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

RX77016

リアルタイム・オペレーティング・システム

機能編

対象デバイス

μPD77016
μPD77017
μPD77018
μPD77018A
μPD77019
μPD77110
μPD77111
μPD77112
μPD77113
μPD77114

資料番号　U14397JJ1V0UM00（第 1 版）
発行年月　July　1999 NS　CP（K）

[メ　　モ]

目次要約

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

PC/AT は米国 IBM 社の商標です。

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

M7A 98.8

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

対象者　このマニュアルは，μPD77016 ファミリ用リアルタイム・オペレーティング・システム RX77016 の機能を理解し，それを用いたアプリケーション・プログラムを設計するユーザを対象とします。

μPD77016 ファミリは，μPD77016, 77017, 77018, 77018A, 77019，77110，77111，77112，77113，77114 の総称です。

このマニュアルでは，特に機能面において違いがないかぎり，μPD77016 を代表品種として説明しています。

目　的　このマニュアルは，RX77016 の基本的な機能について，応用プログラムを用いてユーザに理解していただくことを目的とします。なお，掲載のプログラムは例示的に示したものであり，量産設計を対象とするものではありません。

構　成　このマニュアルでは，大きく分けて次の内容で構成しています。

- 機　能
- システム・コール・インタフェース
- ユーザ記述ファイル

読み方　このマニュアルの読者は，論理回路やマイクロコンピュータに関する一般的知識が必要となります。

このマニュアルでは，タスク（Task）とサブタスク（SubTask）というキー・ワードを使用しますが，アプリケーション・プログラムと同等の意味を持つものではありません。サブタスクとは，OS が管理する最小単位を表し，タスクとは，いくつかのサブタスクを１つの集まりとして管理する単位を表します。

凡　例

データ表記の重み	：左が上位桁，右が下位桁
アクティブ・ロウの表記	：$\overline{\times\times\times}$（端子，信号の名称に上線）
注	：本文中につけた注の説明
注意	：気をつけて読んでいただきたい内容
備考	：本文中の補足説明
数の表記	：2 進数…××××または 0b××××
	10 進数…××××
	16 進数…0x××××

表現形式と対応レジスタ

表現形式	対応レジスタ
ro，ro’，ro”	R0-R7
rl，rl’	R0L-R7L
rh，rh’	R0H-R7H
re	R0E-R7E
reh	R0EH-R7EH
dp	DP0-DP7
dn	DN0-DN7
dm	DMX，DMY
dpx	DP0-DP3
dpy	DP4-DP7
dpx_mod	DPn，DPn++，DPn--，DPn##，DPn%%，!DPn##(**n**=0-3)
dpy_mod	DPn，DPn++，DPn--，DPn##，DPn%%，!DPn##(**n**=4-7)
dp_imm	DPn##imm(n=0-7)
＊＊＊＊	**×××をアドレスとするメモリの内容** **例** DP0 **レジスタの内容が** 1000 **のとき，***DP0 **はメモリの** 1000 **番地の内容を表します。**

関連資料　関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

デバイスに関する資料

資料名 / 品名	パンフレット	データ・シート	ユーザーズ・マニュアル		アプリケーション・ノート	
			アーキテクチャ編	命令編	基本ソフトウエア編	ライブラリ編
μPD77016	U12395J	U10891J	U10503J	U13116J	U11958J	U12021J
μPD77017		U10902J				
μPD77018						
μPD77018A		U11849J				
μPD77019						
μPD77019-013		U13053J				
μPD77110		U12801J	作成中			
μPD77111						
μPD77112						
μPD77113		U14373J				
μPD77114						

開発ツールに関する資料

資　料　名		資料番号
SM77016　ユーザーズ・マニュアル		U11602J
WB77016　ユーザーズ・マニュアル	言語編	U10078J
	操作編	U11506J
ID77016　ユーザーズ・マニュアル		U10118J
IE-77016-98/PC　ユーザーズ・マニュアル	ハードウエア編	U13044J
IE-77016-CM-EM6　ユーザーズ・マニュアル		EEU-984
EB-77017　ユーザーズ・マニュアル		EEU-983
μPD77016　スタータ・キット　ユーザーズ・マニュアル		U13032J
IE-77016-CM-LC　ユーザーズ・マニュアル		U14139J
RX77016　ユーザーズ・マニュアル	機能編	このマニュアル
RX77016　ユーザーズ・マニュアル	コンフィギュレーション・ツール編	U14404J
RX77016　アプリケーション・ノート	HOST API　インタフェース編	U14371J

注意　上記関連資料は，予告なしに内容を変更することがあります。設計などには，必ず最新の資料をご使用ください。

［メ　　モ］

目　次

図の目次

図番号　　タイトル，ページ

表の目次

表番号　　タイトル，ページ

［メ　　モ］

第1章　機　　能

RX77016 は 16 ビット固定小数点ディジタル・シグナル・プロセッサμPD77016 ファミリ上のアプリケーションをタスクとしたマルチタスク環境を提供します。

本 OS は高周波のアプリケーション・プログラムに対応し，すでに完成された多くの DSP アプリケーション・プログラムを容易にタスク化できるようなインタフェースを提供します。

1.1　特　　徴

DSP アプリケーション・プログラムは，時間的制約が非常に厳しく決められています。このようなプログラムを管理する場合，OS が介在するオーバヘッドをできるだけ軽減し，効率よくスケジューリングする必要があります。

本 OS は，タスク／サブタスクとして管理する DSP アプリケーション・プログラムに次のような特徴があるものとして管理します。

- 一連の処理を一定時間内に終了させなければならない。
- 基本的に周期的な性質を持っている。
- 上記 2 点から，一定時間内の実行回数が多いものほどタスクのプライオリティは高い。
- 同じインターバルあるいはほぼ同じインターバルの間に実行開始／終了しなければならないアプリケーション・プログラムは，お互いにタスク・スイッチングの必要性を持たない。
- マルチタスクの場合，ユーザがアプリケーション・プログラム内でタスク・スイッチング可能な箇所を特定しにくい。

本 OS は，これらの特徴を持つ DSP アプリケーション・プログラム向けに次のような設計となっています。

- タスク／サブタスクのスケジューリングを OS 内部で行う。
- タスクの実行状況によるリアルタイムなタスク・スイッチを OS 内部で行う。
- 同じインターバル，あるいはほぼ同じインターバルの間に実行開始／終了するサブタスクをグループ化することができる（これによりタスク管理に要する時間およびメモリ消費の軽減を図っている）。
- タスク／サブタスクのスケジューリングに必要なデータを，パラメータとして OS に与えるだけで，アプリケーション・プログラムを OS に組み込むことができる。

また，本 OS は非周期的なプログラムにも対応します。これにより，本 OS が提供するシステム・コールからイベント処理プログラムに対して実行要求を発行することができます。以降，このような非周期的に実行するプログラムをイベント処理プログラム，OS がイベント処理プログラムを管理する単位をイベント・タスクといいます。

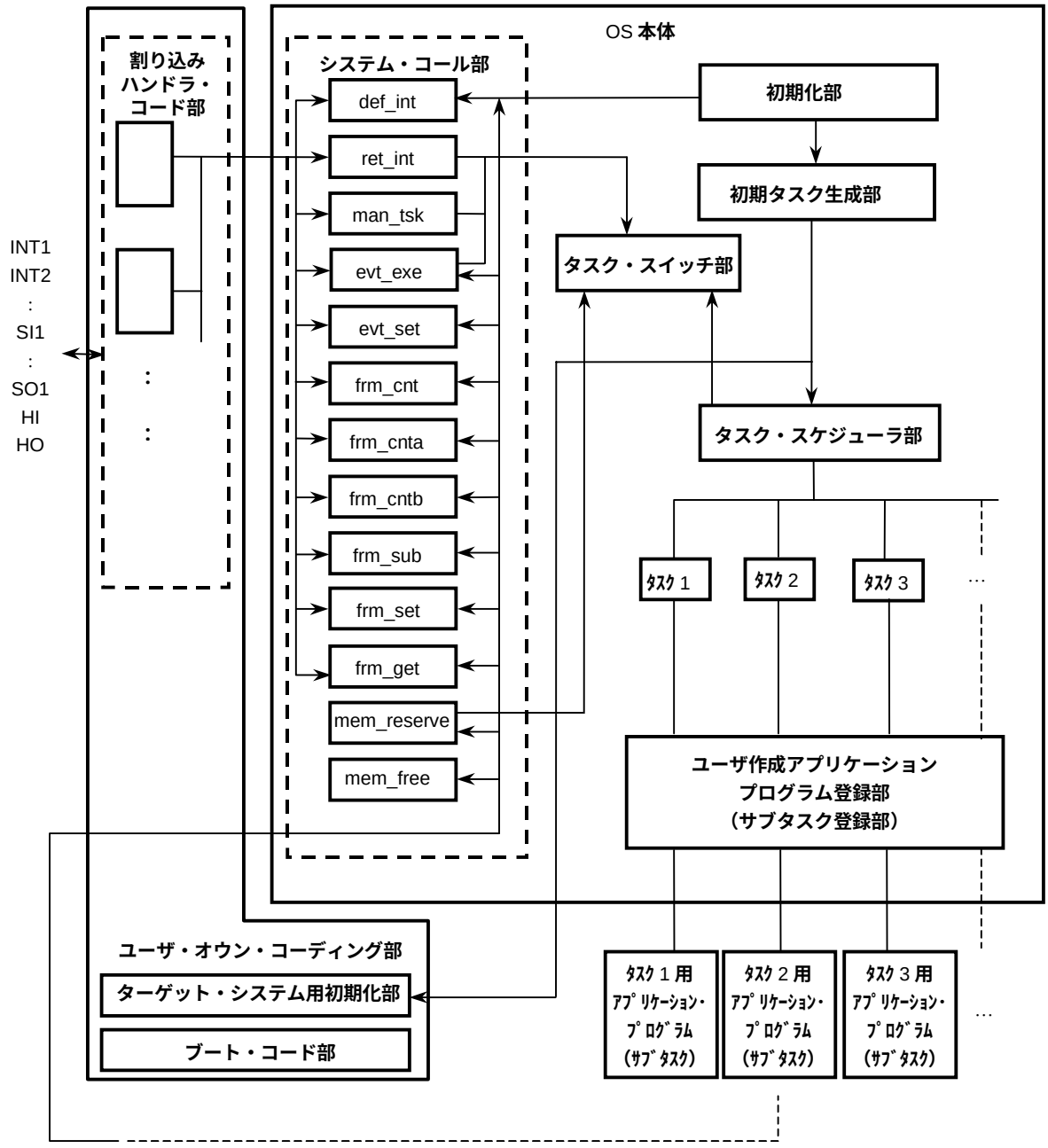
1.2 オーダ名称と対象 OS

ホスト・マシン	オーダ名称（媒体）	対象 OS
PC-9800 シリーズ	μSAA17RX77016（CD-ROM）	Windows™95，Windows NT™4.0
IBM PC/AT™		

1.3 RX77016 機能概要

本 OS は，μPD77016 ファミリ上での DSP アプリケーション・プログラム開発を支援するとともに，マルチメディア・アクセラレータとしても利用できます。また，ユーザのシステムの規模や形態にできるだけ順応できるように，タスク登録の容易性や高周波アプリケーション・プログラムの対応を可能にしています。

図1―1　ブロック図


OS 本体
割り込み
ハンドラ・
コード部
システム・コール部
def_int
ret_int
man_tsk
evt_exe
evt_set
frm_cnt
frm_cnta
frm_cntb
frm_sub
frm_set
frm_get
mem_reserve
mem_free
初期化部
初期タスク生成部
タスク・スイッチ部
タスク・スケジューラ部
INT1
INT2
:
SI1
:
SO1
HI
HO
タスク１
タスク２
タスク３
ユーザ作成アプリケーション
プログラム登録部
（サブタスク登録部）
ユーザ・オウン・コーディング部
ターゲット・システム用初期化部
ブート・コード部
タスク１用
ｱﾌﾟﾘｹｰｼｮﾝ・
ﾌﾟﾛｸﾞﾗﾑ
（ｻﾌﾞﾀｽｸ）
タスク２用
ｱﾌﾟﾘｹｰｼｮﾝ・
ﾌﾟﾛｸﾞﾗﾑ
（ｻﾌﾞﾀｽｸ）
タスク３用
ｱﾌﾟﾘｹｰｼｮﾝ・
ﾌﾟﾛｸﾞﾗﾑ
（ｻﾌﾞﾀｽｸ）

1.3.1 RX77016 の使用形態

本 OS は，HSM77016（ソフトウエア・シミュレータ）にタスク遷移，各タスクの MIPS 状況などの情報を提供し，組み込みシステム開発中のディバグを容易にします。

また，本 OS は，製品組み込み用とディバグ用 OS を 1 つのコードで実現するため，製品時の動作とディバグ時の動作との差異が発生しません。それは，OS の内部データを直接 HSM77016 に提供することで，ディバグ用コードを設けていないためです。

したがって，ディバグから製品化まで一環して同じ OS のコードを使用できます。

1.3.2 機能構成

本 OS は次の 7 つの機能で構成されています。

(1) タスク管理機能

タスクの状態変更やスケジューリングの基本的な機能を提供します。本 OS が管理するタスク資源は 2 つで，タスク・コンテキスト部とタスク状態部です。

タスク・コンテキスト部は，命令メモリに常駐されているものとして管理します。

タスク状態部は本 OS が管理するタスクの情報で，タスク 1 つに対して 1 つのデータ・メモリに割り付けられます。

また，最大タスク生成数の制限値を 30 としますが，タスク生成数が多いほど，OS が占有するメモリ・サイズと MIPS 値が増加します。したがって，ターゲット・システムが OS に許すメモリ・サイズと MIPS 値によっても最大タスク生成数は制限されます。その制限値を越えた場合，メモリ・オーバフローあるいは MIPS オーバフローとなります。

