TOSHIBA

TX00 ペリフェラルドライバ使用例 (TMPM037)

第 2 版

2018 年 3 月 12 日

東芝デバイス＆ストレージ株式会社

CMDR-M037UE-01J

TOSHIBA

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。

TOSHIBA

目次

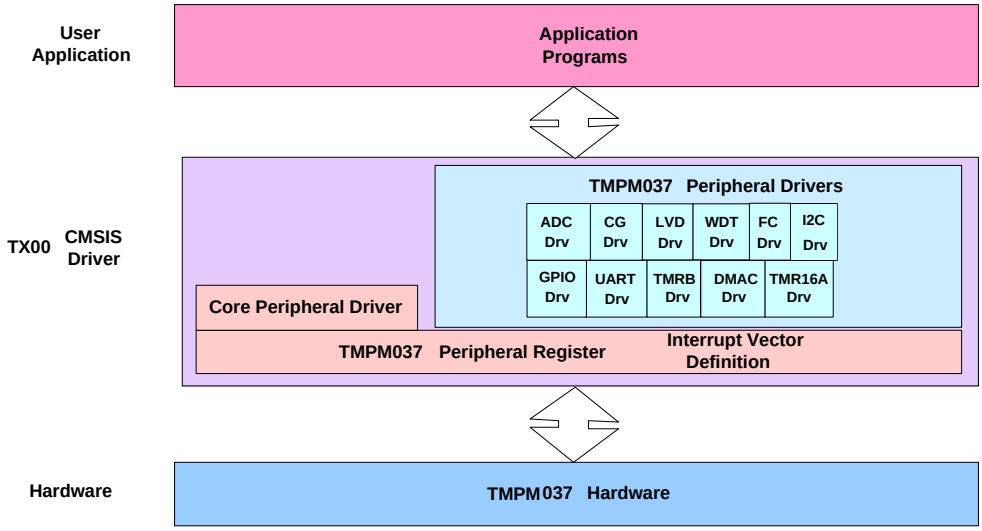
TOSHIBA

1 はしがき

本サンプルプログラムは、東芝マイコンTMPM037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

TX00 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
	PLL	PLL2 逓倍
スタンバイモード	-	使用 (STOP1 モード/IDLE モード)
SysTick	-	未使用
ウォッチドッグタイマ (WDT)	-	使用
外部割込み (INT)	INT0	使用(CG サンプルにおける STOP モードからの復帰)
	INT1	未使用
	INT2	未使用
	INT3	未使用
	INT4	未使用
	INT5	未使用

機能	チャネル	使用/未使用
SIO	SIO0	使用(SIO、UART FIFO)
	SIO1	使用(SIO)
	SIO2	使用(UART リターゲット)
	SIO3	使用(UART FIFO)
	SIO4	未使用
16-bit timerB	TMRB0	使用(汎用タイマ、PPG 出力)
	TMRB1	未使用
	TMRB2	未使用
	TMRB3	未使用
	TMRB4	未使用
	TMRB5	未使用
	TMRB6	未使用
	TMRB7	未使用
16-bit timerA	TMR16A0	使用(汎用タイマ)
	TMR16A1	未使用
LVD	-	使用(LVD サンプル)
10-bit A/D converter	AIN0	使用(ADC サンプル)
	AIN1	未使用
	AIN2	未使用
	AIN3	未使用
	AIN4	未使用
	AIN5	未使用
	AIN6	未使用
	AIN7	未使用
DMAC	-	使用(DMAC サンプル)
I2C	I2C0	使用(I2C 送受信)

4 端子用途

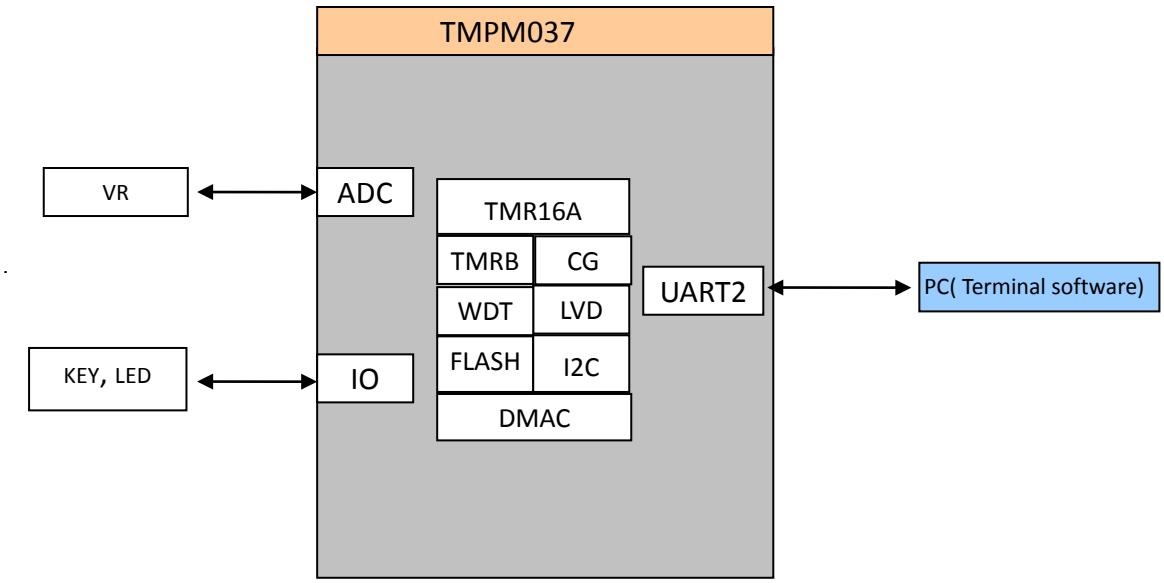
本サンプルプログラムは、開発環境に東芝製TMPM037用評価ボードを代用してテストされています。
以下に、端子用途を説明します。

No.	ポート名	機能
1	PD0	未使用
2	PD1	SC0SCLK
3	PD2	SC0RXD
4	PD3	SC0TXD
5	PD4	未使用
6	PD5	未使用
7	AVDD3	ADC 用電源
8	AVSS	ADC 用 GND
9	PA0	AIN0(ボード RV1)
10	PA1	未使用
11	PA2	未使用
12	PA3	未使用
13	PA4	未使用
14	PA5	未使用
15	PA6	未使用
16	PA7	未使用
17	PF7	SW3(ボード S5)
18	PF6	SW2(ボード S4)
19	PF5	SW1(ボード S3)
20	PF4	SW0(ボード S2)
21	PF3	SC3TXD
22	PF2	SC3RXD
23	PF1	未使用
24	PF0	未使用
25	PC5	LED3(ボード D2)
26	PC4	LED2(ボード D3)
27	PC3	TB0OUT/LED1(ボード D4)
28	PC2	LED0(ボード D5)
29	DVSS	デジタル用 GND
30	DVDD3	デジタル用電源
31	PC1	I2C0SDA
32	PC0	I2C0SCL
33	PG7	未使用
34	PG6	未使用
35	PG5	未使用
36	PG4	未使用
37	PG3	未使用
38	PG2	SC1TXD

39	PG1	SC1RXD
40	PG0	SC1SCLK
41	PE0	DipSW0
42	PE1	DipSW1
43	PE2	未使用
44	PE3	SC2RXD/USB_RXD(ボード CN8)
45	PE4	SC2TXD/USB_TXD(ボード CN8)
46	PE5	未使用
47	PE6	未使用
48	PE7	未使用
49	PB7	未使用
50	PB6	未使用
51	PB5	INT0
52	PB4	未使用
53	PB3	未使用
54	PB2	未使用
55	PB1	未使用
56	PB0	BOOT
57	DVDD3	デジタル用電源
58	X1	高速発振子接続
59	X2	高速発振子接続
60	DVSS	デジタル用 GND
61	REGOUT	レギュレータ用コンデンサ接続
62	RESET	リセット信号入力
63	MODE	MODE 端子 必ず GND に接続してください。
64	RVDD3	レギュレータ用電源.

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:
   TMPM037 用評価ボード

2. 開発ツール:
   - IAR:
     1) IDE:IAR Embedded workbench 7.70.1
   - KEIL
     1) IDE: KEIL uVision 5.24.2.0
   - ICE
     1) Evaluation Board に搭載の CMSIS-DAP

6 機能

6-1 動作モード

本デバイスは 3 つの動作モードがあります: NORMAL, IDLE, STOP1

- **NORMAL モード:**

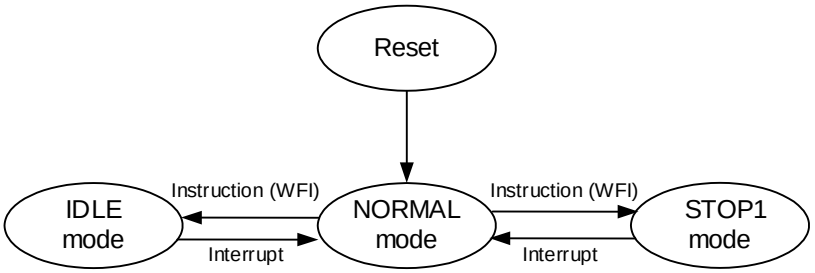
CPU コアおよび周辺ハードウェアを高速クロックで動作させるモードです。リセット解除後は、NORMAL モードになります。
IDLE, STOP1 の各モードは低消費電力モードです。
低消費電力モードへ移行するには、スタンバイコントロールレジスタ CGSTBYCR<STBY2:0> にて IDLE、STOP1 のいずれかのモードを選択し、WFI (Wait For Interrupt)命令を実行します。

- **STOP1 モード:**

すべての内蔵回路が停止するモードです。STOP1 モードが解除されると、STOP1 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。
補足: STOP1 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。

- **IDLE モード:**

CPU が停止するモードです。周辺機能の一部は、各モジュールの中のレジスタに IDLE モード時の動作/ 停止設定レジスタを 1 ビット持ち、IDLE モードでの動作設定が可能です。IDLE モード時に動作停止に設定された周辺機能は、IDLE モードへ遷移した時の状態で停止します。
補足: IDLE 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

PA0/AIN0 に接続されたポテンションメータの値を変更します。測定された電圧値は標準出力ライブラリ(stdout)を経由で出力します。stdout は UART にリターゲットされますので、PC と UART2 を接続してください。AD 変換結果は、ターミナル出力ソフトに表示されます。

6-3 CG

6-3-1 STOP1

CPU 動作モードを変更します。NORMAL モードと STOP1 モードの 2 つのモードを使用します。

現在のモード	アクション (Key)	動作	ターミナル表示	LED 表示
NORMAL	SW1を押す (Lowアクティブ)	NORMAL→ STOP1	“NORMAL MODE” →”STOP MODE”	LED0,1,2,3 オフ
STOP1	最初に SW1 を離して、SW0 を押す	STOP1 →NORMAL	“STOP MODE” →” NORMAL MODE”	LED0,1,2,3 オン

6-3-2 IDLE

CPU 動作モードを変更します。NORMAL モードと IDLE モードの 2 つのモードを使用します。

現在のモード	アクション (Key)	動作	ターミナル表示	LED 表示
NORMAL	SW1を押す (Lowアクティブ)	NORMAL→ IDLE	“NORMAL MODE” →” IDLE MODE”	LED0,1,2,3 オン
IDLE	SW1 を離す	IDLE →NORMAL	“IDLE MODE” →” NORMAL MODE”	LED0,1,2,3 オフ

6-4 DMAC

UART0 から UART1 へのデータ転送を行う処理が実装されています。“TOSHIBA”の文字列データを UART0 から UART1 へ送信されます。
DMAC により、RAM から UART0 データレジスタにデータが送信され、文字列の各文字データは UART0 経由で送信され、UART1 で受信します。UART1 のデータ受信後、データは文字配列に保存されます。

