NEC

ユーザーズ・マニュアル

保守／廃止

78K/0S シリーズ用 OS

基礎編

MX78K0S

対象デバイス

78K/0S シリーズ

資料番号　U12938JJ1V0UM00（第1版）
発行年月　September 1997 NS

保守／廃止

〔メ　モ〕

保守／廃止

目次要約

保守／廃止

○本資料の内容は，後日変更する場合があります。
○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所有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。

M7A 94.11

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/0S シリーズの組み込み用ソフトウェアである，「78K/0S シリーズ用 OS MX78K0S」をお買い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/0S シリーズ用 OS　MX78K0S 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・マニュアルを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，“CC78K0S　Cコンパイラ”と“RA78K0S　アセンブラ・パッケージ”が必要になります。

保守／廃止

≪ 凡　　例 ≫

本マニュアル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
＿　　：重要個所，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	
	和文	英文
CC78K0S Cコンパイラ 操作編　ユーザーズ・マニュアル	U11816J	U11816E
CC78K0S Cコンパイラ 言語編　ユーザーズ・マニュアル	U11817J	U11817E
RA78K0S アセンブリ言語編　ユーザーズ・マニュアル	U11599J	U11599E
RA78K0S 操作編 RA78K0S Ver.1.00以上 ユーザーズ・マニュアル	U11622J	U11622E
RA78K0S 構造化アセンブリ言語編 ST78K0S Ver.1.00　ユーザーズ・マニュアル	U11623J	U11623E
78K/0Sシリーズ 命令編　ユーザーズ・マニュアル	U11047J	U11047E
78K/0Sシリーズ用OS　MX78K0S　テクニカル編 ユーザーズ・マニュアル	U12937J	作成予定
idea-L　スクリーン・エディタ　入門編 ユーザーズ・マニュアル	U10094J	作成中

保守／廃止

〔メ　モ〕

保守／廃止

目　　次

保守／廃止

保守／廃止

保守／廃止

図の目次

保守／廃止

表の目次

保守／廃止

〔メ　モ〕

保守／廃止

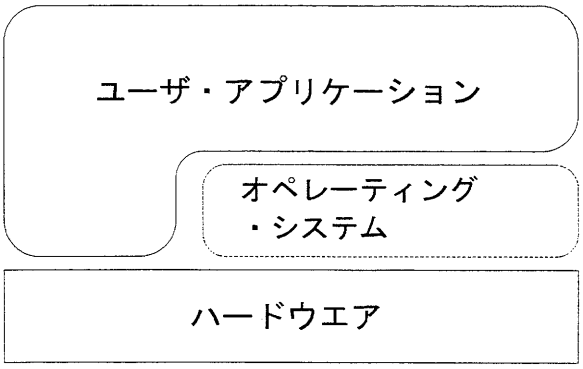
第1章　概　　説

本 OS 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/0S 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



1.2　特　　徴

78K/0Sシリーズ用のOSである本OSの特徴を次に示します。

(1) システム・コールは，μ ITRON 仕様のサブセット

システム・コールは，μ ITRON 仕様のサブセットとなっています。これにより，μ ITRON 仕様準拠のリアルタイム OS である RX78K シリーズに移行することが容易となります。

(2) 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。

(3) 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。

(4) プリエンション機能の削除

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。

(5) 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，次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，次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

次に対応するＣＰＵを示します。

・78K/0Sシリーズ

保守／廃止

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次に示します。

(1) ハードウエア

・PC-9800 シリーズ（日本語版のみ提供）

・IBM PC/AT™およびその互換機（日本語版・英語版を提供）

(2) ソフトウエア

・アセンブラ・パッケージ

NEC 製　RA78K0S（78K/0S 用）

・C コンパイラ

NEC 製　CC78K0S（78K/0S 用）

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次のソフトウエアが必要です。

・NEC製　idea-L　スクリーン・エディタ　Ver.3.1以上

または

・idea-L　Light　Ver.1.0以上

注意　idea-L　Lightではライブラリ機能は使用できません。

(3) オペレーティング・システム

(1) ハードウエアで示したハードウエア上で (2) ソフトウエアで示したソフトウエアが動作するオペレーティング・システムが必要です。

注意　本製品はWindows™版となっておりますが，Windowsに限らず，上記の条件を満たすオペレーティング・システム上で使用することができます。

保守／廃止

〔メ　モ〕

第２章　インストール

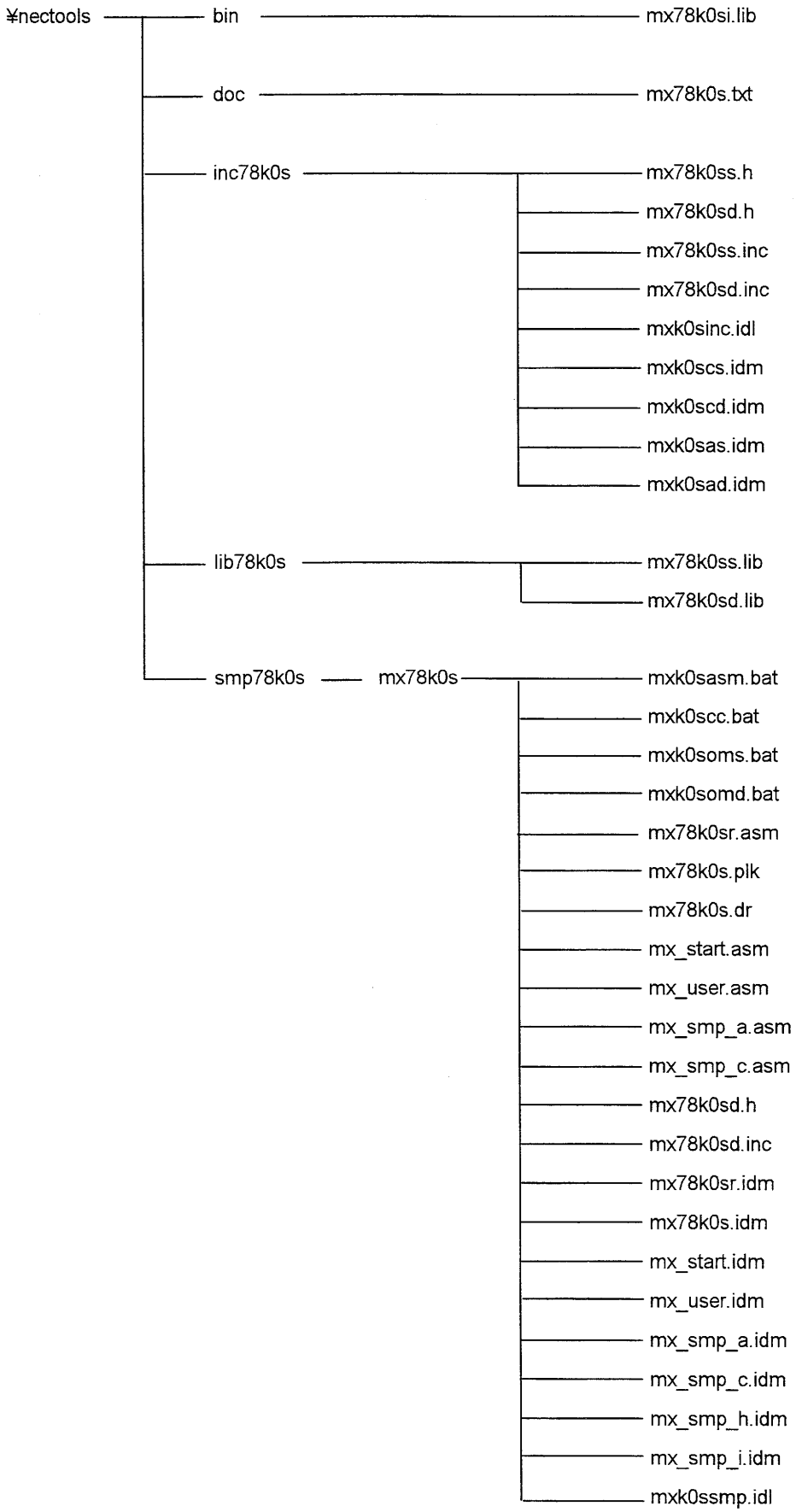
この章では，本OSのホスト・マシンへのインストールについて述べます。

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。次にオブジェクト・ファイル形式とソース・ファイル形式のインストール後の提供ディレクトリ構成を示します。英語版と日本語版のディレクトリ構成は同じです。

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

ヘルプ・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	mx78k0si.lib
内容	idea-L 用関数ライブラリ・ファイル（アセンブリ言語用）
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　ユーザーズ・マニュアル**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，注意事項などが書かれたファイルが入っています。

ファイル名	mx78k0s.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，注意事項などが書かれているテキスト・ファイルです。

(3) ¥nectools¥inc78k0s 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	mx78k0ss.h，mxk0scs.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	定数の定義とシステム・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，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	mx78k0sd.h，mxk0scd.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	定数および例外エラー・コードの定義とシステム・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，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	mx78k0ss.inc，mxk0sas.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	定数の定義とシステム・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，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	mx78k0sd.inc，mxk0sad.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	定数および例外エラー・コードの定義とシステム・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，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	mxk0sinc.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　ユーザーズ・マニュアル** を参照してください。

(4) ¥nectools¥lib78k0s ディレクトリ

本 OS のライブラリ・ファイルが入っています。

ファイル名	mx78k0ss.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	mx78k0sd.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¥smp78k0s¥mx78k0s 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	mxk0sasm.bat
内容	ユーザ・タスク，メモリ設定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，メモリ設定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**　mxk0sasm△9011△usr1　　（ターゲット CPU が μ PD789011，ファイル名が usr1.asm）

ファイル名	mxk0scc.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**　mxk0scc△9011△usr2　　（ターゲット CPU が μ PD789011，ファイル名が usr2.c）

ファイル名	mxk0soms.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	“mx78k0ss.lib”とその他の必要なファイル（”mx78k0s.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。

ファイル名	mxk0somd.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	“mx78k0sd.lib”とその他の必要なファイル（”mx78k0s.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。

ファイル名	mx78k0sr.asm，mx78k0sr.idm（idea-L 用）
内容	メモリ設定ファイルのサンプル・ファイル
使用方法	本 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，同梱のサンプル・プログラム用に記述されていますので，ユーザ・システムを構築する際に参考にしてください。

ファイル名　mx78k0s.dr，mx78k0s.idm（idea-L 用）
内容　リンク・ディレクティブ・ファイルのサンプル・ファイル
使用方法　本 OS が使用するメモリの配置を設定するファイルです。本ファイルは，同梱のサンプル・プログラム用に記述されていますので，ユーザ・システムを構築する際に参考にしてください。

ファイル名　mx78k0s.plk
内容　リンカ・パラメータ・ファイルのサンプル・ファイル
使用方法　“mxk0soms.bat”,”mxk0somd.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。本ファイルは，同梱のサンプル・プログラム用に記述されていますので，ユーザ・システムを構築する際に参考にしてください。

ファイル名　mx_start.asm，mx_start.idm（idea-L 用）
内容　スタート・アップ・ルーチンのサンプル・ファイル
使用方法　システム初期化，本 OS 起動のためのスタート・アップ・ルーチンです。本ファイルは，同梱のサンプル・プログラム用に記述されていますので，ユーザ・システムを構築する際に参考にしてください。

ファイル名　mx_user.asm，mx_user.idm（idea-L 用）
内容　ユーザー・オウン・コーディング部のサンプル・ファイル
使用方法　レディ・キューが空のときの処理ルーチンおよび本 OS のタイマ割り込み許可／禁止ルーチンが入っています。本ファイルは，同梱のサンプル・プログラム用に記述されていますので，ユーザ・システムを構築する際に参考にしてください。

ファイル名　mx_smp_a.asm，mx_smp_a.idm（idea-L 用）
内容　サンプル・プログラム（アセンブリ言語）
使用方法　サンプル・プログラムのタスクがアセンブリ言語で記述されています。ユーザ・システムを構築する際に参考にしてください。

ファイル名　mx_smp_c.asm，mx_smp_c.idm（idea-L 用）
内容　サンプル・プログラム（C 言語）
使用方法　サンプル・プログラムのタスクが C 言語で記述されています。ユーザ・システムを構築する際に参考にしてください。

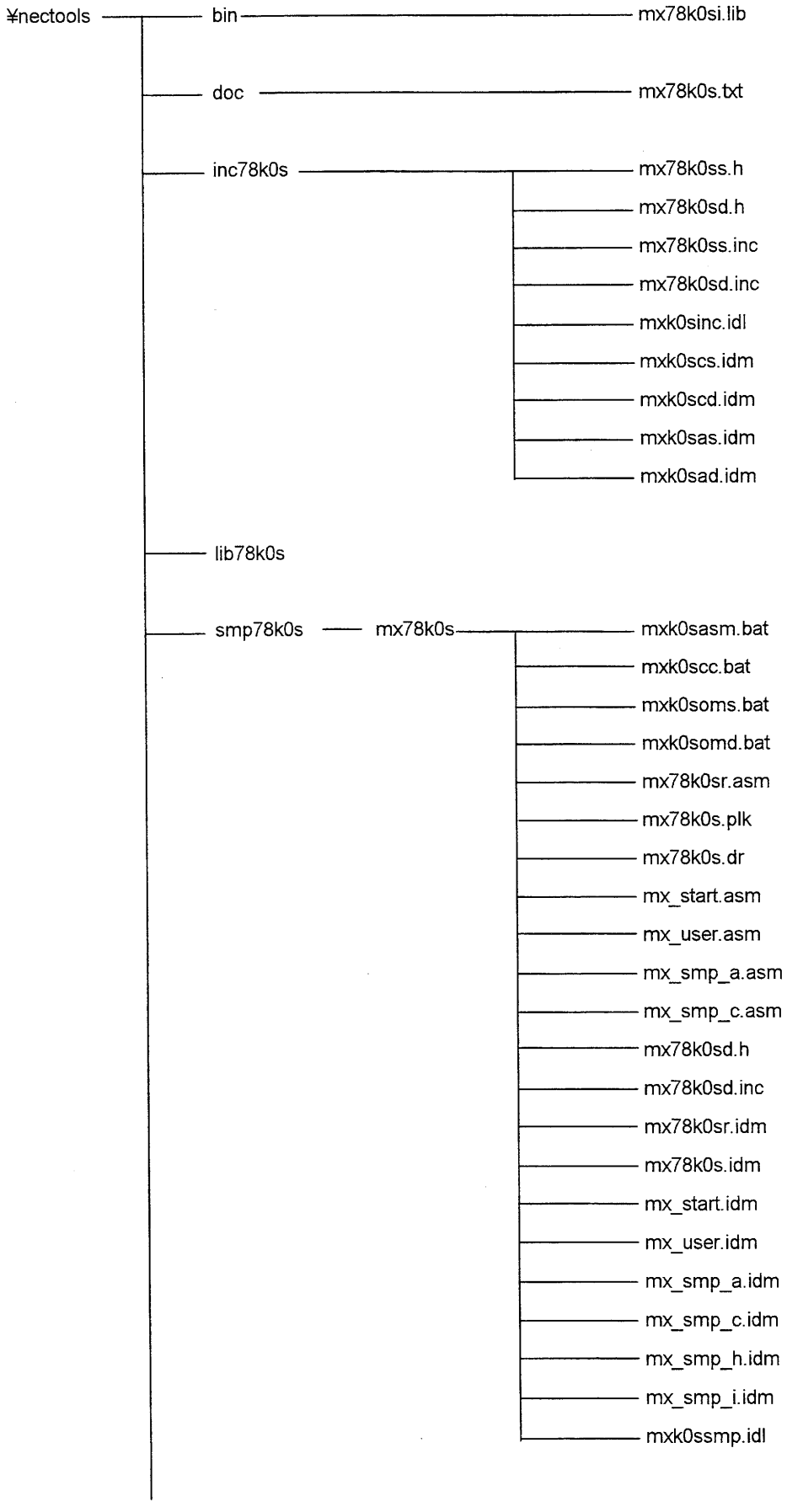
ファイル名　mx78k0sd.h，mx_smp_h.idm（idea-L 用）
内容　サンプル・プログラム用の MX78K0S インクルード・ファイル（C 言語）
使用方法　サンプル・プログラムのメイクに使用します。本ファイルは，サンプル・プログラム（C 言語）で使用するシステム・コールだけに限定して記述されたインクルード・ファイルです。

ファイル名　mx78k0sd.inc，mx_smp_i.idm（idea-L 用）
内容　サンプル・プログラム用の MX78K0S インクルード・ファイル（アセンブリ言語）
使用方法　サンプル・プログラムのメイクに使用します。本ファイルは，サンプル・プログラム（アセンブリ言語）で使用するシステム・コールだけに限定して記述されたインクルード・ファイルです。

