

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ひ申し上げます。

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のですが、誤りが無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、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と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。

アプリケーション・ノート

78K0/Kx2

サンプル・プログラム

マイコン初期化編

この資料は、サンプル・プログラムの「初期設定」の動作概要と、マイコンの基本的な初期設定を説明したものです。サンプル・プログラムでは、クロック周波数の選択や入出力ポートの選択など、マイコンの基本的な初期設定を行ったあとに、2つのスイッチ入力と3つのLED点灯の制御を行います。

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

- 第1章 概要 ... 3
- 第2章 回路イメージ ... 4
 - 2.1 回路イメージ ... 4
 - 2.2 マイコン以外の使用デバイス ... 4
- 第3章 ソフトウェアについて ... 5
 - 3.1 ファイル構成 ... 5
 - 3.2 使用するマイコン内蔵周辺機能 ... 7
 - 3.3 初期設定と動作概要 ... 7
 - 3.4 フロー・チャート ... 8
- 第4章 設定方法について ... 9
 - 4.1 オプション・バイトの設定 ... 9
 - 4.2 ベクタ・テーブルの設定 ... 14
 - 4.3 スタック・ポインタの設定 ... 16
 - 4.4 ウォッチドッグ・タイマの設定と制御 ... 17
 - 4.5 クロックの設定 ... 18
 - 4.6 ポートの設定 ... 24
 - 4.7 メイン処理 ... 27
- 第5章 システム・シミュレータ SM+での動作確認 ... 29
 - 5.1 サンプル・プログラムのビルド ... 29
 - 5.2 SM+での動作 ... 32
 - 5.3 オンチップ・デバッグ時の注意 ... 36
 - 5.4 開発環境のダウンロード、インストール ... 39
- 第6章 関連資料 ... 40
- 付録A 改版履歴 ... 41

- 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いません。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に、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、オプション・バイトの設定、クロック周波数の選択、ポート入出力の設定など、78K0/Kx2マイクロコントローラの基本的な初期設定を行います。また、初期化完了後のメイン処理動作では、2つのスイッチ入力により、3つのLED点灯を制御します。

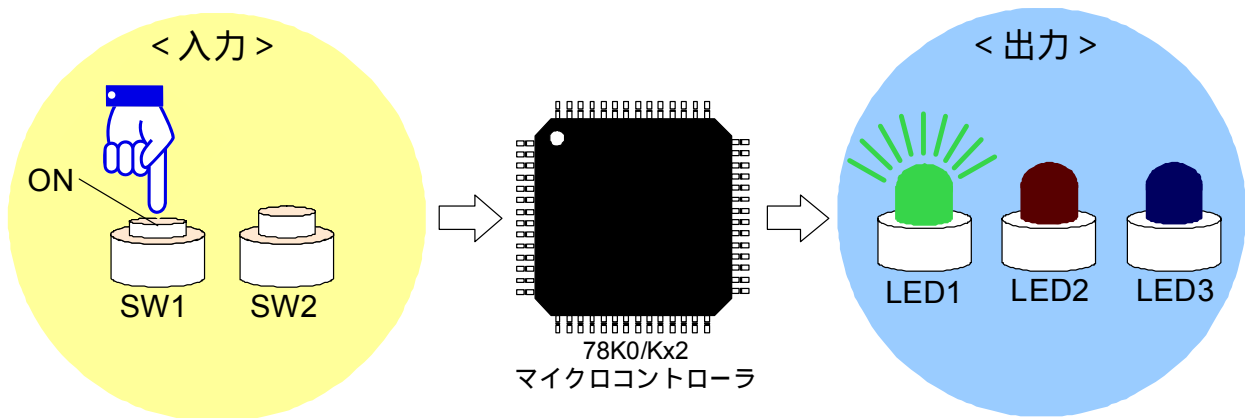
(1) 初期設定の主な内容

- ・オプション・バイトの参照 (2.7 [V] POCモードを使用, ウォッチドッグ・タイマは動作停止に設定)
- ・メモリ・サイズの設定
- ・CPUクロックの設定 (発振回路関係の端子はすべてポートとして使用に設定)
- ・発振クロックの設定 (高速内蔵発振8 MHzを使用)
- ・低電圧検出の設定 (動作禁止)
- ・入出力ポートの設定
- ・タイマの設定 (動作停止)
- ・A/Dコンバータの設定 (動作停止)
- ・UARTの設定 (動作禁止)
- ・割り込みの設定

(2) メイン処理動作の内容

78K0/Kx2マイクロコントローラにて、スイッチ入力 (SW1, SW2) を検出し、LED点灯 (LED1, LED2, LED3) を制御します。

本サンプル・プログラムではスイッチ入力時のチャタリング除去は行っておりません。



スイッチ入力		LED出力		
SW1	SW2	LED1	LED2	LED3
OFF	OFF	OFF	OFF	OFF
ON	OFF	ON	OFF	OFF
OFF	ON	OFF	ON	OFF
ON	ON	OFF	OFF	ON

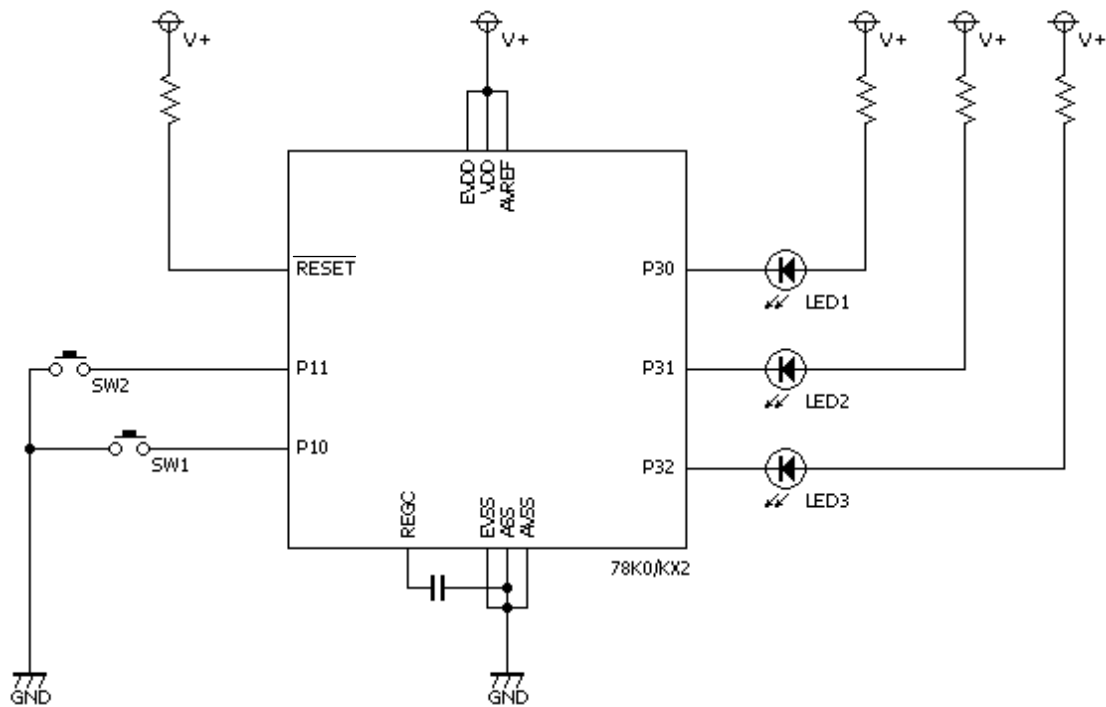
注意 デバイス使用上の注意事項については、[78K0/Kx2 ユーザーズ・マニュアル](#)を参照してください。

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ェアを説明します。

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 μ F) 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) スイッチ (SW1, SW2)

LED点灯制御用の入力として、スイッチを使用します。

(2) LED (LED1, LED2, LED3)

スイッチ入力に対応した出力として、LEDを使用します。




第3章 ソフトウェアについて

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・プログラムの初期設定と動作概要，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_Init.c	マイコンのハードウェア初期化処理とメイン処理のソース・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_Init.prw	統合開発環境 PM+用ワーク・スペース・ファイル			
Kx2_Init.prj	統合開発環境 PM+用プロジェクト・ファイル			
Kx2_Init.pri Kx2_Init.prs Kx2_Init.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_Init0.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 (周辺ハードウェ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_Init.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_init.asm	マイコンのハードウェア初期化処理とメイン処理のソース・ファイル			
Kx2_op.asm	オプション・バイト設定値定義用アセンブラ・インクルード・ファイル			
Kx2_init.prw	統合開発環境 PM+用ワーク・スペース・ファイル			
Kx2_init.prj	統合開発環境 PM+用プロジェクト・ファイル			
Kx2_init.pri Kx2_init.prs Kx2_init.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_init0.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 (周辺ハードウェ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_init.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。

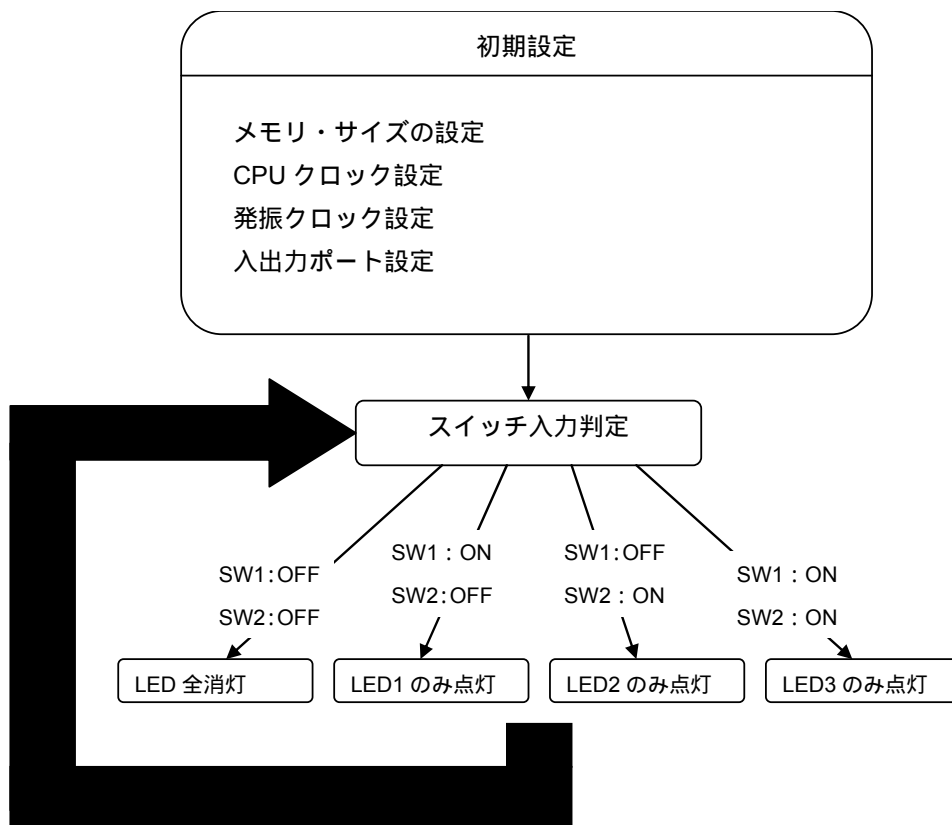
- ・入力ポート（スイッチ入力用） : P10, P11
- ・出力ポート（LED点灯用） : P30, P31, P32

