SH7450 グループ/SH7451 グループ
タイマユニット(TMU)割り込みサンプルコード

R01AN0570JJ0100
Rev. 1.00
2012.2.1

要旨

本アプリケーションノートは、SH7450 グループ/SH7451 グループ(以下、SH7450)のソフトウェアを初めて作成する場合を想定した導入用サンプルコードの説明資料です。サンプルコードは、SH7450 内蔵のタイマユニット(以下、TMU)、TMU の割り込み、およびポートを使用し、SH7450 評価基板: R0K474504C000BR/R0K474504C010BR (以下、SH7450 評価基板)に搭載している LED を点滅させる動作をします。

対象デバイス

SH7450 グループ/SH7451 グループ

本アプリケーションノートを他のマイコンへ適応する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

目次

1. 仕様

本サンプルコードは、SH7450 内蔵の TMU チャネル 0 (以下、TM0)で 1 秒を連続して計測し、SH7450 評価基板に搭載している LED の点灯パターンを 1 秒毎に更新します。TM0 割り込み処理内で、LED と接続しているポート G の出力データを、0 からカウントアップすることにより、LED の点灯パターンを更新します。

※評価基板に関する詳細は、SH7450 評価基板の仕様書を参照してください。

表 1.1に使用する周辺機能と用途を示します。

表1.1 使用する周辺機能と用途

周辺機能	用途
TMU (タイマユニット)	LED 点灯間隔 1 秒のカウント
I/O ポート G	LED 点灯データ出力(8 ビット出力)
割り込みコントローラ (INTC)	TM0 アンダフローによる割り込み要求の制御

2. 動作確認条件

本アプリケーションノートのサンプルコードは、下記の条件で動作を確認しています。

表2.1 動作確認条件

項目	内容
使用マイコン	SH7450 グループ/SH7451 グループ
動作周波数	入力周波数　　：20MHz CPU クロック(Ick)　：240MHz SHwy クロック(SHck)　：80MHz 周辺クロック(Pck)　：40MHz
動作電圧	PVcc=Vcc=PLLVcc=AVcc=5V、Vdd=1.5V
動作モード	シングルチップモード
統合開発環境	ルネサスエレクトロニクス製 High-performance Embedded Workshop (以下、HEW) Version 4.09.00.007
C/C++コンパイラ	ルネサスエレクトロニクス製 C/C++ compiler package for SuperH RISC engine family V.9.04 Release 00
	オプション -cpu=sh4a -object="$(CONFIGDIR)¥$(FILELEAF).obj" -debug -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 -nologo (上記オプションは、統合開発環境のデフォルト設定です)
サンプルコードのバージョン	Version 1.00
使用する評価基板	SH7450 評価基板 (製品型名:R0K474504C000BR/R0K474504C010BR)

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

・SH7450 グループ/SH7451 グループ レジスタ定義ヘッダファイル(R01AN0190JJ0102)

4. ハードウェア説明

4.1 使用端子一覧

表 4.1に使用端子と機能を示します。

表4.1 使用端子と機能

端子名	入出力	内容
ポート G(ビット 7～0)	出力	SH7450 評価基板に搭載の LED 制御

4.2 レジスタ設定

サンプルコードで使用している主なレジスタの設定を以下に示します。なお、設定値はサンプルコードで使用している値であり、初期値とは異なる値もあります。

(1) ステータスレジスタ(SR)

アドレス	設定値	ビット		機能
-	H'4000 0000 or H'5000 0000 or H'7000 0000	31	-	0：予約ビット
		30	MD	0：ユーザモード 1：特権モード
		29	RB	0：R0_BANK0～R7_BANK0 は汎用レジスタ R0～R7 でアクセス R0_BANK1～R7_BANK1 は LDC/STC 命令でアクセス 1：R0_BANK1～R7_BANK1 は汎用レジスタ R0～R7 でアクセス R0_BANK0～R7_BANK0 は LDC/STC 命令でアクセス
		28	BL	0：割り込み要求をマスクしない 1：割り込み要求をマスクする
		27～16	-	0：予約ビット
		15	FD	0：FPU 命令許可
		14～10	-	0：予約ビット
		9	M	0：DIV0S、DIV0U、DIV1 命令で使用します
		8	Q	0：DIV0S、DIV0U、DIV1 命令で使用します
		7～4	IMASK	0：IMASK 以下のレベルの割り込みはマスクします
		3～2	-	0：予約ビット
		1	S	0：MAC 命令の飽和動作を指定します
		0	T	0：真/偽条件、キャリ、ボロー、オーバフローまたはアンダフローなどを表します

(2) ポート G データレジスタ(PGDR)：ポート G の入出力データを格納します。

アドレス	設定値	ビット		機能
H'FFFF 5802	H'0000 ～ H'00FF	15 ～ 8	-	0：予約ビット
		7	PG7DR	出力設定時は、書き込み値を各端子に出力 入力設定時は、端子状態を読み込み
		～	～	
		0	PG0DR	

(3) ポート G・IO レジスタ(PGIOR)：各端子の入出力方向を設定します。

アドレス	設定値	ビット		機能
H'FFFF 5806	H'00FF	15 ～ 8	-	0：予約ビット
		7	PG7IOR	1：対応する端子を出力にする
		～	～	
		0	PG0IOR	

(4) ポート G コントロールレジスタ 1 (PGCR1)：ポート G にあるマルチプレクス端子の機能を選びます。

アドレス	設定値	ビット		機能
H'FFFF 5816	H'0000	15	-	0：予約ビット
		14 ～ 12	PG3MD	PG3 モードビット 000：PG3 入出力(ポート)
		11	-	0：予約ビット
		10 ～ 8	PG2MD	PG2 モードビット 000：PG2 入出力(ポート)
		7	-	0：予約ビット
		6 ～ 4	PG1MD	PG1 モードビット 000：PG1 入出力(ポート)
		3	-	0：予約ビット
		2 ～ 0	PG0MD	PG0 モードビット 000：PG0 入出力(ポート)

(5) ポート G コントロールレジスタ 2 (PGCR2)：ポート G にあるマルチプレクス端子の機能を選びます。

アドレス	設定値	ビット		機能
H'FFFF 5814	H'0000	15	-	0：予約ビット
		14 ～ 12	PG7MD	PG7 モードビット 000：PG7 入出力(ポート)
		11	-	0：予約ビット
		10 ～ 8	PG6MD	PG6 モードビット 000：PG6 入出力(ポート)
		7	-	0：予約ビット
		6 ～ 4	PG5MD	PG5 モードビット 000：PG5 入出力(ポート)
		3	-	0：予約ビット
		2 ～ 0	PG4MD	PG4 モードビット 000：PG4 入出力(ポート)

(6) TM スタートレジスタ (TMSTR)： TMnCNT (n＝0～2) カウンタ動作/停止を選択します。

アドレス	設定値	ビット		機能
H'FFFF D004	H'00 or H'01	7 ～ 3	-	0：予約ビット
		2	STR2	TM2 カウンタスタートビット 0：TM2CNT カウンタのカウント動作は停止
		1	STR1	TM1 カウンタスタートビット 0：TM1CNT カウンタのカウント動作は停止
		0	STR0	TM0 カウンタスタートビット 0：TM0CNT カウンタのカウント動作は停止 1：TM0CNT カウンタはカウント動作する

(7) TM0 コンスタントレジスタ(TM0COR)：TM0CNT カウンタのリロード値を格納します。

アドレス	設定値	ビット		機能
H'FFFF D008	H'0000 9896	31 ～ 0	TM0COR	TM0CNT カウンタアンダフロー時に TM0CNT カウンタにセットする 32 ビットレジスタ値

(8) TM0 カウンタ (TM0CNT)：TM0 のカウンタ値です。

アドレス	設定値	ビット		機能
H'FFFF D00C	H'0000 9896	31 ～ 0	TM0CNT	TM0CR レジスタの TPSC ビットにより選択した入力クロックで、カウントダウン動作を行う

