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

RENESAS

保守／廃止

78K/Ⅳシリーズ用OS

MX78K4

基礎編

資料番号 U11779JJ1V0UM00（第1版）
発行年月 July 1996 P

保守／廃止

TRON は、The Realtime Operating system Nucleus の略称です。
ITRON は、Industrial TRON の略称です。
MS-DOS は、米国マイクロソフト社の登録商標です。
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。

保守／廃止

はじめに

このたびは、NEC 78K/Ⅳシリーズの組み込み用ソフトウェアである『78K/Ⅳシリーズ用 OS MX78K/Ⅳ』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/Ⅳシリーズ用 OS　MX78K/Ⅳ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、“CC78K4 C コンパイラ”と“RA78K4　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　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３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４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５章　割り込み機能
　割り込みについて説明します。

第６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７章　初期化処理
　本 OS の初期化について説明します。

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

保守／廃止

《 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
RA78K4 アセンブラ・パッケージ 言語編	U11162JJ1V0UM00
RA78K4 アセンブラ・パッケージ 操作編	U11334JJ1V0UM00
RA78K シリーズ 構造化アセンブラ・プリプロセッサ	EEU-817
CC78K4 シリーズ C コンパイラ 言語編	EEU-961
CC78K4 シリーズ C コンパイラ 操作編	EEU-960
78K/4 シリーズ 命令編	IEU-844
idea-L　スクリーン・エディタ　入門編	U10094JJ1V0UM00

保守／廃止

目次

保守／廃止

図目次

保守／廃止

保守／廃止

第１章　概要

MX78K4 は、リアルタイム OS よりも OS が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 シリーズ用の OS です。MX78K4 は、78K のリアルタイム OS である RX78K シリーズへの移行性をよくするために、μITRON 仕様のサブセット仕様となっています。

以下この章では、MX78K4 の構成と特徴、環境について述べます。

1.1では構成について、1.2提供ファイルについて、1.3では特徴について、1.4では機能概要について、1.5では実行環境について、1.6ではアプリケーションの開発環境について、1.7では RX78K シリーズとの違いを述べています。

1.1　構成

図 1.1にシステムの構成を示します。

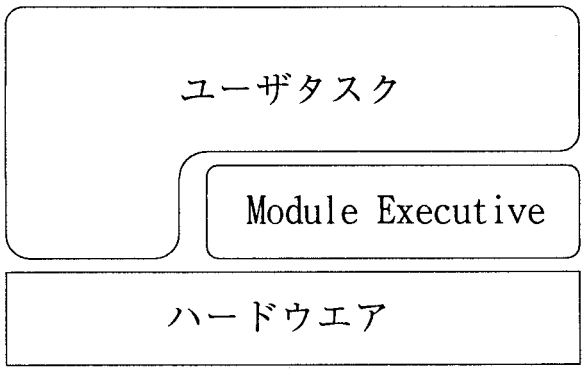


図 1.1　システム構成図

1.2 提供ファイル

提供ファイル一覧（オブジェクト版）

NECTOOLS		
├ BIN		
	├ MX78K4IL.LIB	idea-L用ライブラリ・ファイル（ラージ・モデル用）
	├ MX78K4IS.LIB	idea-L用ライブラリ・ファイル（スモール・モデル用）
	├ MXK4OMLD.BAT	ロードモジュール作成用バッチ・ファイル（ラージ・モデルのディバグ・タイプ用）
	├ MXK4OMLS.BAT	ロードモジュール作成用バッチ・ファイル（ラージ・モデルのスリム・タイプ用）
	├ MXK4OMSD.BAT	ロードモジュール作成用バッチ・ファイル（スモール・モデルのディバグ・タイプ用）
	├ MXK4OMSS.BAT	ロードモジュール作成用バッチ・ファイル（スモール・モデルのスリム・タイプ用）
	├ MXK4ASL.BAT	ソース・ファイル・アセンブル用バッチ・ファイル（ラージ・モデル用）
	└ MXK4ASS.BAT	ソース・ファイル・アセンブル用バッチ・ファイル（スモール・モデル用）
├ DOC		
	├ MX78K4L.TXT	ラージ・モデル用のドキュメント・ファイル
	└ MX78K4S.TXT	スモール・モデル用のドキュメント・ファイル
├ INC78K4		
	├ MX78K4LD.H	ラージ・モデルのディバグ・タイプ用ヘッダ・ファイル
	├ MX78K4LS.H	ラージ・モデルのスリム・タイプ用ヘッダ・ファイル
	├ MX78K4SD.H	スモール・モデルのディバグ・タイプ用ヘッダ・ファイル
	├ MX78K4SS.H	スモール・モデルのスリム・タイプ用ヘッダ・ファイル
	├ MXK4INC.IDL	idea-L用管理ファイル
	├ MX78K4L.IDM	ラージ・モデル用のインクルード・ファイル（idea-L用）
	├ MX78K4LD.IDM	ラージ・モデルのディバグ・タイプ用ヘッダ・ファイル（idea-L用）
	├ MX78K4LS.IDM	ラージ・モデルのスリム・タイプ用ヘッダ・ファイル（idea-L用）
	├ MX78K4S.IDM	スモール・モデル用インクルード・ファイル（idea-L用）
	├ MX78K4SD.IDM	スモール・モデルのディバグ・タイプ用ヘッダ・ファイル（idea-L用）
	├ MX78K4SS.IDM	スモール・モデルのスリム・タイプ用ヘッダ・ファイル（idea-L用）
	├ MX78K4L.INC	ラージ・モデル用のインクルード・ファイル
	└ MX78K4S.INC	スモール・モデル用のインクルード・ファイル
├ LIB78K4		
	├ MX78K4LD.LIB	ラージ・モデルのディバグ・タイプ用ライブラリ
	├ MX78K4LS.LIB	ラージ・モデルのスリム・タイプ用ライブラリ
	├ MX78K4SD.LIB	スモール・モデルのディバグ・タイプ用ライブラリ
	└ MX78K4SS.LIB	スモール・モデルのスリム・タイプ用ライブラリ
└ SMP78K4		
	├ MX78K4RL.ASM	ラージ・モデル用スタートアップ・ルーチン（ソース）
	├ MX78K4RS.ASM	スモール・モデル用スタートアップ・ルーチン（ソース）
	├ MX78K4R.IDL	idea-L用管理ファイル
	├ MX78K4RL.IDM	ラージ・モデル用スタートアップ・ルーチン（idea-L用）
	├ MX78K4RS.IDM	スモール・モデル用スタートアップ・ルーチン（idea-L用）
	├ MX78K4L.PLK	ラージ・モデル用リンク・パラメータ・ファイル
	└ MX78K4S.PLK	スモール・モデル用リンク・パラメータ・ファイル

保守／廃止

1.BIN ディレクトリ

メイク用の各種バッチ・ファイルが入っています。

ファイル名：MX78K4IL.LIB
内容：ラージ・モデルの idea-L 用ライブラリ
使用方法：「idea-L スクリーン・エディタ 入門編」を参照してください。

ファイル名：MX78K4IS.LIB
内容：スモール・モデルの idea-L 用ライブラリ
使用方法：「idea-L スクリーン・エディタ 入門編」を参照してください。

ファイル名：MXK4OMLD.BAT
内容：ラージ・モデルのディバグ・タイプのロードモジュールを生成する際に使用するバッチ・ファイル
使用方法："mx78k4ld.lib"とその他の必要なファイルをリンクし、ロードモジュールを作成します。
このバッチ・ファイルを起動する際には、引数としてチップ種別を渡してください。
例） MXK4OMLD△4026（ターゲット CPU が μ PD784026）

ファイル名：MXK4OMLS.BAT
内容：ラージ・モデルのスリム・タイプのロードモジュールを生成する際に使用するバッチ・ファイル
使用方法："mx78k4ls.lib"とその他の必要なファイルをリンクし、ロードモジュールを作成します。
このバッチ・ファイルを起動する際には、引数としてチップ種別を渡してください。
例） MXK4OMLS△4026（ターゲット CPU が μ PD784026）

ファイル名：MXK4OMSD.BAT
内容：スモール・モデルのディバグ・タイプのロードモジュールを生成する際に使用するバッチ・ファイル
使用方法："mx78k4sd.lib"とその他の必要なファイルをリンクし、ロードモジュールを作成します。
このバッチ・ファイルを起動する際には、引数としてチップ種別を渡してください。
例） MXK4OMSD△4026（ターゲット CPU が μ PD784026）

ファイル名：MXK4OMSS.BAT
内容：スモール・モデルのスリム・タイプのロードモジュールを生成する際に使用するバッチ・ファイル
使用方法："mx78k4ss.lib"とその他の必要なファイルをリンクし、ロードモジュールを作成します。
このバッチ・ファイルを起動する際には、引数としてチップ種別を渡してください。
例） MXK4OMSS△4026（ターゲット CPU が μ PD784026）

ファイル名：MXK4ASL.BAT

内容：ラージ・モデル用にユーザ・タスク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。その際、アセンブル・オプションとして“-nca”を削除すると正常にアセンブル/リンクできない場合があります。このバッチ・ファイルを起動する際には、引数としてチップ種別とファイル名（拡張子は省略）を渡してください。

例）MXK4ASL△4026△Task（ターゲットCPUがμPD784026 ファイル名がTask.asm）

ファイル名：MXK4ASS.BAT

内容：スモール・モデル用にユーザ・タスク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。その際、アセンブル・オプションとして“-nca”を削除すると正常にアセンブル/リンクできない場合があります。このバッチ・ファイルを起動する際には、引数としてチップ種別とファイル名（拡張子は省略）を渡してください。

例）MXK4ASS△4026△Task（ターゲットCPUがμPD784026 ファイル名がTask.asm）

2.DOC ディレクトリ

ユーザーズ・マニュアルに書か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：MX78K4L.TXT

内容：ラージ・モデル用のテキスト・ファイルです。

ファイル名：MX78K4S.TXT

内容：スモール・モデル用のテキスト・ファイルです。

3.INC78K4 ディレクトリ

アセンブリ言語およびC言語用のインクルード・ファイルとヘッダ・ファイルが入っています。

ファイル名：MX78K4LD.H，MX78K4LD.IDM(idea-L用)

内容：ラージ・モデルのディバグ・タイプ用define定義ファイル(C言語用)

使用方法：C 言語でソースを記述する際に必要となるエラー・コード等を define 定義していたり、システムコールの型宣言が定義されていますので、本ファイルをインクルードして使用してください。

ファイル名：MX78K4LS.H，MX78K4LS.IDM(idea-L用)

内容：ラージ・モデルのスリム・タイプ用define定義ファイル(C言語用)

使用方法：C 言語でソースを記述する際に必要となるエラー・コード等を define 定義していたり、システムコールの型宣言が定義されていますので、本ファイルをインクルードして使用してください。

ファイル名：MX78K4SD.H，MX78K4SD.IDM(idea-L 用)
内容：スモール・モデルのディバグ・タイプ用 define 定義ファイル(C 言語用)
使用方法：C 言語でソースを記述する際に必要となるエラー・コード等を define 定義していたり、システムコールの型宣言が定義されていますので、本ファイルをインクルードして使用してください。

ファイル名：MX78K4SS.H，MX78K4SS.IDM(idea-L 用)
内容：スモール・モデルのスリム・タイプ用 define 定義ファイル(C 言語用)
使用方法：C 言語でソースを記述する際に必要となるエラー・コード等を define 定義していたり、システムコールの型宣言が定義されていますので、本ファイルをインクルードして使用してください。

ファイル名：MX78K4L.IDM(idea-L 用)，MX78K4L.INC
内容：ラージ・モデル用 define 定義ファイル(アセンブラ用)
使用方法：C 言語でソースを記述する際に必要となるエラー・コード等を define 定義していたり、システムコールの型宣言が定義されていますので、本ファイルをインクルードして使用してください。

ファイル名：MX78K4S.IDM(idea-L 用)，MX78K4S.INC
内容：スモール・モデル用 define 定義ファイル(アセンブラ用)
使用方法：C 言語でソースを記述する際に必要となるエラー・コード等を define 定義していたり、システムコールの型宣言が定義されていますので、本ファイルをインクルードして使用してください。

ファイル名：MXK4INC.IDL(idea-L 用)
内容：同一ディレクトリにある idea-L 用ファイル（拡張子.IDM）を idea-L が管理するための管理ファイル

4.LIB78K4 ディレクトリ

ライブラリに関するファイルが入っています。

ファイル名：MX78K4LD.LIB
内容：ラージ・モデルのディバグ・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。

ファイル名：MX78K4LS.LIB
内容：ラージ・モデルのスリム・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：MX78K4SD.LIB
内容：スモール・モデルのディバグ・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。

ファイル名：MX78K4SS.LIB
内容：スモール・モデルのスリム・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。

5.SMP78K4 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：MX78K4RL.ASM，MX78K4RL.IDM(idea-L 用)
内容：ラージ・モデル用のスタートアップ・ルーチンのサンプル・ファイル
使用方法：MX78K4 を初期化するルーチンです。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：MX78K4RS.ASM，MX78K4RS.IDM(idea-L 用)
内容：スモール・モデル用のスタートアップ・ルーチンのサンプル・ファイル
使用方法：MX78K4 を初期化するルーチンです。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：MX78K4L.PLK，MX78K4S.PLK
内容：リンク用パラメータ・ファイル
使用方法：リンカに渡すパラメータが入っています。このファイルに、ユーザが作成したファイルを追加すると、MX78K4 と一緒にリンクされます。ただし、本ファイルに記述してある内容を削除した場合は、正常にリンクできない場合があります。

ファイル名：MX78K4R.IDL(idea-L 用)
内容：同一ディレクトリにある idea-L 用ファイル（拡張子.IDM）を idea-L が管理するための管理ファイル

