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。

インフォメーション

RENESAS

保守／廃止

RXシリーズ Q&A集

RX116
RX136
RX320

資料番号　EEI-609A（第2版）
発行年月　May 1992　M

インフォメーション

NEC
電子デバイス

保守／廃止

RXシリーズ Q&A集

RX116
RX136
RX320

保守／廃止

PC DOS™は，米国IBM社の商標です。

コンカレントCP／M™は，米国ディジタル・リサーチ社の商標です。

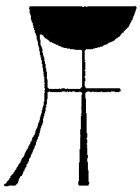
MS-DOS™は，米国マイクロソフト社の商標です。

UNIXオペレーティング・システムは，AT & Tが開発し，ライセンスしています。

V20™, V25™, V25＋™, V30™, V33™, V33A™, V35™, V35＋™, V40™, V40FD™, V50™, V50FD™, V53™は，日本電気株式会社の商標です。

Vシリーズ™は日本電気株式会社の商標です。

VAX™, VMS™, μVAX™, μVMS™, ULTRIX™は，米国ディジタル・イクイップメント社の商標です。



保守／廃止

○本書に記載されている内容は1992年2月現在の資料にもとづいたもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
○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。
○この製品を使用したことにより，第三者の工業所有権等にかかわる問題が発生した場合，当社製品の構造製法に直接かかわるもの以外につきましては，当社はその責を負いませんのでご了承ください。
○当社は，航空宇宙機器，海底中継器，原子力制御システム，生命維持のための医療用機器など極めて高い信頼性が要求される『特定』用途に推奨できる製品を標準的には用意しておりません。当社製品を『特定』用途にご使用をお考えのお客様，および，『標準』品質水準品を当社が意図した用途以外にご使用をお考えのお客様は，事前に販売窓口までご連絡頂きますようお願い致します。
　当社推奨の用途例
　　標準：コンピュータ，OA機器，通信機器，計測機器，工作機械，産業用ロボット，AV機器，家電等
　　特別：輸送機器（航空機，列車，自動車等），交通信号機器，防災／防犯装置等
○この製品は耐放射線設計をしておりません。

保守／廃止

本版で改訂された主な箇所

箇　所	内　　容	箇　所	内　　容	箇　所	内　　容
p. 2	Q1.2　追加	p. 35	Q4.2.5　追加	p. 61	Q4.3.14　追加
p. 2	Q1.3　追加	p. 39	Q4.2.9　追加	p. 63	Q4.4.4　追加
p. 3	Q2.1　追加	p. 40	Q4.2.11　追加	p. 65	Q4.4.7　追加
p. 4	Q2.4　追加	p. 41	Q4.2.12　追加	p. 66	Q4.4.8　追加
p. 5	Q2.5　追加	p. 41	Q4.2.13　追加	p. 67	Q4.4.9　追加
p. 6	Q2.6　追加	p. 45	Q4.2.20　追加	p. 69	Q4.5.3　追加
p. 6	Q2.7　追加	p. 45	Q4.2.21　追加	p. 70	Q4.5.6　追加
p. 9	Q2.12　追加	p. 46	Q4.2.22　追加	p. 71	Q4.5.7　追加
p. 10	Q2.15　追加	p. 48	Q4.2.26　追加	p. 71	Q4.5.8　追加
p. 11	Q2.16　追加	p. 48	Q4.2.27　追加	p. 72	Q4.5.9　追加
p. 13	Q2.18　追加	p. 49	Q4.2.28　追加	p. 72	Q4.5.10　追加
p. 14	Q2.19　追加	p. 49	Q4.2.29　追加	p. 73	Q4.5.11　追加
p. 14	Q2.20　追加	p. 50	Q4.2.30　追加	p. 73	Q4.5.12　追加
p. 18	Q2.24　追加	p. 50	Q4.2.31　追加	p. 74	Q4.6.2　追加
p. 18	Q2.25　追加	p. 51	Q4.3.32　追加	p. 75	Q4.6.3　追加
p. 22	Q2.31　追加	p. 52	Q4.3.2　追加	p. 75	Q4.6.4　追加
p. 25	Q3.4　追加	p. 53	Q4.3.4　追加	p. 76	Q4.6.5　追加
p. 26	Q3.5　追加	p. 54	Q4.3.5　追加	p. 76	Q4.6.6　追加
p. 26	Q3.6　追加	p. 55	Q4.3.7　追加	p. 77	Q4.7.1　追加
p. 27	Q3.7　追加	p. 56	Q4.3.8　追加	p. 78	Q4.8.1　追加
p. 31	Q4.1.2　追加	p. 58	Q4.3.11　追加	p. 80	Q5.3　追加
p. 31	Q4.1.3　追加	p. 59	Q4.3.12　追加	p. 80	Q5.4　追加
p. 32	Q4.1.4　追加	p. 61	Q4.3.13　追加	p. 81	第6章　追加

本文欄外の★印は，本版で改訂された主な箇所を示しています。

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

保守／廃止

はじめに

対象者 このインフォメーションは，RX116，RX136，RX320リアルタイム・オペレーティング・システムの使用を検討されている，またはRXシリーズを使用してアプリケーション・プログラムの開発をされているユーザを対象としています。

目　的 このインフォメーションは，ユーザの方からご質問をいただいた内容をまとめたものです。製品の使用をご検討中，またはプログラム開発中に不明な点が出た場合にご参照いただくことを目的としています。

読み方 不明な内容を目次より索引して参照ください。
このインフォメーションを使用する場合，必ず最新のユーザーズ・マニュアル，アプリケーション・ノートをあわせてご参照ください。

関連資料 ●RX116（Ver 2.×）に関する資料

資　料　名	資料番号
ユーザーズ・マニュアル　基礎編	IEM-5063
ユーザーズ・マニュアル　テクニカル編	EEM-719
ユーザーズ・マニュアル ニュークリアス・インストレーション編	EEU-747
アプリケーション・ノート	EEA-602

●RX136に関する資料

資　料　名	資料番号
ユーザーズ・マニュアル　基礎編	EEU-707
ユーザーズ・マニュアル　テクニカル編	EEU-719
ユーザーズ・マニュアル ニュークリアス・インストレーション編	EEU-775

●RX320に関する資料

資　料　名	資料番号
ユーザーズ・マニュアル　基礎編	EEU-717
ユーザーズ・マニュアル　テクニカル編	EEU-735
ユーザーズ・マニュアル ニュークリアス・インストレーション編	EEU-780

保守／廃止

保守／廃止

目次要約

保守／廃止

保守／廃止

目　次

保守／廃止

保守／廃止

第 1 章　契約関係

Q1.1

現在，RX116を使用していて，新たにV33™またはV53™を使用するとき，RX116を使っていい？

A1.1

量産契約は各RXが対象とするチップ群限定になっていますから，RX116をそのまま使用することはできません。再契約をしてください。

各RXが対象とするチップは，次のとおりです。

RX	対象チップ
RX116	V20™, V30™, V40™, V40FD™, V50™, V50FD™
RX136	V33, V33A™, V53
RX320	V25™, V25＋™, V35™, V35＋™

Q1.2 ★

リアル・タイムのOSの使用について契約違反になる？

(1) 開発中に他社のディバッガ上で使用できる？

(2) 開発したプログラム（OSとアプリケーション）をEPROMに書き込んで，試作品に組み込める？ 試作品は何台組み立ててもよい？

(3) 試作品を展示会に出展してもよい？

(4) 開発研究所内の他グループが他試作品にOSを利用するのはよい？

(5) OSを組み込んだ製品を量産できる？ CPUの種類には関係なく量産できる？ OSを外付けROMに書き込んだものも量産できる？

A1.2

RX320の場合

(1) ディバッガは制限しません。

(2) 試作品は許諾複製数分は作成することができます。

(3) 問題ありません。

(4) (2)の範囲内であれば問題ありません。

(5) ・量産分の契約をしてください。

・CPUはV25，V35に限定されます。

・外部ROMに組み込むこともできます。

HS25の場合（ディバグ用契約の場合）

(1) 契約書上で1つの開発装置に限定します。

(2) EPROMに組み込むことはできません（μPD79011，79021を使用してください）。

(3) 問題ありません。μPD79011，79021を使用してください。

(4) 3部以内であれば問題ありません。

(5) ・μPD79011，79021を使用しないで，V25／V35にHS25を組み込む場合は，量産契約をしてください。

・CPUはV25，V35に限定されます。

・外部ROMに組み込むことはできません（すべてマスク発注になります）。

Q1.3 ★

RX116を購入する際，オブジェクト契約でも，インタフェース・ライブラリ，スタート・アップ・ルーチンはソース供給してもらえる？

A1.3

オブジェクト契約でも提供します。

第２章　システム構築法

Q2.1★

OSを搭載するにあたって，ハードウエアとして以下の内容で必要な条件は？

(1)　OS関連で必要なメモリ容量

(2)　必要な周辺I/O

A2.1

(1)　ROM容量　約10.5Kバイト

RAM容量　最低約2Kバイト

RAMサイズについては，RX116テクニカル編に計算方法を掲載しておりますのでご参照ください。

(2)　必要な周辺I/OはμPD71054および相当品，μPD71059および相当品です。

Q2.2

コンフィギュレータにより，タスク情報（システム情報テーブル）を登録するとき，レーベル名の付け方に条件はある？

例　1タスク目を登録する場合のレーベル名の付け方

TASK_ID0

START_P0←最後尾に0が付く

2タスク目を登録する場合のレーベル名の付け方

TASK_ID1

START_P1←最後尾に1が付く

⋮

A2.2

レーベル名の付け方に，条件はありません。

Q2.3

コンフィギュレータで，クロック周波数に1 MHzよりも小さな値を指定するにはどうすればよい？

A2.3

コンフィギュレーション時は，1 MHzより大きなクロック周波数の値を指定して，コンフィギュレーション・ファイルを作成してください。その後，以下の箇所をエディタで修正してください。

```
clock_tcu   dw   ××××
            dw   ××××
            dw   ○○○○
                 ↑
```

この値をカウンタ値に設定する。

たとえば，10 msecの割り込みで，0.5 MHzならば1 388 Hにする。

Q2.4★

コンフィギュレーションについて

(1) コンフィギュレーション実行中システム・クロックの入力レートで100 msごとにクロック割り込みを発生させたい場合100 msでは入力エラーとなる。なぜ？

(2) タスク情報入力

Task_stack_size 100 ↵

と入力するとエラーとなる。なぜ？

(3) (1), (2)がエラーとなるのはCPUがそれを保証していないから？ それとも，ただ出力されたコンフィギュレーション・リストを書き直せばよい？

A2.4

(1) コンフィギュレータの制限で，21 msec以上の値は使用できません。

(2) この情報以前に入力したdefault stack sizeの値と比較して，これより小さいとエラーにします。

(3) (1)の問題については，コンフィギュレータを直接修正してください。関係するのは，intervalとclock_tcuのカウント値です。

Q2.5★

RX116を用いて①のようなシステムを設計し，②のようにコンフィギュレーション・テーブルを作成したところ動作しなかった。そこで，SYS_ram領域を200Hセグメント～1000Hセグメントに変更したところ動作した。最初に動作しなかった理由は？

① ・タスク数　4（アイドル初期含む）
・メールボックス　1
・メモリ・プール　1
・メモリ・ブロック　1
・NDPなし，タイム・アウト指定なし

② ・SYS_ram　500Hセグメント～7FFHセグメント
・SYS_tsk　40Hセグメント
・タスクのスタック・サイズ　1000H

A2.5

タスクのスタック・サイズは，各タスクごとにとられます。タスク数が4だけで，

1000H×4＝4000Hバイト

とられます。一方SYS_ramは，500H～800Hに指定していますので，

(800H－500H)×10H＝3000Hバイト

これが原因と考えられます。

備考　スタック・サイズが1000Hというのは，非常に大きな値で，誤解があると思われます。

RX116のコンフィギュレーションを変更するより，こちらの値を変更する方が良策と思われます。

Q2.6 ★

RX116のコンフィギュレータで，TCUに供給されているソース・クロックとは，外部クリスタルの周波数か内部クロックの周波数か，TCKSの値で分周された周波数か？

A2.6

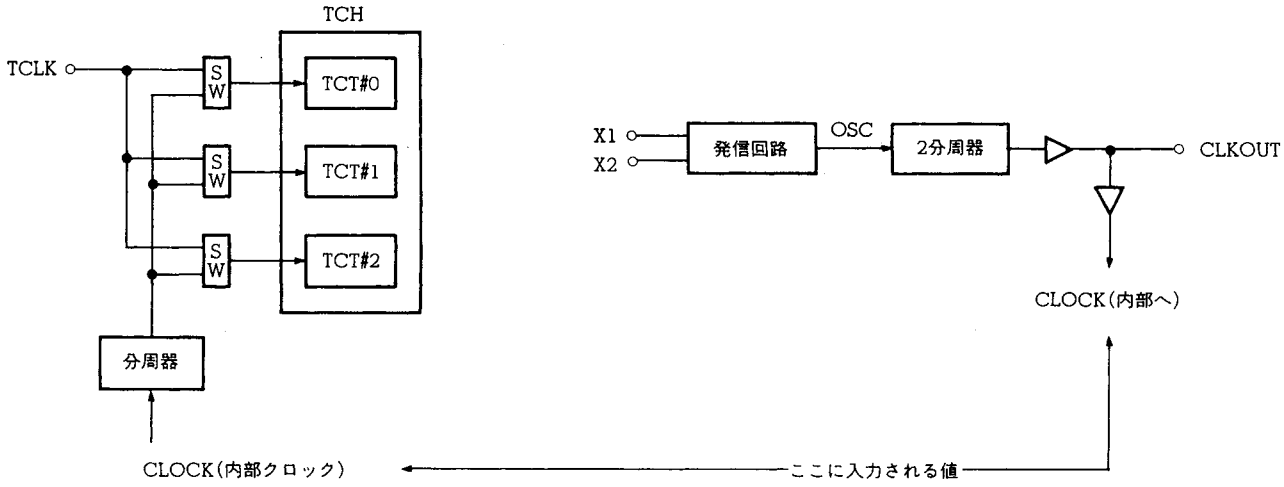
内部クロックを指定してください（下図参照）。

例　システム・クロックを1 msecにした場合

コンフィギュレーション時に指定する値により，タイマ・セットされる値は次のようになります。

16 MHz　→　1F40h(8000)

8 MHz　→　0FA0h(4000)



Q2.7 ★

コンフィギュレータで，default_stack_sizeで指定する値は10進数か16進数か？

A2.7

コンフィギュレータ中の指定はすべて16進数です。

Q2.8

リセット・ルーチンのエントリ・アドレスは，FFFF0Hでよい？　別な位置にロードしてFFFF0Hからジャンプしても問題はない？

A2.8

リセット・ルーチンはFFFF0Hとは別のアドレスに配置して，FFFF0Hからジャンプしてください。

Q2.9

バイアス値を与えてニュークリアスをロードするとき，ベクタのセグメント値，オフセット値は変更しなければいけない？

A2.9

RXシリーズは１つのセグメント内に配置されているので，オフセット値の変更は必要ありません。

セグメント値の変更は，次の方法でRXシリーズ自身が設定します。

① リセット・ルーチン終了後，コンフィギュレーション情報のニュークリアス配置アドレスをみて，ニュークリアスへジャンプ。

② 現在のプログラム・セグメントを割り込みベクタにセット。

Q2.10

RXシリーズのマニュアルには，「ニュークリアスは，アドレス情報として，0000:0を持っています。」とあるが，これはROM化するときに0000:0番地に置かれるということ？