3.3 初期設定と動作概要

このサンプル・プログラムでは、初期設定にて、クロック周波数の選択や、入出力ポートの設定などを行います。

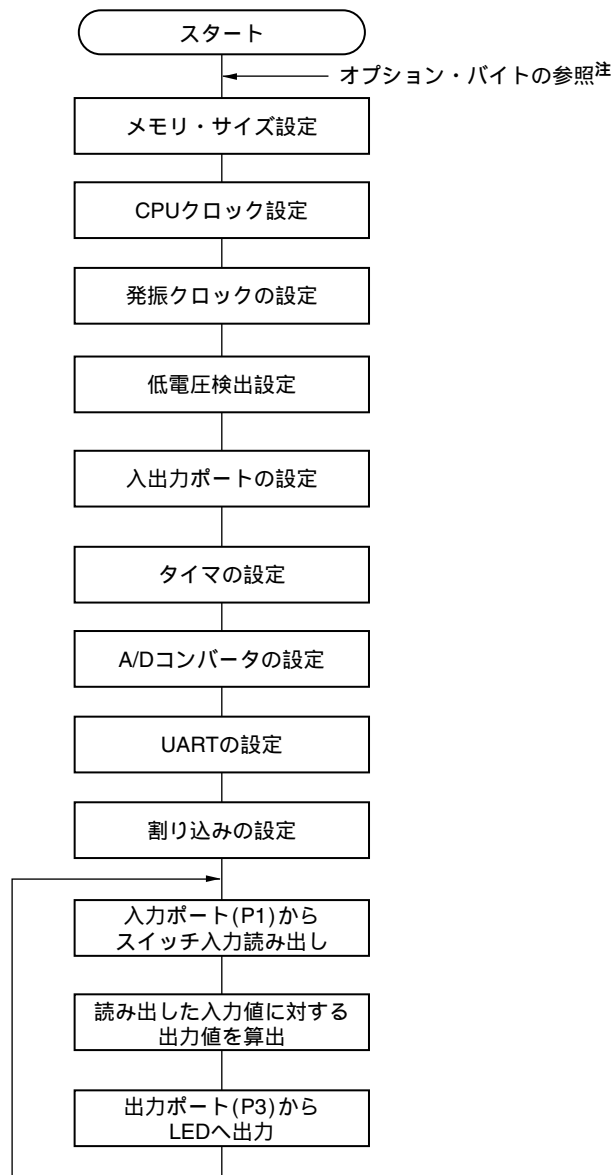
初期設定完了後は、2つのスイッチ入力（SW1, SW2）の組み合わせに応じて、3つのLED（LED1, LED2, LED3）の点灯を制御します。

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。



注 オプション・バイトは別ファイル (Kx2_op.asm) で設定しており、ハードウェアにて自動的に参照します。
このサンプル・プログラムでは、オプション・バイトの参照により、次のような内容が設定されます。

- ・低速内蔵発振器の設定
- ・ウォッチドッグ・タイマの設定
- ・POCモードの設定
- ・オンチップ・デバッグの設定

第4章 設定方法について

この章では、オプション・バイト、ベクタ・テーブル、スタック・ポインタ、ウォッチドッグ・タイマ、クロック周波数、ポートの設定、およびメイン処理について説明します。

C言語によるプログラムを実行させるには、システムへ組み込むためのROM化処理、ユーザー・プログラム(main 関数)の起動などを行うプログラムが必要となります。このプログラムのことをスタート・アップ・ルーチンと呼びます。

このスタート・アップ・ルーチンは一般的に、マイコンがリセット(初期化)後、最初に動くプログラムで、CPU、メモリ、周辺I/Oなどのハードウェア初期設定や、メイン処理ルーチンを動かすための初期設定を行います。プログラムは基本的に、このスタート・アップ・ルーチンから始まり、その次にメイン・ルーチン処理が実行され、その後サブルーチン処理や割り込み処理と続いて実行されます。

本サンプル・プログラムのC言語版では、hdwinit関数によりクロック関連の設定や周辺ハードウェアの初期設定などを行っており、hdwinit関数を実行した後にmain関数を実行するのでメイン処理はmain関数に記述します。アセンブリ言語版ではリセット(初期化)後、ベクタ・テーブルの0000H番地に書かれたIRESET番地からプログラムが実行され、C言語版のhdwinit関数と同じように、クロック関連の設定や周辺ハードウェアの初期設定などを行い、メイン処理に入ります。

なお、本プログラムでは、使用しないタイマなどの周辺ハードウェア設定は動作停止としています。使用時には各用途、機能に合わせて各レジスタを設定してください。

スタート・アップ・ルーチンの詳細については、[CC78K0 操作編 ユーザーズ・マニュアル](#) スタート・アップ・ルーチンの章を参照してください。

レジスタ設定の詳細については、[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、[78K0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 オプション・バイトの設定

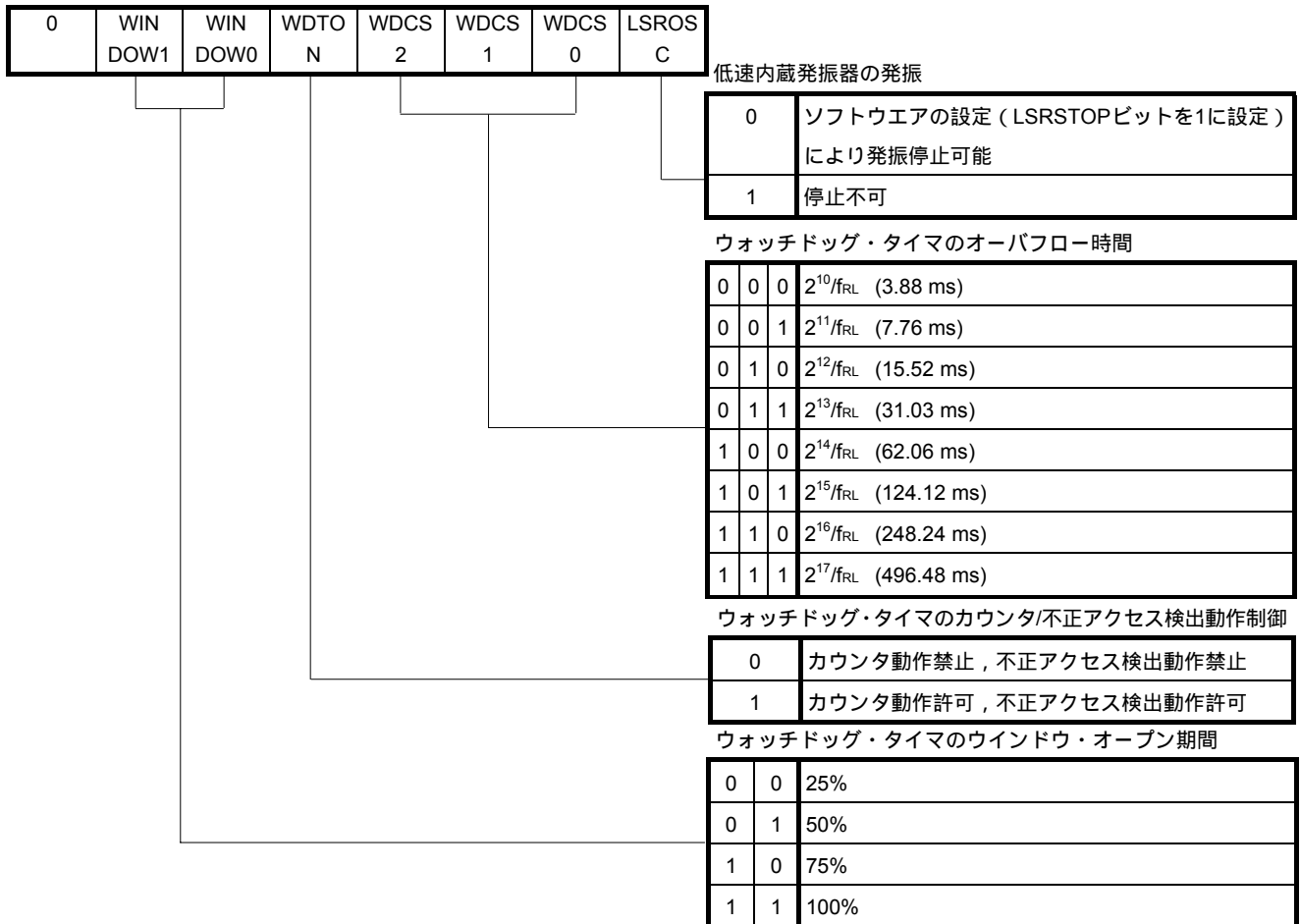
オプション・バイトの設定は、必須です。オプション・バイトで、次の項目を設定します。

- (1) 低速内蔵発振器の発振制御
- (2) ウォッチドッグ・タイマのオーバフロー時間の設定
- (3) ウォッチドッグ・タイマのカウンタ動作設定
- (4) ウォッチドッグ・タイマのウィンドウ・オープン期間の設定
- (5) POCモードの選択
- (6) オンチップ・デバッグ動作制御

本サンプル・プログラムでは、後述の【例 1】の内容で、オプション・バイトを設定しています。

図4 - 1 オプション・バイトのフォーマット(1/2)

アドレス：0080H/1080H^注



注 ブート・スワップ時は、0080Hと1080Hが切り替わるので、あらかじめ1080Hにも0080Hと同じ値を設定してください。

注意1. ビット7には必ず0を書き込んでください。

2. ウォッチドッグ・タイマのカウンタを動作許可としている場合、WDCS2 = WDCS1 = WDCS0 = 0かつ WINDOW1 = WINDOW0 = 0の組み合わせは設定禁止です。

図4 - 1 オプション・バイトのフォーマット(2/2)

アドレス：0081H/1081H^注

0	0	0	0	0	0	0	POCM ODE
---	---	---	---	---	---	---	-------------

POCモードの選択

0	1.59 V POCモード (デフォルト)
1	2.7 V/1.59 V POCモード

注 POCMODEは、専用フラッシュ・ライターによる書き込みのみ設定可能です。セルフ・プログラミング、および、セルフ・プログラミング中のブート・スワップ動作では設定できません。ただし、ブート・スワップ動作時に1081Hの値は0081Hに切り替わるので、ブート・スワップ使用時は、1081Hに0081Hと同じ値を設定しておくことを推奨します。

注意 ビット7-1には必ず0を書き込んでください。

アドレス：0082H/1082H, 0083H/1083H^注

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

注 0082H, 0083Hは予約領域なので必ず0を設定してください。また、ブート・スワップ時は、0082H, 0083Hと、1082H, 1083Hが切り替わるので、あらかじめ1082H, 1083Hにも0を設定してください。

アドレス：0084H/1084H^注

0	0	0	0	0	0	OCDE N1	OCDE N0
---	---	---	---	---	---	------------	------------

オンチップ・デバッグ動作制御

0	0	動作禁止
0	1	設定禁止
1	0	動作許可。オンチップ・デバッグ・セキュリティID認証失敗時にフラッシュ・メモリのデータを消去しない
1	1	動作許可。オンチップ・デバッグ・セキュリティID認証失敗時にフラッシュ・メモリのデータを消去する

注 オンチップ・デバッグ機能を搭載していない製品は、必ず0084Hに00H（オンチップ・デバッグ動作禁止）を設定してください。オンチップ・デバッグ機能を搭載している製品で、オンチップ・デバッグ機能を使用する場合は、0084Hに02Hまたは03Hを設定してください。また、ブート・スワップ時は、0084Hと1084Hが切り替わるので、あらかじめ1084Hに0084Hと同じ値を設定しておいてください。

オンチップ・デバッグ機能搭載 / 非搭載の製品については、次の表を参照してください。

デバイス	78K0/KB2	78K0/KC2	78K0/KD2	78K0/KE2	78K0/KF2
オンチップ・デバッグ機能搭載	μPD78F0503D	μPD78F0513D μPD78F0515D	μPD78F0527D	μPD78F0537D	μPD78F0547D
オンチップ・デバッグ機能非搭載	μPD78F0500 μPD78F0501 μPD78F0502 μPD78F0503	μPD78F0511 μPD78F0512 μPD78F0513 μPD78F0514 μPD78F0515	μPD78F0521 μPD78F0522 μPD78F0523 μPD78F0524 μPD78F0525 μPD78F0526 μPD78F0527	μPD78F0531 μPD78F0532 μPD78F0533 μPD78F0534 μPD78F0535 μPD78F0536 μPD78F0537	μPD78F0544 μPD78F0545 μPD78F0546 μPD78F0547

オプション・バイト設定例

【例 1】 低速内蔵発振器の停止可，オンチップ・デバッグ動作禁止

アドレス：0080H

0	x	x	x	x	x	x	0
---	---	---	---	---	---	---	---

低速内蔵発振器の発振制御^注

0	ソフトウェアの設定（LSRSTOPビットを1に設定）により，発振停止
---	------------------------------------

*このサンプル・プログラムでは，ウォッチドッグ・タイマは使用しません。そのため，ソフトウェアで発振停止できるように設定します。

アドレス：0081H

0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

POCモードの選択

1	2.7V POCモード
---	-------------

アドレス：0082H, 0083H

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

アドレス：0084H

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

オンチップ・デバッグ動作制御

0	0	動作禁止
---	---	------

注 低速内蔵発振器の発振停止やウォッチドッグ・タイマの動作の設定は，リセット解除後に行います。詳細については，[4.4 ウォッチドッグ・タイマの設定と制御](#)，[4.5 クロックの設定](#)を参照してください。