ファイル名　mxk0ssmp.idl
内容　サンプル用 idea-L 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　ユーザーズ・マニュアル**を参照してください。

2.1.2　ソース・ファイル形式のディレクトリ構成

図 2−2　ソース・ファイル形式のディレクトリ構成（1/3）



保守／廃止

図 2−2　ソース・ファイル形式のディレクトリ構成（2/3）

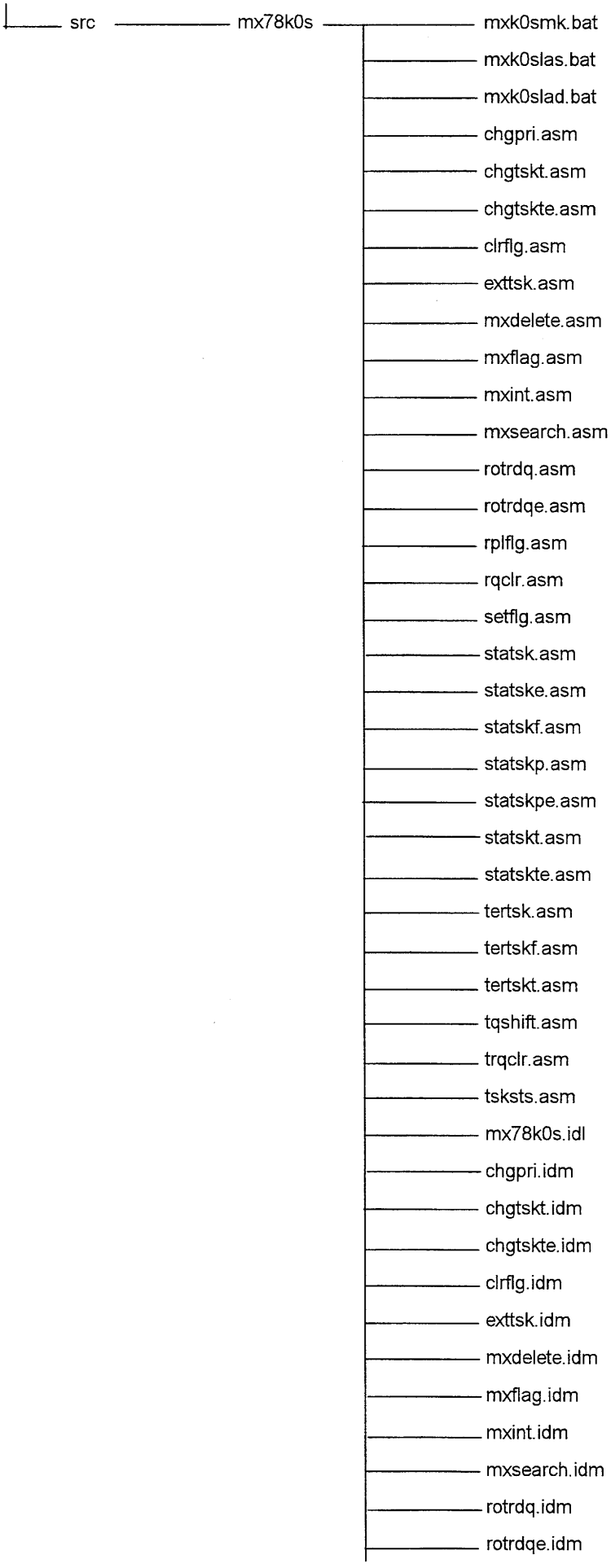
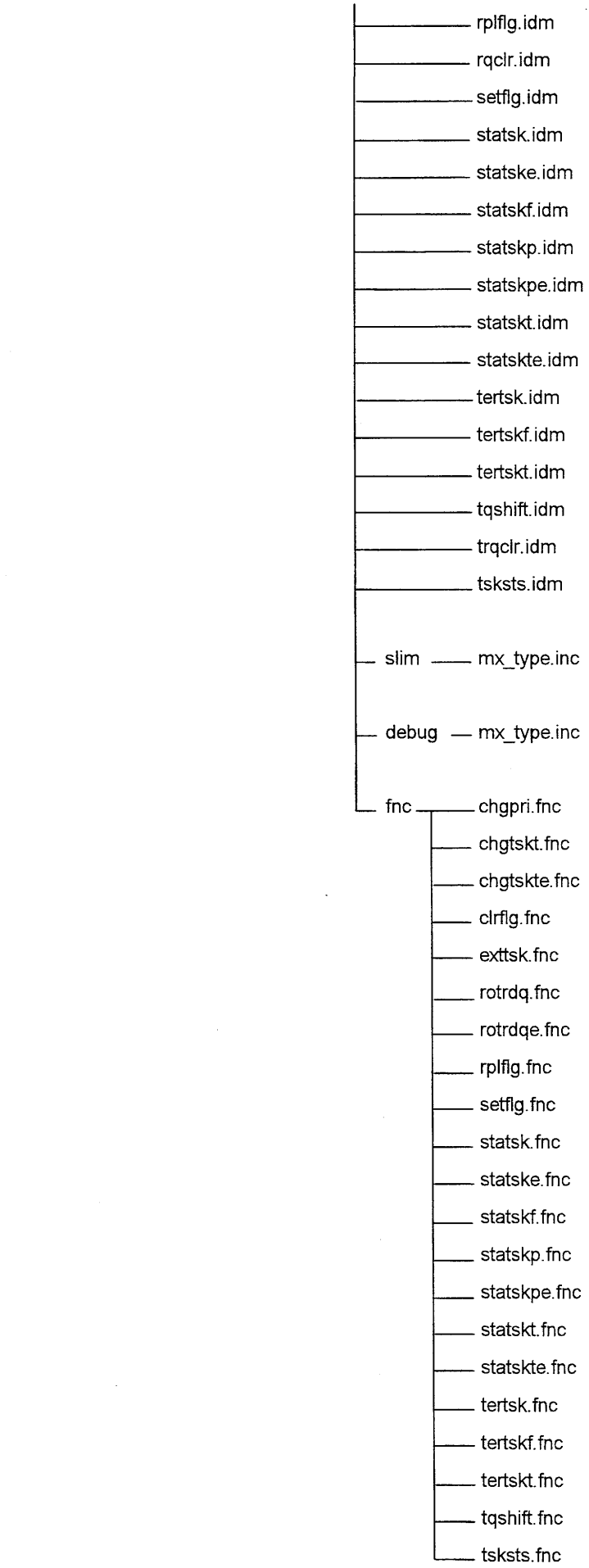


図 2—2　ソース・ファイル形式のディレクトリ構成（3/3）



保守／廃止

(1) ¥nectools¥bin ディレクトリ

ヘルプ・ファイル（idea-L 用関数ライブラリ・ファイル）が入っています。

ファイル名	mx78k0si.lib
内容	MX78K0S の idea-L 用関数ライブラリ・ファイル（アセンブリ言語用）
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などが書かれたファイルが入っています。

ファイル名	mx78k0s.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などが書かれている，テキスト・ファイルです。

(3) ¥nectools¥inc78k0s 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	mx78k0ss.h，mxk0scs.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	定数の定義とシステム・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，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	mx78k0sd.h，mxk0scd.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	定数および例外エラーコードの定義とシステム・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，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	mx78k0ss.inc，mxk0sas.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	定数の定義とシステム・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，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	mx78k0sd.inc，mxk0sad.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	定数および例外エラーコードの定義とシステム・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，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	mxk0sinc.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　ユーザーズ・マニュアル**を参照してください。

(4) ¥nectools¥lib78k0s 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。

(5) ¥nectools¥smp78k0s¥mx78k0s 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　mxk0sasm.bat
内容　ユーザ・タスク，メモリ設定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，メモリ設定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 mxk0sasm△9011△usr1　（ターゲット CPU がμ PD789011，ファイル名が usr1.asm）

ファイル名　mxk0scc.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 mxk0scc△9011△usr2　（ターゲット CPU がμ PD789011，ファイル名が usr2.c）

ファイル名　mxk0soms.bat
内容　スリム・タイプのロード・モジュール“mx78k0ss.lib”を作成する際に使用するバッチ・ファイル
使用方法　“mx78k0ss.lib”とその他の必要なファイル（”mx78k0s.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。

ファイル名　mxk0somd.bat
内容　ディバグ・タイプのロード・モジュール“mx78k0sd.lib”を作成する際に使用するバッチ・ファイル
使用方法　“mx78k0sd.lib”とその他の必要なファイル（”mx78k0s.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。

ファイル名　mx78k0sr.asm，mx78k0sr.idm（idea-L 用）
内容　メモリ設定ファイルのサンプル・ファイル
使用方法　本 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，同梱のサンプル・プログラム用に記述されていますので，ユーザ・システムを構築する際に参考にしてください。

ファイル名　mx78k0s.dr，mx78k0s.idm（idea-L 用）
内容　リンク・ディレクティブ・ファイルのサンプル・ファイル
使用方法　本 OS が使用するメモリの配置を設定するファイルです。本ファイルは，同梱のサンプル・プログラム用に記述されていますので，ユーザ・システムを構築する際に参考にしてください。

保守／廃止

ファイル名	mx78k0s.plk
内容	リンカ・パラメータ・ファイルのサンプル・ファイル
使用方法	“mxk0soms.bat”,”mxk0somd.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。本ファイルは，同梱のサンプル・プログラム用に記述されていますので，ユーザ・システムを構築する際に参考にしてください。

ファイル名	mx_start.asm，mx_start.idm（idea-L 用）
内容	スタート・アップ・ルーチンのサンプル・ファイル
使用方法	システム初期化，本 OS 起動のためのスタート・アップ・ルーチンです。本ファイルは，同梱のサンプル・プログラム用に記述されていますので，ユーザ・システムを構築する際に参考にしてください。

ファイル名	mx_user.asm，mx_user.idm（idea-L 用）
内容	ユーザー・オウン・コーディング部のサンプル・ファイル
使用方法	レディ・キューが空のときの処理ルーチンおよび本 OS のタイマ割込み許可／禁止ルーチンが入っています。本ファイルは，同梱のサンプル・プログラム用に記述されていますので，ユーザ・システムを構築する際に参考にしてください。

ファイル名	mx_smp_a.asm，mx_smp_a.idm（idea-L 用）
内容	サンプル・プログラム（アセンブリ言語）
使用方法	サンプル・プログラムのタスクがアセンブリ言語で記述されています。ユーザ・システムを構築する際に参考にしてください。

ファイル名	mx_smp_c.asm，mx_smp_c.idm（idea-L 用）
内容	サンプル・プログラム（C 言語）
使用方法	サンプル・プログラムのタスクが C 言語で記述されています。ユーザ・システムを構築する際に参考にしてください。

ファイル名	mx78k0sd.h，mx_smp_h.idm（idea-L 用）
内容	サンプル・プログラム用の MX78K0S インクルード・ファイル（C 言語）サンプル・プログラムのメイクに使用します。
使用方法	本ファイルは，サンプル・プログラム（C 言語）で使用するシステム・コールだけに限定して記述されたインクルード・ファイルです。

ファイル名	mx78k0sd.inc，mx_smp_i.idm（idea-L 用）
内容	サンプル・プログラム用の MX78K0S インクルード・ファイル（アセンブリ言語）サンプル・プログラムのメイクに使用します。
使用方法	本ファイルは，サンプル・プログラム（アセンブリ言語）で使用するシステム・コールだけに限定して記述されたインクルード・ファイルです。

ファイル名	mxk0ssmp.idl
内容	サンプル用 idea-L 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　ユーザーズ・マニュアル**を参照してください。

保守／廃止

(6) ¥nectools¥src¥mx78k0s ディレクトリ

本ディレクトリには，OS のソース・ファイルおよびライブラリ構築用のバッチ・ファイルが入っています。ソース・ファイルはスリム・タイプとディバグ・タイプ共通で，タイプ固有の機能についてはソース・ファイル中の$if 制御で区別しています。

ファイル名	mxk0smk.bat
内容	MX78K0S のライブラリ・ファイルを構築する際に使用するバッチ・ファイル
使用方法	本ファイルを使用するときは，"¥nectools¥src¥mx78k0s"のディレクトリで実行し，引数としてチップ種別とタイプ名（スリム・タイプの場合は"s"，ディバグ・タイプの場合は"d"）を指定してください。 **例** mxk0smk△9011△s　（ターゲット CPU がμ PD789011 でスリム・タイプのライブラリを構築する場合）

ファイル名	mxk0slas.bat
内容	mxk0smk.bat から呼び出されるスリム・タイプをアセンブルするバッチ・ファイル
使用方法	本ファイルは通常，ユーザは使用しません。

ファイル名	mxk0slad.bat
内容	mxk0smk.bat から呼び出されるディバグ・タイプをアセンブルするバッチ・ファイル
使用方法	本ファイルは通常，ユーザは使用しません。

ファイル名	*.asm，*.idm（idea-L 用）
内容	各システム・コールおよびサブルーチンのソース・ファイルです。 各ファイルの内容の詳細は，それぞれのソース・ファイルを参照してください。

ファイル名	mx78k0s.idl
内容	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　ユーザーズ・マニュアル**を参照してください。

(7) ¥nectools¥src¥mx78k0s¥slim ディレクトリ

スリム・タイプ用の定義ファイルが入っています。

ファイル名	mx_type.inc
内容	$if 制御用の命令およびスリム・タイプ用の定数の定義が記述されています。

(8) ¥nectools¥src¥mx78k0s¥debug ディレクトリ

ディバグ・タイプ用の定義ファイルが入っています。

ファイル名	mx_type.inc
内容	$if 制御用の命令およびディバグ・タイプ用の定数の定義が記述されています。

(9) ¥nectools¥src¥mx78k0s¥fnc ディレクトリ

idea-L 用関数ライブラリ・ファイル（mx78k0si.lib）作成用の関数ファイルが入っています。

ファイル名	*.fnc
内容	idea-L 用関数ライブラリ・ファイル作成用関数ファイル
使用方法	関数ライブラリの作成方法は **idea-L　スクリーン・エディタ　入門編　ユーザーズ・マニュアル**を参照してください。

