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。

ユーザーズ・マニュアル

MX78K0

オペレーティング・システム

基礎編（Windows™ベース）

対象デバイス
78K/0シリーズ

資料番号　U12257JJ2V1UMJ1（第2版）
発行年月　January 2001 N　CP(K)

保守／廃止

〔メ　モ〕

保守／廃止

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

その他，記載の会社名／製品名は，各社の商標あるいは登録商標です。

TRON**は，**The Realtime Operating system Nucleus**の略称です。**

ITRON**は，**Industrial TRON**の略称です。**

- **本資料の内容は予告なく変更することがありますので，最新のものであることをご確認の上ご使用ください。**
- **文書による当社の承諾なしに本資料の転載複製を禁じます。**
- **本資料に記載された製品の使用もしくは本資料に記載の情報の使用に際して，当社は当社もしくは第三者の知的財産権その他の権利に対する保証または実施権の許諾を行うものではありません。上記使用に起因する第三者所有の権利にかかわる問題が発生した場合，当社はその責を負うものではありませんのでご了承ください。**
- **本資料に記載された回路，ソフトウエア，及びこれらに付随する情報は，半導体製品の動作例，応用例を説明するためのものです。従って，これら回路・ソフトウエア・情報をお客様の機器に使用される場合には，お客様の責任において機器設計をしてください。これらの使用に起因するお客様もしくは第三者の損害に対して，当社は一切その責を負いません。**

M7A　98.8

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。

はじめに

このたびは、NEC 78K/0 シリーズの組み込み用ソフトウェアである『78K/0 シリーズ用 OS MX78K0』をお買上げいただきまして誠にありがとうございます。

本マニュアルは、78K/0 シリーズ用 OS　MX78K0 の機能を、正しく理解していただくことを目的として書かれています。

《 対象者 》

本マニュアルは、デバイスのユーザーズ・マニュアル一読程度の知識があり、ソフトウエア・プログラミングの経験がある方を対象として書かれています。

ただし、本パッケージは OS 単体であるため、実際に本 OS をご使用になる場合には、“CC78K0 C コンパイラ”と“RA78K0　アセンブラ・パッケージ”が必要になります。

保守／廃止

《 構成 》

本マニュアルの構成を以下に示します。

第１章　概要
　本 OS の概要、提供ファイル、特徴などについて説明します。

第２章
　本 OS のインストール方法と提供ファイルのディレクトリ構造、ファイルの説明をします。

第３章　タスク管理
　本 OS が管理するタスクについて説明します。

第４章　イベントフラグ
　イベントフラグについて説明します。

第５章　時間管理
　本 OS がサポートする時間管理について説明します。

第６章　割り込み機能
　割り込みについて説明します。

第７章　システムコール
　本 OS が提供しているシステムコールについて説明します。

第８章　OS の管理領域
　本 OS が使用するメモリ用量の見積もり方法を説明します。

第９章　アプリケーション開発手順
　本 OS を使用するための注意点、タスクの記述方法などを説明します。

第１０章　システム初期化処理
　本 OS が使用する RAM の定義方法を説明します。

付録１　予約語一覧

付録２　システムコール別使用スタックサイズ一覧

保守／廃止

《 凡例 》

本マニュアル中で共通に使用される記号などの意味を示します。

…　　：同一の形式を繰り返す
[ ]　　：[ ]内は省略可能
「 」　：「 」で囲まれた文字そのもの
“ ”　　：“ ”で囲まれた文字そのもの
‘ ’　　：‘ ’で囲まれた文字そのもの
( )　　：( )で囲まれた文字そのもの
太文字：文字そのもの
＿　　：重要箇所、使用例での下線は入力文字
△　　：1文字以上の空白
：　　：プログラム記述の省略形
／　　：区切り記号
＼　　：バックスラッシュ

《 関連資料 》

本マニュアルに関連する資料(ユーザーズ・マニュアルなど)を紹介します。

資料名	資料番号
RA78K アセンブラ・パッケージ 言語編	EEU-815
RA78K アセンブラ・パッケージ 操作編	EEU-809
RA78K シリーズ 構造化アセンブラ・プリプロセッサ	EEU-817
RA78K0 アセンブラ・パッケージ アセンブリ言語編 Ver.3.10 以上	U11801JJ1V0UM00
RA78K0 アセンブラ・パッケージ 操作編 Ver.3.10 以上	U11802JJ1V0UM00
RA78K0 構造化アセンブリ言語編 Ver.3.10 以上	U11789JJ1V0UM00
CC78K0 シリーズ Cコンパイラ 言語編	U11518JJ1V0UM00
CC78K0 シリーズ Cコンパイラ 操作編	U11517JJ1V0UM00
78K/0 シリーズ 命令編	IEU-849
idea-L スクリーン・エディタ 入門編	U10094JJ1V0UM00

保守／廃止

目次

図目次

表目次

保守／廃止

第1章 概要

本 OS は、リアルタイム OS よりも使用可能なメモリ資源が限られている分野におけるソフトウエアの開発に対応するための 78K/0 シリーズ用の OS です。本 OS は、78K シリーズのリアルタイム OS である RX78K シリーズへの移行性のため、μ ITRON 仕様のサブセット仕様となっています。

1.1 構成

図 1.1にシステムの構成を示します。

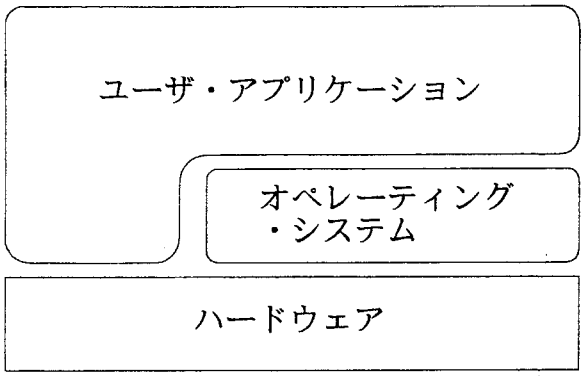


図 1.1：システム構成図

1.2 特徴

78K/0 シリーズ用の OS である本 OS の特徴を以下に示します。

1．高速かつコンパクト
　OS の仕様を、μ ITRON のサブセットとすることにより、高速で、コンパクトな作りになっています。

2．タスクの WAIT 状態の削除
　OS によるメモリの使用量を軽減するために、タスクの WAIT 状態（タスクの待ち状態）を削除しました。これにより、OS が使用する RAM を節減しました。

3．プリエンプション機能の削除
　OS によるメモリの使用量を軽減するために、プリエンプション（タスクの強制中断）を削除することにより、OS が使用する RAM を節減しました。これにより、タスクは ext_tsk() システムコールが発行されるまで止まることなく動作します。

4．スリム・タイプとディバグ・タイプ
　本 OS には、システムコール発行後、エラー検出を行わないスリム・タイプの OS と、エラー検出を行うディバグ・タイプの OS の 2 種類を用意しました。
　必要に応じて、スリム・タイプかディバグ・タイプを使い分けてアプリケーションの開発を行うことができます。

1.3 機能概要

本 OS は、以下の機能を提供しています。

◎ アプリケーション・タスクから発行されたシステムコールに対応した各機能をサービスします。

◎ 各タスクの実行順序を管理し、次に実行するタスクへ切り替え処理を行います。

本 OS には、以下の管理機能を提供しています。

1．タスク管理
本 OS の処理単位であるタスクの起動・終了処理を管理します。

2．イベント管理
システム内外で起きた事象に対応するタスクを起動するためのイベントを管理します。

3．時間管理
指定時間経過後に、タスクを起動するためのタイマを管理します。

4．割り込み管理
割り込みを事象として駆動される処理を管理します。

1.4 実行環境

以下に対応する CPU を示します。

・78K/0 シリーズ

1.5 アプリケーションの開発環境

アプリケーションを開発するシステム環境を以下に示します。

1．ハードウエア
・PC-9801，9821 シリーズ（MS-DOS）
・IBM-PC シリーズ（PC-DOS）　　（予定）

2．ソフトウエア
・アセンブラ・パッケージ
　NEC 製　RA78K0（78K/0 用）
・コンパイラ・パッケージ
　NEC 製　CC78K0（78K/0 用）　Ver.3.0 以上

本製品に添付されている ｉｄｅａ－Ｌ用ファイルを使用して開発を行う場合は、上記の環境に加えて以下のソフトウェアが必要です。

MS-Windows　Ver.3.1 以上
NEC 製　ｉｄｅａ－Ｌ　スクリーン・エディタ　Ver.3.1 以上

第2章　インストール

この章では、本 OS のホスト・マシンへのインストールについて述べます。

2.1　提供ファイル

本 OS の提供形式は、オブジェクト・ファイル形式とソース・ファイル形式の 2 種類あります。ソース・ファイルに関しては、量産用オブジェクト購入時のみ提供できます。以下にオブジェクト・ファイル形式とソース・ファイル形式の提供ディレクトリ構成を示します。

2.1.1　オブジェクト・ファイル形式のディレクトリ構成

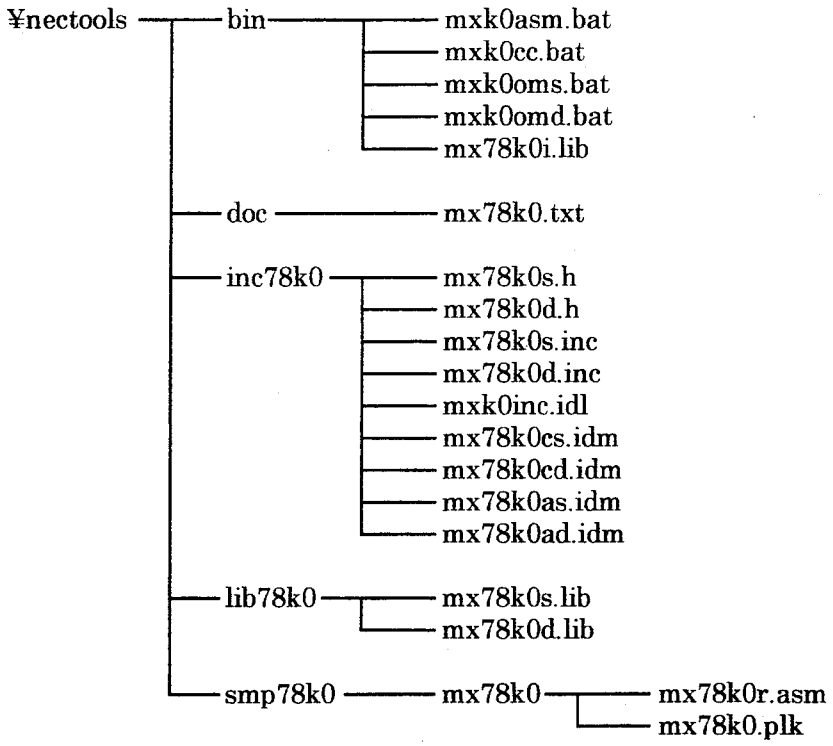


図 2.1　オブジェクト・ファイル形式のディレクトリ構成

1．¥nectools¥bin ディレクトリ

メイク用の各種バッチ・ファイル、ｉｄｅａ－L用のライブラリ・ファイルが入っています。

ファイル名　mxk0asm.bat
内容　ユーザ・タスク、メモリ定義ファイルをアセンブルする際に使用するバッチ・ファイル
使用方法　ユーザ・タスク、メモリ定義ファイルをアセンブルする際に使用します。アセンブル・オプションを変更したい場合は本ファイルの内容を変更してください。
本ファイルを使用するときは、引数としてチップ種別とファイル名（拡張子は省略）を指定してください。
例）mxk0asm△014△usr1（ターゲット CPU がμPD78014、ファイル名が usr1.asm)

ファイル名　mxk0cc.bat
内容　　ユーザ・タスクをコンパイルする際に使用するバッチ・ファイル
使用方法　　ユーザ・タスクをコンパイルする際に使用します。コンパイル・オプションを変更したい場合は、本ファイルの内容を変更してください。
本ファイルを使用するときは、引数としてチップ種別とファイル名（拡張子は省略）を指定してください。
例）mxk0cc△014△usr1　(ターゲット CPU がμPD78014、ファイル名が usr1.asm)

ファイル名　mxk0oms.bat
内容　　スリム・タイプのロード・モジュールを作成する際に使用するバッチ・ファイル
使用方法　　“mx78k0s.lib”とその他の必要なファイル（”mx78k0.plk”に記述）をリンクし、ロード・モジュールを作成します。リンカ・オプションを変更したい場合は本ファイルの内容を変更してください。
本ファイルを使用するときは、引数としてチップ種別を指定してください。
例）mxk0oms△014　(ターゲット CPU がμPD78014)

ファイル名　mxk0omd.bat
内容　　ディバグ・タイプのロードモジュールを作成する際に使用するバッチ・ファイル
使用方法　　“mx78k0d.lib”とその他の必要なファイル（”mx78k0.plk”に記述します）をリンクし、ロード・モジュールを作成します。リンカ・オプションを変更したい場合は本ファイルの内容を変更してください。
本ファイルを使用するときは、引数としてチップ種別を指定してください。
例）mxk0omd△014　(ターゲット CPU がμPD78014)

ファイル名　mx78k0i.lib
内容　　ｉｄｅａ－Ｌ用関数ライブラリ・ファイル
使用方法　　本ファイルの使用方法は、「ｉｄｅａ－Ｌ　スクリーン・エディタ　入門編」を参照してください。

2．¥nectools¥doc ディレクトリ

ユーザーズ・マニュアルに記載されていない情報や、注意事項等が書かれたファイルが入っています。

ファイル名　mx78k0.txt
内容　　ユーザーズ・マニュアルに記載されていない情報や、注意事項等が書かれているテキスト・ファイルです。

3．¥nectools¥inc78k0 ディレクトリ

アセンブリ言語用およびＣ言語用のインクルード・ファイルが入っています。

ファイル名　mx78k0s.h，mx78k0cs.idm（ｉｄｅａ－Ｌ用）
内容　　スリム・タイプ用インクルード・ファイル（Ｃ言語）
使用方法　　システムコールの型宣言が書かれていますので、Ｃ言語でソースを記述する際に本ファイルをインクルードして使用してください。本ファイルを使用する際は、ユーザ・システムで使用するシステムコールの型宣言以外はコメント文にするか、削除してください。

ファイル名　mx78k0d.h，mx78k0cd.idm（ｉｄｅａ－Ｌ用）
内容　　ディバグ・タイプ用インクルード・ファイル（Ｃ言語）
使用方法　　リターン・コードを定義していたり、システムコールの型宣言が書かれていますので、Ｃ言語でソースを記述する際に本ファイルをインクルードして使用してください。本ファイルを使用する際は、ユーザ・システムで使用するシステムコールの型宣言以外はコメント文にするか、削除してください。

ファイル名　mx78k0s.inc，mx78k0as.idm（ｉｄｅａ－Ｌ用）
内容　　スリム・タイプ用インクルード・ファイル（アセンブリ言語）
使用方法　　システムコールのマクロが書かれていますので、アセンブリ言語ソースを記述する際に、本ファイルをインクルードして使用してください。本ファイルを使用する際、使用しないシステムコールのマクロはコメント文に変更してください。

ファイル名　mx78k0d.inc，mx78k0ad.idm（ｉｄｅａ－Ｌ用）
内容　　ディバグ・タイプ用インクルード・ファイル（アセンブリ言語）
使用方法　　リターン・コードを定義していたり、システムコールのマクロが書かれていますので、アセンブリ言語ソースを記述する際に、本ファイルをインクルードして使用してください。本ファイルを使用する際、使用しないシステムコールのマクロはコメント文に変更してください。

ファイル名　mxk0inc.idl
内容　　同一ディレクトリ内のｉｄｅａ－Ｌ用ソース・ファイル（拡張子.idm）をｉｄｅａ－Ｌが管理するための情報ファイル
使用方法　　本ファイルの使用方法は、「ｉｄｅａ－Ｌ　スクリーン・エディタ　入門編」を参照してください。