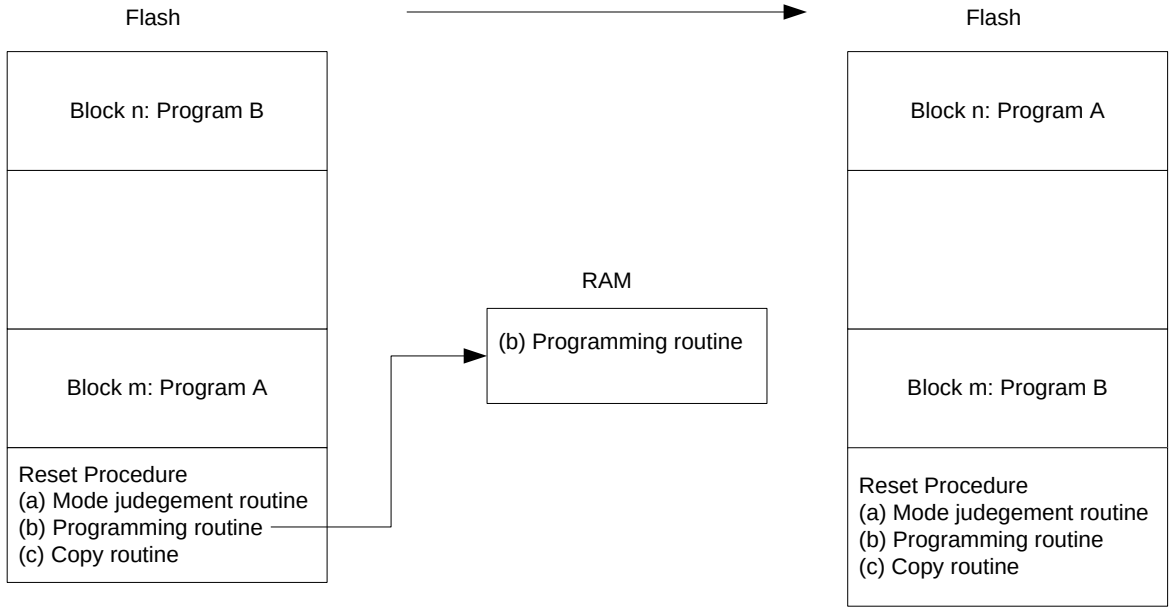
デバッガの変数ウィンドウに文字配列変数を登録して、受信されたデータを確認します。

補足: サンプルソフトウエアを使用するには、TMPM037 ボードの設定変更を行います。
UART0(TX)と UART1(RX)を接続します。

6-5 FLASH

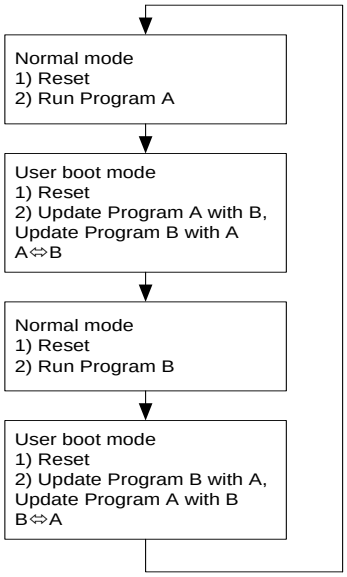
Flash のドライバ API を使用して内蔵フラッシュメモリの消去及び再書き込みを行うサンプルプログラムです。
このプログラムは、シングルチップモードのノーマルモードとユーザブートモードで実行します。
ーリセット動作: モード判定、書き込みルーチン(フラッシュ API)、転送ルーチンで構成されます。
・モード判定ルーチン: ユーザブートモードまたはノーマルモードの判定を行います。
・転送ルーチン: 書き込みルーチンを Flash から RAM へ転送します。
・書き込みルーチン: RAM 上で動作し、フラッシュ上のプログラム A とプログラム B のコードを入れ替えます。
ープログラム A/B (リセット動作)は事前にフラッシュメモリに書き込まれています。

－プログラム A/B (A: LED 0 点滅, B: LED2 点滅)
初期状態は、プログラム A が最初に動作します。
－SW0 キーはリセット動作のモード判定に使います。
SW0 キーが押されていない場合 → Normal モードです。
SW0 キーが押された場合 → ユーザブートモードです。

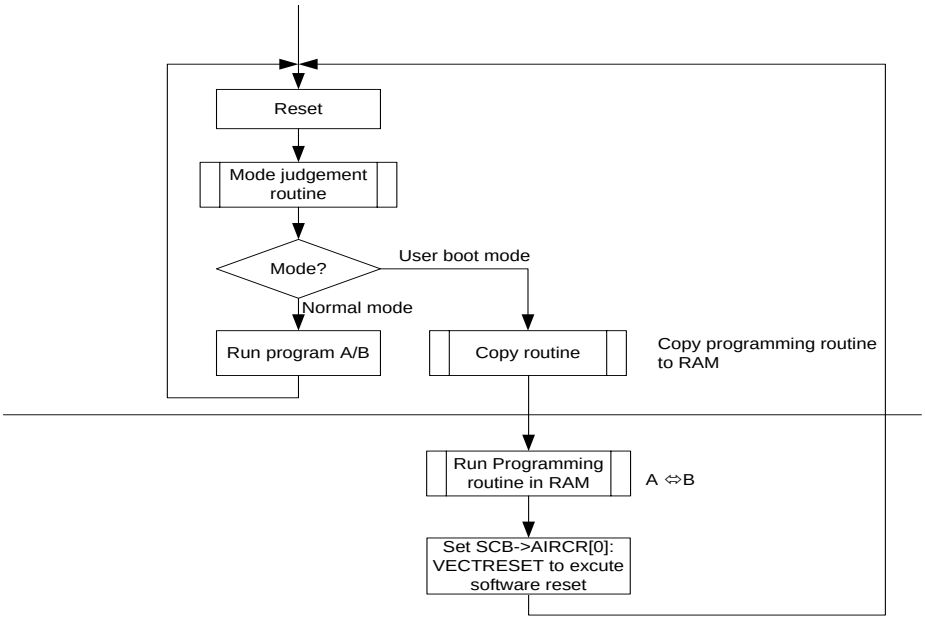


動作シーケンス

(1) 電源投入
まずプログラム A が実行され、LED0 が点滅します。
(2) SW0 を ON している間に RESET ボタンを ON します。その後、SW0 を OFF します。
コピー処理により、書き込み処理を RAM に転送します。
その後、書き込み処理がプログラム A とプログラム B をスワップします。
スワップが完了すると LED3 が点灯→LED1 が点灯します。
(3) スワップ完了後、システムリセットを行い、プログラム B が動作して LED1,3 が消灯→LED2 が点滅します。
(4) SW0 を ON している間に RESET ボタンを ON します。その後、SW0 を OFF します。
手順 (2) を行います。
(5) システムリセットを行い、プログラム A が動作して LED1,3 が消灯→LED0 が点滅します。



・サンプルプログラムのフローチャート



6-6 GPIO

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。LEDとSWの設定を行い、LEDの点滅を行います。

6-7 I2C

6-7-1 I2C スレーブ

ペリフェラルドライバの I2C を使用し、I2C バスのスレーブモード処理を行うサンプルプログラムです。
評価ボードの 2 本の I2C バスを接続します。
サンプルプログラムの動作確認では評価ボードを2つ使用し、片方は I2C バスのスレーブモード/もう片方は I2C バスのマスタモードで動作します。

I2C 割り込みを使用して I2C バスのリード/ライトを行います。
I2CAR で指定されたスレーブアドレス、またはジェネラルコールアドレスと一致したアドレスを受信すると I2C は 9 番目のクロックで SDA を"Low"にし、ACK 信号を出力します。

I2C スレーブは I2C マスタから “TOSHIBA” を受信し、これを UART に出力します。

I2C (スレーブ) サンプル開始

```
K1: I2C SLAVE RECV
```

SW1 が ON の場合、I2C(スレーブ)バッファから受信した文字列 “TOSHIBA” を出力します。

```
TOSHIBA
MASTER I2C to SLAVE I2C OK
```

6-7-2 I2C マスタ

ペリフェラルドライバの I2C を使用し、I2C バスのマスタモード処理を行うサンプルプログラムです。
評価ボードの 2 本の I2C バスを接続します。
サンプルプログラムの動作確認では評価ボードを2つ使用し、片方は I2C バスのマスタモード/もう片方は I2C バスのスレーブモードで動作します。

I2C 割り込みを使用して I2C バスのリード/ライトを行います。

I2C のアドレス: 指定なし
I2C マスタは文字列 “TOSHIBA” を I2C スレーブに送信します。

I2C(マスタ)サンプル開始
SW3 を ON すると、文字列 “TOSHIBA” を I2C(マスタ)から I2C(スレーブ)へ転送します。

6-8 LVD

この例では、LVD によって検出される電圧状態を取得するサンプルプログラムです。
電圧が検出電圧より低い場合は UART に“LOWER”を出力します。
電圧が検出電圧より高い場合は UART に“UPPER”を出力します。

6-9 TMR16A

この例は、MCU の簡易タイマを使って、汎用タイマを実現するサンプルプログラムです。
タイマ設定は 1ms 周期です。
このタイマを使い 1s (500ms で ON、500ms で OFF) 間隔で LED 点滅を行います。

6-10 TMRB

6-10-1 汎用タイマ

この例は、MCU のタイマを使って、汎用タイマを実現するサンプルプログラムです。
タイマ設定は 1ms 周期です。
このタイマを使い 1s (500ms で ON、500ms で OFF) 間隔で LED 点滅を行います。

6-10-2 PPG 波形出力

この例は、SW1 を使い、デューティ可変の PPG(プログラマブル矩形波)の波形を出力する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。
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 波形出力を開始します。
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%

補足: PPG 波形出力の確認のため、PC3 をオシロスコープに接続します。

6-11 SIO/UART

6-11-1 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。
この例は、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うサンプルプログラムです。

補足: サンプルプログラムでは UART2 を使用します。

6-11-2 UART FIFO

UART0 から"TMPM0371"というデータを FIFO を使用して UART3 へ送信し、同時に UART3 から"TMPM0372"というデータを FIFO を使用して UART0 へ送信するサンプルプログラムです。
ResetIdx() 関数の前にブレークポイントを設定しておくと、RxBuffer=”TMPM0372” と RxBuffer1="TMPM0371"となった場合にブレークポイントで停止します。

6-11-3 SIO

SIO 機能を使用して同期式の送受信を行うサンプルプログラムです。
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)

6-12 WDT

ウォッチドッグタイマは、高周波クロックが停止している(STOP1 モードの)場合、使用できません。
リセット後 WDT は有効となり、SystemInit()関数内で無効に設定されます。

ドライバの使い方:

1. 検出時間の設定とカウンタ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 (LED1 点灯→消灯)
   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。(LED0 は常に点滅)

7 ソフトウエア

本ソフトウエアは、TMPM037 MCU の主要機能を TMPM037 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
        │   tmpm037_wdt_int.c
        │   tmpm037_wdt_int.h
        │
        ├─TX00_CMSIS
        │ │ system_TMPM037.c
        │ │ system_TMPM037.h
        │ │ TMPM037.h
        │ │
        │ └─startup
              startup_TMPM037.s
        │
        ├─TX00_Periph_Driver
        │   ├─inc
        │   │   tmpm037_cg.h
        │   │   tmpm037_gpio.h
        │   │   tmpm037_wdt.h
        │   │   TX00_common.h
        │   │
        │   └─src
        │       tmpm037_cg.c
        │       tmpm037_gpio.c
        │       tmpm037_wdt.c
        │
        └─TMPM037-EVAL
                led.c
                led.h
                sw.c
                sw.h
```

KEIL MDK:

```
├─WDT_NMI
        ├─APP
        │   main.c
        │   tmpm037_wdt_int.c
        │
        ├─TX00_CMSIS
        │    system_TMPM037.c
        │    startup_TMPM037.s
        │
        ├─TX00_Periph_Driver
```

