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。

RENESAS

アプリケーション・ノート

78K0/Kx2

サンプル・プログラム

PPG出力編

この資料は，サンプル・プログラムの「PPG」の動作概要と，マイコンの基本的なPPGの設定を説明したものです。
サンプル・プログラムでは，PPGの設定を行ったあとに，その出力制御を行います。

対象デバイス

78K0/KB2マイクロコントローラ
78K0/KC2マイクロコントローラ
78K0/KD2マイクロコントローラ
78K0/KE2マイクロコントローラ
78K0/KF2マイクロコントローラ

目　次

資料番号　U19033JJ1V0AN00（第1版）
発行年月　May 2009 NS

- **本資料に記載されている内容は2009年5月現在のもので，今後，予告なく変更することがあります。**
  量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品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，一切その責を負いません。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。
- 本資料に記載された回路，ソフトウエ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，ソフトウエ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，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。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。当社製品をお客様の機器にご使用の際には，当社製品の不具合の結果として，生命，身体および財産に対する損害や社会的損害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を行ってください。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「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。

  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

  特別水準：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

  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，生命維持のための装置またはシステム等

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。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNEC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う。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E0710J

第1章　概　　要

このサンプル・プログラムでは，PPG出力を使用した16ビットPWM出力の設定を行います。

78K0/Kx2では16ビット・タイマ／イベント・カウンタ00のPPG出力機能を使用して実現します。

本プログラムでは，外部割り込みINTP1でスイッチ入力を検出し，スイッチ入力ごとにデューティを10 [%] ずつ変更します。

なお，PPG出力は，200 [μs] 周期にて出力します。

また，PPG出力機能を使用した16ビットPWM出力は，デューティを変更するために任意のタイミングでCR010を書き換えると1周期の期間でCR010との一致が全く発生しなかったり，一致が2回発生したりするという問題が起こる可能性があります（この場合，デューティが逆転しています）。これを避けるには現状の設定値との関係によりINTTM000の発生タイミング（周期が完了したことを示す）で書き換えたり，INTTM010の発生タイミング（アクティブ期間が完了したことを示す）で書き換えたりする必要があります。

割り込みの基本的な使い方は，以下のようになります。

- INTTM000　：　設定値を大きくする場合に使用する
- INTTM010　：　設定値を小さくする場合に使用する

また，以下のようにこの書き換えタイミングにはあまり余裕がありません。

- 設定値が小さな値の時に設定値を大きくする場合に処理時間がタイトになります。
- デューティが100 %近い時に0 %近くに変更する場合，書き換えタイミングに間に合わなくなります。

(1) PPG設定内容

- タイマ関連の設定
- デューティの設定
- 出力ポートの設定

(2) メイン処理動作の内容

- PPG出力タイマ（16ビット・タイマ／イベント・カウンタ00）を起動し，NOP命令のみの割り込み待ちループに入り，PPG出力をし続けながら，割り込み待ちを行います。

(3) 割り込み処理の内容

【スイッチ入力（INTP1）】

- デューティのデータ（CR010設定値）を格納しているテーブル・アドレスを更新し，実際にデューティの変更を行う割り込みを選択します（INTTM000 or INTTM010）。

【デューティ変更割り込み処理（INTTM000）】

- デューティを大きくする場合の変更処理を行います。

【デューティ変更割り込み処理（INTTM010）】

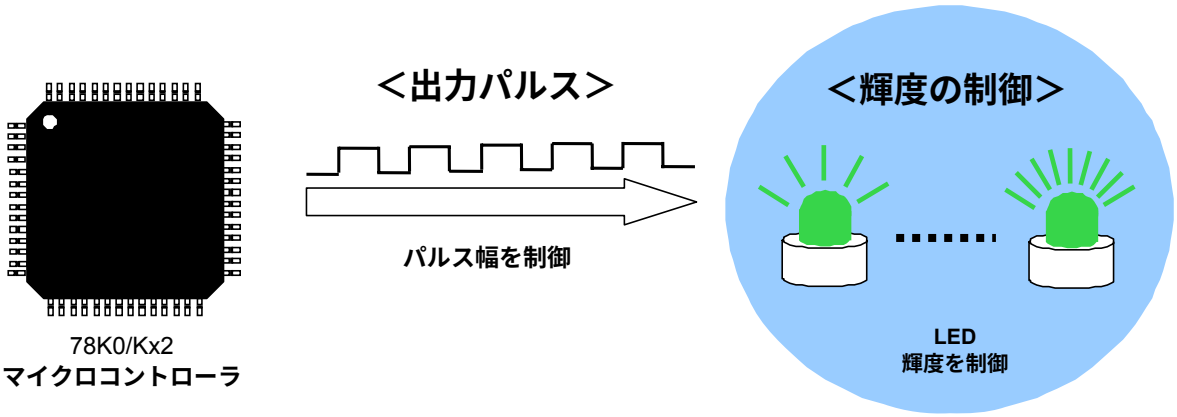
- デューティを小さくする場合の変更処理を行います。

※割り込みベクタは0008H（INTP1），0020H（INTTM000），0022H（INTTM010）に設定します。
（ベクタ・テーブル0008H，0020H，0022H番地の割り込みハンドラ名称を“IINIT”から“IINTP1”，“IINTTM000”，“IINTTM010”に変更します。）

アクティブ・レベルはPPGがハイ・アクティブ，LEDがロウ・アクティブであるのでLEDの輝度は“100－デューティ比”となります。本プログラムでのスイッチ入力回数とPPG出力のデューティおよびLED輝度の関係は以下のようになります。

スイッチ入力回数[回]	デューティ[%]	LED輝度[%]
0	10	90
1	20	80
2	30	70
3	40	60
4	50	50
5	60	40
6	70	30
7	80	20
8	90	10
9	80	20
10	70	30
11	60	40
12	50	50
13	40	60
14	30	70
15	20	80

※16回目以降は，0回から繰り返します。



(4) PPG出力可能なタイマ／カウンタ

PPG出力は，16ビット・タイマのPPG出力機能を使用して実現します。使用できるのは，16ビット・タイマ／イベント・カウンタ00, 01**の2つ**[注]**です。**

注　製品によっては00**のみ使用可能です（次表参照）。**

16ビット・タイマの使用対応デバイス

デバイス		78K0/KB2	78K0/KC2	78K0/KD2	78K0/KE2		78K0/KF2
タイマ		（全製品）	（全製品）	（全製品）	μPD78F0531 μPD78F0532 μPD78F0533	μPD78F0534 μPD78F0535 μPD78F0536 μPD78F0537 μPD78F0537D	（全製品）
16ビット・タイマ／イベント・カウンタ	00	○	○	○	○	○	○
	01	−	−	−	−	○	○

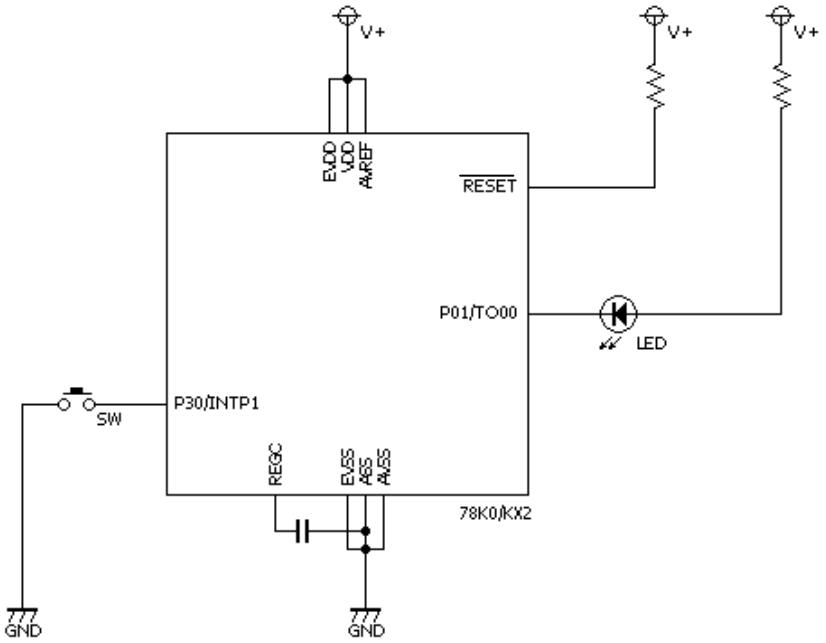
注意　デバイス使用上の注意事項については，78K0/Kx2　**ユーザーズ・マニュアル**を参照してください。**

第2章　回路イメージ

この章では，このサンプル･プログラムで使用する場合の回路イメージおよび周辺ハードウエアを説明します。

2. 1　回路イメージ

回路イメージを次に示します。



注意1. AV$_{REF}$**端子は**V$_{DD}$**に直接接続してください。**

2. AV$_{SS}$**端子は**GND**に直接接続してください。**
3. REGC**端子はコンデンサ（**0.47**～**1 μF**）を介し，**V$_{SS}$**に接続してください。**
4. EV$_{DD}$**端子は**V$_{DD}$**に直接接続してください（**78K0/KE2, 78K0/KF2**のみ）。**
5. EV$_{SS}$**端子は**GND**に直接接続してください（**78K0/KE2, 78K0/KF2**のみ）。**
6. **使用電圧と動作周波数などの詳細については，ユーザーズ・マニュアルを参照してください。**
7. **ポートの出力電流値には上限値があるので,そのスペックの範囲内で駆動する**LED**を使用してください。**

2. 2　マイコン以外の使用デバイス

マイコン以外に使用するデバイスを次に示します。

（1）LED

PPG**出力のデューティにより，**LED**の輝度を制御します。**

（2）スイッチ（INTP1**）**

スイッチ入力を検出することにより，PPG**出力のデューティを**10 [%] **ずつ変更します。**

第3章　ソフトウエアについて

この章では，ダウンロードする圧縮ファイルのファイル構成，使用するマイコンの内蔵周辺機能，サンプル・プログラムのPPG設定と動作概要，およびフロー・チャートを説明します。

3. 1　ファイル構成

ダウンロードする圧縮ファイルのファイル構成は，次のようになっています。

【C言語版】

ファイル名[注]	説 明	同封圧縮（*.zip）ファイル		
				
Kx2_Ppg.c	PPG出力の設定，メイン処理，割り込み処理のソース・ファイル	●	●	
Kx2_func.c	初期化処理を外部関数化したソース・ファイル	●	●	
Kx2_op.asm	オプション・バイト設定用アセンブラ・ソース・ファイル	●	●	
Kx2_Ppg.prw	統合開発環境 PM+用ワーク･スペース・ファイル		●	
Kx2_Ppg.prj	統合開発環境 PM+用プロジェクト・ファイル		●	
Kx2_Ppg.pri Kx2_Ppg.prs Kx2_Ppg.prm	システム・シミュレータ SM+ for 78K0/Kx2用プロジェクト・ファイル			●

注　各ファイル名の”x”部分は，それぞれのデバイスの名前になります。

ex) 78K0/KB2の場合 → ”KB2_Ppg.c”

備考　  ：ソース・ファイルのみ同封

 ：統合開発環境 PM+とシステム・シミュレータ SM+ for 78K0/Kx2で使用するファイルを同封

 ：システム・シミュレータ SM+ for 78K0/Kx2で使用するマイコン動作シミュレーション・ファイルを同封

【アセンブリ言語版】

ファイル名[注]	説 明	同封圧縮（*.zip）ファイル		
				
Kx2_Ppg.asm	PPG出力の設定，メイン処理，割り込み処理のソース・ファイル	●	●	
Kx2_subr.asm	初期化処理をサブルーチン化したソース・ファイル	●	●	
Kx2_op.asm	オプション・バイト設定用アセンブラ・ソース・ファイル	●	●	
Kx2_Ppg.prw	統合開発環境 PM+用ワーク･スペース・ファイル		●	
Kx2_Ppg.prj	統合開発環境 PM+用プロジェクト・ファイル		●	
Kx2_Ppg.pri Kx2_Ppg.prs Kx2_Ppg.prm	システム・シミュレータ SM+ for 78K0/Kx2用プロジェクト・ファイル			●

注　各ファイル名の”x”部分は，それぞれのデバイスの名前になります。

ex) 78K0/KB2の場合 → ”KB2_Ppg.asm”