オプション・バイトの設定値は，上記（x: don't care）となります。ソフトウェアを記述すると，次のようになります（下記の例では「x」を0に設定）。

TOPTIONB	CSEG	AT	0080H
DB	00000000B		
DB	00000001B		
DB	00000000B		
DB	00000000B		
DB	00000000B		
DB	00000000B		
END			

C言語を使用する場合は，アセンブリ言語のオプション・バイト・ソース・ファイル（ファイル名：「*.asm（*：任意）」）を準備し，プロジェクトのソース・ファイルに指定して，他のソース・ファイル（main.c）と一緒にビルドすることでC言語でもオプション・バイトをアセンブリ言語記述で設定できます。

本サンプル・プログラムでは，この手法でアセンブリ言語記述のオプション・バイト・ソース・ファイルを全サンプル・プログラムで引用しています。

【例 2】 低速内蔵発振器の停止可，ウォッチドッグ・カウンタ動作許可，オンチップ・デバッグ動作制御許可

アドレス：0080H

0	0	1	1	1	0	0	0
---	---	---	---	---	---	---	---

低速内蔵発振器の発振制御

0	ソフトウェアの設定（LSRSTOPビットを1に設定）により，発振停止
---	------------------------------------

ウォッチドッグ・タイマのオーバーフロー時間

1	0	0	$2^{14}/f_{RL}$ (62.06 ms)
---	---	---	----------------------------

ウォッチドッグ・タイマのカウンタ / 不正アクセス検出動作制御

1	カウンタ動作許可。不正アクセス検出動作許可
---	-----------------------

ウォッチドッグ・タイマのウィンドウ・オープン期間

0	1	50%
---	---	-----

アドレス：0081H

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

POCモードの選択

0	1.59 V POCモード（デフォルト）
---	----------------------

アドレス：0082H, 0083H

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

アドレス：0084H

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

オンチップ・デバッグ動作制御

1	0	動作許可
---	---	------

ソフトウェアを記述すると，次のようになります。

```

TOPTIONB      CSEG   AT      0080H
               DB     00111000B
               DB     00000000B
               DB     00000000B
               DB     00000000B
               DB     00000000B
               DB     00000010B
               END
    
```



【コラム】 CSEG (Code Segment), DSEG (Data Segment), BSEG (Bit Segment) とは

CSEG, DSEG, BSEGは，命令やデータなどの生成されたコードをどこに配置するかを示す疑似命令です。これらの疑似命令以降に記述された命令やデータは，CSEG疑似命令ではROM領域に，DSEG疑似命令ではRAM領域に，BSEG疑似命令ではRAMのsaddr領域へ配置されます。

例えば，オプション・バイトの設定内容を内部ROM（フラッシュ・メモリ）の0080Hから配置する場合は，まず，CSEG疑似命令とAT属性を使用し，アドレスに0080Hを指定します。そのあとに，DB疑似命令で0080H番地以降に設定したい値を定義し，それをアセンブリ言語にて記述したプログラム内に記述してください。

CSEG疑似命令で指定したROM領域のみ，DB, DW疑似命令は使用可能です。DSEG, BSEG疑似命令で指定したRAM領域では，DB, DW疑似命令の記述はエラーにはなりませんが，使用しないでください。この場合，オブジェクトは生成され，MINICUBE2（オンチップ・デバッグ・エミュレータ）やSM+（システム・シミュレータ）では記述された命令やデータをRAM領域に展開するため，デバッグ動作を行うことができます。しかし，実際のデバイスでは，それらをRAM領域に展開できないため，動作できなくなります。

CSEG, DSEG, BSEG疑似命令の詳細については，[RA78K0 言語編 ユーザーズ・マニュアル](#)を参照してください。

4.2 ベクタ・テーブルの設定

ベクタ・テーブル領域は、リセットや各割り込み要求発生により分岐するときのプログラム・スタート・アドレスを格納する領域です。C言語で記述する場合は、スタートアップ・ルーチンにてリセット・ベクタが自動的に設定されるため、設定は不要です。

【例】 リセット・スタート時に使用するリセット・ベクタのみ設定（サンプル・プログラムの設定と同内容）

```

TVECTTBL      CSEG      AT      0000H      アドレス      機能名
;
; 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
TVECTTBL_TBL1 CSEG      AT      0004H
; DW      IINIT       ;0004H INTLVI
; DW      IINIT       ;0006H INTP0
; DW      IINIT       ;0008H INTP1
; DW      IINIT       ;000AH INTP2
; DW      IINIT       ;000CH INTP3
; DW      IINIT       ;000EH INTP4
; DW      IINIT       ;0010H INTP5
; DW      IINIT       ;0012H INTSRE6
; DW      IINIT       ;0014H INTSR6
; DW      IINIT       ;0016H INTST6
; DW      IINIT       ;0018H INTCSI10 / INTST0
; DW      IINIT       ;001AH INTTMH1
; DW      IINIT       ;001CH INTTMH0
; DW      IINIT       ;001EH INTTM50
; DW      IINIT       ;0020H INTTM000
; DW      IINIT       ;0022H INTTM010
; DW      IINIT       ;0024H INTAD
; DW      IINIT       ;0026H INTSR0
; DW      IINIT       ;0028H
; DW      IINIT       ;002AH INTTM51
; DW      IINIT       ;002CH
; DW      IINIT       ;002EH
; DW      IINIT       ;0030H
; DW      IINIT       ;0032H
; DW      IINIT       ;0034H INTIIC0 / INTDMU
; DW      IINIT       ;0036H
; DW      IINIT       ;0038H
; DW      IINIT       ;003AH
; DW      IINIT       ;003CH
; DW      IINIT       ;003EH BRK
;
;
; *****
;
;      不要な割り込みの処理
;      (ここでは何も処理をしないで戻る)
;
; *****
PUBLIC IINIT
IINIT:
;
;      必要な割り込み処理がある時にはここに記述
;
      RETI
    
```


リセット解除後、プログラムは、リセット・ベクタで指定したアドレス(前述の【例】では、の「IRESET」)からスタートします。

このサンプル・プログラムでは、ベクタ・テーブル・アドレスの0000H以外は使用しません。残りのベクタ・テーブル・アドレスには、すべて「IINIT」を設定しています。このように設定することにより、万が一、割り込みが発生した場合でも「IINIT」に分岐させ、不要な割り込みとして何も処理をさせずに割り込みから復帰させます。



【コラム】#pragma指令とは

#pragma指令は、C言語で使用する前処理命令で、ソース・プログラムの冒頭に記述します。

主な#pragma指令は、次のとおりです。

- ・ #pragma sfr : SFR領域に関する操作をCソース・レベルで記述可能
- ・ #pragma ei : EI命令をCソース・レベルで記述可能
- ・ #pragma di : DI命令をCソース・レベルで記述可能
- ・ #pragma nop : NOP命令をCソース・レベルで記述可能 (CPUを動作させずにクロックを進めることが可能)
- ・ #pragma interrupt : 割り込み関数をCソース・レベルで記述可能

4.4 ウォッチドッグ・タイマの設定と制御

ウォッチドッグ・タイマの動作クロックの設定とオーバフロー時間の設定はオプション・バイトで設定します。詳細は[4.1 オプション・バイトの設定](#)を参照してください。

ウォッチドッグ・タイマ・イネーブル・レジスタ (WDTE) にACHを書き込むことにより、ウォッチドッグ・タイマのカウンタをクリアし、再びカウントを開始します。リセット信号により、WDTEは9AH、または1AH[※]になります。

注 オプション・バイト (0080H) のWDTONの設定値によって、WDTEのリセット値が異なります。

WDTONの設定	WDTEのリセット値
0 (ウォッチドッグ・タイマのカウント動作禁止)	1AH
1 (ウォッチドッグ・タイマのカウント動作許可)	9AH

注意 WDTEにACH以外の値を書き込んだ場合、内部リセット信号を発生します。ただし、ウォッチドッグ・タイマのソース・クロックが停止している場合は、ウォッチドッグ・タイマのソース・クロックが再び動作開始した時点で、内部リセット信号を発生します。



【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。

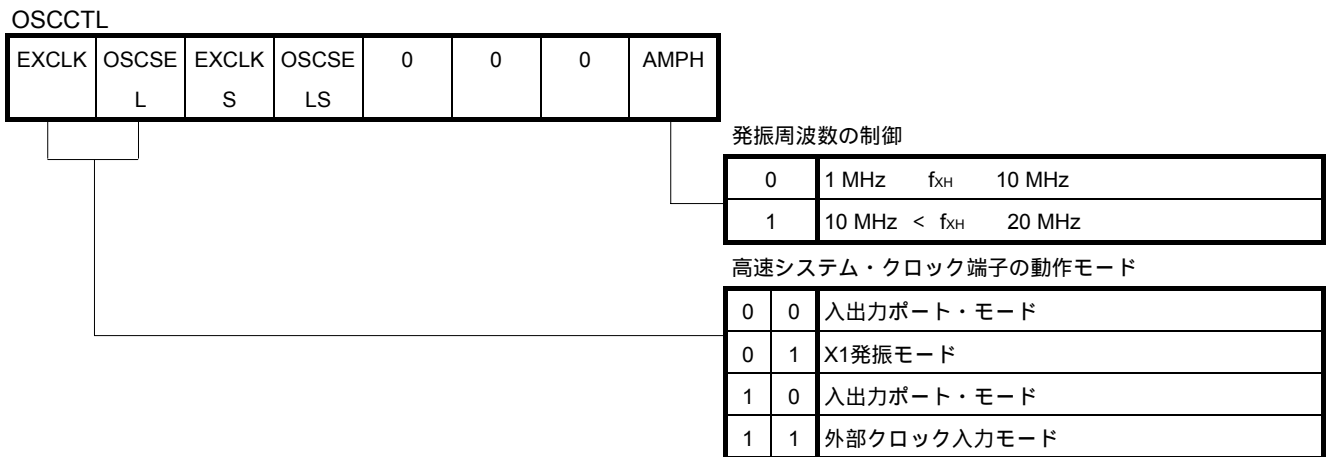
4.5 クロックの設定

(1) クロック周波数の設定

CPUクロック周波数 (f_{CPU}) と周辺ハードウェアへの供給クロック周波数 (f_{XP}) は、メイン・システム・クロックを分周して、生成されます。

OSCCTLでクロック動作モードを選択します。

図4 - 2 クロック動作モード選択レジスタ



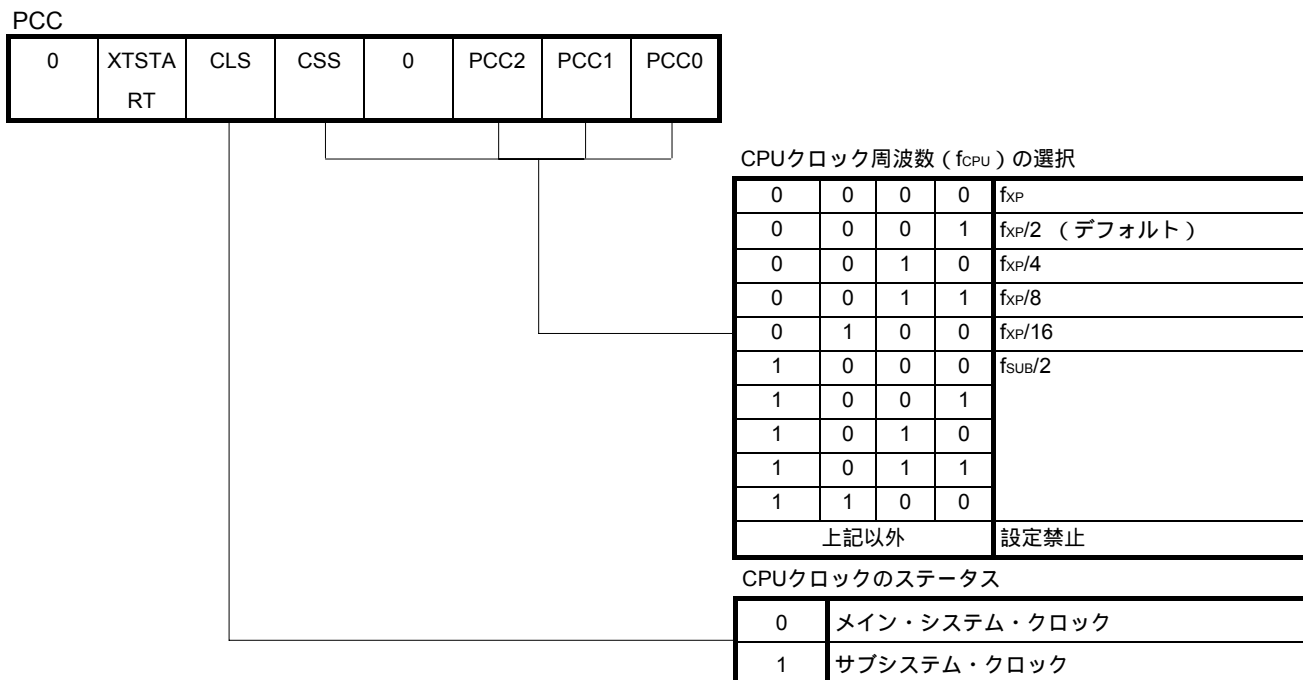
- 注意1. 高速システム・クロック周波数が10 MHzを超える場合は、必ずAMPHに1を設定してください。
2. OSCCTLのビット1-5には必ず0を必ず設定してください(サブシステム・クロック非搭載デバイスの場合)。