4．¥nectools¥lib78k0 ディレクトリ

ニュークリアスのライブラリ・ファイルが入っています。

ファイル名　mx78k0s.lib
内容　　スリム・タイプ用のライブラリ・ファイル
使用方法　　ロードモジュール作成の際に一緒にリンクしてください。

ファイル名　mx78k0d.lib
内容　　ディバグ・タイプ用のライブラリ・ファイル
使用方法　　ロードモジュール作成の際に一緒にリンクしてください。

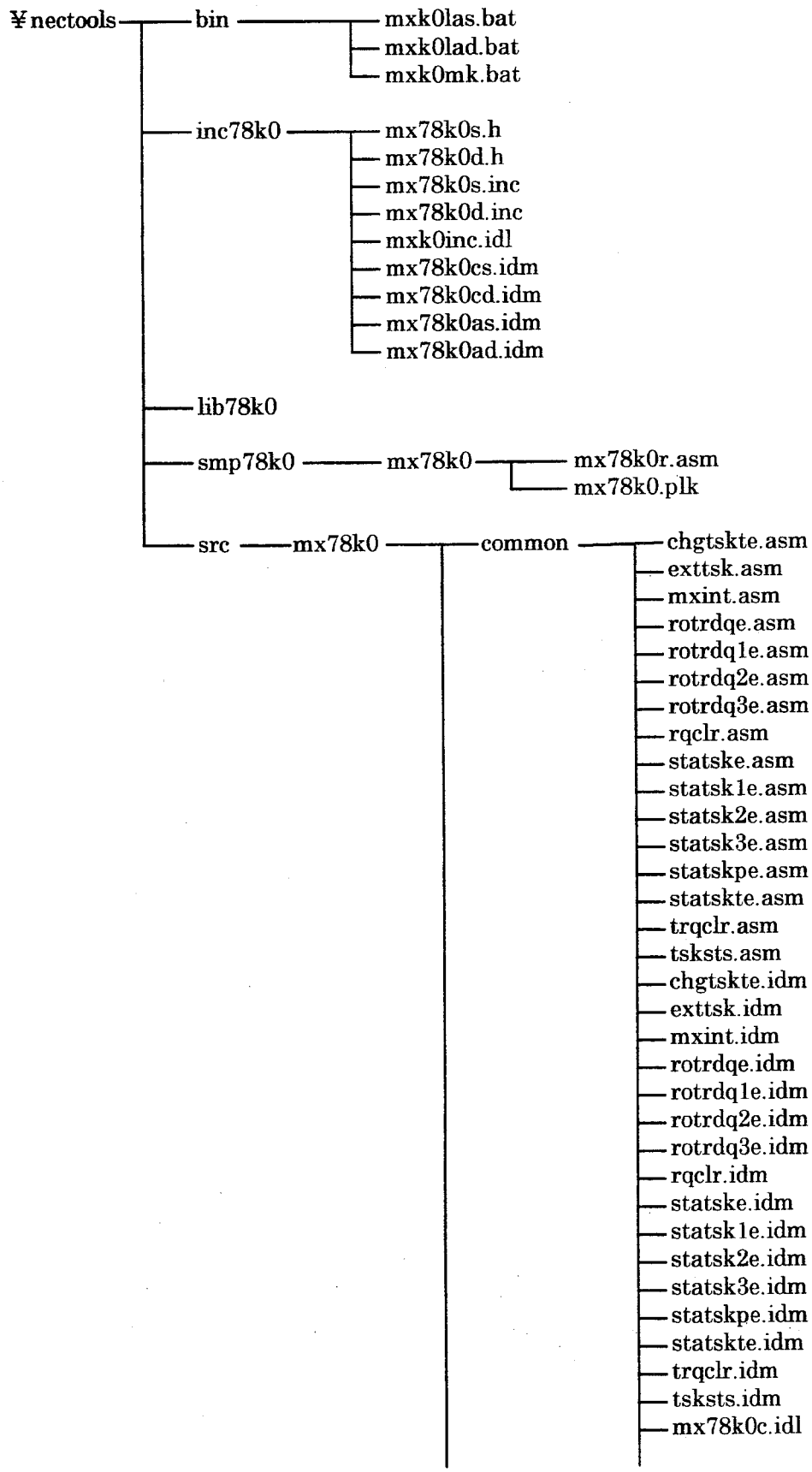
5．¥nectools¥smp78k0¥mx78k0 ディレクトリ

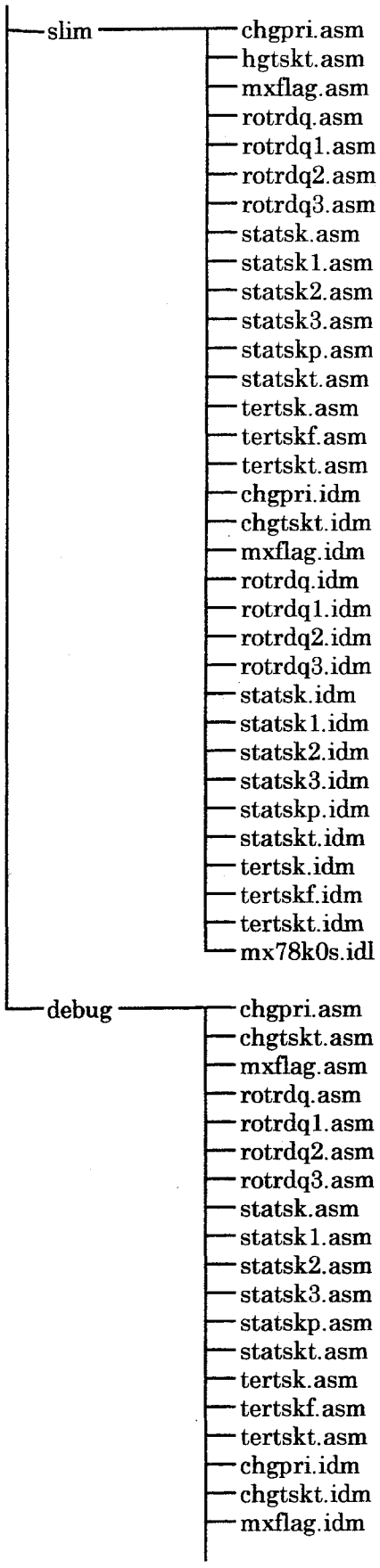
各種サンプル・ファイルが入っています。

ファイル名　mx78k0r.asm
内容　　メモリ設定ファイルのサンプル・ファイル
使用方法　　OS が使用するメモリを設定するファイルです。本ファイルはユーザ・システムに合わせて変更してください。

ファイル名　mx78k0.plk
内容　　リンカ・パラメータ・ファイル
使用方法　　“mxk0oms.bat”,”mxk0omd.bat”を使用してオブジェクトモジュール作成する際にリンクするファイルが書かれています。本ファイルにユーザ・ファイルを追加して使用してください。

2.1.2 ソース・ファイル形式のディレクトリ構成




slim
chgpri.asm
hgtskt.asm
mxflag.asm
rotrdq.asm
rotrdq1.asm
rotrdq2.asm
rotrdq3.asm
statsk.asm
statsk1.asm
statsk2.asm
statsk3.asm
statskp.asm
statskt.asm
tertsk.asm
tertskf.asm
tertskt.asm
chgpri.idm
chgtskt.idm
mxflag.idm
rotrdq.idm
rotrdq1.idm
rotrdq2.idm
rotrdq3.idm
statsk.idm
statsk1.idm
statsk2.idm
statsk3.idm
statskp.idm
statskt.idm
tertsk.idm
tertskf.idm
tertskt.idm
mx78k0s.idl
debug
chgpri.asm
chgtskt.asm
mxflag.asm
rotrdq.asm
rotrdq1.asm
rotrdq2.asm
rotrdq3.asm
statsk.asm
statsk1.asm
statsk2.asm
statsk3.asm
statskp.asm
statskt.asm
tertsk.asm
tertskf.asm
tertskt.asm
chgpri.idm
chgtskt.idm
mxflag.idm

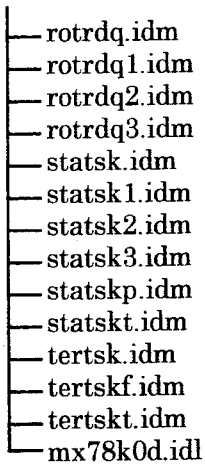


図 2.2　ソース・ファイル形式のディレクトリ構成

1．¥nectools¥bin ディレクトリ

ライブラリ構築用バッチ・ファイルが入っています。

ファイル名　mxk0mk.bat
内容　　システムコール・ライブラリ構築用のバッチ・ファイル
使用方法　　ディバグ・タイプとスリム・タイプの両方のライブラリ・ファイルを構築します。本バッチ・ファイルは"¥nectools¥src¥mx78k0"のディレクトリで実行し、引数としてチップ種別を指定してください。
例）mxk0mk△014　(ターゲット CPU がμPD78014)

ファイル名　mxk0as.bat，mxk0ad.bat
内容　　"mxk0mk.bat"で使用されるバッチ・ファイル（通常ユーザは使用しません）

2．¥nectools¥inc78k0 ディレクトリ

アセンブリ言語用およびC言語用のインクルード・ファイルが入っています。

ファイル名　mx78k0s.h，mx78k0cs.idm（ｉｄｅａ－Ｌ用）
内容　　スリム・タイプ用インクルード・ファイル（C言語）
使用方法　　システムコールの型宣言が書かれていますので、C言語でソースを記述する際に本ファイルをインクルードして使用してください。本ファイルを使用する際、ユーザ・システムで使用しないシステムコールの型宣言はコメント文に変更してください。

ファイル名　mx78k0d.h，mx78k0cd,idm（ｉｄｅａ－Ｌ用）
内容　　ディバグ・タイプ用インクルード・ファイル（C言語）
使用方法　　リターン・コードを定義していたり、システムコールの型宣言が書かれていますので、C言語でソースを記述する際に本ファイルをインクルードして使用してください。本ファイルを使用する際、ユーザ・システムで使用しないシステムコールの型宣言はコメント文に変更してください。

ファイル名　mx78k0s.inc，mx78k0as.idm（ｉｄｅａ－Ｌ用）
内容　　スリム・タイプ用インクルード・ファイル（アセンブリ言語）
使用方法　　システムコールのマクロが書かれていますので、アセンブリ言語ソースを記述する際に、本ファイルをインクルードして使用してください。本ファイルを使用する際、ユーザ・システムで使用しないシステムコールのマクロはコメント文に変更してください。

ファイル名　mx78k0d.inc，mx78k0ad.idm（ｉｄｅａ－Ｌ用）
内容　　ディバグ・タイプ用インクルード・ファイル（アセンブリ言語）
使用方法　　リターン・コードを定義していたり、システムコールのマクロが書かれていますので、アセンブリ言語ソースを記述する際に、本ファイルをインクルードして使用してください。本ファイルを使用する際、ユーザ・システムで使用しないシステムコールのマクロはコメント文に変更してください。

ファイル名　mxk0inc.idl
内容　　同一ディレクトリ内のｉｄｅａ－Ｌ用ソース・ファイル（拡張子.idm）をｉｄｅａ－Ｌが管理するための情報ファイル
使用方法　　本ファイルの使用方法は、「ｉｄｅａ－Ｌ　スクリーン・エディタ　入門編」を参照してください。

3．¥nectools¥lib78k0 ディレクトリ

ユーザによって再構築された、システムコール・ライブラリ・ファイルが格納されるディレクトリです。本ディレクトリにはファイルは入っていません。

4．¥nectools¥smp78k0¥mx78k0 ディレクトリ

各種サンプル・ファイルが入っています。

ファイル名　mx78k0r.asm
内容　　メモリ設定ファイルのサンプル・ファイル
使用方法　　OS が使用するメモリを設定するファイルです。本ファイルはユーザ・システムに合わせて変更してください。

ファイル名　mx78k0.plk
内容　　リンカ・パラメータ・ファイル
使用方法　　オブジェクト・ファイル形式の提供媒体に含まれているロード・モジュール作成用バッチ・ファイル（“mxk0oms.bat”,”mxk0omd.bat”）を使用してオブジェクトモジュールを作成する際にリンクするファイルが書かれています。本ファイルにユーザ・ファイルを追加して使用してください。

5．¥nectools¥src¥mx78k0 ディレクトリ

システムコールのソース・ファイルがタイプ別に入っています。

¥nectools¥src¥mx78k0¥common ディレクトリ

本ディレクトリには、スリム・タイプとディバグ・タイプで共通に使用するソース・ファイルが入っています。

ファイル名　exttsk.asm，exttsk.idm（ｉｄｅａ－Ｌ用）
内容　　“ext_tsk”システムコールおよび本 OS のタスク切り替えルーチンのソース・ファイル
ファイル名　mxint.asm，mxint.idm（ｉｄｅａ－Ｌ用）
内容　　タイマ処理ルーチンのソース・ファイル

ファイル名　rdqclr.asm，rdqclr.idm（ｉｄｅａ－Ｌ用）
内容　　レディ・キュー初期化ルーチンのソース・ファイル

ファイル名　trqclr.asm，trqclr.idm（ｉｄｅａ－Ｌ用）
内容　　レディ・キュー、タイマ・キューの双方を初期化するルーチンのソース・ファイル

ファイル名　（上記以外の）*.asm，*.idm（ｉｄｅａ－Ｌ用）
内容　　各種システムコールのソース・ファイル

ファイル名　mx78k0c.idl
内容　　common ディレクトリ内のｉｄｅａ－Ｌ用ソース・ファイル（拡張子．ｉｄｍ）をｉｄｅａ－Ｌが管理するための情報ファイル
使用方法　　本ファイルの使用方法は、「ｉｄｅａ－Ｌ　スクリーン・エディタ　入門編」を参照してください。

6．¥nectools¥src¥mx78k0¥slim ディレクトリ

スリム・タイプ用のソース・ファイルが入っています。

ファイル名　*.asm，*.idm（ｉｄｅａ－Ｌ用）
内容　　各種システムコールのソース・ファイル

ファイル名　mx78k0s.idl
内容　　slim ディレクトリ内のｉｄｅａ－Ｌ用ソース・ファイル（拡張子.idm）をｉｄｅａ－Ｌが管理するための情報ファイル
使用方法　　本ファイルの使用方法は、「ｉｄｅａ－Ｌ　スクリーン・エディタ　入門編」を参照してください。

7．¥nectools¥src¥mx78k0¥debug ディレクトリ

ディバグ・タイプ用のソース・ファイルが入っています。

ファイル名　*.asm，*.idm（ｉｄｅａ－Ｌ用）
内容　　各種システムコールのソース・ファイル

ファイル名　mx78k0d.idl
内容　　debug ディレクトリ内のｉｄｅａ－Ｌ用ソース・ファイル（拡張子.idm）をｉｄｅａ－Ｌが管理するための情報ファイル
使用方法　　本ファイルの使用方法は、「ｉｄｅａ－Ｌ　スクリーン・エディタ　入門編」を参照してください。

2.2　インストール手順

本 OS の提供媒体をホストマシンにインストールする方法は、ユーザの開発環境によって異なりますので、ここでは、例を挙げて説明します。
なお、入力例中の“A>”および“B>”はプロンプトを、“↓”はリターン・キーの入力を表しています。

1．提供媒体のセット

提供媒体（フロッピィ・ディスク）をフロッピィ・ディスク・ドライブにセットします。

2．カレント・ディスク・ドライブの変更

カレントのディスク・ドライブを提供媒体をセットしたドライブに移動します。

例)　ここではＢドライブに移動します。

```
A>B：↓
```

3．ファイル群の転送

xcopy コマンドを実行し、提供媒体に格納されているファイル群をホストマシンへ転送します。

例)　ここでは、A：¥にＢドライブの提供ファイル群を転送します。

```
B>xcopy　/e　/v　b:¥　a:¥↓
```

4．ファイル群の確認

dir コマンドを実行し、ファイル群がホストマシン上へ転送されたことを確認します。
なお、各ディレクトリの詳細は、「2.1　提供ファイル」を参照してください。

例)

```
B>dir　a:¥↓
```

5．コマンド検索パスの設定

メイク用バッチ・ファイルのあるディレクトリ(¥nectools¥bin)をコマンド検索パスに追加します。

例)ここでは環境設定ファイル（autoexec.bat など）内のｐａｔｈ変数の設定方法を示します。

```
path=a:¥nectools¥bin↓
```

2.3　インストール時の注意事項

1．ソース・ファイル形式のみをインストールする場合

ソース・ファイル形式のみをインストールした場合、ドキュメント・ファイル(”mx78k0.txt”), オブジェクト・ファイル形式付属のバッチ・ファイル(”mxk0asm.bat”, ”mxk0cc.bat”, ”mxk0oms.bat”, “mxk0omd.bat”)および、ｉｄｅａ－Ｌ用関数ライブラリ・ファイル(”mx78k0i.lib”)はインストールされませんので、これらのファイルを使用する場合はオブジェクト・ファイル形式の提供媒体から必要なファイルをホストマシンへ転送してください。

2．インストール先をルート・ディレクトリ以外に指定した場合

インストール先をルート・ディレクトリ以外に指定した場合、付属のメイク用バッチ・ファイルが正常に動作しないことがあります。このような場合には、各バッチ・ファイル内のパス指定記述部分を変更して使用してください。

3．MS-DOS のコマンドについて

インストールの際に使用する MS-DOS のコマンドやコマンド検索パスの設定についての詳細は、MS-DOS のユーザーズ・マニュアルを参照してください。

第3章　タスク管理

タスクとは、OS上でのアプリケーション・プログラムを構成する要素です。OSは、このタスクをプログラムの単位として処理を行います。
この章では、本OSにおけるタスクの管理方法、状態について述べます。

3.1　タスクの概要

タスクは、OS の管理下で実行されるプログラムの最小単位です。起動、実行、終了は、すべてタスク単位で行われています。

本OSでは、タスクにプライオリティはなく、レディ・キューに登録されている順番に処理されます。レディ・キュー登録することのできるタスクの最大数は63個です。

タスクが切り替わるのは、現在実行されているタスクの処理が終了したとき（ext_tsk()システムコールを発行したとき）のみです。

3.2　タスクの起動

システムが立ち上がると、最初に実行状態となるタスクは、システム初期化処理中において最初にsta_tsk()システムコールをかけたタスクです。また、システム初期化処理中に別のタスクに対してsta_tsk()システムコールをかけた場合は、一番最初にsta_tsk()システムコールをかけたタスクの後ろのレディ・キュー領域に登録されます。この時、sta_tskp()システムコールでタスクを起動した場合は、対象タスクをレディ・キューの先頭に登録しますので、一番最初にsta_tsk()システムコールをかけたタスクよりも先にRUN状態になります。