備考　  ：ソース・ファイルのみ同封

 ：統合開発環境 PM+とシステム・シミュレータ SM+ for 78K0/Kx2で使用するファイルを同封

 ：システム・シミュレータ SM+ for 78K0/Kx2で使用するマイコン動作シミュレーション・ファイルを同封

3. 2　使用するマイコン内蔵周辺機能

このサンプル・プログラムでは，マイコンに内蔵する次の周辺機能を使用します。

・16ビット・タイマ／イベント・カウンタ00
・出力ポート　　：P01/TO00
・スイッチ用入力ポート　　：P30/INTP1

※TM01のサンプル・プログラムは，上記TM00とほぼ同内容であるため，本サンプル・プログラムにおいては省略します。
なお，参考として本サンプル・プログラムにおけるTM00との変更点を以下に記載します。

【TM00→TM01への変更点】

① C言語　　：割り込み処理関数宣言のINTTM000とINTTM010をそれぞれINTTM001とINTTM011に変更
アセンブリ言語　　：割り込みベクタテーブルの0020H（INTTM000）と0022H（INTTM010）をそれぞれ0038H（INTTM001）と003AH（INTTM011）に変更
② C，アセンブリ言語　　：“ PPG出力の初期化 ”でのTM00の各レジスタ名称の00を01に変更
③ C，アセンブリ言語　　：“ PPG出力の初期化 ”での割り込み要求フラグと割り込みマスク・フラグの名称の00を01に変更
④ C，アセンブリ言語　　：“ PPG出力の初期化 ”でP01とPM01（TO00）をP06とPM06（TO01）に変更
⑤ C，アセンブリ言語　　：各割り込み処理内でのTM00の割り込み要求フラグと割り込みマスク・フラグ，各レジスタの名称語尾の0を1に変更

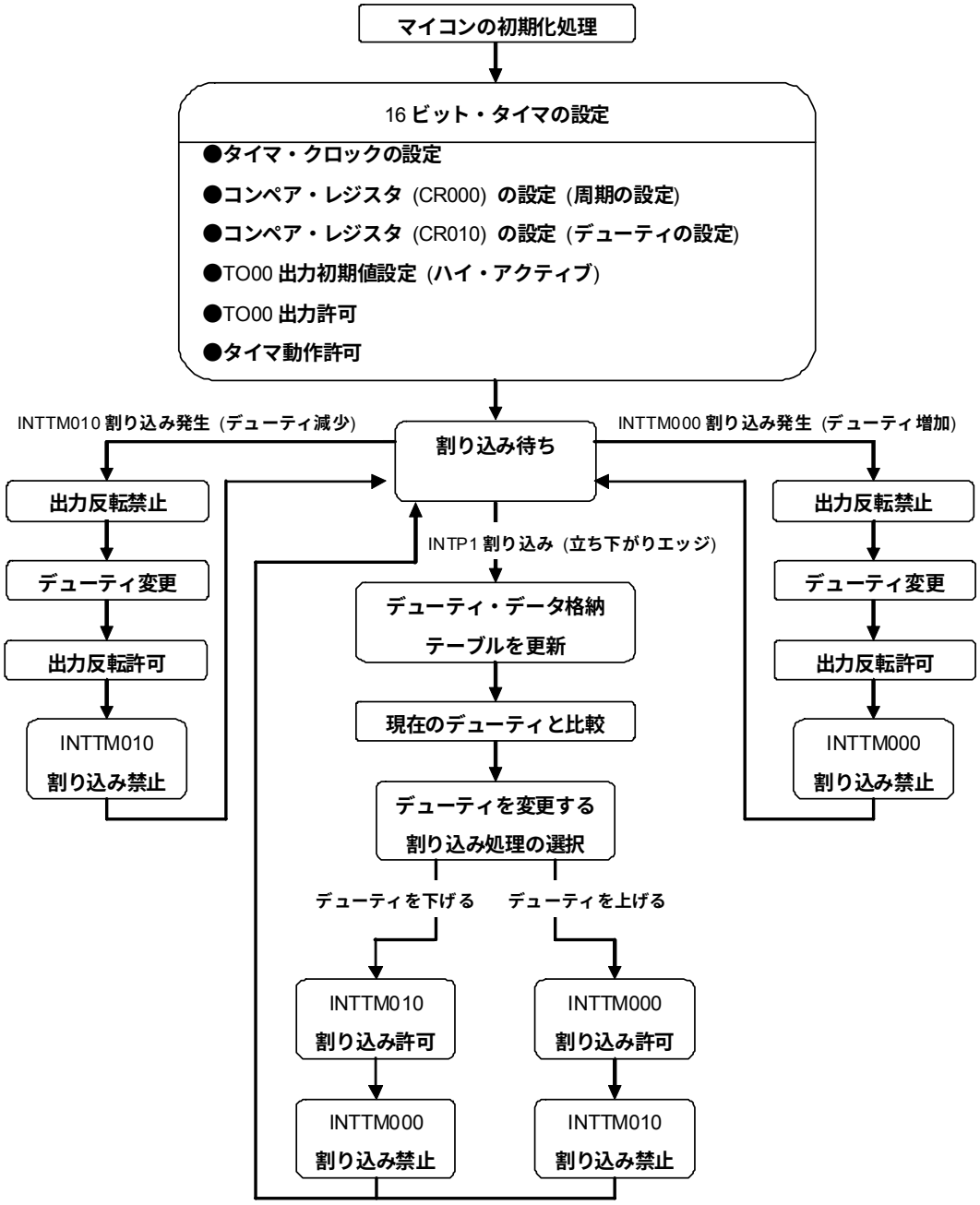
注意　本サンプル・プログラムではタイマのカウント・クロックを2 [MHz] に設定し，PPG出力周期を200 [μs] に設定してありますが，16ビット・タイマ／イベント・カウンタ01（TM01）を使用するとカウント・クロックを選択するレジスタの内容も変わるために割り込み時間にも変化が生じてしまうので，カウント・クロックを選択するレジスタの設定とコンペア・レジスタの設定値は使用するタイマ／カウンタに合わせて設定してください。また，カウント・クロックが変更されるとINTTM010処理時間分の補正値C_OFFSET（C言語）およびCOFFSET（アセンブリ言語）の設定値も適正な値に変更する必要があります。

3. 3　PPG設定と動作概要

このサンプル・プログラムでは，16ビット・タイマ／イベント・カウンタ00のPPG出力機能を使用して，16ビットPWM出力を実現します。

初期設定完了後にPWM出力を開始し，スイッチ入力（INTP1）ごとにデューティを10 %ずつ増減します。

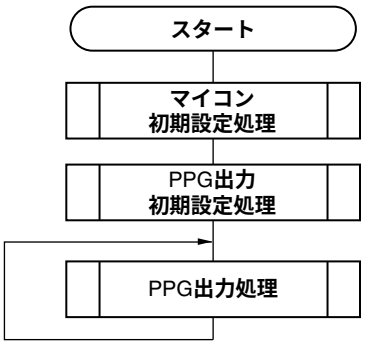
動作概要については，次の状態遷移図（ステート・チャート）に示します。



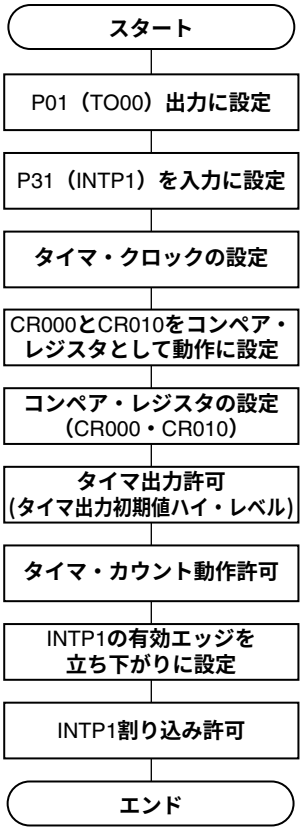
3. 4　フロー・チャート

このサンプル・プログラムのフロー・チャートを次に示します。

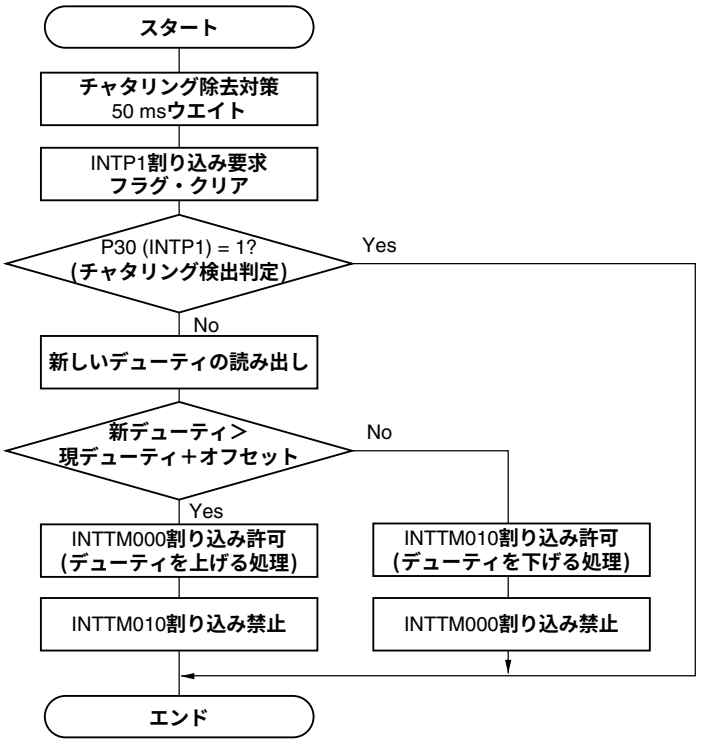
○ゼネラル・フロー (C**言語版：**Kx2_Ppg.c　**アセンブリ言語版：**Kx2_PPG.asm)



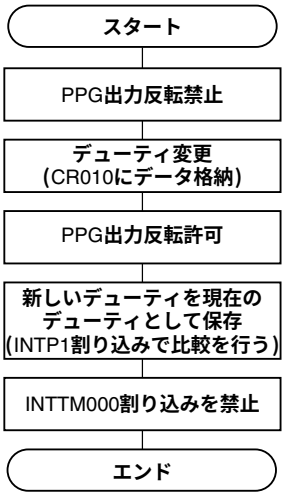
○PPG出力初期設定処理 (C**言語版：**Kx2_Ppg.c　**アセンブリ言語版：**Kx2_PPG.asm)



○INTP1割り込み処理 (C言語版：Kx2_Ppg.c　アセンブリ言語版：Kx2_Ppg.asm)



○INTTM000 (デューティを上げる) 割り込み処理 (C言語版：Kx2_Ppg.c　アセンブリ言語版：Kx2_Ppg.asm)



○INTTM010 (デューティを下げる) 割り込み処理 (C言語版：Kx2_Ppg.c　アセンブリ言語版：Kx2_Ppg.asm)

第4章　設定方法について

この章では，PPG（16ビット・カウンタ／イベント・カウンタ00）の設定，およびメイン処理について説明します。

レジスタ設定方法の詳細については，78K0/Kx2　**ユーザーズ・マニュアル**を参照してください。

アセンブラ命令については，78K/0**シリーズ　命令編　ユーザーズ・マニュアル**を参照してください。

4. 1　前処理指令

C言語において，SFR領域に関する操作，CPU制御命令，割り込み関数などを使用するためには，#pragma指令にてソース・プログラムの冒頭に前処理指令を記述する必要があります。本サンプル・プログラムで使用する前処理指令は以下のとおりです。

【C言語】 (Kx2_Ppg.c)

```
/*==========================================================================
        前処理指令(#pragma指令)
==========================================================================*/
#pragma  sfr
#pragma  di
#pragma  ei
#pragma  nop
#pragma  interrupt  INTP1      fn_intp1
#pragma  interrupt  INTTM000   fn_inttm000
#pragma  interrupt  INTTM010   fn_inttm010
```