- 備考1. f_{XH} : 高速システム・クロック周波数
2. EXCLKS, OSCSELSはサブシステム・クロック搭載デバイスのみ有効です。

デバイス	78K0/KB2	78K0/KC2	78K0/KD2	78K0/KE2	78K0/KF2
サブシステム・クロック搭載	x				

PCCでCPUクロック周波数 (f_{CPU}) の選択, 分周比を設定します。

図4 - 3 プロセッサ・クロック・コントロール・レジスタ (PCC)



- 注意1. PCCのビット3-7には必ず0を必ず設定してください(サブシステム・クロック非搭載デバイスの場合)。
 2. XTSTART, CLS, CSSはサブシステム・クロック搭載デバイスのみ有効です。

- 備考1. f_{XP} : メイン・システム・クロック周波数
 2. f_{SUB} : サブシステム・クロック周波数

78K0/Kx2の一番早い命令はCPUクロック2クロックで実行されます。したがって, CPUクロックと最小命令実行時間の関係は以下になります。

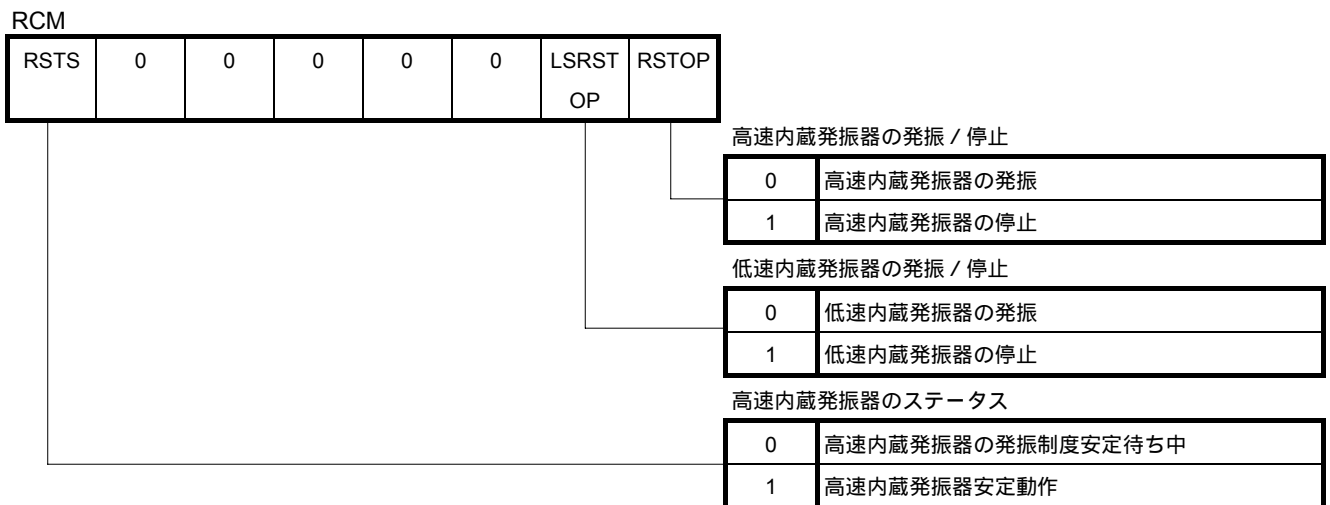
CPUクロック(f _{CPU})	最小命令実行時間 : 2/ f _{CPU}			
	メイン・システム・クロック			サブシステム・クロック
	高速システム・クロック		高速内蔵発振クロック	
	10 MHz動作時	20 MHz動作時	8 MHz (TYP.) 動作時	
f _{XP}	0.2 μs	0.1 μs	0.25 μs (TYP.)	-
f _{XP} /2	0.4 μs	0.2 μs	0.5 μs (TYP.)	-
f _{XP} /4	0.8 μs	0.4 μs	1.0 μs (TYP.)	-
f _{XP} /8	1.6 μs	0.8 μs	2.0 μs (TYP.)	-
f _{XP} /16	3.2 μs	1.6 μs	4.0 μs (TYP.)	-
f _{SUB} /2	-		-	112.1 μs

注 CPUクロックに供給するメイン・システム・クロックの設定 (高速システム・クロック / 高速内蔵発振クロック) は, メイン・クロック・モード・レジスタ (MCM) で行います。

備考 サブシステム・クロックの設定に関しては, 各製品のユーザズ・マニュアルを参照してください。

RCMで内蔵発振モードを選択します。

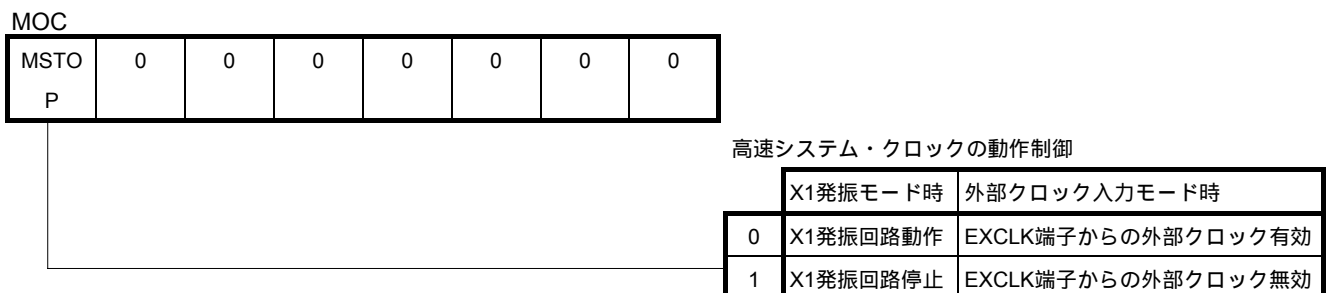
図4 - 4 内蔵発振モード・レジスタ



- 注意1. リセット解除後は00Hですが ,高速内蔵発振器の発振制度安定待ち後に ,自動的に80Hに切り替わります。
2. ビット7はRead Onlyです。
 3. RSTOPに1を設定するとき ,必ずCPUクロックが高速内蔵発振クロック以外で動作していることを確認してください。また ,高速内蔵発振クロックで動作している周辺ハードウェアを停止してから ,RSTOPに1を設定してください。

MOCで高速システム・クロックの動作モードを設定します。

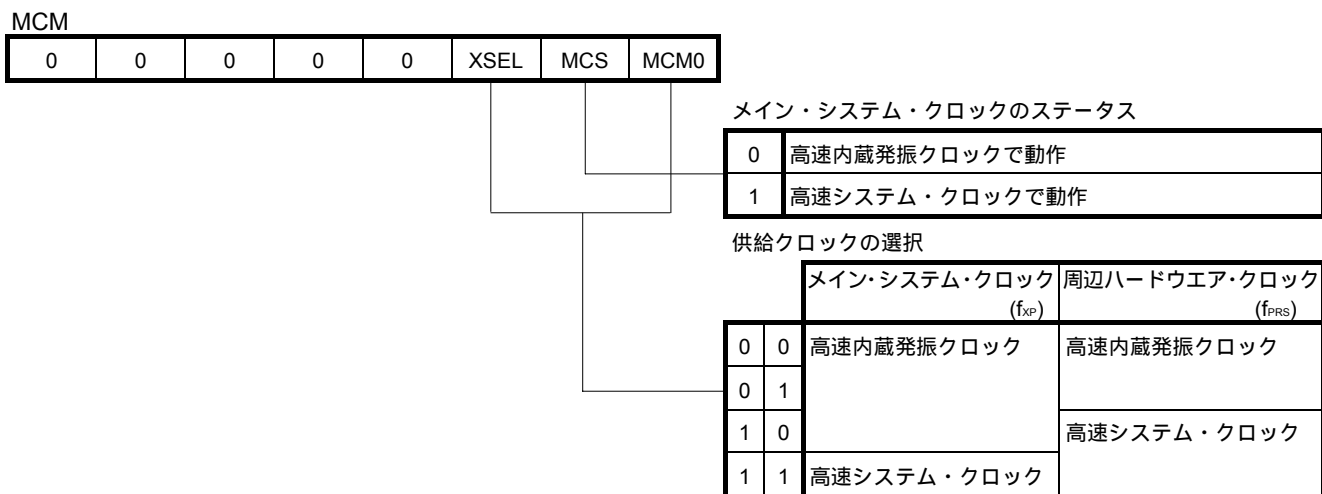
図4 - 5 メインOSCコントロール・レジスタ



- 注意1. MSTOPに1を設定するとき ,必ずCPUクロックが高速システム・クロック以外で動作していることを確認してください。また ,高速システム・クロックで動作している周辺ハードウェアを停止してから ,MSTOPに1を設定してください。
2. クロック動作モード・レジスタ (OSCCTL) のビット6 (OSCSEL) が0のとき (入出力ポート・モード) ,MSTOPに0を設定しないでください。
 3. 周辺ハードウェア・クロックを停止すると ,周辺ハードウェアは動作不可になります。停止後に再開する場合は ,周辺ハードウェアを初期化してください。

MCMでCPUクロックに供給するメイン・システム・クロックの選択と、周辺ハードウェア・クロックに供給するクロックの選択をします。

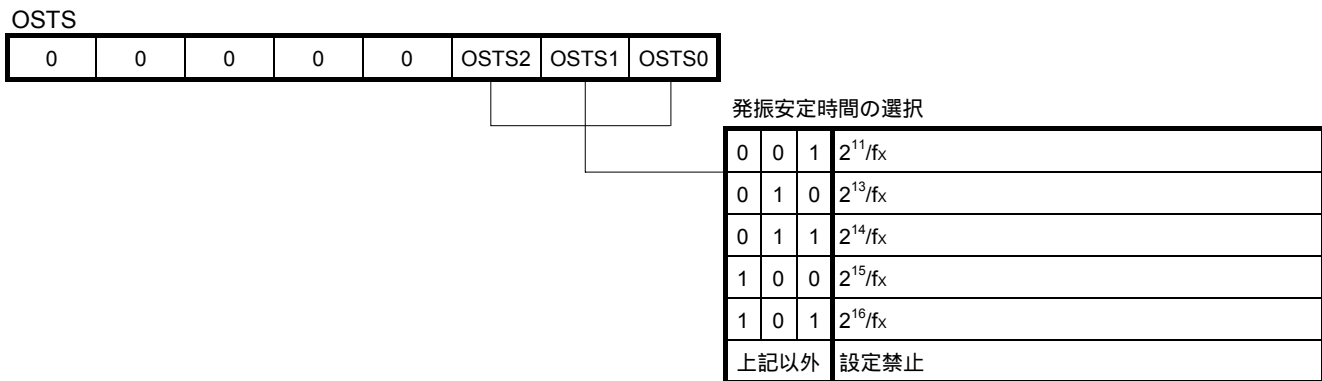
図4 - 6 メイン・クロック・モード・レジスタ



注意 XSELはリセット解除後、1回だけ設定が可能です。

OSTSでX1クロックの発振安定時間を選択します。

図4 - 7 発振安定時間選択レジスタ



注意1. CPUクロックがX1クロック時にSTOPモードへ移行する場合は、STOP命令を実行する前にOSTSを設定してください。

2. X1クロックの発振安定時間中は、OSTSレジスタを変更しないでください。

備考 f_x : X1クロック発振周波数

【例】 システム・クロック (f_x) に「高速内蔵発振クロック (8 MHz (TYP.))」を選択する。

OSCCTL



発振周波数の制御

0	1 MHz	f_{XH}	10 MHz
---	-------	----------	--------

高速システム・クロック端子の動作モード

0	0	出力ポート・モード
---	---	-----------

PCC



CPUクロック周波数 (f_{CPU}) の選択

0	0	0	0	f_{XP}
---	---	---	---	----------

CPUクロックのステータス

0	メイン・システム・クロック
---	---------------

RCM



高速内蔵発振器の発振 / 停止

0	高速内蔵発振器の発振
---	------------

低速内蔵発振器の発振 / 停止

0	低速内蔵発振器の発振
---	------------

MOC



高速システム・クロックの動作制御

	X1発振モード時	外部クロック入力モード時
1	X1発振回路停止	EXCLK端子からの外部クロック無効

MCM



メイン・システム・クロックのステータス

0	高速内蔵発振クロックで動作
---	---------------

供給クロックの選択

	メイン・システム・クロック (f_{XP})	周辺ハードウェア・クロック (f_{PRS})
0	0	高速内蔵発振クロック

ソフトウェアを記述すると、次のようになります。

【C言語】

```
OSCCTL = 0b00000000;  
PCC     = 0b00000000;  
RCM     = 0b00000000;  
MOC     = 0b10000000;  
MCM     = 0b00000000;
```

