TOSHIBA

TX03 ペリフェラルドライバ使用例 (TMPM380)

第一版

2017 年 9 月

東芝デバイス＆ストレージ株式会社

CMDR-M380UE-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。

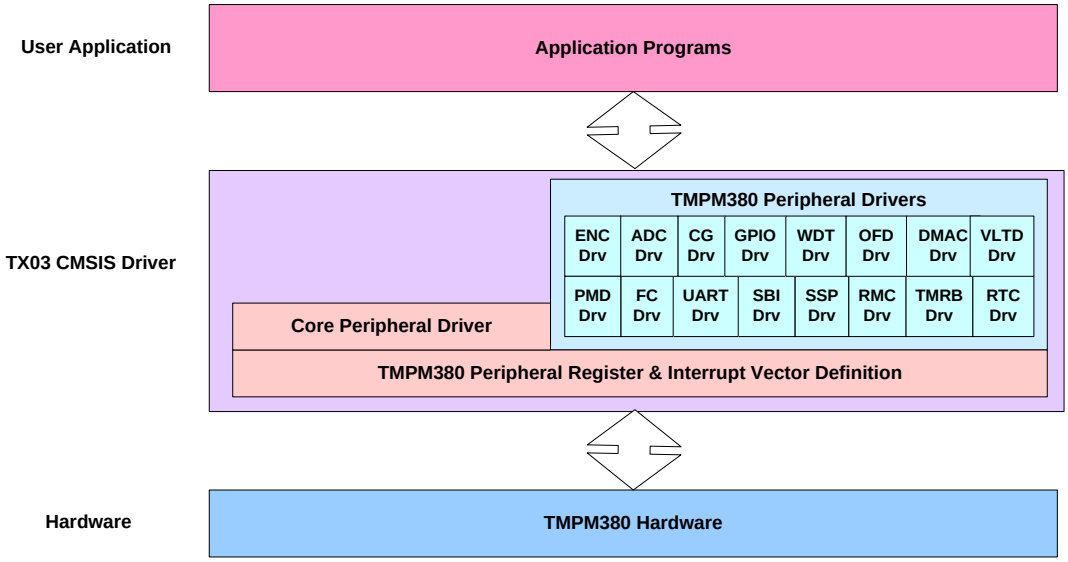
目次

1 はしがき

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TMPM380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
クロック／モード制御 (CG)	クロックギア	使用: CG デモ
	PLL	PLL on (4 逓倍)
低消費電力モード	-	使用: SLEEP モード
SysTick	-	未使用
ウォッチドッグタイマ (WDT)	INT0-INTF	未使用
外部割り込み (INT)		使用: CG サンプル
DMAC	チャネル 1	使用: DMAC サンプル
シリアルチャネル(SIO/UART)	SIO0	使用: DMAC _UART Tx サンプル
	SIO1	使用: DMAC _UART Rx サンプル
	SIO2	使用: UART リターゲットサンプル
	上記以外	未使用
同期式シリアルインタフェース(SSP)	SSP0	使用: SSP 転送サンプル(Tx)

機能	チャネル	使用／未使用
	SSP1	使用: SSP 転送サンプル(Rx)
シリアルバスインタフェース(SBI)	SBI0	使用: マスタ(Tx)
	SBI1	使用: スレーブ(Rx)
リモコン判定機能 (RMC)	RMC0	使用: RMC 信号受信
16 ビットタイマ(TMRB)	TMRB0	使用: TMRB サンプル: 汎用タイマ
	TMRB7	使用: TMRB サンプル: PPG 出力
	上記以外	未使用
16 ビット多目的タイマ	MPT1	使用: PMD サンプル
	上記以外	未使用
RTC	-	使用: RTC サンプル
周波数検知回路 (OFD)	-	使用: OFD サンプル
電圧検出回路(VLTD)	-	使用: VLTD サンプル
エンコーダ入力回路(ENC)	ENC0	使用: ENC サンプル
	ENC1	未使用
10 ビット A/D コンバータ	AIN0	使用: ADC データリード
	上記以外	未使用
Flash	-	使用: FC サンプル

4 端子用途

本サンプルプログラムは、TMPM380-SKボードとTMPM380表か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。

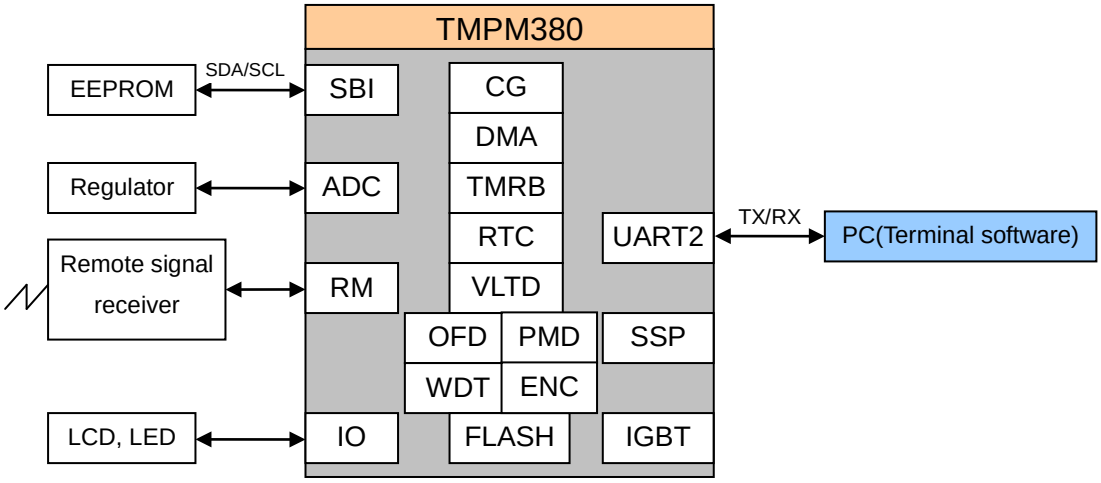
Pin No.	端子名	用途
1	PD4, SCLK2, CTS2	未使用
2	PD3, INT9	未使用
3	PD2, ENCZ0, INTD	SW3
4	PD1, ENCB0, TB5OUT	SW2 / ENCB0
5	PD0, ENCA0, TB5IN, INTC	SW1 / ENCA0
6	DVSS	GND
7	PC7, MT0IN, RX4	未使用
8	PC6, EMG0	未使用
9	PC5, ZO0, MTOUT10, MTTB0IN, SCLK4, CTS4	未使用
10	PC4, WO0, MTOUT00, MTTB0OUT	未使用
11	PC3, YO0, SP0FSS	未使用
12	PC2, VO0, SP0CLK	未使用
13	PC1, XO0, SP0DI, SCL0 / SI0	未使用
14	PC0, UO0, SP0DO, SDA0 / SO0	未使用
15	DVDD5	+5V
16	PP1, XT2	低速クロック
17	PP0, XT1	低速クロック
18	PM1, X2	高速クロック(10MHz)
19	CVSS	GND
20	PM0, X1	高速クロック(10MHz)
21	DVSS	GND
22	PG0, UO1, SDA1 / SO1	PMD UO1出力
23	PG1, XO1, SCL1 / SI1	PMD XO1出力
24	PG2, VO1, SCK1	PMD VO1出力
25	PG3, YO1	PMD YO1出力
26	PG4, WO1, MTOUT01, MTTB1OUT	LED桁選択 / PMD WO1出力
27	PG5, ZO1, MTOUT11, MTTB1IN	LED桁選択 / PMD ZO1出力
28	PG6, EMG1, GEMG1	LED桁選択 / PMD EMG入力
29	PG7, MT1IN	LED桁選択
30	PA0, TB0IN, INT3	LEDデータ
31	PA1, TB0OUT, SCOUT	LEDデータ
32	PA2, TB1IN, INT4	LEDデータ

Pin No.	端子名	用途
33	PA3, TB1OUT, RXIN	LEDデータ
34	PA4, SCLK1, CTS1	LEDデータ/ UART1
35	PA5, TXD1, TB6OUT	LEDデータ/ UART1
36	PA6, RXD1, TB6IN	LEDデータ/ UART1
37	PA7, TB4IN, INT8	LEDデータ
38	PE0, TX0	UART0
39	PE1, RX0	UART0
40	PE2, SCLK0, CTS0	UART0
41	PE3, TB4OUT	未使用
42	DVDD5	+5V
43	PE4, TB2IN, INT5	未使用
44	PE5, TB2OUT	未使用
45	DVSS	GND
46	PE6, TB3IN, INT6	未使用
47	PE7, TB3OUT, INT7	未使用
48	PN0, SP1DO	未使用
49	PN1, SP1DI	未使用
50	PN2, SPICLK	未使用
51	PN3, SPIFSS	未使用
52	PN4, MTOUT02, MTTB2OUT	未使用
53	PN5, MTOUT12, MTTB2IN	未使用
54	PN6, GEMG2	未使用
55	PN7, MT2IN, INTE	未使用
56	PL2, INTF	未使用
57	DVSS	GND
58	MODE	GND
59	RVDD5	+5V
60	RESET	RESET
61	VOUT3	未使用
62	PH0, AIN0, INT0	AIN0
63	PH1, AIN1, INT1	未使用
64	PH2, AIN2, INT2	未使用
65	PH3, AIN3	未使用
66	PH4, AIN4	未使用
67	PH5, AIN5	未使用
68	PH6, AIN6	未使用
69	PH7, AIN7	未使用
70	PI0, AIN8	未使用
71	PI1, AIN9	未使用
72	PJ0, AIN10	未使用
73	PJ1, AIN11	未使用

Pin No.	端子名	用途
74	PJ2, AIN12	未使用
75	PJ3, AIN13	未使用
76	PJ4, AIN14	未使用
77	PJ5, AIN15	未使用
78	PJ6, AIN16, INTA	未使用
79	PJ7, AIN17, INTB	未使用
80	AVSS	GND
81	AVDD5	+5V
82	DVSS	GND
83	DVDD5	+5V
84	PL0, BOOT	未使用
85	FTEST3	未使用
86	PB0, TRACECLK	未使用
87	PB1, TRACEDATA0	未使用
88	PB2, TRACEDATA1	未使用
89	PB3, TMS, SWDIO	TMS
90	PB4, TCK, SWCLK	TCK
91	PB5, TDO, SWV	TDO
92	PB6, TDI	TDI
93	PB7, TRST	TRST
94	PF0, TB7IN	未使用
95	PF1, TB7OUT, ALARM	PPG出力
96	PF2, ENCA1, SCLK3, CTS3	未使用
97	PF3, ENCB1, TX3	未使用
98	PF4, ENCZ1, RX3	未使用
99	PD6, RX2	未使用
100	PD5, TX2	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:
   IAR TMPM380-SK 評価ボード
   TMPM380 評価ボード (非売品)

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 6.40.2
   - KEIL 社:
     1) u-Link: Realview ULINK2
     2) IDE: KEIL uVision MDK 4.54

補足: 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

6 機能説明

6-1 動作モード選択

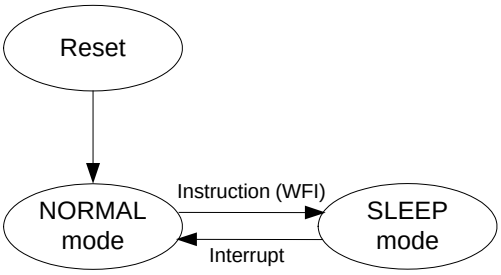
TMPM380 には 5 つの動作モードがあります: NORMAL, SLOW, IDLE, SLEEP, STOP モード
IDLE, SLEEP, STOP モードは低消費電力モードです。
低消費電力モードに移行するには、システム制御レジスタ STBYCR0<STBY2:0>にて IDLE, SLEEP, STOP モードのいずれかを設定し、WFI (Wait For Interrupt)命令を実行します。

- **NORMAL モード:**

CPU コアおよび周辺ハードウェアを高速クロック(fosc1 または fosc2)で動作させるモードです。
リセット解除後は、fosc2(内蔵高速クロック)による NORMAL モードになります。
補足: サンプルプログラムは、SLEEP モードのみ使用します。

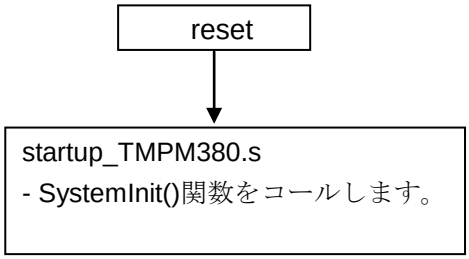
- **SLEEP モード:**

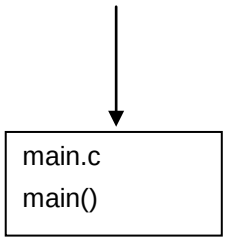
SLEEP モードでは、低速発振(fs)、RTC、RMC が動作します。SLEEP モードが解除されると SLEEP モードへ移行する直前の動作モードへ復帰し、動作を開始します。
RTC 割り込み、RMC 割り込み、外部割込みで SLEEP モードを解除できます。
補足: SLEEP モードのサンプルプログラムは、CG の例に含まれています。



6-2 STARTUP

スタートアップはリセットから main()関数までの間の処理と割り込みベクタが定義されています。





6-3 ADC

IAR 社製 TMPM380-SK の PN0/AIN0 に接続されたポテンションメータの値を変更するサンプルプログラムです。測定された電圧値を評価ボード上の 7 セグメント LED に表示します。

6-4 CG

6-4-1 パワーモード変更

CPU の動作モードを変更するサンプルプログラムです。NORMAL と SLEEP 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 (リモコン)	動作	ターミナル 表示
NORMAL	[9]を押す	NORMAL→ STOP1	Now, Going to Sleep
SLEEP	[7]を押す	STOP1 →NORMAL	Wakeup from Sleep

