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

V850ES/Jx3-L

サンプル・プログラム（初期設定）

LED点灯のスイッチ制御編

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使用する周辺の初期設定を行ったあとに，1つのスイッチ入力により，2つのLED点灯を制御します。

対象デバイス

V850ES/JF3-L マイクロコントローラ
V850ES/JG3-L マイクロコントローラ

目　次

資料番号　U19479JJ1V0AN00
発行年月　October 2008 NS

- **本資料に記載されている内容は2008年10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，クロック周波数の選択，ポート入出力の設定など，V850ES/Jx3-L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，初期設定完了後のメイン処理動作では，1つのスイッチ入力により，2つのLEDの点灯を制御します。

1. 1　初期化

＜オプション・バイトの参照＞

・リセット解除後の発振安定時間を参照

＜初期設定の主な内容＞

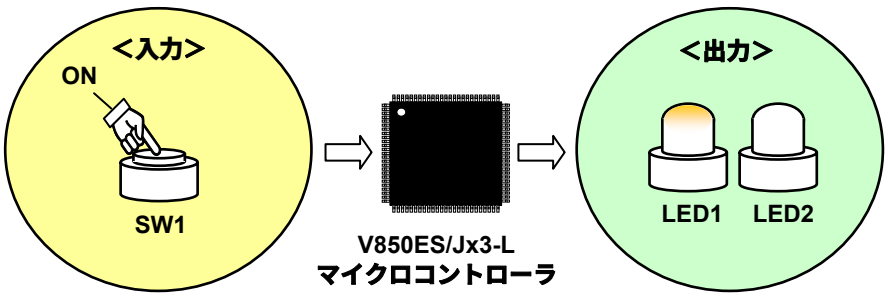
・システム・ウエイト・コントロール・レジスタを1クロックに設定
・オンチップ・デバッグを通常動作モードに設定
・内蔵発振器の停止の設定
・ウォッチドッグ・タイマ2動作停止
・システム・クロックはPLLを使用し4逓倍して20 MHzに設定
・未使用ポートの設定
・SW入力ポート，LED制御ポートの設定

＜ROM化＞

・ROM化処理（初期値あり変数の初期化）　・・・C言語のみ

1. 2　メイン処理動作の内容

V850ES/Jx3-Lマイクロコントローラにて，スイッチ入力（SW1）回数に応じて，2つのLED（LED1, LED2）の点灯を制御します。



スイッチ入力 → LEDの点灯パターンが変化する → スイッチ入力 → LEDの点灯パターンが変化する
→ スイッチ入力 →LEDの点灯パターンが変化する... の繰り返しになります。

表1－1　LED点灯パターン

スイッチ（SW1）の入力回数[注]	LED1	LED2
0回	OFF	OFF
1回	ON	OFF
2回	ON	ON
3回	OFF	ON

注　4回目以降の入力は0回目以降の繰り返しになります。

注意　デバイス使用上の注意事項については，製品のユーザーズ・マニュアル（V850ES/Jx3-L）を参照してください。

【コラム】チャタリングとは

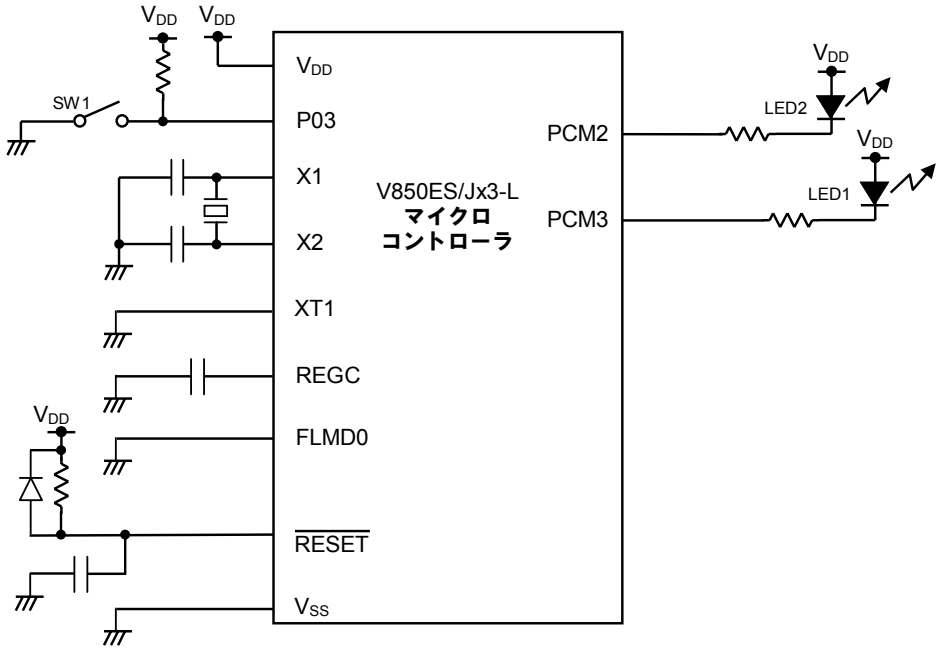
スイッチが切り替わった直後に，接点が機械的にばたつくことにより，電気信号がONとOFFを繰り返す現象のことです。

第2章　回路図

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図および周辺ハードウエアを説明します。

2. 1　回路図

回路図を次に示します。



注意1. EV$_{DD}$**端子**, AV$_{REF0}$**端子**, AV$_{REF1}$**端子は**V$_{DD}$**に直接接続してください。**

2. EV$_{SS}$**端子**, AV$_{SS}$**端子は**GND**に直接接続してください。**
3. FLMD0**端子は，通常モード時は**GND**に接続してください。**
4. REG**はコンデンサ（推奨値：**4.7μF**）を介し，**GND**に接続してください。**
5. **未使用ポートについては，出力ポートとして処理するため，すべてオープンとしてください。**

2. 2　周辺ハードウエア

使用する周辺ハードウエアを次に示します。

（1**）スイッチ（**SW1**）**

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**）**

スイッチ入力回数に対応した出力として，LED**を使用します。**

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（ツリー構造）	説　明	同封圧縮（*.zip）ファイル	
			
c ─ conf ─ crtE.s	スタート・アップ・ルーチン・ファイル 注1	–	●
conf ─ AppNote_LED.dir	リンク・ディレクティブ・ファイル 注2	●	●
conf ─ AppNote_LED.prj	統合開発環境 PM+用プロジェクト･ファイル	–	●
conf ─ AppNote_LED.prw	統合開発環境 PM+用ワーク･スペース･ファイル	–	●
src ─ main.c	マイコンのハードウエア初期化処理とメイン処理を記述したC言語ソース・ファイル	●	●
src ─ minicube2.s	MINICUBE2用の領域予約を行う ソース・ファイル	●	●
src ─ opt_b.s	オプション・バイト設定を行う ソース・ファイル	●	●

注1. ワークスペース新規作成時の「スタート・アップ・ファイルの指定」時に，「サンプルをコピーして使用する(C)」を選択した際にコピーされるスタート・アップ・ファイル。（デフォルト・インストール・パスであれば，C:¥Program Files¥NEC Electronics Tools¥CA850¥*使用バージョン*¥lib850¥r32¥crtE.s のコピーとなります。）

2. ワークスペース新規作成時の「リンク・ディレクティブ・ファイルの指定」時に，「サンプルを作成して使用する(C)」を選択し，「メモリの使用方法：内蔵メモリのみ(I)」をチェックした際に，自動生成されるリンク・ディレクティブ・ファイルに，MINICUBE2用のセグメントを追加したもの。（デフォルト・インストール・パスであれば，C:¥Program Files¥NEC Electronics Tools¥PM+¥*使用バージョン* ¥bin¥w_data¥V850_i.dat が基準となります。）

備考　  　：ソース・ファイルのみ同封

 　：統合開発環境 PM+で使用するファイルを同封

【アセンブラ版】

ファイル名（ツリー構造）	説 明	同封圧縮（*.zip）ファイル	
			
asm ┬ conf ┬ crtE.s	スタート・アップ・ルーチン・ファイル[注1]	–	●
├ AppNote_LED.dir	リンク・ディレクティブ・ファイル[注2]	●	●
├ AppNote_LED.prj	統合開発環境 PM+用プロジェクト･ファイル	–	●
└ AppNote_LED.prw	統合開発環境 PM+用ワーク･スペース･ファイル	–	●
└ src ┬ main.s	マイコンのハードウエア初期化処理とメイン処理を記述したアセンブラソース・ファイル	●	●
├ minicube2.s	MINICUBE2用の領域予約を行うソース・ファイル	●	●
└ opt_b.s	オプション・バイト設定を行うソース・ファイル	●	●

注1. ワークスペース新規作成時の「スタート・アップ・ファイルの指定」時に，「サンプルをコピーして使用する(C)」を選択した際にコピーされるスタート・アップ・ファイル。（デフォルト・インストール・パスであれば，C:¥Program Files¥NEC Electronics Tools¥CA850¥*使用バージョン* ¥lib850¥r32¥crtE.s のコピーとなります。）

2. ワークスペース新規作成時の「リンク・ディレクティブ・ファイルの指定」時に，「サンプルを作成して使用する(C)」を選択し，「メモリの使用方法：内蔵メモリのみ(I)」をチェックした際に，自動生成されるリンク・ディレクティブ・ファイルに，MINICUBE2用のセグメントを追加したもの。（デフォルト・インストール・パスであれば，C:¥Program Files¥NEC Electronics Tools¥PM+¥*使用バージョン* ¥bin¥w_data¥V850_i.dat が基準となります。）

備考　  　：ソース・ファイルのみ同封

 　：統合開発環境 PM+で使用するファイルを同封

3. 2　使用する内蔵周辺機能

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。

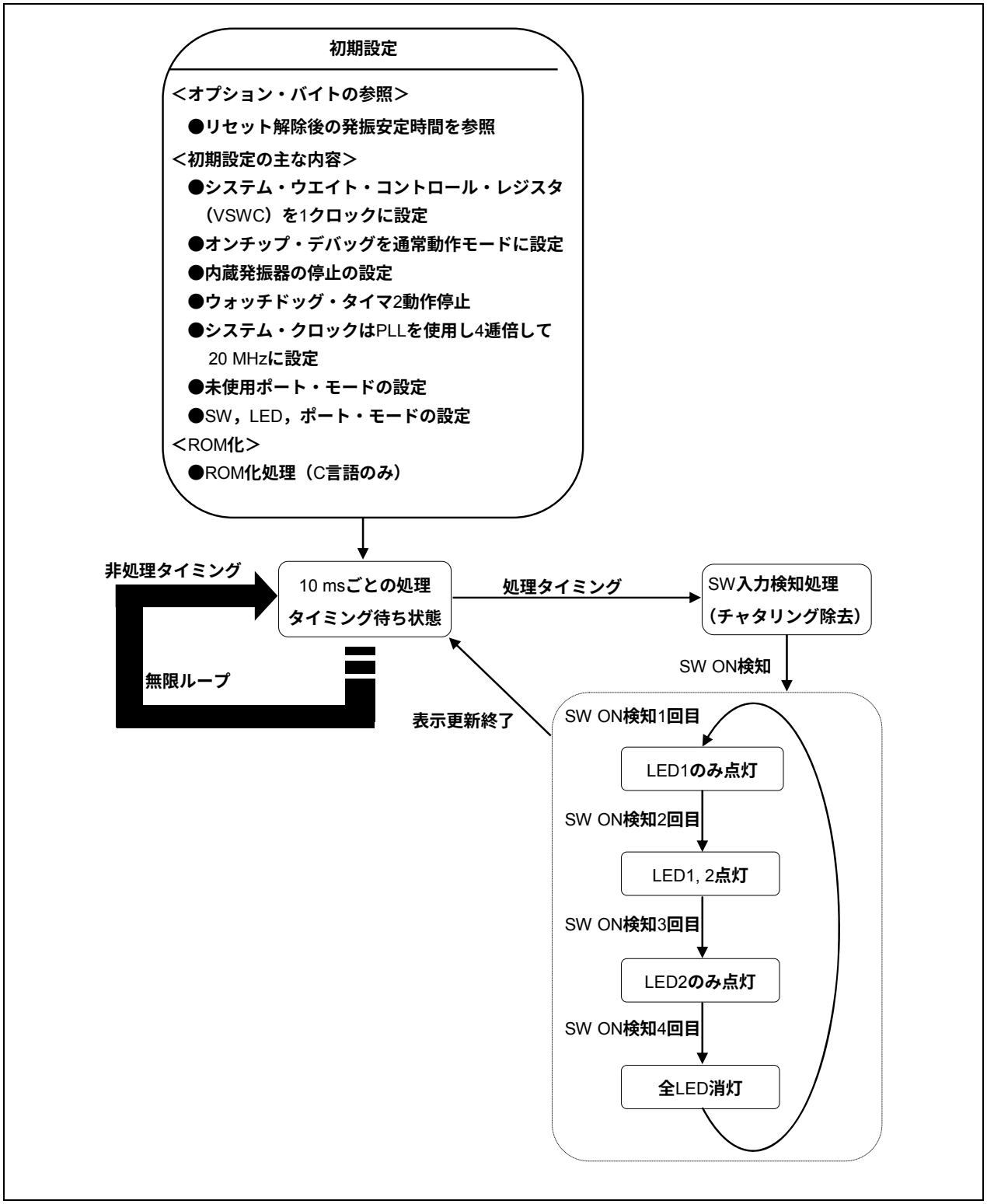
・入力ポート（スイッチ入力用）　：P03（SW1）
・出力ポート（LED点灯用）　：PCM3（LED1），PCM2（LED2）

