お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

アプリケーション・ノート

RX850（Ver.3.1）

リアルタイム・オペレーティング・システム

μSAA13RX703000
μSAB13RX703000
μS3K15RX703000
μS3P16RX703000

対象デバイス
　V850 ファミリ™

資料番号　U13646JJ1V1ANJ1（第１版）
発行年月　January 2001 N　CP(K)

［メ　モ］

V800 シリーズ，V850 ファミリ，V853 は日本電気株式会社の商標です。

UNIX は X/Open カンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標です。

Windows は米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。

本製品が外国為替および外国貿易管理法の規定による規制貨物等（または役務）に該当するか否かは，ユーザ（仕様を決定した者）が判定してください。

M7A　98.8

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

目　次

第1章　はじめに

1.1　対象者

このアプリケーション・ノートは、RX850（Ver.3.1）を使用したアプリケーション・プログラムを作成するユーザを対象とします。

1.2　目 的

このアプリケーション・ノートは「時計盤システム」を作成する手順を例として、RX850 を用いたアプリケーション・プログラムの作成方法を理解していただくことを目的としています。

1.3　構 成

このアプリケーション・ノートは大きく分けて次の内容で構成されています。

- システム概説
- ハードウエア構成
- ソフトウエア開発
- コーディング・リスト
- 回路図

1.4　読み方

このアプリケーション・ノートの読者は、電気、論理回路、及びマイクロコンピュータに関する一般知識が必要です。

1.5　凡 例

データ表記の重み	: 左が上位桁、右が下位桁
アクティブ・ロウの表記	: $\overline{\times\times\times}$（端子、信号名称に上線）
メモリ・マップのアドレス	: 上部―上位、下部―下位
注	: 本文中につけた「注」の説明
注意	: 気をつけて読んでいただきたい内容
備考	: 本文の補足説明
数の表記	: 2 進数 … ×××× または ××××B
	10 進数 … ××××
	16 進数 … ××××H または 0x××××

1.6　関連資料

- RX850（Ver.3.1）に関する資料

資料名	資料番号
ユーザーズ・マニュアル　基礎編	U13430J
ユーザーズ・マニュアル　インストレーション編	U13410J
ユーザーズ・マニュアル　テクニカル編	U13431J
ユーザーズ・マニュアル　RD850 タスク・ディバッガ（Windows ベース）	U11158J
アプリケーション・ノート　時計盤システム編	このマニュアル

- V853 に関する資料

資料名	資料番号
ユーザーズ・マニュアル　ハードウェア編	U10913J
データシート	U12036J
V850 ファミリ　アーキテクチャ編	U10243J

- 開発ツールに関する資料

資料名	資料番号
ユーザーズ・マニュアル　CA830,CA850　C 言語編	U12840J
ユーザーズ・マニュアル　CA850　アセンブリ言語編	U10543J
ユーザーズ・マニュアル　CA830,CA850 操作編 Windows ベース	U12827J
ユーザーズ・マニュアル　CA830,CA850 操作編 UNIX ベース	U12839J
ユーザーズ・マニュアル　CA732/CA830/CA850 プロジェクト・マネージャ編 Windows ベース	U11991J
ユーザーズ・マニュアル　ID850/SM850 操作編　Windows ベース	U11196J
ユーザーズ・マニュアル　AZ850　Windows ベース	U11181J

第2章　　概　説

V850 ファミリ™にはリアルタイム OS　RX850（Ver.3.1）が用意されています。ここでは、この RX850（Ver.3.1）を使用したアプリケーション・プログラムの作成例を示します。

このアプリケーション・ノートでは、例題として「時計盤システム」を取り上げます。いくつかのパーツによって構成されるこのシステムにおいて、リアルタイム OS の具体的な使用法を説明します。

第3章　　システム概説

3.1　　概 要

今回扱う時計盤システムは、デジタル時計機能をはじめ、タイマー機能、ストップウォッチ機能、またセンサによる現在の気温、湿度の測定機能などを有するシステムです。この時計盤システムは次にあげるパーツによって構成されています。

- 7 セグメント LED　　… 4 個
- LED　　… 2 個
- LCD ユニット　　… 1 台
- ステッピング・モータ　　… 1 台
- 圧電ブザーユニット　　… 1 台
- 温度センサ　　… 1 個
- 湿度センサ　　… 1 個
- タッチセンサユニット　　… 1 台
- タクト・スイッチ　　… 8 個

3.2　システムの構成

システムの構成を、図 3.2-1 に示します。

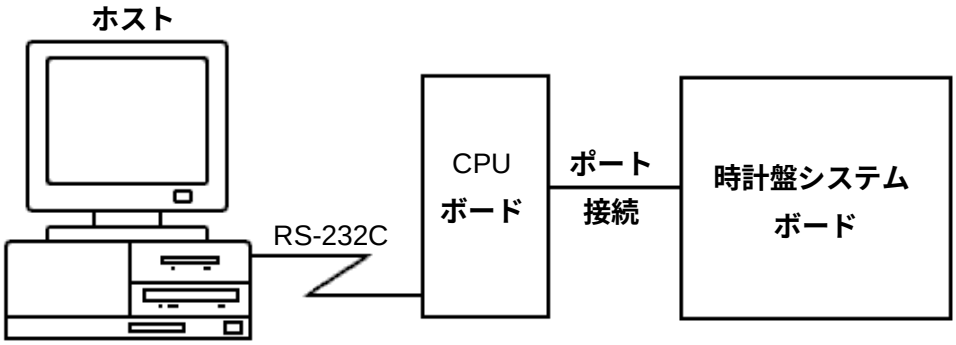

図 3.2-1　システムの構成

システムは CPU ボードから出ているポート端子と、時計盤システムボードとを接続して、システムを駆動します。また CPU ボードとホストマシン（PC 等）を RS-232C で接続し、キャラクタの送受信を行います。システムのハードウエア関連の詳細については「第 4 章 ハードウエア」をご覧ください。

また今回使用する CPU は V853（μPD703003）です。他の V850 ファミリ CPU とは周辺機能や I/O アドレスに違いがあります。I/O アドレスの違いについては、プログラム内での変更によって吸収できます。詳しくはそれぞれのハードウエア編マニュアルを御参照下さい。

3.3　機能概要

このシステムのそれぞれの機能について説明します。

1.　7 セグメント LED

4 桁の 7 セグメント LED を使用して実現するものは次の 4 つです。

- 時刻表示モード

  現在時刻を表示します。単位は 24 時間単位で、時刻の調整も可能にします。他のモードに移っても時刻は刻みつづけ、再び時刻表示モードになったとき、現在時刻を表示します。

- タイマーモード

  タイマーの値をセットし、スタートするとカウントダウンを始めます。設定値は 00:01～99:59 です。タイマー値が 0 になった時、圧電ブザーを 4 回鳴らします。タイマー動作中は、他のモードに移ってもカウントダウンを続け、再びタイマー表示モードになったとき、現在値を表示します。

- ストップウォッチモード

  スタート、ストップ、ラップ表示、リセット機能を持ちます。ストップウォッチの測定単位として、1 秒単位、1/10 秒単位の 2 つのモードを選択できます。ストップウォッチ動作中は、他のモードに移ってもカウントを続け、再びストップウォッチ表示モードになったとき、現在値を表示します。

- スロットマシンモード

  3 桁のスロットを回転し、各桁をスイッチで停止することができます。3 桁すべて数字が揃ったとき、圧電ブザーを 4 回鳴らします。スロット回転中は、他のモードに移っても回転を続け、再びスロットマシンモードになったとき、現在値を表示します（回転を続けたままの状態）。

これらの 4 つのモードは 1 つのタクト・スイッチを押すたびに切り替わります。
『時刻表示モード → タイマーモード → ストップウォッチモード → スロットモード』
という順番に変化し、スロットモードの次は、再び時刻表示モードに戻ります。

2. LED

4 桁の 7 セグメント LED の下二桁目と下三桁目の間に配置し、秒刻みを行います。時刻表示モード、タイマー動作中、ストップウォッチ動作中は、0.5 秒ごとに点灯・消灯を繰り返します。

3. LCD ユニット

7 セグメント LED の動作モードの表示、現在の気温・湿度の表示、またメモ機能として RS-232C を経由して送られてくるキャラクタの表示を行います。

4. ステッピング・モータ

常に回転させます。ただし気温の変化によって回転速度が変わり、高くなると速く回転します。

5. タッチセンサ

メモ表示モードの時、LCD に表示されているそのメモの消去、タイマーを途中で停止した際のリセットに使用します。

6. タクト・スイッチ

7 セグメント LED のモード切り替え、各桁の値の修正、LCD に表示する内容の切り替えなどの割り込み入力に使用します。8 つのタクト・スイッチを機能ごとに割り振ります。

7. 温度センサ

現在の気温を測定します。測定結果は LCD に表示します。

8. 湿度センサ

現在の湿度を測定します。測定結果は LCD に表示します。

第4章　　ハードウエア

4.1　　ハードウエア構成

時計盤システムのハードウエア構成は、図 4.1-1 のようになります。

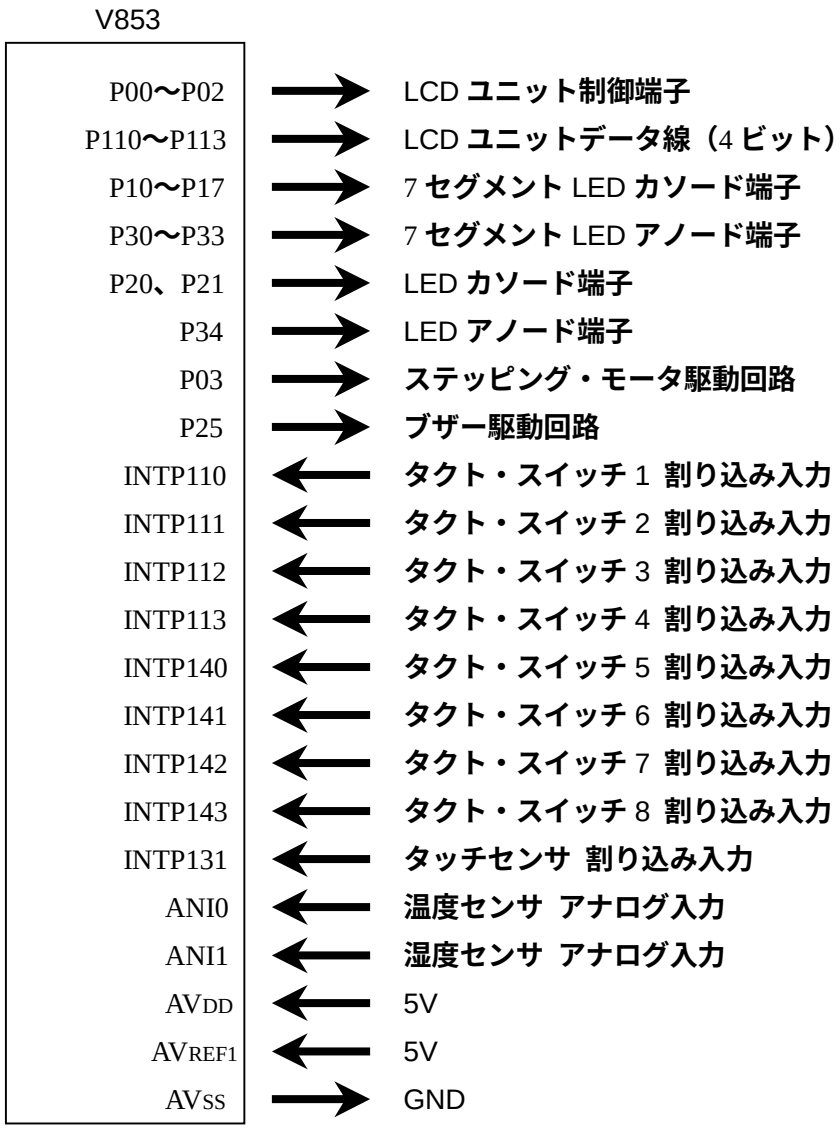

図 4.1-1　時計盤システムのハードウエア構成

4.2　CPU ボード

システムでは CPU ボードとして、

- RTE-V853-PC（マイダス・ラボ社製）+ ICE（IE-703003-MC（日本電気製）など）

または

- RT-V853（日本電気製）

を使用します。

4.3　メモリ・マップ

V853 にはシングルチップ・モード、シングルチップ・モード（外部拡張モード）、ROM レス・モードの 3 つのモードが存在しますが、今回はシングルチップ・モード（外部拡張モード）を使用します。その時の V853 のメモリ・マップを、図 4.3-1 に示します。

図 4.3-1　V853 メモリ・マップ

4.4　時計盤システムボードの説明

4.4.1　7 セグメント LED

今回使用する 4 桁の 7 セグメント LED は、ダイナミック・ドライブ方式で点灯します。ダイナミック・ドライブ方式とは、点灯させる桁を時分割で切り替えて表示する方法です。この切り替えを高速に行うことによって、人間の目には 4 桁すべての LED が点灯しているように見えます。今回はこの切り替えを 16ms で行います。

ダイナミック・ドライブ方式の回路は、図 4.4-1 のようになります。P10～P16（P はポートの意）より点灯させたいセグメントデータを出力し、P30～P33 より点灯させる桁を選択して出力します。スイッチング回路など、詳しい回路図については「第 7 章　回路図」をご参照ください。

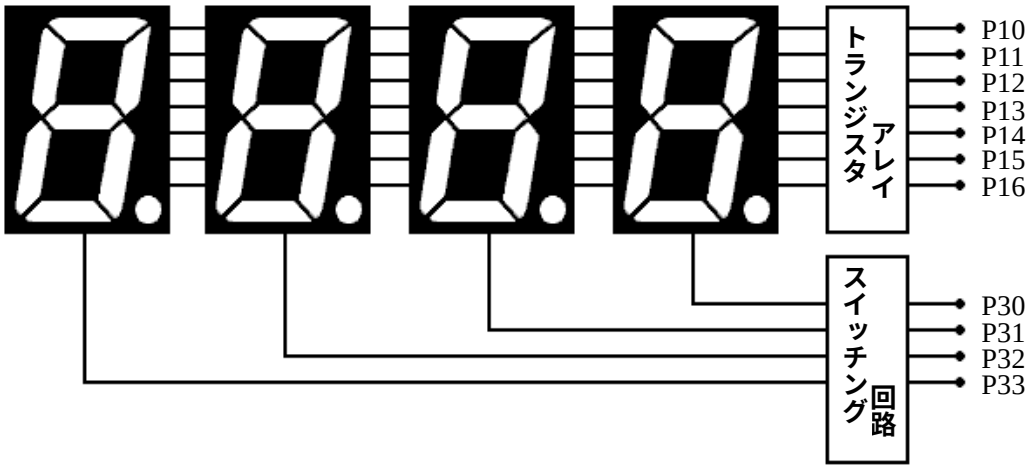

図 4.4-1　7 セグメント LED ダイナミック・ドライブ回路とポートの接続

4.4.2　　LED

7 セグメント LED の下二桁目と下三桁目の間に LED を 2 つ置き、秒を刻む動作を実現します。

回路は図 4.4-3 のようになります。7 セグメント LED の時のようにスイッチング回路を作り、点灯させる LED を選択してポートより出力します。スイッチング回路など、詳しい回路図については「第 7 章　　回路図」をご参照ください。

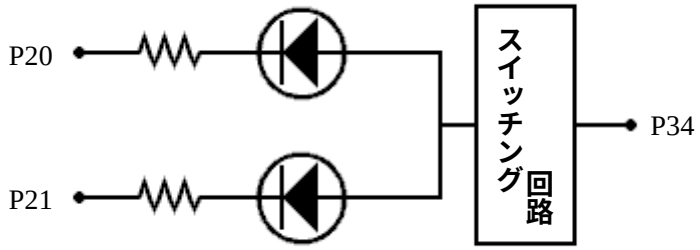

図 4.4-3　LED とポートの接続

4.4.3　　LCD ユニット

メッセージを表示するために、LCD ユニットを使用します。今回使用する LCD ユニットは、ドライバ IC がすでに組み込まれているもので、JIS コードを入力すると、それに対応するキャラクタが表示されます。この LCD ユニットへのデータ転送は、8 ビットのキャラクタコードを 4 ビットずつ 2 回に分けて行います。

図 4.4-4 が接続図となり、D4～D7 のデータ線に 2 回に分けて転送することになります。R/W、RS、E は LCD ユニットを制御するためのデータ線です。詳しい回路図については「第 7 章　　回路図」をご参照ください。

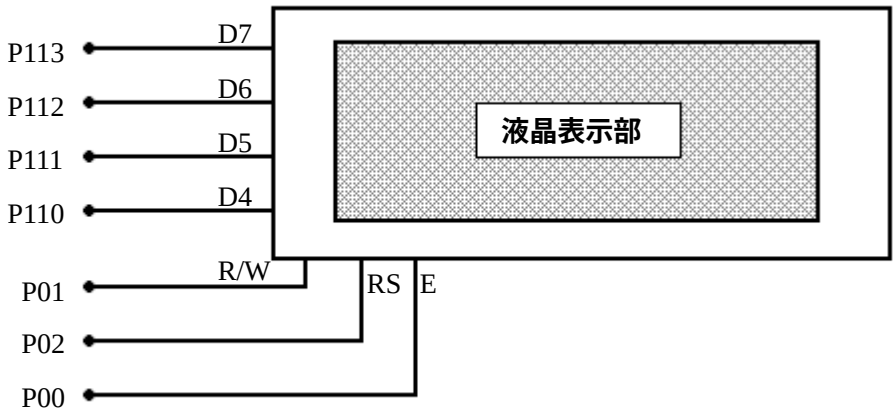

図 4.4-4　LCD ユニットとポートの接続

4.4.4　ステッピング・モータ

ステッピング・モータは、V853 のポートから信号を出力すると、ワンステップ駆動するような仕様です。これを実現するためにステッピング・モータ駆動回路を用意します。V853 からの信号をその回路で受信し、ステッピング・モータを回転させます。今回は V853 の出力ポートとして P03 を使用します。図 4.4-5 は回路の概略です。

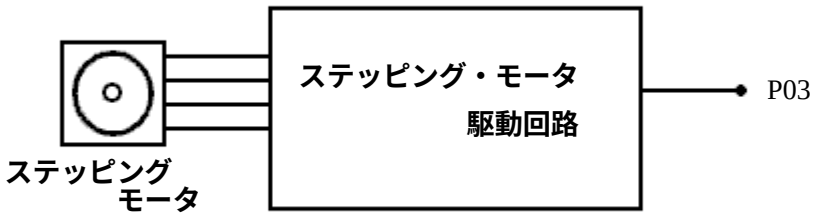

図 4.4-5　ステッピング・モータ駆動回路とポートの接続

4.4.5　圧電ブザー

圧電ブザーは、発振回路などで発振させることによって音を発生します。今回は発振回路を用意し、そこに V853 からの信号を入力することによって、一定時間発振させ、音を発生します。今回は V853 の出力ポートとして P25 を使用します。図 4.4-6 は回路の概略です。

図 4.4-6　圧電ブザーと発振回路とポートの接続

4.4.6　タッチセンサ

金属板に触れられたことを検出し、外部割り込み入力として認識します。金属板に手を触れると出力信号を HIGH にし、手が離れると Low にするような回路を用意します。図 4.4-7 は回路の概略です。

図 4.4-7　タッチセンサ出力回路とポートの接続

4.4.7　タクト・スイッチ

スイッチが押されると信号が HIGH レベルになる回路にします。具体的な回路は「第 7 章　回路図」をご参照ください。なおチャタリングはハードウエアで吸収します。

タクト・スイッチは 8 つ設け、すべて外部割り込みとして検出します。今回使用

する割り込み入力とタクトスイッチ番号の関係を、表 4.4-1 に示します。

タクト・スイッチ番号	入力ポート
1	INTP110
2	INTP111
3	INTP112
4	INTP113
5	INTP140
6	INTP141
7	INTP142
8	INTP143

表 4.4-1　タクトスイッチ番号と割り込み入力ポートの関係

4.4.8　温度センサ・湿度センサ

V853 に内蔵されている A/D コンバータを使用して、気温測定、湿度測定を行います。各センサから出力される電圧を測定することになりますが、A/D コンバータの性能より 0.1V～1.0V の範囲に値が来るように調整します。今回は各センサの出力を、表 4.4-2 に示すアナログ入力端子に接続します。

センサ	アナログ入力端子
温度センサ	ANI0
湿度センサ	ANI1

表 4.4-2　センサとセンサ入力ポートの関係

A/D 変換するには、基準電圧入力が必要となります。そこで、AV_{DD}（正電源供給）に 5V、AV_{REF1}（基準電圧）に 5V を供給し、AV_{SS} （グランド電位）を接地します。

第5章　　ソフトウエア

この章では、RX850 を用いたアプリケーション・プログラムの作成方法を説明します。「3.3 機能概要」で示した仕様に合うように、実際にアプリケーションを作成していきます。

5.1　　全体構成

システム構成の全体図を図 5.1-1、図 5.1-2 に示します。表 5.1-1 には、このアプリケーションに必要なタスクの一覧とその概要を示します。

タスク	概　要
7 セグメント LED ダイナミック・ドライブタスク	7 セグメント LED のダイナミック・ドライビングを行う
時間管理タスク	現在時刻を保持する
タイマー管理タスク	タイマ起動中、タイマ値を管理する
ストップウォッチ管理タスク	ストップウォッチ起動中、値を保持する
スロット管理タスク	スロット起動中、値を保持する
LED 点滅タスク	秒刻み LED を点灯・消灯を管理する
LCD 初期化タスク	LCD ユニットを初期化する
LCD 表示メッセージタスク（7 セグメント LED 表示情報）	LCD に表示するメッセージを LCD 表示データ管理タスクに渡す
LCD 表示データ管理タスク	LCD に表示するデータを管理する
LCD 表示タスク	LCD に表示するデータを LCD に送信する
気温・湿度情報通知タスク	気温・湿度情報を LCD 表示内容管理タスクに通知する
気温・湿度管理タスク	気温・湿度センサの情報を A/D コンバートし、管理する
気温・湿度表示自動切り換えタスク	一定時間システムに対して入力がなかった時に、LCD 内容を自動的に気温・湿度情報に切り替える
メモ管理タスク	シリアル通信で送られてくるデータ（メモデータ）を管理する
受信キャラクタ送信タスク	シリアル通信で受信したデータを送信側に通知する
ステッピング・モータ管理タスク	ステッピング・モータ回転、回転速度を管理する
ブザー管理タスク	圧電ブザーの ON/OFF を管理する
割り込み制御タスク	外部割り込み入力時（タクト・スイッチ）の処理制御を行う

表 5.1-1　タスク一覧とその概要

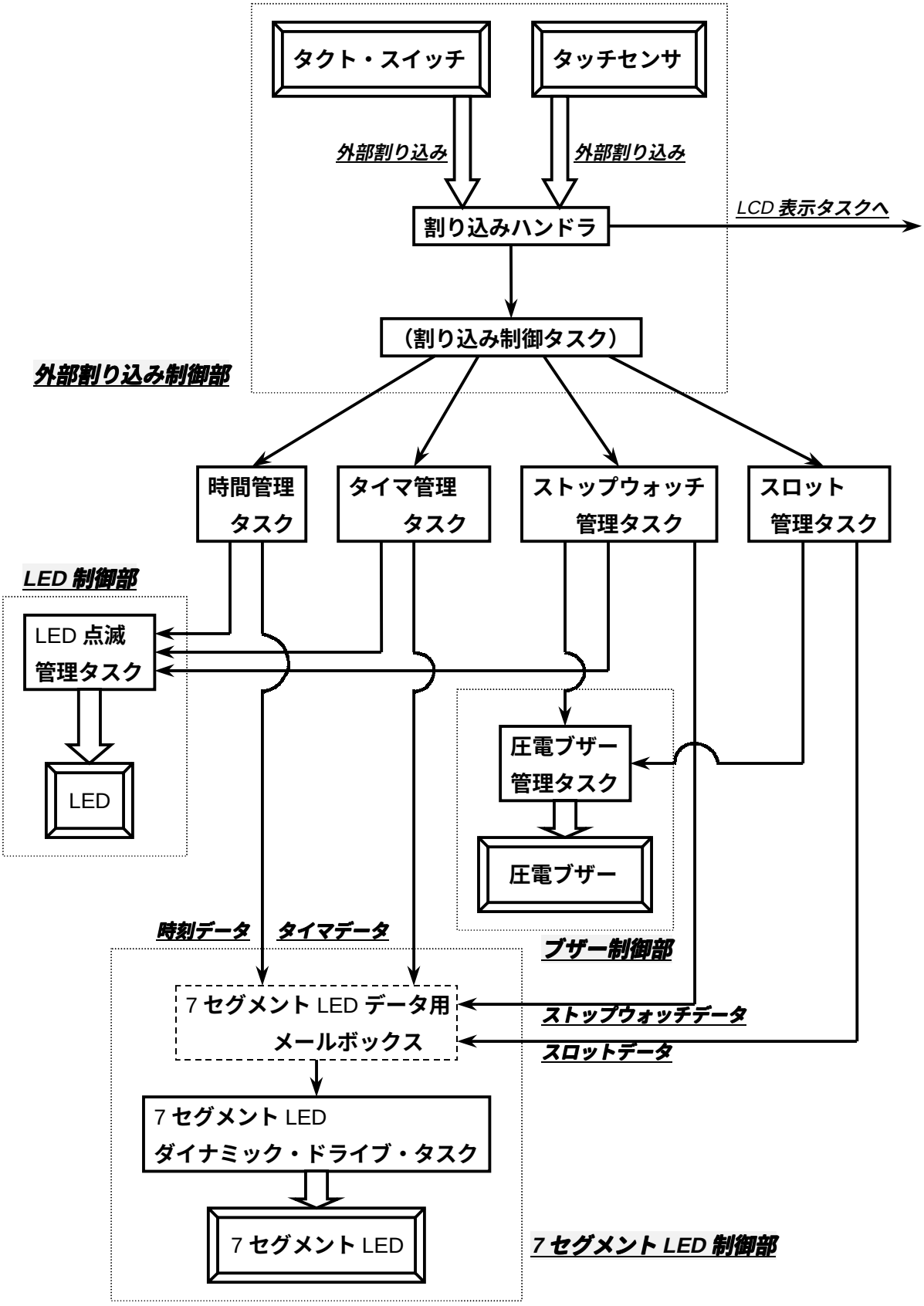
タクト・スイッチ
タッチセンサ
外部割り込み
外部割り込み
LCD 表示タスクへ
割り込みハンドラ
（割り込み制御タスク）
外部割り込み制御部
時間管理
タスク
タイマ管理
タスク
ストップウォッチ
管理タスク
スロット
管理タスク
LED 制御部
LED 点滅
管理タスク
LED
圧電ブザー
管理タスク
圧電ブザー
ブザー制御部
時刻データ
タイマデータ
7 セグメント LED データ用
メールボックス
ストップウォッチデータ
スロットデータ
7 セグメント LED
ダイナミック・ドライブ・タスク
7 セグメント LED
7 セグメント LED 制御部

図 5.1-1　全体図（1）

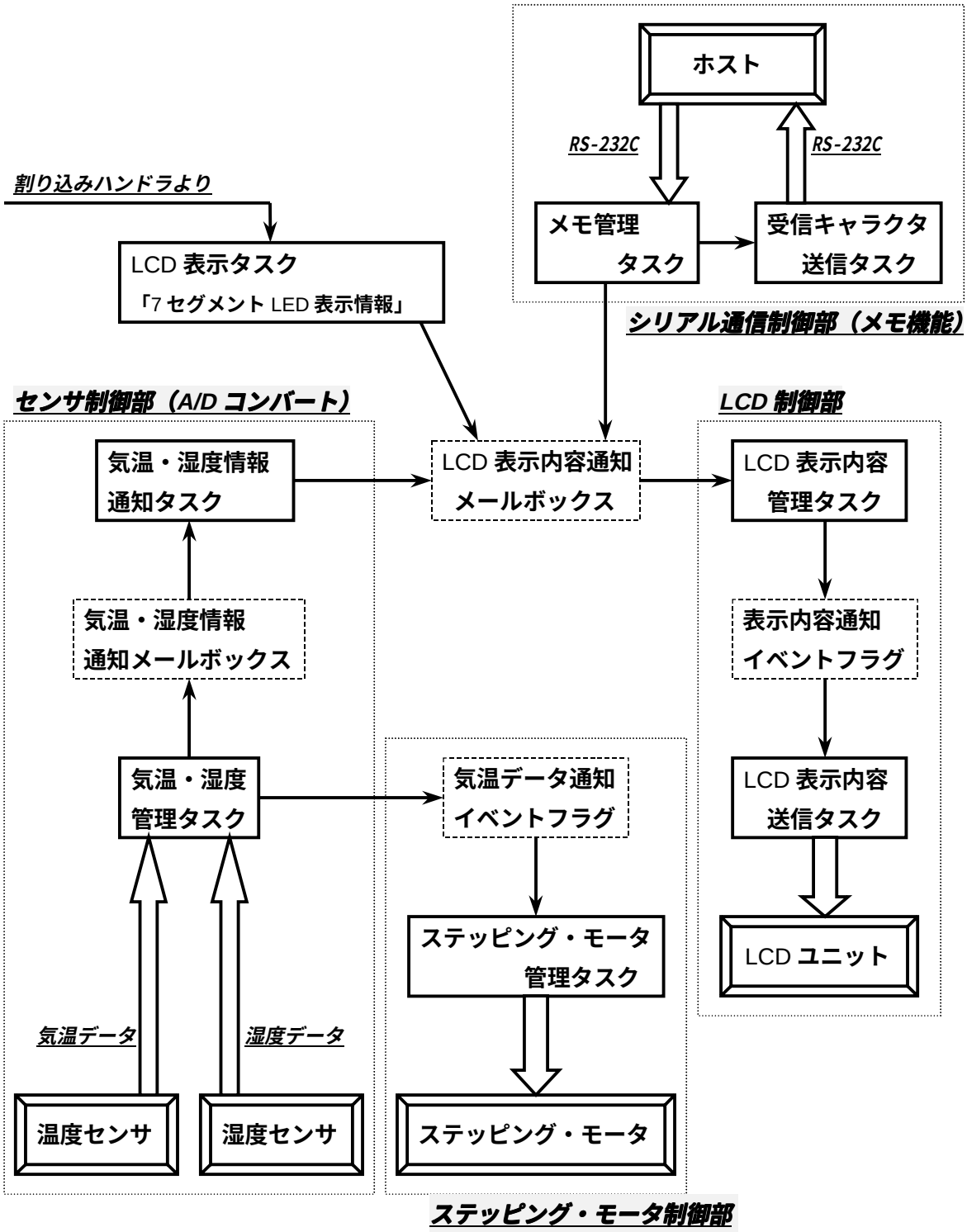
ホスト
RS-232C
RS-232C
割り込みハンドラより
メモ管理
タスク
受信キャラクタ
送信タスク
LCD 表示タスク
「7 セグメント LED 表示情報」
シリアル通信制御部（メモ機能）
センサ制御部（A/D コンバート）
LCD 制御部
気温・湿度情報
通知タスク
LCD 表示内容通知
メールボックス
LCD 表示内容
管理タスク
気温・湿度情報
通知メールボックス
表示内容通知
イベントフラグ
気温・湿度
管理タスク
気温データ通知
イベントフラグ
LCD 表示内容
送信タスク
ステッピング・モータ
管理タスク
LCD ユニット
気温データ
湿度データ
温度センサ
湿度センサ
ステッピング・モータ
ステッピング・モータ制御部

図 5.1-2　全体図（2）

5.2　RX850 の機能

「5.1 全体構成」で説明した全体構成図に従ってアプリケーションを開発していきます。

まず実際のアプリケーション作成例を示す前に、この5.2で RX850 の機能である『タスク管理』 『同期通信管理』 『メモリ管理』 『割り込み処理管理』 『時間管理』について、具体的な使用例をあげながら説明していきます。その後に今回のアプリケーション・プログラムについて説明していきます。

5.2.1　タスク管理

『タスク』はリアルタイム OS の処理単位です。今回のアプリケーションでは、「5.1 全体構成」を見ていただくとわかる通り 18 個のタスクを使用します。このような各々特定の機能を持つタスクを上手に切り替えて実行し、システムを動作させることが、リアルタイム OS を用いたアプリケーションを作成する上で基本となります。

5.2.1.1　タスクの切り分け

リアルタイム OS を用いたアプリケーションを作成するとき、どのようにタスクの切り分けを行うか、つまり機能別による分け方が重要になります。これがシステムの効率や、プログラムの理解しやすさにつながってきます。しかし切り分け方に決まった方法というものはありません。ただ作成するシステムに依存しますが、基本的なタスクの切り分け方と言われるものがあります。それらを次にあげます。

1. I/O に依存した処理部分は、その処理ごとにタスクに分ける

   今回のアプリケーションで行っている LCD 表示処理には、LCD に表示する文字を管理する処理と、実際に LCD にデータを送信する処理があります。LCD の I/O 速度は、V853 の I/O 速度に比べてかなり遅いため、間にウェイトを入れなければ、LCD へのデータ転送がうまく行きません。そのため、この 2 つの処理を別々のタスクに切り分け、タイミングを取りながら転送するようにします。こうするとウエイトを入れている間に他のタスクの処理を行うことができるため、無駄のない処理を実現することができます。

2. 機能的に異なるものは、別のタスクにする

今回のアプリケーションのように「時間管理を行う処理」や「タイマ管理を行う処理」など、機能的に違う処理は別のタスクにします。場合によってはそうしない方が良いこともありますが、プログラムの見やすさという点においても、こうする方が良いとされています。

3. 時間経過によって処理する部分は、該当部分を別タスクにする

今回のアプリケーションの中で、一定時間、システムに対して入力がなかったとき、LCD 表示を「気温・湿度表示」に自動的に切り替える処理があります。時間の管理はリアルタイム OS が行ってくれますので、ユーザは上の処理部分を 1 つのタスクとして、経過時間を気にせずにプログラミングすることが可能となります。

4. 周期的に実行される処理は、別タスクとする

規則正しい間隔で実行されるものは、その処理だけで閉じて 1 つのタスクにした方が良いとされています。例えば今回のアプリケーションで使用している「LED 点滅タスク」は 1 秒ごとに実行される周期的なタスクなので、これを 1 つのタスクとしてあります。周期的な処理はこのようなタスクを使用する方法の他に、周期ハンドラを使用する方法もあります。周期ハンドラに関しては「5.2.5.2 周期ハンドラ」をご覧ください。

5. 多くの計算を要する部分は別タスクとし、より低い優先度にする

多くの計算を必要とする処理を、他の処理を行うタスクと一緒にしてしまうと、その計算のためにずっと run 状態のままとなってしまい、他のタスクに処理が移らなくなってしまう可能性があるためです。今回はこれを意識してタスク分けを行ったところはありませんが、このような処理が必要なときは、別タスクとして扱った方が良いとされています。

5.2.1.2　　タスクの優先度（プライオリティ）

RX850 のタスク実行管理は、優先度（プライオリティ）によって行われています。つまり ready 状態にあるタスクのうち、一番優先度の高いタスクに CPU の使用権を与えて実行させることになります。そのため、この優先度のつけ方によって動作が変わり、場合によっては思い通りに動作しなくなることがあります。優先度のつけ方に関しては、以下の注意点があります。

1.　割り込みから起床されるタスクには、高いプライオリティをつける

　これは割り込みの応答性を高めるためです。割り込みからタスクを起床しても、起床されたタスクより優先度の高いタスクが ready 状態で存在すると、そちらの処理が優先されます。つまり割り込みから起床されたタスクの処理が遅延されてしまい、割り込みの意味がなくなってしまう可能性があるためです。割り込み処理に関しての詳しい説明は「5.2.4 割り込み処理管理」をご参照ください。

2.　周期的に起床されるタスクには、高い優先度をつける

　一定周期で処理するタスクは、きちんと一定の周期ごとに起床することが期待されます。このタスクが起床した時、より優先度の高いタスクが ready 状態で存在すると、周期的に起床されるタスクの処理が遅延されてしまい、一定周期ごとに動作することが出来なくなってしまう可能性があるためです。

3.　タスク間で同期をとっている場合、待ち側のタスクの優先度を高くする

　待ち側のタスクの優先度を、もう一方のタスクよりも高くしておくと事象の発生によって待ち側のタスクが動作するので、応答性がよくなります。

優先度のつけ方も、必ず上であげた方法を取らなくてはならないということではありません。これも作成するシステムに依存します。

しかし前述の通り、優先度を決める際、十分に検討しなければ意図した動作をしなくなることにつながり、バグの原因にもなります。優先度の数は、RX850 では 0～31 の 32 段階ですが、0 は RX850 で予約されているため、実質上 1～31 の 31 段階となります。この数は内部で使用する ready キューのサイズに関係してくるため、少ないほど RX850 が使用するデータ領域が小さくなります。一方、同じ優先度のタスクを複数作成すると、その中の 1 つのタスクが実行を独占するケースが出てきます。これを回避するような機構を作るなど、プログラム設計において工夫が必要となります。これらのことを考慮に入れて優先度の設定を行う必要があります。また設計段階で綿密に検討することも必要ですが、システム全体のテストを行うときに、優先度の微妙な調整を行うことも必要です。

5.2.1.3　　タスクのスタック

タスクのスタックは、タスクごとに必要です。スタックはタスクが関数呼び出しや自動変数、計算結果の一時退避などに使用する以外に、実行するタスクが他のタスクに切り替わるときや、割り込みが入ったときにそのタスクが使用していたレジスタの情報、戻りアドレス等を退避しておくための領域としても使用されます。そのためタスクの数とそのスタック領域のサイズが、システムで使用する RAM のサイズに大きく影響してきます。

関数呼び出しや自動変数などで、ユーザが使用する以外にタスクのスタックに積まれる内容は、

- タスク・コンテキスト
- 割り込みが入ったときにセーブされるレジスタ情報

です。これらの具体的な内容については「5.5.1 タスク・スタックのサイズ」に掲載してありますのでそちらをご覧ください。しかしアプリケーション作成開始時などには、あまりスタック・サイズに関して気にかけられないことが多いと思われます。ですからはじめのうちは、とりあえず大きめにスタックを取り、後で調整することがよく行われます。

5.2.1.4　　タスク実行権グループ

RX850 には「タスク実行権グループ」が存在します。これはタスクのスタックの共有を実現する機構です。つまり同時に実行することのない（同時に ready 状態になることはない）タスク同士のスタックを共有するようにしてメモリを節約します。同じ共有スタックを使用するタスクの集まりを「タスク実行権グループ」と呼びます。

ただし注意しなくてはならないのは、この機構はあくまでも同時に実行することのないタスク同士で有効であるということです。設計を誤ると意図した動作を得られなくなる可能性があります。

また RX850 では、スタックの共有に関してばかりではなく、タスクの実行に関する情報、つまりタスクの状態やイベントフラグの待ち条件などを「タスク実行権グループ管理テーブル」で管理しています。ですからスタックの共有をしないタスクを生成しても「タスク実行権グループ管理テーブル」と「タスク管理テーブル」の両方のテーブルが生成されます。「タスク管理テーブル」では起床要求の

ネスト・カウンタなどが管理されています。詳しくは「ユーザーズ・マニュアル　テクニカル編」をご参照ください。

今回のアプリケーションではこのタスク実行権を使用していません。しかし初期化ハンドラ（init_handler()）で行っている初期化処理を「初期タスク」という一つのタスクとして扱いたい場合は、このタスクと「7 セグメント LED ダイナミック・ドライブタスク」などを同一の実行権グループとしてスタックを共有することが可能です。初期タスクはシステム「起動時に一度だけ実行されればよいタスク」ということになるためで、こうした方がメモリの節約になります。

5.2.1.5　　タスクの処理概要

タスクの処理部分は、無限ループの処理概要で書かれるものが一般的です。ただし、初期タスクのように「一度しか実行されないタスク」や「ある処理をした後に終了させたいタスク」の処理概要は、プログラムの最後に ext_tsk()という自タスクを終了させるシステム・コールが書かれたものになります。図 5.2-1 は C 言語でタスクを記述した場合の処理概要です。左側が無限ループの処理概要、右側が最後に ext_tsk()を最後に記述して、自タスクを終了させる処理概要です。

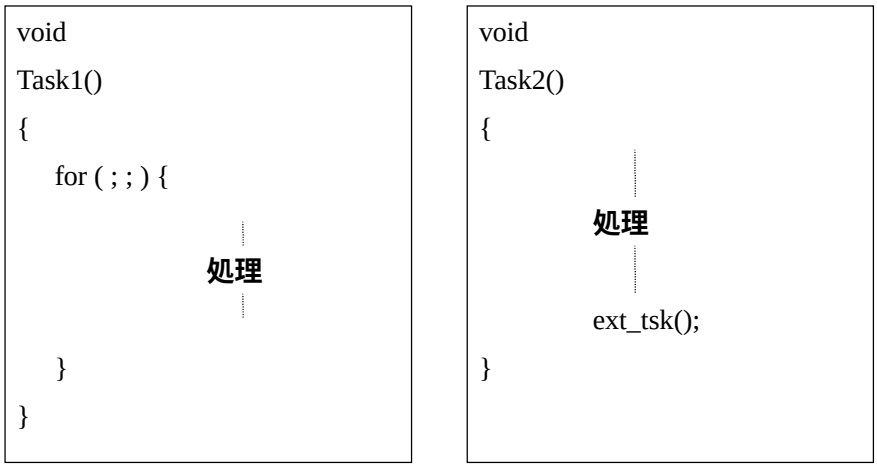

図 5.2-1　タスクの処理概要

5.2.2　同期通信管理

複数のタスクが並行に動作している環境、つまりマルチタスク環境では、あるタスクの処理結果により次に実行するタスクを切り替えるなど、処理内容をタスクの実行に反映させたい場合があります。そのためには他のタスクでの処理結果が出るまで待ったり、処理結果を通知してもらったりするなど、タスク間で同期を取ったり通信を行ったりする手段、つまり同期通信機能が必要になってきます。RX850 には、この機能を実現するために『イベント・フラグ』 『セマフォ』 『メールボックス』があります。

5.2.2.1　イベント・フラグ

タスク同士で同期を取るための機構です。イベント・フラグに値をセットするタスクと、ある値がセットされるのを待つタスクとの間で同期が取られます。RX850 には二種類のイベント・フラグがあります。

それは、値のセットは複数のタスクから行うことが出来ますが、イベント・フラグに待てるタスクは 1 つである「32 ビット幅のイベント・フラグ」と、複数のタスクがイベント・フラグに待てますが、イベントの発生と同時にそれらの待ちがすべて解除される「1 ビット・イベント・フラグ」です。

このイベント・フラグには様々な使用方法がありますが、ここでよく使われる例を 2 つあげます。

1. 複数のタスクが動作した後に、目的のタスクを動作させる

   例えば、タスク A、タスク B、タスク C がそれぞれ目的の処理をした後に、タスク D を動作させたいとします。タスクの優先度の関係は、

   タスク D ＞ タスク A ＞ タスク B ＞ タスク C

   とします。

   まずタスク D を動作させて wai_flg システム・コールを発行し、イベント・フラグ待ちの状態にします。条件としてフラグの内容が 0x7 になったら、待ちが解除されるようにします。

   次にタスク A が run 状態になります。タスク A は目的の処理をした後、set_flg システム・コールを使用して、その時のイベント・フラグの値と 0x1 との OR を取ったものをセットし、起床待ち状態にします。

   次にタスク B が run 状態になります。タスク B は目的の処理をした後、set_flg

システム・コールを使用して、その時のイベント・フラグの値と 0x2 との OR を取ったものをセットし、起床待ち状態にします。

次にタスク C が run 状態になります。タスク C は目的の処理をした後、set_flg システム・コールを使用して、その時のイベント・フラグの値と 0x4 との OR を取ったものをセットします。この時イベント・フラグの内容は 0x7 となり、タスク D は待ちが解除され、run 状態になります。図 5.2-2 はこの事例のイメージです。

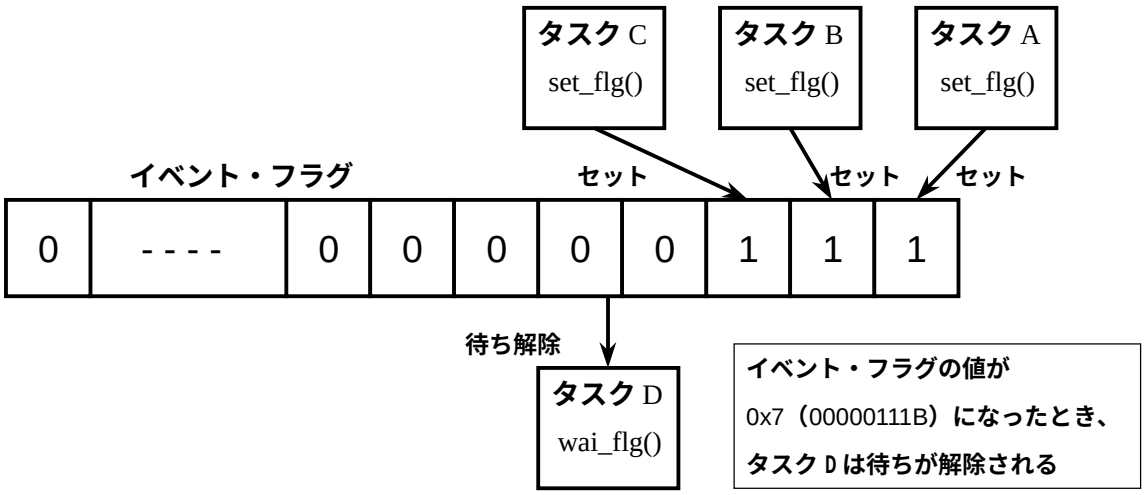

図 5.2-2　イベント・フラグ使用例 1

注意　：　RX850 では、イベント・フラグに値をセットするシステム・コール set_flg を発行すると、セットする前のフラグの値と OR を取った値がセットされます。つまりセット条件として OR や AND を選択することは出来ません。ただし待ち条件として OR、AND を設定することは可能です。

2.　データを受け取り次第、動作を再開する

この方法を使うと、イベント・フラグを 32 ビットのデータとして扱い、データ通信機能を実現することができます。あるタスクがイベント・フラグにデータをセットしたとき、受信側のタスクの待ちが解除され、動作を再開します。イベント・フラグが成立した時のフラグの値を参照することにより[注]渡されたデータを知ることができます。例えば JIS コードを受け渡したいときなどに有効です。この使い方をするときは、イベント・フラグのいずれかのビットに 1 が立ったとき、待ちを解除するようにしておきます。（0xffff で OR 待ちする）図 5.2-3 はこの事例のイメージです。

注 ： wai_flg システム・コールの第一パラメータで指定されたポインタに格納されています。

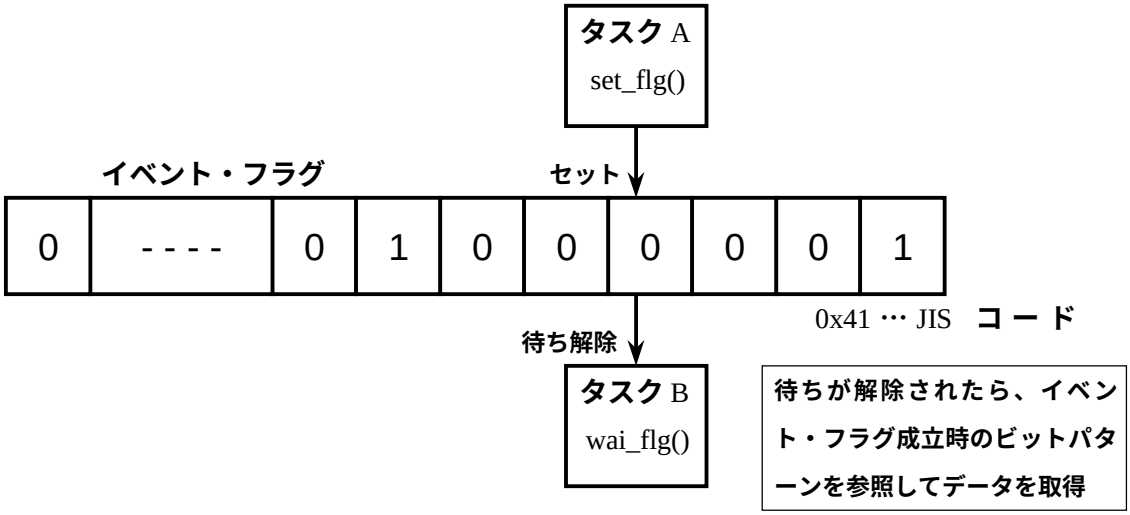

図 5.2-3　イベント・フラグ使用例 2

以上の例がよく使われます。

また wai_flg の代わりに pol_flg を用いると、イベント・フラグのポーリング動作を行うことができます。

1 ビット・イベント・フラグは、あるタイミングで複数のタスクの待ちを解除させたいときに使用されます。

なお 1 ビット・イベント・フラグに関するシステム・コールの種類は、32 ビット幅のイベント・フラグと同じですが、使用するシステム・コールが違います。詳しくは「ユーザーズ・マニュアル 基礎編」をご参照ください。

5.2.2.2　　セマフォ

タスク間の排他制御を行うための機構です。例えば、別々のタスクが同じ I/O やメモリにアクセスするような場合、それらのタスクを排他制御しないと、正常なデータを送信したり、書き込んだり出来なくなる可能性があります。このような競合を防ぐためにセマフォが用いられます。

RX850 のセマフォは非負数の計数型です。タスクがセマフォに対して資源を要求し、資源があれば取得してそのまま動作、資源がなければ取得できるまで待つ、といったように排他制御します。イベントフラグと同様にポーリング動作も用意されています。また 1 つのセマフォに対して、複数のタスクが資源待ちできます。この場合、資源待ちとなった順番で待ちキューにキューイングされます。

例えば、シリアル通信回線を使用するタスクを排他制御したいときは、シリアル通信を排他制御するためのセマフォを 1 つ作ります。タスクがシリアル通信回線を使用したいときは、まずこのセマフォに対して回線使用権、つまり資源を要求します。他にシリアル通信回線を使用しているタスクがなければ、回線使用権を取得でき、シリアル通信を行うことができます。シリアル通信が終わったら、セマフォに回線使用権を返却します。

今回のアプリケーションでは、LCD 表示内容管理タスクから、LCD 表示内容送信タスクに 1 文字ずつ送る際、互いのタイミングを取るためにセマフォを使用しています。

5.2.2.3　　メールボックス

タスク間でデータのやり取りをするための機構です。「5.2.2.1 イベント・フラグ」でも、使用例 2 のように 32 ビット以内のデータであれば、データ通信に使用することができますが、このメールボックスを使用すると、より大きなデータの通信が可能となります。

メールボックスを使用したメッセージ通信の標準的な手順は、次のようになります。

まずメッセージを書き込む領域が必要です。この領域を RX850 が管理しているメモリ・プールから取得します。get_blf システム・コールによって、そのメモリ・プールよりメモリ・ブロックを取得し、メッセージをその領域に書き込みます。そして snd_msg システム・コールによってメッセージをメールボックスへ送信します。受信側は rcv_msg システム・コールでメッセージを受信し、不用になった

メッセージ領域を rel_blf システム・コールによって解放（メモリ・プールに返却）します。（RX850 のメモリ管理に関しては、「5.2.3 メモリ管理」をご参照ください）メモリ・ブロックを使用することで、メモリの管理を OS に委ねることができ、ユーザはこのことを特に意識しないでコーディングすることが可能です。図 5.2-4 がこの方法を用いたメッセージ通信のイメージです。

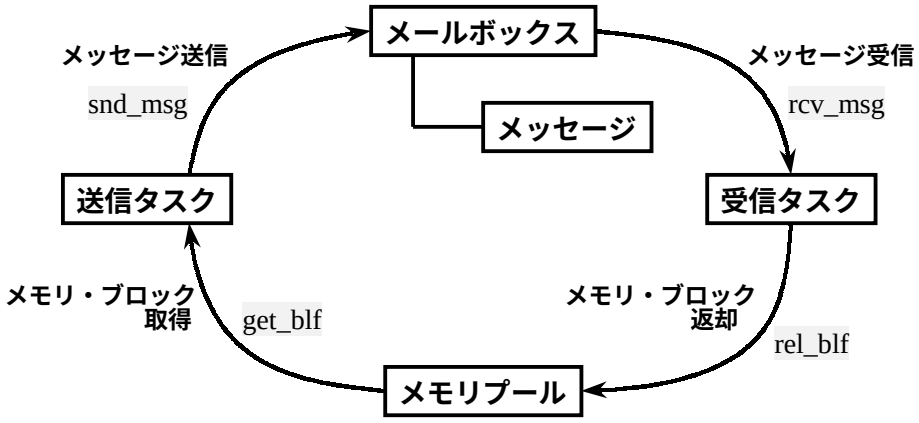

図 5.2-4　標準的なメッセージの送受信

今回のアプリケーションでは、7 セグメント LED に表示する数字の情報や、気温・湿度データを送信するのに用いています。

この他に、一度取得したメモリ・ブロックを他のタスクで使い回すため方法があります。つまりメモリ・ブロックの取得・返却を繰り返さず、一度取得した領域をメッセージ領域として通信します。今回のアプリケーションで LCD に表示する文字列を格納する領域は、この方法を用いて通信しています。LCD に一度に表示できるメッセージは一つしかないため、一度取得した領域をメモリ・プールに返却せずに、使用したいタスクにそのまま渡して再利用した方が効率が良くなるためです。

受信側のタスクが rcv_msg システム・コールを発行した際、メールボックスにメッセージがない時は、メッセージが到着するまで待ち状態になります。prcv_msg システム・コールを発行した場合は、メッセージがあるかどうかを確認し、あれば受信、なければエラーを返すというポーリング動作を行います。

メールボックスを用いたメッセージ通信において注意点が 2 つあります。

一つは、メールボックスで通信されるのは、メッセージそのものではなく、メッセージ領域の先頭アドレスであるということです。メッセージの内容を参照するときは、ポインタで参照することになります。

もう一つはメッセージの構造です。メッセージがメールボックスにキューイングされる際、RX850 はメッセージのリンクアドレスを、メッセージ本体の先頭から 4 バイト分に無条件に書き込みます。またそのメールボックスのメッセージキューイング方法が「優先度順」で指定されていた場合、さらにその後 1 バイト分にメッセージの優先度が書かれることになります。ですからメッセージ本体を先頭から書き込むと、RX850 によって 5 バイト分が書きつぶされてしまうことになります。メッセージの本体は、確保したメッセージ領域の先頭から 5 バイト先より書くようにしてください。（実際はアラインメントも考慮に入れ、8 バイト目から使用します）使用するメモリ・ブロックも、メッセージ本体のサイズ＋5 バイト取得する必要があります。なおメールボックスで通信されるものは、そのリンク領域を含めたものとなります。実際に C 言語でコーディングするときは、メッセージ構造体「T_MSG」を利用すると便利です。詳しくは「ユーザーズマニュアル 基礎編」のメールボックス関係のシステム・コールの説明をご参照下さい。

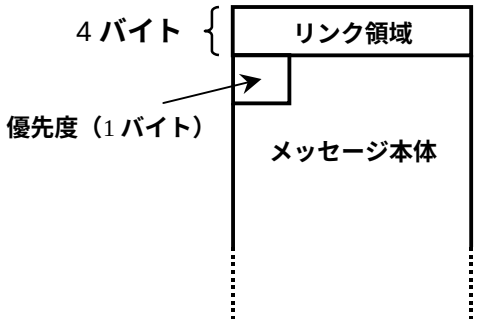

図 5.2-5　メッセージの構造

5.2.3　メモリ管理

RX850 では『メモリ・プール』と『メモリ・ブロック』に分けて、メモリを管理しています。メモリ・プールはコンフィギュレーション・ファイルで静的に生成します。実際にアプリケーションでメモリ領域を取得するときは、メモリ・プールから「1 メモリ・ブロック単位」で動的に取得します。1 つのメモリ・ブロックのサイズ、及びメモリ・プールから取得できるメモリ・ブロックの総数は、コンフィギュレーション・ファイルで指定します。

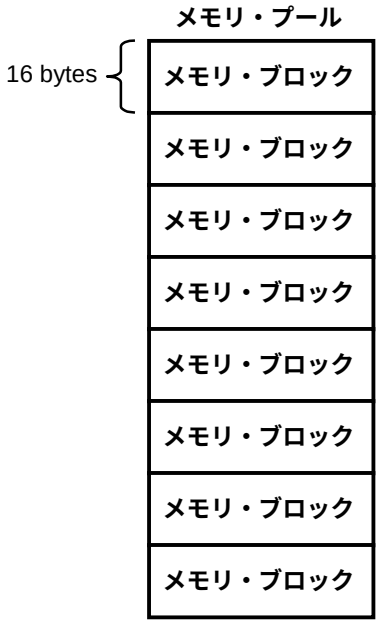

図 5.2-6　メモリ・プールのイメージ

（1 メモリ・ブロック 16 バイト・ブロック総数 8）

図 5.2-6 がメモリ・プールのイメージです。各メモリ・ブロック内には OS が管理するための領域（管理領域など）は取られませんので、指定したサイズ（図の例では 16 バイト）すべてを自由に使用することができます。ただしメモリ・ブロックをメールボックスのメッセージ領域として使用する際は、OS が使用するリンク領域に注意が必要です。詳しくは「5.2.2.3 メールボックス」をご参照ください。

メモリ・ブロックを取得するときは、get_blf システム・コールを、解放（返却）するときは rel_blf システムコールを使用します。ここで注意が必要なのは、get_blf

システム・コールで取得できるブロックは 1 ブロックであるということです。一度に 2 ブロックという具合に、取得するブロック数を指定することはできません。つまり異なるサイズのメモリ・ブロックを、同一のメモリ・プールから取得することはできないということになります。異なるサイズのメモリ・ブロックを使用したいときは、1 ブロックのサイズが違うメモリ・プールを複数作成しておきます。図 5.2-7 のように 1 ブロックのサイズが 16 バイトの「メモリ・プール 1」、1 ブロックのサイズが 32 バイトの「メモリ・プール 2」という具合に、複数のメモリプールを作成し、用途に応じて使い分けることになります。

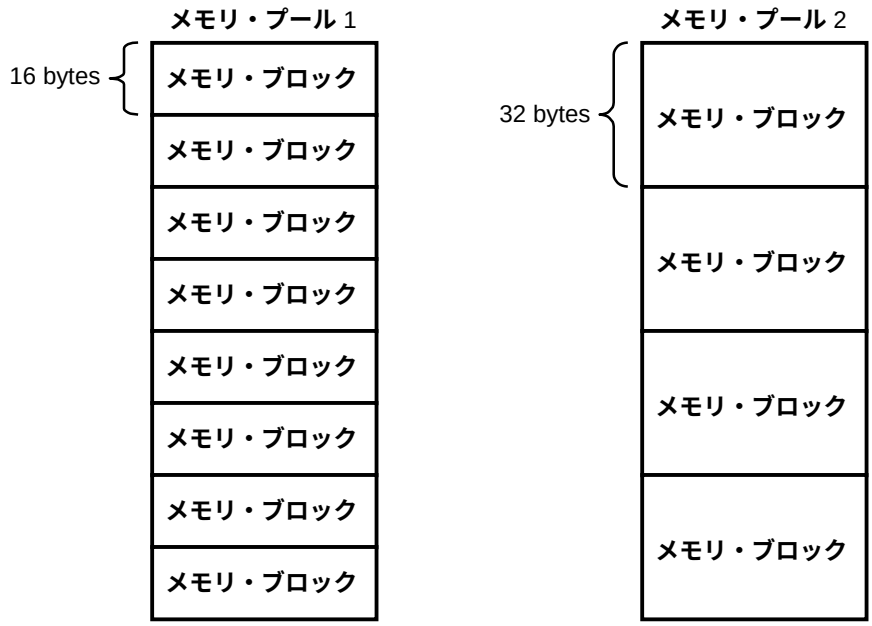

図 5.2-7 ブロックサイズの異なるメモリ・プールのイメージ
（左：1 メモリ・ブロック 16 バイト・ブロック総数 8）
（右：1 メモリ・ブロック 32 バイト・ブロック総数 4）

