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

パルス幅測定編

この資料は，サンプル・プログラムの「パルス幅測定」の動作概要と，マイコンの基本的なパルス幅測定の設定を説明したものです。サンプル・プログラムでは，パルス幅測定の設定を行ったあとに，入力されたパルス幅をキャプチャします。

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　U19036JJ1V0AN00（**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，入力される信号のパルス幅を測定します。測定方法は使用するタイマによって，以下の2通りの方法があります。

・タイマのキャプチャ機能を用いたハードウエアによる測定方法（16ビット・タイマ／イベント・カウンタ使用）

・インターバル・タイマのタイム・ベース割り込みを用いたソフトウエアによる測定方法（8ビット・タイマ使用）

本サンプル・プログラムでは，TM00，TM50，TMH0を用いて上記二つの方法の設定例を紹介しています。それぞれの概要について以下に示します。

【タイマのキャプチャ機能を用いたハードウエアによる測定方法】

この方法では，16ビット・タイマ／イベント・カウンタ00を使用しています。

本プログラムでは，TI000端子に入力された信号の立ち上がりと立ち下がりの両エッジを検出し，その時のタイマのカウント値をCR010レジスタにキャプチャします。キャプチャされた値からTI000端子に入力されたパルスの幅や間隔を知ることができます。

なお，本プログラムでは，TM00のカウント・クロックを$f_{PRS}/2^2$（0.5 [μ s] ←8 [MHz] 動作時）に設定します。しかし，この場合には約32 [ms] まで（OVFフラグ使用時は約65 [ms] ）しかカウントができません。そこで，本プログラムでは上位桁として8ビット変数を追加し，8.3 [s] までの計測を可能としました。測定結果は24ビットで得られますが，クロックの精度（5 %）以上の精度にはならないことに注意してください。

また，その8ビット変数はTM00のオーバフロー時にカウント・アップさせるため，CR000レジスタには0xFFFFを設定し，TM00のオーバフロー時に割り込みを発生させます。この際，INTTM000の割り込み優先順位がINTTM010よりも高くないと正しく動作しないので，INTTM000を高優先順位，INTTM010を低優先順位に設定します。

なお，本プログラムでの測定回数は8回としました。

【インターバル・タイマのタイム・ベース割り込みを用いたソフトウエアによる測定方法】

この方法では，8ビット・タイマ／イベント・カウンタ50および8ビット・タイマH0を使用しています。

本プログラムでは，外部割込みINTP0を用い，外部からのパルスの立ち上がり，立ち下がりエッジを検出し，ハイ・レベル期間のパルス幅を測定します。立ち上がりエッジ検出のINTP0割り込みにてタイマのカウント動作を開始し，0.1 [ms] ごとのコンペア一致割り込みを発生させ，その中で割り込みの回数をカウントします。そして，立ち下がりエッジ検出のINTP0割り込みにてタイマの動作を停止させます。得られたタイマの割り込み回数からパルス幅を計算します。時間測定の間隔が0.1 [ms] ごとなので，これ以上の測定精度は得られません。また，クロックの精度（5%）以上の精度にはならないことに注意してください。

また，本プログラムでは，タイマのカウント・クロックを$f_{PRS}/2^2$（0.5 [μ s] ← 8 [MHz] 動作時）に設定します。

なお，本プログラムでの測定回数は8回としました。

(1) パルス幅測定設定の内容

・タイマ関連の設定
・入力ポートの設定
・割り込みの設定

(2) メイン処理動作の内容

【タイマのキャプチャ機能を用いたハードウエアによる測定方法（TM00）】

・タイマ・カウント動作を開始して割り込みを許可し，最初のキャプチャを待って割り込み待機用ループに入ります。

【インターバル・タイマのタイム・ベース割り込みを用いたソフトウエアによる測定方法】

・変数を初期化して，割り込み待機用ループに入ります。

(3) 割り込み処理内容

【タイマのキャプチャ機能を用いたハードウエアによる測定方法（TM00）】

・INTTM000割り込み処理

キャプチャ割り込みと同時で，かつCR010（キャプチャ・データ）が0xFFFFならば上位桁をカウント・アップしません。それ以外の場合（INTTM000が単独で発生した場合や，キャプチャ割り込みと競合したがCR010（キャプチャ・データ）が0xFFFFでなかった場合）には上位桁をインクリメントします。

・INTTM010割り込み処理

キャプチャ・データを各データ格納テーブルに保存し，テーブル・アドレスを更新します。また，測定回数をカウントし，8回の測定が完了したら各割り込みをマスクします。なお，キャプチャ端子（TI000）の有効エッジは両エッジ検出に設定しているので，1回の測定ごとにパルスのハイ・レベル測定⇔ロウ・レベル測定と切り替わります。

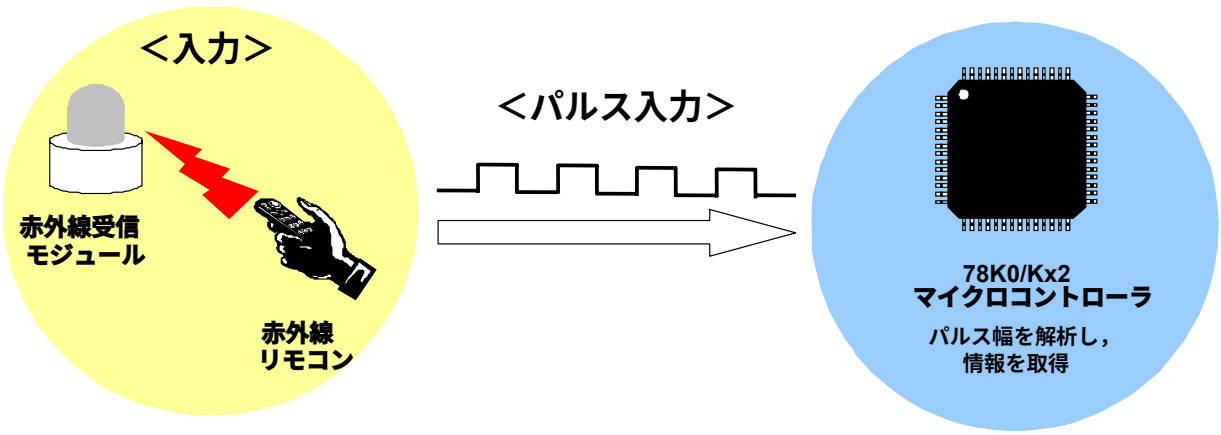
【インターバル・タイマのタイム・ベース割り込みを用いたソフトウエアによる測定方法】

・INTP0割り込み処理

パルスの立ち上がりで測定開始（タイマ動作開始）し，立ち下がりで測定終了（タイマ動作停止）します。

・INTTM50およびINTTMH0割り込み処理

測定値をインクリメントします。



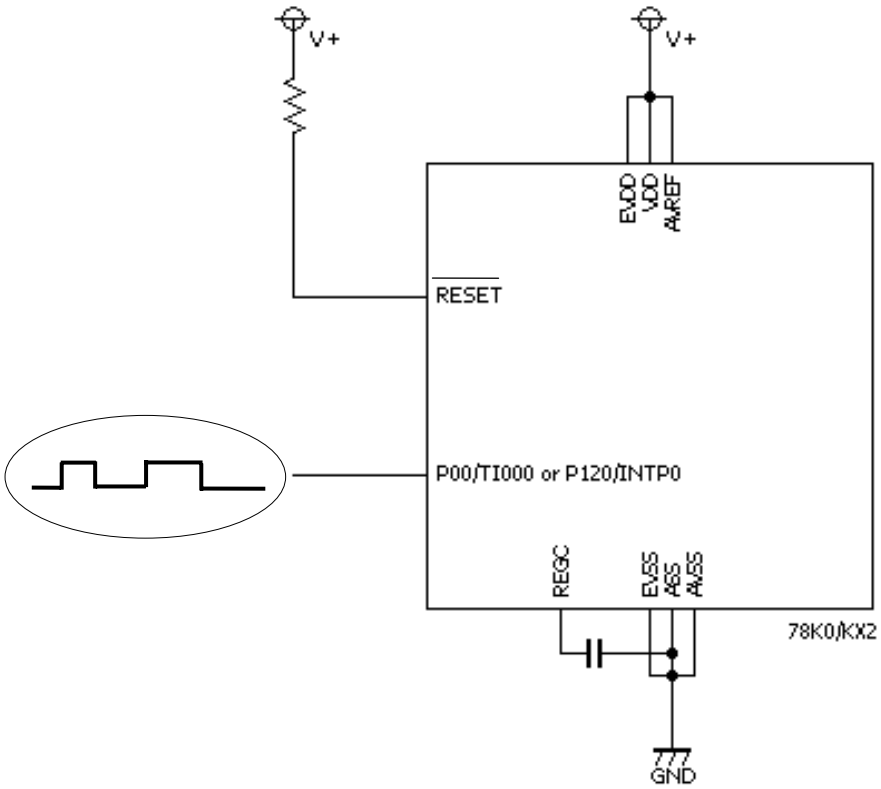
注意　デバイス使用上の注意事項については，78K0/Kx2　ユーザーズ・マニュアルを参照してください。

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。**

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・プログラムの方形波出力設定と動作概要，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_Pulse.c	パルス幅測定の設定，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_Pulse.prw	統合開発環境 PM+用ワーク･スペース・ファイル		●	
Kx2_Pulse.prj	統合開発環境 PM+用プロジェクト・ファイル		●	
Kx2_Pulse.pri Kx2_Pulse.prs Kx2_Pulse.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			●
Kx2_Pulse0.wvi	システム・シミュレータ SM+ for 78K0/Kx2用信号入力エディタ・ファイル（周辺ハードウエア動作を確認するために使用）			●

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_Pulse.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_Pulse.asm	パルス幅測定の設定，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・バイト設定用ソース・ファイル	●	●	
Kx2_Pulse.prw	統合開発環境 PM+用ワーク･スペース・ファイル		●	
Kx2_Pulse.prj	統合開発環境 PM+用プロジェクト・ファイル		●	
Kx2_Pulse.pri Kx2_Pulse.prs Kx2_Pulse.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			●
Kx2_Pulse0.wvi	システム・シミュレータ SM+ for 78K0/Kx2用信号入力エディタ・ファイル（周辺ハードウエア動作を確認するために使用）			●

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_Pulse.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（TM00）
・8ビット・タイマ／イベント・カウンタ50（TM50）
・8ビット・タイマH0（TMH0）

また，このサンプル・プログラムでは，上記3つのタイマについての設定例を紹介しています。
・入力ポート　　：P00/TI000, P120/INTP0
※TM01, TM51, TMH1のサンプル・プログラムは，上記TM00, TM50, TMH0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, TM50, TMH0との変更点をそれぞれ以下に記載します。

【TM00→TM01への変更点】
① C言語　　：割り込み処理関数宣言をINTTM000からINTTM001に，INTTM010からINTTM011に変更
　アセンブリ言語　　：割り込みベクタ・テーブルを0020H（INTTM000）から0038H（INTTM001）に，0022H（INTTM010）から003AH（INTTM011）に変更
② C，アセンブリ言語　：“パルス幅測定の初期化”でのTM00の各レジスタと割り込み関係フラグ名称の語尾を0から1に変更
③ C，アセンブリ言語　：“パルス幅測定の初期化”でP00とPM00（TI000）をP05とPM05（TI001）に変更
④ C，アセンブリ言語　：各割り込み処理内でのTM00の各レジスタと割り込み関係フラグ名称の語尾を0から1に変更

【TM50→TM51への変更点】
① C言語　　：割り込み処理関数宣言をINTTM50からINTTM51に変更
　アセンブリ言語　　：割り込みベクタ・テーブルを001EH（INTTM50）から002AH（INTTM51）に変更
② C，アセンブリ言語　：“パルス幅測定の初期化”でのTM50の各レジスタと割り込み関係フラグ名称の語尾を0から1に変更
③ C，アセンブリ言語　：各割り込み処理内でのTM50の各レジスタと割り込み関係フラグ名称の語尾を0から1に変更

