μITRON4.0 仕様の徹底解説
パート1
「ITRON 仕様 共通規定と静的 API」

富士通デバイス株式会社
稲光一豊

パート 1

● μITRON4.0 仕様共通規定

●静的 API

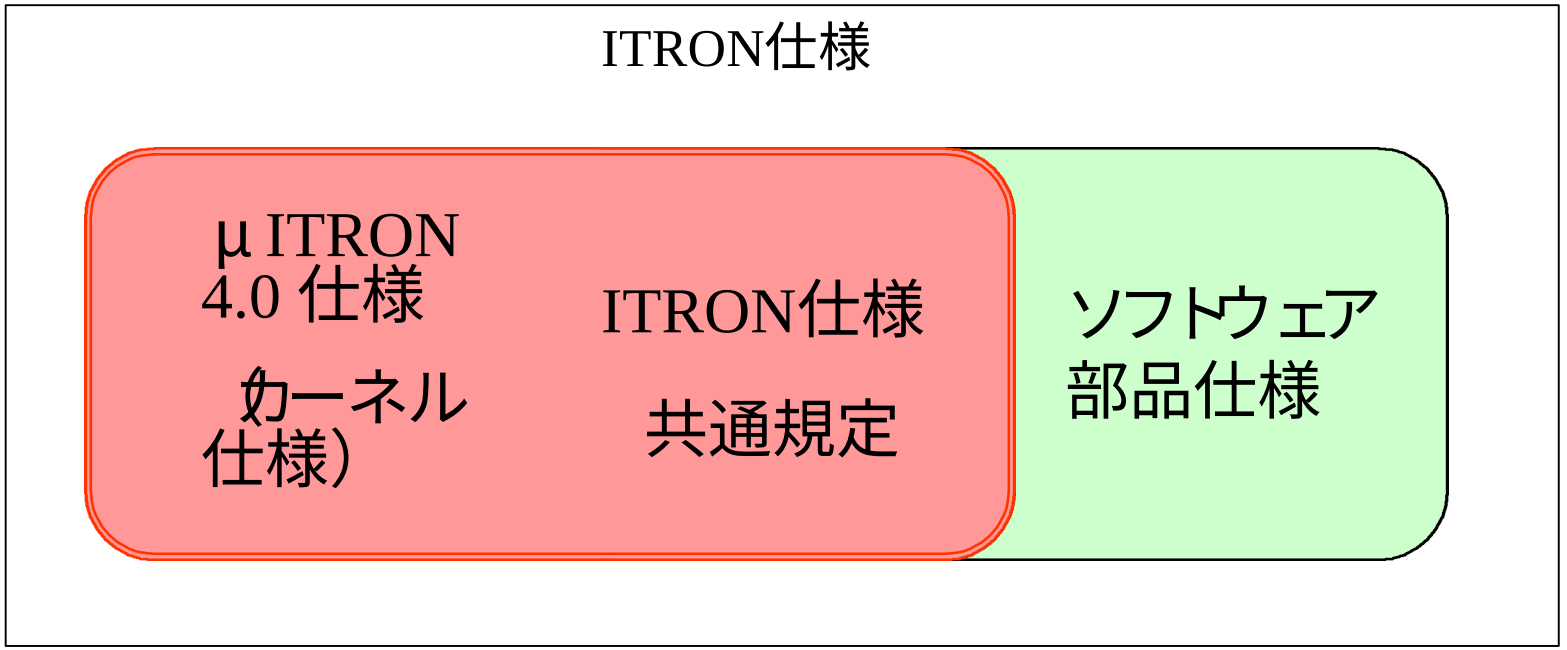
注:　ここでは、μITRON3.0仕様との違いを中心に説明する。

パート 1

- μITRON4.0 仕様共通規定

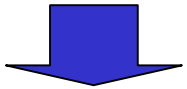
- 静的 API

ITRON仕様共通規定とは



ITRON仕様で用いる4つの用語

μITRON3.0 仕様： インプリメント依存



μITRON4.0 仕様： 4つの用語を定義

1) 実装依存
2) 実装定義
3) 未定義
4) 実装独自

1) 実装依存

定めた機能仕様に、実装やシステム状態が変化すると振舞いが変わる部分

アプリケーションは、実装依存の事項に依存しない記述が可能である。

2) 実装定義

定めた機能仕様に、実装時に規定するよう指示している項目

実装者は、実装した規定をマニュアルなどに示さなければならない。

アプリケーションは、実装定義の事項を移殖時に変更しなければならない。

3) 未定義

ある状況における振舞いに関して何も保証されない事項

アプリケーションがこの状況を作った場合の振舞いは、保証されていない。

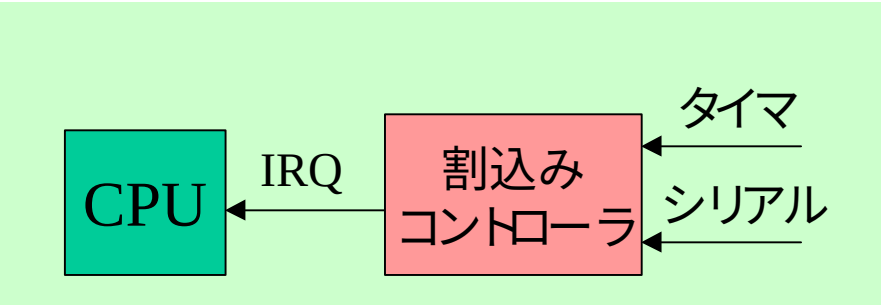
4) 実装独自

定められた範囲外の機能仕様を、実装時に独自に定義した項目

アプリケーションは、CPUまたはOS固有の機能であることを認識して使用すること。

例 1実装依存

一つの割り込み番号に対して複数の割込みサービスルーチンを登録できる。登録された割込みサービスルーチンは全て起動されなければならないが、起動順序は実装依存である。



例 2実装定義

ext_tsk実行時にエラーを検出した場合の振舞いは実装定義である。

```
void   ext_tsk();
```

エラーを返せないが、実装定義によりログに記録するなどの方法で、エラーが発生したことを知らせることが可能となる。

例 3未定義

可変長メモリプールの返却時に、獲得時に返されたメモリブロックの値以外を指定した場合の振舞いは未定義である。

```
get_mpl(id,   blksz,   p_blk);
rel_mpl(id,   *p_blk+1);
```

