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

時計用タイマ編

この資料は，サンプル・プログラムの「時計用タイマ」の動作概要と，マイコンの基本的な時計用タイマの設定を説明したものです。サンプル・プログラムでは，時計用タイマの設定を行ったあとに，タイマのカウントにしたがって7セグメントLED、およびLEDの出力を制御します。

対象デバイス

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　U19037JJ1V0AN00（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，時計用タイマの設定をします。本プログラムでは，ポートに接続した秒表示LEDを10秒ごとに順番に点灯させます。そして，1分後に7セグメントLEDに測定時間を表示させ，秒表示LEDは全消灯させます。それ以降はこの繰り返しで簡易時計を実現させます。

なお，クロック関連の設定や周辺ハードウエアの初期設定などのサンプル・プログラムの初期設定編と同内容の部分は，別ファイルとして外部関数コール（C言語版），サブルーチン・コール（アセンブリ言語版）にて使用しています。

(1) 時計用タイマ設定の内容

- 時計用タイマの設定
- 割り込みの設定

(2) 時計タイマ割り込み処理内容

- 秒カウンタをインクリメントし，10 [s] ごとに秒表示LEDの変更要求を出します。

※割り込みベクタは002EH（INTWT）に設定します。
（ベクタ・テーブル002EH番地の割り込みハンドラ名称を “ IINIT ” から “ IINTWT ” に変更します。）

(3) メイン処理動作の内容

- 各カウンタを初期化し，カウンタ値更新・表示変更の関数を呼び出し続ける無限ループに入り，INTWT割り込みを待ちます。

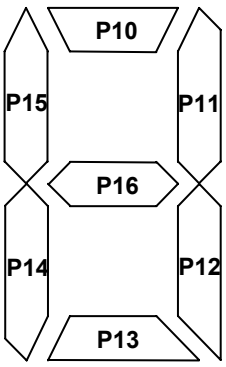
【ポートとLED表示の対応】

簡易時計用7セグメントLEDと秒表示LEDを制御するポートとの対応は下記のとおりです。

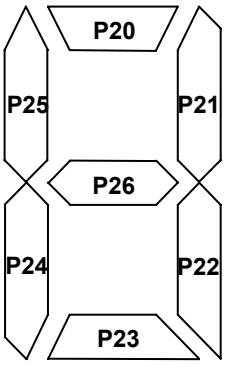
P00	秒表示LED1（10秒）
P01	秒表示LED2（20秒）
P17	秒表示LED3（30秒）
P40	秒表示LED4（40秒）
P41	秒表示LED5（50秒）
P20-P26	7セグメントLED（1分の桁）
P10-P16	7セグメントLED（10分の桁）

【ポートと7セグメントLED表示の対応】

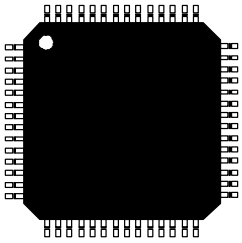
7セグメントLED表示と制御するポートとの対応は以下のとおりです。



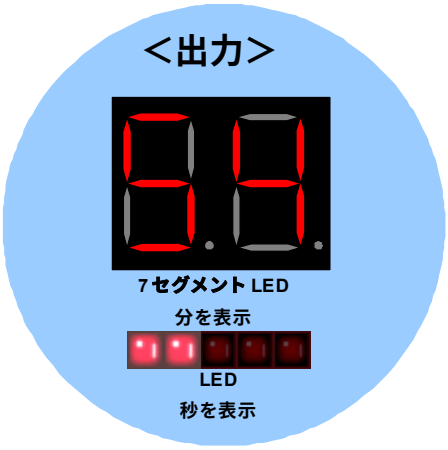
10分の桁



1分の桁



78K0/Kx2
マイクロコントローラ



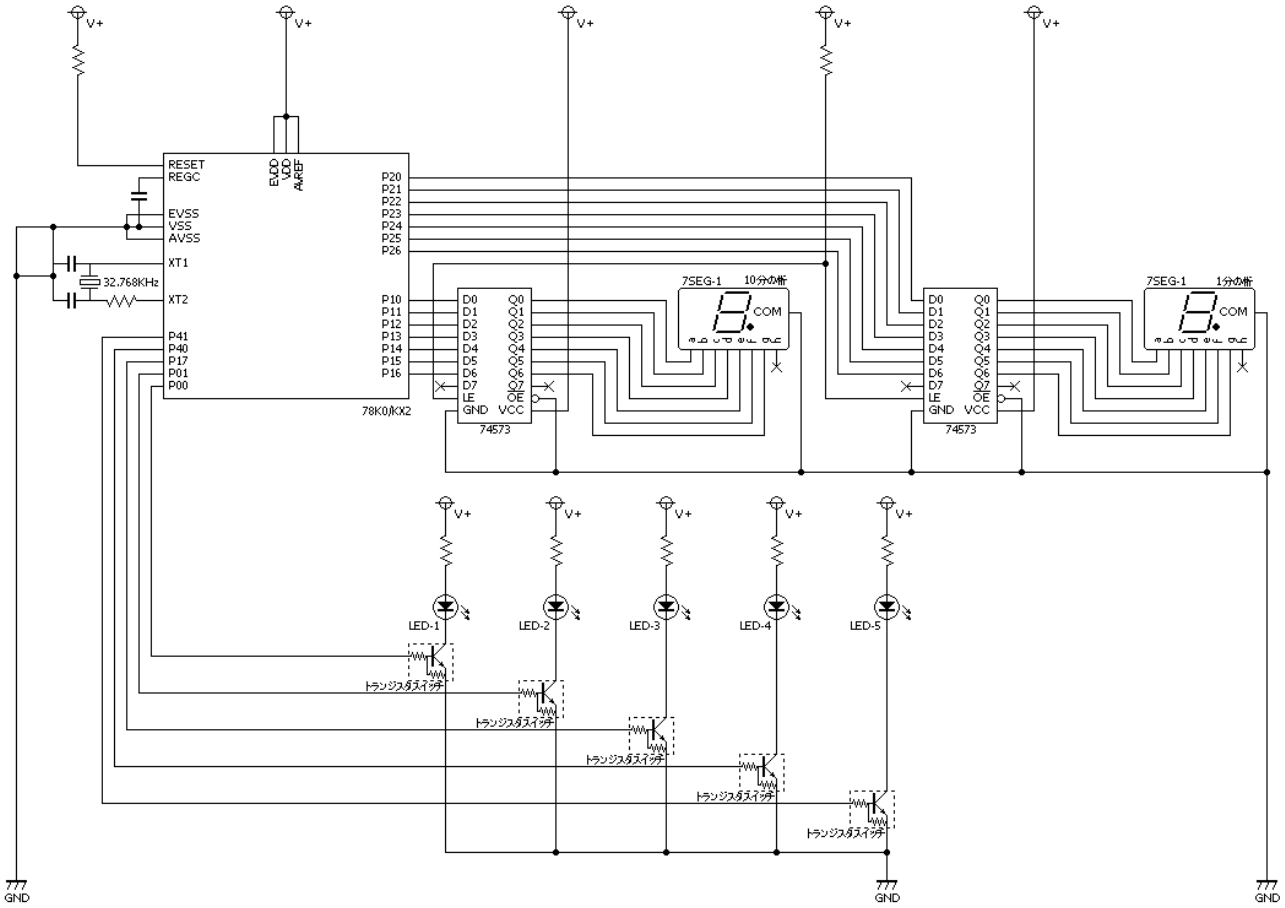
注意　デバイス使用上の注意事項については，78K0/Kx2　ユーザーズ・マニュアルを参照してください。