特殊機能レジスタ（SFR）名称を記述可能にします。

DI命令を記述可能にします。

EI命令を記述可能にします。

NOP命令を記述可能にします。

割り込み関数宣言をします。

4. 2　PPG出力（TM00）の設定

PPG出力（TM00）は，次の項目を設定します。

（1）タイマ・クロック選択

（2）“ CR000およびCR010をコンペア・レジスタとして動作 ” に設定

（3）タイマ・コンペア・レジスタの設定（CR000, CR010）

（4）タイマ出力の設定，タイマ出力反転条件の設定

（5）タイマの動作許可

本サンプル・プログラムでは後述の【例】の内容で設定しています。

（1）カウント・クロックの設定

TM00のカウント・クロック，およびTI000, TI001端子入力の有効エッジを設定します。

図4－1　プリスケーラ・モード・レジスタ00

PRM00

ES101	ES100	ES001	ES000	0	0	PRM001	PRM000

カウント・クロックの選択

0	0	f_{PRS}
0	1	$f_{PRS}/2^2$
1	0	$f_{PRS}/2^8$
1	1	TI000有効エッジ

TI000端子の有効エッジ

0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	設定禁止
1	1	立ち上がり／立ち下がりの両エッジ

TI010端子の有効エッジ

0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	設定禁止
1	1	立ち上がり／立ち下がりの両エッジ

注意　外部クロックには，内部クロック（f_{PRS}）**の2周期分より長いパルスが必要です。**

備考　f_{PRS} ： 周辺ハードウエア・クロック周波数

(2) コンペア・レジスタの制御の設定

CR00, CR01の動作を制御します。

図4－2　キャプチャ／コンペア・コントロール・レジスタ00

CRC00

0	0	0	0	0	CRC002	CRC001	CRC000

CR000の動作モード選択

0	コンペア・レジスタとして動作
1	キャプチャ・レジスタとして動作

CR000のキャプチャ・トリガの選択

0	TI010端子の有効エッジでキャプチャする
1	TI000端子の有効エッジの逆相でキャプチャする

CR010の動作モード選択

0	コンペア・レジスタとして動作
1	キャプチャ・レジスタとして動作

注意1. TI010端子から有効エッジが検出された場合，キャプチャ動作は行われませんが，外部割り込み信号としてINTTM000信号が発生します。

2. TMC003, TMC002 = 11を設定した場合は，CRC000には必ず0を設定してください。

(3) 16ビット・タイマ・キャプチャ／コンペア・レジスタ（CR000/CR010）の設定

コンペア・レジスタとして使用するときはCR000（CR010）に設定した値とTM00を常に比較し，一致したときに割り込み信号を発生します。

図4－3　16ビット・タイマ・キャプチャ／コンペア・レジスタ000 (010)

CR000/CR010

注意　PPG出力として動作する場合は，CR010＜CR000≦FFFFHの範囲で設定します。

PPG出力によって生成されるパルス周期，デューティは次のようになります。

- パルス周期 = (CR000の設定値＋1)×カウント・クロック周期
- デューティ = (CR010の設定値＋1) / (CR000の設定値＋1)

(4) タイマ出力の設定

TOC00端子出力を制御します。

図4－4　16ビット・タイマ出力コントロール・レジスタ00

TOC00

0	OSPT00	OSPE00	TOC004	LVS00	LVR00	TOC001	TOE00

TO00端子出力制御

0	出力禁止（TO00端子出力はロウ・レベルに固定）
1	出力許可

CR000とTM00の一致によるTO00端子出力の制御

0	反転動作禁止
1	反転動作許可

TO00端子の状態設定

0	0	変化しない
0	1	TO00端子出力初期値ロウ・レベル(0)
1	0	TO00端子出力初期値ハイ・レベル(1)
1	1	設定禁止

CR010とTM00の一致によるTO00端子出力の制御

0	反転動作禁止
1	反転動作許可

ワンショット・パルス出力動作の制御

0	連続パルスの出力
1	ワンショット・パルス出力

ソフトウエアによるワンショット・パルス出力トリガ

0	—
1	ワンショット・パルス出力

注意1. TOC001 = 0**でも割り込み信号（**INTTM000**）は発生します。**

2. TOC004 = 0**でも割り込み信号（**INTTM010**）は発生します。**

3. **ワンショット・パルス出力は，フリーランニング・タイマ・モード，**TI000**端子の有効エッジ入力でクリア&スタート・モードのときに正常に動作します。**

4. OSPT00**は，リード値は常に**0**です。ワンショット・パルス出力以外ではセット**(1)**しないでください。**

(5) タイマ・モードの設定

16ビット・カウンタ／イベント・カウンタ00の動作モード，カウンタ制御，出力の状態／制御の設定をします。

図4－5　16ビット・タイマ・モード・コントロール・レジスタ00

TMC00

0	0	0	0	TMC003	TMC002	TMC001	OVF00

オーバフロー・フラグ

0	OVFへの0クリア，またはTMC003，TMC002=00
1	オーバフロー発生

タイマ出力反転条件

0	TM00とCR000の一致，TM00とCR010の一致
1	TM00とCR000の一致，TM00とCR010の一致 TI000端子の有効エッジのトリガ入力

16ビット・タイマ／イベント・カウンタの動作許可

0	0	カウンタ動作禁止。動作クロック供給停止。カウンタクリア
0	1	フリー・ランニング・タイマ・モード
1	0	TI000端子の有効エッジ入力でクリア&スタート・モード
1	1	TM00とCR000の一致でクリア&スタート・モード

注意　OVF00はすべての動作モードで，TM00の値がFFFFHから0000Hになるときセット (1) されます。1を書き込むことでもセットできます。

【例】　・タイマ・クロックを0.5 [μs] に設定（8 MHz動作時）

・CR000・CR010をコンペア・レジスタとして動作に設定

・コンペア・レジスタ（CR000・CR010）の初期設定

→PPG出力周期初期値（CR000）= 200 [μs]，PPG出力デューティ初期値 = 10 [%] に設定

・タイマ出力設定

・タイマ動作許可

（サンプル・プログラムの設定と同内容）

PRM00

0	0	0	0	0	0	0	1

カウント・クロックの選択

0	1	$f_{PRS}/2^2$

TI000端子の有効エッジの選択

0	0	立ち下がりエッジ

TI010端子の有効エッジの選択

0	0	立ち下がりエッジ

CRC00

0	0	0	0	0	0	0	0

CR000の動作モードの選択

0	コンペア・レジスタとして動作

CR000のキャプチャ・トリガの選択

0	TI010端子の有効エッジでキャプチャする

CR010の動作モードの選択

0	コンペア・レジスタとして動作

CR000

0	0	0	0	0	0	0	1	1	0	0	0	1	1	1	1

PPGパルス周期 =（399 + 1）×カウント・クロック周期

※ここでは，399（200 [μs] ）に設定

CR010

0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1

PPGパルス・デューティ = (39 + 1) / (PPGパルス周期 + 1)

※ここでは，39 (10 [%]) に設定

TOC00

0	0	0	1	0→1	0	1	0→1

TO00出力制御

0	出力禁止（TO00出力は，ロウ・レベルに固定）→許可

CR000とTM00の一致によるTO00の出力の制御

1	反転動作許可

TO00の出力状態の設定

0	0	変化しない→TO00出力初期値ハイ・レベル

CR010とTM00の一致によるTO00出力の制御

0	反転動作許可

ワンショット・パルス出力動作の制御

0	連続パルス出力

ソフトウエアによるワンショット・パルス出力トリガ

0	−

TMC00

0	0	0	0	1	1	0	0

TM00のオーバフロー・フラグ

0	OVF00への0クリアまたはTMC003, TMC002 = 00

タイマ出力（TO00）の反転条件

0	TM00とCR000の一致，TM00とCR010の一致

16ビット・タイマ／イベント・カウンタ00の動作許可

1	1	TM00とCR000の一致でクリア&スタート・モード

サンプル・プログラムでは以下のようになります。

【C言語】　(Kx2_Ppg.c)

```
/*========================================================================
;       PPG出力の定義
;    ここでは，PPGパルス周期，PPG出力デューティ（パルス幅）初期値，処理時間補正値，
;    デューティ・テーブルの最終要素番号を定義する。
;========================================================================*/
/*------------------------------------------------------------------------
;       PPGパルス周期の指定(CR000)
;   PPGパルス周期 = 400/(8000000/2^2)
;------------------------------------------------------------------------*/
        #define C_PPG_CYCLE     400     /* PPGパルス出力周期200μs*/
/*------------------------------------------------------------------------
;       PPG出力デューティ（パルス幅）初期値の指定(CR010)
;   PPG出力デューティ = 40/400×100
;------------------------------------------------------------------------*/
        #define C_PPG_INIT      C_PPG_CYCLE/10
                                        /* デューティ10%(40)からスタート     */
/*------------------------------------------------------------------------
;       INTTM010処理時間分補正値の指定
;------------------------------------------------------------------------*/
        #define C_Offset        46/4    /* INTTM010の処理時間分の補正用*/
/*------------------------------------------------------------------------
;       デューティ・テーブルの最終要素番号の指定
;  定数定義にて定義するデューティテーブル（配列）の要素数を変更する際には，ここで
;  定義する最終要素番号も同時に変更する。
;------------------------------------------------------------------------*/
        #define C_MaxLen        15
                                •
                                •
                                •
        PRM00   =       0b00000001;     /*カウント・クロックの設定   */
        CRC00   =       0b00000000;     /*CR000・CR010をコンペア・レジスタ
                                          に設定                    */
        CR000   =       C_PPG_CYCLE-1;  /*コンペア・レジスタの設定   */
        CR010   =       C_PPG_INIT-1;   /*コンペア・レジスタの設定   */
        TOC00   =       0b00010010;     /*タイマ出力の設定          */
        TOE00   =       1;              /*タイマ出力許可            */
        LVS00   =       1;              /*TO00出力初期値ハイ・レベル*/
        TMC00   =       0b00001100;     /*タイマ動作許可            */
                                •
                                •
                                •
```

【アセンブリ言語】 (Kx2_PPG.asm)

```
;=============================================================================
;
;       PPG出力の定義
;
;    ここでは,PPGパルス周期,PPG出力デューティ(パルス幅)初期値,処理時間補正値を
;    定義する。
;    (周期,デューティ初期値を変更する場合には,下記に示す各計算式により計算し,PPG
;    パルス周期を決定した後にデューティ初期値を決定する。)
;=============================================================================
;-----------------------------------------------------------------------------
;
;       PPGパルス周期の指定(CR000)
;
;    PPGパルス周期 = 400/(8000000/2^2)
;    ※本プログラムではメインクロックを8MHz,カウント・クロックをfPRS/2^2としているので
;    上記の計算式となる。なお,ここでは周期200μsとした。
;-----------------------------------------------------------------------------
        CPPGCYCLE       EQU     400     ;PPGパルス出力周期200μs

;-----------------------------------------------------------------------------
;
;       PPG出力デューティ(パルス幅)初期値の指定(CR010)
;
;    PPG出力デューティ = 40/400×100
;    ※本プログラムではPPGパルス周期を400と定義したので,上記の計算式となる。
;   ここではデューティ初期値10%(パルス幅20μs)とする。
;-----------------------------------------------------------------------------
        CPPGINIT        EQU     CPPGCYCLE/10
                                        ;デューティ10%(40)からスタート

;-----------------------------------------------------------------------------
;
;       INTTM010処理時間分補正値の指定
;
;    本プログラムでは,INTTM010割り込み処理内での出力反転許可までの命令クロック
;    数+割り込み応答時間が46で,TM00のカウント・クロックをfPRS/2^2(周辺ハードウェア
;    クロックを4分周)としているので,以下の値となる。
;-----------------------------------------------------------------------------
        COFFSET         EQU     46/4    ;INTTM010の処理時間分の補正用
                                        •
                                        •
                                        •
        MOV     PRM00,  #00000001B      ;カウント・クロックの設定
        MOV     CRC00,  #00000000B      ;CR000・CR010をコンペア・レジスタに設定
        MOVW    CR000,  #CPPGCYCLE-1    ;コンペア・レジスタの設定
        MOVW    CR010,  #CPPGINIT-1     ;コンペア・レジスタの設定
        MOV     TOC00,  #00010010B      ;タイマ出力の設定
        SET1    TOE00                   ;タイマ出力許可
        SET1    LVS00                   ;TO00出力初期値ハイ・レベル
        MOV     TMC00,  #00001100B      ;タイマ動作許可
                                        •
                                        •
                                        •
```

