# NS32032上へのITRONインプリメントの検討

坪田 秀夫　　中村 和夫　　榎本 龍弥
三菱電機（株） LSI研究所

## 1．はじめに

ITRONは32ビット・ノイマン型アーキテクチャを持つマイクロプロセッサ用リアルタイムオペレーティングシステムとして、東京大学情報科学科で機能設計された。現在、ITRONがインプリメントされているのは 8086，68000 系の 16 ビット CPUである。

今回、我々はITRONを米 National Semiconductor社の 32 ビットマイクロプロセッサ NS32032にインプリメントすることについての検討を行なった。本稿では、NS32000 シリーズのアーキテクチャとともに、ITRON/32000 の設計概要、並びにハードウェアに依存する割り込み処理について述べる。

## 2．NS32000 CPU アーキテクチャ

NS32032 は完全な 32 ビットアーキテクチャであり、対称性の良い２アドレッシングモードと高級言語(HLL) 用の命令セットを備えている。 以下、NS32000 シリーズのアーキテクチャ上の特徴について述べる。

### 2.1 命令セット

NS32032 は高級言語向き、効率の良い命令コードをサポートしている。

- 数次元配列操作命令(INDEX) 、及びアドレッシング・モード（スケールド・インデックス）
- 多岐分岐命令(CASE 文)
- ビット、及びビットフィールド操作命令
- 使用頻度の高い命令コードの短縮化( 1 ～ 3バイト)

### 2.2 レジスタ構成

汎用の 32 ビットレジスタを 8本(R0 ～ R7)と、図2.1 に示す専用レジスタ、及び CFGレジスタの計17本から成る。

| | |
|---|---|
| PC | :プログラムカウンタ。 |
| SB | :スタティックデータ領域のベースレジスタ。 |
| FP | :パラメータ、ローカル変数を操作するフレームポインタレジスタ。 |
| SP0,SP1 | :割り込みスタック、及び、ユーザスタックポインタ。 |
| INTBASE | :割り込みディスパッチテーブルアドレス。 |
| MOD | :実行中のモジュールテーブルアドレス。 |
| PSR | :プロセッサステータスレジスタ。 |
| CFG | :コンフィギュレーションレジスタ。 |

### 2.3 モジュラ・プログラミング環境

NS32032 のアーキテクチャ上の特徴はモジュラプログラミング環境をモジュールテーブルと専用レジスタで効率良く実行することである。

・モジュールテーブル

モジュールテーブルはプログラムの各モジュールについて、スタティックデータ領域のアドレス(SB)、リンクテーブルアドレス(LB)、及びモジュールのコードセクションの先頭アドレス(PB)より構成されている。プログラムは MODレジスタに実行するモジュールテーブルアドレスを格納し、その SB，PBエントリを SB,PCレジスタにロードして実行される。

・リンクテーブル

リンクテーブルはモジュール間で変数を共有、あるいは外部プロシージャを起動する際に用いられ、モジュールテーブルの LB エントリで指すリンクテーブルに共有変数の絶対アドレス、あるいは外部プロシージャ記述子（図 2.2）を格納する。

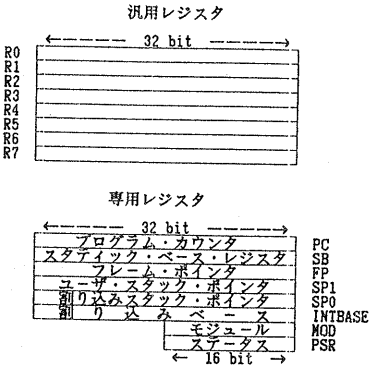


図 2.1 NS32000 CPU のレジスタ構成

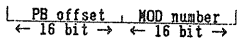


図 2.2 プロシージャ記述子

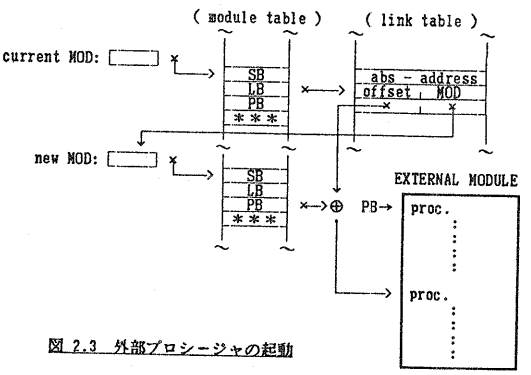


図 2.3 外部プロシージャの起動

外部プロシージャの起動方法を図2.3 に示す。先ず起動元の MOD , PC をスタック上にセーブした後、リンクテーブルより外部プロシージャ記述子を基に起動するモジュール番号を MODレジスタにロードする。モジュールエントリの PB はそのモジュールの先頭コードを示しており、 PB にプロシージャ記述子のオフセットが加えられて外部プロシージャの実行アドレスを決定して起動される。

2.4 周辺チップ

NS32000 シリーズでは CPUの周辺チップとして、割り込みコントロールユニット(ICU:NS32202) 、浮動小数点演算ユニット(FPU:NS32081) を提供している。

・ICU:Interrupt Control Unit :