(9) TM0 コントロールレジスタ(TM0CR)：カウントクロックの選択、アンダフロー発生時の割り込み制御

アドレス	設定値	ビット		機能
H'FFFF D010	H'0024	15 ～ 9	-	0：予約ビット
		8	UNF	アンダフローフラグ [クリア]UNF ビットに"0"を書き込んだとき [セット]TM0CNT カウンタがアンダフローしたとき
		7 ～ 6	-	0：予約ビット
		5	UNIE	アンダフロー割り込み制御ビット 1：アンダフローによる割り込み(TUNI)を許可する
		4 ～ 3	-	0：予約ビット
		2 ～ 0	TPSC	TM0CNT カウンタのカウントクロックを選択します。 100：Pck/1024 でカウント

(10) ユーザ割り込みマスクレベル設定レジスタ(USERIMASK)：UIMASK 設定値以下のレベルの割り込みをマスクします

アドレス	設定値	ビット		機能
H'FFFF F300	H'A500 0000	31 ～ 24	USERIMASKKEY	USERIMASK レジスタライトキーコードビット H'A5：UIMASK ビット書き換え可能
		23 ～ 8	-	0：予約ビット
		7 ～ 4	UIMASK	ユーザ割り込みマスクレベルビット UIMASK 設定値以下のレベルの割り込みをマスク
		3 ～ 0	-	0：予約ビット

(11) 割り込み優先順位設定レジスタ 0 (INT2PRI0)：
内蔵周辺モジュール割り込みの優先順位(レベル 31～0)を設定します。設定する値が大きいほど優先順位が高くなります。個々の割り込み要因を 5 ビットで 32 通り、30 レベル(設定値 H'00 と H'01 は要求がマスクされていることと同じ状態です)の優先レベルに割り付けることができます。

アドレス	設定値	ビット		機能
H'FFFF F400	H'0200 0000	31～29	-	0：予約ビット
		28～24	TUNI0	TUNI0(TMU)割り込みの優先順位(レベル 31～0)を設定
		23～21	-	0：予約ビット
		20～16	TUNI1	TUNI1(TMU)割り込みの優先順位(レベル 31～0)を設定
		15～13	-	0：予約ビット
		12～8	TUNI2	TUNI2(TMU)割り込みの優先順位(レベル 31～0)を設定
		7～5	-	0：予約ビット
		4～0	予約	-

(12) 割り込みマスククリアレジスタ 0 (INT2MSKCR) :

INT2MSKCR レジスタは、割り込みマスクレジスタ 0（INT2MSKR）に設定されたマスクをクリアすることができます。本レジスタの該当ビットに"1"を設定するとそのビットに対応する割り込み要因のマスクがクリアされます。読み出しは常に"0"です。

アドレス	設定値	ビット		機能
H'FFFF F43C	H'0000 0001	31 ～ 30	-	0：予約ビット
		29	CMIG5	タイマ G5 割り込みマスククリア設定ビット
		28	CMIG4	タイマ G4 割り込みマスククリア設定ビット
		27	CMIG3	タイマ G3 割り込みマスククリア設定ビット
		26	CMIG2	タイマ G2 割り込みマスククリア設定ビット
		25	CMIG1	タイマ G1 割り込みマスククリア設定ビット
		24	CMIG0	タイマ G0 割り込みマスククリア設定ビット
		23	TF	タイマ F 割り込みマスククリア設定ビット
		22	TA	タイマ A 割り込みマスククリア設定ビット
		21	ADC	ADC 割り込みマスククリア設定ビット
		20	IICI	IIC3 割り込みマスククリア設定ビット
		19	DRO	DRO 割り込みマスククリア設定ビット
		18	DRI2	DRI2 割り込みマスククリア設定ビット
		17	DRI1	DRI1 割り込みマスククリア設定ビット
		16	DRI0	DRI0 割り込みマスククリア設定ビット
		15	HUDI	H-UDI 割り込みマスククリア設定ビット
		14	RSPI2	RSPI2 割り込みマスククリア設定ビット
		13	RSPI1	RSPI1 割り込みマスククリア設定ビット
		12	RSPI0	RSPI0 割り込みマスククリア設定ビット
		11	SCIF3	SCIF3 割り込みマスククリア設定ビット
		10	SCIF2	SCIF2 割り込みマスククリア設定ビット
		9	SCIF1	SCIF1 割り込みマスククリア設定ビット
		8	SCIF0	SCIF0 割り込みマスククリア設定ビット
		7	DMAC6T11	DMA6～DMA11 割り込みマスククリア設定ビット
		6	DMAC4T5	DMA4～DMA5 割り込みマスククリア設定ビット
		5	DMAC0T3	DMA0～DMA3 割り込みマスククリア設定ビット
		4	WDT	WDT 割り込みマスククリア設定ビット
		3	-	0：予約ビット
		2	TUNI2	TMU2 割り込みマスククリア設定ビット
		1	TUNI1	TMU1 割り込みマスククリア設定ビット
		0	TUNI0	TMU0 割り込みマスククリア設定ビット

5.　ソフトウェア説明

5.1　　動作概要

本サンプルコードは、SH7450 内蔵のTM0 で 1 秒を連続して計測し、TM0 割り込み処理内でポートGの出力データをカウントアップして、SH7450 評価基板に搭載しているLEDを点滅させる動作をします。図 5.1に、処理全体の概略フローを示します。

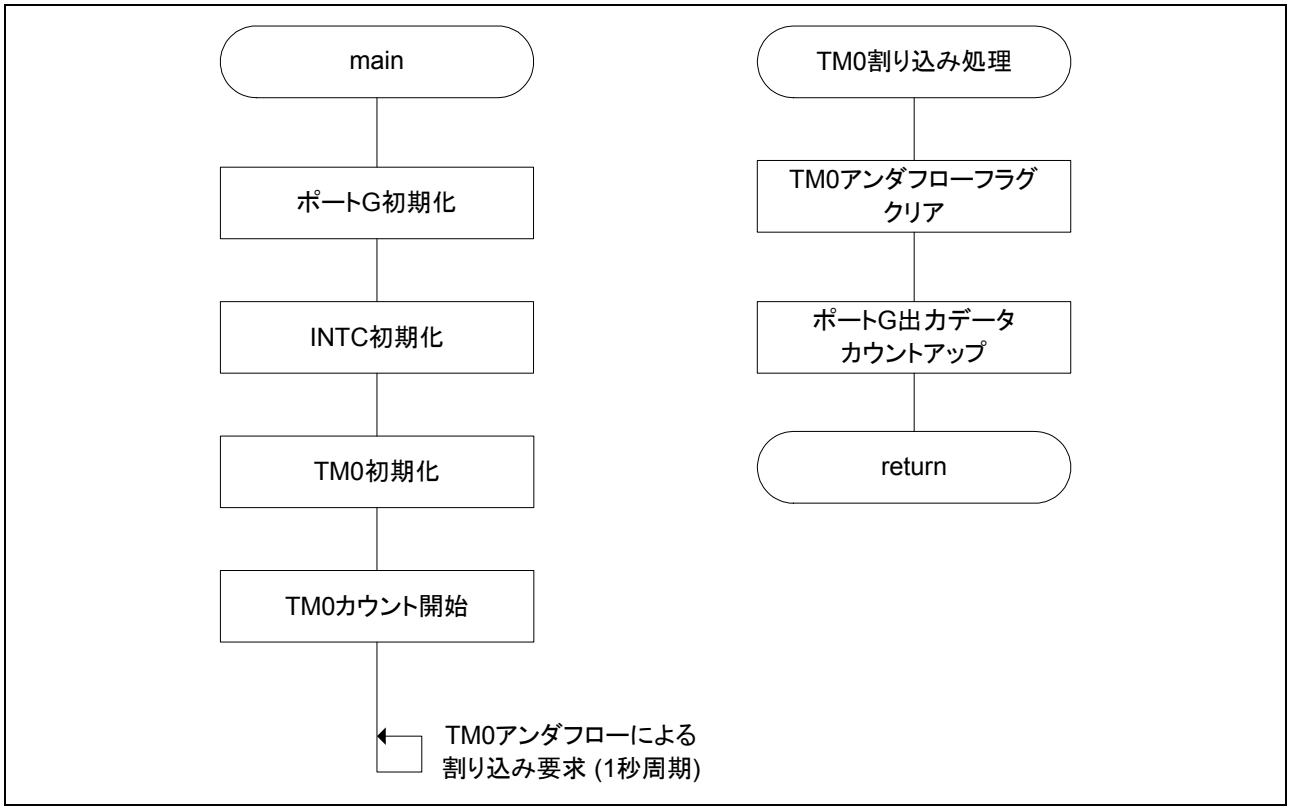


図5.1 処理全体の概略フロー

表 5.1にサンプルコードのセクション情報を示します。

表5.1 セクション情報

アドレス	セクション名	説明
H'0000 0800	INTHandler	例外ハンドラ
	VECTTBL	リセットベクタテーブル
	INTTBL	割り込みベクタテーブル
	IntPRG	割り込み関数プログラム
H'0000 1800	PResetPRG	リセットプログラム
H'0000 2000	P	プログラム領域
	C	定数領域
	C$BSEC	B セクション初期化用テーブル
	C$DSEC	D セクション初期化用テーブル
	D	初期化データ領域 (ROM)
H'A000 0000	RSTHandler	リセットハンドラ
H'E500 E000	B	未初期化データ領域
	R	初期化データ領域 (RAM)
H'E501 1C00	S	スタックアドレス領域