4. 3　割り込みの設定

(1) 割り込み要求の設定

指定の割り込みに対して，割り込み要求フラグをクリアします。

このサンプル・プログラムでは，直接レジスタ・ビットに設定しています。

図4－6　割り込み要求フラグ・レジスタ（IF0L）のフォーマット

IF0L

SREIF6	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0	LVIIF

割り込み要求フラグ

0	要求クリア
1	要求セット

図4－7　割り込み要求フラグ・レジスタ（IF0H）のフォーマット

IF0H

TMIF01	TMIF00	TMIF50	TMIFH	TMIFH	DUALI	STIF6	SRIF6
0	0		0	1	F0		

割り込み要求フラグ

0	要求クリア
1	要求セット

(2) 割り込みマスクの設定

指定の割り込み処理の許可／禁止を設定します。

このサンプル・プログラムでは，直接レジスタ・ビットに設定しています。

図4－8　割り込みマスク・フラグ・レジスタ（MK0L）のフォーマット

MK0L

SREMK	PMK5	PMK4	PMK3	PMK2	PMK1	PMK0	LVIMK
6							

割り込みマスク・フラグ

0	割り込み処理許可
1	割り込み処理禁止

図4－9　割り込みマスク・フラグ・レジスタ（MK0H）のフォーマット

MK0H

TMMK0	TMMK0	TMMK5	TMMK	TMMK	DUALM	STMK6	SRMK6
10	00	0	H0	H1	K0		

割り込みマスク・フラグ

0	割り込み処理許可
1	割り込み処理禁止

(3) 外部割り込みの有効エッジの設定

外部割り込みの有効エッジ（立ち上がりエッジ／立ち下がりエッジ／両エッジ）を設定します。

この設定は下記の2つのレジスタを組み合わせて設定します。

このサンプル・プログラムでは，直接レジスタ・ビットに設定し，立ち下がりエッジを選択しています。

図4－10　外部割り込み立ち上がりエッジ許可レジスタのフォーマット

EGP

EGP7	EGP6	EGP5	EGP4	EGP3	EGP2	EGP1	EGP0

図4－11　外部割り込み立ち下がりエッジ許可レジスタのフォーマット

EGN

EGN7	EGN6	EGN5	EGN4	EGN3	EGN2	EGN1	EGN0

EGPn	EGNn	INTPnの有効エッジ選択
0	0	エッジ検出禁止
0	1	立ち下がりエッジ
1	0	立ち上がりエッジ
1	1	両エッジ

（n=0-7）

4. 4　ポートの設定

(1) ポートの設定

このサンプル・プログラムではPPG出力用にP01/TO00を出力として使用し，スイッチ入力用にP30/INTP1を入力として使用しています。サンプル・プログラムでは，後述の【例】のような内容で設定しています。

図4－12　ポート・モード・レジスタ3（PM3）のフォーマット

PM3

PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30

P3n（n = 0-7）端子の入出力モードの選択

0	出力モード
1	入力モード

図4－13　ポート・レジスタ3（P3）のフォーマット

P3

P37	P36	P35	P34	P33	P32	P31	P30

P3n（n = 0-7）端子の出力ラッチのレベル選択

0	ロウ・レベル出力
1	ハイ・レベル出力

図4－14　プルアップ抵抗オプション・レジスタ3（PU3）のフォーマット

PU3

PU37	PU36	PU35	PU34	PU33	PU32	PU31	PU30

P3n（n = 0-7）端子の内蔵プルアップ抵抗の選択

0	内蔵プルアップ抵抗を接続しない
1	内蔵プルアップ抵抗を接続する

図4－15　ポート・モード・レジスタ0（PM0）のフォーマット

PM0

PM07	PM06	PM05	PM04	PM03	PM02	PM01	PM00

P0n（n = 0-7）端子の入出力モードの選択

0	出力モード
1	入力モード

図4－16　ポート・レジスタ0（P0）のフォーマット

P0

P07	P06	P05	P04	P03	P02	P01	P00

P0n（n = 0-7）端子の出力ラッチのレベル選択

0	ロウ・レベル出力
1	ハイ・レベル出力

【例】　・P30を入力ポートに設定（スイッチ入力）

・P30に内蔵プルアップ抵抗を接続（スイッチ入力）

・P01を出力ポートに設定（タイマ出力）

（サンプル・プログラムの設定と同内容）

PM3

0	0	0	0	0	0	0	1

P30端子の入出力モードの選択

1	入力モード

P3

0	0	0	0	0	0	0	0

PU3

0	0	0	0	0	0	0	1

P30端子の内蔵プルアップ抵抗の選択

1	内蔵プルアップ抵抗を接続する

PM0

0	0	0	0	0	0	0	0

P01端子の入出力モードの選択

0	出力モード

P0

0	0	0	0	0	0	0	0

・PM0の設定値は「00000000」，P0の設定値は「00000000」となります。

・PM3の設定値は「00000001」，PU3の設定値は「00000001」，P3の設定値は「00000000」となります。

【C言語】

```
PM0     =       0b00000000;
P0      =       0b00000000;
PM3     =       0b00000001;
PU3     =       0b00000001;
P3      =       0b00000000;
```

【アセンブリ言語】

```
MOV     PM0,    #00000000B
MOV     P0,     #00000000B
MOV     PM3,    #00000001B
MOV     PU3,    #00000001B
MOV     P3,     #00000000B
```

4.5　メイン処理

メイン処理では次の動作を行います。

【C言語】　(Kx2_Ppg.c)

・PPG出力設定関数を呼び出します（タイマ処理を開始します）。

・メイン処理内では何もせず，割り込みを待つのでNOP命令のみのループ処理になります。

```
/*********************************************************************
;
;       メインループ
;       (PPG出力をし続け,各割り込みを待つ)
;*********************************************************************/
void main(void){

        fn_InitTimer();                 /* PPG出力の初期化          */
        EI();                           /* ベクタ割り込み許可       */

        while (1){
                NOP();
        }
}
```

PPG出力の初期化を行い，処理を開始します。

メイン処理内ではNOP命令のみで割り込みを待ちます。

【アセンブリ言語】　(Kx2_Ppg.asm)

・メイン処理内では何もせず，割り込みを待つのでNOP命令のみのループ処理になります。

```
;*********************************************************************
;
;       メインループ
;       (PPG出力をし続け,各割り込みを待つ)
;*********************************************************************
        EI                              ;ベクタ割り込み許可
MMAIN:
        NOP                             ;何もしないで,割り込み待ち

        BR              MMAIN
```

メイン処理内ではNOP命令のみで割り込みを待ちます。

※アセンブリ言語版でのPPG出力の初期化は，メイン処理に入る前に，PPG出力の設定，割り込みの設定およびポートの設定で示したアセンブリ言語版の設定例と同じ内容で初期化しています。

4.6　変数・定数の定義

ここでは，本サンプル・プログラムで使用する変数および定数の定義について説明します。

【C言語】　(Kx2_Ppg.c)

C言語版では次のような定義をします。

・PPG出力の定義（周期，デューティ（パルス幅）の初期値，処理時間補正値，デューティ・テーブルの最終要素番号）。

・変数の定義（スイッチ入力カウント用8ビット変数，新しいデューティ保存用変数，現在のデューティ保存用変数）

・PPG出力デューティ変更用データ・テーブル（デューティ・テーブル）

```
/*---------------------------------------------------------------------
;       PPGパルス周期の指定(CR000);
;---------------------------------------------------------------------*/
        #define  C_PPG_CYCLE   400              /* PPGパルス出力周期200μs*/
/*---------------------------------------------------------------------
;       PPG出力デューティ(パルス幅)初期値の指定(CR010)
;---------------------------------------------------------------------*/
        #define  C_PPG_INIT    C_PPG_CYCLE/10
                                               /* デューティ10%(40)からスタート*/
/*---------------------------------------------------------------------
;       INTTM010処理時間分補正値の指定
;---------------------------------------------------------------------*/
        #define  C_Offset      46/4             /* INTTM010の処理時間分の補正用*/
/*---------------------------------------------------------------------
;       デューティ・テーブルの最終要素番号の指定
;---------------------------------------------------------------------*/
        #define  C_MaxLen                       15
                                                .
                                                .
                                                .
/*=====================================================================
;       使用する変数・定数の定義
;=====================================================================*/
sreg unsigned char g_ucSWcnt = 0;               /* スイッチ入力カウント用8ビット変数     */
sreg unsigned int g_unNextD = C_PPG_INIT-1;     /* 新しいデューティ(パルス幅)保存用変数  */
sreg unsigned int g_unCurrentD = C_PPG_INIT-1;  /* 現在のデューティ(パルス幅)保存用変数  */
const unsigned int g_unOutData[C_MaxLen + 1] =
{C_PPG_INIT-1,   C_PPG_INIT*2-1,C_PPG_INIT*3-1,C_PPG_INIT*4-1,
 C_PPG_INIT*5-1,C_PPG_INIT*6-1,C_PPG_INIT*7-1,C_PPG_INIT*8-1,
 C_PPG_INIT*9-1,C_PPG_INIT*8-1,C_PPG_INIT*7-1,C_PPG_INIT*6-1,
 C_PPG_INIT*5-1,C_PPG_INIT*4-1,C_PPG_INIT*3-1,C_PPG_INIT*2-1};
                                               /* PPG出力デューティ変更用データ・テーブル*/
```

【計算式】
周期（200us） = 400/(8000000/2^2)
（カウント・クロック）

【計算式】
デューティ（10%） = 40/400×100
（周期設定値）

極端に小さな値（INTTM000の割り込み処理時間＋割り込み応答時間内に入り込む値＝約3 %以下）に設定すると，デューティを上げる処理の際に不要な反転を引き起こす可能性があるので注意してください。

INTTM010割り込み処理内での
（出力反転許可までの命令クロック数
＋割り込み応答時間）/4
TM00のカウント・クロックの4分周

デューティのデータ・テーブルの要素数を変更する際には，ここで定義する最終要素番も同時に変更してください。

スイッチ入力ごとにデューティを10 %ずつ変化させるためのCR010の設定値を定義します。
注意
設定値を変更する際には，各設定値の計算値がCR010＜CR000（デューティ100 %未満）となるようにしてください。

【アセンブリ言語】　(Kx2_Ppg.asm)

アセンブリ言語版では次のような定義をします。

・PPG出力の定義（周期，デューティ（パルス幅）の初期値，処理時間補正値）。

・変数の定義（新しいデューティ保存用変数，現在のデューティ保存用変数）。

・PPG出力デューティ変更用データ・テーブル

```
;-------------------------------------------------------------
;       PPGパルス周期の指定(CR000)
;-------------------------------------------------------------
        CPPGCYCLE       EQU     400     ;PPGパルス出力周期200μs
;-------------------------------------------------------------
;       PPG出力デューティ(パルス幅)初期値の指定(CR010)
;-------------------------------------------------------------
        CPPGINIT        EQU     CPPGCYCLE/10    ;デューティ10%(40)からスタート
;-------------------------------------------------------------
;       INTTM010処理時間分補正値の指定
;-------------------------------------------------------------
        COFFSET         EQU     46/4    ;INTTM010の処理時間分の補正用
                                •
                                •
;=============================================================
;       ROMの定義(PPG出力デューティのデータ定義テーブル)
;=============================================================
XROM    CSEG    AT      0100H
TPWIDTH:
        DW      CPPGINIT-1              ;デューティ10%(パルス幅 20μs)
        DW      CPPGINIT*2-1            ;デューティ20%(パルス幅 40μs)
        DW      CPPGINIT*3-1            ;デューティ30%(パルス幅 60μs)
        DW      CPPGINIT*4-1            ;デューティ40%(パルス幅 80μs)
        DW      CPPGINIT*5-1            ;デューティ50%(パルス幅100μs)
        DW      CPPGINIT*6-1            ;デューティ60%(パルス幅120μs)
        DW      CPPGINIT*7-1            ;デューティ70%(パルス幅140μs)
        DW      CPPGINIT*8-1            ;デューティ80%(パルス幅160μs)
        DW      CPPGINIT*9-1            ;デューティ90%(パルス幅180μs)
        DW      CPPGINIT*8-1            ;デューティ80%(パルス幅160μs)
        DW      CPPGINIT*7-1            ;デューティ70%(パルス幅140μs)
        DW      CPPGINIT*6-1            ;デューティ60%(パルス幅120μs)
        DW      CPPGINIT*5-1            ;デューティ50%(パルス幅100μs)
        DW      CPPGINIT*4-1            ;デューティ40%(パルス幅 80μs)
        DW      CPPGINIT*3-1            ;デューティ30%(パルス幅 60μs)
TPWIDTH_END:
        DW      CPPGINIT*2-1            ;デューティ20%(パルス幅 40μs)
```