保守／廃止

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。

インストール方法には，次の２通りがあります。

・Windows 上で setup.exe を起動する。

・DOS 上で dosinst.bat を起動する。

2.2.1　Windows 上でインストールを行う場合

MX78K0S（日本語版）をドライブＣから読み込み，ドライブＡへインストールする場合について記述します。Windows はすでに起動しているものとします。

＜実行例＞

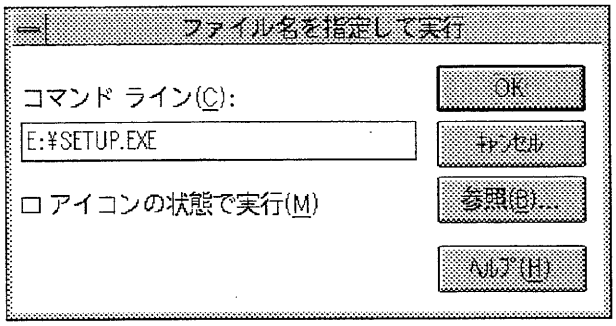
(1)　インストーラを起動します。

①　“MX78K0S(SRC) DISK #1” をＣドライブにセットします。

注意　オブジェクト版は “MX78K0S(OBJ) DISK #1” となります。

②　プログラム・マネージャの［アイコン］－［ファイル名を指定して実行］メニューを選択します。

③　コマンド・ライン入力欄に次のように入力します。

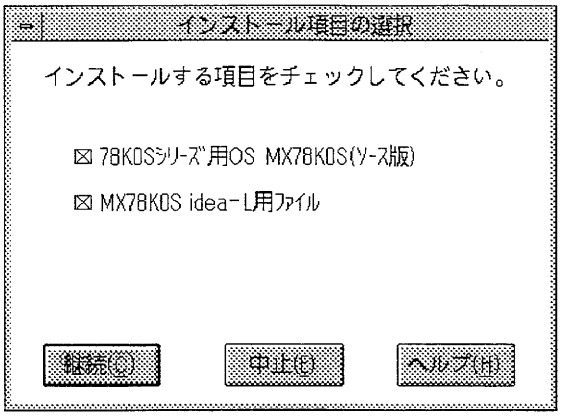


④　“OK” を選択します。

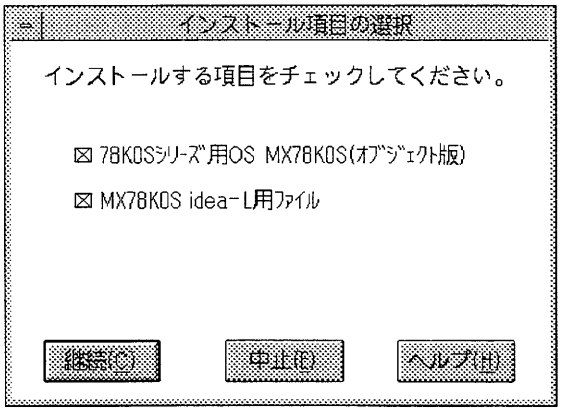
セットアップの初期化処理のあと，インストーラが起動します。

(2)　インストール項目が表示されます。

(a)　ソース版の場合



(b)　オブジェクト版の場合



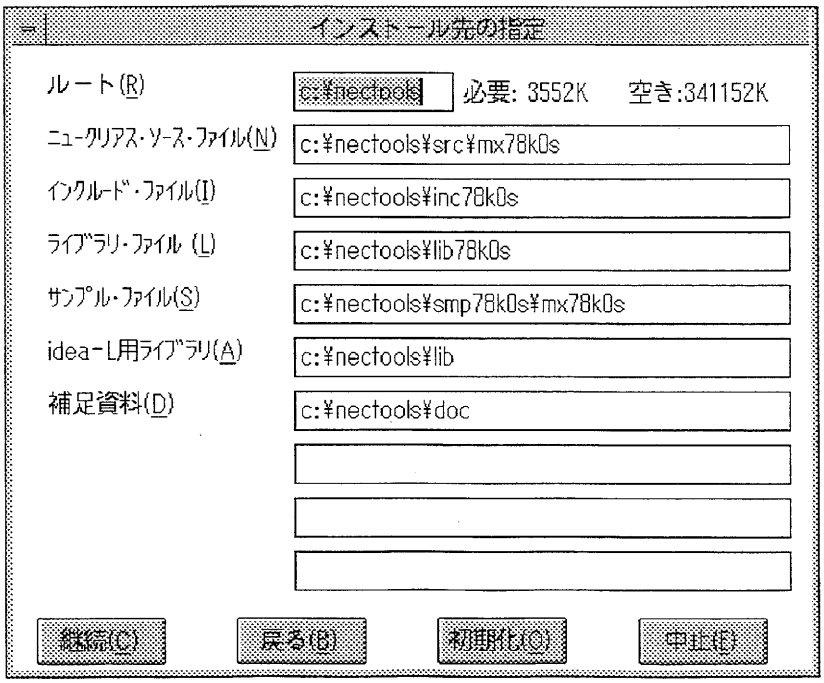
①　インストールを継続する場合は“継続”を選択してください。

②　インストールを中止する場合は“中止”を選択してください。

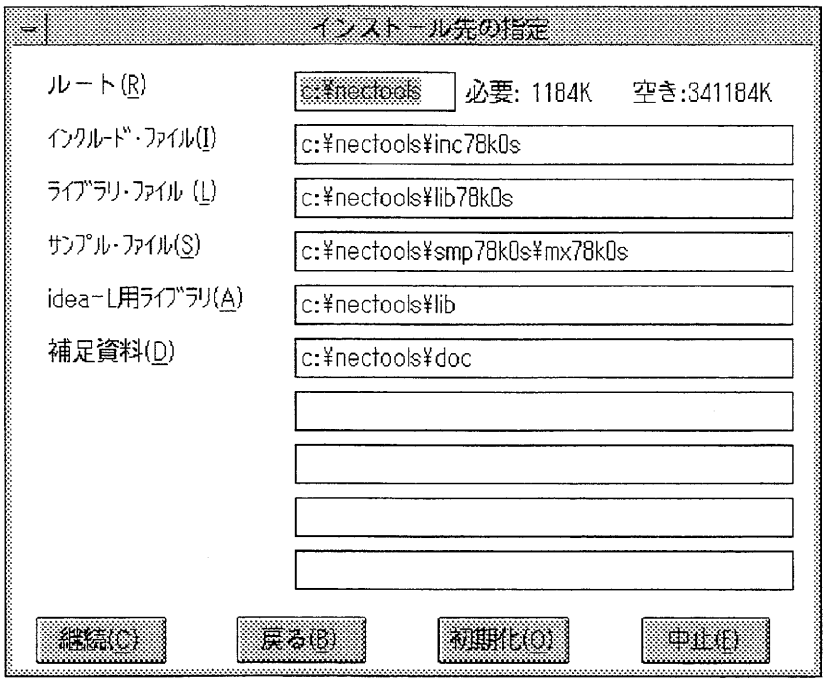
(3) インストール・ディレクトリを指定します。

①　インストール先の指定ダイアログ・ボックスが表示されます。

(a)　ソース版の場合



(b)　オブジェクト版の場合



② インストールするディレクトリをテキスト・ボックスに入力したあと，“継続”を選択します。

③ “戻る”を選択すると，インストール項目の表示に戻ります。

④ “初期化”を選択すると，指定ディレクトリがディフォールトのディレクトリとなります。
インストール先のルートのディフォールトは，Windows がインストールされているドライブの¥nectools となります。すでにインストーラで他の NEC ツールがインストールされている場合は，そこのルートを採用します。ルートをエディットすると，それ以下のディレクトリも連動して変更されます。

⑤ “中止”を選択すると，インストールを中止します。

⑥ “ニュークリアス・ソース・ファイル”は，システム・コールのソースおよび idea-L 関数ファイルを指します。

注意　オブジェクト版ではこの項目は表示されません。英語版では“Nucleus Source Files”と表示します。

⑦ “インクルード・ファイル”はアセンブリ言語用およびC言語用のインクルード・ファイルを指します。

注意　英語版では“Include Files”と表示します。

⑧ “ライブラリ・ファイル”はユーザが構築するライブラリを指します。

注意　オブジェクト版では提供ライブラリ・ファイルを指します。英語版では“Library Files”と表示します。

⑨　“サンプル・ファイル”はメモリ設定ファイルやリンク・ディレクティブ・ファイル，ユーザ・タスクなどのサンプルとして提供するファイルを指します。

注意　英語版では“Sample Files”と表示します。

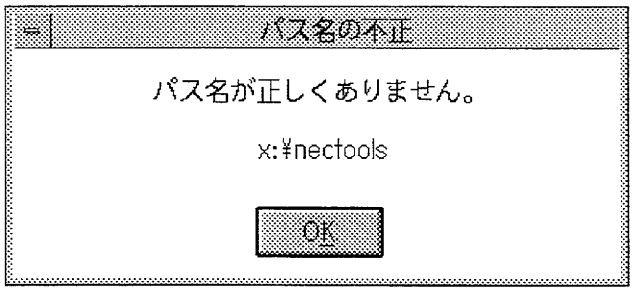
⑩　“idea-L 用ライブラリ・ファイル”は idea-L から使用できるオンライン・ヘルプです。

注意　英語版では“Help File”と表示します。

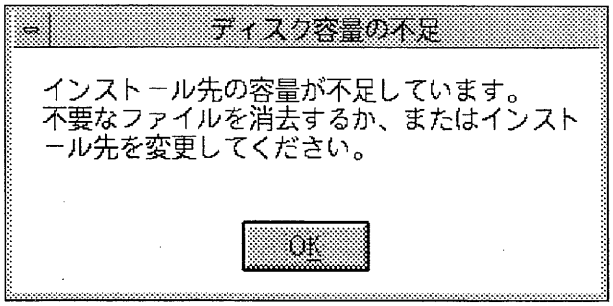
⑪　“補足資料”はマニュアルに記載されていない事項について書かれたテキスト・ファイルを指します。

注意　英語版では“Release Document”と表示します。

⑫　指定したディレクトリが不正の場合はエラーとなり，次のメッセージを表示します。

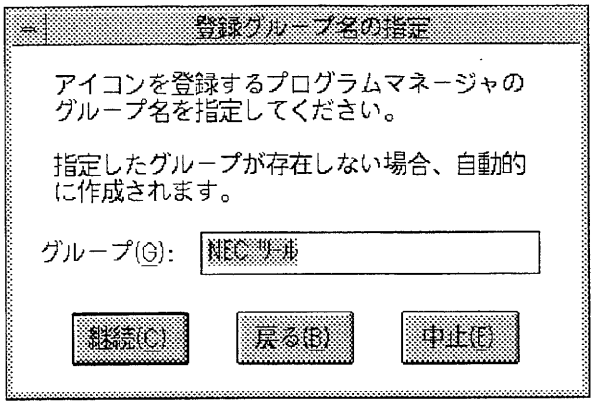


⑬　容量が不足しているときはエラーとなり，次のメッセージを表示します。



(4)　プログラム・マネージャに登録するグループを指定します。

①　登録グループ名の指定ダイアログ・ボックスが表示されます。



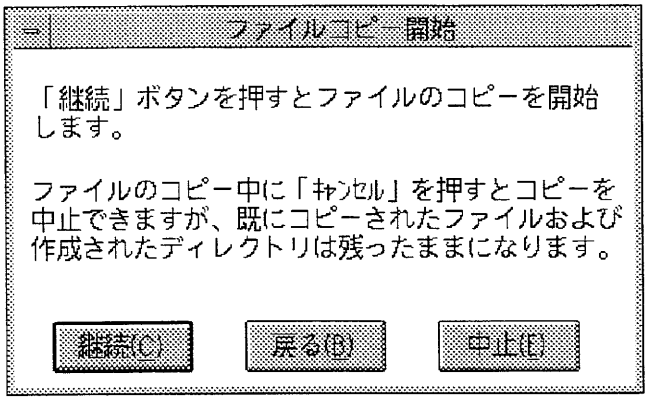
②　登録するグループ名をテキスト・ボックスに入力したあと，“継続”を選択します。
指定したグループが存在しない場合，そのグループが新規に作成されます。また，指定したグループがすでにインストーラを使用して登録してある場合，そのグループを使用します。

③　“戻る”を選択すると，インストール先の指定ダイアログに戻ります。

④　“中止”を選択すると，インストールを中止します。

(5)　ファイル・コピーを開始します。

①　ファイル・コピー開始ダイアログが表示されます。



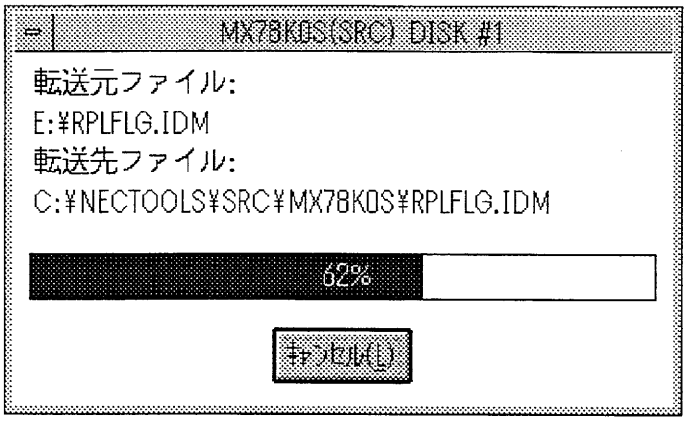
②　“継続”を選択すると，ファイルのコピーを開始します。

③　“戻る”を選択すると，登録グループ名の指定ダイアログに戻ります。

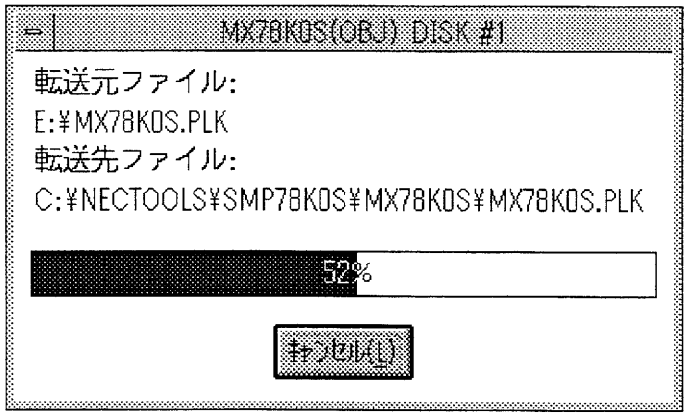
④　“中止”を選択すると，インストールを中止します。

(6)　ファイルのコピーが開始されます。

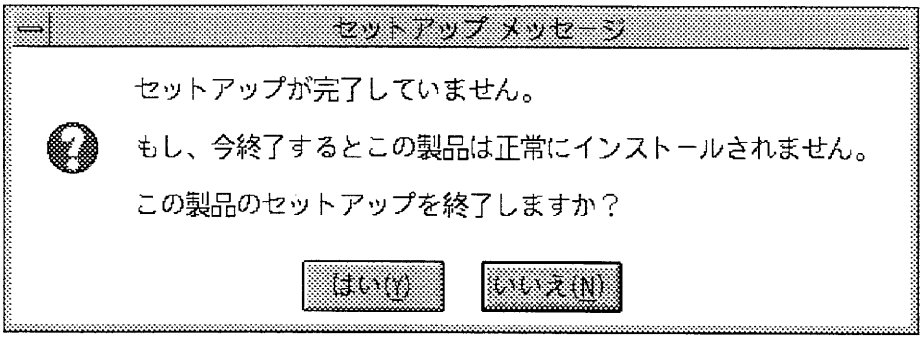
(a)　ソース版の場合



(b)　オブジェクト版の場合



備考　“キャンセル”を選択すると，次のメッセージを表示します。



①　“はい”を選択すると，インストールを中止します。

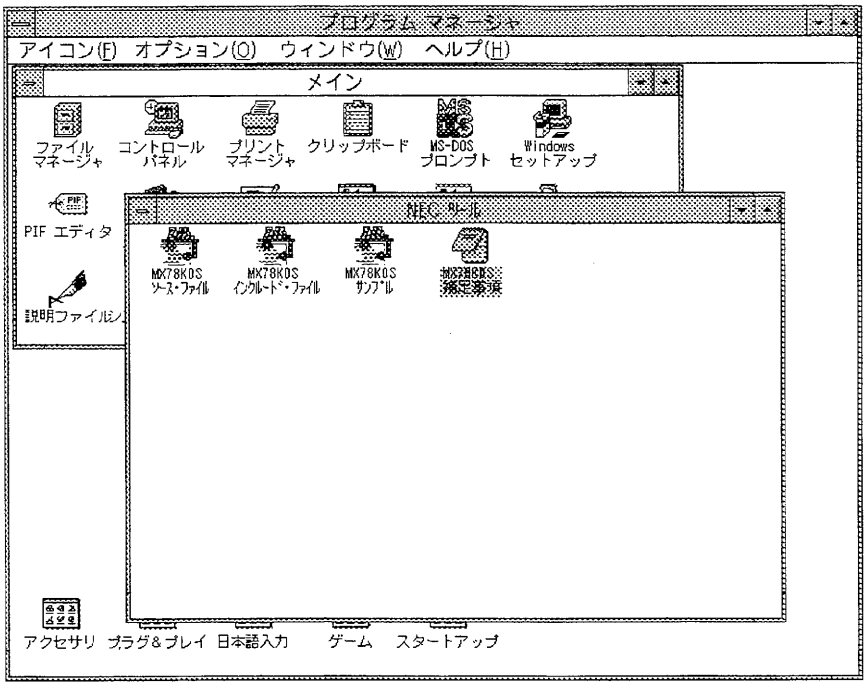
②　“いいえ”を選択すると，ファイル・コピーを再開します。

保守／廃止

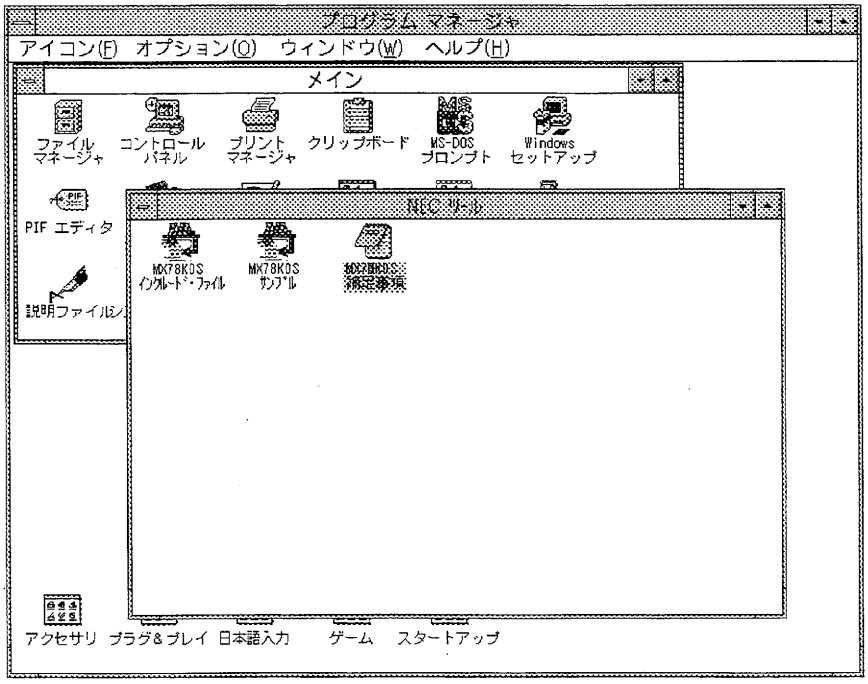
(7)　登録したグループとアイコンが作成されます。

登録したグループに，idea-L 用ファイル（拡張子.idl）と補足事項のアイコンが登録されます。

(a)　ソース版の場合



(b)　オブジェクト版の場合

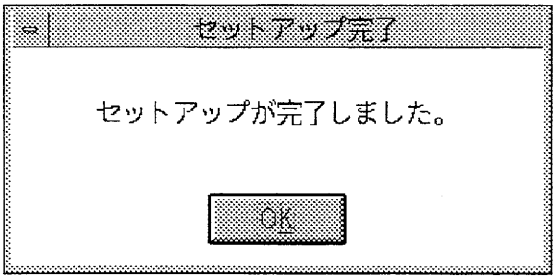


注意1.　オブジェクト版では“MX78K0S ソース・ファイル”アイコンは登録されません。

2.　idea-L または idea-L　Light をインストールしていない場合は idea-L 用ファイルのアイコンは上記とは異なります。また，これらのアイコンからファイルを開くことができませんので削除してください。

