V850 マイクロコントローラ

V850ES/Jx3-L

リアルタイム・カウンタ（RTC バックアップ・モード）編

R01AN0193JJ0100
Rev.1.00
2010.09.30

要旨

この資料は，サンプル・プログラムの動作概要や使用方法，およびリアルタイム・カウンタの設定方法や活用方法を説明したものです。サンプル・プログラムでは，RTC バックアップ・モードを使用することで，電源電圧（V_{DD}）が供給停止になった場合でも，リアルタイム・カウンタを使用した時計の動作が継続することを示します。

対象デバイス

- μPD70F3792
- μPD70F3793
- μPD70F3794
- μPD70F3795
- μPD70F3796

目次

第1章　概　要 … 3
1. 1　サンプル・プログラムの概要 … 4
1. 2　時計表示について … 7
第2章　回路図 … 12
2. 1　回路図 … 12
2. 2　マイコン以外の使用デバイス … 14
第3章　ソフトウエアについて … 15
3. 1　ファイル構成 … 15
3. 2　使用する内蔵周辺機能 … 16
3. 3　初期設定と動作概要 … 16
3. 4　フロー・チャート … 18
第4章　設定方法について … 25
4. 1　リアルタイム・カウンタの設定 … 26
4. 2　RTCバックアップ・モードの設定 … 44
4. 3　時計誤差補正について … 50
第5章　関連資料 … 55
付録A　プログラム・リスト … 56
付録B　改版履歴 … 97

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. 未使用端子の処理

【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。

CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。

2. 電源投入時の処置

【注意】電源投入時は，製品の状態は不定です。

電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。

同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. リザーブアドレスのアクセス禁止

【注意】リザーブアドレスのアクセスを禁止します。

アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

4. クロックについて

【注意】リセット時は、クロックが安定した後、リセットを解除してください。

プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。

リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

5. 製品間の相違について

【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認下さい。

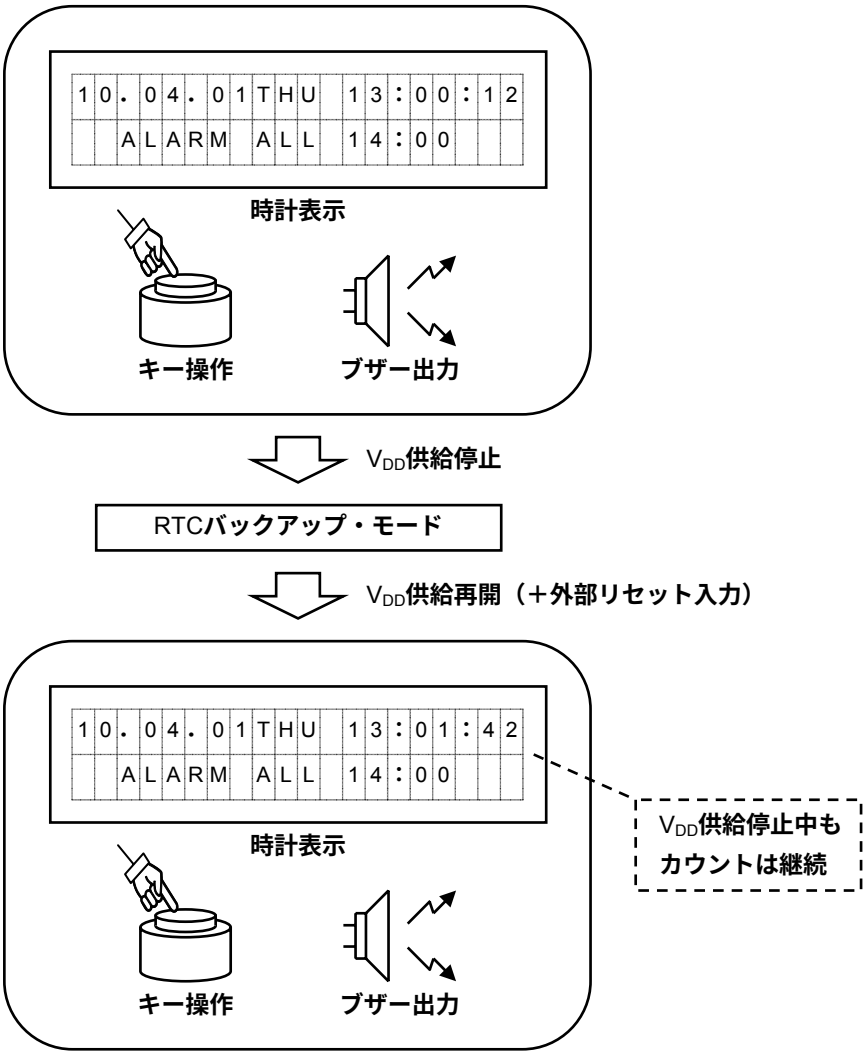
同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。

第1章　概　　要

本資料は，V850ES/JG3-L（μPD70F3792，μPD70F3793，μPD70F3794，μPD70F3795，μPD70F3796）のリアルタイム・カウンタについての資料です。リアルタイム・カウンタの時計機能，アラーム機能，時計誤差補正機能，RTCバックアップ・モードなどについて説明します。

サンプル・プログラムでは，時計機能やアラーム機能，RTCバックアップ・モードなどを使用し，電源電圧（V_{DD}）が供給停止となってもカウントが継続するアラーム付き時計の動作を実現します。

【 動作概要 】



備考1．RTCバックアップ・モードでは，サブクロック発振，およびリアルタイム・カウンタのカウントはRTCバックアップ電源（RV_{DD}）を電源として動作します。

2．このサンプル・プログラムでは，電源電圧（V_{DD}）の供給が停止した場合にもRTCバックアップ電源（RV_{DD}）への電源供給が一定期間は継続できるよう，電気二重層キャパシタ（0.1F）を使用することを想定しています。なお，回路についての詳細は　2. 1　回路図　を参照してください。

1. 1 サンプル・プログラムの概要

このサンプル・プログラムでは，初期設定にてリアルタイム・カウンタの各機能（定周期割り込み機能，アラーム割り込み機能，インターバル割り込み機能）やRTCバックアップ・モード，キー割り込み機能などの設定を行います。

詳細は次のとおりです。

（1）初期設定の主な内容

初期設定の主な内容は，次のとおりです。

＜オプション・バイトでの指定＞

●リセット解除後の発振安定時間を指定

＜リセット解除後の初期化処理での設定＞

●RTCバックアップ・モードの初期設定

・サブクロックの発振モードを通常発振に設定

・RTCバックアップ・モードの使用を許可

●システム・ウエイト・コントロール・レジスタを1ウエイトに設定

●オンチップ・デバッグ・モード・レジスタ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

●入出力ポートの設定

・P50-P53（KR0-KR3）をキー入力用に設定

・P70をブザー制御用に設定

・P90-P96をLCDモジュール制御用に設定

・未使用ポートの設定

●低電圧検出回路[注1]の設定

・低電圧検出レベルを2.8Vに設定

・低電圧検出動作の許可

・電源電圧が2.8V以上になるまで待つ[注2]

●PLLモード（5 MHz×4逓倍 = 20 MHz動作）に設定

●スタンバイ・モード解除時のCPUクロックの発振安定時間を約1.6msに設定

●リアルタイム・カウンタの設定[注3]

・カウント開始時刻を 2010年4月1日（木）13時00分00秒 に設定

・アラーム時刻を 毎日14時00分 に設定

・定周期割り込み動作を0.5秒周期に設定

・アラーム割り込み動作を許可

・インターバル動作を約3.9msに設定

・カウント動作の開始

●LCDモジュールの初期設定

●キー割り込み（KR0-KR3）の動作許可

●割り込みの設定

・INTKR割り込みの許可

・INTRTC0割り込みの許可

・INTRTC1割り込みの許可

・INTLVI割り込みの許可

・マスカブル割り込み要求信号の受け付けを許可

注1. 低電圧検出回路についての詳細は，ユーザーズ・マニュアルを参照してください。

2. ここでの動作条件（電源電圧2.7V以上）を満たすためのウエイトです。

3. リセット解除後，すでにリアルタイム・カウンタが動作している場合は設定を省略します。

（2）初期設定以降の処理内容

初期設定完了後，STOPモードに移行します。以降は，割り込みの発生により処理を実行します。

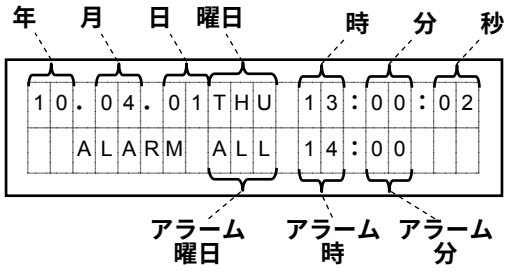
- 0.5秒周期のINTRTC0割り込みが発生した場合，時計表示の更新などを行います。
- アラーム発生によるINTRTC1割り込みが発生した場合，ブザー出力を行います。
- キー入力によるINTKR割り込みが発生した場合，約3.9ms周期のINTRTC2割り込みを使用したキーのサンプリングを開始します。サンプリングにより有効なキー入力が検出された場合，時計表示の更新やリアルタイム・カウンタのカウント値変更などを行います。
- 低電圧検出によるINTLVI割り込みが発生した場合，RTCバックアップ準備状態の設定を行い，STOPモードへ移行します。その後，電源電圧（V_{DD}）が供給停止となった場合，RTCバックアップ・モードとなります。また，STOPモード移行後に低電圧からの復帰によるINTLVI割り込みが発生した場合，RTCバックアップ準備状態を解除し，通常動作に戻ります。

1.2 時計表示について

このサンプル・プログラムでは，LCDモジュール（キャラクタ表示，20文字×2行）を使用し，時計表示を行います。

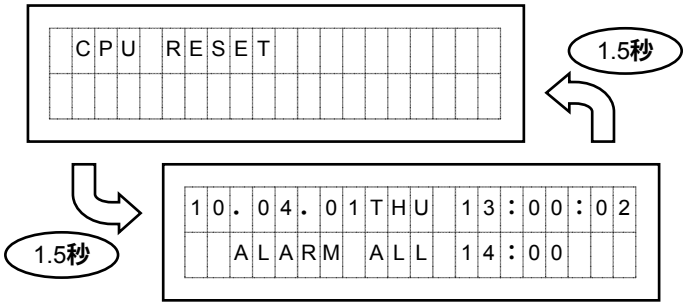
（1）表示内容について

表示内容は日付（年／月／日／曜日），時刻（時／分／秒），およびアラーム設定（曜日／時／分）です。



備考1．現在の日付・時刻がアラーム設定と一致した場合，ブザー出力（1秒間）を行います。

2．リセット解除後は，下記のようにリセットからの復帰を示す表示と時計の表示を一定周期で切り替える動作となります。いずれかのキーを入力すると通常の時計表示となります。



