μITRON4.0検定仕様書(案)

最終更新: 2001年3月27日

(社)トロン協会 ITRON部会

μITRON4.0検定仕様書 (案)

本仕様書は，μITRON4.0検定仕様書の案であり，検定制度の実施方法を検討する過程で改訂される可能性があります．

本仕様ならびに本仕様書に関する問い合わせ先は，下記の通りです．

(社)トロン協会 ITRON部会
〒108-0073 東京都港区三田1丁目3番39号 勝田ビル5階
TEL: 03-3454-3191　FAX: 03-3454-3224

---

§ TRONは“The Real-time Operating system Nucleus”の略称です。
§ ITRONは“Industrial TRON”の略称です。
§ μITRONは“Micro Industrial TRON”の略称です。
§ BTRONは“Business TRON”の略称です。
§ CTRONは“Central and Communication TRON”の略称です。
§ TRON，ITRON，μITRON，BTRON，およびCTRONは，特定の商品ないしは商品群を指す名称ではありません。

第1章　μITRON4.0 検定仕様書の策定

第１節　検定仕様書の目的と位置付け

現在実装されている ITRON 仕様カーネルは実装毎の仕様の違いが大きいために、ソフトウェア部品の流通性が十分に確保できないという問題がある。ソフトウェア部品の流通性を十分に確保するためには、ITRON 仕様に合致していることを検証するための検定が非常に重要であり、ここに検定仕様書を示す。

検定仕様書は、実装されている ITRON 仕様カーネルがあくまでも仕様に合致していくかの確認であり、決して実装されているカーネルの品質をテスト・保証するものではない。検定を受けたいカーネルメーカは、検定仕様書の規定に合致したテストを実施したという申告書（別途示すフォーマット）により申請する。なお、申告方法や提出書類は別途定めるものとする。

さらに検定仕様書は、スタンダードプロファイルと自動車制御用プロファイルに大別し、各々のテストすべき項目とその具体的手順を記載した。その他、製品マニュアルで確認すべき項目や、ITRON 仕様で定義するヘッダファイルで確認する項目も検定内容に含めた。

第２節　検定仕様書作成にあたっての考え方

検定仕様書は「テスト項目」と「テスト手順」から構成されている。

テスト手順は、プログラムによる確認、製品マニュアルによる確認、ヘッダファイルによる確認の３種類で構成されている。

「テスト項目」は、μITRON4.0 仕様書に記載されている内容と対応しており、確認すべき項目を洗い出したものである。

「テスト手順」は、テスト項目確認のための具体的手段を記載している。

- プログラムによる確認：　テストするためのプログラムがより自然に書けるよう、C 言語を意識した記載とし、検定を受ける側の負荷を少なくした。
- マニュアルによる確認：　実装で定義すべき事項が明確に規定されているかを確認するものである。したがって、プログラムによる動作確認は行わずドキュメントによる確認とした。
- ヘッダファイルによる確認：μITRON4.0 仕様書で規定するデータ型や定数に割付けられる値を、ファイル名と共に確認するものである。

2.1　最低限のテスト項目

- μITRON4.0 仕様書へ明示的に記載されている項目についてテストする。
- 多く(または、ほとんどすべて)のサービスコールで発生する可能性のあるメインエラーコードは、μITRON4.0 仕様書ではサービスコール毎に記述しないことを原則としている。したがって、検定仕様書においても、サービスコール毎にテストしないこととする。

2.2　実装に依存する仕様の扱い

- 「実装定義」：製品マニュアルなどで、実装における規定を示していること。かつ、実装における規定通りにプログラムが動作すること。
- 「実装依存」：ソフトウェア部品の流通性が確保できないため、検定仕様書から除く。
- 「未定義」　：ソフトウェア部品の流通性が確保できないため、検定仕様書から除く。
- 「実装独自」：ソフトウェア部品の流通性が確保できないため、検定仕様書から除く。

2.3　エラーコードのテスト方法

- 実装定義で省略することができるエラーコードは、テスト項目、およびテスト手順おいて、当該エラーコードを網掛けすることで、省略できないエラーコードと明確に分けた。
- サービスコールが複数のエラーを検出すべき状況で、どのエラーを示すエラーコードを返すかは、実装依存となっている。例えば、サービスコールを発行する際にパラメータが複数あり、

どのパラメータで E_PAR エラーを検出したのかわからない場合がある。このようなことを防ぐために、検定仕様書では E_PAR エラーを確認する場合は、要因を１つに絞りサービスコールを発行することで E_PAR エラーの確認を行っている。
・各サービスコールの機能説明に記載されているエラーコードは、すべて確認することとした。

2.4　タスク状態の確認

タスクの状態を参照する場合、ref_tsk(タスクの状態参照)サービスコールを用いると効率的なテストを行うことができる。しかし、スタンダードプロファイルや自動車制御用プロファイルでは ref_tsk サービスコールをサポートする必要はない。したがって、タスクの状態を参照する場合は、以下に示す方法で確認する場合も有ることとする。ただし、テストの内容によっては、別の方法でタスクの状態を確認することもある。

・実行状態
テスト手順で示すタスクの実行順序で行う。
・休止状態
get_pri サービスコールを発行し、E_OBJ エラーが返ることで確認する。
・待ち状態
rel_wai サービスコールを発行し、E_OK が返ることで確認する。
・強制待ち状態
rsm_tsk サービスコールを発行し、E_OK が返ることで確認する。

2.5　その他

・タスクの生成時に行うべき処理やタスクの起動時に行うべき処理の確認は、タスクが生成、または起動される度に実施するのではなく、１回の確認に止めた。
・システムの絶対時刻に関係するテストは、システム条件を満たさない場合があるので、テストは実施しない。
・ATT_ISR のみをサポートしているシステムでは、テスト手順の DEF_INH を ATT_ISR に置換してテストを行うこと。

第2章　スタンダードプロファイルのテスト項目

第１節　テスト項目の見方

テスト項目は、μITRON4.0 仕様書に記載されている内容と対応しており、確認すべき項目を洗い出したものである。また、「4.1　タスク管理機能」などタイトルの前の番号は、μITRON4.0 仕様書に記載されてあるものと一致させた。

以下に、act_tsk サービスコールを例にテスト項目の見方を説明する。

No.	区　分	テ　ス　ト　項　目		テスト手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK2-2
2		E_QOVR	キューイングオーバーフロー(起動要求キューイング数のオーバーフロー)	TSK2-20
3	サービスコール処　理	対象タスクが休止状態である場合には、tskid で指定されるタスクを、休止状態から実行可能状態に移行させること。		TSK2-12
4		タスクの起動時に行うべき処理を行うこと。		(Ref)タスク管理機能-3,4,5,6,7
5		タスクを起動する際のパラメータは、生成時に指定したタスクの拡張情報(exinf)と一致すること。また、起動されたタスクでは、拡張情報が正しく受け取れること。		TSK2-12
6		対象タスクが休止状態でない場合には、tskid で指定されるタスクに対する起動要求をキューイング(タスクの起動要求キューイング数に 1 を加える)すること。		TSK2-24
7		tskid に TSK_SELF が指定されると、自タスクを対象とすること。		TSK2-4,6

●No.
当該サービスコールのテスト番号を示す。

●区分
テスト項目をテスト内容毎に分類したものである。

●テスト項目
μITRON4.0 仕様書に記載されている内容であり、機能を確認する項目である。エラーコードに網掛けされてあるものは、実装定義でエラーの検出を省略できることを示す。ただし、これらのエラーコードの検出を省略する場合は、省略する旨が製品マニュアルに記載されていること。

●テスト手順参照
テスト項目を確認する具体的手段が、テスト手順のどの部分に記載されているかを示す。
(Ref)と記述されているものは、別のテスト項目で確認済であるので、ここでの確認項目ではないことを示す。
また、複数の番号が記述されている場合は、当該テスト項目の確認に複数のテスト手順が必要であることを示している。

第 2 節　テスト項目

2.　ITRON 仕様共通定義
2.1　ITRON 仕様共通概念

2.1.1　用語の意味
確認項目なし

2.1.2　API の構成要素

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	スタンダードプロファイルに含まれるサービスコールの名称、パラメータとリターンパラメータの種類・順序・名称・データ型が、μITRON4.0 仕様書と一致していること。	UMA-1
2	スタンダードプロファイルに含まれる静的 API の名称、パラメータの種類・順序・名称・データ型がμITRON4.0 仕様書と一致していること。	UMA-2
3	“kernel.h”から“itron.h”をインクルードしていること。	HED-1
4	オブジェクトの ID 番号などの自動割付けを行うコンフィギュレータは、自動割付けを行った結果を、自動割付け結果ヘッダファイル“ kernel_id.h ”に生成すること。	COM4-7
5	ITRON 仕様で標準化された同じヘッダファイルを複数回インクルードしてもエラーとならないこと。	COM1-1

2.1.3　オブジェクトの ID 番号とオブジェクト番号

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	ID 番号は 1～255 の範囲の正の値をサポートしていること。	SPC16-S256

2.1.4　優先度

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	1～16 の 16 段階のタスク優先度をサポートすること。	COM2-26
2	タスク優先度以上の段階数のメッセージ優先度をサポートすること。	COM2-34

2.1.5　機能コード
確認項目なし

2.1.6　サービスコールの返値とエラーコード

項番	確　　認　　事　　項			ﾃｽﾄ手順参照
1	エラーコードは、下位 8 ビットのメインエラーコードと、それを除いた上位ビットのサブエラーコードで構成される。			COM8-2,3
2	サブエラーコードをサポートする場合は、実装毎に定義しサブエラーコードに関する記述が製品マニュアルに記載されていること。			UMA-3
3	ﾒｲﾝｴﾗｰｺｰﾄﾞ	E_MACV	メモリアクセス違反	COM3-2

2.1.7　オブジェクト属性と拡張情報
確認項目なし

2.1.8　タイムアウトとノンブロッキング

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	タイムアウト時間を 0 に設定する(ポーリング)と、サービスコールの中で待ち状態に入るべき状況になっても、待ち状態に入らないこと。	SEM4-8
2	タイムアウト指定(TMO 型)は、正の値でタイムアウト時間を指定できること。	SEM4-25
3	TMO_FEVR で永久待ちを指定できること。	SEM4-29

2.1.9 相対時間とシステム時刻

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	システム時刻を変更した場合にも、相対時刻を用いて指定したイベントの発生する実世界の時間は変化しないこと。	COM3-16

2.1.10 システムコンフィギュレーションファイル

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	システムコンフィギュレーションファイルには、カーネルやソフトウェア部品の静的 API と ITRON 仕様共通静的 API に加えて、C 言語処理系のプリプロセッサディレクティブを記述できること。	COM4-3,5
2	システムコンフィギュレーションファイルは、まず、C言語のプリプロセッサに通される。次にソフトウェア部品のコンフィギュレータによって順に処理され、最後にカーネルのコンフィギュレータによって処理されること。	COM4-3,4,5
3	カーネルコンフィギュレータでは、カーネル構成・初期化ファイルと ID 自動割付け結果ヘッダファイル(kernel_id.h)を生成すること。	COM4-6,7
4	自分自身に対する静的 API または共通静的 API として解釈できない記述が含まれている場合には、エラーを報告すること。	COM5-1,2,3,4
5	カーネルのコンフィギュレータは、“ # ”で始まる行を無視すること。	COM4-5

2.1.11 静的 API の文法とパラメータ

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	サービスコールに対応する静的 API のパラメータは、対応するサービスコールのC言語 API に準ずること。ただし、パラメータにパケットへのポインタがある場合には、パケット中の各要素を“，”で区切り、“｛”と“｝”で囲んだ形で記述すること。	COM4-2
—	静的 API のパラメータは、記述に使える式によって分類される。	—
2	自動割付け対応整数値パラメータ	COM4-2
3	自動割付け非対応整数値パラメータ	INH1-S7
4	プリプロセッサ定数式パラメータ	COM4-2
5	一般定数式パラメータ	COM4-2
6	自動割付け対応整数値パラメータに単一の識別子が記述された場合、その静的 API を処理するコンフィギュレータが、識別子に整数値を割付けること。	COM4-7
7	整数値を割り付けられた識別子は、コンフィギュレータおよびそれ以降のコンフィギュレータが処理する静的 API のパラメータ中で、割り付けられた整数値に展開されるプリプロセッサマクロと同等なものとして用いることができること。	TEX1-S4
8	静的 API に文法エラーやパラメータ数の過不足があった場合には、それを検出したコンフィギュレータがエラーを報告すること。	COM5-1,2,3,4
9	静的 API の処理中にエラーを検出した場合の扱いについては、実装毎に定義すること。エラー発生時の扱いは製品マニュアルに記載されていること。	UMA-4

2.2. API の名称に関する原則

2.2.1 ソフトウェア部品識別名

確認項目なし

2.2.2 サービスコール

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	実装が独自に用意するサービスコールの名称が、μITRON4.0 仕様書と合致していること。また、その名称が製品マニュアルに記載されていること。	UMA-5

2.2.3 コールバック

確認項目なし

2.2.4 静的 API

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	実装が独自に用意する静的 API の名称が、μITRON4.0 仕様書と合致していること。また、その名称が製品マニュアルに記載されていること。	UMA-5

2.2.5 パラメータとリターンパラメータ

確認項目なし

2.2.6 データ型

確認項目なし

2.2.7 定数

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	実装が独自に定義するメインエラーコードの名称が、μITRON4.0 仕様書と一致していること。また、その名称が製品マニュアルに記載されていること。	UMA-6

2.2.8 マクロ

確認項目なし

2.2.9 ヘッダファイル

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	ITRON 仕様共通定義で規定されるデータ型、定数、マクロの定義などを含むヘッダファイルの名称を“ itron.h ”とすること。	HED-2
2	カーネル仕様で定められるすべてのサービスコールの宣言と、データ型、定数、マクロの定義などを含むヘッダファイルの名称を“ kernel.h ”とすること。	HED-3
3	カーネルのコンフィギュレータが生成する自動割付け結果ヘッダファイルの名称を“ kernel_id.h ”とすること。	HED-4

2.2.10 カーネルとソフトウェア部品の内部識別子

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	カーネルの内部識別子は、C 言語レベルで、_kernel_または_KERNEL_で始まる名称とすることを原則とする。	COM6-1

2.3 ITRON 仕様共通定義

2.3.1 ITRON 仕様共通データ型

項番	確　　認　　事　　項		ﾃｽﾄ手順参照
1	B	符号付き 8 ビット整数	HED-5 COM7-1,3
2	H	符号付き 16 ビット整数	HED-6 COM7-7,9
3	W	符号付き 32 ビット整数	HED-7 COM7-13,15
4	UB	符号無し 8 ビット整数	HED-8 COM7-4,6
5	UH	符号無し 16 ビット整数	HED-9 COM7-10,12
6	UW	符号無し 32 ビット整数	HED-10 COM7-16,18
7	VB	データタイプが定ならない 8 ビットの値　(実装毎に定義)	HED-11 COM7-19
8	VH	データタイプが定ならない 16 ビットの値　(実装毎に定義)	HED-12 COM7-20