システム初期化処理中にsta_tsk()システムコールをかけなかったタスクに関しては、起動されたタスク中で、sta_tsk()システムコール等を発行することによって起動され、レディ・キューに登録されます。（システム初期化処理に関しては、「第 10 章　システム初期化処理」を参照してください。）

タスクの起動は、sta_tsk()システムコールの他に、現在動作しているタスクの次に対象タスクが動作するようレディ・キューに登録するsta_tskp()システムコール、指定時間経過後に指定タスクの起動を行うsta_tskt()システムコール、指定イベントフラグのビット・パターンによって指定タスクの起動を行うsta_tskf()システムコールがあります。

以下それらのシステムコールについて説明します。

3.2.1 sta_tsk( )の仕組み

sta_tsk()システムコールは、指定したタスクをレディ・キューの最後尾に登録する処理を行います。ここで指定したタスクは、そのタスクの実行順序が来るまで実行は待たされます。

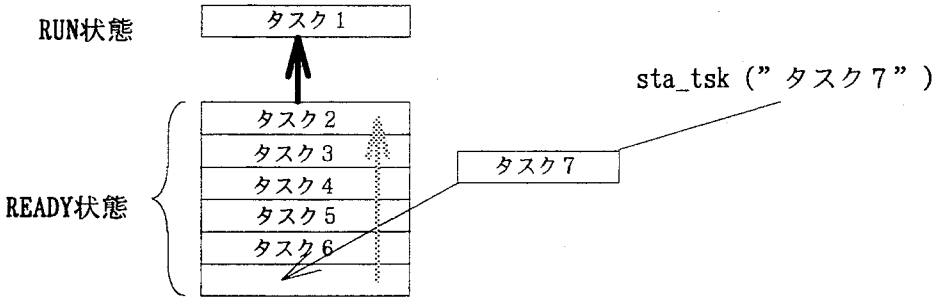


図 3.1：sta_tsk( )の説明図

3.2.2 sta_tskp( )の仕組み

sta_tskp( )システムコールは、指定したタスクを現在実行しているタスクの次に実行するようレディ・キューに登録します。ここで指定したタスクは、現在実行しているタスクの処理が終了すると動作します。

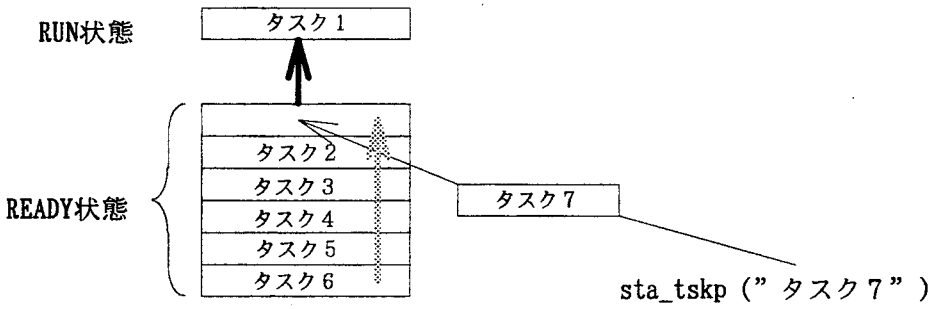


図 3.2：sta_tskp( )の説明図

3.2.3 sta_tskt( )の仕組み

sta_tskt()システムコールは、指定したタスクを指定時間経過後に起動させる処理を行います。指定したタスクは、いったんタイマ・キューに登録され指定時間が経過すると、レディ・キューの先頭に登録され、タイマ･キューから削除されます。レディ・キューに登録されたタスクは、その時点で動作しているタスクの処理が終了すると動作します。タイマ・キューに登録登録することのできるタスクの最大数は６２個です。

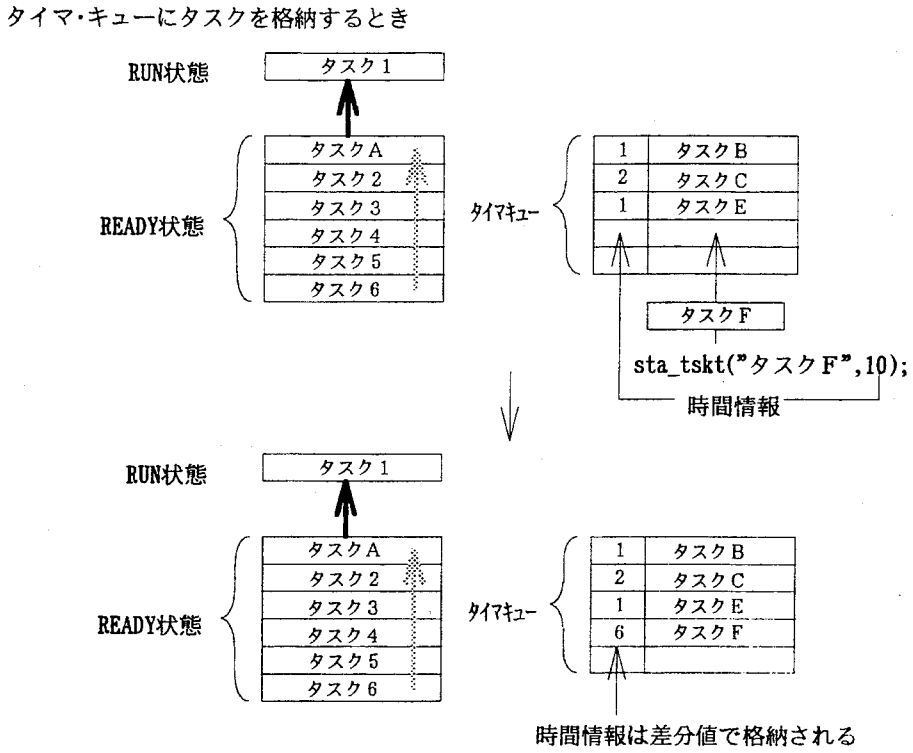


図 3.3：sta_tskt( )の説明図（１／２）

指定時間が経過したとき

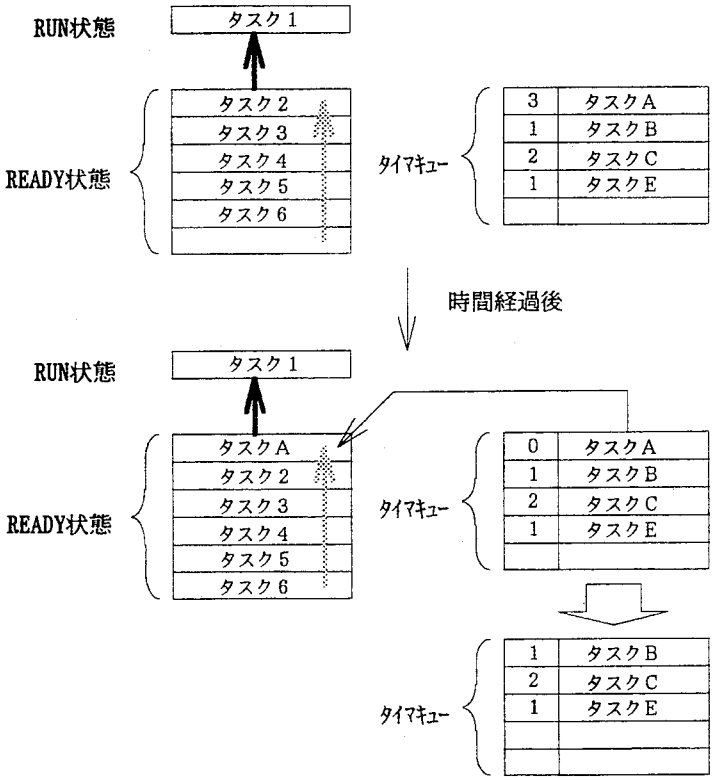
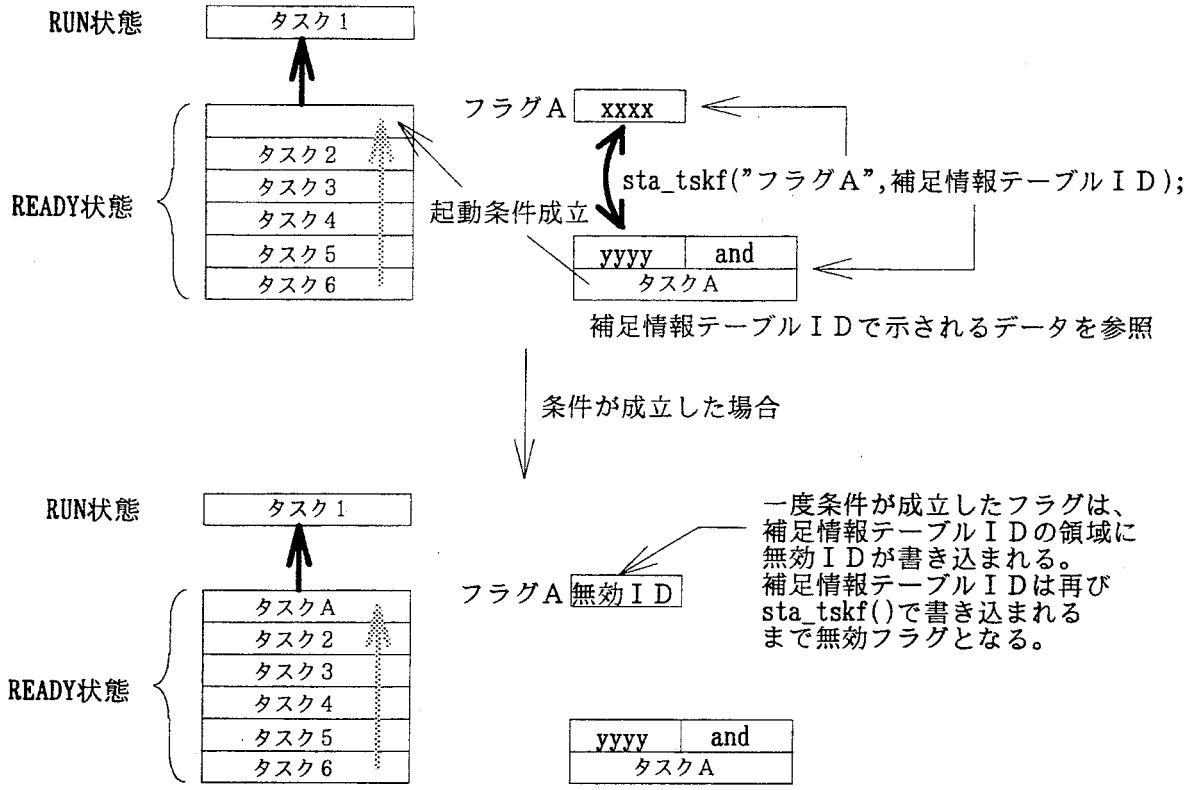


図 3.4：sta_tskt( )の説明図（２／２）

3.2.4 sta_tskf( )の仕組み

sta_tskf( )システムコールは、指定イベントフラグに起動タスクの登録を行います。イベントフラグに登録したタスクは、set_flg( )，rpl_flg( )システムコールによってセットされたビット・パターンと待ちビット・パターンが一致したときにレディ・キューの先頭に登録されます。



補足情報テーブルとイベントフラグは使用された後も、領域は開放されない。

図 3.5 : sta_tskf( )の説明図

3.2.5 レディ・キューにおけるタスクの多重登録について

レディ・キューには、同一タスクを複数個登録することができます。ただし、OS は同一タスクがレディ・キューに何個登録されてるか管理していませんので注意してください。
同一タスクをレディ・キューに複数個登録したくない場合は、下記を参照してください。（ter_tsk(), tsk_sts()システムコールに関しては、「第７章　システムコール」を参照してください）。

例）指定タスクが多重登録になる可能性がある場合

```
void task1( )
{
    unsigned char status;
        :
    task1 の処理;
        :
    tsk_sts(&status,task2);          /* task2 の状態を調べる */
    while(status == TTS_RDY)
    {
        ter_tsk(task2);              /* ter_tsk を task2 に対して発行 */
        tsk_sts(&status,task2);      /* task2 の状態を調べる */
    }
    sta_tsk(task2);                  /* task2 をレディ・キューに登録 */
        :
    task1 の処理;
        :
    ext_tsk( );
}
```

指定タスクを、レディ・キューに登録する直前に、指定タスクがレディ・キューに登録されているかどうか tsk_sts()システムコールを使用してチェックします。
指定タスクが、レディ・キューに登録されていれば ter_tsk()システムコールにて強制終了させます。
これにより、指定タスクは完全にレディ・キューから削除された状態で、登録することができます。

3.3 タスクの状態

本 OS におけるタスクは、動作する過程においてその状態を変えています。この節では、タスクが取り得る状態、およびその管理方法について述べます。
本 OS においてタスクが取り得る状態は、以下の 3 種類です。

◇ RUN 状態（実行状態）
OS から CPU 利用権を割り当てられ、プログラムを実行している状態です。システム全体を通じてこの RUN 状態になるタスクは 1 つだけです。

◇ READY 状態（実行可能状態）
レディ・キューに登録されている状態で、実行の順番を待っている状態です。RUN 状態のタスクが終了すると、レディ・キューの先頭に登録されているタスクが RUN 状態に遷移します。

◇ DORMANT 状態（休止状態）
RUN 状態でも READY 状態でもないタスクです。
タイマ・キューに登録されているタスク、およびイベントフラグによる起動を指定されているタスクの状態は、DORMANT 状態となります。

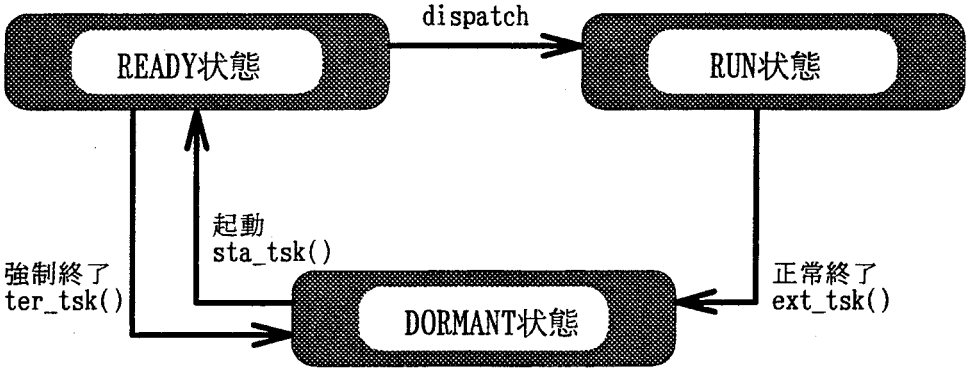


図 3.6：タスク状態遷移図

・RUN 状態のタスクの管理
本 OS において、RUN 状態になるのは READY 状態のタスクの内、レディ・キューの先頭に登録されているタスクです。RUN 状態になったタスクは、レディ・キューから削除されます。

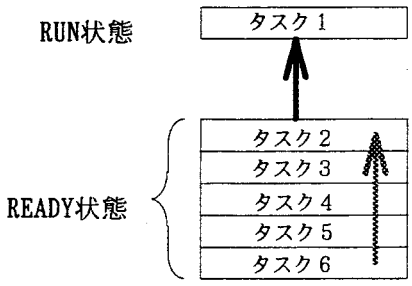


図 3.7：READY 状態のタスクの図

・READY 状態のタスクの管理
　本 OS では、sta_tsk()システムコールなどのタスク起動システムコールにより、READY 状態に遷移したタスクは、レディ・キューに登録され管理されます。レディ・キューの先頭に登録されているタスクが、現在実行しているタスクの次に実行するタスクとなります。

・DORMANT 状態のタスクの管理
　DORMANT 状態となっているタスクは、本 OS によって管理されません。ただし、タイマ・キューに登録されているタスク、およびイベントフラグによる起動を指定されているタスクは、OS によって管理されます。

3.4　タスクのディスパッチ

ディスパッチとは、レディ・キューの先頭に登録されているタスクを、RUN 状態に遷移させることをいいます。その処理機構をディスパッチャーといい、ディスパッチは、実行タスクの処理が終了した時に起こります。

3.5　タスクの終了

実行タスクが DORMANT 状態へ遷移することをタスクの終了といいます。タスクを終了させるシステムコールには以下のものがあります。

・自タスクを終了させる機能を持つシステムコール

ext_tsk( ), sta_tske( ), sta_tsk1e( ), sta_tsk2e( ), sta_tsk3e( ), sta_tskpe( ),
sta_tskte( ), chg_tskte( ), rot_rdqe( ), rot_rdq1e( ), rot_rdq2e( ), rot_rdq3e( )

・指定タスクを終了させる機能を持つシステムコール

ter_tsk( ), ter_tskt( ), ter_tskf( )

各システムコールの機能の説明は「第 7 章　システムコール」を参照してください。

第4章　イベントフラグ

本 OS におけるイベントフラグは、システム内外で起こった事象に対応するタスクを起動させるために使用します。

イベントフラグは、システム中に複数個設定することができますが、1イベントフラグに対して起動指定することのできるタスクは1つです。イベントフラグを複数個設定する場合は、RAM 上に連続して配置してください。この連続して配置したイベントフラグ群を、イベントフラグ本体テーブルといいます。C言語では構造体を使用して定義し、アセンブリ言語ではイベントフラグが連続するように確保してください。