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。**

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

秒表示用のLEDは簡易時計用タイマのカウント値により点灯します。

(2) 7セグメントLED

分表示用の7セグメントLEDは簡易時計用タイマでカウント値により点灯します。

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・プログラムの時計用タイマ設定と動作概要，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_Wt.c	時計用タイマの設定，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_Wt.prw	統合開発環境 PM+用ワーク･スペース・ファイル		●	
Kx2_Wt.prj	統合開発環境 PM+用プロジェクト・ファイル		●	
Kx2_Wt.pri Kx2_Wt.prs Kx2_Wt.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_Wt0.pnl	システム・シミュレータ SM+ for 78K0/Kx2用入出力パネル・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/KC2の場合 → ”KC2_Wt.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_Wt.asm	時計用タイマの設定，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_Wt.prw	統合開発環境 PM+用ワーク･スペース・ファイル		●	
Kx2_Wt.prj	統合開発環境 PM+用プロジェクト・ファイル		●	
Kx2_Wt.pri Kx2_Wt.prs Kx2_Wt.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_Wt0.pnl	システム・シミュレータ SM+ for 78K0/Kx2用入出力パネル・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/KC2**の場合** → ”KC2_Wt.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。

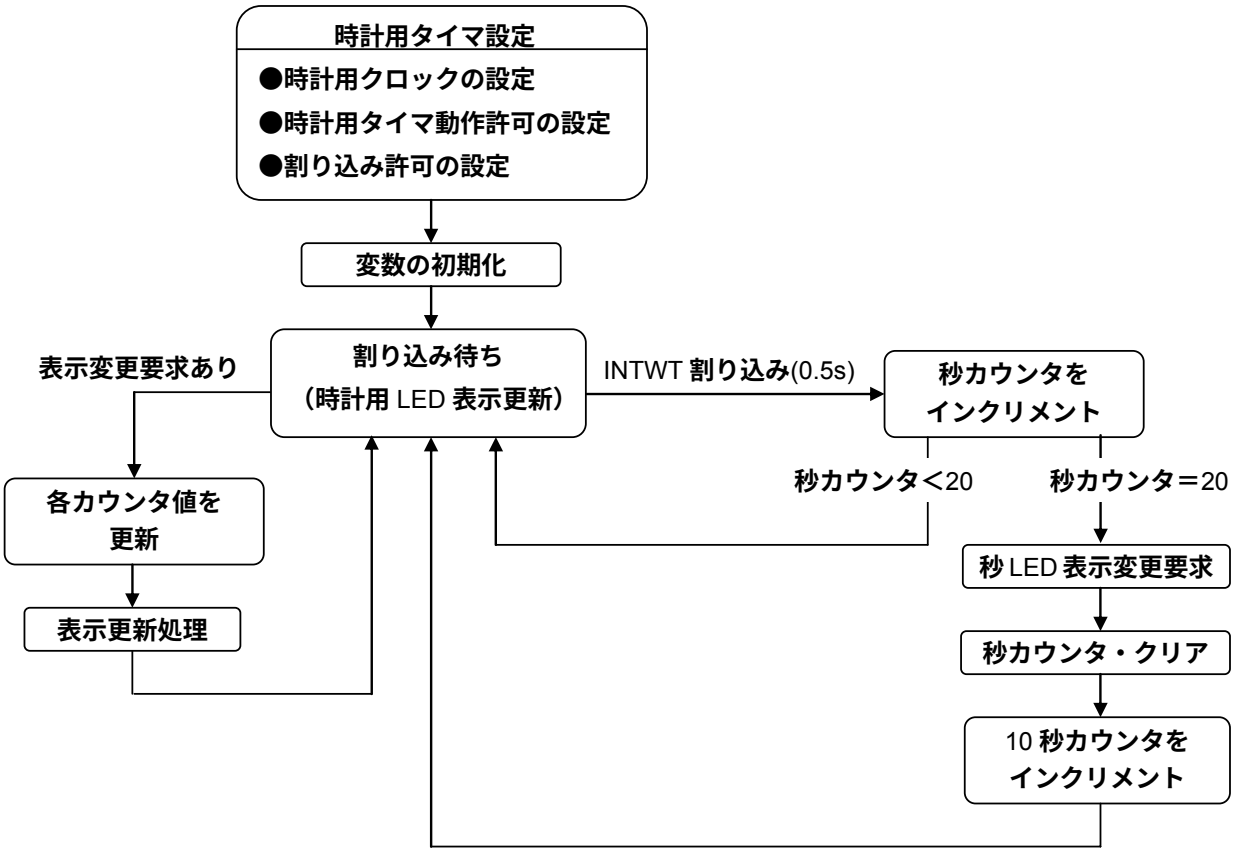
・時計用タイマ
・出力ポート　　：P00, P01, P17, P40, P41（秒表示用LEDで使用）
　　：P10～P16, P20～P26（分表示用7セグメントLEDで使用）

3. 3　時計用タイマ設定と動作概要

このサンプル・プログラムでは，時計用タイマの設定を行います。

設定完了後は，時計用タイマ割り込みで秒カウンタをインクリメントし，10秒ごとに秒表示LEDの変更要求を出し（変更要求フラグをセット），10秒カウンタをインクリメントします。そして，10秒ごとにLEDを順次点灯させ，1分後にLEDはすべて消灯し，7セグメントLEDの時間（分）を更新します。

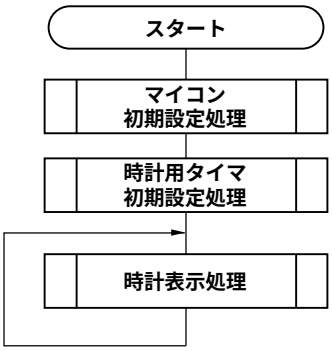
詳細については，次の状態遷移図（ステート・チャート）に示します。



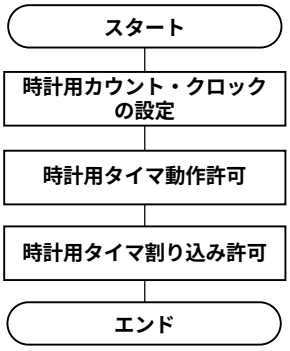
3. 4　フロー・チャート

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。

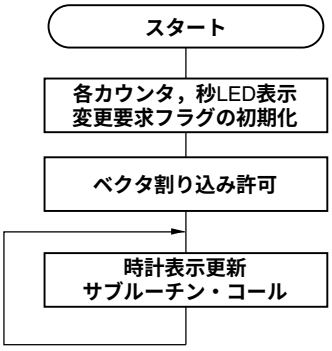
○　**ゼネラル・フロー（C言語版：**Kx2_Wt.c　**アセンブリ言語版：**Kx2_Wt.asm**）**



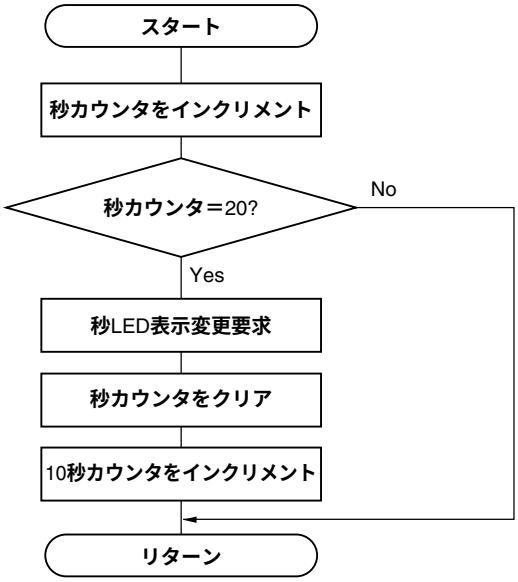
○　**時計用タイマ初期設定処理（C言語版：**Kx2_Wt.c　**アセンブリ言語版：**Kx2_Wt.asm**）**



