μITRON4.0 仕様 マルチコアプロセッサ拡張 ガイドライン
(μITRON/AMP ガイドライン)
Ver. 1.00.00

2012 年 3 月
T-Engine フォーラム
http://www.t-engine.org/

μITRON4.0 仕様 マルチコアプロセッサ拡張ガイドライン
Ver.1.00.00

本ガイドラインに関しては、下記にお問い合わせください。

T-Engine フォーラム事務局
〒141-0031 東京都品川区西五反田2-20-1 第28 興和ビル
YRP ユビキタス・ネットワーキング研究所内
TEL：03-5437-0572 FAX：03-5437-2399
E-mail：office@t-engine.org

目次

第 1 章 背景 1

1.1 マルチコアプロセッサ拡張の目的 1
1.2 方針・目標 1

第 2 章 共通規定 2

2.1 想定するシステム 2
2.1.1 プロセッサとメモリ(構成) 2
2.1.2 プロセッサとμITRON4.0 3
2.2 オブジェクトの ID 番号とオブジェクト番号 3
2.2.1 コア割付けとコア番号 3
2.3 優先度 4
2.4 サービスコールの返値とエラーコード 4
2.5 タイムアウト 4
2.6 システム時刻 4
2.6.1 システム時刻の同期について 4
2.7 システムコンフィギュレーションファイル 4

第 3 章 概念と共通定義 5

3.1 タスク状態とスケジューリング規則 5
3.1.1 タスクの状態遷移と過渡的状態について 5
3.1.2 タスクのスケジューリング規則 5
3.2 割込み処理モデル 5
3.2.1 割込みハンドラと割込みサービスルーチン 5
3.2.2 割込みの指定方法と割込みサービスルーチンの起動 6
3.3 例外処理モデル 6
3.3.1 例外処理の枠組み 6
3.3.2 CPU 例外ハンドラで行える操作 6
3.4 CPU ロック状態 / ディスパッチ禁止状態 6
3.4.1 CPU ロック状態 6
3.4.2 ディスパッチ禁止状態 6
3.5 オブジェクト 7
3.5.1 オブジェクトの動的生成について 7
3.6 システム初期化処理 7
3.6.1 初期化処理の同期について 7

第 4 章 付録 8

4.1 マルチコアプロセッサ サポート機能の拡張 8
4.2 謝辞 8

第1章 背景

1.1 マルチコアプロセッサ拡張の目的

携帯電話やデジタルテレビ、車載システムなど各種組込み機器も高性能化、高機能化が進展している。一方、これらの機器に内蔵されている先端マイクロコンピュータやシステムチップには微細構造の物理的限界から生じる各種の問題が顕在化しつつある。特に大きな問題は消費電力であり、性能を向上させるために従来行われてきた微細加工により周波数を上げると消費電力が飛躍的に増大し、実用上大きな問題となっている。この問題に対応するために、CPU コアを複数個搭載し並列動作させることで、周波数を上げることなく性能を向上させる「マルチコアプロセッサ」がパソコンやサーバで実用化され普及している。組込みシステムにおいてもこの「マルチコアプロセッサ」が検討・実用化され始めている。
組込みシステムにおいて「マルチコアプロセッサ」が検討・実用化され始めると同時に、これまで利用されてきた μITRON4.0 仕様のリアルタイムカーネルを、「マルチコアプロセッサ」上で動作するよう機能拡張が行われ始めた。しかしその拡張された仕様は、各社独自に行われておりマルチコアプロセッサに対応したμITRON 仕様のリアルタイムカーネル上で動作するソフトウェアの移植性・流通性の障壁となる。
そこで、ソフトウェアの移植性・流通性を確保することを目的として、μITRON4.0 仕様のリアルタイムカーネルに対してマルチコアプロセッサ拡張機能を追加する際のガイドラインを策定することとした。