ICU は 16 レベルの割り込み入力が可能で、カスケード接続すれば最大 256レベルの割り込みに対して優先順位を決定し、CPU に１バイトの割り込みベクタを与えて、マルチレベルの割り込み処理を行なうことが可能である。また、２個の 16 ビットカウンタを内蔵しており、インターバルタイマとして使用できる。

・FPU:Floating Processing Unit :

FPU は８本の 32 ビットレジスタを内蔵し、単精度と倍精度の浮動小数点演算をサポートする。

３．ITRON/32000 の設計仕様

ITRON/32000 は NS32032の特徴であるモジュラプログラミング環境が基本モデルとなる。つまり、ある１つの機能を実現するためのタスク、及びその実行環境を１つのモジュールに対応させることにより、タスク間の実行環境の独立性と機能分割が実現できる。

以下、ITRON/32000 の設計について NS32032の特徴と合わせて述べる。

3.1 ITRON/32000 のシステム構成

ITRON/32000 は NS32032に、タスクの時間管理、マルチレベルの割り込み処理をサポートするために 1 ～ 15 個の割り込みコントロールユニット(ICU) を接続したシステム上で動作する。また、高速な浮動小数点演算を必要とするシステムでは浮動小数点演算ユニット(FPU) の管理を行なう。

3.2 ITRON/32000 の実行環境

NS32032 にはプログラム実行モードとしてNS320-32の特権命令が使用可能なスーパバイザモードと使用不可能なユーザモードがあり、PSR の U( user-mode) ビットによって切り替えられる。また、スタックには割り込みスタックとユーザスタックがあり、PSR の S(stack) ビットによって選択される。割り込みが発生、あるいはトラップ命令が実行されると、PSR の U、及び Sビットがクリアされ、割り込みスタックが選択されてスーパバイザモードでプログラムが実行される。ITRON/32000 ではユーザプログラムはユーザスタック、ユーザモードで実行し、OS自身、割り込みハンドラ及び例外処理ハンドラは割り込みスタック、スーパバイザモードで実行する。尚、スーパバイザモードでは PSRの Sビットを操作して使用するスタックを選択することができる。

ITRON/32000 の実行環境を図 3.1に示す。

| | ユーザモード | スーパバイザモード |
|---|---|---|
| ユーザスタック | ユーザタスク | OS<br>割り込みハンドラ<br>例外処理ハンドラ |
| 割り込みスタック | ——— | |

図 3.1 ITRON/32000 の実行環境

3.3 システムコールの起動

ユーザプログラムで実行されているタスクからのITRON/32000 の呼び出しは、トラップを発生させるSVC 命令によって行なう。 SVC命令が実行されると、実行中のプログラムの CPU状態( PSR ) 、モジュール番号( MOD ) 、及びトラップの発生したアドレス( PC )が割り込みスタック上にセーブされてITRON/32000 が起動される。ここで、トラップの場合、SVC 命令の先頭アドレスがスタック上にセーブされるので、そのアドレスを SVC命令の次の命令アドレスになるように OS 側で補正する。

また、NS32032 のSVC 命令にはマルチレベルのトラップベクタがサポートされていないので、全てのシステムコールを SVC命令によって起動する。従って、割り込みハンドラ、例外処理ハンドラ内で使用するシステムコールの起動、レジスタのリストア操作についてITRON の規定と多少異なる。この点については割り込み処理で述べる。

3.4 タスク・コンテキストの切り替え

ITRON/32000 ではタスクをモジュール単位に管理しているため、タスク・コンテキストの切り替えに際してはモジュールの切り替え操作を伴う。タスク・ディスパッチングが発生すると、汎用レジスタ等のセーブと、プリエンプトされたタスクの PSR レジスタ、MOD レジスタ、及び実行再開アドレスを示すPCレジスタをセーブし、ディスパッチされたタスクのコンテキストをリストアする。

浮動小数点演算ユニットには８本の 32 ビットレジスタと 32 ビットのFSR( FPU Status Register ) がある。ディスパッチされたタスクが浮動小数点演算ユニットを使用する場合には、タスク・コンテキスト切り替え時間の短縮を計るため、その時点での浮動小数点演算ユニット・コンテキストが他タスクのものである場合に限ってのみ、浮動小数点演算ユニット・コンテキストの切り替えを行なう。

3.5 オブジェクトアクセス

ITRON ではユーザが使用するオブジェクトの識別子により、 ID 番号版とアクセスアドレス版とが提

案されている。 ID 番号版では各オブジェクト生成時にユーザがオブジェクトに対して ID 番号を任意(1～65535)に設定し、以後、ID番号を用いてオブジェクトを操作する。一方、アクセスアドレス版では、ITRON/32000 がオブジェクトを管理するために作成した管理テーブルのアドレスを基にオブジェクトを操作する。ITRON/32000 ではオブジェクトの操作を高速に行なうためにアクセスアドレスを使用する。但し、アクセスアドレスはタスクがオブジェクト生成システムコールを実行するまでは決定されないので、オブジェクト生成時にID番号を指定し、アクセスアドレスはその生成システムコールのリターン値として与える。タスク間でオブジェクトを共有する場合には、予め、その ID 番号を共有タスク間で決めておき、ID番号を基にしてアクセスアドレスを得て、オブジェクトの操作を行なうことができる。ITRON/32000 は各オブジェクト毎にID番号からアクセスアドレスを求めるシステムコールを提供する。