```
│    tmpm037_cg.c
│    tmpm037_gpio.c
│    tmpm037_wdt.c
│
└─TMPM037-EVAL
      led.c
      sw.c
```

7-1 ADC

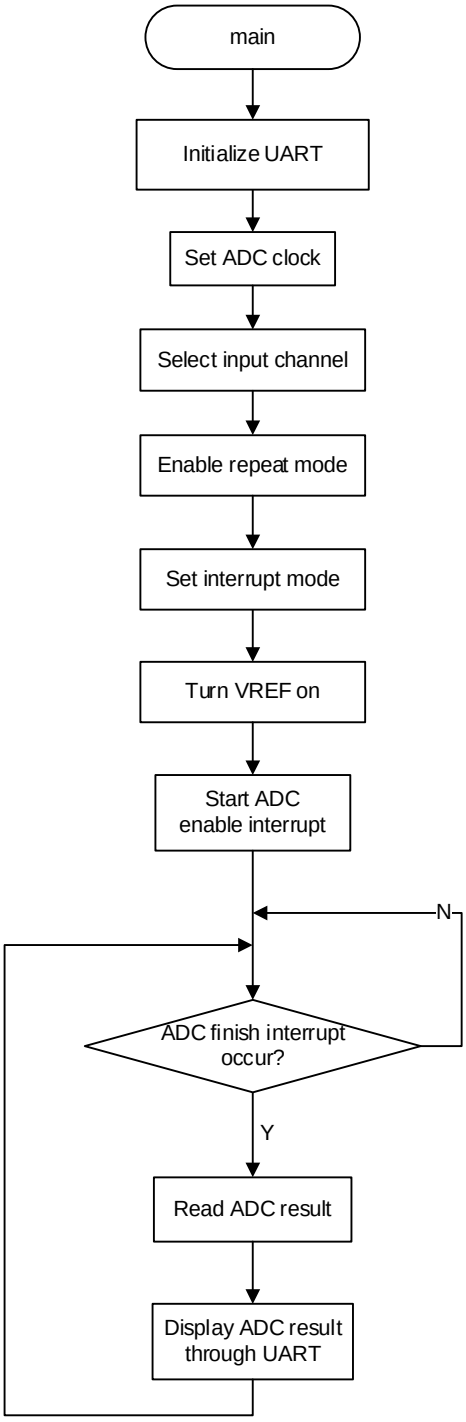
7-1-1 例: ADC データリード

ペリフェラルドライバ(ADC, GPIO, UART)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. チャネル固定リピートモードによる AD 変換を開始し、AD 変換結果を読み出す

- **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 ADC のクロック設定を行い、リピートモードの許可と割り込み要因の設定を行い、VREF を ON します。

```
    /* set ADC clock */
  ADC_SetClk(ADC_CONVERSION_42_CLOCK, ADC_FC_DIVIDE_LEVEL_16);

    /* select ADC input channel */
    ADC_SetInputChannel(ADC_AN_0);

    /* Enable ADC repeat mode */
```

```
    ADC_SetRepeatMode(ENABLE);

    /* Set interrupt mode */
    ADC_SetINTMode(ADC_INT_CONVERSION_8);

    /* Turn VREF on */
    ADC_SetVref(ENABLE);

    /* Wait at least 3us to ensure the voltage is stable */
    Delay(10U);
```

AD 変換を開始し、INTAD を許可します。

```
    ADC_Start();

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

AD 変換を行い、その後、INTAD を待ちます。INTAD の発生後、ADC_Display()をコールします。

```
    while (1) {
        if (fIntADC == 1U) {
            fIntADC = 0U;
            ADC_Display();
        }
    }
```

ADC_Display()では、AD 変換結果を確認するため、ADREG をリードします。

```
    ADC_ResultTypeDef ADC_Result = ADC_GetConvertResult(ADC_REG_0);
```

7-2 CG

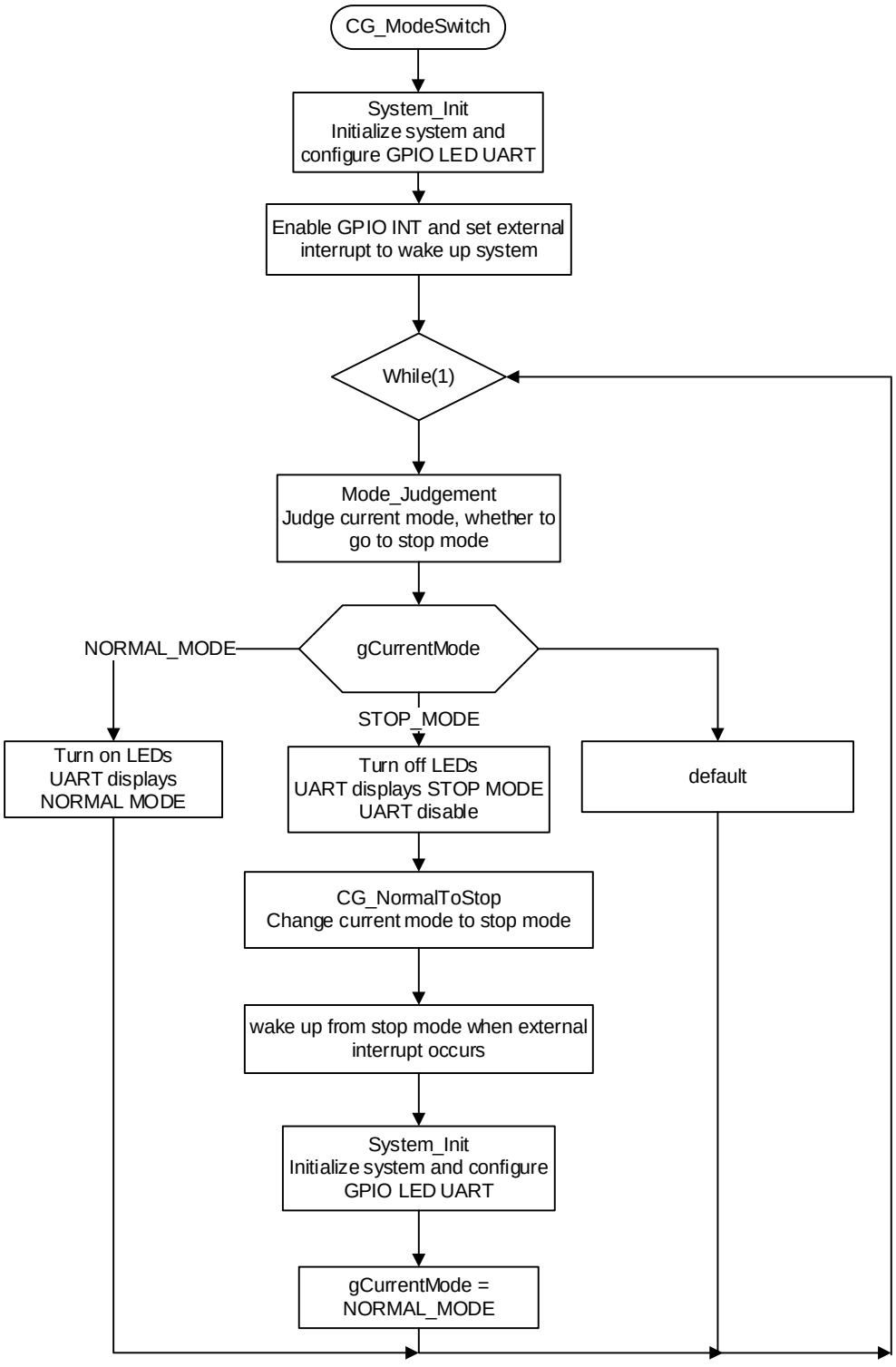
7-2-1 例: NORMAL<->STOP1 モード変更

ペリフェラルドライバ(CG, GPIO, UART)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 STOP1 モードの切り替え方法

- **フローチャート:**



- **サンプルプログラムのコードと説明**

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 20MHz の場合)

```
    if (CG_GetFoscSrc()==CG_FOSC_OSC_INT){
        /* Switch over from IHOSC to EHOSC*/
```

```
        switchFromIHOSCtoEHOSC();
    }
    /* Disable PLL. PLL cannot be used, since fosc = 20MHz that larger than 10 MHz. */
    CG_DisableClkMulCircuit();
    /* Set fgear = fc */
    CG_SetFgearLevel(CG_DIVIDE_1);
    /* Set fperiph to fgear */
    CG_SetPhiT0Src(CG_PHIT0_SRC_FGEAR);
    /* Set ΦT0 = fc */
    CG_SetPhiT0Level(CG_DIVIDE_1);
    /* Set low power consumption mode stop1 */
    CG_SetSTBYMode(CG_STBY_MODE_STOP1);
```

STOP1 モード解除割り込みの設定

STOP1 モードを解除する外部割り込み(INT0)の設定を行います。INT0 割り込みを許可し、割り込み要求をクリアします。

```
    __disable_irq();
    CG_ClearINTReq(CG_INT_SRC_0);
    /* Set external interrupt to wake up system */
    CG_SetSTBYReleaseINTSrc(CG_INT_SRC_0,
                                    CG_INT_ACTIVE_STATE_FALLING, ENABLE);
    NVIC_ClearPendingIRQ(INT0_IRQn);
    NVIC_EnableIRQ(INT0_IRQn);
    __enable_irq();
```

STOP1 モードへの遷移設定

STOP1 モードへの遷移設定を行います。ウォームアップ時間を設定し、__WFI()命令を実行して STOP1 モードへ遷移します。

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

マルチクロック回路の禁止

Fosc が 10MHz を超えているため、PLL を無効にします。

```
  Result retval = ERROR;
  WorkState st = BUSY;
  retval = CG_SetFcSrc(CG_FC_SRC_FOSC);
  if (retval == SUCCESS) {
        /* Set warm up time */
        CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC_EXT_HIGH,
                                                    CG_WUODR_FOSC);
        CG_StartWarmUp();
        do {
            st = CG_GetWarmUpState();
        } while (st != DONE);
        /* fc source is set to fosc, so PLL can be disabled successfully. */
        retval = CG_SetPLL(DISABLE);
  } else {
        /*Do nothing */
  }
```

7-2-2 例: NORMAL <-> IDLE モード変更

ペリフェラルドライバ(CG, GPIO, UART)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 IDLE モードの切り替え方法

- **サンプルプログラムのコードと説明**

スイッチ、LED、UART 用端子の設定を行います。

```
TMRB_InitTypeDef m_tmrb;

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

割り込みタイミングの設定を行います。

```
m_tmrb.Mode = TMRB_INTERVAL_TIMER;
m_tmrb.ClkDiv = TMRB_CLK_DIV_512;
m_tmrb.TrailingTiming = TMRB_1MS;
m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;
m_tmrb.LeadingTiming = TMRB_1MS;
```

TMRB を用いて、低消費電力モード解除用割り込みの設定を行います。

```
TMBR_Enable(TSB_TB0);
TMRB_Init(TSB_TB0,&m_tmrb);
TMRB_SetINTMask(TSB_TB0,
                    TMRB_MASK_MATCH_LEADINGTIMING_INT);
TMRB_SetIdleMode(TSB_TB0,ENABLE);

NVIC_ClearPendingIRQ(ExtINTSrc_IRQn);
NVIC_EnableIRQ(INTTMRB0_IRQn);
```

SW1 を OFF の場合 LED を消灯します。SW1 が ON の場合 LED を点灯。アップカウンタを停止&クリアし、カウントを開始後 IDLE モードへ移行します。

```
while (1U) {
    if (SW_Get(SW1) == 1U) {            /* SW1 is ON */
        LED_On(LED_ALL);
        common_uart_disp("IDLE MODE¥r¥n");
        Delay();        /* Delay sometime to let UART print string. */
        TMRB_SetRunState(TSB_TB0, TMRB_STOP);
        TMRB_SetRunState(TSB_TB0, TMRB_RUN);
        enter_IDLE();
        LED_Off(LED_ALL);
        Common_uart_disp(“NORMAL MODE¥r¥n”);

    } else {
        LED_Off(LED_ALL);
        common_uart_disp("NORMAL MODE¥r¥n");
    }
}
```

IDLE モードに移行する準備を行います。
まず低消費電力モードとして IDLE を選択します。

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

最後に__WFI()命令を使用して IDLE モードに入ります。

```
__WFI();
```

7-3 DMAC

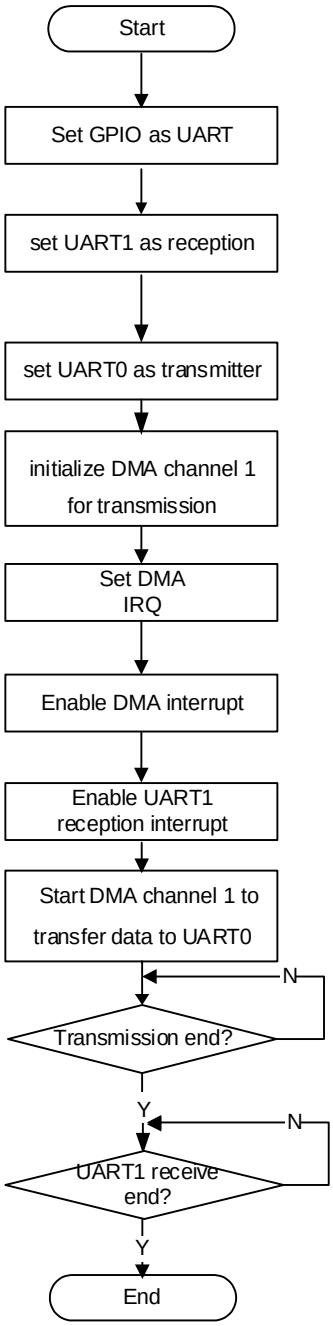
7-3-1 例: メモリから周辺回路

ペリフェラルドライバ (DMAC, GPIO, SIO) 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. UART0/1 の初期化と DMAC の初期化
2. メモリから DMAC 経由で UART0 へデータを転送

- **フローチャート:**



- **サンプルプログラムのコードと説明**

以下は DMAC のドライバを使用して、メモリから UART0 へデータ転送を行うサンプルプログラムです。

まず、UART1 の受信設定とデータ転送のために UART0 の設定と許可を行います。