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，ウォッチドッグ・タイマの停止設定，入出力ポートの設定などを行います。

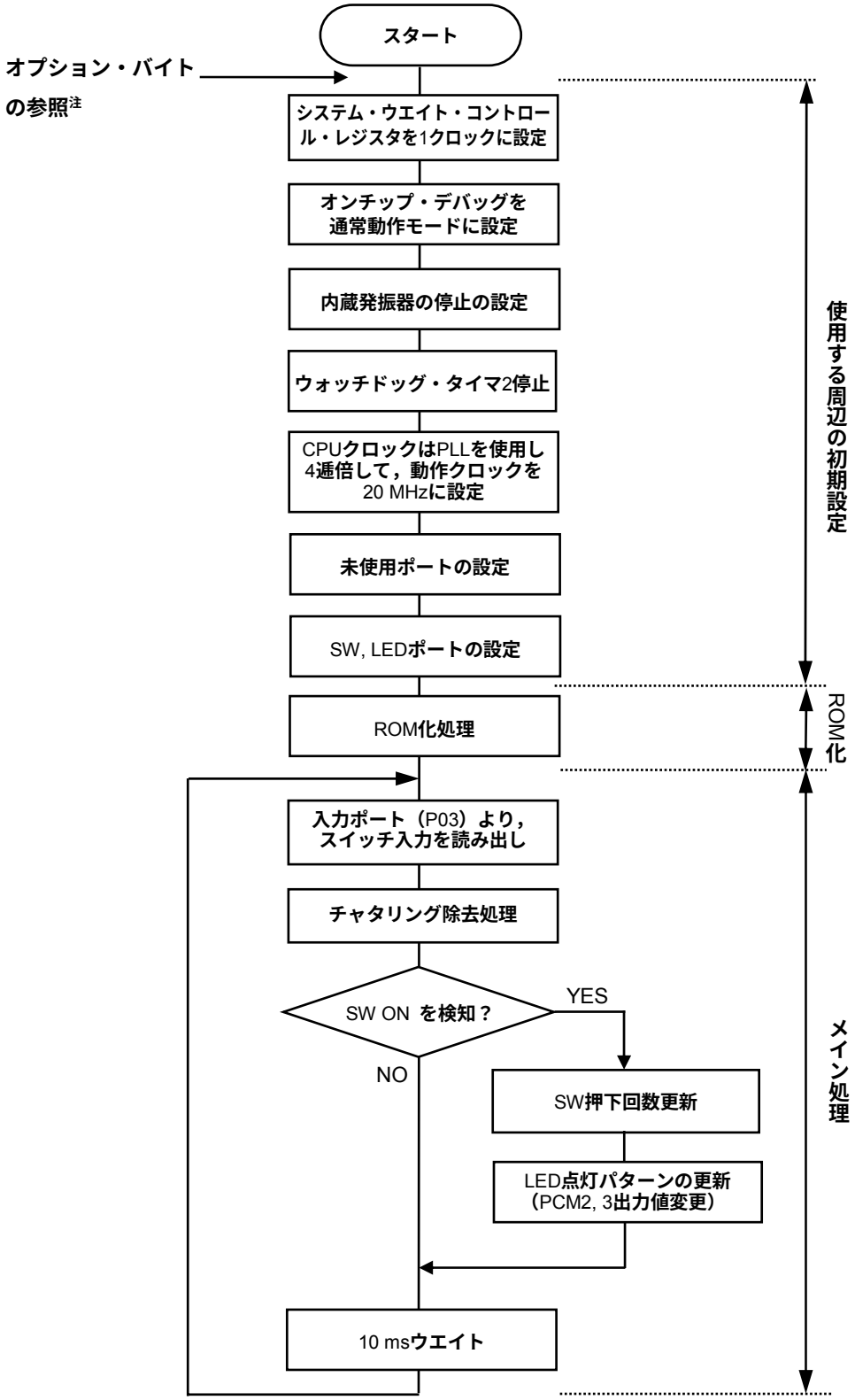
初期設定完了後は，スイッチ入力（SW1）回数に応じて，2つのLED（LED1, LED2）の点灯を制御します。

詳細については，次の状態遷移図に示します。



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・バイトの参照は，リセット解除後にマイコンが自動的に行います。このサンプル・プログラムでは，オプション・バイトでリセット解除後の発振安定時間を6.554 msに設定しています。

3. 5　V850/JG3-Lと V850/JF3-Lの違い

V850ES/JG3-Lは，V850ES/JF3-Lに対して，I/O，タイマ／カウンタ，シリアル・インタフェースなどの機能を拡張したものです。

このサンプル・プログラムにおいては，I/Oの初期化におけるポートの初期化範囲が異なります。

サンプル・プログラムの詳細については，付録A　プログラム・リスト参照してください。

3. 6　ROM化について（C言語のみ）

このサンプル・プログラム（C言語）では，内蔵周辺の初期化後に「ROM化」情報のコピー処理を行っています。

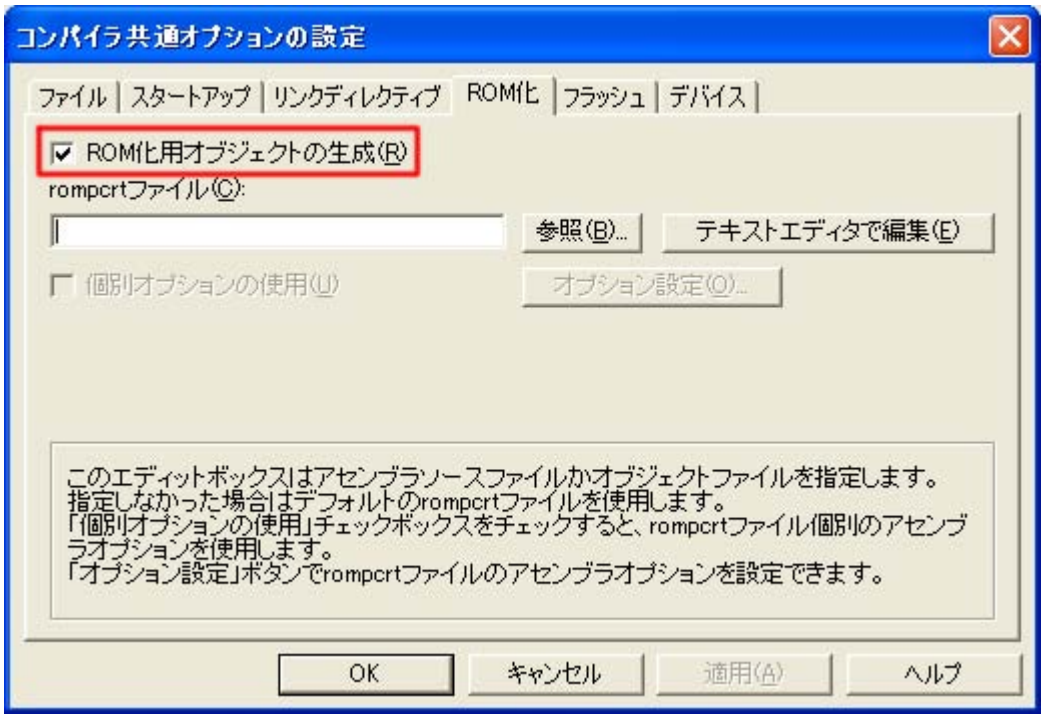
「ROM化」情報とは，初期値のある変数（初期値のある変数が配置されるセクション）の初期値情報のことで，「ROM化」情報をRAMにコピーすることにより，初期値のある変数（初期値のある変数が配置されるセクション）は，初めて初期値を持つことになります[注]。

作成するプログラムで，初期値あり変数を使用している場合は，必ず「ROM化」情報の生成とコピー処理を行う必要があり，コピー処理の実施は，必ずその初期値あり変数を利用する前に行う必要があります。

注　ROM化により，パッキングされる対象となるものは，デフォルトでは「書き込み可能な属性を持つセクションに割り当てられたデータ」です。
その他のデータをパッキングすることも可能です。 詳細はCA850のヘルプを参照してください。

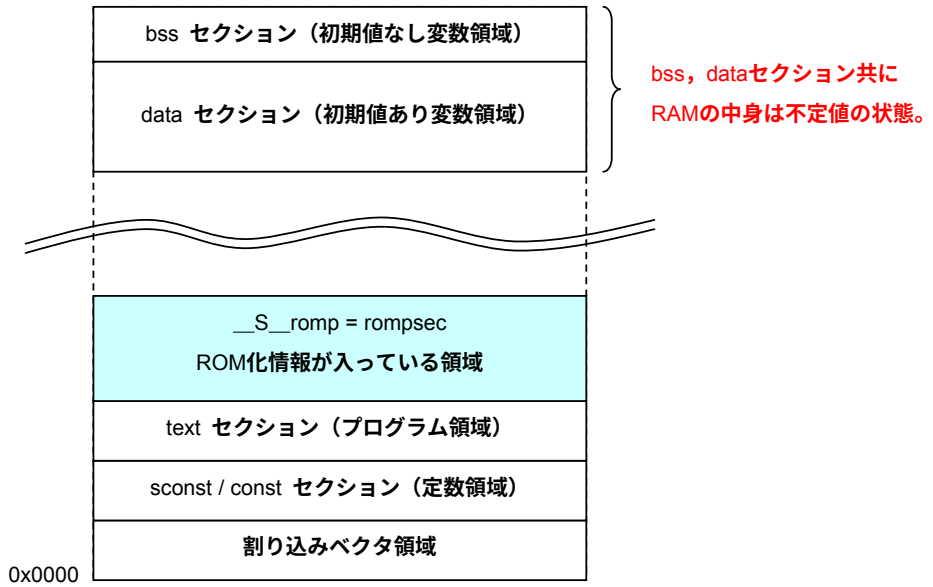
次に，ROM化を行うための手順を説明します。

まず，PM+のコンパイラ共通オプションの〔ROM化〕のタブを選択し，〔ROM化用オブジェクトの生成(R)〕をチェックします。



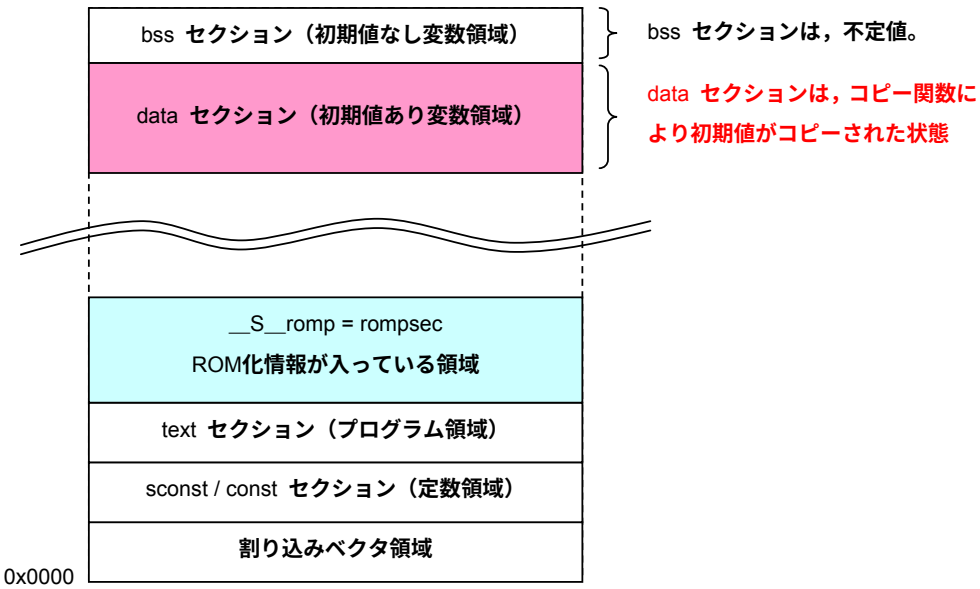
ROM化情報を格納するセクション（rompsec）は，自動でプログラム領域（.text）セクションの直後に追加されますが，このチェックボックス操作によって，デフォルトの「rompcrt.o」で定義されたラベル ”_S_romp” に “rompsec” と同じアドレスを指すコードが生成され，コピー関数が格納されているライブラリ ”libr.a” が自動的にリンクされます。