9	VW	データタイプが定ならない 32 ビットの値　(実装毎に定義)	HED-13 COM7-21
10	VP	データタイプの定まらないものへのポインタ	HED-14 COM7-22
11	FP	プログラムの起動番地(ポインタ)	HED-15 COM7-23
12	INT	16 ビット以上の符号付き整数	HED-16 COM7-24,26
13	UINT	16 ビット以上の符号無し整数	HED-17 COM7-27,29
14	BOOL	真偽値(TRUE または FALSE)	HED-18 COM7-31,33
15	FN	機能コード(16 ビット以上の符号付き整数)	HED-19 COM7-34,36
16	ER	エラーコード(8 ビット以上の符号付き整数)	HED-20 COM7-37,39
17	ID	オブジェクトの ID 番号(16 ビット以上の符号付き整数)	HED-21 COM7-40,42
18	ATR	オブジェクト属性(8 ビット以上の符号無し整数)	HED-22 COM7-43,45
19	STAT	オブジェクトの状態(16 ビット以上の符号無し整数)	HED-23 COM7-46,48
20	MODE	サービスコールの動作モード(8 ビット以上の符号無し整数)	HED-24 COM7-49,51
21	PRI	優先度(16 ビット以上の符号付き整数)	HED-25 COM7-52,54
22	SIZE	メモリ領域のサイズ(ポインタと同じビット幅の符号無し整数)	HED-26 COM7-55,57
23	TMO	タイムアウト指定(16 ビット以上の符号付き整数、時間単位は 1 ミリ秒)	HED-27 COM7-58,60
24	RELTIM	相対時間(16 ビット以上の符号無し整数、時間単位は 1 ミリ秒)	HED-28 COM7-61,63
25	SYSTIM	システム時刻(16 ビット以上の符号無し整数、時間単位は 1 ミリ秒)	HED-29 COM7-64,66
26	VP_INT	データタイプが定まらないものへのポインタまたはプロセッサに自然なサイズの符号付き整数　(実装毎に定義)	HED-30 COM7-67,69
27	ER_BOOL	エラーコードまたは真偽値(符号付き整数)	HED-31 COM7-70,72,74,76
28	ER_ID	エラーコードまたは ID 番号(符号付き整数、負の ID 番号は表現できない)	HED-32 COM7-77,79
29	ER_UINT	エラーコードまたは符号無し整数(符号付き整数、符号無し整数を表現する場合の有効ビット数は UINT より 1 ビット短い)	HED-33 COM7-80,82

2.3.2 ITRON 仕様共通定数

項番	区　　分	確　認　事　項			ﾃｽﾄ手順参照
1	一　　般	NULL	無効ポインタ	0	HED-34
2		TRUE	真	1	HED-35
3		FALSE	偽	0	HED-36
4		E_OK	正常終了	0	HED-37
5	メイン エラーコード	E_SYS	システムエラー	-5	HED-38
6		E_NOSPT	未サポート機能	-9	HED-39
7		E_RSFN	予約機能コード	-10	HED-40
8		E_RSATR	予約属性	-11	HED-41
9		E_PAR	パラメータエラー	-17	HED-42
10		E_ID	不正 ID 番号	-18	HED-43
11		E_CTX	コンテキストエラー	-25	HED-44
12		E_MACV	メモリアクセス違反	-26	HED-45
13		E_ILUSE	サービスコール不正	-28	HED-46
14		E_NOMEM	メモリ不足	-33	HED-47
15		E_OBJ	オブジェクト状態エラー	-41	HED-48
16		E_QOVR	キューイングオーバーフロー	-43	HED-49
17		E_RLWAI	待ち状態の強制解除	-49	HED-50
18		E_TMOUT	ポーリング失敗またはタイムアウト	-50	HED-51
19	オブジェクト属性	TA_NULL	オブジェクト属性を指定しない	0	HED-52
20	タイムアウト 指　　定	TMO_POL	ポーリング	0	HED-53
21		TMO_FEVR	永久待ち	-1	HED-54

2.3.3 ITRON 仕様共通マクロ

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	ER mercd=MERCD(ER ercd)がヘッダファイル“ itron.h ”でマクロ定義され、エラーコードからメインエラーコードが取り出せること。	COM8-4 HED-55
2	ER sercd=SERCD(ER ercd) がヘッダファイル“ itron.h ”でマクロ定義され、エラーコードからサブエラーコードが取り出せること。	COM8-5 HED-56

2.3.4 ITRON 仕様共通静的 API

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	システムコンフィギュレーションファイルから、プリプロセッサ定数式パラメータおよび一般定数式パラメータの解釈に必要なＣ言語の宣言・定数とプリプロセッサマクロの定数などを含んだファイルをインクルードするための指定「INCLUDE(文字列);」ができること。	COM4-5,6

3. μITRON4.0 仕様の概念と共通定義

3.1 基本的な用語の意味

確認項目なし

3.2 タスク状態とスケジューリング規則

3.2.1 タスク状態

項番	確　　認　　事　　項		ﾃｽﾄ手順参照
1	タスク の待ち	タスクの待ち解除とは、タスクが待ち状態の時は実行可能状態に、二重待ち状態の時は強制待ち状態に移行させること。	SPC1-12,14,18
2		タスクの強制待ちからの再開とは、タスクが強制待ち状態のときは実行可能状態に、二重待ち状態のときは待ち状態に移行させること。	SPC1-28,34

3.2.2 タスクのスケジューリング規則

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	実行できるタスクが複数ある場合には、その中で最も優先順位の高いタスクが実行状態となり、他は実行可能状態となること。	SPC2-7
2	タスク間の優先順位は、異なる優先度を持つタスク間では、高い優先度を持つタスクの方が高い優先順位を持つこと。	SPC2-7
3	同じ優先度を持つタスク間では、先に実行できる状態(実行状態または実行可能状態)になったタスクの方が高い優先順位を持つこと。	SPC2-12
4	最も高い優先順位を持つタスクが変化した場合には、ただちにディスパッチが起こり、実行状態のタスクが切り替わること。	SPC2-9
5	ディスパッチが起こらない状態になっている場合には、実行状態のタスクの切替は、ディスパッチが起こる状態となるまで保留されること。	SPC3-7
6	優先順位の高いタスクが実行できる状態にある限り、それより優先順位の低いタスクは全く実行されないこと。すなわち、最も高い優先順位を持つタスクが待ち状態に入るなどの理由で実行できない状態とならない限り、他のタスクは全く実行されないこと。	SPC4-14
7	実行可能状態のタスクが実行状態になった後に実行可能状態に戻った直後には、同じ優先度を持つタスクの中で最高の優先順位を持っている。	SPC2-16
8	実行状態のタスクが待ち状態になった後に待ち解除されて実行できる状態になった直後には、同じ優先度を持つタスクの中で最低の優先順位となること。	SPC2-19

3.3 割込み処理モデル

3.3.1 割込みハンドラと割込みサービスルーチン

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	割込みハンドラの記述方法は実装毎に定義し、製品マニュアルに記載されていること。	UMA-7
2	DEF_INH と ATT_ISR の２つをサポートした場合は、その振舞いが製品マニュアルに記載されていること。	UMA-8
3	カーネルは、ある優先度より高い優先度を持つ割込みを、管理しないものとすることができる。どの優先度より高い優先度を持つものをカーネルの管理外の割込みとするかが製品マニュアルに記載されていること。	UMA-9

3.3.2 割込みの指定方法と割込みサービスルーチンの起動

確認項目なし

3.4 例外処理モデル

3.4.1 例外処理の枠組み

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	CPU 例外ハンドラは、プロセッサが検出する CPU 例外によって起動されること。	SPC5-2
2	CPU 例外ハンドラは、CPU 例外の種類毎に、アプリケーションで登録できること。	SPC5-S3
3	タスク例外処理ルーチンからリターンすると、中断された処理の実行が継続されること。	SPC6-4
4	タスク毎に一つのタスク例外処理ルーチンを、アプリケーションで登録できること。	SPC6-6

3.4.2 CPU 例外ハンドラで行える操作

項番	確　　認　　事　　項	テスト手順参照
1	CPU 例外ハンドラの記述方法は実装毎に定義し、製品マニュアルに記載されていること。	UMA-10
2	CPU 例外ハンドラ内で呼出し可能なサービスコールは実装毎に定義し、製品マニュアルに記載されていること。	UMA-11
—	CPU 例外ハンドラ内で次の操作を行う方法を用意すること。また、その方法は製品マニュアルに記載されていること。	—
3	CPU 例外が発生したコンテキストや状態の読出しができること。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。	UMA-12
4	CPU 例外が発生したタスクの ID 番号が読み出せること(例外を発生させたのがタスクである場合のみ)。	UMA-13
5	タスク例外処理の要求ができること。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	UMA-14

(注) CPU ロック状態で CPU 例外が発生した場合には、項番 4 および項番 5 の操作ができることは要求されない。

3.5 コンテキストとシステム状態

3.5.1 処理単位とコンテキスト

確認項目なし

3.5.2 タスクコンテキストと非タスクコンテキスト

項番	確　　認　　事　　項	テスト手順参照
1	タスクの実行されるコンテキストは、タスクコンテキストであること。	SPC7-2
2	割込みハンドラおよび周期ハンドラが実行されるコンテキストは、非タスクコンテキストである。	SPC7-5,17
3	非タスクコンテキストからサービスコールに渡すパラメータとして、自タスクを指定する TSK_SELF を用いることはできない。渡された場合には、E_ID エラーとなること。	SPC7-7
4	タスクコンテキストを実行中に CPU 例外が発生した場合、CPU 例外ハンドラが実行されるコンテキストが製品マニュアルに記載されていること。	UMA-15
5	非タスクコンテキストを実行中に CPU 例外が発生した場合には、CPU 例外ハンドラが実行されるコンテキストは非タスクコンテキストであること。	SPC7-10

3.5.3 処理の優先順位とサービスコールの不可分性

項番	確　　認　　事　　項	テスト手順参照
1	割込みハンドラの優先順位は、ディスパッチャの優先順位よりも高いこと。	TSK3-11
2	割込みハンドラおよび割込みサービスルーチン相互間の優先順位は、それらを起動する外部割込みの優先度に対応して定めることを基本に、実装毎に定義すること。	UMA-16
3	周期ハンドラの優先順位は、isig_tim を呼び出した割込みハンドラの優先順位以下で、ディスパッチャの優先順位よりも高いという範囲内で、実装毎に定義すること。	UMA-17
4	CPU 例外ハンドラの優先順位は、CPU 例外が発生した処理の優先順位と、ディスパッチャの優先順位のいずれよりも高いこと。	UMA-18
5	CPU 例外ハンドラと、割込みハンドラやタイムイベントハンドラとの間の優先順位は実装毎に定義すること。	UMA-19

3.5.4 CPU ロック状態

項番	確認事項			ﾃｽﾄ手順参照
1	CPU ロック状態では、割込みハンドラ(カーネルの管理外の割込みによって起動されるものを除く)や周期ハンドラは起動されず、ディスパッチも起こらないこと。すなわち、割込みが発生しても各ハンドラは起動されないこと。			SPC8-8,39
—	CPU ロック状態では、以下のサービスコールを呼び出すことができること。			—
2		loc_cpu	CPU ロック状態への移行	SPC8-10
3		iloc_cpu	CPU ロック状態への移行	SPC8-25
4		unl_cpu	CPU ロック状態の解除	SPC8-33
5		iunl_cpu	CPU ロック状態の解除	SPC8-29
6		sns_ctx	コンテキストの参照	SPC8-12
7		sns_loc	CPU ロック状態の参照	SPC8-14
8		sns_dsp	ディスパッチ禁止状態の参照	SPC8-16
9		sns_dpn	ディスパッチ保留状態の参照	SPC8-18
10		sns_tex	タスク例外処理禁止状態の参照	SPC8-20
11	割込みハンドラ内で CPU ロック解除状態にするための方法が用意されていること。			UMA-20
12	割込みハンドラから正しくリターンするための方法が用意されていること。			UMA-21
13	割込みサービスルーチンと周期ハンドラの実行開始直後は、CPU ロック解除状態になっていること。			SPC8-44 SPC9-3
14	CPU 例外ハンドラの起動と、ハンドラからのリターンによって、CPU ロック／ロック解除状態が変化しないこと。			SPC10-5,8,13,16
15	タスクの実行開始直後は、CPU ロック解除状態になっていること。			SPC8-2
16	タスク例外処理ルーチンの起動と、例外処理ルーチンからのリターンによって、CPU ロック／ロック解除状態が変化しないこと。			SPC10-27
—	タスク例外処理ルーチンからのリターン直後			–
17		CPU ロック状態でリターンした場合には CPU ロック状態になっていること。		SPC10-22
18		CPU ロック解除状態でリターンした場合には CPU ロック解除状態になっていること。		SPC10-31

3.5.5 ディスパッチ禁止状態

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	ディスパッチ禁止状態では、ディスパッチが起こらないこと。	SPC11-6
2	ディスパッチ禁止状態でも、自タスクを広義の待ち状態にする可能性がないサービスコールは呼び出せること。	SPC11-10,11
3	非タスクコンテキストから呼び出せるサービスコールは、ディスパッチ禁止状態でも制限を受けないこと。	SPC11-28
4	割込みハンドラの起動と、そこからのリターンによってディスパッチ禁止／許可状態は変化しないこと。	SPC11-23,48
5	割込みサービスルーチンの起動と、そこからのリータンによってディスパッチ禁止／許可状態は変化しないこと。	SPC11-23,48
6	周期ハンドラの起動と、そこからのリータンによってディスパッチ禁止／許可状態は変化しないこと。	SPC11-33,55
7	タスクの実行開始直後は、ディスパッチ許可状態になっていること。	SPC11-2
8	タスク例外処理ルーチンの起動と、そこからのリターンによってディスパッチ禁止／許可状態は変化しないこと。	SPC11-39,60
9	ポーリングを行うサービスコールは、自タスクを待ち状態にする可能性がないため、ディスパッチ禁止状態でも呼び出すことができること。	SPC11-8
10	ディスパッチ禁止状態で CPU ロック状態に移行し、その後 CPU ロックを解除しても、ディスパッチ禁止状態は保持されていること。	SPC11-19
11	CPU ロック状態でも、ディスパッチ禁止状態かディスパッチ許可状態かを参照できること。	SPC11-15,64

（注）

項番 4：割込みハンドラをサポートしていないシステムでは、テストを省略できる。

項番 5：割込みサービスルーチンをサポートしていないシステムでは、テストを省略できる。

3.5.6 ディスパッチ保留状態の間のタスク状態

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	ディスパッチ保留状態では、実行中のタスクがプリエンプトされるべき状況になっても、新たに実行すべき状態となったタスクにはディスパッチされないこと。実行すべきタスクへのディスパッチは、ディスパッチが起こる状態となるまで保留されること。	SPC12-6,8

3.6 非タスクコンテキストからのサービスコール呼出し

3.6.1 非タスクコンテキストから呼び出せるサービスコール

項番	確認事項			ﾃｽﾄ手順参照
－	非タスクコンテキスト専用のサービスコール			－
1		iact_tsk	タスクの起動	TSK3-7
2		iwup_tsk	タスクの起床	SYN3-10
3		irel_wai	待ち状態の強制解除	SYN6-11
4		iras_tex	タスク例外処理の要求	TEX3-16
5		isig_sem	セマフォ資源の返却	SEM3-16
6		iset_flg	イベントフラグのセット	FLG3-9
7		ipsnd_dtq	データキューへの送信(ポーリング)	DTQ4-10
8		ifsnd_dtq	データキューへの強制送信	DTQ7-12
9		isig_tim	タイムティックの供給	SPC13-23
10		irot_rdq	タスクの実行順位の回転	SYS2-27
11		iget_tid	実行状態のタスク ID の参照	SYS2-7
12		iloc_cpu	CPU ロック状態への移行	SYS2-9
13		iunl_cpu	CPU ロック状態の解除	SYS2-17
－	どのコンテキストからでも呼び出せるサービスコール			－
14		sns_ctx	コンテキストの参照	SPC13-2,13
15		sns_loc	CPU ロック状態の参照	SPC13-4,15
16		sns_dsp	ディスパッチ禁止状態の参照	SPC13-6,17
17		sns_dpn	ディスパッチ保留状態の参照	SPC13-8,19
18		sns_tex	タスク例外処理禁止状態の参照	SPC13-10,21