タスクから get_blf システム・コールを発行したとき、メモリ・プールからメモリ・ブロックが取得できなかったときは、取得できるまで待ち状態に入ります。どうしても待ちになっては困る、つまり確実に取得したい場合は、そのタスク専用のメモリ・プールを作った方が賢明です。また pget_blf システム・コールを使用すると、取得できるならそのまま取得、できなければ、エラーを返すというポーリング動作を行います。

RX850 を使用したアプリケーションを作成する際、メモリ領域としてメモリ・

プールを使用することを推奨していますが、必ずメモリ・プールを使用しなくてはならないというわけではありません。ユーザが管理できるメモリ空間であれば、その領域を使用しても構いません。もちろんその領域を使用してメールボックスを使用したメッセージ通信を行うことも可能です。

5.2.4　割り込み処理管理

システムに割り込みが入ると、その割り込みに対応したルーチンに制御が移ります。そのルーチンは「割り込みハンドラ」と呼ばれ、V853 の各ポート割り込みと割り込みハンドラを一対一に対応させることによって、割り込み処理を実現します。

RX850 の割り込みハンドラには、『間接起動割り込みハンドラ』と『直接起動割り込みハンドラ』の二種類があります。

マルチタスク OS では、割り込みが入ったとき、割り込み処理本体を実行する前に、それまで実行していたタスクや割り込みハンドラ（多重割り込み時）が使用していたレジスタ情報等をセーブ（退避）しなくてはなりません。それは割り込み処理から復帰するときに、中断されたタスクや割り込みハンドラを正しく再開させるためです。

このレジスタの退避・復帰作業を RX850 に任せてしまい、多少割り込み反応速度は落ちるものの、ハンドラ内の記述を簡単にできるのが「間接起動割り込みハンドラ」、レジスタの退避・復帰等の作業を記述しなくてはならないものの、ハードウエアの限界に近い割り込み応答速度を実現できるのが「直接起動割り込みハンドラ」です。

5.2.4.1　間接起動割り込みハンドラ

間接起動割り込みハンドラは、前述のようにレジスタ情報等の退避処理を OS に任せてしまいます。ですからユーザは割り込みハンドラの登録と割り込み処理の記述を行えばよいことになります。割り込みハンドラの登録は、コンフィギュレーション・ファイルに割り込みポートと割り込みハンドラの対応を記述することによって行います。

コンフィギュレーション・ファイルで、次のように間接起動割り込みハンドラを登録したとします。

```
inthdr      INTP110    _Handler1
```

… NEC ツールを使用する場合

```
inthdr          4          _Handler1
```

… GHS ツールを使用する場合

これは「INTP110 に割り込みが入ったときに起動される割り込みハンドラは、先頭ア

ドレスが Handler1 である」という意味です。つまり C 言語で言えば、関数 Handler1() ということになります。なおコンフィギュレーション・ファイルの記述方法などについては「ユーザーズ・マニュアル　インストレーション編」をご参照ください。

割り込みハンドラ Handler1()は、次のような形で記述されます。

```
short
Handler1(void)
{
        処理
        return (0xff);
}
```

図 5.2-8　間接起動割り込みハンドラの記述例

処理の最後にある return (0xff) は、間接割り込みハンドラから復帰する命令です。引数が「0xff」であれば、間接起動割り込みハンドラから普通に復帰、「タスク ID」であれば、間接起動割り込みハンドラから復帰するとともに、その ID を持つタスクの起床を行います。なお引数を指定しなかった場合、予期しない動作を引き起こしますので必ず指定します。

5.2.4.2　　直接起動割り込みハンドラ

直接割り込みハンドラを使用すると、割り込みが入ってから終了するまでの処理をすべてユーザが記述しなければなりません。

まず直接割り込みハンドラの登録は、割り込みが発生した際に V853 が制御を移すアドレス、つまり「ハンドラ・アドレス」に直接、割り込み処理を記述する、または直接割り込みハンドラへの分岐命令を記述することによって行います。

ここでは「直接割り込みハンドラへの分岐命令を設定して処理する」方法について、具体的な記述方法を説明します。

直接割り込みハンドラを記述する際は、アセンブリ言語を使用します。ただし割り込み処理本体は C 言語で記述し、ハンドラから jarl 命令を使用して呼び出すという形をとることもできます。（図 5.2-12 参照）

例として INTP110 の割り込みが入ったときに起動される直接割り込みハンドラを登録します。

プログラム中に次のような記述をします。

```
.section　“INTP110”,.text
.extern　 _Handler1
jr　_Handler1
```

図 5.2-9　ハンドラ・アドレスへの登録（NEC 版）

```
.org　 0x000000c0
.extern　 _Handler1
jr　_Handler1
```

図 5.2-10　ハンドラ・アドレスへの登録（GHS 版）

このように記述することによって、INTP110 の割り込みが入ったとき、_Handler1 というラベルへ分岐することになります。_Handler1 には次のような内容を記述します。

```
　.globl　 _Handler1
_Handler1 :
```

1. スタック・ポインタを取得
2. 退避すべきレジスタを退避
3. EIPC・EIPSW を退避
4. スタック・ポインタを割り込みスタック領域に切り替え
5. 元の sp と lp を退避
6. 割り込み本体処理
7. レジスタ復帰
8. ret_int または ret_wup へ処理を移行

図 5.2-11　直接起動割り込みハンドラの記述内容

直接割り込みハンドラから抜けるときは、ret_int システム・コール、または ret_wup システム・コールを使用します。ret_int システム・コールは、直接割り込みハンドラからそのまま復帰するときに使用し、ret_wup システム・コールは、直接割り込みハンドラから復帰するとともに、引数で指定した ID を持つタスクの起床を行ってから復帰するときに使用します。

レジスタの退避・復帰についてですが、RX850 のパッケージにはこれらの処理を記述したマクロが用意されています。プログラム中でそのマクロを記述することにより、レジスタの退避・復帰処理を実現します。なお、マクロの使い方については、「ユーザーズ・マニュアル　基礎編」をご参照ください。

なお、退避・復帰するレジスタは、表 5.2-1 のようになります。使用するレジスタモードによって異なります。、

22 レジスタ・モード	EIPC、EIPSW、r1、r6～r9、r10～r14
26 レジスタ・モード	EIPC、EIPSW、r1、r6～r9、r10～r16
32 レジスタ・モード	EIPC、EIPSW、r1、r6～r9、r10～r19

表 5.2-1　退避・復帰するレジスター覧（r10 番台に違いがあります）

この他にユーザが割り込み処理で使用するレジスタがあれば、マクロの記述の後に、それらを退避する記述を行います。

割り込み処理の最後では、レジスタの復帰作業を行います。復帰作業も専用のマクロを使用して記述し、その後 ret_int システム・コール、ret_wup システム・コールによって割り込みから復帰します。もし、表 5.2-1 以外でユーザが割り込み処理で使用したレジスタがあれば、ret_int システム・コールまたは ret_wup システム・コールを発行する前に、それらを復帰する処理を記述します。図 5.2-12 は直接起動割り込みハンドラの記述例です

```
-- 直接起動割り込みハンドラ
    .globl    _Handler1
    .align  4
_Handler1 :
    RTOS_IntEntry          -- 割り込み開始マクロ 1
    RTOS_IntPrologue       -- 割り込み開始マクロ 2

    extern _inthdr_body
    jarl _inthdr_body, lp  -- 割り込み処理本体へ

    mov   r10,   r6        -- ret_wup で終了するときのみ必要
    RTOS_IntEpilogue       -- 割り込み終了マクロ
    jr   _ret_wup
```

```
/* 割り込みハンドラ本体処理 (C 言語記述) */
#include   “stdrx850.h”
ID inthdr_body(void)
{
    [ ハンドラ本体処理 ]
    return (tskid);    /* tskid : 起床させるタスクの ID 番号 */
}
```

図 5.2-12　直接起動割り込みハンドラの記述例

5.2.4.3　　割り込み処理における注意点

割り込みハンドラ処理内でも、数々のシステム・コールを発行することが可能です[注]。しかし待ち状態を引き起こすシステム・コールなど、RX850 の内部処理の関係で発行することが禁止されているシステム・コールが存在します。発行可能なシステム・コールの一覧については「ユーザーズ・マニュアル 基礎編」をご参照ください。

注 :　ret_int システム・コールと ret_wup システム・コールは、直接割り込みハンドラ内でのみ発行可能です

システム・コール発行後のスケジューリング動作についてですが、タスク内で発行されたときと、割り込みハンドラ内で発行されたときの動作が異なります。タスク内のときは即座にスケジューラが起動されますが、割り込みハンドラ内のときは、即座には起動されず、割り込みハンドラ本体の処理が終わってから一括して行われます。

5.2.4.4　　割り込み処理の実際

割り込み処理は、その反応速度が重要になります。割り込み内での処理が長すぎると応答性が悪くなり、システムに支障をきたす場合があります。ですから、長くなる処理を記述しないようにするのがベストです。もし長くなってしまう処理や、多少緩やかな応答速度でも構わないときは「割り込みタスク」を使用します。「割り込みタスク」とは、割り込みハンドラから起床されるタスクのことで、他のタスクと同様に扱われます。その中で長くなってしまう処理を記述しておけば、システムとしての応答速度を犠牲にすることなく割り込み処理をさせることができるようになります。今回のアプリケーションでは、この割り込みタスクを使った処理と、割り込みハンドラだけを使った処理の両方を使用しています。ただし割り込みタスクを使用したときは、タスクの優先度が関係してくるので注意が必要です。タスクの優先度については、「5.2.1.2 タスクの優先度（プライオリティ）」をご参照ください。

5.2.4.5　　多重割り込みについて

割り込み処理中に割り込みが入ることを、多重割り込みといいます。RX850 では、多重割り込みを使用することが可能です。しかし RX850 では、割り込みハンドラは、割り込みを禁止の状態で実行されますので、そのままでは多重割り込み

ができません。ですから割り込みハンドラ内で、ユーザが EI 命令を使用して割り込みを許可する必要があります。今回のアプリケーションでは多重割り込みは使用していません。

5.2.4.6　　NMI 割り込みについて

NMI（ノンマスカブル割り込み）は、どんな時でも受け付けられる割り込みです。RX850 ではこの NMI 割り込みハンドラ内で「システム・コールを発行することを禁止」しています。使用する際は RX850 の動作に直接影響を及ぼすようなことをしないようにします。

5.2.5　時間管理

時間と同期した処理を実現するための機構です。RX850 には指定した時間だけタスクを待ち状態にするための「時限待ち」、周期的な処理を行うための「周期ハンドラ」があります。

5.2.5.1　時限待ち

自タスクを指定時間だけ待ち状態にしたい、つまり時限待ちをしたい場合は、tslp_tsk システム・コール、または dly_tsk システム・コールを使用します。時限待ちの単位は tick、つまりクロック割り込み周期となります。例えば 1tick が 1ms で、自タスクを 1 秒間待ち状態にしたい場合、tslp_tsk システム・コールを使用する場合は「tslp_tsk(1000)」、dly_tsk システム・コールを使用する場合は「dly_tsk(1000)」と記述します。なお 1tick の値は V853 のタイマ・コントロール・レジスタで設定します。

これらのシステム・コールは良く使われ、今回のアプリケーションでも数箇所で使用しています。tslp_tsk システム・コールと dly_tsk システム・コールは、機能的には同じですが戻り値が違います。tslp_tsk システム・コールの戻り値は、

- 指定時間が経過した
- 他タスクから起床された
- rel_wai システム・コールにより、起床待ち状態を強制的に解除された

dly_tsk システム・コールの戻り値は

- 指定時間が経過した
- rel_wai システム・コールにより、起床待ち状態を強制的に解除された

となります。tslp_tsk システム・コールを使用すると、指定時間が経過して起床したのか、それとも他タスクから起床されたのかを戻り値で知ることができます。今回のアプリケーションで LCD 表示内容を自動的に切り替える処理と、割り込みによって切り替える処理を区別はこれを使用しています。

単純にタスクの処理を遅延させたい場合は、dly_tsk システム・コールを使用した方がよいとされています。例えば、7 セグメント LED のダイナミック・ドライブを行う際、点灯周期を調整するために dly_tsk システム・コールを使用していま

す。

5.2.5.2　　周期ハンドラ

今回のアプリケーションのように時刻を刻みたいときは、1 秒周期に処理を行わなければなりません。そこでこの周期ハンドラを使用します。

周期ハンドラとは、一定周期ごとに起動される「割り込みハンドラ」のことをいいます。周期ハンドラの割り込み要因は、「タイマ割り込み」となります。コンフィギュレーション・ファイルで「システムで使用するクロック要因」を指定しますが、これが割り込み要因となります。

周期ハンドラの記述方法ですが、間接起動割り込みハンドラの記述方法とほとんど同じです。ただし周期ハンドラから復帰するときは、図 5.2-11 のように戻り値を必要としません。

```
void
Time1(void)
{
        処理
        return;
}
```

図 5.2-12　周期ハンドラの記述例

周期ハンドラも割り込み処理なので、その応答速度が重要になります。ハンドラ内での処理が長すぎると応答性が悪くなり、一定周期を刻めないなどシステムに支障をきたす場合があります。ですから長くなる処理を記述しないようにするのがベストです。もし処理が長くなってしまうようなときや、多少緩やかな応答速度でも構わないときは「割り込みタスク」を使用します。今回のアプリケーションで使用している周期ハンドラ内の処理は、すべて割り込みタスクで行っています。

5.3　各機能の実現

今回のアプリケーションの題材である時計盤システムで使用する各機能を、RX850 を使用して実現する方法について、それぞれ説明します。

5.3.1　7 セグメント LED ダイナミック・ドライブ

4 桁の点灯を 1 桁ずつ行い、それを高速に切り替えることによって、4 桁すべてを点灯しているように見せる方法が、LED のダイナミック・ドライブ方式です。今回はこの高速切り替えの間隔を 4ms で行い、1 周期 16ms で点灯を繰り返します。

このダイナミック・ドライブ処理は「7 セグメント LED ダイナミック・ドライブ・タスク」で行います。このタスクで行う処理は次のようになります。

P10～P16 へセグメントデータを出力し、P30～P33 へ点灯させたい桁を選択して出力します。つまり P10～P16 で表示させる数字データの出力を、P30～P33 で桁の切り替えを行います。一つの桁に対してこの動作をした後、4ms の間タスクを待ち状態にさせます。桁を変えながらこの動作を繰り返すことにより、ダイナミック・ドライブが実現します。この処理の流れは、図 5.3-1 のようになります。

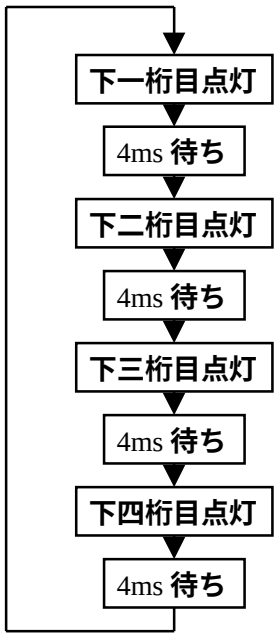

図 5.3-1　7 セグメント LED ダイナミック・ドライブの流れ図

上の「待ち状態」を作るのは dly_tsk システム・コールです。今回はシステム・クロック（タイマ）のインターバルを 1ms に設定してあるので、dly_tsk システム・コールの引数を「4」とすると、タスクを 4ms 待ち状態にすることができます。

次に数字データについてです。7 セグメント LED に表示するデータは、現在時刻、タイマーの値、ストップウォッチの値、スロットの値で、モードが切り替わるとそれぞれその時点での値を表示しなければなりません。つまり表示すべき数字データを逐次受け取る必要があります。このデータ通信の手段としてメールボックスを使用します。

メールボックスからメッセージを受け取るシステム・コールとして rcv_msg があります。しかしこのシステム・コールを使用すると、メールボックスにメッセージがなかった場合、メッセージが到着するまでタスクは待ち状態に移行してしまいます。そうなると 7 セグメント LED の点灯が途中で止まってしまうことになります。これを回避するために prcv_msg システム・コールを使います。これはメールボックスのポーリング動作を行うシステム・コールです。つまりメッセージが届いていれば受け取り、それを数字データとしてポートに出力、なければ今まで点灯していた数字データをそのままポートに出力します。

図 5.3-2 は、C 言語で記述したときの、ダイナミック・ドライブ・タスクの処理概要です。

```
ダイナミック・ドライブ・タスク()
{
    for ( ; ; ) {
        for ( 4 回ループ) {
        switch (点灯する桁) {
                    case   一桁目 :
                        prcv_msg();      /* メールボックスをポーリング */
                        if (メッセージがあったとき) {
                            そのデータを LED に表示;
                            メッセージ領域を解放;
                        } else {
                            それまでのデータを LED に表示;
                        }
                        dly_tsk (4);      /*  4ms 待つ  */
                        break;
                    case   二桁目 :
                            （一桁目と同じ）
                    case   三桁目 :
                            （一桁目と同じ）
                    case   四桁目 :
                            （一桁目と同じ）
            }
        }
    }
}
```

図 5.3-2　LED ダイナミック・ドライブ・タスクの処理概要

5.3.2　秒刻み LED の点滅

秒を刻む LED の点滅方法です。点滅を行う処理は「LED 点滅タスク」で行います。

LED の点滅で秒を刻むとき、点灯してから 0.5 秒後に消灯、さらに 0.5 秒経つと点灯することを繰り返します。点灯を開始するタイミングは、現在時刻やタイマー、ストップウォッチの現在値に依存します。これらを管理しているタスクから点灯の合図が出されたとき、LED 点滅タスクは、LED を点灯してから 0.5 秒待ち、LED を消灯する処理を行います。その後再び点灯の合図が出るまで待てばよいことになります。

このような処理を実現するには、「自タスクを起床待ち状態に移行」、「他タスクの起床」の機構を使用すると便利です。

つまり LED 点滅タスクは、初めは起床待ちの状態です。ある時、他のタスク、この例であれば時間管理タスクなどが LED 点滅タスクを起床し、LED 点滅タスクは点灯、消灯の処理を行って再び起床待ち状態へ移行することを繰り返します。こうすることによって秒を刻む LED の点滅を実現することができます。この処理の流れは図 5.3-3 のようになります。

自タスクを起床待ち状態にするには slp_tsk システム・コールを、他タスクを起床するには wup_tsk システム・コールを使用します。

この時、起床されるタスクの優先度と、起床するタスクの優先度の関係に注意しなければならないときがあります。起床されるタスクの処理に速い応答性が必要なときは、起床するタスクよりも優先度を高くしておく必要があります。今回の LED 点滅タスクの場合、高速な処理は要求されないのでどちらでも構いません。

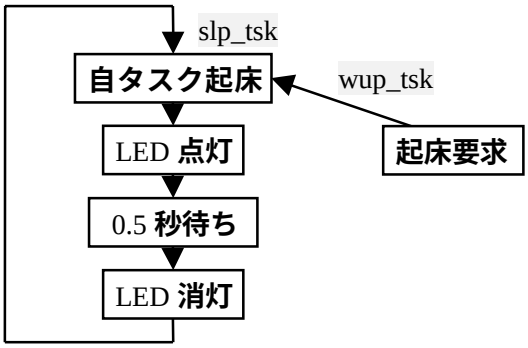

図 5.3-3　LED 点滅タスクの流れ図

図 5.3-4 は C 言語で記述したときの、LED 点滅タスクの処理概要です。

```
LED 点滅タスク()
{
    for ( ; ; ) {
        slp_tsk ();          /*  自タスク起床待ち    */
        LED 点灯;
        dly_tsk (500);      /* 0.5 秒間待ち  */
        LED 消灯;
    }
}
```

図 5.3-4　LED 点滅タスクの処理概要

5.3.3　現在時刻管理

時を刻む処理の管理方法について説明します。時刻は 24 時間単位で管理し、秒刻みを LED の点滅にて表示します。

5.3.3.1　現在時刻を進める処理

一般に時計は 1 秒単位で時を刻みます。このような 1 秒ごとという周期的な処理を RX850 で行うには「周期ハンドラ」を使用します。起動周期を 1 秒に設定すると、周期ハンドラ起動開始時から 1 秒ごとに起動され、処理が行われます。
現在時刻管理の内容には、次のようなものがあります。

- 今、7 セグメント LED に現在時刻が表示されているかどうかの判別
- ダイナミック・ドライブ・タスクへ現在値を送信
- 時を刻む（桁繰り上がり判別等を含む）
- 秒刻み LED タスクの起床

このすべてを周期ハンドラ内で処理すると、「5.2.5.2 周期ハンドラ」で述べたように、スムーズな処理が行えないばかりでなく、他のタスクの処理が遅延されるなど、システムに支障をきたす場合があります。そこでこれらの処理をすべて 1 つのタスクで扱い、周期ハンドラの処理はこのタスクを起床するだけにとどめます。このタスクを「現在時刻管理タスク」と呼ぶことにします。

現在時刻処理の流れは、図 5.3-5 のようになります。

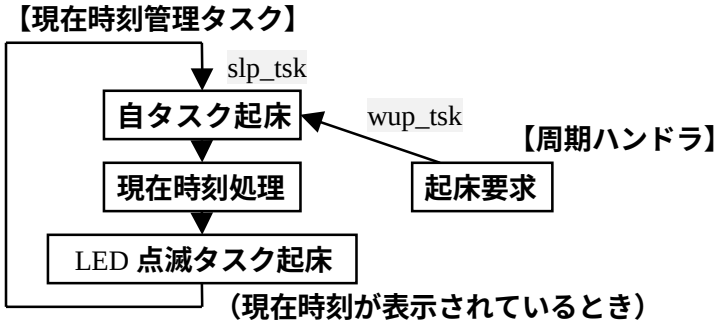

図 5.3-5　現在時刻管理の流れ

5.3.3.2　　タスクと関数

図 5.3-5 の「現在時刻処理」部分には

- ダイナミック・ドライブ・タスクへ現在値を送信
- 時を刻む（桁繰り上がり処理を含む）

が含まれています。つまり時を刻み、現在時刻をダイナミック・ドライブ・タスクへ送信する処理です。現在時刻を表示するモードでないときは、時を刻む処理のみを行います。

次にこの「現在時刻処理」の記述の仕方についてです。そのまま個々のタスク内に記述しても構いませんが、これを一つの関数にしてまとめてしまい、タスクから関数コールする形を採ると、プログラムとしてより見やすくなります。またこのように関数を作成すると、他のタスクからも使用することができます。複数タスクが同じ内容の処理を行う場合には、コードサイズの縮小にもつながり、効率の良いアプリケーションの作成ができます。

例えば、今回のアプリケーションで多用されているメールボックスを使用したメッセージのやり取りがあります。「5.2.2.3 メールボックス」で説明した標準的なメッセージの送受信では、「メモリ・ブロックの取得」「メッセージ書き込み」「メモリ・ブロック解放」を繰り返します。これを毎回記述すると、プログラムが見にくくなるばかりではなく、コードサイズも無駄に大きくなってしまいます。そこで、その 3 つの処理を一つの関数にまとめてしまい、関数の引数でアドレス等の調整を行うようにすると、この問題が解決します。この関数の詳しい仕様に

ついては「第 6 章 コーディング・リスト」をご覧ください。

図 5.3-6 は C 言語で記述したときの周期ハンドラと現在時刻管理タスクの処理概要です。

```
周期ハンドラ()          /* 現在時刻管理タスク起床用   */
{
        wup_tsk (現在時刻管理タスク);      /* 現在時刻管理タスクを起床 */
        return;                          /* 周期ハンドラから復帰       */
}
```

```
現在時刻管理タスク()
{
    for ( ; ; ) {
        slp_tsk ();          /* 自タスク起床待ち   */
        if (現在時刻非表示モードの時) {
            現在時刻進行処理;
        } else if (現在時刻表示モードの時) {
            現在時刻進行処理 + 現在時刻表示;
            秒刻み LED タスク起床;
        }
    }
}
```

図 5.3-6　周期ハンドラと現在時刻管理タスクの処理概要

5.3.3.3　　現在時刻を設定する処理

次に現在時刻の設定を行う処理についてです。設定は 7 セグメント LED の各桁に対応させたタクト・スイッチの割り込みによって行います。（タクト・スイッチの割り込み処理内容については、「5.3.9 タクト・スイッチ / タッチセンサ割り込み管理」をご覧ください）

設定するときは、まず現在時刻を進める処理を中断します。つまり現在時刻管理タスク起床している、周期ハンドラを中断することになります。設定中は特に時間を進めている必要がないのと、新たに設定された時刻から再開するとき、きちんとしたタイミングで行えるようにするためです。周期ハンドラの中断・再開

は act_cyc システム・コールによって行います。

中断した後、各桁のタクト・スイッチが押される度にその桁の数字をインクリメントしていきます。現在時刻設定中のときはこの処理を繰り返します。時間のカウントが再開されたとき、設定値を現在時刻管理タスクに報告し、周期ハンドラを再開します。設定値の報告にはメールボックスを使用します。

この現在時刻を設定する処理は、タクト・スイッチによる割り込み処理にて行われています。

5.3.4　タイマ管理

タイマとは、目的の時間になった時にある動作を開始する、またはその旨を知らせるものです。今回のアプリケーションでは、その目的の時間として絶対時間を設定するのではなく、経過時間を設定するものとします。また設定された時間が経過したときは圧電ブザーを鳴らして知らせるようにします。

5.3.4.1　タイマをカウントダウンする処理

タイマをカウントダウンする処理は、現在時刻の処理とほとんど変わりません。基本的にカウントアップとダウンの違いです。つまりタイマの値を管理しているタスクを、周期ハンドラから起床する処理になります。設定値から 1 秒ごとにカウントダウンしていき、カウントが 0 になると、ブザー管理タスクを起床し、ブザーを鳴らします（ブザー管理タスクに関しては「5.3.7 ブザー管理」をご参照ください)。図 5.3-7 はタイマ管理タスクの処理概要です。カウントが 0 になった後タイマをリセットし、再び値を設定して動作できるようにします。

なおタイマのカウントダウン中は、他の表示モードに移っても動作を続け、再びタイマ表示モードになったときは、その時点の値を表示するようにします。

5.3.4.2　タイマをセットする処理

タイマをセットする処理は、現在時刻を修正する処理とほとんど変わりません。ただしセットできる値の限界値が違うことに注意が必要です。00:01～99:59 までセットできる仕様とします。

```
周期ハンドラ()          /* タイマー管理タスク起床用   */
{
        wup_tsk (タイマー管理タスク);      /* タイマー管理タスク起床   */
        return;                          /* 周期ハンドラから復帰      */
}
```

```
タイマー管理タスク()
{
    for ( ; ; ) {
        slp_tsk ();          /* 自タスク起床待ち   */
        if (タイマー非表示モードの時) {
            タイマーカウントダウン処理のみ;
        } else if (タイマー表示モードの時) {
            タイマーカウントダウン処理 + 現在値表示;
            秒刻み LED タスク起床;
        }
     }
}
```

図 5.3-7　**タイマ管理タスクの処理概要**

5.3.5　ストップウォッチ管理

ストップウォッチは、スタートボタンが押された時からストップボタンが押されるまでの時間を測定するものです。機能として

- スタート
- ストップ
- ラップ表示
- リセット

を持つストップウォッチを実現します。

5.3.5.1　　ストップウォッチを進める処理

ストップウォッチのカウント処理は、現在時刻を進める処理とほとんど変わりません。ただし限界値が違うことに注意が必要です。仕様として 00:00 から 99:59 まで測定できるようにします。なお 99:59 を超えると 00:00 に戻してカウントを続けます。

今回のアプリケーションでは、カウント単位を 2 種類選択できるようにしています。1 秒ごとにカウントする「1/1 秒モード」と、1/10 秒ごとにカウントする「1/10 秒モード」です。ストップウォッチを起動する前に、まずどちらの単位で測定するかを決定します。つまり周期ハンドラを 1 秒ごとに起動するか、1/10 秒ごとに起動するかを選択し、選択された周期ハンドラから「ストップウォッチ管理タスク」を起床することになります。しかし RX850 では周期ハンドラの周期起動間隔を動的に変更することはできません。ですから周期起動間隔の違う周期ハンドラを別々に生成し、選択されたモードによって起動する周期ハンドラを変更します。

この時、どちらのモードでも値をカウントする方法は変わりません。つまり周期ハンドラのプログラム（コード）を共有することが可能です。このようなコードの共有についてはコンフィギュレーション・ファイルで指定します。それぞれ周期ハンドラ定義の記述にある「周期ハンドラの先頭アドレス」を、同一のアドレスにしておくことによって可能になります。図 5.3-8 はコンフィギュレーション・ファイルのストップウォッチ用周期ハンドラの定義部分です。上の記述が 1/1 秒モードの周期ハンドラ、下の記述が 1/10 秒モードの周期ハンドラの記述です。

```
cyclic    CYC_WATCH      StopWatchDrive    TCY_OFF    1000

cyclic    CYC_WATCH10    StopWatchDrive    TCY_OFF    100
```

図 5.3-8　ストップウォッチ用周期ハンドラの定義

5.3.5.2　　スタート / ストップ / ラップ表示

ストップウォッチ管理タスクで処理する内容は、次のようになります。

- ストップウォッチのカウント
- ラップ表示
- 秒刻み LED の起床

ラップ表示とは、ストップウォッチを止めることなく、その時の値を一時的に表示するもので、通過タイム等を見るときなどに使用されます。

このラップ表示は、タクト・スイッチが押されることによって行われます。スイッチが押されると、表示はその瞬間の値で停止しますが、バックグラウンドではストップウォッチのカウントが続けられています。もう一度スイッチが押されると、続けられているカウントの表示を再開します。プログラムでは、ストップウォッチ管理タスクからカウント処理を行う関数をコールするときに、引数によって処理を分けています。図 5.3-9 が周期ハンドラとストップウォッチ管理タスクの処理概要です。

なおストップウォッチのスタート・ストップは、周期ハンドラを起動・中断することによって実現していますが、この処理は別の関数で行っています。

また秒刻み LED の起床ですが、1/10 秒モードのときは 10 回タスクが起動されるごとに 1 回起床するようにしておきます。

```
周期ハンドラ()          /* ストップウォッチ管理タスク起床用   */
{
        wup_tsk (ストップウォッチ管理タスク);
                                    /* ストップウォッチ管理タスク起床   */
        return;                     /* 周期ハンドラから復帰       */
}
```

```
ストップウォッチ管理タスク()
{
    for ( ; ; ) {
        slp_tsk ();         /* 自タスク起床待ち   */
        if (ストップウォッチ非表示モードの時) {
            ストップウォッチカウント処理;
        } else if (ストップウォッチ表示モードの時) {
            if (ラップ表示モードではない時) {
                ストップウォッチカウント処理 + 現在値表示;
            } else if (ラップ表示モードの時) {
                ストップウォッチカウント処理;
                ラップ表示になったときの値を表示;
            }
            秒刻み LED タスク起床;
        }
    }
}
```

図 5.3-9　**ストップウォッチ管理タスクの処理概要**

5.3.6　スロットマシン管理

スロットマシンは、7 セグメント LED の下三桁を使用します。スロットマシンのスタートボタンを押すと、下三桁すべてを 0 から 9 の順番に、かつ高速に表示していき、各桁のストップボタンを押すとその瞬間の値で停止します。三桁すべて同じ値で停止したときは圧電ブザーを鳴らします。

5.3.6.1　スロットマシンのスタート・ストップ

この処理も周期ハンドラと、そこから起床されるタスクによって実現します。スタート・ストップは割り込みによって行われます。

スタートボタンが押されると、スロット管理タスクを起床する周期ハンドラを起動し、三桁すべてを高速表示します（いわゆるスロット回転）。周期ハンドラの起動周期間隔は 70ms です。この値をさらに小さくすると、人間の目には速すぎて一つ一つの数字を見分けるのが難しくなります。

各桁の停止はどの桁から行っても構いません。各桁すべてが停止したときは、周期ハンドラを中断します。これらを繰り返すことによって、スロットマシン動作を実現しています。

なおスロットマシン動作中に、他の表示モードに移ったときはその動作を続け、再びスロットマシン表示モードになったときは、その時点の値が表示されるようにします。図 5.3-10 はスロットマシン管理タスクの処理概要です。

```
周期ハンドラ()          /* スロットマシン管理タスク起床用   */
{
        wup_tsk (スロットマシン管理タスク);
                                        /* スロットマシン管理タスク起床   */
        return;                         /* 周期ハンドラから復帰       */
}
```

```
スロットマシン管理タスク()
{
    for ( ; ; ) {
        slp_tsk ();         /* 自タスク起床待ち   */
        if (スロットマシン非表示モードの時) {
            スロットマシン回転処理のみ;
        } else if (スロットマシン表示モードの時) {
            スロットマシン回転処理 + 現在値表示;
        }
     }
}
```

図 5.3-10　**スロットマシン管理タスクの処理概要**

5.3.7　ブザー管理

ブザーを鳴らす処理です。今回使用する圧電ブザーは、「4.4.5 圧電ブザー」で説明したように回路に信号を送るとブザーが鳴る仕組みになっています。ですからブザーを鳴らす処理を一つのタスクとして扱い、他タスクからブザー管理タスクを起床することによって動作させます。タスクの処理としては「5.3.2 秒刻み LED の点滅」で説明した LED 点滅タスクと同じになります。

今回のアプリケーションでは 1 回の起床命令につき、4 回ブザーを鳴らす処理になっています。図 5.3-11 がブザー管理タスクの処理概要です。ブザーとブザーの間隔は 2 つ目の dly_tsk で調整します。

```
ブザー管理タスク()
{
    for ( ; ; ) {
        slp_tsk ();          /*  自タスク起床待ち    */
        for (4 回ループ) {
            ブザーON;
            dly_tsk (ウェイト);
            ブザーOFF;
            dly_tsk (ブザー間隔);
        }
    }
}
```

図 5.3-11　ブザー管理タスクの処理概要

5.3.8　ステッピング・モータ管理

ステッピング・モータは、その駆動回路にパルス信号 ON、OFF を送ると、ワンステップ駆動する仕組みになっています。そのステップ角度はステッピング・モータによって異なります。例えばステップ数 192 のステッピングモータを使用すると、ワンステップが 1.875 ° になります。このように正確な駆動ができるため、プリンタの紙送りや工作機などによく使用されます。

このステッピング・モータをタスクで駆動する方法について説明します。

5.3.8.1　　ステッピング・モータの回転

ステッピング・モータを駆動する際の処理の流れは、図 5.3-12 のようになります。

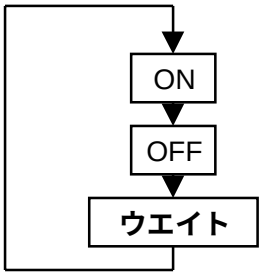

図 5.3-12　ステッピング・モータ回転タスクの流れ図

この図で一つのループがワンステップ駆動となり、無限ループにすることによりステッピング・モータは回転します。

図中にある「ウエイト数」を変えると回転スピードを調整することができます。つまりウエイトを大きくすると遅くなり、ウエイトを小さくすると速くなります。このウエイトを作り出すのに dly_tsk システム・コールを使用します。ですから dly_tsk(1)と指定したときが最速となります。つまりシステム・クロックの間隔（インターバル）に依存することになります。

またステップ数を指定して回転させたいときは、ループ数を指定すればよいことになります。

今回のアプリケーションでは、気温が上昇すると回転数を上げるようになっています。つまり気温の値を受け取り、これを dly_tsk システム・コールの引数に反映してやることによって実現します。図 5.3-13 がステッピング・モータ管理タスクの処理概要です。

```
ステッピング・モータ管理タスク()
{
    for ( ; ; ) {
        pol_flg ();          /* 気温データが到着しているかを確認    */
        if (気温データが到着しているとき) {
            回転スピードを決定;
        }
        dly_tsk (回転スピード);
        ワンステップ駆動;
     }
}
```

図 5.3-13　**ステッピング・モータ管理タスクの処理概要**

ここで気温データを取得するために pol_flg **システム・コールを使用しています。これは**5.2.2.1**であげた例の** 2 **番目の方法を用いてデータ通信を行っています。このイベント・フラグにセットするタスクは、気温・湿度管理タスクです。もちろんメールボックスを使用しても実現できますが、データ幅や処理の手間を考え、イベント・フラグを使用しています。このアプリケーションでは** 5 **秒間隔で気温・湿度を計測していますので、速度変化も** 5 **秒間隔となります。気温・湿度計測に関する詳しい内容は「**5.3.10　**温度センサ** / **湿度センサ」をご覧ください。**

5.3.9　タクト・スイッチ / タッチセンサ割り込み管理

これまでの説明にも何度か出てきた、タクト・スイッチ/タッチセンサによる割り込み処理について説明します。

タクト・スイッチを押す、またはタッチセンサに触れると、V853 に外部割り込みとして入力され、それぞれの割り込み処理を実行します。今回のアプリケーションでは 8 つのタクト・スイッチと 1 つのタッチセンサを扱っているので、9 つの外部割り込み処理が存在します。

5.3.9.1　各割り込み処理の内容

今回のアプリケーションで使用する外部割り込み処理は、すべて間接起動割り込みハンドラを使用します。ですから割り込みハンドラの登録はすべてコンフィギュレーション・ファイルにて行います。なお直接起動割り込みハンドラを使用する場合は「5.2.4.2 直接起動割り込みハンドラ」を参考にしてください。

次にタクト・スイッチ 1～8、およびタッチ・センサの割り込み処理の内容について説明します。

1. タクト・スイッチ – 1
   - 現在時刻表示モードで、かつ現在時刻設定モードのとき
     → 7 セグメント LED の下一桁目の値を設定する
   - タイマー表示モードで、かつタイマー設定モードのとき
     → 7 セグメント LED の下一桁目の値を設定する
   - ストップウォッチ表示モードのとき
     → ストップウォッチのスタート・ストップ
   - スロットマシン表示モードで、かつスロット回転中のとき
     → 7 セグメント LED の下一桁目の値を停止する

2. タクト・スイッチ – 2
   - 現在時刻表示モードで、かつ現在時刻設定モードのとき
     → 7 セグメント LED の下二桁目の値を設定する
   - タイマー表示モードで、かつタイマー設定モードのとき
     → 7 セグメント LED の下二桁目の値を設定する
   - ストップウォッチ表示モードで、かつ動作中のとき

→　ストップウォッチのラップ表示 ON / OFF

- スロットマシン表示モードで、かつスロット回転中のとき

→　7 セグメント LED の下二桁目の値を停止する

3. タクト・スイッチ – 3

- 現在時刻表示モードで、かつ現在時刻設定モードのとき

→　7 セグメント LED の下三桁目の値を設定する

- タイマー表示モードで、かつタイマー設定モードのとき

→　7 セグメント LED の下三桁目の値を設定する

- ストップウォッチ表示モードのとき

→　ストップウォッチのリセット

- スロットマシン表示モードで、かつスロット回転中のとき

→　7 セグメント LED の下三桁目の値を停止する

4. タクト・スイッチ – 4

- 現在時刻表示モードで、かつ現在時刻設定モードのとき

→　7 セグメント LED の下四桁目の値を設定する

- タイマー表示モードで、かつタイマー設定モードのとき

→　7 セグメント LED の下四桁目の値を設定する

- スロットマシン表示モードのとき

→　スロットの回転を開始する

5. タクト・スイッチ – 5

各モードの切り替えに使用。スイッチを押すたびに、図 5.3-14 のようにモードが切り替わります

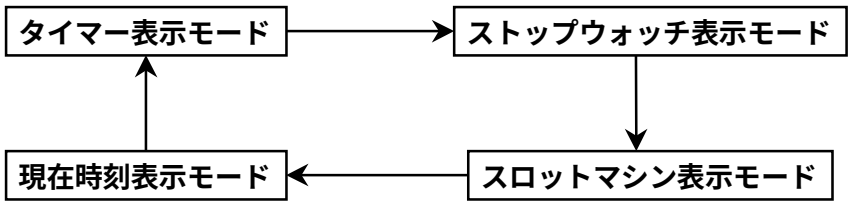

図 5.3-14　　モード切り替わりの順番

6. **タクト・スイッチ** – 6
   - **現在時刻表示モードのとき**
     → **現在時刻設定モードの** ON/OFF
   - **タイマー表示モードのとき**
     → **タイマー動作開始** / **停止**
   - **ストップウォッチ表示モードのとき**
     → **動作モード切り替え（**1/1 **秒モード** or 1/10 **秒モード）**

7. **タクト・スイッチ** – 7
   LCD **に気温・湿度を表示する。**LED **に気温・湿度を表示中のときは、現在の** 7 **セグメント** LED **モードを表示する。**

8. **タクト・スイッチ** – 8
   LCD **に記憶されているメモを表示し、さらにメモ入力を可能にする。メモがすでに表示されているときは、現在の** 7 **セグメント** LED **モードを表示する。**

9. **タッチセンサ**
   - **タイマー表示モードで、かつタイマが終了しているとき**
     → **タイマをリセットする**
   - **スロットマシン表示モードで、かつスロットが停止しているとき**
     → **スロットの値を** 777 **にリセットする**
   - LCD **にメモが表示されているとき**
     → **メモの内容を消去する**

5.3.9.2　　割り込みハンドラと割り込みタスク

「5.3.9.1 **各割り込み処理の内容」にあげた各割り込み処理を、それぞれ対応した割り込みハンドラに記述していくことになります。**

処理内容を見ると、タクト・スイッチ 1～6 **の割り込みが入った時、そのときの** 7 **セグメント** LED **の表示モードを知る必要があります。そのためにグローバル変数を** 1 **つ用意し、現在のモードを保持するフラグとして使用します。各割り込みハンドラ内での分岐処理はこのフラグの値を元に行います。**

各割り込み処理の内容は、さほど長くなるものはないので、割り込みハンドラにすべての処理を記述しても、他の処理が遅延されるなどの影響はありません。しかしゆるやかな処理でも構わないものが存在しますので、それらは割り込みタ

スクを使用して処理します。割り込みタスクに記述する内容は次にあげるものです。

- 現在時刻設定処理
- タイマー設定処理、動作開始処理、動作中断処理
- スロットマシン動作開始処理

5.3.9.3　　割り込み制御

ある状態のときには入っては行けない割り込みというものが一般に存在します。今回の例をあげると、7 セグメント LED に現在時刻が表示されているとき、タクト・スイッチ 1～4 が押されて表示が変わってしまっては困ります。各割り込みハンドラ内で、各モードごとに条件分岐処理を行って回避することもできますが、必要のない処理まで行うことになり、プログラムも複雑になります。ですから現在時刻表示中は、タクト・スイッチ 1～4 の割り込みが入っても何もせずにハンドラから復帰する処理を行った方が効率よくなります。このような割り込み制御について説明します。

割り込みを制御する方法には 2 つあります。1 つはハードウエアで制御する方法、もう 1 つはソフトウエアで制御する方法です。今回はプログラム中にその仕組みを入れる、ソフトウエアで制御します。

割り込みを制御するグローバル変数を用意し、これを制御フラグとしてタクト・スイッチ割り込み入力を制御します。図 5.3-14 はこの割り込み制御フラグのイメージです。

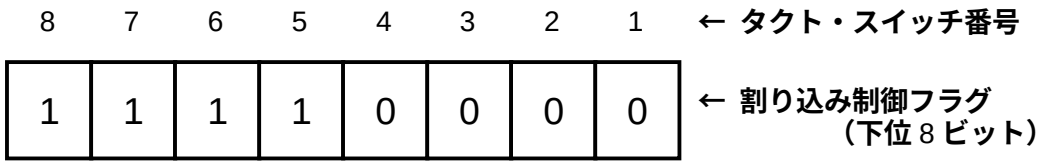

図 5.3-14　割り込み制御フラグ

タクト・スイッチ割り込みは 8 つあるので、その変数の下位 8 ビットを用います。各ビットをそれぞれタクト・スイッチ番号に割り当て、割り込みを許可するときはそのビットに 1 を、禁止するときは 0 を立てます。図 5.3-14 の例であれば、タクト・スイッチ 1～4 は禁止、タクト・スイッチ 5～8 は許可されている状態です。割り込みが入ると、必ずその割り込みに対応した割り込みハンドラに処理が

移るので、割り込みハンドラの最初で、割り込み状態を参照するようにします。もし、割り込みが許可されていれば処理を続行し、復帰直前に割り込み制御フラグを更新します。逆に禁止されていれば、すぐに割り込みハンドラから復帰するように記述しておきます。図 5.3-15～図 5.3-24 は C 言語で記述したときのタクト・スイッチ割り込みハンドラ、タッチセンサ割り込みハンドラ、および割り込みタスクの処理概要です。図中には「割り込み制御フラグを更新する」処理が抜けていますが、随時更新していますので省略してあります。詳しくは「第 6 章 コーディング・リスト」をご覧ください。

なおタッチ・センサの割り込み処理は、さほど場合分けが必要ないため、タクト・スイッチの場合のような割り込み制御は行っていません。

ハードウエアで割り込み制御する場合は、V853 の割り込み制御レジスタの変更によって行います。詳しくはユーザーズ・マニュアル　ハードウェア編を御参照ください。

```
タクトスイッチ-1 ()
{
    if (タクト・スイッチ-1 割り込みが許可されているとき) {
        switch (現在表示されている 7 セグメント LED 情報) {
            case  現在時刻表示 ：
                  割り込みタスク起床;
                  break;
            case  タイマー表示 ：
                  割り込みタスク起床;
                  break;
            case  ストップウォッチ表示 ：
                  if (ストップウォッチ停止中のとき) {
                        ストップウォッチ動作開始;
                  } else if (ストップウォッチ動作中のとき) {
                        ストップウォッチ動作終了;
                  }
                  break;
            case  スロット表示 ：
                  下一桁目停止フラグを立てる;
                  break;
        }
    }
    return (0xff);
}
```

図 5.3-15　**タクト・スイッチ**-1 **割り込みハンドラの処理概要**

```
タクトスイッチ-2 ()
{
    if (タクト・スイッチ-2 割り込みが許可されているとき) {
        switch (現在表示されている 7 セグメント LED 情報) {
            case  現在時刻表示 ：
                  割り込みタスク起床;
                  break;
            case  タイマー表示 ：
                  割り込みタスク起床;
                  break;
            case  ストップウォッチ表示 ：
                  if (ラップタイム表示モードではないとき) {
                        ラップタイム表示モード ON;
                  } else if (ストップウォッチ動作中のとき) {
                        ラップタイム表示モード OFF;
                  }
                  break;
            case  スロット表示 ：
                  下二桁目停止フラグを立てる;
                  break;
        }
    }
    return (0xff);
}
```

図 5.3-16　**タクト・スイッチ**-2 **割り込みハンドラの処理概要**

```
タクトスイッチ-3 ()
{
    if (タクト・スイッチ-3 割り込みが許可されているとき) {
        switch (現在表示されている 7 セグメント LED 情報) {
            case  現在時刻表示 ：
                  割り込みタスク起床;
                  break;
            case  タイマー表示 ：
                  割り込みタスク起床;
                  break;
            case  ストップウォッチ表示 ：
                  ストップウォッチリセット;
                  break;
            case  スロット表示 ：
                  下三桁目停止フラグを立てる;
                  break;
        }
    }
    return (0xff);
}
```

図 5.3-17　**タクト・スイッチ**-3 **割り込みハンドラの処理概要**

```
タクトスイッチ-4 ()
{
    if (タクト・スイッチ-4 割り込みが許可されているとき) {
        switch (現在表示されている 7 セグメント LED 情報) {
            case  現在時刻表示 ：
                  割り込みタスク起床;
                  break;
            case  タイマー表示 ：
                  割り込みタスク起床;
                  break;
            case  ストップウォッチ表示 ：
                  break;
            case  スロット表示 ：
                  下四桁目停止フラグを立てる;
                  break;
        }
    }
    return (0xff);
}
```

図 5.3-18　タクト・スイッチ-4 割り込みハンドラの処理概要

```
タクトスイッチ-5 ()
{
    if (タクト・スイッチ-5 割り込みが許可されているとき) {
        if (現在 LCD に気温・湿度が表示されているとき) {
            自動表示タスクを起床する;
        switch (現在表示されている 7 セグメント LED 情報) {
            case  現在時刻表示 ：
                  タイマー表示モードに変更;
                  --現在のタイマー動作別に処理—
                  LCD 表示タスク起床;注
                  break;
            case  タイマー表示 ：
                  ストップウォッチ表示モードに変更;
                  --現在のストップウォッチ動作別に処理--
                  LCD 表示タスク起床;
                  break;
            case  ストップウォッチ表示 ：
                  スロットマシン表示モードに変更;
                  --現在のストップウォッチ動作別に処理--
                  LCD 表示タスク起床;
                  break;
            case  スロット表示 ：
                  現在時刻表示モードに変更;
                  --現在のストップウォッチ動作別に処理--
                  LCD 表示タスク起床;
                  break;
        }
    }
    return (0xff);
}
```

図 5.3-19　**タクト・スイッチ**-5 **割り込みハンドラの処理概要**

注 ：　LCD **表示タスクとは、**7 **セグメント** LED **の情報を** LCD **に表示するタスクです。詳しくは「**5.3.11.3 **メッセージ内容の扱い」をご覧ください。**

```
タクトスイッチ-6 ()
{
    if (タクト・スイッチ-6 割り込みが許可されているとき) {
        if (現在 LCD に気温・湿度が表示されているとき) {
            自動表示タスクを起床する;
        switch (現在表示されている 7 セグメント LED 情報) {
            case   現在時刻表示 ：
                if (現在時刻設定モードではないとき) {
                    --設定モードへ移行--
                    LCD 表示タスク起床;
                } else if (現在時刻設定モードのとき) {
                    --設定モード解除--
                    LCD 表示タスク起床;
                }
                break;
            case   タイマー表示 ：
                if (タイマー停止中のとき) {
                    割り込みタスク起床;
                    LCD 表示タスク起床;
                } else if (タイマー動作中のとき) {
                    割り込みタスク起床;
                    LCD 表示タスク起床;
                }
                break;
            case   ストップウォッチ表示 ：
                if (1/1 秒モードのとき) {
                    1/10 秒モードに変更;
                } else if (1/10 秒モードのとき) {
                    1/1 秒モードに変更;
                }
                break;
            ---------- つづく ----------
```

```
                ------------------------- つづき -------------------------
                    case   スロット表示 ：
                           break;
                }
        }
        return (0xff);
}
```

図 5.3-20　タクト・スイッチ-6 割り込みハンドラの処理概要

```
タクトスイッチ-7 ()
{
    if (タクト・スイッチ-7 の割り込みが許可されているとき) {
        switch (気温・湿度表示モードフラグ) {
            case   表示モード OFF ：
                   表示フラグを ON にする;
                   気温・湿度定期測定ハンドラを初期化;
                   気温・湿度測定タスク起床
                   break;
            case   表示モード ON ：
                   表示フラグを OFF にする;
                   LCD 表示タスク起床
                   気温・湿度表示自動切り換えタスク起床
                   break;
        }
    }
    return (0xff);
}
```

図 5.3-21　タクト・スイッチ-7 割り込みハンドラの処理概要

```
タクトスイッチ-8 ()
{
    if (タクト・スイッチ-8 の割り込みが許可されているとき) {
        switch (メモ表示フラグ) {
            case  表示 OFF ：
                  表示を ON にする;
                  気温・湿度表示自動切り換えタスクをサスペンド
                  LCD 表示タスク起床
                  シリアル受信許可
                  break;
            case  表示 ON ：
                  表示を OFF にする;
                  気温・湿度表示自動切り換えタスクのサスペンド解除
                  LCD 表示タスク起床
                  シリアル受信禁止
                  break;
        }
    }
    return (0xff);
}
```

図 5.3-22　**タクト・スイッチ**-8 **割り込みハンドラの処理概要**

```
タッチセンサ ()
{
    if (LCD にメモが表示されているとき) {
        メモクリア;
    } else if (LCD にメモが表示されていないとき) {
        switch (現在表示されているの 7 セグメント LED 情報) {
            case  現在時刻表示 ：
                  break;
            case  タイマー表示 ：
                  if (タイマーが終了しているとき) {
                      タイマーカウントクリア;
                      LCD 表示タスク起床;
                  } else if (タイマーが中断しているとき) {
                      タイマー初期化;
                  }
                  break;
            case  ストップウォッチ表示 ：
                  break;
            case  スロット表示 ：
                  スロットの値を 777 で初期化;
                  break;
        }
    }
    return (0xff);
}
```

図 5.3-23　**タッチセンサ割り込みハンドラの処理概要**

```
割り込み制御タスク ()
{
    switch (現在表示されている 7 セグメント LED 情報) {
        case  現在時刻表示 :
              break;
        case  タイマー表示 :
              if (タイマーが停止しているとき) {
                  タイマー値設定;
              } else if (タイマーが開始されたとき) {
                  タイマー初期化;
              } else if (タイマーが中断されたとき) {
                  タイマー停止;
              }
              break;
        case  ストップウォッチ表示 :
              break;
        case  スロット表示 :
              スロットの値を 777 に初期化;
              break;
    }
    return (0xff);
}
```

図 5.3-24　**割り込みタスクの処理概要**

5.3.10　温度センサ / 湿度センサ

温度センサを用いた気温の測定、湿度センサを用いた湿度の測定の方法を説明します。

5.3.10.1　　A/D 変換

センサから出力される信号はアナログ信号です。このアナログの測定値を V853 で取り込むには、信号のデジタル化、つまり A/D 変換が必要になります。今回は V853 に内蔵されている A/D コンバータを使用して、センサ出力結果を取得します。

V853 には 10 ビット分解能 A/D コンバータを 8ch 持っています。ですから 8 つのアナログ入力を扱うことが可能です。今回は温度センサ、湿度センサの 2 つのアナログ入力があるので 2 ch を使用します。それぞれ ANI0、ANI1 端子に接続します。（端子名などハードウエアに関する詳しい内容については、「ユーザーズ・マニュアル　ハードウェア編」をご覧ください）

A/D 変換の流れは、次のようになります。

まず A/D コンバート開始命令を出します。これは A/D コンバータ・モード・レジスタの CE ビットに 1 を書き込むことによって行われます。すると ANI0、ANI1 に入力されたアナログ信号の A/D 変換が順番に行われ、終了すると内部割り込み INTAD が発生します。A/D 変換の結果は ADCR レジスタに格納されていますので、ここから値を読み出すことによって、結果を得ることができます。

この流れを、RX850 を用いて実現すると次のようになります。

まず A/D 変換の開始、結果の取得を行う処理を一つのタスクとします。また 5 秒間隔で測定を繰り返すようにするため、5 秒周期の周期ハンドラを使用し、そのタスクを起動します。また A/D 変換終了時の INTAD 割り込みの発生を知らせる割り込みハンドラを用意します。図 5.3-25 は A/D 変換処理の流れ図です。

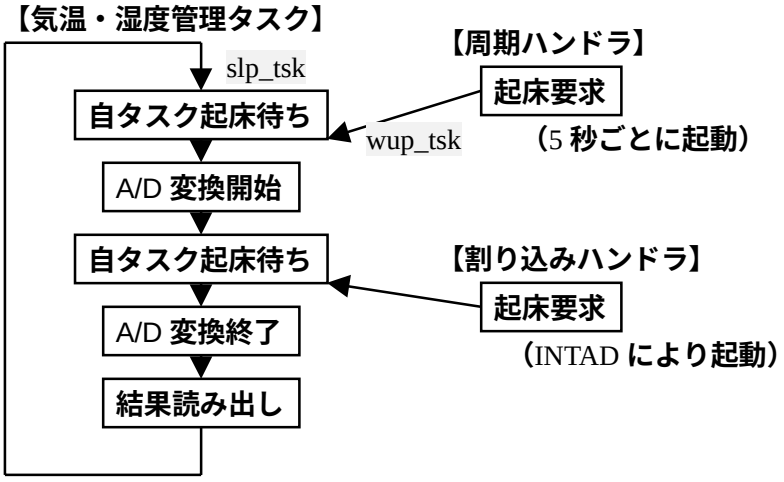

図 5.3-25　A/D コンバート処理の流れ図

今回のアプリケーションには、測定結果をステッピング・モータ管理タスク、LCD 管理タスクへ送信する処理があります。これらの処理は上図の「結果読み出し」処理のあとに続きます。

INTAD 割り込みによる割り込みハンドラは、気温・湿度管理タスクを起床して復帰する処理を行いますが、このような時は wup_tsk システム・コールを使用して起床させるのではなく、return(気温・湿度管理タスクの ID)を使用した方が効率がよくなります。

5.3.10.2　　センサの値

センサから出力された値は、そのセンサの特性表や実測値と比較して調整します。同じセンサでも特性は個々に違っていますので注意が必要です。また限界値を丸める処理も行っておきます。出力値をハードウエアで調整する方法と、ソフトウエアで調整する方法がありますが、今回はすべてソフトウエアで行っています。ですから A/D 変換で得られた結果を加工する関数を用意し、そこで調整するようにします。図 5.3-25 は、気温・湿度管理タスク、周期ハンドラ、INTAD 割り込みハンドラの処理概要です。

```
周期ハンドラ()          /* 気温・湿度管理タスク起床用  */
{
        wup_tsk (気温・湿度管理タスク);      /* 現在時刻管理タスクを起床 */
}
```

```
割り込みハンドラ()      /* 気温・湿度管理タスク起床用  */
{
        return (気温・湿度管理タスクの ID);   /* 現在時刻管理タスクを起床 */
}
```

```
気温・湿度管理タスク()
{
    for ( ; ; ) {
        slp_tsk ();                    /* 自タスク起床待ち  */
        A/D 変換開始
        slp_tsk ();                    /* 自タスク起床待ち (A/D 変換終了待ち)*/
        A/D 変換終了;
        A/D 変換結果読み出し & 値更新;
        ステッピング・モータ管理タスクに気温の値を報告;
        LCD 管理タスク起床 & 測定値報告;
     }
}
```

図 5.3-24　**気温・湿度管理タスクの処理概要**

5.3.11　LCD 表示

今回のアプリケーションでは、LCD に次の項目を表示します。

- 7 セグメント LED モード

  現在 7 セグメント LED に表示されているものを知らせます。表示する内容は　以下の通りで、表示モードが変わると LCD の表示も変えます。

  - 現在時刻モード
  - タイマーモード
  - ストップウォッチモード（1/1 or 1/10 モード）
  - スロットマシンモード

- 気温・湿度

  現在の気温・湿度を知らせます。

- メモ

  シリアル通信で受信した文字を表示します。

「7 セグメント LED モード」が 10 秒間表示され続けた後、またはタクト・スイッチ-7 が押されたとき、「気温・湿度」の表示に変わります。逆に「気温・湿度」が表示されているときにタクト・スイッチ-7 が押されると、「7 セグメント LED モード」の表示に切り替わります。