5.2　ファイル構成

表 5.2にサンプルコードのファイル構成を示します。

表5.2 ファイル構成

ファイル名	内容	備考
dbsct.c	B セクション、D セクション初期化用アドレステーブル	
env.inc	例外処理に関するレジスタアドレスの定義	
intprg.src	割り込み関数プログラム	
iodefine.h	SH7450 グループ/SH7451 グループ用レジスタの定義	
main.c	メイン関数プログラム	
resetprg.c	リセットプログラム	
stacksct.h	スタックサイズの定義	
typedefine.h	データ型の定義	
vect.inc	ベクタの定義	
vecttbl.src	ベクタテーブル	
vhandler.src	リセット/割り込みハンドラプログラム	

5.3　定数一覧

定数は使用していません。

5.4　構造体/共有体一覧

SH7450 グループ/SH7451 グループ用レジスタの定義ファイル(iodefine.h)以外は構造体/共有体を使用しておりません。iodefine.hの構造体/共有体については、3章のアプリケーションノートを参照してください。

5.5　変数一覧

グローバル変数は使用していません。

5.6　関数一覧

表 5.3に関数を示します。

表5.3 関数

関数名	概要
main	C 言語の main 関数。ポート G と TM0 の初期化、および TM0 の起動。
INT_TMU0_TUNI0	TM0 アンダフロー割り込み関数。

5.7　関数仕様

サンプルコードの関数仕様を示します。

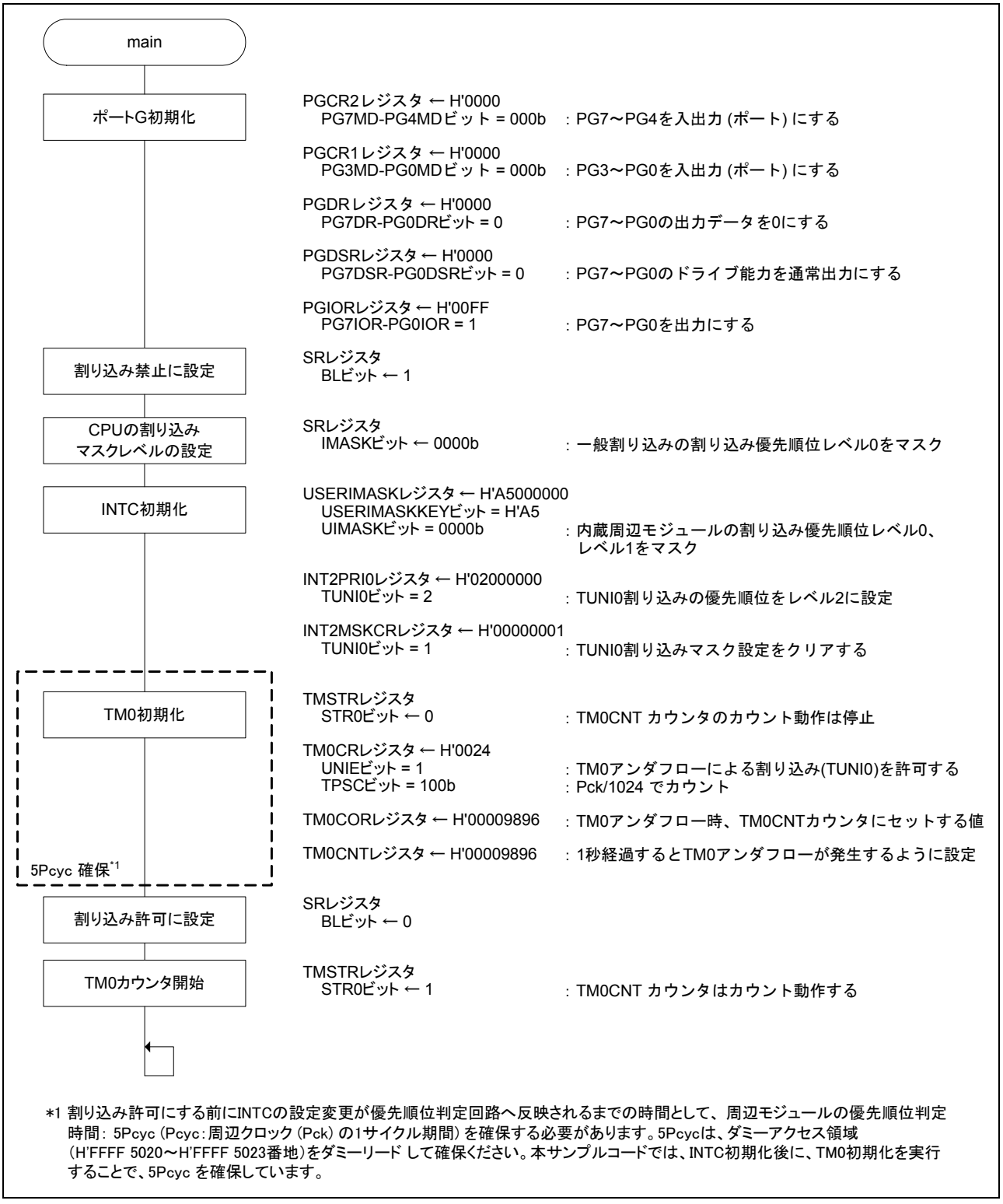
main	
概要	C 言語の main 関数。ポート G と TM0 の初期化を行い、TM0 を起動します。
ヘッダ	なし
宣言	void main(void)
説明	• ポート G と TM0 を初期化し、TM0 カウンタのカウント動作を開始します。 • TM0 カウント開始後は、TM0 アンダフロー割り込み（TUNI0）の発生を待ちます（無限ループ）。
引数	なし
リターン値	なし
備考	なし

INT_TMU0_TUNI0	
概要	TM0 アンダフロー割り込み（TUNI0）関数
ヘッダ	なし
宣言	void INT_TMU0_TUNI0 (void)
説明	• TM0 アンダフローフラグをクリアします。 • ポートGの出力データをカウントアップ(LEDの点灯パターンを更新)します。
引数	なし
リターン値	なし
備考	なし

5.8　フローチャート

5.8.1　main関数

図 5.2にmain関数のフローチャートを示します。



*1 割り込み許可にする前にINTCの設定変更が優先順位判定回路へ反映されるまでの時間として、周辺モジュールの優先順位判定時間：5Pcyc（Pcyc：周辺クロック（Pck）の1サイクル期間）を確保する必要があります。5Pcycは、ダミーアクセス領域（H'FFFF 5020～H'FFFF 5023番地）をダミーリード して確保ください。本サンプルコードでは、INTC初期化後に、TM0初期化を実行することで、5Pcyc を確保しています。