また、イベントフラグには、起動するタスク、待ちビット・パターン、待ち条件を定義したイベントフラグ補足情報が必要になります。これらもイベントフラグ本体同様、複数個設定することができますが、その際は、ROM／RAM 上のどちらかに連続して配置してください。この連続して配置したイベントフラグ補足情報群を、イベントフラグ補足情報テーブルをいいます。イベントフラグ補足情報テーブルに登録した順番に「1,2,3…」と ID 番号が割り付けられ、その最大数は 0xfe です。

4.1　イベントフラグのセット

イベントフラグのセットは、set_flg()システムコールによって行います。

set_flg()システムコールを発行する際にパラメータとして、セットするビット・パターンを指定しますが、新たなイベントフラグのビット・パターンは、システムコールが発行される以前のビット・パターンとシステムコールで指定されたビット・パターンの論理和（OR）をとった値がセットされます。

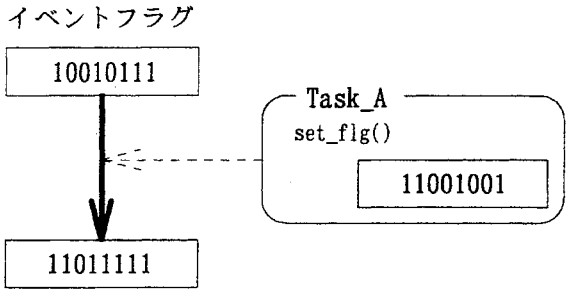


図 4.1：イベントフラグのセット

4.2 事象の発生に対応するタスクの登録

イベントフラグでは、事象の発生に対応するタスクの登録のために、sta_tskf( )システムコールによってイベントフラグ補足情報の ID 番号を指定しなければなりません。イベントフラグ補足情報テーブルの作成はシステムコールによっては行われませんので、ユーザが直接、補足情報テーブルに情報を設定しなければなりません。詳細は、「10.3.6 イベントフラグの確保」を参照してください。

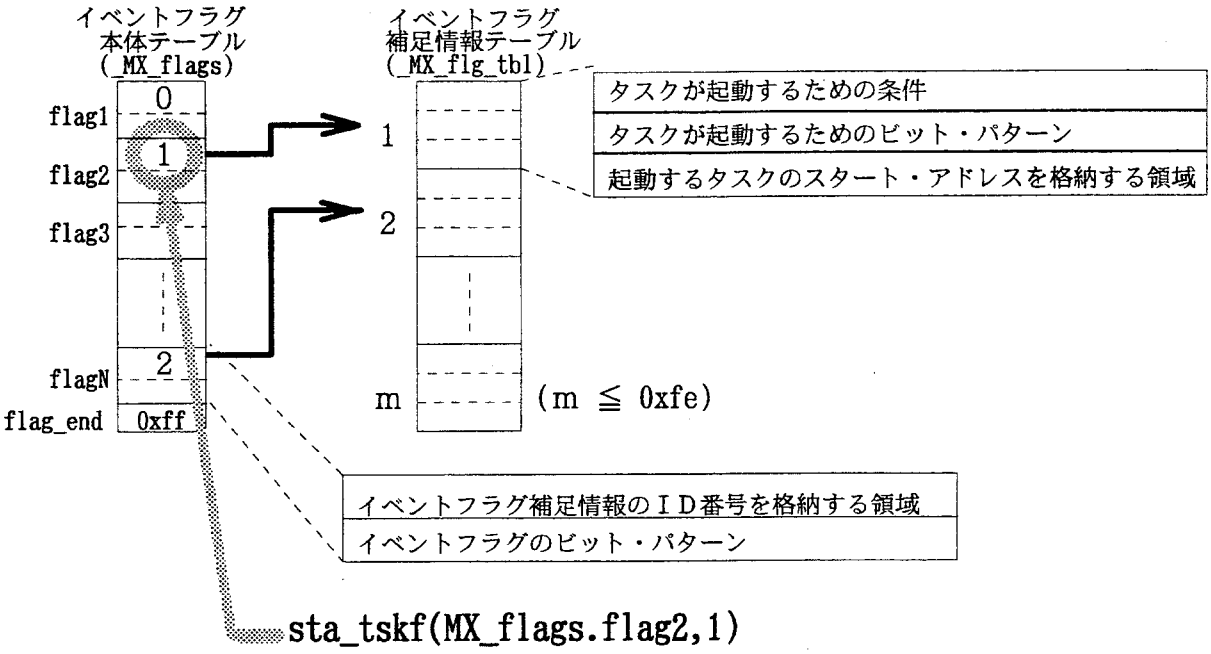


図 4.2 : タスクの登録

タスクが起動するための条件には以下のものがあります。

・ＯＲ条件　タスクが起動するためのビット・パターンの“１”である箇所のうち、イベントフラグの該当するビットが１つでも“１”になった場合に指定タスクを起動します。
・ＡＮＤ条件　タスクが起動するためのビット・パターンと、イベントフラグのビット・パターンが完全に一致した場合に指定タスクを起動します。

4.3 事象の発生に対応するタスクの起動

イベントフラグに登録されたタスクは、set_flg()、または rpl_flg()システムコールによってイベントフラグのビット・パターンがセットされ、タスクの起動条件が成立した場合は、DORMANT 状態から READY 状態へ移ります。

このとき、登録されていたタスクは、レディ・キューの先頭に登録されます。イベントフラグ補足情報の ID 番号を格納する領域には無効 ID が書き込まれ、イベントフラグのビット・パターンは０クリアされます。よって、イベントフラグのビット・パターンが再びセットされても、新たに sta_tskf( )システムコールによって、そのイベントフラグに対し補足情報を登録しない限り、そのイベントフラグによるタスクの起動はしません。

注意）　イベントフラグにタスクを登録した後に、同じイベントフラグに対して別タスクを登録することも可能ですが、その際は、後から登録されたタスクが有効になります。

第５章　時間管理

この章では、本 OS で扱われるタイマについて述べます。
OS におけるタスクの遅延起動は、この時間を基準にして行われています。この時間は、タイマ割り込みを使用します。

5.1　概要

この節では、OS の時間管理とその基準となるタイマ割り込みとの関係について述べます。
OS では、タスクの遅延起動（指定時間経過後にタスクをレディ・キューに登録する）機能があります。この機能を実現するために、８／１６ビットのインターバル・タイマを使用します。
なお、インターバル・タイマの初期化（インターバル値の設定、インターバル・タイマの選択）に関しては、初期化処理ルーチン内でユーザが行わなければなりません。
インターバル・タイマの初期化に関する詳細な説明は、各ＣＰＵのユーザーズ・マニュアルを参照してください。

5.2　遅延起動

本 OS では、指定時間経過後に指定タスクを起動する sta_tskt( )システムコールがあります。sta_tskt( )システムコールを発行すると対象タスクは、いったんタイマ・キューに登録され、指定時間経過、レディ・キューの先頭に登録されます（sta_tskt()に関する詳細は、「7.3.12 sta_tskt」を参照してください）。

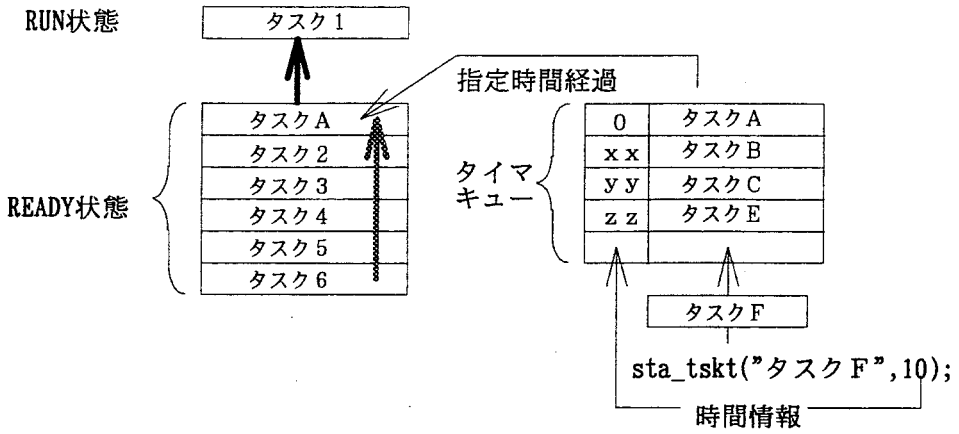


図 5.1：sta_tskt()の説明図

5.3 起動時間の変更

本 OS には、起動時間の変更を行う chg_tskt( )システムコールがあります。
まず、 chg_tskt()システムコールは、指定したタスクがタイマ・キュー、またはレディ・キューに存在するかどうか検索します。
もし、指定タスクが存在した場合は、そのタスクをキューから削除し、起動時間を指定した時間に設定し直してタイマ・キューに再登録します。

5.4 タイマ・キューにおけるタスクの多重登録について

タイマ・キューもレディ・キューと同様に、同一タスクを複数個登録することができます。ただし、OS は同一タスクがタイマ・キューに何個登録されてるか管理していませんので注意してください。
同一タスクをタイマ・キューに複数個登録したくない場合は、sta_tskt( )システムコールで対象タスクをタイマ・キューに登録するのではなく、chg_tskt( )システムコールにて登録してください。（chg_tskt( )システムコールに関しては、「第 7 章　システムコール」を参照してください）。

第６章　割り込み管理

割り込み管理は、割り込みを事象として駆動する処理を管理することです。割り込みにより起動する処理ルーチンを割り込みハンドラと呼び、タスクとは独立した扱いとなります。

OS は、各割り込み要求のイニシャライズを行わないので、各割り込み要求はユーザにより初期化する必要があります。また、割り込みハンドラからの復帰処理も、ユーザによって記述されなければなりません。

6.1　割り込みハンドラの位置付け

割り込みハンドラは、割り込みが発生した際に起動される処理プログラムです。したがって割り込みの応答性を考え、その処理は即座に終了する必要があります。そのため、割り込みハンドラの実行は、タスクや OS よりも優先されます。

6.2　割り込みハンドラ内での処理

割り込みハンドラは、OS を経由せずに割り込み発生時に直接起動されます。したがって、その処理内容には下記のような制限、手続きが必要です。

１．レジスタの退避

レジスタの退避と復帰は、ユーザが割り込みハンドラ内で行わなければなりません。これは、割り込みハンドラが OS を経由せずに直接起動されるためです。また、割り込みハンドラは、OS が使用するレジスタバンクで処理しないでください。

２．システムコール発行の制限

割り込みハンドラは、高速に処理させなければならない処理です。したがってハンドラからのシステムコールも、高速に終了させなければなりません。

さらに、システムコール処理中に発生した割り込みに対する割り込みハンドラで、システムコールを発行することが困難な場合があります。そのため、割り込みハンドラから発行できるシステムコールの制限を設けています。

以下に割り込みハンドラから発行可能なシステムコールを示します。

sta_tskp( ), sta_tskf( ), set_flg( ), clr_flg( ), rpl_flg( )

上記以外のシステムコールを割り込みハンドラから発行した場合は、正常な動作が保証されません。なお、各システムコールの機能については「第７章　システムコール」を参照してください。

３．割り込みハンドラの終了

割り込みハンドラは、CPU 命令である「RETI」命令を発行することで終了します。なお、「RETI」命令に先立って必要に応じてレジスタの復帰を行ってください。

6.3　OS が使用する割り込み

本 OS は、基本的には割り込みを使用しません。ただし、時間管理機能を使用する場合はタイマ割り込みを使用します。時間管理用に使用するタイマ割り込みは、ユーザが任意に設定することができます。

多重割り込みによる時間管理機能の不正な動作を防ぐため、時間管理用に使用するタイマ割り込みは、OS の時間管理ルーチンを呼び出すための専用の割り込みとしてください。

保守／廃止

第７章　システムコール

システムコールとは、タスクが各種操作を行うために用意したサービス、または呼び出し手続きのことをいいます。システムコールにより、タスクの起動・終了、フラグのセット／リセットなどの操作を行うことが可能になります。

この節では、システムコールの機能概要とエントリの方法について述べています。

7.1　機能概要

システムコールは、その機能によって以下の２つのグループに分けることができます。

・タスク関連システムコール

タスク関連システムコールは、タスクの起動・終了などの操作を行うシステムコールのグループです。このグループに属するシステムコールは以下の通りです。

sta_tsk	：指定タスクを起動します。
sta_tske	：指定タスクを起動し、自タスクを終了します。
sta_tskp	：指定タスクを起動します。指定されたタスクは、現在動作しているタスクの次に起動します。
sta_tskpe	：指定タスクを起動し、自タスクを終了します。指定されたタスクは、現在動作しているタスクの次に起動します。
ter_tsk	：レディ・キューにある指定タスクを強制終了させます。
sta_tskt	：指定時間経過後、指定タスクを起動します。
sta_tskte	：指定時間経過後、指定タスクが起動するように設定し、自タスクを終了します。
chg_tskt	：指定したタスクの起動時間を変更します。
chg_tskte	：指定したタスクの起動時間を変更し、自タスクを終了します。
ter_tskt	：タイマ・キューまたはレディ・キューにある指定タスクを強制終了させます。
sta_tskf	：イベントフラグ操作で起動する指定タスクをイベントフラグに登録します。
ter_tskf	：イベントフラク操作で起動されるタスク、またはレディ・キューにある指定タスクを強制終了させます。
rot_rdq	：自タスクをレディ・キューの最後尾に登録します。
rot_rdqe	：自タスクをレディ・キューの最後尾に登録し、自タスクを終了します。
tsk_sts	：タスクの状態を得ます。
chg_pri	：指定したタスクを現在起動しているタスクの次に起動するようにします。
ext_tsk	：自タスクを終了します。

また、sta_tsk( ), sta_tske( ), rot_rdq( ), rot_rdqe( )には、ter_tsk( )および tsk_sts( )での検索を高速化するために、以下のシステムコールが用意されています。

sta_tsk1　：sta_tsk( )の動作と同時に、ter_tsk( )での高速検索の準備を行います。
sta_tsk2　：sta_tsk( )の動作と同時に、tsk_sts( )での高速検索の準備を行います。
sta_tsk3　：sta_tsk( )の動作と同時に、ter_tsk( )および tsk_sts( )での高速検索の準備を行います。

sta_tsk1e　：sta_tske( )の動作と同時に、ter_tsk( )での高速検索の準備を行います。
sta_tsk2e　：sta_tske( )の動作と同時に、tsk_sts( )での高速検索の準備を行います。
sta_tsk3e　：sta_tske( )の動作と同時に、ter_tsk( )および tsk_sts( )での高速検索の準備を行います。

rot_rdq1　：rot_rdq( )の動作と同時に、ter_tsk( )での高速検索の準備を行います。
rot_rdq2　：rot_rdq( )の動作と同時に、tsk_sts( )での高速検索の準備を行います。
rot_rdq3　：rot_rdq( )の動作と同時に、ter_tsk( )および tsk_sts( )での高速検索の準備を行います。

rot_rdq1e　：rot_rdqe( )の動作と同時に、ter_tsk( )での高速検索の準備を行います。
rot_rdq2e　：rot_rdqe( )の動作と同時に、tsk_sts( )での高速検索の準備を行います。
rot_rdq3e　：rot_rdqe( )の動作と同時に、ter_tsk( )および tsk_sts( )での高速検索の準備を行います。

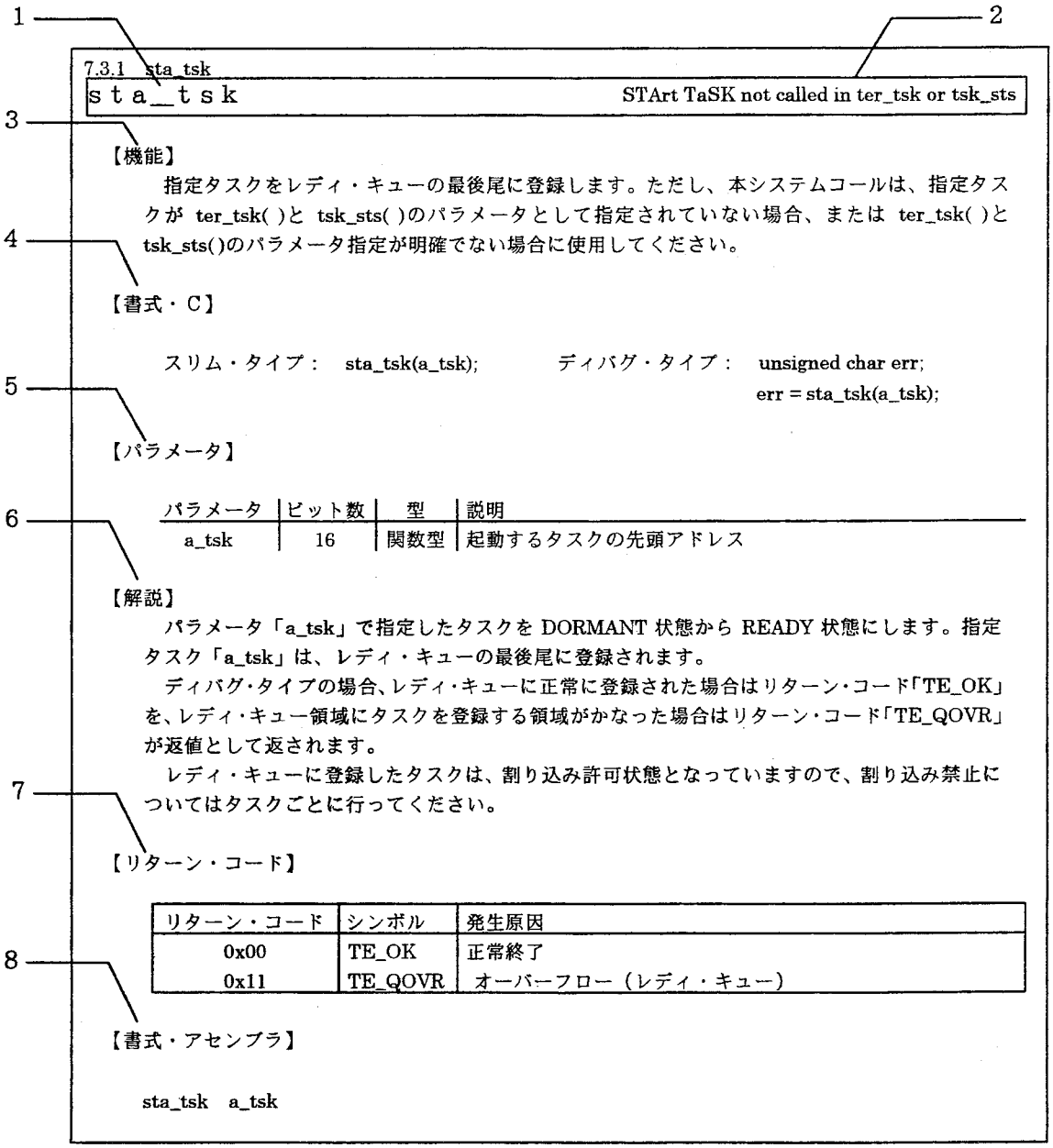
・イベントフラグ関連システムコール

イベントフラグ関連システムコールは、システム内外で起きた事象に対応するタスクを起動するためのフラグを操作するシステムコールのグループです。このグループに属するシステムコールは以下の通りです。

set_flg　：イベントフラグの任意のビットをセットします。
clr_flg　：イベントフラグの任意のビットをクリアします。
rpl_flg　：イベントフラグを任意のビット・パターンと入れ替えます。