「メモ」はその名の通り、一時的に残しておきたい文字を表示し、消去しない限り記憶します。LCD が他の表示のとき、タクト・スイッチ-8 が押されると「メモ表示モード」に切り替わり、逆にメモ表示モードのときは「7 セグメント LED モード」の表示に切り替わります。

5.3.11.1　　LCD へのデータ送信

今回使用する LCD は、LCD ユニットの持つデータ線に JIS コードを入力すると、それに対応した文字が表示される仕様となっています。「4.4.3 LCD ユニット」であげた LCD ユニットとのポート接続図で、D4～D7 の 4 ポートに、8 ビットの JIS コードを 2 回に分けて送信します。なお LCD のカーソル移動などの制御も D4～D7 を使用します。

R/W は書き込み・読み込みの選択を制御するデータ線、E はデータ送信の許可・不許可を制御するデータ線、そして RS は送信データが JIS コードデータであるか、それともカーソル制御などの命令データであるかを指定するデータ線です。

LCD へのデータ送信手順は次のようになります。

- データ転送を許可状態にする
- RS を制御する（ON/OFF）
- 送信データをセット
- データ転送を不許可状態にする

この処理を繰り返すことによってデータを転送します。この処理を一つのタスク「LCD 表示タスク」で行います。

処理は上であげた順番で行っていくことになりますが、一般に LCD のような外部デバイスは処理スピードが V853 のような CPU に比べて遅く、その CPU に合わせて処理をすると外部デバイスはついていけなくなります。こういう場合は CPU 側でタイミングを調整しながら処理を行う、つまりウエイトを入る必要があります。上の送信手順で LCD の仕様にあったウエイトを入れながら、正しく動作できるように処理します。

タスクの中でウエイトを実現するには dly_tsk システム・コールを使用します。処理は図 5.3-25 のような流れになります。dly_tsk の間隔は、使用する LCD の仕様に合わせます。「送信データの受け取り」は「5.2.2.1 イベント・フラグ」で示した例の 2 番目の方法を用いています。

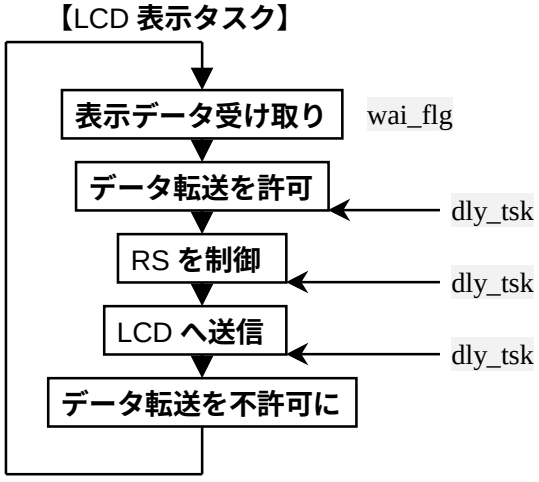

図 5.3-25　LCD 表示内容送信の流れ図

5.3.11.2　　送信データの管理

「LCD 表示タスク」は LCD ユニットにデータの転送を専門に行うタスクです。そのため図 5.3.25 にもある通り、表示するデータは他から受け取る必要があります。そこで表示データを「LCD 表示タスク」に送信するタスクとして「LCD 表示データ管理タスク」用意します。7 セグメント LED の表示情報、気温・湿度情報など LCD に表示するデータを管理し、1 文字ずつ「LCD 表示タスク」へ送信するタスクです。

このとき「LCD 表示タスク」と「LCD 表示データ管理タスク」は、お互いに同期を取る必要があります。それは「LCD 表示データ管理タスク」が、「LCD 表示タスク」にデータを 1 つずつ（8 ビットずつ）送信する必要があるため、そして 5.3.11.1でも述べたように、LCD にデータを送信する際はウエイトが必要なためです。

まず図 5.3-26 はこれらのタスク間で同期を「取らなかったとき」の流れ図です。

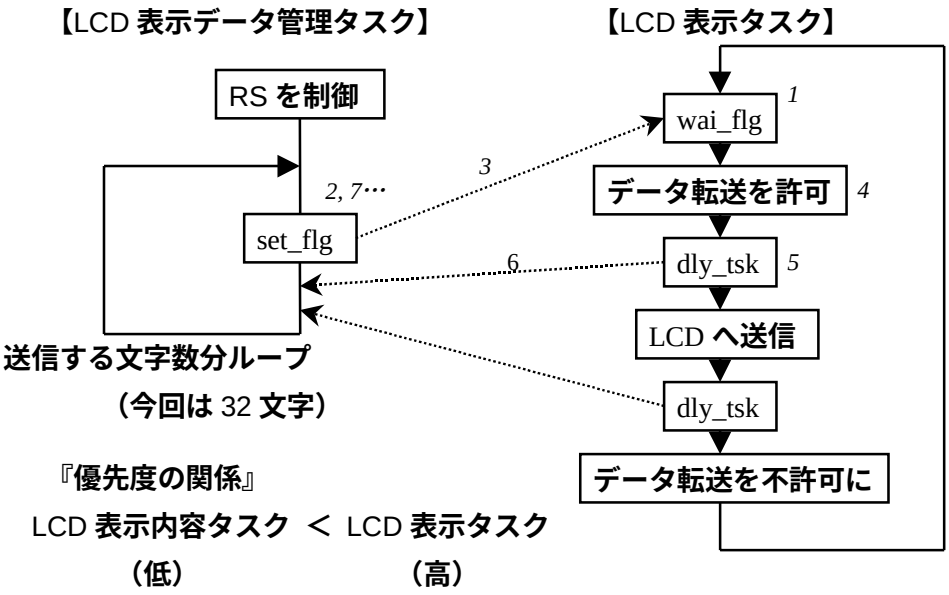

図 5.3-26　タスク間が非同期時の流れ図（番号は処理の順番）

最初に「LCD 表示タスク」はイベントフラグに値がセットされるのを待ちます。次に「LCD 表示データ管理タスク」に処理が移り、イベントフラグに値をセット

します。すると「LCD 表示タスク」は待ちが解除され、LCD ユニットにデータの転送を許可する命令を送信します。しかし次の「LCD へ送信する」前にウエイトを入れなければならないので再び待ち状態になります。すると「LCD 表示データ管理タスク」へ処理が移りますが、「LCD 表示タスク」の時限待ちが解除されるまで次々とデータをイベントフラグへセットしてしまうことになります[注]。これでは LCD ユニットにきちんとデータを送信することができません。

注　：　LCD 表示内容管理タスクがセットする回数は dly_tsk の時限待ち時間などによって異なります。

この問題を解決するため、お互いの処理を排他制御する機構「セマフォ」を使用します。

2 つのセマフォを用意し、「次データの送信を許可するセマフォ」と「LCD ユニットへのデータ送信を許可するセマフォ」を用意します。図 5.3-27 はセマフォを使って同期をとった場合の流れ図です。セマフォの資源を待つシステム・コールは wai_sem、資源を返却するシステム・コールは sig_sem です。

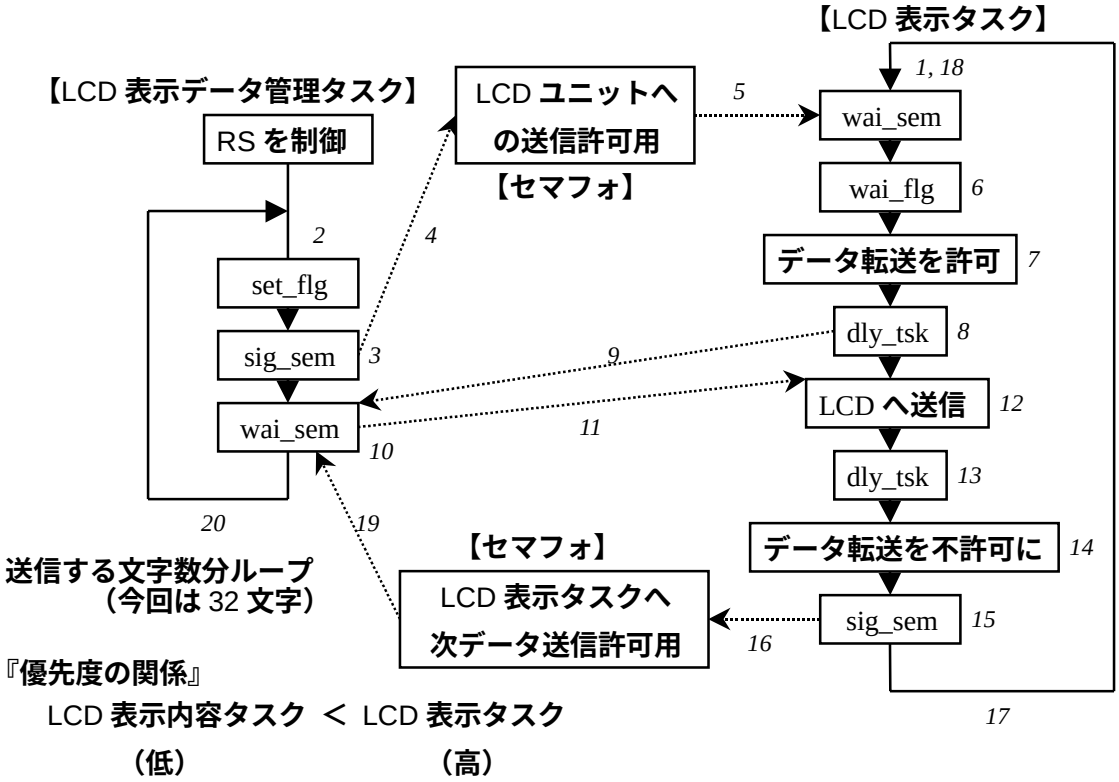

図 5.3-27　タスク間の同期を取った時の流れ図（番号は処理の順番）

まず「LCD 表示タスク」は、LCD ユニットへのデータ送信許可が出るのを待つため、セマフォの資源待ちをします。その後「LCD 表示データ管理タスク」がイベントフラグにデータをセットした後、LCD ユニットへのデータ送信を開始させるために「LCD ユニットへの送信許可用セマフォ」へ資源を返却します。すると「LCD 表示タスク」は待ちが解除されて実行が再開されます。LCD ユニットにデータ転送の許可命令を送信し、ウエイトを入れるために待ち状態になります。すると「LCD 表示データ管理タスク」へ処理が移り、次データの送信許可を待つため、「次データ送信許可用セマフォ」の資源待ちをします。その後「LCD 表示タスク」の時限待ちが解除されるとその続きが実行されます。LCD ユニットへ一文字分のデータ転送がすべて終わったとき、「次データ送信許可用セマフォ」に資源を返却します。その後「LCD 表示タスク」は、LCD ユニットへデータ転送の開始許可が出るまで「LCD 送信許可用セマフォ」の資源待ちに入ります。ここまでが 1 つのループで、再びデータ通信が開始されます。

こうすることで確実にデータのやりとりを実現します。

なお、このデータのやり取りにメールボックスを使用することも可能です。この場合は排他制御をする必要がありません。それは「LCD 表示データ管理タスク」が次々にメールボックスへメッセージ（データ）を送信すると、メッセージが順番にキューイングされて行くからです。ですから「LCD 表示タスク」が順番にデータを受け取ればきちんと通信することができます。しかしメッセージのキューイング作業による OS のオーバヘッドの増加や、メッセージ分のメモリを確保しながら通信しなければならないなど、上記の方法より多少効率が落ちてしまいます。

5.3.11.3　　メッセージ内容の扱い

一般の LCD ユニットは、8 ビットの JIS や ASCII コードだけを扱っているため、英数字、記号、カタカナの表示のみとなります。ですから、7 セグメント LED モードの「現在時刻モード」は「ゲンザイジコクモード」といった具合にすべてカタカナで、もしくは「CURRENT TIME MODE」のような英語で表示することになります。今回のアプリケーションではカタカナを使用します。

このような文字列をモードごとに定義しておき、メールボックスを使用して「LCD 表示データ管理タスク」へ送信します。つまり 7 セグメント LED のモードが変わるたびに、定義された文字列を送ることになります。このように文字列を

管理し、それを LCD 表示データ管理タスクへ送信する処理を一つのタスク「LCD 表示メッセージタスク」で扱います。

今回は LCD に表示する文字列を 16 文字×2 行の 32 文字分と決めておき、定義内容を要素数 32 の一次元配列として扱います。この内容の送信はメモリブロックを獲得しながら、定義文字列を書き込んで送信する方法もありますが、ここでは「5.2.2.3 メールボックス」でもあげた、「メッセージ領域を使い回わす方法」を用います。それは LCD に一度に表示される内容は一種類であるため、送信時にはその領域だけで十分だからです。図 5.3-28 はこの方法を用いたメッセージ通信のイメージです。

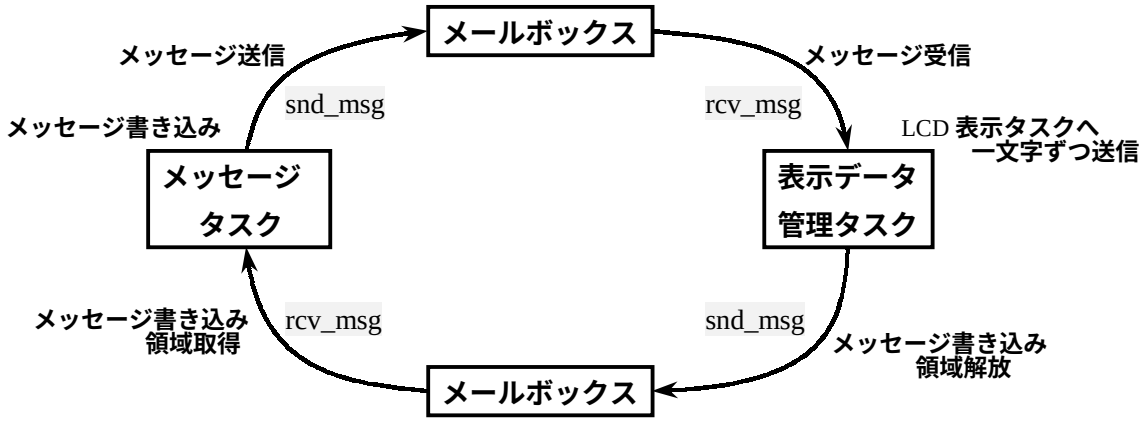

図 5.3-28　メッセージ領域自体をメールボックスでやり取りする方法

上の図で「メッセージタスク」に相当するのが、「LCD メッセージタスク」や「気温・湿度情報通知タスク」、「メモ管理タスク」です。これらのタスクは LCD にメッセージを表示する際に、書き込み領域の取得を試みます。取得出来たときは、その領域にメッセージを書き込み、LCD 表示タスクへメールボックスを介して送信します。

そして取得できなかったときの処理は二通り考えられます。書き込みを行うまで、つまり書き込み領域を取得できるまで待つ方法と、取得できなければ書き込みをあきらめる方法です。前者の方法であれば、必ず目的の文字列を LCD へ表示出来ます。しかし表示の切り替わりが頻繁に起こった場合、LCD ユニットとのウエイトの問題上 LCD 表示が重くなります。今回はこれを避けるために後者の方法を採りました。つまりメッセージ領域を取得するときは prcv_msg システム・コールを使用します。図 5.3-29 がメッセージタスク、図 5.3-30 が LCD 表示データ管理タスク、図 5.3-31 が LCD 表示タスクの処理概要です。

```
メッセージタスク ()
{
    for ( ; ; ) {
        slp_tsk ();
        prcv_msg ();      /* メッセージ領域獲得を試みる   */
        if (領域を獲得できたとき) {
            switch (現在表示されている 7 セグメント LED 情報) {
                case   現在時刻表示 ：
                       定義された文字列を読み出す;
                       break;
                case   タイマー表示 ：
                       定義された文字列を読み出す;
                       break;
                case   ストップウォッチ表示 ：
                       定義された文字列を読み出す;
                       break;
                case   スロット表示 ：
                       定義された文字列を読み出す;
                       break;
              }
        snd_msg();    /* LCD 表示タスクへメッセージ領域を送信   */
        }
    }
}
```

図 5.3-29　**メッセージタスクの処理概要**

```
LCD 表示データ管理タスク ()
{
    for ( ; ; ) {
        rcv_msg ();      /* メッセージ領域獲得を待つ  */
        カーソルを 1 行目先頭に移動;
            for (32 文字分ループ) {
                if (17 文字目のとき) {
                    カーソルを 2 行目先頭に移動;
                }
                wai_sem ();    /* 次データ送信許可待ち */
                set_flg ();     /* 表示内容を送信 */
                sig_sem ();     /* LCD 表示タスク動作許可 */
            }
        snd_msg();    /* メッセージ領域を解放(送信)する  */
        }
    }
}
```

図 5.3-30　LCD 表示データ管理タスクの処理概要

```
LCD 表示タスク ()
{
    for ( ; ; ) {
        wai_sem ();      /* LCD ユニットへの送信許可待ち */
        wai_flg ();      /* 表示内容を受信 */
        LCD ユニットへデータ送信;
        sig_sem ();      /* LCD 表示データ管理タスク動作許可 */
    }
}
```

図 5.3-31　LCD 表示データ管理タスクの処理概要

5.3.12　シリアル通信

外部からシリアル通信を通して文字受信し、それを LCD に表示する処理を行います。そして受信した文字列を記憶しておくことによって簡単なメモ機能を実現します。また受信文字確認のため、受信の度にその文字を送信元へシリアル転送をします。

5.3.12.1　　受信の手順

今回は V853 に内蔵されている UART（アンシンクロナス・シリアル・インタフェース）を用いてシリアル送受信を行います。

シリアル受信処理の流れは図 5.3-32 のようになります。

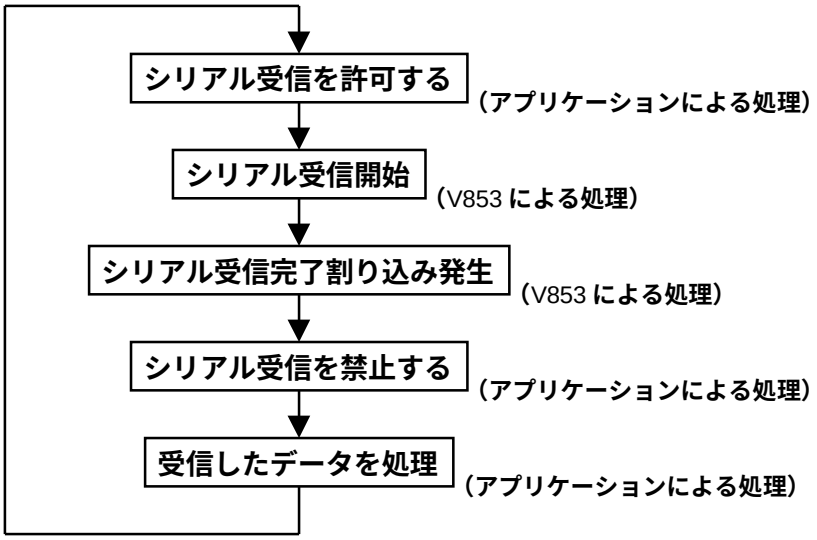

図 5.3-32　シリアル受信処理の流れ図

この図で「シリアル受信を許可する」「シリアル受信を禁止する」そして「受信したデータを処理する」はアプリケーションにて行います。そして「シリアル受信開始」と「シリアル受信完了割り込み発生」は V853 で行われます。

今回のアプリケーションでは、シリアル受信は「メモモード」のときに行われます。ですからタクト・スイッチ 8 の割り込みで、シリアル受信許可の状態にします。データを受信したとき、V853 はシリアル受信完了割り込みを発生します。ですからこの処理のコーディング方法としては、受信完了割り込みが入ったときに起動される割り込みハンドラを作成しておき、そこでシリアル受信禁止処理と

受信データ処理を行うタスクを起床するようにします。今回のアプリケーションでは行っていませんが、シリアル受信エラーが発生したときの処理も入れると確実です。シリアル受信エラーが発生すると「シリアル受信エラー割り込み」が発生しますので、エラー検出用の割り込みハンドラを作成してエラーの対処を行うと、より確実な通信が実現します。

5.3.12.2　受信データの処理

受信データ処理は、LCD に文字を表示する処理や、LCD のカーソル移動処理があります。つまりシリアルで受信した 8 ビットコードによって場合分けをし、LCD にデータを送ります。カーソル位置とデータ記憶領域の場所を一対一にきちんと管理するようにします。

LCD に送信する処理は、「5.3.11　LCD 表示」で作成したタスクをそのまま使用します。図 5.3-33 は C 言語で記述したときのメモ管理タスク関連の処理概要です。

```
間接割り込みハンドラ()        /* メモ管理タスク起床用    */
{
        wup_tsk (メモ管理タスク);       /* メモ管理タスクを起床 */
        シリアル受信禁止;
        シリアル送信許可;
        return (0xff);                  /* 間接割り込みハンドラから復帰        */
}
```

```
メモ管理タスク()
{
    for ( ; ; ) {
        slp_tsk ();          /* 自タスク起床待ち    */
        switch (メモ表示フラグの状態) {
            case   全消去 :
                シリアル受信禁止;
                表示している内容を全部消去;
            case   再表示 :
                シリアル受信禁止;
                記憶している内容を表示;
            case   メモ書き込み :
                メッセージを受信し、内容によって場合分け;
        }
        シリアル受信許可;
     }
}
```

図 5.3-33　**メモ管理タスク関連の処理概要**

5.3.13　初期化ハンドラ

V853 などの CPU や、ユーザが作成したターゲットの初期化など、アプリケーションを起動させる前にしなくてはならない処理があります。これらの処理は、「スタートアップルーチン」または「初期化ハンドラ」で行うと便利です。

初期化ハンドラとは、RX850 がスタートし、スケジューラが起動する前に起動されるハンドラです。ユーザはここで RX850 起動前にしておきたい処理などを記述しておくことが可能です。今回のアプリケーションではこの初期化ハンドラで、CPU の初期化やターゲットの初期化、タスクの起動、グローバル変数や関数の初期化などを行っています。

前にも述べたように、これらの初期化処理を「初期タスク」という一つのタスクを生成して、そこで行うこともできます。しかし限られたタスク数とメモリの中で行うには、初期化ハンドラを使用するほうが賢明です。

5.4　優先度の決定

優先度の決め方によって、アプリケーションの動作が変わってきます。初めのうちは優先度の高低だけを決めておいて開発する方法で構いませんが、最後には各機能の関連や使用優先度数を考慮に入れながら、微妙な調整を行う必要があります。

使用優先度数に余裕があれば、一つ一つに違う優先度をつけて調整することができます。もし優先度数に余裕がなければ、複数のタスクに同一優先度をつける必要があります。この際アプリケーションの動作がきちんと行われるように調整します。

5.5　タスクのスタックサイズ / 割り込みハンドラのスタックサイズの見積もり

タスクのスタックサイズの見積もりについてです。この大きさが使用するメモリサイズに大きく影響してきます。小さく見積もり過ぎるとスタック領域がオーバーフローし、プログラムが暴走するなど予期せぬ動作を引き起こす原因になります。リアルタイム OS のアプリケーションで期待した動作をしない原因の一つとして、このタスクのスタック破壊によるところが多く見られます。前述のように、プログラム作成中はスタックサイズにあまり気にかけられないことが多いですが、最終的にきちんと決める必要があります。

ここでは、タスクのスタックサイズ、そして割り込みが入ったときに使用される割り込みハンドラのスタックサイズの見積もり方について説明します。

5.5.1　タスク・スタックのサイズ

5.2.1.3 でも説明しましたが、関数呼び出しや自動変数などでユーザが使用する以外に、タスクのスタックに積まれる内容は次の通りです。

- タスク・コンテキスト
- 割り込みが入ったときにセーブされるレジスタ情報

タスク・コンテキストは、ディスパッチ時に退避するもので、

- 自動変数
- 関数呼び出しで戻り値を保存するレジスタ
- 戻りアドレス（lp）
- PSW
- そのタスクで使用している ep の値
  （ただし GHS コンパイラを使用し、タスク毎に ep を持つプログラムに設定している場合のみ）

がその内容に当たります。つまりタスクを復帰するために必要な情報を保存する領域です。使用するレジスタモードにより格納される自動変数の数が変わってきます。表 5.5-1 はレジスタモード別のタスク・コンテキストの内容です。

レジスタ・モード	コンテキスト内容	サイズ
22 レジスタ・モード	r10、r25～r29、lp、PSW	32 (0x20) bytes
26 レジスタ・モード	r10、r23～r29、lp、PSW	40 (0x28) bytes
32 レジスタ・モード	r10、r20～r29、lp、PSW	52 (0x34) bytes

表 5.5-1　タスクコンテキストの内容

上のレジスタ情報が、タスクのディスパッチ時にセーブされる内容です。r10 には関数呼び出しで戻り値が、r20～r29 にはタスク内で使用される自動変数が格納されています。タスク毎に ep を設定しているプログラムのときは、上の値に 4 バイト加えたサイズとなります。

表 5.5-2 は割り込みが入ったときに退避されるレジスタ情報です。なおこの時の割り込みは「間接割り込みハンドラの場合」で、OS がスタックに積む内容です。

レジスタ・モード	割り込み時にセーブされる内容	サイズ
22 レジスタ・モード	r1、r6～r9、r10～r14、sp、lp、EIPC、EIPSW	56 (0x38) bytes
26 レジスタ・モード	r1、r6～r9、r10～r16、sp、lp、EIPC、EIPSW	64 (0x40) bytes
32 レジスタ・モード	r1、r6～r9、r10～r19、sp、lp、EIPC、EIPSW	76 (0x4c) bytes

表 5.5-2　割り込み時にセーブされる内容

タスク処理中に割り込みが入った場合、表 5.1-2 の中の sp（スタック・ポインタ）と、lp は割り込みハンドラのスタックに積まれます。つまり表で「sp、lp 以外」のレジスタを積んだ後、割り込みハンドラのスタックにスタック・ポインタが切り替わり、そこから sp、lp が積まれることになります。ですから「タスクのスタックに積まれる情報のサイズ」は、上の表のサイズより 8bytes 少ない値になります。多重割り込みの場合は、これらすべてが割り込みハンドラのスタックに積まれます。

タスク・コンテキストと割り込み時のセーブ内容の両方が積まれる状態があります。それは割り込みハンドラ内で、割り込まれたタスクよりも優先度の高いタスクを起動するような処理を行った場合です。この時優先度の高いタスクに処理が移る前に、割り込み時にセーブされた内容に加えて、割り込まれたタスクのコンテキストが積まれます。もし割り込みハンドラの中で、他のタスクに遷移するようなシステム・コールを発行しないのであれば、両方のサイズを足した分の領域を取る必要はありません。一般的には割り込み内でタスクを遷移させるシステム・コールが発行されることが多いので、最低限、表 5.5-3 で示されたサイズ分を余分に確保します。

レジスタ・モード	サイズ
22 レジスタ・モード	32(0x20) + 48(0x30) =　80(0x50)bytes
26 レジスタ・モード	40(0x28) + 56(0x38) =　96(0x60)bytes
32 レジスタ・モード	52(0x34) + 68(0x44) = 120(0x78)bytes

表 5.5-3　タスクのスタックサイズ

この表でも、タスク毎の ep を使用しているプログラムのときは上の値に 4 バイトを加えたサイズとなります。

また可変長メモリ・プールを使用している場合に限り、システム・コール処理中にタスク・スタックを使用します。消費量は最大 12 バイトです。ですから、可変長メモリ・プールを使用している場合は、さらに 12 バイト余分に確保してください。なお、該当するシステム・コールは

get_blk、pget_blk、tget_blk、rel_blk、ter_tsk、rel_wai

です。可変長メモリ・プールを使用している場合のタスクのスタック・サイズは表 5.5-4 のようになります。

レジスタ・モード	サイズ
22 レジスタ・モード	32（0x20）+ 48（0x30）+ 12　=　92（0x5c）bytes
26 レジスタ・モード	40（0x28）+ 56（0x38）+ 12　=　108（0x6c）bytes
32 レジスタ・モード	52（0x34）+ 68（0x44）+ 12　=　132（0x84）bytes

表 5.5-4　可変長メモリ・プールを使用しているときのタスクのスタックサイズ

なお直接割り込みハンドラを使用するときは、マクロによるレジスタ退避処理の記述方法を使用することにより、コンフィギュレーション・ファイルで指定する割り込みスタックが使用されます。

5.5.2　割り込みハンドラのスタックサイズ

割り込みハンドラのスタックに積まれる内容は、

- 割り込みが入ったときの sp と lp
- RX850 がシステム・クロックが入る度に起動する「タイマ・ハンドラ」が使用するレジスタの情報を退避する領域
- 割り込みハンドラ内での関数コールなどにより、レジスタ情報の退避が必要になったときにセーブする領域
- 多重割り込みも許可する場合にレジスタ情報をセーブする領域

です。

タイマ・ハンドラが起動される際に退避される内容は、r25～r28、lp の 5 つです。多重割り込みを考えないのであれば、使用する割り込みハンドラのスタックサイズは、最低限 sp、lp、r25～r28、lp の分となります。割り込みハンドラ内で関数コールをしている場合などは、関数の復帰アドレス（lp）など割り込みハンドラ内でレジスタ情報退避が必要ですので、その分を加えて確保します。

多重割り込みを考えた場合は、上のサイズに「5.5.1　タスク・スタックのサイズ」で説明した、割り込み時のセーブ内容分のサイズを確保する必要があります。これも何重の割り込みまで使用するかによって、確保するサイズが変わってきます。最も使われた時の状態を考えて確保する必要があります。

第6章　コーディング・リスト

この章では、今回のアプリケーションのコーディング・リストを示します。ファイル名と、そのファイルに含まれるタスク・関数のリストは次のとおりです。

ファイル名	タスク・関数名
init.c	InitTask()
initfunc.c	InitFunc()
displed.c	LedDriveTask()
ledflash.c	LedFlashTask()
ledsend.c	ToMBX(), FromMBX()
timectrl.c	TickTime(), TimeControl()
timecnt.c	TimeCount()
timeset.c	SetTime()
tmrctrl.c	TimerDrive(), TimerControl()
tmrcnt.c	TimerCountDown()
tmrset.c	SetTimerCount()
stpwctrl.c	StopWatchDrive(), StopWatchControl()
stpwcnt.c	StopWatchCount()
stpwact.c	StopWatchAction()
slotctrl.c	SlotDrive(), SlotControl()
slotrot.c	SlotRotation()
stepper.c	Stepper(), OneStep(), RotSpeed()
buzzer.c	Buzzer()
adconv.c	MesureCondition(),ADstart(),ADfinish(),DataCorrection(),MesuringManage()
autodisp.c	AutoDispChange()
initlcd.c	InitLCD()
lcdchar1.c	LCDChar()
lcdchar2.c	LCDCondition()
lcdctrl.c	LCDControl()
lcddisp.c	LCDDisp()
lcdfunc.c	LCD_RS(), LCD_E(), LCDSetvalue(), LCDDataSet()
lcdsend.c	LCDSendChar()
lcdset.c	LCDCharSet()
lcdtime.c	LCDTimeChar(), LCDTimeFixChar()
lcdtimer.c	LCDTimerChar(), LCDTimerWorkChar()
lcdstpw.c	LCDStopWatchChar1, LCDStopWatchChar10()
lcdslot.c	LCDSlotChar()
lcdcond.c	LCDConditionChar(), ConvertJIS()

codeman.c	CodeManager()
memoctrl.c	MemoDrive(), MemoControl()
memoinit.c	MemoInit()
memoset.c	MemoManager()
sndcheck.c	SendCheck(), UARTCharSend()
inttakt1.c	TaktSwitch1()
inttakt2.c	TaktSwitch2()
inttakt3.c	TaktSwitch3()
inttakt4.c	TaktSwitch4()
inttakt5.c	TaktSwitch5()
inttakt6.c	TaktSwitch6()
inttakt7.c	TaktSwitch7()
inttakt8.c	TaktSwitch8()
inttouch.c	TouchSenser()
intad.c	IntADfinish()
intctrl.c	IntControl()
start.850	スタートアップ・ルーチン（GHS 版）
start.s	スタートアップ・ルーチン（NEC 版）
meminit.c	メモリ初期化ルーチン（GHS 用）
port.850	V853 ポート入出力ルーチン（GHS 版）
port.s	V853 ポート入出力ルーチン（NEC 版）
idlhdr.c	アイドル処理ルーチン
halt.850	HALT 処理（GHS 版）
halt.s	HALT 処理（NEC 版）
idle.850	IDLE 処理（GHS 版）
idle.s	IDLE 処理（NEC 版）
stop.850	STOP 処理（GHS 版）
stop.s	STOP 処理（NEC 版）
vector.850	割り込み分岐処理（GHS 版）
vector.s	割り込み分岐処理（NEC 版）
sit.cf	コンフィギュレーション・ファイル（GHS 版）
sit.cf	コンフィギュレーション・ファイル（NEC 版）
common.h	各種定義ヘッダ・ファイル
port.h	V853 用ポート名定義ファイル
sample.h	ヘッダ・ファイル
lcfile.lnk	リンク・マップ・ファイル（GHS 版）
lcfile.lnk	リンク・マップ・ファイル（NEC 版）
makefile	make ファイル（GHS 版）
makefile	make ファイル（NEC 版）

【 init.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :  init.c
   Target              :  V853
   Procedure Defined   :  init_handler ()
*****************************************************************************/
/*** ヘッダファイル ***/
#include   "sample.h"
#include   "common.h"

/*** タスク・関数宣言 ***/
void   init_handler (void);     /* 初期化ハンドラ    */
void   InitFunc (void);         /* 初期化用関数      */

/*** グローバル変数宣言 ***/
int flg_ledmode;        /* 現在の LED 表示モード    */
int flg_figurefix;      /* 調整桁認識フラグ         */

int flg_time;           /* 現在時刻動作フラグ                */
int flg_timer;          /* タイマー動作フラグ                */
int flg_stopwatch;      /* ストップウォッチ動作フラグ        */
int flg_slot;           /* スロットマシン動作フラグ          */
int flg_slotfigure;     /* スロットマシン桁認識フラグ        */
int flg_interrupt;      /* 割り込み制御フラグ                */
int flg_laptime;        /* ラップタイム表示フラグ            */
int flg_swatchmode;     /* ストップウォッチモードフラグ      */

int flg_condition;      /* 温度・湿度表示フラグ   */
int flg_memo;           /* メモ表示フラグ         */
int flg_memodisp;       /* メモ状態フラグ         */

/*****************************************************************************
   種　類　: 初期化ハンドラ
   関数名　: init_handler ()
   機　能　: OS 起動前に呼び出されるハンドラ
             全タスク起動・各種変数初期化を行う
   引き数　: なし
   返り値　: なし
*****************************************************************************/
void
init_handler (void)
{

   struct LEDMAIL   *msgblk;     /* メモリブロックのアクセスアドレス格納領域 */
   ER       err;                 /* エラーコード   */

```

```
48      InitFunc ();                /* ハードウエア初期化 */
49
50      idlemode = HALT_MODE;   /* CPU パワーセーブモード定義      */
51                              /* HALT モード : HALT_MODE       */
52                              /* IDLE モード : IDLE_MODE       */
53                              /* STOP モード : STOP_MODE       */
54      /*****************
55          各タスク起動
56      *****************/
57      err = sta_tsk (TSK_INITLCD, 0x0);           /* LCD 初期タスク                    */
58      err = sta_tsk (TSK_LEDDRIVE, 0x0);          /* LED ダイナミックドライブタスク    */
59      err = sta_tsk (TSK_TIMECTRL, 0x0);          /* 現在時刻管理タスク                */
60      err = sta_tsk (TSK_INTCONTROL, 0x0);        /* 割り込み制御タスク                */
61      err = sta_tsk (TSK_STEPPER, 0x0);           /* ステッピングモータータスク        */
62      err = sta_tsk (TSK_LEDFLASH, 0x0);          /* 秒刻み LED 点灯・消灯タスク       */
63      err = sta_tsk (TSK_TMRCONTROL, 0x0);        /* タイマー制御タスク                */
64      err = sta_tsk (TSK_SWATCHCTRL, 0x0);        /* ストップウォッチ制御              */
65      err = sta_tsk (TSK_SLOTCTRL, 0x0);          /* スロットマシン制御                */
66      err = sta_tsk (TSK_LCDCHAR, 0x0);           /* LCD 表示文字管理(固定文字列)      */
67      err = sta_tsk (TSK_LCDCOND, 0x0);           /* LCD 表示文字管理(温度・湿度)      */
68      err = sta_tsk (TSK_LCDSEND, 0x0);           /* LCD 表示文字を一文字ずつ送信      */
69      err = sta_tsk (TSK_LCDDISP, 0x0);           /* LCD 表示タスク                    */
70      err = sta_tsk (TSK_AUTOCHANGE, 0x0);        /* 表示を温度・湿度表示にする        */
71      err = sta_tsk (TSK_MESURE, 0x0);            /* A/D コンバートするタスク          */
72      err = sta_tsk (TSK_MEMOCONTROL, 0x0);       /* メモ管理タスク                    */
73      err = sta_tsk (TSK_UARTSEND, 0x0);          /* シリアル受信結果送信タスク        */
74      err = sta_tsk (TSK_BUZZER, 0x0);            /* ブザータスク                      */
75
76      /*********************************
77          各グローバル変数初期化
78      *********************************/
79      flg_ledmode      = CURRENT_TIME;    /* LED 表示モードフラグ          */
80      flg_figurefix    = FIGURE_CLR;      /* 調整桁認識フラグ              */
81      flg_time         = TIME_TICK;       /* 現在時刻表示モードフラグ      */
82      flg_timer        = TIMER_STOP;      /* タイマー動作フラグ            */
83      flg_stopwatch    = SWATCH_STOP;     /* ストップウォッチ動作フラグ    */
84      flg_slot         = SLOT_STOP;       /* スロットマシン動作フラグ      */
85      flg_slotfigure = SLOT_ALLCLEAR;     /* スロットマシン桁認識フラグ    */
86      flg_interrupt    = (SW5|SW6|SW7|SW8); /* 割り込み制御フラグ           */
87      flg_condition    = OFF;             /* 温度・湿度表示フラグ          */
88      flg_memo         = OFF;             /* メモ表示フラグ                */
89      flg_laptime      = OFF;             /* ラップタイム表示フラグ        */
90      flg_swatchmode = MODE_1;            /* ストップウォッチモードフラグ  */
91      flg_memo         = OFF;             /* メモ表示状態フラグ            */
92      flg_memodisp     = MEMO_ALLCLEAR;   /* メモ表示フラグ                */
93
94      /*****************
95          関数初期化
96      *****************/
```

```
 97      TimeCount (INIT, OFF);          /* 現在時刻初期化       */
 98      SetTime (INIT);                 /* タイマー初期化       */
 99      MemoManager (MEMO_INIT);        /* メモ初期値設定       */
100
101      get_blf ((T_MSG **)&msgblk, MPL_LCD);   /* LCD 送信用メールボックス用 */
102      snd_msg (MBX_FROM_LCD, (T_MSG *)msgblk);
103                                      /* 確保した領域をメールボックスで引渡し  */
104      set_flg (FLG_TEMPERATURE, TEMPINIT);
105                                      /* ステッピングモータ回転速度初期値送信  */
106
107      return;
108  }
```

【 initfunc.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   initfunc.c
5       Target              :   V853
6       Procedure Defined   :   InitFunc()
7   *****************************************************************************/
8   /*** ヘッダファイル ***/
9   #include   "sample.h"
10  #include   "common.h"
11
12  /*** タスク・関数宣言 ***/
13  void   InitFunc (void);     /* 初期化用関数  */
14
15  /*****************************************************************************
16      種　類  : 関数
17      関数名  : InitFunc ()
18      機　能  : ハード初期化
19                ・各ポート初期化
20                ・システムクロック初期化
21                ・割り込みコントロールユニット初期化
22      引き数  : なし
23      返り値  : なし
24  *****************************************************************************/
25  void
26  InitFunc (void)
27  {
28
29      /***********************************************
30          各ポート初期化
31          [P0、P1、P2、P3、P11]
32      ***********************************************/
33      outpb (P0, 0x0);        /* ポート 0 入出力クリア              */
34      outpb (PM0, 0x0);       /* ポート 0 入出力モード              */
35      outpb (PMC0, 0xf0);     /* P00 出力ポート (LCD : R/W)         */
36                              /* P01 出力ポート (LCD : RS)          */
37                              /* P02 出力ポート (LCD : E)           */
38                              /* P03 出力ポート (ステッピングモーター)          */
39                              /* P04 外部割り込み入力 [INTP110] (タクト SW : 1)*/
40                              /* P05 外部割り込み入力 [INTP111] (タクト SW : 2)*/
41                              /* P06 外部割り込み入力 [INTP112] (タクト SW : 3)*/
42                              /* P07 外部割り込み入力 [INTP113] (タクト SW : 4)*/
43
44      outpb (P1, 0x0);        /* ポート 1 入出力クリア              */
45      outpb (PM1, 0x0);       /* ポート 1 入出力モード              */
46      outpb (PMC1, 0x0);      /* P10 出力ポート (7segLED : a)       */
47                              /* P11 出力ポート (7segLED : b)       */
```

```
48                          /* P12 出力ポート (7segLED : c)  */
49                          /* P13 出力ポート (7segLED : d)  */
50                          /* P14 出力ポート (7segLED : e)  */
51                          /* P15 出力ポート (7segLED : f)  */
52                          /* P16 出力ポート (7segLED : g)  */
53                          /* P17 出力ポート (7segLED : dp)*/
54
55      outpb (P2, 0x80);   /* ポート 2 入出力クリア          */
56      outpb (PM2, 0x58);  /* ポート 2 入出力モード          */
57      outpb (PMC2, 0x0c); /* P20 出力ポート (LED - 上)      */
58                          /* P21 出力ポート (LED - 下)      */
59                          /* P22 TXD0/SO0 出力モード        */
60                          /* P23 RXD0/SI0 入力モード        */
61                          /* P24 SCK0 入出力モード          */
62                          /* P25 出力ポート (ブザー)        */
63
64      outpb (P3, 0x0);    /* ポート 3 入出力クリア                */
65      outpb (PM3, 0x0);   /* ポート 3 入出力モード                */
66      outpb (PMC3, 0x20); /* P30 出力ポート (7segLED : a1)        */
67                          /* P31 出力ポート (7segLED : a10)       */
68                          /* P32 出力ポート (7segLED : a100)      */
69                          /* P33 出力ポート (7segLED : a1000)     */
70                          /* P34 出力ポート (7segLED : LED_C)     */
71                          /* P35 外部割り込み入力 [INTP131]       */
72                          /*      (タッチセンサ入力)              */
73                          /* P36 出力ポート ()                    */
74                          /* P37 出力ポート ()                    */
75
76      outpb (P11, 0x0);   /* ポート 11 入出力クリア         */
77      outpb (PM11, 0x0);  /* ポート 11 入出力モード         */
78      outpb (PMC11, 0xf0); /* P110 出力ポート (LCD : DB4)   */
79                          /* P111 出力ポート (LCD : DB5)   */
80                          /* P112 出力ポート (LCD : DB6)   */
81                          /* P113 出力ポート (LCD : DB7)   */
82                          /* P114 外部割り込み入力 [INTP140] (タクト SW : 5)   */
83                          /* P115 外部割り込み入力 [INTP141] (タクト SW : 6)   */
84                          /* P116 外部割り込み入力 [INTP142] (タクト SW : 7)   */
85                          /* P117 外部割り込み入力 [INTP143] (タクト SW : 8)   */
86
87      /*******************************************
88          A/D コンバータ初期化
89        A/D コンバータ入力 : 2 入力
90            [温度センサ・湿度センサ]
91      *******************************************/
92      outpb (ADM0, 0x01); /* A/D コンバータ モード レジスタ 0 */
93                          /* CE=0,  CS=0(無視),  BS=0,  MS=0  */
94                          /* ANIS0 = 1, ANIS1 = 0, ANIS2 = 0  */
95
96      outpb (ADM1, 0x07); /* A/D コンバータ モード レジスタ 1 */
```

```
97                          /* A/D トリガ モード設定      */
98                          /* TRG0=0(無視),   TRG1=0,   TRG2=0   */
99
100     /*********************************************
101          システム・クロック初期化
102        [タイマ 4 使用]
103        ・クロック割り込み周期   : 1ms
104        ・V853 システムクロック : 25MHz
105     *********************************************/
106     outph (CM4, 0x30);     /* コンペア・レジスタ 4                */
107     outpb (TMC4, 0x86);    /* タイマ・コントロール・レジスタ 4 */
108
109     /*************************************************
110        シリアル・通信設定
111              ・モード・レジスタ
112              ・コントロール・レジスタ
113     *************************************************/
114     outpb (ASIM00, 0x08);     /* アンシンクロナス・シリアル                */
115                               /* インタフェース・モード・レジスタ 00     */
116     outpb (ASIM01, 0x00);     /* アンシンクロナス・シリアル                */
117                               /* インタフェース・モード・レジスタ 01     */
118
119     /*************************************************
120        シリアル・通信 ボー・レート・ジェネレータ設定
121           [UART0 使用]
122              ・ボーレート       : 9600 bps
123              ・パリティ         :  なし
124              ・キャラクタ長     : 8 bit
125              ・ストップビット : 1
126        ・ボーレート・ジェネレータ(BRG0)使用
127     *************************************************/
128     outpb (BRGC0, 0x29);      /* ボー・レート・ジェネレータ          */
129                               /* コンペア・レジスタ 0                */
130     outpb (BPRM0, 0x80);      /* ボー・レート・ジェネレータ          */
131                               /* プリスケーラ・モード・レジスタ 0 */
132
133     /*********************************************
134          割り込みコントロールユニット初期化
135     *********************************************/
136     outpb (INTM1, 0x55);      /* INTP110,INTP111,INTP112,INTP113   */
137                               /*  立ち上がりエッジ有効               */
138     outpb (INTM3, 0x4);       /* INTP131  立ち上がりエッジ有効      */
139     outpb (INTM4, 0x55);      /* INTP140,INTP141,INTP142,INTP143   */
140                               /*  立ち上がりエッジ有効               */
141
142     outpb (CMIC4,   0x0);     /* INTCM4 優先度 = 0,   マスク  OPEN */
143     outpb (P11IC0, 0x1);      /* INTP110 優先度 = 1,  マスク  OPEN */
144     outpb (P11IC1, 0x1);      /* INTP111 優先度 = 1,  マスク  OPEN */
145     outpb (P11IC2, 0x1);      /* INTP112 優先度 = 1,  マスク  OPEN */
```

```
146         outpb (P11IC3, 0x1);        /* INTP113 優先度 = 1,   マスク OPEN */
147         outpb (P13IC1, 0x2);        /* INTP131 優先度 = 2,   マスク OPEN */
148         outpb (P14IC0, 0x3);        /* INTP140 優先度 = 3,   マスク OPEN */
149         outpb (P14IC1, 0x4);        /* INTP141 優先度 = 4,   マスク OPEN */
150         outpb (P14IC2, 0x5);        /* INTP142 優先度 = 5,   マスク OPEN */
151         outpb (P14IC3, 0x6);        /* INTP143 優先度 = 6,   マスク OPEN */
152         outpb (ADIC,     0x7);      /* ADIC    優先度 = 7,   マスク OPEN*/
153         outpb (SRIC0,    0x7);      /* SRIC0   優先度 = 7,   マスク OPEN */
154         outpb (STIC0,    0x7);      /* STIC0   優先度 = 7,   マスク OPEN */
155
156         return;
157     }
```

【 displed.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   displed.c
5       Target              :   V853
6       Procedure Define    :   DynamicDrive ()
7                               LedDriveTask ()
8   *****************************************************************************/
9   /*** ヘッダファイル ***/
10  #include   "sample.h"
11  #include   "common.h"
12
13
14  /*** 関数宣言 ***/
15  void   LedDriveTask (INT);
16
17  /*****************************************************************************
18      種　類　: タスク
19      関数名　: LedDriveTask ()     [ID : TSK_LEDDRIVE]
20      機　能　: 7 セグメント LED のダイナミックドライブ
21                1 周期 16 ms で起動
22      引き数　: 初期情報 (起動時のタスク引き渡し情報)
23      返り値　: なし
24  *****************************************************************************/
25  void
26  LedDriveTask (INT stacd)
27  {
28
29      struct  LEDMAIL   *from_mbx; /* メールボックスからのメッセージ */
30
31      int     figure[4];    /* figure[0] : 下一桁目情報 保管変数    */
32                            /* figure[1] : 下二桁目情報 保管変数    */
33                            /* figure[2] : 下三桁目情報 保管変数    */
34                            /* figure[3] : 下四桁目情報 保管変数    */
35
36      int     power;        /* ダイナミックドライブ電流切り替え */
37      int     fig_num;      /* ループ回数                          */
38
39      ID      mbx_id;       /* メールボックス ID                   */
40      ER      ret_value;    /* システムコール戻り値 格納変数       */
41
42      int digital[] = {
43              0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d,
44              0x7d, 0x27, 0x7f, 0x6f, 0x0
45      };                    /* 7segLED ポート送信値定義 (0 - 9, 非表示) */
46
47
48      for( ; ; ) {
```

```
49
50          /************************************
51              7 SegLED ダイナミックドライブ
52           ************************************/
53          for (fig_num = 0; fig_num < 4; fig_num++) {
54
55              switch (fig_num) {  /* 対象桁切り替え */
56
57                  /* 下一桁目 */
58                  case   0:
59                      mbx_id = MBX_FIGURE1;
60                      power   = FIGURE1_ON;
61                      break;
62
63                  /* 下二桁目 */
64                  case   1:
65                      mbx_id = MBX_FIGURE2;
66                      power   = FIGURE2_ON;
67                      break;
68
69                  /* 下三桁目 */
70                  case   2:
71                      mbx_id = MBX_FIGURE3;
72                      power   = FIGURE3_ON;
73                      break;
74
75                  /* 下四桁目 */
76                  case   3:
77                      mbx_id = MBX_FIGURE4;
78                      power   = FIGURE4_ON;
79                      break;
80              }
81
82              ret_value = prcv_msg ((T_MSG **)&from_mbx, mbx_id);
83                                      /* メールボックス ポーリング   */
84
85              /*****************************************
86                      メールボックスをポーリングした結果
87                  ・新しいメッセージがあった時
88                      値を更新して、メッセージ領域を返却。
89                      その値を LED に点灯。
90                  ・新しいメッセージがなかった時
91                      保持してある値を LED に点灯。
92               *****************************************/
93              if (ret_value == E_OK) {
94
95                  figure[fig_num] = from_mbx -> message;
96                                          /* 値を更新 */
97                  rel_blf(MPL_LED, (T_MSG *)from_mbx);
98                                          /* メッセージ領域を返却   */
99
100             }
```

```
101
102             outpb (P1, digital[figure[fig_num]]);
103             outpb (P3, ((inpb(P3) & 0xf0) | power));
104                                         /* 7 SegLED を点灯  */
105
106             /***********************************************
107                 ダイナミックドライブ間隔分 時限待ちへ。
108              ***********************************************/
109             dly_tsk (LED_INTERVAL);          /* ダイナミックドライブ間隔 */
110         }
111     }
112 }
```

【 ledflash.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   ledflash.c
5       Target              :   V853
6       Procedure Defined   :   LedFlashTask ()
7   *****************************************************************************/
8   /***  ヘッダファイル  ***/
9   #include    "sample.h"
10  #include    "common.h"
11
12
13  /***  タスク・関数宣言  ***/
14  void    LedFlashTask (INT);
15
16  /*****************************************************************************
17      種　類　: タスク
18      関数名　: LedFlashTask ()     [ID : TSK_LEDFLASH]
19      機　能　: 秒刻み LED を点灯・消灯する
20      引　数　: 初期情報 (起動時のタスク引き渡し情報)
21      返り値　: なし
22  *****************************************************************************/
23  void
24  LedFlashTask (INT stacd)
25  {
26
27      for ( ; ; ) {
28
29          slp_tsk ();             /*  自タスク起床待ち */
30
31          outpb (P2, (inpb(P2) & ~LED_SWITCH));
32          outpb (P3, (inpb(P3) | LED_ON));
33                                  /* LED  点灯     */
34
35          dly_tsk (LED_FLASH);    /* 500 ms  後に消灯    */
36
37          outpb (P3, (inpb(P3) & ~LED_ON));
38                                      /* LED  消灯     */
39      }
40  }
```

【 ledsend.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   ledsend.c
5       Target              :   V853
6       Procedure Defined   :   SendtoLED ()
7   *****************************************************************************/
8   /***  ヘッダファイル  ***/
9   #include    "sample.h"
10  #include    "common.h"
11
12
13  /***  タスク・関数宣言  ***/
14  void    ToMBX (int, int, int, MAIL *);
15  int     FromMBX (int, int, int, MAIL *);
16
17  /*****************************************************************************
18      種　類　: 関数
19      関数名　: ToMBX ()
20      機　能　: メールボックスからメールボックスへメッセージ送信
21      引き数　: メッセージの内容
22                  送信元メモリプールの ID
23                  送信先メールボックスの ID
24                  メッセージパケットの先頭アドレスを格納したアドレス
25      返り値　: なし
26  *****************************************************************************/
27  void
28  ToMBX (int contents, int SND_MPL, int RCV_MBX, MAIL *packet)
29  {
30
31      short   err;                    /* pget_blk システムコール戻り値    */
32      struct  LEDMAIL    *sndmail;    /*  メッセージ送信領域              */
33
34
35      sndmail = packet;       /*  ポインタ引渡し      */
36
37      err = pget_blf ((char **)&sndmail, SND_MPL);
38                              /*  メモリブロック獲得  */
39
40      if (err == E_OK) {                          /*  メモリブロックが獲得できた時  */
41          sndmail->message = contents;            /*  メッセージ書き込み            */
42          snd_msg (RCV_MBX, (T_MSG *)sndmail);    /*  メッセージ送信                */
43      }
44
45      return;
46  }
47
48
```

```
49  /*****************************************************************************
50      種　類　：関数
51      関数名　: FromMBX ()
52      機　能　：メールボックスからメッセージ受信
53      引き数　：受信したメッセージの内容を格納する変数
54                受信領域に使われていたメモリプールの ID
55                受信するメールボックスの ID
56      返り値　：受信したメッセージ内容
57  *****************************************************************************/
58  int
59  FromMBX (int RCV_MPL, int RCV_MBX, MAIL *packet)
60  {
61
62      short   err;                    /* prcv_msg システムコール戻り値  */
63      struct  LEDMAIL   *rcvmail;     /*  メッセージ受信領域            */
64      int     value = 0;              /*  メッセージ内容 (default = 0)  */
65
66
67      rcvmail = packet;               /*  ポインタ引渡し    */
68
69      err = prcv_msg ((T_MSG **)&rcvmail, RCV_MBX);
70                                      /*  メッセージ受信    */
71
72      if (err == E_OK) {                          /*  メッセージが獲得できた時 */
73          value = rcvmail->message;               /*  メッセージ書き込み       */
74          rel_blf (RCV_MPL, (T_MSG *)rcvmail);    /*  メッセージ領域解放       */
75      }
76
77      return (value);
78  }
```

【 timectrl.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   timectrl.c
5       Target              :   V853
6       Procedure Defined   :   TickTime ()
7                           :   TimeControl ()
8   *****************************************************************************/
9   /*** ヘッダファイル ***/
10  #include   "sample.h"
11  #include   "common.h"
12
13
14  /*** タスク・関数宣言 ***/
15  void   TickTime (void);
16  void   TimeControl (INT);
17
18  /*****************************************************************************
19      種　類　: 周期ハンドラ
20      関数名　: TickTime ()
21      機　能　: 1 秒ごとにタスク「TimeControl」を起床する。
22      引き数　: なし
23      返り値　: なし
24  *****************************************************************************/
25  void
26  TickTime (void)
27  {
28      wup_tsk (TSK_TIMECTRL);     /* 現在時刻管理タスク起床    */
29
30      return;
31  }
32
33  /*****************************************************************************
34      種　類　: タスク
35      関数名　: TimeControl ()     [ID : TSK_TIMECTL]
36      機　能　: 現在時刻管理タスク
37      引き数　: 初期情報 (起動時のタスク引き渡し情報)
38      返り値　: なし
39  *****************************************************************************/
40  void
41  TimeControl (INT stacd)
42  {
43
44      for ( ; ; ) {
45
46          slp_tsk ();     /* 自タスク起床待ち (TickTime()により起床)  */
47
48          /*****************************
49              現在時刻非表示モード時
```

```
50              ****************************/
51              if (flg_ledmode != CURRENT_TIME) {
52
53                  TimeCount (TICK, OFF);      /* 時間進行のみ  */
54
55              /*************************
56                    現在時刻表示モード時
57              *************************/
58              } else if (flg_ledmode == CURRENT_TIME) {
59
60                  TimeCount (TICK, ON);           /* 時間表示 + 時間進行    */
61                  wup_tsk (TSK_LEDFLASH);     /* 秒刻み LED 起床         */
62              }
63          }
64      }
```

【 timecnt.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]  アプリケーション・ノート
*****************************************************************************
   Module              :  timecnt.c
   Target              :  V853
   Procedure Defined   :  TimeCount ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"

/***  タスク・関数宣言  ***/
void   TimeCount (int, int);

/*****************************************************************************
   種　類　: 関数
   関数名　: TimeCount ()
   機　能　: 現在時刻を LED 表示タスクに送信する。
   引き数　: モード指定 (INIT, DISPLAY, TICK)
             7 セグメント LED 表示・非表示パラメータ
   返り値　: なし
*****************************************************************************/
void
TimeCount (int setmode, int flg_leddisp)
{

    struct  LEDBLOCK    send;         /* LED 転送用メッセージ構造体          */
    struct  LEDMAIL     *from_mbx;    /*  メールボックスからのメッセージ     */

    static  struct  FLAG flag;        /*  桁調整フラグ構造体  */
    static  struct  FIGURE  time;     /*  桁構造体            */
    static  int     second;           /*  秒カウント保持変数  */

    /*************************
        引数による場合分け
     *************************/
    switch (setmode) {

        /*********************************************
            現在時刻表示初期化
              各桁数を初期化 [00:00] し、LED に送信
         *********************************************/
        case   INIT:

            time.figure1 = time.figure2 = time.figure3 = time.figure4 = 0;
                                                /*  各桁初期化  */
            second   = 0;         /*  秒初期化    */

```