提供ファイル一覧（ソース版）

- NECTOOLS
  - BIN
    - MXK4LALD.BAT　ラージ・モデルのディバグ・タイプのライブラリ生成補助バッチ・ファイル
    - MXK4LALS.BAT　ラージ・モデルのスリム・タイプのライブラリ生成補助バッチ・ファイル
    - MXK4LASD.BAT　スモール・モデルのディバグ・タイプのライブラリ生成補助バッチ・ファイル
    - MXK4LASS.BAT　スモール・モデルのスリム・タイプのライブラリ生成補助バッチ・ファイル
    - MXK4MKL.BAT　ラージ・モデル用のライブラリ生成用バッチ・ファイル
    - MXK4MKS.BAT　スモール・モデル用のライブラリ生成用バッチ・ファイル
  - INC78K4
    - MX78K4LD.H　ラージ・モデルのディバグ・タイプ用ヘッダ・ファイル
    - MX78K4LS.H　ラージ・モデルのスリム・タイプ用ヘッダ・ファイル
    - MX78K4SD.H　スモール・モデルのディバグ・タイプ用ヘッダ・ファイル
    - MX78K4SS.H　スモール・モデルのスリム・タイプ用ヘッダ・ファイル
    - MXK4INC.IDL　idea-L用管理ファイル
    - MX78K4L.IDM　ラージ・モデル用のインクルード・ファイル（idea-L用）
    - MX78K4LD.IDM　ラージ・モデルのディバグ・タイプ用ヘッダ・ファイル（idea-L用）
    - MX78K4LS.IDM　ラージ・モデルのスリム・タイプ用ヘッダ・ファイル（idea-L用）
    - MX78K4S.IDM　スモール・モデル用インクルード・ファイル（idea-L用）
    - MX78K4SD.IDM　スモール・モデルのディバグ・タイプ用ヘッダ・ファイル（idea-L用）
    - MX78K4SS.IDM　スモール・モデルのスリム・タイプ用ヘッダ・ファイル（idea-L用）
    - MX78K4L.INC　ラージ・モデル用のインクルード・ファイル
    - MX78K4S.INC　スモール・モデル用のインクルード・ファイル
  - SMP78K4
    - MX78K4RL.ASM　ラージ・モデル用スタートアップ・ルーチン（ソース）
    - MX78K4RS.ASM　スモール・モデル用スタートアップ・ルーチン（ソース）
    - MX78K4R.IDL　idea-L用管理ファイル
    - MX78K4RL.IDM　ラージ・モデル用スタートアップ・ルーチン（idea-L用）
    - MX78K4RS.IDM　スモール・モデル用スタートアップ・ルーチン（idea-L用）
    - MX78K4L.PLK　ラージ・モデル用リンク・パラメータ・ファイル
    - MX78K4S.PLK　スモール・モデル用リンク・パラメータ・ファイル
  - SRC
    - MX78K4
      - LARGE_C
        - EXTTSK.ASM　ライブラリ・ソース（共通部）(ext_tsk)
        - MXINT.ASM　〃　(タイマ処理ルーチン)
        - RQCLR.ASM　〃　(レディ・キューの初期化ルーチン)
        - TRQCLR.ASM　〃　(レディ・キューとタイマ・キューの初期化ルーチン)
        - MX78K4LC.IDL　idea-L用管理ファイル
        - EXTTSK.IDM　ライブラリ・ソース（共通部）(ext_tsk)idea-L用
        - MXINT.IDM　〃　(タイマ処理ルーチン)idea-L用
        - RQCLR.IDM　〃　(レディ・キューの初期化ルーチン)idea-L用
        - TRQCLR.IDM　〃　(レディ・キューとタイマ・キューの初期化ルーチン)idea-L用
      - LARGE_D
        - CHGPRI.ASM　ライブラリ・ソース（ディバグ･タイプ用）(chg_pri)
        - CHGTSKT.ASM　〃　(chg_tskt)
        - CHGTSKTE.ASM　〃　(chg_tskte)
        - MXFLAG.ASM　〃　(set_flg,clr_flg,rpl_flg)
        - ROTRDQ.ASM　〃　(rot_rdq)
        - ROTRDQ1.ASM　〃　(rot_rdq1)
        - ROTRDQ1E.ASM　〃　(rot_rdq1e)
        - ROTRDQ2.ASM　〃　(rot_rdq2)
        - ROTRDQ2E.ASM　〃　(rot_rdq2e)
        - ROTRDQ3.ASM　〃　(rot_rdq3)
        - ROTRDQ3E.ASM　〃　(rot_rdq3e)
        - ROTRDQE.ASM　〃　(rot_rdqe)
        - STATSK.ASM　〃　(sta_tsk)

```
        ├─STATSK1.ASM ライブラリ・ソース（ディバグ・タイプ用）(sta_tsk1)
        ├─STATSK1E.ASM          〃                          (sta_tsk1e)
        ├─STATSK2.ASM           〃                          (sta_tsk2)
        ├─STATSK2E.ASM          〃                          (sta_tsk2e)
        ├─STATSK3.ASM           〃                          (sta_tsk3)
        ├─STATSK3E.ASM          〃                          (sta_tsk3e)
        ├─STATSKE.ASM           〃                          (sta_tske)
        ├─STATSKP.ASM           〃                          (sta_tskp)
        ├─STATSKPE.ASM          〃                          (sta_tskpe)
        ├─STATSKT.ASM           〃                          (sta_tskt)
        ├─STATSKTE.ASM          〃                          (sta_tskte)
        ├─TERTSK.ASM            〃                          (ter_tsk)
        ├─TERTSKF.ASM           〃                          (ter_tskf)
        ├─TERTSKT.ASM           〃                          (ter_tskt)
        ├─TSKSTS.ASM            〃                          (tsk_sts)
        ├─MX78K4LD.IDL  idea-L用管理ファイル
        ├─CHGPRI.IDM  ライブラリ・ソース（ディバグ・タイプ用）(chg_pri)idea-L用
        ├─CHGTSKT.IDM           〃                          (chg_tskt)idea-L用
        ├─CHGTSKTE.IDM          〃                          (chg_tskte)idea-L用
        ├─MXFLAG.IDM            〃              (set_flg,clr_flg,rpl_flg)idea-L用
        ├─ROTRDQ.IDM            〃                          (rot_rdq)idea-L用
        ├─ROTRDQ1.IDM           〃                          (rot_rdq1)idea-L用
        ├─ROTRDQ1E.IDM          〃                          (rot_rdq1e)idea-L用
        ├─ROTRDQ2.IDM           〃                          (rot_rdq2)idea-L用
        ├─ROTRDQ2E.IDM          〃                          (rot_rdq2e)idea-L用
        ├─ROTRDQ3.IDM           〃                          (rot_rdq3)idea-L用
        ├─ROTRDQ3E.IDM          〃                          (rot_rdq3e)idea-L用
        ├─ROTRDQE.IDM           〃                          (rot_rdqe)idea-L用
        ├─STATSK.IDM            〃                          (sta_tsk)idea-L用
        ├─STATSK1.IDM           〃                          (sta_tsk1)idea-L用
        ├─STATSK1E.IDM          〃                          (sta_tsk1e)idea-L用
        ├─STATSK2.IDM           〃                          (sta_tsk2)idea-L用
        ├─STATSK2E.IDM          〃                          (sta_tsk2e)idea-L用
        ├─STATSK3.IDM           〃                          (sta_tsk3)idea-L用
        ├─STATSK3E.IDM          〃                          (sta_tsk3e)idea-L用
        ├─STATSKE.IDM           〃                          (sta_tske)idea-L用
        ├─STATSKP.IDM           〃                          (sta_tskp)idea-L用
        ├─STATSKPE.IDM          〃                          (sta_tskpe)idea-L用
        ├─STATSKT.IDM           〃                          (sta_tskt)idea-L用
        ├─STATSKTE.IDM          〃                          (sta_tskte)idea-L用
        ├─TERTSK.IDM            〃                          (ter_tsk)idea-L用
        ├─TERTSKF.IDM           〃                          (ter_tskf)idea-L用
        ├─TERTSKT.IDM           〃                          (ter_tskt)idea-L用
        └─TSKSTS.IDM            〃                          (tsk_sts)idea-L用

├─LARGE_S─┬─CHGPRI.ASM   ライブラリ・ソース（スリム・タイプ用）  (chg_pri)
        ├─CHGTSKT.ASM           〃                          (chg_tskt)
        ├─CHGTSKTE.ASM          〃                          (chg_tskte)
        ├─MXFLAG.ASM            〃                (set_flg,clr_flg,rpl_flg)
        ├─ROTRDQ.ASM            〃                          (rot_rdq)
        ├─ROTRDQ1.ASM           〃                          (rot_rdq1)
        ├─ROTRDQ1E.ASM          〃                          (rot_rdq1e)
        ├─ROTRDQ2.ASM           〃                          (rot_rdq2)
        ├─ROTRDQ2E.ASM          〃                          (rot_rdq2e)
        ├─ROTRDQ3.ASM           〃                          (rot_rdq3)
        ├─ROTRDQ3E.ASM          〃                          (rot_rdq3e)
        │
```

```
        ├ROTRDQE.ASM  ライブラリ・ソース（スリム･タイプ用）  (rot_rdqe)
        ├STATSK.ASM          〃                          (sta_tsk)
        ├STATSK1.ASM         〃                          (sta_tsk1)
        ├STATSK1E.ASM        〃                          (sta_tsk1e)
        ├STATSK2.ASM         〃                          (sta_tsk2)
        ├STATSK2E.ASM        〃                          (sta_tsk2e)
        ├STATSK3.ASM         〃                          (sta_tsk3)
        ├STATSK3E.ASM        〃                          (sta_tsk3e)
        ├STATSKE.ASM         〃                          (sta_tske)
        ├STATSKP.ASM         〃                          (sta_tskp)
        ├STATSKPE.ASM        〃                          (sta_tskpe)
        ├STATSKT.ASM         〃                          (sta_tskt)
        ├STATSKTE.ASM        〃                          (sta_tskte)
        ├TERTSK.ASM          〃                          (ter_tsk)
        ├TERTSKF.ASM         〃                          (ter_tskf)
        ├TERTSKT.ASM         〃                          (ter_tskt)
        ├TSKSTS.ASM          〃                          (tsk_sts)
        ├MX78K4LS.IDL  idea-L用管理ファイル
        ├CHGPRI.IDM  ライブラリ・ソース（スリム･タイプ用）  (chg_pri)idea-L用
        ├CHGTSKT.IDM         〃                          (chg_tskt)idea-L用
        ├CHGTSKTE.IDM        〃                          (chg_tskte)idea-L用
        ├MXFLAG.IDM          〃          (set_flg,clr_flg,rpl_flg)idea-L用
        ├ROTRDQ.IDM          〃                          (rot_rdq)idea-L用
        ├ROTRDQ1.IDM         〃                          (rot_rdq1)idea-L用
        ├ROTRDQ1E.IDM        〃                          (rot_rdq1e)idea-L用
        ├ROTRDQ2.IDM         〃                          (rot_rdq2)idea-L用
        ├ROTRDQ2E.IDM        〃                          (rot_rdq2e)idea-L用
        ├ROTRDQ3.IDM         〃                          (rot_rdq3)idea-L用
        ├ROTRDQ3E.IDM        〃                          (rot_rdq3e)idea-L用
        ├ROTRDQE.IDM         〃                          (rot_rdqe)idea-L用
        ├STATSK.IDM          〃                          (sta_tsk)idea-L用
        ├STATSK1.IDM         〃                          (sta_tsk1)idea-L用
        ├STATSK1E.IDM        〃                          (sta_tsk1e)idea-L用
        ├STATSK2.IDM         〃                          (sta_tsk2)idea-L用
        ├STATSK2E.IDM        〃                          (sta_tsk2e)idea-L用
        ├STATSK3.IDM         〃                          (sta_tsk3)idea-L用
        ├STATSK3E.IDM        〃                          (rot_rdq3e)idea-L用
        ├STATSKE.IDM         〃                          (sta_tske)idea-L用
        ├STATSKP.IDM         〃                          (sta_tskp)idea-L用
        ├STATSKPE.IDM        〃                          (sta_tskpe)idea-L用
        ├STATSKT.IDM         〃                          (sta_tskt)idea-L用
        ├STATSKTE.IDM        〃                          (sta_tskte)idea-L用
        ├TERTSK.IDM          〃                          (ter_tsk)idea-L用
        ├TERTSKF.IDM         〃                          (ter_tskf)idea-L用
        ├TERTSKT.IDM         〃                          (ter_tskt)idea-L用
        └TSKSTS.IDM          〃                          (tsk_sts)idea-L用

├SMALL_C┬EXTTSK.ASM  ライブラリ・ソース（共通部）(ext_tsk)
        ├MXINT.ASM           〃                  (タイマ処理ルーチン)
        ├RQCLR.ASM           〃            (レディ・キューの初期化ルーチン)
        ├TRQCLR.ASM          〃  (レディ・キューとタイマ・キューの初期化ルーチン)
        ├MX78K4SC.IDL idea-L用管理ファイル
        ├EXTTSK.IDM  ライブラリ・ソース（共通部）(ext_tsk)idea-L用
        ├MXINT.IDM           〃                  (タイマ処理ルーチン)idea-L用
        ├RQCLR.IDM           〃        (レディ・キューの初期化ルーチン)idea-L用
        └TRQCLR.IDM          〃  (レディ・キューとタイマ・キューの初期化ルーチン)idea-L用
```

```
SMALL_D
  CHGPRI.ASM    ライブラリ・ソース（ディバグ・タイプ用）(chg_pri)
  CHGTSKT.ASM       〃                                  (chg_tskt)
  CHGTSKTE.ASM      〃                                  (chg_tskte)
  MXFLAG.ASM        〃                                  (set_flg,clr_flg,rpl_flg)
  ROTRDQ.ASM        〃                                  (rot_rdq)
  ROTRDQ1.ASM       〃                                  (rot_rdq1)
  ROTRDQ1E.ASM      〃                                  (rot_rdq1e)
  ROTRDQ2.ASM       〃                                  (rot_rdq2)
  ROTRDQ2E.ASM      〃                                  (rot_rdq2e)
  ROTRDQ3.ASM       〃                                  (rot_rdq3)
  ROTRDQ3E.ASM      〃                                  (rot_rdq3e)
  ROTRDQE.ASM       〃                                  (rot_rdqe)
  STATSK.ASM        〃                                  (sta_tsk)
  STATSK1.ASM       〃                                  (sta_tsk1)
  STATSK1E.ASM      〃                                  (sta_tsk1e)
  STATSK2.ASM       〃                                  (sta_tsk2)
  STATSK2E.ASM      〃                                  (sta_tsk2e)
  STATSK3.ASM       〃                                  (sta_tsk3)
  STATSK3E.ASM      〃                                  (sta_tsk3e)
  STATSKE.ASM       〃                                  (sta_tske)
  STATSKP.ASM       〃                                  (sta_tskp)
  STATSKPE.ASM      〃                                  (sta_tskpe)
  STATSKT.ASM       〃                                  (sta_tskt)
  STATSKTE.ASM      〃                                  (sta_tskte)
  TERTSK.ASM        〃                                  (ter_tsk)
  TERTSKF.ASM       〃                                  (ter_tskf)
  TERTSKT.ASM       〃                                  (ter_tskt)
  TSKSTS.ASM        〃                                  (tsk_sts)
  MX78K4SD.IDL  idea-L用管理ファイル
  CHGPRI.IDM    ライブラリ・ソース（ディバグ・タイプ用）(chg_pri)idea-L用
  CHGTSKT.IDM       〃                                  (chg_tskt)idea-L用
  CHGTSKTE.IDM      〃                                  (chg_tskte)idea-L用
  MXFLAG.IDM        〃                                  (set_flg,clr_flg,rpl_flg)idea-L用
  ROTRDQ.IDM        〃                                  (rot_rdq)idea-L用
  ROTRDQ1.IDM       〃                                  (rot_rdq1)idea-L用
  ROTRDQ1E.IDM      〃                                  (rot_rdq1e)idea-L用
  ROTRDQ2.IDM       〃                                  (rot_rdq2)idea-L用
  ROTRDQ2E.IDM      〃                                  (rot_rdq2e)idea-L用
  ROTRDQ3.IDM       〃                                  (rot_rdq3)idea-L用
  ROTRDQ3E.IDM      〃                                  (rot_rdq3e)idea-L用
  ROTRDQE.IDM       〃                                  (rot_rdqe)idea-L用
  STATSK.IDM        〃                                  (sta_tsk)idea-L用
  STATSK1.IDM       〃                                  (sta_tsk1)idea-L用
  STATSK1E.IDM      〃                                  (sta_tsk1e)idea-L用
  STATSK2.IDM       〃                                  (sta_tsk2)idea-L用
  STATSK2E.IDM      〃                                  (sta_tsk2e)idea-L用
  STATSK3.IDM       〃                                  (sta_tsk3)idea-L用
  STATSK3E.IDM      〃                                  (sta_tsk3e)idea-L用
  STATSKE.IDM       〃                                  (sta_tske)idea-L用
  STATSKP.IDM       〃                                  (sta_tskp)idea-L用
  STATSKPE.IDM      〃                                  (sta_tskpe)idea-L用
  STATSKT.IDM       〃                                  (sta_tskt)idea-L用
  STATSKTE.IDM      〃                                  (sta_tskte)idea-L用
  TERTSK.IDM        〃                                  (ter_tsk)idea-L用
  TERTSKF.IDM       〃                                  (ter_tskf)idea-L用
  TERTSKT.IDM       〃                                  (ter_tskt)idea-L用
  TSKSTS.IDM        〃                                  (tsk_sts)idea-L用
```