6-5 DMAC

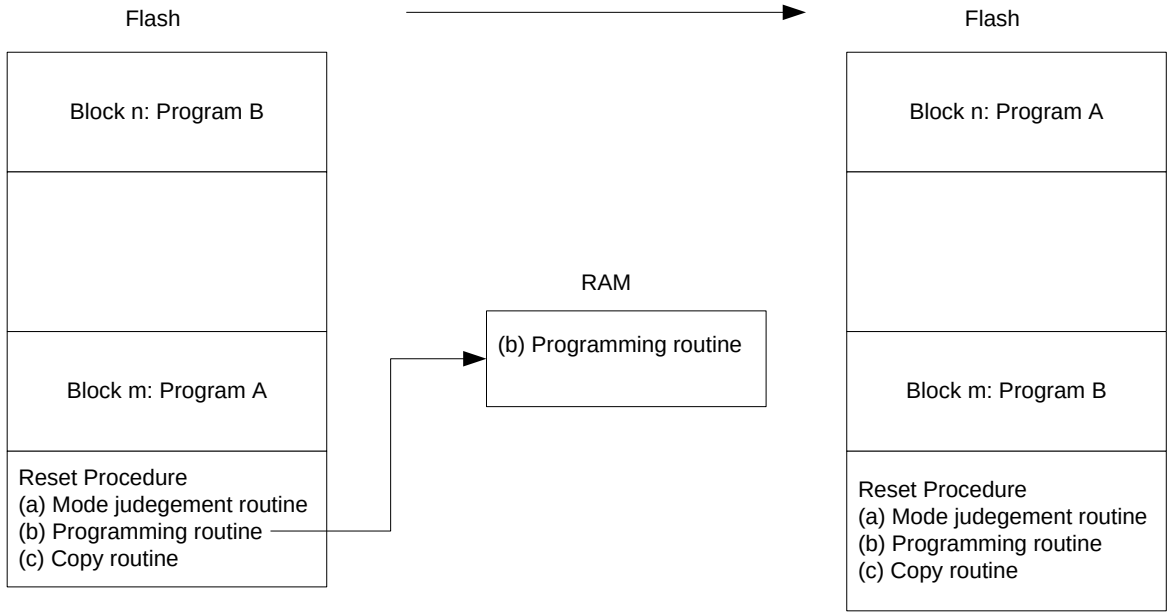
6-5-1 メモリから周辺回路

UART0 から UART1 へ文字列"TOSHIBA"を転送します。この文字列は DMA 転送により RAM 領域から UART0 へ転送され、UART1 で受信します。UART1 からデータを受信した後、文字配列に保存します。
文字配列は RAM 領域に配置されているので、デバッガにて受信データを確認することが可能です。

補足:
プログラムを実行する前に予め UART0 の Tx と UART1 の Rx を接続しておく必要があります。

6-6 FLASH

Flash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ー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。
ープログラム A/B (A: LED4,6,8,10 点滅、B: LED5,7,9,11 点滅)
初期状態は、プログラム A が最初に動作します。
ーSW1 キーはリセット動作のモード判定に使います。
SW1 キーが high 状態 → ユーザブートモードです。
SW1 キーが low 状態 → Normal モードです。

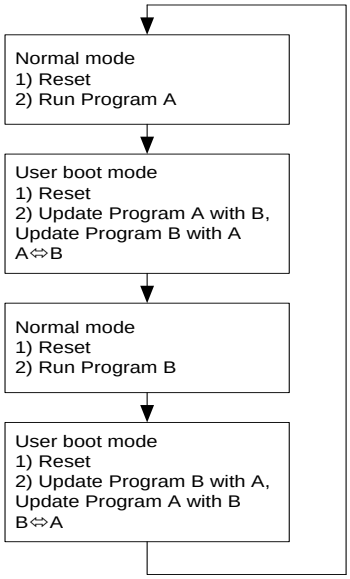


プログラムシーケンス

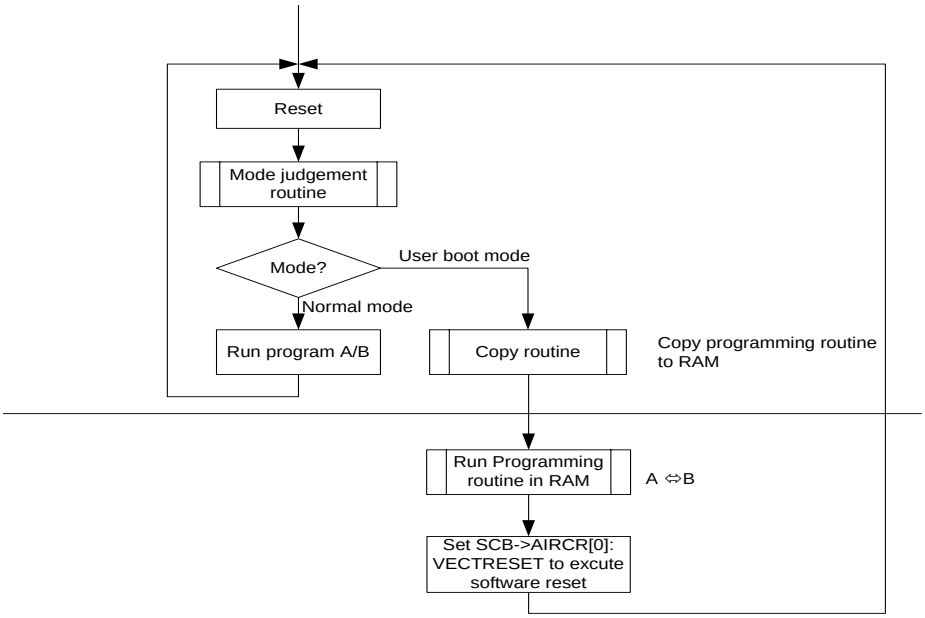
(1) 電源投入
SW1 が High 状態で電源投入またはリセット端子を ON します。
まず Flash 内に書き込まれているプログラム A を実行します。プログラム A は、LED4,6,8,10 を点滅します。

(2) SW1 を Low 状態にし、RESET ボタンを ON します。
リセット処理の中で転送ルーチンが書き換えルーチンを RAM へ転送します。
その後、Flash 内のプログラム A とプログラム B を入れ替えるコードを実行します。
この間、7 セグメント LED に“0"->"1"->"2"->"3"を表示します。

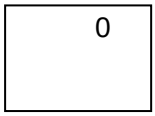
(3) Flash 内のプログラム A と B が入れ替わり、LED に“3”を表示した後に SW1 を Low 状態にすると、プログラムは自動的にソフトウェアリセットします。
その後、Flash 内に書き込まれたプログラム B が実行を始めます。プログラム B は、LED5,7,9,11 を点滅します。



・フローチャート



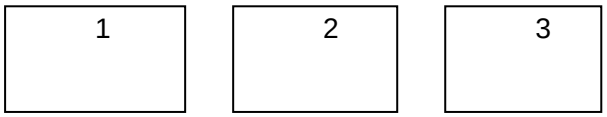
サンプルプログラムがユーザブートモードに移行すると、7 セグメント LED に以下を表示します。



(ユーザブートモード)

RAM 転送か Flash 書き込みを行っている間に 7 セグメント LED は現在の状態を表示します。

以下は 7 セグメント LED 表示の例です。



(RAM 転送中)　(スワップ中)　　(リセット待ち)

6-7　GPIO

GPIO を LED に設定し、LED の点灯/消灯を行うサンプルプログラムです。

6-8　RMC

6-8-1　RMC 受信

リモコン信号データを受信し、デコードするサンプルプログラムです。
またデコードしたデータをデバッガ上のターミナルウィンドウに表示します。
カスタムコードまたはアドレスコードとコマンドコードを Hex フォーマットで表示します。

表示フォーマット:　TMPM380 RMC Demo
RMC_1: XX XX
RMC_1: YY YY

フォーマット	ターミナルウィンドウの表示
1	RMC_1:
2	RMC_2:
3	RMC_3:
4	RMC_4:
5	RMC_5:

補足:
予めリモコン受信を PA3 と接続しておく必要があります。
リモコン受信には 3V 電源を供給する必要があります。

6-9　RTC

RTC を使用して日時を表示するサンプルプログラムです。

初期設定では 2010-10-22, 時刻の設定が 12:50:55, 24 時間表示です。
更新間隔: 1 秒
LCD 表示:

2010/10/22 12:50:55

6-10 SBI

I2C のスレーブモードを確認するサンプルプログラムです。
評価ボード上の 2 本の I2C バス(SBI0 と SBI1)を接続します。
I2C バスの 1 本を I2C マスタとして、もう片方を I2C スレーブとして動作させ、SBI 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 スレーブアドレス: 0xB0
I2C スレーブ(SBI1)は I2C マスタ(SBI0)から“TOSHIBA”を受け取ります。

受信結果はデバッガ内の RAM 変数 gI2CrxData[]によって確認できます。

補足: このサンプルソフトを実行ために評価ボードを以下のように接続します。
PC1(SCL0)端子を PG1(SCL1)端子と接続します。
PC0(SDA0)端子を PG0(SDA1)端子と接続します。

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。

6-11-2 UART FIFO

UART0 から" TMPM3801"というデータを FIFO を使用して UART1 へ送信し、同時に UART1 から" TMPM3802"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=” TMPM3802”と RxBuffer1=" TMPM3801"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

この例では、TMPM380 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)
*補足: この例では、スレーブ側を最初に動作させ、その後、マスタ側を動作させてください。

6-12 SSP

6-12-1 SSP0 から SSP1 への DMAC 転送

SSP0 を SPI マスタ、SSP1 を SPI スレーブに設定し、DMAC 転送を行うサンプルプログラムです。
(評価ボード上の 2 本の SSP バス(SSP0 と SSP1)を接続します)

補足: 評価ボードを以下のように変更します。

	SSP0 端子	内容	SSP1 端子
1	PC3/SP0FSS	PC3 と PN3 を接続します。	PN3/SP1FSS
2	PC2/SP0CLK	PC2 と PN2 を接続します。	PN2/SP1CLK
3	PC1/SP0DI	PC1 と PN0 を接続します。	PN1/SP1DI
4	PC0/SP0DO	PC0 と PN1 を接続します。	PN0/SP1DO

6-12-2 SSP0 セルフループバック

この例は、送信側はデータを送信し、受信側は受信データをチェックするサンプルプログラムです。

6-13 TMRB

6-13-1 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。

6-13-2 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。
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%

6-14 VLTD

VLTDによって電圧状態を検出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、LED4を点灯し、UARTに“Low voltage!”を表示します。(デバッグモード時)
電圧が検出電圧より高い場合、LED4を消灯し、“High voltage!”を表示します。(デバッグモード時)

6-15 WDT

ウォッチドッグタイマは、高速クロックが停止する STOP モードでは使用できません。リセット後ウォッチドッグタイマは有効となり、SystemInit()関数で無効になります。

ドライバ使用方法ステップ:

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。

6-16 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 7 (ENCA0)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 6 (ENCB0)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 LED4 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 LED4 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。

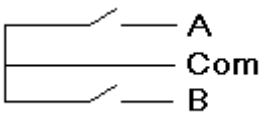


図 1

6-17 PMD

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 30)を DVSS (pin 84)とプルダウン接続します。(LED4 が点灯します)これは EMG 保護状態であることを意味します。
3. EMG1 (pin 30)を DVDD5 (pin 85)とプルアップ接続します。(LED4 が消灯します)これは PMD が通常動作であることを意味します。
4. UO1 (pin 24) をオシロスコープに接続し、波形を確認します。
5. XO1 (pin 25) をオシロスコープに接続し、波形を確認します。
6. VO1 (pin 26) をオシロスコープに接続し、波形を確認します。
7. YO1 (pin 27) をオシロスコープに接続し、波形を確認します。
8. WO1 (pin 28) をオシロスコープに接続し、波形を確認します。
9. ZO1 (pin 29) 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 820us 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 820us 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 820us 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 820us です。ZO1 は WO1 を反転させた形です。

7 ソフトウェア

本ソフトウェアは、TMPM380 MCU の主要機能を TMPM380 評価ボード上でデモ、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
        │   tmpm380_wdt_int.c
        ├─TX03_CMSIS
        │   │   system_TMPM380.c
        │   │   system_TMPM380.h
        │   │   TMPM380.h
        │   │
        │   └─startup
        │         startup_TMPM380.s
        │
        ├─TX03_Periph_Driver
        │   ├─inc
        │   │   tmpm380_wdt.h
        │   │   tmpm380_gpio.h
        │   │   tx03_common.h
        │   │
        │   └─src
        │         tmpm380_wdt.c
        │         tmpm380_gpio.c
        └─TMPM380-EVAL
                  led.c
                  led.h
```

KEIL MDK:

```
├─WDT_NMI
        ├─APP
        │   main.c
        │   tmpm380_wdt_int.c
        │
        ├─TX03_CMSIS
        │   │   system_TMPM380.c
        │   │   system_TMPM380.h
        │   │   TMPM380.h
        │   │
        │   └─startup
        │         startup_TMPM380.s
        │
        ├─TX03_Periph_Driver
        │    tmpm380_wdt.c
```