7.2 システムコールの仕様

この節では、各システムコールのパラメータ、機能などの仕様の詳細について述べています。システムコール仕様の詳細は、下記に示すフォーマットで記述しています。



1. システムコールの名称。
2. システムコールの正式名称。
3. 機能。システムコールの機能概要を述べています。
4. 書式。C言語からシステムコールを呼び出す際の書式。スリム・タイプとディバグ・タイプで書式が異なる場合は、それぞれのタイプについて記述しています。
5. パラメータ。システムコールで必要なパラメータの種類、サイズ(ビット数)、パラメータの意味と有効な範囲を示しています。必要のない場合は、「なし」と記述します。
6. 解説。システムコールの処理内容と、ディバグ・タイプ使用時のリターン・コードの意味について記述しています。
7. リターン・コード。リターン・コード（16進数）、シンボル、意味を記述しています。なお、こ

のリターン・コードを返すのはディバグ・タイプのみです。また、リターン・コードは、C言語の場合、システムコールの返値として返され、アセンブリ言語の場合は、Cレジスタに返されます。ただし、イベントフラグ関連システムコールのリターン・コードは、C言語の場合、引数で指定した変数に返されます。

8. 書式。アセンブリ言語からシステムコールを呼び出す際の書式。アセンブリ言語のシステムコールは、マクロにて提供していますので、ここでは、マクロ名とパラメータを記述しています。

7.3 タスク関連システムコール

この節では、タスク関連のシステムコールについて説明します。タスク関連システムコールは、下記に示すものがあります。

sta_tsk( ),	sta_tsk1( ),	sta_tsk2( ),	sta_tsk3( ),
sta_tske( ),	sta_tsk1e( ),	sta_tsk2e( ),	sta_tsk3e( ),
sta_tskp( ),	sta_tskpe( ),	ter_tsk( ),	sta_tskt( ),
sta_tskte( ),	chg_tskt( ),	chg_tskte( ),	ter_tskt( ),
sta_tskf( ),	ter_tskf( ),		
rot_rdq( ),	rot_rdq1( ),	rot_rdq2( ),	rot_rdq3( ),
rot_rdqe( ),	rot_rdq1e( ),	rot_rdq2e( ),	rot_rdq3e( ),
tsk_sts( ),	chg_pri( ),	ext_tsk( )	

7.3.1 sta_tsk

s t a _ t s k	STArt TaSK not called in ter_tsk or tsk_sts

【機能】

指定タスクをレディ・キューの最後尾に登録します。ただし、本システムコールは、指定タスクが ter_tsk( )と tsk_sts( )のパラメータとして指定されていない場合、または ter_tsk( )と tsk_sts( )のパラメータ指定が明確でない場合に使用してください。

【書式・C】

スリム・タイプ： sta_tsk(a_tsk);

ディバグ・タイプ：
```
unsigned char err;
err = sta_tsk(a_tsk);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動するタスクの先頭アドレス

【解説】

指定タスク「a_tsk」を、レディ・キューの最後尾に登録し、DORMANT 状態から READY 状態にします。

ディバグ・タイプの場合、レディ・キューに正常に登録できた場合はリターン・コード「TE_OK」を、レディ・キューに指定タスクを登録する領域がなかった場合はリターン・コード「TE_QOVR」が返値として返されます。

レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x11	TE_QOVR	オーバフロー（レディ・キュー）

【書式・アセンブラ】

```
sta_tsk  a_tsk
```

7.3.2 sta_tsk1

s t a _ t s k 1	STArt TaSK called in ter_tsk

【機能】

指定タスクをレディ・キューの最後尾に登録します。ただし、本システムコールは、指定タスクが ter_tsk( )のパラメータとして指定され、tsk_sts( )のパラメータとしては指定されない場合に使用してください。

【書式・C】

スリム・タイプ： sta_tsk1(a_tsk);

ディバグ・タイプ：
```
unsigned char err;
err = sta_tsk1(a_tsk);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動するタスクの先頭アドレス

【解説】

指定タスク「a_tsk」を、レディ・キューの最後尾に登録し、DORMANT状態からREADY状態にします。また、ter_tsk( )システムコール発行時の高速検索のための準備処理も行います。

ディバグ・タイプの場合、レディ・キューに正常に登録できた場合はリターン・コード「TE_OK」を、レディ・キューに指定タスクを登録する領域がなかった場合はリターン・コード「TE_QOVR」が返値として返されます。

レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x11	TE_QOVR	オーバフロー（レディ・キュー）

【書式・アセンブラ】

```
sta_tsk1  a_tsk
```

7.3.3 sta_tsk2

s t a _ t s k 2	STArt TaSK called in tsk_sts

【機能】

指定タスクをレディ・キューの最後尾に登録します。ただし、本システムコールは、指定タスクが ter_tsk( )のパラメータとしては指定されず、tsk_sts( )のパラメータとして指定される場合に使用してください。

【書式・C】

スリム・タイプ： sta_tsk2(a_tsk);

ディバグ・タイプ：
```
unsigned char err;
err = sta_tsk2(a_tsk);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動するタスクの先頭アドレス

【解説】

指定タスク「a_tsk」を、レディ・キューの最後尾に登録し、DORMANT 状態から READY 状態にします。また、tsk_sts( )システムコール発行時の高速検索のための準備処理も行います。

ディバグ・タイプの場合、レディ・キューに正常に登録できた場合はリターン・コード「TE_OK」を、レディ・キューに指定タスクを登録する領域がなかった場合はリターン・コード「TE_QOVR」が返値として返されます。

レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x11	TE_QOVR	オーバフロー（レディ・キュー）

【書式・アセンブラ】

```
sta_tsk2  a_tsk
```

保守／廃止

7.3.4 sta_tsk3

ｓｔａ＿ｔｓｋ３	STArt TaSK called in tsk_sts and tsk_sts

【機能】

指定タスクをレディ・キューの最後尾に登録します。ただし、本システムコールは、指定タスクが ter_tsk( )と tsk_sts( )のパラメータとして指定される場合に使用してください。

【書式・C】

スリム・タイプ： sta_tsk3(a_tsk);

ディバグ・タイプ：
```
unsigned char err;
err = sta_tsk3(a_tsk);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動するタスクの先頭アドレス

【解説】

指定タスク「a_tsk」を、レディ・キューの最後尾に登録し、DORMANT 状態から READY 状態にします。また、ter_tsk( )と tsk_sts( )システムコール発行時の高速検索のための準備処理も行います。

ディバグ・タイプの場合、レディ・キューに正常に登録できた場合はリターン・コード「TE_OK」を、レディ・キューに指定タスクを登録する領域がなかった場合はリターン・コード「TE_QOVR」が返値として返されます。

レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x11	TE_QOVR	オーバフロー（レディ・キュー）

【書式・アセンブラ】

```
sta_tsk3    a_tsk
```

保守／廃止

7.3.5 sta_tske

s t a _ _ t s k e	STArt TaSK not called in ter_tsk or tsk_sts, and Exit task

【機能】

指定タスクをレディ・キューの最後尾に登録し、自タスクを正常終了します。ただし、本システムコールは、指定タスクが ter_tsk()と tsk_sts()のパラメータとして指定されない場合、または ter_tsk()と tsk_sts()のパラメータ指定が明確でない場合に使用してください。

【書式・C】

```
sta_tske(a_tsk);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動するタスクの先頭アドレス

【解説】

指定タスク「a_tsk」を、レディ・キューの最後尾に登録し、DORMANT 状態から READY 状態にします。また同時に、自タスクを DORMANT 状態にします。

ディバグ・タイプの場合でも、本システムコールを発行すると処理が終了するためリターン・コードは返されません。

レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

なし

【書式・アセンブラ】

```
sta_tske  a_tsk
```

保守／廃止

7.3.6 sta_tsk1e

s t a _ _ t s k 1 e	STArt TaSK called in ter_tsk, and Exit task

【機能】

指定タスクをレディ・キューの最後尾に登録し、自タスクを正常終了します。ただし、本システムコールは、指定タスクが ter_tsk( )のパラメータとして指定され、tsk_sts( )のパラメータとしては指定されない場合に使用してください。

【書式・C】

```
sta_tsk1e(a_tsk);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動するタスクの先頭アドレス

【解説】

指定タスク「a_tsk」を、レディ・キューの最後尾に登録し、DORMANT 状態から READY 状態にします。また同時に、自タスクを DORMANT 状態にし、ter_tsk( )システムコール発行時の高速検索のための準備処理も行います。

ディバグ・タイプの場合でも、本システムコールを発行すると処理が終了するためリターン・コードは返されません。

レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

なし

【書式・アセンブラ】

```
sta_tsk1e   a_tsk
```

保守／廃止

7.3.7 sta_tsk2e

s t a _ t s k 2 e	STArt TaSK called in tsk_sts, and Exit task

【機能】

指定タスクをレディ・キューの最後尾に登録し、自タスクを正常終了します。ただし、本システムコールは、指定タスクが ter_tsk( )のパラメータとしては指定されず、tsk_sts( )のパラメータとして指定される場合に使用してください。

【書式・C】

```
sta_tsk2e(a_tsk);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動するタスクの先頭アドレス

【解説】

指定タスク「a_tsk」を、レディ・キューの最後尾に登録し、DORMANT 状態から READY 状態にします。また同時に、自タスクを DORMANT 状態にし、tsk_sts( )システムコール発行時の高速検索のための準備処理も行います。

ディバグ・タイプの場合でも、本システムコールを発行すると処理が終了するためリターン・コードは返されません。

レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

なし

【書式・アセンブラ】

```
sta_tsk2e   a_tsk
```

保守／廃止

7.3.8 sta_tsk3e

ｓｔａ＿ｔｓｋ３ｅ	STArt TaSK called in ter_tsk and tsk_sts, and Exit task

【機能】

指定タスクをレディ・キューの最後尾に登録し、自タスクを正常終了します。ただし、本システムコールは、指定タスクが ter_tsk()と tsk_sts()のパラメータとして指定される場合に使用してください。

【書式・C】

```
sta_tsk3e(a_tsk);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動するタスクの先頭アドレス

【解説】

指定タスク「a_tsk」を、レディ・キューの最後尾に登録し、DORMANT 状態から READY 状態にします。また同時に、自タスクを DORMANT 状態にし、ter_tsk( )と tsk_sts( )システムコール発行時の高速検索のための準備処理も行います。

ディバグ・タイプの場合でも、本システムコールを発行すると処理が終了するためリターン・コードは返されません。

レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

なし

【書式・アセンブラ】

```
sta_tsk3e   a_tsk
```

7.3.9 sta_tskp

s t a _ t s k p	STArt TaSK and change Priority

【機能】

指定タスクをレディ・キューの先頭に登録します。

【書式・C】

スリム・タイプ： sta_tskp(a_tsk);　　ディバグ・タイプ： unsigned char err;
err = sta_tskp(a_tsk);

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動するタスクの先頭アドレス

【解説】

指定タスク「a_tsk」を、レディ・キューの先頭に登録し、DORMANT 状態から READY 状態にします。

ディバグ・タイプの場合、レディ・キューに正常に登録できた場合はリターン・コード「TE_OK」を、レディ・キューに指定タスクを登録する領域がなかった場合はリターン・コード「TE_QOVR」が返値として返されます。

レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x11	TE_QOVR	オーバフロー（レディ・キュー）

【書式・アセンブラ】

```
sta_tskp  a_tsk
```

7.3.10 sta_tskpe

s t a _ t s k p e	STArt TaSK and change Priority, and Exit task

【機能】

指定タスクをレディ・キューの先頭に登録し、自タスクを正常終了します。

【書式・C】

```
sta_tskpe(a_tsk);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動するタスクの先頭アドレス

【解説】

指定タスク「a_tsk」を、レディ・キューの先頭に登録し、DORMANT状態からREADY状態にします。また同時に、自タスクをDORMANT状態にします。

ディバグ・タイプの場合でも、本システムコールを発行すると処理が終了するためリターン・コードは返されません。

レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

なし

【書式・アセンブラ】

```
sta_tskpe  a_tsk
```

保守／廃止

7.3.11 ter_tsk

ter_tsk	TERminate TaSK in ready queue

【機能】

レディ・キューに登録されている指定タスクを強制終了させます。

【書式・C】

スリム・タイプ： ter_tsk(a_tsk);

ディバグ・タイプ：
```
unsigned char err;
err = ter_tsk(a_tsk);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	終了させるタスクの先頭アドレス

【解説】

指定タスク「a_tsk」を、レディ・キューから削除しDORMANT状態にします。

ディバグ・タイプの場合、レディ・キューから削除できた場合はリターン・コード「TE_OK」を、レディ・キューに指定タスクがなかった場合はリターン・コード「TE_DMT」が返値として返されます。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x0d	TE_DMT	指定タスクがDORMANT状態である

【書式・アセンブラ】

```
ter_tsk  a_tsk
```

7.3.12 sta_tskt

s t a _ t s k t	STArt TaSK after a time

【機能】

指定時間経過後、指定タスクをレディ・キューの先頭に登録します。

【書式・C】

スリム･タイプ: sta_tskt(a_tsk,tmout);　　ディバグ・タイプ: unsigned char err;
err = sta_tskt(a_tsk,tmout);

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動するタスクの先頭アドレス
tmout	16	unsigned int	タスクを起動するまでの時間

【解説】

指定タスク「a_tsk」を、指定時間「tmout」経過までタイマ・キューに登録し、指定時間経過後レディ・キューの先頭に登録します。

「tmout」で指定する時間は、タイマ割り込みのインターバル時間を設定してください。

ディバグ･タイプの場合、タイマ･キューに正常に登録できた場合はリターン･コード「TE_OK」を、タイマ･キューに指定タスクを登録する領域がなかった場合はリターン･コード「TE_QOVR」が返値として返されます。

指定時間経過後、レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x11	TE_QOVR	オーバーフロー（タイマ・キュー）

【書式・アセンブラ】

sta_tskt　a_tsk,tmout

保守／廃止

7.3.13 sta_tskte

s t a _ _ t s k t e	STArt TaSK after a time, and Exit task

【機能】

指定時間経過後、指定タスクをレディ・キューの先頭に登録し、自タスクを正常終了します。

【書式・C】

```
sta_tskte(a_tsk,tmout);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動するタスクの先頭アドレス
tmout	16	unsigned int	タスクを起動するまでの時間

【解説】

指定タスク「a_tsk」を、指定時間「tmout」経過までタイマ・キューに登録し、自タスクをDORMANT状態にします。指定時間経過後、指定タスクをレディ・キューの先頭に登録します。

「tmout」で指定する時間は、タイマ割り込みのインターバル時間を設定してください。

ディバグ・タイプの場合でも、本システムコールを発行すると処理が終了するためリターン・コードは返されません。

指定時間経過後、レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

なし

【書式・アセンブラ】

```
sta_tskte   a_tsk,tmout
```

7.3.14 chg_tskt

c h g _ t s k t	CHanG TaSK's startTime

【機能】

指定タスクの起動時間を変更します。

【書式・C】

スリム・タイプ： chg_tskt(a_tsk,tmout); ディバグ・タイプ： unsigned char err;
err = chg_tskt(a_tsk,tmout);

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動タスクの先頭アドレス
tmout	16	unsigned int	タスクを起動するまでの時間

【解説】

指定タスク「a_tsk」の起動時間を、指定時間「tmout」に変更します。本システムコールを発行すると、指定タスクをタイマ・キュー、レディ・キューの順番に探しだし、指定タスクが登録されていれば強制終了させ、タイマ・キューに再登録します。指定タスクが各キューに登録されていなくても、指定タスクをタイマ・キューに登録します。

「tmout」で指定する時間は、タイマ割り込みのインターバル時間を設定してください。

ディバグ・タイプの場合、タイマ・キューに正常に登録できた場合はリターン・コード「TE_OK」を、タイマ・キューに指定タスクを登録する領域がなかった場合はリターン・コード「TE_QOVR」が返値として返されます。

指定時間経過後、レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x11	TE_QOVR	オーバーフロー（タイマ・キュー）

【書式・アセンブラ】