(2) タイマ管理機能

μPD77016 ファミリはペリフェラルにハードウエア・タイマを持たないので，本 OS では，シリアル I/O などの周期的割り込みを利用してタイマ機能を実現します。

本 OS を利用する場合，必ず周期的な割り込みが必要となります。

(3) フレーム管理機能

各タスクにフレーム・クロック・テーブルが 1 つずつ割り付けられています。このフレーム・クロック・テーブルは，各タスクごとに登録されているサブタスクの実行タイミングを制御するためのものです。

このフレーム・クロック・テーブルが示す値により，どのサブタスクを実行すべきかを判断します。同時に，1 つのタスク内において，すぐにでも実行しなければならない，または実行中のサブタスクの存在の情報を得て，ほかのタスクへの切り替えの条件の要素にします。

本 OS はタスク管理によってタスク全体の状態を把握し，フレーム管理でタスクの状態を決定します。

本 OS は，フレーム・クロック・テーブルを操作するシステム・コールとして次の 6 つを提供します。これらシステム・コールの詳細は，**2.3　システム・コール一覧**　を参照してください。

- frm_cnta
- frm_cntb
- frm_cnt
- frm_sub
- frm_set
- frm_get

(4) 割り込み管理機能

割り込みハンドラの登録／削除と割り込み終了処理を行います。

ユーザが使用する割り込みハンドラをパラメータとして与えることにより，指定された割り込みハンドラの登録コードを生成します。

ユーザは，指定された場所に割り込みハンドラのコードを記述する必要があります。

また，割り込みハンドラの終了時には，ret_int システム・コールで終了する必要があります。ただし，OS がタイマとして利用している割り込みハンドラの終了時は，man_tsk システム・コールで終了する必要があります。

本 OS は，割り込みハンドラ用のシステム・コールとして次の 3 つを提供します。　これらシステム・コールの詳細は，**2.3　システム・コール一覧**を参照してください。

- def_int
- ret_int
- man_tsk

(5) タスク・スイッチ機能

タスク・スイッチ機能は，タスク・プライオリティの一番高いタスクへのスイッチと，スイッチの対象となる両タスクの状態やタスク・コンテキストの保存と復帰を行います。

フレーム管理機能によって，各タスクの実行優先権の一番高いものが選択され，その結果に従ってタスク管理機能が，タスク・プライオリティを更新します。

タスク・プライオリティに変更があった場合，タスク・スイッチが発生します。

(6) イベント・タスク管理機能

イベント・タスク管理機能は，イベント・タスク（非周期的に実行されるタスク）を管理する機能です。これにより，本 OS が提供するシステム・コールからイベント処理プログラムに対して実行要求を発行することができます。

また，本 OS のイベント管理機能は，ロウ・プライオリティ・イベント・タスク管理とハイ・プライオリティ・イベント・タスク管理の 2 種類の管理方法を提供します。これはオプションにより選択できます。

ロウ・プライオリティ・イベント・タスク管理とは，実行要求発行後，すぐにはイベント処理を行わず，タスク・スイッチ，あるいはサブタスク終了の機会が訪れたときに，イベント処理を実行するための機能です。

この管理方法は，無駄なタスク・スイッチが発生しないため，効率の良いスケジューリングをそのまま維持できます。実行要求発行後，イベント処理プログラムをすぐに実行する必要がなければこの管理方法を使用します。

ハイ・プライオリティ・イベント・タスク管理とは，実行要求発行後，現在実行中のサブタスクおよびタスクをただちに中断し，すぐにイベント処理を行うための機能です。

この管理方法は，イベント処理を行うためにタスク・スイッチが発生しますので，タスク・スケジューリングに負荷がかかります。

また，ロウ・プライオリティ・イベント・タスク管理とハイ・プライオリティ・イベント・タスク管理は併用することが可能です。

本 OS は，イベント用のシステム・コールとして次の 2 つを提供します。これらシステム・コールの詳細は，**2.3　システム・コール一覧**を参照してください。

- evt_set
- evt_exe

(7) メモリ管理機能

メモリ管理機能は，タスク（サブタスク）間でデータ・メモリ空間を共有する場合に使用します。これにより，メモリ空間を効率的に使用するアプリケーションを作成することができます。

本 OS は，メモリ管理機能用のシステム・コールとして次の 2 つを提供します。　これらシステム・コールの詳細は，**2.3　システム・コール一覧**を参照してください。

- mem_reserve
- mem _free

1.3.3 限界値

(1) タスク ID

タスク ID は，1 から 30 までの範囲とします。最大値 30 を越えた場合，動作は不定となります。

(2) サブタスク ID

サブタスク ID は，0 から 31 までの範囲とします。最大値 31 を越えた場合，動作は不定となります。

(3) イベント ID

イベント ID は，0 から 63 までの範囲とします。最大値 63 を越えた場合，動作は不定となります。

(4) モジュール ID

モジュール ID は，0 から 63 までの範囲とします。最大値 63 を越えた場合，動作は不定となります。

1.3.4 レジスタの保証

(1) ユーザ･アプリケーション内レジスタ

サブタスク実行終了後，サブタスクとして登録されたアプリケーション・プログラム内のレジスタは，すべて保証されません。したがって，ほかのサブタスクとして登録されたアプリケーション・プログラムと，レジスタを使って値の引き渡しを行うことはできません。

(2) 割り込みハンドラ内レジスタ

割り込み発生時の R0 と DP0 を保証します。ユーザは割り込みハンドラ内で，割り込み発生時の R0 と DP0 を保存／復帰する処理を記述する必要はありません。それ以外のレジスタを割り込みハンドラ内で使用する場合，ユーザの責任において，保存／復帰処理を行ってください。

(3) システム・コール時のレジスタ

2.2 システム・コール共通インタフェースを参照してください。

1.3.5 シンボル名規定

本 OS は次の規定に従ってシンボル名を定義します。

(1) グローバル･シンボル名

グローバル･シンボル名は，次の形式に従って定義します。

a）RX77016 専用　　：_MOS_XXXX（最初に_MOS_を付加します。）

b）Plug-In Tool 専用　：_AET_XXXX（最初に_AET_を付加します。）

c）システム・コール　：システム・コール仕様に従います。

(2) OS 内部ローカル・シンボル

ローカル・シンボル名は，規定しません。

1.4 各機能の詳細

ここでは，1.3 で説明した各機能の詳細な説明を記述します。

1.4.1 タスク管理機能

(1) タスク状態遷移

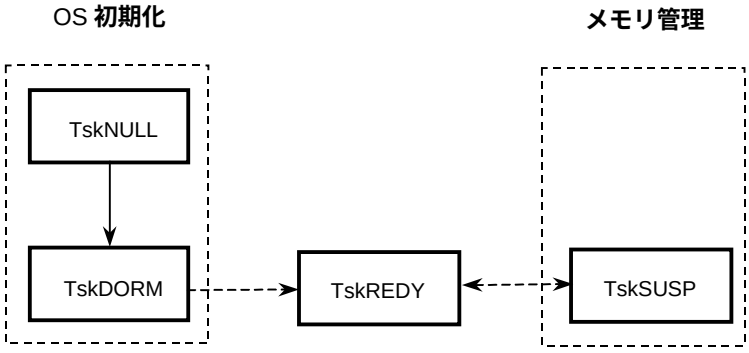
本 OS が管理するタスク状態には次のものがあります。

名　称	内　容
ヌル	タスク管理上存在する状態でタスクがないことを示します。 本 OS でこの状態になるのは OS の初期化処理が終了するまでの期間です。 初期化が終了するとドルマント状態になります。
ドルマント	起動がかけられる状態です。生成するとこの状態になります。 本 OS では，すべてのタスクをレディ状態にしたあとドルマント状態になることはありませんので，この状態もタスクが開始されるまでの期間です。
レディ	実行中でレディ・キューにつながれた状態です。 メモリ管理機能を使用しない場合，本 OS ではすべてのタスクがこの状態になります。
サスペンド	待機中で，レディ・キューに登録されていない状態です。 メモリ管理機能を使用する場合のみ存在します。

タスク状態の遷移はすべて内部的に行われますが，本 OS の特徴として，ユーザがタスク・スケージュリングに関する設計をする必要がありませんので，ユーザがタスク状態の遷移を操作することも，意識する必要もありません。

本 OS がすべての状態遷移を管理します。図 1－2 にタスク遷移図を示します。

図 1－2　タスクの状態遷移図



(2) スケジューリング

本 OS はタスク生成数+1 レベルのプライオリティでスケジューリングします。実行可能なタスクはすべてレディ・キューと呼ばれるキューに各プライオリティ・レベルごとにつながれます。この状態で，プライオリティの低いタスクはプライオリティの高いタスクが実行状態にある間は実行されません。

プライオリティの変更は OS 内部でリアルタイムに行われます。変更されたタスクのプライオリティが現在実行中のタスクより高くなる場合にタスクが切り替わります。また，実行中のタスクのプライオリティをほかのタスクよりも低くした場合もタスク切り替えが行われます。

また，本 OS では，同一プライオリティの指定はできません。

(3) タスク管理構造体

タスクは本 OS で管理するタスク構造体により，状態やタスク・コンテキストを記憶します。

タスク管理構造体には表 1—1 に示すデータが格納されます。

本 OS のタスク生成数は OS コンフィギュレーション時に定義します。タスク管理構造体はここで指定された個数分連続してとられます。タスク管理構造体の割り付けられるメモリ位置は，特定の位置に限定していませんので，開発ツールのリンカによって決定されます。

(a) タスク ID

本 OS のすべてのタスクには，それぞれ異なったタスク ID が与えられます。

タスク ID は，タスク・パラメータの登録順に 1 から昇順に割り付けます。最大値はタスク生成数と同じ値になります（タスク生成数が 4 ならばタスク ID は 1 から 4 までの間です）。

本 OS はタスク ID テーブルを持ち，タスク ID から対応するタスク管理構造体へのアドレスを得ます。タスク ID テーブルはタスク生成数+1 ワードの配列で，タスク管理構造体のアドレスを格納し，タスク ID をインデクスにして情報を引き出します。

(b) ID0

タスク ID が 0 のタスクは存在しません。タスク ID に 0 を指定した場合の動作は不定です。

(c) ID1

タスク ID が 1 のタスクは最初に起動されます。その直後にタスク ID が 1 のタスクから，そのほかのタスクを ID 順にすべて起動します。ただし，タスク ID が 1 以外のタスクはレディ状態のまま実行待ちになります。

(d) ハイ・プライオリティ・イベント・タスク管理指定がある場合

ハイ・プライオリティ・イベント・タスク管理指定がある場合，イベント・タスク用に 1 個のタスクが追加されます。イベント・タスクの ID は 1 に割り付けられるため，通常のタスクは，ID2 から割り付けられます。ただし，ハイ・プライオリティ・イベント・タスク管理指定は，システム・コールの引数などで与えるタスク ID に影響を与えません。これは，OS の内部で調整しているためです。

表1−1　タスク管理構造体の内容

データ名称	内　容
次タスク管理構造体ポインタ	次のタスク管理構造体を指すポインタを1ワードで管理します。次のタスク管理構造体がなければ，レディ・キューへのポインタとなります。
タスクのプライオリティ	2種類のプライオリティを2ワードで管理しています。 1：初期プライオリティ 起動時のプライオリティ値です。生成時に決まります。 0：実行プライオリティ 実際にスケジューリングに使用されるプライオリティです。
タスクの状態	1ワードのフィールドにタスクの状態を管理します。 タスクの状態は，ビット0，1の2ビットを使用しています。 0（TskNULL）…ヌル 1（TskDORM）…ドルマント 2（TskREDY）…レディ 3（TskSUSP）…サスペンド
スタート・アドレス	タスク起動時の開始アドレスを記憶します（1ワード）。
スタック	EIRレジスタの保存用として確保しています（1ワード）。
フレーム・クロック・テーブルへのポインタ	各タスクに1つずつフレーム・クロック・テーブルが割り付けられています。そのタスクが使用するフレーム・クロック・テーブルへの先頭アドレス＋1を記憶します（1ワード）。 フレーム・クロック・テーブルとは，タスク内で管理しているアプリケーション・プログラム単位で周期的実行タイミングを管理するテーブルです。
タスクID	タスクに割り付けられたタスクIDを記憶します（1ワード）。
コンテキスト・エリア	タスク・コンテキストのセーブ／ロード・エリアです。セーブ／ロードされるコンテキストは，常にすべての情報を格納できるように最大エリア（76ワード）を確保しています。常に最大エリアを確保しますが，これは，タスク切り替えタイミングが，アプリケーション・プログラムのどこで発生するか決定できないためです。コンテキストに関しては**第2章　システム・コール・インタフェース**を参照してください。

1.4.2 タイマ管理機能

μPD77016 ファミリのデバイスは，ハードウエア上にタイマを持たないので，本 OS では，OS 内部で擬似的なタイマを保有します。しかし，OS 単独では実現不可能なため，外部からの周期的な割り込みを利用します。そのため，本 OS を利用する場合，必ず周期的な割り込みが必要となります。

本 OS のタイマは，時間的値をとりません。周期的な割り込み 1 回につき 1 をカウントします。値が FRAME_COOUNTER_PERIOD（フレーム・カウンタ値更新モード指定），TIMER_COUNTER_PERIOD（タスク監視モード指定）を越えると 0 に戻ります。