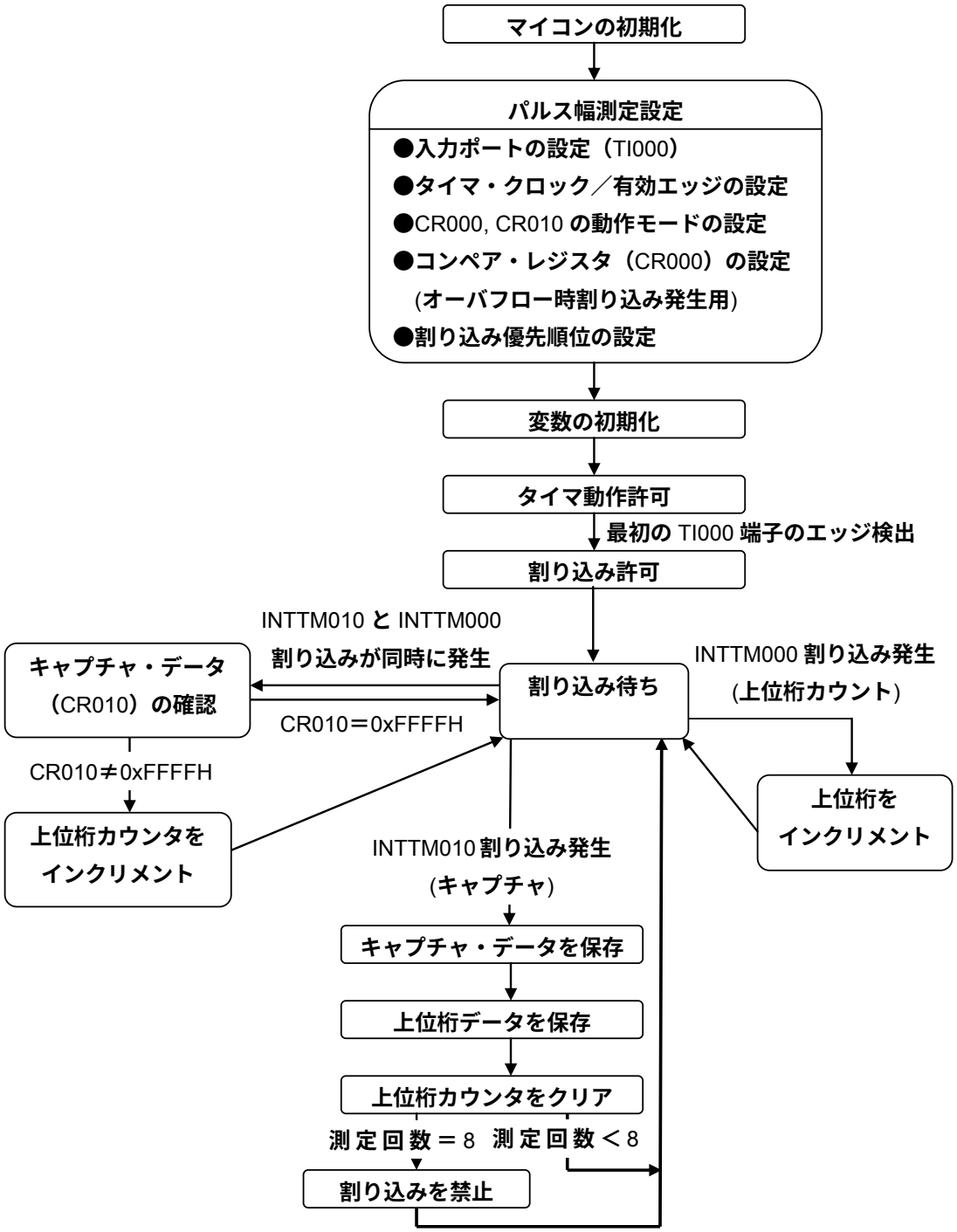
【TMH0→TMH1への変更点】
① C言語　　：割り込み処理関数宣言をINTTMH0からINTTMH1に変更
　アセンブリ言語　　：割り込みベクタ・テーブルを001CH（INTTMH0）から001AH（INTTMH1）に変更
② C，アセンブリ言語　：“パルス幅測定の初期化”でのTMH0の各レジスタと割り込み関係フラグ名称の語尾を0から1に変更
③ C，アセンブリ言語　：各割り込み処理内でのTMH0の各レジスタと割り込み関係フラグ名称の語尾を0から1に変更

注意　上記の変更を行うとカウント・クロックを選択するレジスタの内容にも変化が生じてしまうので，変更の際はカウント・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用途に合わせて設定してください。

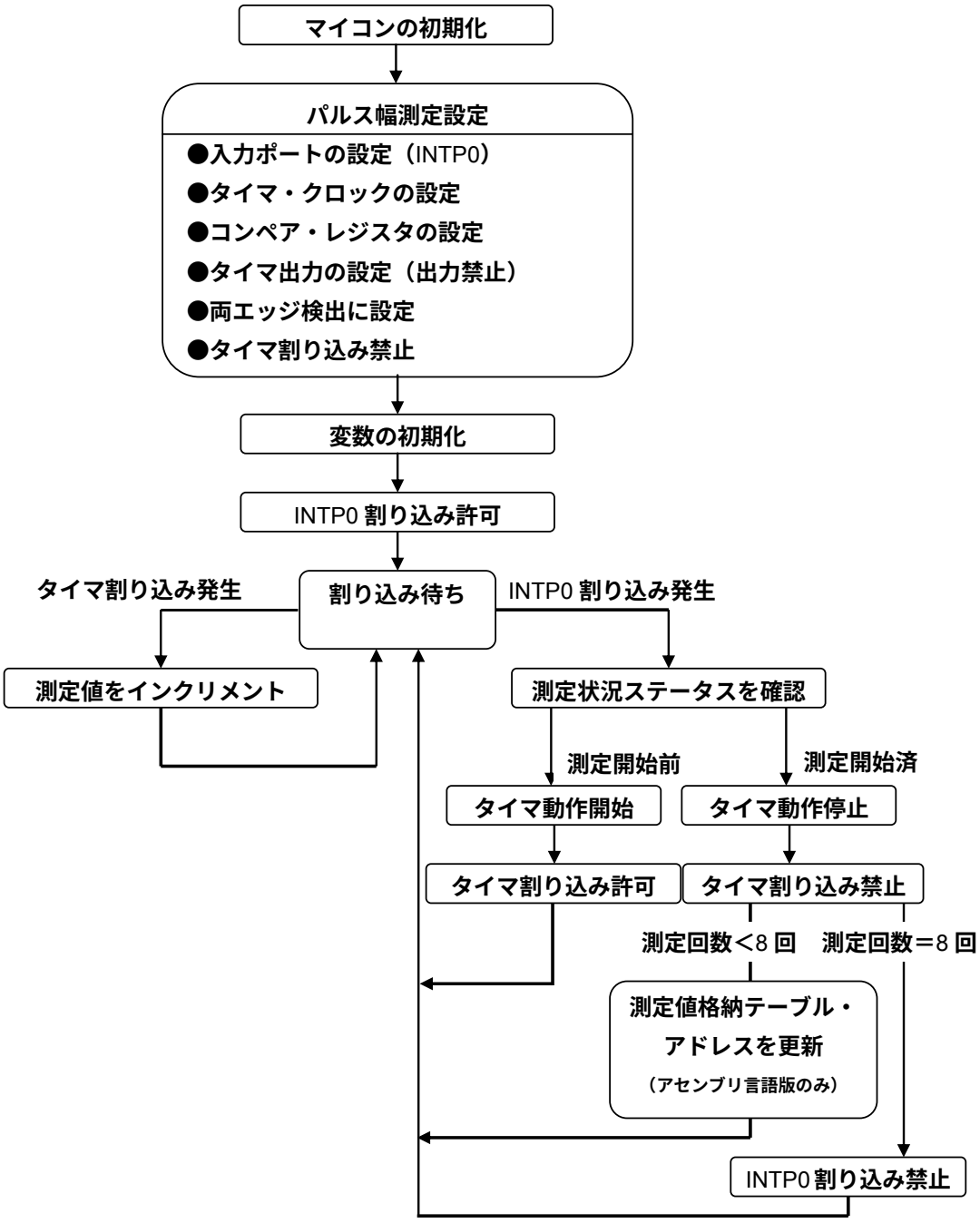
3. 3　パルス幅測定設定と動作概要

このサンプル・プログラムでは，パルス幅測定の設定を行います。
設定完了後は，入力ポートからの入力信号のパルス幅を測定します。
詳細については，次の状態遷移図（ステート・チャート）に示します。

【タイマのキャプチャ機能を用いたハードウエアによる測定方法】



【インターバル・タイマのタイム・ベース割り込みを用いたソフトウエアによる測定方法】

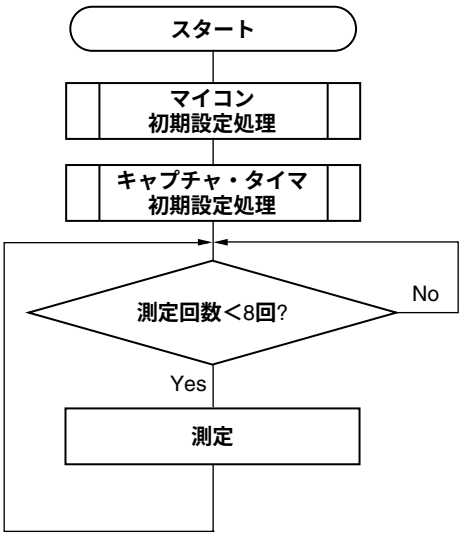


3. 4　フロー・チャート

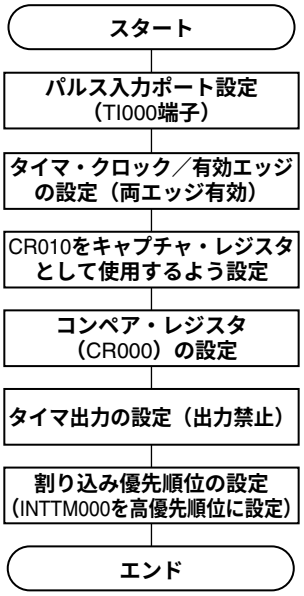
このサンプル・プログラムのフロー・チャートを次に示します。

【タイマのキャプチャ機能を用いたハードウエアによる測定方法】

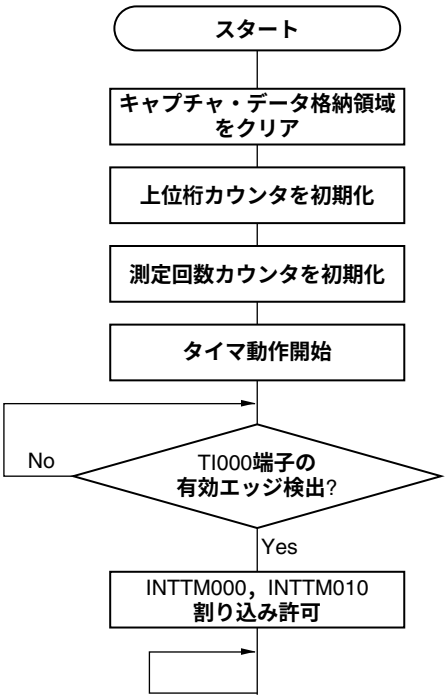
○ゼネラル・フロー（C言語版：Kx2_Pulse.c　アセンブリ言語版：Kx2_Pulse.asm）



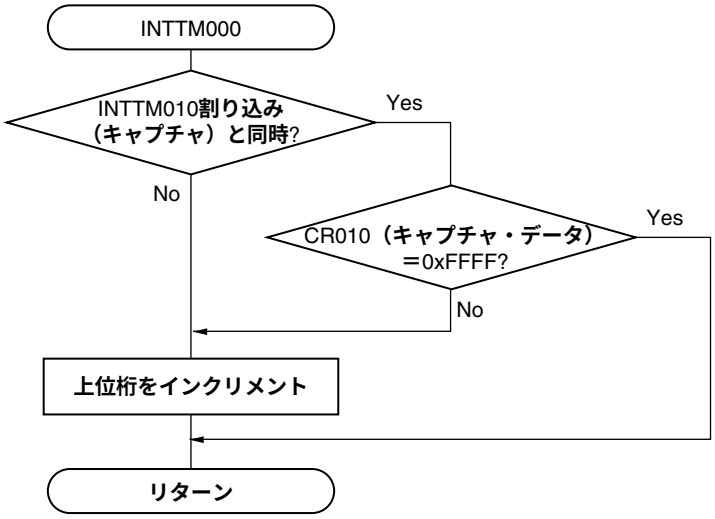
○キャプチャ・タイマ初期設定処理（C言語版：Kx2_Pulse.c　アセンブリ言語版：Kx2_Pulse.asm）