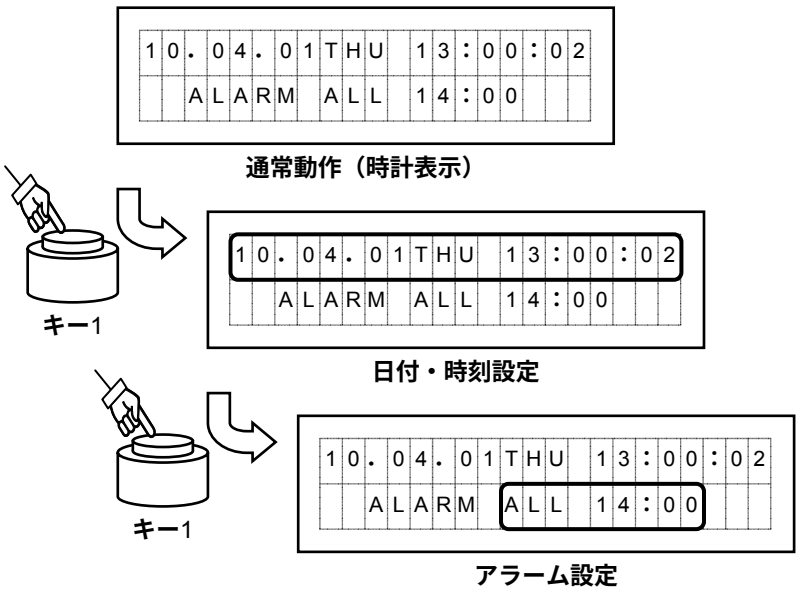
(2) キー操作について

日付，時刻，およびアラーム設定はキー操作（キー1～4）により変更が可能です。

①キー1による操作

キー1は設定キーです。時計表示中の動作を下記の順で切り替えます。

通常動作（時計表示）→日付・時刻設定→アラーム設定→通常動作（時計表示）→…



備考． 図中の太線部が設定中の箇所です。

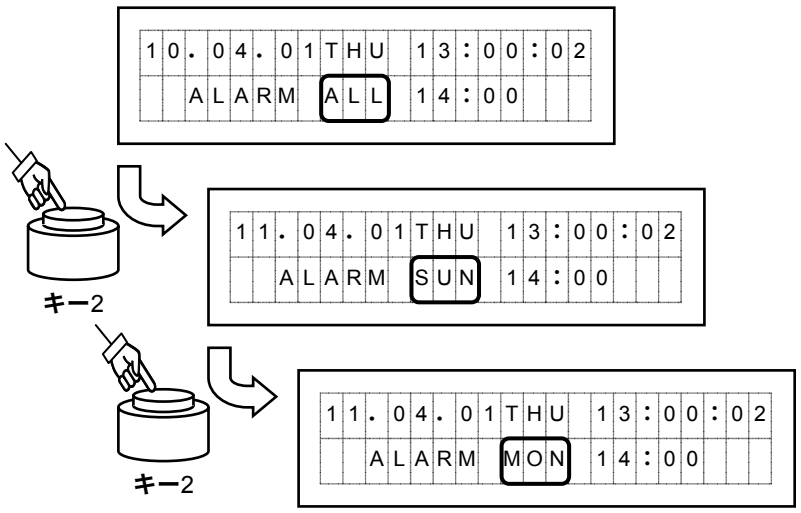
②キー2による操作

キー2はアップ・キーです。

＜アラーム曜日の場合＞

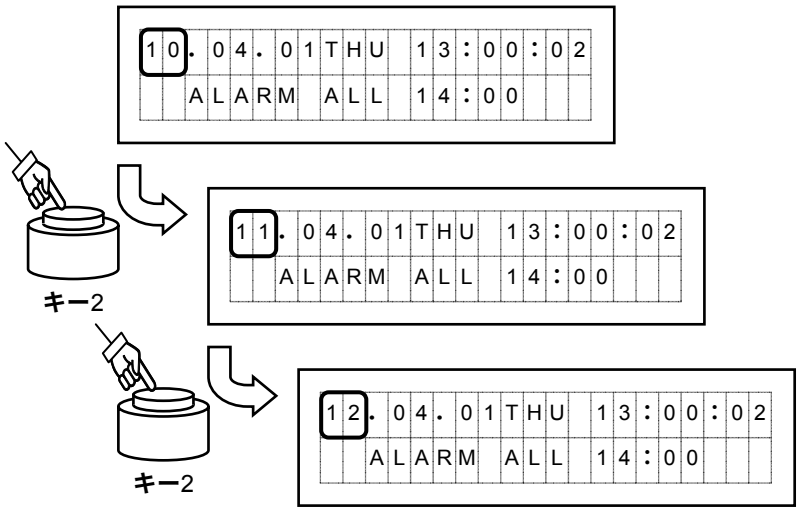
設定中の値を下記のように切り替えます。

ALL[注]→SUN→MON→TUE→WED→THU→FRI→SAT→ALL[注]→…



＜その他の場合＞

設定中の値をインクリメントします。



注. “ALL”は「毎日」を意味しています。

注意. 曜日はユーザ操作により年月日と合わせてください。

備考. 図中の太線部が設定中の箇所です。

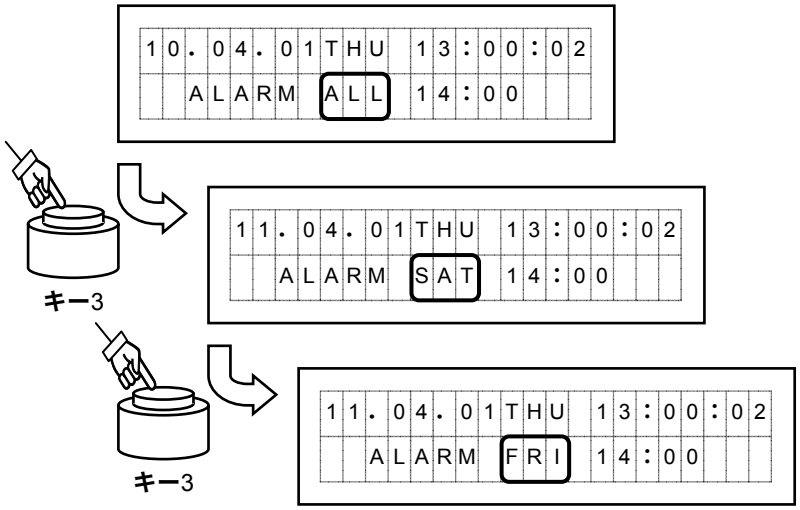
③キー3による操作

キー3はダウン・キーです。

＜アラーム曜日の場合＞

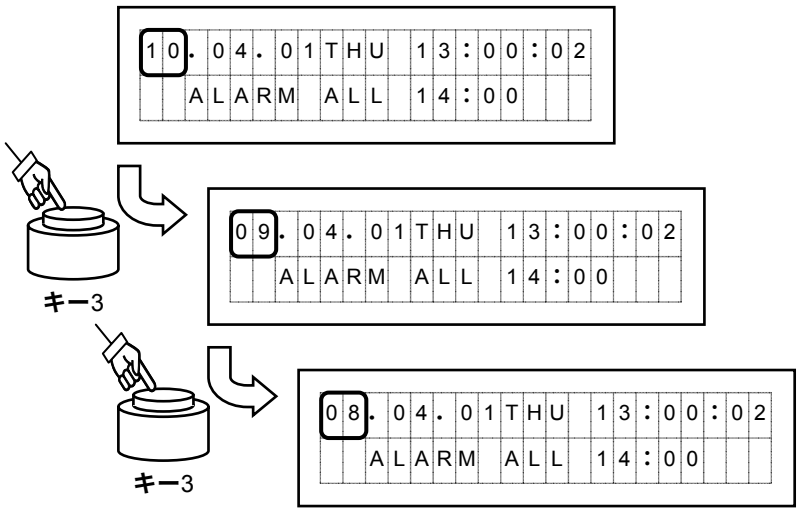
設定中の値を下記のように切り替えます。

ALL[注]→SAT→FRI→THU→WED→TUE→MON→SUN→ALL[注]→…



＜その他の場合＞

設定中の値をデクリメントします。



注. “ALL”は「毎日」を意味しています。

注意. 曜日はユーザ操作により年月日と合わせてください。

備考. 図中の太線部が設定中の箇所です。

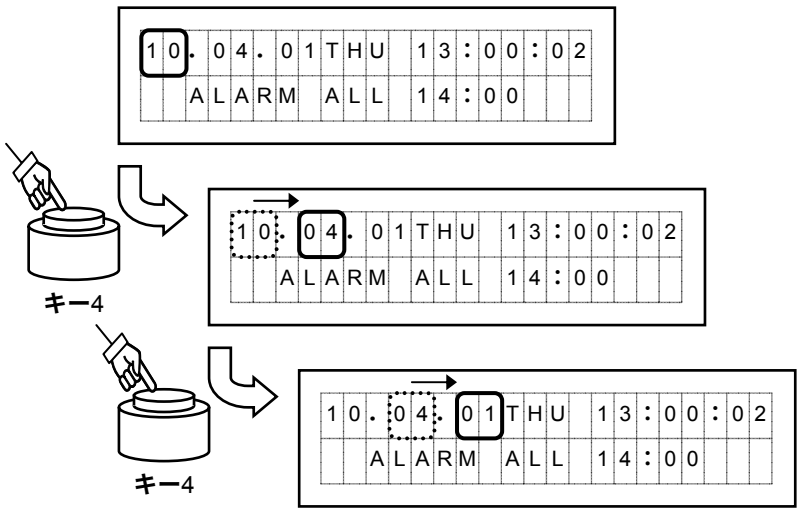
④キー4による操作

キー4は項目キーです。

＜時刻・日付設定の場合＞

設定項目を下記の順で切り替えます。

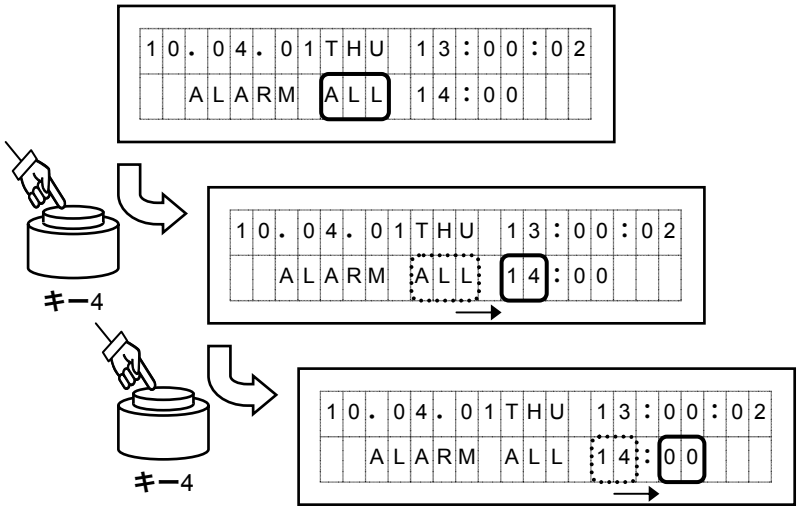
年→月→日→曜日→時→分→秒→年→…



＜アラーム設定の場合＞

設定項目を下記の順で切り替えます。

曜日→時→分→曜日→…



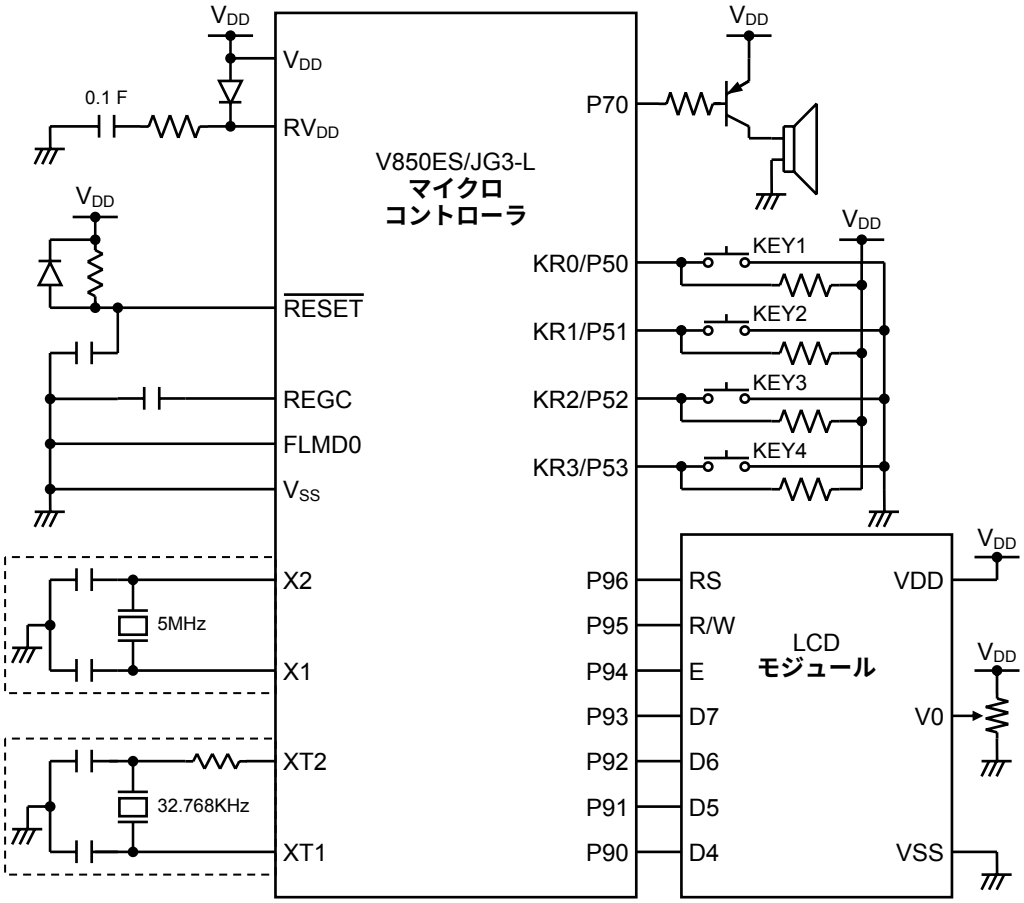
備考． 図中の太線部が設定中の箇所です。

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マイコン以外の使用デバイスを説明します。

2. 1　回路図

回路図を次に示します。



注意1. **通常動作モードのとき，V_{DD}は2.8 V＜V_{DD}≦3.6 Vの電圧範囲で使用してください。**

2. EVDD端子，AVREF0端子，AVREF1端子はV_{DD}に直接接続してください。
3. EVSS端子，AVSS端子はGNDに直接接続してください。
4. REGC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. FLMD0端子は，通常動作モード時はGNDに接続してください。
6. 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。
7. メイン・クロック発振回路およびサブクロック発振回路は，配線容量などの影響を避けるために，図中の破線の部分を次のように配線してください。
   - 配線は極力短くする。
   - 他の信号線と交差させない。
   - 変化する大電流が流れる線に接近させない。
   - 発振回路のコンデンサの接地点は，常にV_{SS}と同電位になるようにする。
   - 大電流が流れるグランド・パターンに接地しない。
   - 発振回路から信号を取り出さない。
8. 電源電圧（V_{DD}）の供給が停止した場合にもRTCバックアップ電源（RV_{DD}）への電源供給が一定期間は継続できるよう，RV_{DD}端子に電気二重層キャパシタ（0.1F）を接続します。

備考. **発振子の選択および発振回路定数についてはお客様において発振評価していただくか,発振子メーカに評価を依頼してください。**

2. 2 マイコン以外の使用デバイス

マイコン以外の使用デバイスを次に示します。

（1）LCDモジュール

表示用デバイスとしてLCDモジュール（キャラクタ表示，20文字×2行）を使用します。

（2）プッシュ・キー（KEY1，KEY2，KEY3，KEY4）

キー操作用にプッシュ・キーを使用します。

（3）電子ブザー

アラームに対応した出力として，電子ブザーを使用します。

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。

ファイル名（ツリー構造）	説 明	同封圧縮（*.zip）ファイル	
			
conf ─ crtE.s	スタート・アップ・ルーチン・ファイル 注1	–	●
├ AppNote_RTC_BUM.dir	リンク・ディレクティブ・ファイル 注2	●	●
├ AppNote_RTC_BUM.prj	統合開発環境 PM+用プロジェクト・ファイル	–	●
└ AppNote_RTC_BUM.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	●	●
├ lcd.c	LCDモジュール（NHD-0220FZ-FSW-GBW-P-3V3）の制御用処理を記述したC言語ソース・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機能を使用します。

＜周辺I/O機能＞

・リアル・タイム・カウンタ：定周期割り込み機能，アラーム割り込み機能，およびインターバル割り込み機能を使用します。また，RTCバックアップ・モードでの動作も行います。
・低電圧検出回路：低電圧検出用，および2.7V≦V_{DD}の確認用に使用します。

＜端子機能＞

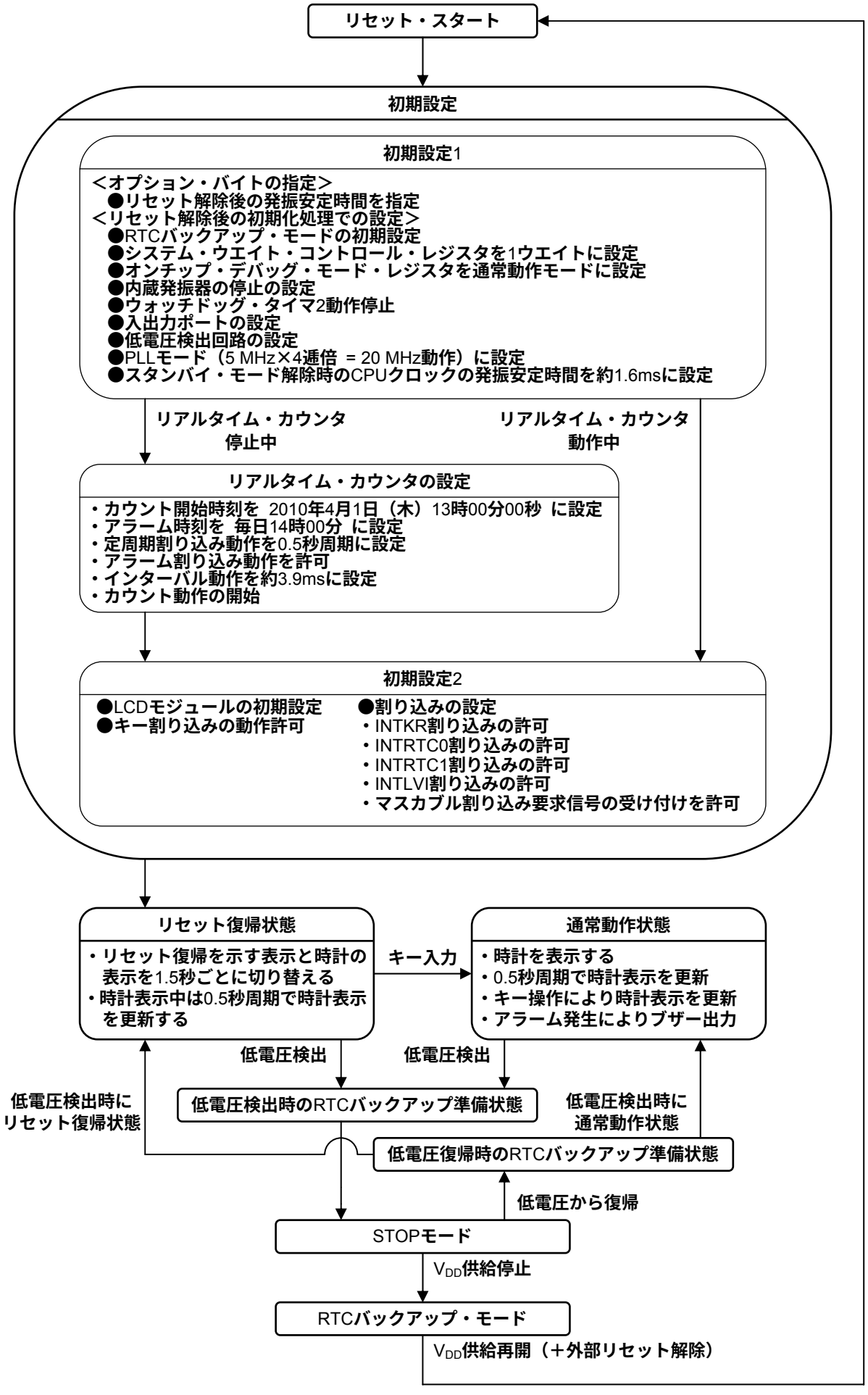
・KR0-KR3：キー入力用に使用します。
・P70：ブザーの制御用に使用します。
・P90-P96：LCDモジュールの制御用に使用します。

3. 3 初期設定と動作概要

このサンプル・プログラムでは，初期設定で，クロック周波数の選択や，ウォッチドッグ・タイマ2の停止設定，入出力ポートや外部割り込みの端子設定，低電圧検出回路の設定，リアルタイム・カウンタの設定，RTCバックアップ・モードの設定，LCDモジュールの設定などを行います。なお，初期設定開始時にすでにリアルタイム・カウンタが動作を開始している場合，リアルタイム・カウンタの設定は省略します。

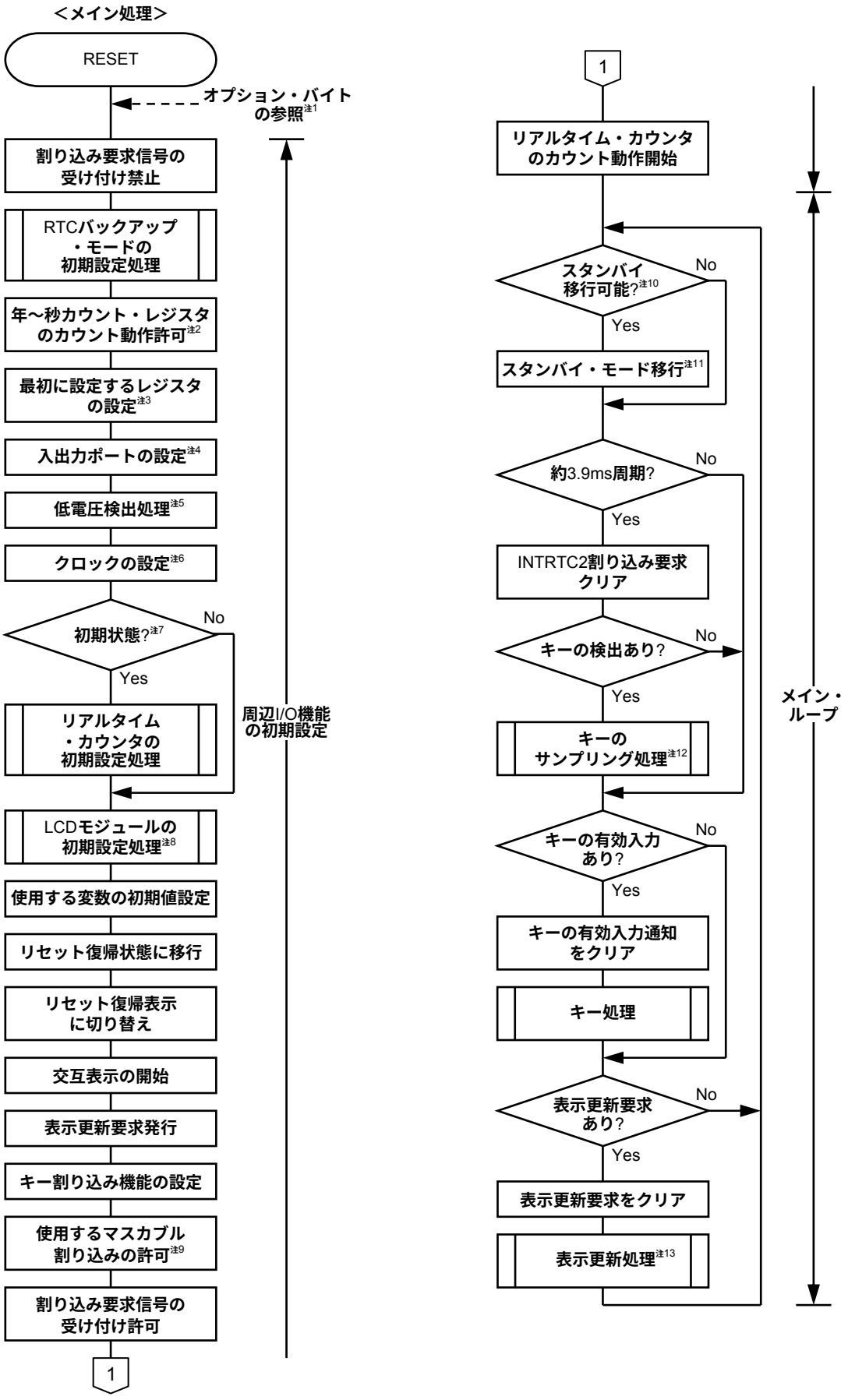
初期設定完了後は，0.5秒周期のINTRTC0割り込み，アラーム発生によるINTRTC1割り込み，約3.9msのインターバル割り込み，およびキー入力によるINTKR割り込みを使用し，キー操作，LCDモジュールの制御，ブザーの制御などを行います。また，低電圧検出または低電圧からの復帰によるINTLVI割り込みを使用し，RTCバックアップ準備状態の設定または解除を行います。

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。


リセット・スタート
初期設定
初期設定1
＜オプション・バイトの指定＞
●リセット解除後の発振安定時間を指定
＜リセット解除後の初期化処理での設定＞
●RTCバックアップ・モードの初期設定
●システム・ウエイト・コントロール・レジスタを1ウエイトに設定
●オンチップ・デバッグ・モード・レジスタ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
●入出力ポートの設定
●低電圧検出回路の設定
●PLLモード（5 MHz×4逓倍 = 20 MHz動作）に設定
●スタンバイ・モード解除時のCPUクロックの発振安定時間を約1.6msに設定
リアルタイム・カウンタ
停止中
リアルタイム・カウンタ
動作中
リアルタイム・カウンタの設定
・カウント開始時刻を 2010年4月1日（木）13時00分00秒 に設定
・アラーム時刻を 毎日14時00分 に設定
・定周期割り込み動作を0.5秒周期に設定
・アラーム割り込み動作を許可
・インターバル動作を約3.9msに設定
・カウント動作の開始
初期設定2
●LCDモジュールの初期設定
●キー割り込みの動作許可
●割り込みの設定
・INTKR割り込みの許可
・INTRTC0割り込みの許可
・INTRTC1割り込みの許可
・INTLVI割り込みの許可
・マスカブル割り込み要求信号の受け付けを許可
リセット復帰状態
・リセット復帰を示す表示と時計の表示を1.5秒ごとに切り替える
・時計表示中は0.5秒周期で時計表示を更新する
キー入力
通常動作状態
・時計を表示する
・0.5秒周期で時計表示を更新
・キー操作により時計表示を更新
・アラーム発生によりブザー出力
低電圧検出
低電圧検出
低電圧検出時に
リセット復帰状態
低電圧検出時のRTCバックアップ準備状態
低電圧検出時に
通常動作状態
低電圧復帰時のRTCバックアップ準備状態
低電圧から復帰
STOPモード
VDD供給停止
RTCバックアップ・モード
VDD供給再開（＋外部リセット解除）

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1. 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，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。

2. リセット解除後，すでにリアルタイム・カウンタが動作を開始している場合，リセット入力前の処理の影響により年～秒カウント・レジスタのカウント動作が停止に設定されていることがあるため，ここでカウント動作を許可しています。

3. VSWCレジスタ，OCDMレジスタ，RCMレジスタ，およびWDTM2レジスタを設定します。なお，設定内容は下記のとおりです。
   - システム・ウエイト・コントロール・レジスタを1ウエイトに設定
   - オンチップ・デバッグ・モード・レジスタを通常動作モードに設定
   - 内蔵発振器停止
   - ウォッチドッグ・タイマ2動作停止

4. P50-P53（KR0-KR3）をキー入力用，P70をブザー制御用，P90-P96をLCDモジュール制御用に設定します。また，未使用のポートはすべてロウ・レベル出力に設定します。

5. 低電圧検出回路の動作を許可し，電源電圧が2.8 V以上になるまで待ちます。

6. PLLモード（5MHz × 4逓倍 = 20MHz）に設定します。また，スタンバイ・モード解除時のCPUクロックの発振安定時間を約1.6 msに設定します。

7. リアルタイム・カウンタが初期状態かどうかを判定します。すでに動作を開始している場合，初期設定を省略します。

8. 使用するLCDモジュールの初期設定を行う処理です。

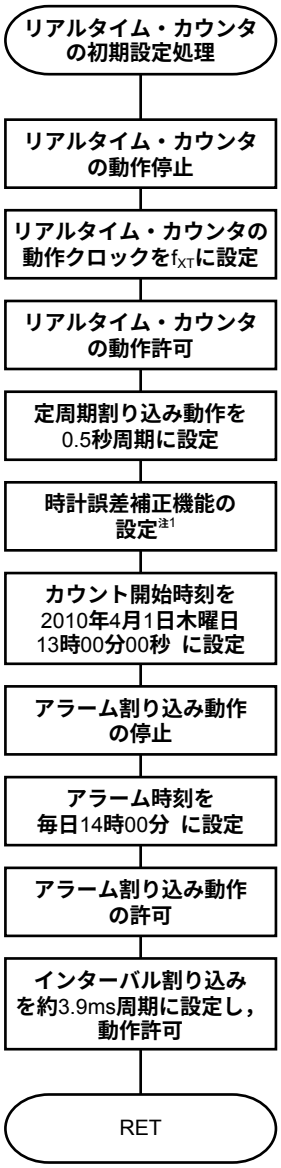
9. INTRTC0割り込み，INTRTC1割り込み，INTKR割り込み，およびINTLVI割り込みを許可します。

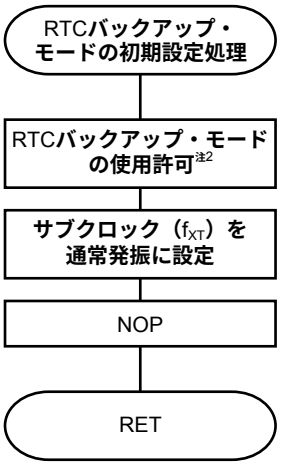
10. キーの検出および表示更新がない場合，スタンバイ・モード移行可能と判定します。

11. INTRTC0割り込み，INTRTC1割り込み，INTKR割り込み，およびINTLVI割り込みの発生によりスタンバイ・モードを解除します。

12. キー入力用ポートのサンプリング（約3.9ms周期×3回）を行う処理です。キー・コード（キー1／キー2／キー3／キー4），およびキーの有効入力通知を発行します。また，キーがオフになった場合，キーの検出通知をクリアします。