A2.10

RXシリーズが持っているディフォールトのアドレス値が0000:0ということです。

RXシリーズのニュークリアスは，どのアドレスにも置くことができます。したがって，この設定を変えるには，次の操作を行ってください。

① リンク・パラメータ・ファイルを変更し，再リンクする。

② IEへのロード時に配置アドレスを変更する（IEの機能を使用して，バイアス・ロードを行う）。

③ 拡張HEX形式のファイル中の配置アドレス情報を変更する。

Q2.11

RX内の使用しない機能について

(1) メールボックスやセマフォの機能を使用しないで構築できる？

(2) 使用しないシステム・コールをRX内から取り外すことができる？

A2.11

(1) 構築できます。

一般に制御系のシステムではメールボックス，セマフォ等タスク間での同期通信機能を使用しますが，使用しなくても結構です。

(2) できます。

コンフィギュレータを使用して，各システム・コールごと使用するかしないかを指定し，ブランチ・テーブル・ファイルを作成してください。

Q2.12 ★

RX116，136を使用したときのシステム・テーブルについて，「システム・テーブルは，コンフィギュレータ時に指定されたフリーRAMエリアの最も低いアドレスに生成されます。」とあるが，たとえば，コンフィギュレータでアドレス0000:0000Hからフリー-RAMを指定した場合，V30系のCPUは，セグメント0000Hのオフセット000～3FFFHのアドレスは割り込みベクタ・テーブルとして固定割り付けされているため，システムテーブルはそのアドレスに生成できないはずである。したがってフリーRAMはアドレス0000:0000Hから0000:03FFHは指定できず，RX116，136での最も低いアドレスとは0000:0400Hであると解釈して，コンフィギュレータで指定しなければならない？

A2.12

フリーRAMは割り込みベクタと重ならないアドレスに指定してください。

Q2.13

コンフィギュレーション時に指定する“タスクの初期情報データ”とは？

A2.13

“タスクへの初期情報データ”とは，タスクに渡される初期情報パラメータのことです。

これは，sta_tsk（a_tsk，initcode）システム・コールにおけるinitcodeと同じ内容であり，起動元タスクと起動されるタスクとの間の初期データ引き渡しに有効です。

initcodeの内容自体は，ユーザが意識して使用する1ワードのデータです。

Q2.14

コンフィギュレーション時に最大タスク数128個以上を生成することは可能？

A2.14

コンフィギュレーション時としては，不可能です。

したがって，タスク処理部にてcre_tskシステム・コールを発行して，タスクを生成してください。

Q2.15 ★

RX116のコンフィギュレーションにおいて，ターゲットのH／Wが決められているが，そのほかの周辺I/Oを使用する場合は動作しない（たとえば，71054×2，71059×1，72001×3などの場合）？

A2.15

RX116は，割り込みコントローラとしてμPD71059(スレーブ接続対応)に対応し，タイマとしてμPD71054を1ch使用します。そのほかの周辺デバイスはユーザのアプリケーションで管理してください。

μPD72001には割り込みコントローラと同等の機能がありますが，その機能は使用できません。μPD72001を使用する場合は，シリアル・コントローラとしてのみ使用できます。

Q2.16 ★

RX116のコンフィギュレーション・ファイルで，V20～V50に関して，SCU，NDPを使用しているサンプルはある？

A2.16

cf70116により作成したコンフィギュレーション・ファイル例（NDP，SCU使用）を次に示します。

入力データ

キャラクタ・バッファ・ポート・アドレス	00E2	パリティ	NO
コマンド・ワード・ポート・アドレス	00E0	TCUカウンタ	#1
ボー・レート	9600	TCUアドレス	00D2
キャラクタ長	8 bit	TCU入力クロック	3.072
ストップ・ビット	1	割り込みレベル	INTP_1

```
;Real time operating system for V-serise

;/*****************************************************************/
;/*                                                               */
;/*                                                               */
;/*              system information table for RX116                */
;/*                                                               */
;/*              Copyright (c) NEC corporation 1990               */
;/*                                                               */
;/*                                                               */
;/*****************************************************************/

sgroup          group           sys_conf
sys_conf        segment         public   'code'
                assume          ds0:sgroup
                public          sys_information

;/*****************************************************/
;/*              system information table             */
;/*****************************************************/

sys_information         label           far
kernel_p                dw              0000h
                        dw              0F000h

sys_info_p              dd              sys_sys
ram_info_p              dd              sys_ram
cpu_info_p              dd              0
ndp_info_p              dd              sys_ndp
icu_info_p              dd              sys_icu
tcu_info_p              dd              sys_tcu
scu_info_p              dd              sys_scu
tsk_info_p              dd              sys_tsk

sys_ndp                 label           far
                        dw              18h

                        even

sys_icu                 label           far
master                  dw              00C0h
                        dw              00C2h
                        db              13h
                        db              38h
                        db              00h
                        db              01h
slave_0                 dw              0FFFFh
                        dw              0FFFFh
                        db              11h
                        db              00h
                        db              00h
                        db              01h
slave_1                 dw              0FFFFh
                        dw              0FFFFh
                        db              11h
```

```
                db          00h
                db          01h
                db          01h
slave_2         dw          0FFFFh
                dw          0FFFFh
                db          11h
                db          00h
                db          02h
                db          01h
slave_3         dw          0FFFFh
                dw          0FFFFh
                db          11h
                db          00h
                db          03h
                db          01h
slave_4         dw          0FFFFh
                dw          0FFFFh
                db          11h
                db          00h
                db          04h
                db          01h
slave_5         dw          0FFFFh
                dw          0FFFFh
                db          11h
                db          00h
                db          05h
                db          01h
slave_6         dw          0FFFFh
                dw          0FFFFh
                db          11h
                db          00h
                db          06h
                db          01h
slave_7         dw          0FFFFh
                dw          0FFFFh
                db          11h
                db          00h
                db          07h
                db          01h

                even
sys_tcu         label       far
interval        dw          20
clock_tcu       dw          00D6h
                dw          00D0h
                dw          0F000h
                db          34h
                db          08h
baud_tcu        dw          00D6h
                dw          00D2h
                dw          0014h
                db          76h
                db          0

sys_scu         label       far
                dw          00E2h
                dw          00E0h
                db          4Eh

                even
sys_sys         label       far
exception       db          1
                db          1
stack_size      dw          0100h

                even
sys_ram         label       far
area_count      dw          1
area_1          dw          0083h
                dw          3000h

                even
sys_tsk         label       far
task_count      dw          1
task_1          dw          0064h
                dw          0000h
                dw          0E000h
                dw          3000h
                dw          0100h
                dw          01h
                dw          0000h

sys_conf        ends
                end
```

Q2.17

RX116は，Intel社の8086上でも動作する？

A2.17

RX116は，Vシリーズのオリジナル命令を用いて高速化を図っていますので，8086上では動作しません。

Q2.18★

コンフィギュレータで使用しないシステム・コールを取り除いた場合，ニュークリアスのサイズは縮小する？

A2.18

縮小します。コード・サイズを次の表に示します。

システム・コール	コード・サイズ（バイト）
cre_tsk	343
sta_tsk	80
del_tsk	64
ext_tsk	16
exd_tsk	48
def_ext	32
abo_tsk	105
ter_tsk	448
tcb_adr	48
tsk_sts	48
rot_rdq	64
chg_pri	375
sus_tsk	96
rsm_tsk	96
wai_tsk	80
wup_tsk	128
can_wup	51
slp_tsk	48
cyc_wup	173
can_cyc	64
cre_flg	115
del_flg	160
set_flg	195
wai_flg	218
flg_adr	32
cre_sem	131

システム・コール	コード・サイズ（バイト）
del_sem	182
sig_sem	160
wai_sem	231
sem_adr	32
cre_mbx	131
del_mbx	208
sen_mes	208
rcv_mes	205
mbx_adr	32
def_exc	144
ret_exc	134
def_int	160
ret_int	99
dis_int	112
ena_int	112
fet_dat	109
get_dvn	144
cre_mpl	160
del_mpl	176
get_blk	234
rel_blk	96
mpl_adr	32
set_tim	32
get_tim	16
iret_wup	192
def_svc	48
get_ver	64

Q2.19 ★

NECのOMFからIntel OMF86へ変換した場合，Intel OMF86はCPU8086に対応しているのでNECのOMFからIntel OMFに変換するとV33固有の命令が出力されないのでは？

A2.19

OMFの中には，大きく分類すると次の4つのデータがあります。

- ヘッダ部　　：OMFの属性に関する情報
- ロウ・データ部　　：実際のプログラムやデータ部分が格納されている
- リロケーション・データ部：リンクするための情報
- シンボル情報　　：ディバグのための情報

命令コードもデータもロウ・データの部分に16進数の並びとして格納されるだけですので，V33固有の命令も変換されます。

Q2.20 ★

RX136のニュークリアス部分はIntel HEX形式で提供されているが，エミュレータではIntel OMF86形式のみの入力しかできない。また，Intel OMF86中にディバグ情報が必要である。

したがって，Intel HEXからIntel OMF86への変換プログラムを作成しようと考えているが，ニュークリアス部分のIntel HEXにディバグ情報は入っている？

A2.20

Intel HEXにはディバグ情報は，入っていません。

Intel OMF86へのコンバータを作るのであれば，次の2つの方法が考えられます。

- Intel OMF86中のディバグ情報は，ダミー値で埋める（RX136中のシンボルは不要と思われます）。
- NECのOMF（RX136.LNK，この中にはディバグ情報が入っている）からIntel OMFへのコンバータを作る（NECのOMF形式はEEI-601Aをご参照ください）。

Q2.21

コンフィギュレータは，どの環境で動作する？

A2.21

コンフィギュレータは，次の環境で動作します。

・RX116の場合

PC-9800シリーズ	(MS-DOS™)
MD-086／116	(コンカレントCP／M™)
VAX™	(UNIX，VMS™)
μVAX™	(ULTRIX™，μVMS™)
IBM PCシリーズ	(PC DOS™)

・RX136，RX320の場合

PC-9800シリーズ	(MS-DOS)
VAX	(UNIX, VMS)
μVAX	(ULTRIX，μVMS)
IBM PCシリーズ	(PC DOS)

Q2.22

エディタを使用して，コンフィギュレータではなくマニュアルでコンフィギュレーション・テーブルを作成しても問題はない？

A2.22

まったく問題はありません。

Q2.23

割り込みコントローラ（ICU）について

(1) μPD71059をレベル・モードに設定して，使用できる？

(2) μPD71059のあるレベルの割り込み要求が発生したとき，その要求に対するCPUからの$\overline{\text{INTAK}}$パルスが返る前に割り込み入力（アクティブ・ハイ）が"L"に変化してしまった場合，ユーザ側はどのような処理を行う？

(3) μPD71059をエッジ・トリガ・モードで使用すると，同時に2つ以上のINT入力がある場合，優先順位の低いINT入力は無視されて，とりこぼすことがあるのでは？

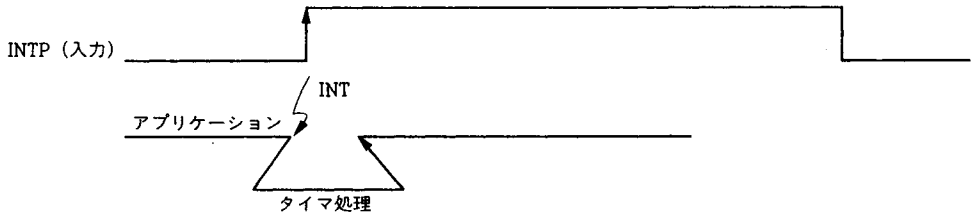
A2.23

(1) 使用できません。エッジ・トリガ・モードでのみ使用してください。

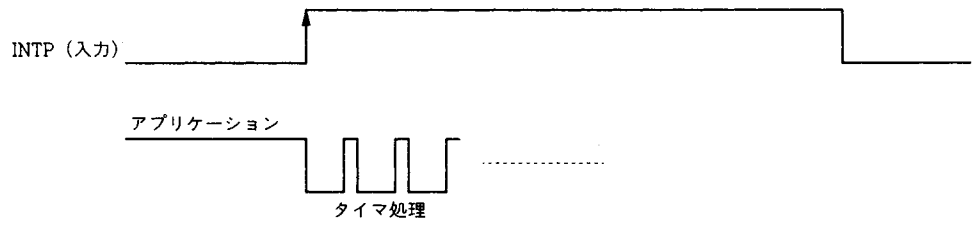
RXでは，割り込みを受け付けたのち，μPD71059に対してはISR（イン・サービス・レジスタ）をクリアする処理のみを行います。

RXのシステム・クロックは，TCU（タイマ）をモード2で使用しますので，レベル・モードで使用した場合，タイマ処理終了後，再度，タイマ処理に分岐します（下図のタイミング参照）。