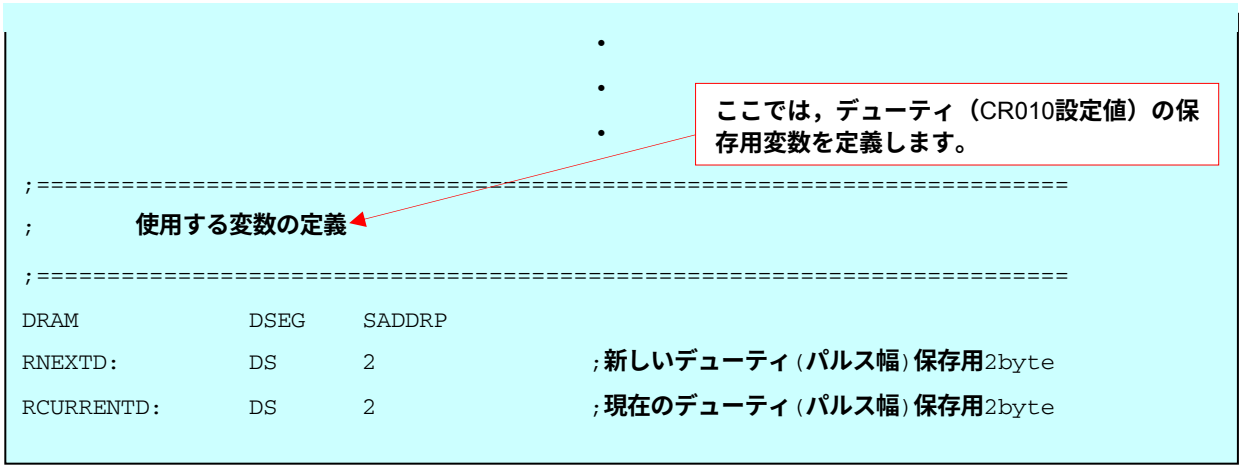
【計算式】
周期（200us） = 400/(8000000/2^2)（カウント・クロック）

【計算式】
デューティ（10%） = 40/400×100（周期設定値）

極端に小さな値（INTTM000の割り込み処理時間＋割り込み応答時間内に入り込む値＝約3 %以下）に設定すると,デューティを上げる処理の際に不要な反転を引き起こす可能性があるので注意してください。

INTTM010割り込み処理内での（出力反転許可までの命令クロック数＋割り込み応答時間）/4
TM00のカウント・クロックの4分周

スイッチ入力ごとにデューティを10 %ずつ変化させるためのCR010の設定値を定義します。
注意
設定値を変更する際には，各設定値の計算値がCR010＜CR000（デューティ100 %未満）となるようにしてください。




```
                                                    •
                                                    •
                                                    •
;=========================================================================
;           使用する変数の定義
;=========================================================================
DRAM            DSEG    SADDRP
RNEXTD:         DS      2               ;新しいデューティ（パルス幅）保存用2byte
RCURRENTD:      DS      2               ;現在のデューティ（パルス幅）保存用2byte
```

4. 7　割り込み処理

4. 7. 1　INTP1割り込み処理（スイッチ入力処理）

INTP1割り込み処理では，次の動作を行います。

【C言語】

・チャタリング除去（50 [ms] ）

・デューティのデータ・テーブルを更新

・現在のデューティと比較して変更する割り込み処理を選択

・実際にデューティを変更する割り込みを許可（INTTM000 or INTTM010）

```
/************************************************************************
;       INTP1割り込み処理
;************************************************************************/
__interrupt    void    fn_intp1(void){
        unsigned       char       ucChat;    /* チャタリング対策用変数              */
        for (ucChat = 0; ucChat < 250+1; ucChat++){
                                              /* チャタリング対策50msウェイトループ
                                                 (200us×250＝50ms)                  */
                TMIF000 = 0;                  /* INTTM000割り込みフラグクリア        */
                while (!TMIF000){             /* INTTM000割り込み待ち                */
                        NOP();                /*(命令クロック分は無視してここでチャ
                                                 タリング対策ウェイトを作る→200μs)  */
                }
        }
```

200 [μ s] ごとのINTTM000割り込みの割り込み要求フラグ（TMIF000）を利用し，250回ループを回すことで50 [ms] のチャタリング除去ウェイトを作成します。

```
        PIF1    =       0;                    /* INTP1割り込み要求をクリア           */
        if(!P3.0){                            /* チャタリング検出判定
                                                 スイッチ入力があれば処理実行       */
                g_ucSWcnt++;                  /* スイッチ入力回数をカウント+1        */
                if(g_ucSWcnt == C_MaxLen  + 1){
                                              /*デューティ・テーブルの最終要素番号を */
                        g_ucSWcnt =           0;/*超えたらをg_ucSWcntを初期化       */
                }
                g_unNextD         =           g_unOutData[g_ucSWcnt];
                                              /* 新しいデューティデータを保存       */
                if(g_unNextD > g_unCurrentD + C_Offset){
                                              /* 新デューティ＜現在のデューティ?     */
```

・

・

新デューティ＞現在のデューティの処理

```
                TMIF000     =     0;
                                  /*INTTM000割り込み要求フラグをクリア  */
                TMMK000     =     0;
                                  /* INTTM000割り込みマスクを解除       */
                TMMK010     =     1;
                                  /* INTTM010割り込みをマスク           */
          }
```

デューティを上げる処理

新デューティ≦現在のデューティの処理

```
          else{
                TMIF010     =     0;
                                  /*INTTM010割り込み要求フラグをクリア  */
                TMMK010     =     0;
                                  /* INTTM010割り込みマスクを解除       */
                TMMK000     =     1;
                                  /* INTTM000割り込みをマスク           */
          }
     }
}
```

デューティを下げる処理

【アセンブリ言語】

・チャタリング除去（50 [ms] ）
・デューティのデータ・テーブルのアドレスを更新
・現在のデューティと比較して変更する割り込み処理を選択
・実際にデューティを変更する割り込みを許可（INTTM000 or INTTM010）

```
        MOVW    AX,             CR010
        MOVW    RCURRENTD,       AX
        MOVW    HL,             #TPWIDTH
                                        •
                                        •
                                        •
;*************************************************************************
;       INTP1割り込み処理
;*************************************************************************
IINTP1:
        PUSH    AX                      ;AXレジスタのデータをスタックへ退避
        MOV     A,      #250+1          ;チャタリング対策の50msウェイト
                                        ;(200us×250＝50ms)
HWAIT_CHAT:
        CLR1    TMIF000                 ;INTTM000割り込み要求フラグクリア
HWAIT_INT:
        BF      TMIF000,$HWAIT_INT      ;INTTM000割り込み待ち
                                        ;(200μs周期→命令クロック分は無視して
                                        ;ここでチャタリング対策ウェイトを作る)
        DEC     A                       ;Aレジスタをデクリメント
        BNZ     $HWAIT_CHAT             ;A = 0でなければHWAIT_CHATに分岐
        CLR1    PIF1                    ;INTP1割り込み要求フラグクリア
        BT      P3.0,   $HEND_INTP1     ;チャタリング検出判定
                                        ;スイッチ入力がなければ分岐する
;-------------------------------------------------------------------------
;       新しいデューティの読み出し
;-------------------------------------------------------------------------
        MOVW    AX,     HL              ;テーブル・アドレスを読み出し
        ADDW    AX,     #2              ;テーブル・アドレスを+2をする
        CMPW    AX,     #TPWIDTH_END+2  ;デューティ・データ最終テーブル・アドレスと比較
        MOVW    HL,     AX              ;更新されたテーブル・アドレスをHLレジスタに格納
        BNZ     $HUPDATE                ;更新されたアドレス＜最終アドレスならば分岐
        MOVW    HL,     #TPWIDTH        ;更新されたアドレス＞最終アドレスならば
                                        ;テーブル・アドレス初期化
```

定義済み変数，HLレジスタの初期化

200 [us] ごとのINTTM000割り込みの割り込み要求フラグ（TMIF000）を利用し，250回ループを回すことで50 [ms] のチャタリング除去ウエイトを作成します。

本サンプル・プログラムでは，HLレジスタを割り込み処理内でグローバル変数（いつまでも残る変数）のように使用しています（テーブル・アドレス更新用）。

```
HUPDATE:
        MOV     A,      [HL]            ;デューティ・データ下位8ビットを読み出し
                                        ;(DWで定義したので8ビットずつ読み出す)
        XCH     A,      X               ;下位8ビット・データをXレジスタに格納
        MOV     A,      [HL+1]          ;デューティ・データ上位8ビットを読み出し
                                        ;(DWで定義したので8ビットずつ読み出す)
        MOVW    RNEXTD, AX              ;新しいデューティ・データをRNEXTDに保存
;-------------------------------------------------------------------------
;       現在のデューティと比較
;-------------------------------------------------------------------------
        MOVW    AX,     RCURRENTD       ;現在のデューティを読み出し
        ADDW    AX,     #COFFSET        ;オフセット時間を加算
        XCH     A,      X               ;下位8ビット・データと上位8ビット・データを交換
                                        ;(Aレジスタに下位データ格納)
        SUB     A,      [HL]            ;下位データ同士新しいデューティと比較
                                        ;(現在のデューティから新しいデューティ
                                        ;を減算し,結果をAレジスタに格納)
        XCH     A,      X               ;Aレジスタの内容と上位8ビットデータを交換
                                        ;(Aレジスタに上位データ格納)
        SUBC    A,      [HL+1]          ;上位データ同士新しいデューティと比較
                                        ;(現在のデューティから新しいデューティ
                                        ;とCYフラグを減算し,Aレジスタに格納)
;-------------------------------------------------------------------------
;       デューティを変更する割り込み処理を指定
;-------------------------------------------------------------------------
        BNC             $HDECDUTY       ;新しいデューティ＜現在のデューティ
                                        ;ならHDECDUTYに分岐(CY=0ならば分岐)
```

新デューティ＞現在のデューティの処理

```
        CLR1    TMIF000                 ;INTTM000割り込み要求フラグをクリア
        CLR1    TMMK000                 ;INTTM000割り込みマスクを解除
        SET1    TMMK010                 ;INTTM010割り込みをマスク
        BR      $HEND_INTP1             ;HEND_INTP1へ分岐
```

デューティを上げる処理

新デューティ≦現在のデューティの処理

```
HDECDUTY:
        CLR1    TMIF010                 ;INTTM010割り込み要求フラグをクリア
        CLR1    TMMK010                 ;INTTM010割り込みマスクを解除
        SET1    TMMK000                 ;INTTM000割り込みをマスク
HEND_INTP1:
        POP     AX                      ;AXレジスタのデータを復帰
        RETI                            ;割り込み処理から復帰
```

デューティを下げる処理

4. 7. 2　INTTM000割り込み処理

INTTM000割り込み処理では，次の動作を行います。

なお，INTTM010割り込み処理は処理内容がほぼ同じ（割り込みマスク・フラグの名称を変えるだけ）であるため，省略します。

【C言語】

- PPG出力の反転を禁止
- CR010の設定値（デューティ）の更新
- PPG出力の反転許可
- 新しいデューティを保存
- INTTM000割り込みをマスク

デューティを上げる割り込み処理

```
/************************************************************************
;            INTTM000割り込み処理
                この手法は，CR010のTM00動作中の特殊な書き換えかたです。
;************************************************************************/
__interrupt void       fn_inttm000(void){

            TOC00.4 = 0;                /* 出力反転禁止                 */
            CR010 = g_unNextD;          /* デューティ(パルス幅)変更     */

            NOP();                      /* タイマ00のカウント・クロック */
            NOP();                      /* 1周期分ウェイト              */

            TOC00.4 = 1;                /* 出力反転許可                 */

            g_unCurrentD = g_unNextD;   /* 新しいデューティを現在のデューティとして保存*/

            TMMK000 = 1;                /* INTTM000割り込みをマスク     */
}
```

【アセンブリ言語】

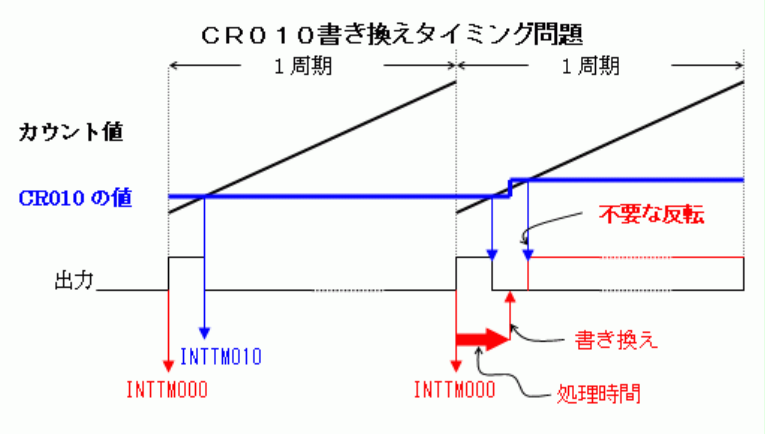
- PPG出力の反転を禁止
- CR010の設定値（デューティ）の更新
- PPG出力の反転許可
- 新しいデューティを保存
- INTTM000割り込みをマスク