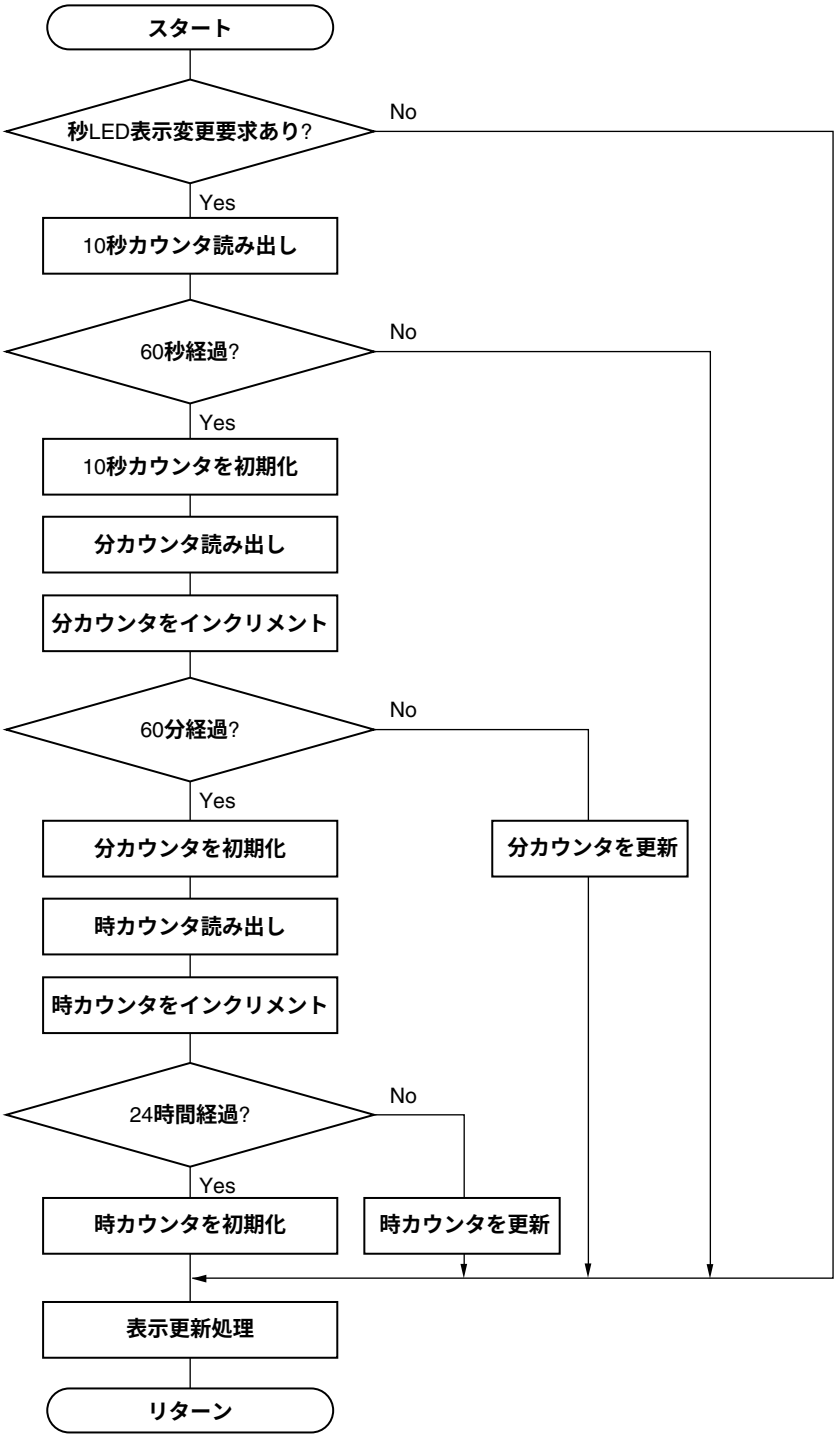
○メイン処理（C言語版：Kx2_Wt.c　**アセンブリ言語版：**Kx2_Wt.asm**）**



○時計用タイマ割り込み処理（C言語版：Kx2_Wt.c　**アセンブリ言語版：**Kx2_Wt.asm**）**



○時計表示更新サブルーチン処理（C言語版：Kx2_Wt.c　**アセンブリ言語版：**Kx2_Wt.asm**）**



第4章　設定方法について

この章では，前処理指令（C言語版），時計用タイマの設定，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。

【C言語】 (Kx2_Pulse.c)

```
/*==========================================================================
        前処理指令(#pragma指令)
==========================================================================*/
#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  INTWT    fn_intWt           <-- 割り込み関数宣言をします。
```

4. 2　時計用タイマの設定

時計用タイマの設定をします。このサンプル・プログラムでは時計用タイマのカウント・クロックとしてサブシステム・クロックを使用します。

(1) 時計用タイマ・クロック設定

時計用タイマのクロック，および，動作モードを設定します

図4－1　時計用タイマ動作モード・レジスタ

WTM

WTM7	WTM6	WTM5	WTM4	WTM3	WTM2	WTM1	WTM0

時計用タイマ動作許可

0	動作停止
1	動作許可

5ビット・カウンタの動作制御

0	動作停止後クリア
1	スタート

時計用タイマの割り込み時間

0	0	$2^{14}/f_w$
0	1	$2^{13}/f_w$
1	0	$2^5/f_w$
1	1	$2^4/f_w$

プリスケーラのインターバル時間の選択

0	0	0	$2^4/f_w$
0	0	1	$2^5/f_w$
0	1	0	$2^6/f_w$
0	1	1	$2^7/f_w$
1	0	0	$2^8/f_w$
1	0	1	$2^9/f_w$
1	1	0	$2^{10}/f_w$
1	1	1	$2^{11}/f_w$

時計用タイマのカウント・クロック周波数（f_w）の選択

0	$f_{PRS}/2^7$
1	f_{SUB}

注意　時計用タイマ動作中に，カウント・クロック，インターバル時間の変更をしないでください。

備考1.　f_{PRS} ： 周辺ハードウエア・クロック周波数

2.　f_{SUB} ： サブシステム・クロック周波数

3.　f_W ： 時計用タイマのカウント・クロック周波数 ($f_{PRS}/2^7$またはf_{SUB})

【例】　・カウント・クロックにサブシステム・クロック周波数を使用
　　・カウンタ・スタートし，時計用タイマ動作許可に設定
　　（サンプル・プログラムの設定と同内容）

WTM

1	0	0	0	0	0	0→1	0→1

時計用タイマ動作許可

0→1	動作停止→動作許可

5ビット・カウンタの動作制御

0→1	動作停止後クリア→スタート

時計用タイマの割り込み時間

0	0	$2^{14}/f_w$

プリスケーラのインターバル時間の選択

0	0	0	$2^4/f_w$

時計用タイマのカウント・クロックの選択

1	f_{SUB}

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_Wt.c)

```
WTM     =       0b10000000;     カウント・クロックをサブシステム・クロックに設定します。
WTM1    =       1;              5ビット・カウンタ動作を開始します。
WTM0    =       1;              時計用タイマの動作を開始します。
```

【アセンブリ言語】 (Kx2_Wt.asm)

```
MOV     WTM,    #10000000B      カウント・クロックをサブシステム・クロックに設定します。
SET1    WTM1                    5ビット・カウンタ動作を開始します。
SET1    WTM0                    時計用タイマの動作を開始します。
```

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セット／クリアします。

図4－2　割り込み要求フラグ・レジスタ（IF1L）のフォーマット

IF1L

0	PIF6	WTIF	KRIF	TMIF51	WTIIF	SRIF0	ADIF

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。

図4－3　割り込みマスク・フラグ・レジスタ（MK1L）のフォーマット

MK1L

1	PMK6	WTMK	KRMK	TMMK5	WTIMK	SRMK0	ADMK
				1			

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

【例】　・WTIFをクリアする

・WTMKに0をセットし割り込みを許可する

（サンプル・プログラムの設定と同内容）

IF1L

		0					

割り込み要求フラグ

0	要求クリア

MK1L

		0					

割り込み要求マスク・フラグ

0	割り込み処理許可

サンプル・プログラムでは1ビット・アクセスにて設定しています。

【C言語】

```
WTIF = 0;
WTMK = 0;
```

INTWT割り込み要求クリア

INTWT割り込み要求クリア

【アセンブリ言語】

```
CLR1    WTIF
CLR1    WTMK
```

INTWT割り込み要求クリア

INTWT割り込み要求クリア

4. 4　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_Wt.c)