### 3.6 システムコール実行中の割り込み

システムコールの起動は SVC命令でトラップを発生させて行なうが、トラップでは PSRの I( interrupt enable )ビットはクリアされないので、ITRON/32000 はシステムコール実行中もできるだけ割り込みの受け付けを許可する。割り込みによってタスク・ディスパッチングが要求された場合は、割り込み処理が終了するまで延期させる。更に、システムコール内でタスク状態を操作していた時は割り込み処理、及び、システムコールの処理が終了した後、ディスパッチする。

## 4. ITRON/32000 の機能

ITRON/32000 は ITRONの規定に従って、 NS32032上にその機能をインプリメントしたものである。 ITRON/32000 のシステムコール一覧表を表 4.1 に示す。

### 4.1 タスク管理

ITRON/32000 ではタスク管理をイベント・ドリブン、プライオリティベースのプリエンプティブ・スケジューリングで行なう。タスクは ID 番号、スタートアドレス等を指定して生成する。ITRON/32000 で生成可

| システムコール名 | | 使用レジスタ | | | | 命令オプション |
|---|---|---|---|---|---|---|
| **1. タスク管理** | | | | | | |
| cre_tsk | Create Task | #R0(=0) | =R1 | R2 | -- *R4 | |
| sta_tsk | Start Task | #R0(=1) | R1 | R2 | -- -- | # --入力およびリターンパラメータに使用 |
| del_tsk | Delete Task | #R0(=2) | R1 | -- | -- -- | |
| def_ext | Define Exit Routine | #R0(=3) | -- | -- | -- R4 | = --リターンパラメータに使用 |
| exi_tsk | Exit Task | R0(=4) | -- | -- | -- -- | |
| exd_tsk | Exit and Delete Task | R0(=5) | -- | -- | -- -- | * --入力パラメータのパケットアドレス |
| abo_tsk | Abort Task | R0(=6) | -- | R2 | -- -- | |
| ter_tsk | Terminate Task | #R0(=7) | R1 | R2 | -- -- | *= --リターンパラメータのパケットアドレス |
| chg_pri | Change Task Priority | #R0(=8) | R1 | R2 | -- -- | |
| rot_rdq | Rotate Ready Queue | #R0(=9) | -- | R2 | -- -- | 無 --入力パラメータ |
| tcb_adr | Get TCB Address | #R0(=10) | =R1 | R2 | -- -- | (=n)--機能コードの値 |
| tsk_sts | Get Task Status | #R0(=11) | R1 | -- | -- *=R4 | |
| sus_tsk | Suspend Task | #R0(=12) | R1 | -- | -- -- | |
| rsm_tsk | Resume Task | #R0(=13) | R1 | -- | -- -- | T_FRCRSM |
| slp_tsk | Sleep Task | #R0(=14) | -- | -- | -- -- | |
| wai_tsk | Wait for Wakeup Task | #R0(=15) | -- | -- | R3 -- | |
| wup_tsk | Wakeup Task | #R0(=16) | R1 | -- | -- -- | |
| can_wup | Cancel Wakeup Task | #R0(=17) | R1 | =R2 | -- -- | |
| cyc_wup | Cyclic Wakeup Task | #R0(=18) | R1 | -- | -- =R4 | T_ABS/T_REL |
| can_cyc | Cancel Cyclic Wakeup Task | #R0(=19) | R1 | -- | -- -- | |
| **2. 同期・通信機能** | | | | | | |
| cre_flg | Create EventFlag | #R0(=20) | =R1 | R2 | -- -- | |
| del_flg | Delete EventFlag | #R0(=21) | R1 | -- | -- -- | |
| set_flg | Set EventFlag | #R0(=22) | R1 | #R2 | -- -- | T_OR,T_AND,T_REP,T_EXOR |
| wai_flg | Wait for EventFlag to be Set | #R0(=23) | R1 | #R2 | R3 -- | T_TMOUT/T_RESET /T_ANDW,T_ORW |
| flg_adr | Get EventFlag AccessAddress | #R0(=24) | =R1 | R2 | -- -- | |
| cre_sem | Create Semaphore | #R0(=25) | =R1 | R2 | R3 -- | T_TPRI,T_TFIFO |
| del_sem | Delete Semaphore | #R0(=26) | R1 | -- | -- -- | |
| sig_sem | Signal Semaphore | #R0(=27) | R1 | R2 | -- -- | |
| wai_sem | Wait on Semaphore | #R0(=28) | R1 | R2 | R3 -- | T_TMOUT |
| sem_adr | Get Semaphore AccessAddress | #R0(=29) | =R1 | R2 | -- -- | |
| cre_mbx | Create Mailbox | #R0(=30) | =R1 | R2 | -- -- | T_TPRI,T_TFIFO /T_MPRI,T_MFIFO |
| del_mbx | Delete Mailbox | #R0(=31) | R1 | -- | -- -- | |
| snd_msg | Send Message | #R0(=32) | R1 | -- | -- R4 | |
| rcv_msg | Receive Message | #R0(=33) | R1 | -- | R3 =R4 | T_TMOUT |
| mbx_adr | Get Mailbox AccessAddress | #R0(=34) | =R1 | R2 | -- -- | |
| **3. 割り込み処理** | | | | | | |
| def_int | Define Interrupt Handler | #R0(=35) | -- | R2 | -- R4 | |
| ret_int | Return from Interrupt Handler | R0(=36) | -- | -- | -- -- | |
| dis_int | Disable Interrupt | #R0(=37) | -- | R2 | -- -- | |
| ena_int | Enable Interrupt | #R0(=38) | -- | R2 | -- -- | |
| get_vec | Get vector No | #R0(=40) | -- | R2 | -- -- | |
| iret_tsk | ret_int & wup_tsk | #R0(=56) | -- | -- | -- -- | *:チップ核複合システムコール |