ここまでの手順で作成された，コピー前のメモリ・イメージは次のようになります。



このままでは，初期値あり変数領域である「data セクション」の中身が不定のままなので，コピー処理を行う必要があります。

ROM化情報のコピーを行うために，_rcopy()関数を呼び出すと，コピー後のメモリ・イメージは次のようになります。



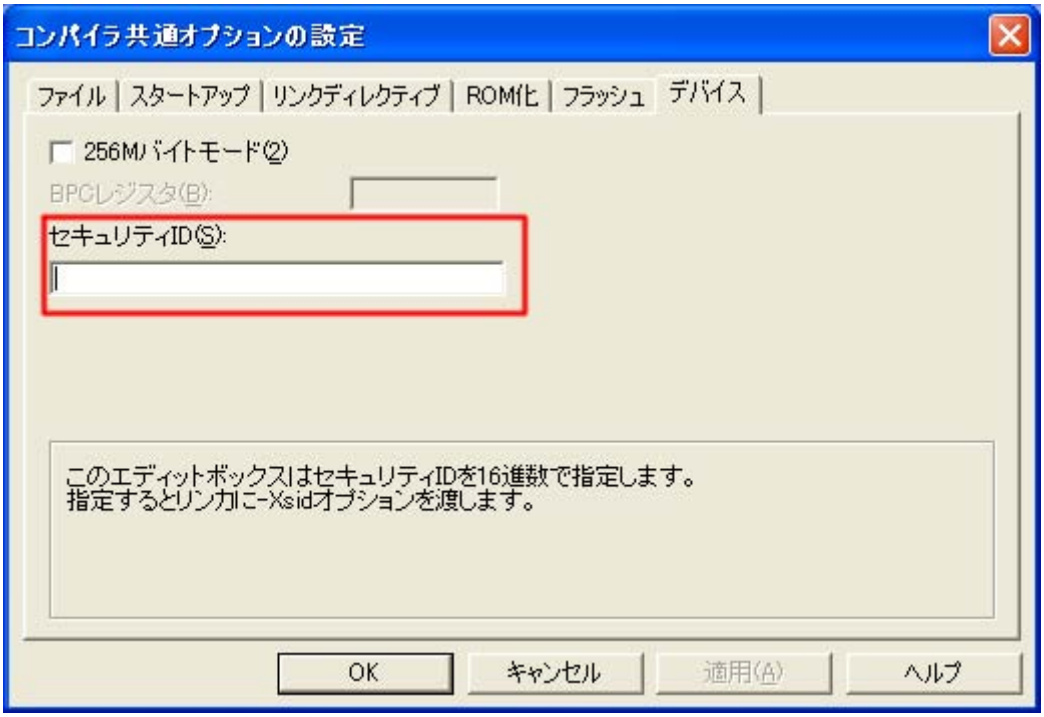
3. 7　セキュリティIDについて

オンチップ・デバッグ・エミュレータによるオンチップ・デバッグ時，フラッシュ・メモリの内容を第三者に読み出されることを防ぐために，10バイトのIDコードによる認証を行います。

IDコードは，あらかじめ内蔵フラッシュ・メモリ領域の0x0000070-0x0000079の10バイト分に設定し，デバッガがID認証を行います。

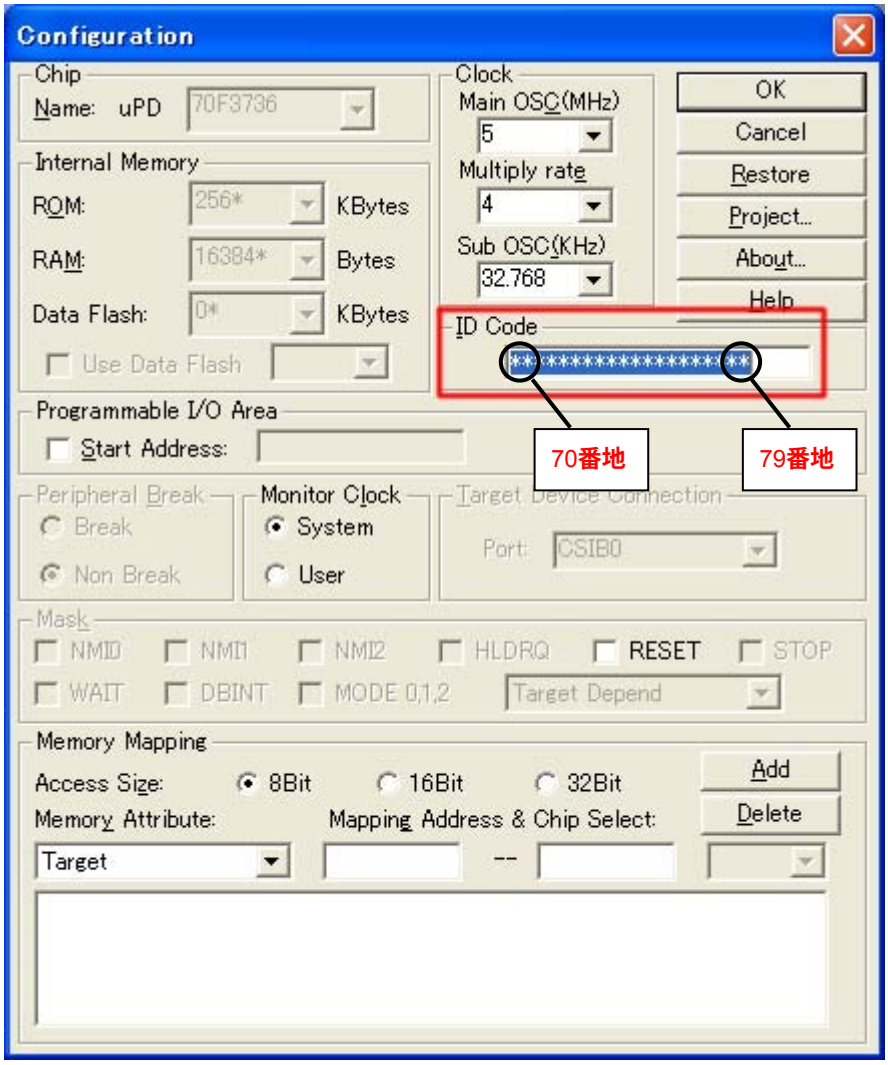
このID照合が一致していれば，セキュリティが解除され，フラッシュ・メモリ読み出し許可，オンチップ・デバッグ・エミュレータ使用許可となります。

このサンプル・プログラム（環境一式版）では，セキュリティIDの設定を行っていないため，デフォルトのセキュリティID値“ 0xFFFF FFFF FFFF FFFF FFFF ”が適用されるようになっています。



備考　コンパイラ共通オプションの設定「セキュリティID」では，フラッシュ・メモリ搭載デバイスの“セキュリティID”を設定します。
IDは，0xで始まる10バイト以内の16進数で指定します。
このオプションの指定，またはアセンブラ記述（.section "SECURITY_ID" 使用）によるセキュリティIDの指定が省略された場合，“ 0xFFFF FFFF FFFF FFFF FFFF ” が指定されたものとして扱います。

このサンプル・プログラム（環境一式版）を使用して，プログラムをダウンロードして動作させた場合，マイコンのセキュリティID領域には，“ 0xFF ”が設定されるため，次回デバッガ接続時のID Code入力エリアに“ FFFF FFFF FFFF FFFF FFFF ”を設定（設定していない場合のデフォルト値）しないと，オンチップ・デバッグ・エミュレータが使用できないため，注意が必要です。



- 10バイトのIDコードのうち，0x0000079のビット7は，オンチップ・デバッグ・エミュレータ使用許可フラグです（0：使用禁止，1：使用許可）。
- オンチップ・デバッグ・エミュレータを起動すると，デバッガがID入力を要求します。
  デバッガ上で入力したIDコードと，0x0000070-0x0000079に埋め込んだIDコードが一致すればデバッガが起動します。
- IDコードが一致しても，オンチップ・デバッグ・エミュレータ使用許可フラグが“0”である場合は，デバッグを行うことはできません。

第4章　レジスタ設定について

この章では，オプション・バイト，システム・ウエイト・コントロール・レジスタ，オンチップ・デバッグ，ウォッチドッグ・タイマ，DMA，内部システム・クロック，PLLモード，端子機能の設定，およびメイン処理について説明します。

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

- V850ES/JG3-L　32ビット･シングルチップ･マイクロコントローラ
  ハードウエア編　ユーザーズ・マニュアル
- V850ES/JF3-L　32ビット･シングルチップ･マイクロコントローラ ハードウエア編
  ハードウエア編　ユーザーズ・マニュアル

C言語，アセンブラ言語の拡張記述の詳細については，次のユーザーズ･マニュアルを参照してください。

- CA850 Cコンパイラ･パッケージ C言語編　ユーザーズ・マニュアル
- CA850 Cコンパイラ･パッケージ アセンブリ言語編 ユーザーズ･マニュアル

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・バイトは，内蔵フラッシュ・メモリの0x000007A番地（内蔵ROM領域）に8ビット・データとして格納しています。この8ピット・データは，リセット解除後の発振安定時間を設定するデータです。リセット解除後は，この設定値により発振安定時間が確保されます。

発振安定時間は，実アプリケーションにおいてV850ES/Jx3-Lと発振子のマッチング評価を行い，発振子の発振安定時間を満たす値以上に設定してください。

このサンプル・プログラムでは，オプション・バイトの設定は，opt_b.sで行っています。

図4－1　オプション・バイトのフォーマット

アドレス：0x0000007A

7	6	5	4	3	2	1	0
0	0	0	0	0	RESOSTS2	RESOSTS1	RESOSTS0

RESOSTS2	RESOSTS1	RESOSTS0	発振安定時間の選択（理論値）			
				fx		
				2.5 MHz	5 MHz	10 MHz
0	0	0	2^{10}/fx	409.6 μs	**設定禁止**	**設定禁止**
0	0	1	2^{11}/fx	819.2 μs	409.6 μs	**設定禁止**
0	1	0	2^{12}/fx	1.638 ms	819.2 μs	409.6 μs
0	1	1	2^{13}/fx	3.277 ms	1.638 ms	819.2 μs
1	0	0	2^{14}/fx	6.554 ms	3.277 ms	1.638 ms
1	**0**	**1**	2^{15}/fx	13.11 ms	**6.554 ms**	3.277 ms
1	1	0	2^{16}/fx	26.21 ms	13.11 ms	6.554 ms
1	1	1	2^{16}/fx	26.21 ms	13.11 ms	6.554 ms

注意　オプション・バイトのセクションには，必ず6バイト分を記述してください。
6バイト未満の場合は，リンクの際にエラーとなります。

備考1.　0x0000007B-0x0000007F番地には 0x00を設定してください。

2.　表の赤字部分がサンプル･プログラムでの設定値となります。

・C言語版／アセンブリ言語版共通

```
.section "OPTION_BYTES"
.byte 0b00000101 -- 0x7a (5MHz:  発振安定時間  6.554ms  に設定)
.byte 0b00000000 -- 0x7b                    ↑
.byte 0b00000000 -- 0x7c                    ↑
.byte 0b00000000 -- 0x7d 0x7b-0x7f番地には0x00を設定する必要がある
.byte 0b00000000 -- 0x7e                    ↓
.byte 0b00000000 -- 0x7f                    ↓
```

4. 2　システム・ウエイト・コントロール・レジスタ（VSWC）設定

VSWCレジスタは，内蔵周辺I/Oレジスタに対するバス・アクセスのウエイトを制御するレジスタです。

内蔵周辺I/Oレジスタへのアクセスは3クロック（ノー・ウエイト時）ですが，V850ES/Jx3-Lでは動作周波数によりウエイトが必要です。使用する動作周波数に応じて，VSWCレジスタには次に示す値を設定してください。

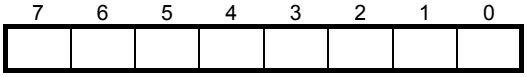
8ビット単位でリード／ライト可能です

リセットにより0x77になります。

図4－2　VSWCレジスタのフォーマット

システム･ウエイト･コントロール･レジスタ（VSWC）

アドレス：0xFFFFF06E



動作周波数（f_{CLK}）	VSWCの設定値	ウエイト数
32 kHz ≦ f_{CLK} ＜ 16.6 MHz	0x00	0（ノー･ウエイト）
16.6 MHz ≦ f_{CLK} ≦ 20 MHz	**0x01**	1

備考　表の赤字部分がサンプル･プログラムでの設定値となります。

VSWCの設定値は「0x01」となります。

・C言語の場合

```
VSWC = 0b00000001;/* 内蔵周辺I/Oレジスタへのアクセス時に1ウエイトかける */
```

・アセンブリ言語の場合

```
mov 0x01, r11        -- 内蔵周辺I/Oレジスタへのアクセス時に1ウエイトかける
st.b r11, VSWC
```

4.3　特定レジスタの設定

初期設定では，オンチップ・デバッグ・モード・レジスタ（OCDM），プロセッサ・クロック・コントロール・レジスタ（PCC）の設定を行います。このレジスタは特定レジスタであり，書き込みを行う場合には特定のシーケンスで行う必要があります。