図5.2 main 関数

5.8.2　INT_TMU0_TUNI0 関数

図 5.3にINT_TMU0_TUNI0 関数のフローチャートを示します。

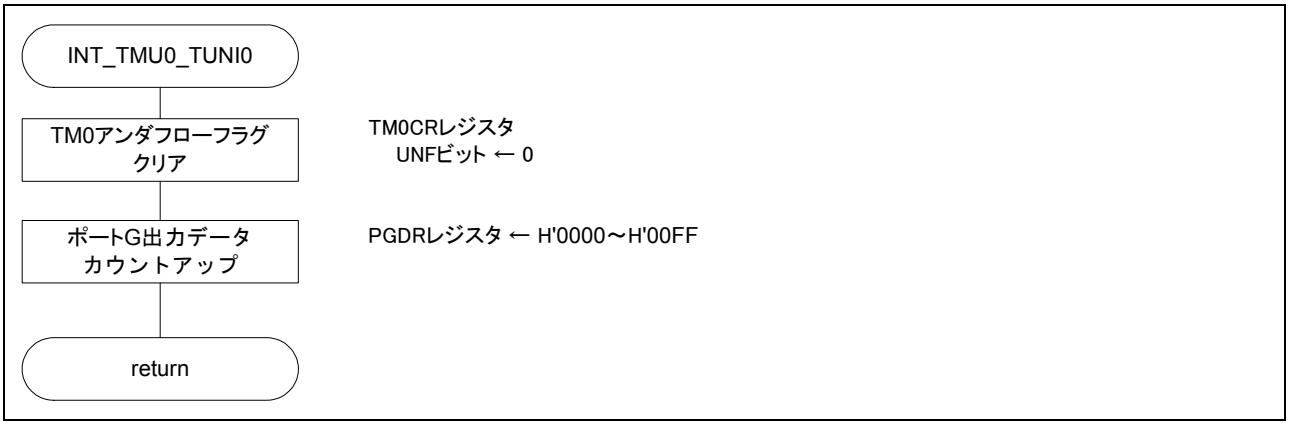


図5.3 INT_TMU0_TUNI0 関数

6. サンプルコード

サンプルコードは、ルネサス エレクトロニクスホームページから入手してください。

7. 参考ドキュメント

SH7450 グループ、SH7451 グループ　ユーザーズマニュアル　ハードウェア編　Rev.1.10
（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート/テクニカルニュース
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

C コンパイラマニュアル
SuperH ファミリ用 C/C++コンパイラパッケージ V.9.04 Release 00
SuperH C/C++コンパイラパッケージ V.9.04 ユーザーズマニュアル
（最新版をルネサス エレクトロニクスホームページから入手してください。）

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry

付録A: 割り込み関数を追加する際の留意点

表A.1にHEWのプログラム開発環境で、内蔵周辺モジュールの割り込みハンドラを追加する際の留意点について列挙します。

表A.1 HEWのプログラム開発環境で内蔵周辺モジュールの割り込みハンドラを追加する際の留意点

留意点	詳細
IMASK と UIMASK の設定	• IMASK はステータスレジスタ(SR)、UIMASK はユーザ割り込みマスクレベル設定レジスタ (USERIMASK) で設定します。 • IMASK 値と UIMASK 値以下のレベルの割り込みはマスクされます。
割り込み優先順位設定	• 割り込み優先順位設定レジスタ 0～12 (INT2PRI0～INT2PRI12) で割り込みの優先順位を設定します。 • 割り込みを許可する場合は、優先レベルを 2 以上に設定します (優先レベル 0 と 1 は割り込みがマスクされている状態です)。 • CPU の割り込み優先順位は 4 ビット (レベル 0～レベル 15) で判定するため、INT2PRI0～INT2PRI12 で設定した 5 ビットの優先順位 (レベル 0～レベル 31) は、最下位 1 ビットを切り捨てた 4 ビットに変換して通知されます。
INTC の割り込みマスク設定クリア	• 割り込みマスククリアレジスタ 0 (INT2MSKCR)、割り込みマスククリアレジスタ 1 (INT2MSKCR1) で、使用する内蔵周辺モジュールに対する INTC の割り込みマスクをクリアします。
各内蔵周辺モジュールの割り込みを許可	• 各内蔵周辺モジュールの割り込みを許可にします。 例 : TMU の場合、TM0 コントロールレジスタ(TM0CR) のアンダフロー割り込み制御ビット (UNIE) を "1" (アンダフローによる割り込み（TUNI）を許可) に設定することで、TMU の TM0 割り込みを許可にします。
HEW の割り込みベクタテーブルへの割り込み関数登録	• 各要因の割り込み関数は、HEW が生成する vecttbl.src ファイルの割り込みベクタテーブルに、既に登録されています。 • 割り込み関数名をそのまま使用する場合は、ベクタテーブルに登録されている割り込み関数の内容を変更して下さい。 • 新しい割り込み関数名を使用する場合は、新しい割り込み関数を外部参照 (import) 宣言した後に、ベクタテーブルに登録して下さい。
割り込み要求のクリア	• 割り込み関数内で、割り込み要求をクリアします。 例 : TM0 の場合、TM0 割り込み関数内で、TM0 コントロールレジスタ (TM0CR) のアンダフローフラグ (UNF) を "0" (TM0CNT カウンタがアンダフローを起こしていない) にすることで、TM0 割り込み要求をクリアします。
HEW の割り込みマスクテーブルへの割り込みマスク値登録	• HEW が生成する vecttbl.src ファイルの、割り込みマスクテーブルに、割り込み優先順位のレベルをマスク値として設定します。 • 割り込みマスクテーブルに割り込み優先順位のレベル以上をマスク値として設定しなかった場合、多重割り込みを許可 (SR レジスタの BL を 0 に設定) にした時点で、再度同じ割り込み要求を CPU が受け付け、割り込みハンドラ処理を実行します。割り込み関数内で割り込み要求をクリアする場合、割り込みハンドラ処理を連続するため、割り込み関数には分岐できません。そのため、割り込み要求をクリアできずに、割り込みハンドラ処理を無限に実行します。

HEW のプログラム開発環境で、内蔵周辺モジュールである TM0 の割り込みハンドラを追加するサンプルコード例を以下に示します。緑色の表記は補足説明を、赤色の表記は内容の変更が必要な箇所を表しています。

- **IMASK と UIMASK 設定、割り込み優先順位設定、INTC の割り込みマスク設定クリア、各内蔵周辺モジュールの割り込みマスク設定クリア**

main.c

```
#include <machine.h>
#include "iodefine.h"

void main(void)
{
    ～(省略)～
    set_cr((int)((unsigned int)get_cr()|0x10000000U)); ←BL=1 (割り込み要求をマスクする)
    set_cr((int)((unsigned int)get_cr()&0xFFFFFF0FU)); ←IMASK 設定
    INTC.USERIMASK.LONG = 0xA5000000UL; ←UIMASK 設定
    INTC.INT2PRI0.LONG = 0x02000000UL; ←割り込み優先順位設定
    INTC.INT2MSKCR.LONG = 0x00000001UL; ←INTC の割り込みマスク設定クリア
    ～(省略)～
    TMU.TM0CR.WORD = 0x0024U; ←各内蔵周辺モジュールの割り込みマスク設定クリア
    ～(省略)～
    set_cr((int)((unsigned int)get_cr()&0xEFFFFFFFU)); ←BL=0 (割り込み要求をマスクしない)
        ⋮
}
```

- **HEW の割り込みマスクテーブルへの割り込みマスク値登録**

vecttbl.src

```
_RESET_Vectors:
;<<VECTOR DATA START (POWER ON RESET)>>
       ;H'000 Power On Reset (H-UDI RESET)
       .data.l        _PowerON_Reset
;<<VECTOR DATA END (POWER ON RESET)>>
; Reserved
       .datab.l       8,H'00000000


       .section    INTTBL,data
       .export     _INT_Vectors

    ～(省略)～
       .export     _INT_MASK
_INT_MASK:
    ～(省略)～                  割り込み優先順位のレベルを、マスクテーブルのマスク値に設定
       ;H'580 TMU0              (INT2PRI0 レジスタに設定した TM0 の割り込み優先順位 (5 ビット) の最下位 1
       .data.b        H'10 ←ビットを切り捨て、4 ビットに変換した値を、マスク値の上位 4 ビットに設定
            ⋮                  します。TM0 割り込み関数実行中、ここで設定したマスク値の上位 4 ビットが
                               IMASK にセットされます。上記 main.c の変更例では、INT2PRI0
                               レジスタで、TM0 の割り込み優先順位を "H'02" に設定しているため、TM0 の
                               マスク値は、"H'10" を設定します。)
```