```
49              flag.move0    = ON;     /* 桁上がりフラグ一桁目初期化 */
50              flag.move1    = OFF;    /* 桁上がりフラグ一桁目初期化 */
51              flag.move2    = OFF;    /* 桁上がりフラグ二桁目初期化 */
52              flag.move3    = OFF;    /* 桁上がりフラグ三桁目初期化 */
53              flag.move4    = OFF;    /* 桁上がりフラグ四桁目初期化 */
54              flag.alloff = OFF;      /* 全桁リセットフラグ         */
55
56              ToMBX (time.figure1, MPL_LED, MBX_FIGURE1, send.led1);
57                                      /* 下一桁目送信  */
58              ToMBX (time.figure2, MPL_LED, MBX_FIGURE2, send.led2);
59                                      /* 下二桁目送信  */
60              ToMBX (time.figure3, MPL_LED, MBX_FIGURE3, send.led3);
61                                      /* 下三桁目送信  */
62              ToMBX (NODISP, MPL_LED, MBX_FIGURE4, send.led4);
63                                      /* 下四桁目送信  */
64              break;
65
66          /*********************************************************
67               他のモードから現在時刻表示モードに切り替わったとき
68                 現在時刻を LED に送信
69           *********************************************************/
70          case    DISPLAY:
71
72              ToMBX (time.figure1, MPL_LED, MBX_FIGURE1, send.led1);
73                                      /* 下一桁目送信  */
74              ToMBX (time.figure2, MPL_LED, MBX_FIGURE2, send.led2);
75                                      /* 下二桁目送信  */
76              ToMBX (time.figure3, MPL_LED, MBX_FIGURE3, send.led3);
77                                      /* 下三桁目送信  */
78
79              if (time.figure4 == 0) { /* 下四桁目が 0 の時は非表示   */
80
81                  ToMBX (NODISP, MPL_LED, MBX_FIGURE4, send.led4);
82
83              } else {            /* 下四桁目が 0 以外の時はその数を送信*/
84
85                  ToMBX (time.figure4, MPL_LED, MBX_FIGURE4, send.led4);
86              }
87
88              break;
89
90          /**********************
91               現在時刻動作処理
92           **********************/
93          case    TICK:
94
95              /***********************************************
96                   現在時刻動作再開時
97                     設定された値を、メールボックスから受け
98                     取り、現在時刻動作開始
99               ***********************************************/
```

```
100             if (flg_time == TIME_START) {
101
102                 time.figure1 = FromMBX (MPL_SET, MBX_SET1, from_mbx);
103                                         /* 下一桁目受信  */
104                 time.figure2 = FromMBX (MPL_SET, MBX_SET2, from_mbx);
105                                         /* 下二桁目受信  */
106                 time.figure3 = FromMBX (MPL_SET, MBX_SET3, from_mbx);
107                                         /* 下三桁目受信  */
108                 time.figure4 = FromMBX (MPL_SET, MBX_SET4, from_mbx);
109                                         /* 下四桁目受信  */
110                 flg_time = TIME_TICK;
111                                 /* 現在時刻動作フラグを「動作中」に変更  */
112                 second = 0;     /* 秒カウントを初期化 */
113
114                 if (time.figure4 == 0) { /* 余分な 0 の桁は非表示 */
115
116                     ToMBX(NODISP, MPL_LED, MBX_FIGURE4, send.led4);
117                                         /* 下四桁情報送信 */
118                 }
119             }
120
121             /************************
122                   秒刻み LED 表示
123              ************************/
124             if (flag.move0 == ON) {     /* 秒変更フラグ有効の時  */
125
126                 second++;               /*  秒インクリメント      */
127
128                 if (second == 60) {     /* 現在の秒数が 60 秒になったとき   */
129
130                     second = 0;         /* 秒数を 0 に戻す          */
131                     flag.move0 = OFF;   /* 下一桁目桁上がり         */
132                     flag.move1 = ON;
133                 }
134             }
135
136             /*********************
137                  下一桁表示変更
138              *********************/
139             if (flag.move1 == ON) {     /* 下一桁目桁上がりフラグ有効の時   */
140
141                 time.figure1++;         /* 下一桁目インクリメント   */
142
143                 if (time.figure1 == 10) {   /* 下一桁目が 9 を越えた時  */
144                     flag.move2 = ON;    /* 下二桁目桁上がり有効      */
145
146                 } else {    /* 下一桁目が 0 から 9 の時 */
147
148                     if (flg_leddisp == ON) {    /* 時刻表示モード ON の時 */
149
150                         ToMBX (time.figure1, MPL_LED, MBX_FIGURE1, send.led1);
```

```
151                                         /* 下一桁情報送信 */
152             }
153             flag.move0 = ON;   /* 秒刻みフラグを有効に       */
154         }
155         flag.move1 = OFF;      /* 下一桁目桁上がりフラグを無効に    */
156     }
157
158     /**********************
159           下二桁表示変更
160      **********************/
161     if (flag.move2 == ON) {       /* 下二桁目桁上がりフラグ有効の時    */
162
163         time.figure1 = 0;         /* 下一桁目リセット              */
164         time.figure2++;           /* 下二桁目インクリメント        */
165
166         if (time.figure2 == 6) {  /* 下二桁目が 5 を越えた時       */
167             flag.move3 = ON;      /* 下三桁目変更フラグ有効        */
168
169         } else {
170
171             if (flg_leddisp == ON) {   /* 時刻表示モード ON の時 */
172
173                 ToMBX (time.figure1, MPL_LED, MBX_FIGURE1, send.led1);
174                                         /* 下一桁情報送信 */
175                 ToMBX (time.figure2, MPL_LED, MBX_FIGURE2, send.led2);
176                                         /* 下二桁情報送信 */
177             }
178             flag.move0 = ON;   /* 秒刻みフラグを有効に   */
179         }
180         flag.move2 = OFF;      /* 下二桁目桁上がりフラグを無効に    */
181     }
182
183     /**********************
184           下三桁表示変更
185      **********************/
186     if (flag.move3 == ON) {       /* 下三桁目桁上がりフラグ有効の時    */
187
188         time.figure2 = 0;     /* 下二桁目リセット              */
189         time.figure3++;       /* 下三桁目インクリメント    */
190
191         if ((time.figure4 == 2)&&(time.figure3 == 4)) {
192             flag.move4 = ON;      /* 上位二桁が 24 を越えた時 */
193
194         } else if ((time.figure4 != 2)&&(time.figure3 == 10)) {
195             flag.move4 = ON;      /* 上位二桁が 09、19 の時    */
196
197         } else {              /* 上記以外の時   */
198
199             if (flg_leddisp == ON) {   /* 時刻表示モード ON の時 */
200
201                 ToMBX (time.figure1, MPL_LED, MBX_FIGURE1, send.led1);
```

```
202                                         /* 下一桁情報送信 */
203                 ToMBX (time.figure2, MPL_LED, MBX_FIGURE2, send.led2);
204                                         /* 下二桁情報送信 */
205                 ToMBX (time.figure3, MPL_LED, MBX_FIGURE3, send.led3);
206                                         /* 下三桁情報送信 */
207             }
208             flag.move0 = ON;   /* 秒刻みフラグを有効に       */
209         }
210         flag.move3 = OFF;      /* 下三桁目桁上がりフラグを無効に    */
211     }
212
213     /**********************
214           下四桁表示変更
215      **********************/
216     if (flag.move4 == ON) {        /* 下四桁目桁上がりフラグ有効の時    */
217
218         time.figure3 = 0;          /* 下三桁目リセット            */
219         time.figure4++;            /* 下四桁目インクリメント     */
220
221         if (time.figure4 == 3) { /* 下四桁目が 2 を越えた時  */
222             flag.alloff = ON;      /* すべてを 0 にリセット      */
223
224         } else {
225             if (flg_leddisp == ON) {    /* 時刻表示モード ON の時 */
226
227                 ToMBX (time.figure1, MPL_LED, MBX_FIGURE1, send.led1);
228                                         /* 下一桁情報送信 */
229                 ToMBX (time.figure2, MPL_LED, MBX_FIGURE2, send.led2);
230                                         /* 下二桁情報送信 */
231                 ToMBX (time.figure3, MPL_LED, MBX_FIGURE3, send.led3);
232                                         /* 下三桁情報送信 */
233                 if (time.figure4 == 0) {
234                                 /* 下四桁目が 0 の時は非表示   */
235                     ToMBX (NODISP, MPL_LED, MBX_FIGURE4, send.led4);
236
237                 } else {    /* 下四桁目が 0 以外の時はその数を送信*/
238
239                     ToMBX (time.figure4, MPL_LED,
240                                         MBX_FIGURE4, send.led4);
241                 }
242             }
243             flag.move0 = ON;   /* 秒刻みフラグを有効に                */
244         }
245         flag.move4 = OFF;      /* 下四桁目桁上がりフラグを無効に    */
246     }
247     /**************************
248      *    全桁すべて 0 表示    *
249      **************************/
250     if (flag.alloff == ON) { /* 23:59 -> 0:00 の時 */
251
252         time.figure1 = 0;     /* 下一桁目リセット */
```

```
253                 time.figure2 = 0;     /* 下二桁目リセット */
254                 time.figure3 = 0;     /* 下三桁目リセット */
255                 time.figure4 = 0;     /* 下四桁目リセット */
256
257                 if (flg_leddisp == ON) {    /* 時刻表示モード ON の時 */
258
259                     ToMBX (time.figure1, MPL_LED, MBX_FIGURE1, send.led1);
260                                             /* 下一桁情報送信 */
261                     ToMBX (time.figure2, MPL_LED, MBX_FIGURE2, send.led2);
262                                             /* 下二桁情報送信 */
263                     ToMBX (time.figure3, MPL_LED, MBX_FIGURE3, send.led3);
264                                             /* 下三桁情報送信 */
265                     ToMBX (NODISP, MPL_LED, MBX_FIGURE4, send.led4);
266                                             /* 下四桁情報送信 */
267                 }
268                 flag.alloff = OFF;    /* 全桁リセットフラグを無効に */
269                 flag.move0 = ON;    /* 秒刻みフラグを有効に          */
270             }
271             break;
272         }
273         return;
274     }
```

【 timeset.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   timeset.c
5       Target              :   V853
6       Procedure Defined   :   SetTime ()
7   *****************************************************************************/
8   /***  ヘッダファイル  ***/
9   #include    "sample.h"
10  #include    "common.h"
11
12
13  /***  タスク・関数宣言  ***/
14  void    SetTime (int);
15
16  /*****************************************************************************
17      種　類　: 関数
18      関数名　: SetTime ()
19      機　能　: 現在時刻を修正する。
20      引き数　: 7 セグメント LED 表示・非表示パラメータ
21      返り値　: なし
22  *****************************************************************************/
23  void
24  SetTime (int setmode)
25  {
26      struct  LEDMAIL         *to_ledmbx;     /* MBX_FIGURE1～4 へのメッセージ  */
27      struct  LEDBLOCKsend;                   /* LED 転送用メッセージ構造体          */
28
29      static  struct  FIGURE  time;           /*  桁構造体          */
30
31      /*************************
32          引数による場合分け
33       *************************/
34      switch (setmode) {
35
36          case    INIT :
37
38              time.figure1 = time.figure2 = time.figure3 = time.figure4 = 0;
39                                                      /*  各桁初期化  */
40              break;
41
42          /***********************************************
43              時間調整中の時
44                  各桁入力別(タクトスイッチ 1 ～ 4)に処理
45           ***********************************************/
46          case    FIX:
47
48              switch (flg_figurefix) {
49
```

```
50              case    FIGURE_1:
51
52                  time.figure1++;     /* 下一桁値インクリメント*/
53
54                  if (time.figure1 == 10)     time.figure1 = 0;
55                                  /* 下一桁目が 9 を越えた時、値を 0 に戻す*/
56
57                  ToMBX (time.figure1, MPL_LED, MBX_FIGURE1, to_ledmbx);
58                                  /* 下一桁目更新値を送信  */
59                  flg_figurefix = flg_figurefix & ~FIGURE_1;
60                                  /* 調整桁フラグ (1 桁目) をリセット  */
61                  break;
62
63              case    FIGURE_2:
64
65                  time.figure2++;     /* 下二桁値インクリメント */
66
67                  if (time.figure2 == 6)      time.figure2 = 0;
68                                  /* 下二桁値が 6 を越えた時、値を 0 に戻す*/
69
70                  ToMBX (time.figure2, MPL_LED, MBX_FIGURE2, to_ledmbx);
71                                  /* 下二桁目更新値を送信  */
72                  flg_figurefix = flg_figurefix & ~FIGURE_2;
73                                  /* 調整桁フラグ (2 桁目) をリセット  */
74                  break;
75
76              case    FIGURE_3:
77
78                  time.figure3++;         /* 下三桁値インクリメント */
79
80                  if (time.figure4 == 2) { /* 20 時台の設定の時    */
81
82                      if (time.figure3 == 4)      time.figure3 = 0;
83                              /* 「24」表示になった時、下三桁目を 0 にする    */
84                  }
85
86                  if (time.figure3 == 10)     time.figure3 = 0;
87                              /* 下三桁値が 10 を越えたとき、値を 0 に戻す    */
88
89                  ToMBX (time.figure3, MPL_LED, MBX_FIGURE3, to_ledmbx);
90                              /* 下三桁目更新値を送信  */
91                  flg_figurefix = flg_figurefix & ~FIGURE_3;
92                              /* 調整桁フラグ (3 桁目) をリセット  */
93
94                  break;
95
96              case    FIGURE_4:
97
98                  time.figure4++;     /* 下四桁値インクリメント   */
99
100                 if ((time.figure4 == 2)&&(time.figure3 > 3)) {
101                             /* 下三桁目に「4」以上がセットされているときに */
```

```
102                         /* 下四桁目に「2」をセットしたとき              */
103
104                 time.figure3 = 3;       /* 下三桁目を 3 にする    */
105
106                 ToMBX (time.figure3, MPL_LED, MBX_FIGURE3, to_ledmbx);
107                                         /* 下三桁目更新値を送信   */
108             }
109
110             if (time.figure4 == 3)      time.figure4 = 0;
111                 /* 下四桁値が 3 を越えたとき、値を 0 に戻す             */
112
113             if (time.figure4 == 0) {
114                                         /* 下四桁目が 0 の時は非表示   */
115                 ToMBX (NODISP, MPL_LED, MBX_FIGURE4, to_ledmbx);
116                                         /* 下四桁目更新値を送信   */
117             } else {    /* 下四桁目が 0 以外の時はその数を送信*/
118
119                 ToMBX (time.figure4, MPL_LED, MBX_FIGURE4, to_ledmbx);
120                                         /* 下四桁目更新値を送信   */
121             }
122             flg_figurefix = flg_figurefix & ~FIGURE_4;
123                                         /* 調整桁フラグ (4 桁目) をリセット */
124             break;
125         }
126         break;
127
128     /***********************************************
129         時間が再スタートされた時
130           現在時刻設定中にセットされた値を
131           TimeCount() に送信。現在時刻をドライブする
132           周期ハンドラを再起動し、カウント開始
133      ***********************************************/
134     case    START:
135
136         ToMBX (time.figure1, MPL_SET, MBX_SET1, send.led1);
137                                         /* 下一桁情報送信 */
138         ToMBX (time.figure2, MPL_SET, MBX_SET2, send.led2);
139                                         /* 下二桁情報送信 */
140         ToMBX (time.figure3, MPL_SET, MBX_SET3, send.led3);
141                                         /* 下三桁情報送信 */
142         ToMBX (time.figure4, MPL_SET, MBX_SET4, send.led4);
143                                         /* 下四桁情報送信 */
144
145         flg_time = TIME_START;      /* 状態フラグ変更    */
146         act_cyc (CYC_TICKTIME, TCY_ON);
147             /* 現在時間を刻む周期ハンドラを起動する */
148
149         break;
150
151     /***********************************************
152         現在時刻カウントがストップ(中断)された時
```

```
153                                         (修正モードに移行した時)
154                 現在時刻をドライブしている周期ハンドラを
155                 停止する
156             ************************************************/
157             case    STOP:
158
159                 act_cyc (CYC_TICKTIME, TCY_OFF);
160                         /* 時間を刻む周期ハンドラを中断する */
161                 break;
162         }
163         return;
164     }
```

【 tmrctrl.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]    アプリケーション・ノート
3   *****************************************************************************
4       Module              :   tmrctrl.c
5       Target              :   V853
6       Procedure Defined   :   TimerDrive ()
7                               TimerControl ()
8   *****************************************************************************/
9   /*** ヘッダファイル ***/
10  #include    "sample.h"
11  #include    "common.h"
12
13  /*** タスク・関数宣言 ***/
14  void    TimerDrive (void);
15  void    TimerControl (INT);
16
17  /*****************************************************************************
18      種　類　: 周期ハンドラ
19      関数名　: TimerDrive ()
20      機　能　: タイマーカウントダウン ドライバ
21      引き数　: なし
22      返り値　: なし
23  *****************************************************************************/
24  void
25  TimerDrive (void)
26  {
27      wup_tsk (TSK_TMRCONTROL);       /* タイマーカウントダウン管理タスク起床    */
28
29      return;
30  }
31
32  /*****************************************************************************
33      種　類　: タスク
34      関数名　: TimerControl ()     [ID : TSK_TMRCONTROL]
35      機　能　: タイマー管理タスク
36      引　数　: 初期情報 (起動時のタスク引き渡し情報)
37      返り値　: なし
38  *****************************************************************************/
39  void
40  TimerControl (INT stacd)
41  {
42
43      for ( ; ; ) {
44
45          slp_tsk ();             /* 自タスク起床待ち */
46
47          if (flg_ledmode != TIMER) {             /* タイマー非表示モード    */
48
49              TimerCountDown (COUNT, OFF);    /* カウントだけする          */
```

```
50
51          } else if (flg_ledmode == TIMER) {      /* タイマー表示モード              */
52
53              TimerCountDown (COUNT, ON);     /* タイマー表示 + タイマー進行*/
54              wup_tsk (TSK_LEDFLASH);         /* 秒刻み LED 起床                 */
55
56          }
57      }
58  }
```

【 tmrcnt.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :   tmrcnt.c
   Target              :   V853
   Procedure Defined   :   TimerCountDown ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"

/***  関数宣言  ***/
void   TimerCountDown (int, int);

/*****************************************************************************
   種　類　: 関数
   関数名　: TimerCountDown ()
   機　能　: タイマーカウントダウン関数
   引　数　: モード指定 (INIT, DISPLAY, COUNT)
               7 セグメント LED 表示・非表示パラメータ
   返り値　: なし
*****************************************************************************/
void
TimerCountDown (int setmode, int flg_leddisp)
{

    struct  LEDMAIL       *from_mbx;     /*  メールボックスからのメッセージ  */
    struct  LEDBLOCK send;               /* LED  転送用メッセージ構造体      */

    static  struct  FLAG flag;           /*  桁調整フラグ構造体  */
    static  struct  FIGURE  timer;       /*  桁構造体                */

    /*************************
        引数による場合分け
     *************************/
    switch (setmode) {

        /*******************************************
            タイマーが動作してなく、タイマーモードに
            切り替わったとき
              各桁数を初期化 [00:00] し、LED に送信
         *******************************************/
        case   INIT:

            timer.figure1 = timer.figure2 = timer.figure3 = timer.figure4 = 0;
                                                /*  各桁初期化  */
            flag.move1 = ON;       /*  桁下がりフラグ一桁目   */
            flag.move2 = OFF;      /*  桁下がりフラグ二桁目   */
```

```
49              flag.move3 = OFF;       /* 桁下がりフラグ三桁目   */
50              flag.move4 = OFF;       /* 桁下がりフラグ四桁目   */
51
52              ToMBX (timer.figure1, MPL_LED, MBX_FIGURE1, send.led1);
53                                      /* 下一桁目送信   */
54              ToMBX (timer.figure2, MPL_LED, MBX_FIGURE2, send.led2);
55                                      /* 下二桁目送信   */
56              ToMBX (timer.figure3, MPL_LED, MBX_FIGURE3, send.led3);
57                                      /* 下三桁目送信   */
58              ToMBX (timer.figure4, MPL_LED, MBX_FIGURE4, send.led4);
59                                      /* 下四桁目送信   */
60              break;
61
62          /*****************************************
63              タイマーが動作中で、タイマーモードに
64              切り替わったとき
65                現在値を LED に送信
66           *****************************************/
67          case   DISPLAY:
68
69              ToMBX (timer.figure1, MPL_LED, MBX_FIGURE1, send.led1);
70                                          /* 下一桁目送信   */
71
72              if ((timer.figure2 == 0)&&(timer.figure3 == 0)
73                      &&(timer.figure4 == 0)) { /* 00:0x 表示になる時   */
74
75                  ToMBX (NODISP, MPL_LED, MBX_FIGURE2, send.led2);
76                                          /* 下二桁目送信(非表示)    */
77              } else {    /* それ以外のとき    */
78
79                  ToMBX (timer.figure2, MPL_LED, MBX_FIGURE2, send.led2);
80                                          /* 下二桁目送信   */
81              }
82
83              if ((timer.figure3 == 0)&&(timer.figure4 == 0)) {
84                                      /* 00:xx 表示になる時   */
85                  ToMBX (NODISP, MPL_LED, MBX_FIGURE3, send.led3);
86                                      /* 下三桁目送信(非表示)    */
87              } else {
88
89                  ToMBX (timer.figure3, MPL_LED, MBX_FIGURE3, send.led3);
90                                      /* 下三桁目送信   */
91              }
92
93              if (timer.figure4 == 0) {     /* 0x:xx 表示になる時   */
94
95                  ToMBX (NODISP, MPL_LED, MBX_FIGURE4, send.led4);
96                                      /* 下三桁目送信(非表示)    */
97              } else {
98
99                  ToMBX (timer.figure4, MPL_LED, MBX_FIGURE4, send.led4);
```

```
100                                         /* 下四桁目送信  */
101             }
102
103             break;
104
105         /**********************
106             タイマー動作処理
107          **********************/
108         case    COUNT:
109
110             /*********************************************
111                 タイマー動作開始時
112                   設定されたタイマー値を、メールボックス
113                   から受け取り、タイマー動作開始
114              *********************************************/
115             if (flg_timer == TIMER_START) {
116
117                 timer.figure1 = FromMBX (MPL_SET, MBX_SET1, from_mbx);
118                                         /* 下一桁目受信  */
119                 timer.figure2 = FromMBX (MPL_SET, MBX_SET2, from_mbx);
120                                         /* 下二桁目受信  */
121                 timer.figure3 = FromMBX (MPL_SET, MBX_SET3, from_mbx);
122                                         /* 下三桁目受信  */
123                 timer.figure4 = FromMBX (MPL_SET, MBX_SET4, from_mbx);
124                                         /* 下四桁目受信  */
125                 flg_timer = TIMER_WORKING;
126                             /* タイマー動作フラグを「動作中」に変更  */
127
128                 /************************
129                     余計な 0 表示を消灯
130                  ************************/
131                 if (timer.figure4 == 0) {
132
133                     ToMBX (NODISP, MPL_LED, MBX_FIGURE4, send.led4);
134                                 /* 下四桁目送信 (非表示)  */
135
136                     if (timer.figure3 == 0) {
137
138                         ToMBX (NODISP, MPL_LED, MBX_FIGURE3, send.led3);
139                                 /* 下三桁目送信 (非表示)  */
140
141                         if (timer.figure2 == 0) {
142
143                             ToMBX (NODISP, MPL_LED, MBX_FIGURE2, send.led2);
144                                 /* 下二桁目送信 (非表示)  */
145                         }
146                     }
147                 }
148             }
149
150             /************************
151                 下一桁表示変更
```

```
152             **********************/
153             if (flag.move1 == ON) {
154
155                 timer.figure1--;  /* 下一桁目デクリメント */
156
157                 if (timer.figure1 == -1) {   /* 下一桁目が 0 未満になった時*/
158                     flag.move1 = OFF;      /* 下二桁目桁下がり              */
159                     flag.move2 = ON;
160
161                 } else {
162
163                     if (flg_leddisp == ON) {
164                                     /* タイマー表示モード ON の時*/
165                         ToMBX (timer.figure1, MPL_LED, MBX_FIGURE1, send.led1);
166                                                 /* 下一桁目送信   */
167                     }
168                 }
169                 /********************
170                       タイマー終了
171                  ********************/
172                 if ((timer.figure1 == 0)&&(timer.figure2 == 0)
173                       &&(timer.figure3 == 0)&&(timer.figure4 == 0)) {
174
175                     flg_timer = TIMER_FINISH;      /* タイマー終了フラグ */
176                     flg_interrupt = flg_interrupt & ~(SW6);
177                                                 /* 割り込み 6 を禁止に*/
178                     wup_tsk (TSK_BUZZER);         /* ブザータスク起床     */
179                     act_cyc (CYC_TIMER, TCY_OFF);
180                                 /* タイマーカウント用周期ハンドラを停止   */
181                     return;         /* 関数終了                              */
182                 }
183
184             }
185
186             /************************
187                   下二桁表示変更
188              ************************/
189             if (flag.move2 == ON) {
190
191                 timer.figure1 = 9;    /* 下一桁目リセット          */
192                 timer.figure2--;      /* 下二桁目デクリメント   */
193
194                 if (timer.figure2 == -1) {   /* 下二桁目が 0 未満になった時*/
195                     flag.move3 = ON;       /* 下三桁目桁下がり              */
196
197                 } else {
198
199                     if (flg_leddisp == ON) {
200                                     /* タイマー表示モード ON の時 */
201
202                         ToMBX (timer.figure1, MPL_LED, MBX_FIGURE1, send.led1);
203                                                 /* 下一桁情報送信 */
```

```
204
205                 if ((timer.figure2 == 0)&&(timer.figure3 == 0)
206                                         &&(timer.figure4 == 0)) {
207
208                     ToMBX (NODISP, MPL_LED, MBX_FIGURE2, send.led2);
209                                         /* 下二桁目送信(非表示)    */
210                 } else if (timer.figure2 != 0) {
211
212                     ToMBX (timer.figure2, MPL_LED,
213                                         MBX_FIGURE2, send.led2);
214                                     /* 下二桁目送信(非表示)        */
215                 }
216             }
217             flag.move1 = ON;        /* 下一桁目フラグを有効に    */
218         }
219         flag.move2 = OFF;           /* 下二桁目桁下がりフラグを無効に    */
220     }
221
222     /***********************
223           下三桁表示変更
224      ***********************/
225     if (flag.move3 == ON) {
226
227         timer.figure2 = 5;    /* 下二桁目リセット          */
228         timer.figure3--;      /* 下三桁目デクリメント  */
229
230         if (timer.figure3 == -1) {    /* 下三桁目が 0 未満になった時*/
231             flag.move4 = ON;        /* 下四桁目桁下がり                */
232
233         } else {
234
235             if (flg_leddisp == ON) {
236                                 /* タイマー表示モード ON の時 */
237
238                 ToMBX (timer.figure1, MPL_LED, MBX_FIGURE1, send.led1);
239                                         /* 下一桁情報送信 */
240                 ToMBX (timer.figure2, MPL_LED, MBX_FIGURE2, send.led2);
241                                         /* 下二桁情報送信 */
242
243                 if ((timer.figure3 == 0)&&(timer.figure4 == 0)) {
244
245                     ToMBX (NODISP, MPL_LED, MBX_FIGURE3, send.led3);
246                                         /* 下三桁情報送信 */
247                 } else if (timer.figure3 != 0) {
248
249                     ToMBX (timer.figure3, MPL_LED,
250                                         MBX_FIGURE3, send.led3);
251                                         /* 下三桁情報送信 */
252                 }
253             }
254             flag.move1 = ON;    /* 下一桁目フラグを有効に    */
255         }
```

```
256                 flag.move3 = OFF;       /* 下三桁目桁下がりフラグを無効に    */
257             }
258
259             /**********************
260                   下四桁表示変更
261              **********************/
262             if (flag.move4 == ON) {
263
264                 timer.figure3 = 9;    /* 下三桁目リセット          */
265                 timer.figure4--;      /* 下四桁目デクリメント   */
266
267                 if (flg_leddisp == ON) {
268                                       /* タイマー表示モード ON の時 */
269
270                     ToMBX (timer.figure1, MPL_LED, MBX_FIGURE1, send.led1);
271                                           /* 下一桁情報送信 */
272                     ToMBX (timer.figure2, MPL_LED, MBX_FIGURE2, send.led2);
273                                           /* 下二桁情報送信 */
274                     ToMBX (timer.figure3, MPL_LED, MBX_FIGURE3, send.led3);
275                                           /* 下三桁情報送信 */
276
277                     if (timer.figure4 == 0) {
278
279                         ToMBX (NODISP, MPL_LED, MBX_FIGURE4, send.led4);
280                                         /* 下四桁情報送信 (非表示) */
281                     } else if (timer.figure4 != 0) {
282
283                         ToMBX (timer.figure4, MPL_LED,
284                                         MBX_FIGURE4, send.led4);
285                                           /* 下四桁情報送信 */
286                     }
287                 }
288                 flag.move1 = ON;        /* 下一桁目フラグを有効に              */
289                 flag.move4 = OFF;       /* 下四桁目桁下がりフラグを無効に    */
290             }
291             break;
292         }
293         return;
294     }
```

【 tmrset.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :  tmrset.c
   Target              :  V853
   Procedure Defined   :  SetTimerCount ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"

/***  関数宣言  ***/
void   SetTimerCount (int);

/*****************************************************************************
   種　類　: 関数
   関数名　: SetTimerCount ()
   機　能　: タイマーカウント設定
                -- タイマー設定・スタート・ストップ --
   引き数　: モード
               ・CLEAR     : タイマークリア
               ・SETTING   : タイマーセッティング
               ・START     : タイマースタート
               ・STOP      : タイマーストップ
   返り値　: なし
*****************************************************************************/
void
SetTimerCount (int setmode)
{

    struct  LEDMAIL      *to_ledmbx;     /* MBX_FIGURE1～4 へのメッセージ  */
    struct  LEDBLOCK     send;           /* LED 転送用メッセージ構造体        */

    static  struct  FIGURE  timer;       /*  桁構造体        */

    /*************************
     *   引数による場合分け   *
     *************************/
    switch (setmode) {

        case   CLEAR:

            timer.figure1 = timer.figure2 = timer.figure3 = timer.figure4 = 0;

            break;

        /***********************************************
         *   タイマー設定中の時                        *
```

```
49          *      各桁入力別(タクトスイッチ 1 ～ 4)に処理    *
50          ************************************************/
51          case    SETTING:
52
53              switch (flg_figurefix) {
54
55                  /****************
56                   *     下一桁目     *
57                   ****************/
58                  case    FIGURE_1:
59
60                      timer.figure1++;     /* 下一桁値インクリメント*/
61
62                      if (timer.figure1 == 10)timer.figure1 = 0;
63                                           /* 下一桁目が 9 を越えた時、値を 0 に戻す*/
64
65                      ToMBX (timer.figure1, MPL_LED, MBX_FIGURE1, to_ledmbx);
66                                           /* 下一桁目更新値を送信   */
67                      flg_figurefix = flg_figurefix & ~FIGURE_1;
68                                           /* 調整桁フラグ (1 桁目) をリセット  */
69                      break;
70
71                  /****************
72                   *     下二桁目     *
73                   ****************/
74                  case    FIGURE_2:
75
76                      timer.figure2++;     /* 下二桁値インクリメント  */
77
78                      if (timer.figure2 == 6)       timer.figure2 = 0;
79                                           /* 下二桁値が 6 を越えた時、値を 0 に戻す*/
80
81                      ToMBX (timer.figure2, MPL_LED, MBX_FIGURE2, to_ledmbx);
82                                           /* 下二桁目更新値を送信   */
83                      flg_figurefix = flg_figurefix & ~FIGURE_2;
84                                           /* 調整桁フラグ (2 桁目) をリセット  */
85                      break;
86
87                  /****************
88                   *     下三桁目     *
89                   ****************/
90                  case    FIGURE_3:
91
92                      timer.figure3++;     /* 下二桁値インクリメント  */
93
94                      if (timer.figure3 == 10)timer.figure3 = 0;
95                                           /* 下三桁値が 10 を越えた時、値を 0 に戻す   */
96
97                      ToMBX (timer.figure3, MPL_LED, MBX_FIGURE3, to_ledmbx);
98                                           /* 下三桁目更新値を送信   */
99                      flg_figurefix = flg_figurefix & ~FIGURE_3;
100                                          /* 調整桁フラグ (3 桁目) をリセット  */
```

```
101                 break;
102
103             /****************
104              *    下四桁目    *
105              ****************/
106             case   FIGURE_4:
107
108                 timer.figure4++;     /* 下四桁値インクリメント */
109
110                 if (timer.figure4 == 10)timer.figure4 = 0;
111                                      /* 下四桁値が 10 を越えた時、値を 0 に戻す  */
112
113                 ToMBX (timer.figure4, MPL_LED, MBX_FIGURE4, to_ledmbx);
114                                      /* 下四桁目更新値を送信  */
115                 flg_figurefix = flg_figurefix & ~FIGURE_4;
116                                      /* 調整桁フラグ (4 桁目) をリセット */
117                 break;
118             }
119             break;
120
121         /************************************************
122          *  タイマーがスタートされた時                  *
123          *    タイマー設定中にセットされた値を          *
124          *    TimerCountDown() に送信。タイマーをドライ  *
125          *    ブする周期ハンドラを起動し、カウント開始   *
126          ************************************************/
127         case   START:
128
129             ToMBX (timer.figure1, MPL_SET, MBX_SET1, send.led1);
130                                      /* 下一桁情報送信 */
131             ToMBX (timer.figure2, MPL_SET, MBX_SET2, send.led2);
132                                      /* 下二桁情報送信 */
133             ToMBX (timer.figure3, MPL_SET, MBX_SET3, send.led3);
134                                      /* 下三桁情報送信 */
135             ToMBX (timer.figure4, MPL_SET, MBX_SET4, send.led4);
136                                      /* 下四桁情報送信 */
137
138             act_cyc (CYC_TIMER, TCY_ON);
139                 /* タイマーカウントする周期ハンドラを起動する */
140             break;
141
142         /************************************************
143          *  タイマーがストップ(中断)された時            *
144          *    タイマーをドライブしている周期ハンドラを  *
145          *    停止し、変数初期化                        *
146          ************************************************/
147         case   STOP:
148
149             act_cyc (CYC_TIMER, TCY_OFF);
150                 /* タイマーカウントする周期ハン  */
151                 /* ドラを終了する        */
152
```

```
153                 timer.figure1 = timer.figure2 = timer.figure3 = timer.figure4 = 0;
154
155                 break;
156             }
157             return;
158         }
```

【 stpwctrl.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   stpwctrl.c
5       Target              :   V853
6       Procedure Defined   :   StopWatchDrive ()
7                               StopWatchControl ()
8   *****************************************************************************/
9   /*** ヘッダファイル ***/
10  #include   "sample.h"
11  #include   "common.h"
12
13
14  /*** タスク・関数宣言 ***/
15  void   StopWatchDrive (void);
16  void   StopWatchControl (INT);
17
18  /*****************************************************************************
19      種　類　: 周期ハンドラ
20      関数名　: StopWatchDrive ()
21      機　能　: ストップウォッチ ドライバ
22      引き数　: なし
23      返り値　: なし
24  *****************************************************************************/
25  void
26  StopWatchDrive (void)
27  {
28      wup_tsk (TSK_SWATCHCTRL);         /* ストップウォッチ管理タスク起床    */
29
30      return;
31  }
32
33  /*****************************************************************************
34      種　類　: タスク
35      関数名　: StopWatchControl ()     [ID : TSK_SWATCHCTRL]
36      機　能　: ストップウォッチ管理タスク
37      引き数　: 初期情報 (起動時のタスク引き渡し情報)
38      返り値　: なし
39  *****************************************************************************/
40  void
41  StopWatchControl (INT param)
42  {
43
44      static  int     led_timing;       /* LED の点灯タイミング調整変数       */
45      ER              err;              /* preq_sem システムコール戻り値      */
46
47      for ( ; ; ) {
48
49          slp_tsk ();     /* 自タスク起床待ち。StopWatchDrive ()により起床 */
```

```
50
51          /*********************************
52              ストップウォッチ非表示モード
53          *********************************/
54          if (flg_ledmode != STOPWATCH) {
55
56              StopWatchCount (COUNT, OFF);    /* カウントだけする */
57
58              if (flg_swatchmode == MODE_10) {
59                          /* 1/10 秒モードの時は、LED 点灯タイミングを計っておく */
60                  led_timing++;
61
62                  if (led_timing == 10)   led_timing = 0;
63                                          /* タイミング初期化 */
64              }
65
66          /*********************************
67              ストップウォッチ表示モード
68          *********************************/
69          } else if (flg_ledmode == STOPWATCH) {
70
71              err = preq_sem (SEM_LEDTIMING);
72
73              if (err == E_OK)        led_timing = 0;
74                      /* ストップウォッチ起動時に LED タイミングを初期化 */
75
76              /*********************
77                   通常モードの時
78               *********************/
79              if (flg_laptime == OFF) {
80
81                  StopWatchCount (COUNT, ON);
82                              /* タイマー表示 + タイマー進行    */
83
84                  if (flg_swatchmode == MODE_10) {
85                              /* 1/10 秒モードの時、10 回に 1 回 LED 点灯 */
86
87                      led_timing++;
88
89                      if (led_timing == 10) {
90
91                          led_timing = 0;                 /* タイミング初期化    */
92                          wup_tsk (TSK_LEDFLASH);     /* 秒刻み LED 起床    */
93                      }
94
95                  } else if (flg_swatchmode == MODE_1) {
96                              /* 1/1 秒モードの時、1 回ごとに LED 点灯 */
97
98                      wup_tsk (TSK_LEDFLASH);             /* 秒刻み LED 起床 */
99                  }
100
101             /*********************************
```

```
102                 ラップタイム表示モードの時
103               ********************************/
104               } else if (flg_laptime == ON) {
105
106                   StopWatchCount (COUNT, OFF);
107                                   /* タイマー非表示 + タイマー進行 */
108
109                   if (flg_swatchmode == MODE_10) {
110                           /* 1/10 秒モードの時、10 回に 1 回 LED 点灯 */
111
112                       led_timing++;
113
114                       if (led_timing == 10) {
115
116                           led_timing = 0;                 /* タイミング初期化 */
117                           wup_tsk (TSK_LEDFLASH);     /* 秒刻み LED 起床 */
118                       }
119
120                   } else if (flg_swatchmode == MODE_1) {
121                           /* 1/1 秒モードの時、1 回ごとに LED 点灯 */
122
123                       wup_tsk (TSK_LEDFLASH);         /* 秒刻み LED 起床 */
124                   }
125               }
126           }
127       }
128   }
```

【 stpwcnt.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   stpwcnt.c
5       Target              :   V853
6       Procedure Defined   :   StopWatchCount ()
7   *****************************************************************************/
8   /***  ヘッダファイル  ***/
9   #include    "sample.h"
10  #include    "common.h"
11
12
13  /***  関数宣言  ***/
14  void    StopWatchCount (int, int);
15
16  /*****************************************************************************
17      種　類　: 関数
18      関数名　: StopWatchCount ()
19      機　能　: タイマーカウントダウン関数
20      引　数　:1 - 処理モード
21                  2 - 7  セグメント  LED  表示・非表示パラメータ
22      返り値　: なし
23  *****************************************************************************/
24  void
25  StopWatchCount (int setmode, int flg_leddisp)
26  {
27
28      struct  LEDBLOCK    send;               /* LED  転送用メッセージ構造体          */
29      struct  LEDMAIL     *from_setmbx;  /*  メールボックスからのメッセージ    */
30
31      static  struct  FLAG flag;          /*  桁調整フラグ構造体  */
32      static  struct  FIGURE  stw;        /*  桁構造体                */
33
34      int     fig2limit;                  /*  下二桁目  桁上がり限界値   */
35
36      /*************************
37          引数による場合分け
38       *************************/
39      switch (setmode) {
40
41          /*********************************************************
42              ストップウォッチが動作してなく、ストップウォッチ
43              モードに切り替わったとき
44                  各桁数を初期化  [00:00]  し、LED  に送信
45           *********************************************************/
46          case    INIT:
47
48              stw.figure1 = stw.figure2 = stw.figure3 = stw.figure4 = 0;
```

```
49                                          /* 各桁初期化 */
50
51          flag.move1    = ON;     /* 桁上がりフラグ一桁目初期化  */
52          flag.move2    = OFF;    /* 桁上がりフラグ二桁目初期化  */
53          flag.move3    = OFF;    /* 桁上がりフラグ三桁目初期化  */
54          flag.move4    = OFF;    /* 桁上がりフラグ四桁目初期化  */
55          flag.alloff = OFF;      /* 全桁リセットフラグ          */
56
57          ToMBX (stw.figure1, MPL_LED, MBX_FIGURE1, send.led1);
58                                          /* 下一桁目送信   */
59          ToMBX (stw.figure2, MPL_LED, MBX_FIGURE2, send.led2);
60                                          /* 下二桁目送信   */
61          ToMBX (stw.figure3, MPL_LED, MBX_FIGURE3, send.led3);
62                                          /* 下三桁目送信   */
63          ToMBX (stw.figure4, MPL_LED, MBX_FIGURE4, send.led4);
64                                          /* 下四桁目送信   */
65          break;
66
67      /*************************************************
68          ストップウォッチが動作中で、ストップウォッチ
69          モードに切り替わったとき
70              現在値を LED に送信
71       *************************************************/
72      case    DISPLAY:
73
74          ToMBX (stw.figure1, MPL_LED, MBX_FIGURE1, send.led1);
75                                          /* 下一桁目送信   */
76          ToMBX (stw.figure2, MPL_LED, MBX_FIGURE2, send.led2);
77                                          /* 下二桁目送信   */
78          ToMBX (stw.figure3, MPL_LED, MBX_FIGURE3, send.led3);
79                                          /* 下三桁目送信   */
80          ToMBX (stw.figure4, MPL_LED, MBX_FIGURE4, send.led4);
81                                          /* 下四桁目送信   */
82          break;
83
84      /*******************************
85          ストップウォッチ動作処理
86       *******************************/
87      case    COUNT:
88
89          /***********************
90              下一桁表示変更
91          ***********************/
92          if (flag.move1 == ON) {
93
94              stw.figure1++;              /* 下一桁目インクリメント    */
95
96              if (stw.figure1 == 10) {    /* 下一桁目が 9 を越えた時  */
97                  flag.move1 = OFF;       /* 下二桁目桁上がり有効      */
98                  flag.move2 = ON;
99
```

```
100             } else {
101
102                 if (flg_leddisp == ON) {
103                         /* ストップウォッチ表示モード ON の時 */
104
105                     ToMBX (stw.figure1, MPL_LED, MBX_FIGURE1, send.led1);
106                                             /* 下一桁目送信  */
107                 }
108             }
109         }
110
111         /**********************
112               下二桁表示変更
113          **********************/
114         if (flag.move2 == ON) {
115
116             stw.figure1 = 0;          /* 下一桁目リセット            */
117             stw.figure2++;            /* 下二桁目インクリメント      */
118
119             if (flg_swatchmode == MODE_1) {
120                         /* モード別桁上り値指定 (1/1 秒の時) */
121                 fig2limit = 6;
122
123             } else if (flg_swatchmode == MODE_10) {
124                         /* モード別桁上り値指定 (1/10 秒の時)   */
125                 fig2limit = 10;
126             }
127
128             if (stw.figure2 == fig2limit) {
129                             /* 下一桁目が限界値を越えた時      */
130                 flag.move3 = ON;   /* 下三桁目桁上がりフラグ有効      */
131
132             } else {
133
134                 if (flg_leddisp == ON) {
135                         /* ストップウォッチ表示モード ON の時 */
136
137                     ToMBX (stw.figure1, MPL_LED, MBX_FIGURE1, send.led1);
138                                             /* 下一桁目送信  */
139                     ToMBX (stw.figure2, MPL_LED, MBX_FIGURE2, send.led2);
140                                             /* 下二桁目送信  */
141                 }
142                 flag.move1 = ON;   /* 下一桁フラグを有効に                */
143             }
144             flag.move2 = OFF;      /* 下二桁目桁上がりフラグを無効に    */
145         }
146
147         /**********************
148               下三桁表示変更
149          **********************/
150         if (flag.move3 == ON) {
151
```

```
152             stw.figure2 = 0;          /* 下二桁目リセット              */
153             stw.figure3++;            /* 下三桁目インクリメント        */
154
155             if (stw.figure3 == 10) { /* 下三桁目が 9 を越えた時       */
156
157                 flag.move4 = ON;   /* 下四桁目桁上がり有効            */
158
159             } else {
160
161                 if (flg_leddisp == ON) {
162                             /* ストップウォッチ表示モード ON の時 */
163
164                     ToMBX (stw.figure1, MPL_LED, MBX_FIGURE1, send.led1);
165                                             /* 下一桁目送信   */
166                     ToMBX (stw.figure2, MPL_LED, MBX_FIGURE2, send.led2);
167                                             /* 下二桁目送信   */
168                     ToMBX (stw.figure3, MPL_LED, MBX_FIGURE3, send.led3);
169                                             /* 下三桁目送信   */
170                 }
171                 flag.move1 = ON;   /* 下一桁フラグを有効に                */
172             }
173             flag.move3 = OFF;      /* 下三桁目桁上がりフラグを無効に      */
174         }
175
176         /**********************
177             下四桁表示変更
178          **********************/
179         if (flag.move4 == ON) {
180
181             stw.figure3 = 0;          /* 下三桁目リセット              */
182             stw.figure4++;            /* 下四桁目インクリメント        */
183
184             if (stw.figure4 == 10) { /* 下四桁目が 9 を越えた時       */
185
186                 flag.alloff = ON;     /* 全桁リセットフラグを有効に  */
187
188             } else {
189
190                 if (flg_leddisp == ON) {
191                             /* ストップウォッチ表示モード ON の時 */
192
193                     ToMBX (stw.figure1, MPL_LED, MBX_FIGURE1, send.led1);
194                                             /* 下一桁目送信   */
195                     ToMBX (stw.figure2, MPL_LED, MBX_FIGURE2, send.led2);
196                                             /* 下二桁目送信   */
197                     ToMBX (stw.figure3, MPL_LED, MBX_FIGURE3, send.led3);
198                                             /* 下三桁目送信   */
199                     ToMBX (stw.figure4, MPL_LED, MBX_FIGURE4, send.led4);
200                                             /* 下四桁目送信   */
201                 }
202                 flag.move1 = ON;   /* 下一桁フラグを有効に              */
```

```
203                 }
204                 flag.move4 = OFF;       /* 下四桁目桁上がりフラグを無効に    */
205             }
206
207             /*******************
208                   全桁 0 表示
209              *******************/
210             if (flag.alloff == ON) {
211
212                 stw.figure1 = stw.figure2 = stw.figure3 = stw.figure4 = 0;
213                                                     /* 各桁リセット   */
214
215                 if (flg_leddisp == ON) {
216                     /* タイマー表示モード ON の時 */
217
218                     ToMBX (stw.figure1, MPL_LED, MBX_FIGURE1, send.led1);
219                                                     /* 下一桁目送信   */
220                     ToMBX (stw.figure2, MPL_LED, MBX_FIGURE2, send.led2);
221                                                     /* 下二桁目送信   */
222                     ToMBX (stw.figure3, MPL_LED, MBX_FIGURE3, send.led3);
223                                                     /* 下三桁目送信   */
224                     ToMBX (stw.figure4, MPL_LED, MBX_FIGURE4, send.led4);
225                                                     /* 下四桁目送信   */
226                 }
227                 flag.alloff = OFF;          /* 全桁リセットフラグを無効に */
228                 flag.move1 = ON;            /* 下一桁フラグを有効に       */
229             }
230             break;
231         }
232         return;
233     }
```

【 stpwact.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   stwact.c
5       Target              :   V853
6       Procedure Defined   :   StopWatchAction ()
7   *****************************************************************************/
8   /***  ヘッダファイル  ***/
9   #include    "sample.h"
10  #include    "common.h"
11
12
13  /***  関数宣言  ***/
14  void    StopWatchAction (int);
15
16  /*****************************************************************************
17      種　類　: 関数
18      関数名　: StopWatchAction ()
19      機　能　: ストップウォッチ動作
20      引　数　: 動作モード
21      返り値　: なし
22  *****************************************************************************/
23  void
24  StopWatchAction (int setmode)
25  {
26
27      switch (setmode) {
28
29          /*********************************
30              ストップウォッチ動作開始の時
31          *********************************/
32          case    START:
33
34              if (flg_swatchmode == MODE_1) {
35                              /* 1/1  秒モードの時  */
36
37                  act_cyc (CYC_WATCH, TCY_ON);
38                              /* 1/1  秒単位動作の周期ハンドラを起動する      */
39
40              } else if (flg_swatchmode == MODE_10) {
41                              /* 1/10  秒モードの時*/
42
43                  act_cyc (CYC_WATCH10, TCY_ON);
44                              /* 1/10  秒単位動作の周期ハンドラを起動する     */
45              }
46              flg_stopwatch = SWATCH_WORKING;
47                              /*  ストップウォッチ動作フラグ変更     */
48              break;
49
```

```
50          /*********************************
51              ストップウォッチ動作停止の時
52          *********************************/
53          case    STOP:
54
55              if (flg_swatchmode == MODE_1) {
56                              /* 1/1  秒モードの時  */
57
58                  act_cyc (CYC_WATCH, TCY_OFF);
59                              /* 1/1  秒単位動作の周期ハンドラを停止する      */
60
61              } else if (flg_swatchmode == MODE_10) {
62
63                  act_cyc (CYC_WATCH10, TCY_OFF);
64                              /* 1/10  秒単位動作の周期ハンドラを停止する        */
65              }
66              flg_stopwatch == SWATCH_STOP;
67                              /*  ストップウォッチ動作フラグ変更    */
68              break;
69      }
70      return;
71  }
```

【 slotctrl.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   slotctrl.c
5       Target              :   V853
6       Procedure Defined   :   SlotDrive ()
7                               SlotControl ()
8   *****************************************************************************/
9   /***  ヘッダファイル  ***/
10  #include    "sample.h"
11  #include    "common.h"
12
13
14  /***  タスク・関数宣言  ***/
15  void    SlotDrive (void);
16  void    SlotControl (INT);
17
18  /*****************************************************************************
19      種　類　: 周期ハンドラ
20      関数名　: SlotDrive ()
21      機　能　: スロットマシン ドライバ
22      引き数　: なし
23      返り値　: なし
24  *****************************************************************************/
25  void
26  SlotDrive (void)
27  {
28      wup_tsk (TSK_SLOTCTRL);     /*  スロットマシン管理タスク起床   */
29
30      return;
31  }
32
33  /*****************************************************************************
34      種　類　: タスク
35      関数名　: SlotControl ()     [ID : TSK_SLOTCTRL]
36      機　能　: スロットマシン管理タスク
37      引き数　: 初期情報 (起動時のタスク引き渡し情報)
38      返り値　: なし
39  *****************************************************************************/
40  void
41  SlotControl (INT stacd)
42  {
43
44      for ( ; ; ) {
45
46          slp_tsk ();     /*  自タスク起床待ち。SlotDrive()によって起床   */
47
48          /*******************************
49              スロットマシン非表示モード
```

```
50          ********************************/
51          if (flg_ledmode != SLOT) {
52
53              SlotRotation (ROTATION, OFF);
54                              /* スロットマシンが回ってて、他の表示      */
55                              /* モードに移った時は、そのまま回しておく*/
56
57          /*******************************
58              スロットマシン表示モード
59          ********************************/
60          } else if (flg_ledmode == SLOT) {
61
62              SlotRotation (ROTATION, ON);
63          }
64      }
65  }
```

【 slotrot.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :   slotrot.c
   Target              :   V853
   Procedure Defined   :   SlotRotation ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"


/***  関数宣言  ***/
void   SlotRotation (int, int);

/*****************************************************************************
   種　類　: 関数
   関数名　: SlotRotation ()
   機　能　: スロットマシン回転関数
   引　数　: 1 - 処理モード
             2 - 7 セグメント LED 表示・非表示パラメータ
   返り値　: なし
*****************************************************************************/
void
SlotRotation (int setmode, int flg_leddisp)
{

   struct  LEDBLOCK        send;  /* LED 転送用メッセージ構造体 */
   static  struct  FIGURE  slot;  /*  桁構造体                  */

   /*************************
       引数による場合分け
    *************************/
   switch (setmode) {

      /*********************************************************
          ストップウォッチが動作していなく、ストップウォッチ
          モードに切り替わったとき
            各桁数を初期化 [ 7:77] し、LED に送信
       *********************************************************/
      case   INIT:

         slot.figure1 = 7;            /*  下一桁目設定値   */
         slot.figure2 = 7;            /*  下二桁目設定値   */
         slot.figure3 = 7;            /*  下三桁目設定値   */
         slot.figure4 = NODISP;       /*  下四桁目設定値   */

         ToMBX (slot.figure1, MPL_LED, MBX_FIGURE1, send.led1);
```

```
49                                          /* 下一桁目送信  */
50          ToMBX (slot.figure2, MPL_LED, MBX_FIGURE2, send.led2);
51                                          /* 下二桁目送信  */
52          ToMBX (slot.figure3, MPL_LED, MBX_FIGURE3, send.led3);
53                                          /* 下三桁目送信  */
54          ToMBX (slot.figure4, MPL_LED, MBX_FIGURE4, send.led4);
55                                          /* 下四桁目送信  */
56          break;
57
58      /*************************************************
59          ストップウォッチが動作中で、ストップウォッチ
60          モードに切り替わったとき
61              現在値を LED に送信
62       *************************************************/
63      case   DISPLAY:
64
65          ToMBX (slot.figure1, MPL_LED, MBX_FIGURE1, send.led1);
66                                          /* 下一桁目送信  */
67          ToMBX (slot.figure2, MPL_LED, MBX_FIGURE2, send.led2);
68                                          /* 下二桁目送信  */
69          ToMBX (slot.figure3, MPL_LED, MBX_FIGURE3, send.led3);
70                                          /* 下三桁目送信  */
71          ToMBX (slot.figure4, MPL_LED, MBX_FIGURE4, send.led4);
72                                          /* 下四桁目送信  */
73          break;
74
75      /*****************************
76          スロットマシン動作処理
77       *****************************/
78      case   ROTATION:
79
80          /***********************
81              下一桁表示変更
82          ***********************/
83          if ((flg_slotfigure & SLOT_FIGURE1) != SLOT_FIGURE1) {
84                              /* 下一桁目の値がまだ止められてない時  */
85
86              slot.figure1++;       /* 下一桁目インクリメント    */
87
88              if (slot.figure1 == 10) {     /* 下一桁目が 9 を越えた時  */
89
90                  slot.figure1 = -1;     /* 0 からインクリメント開始 */
91
92              } else {
93
94                  if (flg_leddisp == ON) {
95                              /* スロット表示モード ON の時*/
96
97                      ToMBX (slot.figure1, MPL_LED, MBX_FIGURE1, send.led1);
98                                                  /* 下一桁目送信  */
99                  }
100             }
```

```
101             }
102
103             /***********************
104                   下二桁表示変更
105              ***********************/
106             if ((flg_slotfigure & SLOT_FIGURE2) != SLOT_FIGURE2) {
107                                 /* 下二桁目の値がまだ止められてない時 */
108
109                 slot.figure2++;       /* 下二桁目インクリメント */
110
111                 if (slot.figure2 == 10) {     /* 下一桁目が 9 を越えた時 */
112
113                     slot.figure2 = -1;     /* 0 からインクリメント開始 */
114
115                 } else {
116
117                     if (flg_leddisp == ON) {
118                                 /* スロット表示モード ON の時 */
119
120                         ToMBX (slot.figure2, MPL_LED, MBX_FIGURE2, send.led2);
121                                                 /* 下二桁目送信 */
122                     }
123                 }
124             }
125
126             /***********************
127                   下三桁表示変更
128              ***********************/
129             if ((flg_slotfigure & SLOT_FIGURE3) != SLOT_FIGURE3) {
130
131                 slot.figure3++;       /* 下三桁目インクリメント */
132
133                 if (slot.figure3 == 10) {     /* 下一桁目が 9 を越えた時 */
134
135                     slot.figure3 = -1;     /* 0 からインクリメント開始 */
136
137                 } else {
138
139                     if (flg_leddisp == ON) {
140                                 /* スロット表示モード ON の時 */
141
142                         ToMBX (slot.figure3, MPL_LED, MBX_FIGURE3, send.led3);
143                                                 /* 下三桁目送信 */
144                     }
145                 }
146             }
147
148             /***********************
149                   すべて停止した時
150              ***********************/
151             if ((flg_slotfigure & SLOT_ALLSTOP) == SLOT_ALLSTOP) {
152
```

```
153                 flg_slot = SLOT_STOP;       /* スロットマシン状態変更    */
154                 flg_interrupt = flg_interrupt | SW4;
155                                 /* タクトスイッチ割り込み 4 を許可   */
156                 act_cyc (CYC_SLOT, TCY_OFF);
157                                 /* スロットマシンドライバ停止       */
158
159                 if ((slot.figure1 == slot.figure2)
160                         &&(slot.figure1 == slot.figure3)) {
161                                     /* 数字がそろった時      */
162                     wup_tsk (TSK_BUZZER);       /* ブザーを鳴らす         */
163
164                 }
165             }
166             break;
167         }
168         return;
169     }
```

【 stepper.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :  stepper.c
   Target              :  V853
   Procedure Defined   :  Stepper ()
                          OneStep ()
                          RotSpeed ()
*****************************************************************************//
*** ヘッダファイル ***/
#include   "sample.h"
#include   "common.h"


/*** 関数宣言 ***/
void              Stepper (INT);
void              OneStep (void);
unsigned long     RotSpeed (short);

/*****************************************************************************
   種　類　: タスク
   関数名　: Stepper ()     [ID : TSK_STEPPER]
   機　能　: ステッピングモーター回転
   引き数　: 初期情報 (起動時のタスク引き渡し情報)
   返り値　: なし
*****************************************************************************/
void
Stepper (INT stacd)
{

    unsigned long    interval;  /* ステッピング間隔              */
    unsigned int  value;        /* 気温データ格納変数            */
    ER            err;          /* pol_flg システムコール変数*/

    for ( ; ; ) {

        err = pol_flg (&value, FLG_TEMPERATURE, 0xff, TWF_ORW|TWF_CLR);
                                            /* 現在気温を受信    */

        if (err == E_OK) {          /* 気温データを受信した時    */

            interval = RotSpeed (value);   /* 回転速度決定   */
        }
        dly_tsk (interval);    /* ステッピングモーター回転間隔   */

        OneStep ();            /* ステッピング   */
    }
}

```

```
50  /*****************************************************************************
51      種　類　: 関数
52      関数名　: OneStep ()
53      機　能　: ステッピングモーター ワンステップ駆動関数
54      引き数　: なし
55      返り値　: なし
56  *****************************************************************************/
57  void
58  OneStep (void)
59  {
60
61      outpb (P0, (inpb (P0) | STEPPER));          /* ステップ ON   */
62      outpb (P0, (inpb (P0) & ~STEPPER));         /* ステップ OFF  */
63
64      return;
65  }
66
67  /*****************************************************************************
68      種　類　: 関数
69      関数名　: RotSpeed ()
70      機　能　: ステッピングモータ回転スピード調整関数
71                気温が高くなると回転をあげる
72      引き数　: 気温
73      返り値　: ステッピング間隔
74  *****************************************************************************/
75  unsigned long
76  RotSpeed (short value)
77  {
78
79      long    interval;           /* ステッピング間隔 */
80
81
82      interval = 33 - value;      /* ステッピング間隔決定式    */
83
84      if (interval <= 0) {        /* 0 ℃ 以下でステッピング間隔最遅    */
85
86          interval = 1;
87
88      } else if (interval > 33) {   /* 33 ℃以上でステッピング間隔最速    */
89
90          interval = 33;
91      }
92
93      return ((unsigned long)interval);
94  }
```

【 buzzer.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :   buzzer.c
   Target              :   V853
   Procedure Defined   :   Buzzer ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"

/***  関数宣言  ***/
void   Buzzer (INT);

/*****************************************************************************
   種　類　: タスク
   関数名　: Buzzer()   [ID : TSK_BUZZER]
   機　能　: ブザーを鳴らす
   引き数　: 初期情報 (起動時のタスク引き渡し情報)
   返り値　: なし
*****************************************************************************/
void
Buzzer (INT stacd)
{

    int     i;      /*  ブザーの鳴る回数 */

    for ( ; ; ) {

        slp_tsk();      /*  自タスク起床待ち */

        for (i = 0; i < 4; i++) {

            outpb (P2, (inpb (P2) | BUZZER));       /*  ブザー  ON          */
            dly_tsk (BZ_INTERVAL1);                 /*  ブザー  ウエイト    */
            outpb (P2, (inpb (P2) & ~BUZZER));      /*  ブザー  OFF         */

            dly_tsk (BZ_INTERVAL2);                 /*  ブザー間隔          */
        }
    }
}
```