※ ret_int, iret_tsk も一般のシステムコールと同様に、機能コードをＯＳに渡すために R0 レジスタを使用する。

| | | | | | | |
|---|---|---|---|---|---|---|
| **4. 例外処理** | | | | | | |
| def_exc | Define Exception Handler | #R0(=41) | -- | -- | -- R4 | T_CPU,T_OS/T_TEXC,T_CEXC |
| ret_exc | Return from Exeption Handler | R0(=42) | -- | -- | -- -- | |

※ ret_exc も一般のシステムコールと同様に、機能コードをＯＳに渡すために R0 レジスタを使用する。

| | | | | | | |
|---|---|---|---|---|---|---|
| **5. メモリ管理** | | | | | | |
| cre_mpl | Create Memory Pool | #R0(=43) | =R1 | R2 | -- R4 | T_TPRI,T_TFIFO /T_VARSZ,T_FIXSZ |
| del_mlp | Delete Memory Pool | #R0(=44) | R1 | -- | -- -- | |
| get_blk | Get Memory Block | #R0(=45) | R1 | R2 | R3 =R4 | T_TMOUT |
| rel_blk | Release Memory Block | #R0(=46) | -- | -- | -- R4 | |
| mpl_adr | Get MemoryPool AccessAddress | #R0(=47) | =R1 | R2 | -- -- | |

※ rel_blk では a_mpl のパラメータを使用しない。

| | | | | | | |
|---|---|---|---|---|---|---|
| **6. 時間管理** | | | | | | |
| set_tim | Set Time | #R0(=48) | -- | R2 | R3 -- | |
| get_tim | Get Time | #R0(=49) | -- | =R2 | =R3 -- | |

表 4.1 ITRON/32000 システムコール一覧表 ～ アセンブラ・インタフェース ～

能なタスクの上限は、MOD レジスタが 16 ビット・ポインタで１モジュールエントリに 16 バイト使用するため、１モジュール１タスクの場合、モジュールテーブルエントリ数と同じ４K個である。ITRON/32000 では、システムコールについて自タスクに対する操作、他タスクに対する操作を厳しく規定している。また、動的に変化するタスクの実行状態を調べる機能や、タスクの実行遅延、周期起床といったタスクの時間管理機能を提供する。

4.2 同期通信

同期通信機能として、イベントフラグ、セマフォ、メイルボックスを提供する。セマフォ、メイルボックスでは同期通信待ちタスクを管理する方法としてFIFOまたはプライオリティを指定できる。

4.3 例外処理

ITRON/32000 では CPU例外とシステムコール例外について各タスク毎に例外処理ハンドラを定義することができる。例外処理ハンドラを定義していないタスクに対しては全タスク共通の例外処理ハンドラが起動される。デフォルトの例外処理ハンドラの機能はシステム生成時に

1 : 何もしない。
2 : 例外発生タスクを強制終了する。
3 : デバッガを起動する。

のいずれかを指定する。

4.4 割り込み処理

ITRON/32000 は割り込みコントロールユニット(ICU) を接続して優先順位に基づいたマルチレベルの割り込みをサポートする。

ICU のベクタ送出機能により、割り込み要因を割り込みハンドラ、割り込みタスクと１対１に対応付けることが可能である。従って、ITRON/32000 は def_int システムコールで割り込みハンドラと共に割り込みタスクを定義可能とし、チップ核複合システムコール：iret_tskを提供することによって、割り込みハンドラからの復帰、及び、定義した割り込みタスクの起動を高速に行なえるようにする。また、ユーザが割り込みハンドラから容易に割り込みを制御できるように、ICU のコントロールレジスタから現在処理中の割り込みベクタ番号の取得、任意割り込みレベルのマスク、アン・マスクを行なうシステムコールを提供する。

4.5 メモリ管理

ITRON/32000 は 32 ビット・リニアアドレス空間(4G)のダイナミック・メモリ・アロケーション機能をサポートする。メモリブロックはファーストフィットアルゴリズムに従って、メモリプール毎に決められた固定長のブロック単位で可変長の割り当てを行なう。。フラグメンテーションを防止する機能として、隣接するフリースペースのマージを行なう。尚、NS32032 では 24 ビットしかアドレスがインプリメントされていないので、アドレス上位 8ビットは 0でなければならない。