・エッジ・トリガ・モードの場合



・レベル・モードの場合

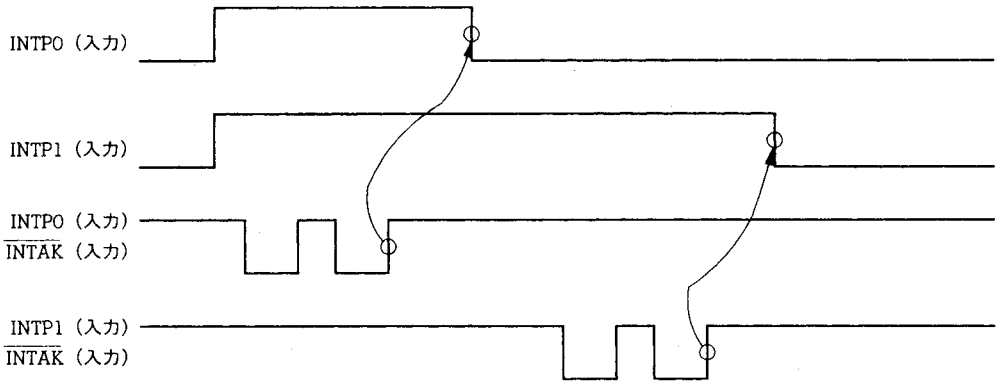


(2)　割り込みコントローラ側では，このような割り込みを不完全割り込みとして扱い，割り込みコントローラのレベル7（最低優先割り込み）の処理を実行します。

したがって，ユーザ側はこのような割り込みが発生する可能性があるときは，不完全割り込み要求に対処できる処理をレベル7に設定しておく必要があります。

ただし，RXではユーザが設定を行わない（使用しない）割り込みベクタには，OSで定義している割り込み処理ルーチン（NOP命令とRETI命令を発行）のアドレスを設定しますので，ユーザ側がレベル7の処理を特別に設定しなくてもRX側で対処可能です。

(3)　エッジ・トリガ・モードでも，$\overline{\text{INTAK}}$信号が返るまでINT信号を“H”に保つので，とりこぼすことはありません（下図のタイミング参照）。



Q2.24 ★

RX136の動作で，エッジ・トリガでの割り込み処理では限界があるため，レベル・トリガをベースにして動作させたい。その際

(1) レベル・トリガで動作する？

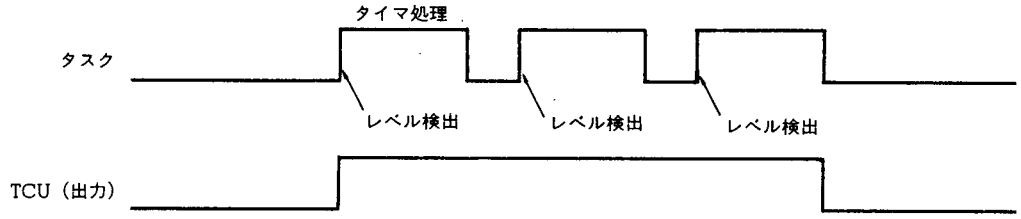
(2) レベル・トリガにしたときカーネルの変更点は？ 問題点は？

A2.24

(1) 社内評価，検査は，すべてエッジ・トリガで行っています。レベル・トリガでの実績はありません。

(2) 通常のままでは，タイマ処理が問題です。

RX136では，μPD71054をモード2に設定してタイマ処理を行うのが一般的ですが，レベル・トリガで使用すると何度もタイマ処理を行ってしまいます（下図参照）。レベル・トリガで使用するには，μPD71054をモード0に設定し，タイマ処理ごとにカウント数を書き込む処理をしてください。



Q2.25 ★

RX116のconfig tb1で，TCUの割り込みレベルとは何？

使用方法は？ ボー・レート用の割り込みレベル予約0は何を意味している？

A2.25

ICUに接続されている割り込みレベルです。

Q2.26

μPD71054の出力をタイマ割り込みとして使用するとき，

(1)　μPD71054の持つすべてのモードを使用できる？

(2)　μPD71054のどのモードを使用するのに最適？

A2.26

(1)　使用することはできますが，すべてのモードで期待正常値（クロック入力の幅）を出力できるとはいえません。

その理由としては，RXはμPD71054に対して，ダウン・カウント値を書き込むという作業をシステム初期化時に1度しか行いません。したがって，1ショット・モードでは，ユーザ側で再度ダウン・カウント値を書き込む必要があり，ここで多少の誤差が生じることになります。

(2)　最適なモードは，モード2（レート・ジェネレータ）です。

モード3（方形波ジェネレータ）も使用可能ですが，モード2の場合は，OUT端子出力が1クロック・サイクルだけしか0になりません。したがって，モード2の出力を割り込み要求として使用する場合，モード3に比べて割り込み要求を長く保つことができます。

V40，V50のTCUの出力をICUに入力し，タイマ割り込みをかける場合には，特にモード2を使用してください。

Q2.27

RXで，MPSCコントローラμPD7201AをINTP5にスレーブ接続できる？

A2.27

RXでは，スレーブのコントローラに対してret_int，iret_wupの各システム・コール中でμPD7201AのEOIコマンドはサポートしていないため，μPD7201Aをスレーブ接続できません。

RXでは，μPD7201AをICUのスレーブとして使用せず，周辺コントローラの1つとして使用してください。

μPD7201Aが何の要因でINTを出したかは，μPD7201AのSR2Bレジスタを読むことで判断できます。

Q2.28

MPSCコントローラμPD72001をμPD71059のスレーブとして接続し，オートベクタ・モード，多重割り込みを使用することはできる？

A2.28

RXシリーズでは，μPD72001を割り込みコントローラとして使用できません。

μPD72001は，通信機能に加えて割り込みコントローラ的な機能を持っていますが，RXシリーズでは，割り込みコントローラとしてμPD71059相当品のシステム上で動作します。そのため，他の割り込みコントローラには対応していません。

μPD72001をRXで使用するときは，周辺コントローラの1つとして使用してください。

Q2.29

CPUにV20を使用したシステムを，V30を使用したシステムに移植する場合，RXはμPD71054，μPD71059に対して，

(1)　ワード・アクセスを行う？

(2)　奇数アドレスに対して，バイト・アクセスを行う必要がある？

A2.29

(1)　ワード・アクセスは行いません。

RX内でのアクセス方法は，次のとおりです。①はポート・アドレス，②はデータです。

・OUT $\underset{①}{DW}$, $\underset{②}{AL}$

・IN $\underset{②}{AL}$, $\underset{①}{DW}$

(2)　RX内では，μPD71054およびμPD71059に対して，(1) で示したようにアクセスしています。

したがって，ポート・アドレスが奇数となるような設定（ハード，ソフトともに）を行えば，RXが奇数アドレスをアクセスする場合もあります。

Q2.30

フリーRAMの指定範囲の上限／下限アドレスはどこまで？

A2.30

上限アドレスは，OSの仕様としては特に制限はありませんが，通常メモリ空間の上位にはROMを配置しますので，ROMアドレスの下限が実質の上限とお考えください。

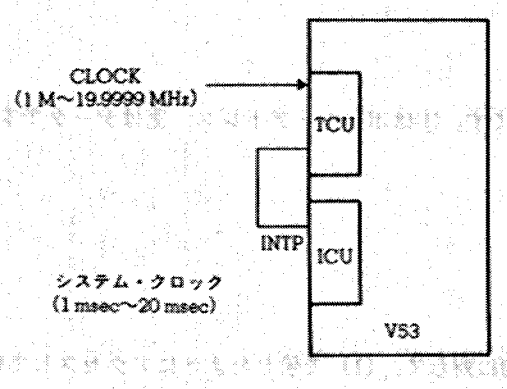
下限アドレスは，400 H注です。これ以下のアドレスはベクタ・アドレスとして使用します。

注　コンフィギュレーション時のLOW_DSは40 Hとなります。

Q2.31★

V53を使ったターゲット・システム上でRX136リアルタイムOSを動作させようとしたとき，RX136のシステム・クロックは下図のように作成するものだと解釈している。コンフィギュレータを使ってシステム情報テーブル（SIT）を作成する場合，システム・クロックの制限は1 ms～20 msの範囲で1 msec毎の設定となっている。

もし，エディタでSITを変更してシステム・クロックを0.1 msecとか1 msecより短くした場合どんな問題がでてくる？



A2.31

・RX136は，システム・クロック周期（min)／msecで動作保証します。

・OS内部では一定周期ごとに特定の処理を行いますが，この処理が17～50 μsec（条件によってはこれより大きくなります）かかります。システム・クロック周期を小さくすると，OSの動作時間（オーバヘッド）が大きくなり，アプリケーションが動く時間が小さくなります。

第３章　開発ツール

Q3.1

ソフトウエアについて

(1)　RXでサポートする高級言語は，C言語のみ？

(2)　他社のCコンパイラを使用して開発できる？

A3.1

(1)　サポートしているのはC言語（NEC製CコンパイラCC70116またはCC70136）のみですが，使用する高級言語の仕様がOSとのインタフェースをインタフェース・ライブラリ内で吸収できるものであれば，インタフェース・ライブラリを書き直すことによりC言語以外の高級言語でも使用できます。

インタフェース・ライブラリの内容および作成方法はRXシリーズのユーザーズ・マニュアルのテクニカル編に記述してあります。

(2)　できます。

ただし，次の3点について，製品に添付されるインタフェース・ライブラリのソース・プログラムの修正が必要です。

①関数コール時のパラメータのスタックへの積み方，戻り値を入れるレジスタがCC70×××と異なるときは，修正が必要です。

②86系のツール（MSC等）を使う場合，対応するアセンブラ（MASM等）に合わせて，ニモニックの変更が必要です。

③セグメント名，関数名も，使用するCコンパイラに合わせて修正が必要です。

Q3.2

RXの開発環境について，アプリケーション・プログラムの開発マシンはパソコン・ベースでNECが保証する環境以外は使用できない？

A3.2

パソコン・ベースでNECが保証する環境は，次のとおりです。

①PC-9800シリーズ（MS-DOS）

②IBM PC　　（PC DOS）

①と②以外のマシンでも互換性があるなら動作すると思われますが，保証はできません。

Q3.3

RXの使用できるCコンパイラのメモリ・モデルは，ラージ・モデル（コード・サイズ，データ領域ともに1 Mバイト参照可）だけをサポートする？

A3.3

ラージ・モデルのみサポートしています。

Q3.4★

カーネル，スタート・アップ・ルーチン，ユーザ・アプリケーション相互のインタフェースは？

A3.4

RX116を起動するためには，下図に示す②のスタート・アップ・ルーチンが動作します。②は，コンフィギュレーション情報のカーネル・ロケーション・アドレスの値を参照してのカーネルにジャンプ（far br）します(このとき，コンフィギュレーション情報のアドレスをDS1:1Yにセットしています)。カーネルは初期化などを行ったあと，コンフィギュレーション情報のイニシャル・タスクに制御を移します（スタック操作とreti命令)。

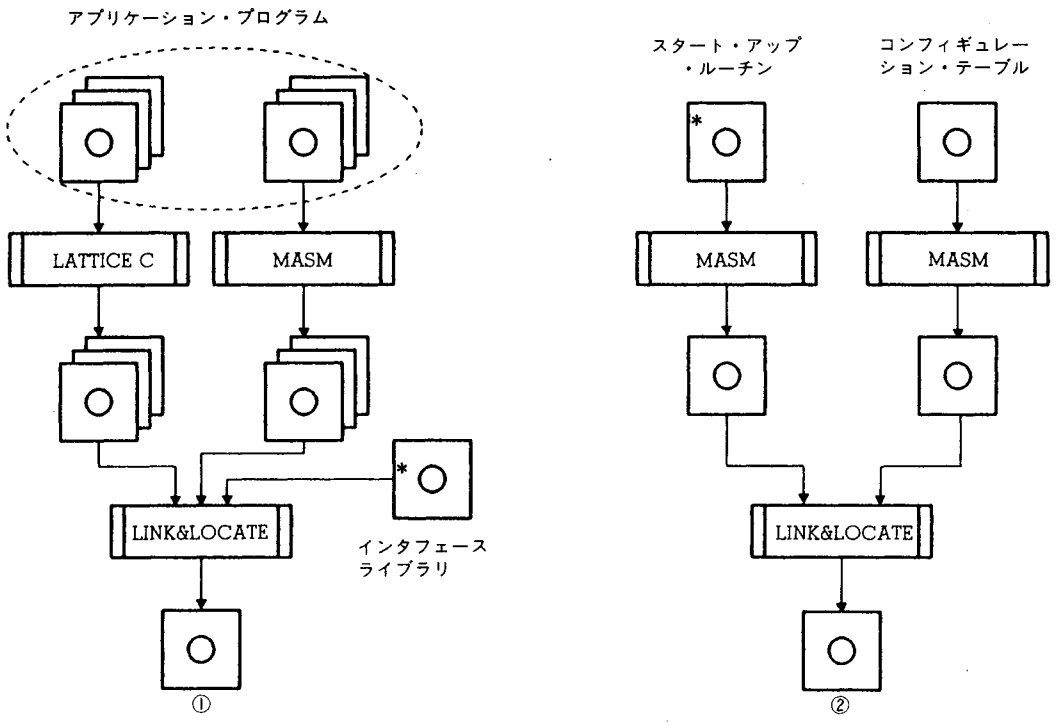
ユーザ・タスクとカーネルとのインタフェースはbrk命令で行われます。

インタフェース・ライブラリでは，C言語でカーネルを呼び出すための文字どおりインタフェース用のサブルーチンです。また，図中“＊”マークのついているものは次のような変更をしてください。

- インタフェース・ライブラリ（オブジェクト，ソース），スタート・アップ・ルーチン（ソース）ともにソースを他社ツール用に変更してください。具体的には次のとおりです。