3.6.2 サービスコールの遅延実行

項番	確認事項	ﾃｽﾄ手順参照
1	サービスコールを遅延実行する場合にも、遅延実行することを許されないものを除いて、サービスコールの実行順序がサービスコールの呼び出された順序に一致しなければならないこと。	SPC14-7

3.6.3 呼び出せるサービスコールの追加

確認項目なし

3.7 システム初期化手順

項番	確認事項	ﾃｽﾄ手順参照
1	ATT_INI 以外の静的 API の処理は、システムコンフィギュレーションファイル中に記述された順序でおこなうこと。	SPC15-8
2	初期化ルーチンはアプリケーションで用意され、ATT_INI を使って登録され、実行されること。	SPC15-3
3	システム時刻は、システム初期化時点で時刻 0 とすること。	TIM-3
4	カーネルの初期化処理を呼び出す方法は、実装毎に定義すること。	UMA-22
5	静的 API の処理中にエラーを検出した場合の扱いは、実装毎に定義すること。	UMA-23
6	カーネルの管理外の割込みを禁止するかどうかは、実装毎に定義すること。	UMA-24
7	初期化ルーチン内でサービスコールを呼び出せるか否か、呼び出せる場合にはどのようなサービスコールが呼び出せるかは、実装毎に定義すること。	UMA-25
8	初期化ルーチンは、システムコンフィギュレーションファイル中に ATT_INI の記述のある順序で実行すること。	SPC15-3
9	カーネルの動作開始時点で初めて割込みを許可すること。	SPC15-4

3.8 オブジェクトの登録とその解除

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	ID 番号で識別されるオブジェクトについては、少なくとも 255 個のオブジェクトが登録できること。	SPC16-4
2	カーネルに登録できるオブジェクトの最大数や ID 番号の範囲は、実装毎に定義すること。	UMA-26
3	自動割付けされる ID 番号の範囲の指定方法は、実装毎に定義すること。	UMA-27

3.9 処理単位の記述形式

確認項目なし

3.10 カーネル構成定数／マクロ

確認項目なし

3.11 カーネル共通定義

3.11.1 カーネル共通定義

項番	区　　分	確　認　事　項			ﾃｽﾄ手順参照
1	オブジェクト属　性	TA_HLNG	高級言語用のインタフェースで処理単位で起動	0x00	HED-57
2		TA_TFIFO	タスクの待ち行列を FIFO 順に	0x00	HED-58
3		TA_TPRI	タスクの待ち行列をタスクの優先度順に	0x01	HED-59
4		TA_MFIFO	メッセージのキューを FIFO 順に	0x00	HED-60
5		TA_MPRI	メッセージのキューをメッセージの優先度順に	0x02	HED-61
6	その他のカーネル共通定義	TSK_SELF	自タスク指定	0	HED-62
7		TSK_NONE	該当するタスクが無い	0	HED-63

3.11.2 カーネル共通構成定数

項番	区　　分	確　認　事　項			ﾃｽﾄ手順参照
1	優先度の範囲	TMIN_TPRI	タスク優先度の最小値	1	HED-64
2		TMAX_TPRI	タスク優先度の最大値	16 以上	HED-65
3		TMIN_MPRI	メッセージ優先度の最小値	1	HED-66
4		TMAX_MPRI	メッセージ優先度の最大値	TMAX_TPRI 以上	HED-67
5	バージョン情報	TKERNEL_MAKER	カーネルのメーカーコード		HED-68
6		TKERNEL_PRID	カーネルの識別番号		HED-69
7		TKERNEL_SPVER	ITRON 仕様のバージョン番号		HED-70
8		TKERNEL_PRVER	カーネルのバージョン番号		HED-71

4. μITRON4.0 仕様の機能

4.1 タスク管理機能

No.	区　　分	テ　ス　ト　項　目			ﾃｽﾄ手順参照
1	タスクの生成時に行うべき処理	タスクの起動要求キューイング数をクリアすること。			TSK1-3
2		タスク例外処理ルーチンの定義されていない状態へ初期化すること。			TSK1-5
3	タスクの起動時に行うべき処理	タスクの優先度を初期化すること。			TSK1-20
4		起床要求キューイング数をクリアすること。			TSK1-22
5		強制待ち要求ネスト数をクリアすること。			SYN7-24
6		保留例外要因をクリアすること。			TEX3-4
7		タスク例外処理を禁止状態へ設定すること。			TEX3-2
8	タスクの終了	タスクのメインルーチンからリターンした場合にはタスクを終了すること。			(Ref) return-1
9	定　　数	TMAX_ACTCNT	タスクの起動要求キューイング数の最大値	1 以上	HED-72

CRE_TSK：タスクの生成（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号	TSK1-S4
2		E_RSATR	予約属性(tskatr が不正あるいは使用できない)	TSK1-S6
3		E_PAR	パラメータエラー(task が不正)	TSK1-S8
4		E_PAR	パラメータエラー(itskpri が不正)	TSK1-S10
5		E_PAR	パラメータエラー(stksz が不正)	TSK1-S12
6		E_OBJ	オブジェクト状態エラー(対象タスクが登録済)	TSK1-S14
7	静的 API 処　　理	タスクの生成時に行うべき処理を行うこと。		(Ref)タスク管理機能-1,2
―		tskatr に TA_ACT の指定がない場合		―
8		対象タスクを未登録状態から休止状態に移行させること。		TSK1-7
―		tskatr に TA_ACT の指定がある場合		―
9		対象タスクを未登録状態から実行可能状態に移行させること。		TSK1-1
10		タスクの起動時に行うべき処理を行うこと。		(Ref)タスク管理機能-3,4,5,6,7
11		拡張情報(exinf)がパラメータとして渡ること。また、起動されたタスクでは、起動情報が正しく受取れること。		TSK1-1

(注)

No.3　E_PAR エラーを検出できる task がシステムに存在しない場合は、テストを省略することができる。

No.5　E_PAR エラーを検出できる stksz がシステムに存在しない場合は、テストを省略することができる。

act_tsk：タスクの起動

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK2-2
2		E_QOVR	キューイングオーバーフロー(起動要求キューイング数のオーバーフロー)	TSK2-20
3	サービスコール 処　　理	対象タスクが休止状態である場合には、tskid で指定されるタスクを、休止状態から実行可能状態に移行させること。		TSK2-12
4		タスクの起動時に行うべき処理を行うこと。		(Ref)タスク管理機能-3,4,5,6,7
5		タスクを起動する際のパラメータは、生成時に指定したタスクの拡張情報(exinf)と一致すること。また、起動されたタスクでは、拡張情報が正しく受け取れること。		TSK2-12
6		対象タスクが休止状態でない場合には、tskid で指定されるタスクに対する起動要求をキューイング(タスクの起動要求キューイング数に１を加える)すること。		TSK2-24
7		tskid に TSK_SELF が指定されると、自タスクを対象とすること。		TSK2-4,6

iact_tsk：タスクの起動

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK3-3
2		E_QOVR	キューイングオーバーフロー(起動要求キューイング数のオーバーフロー) サービスコールを遅延実行する場合で、E_QOVR エラーを返すことを実装定義で省略できる。省略する場合は、E_OK を返すこと。(注)	TSK3-9
3	サービスコール処　　理	対象タスクが休止状態である場合には、tskid で指定されるタスクを、休止状態から実行可能状態に移行させること。		TSK3-13
4		タスクの起動時に行うべき処理を行うこと。		(Ref)ﾀｽｸ管理機能-3,4,5,6,7
5		タスクを起動する際のパラメータは、生成時に指定したタスクの拡張情報(exinf)と一致すること。また、起動されたタスクでは、拡張情報が正しく受け取れること。		TSK3-13
6		対象タスクが休止状態でない場合には、tskid で指定されるタスクに対する起動要求をキューイング(タスクの起動要求キューイング数に１を加える)すること。		TSK3-16
7		tskid に TSK_SELF が指定された場合には、E_ID エラーを返すこと。		TSK3-5

(注)

No.2　サービスコールを遅延実行する場合で、E_QOVR エラーを省略することが製品マニュアルに記載されていれば、E_OK を返すことができる。

can_act：タスク起動要求のキャンセル

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK2-10
2	サービスコール処　　理	タスクの起動要求キューイング数を０にクリアすること。		TSK2-8
3		０クリアする前の起動要求キューイング数を返値として actcnt に返すこと。		TSK2-16
4		tskid に TSK_SELF が指定されると、自タスクを対象とすること。		TSK2-6

ext_tsk：自タスクの終了

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	サービスコール処　　理	自タスクを実行状態から休止状態に移行させること。		TSK4-7
―		自タスクに対する起動要求がキューイングされている場合		―
2			起動要求キューイング数が１以上の場合には、起動要求キューイング数から１を減じること。	TSK4-11
3			タスクの起動時に行うべき処理を行うこと。	(Ref)ﾀｽｸ管理機能-3,4,5,6,7
4			実行可能状態に移行させること。	TSK4-9
5			タスクを起動する際のパラメータとして、タスクの拡張情報(exinf)を渡すこと。また、起動されたタスクでは、拡張情報が正しく受け取れること。	TSK4-9
6		サービスコール内でエラーが発生した場合の処理。		UMA-28

return：自タスクの終了

No.	区　　分	テ　ス　ト　項　目	テスト手順参照
1	サービスコール処　理	自タスクを実行状態から休止状態に移行させること。	TSK5-7
—		自タスクに対する起動要求がキューイングされている場合	—
2		起動要求キューイング数が１以上の場合には、起動要求キューイング数から１を減じること。	TSK5-11
3		タスクの起動時に行うべき処理を行い、実行可能状態に移行させること。	(Ref)タスク管理機能-3,4,5,6,7
4		タスクを起動する際のパラメータとして、タスクの拡張情報(exinf)を渡すこと。また、起動されたタスクでは、拡張情報が正しく受け取れること。	TSK5-9

ter_tsk：タスクの強制終了

No.	区　　分	テ　ス　ト　項　目		テスト手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK6-2
2		E_ILUSE	サービスコール不正使用(対象タスクが自タスク)	TSK6-4
3		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態)	TSK6-6
4	サービスコール処　理	tskid で指定されるタスクを、休止状態に移行させること。		TSK6-10
—		対象タスクに対する起動要求がキューイングされている場合		—
5		起動要求キューイング数から１を減じること。		TSK6-20
6		対象タスクを実行可能状態に移行させること。		TSK6-18
7		タスクの起動すべき処理を行うこと。		(Ref)タスク管理機能-3,4,5,6,7
8		タスクを起動する際のパラメータとして、タスクの拡張情報を渡すこと。また、起動されたタスクでは、拡張情報だ正しく受け取れること。		TSK6-18

chg_pri：タスク優先度の変更

No.	区　　分	テ　ス　ト　項　目		テスト手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK7-2
2		E_PAR	パラメータエラー(tskpri が不正)	TSK7-4
3		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態)	TSK7-6
4	サービスコール処　理	tskid で指定されるタスクの優先度を、tskpri で指定される値に変更すること。		TSK7-24
5		tskid に TSK_SELF が指定されると、自タスクを対象とすること。		TSK7-32
6		tskpri に TPRI_INI が指定されると、タスクの起動時優先度に変更すること。		TSK7-37
—		対象タスクが実行できる状態である場合		—
7		タスクの優先順位を、変更後の優先度にしたがって変化させること。		TSK7-22
8		変更後の優先度と同じ優先度を持つタスクの間では、対象タスクの優先順位を最低とすること。		TSK7-32
—		対象タスクが何らかのタスク優先度順の待ち行列につながれている場合		—
9		その待ち行列の中での順序を、変更後の優先度にしたがって変化させること。		SEM5-45
10		変更後の優先度と同じ優先度を持つタスクの間では、対象タスクを最後につなぐこと。		SEM5-42

get_pri：タスク優先度の参照

No.	区　　分	テ　ス　ト　項　目		テスト手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK7-8
2		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態)	TSK7-10
3	サービスコール処　理	tskid で指定されるタスクの優先度を参照し、tskpri に返すこと。		TSK7-13
4		tskid に TSK_SELF が指定されると、自タスクを対象とすること。		TSK7-16

4.2 タスク付属同期機能

No.	区　　分	テ　ス　ト　項　目			ﾃｽﾄ手順参照
1	定　　数	TMAX_WUPCNT	タスクの起床要求キューイング数の最大値	１以上	HED-73
2		TMAX_SUSCNT	タスクの強制待ち要求ネスト数の最大値	１以上	HED-74

slp_tsk：起床待ち

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	SYN1-17
—	サービスコール処　　理	自タスクに対する起床要求がキューイングされていない場合		—
2		自タスクを起床待ち状態に移行させること。		SYN1-25
—		自タスクに対する起床要求がキューイングされている場合		—
3		起床要求キューイング数から１を減じること。		SYN1-23
4		自タスクを待ち状態に移行させずに、そのまま実行を継続すること。		SYN1-8

tslp_tsk：起床待ち(タイムアウトあり)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_PAR	パラメータエラー(tmout が不正)	SYN2-2
2		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	SYN2-13
3		E_TMOUT	ポーリング失敗(自タスクに対する起床要求がキューイングされていない場合で、TMO_POL を指定した場合)またはタイムアウト	SYN2-4,17
—	サービスコール処　　理	自タスクに対する起床要求がキューイングされていない場合で、TMO_POL 以外を指定した場合		—
4		自タスクを起床待ち状態に移行させること。		SYN2-23
—		自タスクに対する起床要求がキューイングされている場合		—
5		起床要求キューイング数から１を減じること。		SYN2-10
6		自タスクを待ち状態に移行させずに、そのまま実行を継続すること。		SYN2-8
—		tmout の指定		—
7		正の値のタイムアウト時間が指定できること。		SYN2-17
8		TMO_POL が指定できること。		SYN2-8
9		TMO_FEVR が指定できること。		SYN2-23

(注)

No.1　E_PAR エラーを検出できる tmout がシステムに存在しない場合は、テストを省略することができる。

wup_tsk：タスクの起床

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	SYN1-2
2		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態)	SYN1-4
3		E_QOVR	キューイングオーバーフロー(起床要求キューイング数のオーバーフロー)	SYN1-14
4	サービスコール処　　理	tskid で指定されるタスクを、起床待ち状態から待ち解除すること。		SYN1-25
5		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		SYN1-25
6		対象タスクが起床待ち状態でない場合には、タスクの起床要求キューイング数に１を加えること。		SYN1-21
7		tskid に TSK_SELF が指定されると、自タスクを対象とすること。		SYN1-8

iwup_tsk：タスクの起床

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	SYN3-6
2		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態) サービスコールを遅延実行する場合で、E_OBJ エラーを返すことを実装定義で省略できる。省略する場合は E_OK を返すこと。(注)	SYN3-8
3		E_QOVR	キューイングオーバーフロー(起床要求キューイング数のオーバーフロー) サービスコールを遅延実行する場合で、E_QOVR エラーを返すことを実装定義で省略できる。省略する場合は E_OK を返すこと。(注)	SYN3-14
4	サービスコール処　理	tskid で指定されるタスクを、起床待ち状態から待ち解除すること。		SYN3-16
5		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		SYN3-16
6		対象タスクが起床待ち状態でない場合には、タスクの起床要求キューイング数に１を加えること。		SYN3-19
7		tskid に TSK_SELF が指定されると、E_ID エラーを返すこと。		SYN3-4

