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

RENESAS

保守／廃止

RX136, RX423

リアルタイム・オペレーティング・システム

LEDアレイ表示システム編

対象デバイス
V53™
V55PI™

資料番号 EEA-617（第1版）
発行年月 May 1994 P

保守／廃止

V53, V55PI, Vシリーズは日本電気株式会社の商標です。
インターツールは米国インターメトリックス・マイクロシステムズ・ソフトウェア社の商標です。

本資料に掲載の応用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。

M7A 93.9

保守／廃止

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。

保守／廃止

はじめに

対象者　このマニュアルはRX136, RX423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。

目　的　このマニュアルでは「LEDアレイ表示システム」を次の２つの組み合わせで作成する手順を例として，RX136, RX423を用いたアプリケーション・プログラムの作成方法と各種開発ツールの使用方法を理解していただくことを目的としています。

- RX136とコンパイラ(MS-C:マイクロソフト社製)とロケータ(LC70116)…μPD70236（V53）使用
- RX423とインターツール™(SP70116-I)…μPD70433（V55PI）使用

構　成　このマニュアルは大きく分けて次の内容で構成しています。

- システム概説
- ハードウエア構成
- ソフトウエア開発（RX136用, RX423用）
- コーディング・リスト
- 回路図
- アプリケーション・システムの使用例

読み方　このマニュアルの読者には，電気，論理回路，およびマイクロコンピュータに関する一般的知識を必要とします。

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

保守／廃止

関連資料

●RX136, RX423に関する資料

製品名	資料名	資料番号
RX136 (Ver1.2)	ユーザーズ・マニュアル　基礎編	EEU-707
	〃　テクニカル編	EEU-719
	〃　ニュークリアス・インストレーション編	EEU-775
	アプリケーション・ノート　LEDアレイ表示システム編	このマニュアル
RX423 (Ver1.0)	ユーザーズ・マニュアル　基礎編	EEU-845
	〃　テクニカル編	EEU-868
	〃　ニュークリアス・インストレーション編	EEU-844
	アプリケーション・ノート　LEDアレイ表示システム編	このマニュアル

●V53, V55PIに関する資料

製品名	資料名	資料番号
V53	データ・シート	IC-7790
	ユーザーズ・マニュアル	IEU-680
V55PI	データ・シート	IC-8257
	ユーザーズ・マニュアル　ハードウエア編	IEU-776
	〃　命令編	IEU-812

●開発ツールに関する資料

製品名	資料名	資料番号
EB-70433	ユーザーズ・マニュアル	EEU-811
LC70116	〃	EEU-834
RA70116-I,	〃　操作編	EEU-869
SP70116-I	〃　アセンブリ言語編	EEU-861

保守／廃止

目 次 要 約

保守／廃止

保守／廃止

目　次

図の目次(1/2)

保守／廃止

図の目次(2/2)

表の目次

保守／廃止

保守／廃止

リストの目次(1/2)

保守／廃止

リストの目次(2/2)

保守／廃止

第1章　概　説

16ビットVシリーズ™には各製品に対応した4種類のリアルタイムOS（RX116, RX136, RX320, RX423）が用意されており，これらのリアルタイムOSを使用してアプリケーション・プログラムを作成するためのさまざまなソフトウエア開発ツールが使用できます。

このアプリケーション・ノートでは「LEDアレイ表示システム」を基に，次の2種類のアプリケーション・プログラムの作成例を示しています。

- RX136とCコンパイラ(MS-C), ロケータ(LC70116)を使用したアプリケーション・プログラム(V53使用)
- RX423とインターツール(SP70116-I)を使用したアプリケーション・プログラム(V55PI使用)

上記のアプリケーション・プログラムを作成するにあたり，おもに次の項目について説明します。

- LEDアレイをソフトウエアでドライブする方法
- シリアル・データの送受信の方法
- 拡張アドレス空間に配置したデータの使用方法
- 拡張アドレス空間に配置したタスクの使用方法（RX136対象）
- C言語による特殊機能レジスタ（SFR）の使用方法（RX423対象）
- C言語による割り込みハンドラの記述方法

保守／廃止

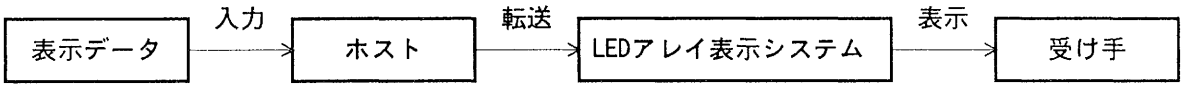
第２章　システム概説

２．１　概　要

LEDアレイ表示システムは，広告やニュースの文字データをLEDアレイに表示して，不特定多数の人に情報を伝えるシステムです。

表示するデータはホストとなるパーソナル・コンピュータ（以下ホストと略す）から送られ，指定された方法でデータを表示します。

図２－１　LEDアレイ表示システムの位置づけ

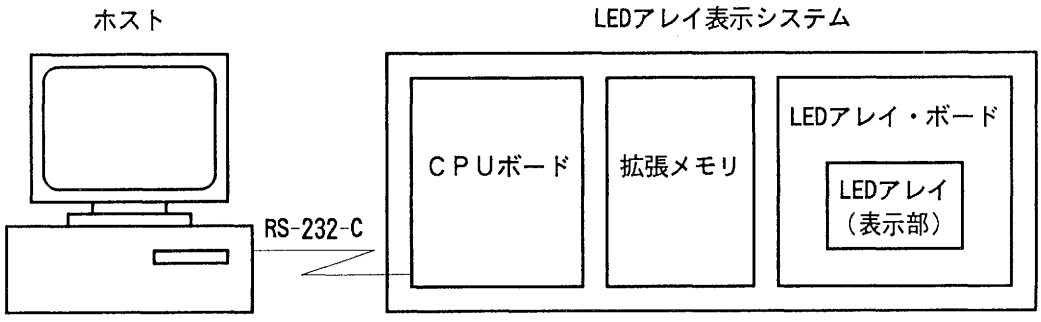


２．２　システムの構成

システムの構成を図２－２に示します。

システムはRS-232-Cで接続されたホストからのデータを受け取り，指定された方法でデータを表示します。CPUにはV53かV55PIのどちらかを使用します。システムのハードウエア関連の詳細については**第３章　ハードウエア**を参照してください。

図２－２　システムの構成



２．３　機能の概要

システムの機能は「ホストとのデータ通信」と「LEDアレイへのデータ表示」に大別されます。機能の概要を表２－１に示します。

システムはホストから送られてきたデータによって，リアルタイムで動作を変更します。

表２－１　システムの機能概要

分　類		機　能
ホストとのデータ通信	ホストからの受信	ホストからのコマンドを受信して実行する
		ホストからの表示データを受信して登録する
		ホストからの時刻データを受信してシステム時刻を変更する
	ホストへの送信	ホストからのコマンドにより登録されている表示データを送信する
		ホストからのコマンドにより現在のシステム時刻を送信する
LEDアレイへのデータ表示		現在の表示モードに従って表示データを選択する
		表示データの文字列をフォント・データに変換する
		ダイナミック点灯方式によりフォント・データをLEDアレイに表示する

２．４　管理するデータの概要

システムで扱うデータの概要を表２－２に示します。

表２－２　管理データの概要（１／２）

データの項目	データの種類	内　容
表示データ	時刻表示データ	現在時刻を表示する文字列データ
	コマーシャル表示データ	宣伝に用いられる文字列データ
	ニュース表示データ	ニュースに用いられる文字列データ
	緊急表示データ	緊急に伝えるために用いられる文字列データ
表示モード	時刻表示モード	時刻データだけを表示（緊急データは無条件に表示）
	コマーシャル表示モード	コマーシャル・データだけを表示（緊急データは無条件に表示）
	ニュース表示モード	ニュース・データだけを表示（緊急データは無条件に表示）
	混在表示モード	すべての表示データを交互に表示（緊急データが優先）
	緊急表示モード	緊急データだけを表示（その他の表示データは保留）

表２－２　管理データの概要（２/２）

データの項目	データの種類	内　容
コマンド・データ（ホストから受信）	モード設定コマンド	指定されたモードに遷移
	時刻設定コマンド	指定された時刻に設定
	表示データ登録コマンド	指定された文字列を指定したモードの表示データに登録
	表示データ削除コマンド	指定されたモードの表示データをすべて削除
	表示データ出力コマンド	指定されたモードに登録されている表示データをすべて出力
	時刻出力コマンド	現在時刻を出力
出力データ（ホストへ送信）	表示データ	表示データ出力コマンドに対する表示データを出力
	現在時刻データ	時刻出力コマンドに対する時刻データを出力
システム・データ	システム時刻	00:00-23:59までの24時間の時刻データ（１分単位）
	現在の表示モード	現在の表示モードを保持しているデータ

２．５　表示モードの遷移

表示モードはモード設定コマンドの受信により表２－３のように遷移します。

表２－３　表示モードの遷移

遷移前のモード	遷移条件	遷移後のモード
システム起動前	システム起動	混在表示モード
すべてのモード	コマーシャル表示モードへの変更コマンド受信	コマーシャル表示モード
	ニュース表示モードへの変更コマンド受信	ニュース表示モード
	時刻表示モードへの変更コマンド受信	時刻表示モード
	混在表示モードへの変更コマンド受信	混在表示モード
	緊急表示モードへの変更コマンド受信	緊急表示モード

保守／廃止

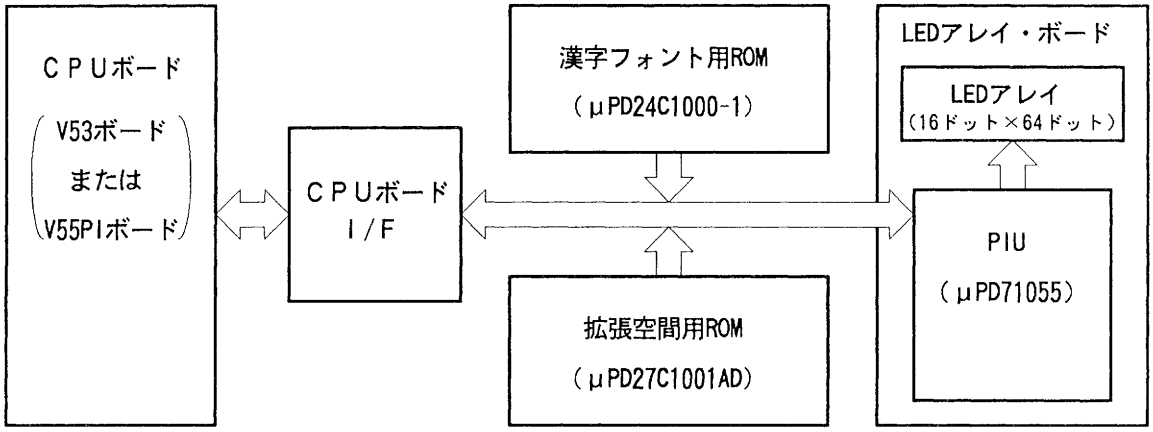
保守／廃止

第３章　ハードウエア

３．１　ハードウエア構成

システムのハードウエア構成は図３－１のようになります。

図３－１　ハードウエア構成



- ●CPUボード　：V53を搭載した「V53ボード」かV55PIを搭載した「V55PIボード」を使用します。
- ●CPUボードI/F　：アドレスのデコードを行います。
- ●PIU　：LEDアレイ・ボードへデータを出力するためのポートです。
- ●漢字フォント用ROM：拡張アドレス空間に配置された漢字フォントを取り出すためのROMです。JIS第一水準漢字が使用できます。
- ●拡張空間用ROM　：半角フォント・データとV53ボード用のプログラムを格納するためのROMです。
- ●LEDアレイ　：縦16ドット，横64ドットのLEDアレイです。同時に漢字４文字を表示できます。

３．２　CPUボード

システムではCPUボードとしてV53を搭載した「V53ボード」か，V55PIを搭載した「V55PIボード」を使用します。

３．２．１　V53ボード

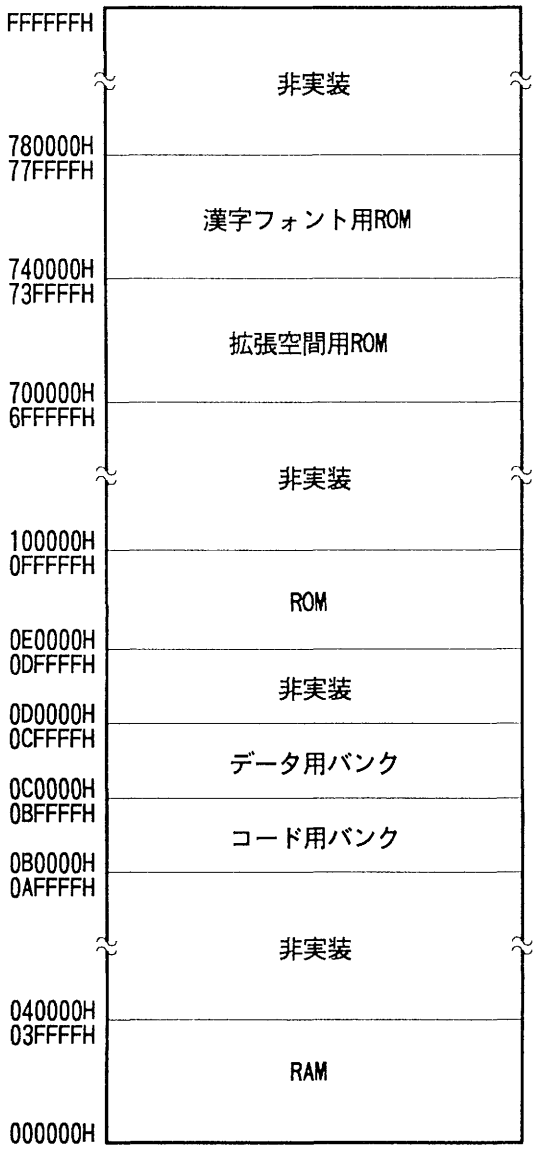
V53を搭載したCPUボードとして「SCB V53-M」（(株)システムサコム製）を使用します。
(株)システムサコムの問い合わせ先は次のとおりです。

〒130 東京都墨田区両国4-38-16
TEL(03)3635-5145
FAX(03)3635-5148

(1)メモリ・マップ

V53ボード使用時のシステムのメモリ・マップを図３-２に示します。

図３-２　V53ボード使用時のメモリ・マップ

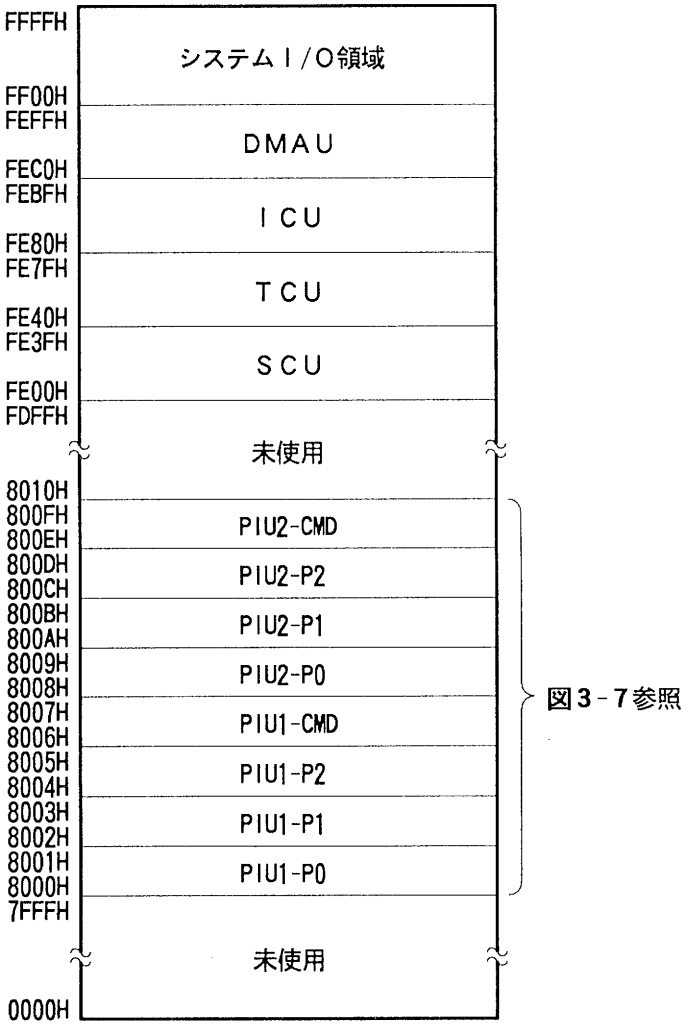


（２）I/Oマップ

V53内蔵のSCU，TCU，ICU，DMAUのI/OアドレスはV53の内蔵ペリフェラル・リロケーション・レジスタによって設定します。

LEDアレイへのデータ出力のためのI/Oは8000H番地から配置されます。

図３－３　V53ボード使用時のI/Oマップ



CMD:コマンド・レジスタ

(３)外部割り込み

V53ボードで使用する外部割り込みを表３－１に示します。

表３－１　V53ボード使用時の外部割り込み

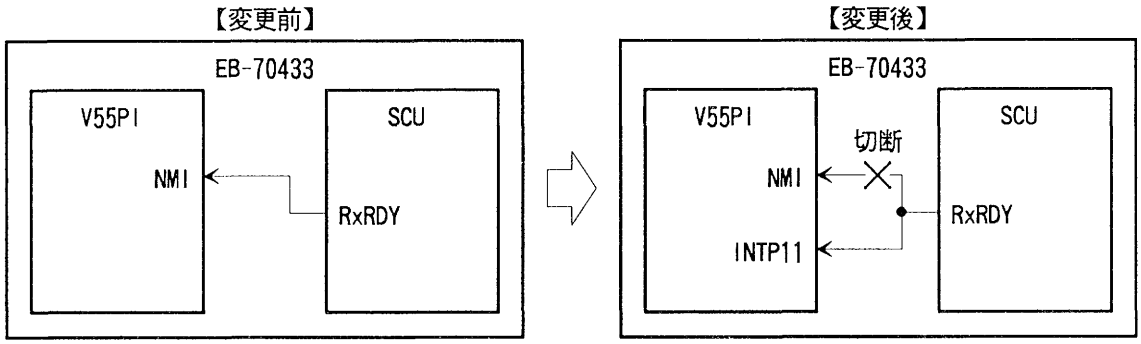
割り込み要因	割り込み要求入力端子	ベクタ番号	処理内容
タイマ割り込み	INTP0	38H	RX136が使用
シリアル受信完了割り込み	INTP1	39H	受信データの取り出し

３．２．２　V55PIボード

V55PIを搭載したCPUボードとして「EB-70433」（NEC製）を使用します。これは，V55PIの評価用ボードです。詳細についてはEB-70433 ユーザーズ・マニュアルを参照してください。

EB-70433ではシリアル・コントローラ（SCU:μPD71051）がV55PIに接続されており，SCUからの受信完了割り込み要求信号（RxRDY）はV55PIのNMI端子に入力されます。しかし，NMIの割り込み処理中ではRX423のシステム・コールが使用できないので，このアプリケーション・ノート用にRxRDY端子とNMI端子の間の接続を切断し，RxRDY端子とV55PIのINTP11端子を接続して使用します。

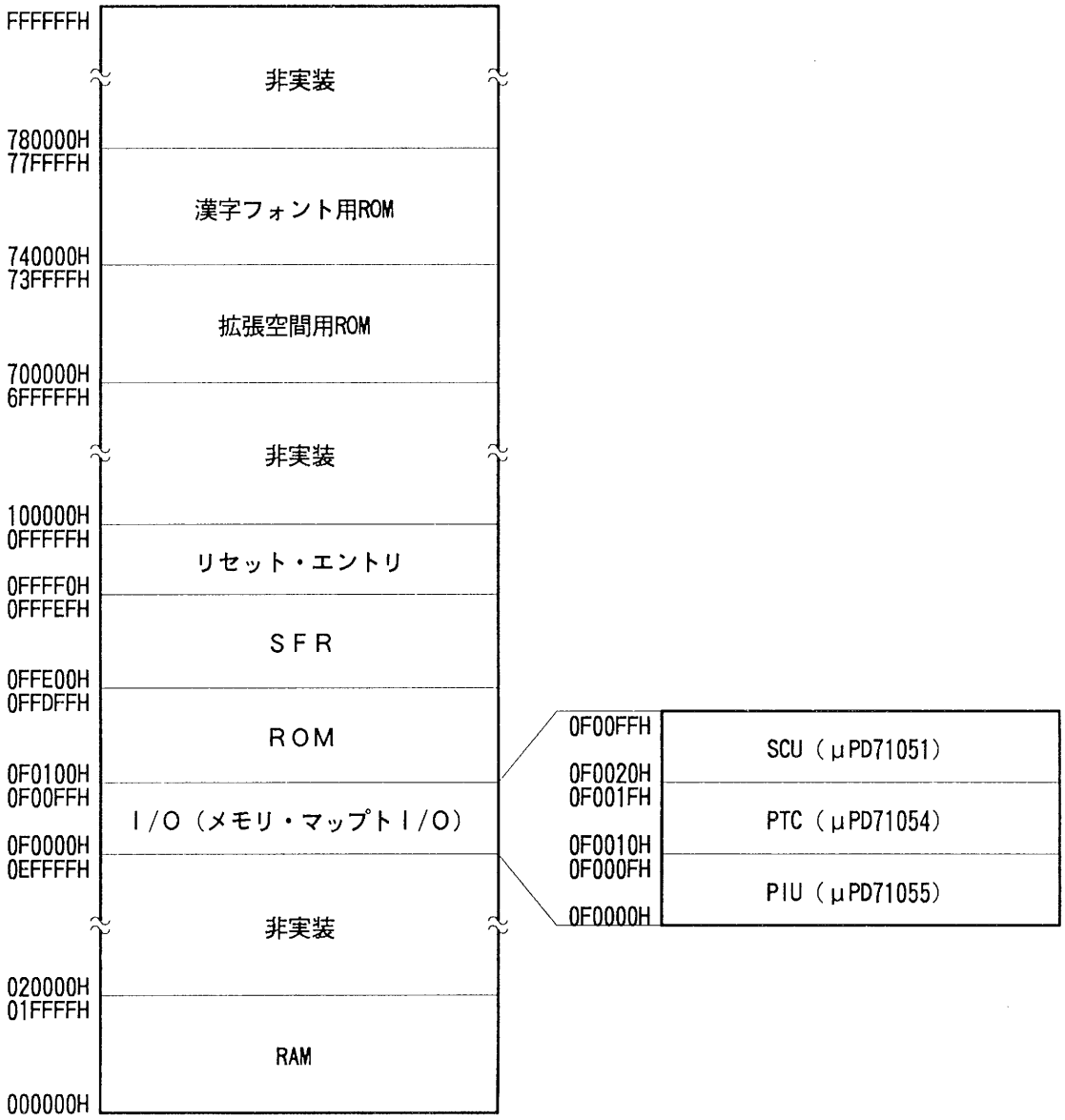
図３－４　V55PIボードの変更



（１）メモリ・マップ

V55PIボード使用時のシステムのメモリ・マップを図３－５に示します。

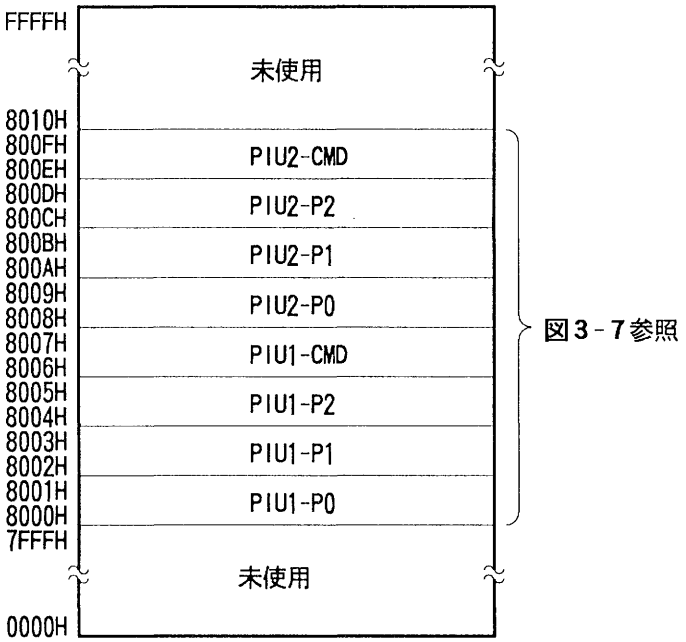
図３－５　V55PIボード使用時のメモリ・マップ



(２)I/Oマップ

LEDアレイへのデータ出力のためのI/Oは8000H番地から配置されます。

図３－６　V55PIボード使用時のI/Oマップ



CMD:コマンド・レジスタ

(３)外部割り込み

V55PIボードで使用する外部割り込みを表３－２に示します。

表３－２　V55PIボード使用時の外部割り込み

割り込み要因	割り込み要求入力端子	ベクタ番号	処理内容
タイマ割り込み	INTCM00	10H	RX423が使用
シリアル受信完了割り込み	INTP1	0AH	受信データの取り出し

３．３　LEDアレイ・ボード

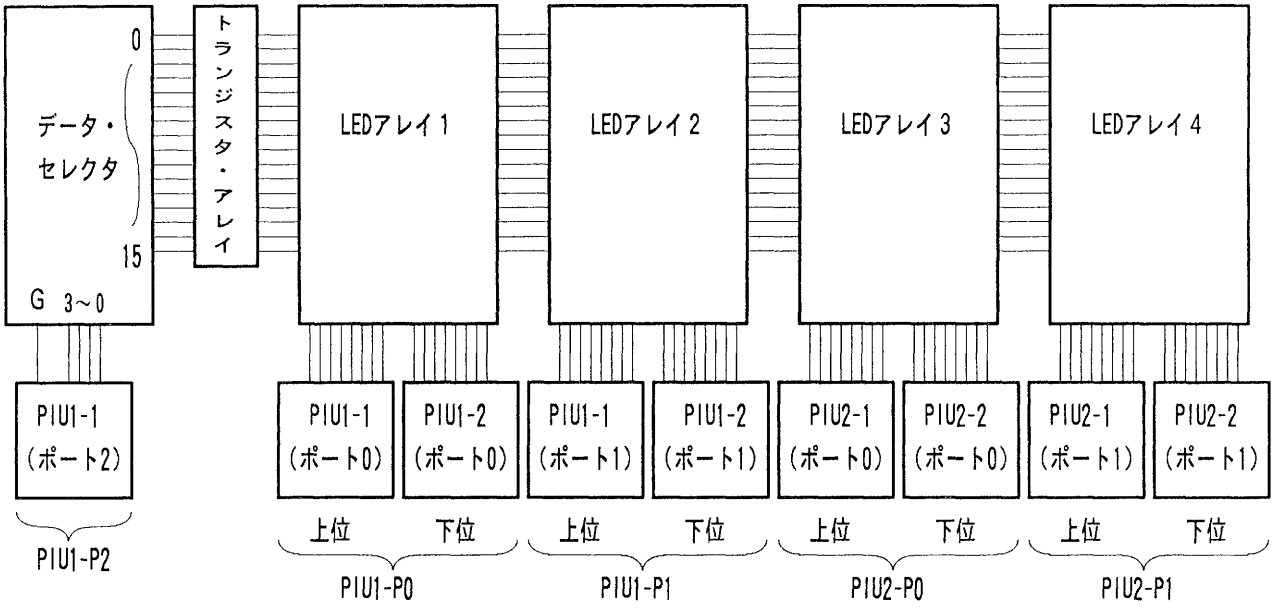
LEDアレイ・ボードでは，パラレル・インタフェース・ユニット（μPD71055）を４つ（PIU1-1,PIU1-2,PIU2-1,PIU2-2（図３－７参照））使用してます。このうち２つずつ並列にLEDアレイに接続しているので，I/Oポートはワード単位でデータをアクセスします。

また，縦16ビット，横64ビットのLED表示アレイを備えており，ダイナミック点灯方式によって使用します。ダイナミック点灯方式によるLEDアレイ・ボードの点灯手順は次のとおりです。

（ａ）縦の何列目を点灯するかを選択（PIU1-P2に列番号を出力）
（ｂ）横１列分（64ビット）の表示データをセット（PIU1-P0，P1とPIU2-P0，P1にフォント・データを出力）
（ｃ）１ms待つ
（ｄ）以上の繰り返し

上記の（ａ）-（ｄ）の手順により，16ms間隔でボード全体の書き換えが行われ，人間の目には全体のLEDが点灯しているように見えます。

図３－７　LEDアレイ・ボード



保守／廃止

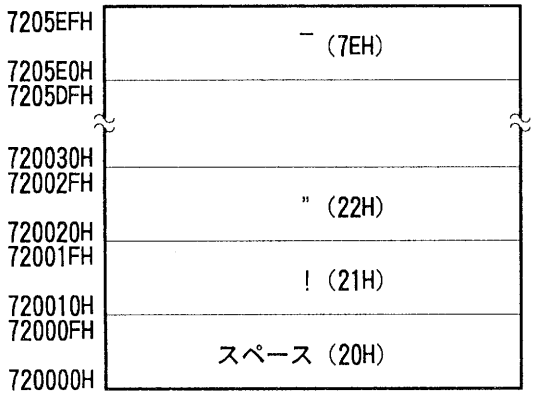
３．４　漢字フォント用ROM

漢字フォント用のROMとしてμPD24C1000-1を使用します。このROMは拡張アドレス空間に配置され，JIS第一水準の漢字が使用できます。メモリ・マップの740000H番地以降の偶数番地だけに接続されます（奇数番地にはアクセスできません）。

３．５　拡張空間用ROM

拡張アドレス空間に配置されるROMとしてμPD27C1001ADを使用します。半角文字のフォント・データが格納され，V53ボードを使用した場合はタスク・プログラムの一部も配置します。半角フォント・データはメモリ・マップの720000H番地から書き込まれており，次の図のような内容です。

図３-８　半角フォント・データのメモリ・マップ



備考　（ ）はコード番号を示します。

一文字の半角コードに対して，16バイト（横８ビット×縦16ビット）のフォント・データとなります。それぞれ拡張空間用ROM中の下位アドレスが表示桁の上となり，１バイト中の最上位ビットが表示文字の左側となります。

第4章　ソフトウエア

この章では，RX136,RX423を用いてアプリケーション・システムを構築する方法について例を挙げて説明します。

4．1　全体構成

システムの全体構成を図4－1に示します。

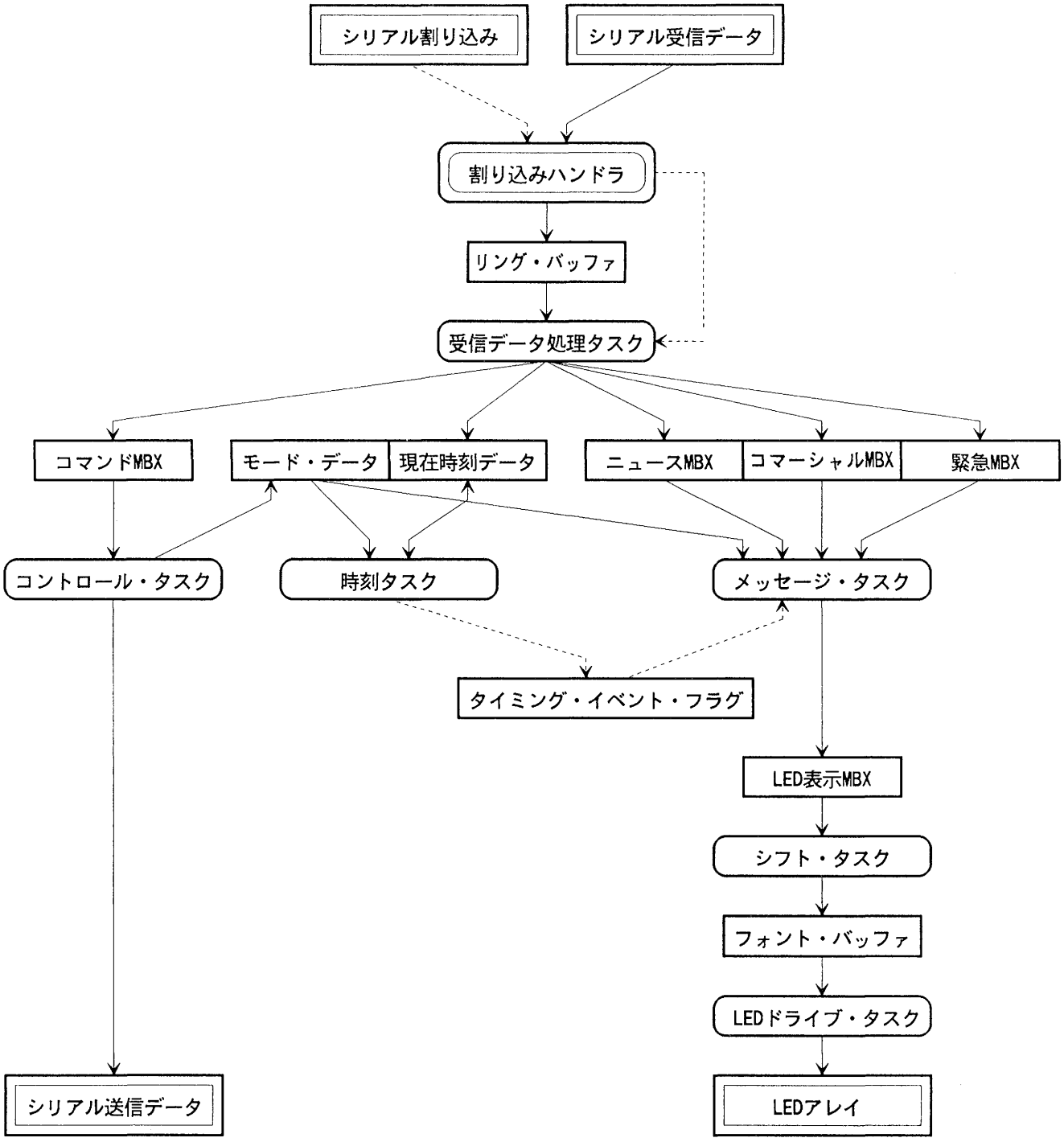
システムはデータの管理全般にメールボックスの特徴を生かした構成となっています。ここで使用するタスクの概要は次のとおりです。

表4－1　タスクの概要

タスク名	概　要
受信データ処理タスク	ホストから送られてきたコマンドや表示データを処理し，該当するタスクに通知する。
コントロール・タスク	表示モードの変更処理と，表示データのホストへの出力を行う。
時刻タスク	1分ごとに起床し，アプリケーション・システムが管理する時刻に関する情報の更新を行う。
メッセージ・タスク	メールボックスに蓄えられた表示データを表示モードに従って選択し，表示するタスクに送信する。
シフト・タスク	LEDアレイに表示するデータを受け取り，LEDアレイのビット・パターンに変換したあとLEDアレイに表示している内容を一定時間ごとに右側へシフトする。
LEDドライブ・タスク	シフト・タスクによってセットされる表示パターンをLEDアレイに出力する。

保守／廃止

図４－１　全体構成



備考　MBX　：メールボックス

：データの受け渡し

：事象の受け渡し

：外部デバイス

：データ，資源

：ハンドラ

：タスク

4．2　各機能の説明

このアプリケーション・システムで用いる代表的な機構について説明します。

4．2．1　メッセージ管理の方法

メールボックスを用いたタスク間通信の長所はデータと事象を同時に受け渡しできることです。

標準的な使用方法では図4-2(a)のようにメッセージを送るとき，メモリ・プールからget_blkシステム・コールによってメモリ・ブロックを取得してからsnd_msgシステム・コールによって送信し，受け手側はrcv_msgシステム・コールでメッセージを受信して，不要になった受信メッセージをrel_blkシステム・コールによって解放します。

このアプリケーション・システムでは，図4-2(b)で示すような方法を用い，メールボックスに同時に送られるメッセージの最大数をあらかじめ決めることによって，容易にソフトウエアの設計ができるようにしています。

この方法は，あるメールボックスに送られるメッセージの大きさを定め，あらかじめ「フリー・メールボックス」に用意しておきます。フリー・メールボックスは通常のメールボックスと同じものですが，役割はメッセージの獲得のためのメモリ・プールと同じように未使用のメッセージを管理する目的で使用します。この未使用のメッセージをここでは「フリー・メッセージ」と呼びます。

タスクが通信のためのメールボックスにメッセージを送るときには，まずフリー・メールボックスからrcv_msgシステム・コールによってフリー・メッセージを確保して送信内容を書き込み，snd_msgシステム・コールによって送信します。

受け手側のタスクが受信したあと，不要になったメッセージはsnd_msgシステム・コールによってフリー・メールボックスに返却します。

図4-2　メッセージの使用方法（1/2）

(a)標準的な方法

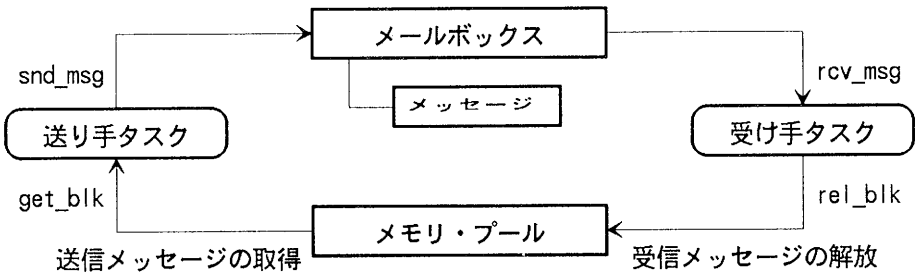
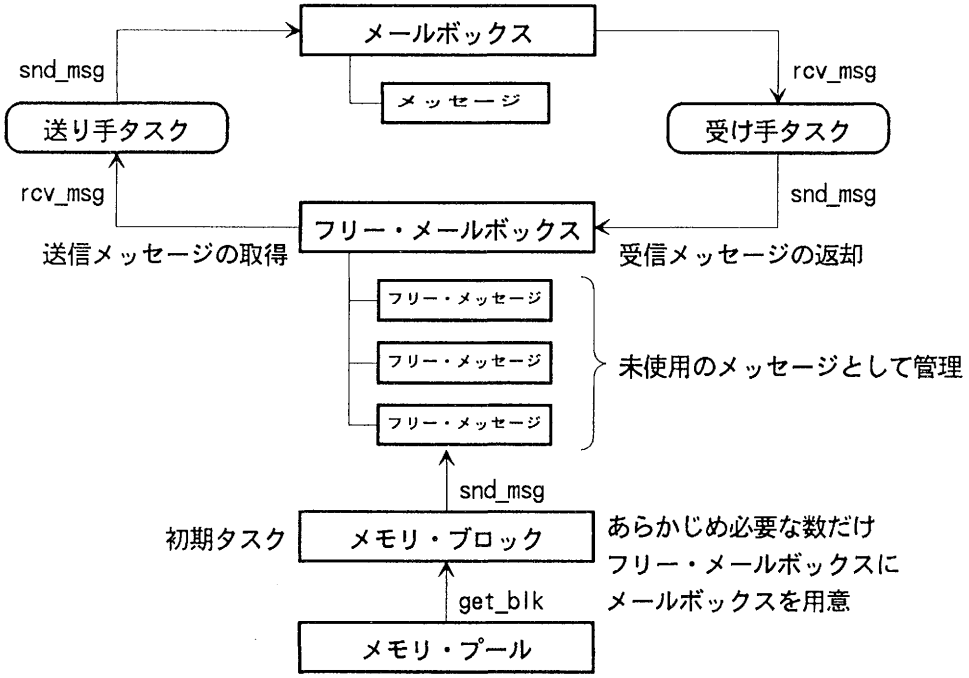


図4-2　メッセージの使用方法（2/2）

（b）アプリケーション・システムでの方法

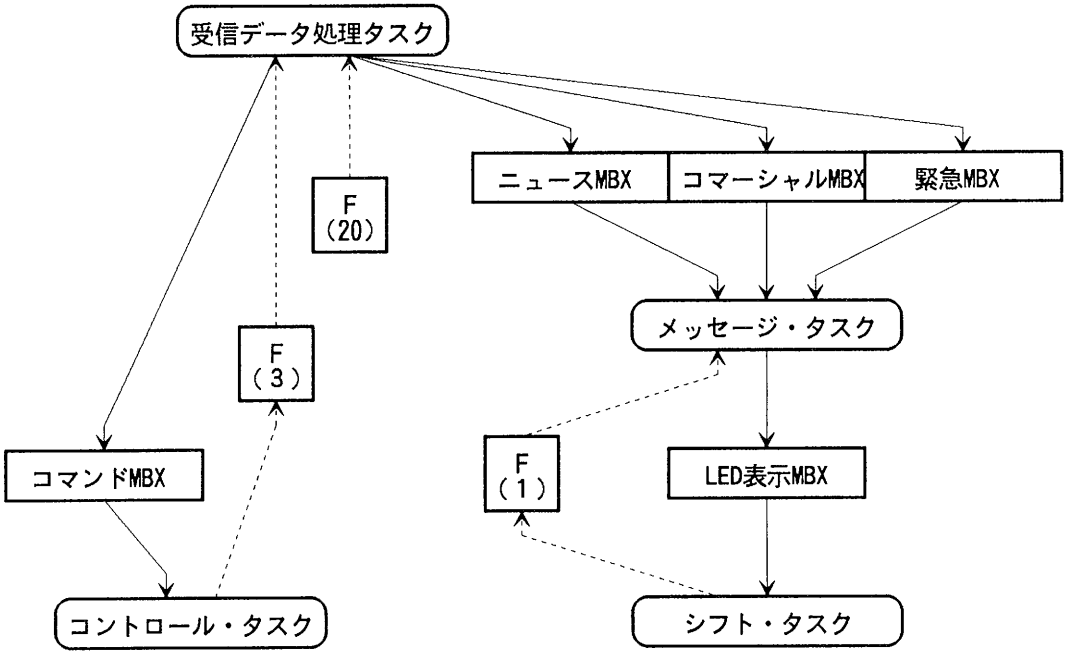


４．２．２　メールボックスの構成

アプリケーション・システムでは図４－３に示すように図４－２（ｂ）の機構を利用した構成を使います。

フリー・メールボックスを使用することにより，タスク間で一度に受け渡しされるメッセージの最大数を限定し，メッセージを効率的に使用できます。

図４－３　メールボックスの構成



備考　MBX　　：メールボックス

⟶　：有効なメッセージの送信/受信

⇢　：フリー・メッセージの取得/返却

F　：フリー・メールボックス（かっこ内の数字は最初に準備するフリー・メッセージの個数）

保守／廃止

4．2．3　シリアル受信データの取り込み

シリアル・コントロール・ユニット（SCU）が1バイトの受信を完了すると受信完了割り込みが発生し，受信完了割り込みの割り込みハンドラによって「リング・バッファ」に受信データを格納します。リング・バッファの構造に関しては，**4．3　データの説明**を参照してください。

また，シリアル受信完了割り込みハンドラによって受信データ処理タスクを起動します。

4．2．4　受信データの処理

シリアル受信完了割り込みハンドラから受信データ処理タスクが起床され，リング・バッファに読み込んだデータの処理を行います。

受信データは大きく分けてホストからのコマンド・データと表示メッセージ・データの2種類があり，受信処理タスクは受信データの内容によって処理を切り分けます。これらホストからのコマンド・フォーマットの詳細は**4．3（9）通信データ・フォーマット**を参照してください。

コマンド・データの場合，コマンド用のメールボックスに対してメッセージを送り，コントロール・タスクが実際の処理を行います。未実行のコマンド・データは最大3つまで保留されます。

表示データの場合，ニュース，コマーシャル，緊急の3種類がありますが，それぞれのメールボックスにメッセージの形で送信し，そのまま保存されます。

ニュース，コマーシャル，緊急の表示データのメールボックスには合計で最大20までのメッセージが格納できます。格納している表示データが20を越えた場合には，越えた分の表示データを無視します。

4．2．5　表示データの管理

アプリケーション・システムではすべての表示データをメールボックスによって管理しています。つまり，表示データは，メールボックスにメッセージの形で保存されます。表示データの追加，参照，削除の機能はすべてsnd_msgとrcv_msg のシステム・コールによって行います。なお，これらのメールボックスはすべてFIFOオプションをつけて生成します。

表示データの保持のために，ニュース，コマーシャル，緊急のそれぞれのデータのための3つのメールボックスと，これらのアクセスの排他制御のためのセマフォによって表示データ管理の機構を構成しています（**図4－4　表示データの管理**参照）。

このセマフォは最大カウント値を1として生成します。また，受信データ処理タスクの表示データ用メッセージの取得と表示終了後のメッセージの返却のためにフリー・メールボックスを1つ用います。

すべての表示データを参照するするために「マーキング・メッセージ」を使います。これは，メモリ・プールから取得した通常のメッセージと同じものですが，メッセージの中のデータにマーキング・メッセージであることを示すデータが書き込まれ，表示データとは区別できるようにしています。

これらメッセージ内容の詳細は**4．3　データの説明**を参照してください。

(1)表示データの追加

表示データの追加は，表示データ用フリー・メールボックスからフリー・メッセージを取得し，該当するメールボックスに送信します。この送信はsnd_msgシステム・コールを使用します。このときのシステム・コールの発行は表示データ制御セマフォに対するwai_semとsig_semで囲んで使用し，他のタスクとの排他的な制御を行います。

(2)表示データの参照

表示データの参照には表示種別ごとのメールボックスに連結されている先頭の表示データのメッセージを１つだけ取り出す場合と全部の表示データをもれなく連続して参照する場合の二通りがあります。

先頭のメッセージを参照する場合は，表示データ制御セマフォで排他制御を行ってrcv_msgによって取り出し，参照したあと，snd_msgにより同じメールボックスに返却します。この処理はメッセージ・タスクがシフト・タスクに送る表示データを取り出すときに使用できます。このとき，返却された表示データはそのメッセージの最後尾に連結されます。

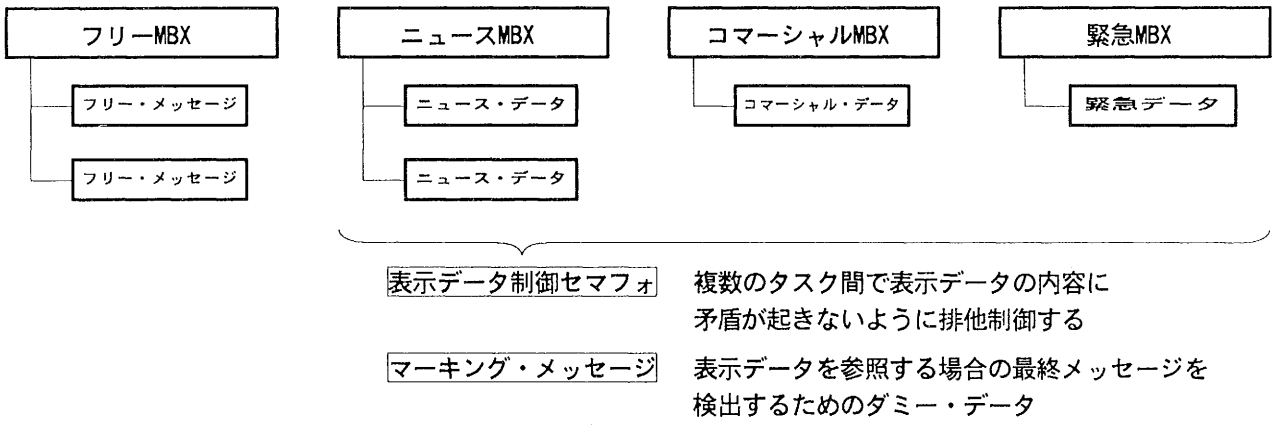
ホストへ現在の表示データを出力する場合に，すべてのメッセージを参照します。

まず，wai_semにより排他制御を開始し，対象となる表示データのメールボックスに対してマーキング・メッセージをsnd_msgによって送ります。そして，このマーキング・メッセージが現れるまでrcv_msgによってメッセージを取り出し，参照後snd_msgによって戻します。マーキング・メッセージが現れた場合はsig_semシステム・コールによって排他制御を終了します。これらのメールボックスのメッセージの変化については**図4-5　表示データの参照**を参照してください。

(3)表示データの削除

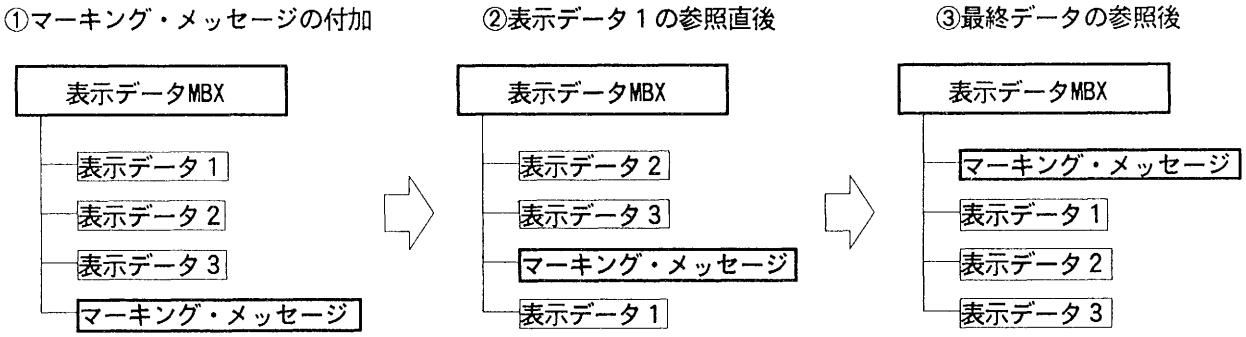
表示データをすべて削除するときは，表示データ制御セマフォによって排他制御を行ったうえで，すべての表示データをrcv_msgにより取り出し，フリー・メールボックスに解放します。

図4-4　表示データの管理



保守／廃止

図４－５　表示データの参照



４．２．６　時刻データの管理

LEDアレイに表示する時刻データは，OSの時刻とは別にアプリケーション側で作っています。

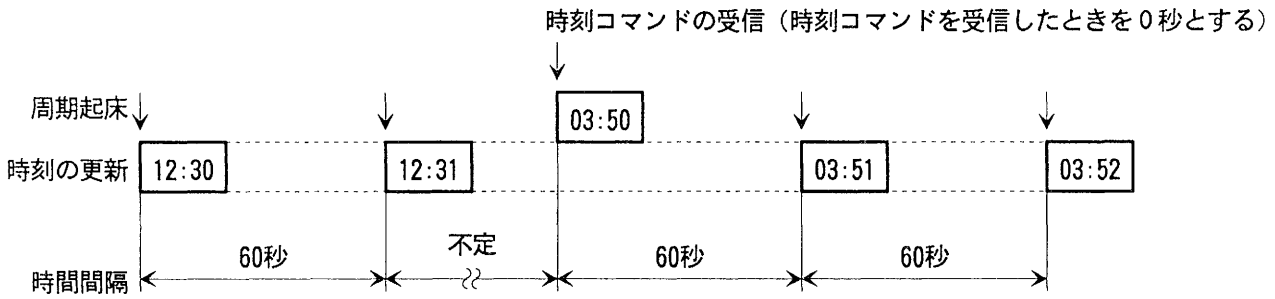
周期起床によって60秒間隔で時刻タスクを起床し，時刻タスクは時刻データを更新して時刻タイミング用のイベント・フラグにビットをセットします。

このイベント・フラグはメッセージ・タスクによって調べられ，ビットがセットされているときは，時刻データを表示します。

受信タスクがホストからの時刻設定コマンドを受信したとき，受信時刻を指定時刻として設定します。また，このとき同時にアプリケーションのシステム情報の中に時刻設定が行われたことを記録し，時刻タスクを起床します。