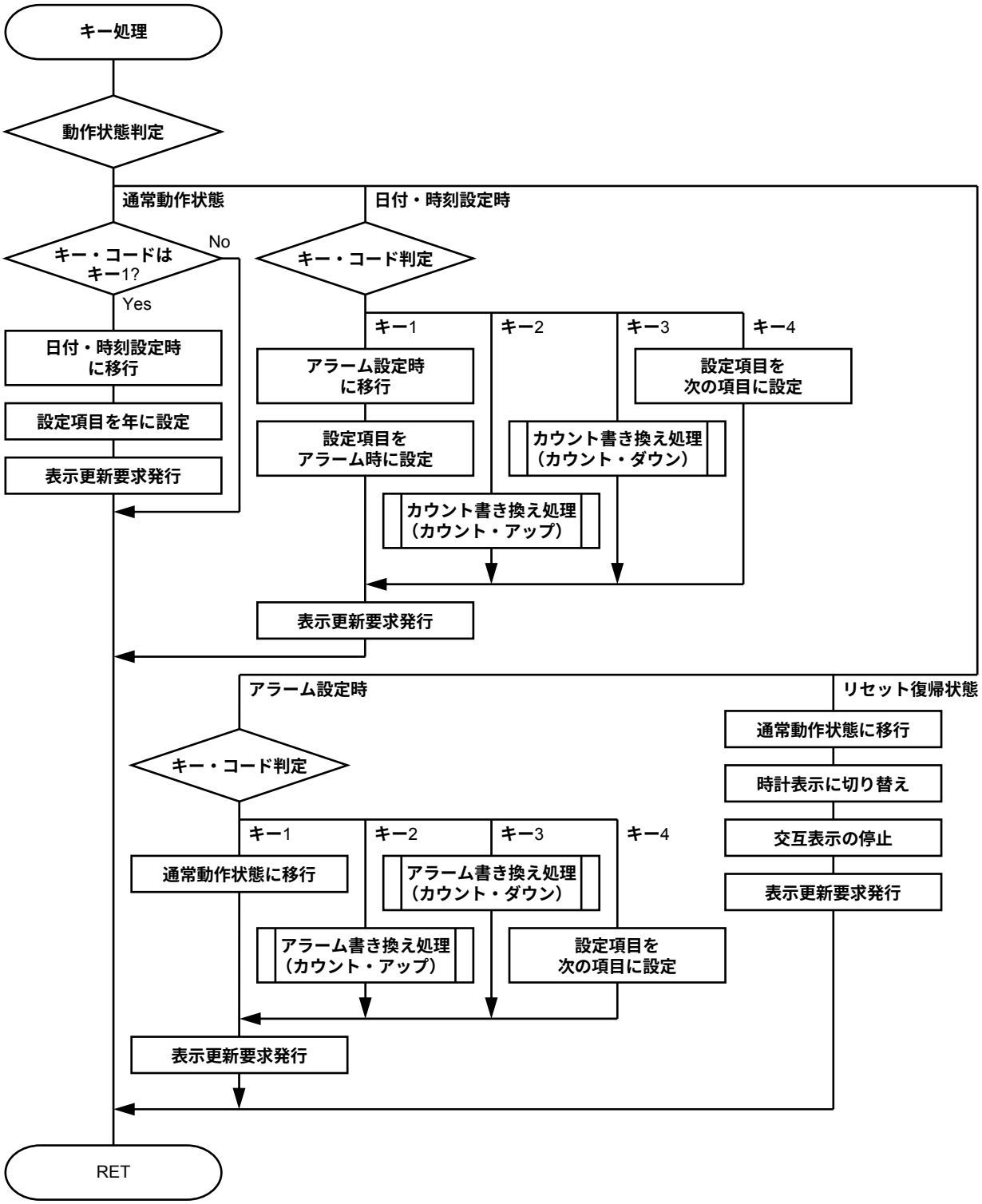
13. 表示用のデータに合わせてLCDモジュールを制御する処理です。

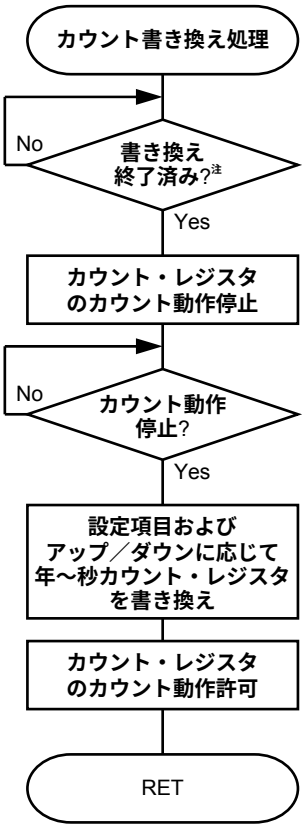




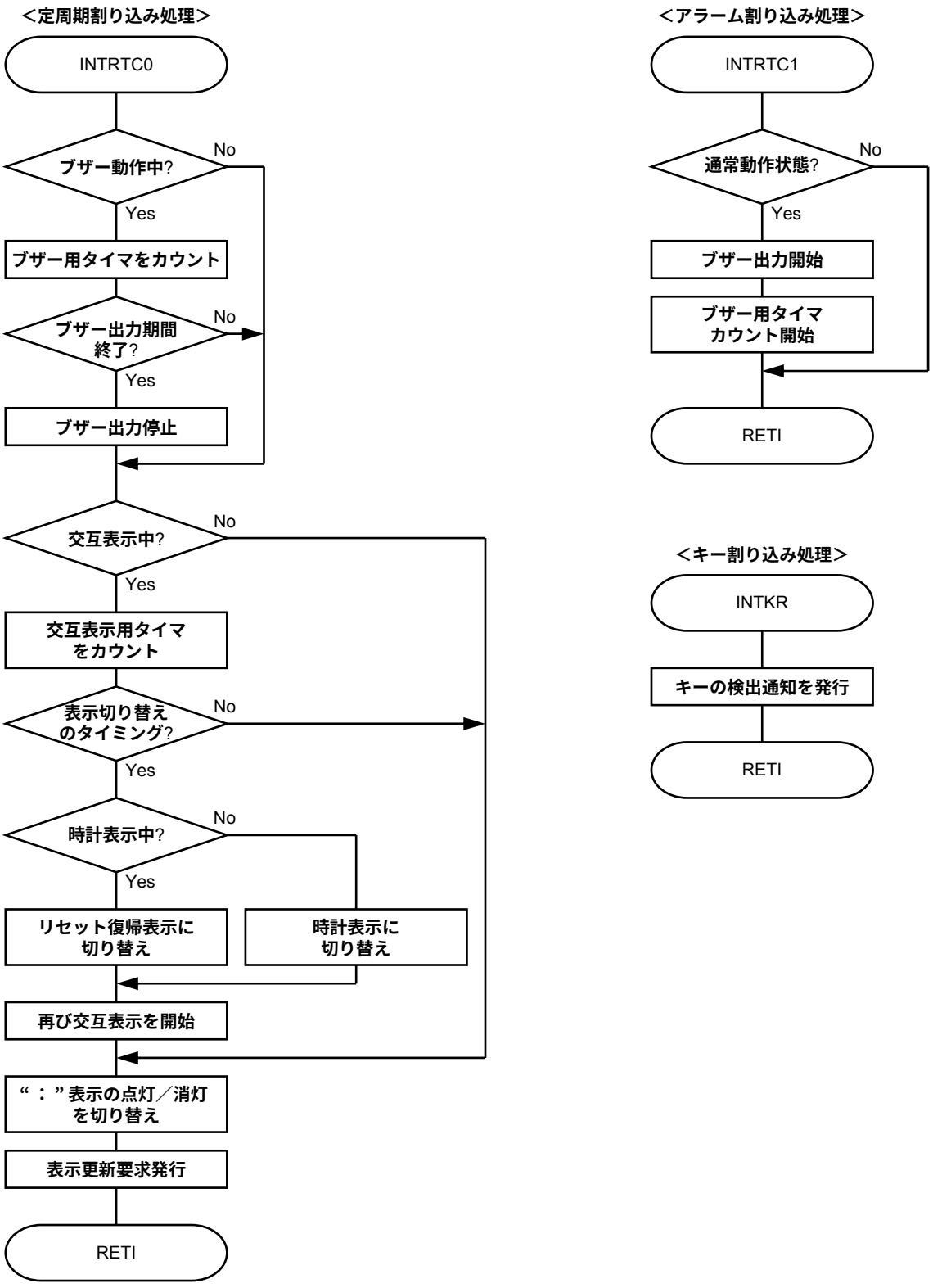
注1. **このサンプル・プログラムでは，時計誤差補正機能は停止としています。**

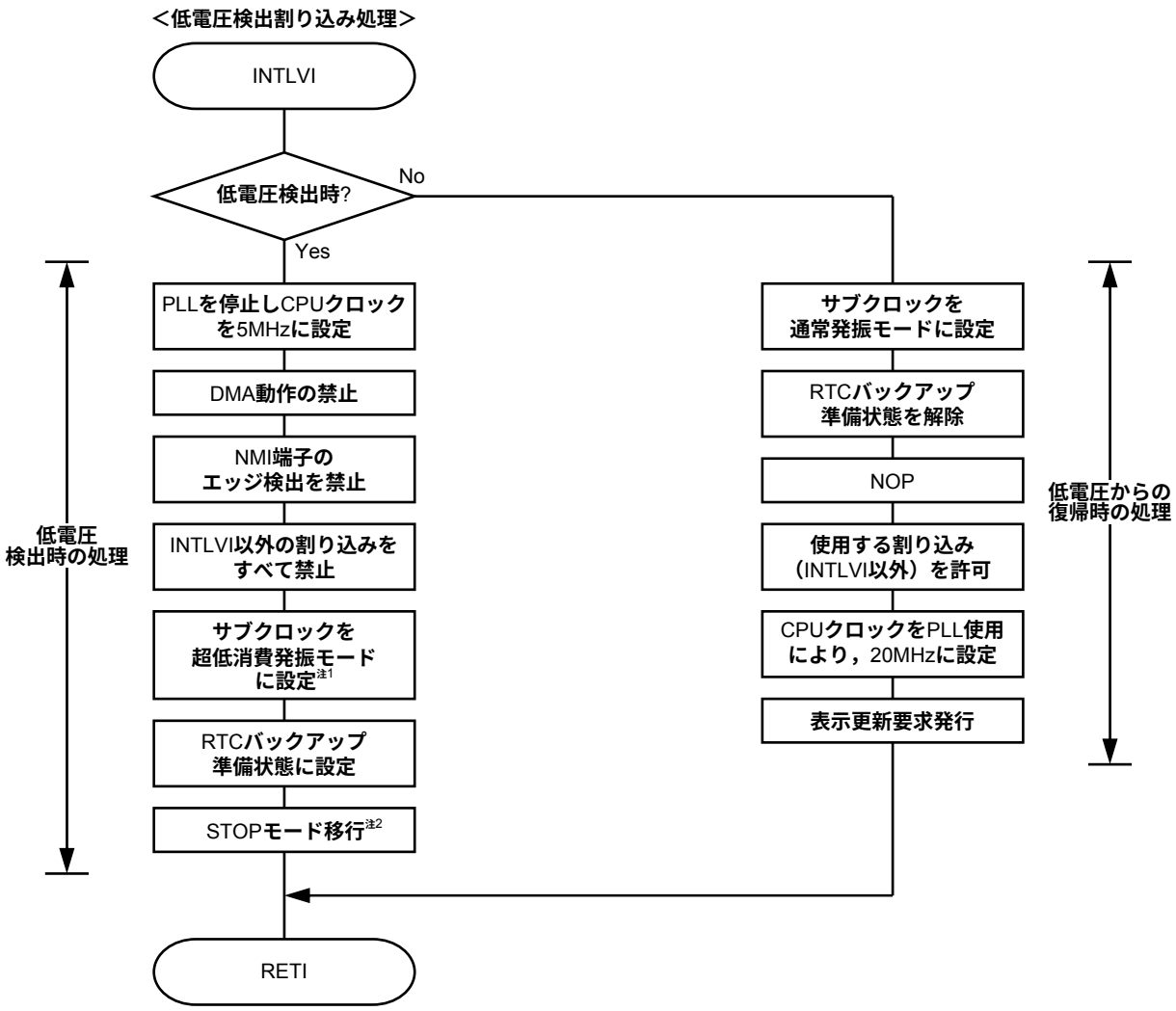
2. **リセット解除後に**RTC**バックアップ準備状態となっていた場合，ここで解除します。**


キー処理
動作状態判定
通常動作状態
日付・時刻設定時
キー・コードは
キー1?
No
Yes
キー・コード判定
キー1
キー2
キー3
キー4
日付・時刻設定時
に移行
設定項目を年に設定
表示更新要求発行
アラーム設定時
に移行
設定項目を
アラーム時に設定
カウント書き換え処理
（カウント・ダウン）
カウント書き換え処理
（カウント・アップ）
設定項目を
次の項目に設定
表示更新要求発行
アラーム設定時
リセット復帰状態
キー・コード判定
通常動作状態に移行
時計表示に切り替え
交互表示の停止
表示更新要求発行
キー1
キー2
キー3
キー4
通常動作状態に移行
アラーム書き換え処理
（カウント・ダウン）
アラーム書き換え処理
（カウント・アップ）
設定項目を
次の項目に設定
表示更新要求発行
RET



注.　前回のカウント・レジスタへの書き込みが終了していることを確認します。


<定周期割り込み処理>
INTRTC0
ブザー動作中?
No
Yes
ブザー用タイマをカウント
ブザー出力期間終了?
No
Yes
ブザー出力停止
交互表示中?
No
Yes
交互表示用タイマをカウント
表示切り替えのタイミング?
No
Yes
時計表示中?
No
Yes
リセット復帰表示に切り替え
時計表示に切り替え
再び交互表示を開始
“：”表示の点灯／消灯を切り替え
表示更新要求発行
RETI
<アラーム割り込み処理>
INTRTC1
通常動作状態?
No
Yes
ブザー出力開始
ブザー用タイマカウント開始
RETI
<キー割り込み処理>
INTKR
キーの検出通知を発行
RETI



注1. サブクロックの超低消費発振のためには必要ですが,RTCバックアップ・モードの必須条件ではありません。

2. STOPモード中，電源電圧（V_{DD}）が供給停止となった場合，RTCバックアップ・モードに移行します。一方，電源電圧（V_{DD}）が低電圧から通常動作電圧に復帰した場合，下記の①～④の順で処理を実行します。

① INTLVI割り込み信号が発生し，STOPモードが解除される。ただし，INTLVI割り込みの受け付けは保留される。

② RETIにより低電圧検出割り込み処理から復帰する。

③ 復帰後，保留されていたINTLVI割り込みが受け付けられ，再び低電圧検出割り込み処理に入る。

④ 上図の「低電圧からの復帰時の処理」に分岐し，RTCバックアップ準備状態を解除する。

第4章　設定方法について

この章では，リアルタイム・カウンタの設定，RTCバックアップ・モードの設定，および時計誤差補正について説明します。

低電圧検出回路の初期設定については，V850ES/Jx3-L　サンプル・プログラム（低電圧検出回路（LVI））　低電圧検出時リセット発生編　アプリケーション・ノートを参照してください。

その他の初期設定については，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　アプリケーション・ノートを参照してください。

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

C言語の拡張記述については，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

4. 1 リアルタイム・カウンタの設定

このサンプル・プログラムでは，次のレジスタでリアルタイム・カウンタを制御します。

- リアルタイム・カウンタ・コントロール・レジスタ0（RC1CC0）
- リアルタイム・カウンタ・コントロール・レジスタ1（RC1CC1）
- リアルタイム・カウンタ・コントロール・レジスタ2（RC1CC2）
- リアルタイム・カウンタ・コントロール・レジスタ3（RC1CC3）
- 秒カウント・レジスタ（RC1SEC）
- 分カウント・レジスタ（RC1MIN）
- 時カウント・レジスタ（RC1HOUR）
- 日カウント・レジスタ（RC1DAY）
- 曜日カウント・レジスタ（RC1WEEK）
- 月カウント・レジスタ（RC1MONTH）
- 年カウント・レジスタ（RC1YEAR）
- 時計誤差補正レジスタ（RC1SUBU）
- アラーム分レジスタ（RC1ALM）
- アラーム時レジスタ（RC1ALH）
- アラーム曜日レジスタ（RC1ALW）

(1) リアルタイム・カウンタ・コントロール・レジスタ0（RC1CC0）

リアルタイム・カウンタ動作の制御，動作クロックの選択を行う8ビットのレジスタです。8/1ビット単位でリード／ライト可能です。

図4－1－1　リアルタイム・カウンタ・コントロール・レジスタ0（RC1CC0）のフォーマット

リアルタイム・カウンタ・コントロール・レジスタ0（RC1CC0）

アドレス：　FFFFFADDH

7	6	5	4	3	2	1	0
RC1PWR	RC1CKS注	0	0	0	0	0	0

RC1PWR	リアルタイム・カウンタの動作の制御
0	リアルタイム・カウンタ動作停止
1	**リアルタイム・カウンタ動作許可**

RC1CKS注	動作クロックの選択
0	**f_{XT}を動作クロックとして選択**
1	f_{BRG}を動作クロックとして選択

注．　f_{XT}を動作クロックとして選択した場合，RTCバックアップ準備状態（RTCBUMCTL0.RBMSET = 1）では，必ずRC1CKSビットを0に設定してください。

注意1．動作中のリアルタイム・カウンタを停止（RC1PWR = 1→0）させる場合は，ユーザーズ・マニュアル 11. 4. 8 リアルタイム・カウンタの初期化　にそった設定を行ってください。

2．RC1CKSビットの書き換えは，リアルタイム・カウンタ動作停止時（RC1PWRビット = 0）のみ可能です。また，RC1PWRビットを“0”から “1”にするのと同時にRC1CKSビットを書き換えることは禁止です。

3．ビット0-5には必ず0を設定してください。

備考1．RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

2．図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。

(2) リアルタイム・カウンタ・コントロール・レジスタ1（RC1CC1）

リアルタイム・カウンタ動作の開始／停止，RTCCL端子／RTC1HZ端子の制御，12/24時間制，定周期割り込み機能を設定する8ビットのレジスタです。8/1ビット単位でリード／ライト可能です。

図4－1－2　リアルタイム・カウンタ・コントロール・レジスタ1（RC1CC1）のフォーマット

リアルタイム・カウンタ・コントロール・レジスタ1（RC1CC1）

アドレス：　FFFFFADEH

7	6	5	4	3	2	1	0
RTCE	0	CLOE1	CLOE0	AMPM	CT2	CT1	CT0

RTCE	各カウンタの動作の制御
0	カウンタ動作停止
1	**カウンタ動作開始**

CLOE1	RTC1HZ端子の出力制御
0	**RTC1HZ端子の出力（1 Hz）禁止**
1	RTC1HZ端子の出力（1 Hz）許可

CLOE0	RTCCL端子の出力制御
0	**RTCCL端子の出力（32.768 kHz）禁止**
1	RTCCL端子の出力（32.768 kHz）許可

AMPM	12時間制／24時間制の選択
0	12時間制（午前／午後を表示）
1	**24時間制**

CT2	CT1	CT0	定周期割り込み（INTRTC0）の選択
0	0	0	定周期割り込み機能を使用しない
0	**0**	**1**	**0.5秒に1度（秒カウントアップに同期）**
0	1	0	1秒に1度（秒カウントアップと同時）
0	1	1	1分に1度（毎分00秒）
1	0	0	1時間に1度（毎時00分00秒）
1	0	0	1日に1度（毎日00時00分00秒）
1	1	x	1月に1度（毎月1日午前00時00分00秒）

注意1．RTCEビット = 1の状態でRTCEビットに“0”を書き込むことは禁止です。RC1PWRビットをクリアすることでRTCEビットをクリアしてください。

2．CLOE1ビットの設定変更時，RTC1HZ出力は次のように動作します。

・0→1に変更した場合：最大2クロック後（2 x 32.768 kHz）に，RTC1HZ出力は1 Hzのパルスを出力。

・1→0に変更した場合：最大2クロック後（2 x 32.768 kHz）に，RTC1HZ出力は出力停止（ロウ・レベル固定）。

3．AMPMビットを書き換えた場合は，RC1HOURレジスタを再設定してください。

4．ビット6には必ず0を設定してください。

備考1．RTCバックアップモード時は，定周期割り込みをマスク（RTC0MKビット = 1）してください。RTCCL端子出力とRTC1HZ端子出力は停止します。

2．RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

3．図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。

(3) リアルタイム・カウンタ・コントロール・レジスタ2（RC1CC2）

アラーム割り込み機能，カウンタのウエイトを制御する8ビットのレジスタです。8/1ビット単位でリード／ライト可能です。

図4－1－3　リアルタイム・カウンタ・コントロール・レジスタ2（RC1CC2）のフォーマット

リアルタイム・カウンタ・コントロール・レジスタ2（RC1CC2）

アドレス： FFFFFADFH

7	6	5	4	3	2	1	0
WALE	0	0	0	0	0	RWST	RWAIT

WALE	アラーム割り込み（INTRTC1）機能の動作制御
0	アラームの一致による割り込みを発生しない
1	アラームの一致による割り込みを発生する

RWST	リアルタイム・カウンタのウエイト状態
0	カウンタ動作中
1	秒～年カウンタのカウント・アップ停止状態 （カウンタ値の読み出し，書き込み許可状態）

RWAIT	リアルタイム・カウンタのウエイト制御
0	カウンタ動作設定
1	秒～年カウンタのカウント動作停止 （カウンタ値の読み出し，書き込みモード。）

注意1．各カウンタ値の読み出し／書き込みを行う場合は，RWSTビットが1になっていることを確認してください。

2．RWAITビットを“0”に設定しても，各カウンタ書き込み中は，RWSTビットは“0”になりません。各カウンタ書き込み完了後に“0”になります。

3．ビット2-6には必ず0を設定してください。

備考1．RTCバックアップモード時は，アラーム割り込みをマスク（RTC1MKビット = 1）してください。

2．RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

3．図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。

(4) リアルタイム・カウンタ・コントロール・レジスタ3（RC1CC3）

インターバル割り込み機能，RTCDIV端子を制御する8ビットのレジスタです。8/1ビット単位でリード／ライト可能です。

図4－1－4　リアルタイム・カウンタ・コントロール・レジスタ3（RC1CC3）のフォーマット

リアルタイム・カウンタ・コントロール・レジスタ3（RC1CC3）

アドレス：　FFFFFAE0H

7	6	5	4	3	2	1	0
RINTE	CLOE2	CKDIV	0	0	ICT2	ICT1	ICT0

RINTE	インターバル割り込み（INTRTC2）の制御
0	インターバル割り込みを発生しない
1	**インターバル割り込みを発生する**

CLOE2	RTCDIV端子の出力制御
0	**RTCDIV端子の出力禁止**
1	RTCDIV端子の出力許可

CKDIV	RTCDIV端子の出力周波数の選択
0	RTCDIV端子から512 Hz（1.95 ms）を出力
1	RTCDIV端子から16.384 kHz（0.061 ms）を出力

ICT2	ICT1	ICT0	インターバル割り込み（INTRTC2）の選択
0	0	0	$2^6/f_{XT}$（1.953125 ms）
0	**0**	**1**	$2^7/f_{XT}$（3.90625 ms）
0	1	0	$2^8/f_{XT}$（7.8125 ms）
0	1	1	$2^9/f_{XT}$（15.625 ms）
1	0	0	$2^{10}/f_{XT}$（31.25 ms）
1	0	0	$2^{11}/f_{XT}$（62.5 ms）
1	1	x	$2^{12}/f_{XT}$（125 ms）

注意1．CLOE2ビットの設定変更時，RTCDIV出力は次のように動作します。

- 0→1に変更した場合：最大2クロック後（2 x 32.768 kHz）に，CKDIVビットで設定したパルスを出力。
- 1→0に変更した場合：最大2クロック後（2 x 32.768 kHz）に，RTCDIV出力は出力停止（ロウ・レベル固定）。

2．ビット3-4には必ず0を設定してください。

備考1．RTCバックアップモード時は，インターバル割り込み，RTCDIV端子出力は停止します。

2．RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

3．図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。

(5) 秒カウント・レジスタ（RC1SEC）

0-59（10進）までの値を取り，秒のカウント値を示す8ビットのレジスタです。8ビット単位でリード／ライト可能です。サブカウント・レジスタ（RC1SUBC）[注]からのオーバフローによりカウント・アップします。

書き込みを行った場合は，バッファに書き込まれ，最大2クロック（2 x 32.768 kHz）後にカウンタへ書き込まれます。また設定する値は10進の00-59をBCDコードで設定してください。

注． サブカウント・レジスタ（RC1SUBC）は0000H-7FFFHまでの値をとり，32.768kHzのクロックで1秒をカウントする16ビットのレジスタです。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。

注意． RC1SECレジスタに00-59以外の値を設定することは禁止です。

備考1． RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

2． このサンプル・プログラムでは，初期値として00Hを設定します。

図4－1－5 秒カウント・レジスタ（RC1SEC）のフォーマット

秒カウント・レジスタ（RC1SEC）

アドレス： FFFFFAD2H

7	6	5	4	3	2	1	0
0							

(6) 分カウント・レジスタ（RC1MIN）

0-59（10進）までの値を取り，分のカウント値を示す8ビットのレジスタです。8ビット単位でリード／ライト可能です。

秒カウンタからのオーバフローによりカウント・アップします。

書き込みを行った場合は，バッファに書き込まれ最大2クロック（2 x 32.768 kHz）後に，カウンタへ書き込まれます。また設定する値は，10進の00-59をBCDコードで設定してください。

注意． RC1MINレジスタに00-59以外の値を設定することは禁止です。

備考1． RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

2． このサンプル・プログラムでは，初期値として00Hを設定します。

図4－1－6 分カウント・レジスタ（RC1MIN）のフォーマット

分カウント・レジスタ（RC1MIN）

アドレス： FFFFFAD3H

7	6	5	4	3	2	1	0
0							

（7）時カウント・レジスタ（RC1HOUR**）**

0-23**または**1-12（10**進）までの値を取り，時のカウント値を示す**8**ビットのレジスタです。8**ビット単位でリード／ライト可能です。

分カウンタからのオーバフローによりカウント・アップします。

書き込みを行った場合は，バッファに書き込まれ最大2**クロック（**2 x 32.768 kHz**）後にカウンタへ書き込みされます。また設定する値は，**10**進の**00-23**または**01-12，21-32**をBCD**コードで設定してください。**