(注)

No.2　サービスコールを遅延実行する場合で、E_OBJ エラーを省略することが製品マニュアルに記載されていれば、E_OK を返すことができる。

No.3　サービスコールを遅延実行する場合で、E_QOVR エラーを省略することが製品マニュアルに記載されていれば、E_OK を返すことができる。

can_wup：タスク起床要求のキャンセル

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	SYN4-2
2		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態)	SYN4-4
3	サービスコール処　理	タスクの起床要求キューイング数を 0 にクリアすること。		SYN4-16
4		クリアする前の起床要求キューイング数を wupcnt に返すこと。		SYN4-14
5		tskid に TSK_SELF が指定されると、自タスクを対象タスクとすること。		SYN4-8

rel_wai：待ち状態の強制解除

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	SYN5-2
2		E_OBJ	オブジェクト状態エラー(対象タスクが待ち状態でない)	SYN5-4
3	サービスコール処理	対象タスクが待ち状態の時は実行可能状態に移行させること。		SYN5-7
4		対象タスクが二重待ち状態の時は強制待ち状態に移行させること。		SYN5-15
5		このサービスコールにより待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_RLWAI を返すこと。		SYN5-7

irel_wai：待ち状態の強制解除

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	SYN6-7
2		E_OBJ	オブジェクト状態エラー(対象タスクが待ち状態でない) サービスコールを遅延実行する場合で、E_OBJ エラーを返すことを実装定義で省略できる。省略する場合は、E_OK を返すこと。(注)	SYN6-9
3	サービスコール処理	対象タスクが待ち状態の時は実行可能状態に移行させること。		SYN6-15
4		対象タスクが二重待ち状態の時は強制待ち状態に移行させること。		SYN6-17
5		このサービスコールにより待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_RLWAI を返すこと。		SYN6-15

(注)

No.2　サービスコールを遅延実行する場合で、E_OBJ エラーを省略することが製品マニュアルに記載されていれば、E_OK を返すことができる。

sus_tsk：強制待ち状態への移行

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	SYN7-3
2		E_CTX	コンテキストエラー(ディスパッチ禁止状態で対象タスクに自タスクを指定)	SYN7-7
3		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態)	SYN7-11
4		E_QOVR	キューイングオーバーフロー(強制待ち要求ネスト数のオーバーフロー)	SYN7-29
5	サービスコール処　　理	対象タスクが実行できる状態の時は強制待ち状態に移行させること。		SYN7-24
6		対象タスクが待ち状態の時は二重待ち状態に移行させること。		SYN7-31
7		対象タスクの強制待ち要求ネスト数に１を加えること。		SYN7-29
8		tskid に TSK_SELF が指定されると、自タスクを対象タスクとすること。		SYN7-37

rsm_tsk：強制待ち状態からの再開

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	SYN7-13
2		E_OBJ	オブジェクト状態エラー(対象タスクが強制待ち状態でない)	SYN7-15
3	サービスコール処　　理	対象タスクの強制待ち要求ネスト数から１を減じること。		SYN7-33
—		減じた後の強制待ち要求ネスト数が０の場合		—
4		対象タスクが強制待ち状態の時は実行可能状態に移行させること。		SYN7-24
5		対象タスクが二重待ち状態の時は待ち状態に移行させること。		SYN7-25,31

frsm_tsk：強制待ち状態からの強制再開

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	SYN8-2
2		E_OBJ	オブジェクト状態エラー(対象タスクが強制待ち状態でない)	SYN8-4
3	サービスコール処　　理	対象タスクの強制待ち要求ネスト数を０とすること。		SYN8-7
4		対象タスクが強制待ち状態の時は実行可能状態に移行させること。		SYN8-7
5		対象タスクが二重待ち状態の時は待ち状態に移行させること。		SYN8-13,15

dly_tsk：自タスクの遅延

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_PAR	パラメータエラー(dlytim が不正)	SYN9-2
2		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	SYN9-9
3	サービスコール処　　理	サービスコールが呼び出された時刻から dlytim で指定された相対時間後に待ち解除されるよう設定し、自タスクを時間経過待ちに移行させること。		SYN9-9
4		指定された相対時間後に待ち解除された場合、このサービスコールは正常終了し、E_OK を返すこと。		SYN9-4

(注)

No.1　E_PAR エラーを検出できる dlytim がシステムに存在しない場合は、テストを省略することができる。

4.3　タスク例外処理機能

No.	区　　分	テ　ス　ト　項　目			ﾃｽﾄ手順参照
―	パラメータ	起動するタスク例外処理ルーチン			―
1		起動時の保留例外要因(texptn)とタスクの拡張情報(exinf)をパラメータとして渡すこと。			TEX2-12,13
2		タスクをタスク例外処理禁止状態に移行させ、保留例外要因を０クリアすること。			TEX2-22
3	起　動	タスク例外処理許可状態であり、保留例外要因が０でなく、タスクが実行状態であり、非タスクコンテキストが実行されていない場合。			TEX2-33
4		タスク例外処理許可状態であり、保留例外要因が０でなく、タスクが実行状態であり、CPU 例外ハンドラが実行されていない場合。			TEX2-33
―	タスク例外処理ルーチンからのリターン	タスク例外処理ルーチンからリターン			―
5		タスク例外処理ルーチンを起動する前に実行していた処理を継続すること。			TEX2-38
6		タスクをタスク例外処理許可状態に移行させること。			TEX2-40
7	再起動	タスク例外処理ルーチンからリターンし、保留例外要因が０でない場合には、再びタスク例外処理ルーチンを起動すること。			TEX2-35
―	ディスパッチ禁止状態	ディスパッチ禁止状態であっても以下の条件では、タスク例外処理ルーチンを起動できること。			―
8		タスク例外処理許可状態であり、保留例外要因が０でなく、タスクが実行状態であり、非タスクコンテキストが実行されていない場合。			TEX2-12
9		タスク例外処理許可状態であり、保留例外要因が０でなく、タスクが実行状態であり、CPU 例外ハンドラが実行されていない場合。			TEX2-12
10	実行するコンテキストと状態	タスク例外処理ルーチンは、タスクと同じコンテキストで実行すること。			TEX2-17
11	定　　数	TBIT_TEXPTN	タスク例外要因のビット数(TEXPTN 型のビット数)	16 ビット以上	HED-75
12		TEXPTN	タスク例外要因のビットパターン(符号無し整数)	16 ビット以上	HED-76

DEF_TEX：タスク例外処理ルーチンの定義（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TEX1-S2
2		E_RSATR	予約属性(texatr が不正あるいは使用できない)	TEX1-S5
3		E_PAR	パラメータエラー(texrtn が不正)	TEX1-S7
4	静的 API 処　　理	tskid で指定されるタスクに、定義情報に基づいてタスク例外処理ルーチンを定義できること。		TEX2-12

ras_tex：タスク例外処理の要求

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TEX2-2
2		E_PAR	パラメータエラー(rasptn が不正)	TEX2-4
3		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態、対象タスクにタスク例外処理ルーチンが定義されていない)	TEX2-6,44
4	サービスコール 処　　理	対象タスクの保留例外要因を、サービスコール呼出し前の保留例外要因と rasptn の値のビット毎の論理和に更新すること。		TEX2-35
5		tskid に TSK_SELF が指定されると、自タスクを対象タスクとすること。		TEX2-12

iras_tex：タスク例外処理の要求

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TEX3-8
2		E_PAR	パラメータエラー(rasptn が不正)	TEX3-10
3		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態、対象タスクにタスク例外処理ルーチンが定義されていない)	TEX3-12,20
4	サービスコール 処　　理	対象タスクの保留例外要因を、サービスコール呼出し前の保留例外要因と rasptn の値のビット毎の論理和に更新すること。		TEX3-24
5		tskid に TSK_SELF が指定されると、E_ID エラーを返すこと。		TEX3-14

dis_tex：タスク例外処理の禁止

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_OBJ	オブジェクト状態エラー(自タスクにタスク例外処理ルーチンが定義されていない)	TEX4-16
2	サービスコール 処　　理	自タスクを、タスク例外処理禁止状態に移行させること。		TEX4-6

ena_tex：タスク例外処理の許可

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_OBJ	オブジェクト状態エラー(自タスクにタスク例外処理ルーチンが定義されていない)	TEX4-18
2	サービスコール 処　　理	自タスクを、タスク例外処理許可状態に移行させること。		TEX4-2

sns_tex：タスク例外処理禁止状態の参照

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　　理	実行状態のタスクが、タスク例外処理禁止状態の場合にはTRUE を返すこと。	TEX2-19
2		実行状態のタスクが、タスク例外処理許可状態の場合に FALSE を返すこと。	TEX2-40
3		非タスクコンテキストから呼び出された場合で、実行状態のタスクがない時には、TRUE を返すこと。	TEX4-22

4.4　同期・通信機能

4.4.1　セマフォ

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	最大資源数	セマフォの最大資源数の最大値	65535 以上	HED-77

CRE_SEM：セマフォの生成（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(semid が不正あるいは使用できない)	SEM1-S2
2		E_RSATR	予約属性(sematr が不正あるいは使用できない)	SEM1-S4
3		E_PAR	パラメータエラー(isemcnt>maxsem)	SEM1-S6
4		E_PAR	パラメータエラー(maxsem が最大資源数を越えた)	SEM1-S8
5		E_OBJ	オブジェクト状態エラー(対象セマフォが登録済)	SEM1-S11
－	静的 API 処　　理	sematr には以下の指定ができ、オブジェクトが生成できること。		－
6		TA_TFIFO		SEM2-29
7		TA_TPRI		SEM5-11
8		isemcnt で示す初期値が正しいこと。		SEM2-14
9		maxsem で示す最大資源数が正しいこと。		SEM2-8

(注)

No.4　E_PAR エラーを検出できる maxsem がシステムに存在しない場合は、テストを省略することができる。

sig_sem：セマフォ資源の返却

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(semid が不正あるいは使用できない)	SEM2-2
2		E_QOVR	キューイングオーバーフロー(最大資源数を越える返却)	SEM2-8
－	サービスコール処　　理	資源の獲得を待っているタスクがある場合		－
3		待ち行列の先頭のタスクを待ち解除すること。		SEM2-29
4		対象セマフォの資源数は変化してはならないこと。		SEM2-31
5		待ち状態に入ったサービスコールの返値として E_OK を返すこと。		SEM2-29
－		資源の獲得を待っているタスクがない場合		－
6		対象セマフォの資源数に１を加えること。		SEM2-16

isig_sem：セマフォ資源の返却

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(semid が不正あるいは使用できない)	SEM3-14
2		E_QOVR	キューイングオーバーフロー(最大資源数を越える返却) サービスコールを遅延実行する場合で、E=QOVR エラーを返すことこを実装定義で省略できる。省略する場合は、E_OK を返すこと。(注)	SEM3-18
—	サービスコール処　　理	資源の獲得を待っているタスクがある場合		—
3		待ち行列の先頭のタスクを待ち解除すること。		SEM3-22
4		対象セマフォの資源数は変化してはならないこと。		SEM3-25
5		待ち状態に入ったサービスコールの返値として E_OK を返すこと。		SEM3-22
—		資源の獲得を待っているタスクがない場合		—
6		対象セマフォの資源数に１を加えること。		SEM3-35

(注)

No.2　サービスコールを遅延実行する場合で、E_OBJ エラーを省略することが製品マニュアルに記載されていれば、E_OK を返すことができる。

wai_sem：セマフォ資源の獲得

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(semid が不正あるいは使用できない)	SEM2-4
2		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	SEM2-37
—	サービスコール処　　理	対象セマフォの資源数が１以上の場合		—
3		セマフォの資源数から１を減じること。		SEM2-10
4		待ち状態とせずにサービスコールの処理を終了すること。		SEM2-10
—		対象セマフォの資源数が０の場合		—
5		対象セマフォの資源数は０のまま変化してはならないこと。		SEM2-33
—		セマフォ属性が TA_TFIFO		—
6			自タスクを待ち行列の末尾につなぐこと。	SEM2-33
—		セマフォ属性が TA_TPRI		—
7			優先度順で待ち行列につなぐこと。	SEM3-22
8			同じ優先度のタスクの中では最後につなぐこと。	SEM3-29

pol_sem：セマフォ資源の獲得(ポーリング)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(semid が不正あるいは使用できない)	SEM2-6
2		E_TMOUT	ポーリング失敗	SEM2-14
—	サービスコール処　　理	対象セマフォの資源数が１以上の場合		—
3		セマフォの資源数から１を減じること。		SEM2-12
4		待ち状態とせずにサービスコールの処理を終了すること。		SEM2-12

twai_sem：セマフォ資源の獲得(タイムアウトあり)

No.	区　　分	テ　ス　ト　項　目		テスト手順参照
1	エラーコード	E_ID	不正 ID 番号(semid が不正あるいは使用できない)	SEM4-2
2		E_PAR	パラメータエラー(tmout が不正)	SEM4-4
3		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	SEM4-29
4		E_TMOUT	タイムアウト	SEM4-25
―	サービスコール処　理	対象セマフォの資源数が１以上の場合		―
5		セマフォの資源数から１を減じること。		SEM4-6
6		待ち状態とせずにサービスコールの処理を終了すること。		SEM4-6
―		対象セマフォの資源数が０の場合		―
7		対象セマフォの資源数は０のまま変化してはならないこと。		SEM4-21
―		セマフォ属性が TA_TFIFO の場合		―
8		自タスクを待ち行列の末尾につなぐこと。		SEM4-29
―		セマフォ属性が TA_TPRI の場合		―
9		優先度順で待ち行列につなぐこと。		SEM5-11
10		同じ優先度のタスクの中では最後につなぐこと。		SEM5-19
―		tmout の指定		―
11		正の値のタイムアウト時間が指定できること。		SEM4-21
12		TMO_POL が指定できること。		SEM4-8
13		TMO_FEVR が指定できること。		SEM4-29

(注)

No.2　E_PAR エラーを検出できる tmout がシステムに存在しない場合は、テストを省略することができる。

4.4.2　イベントフラグ

No.	区　　分	テ　ス　ト　項　目			テスト手順参照
1	定　　数	FLGPTN	イベントフラグのビットパターン(符号無し整数)	16 ビット以上	HED-78
2		TBIT_FLGPTN	イベントフラグのビット数	16 ビット以上	HED-79

CRE_FLG：イベントフラグの生成（静的 API）

No.	区　　分	テ　ス　ト　項　目		テスト手順参照
1	エラーコード	E_ID	不正 ID 番号(flgid が不正あるいは使用できない)	FLG1-S2
2		E_RSATR	予約属性(flgatr が不正あるいは使用できない)	FLG1-S4
3		E_PAR	パラメータエラー(iflgptn が不正)	FLG1-S6
4		E_OBJ	オブジェクト状態エラー(対象イベントフラグが登録済)	FLG1-S9
―	静的 API 処　理	flgatr には以下の指定ができ、オブジェクトが生成できること。		―
5		TA_TFIFO		FLG2-26
6		TA_TPRI		FLG2-28
7		TA_TFIFO｜TA_WSGL		FLG2-26
8		TA_TPRI｜TA_WSGL		FLG2-28
9		TA_TFIFO｜TA_CLR		FLG3-20
10		TA_TPRI｜TA_CLR		FLG2-72
11		TA_TFIFO｜TA_WSGL｜TA_CLR		FLG3-20
12		TA_TPRI｜TA_WSGL｜TA_CLR		FLG2-72
13		iflgptn で示すビットパターンの初期値が正しいこと。		FLG2-73