【アセンブリ言語】

```
MOV     OSCCTL, #00000000B  
MOV     PCC,    #00000000B  
MOV     RCM,    #00000000B  
MOV     MOC,    #10000000B  
MOV     MCM,    #00000000B
```

4.6 ポートの設定

注意 各製品により、内蔵するポートは異なるため、設定するポートも異なります。

	78K0/KB2	78K0/KC2	78K0/KD2	78K0/KE2	78K0/KF2
ポート0	P00, P01	P00, P01	P00-P03	P00-P06	P00-P06
ポート1	P10-P17	P10-P17	P10-P17	P10-P17	P10-P17
ポート2	P20-P23	P20-P27	P20-P27	P20-P27	P20-P27
ポート3	P30-P33	P30-P33	P30-P33	P30-P33	P30-P33
ポート4	-	P40, P41	P40, P41	P40-P43	P40-P47
ポート5	-	-	-	P50-P53	P50-P57
ポート6	P60-P63	P60-P63	P60-P63	P60-P63	P60-P67
ポート7	-	P70-P74, P75	P70-P77	P70-P77	P70-P77
ポート12	P120-P121	P120-P124	P120-P124	P120-P124	P120-P124
ポート13	-	P130	P130	P130	P130
ポート14	-	P140	P140	P140, P141	P140-P145

(1) ポートの入力/出力の設定

PMxxで、ポートを入力ポートまたは出力ポートとして使用するかを設定します。リセット解除後は、入力ポートに設定されます。

PMxxのフォーマットは、PM1を例にして、説明します。

このサンプル・プログラムでは、ポート3を後述の【例 1】、ポート1を【例 2】の内容で設定しています。

図4-8 ポート・モード・レジスタ1 (PM1) のフォーマット

PM1							
PM17	PM16	PM15	PM14	PM13	PM12	PM11	PM10

P1n (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

(2) 出力ポートの出力ラッチの設定

Pxxで、出力ポートの出力ラッチの値を0または1にするかを設定します。リセット解除後は、出力ラッチの値は0に初期化されます。

Pxxのフォーマットは、P1を例にして、説明します。

このサンプル・プログラムでは、ポート3を後述の【例 1】の内容で設定しています。

図4-9 ポート・レジスタ1 (P1) のフォーマット

P1							
P17	P16	P15	P14	P13	P12	P11	P10

P1n (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

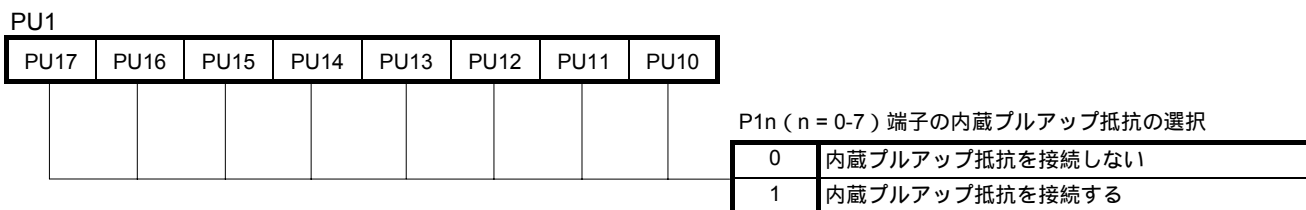
(3) 入力ポートの内蔵プルアップ抵抗接続の設定

PUxxで、入力ポートに内蔵プルアップ抵抗を接続するか否かを設定します。リセット解除後は、内蔵プルアップ抵抗に接続しません。

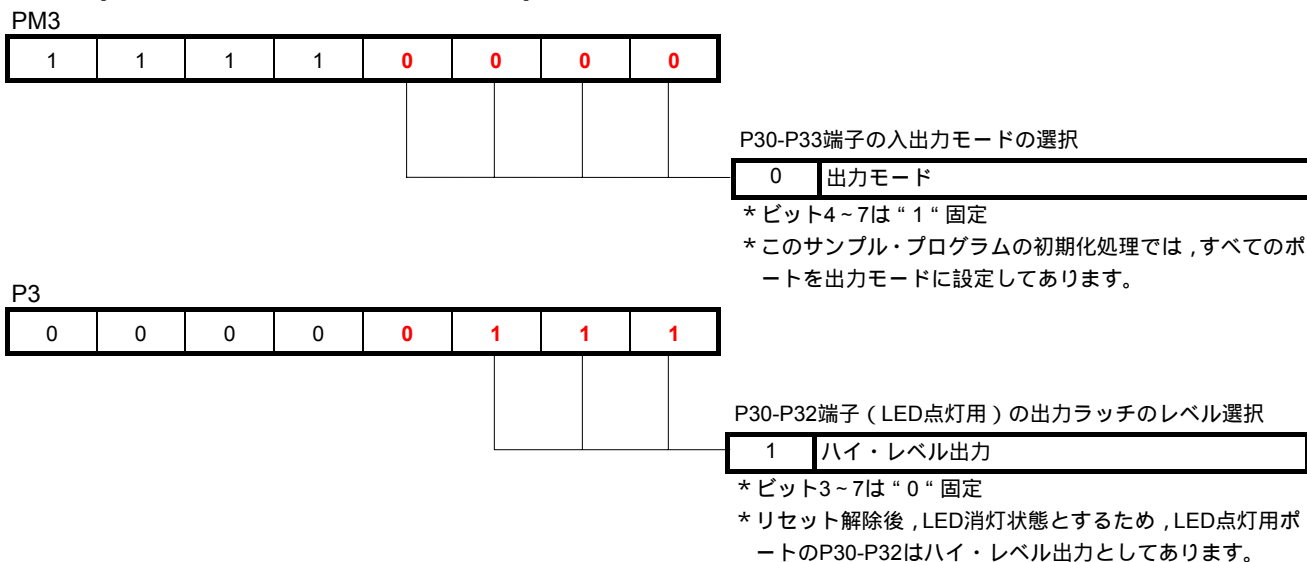
PUxxのフォーマットは、PU1を例にして、説明します。

このサンプル・プログラムでは、ポート1を後述の【例 2】の内容で設定しています。

図4 - 10 プルアップ抵抗オプション・レジスタ1 (PU1) のフォーマット



- 【例 1】
- ・ P30-P33を出力ポートに設定 (P30-P32はLED点灯用)
 - ・ P30-P32の出力ラッチの値を1に設定 (P30-P32はLED点灯用)
- (サンプル・プログラムの設定を同内容)



PM3の設定値は「11110000 (ビット7-4は1に必ず設定)」、P3の設定値は「00000111 (ビット7-3は0に必ず設定)」となります。

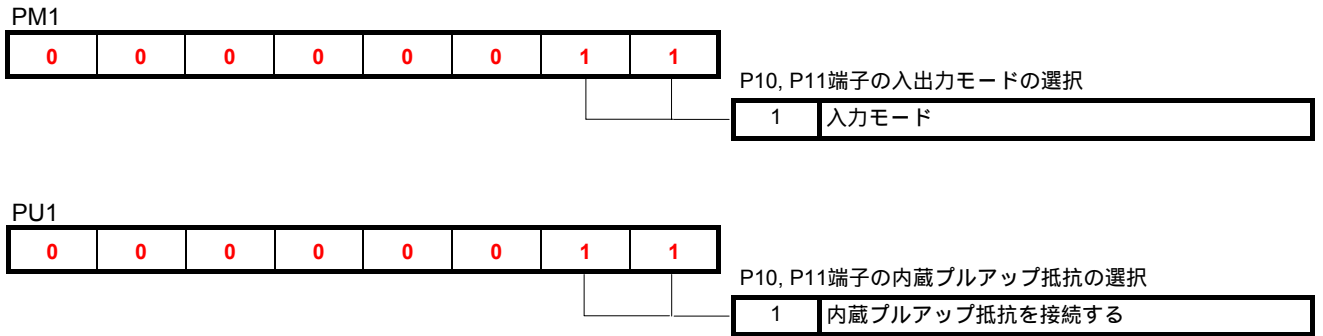
【C言語】

```
PM3 = 0b11110000;
P3 = 0b00000111;
```

【アセンブリ言語】

```
MOV PM3, #11110000B
MOV P3, #00000111B
```

- 【例 2】** ・ P10, P11を入力ポートに設定
 ・ P10, P11に内蔵プルアップ抵抗接続を設定
 (サンプル・プログラムの設定を同内容)



【C言語】

```
PM1 = 0b00000011
PU1 = 0b00000011
```

【アセンブリ言語】

```
MOV    PM1,    #00000011B
MOV    PU1,    #00000011B
```

備考 サンプル・プログラムでのLED処理におけるポートの設定はmain関数内で設定します。

4.7 メイン処理

メイン処理では、次の動作を行います。

入力データと出力データの対応を、配列で設定します。

P1, PM1にデータをセットします。

P10, P11の入力レベル情報を取得します。

取り出したデータを表示データ・パターン配列に当てはめ、出力用のデータを取得し、P3に出力します。

【C言語】 (Kx2_Init.c)

```

/*****
;
;   メイン処理
;
; *****/
const unsigned char ucOutData[4] = {0x03,0x05,0x06,0x07};
/* LCD点灯データパターン用配列*/

void main(void)
{
    unsigned char ucInData; /* スイッチ入力データ変数 */

/*-----
;   各モジュール用の初期化処理実行
;-----*/

    PM1 = 0b00000011; /* P10とP11に入力ポートに設定
                       (スイッチ入力) */

    PU1 = 0b00000011; /* P10とP11を内蔵プルアップ
                       抵抗を接続*/

    PM3 = 0b00000000; /* P3を出力モードに設定 */
    P3 = 0b00000111; /* P30~P32初期値 "1" を出力
                       (LED消灯) */

/*-----
;   各モジュール用のメイン処理実行
;-----*/

    while(1)
    {
        ucInData = P1 & 0b00000011; /* 有効スイッチ情報の取得 */
        P3 = ucOutData[ucInData]; /* テーブルから表示データを
                                   読み出してLED点灯 */
    }
}

```

入力データと出力データの対応は次のようになります。

スイッチ入力	P10, P11	INDATA	OUTDATA	LED点灯
SW1 = ON, SW2 = ON	P10 = 0, P11 = 0	0b00000000	0x03	LED3のみ点灯
SW1 = OFF, SW2 = ON	P10 = 1, P11 = 0	0b00000001	0x05	LED2のみ点灯
SW1 = ON, SW2 = OFF	P10 = 0, P11 = 1	0b00000010	0x06	LED1のみ点灯
SW1 = OFF, SW2 = OFF	P10 = 1, P11 = 1	0b00000011	0x07	全LED消灯