【 adconv.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]    アプリケーション・ノート
3   *****************************************************************************
4       Module              :   adconv.c.c
5       Target              :   V853
6       Procedure Defined   :   MesureCondition ()
7                               ADstart ()
8                               ADfinish ()
9                               DataCorrection ()
10                              MesuringManage ()
11  *****************************************************************************/
12  /***  ヘッダファイル  ***/
13  #include    "sample.h"
14  #include    "common.h"
15
16  /***  関数宣言  ***/
17  void            MesureCondition (INT);
18  void            ADstart (void);
19  void            ADfinish (void);
20  unsigned short  DataCorrection (int, unsigned short);
21  void            MesuringManage (void);
22
23  /*****************************************************************************
24      種　類　: 周期ハンドラ
25      関数名　: MesuringManage ()
26      機　能　: 5 秒ごとにタスク「MesureCondition」を起床する。
27      引き数　: なし
28      返り値　: なし
29  *****************************************************************************/
30  void
31  MesuringManage (void)
32  {
33      wup_tsk (TSK_MESURE);       /*  気温・湿度測定タスク起床 */
34
35      return;
36  }
37
38  /*****************************************************************************
39      種　類　: タスク
40      関数名　: MesureCondition ()     [ID : TSK_MESURE]
41      機　能　: 温度センサー、湿度センサーの値を読み取る
42                  LCD 表示命令があるときは、表示タスクに情報を渡す
43      引き数　: 初期情報 (起動時のタスク引き渡し情報)
44      返り値　: なし
45  *****************************************************************************/
46  void
47  MesureCondition (INT stacd)
48  {
49
```

```
50      struct  DATAMAIL *result;       /* メールボックスへのメッセージ */
51      unsigned short      temperature; /* 気温データ */
52      unsigned short      humidity;    /* 湿度データ */
53
54
55      for ( ; ; ) {
56
57          slp_tsk ();         /* 自タスク起床待ち */
58
59          ADstart ();         /* 気温・湿度値 A/D 変換開始          */
60          slp_tsk ();         /* 変換結果算出後、割り込みによって起床   */
61          ADfinish ();        /* A/D 変換終了   */
62
63          temperature = DataCorrection (TEMPERATURE, inph (ADCR0));
64                                                  /* 気温更正 */
65          humidity      = DataCorrection (HUMIDITY, inph (ADCR1));
66                                                  /* 湿度更正 */
67
68          set_flg (FLG_TEMPERATURE, temperature);
69                      /* 温度をステッピングモータ回転タスクへ通知 */
70
71          switch (flg_condition) {
72                  /* 今現在 LCD に温度・湿度が表示されているか、いないか   */
73
74              case    ON :  /* 表示されている時 */
75
76                  get_blf ((char **)&result, MPL_ADCONV);
77                                          /* メモリブロック獲得 */
78                  result -> data[0] = temperature;        /* 気温代入 */
79                  result -> data[1] = humidity;           /* 湿度代入 */
80                  snd_msg (MBX_ADDATA, (T_MSG *)result);
81                                                  /* 気温、湿度データを送信    */
82                  wup_tsk (TSK_LCDCOND);          /* 表示タスク起床            */
83
84                  break;
85
86              case    OFF : /* 表示されていない時 */
87
88                  break;
89          }
90      }
91  }
92
93  /*****************************************************************************
94      種　類　: 関数
95      関数名　: ADstart ()
96      機　能　: A/D 変換開始 (CE ビットを立てる)
97      引き数　: なし
98      返り値　: なし
99  *****************************************************************************/
100 void
```

```
101  ADstart (void)
102  {
103
104      outpb (ADM0, (inpb (ADM0) | ADM0_CE));       /* CE ビットを立てる    */
105
106      return;
107  }
108
109  /*****************************************************************************
110      種　類　: 関数
111      関数名　: ADfinish ()
112      機　能　: A/D 変換終了 (CE ビットを落とす)
113      引き数　: なし
114      返り値　: なし
115  *****************************************************************************/
116  void
117  ADfinish (void)
118  {
119
120      outpb (ADM0, (inpb (ADM0) & ~ADM0_CE));       /* CE ビットを落とす    */
121
122      return;
123  }
124
125  /*****************************************************************************
126      種　類　: 関数
127      関数名　: DataCorrection()
128      機　能　: A/D コンバート結果を更正する
129                  (温度センサー特性、湿度センサー特性に依存)
130      引き数　: 1 - 更正する要素
131                2 - 更正データ
132      返り値　: 更正結果
133  *****************************************************************************/
134  unsigned short
135  DataCorrection (int element, unsigned short data)
136  {
137
138      switch (element) {
139
140          case   TEMPERATURE:  /* 更正要素が「気温」の時    */
141
142              data = (data * 4) / 8;
143
144              if (data > 100) {      /* 値が 100 を越えた時は 100    */
145                  data = 100;
146
147              } else if (data < 0) {/* 値が 0 を下回った時は 0       */
148                  data = 0;
149
150              }
151
152              break;
```

```
153
154             case    HUMIDITY:     /* 更正要素が「湿度」の時    */
155
156                 data = data / 10;
157
158                 if (data > 100) {     /* 値が 100 を越えた時は 100    */
159                     data = 100;
160
161                 } else if (data < 0) {/* 値が 0 を下回った時は 0       */
162                     data = 0;
163
164                 }
165                 break;
166         }
167         return (data);          /* 更正値を返す   */
168     }
```

【 autodisp.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   autodisp.c
5       Target              :   V853
6       Procedure Defined   :   AutoDispChange ()
7   *****************************************************************************/
8   /***  ヘッダファイル  ***/
9   #include    "sample.h"
10  #include    "common.h"
11
12
13  /***  関数宣言  ***/
14  void    AutoDispChange (INT);
15
16  /*****************************************************************************
17      種　類　: タスク
18      関数名　: AutoDispChange ()     [ID : TSK_AUTOCHANGE]
19      機　能　: 温度・湿度表示以外の時、一定時間経つと温度表示に切り替える
20      引き数　: 初期情報 (起動時のタスク引き渡し情報)
21      返り値　: なし
22  *****************************************************************************/
23  void
24  AutoDispChange (INT stacd)
25  {
26
27      ERerr;          /* tslp_tsk  システムコールの戻り値  */
28
29      for ( ; ; ) {
30
31          /*********************************************
32              気温・湿度が LCD に表示されている時は
33              このタスクは起床待ち状態へ移行
34              LCD に他のメッセージが表示されたら、起床
35              される
36          *********************************************/
37          if (flg_condition == ON) {
38
39              slp_tsk();
40
41          /*********************************************
42              気温・湿度が LCD に表示されていない時
43              (メモ表示時を除く) このタスクは時限待ち
44              状態へ移行される
45          *********************************************/
46          } else if (flg_condition == OFF) {
47
48              err = tslp_tsk (AUTOCHANGE);     /*  時限待ち状態へ    */
49
```

```
50              switch (err) {
51
52                  /*****************************************
53                      タイムアウトした時は、温度・湿度管理して
54                      いるタスクを起床している周期ハンドラの
55                      カウントを初期化。さらに A/D コンバート
56                      タスクを起床。表示フラグを ON にする。
57                  *****************************************/
58                  case    E_TMOUT:
59
60                      act_cyc (CYC_MESURE, TCY_INI|TCY_ON);
61                                              /* 定期測定ハンドラを初期化する   */
62                      wup_tsk (TSK_MESURE);   /* センサー読み取りタスクを起床   */
63                      flg_condition = ON;     /* 表示フラグを ON にする         */
64
65                      break;
66
67                  /*****************************************
68                      他タスクから起床された時は、何もせずに
69                      自タスクを休眠する。一定時間経つ前に他の
70                      表示をするような時の事例
71                  *****************************************/
72                  case    E_OK:
73
74                      break;
75              }
76          }
77      }
78  }
```

【 initlcd.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   initlcd.c
5       Target              :   V853
6       Procedure Defined   :   InitLCD ()
7   *****************************************************************************/
8   /*** ヘッダファイル ***/
9   #include   "sample.h"
10  #include   "common.h"
11
12
13  /*** 関数宣言 ***/
14  void   InitLCD (INT);
15
16  /*****************************************************************************
17      種　類　: タスク
18      関数名　: InitLCD ()   [ID : TSK_INITLCD]
19      機　能　: LCD 初期化 (Model:M1632-0A)
20      引き数　: 初期情報 (起動時のタスク引き渡し情報)
21      返り値　: なし
22  *****************************************************************************/
23  void
24  InitLCD (INT stacd)
25  {
26
27      int     i;      /* ファンクションセット回数 */
28
29
30      LCD_E (ENABLE);         /* LCD イネーブル   */
31      LCD_RS (OFF);           /* RS = 0, R/W = 0     */
32
33      dly_tsk (LCD_INITWAIT);         /* LCD ウェイト      */
34
35      for (i = 0; i < 4; i++) {  /* ファンクションセットを 3 回繰り返す */
36
37          LCDControl (LCD_FUNCSET, INTERFACE4);   /* ファンクションセット   */
38      }
39
40      LCDControl (LCD_FUNCSET1, INTERFACE4); /* ファンクションセット          */
41                                             /* インタフェースを 4 bit 長に  */
42      LCDControl (LCD_FUNCSET2, INTERFACE8); /* ファンクションセット          */
43      LCDControl (LCD_DISPCTRL, INTERFACE8);  /* 表示 ON/OFF コントロール    */
44      LCDControl (LCD_CLEAR, INTERFACE8);     /* 表示クリア                  */
45      LCDControl (LCD_ENTRY, INTERFACE8);     /* エントリーモード            */
46
47      wup_tsk (TSK_LCDCHAR);      /* LCD 表示タスク起床 (現在時刻モード)     */
48
```

```
49          ext_tsk ();
50      }
```

【 lcdchar1.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   lcdchar1.c
5       Target              :   V853
6       Procedure Defined   :   LCDChar ()
7   *****************************************************************************/
8   /*** ヘッダファイル ***/
9   #include   "sample.h"
10  #include   "common.h"
11
12
13  /*** 関数宣言 ***/
14  void   LCDChar (INT);
15
16  /*****************************************************************************
17      種　類　: タスク
18      関数名　: LCDchar ()     [ID : TSK_LCDCHAR]
19      機　能　: モードごとに表示する LCD 内容を定義し、LCD 表示タスクに引き渡す
20      引き数　: 初期情報 (起動時のタスク引き渡し情報)
21      返り値　: なし
22  *****************************************************************************/
23  void
24  LCDChar (INT stacd)
25  {
26
27      struct  LCDMAIL       *p_msgblk;  /* メッセージの領域              */
28      ERerr;                            /* システムコール戻り値          */
29
30
31      for ( ; ; ) {
32
33          slp_tsk ();     /* 自タスク起床待ちへ */
34
35          err = prcv_msg ((T_MSG **)&p_msgblk, MBX_FROM_LCD);
36                          /* メッセージ領域を受け取る、LCD 表示タスクから */
37                          /* メモリが解放されてなかった時は、そのまま抜け  */
38                          /* てしまう                                      */
39
40          if (err == E_OK) {              /* メッセージ領域が確保できた時  */
41
42              switch (flg_ledmode) {      /* LED 表示モードによる場合分け  */
43
44                  /*****************************
45                        現在時刻表示モードの時
46                   *****************************/
47                  case    CURRENT_TIME:
48
49                      if (flg_time == TIME_TICK || flg_time == TIME_START) {
```

```
50                          /* 現在時刻表示中、または再開直後    */
51
52              LCDTimeChar (p_msgblk->data);
53                          /* 文字列定義 (現在時刻モード)      */
54
55          } else if (flg_time == TIME_FIX) {
56                          /* 現在時刻調整中の時  */
57
58              LCDTimeFixChar (p_msgblk->data);
59                      /* 文字列定義 (現在時刻モード 調整中)   */
60          }
61          break;
62
63      /*****************************
64            タイマー表示モードの時
65       *****************************/
66      case    TIMER:
67
68          if (flg_timer == TIMER_STOP
69                      || flg_timer == TIMER_INTERRUPT) {
70                          /* タイマー設定中、または中断した時 */
71              LCDTimerChar (p_msgblk->data);
72                          /* 文字列定義 (タイマーモード)         */
73
74          } else if (flg_timer == TIMER_WORKING
75                              || flg_timer == TIMER_START
76                              || flg_timer == TIMER_FINISH) {
77                          /* タイマー動作中、スタート直後        */
78              LCDTimerWorkChar (p_msgblk->data);
79                          /* 文字列定義 (タイマーモード動作中)*/
80          }
81          break;
82
83      /*********************************
84           ストップウォッチモードの時
85       *********************************/
86      case    STOPWATCH:
87
88          if (flg_swatchmode == MODE_1) {
89                              /* 1 秒モードの時*/
90              LCDStopWatchChar1 (p_msgblk->data);
91                  /* 文字列定義                                  */
92                  /*     (ストップウォッチモード 1/1 モード)   */
93
94          } else if (flg_swatchmode == MODE_10) {
95                              /* 1/10 秒モードの時*/
96
97              LCDStopWatchChar10 (p_msgblk->data);
98                  /* 文字列定義                                  */
99                  /*     (ストップウォッチモード 1/10 モード) */
100         }
101         break;
```

```
102
103                 /*****************************
104                  *     スロットマシンモードの時    *
105                  *****************************/
106                 case    SLOT:
107
108                     LCDSlotChar (p_msgblk->data);
109                                     /* 文字列定義（スロットマシン） */
110                     break;
111             }
112             snd_msg (MBX_TO_LCD, (T_MSG *)p_msgblk);    /* メッセージ送信        */
113         }
114     }
115 }
```

【 lcdchar2.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   lcdchar2.c
5       Target              :   V853
6       Procedure Defined   :   LCDCondition ()
7   *****************************************************************************/
8   /*** ヘッダファイル ***/
9   #include   "sample.h"
10  #include   "common.h"
11
12
13  /*** 関数宣言 ***/
14  void   LCDCondition (INT);
15
16  /*****************************************************************************
17      種　類　: タスク
18      関数名　: LCDCondition ()     [ID : TSK_LCDCOND]
19      機　能　: LCD 温度・湿度表示関数 (Model:M1632-0A)
20      引き数　: 初期情報 (起動時のタスク引き渡し情報)
21      返り値　: なし
22  *****************************************************************************/
23  void
24  LCDCondition (INT stacd)
25  {
26
27      struct  DATAMAIL *value;        /* データ格納領域        */
28      struct  LCDMAIL   *p_msgblk;   /* メッセージの領域      */
29
30      char    temperature;      /* 温度データ */
31      char    humidity;         /* 湿度データ */
32
33      ER      errdata;          /* システムコール戻り値 (気温・湿度データ)    */
34      ER      errmsg;           /* システムコール戻り値   (メッセージ領域)     */
35
36      for ( ; ; ) {
37
38          slp_tsk ();     /* 自タスク起床待ち */
39
40          errdata = prcv_msg ((T_MSG **)&value, MBX_ADDATA);
41                                  /* 気温・湿度データ受け取り */
42
43          if (errdata == E_OK) {      /* データが引き取れたら        */
44
45              temperature = value -> data[0];       /* 気温データ */
46              humidity       = value -> data[1];    /* 湿度データ */
47
48              rel_blf (MPL_ADCONV, (T_MSG *)value);
```

```
49                                  /* 使用したデータ領域を解放 */
50          }
51
52          errmsg = prcv_msg ((T_MSG **)&p_msgblk, MBX_FROM_LCD);
53                                  /* メッセージ領域を受け取り          */
54
55          if (errmsg == E_OK) {       /* メッセージ領域を受け取れたら   */
56
57              LCDConditionChar (p_msgblk->data, temperature, humidity);
58                                  /* 文字列定義（只今の気温・湿度）      */
59              snd_msg (MBX_TO_LCD, (T_MSG *)p_msgblk);
60                                  /* LCD 表示タスクへメッセージ送信   */
61          }
62      }
63  }
```

【 lcdctrl.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   lcdctrl.c
5       Target              :   V853
6       Procedure Defined   :   LCDControl ()
7   *****************************************************************************/
8   /*** ヘッダファイル ***/
9   #include   "sample.h"
10  #include   "common.h"
11
12
13  /*** 関数宣言 ***/
14  void   LCDControl (char, int);
15
16  /*****************************************************************************
17      種　類　: 関数
18      関数名　: LCDControl ()
19      機　能　: LCD 初期化関数 (Model:M1632-0A)
20      引き数　: LCD にセットする値
21      返り値　: なし
22  *****************************************************************************/
23  void
24  LCDControl (char value, int mode)
25  {
26
27      int     i;              /* for ループ用変数                */
28      char    sendchar;       /* LCD に送るデータ (4 bit)   */
29
30      if (mode == INTERFACE8) {          /* 送信するデータが 8 ビットの時 */
31
32          sendchar = (value >> 4) & 0xf;   /* 始めに上位 4 ビット取り出し    */
33
34      } else if (mode == INTERFACE4) { /* 送信するデータが 4 ビットの時 */
35
36          sendchar = value & 0xf;          /* 下位 4 ビット */
37      }
38
39      for (i = 0; i < mode; i++) { /* 8 ビットデータの時は 2 回に分けて*/
40                                   /* 送信、4 ビットデータの時は 1 回      */
41
42          LCDSetvalue (sendchar);          /* 4 bit 値セット          */
43          dly_tsk (LCD_FUNCWAIT);      /* LCD ウェイト          */
44
45          LCD_E (DISABLE);           /* LCD ディスエーブル */
46          dly_tsk (LCD_E_WAIT);      /* LCD ウェイト          */
47          LCD_E (ENABLE);            /* LCD イネーブル       */
48
```

```
49              dly_tsk (LCD_SETWAIT);    /* LCD ウェイト */
50
51              sendchar = value & 0xf;   /* 下位 4 ビット取り出し          */
52                                        /* (8 ビットデータ送信時有効)     */
53          }
54          return;
55      }
```

【 lcddisp.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   lcddisp.c
5       Target              :   V853
6       Procedure Defined   :   LCDDispChar ()
7   *****************************************************************************/
8   /***  ヘッダファイル  ***/
9   #include    "sample.h"
10  #include    "common.h"
11
12
13  /***  関数宣言  ***/
14  void    LCDDispChar (INT);
15
16  /*****************************************************************************
17      種　類　: タスク
18      関数名　: LCDDispChar ()     [ID : TSK_LCDDISP]
19      機　能　: LCD に一文字ずつ送信、表示する
20      引き数　: 初期情報 (起動時のタスク引き渡し情報)
21      返り値　: なし
22  *****************************************************************************/
23  void
24  LCDDispChar (INT stacd)
25  {
26
27      unsigned int  sendchar;     /*  送信する文字列          */
28
29      for ( ; ; ) {
30
31          wai_sem (SEM_LCDDISP);      /*  タスク動作許可待ち  */
32
33          wai_flg (&sendchar, FLG_LCDCHAR, 0xff, TWF_ORW|TWF_CLR);
34                                      /*  送信文字受信                    */
35                                      /*  イベントフラグでセットされた値が */
36                                      /*  そのまま送信する文字となる      */
37
38          LCDCharSet ((char)sendchar);        /*  文字表示                        */
39          sig_sem (SEM_LCDSEND);              /*  文字送信タスクの動作を許可する  */
40      }
41  }
```

【 lcdfunc.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]    アプリケーション・ノート
3   *****************************************************************************
4       Module              :   lcdfunc.c
5       Target              :   V853
6       Procedure Defined   :   LCD_RS ()
7                               LCD_E ()
8                               LCDSetvalue ()
9   *****************************************************************************/
10  /***  ヘッダファイル  ***/
11  #include   "sample.h"
12  #include   "common.h"
13
14
15  /***  関数宣言  ***/
16  void   LCD_RS (int);
17  void   LCD_E (int);
18  void   LCDSetvalue (char);
19  void   LCDDataSet (char, int);
20
21  /*****************************************************************************
22      種　類　: 関数
23      関数名　: LCD_RS ()
24      機　能　: LCD の RS を ON/OFF する
25      引き数　: ON/OFF
26      返り値　: なし
27  *****************************************************************************/
28  void
29  LCD_RS (int COMMAND)
30  {
31
32      switch (COMMAND) {
33
34          case   ON:
35
36              outpb (P0, (inpb(P0) | LCD_RS_ON));       /* RS = 1, R/W = 0 */
37              break;
38
39          case   OFF:
40
41              outpb (P0, (inpb(P0) & ~LCD_RS_ON));   /* RS = 0, R/W = 0 */
42              break;
43      }
44      return;
45  }
46
47  /*****************************************************************************
48      種　類　: 関数
49      関数名　: LCD_E ()
```

```
50      機　能　: LCD の E を ENABLE/DISABLE する
51      引き数　: ENABLE/DISABLE
52      返り値　: なし
53  ********************************************************************************/
54  void
55  LCD_E (int COMMAND)
56  {
57
58      switch (COMMAND) {
59
60          case    ENABLE:
61
62              outpb (P0, (inpb (P0) | LCD_ENABLE));       /* LCD イネーブル       */
63              break;
64
65          case    DISABLE:
66
67              outpb (P0, (inpb (P0) & ~LCD_ENABLE));    /* LCD ディスエーブル */
68              break;
69      }
70      return;
71  }
72
73  /*******************************************************************************
74      種　類　: 関数
75      関数名　: LCDSetvalue ()
76      機　能　: LCD に送信する値をセットする (4 bit 値)
77      引き数　: 送信する値 (4 bit 送信)
78      返り値　: なし
79  ********************************************************************************/
80  void
81  LCDSetvalue (char value)
82  {
83      outpb (P11, ((inpb (P11) & 0xf0) | value));      /* 4 ビットデータ送信  */
84      return;
85  }
86
87  /*******************************************************************************
88      種　類　: 関数
89      関数名　: LCDDataSet ()
90      機　能　: LCD に送信したい値、データ値をセットする
91                受け取ったデータを LCD 用にウェイトを入れ、LCD に表示する
92                タスクとやりとりする
93      引き数　: 送信する値、LCD の RS 値の ON/OFF
94      返り値　: なし
95  ********************************************************************************/
96  void
97  LCDDataSet (char data, int rs)
98  {
99
100     wai_sem (SEM_LCDSEND);          /* タスク動作許可待ち */
101     LCD_RS (rs);                    /* RS = 0, R/W = 0        */
```

```
102         set_flg (FLG_LCDCHAR, data);     /*  データセット                    */
103         sig_sem (SEM_LCDDISP);           /* LCD  表示タスクを動作許可する  */
104
105         return;
106     }
```

【 lcdsend.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :  lcdsend.c
   Target              :  V853
   Procedure Defined   :  LCDSendChar ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"


/***  関数宣言  ***/
void   LCDSendChar (INT);

/*****************************************************************************
   種　類　: タスク
   関数名　: LCDSendChar ()     [ID : TSK_LCDSEND]
   機　能　: 表示する内容を、LCD 表示タスクに一文字ずつ引き渡し、
             一気に LCD に書き込む
   引き数　: 初期情報 (起動時のタスク引き渡し情報)
   返り値　: なし
*****************************************************************************/
void
LCDSendChar (INT stacd)
{

    struct  LCDMAIL   *lcd_msg;    /*  メッセージ領域へのポインタ  */
    int     i;                     /* for  ループ用                */

    for ( ; ; ) {

        rcv_msg ((T_MSG **)&lcd_msg, MBX_TO_LCD);  /*  メッセージ受信        */

        LCDDataSet (LCD_LINE1S, OFF);              /*  カーソル  1  行目先頭*/

        for (i = 0; i < LCD_CHARNUM; i++) {

            if (i == 16) {     /* 16  文字書いたら、2  行目へカーソル移動   */

                LCDDataSet (LCD_LINE2S, OFF);      /*  カーソルを  2  行目先頭へ   */
            }

            wai_sem (SEM_LCDSEND);     /*  タスク動作許可待ち  */

            if ((i == 0)||(i == 16))   LCD_RS (ON);
                                       /*  行移動の後は  RS  を  ON  にする */

            set_flg (FLG_LCDCHAR, lcd_msg->data[i]);
```

```
50                                      /* 表示する内容をセットする      */
51              sig_sem (SEM_LCDDISP);        /* LCD 表示タスクを動作許可する  */
52          }
53
54          LCDDataSet (LCD_LINE1S, OFF); /* カーソルを 1 行目先頭へ       */
55
56          if (flg_memo == OFF) {              /* メモ表示以外はカーソル OFF */
57
58              LCDDataSet (LCD_CURSOR_OFF, OFF);
59
60          } else {        /* メモ表示の時はカーソルをブリンク */
61
62              LCDDataSet (LCD_BLINK_ON, OFF);
63
64          }
65          snd_msg (MBX_FROM_LCD, (T_MSG *)lcd_msg);
66                                  /* 使用したメモリ領域を明け渡す        */
67      }
68  }
```

【 lcdset.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :   lcdset.c
   Target              :   V853
   Procedure Defined   :   LCDCharSet ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"


/***  関数宣言  ***/
void   LCDCharSet( char );

/*****************************************************************************
   種　類　: 関数
   関数名　: LCDCharSet ()
   機　能　: LCD に指定された文字を、上位 4 ビット、下位 4 ビットに分けて
             LCD に送信する (LCD のタイミングを計りながら行う)
   引き数　: 表示する文字コード
   返り値　: なし
*****************************************************************************/
void
LCDCharSet (char value)
{

    char   sendchar;     /* 送信する文字 (8 ビット)   */
    int    i;            /* for ループ用              */

    sendchar = (value >> 4) & 0xf;    /* 上位 4 ビット取り出し */

    for (i = 0; i < 2; i++) {

        LCDSetvalue (sendchar);          /* 値セット     */
        dly_tsk (LCD_FUNCWAIT);          /* ウェイト     */

        LCD_E (DISABLE);            /* LCD ディスエーブル */
        dly_tsk (LCD_E_WAIT);       /* LCD E ウェイト       */
        LCD_E (ENABLE);             /* LCD イネーブル       */

        sendchar = value & 0xf;          /* 下位 4 ビット取り出し */
    }
    return;
}
```

【 lcdtime.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   lcdtime.c
5       Target              :   V853
6       Procedure Defined   :   LCDTimeChar ()
7                               LCDTimeFixChar ()
8   *****************************************************************************/
9   /*** ヘッダファイル ***/
10  #include   "sample.h"
11  #include   "common.h"
12
13
14  /*** 関数宣言 ***/
15  void   LCDTimeChar (char *);
16  void   LCDTimeFixChar (char *);
17
18  /*****************************************************************************
19      種　類　: 関数
20      関数名　: LCDTimeChar ()
21      機　能　: 現在時刻表示時の LCD 表示内容定義
22      引き数　: 文字列格納先の先頭アドレス
23      返り値　: なし
24  *****************************************************************************/
25  void
26  LCDTimeChar (char *timechar)
27  {
28
29      /* LCD 一行目 */
30
31      timechar[0]   = 0xb9;       /* ケ */
32      timechar[1]   = 0xde;       /* "  */
33      timechar[2]   = 0xdd;       /* ン */
34      timechar[3]   = 0xbb;       /* サ */
35      timechar[4]   = 0xde;       /* "  */
36      timechar[5]   = 0xb2;       /* イ */
37      timechar[6]   = 0xbc;       /* シ */
38      timechar[7]   = 0xde;       /* "  */
39      timechar[8]   = 0xba;       /* コ */
40      timechar[9]   = 0xb8;       /* ク */
41      timechar[10] = SPACE;       /*    */
42      timechar[11] = 0xd3;        /* モ */
43      timechar[12] = 0xb0;        /* ー */
44      timechar[13] = 0xc4;        /* ト */
45      timechar[14] = 0xde;        /* "  */
46      timechar[15] = SPACE;       /*    */
47
48      /* LCD 二行目 */
```

```
49
50      timechar[16] = SPACE;    /*      */
51      timechar[17] = SPACE;    /*      */
52      timechar[18] = SPACE;    /*      */
53      timechar[19] = SPACE;    /*      */
54      timechar[20] = SPACE;    /*      */
55      timechar[21] = SPACE;    /*      */
56      timechar[22] = SPACE;    /*      */
57      timechar[23] = SPACE;    /*      */
58      timechar[24] = SPACE;    /*      */
59      timechar[25] = SPACE;    /*      */
60      timechar[26] = SPACE;    /*      */
61      timechar[27] = SPACE;    /*      */
62      timechar[28] = SPACE;    /*      */
63      timechar[29] = SPACE;    /*      */
64      timechar[30] = SPACE;    /*      */
65      timechar[31] = SPACE;    /*      */
66
67      return;
68  }
69
70  /*****************************************************************************
71      種　類　: 関数
72      関数名　: LCDTimeFixChar ()
73      機　能　: 現在時刻調整時の LCD 表示内容定義
74      引き数　: 文字列格納先の先頭アドレス
75      返り値　: なし
76  *****************************************************************************/
77  void
78  LCDTimeFixChar (char *timechar)
79  {
80
81      /* LCD 一行目 */
82
83      timechar[0]  = 0xb9;     /* ケ */
84      timechar[1]  = 0xde;     /* "  */
85      timechar[2]  = 0xdd;     /* ン */
86      timechar[3]  = 0xbb;     /* サ */
87      timechar[4]  = 0xde;     /* "  */
88      timechar[5]  = 0xb2;     /* イ */
89      timechar[6]  = 0xbc;     /* シ */
90      timechar[7]  = 0xde;     /* "  */
91      timechar[8]  = 0xba;     /* コ */
92      timechar[9]  = 0xb8;     /* ク */
93      timechar[10] = SPACE;    /*    */
94      timechar[11] = 0xd3;     /* モ */
95      timechar[12] = 0xb0;     /* ー */
96      timechar[13] = 0xc4;     /* ト */
97      timechar[14] = 0xde;     /* "  */
98      timechar[15] = SPACE;    /*    */
99
100     /* LCD 二行目 */
```

```
101
102         timechar[16] = SPACE;    /*      */
103         timechar[17] = SPACE;    /*      */
104         timechar[18] = SPACE;    /*      */
105         timechar[19] = SPACE;    /*      */
106         timechar[20] = SPACE;    /*      */
107         timechar[21] = 0x2a;     /* *    */
108         timechar[22] = 0xc1;     /* チ   */
109         timechar[23] = 0xae;     /* ョ   */
110         timechar[24] = 0xb3;     /* ウ   */
111         timechar[25] = 0xbe;     /* セ   */
112         timechar[26] = 0xb2;     /* イ   */
113         timechar[27] = 0xc1;     /* チ   */
114         timechar[28] = 0xad;     /* ュ   */
115         timechar[29] = 0xb3;     /* ウ   */
116         timechar[30] = 0x2a;     /* *    */
117         timechar[31] = SPACE;    /*      */
118
119         return;
120     }
```

【 lcdtimer.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   lcdtimer.c
5       Target              :   V853
6       Procedure Defined   :   LCDTimerWorkChar ()
7   *****************************************************************************/
8   /***  ヘッダファイル  ***/
9   #include    "sample.h"
10  #include    "common.h"
11
12
13  /***  関数宣言  ***/
14  void    LCDTimerChar (char *);
15  void    LCDTimerWorkChar (char *);
16
17  /*****************************************************************************
18      種　類　: 関数
19      関数名　: LCDTimerChar ()
20      機　能　: タイマー表示時の LCD 表示内容定義
21      引き数　: 文字列格納先の先頭アドレス
22      返り値　: なし
23  *****************************************************************************/
24  void
25  LCDTimerChar (char *timechar)
26  {
27
28      /* LCD  一行目 */
29
30      timechar[0]   = 0xc0;       /*  タ  */
31      timechar[1]   = 0xb2;       /*  イ  */
32      timechar[2]   = 0xcf;       /*  マ  */
33      timechar[3]   = 0xb0;       /*  ー  */
34      timechar[4]   = SPACE;      /*      */
35      timechar[5]   = 0xd3;       /*  モ  */
36      timechar[6]   = 0xb0;       /*  ー  */
37      timechar[7]   = 0xc4;       /*  ト  */
38      timechar[8]   = 0xde;       /*  "   */
39      timechar[9]   = SPACE;      /*      */
40      timechar[10] = SPACE;       /*      */
41      timechar[11] = SPACE;       /*      */
42      timechar[12] = SPACE;       /*      */
43      timechar[13] = SPACE;       /*      */
44      timechar[14] = SPACE;       /*      */
45      timechar[15] = SPACE;       /*      */
46
47      /* LCD  二行目 */
48
49      timechar[16] = 0x20;    /*  */
```

```
50          timechar[17] = 0x20;    /*  */
51          timechar[18] = 0x20;    /*  */
52          timechar[19] = 0x20;    /*  */
53          timechar[20] = 0x20;    /*  */
54          timechar[21] = 0x20;    /*  */
55          timechar[22] = 0x20;    /*  */
56          timechar[23] = 0x20;    /*  */
57          timechar[24] = 0x20;    /*  */
58          timechar[25] = 0x20;    /*  */
59          timechar[26] = 0x20;    /*  */
60          timechar[27] = 0x20;    /*  */
61          timechar[28] = 0x20;    /*  */
62          timechar[29] = 0x20;    /*  */
63          timechar[30] = 0x20;    /*  */
64          timechar[31] = 0x20;    /*  */
65
66          return;
67      }
68
69      /*****************************************************************************
70          種　類　: 関数
71          関数名　: LCDTimerWorkChar ()
72          機　能　: タイマー動作中の LCD 表示内容定義
73          引き数　: 文字列格納先の先頭アドレス
74          返り値　: なし
75      *****************************************************************************/
76      void
77      LCDTimerWorkChar (char *timechar)
78      {
79
80          /* LCD 一行目 */
81
82          timechar[0]  = 0xc0;      /* タ */
83          timechar[1]  = 0xb2;      /* イ */
84          timechar[2]  = 0xcf;      /* マ */
85          timechar[3]  = 0xb0;      /* ー */
86          timechar[4]  = SPACE;     /*    */
87          timechar[5]  = 0xd3;      /* モ */
88          timechar[6]  = 0xb0;      /* ー */
89          timechar[7]  = 0xc4;      /* ト */
90          timechar[8]  = 0xde;      /* "  */
91          timechar[9]  = SPACE;     /*    */
92          timechar[10] = SPACE;     /*    */
93          timechar[11] = SPACE;     /*    */
94          timechar[12] = SPACE;     /*    */
95          timechar[13] = SPACE;     /*    */
96          timechar[14] = SPACE;     /*    */
97          timechar[15] = SPACE;     /*    */
98
99          /* LCD 二行目 */
100
101         timechar[16] = SPACE;     /*    */
```

```
102         timechar[17] = SPACE;    /*      */
103         timechar[18] = SPACE;    /*      */
104         timechar[19] = SPACE;    /*      */
105         timechar[20] = SPACE;    /*      */
106         timechar[21] = SPACE;    /*      */
107         timechar[22] = SPACE;    /*      */
108         timechar[23] = 0x2a;     /* *    */
109         timechar[24] = 0xc4;     /* ト   */
110         timechar[25] = 0xde;     /* "    */
111         timechar[26] = 0xb3;     /* ウ   */
112         timechar[27] = 0xbb;     /* サ   */
113         timechar[28] = 0xc1;     /* チ   */
114         timechar[29] = 0xad;     /* ュ   */
115         timechar[30] = 0xb3;     /* ウ   */
116         timechar[31] = 0x2a;     /* *    */
117
118         return;
119     }
```

【 lcdstpw.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :  lcdstpw.c
   Target              :  V853
   Procedure Defined   :  LCDStopWatchChar ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"


/***  関数宣言  ***/
void   LCDStopWatchChar1 (char *);
void   LCDStopWatchChar10 (char *);

/*****************************************************************************
   種　類　: 関数
   関数名　: LCDStopWatchChar1 ()
   機　能　: ストップウォッチ表示時の LCD 表示内容定義
               [1/1 秒モードの時]
   引き数　: 文字列格納先の先頭アドレス
   返り値　: なし
*****************************************************************************/
void
LCDStopWatchChar1 (char *timechar)
{

    /* LCD 一行目 */

    timechar[0]  = 0xbd;      /* ス */
    timechar[1]  = 0xc4;      /* ト */
    timechar[2]  = 0xaf;      /* ッ */
    timechar[3]  = 0xcc;      /* フ */
    timechar[4]  = 0xdf;      /* 。 */
    timechar[5]  = 0xb3;      /* ウ */
    timechar[6]  = 0xab;      /* ォ */
    timechar[7]  = 0xaf;      /* ッ */
    timechar[8]  = 0xc1;      /* チ */
    timechar[9]  = SPACE;     /*    */
    timechar[10] = 0xd3;      /* モ */
    timechar[11] = 0xb0;      /* ー */
    timechar[12] = 0xc4;      /* ト */
    timechar[13] = 0xde;      /* "  */
    timechar[14] = SPACE;     /*    */
    timechar[15] = SPACE;     /*    */

    /* LCD 二行目 */
```

```
49
50      timechar[16] = SPACE;    /*      */
51      timechar[17] = SPACE;    /*      */
52      timechar[18] = SPACE;    /*      */
53      timechar[19] = SPACE;    /*      */
54      timechar[20] = SPACE;    /*      */
55      timechar[21] = SPACE;    /*      */
56      timechar[22] = SPACE;    /*      */
57      timechar[22] = '1';      /* 1    */
58      timechar[23] = '/';      /* /    */
59      timechar[24] = '1';      /* 1    */
60      timechar[25] = SPACE;    /*      */
61      timechar[26] = SPACE;    /*      */
62      timechar[27] = 0xd3;     /* モ */
63      timechar[28] = 0xb0;     /* ー */
64      timechar[29] = 0xc4;     /* ト */
65      timechar[30] = 0xde;     /* "    */
66      timechar[31] = SPACE;    /*      */
67
68      return;
69  }
70
71  /*****************************************************************************
72      種　類　: 関数
73      関数名　: LCDStopWatchChar10 ()
74      機　能　: ストップウォッチ表示時の LCD 表示内容定義
75                  [1/10 秒モードの時]
76      引き数　: 文字列格納先の先頭アドレス
77      返り値　: なし
78  *****************************************************************************/
79  void
80  LCDStopWatchChar10 (char *timechar)
81  {
82
83      /* LCD 一行目 */
84
85      timechar[0]  = 0xbd;     /* ス */
86      timechar[1]  = 0xc4;     /* ト */
87      timechar[2]  = 0xaf;     /* ッ */
88      timechar[3]  = 0xcc;     /* フ */
89      timechar[4]  = 0xdf;     /* 。  */
90      timechar[5]  = 0xb3;     /* ウ */
91      timechar[6]  = 0xab;     /* ォ */
92      timechar[7]  = 0xaf;     /* ッ */
93      timechar[8]  = 0xc1;     /* チ */
94      timechar[9]  = SPACE;    /*      */
95      timechar[10] = 0xd3;     /* モ */
96      timechar[11] = 0xb0;     /* ー */
97      timechar[12] = 0xc4;     /* ト */
98      timechar[13] = 0xde;     /* "    */
99      timechar[14] = SPACE;    /*      */
```

```
100         timechar[15] = SPACE;     /*       */
101
102         /* LCD  二行目 */
103
104         timechar[16] = SPACE;     /*       */
105         timechar[17] = SPACE;     /*       */
106         timechar[18] = SPACE;     /*       */
107         timechar[19] = SPACE;     /*       */
108         timechar[20] = SPACE;     /*       */
109         timechar[21] = SPACE;     /*       */
110         timechar[22] = SPACE;     /*       */
111         timechar[22] = '1';       /* 1     */
112         timechar[23] = '/';       /* /     */
113         timechar[24] = '1';       /* 1     */
114         timechar[25] = '0';       /* 0     */
115         timechar[26] = SPACE;     /*       */
116         timechar[27] = 0xd3;      /*  モ   */
117         timechar[28] = 0xb0;      /*  ー   */
118         timechar[29] = 0xc4;      /*  ト   */
119         timechar[30] = 0xde;      /* "     */
120         timechar[31] = SPACE;     /*       */
121
122         return;
123     }
```

【 lcdslot.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :  lcdslot.c
   Target              :  V853
   Procedure Defined   :  LCDSlotChar ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"


/***  関数宣言  ***/
void   LCDSlotChar (char *);

/*****************************************************************************
   種　類　: 関数
   関数名　: LCDSlotChar ()
   機　能　: スロットマシン表示時の LCD 表示内容定義
   引き数　: 文字列格納先の先頭アドレス
   返り値　: なし
*****************************************************************************/
void
LCDSlotChar (char *timechar)
{

    /* LCD  一行目 */

    timechar[0]  = 0xbd;      /*  ス  */
    timechar[1]  = 0xdb;      /*  ロ  */
    timechar[2]  = 0xaf;      /*  ッ  */
    timechar[3]  = 0xc4;      /*  ト  */
    timechar[4]  = 0xcf;      /*  マ  */
    timechar[5]  = 0xbc;      /*  シ  */
    timechar[6]  = 0xdd;      /*  ン  */
    timechar[7]  = SPACE;     /*      */
    timechar[8]  = 0xd3;      /*  モ  */
    timechar[9]  = 0xb0;      /*  ー  */
    timechar[10] = 0xc4;      /*  ト  */
    timechar[11] = 0xde;      /*  "   */
    timechar[12] = SPACE;     /*      */
    timechar[13] = SPACE;     /*      */
    timechar[14] = SPACE;     /*      */
    timechar[15] = SPACE;     /*      */

    /* LCD  二行目 */

    timechar[16] = SPACE;     /*      */
```

```
49          timechar[17] = SPACE;    /*      */
50          timechar[18] = SPACE;    /*      */
51          timechar[19] = SPACE;    /*      */
52          timechar[20] = SPACE;    /*      */
53          timechar[21] = SPACE;    /*      */
54          timechar[22] = SPACE;    /*      */
55          timechar[23] = SPACE;    /*      */
56          timechar[24] = SPACE;    /*      */
57          timechar[25] = SPACE;    /*      */
58          timechar[26] = SPACE;    /*      */
59          timechar[27] = SPACE;    /*      */
60          timechar[28] = SPACE;    /*      */
61          timechar[29] = SPACE;    /*      */
62          timechar[30] = SPACE;    /*      */
63          timechar[31] = SPACE;    /*      */
64
65          return;
66      }
67
```

【 lcdcond.c 】

```
1   /*****************************************************************************
2       Realcond OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   lcdcond.c
5       Target              :   V853
6       Procedure Defined   :   LCDcondChar ()
7                               ConvertAscii ()
8   *****************************************************************************/
9   /*** ヘッダファイル ***/
10  #include   "sample.h"
11  #include   "common.h"
12
13
14  /*** 関数宣言 ***/
15  void   LCDConditionChar (char *, char, char);
16  void   ConvertAscii (char, char *);
17
18
19  /*****************************************************************************
20      種　類　: 関数
21      関数名　: LCDcondChar ()
22      機　能　: 現在時刻表示時の LCD 表示内容定義
23      引き数　: 文字列格納先の先頭アドレス、気温値、湿度値
24      返り値　: なし
25  *****************************************************************************/
26  void
27  LCDConditionChar (char *condchar, char temperature, char humidity)
28  {
29
30      char   tempdata[3];
31      char   humidata[3];
32
33      ConvertAscii (temperature, tempdata);
34      ConvertAscii (humidity, humidata);
35
36      /* LCD 一行目 */
37
38      condchar[0]  = 0xc0;        /* タ */
39      condchar[1]  = 0xc0;        /* タ */
40      condchar[2]  = 0xde;        /* "  */
41      condchar[3]  = 0xb2;        /* イ */
42      condchar[4]  = 0xcf;        /* マ */
43      condchar[5]  = 0xc9;        /* ノ */
44      condchar[6]  = SPACE;       /*    */
45      condchar[7]  = 0xb7;        /* キ */
46      condchar[8]  = 0xb5;        /* オ */
47      condchar[9]  = 0xdd;        /* ン */
48      condchar[10] = SPACE;       /*    */
```

```
49      condchar[11] = tempdata[0];  /* (百)*/
50      condchar[12] = tempdata[1];  /* (十)*/
51      condchar[13] = tempdata[2];  /* (一)*/
52      condchar[14] = 0xdf;         /* 。  */
53      condchar[15] = 0x43;         /* C   */
54
55      /* LCD 二行目 */
56
57      condchar[16] = SPACE;        /*     */
58      condchar[17] = SPACE;        /*     */
59      condchar[18] = SPACE;        /*     */
60      condchar[19] = SPACE;        /*     */
61      condchar[20] = SPACE;        /*     */
62      condchar[21] = SPACE;        /*     */
63      condchar[22] = SPACE;        /*     */
64      condchar[23] = 0xbc;         /* シ  */
65      condchar[24] = 0xc2;         /* ツ  */
66      condchar[25] = 0xc4;         /* ト  */
67      condchar[26] = 0xde;         /* "   */
68      condchar[27] = humidata[0];  /* (百)*/
69      condchar[28] = humidata[1];  /* (十)*/
70      condchar[29] = humidata[2];  /* (一)*/
71      condchar[30] = 0x25;         /* %   */
72      condchar[31] = SPACE;        /*     */
73
74      return;
75  }
76
77  /*****************************************************************************
78      種　類　: 関数
79      関数名　: ConvertAscii ()
80      機　能　: 値を各桁ごとに ASCII コードに変換する
81      引き数　: 変換する値、変換したコードを格納する領域のアドレス
82      返り値　: 格納した領域のポインタ
83  *****************************************************************************/
84  void
85  ConvertAscii (char data, char *p_data)
86  {
87
88      if (data == 100) {       /* データが 100 の時       */
89
90          p_data[0] = 0x31;    /*   1  */
91          p_data[1] = 0x30;    /*   0  */
92          p_data[2] = 0x30;    /*   0  */
93
94      } else if ((data >= 10)&&(data < 100)) {
95                               /* データが 10 から 99 の時 */
96
97          p_data[0] = SPACE;           /*     */
98          p_data[1] = 0x30+(data/10);  /* (十)*/
99          p_data[2] = 0x30+(data%10); /* (一)*/
```

```
100
101         } else if (data <= 9) {
102                                 /* データが 9 以下の時     */
103
104             p_data[0] = SPACE;              /*      */
105             p_data[1] = SPACE;              /*      */
106             p_data[2] = 0x30+(data%10); /* (一)*/
107         }
108         return;
109     }
```

【 codeman.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   codeman.c
5       Target              :   V853
6       Procedure Defined   :   CodeManager ()
7   *****************************************************************************/
8   /*** ヘッダファイル ***/
9   #include   "sample.h"
10  #include   "common.h"
11
12
13  /*** タスク・関数宣言 ***/
14  int CodeManager (char, char *, int);
15
16  /*****************************************************************************
17      種　類　: 関数
18      関数名　: CodeManager ()
19      機　能　: コード別に処理を分岐する
20      引　数　: コード (8 bit)
21                    文字コードを格納する配列のポインタ
22                    現在の LCD カーソルポジション
23      返り値　: 更新されたカーソルポジション
24  *****************************************************************************/
25  int
26  CodeManager (char code, char *memochar, int cur_pos)
27  {
28
29      int     pre_pos;  /* カーソル位置保存変数  */
30
31      /*****************************************************************
32          使用している LCD (M1632-0A) は 16 文字 × 2 行の表示が可能
33          で、内蔵された RAM には 40 文字 × 2 行分を記憶することが
34          できます。
35          今回は LCD のスクロールを考えず、単に 16 文字 × 2 行の範囲
36          内を対象とした操作にしてあります。
37      *****************************************************************/
38
39      pre_pos = cur_pos;      /* カーソルの位置を保存  */
40
41      if (code == LEFT) {     /* 左ボタンが押された時  */
42
43          if (pre_pos == 0) {             /* カーソルが 1 行目先頭だった時 */
44
45              LCDDataSet (LCD_LINE2E, OFF);
46                              /* LCD 操作 : カーソルを 2 行目の最後尾へ移動  */
47              cur_pos = 31;   /* 配列操作 : カーソル位置 32 文字目           */
48
```

```
49          } else if (pre_pos == 16) {     /* カーソルが 1 行目最後尾だった時  */
50
51              LCDDataSet (LCD_LINE1E, OFF);
52                          /* LCD 操作 : カーソルを 1 行目の最後尾へ移動  */
53              cur_pos--;          /* 配列操作: カーソル位置 16 文字目              */
54
55          } else {
56
57              LCDDataSet (LCD_LEFT, OFF);
58                          /* LCD 操作 : カーソルを左に 1 つ移動      */
59              cur_pos--;          /* 配列操作: カーソル位置をデクリメント   */
60          }
61
62      } else if (code == RIGHT) {   /* 右ボタンが押された時   */
63
64          if (pre_pos == 31) {          /* カーソルが 2 行目最後尾だった時  */
65
66              LCDDataSet (LCD_LINE1S, OFF);
67                          /* LCD 操作 : カーソルを 1 行目の先頭へ移動 */
68              cur_pos = 0;        /* 配列操作 : カーソル位置 1 文字目              */
69
70          } else if (pre_pos == 15) {     /* カーソルが 1 行目最後尾だった時  */
71
72              LCDDataSet (LCD_LINE2S, OFF);
73                          /* LCD 操作 : カーソルを 2 行目の先頭へ移動 */
74              cur_pos++;          /* 配列操作: カーソル位置 17 文字目              */
75
76          } else {
77
78              LCDDataSet (LCD_RIGHT, OFF);
79                          /* LCD 操作 : カーソルを 1 行目へ移動      */
80              cur_pos++;          /* 配列操作: カーソル位置をインクリメント */
81          }
82
83      } else if ((code == UP)||(code == DOWN)) {
84                                  /* 上または下ボタンが押された時   */
85
86          if (pre_pos <= 15) {          /* 1 行目にカーソルがある時          */
87
88              LCDDataSet (LCD_LINE2S + pre_pos, OFF);
89                          /* LCD 操作 : カーソルを 2 行目へ移動      */
90              cur_pos = 16 + pre_pos;
91                          /* 配列操作: 配列番号に 16 を加える          */
92
93          } else if (pre_pos >= 16) {     /* 2 行目にカーソルがある時      */
94
95              LCDDataSet (LCD_LINE1S + (pre_pos-16), OFF);
96                          /* LCD 操作 : カーソルを 1 行目へ移動      */
97              cur_pos = pre_pos - 16;
98                          /* 配列操作: 配列番号から 16 を引く          */
99          }
```

```
100
101     } else if (code == CR) {    /* RETURN (ENTER) が押された時      */
102
103         if (pre_pos <= 15) {    /* カーソルが 1 行目にある時          */
104
105             LCDDataSet (LCD_LINE2S, OFF);
106                         /* LCD 操作 : カーソルを 2 行目先頭へ移動    */
107             cur_pos = 16;   /* 配列操作: カーソル位置 17 文字目          */
108
109         } else if (pre_pos >= 16) {     /* カーソルが 2 行目にある時   */
110
111             LCDDataSet (LCD_LINE1S, OFF);
112                         /* LCD 操作 : カーソルを 1 行目先頭へ移動    */
113             cur_pos = 0;    /* 配列操作 : カーソル位置 1 文字目          */
114         }
115
116     } else if (code == BS) {    /* BS が押された時       */
117
118         LCDDataSet (SPACE, ON);
119                     /* LCD 操作 - 1 : スペース出力で文字を削除*/
120
121         /* LCD 操作 - 2 : カーソルを 2 個戻す    */
122
123         if (pre_pos == 0) {         /* カーソルが 1 行目先頭だった時 */
124
125             LCDDataSet (LCD_LINE2E, OFF);
126                         /* LCD 操作 - カーソルを 2 行目最後尾へ戻す */
127             cur_pos = 31;   /* 配列操作 : カーソル位置 32 文字目          */
128
129         } else if (pre_pos == 16) {     /* カーソルが 2 行目先頭だった時 */
130
131             LCDDataSet (LCD_LINE1E, OFF);
132                         /* LCD 操作 - カーソルを 1 行目最後尾へ戻す */
133             cur_pos--;      /* 配列操作: カーソル位置 16 文字目          */
134
135         } else {        /* カーソルが上の 2 つ以外にあった時    */
136
137             /* LCD 操作 - カーソルを 2 文字分戻す  */
138
139             LCDDataSet (LCD_LEFT, OFF);
140                         /* カーソルを 1 行目最後尾へ移動             */
141             LCDDataSet (LCD_LEFT, OFF);
142                         /* カーソルを 1 行目最後尾へ移動             */
143             cur_pos--;  /* 配列操作 : カーソル位置をデクリメント  */
144         }
145
146         memochar[cur_pos] = SPACE;
147                     /* 配列操作 : カーソル部を消す */
148
149     } else if ((code == DELETE)||(code == DELKEY)) {
150                                 /* DELETE が押された時       */
```

```
151
152             LCDDataSet (SPACE, ON);
153                         /* LCD 操作 - 1 : スペース出力で文字を削除 */
154             LCDDataSet (LCD_LEFT, OFF);
155                         /* LCD 操作 - 2 : カーソルを 1 個戻す       */
156             memochar[cur_pos] = SPACE;
157                         /* 配列操作: カーソル部を消す               */
158
159         } else if ((code >= 0x20)&&(code <= 0xdf)) {
160                                     /* 文字キーが押された時       */
161
162             LCDDataSet (code, ON);
163                         /* LCD 操作 - 1 : スペース出力で文字を削除 */
164
165             if (pre_pos == 15) {/* カーソルが 1 行目最後尾だった時  */
166
167                 LCDDataSet (LCD_LINE2S, OFF);
168                             /* LCD 操作 : カーソルを 2 行目先頭へ移動   */
169                 memochar[cur_pos] = code;
170                             /* 配列操作: コードを代入                   */
171
172                 cur_pos++;      /* カーソルポジション インクリメント   */
173
174             } else if (pre_pos == 31) {
175                             /* カーソルが 2 行目最後尾だった時      */
176
177                 LCDDataSet (LCD_LINE1S, OFF);
178                             /* LCD 操作 : カーソルを 2 行目先頭へ移動   */
179                 memochar[cur_pos] = code;
180                             /* 配列操作: コードを代入       */
181                 cur_pos = 0; /* カーソルポジション 先頭へ   */
182
183             } else {
184
185                 memochar[cur_pos] = code;
186                             /* 配列操作 : コードを代入                  */
187                 cur_pos++;      /* カーソルポジション インクリメント   */
188             }
189             set_flg (FLG_MEMOCHECK, code);
190                             /* シリアル受信した結果を、送信側に報告   */
191         }
192         return (cur_pos);
193     }
```

【 memoctrl.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   memoctrl.c
5       Target              :   V853
6       Procedure Defined   :   MemoDrive ()
7                               MemoControl ()
8   *****************************************************************************/
9   /*** ヘッダファイル ***/
10  #include   "sample.h"
11  #include   "common.h"
12
13
14  /*** タスク・関数宣言 ***/
15  INT         MemoDrive (void);
16  void        MemoControl (INT);
17
18  /*****************************************************************************
19      種　類　: 間接割り込みハンドラ
20      関数名　: MemoDrive ()
21      機　能　: メモ機能 ドライバ (シリアル受信ドライバ)
22                  受信完了割り込み受信後、起動
23      引き数　: なし
24      返り値　: 0xff (間接割り込みハンドラから抜ける)
25  *****************************************************************************/
26  INT
27  MemoDrive (void)
28  {
29      wup_tsk (TSK_MEMOCONTROL);      /* メモ管理タスク起床 */
30
31      outpb (ASIM00, (inpb (ASIM00) & ~RCV_ENABLE));
32                                      /* シリアル受信禁止     */
33      outpb (ASIM00, (inpb (ASIM00) | SND_ENABLE));
34                                      /* シリアル送信許可     */
35      return (0xff);
36  }
37
38  /*****************************************************************************
39      種　類　: タスク
40      関数名　: MemoControl ()     [ID : TSK_MEMOCONTROL]
41      機　能　: メモ管理タスク
42      引　数　: 初期情報 (起動時のタスク引き渡し情報)
43      返り値　: なし
44  *****************************************************************************/
45  void
46  MemoControl (INT stacd)
47  {
48
49      for ( ; ; ) {
```

```
50
51          slp_tsk ();     /* 自タスク起床待ち */
52
53          switch (flg_memodisp) {         /* メモ表示フラグによって場合分け     */
54
55              /**********************
56                  すべて消去の時
57              **********************/
58              case    MEMO_ALLCLEAR :
59
60                  outpb (ASIM00, (inpb (ASIM00) & ~RCV_ENABLE));
61                                          /* シリアル受信禁止          */
62                  MemoManager (MEMO_ALLCLEAR);
63                                          /* メモ管理関数呼び出し    */
64                  break;
65
66              /*******************
67                  再表示の時
68              *******************/
69              case    MEMO_ALLDISP :
70
71                  outpb (ASIM00, (inpb (ASIM00) & ~RCV_ENABLE));
72                                          /* シリアル受信禁止          */
73                  MemoManager (MEMO_ALLDISP);
74                                          /* メモ管理関数呼び出し    */
75                  break;
76
77              /**********************
78                  メモ書き込み時
79              **********************/
80              case    MEMO_WRITING :
81
82                  MemoManager (MEMO_WRITING);
83                                          /* メモ管理関数呼び出し    */
84                  break;
85          }
86          outpb (ASIM00, (inpb (ASIM00) | RCV_ENABLE));
87                                          /* シリアル受信許可          */
88      }
89  }
```

【 memoinit.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :   memoinit.c
   Target              :   V853
   Procedure Defined   :   MemoInit ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"


/***  関数宣言  ***/
void   MemoInit (char *);

/*****************************************************************************
   種　類　: 関数
   関数名　: MemoInit ()
   機　能　: メモ初期値を代入する
   引き数　: 文字列格納先の先頭アドレス
   返り値　: なし
*****************************************************************************/
void
MemoInit (char *initchar)
{

   /* LED  一行目 */

   initchar[0]  = SPACE;    /*      */
   initchar[1]  = 0x52;     /* R    */
   initchar[2]  = 0x58;     /* X    */
   initchar[3]  = 0x38;     /* 8    */
   initchar[4]  = 0x35;     /* 5    */
   initchar[5]  = 0x30;     /* 0    */
   initchar[6]  = SPACE;    /*      */
   initchar[7]  = SPACE;    /*      */
   initchar[8]  = SPACE;    /*      */
   initchar[9]  = SPACE;    /*      */
   initchar[10] = SPACE;    /*      */
   initchar[11] = SPACE;    /*      */
   initchar[12] = SPACE;    /*      */
   initchar[13] = SPACE;    /*      */
   initchar[14] = SPACE;    /*      */
   initchar[15] = SPACE;    /*      */