```
│    tmpm380_gpio.c
│
└─TMPM380-EVAL
      led.c
```

7-1 ADC

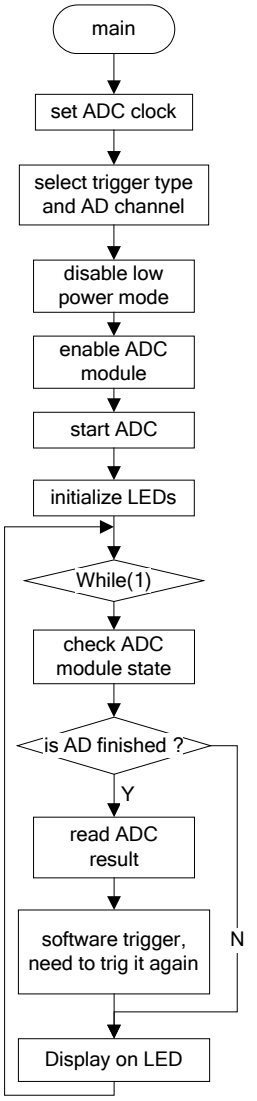
7-1-1 例: ADC データリード

ペリフェラルドライバ(ADC, GPIO)を用いたサンプルプログラムです。

この例では以下を行います。

1. ADC 設定と初期化を行います。
2. AD 変換の開始と AD 変換結果の読み出しを行います。

- **フローチャート:**



- **サンプルプログラムのコードと説明**

最初に AD コンバータクロックの選択、チャネルの選択、ソフトウェア起動モードの許可、消費電力削減の許可を行い、その後 ADC 動作を許可します。

```
/* set ADC clock */
ADC_SetClk(ADC_HOLD_FIX, ADC_FC_DIVIDE_LEVEL_NONE);

/* select trigger and AD channel, this time we use sofeware trigger, */
/* the VR1 is connected to AIN0, remember to input with macro TRG_ENABLE() */
ADC_SetSWTrg(ADC_REG0, TRG_ENABLE(AIN0));

/* to let ADC module work, we must disable low power mode */
ADC_SetLowPowerMode(DISABLE);

/* enable ADC module */
ADC_Enable();
```

AD 変換を開始します。

```
ADC_Start(TRG_SOFTWARE);
```

AD 変換を開始すると、AD 変換終了フラグがセットされるまで待ちます。AD 変換フラグがセットされると、AD 変換結果を取得し、LED に表示します。

```
while (1) {
    /* check ADC module state */
    adcState = ADC_GetConvertState(TRG_SOFTWARE);

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

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

        /* software trigger, need to trig it again */
        ADC_Start(TRG_SOFTWARE);
    }

    delay++;
    if (delay >= adjustLight) {
        delay = 0U;
        DisplayLED(myResult);
    } else {
        /* do nothing */
    }
}
```

7-2 CG

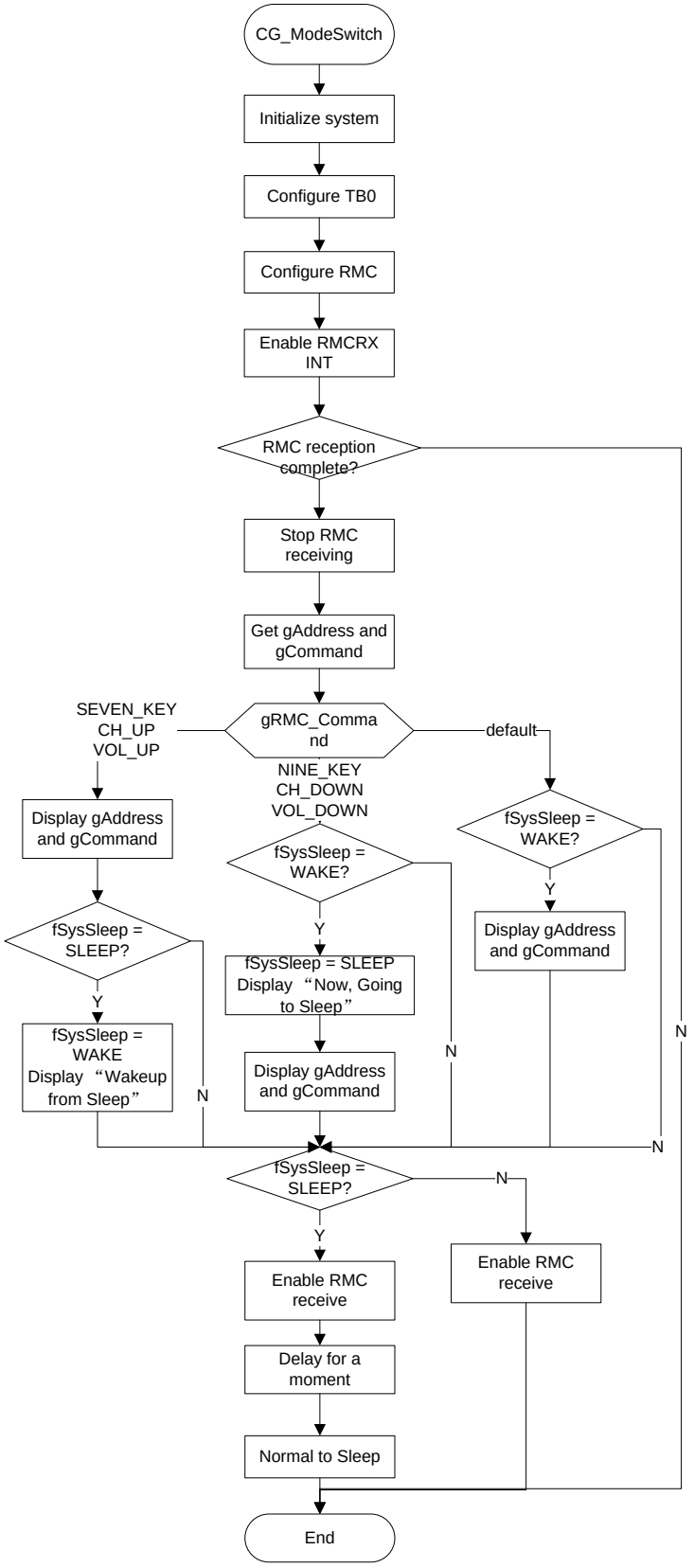
7-2-1 例: パワーモード変更

ペリフェラル・ドライバ(CG, RMC, GPIO, TMRB)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. 基本的な CG 動作の設定
2. NORMAL モードと SLEEP モードの切り替え方法
3. マルチクロック回路の許可方法

- **フローチャート**


CG_ModeSwitch
Initialize system
Configure TB0
Configure RMC
Enable RMCRX INT
RMC reception complete?
Stop RMC receiving
Get gAddress and gCommand
gRMC_Command
SEVEN_KEY
CH_UP
VOL_UP
NINE_KEY
CH_DOWN
VOL_DOWN
default
Display gAddress and gCommand
fSysSleep = SLEEP?
fSysSleep = WAKE
Display "Wakeup from Sleep"
fSysSleep = WAKE?
fSysSleep = SLEEP
Display "Now, Going to Sleep"
Display gAddress and gCommand
fSysSleep = WAKE?
Display gAddress and gCommand
fSysSleep = SLEEP?
Enable RMC receive
Delay for a moment
Normal to Sleep
Enable RMC receive
End
Y
N

- **サンプルプログラムのコードと説明**

以下はペリフェラルドライバ(CG)を使用して NORMAL モードと SLEEP モードの切り替えを行うサンプルプログラムです。

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);
    /* Set SCOUT source */
    CG_SetSCOUTSrc(CG_SCOUT_SRC_PHIT0);
    /* Set port M as HOSC */
    CG_SetPortM(CG_PORTM_AS_HOSC);
    /* Enable high-speed oscillator */
    CG_SetFosc(CG_FOSC_OSC1, ENABLE);
    /* Set fosc source */
    CG_SetFoscSrc(CG_FOSC_OSC1);
    /* Enable low-speed oscillator */
    CG_SetFs(ENABLE);
    /* Set low power consumption mode sleep */
    CG_SetSTBYMode(CG_STBY_MODE_SLEEP);
    /* Set high-speed and low-speed oscillator to be enabled after releasing stop mode
*/
    CG_SetExitStopModeFosc(ENABLE);
    CG_SetExitStopModeFs(ENABLE);
    /* 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();
    /* Set system clock to fgear */
    CG_SetFsysSrc(CG_FSYS_SRC_FGEAR);
```

スタンバイ解除のための RMCRX 割り込みの設定

スタンバイ解除のために TMRB と RMC 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、INTRMCRX を有効にします。

```
    CG_ClearINTReq(CG_INT_SRC_RMC_RX);
    CG_SetSTBYReleaseINTSrc(CG_INT_SRC_RMC_RX,
                                    CG_INT_ACTIVE_STATE_RISING, ENABLE);
    NVIC_ClearPendingIRQ(INTRMCRX_IRQn);
    NVIC_EnableIRQ(INTRMCRX_IRQn);
```

SLEEP モード設定

SLEEP モードに入る設定を行います。まず CPU、RTC、I/O、RMC 以外のモジュールを無効にします。その後、システムクロックソースを fs に変更し、ウォームアップ時間を設定します。スタンバイ解除のために INTRMCRX を設定し、__WFI()命令を使用して SLEEP モードに入ります。

```
    /* Set CG module: Normal ->Sleep mode */
    /* Add code here to disable modules (except CPU, RTC, I/O and RMC) if it has been
enabled... */
    /* Then set system clock to fs */
```

```
    FunctionalState fs_st;
    fs_st = CG_GetFsState();
    if (fs_st == DISABLE) {
        CG_SetFs(ENABLE);
        CG_SetWarmUpTime(CG_WARM_UP_SRC_XT1, CG_WUODR_EXT);
        /* Start warm up */
        CG_StartWarmUp();
        /* Check whether warm up ends or not */
        while (CG_GetWarmUpState() == BUSY);
    } else {
        /* Do nothing */
    }

    CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC2, CG_WUODR_EXT);
    /* Set RMC wake up system */
    CG_SetSTBYReleaseINTSrc(CG_INT_SRC_RMC_RX,
                            CG_INT_ACTIVE_STATE_RISING, ENABLE);
    /* Enter sleep mode */
    __WFI();
}
```

マルチクロック回路の許可

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 DMAC

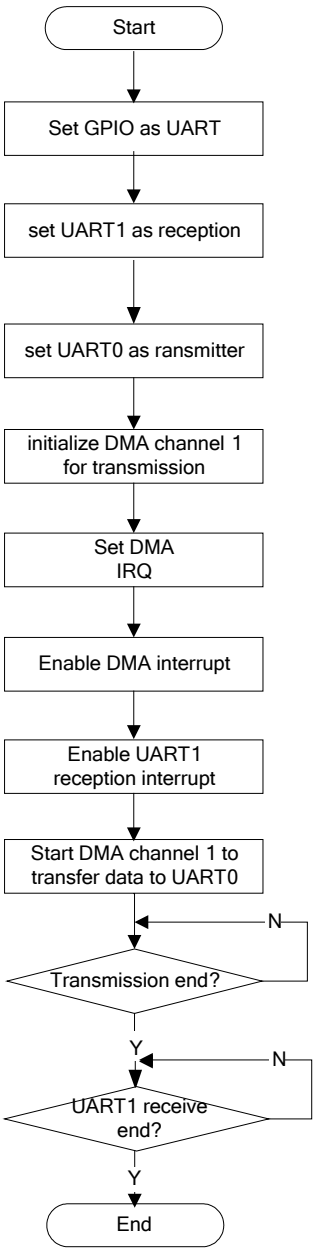
7-3-1 例: メモリから周辺回路

ペリフェラルドライバ(DMAC, GPIO, SIO/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. UART0/1 と DMAC の初期化を行います。
2. DMAC によるメモリ→UART0 転送を行います。

- **フローチャート**



- **サンプルプログラムのコードと説明**

ペリフェラルドライバ(DMAC)を使用してメモリ→UART0転送を行います。

まず UART0 を初期化後、UART0 動作を許可します。

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

データ転送のために DMAC ユニットのチャネル 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.TxPeriph = DMAC_SIO_0_RTX;
DMAC_InitStruct.TxINT = ENABLE;

DMAC_Init(myChannel, &DMAC_InitStruct);
```

DMA 割り込みを許可します。

```
NVIC_ClearPendingIRQ(INTDMACTC_IRQn);
NVIC_EnableIRQ(INTDMACTC_IRQn);
```

DMAC ユニットの許可、DMA チャネルの初期化を行います。その後、DMAC ユニットのチャネル 1 から転送を開始します。