本 OS のタイマは，サイクル時間も，それに使用する割り込みベクタも特に限定しません。任意に指定することができます。タイマ機能を利用するには，タイマとして利用する割り込みハンドラの終了時に，man_tsk システム・コールで終了する必要があります。通常の割り込みハンドラの終了時は，ret_int システム・コールで終了します。なお，ret_int, man_tsk は JMP 命令で呼び出してください。CALL 命令で呼び出した場合，正常に動作しません。

1.4.3 フレーム管理機能

各タスクにフレーム・クロック・テーブルが 1 つずつ割り付けられていますが，このフレーム・クロック・テーブルは，各タスクごとに登録されているサブタスクの実行タイミングを制御するためのものです。

表1−2　フレーム・クロック・テーブル　（1/2）

項目	データ名称	内　容
(1)	カレント・フレーム・クロック・オフセット	サブタスクのスケジューリング時にフレーム・クロック・テーブルのインデクスとして使用されます（1 ワード）。 1 フレーム・クロック・テーブルに 1 つ存在します。
(2)	フレーム・エントリ数	（3）から（6）のフレーム・クロック情報の登録数（1 ワード）です。 1 フレーム・クロック・テーブルに 1 つ存在します。 次に示す（3）から（10）の 8 つの情報は，8 つで 1 セットとなります。
(3)	フレーム・ステータス	通常，複数のフレーム・クロック情報は，それぞれ完全に独立しているため，同じフレーム・クロック情報を持つサブタスク間でも，その実行順に制約がなく不定です。フレームス・テータスは，アプリケーション・プログラムの実行順を決定します（1 ワード）。 0：サブタスクの実行順に制約を与えません（ASYNC_PRESUBTASK）。 1：フレーム・クロック情報の定義順にサブタスクを実行します。 ただし，実行可能状態になっていないサブタスクは，スキップされます（SYNC_PRESUBTASK）。 とくに必要がなければ，実行効率の良い 0 とします。 また，最初のフレーム・クロック情報のフレーム・ステータスは，必ず 0 を指定しなければなりません。
(4)	フレーム・カウンタ値	現在のフレーム・カウンタ値を示します（1 ワード）。 この値が 0 以下である場合は，実行の必要性あり，または実行中であることを示します。 1 以上である場合は，実行の必要性なし，または実行していない状態を示します。

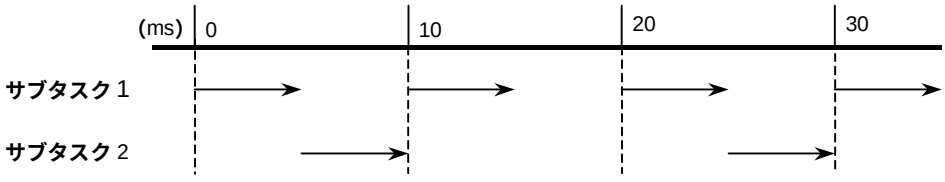
表1−2　フレーム・クロック・テーブル　(2/2)

項目	データ名称	内　容
(5)	フレーム・カウンタ A	(4) フレーム・カウンタ値に対して減算する値を示します (1 ワード)。 タイマを参照し，一定の周期で減算します。 減算する周期は，タイマ・インクリメント 16 回に 1 回，または 8 回に 1 回のように，本 OS のコンフィギュレーション時にユーザによって決定します。
(6)	フレーム・カウンタ B	(4) フレーム・カウンタ値に対して加算する値を示します (1 ワード)。 サブタスクの実行終了後，(4) フレーム・カウンタ値にこの値を加算します。
(7)	サブタスク終了フレーム・カウンタ	サブタスクが終了するときに更新される (4) フレーム・カウンタ値の更新前の値を保存します (1 ワード)。 (7) サブタスク終了フレーム・カウンタは，次に同じサブタスクが実行されて，終了する時期がくるまで値を保持します。
(8)	フレーム・オーバフロー・ステータス	(7) サブタスク終了フレーム・カウンタの値と (6) フレーム・カウンタ B の加算値が 0 以下だった場合，フレーム・オーバフローを表す 1 の値を設定します。初期値は 0 とします (1 ワード)。 ただし，フレーム・オーバフローは必ずしも致命的なものではなく，時間的遅れの状況を，危険信号として知らせるものです。なぜなら，一時的なプログラム実行時間の遅れの場合，I/O データのバッファリング方法によっては，I/O データの貯蓄により，一時的遅れをカバーするシステムなら問題ないからです。どれくらいの時間的遅れまで対応できるかは，I/O データの貯蓄数次第です。
(9)	フレーム・カウンタ最大値	(7) サブタスク終了フレーム・カウンタの過去の最大マイナス値を保存します (1 ワード)。
(10)	フレーム・オーバフロー・カウンタ	(8) フレーム・オーバフロー・ステータスがフレーム・オーバフローになった回数をカウントします (1 ワード)。

(7) から (10) の項目は通常は生成されません。コンフィギュレーション・ファイル (OS_UDEF.H) 内の_FRAME_EXTENSION オプションを 1 (TRUE) に指定すると生成されます。これらの項目は，HSM77016 でディバグする際に，よりリアルタイムなフレーム情報を参照したいときに有効です。

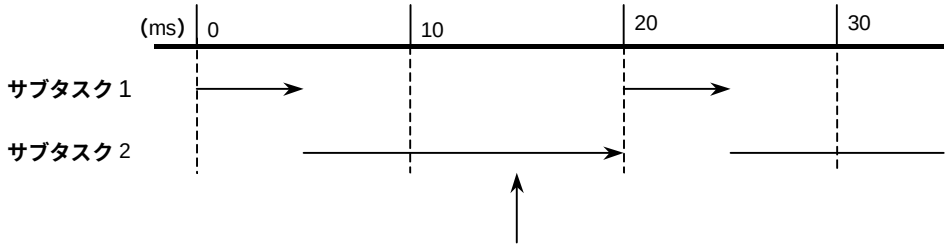
同じタスク内にあるサブタスク間では，サブタスクの切り替えは行われません。したがって，ほかのサブタスクに時間的遅れを発生させるようなサブタスクは，同じタスクに登録しないでください。そのようなサブタスクは，別のタスクに登録する必要があります。

たとえば，実行周期が 10ms で実行時間が 5ms のサブタスク 1 と，実行周期が 20ms で実行時間が 5ms のサブタスク 2 を同じタスク内で実行しても，どちらのサブタスクも時間的遅れは発生しません。



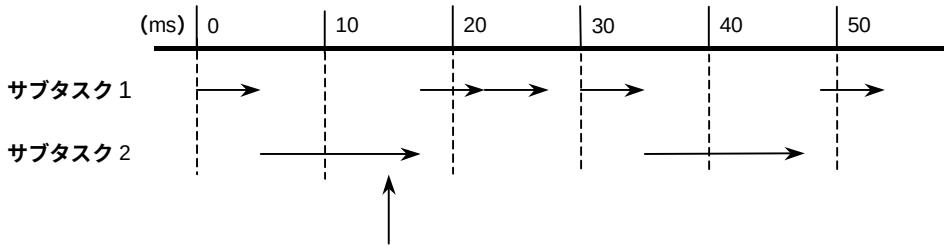
しかし，実行周期が 10ms で実行時間が 5ms のサブタスク 1 と，実行周期が 20ms で実行時間が 15ms のサブタスク 2 を同じタスクで実行した場合，サブタスク 2 の実行時間が長いため，サブタスク 1 の実行開始時間がデッドラインを越えてしまいます。

このようなサブタスク 2 は別のタスクに登録されなければなりません。



この周期にサブタスク 1 を実行できていない。

実行周期が 10ms で実行時間が 5ms のサブタスク 1 と，実行周期が 30ms で実行時間が 13ms のサブタスク 2 を同じタスクで実行した場合も，サブタスク 2 の実行時間が長いため，サブタスク 1 の実行開始時間がデッドラインを越えてしまいます。ただし，この場合は一時的な遅れであり，アプリケーション・プログラムが一時的な遅れに対応できるようになっていれば問題ありません。



この周期にサブタスク 1 が一時的に遅れる。

(1) フレーム・カウンタ値などの設定値の求め方

フレーム・カウンタ値，フレーム・カウンタ A，フレーム・カウンタ B 算出方法を次に示します。
最初にキーワードを定義します。

サブタスクの実行間隔を示す周波数を " SamplingFreq " とします。
タイマとして利用しているクロック周波数を " TimerClock " とします。
サブタスクのフレーム・サイズを " FrameSize " とします。
フレーム・クロック・テーブルの更新は，タイマが 16 回更新されるごとに１回更新されることとし，この値（16）を持つシンボルは " _FRAME_COUNTER_PERIOD " です。

フレーム・カウンタ値： TimerClock * FrameSize / _FRAME_COUNTER_PERIOD
フレーム・カウンタ A： SamplingFreq
フレーム・カウンタ B： TimerClock * FrameSize / _FRAME_COUNTER_PERIOD

たとえば，TimerClock:88200Hz　でタイマは更新されるとします。そして，SamplingFreq:44100Hz, FrameSize:16 のサブタスクであるとすると，各値は次のようになります。

フレーム・カウンタ値：88200 ‥‥ 2
フレーム・カウンタ A：44100 ‥‥ 1
フレーム・カウンタ B：88200 ‥‥ 2

上記の例のようにフレーム・カウンタ A を分母として約分をすることが可能ですので，比率の精度に問題がなければ約分してもかまいません。重要なのはそれぞれの値の比率です。
本 OS は，フレーム・クロック・テーブルを操作するシステム・コールとして次の 6 つを提供します。これらシステム・コールの詳細は，**2.3　システム・コール一覧**を参照してください。

- frm_cnta
- frm_cntb
- frm_cnt
- frm_sub
- frm_set
- frm_get

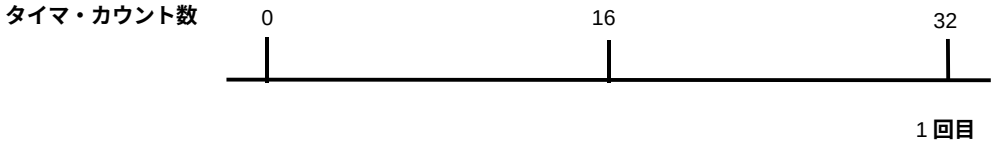
(2) フレーム・カウンタ値更新タイミング

フレーム・カウンタ値を更新する時間間隔のパターンをフレーム・カウンタ値更新モードと呼びます。フレーム・カウンタ値更新モードは，コンフィギュレーション・ファイル（OS_UDEF.H）内の_FRAME_COUNTER_PERIOD のパラメータで指定します。パラメータは，タイマ・カウント数で表し，1 以上の値とします。0 以下の値を指定した場合，動作は保証されません。

次にタイマ・カウント数によるフレーム・カウンタ値更新タイミングを示します。

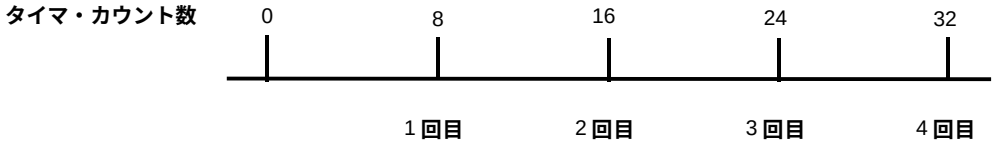
＜タイマ・カウント数 32 のフレーム・カウンタ更新モード＞

タイマが 32 回更新される間に１回更新します。



＜タイマ・カウント数 8 のフレーム・カウンタ更新モード＞

タイマが 8 回更新される間に１回更新します。



1.4.4 割り込み管理機能

本 OS では割り込みハンドラの登録／削除と，割り込み終了を提供します。

(1) 割り込みの登録と削除

コンフィギュレーション・ファイルに，使用する割り込みハンドラを定義することにより，割り込みハンドラをコールするコードを生成しますが，ユーザは指定された位置に割り込みハンドラ・コードを定義しなければなりません。

通常は，最初に定義された割り込みハンドラ・コードへのコール先を変更することはできませんが，オプションにより，割り込みハンドラ・コードへのコール先をタスク内から登録／削除できるようになります。本 OS は，割り込みハンドラを登録／削除するシステム・コールとして def_int システム・コールを提供します。def_int システム・コールは，ユーザの割り込みハンドラの開始アドレスと登録する割り込み要因をパラメータで渡すことで，指定したハンドラを登録／削除します。

ユーザが作成する割り込みハンドラ内では，R0，DP0 レジスタは一時退避することなく使用できます。そのほかのレジスタを使用する場合は，ユーザによる退避／復帰処理が必要です。

また，割り込みハンドラの終了時には，ret_int システム・コールで終了する必要がありますが OS がタイマとして利用している割り込みハンドラの終了時は，man_tsk システム・コールで終了となります。

(2) 割り込みネスト

通常，本 OS は割り込みネストを許しませんが，_USED_INTNEST オプション指定により，可能になります。

ret_int システム・コールは割り込みのネスト状態を調べ，元のタスク走行に戻る直前まで処理を遅延させます（man_tsk システム・コールも同等の機能を持ちます）。

割り込みネストを使用しない場合は，割り込みネストを使用しないように_USED_INTNEST オプションを指定してください。ret_int または man_tsk システム・コールの処理時間分短くなります。

