お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

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。

サンプルプログラム関数仕様　【タイマ制御編】
使用法説明書(V850ES/Kx1,V850ES/Kx1+)

本説明書はV850ES/Kx1，V850ES/Kx1+用のタイマ制御を行うサンプルソフトウェアの動作を説明したものです。

ご注意

本ソフトウェアはあくまで参考用のソフトであり、当社がこの動作を保証するものではありません。

本ソフトウェアを使用する場合、お客様のセット上で十分な評価の上ご使用いただきますようお願いいたします。

目次

関数一覧表は以下のように構成されています。

テーマ　（　ハードウェア略号　）

【機　　能】	テーマの説明
【関　数　名】	サンプル関数の名前
【引　　数】	引数の型と概要
【処 理 内 容】	サンプル関数の処理内容
【起 動 方 法】	関数の呼び出し条件
【使 用 S F R】	レジスタ名と設定内容
【 call 関 数】	呼び出し関数の名前と機能
【変　　数】	サンプル関数での使用変数の型、名前、概要
【割 り 込 み】	関数名
【割り込み 要 因】	名称
【使 用 ハ ー ド】	その他使用リソース
【フ ァ イ ル 名】	対応するサンプルプログラム・ファイル名
【注 意 事 項】	関数使用上の注意。使い方

割り込み関数

【関　数　名】	
【概　　要】	処理の目的
【要　　因】	指定無し
【 call 関 数】	無し
【変　　数】	変数名　機能
【フ ァ イ ル 名】	対応するサンプルプログラム・ファイル名
【注 意 事 項】	無し

16bitタイマ0　インターバルタイマ (TM0n)

KF1 : n = 0-1

KG1 : n = 0-3

KJ1 : n = 0-5

【機　　能】	16ビット・タイマ・キャプチャ/ コンペア・レジスタ000(CR000)に、あらかじめ設定したカウント値とTM00の設定値が一致した時、カウンタのクリア＆スタートと同時に割り込み要求信号を発生します。	
【関　数　名】	timer0_interval_count	
【引　　数】	unsigned int set_CR000 unsigned char set_PRM00	コンペアレジスタCR000の設定 カウントクロックPRM001,PRM000の設定
【処 理 内 容】	・10ms毎に割り込み関数をコールします。	
【起 動 方 法】	・タイマ00が動作停止時コールします。 ・メインクロックを16MHzに設定してください。 ・tm0_intvl_st関数のコールによりタイマを動作して下さい。	
【使 用 S F R】	TMIC00 CRC00 CR000 CR001 PRM00 TMC00	INTTM000割り込みのマスクとレベルの選択 キャプチャ / コンペアレジスタの動作を制御 インターバル時間の設定 ユーザー未使用 カウントクロック、TI000、TI001の有効エッジを設定 TM00のクリアモードと出力タイミング、オーバーフローの検出の設定
【 call 関数】	main	メイン関数
【変　　数】	無し	
【割 り 込 み】	timer0_interval_count	
【割り込み 要 因】	INTTM000	
【フ ァ イ ル 名】	timer0_interval_count\timer0_1.c , timer0_interval_count\MAIN.C	
【注 意 事 項】	・PRM0nレジスタの内容は、チャネル毎に内容が異なるのでカウントクロックの設定を行う際は注意してください。 ・PRM0nレジスタは、必ずタイマ動作を停止してからデータを設定してください。 ・タイマ動作を停止する時には、TMC00レジスタのビット3、2に00をセットしてください。 ・CR000の値を変える場合は、いったんタイマ動作を停止させたのちに行ってください。	

【関　数　名】	tm0_intvl_st
【引　　数】	無し
【処 理 内 容】	timer0_interval_countの起動関数です。
【起 動 方 法】	timer0_interval_count関数の後にコールしてください。
【使 用 S F R】	TMC00　　動作モード・クリアモードの選択
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	timer0_interval_count\timer0_1.c
【注 意 事 項】	無し

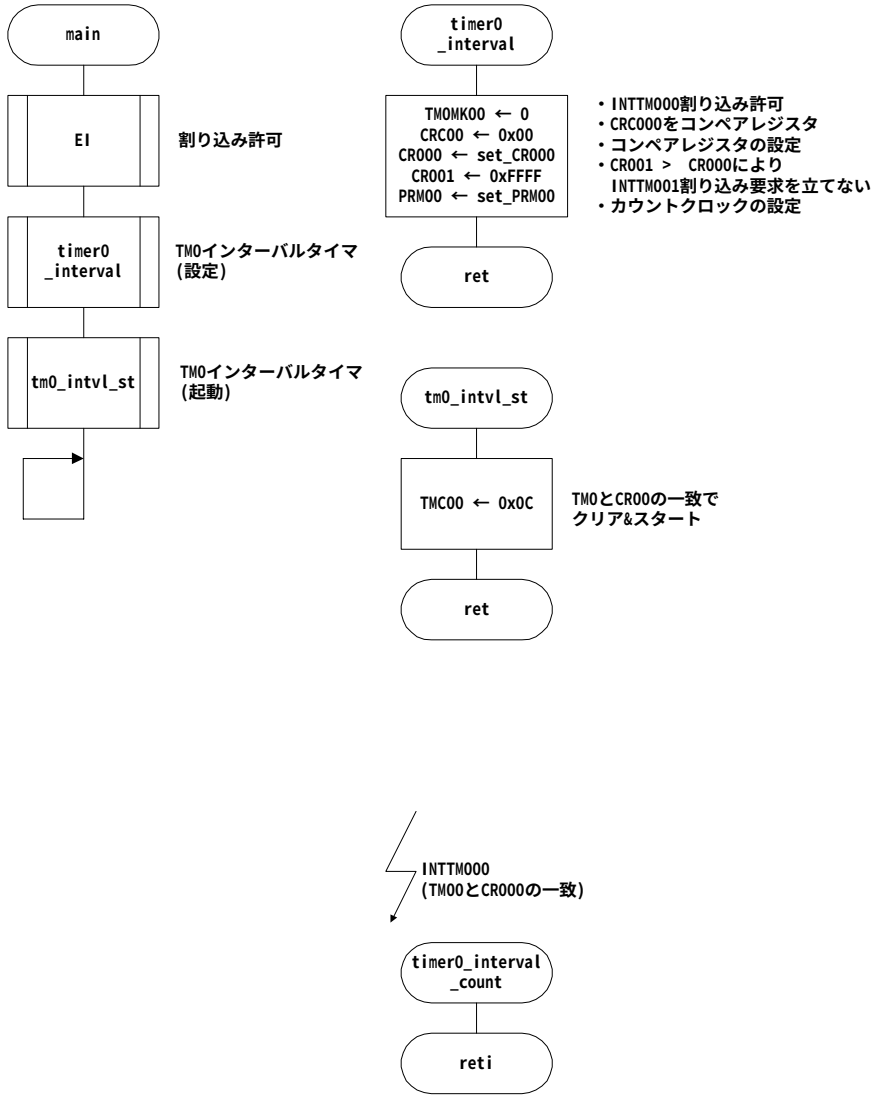
割り込み関数

【関　数　名】	timer0_interval_count
【概　　要】	ユーザー定義
【要　　因】	INTTM000　　TM00とCR000の一致
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	timer0_interval_count\timer0_1.c
【注 意 事 項】	無し

設定時間は以下の式により決定します。（カウントクロック= 2MHz の場合）

設定時間 = (コンペア・レジスタ + 1) ×カウントクロック周期

例） 10ms = (19999 + 1) × 500ns


main
EI
割り込み許可
timer0
_interval
TM0インターバルタイマ
(設定)
tm0_intvl_st
TM0インターバルタイマ
(起動)
timer0
_interval
TM0MK00 ← 0
CRC00 ← 0x00
CR000 ← set_CR000
CR001 ← 0xFFFF
PRM00 ← set_PRM00
・INTTM000割り込み許可
・CRC000をコンペアレジスタ
・コンペアレジスタの設定
・CR001 > CR000により
INTTM001割り込み要求を立てない
・カウントクロックの設定
ret
tm0_intvl_st
TMC00 ← 0x0C
TM0とCR00の一致で
クリア&スタート
ret
INTTM000
(TM00とCR000の一致)
timer0_interval
_count
reti

16bitタイマ0　PPG出力 (TM0n)

KF1 : n = 0-1

KG1 : n = 0-3

KJ1 : n = 0-5

【機　　能】	16ビット・キャプチャ・レジスタCR000に設定したカウント値を1周期 16ビット・キャプチャ・レジスタCR001に設定したカウント値をデューティー比とする矩形波をTO00端子から出力します。	
【関　数　名】	timer0_ppg	
【引　　数】	unsigned int　tm0_ppg_CR000 unsigned int　tm0_ppg_CR001 unsigned char tm0_ppg_PRM00	コンペアレジスタCR000の設定 コンペアレジスタCR001の設定 カウントクロックPRM001,PRM000の設定
【処　理　内　容】	TO00端子からパルス周期50μs、デューティー比1:5の矩形波を出力します。	
【起　動　方　法】	・コールする前にTO00端子の設定を行ってください。 ・メインクロックを16MHzに設定してください。 ・tm0_ppg_st関数のコールによりタイマを動作して下さい。	
【使　用 S　F　R】	CRC00 CR000 CR001 TOC00 PRM00	キャプチャ　/　コンペアレジスタの動作を制御 パルスの周期 デューティー比 16ビット・タイマ・カウンタ0nの出力を制御 カウントクロック、TI000、TI001の有効エッジを設定
【 call　　関数】	main	メイン関数
【変　　数】	無し	
【注　意　事　項】	・PRM0nレジスタの内容は、チャネル毎に内容が異なります。 ・TO00端子の設定は、timer0_ppg_ini関数により行います。 ・PRM0nレジスタは、必ずタイマ動作を停止してからデータを設定してください。	

【関　数　名】	tm0_ppg_st
【引　　数】	無し
【処 理 内 容】	timer0_ppgの起動関数です。
【起 動 方 法】	timer0_ppg関数の後にコールしてください。
【使 用 S F R】	TMC00　　動作モード・クリアモードの選択
【 call　関数】	無し
【変　　数】	無し
【注 意 事 項】	無し

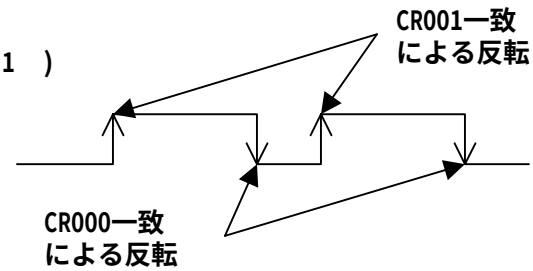
設定時間は以下の式により決定します。 （カウントクロック= 2MHzの場合 ）

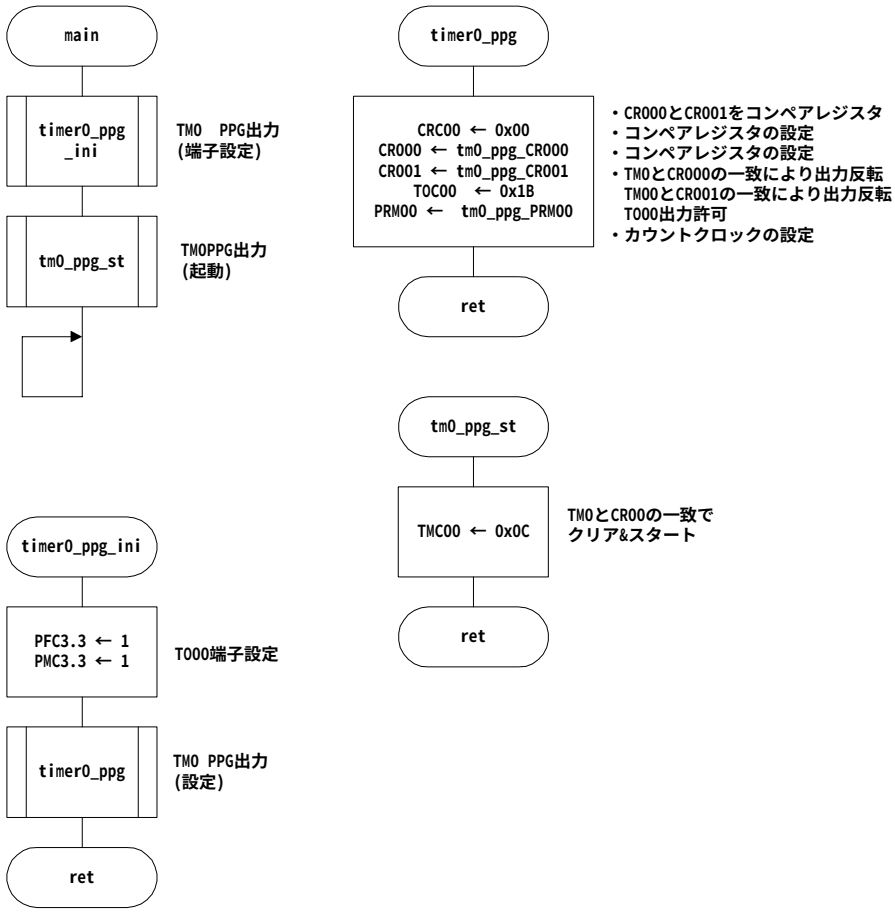
パルス周期　= (コンペア・レジスタ + 1) ×カウントクロック周期

例）50us　= (　99 + 1　)　× 500ns

デューティー比 = (CR001 + 1　)　/　(　CR000 ＋ 1　)

0.2　= (　19 + 1　)　/　(　99 + 1　)




main
timer0_ppg_ini
TM0　PPG出力
(端子設定)
tm0_ppg_st
TM0PPG出力
(起動)
timer0_ppg_ini
PFC3.3 ← 1
PMC3.3 ← 1
TO00端子設定
timer0_ppg
TM0 PPG出力
(設定)
ret
timer0_ppg
CRC00 ← 0x00
CR000 ← tm0_ppg_CR000
CR001 ← tm0_ppg_CR001
TOC00 ← 0x1B
PRM00 ← tm0_ppg_PRM00
・CR000とCR001をコンペアレジスタ
・コンペアレジスタの設定
・コンペアレジスタの設定
・TM0とCR000の一致により出力反転
TM00とCR001の一致により出力反転
TO00出力許可
・カウントクロックの設定
ret
tm0_ppg_st
TMC00 ← 0x0C
TM0とCR00の一致で
クリア&スタート
ret

16bitタイマ0　　パルス幅測定　(TM0n)　KF1 : n = 0-1

KG1 : n = 0-3

KJ1　: n = 0-5

【機　　能】	フリー・ランニング・モードで動作時に、TI000端子にPRM00レジスタで設定した有効エッジを入力すると、タイマのカウント値をキャプチャレジスタに取りこみます。図1のタイミングでタイマカウントを取りこみ、入力する信号のパルス幅を測定します。
【関　数　名】	timer0_pulse
【引　　数】	unsigned char tm0_pul_PRM00　　カウントクロックの設定 有効エッジの選択
【処 理 内 容】	フリー・ランニング・カウンタとキャプチャレジスタを2本(CR000、CR001)によりパルス幅測定を行います。
【起 動 方 法】	・TI000端子の設定を行ってください。 ・メインクロックを16MHzに設定してください。 ・tm0_pul_st関数のコールによりタイマを動作して下さい。
【使 用 S F R】	TM0IC01　　INTTM001割り込みのマスクとレベルの選択 PRM00　　カウントクロック、TI000、TI001の有効エッジを設定 CRC00　　キャプチャ / コンペアレジスタの動作を制御
【 call　関数】	main　　メイン関数
【変　　数】	無し
【割 り 込 み】	cal_pul
【割り込み 要 因】	INTTM001
【フ ァ イ ル 名】	timer0_ppg\timer0_2.c , timer0_ppg\MAIN.C
【注 意 事 項】	・PRM0nレジスタの内容は、チャネル毎に内容が異なります。 ・TI000端子の設定は、timer0_pulse_ini関数により行います。 ・PRM0nレジスタは、必ずタイマ動作を停止してからデータを設定してください。

【関　数　名】	tm0_pul_st
【引　　数】	無し
【処 理 内 容】	timer0_pulseの起動関数です。
【起 動 方 法】	timer0_pulse関数の後にコールしてください。
【使 用 S F R】	TMC00　　動作モード・クリアモードの選択
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	timer0_ppg\timer0_2.c
【注 意 事 項】	無し

割り込み関数

【関　数　名】	cal_pul
【概　　要】	ユーザー定義
【要　　因】	INTTM001　　TM00とCR001の一致
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	timer0_ppg\timer0_2.c
【注 意 事 項】	無し