デューティを上げる割り込み処理

```
;*********************************************************************
;       INTTM000割り込み処理
                この手法は，CR010のTM00動作中の特殊な書き換えかたです。
;*********************************************************************
IINTTM000:
        CLR1    TOC00.4                 ;出力反転禁止

        PUSH    AX                      ;AXレジスタのデータをスタックへ退避
        MOVW    AX,     RNEXTD
        MOVW    CR010,  AX              ;デューティ(パルス幅)変更

        NOP                             ;タイマ00のカウント・クロック
        NOP                             ;1周期分ウェイト

        SET1    TOC00.4                 ;出力反転許可

        MOVW    RCURRENTD, AX           ;新しいデューティを現在のデューティとして保存

        SET1    TMMK000                 ;INTTM000割り込みをマスク

        POP             AX              ;AXレジスタのデータを復帰
        RETI                            ;割り込み処理から復帰
```

【コラム】　C_OFFSET（C言語）およびCOFFSET（アセンブリ言語）について

INTTM000の割り込み要求で書き換える場合，必ず処理時間（割り込み要求を受け付けるまでの時間と割り込み処理での処理時間）が必要になります。その結果，実際にCR010が書き換えられるタイミングが，下図のように，CR010との一致タイミング（INTTM010）より遅くなることが考えられます。書き換えたとき，タイマ00が書き換えた値より小さいと，赤い波形のように2回目の一致が発生（デューティが逆転）します。



この問題に対応するため，INTTM010で処理に必要な時間分を加味して，INTTM000とINTTM010の使い分けを見直します（INTTM010での処理が完了する時間までの増加分までINTTM010で処理して，INTTM000での処理遅れをカバーすることを考えます）。さらに，処理時間に余裕を持たせるため，INTTM010での処理内容を見直します。具体的には，本サンプル・プログラムのようなINTTM010処理時間の補正値C_OFFSET（C言語）およびCOFFSET（アセンブリ言語）のようにINTTM010でCR010を書き換えるタイミングを数十クロックかかるようにします。

なお，本サンプル・プログラムにおいて，C_OFFSETは「46/4」に設定してありますが，以下の場合，適正な値に変更する必要があります。

- CPUクロックを変更した場合
- TM00のカウント・クロックを変更した場合

第5章　システム・シミュレータ SM+での動作確認

この章では，  のアイコンを選択してダウンロードしたC言語用のファイルを用い，サンプル・プログラムが，システム・シミュレータ SM+ for 78K0/Kx2でどのように動作するかを説明します。

5. 1　サンプル・プログラムのビルド

サンプル・プログラムをSM+ for 78K0/Kx2（以降，「SM+」と表記します）で動作確認をするために，サンプル・プログラムをビルドしてから，SM+を起動する必要があります。ここでは，  でダウンロードしたC言語用のファイルを用いて，統合開発環境　PM+にてビルドしてから，SM+を起動するまでの動作の一例を説明します。

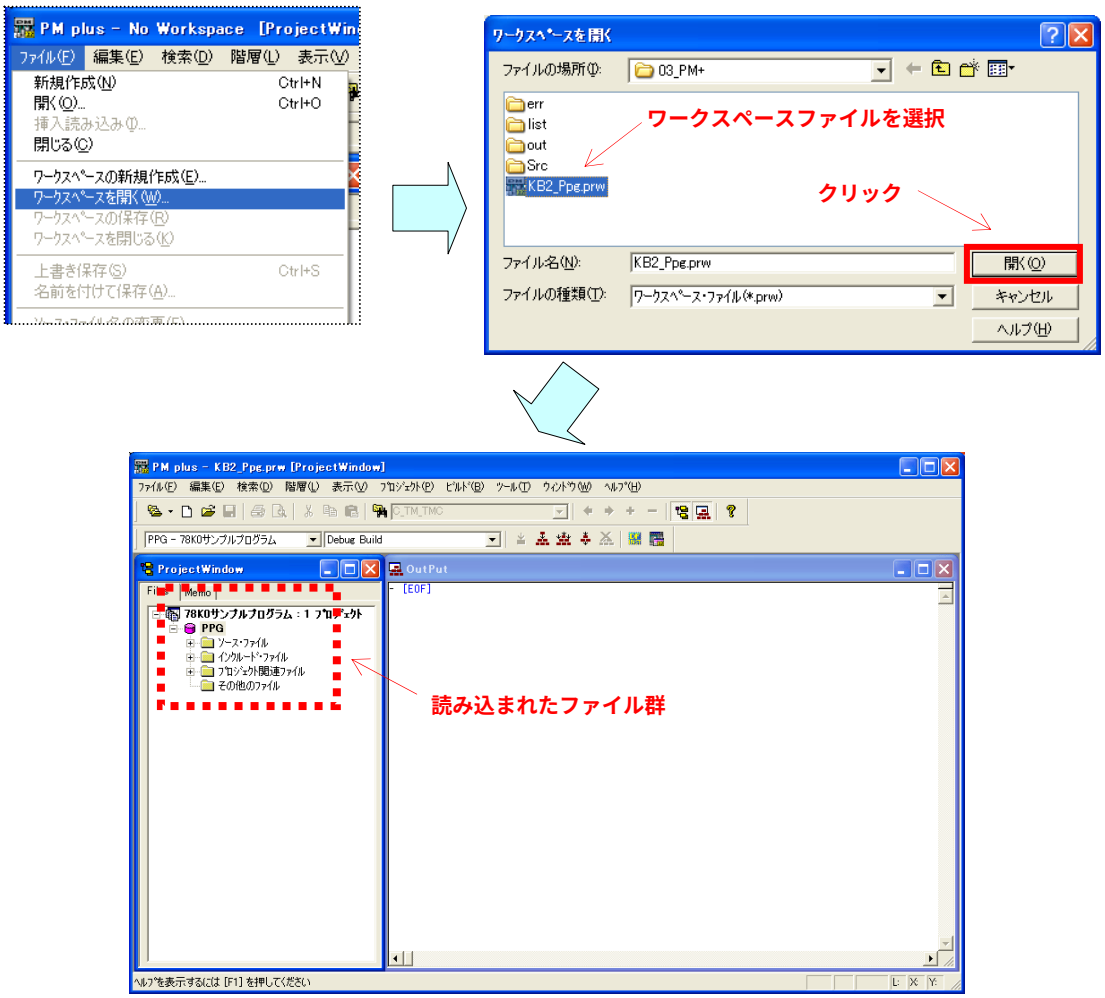
PM+操作方法の詳細については，PM+ プロジェクト・マネージャ　ユーザーズ・マニュアルを参照してください。

(1) PM+を起動してください。

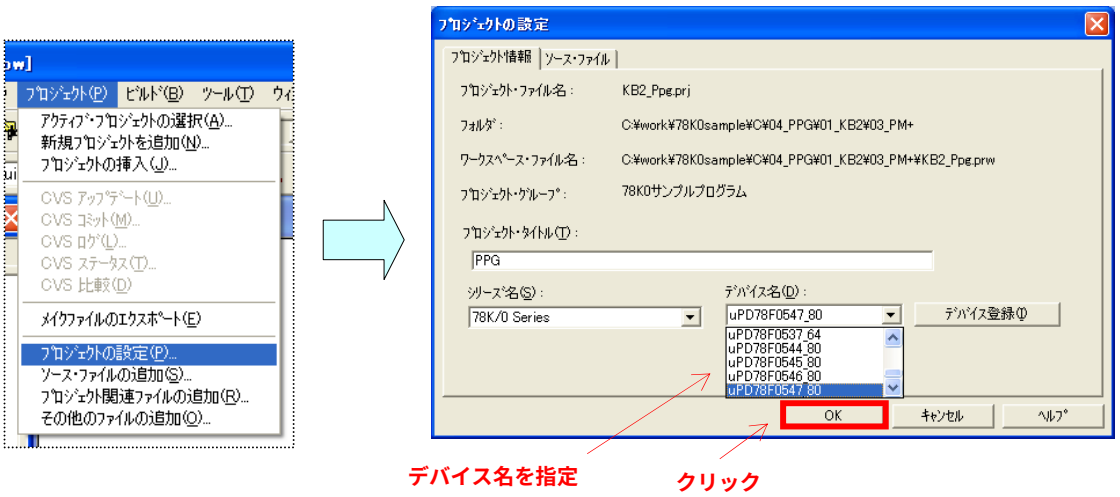
(2) [ファイル] → [ワークスペースを開く] から，「Kx2_Ppg.prw」[注]を選択し，[開く] ボタンをクリックしてください。ワークスペースが作成され，その中にソース・ファイルが自動的に読み込まれます。

注　ファイル名の"x"部分は対象デバイスにあわせて変更してください。

ex) 78K0/KB2の場合「KB2_Ppg.prw」

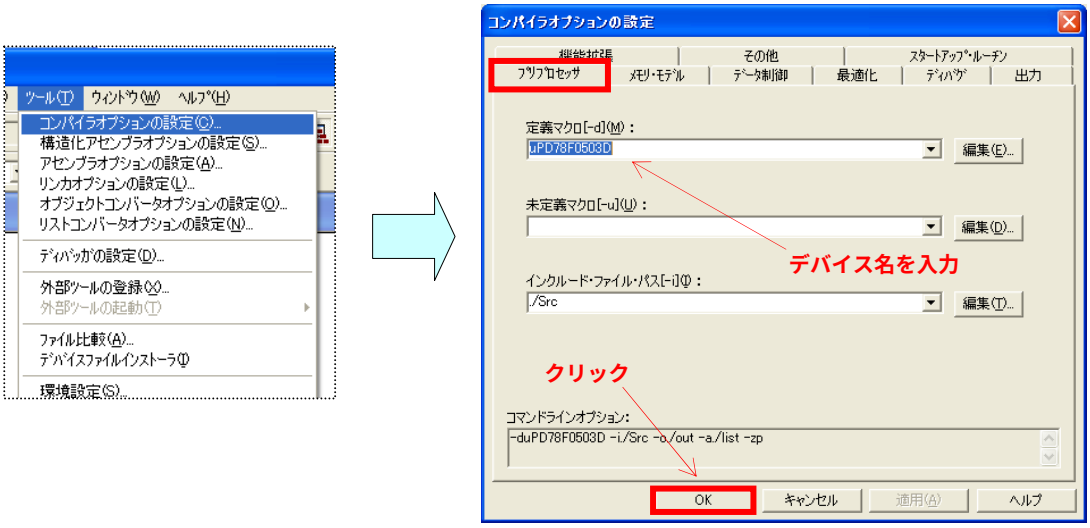


(3) [プロジェクト] → [プロジェクトの設定] を選択してください。[プロジェクトの設定] 画面が表示されたら，使用するデバイス名を選択（デフォルトでは，ROM/RAMサイズの最も大きいデバイスが選択）し，[OK] ボタンをクリックしてください。