・時計用タイマの初期化を行います。

・各カウンタ（変数）を初期化します。

・カウンタ値更新，表示変更の関数を呼び出し続ける無限ループに入り，INTWT割り込みを待ちます。

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

        fn_InitTimer();                 /* 時計用タイマの初期化      */

        g_ucTimeSec     =       0;      秒カウンタを初期化します。
        g_ucTimeSec10   =       0;      10秒カウンタを初期化します。
        g_ucTimeMin     =       0;      分カウンタを初期化します。
        g_ucTimeHour    =       0;      時カウンタを初期化します。

        g_ucSecChg_Flg  =       0;      秒表示LED変更要求フラグを初期化します。
```

【コラム】2進数値の表記

2進数値を表記する場合，アセンブリ言語では2進数値の後ろに「B」または「Y」を，C言語では2進数値の前に「0b」または「0B」を付加してください。

```
}
```

【アセンブリ言語】 (Kx2_Wt.asm)

・各カウンタ（変数）を初期化します。

・カウンタ値更新，表示変更のサブルーチンを呼び出し続ける無限ループに入り，INTWT割り込みを待ちます。

```
;************************************************************************
;       メイン処理
;************************************************************************
        MOV     RTMSEC,         #0      秒カウンタを初期化します。
        MOV     RTMSEC10,       #0      10秒カウンタを初期化します。
        MOV     RTMMIN,         #0      分カウンタを初期化します。
        MOV     RTMHOUR,        #0      時カウンタを初期化します。

        CLR1    BSECCHG_FLG             秒表示LED変更要求フラグを初期化します。

        EI                              ;ベクタ割り込み許可

MMAIN:
        CALL    !STIME                  ;時計カウントサブルーチン呼び出し
        BR      MMAIN
```

4.5　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の定義について説明します。

【C言語】 (Kx2_Wt.c)

・秒カウンタ，10秒カウンタ，分カウンタ，時カウンタ，秒表示LED変更要求フラグ，7セグメントLED表示データ・テーブル（0～10までの表示データを格納）の定義をします。

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

static  unsigned  char   g_ucTimeSec10;

static  unsigned  char   g_ucTimeMin;

static  unsigned  char   g_ucTimeHour;

static  unsigned  char   g_ucSecChg_Flg;

const   unsigned  char   g_ucSegPattern[10] =
{0b00111111,0b00000110,0b01011011,0b01001111,0b01100110,
0b01101101,0b01111101,0b00100111,0b01111111,0b01101111};
```

秒カウンタを定義します。

10秒カウンタを定義します。

分カウンタを定義します。

時カウンタを定義します。

秒表示LED変更要求フラグを定義します。

7セグメントLED表示データ・テーブルを定義します。

【アセンブリ言語】 (Kx2_Wt.asm)

・秒カウンタ，10秒カウンタ，分カウンタ，時カウンタ，秒表示LED変更要求フラグを定義します。

・10秒カウンタに応じて必要な処理部を選択するためのテーブルを定義します。

※表示更新処理部を選択するためのテーブルを定義しているので，本項目には表示更新処理部のプログラムもわかりやすいように記載しています。

```
;=====================================================================
;          使用する変数の定義
;=====================================================================
DRAM            DSEG    SADDR
RTMSEC:         DS      1
RTMSEC10:       DS      1
RTMMIN:         DS      1
RTMHOUR:        DS      1

BRAM            BSEG
BSECCHG_FLG     DBIT
                                    •
                                    •
                                    •
```

秒カウンタを定義します。

10秒カウンタを定義します。

分カウンタを定義します。

時カウンタを定義します。

秒表示LED変更要求フラグを定義します。

```
                                        •
                                        •
                                        •
TADDRTBL:
        DW      SEC00
        DW      SEC10
        DW      SEC20
        DW      SEC30
        DW      SEC40
        DW      SEC50
SEC00:
        MOV     A,      RTMMIN          ;分カウンタ読み出し
        ROR     A,      1               ;分カウンタデータを4ビット右に回転
        ROR     A,      1               ;(BCDコードに補正している分カウンタ
        ROR     A,      1               ;データの10の位を読み出すため)
        ROR     A,      1
        CALL    !SGET7SEG               ;7セグLED表示データをゲット
        MOV     P1,     A               ;10分の桁表示&P17クリア(LED3消灯)
        MOV     A,      RTMMIN          ;分カウンタ読み出し
        CALL    !SGET7SEG               ;7セグLED表示データをゲット
        MOV     P2,     A               ;1分の桁表示
        AND     P0,     #11111100B      ;P00,01をクリア(LED1,2消灯)
        AND     P4,     #11111100B      ;P40,41をクリア(LED4,5消灯)
        RET
SGET7SEG:
        AND     A,      #0FH            ;分カウンタデータをマスク
        MOV     B,      A
        MOVW    HL,     #TSEGPATTERN    ;7セグメントLEDデータアドレス読み出し
        MOV     A,      [HL+B]          ;7セグメントLEDデータ変更
        RET
SEC10:
        SET1    P0.0                    ;秒LED1点灯(10秒)
        RET
SEC20:
        SET1    P0.1                    ;秒LED2点灯(20秒)
        RET
SEC30:
        SET1    P1.7                    ;秒LED3点灯(30秒)
        RET
SEC40:
        SET1    P4.0                    ;秒LED4点灯(40秒)
        RET
SEC50:
        SET1    P4.1                    ;秒LED5点灯(50秒)
        RET
TSEGPATTERN:
        DB      00111111B               ;0
        DB      00000110B               ;1
        DB      01011011B               ;2
        DB      01001111B               ;3
        DB      01100110B               ;4
        DB      01101101B               ;5
        DB      01111101B               ;6
        DB      00100111B               ;7
        DB      01111111B               ;8
        DB      01101111B               ;9
```

10秒ごとの秒表示LEDや7セグメントLED表示更新処理は，処理部を分けてあります。ここには，10秒カウンタの値に応じて必要な処理部を選択するためのテーブルを定義しています。

10分桁の7セグメントLEDの表示更新を行います。

1分桁の7セグメントLEDの表示更新を行います。

7セグメントLED表示処理を行います。
（SEC00内で呼び出されるサブルーチン処理）

各秒LEDの点灯処理を行います。

7セグメントLED表示データ・テーブルを定義します。

4. 6　割り込み処理

0.5秒間隔のINTWT割り込みで起動され，秒カウンタをインクリメントし，10秒ごとにLED表示変更要求を出します。

【C言語】 (Kx2_Wt.c)

```
/************************************************************************
;       割り込み処理INTWT(0.5s)
;************************************************************************/
__interrupt     void    fn_IntWt(void){

        g_ucTimeSec++;                  /* 秒カウンタをインクリメント*/
```

0.5秒ごとに割り込みが入るので，秒カウンタが20回になったら，LED表示変更要求を出します。

```
        if(g_ucTimeSec == 20){          /* 10秒経過なら,秒LED表示変更要求*/

                g_ucSecChg_Flg  =       1;
                                        /* 秒LED表示変更要          */
                g_ucTimeSec             =       0;
                                        /* 秒カウンタをクリア       */
                g_ucTimeSec10++;        /* 10秒カウンタをインクリメント*/
        }
}
```

【アセンブリ言語】 (Kx2_Wt.asm)

```
;************************************************************************
;       割り込み処理INTWT(0.5s)
;************************************************************************
IINTWT:
```

割り込みにおけるレジスタ･バンクは1を用い,レジスタ破壊を防ぎます。

```
        SEL     RB1                     ;レジスタバンクを1に切り替える

        INC     RTMSEC                  ;秒カウンタをインクリメント
```

0.5秒ごとに割り込みが入るので，秒カウンタが20回になったら，LED表示変更要求を出します。

```
        MOV     A,      #20
```

Aレジスタ（20）から秒カウンタ（RTMSEC）の値を引いて0ならば，LED表示変更要求を出します。

```
        SUB     A,      RTMSEC          ;10秒経過の確認
        BNZ     $HSECCHG_END            ;10秒経過なら,秒LED表示変更要求

        SET1    BSECCHG_FLG             ;秒LED表示変更要求
        MOV     RTMSEC, #0              ;秒カウンタをクリア
        INC     RTMSEC10                ;10秒カウンタをインクリメント
HSECCHG_END:
        RETI                            ;割り込みから復帰
```

4.7　表示変更処理

メイン処理の無限ループの中から呼び出されて，LED表示変更要求がセットされていれば，10秒以上の桁の更新と表示更新処理を行います。この処理でも10秒カウンタの値を操作しているので，割り込み処理との競合を避けるために割り込み禁止にすることで排他制御を行っています。ただし，できるだけ割り込み禁止期間が短くなるように配慮しています。