例 4実装独自

優先度は、正の値であるが、負の値も取ることができるようインプリメントする。

（μITRON3.0をインプリメントしたOS仕様が負の値を使用していたため、互換性のために負の値も取れるようにする。）

API

API (Application Program Interface)
の新しい用語

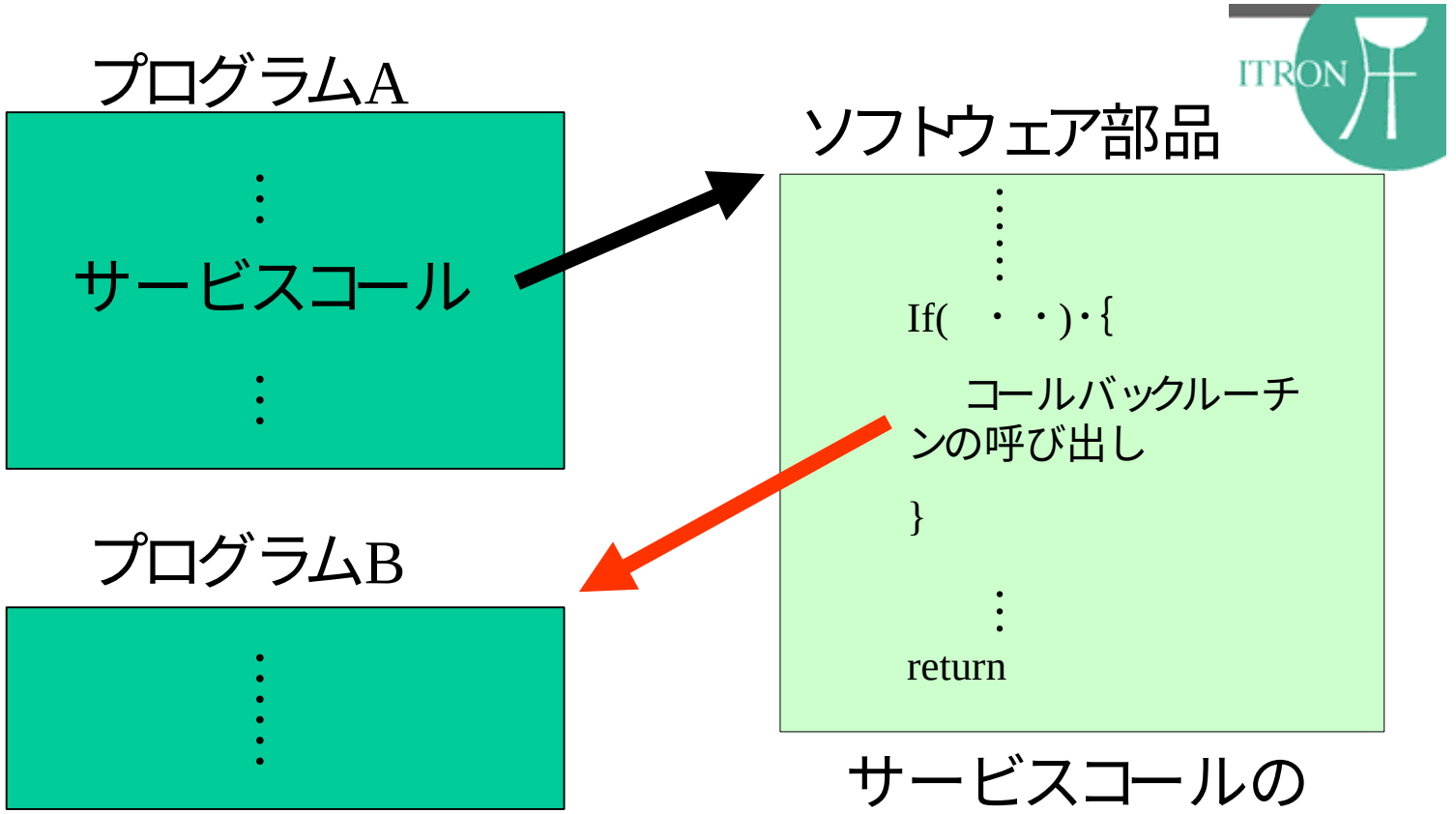
1) サービスコール
2) コールバック
3) 静的API

1) サービスコール

μITRON3.0仕様では、システムコールと呼ばれていたもの

2) コールバック

ソフトウェア部品から、アプリケーションプログラムが登録したルーチンを呼び出すインタフェース。


プログラムA
サービスコール
プログラムB
ソフトウェア部品
If(・・)・{
コールバックルーチンの呼び出し
}
return
プログラムBをコールバックルーチンという
サービスコールの次行へ戻らない方法が可能となる