【アセンブリ言語】 (Kx2_Init.asm)

```

;*****
;
;   メイン処理
;
;*****
;-----
;   各モジュール用の初期化処理実行
;-----
MOV     PM1,    #00000011B           ;P10とP11を入力ポートに設定
                                           ;(スイッチ入力)
MOV     PU1,    #00000011B           ;P10とP11に内蔵プルアップ抵抗を接続
MOV     PM3,    #00000000B           ;P3を出力モードに設定
MOV     P3,     #00000111B           ;P30~P32初期値“1”を出力(LED消灯)
MMAIN:
MOV     A,      P1                    ;スイッチ情報の取得
                                           ;(どのスイッチが押下されたか)
AND     A,      #00000011B           ;有効スイッチ情報の取得
MOV     B,      A                    ;[HL+B]として使うため,Bレジスタに代入
MOVW   HL,     #OUTDATA              ;LED点灯データのアドレスを取得
                                           ;(210H番地)
MOV     A,      [HL+B]               ;テーブルから表示データを読み出す
MOV     P3,     A                    ;出力ポート3にLED点灯データを書き込み,
                                           ;LED点灯
BR     MMAIN

```

入力データと出力データの対応は次のようになります。

スイッチ入力	P10, P11	INDATA	OUTDATA	LED点灯
SW1 = ON, SW2 = ON	P10 = 0, P11 = 0	0b00000000	0x03	LED3のみ点灯
SW1 = OFF, SW2 = ON	P10 = 1, P11 = 0	0b00000001	0x05	LED2のみ点灯
SW1 = ON, SW2 = OFF	P10 = 0, P11 = 1	0b00000010	0x06	LED1のみ点灯
SW1 = OFF, SW2 = OFF	P10 = 1, P11 = 1	0b00000011	0x07	全LED消灯

LED表示用データ・テーブル


```

OUTDATA:
DB     00000011B } ;LED3点灯
DB     00000101B } ;LED2点灯
DB     00000110B } ;LED1点灯
DB     00000111B } ;LEDは全て消灯


```

LED表示用に4つのデータテーブルを定義する。

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

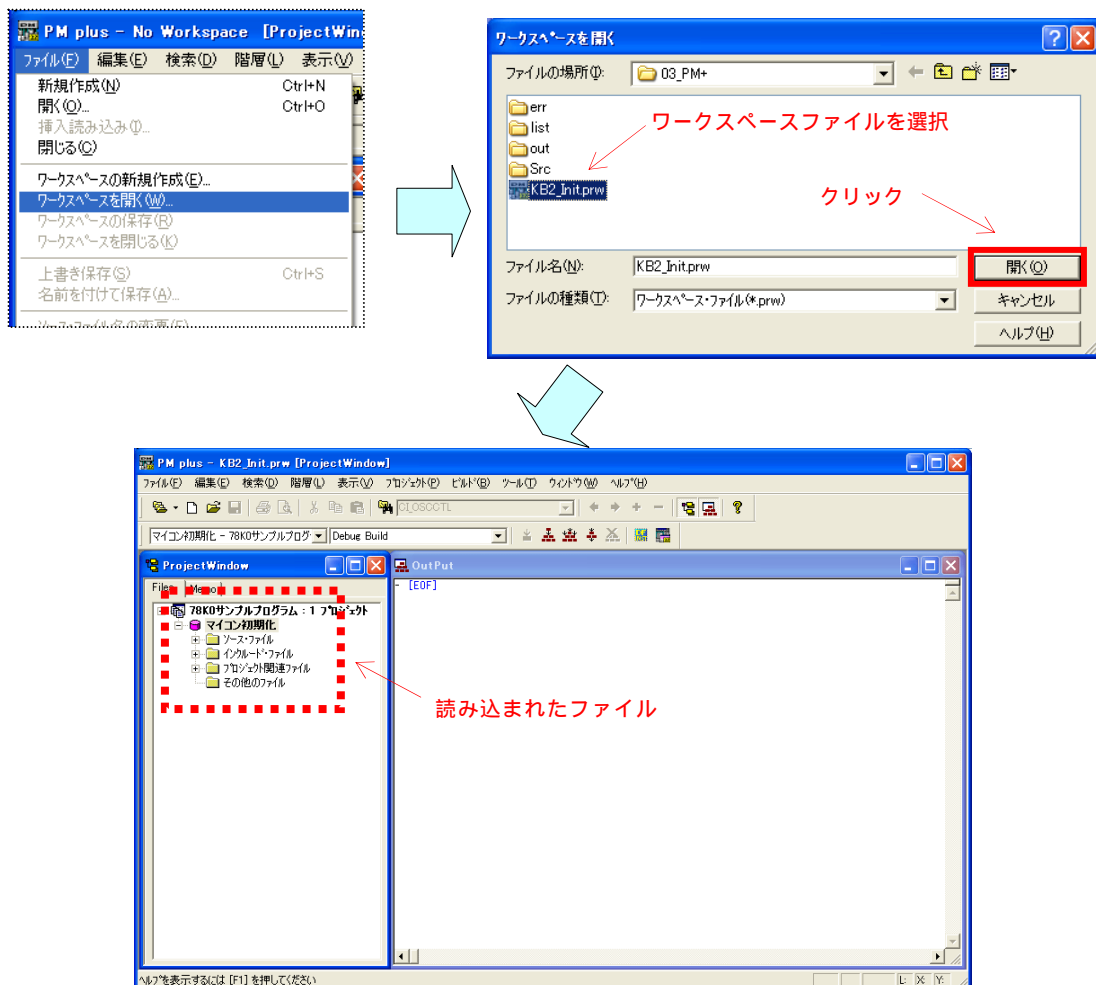
サンプル・プログラムをSM+ for 78K0/Kx2 (以降、「SM+」と表記します)で動作確認をするために、サンプル・プログラムをビルドしてから、SM+を起動する必要があります。ここでは、でダウンロードしたC言語用のファイルを用いて、統合開発環境 PM+にてビルドしてから、SM+を起動するまでの動作の一例を説明します。

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。

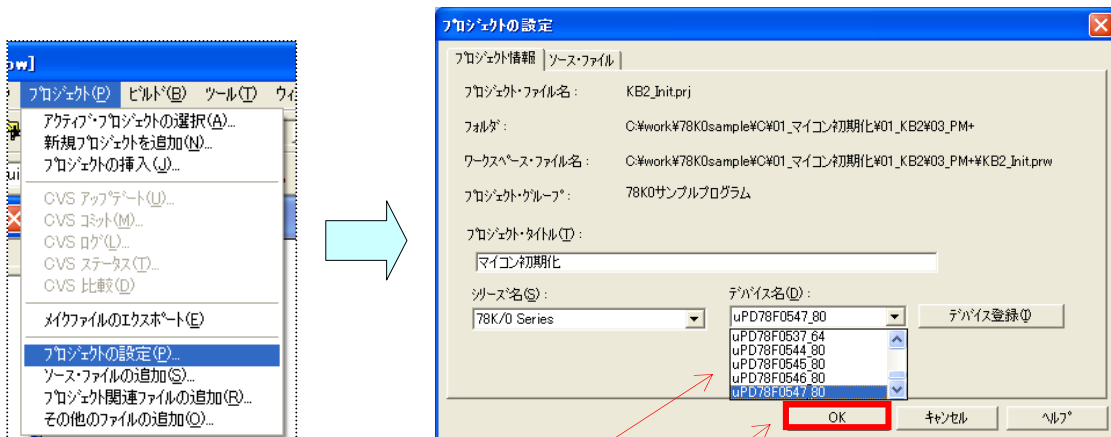
- (1) PM+を起動してください。
- (2) [ファイル] [ワークスペースを開く] から、「Kx2_Init.prw」[※]を選択し、[開く] ボタンをクリックしてください。ワークスペースが作成され、その中にソース・ファイルが自動的に読み込まれます。

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_Init.prw」



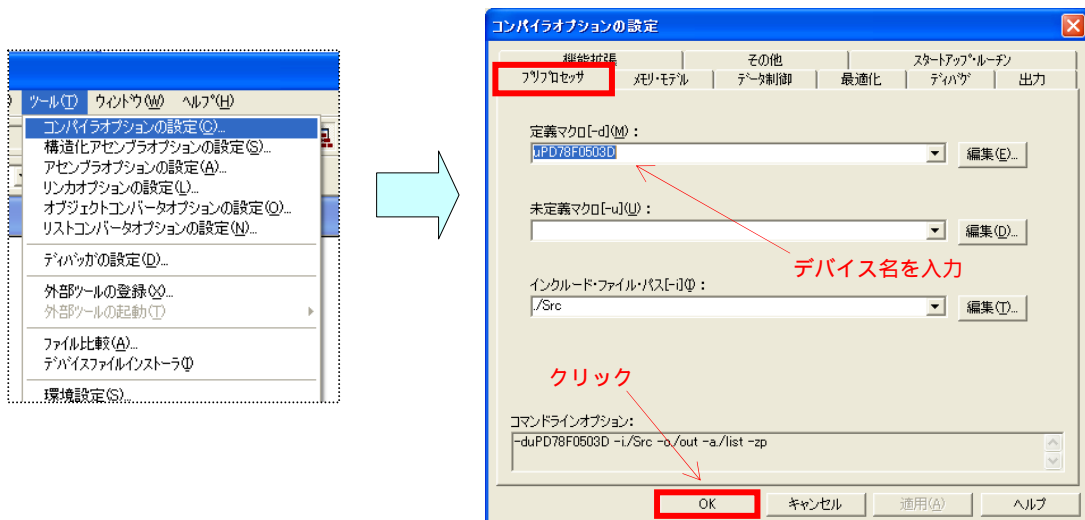
- (3) [プロジェクト] [プロジェクトの設定] を選択してください。[プロジェクトの設定] 画面が表示されたら、使用するデバイス名を選択（デフォルトでは、ROM/RAMサイズの最も大きいデバイスが選択）し、[OK] ボタンをクリックしてください。



デバイス名を指定 クリック


μ PD78F0500_36, μ PD78F0501_36, μ PD78F0502_36, μ PD78F0503_36は選択しないでください。

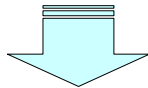
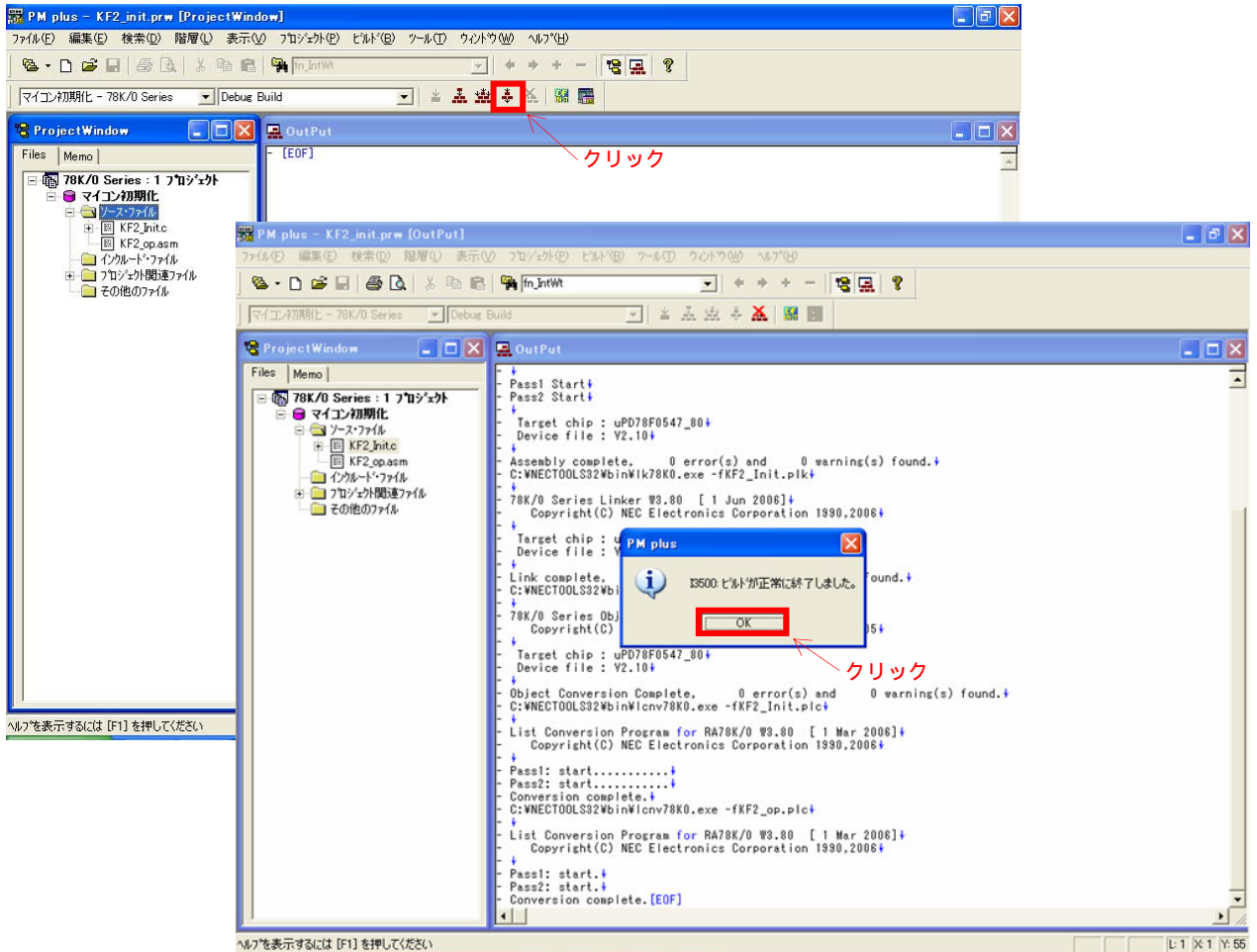
- (4) [ツール] [コンパイラオプションの設定] を選択してください。[コンパイラオプションの設定] 画面が表示されたら、[プリプロセッサ] タグページが表示されているのを確認し、その中の定義マクロ欄に使用するデバイス名を入力し、[OK]をクリックします。