ユーザは，割り込みネスト中も R0, DP0 レジスタを一時退避することなく使用できます。

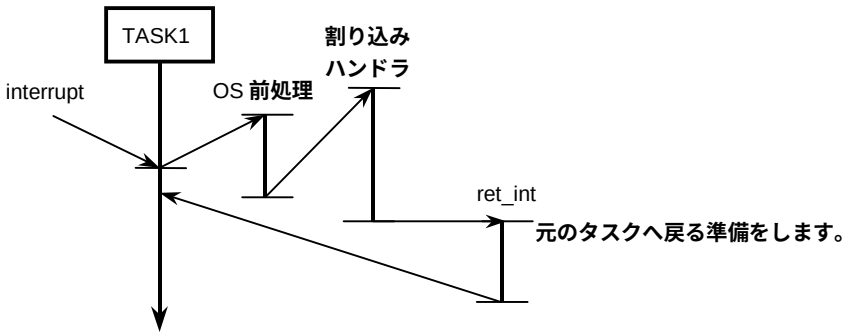
(3) 割り込みの終了処理

通常の割り込み処理の流れを示します。

割り込みハンドラの終了処理を行うために，ret_int システム・コールを提供します。

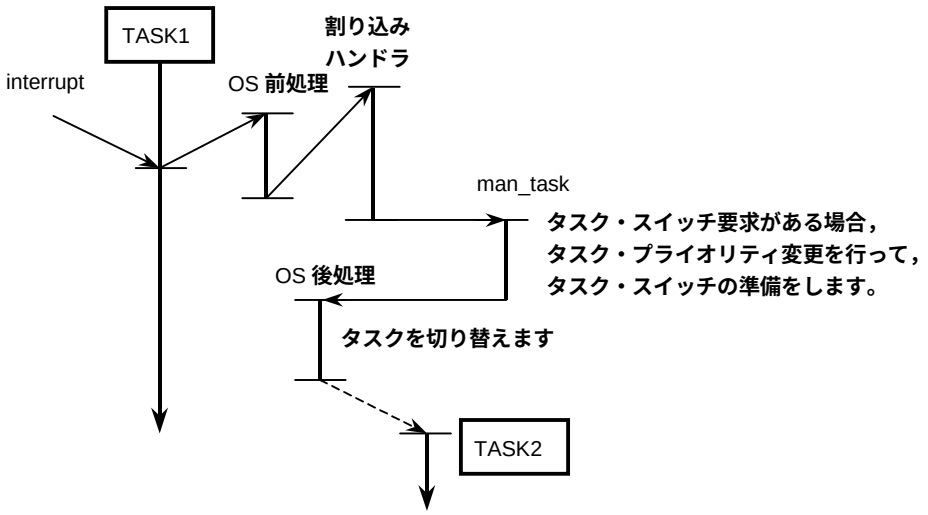
ret_int システム・コールは，R0,DP0 レジスタの復帰を行います（**図 1−3** 参照）。

図 1−3　割り込みハンドラからの ret_int システム・コール



割り込みハンドラ中からのタスク・スイッチを行うために，man_tsk システム・コールを提供します。man_tsk システム・コールは，各タスクの状態を監視し，現在実行中のタスクよりもタスク・プライオリティの高いタスクから実行要求がある場合に，タスク・スイッチを行います。また，man_tsk システム・コールはタイマの更新，フレーム・クロック・テーブルの更新も行います。man_tsk システム・コールは ret_int システム・コールと異なり，タイマとして利用できる割り込みハンドラでのみ呼び出されなければなりません（**図 1−4** 参照）。

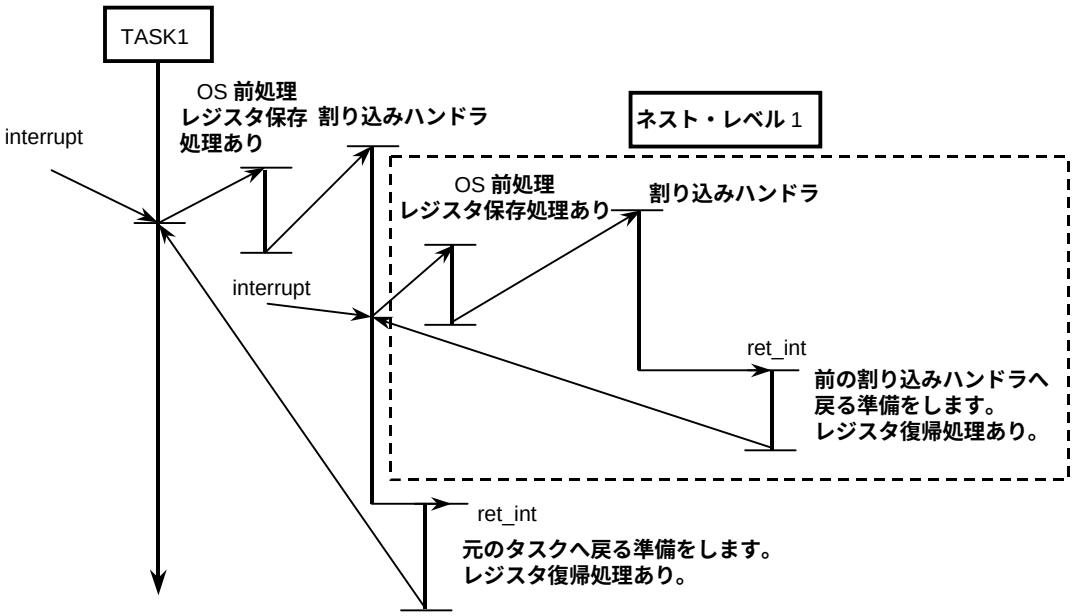
図1−4　割り込みハンドラからの man_tsk システム・コール



割り込みネスト処理の流れを示します。

すべてのネスト・レベルにおいて，R0,DP0 レジスタの保存／復帰を行いますので，ユーザは，割り込みネスト中も R0, DP0 レジスタを一時退避することなく使用できます（図1−5 参照）。

図1−5　割り込みハンドラがネストした場合の動作



1.4.5 タスク・スイッチ機能

(1) タスク・プライオリティ管理

タスクのプライオリティは次のように管理します。

ユーザが指定するプライオリティは，1 からタスク生成数と同じ値までとしていますので，タスク・レディ・キューに割り当てられるのは通常１番地からタスク生成数と同じ値の番地です。

タスク・プライオリティの変更によって，タスク・レディ・キューへの登録順が変わりますが，登録変更方法は次のようになっています。

(a) 最も実行優先度の高いタスクをタスク・レディ・キューの 0 番地に移動します。

(b) タスク・レディ・キューの 0 番地に移動するとき，0 番地に前のタスクが存在する場合，前のタスクを元のプライオリティを示す番地に戻します。

(c) 1 番地に登録されているプライオリティ 1 のタスクは，移動することはありません。
1 番地のタスクが最も実行優先度が高い場合，0 番地に登録されているタスクを元のプライオリティを示す番地に戻します。0 番地にタスク登録がなければ，移動は発生しません。

この結果，タスク・レディ・キューに登録されているもっともプライオリティの高いタスクが実行対象になります。メモリ管理機能を使用しない場合は，タスク・レディ・キューの 0 番地または 1 番地にあるタスクが実行対象となります。

タスク・レディ・キュー

0	プライオリティ 0
1	プライオリティ 1
2	プライオリティ 2
3	プライオリティ 3
:	:
タスク生成数と同じ値までの番地	:

(2) タスク・スイッチ

タスク・スイッチ機能は，タスク・プライオリティの一番高いタスクへのスイッチと，スイッチの対象となる両タスク状態やタスク・コンテキストの保存と復帰を行います。

タスク・プライオリティに変更があった場合，タスク・スイッチが発生します。

タスク・スイッチは次の 3 つの処理から構成されます。

処理 A: 実行要求のあるタスクの検索（約 100－200 実行サイクル）

フレーム・クロック・テーブルから，実行要求のあるタスクを確認します。

処理 B: タスク・プライオリティの変更（約 100－150 実行サイクル）

処理 A で実行要求のあるタスクが存在する場合，タスク・プライオリティなどのタスク状態の変更を行います。ただし，タスク状態の変更のみで，タスク・スイッチは行われません。

処理 C: タスク・コンテキストの保存／復帰（約 200－300 実行サイクル）

割り込みハンドラからタスクに戻るときにタスクが切り替わりますが，そのときタスク・コンテキストの保存／復帰が行われます。

(3) タスク監視

(a) 周期的タスク監視

本 OS のタスクを監視するタイミングは一定時間間隔で行いますが，時間間隔はユーザが任意に決定できます。

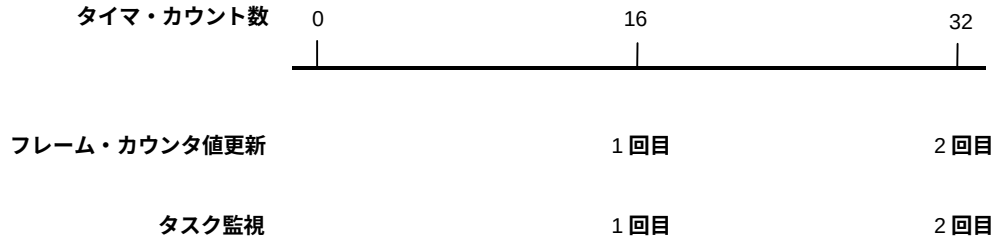
常にタスクを監視していると，タスク監視処理の占有時間が長くなります。

本 OS では，このタスクを監視する時間間隔を選択可能にし，ターゲット・システムに適したタイミングでタスクを監視することができます。このタスクを監視する時間間隔のパターンをタスク監視モード （_TIMER_COUNTER_PERIOD）と呼びます。

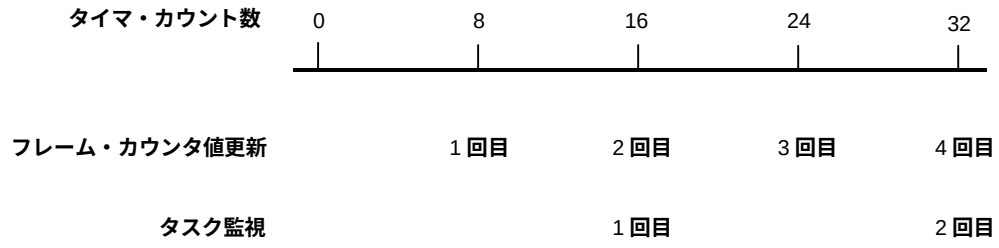
タスク監視モードは，フレーム・カウンタ値更新モード（_FRAME_COUNTER_PERIOD）で指定されたフレーム・カウンタ値更新タイミングをベースにした値です。たとえば，タスク監視モードが 1 の場合，フレーム・カウンタ値更新と同じタイミングでタスクを監視します。タスク監視モードが 2 の場合，フレーム・カウンタ値更新が 2 回行われる間にタスクを監視を 1 回行います。

次にタスク監視タイミングを示します。

・フレーム・カウンタ値更新モード 16，タスク監視モード 1



・フレーム・カウンタ値更新モード 8，タスク監視モード 2



本 OS は，タスク・プライオリティの高いタスクから低い方へ順に実行しますので，その流れから，強制的にタスク・プライオリティの高いタスクへ変更するために周期的タスク監視が実行されます。

したがって，モードを選択する基準は，強制的にタスク・プライオリティの高いタスクへの変更が必要なタイミングとなります。むだなタスク監視を実行することは実行効率の低下につながります。

(b) 非周期的タスク監視

本 OS は（a）の周期的タスク監視以外に非周期的タスク監視も行います。

これは，タスク内にあるサブタスクの実行要求がなくなったときに，タスク監視処理を行い，実行要求のあるタスクへの切り替えを行います。

これにより，タスクのアイドリング時間を少なくし，効率良くタスクを実行できるようになります。

1.4.6 イベント・タスク管理機能

イベント・タスク管理機能は，イベント・タスク（非周期的に実行されるタスク）を管理する機能です。これにより，本 OS が提供するシステム・コールからイベント処理プログラムに対して実行要求を発行することができます。

また，本 OS のイベント管理機能は，ロウ・プライオリティ・イベント・タスク管理とハイ・プライオリティ・イベント・タスク管理の 2 種類の管理方法を提供します。これはオプションにより選択できます。

(1) ロウ・プライオリティ・イベント・タスク管理

ロウ・プライオリティ・イベント・タスク管理とは，実行要求発行後，すぐにはイベント処理を行わず，タスク・スイッチあるいはサブタスク終了の機会が訪れたときに，その間にイベント処理を実行するための機能です。イベント処理プログラムへの分岐は，ID=1 のタスクから行われます（ハイ・プライオリティ・イベント・タスク管理を指定していると ID=2 となります）。新規にイベント用のタスクは生成しません。この管理方法は，むだなタスク・スイッチが発生しないため，効率の良いスケジューリングをそのまま維持できます。実行要求発行後，イベント処理プログラムをすぐに実行する必要がなければこの管理方法を使用します。

ロウ・プライオリティ・イベント・タスク管理を指定することにより，消費するメモリ・サイズは，命令メモリ約 75 ワード，X データ・メモリ約 25 ワードです。通常のタスク管理の処理速度に与える影響は，フレーム・クロック・テーブルにフレーム・クロック情報が１つ（1 セット）追加される分だけ，フレーム・クロック情報の更新／検索に時間を要しますが，ほとんど通常の処理速度に影響はありません。

(2) ハイ・プライオリティ・イベント・タスク管理

ハイ・プライオリティ・イベント・タスク管理とは，実行要求発行後，現在実行中のサブタスクおよびタスクをただちに中断し，すぐにイベント処理を行うための機能です。イベント用のタスクが内部的に ID=1 のタスクで生成され，イベント処理プログラムへの分岐は，そのイベント用のタスクから行われます。この管理方法は，イベント処理を行うためにタスク・スイッチが発生しますので，タスク・スケジューリングに負荷がかかります。