```
DMAC_Enable();
DMAC_SetTxINTConfig(myChannel, DMAC_INT_TX_END, ENABLE);
DMAC_SetSWBurstReq(DMAC_SIO_0_RTX);
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 ユニット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 */
}
```

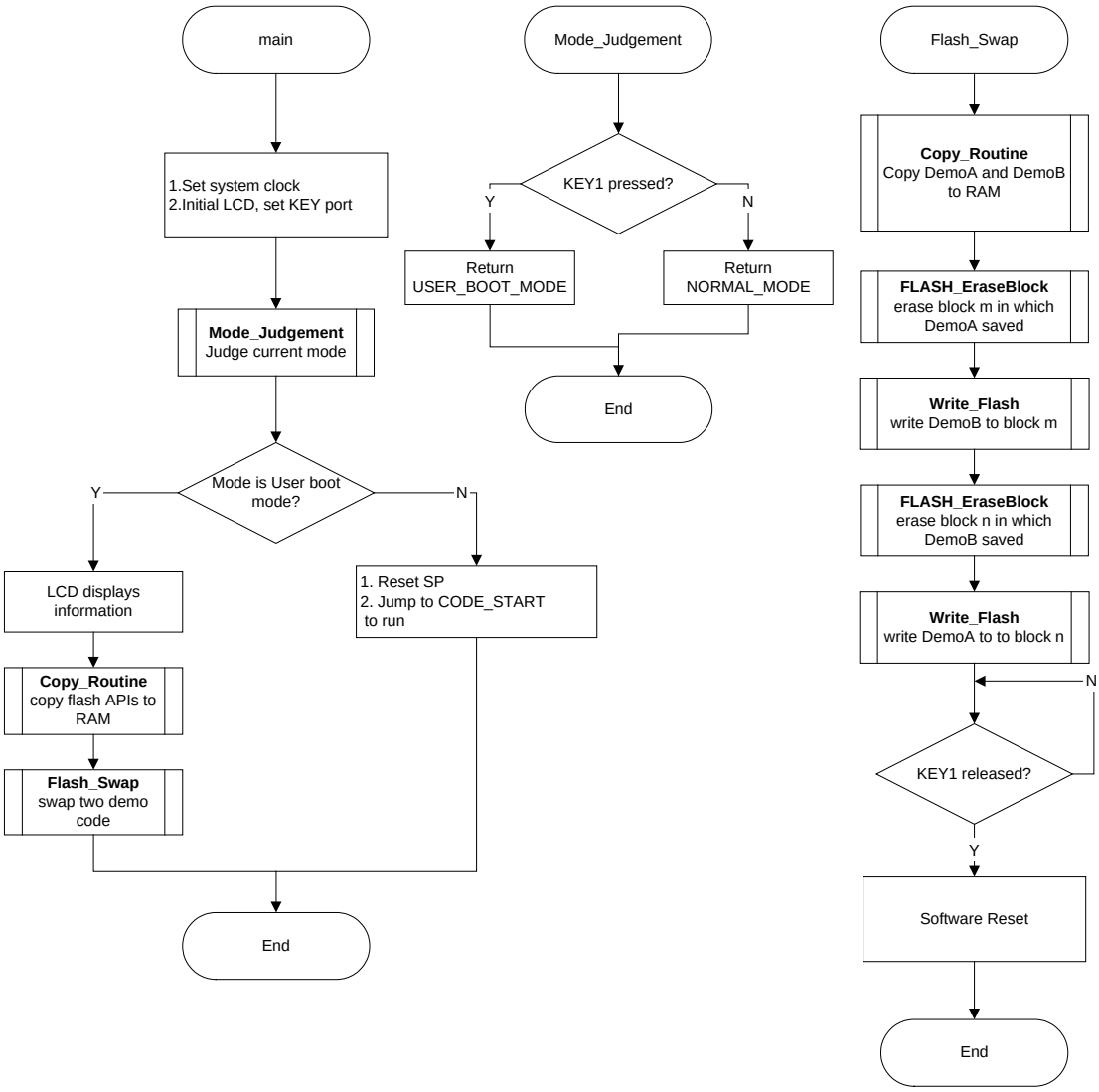
7-4 FLASH

ペリフェラルドライバ(Flash, GPIO, CG)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、ユーザーブートモード)
3. ユーザー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, SW を初期化します。リセット時に現在のモード判定を SW で行います。

```
CG_SetPLL(DISABLE);                              /* Disable PLL */
CG_SetFcSrc(CG_FC_SRC_FOSC);                     /* Select fosc */
GPIO_SetInput(GPIO_PD, GPIO_BIT_0);              /* set port D to input */
LEDInit();                                        /* LED initialization */
LedDisable(LED_CH1 | LED_CH2 | LED_CH4);         /* Turn off all LED channel */
```

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 (GPIO_ReadDataBit(GPIO_PD, GPIO_BIT_0) ==
            GPIO_BIT_VALUE_0) ? USER_BOOT_MODE : NORMAL_MODE;
}
```

SW がリセット時に ON でない場合、動作モードはノーマルモードになります。プログラムは SP を初期化し、アドレス“CODE_START”へジャンプします。

```
#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
      startup = CODE_START;
      startup();                      /* jump to code start address to run */
```

リセット時に SW が Low の場合、動作モードはユーザブートモードに移行し、LED に情報(‘0’)を表示します。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、プログラムはフラッシュ操作ルーチンをアドレス“FLASH_API_ROM”からアドレス“FLASH_API_RAM”へ ROM->RAM 転送し、LED に情報(‘1’)を表示します。

```
Status_Display(ch0);      /* LED status 1: enter user boot mode */
Status_Display(ch1);      /* LED status 2: RAM transferring */
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、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 メモリーから RAM にコピーされます。

Flash_Swap() 関数では、まずプログラム A とプログラム B を Flash ROM から RAM にコピーします。次に、Flash ROM の消去/書き込みを行うため、関数 FC_EraseBlock()と Write_Flash()をコールします。プログラム A とプログラム B は Flash ROM 内にて差し替え、LED に情報(‘2’)を表示します。

```
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 */

Status_Display(ch2);            /* LED status 2: swap the demo */

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

/* write B to block m */
if (FC_SUCCESS == Write_Flash(DEMO_A_FLASH, DEMO_B_RAM,
SIZE_DEMO_B)) {
      /* Do nothing */
} else {
      return ERROR;
}

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

```
}

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

プログラム A とプログラム B のスワップ動作が終了したことを示す情報('3')を LED に表示します。SW が OFF の場合、SCB->AIRCR レジスタにて自動的にソフトウェアリセットを行います。そして、再び NORMAL モードで動作します。プログラム A とプログラム B が入れ替わったため、アドレス "CODE_START"はプログラム B の開始アドレスとなり、プログラム B が実行されます。(LED5, 7, 9, 11 が点滅します)

```
Status_Display(ch3);                    /* LED status 3: complete and restart */

while (GPIO_ReadDataBit(GPIO_PD, GPIO_BIT_0) == GPIO_BIT_VALUE_0) {
            /* wait for KEY1 release */
}
reg_value = SCB->AIRCR;        /* software reset */
reg_value = (uint32_t) 0x05FA0001;
SCB->AIRCR = reg_value;
```

Flash メモリ動作関数 FC_EraseBlock() は自動的に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 "BlockAddr" で指定します。まず、この関数で引数 "BlockAddr" を確認します。次に、Flash ドライバ API FC_GetBlockProtectState()関数をコールし、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。それ以外の場合は、自動的ブロック削除命令を送り、ブロックを削除します。

```
      *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、削除が終了すると、Flash ドライバ API の FC_GetBusyState()関数をコールし、モニタリングを行います。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;
            *addr1 = FC_RESET_CMD;          /* Reset FLASH */
             break;
      } else {
             /* Do nothing */
      }
}
```

Write_Flash()関数は FC_WritePage()関数をコールし、自動的に 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 FC_EraseBlock ()関数と同じです。

```
*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 < 64U; i++) {             /* bus cycle 4~67 */
    *addr3 = *source;
    source++;
}
```

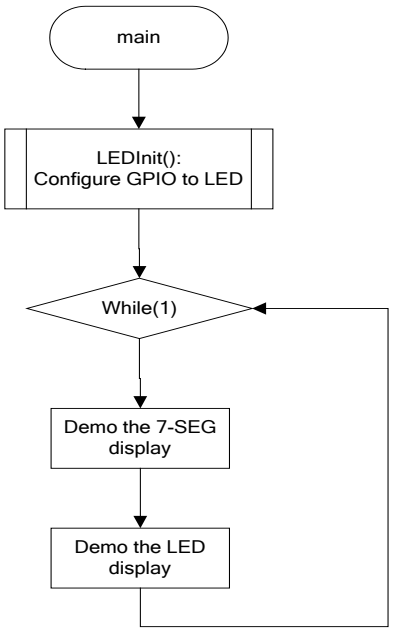
7-5 GPIO

ペリフェラルドライバ 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. GPIO の初期化
2. GPIO へのデータ書き込み

- **フローチャート:**



- **サンプルプログラムのコードと説明:**

まず GPIO を LED に設定します。

```
GPIO_SetOutput(LED_CH_PORT,0XF0U);
```

```
    reg = GPIO_ReadData(LED_CH_PORT);
    reg &= 0x0FU;
    GPIO_WriteData(LED_CH_PORT,reg);

    GPIO_SetOutput(LED_DATA_PORT,0XFFU);
    GPIO_WriteData(LED_DATA_PORT,0XFFU);
```

while(;;)ループの中で、LED プログラムを実行し、LED ON や LED OFF を制御します。

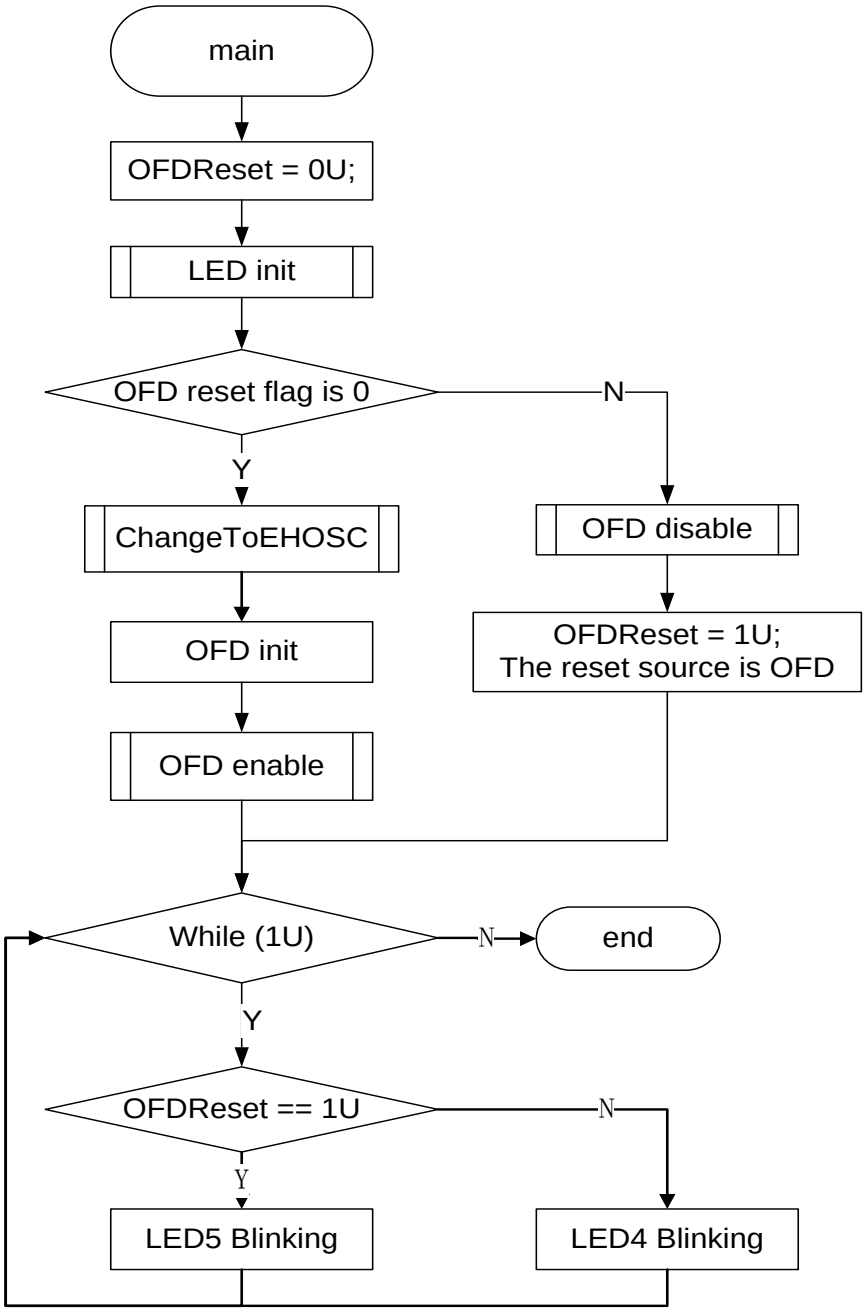
7-6 OFD

ペリフェラルドライバ(OFD, CG, 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. 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。

```
Init_Led();
```

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 */
}
```

このフラグが設定されていない場合、外部発振器の許可と検知周波数を設定して OFD を初期化します。

以下は外部発振器を許可する処理です。

```
void ChangeToEHOSC(void)
{
    /* Use the external oscillation */
    TSB_CG->OSCCR &= 0x000FFFFFUL;
    TSB_CG->OSCCR |= 0xFFF00000UL; /*Set the warm up time WUODR[13:2]*/
    TSB_CG_OSCCR_HOSCON = 1U; /*Set 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 */
}
```

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);
```

設定検知周波数を設定します。

```
    OFD_SetDetectionFrequency(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_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(OSC が乱れる、DEMO2 が定義されている)と OFD は周波数検知リセットを発生します。これにより OFD のリセットフラグが設定されます。
このフラグを検知すると、デフォルト内部発振器で MCU が実行を始め、OFD は無効となり、OFD リセットフラグに'1'がセットされます。

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

LED4 と LED5 はシステムクロックの状態を表示します。外部発振器にて発振している場合は LED4 を点滅し、内部発振器にて発振している場合は、LED5 を点滅します。

```
    while (1U) {
        if (OFDReset == 1U) {
            LED_On(LED5);
            Delay();
            LED_Off(LED5);
            Delay();
        } else {
            LED_On(LED4);
            Delay();
            LED_Off(LED4);
            Delay();
        }
    }
```