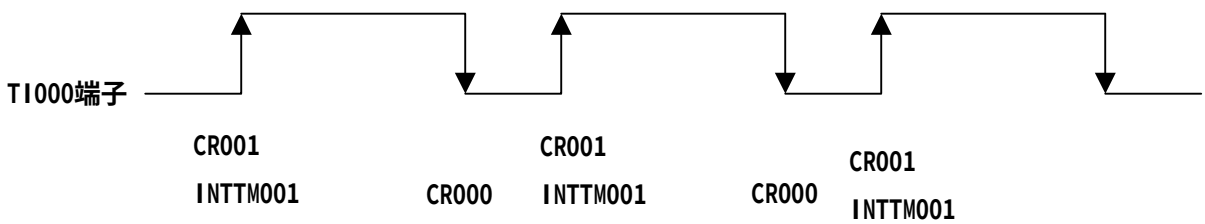
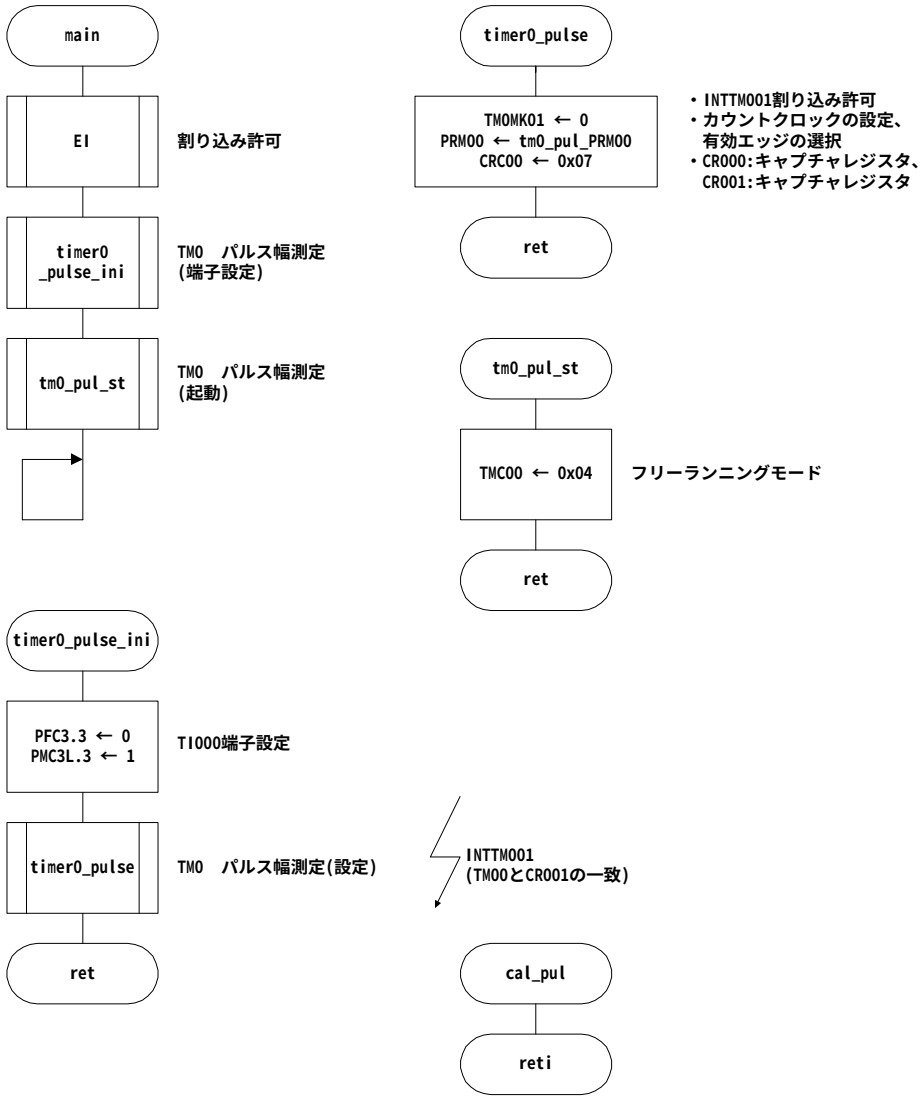


図1.　タイマカウントの取りこみタイミング


main
EI
割り込み許可
timer0
_pulse_ini
TM0　パルス幅測定
(端子設定)
tm0_pul_st
TM0　パルス幅測定
(起動)
timer0_pulse_ini
PFC3.3 ← 0
PMC3L.3 ← 1
TI000端子設定
timer0_pulse
TM0　パルス幅測定(設定)
ret
timer0_pulse
TMOMK01 ← 0
PRM00 ← tm0_pul_PRM00
CRC00 ← 0x07
・INTTM001割り込み許可
・カウントクロックの設定、
有効エッジの選択
・CR000:キャプチャレジスタ、
CR001:キャプチャレジスタ
ret
tm0_pul_st
TMC00 ← 0x04
フリーランニングモード
ret
INTTM001
(TM00とCR001の一致)
cal_pul
reti

16bitタイマ0　方形波出力　(TMOn)

KF1 : n = 0-1

KG1 : n = 0-3

KJ1 : n = 0-5

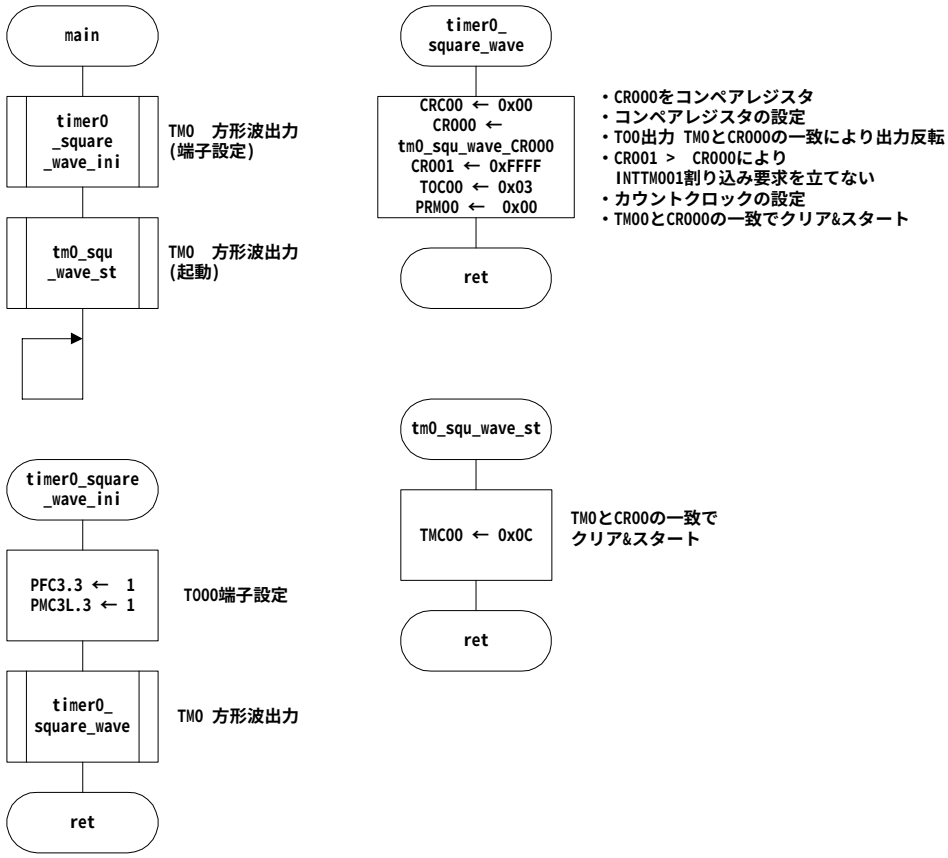
【機　　能】	16ビット・タイマ・キャプチャ/コンペアレジスタ000(CR0n0) とタイマカウントが一致によりT000端子の出力が反転により任意の周波数の方形波出力が可能です。	
【関　数　名】	timer0_square_wave	
【引　　数】	unsigned short int　tm0_squ_wave_CR000	コンペアレジスタ CR000の設定
	unsigned char　　tm0_squ_wave_PRM00	カウントクロックの設定
【処 理 内 容】	T000端子から、周期1msの方形波を出力します。	
【起 動 方 法】	・タイマ00が動作停止時コールします。 ・コールする前にT000端子の設定を行ってください。 ・メインクロックを16MHzに設定してください。 ・tm0_squ_wave_st関数のコールによりタイマを動作して下さい。	
【使 用 S F R】	CRC00　　キャプチャ　/　コンペアレジスタの動作を制御 CR000　　インターバル時間の設定 CR001　　ユーザー未使用 TOC00　　16ビット・タイマ・カウンタ0nの出力を制御 PRM00　　カウントクロック、TI000、TI001の有効エッジを設定 TMC00　　TM00のクリアモードと出力タイミング、 オーバーフローの検出の設定	
【 call　関数】	main　　メイン関数	
【変　　数】	無し	
【フ ァ イ ル 名】	timer0_square_wave\timer0_4.c , timer0_square_wave\MAIN.C	
【注 意 事 項】	・ T000端子の設定は、timer0_square_wave_ini関数により行っています。 ・タイマ00の動作を停止させるには、TMC00レジスタのビット2、3それぞれに0を設定してください。	

【関　数　名】	tm0_squ_wave_st
【引　　数】	無し
【処 理 内 容】	timer0_square_waveの起動関数です。
【起 動 方 法】	timer0_square_wave関数の後にコールしてください。
【使 用 S F R】	TMC00　　動作モード・クリアモードの選択
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	timer0_square_wave\timer0_4.c
【注 意 事 項】	無し

設定時間は以下の式により決定します。（カウントクロック = 8MHｚ動作の場合　）

パルス周期　=（ コンペア・レジスタ + 1 ）　×　カウントクロック周期　×　2

例）1 ms　=（ 3999 + 1 ）　×　　125ns　×　2


main
timer0
_square
_wave_ini
TM0　方形波出力
(端子設定)
tm0_squ
_wave_st
TM0　方形波出力
(起動)
timer0_square
_wave_ini
PFC3.3 ← 1
PMC3L.3 ← 1
TO00端子設定
timer0_
square_wave
TM0 方形波出力
ret
timer0_
square_wave
CRC00 ← 0x00
CR000 ←
tm0_squ_wave_CR000
CR001 ← 0xFFFF
TOC00 ← 0x03
PRM00 ← 0x00
・CR000をコンペアレジスタ
・コンペアレジスタの設定
・TO0出力 TM0とCR000の一致により出力反転
・CR001 > CR000により
INTTM001割り込み要求を立てない
・カウントクロックの設定
・TM00とCR000の一致でクリア&スタート
ret
tm0_squ_wave_st
TMC00 ← 0x0C
TM0とCR00の一致で
クリア&スタート
ret

16bitタイマ0　外部イベントカウンタ　(TM0n)　　KF1 : n = 0-1

KG1 : n = 0-3

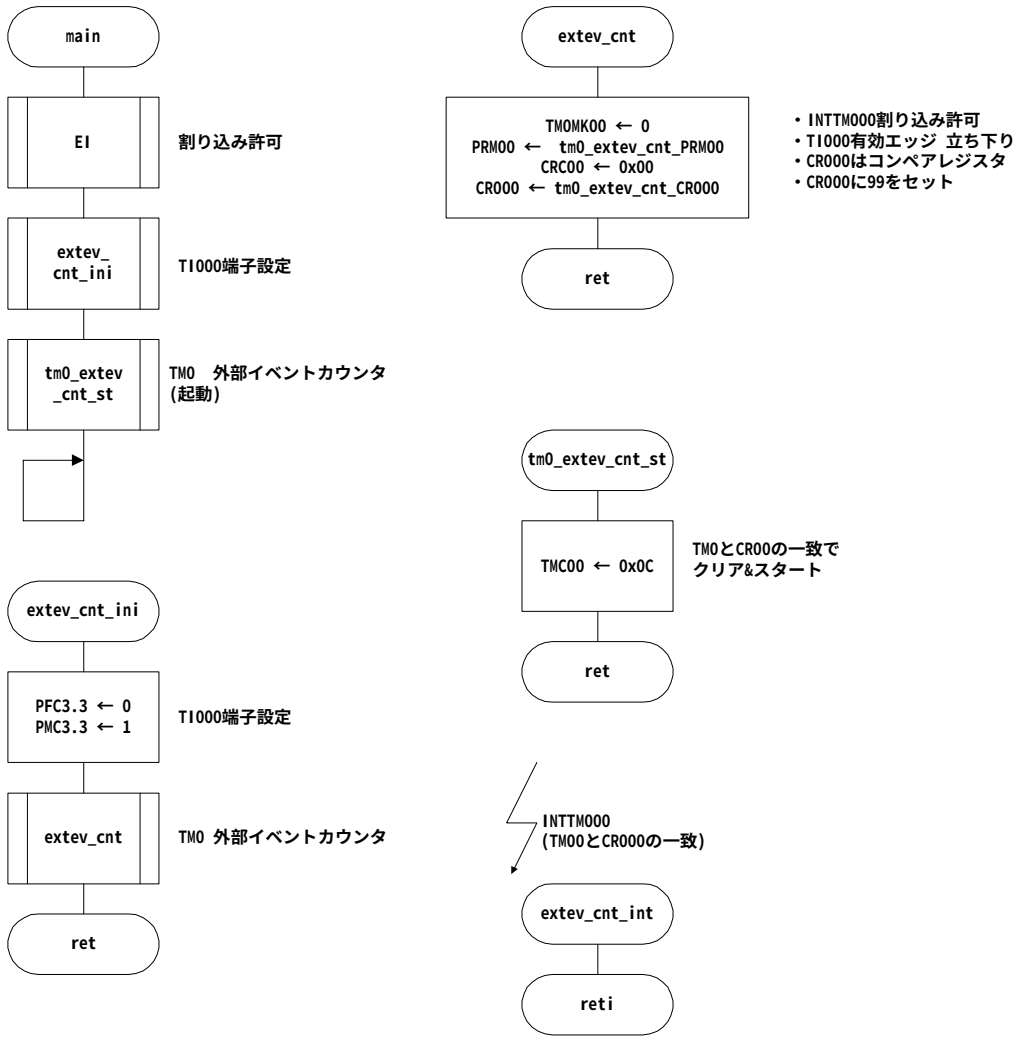
KJ1 : n = 0-5

【機　　能】	TI000端子に入力される外部からのクロックパルス数を16ビット・タイマ・カウンタ00で（TM00）でカウントします。 PRMレジスタで指定した有効エッジの入力により、タイマカウントがインクリメントされCR000の値と一致により、タイマカウントがクリアされ割り込み要求信号(INTTM000)を発生します。
【関　数　名】	extev_cnt
【引　　数】	unsigned char tm0_extev_cnt_PRM00　　有効エッジの選択 unsigned short int tm0_extev_cnt_CR000　　有効エッジの検出数
【処 理 内 容】	TI000端子からの入力波形を取りこみ、有効エッジを100回入力した時に割り込みを発生します。
【起 動 方 法】	・タイマ00が動作停止時コールします。 ・コールする前にTI000端子の設定を行ってください。 ・tm0_extev_cnt_st関数のコールによりタイマを動作して下さい。
【使 用 S F R】	TM0IC00　　INTTM000割り込みのマスクとレベルの選択 PRM00　　カウントクロック、TI000、TI001の有効エッジを設定 CRC00　　キャプチャ　/　コンペアレジスタの動作を制御 CR000　　インターバル時間の設定 TMC00　　TM00のクリアモードと出力タイミング、 オーバーフローの検出の設定
【 call　関数】	main　　メイン関数
【変　　数】	無し
【割 り 込 み】	extev_cnt_int
【割り込み 要 因】	INTTM000
【フ ァ イ ル 名】	extev_cnt\timer0_5.c , extev_cnt\MAIN.C
【注 意 事 項】	・ T000端子の設定は、 extev_cnt_ini関数により行います。 ・タイマ00の動作を停止させるには、TMC00レジスタのビット2、3それぞれに0を設定してください。

【関　数　名】	tm0_extev_cnt_st
【引　　数】	無し
【処 理 内 容】	extev_cntの起動関数です。
【起 動 方 法】	extev_cnt関数の後にコールしてください。
【使 用 S F R】	TMC00　　動作モード・クリアモードの選択
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	extev_cnt\timer0_5.c
【注 意 事 項】	無し

割り込み関数

【関　数　名】	extev_cnt_int
【概　　要】	ユーザー定義
【要　　因】	INTTM000　　TM00とCR000の一致
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	extev_cnt\timer0_5.c
【注 意 事 項】	無し


main
EI
割り込み許可
extev_
cnt_ini
TI000端子設定
tm0_extev
_cnt_st
TM0　外部イベントカウンタ
(起動)
extev_cnt_ini
PFC3.3 ← 0
PMC3.3 ← 1
TI000端子設定
extev_cnt
TM0 外部イベントカウンタ
ret
extev_cnt
TM0MK00 ← 0
PRM00 ← tm0_extev_cnt_PRM00
CRC00 ← 0x00
CR000 ← tm0_extev_cnt_CR000
・INTTM000割り込み許可
・TI000有効エッジ 立ち下り
・CR000はコンペアレジスタ
・CR000に99をセット
ret
tm0_extev_cnt_st
TMC00 ← 0x0C
TM0とCR00の一致で
クリア&スタート
ret
INTTM000
(TM00とCR000の一致)
extev_cnt_int
reti