3) 静的API

システムコンフィグレーションファイル中に記述し、構成やオブジェクトの初期状態を定義するためのインタフェース。

静的API は、サービスコールをシステム初期化時に実行していると考えてもよい

ID番号とオブジェクト番号

ID番号：

タスクID、イベントフラグID、

周期ハンドラID、アラームハンドラID等

Change　　Change

オブジェクト番号：

割込み番号等

外部要因で決定されるもの

優先度

μITRON3.0仕様：
正および負の値、1-8が使えること

μITRON4.0仕様：
正の値のみ

機能コード

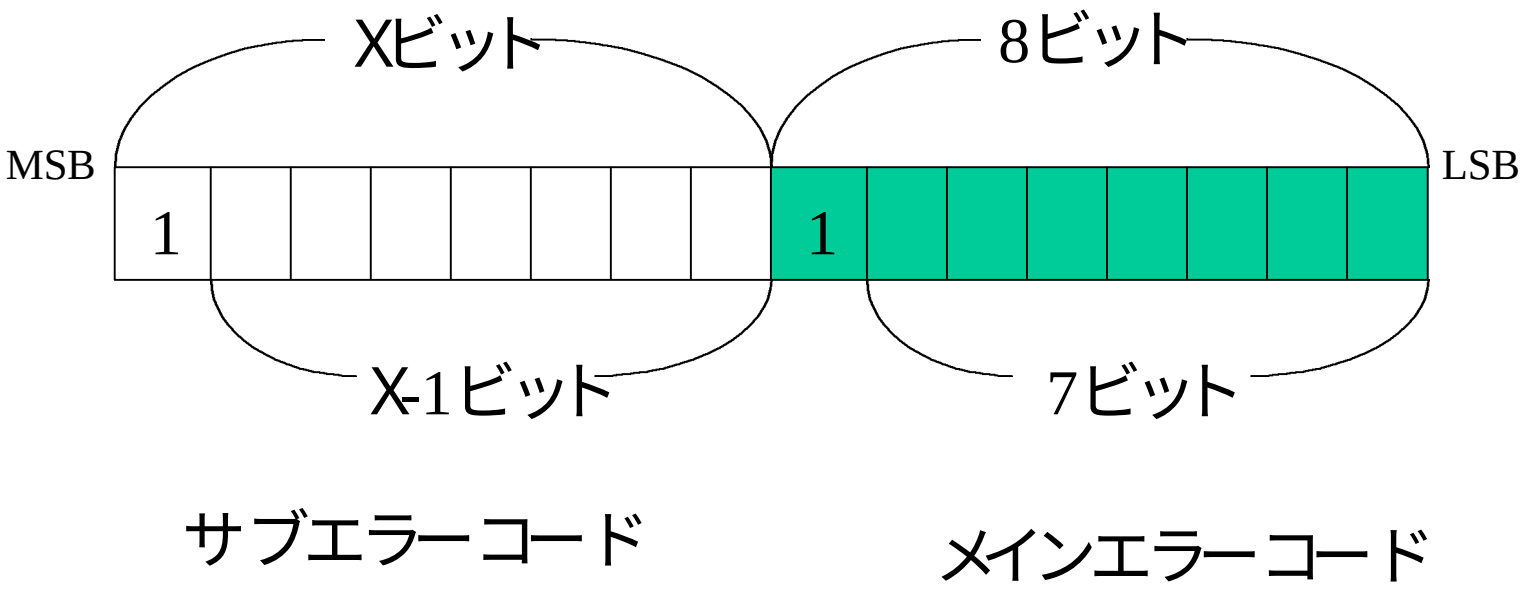
機能コードの値は、再割り当てされた

機能コードに依存したアプリケーションは見直しが必要。

エラーコード

エラーコードは、サブエラーコードとメインエラーコードから構成される