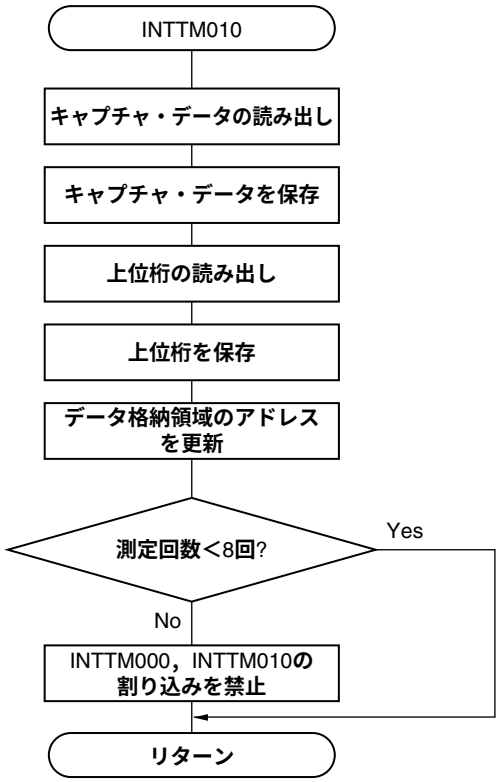
○メイン処理（C言語版：Kx2_Pulse.c　アセンブリ言語版：Kx2_Pulse.asm）



○INTTM000割り込み処理（上位桁カウント）（C言語版：Kx2_Pulse.c　アセンブリ言語版：Kx2_Pulse.asm）

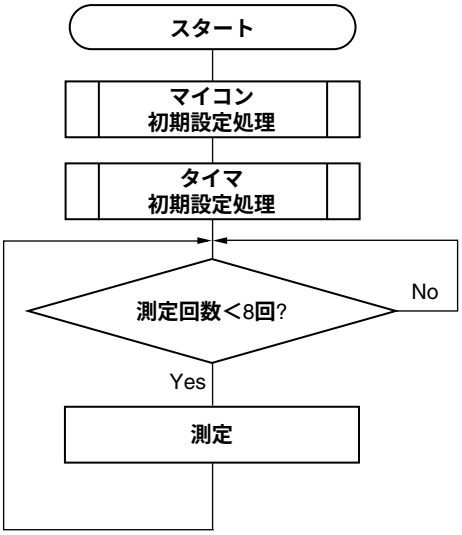


○INTTM010割り込み処理（キャプチャ割り込み）（C言語版：Kx2_Pulse.c　アセンブリ言語版：Kx2_Pulse.asm）

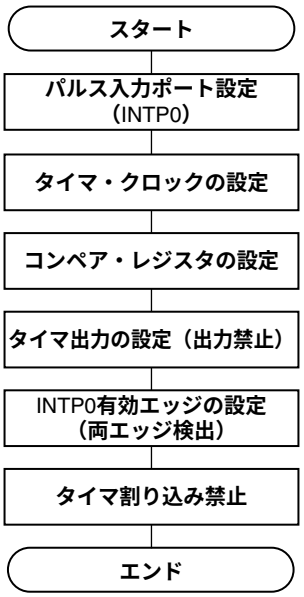


【インターバル・タイマのタイム・ベース割り込みを用いたソフトウエアによる測定方法】

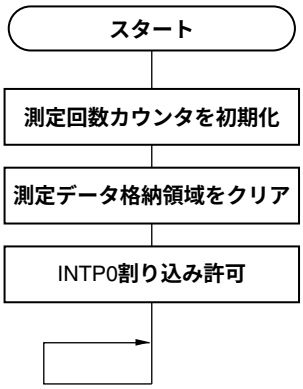
○ゼネラル・フロー　(C**言語版：**Kx2_Pulse.c　**アセンブリ言語版：**Kx2_Pulse.asm)



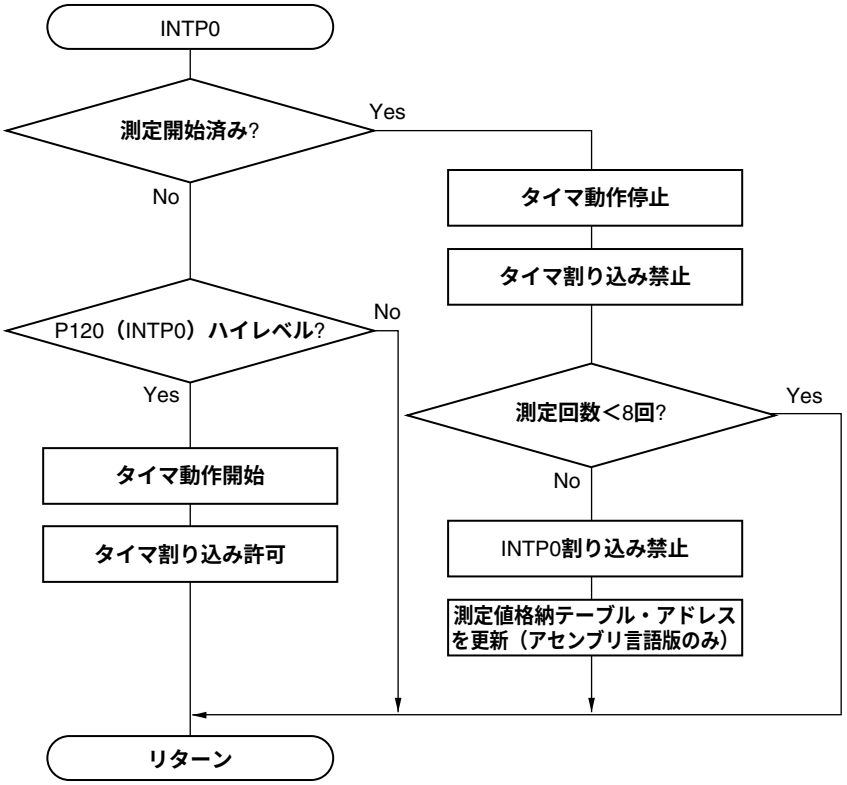
○パルス幅測定用タイマ初期設定処理　(C**言語版：**Kx2_Pulse.c　**アセンブリ言語版：**Kx2_Pulse.asm)



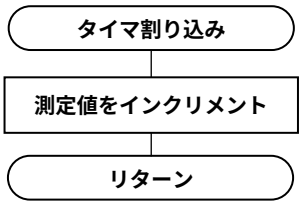
○メイン処理　(C言語版：Kx2_Pulse.c　アセンブリ言語版：Kx2_Pulse.asm)



○INTP0割り込み処理　(C言語版：Kx2_Pulse.c　アセンブリ言語版：Kx2_Pulse.asm)



○タイマ割り込み処理　(C言語版：Kx2_Pulse.c　アセンブリ言語版：Kx2_Pulse.asm)



第4章　設定方法について

この章では，前処理指令（C言語），パルス幅測定（16ビット・カウンタ／イベント・カウンタ00，8ビット・タイマ／イベント・カウンタ50，8ビット・タイマH0）の設定，およびメイン処理，割り込み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命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。ここでは，TM00版のみを例として説明します。

【C言語】 (Kx2_Pulse.c)

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

- #pragma sfr ← 特殊機能レジスタ（SFR）を記述可能にします。
- #pragma di ← DI命令を記述可能にします。
- #pragma ei ← EI命令を記述可能にします。
- #pragma nop ← NOP命令を記述可能にします。
- #pragma interrupt INTTM000 fn_inttm000 / #pragma interrupt INTTM010 fn_inttm010 ← 割り込み関数宣言をします。

4. 2　パルス幅測定タイマの設定

【タイマのキャプチャ機能を用いたハードウエアによる測定方法（TM00）】

パルス幅測定では，次の項目を設定します。

(1) タイマ・クロック選択

(2) CR000, CR010の動作モード設定

(3) タイマ・コンペア・レジスタの設定

(4) タイマ出力の設定

(5) タイマの動作許可，タイマ出力反転条件の設定

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) タイマ・クロック設定（プリスケーラ・モード・レジスタ00）

16ビット・タイマ／イベント・カウンタ00のカウント・クロックおよび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の設定

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端子の有効エッジの選択

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端子の有効エッジの選択

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

備考　f_{PRS} ： 周辺ハードウエア・クロック周波数

(2) キャプチャ／コンペア・コントロール・レジスタ00の設定

キャプチャ／コンペア・コントロール・レジスタ00の動作モードの設定をします。

図4－2　キャプチャ／コンペア・コントロール・レジスタ00の設定

CRC00

0	0	0	0	0	CRC002	CRC001	CRC000

CR000の動作モードの選択

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の動作モードの選択

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

(3) タイマ・コンペア・レジスタの設定

16ビット・タイマ／イベント・カウンタ00のカウンタ値と比較し，2つの値が一致したときに割り込みを発生します。

図4－3　16ビット・タイマ・キャプチャ／コンペア・レジスタ000

CR000

注意　CR000はタイマ停止中（TMC003, TMC002 = 00）に書き換えを行ってください。

(4) タイマ出力の設定

16ビット・タイマ／イベント・カウンタ00のカウンタのタイマ出力の設定をします。

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	OSPT0	OSPE0	TOC004	LVS00	LVR00	TOC001	TOE00
	0	0					

TO00出力制御

0	出力禁止（TO00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出力の制御

0	反転動作禁止
1	反転動作許可

TO00出力の状態の設定

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

CR010とTM00の一致によるTO00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　TOC00を設定するときは，必ず次の手順で設定してください。

① TOC004, TOC001のセット（1）

② TOE00だけを単独でセット（1）

③ LVS00またはLVR00のどちらか片方だけをセット（1）

(5) タイマの動作許可，タイマ出力の反転条件の設定

16ビット・カウンタ／イベント・カウンタ00のカウント・クロックの動作許可，タイマ出力反転条件の設定をします。

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

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
1	オーバフロー発生

タイマ出力（TO00）の反転条件

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00の動作許可

0	0	16ビット・タイマ／イベント・カウンタ00動作禁止。動作クロック供給停止。16ビット・タイマ・カウンタ（TM00）をクリア
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　16ビット・タイマ／イベント・カウンタ00は，TMC003, TMC002に00（動作停止モード）以外の値を設定した時点で動作を開始します。動作を停止させるには，TMC003, TMC002に00を設定してください。

【例】　・タイマ・クロックを0.5 [μs] に設定（8 MHz動作時）
・CR000をコンペア・レジスタ，CR010をキャプチャ・レジスタに設定
・CR000をオーバフロー時の割り込み発生用に0xFFFFを設定
（サンプル・プログラムの設定と同内容）

PRM00

0	0	1	1	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

1	1	両エッジ

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	1	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
- 1：キャプチャ・レジスタとして動作

CR000

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

※本サンプル・プログラムでは，上位桁8ビットをカウントさせるためにタイマ・カウンタがオーバフローをしたときに割り込みを発生させます。そのため，このCR000レジスタには，0xFFFFを設定します。

TOC00

0	0	0	0	0	0	0	0

TO00出力制御
- 0：出力禁止（TO00出力は，ロウ・レベルに固定）

CR000とTM00の一致によるTO00の出力の制御
- 0：反転動作禁止

TO00の出力状態の設定
- 0 0：変化しない

CR010とTM00の一致によるTO00出力の制御
- 0：反転動作禁止

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	0	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 0：TI000端子の有効エッジ入力でクリア&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_Pulse.c)

```
PRM00   =   0b00110001;   /* タイマ・クロックをfPRS/2^2に，TI000端子の
                             有効エッジを両エッジに設定                    */

CRC00   =   0b00000100;   /* CR000をコンペア・レジスタに，CR010をキャプチャ・
                             レジスタに設定                                */

CR000   =   0xFFFF;       /* コンペア・レジスタの設定
                             （オーバフロー時割り込み用）                  */

TOC00   =   0b00000000;   /* タイマ出力はしない                            */
                          •
                          •
                          •
TMC00   =   0b00001000;   /* タイマの動作許可                              */
                          •
                          •
                          •
```

タイマの動作許可は，メイン処理内で設定します。

【アセンブリ言語】　(Kx2_Pulse.asm)

```
MOV   PRM00,   #00110001B    ;タイマ・クロックをfPRS/2^2に，TI000端子の
                             ;有効エッジを両エッジに設定

MOV   CRC00,   #00000100B    ;CR000をコンペア・レジスタに，CR010をキャプチャ・
                             ;レジスタに設定
MOVW  CR000,   #0FFFFH       ;コンペア・レジスタの設定
                             ;（オーバフロー時割り込み用）

MOV   TOC00,   #00000000B    ;タイマ出力はしない
                             •
                             •
                             •
MOV   TMC00,   #00001000B    ;タイマの動作許可
                             •
                             •
                             •
```