- **HEW の割り込みベクタテーブルへの割り込み関数登録（割り込み関数名をそのまま使用する場合）**

vecttbl.src

```
_RESET_Vectors:
;<<VECTOR DATA START (POWER ON RESET)>>
       ;H'000 Power On Reset (H-UDI RESET)
       .data.l       _PowerON_Reset
;<<VECTOR DATA END (POWER ON RESET)>>
; Reserved
       .datab.l       8,H'00000000


       .section    INTTBL,data
       .export     _INT_Vectors

_INT_Vectors:
     ～(省略)～
       ;H'580 TMU0
       .data.l       _INT_TMU0_TUNI0 ←TM0 割り込み関数
         ⋮
```

- **割り込み関数をアセンブリ言語で記述（割り込み関数名をそのまま使用する場合）**

intprg.src

```
    .include    "vect.inc"
    .section    IntPRG, code

    ;H'040 Data TLB miss exception(read)
_INT_TLB_MISS_READ_EXP
     ～(省略)～
    ;H'580 TMU0
_INT_TMU0_TUNI0
     ～(省略)～ ←_アセンブリ言語で、割り込み要求のクリアと、その他実行する処理を記述
     RTS
     ～(省略)～
    ;H'5A0 TMU1
          ⋮
```

- **割り込み関数を C/C++言語で記述（割り込み関数名をそのまま使用する場合）**

intprg.src

```
    .include    "vect.inc"
    .section    IntPRG, code

    ;H'040 Data TLB miss exception(read)
_INT_TLB_MISS_READ_EXP：
：
     ～(省略)～
    ;H'580 TMU0
;_INT_TMU0_TUNI0 ←";(セミコロン)"で、TM0 割り込み関数をコメントアウトまたは削除
          ⋮
```

main.c

```
void INT_TMU0_TUNI0(void) ←TM0 割り込み関数を作成
{
     ～(省略)～ ←C/C++言語で、割り込み要求のクリアと、その他実行する処理を記述
}
```

- **HEW の割り込みベクタテーブルへの割り込み関数登録（新しい割り込み関数名を使用する場合）**

vecttbl.src

```
_RESET_Vectors:
;<<VECTOR DATA START (POWER ON RESET)>>
       ;H'000 Power On Reset (H-UDI RESET)
       .data.l       _PowerON_Reset
;<<VECTOR DATA END (POWER ON RESET)>>
; Reserved
       .datab.l       8,H'00000000


       .section    INTTBL,data
       .export     _INT_Vectors
       .import     _NEW_INT_TMU0_TUNI0 ←新しい TM0 割り込み関数の外部参照宣言

_INT_Vectors:
     ～(省略)～
       ;H'580 TMU0
;       .data.l    _INT_TMU0_TUNI0 ←";(セミコロン)"で既存の TM0 割り込み関数をコメントアウトまたは削除
        .data.l    _NEW_INT_TMU0_TUNI0 ←新しい TM0 割り込み関数を登録
          ⋮
```

- **割り込み関数をアセンブリ言語で記述（新しい割り込み関数名を使用する場合）**

intprg.src

```
    .include    "vect.inc"
    .section    IntPRG, code

    ;H'040 Data TLB miss exception(read)
_INT_TLB_MISS_READ_EXP：

     ～(省略)～
    ;H'580 TMU0
;_INT_TMU0_TUNI0 ←";(セミコロン)"で既存の TM0 割り込み関数をコメントアウトまたは削除
_NEW_INT_TMU0_TUNI0 ←新しい TM0 割り込み関数を作成
   ～(省略)～ ←アセンブリ言語で、割り込み要求のクリアと、その他実行する処理を記述
    RTS
    ～(省略)～
    ;H'5A0 TMU1
      ⋮
```

- **割り込み関数を C/C++言語で記述（新しい割り込み関数名を使用する場合）**

main.c

```
void NEW_INT_TMU0_TUNI0 (void) ←新しい TM0 割り込み関数を作成
{
     ～(省略)～ ←C/C++言語で、割り込み要求のクリアと、その他実行する処理を記述
}
```

付録B: 割り込み要求マスク動作について

表B.1と図B.1に、UIMASK=1、IMASK=2 のときのTM0 アンダフロー割り込みに対するマスク動作例を示します。表B.1と図B.1中の番号①～⑤は対応しています。なお、表B.1と図B.1の中で、レジスタのビットを、"レジスタ名.ビット名"のように表記しています。

表B.1 割り込みマスクレジスタ設定例

	TMU	INTC			CPU
	TM0CR.UNIE	INT2MSKR.TUNI0	INT2PRI0.TUNI0	USERIMASK.UIMASK	SR.IMASK
①	0	0	00000b ～ 11111b	0001b	0010b
		1	00000b ～ 11111b		
②	1	0	00000b ～ 00011b		
③			00100b ～ 00101b		
④			00110b ～ 11111b		
⑤		1	00000b ～ 11111b		

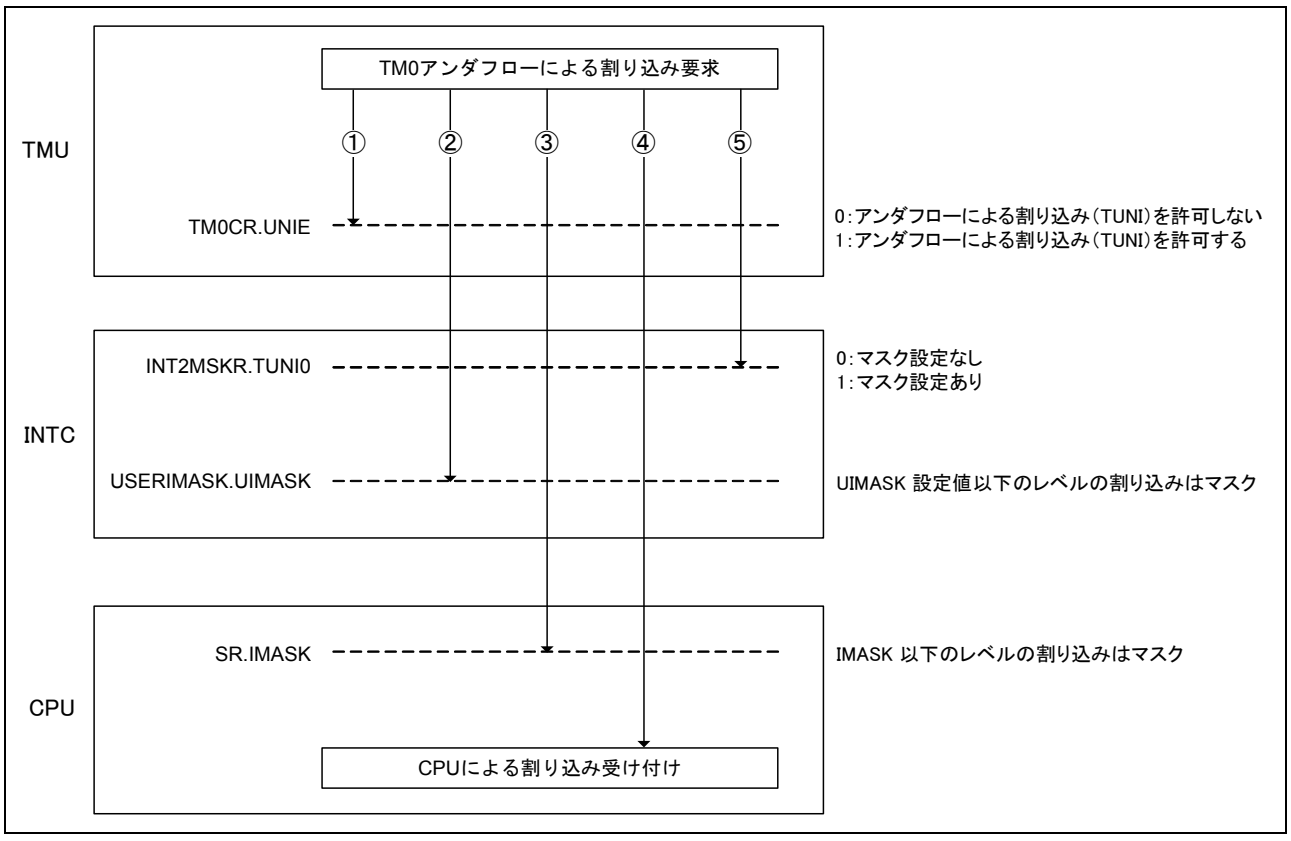


図 B.1 割り込みマスク動作例

①TM0CR レジスタの UNIE ビット が "0" (アンダフローによる割り込み（TUNI）を許可しない) のため、TMU 内で割り込み要求をマスクし、INTC へ割り込み要求を出力しません。そのため、INTC および CPU は TMU の割り込み要求を検出しません。

