＆ストレージ株式会社

CMDR-M372UE-01J

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 1
2 概要 1
3 使用する機能 1
4 端子用途 3
5 開発環境 5
6 機能説明 6
6-1 動作モード 6
6-2 ADC 6
6-3 CG 7
6-3-1 パワーモード変更 7
6-4 FLASH 7
6-5 GPIO 9
6-6 OFD 9
6-7 SIO/UART 9
6-7-1 リターゲット 9
6-7-2 UART FIFO 9
6-7-3 SIO 9
6-8 TMRB 9
6-8-1 汎用タイマ 9
6-8-2 PPG 波形出力 10
6-9 VLTD 10
6-10 WDT 10
6-11 ENC 10
6-12 PMD 11
6-12-1 例: 位相出力 11
6-13 TRMOSC 11
7 ソフトウェア 13
7-1 ADC 14
7-1-1 例: ADC データ読み出し 14
7-2 CG 17
7-2-1 例: パワーモード変更 17
7-3 FLASH 21
7-4 GPIO 25
7-5 OFD 26
7-6 TMRB 29
7-6-1 例: 汎用タイマ 29
7-6-2 例: PPG 波形出力 31
7-7 SIO/UART 34
7-7-1 例: リターゲット 34
7-7-2 例: UART FIFO 38
7-7-3 例: SIO 42
7-8 VLTD 46

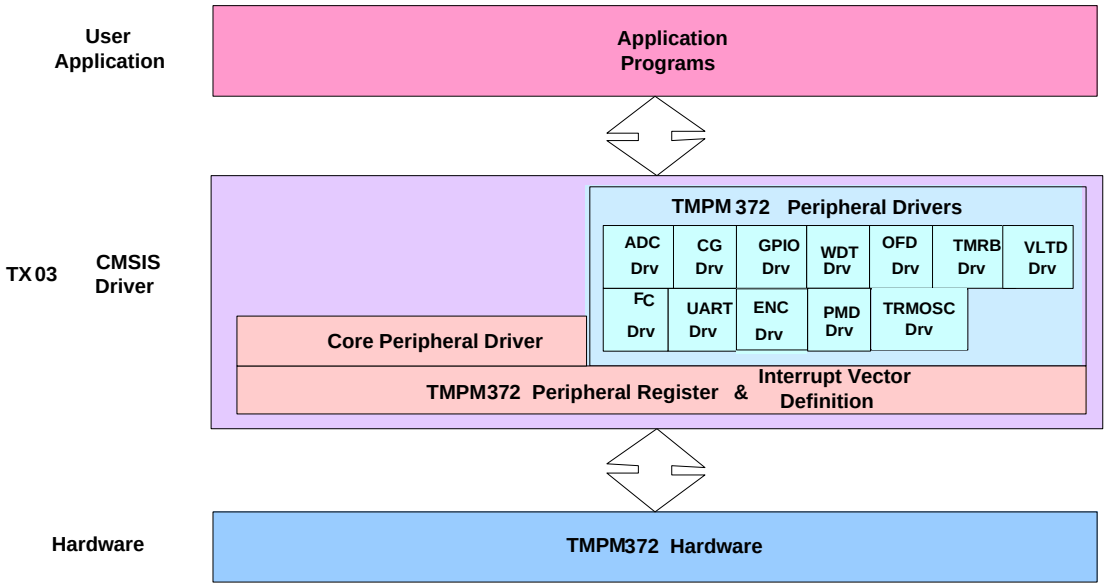
7-8-1 例: VLTD リセット 46
7-9 WDT 48
7-10 ENC 48
7-10-1 例: 回転検出 48
7-11 PMD 51
7-11-1 例: 位相出力 51
7-12 TRMOSC 54
7-12-1 例: TRMOSC 54

1 はしがき

本アプリケーションノートのサンプルプログラムは、東芝製マイコンTMPM372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、希望する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	使用(CG サンプル)
	PLL	PLL8 逓倍
低消費電力モード	-	使用: CG サンプルの STOP モードのみ
SysTick	-	未使用
ウォッチドッグタイマ (WDT)	-	使用: WDT サンプル
外部割り込み (INT)	INT0	未使用
	INT1	未使用

機能	チャネル	使用／未使用
	INT2	未使用
	INT3	未使用
	INT4	未使用
	INT5	未使用
	INT6	使用: CG サンプルにおける STOP モードからの復帰
	INT7	未使用
	INT8	未使用
	INT9	未使用
	INT10	未使用
シリアルチャネル (SIO/UART)	SIO0	使用: UART FIFO、SIO サンプル
	SIO1	使用: UART リターゲット、UART FIFO、SIO サンプル
	SIO2	未使用
	SIO3	未使用
16 ビットタイマ(TMRB)	TMRB0	使用: 汎用タイマー(TMRB)
	TMRB1	未使用
	TMRB2	未使用
	TMRB3	使用: TRMOSC サンプル(タイマ出力)
	TMRB4	未使用
	TMRB5	未使用
	TMRB6	使用: PPG 出力(TMRB) 使用: TRMOSC サンプル(基準信号入力)
	TMRB7	未使用
周波数検知回路(OFD)	-	使用: OFD サンプル
パワーオンリセット回路 (POR)	-	未使用
電圧検出回路(VLTD)	-	使用: VLTD サンプル
エンコーダ入力回路(ENC)	ENC1	使用: ホイールサンプル
モータ制御回路(PMD)	PMD1	使用: PMD 位相出力
12 ビット A/D コンバータ (ADC)	AINB2	使用: ADC データ読み出し
	AINB3	未使用
	AINB4	未使用
	AINB5	未使用
	AINB6	未使用
	AINB7	未使用
	AINB8	未使用
	AINB9	未使用
	AINB10	未使用
	AINB11	未使用
	AINB12	未使用

機能	チャネル	使用／未使用
内蔵高速発振調整機能 (TRMOSC)	-	使用: TRMOSC サンプル

4 端子用途

本サンプルプログラムは、TMPM372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。

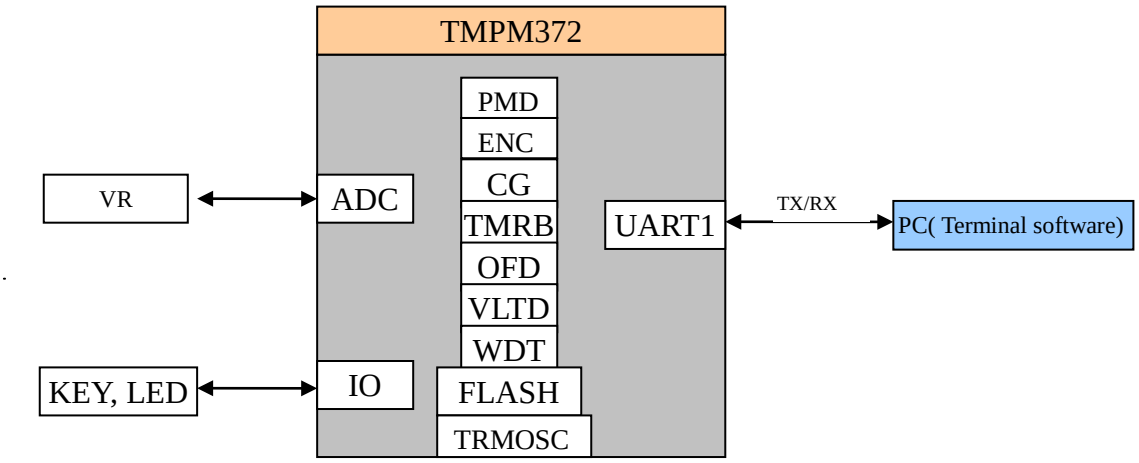
下表はサンプルプログラムで使用する端子表です。

No.	端子名	用途
1	SCLK2/CTS2/PD4	未使用
2	TX2/PD5	未使用
3	RX2/PD6	未使用
4	PG0/UO1	LED0 / PMD UO1 出力
5	PG1/XO1	LED1 / PMD XO1 出力
6	PG2/VO1	LED2 / PMD VO1 出力
7	PG3/YO1	LED3 / PMD YO1 出力
8	PG4/WO1	PMD WO1 出力
9	PG5/ZO1	PMD ZO1 出力
10	PG6/EMG1	PMD EMG1 入力
11	OVV1/PG7	未使用
12	DVSS	GND
13	DVDD5	+5V
14	BOOT/TB7IN/PF0	未使用
15	TRACECLK/PB0	未使用
16	TRACEDATA0/PB1	未使用
17	TRACEDATA1/PB2	未使用
18	PB7/TRST	未使用
19	PB6/TDI	未使用
20	PB5/TD0/SWV	未使用
21	PB4/TCK/SWCLK	未使用
22	PB3/TMS/SWDI0	PMD サンプル用LED4
23	PE0/TX0	未使用
24	PE1/RX0	未使用
25	PE2/SCLK0/CTS0	未使用
26	PE4/TB2IN/INT5	未使用
27	PF2/ENCA/SCLK3/CTS3	ENCA1
28	PF3/ENCB/TX3	ENCB1
29	PF4/ENCZ/RX3	未使用
30	PF1/TB7OUT	未使用
31	PE5/TB2OUT	未使用

No.	端子名	用途
32	PE3/TB4OUT	未使用
33	PA0/INT3/TB0IN	未使用
34	PA1/TB0OUT	未使用
35	PA2/INT4/TB1IN	未使用
36	DVDD5	+5V
37	PM0/X1	高速クロック(10MHz)
38	DVSS	GND
39	PM1/X2	高速クロック(10MHz)
40	VOUT15	未使用
41	MODE	未使用
42	RVDD5	+5V, VLTD検出端子
43	VOUT3	未使用
44	RESET	RESET
45	PI3/AINB2	POT1(VR1)入力
46	PJ0/AINB3	未使用
47	PJ1/AINB4	未使用
48	PJ2/AINB5	未使用
49	PJ3/AINB6	未使用
50	PJ4/AINB7	未使用
51	PJ5/AINB8	未使用
52	PJ6/AINB9/INTC	SW2/6
53	PJ7/AINB10/INTD	SW3/7
54	PK0/AINB11/INTE	SWS
55	PK1/AINB12/INTF	未使用
56	AVSSB/VREFLB	ADコンバータ用GND
57	AVDD5B/VREFHB	ADコンバータ用基準電源+5V
58	INT7/TB3OUT/PE7	SW1/5, TB3OUT
59	INT6/TB3IN/PE6	SW0/4
60	RX1/TB6IN/PA6	UART: RXD1 (UART1データ受信) TMRB:PPG波形出力 TRMOSC: TB6IN
61	TX1/TB6OUT/PA5	TXD1 (UART1データ転送)
62	SCLK1/CTS1/PA4	未使用
63	INT8/TB4IN/PA7	未使用
64	TB1OUT/PA3	未使用

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::
   TMPM372 評価ボード+TMPM372

2. 開発ツール:
   - IAR 社:
     1) J-Link: IAR 社製 J-Link-ARM 7.0 / J-Link 6.0
     2) IDE:IAR 社製 Embedded workbench 6.20 version
   - KEIL 社:
     1) IDE:KEIL uVision 4.14

補足 1: ENC/PMD サンプル動作環境は次の通りです。

- IAR:
  1) J-Link: IAR J-Link-ARM 7.0 / J-Link 6.0
  2) IDE: IAR Embedded workbench 6.50.1 version
- KEIL:
  1) IDE: KEIL uVision 4.60

補足 2: TRMOSC サンプル動作環境は次の通りです。

- IAR:
  3) J-Link: IAR J-Link-ARM 7.0 / J-Link 6.0
  4) IDE: IAR Embedded workbench 7.10.3 version
- KEIL:
  2) IDE: KEIL uVision 5.10

6 機能説明

6-1 動作モード

TMPM372 には 3 つの動作モードがあります: NORMAL, IDLE, STOP モード

- **NORMAL モード:**

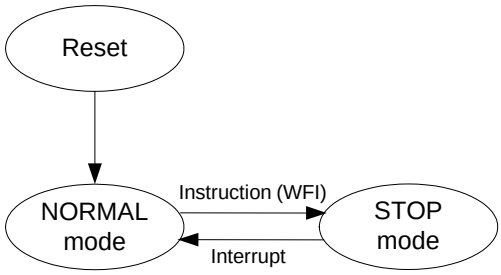
CPU コアおよび周辺ハードウェアを高速クロックで動作させるモードです。リセット解除後は、NORMALモードになります。
IDLEモードとSTOPモードは低消費電力モードです。
低消費電力モードへ移行するには、システ制御レジスタCGSTBYCR<STBY[2:0]>にてIDLEモードまたはSTOPモードを選択し、WFI (Wait For Interrupt)命令を実行します。

注: STOP モードのみサンプルプログラムに含まれます。

- **STOP モード:**

STOPモードでは、すべての内蔵回路が停止します。
STOPモードが解除されると、STOPモードへ移行する直前の動作モードへ復帰し、動作を開始します。
STOPモードでは、CGSTBYCR<DRVE>の設定により端子の状態を設定することができます。

注: STOP モードのサンプルプログラムは CG の例に含まれています。



6-2 ADC

PI3/AINB2 に接続された VR1(POT1)の値を変更します。測定された電圧値により、評価ボード上の 4 つの LED の点滅間隔を変更します。
電圧値が高いほど点滅間隔が短くなります。