7-7 RMC

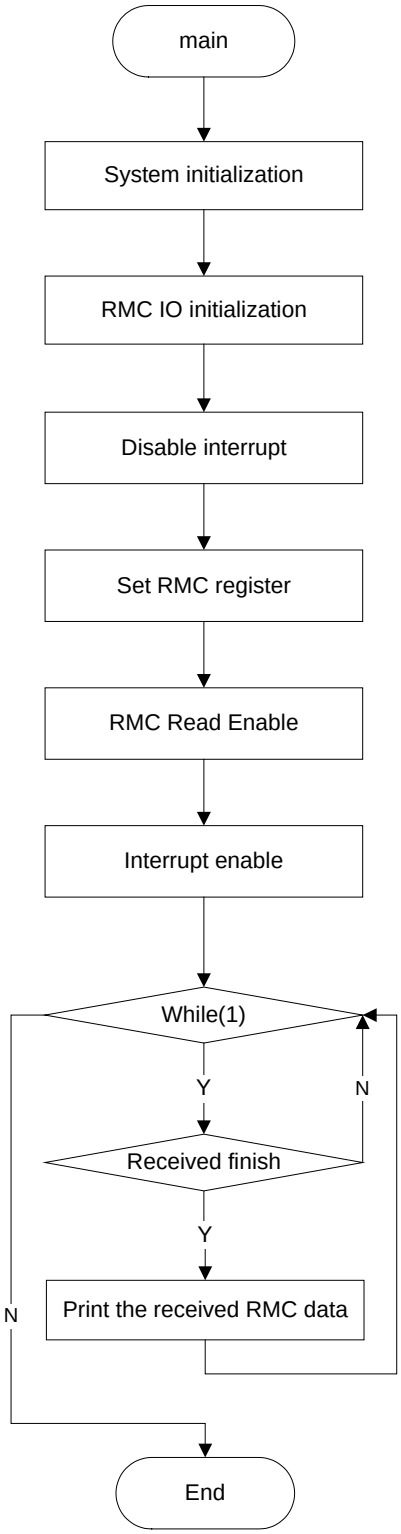
7-7-1 例: RMC 受信

ペリフェラルドライバ(RMC, UART, GPIO)を使用した簡単なサンプルソフトです。

以下の例が含まれます。

1. RMC 用 IO の初期化
2. RMC データの受信

- **フローチャート**



- **サンプルプログラムのコードと説明**

まず main()にて myRMC structure 構造体を作成し、データフィールドにデータを入力します。

```
RMC_InitTypeDef myRMC;
```

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

次に、RMC チャネル 0 を初期化し、有効にします。

```
RMC_Enable(TSB_RMC0);
```

IDLE モード中は RMC が無効になるように設定します。

```
RMC_SetIdleMode(TSB_RMC0, DISABLE);
```

構造体には RMC の基本設定が含まれます。

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

RMC チャネル 0 の受信を有効にする。

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

割り込み INTRMCRX_IRQHandler()にて RMC チャネル 0 の割り込み要因を取得します。

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

RMC チャネル 0 のリーダー検出結果を取得します。

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

RMC チャネル 0 の受信データを取得します。次に、RMC チャネル 0 を初期化し、有効にします。

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

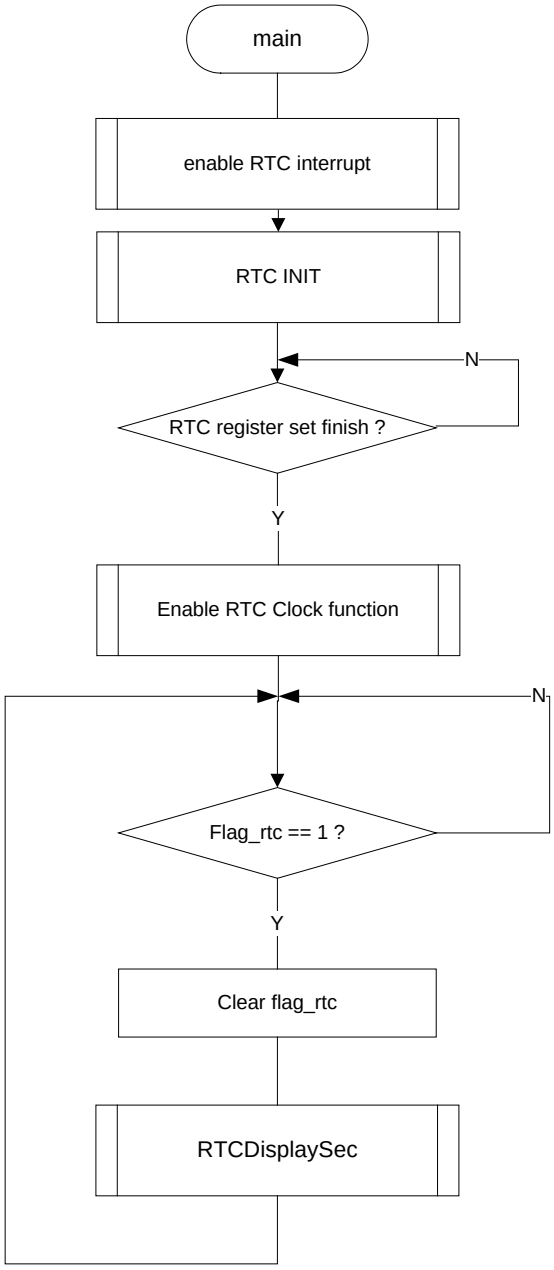
7-8 RTC

ペリフェラルドライバ(RTC, CG, GPIO)を使用した簡単なサンプルソフトです。

以下の例が含まれます。

1. RTC の初期化
2. RTC 秒の取得

- **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、RTC 割り込みによる SLEEP モードの解除設定を行います。

```
CG_SetSTBYReleaseINTSrc(CG_INT_SRC_RTC,
                        CG_INT_ACTIVE_STATE_FALLING, ENABLE);
```

RTC の初期化で、RTC_DateTypeDef と RTC_TimeTypeDef 構造を作成し、全データ項目を設定します。ここでは 2010/10/22 12:50:55, 24 時間表示フォーマットを初期設定としています。

```
Date_Struct.LeapYear = RTC_LEAP_YEAR_2;
Date_Struct.Year = (uint8_t) 10U;
Date_Struct.Month = (uint8_t) 10U;
Date_Struct.Date = (uint8_t) 22U;
Date_Struct.Day = RTC_FRI;
```

```
Time_Struct.HourMode = RTC_24_HOUR_MODE;
Time_Struct.Hour = (uint8_t) 12U;
Time_Struct.Min = (uint8_t) 50U;
Time_Struct.Sec = (uint8_t) 55U;
```

クロックとアラーム機能を禁止します。

```
RTC_DisableClock();
RTC_DisableAlarm();
```

RTC 秒カウンタのリセット、1Hz 割り込みの許可、RTCINT の許可を行います。

```
RTC_ResetClockSec();
RTC_SetAlarmOutput(RTC_PULSE_1_HZ);
RTC_SetRTCINT(ENABLE);
```

RTC の時間と日付の値を設定します。

```
RTC_SetTimeValue(&Time_Struct);
RTC_SetDateValue(&Date_Struct);
```

上記項目を設定後、RTC レジスタ設定の終了を待ち、RTC 時計機能を許可します。その後、RTC クロック機能を許可します。

```
NVIC_EnableIRQ(INTRTC_IRQn);

/* waiting for RTC register set finish */
while (fRTC_1HZ_INT != 1U) {
      /* Do nothing */
}
fRTC_1HZ_INT = 0U;

/* Enable RTC Clock function */
RTC_EnableClock();
```

RTC 割り込みが 1 秒ごとに発生するよう設定を行い、その後、RTC 割り込み要求のクリアを行います。

```
fRTC_1HZ_INT = 1U;
CG_ClearINTReq(CG_INT_SRC_RTC);
```

RTC 割り込み発生後、第 2 番目の値がバイナリで LED ディスプレイに送信されます。
下記に、第 2 番目の値の取得方法と表示方法を表します。

```
Sec = RTC_GetSec();
tmp = Sec % 10U;
SegArrayDisplay(ledchar[tmp])
```

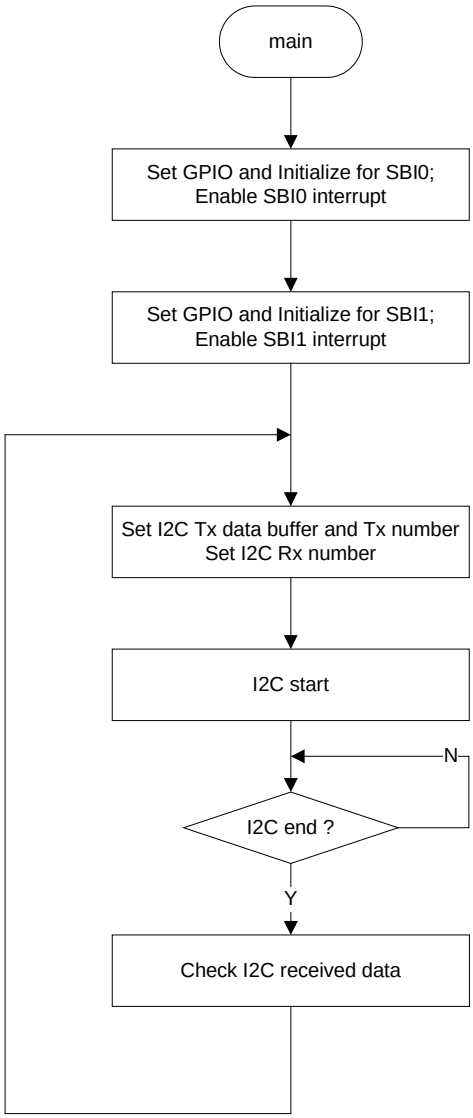
7-9 SBI

ペリフェラルドライバ(SBI, GPIO)を使用した簡単なサンプルソフトです。

以下の例が含まれます。

1. SBI 設定、I2C 初期化
2. I2C マスターによるデータプロセスの送信
3. I2C スレーブによるデータプロセスの受信

- **フローチャート**



- **サンプルプログラムのコードと説明**

まず、SBI0 と SBI1 の GPIO を I2C モードに設定します。

```
SBI0_IO_Configuration();
SBI1_IO_Configuration();
```

次に SBI0 チャネルの初期化と INTI2C をイネーブルにします。

```
myI2C.I2CSelfAddr = SELF_ADDR;
myI2C.I2CDataLen = SBI_I2C_DATA_LEN_8;
myI2C.I2CACKState = ENABLE;
myI2C.I2CClkDiv = SBI_I2C_CLK_DIV_328;
SBI_Enable(TSB_SBI0);
SBI_SWReset(TSB_SBI0);
SBI_InitI2C(TSB_SBI0, &myI2C);
NVIC_EnableIRQ(INTSBI0_IRQn);
```

次に SB1 チャネルの初期化と INTSBI1 をイネーブルにします。

```
    myI2C.I2CSelfAddr = SLAVE_ADDR;
    myI2C.I2CDataLen = SBI_I2C_DATA_LEN_8;
    myI2C.I2CACKState = ENABLE;
    myI2C.I2CClkDiv = SBI_I2C_CLK_DIV_328;
    SBI_Enable(TSB_SBI1);
    SBI_SWReset(TSB_SBI1);
    SBI_InitI2C(TSB_SBI1, &myI2C);
    NVIC_EnableIRQ(INTSBI1_IRQn);
```

上記設定を行った後、I2C 送信開始します。
I2C 受信バッファをクリアし、SBI TX バッファーとバッファ長を設定します。その後 RX バッファーをクリアします。

```
        /* Initialize TRx buffer and Tx length */
    case MODE_SBI_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;
        for (gICnt = 0U; gICnt < 8U; gICnt++) {
            gI2CRxData[gICnt] = 0U;
        }
        gSBIMode = MODE_SBI_I2C_START;
        break;
```

I2C バスが空いているかどうか、“SLAVE_ADDR” データを SBI_SetSendData()に設定します。そして、送信方向を“SBI_I2C_SEND”から SBI データバッファへ設定します。その後、SBI_GenerateI2CStart(TSB_SBI0) を使用して、I2C 動作を開始します。

```
        /* Check I2C bus state and start TRx */
    case MODE_SBI_I2C_START:
        i2c_state = SBI_GetI2CState(TSB_SBI0);
        if (!i2c_state.Bit.BusState) {
            SBI_SetSendData(TSB_SBI0, SLAVE_ADDR | SBI_I2C_SEND);
            SBI_GenerateI2CStart(TSB_SBI0);
            gSBIMode = MODE_SBI_I2C_TRX;
        } else {
            /* Do nothing */
        }
        break;
```

データ転送は INTSBI0 にて処理し、データ受信は INTSBI1 にて処理します。

INTSBI0 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、SBI_SetSendData() で次のデータを送信し、I2C プロセス終了時には、SBI_GenerateI2CStop() で I2C を停止します。

```
void INTSBI0_IRQHandler(void)
{
    TSB_SBI_TypeDef *SBIx;
    SBI_I2CState sbi_sr;

    SBIx = TSB_SBI0;
    sbi_sr = SBI_GetI2CState(SBIx);
```

```
    if (sbi_sr.Bit.MasterSlave) {           /* Master mode */
        if (sbi_sr.Bit.TRx) {     /* Tx mode */
            if (sbi_sr.Bit.LastRxBit) { /* LRB=1: the receiver requires no further data. */
                SBI_GenerateI2CStop(SBIx);
            } else {                /* LRB=0: the receiver requires further data. */
                if (gI2CWCnt <= gI2CTxDataLen) {
                    SBI_SetSendData(SBIx, gI2CTxData[gI2CWCnt]);
                                                                    /* Send next data */
                    gI2CWCnt++;
                } else {            /* I2C data send finished. */
                    SBI_GenerateI2CStop(SBIx);    /* Stop I2C */
                }
            }
        } else {                    /* Rx Mode */
            /* Do nothing */
        }
    } else {                        /* Slave mode */
        /* Do nothing */
    }
}
```

INTSBI1 ハンドラで、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決定します。SBI バッファの受信データ読み出しは SBI_GetReceiveData() 関数を用いて行います。I2C 停止状態はマスタによって制御します。

```
void INTSBI1_IRQHandler(void)
{
    uint32_t tmp = 0U;
    TSB_SBI_TypeDef *SBIy;
    SBI_I2CState sbi1_sr;

    SBIy = TSB_SBI1;
    sbi1_sr = SBI_GetI2CState(SBIy);

    if (!sbi1_sr.Bit.MasterSlave) {         /* Slave mode */
        if (!sbi1_sr.Bit.TRx) { /* Rx Mode */
            if (sbi1_sr.Bit.SlaveAddrMatch) {
                /* First read is dummy read for Slave address recognize */
                tmp = SBI_GetReceiveData(SBIy);
                gI2CRCnt = 0U;
            } else {
                /* Read I2C received data and save to I2C_RxData buffer */
                tmp = SBI_GetReceiveData(SBIy);
                gI2CRxData[gI2CRCnt] = tmp;
                gI2CRCnt++;
            }
        } else {                    /* Tx Mode */
            /* Do nothing */
        }
    } else {                        /* Master mode */
        /* Do nothing */
    }
}
```