②TMU から INTC へ割り込み要求を出力します。しかし、TM0 の割り込み優先順位*1 が、USERIMASK レジスタの UIMASK 値以下のため、INTC は TMU の割り込み要求をマスクし、CPU へ割り込み要求を出力しません。そのため、CPU は TMU の割り込み要求を検出しません。

③TM0 の割り込み優先順位が UIMASK 値以上のため、TMU から INTC を経由して CPU へ割り込み要求を出力します。しかし、TM0 の割り込み優先順位*1 が、SR レジスタの IMASK 値以下のため、CPU は TMU の割り込み要求をマスクし、TMU の割り込み要求を検出しません。

④TM0 の割り込み優先順位が UIMASK 値以上、IMASK 値以上のため、TMU から INTC を経由して CPU へ割り込み要求を出力します。CPU は TMU の割り込み要求を検出して、割り込み要求を受け付けます。

⑤TMU から INTC へ割り込み要求を出力します。しかし、INT2MSKR レジスタの TUNI0 ビットが "1" (TM0 の割り込みマスク設定あり) のため、INTC は TMU の割り込み要求をマスクし、CPU へ割り込み要求を出力しません。そのため、CPU は TMU の割り込み要求を検出しません。

*1 IMASK 値、UIMASK 値は 4 ビットのため、内蔵周辺モジュール割り込みの優先順位 (5 ビット)は、最下位 1 ビットを切り捨てた 4 ビットに変換されます。

付録C: 多重割り込み動作について

本章は、HEW のプログラム開発環境における多重割り込みの動作について、TMU チャネル 0 (TM0)と TMU チャネル 1 (TM1)を用いて説明します。通常処理時では、IMASK/UIMASK、および BL に"0"を設定した例として説明します。

多重割り込み例 1

多重割り込み例 1 は、TM0 割り込み関数処理中に、割り込み優先順位の高い TM1 割り込みが発生した場合の動作例を示します。

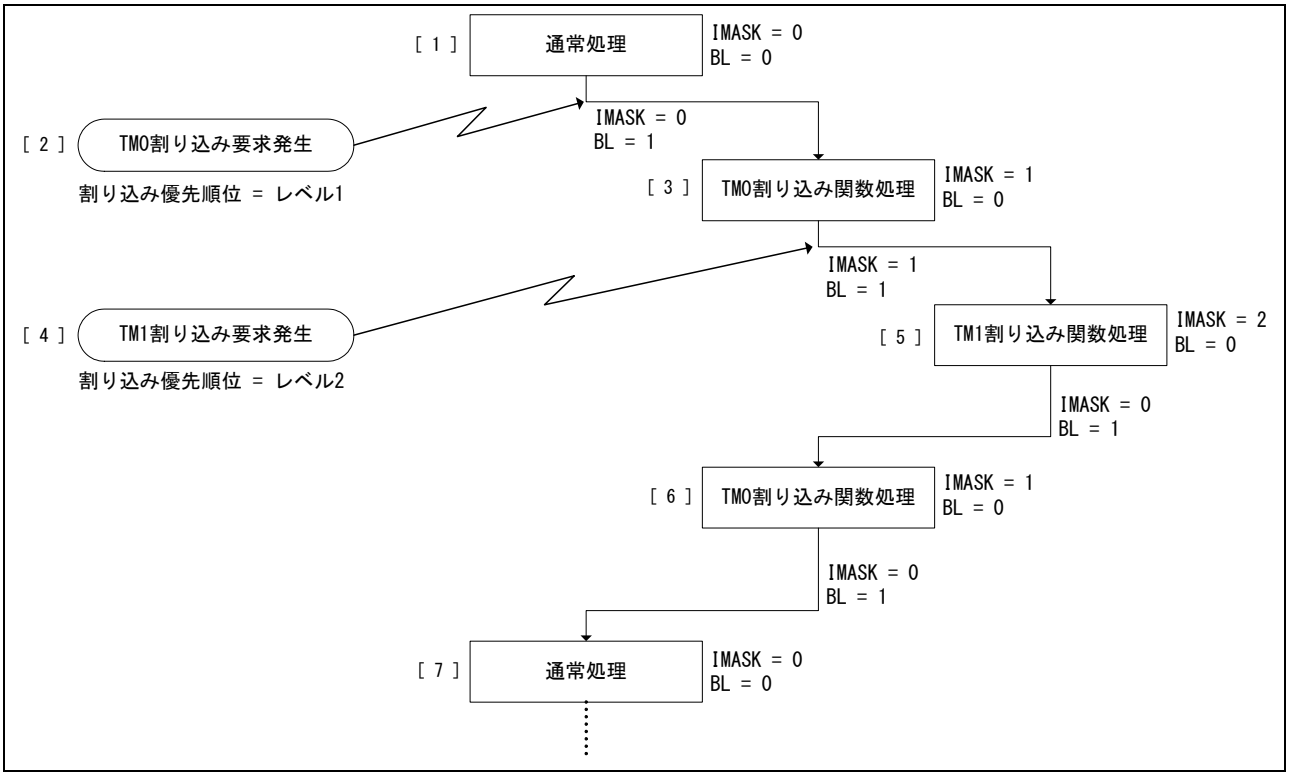


図 C.1 多重割り込み例 1 の処理フロー

[1] 通常処理中では、IMASK が"0" (レベル 0 をマスク)、BL が"0" (割り込み要求をマスクしない)の設定のため、割り込み優先順位がレベル 0 以外の割り込みを受け付けることができます。

[2] TM0 割り込み要求 (レベル 1) が発生したため、通常処理を中断し、割り込み処理に移行します。まずハードウェア処理で、BL を"1" (割り込み要求をマスクする)にセットします。その後、ソフトウェア処理で、通常処理で使用していたレジスタをスタックに退避します。IMASK に TM0 の割り込み優先順位である"1"を、BL に"0"を設定して、TM0 割り込み関数処理に分岐します。

[3] TM0 割り込み関数処理中は、IMASK が"1" (レベル 1 以下をマスク)、BL が"0"の設定のため、割り込み優先順位がレベル 2 以上の割り込みを受け付けることができます。

[4] TM1 割り込み要求 (レベル 2) が発生したため、TM0 割り込み関数処理を中断し、TM1 の割り込み処理に移行します。まずハードウェア処理で、BL を"1"にセットします。その後、ソフトウェア処理で、TM0 割り込み関数処理で使用していたレジスタをスタックに退避します。IMASK に TM1 の割り込み優先順位である"2"を、BL に"0"を設定して、TM1 割り込み関数処理に分岐します。

[5] TM1 割り込み関数処理中は、IMASK が"2" (レベル 2 以下をマスク)、BL が"0"の設定のため、割り込み優先順位がレベル 3 以上の割り込みを受け付けることができます。

TM1 割り込み関数処理が完了すると、ソフトウェア処理で、IMASK に"0"を、BL に"1"をセット後、割り込み前の状態 (TM0 割り込み関数処理で使用していたレジスタ) を復帰して、中断していた処理を再開します。

[6] TM0 割り込み関数処理を再開します。IMASK と BL は、TM1 割り込み要求が発生する前の状態 ([3]と同じ状態)です。

TM0 割り込み関数処理が完了すると、ソフトウェア処理で、IMASK に"0"を、BL に"1"をセット後、割り込み前の状態 (通常処理で使用していたレジスタ) を復帰して、中断していた処理を再開します。