8bitタイマ5　インターバルタイマ　(TM5n)　　KF1 / KG1 / KJ1　: n = 0-1

【機　　能】	8ビット・タイマ・コンペア・レジスタ50(CR50)に、あらかじめ設定したカウント値をインターバルとし、繰り返し割り込みを発生します。	
【関　数　名】	timer5_interval	
【引　　数】	unsigned char tm5_intvl_CR50 unsigned char tm5_intvl_TCL50	コンペアレジスタの設定 カウントクロックの設定
【処 理 内 容】	1ms毎に割り込み関数をコールします。	
【起 動 方 法】	・タイマ50を動作停止してください。 ・メインクロックを16MHzに設定してください。 ・tm5_intvl_st関数のコールによりタイマを動作して下さい。	
【使 用 S F R】	TM5IC0 CR50 TCL50 TMC50	INTTM50割り込みのマスクとレベルの設定 インターバル時間の設定 カウントクロックの選択 ・8ビット・タイマ・カウンタ50(TM50)のカウント動作制御 ・TM50レジスタの動作モードの選択 ・タイマ出力F/F(フリップフロップ)の状態設定 ・タイマ出力F/Fの制御またはPWM(フリー・ランニング)モード時のアクティブレベルの設定 ・タイマ出力の制御
【 call　関数】	main	メイン関数
【変　　数】	無し	
【割 り 込 み】	timer5_count	
【割り込み 要 因】	INTTM50	
【フ ァ イ ル 名】	timer5_interval\timer5_1.c , timer5_interval\MAIN.C	
【注 意 事 項】	・動作を停止するときは、TCE50ビットを0にしてからTCE51ビットを0にしてください。	

【関　数　名】	tm5_intvl_st
【引　　数】	無し
【処 理 内 容】	timer5_intervalの起動関数です。
【起 動 方 法】	timer5_interval関数の後にコールしてください。
【使 用 S F R】	TCE50　　8ビットタイマ50の動作制御
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	timer5_interval\timer5_1.c
【注 意 事 項】	無し

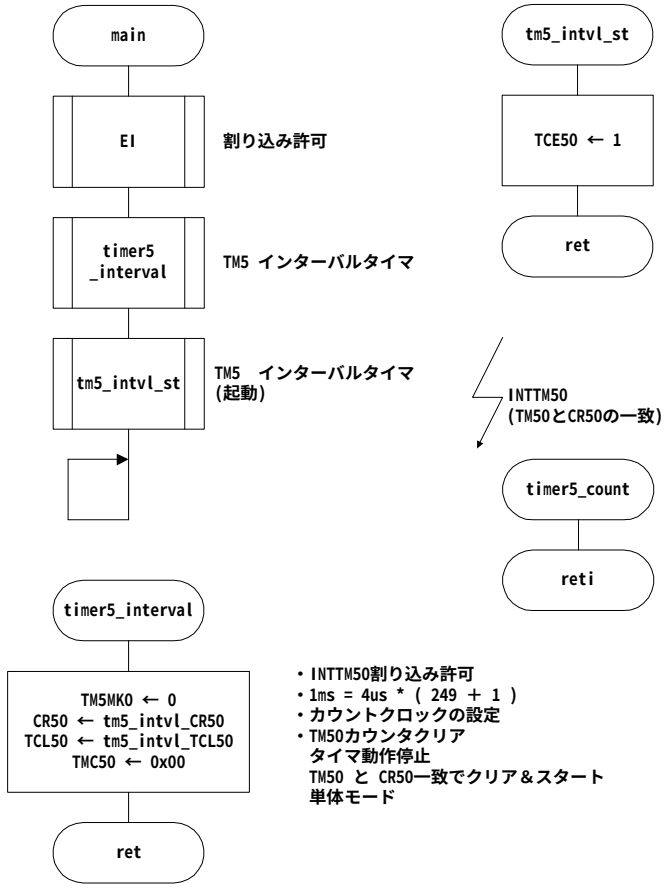
割り込み関数

【関　数　名】	timer5_count
【概　　要】	ユーザー定義
【要　　因】	INTTM50　　TM50とCR50の一致
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	timer5_interval\timer5_1.c
【注 意 事 項】	無し

設定時間は以下の式により決定します。 （ カウントクロック = 250KHz の場合）

設定時間 = （コンペア・レジスタ + 1 ）×カウントクロック周期

1ms = (249 + 1) × 4μs



・INTTM50割り込み許可
・1ms = 4us * (249 ＋ 1)
・カウントクロックの設定
・TM50カウンタクリア
　タイマ動作停止
　TM50 と CR50一致でクリア＆スタート
　単体モード

16bitタイマ5　インターバルタイマ　(TM5n)　　KF1 / KG1 / KJ1　: n = 0-1

【機　　能】	TMC51レジスタのビット4に１をセットする事で8ビットタイマ50、51はカスケード接続となり、16ビットインターバルタイマとして動作します。	
【関　数　名】	timer5_cascade_interval	
【引　　数】	unsigned char tm5_cas_intvl_CR50	コンペアレジスタの設定(下位8ビット)
	unsigned char tm5_cas_intvl_CR51	コンペアレジスタの設定(上位8ビット)
	unsigned char tm5_cas_intvl_TCL50	カウントクロックの設定
【処 理 内 容】	10ms毎に割り込み関数をコールします。	
【起 動 方 法】	・タイマ50、51が動作停止時コールします。 ・メインクロックを16MHzに設定してください。 ・tm5_cas_intvl_st関数のコールによりタイマを動作して下さい。	
【使 用 S F R】	TM5IC0	INTTM50割り込みのマスクとレベルの設定
	CR50	インターバル時間の設定(下位8ビット)
	CR51	インターバル時間の設定(上位8ビット)
	TCL50	カウントクロックの選択
	TMC50	・8ビット・タイマ・カウンタ50(TM50)のカウント動作制御 ・TM50レジスタの動作モードの選択 ・タイマ出力F/F(フリップフロップ)の状態設定 ・タイマ出力F/Fの制御またはPWM(フリー・ランニング)モード時のアクティブレベルの設定 ・タイマ出力の制御
	TMC51	・8ビット・タイマ・カウンタ50(TM50)のカウント動作制御 ・TM50レジスタの動作モードの選択 ・単体モード/カスケード接続モードの選択 ・タイマ出力F/F(フリップフロップ)の状態設定 ・タイマ出力F/Fの制御またはPWM(フリー・ランニング)モード時のアクティブレベルの設定 ・タイマ出力の制御
【 call　関数】	main	メイン関数
【変　　数】	無し	
【割 り 込 み】	timer5_cascade_count	
【割り込み 要 因】	INTTM50	
【フ ァ イ ル 名】	timer5_cascade_interval\timer5_2.c,timer5_cascade_interval\MAIN.C	
【注 意 事 項】	・タイマ動作中にCR5nレジスタの値を書き換えないで下さい（n = 0,1 ） ・動作を停止するときは、TCE50ビットを0にしてからTCE51ビットを0にしてください。	

【関　数　名】	tm5_cas_intvl_st
【引　　数】	無し
【処 理 内 容】	timer5_cascade_intervalの起動関数です。
【起 動 方 法】	timer5_cascade_interval関数の後にコールしてください。
【使 用 S F R】	TCE50　8ビットタイマ50の動作制御 TCE51　8ビットタイマ51の動作制御
【 call　関数】	無し
【変　　数】	無し
【注 意 事 項】	無し

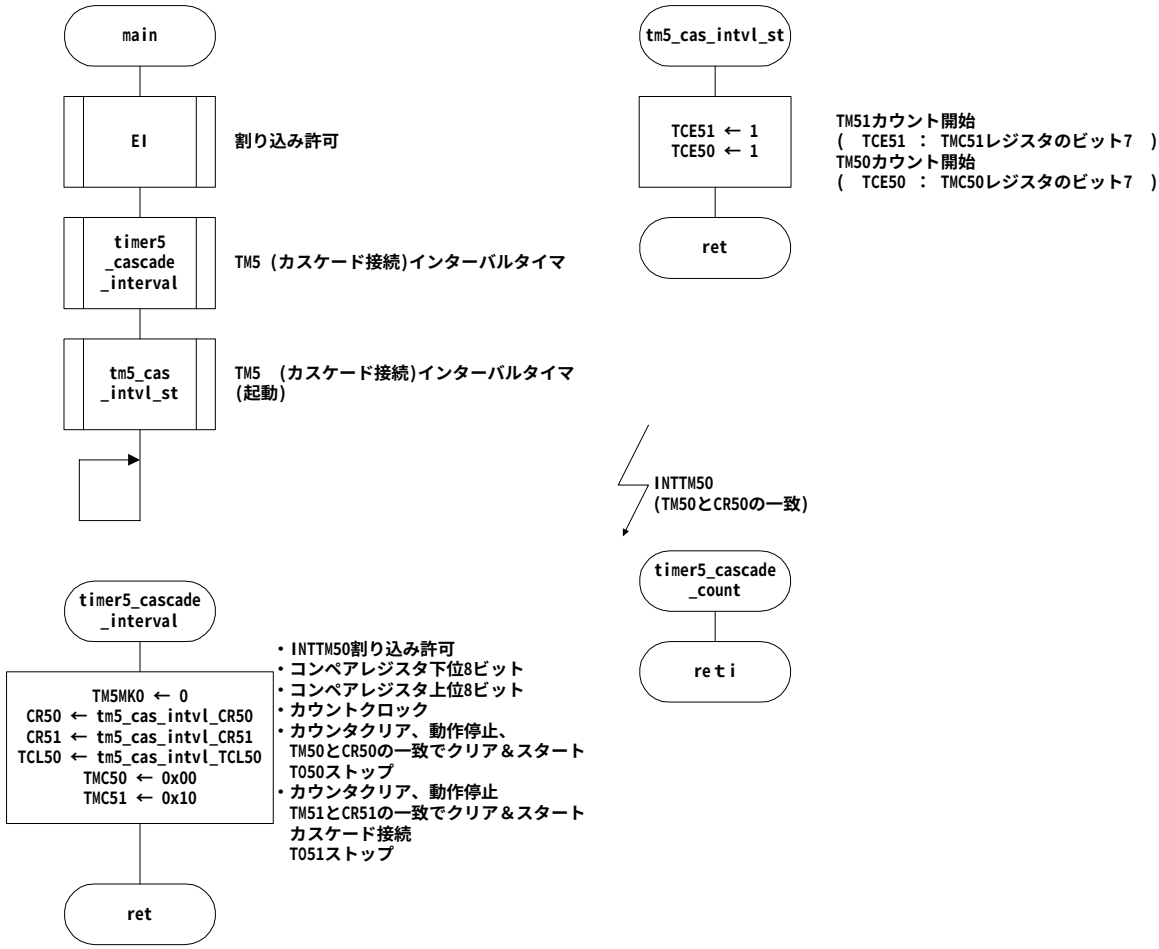
割り込み関数

【関　数　名】	timer5_cascade_count
【概　　要】	ユーザー定義
【要　　因】	INTTM50　TM50とCR50の一致
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	timer5_cascade_interval\timer5_2.c
【注 意 事 項】	無し

設定時間は以下の式により決定します。

設定時間　=　(コンペア・レジスタ + 1) ×カウントクロック周期

10ms　=　(2499 + 1)　× 4ns (　fxx = 16MHz)


main
EI
割り込み許可
timer5
_cascade
_interval
TM5 (カスケード接続)インターバルタイマ
tm5_cas
_intvl_st
TM5 (カスケード接続)インターバルタイマ
(起動)
tm5_cas_intvl_st
TCE51 ← 1
TCE50 ← 1
TM51カウント開始
(TCE51 : TMC51レジスタのビット7)
TM50カウント開始
(TCE50 : TMC50レジスタのビット7)
ret
INTTM50
(TM50とCR50の一致)
timer5_cascade
_count
reti
timer5_cascade
_interval
TM5MK0 ← 0
CR50 ← tm5_cas_intvl_CR50
CR51 ← tm5_cas_intvl_CR51
TCL50 ← tm5_cas_intvl_TCL50
TMC50 ← 0x00
TMC51 ← 0x10
・INTTM50割り込み許可
・コンペアレジスタ下位8ビット
・コンペアレジスタ上位8ビット
・カウントクロック
・カウンタクリア、動作停止、
TM50とCR50の一致でクリア＆スタート
T050ストップ
・カウンタクリア、動作停止
TM51とCR51の一致でクリア＆スタート
カスケード接続
T051ストップ
ret

8bitタイマ5　PWM出力　(TM5n)　　KF1 / KG1 / KJ1　: n = 0-1

【機　　能】	TMCレジスタのビット6に1をセットしPWM（フリー・ランニングモード）に設定します。8ビット・タイマ・コンペア・レジスタ50(CR50)に設定したデューティー比のパルスを出力します。	
【関　数　名】	timer5_pwm	
【引　　数】	unsigned char tm5_pwm_CR50　コンペアレジスタの設定 unsigned char tm5_pwm_TCL50　カウントクロックの設定	
【処 理 内 容】	TO50端子からデューティー比1：4の波形を出力します。	
【起 動 方 法】	・コールする前にTO50端子の設定を行ってください。 ・timer5_pwm_st関数のコールによりタイマを動作して下さい。	
【使 用 S F R】	TCL50	カウントクロックの選択
	CR50	インターバル時間の設定
	TMC50	・8ビット・タイマ・カウンタ50(TM50)のカウント動作制御 ・TM50レジスタの動作モードの選択 ・タイマ出力F/F(フリップフロップ)の状態設定 ・タイマ出力F/Fの制御またはPWM(フリー・ランニング)モード時のアクティブレベルの設定 ・タイマ出力の制御
【 call　関数】	main	メイン関数
【変　　数】	無し	
【フ ァ イ ル 名】	timer5_pwm\timer5_3.c , timer5_pwm\MAIN.C	
【注 意 事 項】	・ TO50端子の設定は、timer5_pwm_ini関数により行っています。	

【関　数　名】	timer5_pwm_st
【引　　数】	無し
【処 理 内 容】	timer5_pwmの起動関数です。
【起 動 方 法】	timer5_pwm関数の後にコールしてください。
【使 用 S F R】	TCE50　　8ビットタイマ50の動作制御
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	timer5_pwm\timer5_3.c
【注 意 事 項】	無し

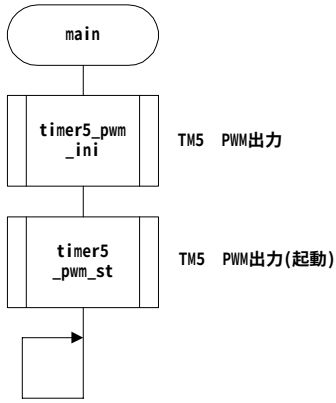
設定時間は以下の式により決定します。（カウントクロック = 4MHz ）

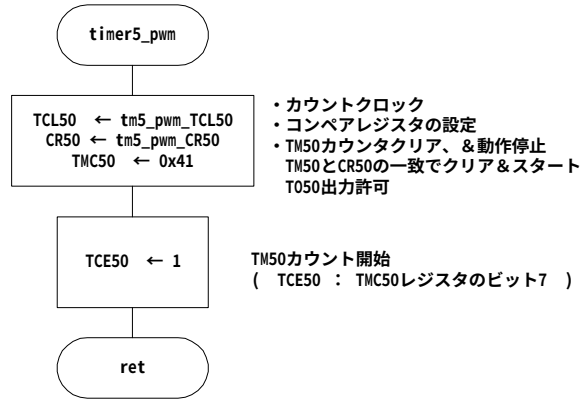
デューティー比 = (コンペア・レジスタ + 1) : 255

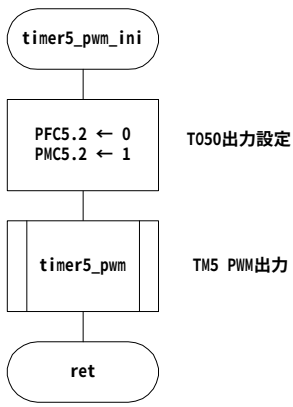
1 : 4 = (63 + 1) : 255

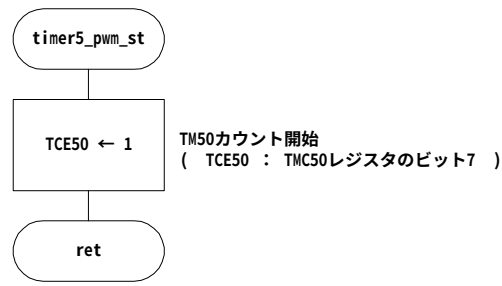
アクティブ時間 = (コンペア・レジスタ + 1) ×カウントクロック

16μs = 64 × 250ns


main
timer5_pwm
_ini
TM5　PWM出力
timer5
_pwm_st
TM5　PWM出力(起動)


timer5_pwm
TCL50　← tm5_pwm_TCL50
CR50 ← tm5_pwm_CR50
TMC50　← 0x41
・カウントクロック
・コンペアレジスタの設定
・TM50カウンタクリア、＆動作停止
TM50とCR50の一致でクリア＆スタート
TO50出力許可
TCE50　← 1
TM50カウント開始
(　TCE50　：　TMC50レジスタのビット7　)
ret


timer5_pwm_ini
PFC5.2 ← 0
PMC5.2 ← 1
TO50出力設定
timer5_pwm
TM5 PWM出力
ret


timer5_pwm_st
TCE50 ← 1
TM50カウント開始
(　TCE50　：　TMC50レジスタのビット7　)
ret

8bitタイマ5　方形波出力　(TM5n)　　KF1 / KG1 / KJ1　: n = 0-1

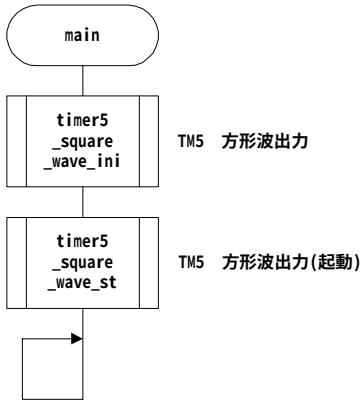
【機　　能】	TMC50のTOE50ビット = 1により8ビット・タイマ・コンペアレジスタ50(CR50)にあらかじめ設定したカウント値をインターバルとしてTO50端子の出力が反転します。これによって任意の周波数の方形波出力(デューティー比　50%)が可能です。	
【関　数　名】	timer5_square_wave	
【引　　数】	unsigned char tm5_squ_wav_CR50 unsigned char tm5_squ_wav_TCL50	コンペアレジスタの設定 カウントクロックの設定
【処 理 内 容】	・タイマ50を8ビット方形波出力として動作させる設定を行います。 TO50端子から、周期5μsの方形波を出力します。	
【起 動 方 法】	・タイマ50が動作停止時コールします。 ・コールする前にTO50端子の設定を行ってください。 ・メインクロックを16MHzに設定してください。 ・timer5_square_wave_st関数のコールによりタイマを動作して下さい。	
【使 用 S F R】	TCL50 CR50 TMC50	カウントクロックの選択 インターバル時間の設定 ・8ビット・タイマ・カウンタ50(TM50)のカウント動作制御 ・TM50レジスタの動作モードの選択 ・タイマ出力F/F(フリップフロップ)の状態設定 ・タイマ出力F/Fの制御またはPWM(フリー・ランニング)モード時のアクティブレベルの設定 ・タイマ出力の制御
【 call　関数】	main	メイン関数
【変　　数】	無し	
【ファイル名】	timer5_square_wave\timer5_4.c , timer5_square_wave\MAIN.C	
【注 意 事 項】	・ TO50端子の設定は、timer5_pwm_ini関数により行っています。	