  Vシリーズ用ニモニック記述 → 86系ニモニック記述

- インタフェース・ライブラリについてはさらに他社ツールでアセンブル後ライブラリを作成をしてください。
- コンパイラの仕様については，大半の86系コンパイラ（LATTICE C，MS-Cなど）は，当社，コンパイラと同じスタックの使い方をしていますので，変更の必要はありません。ただし，インタフェース・ライブラリのうち，ret_int，iret_wupの2つのシステム・コールは当社コンパイラに大きく依存していますので，他社のコンパイラを使用する場合は変更してください。

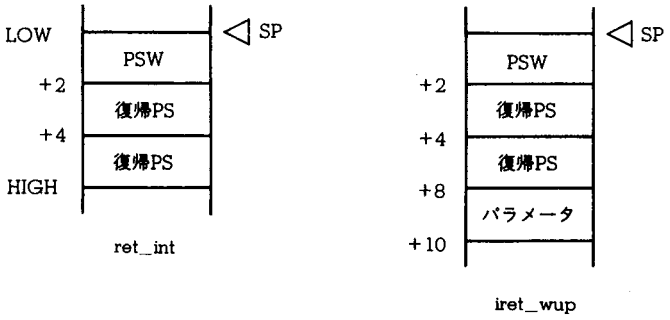


Q3.5 ★

インタフェース・ライブラリ中，ret_int，iret_wupは変更が必要とあるが具体的な内容は？

A3.5

それぞれ，スタックが次の状態でRX116のエントリに入る必要があります。下図に示すような形でOSに分岐（ソフトウエア・ブレーク）するように作成してください。



Q3.6 ★

RX116を用いたターゲット・システムを開発する際，そのNEC提供の開発環境としてOS/2が利用可能？　可能な場合の時期はいつ？

A3.6

・現在の開発ツールは，OS/のDOS互換ボックスのみ動作可能です。

・OS/2のネイティブなモードでのサポートは，予定がありません。現在は，ワーク・ステーション対応（HP9000，SUN3，NEWS）を優先して行っています。

Q3.7 ★

アプリケーション・タスクをPL/M86（Intel）で書くことは可能？

A3.7

可能ですが，インタフェース・ライブラリ（高級言語とのOS間のパラメータ整合を行う関数，標準ではNECのCコンパイラ用にできています）をPL／M用に修正する必要があります。

第4章　ニュークリアス

4．1　ニュークリアスの機能概要

Q4.1.1

RXシリーズのリアルタイムOSと，シングルチップ・マイクロコンピュータ（μPD79011，μPD79021）に組み込んだリアルタイムOSとの違いは？

A4.1.1

RXシリーズは，ITRON1仕様に準拠したOSです。

一方，μPD79011，μPD79021に内蔵されているOS（HS25）は，NECのオリジナル仕様です。両者の仕様概要を表4-1，表4-2に示します。詳細は，各ユーザーズ・マニュアルを参照してください。

表4-1　全体仕様

	RX320	HS25
仕様	ITRON	NECオリジナル
コード・サイズ	12.5 K注	約6 K
システム・コール数	55	25
システム・コール処理時間(sta_tsk)	169 μsec	83 μsec
最大タスク数	65 535	64
タスクの優先度	1－255	0－63
資源の動的な生成／削除	可能	不可
同期通信方法	メールボックス	メールボックス
	セマフォ	セマフォ
	イベントフラグ	直接メッセージ通信
セマフォの数（最大値）	65 535	256
セマフォの型	計数型	計数計
メールボックスの数（最大値）	65 535	256
システム・コールのタイムアウト指定	可能	不可
カーネルのロケーション・アドレス	リロケータブル	固定（0FC000～）
カーネルとのインタフェース	ソフトウエア・ブレーク	far call
提供形態	ソフトウエア	ソフトウエア
		内蔵ROM

注　RX320は，不要なシステム・コールを取り外して，カーネルを再構築できます。
12.5 Kはフルスペック時のサイズです。

保守／廃止

表4-2　システム・コール比較（1／2）

分類		機能	RX320	HS25
タスク管理機能		タスクの生成	cre_tsk	—
		タスクの起動	sta_tsk	sta_tsk
		タスクの削除	del_tsk	—
		タスクの終了	ext_tsk	ext_tsk
		タスクの終了と削除	exd_tsk	—
		異常終了処理の定義	def_ext	—
		タスクの異常終了	abo_tsk	—
		タスクの強制終了	ter_tsk	—
		プライオリティの変更	chg_pri	—
		同一プライオリティのキューの変更	rot_rdq	—
		タスク管理テーブルのアドレス取得	tcb_tdr	—
		タスクの状態取得	tsk_sts	—
		タスクの中断	sus_tsk	sus_tsk
		タスクの中断からの復帰	rsm_tsk	rsm_tsk
		タスクの起床待ち（指定時間なし）	slp_tsk	wai_int
		タスクの起床待ち（指定時間つき）	wai_tsk	—
		タスクの起床要求	wup_tsk	can_int
		タスクの起床要求解除	can_wup	—
		タスクの周期起床要求	cyc_wup	—
		タスクの周期起床要求解除	can_cyc	—
タスク間通信機能	イベント・フラグ	イベント・フラグの生成	cre_flg	—
		イベント・フラグの削除	del_flg	—
		フラグのセット	set_flg	—
		フラグのセット待ち	wai_flg	—
		イベント・フラグ管理テーブルのアドレス取得	flg_adr	—
	セマフォ	セマフォの生成	cre_sem	—
		セマフォの削除	del_sem	—
		セマフォへの返却	sig_sem	rel_src
		セマフォの獲得	wai_sem	req_rsc
				pol_rsc
		セマフォ管理アドレス取得	sem_adr	—
	メールボックス	メールボックス生成	cre_mbx	—
		メールボックス削除	del_mbx	—
		メッセージ送信	snd_msg	snd_msg
				snd_dir
		メッセージ受信	rcv_msg	rcv_msg
		メールボックス管理テーブルのアドレス取得	mbx_sdr	—
	直接通信	直接通信受信	—	rcv_dir
		指定タスクへのメッセージ送信	—	snd_dir
			—	snd_pol

保守／廃止

表4-2　システム・コール比較（2／2）

分　類	機　能	RX320	HS25
割り込み処理	割り込み処理の定義	def_int	def_int
	割り込みからの復帰	ret_int	res_int
	割り込み処理の禁止	dis_int	dis_int
	割り込み処理の許可	ena_int	ena_int
	割り込みハンドラ内でのDS0取得	fet_dat	—
	割り込みからの復帰とタスクの起床	iret_wup	sig_int
	現在の割り込みレベル取得	get_dvn	—
	レジスタ・バンク割り込みからの復帰	ret_rbi	—
	レジスタ・バンク割り込みからの復帰と起床	irrb_wup	—
	リスタート・アドレスの設定	—	res_int
例外処理機能	例外処理の定義	def_exc	—
	例外処理からの復帰	ret_exc	—
メモリ管理機能	メモリ・プールの生成	cre_mpl	—
	メモリ・プールの削除	del_mpl	—
	メモリの獲得	get_blk	get_mem
	メモリの解放	rel_blk	rel_mem
	メモリ管理テーブルのアドレス取得	mpl_adr	—
時刻管理機能	時刻設定	set_tim	set_tim
	時刻取得	get_tim	get_tim
その他	拡張システム・コールの定義	def_svc	—
	バージョンの取得	get_ver	—

Q4.1.2 ★

割り込み処理から拡張システム・コールを発行し，その中でシステム・コールした場合のスケジューリングは割り込み処理後に行われる？

A4.1.2

そのとおりです。

Q4.1.3 ★

各システム・コールが使用するスタック・サイズは？

A4.1.3

ret_int，iret_wupのシステム・コールは次の式で計算できます。

ret_int，iret_wup以外のシステム・コールは，「ユーザーズ・マニュアル　RX136　リアルタイム・オペレーティング・システム　基礎編」をご参照ください。

システム・コール	最大スタック・サイズ（バイト）
ret_int	22
iret_wup	24

Q4.1.4★

資源のキューイングはいくつまで可能？

A4.1.4

数に制限はありません。生成できた資源はすべてキューイングできます。

4.2 タスク管理

Q4.2.1

終了時処理ルーチン内で，発行できるシステム・コールに制限はある？

A4.2.1

終了時処理ルーチンは，プライオリティが1の通常タスクとして動作しています。したがって，この処理ルーチン内で発行できるシステム・コールの制限はありません。

Q4.2.2

ext_tskシステム・コールは，自タスクを休止状態（DORMANT）へ移行しようとしたとき，自ら定義したメッセージ受信用メールボックスにまだ要求があるか否かをチェックする？

A4.2.2

チェックはしません。

また，タスクが獲得した資源（たとえば，メモリ・ブロック）も返却しません。

Q4.2.3

RUN状態のタスクについて

(1)　RUN状態のタスクは，レディ・キューにつながれている？

(2)　同一のプライオリティに対してrot_rdqシステム・コールを発行した場合，RUN状態のタスクはどうなる？

(3)　同一プライオリティのタスクがレディ・キューに複数つながれているとき，同一プライオリティをもつタスクに対してsta_tskシステム・コールを発行した場合，タスク・スイッチングは生じる？

A4.2.3

(1)　つながれています。

RUN状態のタスクとは，実行条件がすべてそろって，実行権を割り当てられる順を待っているだけの状態にあるタスク，つまりレディ・キューにつながっているタスクの中で，最高プライオリティを持ち，かつそのプライオリティのレディ・キューの先頭につながれているタスクのことをいいます。

このように，RXシリーズでは，RUN状態のタスクをレディ・キューにつながれているタスクの中での特別なタスクとして扱っています。

(2)　rot_rdqシステム・コールは，指定プライオリティのレディ・キューの先頭を末尾につなぎ替えるという処理を行うものです。

したがって，RUN状態にあるタスクが同一プライオリティのタスクに対してこのシステム・コールを発行した場合，そのタスクはレディ・キューの末尾につながれることになり，ここでタスクのスイッチングが生じ，RUN状態のタスクが替わります。

(3)　タスク・スイッチングは生じません。

システム・コールを発行したタスクが引き続き動作します。

同一プライオリティのタスクがレディ・キューに複数つながれている場合，“レディ・キューにつながれている順”にRUN状態にします。このとき，sta_tskシステム・コールをかけられたタスクはレディ・キューにつながれますが，これにより，システム・コールを発行したタスクがレディ・キューから外れることはないため，タスクのスイッチングは生じません。

Q4.2.4

起床しているタスクに対して，wup_tskシステム・コールが発行された場合の動作はどうなる？

A4.2.4

起床しているタスクに対して発行された起床要求は，タスクが個々に持つ起床要求カウンタにキューイングされます。

このカウンタの内容は，SLEEP状態でないタスク（起床しているタスク）に対して発行されたwup_tskシステム・コール（起床要求）の回数を保持するエリアで，起床要求が発行されるごとに＋1されていきます。

このカウンタは，1回のslp_tskシステム・コールもしくはwai_tskシステム・コールを相殺していきます。したがって，次に起床しているタスクがslp_tsk（wai_tsk）システム・コールを発行しても，タスクはSLEEP状態にならず，タスクの持つ起床要求カウンタの内容が－1となります。

Q4.2.5 ★

OSが標準で準備しているタスク（コントロール・タスク）のようなものはある？

A4.2.5

ユーザのタスクが動作していないときは，アイドル・タスク（プライオリティ＝FFH）が動作し，HALT命令を実行しています。それ以外は，タスクではありませんが，一定周期（ユーザが設定）ごとにタイマ処理動作します。

Q4.2.6

すべてのタスクがREADY状態で，RUN状態のタスクがない場合はある？

A4.2.6

タスクがすべてREADY状態であるにもかかわらず，RUN状態のタスクがない場合は，通常はありえません。

RXシリーズでは，タスクへ制御を移す直前で割り込みコントローラ（μPD71059相当品）のISRを読み，サービス・ビットが立っていれば割り込みサービス中であると判断します。

ただし，初期設定時のOS内は割り込み禁止状態となっていますので，割り込みを受け付けたとは考えられません。

もし，すべてのタスクがREADY状態で，RUN状態のタスクがない場合は，次の点に注意してください。

①割り込みコントローラの初期化ミスはないか？

②割り込みコントローラのI／Oアドレスは，ターゲットと合っているか？

③割り込みハンドラの最後でret_intシステム・コール注を正常に発行しているか？

注　このシステム・コール中で割り込みコントローラにFIコマンドを発行しているため，正常にハンドラが終了していれば，タスクに制御が移ります。

Q4.2.7

次のシステム・コールで，タスク切り替えが起こる？

- rel_blk
- get_blk
- ret_int
- snd_msg
- rcv_msg

A4.2.7

タスクの切り替えが起こります。

システム・コール名	タスク切り替え注1	タスク切り替えの条件
rel_blk	○	①メモリ・ブロック返却によりget_blk待ちしていたタスクが条件を満たした場合注2 ②タイマ操作，割り込み処理等が行われ，そのためにREADY状態になったタスクがある場合注2
get_blk	○	切り出すべき大きさのメモリ・ブロックがシステム・コール発行時に確保できなく，また，メモリ・ブロックを獲得するために待ちに入る指定を行った場合（WAIT状態に入る）
ret_int	○	割り込みハンドラ内で発行されたシステム・コールにより，割り込み前に動作していたタスクよりプライオリティの高いタスクがREADY状態になった場合，rel_blk②と同様
snd_msg	○ ○	①メッセージ送信によりrcv_msg待ちしていたタスクが条件を満たした場合注2 ②rel_blk②と同様
rcv_msg	○	システム・コール発行時，メールボックスにメッセージが到着していなく，かつ，到着するまで待つための指定を行った場合（WAIT状態に入る）

注1. ○：本システム・コールの発行により，タスク切り替えが行われる可能性があるもの

2. タスクが切り替わる条件として，READY状態に新たになったタスクの方が，現在RUN状態のタスクよりもプライオリティ（優先度）が高いことがあります。

Q4.2.8

各タスクのスタック・サイズがcre_tskシステム・コール発行時に指定した値ではなく，コンフィギュレータで指定した値になってしまうのはなぜ？

A4.2.8

cre_tskシステム・コール内で指定したスタック・サイズがコンフィギュレータで指定したスタック・サイズより小さいときにこの現象が起こります。

RXシリーズでは，コンフィギュレータで指定したスタック・サイズは，システム内のディフォールトのスタック・サイズとして扱います。

このディフォールト・スタック・サイズは，cre_tskシステム・コール発行時にパラメータで指定される各タスクのスタック・サイズがディフォールト値より小さい場合に強制的にセットされるという仕様になっています。