ハイ・プライオリティ・イベント・タスク管理を指定することにより，消費するメモリ・サイズは，命令メモリ約 205 ワード，X データ・メモリ約 120 ワードです。通常のタスク管理の処理速度に与える影響はありませんが，前述したようにイベント処理を行う場合には，そのたびにタスク・スイッチが発生しますので，その負荷は大きくなります。

(3) 両イベント・タスク管理の併用

ロウ・プライオリティ・イベント・タスク管理とハイ・プライオリティ・イベント・タスク管理は併用することが可能です。

ロウ・プライオリティ・イベント・タスク管理とハイ・プライオリティ・イベント・タスク管理を指定した場合に消費するメモリ・サイズは，命令メモリ約 235 ワード，X データ・メモリ約 125 ワードです。

(4) イベント処理プログラム内のイベント用システム・コール使用

イベント処理プログラム内でイベント用システム・コールの使用が可能です。

ただし，イベント処理プログラム内からハイ・プライオリティ・イベント管理処理でイベント処理プログラムを実行する場合，次の点に注意します。

ハイ・プライオリティ・イベント管理処理で実行中のイベント処理プログラムから，ハイ・プライオリティ・イベント管理処理でイベント処理プログラムを実行しようとした場合，現在実行中のイベント処理プログラムが終了してから実行されます。

ロウ・プライオリティ・イベント管理処理で実行中のイベント処理プログラムからハイ・プライオリティ・イベント管理処理で同じイベント処理プログラムを実行しようとした場合，現在実行中のイベント処理プログラムを中断し，同じイベント処理プログラムを実行します（再帰的呼び出しが可能です）。ただし，そのときのイベント処理プログラム内で使用しているデータは，イベント処理プログラム呼び出し前と呼び出し後で異なるため，保持したいデータの保存／復帰は，イベント処理プログラム内で行わなければなりません。

本 OS は，イベント用のシステム・コールとして次の 2 つを提供します。　これらシステム・コールの詳細は，2.3　システム・コール一覧を参照してください。

- evt_set
- evt_exe

1.4.7　メモリ管理機能

本 OS は，メモリ管理機能として，データ・メモリ空間が使用中であるかどうかを確認するための機能を持っています。この機能を使用する場合，OS コンフィギュレーション時に_MEM_RESERVE **オプションを** TRUE **に設定してください。なお，システム・コール内でタスク・スイッチを行うため，割り込みハンドラ内での当システム・コールの使用を禁止します。**

mem_reserve **システム・コールを発行することにより，データ・メモリ空間を排他的に使用することができます。このマニュアルでは排他的にメモリを使用するための操作を，“ 予約 ” と表現します。予約しようとしたデータ・メモリ空間が未使用であった場合，予約情報を** OS **が管理するテーブルに登録し，データ・メモリ空間を予約します。予約しようとしたデータ・メモリ空間がすでに予約されていた場合は，予約動作を失敗とするか，または，予約ができるようになるまで現在実行中のタスクを停止させるかのどちらかを選択することができます。ただし，タスクを停止させる場合，次の** 2 **点に注意してください。**

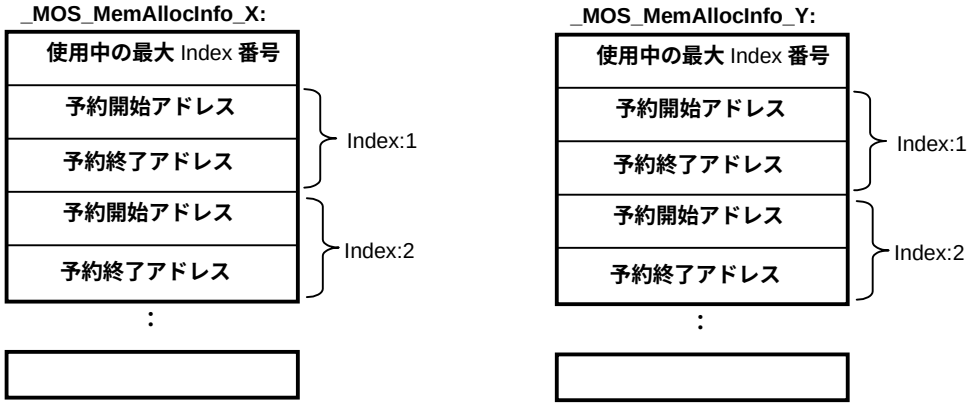
- **デッドロックの発生**
- **メモリ空間を予約しているサブタスクが，再度同じメモリ空間の予約を行なった場合にも，予約に失敗する。**

mem_free **システム・コールは，**mem_reserve **システム・コールで予約した領域を開放し，再利用を可能にします。停止しているタスクが存在する場合，タスクを再開できるかどうかを確認し，再開できる場合にタスクの停止を解除します。**

(1) 予約情報管理テーブル

予約情報管理テーブルは次のような構成になっています。

X メモリ用の登録数の最大値を_MEMBUFSZ_X オプションにより指定できます。Y メモリ用の登録数の最大値を_MEMBUFSZ_Y オプションにより指定できます。



予約開始アドレスが 0xFFFF で，予約終了アドレスが 0x0000 だった場合、予約情報は無効であることを意味します。

1.5 RX77016 のメモリ構成

ここでは，本 OS のメモリ構成について説明します。

本 OS は，DSP のメモリを次の 2 つのエリアに分けています。

- RX77016 本体コード部
- OS メモリ・エリア部

1.5.1 RX77016 本体コード部

本 OS のプログラム本体です。次のモジュールで構成されています。

(1) 初期化部（カーネル内）
(2) 初期タスク生成部（カーネル内）
(3) メイン・タスク・ヘッダ部（カーネル内）
(4) タスク・スイッチ部（カーネル内）
(5) システム・コール部（カーネル内）
(6) コンフィギュレーション部
(7) ユーザ・オウン・コーディング部

(1) 初期化部（カーネル内）

本 OS が使用する作業領域および汎用レジスタの初期化と割り込みハンドラの登録を行います。ここでは，メモリ・システムの検査やウエイト設定は行いませんので，あらかじめ必要な外部デバイスの設定はターゲット・システム用初期化部（_MOS_TargetSysInit サブルーチン）の中で行ってください。

(2) 初期タスク生成部（カーネル内）

初期データに従ってタスクの生成と ID＝1 のタスク起動を行います。タスクの起動直後の割り込みステータス（SR，EIR）はコンフィギュレーション時に決定することができます。

(3) メイン・タスク・ヘッダ部（カーネル内）

ID＝1 のタスク起動後に，ほかのすべてのタスクの起動を行い，タスクの初期化を行います。

(4) タスク・スイッチ部（カーネル内）

タスク・スイッチの機能だけを提供する部分です。タスク・スイッチを必要とするすべてのシステム・コールはこの部分を呼び出します。

(5) システム・コール部（カーネル内）

ret_int, man_tsk, def_int, frm_cnta, frm_cntb, frm_cnt, frm_sub, frm_set, frm_get, evt_set, evt_exe, mem_reserve, mem_free のシステム・コールから成ります。

def_int, frm_cnta, frm_cntb, frm_cnt, frm_sub, frm_set, frm_get, evt_set, evt_exe, mem_reserve, mem_free の呼び出しは CALL 命令ですが，ret_int, man_tsk の呼び出しは，JMP 命令を使用します。

(6) コンフィギュレーション部

コンフィギュレーション部は，次の 3 つに分類することができます。コンフィギュレーション部のすべてのパラメータは，コンフィギュレーション・ツールによって自動生成されます。

(a) コンフィギュレーション・パラメータ設定部

OS_UDEF.H に，OS をコンフィギュレーションするためのパラメータを定義します。

(b) ユーザ作成アプリケーション・プログラム登録部

OS_TASKX.ASM に，ユーザ作成のアプリケーション・プログラム名を登録します。OS は，登録したアプリケーション・プログラムをサブタスクとして管理します。

(c) タスク・パラメータ設定部

OS_IDATA.ASM に，タスク・パラメータを設定します。タスク・パラメータは，各サブタスクの実行タイミングを定義するものであり，OS は，このパラメータをもとにタスク／サブタスクのスケジューリングを行います。

(7) ユーザ・オウン・コーディング部

ユーザ・オウン・コーディング部は，次の 3 つに分類することができます。当部は，ユーザが直接記述しなければならないコード部分のファイルを提供するものです。詳細は，**第 3 章　ユーザ記述ファイル**を参照してください。

(a) ターゲット・システム用初期化部

OS の起動直後に 1 度だけ呼び出されます。

ユーザは OS_TSINI.ASM 内の_MOS_TargetSysInit に，ペリフェラル・レジスタの初期化とユーザ用変数等の初期化を記述できます。

_MOS_TargetSysIni ではすべてのレジスタを自由に使うことができますが，ここで記述したレジスタの初期値がタスクの初期値として使用できるものではありません。

_MOS_TargetSysIni サブルーチンは CALL 命令で呼び出されるため，処理の終了に RET 命令を使用します。

(b) ブート・コード部

OS_BOOT.ASM 内の_MOS_ReBootProc にブート・コードを定義してください。

本コードは，DSP 起動直後に実行されます。

(c) 割り込みハンドラ・コード部

OS_IOHDL.ASM に，それぞれの割り込みハンドラのサブルーチン（JMP RET_INT 命令のみ）を準備していますので，使用する割り込みハンドラのサブルーチンにコードを定義してください。また，割り込みハンドラの登録は，コンフィギュレーション時に使用する割り込みハンドラを指定する必要があります。指定されていない割り込みハンドラのコードはオブジェクト・コードへ出力されません。

1.5.2 OS メモリ・エリア部

OS メモリ・エリアには次のデータがあります。

（1）システム作業領域（16 ワード）
（2）タスク ID テーブル（タスク生成数＋1 ワード）
（3）タスク・レディ・キュー（タスク生成数＋1 ワード）
（4）タスク管理構造体
（5）タスク・スケジューリング・テーブル

(1) システム作業領域

16 ワードの領域に作業用のデータを保存します。

(2) タスク ID テーブル

タスク ID からタスク管理構造体のアドレスに変更するための配列データです（**1.4.1　タスク管理機能**を参照）。

(3) タスク・レディ・キュー

タスク生成数に比例するユーザ・タスク・レベル＋1 システム・レベルのプライオリティ・キューです。

ユーザは，1 からタスク生成数までのプライオリティをタスクに割り付けることができます（**1.4.1　タスク管理機能**を参照）。

(4) タスク管理構造体

各タスクごとのプライオリティや状態などを管理する構造体です（**1.4.1　タスク管理機能**を参照）。

(5) タスク・スケジューリング・テーブル

フレーム・クロック・テーブルを含むタスク・スケジューリングに必要な作業領域です（**1.4.3　フレーム管理機能**を参照）。

第 2 章　システム・コール・インタフェース

この章では，システム・コールのインタフェースについて説明します。

2.1　概　　要

本 OS のシステム・コールはアセンブラ・レベルからの呼び出しとし，単純にアセンブラ命令の CALL または JMP によって呼び出します。パラメータの引き渡しはレジスタを使用します。

2.2　システム・コール共通インタフェース

2.2.1　作業レジスタ

パラメータの引き渡しに使用されるレジスタは，システム・コール内で作業レジスタとして使用されるため，システム・コール終了後のそのレジスタ値は不定となります。パラメータの引き渡しに使用されるレジスタでその値が破壊されないものについては，**2.3 システム・コール一覧** の各システム・コールの説明を参照してください。

パラメータの引き渡しに使用されていないレジスタは，システム・コール終了後のレジスタ値が，システム・コール呼び出し前と同値であることを保証します。ただし，ret_int と man_tsk は，システム・コール呼び出し前の R0 と DP0 の値を保証しません。なぜなら，ret_int と man_tsk は，システム・コール呼び出し前ではなく，割り込み発生時の R0 と DP0 の値を保証するからです。

これらの破壊されるレジスタの値を保持したい場合は，システム・コールの呼び出し前に退避しておく必要があります。

2.2.2　返り値

システム・コールの返り値は，コンフィギュレーション時に，_OSERROR_CHK で有効にしたシステム・コールのみ R0L に格納して返します。R0E, R0H には 0 が格納されます。

表 2−1　返り値一覧

エラー名称	値	内　容
TE_OK	0x0000	正常終了しました。
TE_NOEXIST	0x0001	指定タスクは存在しません。
TE_BADID	0x0010	指定タスク ID は範囲外です。
TE_EVTSTKOV	0x0020	イベント ID スタックがオーバフローしています。
TE_BADEVT	0x0021	指定イベント・コマンドに誤りがあります。
TE_NOEXISTEVT	0x0022	指定メッセージ ID のイベントが存在しません。
TE_OVERLAPPED	0x0000	指定された領域はすでに使用されています。
TE_RESERVEERR	0xffff	予約できません。
TE_BADHANDLE	0x0091	指定されたハンドル値が間違っています。

2.3　システム・コール一覧

表 2－2　システム・コール一覧

ユーザ用システム・コール名	機　　能
def_int	割り込みハンドラを登録します。
ret_int	割り込みハンドラを終了します。
man_tsk	タイマの更新，フレーム・カウンタ・テーブルの更新，タスク・スイッチを行います。
frm_cnta	フレーム・クロック・テーブルのフレーム・カウンタ A を変更します。
frm_cntb	フレーム・クロック・テーブルのフレーム・カウンタ B を変更します。
frm_cnt	フレーム・クロック・テーブルのフレーム・カウンタを 0（実行可能状態）にします。
frm_sub	フレーム・カウンタの値を減算します。
frm_set	スケジュールに使用するフレーム・クロック情報を再設定します。
frm_get	スケジュールに使用しているフレーム・クロック情報を取得します。
evt_set	イベント ID スタックにイベント ID を設定します。
evt_exe	イベント処理プログラムを実行します。
mem_reserve	メモリの予約を行います。
mem_free	メモリを開放します。