注意1．RC1HOUR**レジスタのビット**5**は，**AMPM = 0（12**時間制）を選択した場合，**AM（0）／PM（1）を示します。

2．RC1HOUR**レジスタに**01-12, 21-32（AMPM**ビット** = 0），**または**00-23（AMPM**ビット** = 1）**以外の値を設定することは禁止です。**

備考1．RV_{DD}**の電源投入により，**12H**になります。また，**RV_{DD}**の電源投入以外の要因でリセットが発生した場合は値を保持します。**

2．**このサンプル・プログラムでは，初期値として**13H（AMPM**ビット** = 1）**を設定します。**

図4－1－7　**時カウント・レジスタ（**RC1HOUR**）のフォーマット**

時カウント・レジスタ（RC1HOUR**）**

アドレス： FFFFFAD4H

7	6	5	4	3	2	1	0
0	0						

AMPMビットの設定値とRC1HOURレジスタの値と時間の関係を表4－1に示します。

表4－1　時間桁表示表

12時間表示（AMPMビット = 0）		24時間表示（AMPMビット = 1）	
時間	RC1HOURレジスタ	時間	RC1HOURレジスタ
AM0時	12H	0時	00H
AM1時	01H	1時	01H
AM2時	02H	2時	02H
AM3時	03H	3時	03H
AM4時	04H	4時	04H
AM5時	05H	5時	05H
AM6時	06H	6時	06H
AM7時	07H	7時	07H
AM8時	08H	8時	08H
AM9時	09H	9時	09H
AM10時	10H	10時	10H
AM11時	11H	11時	11H
PM0時	32H	12時	12H
PM1時	21H	13時	13H
PM2時	22H	14時	14H
PM3時	23H	15時	15H
PM4時	24H	16時	16H
PM5時	25H	17時	17H
PM6時	26H	18時	18H
PM7時	27H	19時	19H
PM8時	28H	20時	20H
PM9時	29H	21時	21H
PM10時	30H	22時	22H
PM11時	31H	23時	23H

RC1HOURレジスタの値は，AMPMビットが“0”のとき12時間表示で，“1”のとき24時間表示となります。

12時間表示の場合は，RC1HOURの5ビットが午前／午後を表示し，午前（AM）のときに0に，午後（PM）のときに1となります。

（8）日カウント・レジスタ（RC1DAY）

1-31（10進）までの値を取り，日のカウント値を示す8ビットのレジスタです。8ビット単位でリード／ライト可能です。

時カウンタからのオーバフローによりカウント・アップします。

カウンタは，次に示すようにカウントします。

- 01-31（1，3，5，7，8，10，12月）
- 01-30（4，6，9，11月）
- 01-29（2月 うるう年）
- 01-28（2月 通常年）

書き込みを行った場合は，バッファに書き込まれ最大2クロック（2 x 32.768 kHz）後にカウンタへ書き込まれます。また設定する値は，10進の01-31をBCDコードで設定してください。

注意． RC1DAYレジスタに01-31以外の値を設定することは禁止です。また上記カウント範囲外（2月30日など）を設定することも禁止です。

備考1． RV_{DD}の電源投入により，01Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

2． このサンプル・プログラムでは，初期値として01Hを設定します。

図4－1－8　日カウント・レジスタ（RC1DAY）のフォーマット

日カウント・レジスタ（RC1DAY）

アドレス： FFFFFAD6H

7	6	5	4	3	2	1	0
0	0						

(9) 曜日カウント・レジスタ（RC1WEEK）

0-6（10進）までの値を取り，曜日のカウント値を示す8ビットのレジスタです。8ビット単位でリード／ライト可能です。

日カウンタと同期してカウント・アップします。

書き込みを行った場合は，バッファに書き込まれ最大2クロック（2 x 32.768 kHz）後にカウンタへ書き込まれます。また設定する値は，10進の00-06をBCDコードで設定してください。

図4－1－9　曜日カウント・レジスタ（WEEK）のフォーマット

日カウント・レジスタ（RC1DAY）

アドレス： FFFFFAD5H

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

注意1．RC1WEEKレジスタに00-06以外の値を設定することは禁止です。

2．曜日カウント・レジスタには，月カウント・レジスタおよび日カウント・レジスタに対応した値が自動的に格納されるわけではありません。RV_{DD}の電源投入後，必ず次のように設定してください。

曜日	RC1WEEK
日	00H
月	01H
火	02H
水	03H
木	04H
金	05H
土	06H

備考1．RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

2．このサンプル・プログラムでは，初期値として04Hを設定します。

（10）月カウント・レジスタ（RC1MONTH）

RC1MONTHレジスタは1-12（10進）までの値を取り，月のカウント値を示す8ビットのレジスタです。8ビット単位でリード／ライト可能です。

日カウンタからのオーバフローによりカウント・アップします。

書き込みを行った場合は，バッファに書き込まれ最大2クロック（2 x 32.768 kHz）後にカウンタへ書き込まれます。また設定する値は，10進の01-12をBCDコードで設定してください。

注意． RC1MONTHレジスタに01-12以外の値を設定することは禁止です。

備考1． RV_{DD}の電源投入により，01Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

2． このサンプル・プログラムでは，初期値として04Hを設定します。

図4－1－10　月カウント・レジスタ（RC1MONTH）のフォーマット

月カウント・レジスタ（RC1MONTH）

アドレス： FFFFFAD7H

7	6	5	4	3	2	1	0
0	0	0					

（11）年カウント・レジスタ（RC1YEAR）

0-99（10進）までの値を取り，年のカウント値を示す8ビットのレジスタです。8ビット単位でリード／ライト可能です。

月カウンタからのオーバフローによりカウント・アップします。

00，04，08，・・・，92，96がうるう年となります。

書き込みを行った場合は，バッファに書き込まれ最大2クロック（2 x 32.768 kHz）後にカウンタへ書き込まれます。また設定する値は，10進の00-99をBCDコードで設定してください。

注意． RC1YEARレジスタに00-99以外の値を設定することは禁止です。

備考1． RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

2． このサンプル・プログラムでは，初期値として10Hを設定します。

図4－1－11　年カウント・レジスタ（RC1YEAR）のフォーマット

年カウント・レジスタ（RC1YEAR）

アドレス： FFFFFAD8H

7	6	5	4	3	2	1	0

（12）アラーム分設定レジスタ（RC1ALM）

アラームの分を設定する8ビットのレジスタです。8ビット単位でリード／ライト可能です。

注意1．設定する値は，10進の00-59をBCDコードで設定してください。範囲外の値を設定した場合，アラームは検出されません。

2．リアルタイム・カウンタ動作中にアラーム分設定レジスタ（RC1ALM）を変更する場合の設定手順については，ユーザーズ・マニュアルを参照してください。

備考1．RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

2．このサンプル・プログラムでは，初期値として00Hを設定します。

図4－1－12　アラーム分設定レジスタ（RC1ALM）のフォーマット

アラーム分設定レジスタ（RC1ALM）

アドレス： FFFFFADAH

7	6	5	4	3	2	1	0
0							

（13）アラーム時設定レジスタ（RC1ALH）

アラームの時を設定する8ビットのレジスタです。8ビット単位でリード／ライト可能です。

注意1．設定する値は，10進の00-23または，01-12, 21-32をBCDコードで設定してください。範囲外の値を設定した場合，アラームは検出されません。

2．RC1ALHレジスタのビット5は，AMPMビット = 0（12時間制）を選択した場合，AM（0）／PM（1）を示します。

3．リアルタイム・カウンタ動作中にアラーム時設定レジスタ（RC1ALH）を変更する場合の設定手順については，ユーザーズ・マニュアルを参照してください。

備考1．RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

2．このサンプル・プログラムでは，初期値として14Hを設定します。

図4－1－13　アラーム時設定レジスタ（RC1ALH）のフォーマット

アラーム分設定レジスタ（RC1ALH）

アドレス： FFFFFADBH

7	6	5	4	3	2	1	0
0	0						

（14）アラーム曜日設定レジスタ（RC1ALW）

アラームの曜日を設定する8ビットのレジスタです。8ビット単位でリード／ライト可能です。

図4－1－14　アラーム曜日設定レジスタ（RC1ALW）のフォーマット

アラーム曜日設定レジスタ（RC1ALW）

アドレス： FFFFFADCH

7	6	5	4	3	2	1	0
0	RC1ALW6	RC1ALW5	RC1ALW4	RC1ALW3	RC1ALW2	RC1ALW1	RC1ALW0
	土	金	木	水	火	月	日

RC1ALW6	アラーム割り込み曜日設定ビット6
0	RC1WEEK = 06H（土）のときに，アラーム割り込みを発生しない
1	RC1WEEK = 06H（土）のときに，RC1ALM，RC1ALHレジスタで設定した時間になるとアラーム割り込みを発生する

RC1ALW5	アラーム割り込み曜日設定ビット5
0	RC1WEEK = 05H（金）のときに，アラーム割り込みを発生しない
1	RC1WEEK = 05H（金）のときに，RC1ALM，RC1ALHレジスタで設定した時間になるとアラーム割り込みを発生する

RC1ALW4	アラーム割り込み曜日設定ビット4
0	RC1WEEK = 04H（木）のときに，アラーム割り込みを発生しない
1	RC1WEEK = 04H（木）のときに，RC1ALM，RC1ALHレジスタで設定した時間になるとアラーム割り込みを発生する

RC1ALW3	アラーム割り込み曜日設定ビット3
0	RC1WEEK = 03H（水）のときに，アラーム割り込みを発生しない
1	RC1WEEK = 03H（水）のときに，RC1ALM，RC1ALHレジスタで設定した時間になるとアラーム割り込みを発生する

RC1ALW2	アラーム割り込み曜日設定ビット2
0	RC1WEEK = 02H（火）のときに，アラーム割り込みを発生しない
1	RC1WEEK = 02H（火）のときに，RC1ALM，RC1ALHレジスタで設定した時間になるとアラーム割り込みを発生する

RC1ALW1	アラーム割り込み曜日設定ビット1
0	RC1WEEK = 01H（月）のときに，アラーム割り込みを発生しない
1	RC1WEEK = 01H（月）のときに，RC1ALM，RC1ALHレジスタで設定した時間になるとアラーム割り込みを発生する

RC1ALW0	アラーム割り込み曜日設定ビット0
0	RC1WEEK = 00H（日）のときに，アラーム割り込みを発生しない
1	RC1WEEK = 00H（日）のときに，RC1ALM，RC1ALHレジスタで設定した時間になるとアラーム割り込みを発生する

注意1．ビット7には必ず0を設定してください。

2．リアルタイム・カウンタ動作中にアラーム曜日設定レジスタ（RC1ALW）を変更する場合の設定手順については，ユーザーズ・マニュアルを参照してください。

備考1．RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

2．このサンプル・プログラムでは，初期値として7FH（全曜日）を設定します。

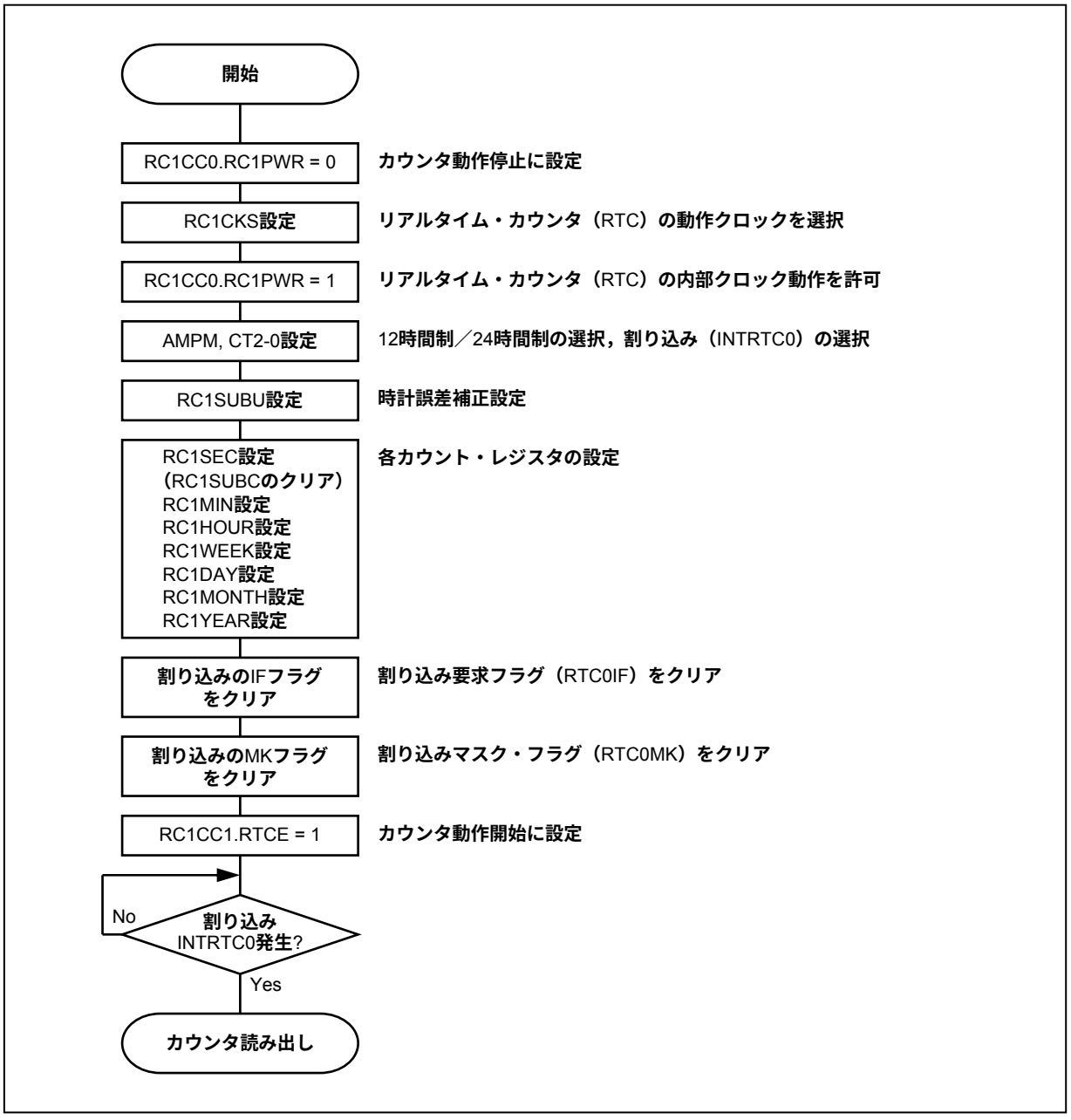
（15）レジスタ設定手順

リアルタイム・カウンタを使用する際のレジスタ設定手順を以下に示します。

① 初期設定

時計機能，定周期割り込み動作をする場合，次のように設定してください。

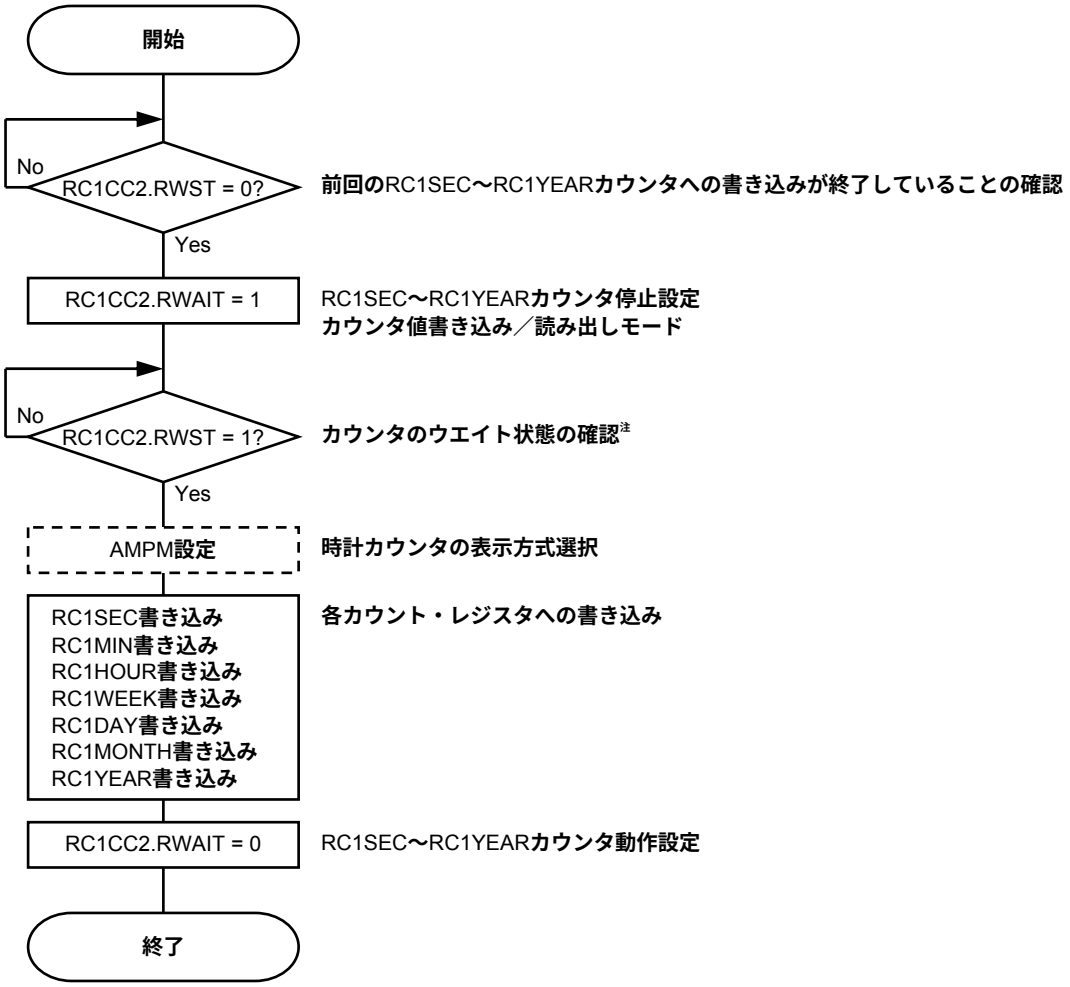
図4－1－15－1　初期設定手順



② リアルタイム・カウンタ動作中の各カウンタの書き換え

リアルタイム・カウンタ動作中に各カウンタを書き換える場合，次のように設定してください。

図4－1－15－2　クロック動作中の各カウンタの書き換え



注．　STOPモードに移行する前には，必ずRWST = 0であることを確認してください。

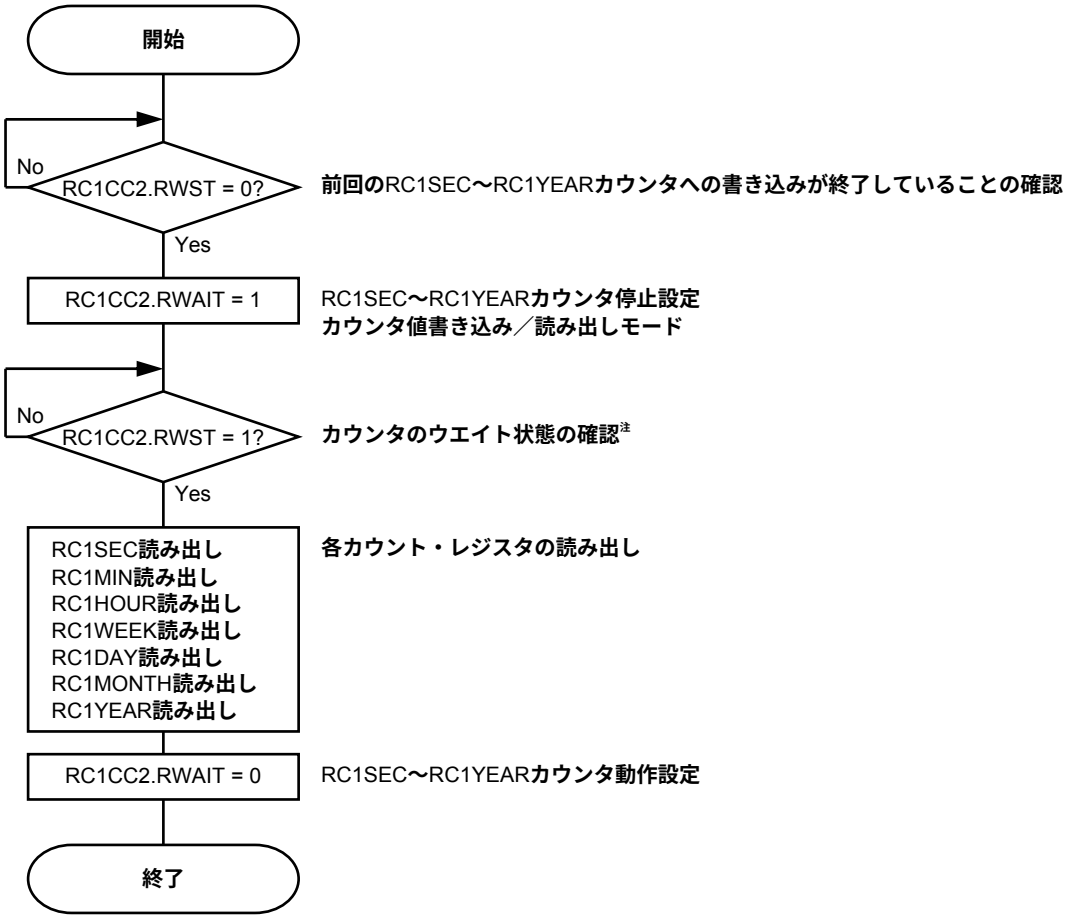
注意．　RWAIT = 1 からRWAIT = 0とするまで1秒以内で行ってください。
RWAIT = 1に設定すると,RC1SEC～RC1YEARの動作を停止します。RWAIT = 1中にRC1SUBCから桁上がりが発生した場合，1回の桁上がりは内部で保持します。しかし2回以上の桁上がりが発生した場合は，桁上がり回数を保持できません。

備考．　RC1SEC, RC1MIN, RC1HOUR, RC1WEEK, RC1DAY, RC1MONTH, RC1YEARの書き換えの順番に制限はありません。
また，すべてのレジスタに書き込む必要はなく，一部のレジスタのみを書き換えても構いません。