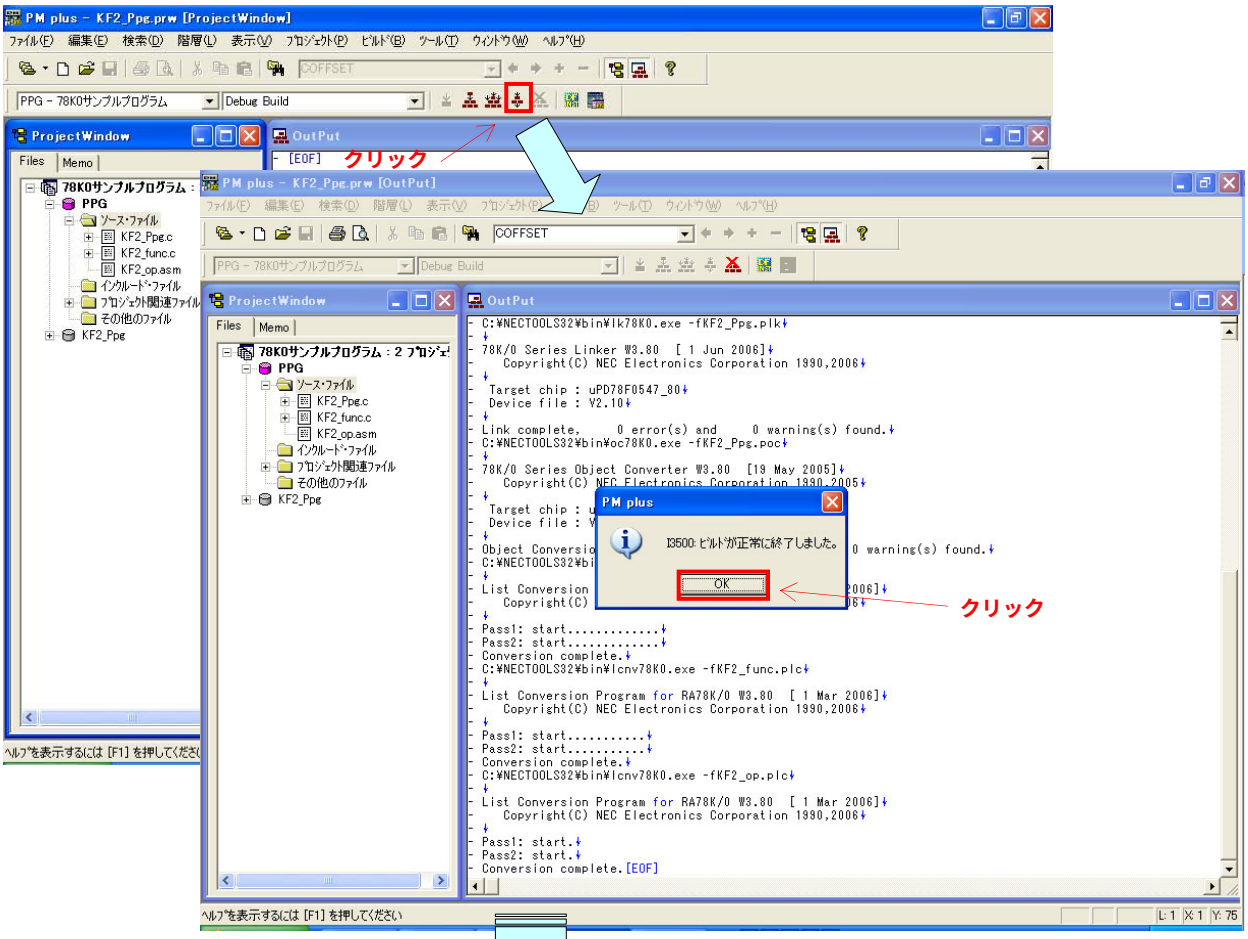
※μPD78F0500_36, μPD78F0501_36, μPD78F0502_36, μPD78F0503_36は選択しないでください。

(4) [ツール] → [コンパイラオプションの設定] を選択してください。[コンパイラオプションの設定] 画面が表示されたら，[プリプロセッサ] タグページが表示されているのを確認し，その中の定義マクロ欄に使用するデバイス名を入力し，[OK] をクリックします。入力するデバイス名は，「Kx2_Res.h」ヘッダファイルの先頭部のコメントを確認してください（以下は78K0/KB2の場合）。



(5)  （「ビルド→ディバグ」ボタン）をクリックしてください。ソース・ファイルの「Kx2_Ppg.c」と「Kx2_func.c」と「Kx2_op.asm」が正常にビルドされると，「I3500:ビルドが正常に終了しました」というメッセージ画面が表示されます。

(6) メッセージ画面にある［OK］ボタンをクリックすると， SM+が自動的に立ち上がります。

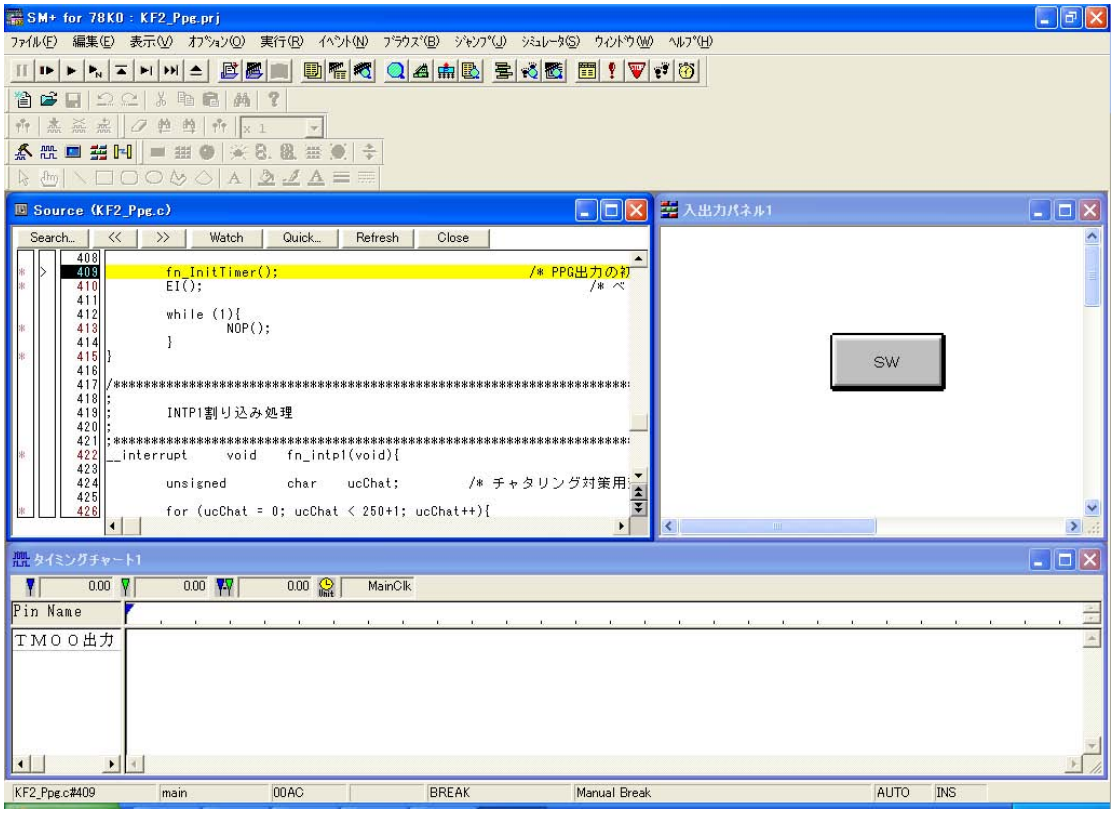


5. 2　SM+での動作

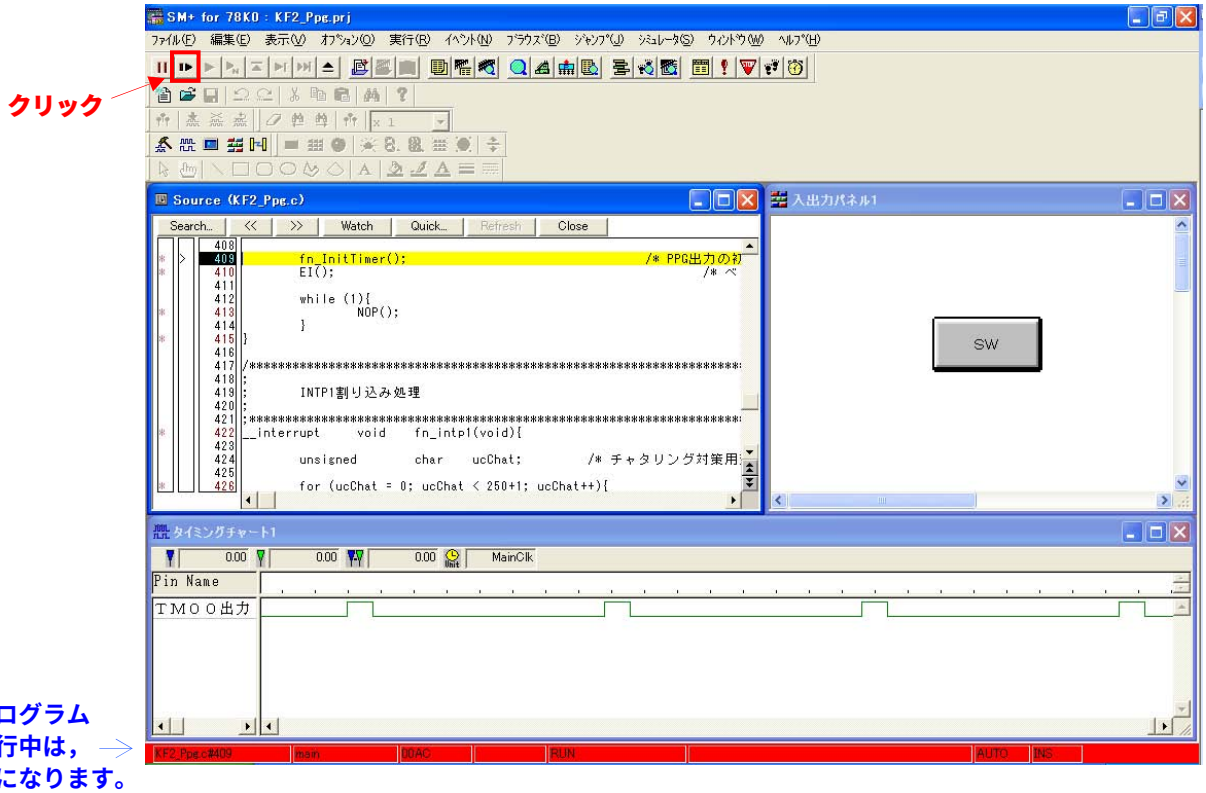
ここでは，SM+の入出力パネル・ウインドウやタイミング・チャート・ウインドウ上での動作確認の例を説明します。

SM+操作方法の詳細については，SM+ システム・シミュレータ　操作編　ユーザーズ・マニュアルを参照してください。

（1）PM+の「ビルド→ディバグ」からSM+を起動（5. 1を参照）すると，次のような画面になります。

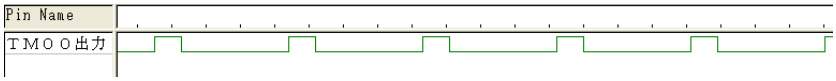
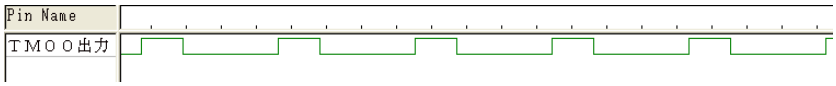
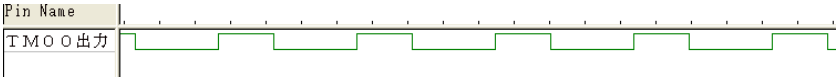
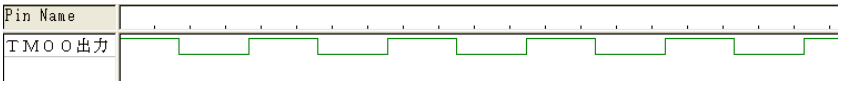
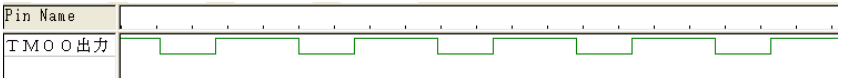
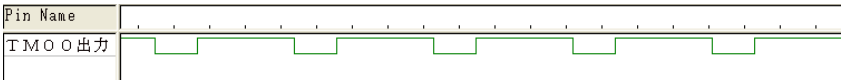
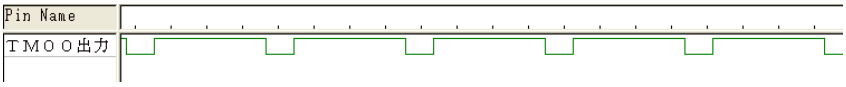
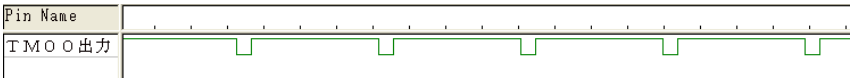
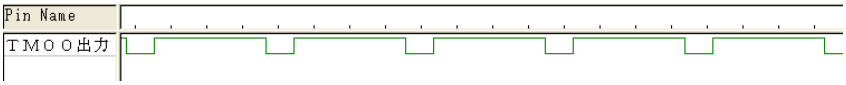
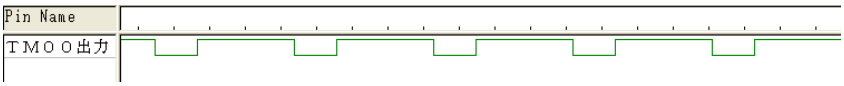
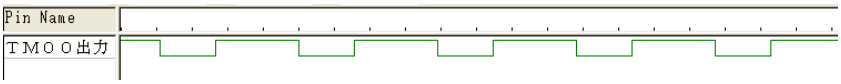
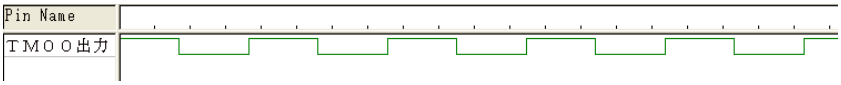
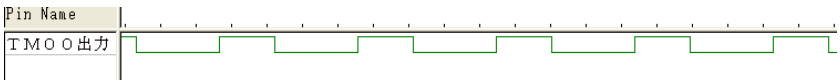
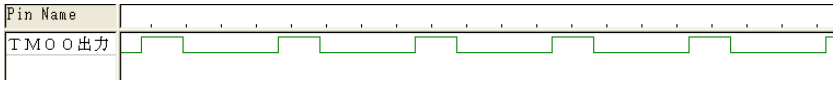
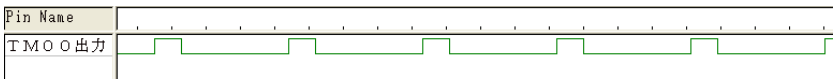