4.6 タイマ管理

ITRON/32000 は、 48 ビットのシステムタイマを内蔵する。タイマの更新はシステム生成時に指定した ICUのカウンタ値単位で行ない、タイマからの時刻の読み出し、設定を行なうシステムコールを提供する。

５．アセンブラ・インタフェース

アセンブリ言語で記述したユーザプログラムとOSとのインタフェースについて述べる。

5.1 パラメータの受渡し

ユーザプログラムをアセンブリ言語で記述した場合は、ITRON/32000 が提供する全てのシステムコールをSVC 命令で起動する。従って、ユーザはシステムコールの機能を選択するために OS に対して起動する機能、及びパラメータを通知する必要がある。ITRON/32000 では OS 内部でのアクセスを高速に行なうためにパラメータの受渡しに R0 ～ R4 レジスタを使用する。各レジスタの割り当てを以下のように規定する。

・起動するシステムコールの機能選択は R0 で指定する。システムコールにオプションが設定されている場合は、 R0 のビット8 ～ 15 で指定する。
・ OS からのリターンコードは R0 にエラーコードが格納されて戻る。
・オブジェクトのアクセスはアクセスアドレスを用いて高速に行なうため、 R1 でアクセスアドレスを指定する。アクセスアドレスはオブジェクト生成時にR1に格納されてユーザに渡される。
・オブジェクト生成、アクセスアドレスを得る場合には、オブジェクト ID 番号を R2 で指定する。
・パラメータが多いシステムコール（タスク生成等）ではパラメータパケットを作成し、 R4 でその先頭アドレス（32ビットアドレス）を指定する。

尚、ITRON/32000 でアドレスを指定するためにレジスタを使用する場合は32ビットアドレスとして扱うため、24ビットしかアドレスのない NS32032では上位 8ビットは必ず 0でなければならない。

ITRON/32000 におけるパラメータレジスタの割り当てを表 5.1にまとめる。

R0 ···· 機能コードと命令オプション、及びエラーコード（戻り値）
R1 ···· アクセスアドレス
R2 ···· ID番号、オブジェクト操作に使用（入力、あるいは操作により得られるデータ（戻り値）
R3 ···· タイムアウト指定、セマフォ初期値
R4 ···· パケットアドレス、ハンドラ等のアドレス

表 5.1 ITRON/32000 におけるパラメータレジスタの割り当て

5.2 スタートアドレスの指定

タスクの生成、割り込みハンドラ、例外処理ハン

ドラ、及び終了時処理ルーチンの定義に際しては、そのプロシージャのスタートアドレスをシステムコールのパラメータとして OS に渡す。NS32032 はプログラムをモジュール単位で実行するため、ITRON/32000 はタスクをモジュール単位で管理する。つまり、3.4 で述べたようにモジュールレジスタ(MOD)を切り替えてタスクを実行するのでタスクプロシージャのモジュール番号を OS に渡さなければならない。また、NS32032 では割り込みハンドラのスタートアドレスは、割り込みベクタに対応する割り込みディスパッチテーブルエントリにそのプロシージャ記述子（図 2.2）を格納するようになっている。

そこで、ITRON/32000 でスタートアドレスを指定する場合は、そのプロシージャ記述子を指定することにした。そのためには、タスク、割り込みハンドラ、例外処理ハンドラ、及び終了時処理ルーチンはそのプロシージャ記述子を要求元タスクのリンクテーブル内に生成する必要があるので、これらのプロシージャを要求元タスク内で外部プロシージャとして定義していなければならない。そして、要求元タスクは LXPD 命令によってリンクテーブルからプロシージャ記述子を得て、スタートアドレスとして OSに渡す。

```
            .IMPORTP   task
start_adr:  .BLKD
            LXPD       task , satrt_adr
```

図5.1 スタートアドレスの指定

## 6．例外処理

ITRON/32000 は CPU例外、システムコール例外について各タスク毎に例外処理ハンドラが定義可能である。表 6.1はITRON/32000 が扱う CPU例外である。ITRON では、タスク実行中に発生した例外はタスクの一部として実行し、非タスク部分実行中に発生した例外は非タスクとして実行できるように実行モードを区別するオプションが設けられている。 CPU例外が検出されるとトラップが発生し、プログラム実行モードがスーパバイザモードに切り替えられ、割り込みスタックが選択される。また、システムコール例外は OS 内部で検出されるため、その時の実行モードもスーパバイザモードである。例外が発生したときの実行モード（タスク、非タスク）により起動する例外処理ハンドラを区別すると、タスク実行中に発生した例外処理のために割り込みスタックからユーザスタックへのパラメータの積み替えが必要になる。従って、ITRON/32000 では例外処理ハンドラの実行はスーパバイザモードで行ない、オプション(T_NEXEC) はインプリメントしない。

## 7．割り込み処理

ITRON/32000 では割り込みコントロールユニット ICU NS32202 を使用することを前提としており、この結果、マルチレベルの割り込みに対処し、ICU のベクタ送出機能により割り込み要因と起動する割り込みハンドラとを１対１に関係付けることができる。OSは ICUの制御レジスタを操作して割り込み状態を判断、制御するためにも ICUを使用する。割り込み処理は ICUを Fixed Priority Modeに設定して優先順位に従って行なう。また、割り込みハンドラ、システムコール実行中も極力、割り込み受け付け可能にして多重割り込みを許可した。