```
chg_tskt  a_tsk,tmout
```

7.3.15 chg_tskte

chg_tskte	CHanG TaSK's startTime, and Exit task

【機能】

指定タスクの起動時間を変更し、自タスクを正常終了します。

【書式・C】

```
chg_tskte(a_tsk,tmout);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	起動するタスクの先頭アドレス
tmout	16	unsigned int	タスクを起動するまでの時間

【解説】

指定タスク「a_tsk」の起動時間を、指定時間「tmout」に変更し、自タスクを正常終了します。本システムコールを発行すると、指定タスクをタイマ・キュー、レディ・キューの順番に探しだし、指定タスクが登録されていれば強制終了させ、タイマ・キューに再登録します。指定タスクが各キューに登録されていなくても、指定タスクをタイマ・キューに登録します。

「tmout」で指定する時間は、タイマ割り込みのインターバル時間を設定してください。

ディバグ・タイプの場合でも、本システムコールを発行すると処理が終了するためリターン・コードは返されません。

指定時間経過後、レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

なし

【書式・アセンブラ】

```
chg_tskte   a_tsk,tmout
```

保守／廃止

7.3.16 ter_tskt

ter_tskt	TERminate TaSK in ready queue or Timer queue

【機能】

タイマ・キュー、またはレディ・キューに登録されている指定タスクを強制終了させます。

【書式・C】

スリム・タイプ： ter_tskt(a_tsk);

ディバグ・タイプ：
```
unsigned char err;
err = ter_tskt(a_tsk);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	終了させるタスクの先頭アドレス

【解説】

指定タスク「a_tsk」をタイマ・キュー、またはレディ・キューから削除しDORMANT状態にします。指定タスクが指定時間待ちのタスクでなかった場合でも、レディ・キューに登録されていれば削除します。

ディバグ・タイプの場合、タイマ・キュー、またはレディ・キューから削除できた場合はリターン・コード「TE_OK」を、タイマ・キュー、またはレディ・キューに指定タスクがなかった場合はリターン・コード「TE_DMT」が返値として返されます。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x0d	TE_DMT	指定タスクが登録されていない

【書式・アセンブラ】

```
ter_tskt  a_tsk
```

7.3.17 sta_tskf

s t a _ _ t s k f	STArt TaSK with event-Flag

【機能】

イベントフラグ操作で起動するタスクを登録します。

【書式・C】

```
sta_tskf(a_flg,pk_finfo);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_flg	16	関数型	対象イベントフラグのアドレス
pk_finfo	8	char	イベントフラグ補足情報の ID 番号

補足情報の各パラメータ	ビット数	型	説明
waiptn	8	char	タスクの待ちビット・パターン
wfmode	8	char	タスクの起動条件
a_tsk	16	関数型	起動するタスクの先頭アドレス

【解説】

指定イベントフラグ「a_flg」に、イベントフラグ補足情報「pk_finfo」の ID 番号で指定したタスクを登録します。パラメータ「pk_finfo」で指定する ID 番号は、イベントフラグ補足情報テーブル内の先頭補足情報から 1，2，3…となります。

イベントフラグ補足情報は、「_MX_flg_TBL」というテーブル名で ROM／RAM のどちらかに存在しなければなりません。イベントフラグ補足情報テーブルの生成については、「10.3.6 イベントフラグの確保」を参照してください。

タスクの起動条件は、AND 条件の場合は「EVENT_AND」を、OR 条件の場合は「EVENT_OR」を指定してください。

【リターン・コード】

なし

【書式・アセンブラ】

```
sta_tskf  a_flg,pk_finfo
```

7.3.18 rot_rdq

r o t _ r d q	ROTate ReaDy Queue(current task is not called in ter_tsk or tsk_sts)

【機能】

自タスクをレディ・キューの最後尾に登録します。ただし、本システムコールは、自タスクがter_tsk( )と tsk_sts( )のパラメータとして指定されない場合、または ter_tsk( )と tsk_sts( )のパラメータ指定が明確でない場合に使用してください。

【書式・C】

スリム・タイプ： rot_rdq( );

ディバグ・タイプ：
```
unsigned char err;
err = rot_rdq( );
```

【パラメータ】

なし

【解説】

自タスクを、レディ・キューの最後尾に登録します。

ディバグ・タイプの場合、レディ・キューの最後尾に正常に登録できた場合はリターン・コード「TE_OK」を、レディ・キューに指定タスクを登録する領域がなかった場合はリターン・コード「TE_QOVR」が返値として返されます。

レディ・キューに登録した自タスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x11	TE_QOVR	オーバーフロー（レディ・キュー）

【書式・アセンブラ】

rot_rdq

保守／廃止

7.3.19 rot_rdq1

r o t _ r d q 1	ROTate ReaDy Queue(current task is called in ter_tsk)

【機能】

自タスクをレディ・キューの最後尾に登録します。ただし、本システムコールは、自タスクがter_tsk()のパラメータとして指定され、tsk_sts()のパラメータとしては指定されない場合に使用してください。

【書式・C】

スリム・タイプ： rot_rdq1();

ディバグ・タイプ：
```
unsigned char err;
err = rot_rdq1();
```

【パラメータ】

なし

【解説】

自タスクを、レディ・キューの最後尾に登録します。また、ter_tsk()システムコール発行時の高速検索のための準備処理も行います。

ディバグ・タイプの場合、レディ・キューの最後尾に正常に登録できた場合はリターン・コード「TE_OK」を、レディ・キューに指定タスクを登録する領域がなかった場合はリターン・コード「TE_QOVR」が返値として返されます。

レディ・キューに登録した自タスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x11	TE_QOVR	オーバーフロー（レディ・キュー）

【書式・アセンブラ】

rot_rdq1

7.3.20 rot_rdq2

r o t _ r d q 2	ROTate ReaDy Queue(current task is called in tsk_sts)

【機能】

自タスクをレディ・キューの最後尾に登録します。ただし、本システムコールは、自タスクがter_tsk()のパラメータとしては指定されず、tsk_sts()のパラメータとして指定される場合に使用してください。

【書式・C】

スリム・タイプ： rot_rdq2();

ディバグ・タイプ：
```
unsigned char err;
err = rot_rdq2();
```

【パラメータ】

なし

【解説】

自タスクを、レディ・キューの最後尾に登録します。また、tsk_sts()システムコール発行時の高速検索のための準備処理を行います。

ディバグ・タイプの場合、レディ・キューの最後尾に正常に登録できた場合はリターン・コード「TE_OK」を、レディ・キューに指定タスクを登録する領域がなかった場合はリターン・コード「TE_QOVR」が返値として返されます。

レディ・キューに登録した自タスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x11	TE_QOVR	オーバーフロー（レディ・キュー）

【書式・アセンブラ】

rot_rdq2

7.3.21 rot_rdq3

r o t _ r d q 3	ROTate ReaDy Queue(current task is called in ter_tsk and tsk_sts)

【機能】

自タスクをレディ・キューの最後尾に登録します。ただし、本システムコールは、指定タスクが ter_tsk( )と tsk_sts( )のパラメータとして指定される場合に使用してください。

【書式・C】

スリム・タイプ： rot_rdq3( );

ディバグ・タイプ：
```
unsigned char err;
err = rot_rdq3( );
```

【パラメータ】

なし

【解説】

自タスクを、レディ・キューの最後尾に登録します。また、ter_tsk( )と tsk_sts( )システムコール発行時の高速検索のための準備処理も行います。

ディバグ・タイプの場合、レディ・キューの最後尾に正常に登録できた場合はリターン・コード「TE_OK」を、レディ・キューに指定タスクを登録する領域がなかった場合はリターン・コード「TE_QOVR」が返値として返されます。

レディ・キューに登録した自タスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x11	TE_QOVR	オーバーフロー（レディ・キュー）

【書式・アセンブラ】

rot_rdq3

7.3.22 rot_rdqe

r o t _ r d q e	ROTate ReaDy Queue and Exit task(current task is called in ter_tsk or tsk_sts)

【機能】

自タスクをレディ・キューの最後尾に登録し、自タスクを正常終了します。ただし、本システムコールは、自タスクが ter_tsk( )と tsk_sts( )のパラメータとして指定されない場合、または ter_tsk( )と tsk_sts( )のパラメータ指定が明確でない場合に使用してください。

【書式・C】

```
rot_rdqe( );
```

【パラメータ】

なし

【解説】

自タスクをレディ・キューの最後尾に登録し、自タスクを DORMANT 状態にします。

ディバグ・タイプの場合でも、本システムコールを発行すると処理が終了するためリターン・コードは返されません。

レディ・キューに登録した自タスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

なし

【書式・アセンブラ】

```
rot_rdqe
```

7.3.23 rot_rdq1e

r o t _ r d q 1 e	ROTate ReaDy Queue and Exit task(current task is called in ter_tsk)

【機能】

自タスクをレディ・キューの最後尾に登録し、自タスクを正常終了します。ただし、本システムコールは、指定タスクが ter_tsk( )のパラメータとして指定され、tsk_sts( )のパラメータとしては指定されない場合に使用してください。

【書式・C】

```
rot_rdq1e( );
```

【パラメータ】

なし

【解説】

自タスクをレディ・キューの最後尾に登録し、自タスクを DORMANT 状態にします。また同時に、ter_tsk( )システムコール発行時の高速検索のための準備処理も行います。

ディバグ・タイプの場合でも、本システムコールを発行すると処理が終了するためリターン・コードは返されません。

レディ・キューに登録した自タスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

なし

【書式・アセンブラ】

```
rot_rdq1e
```

保守／廃止

7.3.24 rot_rdq2e

r o t _ r d q 2 e	ROTate ReaDy Queue and Exit task(current task is called in tsk_sts)

【機能】

自タスクをレディ・キューの最後尾に登録し、自タスクを正常終了します。ただし、本システムコールは、指定タスクが ter_tsk( )のパラメータとしては指定されず、tsk_sts( )のパラメータとして指定される場合に使用してください。

【書式・C】

```
rot_rdq2e( );
```

【パラメータ】

なし

【解説】

自タスクをレディ・キューの最後尾に登録し、自タスクを DORMANT 状態にします。また同時に、tsk_sts( )システムコール発行時の高速検索のための準備処理も行います。

ディバグ・タイプの場合でも、本システムコールを発行すると処理が終了するためリターン・コードは返されません。

レディ・キューに登録した自タスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

なし

【書式・アセンブラ】

```
rot_rdq2e
```

7.3.25 rot_rdq3e

r o t _ _ r d q 3 e	ROTate ReaDy Queue and Exit task(current task is called in ter_tsk and tsk_sts)

【機能】

自タスクをレディ・キューの最後尾に登録し、自タスクを正常終了します。ただし、本システムコールは、自タスクが ter_tsk()と tsk_sts()のパラメータとして指定される場合に使用してください。

【書式・C】

```
rot_rdq3e( );
```

【パラメータ】

なし

【解説】

自タスクをレディ・キューの最後尾に登録し、自タスクを DORMANT 状態にします。また同時に、ter_tsk()と tsk_sts()システムコール発行時の高速検索のための準備処理を行います。

ディバグ・タイプの場合でも、本システムコールを発行すると処理が終了するためリターン・コードは返されません。

レディ・キューに登録した自タスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

なし

【書式・アセンブラ】

```
rot_rdq3e
```

保守／廃止

7.3.26 tsk_sts

tsk_sts	get TaSK STatuS

【機能】

タスクの状態を調べます。

【書式・C】

```
tsk_sts(pk_tskst,a_tsk);
```

【パラメータ】

パラメータ	ビット数	型	説明
pk_tskst	16	関数型	タスクの状態を格納する領域
a_tsk	16	関数型	調べるタスクの先頭アドレス

【解説】

指定タスク「a_tsk」がレディ・キューに登録されていれば「TTS_RDY」を、それ以外の場合は「TTS_DMT」を領域「pk_tskst」に返します。

【リターン・コード】

状態	シンボル	発生原因
0x02	TTS_RDY	指定タスクはREADY状態
0x10	TTS_DMT	指定タスクはDORMANT状態

【書式・アセンブラ】

```
tsk_sts  pk_tskst,a_tsk
```

保守／廃止

7.3.27 chg_pri

c h g _ p r i	CHanGe PRIority

【機能】

指定タスクをレディ・キューの先頭に登録し直します。

【書式・C】

スリム・タイプ： chg_pri(a_tsk);

ディバグ・タイプ：
```
unsigned char err;
err = chg_pri(a_tsk);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	関数型	調べるタスクの先頭アドレス

【解説】

指定タスク「a_tsk」がレディ・キューに登録されていれば、指定タスクをレディ・キューの先頭に登録し直します。

ディバグ・タイプの場合、指定タスクがレディ・キューに登録されていれば、リターン・コード「TE_OK」を返し、登録されていなければリターン・コード「TE_DMT」を返します。

レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x0d	TE_DMT	指定タスクがDORMANT状態である

【書式・アセンブラ】

```
chg_pri  a_tsk
```

保守／廃止

7.3.28 ext_tsk

ｅｘｔ＿ｔｓｋ	Exit task

【機能】

自タスクを正常終了します。

【書式・C】

```
ext_tsk();
```

【パラメータ】

なし

【解説】

自タスクを DORMANT 状態にします。

ディバグ・タイプの場合でも、本システムコールを発行すると処理が終了するためリターン・コードは返されません。

【リターン・コード】

なし

【書式・アセンブラ】

```
ext_tsk
```

保守／廃止

7.4 イベントフラグ関連システムコール

この節では、イベントフラグ関連のシステムコールについて説明します。イベントフラグ関連システムコールは、下記に示すものがあります。

set_flg( ), clr_flg( ), rpl_flg( )

7.4.1 set_flg

s e t _ f l g	SET event-FLaG

【機能】

指定イベントフラグに任意のビット・パターンをセットします。

【書式・C】

スリム・タイプ： set_flg(a_flg,setptn);　　ディバグ・タイプ： unsigned char err;
set_flg(a_flg,setptn,err);

【パラメータ】

パラメータ	ビット数	型	説明
a_flg	16	関数型	指定イベントフラグのアドレス
setptn	8	char	セットするビット・パターン
err	8	unsigned char	リターン・コードを格納する領域（ディバグ・タイプのみ）

【解説】

指定イベントフラグ「a_flg」のビット・パターンと、セット・パターン「setptn」とのORデータを指定イベントフラグにセットします。セット・パターン「setptn」では、セットしたいビット値を1に指定してください。

もし、イベントフラグに登録されているタスクの起動条件を満たした場合、登録タスクをレディ・キューの先頭に登録します。

ディバグ・タイプの場合、イベントフラグをセットしたことによって登録タスクが起動した場合、リターン・コード「TE_OK」を返します。起動条件が満たされ、レディ・キューにタスクを登録しようとしたが、タスクを登録する領域がなかった場合はリターン・コード「TE_QOVR」が返されます。また、イベントフラグが正常にセットされた場合は、リターン・コード「TE_SET」を、指定イベントフラグにタスクが登録されていなかった場合は、リターン・コード「TE_NOENT」が返値として返ります。

起動条件を満たし、レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x11	TE_QOVR	オーバーフロー（レディ・キュー）
0x04	TE_SET	イベントフラグを正常にセット
0x05	TE_NOENT	起動タスクが登録されていない

【書式・アセンブラ】

```
set_flg  a_tsk,setptn
```

7.4.2 clr_flg

c l r _ f l g	CLeaR event-FLaG

【機能】

指定イベントフラグの任意のビット・パターンをクリアします。

【書式・C】

```
clr_flg(a_flg,clrptn);
```

【パラメータ】

パラメータ	ビット数	型	説明
a_flg	16	関数型	指定イベントフラグのアドレス
clrptn	8	char	クリアするビット・パターン

【解説】

指定イベントフラグ「a_flg」のビット・パターンを、クリア・パターン「clrptn」で指定したビット・パターンでクリアします。クリア・パターン「clrptn」では、クリアしたいビット値を0で指定してください（クリアしたくないビットには、1をセットしてください）。イベントフラグをクリアしたことによって登録タスクの条件が成立しても、タスクは起動しません。

【リターン・コード】

なし

【書式・アセンブラ】