(8)　インストールを終了します。

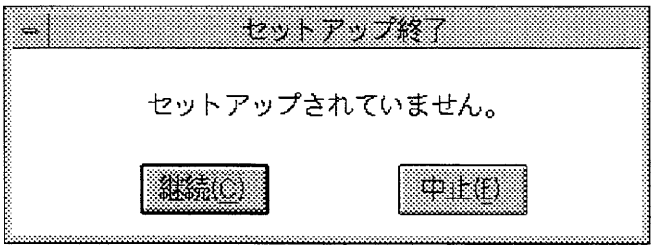
①　次のメッセージを出力します。



②　“OK”を選択するとインストーラが終了します。

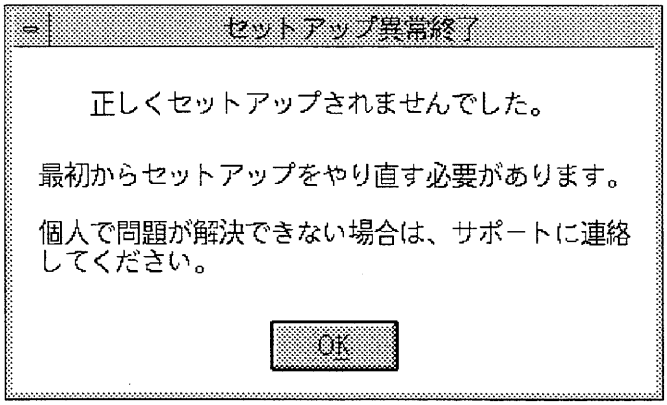
備考　インストーラを中止した場合は次のようになります。

①　次のメッセージを出力します。



②　“継続”を選択すると，その直前のダイアログに戻ります。

③　“中止”を選択すると，次のメッセージを表示します。



④　“OK”を選択すると，インストーラを終了します。

保守／廃止

2.2.2　DOS 上でインストールを行う場合

DOS 上でインストールを行う場合は，提供媒体中の“dosinst.bat”を使用します。本バッチ・ファイルを使用する場合は第 1 パラメータに提供媒体をセットしたドライブ名を，第 2 パラメータにインストール先のドライブ名を指定してください。第 3 パラメータにはインストール先ディレクトリを指定します。第3パラメータを省略すると，¥nectools が指定されたものとしてインストール作業を行います。

例 C ドライブの提供媒体から，A ドライブの ¥nectools にインストールする場合

```
A>  c:dosinst △ c: △ a: △ nectools↓
```

備考 1　入力例中の“A>”はプロンプトを，“△”は空白（スペース）の入力を，“↓”はリターン・キーの入力を表しています。

備考 2　“dosinst.bat”では，idea-L 用ファイルはインストールされません。idea-L 用ファイルをインストールしたい場合は，Windows でインストールを行うか，別途“dosinst.bat”を実行してください。必要なパラメータは“dosinst.bat”と同一です。

注意　**DOS 用インストール・プログラムではインストール先ドライブの空き容量チェックを行いませんので，インストール先には十分な空き容量のあるドライブを指定してください。**

2.3　インストール時の注意事項

インストール時の注意事項や製品の最新情報などが提供媒体中の”readme.txt”に書かれていますので，インストール作業を始める前に目を通しておいてください。

第3章　タスク管理

タスクとは，OS 上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 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。

優先順位に基づいて処理されるRX78Kシリーズとは異なり，本OSでは，レディ・キューに登録さ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()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，初期化処理中において最初にsta_tsk()システム・コールをかけられたタスクです。また，初期化処理中にsta_tsk()システム・コール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・キューに登録されています。初期化処理中にsta_tsk()システム・コール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()システム・コールなど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（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）。

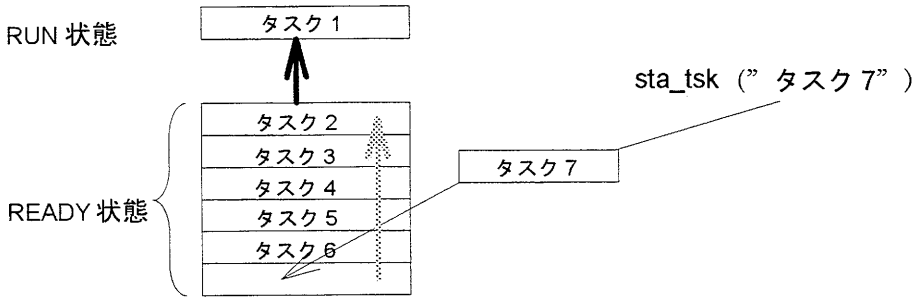
タスクの起動を行うシステム・コールは，sta_tsk()システム・コールの他に，現在動作しているタスクの次に指定タスクが動作するようにレディ・キューの先頭に登録するsta_tskp()システム・コール，指定時間経過後に指定タスクの起動を行なうsta_tskt()システム・コール，指定イベント・フラグのビット・パターンによって指定タスクの起動を行なうsta_tskf()システム・コールがあります。

それらのシステム・コールについて次に説明します。

3. 2. 1 sta_tsk()の仕組み

sta_tsk()システム・コールは，指定されたタスクをレディ・キューの最後尾に登録する処理を行ないます。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。

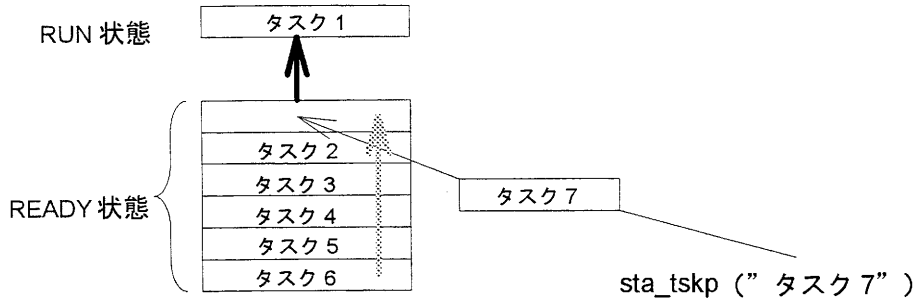
図 3－1 sta_tsk()の説明図



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動作しているタスクの次に起動するようレディ・キューの先頭に登録します。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



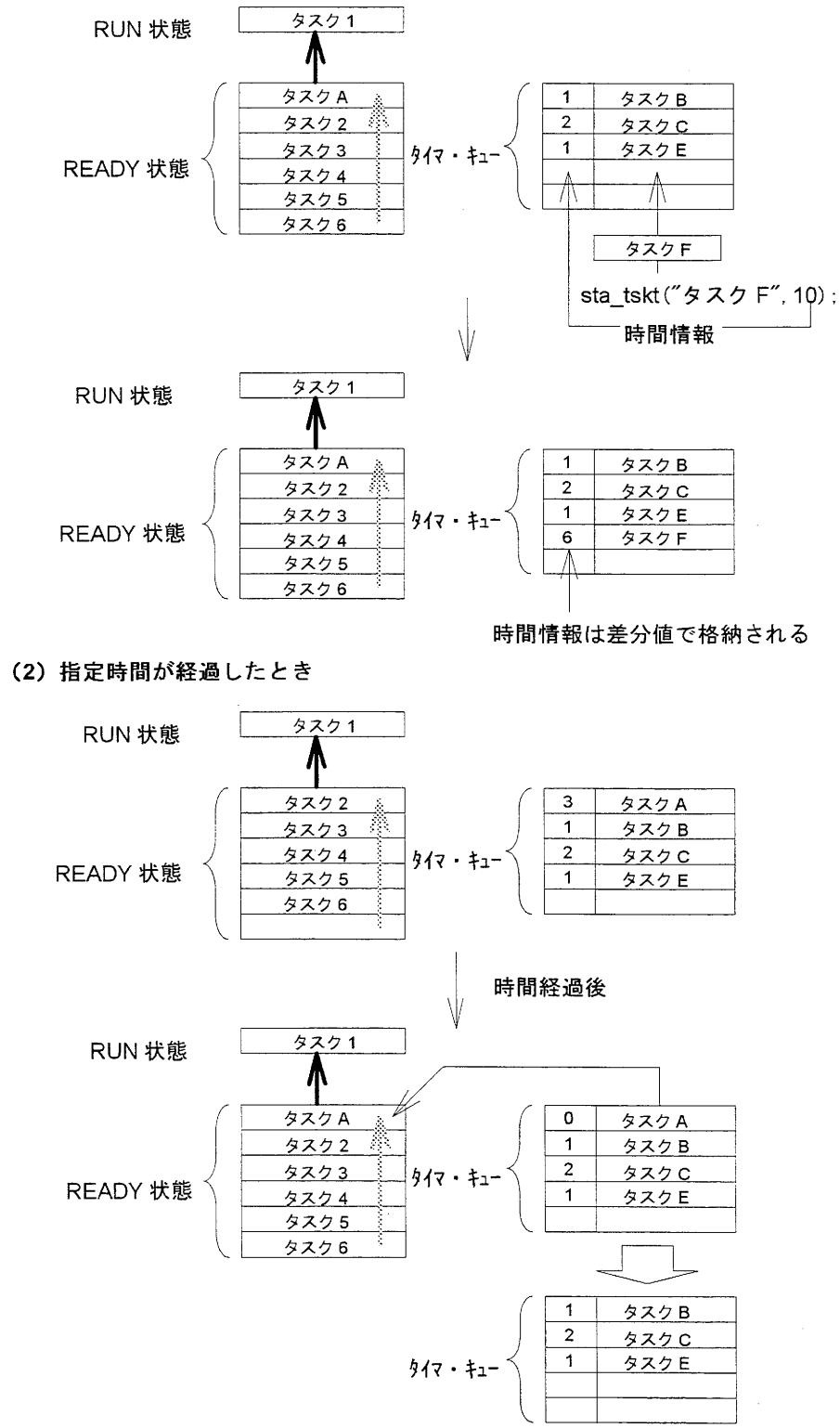
保守／廃止

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・キューに登録されます。指定時間が経過すると，指定したタスクはレディ・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。

図 3−3 sta_tskt()の説明図



保守／廃止

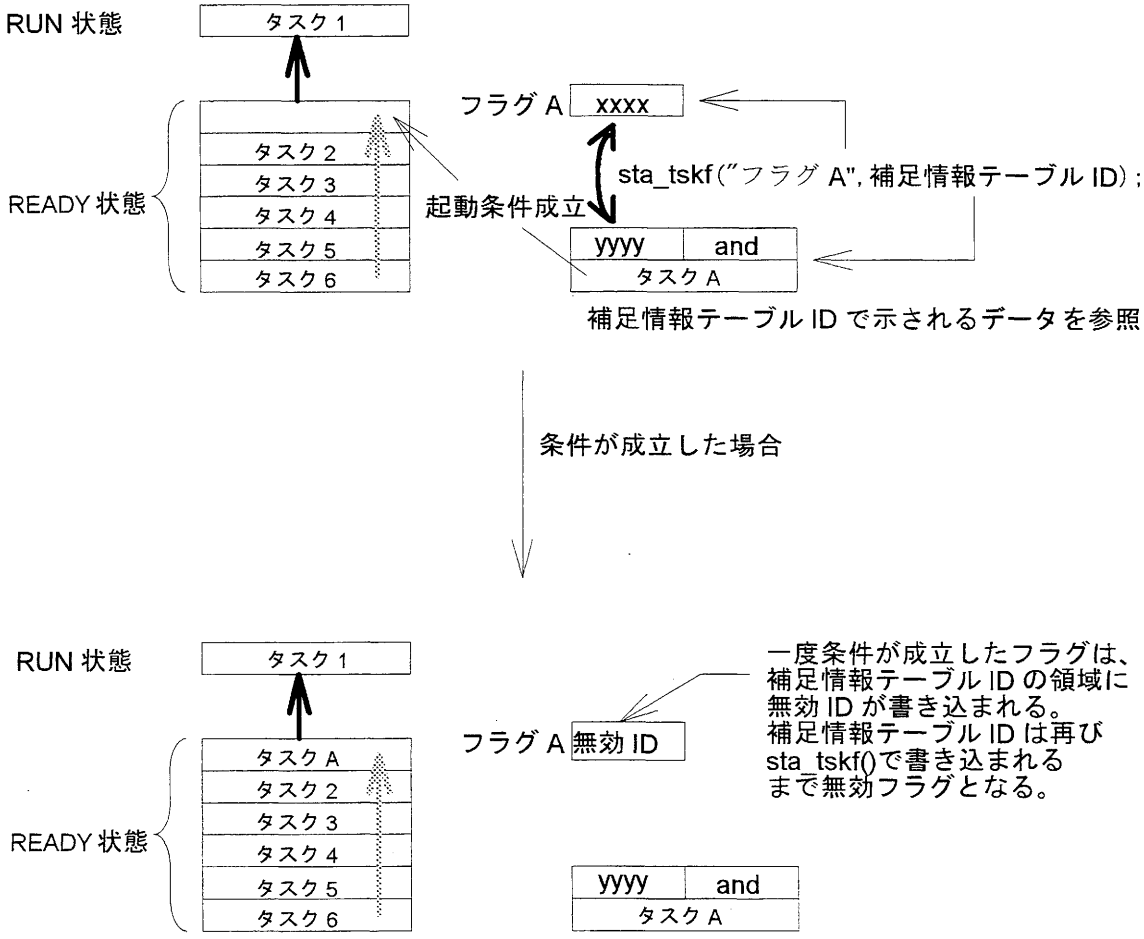
3. 2. 4　sta_tskf()の仕組み

sta_tskf()システム・コールは，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動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指定タスクをレディ・キューの先頭に登録します。そして，指定されたタスクは，現在動作し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，フラグの内容はすべてクリアされ，次にsta_tskf()システム・コールで指定されるまで無効になります。

ただし，ビット・パターン一致によってタスクを起動するのは，set_flg()とrpl_flg()システム・コールによってビット・パターンがセットされた場合だけで，clr_flg()システム・コールによってビット・パターンの一致が起こってもタスクは起動しません。

図3－4 sta_tskf()の説明図



補足情報テーブルとイベント・フラグは使用されたあとも，領域は開放されない。

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い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 satus;
        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。

保守／廃止

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，次の3種類です。

(1) 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。

(2) READY 状態（実行可能状態）

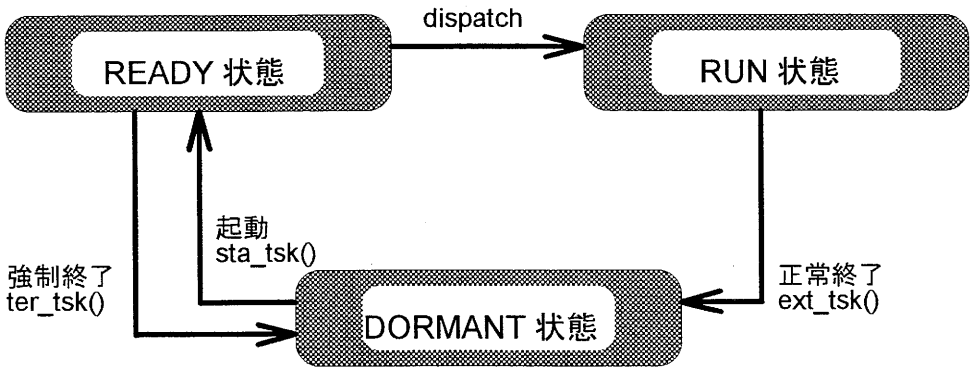
レディ・キューに登録されている状態で，実行の順番を待っている状態です。RUN 状態のタスクが終了すると，レディ・キューの先頭のタスクが RUN 状態に遷移します。