```
    UART_Rx_Cnt = 0U;
    UART_InitStruct.BaudRate = 115200U;
    UART_InitStruct.DataBits = UART_DATA_BITS_8;
    UART_InitStruct.StopBits = UART_STOP_BITS_1;
    UART_InitStruct.Parity = UART_NO_PARITY;
    UART_InitStruct.Mode = UART_ENABLE_RX;
    UART_InitStruct.FlowCtrl = UART_NONE_FLOW_CTRL;
    UART_Enable(UART1);
    UART_Init(UART1, &UART_InitStruct);
```

```
UART_InitStruct.Mode = UART_ENABLE_TX;
UART_Enable(UART0);
UART_Init(UART0, &UART_InitStruct);_
UART_SetTxDMAReq(UART0, ENABLE);
```

データ転送のために DMAC のチャネル 1 の初期化と設定を行います。

```
DMAC_InitStruct.SrcAddr = (uint32_t) SRC_Buffer;
DMAC_InitStruct.SrcIncrementState = ENABLE;
DMAC_InitStruct.SrcBitWidth = DMAC_BYTE;
DMAC_InitStruct.SrcBurstSize = DMAC_1_BEAT;
DMAC_InitStruct.DstAddr = (uint32_t) (UART0_BASE_ADDR +
  UART0_BUF_OFFSET);
DMAC_InitStruct.DstIncrementState = DISABLE;
DMAC_InitStruct.DstBitWidth = DMAC_BYTE;
DMAC_InitStruct.DstBurstSize = DMAC_1_BEAT;
DMAC_InitStruct.TxSize = size;
DMAC_InitStruct.TxDirection = DMAC_MEMORY_TO_PERIPH;
DMAC_InitStruct.TxDstPeriph = DMAC_SIO0_UART0_TX;
DMAC_InitStruct.TxINT = ENABLE;

DMAC_Init(myChannel, &DMAC_InitStruct);
```

DMAC の割り込みを許可します。

```
NVIC_EnableIRQ(INTDMAC_IRQn);
```

DMAC の許可、送信終了割り込みの設定、UART0 へのバースト転送設定を行い、DMAC のチャネル 1 から転送を開始します。

```
DMAC_Enable();
DMAC_SetTxINTConfig(myChannel, DMAC_INT_TX_END, ENABLE);
DMAC_SetSWBurstReq(DMAC_SIO0_UART0_TX);
DMAC_SetDMAChannel(myChannel, ENABLE);
```

DMAC 転送が終わるまで待ちます。

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

DMAC 転送終了時に ISR の DMAC にフラグがセットされます。

```
state = DMAC_GetINTReq();
if (state.Bit.CH1_INTReq == 1U) {
    tmp = DMAC_GetTxINTReq(DMAC_CHANNEL_1);
    if (tmp == DMAC_TX_END_REQ) {
        TxEndFlag  =  DMAC_GetChannelTxState(DMAC_CHANNEL_1);
        DMAC_ClearTxINTReq(DMAC_CHANNEL_1,  DMAC_INT_TX_END);
    } else {
        /* Do nothing */
    }
} else {
    /* Do nothing */
}
```

7-4 FLASH

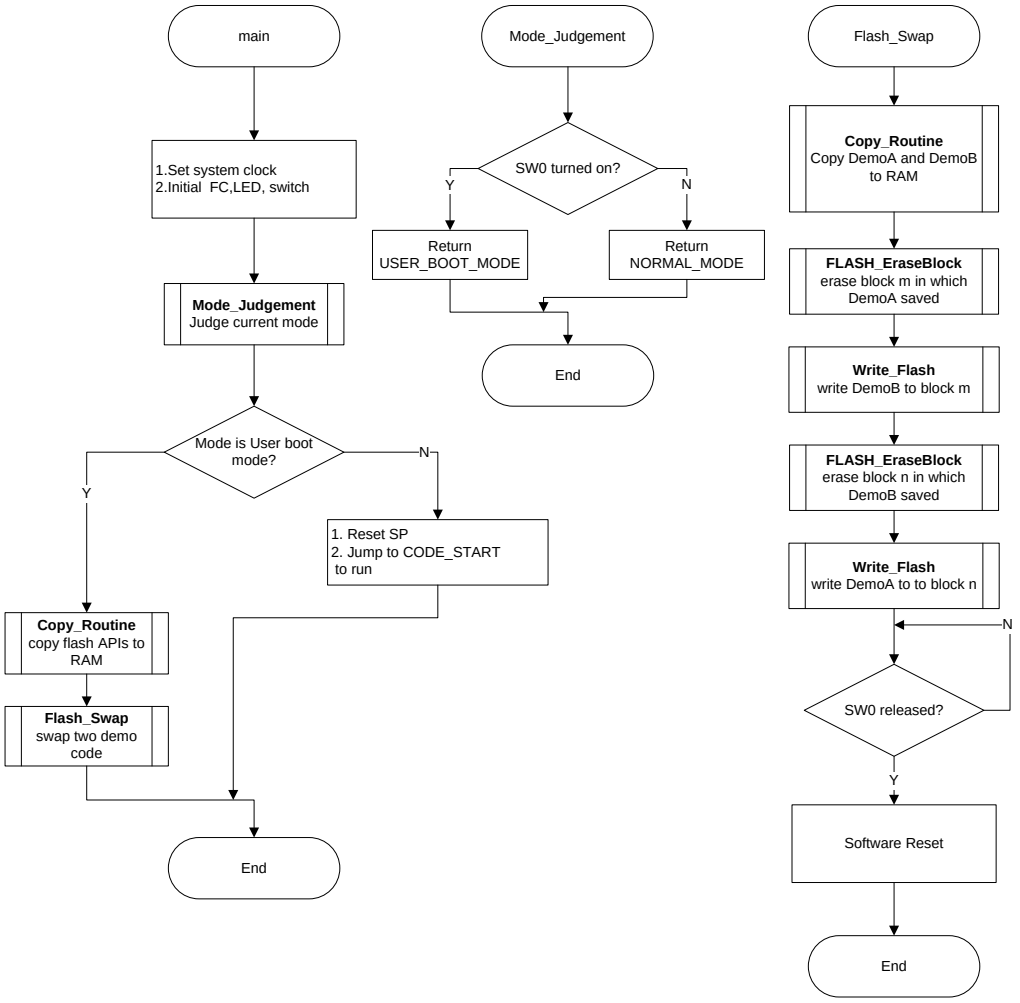
7-4-1 例: ユーザブート

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 (ノーマルモード、ユーザブートモード)
3. ユーザブートモードを使用し、Flash メモリのコードを更新

- **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 SW を初期化します。リセット時に現在のモード判定を SW で、現在の状態表示を LED で行います。

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

リセット後にどのモードになっているかの判断を行うために、Mode_Judgement()関数をコールします。

```
uint8_t Mode_Judgement(void)
{
    return (SW_Get(SW0) == 1U) ? USER_BOOT_MODE : NORMAL_MODE;
}
```

リセット時に SW0 が離されている場合、ノーマルモードになります。SP をリセットし、“CODE_START” にジャンプします。プログラム A はブロックm内の“CODE_START” に保存されているため、プログラム A が動作します(LED0 が点滅)。

```
#if defined ( __CC_ARM   )          /* RealView Compiler */
    ResetSP();                      /* reset SP */
#elif defined ( __ICCARM__ )        /* IAR Compiler */
    c_stack = 0;
    __set_MSP(*c_stack);    /* reset SP */
#endif
    startup = CODE_START;
    startup();                      /* jump to code start address to run */
```

リセット時に SW0 が押されている場合、ユーザブートモードになります。Flash メモリは自分自身で消去/書き込みを実行できないため、Flash 動作用 API を、Flash メモリ内の アドレス “FLASH_API_ROM” から RAM の “FLASH_API_RAM”にコピーします。

```
    Copy_Routine(FLASH_API_RAM, FLASH_API_ROM, SIZE_FLASH_API);
    /* copy flash API to RAM */
```

Flash 動作用 API を RAM にコピー後、ルーチンプログラムは Flash_Swap()関数にジャンプします。この関数は、上記の Copy_Routine() を使用し、Flash メモリから RAM にコピーされます。

Flash_Swap() 関数では、まずサンプル A、サンプル B を Flash メモリから RAM にコピーします。次に、Flash メモリの消去/書き込みを行うため、関数 FC_EraseBlock () と Write_Flash() を呼び出します。Flash メモリ内のプログラム A とプログラム B がスワップします(LED3 が点灯→LED1 が点灯)。

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

```
    if (FC_SUCCESS == Write_Flash(DEMO_A_FLASH, DEMO_B_RAM,
SIZE_DEMO_B)) {        /* write B to A */
        /* Do nothing */
    } else {
        return ERROR;
    }
    if (FC_SUCCESS == FC_EraseBlock((uint32_t) DEMO_B_FLASH)) { /* erase B */
        /* Do nothing */
    } else {
        return ERROR;
    }

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

SW0 が離されると、NVIC_SystemReset()を用いてソフトリセットを行います。その後、ノーマルモードになります。サンプル A とサンプル B が差し替えられているため、アドレス “CODE_START” はサンプル B のスタートアドレスになり、サンプル B が動作します (LED1,3 が消灯→LED2 が点滅)。

```
    while (SW_Get(SW0) == 1U) {
    }
    /* software reset */
    NVIC_SystemReset();
```

Flash メモリ動作関数 FC_EraseBlock() は指定されたブロックを消去します。このブロックは最初に引数 “block_addr” で指定します。まず、この関数で引数“block_addr” を確認します。次に、Flash ドライバ FC_GetBlockProtectState() を使用し、指定されたブロックがプロテクトされているか確認します。

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

ブロックにプロテクトがかかっている場合、“FC_ERROR_PROTECTED” を返します。プロテクトされていない場合は、ブロック消去コマンドにて、ブロックを消去します。

```
        *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、消去が完了すると、Flash ドライバ FC_GetBusyState() を用いビジーチェックを行います。同時に、タイムアウトカウンタを使用し、動作が指定時間を越えていないか確認します。

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

関数 Write_Flash() は FLASH_WritePage() を呼び出し、1 ページにデータを書き込みます。この動作は、自動ページプログラム命令を除き、基本的に FLASH_EraseBlock() と同じです。

```
        *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 < PROGRAM_UNIT; i++) {
            *PA = *source;
            source++;
        }
```

7-5 GPIO

7-5-1 例: 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。

以下の例が含まれます:

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() 関数を用いて GPIO を LED に設定します。また GPIO_SetInput()関数を用いて GPIO を Key に設定します。

```
    GPIO_SetOutput(GPIO_PC, GPIO_BIT_2 | GPIO_BIT_3 | GPIO_BIT_4 |
GPIO_BIT_5);