保守／廃止

```
└SMALL_S ┬ CHGPRI.ASM    ライブラリ・ソース（スリム・タイプ用）   (chg_pri)
         ├ CHGTSKT.ASM          〃                              (chg_tskt)
         ├ CHGTSKTE.ASM         〃                              (chg_tskte)
         ├ MXFLAG.ASM           〃                   (set_flg,clr_flg,rpl_flg)
         ├ ROTRDQ.ASM           〃                              (rot_rdq)
         ├ ROTRDQ1.ASM          〃                              (rot_rdq1)
         ├ ROTRDQ1E.ASM         〃                              (rot_rdq1e)
         ├ ROTRDQ2.ASM          〃                              (rot_rdq2)
         ├ ROTRDQ2E.ASM         〃                              (rot_rdq2e)
         ├ ROTRDQ3.ASM          〃                              (rot_rdq3)
         ├ ROTRDQ3E.ASM         〃                              (rot_rdq3e)
         ├ ROTRDQE.ASM          〃                              (rot_rdqe)
         ├ STATSK.ASM           〃                              (sta_tsk)
         ├ STATSK1.ASM          〃                              (sta_tsk1)
         ├ STATSK1E.ASM         〃                              (sta_tsk1e)
         ├ STATSK2.ASM          〃                              (sta_tsk2)
         ├ STATSK2E.ASM         〃                              (sta_tsk2e)
         ├ STATSK3.ASM          〃                              (sta_tsk3)
         ├ STATSK3E.ASM         〃                              (sta_tsk3e)
         ├ STATSKE.ASM          〃                              (sta_tske)
         ├ STATSKP.ASM          〃                              (sta_tskp)
         ├ STATSKPE.ASM         〃                              (sta_tskpe)
         ├ STATSKT.ASM          〃                              (sta_tskt)
         ├ STATSKTE.ASM         〃                              (sta_tskte)
         ├ TERTSK.ASM           〃                              (ter_tsk)
         ├ TERTSKF.ASM          〃                              (ter_tskf)
         ├ TERTSKT.ASM          〃                              (ter_tskt)
         ├ TSKSTS.ASM           〃                              (tsk_sts)
         ├ MX78K4SS.IDL  idea-L用管理ファイル
         ├ CHGPRI.IDM    ライブラリ・ソース（スリム・タイプ用）   (chg_pri)idea-L用
         ├ CHGTSKT.IDM          〃                              (chg_tskt)idea-L用
         ├ CHGTSKTE.IDM         〃                              (chg_tskte)idea-L用
         ├ MXFLAG.IDM           〃                   (set_flg,clr_flg,rpl_flg)idea-L用
         ├ ROTRDQ.IDM           〃                              (rot_rdq)idea-L用
         ├ ROTRDQ1.IDM          〃                              (rot_rdq1)idea-L用
         ├ ROTRDQ1E.IDM         〃                              (rot_rdq1e)idea-L用
         ├ ROTRDQ2.IDM          〃                              (rot_rdq2)idea-L用
         ├ ROTRDQ2E.IDM         〃                              (rot_rdq2e)idea-L用
         ├ ROTRDQ3.IDM          〃                              (rot_rdq3)idea-L用
         ├ ROTRDQ3E.IDM         〃                              (rot_rdq3e)idea-L用
         ├ ROTRDQE.IDM          〃                              (rot_rdqe)idea-L用
         ├ STATSK.IDM           〃                              (sta_tsk)idea-L用
         ├ STATSK1.IDM          〃                              (sta_tsk1)idea-L用
         ├ STATSK1E.IDM         〃                              (sta_tsk1e)idea-L用
         ├ STATSK2.IDM          〃                              (sta_tsk2)idea-L用
         ├ STATSK2E.IDM         〃                              (sta_tsk2e)idea-L用
         ├ STATSK3.IDM          〃                              (sta_tsk3)idea-L用
         ├ STATSK3E.IDM         〃                              (rot_rdq3e)idea-L用
         ├ STATSKE.IDM          〃                              (sta_tske)idea-L用
         ├ STATSKP.IDM          〃                              (sta_tskp)idea-L用
         ├ STATSKPE.IDM         〃                              (sta_tskpe)idea-L用
         ├ STATSKT.IDM          〃                              (sta_tskt)idea-L用
         ├ STATSKTE.IDM         〃                              (sta_tskte)idea-L用
         ├ TERTSK.IDM           〃                              (ter_tsk)idea-L用
         ├ TERTSKF.IDM          〃                              (ter_tskf)idea-L用
         ├ TERTSKT.IDM          〃                              (ter_tskt)idea-L用
         └ TSKSTS.IDM           〃                              (tsk_sts)idea-L用
```

1.BIN ディレクトリ

メイク用の各種バッチ･ファイルが入っています。

ファイル名：MXK4LALD.BAT
内容：“MXK4MKL.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。）

ファイル名：MXK4LALS.BAT
内容：“MXK4MKL.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。）

ファイル名：MXK4LASD.BAT
内容：“MXK4MKS.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。）

ファイル名：MXK4LASS.BAT
内容：“MXK4MKS.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。）

ファイル名：MXK4MKL.BAT
内容：MX78K4 のラージ・モデルのライブラリ再構築するためのバッチファイル
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・ファイルを生成します。このバッチ・ファイルを起動する際には、引数としてチップ種別を渡してください。また、本バッチ・ファイルは、“¥NECTOOLS¥SRC¥MX78K4”のディレクトリで実行してください。
例） MXK4MKL△4026（ターゲット CPU が μ PD784026）

ファイル名：MXK4MKS.BAT
内容：MX78K4 のスモール・モデルのライブラリを再構築するためのバッチ・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・ファイルを生成します。このバッチ・ファイルを起動する際には、引数としてチップ種別を渡してください。また、本バッチ・ファイルは、“¥NECTOOLS¥SRC¥MX78K4”のディレクトリで実行してください。
例） MXK4MKS△4026（ターゲット CPU が μ PD784026）

3.INC78K4 ディレクトリ

アセンブリ言語および C 言語用のインクルード・ファイルとヘッダ・ファイルが入っています。

ファイル名：MX78K4LD.H，MX78K4LD.IDM(idea-L 用)
内容：オブジェクト版と同様
使用方法：オブジェクト版と同様

ファイル名：MX78K4LS.H，MX78K4LS.IDM(idea-L 用)
内容：オブジェクト版と同様
使用方法：オブジェクト版と同様

ファイル名：MX78K4SD.H，MX78K4SD.IDM(idea-L 用)
内容：オブジェクト版と同様
使用方法：オブジェクト版と同様

ファイル名：MX78K4SS.H，MX78K4SS.IDM(idea-L 用)
内容：オブジェクト版と同様
使用方法：オブジェクト版と同様

ファイル名：MX78K4L.IDM(idea-L 用)，MX78K4L.INC
内容：オブジェクト版と同様
使用方法：オブジェクト版と同様

ファイル名：MX78K4S.IDM(idea-L 用)，MX78K4S.INC
内容：オブジェクト版と同様
使用方法：オブジェクト版と同様

ファイル名：MXK4INC.IDL(idea-L 用)
内容：同一ディレクトリにある idea-L 用ファイル（拡張子.IDM）を idea-L が管理するための管理ファイル

5.SMP78K4 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：MX78K4RL.ASM，MX78K4RL.IDM(idea-L 用)
内容：オブジェクト版と同様
使用方法：オブジェクト版と同様

ファイル名：MX78K4RS.ASM，MX78K4RS.IDM(idea-L 用)
内容：オブジェクト版と同様
使用方法：オブジェクト版と同様

ファイル名：MX78K4L.PLK，MX78K4S.PLK
内容：オブジェクト版と同様
使用方法：オブジェクト版と同様

ファイル名：MX78K4R.IDL(idea-L 用)
内容：同一ディレクトリにある idea-L 用ファイル（拡張子.IDM）を idea-L が管理するための管理ファイル

6.LARGE_C ディレクトリ

ラージ・モデルの共通部用のライブラリ・ソースが入っています。

ファイル名：EXTTSK.ASM，EXTTSK.IDM(idea-L 用)
　　内容：ラージ・モデル用“ext_tsk”のソース・ファイル

ファイル名：MXINT.ASM，MXINT.IDM(idea-L 用)
　　内容：ラージ・モデル用のタイマ処理ルーチンのソース・ファイル

ファイル名：RQCLR.ASM，RQCLR.IDM(idea-L 用)
　　内容：レディ・キューを初期化するルーチンのソース・ファイル

ファイル名：TRQCLR.ASM，TRQCLR.IDM(idea-L 用)
　　内容：レディ・キューとタイマキューを初期化するルーチンのソース・ファイル

ファイル名：MX78K4LC.IDL(idea-L 用)
　　内容：同一ディレクトリにある idea-L 用ファイル（拡張子.IDM）を idea-L が管理するための管理ファイル

7.LARGE_D ディレクトリ

ラージ・モデルのディバグ・タイプ用のライブラリ・ソースが入っています。

ファイル名：*.ASM，*.IDM(idea-L 用)
　　内容：MX78K4 のソース・ファイルです。

ファイル名：MX78K4LD.IDL(idea-L 用)
　　内容：同一ディレクトリにある idea-L 用ファイル（拡張子.IDM）を idea-L が管理するための管理ファイル

8.LARGE_S ディレクトリ

ラージ・モデルのスリム・タイプ用のライブラリ・ソースが入っています。

ファイル名：*.ASM，*.IDM(idea-L 用)
　　内容：MX78K4 のソース・ファイルです。

ファイル名：MX78K4LS.IDL(idea-L 用)
　　内容：同一ディレクトリにある idea-L 用ファイル（拡張子.IDM）を idea-L が管理するための管理ファイル

9.SMALL_C ディレクトリ

スモール・モデルの共通部分用のライブラリ・ソースが入っています。

ファイル名：EXTTSK.ASM，EXTTSK.IDM(idea-L 用)
内容：スモール・モデル用“ext_tsk”のソース・ファイル

ファイル名：MXINT.ASM，MXINT.IDM(idea-L 用)
内容：スモール・モデル用のタイマ処理ルーチンのソース・ファイル

ファイル名：RQCLR.ASM，RQCLR.IDM(idea-L 用)
内容：レディ・キューを初期化するルーチンのソース・ファイル

ファイル名：TRQCLR.ASM，TRQCLR.IDM(idea-L 用)
内容：レディ・キューとタイマキューを初期化するルーチンのソース・ファイル

ファイル名：MX78K4SC.IDL(idea-L 用)
内容：同一ディレクトリにある idea-L 用ファイル（拡張子.IDM）を idea-L が管理するための管理ファイル

10.SMALL_D ディレクトリ

スモール・モデルのディバグ・タイプ用のライブラリ・ソースが入っています。

ファイル名：*.ASM，*.IDM(idea-L 用)
内容：MX78K4 のソース・ファイルです。

ファイル名：MX78K4SD.IDL(idea-L 用)
内容：同一ディレクトリにある idea-L 用ファイル（拡張子.IDM）を idea-L が管理するための管理ファイル

8.SMALL_S ディレクトリ

スモール・モデルのスリム・タイプ用のライブラリ・ソースが入っています。

ファイル名：*.ASM，*.IDM(idea-L 用)
内容：MX78K4 のソース・ファイルです。

ファイル名：MX78K4SS.IDL(idea-L 用)
内容：同一ディレクトリにある idea-L 用ファイル（拡張子.IDM）を idea-L が管理するための管理ファイル

1.3 特徴

78K シリーズ用の OS である MX78K4 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。

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）を削除しました。これにより、タスクは ext_tsk() 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
　MX78K4 にはスリム・タイプとディバグ・タイプがあります。スリム・タイプは、システムコール処理時にエラーの検出を行いませんが、ディバグ・タイプは、システムコール処理時にエラーの検出を行います。ユーザーは必要に応じて、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います。

※　μ ITRON は、TRON(The Real-time Operating system Nucleus)プロジェクトの一環で、16/32 ビットのマイコン向け組み込み制御用リアルタイム OS の仕様 ITRON(Industrial TRON)を、8 ビットのマイコン向けにした仕様です。

1.4 機能概要

MX78K4 は、以下の処理を行います。
・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替処理を行います。

MX78K4 には以下のモジュールが存在します。
1．タスク管理
　MX78K4 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。

1.5 実行環境

78K シリーズ用の OS である MX78K4 は組み込み用 OS です。以下に対応する CPU を示します。
・78K/IV シリーズ

保守／廃止

1.6 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-9800 シリーズ（MS-DOS）

2．ソフトウエア
・リロケータブル・アセンブラ・パッケージ
RA78K4（78K/IV 用）
・コンパイラ・パッケージ
CC78K4（78K/IV 用）

※ CC78K4 を使用する場合は、必ずバージョン 2.00 以上を使用してください。

1.7 RX78K シリーズとの違い

MX78K4 と RX78K シリーズでは大きく以下の点が異なります。

・タスクの状態
MX78K4 には WAIT 状態が存在しません。このため、タスクは資源待ち・時限待ちをすることはできません。

・優先順位がない
MX78K4 にはタスクの優先順位がありません。タスクはレディ・キューにつながれている順番に起動されます。ただし、chg_pri()システムコールによってそのタスクの起動される順番を早めることができます。

・タスクの排他制御・同期・通信機能がない
MX78K4 にはタスクの WAIT 状態が存在しないため、排他制御・同期・通信機能が存在しません。

・メモリ管理
MX78K4 ではメモリの管理を一切行いません。

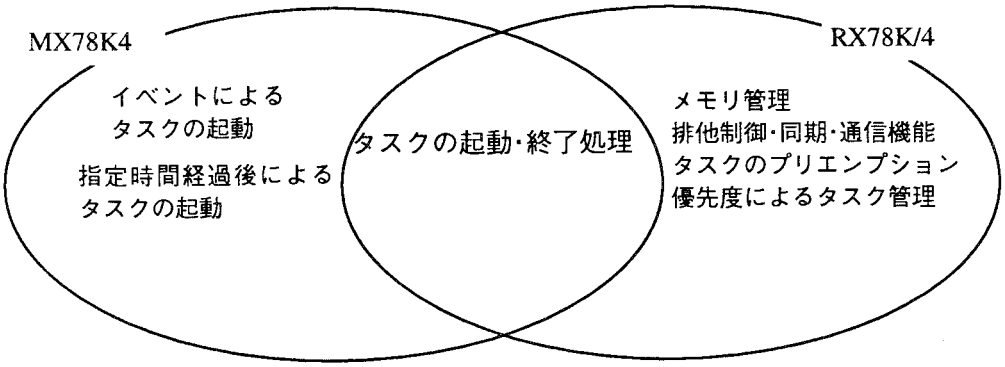


図 1.2 RX78K シリーズとの関係図

第２章　タスク管理

タスクとは、MX78K4 上のアプリケーション・プログラムを構成する要素です。MX78K4 では、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、MX78K4 におけるタスクの管理方法、状態について述べます。
2.1 ではタスクの概要について、2.2 ではタスクの起動について、2.3 ではタスクの状態について、2.4 ではタスクの終了について述べます。

2.1　タスクの概要

タスクは、MX78K4 の管理下で実行される最小単位です。起動、実行、終了は、すべてタスク単位で行われています。
プライオリティに基づいて処理される RX78K シリーズとは異なり、MX78K4 では、レディ・キューにつながれている順番に処理されます。
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()を発行したとき）のみです。

2.2　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、初期化処理中において最初に sta_tsk()をかけられたタスクです。また、初期化処理中において sta_tsk()をかけられたその他のタスクは、システムが立ち上がった時点でレディ・キューにつながれています。レディ・キューとは ready 状態のタスクを OS が管理するために OS 内部に設けている領域のことです。（タスクの状態については、「2.3　タスクの状態」を参照してください。）初期化処理中に sta_tsk()をかけられなかったタスクに関しては、起動されたタスク中で、sta_tsk()等を発行することによって、起動されレディ・キューにつながれます。（初期化処理に関しては、第７章を参照してください。）
タスクの起動は、sta_tsk()の他に、現在動作しているタスクの次に指定タスクが動作するようにレディ・キューにつなぐ sta_tskp()、指定時間経過後に指定タスクの起動を行う sta_tskt()、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 MX78K4 にはあります。

以下それらのシステムコールについて説明します。

2.2.1　sta_tsk()

sta_tsk()は、指定されたタスクをレディ・キューの最後尾につなぐ処理を行います。
ここで指定されたタスクは、そのタスクの実行順序がくるまで、その実行は待たされます。