(3) 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，まだイベント・フラグの条件が成立していないタスクの状態は DORMANT 状態となります。

図 3−5 タスク状態遷移図

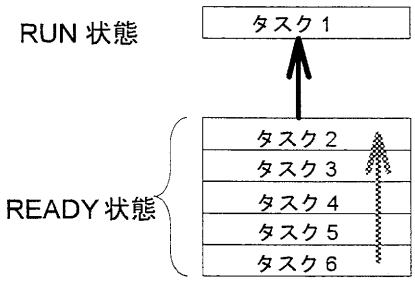


タスクの管理は，次のようになります。

(1) 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 状態のタスクのうち，レディ・キューの先頭に登録されているタスクです。RUN 状態のタスクは，レディ・キューから削除されます。

図 3—6 READY 状態→RUN 状態のタスクの図



(2) 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。

(3) 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，本 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。

保守／廃止

3.5 タスクの終了

RUN状態またはREADY状態のタスクが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・コールには次のものがあります。

・自タスク(RUN状態のタスク)を終了させる機能を持つシステム・コール

ext_tsk(), sta_tske(), sta_tskpe(), sta_tskte(), chg_tskte(), rot_rdqe()

・指定タスク(READY状態のタスク)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。

RUN 状態のタスクが終了すると，次に，レディ･キューの先頭に登録されているタスクが RUN 状態になります。レディ･キューが空の場合は，アイドル・ルーチン（_MX_IDle）を実行し，レディ・キューにタスクが登録されるのを待ちます。

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。複数のイベント・フラグを設定する場合は，RAM 上に連続して配置する必要があります。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。この連続領域全体を「イベント・フラグ本体テーブル」といいます。

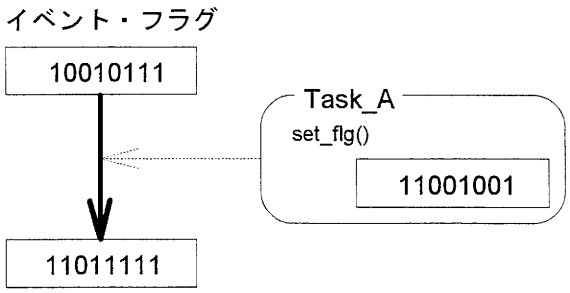
また，イベント・フラグを使用するためには，イベント・フラグによって起動するタスク，タスクを起動するためのイベント・フラグのビット･パターン，ビット・パターンが設定されたときのタスクの起動条件(AND または OR)の 3 つの情報が必要です。これら 3 つの情報をまとめて「補足情報」といいます。補足情報はイベント・フラグと同様に，複数設定することができますが，その際は，ROM/RAM 上のどちらかに連続して配置してください。この連続した領域を「イベント・フラグ補足情報テーブル」といい，個々の補足情報には，先頭から順に 1,2,3,･･･の順で ID 番号が割り当てられます。補足情報 ID 番号の最大値は 0xfe です。

4.1　イベント・フラグのセットとクリア

本OSでは，set_flg()およびrpl_flg()システム・コールによってイベント・フラグをセットします。

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 set_flg()システム・コールのイベント・フラグのセット



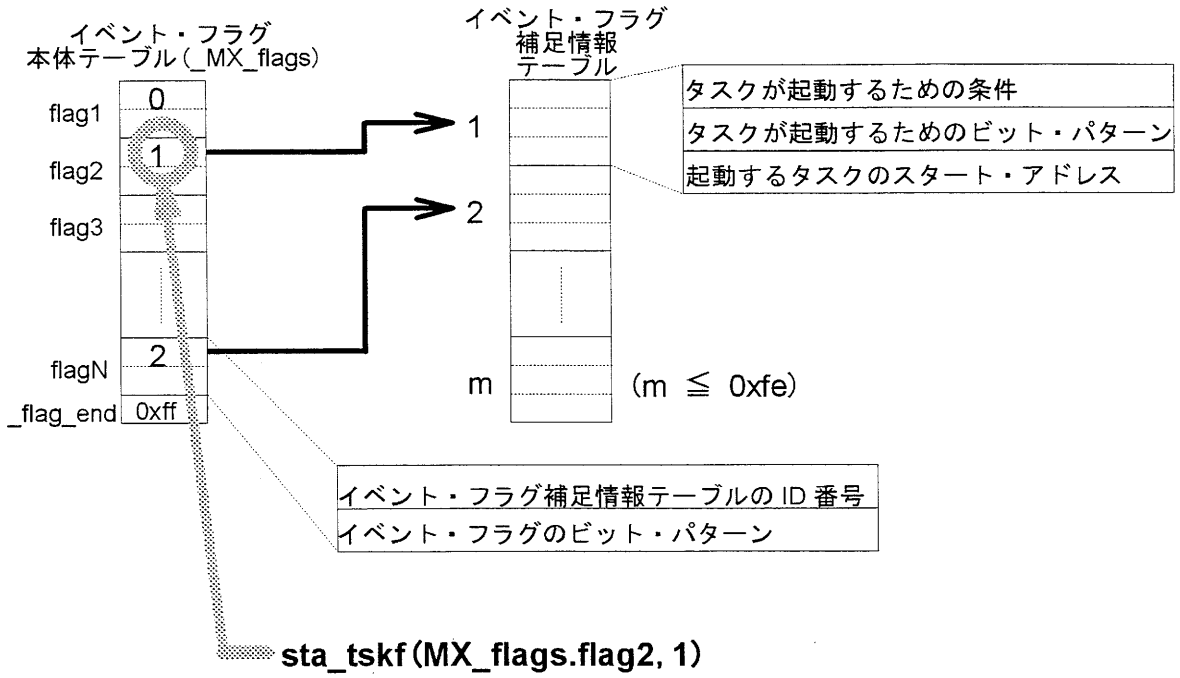
rpl_flg()システム・コールは，パラメータで指定されたビット・パターンをそのままイベント・フラグにセットします。

イベント・フラグの指定ビットのクリアは，clr_flg()システム・コールで行います。 clr_flg()システム・コールを発行する際に，パラメータとしてクリアするビットの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・パターンの論理積（AND）をとった値になります。

4.2 事象の発生に対応するタスクの登録

本OS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・コールによって行われませんので，ユーザが直接，補足情報を設定しなければなりません。

図4−2 タスクの登録



「タスクが起動するための条件」には，次の2種類があります。

- 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」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。

注意　1 つのイベント・フラグに対して複数回 sta_tskf()を発行し，補足情報 ID 番号を設定することは可能ですが，その場合は，前に設定した ID 番号の上に新たに設定する ID 番号が上書きされ，前に設定した ID の補足情報は参照されなくなります。このとき，ディバグ・タイプでは「補足情報を上書き登録した」というエラーが返されます。

保守／廃止

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，イベント・フラグ補足情報IDを格納する領域には無効IDが書き込まれます。これにより，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・コールによって補足情報を登録しないかぎり，そのイベント・フラグによるタスクの起動はしません。

保守／廃止

〔メ モ〕

保守／廃止

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。

本 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ビット・タイマ／イベント・カウンタを使用します。

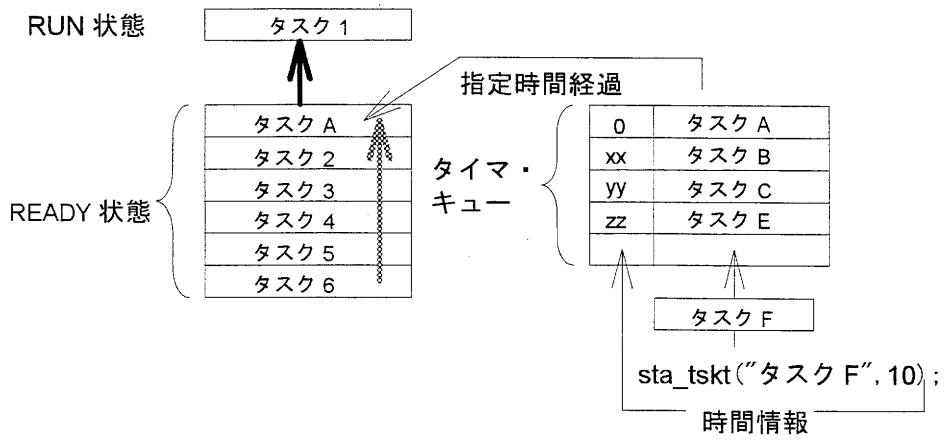
なお，８ビット・タイマ／イベント・カウンタの初期化（タイマ値の設定，カウント・クロックの選択）に関しては，初期化処理ルーチン内でユーザが行わなければなりません。

８ビット・タイマ／イベント・カウンタ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()があります（sta_tskt()に関する詳細は，**7. 3. 6 sta_tskt** を参照してください）。

図５−１ sta_tskt()の説明図



注意　指定時間が経過した時点でレディ・キューがいっぱいだった場合は，対象タスクはタイマ・キューから外され，レディ･キューにも登録されないため，結果的に消滅します。

保守／廃止

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，本 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，**7.3.8 chg_tskt** を参照してください）。

保守／廃止

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。

本 OS は，各割り込み要求の初期化処理を行わないので，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は，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，その処理内容には次のような制限，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。

ディバグ・タイプを使用する場合は，エラー・コードの退避と復帰も行ってください。

(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，本 OS 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。

次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によって終了します。

6.3 本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の種類は，ユーザが任意に設定することができますが，対象となるタイマ割り込みはMX専用とし，ハンドラ内ではなるべくユーザの処理をしないようにしてください。

タイマ処理では，タイマ・キューに格納されているタスクのタイマ値の減算と，指定時間が経過したタスクのレディ・キューへの登録処理を行います。ディバグ・タイプではさらに，レディ・キューへの登録処理におけるエラー処理も行います。

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，タスクの起動・終了，フラグのセット・リセットなどの操作が可能になります。

次の節では，システム・コールの機能概要と呼び出し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，その機能によって次の2つのグループに分けることができます。

（1）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・コールは次のとおりです。

sta_tsk　：指定タスクをレディ･キューの最後尾へ登録します。
sta_tske　：指定タスクをレディ･キューの最後尾へ登録し，自タスクを終了します。
sta_tskp　：指定タスクをレディ･キューの先頭へ登録します。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　：指定タスクをレディ･キューの先頭へ登録し，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，指定タスクをレディ･キューの先頭へ登録するように設定します。
sta_tskte　：指定時間経過後，指定タスクをレディ･キューの先頭へ登録するように設定し，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のID番号をイベント・フラグ本体テーブルに設定します。
ter_tskf　：イベント・フラグ操作で起動されるタスクまたは，レディ・キューにある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　　：指定タスクの状態を得ます。

chg_pri　　：レディ･キューに登録されている指定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，レディ・キューの先頭に登録し直します。

ext_tsk　　：自タスクを終了します。

(2) 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・コールは次のとおりです。

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。

保守／廃止

7.1.1 破壊レジスタ

アセンブリ言語で各システム・コールを呼び出す際に使用するレジスタを表7−1に示します。ユーザ・アプリケーションでこれらのレジスタを使用している場合，および割り込みハンドラからシステム・コールを呼び出す場合は，システム・コールを呼び出す前に待避処理を行ってください。

表7−1 破壊レジスタ

(1) タスク関連システム・コール

システム・コール	破壊レジスタ (スリム・タイプ)	破壊レジスタ (ディバグ・タイプ)
sta_tsk	AX	AX
sta_tske	AX,SP	AX,SP
sta_tskp	AX	AX
sta_tskpe	AX,SP	AX,SP
ter_tsk	AX	AX
sta_tskt	AX,DE	AX,DE
sta_tskte	AX,DE,SP	AX,DE,SP
chg_tskt	AX,DE	AX,DE
chg_tskte	AX,DE,SP	AX,DE,SP
ter_tskt	AX	AX
ter_tskf	AX	AX
rot_rdq	−	−
rot_rdqe	AX,SP	AX,SP
tsk_sts	AX,DE	AX,DE
chg_pri	AX	AX
ext_tsk	AX,SP	AX,SP

(2) イベント・フラグ管理システム・コール

システム・コール	破壊レジスタ (スリム・タイプ)	破壊レジスタ (ディバグ・タイプ)
sta_tskf	AX,E	AX,E
set_flg	AX,E	AX,E
clr_flg	AX,E	AX,E
rpl_flg	AX,E	AX,E

通常，AX レジスタおよび DE レジスタはパラメータの引き渡し用として使用されます。

ext_tsk()システム・コールおよび ext_tsk()複合システム・コールでは，AX レジスタはタスク分岐用として使用され，スタック・ポインタは初期化されます。

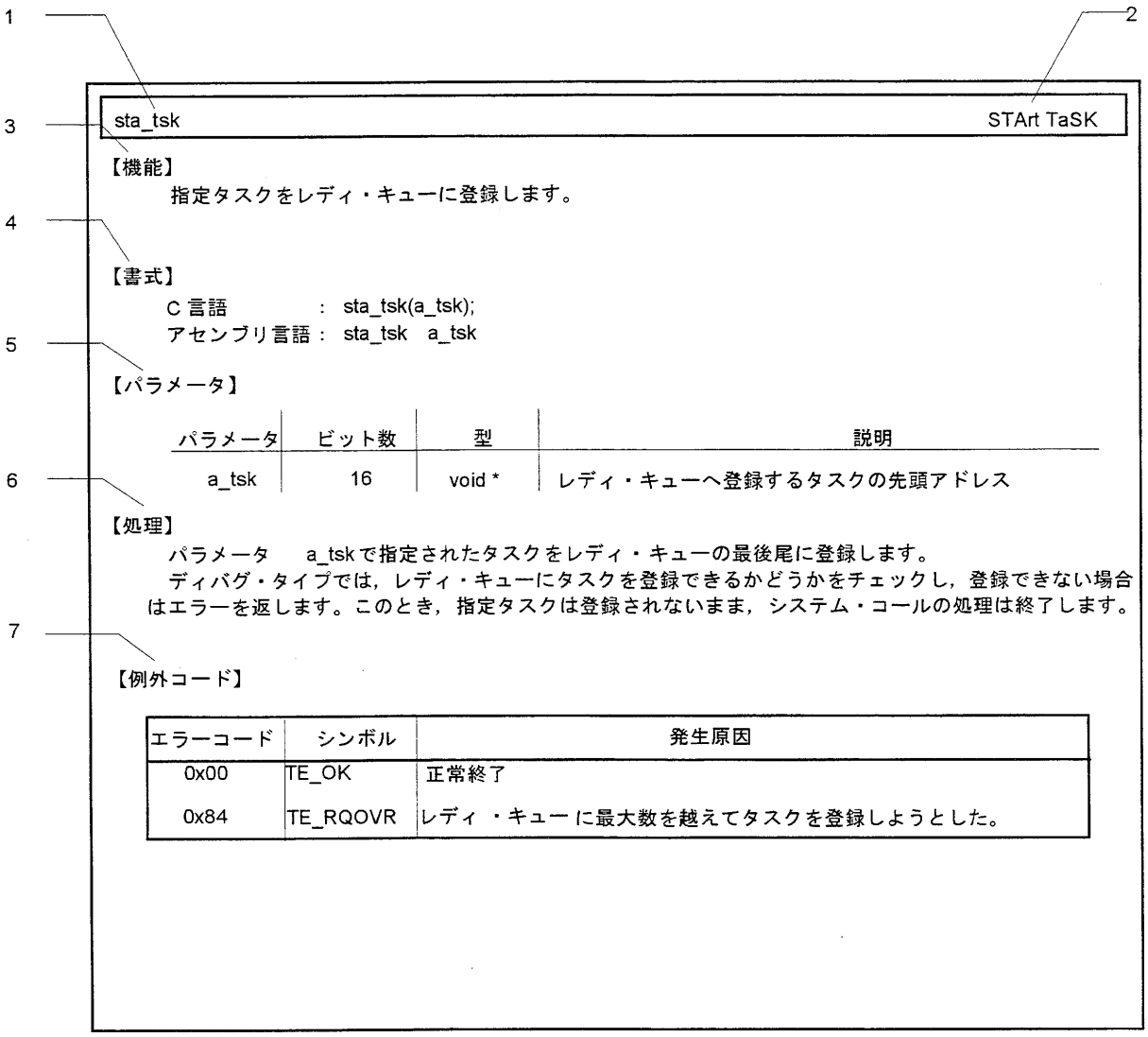
7.1.2 呼び出し方法

本OSのシステム・コールを呼び出す場合は，ユーザ・プログラム上でシステム・コールのインクルード・ファイルをインクルードし，インクルード・ファイルに記述されているマクロ（または関数）を使用します。