タイマの動作許可は，メイン処理内で設定します。

【インターバル・タイマのタイム・ベース割り込みを用いたソフトウエアによる測定方法（TM50）】

パルス幅測定では，次の項目を設定します。

(1) タイマ・クロック選択

(2) タイマ・コンペア・レジスタ（CR50）の設定

(3) タイマ・モードの設定

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) タイマ・クロック設定

8ビット・タイマ／イベント・カウンタ50のカウント・クロックおよび端子入力の有効エッジを設定します。

図4－6　タイマ・クロック選択レジスタ50

TCL50

0	0	0	0	0	TCL50 2	TCL50 1	TCL50 0

カウント・クロックの選択

0	0	0	TI50端子の立ち下がりエッジ
0	0	1	TI50端子の立ち上がりエッジ
0	1	0	f_{PRS}
0	1	1	$f_{PRS}/2$
1	0	0	$f_{PRS}/2^2$
1	0	1	$f_{PRS}/2^6$
1	1	0	$f_{PRS}/2^8$
1	1	1	$f_{PRS}/2^{13}$

注意1. TCL50を同一データに意外に書き換える場合は，いったんタイマ動作を停止させてから書き換えてください。

2. ビット3-7には，必ず0を設定してください。

備考　f_{PRS} ： 周辺ハードウエア・クロック周波数

(2) タイマ・モード設定

8ビット・タイマ・カウンタの動作モード，カウンタ制御，出力の状態／制御の設定をします。

図4－7　8ビット・タイマ・モード・コントロール・レジスタ50

TMC50

TCE50	TMC506	0	0	LVS50	LVR50	TCM501	TOE50

タイマ出力の制御

0	出力禁止
1	出力許可

タイマF/Fの制御（PWMモード以外）

0	反転動作禁止
1	反転動作許可

タイマF/F出力の状態設定

0	0	変化しない
0	1	タイマ出力F/Fをクリア(0)
1	0	タイマ出力F/Fをセット(1)
1	1	設定禁止

TM50の動作モード選択

0	TM50とCR50の一致でクリア&スタート・モード
1	PWM（フリー・ランニング）モード

TM50のカウント動作制御

0	カウントを0にクリア後，カウント動作禁止（カウンタ停止）
1	カウント動作開始

注意1. LVS50とLVR50の設定は，PWMモード以外で有効になります。

2. 以下の設定は同時に行わないでください。また，設定は以下の順で行ってください。
   - ① TMC501, TMC506を設定
   - ② 出力を許可する場合，TOE50を設定
   - ③ LVS50, LVR50を設定
   - ④ TCE50を設定
3. TCE50=1のとき，TMC50の他のビットを設定することは禁止です。

備考1. LVS50とLVR50は読み出すと，0になっています。

2. TMC506, LVS50, LVR50, TMC501, TOE50の各ビットの値は，TCE50の値に関係なくTO50端子に反映されます。

(3) 8ビット・タイマ・コンペア・レジスタ（CR50）の設定

8ビット・タイマ50のカウンタ値と比較し，2つの値が一致したときに割り込みを発生します。

図4－8　8ビット・タイマ・コンペア・レジスタ50

CR50

注意1.　CR50の値は，00H-FFHの範囲で設定します。

2.　TM50と値の一致でクリア&スタート・モードのときは，動作中にCR50に異なる値を書き込まないでください。

【例】　・タイマ・クロックを0.5 [μs] に設定（8 MHz動作時）

・0.1 [ms] ごとに割り込みを発生

（サンプル・プログラムの設定と同内容）

TCL50

0	0	0	0	0	1	0	0

カウント・クロックの選択

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

CR50

1	1	0	0	0	1	1	1

200（CR50設定値＋1）×0.5 [μs] = 0.1 [ms]

※ここでは，199（0.1 [ms] ）に設定

TMC50

0⇔1	0	0	0	0	0	0	0

タイマ出力の制御

0	出力禁止

タイマF/Fの制御（PWMモード以外）

0	反転動作禁止

タイマF/F出力の状態設定

0	0	変化しない

TM50の動作モード選択

0	TM50とCR50の一致でクリア&スタート・モード

TM50のカウント動作制御

0⇔1	カウントを0にクリア後，カウント動作禁止（カウンタ停止）　⇔　動作許可

【C言語】　(Kx2_Pulse.c)

```
TCL50    =        0b00000100;     /* タイマ・クロックをfPRS/2^2に設定 */

CR50     =        200-1;          /* コンペア・レジスタの設定          */

TMC50    =        0b00000000;     /* タイマ出力はしない                */
                                  •
                                  •
                                  •
                                  •
TCE50    =        1⇔0;            /* タイマの動作開始 ⇔ 禁止           */
                                  •
                                  •
                                  •
```

タイマの動作の開始／停止は，INTP0割り込み内で設定します。
詳細は，4.7　**割り込み処理**で説明します。

【アセンブリ言語】　(Kx2_Pulse.asm)

```
MOV      TCL50,   #00000100B      ;タイマ・クロックをfPRS/2^6に設定

MOV      CR50,    #200-1          ;コンペア・レジスタの設定

MOV      TMC50,   #00000000B      ;タイマ出力はしない
                                  •
                                  •
                                  •
                                  •
SET1  TCE50 ⇔ CLR1 TCE50         ;タイマの動作開始 ⇔ 禁止
                                  •
                                  •
                                  •
```

タイマの動作の開始／停止は，INTP0割り込み内で設定します。
詳細は，4.7　**割り込み処理**で説明します。

【インターバル・タイマのタイム・ベース割り込みを用いたソフトウエアによる測定方法（TMH0）】

パルス幅測定では，次の項目を設定します。

(1) タイマ・クロック選択，タイマ・モード設定，タイマ出力制御

(2) タイマ・コンペア・レジスタ（CMP00）の設定

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) タイマ・クロック設定

8ビット・タイマH0のカウント・クロック，タイマ・モード，タイマ出力の設定をします。

図4－9　8ビット・タイマHモード・レジスタ0

TMHMD0

TMHE0	CKS02	CKS01	CKS00	TMMD01	TMMD00	TOLEV0	TOEN0

タイマ出力制御

0	出力禁止
1	出力許可

タイマ出力レベル制御（デフォルト時）

0	ロウ・レベル
1	ハイ・レベル

タイマ動作モード

0	0	インターバル・タイマ・モード
1	0	PWM出力動作モード
上記以外		設定禁止

カウント・クロックの選択

0	0	0	f_{PRS}
0	0	1	$f_{PRS}/2$
0	1	0	$f_{PRS}/2^2$
0	1	1	$f_{PRS}/2^6$
1	0	0	$f_{PRS}/2^{10}$
1	0	1	TM50の出力
上記以外			設定禁止

タイマ動作許可

0	タイマ・カウント動作停止（カウンタは0にクリア）
1	タイマ・カウント動作許可（クロックを入力することでカウント動作開始）

注意　TMHE1 = 1のとき，TMHMD1の他のビットを設定することは禁止です。
ただし，リフレッシュ（同値書き込み）することは可能です。

(2) 8ビット・タイマ・コンペア・レジスタ（CMP00）の設定

8ビット・タイマH0のカウンタ値と比較し，2つの値が一致したときに割り込みを発生します。

図4－10　8ビット・タイマHコンペア・レジスタ00

CMP00

注意　CMP00は，タイマ・カウント動作中に値を書き換えないでください。
ただし，タイマ・カウント動作中にリフレッシュ（同値書き込み）することは可能です。

【例】　・タイマ・クロックを0.5 [μs] に設定（8 MHz動作時）
・0.1 [ms] ごとに割り込みを発生
（サンプル・プログラムの設定と同内容）

TMHMD0

0⇔1	0	1	0	0	0	0	0

タイマ出力制御

0	出力禁止

タイマ出力レベル制御

0	ロウ・レベル

タイマ動作モード

0	0	インターバル・タイマ・モード

カウント・クロックの選択

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

タイマ動作許可

0⇔1	タイマ・カウント動作停止 ⇔ 動作許可

CMP00

1	1	0	0	0	1	1	1

200（CMP00設定値＋1）×0.5 [μs] = 0.1 [ms]
※ここでは，199に設定

【C言語】 (Kx2_Pulse.c)

```
TMHMD0   =   0b00100000;     /* タイマ・クロックをfPRS/2^2に設定
                                タイマ・モードをインターバル・タイマに設定
                                タイマ出力はしない                          */

CMP00    =   200-1;          /* コンペア・レジスタの設定                    */
                             •
                             •
                             •
                             •
TMHE0    =   1⇔0;            /* タイマの動作開始 ⇔ 禁止                     */
                             •
                             •
                             •
```

タイマの動作の開始／停止は，INTP0割り込み内で設定します。
詳細は，4.7　**割り込み処理**で説明します。

【アセンブリ言語】 (Kx2_Pulse.asm)

```
MOV    TMHMD0,  #00110000B   ;タイマ・クロックをfPRS/2^6に設定
                             ;タイマ・モードをインターバル・タイマに設定
                             ;タイマ出力はしない
MOV    CMP00,   #200-1       ;コンペア・レジスタの設定
                             •
                             •
                             •
                             •
SET1   TMHE0 ⇔ CLR1  TMHE0   ;タイマの動作許可 ⇔ 禁止
                             •
                             •
                             •
```

タイマの動作の開始／停止は，INTP0割り込み内で設定します。
詳細は，4.7　**割り込み処理**で説明します。

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4－11　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4－12　割り込み要求フラグ・レジスタ（IF0H）のフォーマット

IF0H

TMIF010	TMIF000	TMIF50	TMIFH0	TMIFH1	DUALIF0	STIF6	SRIF6

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（【例1】，【例2】，【例3】）。

図4－13　割り込みマスク・フラグ・レジスタ（MK0L）のフォーマット

MK0L

SREMK6	PMK5	PMK4	PMK3	PMK2	PMK1	PMK0	LVIMK

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4－14　割り込みマスク・フラグ・レジスタ（MK0H）のフォーマット

MK0H

TMMK010	TMMK000	TMMK50	TMMKH0	TMMKH1	DUALMK0	STMK6	SRMK6

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4－15　外部割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4－16　外部割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）ポートの入力の設定

このサンプル・プログラムではパルス入力ポートとしてP00/TI000【パルス・キャプチャ・レジスタを使用した方法】，P120/INTP0【タイマ・カウントを使用した方法】を使用します。

ここでは，どちらの方法もパルス入力として使用するので入力モードとして使用します。

図4－17　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4－18　ポート・モード・レジスタ12（PM12）のフォーマット

PM12

PM127	PM126	PM125	PM124	PM123	PM122	PM121	PM120

P12n（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

4.5　メイン処理

【タイマのキャプチャ機能を用いたハードウエアによる測定方法】

メイン処理では次の動作を行います。

【C言語】　(Kx2_Pulse.c)

・パルス幅測定設定関数を呼び出します（タイマ処理を開始します）。

・キャプチャ・データ格納領域をクリアし，上位桁カウンタ，測定回数カウンタを初期化します。

・タイマの動作を開始して最初のキャプチャを待ち，割り込みを許可して割り込み待機ループに入ります。

```
/************************************************************************
;       メイン処理
;************************************************************************/
void   main(void){
       fn_InitTimer();              /*パルス幅測定の初期化              */

       for(g_ucTimes = 0; g_ucTimes < 8; g_ucTimes++){
                                    /*上位･下位結果格納領域を初期化する*/
              g_unLowLength[g_ucTimes]  =    0;
                                    /*結果格納領域のクリア              */
              g_ucHighLength[g_ucTimes] =    0;
                                    /*結果格納領域のクリア              */
       }
       g_ucHighCount    =     0;    /*上位桁カウンタを初期化            */
       g_ucTimes        =     8;    /*測定回数を初期化                  */

       TMC00    =        0b0000100; /*タイマ動作開始                    */
       while(TMIF010==0){           /*最初のキャプチャ完了待ち          */
              NOP();
       }
       TMIF000 =        0;          /*INTTM000割り込み要求クリア        */
       TMMK000 =        0;          /*INTTM000割り込みマスク解除        */
       TMIF010 =        0;          /*INTTM010割り込み要求クリア        */
       TMMK010 =        0;          /*INTTM010割り込みフラグ解除        */
       EI();                        /*割り込み許可                      */
       while(1){                    /*割り込み待機無限ループ            */
              NOP();
       }
}
```