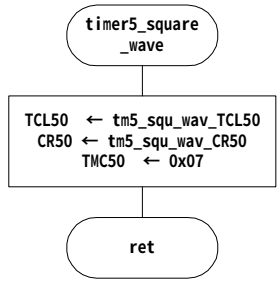
【関　数　名】	timer5_square_wave_st
【引　　数】	無し
【処 理 内 容】	timer5_square_waveの起動関数です。
【起 動 方 法】	timer5_square_wave関数の後にコールしてください。
【使 用 S F R】	TCE50　　8ビットタイマ50の動作制御
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	timer5_square_wave\timer5_4.c
【注 意 事 項】	無し

設定時間は以下の式により決定します。 （カウントクロック = 4MHz ）

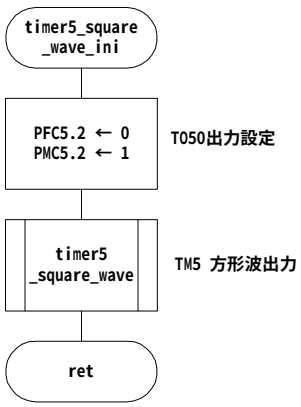
アクティブ時間　= (コンペア・レジスタ + 1) ×カウントクロック × 2

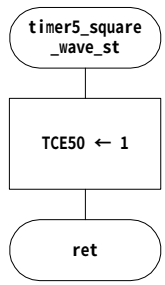
5μs = 10 × 250ns × 2





・カウントクロック
・コンペアレジスタの設定
・カウンタを0クリア後カウントストップ
TM5とCR50の一致でクリア&スタート
TO50反転許可
TO50出力許可





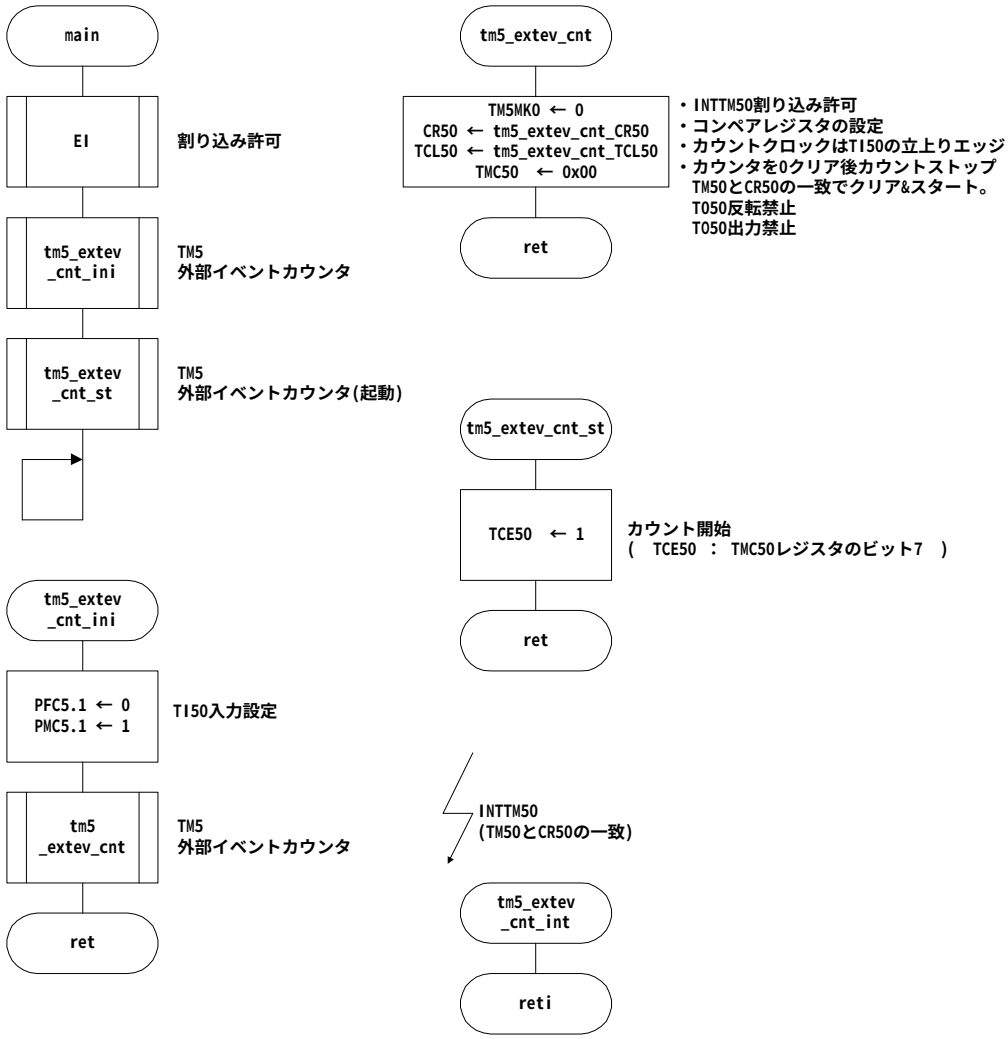
8bitタイマ5　外部イベントカウンタ　(TM5n)　　KF1 / KG1 / KJ1　: n = 0-1

【機　　能】	TI50端子に入力される外部からのクロックパルス数を8ビット・タイマ・カウンタ50で（TM50）でカウントするものです。 TI50端子にTCLレジスタで指定した有効エッジが入力されるたびに、TM50レジスタがインクリメントされCR50の値と一致した時に、TM50レジスタがクリアされ割り込み要求信号(INTTM50)を発生します。	
【関　数　名】	tm5_extev_cnt	
【引　　数】	unsigned char tm5_extev_cnt_CR50 unsigned char tm5_extev_cnt_TCL50	コンペアレジスタの設定 カウントクロックの設定
【処 理 内 容】	・ TI50端子からの入力波形を取りこみ、有効エッジを5回入力した時に割り込みを発生しフラグのセットを行います。	
【起 動 方 法】	・タイマ50が動作停止時コールします。 ・コールする前にTI50端子の設定を行ってください。 ・tm5_extev_cnt_st関数のコールによりタイマを動作して下さい。	
【使 用 S F R】	TM5IC0　INTTM50割り込みのマスクとレベルの設定 CR50　インターバル時間の設定(下位8ビット) TCL50　カウントクロックの選択 TMC50　・8ビット・タイマ・カウンタ50(TM50)のカウント動作制御 ・TM50レジスタの動作モードの選択 ・タイマ出力F/F(フリップフロップ)の状態設定 ・タイマ出力F/Fの制御またはPWM(フリー・ランニング)モード時のアクティブレベルの設定 ・タイマ出力の制御	
【 call　関数】	main　メイン関数	
【変　　数】	なし	
【割 り 込 み】	tm5_extev_cnt_int	
【割り込み 要 因】	INTTM50	
【フ ァ イ ル 名】	tm5_extev_cnt\timer5_5.c , tm5_extev_cnt\MAIN.C	
【注 意 事 項】	TI50端子の設定は、tm5_extev_cnt_ini関数により行っています。	

【関　数　名】	tm5_extev_cnt_st
【引　　数】	無し
【処 理 内 容】	tm5_extev_cntの起動関数です。
【起 動 方 法】	tm5_extev_cnt関数の後にコールしてください。
【使 用 S F R】	TCE50　　8ビットタイマ50の動作制御
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	tm5_extev_cnt\timer5_5.c
【注 意 事 項】	無し

割り込み関数

【関　数　名】	tm5_extev_cnt_int
【概　　要】	ユーザー定義
【要　　因】	INTTM50　TM50とCR50の一致
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	tm5_extev_cnt\timer5_5.c
【注 意 事 項】	無し


main
EI
割り込み許可
tm5_extev
_cnt_ini
TM5
外部イベントカウンタ
tm5_extev
_cnt_st
TM5
外部イベントカウンタ(起動)
tm5_extev
_cnt_ini
PFC5.1 ← 0
PMC5.1 ← 1
TI50入力設定
tm5
_extev_cnt
TM5
外部イベントカウンタ
ret
tm5_extev_cnt
TM5MK0 ← 0
CR50 ← tm5_extev_cnt_CR50
TCL50 ← tm5_extev_cnt_TCL50
TMC50 ← 0x00
・INTTM50割り込み許可
・コンペアレジスタの設定
・カウントクロックはTI50の立上りエッジ
・カウンタを0クリア後カウントストップ
TM50とCR50の一致でクリア&スタート。
T050反転禁止
T050出力禁止
ret
tm5_extev_cnt_st
TCE50 ← 1
カウント開始
(TCE50 ： TMC50レジスタのビット7)
ret
INTTM50
(TM50とCR50の一致)
tm5_extev
_cnt_int
reti

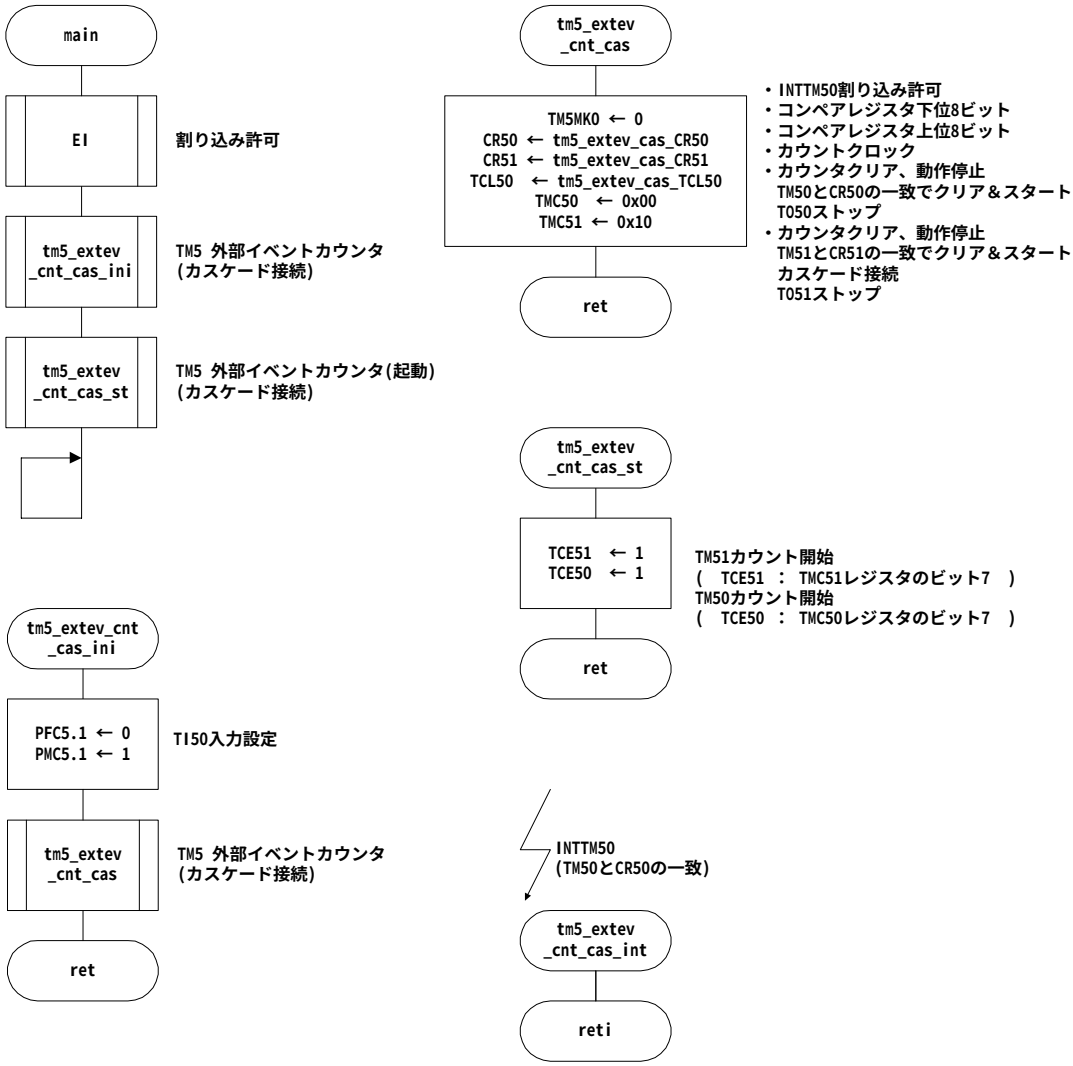
16bitタイマ5　外部イベントカウンタ　(TM5n)　　KF1 / KG1 / KJ1　: n = 0-1

【機　　能】	TMC51レジスタのTMC514ビットに“1”をセットによりTI50端子に入力されるクロックパルス数を16ビット・タイマ・カウンタ5で（TM5）でカウントします。TI50端子にTCL50レジスタで指定した有効エッジが入力されるたびに、TM5レジスタがインクリメントされCR50の値と一致した時に、TM50がクリアされ割り込み要求信号(INTTM50)を発生します。	
【関　数　名】	tm5_extev_cnt_cas	
【引　　数】	unsigned char tm5_extev_cas_CR50	コンペアレジスタの設定(下位8ビット)
	unsigned char tm5_extev_cas_CR51	コンペアレジスタの設定(上位8ビット)
	unsigned char tm5_extev_cas_TCL50	カウントクロックの設定
【処　理　内　容】	・ TI50端子からの入力波形を取りこみ、有効エッジを1000回入力した時に割り込みを発生しフラグのセットを行います。	
【起　動　方　法】	・タイマ50が動作停止時コールします。 ・コールする前にTI50端子の設定を行ってください。 ・tm5_extev_cnt_cas_st関数のコールによりタイマを動作して下さい。	
【使　用 S　F　R】	TM5IC0	INTTM50割り込みのマスクとレベルの設定
	CR50	インターバル時間の設定(下位8ビット)
	CR51	インターバル時間の設定(上位8ビット)
	TCL50	カウントクロックの選択
	TMC50	・8ビット・タイマ・カウンタ50(TM50)のカウント動作制御 ・TM50レジスタの動作モードの選択 ・タイマ出力F/F(フリップフロップ)の状態設定 ・タイマ出力F/Fの制御またはPWM(フリー・ランニング)モード時のアクティブレベルの設定 ・タイマ出力の制御
	TMC51	・8ビット・タイマ・カウンタ50(TM50)のカウント動作制御 ・TM50レジスタの動作モードの選択 ・単体モード/カスケード接続モードの選択 ・タイマ出力F/F(フリップフロップ)の状態設定 ・タイマ出力F/Fの制御またはPWM(フリー・ランニング)モード時のアクティブレベルの設定 ・タイマ出力の制御
【 call　　関数】	main	メイン関数
【変　　数】	無し	
【割　り　込　み】	tm5_extev_cnt_cas_int	
【割り込み 要 因】	INTTM50	
【フ ァ イ ル 名】	tm5_extev_cnt_cas\timer5_6.c , tm5_extev_cnt_cas\MAIN.C	
【注　意　事　項】	TI50端子の設定は、tm5_extev_cnt_cas_ini関数により行っています。	