アセンブリ言語からシステム・コールを呼び出す場合は，インクルード・ファイルに記述されているマクロにならって，パラメータをレジスタにセットしたあと，直接CALL文で呼び出すこともできます。

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，下記のような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。ユーザ・タスクからシステム・コールを呼び出す際の書式。C言語とアセンブリ言語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，本OS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，本ＯＳ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ＯＳのみです。エラー・コードは_MX_errorシンボルで示される領域に格納されます。

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_tske(),	sta_tskp(),	sta_tskpe(),
ter_tsk(),	sta_tskt(),	sta_tskte(),	chg_tskt(),
chg_tskte(),	ter_tskt(),	sta_tskf(),	ter_tskf(),
rot_rdq(),	rot_rdqe(),	tsk_sts(),	chg_pri(),
ext_tsk()			

保守／廃止

7. 3. 1 sta_tsk

sta_tsk	STArt TaSK

【機能】

指定タスクをレディ･キューへ登録します。

【書式】

C 言語　　：sta_tsk(a_tsk);

アセンブリ言語　：sta_tsk　a_tsk

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	void*	レディ･キューへ登録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・キューの最後尾に登録します。

ディバグ・タイプでは，レディ・キューにタスクを登録できるかどうかをチェックし，登録できない場合はエラーを返します。このとき，指定タスクは登録されないまま，システム・コールの処理は終了します。

【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
0x84	TE_RQOVR	レディ・キューに最大タスク数を越えてタスクを登録しようとした

保守／廃止

7. 3. 2 sta_tske

sta_tske	STArt TaSK and Exit task

【機能】

指定タスクをレディ･キューへ登録し，自タスクを終了します。

【書式】

C言語　　：sta_tske(a_tsk);

アセンブリ言語：sta_tske　a_tsk

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	void *	レディ･キューへ登録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・キューの最後尾に登録し，自タスクをDORMANT状態にします。

ディバグ・タイプでは，レディ・キューにタスクを登録できるかどうかをチェックし，登録できない場合はエラーを返します。このとき，指定タスクは登録されないまま，レディ・キューの先頭タスクへの切り替え処理を行います。

タスク切り替え直後の割り込み受け付け状態は，直前に実行していたタスク（sta_tske()を発行したタスク）の状態を引き継ぎます。

【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
0x84	TE_RQOVR	レディ・キューに最大タスク数を越えてタスクを登録しようとした

保守／廃止

7. 3. 3 sta_tskp

sta_tskp	STArt TaSK and change Priority

【機能】

指定タスクをレディ･キューの先頭へ登録します。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。

【書式】

C言語　　：sta_tskp(a_tsk);

アセンブリ言語：sta_tskp　a_tsk

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	void *	レディ･キューへ登録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・キューの先頭に登録します。

ディバグ・タイプでは，レディ・キューにタスクを登録できるかどうかをチェックし，登録できない場合はエラーを返します。このとき，指定タスクは登録されないまま，システム・コールの処理は終了します。

【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
0x84	TE_RQOVR	レディ・キューに最大タスク数を越えてタスクを登録しようとした

保守／廃止

7. 3. 4 sta_tskpe

sta_tskpe	STArt TaSK and change Priority,and Exit task

【機能】

指定タスクをレディ･キューの先頭へ登録し，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。

【書式】

C言語　　：sta_tskpe(a_tsk);

アセンブリ言語：sta_tskpe　a_tsk

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	void *	レディ･キューへ登録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・キューの先頭に登録し，自タスクをDORMANT状態にし，レディ・キュー先頭のタスクへの切替え処理を行います。

ディバグ・タイプでは，レディ・キューにタスクを登録できるかどうかをチェックし，登録できない場合はエラーを返します。このとき，指定タスクは登録されないまま，レディ・キューの先頭タスクへの切り替え処理を行います。

タスク切り替え直後の割り込み受け付け状態は直前に実行していたタスク（sta_tskpe()を発行したタスク）の状態を引き継ぎます。

【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
0x84	TE_RQOVR	レディ・キューに最大タスク数を越えてタスクを登録しようとした

保守／廃止

7. 3. 5 ter_tsk

ter_tsk	TERminate TaSK in READY queue

【機能】

レディ・キューにある指定タスクを強制終了させます。

【書式】

C言語 : ter_tsk(a_tsk);

アセンブリ言語 : ter_tsk a_tsk

【パラメータ】

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

【処理】

a_tskで指定されたタスクがレディ・キューに登録されている場合には，その指定タスクをレディ・キューから外し，DORMANT状態にします。ただし，タスクがレディ・キューに重複登録されている場合は，最初に見つかったものだけをレディ・キューから外します。したがって，タスクがレディ・キューに残る場合があるので，このようにタスクを重複して登録する可能性がある場合は，tsk_sts()と組み合わせて使用してください。

ディバグ・タイプでは，指定タスクがレディ・キューに存在しない場合，エラーを返します。

【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
0x82	TE_DMT	指定したタスクがレディ・キュー上に存在しない

保守／廃止

7. 3. 6 sta_tskt

sta_tskt	STArt TaSK after a Time

【機能】

指定した時間が経過したら，指定タスクをレディ・キューの先頭へ登録するように設定します。

【書式】

C言語　　：sta_tskt(a_tsk,tmout);

アセンブリ言語：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	void *	タイマ・キューへ登録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

【処理】

a_tskで指定されたタスクをタイマ・キューに登録します。tmoutで指定するタスクの起動時間は，タイマ割り込みのインターバル時間を基準として，指定してください。

ディバグ・タイプではタイマ・キューにタスクを登録できるかどうかチェックし，登録できない場合はエラーを返します。このとき，指定タスクは登録されないまま，システム・コールの処理を終了します。

【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
0x88	TE_TQOVR	タイマ・キューに最大タスク数を越えてタスクを登録しようとした

保守／廃止

7. 3. 7 sta_tskte

sta_tskte	STArt TaSK after a Time,and Exit task

【機能】

指定した時間が経過したら，指定タスクをレディ･キューの先頭へ登録するように設定し，自タスクを終了します。

【書式】

C言語　　：sta_tskte(a_tsk,tmout);

アセンブリ言語：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	void *	タイマ･キューへ登録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

【処理】

a_tskで指定されたタスクをタイマ・キューに登録し，自タスクをDORMANT状態にします。tmoutで指定するタスクの起動時間は，タイマ割り込みのインターバル時間を基準として，指定してください。

ディバグ・タイプではタイマ・キューにタスクを登録できるかどうかチェックし，登録できない場合はエラーを返します。このとき，指定タスクは登録されないまま，レディ・キューの先頭タスクへの切り替え処理を行います。

レディ・キューが空の場合は，タイマ・キューへの登録処理のあと，アイドル・ルーチン（_MX_IDle）を実行しますが，その時のCPUの状態は，ユーザが任意に設定できます。

アイドル・ルーチン（_MX_IDle）先頭では，割り込み禁止状態になっています。タスク切り替えが起こった場合は，割り込みの受け付け状態は前のタスク（sta_tskte()を発行したタスク）の状態を引き継ぎます。

【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
0x88	TE_TQOVR	タイマ・キューに最大タスク数を越えてタスクを登録しようとした

保守／廃止

7. 3. 8 chg_tskt

chg_tskt	CHanGe TaSK's startTime

【機能】

指定したタスクの起動時間を変更します。

【書式】

C言語　　：chg_tskt(a_tsk,tmout);

アセンブリ言語：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	void *	起動時間を変更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

【処理】

a_tskで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，タイマ・キューまたはレディ・キューに指定タスクがあれば，強制終了を行ない，指定タスクをタイマ・キューに登録し直します。もし，指定されたタスクがなければ，指定タスクを新たにタイマ・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基準として，指定してください。

また，タイマ・キューとレディ・キューに指定タスクが重複して登録されている場合，起動時間の変更は最初に見つかったものに対してのみ行います。

ディバグ・タイプではタイマ・キューにタスクを登録できるかどうかチェックし，登録できない場合はエラーを返します。

【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
0x88	TE_TQOVR	タイマ・キューに最大タスク数を越えてタスクを登録しようとした

保守／廃止

7. 3. 9 chg_tskte

chg_tskte	CHanGe Task's startTime,and Exit task

【機能】

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。

【書式】

C言語　　：chg_tskte(a_tsk,tmout);

アセンブリ言語：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	void *	起動時間を変更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

【処理】

a_tskで指定されたタスクをタイマ・キュー，レディ・キューの順に検索します。もし，タイマ・キューまたはレディ・キューに指定タスクがあれば，強制終了を行ない，指定タスクをタイマ・キューに登録し直し，自タスクをDORMANT状態にします。

もし，どちらのキューにも指定タスクがなければ，指定タスクを新たにタイマ・キューに登録し，自タスクをDORMANT状態にします。

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基準として，指定してください。

また，タイマ・キューとレディ・キューに指定タスクが重複して登録されている場合，起動時間の変更は最初に見つかったものに対してのみ行います。

ディバグ・タイプではタイマ・キューにタスクを登録できるかどうかチェックし，登録できない場合はエラーを返します。このとき，指定タスクは登録されないまま，レディ・キューの先頭タスクへの切り替え処理を行います。

レディ・キューが空の場合は，タイマ・キューへの登録処理のあと，アイドル・ルーチン（_MX_idle）を実行しますが，その時のCPUの状態は，ユーザが任意に設定できます。

アイドル・ルーチン（_MX_idle）先頭では，割り込み禁止状態になっています。タスク切り替えが起こった場合は，割り込みの受け付け状態は前のタスク（chg_tskte()を発行したタスク）の状態を引き継ぎます。

【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
0x88	TE_TQOVR	タイマ・キューに最大タスク数を越えてタスクを登録しようとした

保守／廃止

7. 3. 10 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・キューにある指定タスクを強制終了させます。

【書式】

C言語　　：ter_tskt(a_tsk);

アセンブリ言語：ter_tskt　a_tsk

【パラメータ】

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

【処理】

a_tskで指定されたタスクをタイマ・キュー，レディ・キューの順に検索し，登録されている場合には，その指定タスクをキューから外し，DORMANT状態にします。

指定タスクがタイマ・キューとレディ・キューに重複して登録している場合は，最初に見つかったものだけをキューから外します。

ディバグ・タイプでは，指定タスクがタイマ・キューにもレディ・キューにも登録されていない場合はエラーを返します。

【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
0x82	TE_DMT	指定したタスクがタイマ・キューにもレディ・キューにも登録されていない

保守／廃止

7. 3. 11 sta_tskf

sta_tskf	STArt TaSK with event-Flag

【機能】

イベント・フラグ操作で起動する指定タスクを指定する補足情報のID番号をイベント・フラグ本体に設定します。

【書式】

C言語　　：sta_tskf(a_flg,pk_finfo);

アセンブリ言語：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	unsigned char *	対象イベント・フラグのアドレス
pk_finfo	8	unsigned char	イベント・フラグの補足情報ID

補足情報の各パラメータ	ビット数	型	説明
waiptn	8	unsigned char	タスクが要求しているビット・パターン
wfmode	8	unsigned char	タスクの起動条件
a_tsk	16	void *	起動するタスクの先頭アドレス

タスク起動条件	値	説明
EVENT_AND	0x00	指定ビットがすべて“1”になったとき，タスクを起動する
EVENT_OR	0x01	指定ビットのどれかが“1”になったとき，タスクを起動する

【処理】

a_flgで指定されたイベント・フラグにイベント・フラグ補足情報IDを設定し，a_flgで起動するタスクおよび条件を登録します。a_flgで指定されたイベント・フラグのビット・パターンは0でクリアされます。ID番号に0を指定すると，イベント・フラグを無効化することができます。

ディバグ・タイプでは，すでに補足情報IDが設定されているイベント・フラグに対して補足情報IDを上書き設定した場合はエラーを返します。

【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
0xa0	TE_DBLENT	すでに有効なフラグに対して補足情報IDを上書き登録した

注意　C 言語でイベント・フラグを扱うときは，フラグ本体を unsigned char 型で 2 バイトのサイズの配列として定義してください。

保守／廃止

7. 3. 12 ter_tskf

ter_tskf	TERminate TaSK in READY queue or event-Flag

【機能】

イベント・フラグ操作で起動される指定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。

【書式】

C言語　　：ter_tskf(a_tsk);

アセンブリ言語：ter_tskf　a_tsk

【パラメータ】

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

【処理】

a_tskで指定されたタスクがイベント・フラグの操作で起動されるタスクの場合は，指定されたタスクを起動するイベント・フラグを無効にします。もし，指定されたタスクがすでにレディ・キューに登録されている場合は，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。

指定タスクがイベント・フラグ操作で起動されるタスクでない場合でも，レディ・キューに登録されているタスクであれば，強制終了させます。

指定タスクがイベント・フラグとレディ・キューに重複して登録されている場合は，最初に見つかったものだけを無効にします（または，キューから外します）。

ディバグ・タイプでは，指定タスクがイベント・フラグにもレディ・キューにも登録されていない場合はエラーを返します。

【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
0x82	TE_DMT	指定タスクがイベント・フラグにもレディ・キューにも登録されていない

保守／廃止

7. 3. 13 rot_rdq

rot_rdq	ROTate READY Queue

【機能】

自タスクをレディ・キューの最後尾に登録します。

【書式】

C言語　　：rot_rdq();

アセンブリ言語：rot_rdq

【パラメータ】

なし

【処理】

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・タイプでは，レディ・キューにタスクを登録できるかどうかをチェックし，登録できない場合はエラーを返します。このとき，指定タスクは登録されないまま，システム・コールの処理は終了します。

【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
0x84	TE_RQOVR	レディー・キューに最大タスク数を越えてタスクを登録しようとした

保守／廃止

7. 3. 14 rot_rdqe

rot_rdqe	ROTate READY Queue and Exit task

【機能】

自タスクをレディ・キューの最後尾に登録し，自タスクを終了させます。

【書式】

C言語　　：rot_rdqe();

アセンブリ言語：rot_rdqe

【パラメータ】

なし

【処理】

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

ディバグ・タイプでは，レディ・キューにタスクを登録できるかどうかをチェックし，登録できない場合はエラーを返します。このとき，指定タスクは登録されないまま，レディ・キューの先頭タスクへの切り替え処理を行います。

タスク切り替え直後の割り込み受け付け状態は直前に実行していたタスク（rot_rdqe()を発行したタスク）の状態を引き継ぎます。

【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
0x84	TE_RQOVR	レディ・キューに最大タスク数を越えてタスクを登録しようとした

保守／廃止

7. 3. 15 tsk_sts

tsk_sts	get TaSK STatuS

【機能】

タスクの状態を得ます。

【書式】

C言語　　：tsk_sts(pk_tskst,a_tsk);

アセンブリ言語：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	unsigned char *	タスクの状態を返す領域へのポインタ
a_tsk	16	void *	タスクの先頭アドレス

【処理】

指定されたタスクがレディ・キューに登録されている場合は pk_tskstで示された領域にTTS_RDYを返し，登録されていない場合はTTS_DMTを返します。

ディバグ・タイプで返すエラー・コードは常にTE_OKです。

[pk_tskst]	シンボル	意味
0x02	TTS_RDY	指定したタスクがREADY状態
0x10	TTS_DMT	指定したタスクがDORMANT状態(起動時間待ち，イベント・フラグ条件成立待ちを含む)

【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

保守／廃止

7. 3. 16 chg_pri

chg_pri	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設定します。

【書式】

C言語　　：chg_pri(a_tsk);

アセンブリ言語：chg_pri　a_tsk

【パラメータ】

パラメータ	ビット数	型	説明
a_tsk	16	void *	登録し直すタスクの先頭アドレス

【処理】

a_tskで指定されたタスクがレディ・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。

レディ・キューにタスクが重複登録されている場合は，最初に見つかったものだけをレディ・キューの先頭に移動します。

ディバグ・タイプでは，指定タスクがレディ・キューに登録されていない場合，エラーを返します。

【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
0x82	TE_DMT	指定タスクがレディ・キューに登録されていない

保守／廃止

7. 3. 17 ext_tsk

ext_tsk	EXiT TaSK

【機能】

自タスクを終了します。

【書式】

C言語　　：ext_tsk();