[7] 通常処理を再開します。IMASK と BL は、TM0 割り込み要求が発生する前の状態 ([1]と同じ状態)です。

多重割り込み例 2

多重割り込み例 2 は、TM1 割り込み関数処理中に、割り込み優先順位が低い TM0 割り込みが発生した場合の動作例を示します。

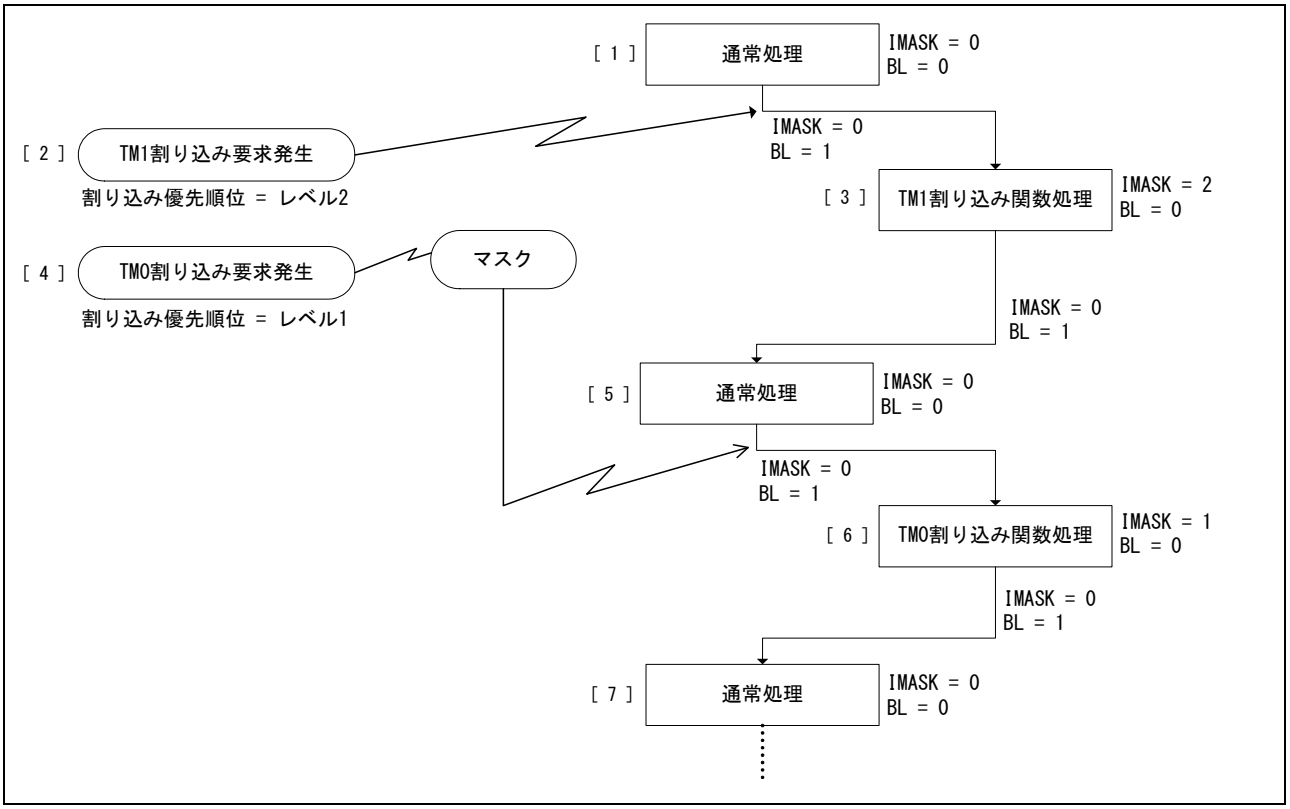


図 C.2 多重割り込み例 2 の処理フロー

[1] 通常処理中では、IMASK が"0" (レベル 0 をマスク)、BL が"0" (割り込み要求をマスクしない)の設定のため、割り込み優先順位がレベル 0 以外の割り込みを受け付けることができます。

[2] TM1 割り込み要求 (レベル 2) が発生したため、通常処理を中断し、TM1 割り込み処理に移行します。まず、ハードウェア処理で、BL を"1" (割り込み要求をマスクする)にセットします。その後、ソフトウェア処理で、通常処理で使用していたレジスタをスタックに退避します。IMASK に TM1 の割り込み優先順位である"2"を、BL に"0"を設定して、TM1 割り込み関数処理に分岐します。

[3] TM1 割り込み関数処理中は、IMASK が"2" (レベル 2 以下をマスク)、BL が"0"の設定のため、割り込み優先順位がレベル 3 以上の割り込みを受け付けることができます。

[4] TM0 割り込み要求 (レベル 1) は発生しても、割り込み優先順位のレベルが IMASK 値以下のため、マスクします。

TM1 割り込み関数処理が完了すると、ソフトウェア処理で、IMASK に"0"を、BL に"1"をセット後、割り込み前の状態 (通常処理で使用していたレジスタ) を復帰して、中断していた処理を再開します。

[5] 通常処理を再開します。IMASK と BL は、TM1 割り込み要求が発生する前の状態 ([1]と同じ状態)です。マスクしていた TM0 割り込み要求のレベルが IMASK 値より大きくなったため、TM0 割り込み要求を受け付けます。通常処理を中断後、ハードウェア処理で、BL を"1" にセットします。その後、ソフトウェア処理で、通常処理で使用していたレジスタをスタックに退避します。IMASK に TM0 の割り込み優先順位である"1"を、BL に"0"を設定して、TM0 割り込み関数処理に分岐します。

[6] TM0 割り込み関数処理中は、IMASK が"1" (レベル 1 以下をマスク)、BL が"0"の設定のため、割り込み優先順位がレベル 2 以上の割り込みを受け付けることができます。

TM0 割り込み関数処理が完了すると、ソフトウェア処理で、IMASK に"0"を、BL に"1"をセット後、割り込み前の状態 (通常処理で使用していたレジスタ) を復帰して、中断していた処理を再開します。

[7] 通常処理を再開します。IMASK と BL は、TM0 割り込み要求が発生する前の状態 ([1]と同じ状態)です。

多重割り込み例 3

多重割り込み例 3 は、割り込み優先順位が低い TM0 割り込みと、割り込み優先順位が高い TM1 割り込みが競合した場合の動作例を示します。

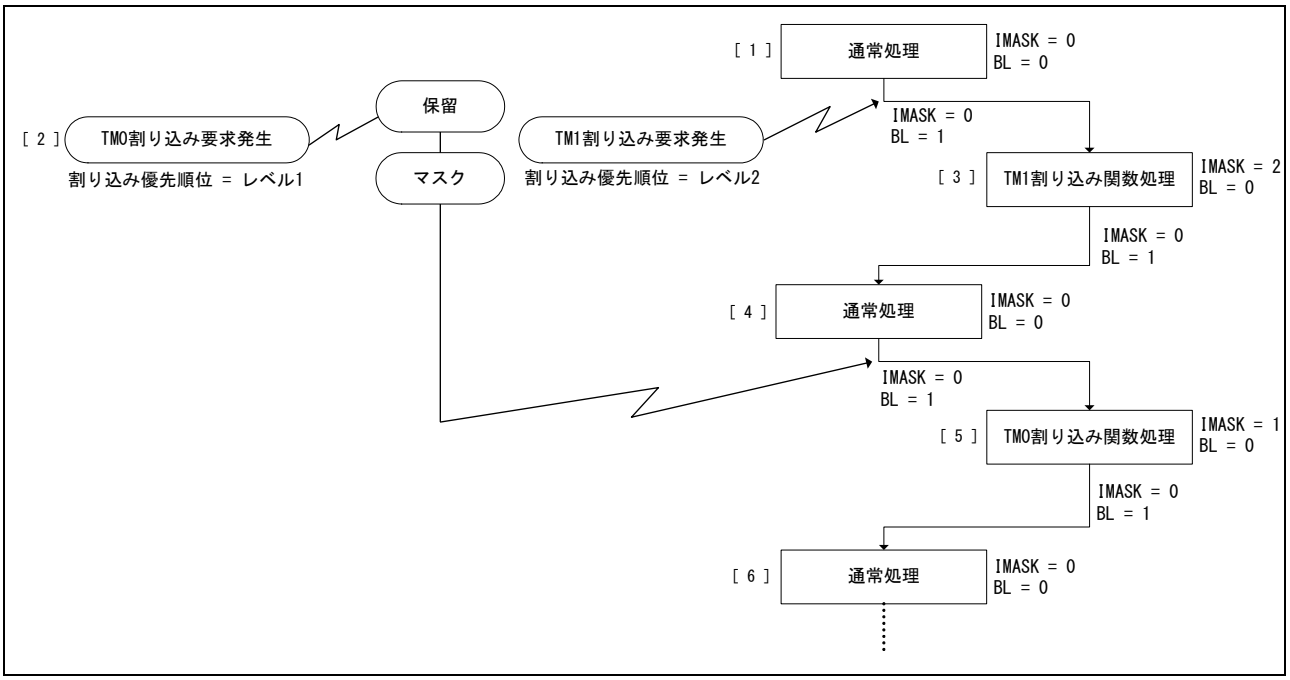


図 C.3 多重割り込み例 3 の処理フロー

[1] 通常処理中では、IMASK が"0" (レベル 0 をマスク)、BL が"0" (割り込み要求をマスクしない)の設定のため、割り込み優先順位がレベル 0 以外の割り込みを受け付けることができます。