【関　数　名】	tm5_extev_cnt_cas_st
【引　　数】	無し
【処 理 内 容】	tm5_extev_cnt_casの起動関数です。
【起 動 方 法】	tm5_extev_cnt_cas関数の後にコールしてください。
【使 用 S F R】	TCE50　8ビットタイマ50の動作制御 TCE51　8ビットタイマ51の動作制御
【 call　関数】	無し
【変　　数】	無し
【注 意 事 項】	無し

割り込み関数

【関　数　名】	tm5_extev_cnt_cas_int
【概　　要】	ユーザー定義
【要　　因】	INTTM50　TM50とCR50の一致
【 call　関数】	無し
【変　　数】	無し
【ファイル名】	tm5_extev_cnt_cas\timer5_6.c
【注 意 事 項】	無し


main
EI
割り込み許可
tm5_extev
_cnt_cas_ini
TM5 外部イベントカウンタ
(カスケード接続)
tm5_extev
_cnt_cas_st
TM5 外部イベントカウンタ(起動)
(カスケード接続)
tm5_extev_cnt
_cas_ini
PFC5.1 ← 0
PMC5.1 ← 1
TI50入力設定
tm5_extev
_cnt_cas
TM5 外部イベントカウンタ
(カスケード接続)
ret
tm5_extev
_cnt_cas
TM5MK0 ← 0
CR50 ← tm5_extev_cas_CR50
CR51 ← tm5_extev_cas_CR51
TCL50 ← tm5_extev_cas_TCL50
TMC50 ← 0x00
TMC51 ← 0x10
・INTTM50割り込み許可
・コンペアレジスタ下位8ビット
・コンペアレジスタ上位8ビット
・カウントクロック
・カウンタクリア、動作停止
TM50とCR50の一致でクリア＆スタート
TO50ストップ
・カウンタクリア、動作停止
TM51とCR51の一致でクリア＆スタート
カスケード接続
TO51ストップ
ret
tm5_extev
_cnt_cas_st
TCE51 ← 1
TCE50 ← 1
TM51カウント開始
(TCE51 : TMC51レジスタのビット7)
TM50カウント開始
(TCE50 : TMC50レジスタのビット7)
ret
INTTM50
(TM50とCR50の一致)
tm5_extev
_cnt_cas_int
reti

16bitタイマ5　方形波出力　(TM5n)　KF1 / KG1 / KJ1　: n = 0-1

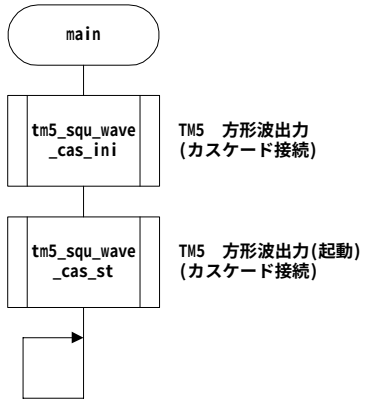
【機　　能】	TMC51レジスタのビット4に“1”をセットする事で16ビット・　タイマ・コンペアレジスタ5(CR50)にあらかじめ設定したカウント値をインターバルとしてTO50端子の出力が反転します。これによって任意の周波数の方形波出力が可能です。	
【関　数　名】	tm5_squ_wave_cas	
【引　　数】	unsigned char tm5_squ_cas_CR50	コンペアレジスタの設定(下位8ビット)
	unsigned char tm5_squ_cas_CR51	コンペアレジスタの設定(上位8ビット)
	unsigned char tm5_squ_cas_TCL50	カウントクロックの設定
【処 理 内 容】	・タイマ5を16ビット方形波出力として動作させる設定を行います。 ・TO50端子から、周期2.5msの方形波を出力します。	
【起 動 方 法】	・タイマ50が動作停止時コールします。 ・コールする前にTO50端子の設定を行ってください。 ・メインクロックを16MHzに設定してください。 ・tm5_squ_wave_cas_st関数のコールによりタイマを動作して下さい。	
【使 用 S F R】	TCL50	カウントクロックの選択
	CR50	インターバル時間の設定(下位8ビット)
	CR51	インターバル時間の設定(上位8ビット)
	TMC50	・8ビット・タイマ・カウンタ50(TM50)のカウント動作制御 ・TM50レジスタの動作モードの選択 ・タイマ出力F/F(フリップフロップ)の状態設定 ・タイマ出力F/Fの制御またはPWM(フリー・ランニング)モード時のアクティブレベルの設定 ・タイマ出力の制御
	TMC51	・8ビット・タイマ・カウンタ50(TM50)のカウント動作制御 ・TM50レジスタの動作モードの選択 ・単体モード/カスケード接続モードの選択 ・タイマ出力F/F(フリップフロップ)の状態設定 ・タイマ出力F/Fの制御またはPWM(フリー・ランニング)モード時のアクティブレベルの設定 ・タイマ出力の制御
【 call　関数】	main	メイン関数
【変　　数】	無し	
【ファイル名】	tm5_7_squ_wave_cas\timer5_7.c , tm5_7_squ_wave_cas\MAIN.C	
【注 意 事 項】	・ TO50端子の設定は、timer5_square_wave_ini関数により行っています。	

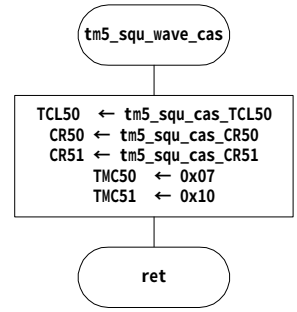
【関　数　名】	tm5_squ_wave_cas_st
【引　　数】	無し
【処 理 内 容】	tm5_squ_wave_casの起動関数です。
【起 動 方 法】	tm5_squ_wave_cas関数の後にコールしてください。
【使 用 S F R】	TCE50　　8ビットタイマ50の動作制御 TCE50　　8ビットタイマ50の動作制御
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	tm5_7_squ_wave_cas\timer5_7.c
【注 意 事 項】	無し

設定時間は以下の式により決定します。（カウントクロック = 4MHz ）

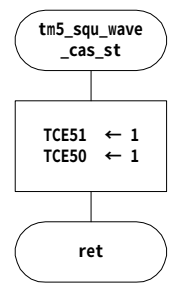
設定時間　=　(コンペア・レジスタ + 1) ×カウントクロック周期

2.5ms　=　(　9999　+　1　)　× 250ns

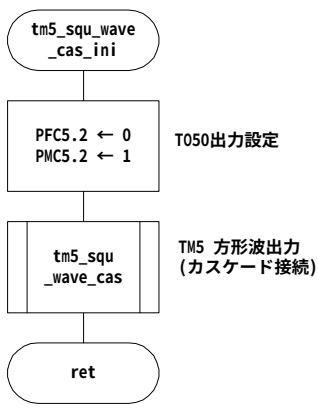




・カウントクロック
・コンペアレジスタ下位8ビットをセット
・コンペアレジスタ上位8ビットをセット
・カウンタクリア、動作停止
TM50とCR50の一致でクリア＆スタート
単独モード
TO50ストップ
・カウンタクリア、動作停止
TM51とCR51の一致でクリア＆スタート
カスケード接続
TO51ストップ



TM51カウント開始
(　TCE51 ： TMC51レジスタのビット7　)
TM50カウント開始
(　TCE50 ： TMC50レジスタのビット7　)



8bitタイマH　インターバルタイマ　(TMHn)　　KF1 / KG1 / KJ1　: n = 0-1

【機　　能】	8ビット・タイマ・カウンタH0のカウント値と8ビット・タイマ・コンペアレジスタ00が一致した場合、割り込み要求信号が発生し、8ビット・タイマ・カウンタ・レジスタを“0”クリアします。
【関　数　名】	timerh_interval
【引　　数】	unsigned char set_TMHMD0　　カウントクロックの設定 unsigned char set_CMP00　　コンペアレジスタの設定
【処 理 内 容】	・1ms毎に割り込み関数をコールします。
【起 動 方 法】	・タイマH0が動作停止時コールします。 ・コールする前にTOH0端子の設定を行ってください。 ・メインクロックを16MHzに設定してください。 ・timerh_interval_st関数のコールによりタイマを動作して下さい。
【使 用 S F R】	TMHIC0　　INTTMH0割り込みのマスクとレベルの設定 TMHMD0　　8ビットタイマH0のモードの制御 CMP00　　8ビットタイマH0コンペアレジスタ CMP01　　8ビットタイマH0コンペアレジスタ
【 call　関数】	main　　メイン関数
【変　　数】	無し
【割 り 込 み】	timerh_interval_int
【割り込み 要 因】	INTTMH0
【フ ァ イ ル 名】	timerh_interval\timerh_1.c , timerh_interval\MAIN.C
【注 意 事 項】	・ TOH0端子の設定は、timerh_interval_ini関数により行っています。

設定時間は以下の式により決定します。　（　fxx = 16MHz（4us）の場合 ）

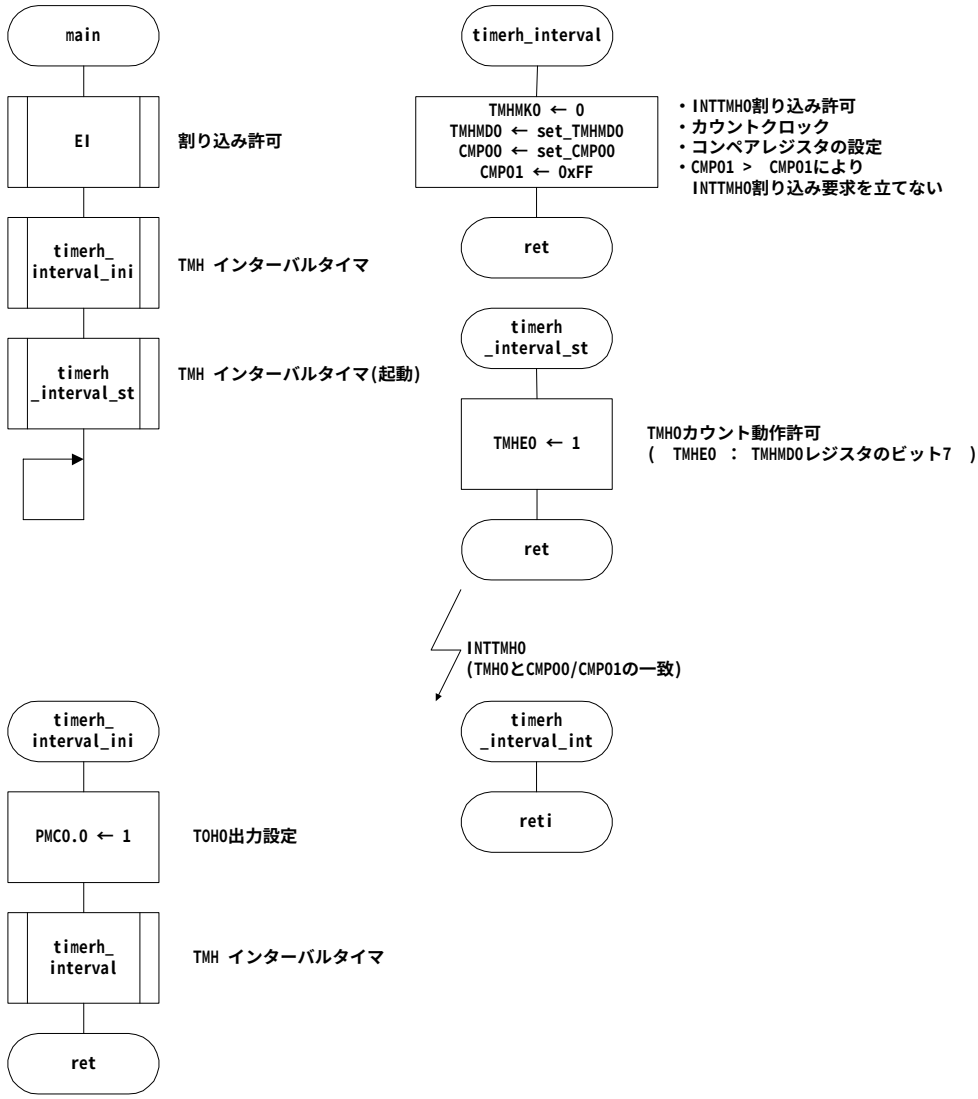
設定時間　=　(コンペア・レジスタ + 1) ×カウントクロック周期

1ms　　=（　249　+　1　）　×　4us

【関　数　名】	timerh_interval_st
【引　　数】	無し
【処 理 内 容】	timerh_intervalの起動関数です。
【起 動 方 法】	timerh_interval関数の後にコールしてください。
【使 用 S F R】	TMHE0　　8ビットタイマH0の動作制御
【 call　関数】	無し
【変　　数】	無し
【注 意 事 項】	無し

割り込み関数

【関　数　名】	timerh_interval_int
【概　　要】	ユーザー定義
【要　　因】	INTTMH0　　TMH0とCMP00/CMP01の一致
【 call 関 数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	timerh_interval\timerh_1.c
【注 意 事 項】	無し


main
EI
割り込み許可
timerh_
interval_ini
TMH インターバルタイマ
timerh
_interval_st
TMH インターバルタイマ(起動)
timerh_interval
TMHMK0 ← 0
TMHMD0 ← set_TMHMD0
CMP00 ← set_CMP00
CMP01 ← 0xFF
・INTTMH0割り込み許可
・カウントクロック
・コンペアレジスタの設定
・CMP01 > CMP01により
INTTMH0割り込み要求を立てない
ret
timerh
_interval_st
TMHE0 ← 1
TMH0カウント動作許可
(TMHE0 ： TMHMD0レジスタのビット7)
ret
INTTMH0
(TMH0とCMP00/CMP01の一致)
timerh_
interval_ini
PMC0.0 ← 1
TOH0出力設定
timerh_
interval
TMH インターバルタイマ
ret
timerh
_interval_int
reti

8bitタイマH　PWM出力　(TMHn)　　KF1 / KG1 / KJ1　: n = 0-1

【機　　能】	TMHMD0レジスタのビット3＝1、ビット2＝0をセットする事で8ビット・タイマ・コンペア・レジスタH0(CMP00)に設定した値で決まるデューティー比のパルスをTOH0端子から出力します。
【関　数　名】	timerh_pwm
【引　　数】	unsigned char tmh_pwm_TMHMD0 カウントクロックの設定 unsigned char tmh_pwm_CMP00　パルス周期の設定 unsigned char tmh_pwm_CMP01　デューティー比の設定
【処 理 内 容】	・タイマH0をPWM出力として動作させる設定を行います。 ・TOH0からデューティー比3：2の波形を出力します。
【起 動 方 法】	・タイマH0が動作停止時コールします。 ・コールする前にTOH0端子の設定を行ってください。 ・timerh_pwm_st関数のコールによりタイマを動作して下さい。
【使 用 S F R】	TMHMD0　8ビットタイマH0のモードの制御 CMP00　インターバル時間の設定 CMP01　デューティー比の設定
【 call　関数】	main　メイン関数
【変　　数】	無し
【ファイル名】	timerh_pwm\timerh_2.c , timerh_pwm\MAIN.C
【注 意 事 項】	・TOH0端子の設定は、timerh_interval_ini関数により行っています。

【関　数　名】	timerh_pwm_st
【引　　数】	無し
【処 理 内 容】	timerh_pwmの起動関数です。
【起 動 方 法】	timerh_pwm関数の後にコールしてください。
【使 用 S F R】	TMHE0　　8ビットタイマH0の動作制御
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	timerh_pwm\timerh_2.c
【注 意 事 項】	無し