   /* LED  二行目 */

   initchar[16] = SPACE;    /*      */
   initchar[17] = SPACE;    /*      */
```

```
50          initchar[18] = 0xb1;     /* ｱ */
51          initchar[19] = 0xcc;     /* ﾌ */
52          initchar[20] = 0xdf;     /* ｡ */
53          initchar[21] = 0xd8;     /* ﾘ */
54          initchar[22] = 0xb9;     /* ｹ */
55          initchar[23] = 0xb0;     /* ｰ */
56          initchar[24] = 0xbc;     /* ｼ */
57          initchar[25] = 0xae;     /* ｮ */
58          initchar[26] = 0xdd;     /* ﾝ */
59          initchar[27] = SPACE;    /*   */
60          initchar[28] = 0xc9;     /* ﾉ */
61          initchar[29] = 0xb0;     /* ｰ */
62          initchar[30] = 0xc4;     /* ﾄ */
63          initchar[31] = SPACE;    /*   */
64
65          return;
66      }
```

【 memoset.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :   memoset.c
   Target              :   V853
   Procedure Defined   :   MemoManager ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"


/***  タスク・関数宣言  ***/
void   MemoManager (int);

/*****************************************************************************
   種　類　: 関数
   関数名　: MemoManager ()
   機　能　: メモ管理関数
   引　数　: コマンド (メモクリア・記憶されたメモ表示・メモ書き込み)
             カーソルポジション
   返り値　: 操作後のカーソルポジション
*****************************************************************************/
void
MemoManager (int command)
{

    struct  LCDMAIL   *p_msgblk;      /*  メッセージの領域            */
    static  char    memochar[32];     /*  メモ本体                    */
    static  int     cur_pos;          /* LCD  カーソルポジション     */

    switch (command) {

        /**********************
             すべて消去の時
        **********************/
        case    MEMO_ALLCLEAR :

            rcv_msg ((T_MSG **)&p_msgblk, MBX_FROM_LCD);
                                      /*  メッセージ領域を受け取る    */

            for (cur_pos = 0; cur_pos < 32; cur_pos++) {

                memochar[cur_pos] = SPACE;
                p_msgblk->data[cur_pos] = SPACE;

            }       /*  すべてにスペースコードを代入        */

            flg_memodisp = MEMO_WRITING;    /*  メモ書き込み状態へ移行     */
```

```
50              cur_pos = 0;                    /* カーソル行を初期値に        */
51              snd_msg (MBX_TO_LCD, (T_MSG *)p_msgblk);
52                                              /* LCD 表示タスクへメッセージ送信    */
53              break;
54
55          /*******************
56              再表示の時
57          *******************/
58          case    MEMO_ALLDISP :
59
60              rcv_msg ((T_MSG **)&p_msgblk, MBX_FROM_LCD);
61                              /* メッセージ領域を受け取る    */
62
63              for (cur_pos = 0; cur_pos < 32; cur_pos++) {
64
65                  p_msgblk->data[cur_pos] = memochar[cur_pos];
66
67              }       /* 記憶しているメモを読み出す  */
68
69              snd_msg (MBX_TO_LCD, (T_MSG *)p_msgblk);
70                                          /* LCD 表示タスクへメッセージ送信    */
71              flg_memodisp = MEMO_WRITING;    /* メモ書き込み状態へ移行    */
72              cur_pos = 0;                    /* カーソル行を初期値に        */
73
74              break;
75
76          /*********************
77              メモ書き込み時
78          *********************/
79          case    MEMO_WRITING :
80
81              cur_pos = CodeManager (inpb (RXB0L), memochar, cur_pos);
82                              /* 受信したメッセージによる処理分け */
83              break;
84
85          /*******************
86              初期化の時
87          *******************/
88          case    MEMO_INIT :
89
90              MemoInit (memochar);/* メモ初期文字を代入  */
91              cur_pos = 0;            /* カーソル行を初期化  */
92
93              break;
94      }
95      return;
96  }
```

【 sndcheck.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   memoctrl.c
5       Target              :   V853
6       Procedure Defined   :   SendCheck ()
7                               UARTCharSend ()
8   *****************************************************************************/
9   /*** ヘッダファイル ***/
10  #include   "sample.h"
11  #include   "common.h"
12
13
14  /*** タスク・関数宣言 ***/
15  INT    SendCheck (void);
16  void   UARTCharSend (INT);
17
18  /*****************************************************************************
19      種　類　: 間接割り込みハンドラ
20      関数名　: SendCheck ()
21      機　能　: メモ送信結果確認ドライバ (シリアル送信ドライバ)
22                送信完了割り込み受信後、起動
23      引き数　: なし
24      返り値　: TSK_UARTSEND (送信タスクを起床して抜ける)
25  *****************************************************************************/
26  INT
27  SendCheck (void)
28  {
29      outpb (ASIM00, (inpb (ASIM00) & ~SND_ENABLE));
30                                          /* シリアル送信禁止 */
31      return (TSK_UARTSEND);          /* 送信タスク起床    */
32  }
33
34  /*****************************************************************************
35      種　類　: タスク
36      関数名　: UARTCharSend ()     [ID : TSK_UARTSEND]
37      機　能　: シリアル受信した結果を、送信側に報告
38      引　数　: 初期情報 (起動時のタスク引き渡し情報)
39      返り値　: なし
40  *****************************************************************************/
41  void
42  UARTCharSend (INT stacd)
43  {
44
45      unsigned int  sendchar;     /* 送信するコードを格納する領域のアドレス*/
46
47      for ( ; ; ) {
48
49          wai_flg (&sendchar, FLG_MEMOCHECK, 0xff, TWF_ORW|TWF_CLR);
```

```
50                                          /* シリアル受信した結果を、送信側に報告   */
51          outpb (TXS0L, sendchar);        /* UART 送信 */
52
53          slp_tsk ();     /* 自タスク起床待ち */
54                          /* 送信が完了したら起床 (SendCheck () によって起床)    */
55
56      }
57  }
```

【 inttakt1.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   inttakt1.c
5       Target              :   V853
6       Procedure Defined   :   TaktSwitch1 ()
7   *****************************************************************************/
8   /*** ヘッダファイル ***/
9   #include   "sample.h"
10  #include   "common.h"
11
12
13  /*** 関数宣言 ***/
14  INT    TaktSwitch1 (void);
15
16  /*****************************************************************************
17      種　類　: 間接割り込みハンドラ
18      関数名　: TaktSwitch1 ()
19      機　能　: タクトスイッチ 1 割り込みの処理
20                  ・現在時刻表示モード       → 時刻更正時に使用
21                  ・タイマー表示モード       → タイマー設定時に使用
22                  ・ストップウォッチモード → スタート・ストップ
23                  ・スロットマシンモード     → スロット停止
24      引き数　: なし
25      返り値　: 0xff (間接割り込みハンドラから抜ける)
26  *****************************************************************************/
27  INT
28  TaktSwitch1 (void)
29  {
30
31      if ((flg_interrupt & SW1) == SW1) {
32              /* タクトスイッチ 1 割り込みが許可状態であるかをチェック  */
33              /* 割り込みが受け付けられない時は、何もせずに抜ける          */
34
35          flg_figurefix = FIGURE_1;
36                      /* タクトスイッチ 1 の割り込みであることを通知 */
37
38          switch (flg_ledmode) {      /* LED 表示モードによる場合分け   */
39
40              /*****************************
41                  現在時刻表示モードの時
42               *****************************/
43              case    CURRENT_TIME:
44
45                  wup_tsk (TSK_INTCONTROL); /* 割り込み制御タスクに処理移行   */
46
47                  break;
48
```

```
49              /***************************
50                   タイマー表示モードの時
51               ***************************/
52              case    TIMER:
53
54                  wup_tsk (TSK_INTCONTROL); /* 割り込み制御タスクに処理移行   */
55
56                  break;
57
58              /************************************
59                   ストップウォッチ表示モードの時
60               ************************************/
61              case    STOPWATCH:
62
63                  if (flg_stopwatch == SWATCH_STOP) {
64                              /* ストップウォッチが動作していない時  */
65
66                      flg_stopwatch = SWATCH_START; /* スロット状態変更     */
67                      StopWatchAction (START);      /* スロット動作開始     */
68                      flg_interrupt = (SW1|SW2|SW5|SW6|SW7|SW8);
69                              /* タクトスイッチ割り込み 1,2,5,6,7,8 を許可    */
70                      sig_sem (SEM_LEDTIMING);
71                              /* LED 表示タイミング変数初期化用               */
72
73                  } else if (flg_stopwatch == SWATCH_WORKING) {
74                              /* ストップウォッチが動作している時 */
75
76                      flg_stopwatch = SWATCH_STOP;  /* スロット状態変更     */
77                      StopWatchAction (STOP);       /* スロット動作停止     */
78                      flg_interrupt = (SW1|SW3|SW5|SW6|SW7|SW8);
79                          /* タクトスイッチ割り込み 1,3,5,6,7,8 を許可    */
80                  }
81                  break;
82
83              /*********************************
84                   スロットマシン表示モードの時
85               *********************************/
86              case    SLOT:
87
88                  flg_slotfigure = flg_slotfigure | SLOT_FIGURE1;
89                                          /* 下一桁目ストップ命令  */
90                  flg_interrupt = flg_interrupt & ~(SW1);
91                                          /* タクトスイッチ割り込み 1 を禁止   */
92                  break;
93          }
94      }
95      return (0xff);
96  }
```

【 inttakt2.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :   inttakt2.c
   Target              :   V853
   Procedure Defined   :   TaktSwitch2 ()
*****************************************************************************/
/*** ヘッダファイル ***/
#include   "sample.h"
#include   "common.h"


/*** 関数宣言 ***/
INT    TaktSwitch2 (void);

/*****************************************************************************
   種　類　: 間接割り込みハンドラ
   関数名　: TaktSwitch2 ()
   機　能　: タクトスイッチ 2 割り込みの処理
               ・現在時刻表示モード       → 時刻更正時に使用
               ・タイマー表示モード       → タイマー設定時に使用
               ・ストップウォッチモード → ラップ表示 ON/OFF
               ・スロットマシンモード     → スロット停止
   引き数　: なし
   返り値　: 0xff ( 間接割り込みハンドラから抜ける )
*****************************************************************************/
INT
TaktSwitch2 (void)
{

    if ((flg_interrupt & SW2) == SW2) {
            /* タクトスイッチ 2 割り込みが許可状態であるかをチェック */
            /* 割り込みが受け付けられない時は、何もせずに抜ける       */

        flg_figurefix = FIGURE_2;
                    /* タクトスイッチ 2 の割り込みであることを通知 */

        switch (flg_ledmode) {     /* LED 表示モードによる場合分け */

            /*****************************
                現在時刻表示モードの時
             *****************************/
            case   CURRENT_TIME:

                wup_tsk (TSK_INTCONTROL);     /* 割り込み制御タスクに処理移行 */

                break;

```

```
49              /***************************
50                   タイマー表示モードの時
51               ***************************/
52              case    TIMER:
53
54                  wup_tsk (TSK_INTCONTROL);     /* 割り込み制御タスクに処理移行   */
55
56                  break;
57
58              /************************************
59                   ストップウォッチ表示モードの時
60               ************************************/
61              case    STOPWATCH:
62
63                  if (flg_laptime == OFF) {
64                                  /*   ラップタイムモードではない時 */
65
66                      flg_laptime = ON;   /* ラップタイム表示モード ON   */
67                      flg_interrupt = (SW2|SW7|SW8);
68                              /* タクトスイッチ割り込み 2,7,8 を許可   */
69
70                  } else if (flg_laptime == ON) {
71                                  /*   ラップタイムモードの時   */
72
73                      StopWatchCount (DISPLAY, OFF);
74                                          /* 保持されている値を LED に表示*/
75                      flg_laptime = OFF;      /* ラップタイム表示モード OFF      */
76                      flg_interrupt = (SW1|SW2|SW5|SW7|SW8);
77                                  /* タクトスイッチ割り込み 1,2,5,7,8 を許可 */
78                  }
79                  break;
80
81              /**********************************
82                   スロットマシン表示モードの時
83               **********************************/
84              case    SLOT:
85
86                  flg_slotfigure = flg_slotfigure | SLOT_FIGURE2;
87                                              /* 下二桁目ストップ命令 */
88                  flg_interrupt = flg_interrupt & ~(SW2);
89                                      /* タクトスイッチ割り込み 2 を禁止   */
90                  break;
91          }
92      }
93      return (0xff);
94  }
```

【 inttakt3.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   inttakt3.c
5       Target              :   V853
6       Procedure Defined   :   TaktSwitch3 ()
7   *****************************************************************************/
8   /*** ヘッダファイル ***/
9   #include   "sample.h"
10  #include   "common.h"
11
12
13  /*** 関数宣言 ***/
14  INT    TaktSwitch3 (void);
15
16  /*****************************************************************************
17      種　類　: 間接割り込みハンドラハンドラ
18      関数名　: TaktSwitch3 ()
19      機　能　: タクトスイッチ 3 割り込みの処理
20                  ・現在時刻表示モード       → 時刻更正時に使用
21                  ・タイマー表示モード       → タイマー設定時に使用
22                  ・ストップウォッチモード → リセットスイッチ
23                  ・スロットマシンモード     → スロット停止
24      引き数　: なし
25      返り値　: 0xff (間接割り込みハンドラから抜ける)
26  *****************************************************************************/
27  INT
28  TaktSwitch3 (void)
29  {
30      if ((flg_interrupt & SW3) == SW3) {
31              /* タクトスイッチ 3 割り込みが許可状態であるかをチェック */
32              /* 割り込みが受け付けられない時は、何もせずに抜ける       */
33
34          flg_figurefix = FIGURE_3;
35                      /* タクトスイッチ 3 の割り込みであることを通知 */
36
37          switch (flg_ledmode) {     /* LED 表示モードによる場合分け */
38
39              /*****************************
40                  現在時刻表示モードの時
41               *****************************/
42              case   CURRENT_TIME:
43
44                  wup_tsk (TSK_INTCONTROL);     /* 割り込み制御タスクに処理移行 */
45
46                  break;
47
48              /*****************************
```

```
49                  タイマー表示モードの時
50              ****************************/
51              case    TIMER:
52
53                  wup_tsk (TSK_INTCONTROL);     /* 割り込み制御タスクに処理移行   */
54
55                  break;
56
57              /***********************************
58                  ストップウォッチ表示モードの時
59              ************************************/
60              case    STOPWATCH:
61
62                  StopWatchCount (INIT, OFF);         /*   リセット   */
63
64                  break;
65
66              /***********************************
67                  スロットマシン表示モードの時
68              ************************************/
69              case    SLOT:
70
71                  flg_slotfigure = flg_slotfigure | SLOT_FIGURE3;
72                                                  /* 下三桁目ストップ命令 */
73                  flg_interrupt = flg_interrupt & ~(SW3);
74                                      /* タクトスイッチ割り込み 3 を禁止   */
75                  break;
76          }
77      }
78      return (0xff);
79  }
```

【 inttakt4.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :   inttakt4.c
   Target              :   V853
   Procedure Defined   :   TaktSwitch4 ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"


/***  関数宣言  ***/
INT    TaktSwitch4 (void);

/*****************************************************************************
   種　類　: 間接割り込みハンドラ
   関数名　: TaktSwitch4 ()
   機　能　: タクトスイッチ 4 割り込みの処理
               ・現在時刻表示モード       → 時刻更正時に使用
               ・タイマー表示モード       → タイマー設定時に使用
               ・ストップウォッチモード → 未使用
               ・スロットマシンモード     → スロット回転
   引き数　: なし
   返り値　: 0xff (  間接割り込みハンドラから抜ける  )
*****************************************************************************/
INT
TaktSwitch4 (void)
{
    if ((flg_interrupt & SW4) == SW4) {
                /*  タクトスイッチ 4 割り込みが許可状態であるかをチェック  */
                /*  割り込みが受け付けられない時は、何もせずに抜ける          */

        flg_figurefix = FIGURE_4;
                        /*  タクトスイッチ 4 の割り込みであることを通知 */

        switch (flg_ledmode) {      /* LED 表示モードによる場合分け   */

            /*****************************
                 現在時刻表示モードの時
             *****************************/
            case    CURRENT_TIME:

                wup_tsk (TSK_INTCONTROL); /*  割り込み制御タスクに処理移行   */

                break;

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

```
49                  タイマー表示モードの時
50                 ****************************/
51              case    TIMER:
52
53                  wup_tsk (TSK_INTCONTROL);     /* 割り込み制御タスクに処理移行   */
54
55                  break;
56
57              /***********************************
58                  ストップウォッチ表示モードの時
59                 ***********************************/
60              case    STOPWATCH:
61
62                  break;
63
64              /***********************************
65                  スロットマシン表示モードの時
66                 ***********************************/
67              case    SLOT:
68
69                  flg_slot = SLOT_WORKING;     /* スロット状態変更              */
70                  flg_interrupt = (SW1|SW2|SW3|SW5|SW7|SW8);
71                                  /* タクトスイッチ割り込み 1,2,3,5,7,8 を許可   */
72                  wup_tsk (TSK_INTCONTROL); /* 割り込み制御タスクに処理移行   */
73
74                  break;
75          }
76      }
77      return (0xff);
78  }
```

【 inttakt5.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   inttakt5.c
5       Target              :   V853
6       Procedure Defined   :   TaktSwitch5 ()
7   *****************************************************************************/
8   /*** ヘッダファイル ***/
9   #include   "sample.h"
10  #include   "common.h"
11
12
13  /*** 関数宣言 ***/
14  INT    TaktSwitch5 (void);
15
16  /*****************************************************************************
17      種　類　: 間接割り込みハンドラ
18      関数名　: TaktSwitch5 ()
19      機　能　: タクトスイッチ 5 割り込みの処理
20                  ・表示モード切り替え
21      引き数　: なし
22      返り値　: 0xff (間接割り込みハンドラから抜ける)
23  *****************************************************************************/
24  INT
25  TaktSwitch5 (void)
26  {
27      if ((flg_interrupt & SW5) == SW5) {
28                  /* タクトスイッチ 5 割り込みが許可状態であるかをチェック */
29                  /* 割り込みが受け付けられない時は、何もせずに抜ける       */
30
31          /*********************************************************
32              ・LCD に気温・湿度が表示されている時
33                  自動表示タスクは sleep 状態なので時限待ち状態へ移行
34              ・LCD に気温・湿度が表示されていない時
35                  自動表示タスクは時限待ちなので、一度時限待ちを解除
36                  してから、もう一度時限待ちにする
37          *********************************************************/
38          if (flg_condition == ON)   flg_condition = OFF;
39                              /* 気温・湿度表示が ON の時は OFF にする   */
40
41          wup_tsk (TSK_AUTOCHANGE);   /* 自動表示タスクに起床要求 */
42
43          switch (flg_ledmode) {      /* LED 表示モードによる場合分け  */
44
45              /*****************************
46                  現在時刻表示モードの時
47               *****************************/
48              case    CURRENT_TIME :
```

```
49
50              flg_ledmode = TIMER;            /* タイマー表示モードに変更 */
51              wup_tsk (TSK_LCDCHAR);      /* LCD 表示タスク起床          */
52
53              /*************************************
54                   タイマーが動作していない時に
55                   タイマー表示モードに切り替わった時
56                     ・初期値 [00:00] を表示するように
57                       TimerCountDown ()を呼ぶ
58               *************************************/
59              if (flg_timer == TIMER_STOP) {
60
61                  flg_interrupt = ALLSW & ~(SW6);
62                              /* タクトスイッチ割り込み 6 を禁止  */
63                  SetTimerCount (CLEAR);          /* タイマー初期化   */
64                  TimerCountDown (INIT, OFF);
65
66              /*************************************
67                   タイマーが中断されたままの状態
68                   だった時
69                     ・中断された時の値を表示しておく
70                       ように TimerCountDown ()を呼ぶ
71               *************************************/
72              } else if (flg_timer == TIMER_INTERRUPT) {
73
74                  flg_interrupt = ALLSW & ~(SW6);
75                              /* タクトスイッチ割り込み 6 を禁止  */
76                  TimerCountDown (DISPLAY, OFF);
77                                      /* 中断された時の値を表示   */
78
79              /*************************************
80                   タイマーが動作中の時
81                     ・現在進行中のタイマー値を表示す
82                       るように TimerCountDown()を呼ぶ
83               *************************************/
84              } else if (flg_timer == TIMER_WORKING) {
85
86                  flg_interrupt = ALLSW & ~(SW1|SW2|SW3|SW4);
87                          /* タクトスイッチ割り込み 1,2,3,4 を禁止   */
88
89                  TimerCountDown (DISPLAY, OFF);/* 現在の値を表示   */
90
91              /*************************************
92                   タイマーが終了している時
93                     ・初期値 [00:00] を表示するように
94                       TimerCountDown()を呼ぶ
95               *************************************/
96              } else if (flg_timer == TIMER_FINISH) {
97
98                  flg_interrupt = ALLSW & ~(SW1|SW2|SW3|SW4|SW6);
99                          /* タクトスイッチ割り込み 1,2,3,4,6 を禁止 */
```

```
100                 TimerCountDown (INIT, OFF);       /* タイマー初期化    */
101               }
102               break;
103
104           /*****************************
105                 タイマー表示モードの時
106            *****************************/
107           case   TIMER :
108
109               flg_ledmode = STOPWATCH;
110                             /* ストップウォッチ表示モードに変更 */
111               wup_tsk (TSK_LCDCHAR);      /* LCD 表示タスク起床     */
112
113               /*****************************************
114                     ストップウォッチが動作していないとき
115                       初期値 [00:00] を表示するように
116                       StopWatchCount()を呼ぶ
117                *****************************************/
118               if (flg_stopwatch == SWATCH_STOP) {
119
120                   flg_interrupt = ALLSW & ~(SW2|SW3|SW4);
121                             /* タクトスイッチ割り込み 2,3,4 を禁止  */
122                   StopWatchCount (INIT, OFF);
123                                         /* ストップウォッチ初期化    */
124
125               /*****************************************
126                     ストップウォッチが動作中のとき
127                       現在のタイマー値を表示するように
128                       StopWatchCount()を呼ぶ
129                *****************************************/
130               } else if (flg_stopwatch == SWATCH_WORKING) {
131
132                   flg_interrupt = ALLSW & ~(SW3|SW4|SW6);
133                             /* タクトスイッチ割り込み 3,4,6 を禁止  */
134                   StopWatchCount (DISPLAY, OFF);      /* 現在値を表示   */
135               }
136               break;
137
138           /*************************************
139                 ストップウォッチ表示モードの時
140            *************************************/
141           case   STOPWATCH :
142
143               flg_ledmode = SLOT;               /* スロット表示モードに変更 */
144               wup_tsk (TSK_LCDCHAR);      /* LCD 表示タスク起床          */
145
146               /*****************************************
147                     スロットマシンが動作していないとき
148                       初期値 [ 7:77] を表示するように
149                       SlotRotation()を呼ぶ
150                *****************************************/
```

```
151                 if (flg_slot == SLOT_STOP) {
152
153                     flg_interrupt = ALLSW & ~(SW1|SW2|SW3|SW6);
154                             /* タクトスイッチ割り込み 1,2,3,6 を禁止    */
155                     SlotRotation (INIT, OFF);           /* スロット初期化    */
156
157                 /****************************************
158                       スロットマシンが動作中のとき
159                         現在の状態を表示するように
160                         SlotRotation()を呼ぶ
161                  ****************************************/
162                 } else if (flg_slot == SLOT_WORKING) {
163
164                     flg_interrupt = ALLSW & ~(SW4|SW6);
165                                 /* タクトスイッチ割り込み 4,6 を禁止*/
166                     SlotRotation (DISPLAY, OFF);        /* 現在値を表示      */
167                 }
168                 break;
169
170             /*********************************
171                   スロットマシン表示モードの時
172              *********************************/
173             case   SLOT :
174
175                 flg_ledmode = CURRENT_TIME;         /* 現在時刻表示モードに変更 */
176                 wup_tsk (TSK_LCDCHAR);              /* LCD 表示タスク起床        */
177
178                 if (flg_time == TIME_FIX) {
179                                     /* 現在時刻がまだ調整中だった時   */
180                     flg_interrupt = ALLSW;    /* 全割り込み許可              */
181                     SetTime(INIT);              /* 値を初期化しておく          */
182
183                 } else {                    /* それ以外の時   */
184
185                     flg_interrupt = (SW5|SW6|SW7|SW8);
186                             /* タクトスイッチ割り込み 5,6,7,8 を許可    */
187                 }
188                 TimeCount (DISPLAY, OFF);     /* 現在値を表示        */
189
190                 break;
191         }
192     }
193     return (0xff);
194 }
```

【 inttakt6.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   inttakt6.c
5       Target              :   V853
6       Procedure Defined   :   TaktSwitch6 ()
7   *****************************************************************************/
8   /*** ヘッダファイル ***/
9   #include   "sample.h"
10  #include   "common.h"
11
12  /*** 関数宣言 ***/
13  INT    TaktSwitch6 (void);
14
15  /*****************************************************************************
16      種　類　: 間接割り込みハンドラ
17      関数名　: TaktSwitch6 ()
18      機　能　: タクトスイッチ 6 割り込みの処理
19                  ・現在時刻表示モード       → 時刻更正モード ON/OFF
20                  ・タイマー表示モード       → タイマーON/OFF
21                  ・ストップウォッチモード → モード切り替え(1 秒 / 1/10 秒)
22                  ・スロットマシンモード     → 未使用
23      引き数　: なし
24      返り値　: 0xff (間接割り込みハンドラから抜ける)
25  *****************************************************************************/
26  INT
27  TaktSwitch6 (void)
28  {
29      if ((flg_interrupt & SW6) == SW6) {
30                  /* タクトスイッチ 6 割り込みが許可状態であるかをチェック */
31                  /* 割り込みが受け付けられない時は、何もせずに抜ける       */
32
33          /*************************************************************
34              ・LCD に気温・湿度が表示されている時
35                  自動表示タスクは sleep 状態なので時限待ち状態へ移行
36              ・LCD に気温・湿度が表示されていない時
37                  自動表示タスクは時限待ちなので、一度時限待ちを解除
38                  してから、もう一度時限待ちにする
39          *************************************************************/
40          if (flg_condition == ON)   flg_condition = OFF;
41                              /* 気温・湿度表示が ON の時は OFF にする   */
42
43          wup_tsk (TSK_AUTOCHANGE);   /* 自動表示タスクに起床要求 */
44
45          switch (flg_ledmode) {      /* LED 表示モードによる場合分け  */
46
47              /*****************************
48                  現在時刻表示モードの時
```

```
49              ****************************/
50              case    CURRENT_TIME:
51
52                  /*****************************
53                       現在時刻が動作中の時
54                         ・調整モードへ移行する
55                   *****************************/
56                  if (flg_time == TIME_TICK) {
57
58                      flg_time = TIME_FIX;            /* 調整モードへ移行     */
59                      wup_tsk (TSK_LCDCHAR);      /* LCD 表示タスク起床 */
60                      flg_interrupt = flg_interrupt | (SW1|SW2|SW3|SW4);
61                                  /* タクトスイッチ割り込み 1,2,3,4 を許可    */
62                      SetTime (STOP);         /* 現在時刻を刻むハンドラを停止   */
63
64                  /*****************************
65                       現在時刻調整モードの時
66                         ・現在時刻刻み再開
67                   *****************************/
68                  } else if (flg_time == TIME_FIX) {
69
70                      flg_time = TIME_START;          /* 調整モード解除       */
71                      wup_tsk (TSK_LCDCHAR);      /* LCD 表示タスク起床 */
72                      flg_interrupt = flg_interrupt & ~(SW1|SW2|SW3|SW4);
73                              /* タクトスイッチ割り込み 1,2,3,4 を禁止    */
74                      SetTime (START);  /* 現在時刻を刻むハンドラを再開   */
75                  }
76                  break;
77
78              /*****************************
79                   タイマー表示モードの時
80               *****************************/
81              case    TIMER:
82
83                  /*******************************
84                       タイマーが動作していない時
85                         ・タイマー動作開始
86                   *******************************/
87                  if (flg_timer == TIMER_STOP) {
88
89                      flg_timer = TIMER_START;        /* タイマー動作開始     */
90                      wup_tsk (TSK_LCDCHAR);      /* LCD 表示タスク起床 */
91                      flg_interrupt = ALLSW & ~(SW1|SW2|SW3|SW4);
92                              /* タクトスイッチ割り込み 1,2,3,4 を禁止    */
93                      wup_tsk (TSK_INTCONTROL);
94                                      /* 割り込み制御タスクに処理移行   */
95
96                  /*************************
97                       タイマーが動作中の時
98                         ・タイマー中断
99                   *************************/
```

```
100                 } else if (flg_timer == TIMER_WORKING) {
101
102
103                     flg_timer = TIMER_INTERRUPT;   /* タイマー中断            */
104                     wup_tsk (TSK_LCDCHAR);         /* LCD 表示タスク起床 */
105                     flg_interrupt = (SW5|SW7|SW8);
106                                     /* タクトスイッチ割り込み 5,7,8 を禁止  */
107                     wup_tsk (TSK_INTCONTROL);
108                                         /* 割り込み制御タスクに処理移行   */
109                 }
110                 break;
111
112             /***********************************
113                   ストップウォッチ表示モードの時
114              ***********************************/
115             case   STOPWATCH:
116
117                 /*****************************************
118                       ストップウォッチが動作していない時
119                  *****************************************/
120                 if (flg_stopwatch == SWATCH_STOP) {
121
122                     /*************************************
123                         モード切り替え
124                           ・1 秒モード ←→ 1/10 秒モード
125                      *************************************/
126                     if (flg_swatchmode == MODE_1) {
127
128                         flg_swatchmode = MODE_10;
129
130                     } else if (flg_swatchmode == MODE_10) {
131
132                         flg_swatchmode = MODE_1;
133                     }
134                     wup_tsk (TSK_LCDCHAR);         /* LCD 表示タスク起床 */
135                 }
136                 break;
137
138             /***********************************
139                   スロットマシン表示モードの時
140              ***********************************/
141             case   SLOT:
142
143                 break;
144         }
145     }
146     return (0xff);
147 }
```

【 inttakt7.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   inttakt7.c
5       Target              :   V853
6       Procedure Defined   :   TaktSwitch7 ()
7   *****************************************************************************/
8   /*** ヘッダファイル ***/
9   #include   "sample.h"
10  #include   "common.h"
11
12
13  /*** 関数宣言 ***/
14  INT    TaktSwitch7 (void);
15
16  /*****************************************************************************
17      種　類　: 間接割り込みハンドラ
18      関数名　: TaktSwitch7 ()
19      機　能　: タクトスイッチ 7 割り込みの処理
20      引き数　: なし
21      返り値　: 0xff (間接割り込みハンドラから抜ける)
22  *****************************************************************************/
23  INT
24  TaktSwitch7 (void)
25  {
26      if ((flg_interrupt & SW7) == SW7) {
27                  /* タクトスイッチ 7 割り込みが許可状態であるかをチェック */
28                  /* 割り込みが受け付けられない時は、何もせずに抜ける       */
29
30          switch (flg_condition) {    /* 気温・湿度表示フラグによる場合分け */
31
32              /*****************************
33                  表示フラグが OFF の時
34              *****************************/
35              case   OFF :
36
37                  flg_condition = ON;             /* 表示フラグを ON にする          */
38                  act_cyc (CYC_MESURE, TCY_INI|TCY_ON);
39                                                  /* 定期測定ハンドラを初期化する   */
40                  wup_tsk (TSK_MESURE);   /* センサー読み取りタスクを起床   */
41
42                  break;
43
44              /*****************************
45                  表示フラグが ON の時
46              *****************************/
47              case   ON :
48
49                  flg_condition = OFF;        /* 表示フラグを OFF にする */
```

```
50                  wup_tsk (TSK_LCDCHAR);      /* LCD 表示タスク起床      */
51                  wup_tsk (TSK_AUTOCHANGE);
52                          /* 自動切替タスクを起床し、時限待ちを開始する  */
53                  break;
54              }
55          }
56          return (0xff);
57      }
```

【 inttakt8.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   inttakt8.c
5       Target              :   V853
6       Procedure Defined   :   TaktSwitch8 ()
7   *****************************************************************************/
8   /*** ヘッダファイル ***/
9   #include   "sample.h"
10  #include   "common.h"
11
12
13  /*** 関数宣言 ***/
14  INT    TaktSwitch8 (void);
15
16  /*****************************************************************************
17      種　類　: 間接割り込みハンドラ
18      関数名　: TaktSwitch8 ()
19      機　能　: タクトスイッチ 8 割り込みの処理
20      引き数　: なし
21      返り値　: 0xff (間接割り込みハンドラから抜ける)
22  *****************************************************************************/
23  INT
24  TaktSwitch8 (void)
25  {
26
27      static  int     flg_tmp;  /* 割り込み制御フラグ保存変数 */
28
29
30      if ((flg_interrupt & SW8) == SW8) {
31              /* タクトスイッチ 8 割り込みが許可状態であるかをチェック */
32              /* 割り込みが受け付けられない時は、何もせずに抜ける       */
33
34          switch (flg_memo) {       /* メモ表示フラグによる場合分け  */
35
36              /*****************************
37                  表示フラグが OFF の時
38              *****************************/
39              case   OFF :
40
41                  if (flg_condition == ON)   flg_condition = OFF;
42                              /* 気温・湿度表示が ON の時は OFF にする  */
43
44                  flg_tmp = flg_interrupt;    /* 現在の割り込み制御フラグ保存      */
45                  flg_interrupt = SW8;        /* タクトスイッチ割り込み 8 を許可  */
46                  flg_memo = ON;              /* メモ表示フラグを ON にする        */
47                  flg_memodisp = MEMO_ALLDISP;
48                      /* 記憶されているメモを LCD に表示するフラグを立てる */
```

```
49                  sus_tsk (TSK_AUTOCHANGE);
50                                      /* 温度・湿度自動表示タスク サスペンド */
51                  wup_tsk (TSK_MEMOCONTROL);     /* メモ管理タスク起床 */
52
53                  outpb (ASIM00, (inpb (ASIM00) | RCV_ENABLE));
54                                              /* シリアル受信許可     */
55
56                  break;
57
58              /****************************
59                   表示フラグが ON の時
60              ****************************/
61              case    ON :
62
63                  flg_interrupt = flg_tmp;            /* 割り込み制御フラグを戻す          */
64                  flg_memo = OFF;                     /* メモ表示フラグを OFF にする       */
65                  wup_tsk (TSK_LCDCHAR);      /* LCD 表示タスク起床                        */
66                  rsm_tsk (TSK_AUTOCHANGE);
67                                      /* 温度・湿度自動表示タスク サスペンド解除   */
68                  outpb (ASIM00, (inpb (ASIM00) & ~RCV_ENABLE));
69                                              /* シリアル受信禁止     */
70                  break;
71          }
72      }
73      return (0xff);
74  }
```

【 inttouch.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :   inttouch.c
   Target              :   V853
   Procedure Defined   :   TouchSenser ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"


/***  関数宣言  ***/
INT    TouchSenser (void);

/*****************************************************************************
   種　類　: 間接割り込みハンドラ
   関数名　: TouchSenser ()
   機　能　: タッチセンサー割り込み処理
               ・メモ表示の時                  → メモ全クリア
               ・メモ非表示の時
                   ・現在時刻表示モード        → 未使用
                   ・タイマー表示モード        → タイマークリア
                   ・ストップウォッチモード    → 未使用
                   ・スロットマシンモード      → リセット
   引き数　: なし
   返り値　: 0xff (  間接割り込みハンドラから抜ける  )
*****************************************************************************/
INT
TouchSenser (void)
{

    /*************************************
        メモ表示モードの時
          ・表示されているメモを消去する
    *************************************/
    if (flg_memo == ON) {

        flg_memodisp = MEMO_ALLCLEAR;
                        /*  記憶されているメモをクリアするフラグを立てる    */
        wup_tsk (TSK_MEMOCONTROL);     /*  メモ管理タスク起床  */

    /*******************************
        メモ表示モードではない時
    *******************************/
    } else if (flg_memo == OFF) {

        switch (flg_ledmode) {      /* LED  表示モードによる場合分け   */
```

```
49
50              /*****************************
51                   現在時刻表示モードの時
52               *****************************/
53              case    CURRENT_TIME:
54
55                  break;
56
57              /*****************************
58                   タイマー表示モードの時
59               *****************************/
60              case    TIMER:
61
62                  /*******************************
63                       タイマーが終了している時
64                   *******************************/
65                  if (flg_timer == TIMER_FINISH) {
66
67                      flg_timer = TIMER_STOP;         /* タイマー動作状態変更       */
68                      TimerCountDown (INIT, OFF);     /* 初期値表示                 */
69                      SetTimerCount (CLEAR);          /* タイマーカウントクリア     */
70                      flg_interrupt = ALLSW & ~(SW6);
71                                              /* タクトスイッチ割り込み 6 を禁止  */
72                      wup_tsk (TSK_LCDCHAR);      /* LCD 表示タスク起床         */
73
74                  /*******************************
75                   *   タイマーが中断している時   *
76                   *******************************/
77                  } else if (flg_timer == TIMER_INTERRUPT) {
78
79                      flg_timer = TIMER_STOP;         /* タイマー動作状態変更       */
80                      TimerCountDown (INIT, OFF);     /* 初期値表示                 */
81                      flg_interrupt = ALLSW & ~(SW6);
82                                              /* タクトスイッチ割り込み 6 を禁止  */
83                  }
84                  break;
85
86              /*********************************
87                   ストップウォッチ表示モードの時
88               *********************************/
89              case    STOPWATCH:
90
91                  break;
92
93              /*********************************
94                   スロットマシン表示モードの時
95               *********************************/
96              case    SLOT:
97
98                  if (flg_slot == SLOT_STOP) {
99                                  /* スロットマシンがストップしている時  */
100                     SlotRotation (INIT, OFF);
```

```
101                                 /* スロットの値を [777]にリセットする  */
102                     }
103                     break;
104                 }
105             }
106             return (0xff);
107     }
```

【 intad.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   intad.c
5       Target              :   V853
6       Procedure Defined   :   IntADfinish ()
7   *****************************************************************************/
8   /***  ヘッダファイル  ***/
9   #include    "sample.h"
10  #include    "common.h"
11
12  /***  関数宣言  ***/
13  INT    IntADfinish (void);
14
15  /*****************************************************************************
16      種　類　: 間接割り込みハンドラ
17      関数名　: IntADfinish ()
18      機　能　: A/D コンバート終了割り込みの処理
19      引き数　: なし
20      返り値　: TSK_MESURE (センサ読み取りタスクを起床して抜ける)
21  *****************************************************************************/
22  INT
23  IntADfinish (void)
24  {
25      return (TSK_MESURE);   /*  センサー読み取りタスクを起床    */
26  }
```

【 intctrl.c 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :   intctrl.c
   Target              :   V853
   Procedure Defined   :   IntControl()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include   "sample.h"
#include   "common.h"


/***  関数宣言  ***/
void   IntControl (INT);

/*****************************************************************************
   種　類　: タスク
   関数名　: IntControl ()
   機　能　: タクトスイッチ割り込み制御
               - 割り込み処理が緩やかでよい場合の処理を記述。
                 急を要するものは、それぞれのタクトスイッチ処理中に記述。
   引き数　: 初期情報 (起動時のタスク引き渡し情報)
   返り値　: なし
*****************************************************************************/
void
IntControl (INT stacd)
{

    for ( ; ; ) {

        slp_tsk ();     /*  自タスク起床待ち */

        switch (flg_ledmode) {      /* LED  表示モードによって場合わけ    */

            /*****************************
                現在時刻表示モードの時
             *****************************/
            case    CURRENT_TIME:

                if (flg_time == TIME_FIX) {       /*  現在時刻調整モードの時  */

                    SetTime (FIX);        /*  現在時刻調整    */
                }
                break;

            /*************************
                タイマーモードの時
             *************************/
            case    TIMER:
```

```
50
51                  if (flg_timer == TIMER_STOP) {
52                                          /* タイマー動作していないとき */
53
54                      SetTimerCount (SETTING);          /* タイマー値設定    */
55                      flg_interrupt = flg_interrupt | SW6;  /* 割り込み制御      */
56
57                  } else if (flg_timer == TIMER_START) {
58                                          /* タイマー動作が開始された時 */
59                      SetTimerCount (START);      /* タイマー動作開始     */
60
61                  } else if (flg_timer == TIMER_INTERRUPT) {
62                                                  /* タイマー動作中の時 */
63                      SetTimerCount (STOP);           /* タイマー動作停止     */
64                      flg_interrupt = (SW5|SW7|SW8);/* 割り込み制御          */
65                  }
66                  break;
67
68              /*********************************
69                   ストップウォッチモードの時
70               *********************************/
71              case   STOPWATCH:
72
73                  break;
74
75              /*********************************
76                   スロットマシンモードの時
77               *********************************/
78              case   SLOT:
79
80                  flg_slotfigure = SLOT_ALLCLEAR;
81                                  /* スロットマシン動作フラグクリア    */
82                  act_cyc (CYC_SLOT, TCY_ON);
83                                  /* スロットマシンドライバ 起動    */
84                  break;
85          }
86      }
87
88  }
```

【 start.850 】(GHS 版)

```
1   /************************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]    アプリケーション・ノート
3   ************************************************************************************
4       Module          :   start.850
5       Target          :   V853
6   ************************************************************************************/
7
8   /************************************************************************************
9       種　類　: 関数
10      関数名　: __start
11      機　能　: スタートアップ・ルーチン
12  ************************************************************************************/
13      .text
14
15      .globl  __start
16      .align  4
17  __start:
18
19      /******************
20              TP セット
21      ******************/
22      mov    .dmy_lbl, r11
23      jarl.dmy_lbl, tp
24  .dmy_lbl:
25      sub     r11, tp
26
27      /***************************************
28              Multi 用 Frame Pointer クリア
29      ***************************************/
30      mov    0, r28
31
32      /******************
33              GP セット
34      ******************/
35      .extern     ___ghsbegin_sdabase   /* top address of SDA area */
36      movhi hi(___ghsbegin_sdabase), r0, gp
37      movealo(___ghsbegin_sdabase), gp, gp
38      addi    0x4000, gp, gp
39      addi    0x4000, gp, gp
40
41      /*************************************************
42              SP セット (スタート・アップ・ルーチン用)
43      *************************************************/
44      mov    0x810000, sp
45      mov    -4, r1
46      and     r1, sp
47
48      /******************
49              EP セット
50      ******************/
```

```
51          movhi hi(__ep), r0, r1
52          addi    lo(__ep), r1, r1
53          be  .no_ep
54          mov    r1, ep
55
56      .no_ep:
57
58          /*****************************
59                  MM レジスタ・セット
60          *****************************/
61          movealo(0xf04c), r0, r1
62          movea0x0f, r0, r11
63          st.b     r11, 0[r1]
64
65          /*****************************
66                  meminit() 呼び出し
67          *****************************/
68          .extern     _meminit
69          jarl_meminit, lp
70
71          /*************************
72                  RX850 スタート
73          *************************/
74          .extern     __urx_start
75          jr   __urx_start
76
77          .section ".secinfo", "a"
```

【 start.s 】(NEC 版)

```
1   /***********************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]    アプリケーション・ノート
3   ***********************************************************************************
4       Module              :   start.s
5       Target              :   V853
6   ***********************************************************************************/
7
8   -- ============================================
9   --              Dummy datas
10  -- ============================================
11
12      .sbss
13      .lcomm    __sbss_dummy, 0, 0
14
15      .bss
16      .lcomm    __bss_dummy, 0, 0
17
18      .sebss
19      .lcomm    __sebss_dummy, 0, 0
20
21      .tidata
22  __tidata_dummy:
23      .space 0
24
25
26  -- ============================================
27  --              RESET table
28  -- ============================================
29
30      .section            "RESET"
31      jr          __start
32
33
34  -- ============================================
35  --              Start up routine
36  -- ============================================
37
38      .text
39      .align  4
40
41      .globl  __start
42  __start:
43  ------ Set TP ------------------------------
44      .extern __tp_TEXT, 4
45      mov   #__tp_TEXT, r11
46      mov   r11, tp
47
48  ------ Set GP ------------------------------
49      .extern __gp_DATA, 4
50      mov   #__gp_DATA, r11
51      mov   r11, gp
```

```
52
53      ------ Set EP ------------------------------
54          .extern __ep_DATA, 4
55          mov   #__ep_DATA, r11
56          mov   r11, ep
57
58      ------ clear sbss section -----------------
59          .extern __ssbss, 4
60          .extern __esbss, 4
61          mov   #__ssbss, r13
62          mov   #__esbss, r12
63          cmp   r12, r13
64          bnl sbss_init_end
65
66      sbss_init_loop:
67          st.w   r0, [r13]
68          add    4, r13
69          cmp   r12, r13
70          bl  sbss_init_loop
71
72      sbss_init_end:
73
74      ------ clear bss section -----------------
75          .extern __sbss, 4
76          .extern __ebss, 4
77          mov   #__sbss, r13                  -- clear bss section
78          mov   #__ebss, r12
79          cmp   r12, r13
80          bnl bss_init_end
81
82      bss_init_loop:
83          st.w   r0, [r13]
84          add    4, r13
85          cmp   r12, r13
86          bl  bss_init_loop
87
88      bss_init_end:
89
90      ------ clear sbss section -----------------
91          .extern __ssebss, 4
92          .extern __esebss, 4
93          mov   #__ssebss, r13
94          mov   #__esebss, r12
95          cmp   r12, r13
96          bnl sebss_init_end
97
98      sebss_init_loop:
99          st.w   r0, [r13]
100         add    4, r13
101         cmp   r12, r13
102         bl  sebss_init_loop
103
104     sebss_init_end:
105
```

```
106     ------ Start RX850 -------------------------------
107
108         .extern     __urx_start
109         jr   __urx_start
```

【 meminit.c 】(GHS 用)

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :  meminit.c
   Target              :  V853
   Procedure Defined   :  meminit ()
*****************************************************************************/
/***  ヘッダファイル  ***/
#include <stdlib.h>
#include <string.h>

/***  関数宣言  ***/
void meminit(void);

/*****************************************************************************
   種　類　: 関数
   関数名　: meminit ()
   機　能　: 初期値なしデータ領域クリア
             : 初期値ありデータコピー
   引き数　: なし
   返り値　: なし
   備　考　: GHS 推奨の方法を用いる
*****************************************************************************/
void
meminit(void)
{
    {
        extern void __ghsbinfo_clear(void);
        extern void __ghseinfo_clear(void);
        void **b = (void **)__ghsbinfo_clear;
        void **e = (void **)__ghseinfo_clear;

        while (b < e){
            void *s, *n, *v;

            s = (char *)(*b++);
            v = *b++;
            n = *b++;
            memset(s, (int)v, (size_t)n);
        }
    }

    {
        extern void __ghsbinfo_copy(void);
        extern void __ghseinfo_copy(void);
        void **b = (void **)__ghsbinfo_copy;
        void **e = (void **)__ghseinfo_copy;

        while (b < e){
```

```
50                  void *s, *ct, *n;
51
52                  s = (char *)(*b++);
53                  ct = (char *)(*b++);
54                  n = *b++;
55                  memcpy(s, ct, (size_t)n);
56              }
57          }
58      }
```

【 port.850 】(GHS 版)

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module              :   port.850
   Target              :   V853
*****************************************************************************/

   .text
/*****************************************************************************
   種　類　: 関数
   関数名　: inpb(), inph(), inpw()
   機　能　: ポート操作 (入力)
   引　数　: 1 - ポートアドレス
   返り値　: 入力値
*****************************************************************************/
/*****************************************************************************
   種　類　: 関数
   関数名　: outpb(), outph(), outpw()
   機　能　: ポート操作 (出力)
   引　数　:   1 - ポートアドレス
               2 - 出力値
               現在の LCD カーソルポジション
   返り値　: なし
*****************************************************************************/

/*******************************
      input from port (byte)
 *******************************/
   .align  4
   .globl  _inpb
   ..lxtdaG._inpb=:0

_inpb:
   ld.b    0[r6], r10
   jmp     [lp]

/*******************************
      output to port (byte)
 *******************************/
   .align  4
   .globl  _outpb
   ..lxtdaG._outpb=:0

_outpb:
   st.b    r7, 0[r6]
   jmp     [lp]
```

```
107  /***********************************
108          input from port (half word)
109    ***********************************/
110       .align  4
111       .globl  _inph
112       ..lxtdaG._inph=:0
113
114  _inph:
115       ld.h    0[r6], r10
116       jmp     [lp]
117
118  /***********************************
119          output to port (half word)
120    ***********************************/
121       .align  4
122       .globl  _outph
123       ..lxtdaG._outph=:0
124
125  _outph:
126       st.h    r7, 0[r6]
127       jmp     [lp]
128
129
130  /*******************************
131          input from port (word)
132    *******************************/
133       .align  4
134       .globl  _inpw
135       ..lxtdaG._inpw=:0
136
137  _inpw:
138       ld.w    0[r6], r10
139       jmp     [lp]
140
141  /*******************************
142          output to port (word)
143    *******************************/
144       .align  4
145       .globl  _outpw
146       ..lxtdaG._outpw=:0
147
148  _outpw:
149       st.w    r7, 0[r6]
150       jmp     [lp]
```

【 port.s 】(NEC 版)

```
1   -- /*************************************************************************
2   --      Realtime OS nucleus [RX850]   アプリケーション・ノート
3   -- *************************************************************************
4   --      Module              :   port.s
5   --      Target              :   V853
6   -- *************************************************************************/
7
8       .text
9   -- /*************************************************************************
10  --      種　類　: 関数
11  --      関数名　: inpb(), inph(), inpw()
12  --      機　能　: ポート操作 (入力)
13  --      引　数　: 1 - ポートアドレス
14  --      返り値　: 入力値
15  -- *************************************************************************/
16  -- /*************************************************************************
17  --      種　類　: 関数
18  --      関数名　: outpb(), outph(), outpw()
19  --      機　能　: ポート操作 (出力)
20  --      引　数　: 1 - ポートアドレス
21  --                2 - 出力値
22  --                現在の LCD カーソルポジション
23  --      返り値　: なし
24  -- *************************************************************************/
25
26  -- /*****************************
27  --          input from port (byte)
28  --    *****************************/
29      .align  4
30      .globl  _inpb
31
32  _inpb:
33      ld.b    0[r6], r10
34      jmp     [lp]
35
36  -- /*****************************
37  --          output to port (byte)
38  --    *****************************/
39      .align  4
40      .globl  _outpb
41
42  _outpb:
43      st.b    r7, 0[r6]
44      jmp     [lp]
45
46
47  -- /*********************************
48  --          input from port (half word)
49  --    *********************************/
```

```
50          .align  4
51          .globl  _inph
52
53      _inph:
54          ld.h    0[r6], r10
55          jmp     [lp]
56
57      -- /**********************************
58      --              output to port (half word)
59      --    **********************************/
60          .align  4
61          .globl  _outph
62
63      _outph:
64          st.h    r7, 0[r6]
65          jmp     [lp]
66
67
68      -- /*******************************
69      --              input from port (word)
70      --    *******************************/
71          .align  4
72          .globl  _inpw
73
74      _inpw:
75          ld.w    0[r6], r10
76          jmp     [lp]
77
78      -- /*******************************
79      --              output to port (word)
80      --    *******************************/
81          .align  4
82          .globl  _outpw
83
84      _outpw:
85          st.w    r7, 0[r6]
86          jmp     [lp]
```

【 idlhdr.c 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   idlhdr.c
5       Target              :   V853
6       Procedure Defined   :   idle_handler ()
7   *****************************************************************************/
8   #include   "sample.h"
9   #include   "common.h"
10
11  /*** 関数宣言 ***/
12  void   __halt(void);
13  void   __stop(void);
14  void   __idle(void);
15  void   idle_handler(void);
16
17  /*** 変数宣言 ***/
18  int     idlemode;    /* アイドル・モード設定変数 */
19
20  /*****************************************************************************
21      種　類　: 関数
22      関数名　: idle_handler ()
23      機　能　: アイドル処理ルーチン
24      引き数　: なし
25      返り値　: なし
26  *****************************************************************************/
27  void
28  idle_handler(void)
29  {
30      switch(idlemode){
31
32      case HALT_MODE :    /* HALT モードに設定しているとき */
33          ena_int();
34          __halt();
35          break;
36
37      case IDLE_MODE :    /* IDLE モードに設定しているとき */
38          __idle();
39          break;
40
41      case STOP_MODE :    /* STOP モードに設定しているとき */
42          __stop();
43          break;
44
45      }
46      return;
47  }
```

【 halt.850 】(GHS 版)

```
1   /***********************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]    アプリケーション・ノート
3   ***********************************************************************************
4       Module              :   halt.850
5       Target              :   V853
6   ***********************************************************************************/
7
8   /***********************************************************************************
9       種　類　: 関数
10      関数名　: ___halt
11      機　能　: CPU halt 処理ルーチン
12  ***********************************************************************************/
13
14      .text
15      .align  4
16
17      ..lxtdaG.___halt=:0
18      .globl  ___halt
19  ___halt:
20      halt                    /* HALT モードに移行  */
21      jmp     [lp]
```

【 halt.s 】(NEC 版)

```
1   /************************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   ************************************************************************************
4       Module              :   halt.s
5       Target              :   V853
6   ************************************************************************************/
7
8   /************************************************************************************
9       種　類　: 関数
10      関数名　: ___halt
11      機　能　: CPU halt 処理ルーチン
12  ************************************************************************************/
13
14      .text
15      .align  4
16
17      .globl  ___halt
18  ___halt:
19      halt                    /* HALT モードに移行  */
20      jmp     [lp]
```

【 idle.850 】(GHS 版)

```
1   /************************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   ************************************************************************************
4       Module              :   idle.850
5       Target              :   V853
6   ************************************************************************************/
7
8   /************************************************************************************
9       種　類　: 関数
10      関数名　: ___idle
11      機　能　: CPU idle 処理ルーチン
12  ************************************************************************************/
13
14  #define   REG_PSW       5               /* システム・レジスタ番号 5 = PSW        */
15  #define   REG_PSC       0xfffff070      /* パワーセーブ・コントロール・レジスタ   */
16  #define   REG_PRCMD     0xfffff170      /* コマンド・レジスタ                     */
17
18  #define   PSW_NP    0x00000080      /* PSW : NMI Pending */
19
20  #define   PSC_IDLE  0x04          /* PSC : IDLE モード */
21
22      .text
23      .align  4
24
25      ..lxtdaG.___idle=:0
26      .globl  ___idle
27  ___idle:
28      stsr    REG_PSW, r6             /* PSW の値を取得                  */
29      ori     PSW_NP, r6, r7          /* NP ビットを立てる(割り込み禁止)  */
30      ldsr    r7, REG_PSW
31
32      st.b    r0, REG_PRCMD[r0]       /* コマンド・レジスタへ書き込み     */
33
34      movea   PSC_IDLE, r0, r8        /* r8 : IDLE モード番号             */
35      st.b    r8, REG_PSC[r0]         /* PSC を IDLE モードに設定         */
36
37      ldsr    r6, REG_PSW             /* NP ビットを落とす(割り込み許可)  */
38      nop
39      nop
40      nop
41      nop
42      nop
43
44      jmp     [lp]
```

【 Idle.s 】(NEC 版)

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]  アプリケーション・ノート
3   *****************************************************************************
4       Module              :  idle.s
5       Target              :  V853
6   *****************************************************************************/
7
8   /*****************************************************************************
9       種　類  : 関数
10      関数名  : ___idle
11      機　能  : CPU idle 処理ルーチン
12  *****************************************************************************/
13
14  #define   REG_PSW     5             /* システム・レジスタ番号 5 = PSW      */
15  #define   REG_PSC     0xfffff070      /* パワーセーブ・コントロール・レジスタ  */
16  #define   REG_PRCMD   0xfffff170      /* コマンド・レジスタ                  */
17
18  #define   PSW_NP    0x00000080    /* PSW : NMI Pending */
19
20  #define   PSC_IDLE  0x04          /* PSC : IDLE モード */
21
22      .text
23      .align  4
24
25      .globl  ___idle
26  ___idle:
27      stsr    REG_PSW, r6             /* PSW の値を取得                  */
28      ori     PSW_NP, r6, r7        /* NP ビットを立てる(割り込み禁止)   */
29      ldsr    r7, REG_PSW
30
31      st.b    r0, REG_PRCMD[r0]     /* コマンド・レジスタへ書き込み      */
32
33      movea   PSC_IDLE, r0, r8      /* r8 : IDLE モード番号              */
34      st.b    r8, REG_PSC[r0]         /* PSC を IDLE モードに設定          */
35
36      ldsr    r6, REG_PSW             /* NP ビットを落とす(割り込み許可)   */
37      nop
38      nop
39      nop
40      nop
41      nop
42
43      jmp     [lp]
```

【 stop.850 】(GHS 版)

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.10]   アプリケーション・ノート
3   *****************************************************************************
4       Module              :   stop.850
5       Target              :   V853
6   *****************************************************************************/
7
8   /*****************************************************************************
9       種　類  : 関数
10      関数名  : ___stop
11      機　能  : CPU stop 処理ルーチン
12  *****************************************************************************/
13
14  #define   REG_PSW       5               /* システム・レジスタ番号 5 = PSW       */
15  #define   REG_PSC       0xfffff070      /* パワーセーブ・コントロール・レジスタ  */
16  #define   REG_PRCMD     0xfffff170      /* コマンド・レジスタ                    */
17
18  #define   PSW_NP    0x00000080      /* PSW : NMI Pending */
19
20  #define   PSC_STP   0x02            /* PSC : STOP モード    */
21  #define   PSC_TBCS  0x00            /* Time Base Count Select   : fxx/2^8   */
22  #define   PSC_CESEL 0x00            /* Crystal/External Select : 0          */
23
24  /* #definePSC_TBCS  0x20 */       /* Time Base Count Select   : fxx/2^9   */
25  /* #definePSC_CESEL 0x10 */       /* Crystal/External Select : 1          */
26
27      .text
28      .align  4
29
30      ..lxtdaG.___stop=:0
31      .globl  ___stop
32  ___stop:
33      stsr    REG_PSW, r6             /* PSW の値を取得                       */
34      ori     PSW_NP, r6, r7          /* NP ビットを立てる(割り込み禁止)   */
35      ldsr    r7, REG_PSW
36
37      st.b    r0, REG_PRCMD[r0]       /* コマンド・レジスタへ書き込み         */
38
39      movea(PSC_TBCS|PSC_CESEL|PSC_STP), r0, r8
40                                      /* r8 : STOP モード番号                 */
41      st.b    r8, REG_PSC[r0]         /* PSC を STOP モードに設定             */
42
43      ldsr    r6, REG_PSW             /* NP ビットを落とす(割り込み許可)   */
44      nop
45      nop
46      nop
47      nop
48      nop
49
```

```
50          jmp     [lp]
```

【 stop.s 】(NEC 版)