時刻タスクは起床されたあとシステム情報を調べ，時刻設定が行われている場合には，一度，周期起床を停止して新たに60秒間隔での周期起床を掛け直します。

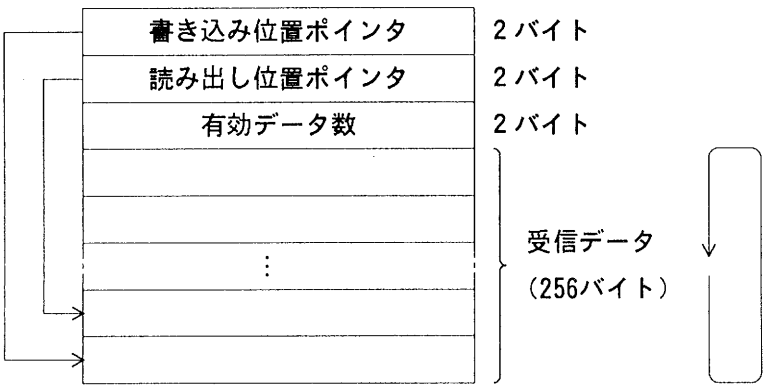
図４－６　時刻の設定例



4．3　データの説明

この節ではシステム内で使用するデータを説明します。データは図の下が上位アドレスです。

(1)リング・バッファ

機　能	シリアルからの受信データを蓄える（バッファ）
構　造	
説　明	論理的にはリング状の形となる受信データを蓄えるためのバッファです。 シリアル受信完了の割り込みハンドラは書き込み位置ポインタの位置から書き込み，受信データ処理タスクは読み出し位置ポインタの位置から読み出します。それぞれのポインタは使われるごとにインクリメントしますが，バッファの最後まで行くとバッファの先頭にポインタを戻します。また，有効データ数は常に取り出し可能なデータ数をバイト単位で保持します。

(2)モード・データ

機　能	現在のシステムの表示モードを保持
構　造	モード・データ　1バイト
説　明	30H：時刻表示モード 31H：ニュース表示モード 32H：コマーシャル表示モード 34H：緊急表示モード 38H：混在表示モード

(3)現在時刻データ

機　能	表示用の時刻を保持
構　造	現在の「時間」　2バイト 現在の「分」　2バイト
説　明	OSの持つ時刻から「時間」と「分」を取り出したデータです。24時間分の時刻を使用し，23:59の次は00:00になります。

(4)時刻変更フラグ

機　能	受信コマンドによって時刻が変更されたかどうかを表示
構　造	時刻変更フラグ　2バイト
説　明	00H：時刻は更新されていない 01H：時刻が更新された
	受信タスクが時刻変更コマンドを受け取り，時刻を更新したことをメッセージ・タスクに伝えるために使用します。このとき，このフラグをセットしたあと，時刻タスクを起床します。時刻タスクはこのフラグがセットされていることが分かった場合，周期起床を掛け直します。

(5)コマンド・メッセージ・ブロック

機　能	コントロール・タスクに対して処理要求を行うメッセージ・ブロック
構　造	未使用　2バイト 使用後の返却MBX　2バイト コマンド　1バイト モード　1バイト
説　明	31H：モードで示されるメッセージの削除 32H：モードで示されるモードへの遷移 38H：モードで示される現在の表示データのシリアルへの出力
	このメッセージ・ブロックはコマンド・メールボックスに送信され，コントロール・タスクが受け取って処理されます。使用後の返却MBXは，このメッセージが受信されたあとに返却するメールボックスのアクセス・アドレスです。モードは上記(2)と同じです。ただし，コマンドが31Hと38Hの場合の混在モードは指定できません。

(6)ニュース，コマーシャル，緊急メッセージ・ブロック

機　能	ニュース，コマーシャル，緊急の各表示データを保持
構　造	未使用　2バイト 使用後の返却MBX　2バイト メッセージID　2バイト 表示残り回数　2バイト 表示データ ⋮ 表示データ　256バイト
説　明	メッセージIDは，通常の表示データとして0000H以外を設定します。0000Hの場合にはマーキング・ブロックを示します。

(7)LEDメッセージ・ブロック

機　能	シフト・タスクに対して送られる表示データ
構　造	メッセージ優先度　2バイト 使用後の返却MBX　2バイト 表示データ ⋮ 表示データ　256バイト
説　明	LEDに表示する文字列が表示データとして格納されます。 メッセージの優先度は次のとおりです。 緊急データ（最高優先）＞時刻データ＞ニュース，コマーシャル・データ

(8)フォント・バッファ

機　能	LEDアレイに出力されるビット・パターンのデータを保持
構　造	データ ⋮ データ } 160バイト（2バイト×16ライン×5桁）
説　明	先頭から32バイトごとにLEDアレイの1-4桁目に表示されるビット・パターンを示します。5桁目のビット・パターンは次に表示される文字のデータです。次に表示するビット・パターンは5桁目にセットします。また，常に1-4桁目のビット・パターンがLEDアレイに出力されます。表示データは1-5桁目が同時にシフトされ，5桁目にセットしたパターンが順次出力されます。それぞれの桁のビット・パターンは2バイト単位で横1ラインを表し，漢字1文字を表示位置では上から下，フォント・バッファのアドレスでは低位から高位の順番の16ラインで表します。桁のラインは2バイトの最上位ビットが表示データの一番左側となるように表示します。フォント・バッファと表示位置の関係は次のとおりです（[ ] 内の数字はフォント・バッファ中の2バイト単位の配列の添字です）。

1桁目	2桁目	3桁目	4桁目	5桁目
[0]	[16]	[32]	[48]	[64]
[1]	[17]	[33]	[49]	[65]
⋮	⋮	⋮	⋮	⋮
[14]	[30]	[46]	[62]	[78]
[15]	[31]	[47]	[63]	[79]

← シフトする方向 ←

図4-7　表示バッファの内容とフォント・バッファ・データの例

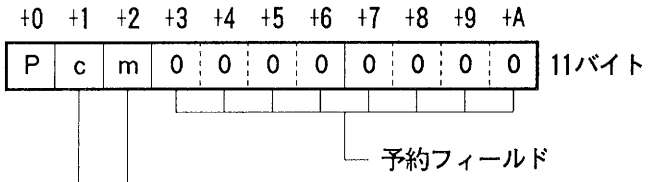
表示内容	配列の添字	データ
□■□□□□□□■□□□■□□□	[N×16+0]	4088H
□□■□□■■■■■■■■■■■	[N×16+1]	27FFH
□□□■□□□□■□□□■□□□	[N×16+2]	1088H
□□□□□□□□□□□□□□□□	[N×16+3]	0000H
□■□□□□■■■■■■■■■□	[N×16+4]	43FEH
□□■□□□■□□□■□□□■□	[N×16+5]	2222H
□□□■□□■□□□■□□□■□	[N×16+6]	1222H
□□□□□□■■■■■■■■■□	[N×16+7]	03FEH
□□□□□□□□□□■□□□□□	[N×16+8]	0020H
□□□□■□■■■■■■■■■□	[N×16+9]	0BFEH
□□□□■□□□□□□■□□□□	[N×16+10]	0820H
□□□■□■■■■■■■■■■■	[N×16+11]	17FFH
□□□■□□□□□□□■□□□□	[N×16+12]	1020H
□□■□□□□□□□■□■□□□	[N×16+13]	2050H
□□■□□□□□■■□□□■■□	[N×16+14]	218CH
□■□□□■■□□□□□□□■■	[N×16+15]	4603H

N：桁の番号（1-5）から1を引いた値

(9)通信データ・フォーマット

ホストとターゲット・システムの間の通信データ・フォーマットは次の5種類があります。

(a)コマンド・データ・フォーマット（ホスト→ターゲット）

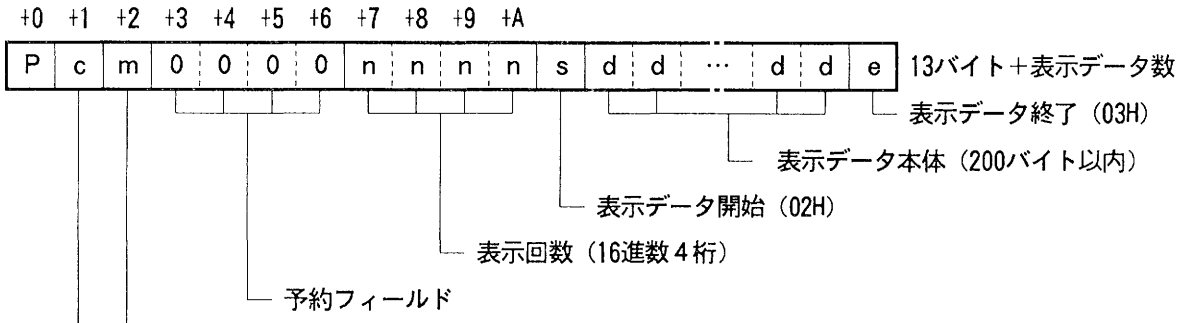


m	モード・フィールド
0	30H：時刻モード
1	31H：ニュース・モード
2	32H：広告モード
4	34H：緊急メッセージ・モード
8	38H：混在モード

c	コマンド・フィールド
0	30H：データ新規登録
1	31H：データ削除
2	32H：モード変更
4	34H：ディフォールト・モードへ変更
8	38H：登録メッセージ出力

保守／廃止

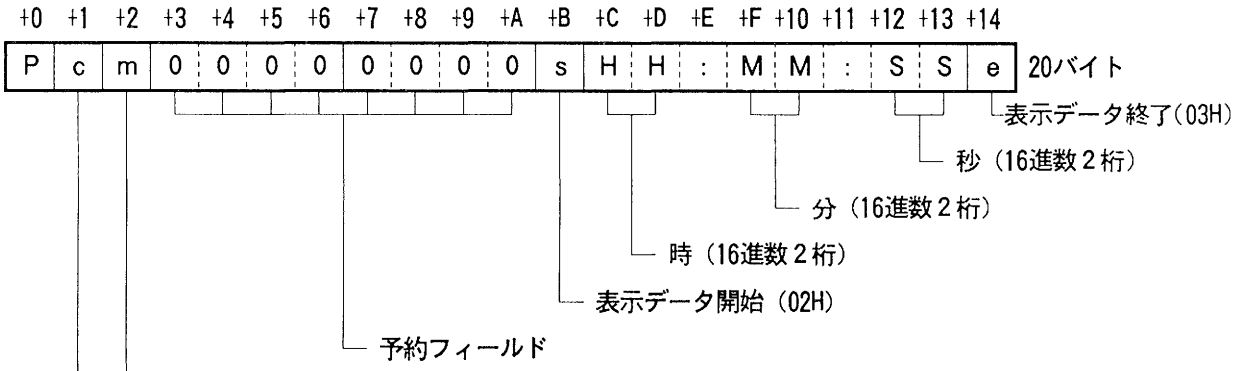
(b)表示データ・フォーマット（ホスト→ターゲット）



m	種別フィールド
1	31H：ニュース・データ
2	32H：広告データ
4	34H：緊急メッセージ・データ

c	コマンド・フィールド
0	30H：データ登録

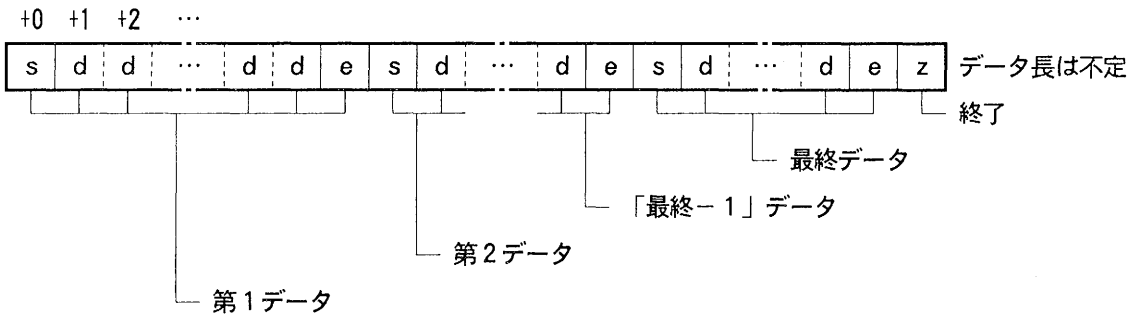
(c)時刻データ・フォーマット（ホスト→ターゲット）



m	種別フィールド
0	30H：時刻データ

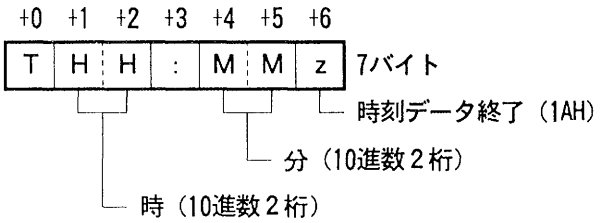
c	コマンド・フィールド
0	30H：データ登録

(d)表示データ出力・フォーマット（ターゲット→ホスト）



s	02H：メッセージ・データ開始
e	03H：メッセージ・データ終了
z	1AH：メッセージ送信終了

(e)現在時刻データ・フォーマット（ターゲット→ホスト）



4．4　同期通信のための資源

この節では，これまでに説明した内容で使用している同期通信用の資源について説明します。

アプリケーション・システムでは次の(１)-(11)に示すセマフォを２つ，イベント・フラグを１つ，メールボックスを８つ使用しています。これらの資源の生成は初期タスクの中で行っています。

(1)表示データ排他制御用セマフォ

ニュース，コマーシャル，緊急の３つの表示データ・メールボックスのアクセスに関して，タスク間での排他制御のために使用します。

(2)システム情報排他制御用セマフォ

システムの管理情報をアクセスする際の他タスクとの排他制御に使用します。このセマフォによって排他制御される情報は，表示モード，時刻データ，時刻更新情報の３つです。

(3)時刻表示タイミング・イベント・フラグ

時刻タスクが時刻を表示するタイミングをメッセージ・タスクに伝えます。このイベント・フラグで使用しているビットは第０ビットで，このビットが０の場合は時刻の表示が必要ないことを表し，１の場合は時刻表示のタイミングであることを表しています。

(4)コマンド・メールボックス

受信タスクが受信したコマンド・データをコントロール・タスクに伝えるために使用します。このメールボックスに送信するメッセージはフリー・コマンド・メールボックスから取得します。

(5)コマンド用フリー・メールボックス

コマンド・メールボックスで使う未使用メッセージを保持します。このメールボックスには最初に３つの未使用メッセージを用意します。

(6)ニュース・メールボックス

ニュース・データを保持します。ニュース用のメッセージは，フリー表示メッセージ・メールボックスから取得します。

(7)コマーシャル・メールボックス

コマーシャル・データを保持します。コマーシャル用のメッセージは，フリー表示メッセージ・メールボックスから取得します。

保守／廃止

(8)緊急メールボックス

緊急メッセージのデータを保持します。緊急用のメッセージはフリー表示メッセージ・メールボックスから取得します。

(9)表示データ用フリー・メールボックス

ニュース，コマーシャル，緊急の３つのメールボックスで使われる未使用メッセージを保持します。このメールボックスは20のフリー・メッセージを用意するので，ニュース，コマーシャル，緊急の３つの表示データの合計で20件の表示データを使用できます。表示データの最大数を変更するときは，このフリー・メールボックスに用意するフリー・メッセージの数を変更することで対応できます。

(10)LED表示メールボックス

LEDに表示する文字列データの受け渡しに使うメールボックスです。このメールボックスに送信された内容の文字列がフォントに変換されてLED表示のフォント・バッファに出力されます。このメールボックスに送信するメッセージはLED表示用フリー・メールボックスから取得します。

(11)LED表示用フリー・メールボックス

LED表示メールボックスで使われる未使用メッセージを保持します。

このメールボックスは１つのフリー・メッセージだけを用意して，前の表示が終了するまで次の表示動作を行わないように制御します。

表4－2　同期通信のための資源

種　類	名　称	機　能	関連タスク	生　成[注]
セマフォ	表示データ排他制御用セマフォ	ニュース，コマーシャル，緊急のアクセスに関する排他制御	受信タスク メッセージ・タスク	ID＝1 ｶｳﾝﾄ＝1 FIFO
	システム情報排他制御用セマフォ	表示モード，時刻データ，時刻更新情報に対する排他制御	受信タスク メッセージ・タスク 時刻タスク コントロール・タスク	ID＝2 ｶｳﾝﾄ＝1 FIFO
イベント・フラグ	時刻表示タイミング・イベント・フラグ	時刻表示のタイミングを示す	時刻タスク メッセージ・タスク	ID＝1
メール・ボックス	コマンド・メールボックス	ホストから送られてきたコマンドを保持する	受信タスク コントロール・タスク	ID＝5 FIFO／FIFO
	コマンド用フリー・メールボックス	コマンド・メールボックスで使用するフリー・メッセージを保持する	受信タスク コントロール・タスク	ID＝7 FIFO／FIFO
	ニュース・メールボックス	ニュース・データを保持する	受信タスク メッセージ・タスク	ID＝1 FIFO／FIFO
	コマーシャル・メールボックス	コマーシャル・データを保持する	受信タスク メッセージ・タスク	ID＝2 FIFO／FIFO
	緊急メールボックス	緊急表示データを保持する	受信タスク メッセージ・タスク	ID＝3 FIFO／FIFO
	表示データ用フリー・メールボックス	ニュース，コマーシャル，緊急の各データで使用するフリー・メッセージを保持する	受信タスク メッセージ・タスク	ID＝6 FIFO／FIFO
	LED表示メールボックス	LEDに表示する文字列データを受け渡しする	メッセージ・タスク シフト・タスク	ID＝4 FIFO／FIFO
	LED表示用フリー・メールボックス	LED表示フリー・メッセージを保持する	メッセージ・タスク シフト・タスク	ID＝8 FIFO／FIFO

注　FIFO／FIFOはタスク／メッセージ待ちのオプションを示します。

4.5　タスク，ハンドラ

このアプリケーション・システムでは次のようなタスクと割り込みハンドラを用いています。

起動の説明の中の時間は，事象が起きてからその処理が開始されるまでに許容される時間です。この時間を考慮して，タスクのプライオリティを決定します。タスクの一覧を表4-3に示します。

表4-3　タスク一覧

名　称	機　能	許容起床時間	プライオリティ	ID
初期タスク	システムの初期化	—	110	1
受信データ処理タスク	受信データの処理	約200ms	140	3
コントロール・タスク	コマンドの処理	約1000ms	150	5
時刻タスク	時刻の更新	約1000ms	160	6
メッセージ・タスク	出力メッセージの決定	約1000ms	170	7
シフト・タスク	出力メッセージのフォント変換	約30ms	130	4
LEDドライブ・タスク	フォント・データのLEDへの出力	約1ms	120	2

(1)初期タスク

機　能	システムの初期化
起　動	リセットによってＯＳから起動される

処　理	初期タスク()
	{
	ハードウエアの初期化();
	シリアル受信バッファの初期化();
	同期通信用資源の初期化();
	タスクの初期化();
	システム情報の初期化();
	タスクの終了と削除();
	}

(2)受信データ処理タスク

機　能	シリアル受信データの処理
起　動	割り込みハンドラからの起床要求から約200ms以内

処　理	受信データ処理タスク()
	{
	while (1) {
	起動待ち();　　);
	受信バッファ処理();
	}
	}

(3)コントロール・タスク

機　能	コマンドの処理
起　動	コントロール・メールボックスからの受信完了から約1000ms以内

処　理	コントロール・タスク()
	{
	マーキング・メッセージの取得();
	while (1) {
	コマンド・メールボックスから受信できるまで待つ();
	switch (受信データの種類) {
	case 削除コマンド:
	指定された表示データをすべて削除する();
	break;
	case モード変更コマンド:
	指定されたモードにシステム情報を変更する();
	break;
	case 表示データ送信コマンド:
	指定されたモードの表示データを送信する();
	break;
	}
	コマンド・メールボックスから受信したメッセージを
	フリー・メールボックスに返却する();
	}
	}

(4)時刻タスク

機　能	時刻表示データの生成
起　動	60000ms間隔の周期起床要求の発生から約1000ms以内

処　理	時刻タスク()
	{
	　自タスクに対して1分間隔で起床周期を設定する();
	　表示タイミング・イベント・フラグをクリアする();
	　while (1) {
	　　起床されるまで待つ();
	　　if (システム時刻が更新された？) {
	　　　周期起床を取り消す;
	　　　起床要求を取り消す;
	　　　自タスクに対して1分間隔で周期起床を設定する();
	　　} else {
	　　　システム時刻を1分間進める();
	　　}
	　　if (時刻モードか混在モードのどちらかであるか？) {
	　　　表示タイミング用イベント・フラグにセットする();
	　　}
	　}
	}

(5)メッセージ・タスク

機　能	表示メッセージの管理
起　動	LED表示フリー・メールボックスからのメッセージ取得から約1000ms以内

処　理	メッセージ・タスク()
	{
	表示データ用フリー・メールボックスからフリー・メッセージが
	受信できるまで待つ();
	緊急メッセージがあれば表示する();
	if (緊急メッセージがなかった？) {
	switch (現在の表示モード) {
	case 混在モード:
	時刻表示タイミングであれば時刻を表示する();
	if (時刻表示タイミングではない？) {
	ニュースの表示の番ならニュースを表示する();
	コマーシャル表示の番ならコマーシャルを
	表示する();
	}
	break;
	case コマーシャル・モード:
	コマーシャル・データがあれば表示する();
	break;
	case ニュース・モード:
	ニュース・データがあれば表示する();
	break;
	case 緊急モード:
	フリー・メッセージを返却する;
	break;
	case 時刻モード:
	時刻タイミングであれば時刻を表示する();
	if (時刻表示タイミングでなかった？) {
	フリー・メッセージを返却する;
	}
	break;
	}
	}
	}

保守／廃止

(6)シフト・タスク

機　能	LEDアレイへの表示データのセットとシフト
起　動	LEDメールボックスからの受信完了から約30ms以内

処　理	シフト・タスク()
	{
	LED表示用バッファの初期化;
	LED表示用バッファのアドレスをメッセージとして送信する; /* RX136だけ */
	初期メッセージをLEDに表示する;
	while(1) {
	LEDメールボックスからLED表示メッセージを永久待ちで受け取る;
	LED表示メッセージをLEDに表示する;
	LED表示用フリー・メールボックスにLED表示メッセージを返却する;
	LEDの表示をクリアする;
	}
	}

(7)LEDドライブ・タスク

機　能	ダイナミック点灯方式によるフォント・バッファ内データのLEDへの出力
起　動	約1ms間隔での周期起床

処　理	LEDドライブ・タスク()
	{
	LED表示バッファ・アドレスのメールボックスから受信する;
	LED表示バッファのアドレスをアクセス用のポインタに代入する;
	受信したメールボックスをメモリ・プールに解放する;
	LED表示バッファ用メールボックスを削除する;
	} RX136だけ
	パラレル・インタフェース・ユニット（PIU）を初期化する();
	while (1) {
	横1列分の表示データをLEDアレイにセットする;
	表示する列番号をLEDアレイにセットする;
	1ms待つ();
	}
	}

(8)受信割り込みハンドラ

機　能	シリアル受信データの取り込み
起　動	シリアル受信割り込み発生から約500μs以内

処　理	受信割り込みハンドラ()
	{
	　　ホストから受信した1文字を取り出す;
	　　if (リング・バッファが256文字？) {
	　　　　何もせずに割り込みを終了();
	　　}
	　　読み出した文字をリング・バッファに書き込む;
	　　受信データ処理タスクを起床して割り込みを終了する();
	}

第５章　RX136用ソフトウエア開発

この章ではRX136を用いたソフトウエアの具体的な開発方法について説明します。

５．１　開発環境

RX136を用いたアプリケーションの開発環境として表５－１のソフトウエア・ツールをMS-DOSの環境下で用います。

表５－１　RX136用ソフトウエア開発ツール一覧

ツール	製品名	ファイル名	機　能
リアルタイムOS	RX136 Ver1.2	RX136.HEX	マルチタスク環境の提供
コンフィギュレータ	(RX136 Ver1.2に付属)	CF70136.EXE	システム情報テーブルの作成
Cコンパイラ	MS-C Ver6.0	CL.EXE	タスクのコンパイル
MAKEコマンド	(MS-C Ver6.0に付属)	NMAKE.EXE	コマンド実行の効率化
アセンブラ	MASM Ver5.1	MASM.EXE	システム情報テーブルのアセンブル
リンカ	MS-C Ver6.0	LINK.EXE	プログラムのリンク
ロケータ	LC70116 Ver1.0	LOCATE.EXE	オブジェクト・ファイルの物理アドレスへの配置とHEXファイルへの変換

５．２　MS-Cによる開発の方法

５．２．１　注意事項

MS-CによってRX136のアプリケーションを作成する際には次の点に注意してください。

(１)初期値あり変数の初期化

初期値あり変数（外部変数やスタティック変数で初期値があるもの）の初期化はRX136では行いません。

初期タスクの先頭をLC70116の提供するスタートアップ・コード（MSC60.ASMを修正して使用）に設定して，ここから本来の初期タスク本体を呼び出します。ただし，スタック・セグメントとスタック・ポインタの値はRX136が設定するので，スタートアップ・コード中からスタックの設定をしている部分を削除してください。

(2)マルチタスクで使えないランタイム・ライブラリの対策

LC70116にはMS-Cのランタイム・ライブラリの中からROM化できるものを取り出し，ライブラリを作る機能があります。しかし，ROM化できるライブラリ・ルーチンの中には，マルチタスクで使えない（リエントラントでない）ものがあります。

マルチタスクに対応できないライブラリ・ルーチンは，メモリ管理（mallocなど），I/Oが必要な関数（fgetcなど），浮動小数点演算全般，sprintf関数などです。

このうち，このアプリケーション・システムでは，sprintf関数とvsprintf関数の機能を限定した独自のルーチンを作成して使用しています。詳細は**付録Ａ　RX136用コーディング・リスト**を参照してください。

(3)RX136インタフェース・ライブラリの修正

RX136には標準のインタフェース・ライブラリのアセンブラ・ソースが添付されていますが，これらは16ビットＶシリーズ用のニモニックで記述されています。MS-Cで使用する場合は，これらのアセンブラ・ソースをMASMのニモニックに修正してライブラリを再構築してください。

ただし，この方法だけではret_int，iret_wupがMS-Cの_interrupt記述子を使ったＣ言語での割り込みハンドラの記述には対応できません。

このアプリケーション・システムでは，MS-Cの_interrupt記述子によって生成される割り込みハンドラでも使えるret_intとiret_wupのインタフェース・ライブラリを作成し，割り込みハンドラをすべてＣ言語で記述しています。

詳細は**付録Ａ　RX136用コーディング・リスト**を参照してください。

(4)拡張メモリに配置したタスクの制限事項

異なるタスク・グループIDのタスクの間での変数の共有やサブルーチンの共有はできません。そのため，タスク・グループごとに違う外部変数のためのデータ・セグメントを設定し，インタフェース・ライブラリを含むサブルーチンもそれぞれ用意する必要があります。

このアプリケーション・システムでは外部変数を共有するためにタスク間通信を行い，変数のポインタを渡して使用する方法を用いています。

詳細は**付録Ａ　RX136用コーディング・リスト**を参照してください。

５．２．２　開発の手順

表５－１に示したソフトウエア・ツールを用いた開発の概要を図５－１に示します。タスク・オブジェクトとスタートアップ・ルーチン・オブジェクトの２種類を作成します。

図５－１　RX136を使用したアプリケーションの開発概要（１/２）

（ａ）タスク・オブジェクトの開発

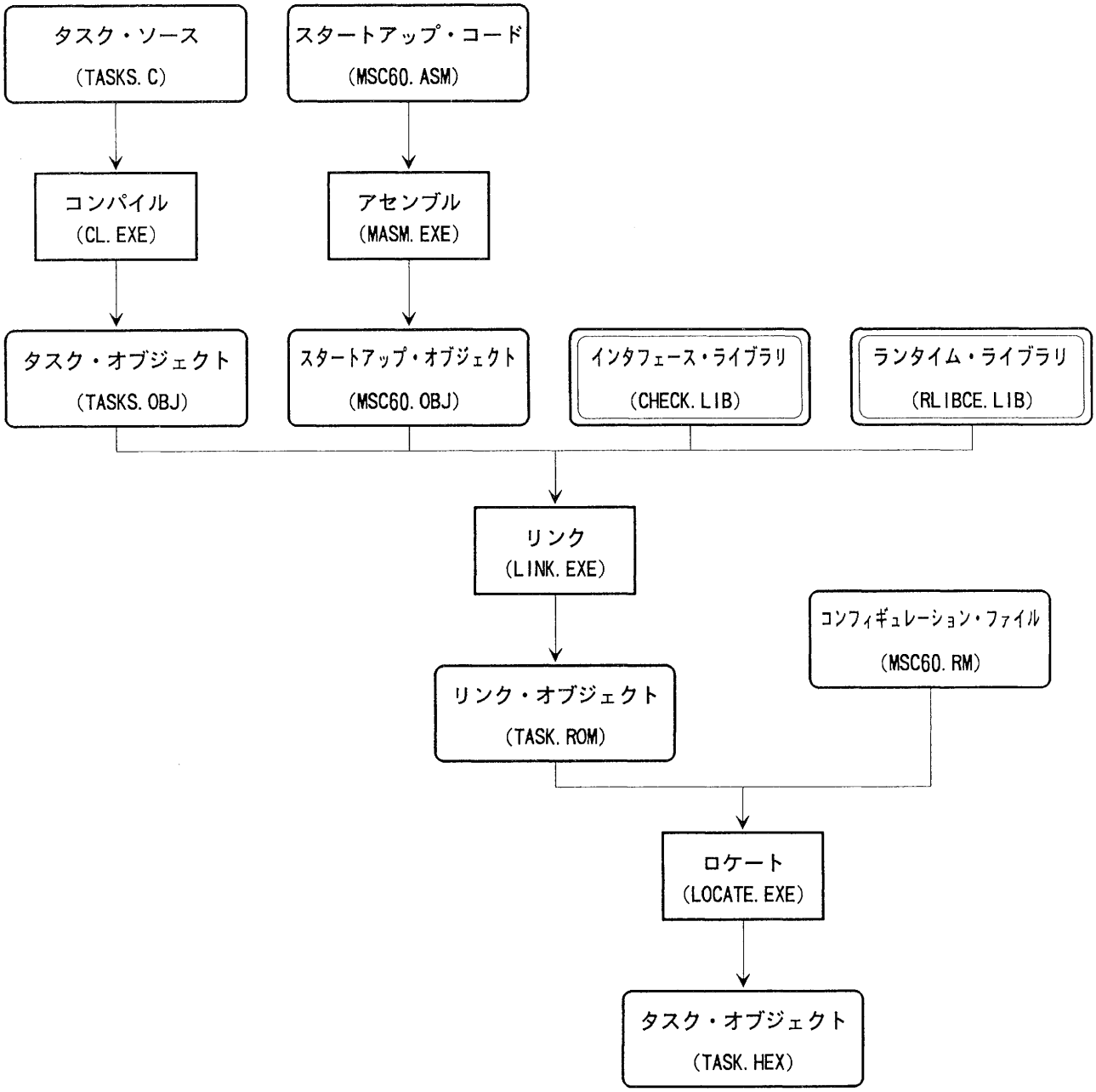
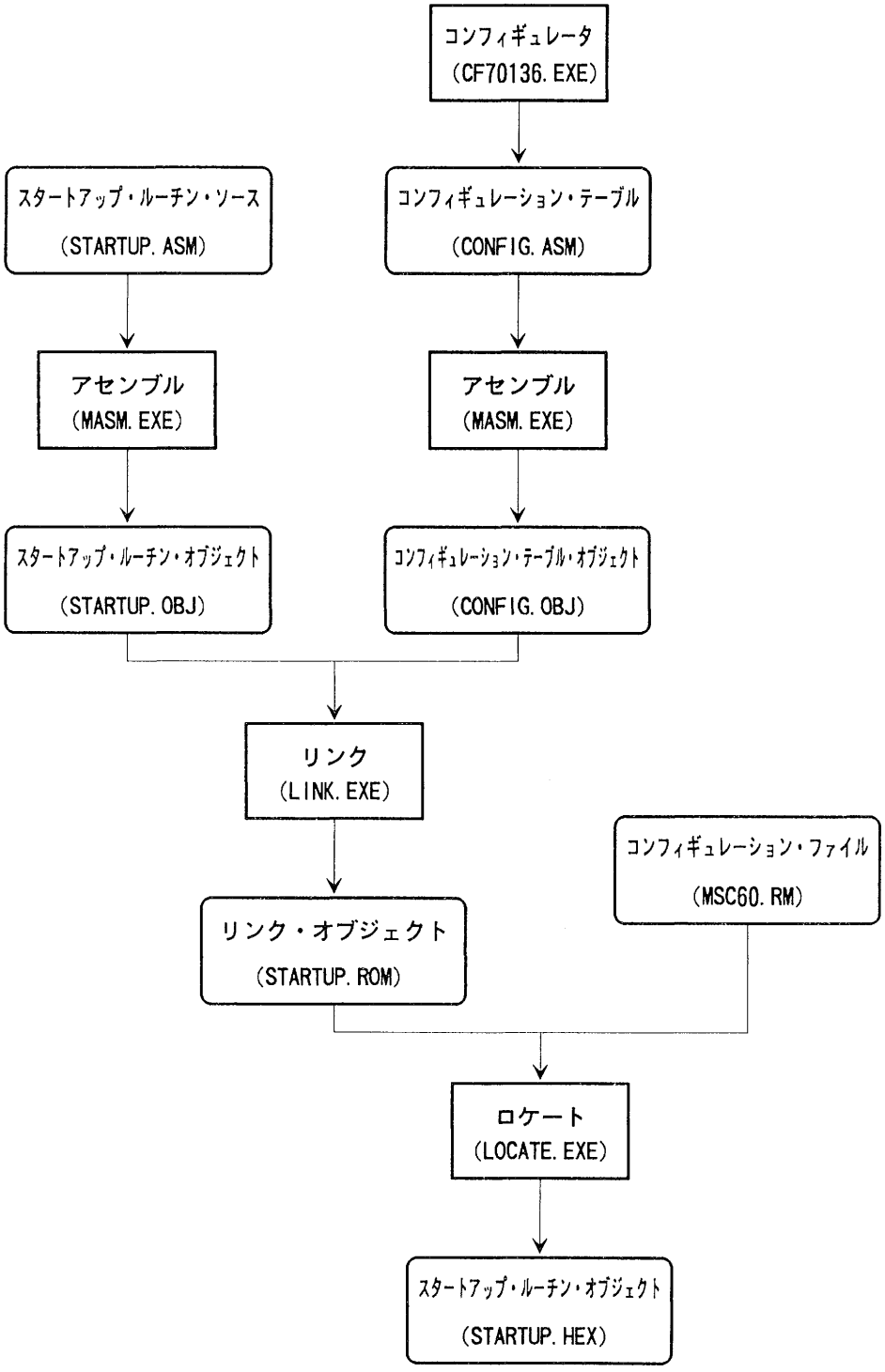


図５－１　RX136を使用したアプリケーションの開発概要（２/２）

（ｂ）スタートアップ・ルーチン・オブジェクトの開発



５．３　ファイルの構成

このアプリケーション・システムで使用するファイルの構成を表５－２に示します。

最終的に生成するオブジェクトは次の４つのHEXファイルです。

- MAIN.HEX（１Mバイトに配置するタスク）
- SHIFT.HEX（シフト・タスク）
- LED.HEX（LEDドライブ・タスク）
- STARTUP.HEX（スタートアップ・ルーチン）

これらのファイルに加えて，RX136.HEX（RX136で提供されるリアルタイムOS本体）を含めてROM化することにより，ターゲットを動かすことができます。このうち，SHIFT.HEXとLED.HEXは拡張メモリ上のROMに配置します。

表５－２　ファイル構成（RX136用）（１/２）

分　類	ファイル名	内　容
C言語タスク・ソース	MAIN.C	初期タスク
	CTRL.C	コントロール・タスク
	LED.C	LEDドライブ・タスク
	MESG.C	メッセージ・タスク
	RECV.C	受信データ処理タスク
	SHIFT.C	シフト・タスク
	TIME.C	時刻タスク
C言語サブルーチン	FUNC.C	タスク用サブルーチン
	INIT.C	CPU初期化サブルーチン
	SPRINTF.C	sprintf関数，vsprintf関数
	STRFONT.C	フォント・データ取り出し関数
C言語割り込みハンドラ	INTHAND.C	受信完了割り込みハンドラ
アセンブラ・ソース・ファイル	MSC60_1.ASM[注]	1Mバイト空間に配置されるタスクのスタートアップ・コード
	MSC60_2.ASM[注]	シフト・タスクのスタートアップ・コード
	MSC60_3.ASM[注]	LEDドライブ・タスクのスタートアップ・コード
	CRETINT.ASM	MS-C用ret_intインタフェース・ルーチン
	CIRETWUP.ASM	MS-C用iret_wupインタフェース・ルーチン
	STARTUP.ASM	スタートアップ・ルーチンのソース・ファイル（MASM用）
	CONFIG.ASM	コンフィギュレーション・テーブル（コンフィギュレータの出力）

注　LC70116に添付されるMSC60.ASMを修正して使用してください（**5.2(1)初期値あり変数の初期化**参照）。

表５－２　ファイル構成（RX136用）（２/２）

分　類	ファイル名	内　容
ロケート・コマンド	MSC60_1.RM	１Mバイト空間に配置されるタスクのロケート情報
	MSC60_2.RM	シフト・タスクのロケート情報
	MSC60_3.RM	LEDドライブ・タスクのロケート情報
	STARTUP.RM	スタートアップ・ルーチンのロケート情報
インクルード・ファイル	MSC60.INC	スタートアップ・コード用インクルード・ファイル（LC70116に添付）
	STARTUP.INC	
	INITINFO.TBL	スタートアップ・ルーチン用インクルード・ファイル（MASM用）
	CONSTANT.TBL	
	LED.H	アプリケーション用インクルード・ファイル
	RX.H	システム・コール用インクルード・ファイル
NMAKE用指定ファイル	MAKESTA	NMAKEによるコマンド実行指定ファイル(スタートアップ部)
	MAKEFILE	NMAKEによるコマンド実行のための指定ファイル(タスク部)

５．４　LC70116（ロケータ）

LC70116によってプログラムの配置と最終オブジェクト・ファイルへの変換を行います。LC70116のパッケージの中には，MS-Cの標準ライブラリの中からROM化可能なルーチンだけを取り出す機能とMS-C用のスタートアップ・コード（MSC60.ASM）が提供されています。スタートアップ・コードは本来リセット時のエントリとして用いるもので，LC70116によって設定された初期値のある変数への初期値のセットなどを行います。

一方，RX136にもスタートアップ・ルーチンが提供されていますが，リセット時のルーチンにはRX136のスタートアップ・ルーチンを使用し，初期タスクの先頭アドレスをLC70116の提供するスタートアップ・コードとして使用します。

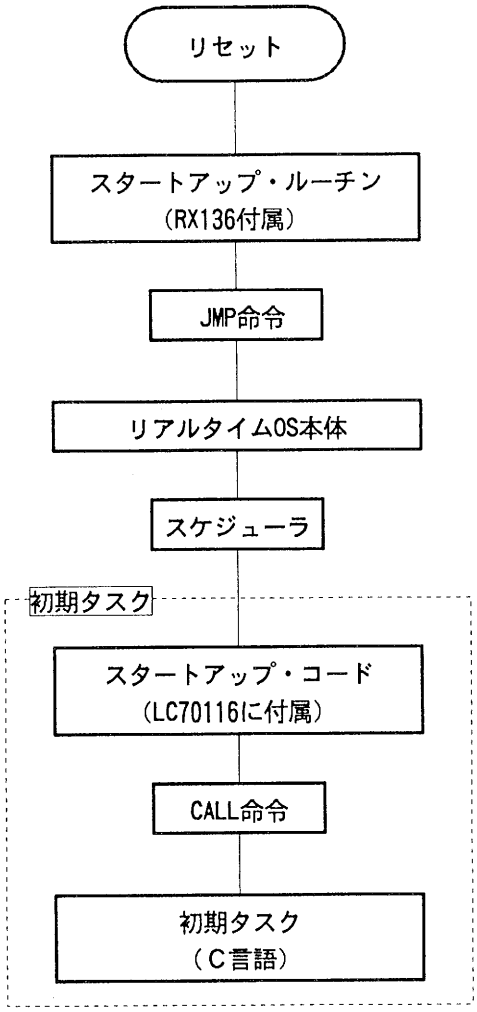
このリセット時のプログラムの実行の様子を図５－２に示します。

このようにLC70116のスタートアップ・コード（アセンブラ・ソース）をタスクの先頭として使用するため，このファイルをコピーしてスタックの設定をはずして使用してください。スタックはRX136が自動的に設定します。

また，拡張メモリにタスクを配置する場合は，各タスク・グループで最初に実行するタスクに，このスタートアップ・コードを付けてください（**５．６　拡張メモリのタスク**参照）。

なお，タスク・グループ#0用のROM内の配置は図５－４を参照してください。

図５－２　リセット時の動作



５．５　拡張メモリ・データのアクセス

システムでは，漢字フォント・データと半角フォント・データを読み出すために，メモリ・プール#16を使用しています。半角フォント・データは720000H番地から，漢字フォント・データは740000H番地からそれぞれ配置しています。メモリ・マップを図５－３に示します。

図５－３　RX136使用時のメモリ・マップ

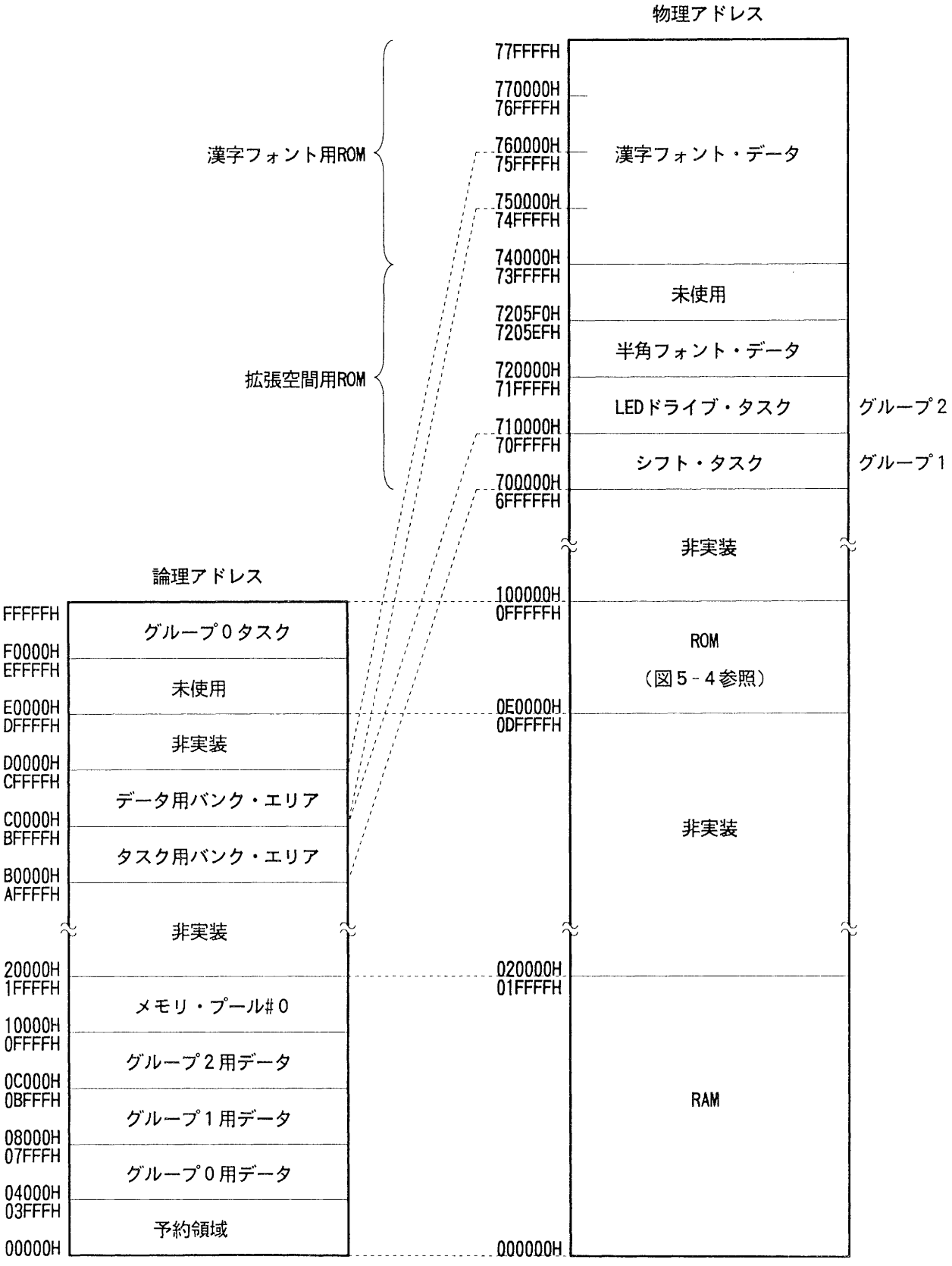
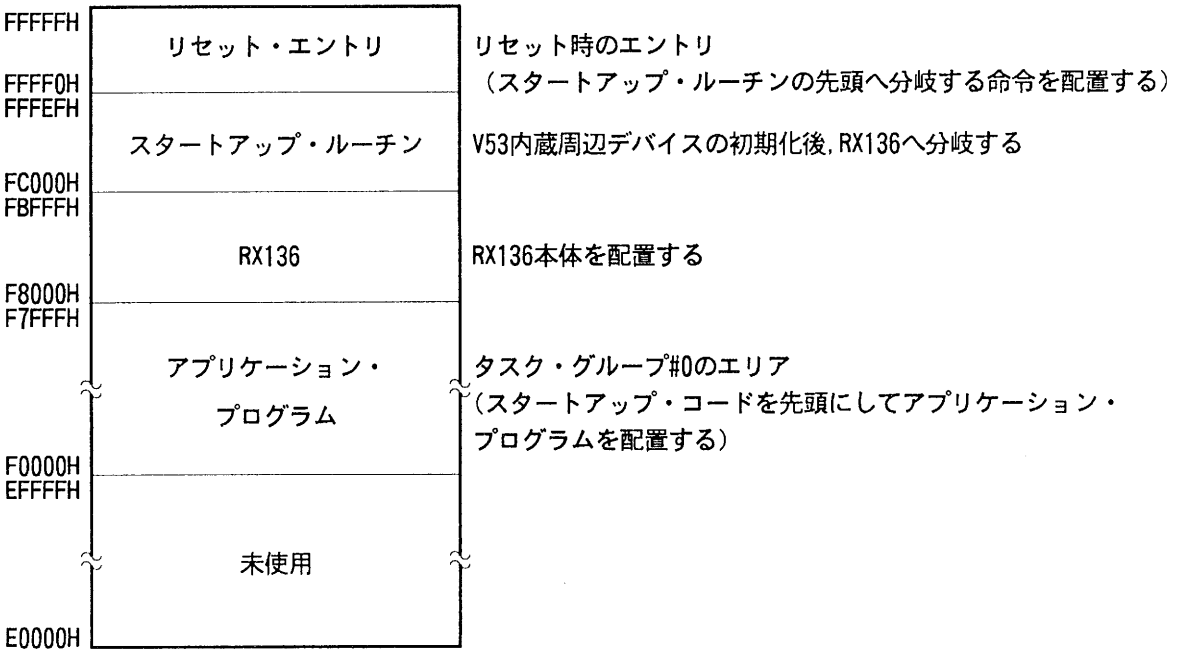


図５－４　ROMの内容



アプリケーション・システムが行う拡張メモリのデータ・アクセスの基本形は次のとおりです。

```
get_blk(..);  /*  メモリ・プール#16から使用するサイズのメモリ・ブロックを確保する  */
map_blk(..);  /*  確保したメモリ・ブロックを16Mバイト空間の必要なアドレスにマッピングする  */
処理；        /*  マッピングしたエリアをアクセスする  */
map_blk(..);  /*  必要であればマッピングを繰り返す  */
処理；
rel_blk(..);  /*  不要になったメモリ・ブロックを解放する  */
```

以上のような手順を使うことにより，マッピング領域（メモリ・プール#16）のサイズが限られていても，複数のタスクが拡張メモリを使用できます。

このアプリケーション・システムでは，拡張メモリを使用するタスクは１つだけですが，仮に別のタスクが同時に拡張メモリを使用しようとしてメモリ・プール#16の残りがなくなった場合は，あとからget_blkを発行したタスクが待ち状態となり，最初に使用を開始したタスクが処理を終わってrel_blkした時点で待ち状態が解除され，排他的な制御が行われます。そのため，複数のタスクが拡張メモリのデータをアクセスする可能性がある場合には，あらかじめ，get_blkによって確保されるメモリ・ブロックの合計を満たすようにメモリ・プール#16のサイズを設定するか，上記のようなget_blk-rel_blkによる排他制御機構を使ってください。この際，メモリ・プール#16からブロックを獲得している間にセマフォなどの待ち状態になると，拡張メモリのアクセス権を握ったまま待ち状態になるので，デッドロックが起きないように注意してください。

５．６　拡張メモリのタスク

MS-CとLC70116の組み合わせによって拡張メモリにタスクを配置する場合，**５．２　MS-Cによる開発の方法**で述べたように拡張メモリのタスクとその他のタスクの間でのシンボル参照はできません。

このアプリケーション・システムでは，シフト・タスクとLEDドライブ・タスクの２つをそれぞれ別の拡張メモリに配置していますが，その方法について説明します。

拡張メモリに配置したタスクのためのバンク・エリアとしてはB0000H番地から64Ｋバイトを使用しています。このエリアに対して，シフト・タスクはグループIDが１で700000H番地から，LEDドライブ・タスクはグループIDが２で710000H番地に配置しています。これらの拡張タスクはLC70116が出力するHEXファイル中でB0000H番地に配置されているので，拡張メモリの該当するアドレスになるようにROMに書き込んでください。

また，これらのタスクの外部変数のための領域として，シフト・タスクには8000H番地から１Ｋバイト，LEDドライブ・タスクにはC000H番地から１Ｋバイトを用意してください。また，グループIDが０の１Ｍバイト空間内のタスクのために4000H番地から１Ｋバイトを用意しています。これらのタスク・グループとメモリの配置の関係を表５－３に示します（メモリ・マップについては**図５－３**を参照してください）。

注意　LC70116によって配置したあと，それぞれ出力する配置情報ファイルを見て，３つのエリアが重なっていないことを確認してください。

外部変数が３つのエリアにあるため，初期値あり変数の設定を３回行います。グループIDが０のエリアは，初期タスクが，グループIDが２のデータはシフト・タスクが，グループIDが３のデータはLEDドライブ・タスクがそれぞれ設定します。

シフト・タスクとLEDドライブ・タスクの間にはLEDアレイに表示するビット・データのバッファを介してデータの転送をするため，このバッファのアドレスを双方で参照する必要があります。このアプリケーション・システムではシフト・タスクがバッファ領域を確保しており，メールボックスを使ってバッファのアドレスをLEDタスクに渡しています。

LEDタスクは受け取ったアドレスをポインタ変数にセットして実際のバッファをアクセスします。

表５－３　タスク・グループとメモリの配置

タスク・グループID	プログラム・アドレス	データ・アドレス	グループを構成するタスク
0	0F0000H	4000H	初期タスク，受信タスク， 時刻タスク，メッセージ・タスク， コントロール・タスク
1	700000H	8000H	シフト・タスク
2	710000H	C000H	LEDドライブ・タスク

５．７　Ｃ言語による割り込みハンドラの記述

この節ではMS-Cを使用し，Ｃ言語で割り込みハンドラを記述する方法について説明します。

MS-Cでは_interrupt記述子を用いることにより，割り込みハンドラをＣ言語レベルで記述できます。

しかし，RX136で提供されるインタフェース・ライブラリはSP70116-Iに対応する形で構成されているため，割り込み処理の先頭でセーブする内容がMS-CとSP70116-Iで異なり，標準のインタフェース・ライブラリ・ソースをMASM用に修正しただけでは，割り込みハンドラで使用できません。使用できないのは，ret_intとiret_wupの２つのシステム・コールです。

このアプリケーション・システムでは，MS-Cに対応できるように修正したこれらのインタフェース・ルーチンを作成して割り込みハンドラをすべてＣ言語で記述できるようにしてあります。

Ｃ言語による割り込みハンドラの基本の形は次のようになります。また，ret_intとiret_wupのシステム・コールはサブルーチンでは発行できません。

```
_interrupt void inthandler( void )
{
        処理
          ⋮
        ret_int();      /* またはiret_wup(タスクのアクセス・アドレス); */
}
```