③ リアルタイム・カウンタ動作中の各カウンタの読み出し

リアルタイム・カウンタ動作中に各カウンタを読み出す場合，次のように設定してください。

図4－1－15－3　クロック動作中の各カウンタの読み出し



注．　STOPモードに移行する前には，必ずRWST = 0であることを確認してください。

注意．　RWAIT = 1からRWAIT = 0とするまで1秒以内で行ってください。
RWAIT = 1に設定すると，RC1SEC～RC1YEARの動作を停止します。RWAIT = 1中にRC1SUBCから桁上がりが発生した場合，1回の桁上がりは内部で保持します。しかし2回以上の桁上がりが発生した場合は，桁上がり回数を保持できません。

備考．　RC1SEC, RC1MIN, RC1HOUR, RC1WEEK, RC1DAY, RC1MONTH, RC1YEARの読み出しの順番に制限はありません。
また，すべてのレジスタを読み出す必要はなく，一部のレジスタのみを読み出しても構いません。

（16）ソフトウエア記述例

リアルタイム・カウンタの設定をソフトウエアで記述する場合の例として，サンプル・プログラムの一部を次に示します。

なお，この例では下記の動作を設定します。

- リアルタイム・カウンタの動作クロックをサブクロック（f_{XT}）に設定
- 定周期割り込み動作を0.5秒周期に設定
- カウント開始時刻を 2010年4月1日（木）13時00分00秒 に設定
- アラーム割り込み動作を許可しアラーム時刻を 毎日14時00分 に設定
- インターバル動作を約3.9msに設定
- INTRTC0割り込み，およびINTRTC1割り込みを許可

① INTRTC0割り込み，およびINTRTC1割り込みの割り込みハンドラを指定する。
② RC1CC0のビット7（RC1PWR）でリアルタイム・カウンタを動作停止にする。
③ RC1CC0のビット6（RC1CKS）でリアルタイム・カウンタの動作クロックをf_{XT}に設定する。
④ RC1CC0のビット7（RC1PWR）でリアルタイム・カウンタの動作を許可する。
⑤ RC1CC1のビット3（AMPM）で24時間制に設定する。また，ビット0-2（CT0-CT2）で定周期割り込み（INTRTC0）の割り込み周期を0.5秒に1度に設定する。
⑥ RC1SEC，RC1MIN，RC1HOUR，RC1WEEK，RC1DAY，RC1MONTH，およびRC1YEARで，カウント開始時刻（ 10年04月01日木曜日 13時00分00秒 ）を設定する。
⑦ RC1CC2のビット7（WALE）でアラーム一致動作を無効にする。
⑧ RC1ALM，RC1ALH，およびRC1ALWで，アラーム時刻（ 全曜日14時00分 ）を設定する。
⑨ RC1CC2のビット7（WALE）でアラーム一致動作を有効にする。
⑩ RC1CC3のビット0-2（ICT0-ICT2）でインターバル割り込み（INTRTC2）の割り込み周期を約3.9msに設定する。また，ビット7（RINTE）でインターバル割り込みを許可する。
⑪ INTRTC0割り込み要求（RTC0IF），INTRTC1割り込み要求（RTC1IF），およびINTRTC2割り込み要求（RTC2IF）をクリア。
⑫ INTRTC0割り込み，およびINTRTC1割り込みを許可する。
⑬ RC1CC1のビット7（RTCE）でリアルタイム・カウンタのカウント動作を許可する。

```
#pragmainterrupt INTRTC0        fn_IntRtc0      /* 定周期割り込み処理 */
#pragmainterrupt INTRTC1        fn_IntRtc1      /* アラーム割り込み処理 */ } ①

    RC1CC0.7 = 0;           /* リアルタイム・カウンタ動作停止 */ } ②

/*-----------------------------------------------------------------
    定周期割り込み動作の設定
-----------------------------------------------------------------*/
    RC1CC0 = 0b00000000;                                          } ③

    RC1CC0.7 = 1;           /* リアルタイム・カウンタ動作許可 */ } ④

    RC1CC1 = 0b00001001;                                          } ⑤

    /* カウント開始時刻の初期値設定 ( 2010年4月1日(木) 13時00分00秒 ) */
    RC1SEC      = 0x00;
    RC1MIN      = 0x00;
    RC1HOUR     = 0x13;
    RC1WEEK     = 0x04;                                           } ⑥
    RC1DAY      = 0x01;
    RC1MONTH    = 0x04;
    RC1YEAR     = 0x10;

/*-----------------------------------------------------------------
    アラーム割り込み動作の設定
-----------------------------------------------------------------*/
    RC1CC2 = 0b00000000;                                          } ⑦

    /* アラーム時刻の初期値設定( 毎日14時00分 ) */
    RC1ALW = 0x7F;
    RC1ALH = 0x14;                                                } ⑧
    RC1ALM = 0x00;

    RC1CC2.7 = 1;           /* アラーム割り込み許可 */           } ⑨

/*-----------------------------------------------------------------
    インターバルの設定
-----------------------------------------------------------------*/
    RC1CC3 = 0b10000001;                                          } ⑩

    RTC0IF = 0;             /* INTRTC0割り込み要求クリア */
    RTC1IF = 0;             /* INTRTC1割り込み要求クリア */       } ⑪
    RTC2IF = 0;             /* INTRTC2割り込み要求クリア */

    RTC0MK = 0;             /* INTRTC0割り込み許可 */
    RTC1MK = 0;             /* INTRTC1割り込み許可 */
    RTC2MK = 1;             /* INTRTC2割り込み禁止 */             } ⑫

    __EI();             /* 割り込み要求信号の受け付け許可 */

    RC1CC1.7 = 1;                                                 } ⑬
```

4. 2　RTCバックアップ・モードの設定

RTCバックアップ・モードを制御するレジスタを次に示します。

- RTCバックアップ制御レジスタ0（RTCBUMCTL0）
- サブクロック低電力動作制御レジスタ（SOSCAMCTL）

（1）RTCバックアップ制御レジスタ0（RTCBUMCTL0）

RTCバックアップ・モードを制御するレジスタです。

図4－2－1　RTCバックアップ制御レジスタ0（RTCBUMCTL0）のフォーマット

RTCバックアップ制御レジスタ0（RTCBUMCTL0）

アドレス： FFFFFB00H

7	6	5	4	3	2	1	0
RBMEN	0	0	0	0	0	0	RBMSET

RBMEN	RTCバックアップ・モードの制御
0	RTCバックアップ・モード使用禁止
1	RTCバックアップ・モード使用許可

RBMSET	RTCバックアップ・モードの制御
0	RTCバックアップ準備状態を解除
1	RTCバックアップ準備状態に設定 RBMSETビットをセット（1）するとRTCは次の状態に切り替わります。 ・RTC入力クロックはサブクロック（f_{XT}）の分周クロックを選択 ・RTCの端子出力機能停止 ・RTCの時計誤差補正機能停止

注意1．RBMENビットとRBMSETビットを同時にセット（1）しないでください。同時にセット（1）した場合RTCバックアップ・モードが正常に動作しない可能性があります。また，RBMENビットを先にセット（1）してからRBMSETビットをあとにセット（1）してください。

2．RBMENビットが0の状態でRBMSETビットをセット（1）しないでください。
RBMENビットが0の状態でRBMSETビットをセット（1）した場合，RBMSETビットはセット（1）されますが，RTCバックアップ準備状態には設定されません。

3．RTCBUMCTL0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。なお，特定レジスタについての詳細はユーザーズ・マニュアルを参照してください。

4．ビット1-6には必ず0を設定してください。

備考1．RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

2．図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。

(2) サブクロック低電力動作制御レジスタ（SOSCAMCTL）

RTCバックアップ・モード時の低電力動作を行うため，サブクロック（f_{XT}）の低電力制御方法を選択するレジスタです。

図4－2－2　サブクロック低電力動作制御レジスタ（SOSCAMCTL）のフォーマット

サブクロック低電力動作制御レジスタ（SOSCAMCTL）

アドレス： FFFFFB03H

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

AMPHS	サブクロック（f_{XT}）の発振モード選択
0	通常発振
1	超低消費発振

注意1．SOSCAMCTL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。なお，特定レジスタについての詳細はユーザーズ・マニュアルを参照してください。

2．ビット1-7には必ず0を設定してください。

備考1．サブクロック（f_{XT}）の超低消費発振時は，ノイズの影響により発振周期の誤カウントが起こりやすくなります。サブクロック（f_{XT}）の超低消費発振を使用する場合は，ノイズの影響を十分評価したあとに決定してください。

2．RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

3．図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。

(3) ソフトウエア記述例

RTCバックアップ・モードの設定をソフトウエアで記述する場合の例として，サンプル・プログラムの一部を以下に示します。

(a) 初期設定

リセット解除後の初期設定にて，下記の設定を行います。

① INTLVI割り込みの割り込み処理ルーチンを定義する。
② RTCBUMCTL0.RBMEN = 1に設定し，RTCバックアップ・モード使用許可状態にする。また，リセット解除後にRTCバックアップ準備状態となっている場合，ここで解除する。
③ SOSCAMCTL.AMPHS = 0に設定し，サブクロック（f_{XT}）を通常発振に設定する。
④ NOPを実行する。
⑤ 低電圧検出回路を割り込みとしての使用に設定する。また，低電圧検出レベルを2.8Vに設定する。
⑥ リアルタイム・カウンタの定周期割り込み動作を設定する。また，サブクロック（f_{XT}）を動作クロックに設定する。
⑦ INTLVI割り込みを許可する。
⑧ リアルタイム・カウンタの動作を許可する。

```
#pragmainterrupt INTLVI   fn_IntLvi /* 低電圧検出割り込み処理 */ } ①

static void fn_InitRtcBum( void )
{
   __asm("push   r10                    ");
   __asm("mov    0x80,   r10            ");
   __asm("st.b   r10,    PRCMD          ");  } ②
   __asm("st.b   r10,    RTCBUMCTL0     ");
   __asm("pop    r10                    ");

   __asm("st.b   r0,     PRCMD          ");
   __asm("st.b   r0,     SOSCAMCTL      ");  } ③

   __NOP();  } ④

static void fn_InitLvi( void )
{
   LVIMK          = 1;                    /* INTLVI割り込み禁止 */
   LVIIF          = 0;                    /* INTLVI割り込み要求クリア */
   __asm("st.b   r0,     PRCMD ");        /* 低電圧検出時の動作を割り込みに設定 */
   __asm("st.b   r0,     LVIM  ");
   LVIS   = 0b00000000;                  /* 低電圧検出レベルを 2.80V に設定 */

   /* 低電圧検出動作許可 */
   __asm("push   r10           ");
   __asm("mov    0x80,   r10   ");
   __asm("st.b   r10,    PRCMD ");
   __asm("st.b   r10,    LVIM  ");
   __asm("pop    r10           ");                                  } ⑤

static void fn_InitRtc( void )
{
   RC1CC0.7       = 0;              /* リアルタイム・カウンタ動作停止 */

   RC1CC0         = 0b00000000;

   RC1CC0.7       = 1;              /* リアルタイム・カウンタ動作許可 */

   RC1CC1         = 0b00001001;

   /* カウント開始時刻の初期値設定 ( 2010年4月1日(木) 13時00分00秒 ) */
   RC1SEC         = 0x00;
   RC1MIN         = 0x00;
   RC1HOUR        = 0x13;
   RC1WEEK        = 0x04;
   RC1DAY         = 0x01;
   RC1MONTH       = 0x04;
   RC1YEAR        = 0x10;                                           } ⑥

   LVIMK          = 0;              /* INTLVI割り込み許可 */
   __EI();                          /* 割り込み要求信号の受け付け許可 */  } ⑦

   RC1CC1.7 = 1;          /* リアルタイム・カウンタ動作許可 */          } ⑧
```

注意1．RTCバックアップ制御レジスタ0（RTCBUMCTL0），サブクロック低電力動作制御レジスタ（SOSCAMCTL），および低電圧検出レジスタ（LVIM）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。なお，特定レジスタについての詳細はユーザーズ・マニュアルを参照してください。

2．低電圧検出回路についての詳細はユーザーズ・マニュアルを参照してください。

(b) RTCバックアップ準備状態に移行するときの設定

INTLVI割り込みの割り込みルーチンにて，低電圧状態（LVIM.LVIF = 1）[注1]であることを確認し，下記の設定を行います。

① PLLを停止し，CPUクロックを低電圧状態に対応した動作クロック（5MHz）に設定する。
② DMA動作を禁止する[注2]
③ NMI割り込み，およびINTLVI割り込み以外のマスカブル割り込みを禁止する。
④ SOSCAMCTL.AMPHS = 1に設定し，サブクロック（f_{XT}）を超低消費発振に設定する。
⑤ RTCBUMCTL0.RBMSET = 1に設定し，RTCバックアップ準備状態に設定する。
⑥ 通常STOPモードに設定する（V_{DD}電圧の供給が停止するとRTCバックアップ・モードへ移行）。

```
__interrupt void fn_IntLvi( void )
{
    if( LVIF )
    {
        SELPLL = 0;
        PLLON  = 0;                                          ①

        E00    = 0;
        E11    = 0;
        E22    = 0;
        E33    = 0;                                          ②

        INTF0 &= 0b01111000;
        INTR0 &= 0b01111000;

        IMR3 = 0b1111111111111111;
        IMR2 = 0b1111111111111111;
        IMR1 = 0b1111111111111111;
        IMR0 = 0b1111111111111110;                           ③

        asm("push    r10                    ");
        asm("mov     0x01,   r10            ");
        asm("st.b    r10,    PRCMD          ");
        asm("st.b    r10,    SOSCAMCTL      ");
        asm("pop     r10                    ");              ④

        asm("mov     0x81,   r10            ");
        asm("st.b    r10,    PRCMD          ");
        asm("st.b    r10,    RTCBUMCTL0     ");              ⑤
```

```
static void fn_SetStandby( void )
{
    PSMR = 0b0000001;

    asm("push    r10            ");
    asm("mov     0x02,   r10    ");
    asm("st.b    r10,    PRCMD  ");
    asm("st.b    r10,    PSC    ");
    asm("pop     r10            ");                          ⑥
```

注1. このサンプル・プログラムでは，2.2 V＜V_{DD}＜2.8 Vである状態を低電圧状態としています。

2. INTLVI割り込み処理ルーチンで，DMA動作を禁止する前にDMA動作が発生した場合，DMA転送が完了する前に電源電圧が最低保証電圧に達すると，RTCバックアップ・モードの設定はできなくなります。

注意. サブクロック低電力動作制御レジスタ（SOSCAMCTL）およびRTCバックアップ制御レジスタ0（RTCBUMCTL0）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。なお，特定レジスタについての詳細はユーザーズ・マニュアルを参照してください。

(c) RTCバックアップ準備状態を解除するときの設定

INTLVI割り込みの割り込みルーチンにて，低電圧から復帰した状態（LVIM.LVIF = 0）であることを確認し，下記の設定を行います。

① SOSCAMCTL.AMPHS = 0に設定し，サブクロック（f_{XT}）を通常発振に設定する。
② RTCBUMCTL0.RBMSET = 0に設定し，RTCバックアップ準備状態を解除する。
③ NOPを実行する。
④ 使用するマスカブル割り込み（INTLVI割り込み以外）を許可する。
⑤ CPUクロックを通常動作用の動作クロック（5MHz × 4逓倍 = 20MHz）に設定する。

```
__interrupt void fn_IntLvi( void )
{
    if( LVIF )
    {
        LVIM.LVIF = 1の場合については省略します。
    }
    else
    {
        asm("st.b       r0,     PRCMD       ");
        asm("st.b       r0,     SOSCAMCTL   ");   ①

        asm("push       r10                 ");
        asm("mov        0x80,   r10         ");
        asm("st.b       r10,    PRCMD       ");
        asm("st.b       r10,    RTCBUMCTL0  ");   ②
        asm("pop        r10                 ");

        NOP();                                    ③

        KRIF    = 0;
        RTC0IF  = 0;
        RTC1IF  = 0;
        KRMK    = 0;                              ④
        RTC0MK = 0;
        RTC1MK = 0;

        PLLON  = 1;
        while( LOCK )
        {
            __NOP();                              ⑤
        }
        SELPLL = 1;
```

注意． サブクロック低電力動作制御レジスタ（SOSCAMCTL），RTCバックアップ制御レジスタ0（RTCBUMCTL0），およびプロセッサ・クロック・コントロール・レジスタ（PCC）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。なお，特定レジスタについての詳細はユーザーズ・マニュアルを参照してください。

4. 3 時計誤差補正について

リアルタイム・カウンタの時計誤差補正について説明します。

(1) 時計誤差補正レジスタ（RC1SUBU）

リアルタイム・カウンタの時計誤差補正を行う場合，**時計誤差補正レジスタ（RC1SUBU）を使用します。時計誤差補正レジスタ（RC1SUBU）は，サブカウント・レジスタ（RC1SUBC）から秒カウンタ・レジスタへオーバフローする値（基準値：7FFFH）を変化させることにより，時計の進みや遅れをより高精度に補正することができるレジスタです。**

図4－3－1 時計誤差補正レジスタ（RC1SUBU）のフォーマット

時計誤差補正レジスタ（RC1SUBU）

アドレス： FFFFFAD9H

7	6	5	4	3	2	1	0
DEV	F6	F5	F4	F3	F2	F1	F0

DEV	時計誤差補正のタイミングの設定
0	RC1SEC（秒カウンタ）が00, 20, 40秒のとき（20秒ごと）に時計誤差補正
1	RC1SEC（秒カウンタ）が00秒のとき（60秒ごと）に時計誤差補正

F6	時計誤差補正値の設定
0	F5-F0ビットで設定した値分，RC1SUBCのカウント値を増加（＋補正） 増加値計算式：（F5-F0ビットの設定値－1）×2
1	F5-F0ビットで設定した値分，RC1SUBCのカウント値を減少（－補正） 減少値計算式：（F5-F0ビットの設定値の反転値データ＋1）×2

注意． F6-F0ビットの値が {1/0, 0, 0, 0, 0, 0, 1/0} のときは，時計誤差補正は行いません。

備考1．RTCバックアップモード時は，時計の誤差補正は停止します。

2．RV_{DD}の電源投入により，00Hになります。また，RV_{DD}の電源投入以外の要因でリセットが発生した場合は値を保持します。

3．このサンプル・プログラムでは，初期値として00Hを設定します。

(2) 時計誤差補正例

発振子に32.768 kHzより速い“＋誤差”が生じた場合は，RC1SUBCのカウント値を増やす事で正確に時計をカウントできます。また同様に発振子に32.768 kHzより遅い“－誤差”が生じた場合は，RC1SUBCのカウント値を減らすことで正確に時計をカウントできます。

RC1SUBCの補正値を決定するのがRC1SUBU.F6-F0ビットです。

F6ビットでRC1SUBCの増加／減少を決定し，F5-F0ビットで値を決定します。

① RC1SUBCのカウント増加

F6ビット=0にすることによって，F5-F0ビットで設定した値分，RC1SUBCのカウント値を増加します。

増加値計算式：（F5-F0ビット設定値－1）×2

【RC1SUBCのカウント増加例：F6ビット = 0】

F5-F0ビットに15H（010101B）を設定した場合

（15H－1）×2 = 40（RC1SUBCカウント値を40増）

RC1SUBCカウント値 = 32768＋40＝32808

② RC1SUBCのカウント減少

F6ビット= 1にすることによって，F5-F0ビット設定した値の反転値分，RC1SUBCのカウント値を減少します。

減少値計算式：（F5-F0ビット設定値の反転値＋1）×2

【RC1SUBCのカウント増加例：F6ビット = 1】

F5-F0ビットに15H（010101B）を設定した場合

15H（010101B）の反転データ= 2AH（101010B）

（2AH＋1）×2 = 86（RC1SUBCカウント値を86減）

RC1SUBCカウント値 = 32768－86 = 32682

③ DEVビットについて

DEVビットは，F6-F0ビットでの設定が有効となるタイミングを決定します。

F6-F0ビットで設定した値は，毎回RC1SUBCカウント値に反映されるわけではなく次のタイミングで，反映されます。

表4－2　DEVビットの設定

DEVビットの場合	RC1SUBCへの反映タイミング
0の場合	RC1SECが00，20，40秒のとき
1の場合	RC1SECが00秒のとき

【F6-F0ビットに0010101Bを設定した場合の例】

・DEVビット= 0の場合

RC1SUBCカウント値は，00秒，20秒，40秒のとき「32808」

それ以外のとき「32768」

・DEVビット= 1の場合

RC1SUBCカウント値は，00秒のとき「32808」

それ以外のとき「32768」

このように，毎秒RC1SUBCカウント値を補正するのではなく，20秒ごと，60秒ごとに補正しているのは，発振子がもつ偏差幅に合わせているためです。

実際に補正できる発振子の周波数範囲は次のようになります。

・DEVビット = 0のとき：32.76180000 kHz～32.77420000 kHz

・DEVビット = 1のとき：32.76593333 kHz～32.77006667 kHz

DEVビット = 0の方が，DEVビット = 1より3倍広い周波数範囲を補正できます。

ただしDEVビット = 1の方が，3倍の精度で周波数を設定できます。

表4－3，表4－4に，DEVビット，F6-F0ビットの設定値と，そのときに補正できる周波数の一覧を示します。

表4－3　DEVビット = 0のときの補正できる周波数範囲

F6	F5-F0	RC1SUBC補正値	接続クロック周波数（定常偏差込み）
0	000000	補正なし	－
0	000001	補正なし	－
0	000010	20秒に1度，RC1SUBCカウント値を＋2	32.76810000 kHz
0	000011	20秒に1度，RC1SUBCカウント値を＋4	32.76820000 kHz
0	000100	20秒に1度，RC1SUBCカウント値を＋6	32.76830000 kHz
⋮			
0	111011	20秒に1度，RC1SUBCカウント値を＋120	32.77400000 kHz
0	111110	20秒に1度，RC1SUBCカウント値を＋122	32.77410000 kHz
0	111111	20秒に1度，RC1SUBCカウント値を＋124	32.77420000 kHz（上限）
1	000000	補正なし	－
1	000001	補正なし	－
1	000010	20秒に1度，RC1SUBCカウント値を－124	32.76180000 kHz（下限）
1	000011	20秒に1度，RC1SUBCカウント値を－122	32.76190000 kHz
1	000100	20秒に1度，RC1SUBCカウント値を－120	32.76200000 kHz
⋮			
1	111011	20秒に1度，RC1SUBCカウント値を－6	32.76770000 kHz
1	111110	20秒に1度，RC1SUBCカウント値を－4	32.76780000 kHz
1	111111	20秒に1度，RC1SUBCカウント値を－2	32.76790000 kHz