2.3.1　def_int

〔機　　能〕

割り込みハンドラを登録します。

〔形　　式〕

・入力パラメータ

R0L＝ハンドラ番号

R1L＝ハンドラ・アドレス

・呼び出し方

CALL def_int;

・リターン値

R0L＝削除された割り込みハンドラのアドレス

〔処　　理〕

R0L で指定されたハンドラ番号に対して R1L で指定したハンドラを間接登録します。R0L で指定できるハンドラ番号は次のとおりです。

指定番号	対応割り込み名	指定番号	対応割り込み名
0	INT1 外部割り込み	5	SO1 シリアル割り込み
1	INT2 外部割り込み	6	SI2 シリアル割り込み
2	INT3 外部割り込み	7	SO2 シリアル割り込み
3	INT4 外部割り込み	8	HI ホスト割り込み
4	SI1 シリアル割り込み	9	HO ホスト割り込み

R1L に 0 が指定された場合，指定ハンドラ番号に対する割り込みハンドラを削除します。割り込みハンドラが削除されている状態で対応する割り込みが発生した場合，直ちに reti 命令を発行して元の処理に戻ります。

2.3.2　ret_int

〔機　　能〕

割り込みハンドラを終了します。

〔形　　式〕

・入力パラメータ

なし

・呼び出し方

JMP ret_int;

・リターン値

なし

〔処　　理〕

このシステム・コールは割り込みハンドラの終了時に必ず発行されなければなりません。ただし，man_tsk システム・コールを発行する場合は例外で，man_tsk は，ret_int を内部で発行しています。

ret_int システム・コールを使わずに直接アセンブラの reti 命令を発行した場合，以後の動作は保証されません。

ret_int システム・コールは返り値を持たずアセンブラの reti 命令の代わりに使用します。

コール／ループ・スタック・レベルが割り込み発生直後と異なった状態で，本システム・コールを使用した場合の動作は保証されません。

2.3.3　man_tsk

〔機　　能〕

タイマの更新，フレーム・クロック・テーブルの更新，周期的タスク監視，必要に応じてタスク・スイッチを行います。

〔形　　式〕

・入力パラメータ

なし

・呼び出し方

JMP man_tsk;

・リターン値

なし

〔処　　理〕

ハードウエア・タイマを使用しない場合，本システム・コールを周期的かつ一定の時間で割り込みの入る割り込みハンドラ処理の最後に発行します。この場合，ret_int システム・コールの発行は必要ありません。

本システム・コールを発行すると，その都度タイマは更新されます。

また，フレーム・クロック・テーブルから，タスク・プライオリティ変更の必要性を確認します。その結果，タスク・プライオリティ変更の必要性がある場合，変更処理後，タスク・スイッチを行います。

2.3.4　frm_cnta

〔機　　能〕

フレーム・クロック・テーブルのフレーム・カウンタ A を変更します。

本システム・コールを使用することにより，周期的に実行するサブタスクの実行開始／停止操作が可能になります。停止する指示を与えたとき，該当するサブタスクが実行中だった場合，そのサブタスクの処理が終了してから有効になります。

〔形　　式〕

・入力パラメータ

R0L＝変更するフレーム・カウンタ A のタスク ID

タスク ID は，OS_IDATA.ASM で定義するタスク・パラメータの登録順に，1 から昇順に割り当てられた値です。本パラメータのタスク ID は，_USED_EVENTASKHIGH の指定による影響を受けません。

R1L＝変更するフレーム・カウンタ A のサブタスク ID

サブタスク ID は，OS_IDATA.ASM で定義するフレーム情報の登録順に，タスクごとに 0 から昇順に割り当てられた値です。本パラメータのサブタスク ID は，_USED_EVENTASKLOW の指定による影響を受けません。

R2L＝フレーム・カウンタ A に設定する値　（frame_sub）

フレーム・カウンタ A の初期値は，OS_IDATA.ASM の frame_sub の値です。フレーム・カウンタ A については，表 1−2　フレーム・クロック・テーブルを参照してください。

・呼び出し方

R0L＝1；タスク ID

R1L＝0；サブタスク ID

R2L＝2；フレーム・カウンタ A

```
CALL frm_cnta;
```

・リターン値

TE_OK…正常終了

TE_BADID…タスク ID が指定可能範囲外

〔処　　理〕

本システム・コールは，サブタスク内，割り込みハンドラ内から呼び出し可能です。

作業用レジスタとして，R0，R1 を使用しますので，R0，R1 の内容を保持したい場合，本システム・コールをコールする前にセーブする必要があります。R2 の内容は破壊されません。

サブタスクを停止したい場合，フレーム・カウンタ A を 0 にします。

実行を開始したい場合は，フレーム・カウンタ A の算出方法に従った値を設定します。

フレーム・カウンタ A の算出方法は，1.4.3（1）フレーム・カウンタ値などの設定値の求め方を参照してください。

2.3.5　frm_cntb

〔機　　能〕

フレーム・クロック・テーブルのフレーム・カウンタ B を変更します。

この変更は，システム・コール発行直後から有効になります。

〔形　　式〕

・入力パラメータ

R0L＝変更するフレーム・カウンタ B のタスク ID

タスク ID は，OS_IDATA.ASM で定義するタスク・パラメータの登録順に，1 から昇順に割り当てられた値です。本パラメータのタスク ID は，_USED_EVENTASKHIGH の指定による影響を受けません。

R1L＝変更するフレーム・カウンタ B のサブタスク ID

サブタスク ID は，OS_IDATA.ASM で定義するフレーム情報の登録順に，タスクごとに 0 から昇順に割り当てられた値です。本パラメータのサブタスク ID は，_USED_EVENTASKLOW の指定による影響を受けません。

R2L＝フレーム・カウンタ B に設定する値　（frame_add）

フレーム・カウンタ B の初期値は，OS_IDATA.ASM の frame_add の値です。フレーム・カウンタ B については，表 1－2　フレーム・クロック・テーブルを参照してください。

・呼び出し方

R0L＝1；タスク ID

R1L＝0；サブタスク ID

R2L＝2；フレーム・カウンタ B

```
CALL frm_cntb;
```

・リターン値

TE_OK…正常終了

TE_BADID…タスク ID が指定可能範囲外

〔処　　理〕

本システム・コールは，サブタスク内，割り込みハンドラ内から呼び出し可能です。

作業用レジスタとして，R0，R1 を使用しますので，R0，R1 の内容を保持したい場合，本システム・コールをコールする前にセーブする必要があります。R2 の内容は破壊されません。

フレーム・カウンタ B に設定する値は，フレーム・カウンタ B の算出方法に従った値を設定します。フレーム・カウンタ B の算出方法は，1.4.3（1）フレーム・カウンタ値などの設定値の求め方を参照してください。

2.3.6　frm_cnt

〔機　　能〕

フレーム・クロック・テーブルのフレーム・カウンタを 0（実行可能状態）にします。

本システム・コールを使用することにより，一時的に実行するサブタスクの実行開始が可能になります。イベント・タスクと異なる点は，通常のサブタスクと同じように扱われることです。そのため，ほかのサブタスクとの実行順の関連づけが可能となります。

〔形　　式〕

・入力パラメータ

R0L＝変更するフレーム・カウンタのタスク ID

タスク ID は，OS_IDATA.ASM で定義するタスク・パラメータの登録順に，1 から昇順に割り当てられた値です。本パラメータのタスク ID は，_USED_EVENTASKHIGH の指定による影響を受けません。

R1L＝変更するフレーム・カウンタのサブタスク ID

サブタスク ID は，OS_IDATA.ASM で定義するフレーム情報の登録順に，タスクごとに 0 から昇順に割り当てられた値です。本パラメータのサブタスク ID は，_USED_EVENTASKLOW の指定による影響を受けません。

フレーム・カウンタの初期値は，OS_IDATA.ASM の frame_counter の値です。フレーム・カウンタについては，**表 1−2　フレーム・クロック・テーブル**を参照してください。

・呼び出し方

R0L＝1；タスク ID

R1L＝0；サブタスク ID

CALL frm_cnt；

・リターン値

TE_OK…正常終了

TE_BADID…タスク ID が指定可能範囲外

〔処　　理〕

本システム・コールは，サブタスク内，割り込みハンドラ内から呼び出し可能です。

作業用レジスタとして，R0，R1 を使用しますので，R0，R1 の内容を保持したい場合，本システム・コールをコールする前にセーブする必要があります。

OS_IDATA.ASM のフレーム情報は，次のとおりです。

```
DW      1    ;フレーム・カウンタ　（frame_counter）
DW      0    ;フレーム・カウンタ A（frame_sub）
DW      1    ;フレーム・カウンタ B（frame_add）
```

フレーム・カウンタ A を 0 にして，初期状態を停止状態にします。

2.3.7　frm_sub

〔機　　能〕

本 OS がスケジュールに使用しているフレーム・クロック・テーブルのフレーム・カウンタから，システム・コールの入力パラメータで指定した値を減算することにより，サブタスクの実行要求を管理することができます。

〔形　　式〕

・入力パラメータ

R0L＝変更するフレーム・カウンタのタスク ID

タスク ID は，OS_IDATA.ASM で定義するタスク・パラメータの登録順に，1 から昇順に割り当てられた値です。本パラメータのタスク ID は，_USED_EVENTASKHIGH の指定による影響を受けません。

R1L＝変更するフレーム・カウンタのサブタスク ID

サブタスク ID は，OS_IDATA.ASM で定義するフレーム情報の登録順に，タスクごとに 0 から昇順に割り当てられた値です。本パラメータのサブタスク ID は，_USED_EVENTASKLOW の指定による影響を受けません。

R2L＝変更するフレーム・カウンタから減算する値

・呼び出し方

R0L＝1；タスク ID

R1L＝0；サブタスク ID

R2L＝2；フレーム・カウンタから減算する値

```
CALL frm_sub;
```

・リターン値

TE_OK…正常終了

TE_BADID…タスク ID が指定可能範囲外

〔処　　理〕

本システム・コールは，サブタスク内，割り込みハンドラ内から呼び出し可能です。

作業用レジスタとして，R0，R1 を使用しますので，R0，R1 の内容を保持したい場合，本システム・コールを発行する前にセーブする必要があります。R2 の内容は破壊されません。

本システム・コールを発行した場合，入力パラメータ R0L,R1L で指定したサブタスクのフレーム・カウンタ値から，入力パラメータ R2L が示す値を減算し，その結果を指定したサブタスクのフレーム・カウンタ値とします。

サブタスクの実行を開始したい場合は，本システム・コールを発行し，対象サブタスクのフレーム・カウンタ値が 0 以下になるように操作してください。

2.3.8　frm_set

〔機　　能〕

本 OS がスケジュールに使用しているフレーム・クロック・テーブルのフレーム・クロック情報を，入力パラメータで指定したフレーム・クロック情報に再設定します。

〔形　　式〕

・入力パラメータ

R0L＝変更するフレーム・カウンタのタスク ID

タスク ID は，OS_IDATA.ASM で定義するタスク・パラメータの登録順に，1 から昇順に割り当てられた値です。本パラメータのタスク ID は，_USED_EVENTASKHIGH の指定による影響を受けません。

R1L＝変更するフレーム・カウンタのサブタスク ID

サブタスク ID は，OS_IDATA.ASM で定義するフレーム情報の登録順に，タスクごとに 0 から昇順に割り当てられた値です。本パラメータのサブタスク ID は，_USED_EVENTASKLOW の指定による影響を受けません。

bit15＝0 のとき，R2L は X メモリ上のアドレスを示します。

bit15＝1 のとき，R2L は Y メモリ上のアドレスを示します。

R2L＝新規フレーム・クロックの情報が格納されているアドレス

指定されたアドレスからフレーム・クロック情報を取得し，指定サブタスクのフレーム・クロック情報を変更します。

指定アドレスにフレーム・ステータス，指定アドレス＋1 にフレーム・カウンタ値，指定アドレス＋2 にフレーム・カウンタ A，指定アドレス＋3 にフレーム・カウンタ B が設定されている必要があります。

・呼び出し方

```
R0L＝1                  ；タスク ID
R1L＝0                  ；サブタスク ID
R2L＝frame_info_type1   ；フレーム・クロック情報の格納されているアドレス
CALL frm_set            ；
```

・リターン値

TE_OK…正常終了

TE_BADID…タスク ID，サブタスク ID が指定可能範囲外

〔処　　理〕

本システム・コールは，サブタスク内，割り込みハンドラ内から呼び出し可能です。

作業用レジスタとして，R0，R1 を使用しますので，R0，R1 の内容を保持したい場合，本システム・コールを発行する前にセーブする必要があります。R2 の内容は破壊されません。

本システム・コールを発行した場合，入力パラメータ R0L,R1L に指定したサブタスクのフレーム・クロック情報を，入力パラメータ R2L が示すフレーム・クロック情報に変更します。

複数のサブタスクのフレーム・クロック情報を同じタイミングで変更したい場合，割り込み禁止中に本システム・コールを発行し，すべてのフレーム・クロック情報を変更してから，割り込み禁止を解除してください。

2.3.9　frm_get

〔機　　能〕

本 OS がスケジュールに使用しているフレーム・クロック・テーブルのフレーム・クロック情報の現在の値を，入力パラメータで指定したアドレスに保存します。

〔形　　式〕

・入力パラメータ

R0L＝変更するフレーム・カウンタのタスク ID

タスク ID は，OS_IDATA.ASM で定義するタスク・パラメータの登録順に，1 から昇順に割り当てられた値です。本パラメータのタスク ID は，_USED_EVENTASKHIGH の指定による影響を受けません。