保守／廃止

第６章　RX423用ソフトウエア開発

この章ではRX423を用いたソフトウエアの具体的な開発方法について説明します。

６．１　開発環境

RX423を用いたアプリケーションの開発環境として表６－１のソフトウエア・ツールをMS-DOSの環境下で用います。

表６－１　RX423用ソフトウエア開発ツール一覧

ツール	製品名	ファイル名	機　能
リアルタイムOS	RX423 Ver1.0	RX423.HEX	マルチタスク環境の提供
コンフィギュレータ	（RX423 Ver1.0に付属）	CF70423.EXE	システム情報テーブルの作成
Ｃコンパイラ	SP70116-I リリース5.1	CV55PI.EXE	タスクのコンパイル
MAKEコマンド	UNIXのmakeに相当するコマンド（またはMS-Cの NMAKE.EXEなど）	MAKE.EXE	コマンド実行の効率化
アセンブラ	SP70116-I リリース5.1	ASMV55PI.EXE	システム情報テーブルのアセンブル
リンキング・ロケータ		LLINK.EXE	プログラムのリンク
フォーマッタ		FORM.EXE	オブジェクトのHEXファイルへの変換

６．２　SP70116-Iによる開発の方法

６．２．１　注意事項

SP70116-IによってRX423のアプリケーションを作成する際には次の点に注意してください。

（１）初期値あり変数の初期化

初期値あり変数（外部変数やスタティック変数で初期値があるもの）の初期化はRX423では行いません。

SP70116-Iの標準ランタイム・ライブラリで提供されているrcopy関数を初期タスクの先頭で呼び出すようにします。詳細は**付録Ｂ　RX423用コーディング・リスト**を参照してください。

(2)拡張アドレスの使用とRX423

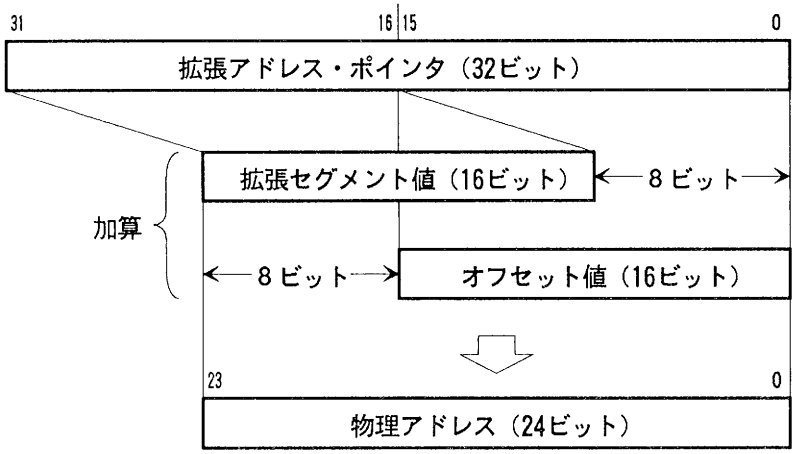
Ｃコンパイラ（CV55PI.EXE）を用いてコンパイルする際には，コンパイルされたコードが拡張アドレスを使うか，通常アドレスだけを使うかのコンパイル・スイッチ（-nx）を付けてください。-nxを付けないでコンパイルした場合，拡張アドレスを使用する指定となり，変数へのポインタはすべて16Mバイト空間を示すことができます。

この拡張アドレス使用の際に生成されるポインタは，図６－１に示すような「拡張セグメント：オフセット」の形となります。拡張セグメントはV55PIのDS2，DS3に相当する拡張セグメント・レジスタの形式で，オフセットに対して８ビット分上位にシフトされた形です。

拡張アドレス用にコンパイルしたプログラムをタスクとして動かす場合には，RX423も拡張アドレス版を使用します。このとき，システム・コールによってポインタを渡すときと，アドレスを返してくるときのポインタはすべて「拡張セグメント：オフセット」の形式となります。

この場合，get_blk，rcv_msgシステム・コールによってリターン・パラメータとなるメモリ・ブロックのアクセス・アドレスは４バイトの拡張アドレス・ポインタとなります。

図６－１　拡張アドレス・ポインタの形式



(3)データ配置の注意事項

拡張アドレスを使用する場合，CV55は，DS0によって示される「dataクラス」の中に次の２つの外部変数（各２バイト）があることを前提としたコード生成をします。

__DSX：「dataクラス」が配置されているDS0の値を拡張セグメント・レジスタの形式に変換した値が入ります。この変数は，DS0に配置された外部変数を指し示す拡張アドレス・ポインタを生成するときに参照されます。
dataクラスは256バイト境界に配置してください。

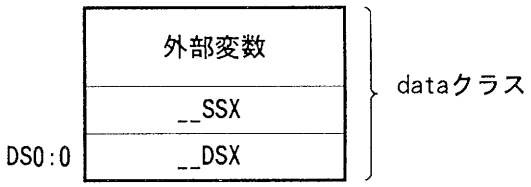
__SSX：スタック・セグメント・レジスタ（SS）の値を拡張セグメント・レジスタの形式に変換した値が入ります。この変数は，スタック上のローカル変数を指す拡張アドレス・ポインタを生成する際に参照されます。

これらの変数はマルチタスクの環境下では，タスク固有となるため，RX423はタスク・スイッチの際に，これらの変数を書き換えます。

__DSXと__SSXはdataクラスの先頭に配置してください。

なお，タスクの生成の際に指定するDS0の値はdataクラスの先頭アドレスを示すセグメント値となるようにします。また，アプリケーション・システムでは外部変数をタスク間で共有するため，すべてのタスクのDS0が同じdataクラスを指します。

図６－２　RX423を使うデータの配置



(4)ランタイム・ライブラリの構築方法について

アプリケーション・システムで使用するＣ言語ルーチンはすべてラージ・モデルでコンパイルする必要があります。

ランタイム・ライブラリを構築する前に，ランタイム・ライブラリのソース・ディレクトリ（RTLIBS¥SRC）にあるXMAIN.A20をエディットして，RX423で使用しないランタイム・ライブラリ中のスタック領域を削除します（" db STACKSIZE dup(?)"という行を削除）。

ランタイム・ライブラリを構築するには次のようにしてください。

```
mklib  cv55s  lm  normal  ic
```

なお，mklibコマンドの詳細については**RA70116-I，SP70116-I　ユーザーズ・マニュアル　操作編**を参照してください。

６．２．２　開発の手順

表６－１に示したソフトウエア・ツールを用いた開発の概要を図６－３に示します。タスク・オブジェクトとスタートアップ・ルーチン・オブジェクトの２種類を作成します。

図６－３　RX423を使用したアプリケーションの開発概要（１/２）

（ａ）タスク・オブジェクトの開発

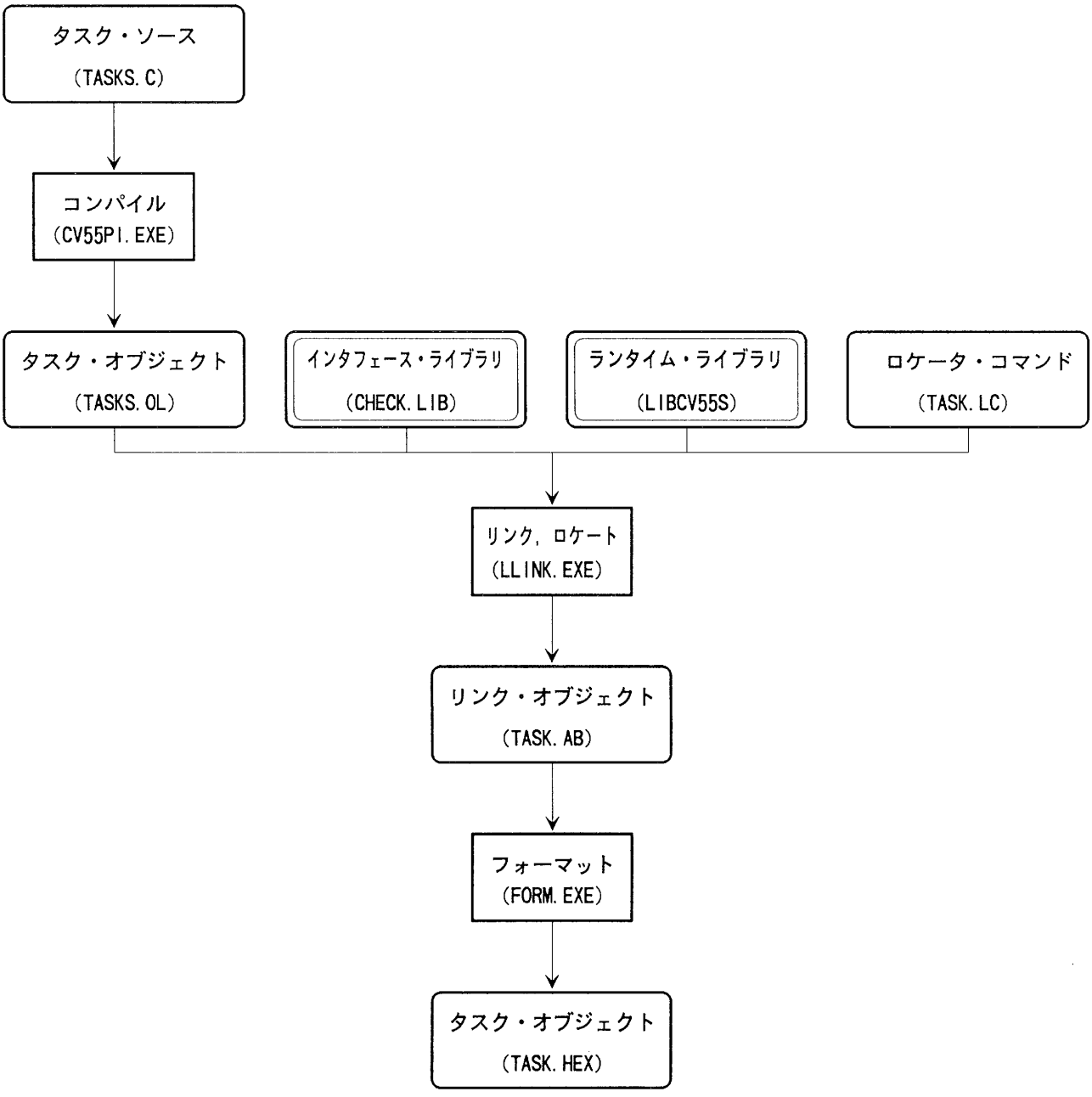
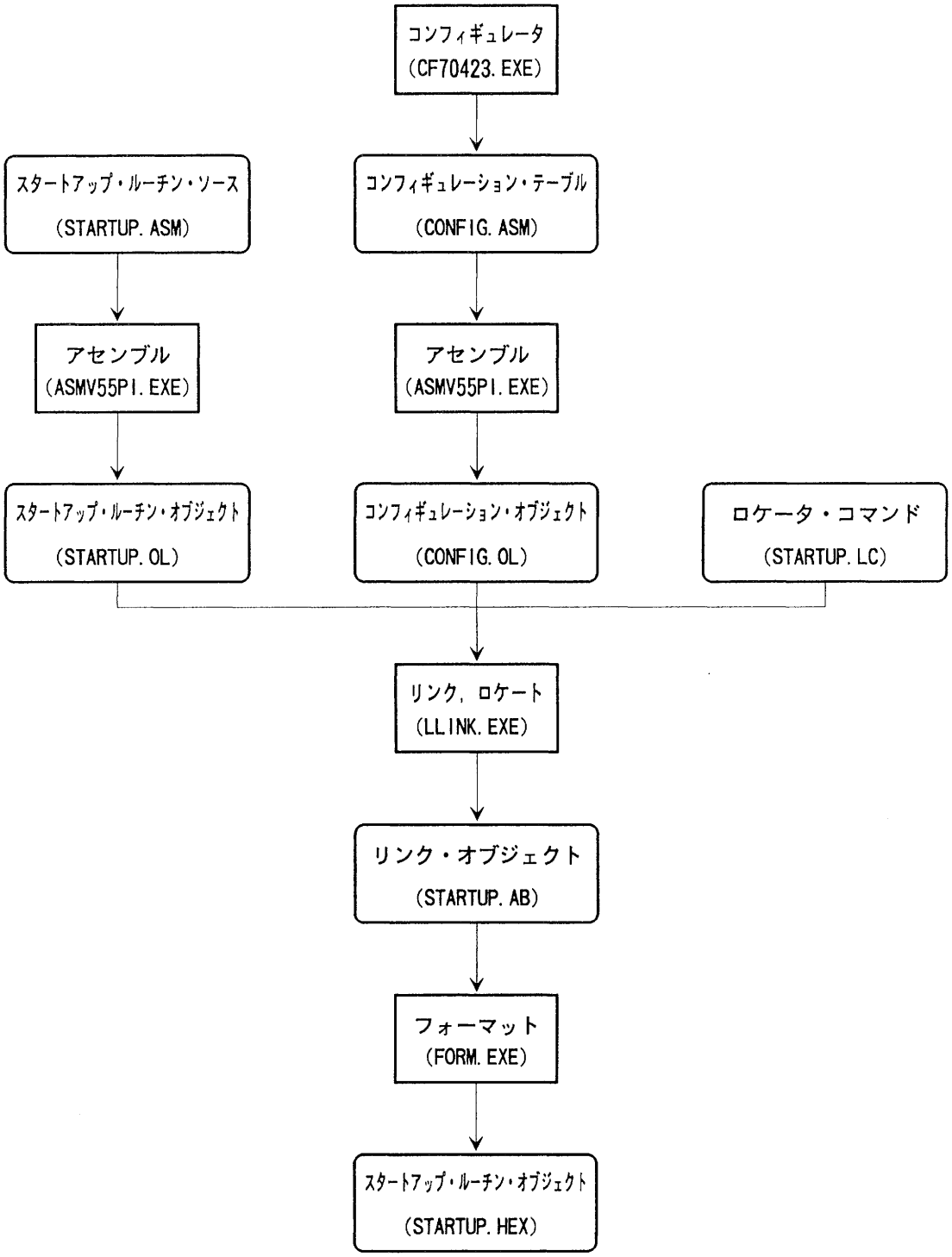


図６－３　RX423を使用したアプリケーションの開発概要（２/２）

（ｂ）スタートアップ・ルーチン・オブジェクトの開発



６．３　ファイルの構成

アプリケーション・システムで使用するファイルの構成を表６－２に示します。

最終的に生成するオブジェクトは，MAIN.HEX（タスクを構成するファイル）とSTARTUP.HEX（スタートアップ・ルーチン）の２つのHEXファイルです。

これらのファイルに加えて，RX423.HEX（RX423で提供されるリアルタイムOS本体）を含めてROM化することにより，ターゲットを動かすことができます。

表６－２　ファイル構成（RX423用）

分　類	ファイル名	内　容
Ｃ言語タスク・ソース	MAIN.C	初期タスク
	CTRL.C	コントロール・タスク
	LED.C	LEDドライブ・タスク
	MESG.C	メッセージ・タスク
	RECV.C	受信データ処理タスク
	SHIFT.C	シフト・タスク
	TIME.C	時刻タスク
Ｃ言語サブルーチン	FUNC.C	タスク用サブルーチン
	INIT.C	CPU初期化サブルーチン
	STRFONT.C	フォント・データ取り出し関数
Ｃ言語割り込みハンドラ	INTHAND.C	受信完了割り込みハンドラ
アセンブラ・ソース・ファイル	STARTUP.ASM	スタートアップ・ルーチン（RX423に添付）
	CONFIG.ASM	コンフィギュレーション・テーブル（コンフィギュレータの出力）
	RESET.ASM	リセット・エントリの分岐処理
ロケート・コマンド	TASK.LC	タスクに対する配置情報
	STARTUP.LC	スタートアップ・ルーチンの配置情報
ロケート・ファイル	OBJLIST	リンクするオブジェクト・ファイルを指定する
インクルード・ファイル	INITINFO.TBL	スタートアップ・ルーチン用インクルード・ファイル（RX423に添付）
	CONSTANT.TBL	
	LED.H	アプリケーション用インクルード・ファイル
	RX.H	システム・コール用インクルード・ファイル
MAKE用指定ファイル	MAKESTA	スタートアップ・ルーチン生成用のMAKE実行指定ファイル
	MAKEFILE	タスク生成のためのMAKE実行指定ファイル

６．４　拡張メモリ・データのアクセス

システムでは，漢字フォント・データと半角フォント・データを読み出すために，メモリ・プール#16を使用しています。半角フォント・データは720000H番地から，漢字フォント・データは740000H番地から，それぞれ配置しています。メモリ・マップを図６－４に示します。

SP70116-Iにより拡張セグメント・アドレスを使用することで，拡張メモリは１Ｍバイト空間のデータと同じようにアクセスできます。

図６－４　RX423使用時のメモリ・マップ

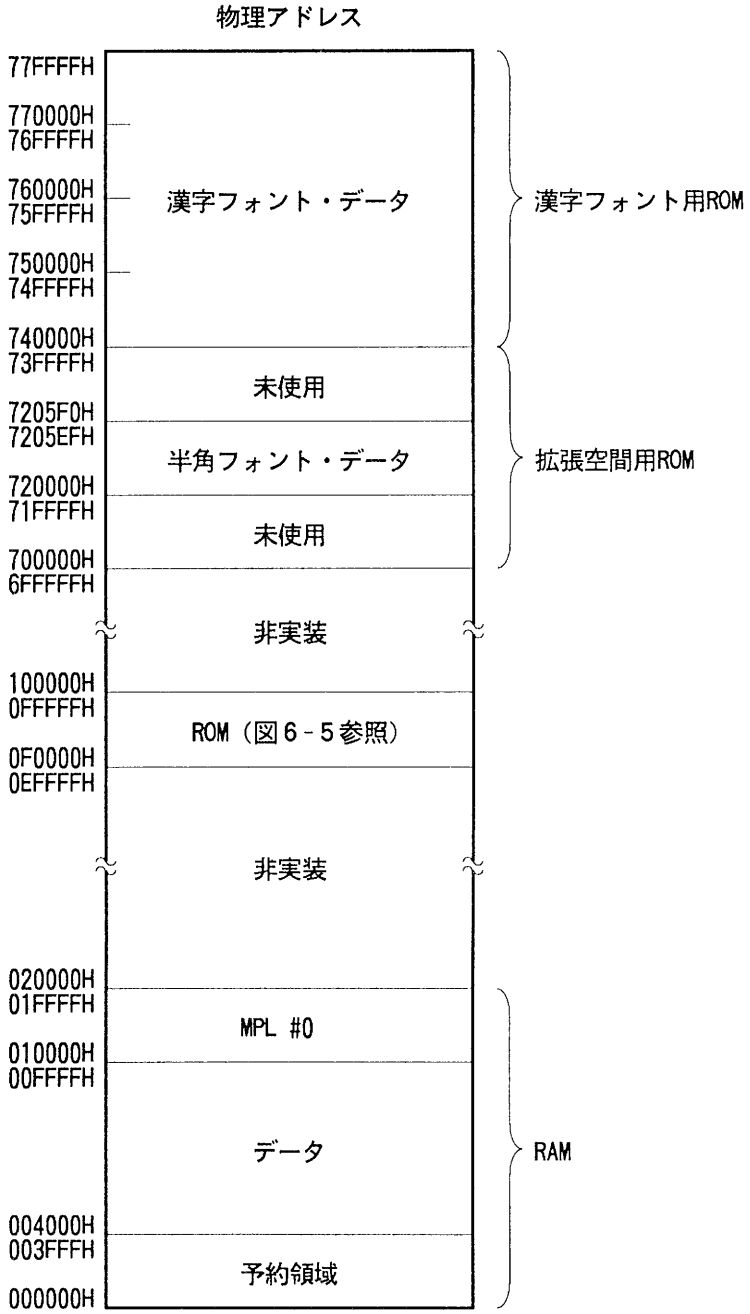
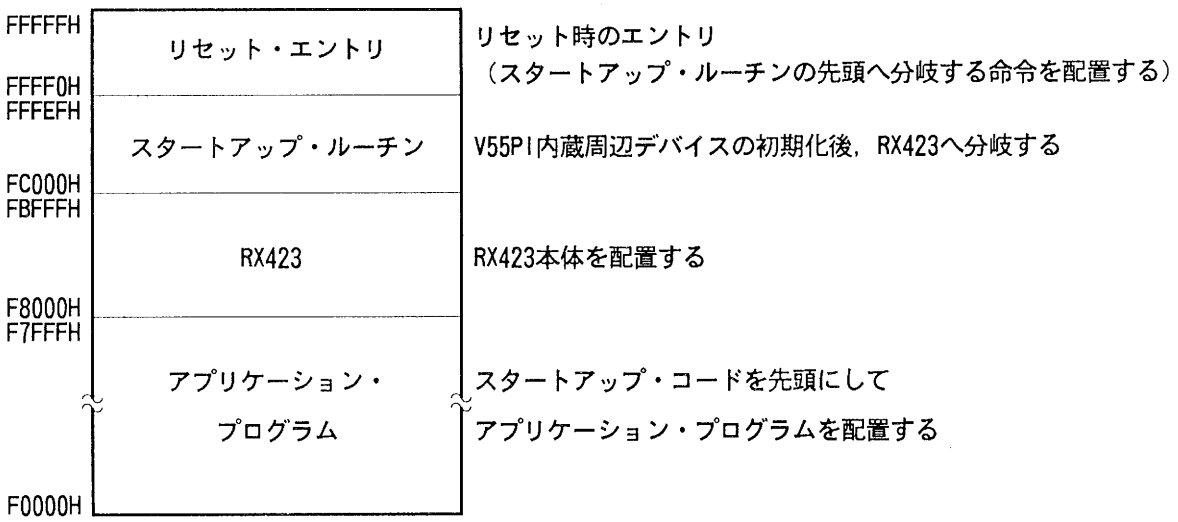


図６－５　ROMの内容



付録A　RX136用コーディング・リスト

A.1　タスク部

リストA-1　MAIN.C（1/6）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE                  : main.c
TARGET                  : V53
PROCEDURES DEFINED      : init_task(),
                          rs_init(),
                          mpl_init(),
                          free_mbx_init(),
                          task_init(),
                          sysdata_init();

*****************************************************************************/

#include        "led.h"

#define         led_task        (void(*)())(0xb0000000)
                                /* LEDドライブ・タスクをマッピング
                                   するアドレス                 */

#define         shift_task      (void(*)())(0xb0000000)
                                /* シフト・タスクをマッピング
                                   するアドレス                 */

void    init_task( void );      /* --------- 初期タスク ---------- */
                                /* 各オブジェクト，タスクの生成，  */
                                /* システム情報の初期化，          */
                                /* 割り込みハンドラの定義をする    */
                                /* ------------------------------ */

void    rs_init( void );        /* シリアル受信割り込みハンドラ定義 */
void    obj_init( void );       /* メモリ・プール#1，メールボックス，フラグ，
                                   セマフォの生成 */
void    free_mbx_init( void );  /* フリー・メールボックスの生成 */
void    task_init( void );      /* タスクの生成，起動 */
void    sysdata_init( void );   /* システム情報の初期化 */

/****  アクセス・アドレス        ****/
short   time_task_aa,           /* 時刻タスク                   */
        led_task_aa,            /* LEDドライブ・タスク          */
        mesg_task_aa,           /* メッセージ・タスク           */
        recv_task_aa,           /* 受信データ処理タスク         */
        ctrl_task_aa,           /* コントロール・タスク         */
        shift_task_aa,          /* シフト・タスク               */
        mpl1_aa,                /* メモリ・プール#1             */
        news_mbx_aa,            /* ニュース・メールボックス     */
        cm_mbx_aa,              /* コマーシャル・メールボックス */
        emg_mbx_aa,             /* 緊急メールボックス           */
```

リストA-1　MAIN.C（2/6）

```
        led_mbx_aa,             /* LED表示メールボックス                */
        ctrl_mbx_aa,            /* コマンド・メールボックス              */
        free_mesg_mbx_aa,       /* 表示データ用フリー・メールボックス    */
        free_ctrl_mbx_aa,       /* コマンド用フリー・メールボックス      */
        free_led_mbx_aa,        /* LED表示用フリー・メールボックス       */
        fdata_mbx_aa,           /* LED表示バッファ用メールボックス       */
        flag_aa,                /* 時刻表示タイミング・イベント・フラグ  */
        sys_sem_aa,             /* システム情報排他制御用セマフォ        */
        msg_sem_aa;             /* 表示データ排他制御用セマフォ          */

struct  SYSTEM  sys;            /* システム情報 */
struct  RINGBUF RingBuf;        /* リング・バッファ */

/* ********************** *
 *      初期タスク        *
 * ********************** */
void    init_task( void )
{
        _disable();             /* 割り込み禁止 */

        eb_init();              /* ハードウエアの初期化 */

        rs_init();              /* シリアル受信バッファの初期化 */

        _enable();              /* 割り込み許可 */

        obj_init();             /* 同期通信用資源の初期化 */

        free_mbx_init();        /* フリー・メールボックスの初期化 */

        task_init();            /* タスクの初期化 */

        sysdata_init();         /* システム情報の初期化 */

        ena_int( INT_RS );      /* 指定デバイス番号の割り込みレベルを
                                   割り込み許可にする */

        exd_tsk();              /* 自タスクの終了，削除 */
}

/* ****************************************** *
 *      シリアル受信割り込みハンドラ定義      *
 * ****************************************** */
void  rs_init( void )
{
        struct  DEF_INT pk_def_int;     /* def_int用構造体 */

        buf_init();                     /* リング・バッファの初期化 */

        pk_def_int.inthdr = (void (*)())rs_inthand;
                /* 定義する割り込みハンドラのスタート・アドレス */

        pk_def_int.intds  = INT_DS;
                /* 定義する割り込みハンドラで使用するDS0値 */

        def_int( INT_RS, &pk_def_int );
                /* シリアル受信割り込みハンドラ定義 */
}
```

リストA-1　MAIN.C（3/6）

```
/* ********************************************************************** *
 *      メモリ・プール#1，メールボックス，フラグ，セマフォの生成          *
 * ********************************************************************** */
void    obj_init( void )
{
        struct  CRE_MPL p_cre_mpl;      /* cre_mpl用構造体 */

        p_cre_mpl.mplsz = MPL_SIZ;      /* 生成するメモリ・プール・サイズ */

        p_cre_mpl.blksz = BLK_SIZ;      /* 生成するメモリ・プールからメモリ・
                                           ブロックを切り出す最小単位 */

        cre_mpl( T_TFIFO, &mpl1_aa, MPL1_ID, &p_cre_mpl );
                /* メモリ・プール#1生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &news_mbx_aa, NEWS_MBX_ID );
                /* ニュース・メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &emg_mbx_aa, EMG_MBX_ID );
                /* 緊急メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &cm_mbx_aa, CM_MBX_ID );
                /* コマーシャル・メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &led_mbx_aa, LED_MBX_ID );
                /* LED表示メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &ctrl_mbx_aa, CTRL_MBX_ID );
                /* コマンド・メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &free_mesg_mbx_aa, FREE_MESG_MBX_ID );
                /* 表示データ用フリー・メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &free_ctrl_mbx_aa, FREE_CTRL_MBX_ID );
                /* コマンド用フリー・メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &free_led_mbx_aa, FREE_LED_MBX_ID );
                /* LED表示用フリー・メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &fdata_mbx_aa, FDATA_MBX_ID );
                /* LED表示バッファ用メールボックス生成 */

        cre_flg( &flag_aa, FLAG_ID );
                /* 時刻表示タイミング・イベント・フラグ生成 */

        cre_sem( T_TFIFO, &msg_sem_aa, MSG_SEM_ID, MSG_SEM_MAX );
                /* 表示データ排他制御用セマフォ生成 */

        cre_sem( T_TFIFO, &sys_sem_aa, SYS_SEM_ID, SYS_SEM_MAX );
                /* システム情報排他制御用セマフォ生成 */
}

/* ********************************** *
 *      フリー・メールボックス初期化    *
 * ********************************** */
void    free_mbx_init( void )
{
        short   blkaa;  /* フリー・メールボックスの未使用メッセージのアクセス・
                           アドレス*/
```

リストA-1　MAIN.C（4/6）

```
        int     i;

        /* 表示データ用フリー・メールボックスの初期化 */
        for ( i = 0; i < FREE_MESG_MAX; i++ ) {
                get_blk( T_NOOPT, &blkaa, mpl1_aa, BLKCNT );
                ((struct NEWS *)CONVA( blkaa ))->free_mbxaa = free_mesg_mbx_aa;
                snd_msg( free_mesg_mbx_aa, blkaa );
        }


        /* コマンド用フリー・メールボックスの初期化 */
        for ( i = 0; i < FREE_CTRL_MAX; i++ ) {
                get_blk( T_NOOPT, &blkaa, mpl1_aa, BLKCNT );
                ((struct NEWS *)CONVA( blkaa ))->free_mbxaa = free_ctrl_mbx_aa;
                snd_msg( free_ctrl_mbx_aa, blkaa );
        }


        /* LED表示用フリー・メールボックスの初期化 */
        for ( i = 0; i < FREE_LED_MAX; i++ ) {
                get_blk( T_NOOPT, &blkaa, mpl1_aa, BLKCNT );
                ((struct LED *)CONVA( blkaa ))->free_mbxaa = free_led_mbx_aa;
                snd_msg( free_led_mbx_aa, blkaa );
        }
}

/* ********************************** *
 *      タスクの生成，起動            *
 * ********************************** */
void    task_init( void )
{
        struct  CRE_TSK pk_cre_tsk;             /* cre_tsk用構造体 */

        pk_cre_tsk.sta          = time_task;    /* タスク・スタート・アドレス*/

        pk_cre_tsk.stksz        = STACK_SIZ;    /* タスクで使用するユーザ・
                                                   スタック・サイズ */

        pk_cre_tsk.tskpri       = TIME_TASK_PRI; /* タスクの初期優先度 */

        pk_cre_tsk.tskopt       = NO_USE;       /* 数値演算プロセッサを使用する
                                                   か否か */

        pk_cre_tsk.tskds        = TASK_DS;      /* タスクで使用するデータ・
                                                   セグメントの値 */

        pk_cre_tsk.tskgid       = TASK_GID_0;   /* タスク・グループ番号 */

        cre_tsk ( &time_task_aa, TIME_TASK_ID, &pk_cre_tsk );
                /* 時刻タスク生成 */

        pk_cre_tsk.sta          = recv_task;
        pk_cre_tsk.stksz        = STACK_SIZ;
        pk_cre_tsk.tskpri       = RECV_TASK_PRI;
        pk_cre_tsk.tskopt       = NO_USE;
        pk_cre_tsk.tskds        = TASK_DS;
        pk_cre_tsk.tskgid       = TASK_GID_0;

        cre_tsk( &recv_task_aa, RECV_TASK_ID, &pk_cre_tsk );
```

リストA-1　MAIN.C（5/6）

```
        /* 受信割り込み処理タスク生成 */

    pk_cre_tsk.sta          = ctrl_task;
    pk_cre_tsk.stksz        = STACK_SIZ;
    pk_cre_tsk.tskpri       = CTRL_TASK_PRI;
    pk_cre_tsk.tskopt       = NO_USE;
    pk_cre_tsk.tskds        = TASK_DS;
    pk_cre_tsk.tskgid       = TASK_GID_0;

    cre_tsk( &ctrl_task_aa, CTRL_TASK_ID, &pk_cre_tsk );
        /* コントロール・タスク生成 */

    pk_cre_tsk.sta          = led_task;
    pk_cre_tsk.stksz        = STACK_SIZ;
    pk_cre_tsk.tskpri       = LED_TASK_PRI;
    pk_cre_tsk.tskopt       = NO_USE;
    pk_cre_tsk.tskds        = TASK_DS_LED;
    pk_cre_tsk.tskgid       = TASK_GID_2;

    cre_tsk( &led_task_aa, LED_TASK_ID, &pk_cre_tsk );
        /* LEDドライブ・タスク生成 */

    pk_cre_tsk.sta          = mesg_task;
    pk_cre_tsk.stksz        = STACK_SIZ;
    pk_cre_tsk.tskpri       = MESG_TASK_PRI;
    pk_cre_tsk.tskopt       = NO_USE;
    pk_cre_tsk.tskds        = TASK_DS;
    pk_cre_tsk.tskgid       = TASK_GID_0;

    cre_tsk( &mesg_task_aa, MESG_TASK_ID, &pk_cre_tsk );
        /* メッセージ・タスク生成 */

    pk_cre_tsk.sta          = shift_task;
    pk_cre_tsk.stksz        = STACK_SIZ;
    pk_cre_tsk.tskpri       = SHIFT_TASK_PRI;
    pk_cre_tsk.tskopt       = NO_USE;
    pk_cre_tsk.tskds        = TASK_DS_SHIFT;
    pk_cre_tsk.tskgid       = TASK_GID_1;

    cre_tsk( &shift_task_aa, SHIFT_TASK_ID, &pk_cre_tsk );
        /* シフト・タスク生成 */

    sta_tsk( led_task_aa );         /* LEDドライブ・タスク・スタート */
    sta_tsk( mesg_task_aa );        /* メッセージ・タスク・スタート */
    sta_tsk( shift_task_aa );       /* シフト・タスク・スタート */
    sta_tsk( time_task_aa );        /* 時刻タスク・スタート */
    sta_tsk( recv_task_aa );        /* 受信割り込み処理タスク・スタート */
    sta_tsk( ctrl_task_aa );        /* コントロール・タスク・スタート */
}

/* ********************************************************************** *
 *      システム時刻の初期化，モード設定，時間変更用フラグの初期化       *
 * ********************************************************************** */
void    sysdata_init( void )
{
    wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

    sys.time.curnt_min = SYS_TIME_INIT_MIN;         /* システム時刻初期化*/
    sys.time.curnt_hour = SYS_TIME_INIT_HOUR;
```

リストA-1　MAIN.C（6/6）

```
        sys.new_time_flag = RESET;                  /* システム時刻変更用フ
                                                       ラグ・リセット*/

        sys.curnt_mode = MIX_MODE;                  /* 混在モード */

        sig_sem( sys_sem_aa, SYS_SEM_COUNT );
}
```

リストA-2　CTRL.C（1/4）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : ctrl.c
TARGET          : V53
PROCEDURES DEFINED      :       ctrl_task(),
                                unlink_and_send(),
                                del_mesg();

*****************************************************************************/

#include        "led.h"

void    ctrl_task( void );      /* -------- コントロール・タスク -------- */
                                /* 受信データ処理タスクから送られたコマン */
                                /* ド・メッセージを受け取り，表示データの */
                                /* 追加，削除，表示モードの変更，表示デー */
                                /* タの参照をする。                       */
                                /* ---------------------------------- */

void    unlink_and_send( unsigned char );       /* 表示データの参照 */
void    del_mesg( unsigned char );              /* 表示データの削除 */

short   dummy_msgaa;                            /* マーキング・メッセージ・アク
                                                   セス・アドレス */

/* ***************************** *
 *     コントロール・タスク      *
 * ***************************** */
void    ctrl_task( void )
{
        short   cmd_msgaa;              /* コマンド・メッセージ・アクセス・
                                           アドレス */

        get_blk( T_NOOPT, &dummy_msgaa, mpl1_aa, BLKCNT );
                /* 表示マーキング・メッセージ用のメモリ・ブロックを獲得する */

        ((struct NEWS *)CONVA(dummy_msgaa))->msg_id = MARKING_ID;
                /* マーキングIDを登録する */

        while (1) {
                rcv_msg( T_NOOPT, &cmd_msgaa, ctrl_mbx_aa );
                        /* コマンド・メッセージ受信 */

                switch( ((struct COMMAND *)CONVA(cmd_msgaa))->command ) {
                        /* コマンド・メッセージのコマンドを見て，
                           それぞれの処理をする */

                        case DEL_MESG:          /* 表示データ削除 */
                                del_mesg(
                                   ((struct COMMAND *)CONVA(cmd_msgaa))->mode
                                );
                                break;

                        case CHANGE_MODE:       /* 表示モード変更 */
                                wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

                                sys.curnt_mode =
```

リストA-2　CTRL.C（2/4）

```
                        ((struct COMMAND *)CONVA(cmd_msgaa))->mode;
                                /* 表示モードを，コマンド・メッセージの
                                   モードに変更 */

                        sig_sem( sys_sem_aa, SYS_SEM_COUNT );

                        break;

                case LOOK_MESG:         /* 表示データの参照 */
                        unlink_and_send(
                           ((struct COMMAND *)CONVA(cmd_msgaa))->mode
                        );
                        break;

        }

        snd_msg(
                ((struct COMMAND *)CONVA(cmd_msgaa))->free_mbxaa,
                cmd_msgaa
                );
                /* マーキング・メッセージをフリー・メールボックスに返却*/
    }
}

/* ************************************************ *
 *      関数名 : unlink_and_send()                  *
 *      機能   : PCへ，表示データを送信する         *
 *      返却値 : なし                               *
 * ************************************************ */
void    unlink_and_send( mode )
unsigned char   mode;           /* コマンド・メッセージのモード */
{
        int     mbxaa;          /* 表示データ・メッセージを受け取るメールボック
                                   スのアクセス・アドレス */

        short   mesgaa;         /* 表示データ・メッセージのアクセス・アドレス*/

        char    put_buf[MSGSIZ];/* PCへの送信用バッファ */


        switch ( mode ) {

                case TIME_MODE:         /* 時刻モード */
                        wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

                        sprintf( put_buf, "T%02d:%02d%c",
                                sys.time.curnt_hour, sys.time.curnt_min, END );
                                /* PCへの送信メッセージを作成 */

                        sig_sem( sys_sem_aa, SYS_SEM_COUNT );

                        rputs( put_buf );       /* PCへ送信 */

                        break;

                case NEWS_MODE:
                case CM_MODE:           /* ニュース，コマーシャル，緊急モード*/
                case EMG_MODE:
                        switch( mode ) {
```

リストA-2　CTRL.C（3/4）

```
        /* モードによって参照するメールボックス・アクセス・
           アドレスをセットする */

            case NEWS_MODE:
                    mbxaa = news_mbx_aa;
                    break;

            case CM_MODE:
                    mbxaa = cm_mbx_aa;
                    break;

            case EMG_MODE:
                    mbxaa = emg_mbx_aa;
                    break;
    }

    wai_sem( T_NOOPT, msg_sem_aa, MSG_SEM_COUNT );

    snd_msg( mbxaa, dummy_msgaa );
            /* マーキング・メッセージ送信 */

    do {
            rcv_msg( T_NOOPT, &mesgaa, mbxaa );
                    /* 表示データ・メッセージ受信 */

            if ( ((struct NEWS *)CONVA(mesgaa))->msg_id
                                    != MARKING_ID ) {
                    /* 表示データ・メッセージがマーキング・
                       メッセージでない場合 */

                    sprintf(
                            put_buf,
                            "%c%s%c",
                            STX,
                            ((struct NEWS *)CONVA(mesgaa))
                                    ->nmsg,
                            ETX
                    );
                            /* PCへの送信メッセージ作成 */

                    snd_msg( mbxaa, mesgaa );
                            /* 表示データ・メッセージを元の
                               メールボックスへ返す */

                    rputs( put_buf ); /* PCへ送信 */
            }
    } while ( ((struct NEWS *)CONVA(mesgaa))->msg_id
                            != MARKING_ID );
            /* マーキング・メッセージを受け取るまで */

    sig_sem( msg_sem_aa, MSG_SEM_COUNT );

    sprintf( put_buf, "%c", END );
            /* 表示データ終了メッセージ作成 */

    rputs( put_buf ); /* PCへ送信 */

    break;
}
```

リストA-2　CTRL.C（4/4）

```
}

/* ************************************************ *
 *      関数名 : del_mesg()                          *
 *      機能   : 登録メッセージを削除する            *
 *      返却値 : なし                                *
 * ************************************************ */
void    del_mesg( mode )
unsigned char mode;             /* コマンド・メッセージのモード */
{
        int     mbxaa,          /* 削除したい表示データ・メッセージを
                                   受け取るメールボックスのアクセス・
                                   アドレス */

                err_rcv;        /* rcv_msg（システム・コール）の返却値 */

        long    time_to_wait0 = 0;      /* タイム・アウト指定時間 */

        short   delete_mesgaa;  /* 表示データ・メッセージのアクセス・アドレス*/

        switch( mode ) {        /* モードによって，表示データ・メッセージを
                                   削除するメールボックスをセットする*/
                case NEWS_MODE:
                        mbxaa = news_mbx_aa;
                        break;

                case CM_MODE:
                        mbxaa = cm_mbx_aa;
                        break;

                case EMG_MODE:
                        mbxaa = emg_mbx_aa;
                        break;
        }

        wai_sem( T_NOOPT, msg_sem_aa, MSG_SEM_COUNT );

        do {
                err_rcv = rcv_msg( T_TMOUT, &delete_mesgaa, mbxaa,
                                        &time_to_wait0 );
                        /* 即時リターンで，表示データ・メッセージを受け取る */

                if ( err_rcv == TE_OK ) {       /* メッセージを受け取ることが
                                                   できた場合 */

                        snd_msg(
                                ((struct NEWS *)CONVA(delete_mesgaa))
                                                        ->free_mbxaa,
                                delete_mesgaa
                        );
                                /* 表示データ用フリー・メールボックスに表示データ・
                                   メッセージを送信する */
                }

        } while ( err_rcv == TE_OK );
                /* 表示データ・メッセージを受け取れている間 */

        sig_sem( msg_sem_aa, MSG_SEM_COUNT );
}
```

リストA-3　LED.C（1/2）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : led.c
TARGET          : V53
PROCEDURES DEFINED      :       led_task(),
                                init_8255();

*****************************************************************************/

#include        "led.h"


void    led_task( void );       /* ---------- LEDドライブ・タスク ---------- */
                                /* 拡張メモリに配置。                        */
                                /* タスク・グループIDは2。                   */
                                /* LED表示バッファにセットされたフォントデー */
                                /* タのビットに対応するLEDアレイの点灯をする。*/
                                /* ------------------------------------- */

void    init_71055( void );     /* uPD71055 の初期化 */

/* ****************************************** *
 *      関数名 : init_71055()                 *
 *      機能   : μPD71055の初期化            *
 *      返却値 : なし                         *
 * ****************************************** */
void    init_71055( void )
{
        outpw( CTRL_PORT_1_2, INITPIU );        /* モード0, すべてのポートを出力
                                                   にする */

        outpw( CTRL_PORT_3_4, INITPIU );        /* モード0, すべてのポートを出力
                                                   にする */
}

/* ******************** *
 *      LEDタスク       *
 * ******************** */
void    led_task( void )
{
        short           fdata_mbxaa,    /* LED表示用バッファのアドレスを得るた
                                           めのメールボックスのアクセス・アドレ
                                           ス */

                        fdata_blkaa;    /* LED表示用バッファのアドレスを得るた
                                           めのブロックのアクセス・アドレス */

        unsigned short  *p_fdata;       /* LED表示用バッファのアドレスを格納す
                                           るためのポインタ */

        static  int     cnt = 0;        /* LED表示バッファを参照するための, カ
                                           ウンタ */

        long    time_to_wait1 = 1;      /* LED表示間隔 */


        mbx_adr( &fdata_mbxaa, FDATA_MBX_ID );
```

リストA-3　LED.C（2/2）

```
        /* LED表示バッファ用のメールボックスのアクセス・アドレスを得る*/

    rcv_msg( T_NOOPT, &fdata_blkaa, fdata_mbxaa );
        /* LED表示バッファのアドレスを格納したブロックを受け取る */

    p_fdata = (unsigned short *)CONVA(fdata_blkaa);
        /* LED表示バッファのアドレスをp_fdataに代入する */

    rel_blk( fdata_blkaa );
        /* rcv_msgで受け取ったブロックを解放する */

    del_mbx( fdata_mbxaa );
        /* LED表示バッファ用メールボックスを削除する */

    init_71055();
        /* パラレル・インタフェース・ユニット（uPD71055）の初期化 */

    while (1) {
        outpw( PARA1_P0, p_fdata[0*16+cnt] );    /* 64ビット（横1列分）の
        outpw( PARA1_P1, p_fdata[1*16+cnt] );       表示データをLEDアレイ
        outpw( PARA2_P0, p_fdata[2*16+cnt] );       にセット */
        outpw( PARA2_P1, p_fdata[3*16+cnt] );

        outpw( PORT+4,  cnt );
            /* 表示する列番号をセットする */

        wai_tsk( &time_to_wait1 );
            /* LED表示間隔 */

        cnt = ((cnt+1) & 0xf);
            /* 0xfと論理積をとり，0から15までの値を順番にインクリメ
               ントさせていく（15の次は0になる）*/
    }

}
```

リストA-4　MESG.C（1/6）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : mesg.c
TARGET          : V53
PROCEDURES DEFINED      :       mesg_task(),
                                send_news_to_led(),
                                send_emg_to_led(),
                                send_time_to_led();

*****************************************************************************/

#include        "led.h"

void    mesg_task( void );              /* ------ メッセージ・タスク ------ */
                                        /* LED表示用メールボックスからメッ  */
                                        /* セージを受け取り，表示モードに対 */
                                        /* 応するメールボックスからメッセー */
                                        /* ジを受け取ってLED表示用メッセー  */
                                        /* ジにコピーし，LEDメールボックス  */
                                        /* に送信する                       */
                                        /* -------------------------------- */

void    send_news_to_led( int, int * ); /* ニュース・メッセージをLEDメールボッ
                                           クスに送信する */

int     send_emg_to_led( int * );       /* 緊急メッセージをLEDメールボックスに
                                           送信する */

int     send_time_to_led( int * );      /* 時刻メッセージを作成し，LEDメールボ
                                           ックスに送信する */

/* ************************** *
 *      メッセージ・タスク      *
 * ************************** */
void    mesg_task( void )
{
        int             led_msgaa,      /* LEDメッセージ・アクセス・アドレス*/
                        err_time,       /* 時刻メッセージ送信ルーチンの返却値*/
                        prptn;          /* 現在のイベント・フラグの内容 */

        static  int     mode_turn = 0;  /* モード変更用（混在モード） */


        while (1) {
                rcv_msg( T_NOOPT, &led_msgaa, free_led_mbx_aa );
                        /* LED表示用メッセージをフリー・メールボックスから受けとる
                           （無限待ち） */

                err_time = send_emg_to_led( &led_msgaa );
                        /* 時刻メッセージ送信 */

                if ( err_time == SEND_OK ) {    /* 時刻メッセージ送信終了 */
                        continue;               /* while(1)まで戻る */
                }

                wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );
```

リストA-4　MESG.C（2/6）

```
switch ( sys.curnt_mode ) {
        case MIX_MODE:          /* 混在モード */
                sig_sem( sys_sem_aa, SYS_SEM_COUNT );

                err_time = send_time_to_led( &led_msgaa );
                        /* 時刻メッセージ送信 */

                if ( err_time == SEND_OK ) {
                        /* 時刻メッセージ送信終了 */
                        continue;
                        /* while(1)まで戻る */
                }

                if ( (mode_turn & 1) == 0 ) {
                     /* mode_turnと1の論理積が1か0かによって,
                        送信するメッセージをニュースまたはコマ
                        ーシャルをどちらにするかを決める */
                        send_news_to_led( news_mbx_aa,
                                        &led_msgaa );
                        /* ニュース・メッセージ送信 */

                } else {
                        send_news_to_led(cm_mbx_aa,&led_msgaa);
                        /* コマーシャル・メッセージ送信
                         */
                }
                mode_turn++;

                break;

        case CM_MODE:           /* コマーシャル・モード */
                sig_sem( sys_sem_aa, SYS_SEM_COUNT );
                send_news_to_led( cm_mbx_aa, &led_msgaa );
                        /* コマーシャル・メッセージ送信 */

                break;

        case NEWS_MODE:         /* ニュース・モード */
                sig_sem( sys_sem_aa, SYS_SEM_COUNT );
                send_news_to_led( news_mbx_aa, &led_msgaa );
                        /* ニュース・メッセージ送信 */

                break;

        case EMG_MODE:          /* 緊急モード */
                sig_sem( sys_sem_aa, SYS_SEM_COUNT );
                snd_msg( ((struct LED *)CONVA(led_msgaa))->
                        free_mbxaa, led_msgaa );
                        /* フリー・メールボックスに, LED表示メッ
                           セージを送信 */

                break;

        case TIME_MODE:         /* 時刻モード */
                sig_sem( sys_sem_aa, SYS_SEM_COUNT );
                err_time = send_time_to_led( &led_msgaa );
                        /* 時刻メッセージ送信 */

                if ( err_time == SEND_OK ) {
```

リストA-4　MESG.C（3/6）

```
                                        /* 時刻メッセージ送信終了 */
                                continue;
                                        /* while(1)まで戻る */
                        }

                        snd_msg( ((struct LED *)CONVA(led_msgaa))->
                                free_mbxaa, led_msgaa );
                                /* フリー・メールボックスに，LED表示メッ
                                   セージを送信 */
                        break;

                }
        }
}

/* ********************************************************************** *
 *      関数名 : send_emg_to_led()                                        *
 *      機能   : 緊急メッセージ用のメールボックスからメッセージを１つ   *
 *               取り出し，LED表示用のメールボックスに送信する           *
 *      返却値 : < 0> 送信正常終了                                        *
 *               <-1> 送信せず（メッセージが受け取れなかったため）       *
 * ********************************************************************** */
int     send_emg_to_led( led_msgaa )
int     *led_msgaa;             /* LED表示メッセージ・アクセス・アドレス */
{
        int     err_emg,        /* rcv_msgの返却値 */

                emg_msgaa,      /* 緊急メッセージのアクセス・アドレス */

                prptn;          /* 現在のフラグの内容 */

        long    time_to_wait0 = 0;      /* rcv_msg（即時リターン）の
                                           タイム・アウト時間 */

        wai_sem( T_NOOPT, msg_sem_aa, MSG_SEM_COUNT );

        err_emg = rcv_msg( T_TMOUT, &emg_msgaa, emg_mbx_aa, &time_to_wait0 );
                /* 緊急メールボックスから，メッセージを受け取る
                   （即時リターン） */
        if ( err_emg != TE_OK ) {
                /* 緊急メッセージが受け取れなかった（メッセージがない ） */
                sig_sem( msg_sem_aa, MSG_SEM_COUNT );

                return( NOT_SEND );     /* 異常終了 */
        }

        ( ((struct NEWS *)CONVA(emg_msgaa))->left_count )--;
                /* 緊急メッセージの表示回数をデクリメント */

        strcpy( ((struct LED *)CONVA(*led_msgaa))->lmsg,
                        ((struct NEWS *)CONVA(emg_msgaa))->nmsg );
                /* LED表示メッセージに，緊急メッセージをコピー */

        if ( ((struct NEWS *)CONVA(emg_msgaa))->left_count > 0 ) {
                /* 緊急メッセージの表示回数が0より大きい場合 */
                snd_msg( emg_mbx_aa, emg_msgaa );
                        /* 緊急メールボックスに返却（送信） */
        } else {
                /* 表示回数が0の場合 */
```

リストA-4　MESG.C（4/6）

```
                snd_msg( ((struct NEWS *)CONVA(emg_msgaa))->free_mbxaa,
                         emg_msgaa );
                        /* フリー・メールボックスに，緊急メッセージを
                           返却（送信）*/
        }

        sig_sem( msg_sem_aa, MSG_SEM_COUNT );

        snd_msg( led_mbx_aa, *led_msgaa );
                /* LED表示メッセージ（緊急メッセージ）を，LEDメールボックスに
                   送信 */

        set_flg( T_REP, &prptn, flag_aa, SETPTN0 );
                /* 時刻表示タイミング・イベント・フラグのビット0に1をセット*/

        return( SEND_OK );      /* 正常終了 */
}

/* ********************************************************************** *
 *      関数名 : send_time_to_led()                                       *
 *      機能   : 時刻変更のフラグの状態を見て，セットされていれば          *
 *               現在時刻のメッセージを作り，LED表示用のメールボ          *
 *               ックスに送信する                                         *
 *      返却値 : < 0> 送信正常終了                                        *
 *               <-1> 送信せず（メッセージが受け取れなかった）            *
 * ********************************************************************** */
int     send_time_to_led( led_msgaa )
int     *led_msgaa;             /* LED表示メッセージ・アクセス・アドレス */
{
        unsigned int    *min,           /* システム時刻（分）を変更するためのポ
                                           インタ*/

                        *hour;          /* システム時刻（時）を変更するためのポ
                                           インタ*/

        int             prptn,          /* 現在のイベント・フラグの内容 */

                        err_flg;        /* wai_flgの返却値 */

        long            time_to_wait0 = 0;      /* wai_flgのタイム・アウト値*/


        min = &sys.time.curnt_min;      /* システム時刻（分） */
        hour = &sys.time.curnt_hour;    /* システム時刻（時） */

        err_flg = wai_flg( T_ORW | T_RESET | T_TMOUT, &prptn, flag_aa,
                                                SETPTN1, &time_to_wait0 );
                /* 時刻表示タイミング・イベント・フラグのビット0に1がセット
                   されているか */

        if ( err_flg != TE_OK ) {
                /* セットされていない */
                return( NOT_SEND );     /* メッセージ送信なし */
        }

        wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

        sprintf(
                ((struct LED *)CONVA(*led_msgaa))->lmsg,
```