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.10]  アプリケーション・ノート
3   *****************************************************************************
4       Module              :  stop.s
5       Target              :  V853
6   *****************************************************************************/
7
8   /*****************************************************************************
9       種　類  : 関数
10      関数名  : ___stop
11      機　能  : CPU stop 処理ルーチン
12  *****************************************************************************/
13
14  #define   REG_PSW     5             /* システム・レジスタ番号 5 = PSW       */
15  #define   REG_PSC     0xfffff070        /* パワーセーブ・コントロール・レジスタ   */
16  #define   REG_PRCMD   0xfffff170        /* コマンド・レジスタ                   */
17
18  #define   PSW_NP      0x00000080      /* PSW : NMI Pending */
19
20  #define   PSC_STP     0x02          /* PSC : STOP モード    */
21  #define   PSC_TBCS    0x00          /* Time Base Count Select   : fxx/2^8   */
22  #define   PSC_CESEL 0x00            /* Crystal/External Select : 0          */
23
24  /* #definePSC_TBCS   0x20 */        /* Time Base Count Select   : fxx/2^9   */
25  /* #definePSC_CESEL 0x10 */         /* Crystal/External Select : 1          */
26
27      .text
28      .align  4
29
30      .globl  ___stop
31  ___stop:
32      stsr    REG_PSW, r6             /* PSW の値を取得                       */
33      ori     PSW_NP, r6, r7          /* NP ビットを立てる(割り込み禁止)   */
34      ldsr    r7, REG_PSW
35
36      st.b    r0, REG_PRCMD[r0]       /* コマンド・レジスタへ書き込み         */
37
38      movea(PSC_TBCS|PSC_CESEL|PSC_STP), r0, r8
39                                      /* r8 : STOP モード番号                 */
40      st.b    r8, REG_PSC[r0]         /* PSC を STOP モードに設定             */
41
42      ldsr    r6, REG_PSW             /* NP ビットを落とす(割り込み許可)   */
43      nop
44      nop
45      nop
46      nop
47      nop
48
49      jmp     [lp]
```

【 vector.850 (GHS 版) / vector.s (NEC 版)】

```
1   /************************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]    アプリケーション・ノート
3   ************************************************************************************
4       Module              :   vector.850
5       Target              :   V853
6   ************************************************************************************/
7
8   /************************************************************************************
9       種　類　: ベクターテーブル
10      関数名　: __reset
11      機　能　: リセット後の定義
12  ************************************************************************************/
13      .org    0x00000000
14      .globl ___reset
15
16  /***************************************
17        リセット後、___start へジャンプ
18  ***************************************/
19  ___reset:
20      .extern         __start
21      jr      __start
22
23
24      /*** Dummy ***/
25      .section    ".dmmy",.text
```

【 sit.cf 】(GHS 版)

```
1   #/*****************************************************************************
2   #   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   #*****************************************************************************
4   #   Module      :   sit.cf
5   #   Target      :   V853
6   #               :   コンフィギュレーション・ファイル
7   #*****************************************************************************/
8
9   ser_def
10  rxsers RX850 V310
11
12  sit_def
13
14  -- /***************************
15  --          タスクの優先度範囲
16  -- ***************************/
17  --      priority_num
18  maxpri    13
19
20  -- /*********************************************
21  --          システムクロックのクロック割り込み要因
22  -- *********************************************/
23  --  For CC850 (INTCM4 = 001C0H)
24  --  clkhdr_name
25  clkhdr 20
26
27  -- /******************
28  --          タスク情報
29  -- ******************/
30  --  << LED ダイナミックドライブタスク >>
31  --      task_name           task_address     stack_size    pri status        stacode   ei/di
32  tsk     TSK_LEDDRIVE  _LedDriveTask  180:pool1    3   TTS_DMT   0x0       ei
33
34  --  << 秒刻み LED 点灯・消灯タスク >>
35  --      task_name           task_address     stack_size    pri status        stacode   ei/di
36  tsk     TSK_LEDFLASH  _LedFlashTask  160:pool1    9   TTS_DMT   0x0       ei
37
38  --  << 現在時刻管理タスク >>
39  --      task_name           task_address     stack_size    pri status        stacode   ei/di
40  tsk     TSK_TIMECTRL  _TimeControl   180:pool1    7   TTS_DMT   0x0       ei
41
42  --  << タイマーカウントダウンタスク >>
43  --      task_name                task_address     stack_size    pri status        stacode   ei/di
44  tsk     TSK_TMRCONTROL     _TimerControl  180:pool1    10 TTS_DMT   0x0       ei
45
46  --  << ストップウォッチ管理タスク >>
47  --      task_name             task_address          stack_size    pri status        stacode   ei/di
48  tsk     TSK_SWATCHCTRL  _StopWatchControl 180:pool0    11 TTS_DMT   0x0       ei
49
```

```
50  -- << スロットマシン管理タスク >>
51  --      task_name          task_address     stack_size    pri status      stacode   ei/di
52  tsk     TSK_SLOTCTRL       _SlotControl     140:pool0     12 TTS_DMT      0x0       ei
53
54  -- << LCD 初期化タスク >>
55  --      task_name       task_address     stack_size    pri status      stacode   ei/di
56  tsk     TSK_INITLCD     _InitLCD         200:pool1     4  TTS_DMT      0x0       ei
57
58  -- << LCD 表示文字管理タスク(固定メッセージ) >>
59  --      task_name          task_address     stack_size    pri status      stacode   ei/di
60  tsk     TSK_LCDCHAR        _LCDChar         200:pool0     8  TTS_DMT      0x0       ei
61
62  -- << LCD 表示文字管理タスク(温度・湿度) >>
63  --      task_name          task_address     stack_size    pri status      stacode   ei/di
64  tsk     TSK_LCDCOND        _LCDCondition    200:pool0     9  TTS_DMT      0x0       ei
65
66  -- << 表示メッセージを LCD 表示タスクへ送信するタスク >>
67  --      task_name          task_address     stack_size    pri status      stacode   ei/di
68  tsk     TSK_LCDSEND        _LCDSendChar     200:pool1     6  TTS_DMT      0x0       ei
69
70  -- << LCD 表示タスク >>
71  --      task_name          task_address     stack_size    pri status      stacode   ei/di
72  tsk     TSK_LCDDISP        _LCDDispChar     200:pool1     5  TTS_DMT      0x0       ei
73
74  -- << 自動的に LCD 表示を温度・湿度表示にするタスク >>
75  --      task_name          task_address          stack_size    pri status      stacode   ei/di
76  tsk     TSK_AUTOCHANGE     _AutoDispChange       200:pool1     8  TTS_DMT      0x0       ei
77
78  -- << メモ管理タスク >>
79  --      task_name          task_address     stack_size    pri status      stacode   ei/di
80  tsk     TSK_MEMOCONTROL    _MemoControl     200:pool0     7  TTS_DMT      0x0       ei
81
82  -- << A/D コンバートタスク >>
83  --      task_name          task_address          stack_size    pri status      stacode   ei/di
84  tsk     TSK_MESURE         _MesureCondition      140:pool0     5  TTS_DMT      0x0       ei
85
86  -- << ステッピングモーター回転タスク >>
87  --      task_name          task_address stack_size    pri status      stacode   ei/di
88  tsk     TSK_STEPPER        _Stepper     140:pool0     13 TTS_DMT      0x0       ei
89
90  -- << シリアル受信結果送信タスク >>
91  --      task_name          task_address          stack_size    pri status      stacode   ei/di
92  tsk     TSK_UARTSEND       _UARTCharSend         140:pool1     6  TTS_DMT      0x0       ei
93
94  -- << ブザータスク >>
95  --      task_name          task_address stack_size    pri status      stacode   ei/di
96  tsk     TSK_BUZZER         _Buzzer      140:pool0     13 TTS_DMT      0x0       ei
97
98  -- << 割り込み制御タスク >>
99  --      task_name             task_address stack_size    pri status      stacode   ei/di
100 tsk     TSK_INTCONTROL     _IntControl  160:pool0     4  TTS_DMT      0x0       ei
101
```

```
102  -- /*************************
103  --       イベントフラグ情報
104  -- *************************/
105  --  << LCD 表示文字用イベントフラグ >>
106  --      evf_name
107  flg     FLG_LCDCHAR
108
109  --  << 現在温度通知イベントフラグ >>
110  --      evf_name
111  flg     FLG_TEMPERATURE
112
113  --  << シリアル受信結果送信用イベントフラグ >>
114  --      evf_name
115  flg     FLG_MEMOCHECK
116
117  -- /*******************
118  --       セマフォ情報
119  -- *******************/
120  --  << LCD 表示管理 >>
121  --          semaph_name          init_count
122  sem         SEM_LCDSEND          1
123
124  --  << LCD 表示管理 >>
125  --          semaph_name          init_count
126  sem         SEM_LCDDISP          0
127
128  --  << ストップウォッチ時の LED 点灯タイミング変数初期化処理用 >>
129  --          semaph_name          init_count
130  sem         SEM_LEDTIMING        0
131
132  --
133  -- /*************************
134  --       メールボックス情報
135  -- *************************/
136  --  << 7SegLED 表示用メールボックス(下一桁) >>
137  --          mbx_name             mwai_opt
138  mbx         MBX_FIGURE1          TA_MFIFO
139
140  --  << 7SegLED 表示用メールボックス(下二桁) >>
141  --          mbx_name             mwai_opt
142  mbx         MBX_FIGURE2          TA_MFIFO
143
144  --  << 7SegLED 表示用メールボックス(下三桁) >>
145  --          mbx_name             mwai_opt
146  mbx         MBX_FIGURE3          TA_MFIFO
147
148  --  << 7SegLED 表示用メールボックス(下四桁) >>
149  --          mbx_name             mwai_opt
150  mbx         MBX_FIGURE4          TA_MFIFO
151
152  --  << LED 点灯用メールボックス >>
153  --          mbx_name         mwai_opt
```

```
154  mbx       MBX_LED       TA_MFIFO
155
156  --  << セッティング用メールボックス(下一桁) >>
157  --          mbx_name       mwai_opt
158  mbx       MBX_SET1      TA_MFIFO
159
160  --  << セッティング用メールボックス(下二桁) >>
161  --          mbx_name       mwai_opt
162  mbx       MBX_SET2      TA_MFIFO
163
164  --  << セッティング用メールボックス(下三桁) >>
165  --          mbx_name       mwai_opt
166  mbx       MBX_SET3      TA_MFIFO
167
168  --  << セッティング用メールボックス(下四桁) >>
169  --          mbx_name       mwai_opt
170  mbx       MBX_SET4      TA_MFIFO
171
172  --  << LCD 表示用メモリ領域 送受信メールボックス >>
173  --          mbx_name           mwai_opt
174  mbx       MBX_TO_LCD     TA_MFIFO
175
176  --  << LCD 表示用メモリ領域 送信用メールボックス >>
177  --          mbx_name             mwai_opt
178  mbx       MBX_FROM_LCD    TA_MFIFO
179
180  --  << 温度・湿度データ送信メールボックス >>
181  --          mbx_name           mwai_opt
182  mbx       MBX_ADDATA     TA_MFIFO
183
184  -- /*************************
185  --        メモリプール情報
186  -- *************************/
187  --  << 7SegLED 表示メールボックス用 >>
188  --          mpl_name       block_size    block_num
189  mpf       MPL_LED        0xc:pool0     10
190
191  --  << 7SegLED 表示更新メールボックス用 >>
192  --          mpl_name       block_size    block_num
193  mpf       MPL_SET        0xc:pool0     5
194
195  --  << LCD 表示文字列 送信メールボックス用 >>
196  --          mpl_name       block_size        block_num
197  mpf       MPL_LCD        0x28:pool0        1
198
199  --  << LCD 表示文字列 送信メールボックス用 >>
200  --          mpl_name           block_size        block_num
201  mpf       MPL_ADCONV     0xc:pool0         1
202
203  -- /************************
204  --        周期ハンドラ情報
205  -- ************************/
```

```
206  -- << 現在時刻保持ハンドラ >>
207  --      cyclic_name         cychdr_addr      activation     interval
208  cyc     CYC_TICKTIME   _TickTime        TCY_ON     1000
209
210  -- << タイマー動作ハンドラ >>
211  --      cyclic_name      cychdr_addr      activation     interval
212  cyc     CYC_TIMER    _TimerDrive      TCY_OFF    1000
213
214  -- << ストップウォッチハンドラ (単位 : 秒) >>
215  --      cyclic_name      cychdr_addr          activation     interval
216  cyc     CYC_WATCH    _StopWatchDrive    TCY_OFF    1000
217
218  -- << ストップウォッチハンドラ (単位 : 1/100 秒) >>
219  --      cyclic_name         cychdr_addr          activation     interval
220  cyc     CYC_WATCH10    _StopWatchDrive    TCY_OFF    100
221
222  -- << スロットマシンハンドラ >>
223  --      cyclic_name      cychdr_addr      activation     interval
224  cyc     CYC_SLOT     _SlotDrive       TCY_OFF    70
225
226  -- << 温度・湿度センサー管理タスク起床ハンドラ >>
227  --      cyclic_name         cychdr_addr              activation     interval
228  cyc     CYC_MESURE     _MesuringManage      TCY_ON     5000
229
230
231  -- /************************************
232  --       割り込みハンドラ用スタック情報
233  -- ************************************/
234  --          stack_size
235  intstk      0x100:pool0
236
237
238  -- /*******************************
239  --       間接割り込みハンドラ情報
240  -- *******************************/
241  -- << タッチセンサー 感知割り込み (INTP131 = 0150H = 13) >>
242  --          interrupt_name      inthdr_address
243  inthdr      13                  _TouchSenser
244
245  -- << タクトスイッチ 1 割り込み (INTP110 = 00C0H = 4) >>
246  --          interrupt_name      inthdr_address
247  inthdr       4                  _TaktSwitch1
248
249  -- << タクトスイッチ 2 割り込み (INTP111 = 00D0H = 5) >>
250  --          interrupt_name      inthdr_address
251  inthdr       5                  _TaktSwitch2
252
253  -- << タクトスイッチ 3 割り込み (INTP112 = 00E0H = 6) >>
254  --          interrupt_name      inthdr_address
255  inthdr       6                  _TaktSwitch3
256
257  -- << タクトスイッチ 4 割り込み (INTP113 = 00F0H = 7) >>
```

```
--          interrupt_name      inthdr_address
inthdr      7                   _TaktSwitch4

-- << タクトスイッチ 5 割り込み (INTP141 = 0180H = 16) >>
--          interrupt_name      inthdr_address
inthdr      16                  _TaktSwitch5

-- << タクトスイッチ 6 割り込み (INTP141 = 0190H = 17) >>
--          interrupt_name      inthdr_address
inthdr      17                  _TaktSwitch6

-- << タクトスイッチ 7 割り込み (INTP142 = 01A0H = 18) >>
--          interrupt_name      inthdr_address
inthdr      18                  _TaktSwitch7

-- << タクトスイッチ 8 割り込み (INTP143 = 01B0H = 19) >>
--          interrupt_name      inthdr_address
inthdr      19                  _TaktSwitch8

-- << シリアル受信完了割り込み(UART0) (INTSR0 = 0220H = 26) >>
--          interrupt_name      inthdr_address
inthdr      26                  _MemoDrive

-- << シリアル送信完了割り込み(UART0) (INTST0 = 0230H = 27) >>
--          interrupt_name      inthdr_address
inthdr      27                  _SendCheck

-- << A/D コンバート終了割り込み (INTAD = 0270H = 31) >>
--          interrupt_name      inthdr_address
inthdr      31                  _IntADfinish
```

【 sit.cf 】(NEC 版)

```
1   #/*****************************************************************************
2   #   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   #*****************************************************************************
4   #   Module          :   sit.cf
5   #   Target          :   V853
6   #                   :   コンフィギュレーション・ファイル
7   #*****************************************************************************/
8
9   ser_def
10  rxsers RX850 V310
11
12  sit_def
13
14  -- /***************************
15  --          タスクの優先度範囲
16  -- ***************************/
17  --      priority_num
18  maxpri    13
19
20  -- /*********************************************
21  --          システムクロックのクロック割り込み要因
22  -- *********************************************/
23  --  For CA850 (INTCM4)
24  --          clkhdr_name
25  clkhdr    INTCM4
26
27  -- /******************
28  --          タスク情報
29  -- ******************/
30  --  << LED ダイナミックドライブタスク >>
31  --      task_name           task_address    stack_size    pri status        stacode   ei/di
32  tsk     TSK_LEDDRIVE    _LedDriveTask   180:pool1    3   TTS_DMT   0x0       ei
33
34  --  << 秒刻み LED 点灯・消灯タスク >>
35  --      task_name           task_address    stack_size    pri status        stacode   ei/di
36  tsk     TSK_LEDFLASH    _LedFlashTask   160:pool1    9   TTS_DMT   0x0       ei
37
38  --  << 現在時刻管理タスク >>
39  --      task_name           task_address    stack_size    pri status        stacode   ei/di
40  tsk     TSK_TIMECTRL    _TimeControl    180:pool1    7   TTS_DMT   0x0       ei
41
42  --  << タイマーカウントダウンタスク >>
43  --      task_name               task_address    stack_size    pri status        stacode   ei/di
44  tsk     TSK_TMRCONTROL      _TimerControl   180:pool1    10 TTS_DMT   0x0       ei
45
46  --  << ストップウォッチ管理タスク >>
47  --      task_name           task_address        stack_size    pri status        stacode   ei/di
48  tsk     TSK_SWATCHCTRL  _StopWatchControl 180:pool0    11 TTS_DMT   0x0       ei
49
```

```
50  -- << スロットマシン管理タスク >>
51  --      task_name          task_address     stack_size    pri status      stacode   ei/di
52  tsk     TSK_SLOTCTRL       _SlotControl     140:pool0     12 TTS_DMT      0x0       ei
53
54  -- << LCD 初期化タスク >>
55  --      task_name       task_address     stack_size    pri status      stacode   ei/di
56  tsk     TSK_INITLCD     _InitLCD         200:pool1     4  TTS_DMT      0x0       ei
57
58  -- << LCD 表示文字管理タスク(固定メッセージ) >>
59  --      task_name          task_address     stack_size    pri status      stacode   ei/di
60  tsk     TSK_LCDCHAR        _LCDChar         200:pool0     8  TTS_DMT      0x0       ei
61
62  -- << LCD 表示文字管理タスク(温度・湿度) >>
63  --      task_name          task_address     stack_size    pri status      stacode   ei/di
64  tsk     TSK_LCDCOND        _LCDCondition    200:pool0     9  TTS_DMT      0x0       ei
65
66  -- << 表示メッセージを LCD 表示タスクへ送信するタスク >>
67  --      task_name          task_address     stack_size    pri status      stacode   ei/di
68  tsk     TSK_LCDSEND        _LCDSendChar     200:pool1     6  TTS_DMT      0x0       ei
69
70  -- << LCD 表示タスク >>
71  --      task_name          task_address     stack_size    pri status      stacode   ei/di
72  tsk     TSK_LCDDISP        _LCDDispChar     200:pool1     5  TTS_DMT      0x0       ei
73
74  -- << 自動的に LCD 表示を温度・湿度表示にするタスク >>
75  --      task_name             task_address        stack_size    pri status      stacode   ei/di
76  tsk     TSK_AUTOCHANGE        _AutoDispChange     200:pool1     8  TTS_DMT      0x0       ei
77
78  -- << メモ管理タスク >>
79  --      task_name             task_address     stack_size    pri status      stacode   ei/di
80  tsk     TSK_MEMOCONTROL       _MemoControl     200:pool0     7  TTS_DMT      0x0       ei
81
82  -- << A/D コンバートタスク >>
83  --      task_name          task_address        stack_size    pri status      stacode   ei/di
84  tsk     TSK_MESURE         _MesureCondition    140:pool0     5  TTS_DMT      0x0       ei
85
86  -- << ステッピングモーター回転タスク >>
87  --      task_name          task_address     stack_size    pri status      stacode   ei/di
88  tsk     TSK_STEPPER        _Stepper         140:pool0     13 TTS_DMT      0x0       ei
89
90  -- << シリアル受信結果送信タスク >>
91  --      task_name          task_address        stack_size    pri status      stacode   ei/di
92  tsk     TSK_UARTSEND       _UARTCharSend    140:pool1     6  TTS_DMT      0x0       ei
93
94  -- << ブザータスク >>
95  --      task_name          task_address     stack_size    pri status      stacode   ei/di
96  tsk     TSK_BUZZER         _Buzzer          140:pool0     13 TTS_DMT      0x0       ei
97
98  -- << 割り込み制御タスク >>
99  --      task_name             task_address     stack_size    pri status      stacode   ei/di
100 tsk     TSK_INTCONTROL        _IntControl      160:pool0     4  TTS_DMT      0x0       ei
101
```

```
102  -- /*************************
103  --       イベントフラグ情報
104  -- *************************/
105  --  << LCD 表示文字用イベントフラグ >>
106  --      evf_name
107  flg     FLG_LCDCHAR
108
109  --  << 現在温度通知イベントフラグ >>
110  --      evf_name
111  flg     FLG_TEMPERATURE
112
113  --  << シリアル受信結果送信用イベントフラグ >>
114  --      evf_name
115  flg     FLG_MEMOCHECK
116
117  -- /*********************
118  --       セマフォ情報
119  -- *********************/
120  --  << LCD 表示管理 >>
121  --          semaph_name          init_count
122  sem         SEM_LCDSEND          1
123
124  --  << LCD 表示管理 >>
125  --          semaph_name          init_count
126  sem         SEM_LCDDISP          0
127
128  --  << ストップウォッチ時の LED 点灯タイミング変数初期化処理用 >>
129  --          semaph_name          init_count
130  sem         SEM_LEDTIMING        0
131
132  --
133  -- /*************************
134  --       メールボックス情報
135  -- *************************/
136  --  << 7SegLED 表示用メールボックス(下一桁) >>
137  --          mbx_name             mwai_opt
138  mbx         MBX_FIGURE1          TA_MFIFO
139
140  --  << 7SegLED 表示用メールボックス(下二桁) >>
141  --          mbx_name             mwai_opt
142  mbx         MBX_FIGURE2          TA_MFIFO
143
144  --  << 7SegLED 表示用メールボックス(下三桁) >>
145  --          mbx_name             mwai_opt
146  mbx         MBX_FIGURE3          TA_MFIFO
147
148  --  << 7SegLED 表示用メールボックス(下四桁) >>
149  --          mbx_name             mwai_opt
150  mbx         MBX_FIGURE4          TA_MFIFO
151
152  --  << LED 点灯用メールボックス >>
153  --          mbx_name         mwai_opt
```

```
154  mbx       MBX_LED       TA_MFIFO
155
156  --  << セッティング用メールボックス(下一桁) >>
157  --          mbx_name      mwai_opt
158  mbx       MBX_SET1      TA_MFIFO
159
160  --  << セッティング用メールボックス(下二桁) >>
161  --          mbx_name      mwai_opt
162  mbx       MBX_SET2      TA_MFIFO
163
164  --  << セッティング用メールボックス(下三桁) >>
165  --          mbx_name      mwai_opt
166  mbx       MBX_SET3      TA_MFIFO
167
168  --  << セッティング用メールボックス(下四桁) >>
169  --          mbx_name      mwai_opt
170  mbx       MBX_SET4      TA_MFIFO
171
172  --  << LCD 表示用メモリ領域 送受信メールボックス >>
173  --          mbx_name          mwai_opt
174  mbx       MBX_TO_LCD    TA_MFIFO
175
176  --  << LCD 表示用メモリ領域 送信用メールボックス >>
177  --          mbx_name              mwai_opt
178  mbx       MBX_FROM_LCD    TA_MFIFO
179
180  --  << 温度・湿度データ送信メールボックス >>
181  --          mbx_name          mwai_opt
182  mbx       MBX_ADDATA    TA_MFIFO
183
184  -- /*************************
185  --        メモリプール情報
186  -- *************************/
187  --  << 7SegLED 表示メールボックス用 >>
188  --          mpl_name      block_size      block_num
189  mpf       MPL_LED       0xc:pool0       10
190
191  --  << 7SegLED 表示更新メールボックス用 >>
192  --          mpl_name      block_size      block_num
193  mpf       MPL_SET       0xc:pool0       5
194
195  --  << LCD 表示文字列 送信メールボックス用 >>
196  --          mpl_name      block_size      block_num
197  mpf       MPL_LCD       0x28:pool0      1
198
199  --  << LCD 表示文字列 送信メールボックス用 >>
200  --          mpl_name          block_size    block_num
201  mpf       MPL_ADCONV    0xc:pool0     1
202
203  -- /************************
204  --        周期ハンドラ情報
205  -- ************************/
```

```
206  -- << 現在時刻保持ハンドラ >>
207  --      cyclic_name         cychdr_addr     activation    interval
208  cyc     CYC_TICKTIME    _TickTime       TCY_ON    1000
209
210  -- << タイマー動作ハンドラ >>
211  --      cyclic_name     cychdr_addr         activation    interval
212  cyc     CYC_TIMER   _TimerDrive         TCY_OFF   1000
213
214  -- << ストップウォッチハンドラ (単位 : 秒) >>
215  --      cyclic_name         cychdr_addr         activation    interval
216  cyc     CYC_WATCH       _StopWatchDrive TCY_OFF   1000
217
218  -- << ストップウォッチハンドラ (単位 : 1/100 秒) >>
219  --      cyclic_name         cychdr_addr         activation    interval
220  cyc     CYC_WATCH10     _StopWatchDrive TCY_OFF   100
221
222  -- << スロットマシンハンドラ >>
223  --      cyclic_name     cychdr_addr     activation    interval
224  cyc     CYC_SLOT    _SlotDrive      TCY_OFF   70
225
226  -- << 温度・湿度センサー管理タスク起床ハンドラ >>
227  --      cyclic_name         cychdr_addr         activation    interval
228  cyc     CYC_MESURE      _MesuringManage TCY_ON    5000
229
230
231  -- /*************************************
232  --      割り込みハンドラ用スタック情報
233  -- *************************************/
234  --          stack_size
235  intstk      0x100:pool0
236
237
238  -- /*******************************
239  --      間接割り込みハンドラ情報
240  -- *******************************/
241  -- << タッチセンサー 感知割り込み (INTP131) >>
242  --          interrupt_name      inthdr_address
243  inthdr      INTP131             _TouchSenser
244
245  -- << タクトスイッチ 1 割り込み (INTP110) >>
246  --          interrupt_name      inthdr_address
247  inthdr      INTP110             _TaktSwitch1
248
249  -- << タクトスイッチ 2 割り込み (INTP111) >>
250  --          interrupt_name      inthdr_address
251  inthdr      INTP111             _TaktSwitch2
252
253  -- << タクトスイッチ 3 割り込み (INTP112) >>
254  --          interrupt_name      inthdr_address
255  inthdr      INTP112             _TaktSwitch3
256
257  -- << タクトスイッチ 4 割り込み (INTP113) >>
```

```
258     --          interrupt_name      inthdr_address
259     inthdr      INTP113             _TaktSwitch4
260
261     -- << タクトスイッチ 5 割り込み (INTP140) >>
262     --          interrupt_name      inthdr_address
263     inthdr      INTP140             _TaktSwitch5
264
265     -- << タクトスイッチ 6 割り込み (INTP141) >>
266     --          interrupt_name      inthdr_address
267     inthdr      INTP141             _TaktSwitch6
268
269     -- << タクトスイッチ 7 割り込み (INTP142) >>
270     --          interrupt_name      inthdr_address
271     inthdr      INTP142             _TaktSwitch7
272
273     -- << タクトスイッチ 8 割り込み (INTP143) >>
274     --          interrupt_name      inthdr_address
275     inthdr      INTP143             _TaktSwitch8
276
277     -- << シリアル受信完了割り込み(UART0) (INTSR0) >>
278     --          interrupt_name      inthdr_address
279     inthdr      INTSR0              _MemoDrive
280
281     -- << シリアル送信完了割り込み(UART0) (INTST0) >>
282     --          interrupt_name      inthdr_address
283     inthdr      INTST0              _SendCheck
284
285     -- << A/D コンバート終了割り込み (INTAD = 0270H = 31) >>
286     --          interrupt_name      inthdr_address
287     inthdr      INTAD               _IntADfinish
```

【 common.h 】

```
1   /*****************************************************************************
2      Realtime OS nucleus [RX850 Ver.3.1]  アプリケーション・ノート
3   *****************************************************************************
4      Module      :  common.h
5      Target      :  V853
6                   :  ヘッダファイル
7   *****************************************************************************/
8
9   /************************************
10         7 セグメント LED 用マクロ
11  ************************************/
12
13  /* 7 セグメント LED 表示モード設定フラグ */
14
15  #define     CURRENT_TIME    0x1     /* 現在時刻モード          */
16  #define     TIMER           0x2     /* タイマーモード          */
17  #define     STOPWATCH       0x4     /* ストップウォッチモード  */
18  #define     SLOT            0x8     /* スロットマシンモード    */
19  #define     LEDMODE_MASK    0xf     /* 全 LED モードマスク     */
20
21  /* 7 セグメント LED 桁   */
22
23  #define     FIGURE_1        0x1     /* 下一桁目          */
24  #define     FIGURE_2        0x2     /* 下二桁目          */
25  #define     FIGURE_3        0x4     /* 下三桁目          */
26  #define     FIGURE_4        0x8     /* 下四桁目          */
27  #define     SELECT_LED      0x2     /* LED               */
28  #define     FIGURE_MASK     0xf     /* 全桁マスク        */
29  #define     FIGURE_CLR      0x0     /* 0 マスク          */
30  #define     NODISP          0xa     /* LED に表示しない  */
31
32  /* 現在時刻動作フラグ */
33
34  #define     TIME_STOP   0x0     /* 現在時刻カウントストップ */
35  #define     TIME_START  0x1     /* 現在時刻カウントスタート */
36  #define     TIME_TICK   0x2     /* 現在時刻カウント中       */
37  #define     TIME_FIX    0x4     /* 現在時刻修正中           */
38
39  /* タイマー動作フラグ */
40
41  #define     TIMER_STOP      0x0     /* タイマー動作終了    */
42  #define     TIMER_START     0x1     /* タイマー動作開始    */
43  #define     TIMER_WORKING   0x2     /* タイマー動作中      */
44  #define     TIMER_FINISH    0x4     /* タイマー完了        */
45  #define     TIMER_INTERRUPT 0x8     /* タイマー中断        */
46
47  /* ストップウォッチ動作フラグ */
```

```
48
49  #define     SWATCH_STOP         0x0     /* ストップウォッチ動作終了     */
50  #define     SWATCH_START        0x1     /* ストップウォッチ動作開始     */
51  #define     SWATCH_WORKING      0x2     /* ストップウォッチ動作中       */
52  #define     SWATCH_LAP          0x4     /* ストップウォッチラップ表示   */
53  #define     MODE_1              0x1     /* 1 秒単位モード               */
54  #define     MODE_10             0x2     /* 1/10 秒単位モード            */
55
56  /* スロットマシン動作フラグ */
57
58  #define     SLOT_STOP           0x0     /* スロットマシン ストップ      */
59  #define     SLOT_WORKING        0x2     /* スロットマシン 動作中        */
60  #define     SLOT_FIGURE1        0x1     /* スロット下一桁動作フラグ     */
61  #define     SLOT_FIGURE2        0x2     /* スロット下二桁動作フラグ     */
62  #define     SLOT_FIGURE3        0x4     /* スロット下三桁動作フラグ     */
63  #define     SLOT_ALLCLEAR       0x0     /* スロット動作クリア           */
64  #define     SLOT_ALLSTOP        0x7     /* スロット動作ストップ         */
65
66  /* タイマー・ストップウォッチ・スロット 共通フラグ  */
67
68  #define     STOP        0x0     /* ストップ                     */
69  #define     START       0x1     /* スタート                     */
70  #define     SETTING     0x2     /* セッティング中 (タイマー)    */
71  #define     FIX         0x2     /* 時間調整中                   */
72  #define     TICK        0x4     /* 現在時刻動作中               */
73  #define     COUNT       0x4     /* カウント動作                 */
74  #define     ROTATION    0x4     /* スロットマシン回転動作       */
75  #define     INIT        0x8     /* 初期化命令                   */
76  #define     DISPLAY     0x10    /* 現在値表示命令               */
77  #define     LAP         0x40    /* ラップ表示                   */
78  #define     CLEAR       0x80    /* クリア命令                   */
79
80  /* 7 セグメント LED ダイナミックドライブ用マクロ */
81
82  #define     LED_INTERVAL    4       /* ダイナミックドライブ間隔     */
83  #define     FIGURE1_ON      0x1     /* 下一桁目に電源切り替え       */
84  #define     FIGURE2_ON      0x2     /* 下二桁目に電源切り替え       */
85  #define     FIGURE3_ON      0x4     /* 下三桁目に電源切り替え       */
86  #define     FIGURE4_ON      0x8     /* 下四桁目に電源切り替え       */
87  #define     LED_SWITCH      0x3     /* LED セット                   */
88  #define     LED_ON          0x10    /* LED 電源 ON                  */
89  #define     LED_FLASH       500     /* LED 点灯・消灯間隔           */
90
91  /* 桁上がりフラグ初期化用  */
92
93  #define     FLGINIT     0x1     /* 初期化・更正後          */
94
95  /* 即座表示要求フラグ */
96
```

```
97  #define     DISPLAY_NOW     0x1     /* 即表示命令          */
98
99  /**************************
100         LCD 表示用マクロ
101 **************************/
102 #define     LCD_INITWAIT    10    /* LCD 初期化ウェイト       */
103 #define     LCD_FUNCWAIT    5     /* LCD Function Set ウェイト */
104 #define     LCD_SETWAIT     10    /* LCD セット間ウェイト      */
105 #define     LCD_E_WAIT      1     /* LCD イネーブルウェイト    */
106
107 #define     LCD_CLEAR       0x1     /* LCD 表示クリア          */
108 #define     LCD_DISPCTRL    0xe     /* 表示 ON/OFF コントロール */
109 #define     LCD_ENTRY       0x6     /* エントリーモード        */
110 #define     LCD_LINE1S      0x80    /* LCD 1 行目先頭          */
111 #define     LCD_LINE1E      0x8f    /* LCD 1 行目最後尾        */
112 #define     LCD_LINE2S      0xc0    /* LCD 2 行目先頭          */
113 #define     LCD_LINE2E      0xcf    /* LCD 2 行目最後尾        */
114 #define     LCD_LEFT        0x10    /* LCD カーソル左シフト    */
115 #define     LCD_RIGHT       0x14    /* LCD カーソル右シフト    */
116 #define     LCD_CURSOR_ON   0xe     /* LCD カーソル ON         */
117 #define     LCD_CURSOR_OFF  0xc     /* LCD カーソル・点滅 OFF  */
118 #define     LCD_BLINK_ON    0xd     /* LCD 点滅 ON             */
119
120 #define     LCD_FUNCSET     0x3     /* ファンクションセット          */
121 #define     LCD_FUNCSET1    0x2     /* ファンクションセット(4 bit)   */
122 #define     LCD_FUNCSET2    0x28    /* ファンクションセット(4 bit)   */
123 #define     LCD_ENABLE      0x4     /* イネーブルスイッチ            */
124 #define     ENABLE          0x1     /* イネーブル                    */
125 #define     DISABLE         0x0     /* ディスエーブル                */
126
127 #define     INTERFACE4      1       /* 4 bit 転送          */
128 #define     INTERFACE8      2       /* 8 bit 転送          */
129
130 #define     LCD_RS_ON       0x2     /* RS スイッチ         */
131
132 #define     LCD_CHARNUM     32      /* LCD 表示文字数          */
133 #define     AUTOCHANGE      10000   /* LCD 表示自動切替間隔(ms) */
134
135 /**********************************
136         A/D コンバーター用マクロ
137 **********************************/
138 #define     ADM0_CE         0x80    /* ADM0 CE ビット      */
139
140 #define     TEMPERATURE     0x1     /* 温度      */
141 #define     HUMIDITY        0x2     /* 湿度      */
142
143 /**************************
144         メモ機能用マクロ
145 **************************/
```

```
146  #define     RCV_ENABLE      0x40    /* シリアル受信許可      */
147  #define     SND_ENABLE      0x80    /* シリアル送信許可      */
148
149  #define     MEMO_WRITING    0x1     /* メモ書き込みモード    */
150  #define     MEMO_ALLDISP    0x2     /* メモ全表示            */
151  #define     MEMO_ALLCLEAR   0x4     /* メモ全消去            */
152  #define     MEMO_INIT       0x8     /* メモ初期化            */
153
154  /***************************
155        文字コードマクロ
156  ***************************/
157  #define     SPACE     0x20    /* スペース    */
158  #define     LEFT      0x02    /* 左          */
159  #define     RIGHT     0x06    /* 右          */
160  #define     UP        0x10    /* 上          */
161  #define     DOWN      0x0e    /* 下          */
162  #define     CR        0x0d    /* リターン (MS-DOS : 0x0d)  */
163  #define     BS        0x08    /* バックスペース            */
164  #define     DELETE    0x04    /* 削除  (ctrl + d)          */
165  #define     DELKEY    0x7f    /* 削除  (DEL)               */
166
167  /************************
168        ブザー用マクロ
169  ************************/
170  #define     BZ_INTERVAL1    0x100   /* ブザー間隔 1      */
171  #define     BZ_INTERVAL2    0x400   /* ブザー間隔 2      */
172  #define     BUZZER          0x20    /* ブザーON/OFF      */
173
174  /**********************
175        共通マクロ
176  **********************/
177  #define     ON        0x1     /* スイッチ ON       */
178  #define     OFF       0x0     /* スイッチ OFF      */
179
180  /***************************************
181        割り込み制御フラグ用マクロ
182  ***************************************/
183  #define     SW1       0x1     /* タクトスイッチ 1  */
184  #define     SW2       0x2     /* タクトスイッチ 2  */
185  #define     SW3       0x4     /* タクトスイッチ 3  */
186  #define     SW4       0x8     /* タクトスイッチ 4  */
187  #define     SW5       0x10    /* タクトスイッチ 5  */
188  #define     SW6       0x20    /* タクトスイッチ 6  */
189  #define     SW7       0x40    /* タクトスイッチ 7  */
190  #define     SW8       0x80    /* タクトスイッチ 8  */
191  #define     ALLSW     0xff    /* 全タクトスイッチ  */
192
193  /*************************************
194        アイドル・モード用マクロ
195  *************************************/
```

```
196  #define   HALT_MODE  0
197  #define   STOP_MODE  1
198  #define   IDLE_MODE  2
199
200  /*****************************************
201        ステッピングモータ駆動用マクロ
202  *****************************************/
203  #define      STEPPER      0x08     /* ステップ回転 ON/OFF  */
204  #define      TEMPINIT     0x25     /* ステップ間隔         */
205
206  /*****************************
207        各種構造体定義
208  *****************************/
209
210  struct LEDMAIL   {      /* メールボックス・メッセージ構造体 */
211      int     addr;       /* メッセージリンク領域          */
212      char    pri;        /* メッセージ優先度              */
213      int     message;    /* メッセージ領域                */
214  };                      /*  LED 表示情報用               */
215
216  struct DATAMAIL  {      /* メールボックス・メッセージ構造体 */
217      int     addr;       /* メッセージリンク領域          */
218      char    pri;        /* メッセージ優先度              */
219      char    data[2];    /* メッセージ領域                */
220  };                      /* 気温・湿度情報用              */
221
222  struct LCDMAIL   {      /* メールボックス・メッセージ構造体 */
223      int     addr;       /* メッセージリンク領域          */
224      char    pri;        /* メッセージ優先度              */
225      char    data[32];   /* メッセージ領域へのポインタ    */
226  };                      /*  LCD 表示情報用               */
227
228  struct FIGURE {         /* LED 桁構造体  */
229      int     figure1;    /* 下一桁目      */
230      int     figure2;    /* 下二桁目      */
231      int     figure3;    /* 下三桁目      */
232      int     figure4;    /* 下四桁目      */
233  };
234
235  struct FLAG {           /* 桁調整フラグ構造体            */
236      int     move0;      /* 秒送りからの桁上がりフラグ    */
237      int     move1;      /* 一桁目からの桁上がりフラグ    */
238      int     move2;      /* 二桁目からの桁上がりフラグ    */
239      int     move3;      /* 三桁目からの桁上がりフラグ    */
240      int     move4;      /* 四桁目からの桁上がりフラグ    */
241      int     alloff;     /* 全桁 0 フラグ                 */
242  };
243
244  struct  LEDBLOCK {              /* LED 転送用メッセージ構造体 */
245      struct  LEDMAIL     *led1;  /* 下一桁用メッセージ領域     */
```

```
246     struct  LEDMAIL     *led2;    /* 下二桁用メッセージ領域   */
247     struct  LEDMAIL     *led3;    /* 下三桁用メッセージ領域   */
248     struct  LEDMAIL     *led4;    /* 下四桁用メッセージ領域   */
249 };
250
251 typedef struct LEDMAIL MAIL;      /* 型定義   */
252
253 /*****************************
254         extern 関数宣言
255 *****************************/
256 extern void   InitFunc (void);              /* ハード初期化関数              */
257
258 extern void   DynamicDrive (void);          /* ダイナミックドライブハンドラ      */
259 extern void   LedDriveTask (INT);           /* ダイナミックドライブタスク  */
260
261 extern void   LedFlashTask (INT);           /* 秒刻み LED 点灯タスク         */
262
263 extern void   TickTime (void);              /* 現在時刻管理ハンドラ          */
264 extern void   TimeControl (INT);            /* 現在時刻管理タスク            */
265 extern void   SetTime (int);                /* 現在時刻調整関数              */
266 extern void   TimeCount (int, int);         /* 現在時刻カウント関数          */
267
268 extern void   TimerDrive (void);            /* タイマー管理ハンドラ          */
269 extern void   TimerControl (INT);           /* タイマー管理タスク            */
270 extern void   SetTimerCount (int);          /* タイマー設定関数              */
271 extern void   TimerCountDown (int, int);    /* タイマーカウントダウン関数  */
272
273 extern void   StopWatchDrive (void);        /* ストップウォッチ管理ハンドラ   */
274 extern void   StopWatchControl (INT);       /* ストップウォッチ管理タスク      */
275 extern void   StopWatchAction (int);        /* ストップウォッチ動作管理関数   */
276 extern void   StopWatchCount (int, int);    /* ストップウォッチカウント関数   */
277
278 extern void   SlotDrive (void);             /* スロット管理ハンドラ       */
279 extern void   SlotControl (INT);            /* スロット管理タスク         */
280 extern void   SlotRotation (int, int);      /* スロット回転関数           */
281
282 extern void   InitLCD (INT);                  /* LCD 初期化タスク                 */
283 extern void   LCDControl (char, int);         /* LCD 初期化関数                   */
284 extern void   LCDCharSet (char);              /* 8 ビット文字分割セット関数       */
285 extern void   LCDChar (INT);                  /* LCD 表示内容管理タスク           */
286 extern void   LCDCondition (INT);             /* LCD 表示内容管理タスク(環境)     */
287 extern void   LCDDispChar (INT);              /* 一文字送信・表示タスク           */
288 extern void   LCDSendChar (INT);              /* 表示内容 全送信タスク            */
289 extern void   LCDTimeChar (char *);           /* 文字定義関数(現在時刻)           */
290 extern void   LCDTimeFixChar (char *);        /* 文字定義関数(現在時刻調整)       */
291 extern void   LCDTimerChar (char *);          /* 文字定義関数(タイマー)           */
292 extern void   LCDTimerWorkChar (char *);      /* 文字定義関数(タイマー動作)       */
293 extern void   LCDStopWatchChar1 (char *);     /* 文字定義関数(1/1  StopWatch)     */
294 extern void   LCDStopWatchChar10 (char *);    /* 文字定義関数(1/10 StopWatch)     */
```

```
295  extern void   LCDSlotChar (char *);          /* 文字定義関数(スロット)          */
296  extern void   LCDConditionChar (char *, char, char);
297                                               /* 文字定義関数(気温・湿度)        */
298  extern void   LCD_RS (int);                  /* LCD RS 管理関数                 */
299  extern void   LCD_E (int);                   /* LCD E 管理関数                  */
300  extern void   LCDSetvalue (char);            /* LCD 4 ビット送信関数            */
301  extern void   LCDDataSet (char, int);        /* LCD データセット関数            */
302  extern void   ConvertAscii (char, char *);   /* アスキーコード変換関数          */
303  extern void   AutoDispChange (INT);          /* LCD 表示自動切り替えタスク      */
304  extern unsigned short   DataCorrection (int, unsigned short);
305                                               /* 気温・湿度データ更正関数        */
306
307  extern INT       MemoDrive (void);           /* メモ機能管理ハンドラ            */
308  extern void   MemoControl (INT);             /* メモ機能管理タスク              */
309  extern void   MemoManager (int);             /* メモ管理関数                    */
310  extern int    CodeManager (char, char *, int);
311                                               /* 受信コード別処理関数            */
312  extern void   MemoInit (char *);             /* 文字定義関数(メモ初期化時)      */
313
314  extern void   Buzzer (INT);                  /* ブザー管理タスク                */
315
316  extern void   Stepper (INT);                 /* ステッパー回転タスク            */
317  extern void   OneStep (void);                /* ステッパー ワンステップ関数     */
318  extern unsigned long   RotSpeed (short);     /* ステッパー回転速度調整関数      */
319
320  extern void   ToMBX (int, int, int, MAIL *);      /* メッセージ転送関数         */
321  extern int    FromMBX (int, int, MAIL *);         /* メッセージ受信関数         */
322
323  extern INT       TaktSwitch1 (void);    /* タクトスイッチ 1 割り込みハンドラ*/
324  extern INT       TaktSwitch2 (void);    /* タクトスイッチ 2 割り込みハンドラ*/
325  extern INT       TaktSwitch3 (void);    /* タクトスイッチ 3 割り込みハンドラ*/
326  extern INT       TaktSwitch4 (void);    /* タクトスイッチ 4 割り込みハンドラ*/
327  extern INT       TaktSwitch5 (void);    /* タクトスイッチ 5 割り込みハンドラ*/
328  extern INT       TaktSwitch6 (void);    /* タクトスイッチ 6 割り込みハンドラ*/
329  extern INT       TaktSwitch7 (void);    /* タクトスイッチ 7 割り込みハンドラ*/
330  extern INT       TaktSwitch8 (void);    /* タクトスイッチ 8 割り込みハンドラ*/
331  extern INT       TouchSenser (void);    /* タッチセンサー 割り込みハンドラ  */
332
333  extern void   idle_handler(void);            /* アイドル・ハンドラ              */
334  /*****************************
335          extern 変数宣言
336  *****************************/
337  extern int    flg_ledmode;          /* 7SegLED 使用モード認識フラグ  */
338  extern int    flg_figurefix;        /* 調整桁認識フラグ              */
339  extern int    flg_time;             /* 使用モード、時刻調整フラグ    */
340  extern int    flg_timer;            /* タイマー動作フラグ            */
341  extern int    flg_stopwatch;        /* ストップウォッチ動作フラグ    */
342  extern int    flg_slot;             /* スロットマシン動作フラグ      */
343  extern int    flg_slotfigure;       /* スロットマシン動作桁判別      */
```

```
344  extern int   flg_interrupt;    /* 割り込み制御フラグ          */
345
346  extern int   flg_condition;    /* 温度・湿度表示フラグ        */
347  extern int   flg_memo;         /* メモ表示状態フラグ          */
348  extern int   flg_memodisp;     /* メモ表示フラグ              */
349
350  extern int   flg_laptime;      /* ラップタイム表示フラグ      */
351  extern int   flg_swatchmode;   /* ストップウォッチモードフラグ */
352
353  extern int   idlemode;         /* アイドル・モード            */
```

【 port.h 】

```
/*****************************************************************************
   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
*****************************************************************************
   Module          :  port.h
   Target          :  V853
                   :  V853 周辺 I/O レジスタ定義
*****************************************************************************/
#define     P0          0xfffff000   /* ポート 0                              */
#define     P1          0xfffff002   /* ポート 1                              */
#define     P2          0xfffff004   /* ポート 2                              */
#define     P3          0xfffff006   /* ポート 3                              */
#define     P4          0xfffff008   /* ポート 4                              */
#define     P5          0xfffff00a   /* ポート 5                              */
#define     P6          0xfffff00c   /* ポート 6                              */
#define     P7          0xfffff00e   /* ポート 7                              */
#define     P9          0xfffff012   /* ポート 9                              */
#define     P11         0xfffff016   /* ポート 11                             */
#define     PM0         0xfffff020   /* ポート 0 モード・レジスタ             */
#define     PM1         0xfffff022   /* ポート 1 モード・レジスタ             */
#define     PM2         0xfffff024   /* ポート 2 モード・レジスタ             */
#define     PM3         0xfffff026   /* ポート 3 モード・レジスタ             */
#define     PM4         0xfffff028   /* ポート 4 モード・レジスタ             */
#define     PM5         0xfffff02a   /* ポート 5 モード・レジスタ             */
#define     PM6         0xfffff02c   /* ポート 6 モード・レジスタ             */
#define     PM9         0xfffff032   /* ポート 9 モード・レジスタ             */
#define     PM11        0xfffff036   /* ポート 11 モード・レジスタ            */
#define     PMC0        0xfffff040   /* ポート 0 モード・コントロール・レジスタ */
#define     PMC1        0xfffff042   /* ポート 1 モード・コントロール・レジスタ */
#define     PMC2        0xfffff044   /* ポート 2 モード・コントロール・レジスタ */
#define     PMC3        0xfffff046   /* ポート 3 モード・コントロール・レジスタ */
#define     MM          0xfffff04c   /* メモリ拡張モード・レジスタ            */
#define     PMC11       0xfffff056   /* ポート 11 モード・コントロール・レジスタ */
#define     PCM         0xfffff05c   /* ポート・コントロール・モード・レジスタ  */
#define     PUO         0xfffff05e   /* プルアップ抵抗オプション・レジスタ      */
#define     DWC         0xfffff060   /* データ・ウェイト・コントロール・レジスタ */
#define     BCC         0xfffff062   /* バス・サイクル・コントロール・レジスタ  */
#define     PSC         0xfffff070   /* パワー・セーブ・コントロール・レジスタ  */
#define     CKC         0xfffff072   /* クロック・コントロール・レジスタ        */
#define     SYS         0xfffff078   /* システム・ステータス・レジスタ          */
#define     BRGC0       0xfffff084   /* ボー・レート・ジェネレータ
                                        コンペア・レジスタ 0                  */
#define     BPRM0       0xfffff086   /* ボー・レート・ジェネレータ
                                        プリスケーラ・モード・レジスタ 0      */
#define     CSIM0       0xfffff088   /* クロック同期式シリアル・インタフェース
                                        モード・レジスタ 0                    */
```

```
46  #define   SIO0     0xfffff08a  /* シリアル I/O シフト・レジスタ 0          */
47  #define   BRGC1    0xfffff094  /* ボー・レート・ジェネレータ
48                                          コンペア・レジスタ 1  */
49  #define   BPRM1    0xfffff096  /* ボー・レート・ジェネレータ
50                                       プリスケーラ・モード・レジスタ 1*/
51  #define   CSIM1    0xfffff098  /* クロック同期式シリアル・インタフェース
52                                          モード・レジスタ 1  */
53  #define   SIO1     0xfffff09a  /* シリアル I/O シフト・レジスタ 1          */
54  #define   BRGC2    0xfffff0a4  /* ボー・レート・ジェネレータ
55                                          コンペア・レジスタ 2  */
56  #define   BPRM2    0xfffff0a6  /* ボー・レート・ジェネレータ
57                                       プリスケーラ・モード・レジスタ 2*/
58  #define   CSIM2    0xfffff0a8  /* クロック同期式シリアル・インタフェース
59                                          モード・レジスタ 2  */
60  #define   SIO2     0xfffff0aa  /* シリアル I/O シフト・レジスタ 2          */
61  #define   CSIM3    0xfffff0b8  /* クロック同期式シリアル・インタフェース
62                                          モード・レジスタ 3  */
63  #define   SIO3     0xfffff0ba  /* シリアル I/O シフト・レジスタ 3          */
64  #define   ASIM00   0xfffff0c0  /* アシンクロナス・シリアル・インタフェース
65                                          モード・レジスタ 00   */
66  #define   ASIM01   0xfffff0c2  /* アシンクロナス・シリアル・インタフェース
67                                          モード・レジスタ 01   */
68  #define   ASIS0    0xfffff0c4  /* アシンクロナス・シリアル・インタフェース
69                                        ステータス・レジスタ 0   */
70  #define   RXB0     0xfffff0c8  /* 受信バッファ 0  (9 ビット)               */
71  #define   RXB0L    0xfffff0ca  /* 受信バッファ 0L (下位 8 ビット)          */
72  #define   TXS0     0xfffff0cc  /* 送信シフトレジスタ 0  (9 ビット)         */
73  #define   TXS0L    0xfffff0ce  /* 送信シフトレジスタ 0L (下位 8 ビット)    */
74  #define   ASIM10   0xfffff0d0  /* アシンクロナス・シリアル・インタフェース
75                                          モード・レジスタ 10   */
76  #define   ASIM11   0xfffff0d2  /* アシンクロナス・シリアル・インタフェース
77                                          モード・レジスタ 11*/
78  #define   ASIS1    0xfffff0d4  /* アシンクロナス・シリアル・インタフェース
79                                        ステータス・レジスタ 0   */
80  #define   RXB1     0xfffff0d8  /* 受信バッファ 1  (9 ビット)               */
81  #define   RXB1L    0xfffff0da  /* 受信バッファ 1L (下位 8 ビット)          */
82  #define   TXS1     0xfffff0dc  /* 送信シフトレジスタ 1  (9 ビット)         */
83  #define   TXS1L    0xfffff0de  /* 送信シフトレジスタ 1L (下位 8 ビット)    */
84  #define   OVIC11   0xfffff100  /* 割り込み制御レジスタ                     */
85  #define   OVIC12   0xfffff102  /* 割り込み制御レジスタ                     */
86  #define   OVIC13   0xfffff104  /* 割り込み制御レジスタ                     */
87  #define   OVIC14   0xfffff106  /* 割り込み制御レジスタ                     */
88  #define   P11IC0   0xfffff108  /* 割り込み制御レジスタ                     */
89  #define   P11IC1   0xfffff10a  /* 割り込み制御レジスタ                     */
90  #define   P11IC2   0xfffff10c  /* 割り込み制御レジスタ                     */
91  #define   P11IC3   0xfffff10e  /* 割り込み制御レジスタ                     */
92  #define   P12IC0   0xfffff110  /* 割り込み制御レジスタ                     */
93  #define   P12IC1   0xfffff112  /* 割り込み制御レジスタ                     */
```

```
 94  #define   P12IC2   0xfffff114  /* 割り込み制御レジスタ                    */
 95  #define   P12IC3   0xfffff116  /* 割り込み制御レジスタ                    */
 96  #define   P13IC0   0xfffff118  /* 割り込み制御レジスタ                    */
 97  #define   P13IC1   0xfffff11a  /* 割り込み制御レジスタ                    */
 98  #define   P13IC2   0xfffff11c  /* 割り込み制御レジスタ                    */
 99  #define   P13IC3   0xfffff11e  /* 割り込み制御レジスタ                    */
100  #define   P14IC0   0xfffff120  /* 割り込み制御レジスタ                    */
101  #define   P14IC1   0xfffff122  /* 割り込み制御レジスタ                    */
102  #define   P14IC2   0xfffff124  /* 割り込み制御レジスタ                    */
103  #define   P14IC3   0xfffff126  /* 割り込み制御レジスタ                    */
104  #define   CMIC4    0xfffff128  /* 割り込み制御レジスタ                    */
105  #define   CSIC0    0xfffff12a  /* 割り込み制御レジスタ                    */
106  #define   CSIC1    0xfffff12c  /* 割り込み制御レジスタ                    */
107  #define   CSIC2    0xfffff12e  /* 割り込み制御レジスタ                    */
108  #define   CSIC3    0xfffff130  /* 割り込み制御レジスタ                    */
109  #define   SEIC0    0xfffff132  /* 割り込み制御レジスタ                    */
110  #define   SRIC0    0xfffff134  /* 割り込み制御レジスタ                    */
111  #define   STIC0    0xfffff136  /* 割り込み制御レジスタ                    */
112  #define   SEIC1    0xfffff138  /* 割り込み制御レジスタ                    */
113  #define   SRIC1    0xfffff13a  /* 割り込み制御レジスタ                    */
114  #define   STIC1    0xfffff13c  /* 割り込み制御レジスタ                    */
115  #define   ADIC     0xfffff13e  /* 割り込み制御レジスタ                    */
116  #define   ISPR     0xfffff166  /* インサービス・プライオリティ・レジスタ*/
117  #define   PRCMD    0xfffff170  /* コマンド・レジスタ                      */
118  #define   INTM0    0xfffff180  /* 外部割り込みモード・レジスタ 0          */
119  #define   INTM1    0xfffff182  /* 外部割り込みモード・レジスタ 1          */
120  #define   INTM2    0xfffff184  /* 外部割り込みモード・レジスタ 2          */
121  #define   INTM3    0xfffff186  /* 外部割り込みモード・レジスタ 3          */
122  #define   INTM4    0xfffff188  /* 外部割り込みモード・レジスタ 4          */
123  #define   TOVS     0xfffff230  /* タイマオーバフロー・ステータス・レジスタ */
124  #define   TUM11    0xfffff240  /* タイマ・ユニット・モード・レジスタ 11    */
125  #define   TMC11    0xfffff242  /* タイマ・コントロール・レジスタ 11        */
126  #define   TOC11    0xfffff244  /* タイマ出力コントロール・レジスタ 11      */
127  #define   TM11     0xfffff250  /* タイマ 11                               */
128  #define   CC110    0xfffff252  /* キャプチャ/コンペア・レジスタ 110        */
129  #define   CC111    0xfffff254  /* キャプチャ/コンペア・レジスタ 111        */
130  #define   CC112    0xfffff256  /* キャプチャ/コンペア・レジスタ 112        */
131  #define   CC113    0xfffff258  /* キャプチャ/コンペア・レジスタ 113        */
132  #define   TUM12    0xfffff260  /* タイマ・ユニット・モード・レジスタ 12    */
133  #define   TMC12    0xfffff262  /* タイマ・コントロール・レジスタ 12        */
134  #define   TOC12    0xfffff264  /* タイマ出力コントロール・レジスタ 12      */
135  #define   TM12     0xfffff270  /* タイマ 12                               */
136  #define   CC120    0xfffff272  /* キャプチャ/コンペア・レジスタ 120        */
137  #define   CC121    0xfffff274  /* キャプチャ/コンペア・レジスタ 121        */
138  #define   CC122    0xfffff276  /* キャプチャ/コンペア・レジスタ 122        */
139  #define   CC123    0xfffff278  /* キャプチャ/コンペア・レジスタ 123        */
140  #define   TUM13    0xfffff280  /* タイマ・ユニット・モード・レジスタ 13    */
141  #define   TMC13    0xfffff282  /* タイマ・コントロール・レジスタ 13        */
```