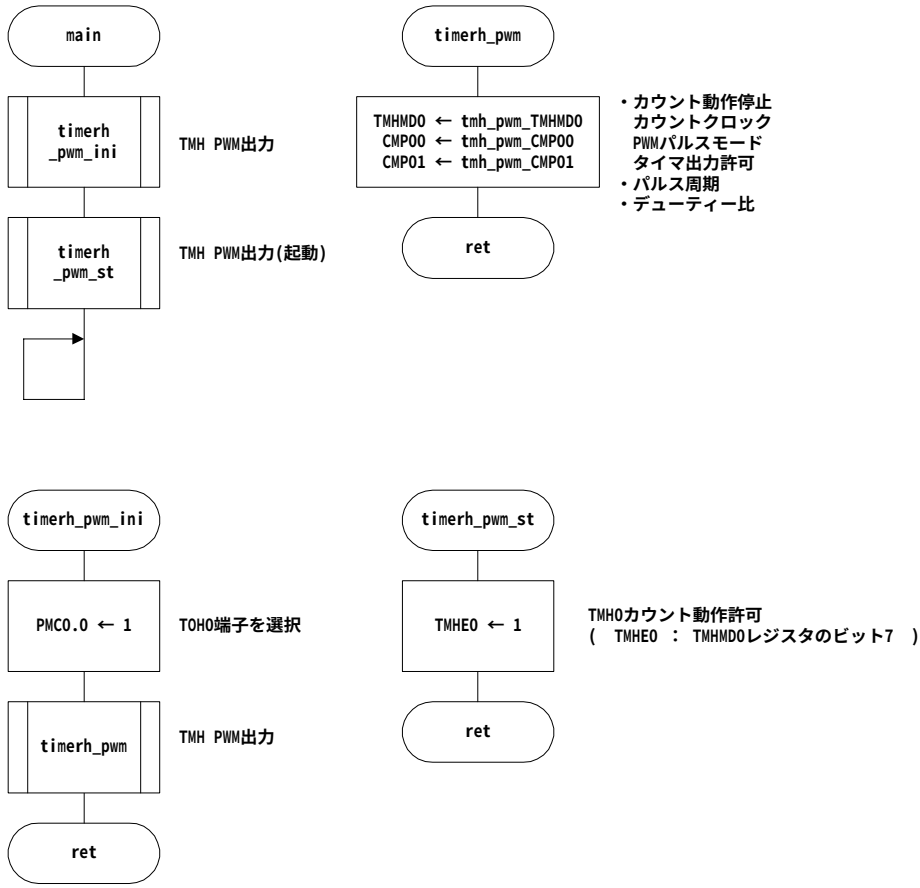
設定時間は以下の式により決定します。（カウントクロック　= 8MHzの場合 ）

分周　　1 / 2 （ 125ns ）

設定時間　=　（ コンペア・レジスタ + 1 ）×カウントクロック周期

12.5μs　= 100 × 125ns

7.5μs　= 60 × 125ns


main
timerh
_pwm_ini
TMH PWM出力
timerh
_pwm_st
TMH PWM出力(起動)
timerh_pwm
TMHMD0 ← tmh_pwm_TMHMD0
CMP00 ← tmh_pwm_CMP00
CMP01 ← tmh_pwm_CMP01
・カウント動作停止
カウントクロック
PWMパルスモード
タイマ出力許可
・パルス周期
・デューティー比
ret
timerh_pwm_ini
PMC0.0 ← 1
TOH0端子を選択
timerh_pwm
TMH PWM出力
ret
timerh_pwm_st
TMHE0 ← 1
TMH0カウント動作許可
(TMHE0 : TMHMD0レジスタのビット7)
ret

リアルタイム出力機能 (RTOn)

KF1 / KG1 : n = 0

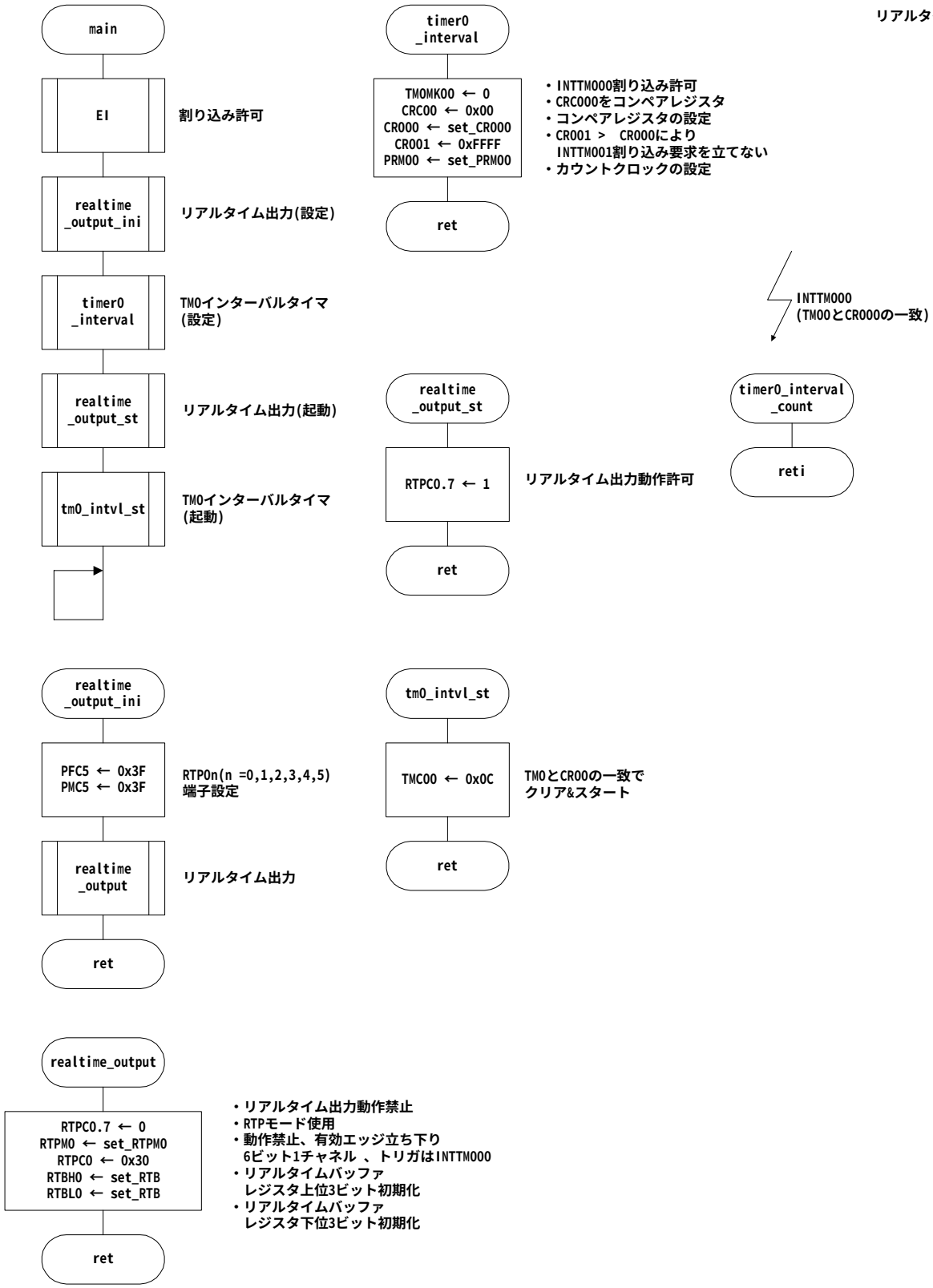
KJ1 : n = 0-1

【機　　能】	リアルタイム出力バッファ・レジスタ0(RTBL0, RTBH0)にあらかじめ設定したデータを、外部割込みまたは外部発生トリガと同時にハードウェアで出力ラッチに転送して外部に出力します。
【関 数 名】	realtime_output
【引　　数】	unsigned char set_RTPM0　リアルタイム出力ポート・モードとポートモードの選択を1ビット単位で設定 unsigned char set_RTB　リアルタイム出力するデータ
【処 理 内 容】	INTTM000信号の有効エッジをトリガとして、RTP0n(n = 0,1,2,3,4,5)端子から、リアルタイム出力バッファ・レジスタn (RTBLn、 RTBHn)に設定したデータを出力します。
【起 動 方 法】	・コールする前にRTP0n(n =0,1,2,3,4,5)端子の設定を行ってください。 ・ realtime_output関数コール後にtimer0_interval関数をｺｰﾙしてください。 ・tm0_intvl_st関数のコールによりタイマを動作して下さい。 ・realtime_output_st関数のコールによりリアルタイム出力を動作して下さい。
【使 用 S F R】	RTPM0　リアルタイム出力ポート・モードとポートモードの選択を1ビット単位で設定 RTPC0　リアルタイム出力ポートの動作モードと出力トリガの設定 RTBH0　リアルタイム出力バッファ上位4ビット RTBL0　リアルタイム出力バッファ下位4ビット
【 call 関数】	main　メイン関数
【変　　数】	無し
【フ ァ イ ル 名】	realtime_output\RT0_1.c , realtime_output\timer0_1.c realtime_output\MAIN.C
【注 意 事 項】	・端子の設定は、realtime_output_ini関数により行います。 ・1度リアルタイム出力動作を禁止 (RTP0Enビット = 0) した場合は、リアルタイム出力バッファ・レジスタn(RTBLn、 RTBHn)を初期化してください。

【関　数　名】	realtime_output_st
【引　　数】	無し
【処 理 内 容】	realtime_outputの起動関数です。
【起 動 方 法】	realtime_output関数の後にコールしてください。
【使 用 S F R】	RTPOE0(ビット操作不可) (RTPC0.7)　　リアルタイム出力制御
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	realtime_output\RTO_1.c
【注 意 事 項】	無し

割り込み関数

【関　数　名】	timer0_interval_count
【概　　要】	ユーザー定義
【要　　因】	INTTM000　　TM00とCR000の一致
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	realtime_output\timer0_1.c
【注 意 事 項】	無し


main
EI
割り込み許可
realtime
_output_ini
リアルタイム出力(設定)
timer0
_interval
TM0インターバルタイマ
(設定)
realtime
_output_st
リアルタイム出力(起動)
tm0_intvl_st
TM0インターバルタイマ
(起動)
timer0
_interval
TM0MK00 ← 0
CRC00 ← 0x00
CR000 ← set_CR000
CR001 ← 0xFFFF
PRM00 ← set_PRM00
ret
・INTTM000割り込み許可
・CRC000をコンペアレジスタ
・コンペアレジスタの設定
・CR001 > CR000により
INTTM001割り込み要求を立てない
・カウントクロックの設定
INTTM000
(TM00とCR000の一致)
realtime
_output_st
RTPC0.7 ← 1
リアルタイム出力動作許可
ret
timer0_interval
_count
reti
realtime
_output_ini
PFC5 ← 0x3F
PMC5 ← 0x3F
RTP0n(n =0,1,2,3,4,5)
端子設定
realtime
_output
リアルタイム出力
ret
tm0_intvl_st
TMC00 ← 0x0C
TM0とCR00の一致で
クリア&スタート
ret
realtime_output
RTPC0.7 ← 0
RTPM0 ← set_RTPM0
RTPC0 ← 0x30
RTBH0 ← set_RTB
RTBL0 ← set_RTB
ret
・リアルタイム出力動作禁止
・RTPモード使用
・動作禁止、有効エッジ立ち下り
6ビット1チャネル 、トリガはINTTM000
・リアルタイムバッファ
レジスタ上位3ビット初期化
・リアルタイムバッファ
レジスタ下位3ビット初期化

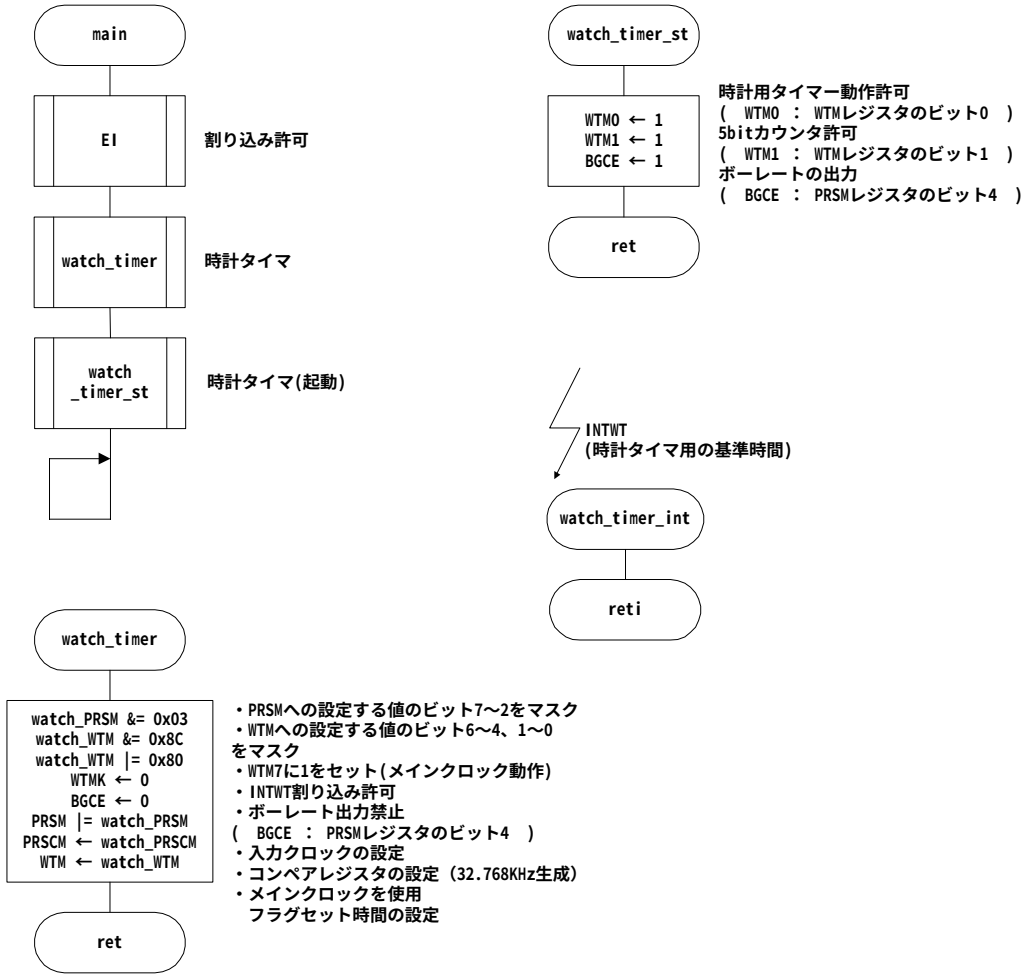
時計用タイマ機能　時計タイマ　(　時計タイマn)　KF1 / KG1 / KJ1　: n = 0

【機　　能】	メインクロックの使用により、0.5秒または0.25秒の時間間隔で割り込み要求を発生します。	
【関　数　名】	watch_timer	
【引　　数】	unsigned char watch_PRSM	メインクロックの分周
	unsigned char watch_PRSCM	32.768KHz = 30.52us 生成のコンペア値の設定
	unsigned char watch_WTM	時計フラグセットのインターバル時間
【処 理 内 容】	・時計用タイマの設定を行います。 ・watch_WTMで設定した時間毎に割り込み関数をコールします。	
【起 動 方 法】	・初期設定時コールしてください。 ・watch_timer_st関数のコールによりタイマを動作して下さい。	
【使 用 S F R】	WTIC	INTWT割り込みのマスクとレベルの設定
	PRSM	時計用タイマのボー・レート信号の生成を制御
	PRSCM	8ビットのコンペアレジスタ（時計用タイマのボー・レート信号の生成に使用）
	WTM	時計タイマのカウント・クロックおよび動作の許可 / 禁止、プリスケーラのインターバル時間、5ビットカウンタの動作制御及び時計フラグのセット時間を設定
【 call　関数】	main	メイン関数
【変　　数】	無し	
【割 り 込 み】	watch_timer_int	
【割り込み 要 因】	INTWT	
【フ ァ イ ル 名】	watch_timer\tokei_timer_1.c , watch_timer\MAIN.C	
【注 意 事 項】	無し	

【関　数　名】	watch_timer_st
【引　　数】	無し
【処 理 内 容】	watch_timerの起動関数です。
【起 動 方 法】	watch_timer関数の後にコールしてください。
【使 用 S F R】	WTM0　　時計用タイマー動作制御 WTM1　　5bitカウンタ動作制御 BGCE　　ボーレートの動作制御
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	watch_timer\tokei_timer_1.c
【注 意 事 項】	無し

割り込み関数

【関　数　名】	watch_timer_int
【概　　要】	ユーザー定義
【要　　因】	INTWT　　時計タイマ用の基準時間
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	watch_timer\tokei_timer_1.c
【注 意 事 項】	無し


main
EI
割り込み許可
watch_timer
時計タイマ
watch
_timer_st
時計タイマ(起動)
watch_timer_st
WTM0 ← 1
WTM1 ← 1
BGCE ← 1
時計用タイマー動作許可
(WTM0 ： WTMレジスタのビット0)
5bitカウンタ許可
(WTM1 ： WTMレジスタのビット1)
ボーレートの出力
(BGCE ： PRSMレジスタのビット4)
ret
INTWT
(時計タイマ用の基準時間)
watch_timer_int
reti
watch_timer
watch_PRSM &= 0x03
watch_WTM &= 0x8C
watch_WTM |= 0x80
WTMK ← 0
BGCE ← 0
PRSM |= watch_PRSM
PRSCM ← watch_PRSCM
WTM ← watch_WTM
・PRSMへの設定する値のビット7～2をマスク
・WTMへの設定する値のビット6～4、1～0
をマスク
・WTM7に1をセット(メインクロック動作)
・INTWT割り込み許可
・ボーレート出力禁止
(BGCE ： PRSMレジスタのビット4)
・入力クロックの設定
・コンペアレジスタの設定（32.768KHz生成）
・メインクロックを使用
フラグセット時間の設定
ret