6-3 CG

6-3-1 パワーモード変更

CPU の動作モードを変更します。NORMAL と STOP の 2 モードのみサポートします。 パワーモードを切り替えるにはキーを押してください。

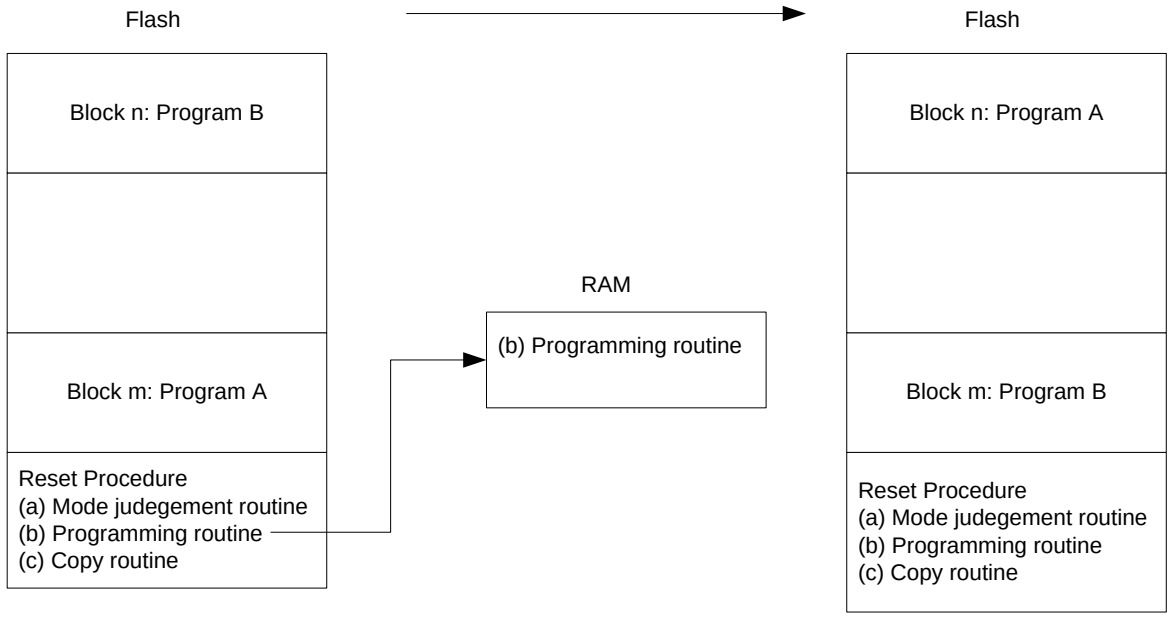
現在のモード	アクション (Key)	動作	LED 表示
NORMAL	SW1 を押す	NORMAL→ STOP	全 LED 消灯
STOP	SW0 を押す	STOP →NORMAL	全 LED 点灯

6-4 FLASH

このアプリケーションは、内蔵フラッシュメモリの消去及び再書き込み操作を行います。
–リセット動作: 書き込みルーチン(フラッシュ API)、転送ルーチンで構成されます。
▪モード判定ルーチン: ユーザブートモード、NORMAL モードをを判定します。
▪転送ルーチン: 書き込みルーチンを Flash から RAM へ転送します。
▪書き込みルーチン: RAM 上で動作し、block1 にデータを書き込みます。
ープログラム A/B (リセット処理)を予め Flash 領域に書き込んでおきます。
ープログラム A/B(A: LED1 点滅, B: LED2 点滅)
最初にプログラム A を実行します。
ーリセット解除直後のモード判定に SW1 を使います。

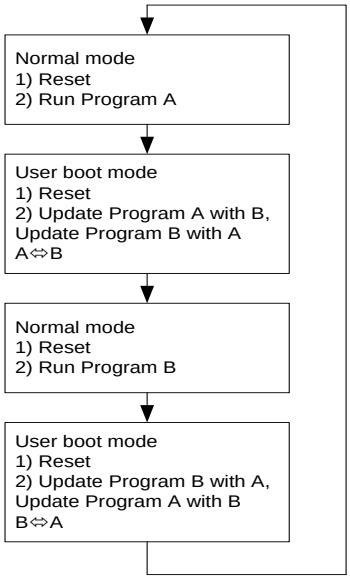
SW1 ON → ユーザブートモード

SW1 OFF → Normal モード

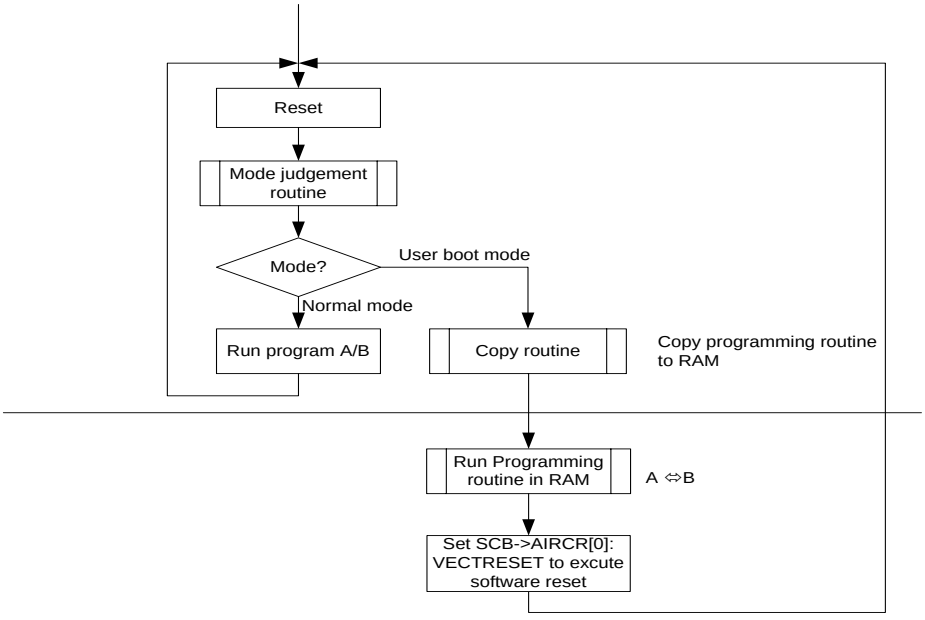


プログラムシーケンス

(1) 電源 ON
評価ボードのリセット動作が行われます。
Flash 上のプログラムが実行を始めます。
SW1 が押されていなければ、LED1 が点滅します。
(2) まず SW1 が押されていると LED3 を点灯し、書き込みルーチンが転送ルーチンにより RAM に転送されます。
次に Flash 上のプログラム A と B をスワップします。
(3) しばらくすると、LED3 は消灯し、スワップ完了したことを意味します。
その後、自動的にソフトウェアリセットを行い、Flash 上のプログラム B が LED2 を点灯します。



・プログラムのフローチャート



6-5 GPIO

この例では GPIO を LED に設定し、LED の点灯/消灯を行います。

6-6 OFD

この例では、OFD の初期化を行います。LED はシステムクロックの状態を表します。

6-7 SIO/UART

6-7-1 リターゲット

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。
サンプルプログラムでは、標準入力と標準出力を、共に UART に設定します。アプリケーションから printf()関数を使ってシリアルポートから出力を行います。

6-7-2 UART FIFO

この例では、UART0 から"TMPM3721"というデータを FIFO を使用した UART1 へ送信します。また同時に UART0 は"TMPM3722"というデータを FIFO を使用した UART1 から受信します。
ResetIdx()関数の前にブレークポイントを設定してください。RxBuffer=”TMPM3722”となり、RxBuffer1="TMPM3721"となると設定したブレークポイントで停止します。

6-7-3 SIO

この例では、TMPM372 の SIO モジュールを使用して同期式の送受信を行います。
SIO のチャネル 0 とチャネル 1 を使用し、これらのチャネル間で同期式データ送信を行います。
(TXD0 と RXD1、TXD1 と RXD0、sclk0 と sclk1 を接続してください)

6-8 TMRB

6-8-1 汎用タイマ

このサンプルは、MCU のタイマを使って、汎用タイマを実現します。
タイマー設定は 1ms 周期です。
このタイマーを使い 1s(500ms でオン、500ms でオフ)間隔で LED 点滅を行います。

6-8-2 PPG 波形出力

SW1 キーを使い、デューティ可変の PPG(プログラマブル矩形波)の波形を出力します。
デューティは 5 段階(10%, 25%, 50%, 75%, 90%)に変更できます。
電源投入すると PPG モードに入り、PPG 出力を開始します。
キーを押すことでデューティーを変更します。
10% → 25% → 50% → 75% → 90% → 10%

6-9 VLTD

この例では、VLTDによって検出される電圧状態を実装しています。
電圧が検出電圧より低い場合はハード的にMCUリセットが行われ、ソフト的にPG0にHIGH レベルを出力します。

6-10 WDT

ウォッチドッグタイマは高速クロックが停止する STOP モードでは使えません。リセットが行われ、その後ウォッチドッグタイマはが有効となり、SystemInit()関数で無効にしています。

本ドライバサンプルの流れは以下です。

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。

6-11 ENC

この例では、ENC ドライバを使用してホールマウスの回転を検知します。

動作シーケンス:

1. ホイールマウスエンコーダーの端子 A(図 1 参照)と端子 27 (ENCA1)を接続します。
2. ホイールマウスエンコーダーの端子 B(図 1 参照)と端子 28(ENCB1)を接続します。
3. USB マウスと PC を接続し、ホイールマウスエンコーダーの Com(図 1)を 5V と接続します。
4. プログラムを実行します(LED1 が点滅します)。
5. マウスのホイールを前に回転すると LED1 はより短い間隔で点滅します。点滅間隔が最も短くなると、点滅間隔は初めの間隔に戻ります。

6. マウスのホイールを後ろに回転するとLED1 はより長い間隔で点滅します。点滅間隔が最も長くなると、点滅間隔は初めの間隔に戻ります。

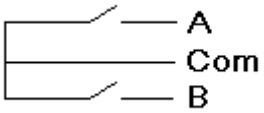


図 1

6-12 PMD

6-12-1 例: 位相出力

この例は、PMD ドライバを使用して位相出力を行います。

動作シーケンス:

1. プロジェクトを開き、DEMO_U_PHASE、または DEMO_3_PHASE と定義されたサンプルを選択します。プログラムを実行します。
2. EMG1 (pin 10)を DVDD5 (pin 13)とプルアップ接続します。(LED4 が消灯します)
3. UO1 (pin 4) をオシロスコープに接続し、波形を確認します。
4. XO1 (pin 5) をオシロスコープに接続し、波形を確認します。
5. VO1 (pin 6) をオシロスコープに接続し、波形を確認します。
6. YO1 (pin 7) をオシロスコープに接続し、波形を確認します。
7. WO1 (pin 8) をオシロスコープに接続し、波形を確認します。
8. ZO1 (pin 9) をオシロスコープに接続し、波形を確認します。

DEMO_U_PHASE 定義を選択した場合、位相波形は同じです。
UO1 のデューティは 25%、出力電圧は 5V、周期は 410us です。XO1 は UO1 を反転させた形です。VO1 と WO1 は UO1 と同じで、YO1 と ZO1 は XO1 と同じです。

DEMO_3_PHASE 定義を選択した場合、位相波形は異なります。
UO1 のデューティは 25%、出力電圧は 5V、周期は 410us です。XO1 は UO1 を反転させた形です。
VO1 のデューティは 50%、出力電圧は 5V、周期は 410us です。YO1 は VO1 を反転させた形です。
WO1 のデューティは 75%、出力電圧は 5V、周期は 410us です。ZO1 は WO1 を反転させた形です。

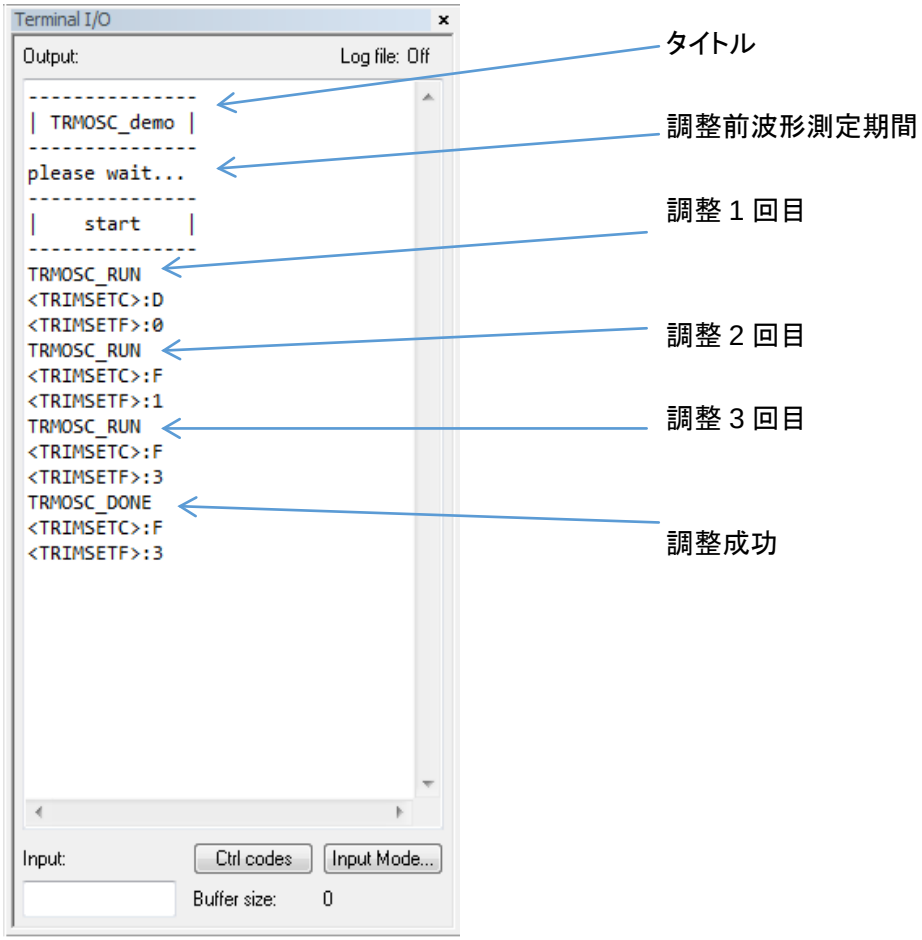
6-13 TRMOSC

内蔵高速発振調整機能(TRMOSC)ドライバを使用して内蔵高速発振器の調整を行うサンプルプログラムです。

外部より TB6IN へ基準信号を入力します。
信号の使用:

ボーレート	2400bps
データ長	8bit
パリティ	なし
ストップビット	1
送信データ	0xF0
周波数換算	240 Hz

サンプルプログラムにて誤差を計算、内蔵発振調整機能レジスタに調整値を設定します。
EWARM のターミナル I/O 機能を使用して調整結果を出力します。



TB3OUT の出力波形をモニタして確認します。

出力信号の仕様

調整後の周波数(期待値):2.375MHz

周波数の計算は次の通りです。

内蔵発振	: 9.5MHz
ソースクロック φT1	: 9.5MHz/2 = 4.75MHz
4.75MHz で TB3OUT を反転するため	: 4.75MHz/2 = 2.375MHz

7 ソフトウェア

本ソフトウェアは、TMPM372 の主要機能を TMPM372 評価ボード上で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
        │   tmpm372_wdt_int.c
        │   tmpm372_wdt_int.h
        │
        ├─TX03_CMSIS
        │   │   system_TMPM372.c
        │   │   system_TMPM372.h
        │   │   TMPM372.h
        │   │
        │   └─startup
        │         startup_TMPM372.s
        │
        ├─TX03_Periph_Driver
        │   ├─inc
        │   │   tmpm372_cg.h
        │   │   tmpm372_gpio.h
        │   │   tmpm372_wdt.h
        │   │   tx03_common.h
        │   │
        │   └─src
        │         tmpm372_cg.c
        │         tmpm372_gpio.c
        │         tmpm372_wdt.c
        │
        └─TMPM372-EVAL
                led.c
                led.h
                sw.c
                sw.h
```

KEIL MDK:

```
├─WDT_NMI
        ├─APP
        │   main.c
        │   tmpm372_wdt_int.c
        │
        ├─TX03_CMSIS
        │    system_TMPM372.c
        │    startup_TMPM372.s
        │
        ├─TX03_Periph_Driver
        │    tmpm372_cg.c
        │    tmpm372_gpio.c
        │    tmpm372_wdt.c
        │
        └─TMPM372-EVAL
             led.c
             sw.c
```

7-1 ADC

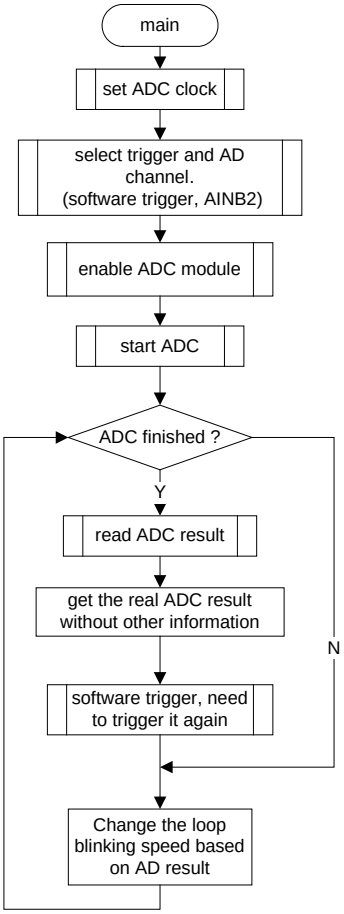
7-1-1 例: ADC データ読み出し

TX03 ペリフェラルドライバ(ADC, GPIO)を使用したサンプルプログラムです。

以下の例が含まれます:

1. ADC 設定と初期化
2. ソフトウェアトリガによる AD 変換を開始し、AD 変換結果を読み出します。
3. AD 変換結果により LED 点滅間隔が変化します。

- **フローチャート:**


main
set ADC clock
select trigger and AD channel. (software trigger, AINB2)
enable ADC module
start ADC
ADC finished ?
Y
N
read ADC result
get the real ADC result without other information
software trigger, need to trigger it again
Change the loop blinking speed based on AD result

- **サンプルプログラムのコードと説明**

まず ADC のクロック設定を行い、トリガタイプや AD チャネルを選択し、ADC モジュールを有効に設定します。

```
    /* 1. set ADC clock */
    ADC_SetClk(TSB_ADB, ADC_HOLD_FIX, ADC_FC_DIVIDE_LEVEL_2);

    /* 2. select trigger and AD channel, this time we use sofeware trigger, */
    /* the VR1 is connected to ADC unit B channel 2, remember to input with macro
TRG_ENABLE() */
    ADC_SetSWTrg(TSB_ADB, ADC_REG0, TRG_ENABLE(ADC_AIN2));

    /* 3. enable ADC module */
    ADC_Enable(TSB_ADB);
```

AD 変換を開始します:

```
    ADC_Start(TSB_ADB, ADC_TRG_SW);
```

AD 変換を行い、その後、ADC モジュールの状態を確認します。AD 変換終了時、他の AD 変換を開始します。

```
    while (1U) {
        /* check ADC module state */
        adcState = ADC_GetConvertState(TSB_ADB, ADC_TRG_SW);

        if (adcState == DONE) {
            /* read ADC result when it is finished */
            result = ADC_GetConvertResult(TSB_ADB, ADC_REG0);

            /* get the real ADC result without other information */
            myResult = result.Bit.ADResult;

            /* software trigger, need to trigger it again */
            ADC_Start(TSB_ADB, ADC_TRG_SW);
        }
```

“myResult”の AD 変換結果を確認後、4 つの LED の点滅間隔を変更します。

7-2 CG

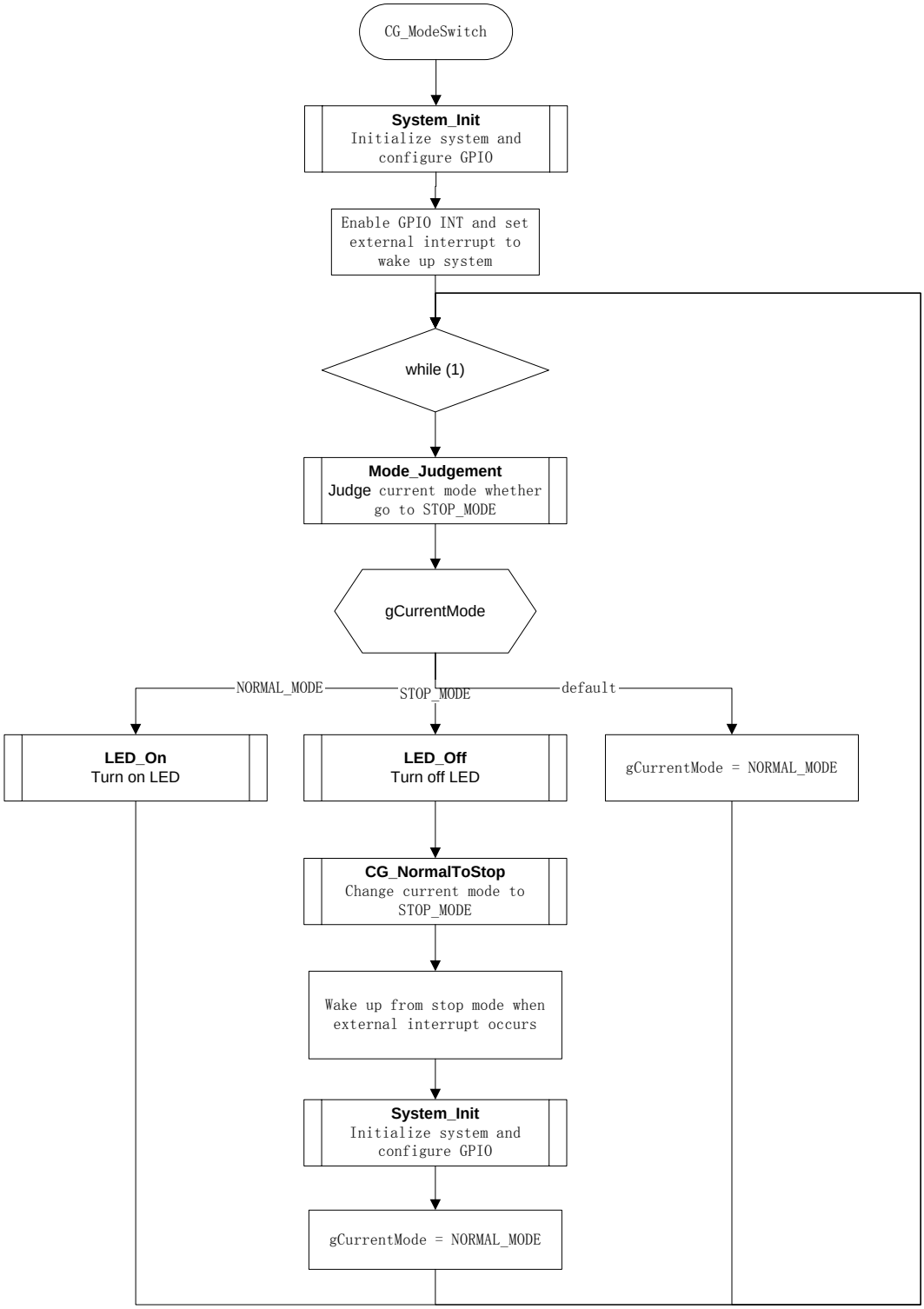
7-2-1 例: パワーモード変更

TX03 ペリフェラル・ドライバ(CG, GPIO)を使用したサンプルプログラムです。

以下の例が含まれます:

1. 基本的な CG 動作の設定
2. normal モードと stop モードの切り替え方法

- **フローチャート**


CG_ModeSwitch
System_Init
Initialize system and configure GPIO
Enable GPIO INT and set external interrupt to wake up system
while (1)
Mode_Judgement
Judge current mode whether go to STOP_MODE
gCurrentMode
NORMAL_MODE
STOP_MODE
default
LED_On
Turn on LED
LED_Off
Turn off LED
gCurrentMode = NORMAL_MODE
CG_NormalToStop
Change current mode to STOP_MODE
Wake up from stop mode when external interrupt occurs
System_Init
Initialize system and configure GPIO
gCurrentMode = NORMAL_MODE

- **サンプルプログラムのコードと説明**

以下は NORMAL モードと STOP モードの切り替えを行うプログラム例です。

通常の CG の初期化(リセット後)

以下は NORMAL モードで CG の設定を行うプログラム例です。(高速発振器 10MHz の場合)

サンプルプログラムのコード:

```
/* Set fgear = fc/2 */
CG_SetFgearLevel(CG_DIVIDE_2);
/* Set fperiph to fgear */
CG_SetPhiT0Src(CG_PHIT0_SRC_FGEAR);
CG_SetPhiT0Level(CG_DIVIDE_64);
/* select external oscillator */
CG_SetFoscSrc(CG_FOSC_OSC1);
CG_SetPortM(CG_PORTM_AS_HOSC);
/* Enable high-speed oscillator */
CG_SetFosc(CG_FOSC_OSC1, ENABLE);
/* Set low power consumption mode stop */
CG_SetSTBYMode(CG_STBY_MODE_STOP);
/* Set pin status in stop mode to "active" */
CG_SetPinStateInStopMode(ENABLE);
/* Set up pll and wait for pll to warm up, set fc source to fpll */
CG_EnableClkMulCircuit();
```

スタンバイ解除のための外部割込みの設定

スタンバイ解除のために INT6 の設定を行います。割り込み保留要求レジスタをクリアし、INT6 を有効にします。

```
/* Set CG module: Normal ->Stop mode */
/* Disable interrupts */
__disable_irq();
/* Enable GPIO INT */
CG_ClearINTReq(CG_INT_SRC_6);
/* Set external interrupt to wake up system */
CG_SetSTBYReleaseINTSrc(CG_INT_SRC_6,
                        CG_INT_ACTIVE_STATE_FALLING, ENABLE);
NVIC_ClearPendingIRQ(INT6_IRQn);
NVIC_EnableIRQ(INT6_IRQn);
__enable_irq();
```

STOP モード設定

STOP モードに入る設定を行います。ウォームアップ時間を設定し、__WFI()命令を使用して STOP モードに入ります。

```
CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC1, CG_WUODR_EXT);
__DSB();
/* Enter stop mode */
__WFI();
```

マルチクロック回路の許可

まず PLL を設定します。そしてウォームアップ時間を設定し、ウォームアップ時間が完了するまで待ちます。その後、fPLL を fc ソースとして設定します。