R1L＝変更するフレーム・カウンタのサブタスク ID

サブタスク ID は，OS_IDATA.ASM で定義するフレーム情報の登録順に，タスクごとに 0 から昇順に割り当てられた値です。本パラメータのサブタスク ID は，_USED_EVENTASKLOW の指定による影響を受けません。

bit15＝0 のとき，R2L は X メモリ上のアドレスを示します。

bit15＝1 のとき，R2L は Y メモリ上のアドレスを示します。

R2L＝フレーム・クロック情報を保存するアドレス

OS_IDATA.ASM で定義するフレーム・クロック情報の形式に従って，次の 4 つの情報が指定アドレスに保存されます。

指定アドレスにフレーム・ステータス，指定アドレス＋1 にフレーム・カウンタ値，指定アドレス＋2 にフレーム・カウンタ A，指定アドレス＋3 にフレーム・カウンタ B が保存されます。

・呼び出し方

```
R0L＝1                  ；タスク ID
R1L＝0                  ；サブタスク ID
R2L＝frame_info_type1   ；フレーム・クロック情報の格納されているアドレス
CALL frm_get            ；
```

・リターン値

TE_OK…正常終了

TE_BADID…タスク ID，サブタスク ID が指定可能範囲外

〔処　　理〕

本システム・コールは，サブタスク内，割り込みハンドラ内から呼び出し可能です。

作業用レジスタとして，R0，R1 を使用しますので，R0，R1 の内容を保持したい場合，本システム・コールを発行する前にセーブする必要があります。R2 の内容は破壊されません。

本システム・コールを発行した場合，入力パラメータ R0L,R1L で指定したサブタスクのフレーム・クロック情報を，入力パラメータ R2L が示すフレーム・クロック情報に保存します。

複数のサブタスクのフレーム・クロック情報を同じタイミングで取得したい場合，割り込み禁止中に本システム・コールを発行し，必要なフレーム・クロック情報を取得してから，割り込み禁止を解除してください。

2.3.10　evt_set

〔機　　能〕

イベント ID スタックにイベント ID を設定します。

入力パラメータにより，イベント ID の設定のみを行うかあるいはイベント処理プログラムを実行するか選択できます。イベント ID の設定のみによって設定されたそのイベント ID は，システム・コール evt_exe で実行することができます。

〔形　　式〕

・入力パラメータ

R0L＝イベント・コマンド（次に示す 16 ビット・データ）

（1）メッセージ ID

ユーザは任意のメッセージ ID を指定できます。

本メッセージ ID は，システム・コール evt_exe で目的のメッセージ ID を受け取るためのものです。

実行 ID が 00 以外のとき，本メッセージ ID は無視されます。

（2）実行 ID

00： イベント ID の設定のみ行います。このイベントはシステム・コール evt_exe を使って実行します。_USED_EVENTASKHIGH が 1 のとき有効です。

01： 指定イベント ID のイベント処理プログラムをロウ・プライオリティ・イベント管理処理で実行します。_USED_EVENTASKLOW が 1 のとき有効です。

10： 指定イベント ID のイベント処理プログラムをハイ・プライオリティ・イベント管理処理で実行します。_USED_EVENTASKHIGH が 1 のとき有効です。

11： _USED_EVENTASKLOW が 1，_USED_EVENTASKHIGH が 0 のとき， 01 のコード指定とみなします。

_USED_EVENTASKLOW が 0，_USED_EVENTASKHIGH が 1 のとき，10 のコード指定とみなします。

_USED_EVENTASKLOW が 1，_USED_EVENTASKHIGH が 1 のとき，01 のコード指定とみなします。

（3）イベント ID

イベント ID は，OS_TASKX.ASM 内のイベント処理プログラム・テーブル（_MOS_EventSubTaskList）登録順に，0 から昇順に割り当てられます。

・呼び出し方

R0L＝0x0181;　イベント・コマンド

CALL evt_set;

・リターン値

TE_OK…正常終了

TE_EVTSTKOV…イベント ID スタック・オーバフロー

指定イベント・コマンドは無視されます。

TE_EVTSTKOV の判断は，リターン値以外にも_MOS_EvtStkOverFlow:X 変数で参照できます。

この変数の初期値は 0 ですが，一度，TE_EVTSTKOV が設定されるとユーザ自身が意図的にクリアするまで保持します。

TE_BADEVT…イベント・コマンドの指定に誤りがあります。

_USED_EVENTASKLOW が 0 であるにも関わらず，実行 ID が 01 で指定されています。

または，_USED_EVENTASKHIGH が 0 であるにも関わらず，実行 ID が 00 または 10 で指定されています。

〔処　　理〕

ロウ・プライオリティ・イベント・タスク管理（_USED_EVENTASKLOW），ハイ・プライオリティ・イベント・タスク管理（_USED_EVENTASKHIGH）のいずれかが指定されていれば，本システム・コールを使用できます。

本システム・コールは，サブタスク内，割り込みハンドラ内から呼び出し可能です。

2.3.11　evt_exe

〔機　　能〕

イベント処理プログラムを実行します。

入力パラメータにより，システム・コール evt_set によって設定されたイベント ID を実行するかあるいは本システム・コールのイベント ID を実行するか選択できます。

〔形　　式〕

・入力パラメータ

R0L＝イベント・コマンド（次に示す 16 ビット・データ）

（1）メッセージ ID

ユーザは任意のメッセージ ID を指定できます。

本メッセージ ID により，システム・コール evt_set で設定されたメッセージ ID のイベント・コマンドを受け取ることができます。

実行 ID が 0 のとき，本メッセージ ID は無視されます。

（2）実行 ID

0： 本システム・コールのイベント ID を実行します。
実行 ID が 0 のとき，メッセージ ID は無視されます。

1： システム・コール evt_set によって設定されたイベント ID スタック中の未処理イベントを実行します。対象は，メッセージ ID と同じメッセージ ID を持つイベント ID です。
同じメッセージ ID が複数存在する場合，時間的に古いものから順にすべて実行します。
同じメッセージ ID が存在しなかった場合，何も処理せずに戻ります。
実行 ID が 1 のとき，イベント ID は無視されます。

（3）予約

（4）イベント ID

イベント ID は，OS_TASKX.ASM 内のイベント処理プログラム・テーブル（_MOS_EventSubTaskList）登録順に，0 から昇順に割り当てられます。
実行 ID が 1 のとき，本イベント ID は無視されます。

・呼び出し方

```
R0L＝0x0001；　イベント・コマンド
CALL evt_exe；
```

・リターン値

TE_OK…正常終了

TE_EVTSTKOV…イベント ID スタック・オーバフロー
指定イベント・コマンドは無視されます。
TE_EVTSTKOV の判断は，リターン値以外にも_MOS_EvtStkOverFlow:X 変数で参照できます。
この変数の初期値は 0 ですが，一度，TE_EVTSTKOV が設定されるとユーザ自身が意図的にクリアするまで保持します。

TE_NOEXISTEVT…該当するメッセージ ID が存在しません。
何も処理せずに終了します。

〔処　　理〕

ハイ・プライオリティ・イベント・タスク管理（_USED_EVENTASKHIGH）が指定されている場合に，本システム・コールを使用できます。

本システム・コールは，サブタスク内からのみ呼び出し可能です。それ以外から呼び出した場合，その後の動作は保証されません。

サブタスク内から本システム・コールを発行した場合，システム・コール内でイベント処理プログラムを実行しますが，イベント処理プログラム内で本システム・コールを発行した場合，現在実行中のイベント処理プログラムが終了してから，そのイベント処理プログラムは実行されます。

2.3.12　mem_reserve

〔機　　能〕

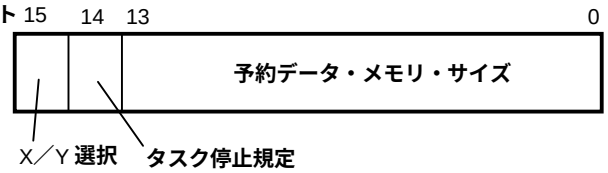
データ・メモリの予約を行います。

〔形　　式〕

・入力パラメータ

R0L＝ベース・アドレス

R1L＝予約のステータス



（1）X／Y 選択

0：X メモリ

1：Y メモリ

（2）タスク停止規定

0：TE_OVERLAPPED エラー発生時に，カレント・タスクを停止します。

1：TE_OVERLAPPED エラー発生時に，エラー・コード TE_OVERLAPPED を返します。

（3）予約データ・メモリ・サイズ

0x0001-0x3fff までの数値を指定できます。

・呼び出し方

```
R0L＝0                  ；ベース・アドレス
R1L＝0 x 4000 | 16      ；予約のステータス
CALL mem_reserve        ；
```

・リターン値（正常終了時）

R0L…ハンドル ID

・リターン値（エラー発生時）

TE_OVERLAPPED…指定メモリ空間はすでに予約されている

TE_RESERVEERR…メモリ予約エラー

エラー要因　・割り当て可能なハンドルが存在しない

・予約ステータスに不正な値が指定された

・タスク数が 1 のとき，予約ステータスのビット 14 が 0 に設定されている

・予約するデータ・メモリのサイズが 0 である

・ベース・アドレス＋予約するデータ・メモリのサイズが 0 x 10000 以上になる

〔処　　理〕

本システム・コールは，サブタスク内から呼び出し可能です。なお，レジスタ R1 の内容は破壊されません。

本システム・コールは、_MEM_RESERVE が TRUE に設定されている場合にサブタスク内，イベント処理プログラム内から使用可能です。

システム・コールの入力パラメータで指定したデータ・メモリ領域がすでに予約されていた場合，システム・コールの戻り値指定に関係なく TE_OVERLAPPED エラーを返します。システム・コールの戻り値指定が有効である場合，TE_OVERLAPPED エラーと同時に、_MOS_MemReserveStatus :X の bit 0 が 1 に設定されます。

TE_RESERVEERR エラーは、システム・コールの戻り値が有効な場合に返されます。このエラーの原因は，予約情報を記録する領域の不足または予約ステータス異常のどちらかです。、前者のエラーが発生した場合は_MOS_MemReserveStatus:X の bit 1 が 1 に設定されます。後者のエラーが発生した場合は_MOS_MemReserveStatus:X の bit 2 が 1 に設定されます。

_MOS_MemReserveStatus:X 変数の各ビットは，ユーザが 0 クリアするまでその情報を保持します。ただし，システム・コールの戻り値指定が無効の場合，_MOS_MemReserveStatus:X の bit 1, bit 2 を参照しないでください。

2.3.13　mem_free

〔機　　能〕

予約したデータ・メモリを開放します。

〔形　　式〕

・入力パラメータ

R0L＝mem_reserve システム・コールによって得られたハンドル ID

・呼び出し方

```
R0L＝*handleID:x　；ハンドル ID
CALL mem_free　　；
```

・リターン値

TE_OK…正常終了

TE_BADHANDLE…不正なハンドル ID が指定された

〔処　　理〕

本システム・コールは，サブタスク内で使用可能です。

本システム・コールは，_MEM_RESERVE が TRUE に設定されている場合にサブタスク内，イベント処理プログラム内から使用可能です。本システム・コールは，予約されたメモリを開放した際に，タスクの停止を解除可能かどうか確認し，可能であればタスクの停止を解除します。

システム・コールの戻り値が有効であるとき，入力パラメータ R0L に不正なハンドル ID を指定した場合，TE_BADHANDLE エラーとなり，_MOS_MemReserveStatus:X の bit 8 が 1 に設定されます。システム・コールの戻り値が無効であるときに，_MOS_MemReserveStatus:X の bit 8 を参照しないでください。

異なるタスクやサブタスクで予約した領域のハンドル ID を指定した場合はエラーとせず，予約の解除を行ないます。

［メ　　モ］

第 3 章　ユーザ記述ファイル

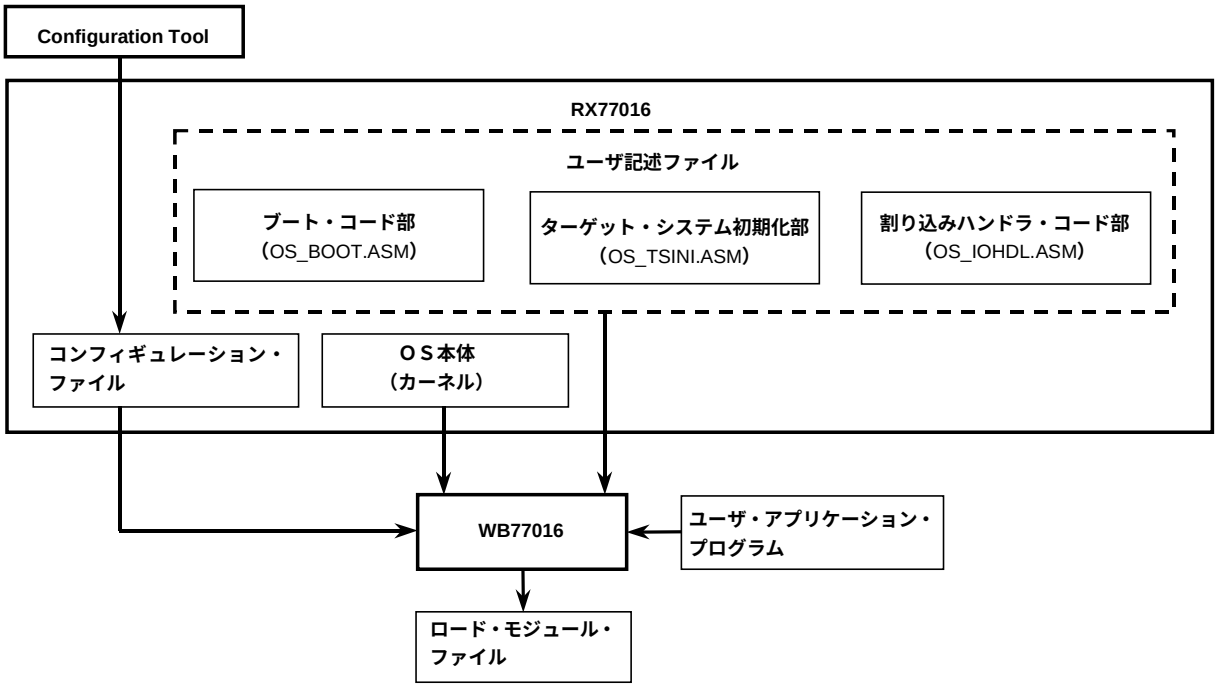
3.1　概　　要

本 OS には，ユーザが記述しなければならないファイルが存在します。これらは，ユーザ独自のコードと本 OS とのリンク作業がよりスムーズ進むように，あらかじめ本 OS から記述するべきファイルを提供するものです。それらのファイルをユーザ記述ファイルと呼びます。ここでは，ユーザ記述ファイルおよびユーザのためのコーディング・ルールについて説明します。