4.3.1　特定レジスタについて

特定レジスタは，プログラムの暴走などにより不正なデータが書き込まれないよう保護されているレジスタです。V850ES/Jx3-Lには次の7個の特定レジスタがあります。

- パワー・セーブ・コントロール・レジスタ（PSC）
- クロック・コントロール・レジスタ（CKC）
- プロセッサ・クロック・コントロール・レジスタ（PCC）
- クロック・モニタ・モード・レジスタ（CLM）
- リセット要因フラグ・レジスタ（RESF）
- 低電圧検出レジスタ（LVIM）
- オンチップ・デバッグ・モード・レジスタ（OCDM）

また，プログラムの暴走により応用システムが不用意に停止しないように，特定レジスタへの書き込み動作に対するプロテクション・レジスタとしてPRCMDレジスタがあり，特定レジスタへのライト・アクセスは特定のシーケンスで行われ，不正なストア動作はSYSレジスタに報告されます。

4.3.2　特定レジスタへのデータ設定

特定レジスタへのデータ設定は次のシーケンスで行います。

① DMA動作を禁止する。
② 任意の汎用レジスタに特定レジスタへ設定するためのデータを用意する。
③ PRCMDレジスタに②で用意したデータを書き込む。
④ 特定レジスタに設定データを書き込む（次の命令で行う）。
  - ストア命令（ST/SST命令）
  - ビット操作命令（SET1/CLR1/NOT1命令）

  （⑤-⑨ NOP命令を挿入する（5命令）。）（PSC.STPビット=1にする場合のみ）
⑩ DMA動作が必要な場合，DMA動作を許可する。

4.3.3　DMA動作を禁止する

特定レジスタへデータ設定を書き込むために，まずDMA動作を禁止する必要があります。

DMAチャネル・コントロール・レジスタ0-3（DCHC0-DCHC3）でDMAチャネルnのDMA転送の許可／禁止を設定できます。

DMAチャネルnのDMA転送の許可／禁止を設定するために，DCHC.Ennビット（ビット0）を設定してください。

備考　リセット解除後，DMAチャネル・コントロール・レジスタの初期値は0x00であり，DMA動作は禁止の状態になっています。初期設定時など，DMA動作が禁止されていることが明確な場合は，①と⑩の手順は省略できます。サンプル・プログラムでは①の手順は含みません。

図4－3　DCHCnレジスタのフォーマット

DMAチャネル･コントロール･レジスタ0-3（DCHC0-DCHC3）

アドレス：0xFFFFF0E0（DCHC0），0xFFFFF0E2（DCHC1），0xFFFFF0E0（DCHC2），0xFFFFF0E0（DCHC3）

7	6	5	4	3	2	1	0
TCn	0	0	0	0	INITn	STGn	Enn

Enn	DMAチャネルnのDMA転送の許可／禁止の設定
0	DMA転送の禁止
1	DMA転送の許可

リセット解除後，DMA転送は禁止（DCHCn.Ennビット=0）状態のためDMAの停止処理は省略できます。

4. 4　オンチップ・デバッグの通常動作モード設定

通常動作モードとオンチップ・デバッグ・モードを切り替えるレジスタで，オンチップ・デバッグ機能が割り付けられている兼用端子をオンチップ・デバッグ用端子として使用するか，通常のポート／周辺機能兼用端子として使用するかを指定します。また同時に，P05/INTP2/$\overline{\text{DRST}}$端子の内蔵プルダウン抵抗の切断を制御します。

OCDMレジスタは特定レジスタです。特定のシーケンスの組み合わせによってだけ書き込みができます（4. 3. 2 特定レジスタへのデータ設定参照）。

OCDMレジスタへの書き込みは，$\overline{\text{DRST}}$端子にロウ・レベルが入力されているときのみ有効です。

8/1ビット単位でリード／ライト可能です。

$\overline{\text{RESET}}$端子入力により0x01になります。ウォッチドッグ・タイマ，クロック・モニタ，低電圧検出回路によるリセットの場合は，OCDMレジスタの値を保持します。

図4－4　OCDMレジスタのフォーマット

オンチップ･デバッグ･モード･レジスタ（OCDM）

アドレス：0xFFFFF9FC

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	OCDM0

OCDM0	動作モード
0	【MINICUBE2（QB-MINI2）を使用する場合，通常動作モードで使用する場合】 通常動作モード（オンチップ・デバッグ兼用端子をポート／周辺機能端子として使用）かつ， P05/INTP2/$\overline{\text{DRST}}$端子の内蔵プルダウン抵抗を切断
1	【MINICUBE（QB-V850MINI）を使用する場合】 $\overline{\text{DRST}}$端子がロウ・レベルの場合： 通常動作モード（オンチップ・デバッグ兼用端子をポート／周辺機能端子として使用） $\overline{\text{DRST}}$端子がハイ・レベルの場合： オンチップ・デバッグ・モード（オンチップ・デバッグ・モード用端子として使用）

備考　表の赤字部分がサンプル･プログラムでの設定値となります。

OCDM特定レジスタの設定データは「0x00」です。

・C言語の場合

```
                        /* OCDMを「通常動作モード」に指定 */
#pragma asm
    st.b r0, PRCMD
    st.b r0, OCDM
#pragma endasm
```

・アセンブリ言語の場合

```
st.b r0, PRCMD          -- OCDMアクセスのため,PRCMDレジスタにダミー値ストア
st.b r0, OCDM           -- OCDMレジスタ設定（通常動作モードに設定）
```

4. 5　内蔵発振モード・レジスタ（RCM）設定

RCMレジスタは，内蔵発振器の動作モードの設定を行う8ビット・レジスタです。

このサンプル・プログラムでは，ウォッチドッグ・タイマを使用しないため，内蔵発振器を停止します。

8/1ビット単位でリード／ライト可能です。

リセットにより0x00になります。

図4－5　RCMレジスタのフォーマット

内蔵発振モード･レジスタ（RCM）

アドレス：0xFFFFF80C

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	RSTOP

RSTOP	内蔵発振器の発振／停止
0	内蔵発振器発振
1	内蔵発振器停止

備考　表の赤字部分がサンプル･プログラムでの設定値となります。

RCMの設定値は「0x01」となります。

・C言語の場合

```
RSTOP = 1;        /*内蔵発振器停止設定 */
```

・アセンブリ言語の場合

```
set1 RSTOP        -- 内蔵発振器の停止設定
```

4. 6　ウォッチドッグ・タイマ2の設定

WDTM2レジスタは，ウォッチドッグ・タイマ2のオーバフロー時間および動作クロックを設定するレジスタです。

ウォッチドッグ・タイマ2は，リセット解除後に自動的にリセット・モードでスタートします。動作を確定するために，WDTM2レジスタへ書き込みを行ってください。

このサンプル・プログラムでは，暴走検出用として，ウォッチドッグ・タイマは使用しないため，停止します。

8/1ビット単位でリード／ライト可能です。

リセットにより0x67になります。

図4－6　WDTM2レジスタのフォーマット

ウォッチドッグ・タイマ･モード･レジスタ2（WDTM2）

アドレス：0xFFFFF6D0

7	6	5	4	3	2	1	0
0	WDM21	WDM20	WDCS24	WDCS23	WDCS22	WDCS21	WDCS20

WDM21	WDM20	ウォッチドッグ・タイマ2の動作モード選択
0	**0**	**動作停止**
0	1	ノンマスカブル割り込み要求モード（INTWDT2信号を発生）
1	-	リセット･モード（WDT2RES信号を発生）

備考　表の赤字部分がサンプル･プログラムでの設定値となります。

WDTM2の設定値は「0x00」となります。

・C言語の場合

```
WDTM2 = 0b00000000;      /* ウォッチドッグ・タイマ2動作停止設定 */
```

・アセンブリ言語の場合

```
st.b    r0, WDTM2          -- ウォッチドッグ・タイマ2動作停止設定
```

4. 7　クロックの設定

このサンプル・プログラムでは，5 MHzのセラミック発振子／水晶振動子をX1, X2端子に接続し，PLLモードで4逓倍して，20 MHzを内部システム・クロックにする例を示しています。また，サブクロックは使用しません。

4. 7. 1　プロセッサ・クロック・コントロール・レジスタ（PCC）の設定

PCCレジスタは，サブクロックとメイン・クロックの内蔵帰還抵抗の選択，メイン・クロック発振回路の制御，内部システム・クロックの選択ができます。

PCCレジスタは特定レジスタです。特定のシーケンスの組み合わせによってだけ書き込みができます。

8/1ビット単位でリード／ライト可能です。

リセットにより0x03になります。

図4－7　PCCレジスタのフォーマット

プロセッサ･クロック･コントロール･レジスタ（PCC）

アドレス：0xFFFFF828

7	6	5	4	3	2	1	0
FRC	MCK	MFRC	CLS	CK3	CK2	CK1	CK0

FRC	サブクロックの内蔵帰還抵抗の選択
0	使用する（サブクロックを接続する）
1	**使用しない（サブクロックを接続しない）**

MFRC	メイン・クロック内蔵帰還抵抗の選択
0	**使用する（セラミック発振子／水晶振動子使用時）**
1	使用しない（外部クロック使用時）

CK3	CK2	CK1	CK0	クロックの選択（f_{CLK}/f_{CPU}）
0	**0**	**0**	**0**	fxx
0	0	0	1	fxx/2
0	0	1	0	fxx/4
0	0	1	1	fxx/8
0	1	0	0	fxx/16
0	1	0	1	fxx/32
0	1	1	X	設定禁止
1	X	X	X	f_{XT}

備考　表の赤字部分がサンプル・プログラムでの設定値となります。

PCCの設定値は「0x80」となります。

・C言語の場合

```
                                /* クロック分周を「なし」に設定 */
#pragma asm
    push r10
    mov 0x80, r10
    st.b r10, PRCMD
    st.b r10, PCC
    pop r10
#pragma endasm
```

・アセンブリ言語の場合

```
mov 0x80, r10           -- 汎用レジスタに特定レジスタへ設定するためのデータ設定
st.b r10, PRCMD         -- PCCアクセスのため,PRCMDレジスタにダミー値ストア
st.b r10, PCC           -- PCCレジスタ設定，メイン・クロック発振選択（fxx）
```

4.7.2　ロック・レジスタ（LOCKR）

PLLが安定（ロック）したことを確認するフラグです。

8/1ビット単位でリードのみ可能です。

リセットにより0x01になり，リセット解除後の発振安定時間経過後に0x00になります。

図4－8　LOCKRレジスタのフォーマット

ロック・レジスタ（LOCKR）

アドレス：0xFFFFF824

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	LOCK

LOCK	動作モード
0	ロック状態
1	アンロック（ロックしていない）状態
LOCKビットはPLLのロック状態をリアルタイムに反映するものではありません。	

備考　リセット解除後，発振安定時間経過後にロック・レジスタはロック状態（LOCKR = 0x01）になります。初期設定時など，PLLを停止せずにPLLモードに移行する場合は，ロック・レジスタの確認を省略できます。

4. 7. 3　PLLコントロール・レジスタ（PLLCTL）設定

PLLコントロール・レジスタでCPU動作クロックを選択します。

PLLを制御する8ビット・レジスタです。8/1ビット単位でリード／ライト可能です。

リセットにより0x01になります。

図4－9　PLLCTLレジスタのフォーマット

PLLコントロール･レジスタ（PLLCTL）

アドレス：0xFFFFF82C

7	6	5	4	3	2	1	0
0	0	0	0	0	0	SELPLL	PLLON

SELPLL	動作モード
0	クロック･スルー･モード
1	PLLモード
SELPLLビット = 1の設定は，PLLクロック周波数が安定した状態のときのみ可能です。安定していないとき（アンロック中）にSELPLLビットをライトすると“0”がライトされます。	

PLLON	PLL動作停止レジスタ
0	PLL停止
1	PLL動作（PLLを動作開始後，周波数が安定するまで所定のロックアップ時間が必要）

備考　表の赤字部分がサンプル･プログラムでの設定値となります。

使用方法：

- リセット解除後は，PLLは動作（PLLCTL.PLLONビット = 1）していますが，初期設定はクロック・スルー・モード（PLLCTL.SELPLLビット = 0）のため，PLLモード（SELPLLビット = 1）に変更してください。
- PLL停止の状態からPLLを動作させる場合はPLLONビット = 1として，LOCKR.LOCKビット = 0となってからSELPLLビット = 1としてください。PLLを停止させる場合は，最初にクロック・スルー・モード（SELPLLビット = 0）として，8クロック以上後に，PLL停止（PLLONビット = 0）としてください。

・C言語の場合

```
/* CPUの動作クロックPLLモード：fx = 2.5～5 MHz(fxx = 10～20 MHz)選択 */
        PLLON = 1;              /* PLL動作許可に設定 */
        SELPLL = 1;             /* PLLモードに設定 */
```