1.2 方針・目標

μITRON4.0 仕様のリアルタイムカーネルに対してマルチコアプロセッサ拡張機能を追加する際のガイドラインを策定するにあたり、以下の方針・目標を設定した。
なお、以降μITRON4.0 仕様及び本ガイドライン記載の推奨仕様をμITRON4.0/AMP と呼び、その仕様に沿ったリアルタイムカーネルをμITRON4.0/AMP OS と呼ぶこととする。

a) μITRON4.0 仕様との互換性保持
本ガイドラインに記述しているμITRON4.0/AMP は、個々のコア上でμITRON4.0 仕様に準拠したリアルタイムカーネルが動作することを基本とし、μITRON4.0 仕様で記述されたソフトウェアを最小限の変更で、マルチコアプロセッサ上で動作させることができるようにすることを目指す。
μITRON4.0 仕様のサービスコールをそのままコア間でも使用することとし、新たな API 追加は行わない。
また、μITRON4.0/AMP とμITRON4.0 仕様 保護機能拡張を同一システムへ実装することも考えられることから、実装時に齟齬が生じないよう考慮する。

b) 機能分散型非対称ソフトウェアモデルが対象
マルチコアプロセッサはソフトウェアの実行モデルから非対称型(AMP:Asymmetric Multi Processing)と対称型(SMP:Symmetric Multi Processing)に大別される。AMP はそれぞれのコアに役割が静的に定められており、プログラムが動作するコアは予め決められている。SMP はコアの役割は決められておらず、プログラムが動作するコアは OS により動的に決められる。このように AMP と SMP ではカーネルの機能・実装が大きく異なる。
そこでITRONがリアルタイム制御を対象としており、リアルタイム性の確保が容易なこと、従来のμITRON4.0 仕様で記述されたアプリケーションをマルチコアプロセッサ用に移行する際のハードルが低いことなどから、コアの役割が静的に定められている AMP を対象とすることとした。
なお、本ガイドラインでは動的な負荷分散機能(タスク処理の実行コア変更機能)は定義しないものとする。

c) μITRON4.0 仕様からの差分仕様として作成
μITRON4.0/AMP は、個々のコア上でμITRON4.0 仕様に準拠しているリアルタイムカーネルが動作することを基本としている。そのため、本ガイドラインには、μITRON4.0 仕様と差分が生じた部分のみ記述を行うこととする。

第2章 共通規定

2.1 想定するシステム

2.1.1 プロセッサとメモリ(構成)

μITRON4.0 仕様マルチコアプロセッサ拡張ガイドラインを記述するにあたり、想定したハードウェア構成を以下に記述する。

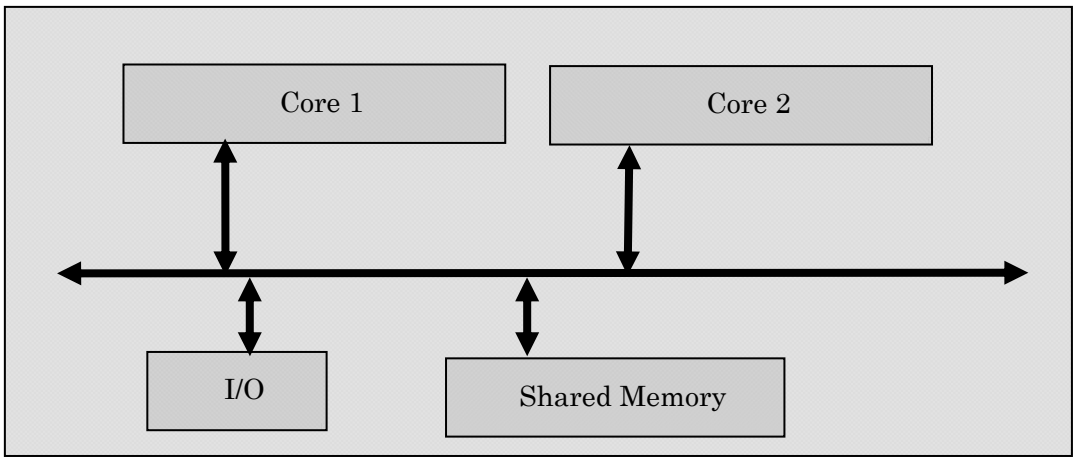

図 2-1 想定システム図

a) コア構成
システムを構成する各コアが同種(ホモジニアス)である場合、異種(ヘテロジニアス)を含む場合のいずれでもよく、本ガイドラインに記したμITRON4.0/AMP OS が個々のコア上で動作するための環境が整っていればよい。