```
WorkState st = BUSY;
```

```
retval = CG_SetPLL(ENABLE);
if (retval == SUCCESS) {
	/* Set warm up time */
	CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC1, CG_WUODR_PLL);
	CG_StartWarmUp();

	do {
		st = CG_GetWarmUpState();
	} while (st != DONE);

	retval = CG_SetFcSrc(CG_FC_SRC_FPLL);
} else {
	/*Do nothing */
}
```

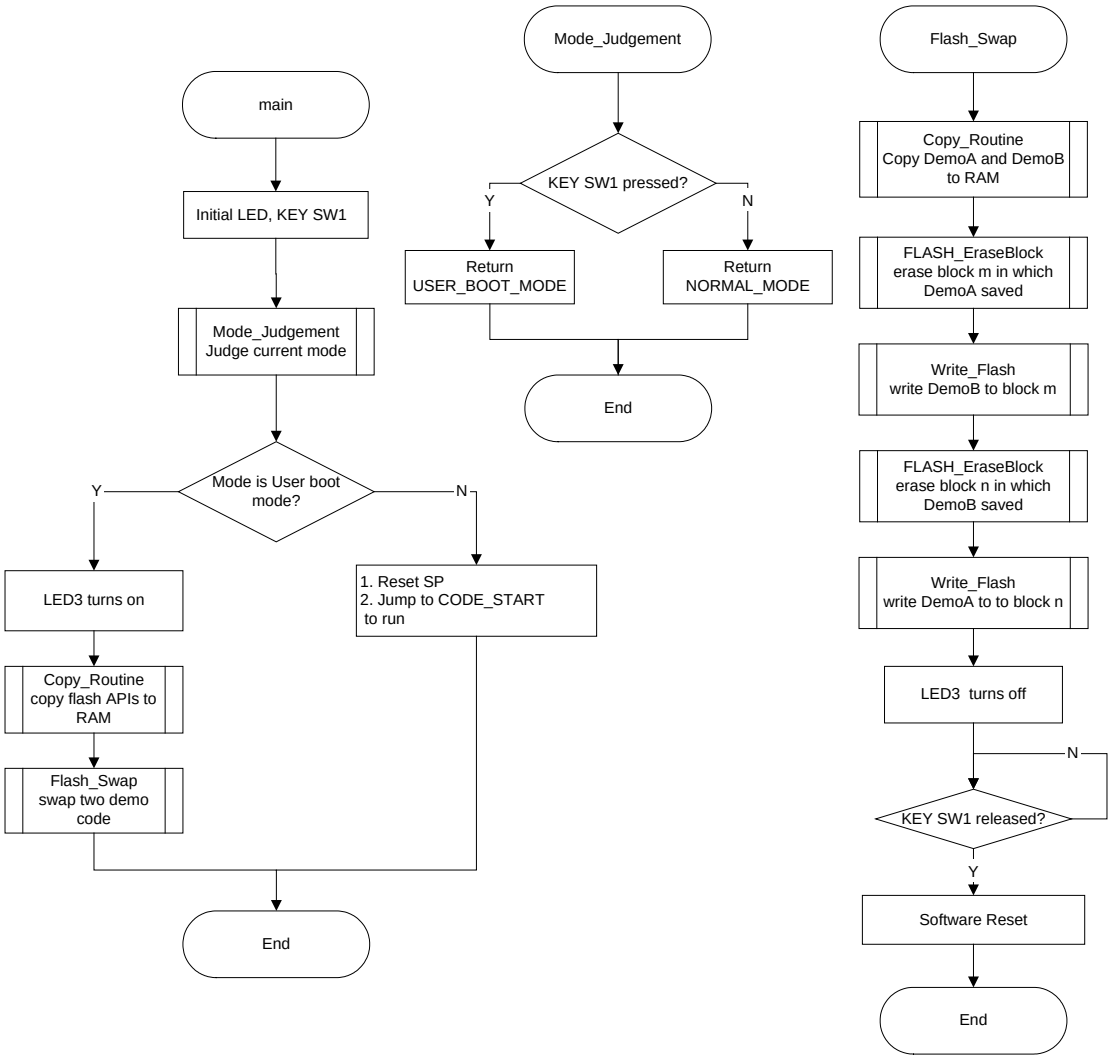
7-3 FLASH

TX03 ペリフェラルドライバ(FLASH, GPIO)を使用したサンプルプログラムです。

以下の例が含まれます:

1. FLASH メモリのオンボードプログラミング (書き込み/消去)
2. 動作モード: シングルチップモード(Normal モードとユーザブートモード)
3. ユーザブートモードを使用した Flash ROM の書き換え

- **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 LED とスイッチを初期化します。

```
    LED_Init();
    SW1_Init();
```

リセット後のモード判定に Mode_Judgement()関数を使います。

```
uint8_t Mode_Judgement(void)
{
    return (SW1_Get() == 1U) ? USER_BOOT_MODE : NORMAL_MODE;
}
```

リセット時に SW1 が OFF の場合、通常処理を行います。この処理は、SP を初期化し、アドレス"CODE_START"へジャンプした後、動作します。サンプル A はブロック m に属するアドレス"CODE_START"に格納され、動作します。このとき LED1 が点滅します。

```
#if defined ( __CC_ARM      )           /* RealView Compiler */
        ResetSP();                      /* reset SP */
#elif defined ( __ICCARM__ )            /* IAR Compiler */
        asm("MOV R0, #0");              /* reset SP */
        asm("LDR SP, [r0]");
#endif
        SCB->VTOR = DEMO_START_ADDR; /* redirect vector table */
        startup = CODE_START;
        startup();                      /* jump to code start address to run */
```

SW1 が ON の場合、動作モードはユーザブートモードに移行します。このモードでは LED3 を点灯します。フラッシュメモリ上で実行しながらフラッシュメモリ自身を消去/プログラムできないため、Flash ROM のアドレス"FLASH_API_ROM"から RAM のアドレス "FLASH_API_RAM"へ、フラッシュ操作するための API をコピーします。

```
    LED_On(LED3);                       /* enter user boot mode */
    Copy_Routine(FLASH_API_RAM, FLASH_API_ROM, SIZE_FLASH_API);
    /* copy flash API to RAM */
```

Flash 操作用 API を RAM にコピーした後、Flash_Swap()関数にジャンプします。
この関数は、Copy_Routine()によって Flash ROM から RAM へコピーされています。

Flash_Swap()関数では、まずサンプル A/B を Flash ROM から RAM へコピーします。次に、Flash ROM を書き換えるため FC_EraseBlock()関数と Write_Flash()関数をコールします。
サンプル A とサンプル B のコードは、Flash ROM 内でスワップします。

```
    Copy_Routine(DEMO_A_RAM, DEMO_A_FLASH, SIZE_DEMO_A);
    /* copy A to RAM */
    Copy_Routine(DEMO_B_RAM, DEMO_B_FLASH, SIZE_DEMO_B);
    /* copy B to RAM */

    /* erase A */
    if (FC_SUCCESS == FC_EraseBlock((uint32_t) DEMO_A_FLASH)) {
        /* Do nothing */
    } else {
        return ERROR;
    }

    /* write B to A */
    if (FC_SUCCESS == Write_Flash(DEMO_A_FLASH, DEMO_B_RAM,
```

```
SIZE_DEMO_B)) {
            /* Do nothing */
        } else {
            return ERROR;
        }

        /* erase B */
        if (FC_SUCCESS == FC_EraseBlock((uint32_t) DEMO_B_FLASH)) {
            /* Do nothing */
        } else {
            return ERROR;
        }

        /* write A to B */
        if (FC_SUCCESS == Write_Flash(DEMO_B_FLASH, DEMO_A_RAM,
SIZE_DEMO_A)) {
            /* Do nothing */
        } else {
            return ERROR;
        }
```

スワップ動作が完了すると、LED3 は消灯します。SW1 が OFF した後、SCB->AIRCR レジスタによるソフトウェアリセットを実行します。
次にこのプログラムは、再度通常処理を行います。サンプル A/B は入れ替わっているため、アドレス"CODE_START"はサンプル B のスタートアドレスになります。
このとき LED2 が点滅します。

```
    delay(0x7FFFFF);
    LED_Off(LED3);                          /* complete and restart */
    delay(0x2FFFFF);

    /* wait for Key SW1 to release*/
    while (SW1_Get() == 1U) {
    }

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

Flash ROM 操作関数 FC_EraseBlock()は指定されたブロックを消去します。消去するブロックは"BlockAddr"で指定します。まず"BlockAddr"の値を確認し、次に指定されたブロックのプロテクト状態を FC_GetBlockProtectState()関数をコールして確認します。

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

ブロックがプロテクトされていた場合、“FC_ERROR_PROTECTED”を返し、プロテクトされていなかった場合はブロック消去コマンドを実行します。

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

次に、FC_GetBusyState()関数をコールして、消去動作の完了を監視します。また同時に timeout_counter を用いてタイムアウト処理も行います。

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

Write_Flash()関数は、FC_WritePage()関数をコールして、自動プログラムコマンドを実行します。

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

7-4 GPIO

TX03 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。

サンプルプログラムは以下を含みます。

1. GPIO の初期化
2. GPIO へのデータ書き込み
3. GPIO からのデータ読み出し

- **サンプルプログラムのコードと説明**

まず GPIO_SetOutput(GPIO_Port GPIO_x, uint8_t Bit_x)を使用して GPIO を LED に設定します。そして GPIO_SetInput(GPIO_Port GPIO_x, uint8_t Bit_x)を使用して GPIO をキーに設定します。以下に例を示します。

```
GPIO_SetOutput(GPIO_PG, GPIO_BIT_0);
GPIO_SetOutput(GPIO_PG, GPIO_BIT_1);
GPIO_SetOutput(GPIO_PG, GPIO_BIT_2);
GPIO_SetOutput(GPIO_PG, GPIO_BIT_3);

GPIO_SetInput(GPIO_PE, GPIO_BIT_6 | GPIO_BIT_7);
GPIO_SetInput(GPIO_PJ, GPIO_BIT_6 | GPIO_BIT_7);
GPIO_SetOutput(GPIO_PK, GPIO_BIT_0);
```

for(; ;)ループの中で、LED サンプルを実行します: スイッチ状態をリードし、LED オンや LED オフを制御します。

GPIO_ReadDataBit(GPIO_Port GPIO_x, uint8_t Bit_x)を使用してスイッチ状態をリードします。

```
if (GPIO_ReadDataBit(GPIO_PE, GPIO_BIT_6) == 1U) {
        tmp = 1U;
    } else {
        /*do nothing */
    }
```

GPIO_WriteDataBit(GPIO_Port GPIO_x, uint8_t Bit_x, uint8_t BitValue)を使用して LED をオンします。

```
uint32_t tmp;
tmp = GPIO_ReadData(GPIO_PG);
tmp |= led;
GPIO_WriteData(GPIO_PG, tmp);
```

GPIO_WriteDataBit(GPIO_Port GPIO_x, uint8_t Bit_x, uint8_t BitValue)を使って LED をオフします。

```
uint32_t tmp;
tmp = GPIO_ReadData(GPIO_PG);
tmp &= ~led;
GPIO_WriteData(GPIO_PG, tmp);
```

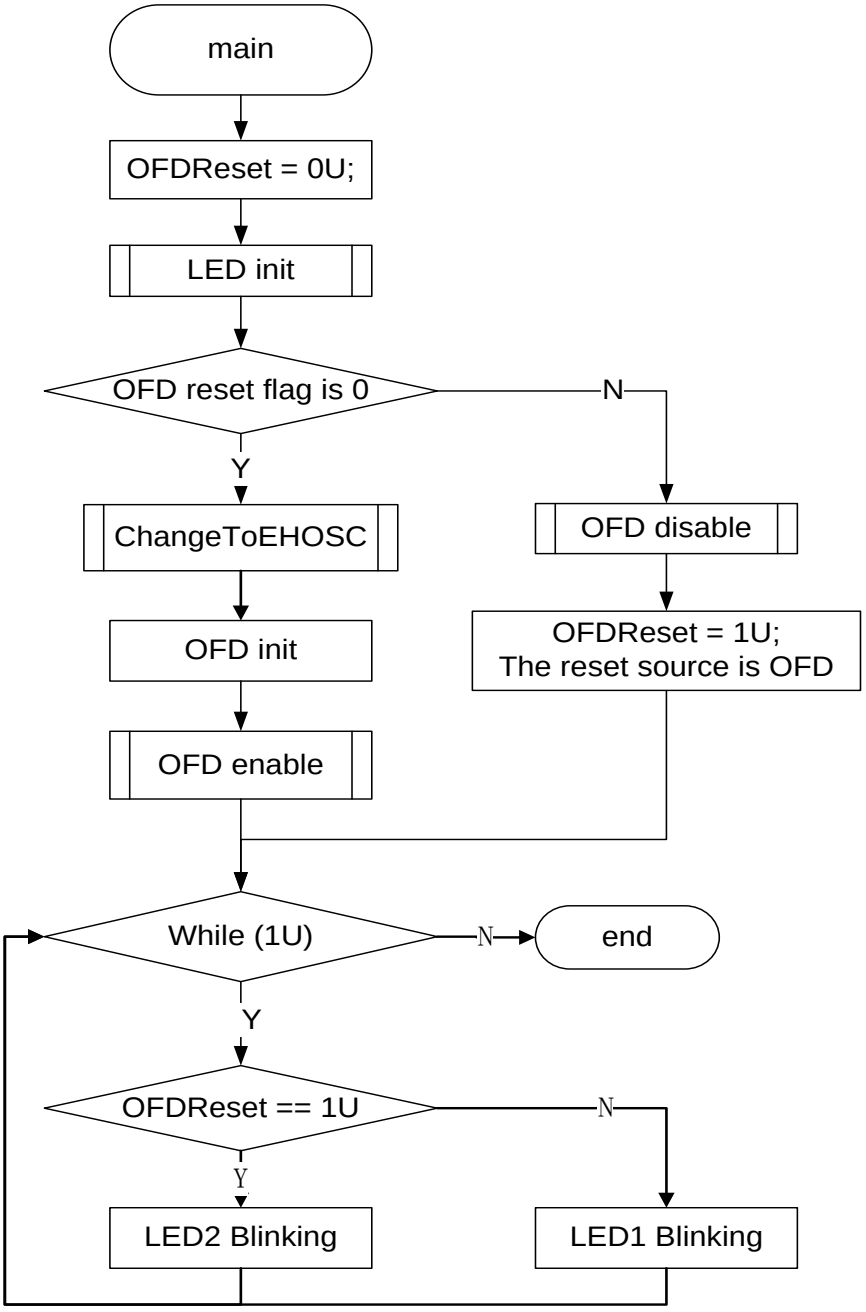
7-5 OFD

TX03 ペリフェラルドライバ(OFD, CG, GPIO)を使用したサンプルプログラムです。

以下の例が含まれます:

1. OFD の初期化(検知周波数範囲を OFD に設定します)
2. OFD リセットフラグの確認
3. OFD の有効／無効

- **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、LED を初期化します。

```
LED_Init ();
```

OFD のリセットフラグを確認します。

```
resetFlag = CG_GetResetFlag();
if (resetFlag.Bit.OFDReset == 0U) {
  /* reset source isn't OFD */
}
```