以下、ICU を使用したITRON/32000 に於ける割り込み処理について述べる。

### 7.1 割り込みの優先順位

ITRON/32000 は ICUを Fixed Priority Modeに設定して優先順位に基づいた割り込み処理を行なう。このモードでは ICUは受け付けた割り込みレベルに対応する ISRV (Interrupt In Service)レジスタのビットをセットして、その優先度以下の割り込みを禁止し、それより高い優先度の割り込みを受け付ける。つまり、ITRON/32000 では高優先度の割り込みのネスティングを許すが、カスケードした時にマスタICU の同一 IR レベルのスレーブ割り込みはマスタICU によって禁止される。従って、割り込みのネスティングは最大 16 レベルである。ユーザがSpecial Mask Modeでマスタ ICU の ISRV を操作すればスレーブ割り込みを受け付けることができるが、ITRON/32000 では割り込みハンドラの起動に OS が関与しないのでサポートできない。

ICU には 'AUTOMATIC RETURN FROM INTERRUPT ' 機能があり、割り込みハンドラが RETI 命令を実行すると、CPU は ICUに対して割り込み処理終了(End Of Interrupt) を通知してペンディングされていた割り込み要求の最優先度の割り込み要求が受け付けられる。

### 7.2 割り込みベクタ

INTBASE レジスタで指される割り込みディスパッチテーブルには 0～255 ベクタの割り込みハンドラを登録でき、NS32032 でベクタ割り込みを行なうと最大 256レベルの割り込みに対処できる。しかし、0H～0FH までは NS32032の固定トラップ、割り込みに使用するため、ITRON/32000 では 10H～FFH までの 240レベルの割り込みを受け付け可能にする。割り込みベクタの割り当てはシステム初期化時にユーザタスクでアセンブリ言語を用いて直接、指定することができるが、ここでは ITRON/32000標準の割り込みベクタ値について述べる。ICU に与える割り込みベクタはカスケード接続することを考慮して CPU

| CPU 例外コード | 例外の種類 |
|---|---|
| TE_FPU | FPU 命令の例外 |
| TE_ILL | 違法オペレーション |
| TE_DVZ | 整数０による除算 |
| TE_FLG | PSR の Fビットがセットされている時に FLAG 命令を実行 |
| TE_UND | 未定義命令実行 |

表 6.1 CPU 例外コード

に直接接続するマスタICU を 10Hとし、マスタICU の割り込み入力端子 IR0をカスケード禁止し、IR1 ～IR15にカスケードするスレーブICU に10H ～FFH を与える（図 7.1）。但し、マスタICU の IR1にカスケード接続するとスレーブICU の割り込みベクタが 10Hとなるため、マスタICU の IR1は IR1～IR15 を全てカスケード接続する場合のみ、カスケード可能とする。表 7.1に割り込みベクタテーブルを示す。

7.3 割り込みハンドラの定義、起動

割り込みハンドラはユーザが作成し、任意の割り込みベクタに対して def_intシステムコールで定義する。def_int システムコールはプロシージャ記述子で指定された割り込みハンドラのスタートアドレスをINTBASE で指される割り込みディスパッチテーブルの該当ベクタのエントリに登録する。割り込みが発生すると CPUは被割り込み処理の PSR,MOD,PC を割り込みスタック上にセーブして、ICU から受け取った割り込みベクタを基に割り込みディスパッチテーブルから得た割り込みハンドラに実行を移す（図7.2 ）。NS32032 は割り込みを受け付けると PSR の Iビットをクリアして割り込み禁止にする。従って、割り込みハンドラ内で PSRの Iビットをセットしない限り、割り込みハンドラは割り込み禁止状態で実行される。

7.4 クロック割り込み

システムタイマの更新、タスクの時間管理に使用するクロックは、ICU に内蔵されている16bit zero-detective counterがインターバル時間毎に割り込みを発生させて CPUにクロック更新を要求する。クロック割り込みが発生すると ITRON/32000が提供するクロック割り込みハンドラが起動されてクロック割り込み処理が行なわれる。従って、クロック割り込みも他の割り込みと同様に割り込みベクタを与え、クロック割り込みハンドラを割り込みディスパッチテーブルに登録する。インターバル時間とクロック割り込みベクタはシステム初期化時にユーザが設定する。割り込みハンドラの起動に対して OS は関与しないため、クロック割り込みを優先させることはしない。つまり、クロック割り込みに低い割り込み優先順位を割り当てると他の割り込み処理の負荷によってクロック分解能が低下する。

7.5 割り込みハンドラの記述

ITRON/32000 は割り込みハンドラの起動に関与しないので割り込みハンドラで使用するレジスタのセーブ、リストア、及び割り込みハンドラ実行中の割り込みの許可／禁止はユーザの責任である。ITRON/32000 は割り込みハンドラの記述、及び、割り込みタスクの起動に関して以下のように規定する。