デバイス名を入力

クリック

- (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_Init.c」と「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。



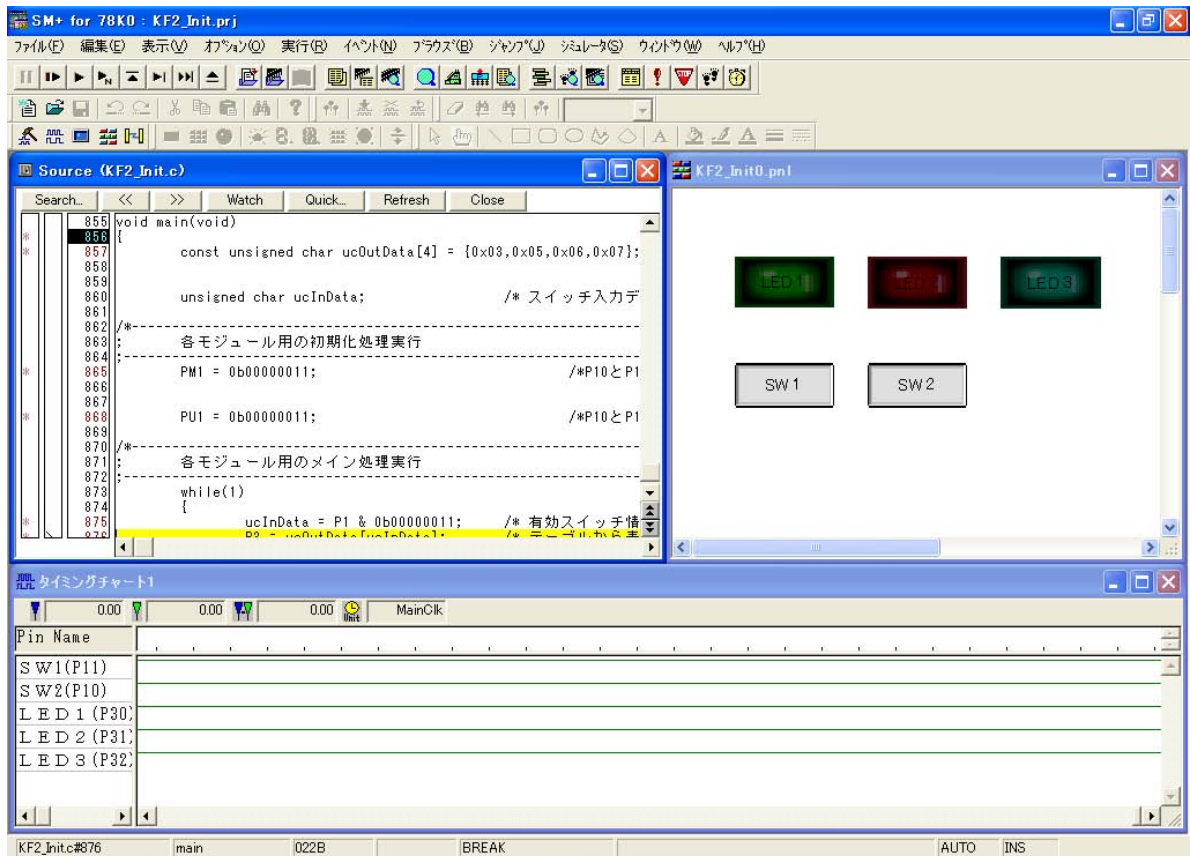
SM+が自動的に起動されます


5.2 SM+での動作

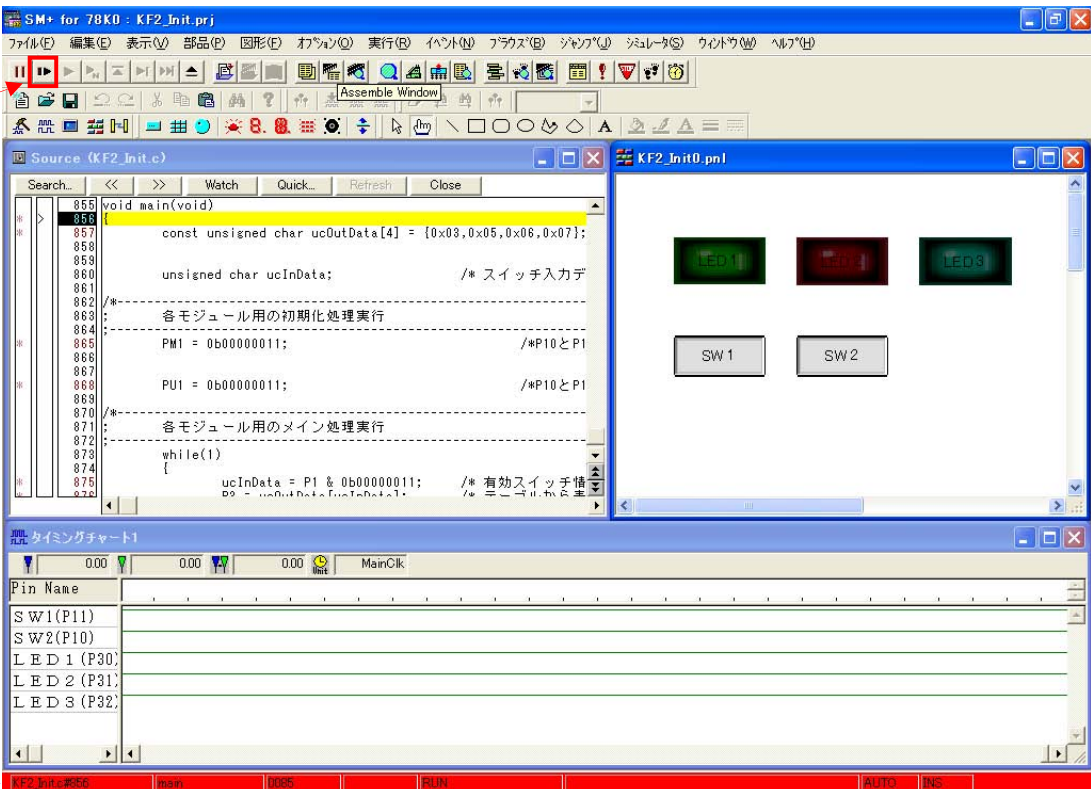
ここでは、SM+の入出力パネル・ウインドウやタイミング・チャート・ウインドウ上での動作確認の例を説明します。

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。



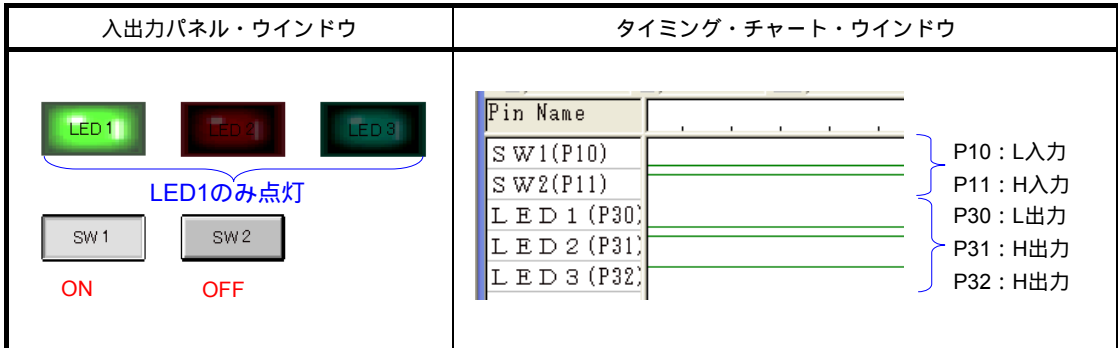
The screenshot shows the SM+ simulator interface for a project named 'KF2_Init.prj'. The main window displays the source code for 'KF2_Init0.pnl'. The code includes a `main` function with initialization and a `while` loop. The graphical window shows three LEDs (LED1, LED2, LED3) and two switches (SW1, SW2). The status bar at the bottom indicates the current state of the simulation, with the text 'KF2_Init0#50' highlighted in red.

クリック

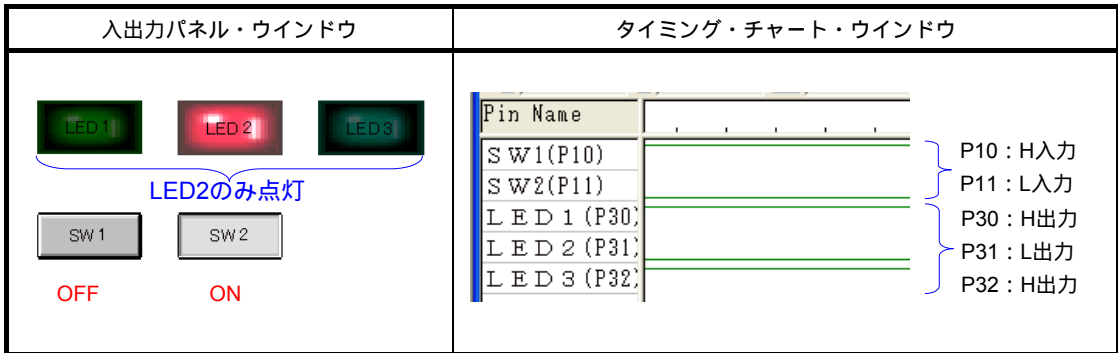
プログラム実行中は、→赤になります。

- (3) プログラム実行中に，入出力パネル・ウインドウ上の [SW1][SW2] ボタンをクリックしてください。
 [SW1][SW2] ボタンの組み合わせにより，入出力パネル・ウインドウ上の [LED1] ~ [LED3] の点灯およびタイミング・チャート・ウインドウ上の波形が変化することを確認してください。