表4－4　DEVビット = 1のときの補正できる周波数範囲

F6	F5-F0	RC1SUBC補正値	接続クロック周波数（定常偏差込み）
0	000000	補正なし	－
0	000001	補正なし	－
0	000010	60秒に1度，RC1SUBCカウント値を＋2	32.76803333 kHz
0	000011	60秒に1度，RC1SUBCカウント値を＋4	32.76806667 kHz
0	000100	60秒に1度，RC1SUBCカウント値を＋6	32.76810000 kHz
⋮			
0	111011	60秒に1度，RC1SUBCカウント値を＋120	32.77000000 kHz
0	111110	60秒に1度，RC1SUBCカウント値を＋122	32.77003333 kHz
0	111111	60秒に1度，RC1SUBCカウント値を＋124	32.77006667 kHz（上限）
1	000000	補正なし	－
1	000001	補正なし	－
1	000010	60秒に1度，RC1SUBCカウント値を－124	32.76593333 kHz（下限）
1	000011	60秒に1度，RC1SUBCカウント値を－122	32.76596667 kHz
1	000100	60秒に1度，RC1SUBCカウント値を－120	32.76600000 kHz
⋮			
1	111011	60秒に1度，RC1SUBCカウント値を－6	32.76790000 kHz
1	111110	60秒に1度，RC1SUBCカウント値を－4	32.76793333 kHz
1	111111	60秒に1度，RC1SUBCカウント値を－2	32.76796667 kHz

(3) ソフトウエア記述例

このサンプル・プログラムでは，時計誤差補正機能の設定用関数が用意されています。

この関数では，引数で指定された1バイトのデータを時計誤差補正レジスタ（RC1SUBU）に設定します。

図4－3－2　時計誤差補正機能の設定用関数

```
static void fn_RtcCorrect( unsigned char ucReg )
{
        RC1SUBU = ucReg;

}
```

この関数の使用例を以下に示します。

・20秒に1度，RC1SUBCカウント値を＋2に補正する場合（接続クロック周波数：32.76810000 kHz）

```
        fn_RtcCorrect( 0b00000010 );
```

・60秒に1度，RC1SUBCカウント値を－4に補正する場合（接続クロック周波数：32.76793333 kHz）

```
        fn_RtcCorrect( 0b11111110 );
```

第5章　関連資料

資　料　名	資料番号
V850ES/JG3-L　ハードウエア編　ユーザーズ・マニュアル	U18953J
V850ES/JG3-L（USBコントローラ内蔵製品）　ハードウエア編　ユーザーズ・マニュアル	U20305J
V850ES アーキテクチャ編	U15943J
PM+ Ver.6.30 ユーザーズ・マニュアル	U18416J
CA850 Ver.3.20 Cコンパイラ・パッケージ 操作編	U18512J
CA850 Ver.3.20 Cコンパイラ・パッケージ C言語編	U18513J
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	U18515J
QB-MINI2 プログラミング機能付きオンチップ・デバッグ・エミュレータ	U18371J
ID850QB Ver.3.40 統合デバッガ 操作編	U18604J

ドキュメント検索URL　http://www2.renesas.com/micro/ja/documentation.html

付録A　プログラム・リスト

プログラム・リスト例として，ソース・プログラムを次に示します。

● main.c （C言語版）

```
/*********************************************************************************

   Renesas Electronics     V850ES/Jx3-Lシリーズ

*********************************************************************************
       V850ES/JG3-Lシリーズ          サンプル･プログラム
*********************************************************************************
       リアルタイム・カウンタ（RTCバックアップ・モード）編
*********************************************************************************
【履歴】
       2010.07  新規作成
*********************************************************************************

【概要】

このサンプル・プログラムは，リアルタイム・カウンタの時計機能，アラーム機能，
およびRTCバックアップ・モードなどを使用し，電源電圧（VDD）が供給停止となっても
カウントが継続するアラーム付き時計の動作を実現するものです。


＜初期設定の主な内容＞

  （オプション・バイトでの指定）
    ・リセット解除後の発振安定時間を指定
  （リセット解除後の初期化処理での設定）
    ・RTCバックアップ・モードの初期設定
      ・サブクロックの発振モードを通常発振に設定
      ・RTCバックアップ・モードの使用を許可
    ・システム・ウエイト・コントロール・レジスタを1ウエイトに設定
    ・オンチップ・デバッグ・モード・レジスタ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
    ・入出力ポートの設定
      ・P50-P53（KR0-KR3）をキー入力用に設定
      ・P70をブザー制御用に設定
```

- P90-P96をLCDモジュール制御用に設定
- 未使用ポートの設定

・低電圧検出回路の設定
- 低電圧検出レベルを2.8Vに設定
- 低電圧検出動作の許可
- 電源電圧が2.8V以上になるまで待つ

・PLLモード（5 MHz×4逓倍 = 20 MHz動作）に設定

・スタンバイ・モード解除時のCPUクロックの発振安定時間を約1.6msに設定

・リアルタイム・カウンタの設定
- カウント開始時刻を 2010年4月1日（木） 13時00分00秒 に設定
- アラーム時刻を 毎日14時00分 に設定
- 定周期割り込み動作を0.5秒周期に設定
- アラーム割り込み動作を許可
- インターバル動作を約3.9msに設定
- カウント動作の開始

・LCDモジュールの初期設定

・キー割り込み（KR0-KR3）の動作許可

・割り込みの設定
- INTKR割り込みの許可
- INTRTC0割り込みの許可
- INTRTC1割り込みの許可
- INTLVI割り込みの許可
- マスカブル割り込み要求信号の受け付けを許可

＜初期設定以降の処理内容＞

初期設定完了後，STOPモードに移行します。
以降は，割り込みの発生により処理を実行します。

・0.5秒周期のINTRTC0割り込みが発生した場合，時計表示の更新などを行います。

・アラーム発生によるINTRTC1割り込みが発生した場合，ブザー出力を行います。

・キー入力によるINTKR割り込みが発生した場合，約3.9ms周期のINTRTC2割り込みを使用したキーのサンプリングを開始します。サンプリングにより有効なキー入力が検出された場合，時計表示の更新やリアルタイム・カウンタのカウント値変更などを行います。

・低電圧検出によるINTLVI割り込みが発生した場合，RTCバックアップ準備状態の設定を行い，STOPモードへ移行します。その後，電源電圧（VDD）が供給停止となった場合，RTCバックアップ・モードとなります。また，STOPモード移行後に低電圧からの復帰によるINTLVI割り込みが発生した場合，RTCバックアップ準備状態を解除し，通常動作に戻ります。