- 割り込みハンドラ内で使用するレジスタは割り込みハンドラの入り口でセーブし、出口でリストアする。
- 割り込みハンドラを割り込み禁止状態で実行し、且つ、システムコールを発行しなければ RETI アセンブラ命令で終了してよい。
- 割り込み可能状態で実行、あるいはシステムコールを発行した割り込みハンドラの出口では遅延されているタスクディスパッチを起動するため、必ず ret_int ,iret_tskシステムコールを発行しなければならない。
- 割り込みハンドラ内では、割り込みタスク起動に使用できるように、自タスクを指定可能なシステムコール、及び、オブジェクトを生成、削除するような処理時間の長いシステムコール以外は全て使用可能とする。但し、割り込みタスクのアクセスアドレスを取得するため、自タスクを指定可能

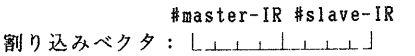


図 7.1 割り込みベクタ（１バイト）

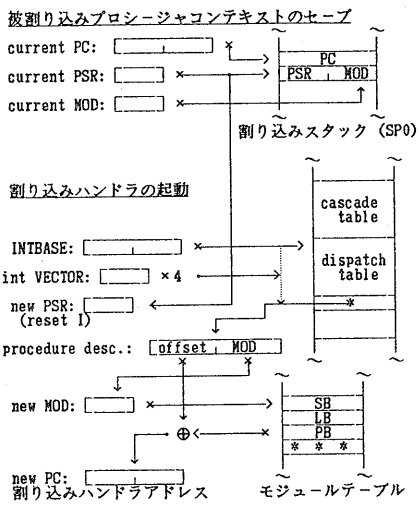


図 7.2 割り込みハンドラの起動

| vector | usage for trap / interrupt vector |
|---|---|
| 00H<br>:<br>0FH | SYSTEM TRAP/INTERRUPT VECTOR |
| 10H<br>:<br>1FH | non-cascade / slave(master IR1)<br>interrupt vector |
| 20H<br>:<br>2FH | slave(master IR2) interrupt vector |
| :<br>: | : |
| F0H<br>:<br>FFH | slave(master IR15) interrupt vector |

表 7.1 割り込みベクタテーブル

なtcb_adr システムコールは使用可能にする。
・割り込みタスクの起動を高速に行なうため、ret_int + wup_tsk の機能を合わせたチップ核複合システムコール：iret_tskを提供する。

割り込みハンドラの記述例を図 7.3に示す。

## 7.6 タスクディスパッチングの遅延

割り込み処理中はそれ以下の優先順位の割り込みはペンディングされるため、時間のかかる処理は割り込みタスクに任せられるように割り込みハンドラ内でもタスク起動等のシステムコールを使用可能にした。システムコールを発行して割り込みタスクがディスパッチされると、図 7.4.aのように割り込みハンドラの割り込みレベル以下の割り込みを禁止したまま実行され、応答性が低下する。そのため、割り込みハンドラ内から起動された割り込みタスクのディスパッチングは全ての割り込み処理が終了して割り込み禁止状態を解除するまで遅延させる（図 7.4.b）。システムコール発行元が割り込みハンドラかユーザタスクかは、OSが ICUの ISRV(Interrupt In Service) レジスタを調べることにより判定することができる。また、被割り込み処理がシステムコール実行中でタスクの状態を操作していた場合は全割り込み処理が終了した後、被割り込み処理を実行してタスク状態が決定するまでタスクディスパッチを遅延させる。

## 7.7 ret_int,iret_tskシステムコールのインプリメンテーション

ret_int,iret_tskの起動時にはレジスタをリストアしているので R0 レジスタで機能コードを指定すると被割り込み処理の R0 の内容が破壊される。ITRON ではこれらのシステムコールについては機能コードを使用せず、専用のトラップベクタを割り当て、他のシステムコールとの起動経路を区別するようにしている。しかし、NS32032 のトラップ命令にはマルチベクタがサポートされていないため、専用のトラップベクタを割り当てることはできない。従って、ret_int,iret_tskの起動に際して、ITRON/32000 では全てのシステムコールを SVC命令で起動するため、OSに機能コードを渡す R0 レジスタはシステムコール起動前にユーザが割り込みスタック上にセーブし、OS側でリストアするようにした。例外処理ハンドラから復帰する ret_excシステムコールについても同様な起動方法をとる。ITRON/32000 が ITRON の規定と異なる点である。

## 8．高級言語とのインタフェース

ITRON の想定する高級言語はＣ言語である。Ｃ言語で記述したタスクからのシステムコールの起動は外部ファンクションを呼び出す形式で行ない、パラメータはスタック上に右から左の方向に従って格納される。ITRON/32000 のパラメータ受渡しにはレジスタを使用するため、使用するレジスタをセーブしてスタック上のパラメータをレジスタに格納し、トラップを発生させて OS を起動するインタフェースライブラリを設けて、Ｃ言語とアセンブリ言語でのOS呼び出し形式のインタフェースをとる。

パラメータで問題になるのはタスク、割り込みハンドラ、例外処理ハンドラ等のスタートアドレスの指定である。 ITRON/32000ではスタートアドレスの指定は 5.2で示した外部プロシージャ記述子で行なう。アセンブリ言語では、 'IMPORTP（ Ｃ言語の extern ）宣言 'した外部プロシージャをタスクとし