Q4.2.9★

システム・コールの使用方法について

(1) cre_tskについて

(a) パラメータ・ストラクチャへのポインタというのはsta, stksz, tskpri, optionおよびtskdsなどのデータ構造体へのポインタ？　また構造体の要素名は上記のように決定している？

(b) タスク・アクセス・アドレスとタスク・スタート・アドレスの違いは？

(2) sta_tskについて

initcodeは，起動するタスクに与える初期情報値とあるが，起動されたタスクは，initcodeという変数名で参照できる？　できるとすればinitcodeは外部変数ということになるが？

(3) タスクの周期起動の最小単位がmsということであるが，100 μs毎にタスクを起動するにはどうすればよい？

A4.2.9

(1) (a) そのとおりです。メンバ名は何でもかまいません（並びかたと型のみ問題となります)。

(b) アクセス・アドレスは，そのタスクの管理データのアドレスのセグメント値です（この値はtcb_adrによって得られます)。

スタート・アドレスはタスクが始まるアドレスです。

(2) この値は，スタックに積まれます。タスク側で受け取るには次のように記述します。

```
task (a)
int a ;/*a is initcode*/
 {…
```

(3) ms以下の精度で指定することはできません。

Q4.2.10

cre_tskシステム・コール発行時に空メモリ・エリアがない場合，タスクは生成されないが，時間待ち指定のオプションはある？

A4.2.10

オプションはありません。

RXシリーズでは，オブジェクト生成関係のシステム・コールはすべてプログラム内の初期化部で記述されることを前提としています。初期化部は通常プログラムの先頭に置かれるため，この時点では，空メモリ・エリアは十分あるはずです。

したがって，オブジェクト生成（cre_）系のシステム・コールには，すべて時間待ち指定のオプションはありません。

Q4.2.11★

8087を使用していますが，タスク・スケジュール時，8087のレジスタもTCBにセーブされる？

A4.2.11

cre_tsk時のパラメータで，8087を使用するかしないかを指定するビットがあり，これで「8087を使用」を指定したタスクの切り替え時には，8087用のレジスタのセーブ/リストアを行います。

Q4.2.12★

420 Kバイトのエリアを1つの構造体として使用できる？

A4.2.12

OSの仕様としては問題ありません。Cコンパイラがサポートしているかが問題です（NECのCコンパイラCC70116では最大32 Kバイトまでは扱えます）。

Q4.2.13★

cre_tskのtskdsとは，データ・セグメントの値？　そうだとしたら1つのタスクで64 Kを越えるデータは扱えない？

A4.2.13

データ・セグメントの値です。ただし，sta_tskしたときに代入する値で，通常のタスク・スイッチのときには，この値は無視しますので，64 Kバイト以上のデータも扱えます。

Q4.2.14

アイドル・タスク内では，何を行っている？

A4.2.14

アイドル・タスク内では，HALT命令を発行しています。

これにより，CPUをスタンバイ状態とさせ，アイドル・タスク走行中，つまりユーザ・タスクが走行していない間，システム全体の消費電力が下がるようにしています。

Q4.2.15

rcv_msgシステム・コールを発行してWAIT状態にあるタスクをwup_tskシステム・コールで，READY状態にすることはできる？

A4.2.15

READY状態にすることはできません。

メッセージ待ちの状態もSLEEP状態も，“待ち（WAIT）”という状態には相違ありませんが，待ち状態からの解除条件（方法）がおのおの違います。

メッセージ待ちは，メッセージの送信によって解除されます（snd_msgシステム・コールの発行）。SLEEP状態は起床要求により待ち状態から解除されます（wup_tskシステム・コールの発行）。

したがって，rcv_msgシステム・コールを発行して待ち状態にあるタスクを待ち状態から解除するには，snd_msgシステム・コールの発行しか行えません。

ちなみに，メッセージ待ちのタスクに起床要求（wup_tskシステム・コール）を発行した場合，そのタスクの起床要求カウンタの内容が＋１となります。

Q4.2.16

タスクを強制終了したとき，そのタスクが保持していたメモリ・ブロック等の資源はRX側で解放する？

A4.2.16

資源の解放はRX側はしません。

メモリ・ブロック等の資源の解放は，ユーザ側が行う必要があります。タスクが終了する際に必要となる処理は，終了時処理ルーチンを定義し，その中で解放処理をしてください。

Q4.2.17

WAIT-SUSPENDED状態から，READY-SUSPENDED状態へ移行はある？

A4.2.17

あります。

WAIT-SUSPENDEDとは，WAIT状態中にSUSPENDされた状態であり，WAIT状態とSUSPEND状態との複合待ち状態のことです。

READY-SUSPENDEDとは，READY状態とSUSPENDED状態との複合状態のことです。したがって，WAIT-SUSPENDED状態において，そのWAIT状態が解除されれば，READY-SUSPENDED状態となります。

READY-SUSPENDED状態は，ただのSUSPENDED状態と同じです。

Q4.2.18

wai_tskシステム・コールを発行して，待ち状態に入っているタスクについて

(1)　タスクが起床したとき，この起床が指定時間経過によるものか，他タスクからの起床要求（wup_tskシステム・コールの発行）によるものかを判別する方法はある？

(2)　wup_tskシステム・コールの発行により起床要求をかけられたのちに指定時間が経過した場合，このタスクのタイマ管理テーブルはどうなっている？

A4.2.18

(1)　判別するには，エラー・コンディション・コードを見てください。

指定時間経過による起床の場合，起床したタスクには，エラー・コンディション・コード10H（タイム・アウト・エラー）が返されます。

これに対し，他タスクからの起床要求により起床した場合，起床したタスクには，エラー・コンディション・コード0H（正常終了）が返されます。

(2)　タイマ管理テーブルは，wup_tskシステム・コール終了時には削除されています。

wai_tskシステム・コールを発行して待ち状態にあるタスクに対してwup_tskシステム・コールが発行された場合，待ち状態にあったタスクはこの時点で待ち状態から完全に解除されます。

つまり，wai_tskシステム・コール発行時に設定されたタイマ管理テーブルは，上記のような場合，wup_tskシステム・コールの処理内で削除されます。

Q4.2.19

タスクで使用するDS0値はどのように設定したらよい？

A4.2.19

cre_tskシステム・コール時のパラメータとして指定します。

このとき，指定するDS0値は，実際にメモリ上にダウン・ロードされるDS0値と一致させる必要があります。

また，cre_tskシステム・コール時に指定しない場合，ユーザ・タスク中で指定します。

Q4.2.20 ★

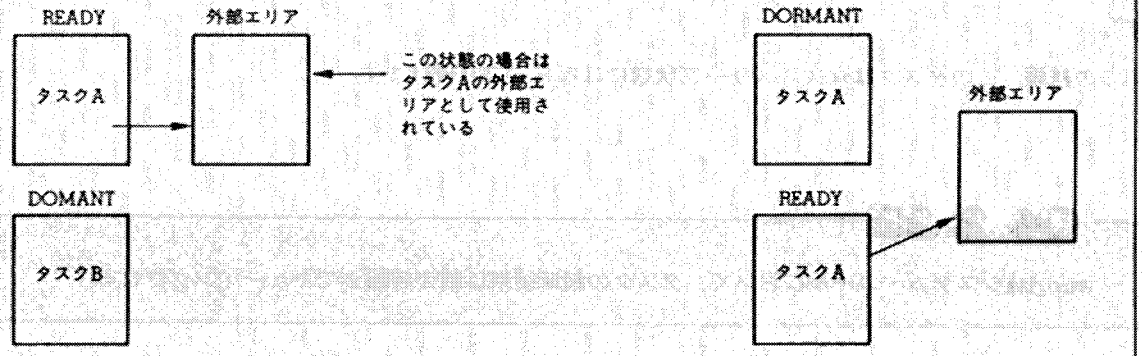
cre_tsk，def_intでデータ・セグメントを指定しているが何に使われているか分からない。すべて同じ(たとえば0)に指定したらどうなる？

A4.2.20

タスクの初期値として与えます。タスクのローカル・データ領域は，通常DS0相対で参照されます。すべてのタスクのDS0が同じでも特に問題はありませんが，タスクのローカル・データ領域を破壊する可能性がありますので好ましくありません。

Q4.2.21 ★

下図に示すDORMANT状態であるタスクBをREADY状態にするためにタスクAをDORMANT状態にし，タスクBにsta_tskを発行した場合，下図のように外部エリアを共通にした場合，READY状態になったタスクは外部エリアをRX116が再設定してくれる？　この再設定は上書きになる？



条件1．タスクA，タスクBのdsは同じで，クリエイト・タスクされる。

条件2．タスクA，タスクBは同時にREADY状態とはならない。

A4.2.21

DORMANT状態は，OS中に管理テーブルが残されている状態で，その際，すべての情報は，イニシャル時の値（cre_tskで指定した値）に設定されています。

sta_tskを実行すると，この管理テーブルの値に従ってタスクをスタートしますのでDS0も初期設定時の値が入ります。

ただし，OSがセットするのはDS0の値のみでこの外部エリアの内容は上書きされます。

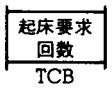
Q4.2.22 ★

cyc_wupシステム・コールを発行されたタスクがあり周期起床の時間をXmsとする。この周期起床を指定されたタスクがXms以内に処理を終了できなかったとき，何らかのエラーが生じる？　また，この場合ほかのタスクやOS自体に何か影響を及ぼすことがある？

さらに，エラーが発生した場合，どのタスクにどんなエラーが報告される？

A4.2.22

エラーはでません。このような場合，永久にこのタスクがready状態になります。詳細は下図を参照してください。



slp_tskを発行すると，TCB中の起床要求回数をチェックして，

要求回数　＞　0ならば1減

要求回数　＝　0ならばwait

になります。

一方cyc_wupを発行すると指定周期ごとにこの値を1増やします（上限値62以上の要求は切り捨てられます）。

この結果，このタスクは永久にスリープ状態にはならず走り続けます。

Q4.2.23

wup_tskシステム・コールにおいて，タスクの起床要求は最大何回までキューイングされる？

A4.2.23

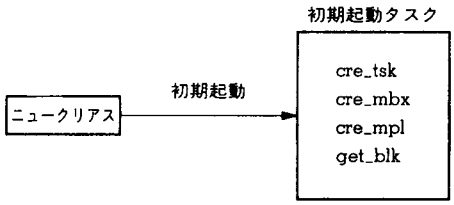
最大62回までキューイングされ，以降はエラーで返されます。

Q4.2.24

ニュークリアスから初期起動タスクへ起動が渡されてから，このタスクで他のタスクの生成，メールボックスの生成，メモリ・プールの生成およびメモリ・ブロック確保を行っても問題はない？

A4.2.24

まったく問題はありません。



Q4.2.25

ext_tskシステム・コールを発行した場合，発行した元のタスクに戻る？

A4.2.25

ext_tskシステム・コールにはエラー・コードがありませんので，基本的には発行したタスクには戻りません。

ただし，次の例外があります。

RXシリーズでは，割り込み処理中はスケジューリングをスキップして元の処理に戻る仕様となっています。割り込み処理中かどうかの判定は，割り込みコントローラのISRの値を読むことで行います。ext_tskシステム・コールを発行したときに，何らかの理由でISRが0以外であるとスケジューリングをスキップして，このシステム・コールを発行したタスクに戻ります。なお，その場合，エラー・コードは0（正常終了）になります。この理由としては，次のことが考えられます。

①コンフィギュレーション時に割り込みコントローラのポート・アドレスが誤っていた。

②ハードウエアの異常。

Q4.2.26 ★

終了時処理ルーチンが実行されているとき，そのタスクはどのような状態で実行されている？

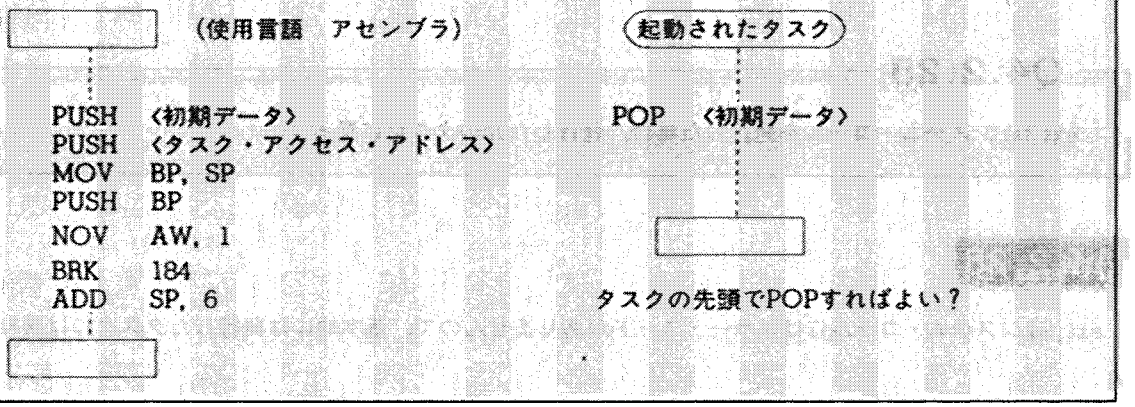
A4.2.26

終了時処理ルーチン動作中は，タスクの続きとして実行されるため，Ready（またはRUN）になります。ただし，終了時処理ルーチン実行中であるかどうかを示すフラグがありますので，このフラグを直接みるか，またはシステム・コール（tsk_sts）で知ることができます。

なお，終了時処理ルーチン実行中は，そのタスクのプライオリティは1に上げられます。

Q4.2.27 ★

システム・コールsta_tskで起動されたタスクでの初期データの受け方は？



A4 2.27

初期値はコンテキスト中＋20H番に入れられています。アセンブラ・プログラムでこれを参照するには，次のようにしてください。

```
mov     AW, [SP+20H]
```