7-10 SIO/UART

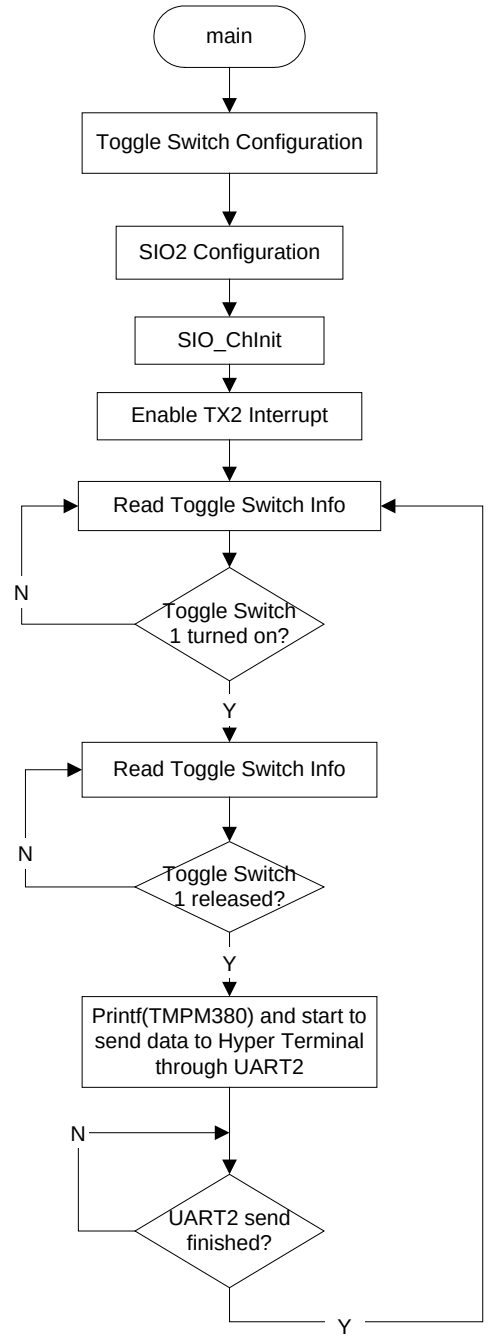
7-10-1 例: リターゲット

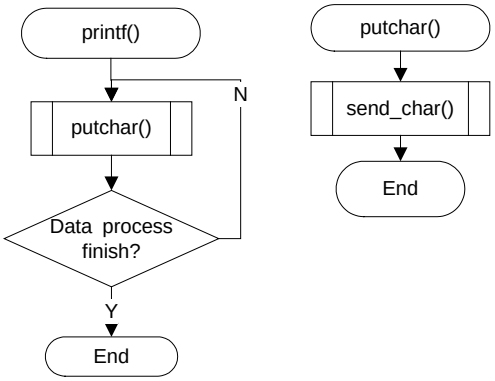
ペリフェラルドライバ(UART、GPIO)を使用した簡単なサンプルプログラムです。

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

- **フローチャート**


main
Toggle Switch Configuration
SIO2 Configuration
SIO_ChInit
Enable TX2 Interrupt
Read Toggle Switch Info
N
Toggle Switch 1 turned on?
Y
Read Toggle Switch Info
N
Toggle Switch 1 released?
Y
Printf(TMPM380) and start to send data to Hyper Terminal through UART2
N
UART2 send finished?
Y


printf()
N
putchar()
Data process finish?
Y
End
putchar()
send_char()
End

- **サンプルプログラムのコードと説明**

まず、GPIO を設定し、トグルスイッチを初期化します。

```
    GPIO_SetPullUp(TSW_PORT, GPIO_BIT_0, ENABLE);
    GPIO_SetPullUp(TSW_PORT, GPIO_BIT_1, ENABLE);
    GPIO_SetPullUp(TSW_PORT, GPIO_BIT_2, ENABLE);
    GPIO_SetInputEnableReg(TSW_PORT, GPIO_BIT_0, ENABLE);
    GPIO_SetInputEnableReg(TSW_PORT, GPIO_BIT_1, ENABLE);
    GPIO_SetInputEnableReg(TSW_PORT, GPIO_BIT_2, ENABLE);
```

次に GPIO を設定し、UART を初期化します。

```
    GPIO_SetOutputEnableReg(GPIO_PD, GPIO_BIT_5, ENABLE);
    GPIO_EnableFuncReg(GPIO_PD, GPIO_FUNC_REG_1, GPIO_BIT_5);
    GPIO_EnableFuncReg(GPIO_PD, GPIO_FUNC_REG_1, GPIO_BIT_6);
    GPIO_SetInputEnableReg(GPIO_PD, GPIO_BIT_6, ENABLE);
    GPIO_SetPullUp(GPIO_PD, GPIO_BIT_6, ENABLE);
    GPIO_SetPullUp(GPIO_PD, GPIO_BIT_5, 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 = 115200;     /* 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 動作を許可し、UART の初期化を行います。

```
    UART_Enable(UART2);
    UART_Init(UART2, &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(INTTX2_IRQn);
```

その後、トグルスイッチの状態を取得します。

```
    TSW_info = GPIO_ReadData(TSW_PORT) & TSWBITS;
```

トグルスイッチが ON の場合は、OFF になるまで待ちます。

```
    if (TSW_info == TSW1) {
    do {
        wait_TSW1 = 0U;
        TSW_info = GPIO_ReadData(TSW_PORT) & TSWBITS;
        } while (TSW_info != TSWRELEASE);                /* wait for TSW1 released */
    }
```

その後データ送信を開始します。ここでの TxBuffer は文字列です。

```
    printf("%s\r\n", TxBuffer);         /* SIO2 send data */
```

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

      return ch;
}
```

データフローの残りのプロセスは UART2 送信割り込みルーチンの ISR にて終了します。

UART2 の送信割り込みルーチン:

```
      void INTTX2_IRQHandler(void)
      {
            if (gSIORdIndex < gSIOWrIndex) {       /* buffer is not empty */
                  UART_SetTxData(UART2, gSIOTxBuffer[gSIORdIndex++]);/* send data */
                  fSIO_INT = SET;                /* SIO0 INT is enable */
            } else {
                  /* disable SIO2 INT */
                  fSIO_INT = CLEAR;
                  NVIC_DisableIRQ(INTTX2_IRQn);
                  fSIOTxOK = YES;
            }
            if (gSIORdIndex >= gSIOWrIndex) {      /* reset buffer index */
                  gSIOWrIndex = CLEAR;
                  gSIORdIndex = CLEAR;
            } else {
                  /* do nothing */
            }
      }
```

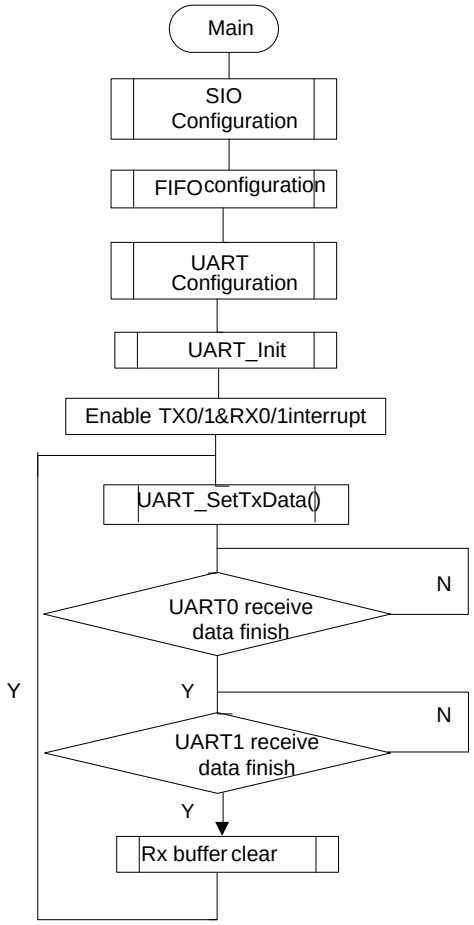
7-10-2　例: UART FIFO

UART と GPIO を使用した簡単な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 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 構造体を用意し、すべてのメンバを設定します。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 のドライバを使用して、UART0/1 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(UART1);
    UART_Init(UART1, &myUART);
```

FIFO の設定を行います。

```
    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 UART0/1 の各割り込みを許可します。

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

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

最後に UART0/1 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。
UART0 の送信割り込み処理ルーチン:

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

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

UART1 の送信割り込み処理ルーチン:

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

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

UART0 の受信割り込み処理ルーチン:

```
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 の受信割り込み処理ルーチン:

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

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

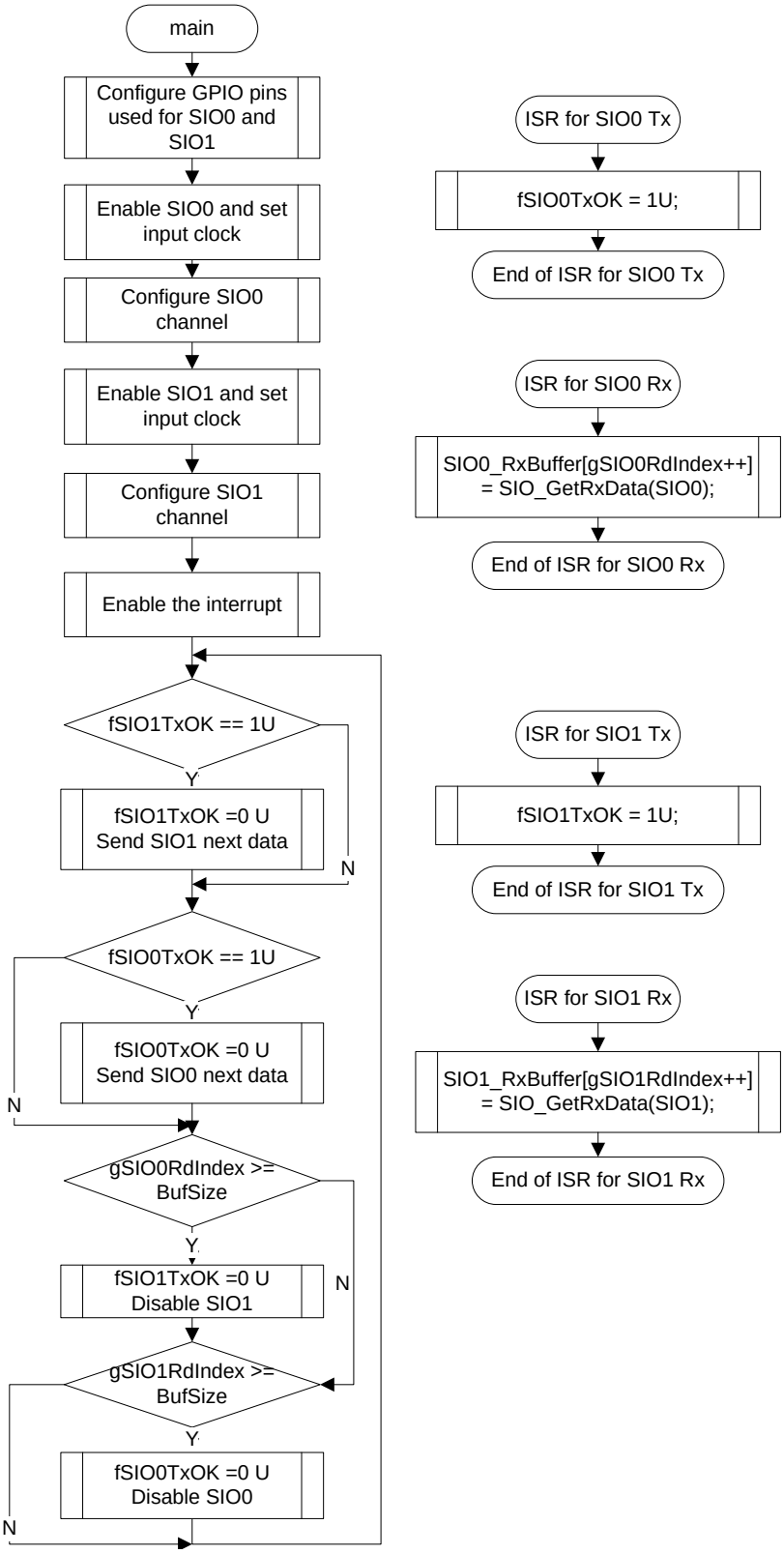
7-10-3　例: SIO

ペリフェラルドライバ(SIO)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. マスタ SIO0⇔スレーブ SIO0 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 ペリフェラルドライバを使い、GPIO を SIO に設定します。その後、SIO 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);
```

SIO の送受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、転送設定を行います。

```
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);
}
```

7-11 SSP

7-11-1 例: SSP0 から SSP1 への DMAC 転送

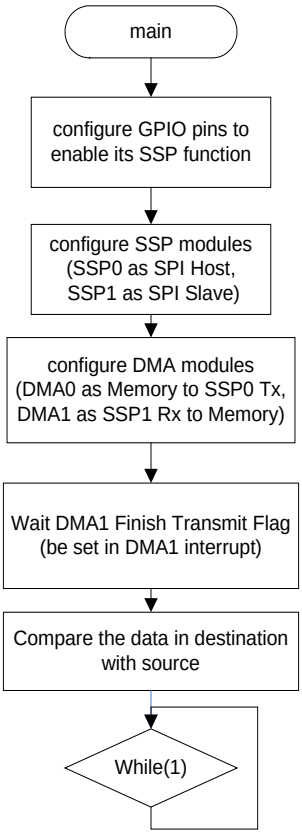
ペリフェラルドライバ(SSP, DMAC, GPIO)を使用したサンプルソフトです。