＜入出力ポートの設定＞

入力ポート：P50-P53

出力ポート：P70, P90-P96

※未使用のポートはすべて出力ポートに設定しておく

```
*********************************************************************************/



/*================================================================================
        前処理指令（#pragma指令）
================================================================================*/
#pragma         ioreg                       /* 周辺I/Oレジスタ名を記述可能にする */

/*================================================================================
        命令定義
================================================================================*/
#define __NOP() __asm("nop")                /* NOP命令を__NOP()として記述可能にする */

/*================================================================================
        割り込みハンドラ指定
================================================================================*/
#pragma         interrupt INTRTC0 fn_IntRtc0        /* 定周期割り込み処理 */
#pragma         interrupt INTRTC1 fn_IntRtc1        /* アラーム割り込み処理 */
#pragma         interrupt INTLVI  fn_IntLvi         /* 低電圧検出割り込み処理 */
#pragma         interrupt INTKR   fn_IntKr          /* キー割り込み処理 */

/*================================================================================
    関数プロトタイプ宣言
================================================================================*/
                void main( void );                  /* メイン処理 */
static void fn_InitFirst( void );                   /* 最初に設定するレジスタの設定処理 */
static void fn_InitPort( void );                    /* ポートの初期設定処理 */
static void fn_InitLvi( void );                     /* 低電圧検出処理 */
static void fn_InitClock( void );                   /* クロックの初期設定処理 */
static void fn_InitRtc( void );                     /* リアルタイム・カウンタの初期設定処理 */
static void fn_InitRtcBum( void );                  /* RTCバックアップ・モードの初期設定処理 */
static void fn_WriteRtcCounter( unsigned char ucInc );      /* RTCのカウント書き換え処理 */
static void fn_WriteRtcAlerm( unsigned char ucInc );        /* RTCのアラーム書き換え処理 */
static void fn_RtcCorrect( unsigned char ucReg );           /* 時計誤差補正処理 */
static void fn_KeySampling( void );                         /* キーのサンプリング処理 */
static void fn_SetStandby( void );                          /* スタンバイ・モード移行処理 */
static void fn_Display( void );                             /* 表示更新処理 */
```

```
static unsigned char fn_BcdCount
( unsigned char ucNum, unsigned char ucInc,
  unsigned char ucMin, unsigned char ucMax );               /* BCD値のカウント処理 */

/* 外部参照 */
extern void fn_InitLcd( void );                             /* LCDモジュールの初期設定処理 */
extern void fn_LcdChrDisp
( unsigned char ucChar, unsigned char ucPos, unsigned char ucType ); /* キャラクタ表示処理 */
extern void fn_LcdCurDisp( unsigned char ucPos );                    /* カーソル表示処理 */

/*==============================================================================
        使用する変数／定数の定義
==============================================================================*/
unsigned char ucMode;                                       /* 状態種別 */
        #define  MODE_WATCH                 0x00            /* 通常動作状態 */
        #define  MODE_TIMESETUP             0x01            /* 日付・時刻設定時 */
        #define  MODE_ALARMSETUP            0x02            /* アラーム設定時 */
        #define  MODE_RESET                 0x03            /* リセット復帰状態 */
unsigned char ucDisplay;                                    /* 表示種別 */
        #define  DISP_WATCH                 0x00            /* 時計表示 */
        #define  DISP_RESET                 0x01            /* リセット復帰表示 */
unsigned char ucSetupTime;                                  /* 時刻設定項目 */
        #define  SETUP_TMNONE               0xff            /* 設定項目なし */
        #define  SETUP_TMYEAR               0x00            /* 年 */
        #define  SETUP_TMMONTH              0x01            /* 月 */
        #define  SETUP_TMDAY                0x02            /* 日 */
        #define  SETUP_TMWEEK               0x03            /* 曜日 */
        #define  SETUP_TMHOUR               0x04            /* 時 */
        #define  SETUP_TMMIN                0x05            /* 分 */
        #define  SETUP_TMSEC                0x06            /* 秒 */
        #define  SETUP_TIME_NUM             7               /* 項目数 */
unsigned char ucSetupAlarm;                                 /* アラーム設定項目 */
        #define  SETUP_ALNONE               0xff            /* 設定項目なし */
        #define  SETUP_ALWEEK               0x00            /* アラーム 曜日 */
        #define  SETUP_ALHOUR               0x01            /* アラーム 時 */
        #define  SETUP_ALMIN                0x02            /* アラーム 分 */
        #define  SETUP_ALARM_NUM            3               /* 項目数 */
unsigned char ucColonBlink;                                 /* ":"表示 (False:消灯 True:点灯) */
unsigned char ucBlinkDispTimer;                             /* 交互表示用タイマ */
        #define  BLINKDISPTIME              3               /* 切り替えの間隔(= n * 0.5s ) */
unsigned char ucDispReq;                                    /* 表示更新要求 (False:なし True:あり) */

unsigned char ucKeySamplingCounter;                         /* キーのサンプリング用カウンタ */
```

```
        #define  KEYSAMPLINGCOUNT  3                /* サンプリング回数( 約3.9ms周期 ) */
unsigned char ucCodeBuffer;                         /* コード用バッファ */
unsigned char ucKeyCode;                            /* キー・コード */
        #define  KEYCODEOFF        0x00             /* オフ */
        #define  KEYCODE1          0x01             /* キー1 */
        #define  KEYCODE2          0x02             /* キー2 */
        #define  KEYCODE3          0x03             /* キー3 */
        #define  KEYCODE4          0x04             /* キー4 */
        #define  KEYCODEMLT        0xff             /* 多重押し */
unsigned char ucKeyEdgeInfo;                        /* キーの検出通知(False:なし True:あり) */
unsigned char ucKeyValidInfo;                       /* キーの有効入力通知(False:なし True:あり) */

unsigned char ucBuzzerTimer;                        /* ブザー用タイマ */
        #define  BUZZER_TIME       2                /* ブザー出力期間(= n * 0.5s ) */



/*****************************************************************************
* Title  : メイン処理
*****************************************************************************
* Module : void main( void )
* Arg    :
* Ret    :
*----------------------------------------------------------------------------
* Note   : このサンプル・プログラムのメイン処理です。
*****************************************************************************/
void main( void )
{
        __DI();                     /* 割り込み要求信号の受け付け禁止 */

/*---------------------------------------------------------------------
        周辺I/O機能の初期設定
---------------------------------------------------------------------*/
        fn_InitRtcBum();            /* RTCバックアップ・モードの初期設定処理 */
        RC1CC2.0 = 0;               /* カウント・レジスタのカウント動作許可 */
                                    /* (※カウント停止中にCPUリセットされた場合への対応) */

        fn_InitFirst();             /* 最初に設定するレジスタの設定処理 */
        fn_InitPort();              /* ポートの初期設定処理 */
        fn_InitLvi();               /* 低電圧検出処理 */
        fn_InitClock();             /* クロックの初期設定処理 */
```

```
        /* リアルタイム・カウンタが初期状態の場合 */
        if( !RC1CC1.7 )
        {
                fn_InitRtc();       /* リアルタイム・カウンタの初期設定処理 */
        }

        fn_InitLcd();               /* LCDモジュールの初期設定処理 */

/*-----------------------------------------------------------------------
        使用する変数の初期値設定
-----------------------------------------------------------------------*/
        ucKeySamplingCounter = KEYSAMPLINGCOUNT;    /* キーのサンプリング回数リセット */
        ucCodeBuffer         = KEYCODEOFF;          /* コード用バッファ:オフ */
        ucKeyCode            = KEYCODEOFF;          /* キー・コード:オフ */
        ucKeyEdgeInfo        = 0;                   /* キーの検出:なし */
        ucKeyValidInfo       = 0;                   /* キーの有効入力:なし */
        ucBuzzerTimer        = 0;                   /* ブザー用タイマ停止 */
        ucSetupTime         = SETUP_TMNONE;         /* 時刻設定なし */
        ucSetupAlarm         = SETUP_ALNONE;        /* アラーム設定なし */
        ucColonBlink         = 1;                   /* ":"表示 点灯 */
        ucMode              = MODE_RESET;           /* リセット復帰状態に設定 */
        ucDisplay            = DISP_RESET;          /* リセット復帰表示に切り替え */
        ucBlinkDispTimer = BLINKDISPTIME;           /* 交互表示用タイマ カウント開始 */
        ucDispReq            = 1;                   /* 表示更新要求発行 */

/*-----------------------------------------------------------------------
        使用する割り込みの設定
-----------------------------------------------------------------------*/
        KRM      = 0b00001111;                      /* キー割り込み(KR0-KR3)動作許可 */

        KRIF     = 0;                               /* INTKR割り込み要求クリア */
        RTC0IF   = 0;                               /* INTRTC0割り込み要求クリア */
        RTC1IF   = 0;                               /* INTRTC1割り込み要求クリア */
        RTC2IF   = 0;                               /* INTRTC2割り込み要求クリア */
        LVIIF    = 0;                               /* INTLVI割り込み要求クリア */
        KRMK     = 0;                               /* INTKR割り込み許可 */
        RTC0MK   = 0;                               /* INTRTC0割り込み許可 */
        RTC1MK   = 0;                               /* INTRTC1割り込み許可 */
        RTC2MK   = 1;                               /* INTRTC2割り込み禁止 */
        LVIMK    = 0;                               /* INTLVI割り込み許可 */

        CB3RMK   = 0;                               /* INTCB3R割り込み許可(オンチップ・デバッグ用) */
```

```
        __EI();                                         /* 割り込み要求信号の受け付け許可 */

        RC1CC1.7 = 1;                                   /* リアルタイム・カウンタ動作許可 */

/*------------------------------------------------------------------------
        メイン・ループ
------------------------------------------------------------------------*/
        while (1)
        {
                /* キーの検出および表示更新がないことを確認 */
                if( !ucKeyEdgeInfo && !ucDispReq )
                {
                        fn_SetStandby();            /* スタンバイ・モードに移行 */
                }

                /*================================================================*/
                /*      キーのサンプリング                                        */
                /*================================================================*/
                /* 約3.9ms周期 */
                if( RTC2IF )
                {
                        RTC2IF = 0;                         /* INTRTC2割り込み要求クリア */

                        /* キーの検出あり */
                        if( ucKeyEdgeInfo )
                        {
                                fn_KeySampling();           /* キーのサンプリング処理 */
                        }
                }

                /*================================================================*/
                /*      キー処理                                                  */
                /*================================================================*/
                /* キーの有効入力あり */
                if( ucKeyValidInfo )
                {
                        ucKeyValidInfo = 0;         /* キーの有効入力通知 クリア */

                        /* 状態種別判定 */
                        switch( ucMode )
                        {
```

```
/*-------------------------------------------------------*/
/*        通常動作状態                                   */
/*-------------------------------------------------------*/
case MODE_WATCH:
        /* キー・コード判定 */
        switch( ucKeyCode )
        {
                /* キー1 */
                case KEYCODE1:
                        /* 日付・時刻設定に移行 */
                        ucMode = MODE_TIMESETUP;
                        /* 設定項目を年に設定 */
                        ucSetupTime = SETUP_TMYEAR;
                        /* 表示更新要求発行 */
                        ucDispReq = 1;
                        break;
                /* キー2/3/4 */
                default:
                        /* NOP */
                        break;
        }
        break;
/*-------------------------------------------------------*/
/*        日付・時刻設定時                               */
/*-------------------------------------------------------*/
case MODE_TIMESETUP:
        /* キー・コード判定 */
        switch( ucKeyCode )
        {
                /* キー1 */
                case KEYCODE1:
                        /* アラーム設定に移行 */
                        ucMode = MODE_ALARMSETUP;
                        /* 設定項目をアラーム曜日に設定 */
                        ucSetupAlarm = SETUP_ALWEEK;
                        ucSetupTime = SETUP_TMNONE;
                        break;
                /* キー2 */
                case KEYCODE2:
                        /* 時刻 アップ */
                        fn_WriteRtcCounter( 0 );
                        break;
                /* キー3 */
```

```
                case KEYCODE3:
                        /* 時刻 ダウン */
                        fn_WriteRtcCounter( 1 );
                        break;
                /* キー4 */
                case KEYCODE4:
                        /* 次の項目に変更 */
                        ucSetupTime++;
                        ucSetupTime %= SETUP_TIME_NUM;
                        break;
                default:
                        break;
        }
        ucDispReq = 1;    /* 表示更新要求発行 */
        break;
/*-------------------------------------------------------*/
/*      アラーム設定時                                   */
/*-------------------------------------------------------*/
case MODE_ALARMSETUP:
        /* キー・コード判定 */
        switch( ucKeyCode )
        {
                /* キー1 */
                case KEYCODE1:
                        /* 通常動作状態に移行 */
                        ucMode = MODE_WATCH;
                        /* 設定項目:なし */
                        ucSetupAlarm = SETUP_ALNONE;
                        break;
                /* キー2 */
                case KEYCODE2:
                        /* アラーム アップ */
                        fn_WriteRtcAlerm( 0 );
                        break;
                /* キー3 */
                case KEYCODE3:
                        /* アラーム ダウン */
                        fn_WriteRtcAlerm( 1 );
                        break;
                /* キー4 */
                case KEYCODE4:
                        /* 次の項目に変更 */
                        ucSetupAlarm++;
```

```
                                                ucSetupAlarm %= SETUP_ALARM_NUM;
                                                break;
                                        default:
                                                break;
                                }
                                ucDispReq = 1;    /* 表示更新要求発行 */
                                break;
                        /*-----------------------------------------------------*/
                        /*      リセット復帰状態                               */
                        /*-----------------------------------------------------*/
                        case MODE_RESET:
                                ucMode = MODE_WATCH;       /* 通常動作状態に移行 */
                                ucDisplay = DISP_WATCH;    /* 時計表示に変更 */
                                ucBlinkDispTimer = 0;      /* 交互表示の停止 */
                                ucDispReq = 1;             /* 表示更新要求発行 */
                                break;
                        default:
                                break;
                }
        }

        /*=============================================================*/
        /*      表示更新処理                                           */
        /*=============================================================*/
        /* 表示更新要求あり */
        if( ucDispReq )
        {
                ucDispReq = 0;    /* 表示更新要求クリア */
                fn_Display();     /* 表示更新 */
        }
    }
}

/*********************************************************************
*      Title    : 最初に設定するレジスタの設定処理
*********************************************************************
*      Module   : static void fn_InitFirst( void )
*      Arg              :
*      Ret              :
*--------------------------------------------------------------------
*      Note     : VSWC, OCDM, RCM, WDTM2レジスタを設定します。
*********************************************************************/
static void fn_InitFirst( void )
```

```
{
/*--------------------------------------------------------------------
        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
--------------------------------------------------------------------*/
        VSWC = 0x01;                                    /* ウエイト数:1 に設定 */

/*--------------------------------------------------------------------
        オンチップ・デバッグ・モード・レジスタの設定
--------------------------------------------------------------------*/
        __asm("st.b       r0,       PRCMD     ");       /* 通常動作モードに設定 */
        __asm("st.b       r0,       OCDM      ");

/*--------------------------------------------------------------------
        ウォッチドッグ・タイマ2の設定
--------------------------------------------------------------------*/
        RSTOP = 1;                                      /* 内蔵発振器の停止設定 */
        WDTM2 = 0b00000000;                             /* ウォッチドッグ・タイマ2動作停止設定     */

}

/*********************************************************************
*       Title    : ポートの初期設定処理
*********************************************************************
*       Module   : static void fn_InitPort( void )
*       Arg              :
*       Ret              :
*--------------------------------------------------------------------
*       Note     : 入出力ポートの設定を行います。
*********************************************************************/
static void fn_InitPort( void )
{
/*--------------------------------------------------------------------
        ポート0の設定
--------------------------------------------------------------------*/
        P0       = 0b00000000;                                  /* P02-P06の出力データを0に設定 */
        PM0      = 0b10000011;                                  /* P02-P06を出力モードに設定 */
                                                                /* P02-P06:未使用 */

/*--------------------------------------------------------------------
        ポート1の設定
--------------------------------------------------------------------*/
        P1       = 0b00000000;                                  /* P10-P11の出力データを0に設定 */
        PM1      = 0b11111100;                                  /* P10-P11を出力モードに設定 */
```

```
                                                    /* P10-P11:未使用 */


/*--------------------------------------------------------------------
        ポート3の設定
--------------------------------------------------------------------*/
        P3      = 0b0000000000000000;               /* P30-P39の出力データを0に設定 */
        PM3     = 0b1111110000000000;               /* P30-P39を出力モードに設定 */
                                                    /* P30-P39:未使用 */

/*--------------------------------------------------------------------
        ポート4の設定
--------------------------------------------------------------------*/
        P4      = 0b00000000;                       /* P40-P42の出力データを0に設定        */
        PM4     = 0b11111000;                       /* P40-P42を出力モードに設定 */
                                                    /* P40-P42:未使用 */

/*--------------------------------------------------------------------
        ポート5の設定
--------------------------------------------------------------------*/
        P5      = 0b00001111;                       /* P50-P53の出力データを1に設定 */
                                                    /* P54-P55の出力データを0に設定 */
        PM5     = 0b11001111;                       /* P50-P53を入力モードに設定 */
                                                    /* P54-P55を出力モードに設定 */
        PFCE5   = 0b00000000;                       /* P50-P53をKR0-KR3に設定 */
        PFC5    = 0b00001111;
        PMC5    = 0b00001111;
                                                    /* P50-P53:キー入力用に使用 */
                                                    /* P54-P55:未使用 */

/*--------------------------------------------------------------------
        ポート7の設定
--------------------------------------------------------------------*/
        P7L     = 0b00000001;                       /* P70の出力データを1に設定 */
                                                    /* P71-P77の出力データを0に設定 */
        P7H     = 0b00000000;                       /* P78-P711の出力データを0に設定 */
        PM7L    = 0b00000000;                       /* P70-P77を出力モードに設定 */
        PM7H    = 0b11110000;                       /* P78-P711を出力モードに設定 */
                                                    /* P70:ブザー制御用に使用 */
                                                    /* P71-P711:未使用 */
```

```
/*--------------------------------------------------------------------
        ポート9の設定
--------------------------------------------------------------------*/
        P9      = 0b0000000000000000;           /* P90-P915の出力データを0に設定 */
        PM9     = 0b0000000000000000;           /* P90-P915を出力モードに設定 */
                                                /* P90-P96:LCDモジュール制御用に使用 */
                                                /* P97-P915:未使用 */
                                                /* ※オンチップ・デバッグの通信用に */
                                                /*   CSIB3を使用する場合，PFC910-PFC912 */
                                                /*   およびPMC910-PMC912は変更しないで */
                                                /*   ください。 */

/*--------------------------------------------------------------------
        ポートCMの設定
--------------------------------------------------------------------*/
        PCM.1   = 0;                            /* PCM1の出力データを0に設定 */
        PCM.2   = 0;                            /* PCM2の出力データを0に設定 */
        PCM.3   = 0;                            /* PCM3の出力データを0に設定 */
        PMCM.1  = 0;                            /* PCM1を出力モードに設定 */
        PMCM.2  = 0;                            /* PCM2を出力モードに設定 */
        PMCM.3  = 0;                            /* PCM3を出力モードに設定 */
                                                /* PCM0-PCM3:未使用 */
                                                /* ※オンチップ・デバッグの通信用に */
                                                /*   CSIB3を使用する場合，PCM0および */
                                                /*   PMCM0は変更しないでください。 */

/*--------------------------------------------------------------------
        ポートCTの設定
--------------------------------------------------------------------*/
        PCT     = 0b00000000;                   /* PCT0,1,4,6の出力データを0に設定 */
        PMCT    = 0b10101100;                   /* PCT0,1,4,6を出力モードに設定 */
                                                /* PCT0,1,4,6:未使用 */

/*--------------------------------------------------------------------
        ポートDHの設定
--------------------------------------------------------------------*/
        PDH     = 0b00000000;                   /* PDH0-PDH4の出力データを0に設定 */
        PMDH    = 0b11100000;                   /* PDH0-PDH4を出力モードに設定 */
                                                /* PDH0-PDH4:未使用 */
```

```
/*--------------------------------------------------------------------
        ポートDLの設定
--------------------------------------------------------------------*/
        PDL      = 0b0000000000000000;              /* PDL0-PDL15の出力データを0に設定 */
        PMDL     = 0b0000000000000000;              /* PDL0-PDL15を出力モードに設定 */
                                                    /* PDL0-PDL15:未使用 */

}

/*********************************************************************
*       Title    : 低電圧検出処理
*********************************************************************
*       Module   : static void fn_InitLvi( void )
*       Arg              :
*       Ret              :
*--------------------------------------------------------------------
*       Note     : 低電圧検出動作を開始します。
*                : 電源電圧がCPUクロックの動作範囲になっているか確認します。
*********************************************************************/
static void fn_InitLvi( void )
{
        unsigned char ucCounter;

        LVIMK    = 1;                               /* INTLVI割り込み禁止 */
        LVIIF    = 0;                               /* INTLVI割り込み要求クリア */
        __asm("st.b       r0,      PRCMD    ");     /* 低電圧検出時の動作を割り込みに設定 */
        __asm("st.b       r0,      LVIM     ");
        LVIS     = 0b00000000;                      /* 低電圧検出レベルを 2.80V に設定 */

        /* 低電圧検出動作許可 */
        __asm("push       r10                                ");
        __asm("mov        0x80,    r10               ");
        __asm("st.b       r10,     PRCMD    ");
        __asm("st.b       r10,     LVIM     ");
        __asm("pop        r10                                ");

        /* 低電圧検出回路の動作安定待ち( 約0.2ms ) */
        for( ucCounter=0; ucCounter<15; ucCounter++ ){
                __NOP();
        }

        /* 電源電圧>低電圧検出レベル になるまでのウエイト */
        while( LVIF )
```

```
        {
                __NOP();
        }

}

/************************************************************************
*       Title    : クロックの初期設定処理
************************************************************************
*       Module   : static void fn_InitClock( void )
*       Arg              :
*       Ret              :
*-----------------------------------------------------------------------
*       Note     : クロックを設定します。
************************************************************************/
static void fn_InitClock( void )
{
        SELPLL   = 1;     /* PLLモードに設定 */

        /* CPUクロックを分周なし，サブクロックを通常発振に設定 */
        __asm("st.b       r0,       PRCMD    ");
        __asm("st.b       r0,       PCC              ");

        /* スタンバイ・モード解除時のCPUクロック発振安定時間を設定 */
        OSTS     = 0b00000011;
/*                  |||||+++---   OSTS2-OSTS0 [発振安定時間／セットアップ時間の選択]
                    |||||          000: 2^10/fX
                    |||||          001: 2^11/fX
                    |||||          010: 2^12/fX
                    |||||          011: 2^13/fX
                    |||||          100: 2^14/fX
                    |||||          101: 2^15/fX
                    |||||          110: 2^16/fX
                    |||||          111: 設定禁止
                    +++++-------   必ず0に設定
*/

}
```

```
/************************************************************************
*       Title    : リアルタイム・カウンタの初期設定処理
************************************************************************
*       Module   : static void fn_InitRtc( void )
*       Arg              :
*       Ret              :
*-----------------------------------------------------------------------
*       Note     : カウント開始時刻を 2010年4月1日(木) 13時00分00秒 に設定します。
*                        : 定周期割り込み(INTRTC0)を 0.5秒周期 に設定します。
*                        : アラーム割り込み(INTRTC1)を 毎日14時00分 に設定します。
*                        : インターバル割り込み(INTRTC2)を 約3.9ms周期 に設定します。
************************************************************************/
static void fn_InitRtc( void )
{
       RC1CC0.7 = 0;     /* リアルタイム・カウンタ動作停止 */

/*---------------------------------------------------------------------
       定周期割り込み動作の設定
---------------------------------------------------------------------*/
       RC1CC0   = 0b00000000;
/*                  ||||||||
                    ||++++++---   必ず0に設定
                    |+---------   RC1CKS [動作クロックの選択]
                    |                      0: fXTを動作クロックとして選択
                    |                      1: fBRGを動作クロックとして選択
                    +----------   RC1PWR [リアルタイム・カウンタの動作の制御]
                                           0: リアルタイム・カウンタ動作停止
                                           1: リアルタイム・カウンタ動作許可
*/

       RC1CC0.7 = 1;     /* リアルタイム・カウンタ動作許可 */

       RC1CC1   = 0b00001001;
/*                  |||||+++---   CT2-CT0 [定周期割り込み(INTRTC0)の選択]
                    |||||          000: 定周期割り込みを使用しない
                    |||||          001: 0.5秒に1度（秒カウント・アップに同期）
                    |||||          010: 1秒に1度（秒カウント・アップと同時）
                    |||||          011: 1分に1度（毎分00秒）
                    |||||          100: 1時間に1度（毎時00分00秒）
                    |||||          101: 1日に1度（毎日00時00分00秒）
                    |||||          11x: 1月に1度（毎月1日午前00時00分00秒）
                    ||||+------   AMPM [12時間制／24時間制の選択]
                    ||||           0: 12時間制表示（午前／午後を表示）
```

```
                    ||||           1: 24時間制表示
                    |||+-------   CLOE0 [RTCCL端子の出力制御]
                    |||            0: RTCCL端子の出力 (32.768 kHz) 禁止
                    |||            1: RTCCL端子の出力 (32.768 kHz) 許可
                    ||+--------   CLOE1 [RTC1HZ端子の出力制御]
                    ||             0: RTC1HZ端子の出力 (1 Hz) 禁止
                    ||             1: RTC1HZ端子の出力 (1 Hz) 許可
                    |+---------   必ず0に設定
                    +----------   RTCE [各カウンタの動作の制御]
                                   0: カウンタ動作停止
                                   1: カウンタ動作許可
*/

        /* 時計誤差補正機能の設定 */
        fn_RtcCorrect( 0b00000000 );

        /* カウント開始時刻の初期値設定 ( 2010年4月1日(木) 13時00分00秒 ) */
        RC1SEC              = 0x00;
        RC1MIN              = 0x00;
        RC1HOUR             = 0x13;
        RC1WEEK             = 0x04;
        RC1DAY              = 0x01;
        RC1MONTH            = 0x04;
        RC1YEAR             = 0x10;

/*-----------------------------------------------------------------------
        アラーム割り込み動作の設定
-----------------------------------------------------------------------*/
        RC1CC2    = 0b00000000;
/*                  |||||||+---   RWAIT [リアルタイム・カウンタのウエイト制御]
                    |||||||                0: カウンタ動作設定
                    |||||||                1: 秒～年カウンタのカウント動作停止
                    |||||||                   (カウンタ値の読み出し，書き込みモード)
                    ||||||+----   RWST [リアルタイム・カウンタのウエイト状態]
                    ||||||                 0: カウンタ動作中
                    ||||||                 1: 秒～年カウンタのカウント・アップ停止状態
                    ||||||                    (カウンタ値の読み出し，書き込み許可状態)
                    |++++++-----   必ず0を設定
                    +----------   WALE [アラーム割り込み (INTRTC1) の動作制御]
                                           0: アラームの一致による割り込みを発生しない
                                           1: アラームの一致による割り込みを発生する
*/
```

```
        /* アラーム時刻の初期値設定( 毎日14時00分 ) */
        RC1ALW   = 0x7F;            /* 曜日 */
        RC1ALH   = 0x14;            /* 時 */
        RC1ALM   = 0x00;            /* 分 */

        RC1CC2.7 = 1;               /* アラーム割り込み許可 */

/*-----------------------------------------------------------------
        インターバル動作の設定
-----------------------------------------------------------------*/
        RC1CC3   = 0b10000001;
/*                  |||||+++---   ICT2-ICT0 [インターバル割り込み (INTRTC2) の選択]
                    |||||          000: 2^6/fXT (1.953125 ms)
                    |||||          001: 2^7/fXT (3.90625 ms)
                    |||||          010: 2^8/fXT (7.8125 ms)
                    |||||          011: 2^9/fXT (15.625 ms)
                    |||||          100: 2^10/fXT (31.25 ms)
                    |||||          101: 2^11/fXT (62.5 ms)
                    |||||          11x: 2^12/fXT (125 ms)
                    |||++-------   必ず0を設定
                    ||+--------   CKDIV [RTCDIV端子の出力周波数の選択]
                    ||             0: RTCDIV端子から512 Hz (1.95 ms) を出力
                    ||             1: RTCDIV端子から16.384 kHz (0.061 ms) を出力
                    |+---------   CLOE2 [RTCDIV端子の出力制御]
                    |              0: RTCDIV端子の出力禁止
                    |              1: RTCDIV端子の出力許可
                   +----------   RINTE [インターバル割り込み (INTRTC2) の制御]
                                  0: インターバル割り込みを発生しない
                                  1: インターバル割り込みを発生する
*/

}

/*********************************************************************
*       Title    : RTCバックアップ・モードの初期設定処理
*********************************************************************
*       Module   : static void fn_InitRtcBum( void )
*       Arg              :
*       Ret              :
*--------------------------------------------------------------------
*       Note     : RTCバックアップ・モードの使用を許可します。
*********************************************************************/
static void fn_InitRtcBum( void )
```

```
{
        /* RTCバックアップ・モード使用許可 */
        /* （※リセット解除後，すでにRTCバックアップ準備状態となっている場合，*/
        /*    ここで解除します） */
        __asm("push       r10                                          ");
        __asm("mov        0x80,    r10                          ");
        __asm("st.b       r10,     PRCMD              ");
        __asm("st.b       r10,     RTCBUMCTL0         ");
        __asm("pop        r10                                          ");