Q4.2.28 ★

あるタスクが，アセンブラでコーディングされていた場合sta_tskで渡されるコードはどのように引き取ればよい？　次に示すC言語で引き取れる？

```
void
main (lint)
```

A4.2.28

・スタックに積まれて渡されます。

・センブラで記述する場合は，スタック内を参照してください。

・Cで記述する場合

```
void  main (para)
short  para; /*  short=int  */
```

で受け取れます。

Q4.2.29 ★

RX116でTCB情報のタスク・ステータスで，タスクの現在の状態で4001Hという状態が発生している。これはどのような場合に発生する？

A4.2.29

マシン例外ハンドラ（4000H）処理中で，その状態がREADY（1H）の状態です。

Q4.2.30 ★

TCBの内容の情報は提供してもらえる？

A4.2.30

購入していただいたお客様にのみディバグ用として情報提供しております。

Q4.2.31 ★

TCBのタスク状態が0226Hになることはある？

A4.2.31

そのような状態になることはありません。何らかの理由によるデータ破壊が考えられます。

226＝200＋20＋4＋2

200：link by priority
20：message wait
4：time out
2：sleep ← sleep中のタスクがmessageを待つことはありえない。

Q4.2.32 ★

タイマ・オペレーションの周期起床と遅延起床は，条件（時間）に達した時点で，ほかのタスクが動作中にかかわらず優先順位判定によりすぐに動作タスクが切り替わるのか？　それとも動作中のタスクが終了した時点で優先順位判定を行うのか？

A4.2.32

ほかのタスクが動作中かに関係なく切り替わります。

4.3 同期通信管理

Q4.3.1

1つのイベント・フラグに対して複数タスクがフラグをセットする場合，イベント待ちタスクのプライオリティのセットおよび解析の方法は？

A4.3.1

1つのイベントに対応させてタスクを起床させるときは，フラグがセットされるのを待っているタスクのプライオリティを高くしておく必要があります。

単純に事象の発生の有無を知るときは，イベント・フラグの各ビットにユーザ側で意味を持たせてください。

ただし，wai_flgシステム・コールが発行されるよりも先にset_flgシステム・コールが発行される可能性があるならば，set_flg時にリターン・パラメータとして返される“フラグ・セット以前のビット・パターン”を参照して解析する必要があります。

Q4.3.2★

イベント・フラグの操作のタイミングについて

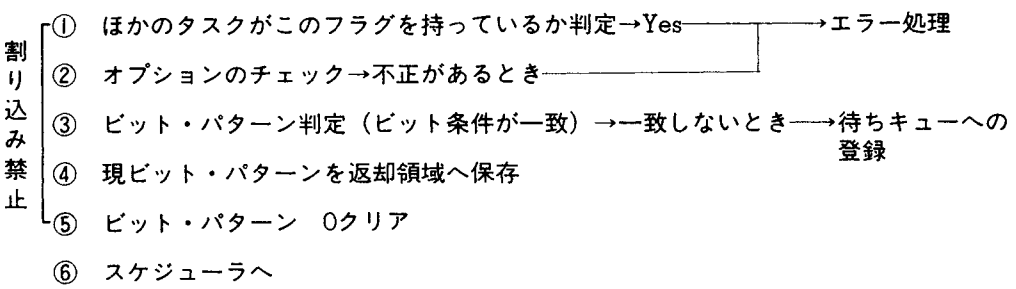
(1) wai_flgのシステム・コールで，オプションのbit2を“1”つまり条件を満足したあと，イベント・フラグのすべてのビットを“0”にするモードでwia_flgのシステム・コールを行ったとき，現ビット・パターン・ポインタで指定したアドレスには，すべてのビットを“0”にする前のイベント・フラグのビット・パターンが返る？　それとも常にすべてのビットにすべての“0”のパターンが返る？

(2) OS内でフラグの全ビットを“0”にするタイミングと，割り込みハンドラ内でフラグ・ビットするタイミングが競合（セットされるべきビットが“0”になる）することはない？

A4.3.2

(1) 0クリアする前のビット・パターンが返ります。

(2) wai_flg中で，フラグが0クリアされるまでの流れを下図に示します。この処理中はD1(割り込み禁止）で動作しますので，割り込み処理からのset_flgシステム・コールと競合することはありません。



Q4.3.3

OR条件でイベント・フラグがセットされるのを待っているタスクがあるとき，この待ちタスクよりもプライオリティの高い2つのタスクがset_flgシステム・コールを発行し，イベント・フラグをセットさせるとする。このタスクの待ち状態が解除される時点はいつになる？

A4.3.3

2つのタスクのセットする内容が，どちらがセットしても待ち状態から解除される条件を満たしているとすると，どちらかのタスクがフラグをセットした時点で待ちタスクは解除されます。

もう1つのタスクのプライオリティが待ちタスクより高いとしても，set_flgシステム・コールの一連の処理の中で待ち状態からの解除が行われるため，最初のset_flgシステム・コールによって，待ちタスクはWAIT状態からREADY状態へと移行します。

したがって，もう1つのタスクがset_flgシステム・コールを実行する時点では，待ちタスクはREADY状態にあります。

Q4.3.4★

wai_flgの5番目のパラメータであるポインタの指すデータ・エリアのフォーマットは？

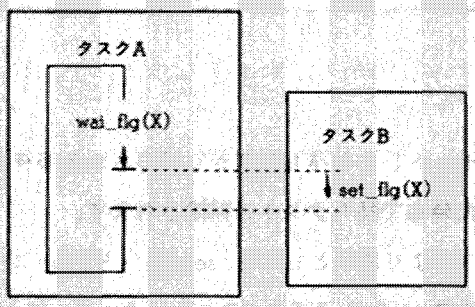
A4.3.4

long *p_tmoutです。

Q4.3.5 ★

イベント・フラグ処理について

(1) イベント・フラグのセット・タイミングにより下図のような局面でタスクAの動きはどうなる？



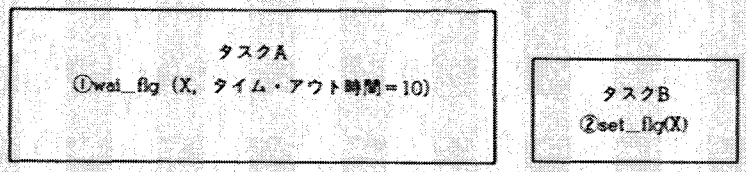
(a) タスクAが処理中タスクBに割り込まれて，タスクBがset_flgを発行する？

(b) タスクAがループ処理でwai_flgを発行した際に，

・タスクAはwai_flgのパターンが合っていた場合にすぐOSから起動される？

・wai_flgがタイム・アウト値を指定して発行されている場合，タイム・アウト値はタイマ管理テーブルに管理され，起動がかかった時点でタイマ管理テーブルが無効になるためタイム・アウトでOSから再起動がかかることはない？

(2) 下図に示すタスクAがwai_flg，タスクBがset_flgを発行し同期通信を行う場合，



(a) ①の時点で，タイマ管理テーブルが生成され，時限監視がOSによって開始される？

(b) ②の時点で，タスクAが再開した場合に，経過したタイム・アウト時間を知ることは可能？

A4.3.5

(1) (a) フラグ・パターンが合えば，タスクAがready状態になります。

(b) ・wai_flg発行時，パターンが合えば（すでにセットされている）すぐにでも実行します。

・タイム・アウト指定していても，待ちパターンがセットされれば，TMBからはずされます。

(2) アプリケーション・レベル（システム・コールを使用して）では，知ることはできません。

OS内部を直接操作する場合は，何クロックでタイム・アウトするかを知ることができます。ただし，この方法はOS内部を直接操作するので，OSバージョン・アップ時の動作保証はできません。

Q4.3.6

メールボックスに到着しているメッセージについて

(1)　メッセージを一括してリリースできる？

(2)　メールボックスを削除すると，到着しているメッセージはどうなる？

A4.3.6

(1)　一括してリリースすることはできます。

(2)　メッセージが到着しているメールボックスを削除すると（del_mbxシステム・コール発行），到着しているメッセージは，すべて強制的に切り出されたメモリ・プールに返却されます。

Q4.3.7 ★

1個のメールボックスから2個のタスクが受信する，アプリケーションの応用例はある？

A4.3.7

適切な例はありませんが，配列タスク（プログラムは1つで別タスクとして登録する）の場合は，このような仕様がないと動作しません。

Q4.3.8 ★

メールボックスでメールを渡すとき，アクセス・アドレスを共有体宣言しておくとよいとあるが具体的な方法は？

A4.3.8

当社発行のアプリケーション・ノートをご参照ください。

Q4.3.9

1つのメモリ・ブロックを複数のメールボックスに送ることはできる？

A4.3.9

できません。

複数のメールボックスに同じ内容を送りたい場合は，メールボックスの数だけメモリ・ブロックを獲得して，それぞれに送るようにしてください。

Q4.3.10

メールボックス処理関連システム・コールについて

(1) snd_msgシステム・コールを発行したとき，どのような場合にタスクの切り替えが起きる？

(2) 処理速度を上げる目的で登録タスク数を極力減らすために，タスク間のメッセージの受信をrcv_msgシステム・コールと同様にext_tskシステム・コールとsta_tskシステム・コールで制御できる？

(3) rcv_msgシステム・コールを発行したときに，timeout＝0（即時リターン）と指定する。タスクにメッセージが到着しているか否かで，エラー・コンディション・コードはどう違う？

A4.3.10

(1) タスクの切り替えが起きるかどうかは，送ったメッセージが受け取られたか否かによります。

・メッセージは送ったが，受け取りタスクが存在しない場合

→タスクは切り替わりません。

snd_msgシステム・コールを発行したタスクは，そのまま動作を続けます。

・メッセージを送ったメールボックスにすでにタスクが待っている場合

→メッセージを送ったタスクとメッセージを受け取ったタスクとのプライオリティにより，タスクが切り替わるか否かが決定されます。

これは，メッセージを受け取ったタスクの状態が遷移するからです。

（WAIT状態→READY状態）

「メッセージを送ったタスクのプライオリティ≧メッセージを受け取ったタスクのプライオリティ」の場合は，切り替わりません。

「メッセージを送ったタスクのプライオリティ＜メッセージを受け取ったタスクのプライオリティ」の場合は，切り替わります。

(2) ext_tskシステム・コールとsta_tskシステム・コールで制御する方法では，システム・コール処理時間がかかり，処理自体がかなり遅くなります。

タスク間のメッセージの送信／受信については，一般的には，rcv_msgシステム・コールを使います。

(3) エラー・コンディション・コードは，次のとおりになります。

・メッセージが到着している場合

エラー・コンディション・コード＝0H（正常終了）

・メッセージが到着していない場合

エラー・コンディション・コード＝10H（タイム・アウト・エラー）

Q4.3.11★

フラグ，セマフォについて

(1) フラグ，セマフォをwaitしているタスクに "wup_tsk" をかけるとそのタスクの状態はどうなる？ wup要求はどうなる？

(2) セマフォのリセットはやはり一度セマフォをデリートしてからクリエイトするしかない？その場合セマフォ待ちしていたタスクはどうなる？

A4.3.11

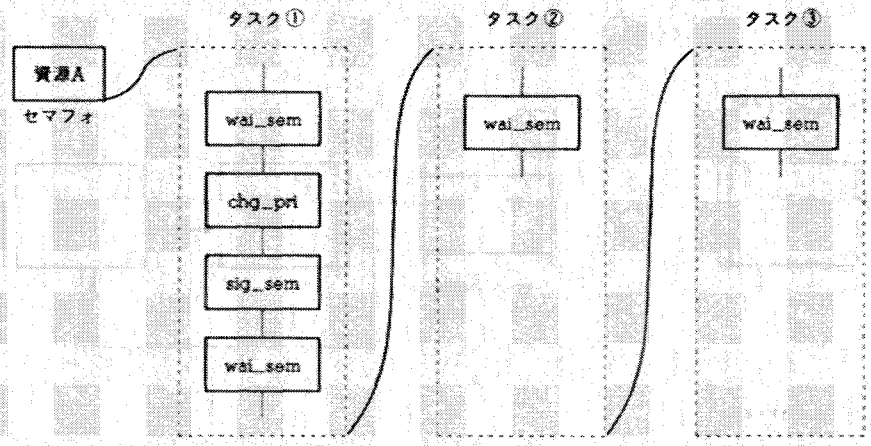
(1) フラグやセマフォをwaitしているタスクにwup_tskを発行しても，タスクのwaitは解除されません（wupはsleepを解除する要求です）。

wup要求自体は保存されます（複数発行された場合は，ネストして保存されます）。

(2) セマフォを一定値入れるシステム・コールがないため，セマフォをリセットするには一度デリートしてあらたにクリエイトしなければできません。なお，待ちタスクがあるセマフォをデリートすると，待っているタスクはwaitが解除されます（待ちタスクのwait_tskシステム・コールにはエラーが返ります）。