この OFD フラグが'0'の場合(通常リセットの場合、電源投入にて MCU がリセットされると、この OFD フラグは'0'になります)、プログラムにて外部発振器を選択、検知周波数範囲を OFD に設定し、OFD を初期化します。

下記コードは、外部発振器を使うための CG の設定を行います。

```
void ChangeToEHOSC(void)
{
    /* Use the external oscillation */
    TSB_CG->OSCCR &= OSCCR_WUODR_MASK;
    TSB_CG->OSCCR |= OSCCR_WUODR_EXT;          /*Set the warm up time WUODR[13:2]*/
    TSB_CG_OSCCR_HOSCON = 1U;     /*Set gpio port as X1/X2 for external oscillator */
    TSB_CG_OSCCR_XEN1 = 1U;       /*Enable external oscillator */
    TSB_CG_OSCCR_WUPSEL2 = 1U;
    TSB_CG_OSCCR_WUPSEL1 = 0U;    /*Select warm-up clock */
    TSB_CG_OSCCR_WUEON = 1U;      /*Start warm up */
    while (TSB_CG_OSCCR_WUEF) {}/* Warm-up */
    TSB_CG_OSCCR_OSCSEL = 1U;     /*Use the external oscillation */
    TSB_CG_OSCCR_XEN2 = 1U;       /*Enable internal oscillator for ofd */
    TSB_CG_OSCCR_PLLON = 1U;/**/
    TSB_CG_OSCCR_WUEON = 1U;
    while(TSB_CG_OSCCR_WUEF) {    /* Warm-up          */
        /* Do nothing */
    }
  TSB_CG->PLLSEL = PLLSEL_Val;
  TSB_CG->CKSEL  = CKSEL_Val;
}
```

OFD の設定を行うため、まずレジスタ書き込み許可コードを設定します。

```
OFD_SetRegWriteMode(ENABLE);
```

PLL ON 時の検知周波数の設定を行います。

```
OFD_SetDetectionFrequency(OFD_PLL_ON, OFD_HIGHER_COUNT,
                                    OFD_LOWER_COUNT);
```

DEMO2 を定義すると異常な周波数範囲が設定されます。

```
OFD_SetDetectionFrequency(OFD_PLL_ON,
    OFD_HIGHER_COUNT_ABNORMAL, OFD_LOWER_COUNT_ABNORMAL);
```

初期化し、その後、OFD を有効にします。

```
OFD_Enable();
```

上記の設定を行い、その後 OFD は外部クロックを検知します。このクロックが検知周波数範囲を超える(例えば外部発振が範囲外になる、DEMO2 が定義されている)と OFD は CG リセット要因フラグをセットして、MCU をリセットします。
OFD リセットフラグを検知すると、MCU は内蔵発信にて動き始め、OFD 動作を無効化します。

```
OFD_Disable();
OFDReset = 1U;
```

LED1 と LED2 はシステムクロックの状態を表します。

LED の状態	意味
LED1 点滅	MCU は外部クロック発振にて正常動作しています。OFD が機能しており、異常周波数を検知することが可能です。
LED2 点滅	外部クロックが異常状態であり、OFD リセットを行い、MCU はデフォルトの内部クロックで動作します。また OFD は機能していません。

```
while (1U) {
    if (OFDReset == 1U) {
        LED_On(LED2);
        Delay();
        LED_Off(LED2);
        Delay();
    } else {
        LED_On(LED1);
        Delay();
        LED_Off(LED1);
        Delay();
    }
}
```

7-6 TMRB

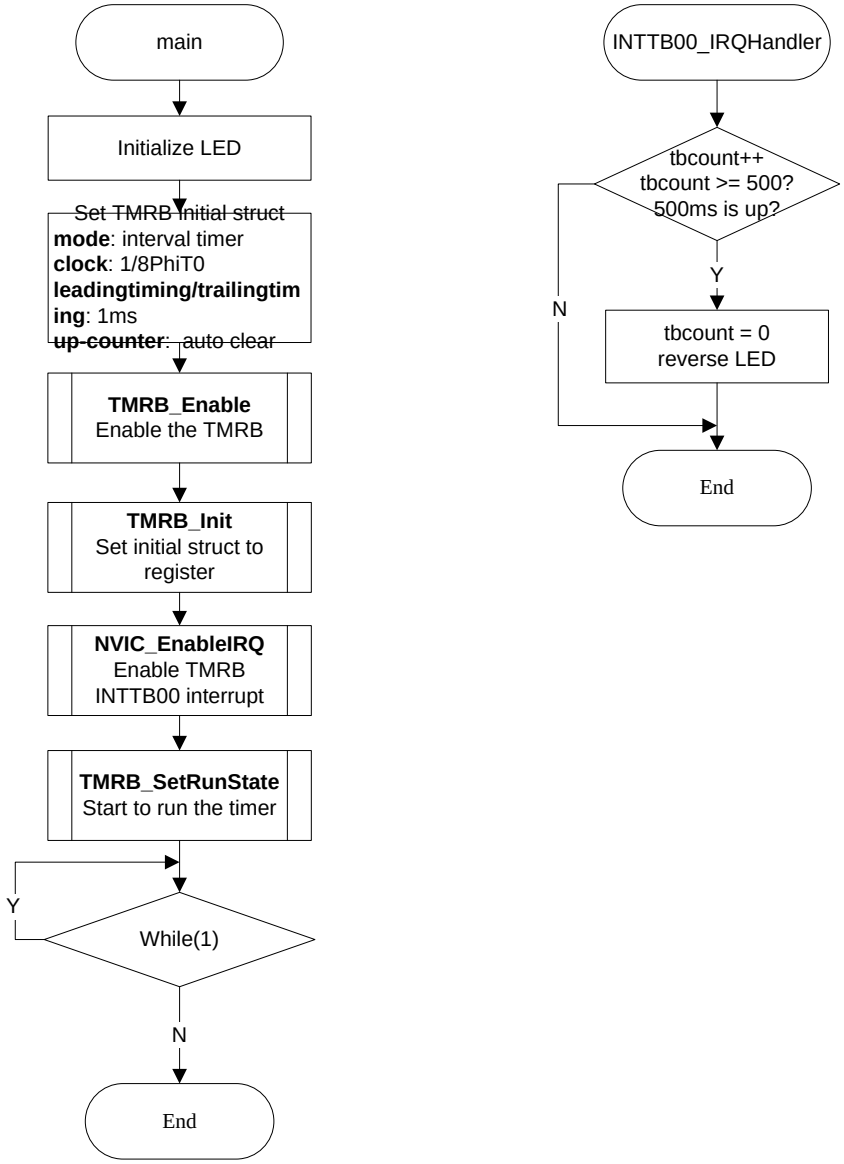
7-6-1 例: 汎用タイマ

TX03 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。

以下の例が含まれます:

1. TMRB0 の初期化
2. 1ms の汎用タイマー

- **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 LED チャネルを初期化し、評価ボード上の LED をオンします。

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

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メソッド、サイクル、デューティーを設定します。本サンプルプログラムでは、サイクルとデューティーを 1ms に設定します。1ms のカウント値は 0x2710(TMRB_1MS マクロ定義)です。
算出式:phiT0 = fsys = fc = 10MHz * PLL = 80MHz, ftmrb = 1/8 fphiT0 = 10MHz, Ttmrb = 0.1us, 1ms/0.1us = 10000 = 0x2710 (クロック設定に関する詳細は、データシートのクロック/モード制御章を参照ください)

```
TMRB_InitTypeDef m_tmrb;

m_tmrb.Mode = TMRB_INTERVAL_TIMER;      /* internal timer */
m_tmrb.ClkDiv = TMRB_CLK_DIV_8;         /* 1/8PhiT0 */
m_tmrb.TrailingTiming = TMRB_1MS;       /* periodic time is 1ms */
m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;     /* up-counter auto clear */
m_tmrb.LeadingTiming = TMRB_1MS;        /* periodic time is 1ms */
```

TMRB モジュールを有効にした後、初期化構造体を指定のレジスタに設定します。INTTB00 割り込み(1ms ごとにトリガ)を有効にします。最後に TMRB を動作させます。

```
TMRB_Enable(TSB_TB0);                          /* enable the TMRB0 */
TMRB_Init(TSB_TB0, &m_tmrb);                   /* initial the TMRB0 */
NVIC_EnableIRQ(INTTB00_IRQn);                  /* enable INTTB00 interrupt */
TMRB_SetRunState(TSB_TB0, TMRB_RUN);           /* run TMRB0*/
```

メインルーチンは、"While(1) "に入り、割り込みの発生を待ちます。割り込みルーチンでは、カウンタでカウントを行い、500ms をカウントすると、 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 */
}
```

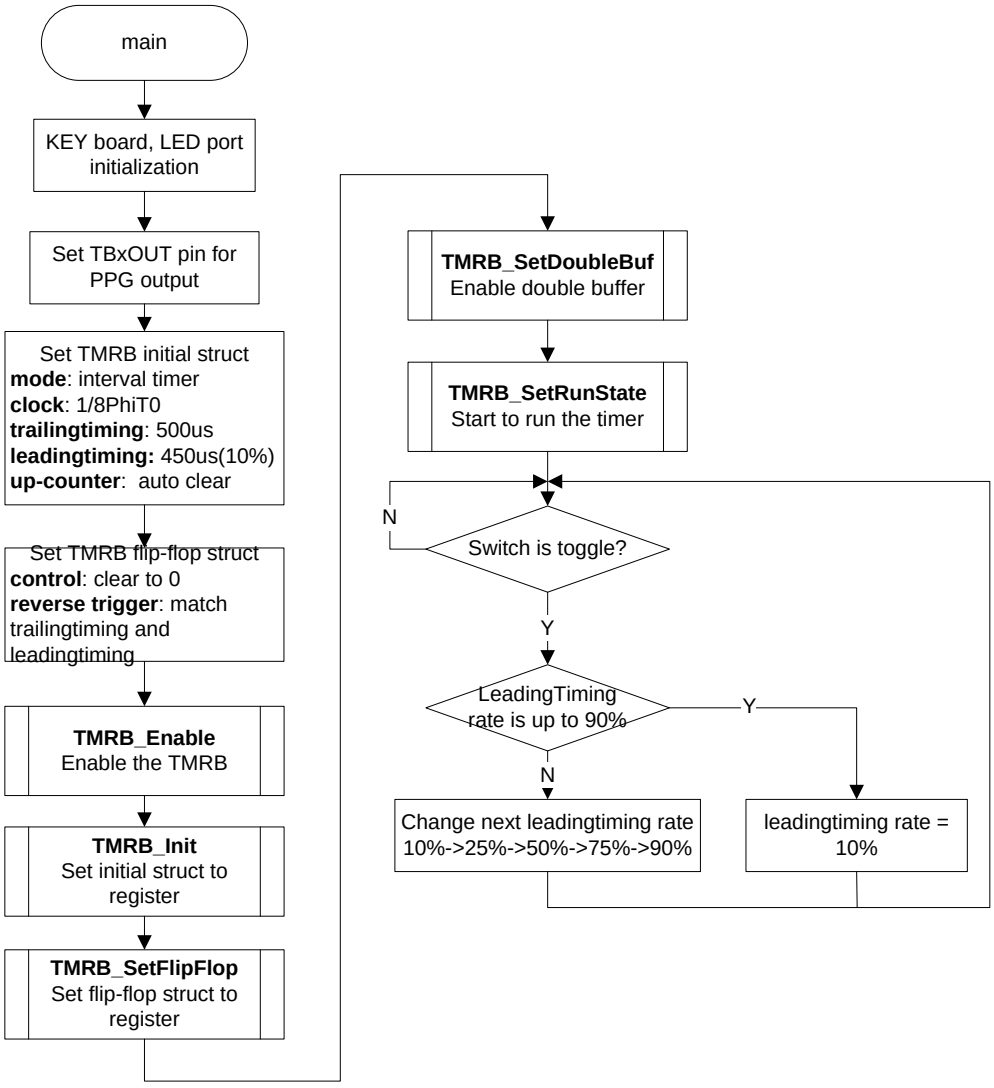
7-6-2 例: PPG 波形出力

TX03 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。

以下の例が含まれます:

1. TMRB6 の初期化
2. PPG 機能の設定と開始
3. PPG デューティーの調整

- **フローチャート**



- **サンプルプログラムのコードと説明**

最初にスイッチと LED を初期化し、PA5 を PPG 出力するために TB6OUT に設定します。

```
GPIO_SetInput(KEYPORT, GPIO_BIT_6);                    /* set KEY port to input */

LED_Init();                                            /* LED initialization*/

/* Set PA5 as TB6OUT for PPG output */
GPIO_SetOutput(GPIO_PA, GPIO_BIT_5);
GPIO_EnableFuncReg(GPIO_PA, GPIO_FUNC_REG_2, GPIO_BIT_5);
```

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メソッド、サイクル、デューティーを設定します。本サンプルでは 500us の周期を設定します。500us のカウント値は 0x09C4(TMRB6TIME マクロ定義)です。

fphiT0 = fsys = fc = 10MHz * PLL = 80MHz, ftmrb = 1/8 fphiT0 = 10MHz, Ttmrb = 0.1us, 500us/0.1us = 5000 = 0x1388 (クロック設定に関する詳細は、データシートのクロック/モード制御章を参照ください)

```
TMRB_InitTypeDef m_tmrb;

m_tmrb.Mode = TMRB_INTERVAL_TIMER;      /* internal timer */
m_tmrb.ClkDiv = TMRB_CLK_DIV_8;          /* 1/8PhiT0 */
m_tmrb.TrailingTiming = TMRB6TIME;       /* trailing timing is 500us */
m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;      /* up-counter auto clear */
m_tmrb.LeadingTiming = LeadingTiming[Rate]; /* leading timing, initial value 10% */
```

フリッププロップ初期化構造体を用意し、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、デューティーとサイクルを一致するように設定します。

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

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_TB6);
TMRB_Init(TSB_TB6, &m_tmrb);
TMRB_SetFlipFlop(TSB_TB6, &PPGFFInital);
/* enable double buffer */
TMRB_SetDoubleBuf(TSB_TB6,ENABLE, TMRB_WRITE_REG_SEPARATE);
TMRB_SetRunState(TSB_TB6, TMRB_RUN);
```