(注)

No.3　E_PAR エラーを検出できる iflgptn がシステムに存在しない場合は、テストを省略することができる。

set_flg：イベントフラグのセット

No.	区　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(flgid が不正あるいは使用できない)	FLG2-2
2		E_PAR	パラメータエラー(setptn が不正)	FLG2-4
3	サービスコール処　理	対象イベントフラグのビットパターンを、サービスコール呼出し前のビットパターンと setptn の値のビット毎の論理和に更新すること。		FLG2-49
4		対象イベントフラグのビットパターンが更新された結果、そのイベントフラグで待っているタスクの待ち解除条件を満たした場合には、該当するタスクを待ち解除すること。		FLG2-64
—		待ち解除されたタスクに対しての処理		—
5		待ち状態に入ったサービスコールの返値として E_OK を返すこと。		FLG2-64,84
6		待ち解除時のビットパターンとして、この時の(待ち解除条件を満たした)ビットパターンを返すこと。		FLG2-65,85
7		TA_CLR が指定されている場合には、イベントフラグのビットパターンのすべてのビットをクリアし、サービスコールの処理を終了すること。		FLG2-87

（注）

No.2　E_PAR エラーを検出できる setptn がシステムに存在しない場合は、テストを省略することができる。

iset_flg：イベントフラグのセット

No.	区　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(flgid が不正あるいは使用できない)	FLG3-5
2		E_PAR	パラメータエラー(setptn が不正)	FLG3-7
3	サービスコール処　理	対象イベントフラグのビットパターンを、サービスコール呼出し前のビットパターンと setptn の値のビット毎の論理和に更新すること。		FLG3-18
4		対象イベントフラグのビットパターンが更新された結果、そのイベントフラグで待っているタスクの待ち解除条件を満たした場合には、該当するタスクを待ち解除すること。		FLG3-17
—		待ち解除されたタスクに対しての処理		—
5		待ち状態に入ったサービスコールの返値として E_OK を返すこと。		FLG3-17
6		待ち解除時のビットパターンとして、この時の(待ち解除条件を満たした)ビットパターンを返すこと。		FLG3-18
7		TA_CLR が指定されている場合には、イベントフラグのビットパターンのすべてのビットをクリアし、サービスコールの処理を終了すること。		FLG3-23

（注）

No.2　E_PAR エラーを検出できる setptn がシステムに存在しない場合は、テストを省略することができる。

clr_flg：イベントフラグのクリア

No.	区　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(flgid が不正あるいは使用できない)	FLG2-6
2		E_PAR	パラメータエラー(clrptn が不正)	FLG2-8
3	サービスコール処　理	対象イベントフラグのビットパターンを、サービスコール呼出し前のビットパターンと clrptn の値のビット毎の論理積に更新すること。		FLG2-37

（注）

No.2　E_PAR エラーを検出できる clrptn がシステムに存在しない場合は、テストを省略することができる。

wai_flg：イベントフラグ待ち

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(flgid が不正あるいは使用できない)	FLG2-10
2		E_PAR	パラメータエラー(waiptn が不正)	FLG2-12
3		E_PAR	パラメータエラー(wfmode が不正)	FLG2-14
4		E_PAR	パラメータエラー(p_flgptn が不正)	FLG2-16
5		E_ILUSE	サービスコール不正使用(TA_WSGL 属性が指定されたイベントフラグで待ちタスクあり)	FLG2-56
6		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	FLG2-60
—	サービスコール処　理	wfmode には、以下が指定できること。		—
7		TWF_ANDW		FLG2-56,69
8		TWF_ORW		FLG2-64,72
—		対象イベントフラグのビットパターンが waiptn と wfmode で指定される待ち条件を満たしている場合		—
9		自タスクを待ち状態とせずにサービスコールの処理を終了すること。		FLG2-30,69
10		flgptn には、この時の(待ち解除条件を満たした)ビットパターンを返すこと。		FLG2-31,70
11		TA_CLR が指定されている場合には、イベントフラグのビットパターンのすべてのビットをクリアすること。		FLG2-94
—		対象イベントフラグのビットパターンが waiptn と wfmode で指定される解除条件を満たしていない場合		—
12		自タスクを待ち行列につなぎ、イベントフラグ待ち状態に移行させること。		FLG2-60
13		waiptn に 0 を指定した場合には、E_PAR エラーを返すこと。		FLG2-67

(注)

No.2　E_PAR エラーを検出できる waiptn がシステムに存在しない場合は、テストを省略することができる。

No.4　E_PAR エラーを検出できる p_flgptn がシステムに存在しない場合は、テストを省略することができる。

pol_flg：イベントフラグ待ち(ポーリング)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(flgid が不正あるいは使用できない)	FLG2-18
2		E_PAR	パラメータエラー(waiptn が不正)	FLG2-20
3		E_PAR	パラメータエラー(wfmode が不正)	FLG2-22
4		E_PAR	パラメータエラー(p_flgptn が不正)	FLG2-24
5		E_ILUSE	サービスコール不正使用(TA_WSGL 属性が指定されたイベントフラグで待ちタスクあり)	FLG2-58
6		E_TMOUT	ポーリング失敗	FLG2-33
—	サービスコール処　理	wfmode には、以下が指定できること。		—
7		TWF_ANDW		FLG2-37,77
8		TWF_ORW		FLG2-43,84
—		対象イベントフラグのビットパターンが waiptn と wfmode で指定される待ち条件を満たしている場合		—
9		flgptn には、この時の(待ち解除条件を満たした)ビットパターンを返すこと。		FLG2-44
10		属性に TA_CLR が指定されている場合には、イベントフラグのビットパターンのすべてのビットをクリアすること。		FLG2-80
11		waiptn に 0 を指定した場合には、E_PAR エラーを返すこと。		FLG2-39

（注）

No.2　E_PAR エラーを検出できる waiptn がシステムに存在しない場合は、テストを省略することができる。

No.4　E_PAR エラーを検出できる p_flgptn がシステムに存在しない場合は、テストを省略することができる。

twai_flg：イベントフラグ待ち(タイムアウトあり)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(flgid が不正あるいは使用できない)	FLG4-2
2		E_PAR	パラメータエラー(waiptn が不正)	FLG4-4
3		E_PAR	パラメータエラー(wfmode が不正)	FLG4-6
4		E_PAR	パラメータエラー(p_flgptn が不正)	FLG4-8
5		E_PAR	パラメータエラー(tmout が不正)	FLG4-10
6		E_ILUSE	サービスコール不正使用(TA_WSGL 属性が指定されたイベントフラグで待ちタスクあり)	FLG4-26
7		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	FLG4-40
8		E_TMOUT	タイムアウト	FLG4-22
―	サービスコール処　　理	wfmode には、以下が指定できること。		―
9		TWF_ANDW		FLG4-28,56
10		TWF_ORW		FLG4-16,44
―		対象イベントフラグのビットパターンが waiptn と wfmode で指定される待ち条件を満たしている場合		―
11		自タスクを待ち状態とせずにサービスコールの処理を終了すること。		FLG4-16
12		flgptn には、この時の(待ち解除条件を満たした)ビットパターンを返すこと。		FLG4-17
13		属性に TA_CLR が指定されている場合には、イベントフラグのビットパターンのすべてのビットをクリアすること。		FLG4-54
―		対象イベントフラグのビットパターンが waiptn と wfmode で指定される解除条件を満たしていない場合		―
14		自タスクを待ち行列につなぎ、イベントフラグ待ち状態に移行させること。		FLG4-28
15		waiptn に 0 を指定した場合には、E_PAR エラーを返すこと。		FLG4-12
―		tmout の指定		―
16		正の値のタイムアウト時間が指定できること。		FLG4-22
17		TMO_POL が指定できること。		FLG4-16
18		TMO_FEVR が指定できること。		FLG4-40

（注）

No.2　E_PAR エラーを検出できる waiptn がシステムに存在しない場合は、テストを省略することができる。

No.4　E_PAR エラーを検出できる p_flgptn がシステムに存在しない場合は、テストを省略することができる。

No.5　E_PAR エラーを検出できる tmout がシステムに存在しない場合は、テストを省略することができる。

4.4.3 データキュー

CRE_DTQ：データキューの生成（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ1-S2
2		E_RSATR	予約属性(dtqatr が不正あるいは使用できない)	DTQ1-S4
3		E_PAR	パラメータエラー(dtqcnt が不正)	DTQ1-S6
4		E_OBJ	オブジェクト状態エラー(対象データキューが登録済)	DTQ1-S9
－	静的 API 処　　理	dtqatr には以下の指定ができ、オブジェクトが生成できること。		－
5		TA_TFIFO		DTQ2-16
6		TA_TPRI		DTQ2-56
7		データキュー領域に格納できるデータの個数が dtqcnt で指定できること。		DTQ2-4

(注)

No.3　E_PAR エラーを検出できる dtqcnt がシステムに存在しない場合は、テストを省略することができる。

snd_dtq：データキューへの送信

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ2-2
2		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	DTQ2-52
3	サービスコール 処　　理	dtqid で指定されるデータキューに、data で指定されるデータを送信できること。		DTQ2-8
4		送信されるデータは、送信側から受信側にコピーされること。		DTQ2-8
－		対象データキューで受信を待っているタスクがある場合		－
5		受信待ち行列の先頭のタスクに送信するデータを渡し、そのタスクの待ちを解除すること。		DTQ2-16
6		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ2-16
7		データキューから受信したデータとして VP_INT data の値を返すこと。		DTQ2-17
－		対象データキューで受信を待っているタスクがない場合		－
8		送信するデータをデータキューの末尾に入れること。		DTQ2-81
9		データキュー領域に空きがない場合には、自タスクを送信待ち行列につなぎ、データキューへの送信待ち状態に移行させること。		DTQ2-42
－		他のタスクがすでに送信待ち行列につながっている場合で		－
－		データキュー属性が TA_TFIFO の場合		－
10		自タスクを送信待ち行列の末尾につなぐこと。		DTQ2-47
－		データキュー属性 TA_TPRI の場合		－
11		自タスクを優先度順で送信待ち行列につなぐこと。		DTQ2-56
12		同じ優先度のタスクの中では、自タスクを最後につなぐこと。		DTQ2-72
－		データキューで送信されるデータ(１ワードのメッセージ)		－
13		整数値であること。		DTQ2-8
14		送信側と受信側で共有しているメモリ上に置かれたメッセージの先頭番地であること。		DTQ2-22

psnd_dtq：データキューへの送信(ポーリング)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ3-2
2		E_TMOUT	ポーリング失敗(対象データキューで受信を待っているタスクがなく、データキュー領域に空きがない場合には、E_TMOUT エラーを返すこと。)	DTQ3-4
3	サービスコール処　　理	dtqid で指定されるデータキューに、data で指定されるデータを送信できること。		DTQ3-8
4		送信されるデータは、送信側から受信側にコピーされること。		DTQ3-8
―		対象データキューで受信を待っているタスクがある場合		―
5		受信待ち行列の先頭のタスクに送信するデータを渡し、そのタスクの待ちを解除すること。		DTQ3-7
6		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ3-7
7		データキューから受信したデータとして data の値を返すこと。		DTQ3-8
―		データキューで送信されるデータ(１ワードのメッセージ)		―
8		整数値であること。		DTQ3-8
9		送信側と受信側で共有しているメモリ上に置かれたメッセージの先頭番地であること。		DTQ3-13

ipsnd_dtq：データキューへの送信(ポーリング)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ4-4
2		E_TMOUT	ポーリング失敗(サービスコールを遅延実行する場合で、E_TMOUT エラーを返すことを実装定義で省略できる。省略する場合は E_OK を返すこと。)	DTQ4-6
3	サービスコール処　　理	dtqid で指定されるデータキューに、data で指定されるデータを送信できること。		DTQ4-13
4		送信されるデータは、送信側から受信側にコピーされること。		DTQ4-13
―		対象データキューで受信を待っているタスクがある場合		―
5		受信待ち行列の先頭のタスクに送信するデータを渡し、そのタスクの待ちを解除すること。		DTQ4-12
6		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ4-12
7		データキューから受信したデータとして VP_INT data の値を返すこと。		DTQ4-12
―		データキューで送信されるデータ(１ワードのメッセージ)		―
8		整数値であること。		DTQ4-13
9		送信側と受信側で共有しているメモリ上に置かれたメッセージの先頭番地であること。		DTQ4-16

(注)

No.2　サービスコールを遅延実行する場合で、E_TMOUT エラーを省略することが製品マニュアルに記載されていれば、E_OK を返すことができる。

tsnd_dtq：データキューへの送信(タイムアウトあり)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ5-2
2		E_PAR	パラメータエラー(tmout が不正)	DTQ5-4
3		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	DTQ5-49
4		E_TMOUT	タイムアウト	DTQ5-45
5	サービスコール処　　理	dtqid で指定されるデータキューに、data で指定されるデータを送信できること。		DTQ5-10
6		送信されるデータは、送信側から受信側にコピーされること。		DTQ5-10
—		対象データキューで受信を待っているタスクがある場合		—
7		受信待ち行列の先頭のタスクに送信するデータを渡し、そのタスクの待ちを解除すること。		DTQ5-9
8		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ5-9
9		データキューから受信したデータとして data の値を返すこと。		DTQ5-10
—		対象データキューで受信を待っているタスクがない場合		—
10		送信するデータをデータキューの末尾に入れること。		DTQ5-22
11		データキュー領域に空きがない場合には、自タスクを送信待ち行列につなぎ、データキューへの送信待ち状態に移行させること。		DTQ5-16
—		他のタスクがすでに送信待ち行列につながっている場合で		—
—		データキュー属性が TA_TFIFO の場合		—
12		自タスクを送信待ち行列の末尾につなぐこと。		DTQ5-45
—		データキュー属性 TA_TPRI の場合		—
13		自タスクを優先度順で送信待ち行列につなぐこと。		DTQ5-67
14		同じ優先度のタスクの中では、自タスクを最後につなぐこと。		DTQ5-83
—		tmout の指定		—
15		正の値のタイムアウト時間が指定できること。		DTQ5-40
16		TMO_POL が指定できること。		DTQ5-14
17		TMO_FEVR が指定できること。		DTQ5-49
—		データキューで送信されるデータ(１ワードのメッセージ)		—
18		整数値であること。		DTQ5-10
19		送信側と受信側で共有しているメモリ上に置かれたメッセージの先頭番地であること。		DTQ5-19

(注)