リストA-4　MESG.C（5/6）

```
                "只今の時刻は%02d時%02d分です。",
                *hour,
                *min
                );
                /* 時刻メッセージを作成 */

        sig_sem( sys_sem_aa, SYS_SEM_COUNT );

        snd_msg( led_mbx_aa, *led_msgaa );
                /* LED表示メッセージ（時刻メッセージ）を，LEDメールボックスに
                   送信 */

        return( SEND_OK );      /* 正常終了（送信終了）*/

}

/* ********************************************************************** *
 *      関数名 : send_news_to_led()                                       *
 *      機能   : ニュース，コマーシャルのメールボックスからメッセージを *
 *               1つ取り出し，LED表示用のメールボックスに送信する       *
 *      返却値 : なし                                                     *
 * ********************************************************************** */
void    send_news_to_led( mbxaa, led_msgaa )
int     mbxaa;                  /* メールボックス・アクセス・アドレス */
int     *led_msgaa;             /* LED表示メッセージ・アクセス・アドレス */
{
        int     err_news;       /* rcv_msgの返却値 */

        short   msgaa;          /* ニュース（コマーシャル）メッセージ・アクセス・
                                   アドレス */

        long    time_to_wait0 = 0;      /* rcv_msgのタイム・アウト値 */

        wai_sem( T_NOOPT, msg_sem_aa, MSG_SEM_COUNT );

        err_news = rcv_msg( T_TMOUT, &msgaa, mbxaa, &time_to_wait0 );
                /* メールボックス（ニュースまたはコマーシャル）から，メッセージを
                   受け取る */

        if ( err_news == TE_OK ) {      /* メッセージ受け取り成功 */

                ( ((struct NEWS *)CONVA(msgaa))->left_count )--;
                        /* メッセージの表示回数をデクリメント */

                strcpy( ((struct LED *)CONVA(*led_msgaa))->lmsg,
                                ((struct NEWS *)CONVA(msgaa))->nmsg );
                        /* LED表示メッセージに，ニュース（コマーシャル）メッセ
                           ージをコピー */

                if ( ((struct NEWS *)CONVA(msgaa))->left_count > 0 ) {
                        /* 表示回数が0より多い場合 */

                        snd_msg( mbxaa, msgaa );
                                /* ニュース（コマーシャル）メールボックスに，
                                   ニュース（コマーシャル）メッセージを返却
                                   （送信） */
                } else {
                        /* 表示回数が0の場合 */
```

リストA-4　MESG.C（6/6）

```
                snd_msg( ((struct NEWS *)CONVA(msgaa))->free_mbxaa,
                                msgaa );
                        /* フリー・メールボックスに，ニュース（コマーシャ
                           ル）メッセージを返却（送信）*/
            }

            sig_sem( msg_sem_aa, MSG_SEM_COUNT );

            snd_msg( led_mbx_aa, *led_msgaa );
                    /* LED表示メッセージ（ニュースまたはコマーシャル・
                       メッセージ）を，LEDメールボックスに送信 */

        } else {        /* ニュース（コマーシャル）メッセージが受け取れない
                           （メッセージがない） */

            sig_sem( msg_sem_aa, MSG_SEM_COUNT );

            snd_msg( ((struct LED *)CONVA(*led_msgaa))->free_mbxaa,
                     *led_msgaa );
                    /* フリー・メールボックスに，LED表示メッセージを返却
                       （送信） */
        }
}
```

リストA-5　RECV.C（1/7）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : recv.c
TARGET          : V53
PROCEDURES DEFINED      :       recv_task(),
                                buf_read(),
                                get_msg_data(),
                                new_entry(),
                                new_time_set(),
                                get_command_data(),
                                copy_command();

*****************************************************************************/

#include        "led.h"

void    recv_task( void );              /* ------ 受信データ処理タスク ----- */
                                        /* 受信割り込みハンドラからiret_wup  */
                                        /* で起こされ，リング・バッファの文  */
                                        /* 字を読み出し，コマンド・データ，  */
                                        /* 表示データの送信，時刻の変更を行  */
                                        /* う                                */
                                        /* --------------------------------- */

void    buf_read( void );               /* PCから送信されたコマンド・データ
                                           や表示データを読み出す */

void    get_msg_data( char *, int );    /* リング・バッファから表示データを
                                           読み出す */

void    new_entry( char *, struct RCV_COM * );
                /* 時刻変更または各メールボックスに表示データを送信する */

int     new_time_set( char *, unsigned int *, unsigned int * );
                /* 時刻の変更をする */

void    get_command_data( struct RCV_COM * );
                /* リング・バッファからコマンド・データを読み出す */

/* ********************************** *
 *      受信タスク                    *
 * ********************************** */
void    recv_task( void )
{
        while (1) {
                slp_tsk();      /* 受信割り込みハンドラから，iret_wupで
                                   起こされる */

                buf_read();     /* データ読み出し */
        }
}

/* ****************************************************************** *
 *      関数名  :  buf_read()                                         *
 *      機能    :  PCから送信されたコマンドや表示データを読み出す     *
 *      返却値  :  なし                                               *
 * ****************************************************************** */
```

リストA-5　RECV.C（2/7）

```
void    buf_read( void )
{
        int                     lrp,            /* ETXサーチ用ポインタ */
                                datasiz,        /* 表示データ文字数 */
                                dnum;           /* リング・バッファ書き込み文字数
                                                 */
        char                    msg[MSGSIZ];    /* 表示データ格納バッファ */
        short                   cmdaa;          /* コマンド・メッセージ・アクセ
                                                   ス・アドレス */
        static  struct  RCV_COM cmd;            /* 受信したコマンド・データを格
                                                   納する構造体 */

        while (1) {     /* PCから送信されるデータのヘッダ'P'または'STX'が出て来る
                           まで，リング・バッファから文字を取り出し続ける
                         */

                while ( RingBuf.buf[RingBuf.rp] != 'P' &&
                        RingBuf.buf[RingBuf.rp] != STX ) {

                                /* 書き込みポインタの文字が'P'か'STX'でない
                                   場合 */
                        if ( rgetc() == D_EMPTY ) {
                                        /* リング・バッファから文字を取り出す */
                                return;
                                        /* リング・バッファが空になったら，リタ
                                           ーン */
                        }
                }

                dnum = RingBuf.dnum;    /* リング・バッファの書き込み文字数を
                                           保存しておく */

                switch ( RingBuf.buf[RingBuf.rp] ) {
                        case 'P':       /* コマンド・データのヘッダ */

                                if(dnum < sizeof(struct RCV_COM)) {
                                        /* コマンド・データの文字数が少ない */

                                        return;
                                }

                                get_command_data( &cmd );
                                        /* リング・バッファからコマンド・データ
                                           を取り出す */

                                if( cmd.command == ADD_MESG ) {
                                        /* 表示データ追加コマンド */

                                        break;
                                }
                                rcv_msg( T_NOOPT, &cmdaa, free_ctrl_mbx_aa );
                                        /* フリー・メールボックスから，コマンド・
                                           メッセージ用のフリー・メッセージを受け
                                           取る（無限待ち）*/


                                ((struct COMMAND *)CONVA(cmdaa))->command
                                                        = cmd.command;
                                        /* コマンドをコピーする */
```

リストA-5　RECV.C（3/7）

```
                ((struct COMMAND *)CONVA(cmdaa))->mode
                                                = cmd.mode;
                        /* モードをコピーする */

                snd_msg( ctrl_mbx_aa, cmdaa );
                        /* コントロール・タスクにコマンド・メッ
                           セージを送信する*/

                break;

        case STX:       /* 表示データのヘッダ */
                if ( dnum == BUF_SIZ ) {
                        rgetc();
                        return;
                        /* 表示データがリング・バッファ・サイズ
                           を越えている場合は，リターン */
                }

                for ( lrp = 0; dnum > lrp; lrp++ ) {
                        if ( RingBuf.buf[(RingBuf.rp+lrp)
                                & BUF_MASK] == ETX ) {
                                break;
                        }
                }
                        /* 表示データの文字数を数える */
                        /* ETXが現れるまで，lrpをインクリメント
                           する */


                if ( dnum == lrp ) {
                        return; /* ETXが見つからなかった */
                }

                datasiz = lrp-1;
                        /* 表示データの文字数＝lrp-1 */

                get_msg_data( msg, datasiz );
                        /* リング・バッファから表示データを取り
                           出す */

                        /* 表示データ追加 */
                if ( cmd.command == ADD_MESG ) {
                        if ( datasiz > 200 ) {
                                return;
                                /* 表示データが200文字を越えた
                                   場合は，データを無効にする */
                        }

                        new_entry( msg, &cmd );
                                /* 表示データをメールボックスに
                                   送信する */
                }
                break;
        }
    }
}
```

リストA-5　RECV.C（4/7）

```
/* ********************************************************************** *
 *      関数名 : get_msg_data()                                           *
 *      機能   : リング・バッファから指定文字数，データを取り出す          *
 *      返却値 : なし                                                     *
 * ********************************************************************** */
void    get_msg_data( p, bsiz )
char    *p;             /* 取り出した表示データを格納するバッファへのポイン
                           タ*/
int     bsiz;           /* 表示データのバイト数 */
{
        int     i;

        rgetc();                        /* ヘッダのSTXを取り出す*/

        for ( i = 0; i < bsiz; i++ ) {  /* 表示データを取り出し，配列に入れる*/
                *(p+i) = rgetc();
        }

        *(p+i) = (char)NULL;            /* 配列の終わり */

        rgetc();                        /* ETXを取り出す */
}

/* ********************************************************** *
 *      関数名 : get_command_data()                           *
 *      機能   : リング・バッファからコマンド・データを取り出す *
 *      返却値 : なし                                         *
 * ********************************************************** */
void    get_command_data( p )
struct RCV_COM  *p;             /* コマンド・データの構造体へのポインタ */
{
        int     i;

        p->header = (unsigned char)rgetc();     /* ヘッダ */
        p->command = (unsigned char)rgetc();    /* コマンド */
        p->mode = (unsigned char)rgetc();       /* モード */

        for ( i = 0; i < 4; i++ ) {
                rgetc();        /* 送信されたメッセージIDは無視する */
        }

        for ( i = 0; i < 4; i++ ) {
                p->left_cnt[i] = (unsigned char)rgetc(); /* 表示回数 */
        }
}

/* ********************************************************************** *
 *      関数名 : new_entry()                                              *
 *      機能   : 時刻設定，各メールボックスにメッセージを送信する        *
 *      返却値 : なし                                                     *
 * ********************************************************************** */
void    new_entry( p_msg, p_cmd )
char    *p_msg;                 /* 表示データを格納したバッファへのポインタ */
struct RCV_COM  *p_cmd;         /* コマンド・データの構造体へのポインタ */
{
        unsigned int    *min,   /* システム時刻（分）を変更するためのポインタ*/

                        *hour;  /* システム時刻（時）を変更するためのポインタ*/
```

リストA-5　RECV.C（5/7）

```
        short           msgaa;  /* 送信する表示データを格納するメッセージの
                                   アクセス・アドレス */

        int             err_free, /* フリー・メールボックスからメッセージをrcv_msg
                                   で受け取るときの返却値 */

                        err_time; /* 時刻変更関数からの返却値 */

        long            time_to_wait0 = 0;/* rcv_msg（即時リターン）のタイム・
                                   アウト時間 */

        min = &sys.time.curnt_min;      /* システム時刻（分） */
        hour = &sys.time.curnt_hour;    /* システム時刻（時） */

        switch ( p_cmd->mode ) {                /* モード */
                case TIME_MODE:         /* 時刻モード */
                        err_time = new_time_set( p_msg, min, hour );
                                        /* 時刻を変更する */

                        if ( err_time == DATA_ERR ) {
                                /* 時刻が正しくなかった */
                                return;
                        }

                        wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

                        sys.new_time_flag = SET; /* システム時刻変更用フラグを
                                                    セットする */

                        sig_sem( sys_sem_aa, SYS_SEM_COUNT );

                        wup_tsk(time_task_aa ); /* 時刻タスクを起床させる */

                        break;

                case NEWS_MODE: /* ニュース，コマーシャル，緊急モード */
case CM_MODE:
                case EMG_MODE:
                        err_free = rcv_msg( T_TMOUT, &msgaa, free_mesg_mbx_aa,
                                                        &time_to_wait0 );
                                /* 表示データのためのフリー・メッセージを受け取る
                                   （即時リターン） */

                        if ( err_free != TE_OK ) {
                                /* メッセージが取れない */
                             return;
                        }

                        ((struct NEWS *)CONVA(msgaa))->msg_id = 0xffff;
                                /* メッセージID（すべて，0xffff） */

                        ((struct NEWS *)CONVA(msgaa))->left_count =
                                                chg_hex( p_cmd->left_cnt, 4 );
                                /* 表示回数 */

                        strcpy( ((struct NEWS *)CONVA(msgaa))->nmsg, p_msg );

                                /* 表示データをコピー */
                        wai_sem( T_NOOPT, msg_sem_aa, MSG_SEM_COUNT );
```

リストA-5　RECV.C（6/7）

```
                        switch ( p_cmd->mode ) {
                                /* モードを見て，それぞれのメールボックスに
                                   メッセージを送信 */

                                case NEWS_MODE:
                                        snd_msg( news_mbx_aa, msgaa );
                                        break;
                                case CM_MODE:
                                        snd_msg( cm_mbx_aa, msgaa );
                                        break;
                                case EMG_MODE:
                                        snd_msg( emg_mbx_aa, msgaa );
                                        break;
                        }

                        sig_sem( msg_sem_aa, MSG_SEM_COUNT );
                break;
        }
}

/* ************************************************** *
 *      関数名 : new_time_set()                       *
 *      機能   : システム時刻を新規に設定する        *
 *      返却値 : < 0> 正常終了                        *
 *               <-1> 時刻データ不正                  *
 * ************************************************** */
int     new_time_set( p, min, hour )
char            *p;             /* 変更する時間データを格納したバッファへのポイ
                                   ンタ */
unsigned int    *min;           /* システム時刻（分）へのポインタ */
unsigned int    *hour;          /* システム時刻（時）へのポインタ */
{
        int             i;
        unsigned int    set_hour,       /* 変更する時刻を代入する変数（時）*/
                        set_min,        /* 変更する時刻を代入する変数（分）*/
                        time = 0;       /* 変更する時刻を代入する変数 */

                /* 配列の指定位置の２文字を数値に変換し，変数に代入する */
                /* 文字が不正（'0'から'9'の間にない）だったら，異常終了 */
        for ( i = 0; i < 2; i++ ) {
                if ( *(p+i*3) <= '9' && *(p+i*3) >= '0' ) {
                        if ( *(p+i*3+1) <= '9' && *(p+i*3+1) >= '0' ) {
                                time = 10*((int)*(p+i*3) & 0x0f)
                                         + ((int)*(p+i*3+1) & 0x0f );

                                if ( i == 0 ) { /* 時データ */
                                        set_hour = time;
                                } else {        /* 分データ */
                                        set_min = time;
                                }

                        } else {
                                return( DATA_ERR );
                        }
                } else {
                        return( DATA_ERR );     /* 異常終了（時刻データ不正）*/
                }
        }
```

リストA-5　RECV.C（7/7）

```
        if ( set_min > SYS_MIN_MAX || set_hour > SYS_HOUR_MAX ) {
                /* 分が60以上，時が24以上の場合 */

                return( DATA_ERR );     /* 異常終了（時刻データ不正） */
        }

        wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

        *hour = set_hour;               /* システム時刻を変更 */
        *min = set_min;

        sig_sem( sys_sem_aa, SYS_SEM_COUNT );

        return( DATA_OK );              /* 正常終了（時刻データ変更） */
}
```

リストA-6　SHIFT.C（1/4）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE                  : shift.c
TARGET                  : V53
PROCEDURES DEFINED      : shift_task(),
                          shift_bit(),
                          lputs(),
                          lprintf();

*****************************************************************************/

#include        "led.h"

void    shift_task( void );     /* ------------ シフト・タスク ------------ */
                                /* 拡張メモリに配置。                       */
                                /* タスク・グループIDは1。                  */
                                /* LEDメールボックスからLED表示メッセージを  */
                                /* 受け取り，フォント・データに変換したあと  */
                                /* LED表示バッファにフォント・データをセット */
                                /* する。                                   */
                                /* そして，LED表示バッファを1ビット・シフし，*/
                                /* SHIFT_TIME時間wai_tskする。              */
                                /* ---------------------------------------- */

void    shift_bit( void );      /* LED表示バッファを1ビット・シフトする */

void    lputs( char * );        /* NULLが出るまで，バッファの内容をLEDに表示
                                   する */

void    lprintf( char *fmt, ... ); /* NULLが出るまで，バッファの内容をLEDに
                                      表示する（書式付き） */

unsigned short  fdata[MAX_BUF][MAX_COLUMN];     /* LED表示用バッファ */

#define         P_SIZ           ((sizeof(unsigned short *) + 15) / 16 )
                                                /* FDATAのポインタ・サイズ */

/* ***************************** *
 *      シフト・タスク           *
 * ***************************** */
void    shift_task( void )
{
        int     i;
        short   led_msgaa,      /* LED表示メッセージ・アクセス・アドレス */

                led_mbxaa,      /* LEDメールボックス・アクセス・アドレス */

                fdata_mbxaa,    /* LED表示用バッファのアドレスを渡すためのメー
                                   ルボックスのアクセス・アドレス*/

                mpl0_aa,        /* メモリ・プール#0アクセス・アドレス */

                fdata_blkaa;    /* LED表示用バッファのアドレスを格納するための
                                   ブロックのアクセス・アドレス */

        long    time_to_wait3000 = 3000;/* wai_tskタイム・アウト時間 */
```

リストA-6　SHIFT.C（2/4）

```
        for ( i = 0; i < MAX_COLUMN; i++ ) {     /* LED表示用バッファの初期化 */
                fdata[0][i] = 0;
                fdata[1][i] = 0;
                fdata[2][i] = 0;
                fdata[3][i] = 0;
                fdata[4][i] = 0;
        }

        mbx_adr( &led_mbxaa, LED_MBX_ID );
                /* LEDメールボックスのアクセス・アドレスを得る */

        mbx_adr( &fdata_mbxaa, FDATA_MBX_ID );
                /* LED表示用バッファのメールボックスのアクセス・アドレスを得る
                 */

        mpl_adr( &mpl0_aa, MPL0_ID );
                /* メモリ・プール#0のアクセス・アドレスを得る */

        get_blk( T_NOOPT, &fdata_blkaa, mpl0_aa, P_SIZ );
                /* メモリ・プール#0から，FDATAアドレス用のブロックを得る */

        *((unsigned short **)CONVA(fdata_blkaa)) =
                        (unsigned short *)(&fdata[0][0]);
                /* LED表示用バッファの先頭アドレスを，ブロックに代入する */

        snd_msg( fdata_mbxaa, fdata_blkaa );
                /* LED表示用バッファのメールボックスにLED表示用バッファ・アド
                   レス・メッセージを送信する */

        lprintf("*** LED SYSTEM ***    V53    " );
                /* 初期メッセージを LED に表示する */

        wai_tsk( &time_to_wait3000 );
                /* LED画面を一定時間止める */

        while (1) {

                rcv_msg( T_NOOPT, &led_msgaa, led_mbxaa );
                        /* LEDメールボックスから，LED表示メッセージを受け
                           取る */

                lputs( ((struct LED *)CONVA(led_msgaa))->lmsg );
                        /* LED表示メッセージをLEDに表示する */

                snd_msg( ((struct LED *)CONVA(led_msgaa))->free_mbxaa,
                                led_msgaa );
                        /* LED表示用フリー・メールボックスに，LED表示メッセージ
                           を返却（送信）する */

                lputs( "        " );
                        /* LEDをクリアする */
        }
}

/* ************************************************************ *
 *      関数名  :  shift_bit()                                  *
 *      機能    :  LED表示用バッファを１ビット・シフトする      *
 *      返却値  :  なし                                         *
 * ************************************************************ */
```

リストA-6　SHIFT.C（3/4）

```
void    shift_bit( void )
{
        int     buf_num,        /* LED表示用バッファ（fdata）のバッファ番号 */

                column;         /* LED表示用バッファ（fdata）の行番号 */

                /* LED表示用バッファ全体を1ビット・シフトする */
        for ( column = 0; column < MAX_COLUMN; column++ ) {
                /*  MAX_COLUMN行分 */

                for ( buf_num = 0; buf_num < (MAX_BUF-1); buf_num++ ) {
                        /* バッファのMAX_BUF-1番目まで */

                        fdata[buf_num][column] <<= 1;
                                /* バッファの1行を1ビット・シフトする */

                        fdata[buf_num][column] |= ((fdata[buf_num+1][column]
                                & 0x8000) >> (MAX_COLUMN-1));
                                /* buf_num番目のバッファの0ビット目に,
                                   buf_num+1番目のバッファのMAX_COLUMN-1ビット
                                   目の内容を代入する */
                }
                fdata[buf_num][column] <<= 1;
                        /* MAX_BUF番目のバッファを1ビット・シフトする */
        }
}


/* ********************************************************************** *
 *      関数名  :  lputs()                                                *
 *      機能    :  NULLが出るまで，バッファの内容をLEDに表示する          *
 *      返却値  :  なし                                                   *
 * ********************************************************************** */
void    lputs( str )
char    *str;           /* LEDに表示させたいバッファの先頭アドレス */

{
        int     i, ii;
        int     delta;          /* フォント・データ取り出しルーチンからの返却値
                                        1=半角，2=全角 */

        long    shift_time = SHIFT_TIME;
                                /* シフト間隔時間 */


        for ( i = 0; str[i] != (char)NULL; i += delta ) {
                /* 表示データ・バッファにNULLが出るまで繰り返す */

                if ( (str[i] == CR) || (str[i] == LF) ) {
                        /* 表示データ・バッファがCRかLFの場合 */

                        for ( ii = 0; ii < MAX_COLUMN; ii++ ) {
                                fdata[MAX_BUF-1][ii] = 0;
                                        /* MAX_BUF-1番目のLED表示バッファに0を
                                           代入する */
                        }

                        delta = 1;
                                /* 表示データ・バッファの内容を1進めるため */
```

リストA-6　SHIFT.C（4/4）

```
                        for ( ii = 0; ii < (MAX_COLUMN*4) ; ii++ ) {
                                /* LEDが全画面スクロールするまで */

                                shift_bit();
                                        /* LED表示バッファを1ビット・シフト */

                                wai_tsk( &shift_time );
                                        /* シフト間隔 */
                        }

                } else {

                        delta = strfont( &str[i], &fdata[MAX_BUF-1][0] );
                                /* 表示データ・バッファの内容のフォント・データを
                                   LED表示バッファに格納する */

                        for ( ii = 0; ii < (delta*8); ii++ ) {
                                /* delta*8(deltaが1なら半角8ビット・シフト,
                                   delta が2なら全角16ビット・シフトする */

                                shift_bit();
                                wai_tsk( &shift_time );
                        }
                }
        }
}

/* ********************************************************************** *
 *      関数名  :  lprintf()                                              *
 *      機能    :  NULLが出るまで，バッファの内容をLED表示する            *
 *                 (書式付き)                                             *
 *      返却値  :  なし                                                   *
 * ********************************************************************** */
void    lprintf( fmt, ... )
char    *fmt;           /* 書式 */
{
        char    str[64];
        long    shift_time = SHIFT_TIME;
        va_list ap;


        va_start( ap, fmt );
        vsprintf( str, fmt, ap );
        va_end( ap );

        lputs( str );
}
```

リストA-7　TIME.C（1/2）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : time.c
TARGET          : V53
PROCEDURES DEFINED      :       time_task(),
                                change_time();

*****************************************************************************/

#include        "led.h"

void    time_task( void );      /* -------------- 時刻タスク -------------- */
                                /* 一定間隔で周期起床する（または受信割り込 */
                                /* み処理タスクから起こされる）。           */
                                /* 起床後，時刻変更用フラグがセットされてい */
                                /* たら，フラグをリセットし，周期起床をやり */
                                /* 直す。                                   */
                                /* セットされていなかったら，時刻の更新をす */
                                /* る。                                     */
                                /* ---------------------------------------- */

void    change_time( void );    /* 時刻を更新する */

/* ******************** *
 *      時刻タスク      *
 * ******************** */
void    time_task( void )
{
        struct  CYC_WUP pk_cyc_wup;             /* cyc_wup用構造体 */
        int             prptn,                  /* 現在のフラグの内容 */
                        p_wupcnt;               /* 起床要求の回数 */

        pk_cyc_wup.utime = INIT_UTIME;          /* 初期起床時刻（上位） */
        pk_cyc_wup.ltime = pk_cyc_wup.interval = CYC_TIME;
                /* 周期起床間隔 */
        pk_cyc_wup.count = UNTIL_DEL_TSK;
                /* 対象タスク（時刻タスク）が削除されるまで，起床要求を出し続
                   ける */

        cyc_wup( T_REL, CURNT_TASK, &pk_cyc_wup );
                /* 周期起床（自タスク，相対時間） */

        set_flg( T_OR, &prptn, flag_aa, SETPTN1 );
                /* 時刻表示タイミング・イベント・フラグのビット0に1をセット */

        while (1) {

                slp_tsk();
                        /* 周期起床によって起きるか，受信割り込み処理タスクから
                           の起床要求によって起こされる */

                wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT);

                if ( sys.new_time_flag == SET ) {/* 時刻変更用フラグがセットされ
                                                    ている場合 */
                        sys.new_time_flag = RESET;
                                /* フラグ・クリア */
```

リストA-7　TIME.C（2/2）

```
                        can_cyc( CURNT_TASK );
                                /* 周期起床をキャンセルする */

                        can_wup( &p_wupcnt, CURNT_TASK );
                                /* 起床要求をキャンセルする */

                        cyc_wup( T_REL, CURNT_TASK, &pk_cyc_wup );
                                /* 周期起床のやり直し */

                } else  {       /* 時刻変更用フラグがセットされていない場合 */

                        change_time();
                                /* 時刻を更新する */
                }

                sig_sem( sys_sem_aa, SYS_SEM_COUNT );

                if (sys.curnt_mode == TIME_MODE || sys.curnt_mode == MIX_MODE) {
                        /* 時刻モードまたは混在モードの場合だけ，時刻表示タイ
                           ミング・イベント・フラグのビット0に1をセットする */

                        set_flg( T_OR, &prptn, flag_aa, SETPTN1 );
                                /* 時刻表示タイミング・イベント・フラグのビット0
                                   に1をセットする */
                }
        }
}

/* ************************************************** *
 *      関数名 : change_time()                        *
 *      機能   : システム時刻を更新する              *
 *      返却値 : なし                                 *
 * ************************************************** */
void    change_time( void )
{
        unsigned int    *min,   /* システム時刻（分）を変更するためのポインタ*/
                        *hour;  /* システム時刻（時）を変更するためのポインタ*/

        min = &sys.time.curnt_min;      /* システム時刻 */
        hour = &sys.time.curnt_hour;

        (*min)++;                       /* システム時刻（分）をインクリメント*/

        if ( *min >= SYS_MIN_MAX ) {    /* システム時刻（分）が，SYS_MIN_MAX以
                                           上になった場合 */

                *min = 0;               /* システム時刻（分）を0にする */

                (*hour)++;              /* システム時刻（時）をインクリメントす
                                           る */

                if ( *hour >= SYS_HOUR_MAX ) {  /* システム時刻（時）が，SYS_
                                                   HOUR_MAX以上になった場合 */
                        *hour = 0;      /* システム時刻（時）を0にする */
                }
        }
}
```

リストA-8　FUNC.C（1/4）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : func.c
TARGET          : V53
PROCEDURES DEFINED      :       rprintf(),
                                rputc(),
                                rputs(),
                                rgetc(),
                                rgetch(),
                                rgets(),
                                buf_init(),
                                chg_hex();

*****************************************************************************/

#include        "led.h"
#include        "stdarg.h"

void    rprintf( char *, ... ); /* リング・バッファから，リターン・コードが来る
                                   まで文字を取り出す（書式付き ）（未使用） */

int     rgetch( void );         /* リング・バッファから1文字取り出す（待ちあ
                                   り）（未使用） */

void    rgets( char * );        /* リング・バッファから，リターン・コードが来る
                                   まで文字を取り出す（未使用） */

int     rgetc( void );          /* リング・バッファから1文字取り出す（待ちな
                                   し） */

int     rputc( char );          /* 1文字をシリアル送信する（未使用） */

void    rputs( char * );        /* NULLが来るまで，表示バッファの内容をシリア
                                   ル送信する（未使用） */

void    buf_init( void );       /* リング・バッファの初期化 */

int     chg_hex( char *, int ); /* 文字列を数値（１６進）に変換する */

/* ********************************************************************** *
 *      関数名 : rprintf()                                                *
 *      機能   : バッファから，CR（リターン・コード）が来るまで           *
 *               PCに文字を送信する（書式付き）（未使用）                 *
 *      返却値 : なし                                                     *
 * ********************************************************************** */
void    rprintf( fmt, ... )
char    *fmt;
{
        char    buf[64];        /* 送信するためのバッファ */
        va_list ap;             /* 各名なし引き数を順々に指す */
        va_start( ap, fmt );    /* 最初の名なし引き数を指すようにする */
        vsprintf( buf, fmt, ap );
                /* 書式どおりに文字列に変換し，bufに格納する */

        va_end( ap );           /* クリーン・アップ */

        rputs( buf );           /* PCに送信CRが来るまで */
}
```

保守／廃止

リストA-8　FUNC.C（2/4）

```
/* ********************************************************************** *
 *      関数名 : rgetch()                                                 *
 *      機能   : リング・バッファから1文字取り出す（待ちあり）（未使用）*
 *      返却値 : リング・バッファから取り出した 1 文字                    *
 * ********************************************************************** */
int  rgetch( void )
{
        int     getdata;        /* リング・バッファから取り出した1文字 */


        while ( (getdata = rgetc()) == D_EMPTY );
                /* D_EMPTY（バッファが空）の間，rgetcし続ける */

        return( getdata );
                /* 1文字取り出したあと，その文字を返す */
}

/* ********************************************************************** *
 *      関数名 : rgets()                                                  *
 *      機能   : リング・バッファから，CR（リターン・コード）が来るまで *
 *               文字を取り出す（未使用）                                 *
 *      返却値 : なし                                                     *
 * ********************************************************************** */
void  rgets( p )
char    *p;             /* リング・バッファから取り出した文字を格納するバッファ
                           へのポインタ */
{
        int     getdata;        /* リング・バッファから取り出した1文字 */
        int     i;

        for ( i = 1; i < BUF_SIZ; i++ ) { /* BUF_SIZ-1回繰り返す */

                  /* リング・バッファの先頭から最後まで */

                getdata = rgetch();
                        /* 1文字取り出し */

                if ( getdata == CR ) {

                        /* 取り出した1文字がリターン・コードだったら，forルー
                           プを抜ける */
                        break;
                }

                *(p++) = (char)getdata;
                        /* バッファをインクリメントして，取り出した文字をバッフ
                           ァに入れる */
        }

        *p = (char)NULL;
                /* バッファの終わり */
}

/* ********************************************************************** *
 *      関数名 : rgetc()                                                  *
 *      機能   : リング・バッファから１文字取り出す（待ちなし）          *
 *      返却値 : リング・バッファから取り出した1文字                      *
 *               <-1> 文字取り出し失敗（リング・バッファが空）   *
 * ********************************************************************** */
```

リストA-8　FUNC.C（3/4）

```
int     rgetc( void )
{
        unsigned char   getdata;        /* リング・バッファから取り出した1文
                                           字 */

        if ( RingBuf.dnum <= 0 ) {      /* 文字が受信されてなかったら，D_EMPTY
                                           を返す */
                return( D_EMPTY );
        }

        getdata  = RingBuf.buf[RingBuf.rp];
                /* リング・バッファから，1文字を取り出す */

        RingBuf.rp = (RingBuf.rp+1) & BUF_MASK;
                /* 読み込みポインタをインクリメントする。もし，読み込みポインタ
                   がリング・バッファのサイズより大きくなったら，読み込みポイン
                   タを0にする */

        _disable();             /* 割り込み禁止 */

        RingBuf.dnum--;         /* 書き込み文字数をデクリメントする */

        _enable();              /* 割り込み許可 */

        return( (int)getdata ); /* 取り出した文字を返す */
}

/* ************************************************** *
 *      関数名 : rputc()                                *
 *      機能   : 1文字をシリアル送信する（未使用）      *
 *      返却値 : シリアルのステータス                   *
 * ************************************************** */
int     rputc( putdata )
char    putdata;                /* 送信する1文字 */
{
        unsigned char   status;         /* シリアル・ステータス */

        while (1) {
                if ( ( status = inp( SST ) ) & 1 ) {
                        /* シリアルのステータスを見て，送信データ・バッファ（STB
                           ）が空になっている（送信データ書き込み可能な）場合 */

                        outp( SCU, putdata );
                                /* 送信データの書き込み */

                        return( (int)status );
                                /* シリアル・ステータスを返す */
                }
        }
}

/* ********************************************************************** *
 *      関数名 : rputs()                                                  *
 *      機能   : NULLが来るまで，表示バッファの内容をシリアル送信する     *
 *                                          （未使用）                    *
 *      返却値 : なし                                                     *
 * ********************************************************************** */
void  rputs( p )
char    *p;             /* 表示バッファへのポインタ */
```

リストA-8　FUNC.C（4/4）

```
{
        while ( *p != (char)NULL ) {
                /* 表示バッファにNULLが来るまで */

                rputc( *p );    /* 1文字送信 */
                p++;            /* バッファをインクリメントする */
        }
}

/* ****************************************** *
 *      関数名 : buf_init()                   *
 *      機能   : リング・バッファの初期化     *
 *      返却値 : なし                         *
 * ****************************************** */
void    buf_init( void )
{
        int     i;

        RingBuf.wp = RingBuf.rp = RingBuf.dnum = 0;
        for ( i = 0; i < BUF_SIZ; i++ ) {
                RingBuf.buf[i] = 0;
        }
                /* 書き込みポインタ，読み込みポインタ，書き込み文字数，バッファ
                   をクリアする */
}

/* ********************************************************** *
 *      関数名 : chg_hex()                                    *
 *      機能   : 文字列を数値（１６進）に変換する             *
 *      返却値 :  変換した数値                                *
 *                <-1> データ・エラー（変換不可能文字）       *
 * ********************************************************** */
int     chg_hex( p, count )
char    *p;             /* 変換する文字列へのポインタ */
int     count;          /* 変換したい文字数 */
{
        int     i,
                num;    /* 16進に変換した数値 */

        for ( i = 0, num = 0; i < count; i++ ) {
                if ( '0' <= *(p+i) && *(p+i) <= '9' ) {
                        num = 0x10*num + (int)(*(p+i) & 0xf);
                } else if ( 'A' <= *(p+i) && *(p+i) <= 'F' ) {
                        num = 0x10*num + (int)(*(p+i) - 0x37);
                } else {
                        return( DATA_ERR );
                }
        }

                /* '0'から'9'または'A'から'F'の文字だけを16進に変換す
                   る。それ以外の場合は，DATA_ERRを返す */

        return( num );          /* 変換した数値を返す */
}
```

リストA-9　INIT.C（1/2）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : init.c
TARGET          : V53
PROCEDURES DEFINED      :       eb_init(),
                                scu_init(),
                                sys_init(),
                                wcu_init(),
                                wait();

*****************************************************************************/

#include        "led.h"

void    eb_init( void );        /* 内蔵周辺デバイスの初期化 */

void    scu_init( void );       /* SCU（シリアル・コントロール・ユニット）の初
                                   期化 */

void    sys_init( void );       /* システムI/Oの初期化 */

void    wcu_init( void );       /* WCU（ウエイト・コントロール・ユニット）の初
                                   期化 */

void    wait( void );           /* I/Oウエイト */

/* ******************************************** *
 *      関数名 : eb_init()                      *
 *      機能   : 内蔵周辺デバイスの初期化       *
 *      返却値 : なし                           *
 * ******************************************** */
void    eb_init( void )
{
        _disable();     /* 割り込み禁止 */
        sys_init();     /* システムI/Oの初期化 */
        scu_init();     /* SCUの初期化 */
        wcu_init();     /* WCUの初期化 */
}

/* ******************************************** *
 *      関数名 : sys_init()                     *
 *      機能   : システムI/Oの初期化            *
 *      返却値 : なし                           *
 * ******************************************** */
void    sys_init( void )
{
        outp( SCTL, 0x10 );     /* ボー・レート・ジェネレータ使用 */
        wait();                 /* 内蔵I/Oアドレス16ビット・バウンダリ */

        outp( OPHA, 0xfe );     /* I/Oアドレス上位8ビット=0xfe */
        wait();

        outp( SULA, 0x00 );
        wait();

        outp( OPSEL, 0x0f );    /* 内蔵DMAU, ICU, TCU, SCU使用可 */
        wait();
}
```

リストA-9　INIT.C（2/2）

```
/* ********************************************************************** *
 *      関数名 : scu_init()                                               *
 *      機能   : SCU（シリアル・コントロール・ユニット）の初期化          *
 *      返却値 : なし                                                     *
 * ********************************************************************** */
void    scu_init( void )
{
        outp( BRC, 0x1a );      /* BRCの初期化（9600ボー） */
        wait();

        outp( SMD, 0xcf );
        wait();
        /* ストップ・ビット2,  パリティなし, 8ビット,  RTCLK/16 */

        outp( SCM, 0x37 );      /* RTS low, error clear, SBRK=0 */
        wait();                 /* RE enable, DTR low, TE enable */

        outp( SIMK, 0x2 );      /* RBRDY no-mask, TBRDY mask */
        wait();
}

/* ********************************************************************** *
 *      関数名 : wcu_init()                                               *
 *      機能   : WCU（ウエイト・コントロール・ユニット）の初期化          *
 *      返却値 : なし                                                     *
 * ********************************************************************** */
void    wcu_init( void )
{
        outp( WMB0, 0x65 );     /* 16Mbyte 下位アドレス 000000H ― 6FFFFFH  */
        wait();                 /*         中位         700000H ― 9FFFFFH  */
                                /*         上位         A00000H ― FFFFFFH  */

        outp( WCY0, 0x07 );     /* 16Mbyte 上位 7 wait  */
        wait();

        outp( WCY1, 0x71 );     /* 16Mbyte 中位 7 wait  */
        wait();                 /*         下位 1 wait  */

        outp( WAC, 0 );         /* 1Mbyte = 000000H ― 0FFFFFH  */
        wait();

        outp( WMB1, 0x53 );     /* 1Mbyte 下位   00000H ― 3FFFFH  */
        wait();                 /*        中位   40000H ― DFFFFH  */
                                /*        上位   E0000H ― FFFFFH  */

        outp( WCY3, 0x12 );     /* 1Mbyte 上位 2 wait , I/O 1 wait */
        wait();

        outp( WCY2, 0x71 );     /* 1Mbyte 中位 7 wait  */
        wait();                 /*        下位 1 wait  */
}

/* ******************** *
 *      I/Oウエイト     *
 * ******************** */
void    wait( void )
{
}
```

リストA-10　SPRINTF.C（1/5）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : sprintf.c
TARGET          : V53
PROCEDURES DEFINED      :       sprintf(),
                                vsprintf(),
                                ItoS(),
                                form(),
                                intlen();

*****************************************************************************/

#include <stdarg.h>

#define ON      1                               /* フラグのセット */
#define OFF     0                               /* フラグの解除 */
#define MINUS   '-'                             /* マイナス記号 */
#define MASK_F  0xf                             /* マスク値     */
#define DECIMAL 10                              /* 10進数       */
#define HEX     4                               /* 16進数       */
#define HEX_A   0xa                             /* 16進数の10   */

int     sprintf( char*, const char*, ... );     /* 書式に従って可変個の */
                                                /* 引き数を書式付けする */
int     vsprintf( char*, const char*, va_list );/* 書式に従って可変個の */
                                                /* 引き数を書式付けする */
void    ItoS( char*, int, char );               /* 数値を文字列に変換 */
void    form( char*, short, short );            /* 指定桁数にそろえる */
short   intlen( unsigned int, char );           /* 数値の桁を求める */

/*********************************************************/
/*      関数名：sprintf()                                */
/*      機能   ：書式に従って可変個の引き数を書式付けする*/
/*      返却値：なし                                     */
/*********************************************************/
int
sprintf( buf, fmt, ... )
char    *buf;                                   /* 作成文字列の格納 */
const char      *fmt;                           /* 書式 */
{
        va_list argp;                           /* 名前なし引き数を指定 */


        va_start( argp, fmt );                  /* 最初の名前なし引き数を指定 */
        vsprintf( buf, fmt, argp );
        va_end( argp );                         /* 終了処理     */
}

/*********************************************************/
/*      関数名：vsprintf()                               */
/*      機能   ：書式に従って可変個の引き数を書式付けする*/
/*               sprintf()関数との違いは可変個の引き数を */
/*               単一の引き数に置き換えている点である    */
/*      返却値：なし                                     */
/*********************************************************/
int
vsprintf( buf, fmt, argp )
```

リストA-10　SPRINTF.C（2/5）

```
char    *buf;                                   /* 作成文字列の格納 */
const char      *fmt;                           /* 書式 */
va_list argp;                                   /* 名前なし引き数を指定 */
{
        char    *sval;                          /* char*型引き数の一時格納 */
        int     ival;                           /* int, char型引き数の一時格納 */
        short   keta_flg;                       /* 桁フラグ */
        short   zero_flg;                       /* ゼロ埋めフラグ */
        short   cnt;                            /* カウント */


        /* 書式の読み込みループ */
        for( *buf = (char)NULL; *fmt; fmt++ ) {         /* ループSTART */
                keta_flg = zero_flg = OFF;              /* フラグ初期化 */
                cnt = (short)NULL;                      /* カウント初期化 */


                if( *fmt != '%' ) {                     /* '%'の出現判定 */
                        *buf = *fmt;
                        *++buf = (char)NULL;
                        continue;
                }

                if( *++fmt == '%' ) {                   /* '%'の出現判定 */
                        *buf = '%';
                        *++buf = (char)NULL;
                        continue;
                }

                if( *fmt == '0' ) {                     /* ゼロ埋めの判定 */
                        if( *(fmt+1) >= '1' && *(fmt+1) <= '9' ) {
                                zero_flg = ON;
                                fmt++;
                        } else {
                                *buf = '0';
                                *++buf = (char)NULL;
                                continue;
                        }
                }

                if( *fmt >= '1' && *fmt <= '9' ) {      /* 桁指定の判定 */
                        keta_flg = ON;
                        cnt = *fmt++ - '0';
                }

                switch ( *fmt ) {
                case 'X':                               /* 16/10進数の場合 */
                case 'x':
                case 'u':
                case 'd':
                        ival = va_arg( argp, int );
                        ltoS( buf, ival, *fmt );
                        break;
                case 'c':                               /* char型の場合 */
                        ival = va_arg( argp, int );
                        *buf = ( char )ival;
                        *(buf+1) = (char)NULL;
                        break;
                case 's':                               /* char*型の場合 */
```

リストA-10　SPRINTF.C（3/5）

```
                sval = va_arg( argp, char* );
                strcpy( buf, sval );
                break;
        default:                                        /* その他の場合 */
                cnt = zero_flg + keta_flg;
                for( ; cnt > (int)NULL; cnt-- ) {
                        *buf = *(fmt-cnt);
                        *++buf = (char)NULL;
                }
                fmt--;
                continue;
        }

        if( keta_flg ) {                                /* 桁フラグの判定 */
                form( buf, cnt, zero_flg );             /* 桁合わせ       */
        }
        buf += strlen( buf );
    }                                                   /* ループEND      */
}

/*********************************************************/
/*      関数名：ltoS()                                   */
/*      機能　：数値を文字列に変換する                   */
/*      返却値：なし                                     */
/*********************************************************/
void
ltoS( str, val, type )
char    *str;                                           /* 文字列         */
int     val;                                            /* 数値           */
char    type;                                           /* 指定進数       */
{
        unsigned int    dval;                           /* 絶対値の格納 */


        dval = (unsigned int)val;                       /* 絶対値の格納 */
        if( type == 'd' && val < (int)NULL ) {          /* 負数の判定   */
                dval = -( --dval );                     /* 絶対値を求める */
                *str++ = MINUS;                         /* 先頭に'-'を入れる */
        }

        *(str += intlen( dval, type )) = (char)NULL;    /* 変換開始位置指定 */

        /* 数値を文字列に変換するループ */
        do{                                             /* ループSTART  */
                str--;
                switch ( type ) {
                case 'd':                               /* 10進数の場合 */
                case 'u':
                        *str = dval % DECIMAL + '0';
                        dval /= DECIMAL;
                        break;
                case 'X':                               /* 16進数の場合 */
                case 'x':
                        *str = dval & MASK_F;
                        if( *str < HEX_A ) {
                                *str += '0';
                        } else {
                                *str -= HEX_A;
                                *str += ( type == 'X' ) ? 'A' : 'a';
```

リストA-10　SPRINTF.C（4/5）

```
                        }
                        dval >>= HEX;
                        break;
                default:
                        break;
                }
        } while( dval > (int)NULL );                    /* ループEND    */
}

/*********************************************************/
/*      関数名：form()                                   */
/*      機能　：文字列を指定された桁数に合わせる         */
/*      返却値：なし                                     */
/*********************************************************/
void
form( str, keta, zero_flg )
char    *str;                                           /* 文字列       */
short   keta;                                           /* 指定桁数     */
short   zero_flg;                                       /* ゼロ埋めフラグ */
{
        short   len;                                    /* 文字列の桁数 */


        len = strlen( str );                            /* 文字列の桁数 */
        str += keta;                                    /* 桁合わせ開始位置 */

        /* 先頭に '-' がある場合（負数）の処理 */
        if( zero_flg && *(str-keta) == MINUS ) {
                len--;
                keta--;
        }

        /* 桁合わせループ */
        /* 文字列が指定桁数より少ない場合                       */
        /* 空いている所には，ゼロ埋めフラグが ON ならば 0 を    */
        /* OFF ならば空白を入れる                               */
        for( ; keta >= (int)NULL; len--, keta--, str-- ) {
                *str = ( len >= (int)NULL ) ?
                                *(str-keta+len):
                                ( zero_flg ) ? '0' : ' ';
        }
}

/*********************************************************/
/*      関数名：intlen()                                 */
/*      機能　：unsigned int型の数値の                   */
/*              指定進数に対する桁数を求める             */
/*      返却値：数値の桁数                               */
/*********************************************************/
short
intlen( val, type )
unsigned int    val;                            /* 変換される数値 */
char    type;                                   /* 指定進数     */
{
        short   len;                            /* 桁数の格納   */


        len = (short)NULL;                      /* 桁数の初期化 */
```

リストA-10　SPRINTF.C（5/5）

```
        /* 桁数を求めるループ */
        /* 指定進数で何回割れるかで桁数を求める */
        do{                                     /* ループSTART */
                len++;                          /* 桁数のインクリメント */

                switch ( type ) {
                case 'd':                       /* 10進数の場合 */
                case 'u':
                        val /= DECIMAL;
                        break;
                case 'X':                       /* 16進数の場合 */
                case 'x':
                        val >>= HEX;
                        break;
                default:
                        break;
                }
        } while( val > (int)NULL );             /* ループEND    */

        return( len );                          /* 桁数を返す   */
}
```

リストA-11　STRFONT.C（1/4）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : strfont.c
TARGET          : V53
PROCEDURES DEFINED      :       strfont(),
                                stoj(),
                                getfont();

*****************************************************************************/
#include        "led.h"

#define         KANFONT_ADDR    0x740000        /* 全角フォント・データのアドレス */

#define         HANFONT_ADDR    0x720000        /* 半角フォント・データのアドレス */

#define         NODATA          0               /* バッファが NULL */

#define         HANKAKU         1               /* 半角 */

#define         ZENKAKU         2               /* 全角 */

#define         FONT_BLKCNT     4               /* フォント・データ用ブロック数 */
#define         iskanji(c) ¥
                (((c)>=0x81 && (c)<=0x9f) || ((c)>=0xe0 && (c)<=0xfc))
                        /* 全角か半角かを調べる */

int             strfont( unsigned char *, unsigned char * );
                        /* バッファの文字に対応するフォント・データを取り出す*/

unsigned short  stoj( unsigned short );
                        /* シフトJISコードをJISコードに変換する */

void            getfont( unsigned short , unsigned char * );
                        /* 漢字ROMから漢字フォントを取り出す */

/* ********************************************************************** *
 *      関数名 : strfont()                                                *
 *      機能   : バッファの文字に対応するフォント・データを取り出す      *
 *      返却値 : <0> データなし                                           *
 *               <1> 半角データ                                           *
 *               <2> 全角データ                                           *
 * ********************************************************************** */
int     strfont( str, buf )
unsigned char   *str;           /* 文字 */
unsigned char   *buf;           /* getfontで取り出した文字のフォント・データを格納するバッファ
 */
{
        if ( *str == (unsigned char)NULL ) {
                return( NODATA );       /* データなし */
        }

        if ( iskanji( *str ) ) {
                getfont( stoj((*str << 8) | (*(str+1)) & 0xff), buf );
                return( ZENKAKU );      /* 全角データ */
```

リストA-11　STRFONT.C（2/4）

```
        } else {
                getfont( 0x8000 | *str, buf );
                return( HANKAKU );      /* 半角データ */
        }
}

/* ********************************************************** *
 *      関数名 : stoj()                                       *
 *      機能   : シフトJISコードをJISコードに変換する         *
 *      返却値 : 変換したJISコードの文字                      *
 * ********************************************************** */
unsigned short stoj( sjiscode )
unsigned short  sjiscode;       /* 変換するシフトJISコードの文字 */
{
        unsigned short  high,
                        low;

        high = (sjiscode & 0xff00) >> 8;
        low  = sjiscode & 0xff;

        if (high < 0xa0) {
                high -= 0x71;
        } else {
                high -= 0xb1;
        }

        high = (high << 1) + 1;

        if (low > 0x7f) {
                low -= 1;
        }

        if (low > 0x9d) {
                low -= 0x7d;
                high += 1;
        } else {
                low -= 0x1f;
        }
        return((high << 8) | low);
}

/* ********************************************************************** *
 *      関数名 : getfont()                                                *
 *      機能   : 拡張ROMから半角，全角のフォント・データを取り出す        *
 *      返却値 : なし                                                     *
 * ********************************************************************** */
void    getfont( code, buf )
unsigned short  code;           /* 文字 */
unsigned char   *buf;           /* フォント・データを格納するバッファ */
{
        int             i, k;
        short           mpl16_addr,     /* メモリ・プール#16アクセス・アドレス
                                           */

                        blk_aa;         /* map_blkするためのブロック・アクセス・
                                           アドレス */

        unsigned long   paddr;          /* フォント・データの物理アドレス */
```

リストA-11　STRFONT.C（3/4）

```
unsigned char   *p;              /* フォント・データのアドレスへのポイン
                                    タ */

mpl_adr( &mpl16_addr, 16 );      /* メモリ・プール#16のアクセス・アドレス
                                    を得る */

get_blk( T_NOOPT, &blk_aa, mpl16_addr, FONT_BLKCNT );
        /* メモリ・プール#16からブロックを獲得する */

if ( code & 0x8000 ) {  /* 半角データ */

        map_blk( blk_aa, (long)HANFONT_ADDR );
                /* 拡張ROMの半角フォント・データのアドレスに，メモリ・
                   プール#16からget_blkしたブロックをマッピングする */

        p = (char *)CONVA(blk_aa);
                /* フォント・データのアドレスへのポインタに，ブロックの
                   アドレスをセットする */

        k = (int)(code - 0x8020);

        for ( i = 0; i < 16; i++ ) {
                buf[i*2+1] = *(p+k*16+i);
        }

        rel_blk( blk_aa );       /* メモリ・プール#16から獲得したブロック
                                    を解放する */
        return;
}