パルス幅測定の設定関数を呼び出します。

上位，下位の8回測定分の領域をクリアします。

g_unLowLength[0]～[7]をクリアします。

g_unHighLength[0]～[7]をクリアします。

結果格納領域の初期化ですでに8回に初期化はされていますが，念のためここでも再び8回に初期化します。

最初のTMIF010の立ち上がり（キャプチャ割り込み）を待ち，TMIF010が立ち上がったら割り込みを許可します。

【アセンブリ言語】　(Kx2_Pulse.asm)

・キャプチャ・データ格納領域をクリアし，上位桁カウンタ，測定回数カウンタを初期化します。

・タイマの動作を開始して最初のキャプチャを待ち，割り込みを許可して割り込み待機ループに入ります。

```
;*************************************************************************;
;       メイン処理
;*************************************************************************
        SEL     RB1                     ;レジスタバンクを1に切り替える

        MOVW    HL,     #RLOWLENGTH-1   ;下位16ビット格納用テーブルアドレスを
                                        ;HLレジスタに格納
        MOVW    DE,     #RHIGHLENGTH    ;上位8ビット格納用テーブルアドレスを
                                        ;DEレジスタに格納

        MOV     B,      #8*3            ;データ数をセット

        MOV     A,      #0
ILENGTH_INIT:
        MOV     [HL+B], A               ;結果格納領域のクリア
        DBNZ    B,      $ILENGTH_INIT   ;8*3=24回繰り返す
        INCW    HL                      ;RLOWLENGTH-1しているので元に戻す

        SEL     RB0                     ;レジスタバンクを0に戻す

        MOV     RHIGHCOUNT,     #0      ;上位桁カウンタをクリア

        MOV     RTIMES,         #8      ;測定回数カウンタを初期化(測定回数8回)

        MOV     TMC00,  #0000100        ;[モード制御]
LWAITCAP:
        NOP
        BF      TMIF010, $LWAITCAP      ;最初のキャプチャ完了待ち
        CLR1    TMIF000                 ;INTTM000割り込み要求クリア
        CLR1    TMMK000                 ;INTTM000割り込みマスク解除
        CLR1    TMIF010                 ;INTTM010割り込み要求クリア
        CLR1    TMMK010                 ;INTTM000割り込みマスク解除
        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
```

割り込みにおけるレジスタ・バンクは1を使用してレジスタ破壊を防ぎます。

結果格納領域クリア（初期化）用に－1します。

上位8ビット，下位16ビットの測定回数8回分の結果格納領域をクリアします。

結果格納領域クリア用にRLOWLENGTHを－1していたので，ここでアドレスを元に戻します。

最初のTMIF010の立ち上がり（キャプチャ割り込み）を待ち，TMIF010が立ち上がったら割り込みを許可します。

※アセンブリ言語版では，メイン処理に入る前に，パルス幅測定タイマの設定，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，パルス幅測定の初期化をしています。

【インターバル・タイマのタイム・ベース割り込みを用いたソフトウエアによる測定方法】

メイン処理では次の動作を行います。

【C言語】　(Kx2_Pulse.c)

・パルス幅測定設定関数を呼び出します（タイマの設定）。

・測定状況ステータス・フラグ，測定値格納領域，測定回数カウンタを初期化して割り込み待ちループに入ります。

※測定状況ステータス・フラグについて，

0　：　測定終了

1　：　測定開始

として，測定状況を確認します。

```
/************************************************************************
;       メイン処理
;************************************************************************/
void   main(void){

        fn_InitTimer();                 /*パルス幅測定の初期化          */

        for(g_ucTimes = 0; g_ucTimes < 8; g_ucTimes++){
                                        /*パルス幅データ格納領域を初期化する*/
                g_unTimeCnt[g_ucTimes]    =    0;
                                        /*結果格納領域のクリア          */
        }

        g_ucCapStat   =   0;            /*ステータスフラグを初期化      */
        g_ucTimes     =   8;            /*測定回数カウンタを初期化      */


        PIF0          =   0;            /*INTP0割り込み要求クリア       */
        PMK0          =   0;            /*INTP0割り込み許可             */


        EI();                           /*割り込み許可                  */

        while(1){                       /*割り込み待機無限ループ        */
                NOP();
        }
}
```

パルス幅測定の設定関数を呼び出します。

8回測定分のパルス幅測定データ格納領域をクリアします。

g_unTimeCnt[0]～[7]をクリアします。

結果格納領域の初期化ですでに8回に初期化はされていますが，念のためここでも再び8回に初期化します。

INTP0の立ち上がり，立ち下がりの両エッジを検出します。

【アセンブリ言語】　(Kx2_Pulse.asm)

・測定状況ステータス・フラグ，測定値格納領域，測定回数カウンタを初期化して割り込み待ちループに入ります。

※測定状況ステータス・フラグについて，

0　：　測定終了

1　：　測定開始

として，測定状況を確認します。

```
;*************************************************************************
;       メイン処理
;*************************************************************************
        CLR1    BCAPSTAT                ;ステータスフラグを初期化

        MOVW    HL,         #RTIMCNT-1  ;測定値格納変数のアドレスをHLに格納
        MOV     RTIMES,     #8          ;測定回数カウンタを初期化(8回測定)

        MOV     B,          #8*2        ;データ数をセット

        MOV     A,          #0
IRTIM_INIT:
        MOV     [HL+B],     A           ;結果格納領域のクリア
        DBNZ    B,          $IRTIM_INIT ;8*2=16回繰り返す

        INCW    HL                      ;RTIMCNT-1しているのでここで元に戻す

        CLR1    PIF0                    ;INTP0割り込み要求クリア
        CLR1    PMK0                    ;INTP0割り込み許可

        EI
MMAIN:
        BR      MMAIN
```

結果格納領域クリア（初期化）用に－1します。

8回測定分のパルス幅測定データ格納領域をクリアします。

結果格納領域クリア用にRLOWLENGTHを－1していたので，ここでアドレスを元に戻します。

INTP0の立ち上がり，立ち下がりの両エッジを検出します。

※アセンブリ言語版では，メイン処理に入る前に，パルス幅測定タイマの設定，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，パルス幅測定の初期化をしています。

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。

【タイマのキャプチャ機能を用いたハードウエアによる測定方法】

【C言語】　(Kx2_Pulse.c)

C言語版では次のような定義をします。

・上位と下位のパルス幅測定データ格納変数を定義します。
・パルス幅の上位桁をカウントするカウンタを定義します。
・測定回数をカウントするカウンタを定義します。

```
/*=========================================================================
;       使用する変数定義（グローバル変数）
;=========================================================================*/

static unsigned int      g_unLowLength[8];     ← 下位の16ビット・パルス幅測定データ格納変数を定義します。

static unsigned char     g_ucHighLength[8];    ← 上位の8ビット・パルス幅測定データ格納変数を定義します。

sreg   static   unsigned char   g_ucHighCount; ← パルス幅の上位桁をカウントするカウンタを定義します。

sreg   static   unsigned char   g_ucTimes;     ← 測定回数をカウントするカウンタを定義します。
```

【アセンブリ言語】　(Kx2_Pulse.asm)

アセンブリ言語版では次のような定義をします。

・上位と下位のパルス幅測定データ格納変数を定義します。
・パルス幅の上位桁をカウントするカウンタを定義します。
・測定回数をカウントするカウンタを定義します。

```
;=========================================================================
;       使用する変数の定義
;=================================
RLOWLENGTH:     DS      16      ← 下位の16ビット・パルス幅測定データ格納変数を定義します。
                                  16ビット・データを8回測定するので，16バイトの作業領域を確保します。

RHIGHLENGTH:    DS      8       ← 上位の8ビット・パルス幅測定データ格納変数を定義します。
                                  8ビット・データを8回測定するので，8バイトの領域を確保します。

DRAM    DSEG    SADDR
RHIGHCOUNT:     DS      1       ← パルス幅の上位桁をカウントするカウンタを定義します。

RTIMES:         DS      1       ← 測定回数をカウントするカウンタを定義します。
```

【インターバル・タイマのタイム・ベース割り込みを用いたソフトウエアによる測定方法】

【C言語】　(Kx2_Pulse.c)

C言語版では次のような定義をします。

- パルス幅格納変数を定義します。
- 測定回数をカウントするカウンタを定義します。
- 測定状況を確認するステータス・フラグを定義します。

```
/*=========================================================================
;       使用する変数定義（グローバル変数）
;=========================================================================*/

static unsigned int       g_unTimeCnt[8];

sreg   static   unsigned  char  g_ucTimes;

sreg   static   unsigned  char  g_ucCapStat;
```

パルス幅格納変数を定義します。

測定回数をカウントするカウンタを定義します。

測定状況を確認するステータス・フラグを定義します。

【アセンブリ言語】　(Kx2_Pulse.asm)

アセンブリ言語版では次のような定義をします。

- パルス幅格納変数を定義します。
- 測定回数をカウントするカウンタを定義します。
- 測定状況を確認するステータス・フラグを定義します。

```
;=========================================================================
;       使用する変数の定義
;=========================================================================
DRAM            DSEG      SADDR
RTIMCNT:        DS        16

RTIMES:         DS        1

BRAM            BSEG
BCAPSTAT        DBIT
```

パルス幅格納変数を定義します。
16ビット・データを8回測定するため16バイトの作業領域を確保します。

測定回数をカウントするカウンタを定義します。

測定状況を確認するステータス・フラグを定義します。

4.7　割り込み処理

【タイマのキャプチャ機能を用いたハードウエアによる測定方法　：　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。

【C言語】 (Kx2_Pulse.c)

・キャプチャ割り込み（INTTM010）と同時でなく，かつCR010（キャプチャ・データ）が0xFFFFでなければ上位桁をインクリメントする。

```
/***********************************************************************
;       割り込み処理INTTM000
;***********************************************************************/
__interrupt    void   fn_inttm000(void){

       if(!((TMIF010==1)   &&   (CR010==0xFFFF))){

              g_ucHighCount++;
       }
}
```

キャプチャ割り込み（INTTM010）と同時でないか，CR010（キャプチャ・データ）が0xFFFFでなければ上位桁をインクリメントします。

上位桁をインクリメントします。

【アセンブリ言語】 (Kx2_Pulse.asm)

・キャプチャ割り込み（INTTM010）と同時でなく，かつCR010（キャプチャ・データ）が0xFFFFでなければ上位桁をインクリメントする。

```
;***********************************************************************
;       割り込み処理INTTM000
;***************************************
IINTTM000:
       SEL     RB1                     ;レジスタバンクを1に切り替え

       BF      TMIF010,  $HINCHIGHCOUNT  ;INTTM010割り込み(キャプチャ割り込み)
                                         ;と同時なら割り込みから復帰

       MOVW    AX,       CR010
       CMPW    AX,       #0FFFFH         ;CR010(キャプチャデータ)と0xFFFFを比較

       BZ      $HENDINTTM000             ;0xFFFFならば割り込みから復帰

HINCHIGHCOUNT:
       INC     RHIGHCOUNT
HENDINTTM000:
       RETI                              ;割り込み処理から復帰
```

割り込みにおけるレジスタ・バンクは1を使用してレジスタ破壊を防ぎます。

キャプチャ割り込み（INTTM010）と同時でないか，CR010（キャプチャ・データ）が0xFFFFでなければ上位桁をインクリメントします。

上位桁をインクリメントします。

【タイマのキャプチャ機能を用いたハードウエアによる測定方法　：　INTTM010割り込み処理】

INTTM010割り込み処理では，次の動作を行います。

【C言語】 (Kx2_Pulse.c)

- キャプチャ・データと上位桁データを各パルス幅データ格納変数に保存します。
- 測定回数をカウントして，8回の測定が完了したら各割り込みをマスクします。

```
/***********************************************************************
;     割り込みINTTM010
;***********************************************************************/
__interrupt void    fn_inttm010(void){

      g_unLowLength[8 - g_ucTimes]  =  CR010;
                                       /*キャプチャデータの読み出し          */

      g_ucHighLength[8 - g_ucTimes] =  g_ucHighCount;
                                       /*上位桁の読み出し                    */

      g_ucHighCount       =    0;      /*上位桁をクリア                      */

      g_ucTimes--;                     /*測定回数をデクリメント              */

      if(g_ucTimes  ==  0){            /*8回測定終了ならば各割り込みマスク   */
            TMMK000 = 1;               /*INTTM000割り込みをマスク            */
            TMMK010 = 1;               /*INTTM010割り込みをマスク            */
      }
}
```