b) メモリ構成
メモリ構成は、システムを構成する全てのコアからアクセス可能なメモリ領域(共有メモリ)を必須とする。各コアはこの共有メモリを分断して排他的に使用する。
なお、各コアに固有のメモリ領域については必ずしも必要ではない。
複数のコアから同一カーネルオブジェクトを操作する際にコア間の排他機能が必要となる場合があるが、この排他機能の具体的実現手段について、本ガイドラインでは言及しない。

c) コア間通信機能
あるコアと他のコアとのデータの受け渡しや、事象発生の通知が可能なコア間通信機能を有する。
コア間通信機能の具体的実現手段について、本ガイドラインでは言及しない。

【補足事項】
μITRON4.0 仕様では割込み管理機能が規定されているが、マルチコアプロセッサ拡張においてこの規定に変更はなく、シングルコアプロセッサの場合と同様に、プロセッサの割込みアーキテクチャや IRC (割込みコントローラ)に依存するため、実装依存とする。
ただし、割込みが複数のコアに入力される場合や、複数のコアで1つの IRC を操作する場合などカーネル内または、アプリケーションでの対応が必要な場合があることに注意が必要である。

2.1.2 プロセッサとμITRON4.0

システムを構成するコア毎に1つのμITRON4.0/AMP OS が動作する。

2.2 オブジェクトの ID 番号とオブジェクト番号

【マルチコア拡張】
μITRON4.0/AMP OS 上のオブジェクトの ID 番号は、システム全体で正の値を用いることを推奨する。また、オブジェクトをユーザオブジェクトとシステムオブジェクトに分類する場合、ユーザオブジェクトにはシステム全体で正の ID 番号、システムオブジェクトにはシステム全体で(-5)から小さい方の負の ID 番号を用いることを推奨する。

2.2.1 コア割付けとコア番号

μITRON4.0/AMP OS が動作する複数のコアは、ID 番号により識別する。コアを識別する ID 番号には、1から連続した正の値を用いる。システムの起動時に特定のコアが動作する場合は、そのコアの ID 番号を1とし、その他のコアへの ID 番号は実装定義とする。
μITRON4.0/AMP OS 上のオブジェクトはいずれかのコアに属しており、生成時にどのコアに属しているかを指定する必要がある。

オブジェクトが属しているコアの指定方法として、次の方法を推奨する。
カーネルオブジェクトを静的に生成する場合、コンフィギュレーションファイルに次の記述を行うことにより、それぞれのコアにカーネルオブジェクトを割付ける。

```
define_core <コアの ID 番号> {
    当該コアに属するカーネルオブジェクトの登録など
};
```

また、カーネルオブジェクトを動的に生成するサービスコール(cre_yyy / acre_yyy)で生成されたカーネルオブジェクトは、オブジェクト生成サービスコールを発行(=実行)したコアに属するものとする。なお、他のコアにオブジェクトを生成するサービスコールを実装独自に追加してもよい。

上記の推奨する仕様のほかに、次の2つの方法によるコア ID の指定も許される。

a) オブジェクトが属しているコアの指定方法として、オブジェクト生成用サービスコール(CRE_YYY/ cre_yyy / acre_yyy)のオブジェクト ID 指定パラメータの上位ビットを使用してコア ID 番号の指定を行う方法。本方法を使用する場合において、コア ID 番号を指定するビット幅については実装依存である。
b) オブジェクトが属しているコアの指定方法として、オブジェクト生成用サービスコール(CRE_YYY / cre_yyy / acre_yyy)のパラメータにオブジェクトが属するコアの ID 番号を追加する方法。本方法を使用する場合において、コアの ID 番号を指定するパラメータの型は ID 型とする。

【補足説明】
上記方法で生成されたカーネルオブジェクトを利用する際には、従来通りID番号を指定して利用する(各サービスコールのパラメータに変更はない)。各オブジェクトが属しているコア ID の指定は必要ない。これは、本マルチコア拡張においても、カーネルオブジェクトに割付けられる ID 番号はオブジェクトの種類毎にシステム全体でユニークであるためである(例えば、ID 番号が1のセマフォはシステム全体で1つしか存在せず、ID が1のセマフォはいずれのコアからも同じセマフォ・オブジェクトを示す)。