/* 全角データ */

paddr = (unsigned long)((code & 0x1f) | ((code & 0x700)>>1));

switch ( code & 0x7000 ) {
        case 0x2000:
                paddr |= (code & 0x60) << 5;
                break;
        case 0x3000:
        case 0x4000:
                paddr |= (code & 0x60) | ((code & 0x800) >> 1) |
                        ((code & 0x4000) >> 3);
                break;
}

paddr <<= 6;

if ( paddr < 0x10000 ) {
        map_blk( blk_aa, (long)KANFONT_ADDR );
} else if( paddr >= 0x10000 && paddr < 0x20000 ) {
        map_blk( blk_aa, (long)(KANFONT_ADDR + 0x10000) );
} else if( paddr >= 0x20000 && paddr < 0x30000 ) {
        map_blk( blk_aa, (long)(KANFONT_ADDR + 0x20000) );
} else if( paddr >= 0x30000 && paddr < 0x40000 ) {
        map_blk( blk_aa, (long)(KANFONT_ADDR + 0x30000) );
}
        /* paddrの値によって，マッピングするアドレスを変える */

p = (char *)(((long)blk_aa << 16) + ( paddr & 0x0ffff ));
```

リストA-11　STRFONT.C（4／4）

```
                /* blk_aaの64 バイト境界アドレスに，取り出すフォント・データの
                   アドレスのオフセットを足し，フォント・データのアドレスへの
                   ポインタにセットする */

        for ( i = 0; i < 16; i++ ) {
                buf[i*2]   = p[i*4+2];
                buf[i*2+1] = p[i*4];
        }

        rel_blk( blk_aa );      /* メモリ・プール#16から獲得したブロック
                                   を解放する */
}
```

リストA-12　INTHAND.C

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : inthand.c
TARGET          : V53
PROCEDURES DEFINED      :       rs_inthand();

*****************************************************************************/

#include        "led.h"

void    interrupt       rs_inthand( void );
                /* ------------ シリアル受信割り込みハンドラ ------------ */
                /* シリアル受信割り込みが起きると，リング・バッファに1文  */
                /* 字書き込み，受信割り込み処理タスクを起こす             */
                /* ------------------------------------------------------ */

/* ********************************** *
 *      シリアル受信割り込みハンドラ    *
 * ********************************** */
void    interrupt       rs_inthand()
{
        char    putdata;        /* PCから送信された文字 */

        putdata = inp( SCU );   /* PCから送信された文字を読み出す */

        if ( RingBuf.dnum == BUF_SIZ ) {
                /* 文字数が BUF_SIZ を越えていた場合 */

                ret_int();      /* 何も処理をしないで割り込み終了 */
        }

        RingBuf.buf[RingBuf.wp] = putdata;
                /* 読み出した文字をリング・バッファに書き込む */

        RingBuf.wp = (RingBuf.wp+1) & BUF_MASK;
                /* 書き込みポインタをインクリメント。書き込みポインタがバッファ・
                   サイズを越えたら，0にする */

        RingBuf.dnum++;
                /* 書き込み文字数をインクリメント */

        iret_wup( recv_task_aa );       /* 受信割り込み処理タスクを起こす */
}
```

リストA-13　CRETINT.ASM

```
.186
;/*             sccsid[] = "@(#)cretint.asm     1.1             2/6/89";    */

;/*****************************************************************************/
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*         Real-time OS kernel for uPD70136/236        ver 1.00              */
;/*                Copyright (C) NEC Corporation 1987                         */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*****************************************************************************/

;/*****************************************************************************/
;/*             module name     : cretint.asm                                 */
;/*             description     : system_call interface                       */
;/*             history         :                                             */
;/*****************************************************************************/

 include ..¥include¥except.pam
 include ..¥include¥check.pam

__ret_int       segment public 'code'
                assume  CS:__ret_int

;/*****************************************************************************/
;/*             system call     : ret_int                                     */
;/*             function no.    : none                                        */
;/*             detected error  : none                                        */
;/*****************************************************************************/

                public  _ret_int
;               public  __RET_INT
_ret_int        proc    far
;__RET_INT      label   far
;   Recover working registers
        cli
        lea     sp, [bp]
        pop     es
        pop     ds
        popa
;   Call the nucleus
        int     n_ret_int
_ret_int        endp

__ret_int       ends
                end
```

リストA-14　CIRETWUP.ASM（1／2）

```
.186
;/*             sccsid[] = "@(#)ciretwup.asm     1.1             2/6/89";     */

;/*****************************************************************************/
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*         Real-time OS kernel for uPD70136/236          ver 1.00            */
;/*                Copyright (C) NEC Corporation 1987                         */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*                                                                           */
;/*****************************************************************************/

;/*****************************************************************************/
;/*             module name      : ciretwup.asm                               */
;/*             description      : system_call interface                      */
;/*             history          :                                            */
;/*****************************************************************************/

 include ..¥include¥except.pam
 include ..¥include¥check.pam

__iret_wup      segment public 'code'
                assume  CS:__iret_wup

;/*****************************************************************************/
;/*             system call      : iret_wup                                   */
;/*             function no.     : none                                       */
;/*             detected error   : none                                       */
;/*****************************************************************************/

                public  _iret_wup
;               public  __IRET_WUP
_iret_wup       proc    far
;__IRET_WUP     label   far
;   Set address of parameters and recover working registers
        cli
        add     sp, pam_position
        pop     ax
        lea     sp, [bp]
        pop     es
        pop     ds
        pop     di
        pop     si
        pop     bp
        pop     bx
        pop     bx
        pop     dx
        pop     cx
        push    bp
```

リストA-14　CIRETWUP.ASM（2／2）

```
        mov     bp, sp
        xchg    [bp+2], ax
        pop     bp
;   Call the nucleus
        int     n_iret_wup
_iret_wup       endp

__iret_wup      ends
                end
```

リストA-15　LED.H（1/7）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : led.h
TARGET          : V53

*****************************************************************************/

/* ***************************** *
 *       include file            *
 * ***************************** */
#include        "stdio.h"
#include        "rx.h"
#include        "stdarg.h"

/* ***************************** *
 *      I/Oポート・アドレス      *
 * ***************************** */
#define         SCTL            0xfffe  /* システム・コントロール・レジスタ */
#define         OPHA            0xfffc  /* 内蔵ペリフェラル・リロケーション・
                                           レジスタ（上位8ビット）       */

#define         SULA            0xfff8  /* 内蔵ペリフェラル・リロケーション・
                                           レジスタ（SCU）               */

#define         OPSEL           0xfffd  /* 内蔵ペリフェラル選択レジスタ */
#define         SCU             0xfe00  /* SCU I/Oポート・アドレス      */
#define         SCM             (SCU+2) /* シリアル・コントロール・レジスタ */
#define         SST             (SCU+2) /* シリアル・ステータス・レジスタ */
#define         SMD             (SCU+4) /* シリアル・モード・レジスタ    */
#define         SIMK            (SCU+6) /* シリアル割り込みマスク・レジスタ */
#define         BRC             0xffe9  /* ボー・レート・カウンタ        */
#define         WMB0            0xffea  /* プログラマブル・ウエイト・メモリ領域
                                           設定レジスタ0                 */

#define         WMB1            0xfff3  /* プログラマブル・ウエイト・メモリ領域
                                           設定レジスタ1                 */

#define         WAC             0xffed  /* プログラマブル・ウエイト・メモリ・
                                           アドレス・コントロール・レジスタ */

#define         WCY0            0xffec  /* プログラマブル・ウエイト・サイクル数
                                           設定レジスタ0                 */

#define         WCY1            0xffeb  /* プログラマブル・ウエイト・サイクル数
                                           設定レジスタ1                 */

#define         WCY2            0xfff4  /* プログラマブル・ウエイト・サイクル数
                                           設定レジスタ2                 */

#define         WCY3            0xfff5  /* プログラマブル・ウエイト・サイクル数
                                           設定レジスタ3                 */

#define         PORT            0x8000          /* uPD71055 I/Oポート    */
#define         CTRL_PORT_1_2   PORT+6          /* コントロール・ポート1,2 */
#define         CTRL_PORT_3_4   PORT+0xe        /* コントロール・ポート3,4 */
#define         PARA1_P0        PORT            /* PIU1-P0       */
#define         PARA1_P1        PORT+2          /* PIU1-P1       */
```

リストA-15　LED.H（2/7）

```
#define         PARA2_P0        PORT+8          /* PIU2-P0              */
#define         PARA2_P1        PORT+10         /* PIU2-P1              */
#define         PARA1_P2        PORT+4          /* PIU1-P2              */

#define         INITPIU         0x8080
/* ************************** *
 *      転送データ用コード      *
 * ************************** */
#define         STX             0x02    /* 表示メッセージの先頭         */
#define         ETX             0x03    /* 表示メッセージの最後         */
#define         END             0x1a    /* 登録メッセージ送信終わり     */
#define         CR              0xd     /* 改行コード                   */
#define         LF              0xa     /* 左寄せコード                 */

/* ********************************** *
 *      OSシステム・コール用マクロ      *
 * ********************************** */
#define         TASK_DS_SHIFT   0x800   /* シフト・タスクで使用する
                                           データ・セグメント値         */

#define         TASK_DS_LED     0xc00   /* LEDドライブ・タスクで使用
                                           するデータ・セグメント値     */

#define         BLKCNT          (( sizeof( union maxblk ) + 15 )/16 )
                                        /* get__blkするブロック数       */

#define         CURNT_TASK      0       /* 自タスク                     */
#define         TASK_DS         0x400   /* タスクで使用するデータ・
                                           セグメント値                 */

#define         STACK_SIZ       0x400   /* タスクで使用するユーザ・
                                           スタック・サイズ             */

#define         INT_RS          0x0018  /* 受信割り込みデバイス番号     */
#define         INT_DS          0       /* 受信割り込みハンドラで使用する
                                           DS0値                        */

#define         SHIFT_TIME      30      /* シフト・タスクのウエイト時間 */
#define         INIT_UTIME      0       /* 初期起床時刻（上位）         */
#define         CYC_TIME        60000   /* 周期起床時間間隔             */
#define         UNTIL_DEL_TSK   0       /* 対象タスクが削除されるまで，周期起床
                                           要求を出し続ける             */

#define         MPL_SIZ         ((FREE_MESG_MAX + FREE_CTRL_MAX + FREE_LED_MAX¥
                                  + 1)*((BLKCNT*BLK_SIZ)+0x10))
                                        /* メモリ・プール・サイズ       */

#define         BLK_SIZ         0x10    /* メモリ・プールからメモリ・ブロック
                                           を切り出す最小単位           */

#define         NO_USE          0       /* 数値演算プロセッサを使用しない */

#define         TASK_GID_0      0       /* タスク・グループID 0         */
#define         TASK_GID_1      1       /* タスク・グループID 1         */
#define         TASK_GID_2      2       /* タスク・グループID 2         */

#define         SETPTN0         0       /* set_flgで，ビット0を0にクリア */
#define         SETPTN1         1       /* set_flgで，ビット0を1にセット */
```

リストA-15　LED.H（3/7）

```
#define         SYS_SEM_MAX     1       /* システム情報排他制御用セマフォが
                                           管理する資源の最大値          */

#define         MSG_SEM_MAX     1       /* 表示データ排他制御用セマフォが
                                           管理する資源の最大値          */

#define         SYS_SEM_COUNT   1       /* システム情報排他制御用セマフォで
                                           獲得したい（操作する）資源数 */

#define         MSG_SEM_COUNT   1       /* 表示データ排他制御用セマフォで
                                           獲得したい（操作する）資源数 */

/* ********************************** *
 *      リング・バッファ用マクロ        *
 * ********************************** */
#define         BUF_SIZ         256             /* リング・バッファ・サイズ */
#define         D_EMPTY         -1              /* バッファが空           */
#define         BUF_MASK        BUF_SIZ-1       /* リング・バッファ折り返し
                                                   マスク                 */

/* ***************************** *
 *       システム情報マクロ      *
 * ***************************** */
#define         TIME_MODE       '0'     /* 時刻モード             */
#define         NEWS_MODE       '1'     /* ニュース・モード       */
#define         CM_MODE         '2'     /* コマーシャル・モード */
#define         EMG_MODE        '4'     /* 緊急モード             */
#define         MIX_MODE        '8'     /* 混在モード             */

#define         SET             1       /* システム時刻変更用フラグ，セット  */
#define         RESET           0       /* システム時刻変更用フラグ，リセット*/

#define         SYS_TIME_INIT_MIN       0
                                        /* システム時刻，初期時間（分） */

#define         SYS_TIME_INIT_HOUR      0
                                        /* システム時刻，初期時間（時） */

#define         SYS_MIN_MAX     60      /* システム時刻（分）最大値       */
#define         SYS_HOUR_MAX    24      /* システム時刻（時）最大値       */

#define         MAX_BUF         5       /* LED表示用バッファ個数          */
#define         MAX_COLUMN      16      /* LED表示用バッファ列の個数      */

/* ***************************** *
 *       コマンド用マクロ        *
 * ***************************** */
#define         ADD_MESG        '0'     /* 表示データ追加         */
#define         DEL_MESG        '1'     /* 表示データ削除         */
#define         CHANGE_MODE     '2'     /* 表示モード変更         */
#define         LOOK_MESG       '8'     /* 表示データの参照       */

#define         MARKING_ID      0       /* マーキング・メッセージID */
#define         MSGSIZ          BUF_SIZ /* 表示データ・サイズ */

/* ********************************** *
 *       フリー・メールボックス初期値   *
 * ********************************** */
```

リストA-15　LED.H（4/7）

```
#define         FREE_MESG_MAX   20      /* 表示データ用フリー・メールボックスの
                                           最大値 */

#define         FREE_CTRL_MAX   3       /* コマンド用フリー・メールボックスの
                                           最大値 */

#define         FREE_LED_MAX    1       /* LED表示用フリー・メールボックスの
                                           最大値 */

/* ************************************************** *
 *      ブロック・アクセス・アドレス（セグメント）      *
 *              ->16 ビット・アドレス変換マクロ         *
 * ************************************************** */
#define         CONVA( blk_addr )  (((long)(blk_addr) << 16))

/* ********************************************************** *
 *      タスク，メールボックス，フラグ，セマフォID番号          *
 * ********************************************************** */
#define         INIT_TASK_ID            1       /* 初期タスクID             */
#define         LED_TASK_ID             2       /* LEDドライブ・タスクID    */
#define         RECV_TASK_ID            3       /* 受信データ処理タスクID   */
#define         SHIFT_TASK_ID           4       /* シフト・タスクID         */
#define         CTRL_TASK_ID            5       /* コントロール・タスクID   */
#define         TIME_TASK_ID            6       /* 時刻タスクID             */
#define         MESG_TASK_ID            7       /* メッセージ・タスクID     */
#define         NEWS_MBX_ID             1       /* ニュース・
                                                   メールボックスID     */

#define         CM_MBX_ID               2       /* コマーシャル・
                                                   メールボックスID     */

#define         EMG_MBX_ID              3       /* 緊急メールボックスID     */
#define         LED_MBX_ID              4       /* LED表示メールボックスID  */
#define         CTRL_MBX_ID             5       /* コマンド・メールボックスID*/
#define         FREE_MESG_MBX_ID        6       /* 表示データ用フリー・
                                                   メールボックスID     */

#define         FREE_CTRL_MBX_ID        7       /* コマンド用フリー・
                                                   メールボックスID     */

#define         FREE_LED_MBX_ID         8       /* LED表示用フリー・
                                                   メールボックスID     */

#define         FDATA_MBX_ID            9       /* LED表示バッファ用
                                                   メールボックスID     */

#define         MPL0_ID                 0       /* メモリ・プール#0 ID      */
#define         MPL1_ID                 1       /* メモリ・プール#1 ID      */

#define         FLAG_ID                 1       /* 時刻表示タイミング・
                                                   イベント・フラグID   */

#define         MSG_SEM_ID              1       /* 表示データ排他制御用
                                                   セマフォID           */

#define         SYS_SEM_ID              2       /* システム情報排他制御用
                                                   セマフォID           */
```

リストA-15　LED.H（5/7）

```
/* ***************************** *
 *    タスクのプライオリティ    *
 * ***************************** */
#define         INIT_TASK_PRI           110     /* 初期タスク・プライオリティ*/

#define         LED_TASK_PRI            120     /* LEDドライブ・タスク・
                                                   プライオリティ       */

#define         SHIFT_TASK_PRI          130     /* シフト・タスク・
                                                   プライオリティ       */

#define         RECV_TASK_PRI           140     /* 受信データ処理タスク・
                                                   プライオリティ       */

#define         CTRL_TASK_PRI           150     /* コントロール・タスク・
                                                   プライオリティ       */

#define         TIME_TASK_PRI           160     /* 時刻タスク・プライオリティ*/

#define         MESG_TASK_PRI           170     /* メッセージ・タスク・
                                                   プライオリティ       */
/* ********************************** *
 *      other define                  *
 * ********************************** */
#define         DATA_OK         0
#define         DATA_ERR        -1

#define         SEND_OK          0      /* メッセージを送信した */
#define         NOT_SEND        -1      /* メッセージを送信しなかった */

#define         MAX_BUF         5       /* LED表示用バッファ個数        */
#define         MAX_COLUMN      16      /* LED表示用バッファ列の個数    */

/* ********************************** *
 *      externアクセス・アドレス      *
 * ********************************** */

extern  short   time_task_aa,           /* 時刻タスク                   */
                led_task_aa,            /* LEDドライブ・タスク          */
                mesg_task_aa,           /* メッセージ・タスク           */
                recv_task_aa,           /* 受信データ処理タスク         */
                ctrl_task_aa,           /* コントロール・タスク         */
                shift_task_aa,          /* シフト・タスク               */
                mpl1_aa,                /* メモリ・プール#1             */
                news_mbx_aa,            /* ニュース・メールボックス     */
                cm_mbx_aa,              /* コマーシャル・メールボックス */
                emg_mbx_aa,             /* 緊急メールボックス           */
                led_mbx_aa,             /* LED表示メールボックス        */
                ctrl_mbx_aa,            /* コマンド・メールボックス     */
                free_mesg_mbx_aa,       /* 表示データ用フリー・メールボックス*/
                free_ctrl_mbx_aa,       /* コマンド用フリー・メールボックス */
                free_led_mbx_aa,        /* LED表示用フリー・メールボックス  */
                fdata_mbx_aa,           /* LED表示バッファ用メールボックス  */
                flag_aa,                /* 時刻表示タイミング・イベント・フラグ*/
                sys_sem_aa,             /* システム情報排他制御用セマフォ   */
                msg_sem_aa;             /* 表示データ排他制御用セマフォ     */
```

リストA-15　LED.H（6/7）

```
/* ********************************** *
 *      externタスク，ハンドラ          *
 * ********************************** */
extern  void    init_task( void );              /* 初期タスク           */
extern  void    led_task( void );               /* LEDドライブ・タスク  */
extern  void    recv_task( void );              /* 受信データ処理タスク */
extern  void    shift_task( void );             /* シフト・タスク       */
extern  void    ctrl_task( void );              /* コントロール・タスク */
extern  void    time_task( void );              /* 時刻タスク           */
extern  void    mesg_task( void );              /* メッセージ・タスク   */
extern  void interrupt  rs_inthand( void );     /* 受信割り込みハンドラ */

/* ********************************** *
 *      OSシステム・コール用構造体      *
 * ********************************** */
struct  CYC_WUP {       /*** タスクの周期起床（cyc_wup)用構造体 ***/
        long    ltime;          /* 初期起床時刻（下位）          */
        short   utime;          /* 初期起床時刻（上位）          */
        long    interval;       /* 周期起床時間間隔              */
        short   count;          /* 起床回数                      */
};

struct  CRE_TSK {       /*** タスクの生成（cre_tsk）用構造体 ***/
        void    (*sta)();       /* タスク・スタート・アドレス          */
        short   stksz;          /* タスクで使用するユーザ・スタック・サイズ */
        short   tskpri;         /* 生成するタスクの初期優先度          */
        short   tskopt;         /* 数値演算プロセッサを使用するか否か  */
        short   tskds;          /* 生成するタスクで使用する
                                        データ・セグメントの値          */
        short   tskgid;         /* タスク・グループ番号                */
};

struct  CRE_MPL {       /*** メモリ・プールの生成（cre_mpl）用構造体 ***/
        long    mplsz;          /* メモリ・プール全体のサイズ          */
        short   blksz;          /* メモリ・プールからメモリ・ブロックを切り出す
                                   最小単位（バイト）                 */
};

struct  DEF_INT {       /*** 割り込みハンドラ定義（def_int）用構造体 ***/
        void    (*inthdr)();    /* 割り込みハンドラのスタート・アドレス */
        short   intds;          /* 割り込みハンドラ内で使用するDS0値    */
};

/* ********************************** *
 *      LEDシステム構造体               *
 * ********************************** */
struct  RINGBUF {               /*** リング・バッファ構造体 ***/
        int     wp;                     /* 書き込みポインタ     */
        int     rp;                     /* 読み出しポインタ     */
        int     dnum;                   /* 書き込み文字数       */
        unsigned  char  buf[BUF_SIZ];   /* リング・バッファの配列       */
};

struct  TIME {                  /*** システム時刻構造体 ***/
        unsigned int    curnt_hour;     /* システム時刻（時間） */
        unsigned int    curnt_min;      /*              （分）  */
};

struct  NEWS {          /*** ニュース，コマーシャル，緊急メッセージ構造体 ***/
```

リストA-15　LED.H（7/7）

```
        short   free_mbxaa;             /* 使用後返却されるフリー・メールボックス・
                                           アクセス・アドレス          */

        unsigned int    msg_id;         /* メッセージID                */
        unsigned int    left_count;     /* 残り表示回数                */
        char    nmsg[MSGSIZ];           /* メッセージを格納する配列    */
};

struct  COMMAND {               /*** コマンド構造体 ***/
        short   free_mbxaa;             /* 使用後返却されるフリー・メールボックス・
                                           アクセス・アドレス          */

        unsigned char   command;        /* コントロール・コマンド      */
        unsigned char   mode;           /* モード                      */
};

struct RCV_COM {                /*** 受信コマンド構造体 ***/
        unsigned char   header;         /* 受信ヘッダ'P'格納変数       */
        unsigned char   command;        /* コントロール・コマンド      */
        unsigned char   mode;           /* モード                      */
        unsigned char   msg_id[4];      /* メッセージIDを格納する配列  */
        unsigned char   left_cnt[4];    /* 残り表示回数を格納する配列  */
};

struct  LED {                   /*** LED表示メッセージ構造体 ***/
        short   free_mbxaa;             /* 使用後返却されるフリー・メールボックス・
                                           アクセス・アドレス          */

        char    lmsg[MSGSIZ];           /* 表示メッセージを格納する配列 */
};

union   maxblk {                /*** コマンド，メッセージ構造体の共用体 ***/
        struct NEWS     news;           /* ニュース，コマーシャル，
                                           緊急メッセージ構造体        */
        struct COMMAND  command;        /* コマンド構造体              */
        struct LED      led;            /* LED表示メッセージ構造体     */
};

struct  SYSTEM {                /*** システム・データ構造体 ***/
        int      new_time_flag;         /* システム時刻変更用フラグ    */
        unsigned char   curnt_mode;     /* 現在のモード                */
        struct  TIME    time;           /* システム時刻                */
};

/* ********************************** *
 *      extern LEDシステム外部変数    *
 * ********************************** */
extern  struct  RINGBUF         RingBuf;                /* リング・バッファ  */

extern  unsigned short fdata[MAX_BUF][MAX_COLUMN];      /* LED表示用バッファ */
extern  struct SYSTEM   sys;                            /* システム・データ  */

/* ******************** *
 *      extern 関数     *
 * ******************** */
extern  int     rgetc( void );          /* リング・バッファから１文字取り出し*/
extern  void    buf_init( void );       /* リング・バッファの初期化 */
extern  int     chg_hex( char *, int ); /* 文字列ー>１６進数変換            */
/*END OF FILE*/
```

リストA-16　RX.H（1/3）

```
/****************************************
*          ITRONエラー・コード           *
****************************************/
#define TE_OK           0x0     /* 正常終了 */

#define TE_MEM          0x1     /* メモリ領域が確保できない */

#define TE_UDF          0x2     /* 未定義のシステム・コールを発行した */

#define TE_CTX          0x3     /* 発行されるべきでない環境でシステム・コールを
                                   発行した */

#define TE_OBJ          0x4     /* 指定したアクセス・アドレスは対象とするオブジ
                                   ェクト以外を指している */

#define TE_AA           0x5     /* アクセス・アドレスが不適当 */

#define TE_PA           0x6     /* パケット・アドレスが不適当 */

#define TE_MA           0x7     /* メッセージ・アドレス，メモリ・ブロック・アド
                                   レスが不適当 */

#define TE_SA           0x8     /* タスク，例外処理，終了処理，割り込みハンドラ
                                   のスタート・アドレスが不適当 */

#define TE_EXS          0x9     /* 生成しようとしているIDを持つオブジェクトはす
                                   でに存在している */

#define TE_NOEXS        0xa     /* 指定したキー・ワードのオブジェクトが生成され
                                   ていない */

#define TE_NODMT        0xb     /* 指定タスクはDORMANT状態ではない */

#define TE_NOSUS        0xc     /* 指定タスクはSUSPEND状態ではない */

#define TE_DMT          0xd     /* タスクの状態が，DORMANT状態である */

#define TE_IDZR         0xe     /* ID番号が0である */

#define TE_IDOVR        0xf     /* ID番号が大き過ぎる */

#define TE_TMOUT        0x10    /* 指定時刻を経過した */

#define TE_QOVR         0x11    /* キューイングのカウントがオーバフローした */

#define TE_SELF         0x12    /* 自タスクの指定はできない */

#define TE_DLT          0x13    /* 待っている間に，対象オブジェクトが削除された
                                   */

#define TE_OPT          0x14    /* サポートしていないオプションを指定した */

#define TE_WEVF         0x15    /* 指定されたイベント・フラグはすでに他のタスク
                                   が使用している */

#define TE_NOTMR        0x16    /* タイマ・コントロール・テーブルが確保できない
                                   */

#define TE_TPRI         0x17    /* タスク優先度エラー */
```

リストA-16　RX.H（2/3）

```
#define TE_IPRI         0x18    /* 割り込み優先度エラー */

#define TE_NOCYC        0x19    /* cyc_wupが発行されていないタスクに対する
                                   can_wupの発行 */

#define TE_MPLSZ        0x1a    /* メモリ・プール・サイズ・エラー */

#define TE_INTER        0x1c    /* 割り込みレベルの指定で，ビット7-14のいずれか
                                   がセットされている */

#define TE_DVN          0x1c    /* 指定された割り込みレベルは存在しない */

#define TE_SYS          0x1e    /* その他のエラー */

#define TE_PAR          0x1f    /* パラメータが不正 */

#define TE_SVC          0x40    /* 拡張システム・コールの機能コードが不正 */

/**************************************
*       ITRON systemcall options       *
**************************************/
#define T_NOOPT         0x0     /* オプションなし（無限待ちなど） */

#define T_FCOPR         0x1     /* cre_tsk : 数値演算プロセッサを使用する */

#define T_FRCRSM        0x1     /* rsm_tsk : すべてのサスペンド要求を解除 */

#define T_ABS           0x0     /* cyc_wup : 絶対時間 */

#define T_REL           0x1     /* cyc_wup : 相対時間 */

#define T_OR            0x0     /* set_flg : SET */

#define T_AND           0x1     /* set_flg : RESET */

#define T_REP           0x2     /* set_flg : REPLACE */

#define T_EXOR          0x3     /* set_flg : EXOR */

#define T_TMOUT         0x1     /* タイム・アウト指定あり */

#define T_ANDW          0x0     /* wai_flg : mskptnで1にセットされているビット
                                             の全ビットがセットされるまで待つ
                                 */

#define T_ORW           0x2     /* wai_flg : mskptnで1にセットされているビット
                                             のいずれかがセットされるまで待つ
                                 */

#define T_RESET         0x4     /* wai_flg : フラグがセットされ条件が満足したあと
                                             セットされたフラグをリセットする*/

#define T_TPRI          0x0     /* cre_mbx : タスクの優先度順 */

#define T_TFIFO         0x1     /* cre_mbx : タスクのFIFO順 */

#define T_MPRI          0x0     /* cre_mbx : メッセージに付けられた優先度順 */

#define T_MFIFO         0x2     /* cre_mbx : メッセージのFIFO順 */
```

リストA-16　RX.H（3/3）

```
#define T_CPU           0x0     /* def_exc : マシン例外 */

#define T_OS            0x1     /* def_exc : ソフトウエア例外 */

#define T_TEXC          0x0     /* def_exc : 各タスク固有 */

#define T_NEXC          0x2     /* def_exc : 非タスク部 */

#define T_CEXC          0x4     /* def_exc : タスク共通 */
```

リストA-17　MSC60_1.RM

```
cputype D70236    // V53 chip

// This configuration is for ROMable version used with Microsoft C 6.0.

listfile        segments publics by addres      // Generate a segment map=

map     0x00000 to 0x03fff as reserved
map     0x04000 to 0x07fff as rdwr      // data area
map     0x08000 to 0xeffff as reserved
map     0xf0000 to 0xf7fff as rdonly    // program area
map     0xf8000 to 0xfffff as reserved

dup             DATA ROMDATA            // Make a copy of initialized data
dup             CONST ROMDATA
dup             MSG ROMDATA
dup             FAR_DATA ROMFARDATA     // Don't forget the far data

class           CODE = 0xf000           // Assume loading at address f0000H
class           DATA = 0x0400           // Data at address 04000H

order   DATA    ¥                       // RAM class organization
        CONST   ¥
        MSG     ¥
        BSS     ¥
        STACK   ¥
        FAR_DATA ENDFAR_DATA    ¥
        FAR_BSS ENDFAR_BSS

order   CODE    ¥                       // ROM class organization
        ROMDATA ENDROMDATA      ¥
        ROMFARDATA ENDROMFARDATA

output  CODE    ¥                       // Output classes containing code/data
        ROMDATA ENDROMDATA      ¥
        ROMFARDATA ENDROMFARDATA
```

リストA-18　MSC60_2.RM

```
cputype D70236    // V53 chip

// This configuration is for ROMable version used with Microsoft C 6.0.

listfile        segments publics by addres      // Generate a segment map=

map     0x00000 to 0x07fff as reserved  // Interrupt vector table
map     0x08000 to 0x0bfff as rdwr      // 64KB RAM address space
map     0x0c000 to 0xaffff as reserved  // No access
map     0xb0000 to 0xbffff as rdonly
map     0xc0000 to 0xfffff as reserved  // 64KB EPROM address space

dup             DATA ROMDATA            // Make a copy of initialized data
dup             CONST ROMDATA
dup             MSG ROMDATA
dup             FAR_DATA ROMFARDATA     // Don't forget the far data

class           CODE = 0xb000           // Assume loading at address 30000H
class           DATA = 0x0800           // Data at address 04000H

order   DATA    ¥                       // RAM class organization
        CONST   ¥
        MSG     ¥
        BSS     ¥
        STACK   ¥
        FAR_DATA ENDFAR_DATA    ¥
        FAR_BSS ENDFAR_BSS

order   CODE    ¥                       // ROM class organization
        ROMDATA ENDROMDATA      ¥
        ROMFARDATA ENDROMFARDATA

output  CODE    ¥                       // Output classes containing code/data
        ROMDATA ENDROMDATA      ¥
        ROMFARDATA ENDROMFARDATA
```

リストA-19　MSC60_3.RM

```
cputype D70236    // V53 chip

// This configuration is for ROMable version used with Microsoft C 6.0.

listfile        segments publics by addres      // Generate a segment map=

map     0x00000 to 0x03fff as reserved  // Interrupt vector table
map     0x0c000 to 0x0ffff as rdwr      // 64KB RAM address space
map     0x10000 to 0xaffff as reserved  // No access
map     0xb0000 to 0xbffff as rdonly
map     0xc0000 to 0xfffff as reserved  // 64KB EPROM address space

dup             DATA ROMDATA            // Make a copy of initialized data
dup             CONST ROMDATA
dup             MSG ROMDATA
dup             FAR_DATA ROMFARDATA     // Don't forget the far data

class           CODE = 0xb000           // Assume loading at address 30000H
class           DATA = 0x0c00           // Data at address 04000H

order   DATA    ¥                       // RAM class organization
        CONST   ¥
        MSG     ¥
        BSS     ¥
        STACK   ¥
        FAR_DATA ENDFAR_DATA    ¥
        FAR_BSS ENDFAR_BSS

order   CODE    ¥                       // ROM class organization
        ROMDATA ENDROMDATA      ¥
        ROMFARDATA ENDROMFARDATA

output  CODE    ¥                       // Output classes containing code/data
        ROMDATA ENDROMDATA      ¥
        ROMFARDATA ENDROMFARDATA
```

リストA-20　MAKEFILE（1／2）

```
#############################################################################
#       Sample NMAKE makefile for building an embedded system using         #
#       Microsoft C 6.0                                                     #
#############################################################################

AFLAGS  =       /mx /D__L__ /i.
LFLAGS  =       /MAP /CO
CFLAGS  =       /c /ALu /Gm /Zi /Gs /G0

LIBS    =       b:¥lc70116¥msc60¥lib¥rLlibcE ¥
                g:¥lib_chk¥check
                                        # INSLIBによって変更したライブラリ &
                                        # インタフェース・ライブラリ

OBJS    =  main.obj init.obj func.obj inthand.obj ctrl.obj led.obj mesg.obj recv.obj time.obj
sprintf.obj

.c.obj:
        cl $(CFLAGS) $*.c

.asm.obj:
        masm $(AFLAGS) $* ;

all     : main.hex shift.hex led.hex
        @echo 'all done'

main.hex:       main.rom msc60_1.rm makefile
        locate -He -cmsc60_1.rm $*

shift.hex:      shift.rom msc60_2.rm makefile
        locate -He -cmsc60_2.rm $*

led.hex:        led.rom msc60_3.rm makefile
        locate -He -cmsc60_3.rm $*

main.rom:       msc60_1.obj $(OBJS) makefile
        link @<<
                $(LFLAGS) msc60_1.obj $(OBJS)
                main.rom

                $(LIBS)

<<

shift.rom:      msc60_2.obj shift.obj strfont.obj sprintf.obj makefile
        link @<<
                $(LFLAGS) msc60_2.obj shift.obj strfont.obj sprintf.obj
                shift.rom

                $(LIBS)

<<

led.rom:        msc60_3.obj led.obj makefile
        link @<<
                $(LFLAGS) msc60_3.obj led.obj
                led.rom

                $(LIBS)
```

リストA-20　MAKEFILE（2/2）

```
<<

main.obj:       main.c

func.obj:       func.c

init.obj:       init.c

inthand.obj:    inthand.c

ctrl.obj:       ctrl.c

led.obj:        led.c

mesg.obj:       mesg.c

recv.obj:       recv.c

strfont.obj:    strfont.c

shift.obj:      shift.c

time.obj:       time.c

sprintf.obj:    sprintf.c
```

A. 2　スタート・アップ部

リストA-21　STARTUP. ASM（1/5）

```
.186
;/*             sccsid[] = "@(#)startup.asm      1.1             10/27/89";   */
                name    sample_start
;/*************************************************************************/
;/*                                                                       */
;/*                                                                       */
;/*                                                                       */
;/*                                                                       */
;/*                                                                       */
;/*                                                                       */
;/*                                                                       */
;/*       Real Time OS Nucleus for 16bits V-series Micro Processors       */
;/*                  Copyright (C) NEC Corporation  1987                  */
;/*                                                                       */
;/*                                                                       */
;/*                                                                       */
;/*                                                                       */
;/*                                                                       */
;/*                                                                       */
;/*                                                                       */
;/*************************************************************************/

;/*************************************************************************/
;/*     module name     : startup.asm                                     */
;/*     description     : In this module, prepare hardware environment to */
;/*                       start the nucleus normally. Supported I/O devices */
;/*                       are as follows;                                 */
;/*                                                                       */
;/*                       master ICU ( uPD71059 )                         */
;/*                       slave  ICU (     "    )                         */
;/*                       TCU        ( uPD71054 )                         */
;/*                       FPP        ( uPD72291 )                         */
;/*                                                                       */
;/*     history         : created on Jun. 24. 1989                        */
;/*************************************************************************/

 include initinfo.tbl
 include constant.tbl

 sgroup         group   sys_conf
 sys_conf       segment public 'code'
                assume  CS:sys_conf

                extrn   sys_information : far

;/*************************************************************************/
;/*     procedure name  : statup_up_entry                                 */
;/*     function        : initialize hardwares according to configuration */
;/*                       table. this routine hand following parameters to */
;/*                       nuclues;                                        */
;/*                                                                       */
;/*                       ds1:iy = address of configuration table         */
;/*                                                                       */
;/*************************************************************************/

                public  start_up_entry
```

リストA-21　STARTUP.ASM（2/5）

```
start_up_entry  proc    far
        CLI
;   Get address of configuration table
        mov     BX,seg sys_information
        mov     ES,BX
        mov     DI,offset sys_information       ; ds1:iy = address of information

;/*************************************************************************/
;/*      procedure name  : none                                           */
;/*      function        : initialize FPP                                 */
;/*************************************************************************/
;
;       * registers
;               ds1:iy : address of configuration table

;   Get address of FPP information
        LDS     SI,ES:[DI].sys_sys_info_p
;   Check FPP definition
        xor     CX,CX
        mov     cl,[SI].fpp_exist               ; if the address is "0", FPP
        JCXZ    x_check_cpu                     ; isn't used in this system.
        finit

;/*************************************************************************/
;/*      procedure name  : none                                           */
;/*      function        : initlaize internal registers                   */
;/*************************************************************************/
;
;       * registers
;               ds1:iy : address of configuration table

;   Get address of CPU information
 x_check_cpu:
        mov     DX,pgr1_p
        xor     AX,AX
        mov     CX,64
 x_loop2:
        out     DX,AX
        inc     DX
        inc     DX
        inc     AX
        LOOP    x_loop2
;
        LDS     SI,ES:[DI].sys_cpu_info_p       ; ds0:ix = address of CPU information
;   Check CPU type
        mov     CX,DS
        or      CX,SI
        JCXZ    x_init_icu
;   Initialize SCTL
        mov     al,[SI].sctl
        mov     DX,sctl_p
        out     DX,al
;   Initialize OPSEL
        mov     al,[SI].opsel
        mov     DX,opsel_p
        out     DX,al
;   Initialize TCKS
        mov     al,[SI].tcks
        mov     DX,tcks_p
```

リストA-21　STARTUP.ASM（3/5）

```
        out     DX,al
;   Initialize SBCR
        mov     al,[SI].sbcr
        mov     DX,sbcr_p
        out     DX,al
;   Initialize TULA
        mov     al,[SI].tula
        mov     DX,tula_p
        out     DX,al
;   Initialize IULA
        mov     al,[SI].iula
        mov     DX,iula_p
        out     DX,al
;   Initialize OPHA
        mov     al,[SI].opha
        mov     DX,opha_p
        out     DX,al

;       purge   x_check_cpu
;       purge   x_loop

;/*************************************************************************/
;/*      procedure name  : none                                           */
;/*      function        : initialize master and slave ICUs               */
;/*************************************************************************/
;
;       * registers
;               ds1:iy : address of configuration table

 x_init_icu:
        LDS     SI,ES:[DI].sys_icu_info_p       ; ds0:ix = address of ICU information
;   Initialize master ICU
        mov     DX,[SI].low_port
        mov     al,[SI].iw1
        out     DX,al
        mov     DX,[SI].high_port
        mov     al,[SI].iw2
        out     DX,al
        mov     al,[SI].iw3
;   Check slave ICUs
        or      al,al                           ; if data of IW3 is "0", slave
        JZ      x_single                        ; ICUs aren't connected.
        out     DX,al
 x_single:
        mov     al,[SI].iw4
        out     DX,al
        mov     al,[SI].iw3
;   Mask interrupts
        not     al
        out     DX,al
        not     al
        or      al,al
        JZ      x_init_tcu

;   Initialize slave ICUs
        mov     CX,8
 x_loop1:
;   Get address of slave ICU informations
        add     SI,size sys_icu_info            ; ds0:ix = address of slave ICU information
```

リストA-21　STARTUP.ASM（4/5）

```
        mov     DX,[SI].low_port
;   Check definition of slave ICU
        not     DX
        or      DX,DX                           ; if I/O address is "FFFFh",
        JZ      x_skip                          ; initilaize next slave ICU.
;   Send initliaze command
        not     DX
        mov     al,[SI].iw1
        out     DX,al
        mov     DX,[SI].high_port
        mov     al,[SI].iw2
        out     DX,al
        mov     al,[SI].iw3
        out     DX,al
        mov     al,[SI].iw4
        out     DX,al
;   Mask all interrupts
        xor     al,al
        not     al
        out     DX,al                           ; all interrupt are masked
 x_skip:
        LOOP    x_loop1

;       purge   x_loop, x_skip, x_single

;/*************************************************************************/
;/*      procedure name  : none                                           */
;/*      function        : initialize TCU for system clock                */
;/*************************************************************************/
;
;       * registers
;               ds1:iy : address of configuration table

 x_init_tcu:
;   Get address of TCU information
        LDS     SI,ES:[DI].sys_tcu_info_p       ; ds0:ix = address of TCU information
;   Initilaize TCU
        mov     DX,[SI].control_port
        mov     al,[SI].control_word
        out     DX,al
        mov     DX,[SI].counter_port
        mov     AX,[SI].counter_value
        out     DX,al
        mov     al,ah
        out     DX,al

        purge   x_init_tcu

;/*************************************************************************/
;/*      procedure name  : none                                           */
;/*      function        : branch to initial entry of nuclues             */
;/*************************************************************************/
;
;       * registers
;               ds1:iy : address of configuration table

 x_kernel:
        JMP      dword ptr ES:[DI].sys_kernel_p
```

リストA-21　STARTUP.ASM（5/5）

```
start_up_entry  endp
        purge   x_kernel

sys_conf        ends
                end     start_up_entry
```

リストA-22　CONFIG.ASM（1/5）

```
;Real time operating system for V-serise

;/*********************************************************************/
;/*                                                                   */
;/*                                                                   */
;/*              system information table for RX136                   */
;/*                                                                   */
;/*              Copyright (c) NEC corporation 1989                   */
;/*                                                                   */
;/*                                                                   */
;/*********************************************************************/

sgroup          group           sys_conf
sys_conf        segment         public  'code'
                assume          ds:sgroup
                public          sys_information

;/*******************************************************/
;/*              system information table               */
;/*******************************************************/

sys_information         label           far
kernel_p                dw              00000h          ; ニュークリアスの
                                                        ; ロード・
                                                        ; オフセット・アドレス

                        dw              0f800h          ; ニュークリアスの
                                                        ; ロード・
                                                        ; セグメント・アドレス

sys_info_p              dd              sys_sys         ; システム情報
                                                        ; エリアへのポインタ

ram_info_p              dd              sys_ram         ; フリーRAMエリアへの
                                                        ; ポインタ

eram_info_p             dd              sys_eram        ; 拡張メモリ情報
                                                        ; エリアへのポインタ

bram_info_p             dd              sys_bram        ; メモリ・バンク情報
                                                        ; エリアへのポインタ

cpu_info_p              dd              sys_cpu         ; V53用情報エリアへの
                                                        ; ポインタ

icu_info_p              dd              sys_icu         ; ICU情報エリアへの
                                                        ; ポインタ

tcu_info_p              dd              sys_tcu         ; TCU情報エリアへの
                                                        ; ポインタ

pgr_info_p              dd              sys_pgr         ; PGR情報エリアへの
                                                        ; ポインタ

tsk_info_p              dd              sys_tsk         ; タスク情報エリアへの
                                                        ; ポインタ

                        even
sys_cpu                 label           far
```

保守／廃止

リストA-22　CONFIG.ASM（2/5）

```
                db          00h             ; 設定
                db          06h             ; 設定
                db          00h             ; 設定
                db          0FEh            ; 設定
                db          80h             ; 設定
                db          46h             ; 設定
                db          00h             ; 予約


                even
sys_icu         label       far
master          dw          0FE80h          ; IW1, PFCWQ,
                                            ; MCWのライト,
                                            ; ポール・モード,
                                            ; IRR, ISRの
                                            ; リード・アドレス

                dw          0FE82h          ; IW2, IW3, IW4,
                                            ; IMWのライト,
                                            ; IMRのリード・アドレス

                db          13h             ; IW1

                db          38h             ; IW2（38h固定）

                db          00h             ; IW3

                db          01h             ; IW4

slave_0         dw          0FFFFh          ; 未使用
                dw          0FFFFh
                db          11h
                db          00h
                db          00h
                db          01h

slave_1         dw          0FFFFh          ; 未使用
                dw          0FFFFh
                db          11h
                db          00h
                db          01h
                db          01h

slave_2         dw          0FFFFh          ; 未使用
                dw          0FFFFh
                db          11h
                db          00h
                db          02h
                db          01h

slave_3         dw          0FFFFh          ; 未使用
                dw          0FFFFh
                db          11h
                db          00h
                db          03h
                db          01h

slave_4         dw          0FFFFh          ; 未使用
```

リストA-22　CONFIG.ASM（3/5）

```
                dw      0FFFFh
                db      11h
                db      00h
                db      04h
                db      01h

slave_5         dw      0FFFFh          ; 未使用
                dw      0FFFFh
                db      11h
                db      00h
                db      05h
                db      01h

slave_6         dw      0FFFFh          ; 未使用
                dw      0FFFFh
                db      11h
                db      00h
                db      06h
                db      01h

slave_7         dw      0FFFFh          ; 未使用
                dw      0FFFFh
                db      11h
                db      00h
                db      07h
                db      01h

                even
sys_tcu         label   far
interval        dw      1               ; システム・クロックの
                                        ; インターバル・タイム

clock_tcu       dw      0FE46h          ; コントロール・ワード
                                        ; 入力用ポート・アドレス

                dw      0FE40h          ; カウンタ・ポート・
                                        ; アドレス

                dw      1F40h           ; カウント値

                db      34h             ; コントロール・ワード

                db      08h             ; 割り込みレベル

                even
sys_sys         label   far
exception       db      0               ; システム・コール
                                        ; 例外情報

                db      0               ; マシン例外情報

stack_size      dw      0400h           ; ユーザ・タスクの
                                        ; ディフォールト・スタック・
                                        ; サイズ

                db      0               ; μPD72291情報→未使用
                db      1               ; メモリ・モード
                                        ; →拡張用
```

リストA-22　CONFIG.ASM（4/5）

```
                even
sys_ram         label   far
area_count      dw      1               ; 以降定義するエリアの
                                        ; 個数

area_1          dw      1000h           ; low
                                        ; パラグラフ・アドレス

                dw      2000h           ; high
                                        ; パラグラフ・アドレス

                even
sys_eram        label   far
e_area          dw      0C000h          ; 拡張メモリの
                                        ; low
                                        ; パラグラフ・アドレス

                dw      0D000h          ; 拡張メモリの
                                        ; high
                                        ; パラグラフ・アドレス

                even
sys_bram        label    far
b_area          dw      0B000h          ; バンク・メモリ領域の
                                        ; low
                                        ; パラグラフ・アドレス

                dw      0C000h          ; バンク・メモリ領域の
                                        ; high
                                        ; パラグラフ・アドレス

                even
sys_pgr         label    far
                dw      0002h           ; 以降定義する
                                        ; タスク・グループの個数
                dw      01C4h           ; 配置アドレスの
                                        ; 上位10ビット値

                dw      01C8h           ;       〃

                even
sys_tsk         label   far
task_count      dw      1               ; 以降定義する
                                        ; タスクの個数

task_1          dw      1h              ; ID番号

                dw      0h              ; start address offset

                dw      0F000h          ; start address segment

                dw      0400h           ; DS0の値

                dw      0400h           ; タスクの
                                        ; スタック・サイズ

                dw      6Eh             ; プライオリティ(110)

                dw      0h              ; μPD72291→未使用
```

リストA-22　CONFIG.ASM（5/5）

```
                dw          0h          ; 初期値
                dw          0h          ; このタスクのGROUP-ID
sys_conf        ends
                end
```

リストA-23　INITINFO.TBL（1/2）

```
;$nolist
;/*            sccsid[] = "@(#)initinfo.src    1.1            10/24/89";   */
;/*************************************************************************/
;/*            block name      : initinfo.str                             */
;/*            description     : system information table structure       */
;/*************************************************************************/

 sys_info                   struc
        sys_kernel_p        dd      ?       ; +00 address of kernel entry
        sys_sys_info_p      dd      ?       ; +04 address of system information
        sys_ram_info_p      dd      ?       ; +08 address of memory information
        sys_ext_ram_info_p  dd      ?       ; +0c address of memory information
        sys_bnk_ram_info_p  dd      ?       ; +10 address of memory information
        sys_cpu_info_p      dd      ?       ; +14 address of CPU information
        sys_icu_info_p      dd      ?       ; +18 address of ICU information
        sys_tcu_info_p      dd      ?       ; +1c address of TCU information
        sys_pgr_info_p      dd      ?       ; +20 address of PGR data information
        sys_task_info_p     dd      ?       ; +24 address of task information
 sys_info                   ends

 sys_sys_info               struc
        default_task_exc    db      ?       ; +00 default behavior of system call exceptio
n
                                            ;       0: return error code
                                            ;       1: abort task
                            db      ?       ; +01 default behavior of machine exception
                                            ;       0: return error code
                                            ;       1: abort task
        stack_size          dw      ?       ; +02 default stack size for user's task
        fpp_exist           db      ?       ; +03
                                            ;       0: FPP isn't used
                                            ;       1; FPP is used
        memory_mode         db      ?       ; +04
                                            ;       0: normal memory mode
                                            ;       1: extended memory mode
 sys_sys_info               ends

 sys_cpu_info               struc
        sctl                db      ?       ; +00 system control register
        opsel               db      ?       ; +01 on-chip peripheral select
        tcks                db      ?       ; +02 timer clock select
        opha                db      ?       ; +03 on-chip peripheral high address
        iula                db      ?       ; +04 icu low address
        tula                db      ?       ; +05 tcu low address
        sbcr                db      ?       ; +06 stand by mode control register
 sys_cpu_info               ends


 sys_ram_info               struc
        lower_ds            dw      ?       ; +00 lower paragraph address
        upper_ds            dw      ?       ; +02 upper paragraph address
 sys_ram_info               ends

 sys_icu_info               struc
        low_port            dw      ?       ; +00 address of isr, iw1
        high_port           dw      ?       ; +02 address of imr, etc.
        iw1                 db      ?       ; +04 value of iw1
        iw2                 db      ?       ; +05 value of iw2
        iw3                 db      ?       ; +06 value of iw3
```

リストA-23　INITINFO.TBL（2/2）

```
        iw4                 db      ?       ; +07 value of iw4
sys_icu_info                ends

sys_tcu_info                struc
        clock_interval      dw      ?       ; +00 system clock interval
        control_port        dw      ?       ; +02 address of control word
        counter_port        dw      ?       ; +04 address of counter
        counter_value       dw      ?       ; +06 initial value of counter
        control_word        db      ?       ; +08 value of control word
        timer_level         db      ?       ; +09 interrupt level of counter output
sys_tcu_info                ends

sys_task_info               struc
        task_id             dw      ?       ; +00 id number
        task_start_p        dd      ?       ; +02 start address --- pc
                                            ; +04                --- ps
        task_ds             dw      ?       ; +06 value of ds0 register
        task_stack_size     dw      ?       ; +08 initial stack size
        task_priority       dw      ?       ; +10 initial priority
        task_fpp            dw      ?       ; +12 FPP usage option
        task_initial        dw      ?       ; +14 value for initial task
        task_g_id           dw      ?       ; +16 task group id number
sys_task_info               ends

;$list
```

リストA-24　CONSTANT.TBL（1/4）

```
;$nolist
;/*            sccsid[] = "@(#)constant.src    1.1            10/24/89";   */
;/*************************************************************************/
;/*     block name      : equ.dfn                                         */
;/*     description     : value equation for nucleus                      */
;/*************************************************************************/

;/*************************************************************************/
;/*             uPD70236 internal control register addresses              */
;/*************************************************************************/
        sctl_p                  equ     0fffeh
        opsel_p                 equ     0fffdh
        opha_p                  equ     0fffch
        iula_p                  equ     0fffah
        tula_p                  equ     0fff9h
        sbcr_p                  equ     0fff1h
        tcks_p                  equ     0fff0h
        pgr1_p                  equ     0ff00h

;/*************************************************************************/
;/*                        constant of kernel                             */
;/*************************************************************************/
        xa_entry                equ     181
        xa_return               equ     182
        nucleus                 equ     184
        isr_read_order          equ     0bh
        fi_command              equ     20h
        sysvtbase               equ     183*4+2
        icu_intbase             equ     56

;/*************************************************************************/
;/*                    return code of system call                         */
;/*************************************************************************/