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

```
do {                                /* wait if switch is Low */
        keyvalue = GPIO_ReadDataBit(KEYPORT, GPIO_BIT_7);
        LeadingTiming_display(); /* display current leading timing */
    } while (GPIO_BIT_VALUE_0 == keyvalue);
```

スイッチが High の場合、下記のようにデューティーを設定します。

10%->25%->50%->75% ->90%その後 再び 90%から 10%になります。

```
Rate++;
if (Rate >= LEADINGTIMINGMAX) {
```

```
        Rate = LEADINGTIMINGINIT;
        } else {
            /* Do nothing */
        }

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

デューティの算出方法:
Trailing timing = 500us, ftmrb = 1/8 fphiT0 = 10MHz, Ttmrb = 0.1us (これらの時間に関するパラメータは、CG 設定により異なります)

Leading timing = 10%: High 幅は 500*10% = 50us, Low 幅は 500-50 = 450us, カウンタ値= 450us/Ttmrb = 0x1194

Leading timing = 25%: High 幅は 500*25% = 125us, Low 幅は 500-125 = 375us, カウンタ値= 375us/Ttmrb = 0xEA6

Leading timing = 50%: High 幅は 500*50% = 250us, Low 幅は 500-250 = 250us, カウンタ値= 250us/Ttmrb = 0x9C4

Leading timing = 75%: High 幅は 500*75% = 375us, Low 幅は 500-375 = 125us, カウンタ値= 125us/Ttmrb = 0x4E2

Leading timing = 90%: High 幅は 500*90% = 450us, Low 幅は 500-450 = 50us, カウンタ値= 50us/Ttmrb = 0x1F4

これは上記計算式から求めたデューティ値の配列です。

```
    uint32_t LeadingTiming[5] = { 0x1194U, 0xEA6U, 0x9C4U, 0x4E2U, 0x1F4U };
    /* leading timing: 10%, 25%, 50%, 75%, 90% */
```

7-7 SIO/UART

7-7-1 例: リターゲット

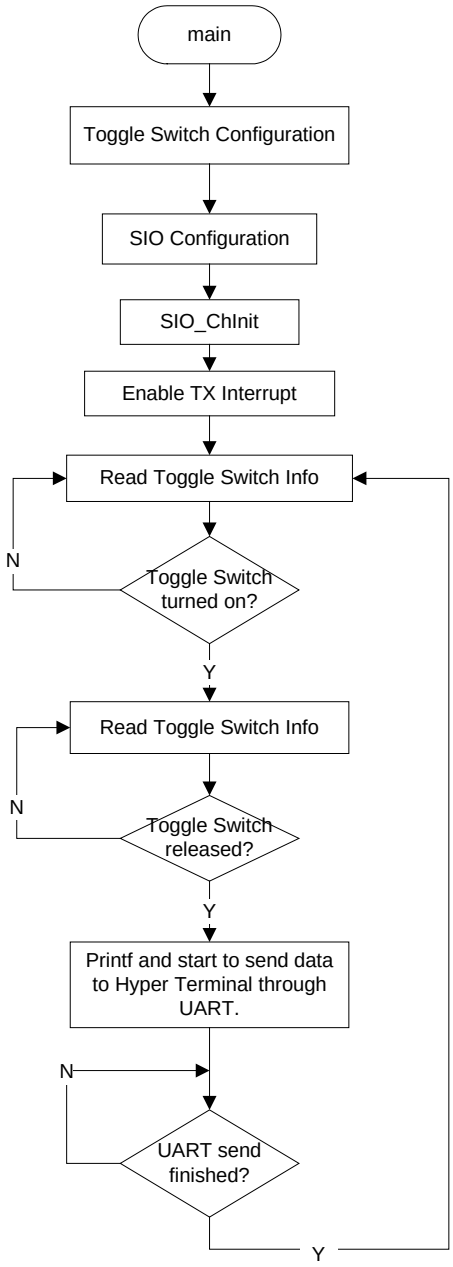
TX03 ペリフェラルドライバ (UART, GPIO)を使用したサンプルプログラムです。

以下の例が含まれます:

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

この UART チャネルはハードウェアにより制限されています。下記の例では UART1 を使用しています。

- **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 をスイッチに設定します。

```
GPIO_SetOutputEnableReg(GPIO_PA, GPIO_BIT_5, ENABLE);
GPIO_EnableFuncReg(GPIO_PA, GPIO_FUNC_REG_1, GPIO_BIT_5);
GPIO_EnableFuncReg(GPIO_PA, GPIO_FUNC_REG_1, GPIO_BIT_6);
GPIO_SetInputEnableReg(GPIO_PA, GPIO_BIT_6, ENABLE);
```

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;

myUART.BaudRate = 115200U;    /* baut 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_RX | UART_ENABLE_TX;
 myUART.FlowCtrl = UART_NONE_FLOW_CTRL;
```

UART を許可し、UART チャネルの初期化を行います。

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

その後、UART の送信割り込みを有効にします。

```
NVIC_EnableIRQ(INTTX1_IRQn);
```

データ送信を開始します。TxBuffer は文字列配列です。

```
UART_SetTxData(UART1, TxBuf[0]);
```

データフローの残りのプロセスは UART1 送信割り込みルーチンの ISR にて終了します。
UART1 の送信割り込みルーチン:

```
void INTTX1_IRQHandler(void)
{
    if (gSIORdIndex < gSIOWrIndex) {       /* buffer is not empty */
        UART_SetTxData(UART_RETARGET,          gSIOTxBuffer[gSIORdIndex++]);
/* send data */
        fSIO_INT = SET;              /* SIO0 INT is enable */
    } else {
        /* disable SIO0 INT */
        fSIO_INT = CLEAR;
        NVIC_DisableIRQ(INTTX1_IRQn);
        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 */
```

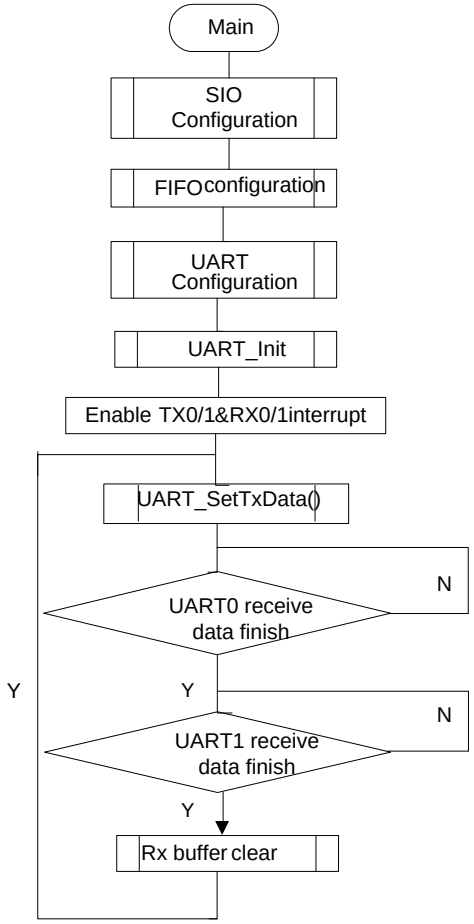
```
struct __FILE {
      int handle;                         /* Add whatever you need here */
};
FILE __stdout;
FILE __stdin;
int fputc(int ch, FILE * f)
#elif defined ( __ICCARM__ )       /*IAR Compiler */
int putchar(int ch)
#endif
{
      return (send_char(ch));
}

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(UART0, gSIOTxBuffer[gSIORdIndex++]);
            NVIC_EnableIRQ(INTTX0_IRQn);
      }

      return ch;
}
```

7-7-2 例: UART FIFO

TX03 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。

以下の例が含まれます。
1. UART と FIFO の設定と初期化
2. FIFO を使用した UART 送受信制御

- **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 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 UART0 と UART1 に設定します。

```
void SIO_Configuration(TSB_SC_TypeDef * SCx)
{
	if (SCx == TSB_SC0) {
		TSB_PE->CR |= GPIO_BIT_0;
		TSB_PE->FR1 |= GPIO_BIT_0;
		TSB_PE->FR1 |= GPIO_BIT_1;
```

```
            TSB_PE->IE |= GPIO_BIT_1;
        } else if (SCx == TSB_SC1) {
            TSB_PA->CR |= GPIO_BIT_5;
            TSB_PA->FR1 |= GPIO_BIT_5;
            TSB_PA->FR1 |= GPIO_BIT_6;
            TSB_PA->IE |= 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(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);
```

UART ペリフェラルドライバを使い、UART のチャネル 0 と 1 を初期化します。

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

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

FIFO の初期化を行います。

```
    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);
```

上記設定を行い、その後、UART0 と 1 の割り込みを有効にします。

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

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

データフローの残りのプロセスは UART0 と 1 の送受信割り込みの ISR にて終了します。
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);
    }
}
```

UART1 の送信割り込みルーチン:

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

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

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);
    }
}
```

UART1 の受信割り込みルーチン:

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

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

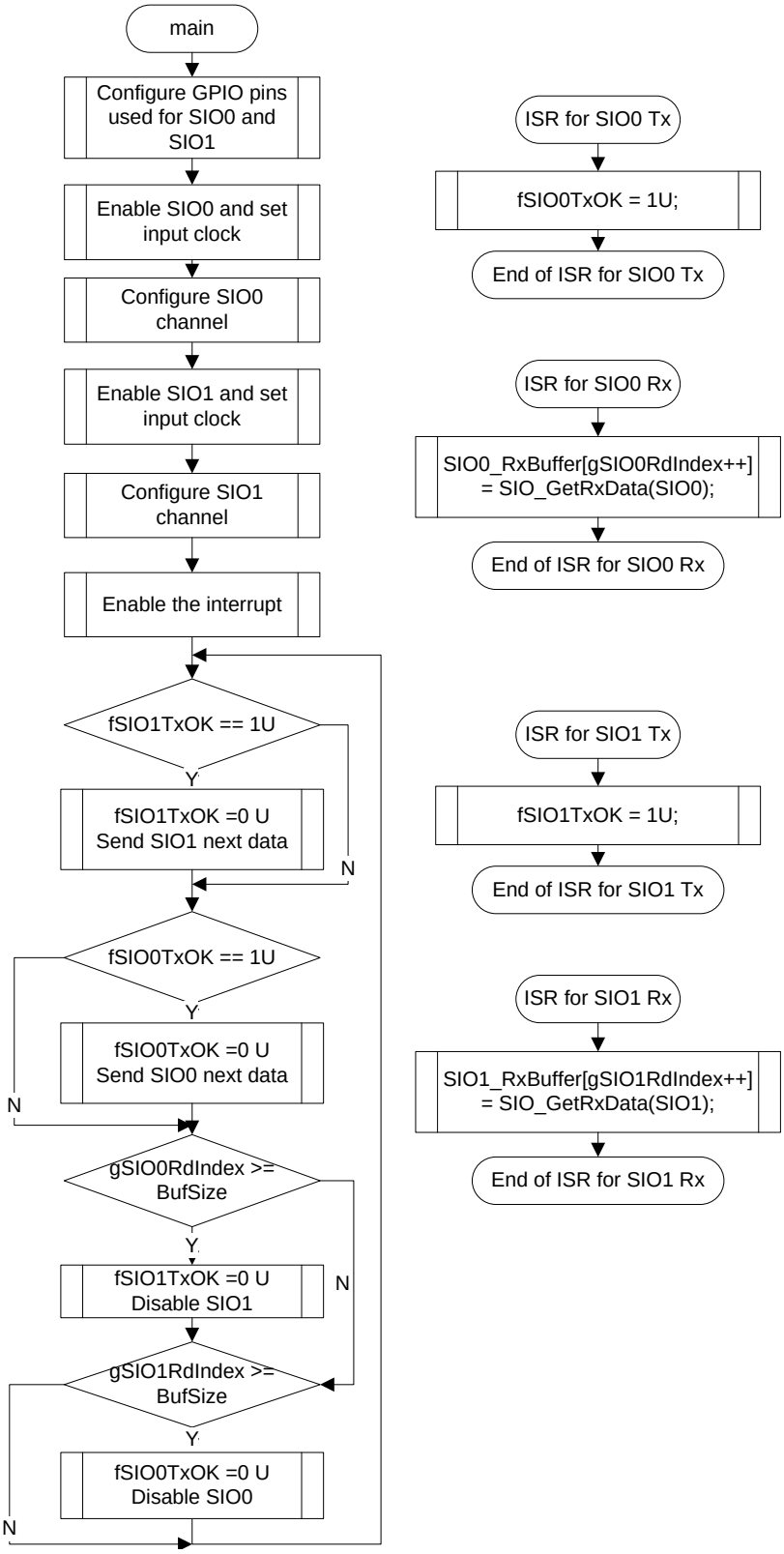
```
    }
}
```

7-7-3 例: SIO

TX03 ペリフェラルドライバ (SIO, GPIO)を用いたサンプルプログラムです。

以下の例が含まれます。
1. SIO の基本設定
2. SIO0⇔SIO1 間の通信制御
3. 送受信に SIO 割り込みを使用

- **フローチャート**


main
Configure GPIO pins used for SIO0 and SIO1
Enable SIO0 and set input clock
Configure SIO0 channel
Enable SIO1 and set input clock
Configure SIO1 channel
Enable the interrupt
fSIO1TxOK == 1U
Y
fSIO1TxOK =0 U
Send SIO1 next data
N
fSIO0TxOK == 1U
Y
fSIO0TxOK =0 U
Send SIO0 next data
N
gSIO0RdIndex >= BufSize
Y
fSIO1TxOK =0 U
Disable SIO1
N
gSIO1RdIndex >= BufSize
Y
fSIO0TxOK =0 U
Disable SIO0
N
ISR for SIO0 Tx
fSIO0TxOK = 1U;
End of ISR for SIO0 Tx
ISR for SIO0 Rx
SIO0_RxBuffer[gSIO0RdIndex++] = SIO_GetRxData(SIO0);
End of ISR for SIO0 Rx
ISR for SIO1 Tx
fSIO1TxOK = 1U;
End of ISR for SIO1 Tx
ISR for SIO1 Rx
SIO1_RxBuffer[gSIO1RdIndex++] = SIO_GetRxData(SIO1);
End of ISR for SIO1 Rx