Q4.3.12 ★

下図のように同一プライオリティのタスクが資源Aのセマフォの待ちキューにならんでいる。



このとき，次に示すタスクの動作とセマフォの解釈に間違いはある？

タスク動作

↓

wai_semシステム・コールでセマフォの待ちキューにあったタスク①がREADY状態になり実行される

↓

自タスク（タスク①）のプライオリティを上げる（chg_priシステム・コール）

↓

いったん資源を開放する（sig_semシステム・コール）

↓

スケジューリングの結果，タスク①が続いて動作する

↓

タスク①が資源Aに対してタスクのプライオリティ優先順位のwai_semシステム・コールを発行

↓

スケジューリングによりタスク①が動作し，資源Aを使用した動作を行う

資源Aのセマフォの移り変わり

↓

タスク①

↓

タスク①

↓

タスク①　タスク②

↓

タスク②

↓

タスク②→タスク①

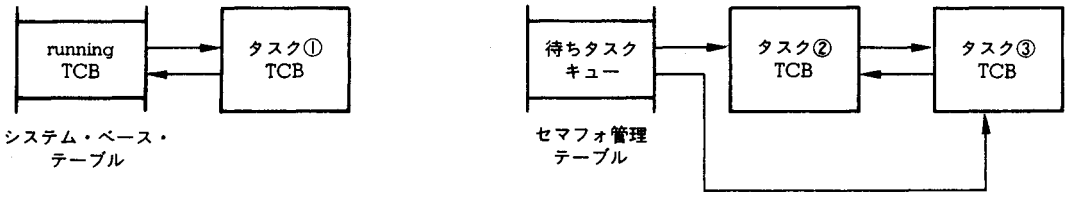
↓

タスク①

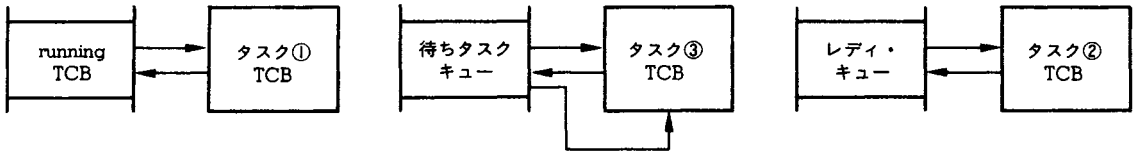
A4.3.12

5のとき，資源Aはタスク②→タスク①へは渡されません。つまり6の時点では，スケジューリングによってタスク②が実行されます。詳細は下図を参照してください。

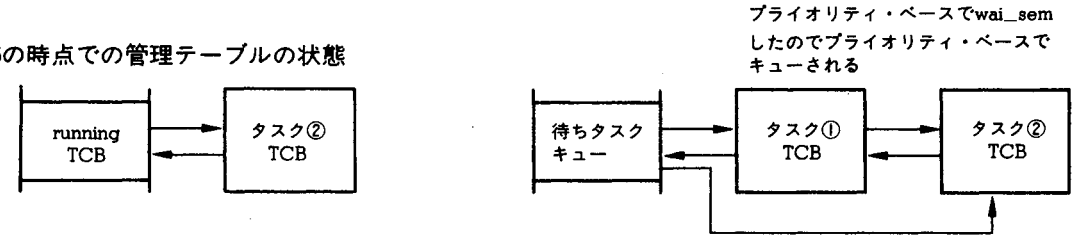
1，2の時点での管理テーブルの状態



3，4の時点での管理テーブルの状態



5，6の時点での管理テーブルの状態



Q4.3.13 ★

タスクAからタスクB，タスクCへSND_MSGする場合get_blkで確保したエリアを共有に使用してメールを送ったときOSは同じエリアを連続して使用したことにより，err＝04を返すことがある？

A4.3.13

・あります。SND_MSGする領域は，共有できません。

・もし，複数のタスクへ同一内容のメッセージを送るときは，複数個get_blkしてそれぞれメッセージを書き込んで送ってください。

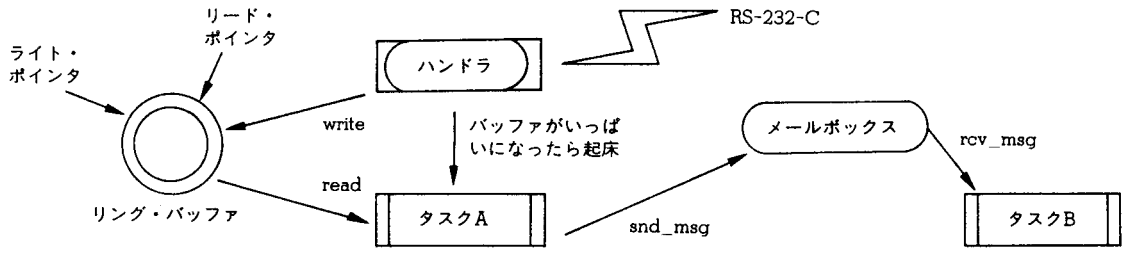
Q4.3.14 ★

SCUの割り込みハンドラ内よりメールを送りたいが，ハンドラ内でget_blkは不可能であるため，OSの管理外のRAM領域にエリアを設け，そのポインタをSND_MSGすることが可能？　また，そのほかにハンドラからタスクへの通信手段はある？

A4.3.14

・OS管理外のメモリ領域を用いて，メッセージ送信（メールボックス）を行った場合の動作保証はできません。

・RS-232-Cからのデータ受け取り→受け取りデータ送信であれば，次の方法が考えられます。



4.4 メモリ管理

Q4.4.1

メールボックスを使用してメッセージの交換を行うとき，メモリ・プール外のエリアをメッセージ・エリアとして使用できる？

A4.4.1

メッセージ交換にメモリ・プール外のエリアを使用しないでください（この場合の動作の保証はできません）。

必ず，get_blkシステム・コールを発行して獲得したエリア（メモリ・ブロック）を使用してください。その理由は，get_blkシステム・コールで獲得したエリアには，特別の管理情報が保持されるエリアがあり，OSはこの情報を参照して各処理を行っているためです。

Q4.4.2

メモリ・プール0番について

(1) ユーザが，メモリ・プール0番からget_blkシステム・コールを発行してメモリ・ブロックを獲得することはできる？

(2) メモリ・プール0番のブロック・サイズは？

A4.4.2

(1) ユーザが，メモリ・プール0番からメモリ・ブロックを獲得することはできます。

メモリ・プール0番はRXシリーズが自動的に生成するものであり，この点は他のメモリ・プール（1番-15番）とは違います。また，コンフィギュレーション時にユーザが指定したフリーRAMエリアすべてがメモリ・プール0番として生成されます。したがって，メモリ・プール0番の生成および削除についてはユーザの意志で行うことはできません。

また，OSが使用するメモリ・エリアは，すべてメモリ・プール0番から切り出されて使用されます（例 各管理テーブル等）。

これらの事柄からメモリ・プール0番は他のメモリ・ブロックと比べて特殊といえますが，ユーザがシステム・コールを発行し，0番のエリアの一部を使用することに問題はありません。

(2) 16バイトです。

したがって，get_blkシステム・コール発行の際，パラメータの連続領域の指定で，“1”を指定すると，16バイトのメモリ・エリアが獲得されます。

Q4.4.3

メモリ・プールからメモリ・ブロックを獲得する際の待ち方はどうなる？

A4.4.3

メモリ・プールからメモリ・ブロックを獲得する際の待ち方は，FIFO（First-In First-Out）固定となっています。

このことは，メモリ・プール番号には関係せず，どのメモリ・プールにおいても同様です。したがって，メモリ・ブロック獲得待ちをするときには，タスクは獲得要求を出した順に待ち行列につながれます。

Q4.4.4 ★

コンフィギュレータで設定したメモリ・プールのスタート・アドレスおよびエンド・アドレスに対して，cre_mplのシステム・コールを行ったあとの残量を示す値の入っているアドレスは？

A4.4.4

OSでは空きサイズを管理していません。次の方法で計算してください。

(1) mpl_adrシステム・コールでmpl #0の管理テーブルのセグメント値を得る。

(2) オフセット0から空きブロック管理テーブルのセグメント値を得る。

(3) (2)の値のオフセット＋｛08（ブロック・エンド・アドレス)｝－｛+06（ブロック・スタート・アドレス)｝でブロック・サイズを得て，空きサイズに加算する。

(4) (2)の値のオフセット0から次の空きブロック管理テーブルのセグメント値を得る。もし，この値が0ならば，キューの終わりとして処理を抜ける。

あとは(3)，(4)を繰り返し行います。

Q4.4.5

メモリ・ブロックについて

(1)　メモリ・ブロックの最大生成可能数に制限はある？

(2)　メモリ・ブロック管理テーブルのメモリ・アドレスは？

(3)　あるメモリ・プールからメモリ・ブロックを切り出したままで，そのメモリ・プールを削除した場合，切り出されていたメモリ・ブロックはどうなる？

A4.4.5

(1)　メモリ・ブロック生成自体に数の制限はありません。

したがって，メモリ・ブロックの制限とは“切り出すメモリ・エリア”がなくなるまでということになります。

(2)　メモリ・ブロック管理テーブルのアクセス・アドレスは，セグメント値：オフセット値の形式で，（メモリ・ブロック・アクセス・アドレス）：（－16）になります。

詳細は，RXシリーズのユーザーズ・マニュアル　テクニカル編の**「3.7.2　メモリ・ブロック」**を参照してください。

(3)　削除されるメモリ・プールから切り出されていたメモリ・ブロックは何の処理もされません。また，このメモリ・ブロックを使用しているタスクに対しても，何の保障もされません。つまり，メモリ・プールごと返却されてしまい，すべてメモリ・プール0番に属する空エリアとなります。

したがって，このブロックを継続して使用することは誤動作の原因となります。

Q4.4.6

解放済みのメモリ・ブロックに対して，もう1度rel_blkシステム・コールを発行した場合，リターン・パラメータはどうなる？

A4.4.6

2度目の解放はエラーとなります（エラー・コンディション・コード'04 H'）。

Q4.4.7 ★

同一タスク内でget_blk（・・・・）を2回行った場合，パラメータが同一でも別々のブロックが切り出される？

A4.4.7

別々のブロックが切り出されます。ただし，k_blkの値（獲得したブロックのアドレス）は，別の変数に代入しておく必要があります。

Q4.4.8 ★

CPUにV50，OSにRX116でメモリ空間が1Mバイト以上になった場合どのような対策をとればよい（バンク切り替え方法等）？

A4.4.8

次の2つの方法が考えられます。

(1) RX116のカーネルを改造する。具体的には，TCB（タスク・コントロール・ブロック）を拡張し，タスクが使用するバンク番号を格納します。また，スケジューラも修正し，ディスパッチ(タスクの切り替え）が行われたときにマッピングし直すようにします。

長所：汎用性が高い。

短所：リスクが大きい，費用がかかる（ソース購入が必要)。

(2) バンク・メモリのアクセス権をセマフォで排他制御する。

長所：(1)の方法と比べて簡単リスクが小さい。

短所：バンク・メモリを操作できるタスクが限られるのでアプリケーションによっては非常に効率が悪い。

Q4.4.9★

V33で，リピート命令を使ってブロック転送を行いたいが，最大64 Kバイトのメモリ・データをほかのメモリ空間へブロック転送を行うと，リピート命令は外部割り込みが不可のためOS，ほかの割り込みの機能が約10 msecのオーダでまったく停止することになり，OSの動作，ほかの割り込み処理の保証ができない。本当にリピート命令中は外部割り込みは受け付けない？　OSの動作，ほかの割り込み処理を保証する方法は？

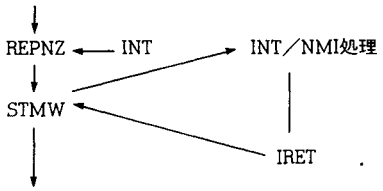
A4.4.9

リピート・プリフィクス命令付き，転送命令については次のようになります。

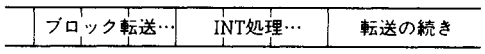
・リピート・プリフィクス命令と次の命令（プリミティブ・ブロック転送命令）の間は割り込みを受け付けない。

・プリミティブ・ブロック転送命令の終了時は受け付ける。

したがって，次のようなイメージとなります。



このときのバスの状態は



以上のことから，ブロック転送中にH／W割り込みが発生した場合，ブロック転送を中断し，割り込み処理を行います。割り込み処理終了時，IRET命令によりブロック転送のつづきを行います。

注意　割り込み処理にてCWの値が変化するとブロック転送数がそれにあわせて変わります。割り込み処理では，先頭でCWを退避しIRET直前でCWを復帰するなどして割り込み処理で変化することのないようにしてください。

4.5 割り込み管理

Q4.5.1

割り込みコントローラの割り込み優先順位を変更して使用できる？

A4.5.1

割り込み優先順位は，固定で使用してください。その理由は，OS内部で割り込みコントローラに対してFIコマンドを発行しているためで，割り込みルーチン内で優先順位の変更を行った場合などは正常動作しません。

Q4.5.2

割り込み処理ハンドラからの復帰を行うiret_wupシステム・コールをアセンブラで呼ぶには，どのような記述をする？

A4.5.2

次に，割り込み処理ハンドラ内の一般的な記述例を示します。

例

```
sample  proc   near（またはfar）
        push   R
          ⋮      ;割り込みハンドラの処理
        pop    R
        push   （タスク・アクセスadr）
        brk    192
sample  endp
```

Q4.5.3 ★

RETIを用いて割り込みハンドラを終了するときFIコマンドを発行する必要がある？

A4.5.3

FIコマンドが必要です。

Q4.5.4

RXシリーズの割り込み処理終了の方法は？

A4.5.4

次に，割り込み処理終了方法を示します。

①iret_wupシステム・コールまたはret_intシステム・コールを発行する。

②割り込み処理ルーチン内で割り込みコントローラに対しFIコマンドを発行し，IRET命令でリターンする。

①と②の違いは，①はスケジューラを通るが，②はスケジューラを通らないことです。RXを使用したシステムにおいて，②の方法を用いても結構ですが，割り込み処理終了後にスケジューラを通らないため，割り込み処理ルーチン内でタスクの起床等を行っても，必ず割り込まれたタスクが実行されることになります(タスクのディスパッチングは行われません)。

Q4.5.5

0番地からの割り込みベクタは，ROM化できる？

A4.5.5

割り込みベクタはRXが初期化しますので，この部分をROM化することはできません。

Q4.5.6★

タスクが割り込み状態で，システム・コールを発行した場合に，ほかのタスクが存在しない状態で，システム・コールの処理からタスクへ制御が移行した際には，割り込み禁止状態は保証される？　保証する方法は？

A4.5.6

・OS内でEIにする場合，そのとき実行していたタスクのステータス・フラグ（PSW）をチェックして，EIの場合のみ行っています（PSWはスタックに積まれている）。

・この動作（つまり，タスクがDIならばOS内部もDI）は保証します。

Q4.5.7 ★

割り込み禁止について

(1) システム・コールでの割り込みを禁止する理由は？

(2) 割り込み禁止時間中に割り込みを受け付けた場合（速いボー・レートでの受信割り込み）の対策は？

A4.5.7

(1) OSの中ではさまざまな情報をキューにして管理していますが，OS内でこのキューを操作する間は割り込み禁止（DI）にしなければなりません。なぜなら，キュー操作中（キューは不完全な状態）に割り込みが入り，システム・コールが発行されると不完全なキューのまま処理を行い，暴走する危険があるからです。