    GPIO_SetInput(GPIO_PF, GPIO_BIT_4 | GPIO_BIT_5 | GPIO_BIT_6 |
GPIO_BIT_7);
```

以下は for(; ;)ループにて LED の On/Off を繰り返します。

GPIO_ReadDataBit()関数を用いて、Key の状態を取得します。

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

GPIO_WriteData()関数を用いて LED を On します。

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

GPIO_WriteData()関数を用いて LED を Off します。

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

7-6 I2C

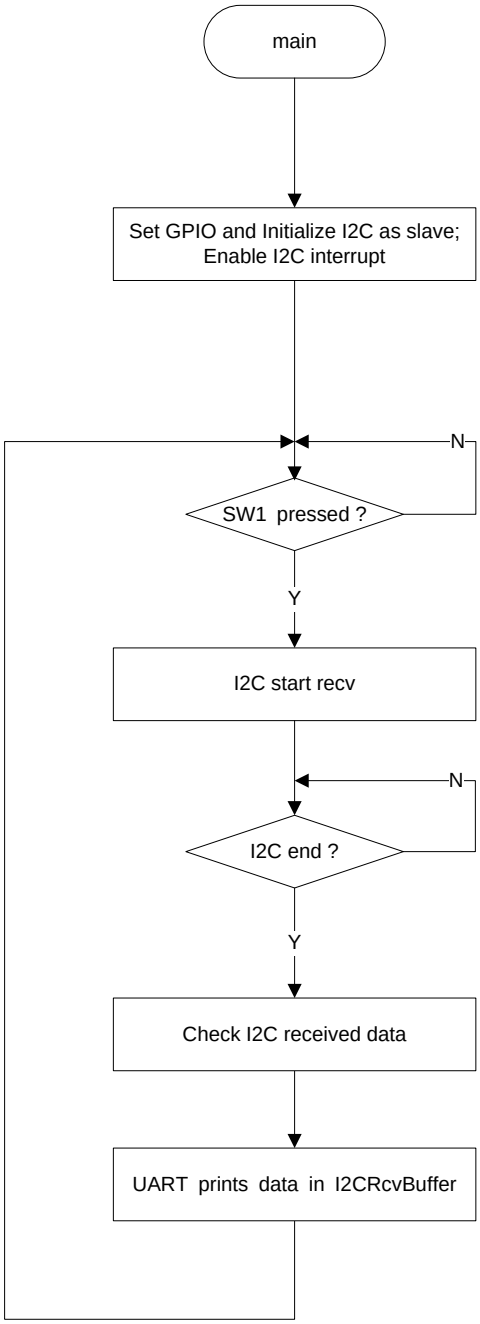
7-6-1 例: I2C スレーブ

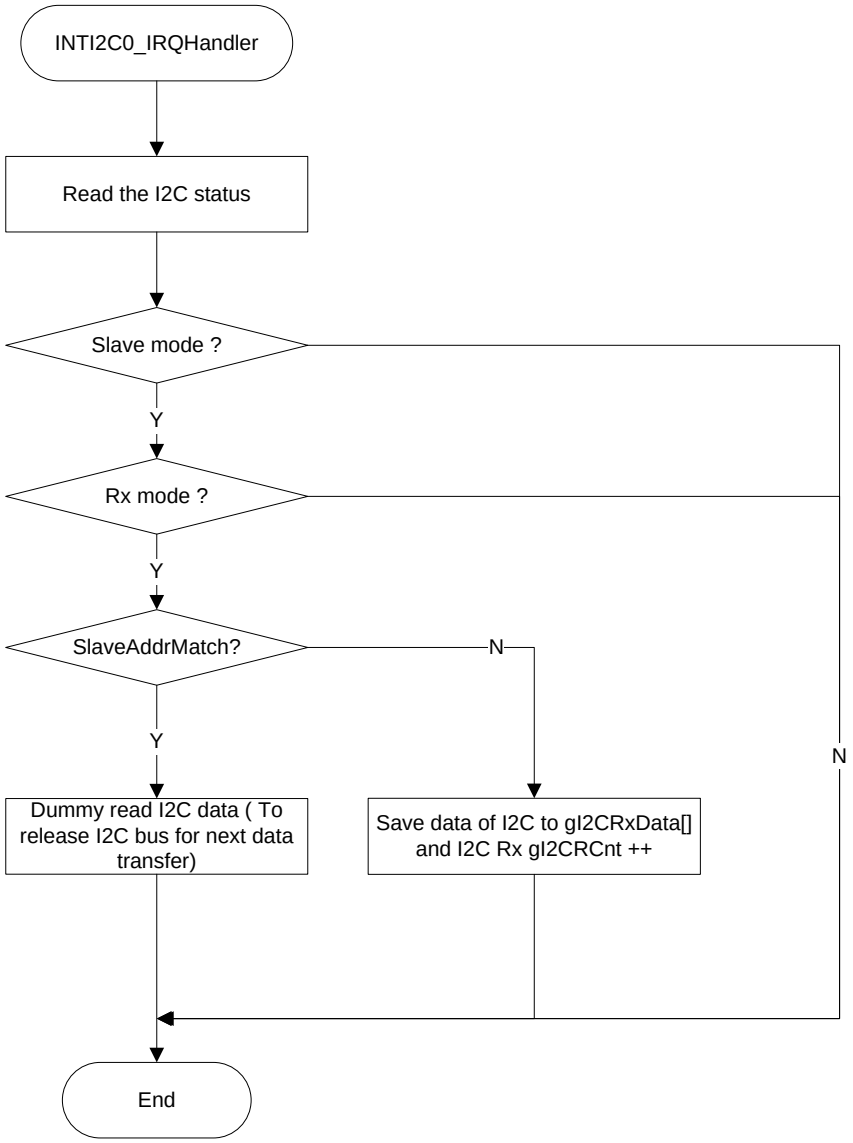
ペリフェラルドライバ(I2C, GPIO)を使用したサンプルプログラムです。

以下の例が含まれます:

1. I2C 初期化
2. I2C スレーブによるデータ受信

- **フローチャート:**


main
Set GPIO and Initialize I2C as slave;
Enable I2C interrupt
N
SW1 pressed ?
Y
I2C start recv
N
I2C end ?
Y
Check I2C received data
UART prints data in I2CRcvBuffer



- **サンプルプログラムのコードと説明**

まず、GPIO を I2C に設定します。

```
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_1, GPIO_BIT_0);
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_1, GPIO_BIT_1);
GPIO_SetOutputEnableReg(GPIO_PC, GPIO_BIT_0, ENABLE);
GPIO_SetOutputEnableReg(GPIO_PC, GPIO_BIT_1, ENABLE);
GPIO_SetInputEnableReg(GPIO_PC, GPIO_BIT_0, ENABLE);
GPIO_SetInputEnableReg(GPIO_PC, GPIO_BIT_1, ENABLE);
GPIO_SetOpenDrain(GPIO_PC, GPIO_BIT_0, ENABLE);
GPIO_SetOpenDrain(GPIO_PC, GPIO_BIT_1, ENABLE);
```

次に、I2C をイネーブルにし、その後 INTI2C0 をイネーブルにします。

```
myI2C.I2CSelfAddr = SLAVE_ADDR;
myI2C.I2CDataLen = I2C_DATA_LEN_8;
myI2C.I2CACKState = ENABLE;
```

```
    myI2C.I2CClkDiv = I2C_CLK_DIV_32;
    myI2C.PrescalerClkDiv = I2C_PRESCALER_DIV_12;
    I2C_SWReset();
    I2C_Init(&myI2C);
    NVIC_EnableIRQ(INTI2C0_IRQn);
    I2C_SetINTReq(ENABLE);
```

上記設定を行った後、I2C 受信を開始します。
I2C RX バッファを設定します。

```
    case MODE_I2C_INITIAL:
        for (gICnt = 0U; gICnt < 8U; gICnt++) {
            gI2CRxData[gICnt] = 0U;
        }
        gI2CMode = MODE_I2C_RCV;
        break;
```

データ受信は INTI2C ハンドラ内で行います。
INTI2C0 ハンドラ内で、I2C バス状態を取得し、その値で I2C スレーブ受信プロセスを行います。I2C_GetReceiveData()関数を用いて I2C バッファから受信データをリードします。I2C 動作はマスタによって制御されます。

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

    i2c0_sr = I2C_GetState();
    if (!i2c0_sr.Bit.MasterSlave) {        /* Slave mode */
        if (!i2c0_sr.Bit.TRx) { /* Rx Mode */
            if (i2c0_sr.Bit.SlaveAddrMatch) {
                /* First read is dummy read for Slave address recognize */
                tmp = I2C_GetReceiveData();
                gI2CRCnt = 0U;
                tmp = 0U;
                I2C_SetSendData(tmp);
            } else {
                /* Read I2C received data and save to I2C_RxData buffer */
                tmp = I2C_GetReceiveData();
                i2c0_sr = I2C_GetState();
                gI2CRxData[gI2CRCnt] = tmp;
                gI2CRCnt++;
                tmp = 0U;
                I2C_SetSendData(tmp);
            }
        } else {                        /* Tx Mode */
```

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

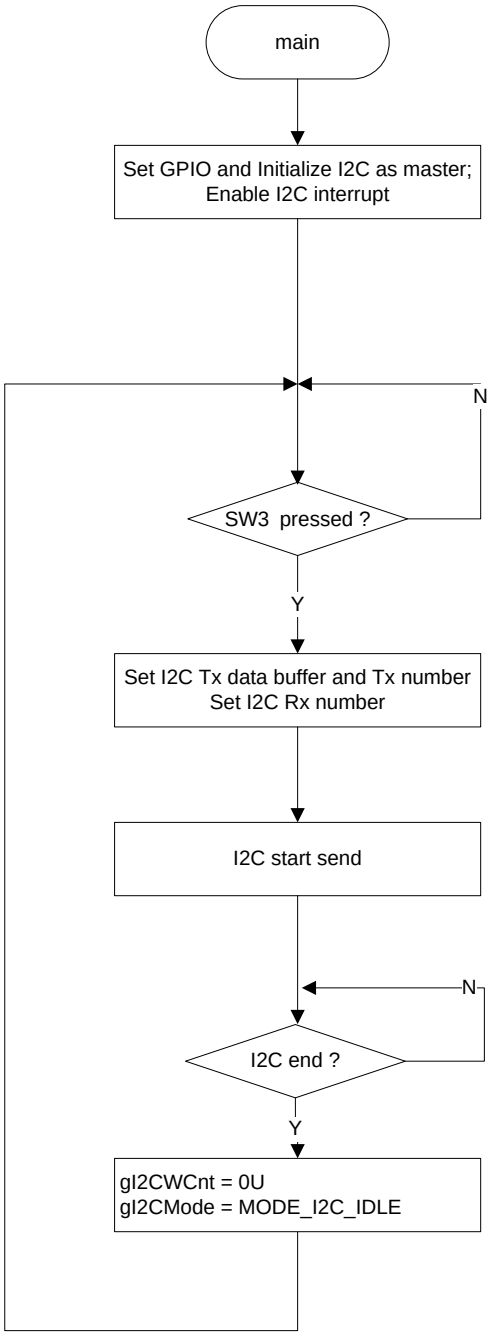
7-6-2 例: I2C マスタ

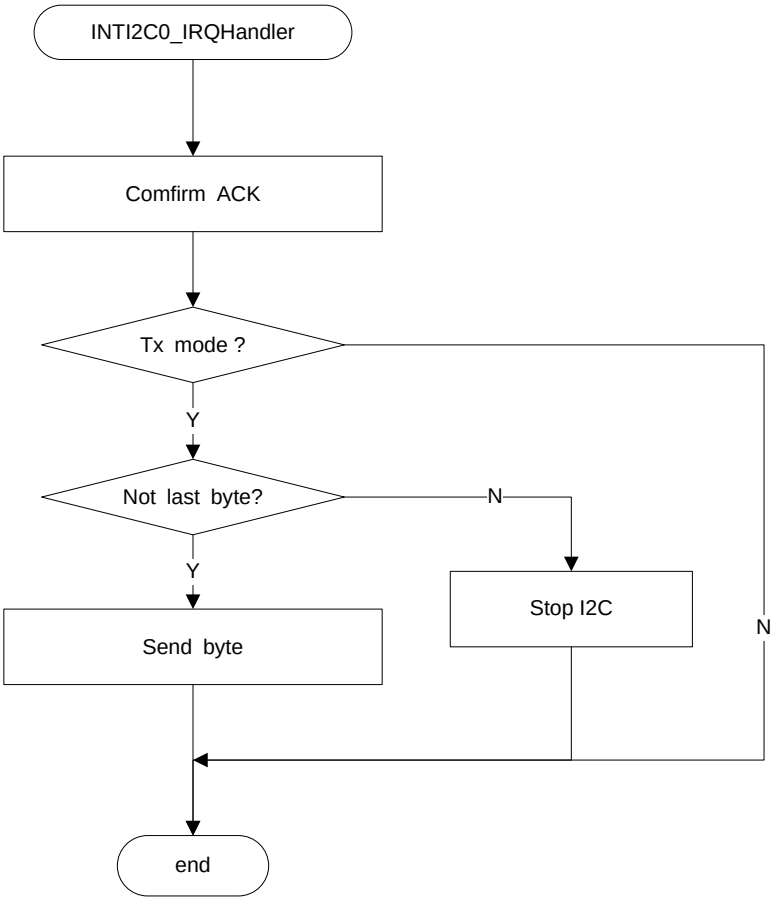
ペリフェラルドライバ(I2C, GPIO)を使用したサンプルプログラムです。

以下の例が含まれます:

1. I2C 初期化
2. I2C マスタによるデータ送信

- **フローチャート:**


main
Set GPIO and Initialize I2C as master;
Enable I2C interrupt
N
SW3 pressed ?
Y
Set I2C Tx data buffer and Tx number
Set I2C Rx number
I2C start send
N
I2C end ?
Y
gI2CWCnt = 0U
gI2CMode = MODE_I2C_IDLE



- **サンプルプログラムのコードと説明**

まず、GPIO を I2C に設定します。

```
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_1, GPIO_BIT_0);
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_1, GPIO_BIT_1);
GPIO_SetOutputEnableReg(GPIO_PC, GPIO_BIT_0, ENABLE);
GPIO_SetOutputEnableReg(GPIO_PC, GPIO_BIT_1, ENABLE);
GPIO_SetInputEnableReg(GPIO_PC, GPIO_BIT_0, ENABLE);
GPIO_SetInputEnableReg(GPIO_PC, GPIO_BIT_1, ENABLE);
GPIO_SetOpenDrain(GPIO_PC, GPIO_BIT_0, ENABLE);
GPIO_SetOpenDrain(GPIO_PC, GPIO_BIT_1, ENABLE);
```

次に、I2C0 をイネーブルし、初期化します。その後 INTI2C0 をイネーブルにします。

```
myI2C.I2CSelfAddr = SELF_ADDR;
myI2C.I2CDataLen = I2C_DATA_LEN_8;
myI2C.I2CACKState = ENABLE;
myI2C.I2CClkDiv = I2C_SCK_CLK_DIV_32;
myI2C.PrescalerClkDiv = I2C_PRESCALER_DIV_12;
```

```
    I2C_SWReset();
    I2C_Init(&myI2C);
    NVIC_EnableIRQ(INTI2C0_IRQn);
    I2C_SetINTReq(ENABLE);
```

上記設定を行った後、I2C 送信を開始します。
I2C 送信バッファと送信バッファ長を設定します。

```
    /* Initialize Tx buffer and Tx length */
    case MODE_I2C_INITIAL:
        gI2CTxDataLen = 7U;
        gI2CTxData[0] = gI2CTxDataLen;
        gI2CTxData[1] = 'T';
        gI2CTxData[2] = 'O';
        gI2CTxData[3] = 'S';
        gI2CTxData[4] = 'H';
        gI2CTxData[5] = 'I';
        gI2CTxData[6] = 'B';
        gI2CTxData[7] = 'A';

        gI2CWCnt = 0U;
        gI2CMode = MODE_I2C_START;
        break;
```

I2C バスが空いているかどうかを確認し、I2C_SetSendData()関数にて“SLAVE_ADDR”データを設定します。また送信方向を“I2C_SEND”から I2C データバッファへ設定します。その後、I2C_GenerateI2CStart()関数を使用して、I2C 動作を開始します。

```
    /* Check I2C bus state and start TRx */
    case MODE_I2C_START:
        i2c_state = I2C_GetState();
        if (!i2c_state.Bit.BusState) {
            gI2CMode = MODE_I2C_TRX;
            I2C_SetSendData(SLAVE_ADDR | I2C_SEND);
            I2C_GenerateStart();
        } else {
            /* Do nothing */
        }
        break;
```

データ送信は INTI2C0 ハンドラ内で行います。
INTI2C0 ハンドラ内で、I2C バス状態を取得し、その値で I2C マスタ送信プロセスを決定します。I2C マスタ送信中は、I2C_SetSendData() で次のデータを送信し、I2C プロセス終了時には、I2C_GenerateStop() で I2C を停止します。

```
void INTI2C_IRQHandler(void)
{
  I2C_I2CState i2c_sr;
```

```
  i2c_sr = I2C_GetState();
  if (i2c_sr.Bit.MasterSlave) {            /* Master mode */
     if (i2c_sr.Bit.TRx) {     /* Tx mode */
        if (i2c_sr.Bit.LastRxBit) { /* LRB=1: the receiver requires no further data. */
           I2C_GenerateStop();
        } else {                    /* LRB=0: the receiver requires further data. */
          if (gI2CWCnt <= gI2CTxDataLen) {
           I2C_SetSendData(gI2CTxData[gI2CWCnt]);             /* Send next data */
            gI2CWCnt++;
          } else {             /* I2C data send finished. */
           I2C_GenerateStop();    /* Stop I2C */
           gI2C_stop_flag = 1U;
          }
     }
    } else {                          /* Rx Mode */
        /* Do nothing */
    }
  } else {                          /* Slave mode */
          /* Do nothing */
     }
}
```

7-7 LVD

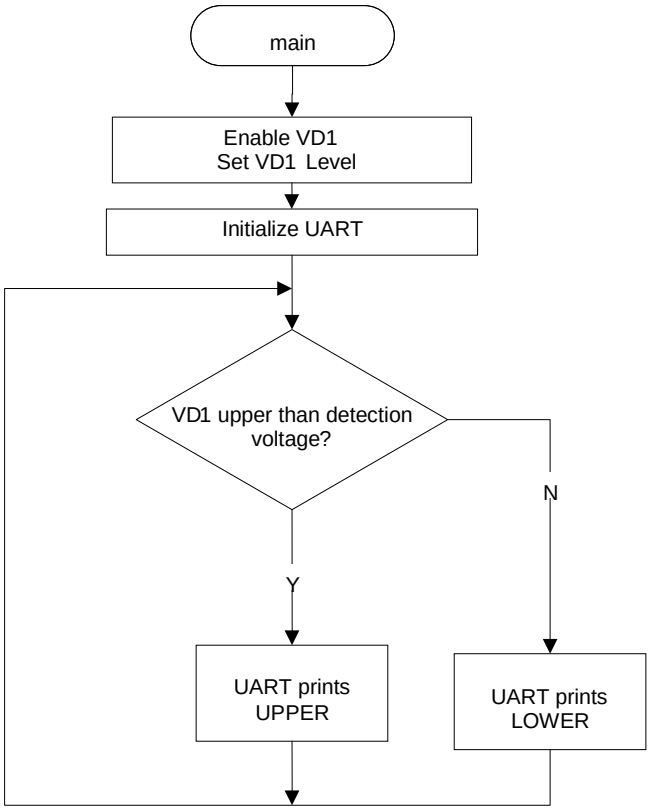
7-7-1 例: LVD

ペリフェラルドライバ(LVD, 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. LVD の初期化
2. LVD 状態の監視

- **フローチャート:**



- **サンプルプログラムのコードと説明**

電圧検出 1 の設定モードと検出レベルをイネーブルにします。

```
LVD_EnableVD1();
LVD_SetVD1Level(LVD_VDLVL1_290);
```

次にシステムを初期化します。

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

その後 while(1)内にて VD1 の状態を監視します。
VD1 が検出電圧より高い場合、UART に“UPPER”を出力し、VD1 が検出電圧より低い場合、UART 上には“LOWER”と出力します。

```
while (1) {
    if (LVD_GetVD1Status() == LVD_VD_UPPER) {
        common_uart_disp(“UPPER\n”);
    } else {
        common_uart_disp(“LOWER\n”);
    }
}
```

7-8 TMR16A

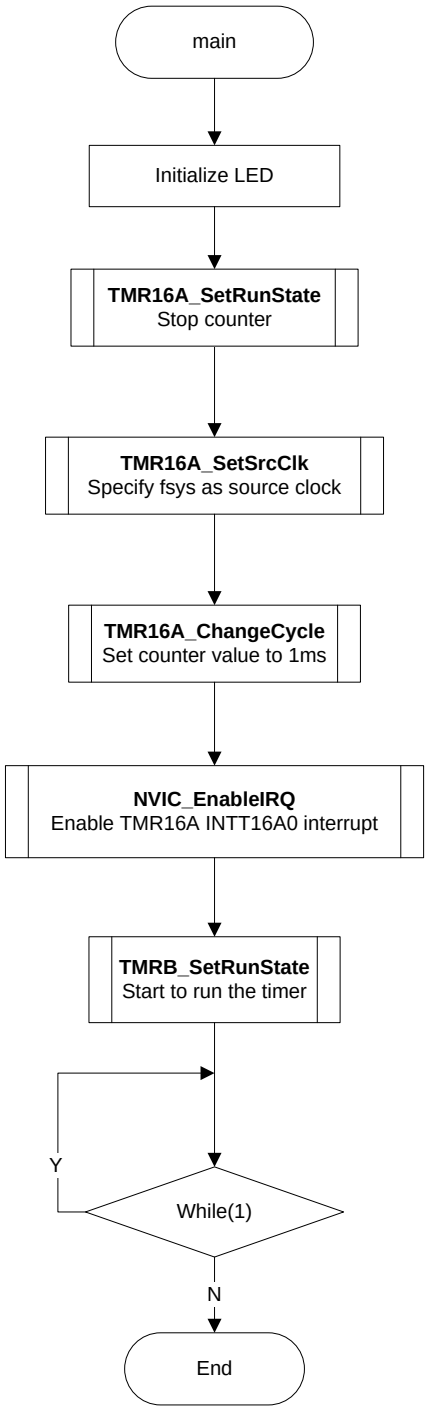
7-8-1 例: 汎用タイマ

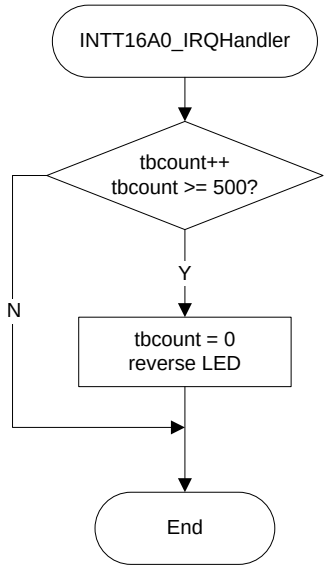
ペリフェラルドライバ (TMR16A, GPIO) を使用したプログラムサンプルです。

以下の例が含まれます:

1. TMR16A の初期化
2. 1ms の汎用タイマを生成

- **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、評価ボード上の LED を初期化し、LED を On します。

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

このサンプルプログラムでは、周期の設定を TMR16A_1MS マクロにて 0x4E20U を定義することで 1ms の周期に設定します。このカウント値は以下により算出されます。
ftmrb = fphiT0 = fsys = fc = 20MHz、Ttmrb = 1/20us, 1ms*20 us = 20000 = 0x4E20 (クロック設定の詳細は CG 章を参照してください。)

その後、カウントを開始します。

```
    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);
```

main 関数の“While(1)”ループの中で、割り込み発生を待ちます。割り込みルーチンの中で、カウントアップし、500ms をカウントすると LED の状態を反転させ、再度カウントアップします。

```
    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 */
    }
```

7-9 TMRB

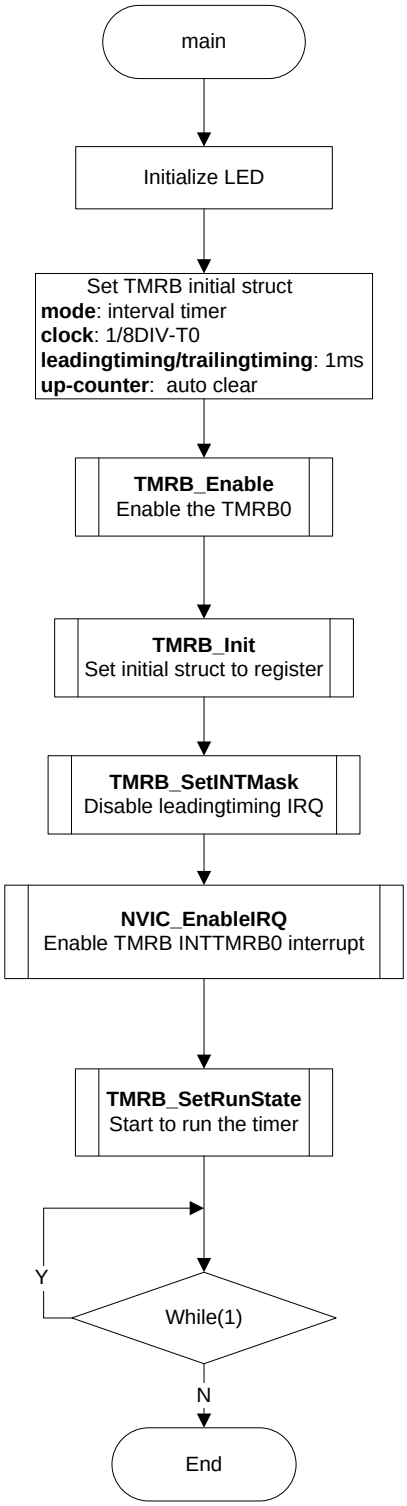
7-9-1 例: 汎用タイマ

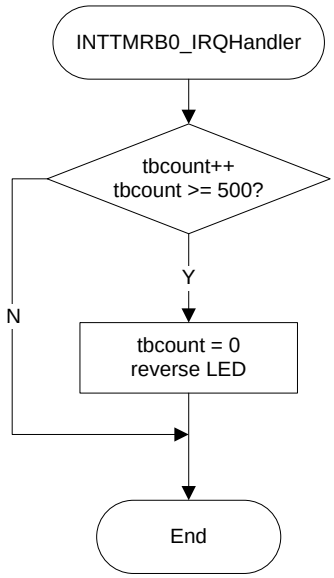
ペリフェラルドライバ (TMRB, GPIO) を使用したプログラムサンプルです。