        /* サブクロックを通常発振モードに設定 */
        __asm("st.b       r0,               PRCMD              ");
        __asm("st.b       r0,               SOSCAMCTL          ");

        __NOP();

}

/***********************************************************************
*       Title    : リアルタイム・カウンタのカウント書き換え処理
***********************************************************************
*       Module   : static void fn_WriteRtcCounter( unsigned char ucInc )
*       Arg              : False:インクリメント True:デクリメント
*       Ret              :
*----------------------------------------------------------------------
*       Note     : リアルタイム・カウンタの各カウント値を書き換えます。
***********************************************************************/
static void fn_WriteRtcCounter( unsigned char ucInc )
{
        /* 月別の日数 */
        const unsigned char aDays[12]
         = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
        unsigned char ucWork, ucRet;

        while( RC1CC2.1 ) /* 前回の書き換えが終了していることを確認 */
        {
                __NOP();
        }
        RC1CC2.0 = 1;              /* カウント・レジスタのカウント動作停止 */
        while( !RC1CC2.1 )         /* カウンタのウエイト状態の確認 */
        {
                __NOP();
        }
```

```
/* 設定項目の判定 */
switch( ucSetupTime )
{
        /* 年 */
        case SETUP_TMYEAR:
                RC1YEAR = fn_BcdCount( RC1YEAR, ucInc, 0, 99 );
                break;
        /* 月 */
        case SETUP_TMMONTH:
                RC1MONTH = fn_BcdCount( RC1MONTH, ucInc, 1, 12 );
                break;
        /* 日 */
        case SETUP_TMDAY:
                /* 月を整数に変換 */
                ucWork = ( (RC1MONTH >> 4)*10 + (RC1MONTH & 0x0f) );

                /* うるう年の2月の場合 */
                if( !( ( (RC1YEAR >> 4)*10 + (RC1YEAR & 0x0f) ) % 4 )
                 && (RC1MONTH == 0x02) )
                {
                        RC1DAY = fn_BcdCount( RC1DAY, ucInc, 1, aDays[ucWork-1]+1 );
                }
                /* その他の月の場合 */
                else
                {
                        RC1DAY = fn_BcdCount( RC1DAY, ucInc, 1, aDays[ucWork-1] );
                }
                break;
        /* 曜日 */
        case SETUP_TMWEEK:
                RC1WEEK = fn_BcdCount( RC1WEEK, ucInc, 0, 6 );
                break;
        /* 時 */
        case SETUP_TMHOUR:
                RC1HOUR = fn_BcdCount( RC1HOUR, ucInc, 0, 23 );
                break;
        /* 分 */
        case SETUP_TMMIN:
                RC1MIN = fn_BcdCount( RC1MIN, ucInc, 0, 59 );
                break;
        /* 秒 */
        case SETUP_TMSEC:
```

```
                        RC1SEC = fn_BcdCount( RC1SEC, ucInc, 0, 59 );
                        break;
                default:
                        break;
        }

        RC1CC2.0 = 0;       /* カウント・レジスタのカウント動作許可 */
}

/************************************************************************
*       Title    : リアルタイム・カウンタのアラーム書き換え処理
************************************************************************
*       Module   : static void fn_WriteRtcAlerm( unsigned char ucInc )
*       Arg              : False:インクリメント True:デクリメント
*       Ret              :
*-----------------------------------------------------------------------
*       Note     : リアルタイム・カウンタのアラーム設定を書き換えます。
************************************************************************/
static void fn_WriteRtcAlerm( unsigned char ucInc )
{
        unsigned char ucCnt;

        RTC1MK = 1;         /* INTRTC1割り込み禁止 */
        RC1CC2.7 = 0;       /* アラーム割り込み禁止 */

        /* 設定項目の判定 */
        switch( ucSetupAlarm )
        {
                /* 曜日 */
                case SETUP_ALWEEK:
                        /* インクリメントの場合 */
                        if( !ucInc )
                        {
                                /* "毎日"の場合 */
                                if( RC1ALW == 0b01111111 )
                                {
                                        RC1ALW = 0b00000001;        /* "日" */
                                }
                                /* "土"の場合 */
                                else if( RC1ALW == 0b01000000 )
                                {
                                        RC1ALW = 0b01111111;        /* "毎日" */
                                }
```

```
                                /* その他の場合 */
                                else
                                {
                                        RC1ALW <<= 1;                   /* 次の曜日 */
                                }
                        }
                        /* デクリメントの場合 */
                        else
                        {
                                /* "毎日"の場合 */
                                if( RC1ALW == 0b01111111 )
                                {
                                        RC1ALW = 0b01000000;    /* "土" */
                                }
                                /* "日"の場合 */
                                else if( RC1ALW == 0b00000001 )
                                {
                                        RC1ALW = 0b01111111;    /* "毎日" */
                                }
                                else
                                {
                                        RC1ALW >>= 1;                   /* 前の曜日 */
                                }
                        }
                        break;
                /* 時 */
                case SETUP_ALHOUR:
                        RC1ALH = fn_BcdCount( RC1ALH, ucInc, 0, 23 );
                        break;
                /* 分 */
                case SETUP_ALMIN:
                        RC1ALM = fn_BcdCount( RC1ALM, ucInc, 0, 59 );
                        break;
                default:
                        break;
        }

        RTC1IF = 0;     /* INTRTC1割り込み要求クリア */
        RTC1MK = 0;     /* INTRTC1割り込み許可 */

        RC1CC2.7 = 1;   /* アラーム割り込み許可 */
}
```

```
/************************************************************************
*       Title    : BCD値のカウント処理
************************************************************************
*       Module   : static unsigned char fn_BcdCount
*                        : ( unsigned char ucNum, unsigned char ucInc,
*                        :   unsigned char ucMin, unsigned char ucMax )
*       Arg              : ucNum :被加算値(BCD形式)
*                        : ucInc :Inc/Dec (True:インクリメント False:デクリメント)
*                        : ucMin :最小値
*                        : ucMax :最大値
*       Ret              : 演算結果(BCD形式)
*-----------------------------------------------------------------------
*       Note     : BCD値のインクリメント／デクリメントを行います。
************************************************************************/
static unsigned char fn_BcdCount
( unsigned char ucNum, unsigned char ucInc,
  unsigned char ucMin, unsigned char ucMax )
{
        unsigned char ucWork;

        /* 被加算値を整数に変換 */
        ucWork = ((ucNum & 0xf0)>>4)*10 + (ucNum & 0x0f);

        /* インクリメント */
        if( !ucInc )
        {
                ucWork++;

                /* 最大値を超えた場合 */
                if( ucWork > ucMax )
                {
                        ucWork = ucMin;   /* 最小値設定 */
                }
        }
        /* デクリメント */
        else
        {
                /* 最小値の場合 */
                if( ucWork == ucMin )
                {
                        ucWork = ucMax;   /* 最大値設定 */
                }
                /* その他の場合 */
```

```
                else
                {
                        ucWork--;
                }
        }

        /* 演算結果をBCD形式に変換 */
        ucWork = ((ucWork / 10)<< 4) + (ucWork % 10);

        return ucWork;
}

/*********************************************************************
*       Title    : スタンバイ・モード移行処理
*********************************************************************
*       Module   : static void fn_SetStandby( void )
*       Arg              :
*       Ret              :
*--------------------------------------------------------------------
*       Note     : スタンバイ・モードに移行します。
*********************************************************************/
static void fn_SetStandby( void )
{
        /* 使用するスタンバイ機能：通常STOPモード */
        PSMR = 0b0000001;

        /* STOPモードに移行 */
        __asm("push      r10                              ");
        __asm("mov       0x02,    r10             ");
        __asm("st.b      r10,     PRCMD    ");
        __asm("st.b      r10,     PSC             ");
        __asm("pop       r10                              ");

        __NOP();
        __NOP();
        __NOP();
        __NOP();
        __NOP();
}

/*********************************************************************
*       Title    : 時計誤差補正処理
*********************************************************************
```

```
*       Module   : static void fn_RtcCorrect( unsigned char ucReg )
*       Arg              : 設定値
*       Ret              :
*-------------------------------------------------------------------------
*       Note     : リアルタイム・カウンタの時計誤差補正機能を設定します。
************************************************************************/
static void fn_RtcCorrect( unsigned char ucReg )
{
        RC1SUBU = ucReg;

/*      RC1SUBU  = 0bxxxxxxxx;
                    ||++++++---   F5-F0
                    |+---------   F6 [時計誤差補正値の設定]
                    |                       0: F5-F0ビットで設定した値分，RC1SUBCの
                    |                          カウント値を増加（＋補正）
                    |                          増加値計算式
                    |                          ：(F5-F0ビットの設定値－1)×2
                    |                       1: F5-F0ビットで設定した値分，RC1SUBCの
                    |                          カウント値を減少（－補正）
                    |                          減少値計算式
                    |                          ：(F5-F0ビットの設定値の反転値データ＋1)×2
                    |                      ※F6-F0ビットの値が｛1/0，0，0，0，0，0，1/0｝
                    |                        のときは，時計誤差補正は行いません。
                    +----------   DEV [時計誤差補正のタイミングの設定]
                                          0: RC1SEC（秒カウンタ）が00，20，40秒のとき
                                             （20秒ごと）に時計誤差補正
                                          1: RC1SEC（秒カウンタ）が00秒のとき
                                             （60秒ごと）に時計誤差補正
*/

}

/************************************************************************
*       Title    : 定周期割り込み処理(INTRTC0割り込み使用)
************************************************************************
*       Module   : __interrupt void fn_IntRtc0( void )
*       Arg              :
*       Ret              :
*-------------------------------------------------------------------------
*       Note     : INTRTC0割り込みを使用した処理を実行します。
************************************************************************/
__interrupt void fn_IntRtc0( void )
{
```

```
/* ブザー出力中の場合 */
if( ucBuzzerTimer )
{
        ucBuzzerTimer--;  /* ブザー用タイマ カウント */

        /* ブザー出力期間が終了した場合 */
        if( !ucBuzzerTimer )
        {
                P7L.0 = 1;        /* ブザー出力停止 */
        }
}

/* 交互表示中の場合 */
if( ucBlinkDispTimer )
{
        ucBlinkDispTimer--;       /* 交互表示用タイマ カウント */

        /* 表示を切り替えるタイミングになった場合 */
        if( !ucBlinkDispTimer )
        {
                /* 時計表示・リセット復帰表示 */
                if( ucDisplay == DISP_WATCH )
                {
                        ucDisplay = DISP_RESET;
                }
                /* リセット復帰表示・時計表示 */
                else
                {
                        ucDisplay = DISP_WATCH;
                }

                /* 交互表示用タイマ カウント再開 */
                ucBlinkDispTimer = BLINKDISPTIME;
        }
}

ucColonBlink ^= 1;        /* "："表示 点灯/消灯切り替え */
ucDispReq = 1;            /* 表示更新要求発行 */
```

```
        /*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*/
        /*                                                                */
        /*                                                                */
        /*        定周期割り込みを使用した処理を追加する場合，                */
        /*        ここに記述してください。                                  */
        /*                                                                */
        /*                                                                */
        /*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*/

}

/***********************************************************************
*       Title     : アラーム割り込み処理(INTRTC1割り込み使用)
************************************************************************
*       Module    : __interrupt void fn_IntRtc1( void )
*       Arg                :
*       Ret                :
*-----------------------------------------------------------------------
*       Note      : INTRTC1割り込みを使用した処理を実行します。
***********************************************************************/
__interrupt void fn_IntRtc1( void )
{
        /* 通常動作状態の場合 */
        if( ucMode == MODE_WATCH )
        {
                P7L.0 = 0;                          /* ブザー出力開始 */
                ucBuzzerTimer = BUZZER_TIME;        /* ブザー用タイマ カウント開始 */
        }

        /*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*/
        /*                                                                */
        /*                                                                */
        /*        アラーム割り込みを使用した処理を追加する場合，              */
        /*        ここに記述してください。                                  */
        /*                                                                */
        /*                                                                */
        /*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*/

}
```

```
/************************************************************************
*       Title    : 低電圧検出割り込み処理(INTLVI割り込み使用)
************************************************************************
*       Module   : __interrupt void fn_IntLvi( void )
*       Arg              :
*       Ret              :
*-----------------------------------------------------------------------
*       Note     : RTCバックアップ準備状態の設定／解除を行います。
************************************************************************/
__interrupt void fn_IntLvi( void )
{
        /*--------------------------------------------------------------*/
        /*        通常動作電圧・低電圧検出レベル に変化したとき                */
        /*--------------------------------------------------------------*/
        if( LVIF )
        {
                /* 動作クロックを低電圧検出状態に対応した周波数に設定する */
                SELPLL   = 0;      /* クロック・スルー・モードに設定 */
                PLLON    = 0;      /* PLL動作停止 */

                /* DMA動作の禁止 */
                E00              = 0;
                E11              = 0;
                E22              = 0;
                E33              = 0;

                /* NMI端子のエッジ検出を禁止 */
                INTF0    &= 0b01111000;
                INTR0    &= 0b01111000;
/*                              +|||||++--   必ず0に設定
                                 ||||+----   NMI
                                 |||+-----   INTP0
                                 ||+------   INTP1
                                 |+-------   INTP2
                                 +--------   INTP3
*/

                /* INTLVI以外の割り込みをすべて禁止 */
                IMR3     = 0b1111111111111111;
                IMR2     = 0b1111111111111111;
                IMR1     = 0b1111111111111111;
                IMR0     = 0b1111111111111110;
```

```
        /* サブクロックを超低消費発振モードに設定 */
        __asm("push     r10                                     ");
        __asm("mov      0x01,   r10                     ");
        __asm("st.b     r10,    PRCMD           ");
        __asm("st.b     r10,    SOSCAMCTL       ");
        __asm("pop      r10                                     ");

        /* RTCバックアップ準備状態に設定 */
        __asm("mov      0x81,   r10                     ");
        __asm("st.b     r10,    PRCMD           ");
        __asm("st.b     r10,    RTCBUMCTL0      ");

        /* スタンバイ・モードに移行 */
        fn_SetStandby();
}
/*-----------------------------------------------------------------*/
/*      低電圧検出レベル・通常動作電圧 に変化したとき               */
/*-----------------------------------------------------------------*/
else
{
        /* サブクロックを通常発振モードに設定 */
        __asm("st.b     r0,     PRCMD           ");
        __asm("st.b     r0,     SOSCAMCTL       ");

        /* RTCバックアップ準備状態を解除 */
        __asm("push     r10                                     ");
        __asm("mov      0x80,   r10                     ");
        __asm("st.b     r10,    PRCMD           ");
        __asm("st.b     r10,    RTCBUMCTL0      ");
        __asm("pop      r10                                     ");

        __NOP();

        /* INTLVI以外で使用する割り込みの許可 */
        KRIF    = 0;    /* INTKR割り込み要求クリア */
        RTC0IF  = 0;    /* INTRTC0割り込み要求クリア */
        RTC1IF  = 0;    /* INTRTC1割り込み要求クリア */
        KRMK    = 0;    /* INTKR割り込み許可 */
        RTC0MK  = 0;    /* INTRTC0割り込み許可 */
        RTC1MK  = 0;    /* INTRTC1割り込み許可 */

        /* 動作クロックを通常動作用に設定する */
        PLLON   = 1;    /* PLL動作許可 */
```

```
                while( LOCK )
                {
                        __NOP(); /* PLL動作安定待ち */
                }
                SELPLL   = 1;     /* PLLモードに設定 */

                ucDispReq = 1;    /* 表示更新要求発行 */
        }
}

/************************************************************************
*       Title    : キー割り込み処理(INTKR割り込み使用)
************************************************************************
*       Module   : __interrupt void fn_IntKr( void )
*       Arg              :
*       Ret              :
*-----------------------------------------------------------------------
*       Note     : キーの検出通知を発行します。
************************************************************************/
__interrupt void fn_IntKr( void )
{
        ucKeyEdgeInfo = 1;         /* キーの検出通知を発行 */
}

/************************************************************************
*       Title    : キーのサンプリング処理
************************************************************************
*       Module   : static void fn_KeySampling( void )
*       Arg              :
*       Ret              :
*-----------------------------------------------------------------------
*       Note     : キー入力用ポートのサンプリングを行います。
************************************************************************/
static void fn_KeySampling( void )
{
        unsigned char ucWork;

        /* キー入力用ポートの状態を取得し，コード化する */
        switch( P5 & 0x0f )
        {
                /* アクティブ・レベル：なし */
                case 0b00001111:
                        /* コード：オフ */
```

```
                ucWork = KEYCODEOFF;
                break;
        /* アクティブ・レベル：ビット0 */
        case 0b00001110:
                /* コード：キー1 */
                ucWork = KEYCODE1;
                break;
        /* アクティブ・レベル：ビット1 */
        case 0b00001101:
                /* コード：キー2 */
                ucWork = KEYCODE2;
                break;
        /* アクティブ・レベル：ビット2 */
        case 0b00001011:
                /* コード：キー3 */
                ucWork = KEYCODE3;
                break;
        /* アクティブ・レベル：ビット3 */
        case 0b00000111:
                /* コード：キー4 */
                ucWork = KEYCODE4;
                break;
        /* アクティブ・レベル：複数 */
        default:
                /* コード：多重押し */
                ucWork = KEYCODEMLT;
                break;
}

/* 前回のコードと一致しない場合 */
if( ucCodeBuffer != ucWork )
{
        /* サンプリング回数リセット */
        ucKeySamplingCounter = KEYSAMPLINGCOUNT;
        /* 今回のコードを保存 */
        ucCodeBuffer = ucWork;
}
/* 前回のコードと一致した場合 */
else
{
        /* サンプリング回数カウント */
        ucKeySamplingCounter--;
```

```
        /* サンプリング回数が指定回数に到達 */
        if( !ucKeySamplingCounter )
        {
                /* サンプリング回数リセット */
                ucKeySamplingCounter = KEYSAMPLINGCOUNT;

                /* キー・コードを更新する場合 */
                if( ucCodeBuffer != ucKeyCode )
                {
                        /* 多重押しからの更新の場合 */
                        if( ucKeyCode == KEYCODEMLT )
                        {
                                /* 多重押し･オフ の場合 */
                                if( ucCodeBuffer == KEYCODEOFF )
                                {
                                        /* キー・コード更新：キー・オフ */
                                        ucKeyCode = KEYCODEOFF;
                                }
                        }
                        /* その他の場合 */
                        else
                        {
                                /* オフ または 多重押し ではない場合 */
                                if( (ucCodeBuffer != KEYCODEOFF)
                                 && (ucCodeBuffer != KEYCODEMLT) )
                                {
                                        /* キーの有効入力通知 発行 */
                                        ucKeyValidInfo = 1;
                                }

                                /* キー・コード更新 */
                                ucKeyCode = ucCodeBuffer;
                        }

                        /* キー・オフの場合 */
                        if( ucKeyCode == KEYCODEOFF )
                        {
                                /* キーの検出通知をクリア */
                                ucKeyEdgeInfo = 0;
                        }
                }
        }
}
```

```
}


/*********************************************************************
*       Title    : 表示更新処理
*********************************************************************
*       Module   : static void fn_Display( void )
*       Arg              :
*       Ret              :
*--------------------------------------------------------------------
*       Note     : 表示データに合わせて表示用デバイスを制御します。
*********************************************************************/
static void fn_Display( void )
{
        unsigned char ucPos, ucBar;

        ucPos = 0;        /* 表示位置 初期値設定 */

        switch( ucDisplay )
        {
                /*=========================================================*/
                /*       時計表示                                          */
                /*=========================================================*/
                /*       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+            */
                /*       |1|0|.|0|4|.|0|1|T|H|U| |1|3|:|0|0|:|0|0|            */
                /*       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+            */
                /*       | | |A|L|A|R|M| |A|L|L| |1|4|:|0|0| | | |            */
                /*       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+            */
                /*=========================================================*/
                case DISP_WATCH:
                        /*---------------------------------------------------------*/
                        /*       キャラクタの表示設定                              */
                        /*---------------------------------------------------------*/
                        while( RC1CC2.1 ) /* 前回の書き換えが終了していることを確認 */
                        {
                                __NOP();
                        }
                        RC1CC2.0 = 1;             /* カウント・レジスタのカウント動作停止 */
                        while( !RC1CC2.1 )        /* カウンタのウエイト状態の確認 */
                        {
                                __NOP();
                        }

                        /* "年" */
```

```
fn_LcdChrDisp( ((RC1YEAR >> 4) & 0x0f), ucPos++, 0 );/* 上位桁 */
fn_LcdChrDisp( (RC1YEAR & 0x0f), ucPos++, 0 );       /* 下位桁 */

/* '.' */
fn_LcdChrDisp( '.', ucPos++, 1 );

/* "月" */
fn_LcdChrDisp( ((RC1MONTH >> 4) & 0x0f), ucPos++, 0 );/* 上位桁 */
fn_LcdChrDisp( (RC1MONTH & 0x0f), ucPos++, 0 );      /* 下位桁 */

/* '.' */
fn_LcdChrDisp( '.', ucPos++, 1 );

/* "日" */
fn_LcdChrDisp( ((RC1DAY >> 4) & 0x0f), ucPos++, 0 ); /* 上位桁 */
fn_LcdChrDisp( (RC1DAY & 0x0f), ucPos++, 0 );        /* 下位桁 */

/* "曜日" */
switch( RC1WEEK )
{
        /* 日 */
        case 0x00:
                fn_LcdChrDisp( 'S', ucPos++, 1 );
                fn_LcdChrDisp( 'U', ucPos++, 1 );
                fn_LcdChrDisp( 'N', ucPos++, 1 );
                break;
        /* 月 */
        case 0x01:
                fn_LcdChrDisp( 'M', ucPos++, 1 );
                fn_LcdChrDisp( 'O', ucPos++, 1 );
                fn_LcdChrDisp( 'N', ucPos++, 1 );
                break;
        /* 火 */
        case 0x02:
                fn_LcdChrDisp( 'T', ucPos++, 1 );
                fn_LcdChrDisp( 'U', ucPos++, 1 );
                fn_LcdChrDisp( 'E', ucPos++, 1 );
                break;
        /* 水 */
        case 0x03:
                fn_LcdChrDisp( 'W', ucPos++, 1 );
                fn_LcdChrDisp( 'E', ucPos++, 1 );
                fn_LcdChrDisp( 'D', ucPos++, 1 );
```

```
                break;
        /* 木 */
        case 0x04:
                fn_LcdChrDisp( 'T', ucPos++, 1 );
                fn_LcdChrDisp( 'H', ucPos++, 1 );
                fn_LcdChrDisp( 'U', ucPos++, 1 );
                break;
        /* 金 */
        case 0x05:
                fn_LcdChrDisp( 'F', ucPos++, 1 );
                fn_LcdChrDisp( 'R', ucPos++, 1 );
                fn_LcdChrDisp( 'I', ucPos++, 1 );
                break;
        /* 土 */
        case 0x06:
                fn_LcdChrDisp( 'S', ucPos++, 1 );
                fn_LcdChrDisp( 'A', ucPos++, 1 );
                fn_LcdChrDisp( 'T', ucPos++, 1 );
                break;
        default:
                break;
}

/* 空白 */
fn_LcdChrDisp( ' ', ucPos++, 1 );

/* "時" */
fn_LcdChrDisp( ((RC1HOUR >> 4) & 0x0f), ucPos++, 0 );/* 上位桁 */
fn_LcdChrDisp( (RC1HOUR & 0x0f), ucPos++, 0 );       /* 下位桁 */

/* ':' 点灯 */
if( ucColonBlink )
{
        fn_LcdChrDisp( ':', ucPos++, 1 );
}
/* ':' 消灯 */
else
{
        fn_LcdChrDisp( ' ', ucPos++, 1 );
}

/* "分" */
fn_LcdChrDisp( ((RC1MIN >> 4) & 0x0f), ucPos++, 0 ); /* 上位桁 */
```

```
fn_LcdChrDisp( (RC1MIN & 0x0f), ucPos++, 0 );         /* 下位桁 */

/* ':' 点灯 */
if( ucColonBlink )
{
        fn_LcdChrDisp( ':', ucPos++, 1 );
}
/* ':' 消灯 */
else
{
        fn_LcdChrDisp( ' ', ucPos++, 1 );
}

/* "秒" */
fn_LcdChrDisp( ((RC1SEC >> 4) & 0x0f), ucPos++, 0 ); /* 上位桁 */
fn_LcdChrDisp( (RC1SEC & 0x0f), ucPos++, 0 );         /* 下位桁 */

RC1CC2.0 = 0;     /* カウント・レジスタのカウント動作許可 */

/* アラーム 固定表示 */
fn_LcdChrDisp( ' ', ucPos++, 1 );
fn_LcdChrDisp( ' ', ucPos++, 1 );
fn_LcdChrDisp( 'A', ucPos++, 1 );
fn_LcdChrDisp( 'L', ucPos++, 1 );
fn_LcdChrDisp( 'A', ucPos++, 1 );
fn_LcdChrDisp( 'R', ucPos++, 1 );
fn_LcdChrDisp( 'M', ucPos++, 1 );
fn_LcdChrDisp( ' ', ucPos++, 1 );

/* アラーム曜日 */
switch( RC1ALW )
{
        /* 毎日 */
        case 0b01111111:
                fn_LcdChrDisp( 'A', ucPos++, 1 );
                fn_LcdChrDisp( 'L', ucPos++, 1 );
                fn_LcdChrDisp( 'L', ucPos++, 1 );
                break;
        /* 日 */
        case 0b00000001:
                fn_LcdChrDisp( 'S', ucPos++, 1 );
                fn_LcdChrDisp( 'U', ucPos++, 1 );
                fn_LcdChrDisp( 'N', ucPos++, 1 );
```

```
                break;
            /* 月 */
            case 0b00000010:
                fn_LcdChrDisp( 'M', ucPos++, 1 );
                fn_LcdChrDisp( 'O', ucPos++, 1 );
                fn_LcdChrDisp( 'N', ucPos++, 1 );
                break;
            /* 火 */
            case 0b00000100:
                fn_LcdChrDisp( 'T', ucPos++, 1 );
                fn_LcdChrDisp( 'U', ucPos++, 1 );
                fn_LcdChrDisp( 'E', ucPos++, 1 );
                break;
            /* 水 */
            case 0b00001000:
                fn_LcdChrDisp( 'W', ucPos++, 1 );
                fn_LcdChrDisp( 'E', ucPos++, 1 );
                fn_LcdChrDisp( 'D', ucPos++, 1 );
                break;
            /* 木 */
            case 0b00010000:
                fn_LcdChrDisp( 'T', ucPos++, 1 );
                fn_LcdChrDisp( 'H', ucPos++, 1 );
                fn_LcdChrDisp( 'U', ucPos++, 1 );
                break;
            /* 金 */
            case 0b00100000:
                fn_LcdChrDisp( 'F', ucPos++, 1 );
                fn_LcdChrDisp( 'R', ucPos++, 1 );
                fn_LcdChrDisp( 'I', ucPos++, 1 );
                break;
            /* 土 */
            case 0b01000000:
                fn_LcdChrDisp( 'S', ucPos++, 1 );
                fn_LcdChrDisp( 'A', ucPos++, 1 );
                fn_LcdChrDisp( 'T', ucPos++, 1 );
                break;
            default:
                break;
    }

    /* 空白 */
    fn_LcdChrDisp( ' ', ucPos++, 1 );
```

```
/* アラーム時 */
fn_LcdChrDisp( ((RC1ALH >> 4) & 0x0f), ucPos++, 0 ); /* 上位桁 */
fn_LcdChrDisp( (RC1ALH & 0x0f), ucPos++, 0 );        /* 下位桁 */

/* ':' */
fn_LcdChrDisp( ':', ucPos++, 1 );

/* アラーム分 */
fn_LcdChrDisp( ((RC1ALM >> 4) & 0x0f), ucPos++, 0 ); /* 上位桁 */
fn_LcdChrDisp( (RC1ALM & 0x0f), ucPos++, 0 );        /* 下位桁 */

/* 空白 */
fn_LcdChrDisp( ' ', ucPos++, 1 );
fn_LcdChrDisp( ' ', ucPos++, 1 );
fn_LcdChrDisp( ' ', ucPos++, 1 );

/*-----------------------------------------------------*/
/*        カーソルの表示設定
        */
/*-----------------------------------------------------*/
/* 設定項目の判定 */
switch( ucSetupTime )
{
        /* "年" 設定中 */
        case SETUP_TMYEAR:
                /* カーソル位置："年" 下位桁 */
                fn_LcdCurDisp( 1 );
                break;
        /* "月" 設定中 */
        case SETUP_TMMONTH:
                /* カーソル位置："月" 下位桁 */
                fn_LcdCurDisp( 4 );
                break;
        /* "日" 設定中 */
        case SETUP_TMDAY:
                /* カーソル位置："日" 下位桁 */
                fn_LcdCurDisp( 7 );
                break;
        /* "曜日" 設定中 */
        case SETUP_TMWEEK:
                /* カーソル位置："曜日" 1文字目 */
                fn_LcdCurDisp( 8 );
```

```
                        break;
                /* "時" 設定中 */
                case SETUP_TMHOUR:
                        /* カーソル位置："時" 下位桁 */
                        fn_LcdCurDisp( 13 );
                        break;
                /* "分" 設定中 */
                case SETUP_TMMIN:
                        /* カーソル位置："分" 下位桁 */
                        fn_LcdCurDisp( 16 );
                        break;
                /* "秒" 設定中 */
                case SETUP_TMSEC:
                        /* カーソル位置："秒" 下位桁 */
                        fn_LcdCurDisp( 19 );
                        break;
                default:
                        break;
        }
        switch( ucSetupAlarm )
        {
                /* "アラーム曜日" 設定中 */
                case SETUP_ALWEEK:
                        /* カーソル位置："アラーム曜日" 1文字目 */
                        fn_LcdCurDisp( 28 );
                        break;
                /* "アラーム時" 設定中 */
                case SETUP_ALHOUR:
                        /* カーソル位置："アラーム時" 下位桁 */
                        fn_LcdCurDisp( 33 );
                        break;
                /* "アラーム分" 設定中 */
                case SETUP_ALMIN:
                        /* カーソル位置："アラーム分" 下位桁 */
                        fn_LcdCurDisp( 36 );
                        break;
                default:
                        break;
        }

        break;
/*=========================================================*/
/*      リセット復帰表示
```

```
                  */
/*===========================================================*/
/*        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+          */
/*        | |C|P|U| |R|E|S|E|T| | | | | | | | | | |          */
/*        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+          */
/*        | | | | | | | | | | | | | | | | | | | | |          */
/*        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+          */
/*===========================================================*/
case DISP_RESET:
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( 'C', ucPos++, 1 );
         fn_LcdChrDisp( 'P', ucPos++, 1 );
         fn_LcdChrDisp( 'U', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( 'R', ucPos++, 1 );
         fn_LcdChrDisp( 'E', ucPos++, 1 );
         fn_LcdChrDisp( 'S', ucPos++, 1 );
         fn_LcdChrDisp( 'E', ucPos++, 1 );
         fn_LcdChrDisp( 'T', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
         fn_LcdChrDisp( ' ', ucPos++, 1 );
```

```
                        fn_LcdChrDisp( ' ', ucPos++, 1 );
                        fn_LcdChrDisp( ' ', ucPos++, 1 );
                        fn_LcdChrDisp( ' ', ucPos++, 1 );
                        fn_LcdChrDisp( ' ', ucPos++, 1 );
                        fn_LcdChrDisp( ' ', ucPos++, 1 );
                        fn_LcdChrDisp( ' ', ucPos++, 1 );
                        break;
                default:
                        break;
        }
}
```

付録B　改版履歴

版　数	発行年月	改版箇所	改版内容
第1版	2010.09.30	－	－

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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

Colophon 1.0