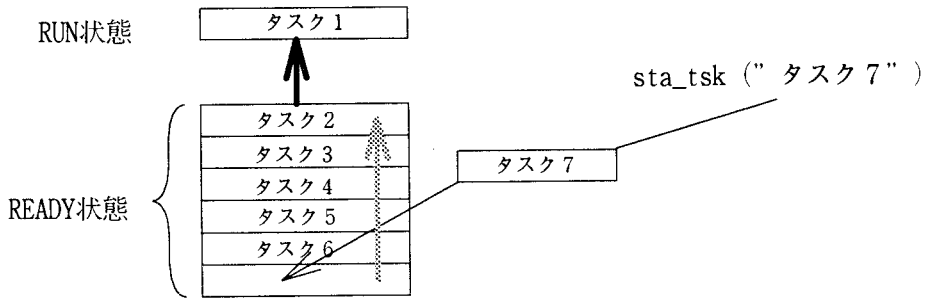


図 2.1　sta_tsk()の説明図

2.2.2 sta_tskp()

sta_tskp()は、指定されたタスクを現在動作しているタスクの次に起動するようにレディ・キューにつなぎます。ここで指定されたタスクは、現在動いているタスクの処理が終了すると動作します。

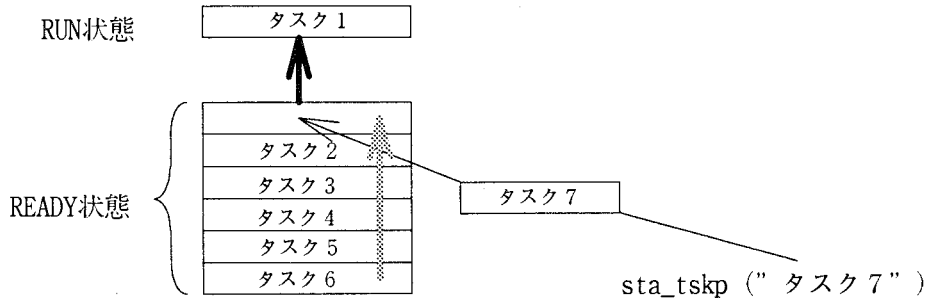


図 2.2 sta_tskp()の説明図

2.2.3 sta_tskt()

sta_tskt()は、指定されたタスクを指定時間後に起床させる処理を行います。指定されたタスクは、いったんタイマ・キューにつながれ指定時間が経過した時点で、そのときに起動しているタスクの次に起動するようにレディ・キューにつながれます。そして指定されたタスクは、現在動いているタスクの処理が終了すると動作します。

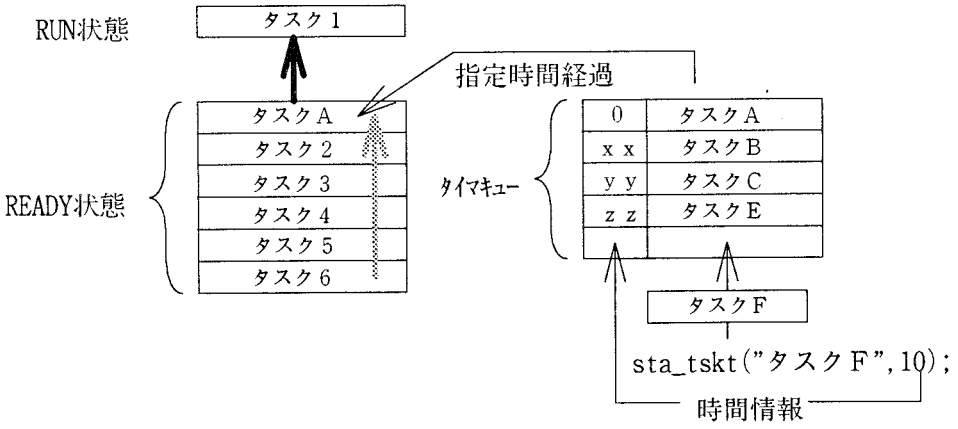


図 2.3 sta_tskt()の説明図

2.2.4 sta_tskf()

sta_tskf()は、指定したビット・パターンと指定したイベントフラグのビット・パターンが一致したときに指定されたタスクを起床させる処理を行います。指定したビット・パターンとフラグのビット・パターンが一致すると、そのときに動作しているタスクの次に起動されるようにレディ・キューにつなぎます。そして、指定されたタスクは、現在動いているタスクの処理が終了すると動作します。タスクが起動された時点で、フラグは、すべてクリアされます。

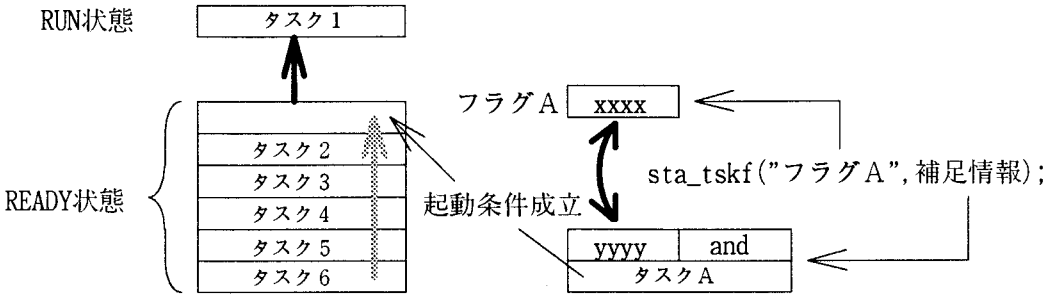


図 2.4 sta_tskf()の説明図

2.3 タスクの状態

MX78K4 におけるタスクの状態は、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ていて、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・キューにつながれている状態で、実行の順番を待っている状態です。

・DORMANT 状態（休止状態）

RUN 状態でも READY 状態でもないタスクです。

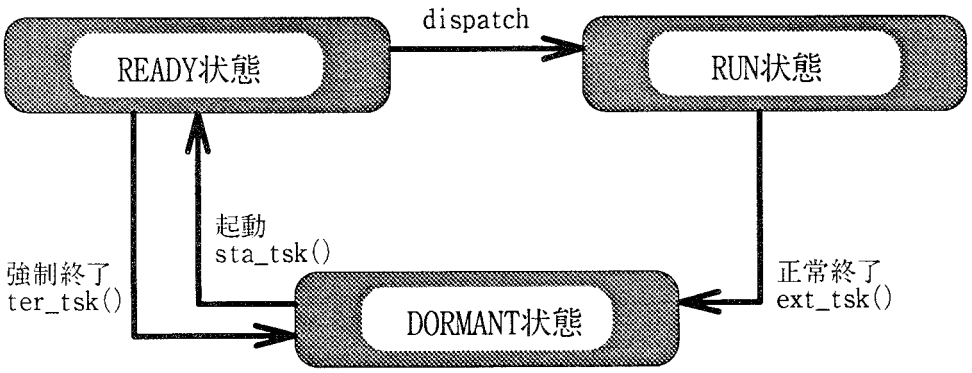


図 2.5　タスク状態遷移図

・RUN 状態のタスクの管理

MX78K4 において RUN 状態になるのは READY 状態のタスクの内、レディ・キューの先頭につながれているタスクです。

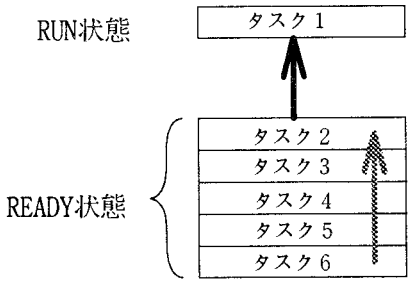


図 2.6　READY 状態のタスクの図

・READY 状態のタスクの管理

MX78K4 では、sta_tsk()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・キューにつながれ管理されます。

・DORMANT 状態のタスクの管理

DORMANT 状態となっているタスクは、MX78K4 によって管理されません。

2.4 タスクの終了

起動したタスクが 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。

ext_tsk: 自タスクを終了します。

sta_tske: 他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()と tsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

sta_tsk1e: 他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

sta_tsk2e: 他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

sta_tsk3e: 他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()と tsk_sts()で使用されたタスクを指定してください。

sta_tskpe: 他タスクを起動し、自タスクを終了します。指定された他タスクは、現在動作しているタスクの次に起動します。

sta_tskte: 指定した時間が経過したら、他タスクを起動し、自タスクを終了します。

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。

sta_tskfe: イベントフラグ操作で起動する他タスクの条件を設定し、自タスクを終了します。

rot_rdqe: レディ・キューを回転させ、終了します。ただし、回転する自タスクは、 ter_tsk()と tsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

rot_rdq1e: レディ・キューを回転させ、終了します。ただし、回転する自タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

rot_rdq2e: レディ・キューを回転させ、終了します。ただし、回転する自タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

rot_rdq3e: レディ・キューを回転させ、終了します。ただし、回転する自タスクは、ter_tsk()と tsk_sts()で使用されたタスクを指定してください。

ter_tsk: レディ・キューにある他タスクを強制終了させます。

ter_tskt: タイマ・キューまたはレディ・キューにある他タスクを強制終了させます。

ter_tskf: イベントフラク操作で起動される他タスクまたはレディ・キューにある他タスクを強制終了させます。

第３章　イベントフラグ

MX78K4 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、システム中に複数設定することが可能ですが、１つのイベントフラグに対して起動を設定することのできるタスクは１つです。ただし、イベントフラグは、メモリ上に連続して配置しなければなりません。
3.1では、イベントフラグのセットについて、3.2では、事象の発生に対応するタスクの登録について、3.3では、事象の発生に対応するタスクの起動について説明します。

3.1　イベントフラグのセット

MX78K4 では、set_flg()によってイベントフラグのセットを行います。
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・パターンを指定するが、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。

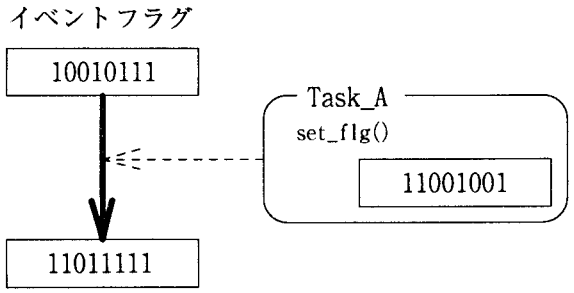


図 3.1　イベントフラグのセット

3.2 事象の発生に対応するタスクの登録

MX78K4 では、sta_tskf()によってイベントの発生に対応するタスクの登録を行います。
sta_tskf()のパラメータには、対象イベントフラグ、起動されるタスク、待ちビット・パターン、タスクの起動条件の 4 つがあります。タスクの起動条件とは、イベントフラグがセットされタスクが起動するための条件で、以下の 2 種類があります。

OR 条件　　指定ビット・パターンの“1”である箇所のうちイベントフラグが 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　指定ビット・パターンの“1”である箇所のうちイベントフラグすべてが“1”になった場合に指定タスクを起動します。

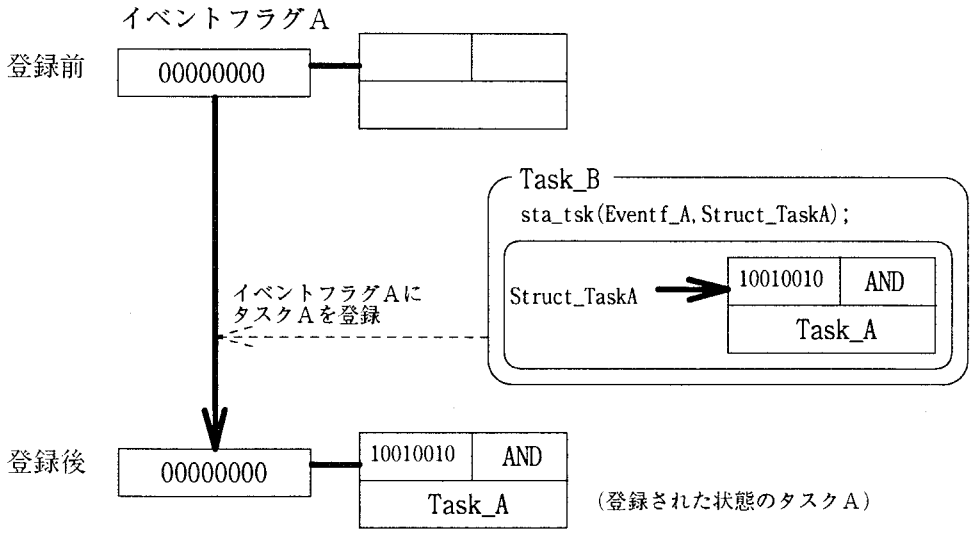


図 3.2　タスクの登録

※注意　　タスクの処理において、事象の発生に対応するタスクを sta_tskf()によって登録し、事象の発生前に同じイベントフラグに対し再び sta_tskf()を発行し、別のタスクを登録する事も可能であるが、このイベントフラグに対する事象の発生によって起動されるタスクは、先に登録したタスクではなく、後に登録しなおしたタスクです。

3.3 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　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 状態へ移します。
このとき、登録されていたタスクは、レディ・キューの先頭につながれ、イベントフラグに登録されているタスクは存在しなくなります。よって、イベントフラグのビット・パターンが再び 0 にセットされても、新たに sta_tskf()によって、そのイベントフラグに対しタスクを登録しない限り、そのイベントフラグによるタスクの起動はしません。

第 4 章　時間管理

この章では、MX78K4 で扱われるタイマについて述べます。
MX78K4 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。

4.1　概要

この節では、MX78K4 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。
MX78K4 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・キューにつなぐ）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。

4.2　遅延起動

MX78K4 では、指定時間経過後に指定タスクを起動する sta_tskt()があります。（sta_tskt()に関する詳細は、第６章システムコールの6.2.1タスク関連システムコールを参照してください。）

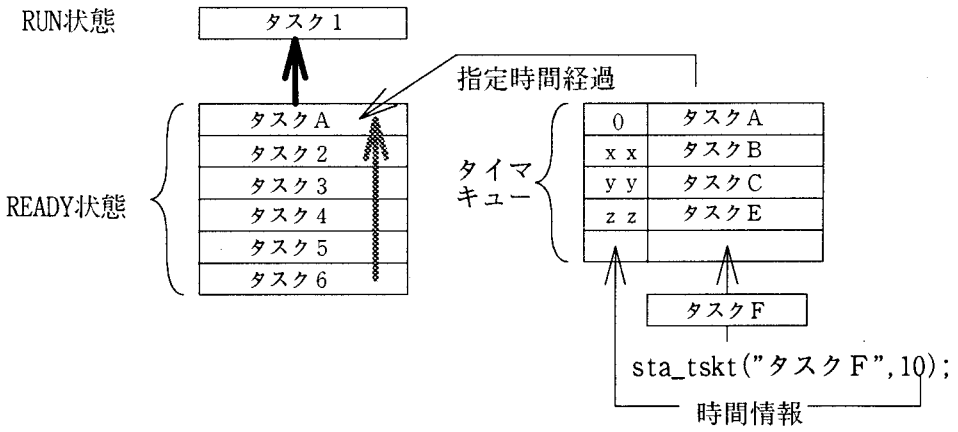


図 4.1　sta_tskt()の説明図

4.3　起動時間の変更

MX78K4 では、起動時間の変更を行う chg_tskt()があります。
まず、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、存在した場合は、そのタスクを強制終了します。次に指定したタスクを指定した時間に設定します。

第５章　割り込み機能

この章では、MX78K4 における割り込み処理について述べます。

5.1　概要

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

MX78K4 は、各割り込みソースのイニシャライズ処理を行わないので、各ソースはユーザにより初期化される必要があります。また、割り込みハンドラからの復帰処理は、ユーザによって記述されなければなりません。

5.2　割り込みハンドラ

5.2.1　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、割り込みハンドラは、タスクや OS よりも優先されます。