2.3 優先度

【マルチコア拡張】
μITRON4.0/AMP OS では、タスクやメッセージなどの優先度は、1から連続した正の値を用い、コア毎に値が小さいほど優先して処理されることを推奨する。

【補足説明】
異なるコアに属している複数のタスクが、あるコアに属しているカーネルオブジェクト(オブジェクトの属性にTA_TPRI(タスク優先度順の待ち行列)の指定あり)に対して資源の獲得要求を行い、いずれのタスクも資源が獲得できずに待ち状態になった場合、同一の待ち行列にそれぞれのタスク優先度に従ってつながれることになる。この場合、タスク優先度として同一値を保持しているタスクは同一優先度を持つものとして扱われることに注意が必要である。

2.4 サービスコールの返値とエラーコード

【マルチコア拡張】
新たに本ガイドライン内でエラー検出を推奨する場合に、どのエラーを返すかは実装定義とする。

2.5 タイムアウト

【マルチコア拡張】
μITRON4.0/AMPにおいてタイムアウト時間の測定は、そのサービスコールを呼び出したタスクが属しているμITRON4.0/AMP OS で行われることを推奨する。

2.6 システム時刻

【マルチコア拡張】
時間管理は、各コア上のμITRON4.0/AMP OS 毎に独立して行われる。また経過時間の測定も各μITRON4.0/AMP OS で行われることを推奨する。
カーネル仕様には現在のシステム時刻を設定する機能が用意されているが、指定された時刻が設定されるのは、そのシステムコールを発行(=実行)したμITRON4.0/AMP OS で管理しているシステム時刻のみ(他のコアで管理されているシステム時刻に影響を与えないもの)とする。

2.6.1 システム時刻の同期について

【マルチコア拡張】
システム時刻は各コア上のμITRON4.0/AMP OS の初期化処理において初期化されるが、その初期化処理の同期及び初期化処理以降のシステム時刻の同期については実装定義とする。

2.7 システムコンフィギュレーションファイル

【マルチコア拡張】
カーネルオブジェクトを各コアに割付けるための記述が追加された以外、システムコンフィギュレーションファイルの記述方法やその処理手順に変更はないが、その処理結果として出力されるカーネルの構成や初期化に必要なファイル kernel_cfg.c がコア毎に生成されるかシステムで1つ生成されるかは実装定義である。

第3章　概念と共通定義

3.1　タスク状態とスケジューリング規則

3.1.1 タスクの状態遷移と過渡的状態について

μITRON4.0/AMP におけるタスクの状態遷移はμITRON4.0 仕様が規定する状態以外の過渡的なタスク状態あるいは実装独自に拡張された状態がアプリケーションから観測できることを許容する。μITRON4.0 仕様が規定するタスク状態が拡張される場合、その状態を ref_tsk および ref_tst で参照可能かは実装定義である。
過渡的あるいは実装独自の状態を導入した場合には、その状態がアプリケーションからどのように観測されるか、またその導入によってどのような制約が生じるかを明示しなければならない。
また，μITRON4.0/AMP では，μITRON4.0 仕様の「3.5.3 処理の優先順位とサービスコールの不可分性」に記載の「サービスコールの不可分性」を保証しないことを認める。

【補足説明】
「過渡的あるいは実装独自に拡張されたタスク状態が観測されること」および「サービスコールの不可分性を保証しないこと」を許容した理由は次の通りである。
サービスコールを別コアのカーネルに依頼する実装方式の場合、サービスコールの不可分性を保つには対象コアのカーネルからの応答が帰ってくるまでの間、自コアでは次の処理を実行できないことになり CPU 時間の無駄が大きくなる。サービスコールを呼び出したタスクを、対象コアのカーネルからの応答が帰ってくるまで何らかの待ち状態に遷移させ、その間に別のタスクを実行できるようにすることでこのような無駄を排除できるが、この場合には過渡的な「何らかの待ち状態」がアプリケーションから観測できることになり、サービスコールの不可分性も保証できなくなる。