以下の例が含まれます:

1. TMRB0 の初期化
2. 1ms の汎用タイマを生成

- **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 を初期化し、LED を On します。

```
LED_Init();                                    /* LED initialize */
LED_Off(LED_ALL);                              /* Turn off 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。このデモでは、1ms の周期とデューティを設定します。TMRB_1MS マクロは、0x9C4 です。(φT0=fsys=20MHz, ftmrb = 1/8φT0 = 2.5MHz, Ttmrb = 0.4us, 1ms/0.4us = 2500 = 0x9C4 (クロック設定についての詳細は CG 章を参照してください))

```
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。INTTB0 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 */
TMRB_SetINTMask(TSB_TB0,
                TMRB_MASK_MATCH_LEADINGTIMING_INT)
NVIC_EnableIRQ(INTTMRB0_IRQn);          /* enable INTTMRB0 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 */
}
```

7-9-2 例: PPG 波形出力

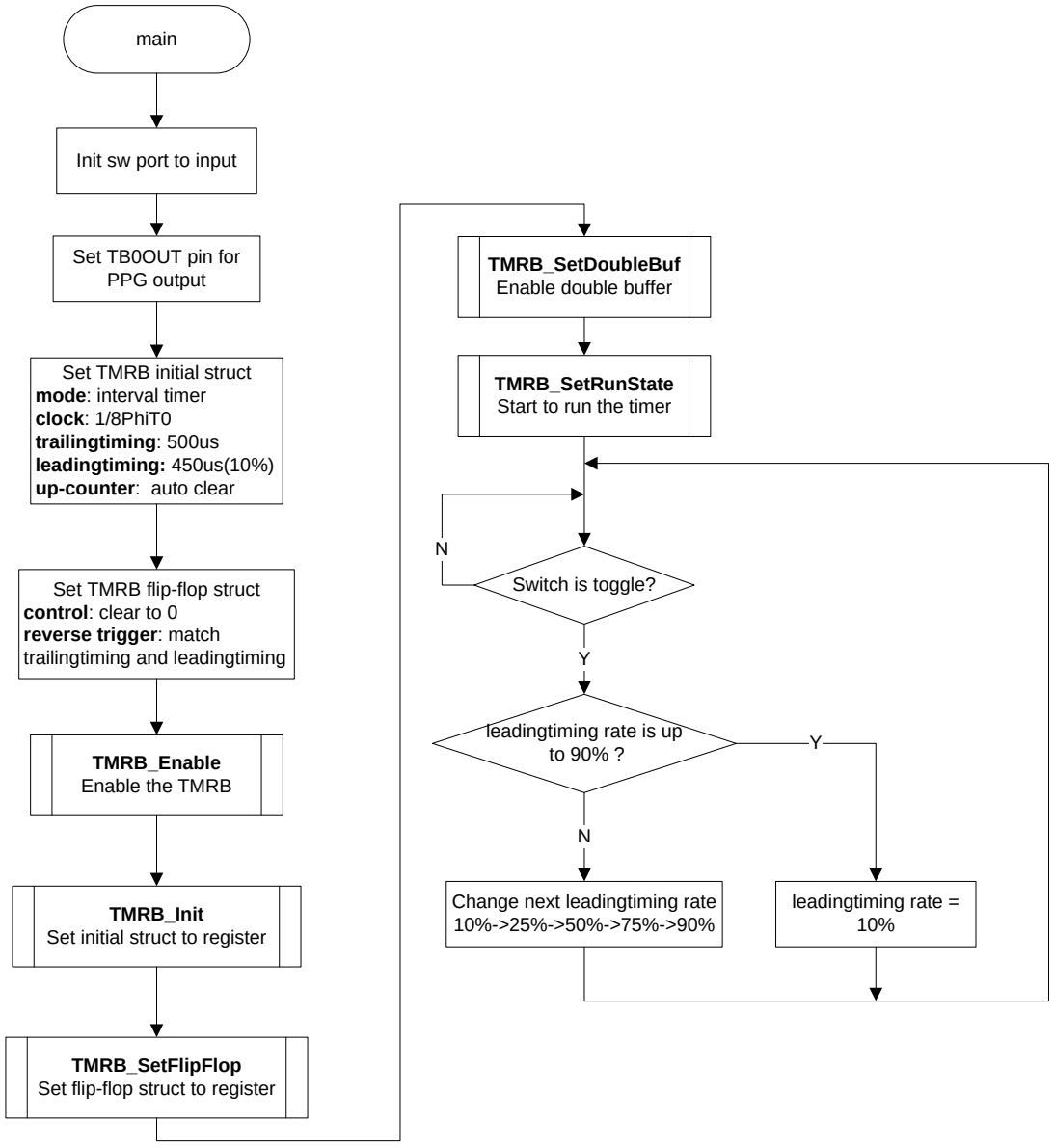
ペリフェラルドライバ (TMRB, GPIO) を使用したサンプルプログラムです。

以下の例が含まれます:

1. TMRB0 の初期化

2. PPG 機能の設定と開始
3. PPG デューティの調整

- **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 PPG 波形出力用に PC3 を TB0OUT に設定します。

```
SW_Init();                          /* Init sw port to input */
/* Set PC3 as TB0OUT for PPG output */
GPIO_SetOutput (GPIO_PC, GPIO_BIT_3);
GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_1, GPIO_BIT_3);
```

TMRB の初期化用構造体を準備し、TMRB モード、クロック、アップカウンタクリア設定、サイクル、デューティを設定します。この例では 500us のサイクルを設定するため、

TMRB0TIME マクロを定義してあります。このマクロは 0x4E2 です。(φT0 = fsys = fc = 20MHz, ftmrb = 1/8 φT0 = 2.5MHz, Ttmrb = 0.4us, 500us/0.4us = 1250 = 0x4E2 (クロック設定の詳細は CG 章を参照してください))

```
    TMRB_InitTypeDef m_tmrb;

    m_tmrb.Mode = TMRB_INTERVAL_TIMER;          /* internal timer */
    m_tmrb.ClkDiv = TMRB_CLK_DIV_8;             /* 1/8PhiT0 */
    m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;         /* up-counter auto clear */
    m_tmrb.TrailingTiming = TMRB0TIME;          /* T = 500us */
    m_tmrb.LeadingTiming = LeadingTiming[Rate];  /* leadingtiming, initial
value 10% */
```

フリップフ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_TB0);
    TMRB_Init(TSB_TB0, &m_tmrb);
    TMRB_SetFlipFlop(TSB_TB0, &PPGFFInital);
    TMRB_SetDoubleBuf(TSB_TB0, ENABLE);          /* enable double buffer */
    TMRB_SetRunState(TSB_TB0, TMRB_RUN);
```

スイッチ状態の変化を待ちます。

```
    keyvalue = SW_Get(SW1);
    if (keyvalue == 1U) {
        delay(0xFFFFU);         /* noise cancel */
        keyvalue = SW_Get(SW1);
        if (keyvalue == 0U) {
            changestatus = 1U;
        } else {
                /* Do nothing */
        }
    } else {
            /* Do nothing */
    }
```

スイッチ状態が変化すると、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 再び 90%から 10%になります。

```
    if (changestatus == 1U) {
```

```
            Rate++;                    /* change leadingtiming rate */
            if (Rate >= LEADINGTIMINGMAX) {
                Rate = LEADINGTIMINGINIT;
            } else {
                /* Do nothing */
            }
            TMRB_ChangeLeadingTiming(TSB_TB0, LeadingTiming[Rate]);
            changestatus = 0U;
        } else {
            /* Do nothing */
        }
```

デューティの算出方法:
Trailing Timing = 500us, ftmrb = 1/8 fphiT0 = 2.5MHz, Ttmrb = 0.4us (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 = 0x465U

LeadingTiming = 25%: High 幅は 500*25% = 125us, Low 幅は 500-125 = 375us, カウンタ値= 375us/Ttmrb = 0x3A9U

LeadingTiming = 50%: High 幅は 500*50% = 250us, Low 幅は 500-250 = 250us, カウンタ値= 250us/Ttmrb = 0x271U

LeadingTiming = 75%: High 幅は 500*75% = 375us, Low 幅は 500-375 = 125us, カウンタ値 = 125us/Ttmrb = 0x138U

LeadingTiming = 90%: High 幅は 500*90% = 450us, Low 幅は 500-450 = 50us, カウンタ値 = 50us/Ttmrb = 0x7DU

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] = { 0x465U, 0x3A9U, 0x271U, 0x138U, 0x7DU };
/* leadingtiming: 10%, 25%, 50%, 75%, 90% */
```

7-10 SIO/UART

7-10-1 例: リターゲット

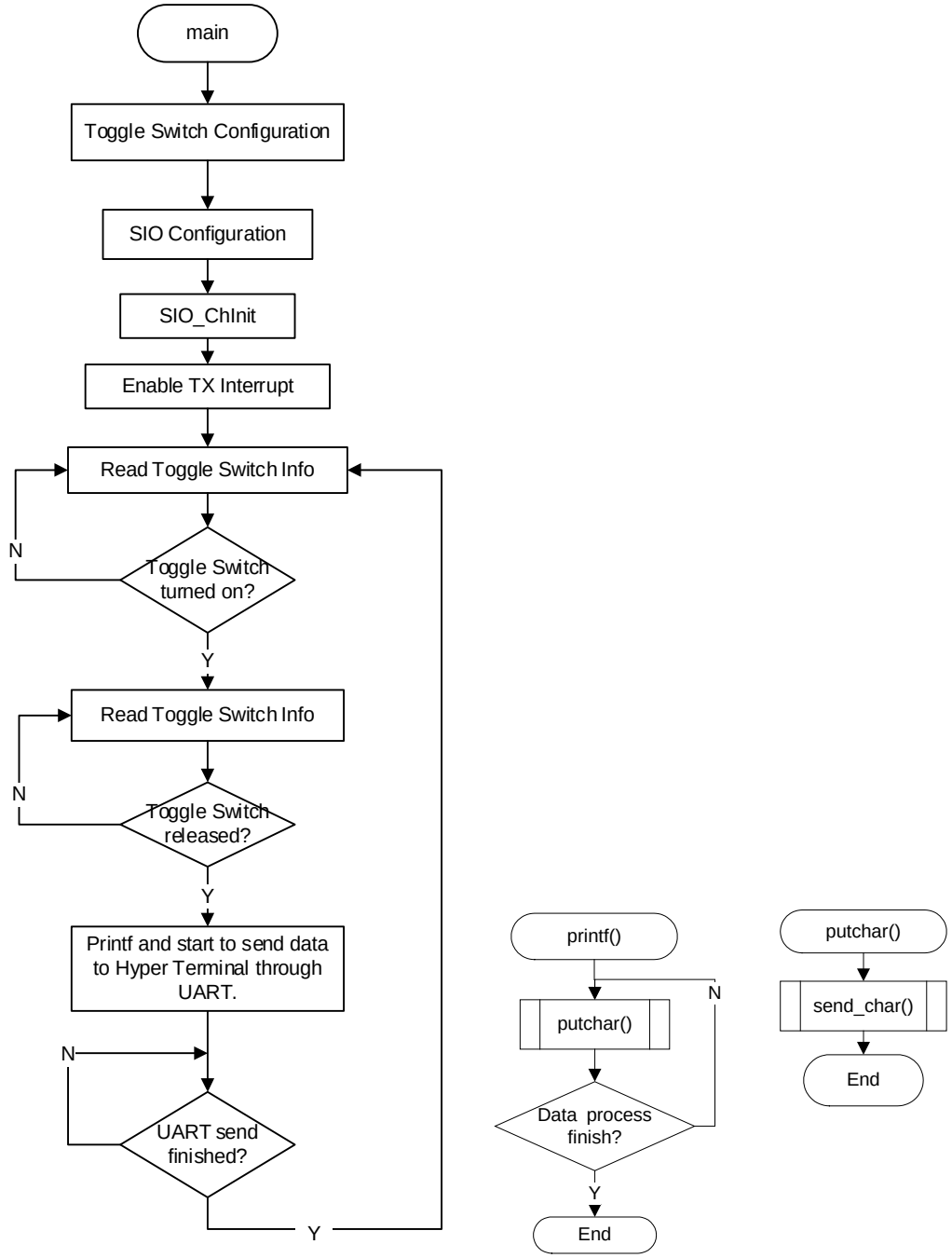
ペリフェラルドライバ (UART, GPIO)を使用したサンプルプログラムです。

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 UART2 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:**



- **サンプルプログラムのコードと説明:**

GPIO を UART に設定します。