5.2.2　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。

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。

よって、MX78K4 では、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。

3．割り込みハンドラの終了

割り込みハンドラは、「reti」を発行することによって終了します。

保守／廃止

5.3 割り込みハンドラから発行可能なシステムコール

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: 指定された他タスクは、現在動作しているタスクの次に起動します。

sta_tskf: イベントフラグ操作で起動する他タスクの条件を設定します。

set_flg: 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。

clr_flg: 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。

rpl_flg: イベントフラグに任意のビット・パターンと入れ替えます。

第６章　システムコール

6.1　概要

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されている OS 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、タスクは、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。

6.1.1　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。

・タスク管理システムコール

タスク管理システムコールは、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:　他タスクを起動します。ただし、この指定タスクは、ter_tsk()と tsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

sta_tsk1:　他タスクを起動します。ただし、この指定タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

sta_tsk2:　他タスクを起動します。ただし、この指定タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

sta_tsk3:　他タスクを起動します。ただし、この指定タスクは、ter_tsk()と tsk_sts()で使用されたタスクを指定してください。

sta_tske:　他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()と tsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

sta_tsk1e:　他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

sta_tsk2e:　他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

sta_tsk3e:　他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()と tsk_sts()で使用されたタスクを指定してください。

sta_tskp:　指定された他タスクは、現在動作しているタスクの次に起動します。

sta_tskpe:　他タスクを起動し、自タスクを終了します。指定された他タスクは、現在動作しているタスクの次に起動します。

sta_tskf:　イベントフラグ操作で起動する他タスクの条件を設定します。

sta_tskt:　指定した時間が経過したら、他タスクを起動します。

sta_tskte:　指定した時間が経過したら、他タスクを起動し、自タスクを終了します。

chg_tskt:　指定した他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_tsk:　レディ・キューにある他タスクを強制終了させます。

ter_tskt:　タイマ・キューまたはレディ・キューにある他タスクを強制終了させます。

ter_tskf:　イベントフラク操作で起動される他タスクまたはレディ・キューにある他タスクを強制終了させます。

rot_rdq:　レディ・キューを回転させます。ただし、回転する自タスクは、 ter_tsk()と tsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

rot_rdq1:　レディ・キューを回転させます。ただし、回転する自タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

rot_rdq2:　レディ・キューを回転させます。ただし、回転する自タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

rot_rdq3:　レディ・キューを回転させます。ただし、回転する自タスクは、ter_tsk()と tsk_sts()で使用されたタスクを指定してください。

rot_rdqe:　レディ・キューを回転させ、終了します。ただし、回転する自タスクは、 ter_tsk()と tsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

rot_rdq1e:　レディ・キューを回転させ、終了します。ただし、回転する自タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

rot_rdq2e:　レディ・キューを回転させ、終了します。。ただし、回転する自タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

rot_rdq3e:　レディ・キューを回転させ、終了します。ただし、回転する自タスクは、ter_tsk()と tsk_sts()で使用されたタスクを指定してください。

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。

保守／廃止

・イベントフラグ管理システムコール

イベントフラグ管理システムコールは、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:　　イベントフラグに任意のビット・パターンをセットします。

clr_flg:　　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。

rpl_flg:　　イベントフラグに任意のビット・パターンと入れ替えます。

6.1.2 パラメータ

各システムコールのパラメータは、以下に示すようなレジスタ、メモリを使用します。

・スモール・モデル

	RP2	AX	DE
sta_tsk	a_tsk	--	--
sta_tsk1	a_tsk	--	--
sta_tsk2	a_tsk	--	--
sta_tsk3	a_tsk	--	--
sta_tske	a_tsk	--	--
sta_tsk1e	a_tsk	--	--
sta_tsk2e	a_tsk	--	--
sta_tsk3e	a_tsk	--	--
sta_tskp	a_tsk	--	--
sta_tskpe	a_tsk	--	--
sta_tskt	a_tsk	tmout	--
sta_tskte	a_tsk	tmout	--
chg_tskt	a_tsk	tmout	--
chg_tskte	a_tsk	tmout	--
ter_tsk	a_tsk	--	--
ter_tskt	a_tsk	--	--
ter_tskf	a_tsk	--	--
rot_rdq	--	--	--
rot_rdq1	--	--	--
rot_rdq2	--	--	--
rot_rdq3	--	--	--
rot_rdqe	--	--	--
rot_rdq1e	--	--	--
rot_rdq2e	--	--	--
rot_rdq3e	--	--	--
tsk_sts	pk_tsks	--	a_tsk
chg_pri	a_tsk	--	--
ext_tsk	--	--	--

	RP2	a_flg	a_flg+1
sta_tskf	--	pk_finfo	--
set_flg	a_flg	--	setptn
clr_flg	--	--	clrptn
rpl_flg	a_flg	--	rplptn

a_tsk: タスクの先頭アドレス（タスク名）
tmout: タイムアウトまでの時間（数値）
pk_tsks: タスクステータスの結果が返される変数のアドレス
a_flg: イベントフラグのアドレス
a_flg+1: イベントフラグのアドレス＋１
pk_finfo: イベントフラグの補足情報が格納しているテーブルのポインタ
setptn: セットされる値（数値）
clrptn: クリアする値（数値）
rplptn: 置き換える値（数値）

・ラージ・モデル

	TDE	AX	WHL,_@NRARG0
sta_tsk	a_tsk	--	--
sta_tsk1	a_tsk	--	--
sta_tsk2	a_tsk	--	--
sta_tsk3	a_tsk	--	--
sta_tske	a_tsk	--	--
sta_tsk1e	a_tsk	--	--
sta_tsk2e	a_tsk	--	--
sta_tsk3e	a_tsk	--	--
sta_tskp	a_tsk	--	--
sta_tskpe	a_tsk	--	--
sta_tskt	a_tsk	tmout	--
sta_tskte	a_tsk	tmout	--
chg_tskt	a_tsk	tmout	--
chg_tskte	a_tsk	tmout	--
ter_tsk	a_tsk	--	--
ter_tskt	a_tsk	--	--
ter_tskf	a_tsk	--	--
rot_rdq	--	--	--
rot_rdq1	--	--	--
rot_rdq2	--	--	--
rot_rdq3	--	--	--
rot_rdqe	--	--	--
rot_rdq1e	--	--	--
rot_rdq2e	--	--	--
rot_rdq3e	--	--	--
tsk_sts	pk_tsks	--	pk_tsks
chg_pri	a_tsk	--	--
ext_tsk	--	--	--

	TDE	a_flg	a_flg+1
sta_tskf	--	pk_finfo	--
set_flg	a_flg	--	setptn
clr_flg	--	--	clrptn
rpl_flg	a_flg	--	rplptn

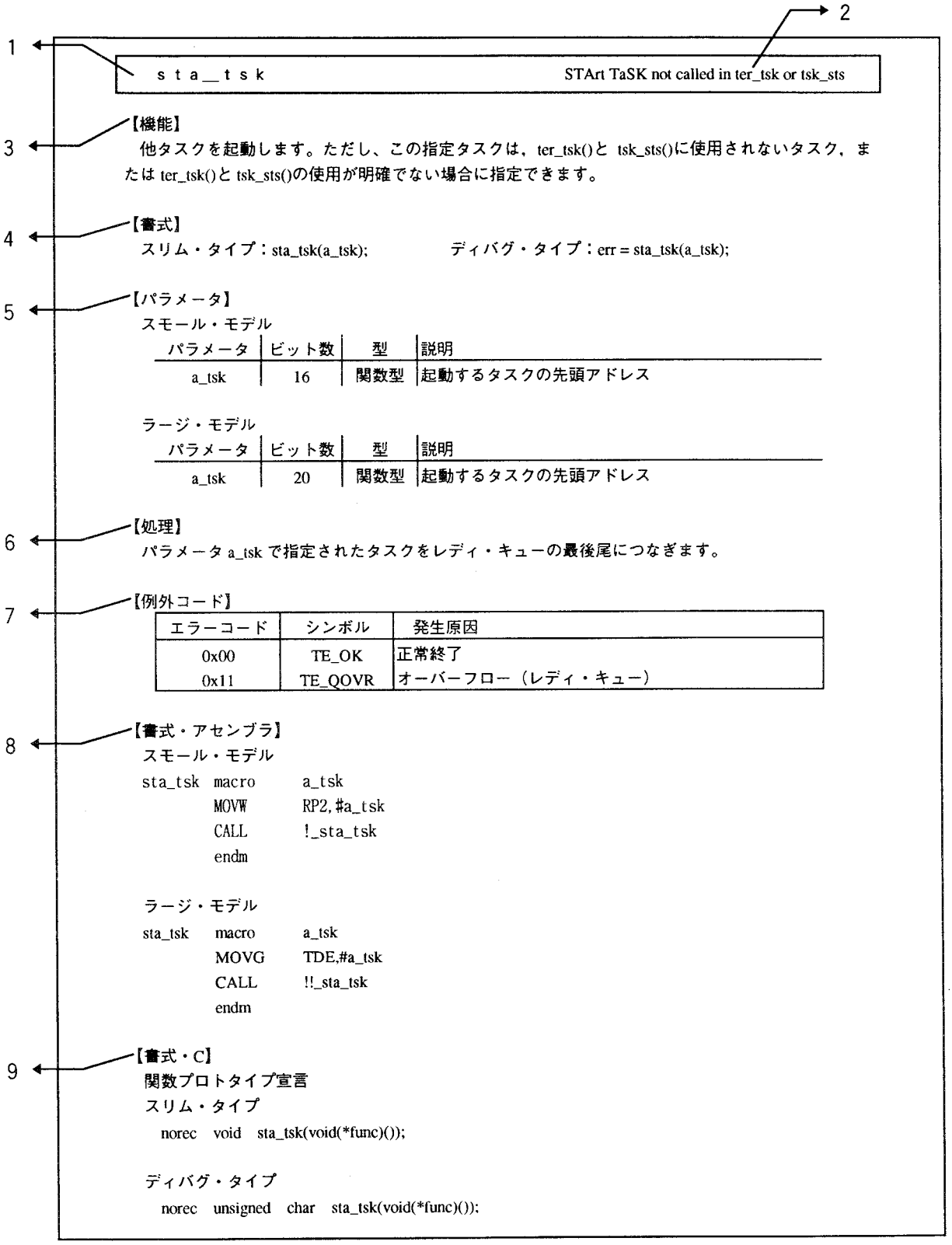
a_tsk: タスクの先頭アドレス（タスク名）
tmout: タイムアウトまでの時間（数値）
pk_tsks: タスクステータスの結果が返される変数のアドレス
a_flg: イベントフラグのアドレス
a_flg+1: イベントフラグのアドレス＋１
pk_finfo: イベントフラグの補足情報が格納しているテーブルのポインタ
setptn: セットされる値（数値）
clrptn: クリアする値（数値）
rplptn: 置き換える値（数値）

6.1.3 呼び出し方法

MX78K4 のシステムコールの呼び出しには、「CALL 命令」を用います。

6.2 システムコール仕様

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、下記のようなフォーマットで記述しています。



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

【機能】

他タスクを起動します。ただし、この指定タスクは，ter_tsk()と tsk_sts()に使用されないタスク，または ter_tsk()と tsk_sts()の使用が明確でない場合に指定できます。

【書式】

スリム・タイプ：sta_tsk(a_tsk);　　ディバグ・タイプ：err = sta_tsk(a_tsk);

【パラメータ】

スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス

【処理】

パラメータ a_tsk で指定されたタスクをレディ・キューの最後尾につなぎます。

【例外コード】

エラーコード	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】

スモール・モデル

```
sta_tsk  macro     a_tsk
         MOVW      RP2,#a_tsk
         CALL      !_sta_tsk
         endm
```

ラージ・モデル

```
sta_tsk  macro     a_tsk
         MOVG      TDE,#a_tsk
         CALL      !!_sta_tsk
         endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ

```
norec  void   sta_tsk(void(*func)());
```

ディバグ・タイプ

```
norec  unsigned  char  sta_tsk(void(*func)());
```

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。
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値の範囲を示しています。必要のない場合には、「なし」と記述しています。
6．処理内容。システムコールを実行する際の内部処理の手順を記述しています。この項目はタスクから見えるものでなくMX78K4の設計を行う際の処理概要を示したものです。
7．エラーコード。エラーコードの値(16進数)と、シンボル、エラーの内容を記述しています。なお、MX78K4 にはスリム・タイプとディバグ・タイプがあり、エラーコードを返すのは、ディバグ・タイプのMX78K4のみです。
8．書式・アセンブラ。アセンブラからシステムコールを呼び出す際の書式。(マクロ定義の形式で記述)
9．書式・C。C言語でシステムコールを呼び出す際の書式。(関数プロトタイプまたは関数マクロの形式で記述)

6.2.1 タスク関連システムコール

この節では、下記のシステムコールについて述べています。

sta_tsk: 他タスクを起動します。ただし、この指定タスクは、ter_tsk()と tsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

sta_tsk1: 他タスクを起動します。ただし、この指定タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

sta_tsk2: 他タスクを起動します。ただし、この指定タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

sta_tsk3: 他タスクを起動します。ただし、この指定タスクは、ter_tsk()と tsk_sts()で使用されたタスクを指定してください。

sta_tske: 他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()と tsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

sta_tsk1e: 他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

sta_tsk2e: 他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

sta_tsk3e: 他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()と tsk_sts()で使用されたタスクを指定してください。

sta_tskp: 指定された他タスクは、現在動作しているタスクの次に起動します。

sta_tskpe: 他タスクを起動し、自タスクを終了します。指定された他タスクは、現在動作しているタスクの次に起動します。

sta_tskf: イベントフラグ操作で起動する他タスクの条件を設定します。

sta_tskt: 指定した時間が経過したら、他タスクを起動します。

sta_tskte: 指定した時間が経過したら、他タスクを起動し、自タスクを終了します。

chg_tskt: 指定した他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・キューにある他タスクを強制終了させます。

ter_tsk: レディ・キューにある他タスクを強制終了させます。

ter_tskf: イベントフラク操作で起動される他タスクまたはレディ・キューにある他タスクを強制終了させます。

rot_rdq: レディ・キューを回転させます。ただし、回転する自タスクは、ter_tsk()と tsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

rot_rdq1: レディ・キューを回転させます。ただし、回転する自タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

rot_rdq2: レディ・キューを回転させます。ただし、回転する自タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

rot_rdq3: レディ・キューを回転させます。ただし、回転する自タスクは、ter_tsk()と tsk_sts()で使用されたタスクを指定してください。

rot_rdqe: レディ・キューを回転させ、終了します。ただし、回転する自タスクは、ter_tsk()とtsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

rot_rdq1e: レディ・キューを回転させ、終了します。ただし、回転する自タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

rot_rdq2e: レディ・キューを回転させ、終了します。。ただし、回転する自タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

rot_rdq3e: レディ・キューを回転させ、終了します。ただし、回転する自タスクは、ter_tsk()とtsk_sts()で使用されたタスクを指定してください。

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。

保守／廃止

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

【機能】

他タスクを起動します。ただし、この指定タスクは、ter_tsk()と tsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

【書式】

スリム・タイプ：sta_tsk(a_tsk);　　ディバグ・タイプ：err = sta_tsk(a_tsk);

【パラメータ】

スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス

【処理】

パラメータ a_tsk で指定されたタスクをレディ・キューの最後尾につなぎます。

【例外コード】

エラーコード	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】

スモール・モデル

```
sta_tsk     macro   a_tsk
            MOVW    RP2,#a_tsk
            CALL    !_sta_tsk
            endm
```

ラージ・モデル

```
sta_tsk     macro   a_tsk
            MOVG    TDE,#a_tsk
            CALL    !!_sta_tsk
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ

```
norec  void  sta_tsk(void(*func)());
```