3.1.2 タスクのスケジューリング規則

μITRON4.0/AMP OS は、コア毎に独立したμITRON4.0 仕様が規定するタスクスケジューリングを行う。つまり、タスクの優先度空間はコアごとに独立している。
シングルコアプロセッサを想定したμITRON4.0 仕様の範囲内では、タスク優先度によって暗黙的にタスク間の実行順序を制御(排他制御)する利用方法があったが、μITRON4.0/AMP OS ではコア毎に優先度空間が独立しているため、コアをまたがるタスク間ではこの方法は利用できない(排他制御できない)。
また、優先度空間はコア毎に独立していることから、ミューテックス機能はコアをまたがって使用することはできない。別コアのミューテックスに対するサービスコールを行った場合の振る舞いは実装定義である。

【補足説明】
別コアのミューテックスに対してサービスコールを呼び出した場合は、エラーを報告することを推奨とする。

3.2　割込み処理モデル

3.2.1 割込みハンドラと割込みサービスルーチン

μITRON4.0/AMP OS の割込み処理モデルはμITRON4.0 仕様と同様に、割込みハンドラと割込みサービスルーチンがあるものとする。割込みハンドラと割込みサービスルーチンの役割はμITRON4.0 仕様と同じである。
外部割込みと実行するコアの対応は、プロセッサの仕様だけではなく、システムの仕様に大きく依存する。このため、μITRON4.0/AMP では規定せず、実装定義とする。
ただし、どのようなハードウェア仕様であったとしても、割込み処理の開始から終了まで(割込みハンドラ、割込みサービスルーチン)は同一のコンテキストとして扱われることを保証する。すなわち、割込みハンドラと割込みサービスルーチンは同一のコアでシーケンシャルに実行され、割込みハンドラから起動された割込みサービスルーチンが、別のコアで(割込みハンドラと並列に)実行されることはない。

なお、割込みハンドラと割込みサービスルーチンは、それが実行されているコアとは別のコアで実行されているプログラムとは無関係に起動される。このため、割込み処理とタスクの間での排他制御が必要になる場合があるが、このような場合に排他制御を行うための機能は実装依存とする。

3.2.2 割込みの指定方法と割込みサービスルーチンの起動

μITRON4.0/AMP OS での割込み番号と割込みハンドラ番号の意味はμITRON4.0 仕様と同じである。
割込み番号と割込みハンドラ番号はそれぞれシステム全体で一意とする方式を推奨仕様とする。
ただし、割込み処理の構成は CPU の仕様に大きく依存するため実装定義とする。

3.3 例外処理モデル

3.3.1 例外処理の枠組み

CPU 例外ハンドラとタスク例外処理の定義は、μITRON4.0 仕様と同様の定義とする。
ただし、CPU 例外ハンドラは、コア毎に発生(検出)される CPU 例外によって起動されるものとし、CPU 例外を検出したコアとは異なる(別の)コアによって CPU 例外ハンドラが実行されることはないものとする。
CPU 例外を検出したコアとは異なるコアによって CPU 例外ハンドラが実行されるハードウェアでの動作はμITRON4.0/AMP では規定せず、実装定義とする。

3.3.2 CPU 例外ハンドラで行える操作

CPU 例外ハンドラで行える操作の定義はμITRON4.0 仕様と同様に実装定義とする。
CPU 例外ハンドラ内で呼び出し可能なサービスコールに要求される条件(下記)も同様である。

(a) CPU 例外が発生したコンテキストや状態の読出し. 具体的には, CPU 例外が発生した処理で sns_yyy を呼び出した場合の結果を, CPU 例外ハンドラ内で取り出せること.
(b) CPU 例外が発生したタスクの ID 番号の読出し(例外を発生させたのがタスクである場合のみ).
(c) タスク例外処理の要求. 具体的には, CPU 例外ハンドラ内で, ras_tex と同等の操作ができること.