3.2　ユーザ記述ファイル一覧

図 3－1 に本 OS のファイル構成を示します。ユーザ記述ファイルには，OS_BOOT.ASM，OS_IOHDL.ASM，OS_TSINI.ASM の 3 個のファイルが存在します。各ユーザ記述ファイルの説明を 3.3 以降に示します。

図 3－1　RX77016 のファイル構成



3.3　OS_BOOT.ASM

OS_BOOT.ASM は，リブート・コードを記述するためのファイルです。

OS_BOOT.ASM のファイル形式とコーディング・ルールを次に示します。ユーザは，リブート・コードをこのファイルに記述しなければなりません。

3.3.1　ファイル形式

```
/* Include files */
#include <os_confg.h>

/* Public labels */
PUBLIC _MOS_ReBootProc

MEDIAOS_UIF_ReBoot         IMSEG at 0x240
_MOS_ReBootProc:
/*************************************/
/* You can write the reboot code here. */
/*************************************/
```

ユーザは，ここにリブート・コードを記述します。

```
         RET               ;
         END
```

3.3.2　コーディング・ルール

（1）OS_BOOT.ASM 内の_MOS_ReBootProc ルーチン内にリブート・コードを記述します。
　また，このルーチンからほかのファイルにあるリブート・コードを呼び出すことも可能です。

（2）_MOS_ReBootProc シンボル名は変更できません。

（3）最後は必ず RET 命令で終了しなくてはなりません。

（4）本ルーチンは，DSP 動作開始時に，0x200 番地から呼び出されます。

（5）本ルーチン内で割り込み禁止状態を解除することはできません。

（6）OS_UDEF.H 内の以下のパラメータ指定により，本ルーチンの呼び出しを解除できます。

　　#DEFINE _OSSIMULATE_DEBUG を TRUE（1）に設定する。

　　#DEFINE _USED_BOOTCODE を FALSE（0）に設定する。

　2 つのパラメータの内，いずれかの条件を満たしている場合に，本ルーチンの呼び出しを解除します。

3.4　OS_IOHDL.ASM

OS_IOHDL.ASM は，割り込みハンドラ・コードを記述するためのファイルです。

OS_IOHDL.ASM のファイル形式とコーディング・ルールを次に示します。ユーザは，割り込みハンドラ・コードをこのファイルを使って記述しなければなりません。

3.4.1　ファイル形式

(1/4)

```
/* Include files */
#include <os_cmn.h>
#include <os_scext.h>

/* Public labels */
#IF _USED_ivINT1 == ON
PUBLIC _MOS_ivINT1
#ENDIF /* _USED_ivINT1 */
#IF _USED_ivINT2 == ON
PUBLIC _MOS_ivINT2
#ENDIF /* _USED_ivINT2 */
#IF _USED_ivINT3 == ON
PUBLIC _MOS_ivINT3
#ENDIF /* _USED_ivINT3 */
#IF _USED_ivINT4 == ON
PUBLIC _MOS_ivINT4
#ENDIF /* _USED_ivINT4 */
#IF _USED_ivSI1 == ON
PUBLIC _MOS_ivSI1
#ENDIF /* _USED_ivSI1 */
#IF _USED_ivSO1 == ON
PUBLIC _MOS_ivSO1
#ENDIF /* _USED_ivSO1 */
#IF _USED_ivSI2 == ON
PUBLIC _MOS_ivSI2
#ENDIF /* _USED_ivSI2 */
#IF _USED_ivSO2 == ON
PUBLIC _MOS_ivSO2
#ENDIF /* _USED_ivSO2 */
#IF _USED_ivHI == ON
PUBLIC _MOS_ivHI
#ENDIF /* _USED_ivHI */
#IF _USED_ivHO == ON
PUBLIC _MOS_ivHO
#ENDIF /* _USED_ivHO */
/********************************* */
/* INTERRUPT VECTORS               */
/********************************* */
MEDIAOS_UIF_IOHandler      IMSEG    _IMEM_ALLOC_TYPE
#IF _USED_ivINT1 == ON
_MOS_ivINT1:                                        ; INT1
```

(2/4)

```
/*********************************************/
/* You can write the handler for INT1 here.     */
/*********************************************/
```

ユーザは，ここに INT1 **の割り込みハンドラ・コードを記述します。**

```
        JMP ret_int                                 ; System call function.
#ENDIF /* _USED_ivINT1 */

#IF _USED_ivINT2 == ON
_MOS_ivINT2:                                        ; INT2
/*********************************************/
/* You can write the handler for INT2 here.     */
/*********************************************/
```

ユーザは，ここに INT2 **の割り込みハンドラ・コードを記述します。**

```
        JMP ret_int                                 ; System call function.
#ENDIF /* _USED_ivINT2 */

#IF _USED_ivINT3 == ON
_MOS_ivINT3:                                        ; INT3
/*********************************************/
/* You can write the handler for INT3 here.     */
/*********************************************/
```

ユーザは，ここに INT3 **の割り込みハンドラ・コードを記述します。**

```
        JMP ret_int                                 ; System call function.
#ENDIF /* _USED_ivINT3 */

#IF _USED_ivINT4 == ON
_MOS_ivINT4:                                        ; INT4
/*********************************************/
/* You can write the handler for INT4 here.     */
/*********************************************/
```

ユーザは，ここに INT4 **の割り込みハンドラ・コードを記述します。**

```
        JMP ret_int                                 ; System call function.
#ENDIF /* _USED_ivINT4 */

#IF _USED_ivSI1 == ON
_MOS_ivSI1:                                         ; SI1/INT5
```

(3/4)

```
/*********************************************/
/* You can write the handler for SI1/INT5 here   */
/*********************************************/
```

ユーザは，ここに SI1 **の割り込みハンドラ・コードを記述します。**

```
        JMP ret_int                             ; System call function.
#ENDIF /* _USED_ivSI1 */

#IF _USED_ivSO1 == ON
_MOS_ivSO1:                                     ; SO1/INT6
/*********************************************/
/* You can write the handler for SO1/INT6 here. */
/*********************************************/
```

ユーザは，ここに SO1 **の割り込みハンドラ・コードを記述します。**

```
        JMP ret_int                             ; System call function.
#ENDIF /* _USED_ivSO1 */

#IF _USED_ivSI2 == ON
_MOS_ivSI2:                                     ; SI2/INT7
/*********************************************/
/* You can write the handler for SI2/INT7 here. */
/*********************************************/
```

ユーザは，ここに SI2 **の割り込みハンドラ・コードを記述します。**

```
        JMP ret_int                             ; System call function.
#ENDIF /* _USED_ivSI2 */

#IF _USED_ivSO2 == ON
_MOS_ivSO2:                                     ; SO2/INT8
/*********************************************/
/* You can write the handler for SO2/INT8 here. */
/*********************************************/
```

ユーザは，ここに SO2 **の割り込みハンドラ・コードを記述します。**

```
        JMP ret_int                             ; System call function.
#ENDIF /* _USED_ivSO2 */

#IF _USED_ivHI == ON
_MOS_ivHI:                                      ; HI/INT9
```

(4/4)

```
/*********************************************/
/* You can write the handler for HI/INT9 here.   */
/*********************************************/
```

ユーザは，ここに HI の割り込みハンドラ・コードを記述します。

```
        JMP ret_int                                ; System call function.
#ENDIF /* _USED_ivHI */

#IF _USED_ivHO == ON
_MOS_ivHO:                                         ; HO/INT10
/*********************************************/
/* You can write the handler for HO/INT10 here.  */
/*********************************************/
```

ユーザは，ここに HO の割り込みハンドラ・コードを記述します。

```
        JMP ret_int                                ; System call function.
#ENDIF /* _USED_ivHO */

END
```

3.4.2　コーディング・ルール

(1) OS_IOHDL.ASM 内の_MOS_ivXXX ルーチンに割り込みハンドラ・コードを記述します。
また,このルーチンからほかのファイルにある割り込みハンドラ・コードを呼び出すことも可能です。

(2) _MOS_ivXXX シンボル名は変更できません。

(3) 最後は必ず JMP ret_int 命令で終了しなくてはなりません。
ただし，タイマとして使用する割り込みハンドラの最後は，JMP man_tsk で終了します。

3.5　OS_TSINI.ASM

OS_TSINI.ASM は，初期化コードを記述するファイルです。
初期化コードは，OS のスタートアップ時に１回だけ実行されます。
OS_TSINI.ASM のファイル形式とコーディング・ルールを次に示します。ユーザは，この初期化コードをこのファイルを使って記述しなければなりません。

3.5.1　ファイル形式

```
/* Include files */
#include <os_confg.h>

/* Public labels */
PUBLIC _MOS_TargetSysInit

MEDIAOS_UIF_TargetSysInit  IMSEG    _IMEM_ALLOC_TYPE
_MOS_TargetSysInit:
/*       CLR(R4) already 0 clear in OS      */
/*       CLR(R5) already 0 clear in OS      */
/*       CLR(R6) already 0 clear in OS      */
/*       CLR(R7) already 0 clear in OS      */

/************************************************************ */
/* You can write the initialize code for the registers and the   */
/*  peripherals here.                                          */
/************************************************************ */
```

ユーザは，ここに各種レジスタおよびペリフェラル・レジスタなどの初期化コードを記述します。
下記のコードは，SST2, SDT2 **の初期化サンプル・コードです。適切なコードに変更してください。**

```
/************************************************************/
/* The following is sample codes.                          */
/************************************************************/

        R0L=0x0200              ; initialize port
        *SST2:X = R0L           ;
        CLR(R0)                 ; start output transfer
        *SDT2:X=R0L             ;

/************************************************************ */
/* You can write the initialize code for your variables here.    */
/************************************************************ */
```

ユーザは，ここに変数などの初期化コードを記述します。

```
        RET                     ;
        END
```

3.5.2　コーディング・ルール

(1) OS_TSINI.ASM **内の**_MOS_TargetSysInit **ルーチン内に初期化コードを記述します。**
また，このルーチンからほかのファイルにある初期化コードを呼び出すことも可能です。

(2) _MOS_TargetSysInit **シンボル名は変更できません。**

(3) 最後は必ず RET **命令で終了しなくてはなりません。**

(4) レジスタ R4-R7 **は，**0 **クリアされて本ルーチンが呼び出されますので，**0 **クリアの必要はありません。**

お問い合わせ先

【技術的なお問い合わせ先】

ＮＥＣ半導体テクニカルホットライン（インフォメーションセンター）
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−548−8899
FAX　：044−548−7900
E-mail　：s-info@saed.tmg.nec.co.jp

【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		〒108-8001　東京都港区芝５－７－１　（日本電気本社ビル）	(03)3454−1111
中部支社	半導体第一販売部 半導体第二販売部	〒460-8525　愛知県名古屋市中区錦１－17－１　（日本電気中部ビル）	(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	〒540-8551　大阪府大阪市中央区城見１－４－24　（日本電気関西ビル）	(06)6945−3178 (06)6945−3200 (06)6945−3208

北海道支社	札　幌	(011)251−5599	宇都宮支店	宇都宮	(028)621−2281	北陸支社	金　沢	(076)232−7303
東北支社	仙　台	(022)267−8740	小山支店	小　山	(0285)24−5011	京都支社	京　都	(075)344−7824
岩手支店	盛　岡	(019)651−4344	甲府支店	甲　府	(055)224−4141	神戸支社	神　戸	(078)333−3854
郡山支店	郡　山	(024)923−5511	長野支社	松　本	(0263)35−1662	中国支社	広　島	(082)242−5504
いわき支店	いわき	(0246)21−5511	静岡支社	静　岡	(054)254−4794	鳥取支店	鳥　取	(0857)27−5311
長岡支店	長　岡	(0258)36−2155	立川支社	立　川	(042)526−5981,6167	岡山支店	岡　山	(086)225−4455
水戸支店	水　戸	(029)226−1717	埼玉支社	大　宮	(048)649−1415	松山支店	松　山	(089)945−4149
土浦支店	土　浦	(0298)23−6161	千葉支社	千　葉	(043)238−8116	九州支社	福　岡	(092)261−2806
群馬支店	高　崎	(027)326−1255	神奈川支社	横　浜	(045)682−4524			
太田支店	太　田	(0276)46−4011	三重支店	津	(059)225−7341			

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］RX77016　ユーザーズ・マニュアル　機能編

（U14397JJ1V0UM00（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半導体テクニカルホットライン
FAX：（044）548-7900

キリトリ