(2) 割り込み処理中であっても，割り込み許可（EI）にしておけばそれより高い優先度の割り込みを受け付けます。

Q4.5.8 ★

RX116は，多重割り込み（割り込みハンドラ内でEI命令を実行）は可能？

A4.5.8

可能です。問題ありません。

Q4.5.9 ★

RX116で，システム・コール時間＝割り込み禁止時間ではないはず。システム・コール内で，外部割り込みやインターバル割り込みが発生しても問題はない？

A4.5.9

問題ありません。OS内では，割り込み処理に分岐しては困る期間（キューの操作中）のみDIにしています。

Q4.5.10 ★

割り込みハンドラ中で，wup_tskシステム・コールを発行するとき，起床させるタスクが割り込み前に実行中だと，エラー（12H）を返してしまう？

A4.5.10

割り込みハンドラ内で“wup_tsk”を自タスクに発行した場合でもエラーにしない仕様です。

Q4.5.11 ★

ret_int, iret_wupをアセンブラ内からマクロ・コールしたいが，C言語からのコールの場合の使用方法は？　その場合のレジスタ退避の方法は？

A4.5.11

インタフェース・ライブラリのうち，ret_intとiret_wupは，NEC製Cコンパイラに依存した作りにしています。

ret_intの場合	iret_wupの場合
brk　185	push　タスク・アクセス・アドレス
	brk　192

Q4.5.12 ★

RX116, 136とμPD71059を使用しているシステムで，μPD71059のINT7の不正割り込みが発生した場合，INT7用の割り込みハンドラ内で不正割り込み処理を行うときret_intシステム・コールを使用できる？

A4.5.12

・不完全割り込みは，ハードウエアのロジックが誤っているため起こるものです。RXシリーズは，正常なハードウエアを想定して作られているため，完全割り込みに対する処理は考慮しておりません。

4.6 タイマ管理

Q4.6.1

タイマ管理機能として，次の処理は可能？

①各タイマでタイマ要求が発生した場合，タイマ管理エリアにタイマ情報をセットする。

②タイマ管理タスクで，OSからの周期起床を待ち，タイマ管理エリアにセットされている各タイマ値をカウント・ダウンする。このとき，値が0となった場合は，タイム・アウトが発生したとみなして，タイム・アウト・メールを該当するタスクに通知する。

A4.6.1

①②ともに問題はありませんが，一定周期でこの処理を行うことになりますので，CPUへのオーバヘッドとならないように，適切な周期を設定してください。

Q4.6.2★

RX116タイマ・クロックで，CPUを動かすクロックをダウン・カウントしてOSのタイマ割り込みを発生させていると思われるが，そのCPUのクロックとして上限，下限はある？

A4.6.2

周波数に関して制限はありませんが，ダウン・カウントする値は2バイトの領域ですので，次の制限があります。

例 5 MHz[注]を入力させた場合

システム・クロック≦（1000／5000，000）×FFFFH（msec）≒13（msec）

注 この値は，タイマ・コントロール・ユニットに入る値です。

Q4.6.3 ★

RX116で，システム・クロックを2 msecにしてタイマーの設定値をその倍数以外にした場合どうなる？

A4.6.3

システム・クロックで設定する値は，精度を意味します。

システム・クロックにn msecを指定した場合，システム・コールで指定した時間（m msec）の精度は，

m－n～m＋n

になります。

精度を高くするには，OSのシステム・クロックを小さくすればよいのですが，そうするとOSのオーバヘッドが大きくなりますので，両者のかねあいを考えて決めてください（一般的には5 msec以上で使うケースが多いようです）。

Q4.6.4 ★

V53の最低動作クロックが2 MHzのとき，1 msのシステム・クロックでRX136は完全に動作する？

A4.6.4

CPUのパワーの大半をOSのタイマ処理が使っています。

たとえば，タイマ処理が

90×16／2×1／1000×100＝72％

少なくとも72％がOSを使ってしまいます。

Q4.6.5 ★

RCV_MSGのシステム・コール発行時，タイム・アウト指定を次のようにした場合で，タイム・アウト指定時間までにメッセージを受信した場合タイマはキャンセルされる？　キャンセルされない場合，再び受信待ちの状態で，先のタイム・アウトでリターンされるのか？

A4.6.5

RX116の仕様では，タイム・アウト指定してメッセージを受け取ったときタイマはキャンセルされます。

Q4.6.6 ★

RX116の初期設定で，タイマ・クロックをV50よりもらう必要がある？　その方法は？　また，OSのタイマを動かすのにクロック値の制限は？

A4.6.6

・RX116をV40／V50で使用する場合，クロックは内部TCU（タイマ・コントロール・ユニット）からもらいます。

・クロック割り込みの周期があまり小さいとCPUに対するOSのオーバヘッドが大きくなります。実用上の最小値は1 msecくらいです。

最大値は，ダウン・カウント値領域が2バイトなので，0FFFFHが上限になります。

4.7 例外処理

Q4.7.1★

マシン例外が発生し，エラー・コード807AHが返されることがある。原因は？

A4.7.1

807AHは，未定義命令例外のエラー・コードです。プログラムが暴走して，データ領域を実行していることなどが考えられます。例外時のエラー・コードは次のとおりです。

8000h	ゼロ除算例外
8004h	算術オーバフロー例外
8005h	チェック・インデクス境界オーバフロー例外
8007h	不正brk命令例外
807ah	未定義命令例外
8082h	コプロセッサ不在例外

4.8 拡張システム・コール

Q4.8.1 ★

拡張システム・コールから拡張システム・コールをコールしても問題はない？

A4.8.1

問題ありません。

第５章　初期化

Q5.1

割り込みコントローラμPD71059およびタイマμPD71054の設定を変更することはできる？

A5.1

周辺のコントローラのうち，割り込みコントローラとタイマ（システム・クロックとして使うチャネルのみ）の設定は，ユーザ側は変更しないでください。

Q5.2

RX側で，μPD71059に対して初期化中，割り込み要求の優先順位はどのように設定する？

A5.2

回転なしのモードです。したがって，優先順位は固定，レベル０＞レベル７となります。

Q5.3 ★

内部DMA，シリアルについては，ユーザが初期化しなければいけない？　コンフィギュレータでは設定できない？

A5.3

そのとおりです。設定する場所は，スタート・アップ・ルーチンが適当と思われます。

Q5.4 ★

コンフィギュレーション・テーブルおよびリセット・ルーチンのリストを見るかぎり，OSのリセット・ルーチンではwcuの設定（WCY1，WCY2，WMB）を行っていないが，wcuに対するリセットによるディフォールト値以外の設定を行う場合は，アプリケーション・レベルで行わなければならない？　また，OSのリセット・ルーチンで行う方法がある？

A5.4

リセット・ルーチンは，オブジェクト契約でもソース提供されますので，ユーザ側で自由にイニシャライズ項目を変更できます。

第 6 章　その他

Q6.1 ★

RX116ディバグ・シミュレータに関して

(1)　標準インテルHEX形式のアプリケーションをディバグ可能？

(2)　HEX形式のディバグを行う場合，アプリケーションのロード可能なアドレスの範囲はどうやって算出する？　計算式，具体例は？

A6.1

(1)　可能です。

(2)　PC-9800のVRAMアドレスA800:0～C000:0の96 Kバイトを使用してください。

Q6.2 ★

ターゲット側に組み込むRXDEB core 116の容量（通信BIOSも含む）は何バイト？
また，通信BIOSは何Kバイト？

A6.2

通信BIOS込みで約23 Kバイトです。

Q6.3★

RX136マルチ・タスク・ディバッガにおけるフロー制御（XON／XOFF）とは，μPD71051のCTS，RTS端子を使用した送信制御と解釈すればよい？

A6.3

マルチ・タスク・ディバッガにおけるフロー制御は，ソフトウエア的に行います。このため，μPD71051のCTS，RTS信号は使用していません。

たとえば，

・ホスト・ディバッガがデータ受信時データを受け付けなくなると（たとえば，キーボードでCTRL_Sを押すとか），ホストからターゲットへCTRL_Sデータを送信します。

・ターゲット・ディバッガはこの信号を受け取って送信を停止します。

Q6.4★

RX116ディバグ・シミュレータについて

(1) RX116ディバグ・シミュレータを利用してディバグを行う場合，作成したアプリケーション・タスクにディバグ用の命令（MS-DOSのファンクション・コールを利用）を組み込んで行うことが可能（複数のタスクから同時にファンクション・コールを行っても正常に動作する）？

(2) ディバグ・シミュレータでディバグできるアプリケーション・プログラムのサイズ（最大）は？

A6.4

(1) MS-DOSがリエントラントな作りになっていませんので，利用できません（1つのタスクからでしたら利用できます）。

(2) 使用可能なMS-DOSの空きサイズに依存します。

CHKDSKコマンド時の空きサイズから，次の大きさを引いた残りがアプリケーションのサイズです。

例

シミュレータ・プログラムのサイズ：120 Kバイト

シミュレータ・プログラムのRAM：32 Kバイト

RX116のサイズ：11 K

Q6.5 ★

RX116上で動作するリモート・ディバッガ（簡易ディバッガ）はある？

A6.5

あります。

Q6.6 ★

ディバグのため，2つのタスク（イニシャル・タスクとディバグ対象タスク）で起動しようとすると暴走するのはなぜ？

・設定条件

イニシャル・タスクでは，クリエイト・タスク・ルーチンは1個しか発行していないが，TCBを見ると，次に示すように4個のタスクがリンクされてしまっている。

イニシャル・タスク	E0000H	プライオリティ1，ID1
意味不明なタスク	E1000H	プライオリティ1，ID2
ディバグ対象タスク	E8000H	プライオリティ1，ID3
DUMMYタスク		

A6.6

いつこの意味不明なタスクが生成されているかにより原因が異なります。

・イニシャル・タスクに分岐したときすでにある場合

→コンフィギュレーションの誤りが考えられます。

・イニシャル・タスク起動後の場合

→ユーザ・プログラムの誤り(cre_tskのパラメータの誤り，管理テーブルの破壊)が考えられます。

Q6.7 ★

ITRONのディバグで，ディバグ時にパーソナル・コンピュータとV40，V50のSCUと通信する場合，TxD，RxDの通信のみでよいのか？　それともSRDYも含めてすべて使用しなければいけない？

A6.7

TxRDYとRxRDYで制御を行います。SRDYの制御は行いません。

Q6.8 ★

RX116マルチ・タスク・ディバッガで「割り込みハンドラなどの非タスク部分はディバグできません。」とあるが，具体的にどういうこと？

A6.8

このディバッガは，OSの機能を使用してブレークをかけています（タスクをsuspendします）。このため，非タスクのもの（ハンドラ）にはブレークがかけられません。

アンケート記入のお願い

保守／廃止

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］ RXシリーズ Q & A集 インフォメーション

(EEI-609A(第2版), May 1992 M)

［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） (　　)

お電話番号 (　　)

お仕事の内容 (　　)

お名前 (　　)

1. ご評価（各欄に○をご記入ください）

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

2. わかりやすい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

3. わかりにくい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

4. ご意見，ご要望

5. このドキュメントをお届けしたのは

NEC販売員，特約店販売員，NEC半応技本部員，その他（　　）

ご協力ありがとうございました。

下記あてにFAXで送信いただくか，最寄りの販売員にコピーをお渡しください。

NEC半導体応用技術本部インフォメーションセンター

FAX : (044)548-7900

キリトリ

保守／廃止

保守／廃止

NEC 日本電気株式会社

本社	〒108-01 東京都港区芝五丁目7番1号(日本電気本社ビル)	
半導体第一第二販売事業部	〒108-01 東京都港区芝五丁目7番1号(日本電気本社ビル)	東京(03)3454－1111
関西支社 半導体販売部	〒540 大阪市中央区城見一丁目4番24号(日本電気関西ビル)	大阪(06)945－3178 大阪(06)945－3200
中部支社 半導体販売部	〒460 名古屋市中区栄四丁目14番5号(松下中日ビル)	名古屋(052)242－2755

北海道支社	札幌(011)231－0161	立川支社	立川(0425)26－0911
東北支店	仙台(022)261－5511	千葉支社	千葉(0472)27－5441
盛岡支店	盛岡(0196)51－4344	静岡支社	静岡(054)255－2211
山形支店	山形(0236)23－5511	沼津支店	沼津(0559)63－4455
郡山支店	郡山(0249)23－5511	浜松支店	浜松(053)452－2711
いわき支店	いわき(0246)21－5511	北陸支社	金沢(0762)23－1621
長岡支店	長岡(0258)36－2155	福井支店	福井(0776)22－1866
水戸支店	水戸(0292)26－1717	富山支店	富山(0764)31－8461
神奈川支社	横浜(045)324－5511	京都支社	京都(075)221－8511
群馬支店	高崎(0273)26－1255	神戸支社	神戸(078)332－3311
太田支店	太田(0276)46－4011	中国支社	広島(082)242－5504
宇都宮支店	宇都宮(0286)21－2281	鳥取支店	鳥取(0857)27－5311
小山支店	小山(0285)24－5011	岡山支店	岡山(0862)25－4455
長野支店	長野(0262)35－1444	四国支社	高松(0878)36－1200
松本支店	松本(0263)35－1666	新居浜支店	新居浜(0897)32－5001
諏訪支店	諏訪(0266)53－5350	松山支店	松山(0899)45－4111
甲府支店	甲府(0552)24－4141	九州支社	福岡(092)271－7700
大宮支社	大宮(048)641－1411	北九州支店	北九州(093)541－2887

（技術お問い合せ先）

半導体応用技術本部 第一応用システム技術部	〒108-01 東京都港区芝五丁目7番1号(日本電気本社ビル)	東京(03)3798－6105
半導体応用技術本部 第二応用システム技術部	〒540 大阪市中央区城見一丁目4番24号(日本電気関西ビル)	大阪(06)945－3383
半導体応用技術本部 第三応用システム技術部	〒460 名古屋市中区栄四丁目14番5号(松下中日ビル)	名古屋(052)242－2762
半導体応用技術本部 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)548－8890

インフォメーションセンター
FAX(044)548－7900