ただし、CPU 例外が発生したコアと CPU 例外ハンドラを実行しているコアが同一であることをサービスコール実行可能の前提条件とする。CPU 例外が発生したコアと CPU 例外ハンドラを実行しているコアが同一でない場合は実装定義とし、上記の操作を用意しなくてもよい。

3.4 CPU ロック状態 / ディスパッチ禁止状態

3.4.1 CPU ロック状態

CPU ロック状態は、各コア上のμITRON4.0/AMP OS 毎に独立して管理される。
loc_cpu は、サービスコールを呼び出したコアを CPU ロック状態にし、unl_cpu は、サービスコールを呼び出したコアを CPU ロック解除状態にする。

【補足説明】
CPU ロック状態のコアに属しているカーネルオブジェクトの操作行った場合の動作については実装定義とする。

3.4.2 ディスパッチ禁止状態

ディスパッチ禁止状態は、各コア上のμITRON4.0/AMP OS 毎に独立して管理される。

dis_dsp は、サービスコールを呼び出したコアをディスパッチ禁止状態にし、ena_dsp は、サービスコールを呼び出したコアをディスパッチ許可状態にする。dis_dsp を呼び出したコア以外で動作しているタスクは動作し続ける。

【補足説明】
ディスパッチ禁止状態のコアに属しているカーネルオブジェクトの操作を行った場合の動作については実装定義とする。
また、ディスパッチ禁止状態のコアから他のコアに属しているオブジェクトに対する操作を行った場合の動作についても実装定義とする。

3.5 オブジェクト

3.5.1 オブジェクトの動的生成について

acre_yyy または追加された API によって、オブジェクトの動的生成をサポートする場合、生成されるオブジェクトの ID 番号は、CRE_YYY/cre_yyy で指定する ID 番号の取り扱いと矛盾しない ID 番号を返さなければならない。

3.6 システム初期化処理

3.6.1 初期化処理の同期について

システム起動時において、各コアに割り当てられたμITRON4.0/AMP OS は、それぞれ独立して初期化処理を行うものとし、初期化処理終了のタイミングで他のコアで動作しているμITRON4.0/AMP OS と同期を取るための手段を用意しなければならない。初期化処理の同期を行うかどうかを、アプリケーションによって選択できることが望ましいが、必ず同期を行う実装も許される。
なお、初期化処理終了の同期方法については実装定義とする。

【補足説明】
μITRON4.0/AMP ではカーネルオブジェクトの管理やデバイス、メモリの管理等はコア毎に行われているため、それらの初期化処理においても各コアがそれぞれ独立して自身の管理下のものを初期化し、全体としては非同期で処理が行われる。
しかし、システム起動直後にコア間の同期・通信を行うアプリケーションにとって不都合が生じる場合がある。これは各コアの初期化処理終了タイミングがばらつくことによって、同期・通信先のμITRON4.0/AMP OS が管理するカーネルオブジェクトが未生成のタイミングで同期・通信を開始してしまうことがあるからである。これを防ぐために、μITRON4.0/AMP OS の初期化処理終了タイミングで同期をとる手段を用意することとした。
なお、初期化処理中のシステム時刻を 0 とするタイミングのコア間の同期については、「2.6.1 システム時刻の同期について」を参照のこと。

第4章　付録

4.1　マルチコアプロセッサ サポート機能の拡張

μITRON4.0 仕様書及び本ガイドラインに記載されていない機能を実現するために、マルチコアプロセッサのサポート機能として実装独自にサービスコールを追加する場合には、μITRON4.0 仕様書「5.1.3 μITRON4.0 仕様に対する拡張」記載の内容に準じていることが望ましい。

【補足事項】
マルチコアプロセッサのサポート機能としてコア間排他機能を実現するためのスピンロック機能などが上げられるが、これらをサービスコールとして実装するか、ライブラリ関数または C 言語のマクロとして実現するかは実装依存である。

4.2　謝辞

本μITRON4.0 仕様 マルチコアプロセッサ拡張ガイドラインの検討に参加いただいたメンバーを以下に記載するとともに、そのご協力に対して謝意を表します。

	金子智範	（イーソル株式会社）
	木下稔章	（株式会社デンソークリエイト）
座長：	小林康浩	（富士通セミコンダクター株式会社）
	宮下光明	（株式会社グレープシステム）
	檜原弘樹	（NEC 東芝スペースシステム株式会社）
	西林浩司	（ルネサス エレクトロニクス株式会社）
	高倉規彰	（ルネサス エレクトロニクス株式会社）
	山田真二郎	（ルネサス エレクトロニクス株式会社）
	由良修二	（YRPユビキタス・ネットワーキング研究所）

（会社名アルファベット順、敬称略）