・アセンブリ言語の場合

```
   -- CPUの動作クロックPLLモード：fx = 2.5～5 MHz(fxx = 10～20 MHz)選択
   set1  PLLON                  -- PLL動作許可に設定
   set1  SELPLL                 -- PLLモードに設定
```

4. 8　ポート設定

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。

	V850ES/JF3-L	V850ES/JG3-L
ポート0	P02-P06	P02-P06
ポート1	P10	P10-P11
ポート3	P30-P35, P38, P39	P30-P39
ポート4	P40-P42	P40-P42
ポート5	P50-P55	P50-P55
ポート7	P70-P77	P70-P711
ポート9	P90, P91, P96-P99, P913-P915	P90-P915
ポートCM	PCM0-PCM3	PCM0-PCM3
ポートCT	PCT0, PCT1, PCT4, PCT6	PCT0, PCT1, PCT4, PCT6
ポートDH	PDH0, PDH1	PDH0-PDH5
ポートDL	PDL0-PDL15	PDL0-PDL15

4. 8. 1　ポートnレジスタ（Pn）

外部とのデータ入出力は，Pnレジスタへの書き込み，および読み出しによって行います。Pnレジスタは，出力データを保持する出力ラッチ，および端子の状態を読み込む回路で構成されています。

Pnレジスタの各ビットは，それぞれポートnの端子1本ずつに対応しており，1ビット単位でリード／ライト可能です。

このサンプル・プログラムでは，ポート0を後述の【例1】，ポートCMを後述の【例2】の内容で設定しています。また，未使用ポート端子は出力ポートに設定しています。

リセットにより0x00になります。

図4－10　Pnレジスタのフォーマット

ポートn レジスタ（Pn）

7	6	5	4	3	2	1	0
Pn7	Pn6	Pn5	Pn4	Pn3	Pn2	Pn1	Pn0

Pnm	出力データの制御（出力モード時）
0	0を出力
1	1を出力

備考　赤太字が未使用ポートに対しての設定値となります。

【コラム】未仕様端子の処理について

ポート端子はリセットにより入力端子に設定されます。したがって，未使用端子の処理は，個別に抵抗を介してVDDまたはGNDに接続することを推奨します。

しかし，抵抗の部品点数の削減のために端子をオープンにして出力設定にすることも認めています。

4.8.2　ポートnモード・レジスタ（PMn）

ポートの入力モード／出力モードを指定します。

PMnレジスタの各ビットは，それぞれポートnの端子1本ずつに対応しており，1ビット単位で指定可能です。

リセットにより0xFFになります。

図4－11　PMnレジスタのフォーマット

ポートn モード･レジスタ（PMn）

7	6	5	4	3	2	1	0
PMn7	PMn6	PMn5	PMn4	PMn3	PMn2	PMn1	PMn0

PMnm	入出力モードの制御
0	**出力モード**
1	入力モード

備考　赤太字が未使用ポートに対しての設定値となります。

4.8.3　ポートnモード・コントロール・レジスタ

ポート・モード／兼用機能モードを指定します。

PMCnレジスタの各ビットは，それぞれのポートnの端子1本ずつに対応しており，1ビット単位で指定可能です。

リセットにより0x00になります。

図4－12　PMCnレジスタのフォーマット

ポートn モード･コントロール･レジスタ（PMCn）

7	6	5	4	3	2	1	0
PMCn7	PMCn6	PMCn5	PMCn4	PMCn3	PMCn2	PMCn1	PMCn0

PMCnm	動作モードの指定
0	**ポート･モード**
1	兼用機能モード

備考　赤太字が未使用ポートに対しての設定値となります。

【コラム】Pnレジスタへの書き込み／読み出しについて

Pnレジスタへの書き込みは，出力ラッチに対しての書き込みとなります。

PMnレジスタ設定によって「入力モード」に指定されている端子については，Pnレジスタへの書き込み値に関わらず，入力端子の状態に影響はありません。

また，出力ラッチに書き込まれた値は，再度出力ラッチに値を書き込まれるまで保持されます。

【例1】・P03を入力ポートに設定

PM0

1	0	0	0	1	0	1	1

P03端子の入出力モードの選択

1	入力モード

PM0の設定値は，「0x8B」となります。

・C言語の場合

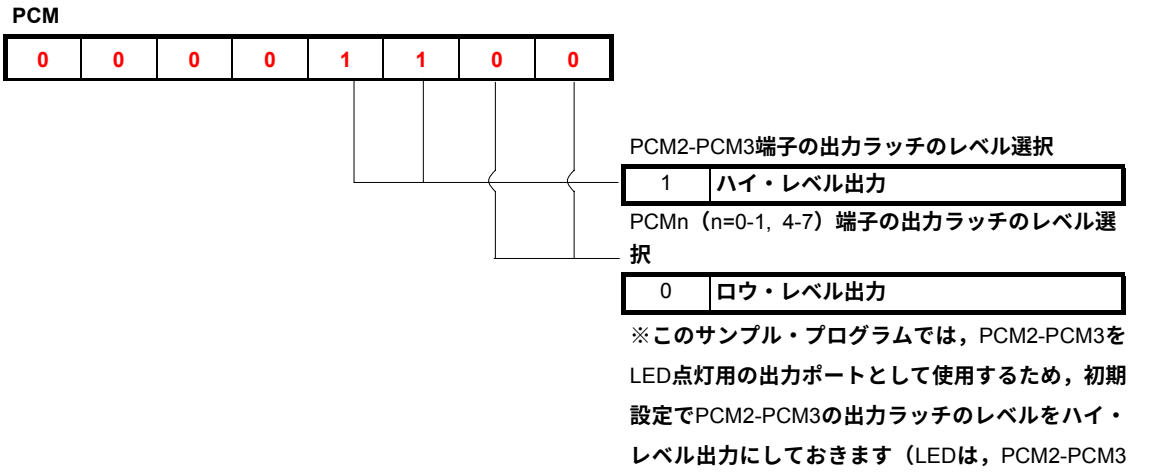
```
PM0 = 0b10001011;        /* P02-P06を出力LOWに設定(P03除) */
```

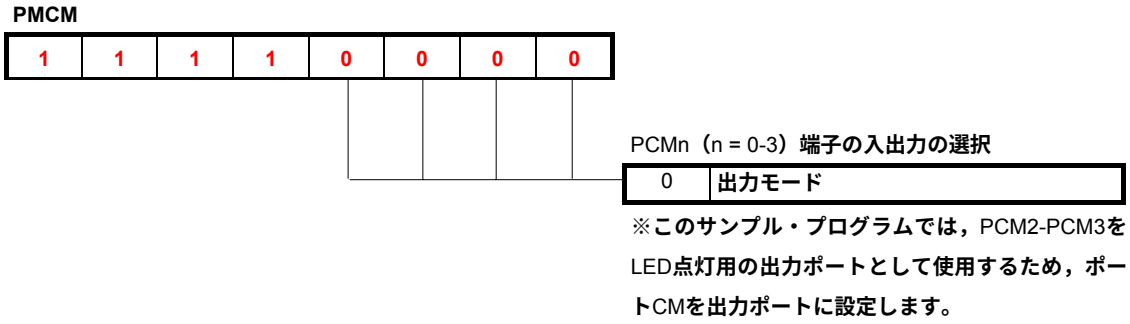
・アセンブリ言語の場合

```
mov 0x8b, r11            -- P02-P06を出力LOWに設定(P03除)
st.b r11, PM0
```

【例2】 ・PCM2, PCM3の出力ラッチをハイ・レベル出力設定

・PCM2, PCM3を出力ポートに設定





PCMの設定値は「0x0C」，PMCMの設定値は「0xF0」となります。

・C言語の場合

```
PCM =  0b00001100;              /* PCM2-PCM3の出力ラッチHigh */
PMCM = 0b11110000;              /* PCM0-PCM3を出力ポートに設定 */
```

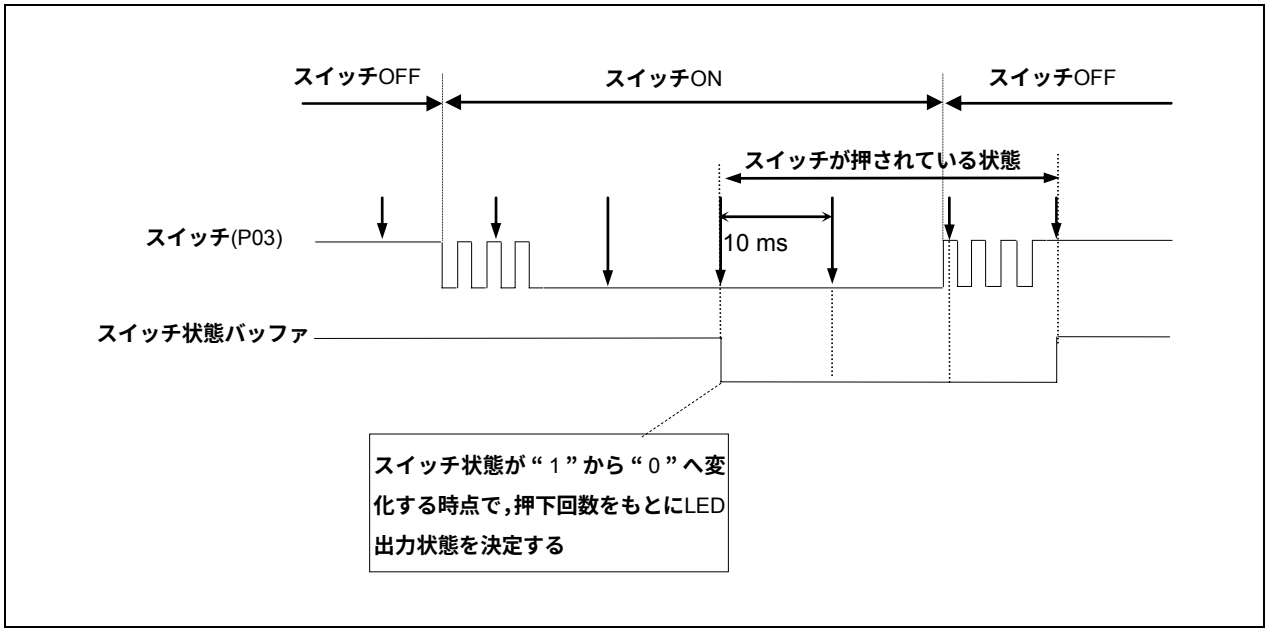
・アセンブリ言語の場合

```
mov 0x0c, r11           -- PCM2-PCM3の出力ラッチHigh
st.b r11, PCM
mov 0xF0, r11           -- PCM0-PCM3を出力ポートに設定
st.b r11, PMCM
```

4.9　メイン処理

4.9.1　チャタリング対策

チャタリングを除去するために，10 msごとに入力を読み込み，2回連続でスイッチの状態に同じレベルを検出したときにスイッチ状態変化を確定します。



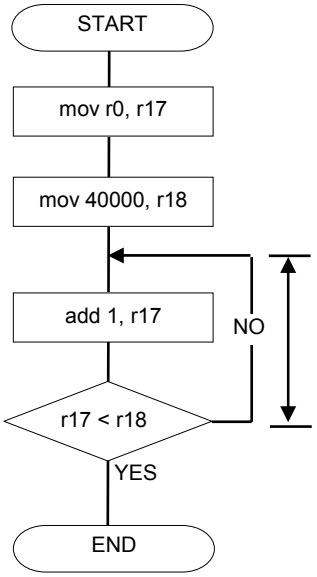
10msウエイトの処理は次の動作を行います。

• C言語の場合

```
unsigned long loop_wait;        /* forループ用カウンタ */

/* 10msウエイト */
for ( loop_wait = 0; loop_wait <= VAL_TIMER_WAIT; loop_wait++ )
{
       __nop();
}
```

・アセンブリ言語の場合



ループ処理を1回実行時，25x10^{-5} ms時間消費されるので，10 msウエイトのために40000回ループさせます。

※命令実行の時間

ニモニック	実行クロック	時間
add	1	5x10^{-5}ms
cmp	1	5x10^{-5}ms
jlt	3	15x10^{-5}ms
合計	5	25x10^{-5}ms