- **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 SIO0 に設定します。その後、SIO0 の初期化構造体を準備し、入力クロックの初期化を行います。

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

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

SIO1 の初期化構造体を準備し、入力クロックの初期化を行います。

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

/*initialize the SIO1 struct */
SIO1_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
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);
```

上記設定を行い、その後、SIO の送信割り込みを有効にします。

```
/* 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);
```

すべての基本設定を行い、その後、データ送信を行います。

```
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 */
        }    }
```

SIO0 送信の ISR にて、転送設定を行います。

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

SIO0 受信の ISR にて、受信バッファからデータを受信します。

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

SIO1 送信の ISR にて、送信設定を行います。

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

SIO1 受信の ISR にて、受信バッファからデータを受信します。

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

7-8 VLTD

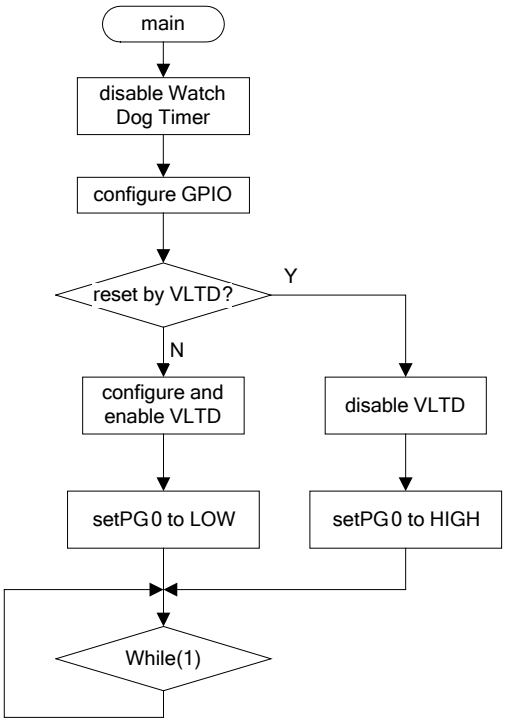
7-8-1 例: VLTD リセット

TX03 ペリフェラルドライバ(VLTD, CG, GPIO, WDT)を使用したサンプルプログラムです。

以下の例が含まれます:

1. VLTD 設定

- **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、ウォッチドッグタイマを禁止します。その後、PG0 を出力ポートに設定します。

```
WDT_Disable();
GPIO_SetOutput(GPIO_PG, GPIO_BIT_0);
```

その後、VLTD リセットフラグを読み出し、もし VLTD リセットされていなければ、VLTD の設定と有効化を行い、PG0 を LOW レベルにします。

```
if (CG_GetResetFlag().Bit.VLTDReset == 0U) {
        VLTD_SetVoltage(VLTD_DETECT_VOLTAGE_46);
        VLTD_Enable();
        GPIO_WriteData(GPIO_PG, 0x00);
```

VLTD リセットされている場合は、VLTD を無効にし、PG0 を HIGH レベルにします。

```
} else {
    VLTD_Disable();
    GPIO_WriteData(GPIO_PG, 0x01);
}
```

7-9 WDT

TX03 ペリフェラルドライバ (WDT, GPIO) のプログラムサンプルです。

以下の例が含まれます:

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 に設定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 の点滅を停止します。

```
WDT_Disable();
```

DEMO2 では、WDT クリアを行い、常に LED を点滅させます。

```
WDT_WriteClearCode();
```

7-10 ENC

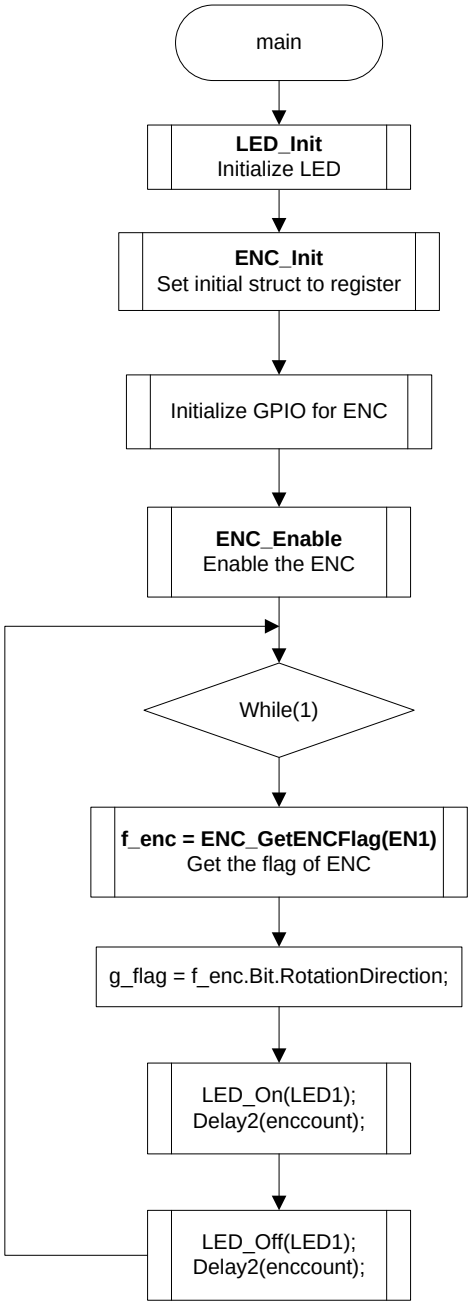
7-10-1 例: 回転検出

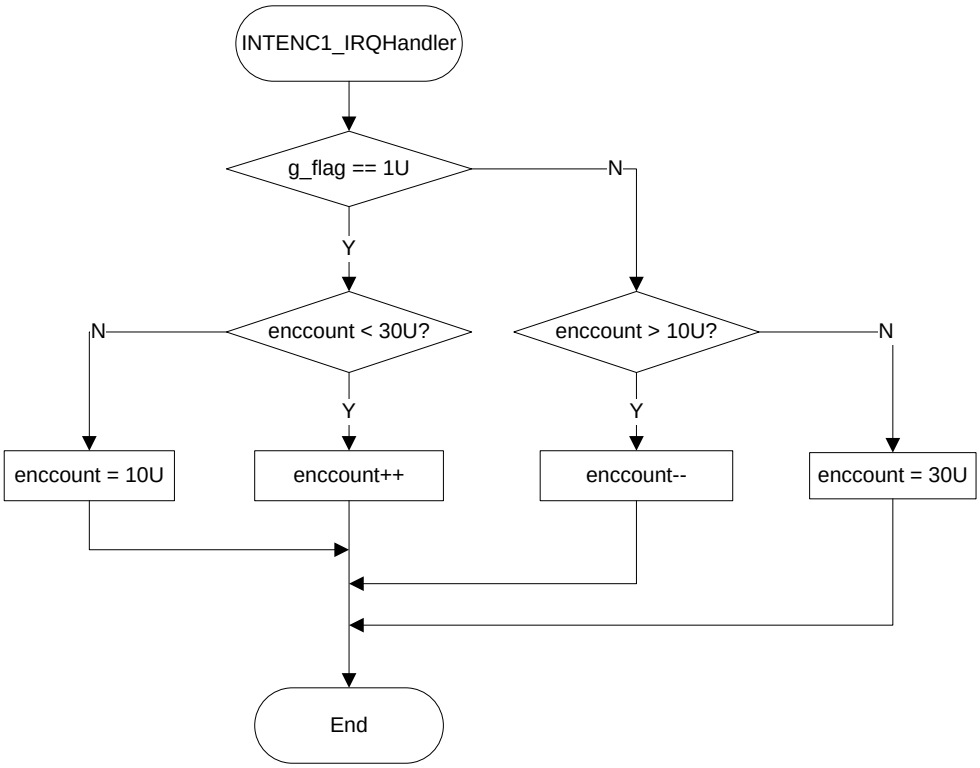
TX03 ペリフェラルドライバ(ENC, GPIO)を用いたサンプルプログラムです。

以下の例が含まれます:

1. ENC1 の初期化
2. ホイールマウスの回転検出

- **フローチャート**


main
LED_Init
Initialize LED
ENC_Init
Set initial struct to register
Initialize GPIO for ENC
ENC_Enable
Enable the ENC
While(1)
f_enc = ENC_GetENCFlag(EN1)
Get the flag of ENC
g_flag = f_enc.Bit.RotationDirection;
LED_On(LED1);
Delay2(enccount);
LED_Off(LED1);
Delay2(enccount);



- **サンプルプログラムのコードと説明**

TX03 ペリフェラルドライバ(ENC)を用いてホイールマウスの回転を検出するサンプルプログラムです。

まず、評価ボード上の LED を初期化します。

```
    /* LED initialization */
    LED_Init();
```

ENC 初期化構造体を用意し、ENC1 を初期化します。

```
    /* ENC1 initialization */
    m_enc.ModeType = ENC_ENCODER_MODE;
    m_enc.PhaseType = ENC_TWO_PHASE;
    m_enc.CompareStatus = ENC_COMPARE_DISABLE;
    m_enc.ZphaseStatus = ENC_ZPHASE_DISABLE;
    m_enc.FilterValue = ENC_FILTER_VALUE31;
    m_enc.IntEn = ENC_INTERRUPT_ENABLE;
    m_enc.PulseDivFactor = ENC_PULSE_DIV1;

    ENC_Init(EN1,&m_enc);
    ENC_SetCounterReload(EN1,0xFFFU);
```

GPIO を ENC1 に設定します。PF2 と PF3 は ENC1 入力に設定します。

```
    /* GPIO initialization */
    GPIO_SetInputEnableReg(GPIO_PF, GPIO_BIT_2, ENABLE);/*Set PF2 as
input */
    GPIO_EnableFuncReg(GPIO_PF, GPIO_FUNC_REG_1, GPIO_BIT_2);/*Set
PF2 as ENCA1 */
    GPIO_SetInputEnableReg(GPIO_PF, GPIO_BIT_3, ENABLE);/*Set PF3 as
```

```
input */
    GPIO_EnableFuncReg(GPIO_PF, GPIO_FUNC_REG_1, GPIO_BIT_3);/*Set
PF3 as ENCB1 */
```

ENC1 と ENC1 割り込みを許可します。

```
    /* Enable ENC1 */
    ENC_Enable(EN1);
    /* Enable ENC1 interrupt */
    NVIC_EnableIRQ(INTENC1_IRQn);
```

ホイールマウスの回転を検出するため ENC1 の回転方向検出状態を取得します。回転数に応じて LED1 を点滅します。

```
    /* LED1 blink speed based on the rolling of mouse wheel */
    while(1){
        f_enc = ENC_GetENCFlag(EN1);
        g_flag = f_enc.Bit.RotationDirection;
        LED_On(LED1);
        Delay2(enccount);
        LED_Off(LED1);
        Delay2(enccount);
    }
```

ENC カウントは、回転した数で、割り込みルーチンで変化します。
ホイールマウスが前方向に回転すると、ENC カウンタが 10 まで減ると ENC カウンタは 30 に戻り、ENC カウンタが 30 まで増えると ENC カウンタが 10 に戻ります。

```
    if(g_flag == 1U){
        if(enccount < 30U){
            enccount++;
        } else {
            enccount = 10U;
        }
    } else {
        if(enccount > 10U){
            enccount--;
        } else {
            enccount = 30U;
        }
    }
```

7-11 PMD

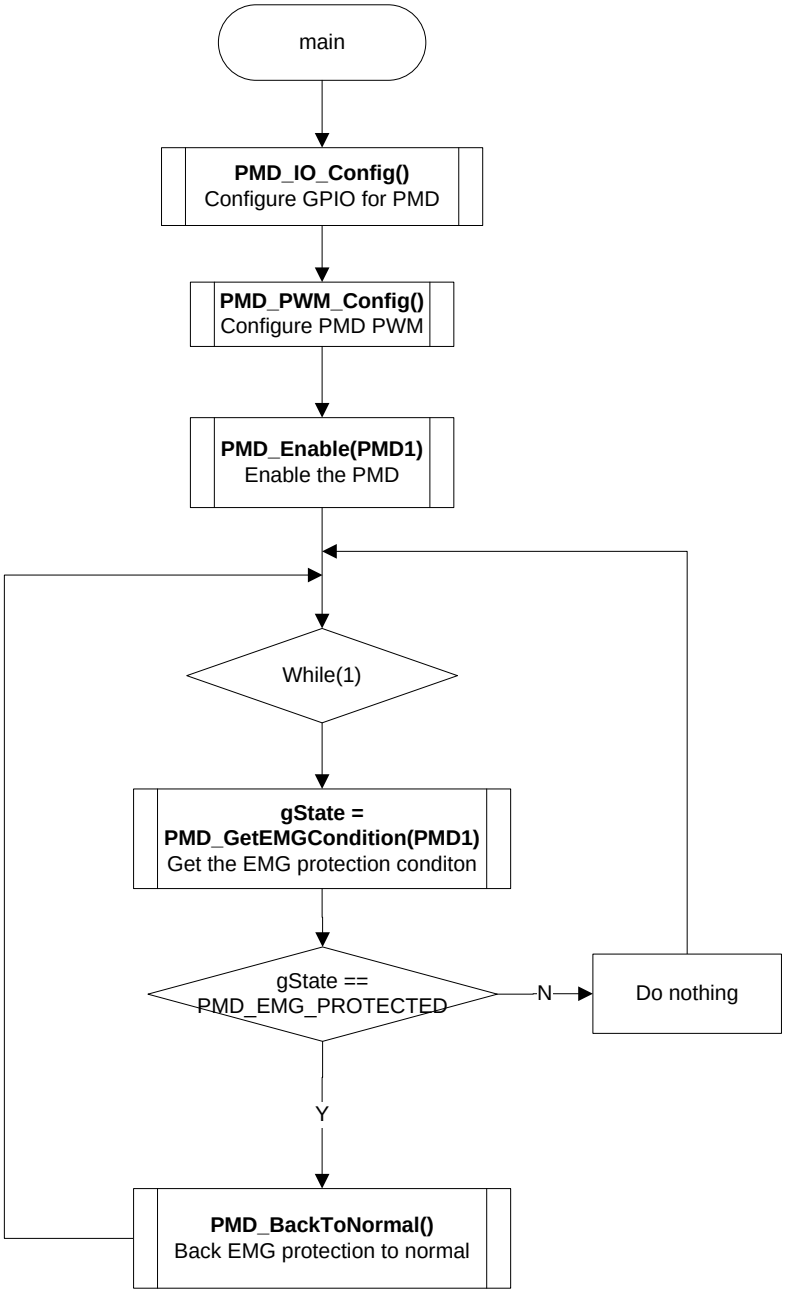
7-11-1 例: 位相出力

TX03 ペリフェラルドライバ (PMD, 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. PMD と GPIO の初期化
2. EMG 保護検出出力
3. EMG 保護検出出力から通常出力への状態変化

- **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 LED、PMD、GPIO の初期化を行います。