アセンブリ言語：ext_tsk

【パラメータ】

なし

【処理】

自タスクをDORMANT状態にします。

レディ・キューにタスクが登録されていれば，レディ・キューの先頭のタスクを実行します。

レディ・キューが空の場合は，アイドル・ルーチン（_MX_idle）を実行しますが，その時のCPUの状態は，ユーザが任意に設定できます。

アイドル・ルーチン（_MX_idle）先頭では，割り込み禁止状態になっています。タスク切り替えが起こった場合は，割り込みの受け付け状態は前のタスク（ext_tsk()を発行したタスク）の状態を引き継ぎます。

【例外コード】

なし

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()

保守／廃止

7. 4. 1 set_flg

set_flg	SET event-FLaG

【機能】

イベント・フラグの任意のビットをセットします。

【書式】

C言語　　：set_flg(a_flg,setptn);

アセンブリ言語：set_flg　a_flg,setptn

【パラメータ】

パラメータ	ビット数	型	説明
a_flg	16	unsigned char *	対象イベント・フラグのアドレス
setptn	8	unsigned 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・キューの先頭に登録し，補足情報 ID 格納領域に無効 ID を書き込みます。setptn はセットしたいビットの値を 1 にして指定してください。

ディバグ・タイプでは，イベント・フラグにビット・パターンをセットできるかどうか，起動条件が成立したとき，タスクを登録できるかどうかについてチェックを行い，エラーを返します。

【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(起動条件が成立し，タスクがレディ･キューに登録された)
0x01	TE_SET	正常終了(イベント・フラグはセットされたが，起動条件は成立しない)
0x90	TE_NOENT	指定イベント・フラグが無効状態(起動タスクが登録されていない)
0x84	TE_RQOVR	レディ･キューのオーバフロー(起動条件が成立したがタスクがレディ･キューに登録できない)

注意　C言語でイベント・フラグを扱うときは，フラグ本体をunsigned char型で2バイトのサイズの配列として定義してください。

保守／廃止

7. 4. 2 clr_flg

clr_flg	CLeaR event-FLaG

【機能】

イベント・フラグの任意のビットをクリアします。

【書式】

C言語　　：clr_flg(a_flg,clrptn);

アセンブリ言語：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	unsigned char *	対象イベント・フラグのアドレス
clrptn	8	unsigned char	クリアするビットを示す情報

【処理】

a_flgで指定されたイベント・フラグのビット・パターンとclrptnで指定されたビット・パターンのANDをとり，イベント・フラグのビット・パターンを変更します。clrptnはクリアしたいビットの値を0にして指定してください。

ディバグ・タイプでは，a_flgに無効フラグが指定された場合はエラーを返します。

【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
0x90	TE_NOENT	指定イベント・フラグが無効状態(起動タスクが登録されていない)

注意　C言語でイベント・フラグを扱うときは，フラグ本体をunsigned char型で2バイトのサイズの配列として定義してください。

保守／廃止

7. 4. 3 rpl_flg

rpl_flg	RePLace event-FLaG

【機能】

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。

【書式】

C言語 : rpl_flg(a_flg,rplptn);

アセンブリ言語 : rpl_flg a_flg,rplptn

【パラメータ】

パラメータ	ビット数	型	説明
a_flg	16	unsigned char *	対象イベント・フラグのアドレス
rplptn	8	unsigned char	入れ替えるビット・パターン

【処理】

a_flg で指定されたイベント・フラグに rplptn で指定されたビット・パターンを設定します。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・キューの先頭に登録し，補足情報 ID 格納領域に無効 ID を書き込みます。

ディバグ・タイプでは，イベント・フラグにビット・パターンをセットできるかどうか，起動条件が成立したとき，タスクを登録できるかどうかについてチェックを行い，エラーを返します。

【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(起動条件が成立し，タスクがレディ･キューに登録された)
0x01	TE_SET	正常終了(イベント・フラグはセットされたが，起動条件は成立しない)
0x90	TE_NOENT	指定イベント・フラグが無効状態(起動タスクが登録されていない)
0x84	TE_RQOVR	レディ･キューのオーバフロー(起動条件が成立したがタスクがレディ･キューに登録できない)

注意 C言語でイベント・フラグを扱うときは，フラグ本体をunsigned char型で2バイトのサイズの配列として定義してください。

保守／廃止

7.5 システム・コールのエラー

ここでは，本OSのディバグ・タイプが返すエラーについて説明します。

エラー・コードは，SWA中の_MX_error領域に格納されます。この領域は，システム・コール処理の先頭でリセット(ゼロ・クリア)されます。ただし，ext_tskシステム・コールではリセットされません。エラー・コードのセットはシステム・コール中およびタイマ処理中に行われます。

エラー領域の各ビットの意味を表7−2に示します。表7−2でビット番号は上位ビットから順に7，6，……，0と数えています。

表7−2 エラー領域のビット内容

ビット番号	意味
7	エラーが発生したとき(ビット1〜6がセットされるとき)，セットされます。
6	タイマ処理中のタスク起動処理でエラーが発生したとき，セットされます。
5	イベント・フラグに補足情報IDを上書き登録したとき，セットされます。
4	無効フラグにビット・パターンを設定したとき，セットされます。
3	タイマ・キューに最大タスク数を越えてタスクを登録しようとしたとき，セットされます。
2	レディ・キューに最大タスク数を越えてタスクを登録しようとしたとき，セットされます。
1	検索対象オブジェクト管理ブロック内に指定タスクが登録されていないとき，セットされます。
0	有効フラグにビット・パターンを設定したが起動条件を満たさないとき，セットされます。

表7−2に示したビット内容から，システム・コールでセットされるエラー・コードは表7−3のようになります。また，ビット6はシステム・コールではなく，タイマ処理でエラーが発生した場合にセットされます。

エラー領域はシステム・コール発行ごとにリセットされますので，システム・コール中に発生した割り込み時にエラー内容を保護するために，割り込みハンドラからシステム・コールを発行する場合は，割り込みハンドラの最初と最後でエラー領域の内容を待避・復帰してください。

表7−3 システム・コールによってセットされるエラー・コード

エラー・コード	シンボル	発生原因
0x00	TE_OK	正常終了
0x01	TE_SET	イベント・フラグにビット・パターンをセット(タスク起動なし)
0x82	TE_DMT	検索範囲内(レディ・キュー，タイマ・キュー，イベント・フラグ)にタスクが登録されていない
0x84	TE_RQOVR	レディ・キューに最大タスク数を越えてタスクを登録しようとした
0x88	TE_TQOVR	タイマ・キューに最大タスク数を越えてタスクを登録しようとした
0x90	TE_NOENT	無効フラグに対してビット・パターンを設定した
0xa0	TE_DBLENT	すでに有効なイベント・フラグに対して補足情報IDを上書き登録した

表7−4 タイマ処理によってセットされるエラー・コード

エラー・コード	シンボル	発生原因
0xc4	TE_TMERR	タイマ処理中のタスクのレディ・キューへの登録処理において，レディ・キューに最大タスク数を越えてタスクを登録しようとした（タイマ・キュー内のタスクが消滅した）

〔メ モ〕

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，次のとおりです。

・コード・サイズ

・スリム・タイプ： 最大約 1.4K バイト

・ディバグ・タイプ： 最大約 1.6K 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

スリム・タイプ： 6 バイト（タイマ使用時）

ディバグ・タイプ： 9 バイト（タイマ使用時）

備考 タイマを使用しない場合は，各タイプから 2 バイトを差し引いた値となる。

8.2 メモリ容量の見積もり方

本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）
- 本OSはディバグ・タイプを使用

備考 タスク数は40個ありますが，レディ・キューに同時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・キューに同時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：	9バイト
合計	153バイト

例題のシステムでは，153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

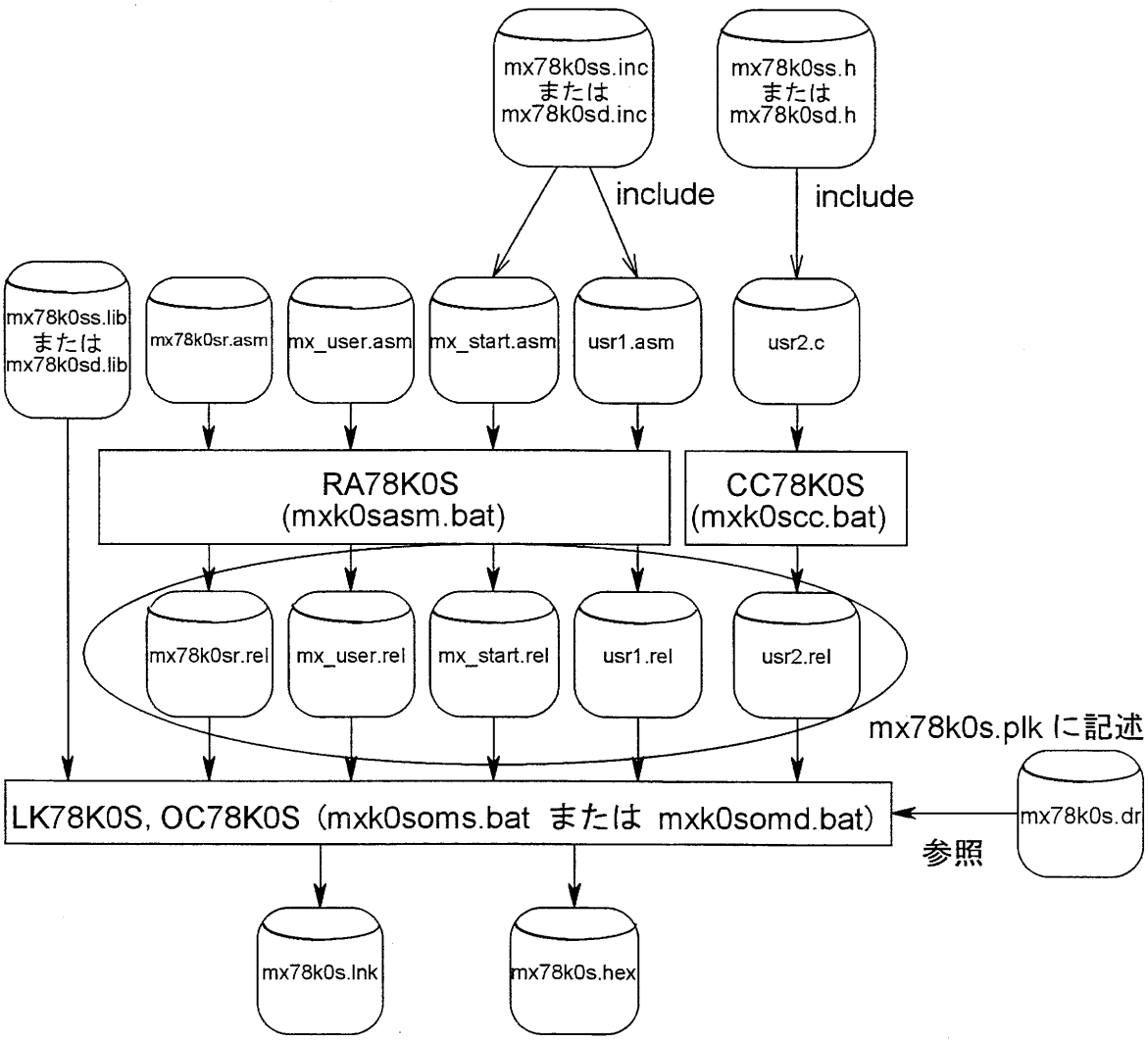


図 9−1 に示した各種ファイルの意味を次に示します。下線で示したファイルは製品に添付されているファイルです。ただし，サンプルとして添付されているファイルも含みます。

mx78k0ss.inc	アセンブリ言語用スリム・タイプ MX78K0S インクルード・ファイル
mx78k0sd.inc	アセンブリ言語用ディバグ・タイプ MX78K0S インクルード・ファイル
mx78k0ss.h	C 言語用スリム・タイプ MX78K0S インクルード・ファイル
mx78k0sd.h	C 言語用ディバグ・タイプ MX78K0S インクルード・ファイル
mx78k0ss.lib	スリム・タイプ MX78K0S システム・コール・ライブラリ・ファイル
mx78k0sd.lib	ディバグ・タイプ MX78K0S システム・コール・ライブラリ・ファイル
mx78k0sr.asm	MX78K0S メモリ設定ファイル
mx_start.asm	スタート・アップ・ルーチン
mx_user.asm	ユーザー・オウン・コーディング部
usr1.asm	ユーザ作成タスク（アセンブリ言語）… mx_smp_a.asm として提供
usr2.c	ユーザ作成タスク（C 言語）… mx_smp_c.c として提供
mx78k0s.dr	リンク・ディレクティブ・ファイル
mx78k0s.plk	リンカ・パラメータ・ファイル
mx78k0s.lnk	ロード・モジュール（ROM 化情報を含まない）
mx78k0s.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(mxk0sasm.bat)，コンパイラ起動用バッチ・ファイル(mxk0scc.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(mx78k0s.plk)とリンクさせるファイル群(拡張子.rel)，および，リンク・ディレクティブ・ファイル(mx78k0s.dr)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￥lib78k0s）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 LK78K0S のオプションを変更してください。

(2) アセンブラ（RA78K0S）のオプション

RA78K0S 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行うと，正常にアセンブル/リンクできない場合があります。

保守／廃止

(3) コンパイラ（CC78K0S）のオプション

CC78K0S のオプションは"-qc -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行うと，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。

(4) リンカ（LK78K0S）のオプション

LK78K0S 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"オプションを削除した場合は，メモリ設定ファイル（"mx78k0sr.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。

(5) リンク・ディレクティブ・ファイル

メモリ配置はリンク・ディレクティブ・ファイルを用いて行います。システムで予約されているセグメント名を表 9−1 に示します。MX78K0S 本体と SWA の配置には，この予約セグメント名を使用してください。なお，その他のモジュール（キュー，タスクなど）はユーザが任意にセグメント名を設定して配置指定を行ってください。配置指定を省略した場合は，リンカによって空き領域に自動配置されます。

表 9−1　予約セグメント名

セグメント名	セグメント種類	再配置属性	内容
mx78k0s	CSEG	UNIT	MX78K0S 本体
mxk0sSWA	DSEG	SADDRP	SWA 領域

保守／廃止

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。

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
$INCLUDE    (MX78K0SS.INC)                    ①
EXTRN       n_500ms                           ②
EXTRN       _task3
PUBLIC      _task1
PUBLIC      _task2
            CSEG
_task1:     CLR1        P1.0
            sta_tsk     _task2
            sta_tskt    _task3,n_500ms        ②
            ext_tsk                           ③
_task2:     BT          P1.0,$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本 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 1ms の場合，n_500ms＝500 に定義する。

③タスクの最後は"ext_tsk"などの，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。

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)
#include "mx78k0ss.h"                                    ①
#pragma SFR
extern int n_10ms;                                       ②
extern void task2();
void task3();
void task4();

void task3()
{
        int pk_tskst;
        int i;
        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本 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"などの，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と補足情報テーブルの型宣言はユーザが自身で行ってください（詳細は **10. 3. 8 イベント・フラグの設定**を参照してください）。

保守／廃止

9.3.3 レジスタについての注意事項

各システム・コールは引数の受け渡しに AX レジスタおよび DE レジスタを使用します。したがって，システム・コールを使用するとこれらのレジスタは破壊されますので，注意してください。

9.3.4 スタックについて

各システム・コールではユーザ・タスクと共通のスタック領域を使用します。各システム・コールが消費するスタック・サイズについては，製品添付の補足説明ファイル（"mx78k0s.txt"）を参照してください。

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エアおよびソフトウエアの初期化を行う部分です。初期化処理については第10章 **システム初期化処理**を参照してください。

9.4.2 ext_tsk アイドル処理部

レディ・キューが空のときに ext_tsk を発行した場合（次に起動するタスクがない場合）に行う処理を記述します。_MX_IDle 先頭では割り込み禁止状態になっていますので，EI 命令などで割り込みを許可してください。割り込み禁止状態のままでは，マスカブル割り込みが発生しないため，永久に新規タスクをレディ・キューに登録できません。

例 CPU をスタンバイ状態にする場合

```
PUBLIC      _MX_Idle
EXTRN       _MX_start
            CSEG
_MX_Idle:   HALT                    ;HALT モードにします
            EI                      ;割り込み要求で HALT から復帰後,
            NOP                     ;割り込み許可状態にします
            BR          _MX_start   ;割り込みから復帰後は，_MX_start に
                                    ;分岐します
```

保守／廃止

9.4.3 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，タイマ割り込みベクタの設定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) タイマ割り込みベクタの設定と割り込みハンドラの作成