No.2　E_PAR エラーを検出できる tmout がシステムに存在しない場合は、テストを省略することができる。

fsnd_dtq：データキューへの強制送信

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ6-2
2		E_ILUSE	サービスコール不正使用(対象データキューのデータキュー領域の容量が０)	DTQ6-4
—	サービスコール処　　理	対象データキューで受信を待っているタスクがある場合		—
3		受信待ち行列の先頭のタスクに送信するデータを渡し、そのタスクの待ちを解除すること。		DTQ6-14
4		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ6-14
5		データキューから受信したデータとして data の値を返すこと。		DTQ6-15
—		受信を待っているタスクがない場合		—
6		送信するデータをデータキューの末尾に入れること。		DTQ6-25
7		データキュー領域に空きがない場合には、データキューの先頭のデータを抹消し、データキュー領域に必要な領域を確保すること。		DTQ6-11

ifsnd_dtq：データキューへの強制送信

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ7-4
2		E_ILUSE	サービスコール不正使用(対象データキューのデータキュー領域の容量が０)	DTQ7-6
—	サービスコール処　　理	対象データキューで受信を待っているタスクがある場合		—
3		受信待ち行列の先頭のタスクに送信するデータを渡し、そのタスクの待ちを解除すること。		DTQ7-18
4		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ7-18
5		データキューから受信したデータとして VP_INT data の値を返すこと。		DTQ7-19
—		受信を待っているタスクがない場合		—
6		送信するデータをデータキューの末尾に入れること。		DTQ7-29
7		データキュー領域に空きがない場合には、データキューの先頭のデータを抹消し、データキュー領域に必要な領域を確保すること。		DTQ7-23

rcv_dtq：データキューからの受信

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ8-2
2		E_PAR	パラメータエラー(p_data が不正)	DTQ8-4
3		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	DTQ8-37
—	サービスコール処　理	対象データキューにデータが入っている場合		—
4		先頭のデータを取り出し、data に返すこと。		DTQ8-9
—		データキューで送信を待っているタスクがある場合		—
5		送信待ち行列の先頭のタスクが送信しようとしているデータをデータキューの末尾に入れ、そのタスクを待ち解除すること。		DTQ8-14,20
6		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ8-14
—		データが入っていない場合		—
—		対象データキューで送信を待っているタスクがある場合		—
7		送信待ち行列の先頭のタスクから、そのタスクが送信しようとしているデータを受け取り、そのタスクの待ちを解除すること。		DTQ8-26
8		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ8-26
9		data には、受け取ったデータ返すこと。		DTQ8-29
—		送信を待っているタスクがない場合		—
10		自タスクを受信待ち行列につなぎ、データキューからの受信待ち状態に移行させること。		DTQ8-32
11		他のタスクがすでに受信待ち行列につながっている場合には、自タスクを受信待ち行列の末尾につなぐこと。		DTQ8-37

（注）

No.2　E_PAR エラーを検出できる p_data がシステムに存在しない場合は、テストを省略することができる。

prcv_dtq：データキューからの受信(ポーリング)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ9-2
2		E_PAR	パラメータエラー(p_data が不正)	DTQ9-5
3		E_TMOUT	ポーリング失敗	DTQ9-7
—	サービスコール処　理	対象データキューにデータが入っている場合		—
4		先頭のデータを取り出し、data に返すこと。		DTQ9-15
—		データキューで送信を待っているタスクがある場合		
5		送信待ち行列の先頭のタスクが送信しようとしているデータをデータキューの末尾に入れ、そのタスクを待ち解除すること。		DTQ9-18,29
6		この時、待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ9-18

（注）

No.2　E_PAR エラーを検出できる p_data がシステムに存在しない場合は、テストを省略することができる。

trcv_dtq：データキューからの受信(タイムアウトあり)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ10-2
2		E_PAR	パラメータエラー(p_data が不正)	DTQ10-5
3		E_PAR	パラメータエラー(tmout が不正)	DTQ10-7
4		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	DTQ10-50
5		E_TMOUT	タイムアウト	DTQ10-46
－	サービスコール処　理	対象データキューにデータが入っている場合		－
6		先頭のデータを取り出し、data に返すこと。		DTQ10-12
－		データキューで送信を待っているタスクがある場合		－
7		送信待ち行列の先頭のタスクが送信しようとしているデータをデータキューの末尾に入れ、そのタスクを待ち解除すること。		DTQ10-19,25
8		この時、待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ10-19
－		データが入っていない場合		－
－		対象データキューで送信を待っているタスクがある場合		－
9		送信待ち行列の先頭のタスクから、そのタスクが送信しようとしているデータを受け取り、そのタスクの待ちを解除すること。		DTQ10-31
10		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ10-31
11		data には、受け取ったデータを返すこと。		DTQ10-34
－		送信を待っているタスクがない場合		－
12		自タスクを受信待ち行列につなぎ、データキューからの受信待ち状態に移行させること。		DTQ10-41
13		他のタスクがすでに受信待ち行列につながっている場合には、自タスクを受信待ち行列の末尾につなぐこと。		DTQ10-50
－		tmout の指定		－
14		正の値のタイムアウト時間が指定できること。		DTQ10-41
15		TMO_POL が指定できること。		DTQ10-11
16		TMO_FEVR が指定できること。		DTQ10-50

（注）

No.2　E_PAR エラーを検出できる p_data がシステムに存在しない場合は、テストを省略することができる。

No.3　E_PAR エラーを検出できる tmout がシステムに存在しない場合は、テストを省略することができる。

4.4.4　メールボックス

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	データ型	T_MSG	メールボックスのメッセージヘッダ	HED-80
2		T_MSG_PRI	メールボックスの優先度付きメッセージヘッダ	HED-81

CRE_MBX：メールボックスの生成（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(mbxid が不正あるいは使用できない)	MBX1-S2
2		E_RSATR	予約属性(mbxatr が不正あるいは使用できない)	MBX1-S4
3		E_PAR	パラメータエラー(maxmpri が不正)	MBX1-S6
4		E_OBJ	オブジェクト状態エラー(対象メールボックスが登録済)	MBX1-S9
―	静的 API 処　　理	mbxatr には以下の指定ができ、オブジェクトが生成できること。		－
5		TA_TFIFO|TA_MFIFO		MBX2-19
6		TA_TFIFO|TA_MPRI		MBX3-3
7		TA_TPRI|TA_MFIFO		MBX4-11
8		TA_TPRI|TA_MPRI		MBX6-18
9		maxmpri に 0 を指定することはできない。指定した場合には、E_PAR エラーを返すこと。		MBX1-S11
10		maxmpri には、メールボックスに送信されるメッセージの優先度の最大値が指定できること。		MBX3-24

snd_mbx：メールボックスへの送信

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(mbxid が不正あるいは使用できない)	MBX2-2
2		E_PAR	パラメータエラー(pk_msg が不正)	MBX2-5
3		E_PAR	パラメータエラー(メッセージパケット中のメッセージ優先度(msgpri)が不正)	MBX3-27
―	サービスコール 処　　理	対象メールボックスで受信を待っているタスクがある場合		―
4		受信待ち行列の先頭のタスクに pk_msg で指定されたメッセージパケットの先頭番地を渡し、そのタスクを待ち解除すること。		MBX2-54
5		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		MBX2-53
6		待ち解除されたタスクに対しては、メールボックスから受信したメッセージパケットの先頭番地として pk_msg の値を返すこと。		MBX2-54
―		受信を待っているタスクがない場合		―
―		メールボックス属性が TA_MFIFO		―
7			pk_msg を先頭番地とするメッセージパケットをメッセージキューの末尾に入れること。	MBX2-36,39,42
―		メールボックス属性が TA_MPRI		
8			メッセージパケットを優先度順でメッセージキューに入れること。	MBX3-15
9			同じ優先度のメッセージパケットの中では、新たに送信されたメッセージパケットを最後に入れること。	MBX3-9

rcv_mbx：メールボックスからの受信

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(mbxid が不正あるいは使用できない)	MBX2-7
2		E_PAR	パラメータエラー(ppk_msg が不正)	MBX2-10
3		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	MBX2-63
―	サービスコール処　理	メッセージキューにメッセージが入っている場合		―
4		先頭のメッセージパケットを取り出し、その先頭番地を pk_msg に返すこと。		MBX2-22
―		メッセージキューにメッセージが入っていない場合		―
5		自タスクを待ち行列につなぎ、メールボックスからの受信待ち状態に移行させること。		MBX2-53
―		他のタスクがすでに待ち行列につながっている場合		―
―		属性が TA_TFIFO の場合		―
6			自タスクを待ち行列の末尾につなぐこと。	MBX2-58
―		属性が TA_TPRI の場合		―
7			自タスクを優先度順で待ち行列につなぐこと。	MBX4-11
8			同じ優先度のタスクの中では、自タスクを最後につなぐこと。	MBX4-21

(注)

No.2　E_PAR エラーを検出できる ppk_msg がシステムに存在しない場合は、テストを省略することができる。

prcv_mbx：メールボックスからの受信(ポーリング)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(mbxid が不正あるいは使用できない)	MBX2-12
2		E_PAR	パラメータエラー(ppk_msg が不正)	MBX2-14
3		E_TMOUT	ポーリング失敗	MBX2-17
―	サービスコール処　理	メッセージキューにメッセージが入っている場合		―
4		先頭のメッセージパケットを取り出し、その先頭番地を pk_msg に返すこと。		MBX2-27

(注)

No.2　E_PAR エラーを検出できる ppk_msg がシステムに存在しない場合は、テストを省略することができる。

trcv_mbx：メールボックスからの受信(タイムアウトあり)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(mbxid が不正あるいは使用できない)	MBX5-2
2		E_PAR	パラメータエラー(ppk_msg が不正)	MBX5-5
3		E_PAR	パラメータエラー(tmout が不正)	MBX5-7
4		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	MBX5-36
5		E_TMOUT	タイムアウト	MBX5-30
―	サービスコール処　理	メッセージキューにメッセージが入っている場合		―
6		先頭のメッセージパケットを取り出し、その先頭番地を pk_msg に返すこと。		MBX5-12
―		メッセージキューにメッセージが入っていない場合		―
7		自タスクを待ち行列につなぎ、メールボックスからの受信待ち状態に移行させること。		MBX5-25
―		他のタスクがすでに待ち行列につながっている場合		―
―		属性が TA_TFIFO の場合		―
8		自タスクを待ち行列の末尾につなぐこと。		MBX5-36
―		属性が TA_TPRI の場合		―
9		自タスクを優先度順で待ち行列につなぐこと。		MBX6-12
10		同じ優先度のタスクの中では、自タスクを最後につなぐこと。		MBX6-24
―		tmout の指定		―
11		正の値のタイムアウト時間が指定できること。		MBX5-25
12		TMO_POL が指定できること。		MBX5-11
13		TMO_FEVR が指定できること。		MBX5-36

(注)

No.2　E_PAR エラーを検出できる ppk_msg がシステムに存在しない場合は、テストを省略することができる。

No.3　E_PAR エラーを検出できる tmout がシステムに存在しない場合は、テストを省略することができる。

4.6 メモリプール管理機能

4.6.1　固定長メモリプール

CRE_MPF：固定長メモリプールの生成（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(mpfid が不正あるいは使用できない)	MPF1-S2
2		E_RSATR	予約属性(mpfatr が不正あるいは使用できない)	MPF1-S4
3		E_PAR	パラメータエラー(blkcnt が 0)。	MPF1-S6
4		E_PAR	パラメータエラー(blksz が 0)。	MPF1-S8
5		E_OBJ	オブジェクト状態エラー(対象固定長メモリプールが登録済)	MPF1-S11
―	静的 API 処　　理	mpfatr には以下の指定ができ、オブジェクトが生成できること。		―
6		TA_TFIFO		MPF2-16
7		TA_TPRI		MPF3-2
8		blkcnt で指定する個数でオブジェクトが生成できること		MPF2-37
9		blksz で指定するバイト数でオブジェクトが生成できること		MPF2-37

get_mpf：固定長メモリブロックの獲得

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(mpfid が不正あるいは使用できない)	MPF2-2
2		E_PAR	パラメータエラー(p_blk が不正)	MPF2-7
3		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	MPF2-43
—	サービスコール処　理	空きメモリブロックがある場合		—
4		その内のいずれかを選んで獲得された状態とし、その先頭番地を blk に返すこと。		MPF2-25,26
—		空きメモリブロックがない場合		—
5		自タスクを待ち行列につなぎ、固定長メモリブロックの獲得待ち状態に移行させること。		MPF2-37
—		他のタスクがすでに待ち行列につながっている場合		—
—		属性が TA_TFIFO の場合		—
6		自タスクを待ち行列の末尾につなぐこと。		MPF2-40
—		属性が TA_TPRI の場合		—
7		自タスクを優先度順で待ち行列につなぐこと。		MPF3-13
8		同じ優先度のタスクの中では、自タスクを最後につなぐこと。		MPF3-21

(注)

No.2　E_PAR エラーを検出できる p_blk がシステムに存在しない場合は、テストを省略することができる。

pget_mpf：固定長メモリブロックの獲得(ポーリング)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(mpfid が不正あるいは使用できない)	MPF2-4
2		E_PAR	パラメータエラー(p_blk が不正)	MPF2-9
3		E_TMOUT	ポーリング失敗	MPF2-21
—	サービスコール処　理	空きメモリブロックがある場合		—
4		その内のいずれかを選んで獲得された状態とし、その先頭番地を blk に返すこと。		MPF2-16,17
—		空きメモリブロックがない場合		—
5		E_TMOUT エラーを返すこと。		MPF2-21

(注)

No.2　E_PAR エラーを検出できる p_blk がシステムに存在しない場合は、テストを省略することができる。

tget_mpf：固定長メモリブロックの獲得(タイムアウトあり)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(mpfid が不正あるいは使用できない)	MPF4-2
2		E_PAR	パラメータエラー(p_blk が不正)	MPF4-5
3		E_PAR	パラメータエラー(tmout が不正)	MPF4-7
4		T_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	MPF4-30
5		E_TMOUT	タイムアウト	MPF4-26
—	サービスコール処　理	空きメモリブロックがある場合		—
6		その内のいずれかを選んで獲得された状態とし、その先頭番地を blk に返すこと。		MPF4-9,10
—		空きメモリブロックがない場合		—
7		自タスクを待ち行列につなぎ、固定長メモリブロックの獲得待ち状態に移行させること。		MPF4-21
—		他のタスクがすでに待ち行列につながっている場合		—
—		属性が TA_TFIFO の場合		—
8		自タスクを待ち行列の末尾につなぐこと。		MPF4-30

—			属性が TA_TPRI の場合	—
9			自タスクを優先度順で待ち行列につなぐこと。	MPF5-13
10			同じ優先度のタスクの中では、自タスクを最後につなぐこと。	MPF5-23
—		tmout の指定		—
11			正の値のタイムアウト時間が指定できること。	MPF4-21
12			TMO_POL が指定できること。	MPF4-9
13			TMO_FEVR が指定できること。	MPF4-30

(注)

No.2　E_PAR エラーを検出できる p_blk がシステムに存在しない場合は、テストを省略することができる。

No.3　E_PAR エラーを検出できる tmout がシステムに存在しない場合は、テストを省略することができる。

rel_mpf：固定長メモリブロックの返却

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(mpfid が不正あるいは使用できない)	MPF2-11
2		E_PAR	パラメータエラー(異なるメモリプールへの返却)	MPF2-19
3		E_PAR	パラメータエラー(獲得したメモリブロックの先頭番地以外の返却)	MPF2-14
4	サービスコール処　理	mpfid で指定される固定長メモリプールに対して、blk を先頭番地とするメモリブロックを返却すること。		MPF2-41
—		メモリブロックの獲得を待っているタスクがある場合		—
5		返却したメモリブロックを待ち行列の先頭のタスクに獲得させ、そのタスクを待ち解除すること。		MPF2-49
6		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		MPF2-40
7		固定長メモリブロックから獲得したメモリブロックの先頭番地として blk の値を返すこと。		MPF2-41