```
/* LED initialization */
LED4_Init();

/* GPIO configuration*/
PMD_IO_Config();

/* PMD PWM configuration*/
PMD_PWM_Config();
```

DEMO_U_PHASE 定義を行う場合、DUTY モードは U 相共通です。
DEMO_3_PHASE 定義を行う場合、DUTY モードは 3 相独立です。

```
/* PMD1 initialization */
    m_pmd.CycleMode = PMD_PWM_NORMAL_CYCLE;
#ifdef DEMO_U_PHASE
    m_pmd.DutyMode = PMD_DUTY_MODE_U_PHASE;          /* U-phase in
common */
#endif
#ifdef DEMO_3_PHASE
    m_pmd.DutyMode = PMD_DUTY_MODE_3_PHASE;          /* 3-phase
independent */
#endif
    m_pmd.IntTiming = PMD_PWM_INT_TIMING_MINIMUM;
    m_pmd.IntCycle = PMD_PWM_INT_CYCLE_1;
    m_pmd.CarrierMode = PMD_CARRIER_WAVE_MODE_1;     /* PWM mode 1
(center PWM, triangle wave) */
    m_pmd.CycleTiming = 0x3FFFU;

    PMD_Init(PMD1,&m_pmd);
```

3 相のコンペア値を設定します。
DUTY モードが U 相共通の場合、U 相と同じ出力です。
DUTY モードが 3 相独立の場合、Y/V/W それぞれの出力です。

```
    PMD_SetAllPhaseCompareValue(PMD1,0x0FFFU,0x1FFFU,0x2FFFU);
```

PMD 動作を許可します。

```
    PMD_Enable(PMD1);
```

EMG 保護検出の判定を行います。保護検出状態の場合は LED4 を点灯し、EMG 保護検出出力から通常出力へ戻します。保護検出状態ではない場合は LED4 を消灯します。

```
    while(1){
        /* Get the EMG protection condition*/
        gState = PMD_GetEMGCondition(PMD1);
        /* Judge the EMG protection condition */
        if (gState == EMG_PROTECTION) {
            /* LED4 will light on if the condition is protection */
            LED4_On();
            /* Back EMG protection to normal */
            PMD_BackToNormal();
            Delay(1000U);
        } else {
            /* LED4 will light off if the condition is not protection */
            LED4_Off();
        }    }
```

7-12 TRMOSC

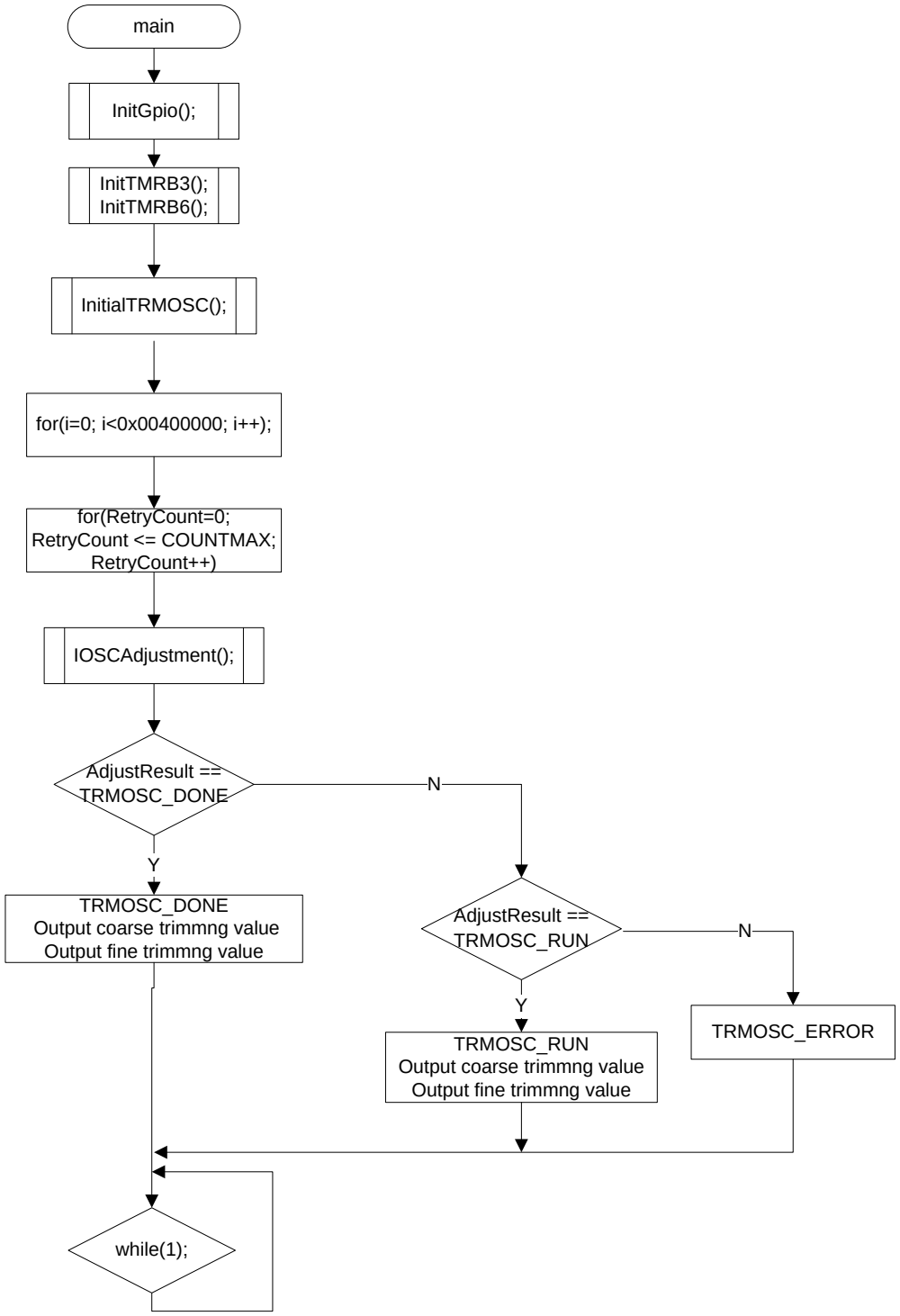
7-12-1 例: TRMOSC

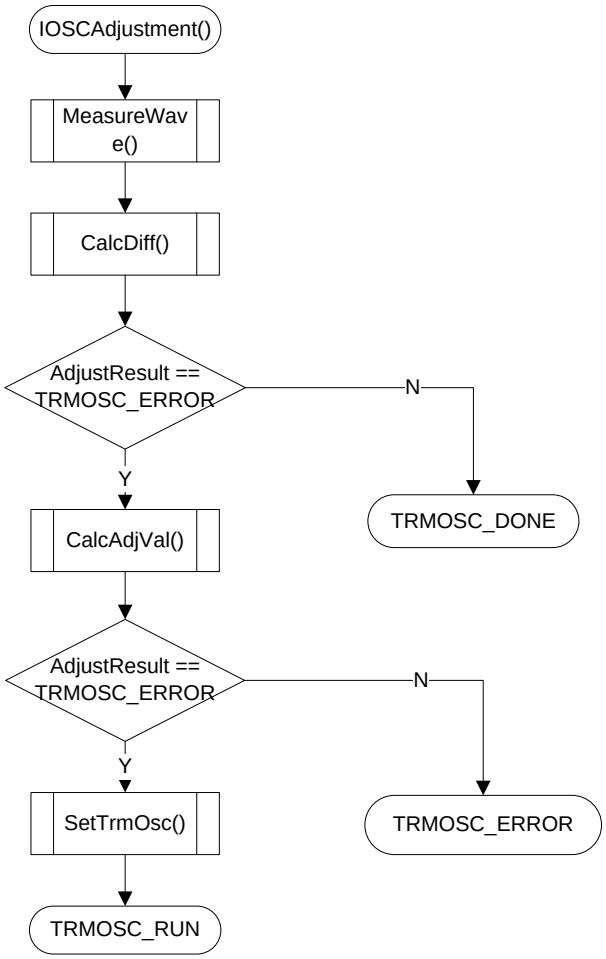
TX03 ペリフェラルドライバ(GPIO, CG, TMRB, TRMOSC)を使用したサンプルプログラムです。

以下の例が含まれます。

1. TMRB の初期化
2. 内蔵高速発振の調整

- **フローチャート**


main
InitGpio();
InitTMRB3();
InitTMRB6();
InitialTRMOSC();
for(i=0; i<0x00400000; i++);
for(RetryCount=0;
RetryCount <= COUNTMAX;
RetryCount++)
IOSCAdjustment();
AdjustResult ==
TRMOSC_DONE
N
Y
TRMOSC_DONE
Output coarse trimmng value
Output fine trimmng value
AdjustResult ==
TRMOSC_RUN
N
Y
TRMOSC_RUN
Output coarse trimmng value
Output fine trimmng value
TRMOSC_ERROR
while(1);



- **サンプルプログラムのコードと説明**

GPIO を TMRB に設定します。

```
/* TB6IN function */
GPIO_EnableFuncReg(GPIO_PA, GPIO_FUNC_REG_2, GPIO_BIT_6);
GPIO_SetInputEnableReg(GPIO_PA, GPIO_BIT_6, ENABLE);
/* TB3OUT function */
GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_7);
GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_1, ENABLE);
```

TMRB の初期化を行い、TMRB の動作を許可します。

```
/* TB6IN function */
TMRB_InitTypeDef m_tmrb;

m_tmrb.Mode = TMRB_INTERVAL_TIMER;
m_tmrb.ClkDiv = TMRB_CLK_DIV_2;
m_tmrb.TrailingTiming = TMRB6COUNT;
m_tmrb.UpCntCtrl = TMRB_FREE_RUN;
m_tmrb.LeadingTiming = TMRB6COUNT;

TMRB_Enable(TSB_TB6);
TMRB_Init(TSB_TB6, &m_tmrb);
TMRB_SetCaptureTiming(TSB_TB6, TMRB_CAPTURE_IN_RISING);
TMRB_SetExtStartTrg(TSB_TB6, ENABLE, TMRB_TRG_EDGE_FALLING);
```

```
    TMRB_SetRunState(TSB_TB6, TMRB_STOP);

    /* TB3OUT function */
    TMRB_InitTypeDef m_tmrb;
    TMRB_FFOutputTypeDef m_tmrb_ff;

    m_tmrb.Mode = TMRB_INTERVAL_TIMER;
    m_tmrb.ClkDiv = TMRB_CLK_DIV_2;
    m_tmrb.TrailingTiming = TMRB3COUNT;
    m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;
    m_tmrb.LeadingTiming = TMRB3COUNT;

    m_tmrb_ff.FlipflopCtrl = TMRB_FLIPFLOP_CLEAR;
    m_tmrb_ff.FlipflopReverseTrg =
    TMRB_FLIPFLOP_MATCH_LEADINGTIMING;

    TMRB_Enable(TSB_TB3);
    TMRB_Init(TSB_TB3, &m_tmrb);
    TMRB_SetFlipFlop(TSB_TB3, &m_tmrb_ff);
    TMRB_SetRunState(TSB_TB3, TMRB_RUN);
```

内蔵高速発振器を調整します。

```
    MeasureWave(&IoscAdjust);                /* Measurement Waveform */
    retval = CalcDiff(&IoscAdjust);          /* Calculation of difference */
    if(retval == TRMOSC_ERROR) {
        retval = CalcAdjVal(&IoscAdjust);    /* Calculation of adjustment value */
        if(retval == TRMOSC_RUN) {
            AdjustData.TrimmingControl       = TRMOSC_CONTROL_ENABLE;
            AdjustData.CoarseTrimmingValue = IoscAdjust.CoarseTrim;
            AdjustData.FineTrimmingValue     = IoscAdjust.FineTrim;
            SetTrmOsc(&AdjustData);          /* Set TRMOSC register */
        }
        else {                               /* TRMOSC_ERROR */
            /* do nothing */
        }
    }
    else {                                   /* TRMOSC_DONE   */
        /* do nothing */
    }
```