割り込みベクタ・テーブルの本 OS が使用するタイマ割り込みに該当するアドレスに，本 OS のタイマ処理ルーチン（_MX_int）を呼び出す割り込みハンドラの先頭アドレスを設定します（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）。

さらに，OS のタイマ処理ルーチンを呼び出す割り込みハンドラを作成してください。

例 1. OS のタイマ割り込みに INTTM0 を使用する場合のベクタ・テーブル設定

```
EXTRN       TM0_INT
MX_VECT     CSEG        AT    10H   ;INTTM0 のベクタ・テーブル・アドレス
            DW          TM0_INT     ;タイマ処理を呼び出す割り込みハンドラ
```

例 2. タイマ割り込み処理ルーチンを呼び出す割り込みハンドラ

```
EXTRN       _MX_int
TM0_INT:
            CALL        !_MX_int
            RETI
```

(2) タイマ割り込み禁止ルーチン

本 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 本 OS が INTTM0 を使用している場合

```
_Prohibit_MX_int:
            SET1        TMMK0
            RET
```

(3) タイマ割り込み許可ルーチン

本 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 本 OS が INTTM0 を使用している場合

```
_Permit_MX_int:
            CLR1        TMMK0
            RET
```

保守／廃止

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。

ディバグ・タイプを使用する場合，例外エラー・コードはシステム・コール発行ごとにクリアされるので，割り込みハンドラ中でシステム・コールを発行する場合は，割り込みハンドラの入口／出口で例外エラー・コードを待避／復帰してください（割り込みハンドラで_MX_intを呼び出す場合も同様です）。

例 割り込みハンドラ中でsta_tskp()を発行する場合

```
                CSEG
_inttask:
                PUSH    AX              ;AX 保存
                MOV     A,_MX_error
                PUSH    AX              ;_MX_error 保存
                sta_tskp _task1         ;sta_tskp 発行
                CMP     _MX_error,#TE_OK ;エラー・チェック
                BZ      $no_error
                CALL    !_Error
no_error:       POP     AX              ;_MX_error 復帰
                MOV     _MX_error
                POP     AX              ;AX 復帰
                RETI
```

9.5 ライブラリ・ファイルの再構築

ここでは本 OS のソース・ファイルから，システム・コール・ライブラリ・ファイルを再構築する方法について述べます。

ライブラリ・ファイルの再構築には，ソース・ファイル形式の提供媒体に含まれているライブラリ構築用バッチ・ファイル(mxk0smk.bat)を使用します。バッチ・ファイルの使用方法については第 2 章　インストールを参照してください。

保守／廃止

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エア，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エアの初期化を行う部分とソフトウエ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。

表 10－1　初期化処理項目の概要

処理内容	
ハードウエア	CPU の初期化 ・ポート ・タイマ ・割り込み ・RAM 周辺の初期化
ソフトウエア	OS が使用する領域の確保 ・スタック・ポインタの設定 ・キューの確保 ・SWA 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

10. 2　ハードウエアの初期化

ハードウエアの初期化では，ユーザ・システムが動作するのに必要なハードウエアの初期化を行ってください（ハードウエ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，”sta_tskt”,”sta_tskte”,”chg_tskt”,”chg_tskte”,”ter_tskt”を使用する場合は，本 OS 用にタイマを 1 本確保してください。タイマのインターバル時間は 1ｍｓ程度を推奨します。

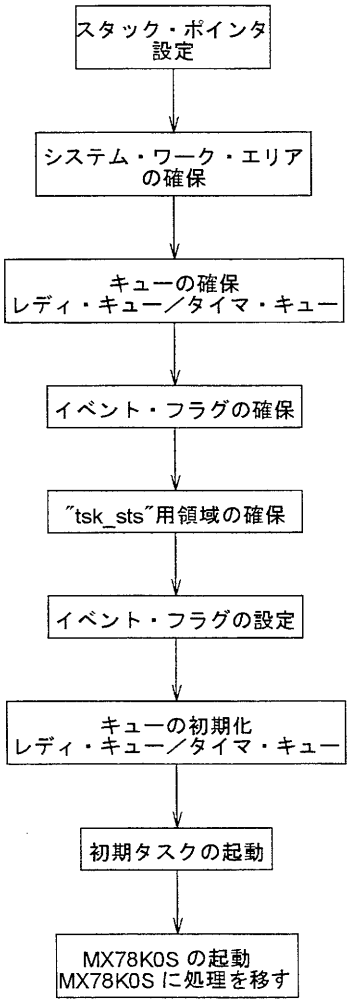
保守／廃止

10.3 ソフトウエアの初期化

ソフトウエアの初期化では，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。ソフトウエ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（"mx78k0sr.asm"）を修正することにより，行うことができます。

図 10−1 にソフトウエ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，初期化処理ルーチン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。

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。

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

保守／廃止

10. 3. 2 レディ・キューの確保

レディ・キューに登録されるタスクの最大数を設定します。レディ・キューには最大 63 個までタスクを登録することができます。

```
_MX_rq_Max              EQU     xx                      ;最大タスク数
mxk0sRQ                 DSEG
_MX_rq_StartAddress:    DS      (_MX_rq_Max+1)*4        ;レディ・キュー領域の先頭アドレス
_MX_rq_e0               EQU     _MX_rq_Max*4            ;レディ・キューの物理最終キュー
                                                        ;へのポインタ
```

10. 3. 3 タイマ・キューの確保

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。

ただし，"sta_tskt","sta_tskte","chg_tskt","chg_tskte","ter_tskt"のいずれのシステム・コールも使用しない場合は，タイマ・キューの設定をする必要はありません。

タイマ・キューには最大 63 個までタスクを登録することができます。

```
_MX_tq_Max              EQU     xx                      ;最大タスク数
mxk0sTQ                 DSEG
_MX_tq_StartAddress:    DS      (_MX_tq_Max+1)*4        ;タイマ・キュー領域の先頭アドレス
_MX_tq_r_e0             EQU     _MX_tq_Max*4            ;タイマ・キューの物理最終キュー
                                                        ;へのポインタ
_MX_tq_half             EQU     ((_MX_tq_Max+1) SHR 1)*4 ;タイマ・キューの物理中央キュー
                                                        ;へのポインタ
```

10. 3. 4 システム作業領域の確保

本OSのシステムが使用する作業領域（SWA）は下表に示す構成になっています。この領域は，本OSによって自動的に確保されますので，この領域の先頭アドレスだけをリンク・ディレクティブ・ファイルで指定してください。

SWA		
情報名	サイズ	説明
_MX_ctask	0x02	RUN状態のタスクのスタート・アドレス
_MX_rq_s	0x01	レディ・キューの先頭ポインタ
_MX_rqv_s	0x01	レディ・キューの空きエントリの先頭ポインタ
_MX_tq_s	0x01	タイマ・キューの先頭ポインタ
_MX_tqv_s	0x01	タイマ・キューの空きエントリの先頭ポインタ
az_arg	0x02	ディバグ用システム領域
_MX_error	0x01	エラー・コード格納領域

ただし，タイマ・キューを使用しない場合は，_MX_tq_sおよび_MX_tqv_sは確保されません。また，スリム・タイプを使用する場合は，az_argおよび_MX_errorは確保されません。

なお，SWAの先頭は必ずショート・ダイレクト・アドレシング領域の偶数アドレスに設定してください。

保守／廃止

10. 3. 5 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，"sta_tskf","ter_tskf","set_flg","rpl_flg","clr_flg"のいずれのシステム・コールも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 上に配置します。補足情報テーブルの配置の制限はありません。各テーブルの配置は必ずしも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

```
                        DSEG
_MX_flags:                          ; イベント・フラグ本体テーブル
_flag1:         DS      2           ; (イベント・フラグの個数分確保)
_flag2:         DS      2
_flag3:         DS      2
_flag_end:      DS      1           ; フラグ終端設定用領域
                                    ; (ter_tskf を使用しない場合は不要)
                        CSEG
_MX_flg_tbl:                        ; 補足情報テーブルの設定
                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
```

注意1. 本体テーブルの先頭レーベルは"_MX_flags"，終端子レーベルは"_flag_end"，補足情報テーブルの先頭レーベルは"_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。

2. ter_tskf を使用するときは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。

3. ter_tskf を使用しないときは，終端用領域の確保は原則として必要ありませんが，MX78K0S 対応ディバッガ（MD78K0S）を使用するときは，ter_tskf を使用するか否かに関わらず，"_flag_end"のレーベル名で終端領域を確保してください。

(2) C 言語で配置する場合

```
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 */
};

struct flaginfo
{
        unsigned char flag1[2];                     /* イベント・フラグ確保 */
        unsigned char flag2[2];                     /* (個数分確保) */
        unsigned char flag3[2];
        unsigned char flag_end;                     /* フラグ終端子用領域
                                                    (ter_tskf を使用しないときは不要) */
} MX_flags;

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

注意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。

2. ter_tskf を使用するときは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。
3. 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。
4. イベント・フラグ本体は 1 個につき，2 バイトの unsigned char 型領域を確保してください。
5. MX78K0S 対応ディバッガ（MD78K0S）を使用するときは，終端用領域名"_flag_end"を参照するため，イベント・フラグ本体テーブルの配置はアセンブリ言語で行ってください。

10. 3. 6 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
```

注意 "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。

保守／廃止

10. 3. 7　キューと SWA の初期化

各領域の確保が終了したあと，キューと SWA の初期化を行います。

(1) レディ・キューのみを使用する場合

レディ・キュー／SWA 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

タイマ・キュー／レディ・キュー／SWA 初期化ルーチンを呼び出します。

```
CALL        !_MX_init_tq
```

10. 3. 8　イベント・フラグの設定

イベント・フラグ本体テーブルを初期化し，終端子を設定します。本体テーブルの初期化は，sta_tskf の第 2 パラメータに 0x00 を与えることで行うことができます。ter_tskf を使用する場合は，終端子領域に 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。

(1) アセンブリ言語で設定する場合

```
MOV         A,0FFH              ; 終端子(0xff)設定
MOV         !_flag_end,A        ; (ter_tskf 使用時のみ)
sta_tskf    _flag1,0            ; _flag1 初期化
sta_tskf    _flag2,0            ; _flag2 初期化
sta_tskf    _flag3,0            ; _flag3 初期化

MOVW        AX,#_MX_flg_tbl     ; 補足情報設定
MOVW        HL,AX
MOV         [HL],#EVENT_XXX     ; 1 つ目のフラグの待ちモード
MOV         [HL+1],#xxxxxxxxB   ; 待ちビット・パターン
MOVW        AX,xxxx             ; 起動タスク
MOVW        [HL+2],AX
MOV         [HL+3],#EVENT_XXX   ; 2 つ目のフラグの待ちモード
MOV         [HL+4],#xxxxxxxxB   ; 待ちビット・パターン
MOVW        AX,xxxx             ; 起動タスク
MOVW        [HL+5],AX
```

（2）C 言語で設定する場合

```
void flag_init()
{
        MX_flags.flag_end = 0xff;                   /* 終端子(0xff)設定
                                                    (ter_tskf 使用時のみ) */
        sta_tskf(MX_flags.flag1,0);                 /* flag1 初期化 */
        sta_tskf(MX_flags.flag2,0);                 /* flag2 初期化 */
        sta_tskf(MX_flags.flag3,0);                 /* flag3 初期化 */

        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_f;g_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_f;g_tbl[1].func = xxxx;                  /* 起動タスク */
};
```

10. 3. 9　初期タスクの起動

メモリ領域の初期化が終了したあと，始めに起動するタスクを起動し，OS に制御を移します。

```
sta_tsk   xxxx
ext_tsk
```

〔メ モ〕

保守／廃止

付録　予約語一覧

本 OS の予約語およびシステムで使用されているセグメント名，オブジェクト・モジュール名の一覧を次に示します。

(1) 予約シンボル名

次のシンボルは本 OS の予約シンボル名として定義されています。ユーザ・システムで重複して定義しないように注意してください（メモリ設定ファイルおよびユーザ・オウン・コーディング部で定義するように指定されているものは除きます）。

TTS_RDY,TTS_DMT,TE_OK,TE_SET,TE_DMT,TE_RQOVR,TE_TQOVR,TE_TMERR,EVENT_AND,EVENT_OR,
_@STBEG,az_arg,_chg_pri,_chg_tskt,_chg_tskte,_clr_flg,_ext_tsk,_ext_tsk9,_end_itdsp,_flag_end,
_MX_ctask,_MX_DeleteTask,_MX_end_dsp,_MX_error,_MX_flags,_MX_flg_cal,_MX_flg_tbl,_MX_IDle,
_MX_init_rq,_MX_init_tq,_MX_int,_MX_JudgeSft,_MX_rqv_s,_MX_RQ_cal,_MX_RQ_calt,_MX_rq_e0,
_MX_RQ_regh,_MX_RQ_regt,_MX_rq_s,_MX_rq_StartAddress,_MX_Search,_MX_Search1,_MX_SftSub0,
_MX_SftSub1,_MX_start,_MX_tqv_s,_MX_TQ_cal,_MX_TQ_calh,_MX_tq_half,_MX_tq_r_e0,_MX_tq_s,
_MX_tq_StartAddress,
_Permit_MX_int,_Prohibit_MX_int,_rot_rdq,_rot_rdqe,_rpl_flg,_set_flg,
_sta_tsk,_sta_tske,_sta_tskf,_sta_tskp,_sta_tskpe,_sta_tskt,_sta_tskt9,
_sta_tskte,_ter_tsk,_ter_tsk9,_ter_tskf,_ter_tskt,_ter_tskt9,_tsetflg,_tsk_sts

(2) 予約セグメント名

次のセグメント名は，OS 内部で定義されています。再配置属性等の詳細は，**表 9−1 予約セグメント名**を参照してください。

OS 本体のセグメント名	:	mx78k0s
SWA のセグメント名	:	mxk0sSWA

(3) オブジェクト・モジュール名

本 OS を構成するオブジェクト・モジュール名には，次のものがあります。

CHGPRI, CHGTSKT, CHGTSKTE, CLRFLG, EXTTSK, ROTRDQ, ROTDRQE, SETFLG, STATSK, STATSKE,
STATSKF, STATSKP, STATSKPE, STATSKT, STATSKTE, TERTSK, TERTSKF, TERTSKT, TSKSTS,
MXDELETE, MXFLAG, MXINT, MXSEARCH, RQCLR, TRQCLR

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 半導体第一販売部 半導体第二販売部	〒460　名古屋市中区錦一丁目17番１号（NEC中部ビル）	名古屋 名古屋	(052)222-2170 (052)222-2190
関西支社 半導体第一販売部 半導体第二販売部 半導体第三販売部	〒540　大阪市中央区城見一丁目４番24号（NEC関西ビル）	大　阪 大　阪 大　阪	(06) 945-3178 (06) 945-3200 (06) 945-3208

北海道支社 札幌 (011)251-5599	太田支店 太田 (0276)46-4011	福井支店 福井 (0776)22-1866
東北支社 仙台 (022)267-8740	宇都宮支店 宇都宮 (028)621-2281	富山支店 富山 (0764)31-8461
岩手支店 盛岡 (019)651-4344	小山支店 小山 (0285)24-5011	三重支店 津 (0592)25-7341
郡山支店 郡山 (0249)23-5511	長野支社 松本 (0263)35-1662	京都支社 京都 (075)344-7824
いわき支店 いわき (0246)21-5511	甲府支店 甲府 (0552)24-4141	神戸支社 神戸 (078)333-3854
長岡支店 長岡 (0258)36-2155	埼玉支社 大宮 (048)649-1415	中国支社 広島 (082)242-5504
土浦支店 土浦 (0298)23-6161	立川支社 立川 (0425)26-5981	鳥取支店 鳥取 (0857)27-5311
水戸支店 水戸 (029)226-1717	千葉支社 千葉 (043)238-8116	岡山支店 岡山 (086)225-4455
神奈川支社 横浜 (045)682-4524	静岡支社 静岡 (054)254-4794	松山支店 松山 (089)945-4149
群馬支店 高崎 (0273)26-1255	北陸支社 金沢 (076)232-7303	九州支社 福岡 (092)261-2806

【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)

C97.8

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］78K/0S シリーズ用 OS　MX78K0S　基礎編　ユーザーズ・マニュアル
（U12938JJ1V0UM00（第 1 版））

［お名前など］（さしつかえのない範囲で）

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

1. ご評価（各欄に○をご記入ください）

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

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

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

4. ご意見，ご要望

5. このドキュメントをお届けしたのは
NEC 販売員，特約店販売員，NEC 半導体ソリューション技術本部員，
その他（　　）

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

NEC 半導体インフォメーションセンター
FAX：（044）548-7900

キリトリ