【C言語】（表示変更処理関数）

```
/************************************************************************
;       時計カウント関数
;************************************************************************/
      unsigned char g_ucSecwork;          /* 10秒カウンタ作業用       */

void  fn_Time(void){
      DI();                               /* 割り込み禁止             */
      if(g_ucSecChg_Flg == 1){            /* 秒LED表示変更要求ありならば処理実行*/
              g_ucSecChg_Flg              =       0;
                                          /* 秒LED表示変更要求フラグクリア      */
              g_ucSecwork = g_ucTimeSec10;

              if(g_ucSecwork >= 6){       /* 60秒経過ならば処理実行   */
                   g_ucTimeSec10          -=      6;
                                          /* 10秒カウンタリセット     */
                   EI();                  /* 割り込み許可             */
                   g_ucTimeMin++;         /* 分カウンタをインクリメント */

                   if(g_ucTimeMin == 60){ /* 60分経過ならば処理実行   */
                          g_ucTimeMin             =       0;
                                          /* 分カウンタをリセット     */
                          g_ucTimeHour++;
                                          /* 時カウンタをインクリメント*/

                          if(g_ucTimeHour         == 24){
                                          /* 24時間経過ならば処理実行*/
                          g_ucTimeHour    =       0;
                                          /* 時カウンタをリセット     */
                          }
                   }
              }
              EI();                       /* 割り込み許可             */
                                          •
                                          •
                                          •
```

この関数処理内でINTWT割り込みに入ることを防ぐため，関数処理の頭でDI（割り込み禁止）をします。

秒，分，時カウンタの更新を行います。

本サンプル・プログラムでは，時表示は行いませんが，本プログラムから応用し，時表示も容易実現可能とするため，時カウンタの処理も行います。

```
                                        •
                                        •
                                        •
        switch(g_ucSecwork){

                        case 1:         /* 10秒カウンタが1である場合   */
                                P0.0    =       1;
                                        /* LED1を点灯                  */
                                break;
                        case 2:         /* 10秒カウンタが2である場合   */
                                P0.1    =       1;
                                        /* LED2を点灯                  */
                                break;
                        case 3:         /* 10秒カウンタが3である場合   */
                                P1.7    =       1;
                                        /* LED3を点灯                  */
                                break;
                        case 4:         /* 10秒カウンタが4である場合   */
                                P4.0    =       1;
                                        /* LED4を点灯                  */
                                break;
                        case 5:         /* 10秒カウンタが5である場合   */
                                P4.1    =       1;
                                        /* LED5を点灯                  */
                                break;

                        default:        /* 10秒カウンタが1～5以外の場合*/
                                P0      &=      0b11111100;
                                        /* P00,P01クリア(LED1,2消灯)*/
                                P4      &=      0b11111100;
                                        /* P40,P41クリア(LED4,5消灯)*/
                                P2      =       g_ucSegPattern[g_ucTimeMin%10];
                                        /* 1分の桁を表示               */
                                P1      =       g_ucSegPattern[g_ucTimeMin/10];
                                        /* 10分の桁表示&P17クリア(LED3消灯)*/
                }
        }
                EI();                   /* 割り込み許可                */

}
```

秒表示LEDの更新処理を行います。（10秒カウンタの値を読み出し，LEDをカウンタの値に対応するLEDを点灯させます。）

7セグメントLEDの表示更新処理を行います。（10秒カウンタの値を読み出し，1～5以外であったら，分カウンタの値に応じて7セグメントLEDの表示更新処理を行います。）

【アセンブリ言語】（表示変更処理サブルーチン）

```
;*********************************************************************
;       時計カウントサブルーチン                                    ********
STIME:
        DI                              ;ベクタ割り込み禁止
        BTCLR   BSECCHG_FLG,   $JTIME_SEC
                                        ;秒LED表示変更要求ありならば分岐
        EI                              ;ベクタ割り込み許可
        RET                             ;サブルーチンから復帰
                        秒カウント判定
JTIME_SEC:
        MOV     A,      RTMSEC10        ;10秒カウンタ値読み出し
        EI                              ;ベクタ割り込み許可
        MOV     X,      A               ;作業用にコピー
        CMP     A,      #6              ;60秒経過?
        BC      $JDISP                  ;Noならば表示処理分岐
        SUB     RTMSEC10,#6             ;10秒カウンタをリセット
                        分カウント判定
JTIME_MIN:
        MOV     A,      RTMMIN
        ADD     A,      #1              ;分カウンタを+1する(+1分)
        ADJBA                           ;BCD補正
        CMP     A,      #60H            ;60分経過?
        BNC     $JTIME_HOUR             ;Yesならば分岐
        MOV     RTMMIN,A                ;分カウンタを更新
        BR      JUPDT_SEC
                        時カウント判定
JTIME_HOUR:
        MOV     RTMMIN,#0               ;分カウンタをリセット
        MOV     A,      RTMHOUR
        ADD     A,      #1              ;時カウンタを+1する(+1時間)
        ADJBA                           ;BCD補正
        CMP     A,      #24H            ;24時間経過?
        BC      $JTIME_HOUR_UPDT        ;Noならば分岐
        MOV     A,      #0
JTIME_HOUR_UPDT:
        MOV     RTMHOUR,A               ;時間カウンタを更新
                                        •
                                        •
                                        •
```

割り込み処理とサブルーチンでの処理で同じ変数を使用しているので，その変数が競合しないように排他制御を行う必要があります。そこで，このサブルーチン処理内では，割り込み処理と共有している変数を使用する時に，DIすることで割り込み禁止して競合を防ぎます。

秒，分，時カウンタの更新を行います。

本サンプル・プログラムでは，時表示は行いませんが，本プログラムから応用し，時表示も容易実現可能とするため，時カウンタの処理も行います。

```
                                    •
                                    •
                                    •
JUPDT_SEC:
      MOV     A,      X             ;10秒カウンタ値を復帰

JDISP:
      ADD     A,      A             ;
      ADD     A,      #LOW TADDRTBL ;アドレステーブル下位計算
      MOV     L,      A
      MOV     A,      #HIGH TADDRTBL ;アドレステーブル上位
      ADDC    A,      #0
      MOV     H,      A
      MOV     A,      [HL]          ;下位アドレス読み出し
      MOV     X,      A
      MOV     A,      [HL+1]        ;上位アドレス読み出し
      BR      AX                    ;処理アドレスに分岐
```