時計用タイマ機能　インターバル・タイマ　（　時計タイマn ）

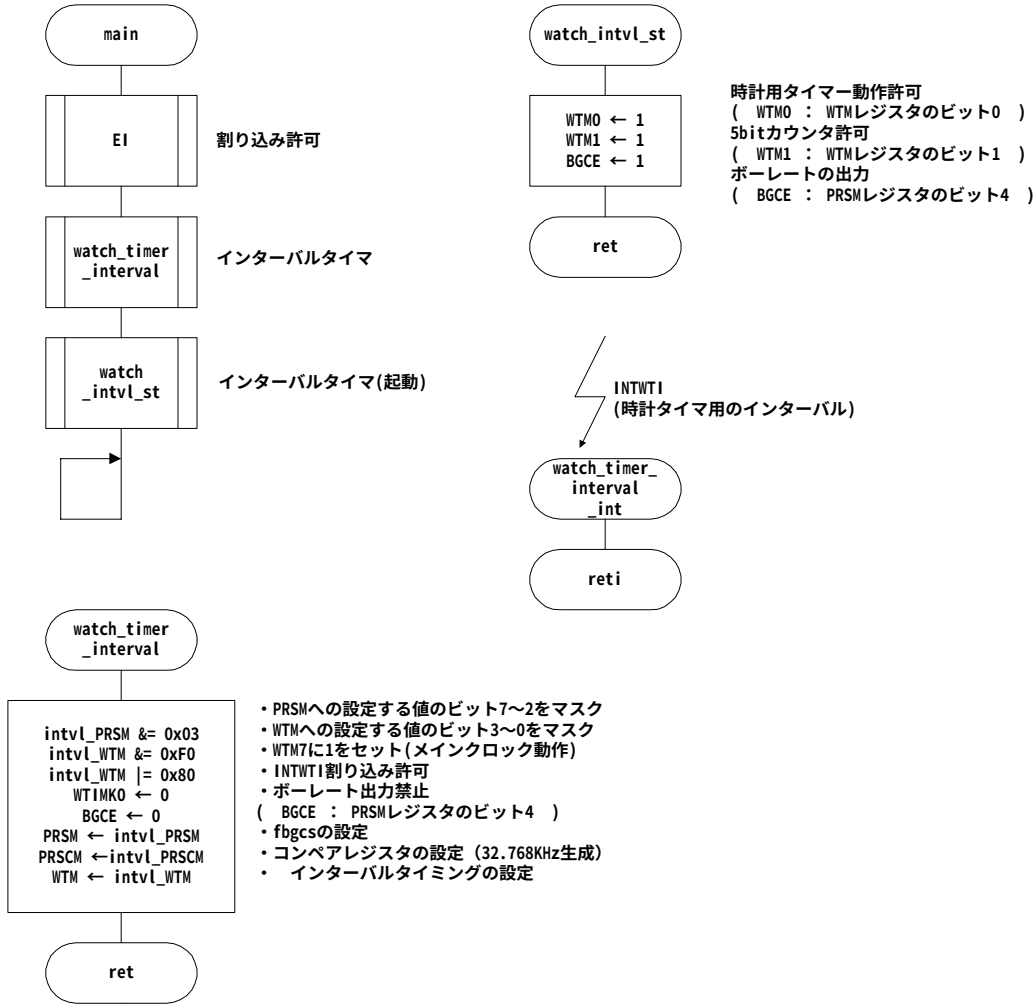
KF1 / KG1 / KJ1　: n = 0

【機　　能】	WTMレジスタに設定した値をインターバルとし、繰り返し割り込みを発生するインターバルタイマとして動作します。	
【関　数　名】	watch_timer_interval	
【引　　数】	unsigned char intvl_PRSM	メインクロックの分周
	unsigned char intvl_PRSCM	32.768KHz　=　30.52us 生成のコンペア値の設定
	unsigned char intvl_WTM	割り込みインターバル時間
【処　理　内　容】	・intvl_WTMで指定したインターバルで割り込み処理を発生します。	
【起　動　方　法】	・初期設定時にコールしてください。 ・watch_intvl_st関数のコールによりタイマを動作して下さい。	
【使　用 S　F　R】	WTIIC	INTWTI割り込みのマスクとレベルの設定
	PRSM	時計用タイマのボー・レート信号の生成を制御
	PRSCM	8ビットのコンペアレジスタ（　時計用タイマのボー・レート信号の生成に使用　）
	WTM	時計タイマのカウント・クロックおよび動作の許可　/　禁止、プリスケーラのインターバル時間、5ビットカウンタの動作制御及び時計フラグのセット時間を設定
【 call　　関数】	main	メイン関数
【変　　数】	無し	
【割　り　込　み】	watch_timer_int	
【割り込み 要 因】	INTWTI	
【フ ァ イ ル 名】	watch_timer_interval\tokei_timer_2.c , watch_timer_interval\MAIN.C	
【注　意　事　項】	無し	

【関　数　名】	watch_intvl_st
【引　　数】	無し
【処 理 内 容】	watch_timer_intervalの起動関数です。
【起 動 方 法】	watch_timer_interval関数の後にコールしてください。
【使 用 S F R】	WTM0　　時計用タイマー動作制御 WTM1　　5bitカウンタ動作制御 BGCE　　ボーレートの動作制御
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	watch_timer_interval\tokei_timer_2.c
【注 意 事 項】	無し

割り込み関数

【関　数　名】	watch_timer_interval_int
【概　　要】	ユーザー定義
【要　　因】	INTWTI　　時計タイマ用のインターバル
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	watch_timer_interval\tokei_timer_2.c
【注 意 事 項】	無し


main
EI
割り込み許可
watch_timer
_interval
インターバルタイマ
watch
_intvl_st
インターバルタイマ(起動)
watch_intvl_st
WTMO ← 1
WTM1 ← 1
BGCE ← 1
時計用タイマー動作許可
(WTMO : WTMレジスタのビット0)
5bitカウンタ許可
(WTM1 : WTMレジスタのビット1)
ボーレートの出力
(BGCE : PRSMレジスタのビット4)
ret
INTWTI
(時計タイマ用のインターバル)
watch_timer_
interval
_int
reti
watch_timer
_interval
intvl_PRSM &= 0x03
intvl_WTM &= 0xF0
intvl_WTM |= 0x80
WTIMK0 ← 0
BGCE ← 0
PRSM ← intvl_PRSM
PRSCM ←intvl_PRSCM
WTM ← intvl_WTM
・PRSMへの設定する値のビット7〜2をマスク
・WTMへの設定する値のビット3〜0をマスク
・WTM7に1をセット(メインクロック動作)
・INTWTI割り込み許可
・ボーレート出力禁止
(BGCE : PRSMレジスタのビット4)
・fbgcsの設定
・コンペアレジスタの設定（32.768KHz生成）
・ インターバルタイミングの設定
ret

時計用タイマ機能　時計タイマ　サブクロック動作

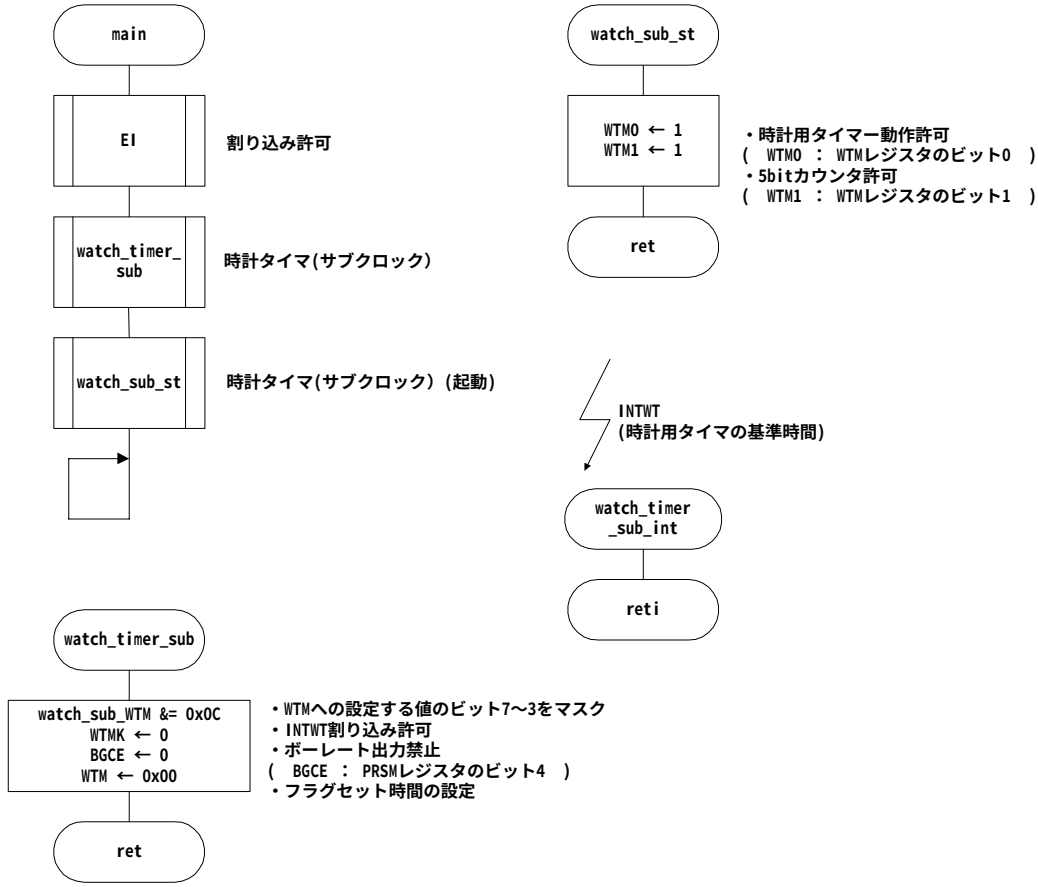
(　時計タイマn)　　KF1 / KG1 / KJ1　: n = 0

【機　　能】	サブクロックの使用により、0.5秒または0.25秒の時間間隔で割り込み要求を発生します	
【関　数　名】	watch_timer_sub	
【引　　数】	unsigned char watch_sub_WTM	時計フラグセットのインターバル時間
【処　理　内　容】	・時計用タイマの設定を行います。(サブクロック動作) ・ watch_sub_WTMの設定値毎に割り込み関数をコールします。	
【起　動　方　法】	・初期設定時にコールしてください。 ・クロックをサブクロックに設定してください。 ・watch_sub_st関数のコールによりタイマを動作して下さい。	
【使　用 S　F　R】	WTIC PRSM WTM	INTWT割り込みのマスクとレベルの設定 時計用タイマのボー・レート信号の生成を制御 時計タイマのカウント・クロックおよび動作の許可　/禁止、プリスケーラのインターバル時間、5ビットカウンタの動作制御及び時計フラグのセット時間を設定
【 call　　関数】	main	メイン関数
【変　　数】	無し	
【割　り　込　み】	watch_timer_sub_int	
【割り込み 要 因】	INTWT	
【フ ァ イ ル 名】	watch_timer_sub\tokei_timer_3.c , watch_timer_sub\MAIN.C	
【注　意　事　項】	無し	

【関　数　名】	watch_sub_st
【引　　数】	無し
【処 理 内 容】	watch_timer_subの起動関数です。
【起 動 方 法】	watch_timer_sub関数の後にコールしてください。
【使 用 S F R】	WTM0　　時計用タイマー動作制御 WTM1　　5bitカウンタ動作制御
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	watch_timer_sub\tokei_timer_3.c
【注 意 事 項】	無し

割り込み関数

【関　数　名】	watch_timer_sub_int
【概　　要】	watch_sub_WTMの設定したインターバル時間ごとにコールします。
【要　　因】	INTWT　　時計タイマ用の基準時間
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	watch_timer_sub\tokei_timer_3.c
【注 意 事 項】	無し


main
EI
割り込み許可
watch_timer_
sub
時計タイマ(サブクロック)
watch_sub_st
時計タイマ(サブクロック)（起動)
watch_sub_st
WTM0 ← 1
WTM1 ← 1
・時計用タイマー動作許可
(WTM0 ： WTMレジスタのビット0)
・5bitカウンタ許可
(WTM1 ： WTMレジスタのビット1)
ret
INTWT
(時計用タイマの基準時間)
watch_timer
_sub_int
reti
watch_timer_sub
watch_sub_WTM &= 0x0C
WTMK ← 0
BGCE ← 0
WTM ← 0x00
・WTMへの設定する値のビット7～3をマスク
・INTWT割り込み許可
・ボーレート出力禁止
(BGCE ： PRSMレジスタのビット4)
・フラグセット時間の設定
ret

時計用タイマ機能　インターバル・タイマ　サブクロック動作

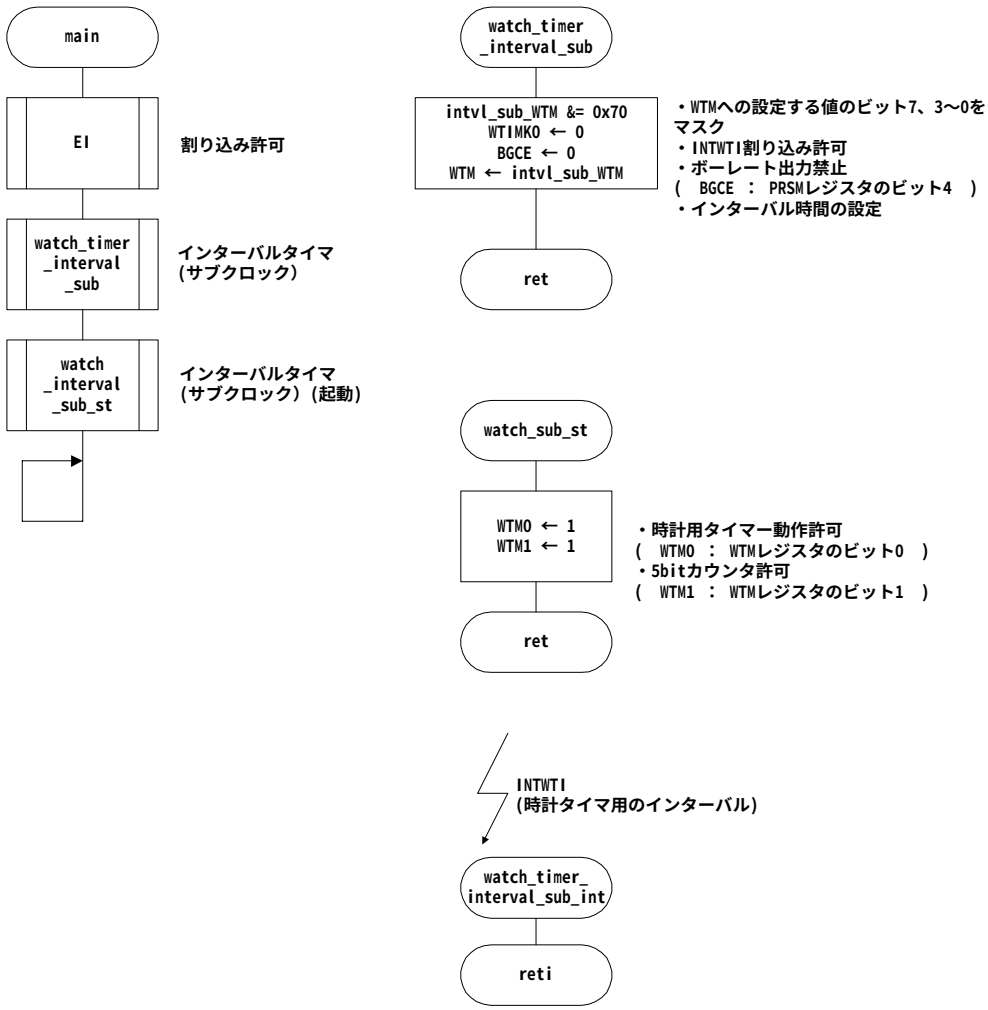
(　時計タイマn)　　KF1 / KG1 / KJ1　: n = 0

【機　　能】	WTMレジスタに設定した値をインターバルとし、繰り返し割り込みを発生するインターバルタイマとして動作します。	
【関　数　名】	watch_timer_interval_sub	
【引　　数】	unsigned char intvl_sub_WTM	割り込みインターバル時間
【処　理　内　容】	・intvl_sub_WTMで設定したインターバルで割　り込み処理を発生します。	
【起　動　方　法】	・初期設定時にコールしてください。 ・クロックをサブクロックに設定してください。 ・watch_interval_sub_st関数のコールによりタイマを動作して下さい。	
【使　用 S　F　R】	WTIIC PRSM WTM	INTWT割り込みのマスクとレベルの設定 時計用タイマのボー・レート信号の生成を制御 時計タイマのカウント・クロックおよび動作の許可　/禁止、プリスケーラのインターバル時間、5ビットカウンタの動作制御及び時計フラグのセット時間を設定
【 call　　関数】	main	メイン関数
【変　　数】	無し	
【割　り　込　み】	watch_timer_interval_sub_int	
【割り込み 要 因】	INTWT	
【フ ァ イ ル 名】	watch_timer_interval_sub\tokei_timer4.c watch_timer_interval_sub\MAIN.C	
【注　意　事　項】	無し	

【関　数　名】	watch_interval_sub_st
【引　　数】	無し
【処 理 内 容】	watch_timer_interval_subの起動関数です。
【起 動 方 法】	watch_timer_interval_sub関数の後にコールしてください。
【使 用 S F R】	WTM0　　時計用タイマー動作制御 WTM1　　5bitカウンタ動作制御
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	watch_timer_interval_sub\tokei_timer4.c
【注 意 事 項】	無し