μITRON3.0仕様のエラーコードは、メインエラーコードとなった。ただし、メインエラーコードの値は、再割り当てされている。



エラーコードの構成図

エラーコード取得用マクロ

ER mercd = MERCD (ER ercd)
　エラーコードからメインエラーコードを取り出す。
ER sercd = SERCD (ER ercd)
　エラーコードからサブエラーコードを取り出す。

例： ER_UINT actcnt = can_act (ID tskid) ;

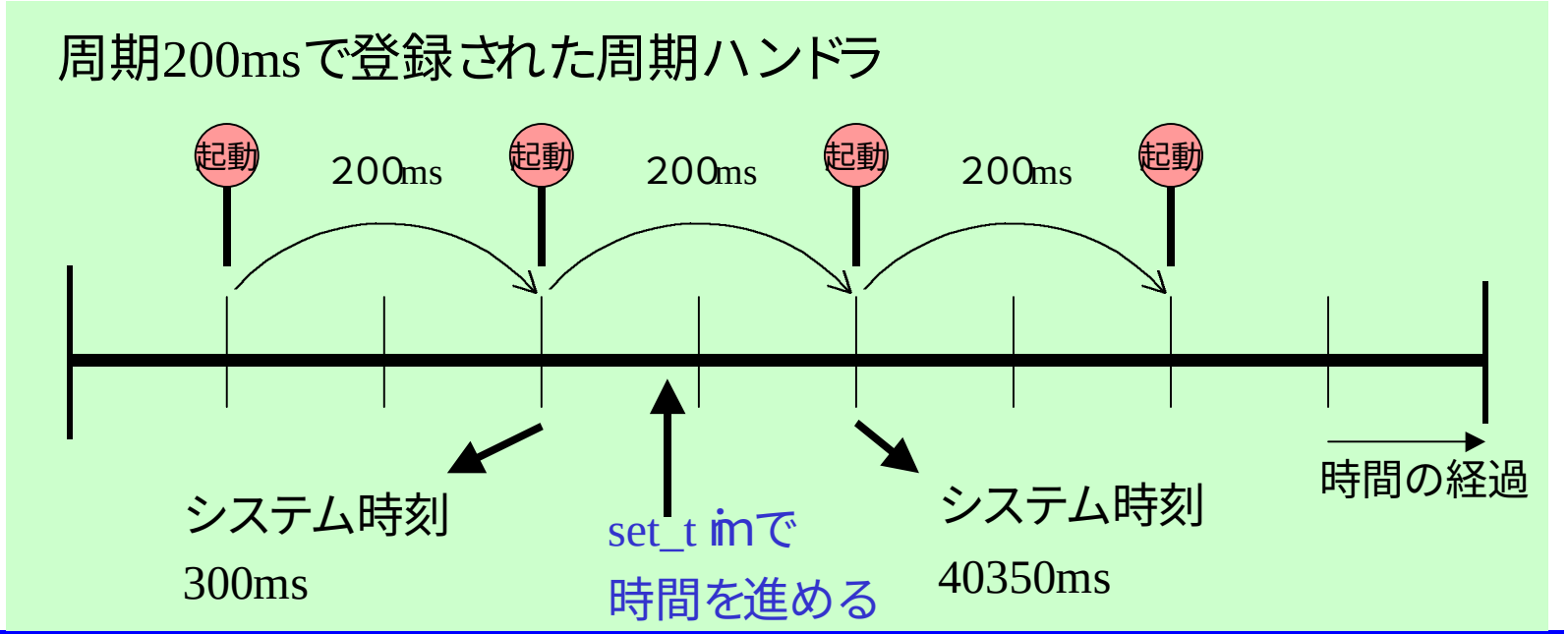
```
if( mercd = MERCD ( (ER)can_act( tskid) ) < 0) {
        ／＊ エラー ＊／
}else{
        ／＊キューイングされていた起動要求回数＊／
}
```