アドレス・テーブルの下位を更新するため，10秒カウンタを倍にします。

秒カウンタからTADDRTBLで定義した（“4.5 変数・定数の定義”参照）アドレスを算出し，そのアドレスに応じた処理を実行します。

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。

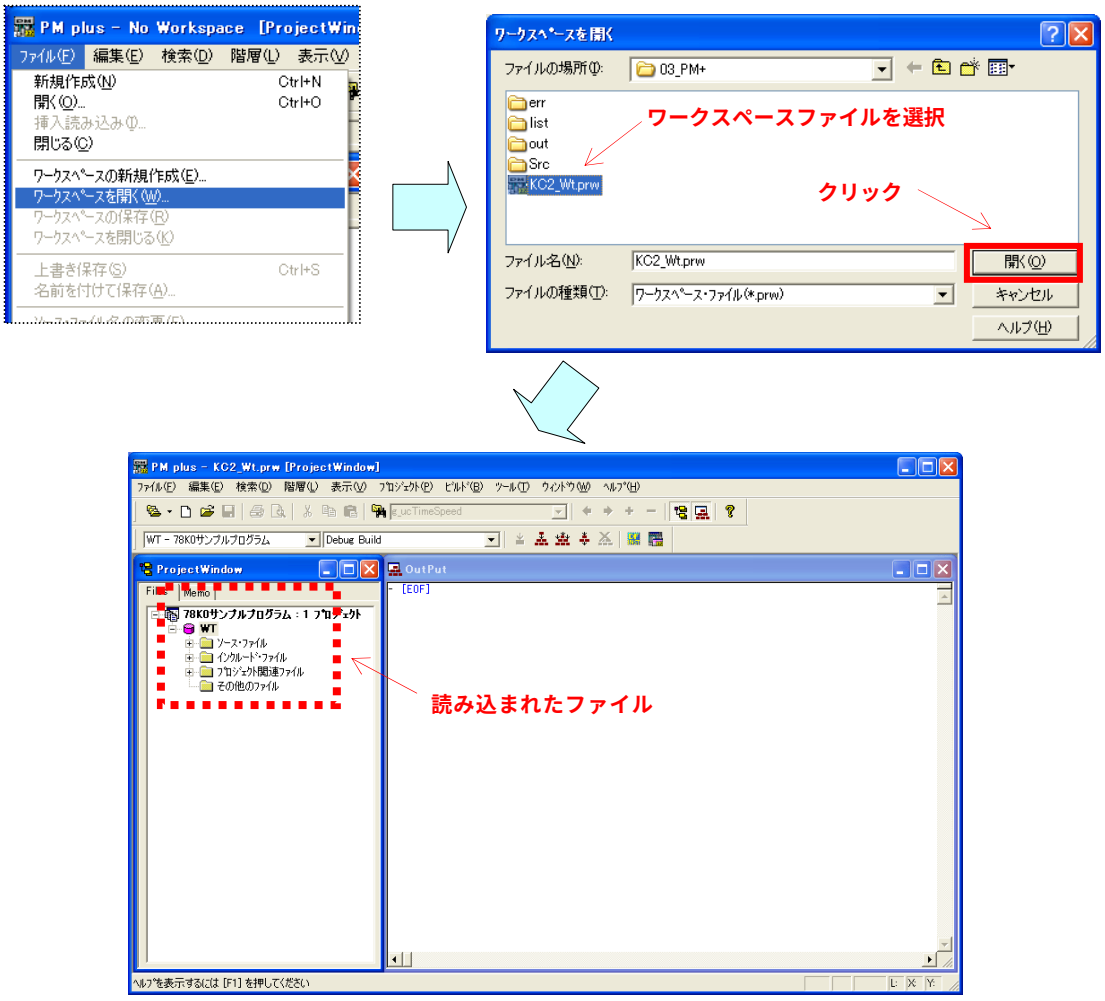
PM+操作方法の詳細については，PM+ プロジェクト・マネージャ　ユーザーズ・マニュアルを参照してください。

(1) PM+を起動してください。

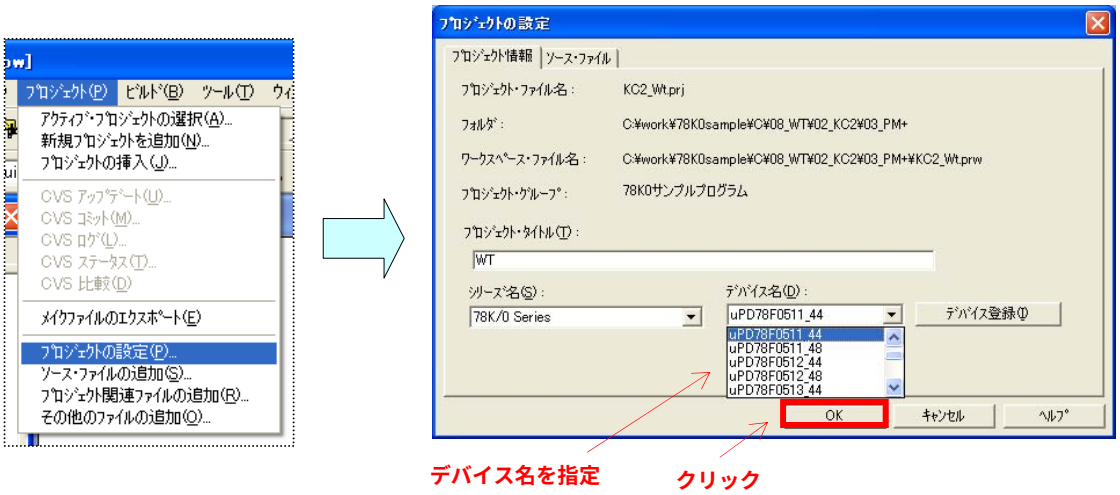
(2) [ファイル] → [ワークスペースを開く] から，「Kx2_Wt.prw」[注]を選択し，[開く] ボタンをクリックしてください。ワークスペースが作成され，その中にソース・ファイルが自動的に読み込まれます。

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/KC2の場合「KC2_Wt.prw」

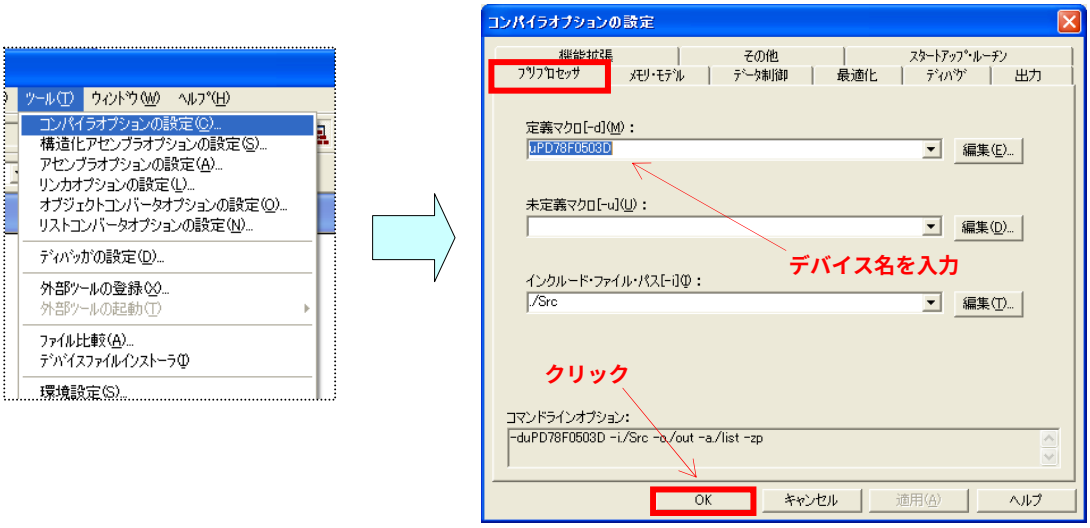


(3) [プロジェクト] → [プロジェクトの設定] を選択してください。[プロジェクトの設定] 画面が表示されたら，使用するデバイス名を選択（デフォルトでは，ROM/RAMサイズの最も大きいデバイスが選択）し，[OK] ボタンをクリックしてください。