;/*---------------- system call level exception code ----------------------*/
        e_ok                    equ     0       ; 00h  normal exit
        e_mem                   equ     1       ; 01h  can't obtain memory area
        e_undef_svc             equ     2       ; 02h  illegal system call
        e_context               equ     3       ; 03h  context error
        e_access_adr            equ     4       ; 04h  access address error
        e_illegal_adr           equ     5       ; 05h  illegal address ( access address )
        e_packet_adr            equ     6       ; 06h  illegal address ( packet address )
        e_block_adr             equ     7       ; 07h  illegal address ( mesage address, memor
y block address )
        e_start_adr             equ     8       ; 08h  illegal address ( procedure start addre
ss )
        e_already_exist         equ     9       ; 09h  existent object
        e_not_exist             equ     10      ; 0ah  not exist object
        e_not_idle              equ     11      ; 0bh  task status isn't idle
        e_not_suspend           equ     12      ; 0ch  task status isn't suspend
        e_idle                  equ     13      ; 0dh  task status is idle
        e_id0                   equ     14      ; 0eh  task id equal '0'
        e_id_bound              equ     15      ; 0fh  task id is greater than max_id
        e_timeout               equ     16      ; 10h  timeout error
        e_count_over            equ     17      ; 11h  queue count overflow
        e_self_task             equ     18      ; 12h  self task no-good
        e_delete_obj            equ     19      ; 13h  delete obj on something_wait
        e_option                equ     20      ; 14h  can't use this option
        e_flg_wait              equ     21      ; 15h  task already wait on this evf
```

リストA-24　CONSTANT.TBL（2/4）

```
e_timer                 equ     22      ; 16h  can't use timer
e_priority              equ     23      ; 17h  priority error
e_interrupt_priority    equ     24      ; 18h  interrut priority error
e_non_cyclic            equ     25      ; 19h  can_cwak to non-cyclic task
e_pool_size             equ     26      ; 1ah  memory pool size error
e_memory_block          equ     27      ; 1bh  rel_blk to not system memory
e_device_no             equ     28      ; 1ch  illegal device number
e_mes_over              equ     29      ; 1dh  message buffer overflow
e_error                 equ     30      ; 1eh  others system error
e_param                 equ     31      ; 1fh  illegal parameter

; original error code
e_exception             equ     128     ; 80h error in exception proc.
e_sys_object            equ     129     ; 81h this object is system task
e_not_extend_pool       equ     130     ; 82h blockaa to not extend pool

;/*-------------------- machine level exception code ----------------------*/
e_zero_divide           equ     8000h   ; zero divide
e_overflow              equ     8004h   ; arithmetic overflow
e_index                 equ     8005h   ; index boundaly check error
e_break                 equ     8007h   ; illegal 'brk' instruction
e_undef                 equ     807ah   ; execute undefined instruction
e_fpp_exist             equ     8082h   ; FPP dosen't exist

;/*---------------------- control block ident.no. -------------------------*/
sy_free_id              equ     0       ; free memory block
sy_usr_id               equ     1       ; usr's memory block
sy_tmb_id               equ     2       ; timer block
sy_tcb_id               equ     3       ; task control block
sy_sema_id              equ     4       ; semaphore control block
sy_mbox_id              equ     5       ; mailbox control bolck
sy_evf_id               equ     6       ; eventflag control block
sy_msg_id               equ     7       ; mesage
sy_free_area_id         equ     8       ;

;/*------------------ tmb (time monitor block) type ----------------------*/
t_start                 equ     1       ; delayed start
t_cyclic_wakeup         equ     2       ; cyclic wakeup
t_semaphore             equ     3       ; semaphore
t_message               equ     4       ; message
t_eventflag             equ     5       ; eventflag
t_memory                equ     6       ; memory block

;/*--------------- pre_declared system task and resource no ---------------*/
no_idle_task            equ     0ffffh  ; system idle task

;/*----------------------  additional system call -------------------------*/
svc_svc_def             equ     64      ;
svc_map_blk             equ     192     ;
svc_max                 equ     50      ;

;/*-----------------  system value  -------------------------------------*/
v_svc_diff              equ     127
v_tsk_exc_return        equ     0
v_tsk_exc_abort         equ     1
v_tsk_exc_debugger      equ     2
v_int_exc_return        equ     0
v_int_exc_debugger      equ     2
v_extend_pool           equ     10h     ; extended memory pool id
```

リストA-24　CONSTANT.TBL（3/4）

```
v_extend_init           equ     224
v_extend_limit          equ     255
v_init_flag             equ     0200h   ; flag for task start
v_suspend_limit         equ     255     ; max suspend count
v_wakeup_limit          equ      62     ; max wakeup count
v_evf_set               equ       0    ; evf set
v_evf_reset             equ       1    ; evf reset
v_evf_rep               equ       2    ; evf replace
v_evf_negate            equ       3    ; evf exor
v_evf_wait_time         equ       1    ; evf wait with timeout
v_evf_wait_or           equ       2    ; evf wait of or
v_evf_wait_reset        equ       4    ; evf wait with reset
v_mbox_msg_pri          equ       2    ; mbox msg is priority base
v_exit_pri              equ       1    ; priority of exit routine
v_except_flag           equ     0200h
v_exit_flag             equ     0200h
v_init_tag              equ     0070h

b_clk_duplicate         equ       0
b_int_enable            equ       9
s_int_enable            equ     0200h

;/*****************************************************************************/
;/*              constant table of system call options                        */
;/*****************************************************************************/

o_fifo                  equ       0
o_sys                   equ       0
o_int                   equ       1
o_common                equ       2

;/*****************************************************************************/
;/*                  constant table of task status                            */
;/*****************************************************************************/

s_ready                 equ     0001h   ; ready
s_sleep                 equ     0002h   ; sleeping
s_timeout               equ     0004h   ; time out
s_evflag                equ     0008h   ; event wait
s_sema                  equ     0010h   ; semaphore wait
s_message               equ     0020h   ; message wait
s_memory                equ     0040h   ; memory wait
s_suspend               equ     0080h   ; suspended
s_dormant               equ     0100h   ; dormant
s_priority              equ     0200h   ; link by priority
s_preempt               equ     1000h   ; switched by preemption
s_exit                  equ     2000h   ; in exit routine
s_cpu_exc               equ     4000h   ; in cpu exception procedure
s_sys_exc               equ     8000h   ; in system call exception procedure

;/*****************************************************************************/
;/*            constant table of task status ( bit location )                 */
;/*****************************************************************************/

b_ready                 equ     0       ; ready
b_sleep                 equ     1       ; sleeping
b_timeout               equ     2       ; time out
b_evflag                equ     3       ; event wait
b_sema                  equ     4       ; semaphore wait
```

リストA-24　CONSTANT.TBL（4／4）

```
        b_message               equ     5       ; message wait
        b_memory                equ     6       ; memory wait
        b_suspend               equ     7       ; suspended
        b_dormant               equ     8       ; dormant
        b_priority              equ     9       ; link by priority
        b_preempt               equ     0ch     ; switched by preemption
        b_exit                  equ     0dh     ; in exit routine
        b_cpu_exc               equ     0eh     ; in cpu exception procedure
        b_sys_exc               equ     0fh     ; in system call exception procedure
;$list
```

リストA-25　STARTUP.RM

```
// configuration for startup routine
initcode reset
listfile        segments publics by addres      // Generate a segment map

map     0x00000 to 0xfbfff as reserved
map     0xfc000 to 0xfefff as rdonly

class   CODE = 0xfc00
output  CODE
```

リストA-26　MAKESTA

```
#############################################################################
#       Sample NMAKE makefile for building an embedded system using      #
#       Microsoft C 6.0                                                  #
#############################################################################
STARTUP =       startup
CONFIG  =       config

AFLAGS  =       /mx
LFLAGS  =       /MAP /CO
CFLAGS  =       /c /AL /Gm /Zi /Gs

OBJS    =       $(STARTUP).obj $(CONFIG).obj

#################### dependancies ##################
.c.obj:
        cl $(CFLAGS) $*.c

.asm.obj:
        masm $(AFLAGS) $* ;

$(STARTUP).hex: $(STARTUP).rom startup.rm
        locate -He -cstartup.rm $*

$(STARTUP).rom: $(OBJS)
        @echo $(OBJS)
        link $(LFLAGS) $(OBJS),$(STARTUP).rom;
```

付録B　RX423用コーディング・リスト

B．1　タスク部

リストB-1　MAIN.C（1/5）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : main.c
TARGET          : V55PI
PROCEDURES DEFINED      :       init_task(),
                                rs_init(),
                                obj_init(),
                                free_mbx_init(),
                                task_init(),
                                sysdata_init();

*****************************************************************************/

#include        "led.h"

struct  _STRUCT_SFR     _SFRSEP;

void    init_task( void );      /* --------- 初期タスク ---------- */
                                /* 各オブジェクト，タスクの生成，   */
                                /* システム情報の初期化，           */
                                /* 割り込みハンドラの定義をする     */
                                /* ------------------------------ */

void    rs_init( void );        /* シリアル受信割り込みハンドラ定義 */
void    obj_init( void );       /* メモリ・プール#1，メールボックス，フラグ，
                                   セマフォの生成 */
void    free_mbx_init( void );  /* フリー・メールボックスの生成 */
void    task_init( void );      /* タスクの生成，起動 */
void    sysdata_init( void );   /* システム情報の初期化 */

/****  アクセス・アドレス  ****/
short   time_task_aa,           /* 時刻タスク                       */
        led_task_aa,            /* LEDドライブ・タスク              */
        mesg_task_aa,           /* メッセージ・タスク               */
        recv_task_aa,           /* 受信データ処理タスク             */
        ctrl_task_aa,           /* コントロール・タスク             */
        shift_task_aa,          /* シフト・タスク                   */
        mpl1_aa,                /* メモリ・プール#1                 */
        news_mbx_aa,            /* ニュース・メールボックス         */
        cm_mbx_aa,              /* コマーシャル・メールボックス     */
        emg_mbx_aa,             /* 緊急メールボックス               */
        led_mbx_aa,             /* LED表示メールボックス            */
        ctrl_mbx_aa,            /* コマンド・メールボックス         */
        free_mesg_mbx_aa,       /* 表示データ用フリー・メールボックス */
        free_ctrl_mbx_aa,       /* コマンド用フリー・メールボックス */
        free_led_mbx_aa,        /* LED表示用フリー・メールボックス  */
        flag_aa,                /* 時刻表示タイミング・イベント・フラグ */
        sys_sem_aa,             /* システム情報排他制御用セマフォ   */
```

リストB-1　MAIN.C（2/5）

```
        msg_sem_aa;             /* 表示データ排他制御用セマフォ          */

struct  SYSTEM  sys;            /* システム情報 */
struct  RINGBUF RingBuf;        /* リング・バッファ */

/* ******************** *
 *      初期タスク      *
 * ******************** */
void    init_task( void )
{
        _DI();                  /* 割り込み禁止 */

        eb_init();              /* ハードウエアの初期化 */

        rcopy( romidata );      /* 初期値あり変数の初期化 */

        rs_init();              /* シリアル受信バッファの初期化 */

        _EI();                  /* 割り込み許可 */

        obj_init();             /* 同期通信用資源の初期化 */

        free_mbx_init();        /* フリー・メールボックスの初期化 */

        task_init();            /* タスクの初期化 */

        sysdata_init();         /* システム情報の初期化 */

        ena_int( INT_RS );      /* 指定デバイス番号の割り込みレベルを
                                   割り込み許可にする */

        exd_tsk();              /* 自タスクの終了，削除 */
}

/* ****************************************** *
 *      シリアル受信割り込みハンドラ定義      *
 * ****************************************** */
void  rs_init( void )
{
        struct  DEF_INT pk_def_int;     /* def_int用構造体 */

        buf_init();             /* リング・バッファの初期化 */

        pk_def_int.inthdr = (void (*)())rs_inthand;
                /* 定義する割り込みハンドラのスタート・アドレス */

        pk_def_int.intds  = INT_DS;
                /* 定義する割り込みハンドラで使用するDS0値 */

        def_int( INT_RS, &pk_def_int );
                /* シリアル受信割り込みハンドラ定義 */

        _MK0L &= ~0x08;
        _INTM0 = 0x55;
        _IC11  = 0;

        /***** mask nmi with hardware *****/
        *(char *)PIU = MASK_NMI;
}
```

リストB-1　MAIN.C（3/5）

```
/* ********************************************************************** *
 *      メモリ・プール#1，メールボックス，フラグ，セマフォの生成          *
 * ********************************************************************** */
void    obj_init( void )
{
        struct  CRE_MPL p_cre_mpl;      /* cre_mpl用構造体 */

        p_cre_mpl.mplsz = MPL_SIZ;      /* 生成するメモリ・プール・サイズ */

        p_cre_mpl.blksz  = BLK_SIZ;     /* 生成するメモリ・プールからメモリ・
                                           ブロックを切り出す最小単位 */

        cre_mpl( T_TFIFO, &mpl1_aa, MPL1_ID, &p_cre_mpl );
                /* メモリ・プール#1生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &news_mbx_aa, NEWS_MBX_ID );
                /* ニュース・メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &emg_mbx_aa, EMG_MBX_ID );
                /* 緊急メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &cm_mbx_aa, CM_MBX_ID );
                /* コマーシャル・メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &led_mbx_aa, LED_MBX_ID );
                /* LED表示メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &ctrl_mbx_aa, CTRL_MBX_ID );
                /* コマンド・メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &free_mesg_mbx_aa, FREE_MESG_MBX_ID );
                /* 表示データ用フリー・メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &free_ctrl_mbx_aa, FREE_CTRL_MBX_ID );
                /* コマンド用フリー・メールボックス生成 */

        cre_mbx( T_MFIFO | T_TFIFO, &free_led_mbx_aa, FREE_LED_MBX_ID );
                /* LED表示用フリー・メールボックス生成 */

        cre_flg( &flag_aa, FLAG_ID );
                /* 時刻表示タイミング・イベント・フラグ生成 */

        cre_sem( T_TFIFO, &msg_sem_aa, MSG_SEM_ID, MSG_SEM_MAX );
                /* 表示データ排他制御用セマフォ生成 */

        cre_sem( T_TFIFO, &sys_sem_aa, SYS_SEM_ID, SYS_SEM_MAX );
                /* システム情報排他制御用セマフォ生成 */
}

/* ********************************** *
 *      フリー・メールボックス初期化    *
 * ********************************** */
void    free_mbx_init( void )
{
        struct NEWS     *news_blkaa;    /* 表示データ用フリー・メールボックスに送
                                           信する未使用メッセージのアクセス・
                                           アドレス */

        struct COMMAND  *com_blkaa;     /* コマンド用フリー・メールボックスに送信
```

リストB-1　MAIN.C（4/5）

```
                                        する未使用メッセージのアクセス・
                                        アドレス */

        struct LED      *led_blkaa;     /* LED表示用フリー・メールボックスに送信
                                        する未使用メッセージのアクセス・
                                        アドレス */
        int     i;

        /* 表示データ用フリー・メールボックスの初期化 */
        for( i = 0; i < FREE_MESG_MAX; i++ ) {
                get_blk( T_NOOPT, &news_blkaa, mpl1_aa, BLKCNT );
                news_blkaa->free_mbxaa = free_mesg_mbx_aa;
                snd_msg( free_mesg_mbx_aa, news_blkaa );
        }

        /* コマンド用フリー・メールボックスの初期化 */
        for( i = 0; i < FREE_CTRL_MAX; i++ ) {
                get_blk( T_NOOPT, &com_blkaa, mpl1_aa, BLKCNT );
                com_blkaa->free_mbxaa = free_ctrl_mbx_aa;
                snd_msg( free_ctrl_mbx_aa, com_blkaa );
        }

        /* LED表示用フリー・メールボックスの初期化 */
        for( i = 0; i < FREE_LED_MAX; i++ ) {
                get_blk( T_NOOPT, &led_blkaa, mpl1_aa, BLKCNT );
                led_blkaa->free_mbxaa = free_led_mbx_aa;
                snd_msg( free_led_mbx_aa, led_blkaa );
        }
}

/* ********************************** *
 *      タスクの生成，起動              *
 * ********************************** */
void    task_init( void )
{
        struct  CRE_TSK pk_cre_tsk;             /* cre_tsk用構造体 */

        pk_cre_tsk.sta          = time_task;    /* タスク・スタート・アドレス*/

        pk_cre_tsk.stksz        = STACK_SIZ;    /* タスクで使用するユーザ・
                                                   スタック・サイズ */

        pk_cre_tsk.tskpri       = TIME_TASK_PRI; /* タスクの初期優先度 */

        pk_cre_tsk.tskds        = TASK_DS;      /* タスクで使用するデータ・
                                                   セグメントの値 */

        cre_tsk( &time_task_aa, TIME_TASK_ID, &pk_cre_tsk );
                /* 時刻タスク生成 */

        pk_cre_tsk.sta          = recv_task;
        pk_cre_tsk.stksz        = STACK_SIZ;
        pk_cre_tsk.tskpri       = RECV_TASK_PRI;
        pk_cre_tsk.tskds        = TASK_DS;

        cre_tsk( &recv_task_aa, RECV_TASK_ID, &pk_cre_tsk );
                /* 受信割り込み処理タスク生成 */

        pk_cre_tsk.sta          = ctrl_task;
```

リストB-1　MAIN.C（5/5）

```
        pk_cre_tsk.stksz        = STACK_SIZ;
        pk_cre_tsk.tskpri       = CTRL_TASK_PRI;
        pk_cre_tsk.tskds        = TASK_DS;

        cre_tsk( &ctrl_task_aa, CTRL_TASK_ID, &pk_cre_tsk );
                /* コントロール・タスク生成 */

        pk_cre_tsk.sta          = led_task;
        pk_cre_tsk.stksz        = STACK_SIZ;
        pk_cre_tsk.tskpri       = LED_TASK_PRI;
        pk_cre_tsk.tskds        = TASK_DS;

        cre_tsk( &led_task_aa, LED_TASK_ID, &pk_cre_tsk );
                /* LEDドライブ・タスク生成 */

        pk_cre_tsk.sta          = mesg_task;
        pk_cre_tsk.stksz        = STACK_SIZ;
        pk_cre_tsk.tskpri       = MESG_TASK_PRI;
        pk_cre_tsk.tskds        = TASK_DS;

        cre_tsk( &mesg_task_aa, MESG_TASK_ID, &pk_cre_tsk );
                /* メッセージ・タスク生成 */

        pk_cre_tsk.sta          = shift_task;
        pk_cre_tsk.stksz        = STACK_SIZ;
        pk_cre_tsk.tskpri       = SHIFT_TASK_PRI;
        pk_cre_tsk.tskds        = TASK_DS;

        cre_tsk( &shift_task_aa, SHIFT_TASK_ID, &pk_cre_tsk );
                /* シフト・タスク生成 */

        sta_tsk( led_task_aa );         /* LEDドライブ・タスク・スタート */
        sta_tsk( mesg_task_aa );        /* メッセージ・タスク・スタート */
        sta_tsk( shift_task_aa );       /* シフト・タスク・スタート */
        sta_tsk( time_task_aa );        /* 時刻タスク・スタート */
        sta_tsk( recv_task_aa );        /* 受信割り込み処理タスク・スタート */
        sta_tsk( ctrl_task_aa );        /* コントロール・タスク・スタート */
}

/* ********************************************************************** *
 *      システム時刻の初期化，モード設定，時間変更用フラグの初期化       *
 * ********************************************************************** */
void    sysdata_init( void )
{
        wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

        sys.time.curnt_min = SYS_TIME_INIT_MIN;         /* システム時刻初期化*/
        sys.time.curnt_hour = SYS_TIME_INIT_HOUR;

        sys.new_time_flag = RESET;                      /* システム時刻変更用フ
                                                           ラグ・リセット*/

        sys.curnt_mode = MIX_MODE;                      /* 混在モード */

        sig_sem( sys_sem_aa, SYS_SEM_COUNT );
}

/* dummy for InterTools runtime library */
main()
{
}
```

リストB-2　CTRL.C（1/4）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : ctrl.c
TARGET          : V55PI
PROCEDURES DEFINED      :       ctrl_task(),
                                unlink_and_send(),
                                del_mesg();

*****************************************************************************/

#include        "led.h"

void    ctrl_task( void );      /* -------- コントロール・タスク -------- */
                                /* 受信データ処理タスクから送られたコマン */
                                /* ド・メッセージを受け取り，表示データの */
                                /* 追加，削除，表示モードの変更，表示デー */
                                /* タの参照をする。                       */
                                /* -------------------------------------- */

void    unlink_and_send( unsigned char );       /* 表示データの参照 */
void    del_mesg( unsigned char );              /* 表示データの削除 */

struct NEWS     *dummy_msgaa;                   /* マーキング・メッセージ */

/* ************************** *
 *     コントロール・タスク     *
 * ************************** */
void    ctrl_task( void )
{
        struct  COMMAND *cmd_msgaa;     /* コマンド・メッセージ・アクセス
                                           アドレス */

        get_blk( T_NOOPT, &dummy_msgaa, mpl1_aa, BLKCNT );
                /* 表示マーキング・メッセージ用のメモリ・ブロックを獲得する */

        dummy_msgaa->msg_id = MARKING_ID;
                /* マーキングIDを登録する */

        while (1) {

                rcv_msg( T_NOOPT, &cmd_msgaa, ctrl_mbx_aa );
                        /* コマンド・メッセージ受信 */

                switch ( cmd_msgaa->command ) {
                        /* コマンド・メッセージのコマンドを見て，
                           それぞれの処理をする */

                        case DEL_MESG:          /* 表示データ削除 */
                                del_mesg( cmd_msgaa->mode );
                                break;

                        case CHANGE_MODE:       /* 表示モード変更 */
                                wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

                                sys.curnt_mode = cmd_msgaa->mode;
                                        /* 表示モードを，コマンド・メッセージの
                                           モードに変更 */
```

リストB-2　CTRL.C（2/4）

```
                        sig_sem( sys_sem_aa, SYS_SEM_COUNT );

                        break;

                    case LOOK_MESG:         /* 表示データの参照 */

                        unlink_and_send( cmd_msgaa->mode );
                        break;
                }
                snd_msg( cmd_msgaa->free_mbxaa, cmd_msgaa );
                    /* マーキング・メッセージをフリー・メールボックスに返却*/
        }
}

/* ************************************************** *
 *      関数名 : unlink_and_send()                    *
 *      機能   : PCへ，表示データを送信する           *
 *      返却値 : なし                                 *
 * ************************************************** */
void    unlink_and_send( unsigned char mode )/* コマンド・メッセージのモード */
{
        int             mbxaa;          /* 表示データ・メッセージを受け取るメー
                                           ルボックスのアクセス・アドレス */

        struct NEWS     *mesgaa;        /* 表示データ・メッセージのアクセス・ア
                                           ドレス*/

        char            put_buf[MSGSIZ];/* PCへの送信用バッファ */


        switch ( mode ) {

                case TIME_MODE:         /* 時刻モード */
                    wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

                    sprintf( put_buf, "T%02d:%02d%c",
                             sys.time.curnt_hour, sys.time.curnt_min, END);
                            /* PCへの送信メッセージを作成 */

                    sig_sem( sys_sem_aa, SYS_SEM_COUNT );

                    rputs( put_buf );       /* PCへ送信 */

                    break;

                case NEWS_MODE:
                case CM_MODE:           /* ニュース，コマーシャル，緊急モード*/
                case EMG_MODE:
                    switch ( mode ) {
                        /* モードによって参照するメールボックス・アクセス・
                           アドレスをセットする */

                        case NEWS_MODE:
                                mbxaa = news_mbx_aa;
                                break;

                        case CM_MODE:
                                mbxaa = cm_mbx_aa;
                                break;
```

リストB-2　CTRL.C（3/4）

```
                case EMG_MODE:
                        mbxaa = emg_mbx_aa;
                        break;
        }

        wai_sem( T_NOOPT, msg_sem_aa, MSG_SEM_COUNT );

        snd_msg( mbxaa, dummy_msgaa );
                /* マーキング・メッセージ送信 */

        do {
                rcv_msg( T_NOOPT, &mesgaa, mbxaa );
                        /* 表示データ・メッセージ受信 */

                if ( mesgaa->msg_id != MARKING_ID ) {
                        /* 表示データ・メッセージがマーキング・
                           メッセージでない場合 */

                        sprintf( put_buf, "%c%s%c",
                                STX, mesgaa->nmsg, ETX );
                                /* PCへの送信メッセージ作成 */

                        snd_msg( mbxaa, mesgaa );
                                /* 表示データ・メッセージを元の
                                   メールボックスへ返す */

                        rputs( put_buf );  /* PCへ送信 */

                }

        } while ( mesgaa->msg_id != MARKING_ID );
                /* マーキング・メッセージを受け取るまで */

        sig_sem( msg_sem_aa, MSG_SEM_COUNT );

        sprintf( put_buf, "%c", END );
                /* 表示データ終了メッセージ作成 */

        rputs( put_buf ); /* PCへ送信 */

        break;
    }
}

/* ************************************************** *
 *      関数名 : del_mesg()                           *
 *      機能   : 登録メッセージを削除する             *
 *      返却値 : なし                                 *
 * ************************************************** */
void    del_mesg( unsigned char mode )  /* コマンド・メッセージのモード */
{
        int             mbxaa,                  /* 削除したい表示データ・メッセ
                                                   ージを受け取るメールボックス
                                                   のアクセス・アドレス */

                        err_rcv;                /* rcv_msgの返却値 */

        long            time_to_wait0 = 0;      /* タイム・アウト指定時間 */
```

リストB-2　CTRL.C（4/4）

```
        struct  NEWS    *delete_mesgaa;         /* 表示データ・メッセージのアク
                                                   セス・アドレス */

        switch ( mode ) {       /* モードによって，表示データ・メッセージを削除
                                   するメールボックスをセットする */

                case NEWS_MODE:
                        mbxaa = news_mbx_aa;
                        break;

                case CM_MODE:
                        mbxaa = cm_mbx_aa;
                        break;

                case EMG_MODE:
                        mbxaa = emg_mbx_aa;
                        break;
        }

        wai_sem( T_NOOPT, msg_sem_aa, MSG_SEM_COUNT );

        do {
                err_rcv = rcv_msg( T_TMOUT, &delete_mesgaa, mbxaa,
                                        &time_to_wait0 );
                        /* 即時リターンで，表示データ・メッセージを受け取る */

                if ( err_rcv == TE_OK ) {       /* メッセージを受け取ることがで
                                                   きた場合 */

                        snd_msg( delete_mesgaa->free_mbxaa, delete_mesgaa );
                                /* 表示データ用フリー・メールボックスに表示データ・
                                   メッセージを送信する */
                }
        } while ( err_rcv == TE_OK );
                /* 表示データ・メッセージを受け取れている間 */

        sig_sem( msg_sem_aa, MSG_SEM_COUNT );

}
```

リストB-3　LED.C（1/2）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : led.c
TARGET          : V55PI
PROCEDURES DEFINED      :       led_task(),
                                init_71055(),

*****************************************************************************/

#include        "led.h"

void    led_task( void );       /* ---------  LEDドライブ・タスク  --------- */
                                /* LED表示バッファにセットされたフォント・デ */
                                /* ータのビットに対応するLEDアレイの点灯をす */
                                /* る。                                      */
                                /* ----------------------------------------- */

void    init_71055( void );     /* uPD71055の初期化 */

/* ***************************************** *
 *      関数名  :  init_71055()                *
 *      機能    :  uPD71055の初期化            *
 *      返却値  :  なし                        *
 * ***************************************** */
void    init_71055( void )
{
        outpw( CTRL_PORT_1_2, INIT_PIU );       /* モード0, すべてのポートを出力
                                                   にする */

        outpw( CTRL_PORT_3_4, INIT_PIU );       /* モード0, すべてのポートを出力
                                                   にする */
}

/* ********************************** *
 *      LEDドライブ・タスク             *
 * ********************************** */
void    led_task( void )
{
        static  int     cnt = 0;                /* LED表示バッファを参照するた
                                                   めのカウンタ */

        long            time_to_wait1 = 1;      /* LED表示間隔 */

        init_71055();
                /* パラレル・インタフェース・ユニット（uPD71055）の初期化 */

        while (1) {
                outpw( PIU_PORT,    fdata[0][cnt] );
                outpw( PIU_PORT+2,  fdata[1][cnt] );
                outpw( PIU_PORT+8,  fdata[2][cnt] );
                outpw( PIU_PORT+10, fdata[3][cnt] );
                        /* 64ビット（横1列分）の表示データをLEDアレイにセット */
                outpw( PIU_PORT+10, fdata[3][cnt] );

                outpw( PIU_PORT+4,  cnt );
                        /* 表示する列番号をセットする */
```

リストB-3　LED.C（2/2）

```
                wai_tsk( &time_to_wait1 );
                        /* LED表示間隔 */

                cnt = ((cnt+1) & 0xf);
                        /* 0xfと論理積をとり，0から15までの値を順番にインク
                           リメントさせていく（15の次は0になる） */
        }
}
```

リストB-4　MESG.C（1/6）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : mesg.c
TARGET          : V55PI
PROCEDURES DEFINED      :       mesg_task(),
                                send_news_to_led(),
                                send_emg_to_led(),
                                send_time_to_led();

*****************************************************************************/

#include        "led.h"

void    mesg_task( void );              /* --------------------------------*/
                                        /* LED表示用メールボックスからメッ  */
                                        /* セージを受け取り，表示モードに対 */
                                        /* 応するメールボックスからメッセー */
                                        /* ジを受け取ってLED表示用メッセー  */
                                        /* ジにコピーし，LEDメールボックス  */
                                        /* に送信する                       */
                                        /* -------------------------------- */

void    send_news_to_led( int, struct LED * );
        /* ニュース・メッセージをLEDメールボックスに送信する */

int     send_emg_to_led( struct LED * );
        /* 緊急メッセージをLEDメールボックスに送信する */

int     send_time_to_led( struct LED * );
        /* 時刻メッセージを作成し，LEDメールボックスに送信する */

/* ***************************** *
 *      メッセージ・タスク       *
 * ***************************** */
void    mesg_task( void )
{
        struct LED      *led_msgaa;     /* LEDメッセージ・アクセス・アドレス */
int             err_time,       /* 時刻メッセージ送信ルーチンの返却値 */
        err_emg;        /* 現在のイベント・フラグの内容 */
        static int      mode_turn = 0;  /* モード変更用（混在モード） */

        while (1) {
                rcv_msg( T_NOOPT, &led_msgaa, free_led_mbx_aa );
                        /* LED表示用メッセージをフリー・メールボックスから受け取
                           る（無限待ち） */

                err_emg = send_emg_to_led( led_msgaa );
                        /* 時刻メッセージ送信 */

                if ( err_emg == SEND_OK ) {     /* 時刻メッセージ送信終了 */
                        continue;               /* while(1)まで戻る */
                }

                wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

                switch ( sys.curnt_mode ) {
                        case MIX_MODE:          /* 混在モード */
```

リストB-4　MESG.C（2/6）

```
        sig_sem( sys_sem_aa, SYS_SEM_COUNT );

        err_time = send_time_to_led( led_msgaa );
                /* 時刻メッセージ送信 */

        if ( err_time == SEND_OK ) {
                /* 時刻メッセージ送信終了 */
                continue;
                        /* while(1)まで戻る */
        }

        if ( (mode_turn & 1) == 0 ) {
                /* mode_turnと1の論理積が1か0かによって，
                   送信するメッセージをどちらにするかを
                   決める */

                send_news_to_led( news_mbx_aa,
                                        led_msgaa );
                        /* ニュース・メッセージ送信 */
        } else {
                send_news_to_led(cm_mbx_aa, led_msgaa);
                        /* コマーシャル・メッセージ送信
                          */
        }
        mode_turn++;

        break;

case CM_MODE:           /* コマーシャル・モード */

        sig_sem( sys_sem_aa, SYS_SEM_COUNT );

        send_news_to_led( cm_mbx_aa, led_msgaa );
                /* コマーシャル・メッセージ送信 */
        break;

case NEWS_MODE:         /* ニュース・モード */

        sig_sem( sys_sem_aa, SYS_SEM_COUNT );

        send_news_to_led( news_mbx_aa, led_msgaa );
                /* ニュース・メッセージ送信 */
        break;

case EMG_MODE:          /* 緊急モード */
        sig_sem( sys_sem_aa, SYS_SEM_COUNT );

        snd_msg( led_msgaa->free_mbxaa, led_msgaa );
                /* フリー・メールボックスに，LED表示メッセー
                   ジを送信 */

        break;

case TIME_MODE:         /* 時刻モード */

        sig_sem( sys_sem_aa, SYS_SEM_COUNT );

        err_time = send_time_to_led( led_msgaa );
                /* 時刻メッセージ送信 */
```

リストB-4　MESG.C（3/6）

```
                        if ( err_time == SEND_OK ) {
                                /* 時刻メッセージ送信終了 */

                                continue;
                                        /* while(1)まで戻る */
                        }

                        snd_msg( led_msgaa->free_mbxaa, led_msgaa );
                                /* フリー・メールボックスに，LED表示メッセー
                                   ジ送信 */
                        break;
                }
        }
}

/* ************************************************************************ *
 *      関数名   :  send_emg_to_led()                                       *
 *      機能     :  緊急メッセージ用のメールボックスからメッセージを１つ     *
 *                  取り出し，LED表示用のメールボックスに送信する           *
 *      返却値   :  < 0> 送信正常終了                                       *
 *                  <-1> 送信せず（メッセージが受け取れなかったため）       *
 * ************************************************************************ */
int     send_emg_to_led( led_msgaa )
struct LED      *led_msgaa;
{                               /* LED表示メッセージ・アクセス・アドレス */

        int             err_emg;                /* rcv_msgの返却値 */

        struct NEWS     *emg_msgaa;             /* 緊急メッセージのアクセス・ア
                                                   ドレス */

        int             prptn;                  /* 現在のフラグの内容 */

        long            time_to_wait0 = 0;      /* rcv_msg（即時リターン）のタ
                                                   イム・アウト時間 */

        wai_sem( T_NOOPT, msg_sem_aa, MSG_SEM_COUNT );

        err_emg = rcv_msg( T_TMOUT, &emg_msgaa, emg_mbx_aa, &time_to_wait0 );
                /* 緊急メールボックスから，メッセージを受け取る */

        if ( err_emg != TE_OK ) {
                /* 緊急メッセージが受け取れなかった（メッセージがない） */

                sig_sem( msg_sem_aa, MSG_SEM_COUNT );

                return( NOT_SEND );     /* 異常終了 */
        }

        /* メッセージが受け取れた場合 */

        (emg_msgaa->left_count)--;
                /* 緊急メッセージの表示回数をデクリメント */

        strcpy( led_msgaa->lmsg, emg_msgaa->nmsg );
                /* LED表示メッセージに，緊急メッセージをコピー */

        if ( emg_msgaa->left_count > 0 ) {
                /* 緊急メッセージの表示回数が0より多い場合 */
```

リストB-4　MESG.C（4/6）

```
                snd_msg( emg_mbx_aa, emg_msgaa );
                        /* 緊急メールボックスに返却（送信）*/
        } else {
                /* 表示回数が0の場合 */

                snd_msg( emg_msgaa->free_mbxaa, emg_msgaa );
                        /* フリー・メールボックスに，緊急メッセージを返却（送信）
         */
        }

        sig_sem( msg_sem_aa, MSG_SEM_COUNT );

        snd_msg( led_mbx_aa, led_msgaa );
                /* LED 表示メッセージ（緊急メッセージ）を，LED メールボックスに
   送信 */

        set_flg( T_REP, &prptn, flag_aa, SETPTN0 );
                /* 時刻表示タイミング・イベント・フラグのビット0に1をセット */

        return( SEND_OK );
                /* 正常終了 */

}

/* ********************************************************************** *
 *      関数名  :  send_time_to_led()                                     *
 *      機能    :  時刻変更のフラグの状態を見て，セットされていれば       *
 *                 現在時刻のメッセージを作り，LED表示用のメール          *
 *                 ボックスに送信する                                     *
 *      返却値  :  < 0> 送信正常終了                                      *
 *                 <-1> 送信せず（メッセージが受け取れなかった）          *
 * ********************************************************************** */
int     send_time_to_led( led_msgaa )
struct LED      *led_msgaa;     /* LED表示メッセージ・アクセス・アドレス */
{
        unsigned int    *min,                   /* システム時刻（分）を変更する
                                                   ためのポインタ */

                        *hour;                  /* システム時刻（時）を変更する
                                                   ためのポインタ */

        int             prptn,                  /* 現在のイベント・フラグの内容
                                                    */

                        err_flg;                /* wai_flgの返却値 */

        long            time_to_wait0 = 0;      /* wai_flgのタイム・アウト値 */

        min = &sys.time.curnt_min;              /* システム時刻（分） */
        hour = &sys.time.curnt_hour;            /* システム時刻（時） */

        err_flg = wai_flg( T_ORW | T_RESET | T_TMOUT, &prptn, flag_aa, SETPTN1,
                                        &time_to_wait0 );
                /* 時刻表示タイミング・イベント・フラグのビット0に1がセットされ
                   ているか */

        if ( err_flg != TE_OK ) {
                /* セットされていない */
                return( NOT_SEND );     /* メッセージ送信せず */
```

リストB-4　MESG.C（5/6）

```
        }

        wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

        sprintf( led_msgaa->lmsg, "ただ今の時刻は %02d 時 %02d 分です", *hour, *min );
                /* 時刻メッセージを作成 */

        sig_sem( sys_sem_aa, SYS_SEM_COUNT );

        snd_msg( led_mbx_aa, led_msgaa );
                /* LED表示メッセージ（時刻メッセージ）を，LEDメールボックスに
                   送信 */

        return( SEND_OK );      /* 正常終了（送信終了） */

}

/* ********************************************************************** *
 *      関数名  :  send_news_to_led()                                     *
 *      機能    :  ニュース，コマーシャルのメールボックスからメッセージ *
 *                 を１つ取り出し，LED表示用のメールボックスに送信する  *
 *      返却値  :  なし                                                   *
 * ********************************************************************** */
void    send_news_to_led( mbxaa, led_msgaa )
int             mbxaa;          /* メールボックス・アクセス・アドレス */
struct LED      *led_msgaa;     /* LED表示メッセージ・アクセス・アドレス */
{
        int             err_news;       /* rcv_msgの返却値 */

        struct  NEWS    *p_news;        /* ニュース（コマーシャル）のメッセージ・
                                           アクセス・アドレス */

        long            time_to_wait0 = 0;
                                        /* rcv_msgのタイム・アウト値 */

        wai_sem( T_NOOPT, msg_sem_aa, MSG_SEM_COUNT );

        err_news = rcv_msg( T_TMOUT, &p_news, mbxaa, &time_to_wait0 );
                /* メールボックス（ニュースまたはコマーシャル）から，メッセージを
                   受け取る */

        if ( err_news == TE_OK ) {      /* メッセージ受け取り成功 */

                (p_news->left_count)--;
                        /* メッセージの表示回数をデクリメント */

                strcpy( led_msgaa->lmsg, p_news->nmsg );
                        /* LED表示メッセージに，ニュース（コマーシャル）メッセー
                           ジをコピー */

                if ( p_news->left_count > 0 ) {
                        /* 表示回数が0より多い場合 */

                        snd_msg( mbxaa, p_news );
                                /* ニュース（コマーシャル）メールボックスに，
                                   ニュース（コマーシャル）メッセージを返却
                                   （送信） */
                } else {
```

リストB-4　MESG.C（6/6）

```
                snd_msg( p_news->free_mbxaa, p_news );
                        /* フリー・メールボックスに，ニュース（コマーシャル）
                           メッセージを返却（送信） */
            }

            sig_sem( msg_sem_aa, MSG_SEM_COUNT );

            snd_msg( led_mbx_aa, led_msgaa );
                    /* LED表示メッセージ（ニュースまたはコマーシャル・メッセ
                       ージ）を，LEDメールボックスに送信 */

        } else {        /* ニュース（コマーシャル）メッセージが受け取れない
                           （メッセージがない） */

            sig_sem( msg_sem_aa, MSG_SEM_COUNT );

            snd_msg( led_msgaa->free_mbxaa, led_msgaa );
                    /* フリー・メールボックスに，LED表示メッセージを返却
                       （送信） */
        }
}
```

リストB-5　RECV.C（1/7）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : recv.c
TARGET          : V55PI
PROCEDURES DEFINED      :       recv_task(),
                                buf_read(),
                                get_msg_data(),
                                new_entry(),
                                new_time_set(),
                                get_command_data(),
                                copy_command();

*****************************************************************************/

#include        "led.h"

void    recv_task( void );              /* ----- 受信データ処理タスク  ----- */
                                        /* 受信割り込みハンドラからiret_wup  */
                                        /* で起こされ，リング・バッファの文  */
                                        /* 字を読み出し，コマンド・データ，  */
                                        /* 表示データの送信，時刻の変更を行う*/
                                        /* ------------------------------- */

int     buf_read( void );               /* PCから送信されたコマンド・データや表
                                           示データを読み出す  */

void    get_msg_data( char *, int );    /* リング・バッファから表示データを読み
                                           出す */

void    new_entry( char *, struct RCV_COM * );
        /* 時刻変更または各メールボックスに表示データを送信する */

int     new_time_set( char *, unsigned int *, unsigned int * );
        /* 時刻の変更をする */

void    get_command_data( struct RCV_COM * );
        /* リング・バッファからコマンド・データを読み出す */

/* ************************************ *
 *      受信タスク                      *
 * ************************************ */
void    recv_task( void )
{
        while (1) {
                slp_tsk();      /* 受信割り込みハンドラから，iret_wupで起こさ
                                   れる */

                buf_read();     /* データ読み出し */
        }
}

/* ******************************************************************** *
 *      関数名  :  buf_read()                                           *
 *      機能    :  PCから送信されたコマンドや表示メッセージを読み出す   *
 *      返却値  :  なし                                                 *
 * ******************************************************************** */
int     buf_read( void )
```

リストB-5　RECV.C（2/7）

```
{
        int                     lrp,            /* ETXサーチ用ポインタ */

                                datasiz,        /* 表示データ文字数 */

                                dnum;           /* リング・バッファ
                                                   書き込み文字数 */

        char                    msg[MSGSIZ];    /* 表示データ格納バッファ */

        struct  COMMAND         *cmdaa;         /* コマンド・メッセージ・
                                                   アドレス */

        static  struct  RCV_COM cmd;            /* 受信したコマンド・データを
                                                   格納する構造体 */

        while (1) {     /* PCから送信されるデータのヘッダ'P'または'STX'が出て
                           来るまで，リング・バッファから文字を取り出し続ける
          */

                while ( RingBuf.buf[RingBuf.rp] != 'P' &&
                        RingBuf.buf[RingBuf.rp] != STX ) {
                        /* 書き込みポインタの文字が'P'または'STX'で
                           ない場合 */

                        if ( rgetc() == D_EMPTY ) {
                                /* リング・バッファから文字を取り出す */

                                return;
                                /* リング・バッファが空になったら，リターン */
                        }
                }

                dnum = RingBuf.dnum;    /* リング・バッファの書き込み文字数を保
                                           存しておく */

                switch ( RingBuf.buf[RingBuf.rp] ) {
                        case 'P':       /* コマンド・データのヘッダ */

                                if ( dnum < sizeof(struct RCV_COM) ) {
                                        /* コマンド・データの文字数が少ない */

                                        return;
                                }

                                get_command_data( &cmd );
                                        /* リング・バッファからコマンド・データ
                                           を取り出す */

                                if ( cmd.command == ADD_MESG ) {
                                        /* 表示データ追加コマンド */

                                        return;
                                }

                                rcv_msg( T_NOOPT, &cmdaa, free_ctrl_mbx_aa );
                                        /* フリー・メールボックスから，コマンド・メ
                                           ッセージ用のフリー・メッセージを受け取
                                           る（無限待ち） */
```

リストB-5　RECV.C（3/7）

```
                cmdaa->command = cmd.command;
                        /* コマンドをコピーする */

                cmdaa->mode = cmd.mode;
                        /* モードをコピーする */

                snd_msg( ctrl_mbx_aa, cmdaa );
                        /* コントロール・タスクにコマンド・メッ
                           セージを送信する */

                break;

        case STX:       /* 表示データのヘッダ */

                if ( dnum == BUF_SIZ ) {
                        rgetc();
                        return;
                        /* 表示データがリング・バッファ・サイズ
                           を越えている場合は，リターン */
                }

                for ( lrp = 0; dnum > lrp; lrp++ ) {
                        if( RingBuf.buf[(RingBuf.rp+lrp)
                                & BUF_MASK] == ETX ) {
                                break;
                        }
                }
                        /* 表示データの文字数を数える */
                        /* ETXが現れるまで，lprをインクリメン
                           トする */

                if ( dnum == lrp ) {
                        return; /* ETXが見つからなかった */
                }

                datasiz = lrp-1;
                        /* 表示データの文字数＝lrp-1 */

                get_msg_data( msg, datasiz );
                        /* リング・バッファから表示データを取り
                           出す */

                if ( cmd.command == ADD_MESG ) {
                        if ( datasiz > 200 ) {
                                return;
                                /* 表示データが200文字を越えた
                                   場合は，データを無効にする */
                        }

                        new_entry( msg, &cmd );
                                /* 表示データをメールボックスに
                                   送信する */
                }
        break;
        }
    }
}
```

リストB-5　RECV.C（4/7）

```
/* ********************************************************************** *
 *      関数名  :  get_command_data()                                     *
 *      機能    :  リング・バッファから指定文字数，データを取り出す       *
 *      返却値  :  なし                                                   *
 * ********************************************************************** */
void    get_msg_data( p, bsiz )
char    *p;     /* 取り出した表示データを格納するバッファへのポインタ */
int     bsiz;   /* 表示データのバイト数 */
{
        int     i;

        rgetc();        /* ヘッダのSTXを取り出す */

        for ( i = 0; i < bsiz; i++ ) {
                /* 表示データを取り出し，配列に入れる */

                *(p+i) = rgetc();
        }

        *(p+i) = (char)NULL;    /* 配列の終わり */
        rgetc();                /* ETXを取り出す */
}

void    get_command_data( p )
struct RCV_COM  *p;
{
        int     i;

        p->header = (unsigned char)rgetc();     /* ヘッダ */
        p->command = (unsigned char)rgetc();    /* コマンド */
        p->mode = (unsigned char)rgetc();       /* モード */

        for ( i = 0; i < 4; i++ ) {
                rgetc();        /* 送信されたメッセージIDは無視する */

        }

        for ( i = 0; i < 4; i++ ) {
                p->left_cnt[i] = (unsigned char)rgetc();  /* 表示回数 */
        }
}

/* ********************************************************************** *
 *      関数名  :  new_entry()                                            *
 *      機能    :  時刻設定，各メールボックスにメッセージを送信する       *
 *      返却値  :  なし                                                   *
 * ********************************************************************** */
void    new_entry( p_msg, p_cmd )
char    *p_msg;                 /* 表示データを格納したバッファへのポインタ */
struct RCV_COM  *p_cmd;         /* コマンド・データの構造体へのポインタ */
{
        unsigned int    *min,   /* システム時刻（分）を変更するためのポインタ */
                        *hour;  /* システム時刻（時）を変更するためのポインタ */
        struct  NEWS    *msgaa; /* 送信する表示データを格納するメッセージのアク
                                   セス・アドレス */

        int             err_free, /* フリー・メールボックスからメッセージをrcv_msgで
                                     受け取るときの返却値 */
```

リストB-5　RECV.C（5/7）

```
                err_time; /* 時刻変更関数からの返却値 */

long            time_to_wait0 = 0;
                        /* rcv_msg（即時リターン）のタイム・アウト時間
                         */

min = &sys.time.curnt_min;
hour = &sys.time.curnt_hour;
        /* システム時刻を変更するため，ポインタにアドレスを設定 */

switch ( p_cmd->mode ) {        /* モード */

        case TIME_MODE:         /* 時刻モード */
                err_time = new_time_set( p_msg, min, hour );
                        /* 時刻を変更する */

                if ( err_time == DATA_ERR ) {
                        /* 変更する時刻のデータが不正 */

                        return;
                }

                wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

                sys.new_time_flag = SET;
                        /* システム時刻変更用フラグをセットする */

                sig_sem( sys_sem_aa, SYS_SEM_COUNT );

                wup_tsk( time_task_aa );
                        /* 時刻タスクを起床させる */

                break;

        case NEWS_MODE: /* ニュース，コマーシャル，緊急モード */
        case CM_MODE:
        case EMG_MODE:
                err_free = rcv_msg( T_TMOUT, &msgaa, free_mesg_mbx_aa,
                                                &time_to_wait0 );
                        /* 表示データのためのフリー・メッセージを受け取る
                           （即時リターン） */

                if ( err_free != TE_OK ) {
                        /* メッセージが取れない */

                        return;
                }

                msgaa->msg_id = 0xffff;
                        /* メッセージID（すべてて0xffff） */

                msgaa->left_count = chg_hex( p_cmd->left_cnt, 4 );
                        /* 表示回数 */

                strcpy( msgaa->nmsg, p_msg );
                        /* 表示データをコピー */

                wai_sem( T_NOOPT, msg_sem_aa, MSG_SEM_COUNT );
```

リストB-5　RECV.C（6/7）

```
                        switch ( p_cmd->mode ) {
                                /* モードを見て，それぞれのメールボックスにメッ
                                   セージを送信 */

                                case NEWS_MODE:
                                        snd_msg( news_mbx_aa, msgaa );
                                        break;
                                case CM_MODE:
                                        snd_msg( cm_mbx_aa, msgaa );
                                        break;
                                case EMG_MODE:
                                        snd_msg( emg_mbx_aa, msgaa );
                                        break;
                        }

                        sig_sem( msg_sem_aa, MSG_SEM_COUNT );
                break;
        }
}

/* ************************************************** *
 *      関数名   :  new_time_set()                    *
 *      機能     :  システム時刻を新規に設定する      *
 *      返却値   :  < 0> 正常終了                     *
 *                  <-1> 時刻データ不正               *
 * ************************************************** */
int     new_time_set( p, min, hour )
char    *p;             /* 変更する時間データを格納したバッファへのポインタ */
unsigned int *min;      /* システム時刻（分）へのポインタ */
unsigned int *hour;     /* システム時刻（時）へのポインタ */
{
        int             i;
        unsigned int    set_hour,       /* 変更する時刻を代入する変数（時） */
                        set_min,        /* 変更する時刻を代入する変数（分） */
                        time = 0;       /* 変更する時刻を代入する変数 */

                /* 配列の指定位置の文字列の2文字を数値に変換し，変数に代入する
                   文字が不正（'0'から'9'の間にない）だったら，異常終了 */
        for ( i = 0; i < 2; i++ ) {
                if ( *(p+i*3) <= '9' && *(p+i*3) >= '0' ) {
                        if ( *(p+i*3+1) <= '9' && *(p+i*3+1) >= '0' ) {
                                time = 10*((int)*(p+i*3) & 0x0f)
                                         + ((int)*(p+i*3+1) & 0x0f );

                                if ( i == 0 ) { /* 時データ */
                                        set_hour = time;

                                } else {        /* 分データ */
                                        set_min = time;
                                }

                        } else {
                                return( DATA_ERR );
                        }               /* 異常終了（時刻データ不正） */
                } else {
                        return( DATA_ERR );
                }
        }
```

リストB-5　RECV.C（7/7）

```
        if ( set_min >= SYS_MIN_MAX || set_hour >= SYS_HOUR_MAX ) {
                /* 分が60以上，時が24以上の場合 */

                return( DATA_ERR );      /* 異常終了（時刻データ不正） */
        }

        wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

        *hour = set_hour;          /* システム時刻を変更 */
        *min = set_min;

        sig_sem( sys_sem_aa, SYS_SEM_COUNT );

        return( DATA_OK );         /* 正常終了（時刻データ変更） */

}
```

リストB-6　SHIFT.C（1/4）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : shift.c
TARGET          : V55PI
PROCEDURES DEFINED      :       shift_task(),
                                shift_bit();