[2] TM0 割り込み要求(レベル 1)と TM1 割り込み要求(レベル 2)で競合が発生します。INTC で、割り込み優先順位が比較され、割り込み優先順位が高い TM1 割り込み要求を出力します。TM0 割り込み要求は保留します。

TM1 割り込み要求の発生により、通常処理を中断します。ハードウェア処理で、BL を"1" (割り込み要求をマスクする)にセットします。その後、ソフトウェア処理で、通常処理で使用していたレジスタをスタックに退避します。IMASK に TM1 の割り込み優先順位である"2"を、BL に"0"を設定して、TM1 割り込み関数処理に分岐します。

[3] TM1 割り込み関数処理中は、IMASK が"2" (レベル 2 以下をマスク)、BL が"0"の設定のため、割り込み優先順位がレベル 3 以上の割り込みを受け付けることができます。

TM0 割り込み要求は、割り込み優先順位のレベルが IMASK 値以下であるため、マスクします。

TM1 割り込み関数処理が完了すると、ソフトウェア処理で、IMASK に"0"を、BL に"1"をセット後、割り込み前の状態 (通常処理で使用していたレジスタ) を復帰して、中断していた処理を再開します。

[4] 通常処理を再開します。IMASK と BL は、TM1 割り込み要求が発生する前の状態 ([1]と同じ状態)です。マスクしていた TM0 割り込み要求のレベルが IMASK 値より大きくなったため、TM0 割り込み要求を受け付けます。通常処理を中断後、ハードウェア処理で、BL を"1" にセットします。その後、ソフトウェア処理で、通常処理で使用していたレジスタをスタックに退避します。IMASK に TM0 割り込み優先順位である"1"を、BL に"0"を設定して、TM0 割り込み関数処理に分岐します。

[5] TM0 割り込み関数処理中は、IMASK が"1" (レベル 1 以下をマスク)、BL が"0"の設定のため、割り込み優先順位がレベル 2 以上の割り込みを受け付けることができます。

TM0 割り込み関数処理が完了すると、ソフトウェア処理で、IMASK に"0"を、BL に"1"をセット後、割り込み前の状態 (通常処理で使用していたレジスタ) を復帰して、中断していた処理を再開します。

[6] 通常処理を再開します。IMASK と BL は、TM0 割り込み要求が発生する前の状態 ([1]と同じ状態)です。

HEW のプログラム開発環境における、上記多重割り込みのサンプルコード例を以下に示します。緑色の表記は補足説明を、赤色の表記は内容の変更が必要な箇所を表しています。

- **IMASK (UIMASK)設定、割り込み優先順位設定、INTC の割り込みマスク設定クリア、各内蔵周辺モジュールの割り込みマスク設定クリア**

main.c

```
#include <machine.h>
#include "iodefine.h"

void main(void)
{
     ～(省略)～
     set_cr((int)((unsigned int)get_cr()|0x10000000U)); ←BL=1 (割り込み要求をマスクする)
     set_cr((int)((unsigned int)get_cr()&0xFFFFFF0FU)); ←IMASK を設定
     INTC.USERIMASK.LONG = 0xA5000000UL; ←UIMASK を設定

     INTC.INT2PRI0.LONG = 0x02040000UL; ←TM0 の割り込み優先順位を設定
                                          TM1 の割り込み優先順位を設定

     INTC.INT2MSKCR.LONG = 0x00000003UL; ←TM0 の割り込みマスククリア
                                           TM1 の割り込みマスククリア
     ～(省略)～
     TMU.TM0CR.BIT.UNIE = 1U; ←TM0 アンダフローによる割り込みを許可
     TMU.TM1CR.BIT.UNIE = 1U; ←TM1 アンダフローによる割り込みを許可
     ～(省略)～
     set_cr((int)((unsigned int)get_cr()&0xEFFFFFFFU)); ←BL=0 (割り込み要求をマスクしない)
          ⋮
}
```

- **割り込みマスクテーブルへの割り込みマスク値登録**

vecttbl.src

```
_RESET_Vectors:
;<<VECTOR DATA START (POWER ON RESET)>>
       ;H'000 Power On Reset (H-UDI RESET)
       .data.l       _PowerON_Reset
;<<VECTOR DATA END (POWER ON RESET)>>
; Reserved
       .datab.l       8,H'00000000


       .section    INTTBL,data
       .export     _INT_Vectors

     ～(省略)～
       .export     _INT_MASK
_INT_MASK:
     ～(省略)～
       ;H'580 TMU0
       .data.b       H'10 ←TM0 割り込み関数実行中、割り込み優先順位がレベル 1 以下の割り込みをマスク
       ;H'5A0 TMU1
       .data.b       H'20 ←TM1 割り込み関数実行中、割り込み優先順位がレベル 2 以下の割り込みをマスク
          ⋮
```

- **割り込みベクタテーブルへの割り込み関数登録**

vecttbl.src

```
_RESET_Vectors:
;<<VECTOR DATA START (POWER ON RESET)>>
       ;H'000 Power On Reset (H-UDI RESET)
       .data.l       _PowerON_Reset
;<<VECTOR DATA END (POWER ON RESET)>>
; Reserved
       .datab.l       8,H'00000000


       .section    INTTBL,data
       .export     _INT_Vectors

_INT_Vectors:
     ～(省略)～
       ;H'580 TMU0
       .data.l       _INT_TMU0_TUNI0 ←TM0 割り込み関数
       ;H'5A0 TMU1
       .data.l       _INT_TMU1_TUNI1 ←TM1 割り込み関数
          ⋮
```

- **割り込み関数を C/C++言語で記述**

intprg.src

```
    .include    "vect.inc"
    .section    IntPRG, code

    ;H'040 Data TLB miss exception(read)
_INT_TLB_MISS_READ_EXP：
：
     ～(省略)～
    ;H'580 TMU0
;_INT_TMU0_TUNI0 ←";(セミコロン)"で、TM0 割り込み関数をコメントアウトまたは削除
    ;H'5A0 TMU1
;_INT_TMU1_TUNI1 ←";(セミコロン)"で、TM1 割り込み関数をコメントアウトまたは削除
          ⋮
```

main.c

```
void INT_TMU0_TUNI0(void) ←TM0 割り込み関数を作成
{
     ～(省略)～ ←C/C++言語で、割り込み要求のクリアと、その他実行する処理を記述
}


void INT_TMU1_TUNI1(void) ←TM1 割り込み関数を作成
{
     ～(省略)～ ←C/C++言語で、割り込み要求のクリアと、その他実行する処理を記述
}
```

改訂記録	SH7450 グループ/SH7451 グループ タイマユニット(TMU)割り込みサンプルコード

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2012.2.1	—	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. 未使用端子の処理

   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。

   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。

2. 電源投入時の処置

   【注意】電源投入時は，製品の状態は不定です。

   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。

   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。

   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. リザーブアドレスのアクセス禁止

   【注意】リザーブアドレスのアクセスを禁止します。

   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

4. クロックについて

   【注意】リセット時は、クロックが安定した後、リセットを解除してください。

   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。

   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

5. 製品間の相違について

   【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認下さい。

   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

RENESAS

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

Colophon 1.0