ディバグ・タイプ

```
norec  unsigned  char  sta_tsk(void(*func)());
```

ｓｔａ＿ｔｓｋ１　STArt TaSK called in ter_tsk

【機能】

他タスクを起動します。ただし、この指定タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

【書式】

スリム・タイプ：sta_tsk1(a_tsk);　　ディバグ・タイプ：err = sta_tsk1(a_tsk);

【パラメータ】

スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス

【処理】

パラメータ a_tsk で指定されたタスクをレディ・キューの最後尾につなぎます。
ter_tsk()での高速検索のための準備処理を同時に行います。

【例外コード】

エラーコード	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】

スモール・モデル

```
sta_tsk1    macro   a_tsk
            MOVW    RP2,#a_tsk
            CALL    !_sta_tsk1
            endm
```

ラージ・モデル

```
sta_tsk1    macro   a_tsk
            MOVG    TDE,#a_tsk
            CALL    !!_sta_tsk1
            endm
```

【書式・Ｃ】

関数プロトタイプ宣言

スリム・タイプ

```
norec  void  sta_tsk1(void(*func)());
```

ディバグ・タイプ

```
norec  unsigned  char  sta_tsk1(void(*func)());
```

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

【機能】

他タスクを起動します。ただし、この指定タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

【書式】

スリム・タイプ：sta_tsk2(a_tsk);　　ディバグ・タイプ：err = sta_tsk2(a_tsk);

【パラメータ】

スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス

【処理】

パラメータ a_tsk で指定されたタスクをレディ・キューの最後尾につなぎます。
tsk_sts()での高速検索のための準備処理を同時に行います。

【例外コード】

エラーコード	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】

スモール・モデル

```
sta_tsk2        macro   a_tsk
                MOVW    RP2,#a_tsk
                CALL    !_sta_tsk2
                endm
```

ラージ・モデル

```
sta_tsk2        macro   a_tsk
                MOVG    TDE,#a_tsk
                CALL    !!_sta_tsk2
                endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ

```
norec   void    sta_tsk2(void(*func)());
```

ディバグ・タイプ

```
norec   unsigned   char   sta_tsk2(void(*func)());
```

s t a __ t s k 3　　STArt TaSK called in ter_tsk and tsk_sts

【機能】

他タスクを起動します。ただし、この指定タスクは、ter_tsk()と tsk_sts()で使用されたタスクを指定してください。

【書式】

スリム・タイプ：sta_tsk3(a_tsk);　　ディバグ・タイプ：err = sta_tsk3(a_tsk);

【パラメータ】

スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス

【処理】

パラメータ a_tsk で指定されたタスクをレディ・キューの最後尾につなぎます。
ter_tsk()と tsk_sts()での高速検索のための準備処理を同時に行います。

【例外コード】

エラーコード	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】

スモール・モデル

```
sta_tsk3    macro   a_tsk
            MOVW    RP2,#a_tsk
            CALL    !_sta_tsk3
            endm
```

ラージ・モデル

```
sta_tsk3    macro   a_tsk
            MOVG    TDE,#a_tsk
            CALL    !!_sta_tsk3
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ

```
norec  void  sta_tsk3(void(*func)());
```

ディバグ・タイプ

```
norec  unsigned  char  sta_tsk3(void(*func)());
```

保守／廃止

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

【機能】

他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()と tsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

【書式】

スリム・タイプ／ディバグ・タイプ

```
sta_tske(a_tsk);
```

【パラメータ】

スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス

【処理】

パラメータ a_tsk で指定されたタスクをレディ・キューの最後尾につなぎ、DORMANT 状態にします。

【例外コード】

なし

【書式・アセンブラ】

スモール・モデル

```
sta_tske    macro   a_tsk
            MOVW    RP2,#a_tsk
            CALL    !_sta_tske
            endm
```

ラージ・モデル

```
sta_tske    macro   a_tsk
            MOVG    TDE,#a_tsk
            CALL    !!_sta_tske
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ／ディバグ・タイプ

```
norec   void   sta_tske(void(*func)());
```

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

【機能】

他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

【書式】

スリム・タイプ／ディバグ・タイプ

```
sta_tsk1e(a_tsk);
```

【パラメータ】

スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス

【処理】

パラメータ a_tsk で指定されたタスクをレディ・キューの最後尾につなぎ、DORMANT 状態にします。ter_tsk()での高速検索のための準備処理を同時に行います。

【例外コード】

なし

【書式・アセンブラ】

スモール・モデル

```
sta_tsk1e       macro   a_tsk
                MOVW    RP2,#a_tsk
                CALL    !_sta_tsk1e
                endm
```

ラージ・モデル

```
sta_tsk1e       macro   a_tsk
                MOVG    TDE,#a_tsk
                CALL    !!_sta_tsk1e
                endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ／ディバグ・タイプ

```
norec   void   sta_tsk1e(void(*func)());
```

保守／廃止

ｓｔａ＿ｔｓｋ２ｅ　STArt TaSK called in tsk_sts,and Exit task

【機能】
　他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

【書式】
　スリム・タイプ／ディバグ・タイプ
```
sta_tsk2e(a_tsk);
```

【パラメータ】
スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス

【処理】
　パラメータ a_tsk で指定されたタスクをレディ・キューの最後尾につなぎ、DORMANT 状態にします。tsk_sts()での高速検索のための準備処理を同時に行います。

【例外コード】
　なし

【書式・アセンブラ】
スモール・モデル
```
sta_tsk2e       macro   a_tsk
                MOVW    RP2,#a_tsk
                CALL    !_sta_tsk2e
                endm
```

ラージ・モデル
```
sta_tsk2e       macro   a_tsk
                MOVG    TDE,#a_tsk
                CALL    !!_sta_tsk2e
                endm
```

【書式・Ｃ】
関数プロトタイプ宣言
　スリム・タイプ／ディバグ・タイプ
```
  norec  void  sta_tsk2e(void(*func)());
```

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

【機能】

他タスクを起動し、自タスクを終了します。ただし、この指定タスクは、ter_tsk()と tsk_sts()で使用されたタスクを指定してください。

【書式】

スリム・タイプ／ディバグ・タイプ

```
sta_tsk3e(a_tsk);
```

【パラメータ】

スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス

【処理】

パラメータ a_tsk で指定されたタスクをレディ・キューの最後尾につなぎ、DORMANT 状態にします。ter_tsk()と tsk_sts()での高速検索のための準備処理を同時に行います。

【例外コード】

なし

【書式・アセンブラ】

スモール・モデル

```
sta_tsk3e       macro   a_tsk
                MOVW    RP2,#a_tsk
                CALL    !_sta_tsk3e
                endm
```

ラージ・モデル

```
sta_tsk3e       macro   a_tsk
                MOVG    TDE,#a_tsk
                CALL    !!_sta_tsk3e
                endm
```

【書式・Ｃ】

関数プロトタイプ宣言

スリム・タイプ／ディバグ・タイプ

```
norec   void   sta_tsk3e(void(*func)());
```

保守／廃止

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

【機能】

指定された他タスクは、現在動作しているタスクの次に起動します。

【書式】

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

【パラメータ】

スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス

【処理】

パラメータ a_tsk で指定されたタスクをレディ・キューの先頭につなぎます。

【例外コード】

エラーコード	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】

スモール・モデル

```
sta_tskp    macro   a_tsk
            MOVW    RP2,#a_tsk
            CALL    !_sta_tskp
            endm
```

ラージ・モデル

```
sta_tskp    macro   a_tsk
            MOVG    TDE,#a_tsk
            CALL    !!_sta_tskp
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ

```
norec  void  sta_tskp(void(*func)());
```

ディバグ・タイプ

```
norec  unsigned  char  sta_tskp(void(*func)());
```

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

【機能】

他タスクを起動し、自タスクを終了します。指定された他タスクは、現在動作しているタスクの次に起動します。

【書式】

スリム・タイプ／ディバグ・タイプ

```
sta_tskpe(a_tsk);
```

【パラメータ】

スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス

【処理】

パラメータ a_tsk で指定されたタスクをレディ・キューの先頭につなぎ、DORMANT 状態にします。

【例外コード】

なし

【書式・アセンブラ】

スモール・モデル

```
sta_tskpe       macro   a_tsk
                MOVW    RP2,#a_tsk
                CALL    !_sta_tskpe
                endm
```

ラージ・モデル

```
sta_tskpe       macro   a_tsk
                MOVG    TDE,#a_tsk
                CALL    !!_sta_tskpe
                endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ／ディバグ・タイプ

```
norec   void   sta_tskpe(void(*func)());
```

保守／廃止

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他タスクの条件を設定します。

【書式】
　スリム・タイプ：sta_tskf(a_flg,pk_finfo);　　ディバグ・タイプ：sta_tskf(a_flg,pk_finfo);

【パラメータ】
スモール・モデル

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	イベントフラグの補足情報を格納しているテーブルの ID 番号
テーブルの各パラメータ	ビット数	型	説明
waiptn	8	char	タスクが要求しているビット・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

ラージ・モデル

パラメータ	ビット数	型	説明
a_flg	24	関数型	対象イベントフラグのアドレス
pk_finfo	8	char	イベントフラグの補足情報を格納しているテーブルの ID 番号
テーブルの各パラメータ	ビット数	型	説明
waiptn	8	char	タスクが要求しているビット・パターン
wfmode	8	char	タスクの起動条件
a_tsk	20	関数型	起動するタスクの先頭アドレス

【処理】
　対象イベントフラグのアドレスとイベントフラグの補足情報が格納しているテーブルのポインタを設定します。

【例外コード】
　なし

【書式・アセンブラ】
スモール・モデル

```
sta_tskf    macro   a_flg,pk_finfo
            MOV     a_flg,#pk_finfo
            endm
```

ラージ・モデル

```
sta_tskf    macro   a_flg,pk_finfo
            MOV     a_flg,#pk_finfo
            endm
```

【書式・C】
関数マクロ
スリム・タイプ／ディバグ・タイプ
#define sta_tskf(a,b) a.addrs=b

イベントフラグの型宣言はディフォールトで以下のようになっています。

```
struct event_flag     {
            unsigned char addrs;
            unsigned char setptn;
};
```

保守／廃止

s t a __ t s k t　　STArt TaSK after a Time

【機能】
　指定した時間が経過したら、他タスクを起動します。

【書式】
　スリム・タイプ：sta_tskt(a_tsk,tmout);　　ディバグ・タイプ：err = sta_tskt(a_tsk,tmout)

【パラメータ】
スモール・モデル

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	起動するまでの時間を示す

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス
tmout	16	unsigned int	起動するまでの時間を示す

【処理】
　指定されたタスクをタイマ・キューにつなぎます。指定する時間は、システム・クロックの倍数で指定します。

【例外コード】

エラーコード	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】
スモール・モデル

```
sta_tskt    macro   a_tsk,tmout
            MOVW    RP2,#a_tsk
            MOVW    AX,#tmout
            CALL    !_sta_tskt
            endm
```

ラージ・モデル

```
sta_tskt    macro   a_tsk,tmout
            MOVG    TDE,#a_tsk
            MOVW    AX,#tmout
            CALL    !!_sta_tskt
            endm
```

【書式・C】
関数プロトタイプ宣言
スリム・タイプ

```
norec   void    sta_tskt(void(*func)(),unsigned int);
```

ディバグ・タイプ

```
norec   unsigned    char    sta_tskt(void(*func)(),unsigned int);
```

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

【機能】
　指定した時間が経過したら、他タスクを起動し、自タスクを終了します。

【書式】
　スリム・タイプ／ディバグ・タイプ
　　sta_tskte(a_tsk,tmout);

【パラメータ】
スモール・モデル

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	起動するまでの時間を示す

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス
tmout	16	unsigned int	起動するまでの時間を示す

【処理】
　指定されたタスクをタイマ・キューにつなぎ、 DORMANT 状態にします。指定する時間は、システム・クロックの倍数で指定します。

【例外コード】
　なし

【書式・アセンブラ】
スモール・モデル

```
sta_tskte    macro   a_tsk,tmout
             MOVW    RP2,#a_tsk
             MOVW    AX,#tmout
             CALL    !_sta_tskte
             endm
```

ラージ・モデル

```
sta_tskte    macro   a_tsk,tmout
             MOVG    TDE,#a_tsk
             MOVW    AX,#tmout
             CALL    !!_sta_tskte
             endm
```

【書式・C】
関数プロトタイプ宣言
　スリム・タイプ／ディバグ・タイプ

```
norec  void  sta_tskte(void(*func)(),unsigned int);
```

保守／廃止

c h g __ t s k t　　CHanGe TaSK's startTime

【機能】
　指定した他タスクの起動時間を変更します。

【書式】
　スリム・タイプ：chg_tskt(a_tsk,tmout);　　ディバグ・タイプ：err = chg_tskt(a_tsk,tmout)

【パラメータ】
スモール・モデル

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	起動するまでの時間を示す

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス
tmout	16	unsigned int	起動するまでの時間を示す

【処理】
　指定されたタスクをタイマ・キュー、レディ・キューの順に捜します。もし、指定されたタスクがあれば、強制終了を行い、指定されたタスクをタイマ・キューにつなぎなおします。指定する時間は、システム・クロックの倍数で指定します。

【例外コード】

エラーコード	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】
スモール・モデル

```
chg_tskt    macro   a_tsk,tmout
            MOVW    RP2,#a_tsk
            MOVW    AX,#tmout
            CALL    !_chg_tskt
            endm
```

ラージ・モデル

```
chg_tskt    macro   a_tsk,tmout
            MOVG    TDE,#a_tsk
            MOVW    AX,#tmout
            CALL    !!_chg_tskt
            endm
```

【書式・C】
関数プロトタイプ宣言
スリム・タイプ

```
norec   void    chg_tskt(void(*func)(),unsigned int);
```

ディバグ・タイプ

```
norec   unsigned    char    chg_tskt(void(*func)(),unsigned int);
```

保守／廃止

c h g _ _ t s k t e　　CHanGe Task's startTime,and Exit task

【機能】
　指定した他タスクの起動時間を変更し、自タスクを終了します。

【書式】
　スリム・タイプ／ディバグ・タイプ
　　chg_tskte(a_tsk,tmout);

【パラメータ】
スモール・モデル

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	起動するまでの時間を示す

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	起動するタスクの先頭アドレス
tmout	16	unsigned int	起動するまでの時間を示す

【処理】
　指定されたタスクをタイマ・キュー、レディ・キューの順に捜します。もし、指定されたタスクがあれば、強制終了を行い、指定されたタスクをタイマ・キューにつなぎなおし、DORMANT 状態にします。指定する時間は、システム・クロックの倍数で指定します。

【例外コード】
　なし

【書式・アセンブラ】
スモール・モデル

```
chg_tskte    macro    a_tsk,tmout
             MOVW     RP2,#a_tsk
             MOVW     AX,#tmout
             CALL     !_chg_tskte
             endm
```

ラージ・モデル

```
chg_tskte    macro    a_tsk,tmout
             MOVG     TDE,#a_tsk
             MOVW     AX,#tmout
             CALL     !!_chg_tskte
             endm
```

【書式・C】
関数プロトタイプ宣言
　スリム・タイプ／ディバグ・タイプ

```
norec  void  chg_tskte(void(*func)(),unsigned int);
```

t e r __ t s k　　TERminate　TaSK in ready queue

【機能】

レディ・キューにある他タスクを強制終了させます。

【書式】

スリム・タイプ：ter_tsk(a_tsk);　　ディバグ・タイプ：err = ter_tsk(a_tsk);

【パラメータ】

スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	終了させるタスクの先頭アドレス

【処理】

指定されたタスクがレディ・キューにつながれている場合には、その指定されたタスクをレディ・キューから外し、DORMANT 状態にします。

【例外コード】

エラーコード	シンボル	発生原因
0x00	TE_OK	正常終了
0x0d	TE_DMT	指定したタスクが DORMANT 状態

【書式・アセンブラ】

スモール・モデル

```
ter_tsk     macro   a_tsk
            MOVW    RP2,#a_tsk
            CALL    !_ter_tsk
            endm
```