キャプチャ・データをパルス幅格納変数に保存します。

上位桁データをパルス幅格納変数に保存します。

上位桁カウンタをクリアします。

測定回数をカウントして，8回の測定が完了したらINTTM000，INTTM010割り込みをマスクします。

【アセンブリ言語】 (Kx2_Pulse.asm)

・キャプチャ・データと上位桁データを各パルス幅データ格納変数（HLレジスタを使用）に保存し，そのテーブル・アドレスを更新します。

・測定回数をカウントして，8回の測定が完了したら各割り込みをマスクします。

```
;*************************************************************************
;       割り込み処理INTTM010
;*************************************
IINTTM010:
        SEL     RB1                     ;レジスタバンクを1に切り替え

        MOVW    AX,     CR010           ;キャプチャデータの読み出し

        MOV     [HL+1], A               ;下位16ビット格納テーブルにキャプチャ
                                        ;データ16ビット中上位8ビットを保存

        XCH     A,      X               ;A,Xレジスタのデータ交換

        MOV     [HL],   A               ;下位16ビット格納テーブルにキャプチャ
                                        ;データ16ビット中下位8ビットを保存

        MOV     A,      RHIGHCOUNT      ;上位桁の読み出し
        MOV     [DE],   A               ;上位8ビット格納テーブルに保存

        INC     L                       ;下位16ビット格納用テーブルアドレス+2
        INC     L
        INC     E                       ;上位8ビット格納用テーブルアドレス+1

        MOV     RHIGHCOUNT,     #0      ;上位桁をクリア

        DBNZ    RTIMES, $HENDINTTM010   ;測定回数(RTIMES)をデクリメントして0
                                        ;(8回終了)でなければHENDINTTM010に分岐

        SET1    TMMK000                 ;INTTM000割り込みをマスク
        SET1    TMMK010                 ;INTTM000割り込みをマスク

HENDINTTM010:
        RETI                            ;割り込み処理から復帰
```

割り込みにおけるレジスタ・バンクは1を使用してレジスタ破壊を防ぎます。

キャプチャ・データをHLレジスタ（パルス幅格納変数）に保存します。

上位桁データをDEレジスタ（パルス幅格納変数）に保存します。

パルス幅データ格納テーブルのアドレスを更新します。

上位桁カウンタをクリアします。

測定回数をカウントして，8回の測定が完了したらINTTM000，INTTM010割り込みをマスクします。

【インターバル・タイマのタイム・ベース割り込みを用いたソフトウエアによる測定方法　：　INTP0割り込み処理】

INTP0割り込み処理では，次の動作を行います。

【C言語】 (Kx2_Pulse.c)

・パルスの立ち上がりで測定開始し，立ち下がりで測定終了します（ハイ・レベル期間を測定）。

※ここでは，タイマ・カウンタを使用した方法のTM50版を例に取って説明します。

```
/************************************************************************
;       割り込み処理INTP0
;************************************************************************/
__interrupt void  fn_intp0(void){

      if((g_ucCapStat == 0) && (P12.0 == 1)){

          TCE50   =     1;              /*タイマ動作開始              */
          TMIF50  =     0;              /*タイマ割り込み要求クリア    */
          TMMK50  =     0;              /*タイマ割り込み許可          */

          g_ucCapStat   =               1;

      }
      else if((g_ucCapStat == 1) && (P12.0 == 0)){

          TCE50   =     0;              /*タイマ動作停止              */
          TMIF50  =     0;              /*タイマ割り込み要求クリア    */
          TMMK50  =     1;              /*タイマ割り込み禁止          */

          g_ucCapStat   =               0;

          g_ucTimes--;                  /*測定回数カウンタをデクリメント*/


          if(g_ucTimes == 0){           /*測定回数カウンタが0ならばINTP0割り込みを禁止する*/

                  PMK0  =               1;
                                        /*INTP0割り込み禁止           */
          }
      }
}
```

測定開始前かつパルスがハイ・レベルならば，タイマの動作を開始します。

ステータス・フラグをセット（測定開始）します。

測定開始済みかつパルスがロウ・レベルならば，タイマの動作を停止します。

ステータス・フラグをクリア（測定終了）します。

測定回数をカウントし，8回の測定が完了したらINTP0割り込みを禁止して，全測定を終了します。

【アセンブリ言語】 (Kx2_Pulse.asm)

・パルスの立ち上がりで測定開始し，立ち下がりで測定終了します（ハイ・レベル期間を測定）。

※ここでは，タイマ・カウンタを使用した方法のTM50版を例に取って説明します。

```
;*************************************************************************
;       割り込み処理INTP0
;*************************************************************************
IINTP0:
        PUSH    AX                      ;AXレジスタのデータをスタックに退避

        BT      BCAPSTAT,       $HPLSEND
        BF      P12.0,  $HPLSRET

        SET1    TCE50                   ;タイマ動作開始
        CLR1    TMIF50                  ;タイマ割り込み要求クリア
        CLR1    TMMK50                  ;タイマ割り込み許可
        SET1    BCAPSTAT

        BR      HPLSRET
HPLSEND:
        CLR1    TCE50                   ;タイマ動作停止
        CLR1    TMIF50                  ;タイマ割り込み要求クリア
        SET1    TMMK50                  ;タイマ割り込み禁止

        CLR1    BCAPSTAT

        DBNZ    RTIMES, $HPLSUPDATE

        SET1    PMK0                    ;INTP0割り込み禁止

HPLSUPDATE:
        INC     L                       ;測定値格納変数テーブルアドレス更新
        INC     L                       ;テーブルアドレス+2(16bitデータのため)

HPLSRET:
        POP     AX                      ;AXレジスタのデータを復帰
        RETI                            ;割り込みから復帰
```

測定開始前かつパルスがハイ・レベルならば，タイマの動作を開始し，測定開始済みかつパルスがロウ・レベルならば，タイマの動作を停止します。

ステータス・フラグをセット（測定開始）します。

ステータス・フラグをクリア（測定終了）します。

測定回数をカウントし，8回の測定が完了したらINTP0割り込みを禁止して，全測定を終了します。

パルス幅データ格納テーブルのアドレスを更新します。

【インターバル・タイマのタイム・ベース割り込みを用いたソフトウエアによる測定方法　：　タイマ割り込み処理】

タイマ割り込み処理では，次の動作を行います。

【C言語】 (Kx2_Pulse.c)

・パルス幅データ格納変数をインクリメントします。

※ここでは，タイマ・カウンタを使用した方法のTM50版を例に取って説明します。

```
/***********************************************************************
;       割り込み処理INTTM50
;***********************************************************************/
__interrupt     void    fn_inttm50(void){

        g_unTimeCnt[8 - g_ucTimes]++;

}
```

パルス幅データ格納変数をインクリメントします。

【アセンブリ言語】 (Kx2_Pulse.asm)