割り込み関数

【関　数　名】	watch_timer_interval_sub_int
【概　　要】	ユーザー定義
【要　　因】	INTWTI　　時計タイマ用のインターバル
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	watch_timer_interval_sub\tokei_timer4.c
【注 意 事 項】	無し


main
EI
割り込み許可
watch_timer
_interval
_sub
インターバルタイマ
(サブクロック)
watch
_interval
_sub_st
インターバルタイマ
(サブクロック)(起動)
watch_timer
_interval_sub
intvl_sub_WTM &= 0x70
WTIMK0 ← 0
BGCE ← 0
WTM ← intvl_sub_WTM
・WTMへの設定する値のビット7、3～0を
マスク
・INTWTI割り込み許可
・ボーレート出力禁止
(BGCE : PRSMレジスタのビット4)
・インターバル時間の設定
ret
watch_sub_st
WTM0 ← 1
WTM1 ← 1
・時計用タイマー動作許可
(WTM0 : WTMレジスタのビット0)
・5bitカウンタ許可
(WTM1 : WTMレジスタのビット1)
ret
INTWTI
(時計タイマ用のインターバル)
watch_timer_
interval_sub_int
reti

ウオッチドッグ・タイマ１

(WDT1)　KF1 / KG1 / KJ1

【機　　能】	WDTM1レジスタのビット4に1をセットすることにより、プログラムの暴走を検出するウオッチドッグ・タイマ1として動作します。 WDTMレジスタのビット7(RUN1ビット)に1をセットすることでカウント動作を開始します。 ・検出時間範囲内にRUN1ビットに1をセットすることでカウントをリセットします。 ・検出時間範囲内にRUN1ビットに1をセットしなかった場合 WDTM1レジスタのビット3、4の組み合わせにより、マスカブル割り込み、ノンマスカブル割り込み、リセットのいずれかを発生します。	
【関　数　名】	watchdog_timer	
【引　　数】	unsigned char wdt1_OSTS unsigned char wdt1_WDCS unsigned char wdt1_WDTM1	発振安定時間の選択 ウオッチドッグ・タイマ1、インターバル・タイマオーバーフロー時間 ウオッチドッグ・タイマ1の動作モード、カウント許可　/　禁止の設定
【処 理 内 容】 【起 動 方 法】	・WDCS、WDTM1の設定によりWDT1を動作します。 ・初期設定でコールしてください。 ・wdt1_st関数のコールによりタイマを動作して下さい。	
【使 用 S F R】	OSTS WDCS WDT1IC WDTM1	発振安定時間の選択 ウオッチドッグ・タイマ1、インターバル・タイマオーバーフロー時間 INTWDT1割り込みのマスクとレベルの設定 ウオッチドッグ・タイマ1の動作モード、カウント許可/禁止の設定
【 call　関数】	main	メイン関数
【変　　数】	set_wdtm1	引数wdt1_WDTM1を格納
【割 り 込 み】	watchdog_timer_int interval_timer_int	
【割り込み 要 因】	INTWDT1 INTWDTM1	
【フ ァ イ ル 名】	watchdog_timer\wdt_1.c , watchdog_timer\MAIN.C	
【注 意 事 項】	・カウンタをクリアする場合は、検出時間範囲内にwdt1_cnt_clr関数をコールしてください。 ・WDTM1レジスタは特定レジスタの為PRCMDレジスタに書きこんでからライトしてください。 PRCMDレジスタへのライトにはWDTM1レジスタと同じ汎用レジスタを使用してください。	

ウオッチドッグ・タイマ1

(WDT1)　KF1 / KG1 / KJ1

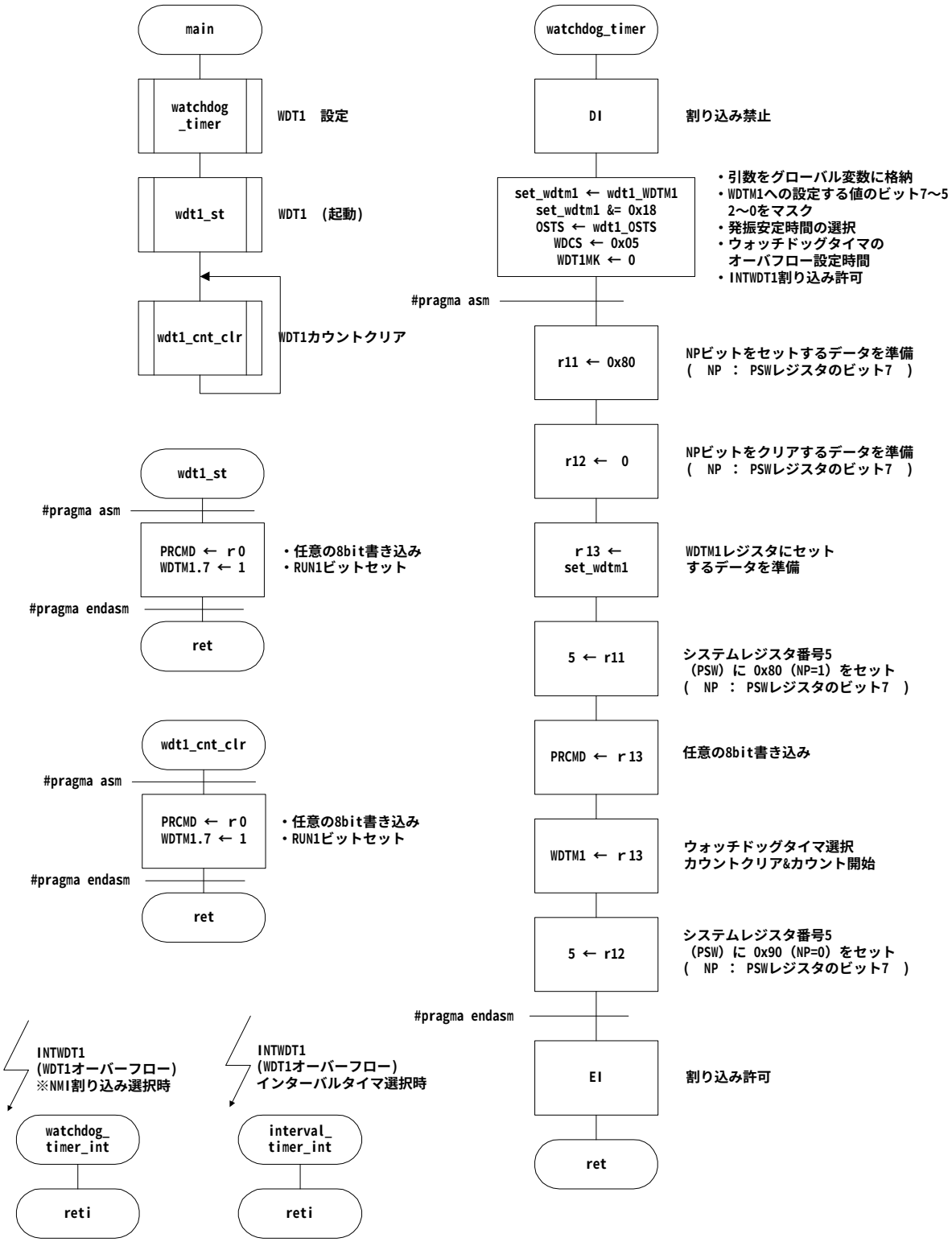
【関　数　名】	wdt1_st
【引　　数】	無し
【処 理 内 容】	watchdog_timerの起動関数です。
【起 動 方 法】	watchdog_timer関数の後にコールしてください。
【使 用 S F R】	RUN1　　WDT1のカウンタの動作制御
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	watchdog_timer\wdt_1.c
【注 意 事 項】	無し

【機　　能】	・検出時間範囲内にRUN1ビットに“1”をセットすることでカウントをリセットします。
【関　数　名】	wdt1_cnt_clr
【引　　数】	無し
【処 理 内 容】	RUN1ビット“1”をセットする事でWDT1のカウンタをクリアします。
【起 動 方 法】	無し
【使 用 S F R】	WDTM1　　ウオッチドッグ・タイマ1の動作モード、カウント許可 / 禁止の設定
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	watchdog_timer\wdt_1.c
【注 意 事 項】	WDTM1レジスタは特定レジスタの為PRCMDレジスタに書きこんでからライトしてください。 PRCMDレジスタへのライトにはWDTM1レジスタと同じ汎用レジスタを使用してください。

割り込み関数

【関　数　名】	watchdog_timer_int
【概　　要】	ユーザー処理
【要　　因】	INTWDT1　　WDT1オーバーフロー
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	watchdog_timer\wdt_1.c
【注 意 事 項】	無し

【関　数　名】	interval_timer_int
【概　　要】	ユーザー処理
【要　　因】	INTWDTM1　　WDT1オーバーフロー
【 call　関数】	無し
【変　　数】	無し
【フ ァ イ ル 名】	watchdog_timer\wdt_1.c
【注 意 事 項】	無し

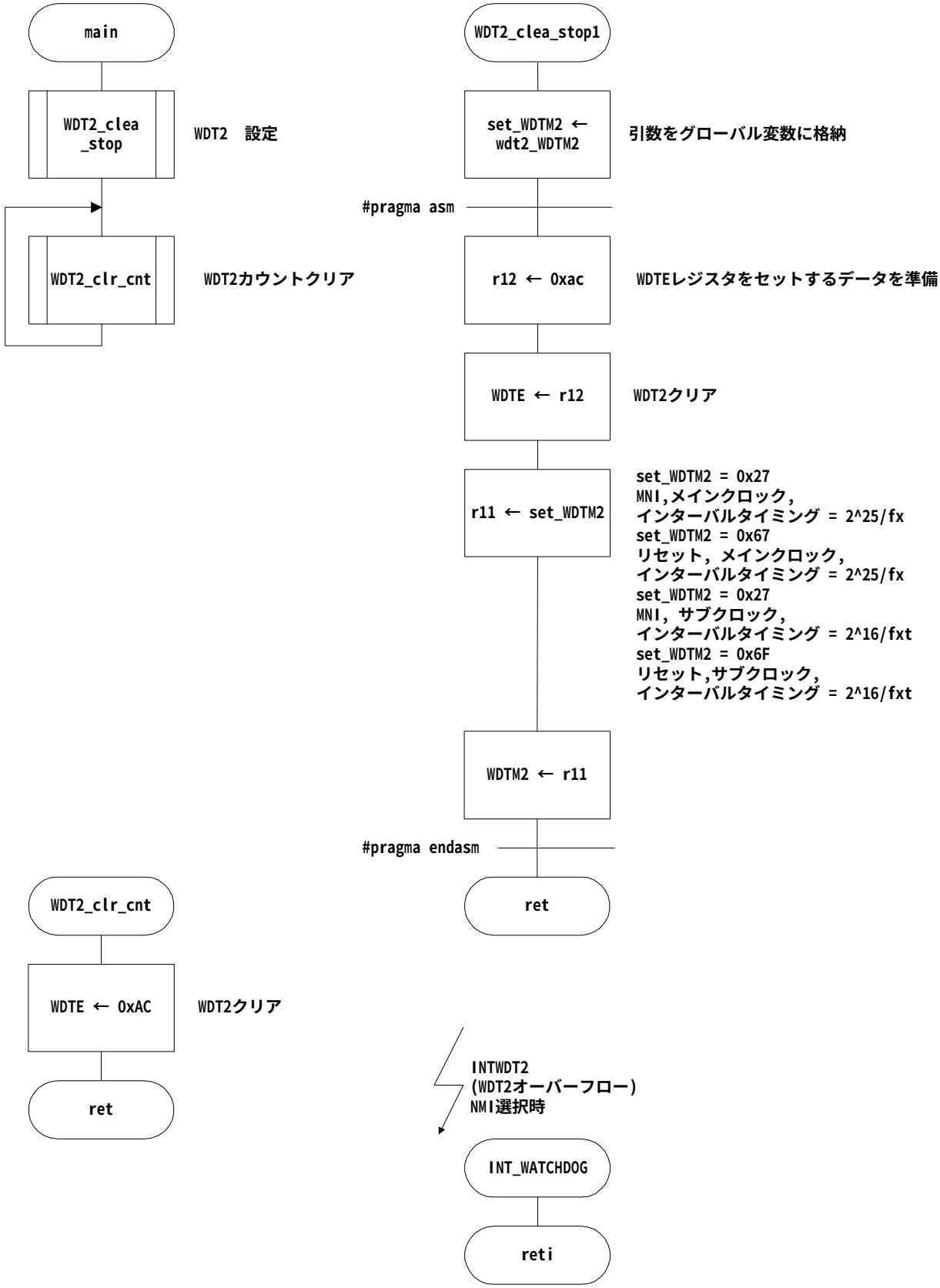

main
watchdog_timer
WDT1　設定
WDT1　(起動)
WDT1カウントクリア
wdt1_st
wdt1_cnt_clr
#pragma asm
PRCMD ← r0
WDTM1.7 ← 1
・任意の8bit書き込み
・RUN1ビットセット
#pragma endasm
ret
INTWDT1
(WDT1オーバーフロー)
※NMI割り込み選択時
watchdog_
timer_int
reti
INTWDT1
(WDT1オーバーフロー)
インターバルタイマ選択時
interval_
timer_int
DI
割り込み禁止
set_wdtm1 ← wdt1_WDTM1
set_wdtm1 &= 0x18
OSTS ← wdt1_OSTS
WDCS ← 0x05
WDT1MK ← 0
・引数をグローバル変数に格納
・WDTM1への設定する値のビット7〜5
2〜0をマスク
・発振安定時間の選択
・ウォッチドッグタイマの
オーバフロー設定時間
・INTWDT1割り込み許可
r11 ← 0x80
NPビットをセットするデータを準備
(NP : PSWレジスタのビット7)
r12 ← 0
NPビットをクリアするデータを準備
(NP : PSWレジスタのビット7)
r13 ←
set_wdtm1
WDTM1レジスタにセット
するデータを準備
5 ← r11
システムレジスタ番号5
(PSW) に 0x80 (NP=1) をセット
(NP : PSWレジスタのビット7)
PRCMD ← r13
任意の8bit書き込み
WDTM1 ← r13
ウォッチドッグタイマ選択
カウントクリア&カウント開始
5 ← r12
システムレジスタ番号5
(PSW) に 0x90 (NP=0) をセット
(NP : PSWレジスタのビット7)
EI
割り込み許可

ウオッチドッグ・タイマ2 (WDT2) KF1 / KG1 / KJ1 : 1

【機　　能】	リセット解除後に自動的にリセット・モードでスタートしており、 次のインターバル時間内で設定を行う必要があります。 WDT2はタイマのには、WDTM2の設定で次のような動作を行います。 ・WDT2タイマ・オーバーフローによるリセット動作 ・WDT2タイマ・オーバーフローによるNMI動作 ソース・クロックにはメインクロックとサブクロックが選択できます。	
【引　　数】	unsigned char wdt2_WDTM2	動作モード・選択クロックの設定
【処 理 内 容】	引数wdt2_WDTM2に設定した値でWDT2を動作します。	
【起 動 方 法】	・初期設定でコールしてください。 ・WDT2タイマ・オーバーフロー時間内にWDT2_clr_cnt関数をコールし WDT2のカウントをクリアしてください。	
【使 用 S F R】	WDTE WDTM2	WDT2のカウントをクリア 動作モード・選択クロックの設定
【 call 関数】	無し	
【変　　数】	set_WDTM2	引数wdt2_WDTM2を格納
【割 り 込 み】	INT_WATCHDOG	
【割り込み 要 因】	INTWDT2	
【フ ァ イ ル 名】	wdt2\wdt_2.c , wdt2\MAIN.C , wdt2\int\startup_wdt_1.s	
【注 意 事 項】	・WDTM2レジスタへの書きこみは1回しか、行えません。 ・リセット解除後に自動的にリセット・モードでスタートしているので 必要の無い場合も、書きこみを行ってください。	

割り込み関数

【関　数　名】	wdt2_int	
【概　　要】	ユーザー処理	
【要　　因】	INTWDT2	WDT2タイマ・オーバーフロー
【 call　関数】	無し	
【変　　数】	無し	
【ファイル名】	wdt2\wdt_2.c	
【注 意 事 項】	無し	


main
WDT2_clea
_stop
WDT2 設定
WDT2_clr_cnt
WDT2カウントクリア
WDT2_clea_stop1
set_WDTM2 ←
wdt2_WDTM2
引数をグローバル変数に格納
#pragma asm
r12 ← 0xac
WDTEレジスタをセットするデータを準備
WDTE ← r12
WDT2クリア
r11 ← set_WDTM2
set_WDTM2 = 0x27
MNI,メインクロック,
インターバルタイミング = 2^25/fx
set_WDTM2 = 0x67
リセット, メインクロック,
インターバルタイミング = 2^25/fx
set_WDTM2 = 0x27
MNI, サブクロック,
インターバルタイミング = 2^16/fxt
set_WDTM2 = 0x6F
リセット,サブクロック,
インターバルタイミング = 2^16/fxt
WDTM2 ← r11
#pragma endasm
ret
WDT2_clr_cnt
WDTE ← 0xAC
WDT2クリア
ret
INTWDT2
(WDT2オーバーフロー)
NMI選択時
INT_WATCHDOG
reti