以下の例を含みます。

1. SSP0 を SPI ホストとする設定と、SPI スレーブとする設定
2. DMAC0 をメモリから周辺回路(SSP0 TX)とする設定と DMAC1 を周辺回路(SSP1 RX)からメモリとする設定

3. データ転送フローの確認: メモリ→SSP0 TX→SSP1 RX→メモリ

- **フローチャート**



- **サンプルプログラムのコードと説明**

以下は上記フローチャートの main.c です。

```
int main(void)
{
	GPIO_SetSSP();
	initSSP();
	InitDMA();

	/* Wait the end of transmission */
	while (TxEndFlag != DONE) {
		/* Do nothing */
	}

	/* now DMA is finished, Set a Break Point here, */
	/* after function Buffercompare() is called, result == SAME */
	result = Buffercompare( SRC_Buffer, DST_Buffer, BUFFER_SIZE);

	while(1) {
		/* do nothing */
	}
}
```

上記関数 GPIO_SetSSP(), initSSP(), InitDMA()の詳細は、コードのコメントを参照してください。

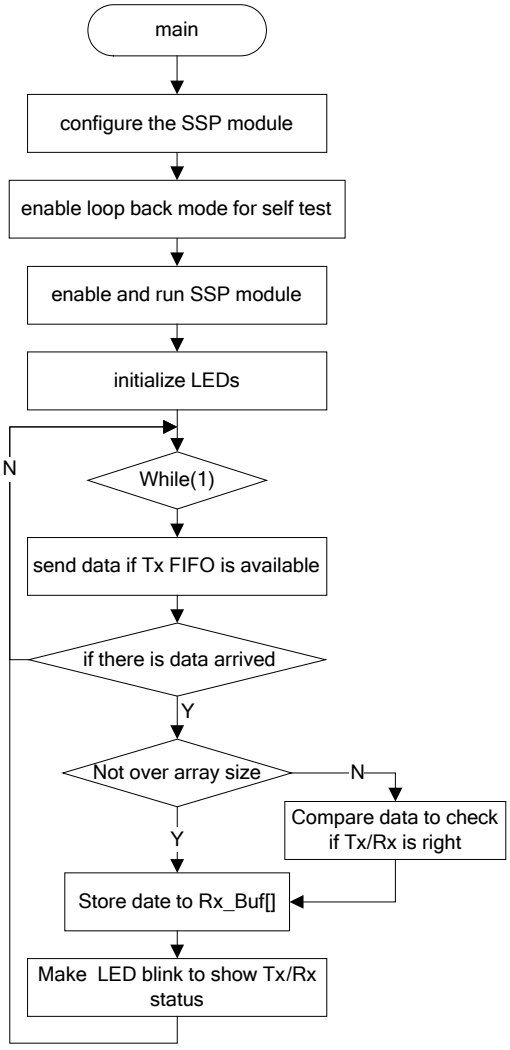
7-11-2　例: SSP0 セルフループバック

ペリフェラルドライバ(SSP, GPIO)を使用したサンプルソフトです。

以下の例を含みます。

1. SSP0 の設定と初期化
2. 自身で送受信を行うループバックモードの許可

- **フローチャート**



- **サンプルプログラムのコードと説明**

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

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

```
    uint32_t cntRx = 0U;

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

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

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

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

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

    /* initialize LEDs on M380-SK board before display something */
    LEDInit();

    while (1) {

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

        /* check if there is data arrived */
        fifoState = SSP_GetFIFOState(TSB_SSP0, SSP_RX);
        if ((fifoState == SSP_FIFO_FULL) || (fifoState == SSP_FIFO_NORMAL)) {
            receive = SSP_GetRxData(TSB_SSP0);
            if (cntRx < MAX_BUFSIZE) {
                Rx_Buf[cntRx] = receive;
                cntRx++;
            } else {
                /* Place a break point here to check if receive data is right.    */
```

```
                    /* Success Criteria:                                        */
                    /* Every data transmited from Tx_Buf is received in Rx_Buf. */
                    /* When the line "#define BITRATE_MIN" is commented, the SSP is
run in maxium */
                    /* bit rate, so we can find there is enough time to transmit date from 1
to */
                    /* MAX_BUFSIZE one by one. but if we uncomment that line, SSP is
run in */
                    /* minimum bit rate, we will find that receive data can't catch "datTx++",
*/
                    /* in this so slow bit rate, when the Tx FIFO is available, the cntTx */
                    /* has been increased so much.                              */
                    __NOP();
                    result = Buffercompare( Tx_Buf, Rx_Buf, MAX_BUFSIZE);
                }

            } else {
                /* do nothing */
            }

            DisplayLED(receive);
        }
}
```

7-12 TMRB

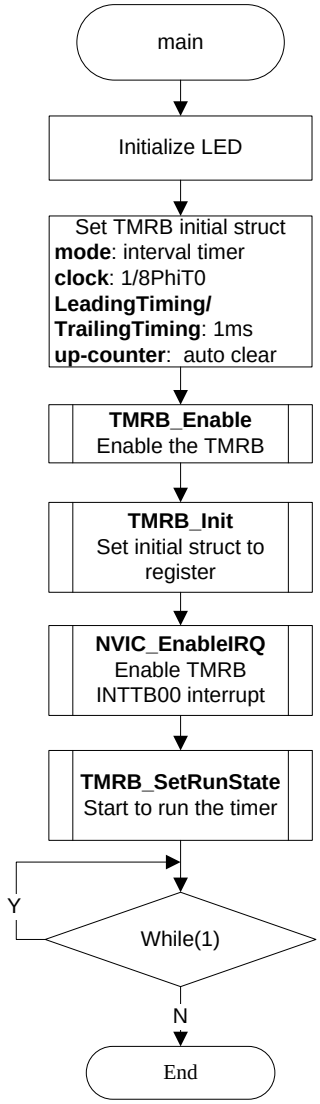
7-12-1 例: 汎用タイマ

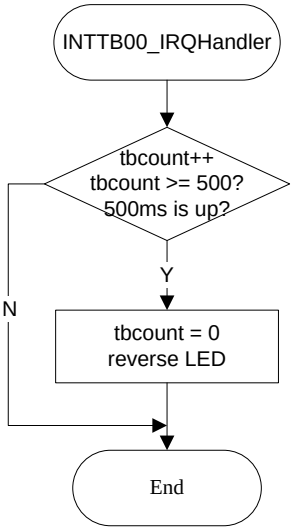
ペリフェラルドライバ(TMRB, GPIO)を使用した簡単なサンプルプログラムです。

以下の例が含まれます:

1. TMRB0 の初期化
2. 1ms の汎用タイマ

- **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 を点灯します。

```
LEDInit();                                          /* LED initialize */
LedDisable(LED_CH1 | LED_CH2 | LED_CH3);
LedOn(LED11);                                       /* Turn on LED11 */
```

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、0x1388 です。(φT0=fsys=10MHz * PLL* = 40MHz, ftmrb = 1/8φT0 = 5MHz, Ttmrb = 0.2us, 1ms/0.2us = 5000 = 0x1388 (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。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(INTTB0_IRQn);                    /* enable INTTB0 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 */
}
```

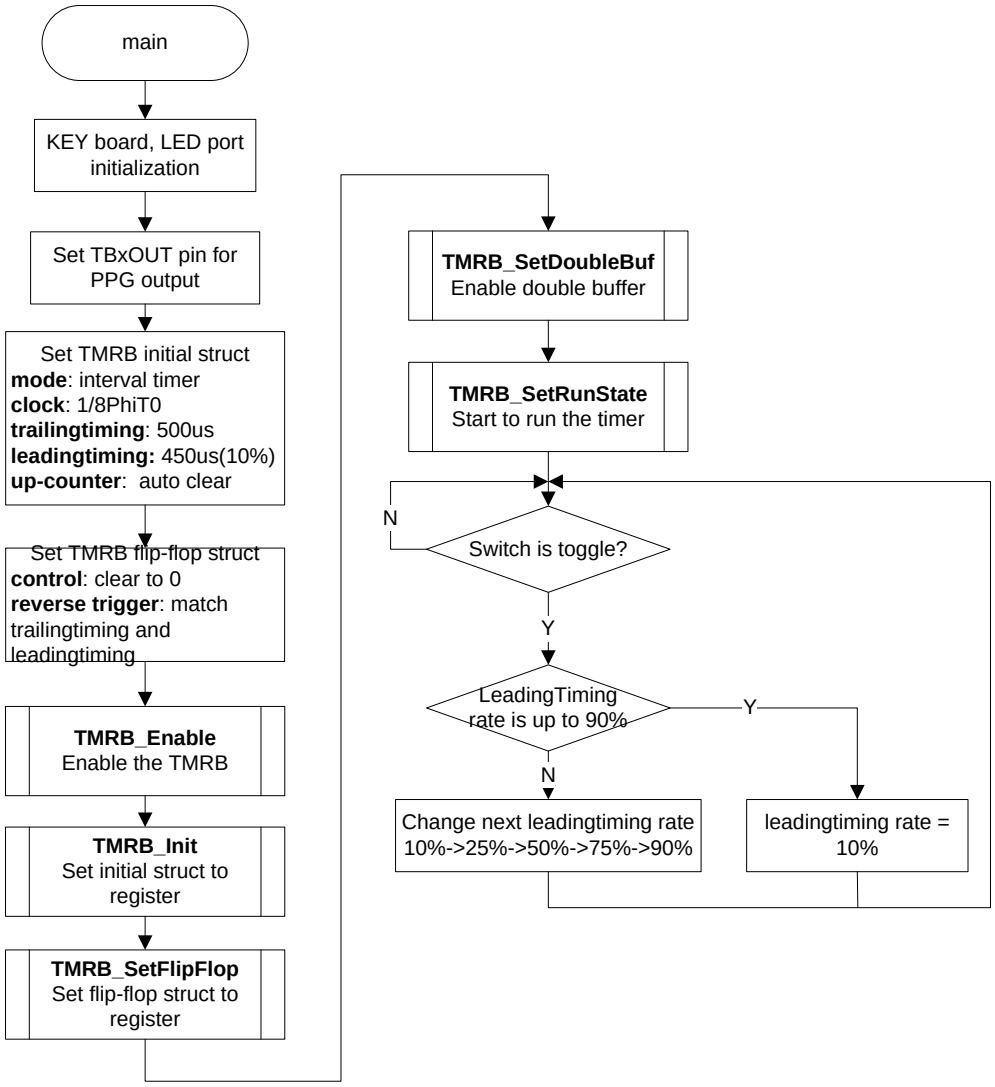
7-12-2 例: PPG 波形出力

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

以下の例が含まれます:

1. TMRB7 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**

まず、LED とスイッチの初期化を行い、PPG 出力用に PF1 を TB7OUT に設定します。

```
    GPIO_SetInput(KEYPORT, GPIO_BIT_0);                    /* set KEY port to input */

    LEDInit();                                                          /* LED initialization */
    LedDisable(LED_CH1 | LED_CH2 | LED_CH3 | LED_CH4);   /* Turn off all LED */

    /* Set PF1 as TB7OUT for PPG output */
    GPIO_SetOutput(GPIO_PF, GPIO_BIT_1);
    GPIO_EnableFuncReg(GPIO_PF, GPIO_FUNC_REG_1, GPIO_BIT_1);
    GPIO_SetPullUp(GPIO_PF, GPIO_BIT_1, ENABLE);           /* Enable pull up */
```

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、TMRB6TIME マクロを定義してあります。このマクロは 0x09C4 です。(φT0 = fsys = fc = 10MHz * PLL * = 40MHz, ftmrb = 1/8 φT0 = 5MHz, Ttmrb = 0.2us, 500us/0.2us = 2500 = 0x09C4 (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.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_TB7);
    TMRB_Init(TSB_TB7, &m_tmrb);
    TMRB_SetFlipFlop(TSB_TB7, &PPGFFInital);
    /* enable double buffer */
    TMRB_SetDoubleBuf(TSB_TB7, ENABLE, TMRB_WRITE_REG_SEPARATE);
    TMRB_SetRunState(TSB_TB7, TMRB_RUN);
```

スイッチ状態の変化を待ち、現在のデューティ状態を 7 セグメント LED に表示します。

```
    do {                            /* wait if switch is Low */
            keyvalue = GPIO_ReadDataBit(KEYPORT, GPIO_BIT_0);
            SegDisplay(leadingtiming_num[Rate]); /* display current leadingtiming */
       } while (GPIO_BIT_VALUE_0 == keyvalue);
```

スイッチ状態が変化すると、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_TB7,  LeadingTiming[Rate]);       /*  change
leading timing rate */
```

デューティの算出方法:
TrailingTiming = 500us, ftmrb = 1/8 fphiT0 = 5MHz, Ttmrb = 0.2us (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)

LeadingTiming = 10%: High 幅は 500*10% = 50us, Low 幅は 500-50 = 450us, カウンタ値 = 450us/Ttmrb = 0x8CAU

LeadingTiming = 25%: High 幅は 500*25% = 125us, Low 幅は 500-125 = 375us, カウンタ値= 375us/Ttmrb = 0x753U

LeadingTiming = 50%: High 幅は 500*50% = 250us, Low 幅は 500-250 = 250us, カウンタ値= 250us/Ttmrb = 0x4E2

LeadingTiming = 75%: High 幅は 500*75% = 375us, Low 幅は 500-375 = 125us, カウンタ値 = 125us/Ttmrb = 0x271

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

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] = { 0x8CAU, 0x753U, 0x4E2U, 0x271U, 0xFAU };
/* leading timing: 10%, 25%, 50%, 75%, 90% */
```