(2)  （「リスタート」ボタン）をクリックしてください。CPUリセット後，プログラムが実行され，次のような画面になります。



(3) プログラムを実行すると，タイミング・チャート上にP01の出力結果が表示されるのがわかります。
スイッチ入力ごとにデューティが10%ずつ変化していきます。

以下の表に示したタイミング・チャートのように，デューティが変化していきます。

スイッチ入力回数	デューティ比	タイミング・チャート・ウインドウ
0回	10%	
1回	20%	
2回	30%	
3回	40%	
4回	50%	
5回	60%	
6回	70%	
7回	80%	
8回	90%	
9回	80%	
10回	70%	
11回	60%	
12回	50%	
13回	40%	
14回	30%	
15回	20%	

※16回目以降は0回からの繰り返し

5. 3　オンチップ･デバッグ時の注意

ここでは，サンプル・プログラムを用いて，オンチップ･デバッグを行う際の手順を説明します。
オンチップ･デバッグ機能については，ユーザーズ・マニュアルを参照してください。

(1) オプション・バイトの設定

本サンプル・プログラムはオプション・バイトの初期設定でオンチップ･デバッグ禁止になっています。
オプション・バイトを設定し直して，オンチップ･デバッグを許可します。
オプション・バイト設定は Kx2_op.asm で行っています。
次に，そのKx2_op.asmファイル内のオンチップ･デバッグ設定部分のみ抜粋して記載します。

```
;                                   ↓           ★印が設定値
        DB      00000000B           ;0084H      :[オンチップデバッグ]
;               ||||||++---         OCDEN1-0    :[オンチップデバッグ動作制御]
;               ||||||              ★          00:動作禁止
;               ||||||                         01:設定禁止
;               ||||||                         10:動作許可（認証失敗でフラッシュ消去せず）
;               ||||||                         11:動作許可（認証失敗でフラッシュ消去）
;               ++++++----------     0          必ず0に設定
```

動作許可（認証失敗でフラッシュ消去せず）に設定

```
;                                   ↓           ★印が設定値
        DB      00000010B           ;0084H      :[オンチップデバッグ]
;               ||||||++---         OCDEN1-0    :[オンチップデバッグ動作制御]
;               ||||||                         00:動作禁止
;               ||||||                         01:設定禁止
;               ||||||              ★          10:動作許可（認証失敗でフラッシュ消去せず）
;               ||||||                         11:動作許可（認証失敗でフラッシュ消去）
;               ++++++----------     0          必ず0に設定
```

(2) オンチップ･デバッグ使用領域の確保（アセンブリ言語版のみ）

アセンブリ言語版はオンチップ･デバッグ使用領域を確保する必要があります。

本サンプル・プログラムでは，以下のようにオンチップ･デバッグ使用領域を確保しています。

```
;=============================================================================
;         ベクタテーブル
;
;  このサンプル・プログラムでは割り込みは使用していない。割り込み
;ベクタ・テーブルは全て不要割り込み処理アドレスに定義する。
;=============================================================================
TVECTTBL        CSEG    AT      0000H

        DW      IRESET          ;0000H RESET入力，POC，LVI，WDT
;       DW      IINIT           ;0002Hはオンチップデバッグ用に空ける

TVECT_TBL1      CSEG    AT      0004H
        DW      IINIT           ;0004H INTLVI
```

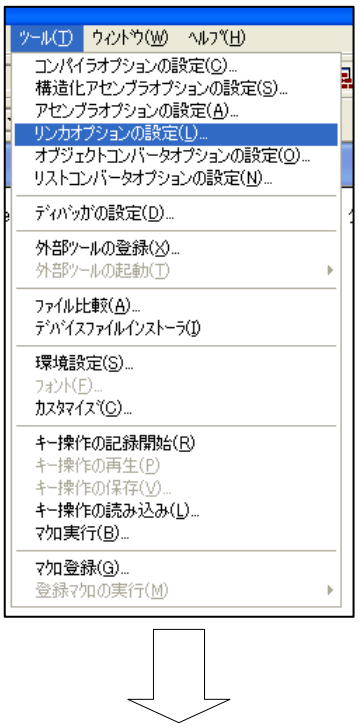
0002H番地はオンチップ・デバッグ使用領域として空けるため，0002H番地はコメント・アウトして，CSEGで再び0004H番地を定義しています。

※C言語版では不要です。

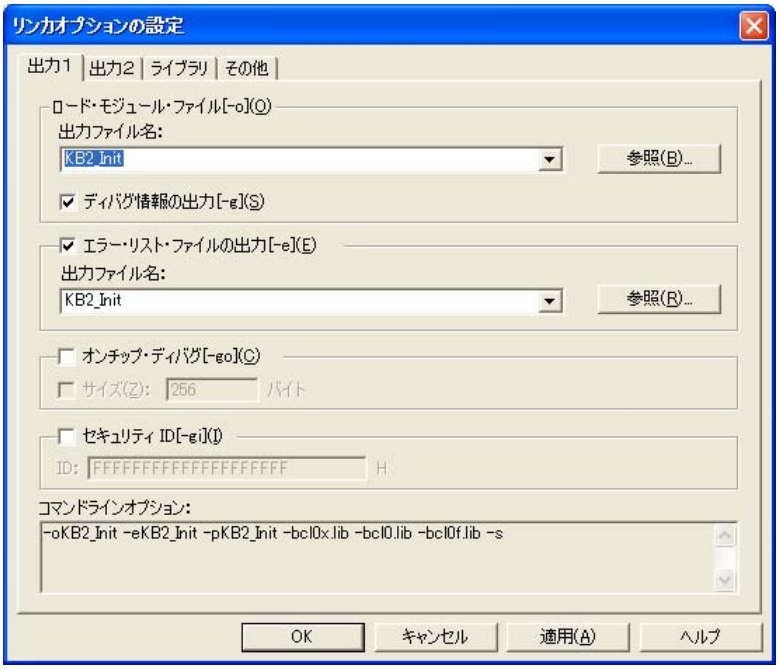
(3) リンカの設定

オンチップ･デバッグを行う場合，ビルドの際，リンカの設定を行う必要があります。

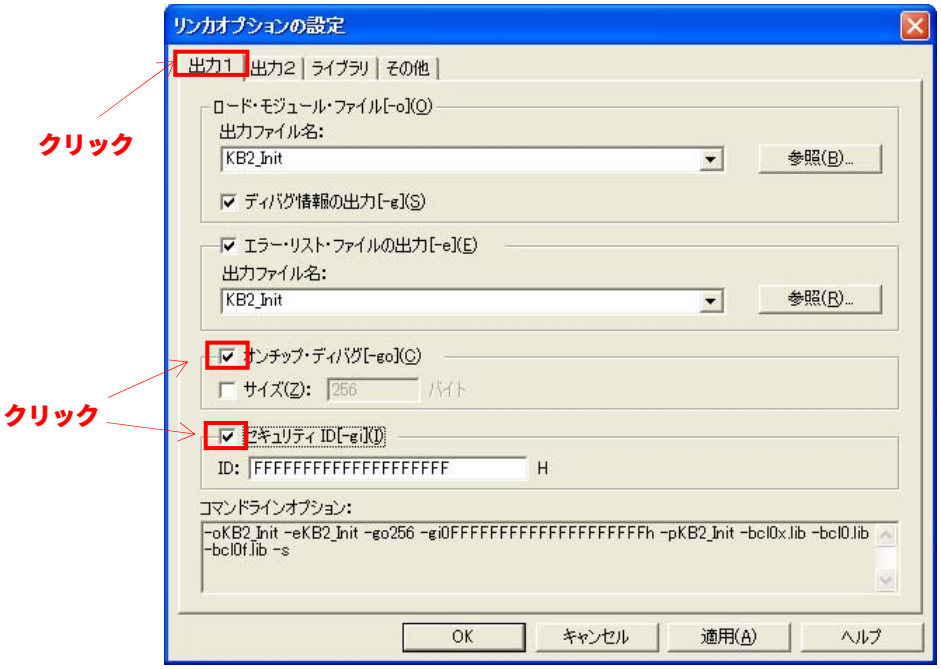
PM+の「ツール」メニューから「リンカオプションの設定」を選択してください。



「リンカオプションの設定」を選択するとリンカオプションの設定ダイアログが表示されます。



リンカオプションの設定ダイアログの「出力1」タブ上にある「オンチップ・ディバグ」と「セキュリティID」のチェックボックスをONしてください。



OKボタンを押下して設定完了です。

5. 4　開発環境のダウンロード，インストール

78K0/Kx2マイクロコントローラの開発ツールのフリーツールは，次のサイトより入手可能です。

→http://www.necel.com/micro/ja/freesoft/78k0/kx2/index.html

「SM+ for 78K0/Kx2」「RA78K0」「CC78K0」「78K0/Kx2用デバイス・ファイル」の4ファイルをダウンロードし，インストールすることで，サンプル・プログラムの動作確認が可能となります。

ダウンロード，インストールは，上記サイトの画面および説明に従って，行ってください。

備考1.　PM+は，RA78K0に同封されています。

2.　ダウンロード後，登録したEメール・アドレスに，RA78K0，CC78K0，SM+ for 78K0/Kx2のプロダクトIDが送付されます。このプロダクトIDは，各ツールのインストール時に必要となります。

第6章　関連資料

資　料　名		和文／英文
78K0/Kx2　ユーザーズ・マニュアル		PDF
78K/0シリーズ　命令編　ユーザーズ・マニュアル		PDF
RA78K0　アセンブラ・パッケージ	言語編	PDF
ユーザーズ・マニュアル	操作編	PDF
CC78K0　Cコンパイラ	言語編	PDF
ユーザーズ・マニュアル	操作編	PDF
PM+　プロジェクト・マネージャ　ユーザーズ・マニュアル		PDF
SM+　システム・シミュレータ　操作編　ユーザーズ・マニュアル		PDF

付録A　改版履歴

版　数	発行年月	改版箇所	改版内容
第1版	May 2009	－	－

【発　行】

NECエレクトロニクス株式会社

〒211-8668　神奈川県川崎市中原区下沼部1753

電話（代表）：(044)435−5111

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。

URL（アドレス）　http://www.necel.co.jp/

【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクスの販売特約店へお申し付けください。

お問い合わせ先

【営業関係，デバイスの技術関係お問い合わせ先】

半導体ホットライン

（電話：午前 9:00〜12:00，午後 1:00〜5:00）

電　話　：(044)435−9494

E-mail　：info@necel.com

【マイコン開発ツールの技術関係お問い合わせ先】

開発ツールサポートセンター

E-mail　：toolsupport-micom@ml.necel.com

C0901-A-01