ラージ・モデル

```
ter_tsk     macro   a_tsk
            MOVG    TDE,#a_tsk
            CALL    !!_ter_tsk
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ

```
norec  void  ter_tsk(void(*func)());
```

ディバグ・タイプ

```
norec  unsigned  char  ter_tsk(void(*func)());
```

保守／廃止

t e r _ _ t s k t　　TERminate TaSK in ready queue or Timer queue

【機能】

タイマ・キューまたはレディ・キューにある他タスクを強制終了させます。

【書式】

スリム・タイプ：ter_tskt(a_tsk);　　ディバグ・タイプ：err = ter_tskt(a_tsk);

【パラメータ】

スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	終了させるタスクの先頭アドレス

【処理】

指定されたタスクがタイマ・キューまたはレディ・キューにつながれている場合には、その指定されたタスクをレディ・キューから外し、DORMANT 状態にします。

【例外コード】

エラーコード	シンボル	発生原因
0x00	TE_OK	正常終了
0x0d	TE_DMT	指定したタスクが DORMANT 状態

【書式・アセンブラ】

スモール・モデル

```
ter_tskt    macro   a_tsk
            MOVW    RP2,#a_tsk
            CALL    !_ter_tskt
            endm
```

ラージ・モデル

```
ter_tskt    macro   a_tsk
            MOVG    TDE,#a_tsk
            CALL    !!_ter_tskt
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ

```
norec  void  ter_tskt(void(*func)());
```

ディバグ・タイプ

```
norec  unsigned  char  ter_tskt(void(*func)());
```

t e r _ _ t s k f　TERminate TaSK in ready queue or event-Flag

【機能】
　イベントフラク操作で起動される他タスクまたはレディ・キューにある他タスクを強制終了させます。

【書式】
　スリム・タイプ：ter_tsk(a_tsk);　　ディバグ・タイプ：err = ter_tskf(a_tsk);

【パラメータ】
スモール・モデル

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

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	終了させるタスクの先頭アドレス

【処理】
　指定されたタスクがイベントフラクの操作で起動されるタスクまたはレディ・キューにつながれている場合には、その指定されたタスクをレディ・キューから外し、DORMANT 状態にします。

【例外コード】

エラーコード	シンボル	発生原因
0x00	TE_OK	正常終了
0x0d	TE_DMT	指定したタスクが DORMANT 状態

【書式・アセンブラ】
スモール・モデル

```
ter_tskf    macro   a_tsk
            MOVW    RP2,#a_tsk
            CALL    !_ter_tskf
            endm
```

ラージ・モデル

```
ter_tskf    macro   a_tsk
            MOVG    TDE,#a_tsk
            CALL    !!_ter_tskf
            endm
```

【書式・C】
関数プロトタイプ宣言
　スリム・タイプ

```
norec   void    ter_tskf(void(*func)());
```

　ディバグ・タイプ

```
norec   unsigned   char   ter_tskf(void(*func)());
```

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

【機能】

レディ・キューを回転させます。ただし、回転する自タスクは、ter_tsk()と tsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

【書式】

スリム・タイプ：rot_rdq();　　ディバグ・タイプ：err = rot_rdq();

【パラメータ】

なし

【処理】

自タスクのアドレスをレディ・キューの最後尾につなぎます。

【例外コード】

エラーコード	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】

スモール・モデル

```
rot_rdq     macro
            CALL    !_rot_rdq
            endm
```

ラージ・モデル

```
rot_rdq     macro
            CALL    !!_rot_rdq
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ

```
norec  void  rot_rdq();
```

ディバグ・タイプ

```
norec  unsigned  char  rot_rdq();
```

保守／廃止

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

【機能】

レディ・キューを回転させます。ただし、回転する自タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

【書式】

スリム・タイプ：rot_rdq1();　　ディバグ・タイプ：err = rot_rdq1();

【パラメータ】

なし

【処理】

自タスクのアドレスをレディ・キューの最後尾につなぎます。
ter_tsk()での高速検索のための準備処理を同時に行います。

【例外コード】

エラーコード	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】

スモール・モデル

```
rot_rdq1    macro
            CALL    !_rot_rdq1
            endm
```

ラージ・モデル

```
rot_rdq1    macro
            CALL    !!_rot_rdq1
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ

```
norec  void  rot_rdq1();
```

ディバグ・タイプ

```
norec  unsigned  char  rot_rdq1();
```

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

【機能】
　レディ・キューを回転させます。ただし、回転する自タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

【書式】
　スリム・タイプ：rot_tdq2();　　ディバグ・タイプ：err = rot_rdq2();

【パラメータ】
　なし

【処理】
　自タスクのアドレスをレディ・キューの最後尾につなぎます。
　tsk_sts()での高速検索のための準備処理を同時に行います。

【例外コード】

エラーコード	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】
　スモール・モデル

```
rot_rdq2    macro
            CALL    !_rot_rdq2
            endm
```

　ラージ・モデル

```
rot_rdq2    macro
            CALL    !!_rot_rdq2
            endm
```

【書式・C】
　関数プロトタイプ宣言
　　スリム・タイプ

```
norec  void  rot_rdq2();
```

　　ディバグ・タイプ

```
norec  unsigned  char  rot_rdq2();
```

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

【機能】

レディ・キューを回転させます。ただし、回転する自タスクは、ter_tsk()と tsk_sts()で使用されたタスクを指定してください。

【書式】

スリム・タイプ：rot_rdq3();　　ディバグ・タイプ：err = rot_rdq3();

【パラメータ】

なし

【処理】

自タスクのアドレスをレディ・キューの最後尾につなぎます。
ter_tsk()と tsk_sts()での高速検索のための準備処理を同時に行います。

【例外コード】

エラーコード	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】

スモール・モデル

```
rot_rdq3    macro
            CALL    !_rot_rdq3
            endm
```

ラージ・モデル

```
rot_rdq3    macro
            CALL    !!_rot_rdq3
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ

```
norec  void  rot_rdq3();
```

ディバグ・タイプ

```
norec  unsigned  char  rot_rdq3();
```

r o t __ r d q e　ROTate ReaDy Queue and Exit task (current task is not called in ter_tsk or tsk_sts)

【機能】

レディ・キューを回転させ、終了します。ただし、回転する自タスクは、 ter_tsk()と tsk_sts()に使用されないタスク、または ter_tsk()と tsk_sts()の使用が明確でない場合に指定してください。

【書式】

スリム・タイプ：rot_rdqe();　　ディバグ・タイプ：rot_rdqe();

【パラメータ】

なし

【処理】

自タスクのアドレスをレディ・キューの最後尾につなぎ、自タスクを終了させ、DORMANT 状態にします。

【例外コード】

なし

【書式・アセンブラ】

スモール・モデル

```
rot_rdqe    macro
            CALL    !_rot_rdqe
            endm
```

ラージ・モデル

```
rot_rdqe    macro
            CALL    !!_rot_rdqe
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ／ディバグ・タイプ

```
norec  void  rot_rdqe();
```

保守／廃止

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

【機能】

レディ・キューを回転させ、終了します。ただし、回転する自タスクは、ter_tsk()で使用され、tsk_sts()には使用されていないタスクを指定してください。

【書式】

スリム・タイプ／ディバグ・タイプ

```
rot_rdq1e();
```

【パラメータ】

なし

【処理】

自タスクのアドレスをレディ・キューの最後尾につなぎ、自タスクを終了させ、DORMANT 状態にします。

ter_tsk()での高速検索のための準備処理を同時に行います。

【例外コード】

なし

【書式・アセンブラ】

スモール・モデル

```
rot_rdq1e       macro
                CALL    !_rot_rdq1e
                endm
```

ラージ・モデル

```
rot_rdq1e       macro
                CALL    !!_rot_rdq1e
                endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ／ディバグ・タイプ

```
norec   void   rot_rdq1e();
```

保守／廃止

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

【機能】
　レディ・キューを回転させ、終了します。ただし、回転する自タスクは、ter_tsk()には使用されず、tsk_sts()で使用されているタスクを指定してください。

【書式】
　スリム・タイプ／ディバグ・タイプ
```
rot_rdq2e();
```

【パラメータ】
　なし

【処理】
　自タスクのアドレスをレディ・キューの最後尾につなぎ、自タスクを終了させ、DORMANT 状態にします。
　tsk_sts()での高速検索のための準備処理を同時に行います。

【例外コード】
　なし

【書式・アセンブラ】
　スモール・モデル
```
rot_rdq2e       macro
                CALL    !_rot_rdq2e
                endm
```

　ラージ・モデル
```
rot_rdq2e       macro
                CALL    !!_rot_rdq2e
                endm
```

【書式・C】
　関数プロトタイプ宣言
　　スリム・タイプ／ディバグ・タイプ
```
norec   void   rot_rdq2e();
```

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】
　レディ・キューを回転させ、終了します。ただし、回転する自タスクは、ter_tsk()と tsk_sts()で使用されたタスクを指定してください。

【書式】
　スリム・タイプ／ディバグ・タイプ
　　rot_rdq3e();

【パラメータ】
　なし

【処理】
　自タスクのアドレスをレディ・キューの最後尾につなぎ、自タスクを終了させ、DORMANT 状態にします。
　ter_tsk()と tsk_sts()での高速検索のための準備処理を同時に行います。

【例外コード】
　なし

【書式・アセンブラ】
　スモール・モデル

```
rot_rdq3e    macro
             CALL    !_rot_rdq3e
             endm
```

　ラージ・モデル

```
rot_rdq3e    macro
             CALL    !!_rot_rdq3e
             endm
```

【書式・C】
　関数プロトタイプ宣言
　　スリム・タイプ／ディバグ・タイプ

```
norec  void  rot_rdq3e();
```

t s k __ s t s　　get TaSK STatuS

【機能】

タスクの状態を得ます。

【書式】

スリム・タイプ／ディバグ・タイプ

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

【パラメータ】

スモール・モデル

パラメータ	ビット数	型	説明
pk_tskst	16	関数型	タスクの状態を返すエリアへのポインタ
a_tsk	16	関数型	タスクの先頭アドレス

ラージ・モデル

パラメータ	ビット数	型	説明
pk_tskst	20	関数型	タスクの状態を返すエリアへのポインタ
a_tsk	20	関数型	タスクの先頭アドレス

【処理】

指定されたタスクがレディ・キューにつながれている場合には pk_tskst で示されたエリアに TTE_RDY を返し、つながれていない場合には TTE_DMT を返します。

リターン・コード	シンボル	説明
0x02	TTS_RDY	対象タスクが READY 状態である
0x10	TTS_DMT	対象タスクが DORMANT 状態である （時間待ち、フラグ待ちも含む）

【例外コード】

なし

【書式・アセンブラ】

スモール・モデル

```
tsk_sts     macro   pk_tsks,a_tsk
            MOVW    RP2,#pk_tsks
            MOVW    DE,# a_tsk
            CALL    !_tsk_sts
            endm
```

ラージ・モデル

```
tsk_sts     macro   pk_tsks,a_tsk
            MOVG    TDE,#pk_tsks
            MOVG    WHL,#a_tsk
            MOVG    _@NRARG0,WHL
            CALL    !!_tsk_sts
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ／ディバグ・タイプ

```
norec   void   tsk_sts(unsigned   char *,void(*func)());
```

保守／廃止

c h g __ p r i　CHanGe PRIority

【機能】

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。

【書式】

スリム・タイプ：chg_pri(a_tsk);　　ディバグ・タイプ：err = chg_pri(a_tsk);

【パラメータ】

スモール・モデル

パラメータ	ビット数	型	説明
a_tsk	16	関数型	つなぎなおすタスクの先頭アドレス

ラージ・モデル

パラメータ	ビット数	型	説明
a_tsk	20	関数型	つなぎなおすタスクの先頭アドレス

【処理】

a_tsk で指定されたタスクをレディ・キューから捜し出し、レディ・キューの先頭につなぎます。

【例外コード】

エラーコード	シンボル	発生原因
0x00	TE_OK	正常終了
0x0d	TE_DMT	指定したタスクが DORMANT 状態

【書式・アセンブラ】

スモール・モデル

```
chg_pri     macro   a_tsk
            MOVW    RP2,#a_tsk
            CALL    !_chg_pri
            endm
```

ラージ・モデル

```
chg_pri     macro   a_tsk
            MOVG    TDE,#a_tsk
            CALL    !!_chg_pri
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ

```
norec  void  chg_pri(void(*func)());
```

ディバグ・タイプ

```
norec  unsigned  char  chg_pri(void(*func)());
```

保守／廃止

ｅｘｔ＿ｔｓｋ　EXiT TaSK

【機能】
　自タスクを終了します。

【書式】
　スリム・タイプ／ディバグ・タイプ
```
ext_tsk();
```

【パラメータ】
　なし

【処理】
　自タスクを DORMANT 状態にします。

【例外コード】
　なし

【書式・アセンブラ】
　スモール・モデル
```
ext_tsk     macro
            CALL    !_ext_tsk
            endm
```

　ラージ・モデル
```
ext_tsk     macro
            CALL    !!_ext_tsk
            endm
```

【書式・Ｃ】
　関数プロトタイプ宣言
　スリム・タイプ／ディバグ・タイプ
```
norec  void  ext_tsk();
```

6.2.2 イベントフラグ関連システムコール

この節では、下記のシステムコールについて述べます。

set_flg: 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。

clr_flg: 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。

rpl_flg: イベントフラグに任意のビット・パターンと入れ替えます。

ｓｅｔ＿ｆｌｇ　SET event-FLaG

【機能】
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】
スリム・タイプ：set_flg(a_flg,setptn);　　ディバグ・タイプ：set_flg(a_flg,setptn,err);

【パラメータ】
スモール・モデル

パラメータ	ビット数	型	説明
a_flg	16	関数型	ビット・パターンをセットする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	char	例外コードを格納する領域

ラージ・モデル

パラメータ	ビット数	型	説明
a_flg	24	関数型	ビット・パターンをセットする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	char	例外コードを格納する領域

【処理】
setptn で指定されたビット・パターンと、a_flg で指定されたイベントフラグのビット・パターンとのORをとり、イベントフラグのビット・パターンを変更します。もし、イベントフラグが、指定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ていれば、登録されているタスクをレディ・キューの先頭につなぎます。

【例外コード】

エラーコード	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】
スモール・モデル

```
set_flg     macro   a_flg,setptn
            LOCAL   setflge
            OR      (a_flg+1),#setptn
            CMP     a_flg,#00h
            BE      $setflge
            MOVW    RP2,#a_flg
            CALL    !_tsetflg
setflge:
            endm
```

ラージ・モデル

```
set_flg     macro   a_flg,setptn
            LOCAL   setflge
            OR      (a_flg+1),#setptn
            CMP     a_flg,#00h
            BE      $setflge
            MOVG    TDE,#a_flg
            CALL    !!_tsetflg
setflge:
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ

```
norec   void   tsetflg(unsigned char *);
```

ディバグ・タイプ

```
norec   unsigned   char   tsetflg(unsigned char *);
```

関数マクロ

スリム・タイプ

```
#define   set_flg(a,b) a.setptnl=b;if(a.addrs!=0) {tsetflg(&a.addrs);}
```

ディバグ・タイプ

```
#define   set_flg(a,b,c) a.setptnl=b;if(a.addrs!=0) {c=tsetflg(&a.addrs);}
```

イベントフラグの型宣言はディフォールトで以下のようになっています。

```
struct event_flag     {
            unsigned char addrs;
            unsigned char setptn;
};
```

c l r __ f l g　　CLeaR event-FLaG

【機能】
　指定したイベントフラグの指定したビットをクリアします。

【書式】
　スリム・タイプ／ディバグ・タイプ
```
clr_flg(a_flg,setptn);
```

【パラメータ】
スモール・モデル

パラメータ	ビット数	型	説明
a_flg	16	関数型	ビット・パターンをセットするイベントフラグのアドレス
setptn	8	char	クリアするビットを示す情報

ラージ・モデル

パラメータ	ビット数	型	説明
a_flg	24	関数型	ビット・パターンをセットするイベントフラグのアドレス
setptn	8	char	クリアするビットを示す情報