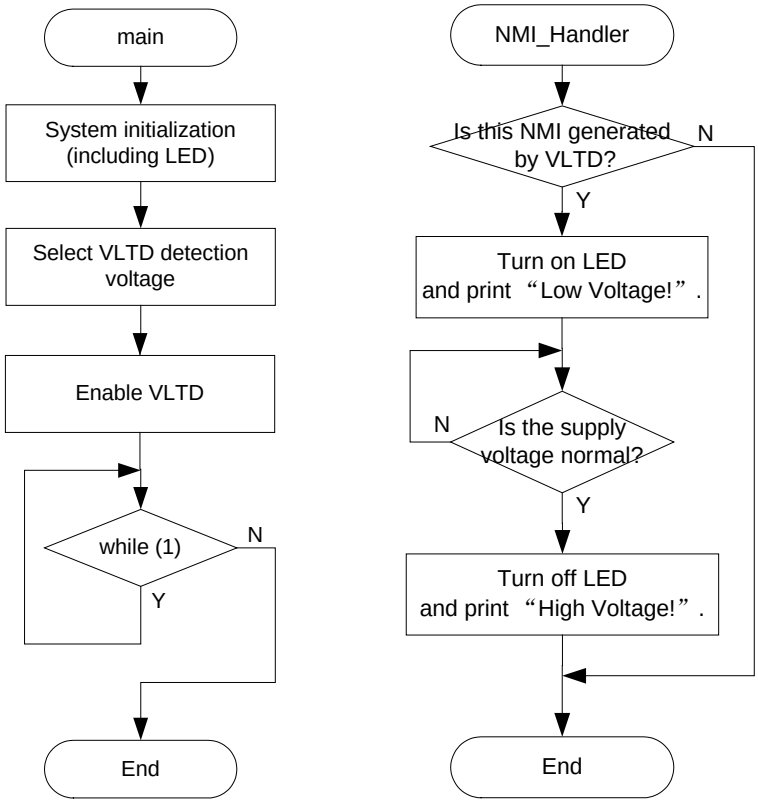
7-13 VLTD

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

この例では以下を行います。

1. VLTD の初期化。
2. VLTD 状態の初期化。

- **フローチャート**



- **サンプルプログラムのコードと説明**

最初にシステムを初期化します。不用意な WDT 割割り込みを避けるため WDT を無効にします。

```
WDT_Disable ();
LEDInit();
```

検出電圧の設定と VLTD 動作を許可します。

```
VLTD_SetVoltage(VLTD_DETECT_VOLTAGE_46);
VLTD_Enable ();
```

その後、while(1)ループさせます。
VLTD が電源電圧の低下を検出し NMI の発生を検出すると、LED を点灯し、ターミナルウィンドウにメッセージを表示します。

```
if (CG_GetNMIFlag().Bit.VoltageDetection == 1U) {
{
        LedOn(LED4);
#ifdef DEBUG
        printf("Low voltage!\n");
#endif
```

その後、電源電圧が通常に戻ると、LED を消灯し、ターミナルウィンドウにメッセージを表示します。

```
        while (VLTD_GetStatus() == 1U)
        {
            __NOP();
        }
        LedOff(LED3);
#ifdef DEBUG
        printf("High voltage!\n");
#endif
}
```

最後に NMI_Handler()関数に戻ります。

7-14 WDT

ペリフェラルドライバ (WDT, GPIO) のプログラムサンプルです。

この例では以下を行います。

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

- **サンプルプログラムのコードと説明**

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

```
WDT_InitTypeDef WDT_InitStruct;
```

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

WDT を初期化し、その後 WDT を有効にします。

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

DEMO1 では、NMI 割り込みの発生を待ちます。

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

DEMO1 では、NMI 割り込み発生時に WDT を禁止にし、LED1 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();
```

7-15 ENC

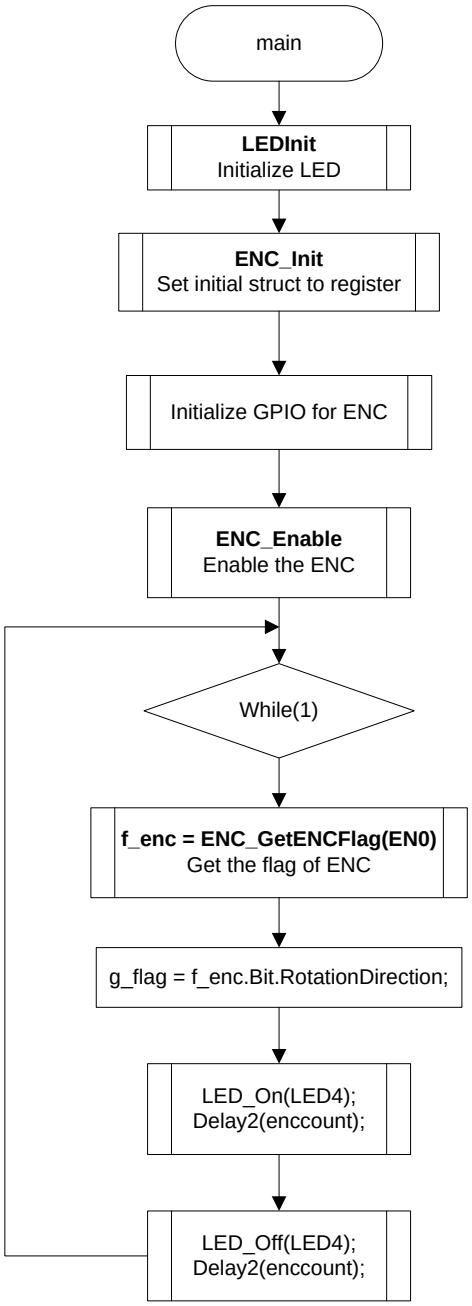
7-15-1 例: 回転検出

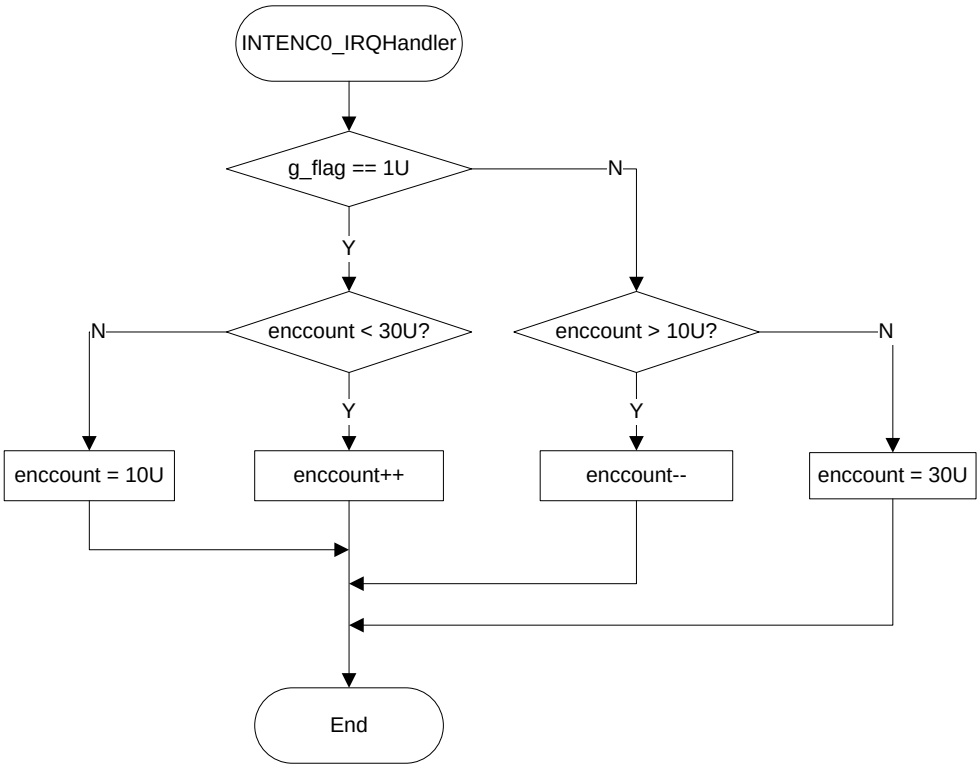
TX03 ペリフェラルドライバ(ENC, GPIO)を用いたサンプルプログラムです。

以下の例が含まれます:

1. ENC0 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
LEDInit
Initialize LED
ENC_Init
Set initial struct to register
Initialize GPIO for ENC
ENC_Enable
Enable the ENC
While(1)
f_enc = ENC_GetENCFlag(EN0)
Get the flag of ENC
g_flag = f_enc.Bit.RotationDirection;
LED_On(LED4);
Delay2(enccount);
LED_Off(LED4);
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 */
    LEDInit();
```

ENC 初期化構造体を用意し、ENC0 を初期化します。

```
    /* ENC0 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(EN0,&m_enc);
    ENC_SetCounterReload(EN0,0xFFFU);
```

GPIO を ENC0 に設定します。PD0 と PD1 は ENC0 入力に設定します。

```
    /* GPIO initialization */
    GPIO_SetInputEnableReg(GPIO_PD, GPIO_BIT_0, ENABLE);/*Set PD0 as
input */
    GPIO_EnableFuncReg(GPIO_PD, GPIO_FUNC_REG_1, GPIO_BIT_0);/*Set
PD0 as ENCA0 */
    GPIO_SetInputEnableReg(GPIO_PD, GPIO_BIT_1, ENABLE);/*Set PD1 as
```

```
input */
    GPIO_EnableFuncReg(GPIO_PD, GPIO_FUNC_REG_1, GPIO_BIT_1);/*Set
PD1 as ENCB0 */
```

ENC0 と ENC0 割り込みを許可します。

```
    /* Enable ENC0 */
    ENC_Enable(EN0);
    /* Enable ENC0 interrupt */
    NVIC_EnableIRQ(INTENC0_IRQn);
```

ホイールマウスの回転を検出するため ENC0 の回転方向検出状態を取得します。回転数に応じて LED4 点滅します。

```
    /* LED4 blink speed based on the rolling of mouse wheel */
    while(1){
        f_enc = ENC_GetENCFlag(EN0);
        g_flag = f_enc.Bit.RotationDirection;
        LedOn(LED4);
        Delay2(enccount);
        LedOff(LED4);
        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;
        }
    }
```

7-16 PMD

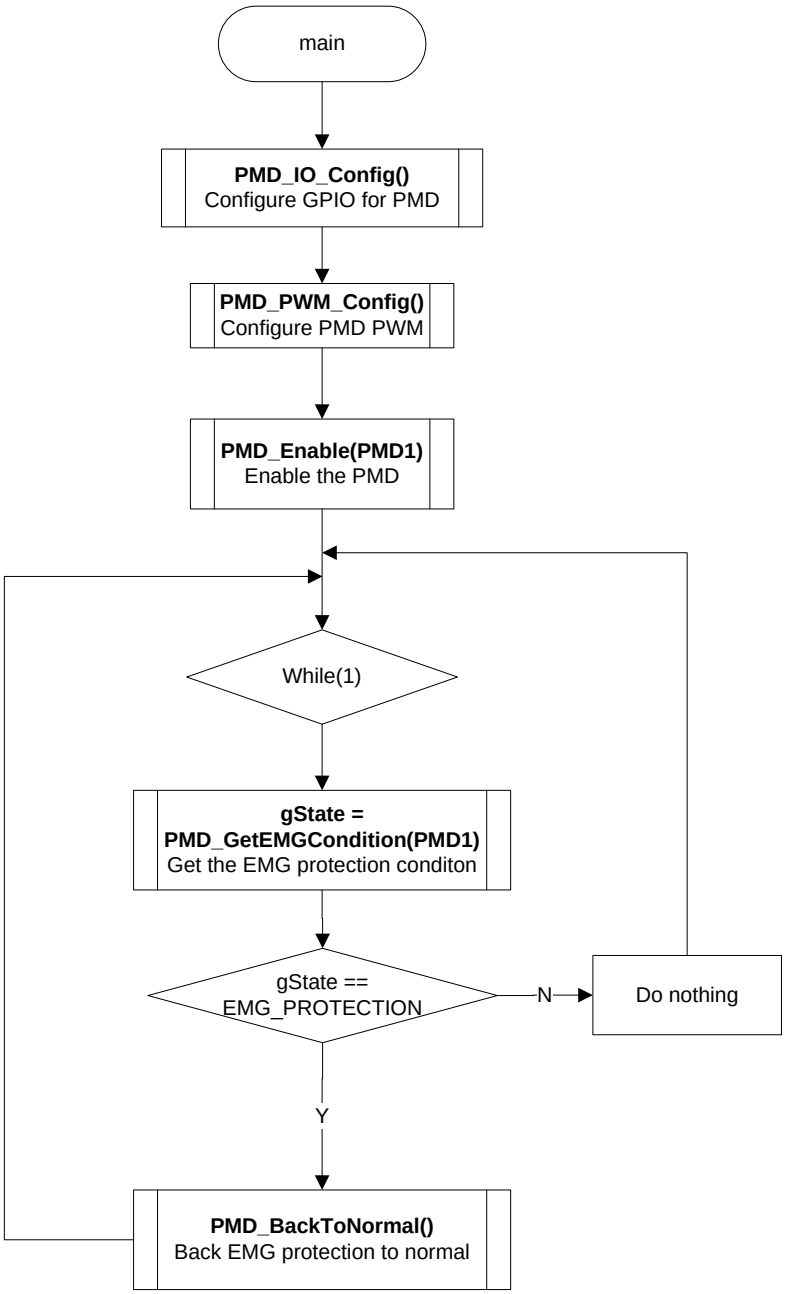
7-16-1 例: 位相出力

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 */
LEDInit();

/* 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 in protection */
            LedOn(LED4);
            /* Back EMG protection to normal */
            PMD_BackToNormal();
            Delay(1000U);
        } else {
            /* LED4 will light off if the condition is not in protection */
            LedOff(LED4);
        }
    }
```