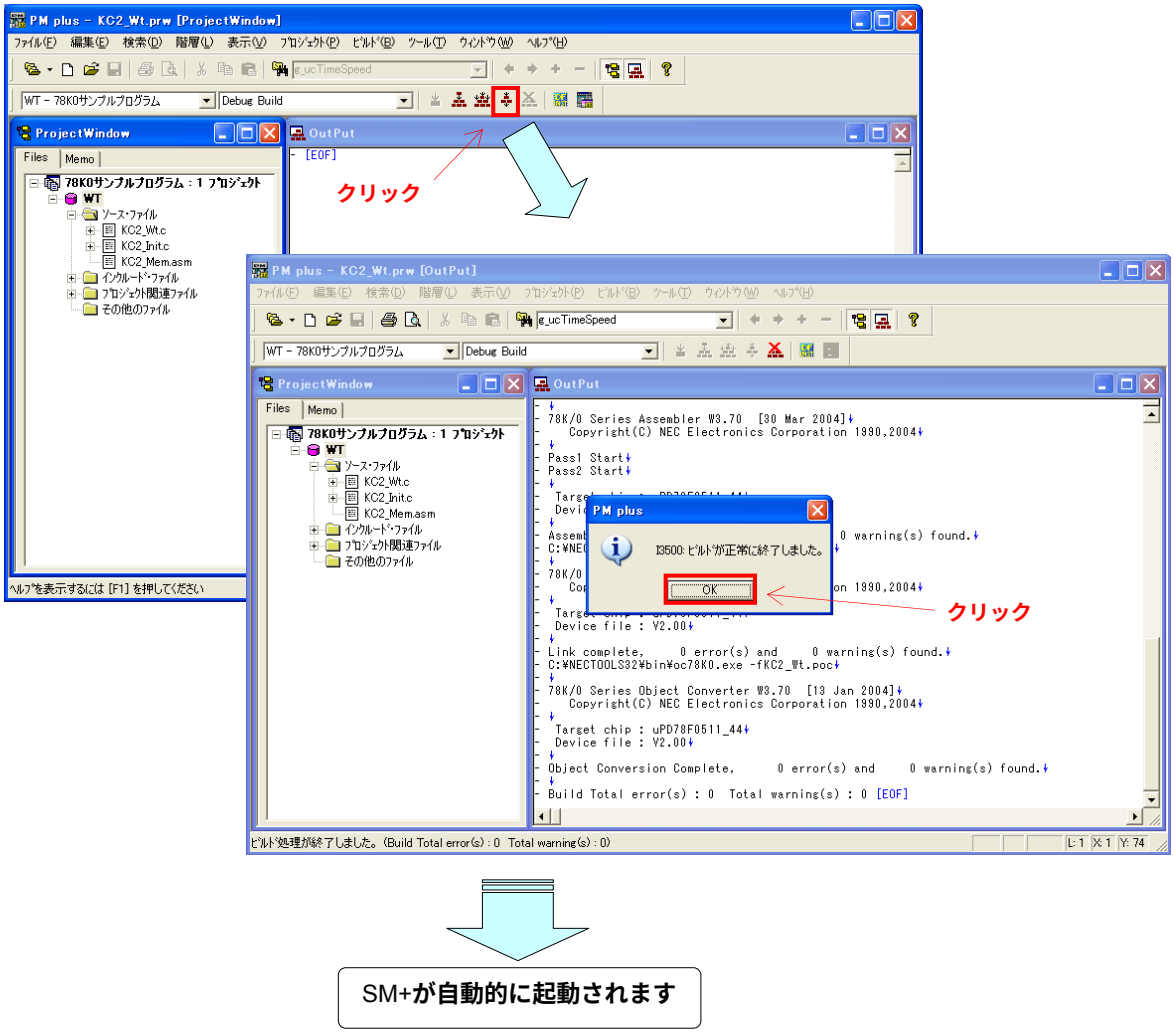
※μPD78F0500_36, μPD78F0501_36, μPD78F0502_36, μPD78F0503_36は選択しないでください。

(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/KC2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_Wt.c」と「Kx2_Init.c」と「Kx2_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。

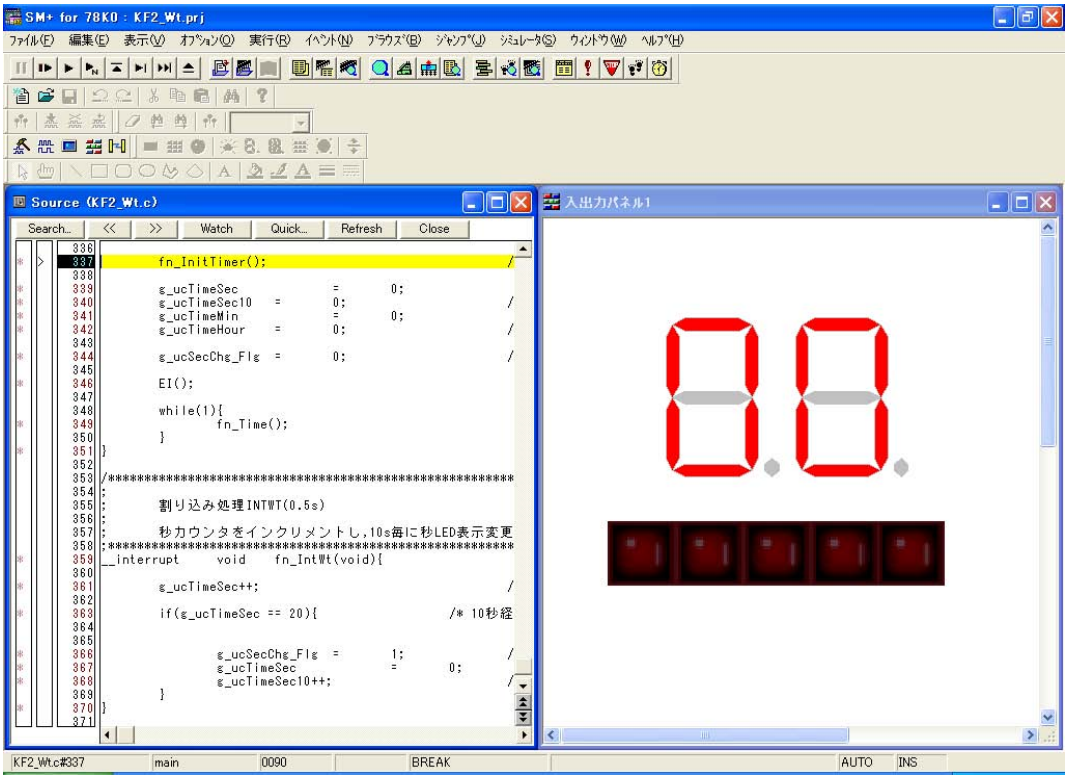


5. 2　SM+での動作

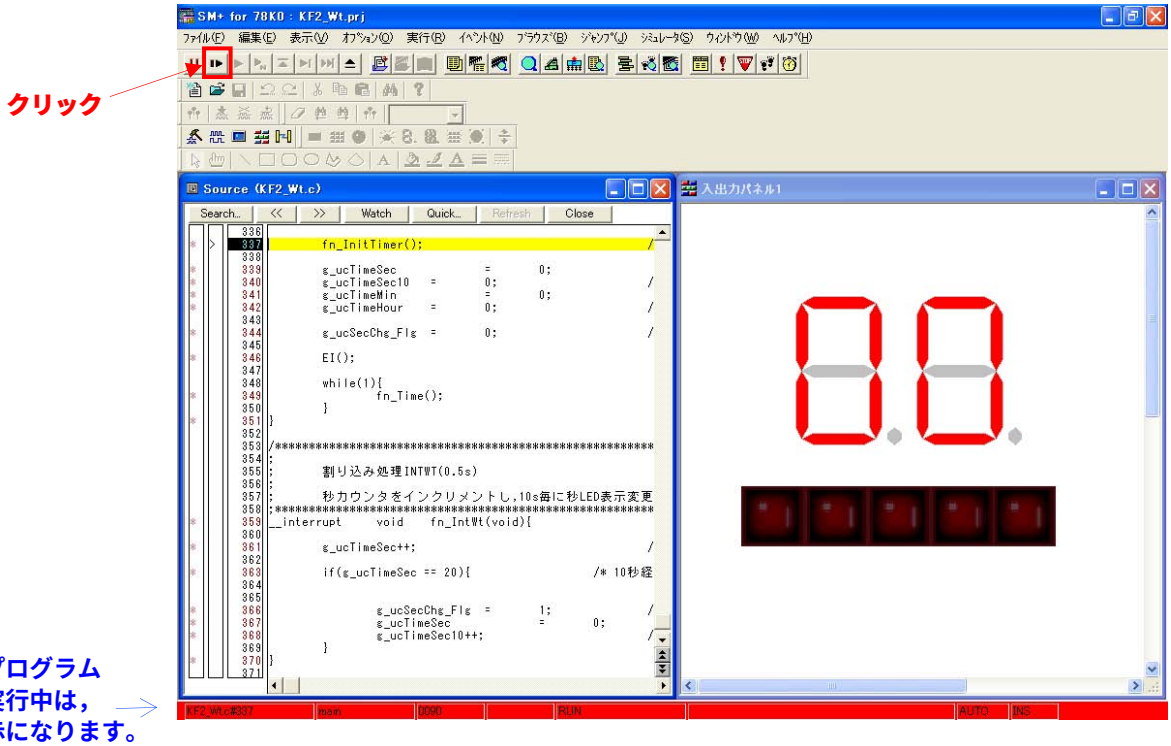
ここでは，SM+の入出力パネル・ウインドウやタイミング・チャート・ウインドウ上での動作確認の例を説明します。