【処理】
　指定されたイベントフラグのビット・パターンのうち、setptn で指定されたビットが 1 である箇所のビットを 0 クリアします。

【例外コード】
　なし

【書式・アセンブラ】
スモール・モデル
```
clr_flg     macro   a_flg,setptn
                    AND     (a_flg+1),#setptn
                    endm
```

ラージ・モデル
```
clr_flg     macro   a_flg,setptn
                    AND     (a_flg+1),#setptn
                    endm
```

【書式・C】
関数マクロ
　スリム・タイプ／ディバグ・タイプ
```
#define  clr_flg(a,b) a.setptn&=b
```

イベントフラグの型宣言はディフォールトで以下のようになっています。

```
struct event_flag    {
            unsigned char addrs;
            unsigned char setptn;
};
```

r p l __ f l g　　RePLace event-FLaG

【機能】
イベントフラグに任意のビット・パターンと入れ替えます。

【書式】
スリム・タイプ：rpl_flg(a_flg,setptn);　　ディバグ・タイプ：rpl_flg(a_flg,setptn,err);

【パラメータ】
スモール・モデル

パラメータ	ビット数	型	説明
a_flg	16	関数型	ビット・パターンをセットする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

ラージ・モデル

パラメータ	ビット数	型	説明
a_flg	24	関数型	ビット・パターンをセットする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

【処理】
setptn で指定されたビット・パターンを、a_flg で指定されたイベントフラグのビット・パターンに入れ替えます。もし、指定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ていれば、登録されているタスクをレディ・キューの先頭につなぎます。

【例外コード】

エラーコード	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】
スモール・モデル

```
rpl_flg     macro   a_flg,setptn
            LOCAL   rplflge
            MOV     (a_flg+1),#setptn
            CMP     a_flg,#00h
            BE      $rplflge
            MOVW    RP2,#a_flg
            CALL    !_tsetflg
rplflge:
            endm
```

ラージ・モデル

```
rpl_flg     macro   a_flg,setptn
            LOCAL   rplflge
            MOV     (a_flg+1),#setptn
            CMP     a_flg,#00h
            BE      $rplflge
            MOVG    TDE,#a_flg
            CALL    !!_tsetflg
rplflge:
            endm
```

【書式・C】

関数プロトタイプ宣言

スリム・タイプ

```
norec   void   tsetflg(unsigned char *);
```

ディバグ・タイプ

```
norec   unsigned   char   tsetflg(unsigned char *);
```

関数マクロ

スリム・タイプ

```
#define   rpl_flg(a,b) a.setptn=b;if(a.addrs!=0) {tsetflg(&a.addrs);}
```

ディバグ・タイプ

```
#define   rpl_flg(a,b,c) a.setptn=b;if(a.addrs!=0) {c=tsetflg(&a.addrs);}
```

イベントフラグの型宣言はディフォールトで以下のようになっています。

```
struct event_flag     {
            unsigned char addrs;
            unsigned char setptn;
};
```

6.3 システムコールのエラー

MX78K4 にはスリム・タイプと、ディバグ・タイプの 2 種類があります。
このうち、システムコールがエラーを返すのはディバグ・タイプのみで、スリム・タイプはエラーコードを返しません。以下に MX78K4 のディバグ・タイプが返すエラーの一覧を示します。

エラーコード	シンボル	発生原因
0x00	TE_OK	正常終了
0x0d	TE_DMT	指定のタスクが DORMANT 状態
0x11	TE_QOVR	オーバフロー（レディ・キュー、タイマ・キュー）

6.4 システムコール別スタック使用量

システムコール処理に必要なスタック容量を以下に示します。なお、下記の数値はアセンブラにてシステムコールを発行した場合の数値です。

システムコール名	ラージ・モデル	スモール・モデル
sta_tsk	8 バイト	7 バイト
sta_tsk1	8 バイト	7 バイト
sta_tsk2	8 バイト	7 バイト
sta_tsk3	8 バイト	7 バイト
sta_tske	8 バイト	7 バイト
sta_tsk1e	8 バイト	7 バイト
sta_tsk2e	8 バイト	7 バイト
sta_tsk3e	8 バイト	7 バイト
sta_tskp	15 バイト	13 バイト
sta_tskpe	8 バイト	7 バイト
sta_tskf	0 バイト	0 バイト
sta_tskt	18 バイト	15 バイト
sta_tskte	21 バイト	18 バイト
chg_tskt	24 バイト	21 バイト
chg_tskte	24 バイト	21 バイト
ter_tsk	12 バイト	12 バイト
ter_tskt	16 バイト	14 バイト
ter_tskf	13 バイト	12 バイト
rot_rdq	5 バイト	5 バイト
rot_rdq1	5 バイト	5 バイト
rot_rdq2	5 バイト	5 バイト
rot_rdq3	5 バイト	5 バイト
rot_rdqe	5 バイト	7 バイト
rot_rdq1e	5 バイト	7 バイト
rot_rdq2e	5 バイト	7 バイト
rot_rdq3e	5 バイト	7 バイト
tsk_sts	13 バイト	12 バイト
chg_pri	11 バイト	10 バイト
ext_tsk	5 バイト	15 バイト
set_flg	18 バイト	15 バイト
clr_flg	0 バイト	0 バイト
rpl_flg	18 バイト	15 バイト
タイマ処理	14 バイト	15 バイト
レディキューの初期化	5 バイト	7 バイト
タイマキューの初期化	5 バイト	7 バイト

第 7 章　初期化処理

この章での初期化処理とは、MX78K4 が動作を開始するうえで必要な環境を作り上げる処理です。

7.1　初期化処理の概要

初期化処理には、以下の項目であります。

・MX78K4 のキューを初期化

MX78K4 がタスクを管理するためのキュー（レディ・キュー、タイマ・キュー）を初期化します。

・割り込み準備

タイマ・キューを使用する場合、タイマ・キューの基準タイマを設定します。

・タスクの起動

MX78K4 が起動したときに、起動していなければならないタスクを順番に起動手続きを行います。

・イベントフラグの準備

イベントフラグは、タスクの中での構造体変数の宣言によって、その領域は確保・初期化されます。また、イベントフラグの補足情報は、あらかじめ指定構造で ROM 中に存在していても構いません（静的生成）。さらに、初期化処理中またはタスクの処理中に構造体変数の定義・宣言を行っても構わない（動的生成）ソフトウエアの初期化が終了すると、初期タスクを起動させる処理に移ります。

7.2　MX78K4 の初期設定

MX78K4 を初期化する前に、スタック・ポインタと MX78K4 がワークで使用するレジスタバンクを指定する必要があります。スタック・ポインタに関しては、自動設定を行わない場合に設定してください。下記にスタック・ポインタの自動設定を行わなかった場合の例を示します。

```
PUBLIC      _@STBEG
PUBLIC      _MX_bank

_@STBEG     EQU     ________h           ; スタック・ポインタの先頭アドレス
_MX_bank    EQU     0___0000h           ; ___で MX78K4 が使用するレジスタバンク
                                        ; を示す。
```

7.3　MX78K4 のキューの初期化

MX78K4 には、レディ・キューとタイマ・キューの 2 種類のキューがあります。それぞれのキューは、システム初期化時に領域の確保と初期化することになります。レディ・キューのサイズは、ユーザ・システムによって、サイズが異なります。また、スモール・モデルとラージ・モデルによってもサイズが異なります。以下にレディ・キューのサイズの算出方法を示します。

ユーザ・システムにおいて、同時に ready 状態になるタスクの最大数を n とすると、

スモール・モデル ：（n＋1）× 4 byte
ラージ・モデル ：（n＋1）× 5 byte

となります。

タイマ・キューについても同じで、ユーザ・システム、スモール・モデルとラージ・モデルでサイズが異なります。以下にタイマ・キューのサイズの算出方法を示します。

ユーザ・システムにおいて、同時に時間待ち状態になるタスクの最大数を n とすると、

スモール・モデル ：（n＋1）× 4 byte
ラージ・モデル ：（n＋1）× 5 byte

となります。

例．スモール・モデルを使用し、同時に ready 状態になるタスクの最大数が 8 個とします。

レディ・キューのサイズ＝（8＋1）×4 byte

上記の算出法により、この場合のレディ・キューのサイズは 36 byte となります。

レディ・キューの領域確保は、添付されている初期化ルーチン“MX78K4RL.ASM”（ラージ・モデル用）で行ってください（スモール・モデル用の初期化ルーチンは“MX78K4RS.ASM”となります）。下記にレディ・キューの配置情報とその確保のサンプルを示します。

```
PUBLIC      _MX_rq_StartAddress
PUBLIC      _MX_rq_e0
PUBLIC      az_arg

_MX_rq_StartAddress     EQU     ________h       ; レディキューの先頭番地(RAM 配置制限なし)
_MX_rq_MAX              EQU     ________        ; レディキューに格納される最大タスク数
                                                ; (最大 50 個)
L_MXK4Rq    DSEG    AT      _MX_rq_StartAddress
_MX_rq_area:    DS      (_MX_rq_Max+1)*5        ;スモール・モデルの場合は“*4”
_MX_rq_e0       EQU     _MX_rq_Max*5            ;スモール・モデルの場合は“*4”
                                                ;レディキューの最終キューのアドレス値
az_arg          DS      2                       ; AZ78K4 用（必ず確保してください。）
```

ユーザ・システムで、”sta_tskt”,”sta_tskte”,”chg_tskt”,”chg_tskte”,”ter_tskt”システムコールを使用する場合は、タイマ・キュー用の領域を確保してください。

タイマ・キューの領域確保は、添付されている初期化ルーチン“MX78K4RL.ASM”（ラージ・モデル用）で行ってください（スモール・モデル用の初期化ルーチンは“MX78K4RS.ASM”となります）。下記にタイマ・キューの配置情報とその確保のサンプルを示します。

```
PUBLIC          _MX_tq_StartAddress
PUBLIC          _MX_tq_r_e0
PUBLIC          _MX_tq_a_e0
PUBLIC          _MX_tq_a_e0h
PUBLIC          _MX_tq_a_e01
PUBLIC          _MX_tq_half
PUBLIC          _Prohibit_MX_int
PUBLIC          _Permit_MX_int

_MX_tq_StartAddress     EQU     ________h       ; タイマキューの先頭番地(RAM 配置制限なし)
_MX_tq_MAX              EQU     ________        ; タイマキューに格納される最大タスク数
                                                ;(最大 50 個)
mx78k4tq        DSEG    AT      _MX_tq_StartAddress
_MX_tq_area:    DS      (_MX_tq_Max+1)*5        ;スモール・モデルの場合は“*4”
_MX_tq_r_e0     EQU     _MX_tq_Max*5            ;スモール・モデルの場合は“*4”
                                                ;タイマキューの相対最終キューのアドレス値
_MX_tq_a_e0     EQU     _MX_tq_StartAddress+_MX_tq_r_e0
                                                ;タイマキューの絶対最終キューのアドレス値
_MX_tq_a_e0h    EQU     _MX_tq_a_e0 SHR 16      ;タイマキューの絶対最終キューのアドレス値
                                                ;の上位 8bit
_MX_tq_a_e01    EQU     _MX_tq_a_e0 AND 0ffffH  ;タイマキューの絶対最終キューのアドレス値
                                                ;の下位 16bit
_MX_tq_half     EQU     ((_MX_tq_Max+1) SHR 1)*5 ; スモール・モデルの場合は“*4”
```

7.4 OS 作業用レジスタバンクの指定

78K/Ⅳシリーズには、8 つのレジスタバンクが存在しますが、MX78K4 が動作する上ためには、1 レジスタバンクを占有して使用できるように指定しなければなりません。以下にレジスタバンク指定のサンプルを示します。

```
PUBLIC          _MX_bank

_MX_bank                EQU     0___0000b       ; ___でバンクを示す
```

7.5 イベントフラグの領域確保と初期化

イベントフラグを使用する場合は、イベントフラグ本体と、イベントフラグ補足情報テーブル領域を確保してください。

イベントフラグ本体は、イベントフラグ毎に 2byte の領域が、イベントフラグ補足情報テーブルは、テーブル毎にスモール・モデルは 4byte、ラージ・モデルは 5byte 使用します。

以下にイベントフラグの配置情報とその確保のサンプルを示します。

```
PUBLIC      _MX_flags
PUBLIC      _flag_end

EXTERN      _________,_________,_________    ;イベントフラグの外部参照宣言

mx78k4ef    DSEG
_MX_flags:
_________:  DS      2                       ;各イベントフラグ
_________:  DS      2                       ;各イベントフラグ
_________:  DS      2                       ;各イベントフラグ
_flag_end:  DS      1                       ;ユーザプログラムで“0ffh”をセット（“0ffh”は
                                            ;イベントフラグの終端を示す）
```

次にイベントフラグ補足情報テーブルの配置情報とその確保のサンプルを示します。

```
PUBLIC      _MX_flg_tbl

EXTERN      _________,_________             ;複数のイベントフラグある場合、全てを宣言

mx78k4et    CSEG
_MX_flg_tbl:
            DB      EVENT_xxx               ;ID=1   イベントフラグの待ちモード
                                                    (EVENT_OR,EVENT_AND)
            DB      __h                     ;       イベントフラグの待ちパターン
            DG      _________              ;       起動するタスク(ラベル)
                                            ;スモール・モデルの場合は”DW”で確保
            DB      EVENT_xxx               ;ID=2   イベントフラグの待ちモード
                                                    (EVENT_OR,EVENT_AND)
            DB      __h                     ;       イベントフラグの待ちパターン
            DG      _________              ;       起動するタスク(ラベル)
                                            ;スモール・モデルの場合は”DW”で確保
```

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】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部	〒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	名古屋市中区錦一丁目17番1号（NEC中部ビル）	名古屋 (052)222-2170
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 (06) 945-3178 大　阪 (06) 945-3200 大　阪 (06) 945-3208

北海道支社	札　幌 (011)231-0161	宇都宮支店	宇都宮 (028)621-2281	富山支店	富　山 (0764)31-8461
東北支社	仙　台 (022)261-5511	小山支店	小　山 (0285)24-5011	三重支店	津　(0592)25-7341
岩手支店	盛　岡 (0196)51-4344	長野支社	長　野 (026)235-1444	京都支社	京　都 (075)344-7824
山形支店	山　形 (0236)23-5511	松本支店	松　本 (0263)35-1666	神戸支社	神　戸 (078)333-3854
郡山支店	郡　山 (0249)23-5511	上諏訪支店	諏　訪 (0266)53-5350	中国支社	広　島 (082)242-5504
いわき支店	いわき (0246)21-5511	甲府支店	甲　府 (0552)24-4141	鳥取支店	鳥　取 (0857)27-5311
長岡支店	長　岡 (0258)36-2155	埼玉支社	大　宮 (048)641-1411	岡山支店	岡　山 (086)225-4455
土浦支店	土　浦 (0298)23-6161	立川支社	立　川 (0425)26-5981	四国支社	高　松 (0878)36-1200
水戸支店	水　戸 (0292)26-1717	千葉支社	千　葉 (043)238-8116	新居浜支店	新居浜 (0897)32-5001
神奈川支社	横　浜 (045)324-5511	静岡支社	静　岡 (054)255-2211	松山支店	松　山 (089)945-4111
群馬支店	高　崎 (0273)26-1255	北陸支社	金　沢 (0762)23-1621	九州支社	福　岡 (092)271-7700
太田支店	太　田 (0276)46-4011	福井支店	福　井 (0776)22-1866	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】

半導体ソリューション技術本部 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
半導体販売技術本部 東日本販売技術部	〒108-01	東京都港区芝五丁目7番1号（NEC本社ビル）	東　京	(03)3798-9619
半導体販売技術本部 中部販売技術部	〒460	名古屋市中区錦一丁目17番1号（NEC中部ビル）	名古屋	(052)222-2125
半導体販売技術本部 西日本販売技術部	〒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

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）

C95.12