・パルス幅データ格納変数（HLレジスタを使用）をインクリメントします。

※ここでは，タイマ・カウンタを使用した方法のTM50版を例に取って説明します。

```
;***********************************************************************
;       割り込み処理INTTM50
;***********************************************************************
IINTTM50:
        PUSH    AX                      ;AXレジスタのデータをスタックに退避

        MOV     A,      [HL]
        ADD     A,      #1              ;測定値+1
        MOV     [HL],   A

        MOV     A,      [HL+1]
        ADDC    A,      #0              ;測定値+CY
        MOV     [HL+1], A

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

HLレジスタにアドレスされた16ビットのパルス幅データ格納変数をそのHLレジスタを用いてカウント・アップします。

パルス幅データが0xFFを越えるときに1バイト目のオーバフローによってキャリー・フラグがセットされるので，そのキャリー・フラグを2バイト目のデータに加算していくことで2バイトのデータを測定しています。

【コラム】アセンブリ言語での汎用レジスタの使用法について

パルス幅測定のサンプル・プログラムのアセンブリ言語版では，以下のように汎用レジスタをグローバル変数（いつまでも残る変数）として使用しています。

【タイマのキャプチャ機能を用いたハードウエアによる測定方法】

HLレジスタを下位16ビットのパルス幅の測定に用い，DEレジスタを上位8ビットのパルス幅の測定に用います。また，割り込み処理において，レジスタ・バンクは1を使用してレジスタ破壊を防いでいます。

【インターバル・タイマのタイム・ベース割り込みを用いたソフトウエアによる測定方法】

HLレジスタのみをパルス幅の測定に用います。また，HLレジスタはグローバル変数として使用しているので，メイン処理ではHLレジスタは使えません。

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。

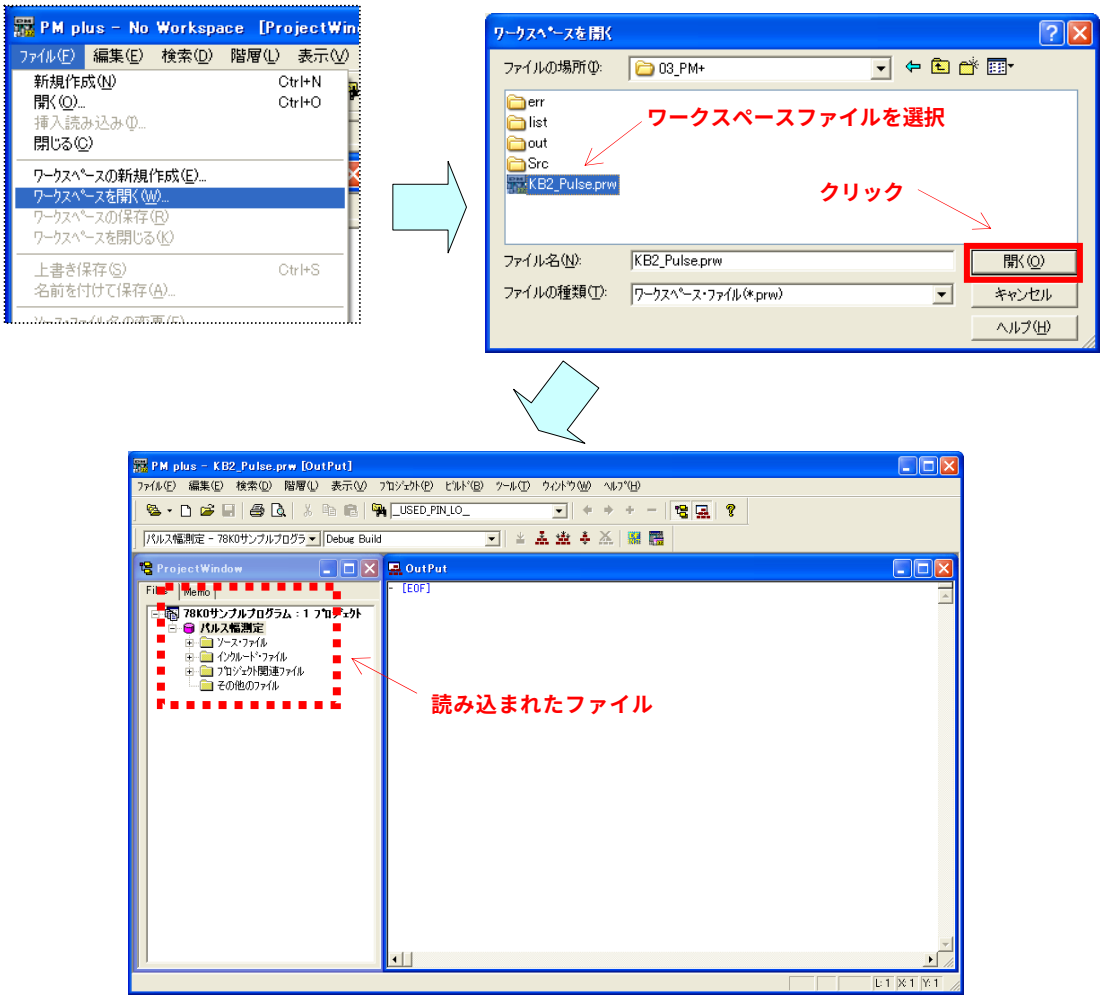
PM+操作方法の詳細については，PM+ プロジェクト・マネージャ　ユーザーズ・マニュアルを参照してください。

(1) PM+を起動してください。

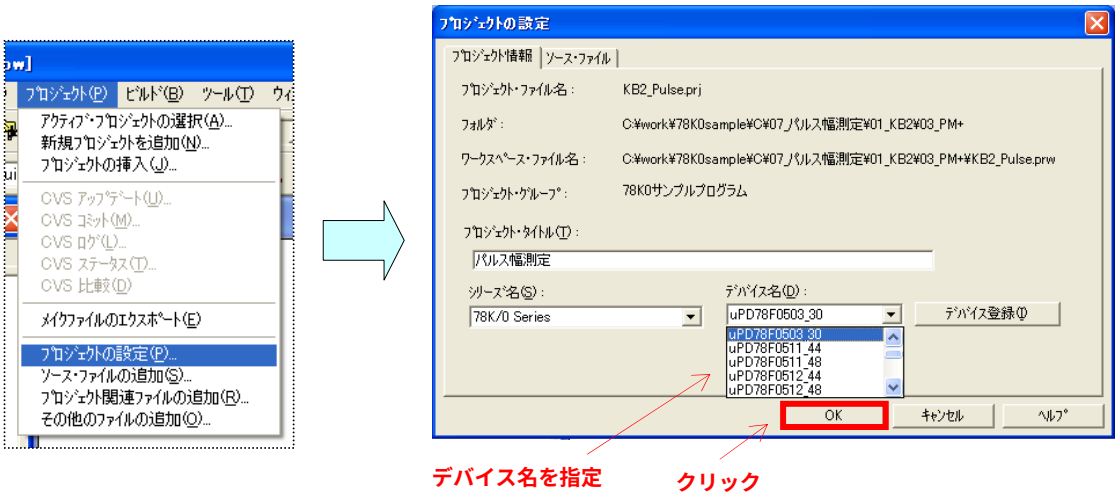
(2) [ファイル] → [ワークスペースを開く] から，「Kx2_Pulse.prw」[注]を選択し，[開く] ボタンをクリックしてください。ワークスペースが作成され，その中にソース・ファイルが自動的に読み込まれます。

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_Pulse.prw」

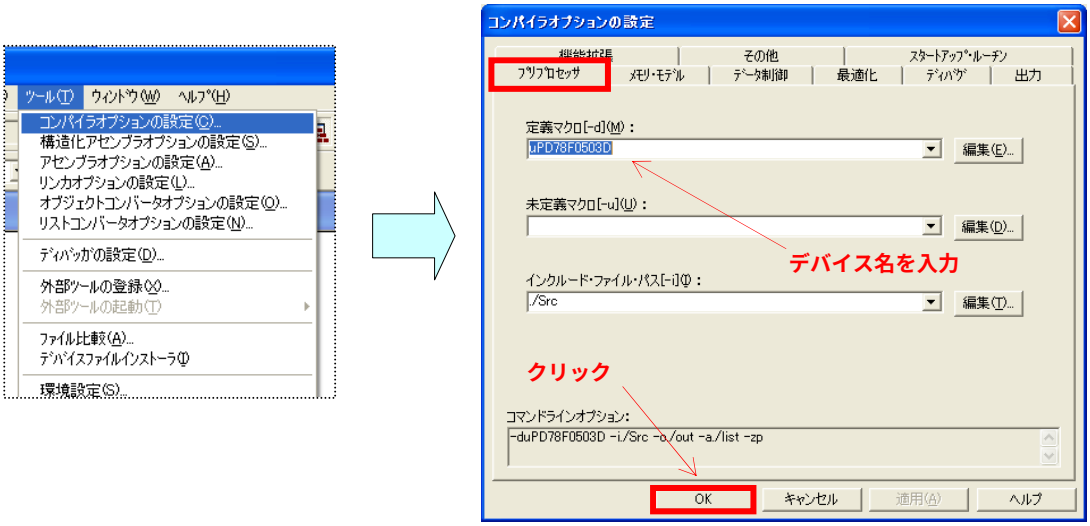


(3) [プロジェクト] → [プロジェクトの設定] を選択してください。[プロジェクトの設定] 画面が表示されたら，使用するデバイス名を選択（デフォルトでは，ROM/RAMサイズの最も大きいデバイスが選択）し，[OK] ボタンをクリックしてください。



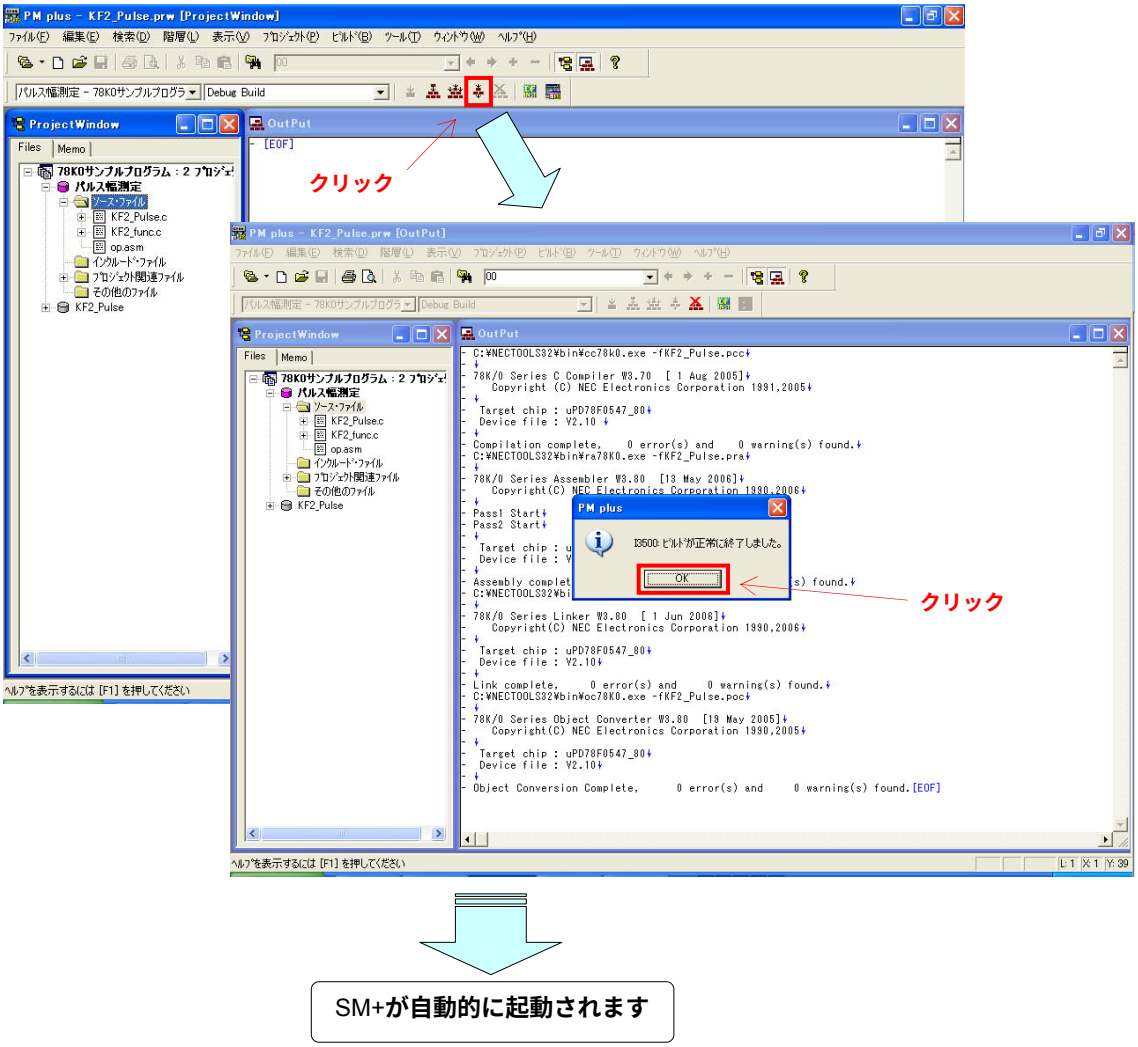
※μPD78F0500_36, μPD78F0501_36, μPD78F0502_36, μPD78F0503_36は選択しないでください。

(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_Pulse.c」と「Kx2_Init.c」と「KB2_Mem.sam」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。

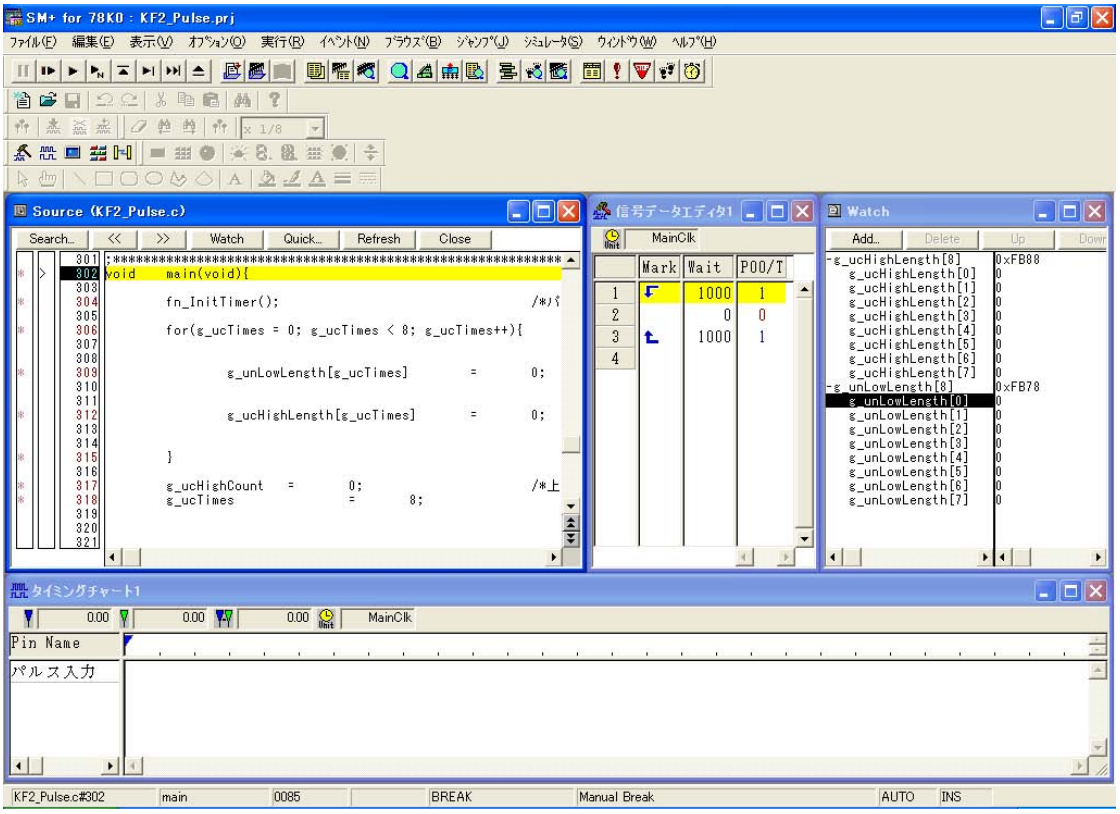


5. 2　SM+での動作

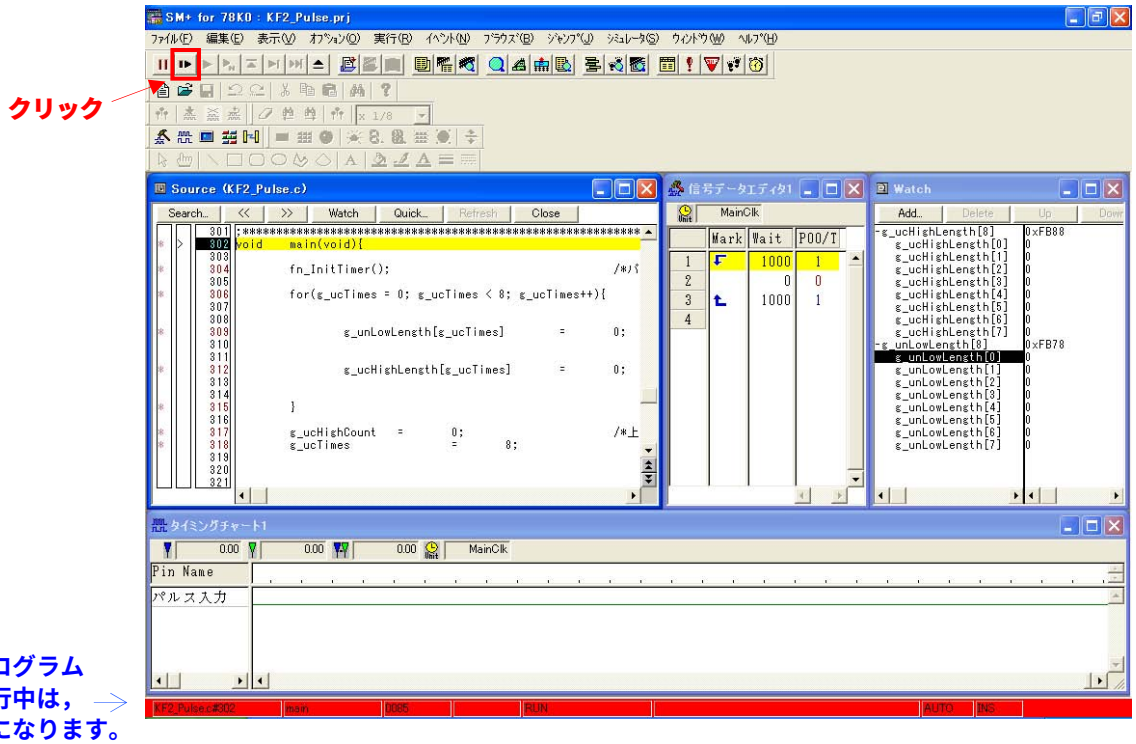
ここでは，SM+の入出力パネル・ウインドウやタイミング・チャート・ウインドウ上での動作確認の例を説明します。

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。

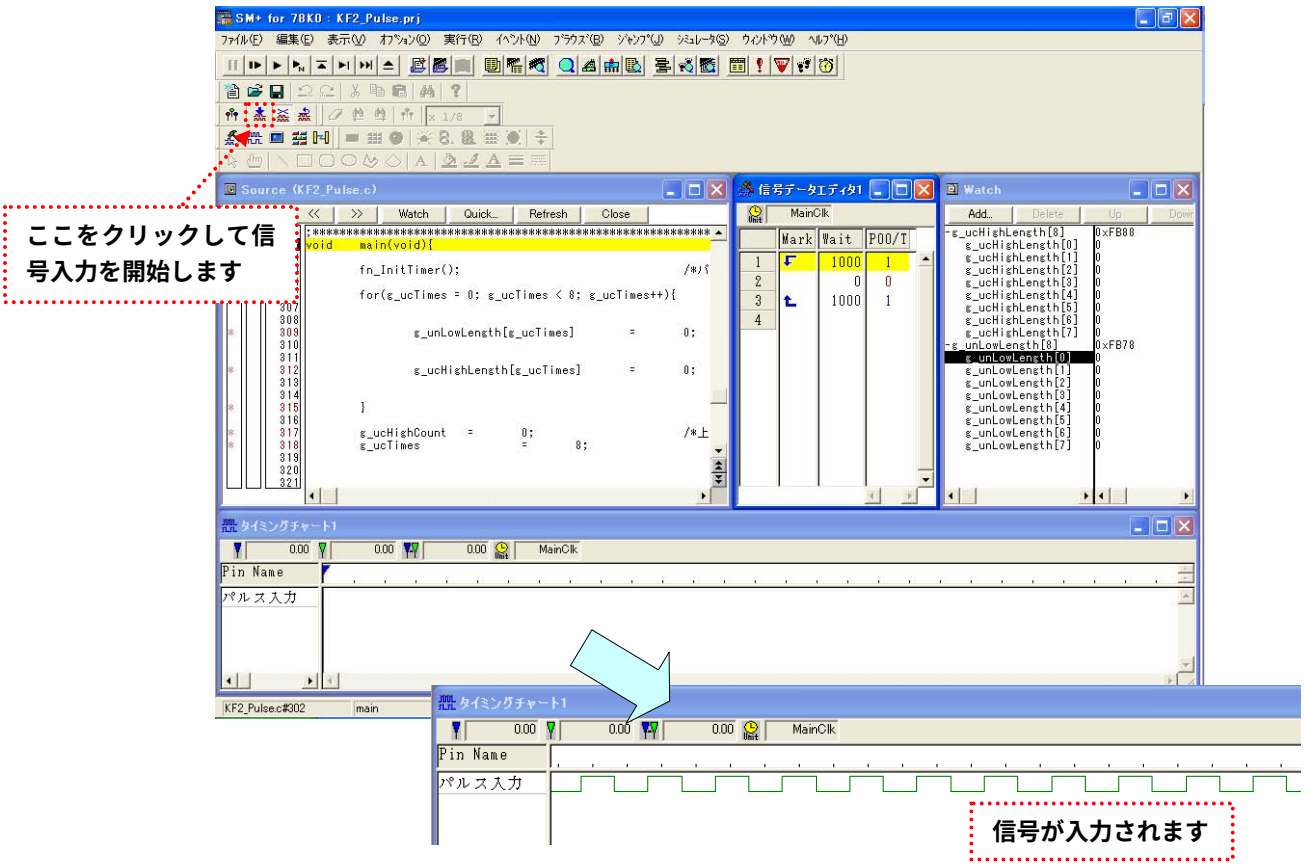
（1）PM+の「ビルド→ディバグ」からSM+を起動（5. 1を参照）すると，次のような画面になります。



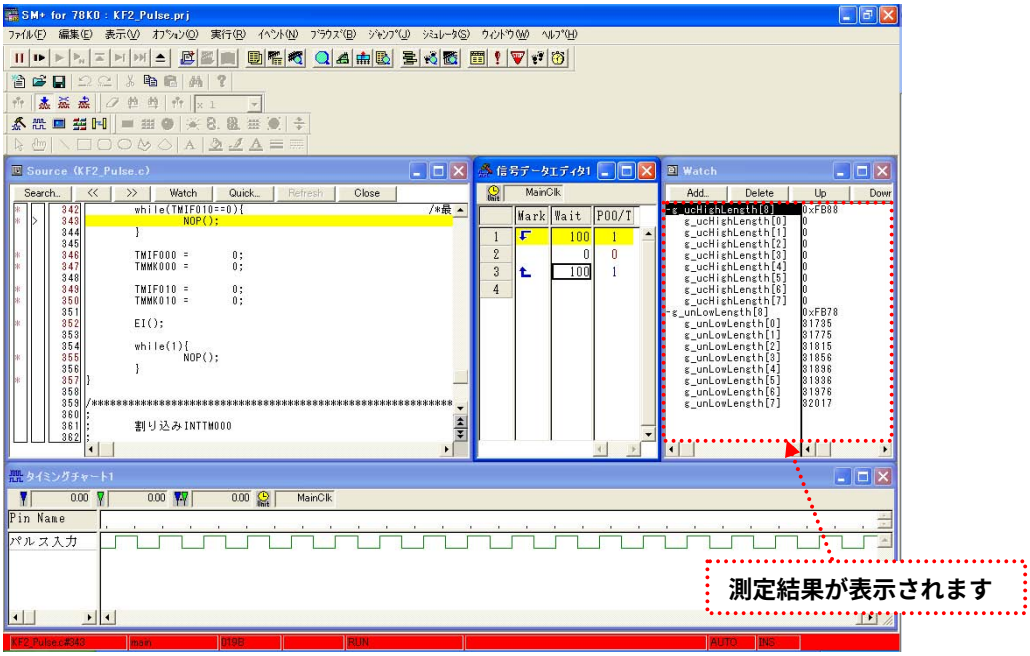
(2)  （「リスタート」ボタン）をクリックしてください。CPUリセット後，プログラムが実行され，次のような画面になります。



(3) 信号データ・エディタで作成した波形の信号入力を開始します。



（4）信号入力が終了すると，Watchウインドウ内に登録したパルス幅データ変数の値が表示されます。



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
```