4.7 時間管理機能

4.7.1　システム時刻管理

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	初期化	システム時刻は、システム初期化時に 0 とすること。		TIM-3
2	時刻の更新	アプリケーションが定められた周期で isig_tim を呼出し、システム時刻が更新されること。		TIM-22
3		システム時刻更新機能がカーネル内部にある場合は、isig_tim を呼び出さなくてもシステム時刻が更新されること。ただし、システム時刻更新方法が製品マニュアルに記載されていること。		UMA-29,30
—	システム時刻に依存して行う処　理	システム時刻に依存して行う処理は、以下の３つであること。		—
4		タイムアウト処理		MPF4-26
5		dly_tsk による時間経過待ち状態からの解除		SYN9-4
6		周期ハンドラの起動		CYC2-32
7	定　　数	TIC_NUME	タイムティックの周期の分子	HED-82
8		TIC_DENO	タイムティックの周期の分母	HED-83

set_tim：システム時刻の設定

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_PAR	パラメータエラー(p_systim が不正)	TIM-8
2		E_PAR	パラメータエラー(systim が不正)	TIM-11
3	サービスコール処　　理	現在のシステム時刻を、systim で指定される時刻に設定できること。		TIM-17

(注)

No.1　E_PAR エラーを検出できる p_systim がシステムに存在しない場合は、テストを省略することができる。

No.2　E_PAR エラーを検出できる systim がシステムに存在しない場合は、テストを省略することができる。

get_tim：システム時刻の参照

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_PAR	パラメータエラー(p_systim が不正)	TIM-6
2	サービスコール処　　理	現在のシステム時刻を呼び出し、systim に返すこと。		TIM-17

(注)

No.1　E_PAR エラーを検出できる p_systim がシステムに存在しない場合は、テストを省略することができる。

isig_tim：タイムティックの供給

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　　理	システム時刻を更新できること。	TIM-22
2		システム時刻を更新する機構をカーネル内部に持つ場合には、このサービスコールをサポートする必要はない。	UMA-29

4.7.2　周期ハンドラ

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	拡張情報	周期ハンドラを起動すべき時刻になると、その周期ハンドラの拡張情報(exinf)をパラメータとして、周期ハンドラを起動すること。	CYC2-32
2	停止状態	周期ハンドラを起動すべき時刻になっても周期ハンドラを起動せず、次に起動すべき時刻の決定のみを行うこと。	CYC2-10

CRE_CYC：周期ハンドラの生成（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_RSATR	予約属性(cycatr が不正あるいは使用できない)	CYC1-S2
2		E_PAR	パラメータエラー(cychdr が不正)	CYC1-S4
3		E_PAR	パラメータエラー(cyctim が 0)	CYC1-S6
4		E_PAR	パラメータエラー(cycphs が不正)	CYC1-S8
―	静的 API 処　　理	TA_STA 指定の場合		―
5		周期ハンドラを生成した後に、周期ハンドラを動作している状態とすること。		CYC2-11
―		TA_STA 指定なしの場合		―
6		周期ハンドラを動作していない状態とすること。		CYC2-10
7		周期ハンドラを最初に起動すべき時刻は、これらのサービスコールが呼び出された時刻(静的 API の場合にはシステム初期化の時刻)に、指定された起動位相を加えた時刻とすること。		CYC2-11

(注)

No.2　E_PAR エラーを検出できる cychdr がシステムに存在しない場合は、テストを省略することができる。

sta_cyc：周期ハンドラの動作開始

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号 (cycid が不正あるいは使用できない)	CYC2-3
2	サービスコール処　　理	cycid で指定される周期ハンドラを、動作している状態に移行させること。		CYC2-22
3		対象周期ハンドラに、周期ハンドラ属性に TA_PHS が指定されていないで動作している状態の周期ハンドラが指定された場合には、周期ハンドラを次に起動すべき時刻の再設定のみを行うこと。		CYC2-27

stp_cyc：周期ハンドラの動作停止

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号 (cycid が不正あるいは使用できない)	CYC2-5
2	サービスコール処　　理	cycid で指定される周期ハンドラを、動作していない状態に移行させること。		CYC2-17
3		動作していない状態の周期ハンドラが指定された場合には、何もしないこと。		CYC2-7

4.8　システム状態管理機能

rot_rdq：タスクの優先順位の回転

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_PAR	パラメータエラー(tskpri が不正)	SYS1-2
2	サービスコール処　　理	対象優先度を持った実行できる状態のタスクの中で、最も高い優先順位を持つタスクを、同じ優先度を持つタスクの中で最低の優先順位とすること。		SYS1-66
3		TPRI_SELF が指定されると、自タスクの優先度とすること。		SYS1-50

irot_rdq：タスクの優先順位の回転

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_PAR	パラメータエラー(tskpri が不正)	SYS2-3
2	サービスコール処　　理	対象優先度を持った実行できる状態のタスクの中で、最も高い優先順位を持つタスクを、同じ優先度を持つタスクの中で最低の優先順位とすること。		SYS2-31
3		非タスクコンテキストから TPRI_SELF の呼び出しが行われた場合には、E_PAR エラーを返すこと。		SYS2-5

get_tid：実行状態のタスク ID の参照

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処理	実行状態のタスクの ID 番号を参照し、tskid に返すこと。	SYS1-5

iget_tid：実行状態のタスク ID の参照

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処理	実行状態のタスクの ID 番号を参照し、tskid に返すこと。	SYS2-7
2		実行状態のタスクがない時は、tskid に TSK_NONE を返すこと。	SYS2-34

loc_cpu：CPU ロック状態への移行

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　　理	CPU ロック状態に移行すること。	SYS1-11
2		CPU ロック状態で呼び出された場合には何もしないこと。	SYS1-17

iloc_cpu：CPU ロック状態への移行

No.	区　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　理	CPU ロック状態に移行すること。	SYS2-11
2		CPU ロック状態で呼び出された場合には何もしないこと。	SYS2-15

unl_cpu：CPU ロック状態の解除

No.	区　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　理	CPU ロック解除状態に移行すること。	SYS1-21
2		CPU ロック解除状態で呼び出された場合には何もしないこと。	SYS1-25

iunl_cpu：CPU ロック状態の解除

No.	区　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　理	CPU ロック状態解除に移行すること。	SYS2-19
2		CPU ロック解除状態で呼び出された場合には何もしないこと。	SYS2-23

dis_dsp：ディスパッチの禁止

No.	区　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　理	ディスパッチ禁止状態に移行すること。	SYS1-31
2		ディスパッチ禁止状態で呼び出された場合には何もしないこと。	SYS1-37

ena_dsp：ディスパッチの許可

No.	区　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　理	ディスパッチ許可状態に移行すること。	SYS1-41
2		ディスパッチ許可状態で呼び出された場合には何もしないこと。	SYS1-45

sns_ctx：コンテキストの参照

No.	区　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　理	タスクコンテキストから呼び出された場合に FALSE を返すこと。	SYS1-27
2		非タスクコンテキストから呼び出された場合に TRUE を返すこと。	SYS2-25

sns_loc：CPU ロック状態の参照

No.	区　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　理	システムが、CPU ロック状態の場合に TRUE を返すこと。	SYS1-11
2		システムが、CPU ロック解除状態の場合に FALSE を返すこと。	SYS1-21

sns_dsp：ディスパッチ禁止状態の参照

No.	区　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　理	システムが、ディスパッチ禁止状態の場合に TRUE を返すこと。	SYS1-31
2		システムが、ディスパッチ許可状態の場合に FALSE を返すこと。	SYS1-41

sns_dpn：ディスパッチ保留状態の参照

No.	区　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
―	サービスコール処　理	システムが、以下に示すディスパッチ保留状態の場合には TRUE を返すこと。	―
1		ディスパッチャよりも優先順位の高い処理が実行されている間	SYS2-36
2		CPU ロック状態の間	SYS1-13
3		ディスパッチ禁止状態の間	SYS1-33
4		システムが、ディスパッチ保留状態でない場合に FALSE を返すこと。	SYS1-7

4.9 割込み管理機能

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	データ型	INHNO	割込みハンドラ番号	HED-84
2		INTNO	割込み番号	HED-85
3	記述形式	製品マニュアル記載の通りに割込みハンドラを記述し、期待する動作を行うこと。		UMA-31
4	拡張情報	割込みサービスルーチンを起動する際、その割込みサービスルーチンの拡張情報(exinf)をパラメータとして渡すこと。		ISR1-2
5	CPU ロック解除	割込みハンドラ内で CPU ロック解除状態にする方法が、製品マニュアルに明示されていること。		UMA-32
6	リターン	CPU ロック解除した後に、割込みハンドラから正しくリターンするための方法が製品マニュアルに明示されていること。		UMA-33

DEF_INH：割込みハンドラの定義（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_RSATR	予約属性(inhatr が不正あるいは使用できない)	INH1-S2
2		E_PAR	パラメータエラー(inhno が不正)	INH1-S4
3		E_PAR	パラメータエラー(inthdr が不正)	INH1-S6
4	静的 API 処　　理	inhno で指定される割込みハンドラ番号に対して、定義情報に基づいて割込みハンドラを定義できること。		INH1-2
5		inhno の具体的な意味が実装で定義されていること。		UMA-34

(注)

No.3　E_PAR エラーを検出できる inthdr がシステムに存在しない場合は、テストを省略することができる。

ATT_ISR：割込みサービスルーチンの追加（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_RSATR	予約属性(isratr が不正あるいは使用できない)	ISR1-S2
2		E_PAR	パラメータエラー(intno が不正)	ISR1-S4
3		E_PAR	パラメータエラー(isr が不正)	ISR1-S6
4	静的 API 処　　理	割込みサービスルーチンを起動する時に、パラメータとして exinf を渡すこと。		ISR1-2

(注)

No.3　E_PAR エラーを検出できる isr がシステムに存在しない場合は、テストを省略することができる。

4.11 システム構成管理機能

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	データ型	EXCNO	CPU 例外ハンドラ番号	HED-86
2	CPU 例外ハンドラの記述形式	製品マニュアル記載の通りに CPU 例外ハンドラを記述し、期待する動作を行うこと。		UMA-35
3	拡張情報	初期化ルーチンを起動する際には、その初期化ルーチンの拡張情報(exinf)をパラメータとして渡すこと。		INI1-2

DEF_EXC：CPU 例外ハンドラの定義（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_RSATR	予約属性(excatr が不正あるいは使用できない)	EXC1-S2
2		E_PAR	パラメータエラー(excno が不正)	EXC1-S4
3		E_PAR	パラメータエラー(exchdr が不正)	EXC1-S6
4	静的 API 処　　理	excno の具体的な意味は実装定義であるが、一般的な実装では、プロセッサで区別される例外の種類に対応する。。		UMA-36
5		すでに CPU 例外ハンドラが定義されている CPU 例外ハンドラ番号に対して、再度 CPU 例外ハンドラを定義した場合には、以前の定義を解除し、新しい定義に置き換えること。		EXC1-2

(注)

No.3　E_PAR エラーを検出できる exchdr がシステムに存在しない場合は、テストを省略することができる。

ATT_INI：初期化ルーチンの追加（静的 API）

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	静的 API 処理	追加された初期化ルーチンは、システム初期化時に、静的 API の処理の一環として実行すること。	INI1-1

第3章　スタンダードプロファイルのテスト手順

第１節　プログラムによる確認の見方

1.1　テストに必要なシステム条件

スタンダードプロファイルのテスト手順を実施する場合のシステム条件を示す。

(1) ソフトウェアから何らかの CPU 例外が発生できること。
(2) ソフトウェアから何らか割込みが発生できること。
(3) 1msec 周期で isig_tim サービスコールを呼び出し、タイマを更新すること。

1.2　静的 API のテスト手順の見方

以下に、静的 API のテストを行う場合のテスト手順の見方を、タスク管理機能のテスト手順その１で説明する。また、「4.1　タスク管理機能」などタイトルの前の番号は、μITRON4.0 仕様書に記載されてあるものと一致させた。

TSK1：タスク管理機能のテスト手順　その１

No	テ ス ト 手 順 内 容	テスト項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})	
S3	CRE_TSK(ERR_TASK_ID,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S4	*E_ID を検出できること*	CRE_TSK-1
S5	CRE_TSK(TASK_ID3,{ERR_TSKATR,TASK1,ITSKPRI_1,STKSZ,NULL})	
S6	*E_RSATR を検出できること*	CRE_TSK-2
S7	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_1,ERR_TASK,ITSKPRI_3,STKSZ,NULL})	
S8	*E_PAR を検出できること*	CRE_TSK-3
S9	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_1,TASK1,ERR_ITSKPRI,STKSZ,NULL})	
S10	*E_PAR を検出できること*	CRE_TSK-4
S11	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_1,TASK3,ITSKPRI_3,ERR_STKSZ,NULL})	
S12	*E_PAR を検出できること*	CRE_TSK-5
S13	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S14	*E_OBJ を検出すること*	CRE_TSK-6

(1) No.
静的 API の記述順序を示す。

(2) テスト手順内容
システムコンフィギュレーションファイルに記述する内容である。
「TASK_ID」等は、μITRON 検定仕様書で使用する定数一覧でデータ型とその値を定義している。
網掛けされてあるものは、実装定義でエラーの検出を省略できることを示す。ただし、これらのエラーコードの検出を省略する場合は、省略する旨が製品マニュアルに記載されていること。
No.S4 でエラーを検出するとコンフィギュレータが停止するシステムの場合は、No.S4 のテストが完了したら、No.S3 と No.S4 をエディタ等で削除し、再度コンフィギュレータを実行する。エラーを検出して停止したら当該行を削除する作業を最後まで繰り返す。

(3) テスト項目参照
テスト手順で確認すべき事項が、テスト項目のどの部分に記載されているかを示す。

1.3 静的 API 以外のテスト手順の見方

以下に、タスク管理機能のテスト手順　その２を例にテスト手順の見方を説明する。

TSK2：タスク管理機能のテスト手順　その２

No	テ ス ト 手 順 内 容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK1,ITSKPRI_2,STKSZ,NULL})		
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2:READY>**	－
1		ercd = act_tsk(ERR_TASK_ID)	
2		*MERCD(ercd) ==E_ID*	act_tsk_1
3		ercd = act_tsk(TSK_SELF)	
4		*MERCD(ercd) == E_OK*	act_tsk-7
5		act_cnt = can_act(TSK_SELF)	
6		*actcnt == 1*	act_tsk-7 can_act-4
7		act_cnt = can_act(TSK_SELF)	
8		*actcnt == 0*	can_act-2
9		act_cnt = can_act(ERR_TASK_ID)	
10		*actcnt == E_ID*	can_act-1
11		ercd = act_tsk(TASK_ID1)	
12	*exinf == EXINF_1*		act_tsk-3,5
13	ercd = act_tsk(TASK_ID2)		
14	*MERCD(ercd) == E_OK*		
15	actcnt = can_act(TASK_ID2)		

(1) No.
プログラムの実行順序を示す。必ずこの順序で実行されていることを確認しなければならない。
番号の前にＳが付加されているものは、静的 API の記述順序を示す。
番号のみの場合は、タスクやハンドラの実行順序であることを示す。したがって、タスクやハンドラは、必ず昇順で実行されることを確認すること。
「－」は、補足的説明を加えている行を示し、プログラム化を行う必要はない。