```
clr_flg  a_flg,clrptn
```

7.4.3 rpl_flg

r p l _ f l g	RePLace event-FLaG

【機能】

指定イベントフラグのビット・パターンを、指定ビット・パターンと置き換えます。

【書式・C】

スリム・タイプ： rpl_flg(a_flg,rplptn); ディバグ・タイプ： unsigned char err;
rpl_flg(a_flg,rplptn,err);

【パラメータ】

パラメータ	ビット数	型	説明
a_flg	16	関数型	指定イベントフラグのアドレス
rplptn	8	char	置き換えるビット・パターン
err	8	unsigned char	リターン・コードを格納する領域（ディバグ・タイプのみ）

【解説】

指定イベントフラグ「a_flg」のビット・パターンを、指定パターン「rplptn」と置き換えます。

もし、イベントフラグに登録されているタスクの起動条件を満たした場合、登録タスクをレディ・キューの先頭に登録します。

ディバグ・タイプの場合、イベントフラグを置き換えたことによって登録タスクが起動した場合、リターン・コード「TE_OK」を返します。起動条件が満たされ、レディ・キューにタスクを登録しようとしたが、タスクを登録する領域がなかった場合はリターン・コード「TE_QOVR」が返されます。また、イベントフラグが正常に置き換えられた場合は、リターン・コード「TE_SET」を、指定イベントフラグにタスクが登録されていなかった場合は、リターン・コード「TE_NOENT」が返値として返ります。

起動条件を満たし、レディ・キューに登録したタスクが動作する際は、一つ前に動作していたタスクの割り込み受付状態を引き継ぎますので、割り込みの受付状態についてはタスクごとに行ってください。

【リターン・コード】

リターン・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x11	TE_QOVR	オーバーフロー（レディ・キュー）
0x04	TE_SET	イベントフラグを正常にセット
0x05	TE_NOENT	起動タスクが登録されていない

【書式・アセンブラ】

rpl_flg a_tsk,rplptn

保守／廃止

第８章　OSの管理領域

8.1　メモリ容量

本OSが使用するメモリ容量は、以下の通りです。

・コード・サイズ
　・スリム・タイプ：　　最大約 1.7K バイト
　・ディバグ・タイプ：　最大約 1.8K バイト

・データ・サイズ
　・レディ・キュー
　　４×（同時にレディ・キューに登録するタスクの最大数＋１）バイト

　・タイマ・キュー
　　４×（同時にタイマ・キューに登録するタスクの最大数＋１）バイト

　・イベントフラグ本体
　　２×（イベントフラグの個数）＋１バイト

　・イベントフラグ補足情報
　　４×（イベントフラグ補足情報の個数）バイト

　・システム管理領域（saddr 領域）
　　８バイト

8.2　メモリ容量の見積もり方

OSが使用するRAMの算出方法を示します。

例）
・タスク数：　　４０個（タイマを使用するタスクが１３個）
・イベントフラグ数　　１０個
・イベントフラグ補足情報　　５個（RAM上に定義）

※　タスク数は４０個ありますが、レディ・キューに登録するタスクの最大数を１５個とします。
※　タイマを使用するタスク数は 13 個ありますが、タイマ・キューに登録するタスクの最大数を９個とします。

レディ・キューのサイズ：	４×（１５個＋１）バイト＝	６４バイト
タイマ・キューのサイズ：	４×（　９個＋１）バイト＝	４０バイト
イベントフラグ本体テーブルのサイズ：	２×１０個バイト＝	２０バイト
イベントフラグ補足情報テーブルのサイズ：	４×　５個バイト＝	２０バイト
システム管理領域：		８バイト
合計		１５２バイト

例題のシステムでは、１５２バイトのRAMが必要になります。

第9章　アプリケーション開発手順

この章では実際にユーザ・アプリケーションを作成する手順と注意事項、本OSの初期化処理について述べます。

9.1　ロード・モジュール作成手順

図 9.1にロード・モジュールの作成手順を示します。

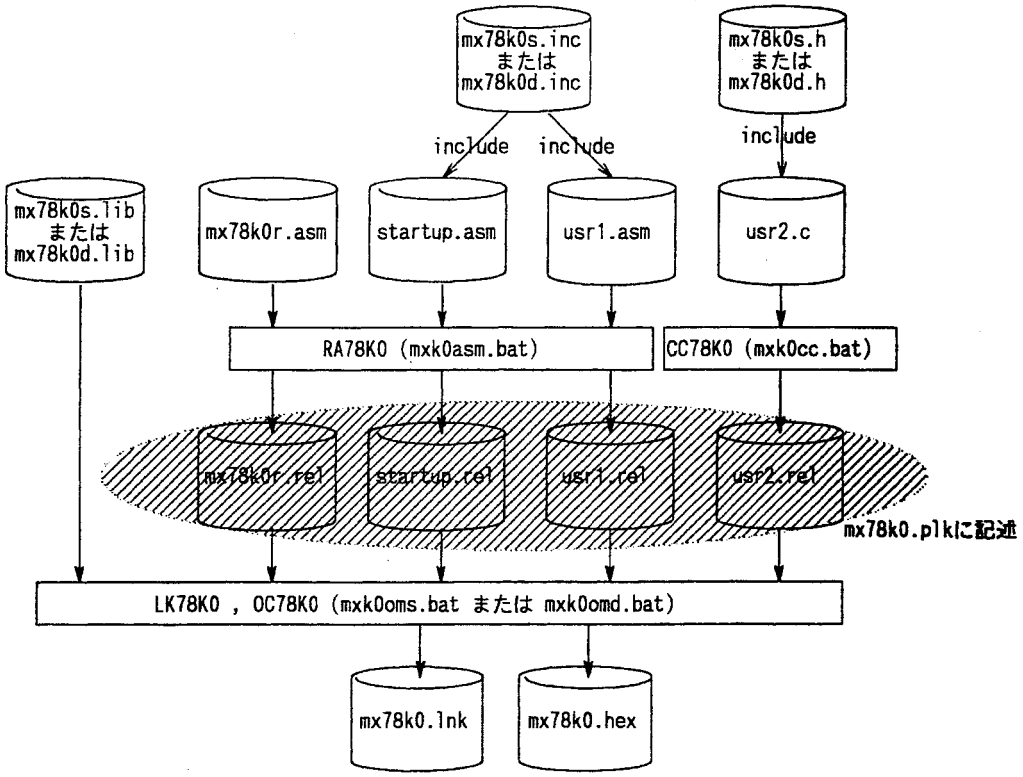


図 9.1：ロードモジュール作成手順

図 9.1に示した各種ファイルの意味を以下に示します。（製品に添付されているファイルは下線で示します。但し、サンプルとして添付されているファイルも含みます。）

mx78k0s.inc	アセンブリ言語用スリム・タイプ・インクルード・ファイル
mx78k0d.inc	アセンブリ言語用ディバグ・タイプ・インクルード・ファイル
mx78k0s.h	C言語用スリム・タイプ・インクルード・ファイル
mx78k0d.h	C言語用ディバグ・タイプ・インクルード・ファイル
mx78k0s.lib	スリム・タイプ・システムコール・ライブラリ・ファイル
mx78k0d.lib	ディバグ・タイプ・システムコール・ライブラリ・ファイル
mx78k0r.asm	メモリ設定ファイル
startup.asm	スタートアップ・ルーチン
usr1.asm	ユーザ作成タスク（アセンブリ言語）
usr2.c	ユーザ作成タスク（C言語）
mx78k0.plk	リンカ・パラメータ・ファイル
mx78k0.lnk	ロード・モジュール（ROM化情報を含まない）
mx78k0.hex	ロード・モジュール（ROM化情報を含む）

9.2 ロードモジュール作成上の注意事項

ここでは付属のメイク用バッチ・ファイルを使用する上での注意事項と、オプションについて述べます。

1．バッチ・ファイルについて
- アセンブラ起動用バッチ・ファイル(mxk0asm.bat)、コンパイラ起動用バッチ・ファイル(mxk0cc.bat)の起動場所は特に決まっていません。これらのバッチ・ファイルにより出力されるファイル（拡張子.rel）はカレント・ディレクトリに作成されます。
- ロードモジュール作成用バッチ・ファイルを使用する場合は、リンカ・パラメータ・ファイル(mx78k0.plk)とリンクさせるファイル群(拡張子.rel)を同一ディレクトリに置き、それらが存在するディレクトリからバッチ・ファイルを起動してください。
- また、システムコール・ライブラリ・ファイルをディフォールト（¥nectools¥lib78k0）以外のディレクトリにインストールした場合は、ロードモジュール作成用バッチ・ファイルに記述されている LK78K0 のオプションを変更してください。

2．アセンブラ（RA78K0）のオプション
- RA78K0 のオプションは”-s -nca”を使用してください。このオプションを無効にするようなオプション指定をおこなうと、正常にアセンブル、リンクできない場合があります。

3．コンパイラ（CC78K0）のオプション
- CC78K0 のオプションは”-qcr -s -nca”を使用してください。このオプションを無効にするようなオプション指定をおこなうと、正常にコンパイル、リンクできない場合があります。
- “-za”オプションは使用しないでください。
- “-zo”オプションは使用しないでください。

4．リンカ（LK78K0）のオプション
- LK78K0 のオプションから”-s”を削除すると、スタック・ポインタの自動生成が行われません。”-s”オプションを削除する場合は、メモリ設定ファイル（“mx78k0r.asm”）でスタック・ポインタの先頭番地（”_@STBEG”）を定義してください。

9.3 ユーザ・タスク作成上の注意事項

ここでは、ユーザ・タスク作成の際の注意事項について述べます。但し、各システムコールの使用方法(書式)については、「第７章　システムコール」を参照してください。

9.3.1 アセンブリ言語でタスクを記述する場合

ユーザ・タスク記述例(アセンブリ言語)

```
; User Task Sample (usr1.asm)
NAME    USR1
$INCLUDE(MX78K0S.INC)           ①
EXTRN   n_500ms                 ②
EXTRN   _task3
PUBLIC  _task1
PUBLIC  _task2
        CSEG
_task1: CLR1    P1.0
        sta_tsk _task2
        sta_tskt _task3,n_500ms ②
        ext_tsk                 ③
_task2: BT      P0.0,$label1
        rot_rdqe                ③
label1: ext_tsk                 ③
        END
```

①使用するタイプに合ったインクルード・ファイルを使用してください。
(上記の例ではスリム・タイプを使用しています。)

②時間を引数にとるシステムコール（sta_tskt,chg_tskt など）を使う場合、引数に指定する時間は、OS が使用するタイマ割り込みのインターバル時間から計算した値を指定してください。
(例)インターバル時間が 10ms の場合、n_500ms＝50 に定義する。

③タスクの最後は（ext_tsk）等の、自タスクの終了を伴うシステムコールを記述してください。

④ディバグ・タイプを使用する場合、システムコールのリターン・コードはＣレジスタに格納されます。

9.3.2　C言語でタスクを記述する場合

ユーザ・タスク記述例（C言語）

```
/* User Task Sample (usr2.c) */
#pragma SFR
#include "mx78k0s.h"                    ①
extern int n_10ms;                      ②
extern void task1( );
extern void task2( );
void task3( );
void task4( );

void task3( )
{
        unsigned char pk_tskst;
        tsk_sts(&pk_tskst,task2);
        if (pk_tskst == TTS_RDY)
        {
                ter_tsk(task2);
                P1.0 = 1;
                sta_tskte(task4,n_10ms);②③
        }
        sta_tske(task1);                ③
}
void task4( )
{
        P1.0 = 0;
        sta_tske(task1);                ③
}
```

①使用するタイプに合ったインクルード・ファイルを使用してください。
（上記の例ではスリム・タイプを使用しています。）

②時間を引数にとるシステムコール（sta_tskt,chg_tskt など）を使う場合、引数に指定する時間は、OS が使用するタイマ割り込みのインターバル時間から計算した値を指定してください。

③タスクの最後は（ext_tsk）等の、自タスクの終了を伴うシステムコールを記述してください。

④イベントフラグを使用する場合、イベントフラグ本体テーブルおよび補足情報テーブルの型宣言と終端の設定はユーザが自身で行ってください。詳細は、「10.3.6　イベントフラグの確保」を参照してください。

⑤ディバグ・タイプを使用する場合、システムコールのリターン・コードはシステムコールの返値（unsigned char 型）として返されます。

9.4 ユーザ・オウン・コーディング部

ここでは、ユーザ・タスク以外にユーザが自身で作成しなければならない部分について述べます。

9.4.1 初期化処理部

システム起動時に行うハードウェアおよびソフトウェアの初期化を行う部分です。システム初期化処理については、「第 10 章　システム初期化」を参照してください。

9.4.2 タイマ処理部

タイマ・キューを使用するシステムコールを使用する場合は、タイマ・ハンドラおよびタイマ割り込み禁止／許可ルーチンの作成が必要です。

1．タイマ・ハンドラ

OS が使用するタイマの割り込み処理を記述します。あらかじめ、初期化処理内で OS が占有するタイマを 1 本確保し、割り込みベクタを設定しておいてください。（詳細は各デバイスのユーザーズ・マニュアルを参照してください。）
割り込み処理内では、OS のタイマ処理ルーチン（"_MX_int"）を呼び出す命令を記述してください。

例)タイマ割り込みハンドラの先頭番地が"TM1INT"の場合

```
TM1INT:
        CALL    !_MX_int
        RETI
```

(注)タイマ割り込みハンドラ内では OS のタイマ処理ルーチン("_MX_int")を呼び出す命令以外は記述しないでください。ハンドラ内にユーザ・プログラムを記述すると、多重割り込みの影響などによって、OS のタイマ処理やユーザ・プログラムが正常に動作しない場合があります。

2．タイマ割り込み禁止ルーチン

OS が使用するタイマ割り込みを禁止するルーチンを作成します。この処理の先頭番地は"_Prohibit_MX_int"とし、"RET"命令で終了してください。

例)ＴＭ１を使用している場合

```
_Prohibit_MX_int:
        PUSH    PSW
        DI
        SET1    MKOH.1          ; TMMK1←1
        POP     PSW
        RET
```

3．タイマ割り込み許可ルーチン

OS が使用するタイマ割り込みを許可するルーチンを作成します。この処理の先頭番地は"_Permit_MX_int"とし、"RET"命令で終了してください。

例)ＴＭ１を使用している場合

```
_Permit_MX_int:
        PUSH    PSW
        DI
        CLR1    MKOH.1          ; TMMK1←0
        POP     PSW
        RET
```

9.4.3 タスクを記述する際の注意事項

タスクが使用するレジスタバンクは、OS が使用するレジスタバンクと一緒にしないでください。タスクが、OS を同じレジスタバンクを使用すると動作の保証ができなくなります。

9.4.4 割り込みハンドラを記述する際の注意事項

割り込みハンドラから発行できるシステムコールには制限があります。対象外のシステムコールを発行した場合、アプリケーションが正常に動作しないことがありますので注意してください。
割り込みハンドラの終わりは"RETI"命令を使用してください
また、割り込みハンドラが使用するレジスタバンクは、OS が使用するレジスタバンクと一緒にしないでください。割り込みハンドラが、OS を同じレジスタバンクを使用すると動作の保証ができなくなります。

第１０章　システム初期化処理

この章ではシステム起動時に行うハードウェア、および本 OS のシステム・エリアの初期化について述べます。システム初期化処理にはハードウェアの初期化を行う部分とソフトウェアの初期化を行う部分があります。

10.1　システム初期化処理の概要

システム初期化処理は下記のような項目を行ってください。

	処理内容
ハードウェア	ＣＰＵの初期化 ・ポート ・タイマ ・割り込み ・ＲＡＭ 周辺の初期化
ソフトウェア	OS が使用する領域の確保 ・スタック・ポインタの設定 ・レジスタバンクの設定 ・キューの確保 ・システム・ワーク・エリアの確保 ・イベントフラグの確保 ・“tsk_sts”用領域の確保 OS の初期化 ・キューの初期化 ・イベントフラグの設定 ・初期タスクの起動

表 10.1　初期化処理例

10.2　ハードウェアの初期化

ハードウェアの初期化は、ユーザ・システムが動作するのに必要なハードウェアの初期化を行ってください。（ハードウェアの初期化に関しては、各デバイスのユーザーズ・マニュアルを参考にしてください。）

また、タイマ関連のシステムコールを使用する場合は、OS 用にタイマを１本確保してください。

ハードウェアの初期化に関する部分はサンプルが添付されていませんので、ユーザが自身で作成する必要があります。

10.3 ソフトウェアの初期化

ソフトウェアの初期化では、OSが使用する領域の確保・初期化、初期タスクの起動を行います。ソフトウェアの初期化は添付されているメモリ設定ファイルのサンプル・ファイル（"mx78k0r.asm"）を修正することにより、行うことができます。（以下の説明中の例では、ユーザが設定する部分を xx で示します。そのほかの部分はサンプルのメモリ設定ファイルに記述されていますので、特に変更する必要はありません。記述例中の PUBLIC 宣言は、OSに対して最低限必要なものについてのみ記述していますので、実際にはユーザ・システムに合わせて追加する必要があります。また、EXTRN 宣言は省略しています。）

以下にソフトウェアの初期化手順を示します。

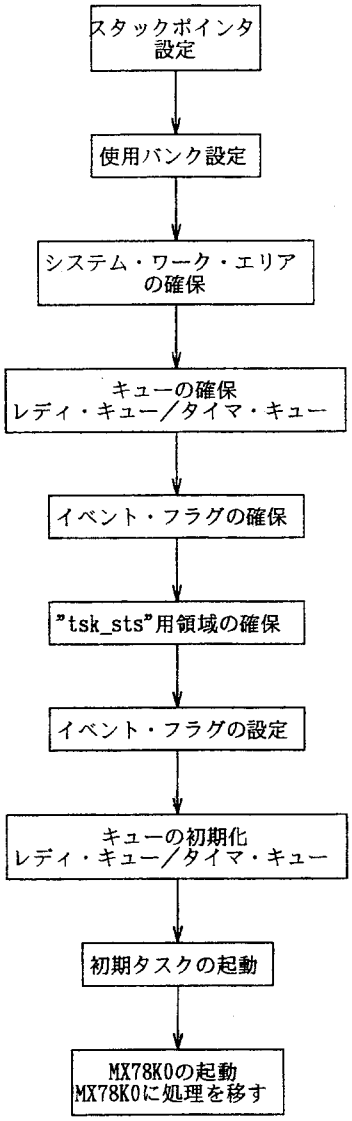


図 10.1 ソフトウエアの初期化手順

10.3.1 スタック・ポインタの設定

本 OS は、リンカで自動生成できるスタック・ポインタを使用しています。添付のメイク用バッチ・ファイルを使用する場合は、スタック・ポインタは自動生成されるので、初期化処理内で設定する必要はありませんが、リンカ・オプションで"-s"を指定しない場合は、メモリ設定ファイルに下記の記述を追加してスタック・ポインタ（_@STBEG）を設定してください。