表1　エラーコードの差分

	エラーコード	コメント
μITRON4.0 で追加されたメインエラーコード	E_ILUSE	サービスコール不正使用
	E_NOID	ID番号不足
	E_CLS	待ちオブジェクトの状態変化
μITRON3.0 にあったが削除されたエラーコード	EN_XXXXX	（接続機能用のエラーコード）
	E_INOSPT	（ITRON/FILE仕様でのみ使用）

相対時間とシステム時刻

RELTIM と SYSTIM は別の時間



パート 1

● μITRON4.0 仕様共通規定

●静的 API

静的 API とコンフィギュレータ

ソフトウェア部品の、初期化プログラムやカーネルに要求する資源の生成が汎用的に記述可能

ソフトウェア部品提供者が要求する、ポータビリティがあるしくみを提供。

静的 API 一覧

CRE_TSK	CRE_MBF	DEF_INH
DEF_TEX	CRE_POR	ATT_ISR
CRE_SEM	CRE_MPF	DEF_SVC
CRE_FLG	CRE_MPL	DEF_EXC
CRE_DTQ	CRE_CYC	ATT_INI
CRE_MBX	CRE_ALM	
CRE_MTX	DEF_OVR	

システムコンフィグレーションファイル

静的APIと、プリプロセッサディレクティブ（“#define ”，“#ifdef ”等）が記述されたファイル

コンフィギュレータの入力ファイル

コンフィギュレータ

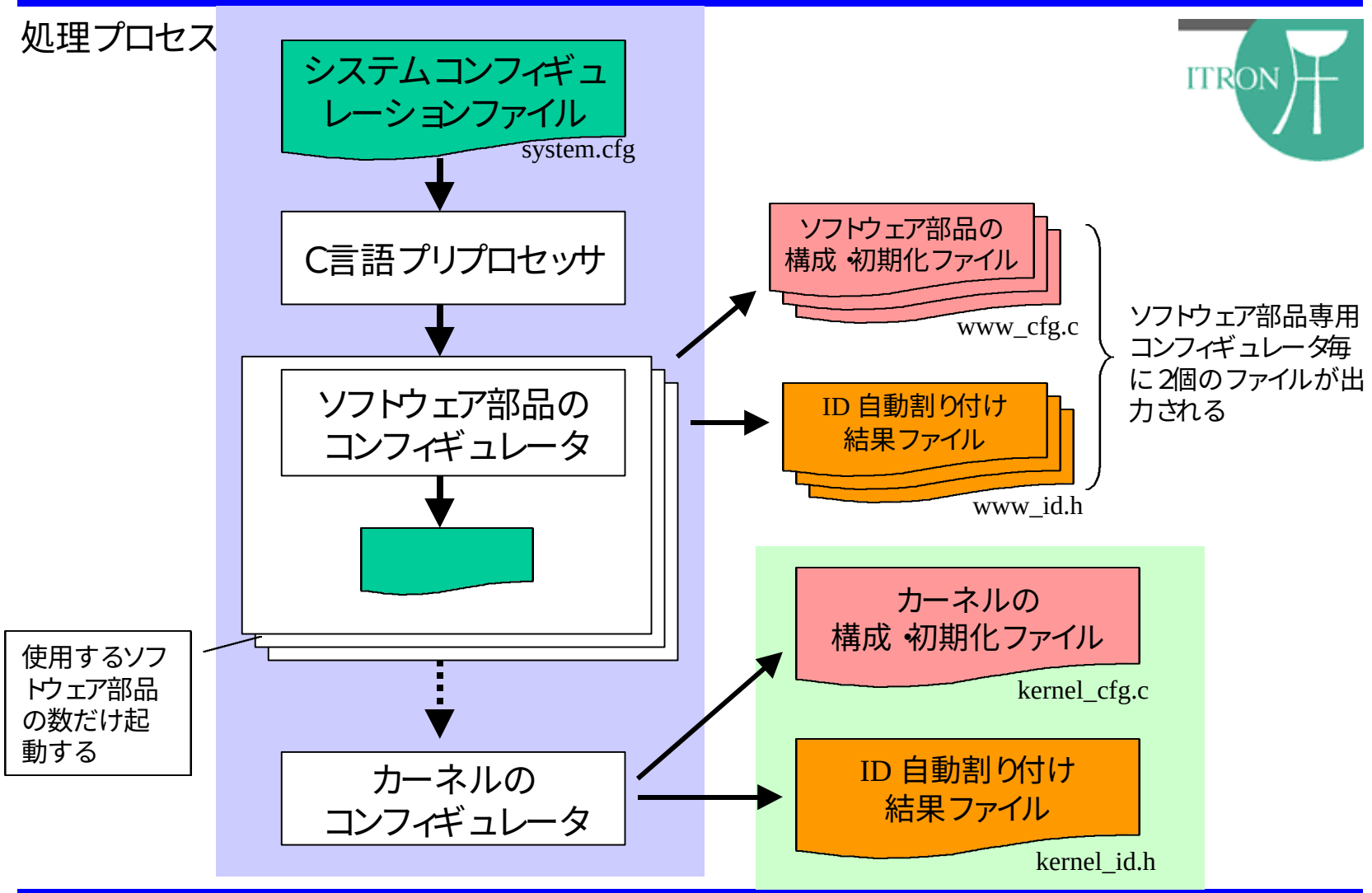
2種類のコンフィギュレータ：

- カーネルのコンフィギュレータ
- ソフトウェア部品のコンフィギュレータ

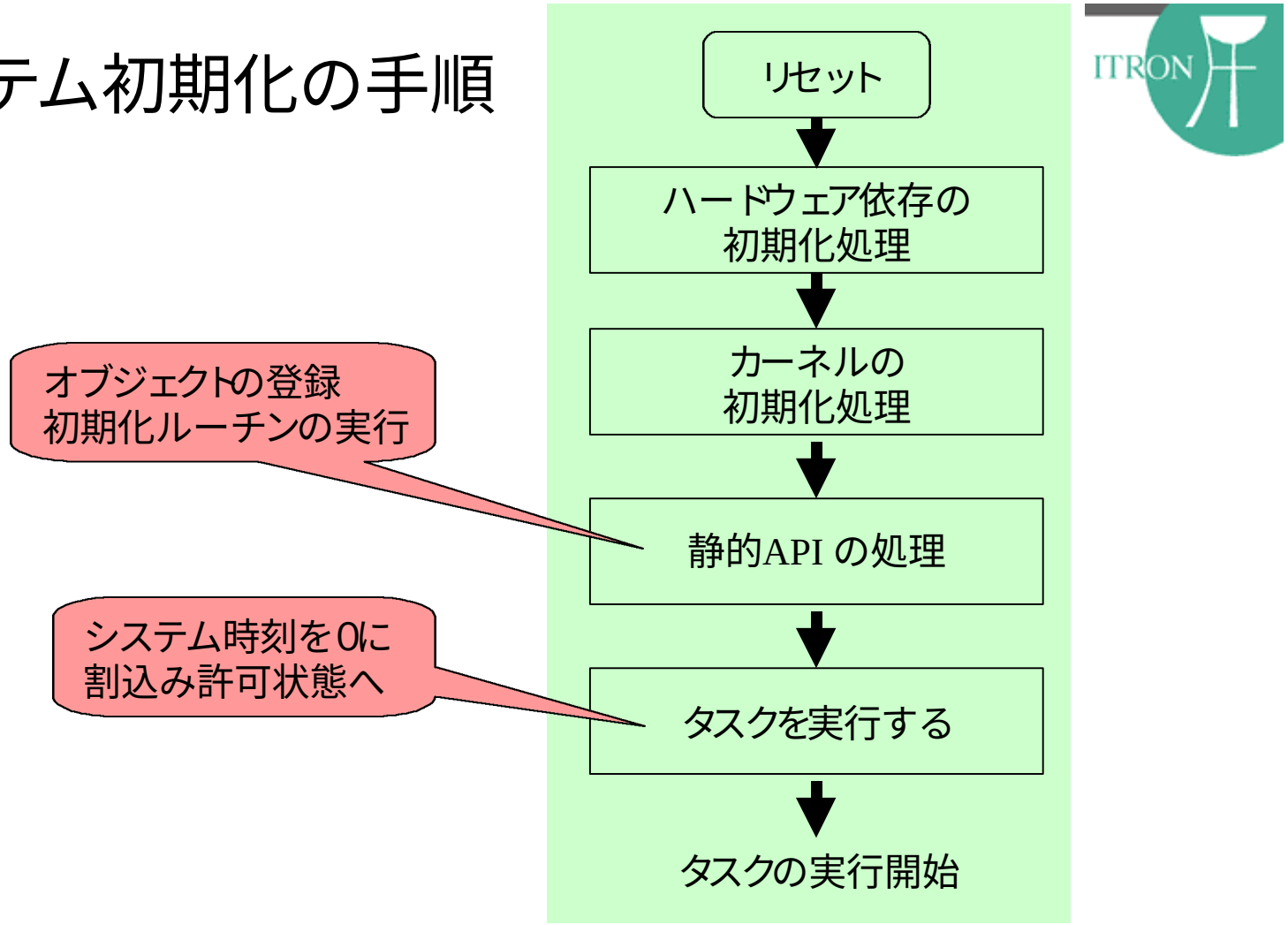
処理内容：

- 静的 APIの処理
- IDの自動割当て処理

処理プロセス


システムコンフィギュレーションファイル
system.cfg
C言語プリプロセッサ
ソフトウェア部品のコンフィギュレータ
使用するソフトウェア部品の数だけ起動する
カーネルのコンフィギュレータ
ソフトウェア部品の構成・初期化ファイル
www_cfg.c
ID 自動割り付け結果ファイル
www_id.h
ソフトウェア部品専用コンフィギュレータ毎に2個のファイルが出力される
カーネルの構成・初期化ファイル
kernel_cfg.c
ID 自動割り付け結果ファイル
kernel_id.h

システム初期化の手順


リセット
ハードウェア依存の
初期化処理
カーネルの
初期化処理
オブジェクトの登録
初期化ルーチンの実行
静的API の処理
システム時刻を0に
割込み許可状態へ
タスクを実行する
タスクの実行開始

カーネル構成定数

ソフトウェア部品がカーネルの構成情報を得るための定数

ソフトウェア部品のタスクを一時的に最低タスク優先度にしたい場合、“TMIN_TPRI ”をパラメータに指定する。

```
ercd = chg_pri ( _www_tskid, TMIN_TPRI) ;
```

カーネル共通構成定数一覧

構成定数	意味
TMIN_TPRI	タスク優先度の最小値
TMAX_TPRI	タスク優先度の最大値
TMIN_MPRI	メッセージ優先度の最小値
TMAX_MPRI	メッセージ優先度の最大値
TKERNEL_MAKER	カーネルのメーカコード
TKERNEL_PRID	カーネルの識別子番号
TKERNEL_SPVAR	μITRON 仕様のバージョン番号
TKERNEL_PRVER	カーネルのバージョン番号

内部識別子

カーネルやソフトウェア部品のプログラムで使用されるシンボルで、外部のプログラムから参照可能なもの

カーネル内部識別子：

kernel　または　_KERNEL_　で始まること

ソフトウェア部品内部識別子：

*www*　または　_*WWW*_　で始まること