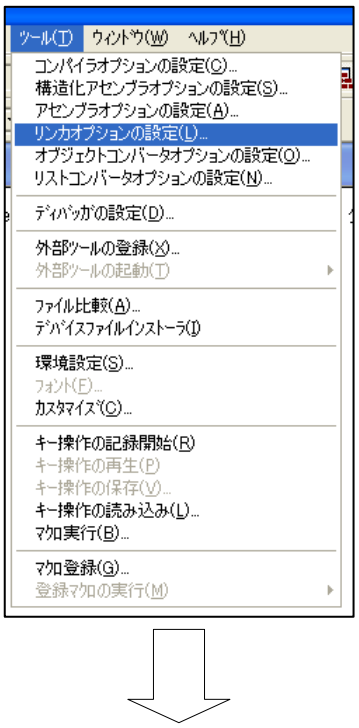
0002H番地はオンチップ・デバッグ使用領域として空けるため，0002H番地はコメント・アウトして，CSEGで再び0004H番地を定義しています。

※C言語版では不要です。

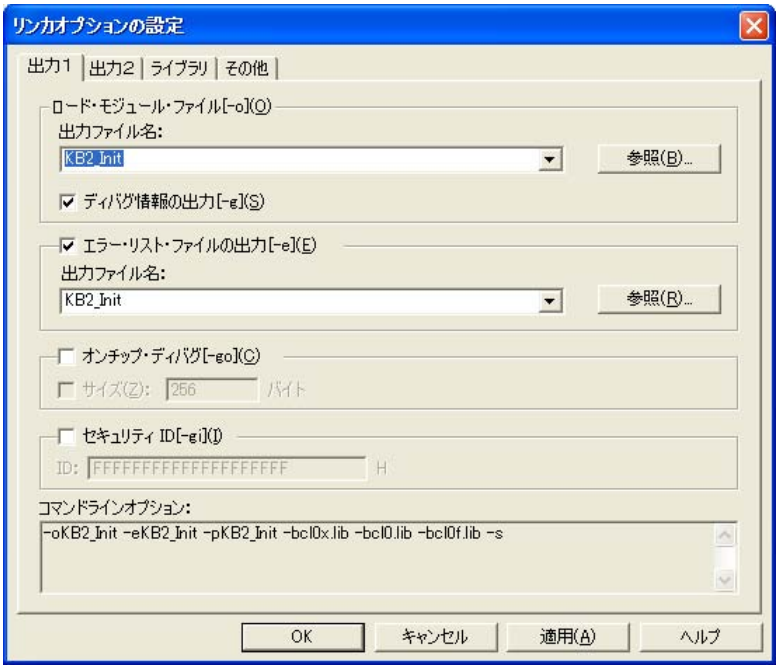
(3) リンカの設定

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。

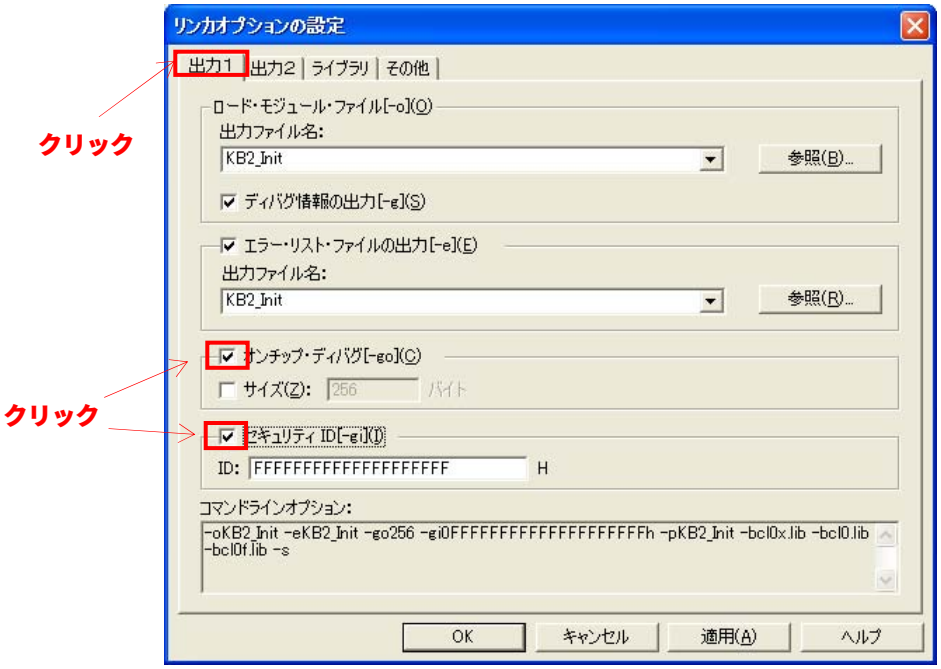
PM+の「ツール」メニューから「リンカオプションの設定」を選択してください。



「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