```
GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_4, ENABLE);
GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_4, DISABLE);
GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_4);
```

UART_InitTypeDef 構造体を準備します。データを設定後、UART を初期化します。以

下は設定例です。

```
	UART_InitTypeDef myUART;

	/* configure SIO2 for reception */
	UART_Enable(UART_RETARGET);
	myUART.BaudRate = 115200U; /* baud rate = 115200 */
	myUART.DataBits = UART_DATA_BITS_8; /* no handshake, 8-bit data, clock by baud rate generator */
	myUART.StopBits = UART_STOP_BITS_1; /* 1-bit stop, LSB, W-buff enable */
	myUART.Parity = UART_NO_PARITY;
	myUART.Mode = UART_ENABLE_TX;
	myUART.FlowCtrl = UART_NONE_FLOW_CTRL;
	UART_Init(UART_RETARGET, &myUART);
```

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)
```

データ送信を行います。UART 経由で printf()にてデータを送信します。

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

最後に UART2 の送信割り込み処理ルーチンを準備します。
以下は UART2 の送信割り込み処理ルーチンです。

```
void INTTX2_IRQHandler(void)
{
	if (gSIORdIndex < gSIOWrIndex) {	/* buffer is not empty */
		UART_SetTxData(UART_RETARGET,	gSIOTxBuffer[gSIORdIndex++]); /* send data */
		fSIO_INT = SET;		/* SIO2 INT is enable */
	} else {
		/* disable SIO2 INT */
		fSIO_INT = CLEAR;
		NVIC_DisableIRQ(RETARGET_INT);
		fSIOTxOK = YES;
	}
	if (gSIORdIndex >= gSIOWrIndex) {	/* reset buffer index */
		gSIOWrIndex = CLEAR;
		gSIORdIndex = CLEAR;
	} else {
		/* Do nothing */
	}
}
```

IAR コンパイラでは printf()関数が putchar()関数をコールし、RealView コンパイラでは fputc()関数をコールし、UART2 へデータを出力します。

```
#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(UART_RETARGET, gSIOTxBuffer[gSIORdIndex++]);
        NVIC_EnableIRQ(RETARGET_INT);
    }

    return ch;
}
```

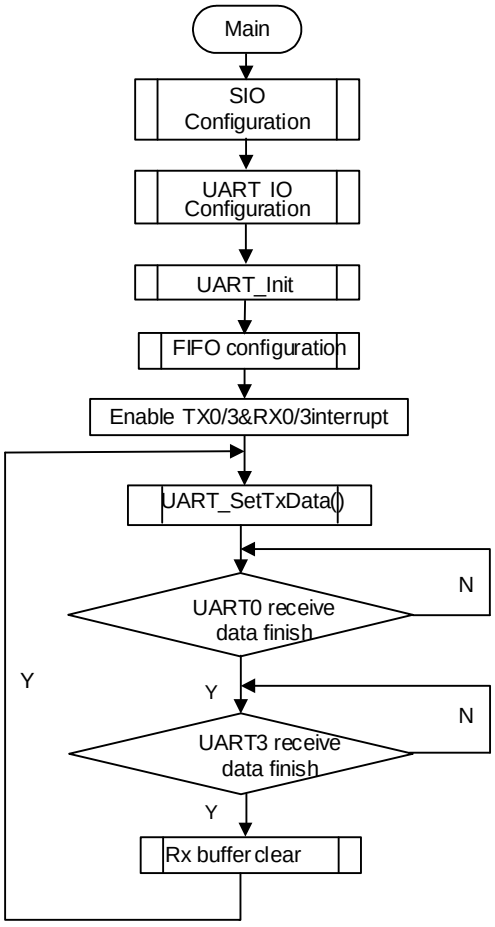
7-10-2　例: UART FIFO

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 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**

最初に GPIO の設定と UART の初期化を行います。
GPIO ドライバを使用して、GPIO を UART0 と UART3 に設定します。

```
void SIO_Configuration(TSB_SC_TypeDef * SCx)
{
	if (SCx == TSB_SC0) {
		TSB_PD->CR |= GPIO_BIT_3;
		TSB_PD->FR1 |= GPIO_BIT_3;
		TSB_PD->FR1 |= GPIO_BIT_2;
		TSB_PD->IE |= GPIO_BIT_2;
	} else if (SCx == TSB_SC1) {
		TSB_PG->CR |= GPIO_BIT_2;
		TSB_PG->FR1 |= GPIO_BIT_2;
		TSB_PG->FR1 |= GPIO_BIT_1;
		TSB_PG->IE |= GPIO_BIT_1;
	} else if (SCx == TSB_SC2) {
		TSB_PE->CR |= GPIO_BIT_4;
		TSB_PE->FR1 |= GPIO_BIT_4;
		TSB_PE->FR1 |= GPIO_BIT_3;
		TSB_PE->IE |= GPIO_BIT_3;
	} else if (SCx == TSB_SC3) {
		TSB_PF->CR |= GPIO_BIT_3;
		TSB_PF->FR2 |= GPIO_BIT_3;
		TSB_PF->FR2 |= GPIO_BIT_2;
		TSB_PF->IE |= GPIO_BIT_2;
	}
}
```

UART_InitTypeDef 構造体を用意し、すべてのメンバを設定します。

```
	UART_InitTypeDef myUART;

	/* configure SIO0 for reception */
	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 のドライバを使用して、UART0/3 の各チャネルの許可と初期設定を行います。

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

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

FIFO の設定を行います。

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

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

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

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

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

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

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

    UART_RxFIFOClear(UART0);
    UART_RxFIFOClear(UART3);

    UART_TxBufferClear(UART0);
    UART_TxBufferClear(UART3);

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

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

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

上記設定を行い、その後 UART0/3 の各割り込みを許可します。

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

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

最後に UART0/3 の送信割り込みと受信割り込みの割り込み処理ルーチンを準備します。
以下は UART0 の送信割り込み処理ルーチンです。

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

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

以下は UART3 の送信割り込み処理ルーチンです。

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

    if (TxCounter1 < NumToBeTx1) {
```

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

以下は UART0 の受信割り込み処理ルーチンです。

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

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

以下は UART3 の受信割り込み処理ルーチンです。

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

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

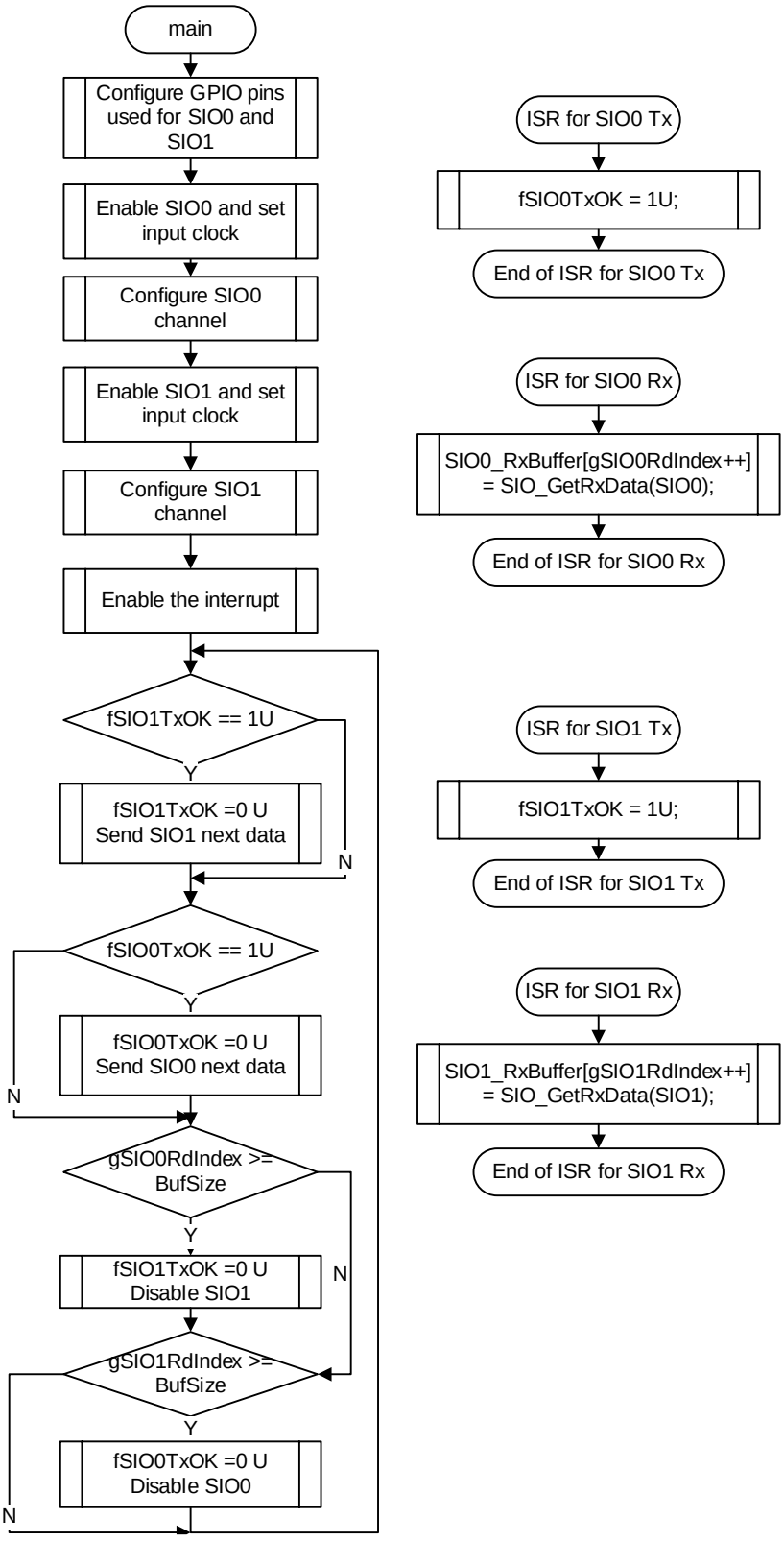
7-10-3　例: SIO

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。

以下の例が含まれます:

1. SIO 動作の基本設定
2. SIO0~SIO1 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**

最初に GPIO 端子を 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 IO port of SIO */
SIO_Configure();
/*Enable the SIO0 channel */
```

```
SIO_Enable(SIO0);

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

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

SIO1 を有効にし、入力クロックの設定と SIO1 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.TIDLE = SIO_TIDLE_HIGH;
SIO1_Init.TransferMode = SIO_TRANSFER_FULLDPX;
SIO1_Init.TransferDir = SIO_LSB_FRIST;
SIO1_Init.Mode = SIO_ENABLE_TX | SIO_ENABLE_RX;
SIO1_Init.DoubleBuffer = SIO_WBUF_ENABLE;
SIO1_Init.TXDEMP = SIO_TXDEMP_HIGH;
SIO1_Init.EHOLDTime = SIO_EHOLD_FC_64;

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

SIO の送信割り込みと受信割り込みを許可します。

```
/* 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);
```

すべての基本的な設定を行い、その後、データ転送処理に入ります。

```
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 */
        }
    }
```

以下は SIO0 の送信割り込み処理ルーチンです。送信完了フラグをセットします。

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

以下は SIO0 の受信割り込み処理ルーチンです。受信バッファから受信データを取得します。

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

以下は SIO1 の送信割り込み処理ルーチンです。送信完了フラグをセットします。

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

以下は SIO1 の受信割り込み処理ルーチンです。受信バッファから受信データを取得します。

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

7-11 WDT

7-11-1 例: WDT

ペリフェラルドライバ(WDT, 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. 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
3. DEMO2 では、オーバーフロー前に WDT クリアを行い、常時 LED1 を点滅

- **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 LED0 を点滅させます。

```
WDT_WriteClearCode();
```