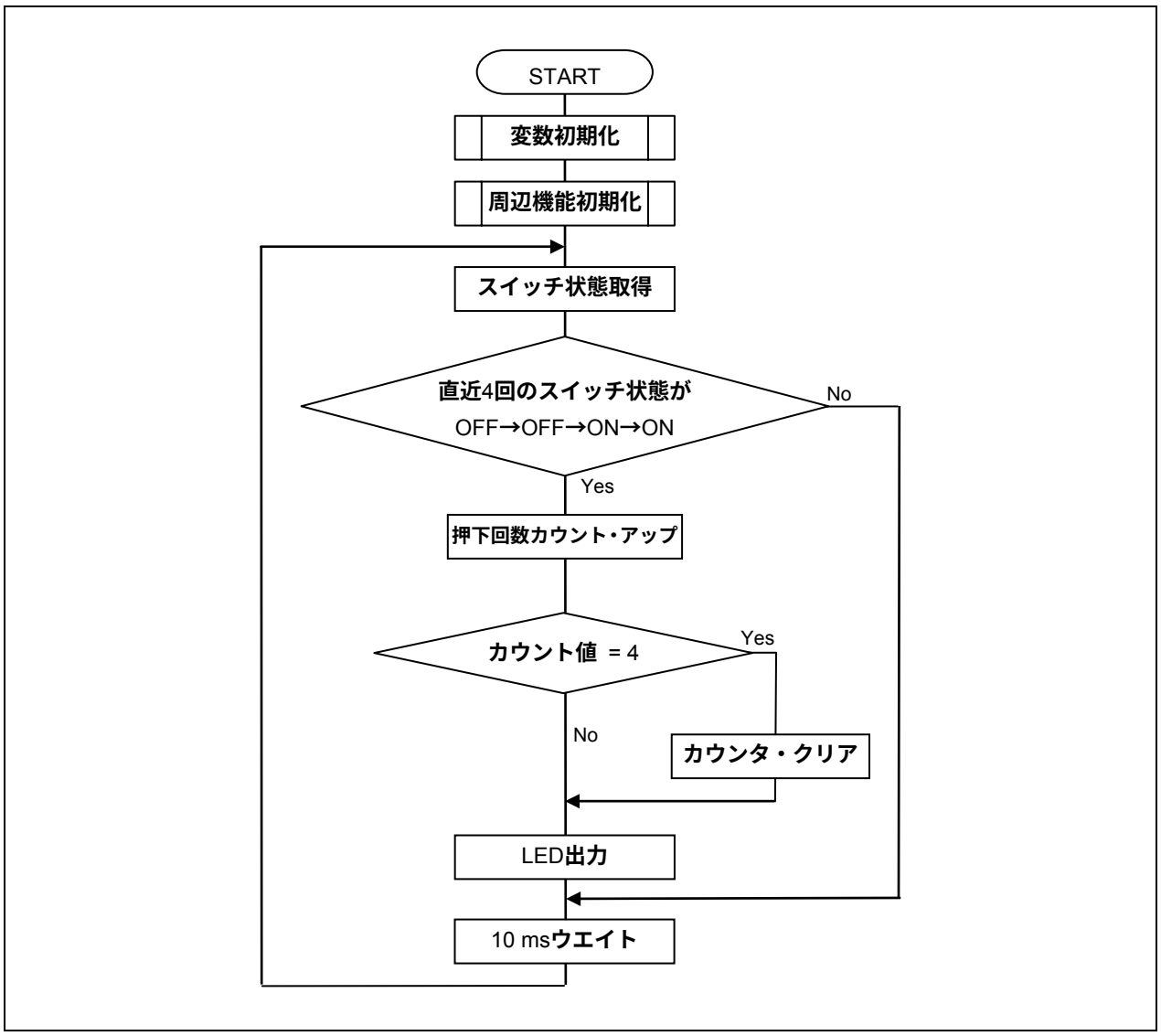
V850ES/Jx3-L ユーザーズ・マニュアルの付録Dを参考してください。

```
.wait10ms:
-- 10msウエイト
        mov     r0, r17
        mov     40000, r18
.not_equal_10ms:
        add     1, r17
        cmp     r18, r17
        jlt     .not_equal_10ms
```

4.9.2　メイン処理

C言語のメイン処理は次の動作を行います。

C言語の場合は，入力データと出力データの対応を，配列で設定します。



```
/************************************************************************
        メイン処理
************************************************************************/
void main( void )
{
    extern unsigned int _S_romp;              /* ROM化シンボルの外部参照      */

    /*--------------------------------------*/
    /* 変数宣言や変数初期設定    */
    /*--------------------------------------*/
    const unsigned char outdata[] = {         /* 表示データパターン用配列            */
        0x0c,                                 /* 全LED消灯                          */
        0x04,                                 /* LED1点灯                           */
        0x00,                                 /* LED1とLED2点灯                     */
        0x08                                  /* LED2点灯                           */
    };
    unsigned char indata = 0b00000001;        /* スイッチ押下状態記憶(前回値OFFで初期化)  */
    unsigned char count;                      /* スイッチ押下回数                    */
    unsigned long loop_wait;                  /* forループ用カウンタ                 */

    count = VAL_RST_COUNT;                    /* スイッチ押下回数初期化              */

    /*--------------------------------------*/
    /* 周辺機能初期化           */
    /*--------------------------------------*/
    f_init_vswc();                            /* VSWCレジスタの設定                  */
    f_init_ocdm();                            /* オンチップ・デバッグを通常動作モードに設定      */
    f_init_rcm();                             /* 内蔵発振器の禁止設定                */
    f_init_wdtm2();                           /* ウォッチドッグ・タイマ2の設定       */
    f_init_lock();                            /* CPUの動作クロックをPLLモードに設定   */
    f_init_blank_port();                      /* 未使用ポート設定                    */
    f_init_use_port();                        /* SW1・LEDポート設定                 */

    /*--------------------------------------*/
    /* ROM化処理             */
    /*--------------------------------------*/
    _rcopy( &_S_romp, -1 );                   /* ROM化処理実施                       */

    /*--------------------------------------*/
    /* LED点灯処理            */
    /*--------------------------------------*/
    while ( 1 )
    {
        indata <<= 1;                         /* スイッチ状態前回値を更新*/
        indata |= P0.3;                       /* スイッチ状態今回値を更新*/
```

{　} 内に4個のデータを定義し，
その中に出力データを設定しています。

```
        if ( ( indata & 0b00001111 ) == 0b00001100 )
        {
            count++;                    /* スイッチ押下回数更新 */
            count &= 0b00000011
            PCM = outdata[count];               /* テーブルから表示データを読み出して表示 */;
        }

        /* 10msウエイト */
        for ( loop_wait = 0; loop_wait <= VAL_TIMER_WAIT; loop_wait++ )
        {
            __nop();
        }
    }

    return;
}
```

入力データと出力データの対応は次のようになります。

スイッチ入力回数	COUNT	OUTDATA	LED点灯
0回	0	0b00001100	全LED消灯
1回	1	0b00000100	LED1のみ点灯
2回	2	0b00000000	LED1/2点灯
3回	3	0b00001000	LED2のみ点灯

アセンブリ言語のメイン処理も，C言語と同様な動作を行います。

① スイッチ状態記憶レジスタ（r28）のbit1に前回スイッチ状態をセットします。
② P03データを読み込んで,スイッチ状態記憶レジスタ(r28)のbit0に最新スイッチ状態をセットします。
③ 直近のスイッチ状態（r28）を0xC（2連続OFF検知後に2連続ON検知状態）と比較します。
④ ③が一致する場合，スイッチ押下回数を基にLED出力を変更して，⑥に進みます。
⑤ ③が一致しない場合，⑥に進みます。
⑥ 10 msウエイトします。
⑦ ①に戻ります。

ROM領域の設定

```
        .data                   -- dataセクション
        .align 4
data_area:
        .byte   0x0C            -- 0回目，4回目     --> 全LED消灯
        .byte   0x04            -- 1回目            --> LED1点灯
        .byte   0x00            -- 2回目            --> LED1とLED2点灯
        .byte   0x08            -- 3回目            --> LED2点灯
```

```
    .globl _main
_main:
    -- 周辺機能初期化
    jarl    _init, lp

    -- 変数初期化
    mov      1, r28              -- indata   bit0:今回値 /bit1:前回値
    mov     r0, r29              -- count    スイッチ押下回数

.main_loop:
    -- LED点灯処理
    -- indata更新
    shl      1, r28              -- indata.1 = indata.0
    tst1     3, P0               -- スイッチ入力値取得
    setfnz  r14
    or       r14, r28            -- indata.0 = P03

.sw_on_chk:
    -- スイッチONチェック
    andi     0xF, r28, r17
    cmp      0xC, r17            -- スイッチONを2連続検知したらスイッチON検知へ
    jne      .wait10ms           -- スイッチON検知しなかった時は，10msウエイトへ
    -- スイッチON検知
    -- count更新
    add      1, r29              -- count = count + 1
    and      3, r29
    -- LED点灯状態更新
    mov      r29, r15
    add      #data_area, r15
    ld.b    [r15], r16           -- r16 = PCMポート出力値
    st.b    r16, PCM             -- PCMポート(LED)に出力
.wait10ms:
    -- 10msウエイト
    mov      r0, r17
    mov      40000, r18
.not_equal_10ms:
    add      1, r17
    cmp      r18, r17
    jlt      .not_equal_10ms

    jr       .main_loop
```

第5章　関連資料

資　料　名	和文／英文
V850ES/JF3-L　**ハードウエア編　ユーザーズ・マニュアル**	PDF
V850ES/JG3-L　**ハードウエア編　ユーザーズ・マニュアル**	PDF
PM+ Ver.6.30 **ユーザーズ・マニュアル**	PDF
CA850 Ver.3.20 C**コンパイラ・パッケージ 操作編**	PDF
CA850 Ver.3.20 C**コンパイラ・パッケージ** C**言語編**	PDF
CA850 Ver.3.20 C**コンパイラ・パッケージ アセンブリ言語編**	PDF
CA850 Ver.3.20 C**コンパイラ・パッケージ リンク・ディレクティブ編**	PDF
V850ES **アーキテクチャ編**	PDF

付録A　プログラム･リスト

プログラム･リスト例として，V850ES/Jx3-L**マイクロコントローラのソース･プログラムを次に示します。**

- opt_b.s　**（アセンブリ言語版・**C**言語版共通）**

```
#-------------------------------------------------------------------------------
#
#      NEC Electronics    V850ES/Jx3-L シリーズ
#
#-------------------------------------------------------------------------------
#      V850ES/JG3-L JF3-L  サンプル・プログラム
#-------------------------------------------------------------------------------
#      LED点灯のスイッチ制御
#-------------------------------------------------------------------------------
#【履歴】
#      2008.7.--   新規作成
#-------------------------------------------------------------------------------
#【概要】
#      本サンプル・プログラムは，オプション・バイトの設定を行う
#-------------------------------------------------------------------------------


      .section "OPTION_BYTES"
      .byte 0b00000101 -- 0x7a (5MHz: 発振安定時間 6.554ms に設定)
      .byte 0b00000000 -- 0x7b               ↑
      .byte 0b00000000 -- 0x7c               ↑
      .byte 0b00000000 -- 0x7d 0x7b-0x7f番地には0x00を設定する必要がある
      .byte 0b00000000 -- 0x7e               ↓
      .byte 0b00000000 -- 0x7f               ↓
```

- minicube2.s　（アセンブリ言語版・C言語版共通）

```
#-------------------------------------------------------------------------------
#
#       NEC Electronics     V850ES/Jx3-L シリーズ
#
#-------------------------------------------------------------------------------
#       V850ES/JG3-L JF3-L  サンプル・プログラム
#-------------------------------------------------------------------------------
#       LED点灯のスイッチ制御
#-------------------------------------------------------------------------------
#【履歴】
#       2008.7.--    新規作成
#-------------------------------------------------------------------------------
#【概要】
#       本サンプル・プログラムは，MINICUBE2使用時に必要なリソースの確保を行っている
#       (CSIB0を用いてMINICUBE2を使用する場合の例)
#-------------------------------------------------------------------------------

        --monitorROMセクションとして2Kバイトの空間を確保
        .section "MonitorROM", const
        .space 0x800, 0xff

        --デバッグ用割り込みベクタの確保
        .section "DBG0"
        .space 4, 0xff

        --シリアル通信用受信割り込みベクタの確保
        .section "INTCB0R"
        .space 4, 0xff

        --MonitorRAMセクションとして16バイトの空間を確保
        .section "MonitorRAM", bss
        .lcomm monitorramsym, 16, 4
```

- AppNote_LED.dir **（アセンブリ言語版・C言語版共通）**

```
#   Sample link directive file (not use RTOS/use internal memory only)
#
#   Copyright (C) NEC Electronics Corporation 2002
#   All rights reserved by NEC Electronics Corporation.
#
#   This is a sample file.
#   NEC Electronics assumes no responsibility for any losses incurred by customers or
#   third parties arising from the use of this file.
#
#   Generated      : PM+ V6.31  [ 9 Jul 2007]
#   Sample Version : E1.00b [12 Jun 2002]
#   Device         : uPD70F3738 (C:¥Program Files¥NEC Electronics Tools¥DEV¥DF3738.800)
#   Internal RAM   : 0x3ffb000 - 0x3ffefff
#
#   NOTICE:
#       Allocation of SCONST, CONST and TEXT depends on the user program.
#
#       If interrupt handler(s) are specified in the user program then
#       the interrupt handler(s) are allocated from address 0 and
#       SCONST, CONST and TEXT are allocated after the interrupt handler(s).

SCONST  : !LOAD ?R {
        .sconst         = $PROGBITS     ?A .sconst;
};

CONST   : !LOAD ?R {
        .const          = $PROGBITS     ?A .const;
};

TEXT    : !LOAD ?RX {
        .pro_epi_runtime = $PROGBITS    ?AX .pro_epi_runtime;
        .text            = $PROGBITS    ?AX .text;
};
```