*****************************************************************************/

#include        "led.h"

void    shift_task( void );     /* ------------  シフト・タスク  ----------- */
                                /* LEDメールボックスからLED表示メッセージを  */
                                /* 受け取り，フォント・データに変換したあと  */
                                /* LED表示バッファにフォント・データをセット */
                                /* する。                                    */
                                /* そして，LED表示バッファを1ビット・シフト  */
                                /* し，SHIFT_TIME時間wai_tskする。           */
                                /* ----------------------------------------- */

void    shift_bit( void );      /* LED表示バッファを1ビット・シフトする */

void    lputs( char * );        /* NULLが出るまで，バッファの内容をLEDに表示
                                   する */

void    lprintf( char *, ... ); /* NULLが出るまで，バッファの内容をLEDに表示
                                   する（書式付き） */

unsigned short  fdata[MAX_BUF][MAX_COLUMN];     /* LED表示用バッファ */

/* ***************************** *
 *        シフト・タスク         *
 * ***************************** */
void    shift_task( void )
{
        int             i;
        struct  LED     *led_msgaa;     /* LED表示メッセージのアクセス・アドレ
                                           ス */

        long            time_to_wait3000 = 3000;
                                /* wai_tskタイム・アウト時間 */

        for ( i = 0; i < MAX_COLUMN; i++ ) {    /* LED表示用バッファの初期化*/
                fdata[0][i] = 0;
                fdata[1][i] = 0;
                fdata[2][i] = 0;
                fdata[3][i] = 0;
                fdata[4][i] = 0;
        }

        lprintf("*** LED SYSTEM ***    V55PI    " );
                /* 初期メッセージをLEDに表示する */

        wai_tsk( &time_to_wait3000 );
                /* LED画面を一定時間止める */

        while (1) {
```

リストB-6　SHIFT.C（2/4）

```
                rcv_msg( T_NOOPT, &led_msgaa, led_mbx_aa );
                        /* LEDメールボックスから，LED表示メッセージを受け取る
                         */

                lputs( led_msgaa->lmsg );
                        /* LED表示メッセージをLEDに表示する */

                snd_msg( led_msgaa->free_mbxaa, led_msgaa );
                        /* LED表示用フリー・メールボックスに，LED表示メッセージ
                           を返却（送信）する */

                lputs( "        " );
                        /* LED画面をクリアする */

        }
}

/* ************************************************************ *
 *      関数名  :  shift_bit()                                  *
 *      機能    :  LED表示用の配列を１ビット・シフトする        *
 *      返却値  :  なし                                         *
 * ************************************************************ */
void    shift_bit( void )
{
        int     buf_num,        /* LED表示用バッファ（fdata）のバッファ番号 */
                column;         /* LED表示用バッファ（fdata）の行番号 */

                /* LED表示用バッファ全体を1ビット・シフトする */
        for ( column = 0; column < MAX_COLUMN; column++ ) {
                /* MAX_COLUMN行分 */

                for ( buf_num = 0; buf_num < (MAX_BUF-1); buf_num++ ) {
                        /* バッファのMAX_BUF-1番目まで */

                        fdata[buf_num][column] <<= 1;
                                /* バッファの1行を1ビット・シフトする */

                        fdata[buf_num][column] |= ((fdata[buf_num+1][column]
                                & 0x8000) >> (MAX_COLUMN-1));
                                /* buf_num番目のバッファの0ビット目に，
                                   buf_num+1番目のバッファのMAX_COLUMN-1
                                   ビット目の内容を代入する */

                }
                fdata[buf_num][column] <<= 1;
                        /* MAX_BUF番目のバッファを1ビット・シフトする */

        }
}

/* ********************************************************************** *
 *      関数名  :  lputs()                                                *
 *      機能    :  NULLが出るまで，バッファの内容をLEDに表示する          *
 *      返却値  :  なし                                                   *
 * ********************************************************************** */
void    lputs( str )
char    *str;           /* LEDに表示させたいバッファの先頭アドレス */
{
        int     i, ii;
```

リストB-6　SHIFT.C（3/4）

```
        int     delta;          /* フォント・データ取り出しルーチンからの返却値
                                   1=半角, 2=全角 */

        long    shift_time = SHIFT_TIME;
                                /* シフト間隔時間 */

        for ( i = 0; str[i] != (char)NULL; i += delta ) {
                /* 表示データ・バッファにNULLが出るまで繰り返す */

                if ( (str[i] == CR) || (str[i] == LF) ) {
                        /* 表示データ・バッファがCRかLFの場合 */

                        for ( ii = 0; ii < MAX_COLUMN; ii++ ) {
                                fdata[MAX_BUF-1][ii] = 0;
                                        /* MAX_BUF-1番目のLED表示バッファに0を
                                           代入する */
                        }

                        delta = 1;
                                /* 表示データ・バッファの内容を1進めるため */

                        for ( ii = 0; ii < (MAX_COLUMN*4) ; ii++ ) {
                                /* LEDが全画面スクロールするまで */

                                shift_bit();
                                        /* LED表示バッファを1ビット・シフト */

                                wai_tsk( &shift_time );
                                        /* シフト間隔 */
                        }

                } else {

                        delta = strfont( &str[i], &fdata[MAX_BUF-1][0] );
                                /* 表示データ・バッファの内容のフォント・データを
                                   LED表示バッファに格納する */

                        for ( ii = 0; ii < (delta*8); ii++ ) {
                                /* delta*8(deltaが1なら半角8ビット・シフト,
                                   deltaが2なら全角16ビット・シフトする */

                                shift_bit();
                                wai_tsk( &shift_time );
                        }
                }
        }
}

/* ********************************************************************** *
 *      関数名  :  lprintf()                                              *
 *      機能    :  NULLが出るまで, バッファの内容をLED表示する            *
 *                 (書式付き)                                             *
 *      返却値  :  なし                                                   *
 * ********************************************************************** */
void    lprintf( char *fmt, ... )
/* char *fmt;   */      /* 書式 */
{
        char    str[64];
        va_list ap;
```

リストB-6　SHIFT.C（4/4）

```
        va_start( ap, fmt );
        vsprintf( str, fmt, ap );
        va_end( ap );

        lputs( str );
}
```

リストB-7　TIME.C（1/2）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : time.c
TARGET          : V55PI
PROCEDURES DEFINED      :       time_task(),
                                change_time();

*****************************************************************************/

#include        "led.h"

void    time_task( void );      /* -------------  時刻タスク  ------------- */
                                /* 一定間隔で周期起床する（または受信割り込 */
                                /* み処理タスクから起こされる）。           */
                                /* 起床後，時刻変更用フラグがセットされてい */
                                /* たら，フラグをリセットし，周期起床をやり */
                                /* 直す。                                   */
                                /* セットされていなかったら，時刻の更新をす */
                                /* る。                                     */
                                /* ---------------------------------------- */

void    change_time( void );    /* 時刻を変更する */

/* ******************** *
 *    タイム・タスク    *
 * ******************** */
void    time_task( void )
{
        struct  CYC_WUP pk_cyc_wup;             /* cyc_wup用構造体 */
        int             prptn,                  /* 現在のフラグの内容 */
                        p_wupcnt;               /* 起床要求の回数 */

        pk_cyc_wup.utime = INIT_UTIME;          /* 初期起床時刻（上位） */
        pk_cyc_wup.ltime = pk_cyc_wup.interval = CYC_TIME;
                /* 周期起床間隔 */

        pk_cyc_wup.count = UNTIL_DEL_TASK;
                /* 対象タスク（時刻タスク）が削除されるまで，起床要求を出し続ける */

        cyc_wup( T_REL, CURNT_TASK, &pk_cyc_wup );
                /* 周期起床（自タスク，相対時間） */

        set_flg( T_OR, &prptn, flag_aa, SETPTN1 );
                /* 時刻表示タイミング・イベント・フラグのビット0に1をセットする */

        while (1) {

                slp_tsk();
                        /* 周期起床によって起きるか，受信割り込み処理タスクから
                           の起床要求によって起こされる */

                wai_sem( T_NOOPT, sys_sem_aa, SYS_SEM_COUNT );

                if ( sys.new_time_flag == SET ) { /* 時刻変更用フラグがセットさ
                                                     れている場合 */
                        sys.new_time_flag = RESET;
                                /* フラグ・クリア */
```

リストB-7　TIME.C（2/2）

```
                        can_cyc( CURNT_TASK );
                                /* 周期起床をキャンセルする */

                        can_wup( &p_wupcnt, CURNT_TASK );
                                /* 周期起床のやり直し */

                        cyc_wup( T_REL, CURNT_TASK, &pk_cyc_wup );
                                /* 時刻変更用フラグがセットされていない場合 */

                } else {
                        change_time();
                                /* 時刻を更新する */

                }

                sig_sem( sys_sem_aa, SYS_SEM_COUNT );

                if (sys.curnt_mode == TIME_MODE || sys.curnt_mode == MIX_MODE) {
                         /* 時刻モードまたは混在モードの場合だけ，時刻表示タイミ
                            ング・イベント・フラグのビット0に1をセットする */

                        set_flg( T_OR, &prptn, flag_aa, SETPTN1 );
                                /* 時刻表示タイミング・イベント・フラグのビット0
                                   に1をセットする */

                }

        }
}

/* ********************************************************** *
 *      関数名  :  change_time()                              *
 *      機能    :  システム時刻を更新する                     *
 *      返却値  :  なし                                       *
 * ********************************************************** */
void    change_time( void )
{
        unsigned int    *min,   /* システム時刻（分）を変更するためのポインタ */
        *hour;  /* システム時刻（時）を変更するためのポインタ */
        min = &sys.time.curnt_min;      /* システム時刻 */
        hour = &sys.time.curnt_hour;

        (*min)++;                       /* システム時刻（分）をインクリメント */
        if ( *min >= SYS_MIN_MAX ) {    /* システム時刻（分）が，SYS_MIN_MAX以上
                                           になった場合 */

                *min = 0;               /* システム時刻（分）を0にする */

                (*hour)++;              /* システム時刻（時）をインクリメントす
                                           る */

                if ( *hour >= SYS_HOUR_MAX ) {  /* システム時刻（時）が，SYS_
                                                   HOUR_MAX 以上になった場合 */

                        *hour = 0;      /* システム時刻を0にする */
                }
        }
}
```

リストB-8　FUNC.C（1/4）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : func.c
TARGET          : V55PI
PROCEDURES DEFINED      :       rprintf(),
                                rputc(),
                                rputs(),
                                rgetc(),
                                rgetch(),
                                rgets(),
                                buf_init(),
                                chg_hex();

*****************************************************************************/

#include        "led.h"

void    rprintf( char *, ... ); /* リング・バッファから，リターン・コードが来る
                                   まで文字を取り出す（書式付き）（未使用） */

int     rgetch( void );         /* リング・バッファから1文字取り出す（待ちあ
                                   り）（未使用） */

void    rgets( char * );        /* リング・バッファから，リターン・コードが来る
                                   まで文字を取り出す（未使用） */

int     rgetc( void );          /* リング・バッファから1文字取り出す（待ちなし）*/

int     rputc( char );          /* 1文字をシリアル送信する（未使用） */

void    rputs( char * );        /* NULLが来るまで，表示バッファの内容をシリア
                                   ル送信する（未使用） */

void    buf_init( void );       /* リング・バッファの初期化 */

int     chg_hex( char *, int ); /* 文字列を数値（１６進）に変換する */

/* ********************************************************************** *
 *      関数名 : rprintf()                                                *
 *      機能   : バッファから，CR（リターン・コード）が来るまで           *
 *               PCに文字を送信する（書式付き）（未使用）                 *
 *      返却値 : なし                                                     *
 * ********************************************************************** */
void    rprintf( char *fmt, ... )
{
        char    buf[64];        /* 送信するためのバッファ */
        va_list ap;             /* 各名なし引き数を順々に指す */

        va_start( ap, fmt );    /* 最初の名なし引き数を指すようにする */

        vsprintf( buf, fmt, ap );
                /* 書式どおりに文字列に変換し，bufに格納する */

        va_end( ap );           /* クリーン・アップ */

        rputs( buf );           /* PCに送信CRが来るまで */
}
```

保守／廃止

リストB-8　FUNC.C（2/4）

```
/* ********************************************************************** *
 *      関数名 : rgetch()                                                 *
 *      機能   : リング・バッファから1文字取り出す（待ちあり）（未使用）*
 *      返却値 : リング・バッファから取り出した1文字                      *
 * ********************************************************************** */
int  rgetch( void )
{
        int     getdata;        /* リング・バッファから取り出した1文字 */

        while ( (getdata = rgetc()) == D_EMPTY );
                /* D_EMPTY（バッファが空）の間，rgetcし続ける */

        return( getdata );
                /* 1文字取り出したあと，その文字を返す */
}

/* ********************************************************************** *
 *      関数名 : rgets()                                                  *
 *      機能   : リング・バッファから，CR（リターン・コード）が来るまで *
 *               文字を取り出す（未使用）                                 *
 *      返却値 : なし                                                     *
 * ********************************************************************** */
void  rgets( p )
char    *p;             /* リング・バッファから取り出した文字を格納するバッファ
                           へのポインタ */
{
        int     getdata;        /* リング・バッファから取り出した1文字 */
        int     i;

        for ( i = 1; i < BUF_SIZ; i++ ){ /* BUF_SIZ-1回繰り返す */

                  /* リング・バッファの先頭から最後まで */

                getdata = rgetch();
                        /* 1文字取り出し */

                if( getdata == CR ) {

                        /* 取り出した1文字がリターン・コードだったら，forループ
                           を抜ける */
                        break;
                }

                *(p++) = getdata;
                        /* バッファをインクリメントして，取り出した文字をバッフ
                           ァに入れる */
        }

        *p = (char)NULL;
                /* バッファの終わり */
}

/* ********************************************************************** *
 *      関数名 : rgetc()                                                  *
 *      機能   : リング・バッファから１文字取り出す（待ちなし）          *
 *      返却値 : リング・バッファから取り出した1文字                      *
 *               <-1> 文字取り出し失敗（リング・バッファが空）  *
 * ********************************************************************** */
int     rgetc( void )
```

リストB-8　FUNC.C（3/4）

```
{
        unsigned char   getdata;        /* リング・バッファから取り出した1文
                                           字 */

        if ( RingBuf.dnum <= 0 ) {      /* 文字が受信されてなかったら，D_EMPTY
                                           を返す */
                return( D_EMPTY );
        }

        getdata  = RingBuf.buf[RingBuf.rp];
                /* リング・バッファから，1文字を取り出す */

        RingBuf.rp = (RingBuf.rp+1) & BUF_MASK;
                /* 読み込みポインタをインクリメントする。もし，読み込みポインタ
                   がリング・バッファのサイズより大きくなったら，読み込みポイン
                   タを0にする */

        _DI();                  /* 割り込み禁止 */

        RingBuf.dnum--;         /* 書き込み文字数をデクリメントする */

        _EI();                  /* 割り込み許可 */

        return((int)getdata );  /* 取り出した文字を返す */
}

/* ************************************************** *
 *      関数名 : rputc()                                *
 *      機能   : 1文字をシリアル送信する（未使用）      *
 *      返却値 : シリアルのステータス                   *
 * ************************************************** */
int     rputc( char putdata )           /* 送信する1文字 */
{
        unsigned char   status;         /* シリアル・ステータス */

        while (1) {

                if ( ( status = *((unsigned char *)SCU_STAT) ) & 1 ) {
                        /* シリアルのステータスを見て，送信データ・バッファ（STB）
                           が空になっている（送信データ書き込み可能な）場合 */

                        *((char *)SCU_DATA) = putdata;
                                /* 送信データの書き込み */

                        return( (int)status );
                                /* シリアル・ステータスを返す */
                }
        }
}

/* ********************************************************************** *
 *      関数名 : rputs()                                                  *
 *      機能   : NULLが来るまで，表示バッファの内容をシリアル送信する     *
 *                                          （未使用）                    *
 *      返却値 : なし                                                     *
 * ********************************************************************** */
void  rputs( p )
char    *p;             /* 表示バッファへのポインタ */
{
```

リストB-8　FUNC.C（4/4）

```
        while ( *p != (char)NULL ) {
                /* 表示バッファにNULLが来るまで */

                rputc( *p );    /* 1文字送信 */
                p++;            /* バッファをインクリメントする */
        }
}

/* ****************************************** *
 *      関数名 : buf_init()                   *
 *      機能   : リング・バッファの初期化     *
 *      返却値 : なし                         *
 * ****************************************** */
void    buf_init( void )
{
        int     i;

        RingBuf.wp = RingBuf.rp = RingBuf.dnum = 0;
        for ( i = 0; i < BUF_SIZ; i++ ) {
                RingBuf.buf[i] = 0;
        }
                /* 書き込みポインタ，読み込みポインタ，書き込み文字数，バッファ
                   をクリアする */
}

/* ********************************************************** *
 *      関数名 : chg_hex()                                    *
 *      機能   : 文字列を数値（１６進）に変換する             *
 *      返却値 : 変換した数値                                 *
 *               <-1> データ・エラー（変換不可能文字）        *
 * ********************************************************** */
int     chg_hex( p, count )
char    *p;             /* 変換する文字列へのポインタ */
int     count;          /* 変換したい文字数 */
{
        int     i,
                num;    /* 16進に変換した数値 */

        for ( i = 0, num = 0; i < count; i++ ) {
                if ( '0' <= *(p+i) && *(p+i) <= '9' ) {
                        num = 0x10*num + (int)(*(p+i) & 0xf);
                } else if( 'A' <= *(p+i) && *(p+i) <= 'F' ) {
                        num = 0x10*num + (int)(*(p+i) - 0x37);
                } else {
                        return( DATA_ERR );
                }
        }
                /* '0'から'9'または'A'から'F'の文字だけを16進に変換する。
                   それ以外の場合は，DATA_ERRを返す */

        return( num );          /* 変換した数値を返す */
}
```

リストB-9　INIT.C（1/3）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : init.c
TARGET          : V55PI
PROCEDURES DEFINED      :       eb_init(),
                                sfr_init(),
                                piu_init(),
                                tcu_init(),
                                scu_init(),
                                wait();
*****************************************************************************/
#include        "led.h"

void    eb_init( void );        /* EB-70433の初期化 */

void    sfr_init( void );       /* SFR（特殊機能）レジスタの初期化 */

void    piu_init( void );       /* uPD71055（パラレル・インタフェース・ユニット）
                                   の初期化 */

void    tcu_init( void );       /* uPD71054（タイマ・コントロール・ユニット）の
                                   初期化 */

void    scu_init( void );       /* uPD71051（シリアル・コントロール・ユニット）の
                                   初期化 */

void    wait( void );           /* I/Oウエイト */

/* ****************************************** *
 *      関数名  :  eb_init()                  *
 *      機能    :  EB-70433初期化             *
 *      返却値  :  なし                       *
 * ****************************************** */
void    eb_init( void )
{
        _DI();          /* 割り込み禁止 */

        sfr_init();     /* SFR（特殊機能）レジスタの初期化 */

        piu_init();     /* uPD71055の初期化 */

        tcu_init();     /* uPD71054の初期化 */

        scu_init();     /* uPD71051の初期化 */
}

/* ************************************************** *
 *      関数名  :  sfr_init()                         *
 *      機能    :  SFRレジスタ初期化ルーチン          *
 *      返却値  :  なし                               *
 * ************************************************** */
void    sfr_init( void )
{
        _INTM0 = 0x1;           /* 外部割り込みモード・レジスタ0
                                   NMI立ち上がりエッジ */

        _MBC = 0x61;            /* メモリ・ブロック・コントロール・レジスタ
```

リストB-9　INIT.C（2/3）

```
                                        ブロック0の上限アドレス:        128K
                                        ブロック1の上限アドレス:        0
                                        ブロック2の上限アドレス:        896K
                                        ブロック4の上限アドレス:        4M   */

        _PWC1= 0xfc;            /* プログラマブル・ウエイト・コントロール・レジ
                                   スタ1
                                        ブロック0のウエイト・ステート: 0
                                        ブロック1のウエイト・ステート: 3
                                        ブロック2のウエイト・ステート: 3
                                        ブロック3のウエイト・ステート: 3 */

        _PWC0= 0xaf;            /* プログラマブル・ウエイト・コントロール・レジ
                                   スタ0
                                        ブロック4のウエイト・ステート: 3
                                        ブロック5のウエイト・ステート: 3
                                        I/O空間のウエイト・ステート   : 2
                                        ブロック1のアドレス・ウエイト: 挿入しない
                                        ブロック4のアドレス・ウエイト: 挿入する */
}

/* ************************************************** *
 *      関数名   :  piu_init()                        *
 *      機能     :  μPD71055初期化ルーチン            *
 *      返却値   :  なし                              *
 * ************************************************** */
void    piu_init( void )
{
        *(char *)PMD = 0x92;    /* モード選択
                                        <グループ0>
                                                ・モード0
                                                ・ポート0: 入力
                                                ・ポート2（上位）: 出力
                                        <グループ1>
                                                ・モード0
                                                ・ポート1: 出力
                                                ・ポート2（下位）: 入力 */
        wait();

        *(char *)PORT2 = 0x0;
        wait();
        *(char *)PORT2 = 0x1;
        wait();
}

/* ************************************************************ *
 *      関数名   :  tcu_init()                                  *
 *      機能     :  μPD71054初期化ルーチン                      *
 *      返却値   :  なし                                        *
 * ************************************************************ */
void    tcu_init( void )
{
        *(char *)TMD = 0x32;
        wait();
        *(char *)TMD = 0x76;
        wait();
        *(char *)TMD = 0xb6;
        wait();
        *(char *)TCT0 = 0xff;
```

リストB-9　INIT.C（3/3）

```
        wait();
        *(char *)TCT0 = 0x0;
        wait();
        *(char *)TCT1 = 0x40;
        wait();
        *(char *)TCT1 = 0x1f;
        wait();
        *(char *)TCT2 = 0x34;
        wait();
        *(char *)TCT2 = 0x00;
        wait();
}

/* ************************************************************ *
 *      関数名  :  scu_init()                                   *
 *      機能    :  μPD71051初期化ルーチン                       *
 *      返却値  :  なし                                         *
 * ************************************************************ */
void    scu_init( void )
{
        *(char *)SMD = 0x0;
        wait();
        *(char *)SMD = 0x0;
        wait();
        *(char *)SMD = 0x0;
        wait();
        *(char *)SMD = 0x40;
        wait();
        *(char *)SMD = 0xce;
        wait();
        *(char *)SMD = 0x35;
        wait();
}

/* ********************************** *
 *      関数名  :  wait()             *
 *      機能    :  I/Oウエイト        *
 *      返却値  :  なし               *
 * ********************************** */
void    wait( void )
{
}
```

リストB-10　STRFONT.C（1/3）

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : strfont.c
TARGET          : V55PI
PROCEDURES DEFINED      :       strfont(),
                                stoj(),
                                getfont();

*****************************************************************************/

#define         KANFONT_ADDR    0x740000        /* 全角フォント・データのアドレス */

#define         HANFONT_ADDR    0x720000        /* 半角フォント・データのアドレス */

#define         NODATA          0               /* バッファがNULL */

#define         HANKAKU         1               /* 半角 */

#define         ZENKAKU         2               /* 全角 */

#define         iskanji(c) ¥
                        (((c)>=0x81 && (c)<=0x9f) || ((c)>=0xe0 && (c)<=0xfc))
                        /* 全角か半角かを調べる */

int             strfont( unsigned char *, unsigned char * );
                        /* バッファの文字に対応するフォント・データを取り出す */
unsigned short stoj( unsigned short );
                        /* シフトJISコードをJISコードに変換する */

void            getfont( unsigned short , char *buf );
                        /* 漢字ROMから漢字フォントを取り出す */

/* ********************************************************************** *
 *      関数名 : strfont()                                                *
 *      機能   : バッファの文字に対応するフォント・データを取り出す      *
 *      返却値 : <0> データなし                                           *
 *               <1> 半角データ                                           *
 *               <2> 全角データ                                           *
 * ********************************************************************** */
int     strfont( str, buf )
unsigned char   *str;           /* 文字 */
unsigned char   *buf;           /* getfontで取り出した文字のフォント・データを
                                   格納するバッファ */
{
        if ( *str == 0 ) {
                return( NODATA );       /* データなし */
        }

        if ( iskanji( *str ) ) {
                getfont( stoj((*str << 8) | (*(str+1)) & 0xff), buf );
                return( ZENKAKU );      /* 全角データ */
        } else {
                getfont( 0x8000 | *str, buf );
                return( HANKAKU );      /* 半角データ */
        }
```

リストB-10　STRFONT.C（2/3）

```
}

/* ************************************************************ *
 *      関数名 : stoj()                                         *
 *      機能   : シフトJISコードをJISコードに変換する           *
 *      返却値 : 変換したJISコードの文字                        *
 * ************************************************************ */
unsigned short stoj( sjiscode )
unsigned short  sjiscode;       /* 変換するシフトJISコードの文字 */
{
        unsigned short  high,
                        low;

        high = (sjiscode & 0xff00) >> 8;
        low  = sjiscode & 0xff;

        if ( high < 0xa0 ) {
                high -= 0x71;
        } else {
                high -= 0xb1;
        }

        high = (high << 1) + 1;

        if ( low > 0x7f ) low -= 1;
        if ( low > 0x9d ) {
                low -= 0x7d;
                high += 1;
        } else {
                low -= 0x1f;
        }
        return((high << 8) | low);
}

/* ************************************************************************ *
 *      関数名 : getfont()                                                  *
 *      機能   : 拡張ROMから半角，全角のフォント・データを取り出す          *
 *      返却値 : なし                                                       *
 * ************************************************************************ */
void    getfont( code, buf )
unsigned short  code;           /* 文字 */
unsigned char   *buf;           /* フォント・データを格納するバッファ */
{
        int             i, k;

        unsigned long   paddr;          /* フォント・データの物理アドレス */

        unsigned char   *p;             /* フォント・データのアドレスへのポインタ */


        if ( code & 0x8000 ) {  /* 半角データ */

                p = (char *)(HANFONT_ADDR << 8);
                        /* 拡張ROMの半角フォント・データのアドレスをセット */

                k = (code & 0x7fff) - 0x20; /* (int)(code - 0x8020); */

                for ( i = 0; i < 16; i++ ) {
                        buf[i*2+1] = *(p+k*16+i);
```

リストB-10　STRFONT.C（3/3）

```
        }
        return;
}

/* 全角データ */

paddr = (unsigned long)((code & 0x1f) | ((code & 0x700)>>1));

switch ( code & 0x7000 ) {
        case 0x2000:
                paddr |= (code & 0x60) << 5;
                break;
        case 0x3000:
        case 0x4000:
                paddr |= (code & 0x60) | ((code & 0x800) >> 1) |
                        ((code & 0x4000) >> 3);
                break;
}

paddr = (paddr << 6) | KANFONT_ADDR;

p = (char *)(((paddr & 0xffff00) << 8) | (paddr & 0xff) );

for ( i = 0; i < 16; i++ ) {
        buf[i*2]   = p[i*4+2];
        buf[i*2+1] = p[i*4];
}
}
```

リストB-11　INTHAND.C

```
/*****************************************************************************
        LED表示システム
*****************************************************************************

MODULE          : inthand.c
TARGET          : V55PI
PROCEDURES DEFINED      :       rs_inthand();

*****************************************************************************/

#include        "led.h"

_SWI    void    rs_inthand( void );
                /* ------------ シリアル受信割り込みハンドラ ------------ */
                /* シリアル受信割り込みが起きると，リング・バッファに1文  */
                /* 字書き込み，受信割り込み処理タスクを起こす             */
                /* ------------------------------------------------- */

/* ********************************** *
 *      シリアル受信割り込みハンドラ    *
 * ********************************** */
_SWI    void    rs_inthand( void )
{
        char    putdata;                /* PCから送信された文字 */

        putdata = *((char *)SCU_DATA);  /* PCから送信された文字を読み出す */

        if ( RingBuf.dnum == BUF_SIZ ) {
                /* 文字数がBUF_SIZを越えていた場合 */

                ret_int();      /* 何も処理をしないで割り込み終了 */
        }

        RingBuf.buf[RingBuf.wp] = putdata;
                /* 読み出した文字をリング・バッファに書き込む */

        RingBuf.wp = (RingBuf.wp+1) & BUF_MASK;
                /* 書き込みポインタをインクリメント。書き込みポインタがバッファ・
                   サイズを越えたら，0にする */

        RingBuf.dnum++;
                /* 書き込み文字数をインクリメント */

        iret_wup( recv_task_aa );       /* 受信割り込み処理タスクを起こす */

}
```

リストB-12　LED.H（1/9）

```
/*****************************************************************************
       LED表示システム
*****************************************************************************

MODULE          : led.h
TARGET          : V55

*****************************************************************************/

/* ************************** *
 *       include file         *
 * ************************** */
#include        "sfrs55pi.h"
#include        "stdio.h"
#include        "rcopy.h"
#include        "rx.h"
#include        "stdarg.h"

/* ************************** *
 *      I/Oポート・アドレス     *
 * ************************** */
#define         SCU_DATA        0x0F000020 /* メモリ・マップトI/Oアドレス */

#define         SCU_STAT        0x0F000022 /* シリアル・ステータス・レジスタ */

#define         PIU             0x0F000004 /* パラレル・インタフェース・ユニ
                                              ット(UPD71055) */

#define         PIU_PORT        0x8000          /* uPD71055 I/Oポート    */

#define         CTRL_PORT_1_2   PIU_PORT+6      /* コントロール・ポート1,2 */

#define         CTRL_PORT_3_4   PIU_PORT+0xe    /* コントロール・ポート3,4 */

#define         PARA1_P0        PIU_PORT        /* PIU1-P0       */

#define         PARA1_P1        PIU_PORT+2      /* PIU1-P1       */

#define         PARA2_P0        PIU_PORT+8      /* PIU2-P0       */

#define         PARA2_P1        PIU_PORT+10     /* PIU2-P1       */

#define         PARA1_P2        PIU_PORT+4      /* PIU1-P2       */

#define         PMD             0xf000006       /* uPD71055モード選択レジスタ
                                                   */

#define         PORT2           0xf000004       /* ポート2 */

#define         TMD             0xf000016       /* タイマ・モード・レジスタ */

#define         TCT0            0xf000010       /* タイマ・カウント・レジスタ0
                                                   */

#define         TCT1            0xf000012       /* タイマ・カウント・レジスタ1
                                                   */

#define         TCT2            0xf000014       /* タイマ・カウント・レジスタ2
                                                   */
```

リストB-12　LED.H（2/9）

```
#define         SMD             0xf000022       /* シリアル・モード・レジスタ */

#define         INIT_PIU        0x8080          /* uPD71055初期化データ（ポート
                                                   0-3をすべて出力にする */

/* ***************************** *
 *      転送データ用コード       *
 * ***************************** */
#define         STX             0x02    /* 表示メッセージの先頭          */

#define         ETX             0x03    /* 表示メッセージの最後          */

#define         END             0x1a    /* 登録メッセージ送信終わり      */

#define         CR              0xd     /* 改行コード                    */

#define         LF              0xa     /* 左寄せコード                  */

/* ********************************** *
 *      OSシステム・コール用マクロ    *
 * ********************************** */
#define         BLKCNT          (( sizeof( union maxblk ) + 15 )/16 )
                                        /* get__blkするブロック数 */

#define         CURNT_TASK      0       /* 自タスク */

#define         TASK_DS         0x400   /* タスクで使用するデータ・
                                           セグメント値 */

#define         STACK_SIZ       0x400   /* タスクで使用するユーザ・
                                           スタック・サイズ */

#define         INT_RS          0x800B  /* 受信割り込みベクタ番号 */

#define         MASK_NMI        0x2     /* NMIマスク値 */

#define         INT_DS          TASK_DS /* 割り込みハンドラで使用するDS0値
                                           */

#define         SHIFT_TIME      30      /* シフト・タスクのウエイト時間 */

#define         INIT_UTIME      0       /* 初期起床時刻（上位） */

#define         CYC_TIME        60000   /* 周期起床時間間隔 */

#define         UNTIL_DEL_TASK  0
                        /* 対象タスクが削除されるまで，周期起床要求を出し続ける
        */

#define         MPL_SIZ         ((FREE_MESG_MAX + FREE_CTRL_MAX + FREE_LED_MAX¥
                                        + 1)*((BLKCNT*BLK_SIZ) + 0x10))
                                        /* メモリ・プール・サイズ        */

#define         BLK_SIZ         0x10    /* メモリ・プールからメモリ・ブロック
                                           を切り出す最小単位            */

#define         SETPTN0         0       /* set_flgで，ビット0を0にクリア */

#define         SETPTN1         1       /* set_flgで，ビット0を1にセット */
```

リストB-12　LED.H（3/9）

```
#define         SYS_SEM_MAX     1       /* システム情報排他制御用セマフォが
                                           管理する資源の最大値          */

#define         MSG_SEM_MAX     1       /* 表示データ排他制御用セマフォが
                                           管理する資源の最大値          */

#define         SYS_SEM_COUNT   1       /* システム情報排他制御用セマフォで
                                           獲得したい（操作する）資源数 */

#define         MSG_SEM_COUNT   1       /* 表示データ排他制御用セマフォで
                                           獲得したい（操作する）資源数 */

/* ********************************** *
 *      リング・バッファ用マクロ        *
 * ********************************** */
#define         BUF_SIZ         256             /* リング・バッファ・サイズ */

#define         D_EMPTY         -1              /* バッファが空             */

#define         BUF_MASK        BUF_SIZ-1       /* リング・バッファ折り返し */

/* ***************************** *
*        システム情報マクロ      *
* ***************************** */
#define         TIME_MODE       '0'     /* 時刻モード           */

#define         NEWS_MODE       '1'     /* ニュース・モード     */

#define         CM_MODE         '2'     /* コマーシャル・モード */

#define         EMG_MODE        '4'     /* 緊急モード           */

#define         MIX_MODE        '8'     /* 混在モード           */

#define         SET             1       /* システム時刻変更用フラグ，セット   */

#define         RESET           0       /* システム時刻変更用フラグ，リセット */

#define         SYS_MIN_MAX     60
                                        /* システム時刻（分）最大値     */

#define         SYS_HOUR_MAX    24      /* システム時刻（時）最大値     */

#define         SYS_TIME_INIT_MIN       0
                                        /* システム時刻，初期時間（分） */

#define         SYS_TIME_INIT_HOUR      0
                                        /* システム時刻，初期時間（時） */

/* ***************************** *
 *      コマンド用マクロ         *
 * ***************************** */
#define         ADD_MESG        '0'     /* 表示データ追加       */

#define         DEL_MESG        '1'     /* 表示データ削除       */

#define         CHANGE_MODE     '2'     /* 表示モード変更       */

#define         LOOK_MESG       '8'     /* 表示データの参照     */
```

リストB-12　LED.H（4/9）

```
#define         MARKING_ID      0               /* マーキング・メッセージID */

#define         MSGSIZ          BUF_SIZ         /* 表示データ・サイズ       */

/* ********************************** *
 *       フリー・メールボックス初期値   *
 * ********************************** */
#define         FREE_MESG_MAX   20      /* 表示データ用フリー・メールボックスの
                                           最大値 */

#define         FREE_CTRL_MAX   3       /* コマンド用フリー・メールボックスの
                                           最大値 */

#define         FREE_LED_MAX    1       /* LED表示用フリー・メールボックスの
                                           最大値 */

/* ********************************************************** *
 *        タスク，メールボックス，フラグ，セマフォID番号                *
 * ********************************************************** */
#define         INIT_TASK_ID            1       /* 初期タスクID            */

#define         LED_TASK_ID             2       /* LEDドライブ・タスクID */

#define         RECV_TASK_ID            3       /* 受信データ処理タスクID */

#define         SHIFT_TASK_ID           4       /* シフト・タスクID        */

#define         CTRL_TASK_ID            5       /* コントロール・タスクID */

#define         TIME_TASK_ID            6       /* 時刻タスクID            */

#define         MESG_TASK_ID            7       /* メッセージ・タスクID    */

#define         NEWS_MBX_ID             1       /* ニュース・
                                                       メールボックスID  */

#define         CM_MBX_ID               2       /* コマーシャル・
                                                       メールボックスID  */

#define         EMG_MBX_ID              3       /* 緊急メールボックスID    */

#define         LED_MBX_ID              4       /* LED表示メールボックスID */

#define         CTRL_MBX_ID             5       /* コマンド・メールボックスID */

#define         FREE_MESG_MBX_ID        6       /* 表示データ用フリー・
                                                       メールボックスID  */

#define         FREE_CTRL_MBX_ID        7       /* コマンド用フリー・
                                                       メールボックスID  */

#define         FREE_LED_MBX_ID         8       /* LED表示用フリー・
                                                       メールボックスID  */

#define         MPL0_ID                 0       /* メモリ・プール#0 ID  */

#define         MPL1_ID                 1       /* メモリ・プール#1 ID  */
```

リストB-12　LED.H（5/9）

```
#define         FLAG_ID                 1       /* 時刻表示タイミング・
                                                        イベント・フラグID */

#define         MSG_SEM_ID              1       /* 表示データ排他制御用
                                                        セマフォID      */

#define         SYS_SEM_ID              2       /* システム情報排他制御用
                                                        セマフォID      */

/* ***************************** *
 *    タスクのプライオリティ    *
 * ***************************** */
#define         INIT_TASK_PRI           110     /* 初期タスク・プライオリティ */

#define         LED_TASK_PRI            120     /* LEDドライブ・タスク・
                                                        プライオリティ  */

#define         SHIFT_TASK_PRI          130     /* シフト・タスク・
                                                        プライオリティ  */

#define         RECV_TASK_PRI           140     /* 受信データ処理タスク・
                                                        プライオリティ  */

#define         CTRL_TASK_PRI           150     /* コントロール・タスク・
                                                        プライオリティ  */

#define         TIME_TASK_PRI           160     /* 時刻タスク・プライオリティ*/

#define         MESG_TASK_PRI           170     /* メッセージ・タスク・
                                                        プライオリティ  */
/* ********************************** *
 *      other define                  *
 * ********************************** */

#define         DATA_OK         0

#define         DATA_ERR        -1

#define         SEND_OK         0       /* メッセージを送信した */

#define         NOT_SEND        1       /* メッセージを送信しなかった */

#define         MAX_BUF         5       /* LED表示用バッファ個数        */

#define         MAX_COLUMN      16      /* LED表示用バッファ列の個数    */

/* ********************************** *
 *       externアクセス・アドレス     *
 * ********************************** */

extern  short   time_task_aa,           /* 時刻タスク                   */

                led_task_aa,            /* LEDドライブ・タスク          */

                mesg_task_aa,           /* メッセージ・タスク           */

                recv_task_aa,           /* 受信データ処理タスク         */

                ctrl_task_aa,           /* コントロール・タスク         */
```

リストB-12　LED.H（6/9）

```
                shift_task_aa,          /* シフト・タスク                  */
                mpl1_aa,                /* メモリ・プール#1                */
                news_mbx_aa,            /* ニュース・メールボックス        */
                cm_mbx_aa,              /* コマーシャル・メールボックス    */
                emg_mbx_aa,             /* 緊急メールボックス              */
                led_mbx_aa,             /* LED表示メールボックス           */
                ctrl_mbx_aa,            /* コマンド・メールボックス        */
                free_mesg_mbx_aa,       /* 表示データ用フリー・メールボックス */
                free_ctrl_mbx_aa,       /* コマンド用フリー・メールボックス  */
                free_led_mbx_aa,        /* LED表示用フリー・メールボックス */
                flag_aa,                /* 時刻表示タイミング・イベント・フラグ */
                sys_sem_aa,             /* システム情報排他制御用セマフォ  */
                msg_sem_aa;             /* 表示データ排他制御用セマフォ    */

/* ********************************** *
 *       externタスク，ハンドラ       *
 * ********************************** */
extern  void    init_task( void );              /* 初期タスク          */

extern  void    led_task( void );               /* LEDドライブ・タスク */

extern  void    recv_task( void );              /* 受信データ処理タスク */

extern  void    shift_task( void );             /* シフト・タスク      */

extern  void    ctrl_task( void );              /* コントロール・タスク */

extern  void    time_task( void );              /* 時刻タスク          */

extern  void    mesg_task( void );              /* メッセージ・タスク  */

_SWI    extern  void    rs_inthand( void );     /* 受信割り込みハンドラ */

/* ********************************** *
 *     OSシステム・コール用構造体     *
 * ********************************** */
struct  CYC_WUP {       /*** タスクの周期起床（cyc_wup)用構造体 ***/
        long    ltime;          /* 初期起床時刻（下位）         */
        short   utime;          /* 初期起床時刻（上位）         */
        long    interval;       /* 周期起床時間間隔             */
        short   count;          /* 起床回数                     */
};

struct  CRE_TSK {       /*** タスクの生成（cre_tsk）用構造体 ***/
        void    (*sta)();       /* タスク・スタート・アドレス          */
        short   stksz;          /* タスクで使用するユーザ・スタック・サイズ */
        short   tskpri;         /* 生成するタスクの初期優先度          */
```

リストB-12　LED.H（7/9）

```
        short   tskopt;         /* 使用しない                          */
        short   tskds;          /* 生成するタスクで使用する
                                   データ・セグメントの値              */
};

struct  CRE_MPL {       /*** メモリ・プールの生成（cre_mpl）用構造体 ***/
        long    mplsz;          /* メモリ・プール全体のサイズ          */
        short   blksz;          /* メモリ・プールからメモリ・ブロックを切り出す
                                   最小単位（バイト）                  */
};

struct  DEF_INT {       /*** 割り込みハンドラ定義（def_int）用構造体 ***/
        void    (*inthdr)();    /* 割り込みハンドラのスタート・アドレス*/
        short   intds;          /* 割り込みハンドラ内で使用するDS0値 */
};

/* ************************** *
 *      LEDシステム構造体     *
 * ************************** */
struct  RINGBUF {       /*** リング・バッファ構造体 ***/
        int     wp;                     /* 書き込みポインタ     */
        int     rp;                     /* 読み出しポインタ     */
        int     dnum;                   /* 書き込み文字数       */
        unsigned  char  buf[BUF_SIZ];   /* リング・バッファの配列 */
};

struct  TIME {          /*** システム時刻構造体 ***/
        unsigned int    curnt_hour;     /* システム時刻（時間） */
        unsigned int    curnt_min;      /*              （分）  */
};

struct  NEWS {          /*** ニュース，コマーシャル，緊急メッセージ構造体 ***/
        short   free_mbxaa;             /* 使用後返却されるフリー・メールボックス・
                                           アクセス・アドレス */

        unsigned int    msg_id;         /* メッセージID                 */
        unsigned int    left_count;     /* 残り表示回数                 */
        char    nmsg[MSGSIZ];           /* メッセージを格納する配列     */
};

struct  COMMAND {       /*** コマンド構造体 ***/
        short   free_mbxaa;             /* 使用後返却されるフリー・メールボックス・
                                           アクセス・アドレス           */

        unsigned char   command;        /* コントロール・コマンド       */
        unsigned char   mode;           /* モード                       */
};

struct RCV_COM {        /*** 受信コマンド構造体 ***/
        unsigned char   header;         /* 受信ヘッダ'P'格納変数        */
        unsigned char   command;        /* コントロール・コマンド       */
        unsigned char   mode;           /* モード                       */
        unsigned char   msg_id[4];      /* メッセージIDを格納する配列   */
        unsigned char   left_cnt[4];    /* 残り表示回数を格納する配列   */
};

struct  LED {           /*** LED表示メッセージ構造体 ***/
        short   free_mbxaa;             /* 使用後返却されるフリー・メールボックス・
                                           アクセス・アドレス           */
```

リストB-12　LED.H（8/9）

```
        char    lmsg[MSGSIZ];           /* 表示メッセージを格納する配列 */
};

union   maxblk {        /*** コマンド，メッセージ構造体の共用体  ***/
        struct NEWS     news;           /* ニュース，コマーシャル，
                                           緊急メッセージ構造体 */
        struct COMMAND  command;        /* コマンド構造体               */
        struct LED      led;            /* LED表示メッセージ構造体      */
};


struct  SYSTEM {        /****  システム・データ構造体 ***/
        int     new_time_flag;          /* システム時刻変更用フラグ     */
        unsigned char   curnt_mode;     /* 現在のモード                 */
        struct  TIME    time;           /* システム時刻                 */
};

/* ********************************** *
 *      extern LEDシステム外部変数      *
 * ********************************** */
extern  struct  RINGBUF         RingBuf;        /* リング・バッファ     */

extern  unsigned short fdata[5][16];            /* LED表示用バッファ    */

extern  struct SYSTEM   sys;                    /* システム・データ     */

/* ******************** *
 *      extern関数      *
 * ******************** */
extern  int     rgetc( void );          /* リング・バッファから１文字取り出し*/

extern  void    buf_init( void );       /* リング・バッファの初期化     */

extern  int     chg_hex( char *, int ); /* 文字列−>１６進数変換        */

extern  void    rprintf( char *, ... ); /* リング・バッファから，リターン・コー
                                           ドが来るまで文字を取り出す（書式付き
                                           ）（未使用） */

extern  int     rgetch( void );         /* リング・バッファから1文字取り出す
                                           （待ちあり）（未使用） */

extern  void    rgets( char * );        /* NULLが来るまで，表示バッファの内容
                                           をシリアル送信する（未使用） */

extern  int     rputc( char );          /* 1文字をシリアル送信する（未使用） */

extern  void    rputs( char * );        /* NULLが来るまで，表示バッファの内容
                                           をシリアル送信する（未使用） */

extern  int     romidata();

_ASM    int outpw( int port, int val ) {
%con port con val
        mov     AW,     %val
        mov     DW,     %port
        out     DW,     AW
%con port reg val
        mov     AW,     %val
```

リストB-12　LED.H（9/9）

```
        mov     DW,     %port
        out     DW,     AW
%con port mem val
        mov     AW,     WORD PTR %val
        mov     DW,     %port
        out     DW,     AW
%reg port con val
        mov     AW,     %val
        mov     DW,     %port
        out     DW,     AW
%reg port reg val
        mov     AW,     %val
        mov     DW,     %port
        out     DW,     AW
%reg port mem val
        mov     AW,     WORD PTR %val
        mov     DW,     %port
        out     DW,     AW
%mem port con val
        mov     AW,     %val
        mov     DW,     WORD PTR %port
        out     DW,     AW
%mem port reg val
        mov     AW,     %val
        mov     DW,     WORD PTR %port
        out     DW,     AW
%mem port mem val
        mov     AW,     WORD PTR %val
        mov     DW,     WORD PTR %port
        out     DW,     AW
}

_ASM    int inpw( int port ) {
%con port
        in      AW,     %port
%reg port
        mov     DW,     %port
        in      AW,     DW
%mem port
        mov     DW,     WORD PTR %port
        in      AL,     DW
}


/*END OF FILE*/
```

リストB-13　RX.H（1/3）

```
/****************************************
*       ITRONエラー・コード              *
****************************************/
#define TE_OK           0x0     /* 正常終了 */

#define TE_MEM          0x1     /* メモリ領域が確保できない */

#define TE_UDF          0x2     /* 未定義のシステム・コールを発行した*/

#define TE_CTX          0x3     /* 発行されるべきでない環境でシステム・コールを
                                   発行した */

#define TE_OBJ          0x4     /* 指定したアクセス・アドレスは対象とするオブジ
                                   ェクト以外を指している */

#define TE_AA           0x5     /* アクセス・アドレスが不適当 */

#define TE_PA           0x6     /* パケット・アドレスが不適当 */

#define TE_MA           0x7     /* メッセージ・アドレス，メモリ・ブロック・アド
                                   レスが不適当 */

#define TE_SA           0x8     /* タスク，例外処理，終了処理，割り込みハンドラ
                                   のスタート・アドレスが不適当 */

#define TE_EXS          0x9     /* 生成しようとしているIDを持つオブジェクトはす
                                   でに存在している */

#define TE_NOEXS        0xa     /* 指定したキー・ワードのオブジェクトが生成され
                                   ていない */

#define TE_NODMT        0xb     /* 指定タスクはDORMANT状態ではない */

#define TE_NOSUS        0xc     /* 指定タスクはSUSPEND状態ではない */

#define TE_DMT          0xd     /* タスクの状態が，DORMANT状態である */

#define TE_IDZR         0xe     /* ID番号が0である */

#define TE_IDOVR        0xf     /* ID番号が大き過ぎる */

#define TE_TMOUT        0x10    /* 指定時刻を経過した */

#define TE_QOVR         0x11    /* キューイングのカウントがオーバフローした */

#define TE_SELF         0x12    /* 自タスクの指定はできない */

#define TE_DLT          0x13    /* 待っている間に，対象オブジェクトが削除された
                                   */

#define TE_OPT          0x14    /* サポートしていないオプションを指定した */

#define TE_WEVF         0x15    /* 指定されたイベント・フラグはすでに他のタスク
                                   が使用している */

#define TE_NOTMR        0x16    /* タイマ・コントロール・テーブルが確保できない
                                   */

#define TE_TPRI         0x17    /* タスク優先度エラー */
```

リストB-13　RX.H（2/3）

```
#define TE_IPRI         0x18    /* 割り込み優先度エラー */

#define TE_NOCYC        0x19    /* cyc_wupが発行されていないタスクに対する
                                   can_wupの発行 */

#define TE_MPLSZ        0x1a    /* メモリ・プール・サイズ・エラー */

#define TE_INTER        0x1c    /* 割り込みレベルの指定で，ビット7-14のいずれか
                                   がセットされている */

#define TE_DVN          0x1c    /* 指定された割り込みレベルは存在しない */

#define TE_SYS          0x1e    /* その他のエラー */

#define TE_PAR          0x1f    /* パラメータが不正 */

#define TE_SVC          0x40    /* 拡張システム・コールの機能コードが不正 */

/*************************************
*       ITRON systemcall options       *
*************************************/
#define T_NOOPT         0x0     /* オプションなし（無限待ちなど） */

#define T_FCOPR         0x1     /* cre_tsk : 数値演算プロセッサを使用する */

#define T_FRCRSM        0x1     /* rsm_tsk : すべてのサスペンド要求を解除 */

#define T_ABS           0x0     /* cyc_wup : 絶対時間 */

#define T_REL           0x1     /* cyc_wup : 相対時間 */

#define T_OR            0x0     /* set_flg : SET */

#define T_AND           0x1     /* set_flg : RESET */

#define T_REP           0x2     /* set_flg : REPLACE */

#define T_EXOR          0x3     /* set_flg : EXOR */

#define T_TMOUT         0x1     /* タイム・アウト指定あり */

#define T_ANDW          0x0     /* wai_flg : mskptnで1にセットされているビットの
                                             全ビットがセットされるまで待つ
                                   */

#define T_ORW           0x2     /* wai_flg : mskptnで1にセットされているビットの
                                             いずれかがセットされるまで待つ
                                   */

#define T_RESET         0x4     /* wai_flg : フラグがセットされ条件が満足したあと
                                             セットされたフラグをリセットする */

#define T_TPRI          0x0     /* cre_mbx : タスクの優先度順 */

#define T_TFIFO         0x1     /* cre_mbx : タスクのFIFO順 */

#define T_MPRI          0x0     /* cre_mbx : メッセージに付けられた優先度順 */
```

リストB-13　RX.H（3/3）

```
#define T_MFIFO         0x2     /* cre_mbx : メッセージのFIFO順 */

#define T_CPU           0x0     /* def_exc : マシン例外 */

#define T_OS            0x1     /* def_exc : ソフトウエア例外 */

#define T_TEXC          0x0     /* def_exc : 各タスク固有 */

#define T_NEXC          0x2     /* def_exc : 非タスク部 */

#define T_CEXC          0x4     /* def_exc : タスク共通 */
```

リストB-14　TASK.LC

```
-- Limit total memory to 128K bytes
MEMORY (#100000);
RESERVE ( #00000 TO #4000 );
LOCATE ( S__V55TEMPS {data} : #4000 TO #10000 );
LOCATE ( S_init_task {code} {bram} _romidata {constant} {news} {} : #F0100 TO #F8000 );
LOCATE ( __sfrseg : #FFE00 );
```

リストB-15　MAKEFILE