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。

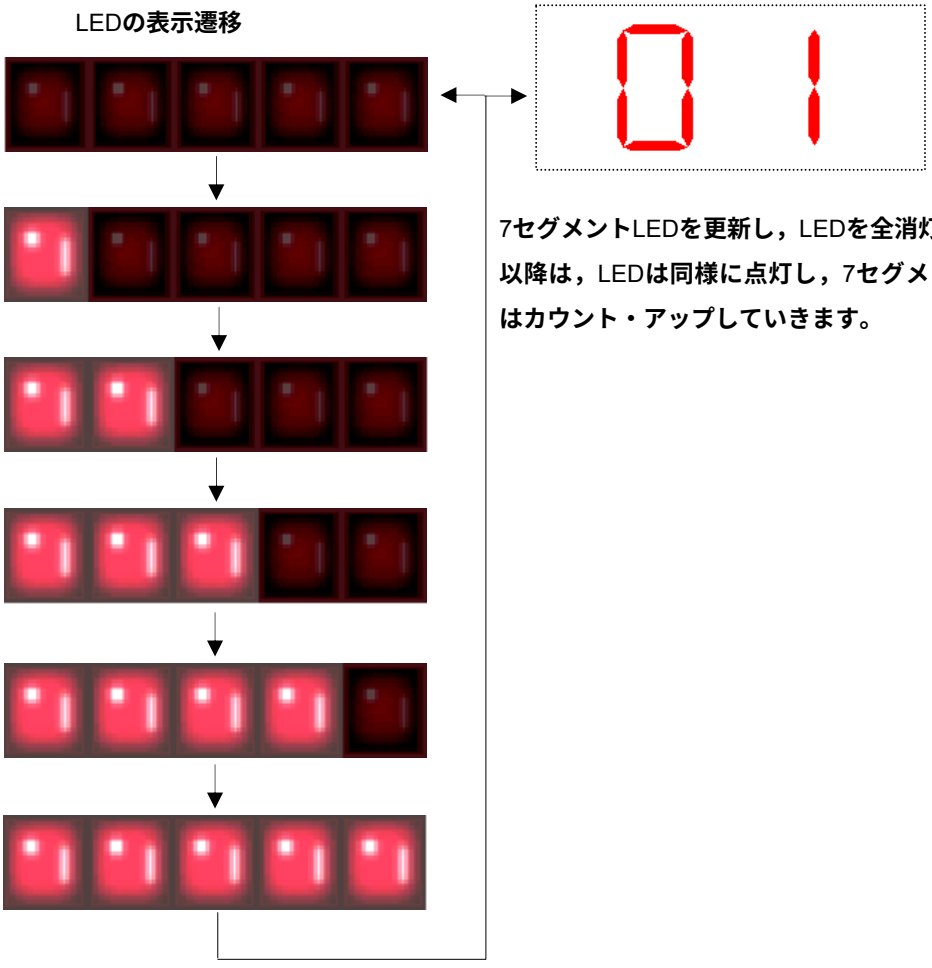
（1）PM+の「ビルド→ディバグ」からSM+を起動（5. 1を参照）すると，次のような画面になります。



(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)　10秒おきにLEDが順次点灯して，1分後に7セグメントLEDの数値が更新されます。



7セグメントLEDを更新し，LEDを全消灯します。
以降は，LEDは同様に点灯し，7セグメントLEDはカウント・アップしていきます。

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
```

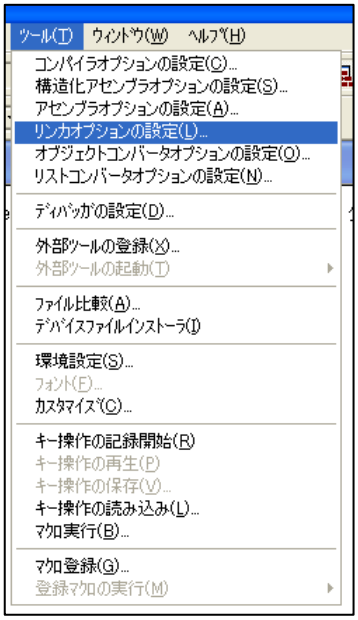
0002H番地はオンチップ・デバッグ使用領域として空けるため，0002H番地はコメント・アウトして，CSEGで再び0004H番地を定義しています。

※C言語版では不要です。

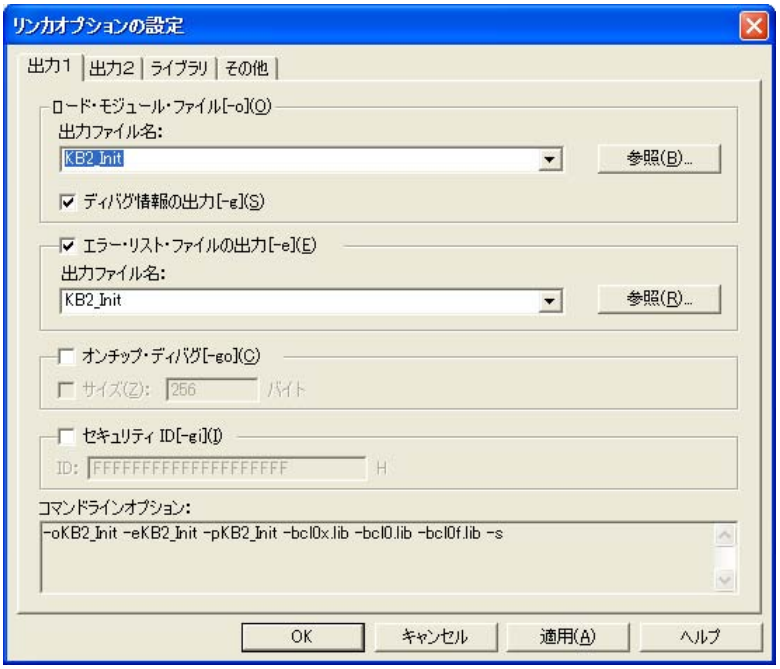
(3) リンカの設定

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。

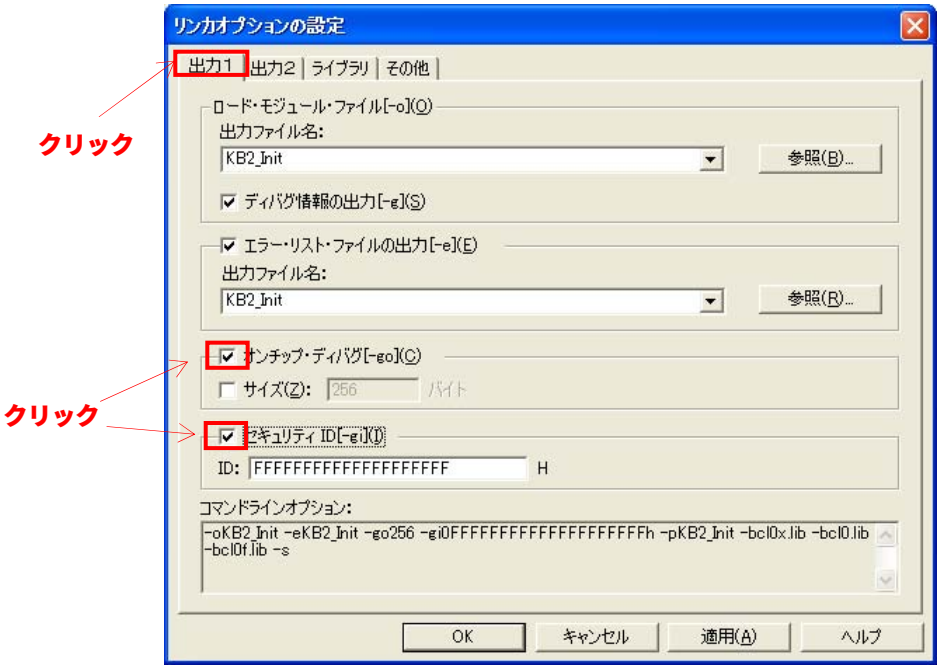
PM+の「ツール」メニューから「リンカオプションの設定」を選択してください。



「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