内蔵ROMサイズが128Kバイト製品の場合は 0x01F800

```
### MINICUBE2用###
MROMSEG : !LOAD ?R V0x03F800{
        MonitorROM = $PROGBITS ?A MonitorROM;
};
```

デフォルトのリンク･ディレクティブファイルとの差分（追加コード）

MINICUBE2用の予約領域を確保。

```
SIDATA  : !LOAD ?RW V0x3ffb000 {
        .tidata.byte    = $PROGBITS     ?AW .tidata.byte;
        .tibss.byte     = $NOBITS       ?AW .tibss.byte;
        .tidata.word    = $PROGBITS     ?AW .tidata.word;
        .tibss.word     = $NOBITS       ?AW .tibss.word;
        .tidata         = $PROGBITS     ?AW .tidata;
        .tibss          = $NOBITS       ?AW .tibss;
        .sidata         = $PROGBITS     ?AW .sidata;
        .sibss          = $NOBITS       ?AW .sibss;
};

DATA    : !LOAD ?RW V0x3ffb100 {
        .data           = $PROGBITS     ?AW  .data;
        .sdata          = $PROGBITS     ?AWG .sdata;
        .sbss           = $NOBITS       ?AWG .sbss;
        .bss            = $NOBITS       ?AW  .bss;
};
```

```
### MINICUBE2用###
MRAMSEG : !LOAD ?RW V0x03FFEFF0{
        MonitorRAM = $NOBITS ?AW MonitorRAM;
};
```

デフォルトのリンク･ディレクティブファイルとの差分（追加コード）

MINICUBE2用の予約領域を確保。

```
__tp_TEXT @ %TP_SYMBOL;
__gp_DATA @ %GP_SYMBOL &__tp_TEXT{DATA};
__ep_DATA @ %EP_SYMBOL;
```

- main.c （C言語版）

```
/*---------------------------------------------------------------------------*/
/*
/*  NEC Electronics    V850ES/Jx3-Lシリーズ
/*
/*---------------------------------------------------------------------------*/
/*  V850ES/JG3-L サンプル・プログラム
/*---------------------------------------------------------------------------*/
/*  LED点灯のスイッチ制御
/*---------------------------------------------------------------------------*/
/*【履歴】
/*  2008.07.--  新規作成
/*---------------------------------------------------------------------------*/
/*【概要】
/*  このサンプル・プログラムでは，クロック周波数の選択，ポート入出力の設定など，
/*  V850ES/JG3-L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，初期設定完了後のメイン処理動作では，1つのスイッチ入力により2つのLED点灯
/*  を制御します。
/*
/*  リセット解除後に動作停止状態の周辺機能について，このサンプル・プログラムで
/*  使用しない周辺機能は，設定していません。
/*
/*
/* ＜初期設定の主な設定内容＞
/*  ・システム・ウエイト・コントロール・レジスタを1クロックに設定
/*  ・オンチップ・デバッグを通常動作モードに設定
/*  ・内蔵発振器の停止設定
/*  ・ウォッチドッグ・タイマ2動作停止
/*  ・システム・クロックはPLLを使用し4逓倍して20Mhzに設定
/*  ・未使用ポートの設定
/*  ・SW入力ポート，LED制御ポートの設定/*
/*
/* ＜メイン処理の主な内容＞
/*  ・スイッチ入力回数を検出
/*  ・LED点灯
/*
```

```
/* ＜スイッチ入力とLED点灯＞
/*
/* +-------------------------------------+
/* |スイッチ押下回数 | LED1     | LED2     |
/* |      (P03)      |  (PCM3)  |  (PCM2)  |
/* |-----------------|----------|----------|
/* |     0回目       |  OFF     |  OFF     |
/* |     1回目       |  ON      |  OFF     |
/* |     2回目       |  ON      |  ON      |
/* |     3回目       |  OFF     |  ON      |
/* +-------------------------------------+
/*      ※4回目以降は0回からの繰り返し
/*
/*
/*【ポート入出力の設定】
/*
/*  ・入力ポート     ：P03
/*  ・出力ポート     ：PCM2, PCM3
/*  ・未使用のポート：P02, P04-P06, P10-P11, P3H0-P3H1, P3L0-P3L7, P40-P42,
/*                    P50-P55, P7H0-P7H3, P7L0-P7L7, P9H0-P9H7, P9L0-P9L7,
/*                    PCM0-PCM1, PCT0,1,4,6, PDH0-PDH5, PDLH0-PDLH7, PDLL0-PDLL7
/*  ※未使用のポートは全て出力ポート(LOW出力)に設定しておく
/*
/*-------------------------------------------------------------------------------*/



/*---------------------------*/
/* pragma指令       */
/*---------------------------*/
#pragma ioreg                           /* 周辺I/Oレジスタ名有効化指定          */



/*---------------------------*/
/* 定数定義         */
/*---------------------------*/
#define VAL_RST_COUNT      (0)          /* スイッチ押下回数初期値               */
#define VAL_TIMER_WAIT     (28700)      /* 10msウェイト                          */
```

```
/*---------------------------*/
/* プロトタイプ宣言   */
/*---------------------------*/
static  void f_init_vswc( void );              /* VSWCレジスタ設定処理                  */
static  void f_init_ocdm( void );              /* オンチップ・デバッグ通常動作モード設定処理
*/
static  void f_init_rcm( void );               /* 内蔵発振器の停止設定                  */
static  void f_init_wdtm2( void );             /* ウォッチドッグ・タイマ2の設定処理   */
static  void f_init_lock( void );              /* CPU動作クロック設定処理               */
static  void f_init_blank_port( void );        /* 未使用ポート設定初期化処理            */
static  void f_init_use_port( void );          /* SW1・LEDポート設定初期化処理          */
       void main( void );                      /* メイン処理                            */


/*****************************/
/* 周辺初期設定        */
/*****************************/
/*---------------------------*/
/* VSWCレジスタ設定 */
/*---------------------------*/
static void f_init_vswc( void )
{
   VSWC = 0b00000001;              /* 内蔵周辺I/Oレジスタへのアクセス時に1ウエイトかける   */

   return;
}


/*-------------------------------------------*/
/* オンチップ・デバッグを通常動作モードに設定 */
/*-------------------------------------------*/
static void f_init_ocdm( void )
{
                                       /* OCDMを「通常動作モード」に指定        */
#pragma asm
   st.b    r0, PRCMD
   st.b    r0, OCDM
#pragma endasm

   return;
}
```

```
/*-------------------------------------*/
/* 内蔵発振モード・レジスタ(RCM)設定 */
/*-------------------------------------*/
static void f_init_rcm( void )
{
    RSTOP = 1;                          /* 内蔵発振器の停止設定                */

    return;
}



/*-------------------------------------*/
/* ウォッチドッグ・タイマ2(WDTM2)設定 */
/*-------------------------------------*/
static void f_init_wdtm2( void )
{
    WDTM2 = 0b00000000;                 /* ウォッチドッグ・タイマ2動作停止設定  */

    return;
}

/*-------------------------------------*/
/* CPUの動作クロックをPLLモードに設定 */
/*-------------------------------------*/
static void f_init_lock( void )
{
    /* PLLモードに設定,PLL動作設定 (PLLCTLレジスタ設定) */
    /* CPUの動作クロックPLLモード：fx = 2.5～5 MHz(fxx = 10～20 MHz)選択 */
    PLLON = 1;                          /* PLL動作許可に設定                   */
    SELPLL = 1;                         /* PLLモードに設定 */

    /* PCCレジスタ設定 */
                                        /* クロック分周を「なし」に設定 */
#pragma asm
    push   r10
    mov    0x80, r10
    st.b    r10, PRCMD
    st.b    r10, PCC
    pop    r10
#pragma endasm

    return;
```

```
}
/*----------------------------*/
/* 未使用ポートの設定 */
/*----------------------------*/
static void f_init_blank_port( void )
{
    P0 =  0b00000000;                   /* P02-P06を出力LOWに設定(P03除)       */
    PM0 = 0b10001011;

    P1 =  0b00000000;                   /* P10-P11を出力LOWに設定              */
    PM1 = 0b11111100;
```

V850ES/JF3-Lでは，P1はP10のみ設定。

```
    P1 =  0b00000000;                   /* P10を出力LOWに設定                  */
    PM1 = 0b11111110;

    P3H =  0b00000000;                  /* P3H0-P3H1を出力LOWに設定            */
    PM3H = 0b11111100;
    P3L =  0b00000000;                  /* P3L0-P3L7を出力LOWに設定            */
    PM3L = 0b00000000;
```

V850ES/JF3-Lでは，P3はP3H0-P3H1, P3L0-P3L5のみ設定。

```
    P3H =  0b00000000;                  /* P3H0-P3H1を出力LOWに設定            */
    PM3H = 0b11111100;
    P3L =  0b00000000;                  /* P3L0-P3L5を出力LOWに設定            */
    PM3L = 0b11000000;

    P4 =  0b00000000;                   /* P40-P42を出力LOWに設定              */
    PM4 = 0b11111000;

    P5 =  0b00000000;                   /* P50-P55を出力LOWに設定              */
    PM5 = 0b11000000;

    P7H =  0b00000000;                  /* P7H0-P7H3を出力LOWに設定            */
    PM7H = 0b11100000;
    P7L =  0b00000000;                  /* P7L0-P7L7を出力LOWに設定            */
    PM7L = 0b00000000;
```

V850JF3-Lでは，P7はP7L0-P7L7のみ設定。

```
    P7L =  0b00000000;                  /* P7L0-P7L7を出力LOWに設定            */
    PM7L = 0b00000000;
```

```
    P9H =  0b00000000;                  /* P9H0-P9H7を出力LOWに設定          */
    PM9H = 0b00000000;
    P9L =  0b00000000;                  /* P9L0-P9L7を出力LOWに設定          */
    PM9L = 0b00000000;
```

V850ES/JF3-Lでは，P9はP9H0,1,5,6,7; P9L0,1,6,7のみ設定。

```
    P9H =  0b00000000;                  /* P9H0-P9H1とP9H5-P9H7を出力LOWに設定  */
    PM9H = 0b00011100;
    P9L =  0b00000000;                  /* P9L0-P9L1とP9L6-P9L7を出力LOWに設定  */
    PM9L = 0b00111100;

    PCM =  0b00000000;                  /* PCM0-PCM1を出力LOWに設定          */
    PMCM = 0b11111100;

    PCT =  0b00000000;                  /* PCT0,1,4,6を出力LOWに設定         */
    PMCT = 0b10101100;

    PDH =  0b00000000;                  /* PDH0-PDH5を出力LOWに設定          */
    PMDH = 0b11000000;
```

V850ES/JF3-Lでは，PDHはPDH0-PDH1のみ設定。

```
    PDH =  0b00000000;                  /* PDH0-PDH1を出力LOWに設定          */
    PMDH = 0b11111100;

    PDLH =  0b00000000;                 /* PDLH0-PDLH7を出力LOWに設定        */
    PMDLH = 0b00000000;
    PDLL =  0b00000000;                 /* PDLL0-PDLL7を出力LOWに設定        */
    PMDLL = 0b00000000;

    return;
}
```

```
/*----------------------------------------*/
/* SW入力ポート，LED制御ポートの設定  */
/*----------------------------------------*
/static void f_init_use_port( void )
{
    /* SW出力ポート設定 */
    P0 =  0b00000000;                   /* P03を入力設定                        */
    PM0 = 0b10001011;

    /* LED出力ポート設定 */
    PCM =  0b00001100;                  /* PCM2-PCM3を出力HIGHに設定            */
    PMCM = 0b11110000;

    return;
}



/*****************************/
/* メインモジュール      */
/*****************************/
void main( void )
{
    extern unsigned int _S_romp;        /* ROM化シンボルの外部参照                  */

    /*----------------------------------------*/
    /* 変数宣言や変数初期設定    */
    /*----------------------------------------*/
    const unsigned char outdata[] = {   /* 表示データパターン用配列                 */
        0x0c,                           /* 全LED消灯                                */
        0x04,                           /* LED1点灯                                 */
        0x00,                           /* LED1とLED2点灯                           */
        0x08                            /* LED2点灯                                 */
    };
    unsigned char indata = 0b00000001;  /* スイッチ押下状態記憶(前回値OFFで初期化)   */
    unsigned char count;                /* スイッチ押下回数                         */
    unsigned long loop_wait;            /* forループ用カウンタ                      */

    count = VAL_RST_COUNT;              /* スイッチ押下回数初期化                   */
```

```
    /*---------------------------------------*/
    /* 周辺機能初期化            */
    /*---------------------------------------*/
    f_init_vswc();                      /* VSWCレジスタの設定                    */
    f_init_ocdm();                      /* オンチップ・デバッグを通常動作モードに設定 */
    f_init_rcm();                       /* 内蔵発振器の禁止設定                  */
    f_init_wdtm2();                     /* ウォッチドッグ・タイマ2の設定         */
    f_init_lock();                      /* CPUの動作クロックをPLLモードに設定    */
    f_init_blank_port();                /* 未使用ポート設定                      */
    f_init_use_port();                  /* SW入力ポート，LED制御ポートの設定     */

    /*---------------------------------------*/
    /* ROM化処理             */
    /*---------------------------------------*/
    _rcopy( &_S_romp, -1 );             /* ROM化処理実施                         */

    /*---------------------------------------*/
    /* ＬＥＤ点灯処理            */
    /*---------------------------------------*/
    while ( 1 )
    {
        indata <<= 1;                   /* スイッチ状態前回値を更新              */
        indata |= P0.3;                 /* スイッチ状態今回値を更新              */
        if ( ( indata & 0b00001111 ) == 0b00001100 )
        {
            count++;                    /* スイッチ押下回数更新                  */
            count &= 0b00000011;
            PCM = outdata[count];       /* テーブルから表示データを読み出して表示 */
        }

        /* 10msウエイト */
        for ( loop_wait = 0; loop_wait <= VAL_TIMER_WAIT; loop_wait++ )
        {
            __nop();
        }
    }

    return;
}
```