```
ROOTS           = b:¥itools
CHECK           = b:¥chk_lib¥check.lib
LDIR            = $(ROOTS)¥rtlibs¥cv55s¥lm¥libcv55s
INCDIR          = $(ROOTS)¥rtlibs¥inc
GSMAP           = $(ROOTS)¥x¥gsmap
FORM            = $(ROOTS)¥x¥form
CV55            = $(ROOTS)¥x¥cv55pi
ASMV55          = $(ROOTS)¥x¥asmv55pi
LLINK           = $(ROOTS)¥x¥llink
HEX             = hex
FILE            = strfont.ol time.ol main.ol init.ol led.ol shift.ol mesg.ol recv.ol func.ol c
trl.ol inthand.ol task.lc

main.map        : main.$(HEX)
        $(GSMAP) main.ab -n -o

main.$(HEX)     : main.ab
        $(FORM) main.ab -f s37 -e __sfrseg udata STARTVEC

main.ab         : $(FILE)
        $(LLINK) -i objlist -c task.lc -L $(LDIR) -L $(CHECK) -o main.ab -rs idata -b _romidat
a


main.ol         : main.c
        $(CV55) main.c -ic -is -D s2s -o -S $(INCDIR)

func.ol         : func.c
        $(CV55) func.c -ic -is -D s2s -o -S $(INCDIR)

time.ol         : time.c
        $(CV55) time.c -ic -is -D s2s -o -S $(INCDIR)

led.ol          : led.c
        $(CV55) led.c -ic -is -D s2s -o -ia -i -S $(INCDIR)

shift.ol        : shift.c
        $(CV55) shift.c -ic -is -D s2s -o -S $(INCDIR)

mesg.ol         : mesg.c
        $(CV55) mesg.c -ic -is -D s2s -o -S $(INCDIR)

recv.ol         : recv.c
        $(CV55) recv.c -ic -is -D s2s -o -S $(INCDIR)

ctrl.ol         : ctrl.c
        $(CV55) ctrl.c -ic -is -D s2s -o -S $(INCDIR)

strfont.ol      : strfont.c
        $(CV55) strfont.c -ic -is -D s2s -o -S $(INCDIR)

inthand.ol      : inthand.c
        $(CV55) inthand.c -ic -is -D s2s -o -S $(INCDIR)

init.ol         : init.c
        $(CV55) init.c -ic -is -D s2s -o -S $(INCDIR)

clean           :
        rm *.ol *.ab *.hex *.map .nfs*
```

リストB-16　OBJLIST

```
main.ol
time.ol
init.ol
led.ol
shift.ol
mesg.ol
recv.ol
func.ol
ctrl.ol
strfont.ol
inthand.ol
```

B.2　スタート・アップ部

リストB-17　CONFIG.ASM（1/2）

```
;Real time operating system for V-series

;/*********************************************************************/
;/*                                                                   */
;/*                                                                   */
;/*              system information table for RX423                   */
;/*                                                                   */
;/*              Copyright (c) NEC corporation 1993                   */
;/*                                                                   */
;/*                                                                   */
;/*********************************************************************/

sgroup          group           sys_conf
sys_conf        segment         public  'code'
                assume          ds0:sgroup
                public          sys_information

;/*****************************************************/
;/*              system information table               */
;/*****************************************************/

sys_information         label           far
kernel_p                dw              0000h
                        dw              0F800h
sys_info_p              dd              sys_sys
ram_info_p              dd              sys_ram
eram_info_p             dd              0
cpu_info_p              dd              sys_cpu
icu_info_p              dd              sys_icu
tcu_info_p              dd              sys_tcu
tsk_info_p              dd              sys_tsk

                        even
sys_cpu                 label           far
prc_reg                 db              0E0h

                        even
sys_icu                 label           far
imc_reg                 db              80h

                        even
sys_tcu                 label           far
interval                dw              1
stmc_reg                dw              0C35h
tim_int_pri             db              00h
bank_no                 db              0FFh


                        even
sys_sys                 label           far
exception               db              0
                        db              0
stack_size              dw              0100h

                        even
sys_ram                 label           far
```

リストB-17　CONFIG.ASM（2/2）

```
area_count      dw      1
area_1          dw      1000h
                dw      2000h

                even
sys_tsk         label   far
task_count      dw      1
task_1          dw      0100h   ; task id
                dw      0000h   ; task offset
                dw      0F010h  ; task segment
                dw      400h    ; task ds0
                dw      0400h   ; stack size
                dw      64h     ; priority
                dw      00000h  ; initial data

sys_conf        ends
                end
```

リストB-18　RESET.ASM

```
;
; reset.asm
;
extrn   start_up_entry:far

reset   segment public 'reset'
        br      far ptr start_up_entry
reset   ends
        end
```

リストB-19　STARTUP.LC

```
LOCATE (sys_conf : #FC000);
LOCATE (reset : #FFFF0);
```

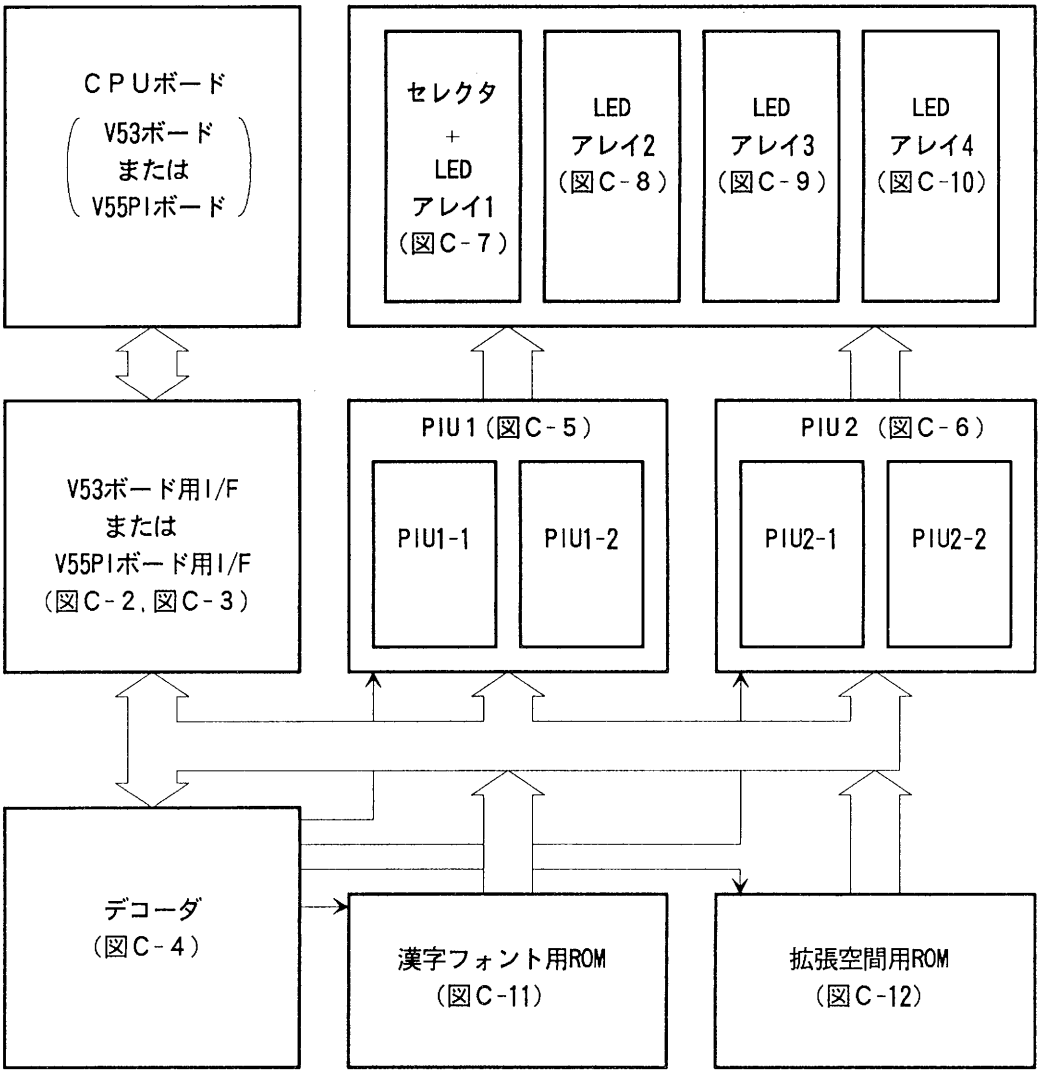
リストB-20　MAKESTA

```
ROOTS           = b:¥itools
FORM            = $(ROOTS)¥x¥form
GSMAP           = $(ROOTS)¥x¥gsmap
LLINK           = $(ROOTS)¥x¥llink
ASMV55          = $(ROOTS)¥x¥asmv55pi
MAK             = startup.lc

startup.hex     : startup.ab startup.map
        $(FORM) startup.ab -f s37 -o startup.hex

startup.map     : startup.ab
        $(GSMAP) startup.ab -o startup.map

startup.ab      : startup.ol reset.ol config.ol
        $(LLINK) startup.ol config.ol reset.ol -c $(MAK) -o startup.ab

config.ol       : config.asm
        $(ASMV55) config.asm

startup.ol      : startup.asm
        $(ASMV55) startup.asm

reset.ol        : reset.asm
        $(ASMV55) reset.asm

clean   :
        rm *.ol *.ab *.hex *.map
```

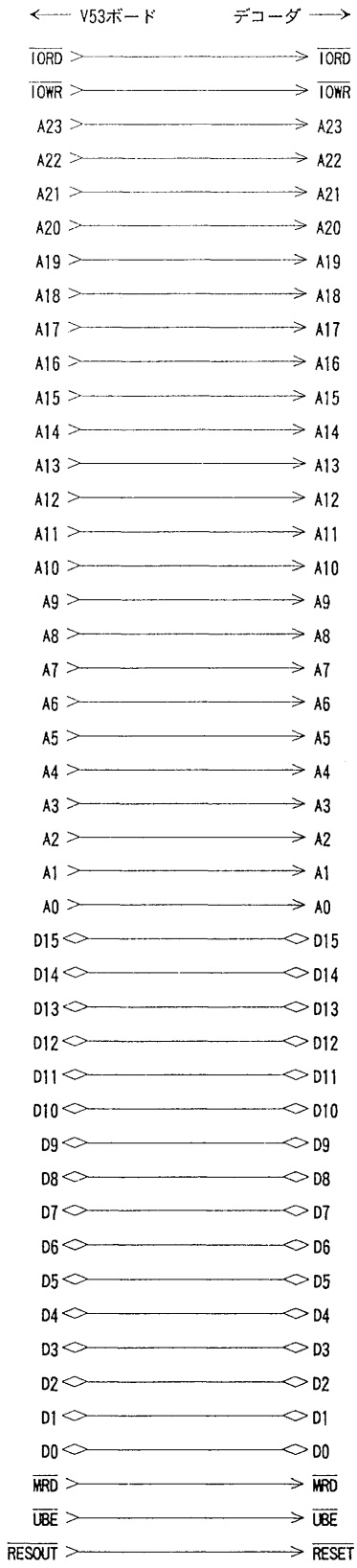
保守／廃止

付録Ｃ　回路図

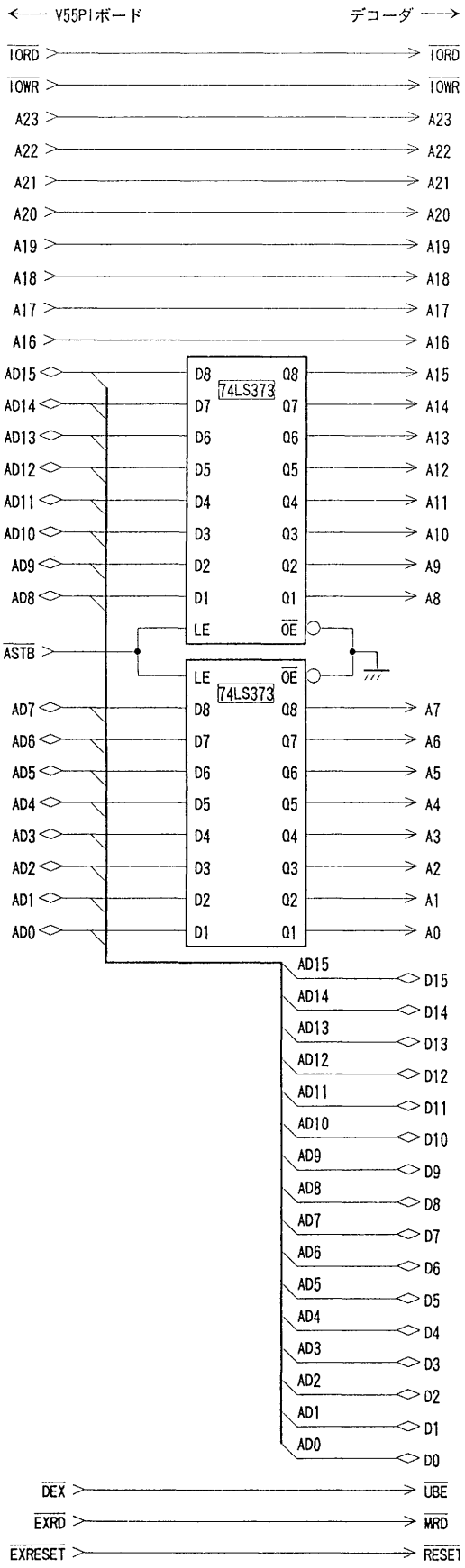
図Ｃ－１　ブロック図



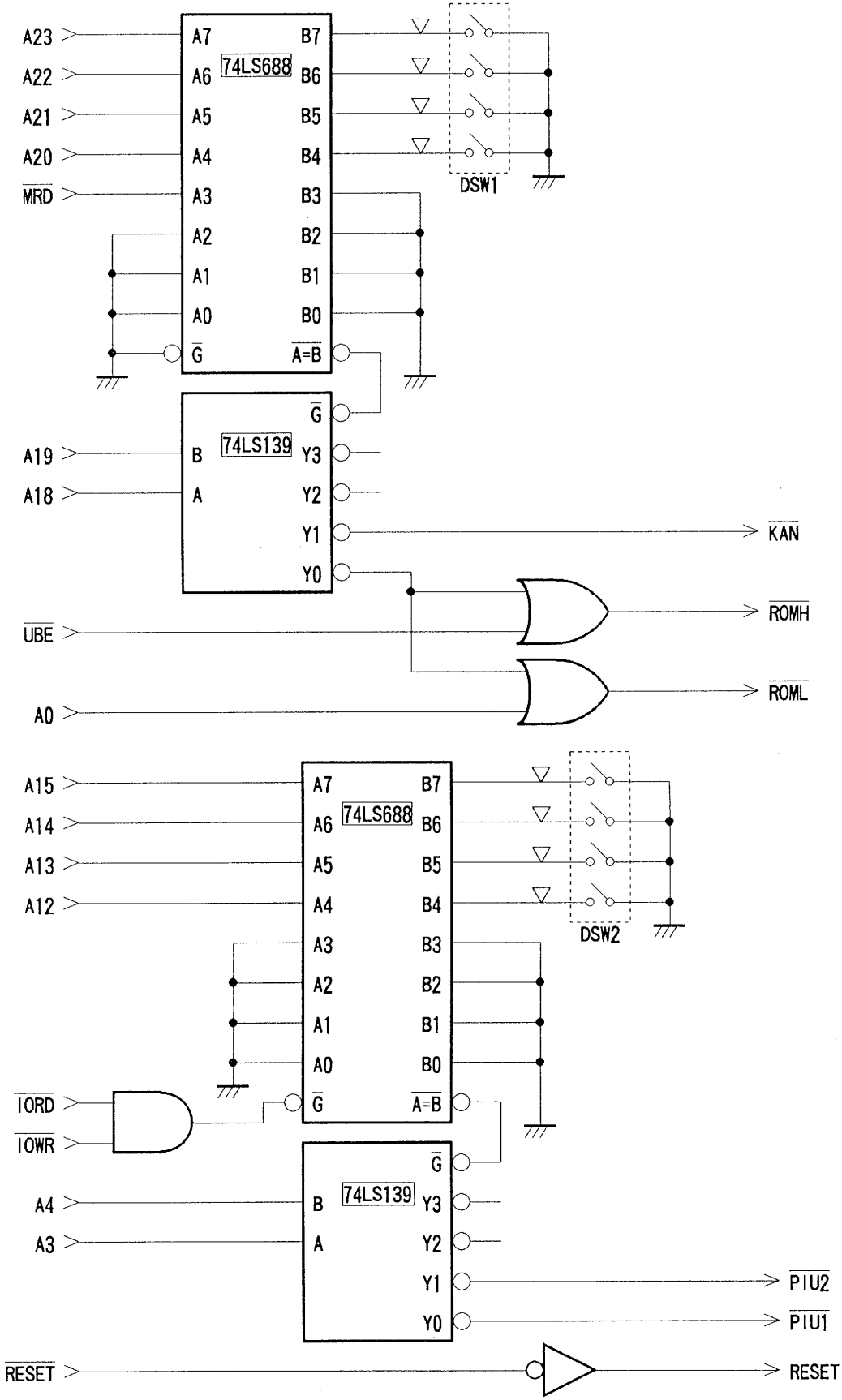
図C－2　V53ボード用I/F



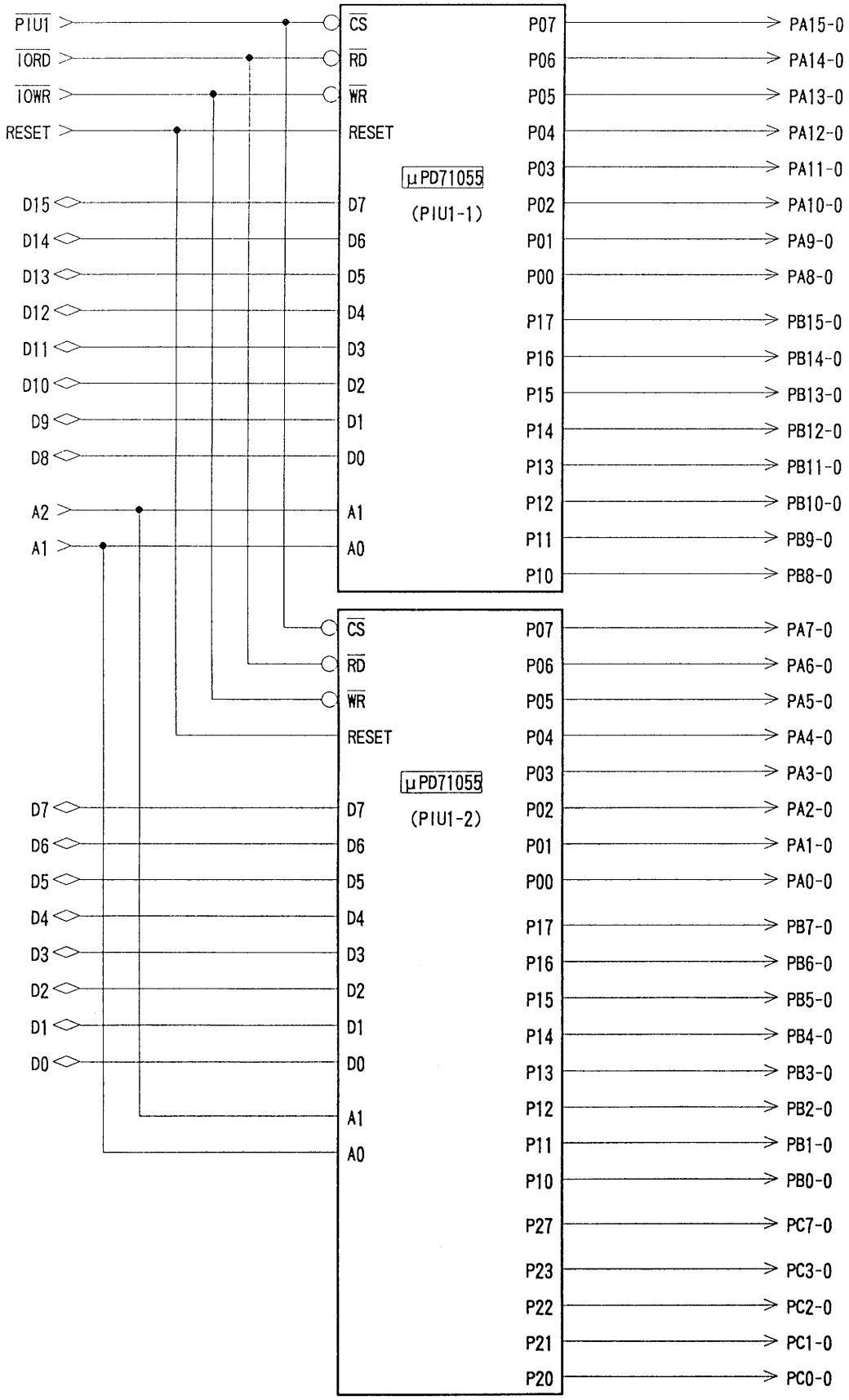
図C－3　V55PIボード用I/F



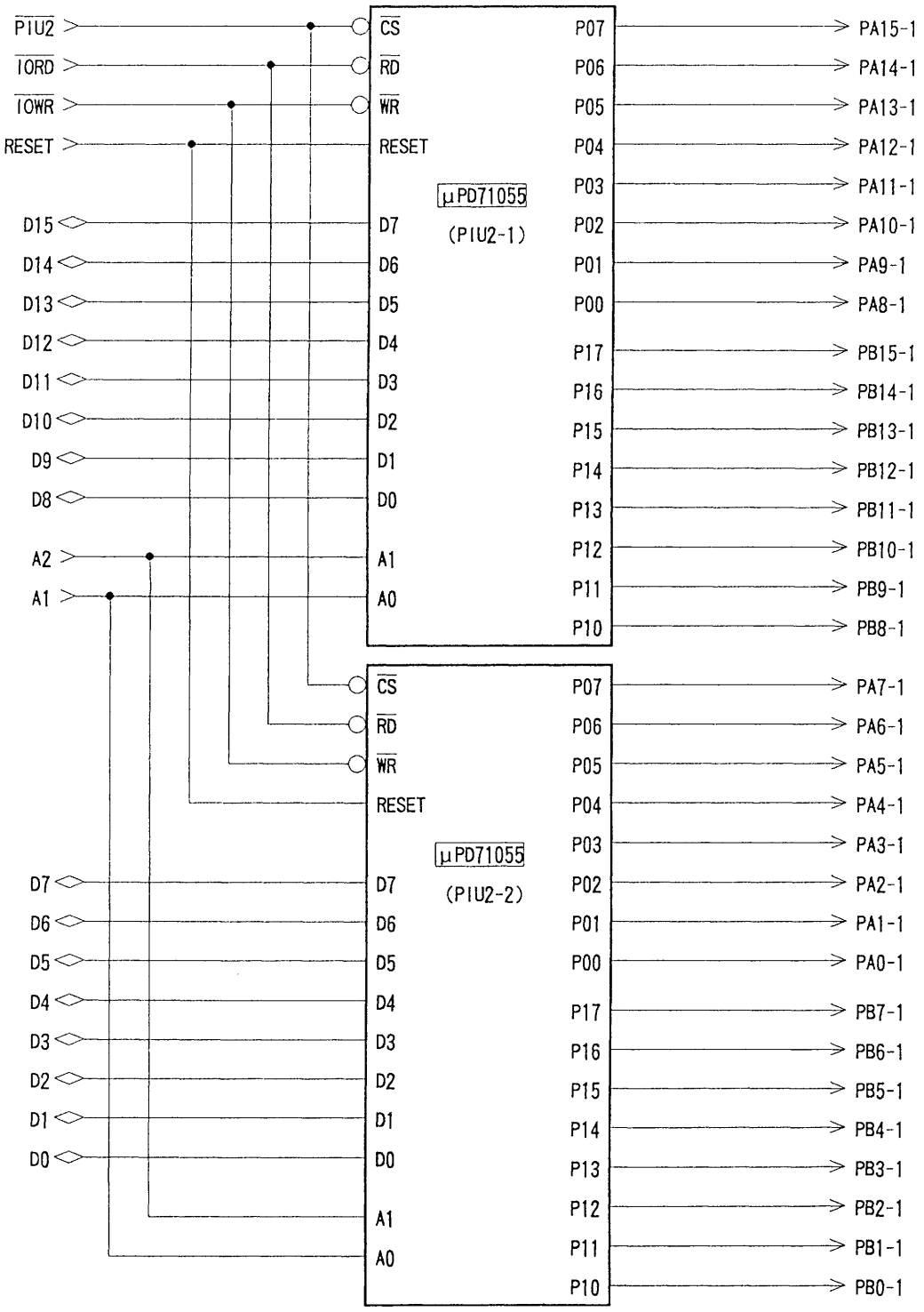
図C-4　デコーダ



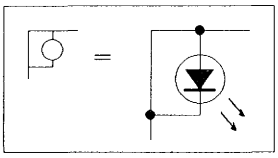
図C-5　PIU1

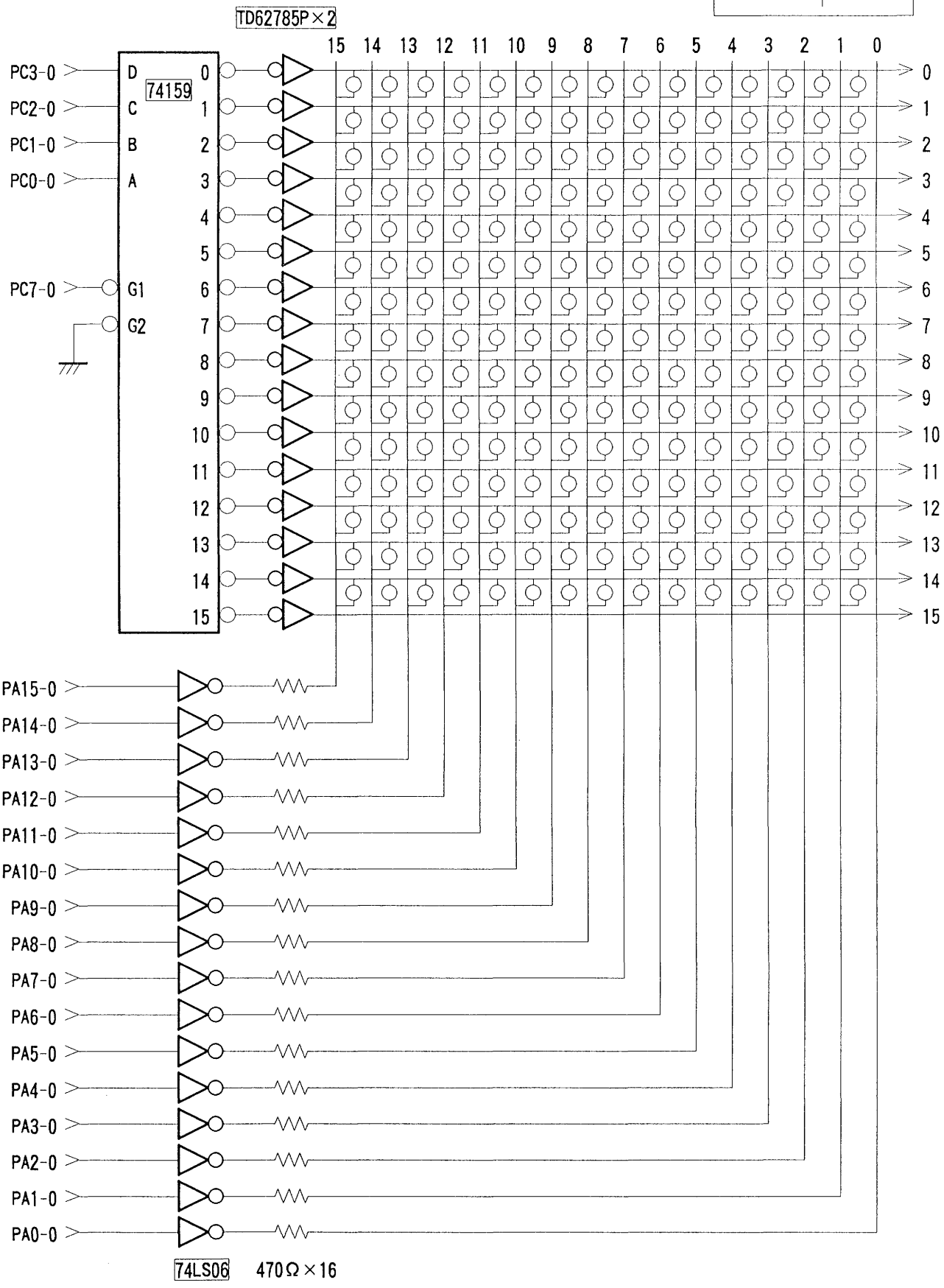


図C-6　PIU2

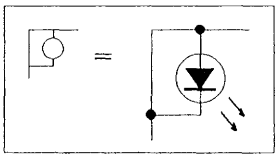


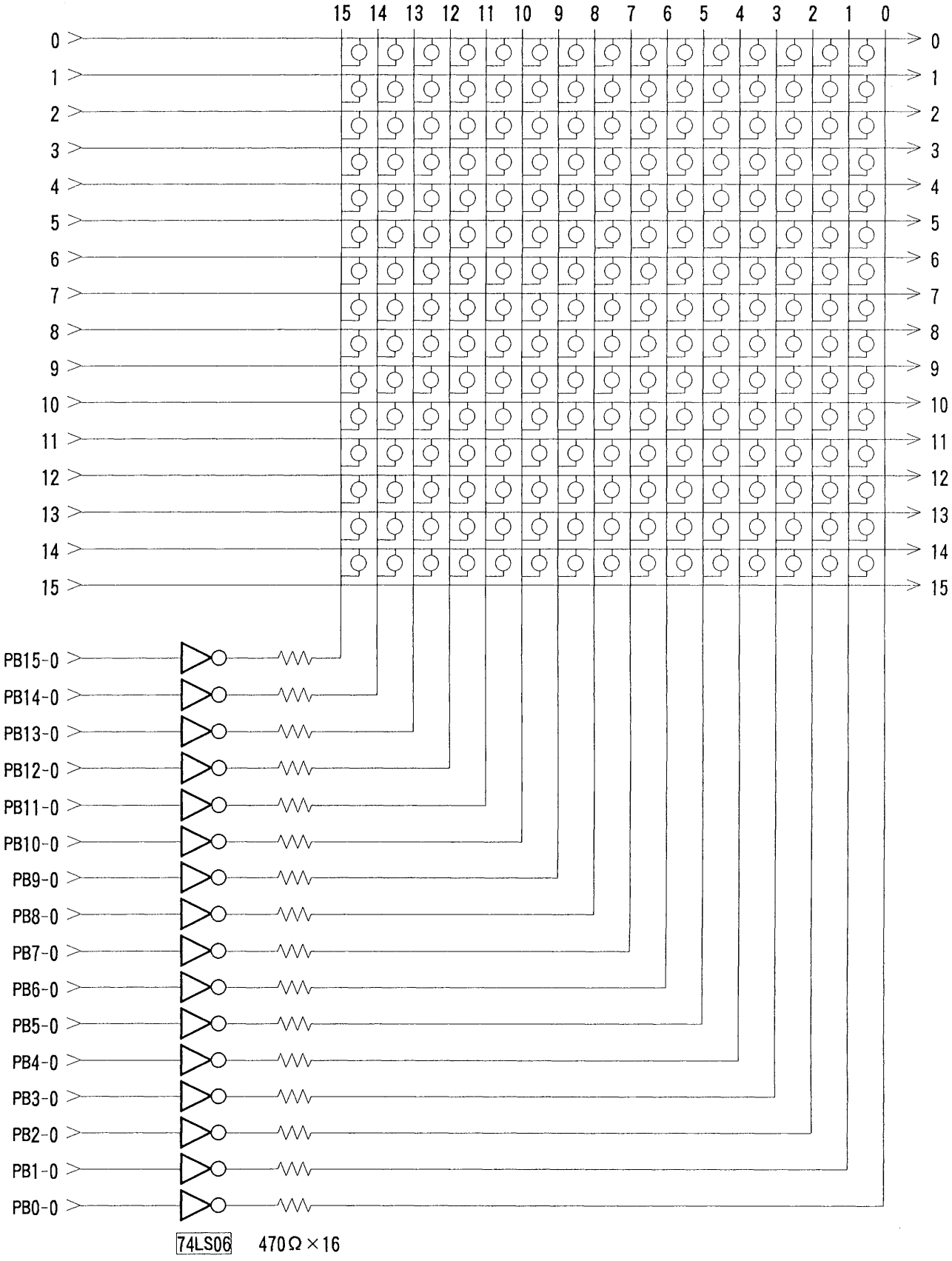
図C-7　セレクタとLEDアレイ1



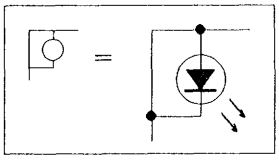


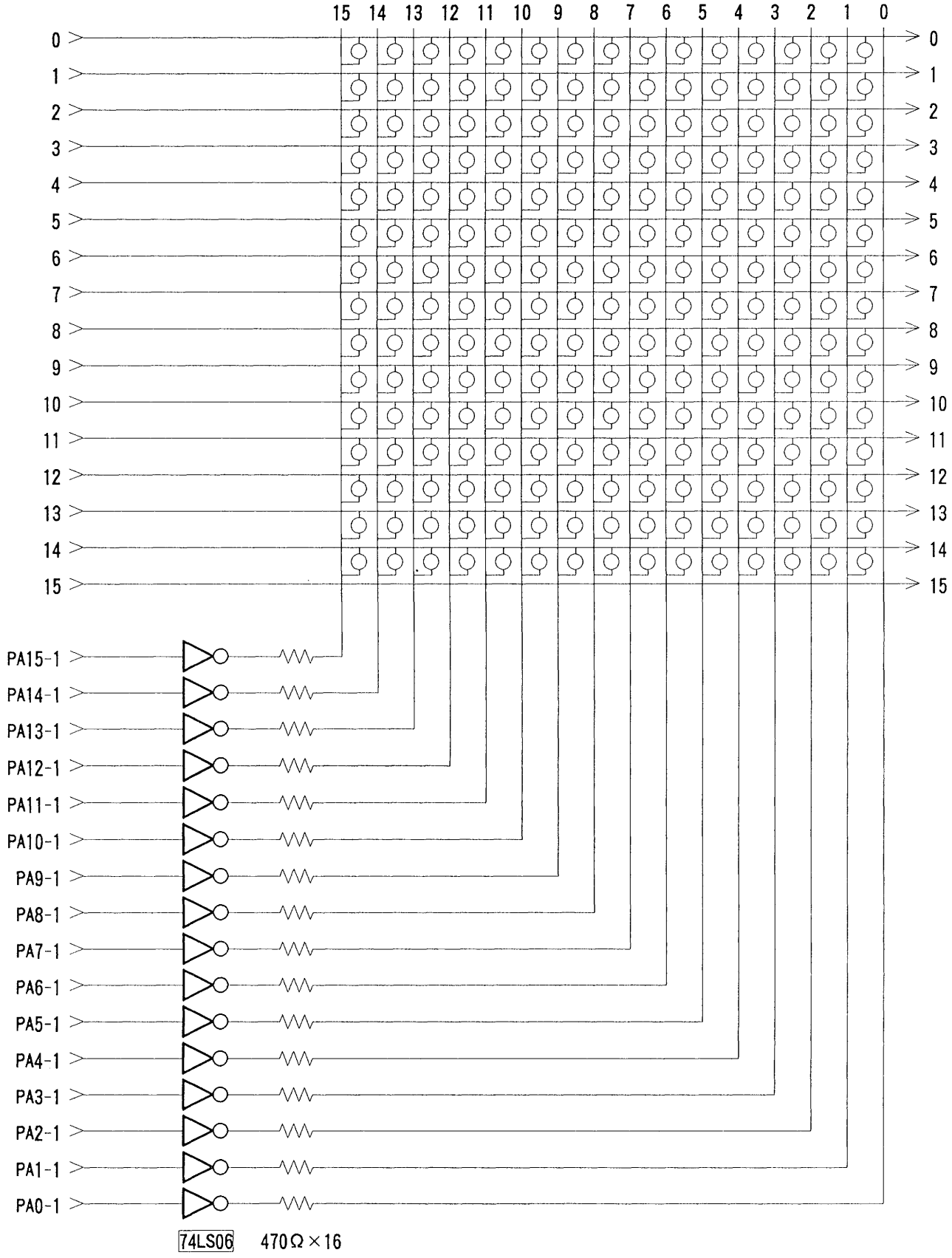
図C-8　LEDアレイ2



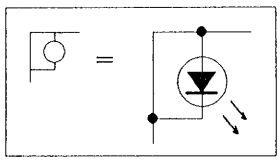


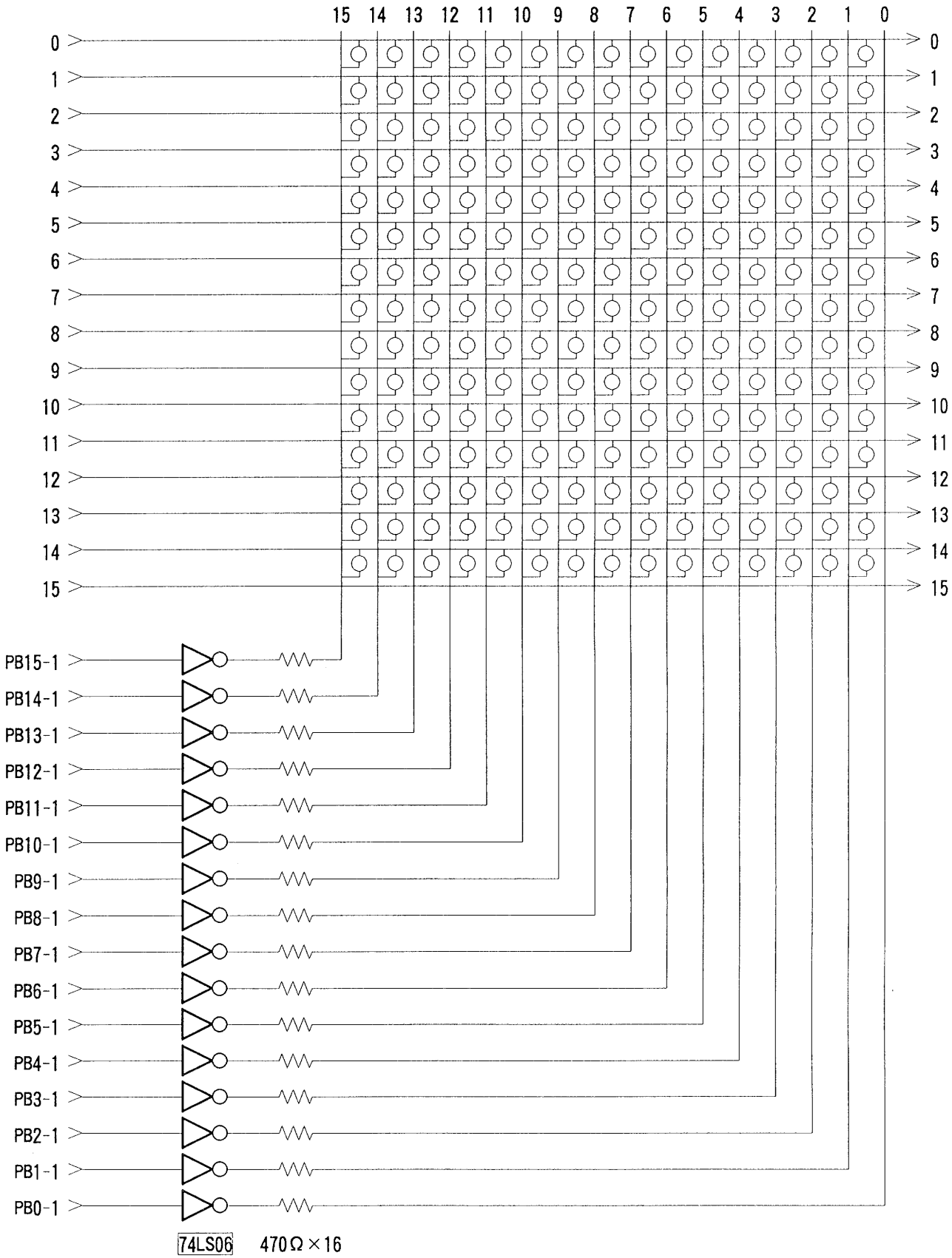
図C-9　LEDアレイ3



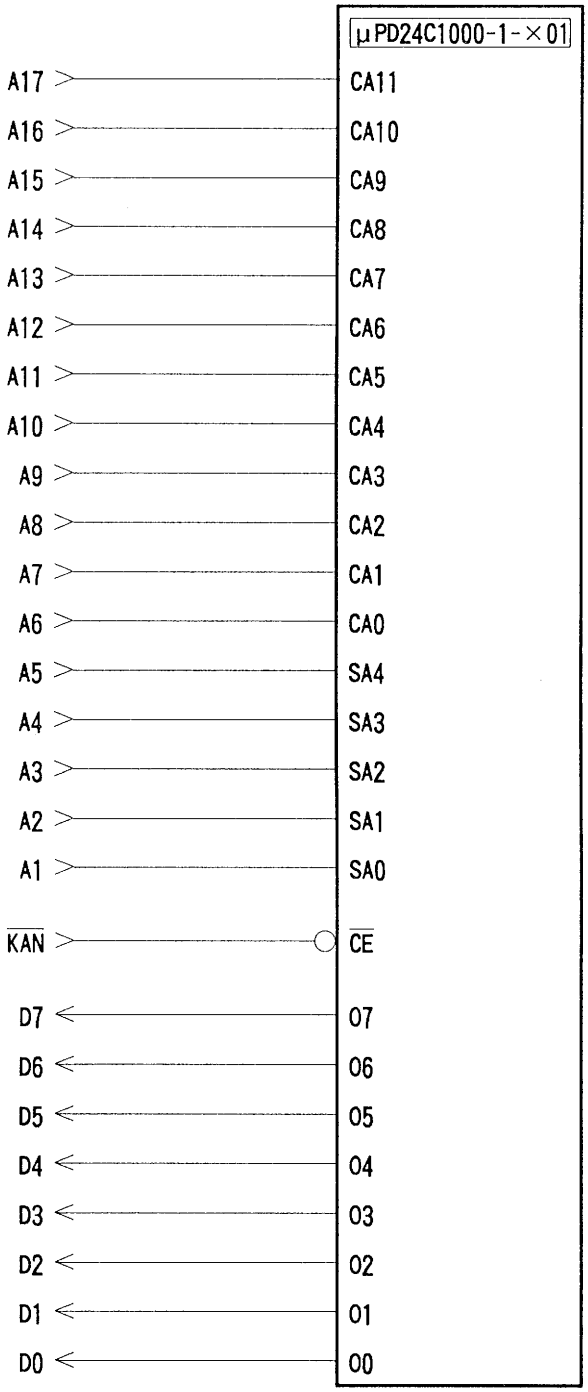


図C-10　LEDアレイ4

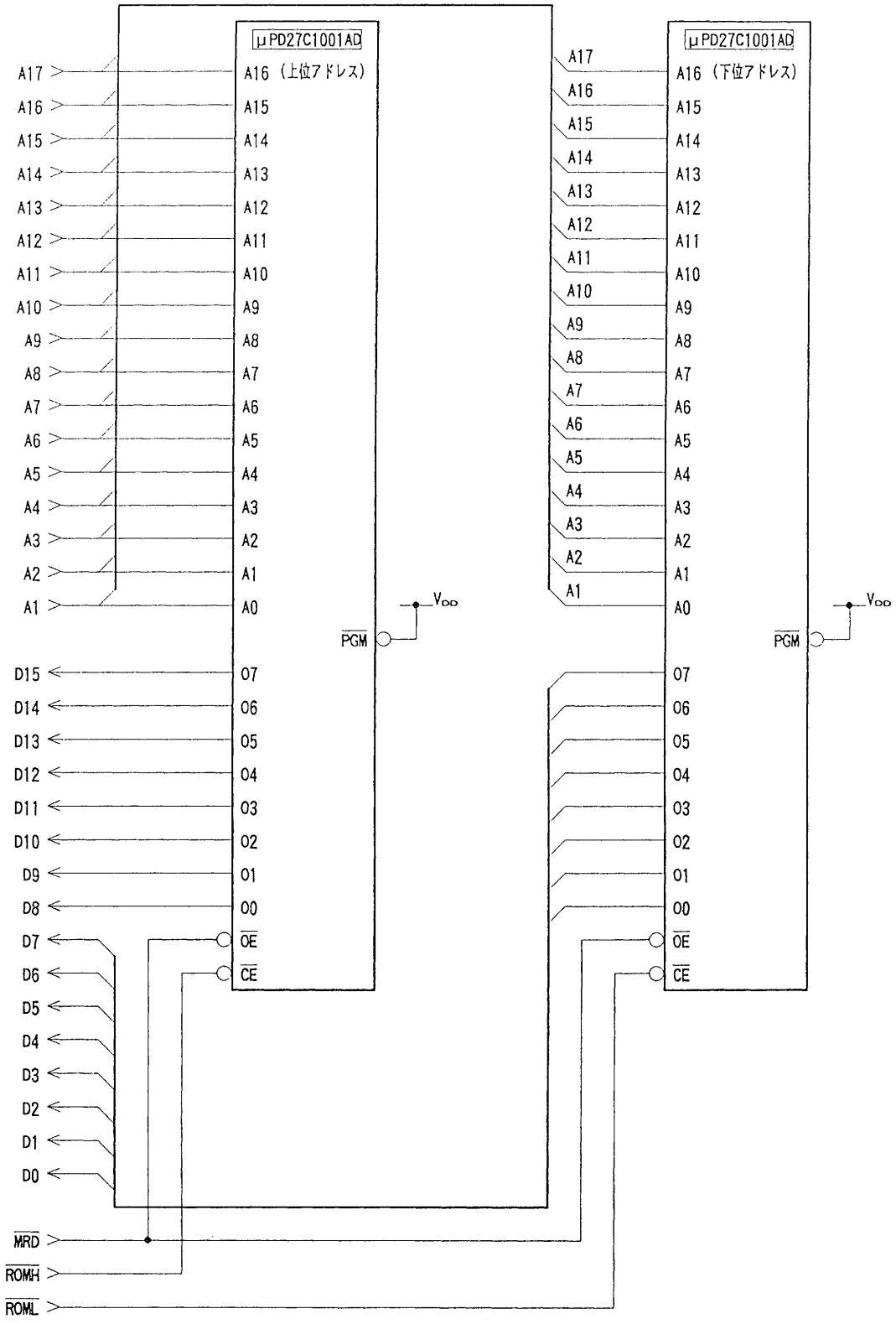




図C-11　漢字フォント用ROM



図C-12　拡張空間用ROM



保守／廃止

保守／廃止

付録D　アプリケーション・システムの使用例

この例では，ホストとしてPC-9800シリーズを使用します。
ホストには，あらかじめRSDRV.SYSを組み込んでください。
なお，通信データのフォーマットについては，**第4章　ソフトウエア**を参照してください。

D．1　転送手順の概要

転送方向	ホスト→ターゲット	ターゲット→ホスト
転送手順	①コマンド，モードの指定（送信） ②メッセージの送信（文字列データが必要な場合）	①参照する登録内容のモード指定（送信） ②登録内容の受信 ③登録内容の参照

D．2　ホストからターゲットへのデータ転送

(1)データの転送可否

コマンド，モード	新規登録	データ削除	表示モード変更
時刻	可（表示データが必要）	不可	可
ニュース	可 （表示回数の指定と 表示データが必要）	可	
広告			
緊急			
混在	不可	不可	

(2)転送の手順

①コマンド，モードの指定（送信）

COPYA ファイル名 AUX

COPYAはMS-DOSのコマンドです。ファイルはコマンドかモード・データです。

②メッセージの送信（文字列データが必要な場合）

COPYA ファイル名 AUX

COPYAはMS-DOSのコマンドです。ファイルはメッセージ・データです。

(3)使用例

時刻設定	コマンド	新規登録
	モード	時刻モード
	登録データ	17:45
	TIME_C.DAT	P0000000000
	TIME_D.DAT	{STX}17:45{ETX}
	転送	COPYA TIME_C.DAT AUX COPYA TIME_D.DAT AUX
	備考	転送後，ターゲットの時刻は17:45になります。
緊急メッセージ登録	コマンド	新規登録
	モード	緊急モード
	表示回数	16回
	登録データ	緊急事態発生!!
	EMG_C.DAT	P0400000010
	EMG_D.DAT	{STX}緊急事態発生!!{ETX}
	転送	COPYA EMG_C.DAT AUX COPYA EMG_D.DAT AUX
	備考	転送後，ターゲット（LEDアレイ）に緊急メッセージを16回表示します。
広告データ削除	コマンド	データ削除
	モード	広告モード
	CM_C.DAT	P1200000000
	転送	COPYA CM_C.DAT AUX
	備考	転送後，広告データをすべて削除します。
表示モード変更	コマンド	表示モード変更
	モード	混在モード
	MIX_C.DAT	P2800000000
	転送	COPYA MIX_C.DAT AUX
	備考	転送後，表示モードは「混在表示モード」になります。

備考　{STX}は文字コードの2H，{ETX}は3Hを表します。

D．3　ターゲットからホストへのデータ転送

(1)データの転送可否

コマンド，モード	表示内容表示
時刻	可
ニュース	
広告	
緊急	
混在	不可

(2)転送の手順

①参照する登録内容のモード指定（送信）

COPYA ファイル名 AUX

COPYAはMS-DOSのコマンドです。ファイルはコマンドかモード・データです。

②登録内容の受信（受信したデータをファイルに格納）

COPY AUX ファイル名

COPYAはMS-DOSのコマンドです。ファイルは受信データ格納用です。

③登録内容の参照

受信データの格納されたファイルをエディタ，またはTYPEコマンドによって参照します。

(3)使用例

ターゲット側の時刻を参照する	コマンド	登録内容参照
	モード	時刻モード
	TIME_C.DAT	P8000000000
	転送	COPYA TIME_C.DAT AUX COPY AUX PRT.DAT
	参照	TYPE PRT.DAT T10時00分{EOF}
現在のニュース・データを参照する	コマンド	登録内容参照
	モード	ニュース・モード
	NEWS_C.DAT	P8200000000
	転送	COPYA NEWS_C.DAT AUX COPY AUX PRT.DAT
	参照	TYPE　PRT.DAT {STX}明日は晴天です{ETX} {STX}海外旅行者の数が急激に増加しています{ETX}{EOF}

備考　{STX}は文字コードの2Hを，{ETX}は3Hを{EOF}は1AHをそれぞれ表します。

保守／廃止

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］ RX136,RX423　アプリケーション・ノート　LEDアレイ表示システム編
（ EEA-617（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，NEC半応技本部員，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。

NEC半導体インフォメーションセンター
FAX：(044)548-7900

キリトリ

保守／廃止

保守／廃止

お問い合わせは，最寄りのNECへ

【営業関係お問い合わせ先】

コンシューマ半導体販売事業部 OA半導体販売事業部 インダストリ半導体販売事業部	〒108-01 東京都港区芝五丁目7番1号 (NEC本社ビル)	東京	(03)3454-1111 (大代表)
中部支社 半導体販売部	〒460 名古屋市中区栄四丁目14番5号 (松下中日ビル)	名古屋	(052)242-2755
関西支社 半導体第一販売部 関西支社 半導体第二販売部 関西支社 半導体第三販売部	〒540 大阪市中央区城見一丁目4番24号 (NEC関西ビル)	大阪 大阪 大阪	(06) 945-3178 (06) 945-3200 (06) 945-3208

北海道支社	札幌 (011)231-0161	小山支店	小山 (0285)24-5011	福井支店	福井 (0776)22-1866
東北支社	仙台 (022)261-5511	長野支社	長野 (0262)35-1444	富山支店	富山 (0764)31-8461
岩手支店	盛岡 (0196)51-4344	松本支店	松本 (0263)35-1666	京都支社	京都 (075)344-7824
山形支店	山形 (0236)23-5511	上諏訪支店	諏訪 (0266)53-5350	神戸支社	神戸 (078)332-3311
郡山支店	郡山 (0249)23-5511	甲府支店	甲府 (0552)24-4141	中国支社	広島 (082)242-5504
いわき支店	いわき (0246)21-5511	埼玉支社	大宮 (048)641-1411	鳥取支店	鳥取 (0857)27-5311
長岡支店	長岡 (0258)36-2155	立川支社	立川 (0425)26-5981	岡山支店	岡山 (086)225-4455
水戸支店	水戸 (0292)26-1717	千葉支社	千葉 (043)238-8116	四国支社	高松 (0878)36-1200
神奈川支社	横浜 (045)324-5511	静岡支社	静岡 (054)255-2211	新居浜支店	新居浜 (0897)32-5001
群馬支店	高崎 (0273)26-1255	沼津支店	沼津 (0559)63-4455	松山支店	松山 (0899)45-4111
太田支店	太田 (0276)46-4011	浜松支店	浜松 (053)452-2711	九州支社	福岡 (092)271-7700
宇都宮支店	宇都宮 (0286)21-2281	北陸支社	金沢 (0762)23-1621	北九州支店	北九州 (093)541-2887

【本資料に関する技術お問い合わせ先】

半導体応用技術本部 マイクロコンピュータ技術部	〒210 川崎市幸区塚越三丁目484番地	川崎	(044)548-7950	半導体 インフォメーションセンター FAX(044)548-7900 (FAXにてお願い致します)
半導体応用技術本部 中部応用システム技術部	〒460 名古屋市中区栄四丁目14番5号 (松下中日ビル)	名古屋	(052)242-2762	
半導体応用技術本部 西日本応用システム技術部	〒540 大阪市中央区城見一丁目4番24号 (NEC関西ビル)	大阪	(06) 945-3383	

C93.11