```
PUBLIC  _@STBEG
_@STBEG EQU     xxxxH   ; スタック・ポインタの先頭番地
```

10.3.2 レジスタ・バンクの設定

本 OS は、作業用に任意のレジスタ・バンクを 1 バンク占有します。メモリ設定ファイルでは、下表に基づいて値を設定します。

使用する レジスタ・バンク	設定値
バンク０	０Ｄ０６１Ｈ
バンク１	０Ｄ８６１Ｈ
バンク２	０Ｆ０６１Ｈ
バンク３	０Ｆ８６１Ｈ

表 10.2：レジスタ・バンクの値

```
PUBLIC  _MX_Bank
_MX_BankEQU     xxxxH   ; レジスタ・バンク設定
```

10.3.3 システム・ワーク・エリアの確保

OS が使用するシステム・ワーク・エリア（SWA）を確保します。メモリ設定ファイルでは、SWA の先頭番地を設定することにより、領域を確保します。SWA は、必ずショート・ダイレクト・アドレッシング領域に配置してください。

```
PUBLIC          _MX_ctask, _MX_tq_s, _MX_tqv_s
PUBLIC          _MX_ter_mem, _MX_sts_mem, az_arg
_MX78K0_WORK    EQU     xxxxH           ; SWA2 の先頭番地
_MX_SWA2        DSEG    AT _MX78K0_WORK
_MX_ctask:      DS      2
_MX_tq_s:       DS      1
_MX_tqv_s:      DS      1
_MX_ter_mem:    DS      1
_MX_sts_mem:    DS      1
az_arg:         DS      2
```

10.3.4 レディ・キューの確保

レディ・キューを確保するRAM先頭アドレスと、レディ・キューに同時に登録されるタスクの最大数を設定し、レディ・キューの領域を確保します。同時に登録されるタスクの最大数は1～63の範囲で設定することができます。

```
PUBLIC          _MX_rq_StartAddress
PUBLIC          _MX_rq_e0
_MX_rq_StartAddress     EQU     xxxxH           ; レディ・キューの先頭番地
_MX_rq_Max              EQU     xx              ; 最大タスク数
mx78k0rq                DSEG    _MX_rq_Start_Address
_MX_rq_area:            DS      (_MX_rq_Max + 1) * 4
_MX_rq_e0               EQU     _MX_rq_Max*4
```

(注)レディ・キューに使用する領域は、RAM上に連続して配置してください。

10.3.5 タイマ・キューの確保

タイマ・キューを確保するRAM先頭アドレスと、タイマ・キューに登録されるタスクの最大数を設定します。但し、タイマ関連のシステムコールを使用しない場合は、タイマ・キューの設定をする必要はありません。

タイマ・キューに登録されるタスクの最大数は1～62の範囲で設定することができます。

```
PUBLIC          _MX_tq_StartAddress
PUBLIC          _MX_tq_a_e0
PUBLIC          _MX_tq_r_e0
PUBLIC          _MX_tq_half
_MX_tq_StartAddress     EQU     xxxxH           ; タイマ・キューの先頭番地
_MX_tq_Max              EQU     xx              ; 最大タスク数
mx78k0tq                DSEG    _MX_tq_Start_Address
_MX_tq_area:            DS      (_MX_tq_Max + 1) * 4
_MX_tq_r_e0             EQU     _MX_tq_Max*4
_MX_tq_a_e0             EQU     _MX_tq_StartAddress + _MX_tq_r_e0
_MX_tq_half             EQU     ((_MX_tq_Max + 1) SHR 1) * 4
```

(注)タイマ・キューに使用する領域は、RAM上に連続して配置してください。

10.3.6 イベントフラグの確保

イベントフラグ本体テーブルとイベントフラグ補足情報テーブルの配置を行います。ただし、イベントフラグ関連のシステムコールを使用しない場合は配置の必要はありません。イベントフラグ本体テーブルは、RAM 上の連続した領域に配置してください。また、イベントフラグ本体テーブルの終端には１バイトの領域を確保し、イベントフラグ使用前に 0xff を設定してください。イベントフラグ補足情報テーブルは ROM／RAM のどちらに配置しても構いませんが、テーブル全体を連続した領域に配置しなければなりません。各テーブルの配置は必ずしもシステム初期化処理内で行う必要はありません。

１．アセンブリ言語で配置する場合

```
    PUBLIC          _MX_flags
    PUBLIC          _MX_flg_tbl
            DSEG
    _MX_flags:                          ; イベントフラグ本体テーブル
    _flag1:         DS      2           ; フラグ_flag1 の領域
    _flag2:         DS      2           ; フラグ_flag2 の領域
    _flag3:         DS      2           ; フラグ_flag3 の領域
    _flag_end:      DS      1           ; フラグ終端設定用領域

                    CSEG
    _MX_flg_tbl:                        ; 補足情報テーブル
            DB      EVENT_XXX           ; 待ちモード(EVENT_OR,EVENT_AND)
            DB      xxxxxxxxB           ; 待ちビットパターン
            DW      xxxx                ; 起動するタスクの先頭番地
            DB      EVENT_XXX           ; 補足情報の個数分設定
            DB      xxxxxxxxB
            DW      xxxx
```

(注 1) 本体テーブルの先頭ラベルは"_MX_flags"、補足情報テーブルの先頭ラベルは"_MX_flg_tbl"とし、これらを省略したり変更したりしないでください。それ以外のラベルは例として挙げたものであり、ユーザが自由に変更することができます。

(注２) フラグ使用前に必ず本体テーブルの終端を 0xff に設定してください。

(注３) 上例では補足情報テーブルを CSEG 擬似命令を用いて ROM に配置しています。RAM に配置するときは DSEG 擬似命令を使用して配置し、補足情報は初期化処理内で別に設定してください。詳細は「10.3.9 イベントフラグの設定」を参照してください。

2．C言語で配置する場合

```
struct flaginfo                          /* 本体テーブルの配置 */
{
        unsigned char flag1[2];          /* イベントフラグ確保 */
        unsigned char flag2[2];
        unsigned char flag3[2];
        unsigned char flgend;            /* フラグ終端子用領域 */
} MX_flags;

struct event_tbl                         /* 補足情報の型宣言 */
{
        const unsigned char wfmode;      /* 待ちモード(EVENT_OR,EVENT_AND) */
        const unsigned char waiptn;      /* 待ちビットパターン */
        const void (*func)(void);        /* 起動するタスクの先頭番地 */
};

const struct event_tbl MX_flg_tbl[2] =   /* 補足情報テーブルの配置 */
{                                        /* 補足情報の個数が2個の場合 */
        {EVENT_XXX, XX, XXXX},           /* {wfmode,waiptn,func} */
        {EVENT_XXX, XX, XXXX}
};
```

(注1) 本体テーブルの変数名は"MX_flags"、補足情報テーブルの変数名は"MX_flg_tbl[ ]"を使用し、これらの変数名を変更しないでください。ただし、各テーブルの構造体のメンバ名は例として挙げたものであり、ユーザが自由に変更することができます。

(注2) 本体テーブルの各イベントフラグは上の例に記述されているように、unsigned char 型の大きさが2バイトとなる配列を使用してください。同じ2バイトの大きさの領域でも、int 型を使用して確保すると正常に動作しない場合があります。

(注3) 補足情報テーブルの配置をC言語で行わない場合は、補足情報テーブルの型宣言を行う必要はありません。

(注4) フラグ使用前に必ず本体テーブルの終端を 0xff に設定してください。

(注5) フラグ本体テーブルおよび補足情報テーブルは、外部変数として定義してください。

10.3.7 tsk_sts 用領域の確保

tsk_sts()システムコールを使用する場合は、タスクの状態を返すために使用する領域を確保します。tsk_sts()システムコールの第一引数にはここで確保した領域のアドレスを指定します。

```
_pk_tskst:      DS      1
```

(注) tsk_sts()システムコール用領域の確保は、メモリ設定ファイル内で確保してシステム全体で一つの領域を使用する方法の他に、tsk_sts()システムコールを使用するタスクごとに1バイトの大きさの変数を確保して使用する方法もあります。

10.3.8　キューの初期化

各領域の確保が終了した後、キューの初期化を行います。キューの初期化の方法は使用するキューの種類によって２通りあります。

１．レディ・キューのみを使用する場合

レディ・キュー初期化ルーチンを呼び出します。

```
        CALL    !_MX_init_rq
```

２．レディ・キューとタイマ・キューを使用する場合

レディ・キュー／タイマ・キュー初期化ルーチンを呼び出します。

```
        CALL    !_MX_init_tq
```

10.3.9 イベントフラグの設定

イベントフラグ本体テーブルの内容を０で初期化し、終端子(０ｘｆｆ)を設定します。
また、イベントフラグ補足情報テーブルを RAM 上に配置した場合は、イベントフラグ補足情報の設定を行います。

※ 以下の例のイベントフラグ本体テーブルおよび補足情報テーブルの構造は、「10.3.6 イベントフラグの確保」に記述されている例に基づいて記述されています。

１．アセンブリ言語で設定する場合

```
        MOVW    AX,#00H
        MOVW    !_flag1,AX              ; フラグ_flag1 の内容初期化
        MOVW    !_flag2,AX              ; フラグ_flag2 の内容初期化
        MOVW    !_flag3,AX              ; フラグ_flag3 の内容初期化
        MOV     A,#0FFH
        MOV     !_flag_end,A            ; 終端子(0xff)設定

        MOVW    AX,#_MX_flg_tbl         ; 補足情報の設定
        MOVW    HL,AX
        MOV     A,#EVENT_XXX            ; １つ目のフラグの待ちモード
        MOV     [HL],A
        MOV     A,#xxxxxxxxB            ; 待ちビット・パターン
        MOV     [HL+1],A
        MOVW    AX,#xxxx                ; 起動タスク
        MOV     [HL+3],A
        XCH     A,X
        MOV     [HL+2],A

        MOV     A,#EVENT_XXX            ; ２つ目のフラグの待ちモード
        MOV     [HL+4],A
        MOV     A,#xxxxxxxxB            ; 待ちビット・パターン
        MOV     [HL+5],A
        MOVW    AX,#xxxx                ; 起動タスク
        MOV     [HL+7],A
        XCH     A,X
        MOV     [HL+6],A
```

２．Ｃ言語で設定する場合

```
void flag_init( )
{
        MX_flags.flag1[0] = 0x00;               /* 各フラグ内容のクリア */
        MX_flags.flag1[1] = 0x00;
        MX_flags.flag2[0] = 0x00;
        MX_flags.flag2[1] = 0x00;
        MX_flags.flag3[0] = 0x00;
        MX_flags.flag3[1] = 0x00;
        MX_flags.flgend = 0xff;                 /* 終端子(0xff)設定 */

                                                /* 補足情報の設定 */
        MX_flg_tbl[0].wfmode = EVENT_XXX;       /* １つ目のフラグの待ちモード */
        MX_flg_tbl[0].waiptn = xx;              /* 待ちビット・パターン */
        MX_flg_tbl[0].func = xxxx;              /* 起動タスク */
        MX_flg_tbl[1].wfmode = EVENT_XXX;       /* ２つ目のフラグの待ちモード */
        MX_flg_tbl[1].waiptn = xx;              /* 待ちビット・パターン */
        MX_flg_tbl[1].func = xxxx;              /* 起動タスク */
}
```

10.3.10　初期タスクの起動

OS が起動した時点で、レディ・キューに登録されていなければならないタスクを起動し、レディ・キューに登録します。

```
sta_tsk xxxx
sta_tsk xxxx
```

10.3.11　OS の起動

制御を OS に移します。

```
BR      _MX_start
```

OS が動作すると、レディ・キューの先頭に登録されているタスクが実行を開始します。

付録１　予約語一覧

予約語を以下に示します。

az_arg,
chg_pri, chg_tskt, chg_tskte, clr_flg,
EVENT_AND, EVENT_OR, ext_tsk,
JudgeShift,
MEM_INVALID,
rot_rdq, rot_rdq1e, rot_rdq2e, rot_rdq3e, rot_rdqe, rot_rdq1e, rot_rdq2e, rot_rdq3e, rpl_flg,
set_flg, sta_tsk, sta_tsk1, sta_tsk2, sta_tsk3, sta_tske, sta_tsk1e, sta_tsk2e, sta_tsk3e,
sta_tskf, sta_tskp, sta_tskpe, sta_tskt, sta_tskte,
TE_DMT, TE_OK, TE_QOVR, ter_tsk, ter_tskf, ter_tskt, tsk_sts, TTS_DMT, TTS_RDY,
_@STBEG,
_chg_pri, _chg_tskt, _chg_tskte,
_end_itdsp, _ext_tsk,
_MX_BACK, _MX_Bank, _MX_ctask, _MX_DeleteTask, _MX_end, _MX_end_dsp, _MX_flags, _MX_flg_tbl,
_MX_FRONT, _MX_init_rq, _MX_init_tq, _MX_int, _MX_rq_e0, _MX_rq_StartAddress, _MX_Search1,
_MX_Search2, _MX_start, _MX_sts_mem, _MX_ter_mem, _MX_tqv_s, _MX_tq_a_e0,_MX_tq_half,
_MX_tq_r_e0, _MX_tq_s, _MX_tq_StartAddress, _Permit_MX_int, _Prohibit_MX_int,
_rot_rdq, _rot_rdq1e, _rot_rdq2e, _rot_rdq3e, _rot_rdq9, _rot_rdqe, _rot_rdq1e, _rot_rdq2e,
_rot_rdq3e, _rot_rdq9e,
_Shift1, _Shift2, _sta_tsk, _sta_tsk1, _sta_tsk2, _sta_tsk3,_sta_tsk9, _sta_tske, _sta_tsk1e,
_sta_tsk2e, _sta_tsk3e, _sta_tsk9e, _sta_tskp, _sta_tskpe, _sta_tskt, _sta_tskte,
_ter_tsk, _ter_tskf, _ter_tskt, _tsetflg, _tsk_sts
TE_SET,TE_NOENT

付録２　システムコール別使用スタックサイズ一覧

各システムコール内部で使用するスタックのサイズを以下に示します。

システムコール名	使用スタックサイズ	
	スリム・タイプ	ディバグ・タイプ
sta_tsk	5バイト	5バイト
sta_tsk1	5バイト	5バイト
sta_tsk2	5バイト	5バイト
sta_tsk3	5バイト	5バイト
sta_tske	5バイト	5バイト
sta_tsk1e	5バイト	5バイト
sta_tsk2e	5バイト	5バイト
sta_tsk3e	5バイト	5バイト
sta_tskp	11バイト	11バイト
sta_tskpe	5バイト	5バイト
ter_tsk	10バイト	10バイト
sta_tskt	14+αバイト	14+αバイト
sta_tskte	16+αバイト	16+αバイト
chg_tskt	16+αバイト	16+αバイト
chg_tskte	16+αバイト	16+αバイト
ter_tskt	10+αバイト	10+αバイト
sta_tskf	0バイト	0バイト
ter_tskf	10バイト	10バイト
rot_rdq	3バイト	3バイト
rot_rdq1	3バイト	3バイト
rot_rdq2	3バイト	3バイト
rot_rdq3	3バイト	3バイト
rot_rdqe	3バイト	3バイト
rot_rdq1e	3バイト	3バイト
rot_rdq2e	3バイト	3バイト
rot_rdq3e	3バイト	3バイト
tsk_sts	12バイト	12バイト
chg_pri	10バイト	10バイト
ext_tsk	2バイト	2バイト
set_flg	15バイト	15バイト
clr_flg	0バイト	0バイト
rpl_flg	15バイト	15バイト
タイマ割り込み処理 (_MX_int)	14バイト	14バイト
ﾚﾃﾞｨｷｭｰの初期化 (_MX_init_rq)	3バイト	3バイト
ﾀｲﾏｷｭｰの　初期化 (_MX_init_tq)	3バイト	3バイト

α：タイマ割り込み禁止／許可ルーチンで使用するスタックサイズ

付録２：スタック・サイズ

保守／廃止

お問い合わせ先

【技術的なお問い合わせ先】

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

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

【営業関係お問い合わせ先】

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

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

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

【資料の請求先】

上記営業関係お問い合わせ先またはNEC特約店へお申しつけください。

【インターネット電子デバイス・ニュース】

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

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］MX78K0　オペレーティング・システム　ユーザーズ・マニュアル　基礎編
（U12257JJ2V1UMJ1（第2版））

［お名前など］（さしつかえのない範囲で）

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

1．ご評価（各欄に○をご記入ください）

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

2．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

3．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

4．ご意見，ご要望

5．このドキュメントをお届けしたのは
NEC販売員，特約店販売員，その他（　　）

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

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

2000.6

キリトリ