- main.s　（アセンブリ言語版）

```
#-------------------------------------------------------------------------------
#
#   NEC Electronics     V850ES/Jx3-Lシリーズ
#
#-------------------------------------------------------------------------------
#   V850ES/JG3-L     サンプル・プログラム
#-------------------------------------------------------------------------------
#   LED点灯のスイッチ制御
#-------------------------------------------------------------------------------
#【履歴】
#   2008.07.--  新規作成
#-------------------------------------------------------------------------------
#【概要】
#    このサンプル・プログラムでは，クロック周波数の選択，ポート入出力の設定など，
#    V850ES/JG3-L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，初期設定完了後のメイン処理動作では，1つのスイッチ入力により，2つのLED
#    点灯を制御します。
#
#    リセット解除後に動作停止状態の周辺機能について，このサンプル・プログラムで
#    使用しない周辺機能は，設定していません。
#
# ＜初期設定の主な設定内容＞
#     ・システム・ウエイト・コントロール・レジスタを1クロックに設定
#     ・オンチップ・デバッグを通常動作モードに設定
#     ・内蔵発振器の禁止設定
#     ・ウォッチドッグ・タイマ2動作停止
#     ・システム・クロックはPLLを使用し4逓倍して20Mhzに設定
#     ・未使用ポートの設定
#     ・SW入力ポート，LED制御ポートの設定
#
# ＜メイン処理の主な内容＞
#     ・スイッチ入力回数を検出
#     ・LED点灯
#
```

```
# ＜スイッチ入力とLED点灯＞
#
/* +-------------------------------------+
/* |スイッチ押下回数 | LED1     | LED2     |
/* |      (P03)     |  (PCM3)  |  (PCM2)  |
/* |----------------|----------|----------|
/* |     0回目      |  OFF     |  OFF     |
/* |     1回目      |  ON      |  OFF     |
/* |     2回目      |  ON      |  ON      |
/* |     3回目      |  OFF     |  ON      |
/* +-------------------------------------+
#     ※4回目以降は0回からの繰り返し
#
#【ポート入出力の設定】
#   ・入力ポート     ：P03
#   ・出力ポート     ：PCM2, PCM3
#   ・未使用のポート：P02, P04-P06, P10-P11, P3H0-P3H1, P3L0-P3L7, P40-P42,
#                     P50-P55, P7H0-P7H3, P7L0-P7L7, P9H0-P9H7, P9L0-P9L7,
#                     PCM0-PCM1, PCT0,1,4,6, PDH0-PDH5, PDLH0-PDLH7, PDLL0-PDLL7
#   ※未使用のポートは全て出力ポート(LOW出力)に設定しておく
#
#-------------------------------------------------------------------------------



#-------------------------------------------------------------------------------#
# 表示データパターン用配列, ROMの定義                #
#-------------------------------------------------------------------------------#
.data                       -- dataセクション
    .align 4
data_area:
    .byte 0x0c              -- 0回目，4回目 --> 全LED消灯
    .byte 0x04              -- 1回目    --> LED1点灯
    .byte 0x00              -- 2回目    --> LED1とLED2点灯
    .byte 0x08              -- 3回目    --> LED2点灯



#-------------------------------------------------------------------------------#
# 使用する周辺の初期設定                             #
#-------------------------------------------------------------------------------#
.text
_init:
```

```
#---------------------------------------#
# VSWCレジスタ設定          #
#---------------------------------------#
    mov    0x01, r11              -- 内蔵周辺I/Oレジスタへのアクセス時に1ウエイトかける
    st.b   r11, VSWC


#-------------------------------------------#
# オンチップ・デバッグを通常動作モードに設定 #
#-------------------------------------------#
    st.b   r0, PRCMD              -- OCDMアクセスのため,PRCMDレジスタにダミー値ストア
    st.b   r0, OCDM               -- OCDMレジスタ設定（通常動作モードに設定）


#---------------------------------------#
# 内蔵発振モード・レジスタ(RCM)設定 #
#---------------------------------------#
    set1   RSTOP                  -- 内蔵発振器の停止設定


#---------------------------------------#
# ウォッチドッグ・タイマ2 (WDTM2) 設定 #
#---------------------------------------#
    st.b   r0, WDTM2              -- ウォッチドッグ・タイマ2動作停止設定


#---------------------------------------#
# CPUの動作クロックをPLLモードに設定 #
#---------------------------------------#
    --【PLLモードに設定，PLL動作設定 (PLLCTLレジスタ設定)】
    -- CPUの動作クロックPLLモード：fx = 2.5〜5 MHz(fxx = 10〜20 MHz)選択
    set1   PLLON                  -- PLL動作許可に設定
    set1   SELPLL                 -- PLLモードに設定

--【PCCレジスタ設定】
    mov    0x80, r10              -- 汎用レジスタに特定レジスタへ設定するためのデータ設定
    st.b   r10, PRCMD             -- PCCアクセスのため，PRCMDレジスタにダミー値ストア
    st.b   r10, PCC               -- PCCレジスタ設定，メイン・クロック発振選択（fxx）


#---------------------------------------#
# 未使用ポートの設定        #
#---------------------------------------#
```

```
--【ポート0の設定】
    mov    0x00, r11            -- P02-P06を出力LOWに設定(P03除)
    st.b    r11, P0
    mov    0x8b, r11
    st.b    r11, PM0

--【ポート1の設定】
    mov    0x00, r11            -- P10-P11を出力LOWに設定
    st.b    r11, P1
    mov    0xfc, r11
    st.b    r11, PM1
```

V850ES/JF3-Lでは，P1はP10のみ設定。

```
--【ポート1の設定】
    mov    0x00, r11            -- P10を出力LOWに設定
    st.b    r11, P1
    mov    0xfc, r11
    st.b    r11, PM1
```

```
--【ポート3の設定】
    mov    0x00, r11            -- P3H0-P3H1を出力LOWに設定
    st.b    r11, P3H
    mov    0xfc, r11
    st.b    r11, PM3H

    mov    0x00, r11            -- P3L0-P3L7を出力LOWに設定
    st.b    r11, P3L
    mov    0x00, r11
    st.b    r11, PM3L
```

V850ES/JF3-Lでは，P3はP3H0-P3H1, P3L0-P3L5のみ設定。

```
--【ポート3の設定】
    mov    0x00, r11            -- P3H0-P3H1を出力LOWに設定
    st.b    r11, P3H
    mov    0xfc, r11
    st.b    r11, PM3H

    mov    0x00, r11            -- P3L0-P3L5を出力LOWに設定
    st.b    r11, P3L
    mov    0xc0, r11
    st.b    r11, PM3L
```

```
--【ポート4の設定】
    mov    0x00, r11              -- P40-P42を出力LOWに設定
    st.b    r11, P4
    mov    0xfc, r11
    st.b    r11, PM4

--【ポート5の設定】
    mov    0x00, r11              -- P50-P55を出力LOWに設定
    st.b    r11, P5
    mov    0xc0, r11
    st.b    r11, PM5

--【ポート7の設定】
    mov    0x00, r11              -- P7H0-P7H3を出力LOWに設定
    st.b    r11, P7H
    mov    0xf0, r11
    st.b    r11, PM7H

    mov    0x00, r11              -- P7L0-P7L7を出力LOWに設定
    st.b    r11, P7L
    mov    0x00, r11
    st.b    r11, PM7L
```

V850ESJF3-Lでは，P7はP7L0-P7L7のみ設定。

```
--【ポート7の設定】
    mov    0x00, r11              -- P7L0-P7L7を出力LOWに設定
    st.b    r11, P7L
    mov    0x00, r11
    st.b    r11, PM7L
```

```
--【ポート9の設定】
    mov    0x00, r11            -- P9H0-P9H7を出力LOWに設定
    st.b    r11, P9H
    mov    0x00, r11
    st.b    r11, PM9H

    mov    0x00, r11            -- P9L0-P9L7を出力LOWに設定
    st.b    r11, P9L
    mov    0x00, r11
    st.b    r11, PM9L
```

V850ES/JF3-Lでは，P9はP9H0,1,5,6,7; P9L0,1,6,7のみ設定。

```
--【ポート9の設定】
    mov    0x00, r11            -- P9H0-P9H1とP9H5-P9H7を出力LOWに設定
    st.b    r11, P9H
    mov    0x1c, r11
    st.b    r11, PM9H

    mov    0x00, r11            -- P9L0-P9L1とP9L6-P9L7を出力LOWに設定
    st.b    r11, P9L
    mov    0x3c, r11
    st.b    r11, PM9L
```

```
--【ポートCMの設定】
    mov    0x00, r11            -- PCM0-PCM1を出力LOWに設定
    st.b    r11, PCM
    mov    0xfc, r11
    st.b    r11, PMCM

--【ポートCTの設定】
    mov    0x00, r11            -- PCT0-PCT1, PCT4, PCT6を出力LOWに設定
    st.b    r11, PCT
    mov    0xac, r11
    st.b    r11, PMCT
```

```
--【ポートDHの設定】
    mov    0x00, r11              -- PDH0-PDH5を出力LOWに設定
    st.b    r11, PDH
    mov    0xc0, r11
    st.b    r11, PMDH
```

V850ES/JF3-Lでは，PDHはPDH0-PDH1のみ設定。

```
--【ポートDHの設定】
    mov    0x00, r11              -- PDH0-PDH1を出力LOWに設定
    st.b    r11, PDH
    mov    0xfc, r11
    st.b    r11, PMDH
```

```
--【ポートDLの設定】
    mov    0x00, r11              -- PDLH0-PDLH7を出力LOWに設定
    st.b    r11, PDLH
    mov    0x00, r11
    st.b    r11, PMDLH

    mov    0x00, r11              -- PDLL0-PDLL7を出力LOWに設定
    st.b    r11, PDLL
    mov    0x00, r11
    st.b    r11, PMDLL


#---------------------------------------#
# SW入力ポート，ＬＥＤ制御ポートの設定 #
#---------------------------------------#

--【ポート0の設定】
    mov    0x00, r11              -- P03を入力設定
    st.b    r11, P0
    mov    0x8b, r11
    st.b    r11, PM0


--【ポートCMの設定】
    mov    0x0c, r11              -- PCM2-PCM3を出力HIGHに設定
    st.b    r11, PCM
    mov    0xf0, r11
    st.b    r11, PMCM


    jmp [lp]
```

```
#-------------------------------------------------------------------------------#
# メイン処理                                                                    #
#-------------------------------------------------------------------------------#
    .globl _main
_main:
    -- 周辺機能初期化
    jarl    _init, lp

    -- 変数初期化
    mov    1, r28           -- indata bit0:今回値 /bit1:前回値/bit2:2回前の値/bit3:3回前の値
    mov    r0, r29          -- count  スイッチ押下回数


.main_loop:
    -- ＬＥＤ点灯処理
    -- indata更新
    shl     1, r28           -- indata.1 = indata.0
    tst1    3, P0            -- スイッチ入力値取得
    setfnz  r14
    or      r14, r28         -- indata.0 = P03


.sw_on_chk:
    -- スイッチONチェック
    andi    0xF, r28, r17
    cmp     0xC, r17         -- スイッチONを2連続検知したらスイッチON検知へ
    jne     .wait10ms        -- スイッチON検知しなかった時は，10msウエイトへ
    -- スイッチON検知
    -- count更新
    add     1, r29              -- count = count + 1
    and     3, r29
    -- LED点灯状態更新
    mov     r29, r15
    add     #data_area, r15
    ld.b    [r15], r16          -- r16 = PCMポート出力値
    st.b    r16, PCM            -- PCMポート(LED)に出力
.wait10ms:
    -- 10msウエイト
    mov     r0, r17
    mov     40000, r18
.not_equal_10ms:
    add     1, r17
    cmp     r18, r17
    jlt      .not_equal_10ms

    jr  .main_loop
```

【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

お問い合わせ先

【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/

【営業関係，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

【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。