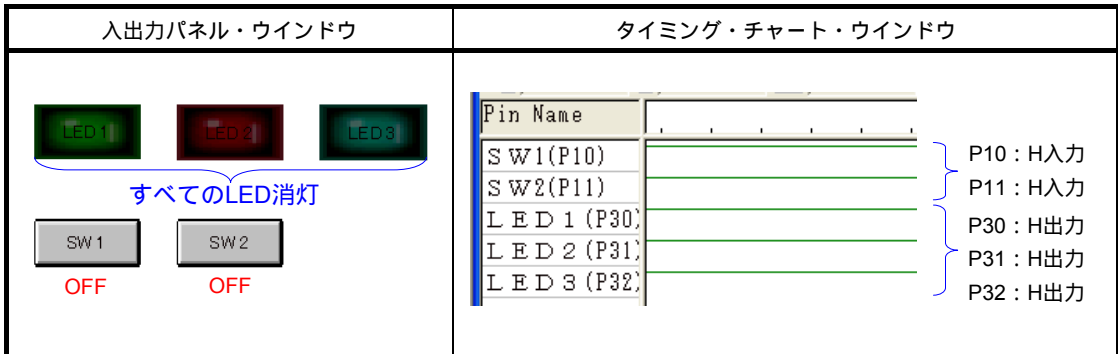
例1. SW1 : ON , SW2 : OFFの場合



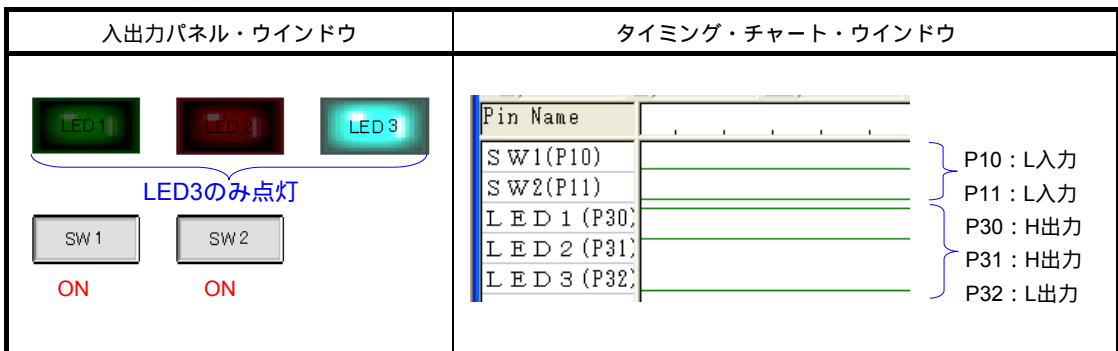
例2. SW1 : OFF , SW2 : ONの場合



例3. SW1 : OFF , SW2 : OFFの場合



例4. SW1 : ON , SW2 : ONの場合



備考 H : ハイ・レベル , L : ロウ・レベル

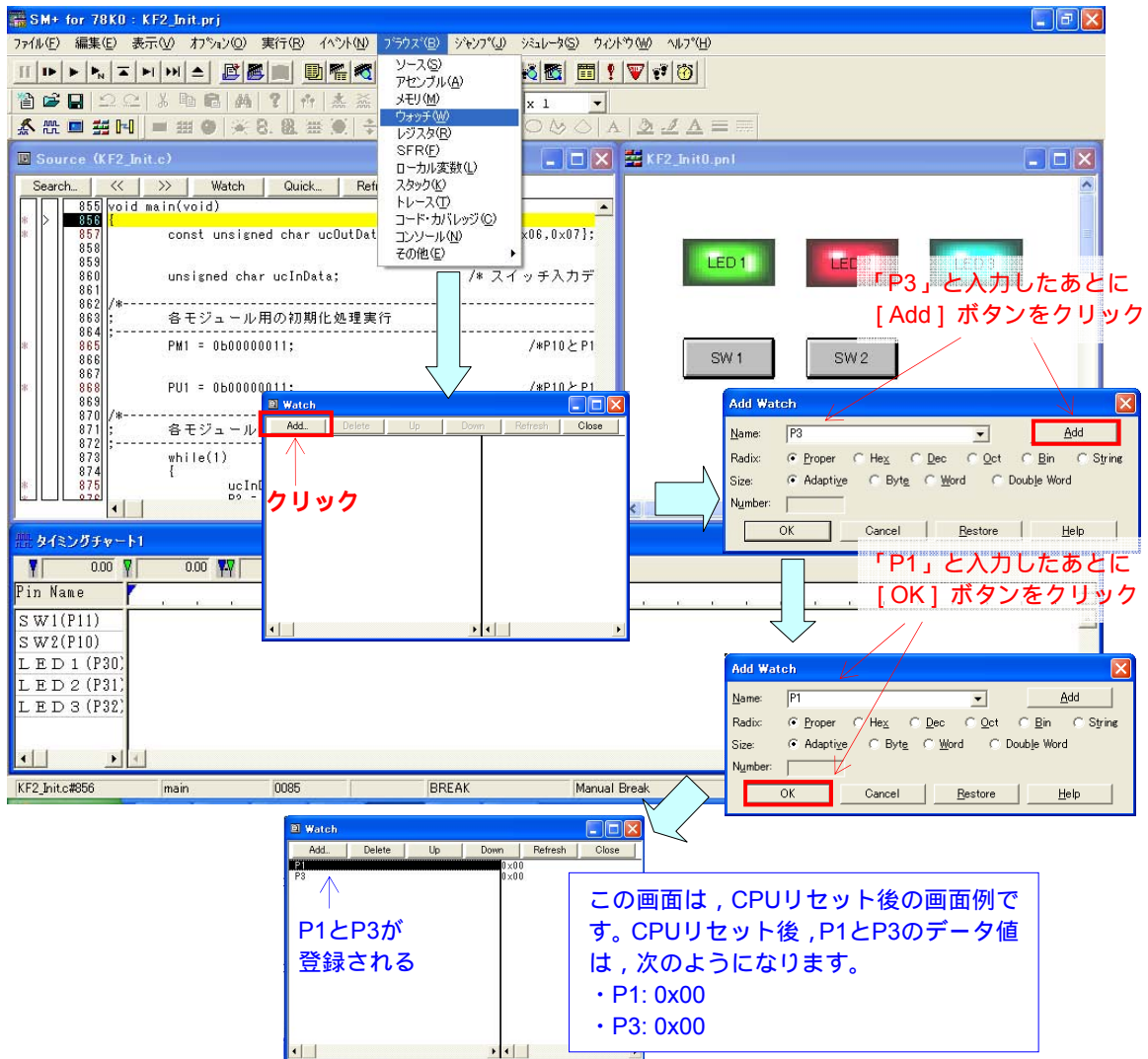
【補 足】SM+のウォッチ機能を使用することにより、ポート1, 2のデータ値の変化を確認することができます。


[ブラウズ] [ウォッチ] を選択してください。[Watch] ウィンドウが立ち上がります。

[Add] ボタンをクリックすると [Add Watch] ウィンドウが立ち上がります(このとき [Watch] ウィンドウは開いたままです)。

Nameに「P3」と入力し,[Add] ボタンをクリックすると,[Watch] ウィンドウに、「P3」が登録されます(このとき,[Add Watch] ウィンドウは開いたままです)。

次に,Nameに「P1」と入力し,[OK] ボタンをクリックすると,[Watch] ウィンドウに、「P1」が登録され,[Add Watch] ウィンドウが閉じられます。



プログラムを実行し、入出力パネル・ウィンドウ上の [SW1] [SW2] ボタンをクリックしてください。[SW1] [SW2] ボタンの組み合わせにより、[Watch] ウィンドウ上のP1とP3のデータ値が変化することを確認してください ( (「ストップ」ボタン) をクリックすることで、確認できます)。

SW1とSW2の組み合わせ	[Watch] ウィンドウのデータ値
SW1: ON, SW2: OFF	P1: 0x02, P3: 0x06
SW1: OFF, SW2: ON	P1: 0x01, P3: 0x05
SW1: OFF, SW2: OFF	P1: 0x03, P3: 0x07
SW1: ON, SW2: ON	P1: 0x00, P3: 0x03

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。

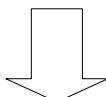
(1) オプション・バイトの設定

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

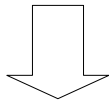
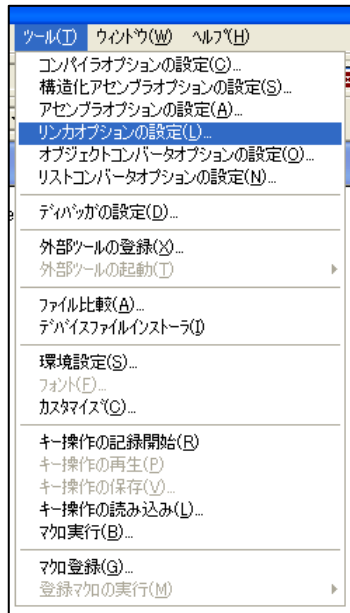
TVECT_TBL1    CSEG      AT      0004H
              DW      IINIT      ;0004H INTLVI
    
```

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。

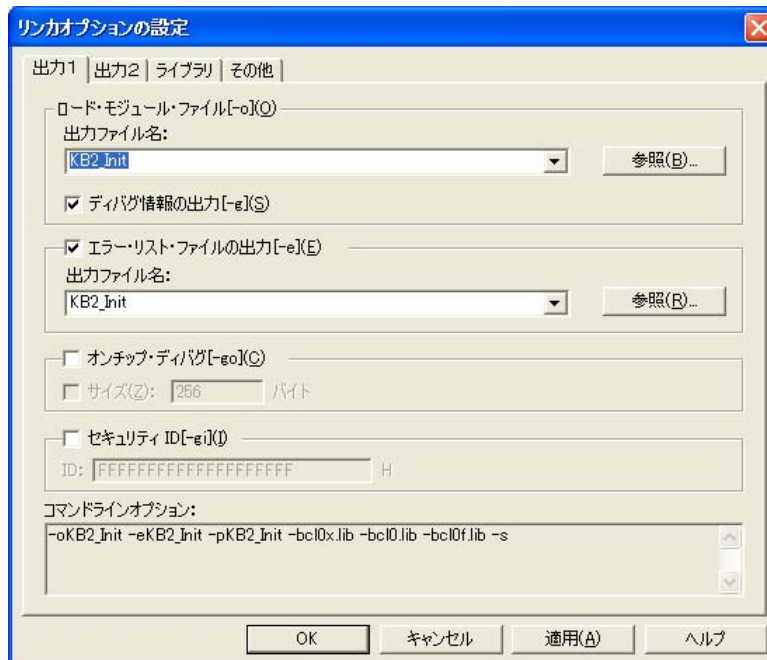
C言語版では不要です。

(3) リンカの設定

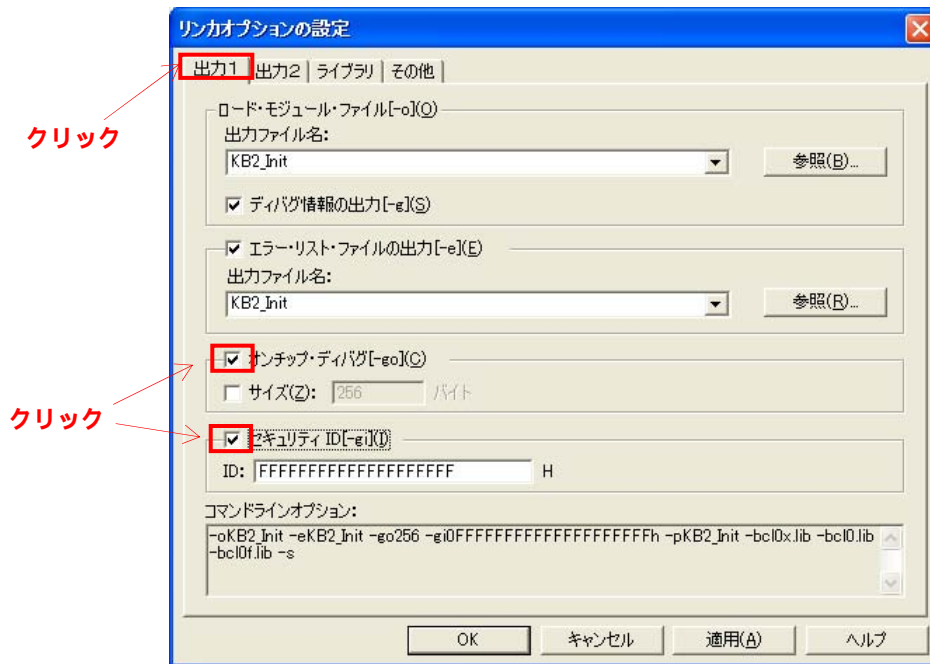
オンチップ・デバッグを行う場合、ビルドの際、リンカの設定を行う必要があります。
PM+の「ツール」メニューから「リンカオプションの設定」を選択してください。



「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・デバッグ」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	November 2007	-	-
第2版	May 2009	全般	全面見直し

【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