```
142  #define   TOC13   0xfffff284  /* タイマ出力コントロール・レジスタ 13 */
143  #define   TM13    0xfffff290  /* タイマ 13                          */
144  #define   CC130   0xfffff292  /* キャプチャ/コンペア・レジスタ 130   */
145  #define   CC131   0xfffff294  /* キャプチャ/コンペア・レジスタ 131   */
146  #define   CC132   0xfffff296  /* キャプチャ/コンペア・レジスタ 132   */
147  #define   CC133   0xfffff298  /* キャプチャ/コンペア・レジスタ 133   */
148  #define   TUM14   0xfffff2a0  /* タイマ・ユニット・モード・レジスタ 14 */
149  #define   TMC14   0xfffff2a2  /* タイマ・コントロール・レジスタ 14    */
150  #define   TOC14   0xfffff2a4  /* タイマ出力コントロール・レジスタ 14  */
151  #define   TM14    0xfffff2b0  /* タイマ 14                          */
152  #define   CC140   0xfffff2b2  /* キャプチャ/コンペア・レジスタ 140   */
153  #define   CC141   0xfffff2b4  /* キャプチャ/コンペア・レジスタ 141   */
154  #define   CC142   0xfffff2b6  /* キャプチャ/コンペア・レジスタ 142   */
155  #define   CC143   0xfffff2b8  /* キャプチャ/コンペア・レジスタ 143   */
156  #define   TMC4    0xfffff342  /* タイマ・コントロール・レジスタ 4     */
157  #define   TM4     0xfffff350  /* タイマ 4                           */
158  #define   CM4     0xfffff352  /* コンペア・レジスタ 4                */
159  #define   PWMC    0xfffff360  /* PWM コントロール・レジスタ          */
160  #define   PWPR    0xfffff362  /* PWM プリスケーラ・レジスタ          */
161  #define   PWM0    0xfffff364  /* PWM バッファ・レジスタ 0  (12 ビット)    */
162  #define   PWM0L   0xfffff366  /* PWM バッファ・レジスタ 0L (下位 8 ビット) */
163  #define   PWM1    0xfffff368  /* PWM バッファ・レジスタ 1  (12 ビット)    */
164  #define   PWM1L   0xfffff36a  /* PWM バッファ・レジスタ 1L (下位 8 ビット) */
165  #define   ADM0    0xfffff380  /* A/D コンバータ・モード・レジスタ 0    */
166  #define   ADM1    0xfffff382  /* A/D コンバータ・モード・レジスタ 1    */
167  #define   ADCR0   0xfffff390  /* A/D 変換結果レジスタ 0              */
168  #define   ADCR0H  0xfffff392  /* A/D 変換結果レジスタ 0H             */
169  #define   ADCR1   0xfffff394  /* A/D 変換結果レジスタ 1              */
170  #define   ADCR1H  0xfffff396  /* A/D 変換結果レジスタ 1H             */
171  #define   ADCR2   0xfffff398  /* A/D 変換結果レジスタ 2              */
172  #define   ADCR2H  0xfffff39a  /* A/D 変換結果レジスタ 2H             */
173  #define   ADCR3   0xfffff39c  /* A/D 変換結果レジスタ 3              */
174  #define   ADCR3H  0xfffff39e  /* A/D 変換結果レジスタ 3H             */
175  #define   ADCR4   0xfffff3a0  /* A/D 変換結果レジスタ 4              */
176  #define   ADCR4H  0xfffff3a2  /* A/D 変換結果レジスタ 4H             */
177  #define   ADCR5   0xfffff3a4  /* A/D 変換結果レジスタ 5              */
178  #define   ADCR5H  0xfffff3a6  /* A/D 変換結果レジスタ 5H             */
179  #define   ADCR6   0xfffff3a8  /* A/D 変換結果レジスタ 6              */
180  #define   ADCR6H  0xfffff3aa  /* A/D 変換結果レジスタ 6H             */
181  #define   ADCR7   0xfffff3ac  /* A/D 変換結果レジスタ 7              */
182  #define   ADCR7H  0xfffff3ae  /* A/D 変換結果レジスタ 7H             */
183  #define   DACS0   0xfffff3c0  /* D/A コンバータ変換値設定レジスタ 0   */
184  #define   DACS1   0xfffff3c2  /* D/A コンバータ変換値設定レジスタ 1   */
185  #define   DAM     0xfffff3d0  /* D/A コンバータ・モード・レジスタ     */
```

【 sample.h 】

```
1   /*****************************************************************************
2       Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   *****************************************************************************
4       Module      :   sample.h
5       Target      :   V853
6                   :   ヘッダファイル
7   *****************************************************************************/
8   #include  "stdrx850.h"
9   #include  "sit.h"
10  #include  "port.h"
```

【 リンク・マップ・ファイル 】（GHS 版 : lcfile.lnk）

※ これは一例です。各自配置したいアドレスを指定します。

```
1   -sec {
2       .sit     0x000280    :
3       .text                :
4       .syscall        :
5       .secinfo        :
6       .fixaddr        :
7       .fixtype        :
8
9               .rodata    ROM(.data)   :
10              .rosdata   ROM(.sdata):
11              .rozdata   ROM(.zdata):
12
13      .pool1        0x800000   :
14
15      .pool0          0xffffe000    :
16          .data                  :
17          .SDATA1               :
18          .sdata                 :
19          .SDATA2               :
20          .SBSS1                :
21          .sbss                  :
22          .SBSS2                :
23          .bss                   :
24      .zdata                     :
25          .zbss                  :
26  }
```

【 リンク・マップ・ファイル 】（NEC 版 : lcfile.lnk）

※ これは一例です。各自配置したいアドレスを指定します。

```
1   SIT    : !LOAD ?RX
2
3       .sit = $PROGBITS  ?AX .sit;
4   };
5
6   TEXT : !LOAD ?RX {
7       .text    = $PROGBITS  ?AX .text;
8   };
9
10  CONST   : !LOAD ?R {
11      .const = $PROGBITS  ?A .const;
12  };
13
14  EDATA   : !LOAD ?RW V0x800000{
15      .pool1 = $NOBITS  ?AW .pool1;
16  };
17
18  SEDATA : !LOAD ?RW V0xffd000{
19      .sedata     = $PROGBITS  ?AW .sedata;
20      .sebss = $NOBITS  ?AW .sebss;
21  };
22
23  SIDATA  : !LOAD ?RW V0xffe000{
24      .tidata.byte    = $PROGBITS  ?AW .tidata.byte;
25      .tidata.word  = $PROGBITS  ?AW .tidata.word;
26      .tidata = $PROGBITS  ?AW .tidata;
27      .sidata = $PROGBITS  ?AW .sidata;
28  };
29
30  IDATA    : !LOAD ?RW {
31      .pool0 = $NOBITS  ?AW .pool0;
32      .data   = $PROGBITS  ?AW .data;
33      .sdata  = $PROGBITS  ?AWG .sdata;
34      .sbss   = $NOBITS  ?AWG .sbss;
35      .bss    = $NOBITS  ?AW .bss;
36  };
37
38  __tp_TEXT @ %TP_SYMBOL {TEXT};
39  __gp_DATA @ %GP_SYMBOL {IDATA};
40  __ep_DATA @ %EP_SYMBOL ;
```

【 make ファイル 】（GHS 版）

※GHS のバージョンが 1.8.8 以上の時は、バッチファイルを作成するか、ビルダを使用してロードモジュールを作成します。

```
1   #/*****************************************************************************
2   #   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   #*****************************************************************************
4   #   Module          :   makefile (For GHS Tools)
5   #   Target          :   V853
6   #*****************************************************************************/
7
8   # ***  レジスタモード
9   REGS = 32
10
11  # ***    Tools Dirs     各自環境に合わせて書き換え
12  TOOLS   = d:/ghs
13
14  # ***    Nucleus Dirs
15  NUCLEUS_TOP =    d:/ghstools
16  RXLIB_DIR = $(NUCLEUS_TOP)/lib850/r$(REGS)
17  RXLIB_INC = $(NUCLEUS_TOP)/inc850
18
19  # *** Nucleus Library Name
20  RXLIB_NAME = rx
21
22  # ***    Sample Dirs
23  SMPL_DIR  = ..
24  SRC_DIR     = $(SMPL_DIR)/src
25
26  # *** Library Dirs
27  GHLIBDIR = $(TOOLS)/850e
28
29  # *** Tools
30  CC= $(TOOLS)/cc850e -elf
31  AS= $(TOOLS)/as850 -elf
32  LD= $(TOOLS)/lx
33  ROMP     = $(TOOLS)/grom
34  HX     = $(TOOLS)/gsrec
35  CF = $(NUCLEUS_TOP)/bin/cf850
36
37  # *** Link Directive file
38  LC_FILE = $(SRC_DIR)/linkmap/lcfile.lnk
39
40  #    *** Options
41  REGS_OPT = -D__850_$(REGS)__ # -cpu=850_$(REGS)
42
43  #    *** Options for cc850
44  CFLAGS    = -c $(REGS_OPT)       ¥
45          -pic -pid -ANSI       ¥
46          -I$(SMPL_DIR)/include         ¥
47          -I$(NUCLEUS_TOP)/inc850   ¥
```

```
48          -I$(TOOLS)/ansi     ¥
49          -G                  ¥
50          -I.
51
52      #   *** Options for as850
53      ASFLAGS   = -w
54
55      #   *** Options for cf850
56      CFFLAGS   =
57
58      #   *** Options for lx
59      LDFLAGS = @$(LC_FILE) -sda     ¥
60              -map -L $(RXLIB_DIR) -L $(GHLIBDIR) -tda
61      LDFLAGS_2 = -l$(RXLIB_NAME) -lansi -lind
62
63      #   *** Options for coffrom
64      ROMPFLAGS = -base .text         ¥
65          -section .tdata         ¥
66          -section .data          ¥
67          -section .SDATA1        ¥
68          -section .sdata         ¥
69          -section .SDATA2        ¥
70          -section .zdata
71
72      #   *** Options for coff2sr
73      HXFLAGS   = -noS5
74
75      all : sample.hex
76
77      sample.hex : sample.rom
78          $(HX) $(HXFLAGS) -o sample.hex sample.rom
79
80      sample.rom : sample.out
81          $(ROMP) $(ROMPFLAGS) sample.out sample.rom
82
83      sample.out :  start.o meminit.o ¥
84              sit.o ¥
85              port.o ¥
86              init.o initfunc.o ¥
87              timectrl.o timecnt.o timeset.o ¥
88              tmrctrl.o tmrcnt.o tmrset.o ¥
89              stpwctrl.o stpwcnt.o stpwact.o ¥
90              slotctrl.o slotrot.o ¥
91              displed.o ledsend.o ledflash.o ¥
92              stepper.o ¥
93              initlcd.o lcdctrl.o lcdset.o lcdchar1.o lcdchar2.o ¥
94              lcddisp.o autodisp.o lcdsend.o lcdfunc.o ¥
95              lcdtime.o lcdtimer.o lcdstpw.o    lcdslot.o lcdcond.o ¥
96              buzzer.o ¥
97              adconv.o ¥
98              memoctrl.o memoset.o memoinit.o codeman.o sndcheck.o ¥
99              intad.o intctrl.o ¥
100             inttakt1.o inttakt2.o inttakt3.o inttakt4.o inttakt5.o ¥
101             inttakt6.o inttakt7.o inttakt8.o inttouch.o ¥
```

```
102             $(LC_FILE)
103         $(LD) $(LDFLAGS) -o sample.out ¥
104             start.o ¥
105             sit.o ¥
106             port.o ¥
107             init.o initfunc.o ¥
108             timectrl.o timecnt.o timeset.o ¥
109             tmrctrl.o tmrcnt.o tmrset.o ¥
110             stpwctrl.o stpwcnt.o stpwact.o ¥
111             slotctrl.o slotrot.o ¥
112             displed.o ledsend.o ledflash.o ¥
113             stepper.o ¥
114             initlcd.o lcdctrl.o lcdset.o lcdchar1.o lcdchar2.o ¥
115             lcddisp.o autodisp.o lcdsend.o lcdfunc.o ¥
116             lcdtime.o lcdtimer.o lcdstpw.o    lcdslot.o lcdcond.o ¥
117             buzzer.o ¥
118             adconv.o ¥
119             memoctrl.o memoset.o memoinit.o codeman.o sndcheck.o ¥
120             intad.o intctrl.o ¥
121             inttakt1.o inttakt2.o inttakt3.o inttakt4.o inttakt5.o ¥
122             inttakt6.o inttakt7.o inttakt8.o inttouch.o ¥
123             $(LDFLAGS_2)
124
125     # /*** スタートアップ ルーチン ***/
126     start.o : $(SRC_DIR)/start/start.850
127         $(CC) $(CFLAGS) -o start.o $(SRC_DIR)/start/start.850
128
129     # /*** メモリ初期化 ***/
130     meminit.o     : $(SRC_DIR)/start/meminit.c ./sit.h ¥
131                 $(SMPL_DIR)/include/common.h ./makefile
132         $(CC) -c $(CFLAGS) -o meminit.o $(SRC_DIR)/start/meminit.c
133
134     # /*** SIT ***/
135     sit.o    : sit.s
136         $(AS) $(ASFLAGS) sit.s -o sit.o
137
138     sit.s    : $(SRC_DIR)/sit/sit.cf ./makefile
139         $(CF) $(CFFLAGS) -i ./sit.h -o ./sit.s $(SRC_DIR)/sit/sit.cf
140
141     # /*** ポート関数 ***/
142
143     port.o : $(SRC_DIR)/port/port.850 ./makefile
144         $(CC) $(CFLAGS) -o port.o $(SRC_DIR)/port/port.850
145
146     # /*** 初期化 ***/
147
148     init.o  : $(SRC_DIR)/init/init.c ./sit.h ¥
149             $(SMPL_DIR)/include/common.h ./makefile
150         $(CC) -c $(CFLAGS) -o init.o $(SRC_DIR)/init/init.c
151
152     initfunc.o : $(SRC_DIR)/init/initfunc.c ./sit.h ¥
153                 $(SMPL_DIR)/include/common.h ./makefile
154         $(CC) -c $(CFLAGS) -o initfunc.o $(SRC_DIR)/init/initfunc.c
```

```
155
156  # /*** 現在時刻 ***/
157
158  timecnt.o : $(SRC_DIR)/time/timecnt.c ./sit.h ¥
159              $(SMPL_DIR)/include/common.h ./makefile
160      $(CC) -c $(CFLAGS) -o timecnt.o $(SRC_DIR)/time/timecnt.c
161
162  timectrl.o : $(SRC_DIR)/time/timectrl.c ./sit.h ¥
163              $(SMPL_DIR)/include/common.h ./makefile
164      $(CC) -c $(CFLAGS) -o timectrl.o $(SRC_DIR)/time/timectrl.c
165
166  timeset.o  : $(SRC_DIR)/time/timeset.c ./sit.h ¥
167              $(SMPL_DIR)/include/common.h ./makefile
168      $(CC) -c $(CFLAGS) -o timeset.o $(SRC_DIR)/time/timeset.c
169
170  # /*** タイマー ***/
171
172  tmrctrl.o  : $(SRC_DIR)/timer/tmrctrl.c ./sit.h ¥
173              $(SMPL_DIR)/include/common.h ./makefile
174      $(CC) -c $(CFLAGS) -o tmrctrl.o $(SRC_DIR)/timer/tmrctrl.c
175
176  tmrcnt.o   : $(SRC_DIR)/timer/tmrcnt.c ./sit.h ¥
177              $(SMPL_DIR)/include/common.h ./makefile
178      $(CC) -c $(CFLAGS) -o tmrcnt.o $(SRC_DIR)/timer/tmrcnt.c
179
180  tmrset.o   : $(SRC_DIR)/timer/tmrset.c ./sit.h ¥
181              $(SMPL_DIR)/include/common.h ./makefile
182      $(CC) -c $(CFLAGS) -o tmrset.o $(SRC_DIR)/timer/tmrset.c
183
184  # /*** ストップウォッチ ***/
185
186  stpwact.o : $(SRC_DIR)/stwatch/stpwact.c ./sit.h ¥
187              $(SMPL_DIR)/include/common.h ./makefile
188      $(CC) -c $(CFLAGS) -o stpwact.o $(SRC_DIR)/stwatch/stpwact.c
189
190  stpwcnt.o : $(SRC_DIR)/stwatch/stpwcnt.c ./sit.h ¥
191              $(SMPL_DIR)/include/common.h ./makefile
192      $(CC) -c $(CFLAGS) -o stpwcnt.o $(SRC_DIR)/stwatch/stpwcnt.c
193
194  stpwctrl.o: $(SRC_DIR)/stwatch/stpwctrl.c ./sit.h ¥
195              $(SMPL_DIR)/include/common.h ./makefile
196      $(CC) -c $(CFLAGS) -o stpwctrl.o $(SRC_DIR)/stwatch/stpwctrl.c
197
198  # /*** スロットマシン ***/
199
200  slotctrl.o  : $(SRC_DIR)/slot/slotctrl.c ./sit.h ¥
201              $(SMPL_DIR)/include/common.h ./makefile
202      $(CC) -c $(CFLAGS) -o slotctrl.o $(SRC_DIR)/slot/slotctrl.c
203
204  slotrot.o   : $(SRC_DIR)/slot/slotrot.c ./sit.h ¥
205              $(SMPL_DIR)/include/common.h ./makefile
206      $(CC) -c $(CFLAGS) -o slotrot.o $(SRC_DIR)/slot/slotrot.c
207
```

```
208  # /*** LED ***/
209
210  displed.o : $(SRC_DIR)/led/displed.c ./sit.h ¥
211              $(SMPL_DIR)/include/common.h ./makefile
212      $(CC) -c $(CFLAGS) -o displed.o $(SRC_DIR)/led/displed.c
213
214  ledflash.o: $(SRC_DIR)/led/ledflash.c ./sit.h ¥
215              $(SMPL_DIR)/include/common.h ./makefile
216      $(CC) -c $(CFLAGS) -o ledflash.o $(SRC_DIR)/led/ledflash.c
217
218  ledsend.o : $(SRC_DIR)/led/ledsend.c ./sit.h ¥
219              $(SMPL_DIR)/include/common.h ./makefile
220      $(CC) -c $(CFLAGS) -o ledsend.o $(SRC_DIR)/led/ledsend.c
221
222  # /***  ステッピングモーター  ***/
223
224  stepper.o  : $(SRC_DIR)/stepper/stepper.c ./sit.h ¥
225              $(SMPL_DIR)/include/common.h ./makefile
226      $(CC) -c $(CFLAGS) -o stepper.o $(SRC_DIR)/stepper/stepper.c
227
228  # /*** LCD ***/
229
230  lcdctrl.o   : $(SRC_DIR)/lcd/lcdctrl.c ./sit.h ¥
231              $(SMPL_DIR)/include/common.h ./makefile
232      $(CC) -c $(CFLAGS) -o lcdctrl.o $(SRC_DIR)/lcd/lcdctrl.c
233
234  initlcd.o   : $(SRC_DIR)/lcd/initlcd.c ./sit.h ¥
235              $(SMPL_DIR)/include/common.h ./makefile
236      $(CC) -c $(CFLAGS) -o initlcd.o $(SRC_DIR)/lcd/initlcd.c
237
238  lcdset.o    : $(SRC_DIR)/lcd/lcdset.c ./sit.h ¥
239              $(SMPL_DIR)/include/common.h ./makefile
240      $(CC) -c $(CFLAGS) -o lcdset.o $(SRC_DIR)/lcd/lcdset.c
241
242  lcdchar1.o     : $(SRC_DIR)/lcd/lcdchar1.c ./sit.h ¥
243              $(SMPL_DIR)/include/common.h ./makefile
244      $(CC) -c $(CFLAGS) -o lcdchar1.o $(SRC_DIR)/lcd/lcdchar1.c
245
246  lcdchar2.o     : $(SRC_DIR)/lcd/lcdchar2.c ./sit.h ¥
247              $(SMPL_DIR)/include/common.h ./makefile
248      $(CC) -c $(CFLAGS) -o lcdchar2.o $(SRC_DIR)/lcd/lcdchar2.c
249
250  lcddisp.o  : $(SRC_DIR)/lcd/lcddisp.c ./sit.h ¥
251              $(SMPL_DIR)/include/common.h ./makefile
252      $(CC) -c $(CFLAGS) -o lcddisp.o $(SRC_DIR)/lcd/lcddisp.c
253
254  lcdsend.o : $(SRC_DIR)/lcd/lcdsend.c ./sit.h ¥
255              $(SMPL_DIR)/include/common.h ./makefile
256      $(CC) -c $(CFLAGS) -o lcdsend.o $(SRC_DIR)/lcd/lcdsend.c
257
258  lcdslot.o  : $(SRC_DIR)/lcd/lcdslot.c ./sit.h ¥
259              $(SMPL_DIR)/include/common.h ./makefile
260      $(CC) -c $(CFLAGS) -o lcdslot.o $(SRC_DIR)/lcd/lcdslot.c
261
```

```
262  lcdstpw.o : $(SRC_DIR)/lcd/lcdstpw.c ./sit.h ¥
263              $(SMPL_DIR)/include/common.h ./makefile
264      $(CC) -c $(CFLAGS) -o lcdstpw.o $(SRC_DIR)/lcd/lcdstpw.c
265
266  lcdtime.o : $(SRC_DIR)/lcd/lcdtime.c ./sit.h ¥
267              $(SMPL_DIR)/include/common.h ./makefile
268      $(CC) -c $(CFLAGS) -o lcdtime.o $(SRC_DIR)/lcd/lcdtime.c
269
270  lcdtimer.o: $(SRC_DIR)/lcd/lcdtimer.c ./sit.h ¥
271              $(SMPL_DIR)/include/common.h ./makefile
272      $(CC) -c $(CFLAGS) -o lcdtimer.o $(SRC_DIR)/lcd/lcdtimer.c
273
274  lcdfunc.o : $(SRC_DIR)/lcd/lcdfunc.c ./sit.h ¥
275              $(SMPL_DIR)/include/common.h ./makefile
276      $(CC) -c $(CFLAGS) -o lcdfunc.o $(SRC_DIR)/lcd/lcdfunc.c
277
278  autodisp.o    : $(SRC_DIR)/lcd/autodisp.c ./sit.h ¥
279              $(SMPL_DIR)/include/common.h ./makefile
280      $(CC) -c $(CFLAGS) -o autodisp.o $(SRC_DIR)/lcd/autodisp.c
281
282  lcdcond.o: $(SRC_DIR)/lcd/lcdcond.c ./sit.h ¥
283              $(SMPL_DIR)/include/common.h ./makefile
284      $(CC) -c $(CFLAGS) -o lcdcond.o $(SRC_DIR)/lcd/lcdcond.c
285
286  # /*** A/D コンバート ***/
287
288  adconv.o  : $(SRC_DIR)/senser/adconv.c ./sit.h ¥
289              $(SMPL_DIR)/include/common.h ./makefile
290      $(CC) -c $(CFLAGS) -o adconv.o $(SRC_DIR)/senser/adconv.c
291
292  # /*** メモ機能 ***/
293
294  codeman.o    : $(SRC_DIR)/memo/codeman.c ./sit.h ¥
295              $(SMPL_DIR)/include/common.h ./makefile
296      $(CC) -c $(CFLAGS) -o codeman.o $(SRC_DIR)/memo/codeman.c
297
298  memoctrl.o   : $(SRC_DIR)/memo/memoctrl.c ./sit.h ¥
299              $(SMPL_DIR)/include/common.h ./makefile
300      $(CC) -c $(CFLAGS) -o memoctrl.o $(SRC_DIR)/memo/memoctrl.c
301
302  memoset.o    : $(SRC_DIR)/memo/memoset.c ./sit.h ¥
303              $(SMPL_DIR)/include/common.h ./makefile
304      $(CC) -c $(CFLAGS) -o memoset.o $(SRC_DIR)/memo/memoset.c
305
306  memoinit.o   : $(SRC_DIR)/memo/memoinit.c ./sit.h ¥
307              $(SMPL_DIR)/include/common.h ./makefile
308      $(CC) -c $(CFLAGS) -o memoinit.o $(SRC_DIR)/memo/memoinit.c
309
310  sndcheck.o   : $(SRC_DIR)/memo/sndcheck.c ./sit.h ¥
311              $(SMPL_DIR)/include/common.h ./makefile
312      $(CC) -c $(CFLAGS) -o sndcheck.o $(SRC_DIR)/memo/sndcheck.c
313
314  # /*** ブザー ***/
315
```

```
316  buzzer.o   : $(SRC_DIR)/buzzer/buzzer.c ./sit.h ¥
317               $(SMPL_DIR)/include/common.h ./makefile
318       $(CC) -c $(CFLAGS) -o buzzer.o $(SRC_DIR)/buzzer/buzzer.c
319
320  # /*** 割り込み処理 ***/
321
322  intctrl.o    : $(SRC_DIR)/int/intctrl.c ./sit.h ¥
323               $(SMPL_DIR)/include/common.h ./makefile
324       $(CC) -c $(CFLAGS) -o intctrl.o $(SRC_DIR)/int/intctrl.c
325
326  inttakt1.o : $(SRC_DIR)/int/inttakt1.c ./sit.h ¥
327               $(SMPL_DIR)/include/common.h ./makefile
328       $(CC) -c $(CFLAGS) -o inttakt1.o $(SRC_DIR)/int/inttakt1.c
329
330  inttakt2.o : $(SRC_DIR)/int/inttakt2.c ./sit.h ¥
331               $(SMPL_DIR)/include/common.h ./makefile
332       $(CC) -c $(CFLAGS) -o inttakt2.o $(SRC_DIR)/int/inttakt2.c
333
334  inttakt3.o : $(SRC_DIR)/int/inttakt3.c ./sit.h ¥
335               $(SMPL_DIR)/include/common.h ./makefile
336       $(CC) -c $(CFLAGS) -o inttakt3.o $(SRC_DIR)/int/inttakt3.c
337
338  inttakt4.o : $(SRC_DIR)/int/inttakt4.c ./sit.h ¥
339               $(SMPL_DIR)/include/common.h ./makefile
340       $(CC) -c $(CFLAGS) -o inttakt4.o $(SRC_DIR)/int/inttakt4.c
341
342  inttakt5.o : $(SRC_DIR)/int/inttakt5.c ./sit.h ¥
343               $(SMPL_DIR)/include/common.h ./makefile
344       $(CC) -c $(CFLAGS) -o inttakt5.o $(SRC_DIR)/int/inttakt5.c
345
346  inttakt6.o : $(SRC_DIR)/int/inttakt6.c ./sit.h ¥
347               $(SMPL_DIR)/include/common.h ./makefile
348       $(CC) -c $(CFLAGS) -o inttakt6.o $(SRC_DIR)/int/inttakt6.c
349
350  inttakt7.o : $(SRC_DIR)/int/inttakt7.c ./sit.h ¥
351               $(SMPL_DIR)/include/common.h ./makefile
352       $(CC) -c $(CFLAGS) -o inttakt7.o $(SRC_DIR)/int/inttakt7.c
353
354  inttakt8.o : $(SRC_DIR)/int/inttakt8.c ./sit.h ¥
355               $(SMPL_DIR)/include/common.h ./makefile
356       $(CC) -c $(CFLAGS) -o inttakt8.o $(SRC_DIR)/int/inttakt8.c
357
358  inttouch.o: $(SRC_DIR)/int/inttouch.c ./sit.h ¥
359               $(SMPL_DIR)/include/common.h ./makefile
360       $(CC) -c $(CFLAGS) -o inttouch.o $(SRC_DIR)/int/inttouch.c
361
362  intad.o          : $(SRC_DIR)/int/intad.c ./sit.h ¥
363               $(SMPL_DIR)/include/common.h ./makefile
364       $(CC) -c $(CFLAGS) -o intad.o $(SRC_DIR)/int/intad.c
365
366  # /*** make clean ***/
367
368  clean  :
369       rm -f *.o
```

```
370         rm -f *.s
371         rm -f *.h
372         rm -f *.out
373         rm -f *.hex
374         rm -f *.map
375         rm -f *.rom
```

【 make ファイル 】（NEC 版）

```
1   #/*****************************************************************************
2   #   Realtime OS nucleus [RX850 Ver.3.1]   アプリケーション・ノート
3   #*****************************************************************************
4   #   Module          :   makefile (For NEC Tools)
5   #   Target          :   V853
6   #*****************************************************************************/
7
8   # ***  レジスタモード
9   REGS = 32
10
11  # *** CPU
12  CPU   = 3003
13
14  # ***    Tools Dirs
15  TOOLS   = b:/nectools
16
17  # ***    Nucleus Dirs
18  NUCLEUS_TOP =    b:/nectools
19  RXLIB_DIR = $(NUCLEUS_TOP)/lib850/r$(REGS)
20  RXLIB_INC = $(NUCLEUS_TOP)/inc850
21
22  # *** Nucleus Library Name
23  RXLIB_NAME = librxa.a    # for AZ850
24  # RXLIB_NAME = librx.a
25
26
27  # *** Device file Dir
28  DEVPATH    = $(TOOLS)/dev
29
30  # ***    Sample Dirs
31  SMPL_DIR  = ..
32  SRC_DIR     = $(SMPL_DIR)/src
33
34  # *** Tools
35  CC= $(TOOLS)/bin/ca850
36  AS= $(TOOLS)/bin/as850
37  LD= $(TOOLS)/bin/ld850
38  ROMP     = $(TOOLS)/bin/romp850
39  HX     = $(TOOLS)/bin/hx850
40  CF = $(NUCLEUS_TOP)/bin/cf850
41
42  # *** Link Directive file
43  LC_FILE = $(SRC_DIR)/linkmap/lcfile.lnk
44
45  #    *** Options
46  # REGS_OPT = -reg
47
48  #    *** Options for ca850
49  CFLAGS    =    -g                    ¥
50              -I$(SMPL_DIR)/include    ¥
```

```
51              -I$(NUCLEUS_TOP)/inc850  ¥
52              -I$(TOOLS)/include       ¥
53              -I.
54
55      #-devpath=$(DEVPATH)     ¥
56      #           -cpu $(CPU)          ¥
57      #           $(REGS_OPT)      ¥
58
59
60      # *** Option for AS850
61      ASFLAGS    = -F $(DEVPATH) -cpu $(CPU)
62
63      # *** Option for CF850
64      CFFLAGS = -devpath=$(DEVPATH) -cpu $(CPU) ¥
65            -i ./sit.h -o ./sit.s -az
66
67      # *** Option For LD850
68      LDFLAGS    = -D $(LC_FILE)
69
70      # *** Option for ROMP850
71      ROMPFLAGS = -F $(DEVPATH)
72
73
74      # *** Option for HX850
75      HXFLAGS    =
76
77      all : sample.out
78
79      sample.hex : sample.rom
80          $(HX) $(HXFLAGS) -o sample.hex sample.rom
81
82      sample.rom : sample.out
83          $(ROMP) $(ROMPFLAGS) sample.out sample.rom
84
85      sample.out :  start.o ¥
86              sit.o ¥
87              port.o ¥
88              init.o initfunc.o ¥
89              timectrl.o timecnt.o timeset.o ¥
90              tmrctrl.o tmrcnt.o tmrset.o ¥
91              stpwctrl.o stpwcnt.o stpwact.o ¥
92              slotctrl.o slotrot.o ¥
93              displed.o ledsend.o ledflash.o ¥
94              stepper.o ¥
95              initlcd.o lcdctrl.o lcdset.o lcdchar1.o lcdchar2.o ¥
96              lcddisp.o autodisp.o lcdsend.o lcdfunc.o ¥
97              lcdtime.o lcdtimer.o lcdstpw.o    lcdslot.o lcdcond.o ¥
98              buzzer.o ¥
99              adconv.o ¥
100             memoctrl.o memoset.o memoinit.o codeman.o sndcheck.o ¥
101             intad.o intctrl.o ¥
102             inttakt1.o inttakt2.o inttakt3.o inttakt4.o inttakt5.o ¥
103             inttakt6.o inttakt7.o inttakt8.o inttouch.o ¥
104             $(LC_FILE)
```

```
105         $(LD) $(LDFLAGS) -o sample.out ¥
106             start.o ¥
107             sit.o ¥
108             port.o ¥
109             init.o initfunc.o ¥
110             timectrl.o timecnt.o timeset.o ¥
111             tmrctrl.o tmrcnt.o tmrset.o ¥
112             stpwctrl.o stpwcnt.o stpwact.o ¥
113             slotctrl.o slotrot.o ¥
114             displed.o ledsend.o ledflash.o ¥
115             stepper.o ¥
116             initlcd.o lcdctrl.o lcdset.o lcdchar1.o lcdchar2.o ¥
117             lcddisp.o autodisp.o lcdsend.o lcdfunc.o ¥
118             lcdtime.o lcdtimer.o lcdstpw.o    lcdslot.o lcdcond.o ¥
119             buzzer.o ¥
120             adconv.o ¥
121             memoctrl.o memoset.o memoinit.o codeman.o sndcheck.o ¥
122             intad.o intctrl.o ¥
123             inttakt1.o inttakt2.o inttakt3.o inttakt4.o inttakt5.o ¥
124             inttakt6.o inttakt7.o inttakt8.o inttouch.o ¥
125             $(LDFLAGS_2)
126
127     # /***  スタートアップ ルーチン  ***/
128     start.o : $(SRC_DIR)/start/start.850
129         $(CC) $(CFLAGS) -o start.o $(SRC_DIR)/start/start.850
130
131     # /*** SIT ***/
132     sit.o    : sit.s
133         $(AS) $(ASFLAGS) sit.s -o sit.o
134
135     sit.s    : $(SRC_DIR)/sit/sit.cf ./makefile
136         $(CF) $(CFFLAGS) -i ./sit.h -o ./sit.s $(SRC_DIR)/sit/sit.cf
137
138     # /***  ポート関数  ***/
139
140     port.o : $(SRC_DIR)/port/port.850 ./makefile
141         $(CC) $(CFLAGS) -o port.o $(SRC_DIR)/port/port.850
142
143     # /***  初期化  ***/
144
145     init.o  : $(SRC_DIR)/init/init.c ./sit.h ¥
146             $(SMPL_DIR)/include/common.h ./makefile
147         $(CC) -c $(CFLAGS) -o init.o $(SRC_DIR)/init/init.c
148
149     initfunc.o : $(SRC_DIR)/init/initfunc.c ./sit.h ¥
150                 $(SMPL_DIR)/include/common.h ./makefile
151         $(CC) -c $(CFLAGS) -o initfunc.o $(SRC_DIR)/init/initfunc.c
152
153     # /***  現在時刻  ***/
154
155     timecnt.o : $(SRC_DIR)/time/timecnt.c ./sit.h ¥
156                 $(SMPL_DIR)/include/common.h ./makefile
157         $(CC) -c $(CFLAGS) -o timecnt.o $(SRC_DIR)/time/timecnt.c
```

```
158
159  timectrl.o : $(SRC_DIR)/time/timectrl.c ./sit.h ¥
160               $(SMPL_DIR)/include/common.h ./makefile
161       $(CC) -c $(CFLAGS) -o timectrl.o $(SRC_DIR)/time/timectrl.c
162
163  timeset.o  : $(SRC_DIR)/time/timeset.c ./sit.h ¥
164               $(SMPL_DIR)/include/common.h ./makefile
165       $(CC) -c $(CFLAGS) -o timeset.o $(SRC_DIR)/time/timeset.c
166
167  # /*** タイマー ***/
168
169  tmrctrl.o  : $(SRC_DIR)/timer/tmrctrl.c ./sit.h ¥
170               $(SMPL_DIR)/include/common.h ./makefile
171       $(CC) -c $(CFLAGS) -o tmrctrl.o $(SRC_DIR)/timer/tmrctrl.c
172
173  tmrcnt.o   : $(SRC_DIR)/timer/tmrcnt.c ./sit.h ¥
174               $(SMPL_DIR)/include/common.h ./makefile
175       $(CC) -c $(CFLAGS) -o tmrcnt.o $(SRC_DIR)/timer/tmrcnt.c
176
177  tmrset.o   : $(SRC_DIR)/timer/tmrset.c ./sit.h ¥
178               $(SMPL_DIR)/include/common.h ./makefile
179       $(CC) -c $(CFLAGS) -o tmrset.o $(SRC_DIR)/timer/tmrset.c
180
181  # /*** ストップウォッチ ***/
182
183  stpwact.o : $(SRC_DIR)/stwatch/stpwact.c ./sit.h ¥
184               $(SMPL_DIR)/include/common.h ./makefile
185       $(CC) -c $(CFLAGS) -o stpwact.o $(SRC_DIR)/stwatch/stpwact.c
186
187  stpwcnt.o : $(SRC_DIR)/stwatch/stpwcnt.c ./sit.h ¥
188               $(SMPL_DIR)/include/common.h ./makefile
189       $(CC) -c $(CFLAGS) -o stpwcnt.o $(SRC_DIR)/stwatch/stpwcnt.c
190
191  stpwctrl.o: $(SRC_DIR)/stwatch/stpwctrl.c ./sit.h ¥
192               $(SMPL_DIR)/include/common.h ./makefile
193       $(CC) -c $(CFLAGS) -o stpwctrl.o $(SRC_DIR)/stwatch/stpwctrl.c
194
195  # /*** スロットマシン ***/
196
197  slotctrl.o  : $(SRC_DIR)/slot/slotctrl.c ./sit.h ¥
198               $(SMPL_DIR)/include/common.h ./makefile
199       $(CC) -c $(CFLAGS) -o slotctrl.o $(SRC_DIR)/slot/slotctrl.c
200
201  slotrot.o   : $(SRC_DIR)/slot/slotrot.c ./sit.h ¥
202               $(SMPL_DIR)/include/common.h ./makefile
203       $(CC) -c $(CFLAGS) -o slotrot.o $(SRC_DIR)/slot/slotrot.c
204
205  # /*** LED ***/
206
207  displed.o  : $(SRC_DIR)/led/displed.c ./sit.h ¥
208               $(SMPL_DIR)/include/common.h ./makefile
209       $(CC) -c $(CFLAGS) -o displed.o $(SRC_DIR)/led/displed.c
210
211  ledflash.o : $(SRC_DIR)/led/ledflash.c ./sit.h ¥
```

```
212             $(SMPL_DIR)/include/common.h ./makefile
213         $(CC) -c $(CFLAGS) -o ledflash.o $(SRC_DIR)/led/ledflash.c
214
215     ledsend.o : $(SRC_DIR)/led/ledsend.c ./sit.h ¥
216             $(SMPL_DIR)/include/common.h ./makefile
217         $(CC) -c $(CFLAGS) -o ledsend.o $(SRC_DIR)/led/ledsend.c
218
219     # /*** ステッピングモーター ***/
220
221     stepper.o  : $(SRC_DIR)/stepper/stepper.c ./sit.h ¥
222             $(SMPL_DIR)/include/common.h ./makefile
223         $(CC) -c $(CFLAGS) -o stepper.o $(SRC_DIR)/stepper/stepper.c
224
225     # /*** LCD ***/
226
227     lcdctrl.o   : $(SRC_DIR)/lcd/lcdctrl.c ./sit.h ¥
228             $(SMPL_DIR)/include/common.h ./makefile
229         $(CC) -c $(CFLAGS) -o lcdctrl.o $(SRC_DIR)/lcd/lcdctrl.c
230
231     initlcd.o   : $(SRC_DIR)/lcd/initlcd.c ./sit.h ¥
232             $(SMPL_DIR)/include/common.h ./makefile
233         $(CC) -c $(CFLAGS) -o initlcd.o $(SRC_DIR)/lcd/initlcd.c
234
235     lcdset.o    : $(SRC_DIR)/lcd/lcdset.c ./sit.h ¥
236             $(SMPL_DIR)/include/common.h ./makefile
237         $(CC) -c $(CFLAGS) -o lcdset.o $(SRC_DIR)/lcd/lcdset.c
238
239     lcdchar1.o     : $(SRC_DIR)/lcd/lcdchar1.c ./sit.h ¥
240             $(SMPL_DIR)/include/common.h ./makefile
241         $(CC) -c $(CFLAGS) -o lcdchar1.o $(SRC_DIR)/lcd/lcdchar1.c
242
243     lcdchar2.o     : $(SRC_DIR)/lcd/lcdchar2.c ./sit.h ¥
244             $(SMPL_DIR)/include/common.h ./makefile
245         $(CC) -c $(CFLAGS) -o lcdchar2.o $(SRC_DIR)/lcd/lcdchar2.c
246
247     lcddisp.o  : $(SRC_DIR)/lcd/lcddisp.c ./sit.h ¥
248             $(SMPL_DIR)/include/common.h ./makefile
249         $(CC) -c $(CFLAGS) -o lcddisp.o $(SRC_DIR)/lcd/lcddisp.c
250
251     lcdsend.o : $(SRC_DIR)/lcd/lcdsend.c ./sit.h ¥
252             $(SMPL_DIR)/include/common.h ./makefile
253         $(CC) -c $(CFLAGS) -o lcdsend.o $(SRC_DIR)/lcd/lcdsend.c
254
255     lcdslot.o  : $(SRC_DIR)/lcd/lcdslot.c ./sit.h ¥
256             $(SMPL_DIR)/include/common.h ./makefile
257         $(CC) -c $(CFLAGS) -o lcdslot.o $(SRC_DIR)/lcd/lcdslot.c
258
259     lcdstpw.o : $(SRC_DIR)/lcd/lcdstpw.c ./sit.h ¥
260             $(SMPL_DIR)/include/common.h ./makefile
261         $(CC) -c $(CFLAGS) -o lcdstpw.o $(SRC_DIR)/lcd/lcdstpw.c
262
263     lcdtime.o : $(SRC_DIR)/lcd/lcdtime.c ./sit.h ¥
264             $(SMPL_DIR)/include/common.h ./makefile
265         $(CC) -c $(CFLAGS) -o lcdtime.o $(SRC_DIR)/lcd/lcdtime.c
```

```
266
267  lcdtimer.o: $(SRC_DIR)/lcd/lcdtimer.c ./sit.h ¥
268              $(SMPL_DIR)/include/common.h ./makefile
269      $(CC) -c $(CFLAGS) -o lcdtimer.o $(SRC_DIR)/lcd/lcdtimer.c
270
271  lcdfunc.o : $(SRC_DIR)/lcd/lcdfunc.c ./sit.h ¥
272              $(SMPL_DIR)/include/common.h ./makefile
273      $(CC) -c $(CFLAGS) -o lcdfunc.o $(SRC_DIR)/lcd/lcdfunc.c
274
275  autodisp.o    : $(SRC_DIR)/lcd/autodisp.c ./sit.h ¥
276              $(SMPL_DIR)/include/common.h ./makefile
277      $(CC) -c $(CFLAGS) -o autodisp.o $(SRC_DIR)/lcd/autodisp.c
278
279  lcdcond.o: $(SRC_DIR)/lcd/lcdcond.c ./sit.h ¥
280              $(SMPL_DIR)/include/common.h ./makefile
281      $(CC) -c $(CFLAGS) -o lcdcond.o $(SRC_DIR)/lcd/lcdcond.c
282
283  # /*** A/D コンバート ***/
284
285  adconv.o  : $(SRC_DIR)/senser/adconv.c ./sit.h ¥
286              $(SMPL_DIR)/include/common.h ./makefile
287      $(CC) -c $(CFLAGS) -o adconv.o $(SRC_DIR)/senser/adconv.c
288
289  # /*** メモ機能 ***/
290
291  codeman.o    : $(SRC_DIR)/memo/codeman.c ./sit.h ¥
292              $(SMPL_DIR)/include/common.h ./makefile
293      $(CC) -c $(CFLAGS) -o codeman.o $(SRC_DIR)/memo/codeman.c
294
295  memoctrl.o   : $(SRC_DIR)/memo/memoctrl.c ./sit.h ¥
296              $(SMPL_DIR)/include/common.h ./makefile
297      $(CC) -c $(CFLAGS) -o memoctrl.o $(SRC_DIR)/memo/memoctrl.c
298
299  memoset.o    : $(SRC_DIR)/memo/memoset.c ./sit.h ¥
300              $(SMPL_DIR)/include/common.h ./makefile
301      $(CC) -c $(CFLAGS) -o memoset.o $(SRC_DIR)/memo/memoset.c
302
303  memoinit.o   : $(SRC_DIR)/memo/memoinit.c ./sit.h ¥
304              $(SMPL_DIR)/include/common.h ./makefile
305      $(CC) -c $(CFLAGS) -o memoinit.o $(SRC_DIR)/memo/memoinit.c
306
307  sndcheck.o   : $(SRC_DIR)/memo/sndcheck.c ./sit.h ¥
308              $(SMPL_DIR)/include/common.h ./makefile
309      $(CC) -c $(CFLAGS) -o sndcheck.o $(SRC_DIR)/memo/sndcheck.c
310
311  # /*** ブザー ***/
312
313  buzzer.o  : $(SRC_DIR)/buzzer/buzzer.c ./sit.h ¥
314              $(SMPL_DIR)/include/common.h ./makefile
315      $(CC) -c $(CFLAGS) -o buzzer.o $(SRC_DIR)/buzzer/buzzer.c
316
317  # /*** 割り込み処理 ***/
318
```

```
319  intctrl.o    : $(SRC_DIR)/int/intctrl.c ./sit.h ¥
320                 $(SMPL_DIR)/include/common.h ./makefile
321      $(CC) -c $(CFLAGS) -o intctrl.o $(SRC_DIR)/int/intctrl.c
322
323  inttakt1.o : $(SRC_DIR)/int/inttakt1.c ./sit.h ¥
324                 $(SMPL_DIR)/include/common.h ./makefile
325      $(CC) -c $(CFLAGS) -o inttakt1.o $(SRC_DIR)/int/inttakt1.c
326
327  inttakt2.o : $(SRC_DIR)/int/inttakt2.c ./sit.h ¥
328                 $(SMPL_DIR)/include/common.h ./makefile
329      $(CC) -c $(CFLAGS) -o inttakt2.o $(SRC_DIR)/int/inttakt2.c
330
331  inttakt3.o : $(SRC_DIR)/int/inttakt3.c ./sit.h ¥
332                 $(SMPL_DIR)/include/common.h ./makefile
333      $(CC) -c $(CFLAGS) -o inttakt3.o $(SRC_DIR)/int/inttakt3.c
334
335  inttakt4.o : $(SRC_DIR)/int/inttakt4.c ./sit.h ¥
336                 $(SMPL_DIR)/include/common.h ./makefile
337      $(CC) -c $(CFLAGS) -o inttakt4.o $(SRC_DIR)/int/inttakt4.c
338
339  inttakt5.o : $(SRC_DIR)/int/inttakt5.c ./sit.h ¥
340                 $(SMPL_DIR)/include/common.h ./makefile
341      $(CC) -c $(CFLAGS) -o inttakt5.o $(SRC_DIR)/int/inttakt5.c
342
343  inttakt6.o : $(SRC_DIR)/int/inttakt6.c ./sit.h ¥
344                 $(SMPL_DIR)/include/common.h ./makefile
345      $(CC) -c $(CFLAGS) -o inttakt6.o $(SRC_DIR)/int/inttakt6.c
346
347  inttakt7.o : $(SRC_DIR)/int/inttakt7.c ./sit.h ¥
348                 $(SMPL_DIR)/include/common.h ./makefile
349      $(CC) -c $(CFLAGS) -o inttakt7.o $(SRC_DIR)/int/inttakt7.c
350
351  inttakt8.o : $(SRC_DIR)/int/inttakt8.c ./sit.h ¥
352                 $(SMPL_DIR)/include/common.h ./makefile
353      $(CC) -c $(CFLAGS) -o inttakt8.o $(SRC_DIR)/int/inttakt8.c
354
355  inttouch.o: $(SRC_DIR)/int/inttouch.c ./sit.h ¥
356                 $(SMPL_DIR)/include/common.h ./makefile
357      $(CC) -c $(CFLAGS) -o inttouch.o $(SRC_DIR)/int/inttouch.c
358
359  intad.o          : $(SRC_DIR)/int/intad.c ./sit.h ¥
360                 $(SMPL_DIR)/include/common.h ./makefile
361      $(CC) -c $(CFLAGS) -o intad.o $(SRC_DIR)/int/intad.c
362
363  # /*** make clean ***/
364
365  clean  :
366      del *.o
367      del *.s
368      del *.h
369      del *.out
370      del *.hex
371      del *.map
372      del *.rom
```

第7章　　回路図

今回使用したハードウェアの回路図です。ここに掲載した回路および回路定数は、例示的に示したものであり、量産設計を対象とするものではありません。

7.1　7 セグメント LED・LED

図 7.1-1 は 7 セグメント LED の電極配置図です。

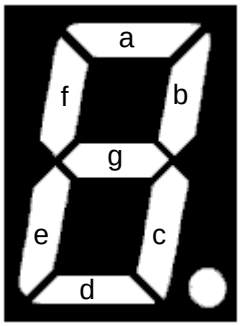

図 7.1-1　7 セグメント LED の電極配置図

今回使用した 7 セグメント LED はアノード・コモン型です。これはポートで選択されたセグメントに、スイッチング回路からの電流が流れて点灯する、いわゆる吸い込み電流による駆動方式です。図 7.1-2 は、7 セグメント LED のカソード側の回路図です。

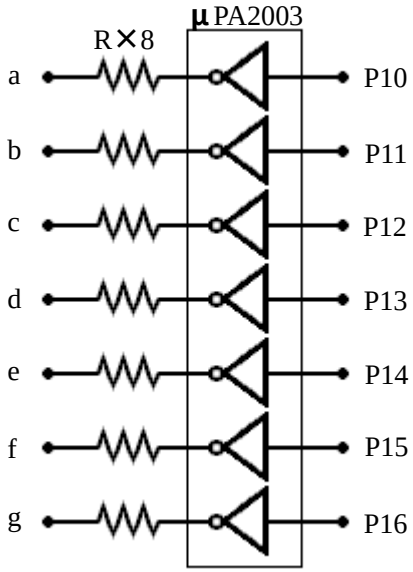

図 7.1-2　7 セグメント LED のカソード側の回路図

図中で a～g は 7 セグメント LED のセグメントに接続します。入力先は V853 のポートで、今回は P10～P16 を使用しています。

V853 のポートからの信号によって 7 セグメント LED への電流を ON/OFF するために、μPA2003 等のトランジスタ・アレイを用います。この回路を使用すると、ポートからの信号が ON（HIGH）になったとき、このカソード側の電圧が 0V になり、アノード側であるスイッチング回路から電流が流れるようになります。

トランジスタ・アレイと 7 セグメント LED に接続する途中の抵抗 R は、電流の調整を行うためのものです。抵抗値はトランジスタ・アレイの仕様（電気的特性）に依存しますので、データ・シートを参考に決めます。大型の 7 セグメント LED で、入力電圧が 6V～9V、μPA2003 を使用したときの抵抗値は、およそ 200Ω です。

LED も同じように駆動します。今回は LED 駆動の際、トランジスタ・アレイを使用せず、カソード側を 0V にしたままにし、アノード側を ON/OFF することで点灯します。図 7.1-3 で、カソード側は V853 ポートに接続します。今回は P20、P21 を使用しています。図中の抵抗 R は電流の調整を行うものです。およその抵抗値は 150Ω～1KΩ です。

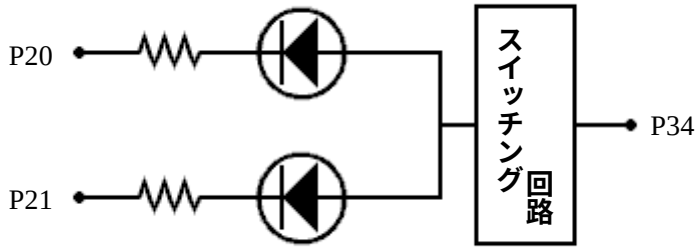

図 7.1-3　LED 駆動回路

図 7.1-4 は 7 セグメント LED、LED のアノード側であるスイッチング回路です。

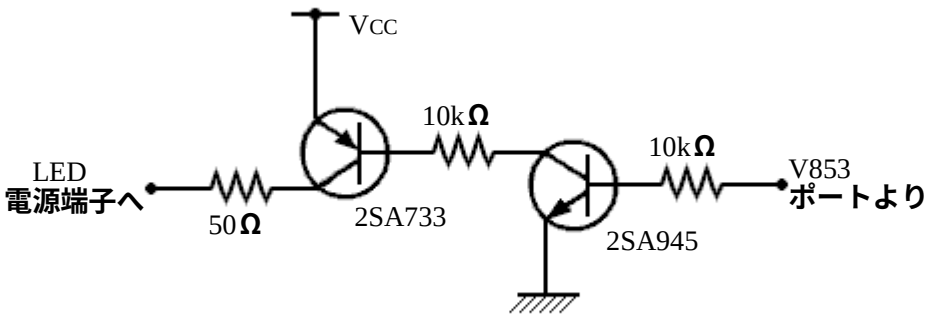

図 7.1-4　スイッチング回路（Vcc = 6V～9V）

入力は V853 ポートより行い、今回は 7 セグメント LED では P30～P33、LED では P34 を使用しています。図中の 50Ωの抵抗は LED への電流調整で、これによって LED の明るさ調整を行います。この回路における Vcc は 6V～9V が適当で、大型の 7 セグメント LED を使用した場合に最も適します。5V で駆動可能な 7 セグメント LED においても使用可能ですが、その場合は、このようなトランジスタ 2 個使用の回路を使わなくても、図 7.1-5 のようなトランジスタ 1 個使用の回路でも十分駆動可能です。LED の点灯だけなら、図 7.1-5 の回路で十分です。

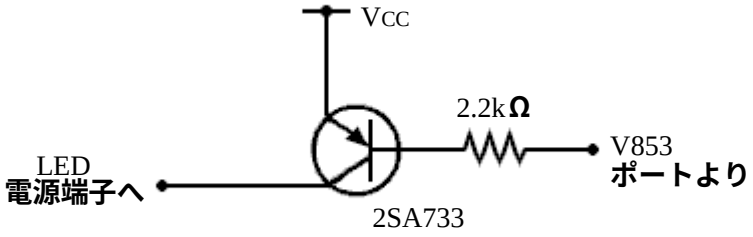

図 7.1-5　スイッチング回路（Vcc = 5V）

7.2　LCD ユニット

図 7.2-1 は LCD ユニットの接続図です。LCD は使用するデータ線を V853 のポートへ接続し、接続しないデータ線を接地します。電源部分は、図 7.2-1 のように可変抵抗を使用し、液晶の濃淡を調整できるようにしてあります。その必要がない時は VDD を Vcc に接続し、Vss を接地します。LCD ユニットの制御に使用する E、R/W、RS は、それぞれ P00～P02 へ接続します。

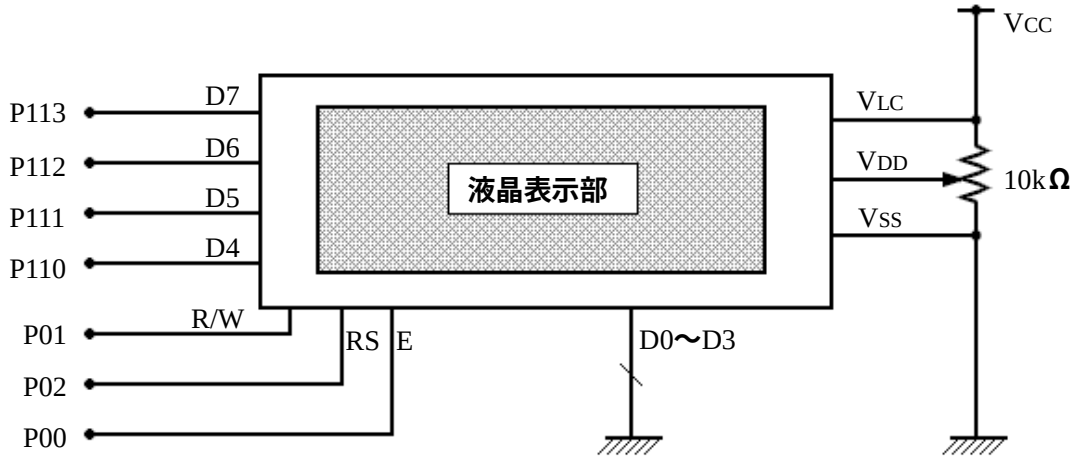

図 7.2-1　LCD ユニットとの接続図

7.3　タクト・スイッチ

図 7.3-1 はタクト・スイッチとの接続図です。この回路によってタクト・スイッチが押されると、信号レベルが HIGH になり、V853 のポートに入力されます。

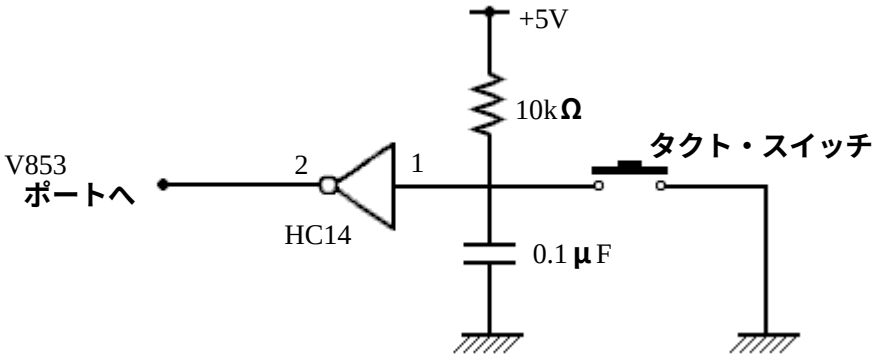

図 7.3-1　タクト・スイッチとの接続図

タクト・スイッチのような機械スイッチでは、接点が ON/OFF された瞬間に、電気的に ON/OFF が繰り返される「チャタリング」と呼ばれる現象が発生します。そのため一度しか押していないにもかかわらず、何度かスイッチが押された状態が作られてしまいます。それを回避するため、図 7.3-1 のようにコンデンサを接続します。チャタリング回避機能が内蔵されたタクト・スイッチであれば、そのまま接続しても問題ありません。

お問い合わせ先

【技術的なお問い合わせ先】

ＮＥＣ半導体テクニカルホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−435−9494
FAX　：044−435−9608
E-mail　：s-info@saed.tmg.nec.co.jp

【営業関係お問い合わせ先】

第一販売事業部
東　京　(03)3798−6106, 6107, 6108
名古屋　(052)222−2375
大　阪　(06)6945−3178, 3200, 3208, 3212
仙　台　(022)267−8740
郡　山　(024)923−5591
千　葉　(043)238−8116

第二販売事業部
東　京　(03)3798−6110, 6111, 6112
立　川　(042)526−5981, 6167
松　本　(0263)35−1662
静　岡　(054)254−4794
金　沢　(076)232−7303
松　山　(089)945−4149

第三販売事業部
東　京　(03)3798−6151, 6155, 6586, 1622, 1623, 6156
水　戸　(029)226−1702
広　島　(082)242−5504
高　崎　(027)326−1303
鳥　取　(0857)27−5313
太　田　(0276)46−4014
名古屋　(052)222−2170, 2190
福　岡　(092)261−2806

【資料の請求先】

上記営業関係お問い合わせ先またはNEC特約店へお申しつけください。

【インターネット電子デバイス・ニュース】

NECエレクトロンデバイスの情報がインターネットでご覧になれます。　URL(アドレス)　http://www.ic.nec.co.jp/

アンケート記入のお願い

お手数ですが，このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

［ドキュメント名］RX850（Ver.3.1）　リアルタイム・オペレーティング・システム　アプリケーション・ノート
（U13646JJ1V1ANJ1（第１版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他）　（　　）
ご住所　（　　）
お電話番号　（　　）
お仕事の内容　（　　）
お名前　（　　）

１．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
その他（　　）					
（　　）					

２．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

３．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

４．ご意見，ご要望

５．このドキュメントをお届けしたのは

NEC販売員，特約店販売員，その他（　　）

ご協力ありがとうございました。
下記あてにFAXで送信いただくか，最寄りの販売員にコピーをお渡しください。

日本電気（株）NECエレクトロンデバイス
半導体テクニカルホットライン
FAX：（044）435-9608

2000.6

キリトリ