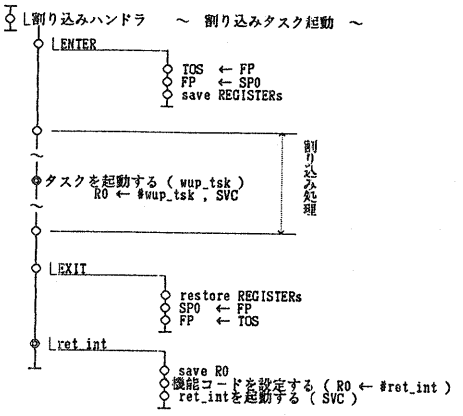


図 7.3 割り込みハンドラ記述例

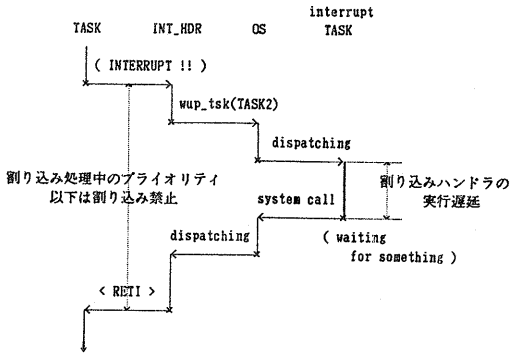


図 7.4.a 割り込みハンドラからのタスク起動（ディスパッチング遅延なし）

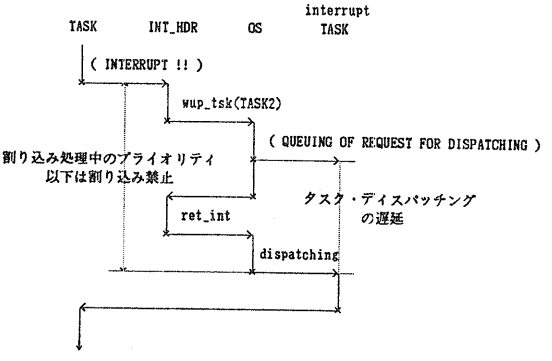


図 7.4.b 割り込みハンドラからのタスク起動（ディスパッチング遅延）

て生成する場合、リンクテーブルを調べればプロシージャ記述子を取得することができた。C言語でも同様に ' extern 宣言 'した外部ファンクションをタスクとして生成するする場合は、'void 型宣言 ' したポインタでタスクのスタートアドレスを得ることができる（図 8.1）。しかし、現在のところ、

```
extern  task( )          ;
void ( *start_adr ) ( ) ;
start_adr = task         ;
```

図 8.1 スタートアドレスの指定

NS32000 シリーズ用Cコンパイラが 'void型宣言 ' したポインタに 'extern宣言 'したファンクションの外部プロシージャ記述子を返すようになっているかどうか確認していない。また、タスクを同一モジュール内のローカル・ファンクションとして定義していた場合、アセンブリ言語では 'IMPORTP 宣言 ' すればその外部プロシージャ記述子がリンクテーブルに登録されるので指定可能である。C言語で図 8.1のようにローカル・ファンクションに対して'extern宣言 'した時に同様な結果が得られるかは確認する必要がある。

## 9．おわりに

NS32032 のアーキテクチャ上の特徴は、専用レジスタとモジュールテーブルでモジュラプログラミング環境を制御することであり、ITRON/32000 はその特徴を取り入れたモジュール単位のタスク管理を行なうことである。従って、システムコールのパラメータの指定に於いて、タスク等のスタートアドレスはモジュール番号とその先頭コードからのオフセットから成るプロシージャ記述子を使用することにした。

また、ITRON の仕様との相違点は、NS32032 の SVC 命令がマルチトラップベクタをサポートしていない関係で、割り込み、例外処理ハンドラで使用するシステムコールの起動にも機能コードを必要とする。そのため、機能コードを指定する R0 レジスタのリストアを OS 側で行なうようにした。

高級言語との整合性を考慮するためには NS32000 シリーズ用Cコンパイラの調査、検討が必要である。

最後に、ITRON に関して御指導頂いた東京大学の坂村先生、並びにＬＳＩ研究所の関係者各位に深く感謝致します。

［参考文献］

・National Semiconductor Corp.「NS32000 データブック」
・National Semiconductor Corp.「Series 32000命令セット・リファレンス・マニュアル」
・National Semiconductor Corp.「NS16000 Cross-Assembler Reference Manual」
・坂村　健「リアルタイムオペレーティングシステム－ＩＴＲＯＮ」
情報処理学会第24回オペレーティングシステム研究会
・坂村　健「ＩＴＲＯＮアーキテクチャ」
「ＩＴＲＯＮ／68K 」
「ＩＴＲＯＮ／86 」
情報処理学会第29回（昭和59年後期）全国大会
・長谷川、高橋、門田「Ｖ20／Ｖ30用リアルタイムＯＳのインプリメンテーション（１）、（２）」
情報処理学会第29回（昭和59年後期）全国大会