(2) テスト手順内容
静的 API でテスト環境を整え、No.1 から順次実行していく。また、No.1 で定数として使用している「ERR_TASK_ID」は、μITRON 検定仕様書で使用する定数一覧でデータ型とその値を定義している。さらに、No.5 で変数と使用している「actcnt」は、μITRON 検定仕様書で使用する変数一覧でデータ型を定義している。
*ercd == E_OK*など、*イタリック体*で記述している行では、値を比較し必ず一致しなければならないことを示す。
テスト項目と同様に、エラーコードに網掛けされてあるものは、実装定義でエラーの検出を省略できることを示す。ただし、これらのエラーコードの検出を省略する場合は、省略する旨が製品マニュアルに記載されていること。したがって、製品マニュアルに省略する旨が記載されている場合に限り、当該テストを省略できる。
No.12 で示すように、記述位置が変わっている場合は、状態が遷移しなければならないことを示す。
カーネル構成定数によって結果が異なる場合がある。
例えば、
　TMAX_ACTCNT == 1　：*actcnt == 1*
　TMAX_ACTCNT ＞ 1　：*actcnt == 2*　　などである。
定義している値によって、どちらかを選択すること。
静的 API 記述の次の行の**<TASK_ID1:DORMANT>**等、太字で記述しているものはタスクやハンドラの初期状態を示している。

(3) テスト項目参照
テスト手順で確認すべき事項が、テスト項目のどの部分に記載されているかを示す。

1.4　タスク・ハンドラの記述形式

テスト手順は、テスト項目確認のための具体的手段を記載している。テストするためのプログラムがより自然に書けるよう、C 言語を意識した記載とした。
タスクおよびハンドラは、μITRON4.0 仕様書の規定の沿って、以下の通り記述する。

(1)タスクの C 言語による記述形式

```
void task(VP_INT exinf)
{
        タスク本体
        ext_tsk( )
}
```

(2)タスクの C 言語による記述形式(return で終了する場合)

```
void task(VP_INT exinf)
{
        タスク本体
        return
}
```

(3)タスク例外処理ルーチンの C 言語による記述形式

```
void texrtn(TEXPTN texptn, VP_INT exinf)
{
        タスク例外処理ルーチン本体
}
```

(4)周期ハンドラの C 言語による記述形式

```
void cychdr(VP_INT exinf)
{
        周期ハンドラ本体
}
```

(5)割込みサービスルーチンの C 言語による記述形式

```
void isr(VP_INT exinf)
{
        割込みサービスルーチン本体
}
```

(6)初期化ルーチンの C 言語による記述形式

```
void inirtn(VP_INT exinf)
{
        初期化ルーチン本体
}
```

1.5 スタンダードプロファイルのテスト手順で使用する定数一覧

スタンダードプロファイルのテスト手順では、以下に示す定数を定義しテストを実施すること。

ID		
ID	TASK_ID1	コンフィギュレータで ID の自動割付けを行う
ID	TASK_ID1_1	
ID	TASK_ID1_2	
ID	TASK_ID2	
ID	TASK_ID2_1	
ID	TASK_ID2_2	
ID	TASK_ID3	
ID	TASK_ID4	
ID	TASK_ID5~TASK_ID16	
ID	SEM_ID1~SEM_ID255	
ID	FLG_ID1	
ID	FLG_ID2	
ID	FLG_ID3	
ID	FLG_ID4	
ID	FLG_ID5	
ID	FLG_ID6	
ID	FLG_ID7	
ID	FLG_ID8	
ID	DTQ_ID1	
ID	DTQ_ID2	
ID	DTQ_ID3	
ID	DTQ_ID4	
ID	MBX_ID1	
ID	MBX_ID1_2	
ID	MPF_ID1	
ID	MPF_ID2	
ID	CYC_ID1	
ID	CYC_ID2	
タスク例外要因パターン		
TEXPTN	RASPTN_1	0x0001
TEXPTN	RASPTN_2	0x0002
TEXPTN	RASPTN_4	0x0003
TEXPTN	RASPTN_1111	0x1111
TEXPTN	RASPTN_55AA	0x55AA
TEXPTN	RASPTN_AA55	0xAA55
TEXPTN	RASPTN_AAAA	0xAAAA
TEXPTN	RASPTN_AAAE	0xAAAE
TEXPTN	RASPTN_FFFF	0xFFFF
拡張情報		
VP_INT	EXINF_1	(VP_INT)0x00000001
VP_INT	EXINF_1_1	(VP_INT)0x00000011
VP_INT	EXINF_1_2	(VP_INT)0x00000012
VP_INT	EXINF_2	(VP_INT)0x00000002
VP_INT	EXINF_2_1	(VP_INT)0x00000021
VP_INT	EXINF_2_2	(VP_INT)0x00000022
VP_INT	EXINF_3	(VP_INT)0x00000003
VP_INT	EXINF_4	(VP_INT)0x00000004
VP_INT	EXINF_CYC1	(VP_INT)0x00000001
VP_INT	EXINF_CYC2	(VP_INT)0x00000002
初期値		
FLGPTN	IFLGPTN_0	0x0000
FLGPTN	IFLGPTN_1	0x0001
FLGPTN	IFLGPTN_2	0x0002
FLGPTN	IFLGPTN_3	0x0003
FLGPTN	IFLGPTN_4	0x0004
FLGPTN	IFLGPTN_5	0x0005
FLGPTN	IFLGPTN_6	0x0006
FLGPTN	IFLGPTN_7	0x0007
FLGPTN	IFLGPTN_8	0x0008
UINT	ISEMCNT_0	0
UINT	ISEMCNT_1	1
UINT	ISEMCNT_2	2
UINT	ISEMCNT_FFFE	0xfffe

起動番地		
FP	TASK1	TASK_ID1 起動番地
FP	TASK1_1	TASK_ID1_1 起動番地
FP	TASK1_2	TASK_ID1_2 起動番地
FP	TASK2	TASK_ID2 起動番地
FP	TASK2_1	TASK_ID2_1 起動番地
FP	TASK2_2	TASK_ID2_2 起動番地
FP	TASK3	TASK_ID3 起動番地
FP	TASK4	TASK_ID4 起動番地
FP	TASK5～TASK16	TASK_ID5～TASK_ID16 起動番地
FP	TEXRTN_1	タスク例外処理ルーチン起動番地
FP	TEXRTN_2	タスク例外処理ルーチン起動番地
FP	INTHDR_1	割込みハンドラ起動番地
FP	ISR_ADR_0	0
FP	ISR_1	割込みサービスルーチン起動番地
FP	EXCHDR_1	CPU 例外ハンドラ起動番地
FP	EXCHDR_2	CPU 例外ハンドラ起動番地
FP	INIRTN_1	初期化ルーチン起動番地
FP	INIRTN_2	初期化ルーチン起動番地
FP	CYCHDR_1	周期ハンドラ起動番地
FP	CYCHDR_2	周期ハンドラ起動番地
INHNO	INHNO	何らかの割込みハンドラを実装毎に定義する
INHNO	INHNO_1	実装で定義
EXCNO	EXCNO_1	実装で定義
INTNO	INTNO_1	実装で定義
優先度		
PRI	ITSKPRI_1~ITSKPRI_16	0x0001～0x0010
PRI	TSKPRI_3	0x0003
PRI	TSKPRI_4	0x0004
PRI	MSGPRI_1	0x0001
PRI	MSGPRI_2	0x0002
PRI	MSGPRI_3	0x0003
PRI	MAXMPRI_0	0x00000000
PRI	MAXMPRI_3	0x00000003
PRI	MAXMPRI_5	0x00000005
PRI	MAXMPRI_16	0x00000010
属性		
ATR	INHATR	実装毎に値を定義する
ATR	INHATR_1	実装毎に値を定義する
ATR	EXCATR	実装毎に値を定義する
ATR	INIATR	実装毎に値を定義する
サイズ		
SIZE	STKSZ	0x0100
SIZE	BLKSZ_0	0
SIZE	BLKSZ_10	10
SIZE	BLKSZ_256	256
タイムアウト値		
TMO	TMO_1	1
TMO	TMO_2	2
TMO	TMO_10	10
TMO	TMO_100	100
TMO	TMO_200	200
RELTIM	DLYTIM_2	2
RELTIM	DLYTIM_5	5
RELTIM	DLYTIM_10	10
RELTIM	DLYTIM_20	20
RELTIM	DLYTIM_50	50
RELTIM	DLYTIM_100	100
RELTIM	DLYTIM_300	300
RELTIM	DLYTIM_500	500
RELTIM	DLYTIM_1000	1000
SYSTIM	SYSTIM_0	0
SYSTIM	SYSTIM_1000	1000

周期ハンドラ		
RELTIM	CYCTIM_0	0
RELTIM	CYCTIM_10	10
RELTIM	CYCTIM_55	55
RELTIM	CYCPHS_0	0
RELTIM	CYCPHS_5	5
最　大　値		
UINT	MAXSEM_1	1
UINT	MAXSEM_2	2
UINT	MAXSEM_FFFF	0xffff
フラグパターン		
FLGPTN	SETPTN_1	0x0001
FLGPTN	CLRPTN_0000	0x0000
FLGPTN	CLRPTN_1	0x0001
FLGPTN	SETPTN_0001	0x0001
FLGPTN	SETPTN_0002	0x0002
FLGPTN	SETPTN_0005	0x0005
FLGPTN	SETPTN_0055	0x0055
FLGPTN	SETPTN_0500	0x0500
FLGPTN	SETPTN_0505	0x0505
FLGPTN	SETPTN_5500	0x5500
FLGPTN	SETPTN_1111	0x1111
FLGPTN	SETPTN_5555	0x5555
FLGPTN	SETPTN_8000	0x8000
FLGPTN	SETPTN_AAAA	0xAAAA
FLGPTN	WAIPTN_0000	0x0000
FLGPTN	WAIPTN_1	0x0001
FLGPTN	WAIPTN_0001	0x0001
FLGPTN	WAIPTN_0002	0x0002
FLGPTN	WAIPTN_0020	0x0020
FLGPTN	WAIPTN_0050	0x0050
FLGPTN	WAIPTN_0055	0x0055
FLGPTN	WAIPTN_0505	0x0505
FLGPTN	WAIPTN_5500	0x5500
FLGPTN	WAIPTN_5555	0x5555
FLGPTN	WAIPTN_8000	0x8000
FLGPTN	WAIPTN_AAAA	0xAAAA
FLGPTN	FLGPTN_0055	0x0055
FLGPTN	FLGPTN_5555	0x5555
そ　の　他		
VP_INT	DATA_00	(VP_INT)0x00000000
VP_INT	DATA_01	(VP_INT)0x00000001
VP_INT	DATA_02	(VP_INT)0x00000002
VP_INT	DATA_03	(VP_INT)0x00000003
VP_INT	DATA_04	(VP_INT)0x00000004
VP_INT	DATA_05	(VP_INT)0x00000005
VP_INT	DATA_11	(VP_INT)0x00000011
VP_INT	DATA_22	(VP_INT)0x00000022
VP_INT	DATA_33	(VP_INT)0x00000033
VP_INT	DATA_44	(VP_INT)0x00000044
VP_INT	DATA_55	(VP_INT)0x00000055
VP_INT	DATA_66	(VP_INT)0x00000066
VP_INT	DATA_77	(VP_INT)0x00000077
VP_INT	DATA_88	(VP_INT)0x00000088
VP_INT	DATA_99	(VP_INT)0x00000099
VP_INT	DATA_AA	(VP_INT)0x000000AA
VP_INT	DATA_CC	(VP_INT)0x000000CC
VP_INT	DATA_DD	(VP_INT)0x000000DD
VP_INT	DATA_EE	(VP_INT)0x000000EE
VP_INT	DATA_FF	(VP_INT)0x000000FF
UW	MEMORY	共有メモリアドレス
UW	SYSTIM_50	50

カウンタ		
UINT	DTQCNT_0	0x00000000
UINT	DTQCNT_1	0x00000001
UINT	DTQCNT_2	0x00000002
UINT	BLKCNT_0	0
UINT	BLKCNT_1	1
UINT	BLKCNT_10	10
UINT	BLKCNT_2048	2048
エラー定数		
FP	ERR_TASK	E_PARエラーを返すアドレスを実装毎に定義する
ID	ERR_TASK_ID	-1
ATR	ERR_TSKATR	0xff
PRI	ERR_ITSKPRI	-1
PRI	ERR_TSKPRI	-2
SIZE	ERR_STKSZ	E_PARエラーを返す値を実装毎に定義する
TMO	ERR_TMO	-2
RELTIM	ERR_DLYTIM	E_PARエラーを返す値を実装毎に定義する
ATR	ERR_TEXATR	0xff
FP	ERR_TEXRTN	0
FP	ERR_RASPTN	0
INHNO	ERR_INHNO	E_PARエラーを返す値を実装毎に定義する
ATR	ERR_INHATR	E_PARエラーを返す値を実装毎に定義する
ATR	ERR_ISRATR	2
INHNO	ERR_INTNO	未サポート割込み番号
EXCNO	ERR_EXCNO	E_PARエラーを返す値を実装毎に定義する
ATR	ERR_EXCATR	E_PARエラーを返す値を実装毎に定義する
ATR	ERR_SEMATR	0x55
ID	ERR_SEM_ID	-2
UINT	MAXSEM_OVER	E_PARエラーを返す値を実装毎に定義する
ID	ERR_FLG_ID	-3
ATR	ERR_FLGATR	0xaa
FLGPTN	ERR_IFLGPTN	E_PARエラーを返す値を実装毎に定義する
FLGPTN	ERR_SETPTN	0xffffffff
FLGPTN	ERR_CLRPTN	0xffffffff
FLGPTN	ERR_WAIPTN	0xffffffff
MODE	ERR_WFMODE	0x02
FP	ERR_POINTER	E_PARエラーを返すポインタ値を実装毎に定義する
ID	ERR_DTQID	-2
ATR	ERR_DTQATR	0xaa
UINT	ERR_DTQCNT	0xffffffff
ID	ERR_MBXID	-1
ATR	ERR_MBXATR	0xbb
PRI	ERR_MAXMPRI	0xffffffff
PRI	ERR_MSGPRI	-1
ID	ERR_MPFID	-3
ATR	ERR_MPFATR	0xcc
ID	ERR_CYCID	0
ATR	ERR_CYCATR	0xdd
RELTIM	ERR_CYCPHS	E_PARエラーを返す値を実装毎に定義する
FP	ERR_INTHDR	E_PARエラーを返す値を実装毎に定義する
FP	ERR_EXCHDR	E_PARエラーを返す値を実装毎に定義する
FP	ERR_CYCHDR	E_PARエラーを返す値を実装毎に定義する

1.6 スタンダードプロファイルのテスト手順で使用する変数一覧

スタンダードプロファイルのテスト手順では、以下に示す変数を定義しテストを実施すること。

型	変数名	型	変数名	型	変数名
T_MSG_PRI	msg1	T_MSG_PRI	msg2	T_MSG_PRI	msg21
T_MSG_PRI	msg22	T_MSG_PRI	msg3	T_MSG_PRI	msg8
T_MSG_PRI	msg16	T_MSG_PRI	*pk_msg	T_MSG_PRI	**ppk_msg
ER	ercd	VP_INT	exinf	BOOL	state
ER_UINT	actcnt	PRI	tskpri	ID	*p_tskid
PRI	*p_tskpri	ER_UNIT	wupcnt	FLGPTN	*p_flgptn
TEXPTN	texptn	FLGPTN	flgptn	VP_INT	*p_data
VP_INT	data	VP	*p_blk	VP	blk
SYSTIM	systim	SYSTIM	systim1	STSTIM	systim2
UW	cyc_work1	UW	cyc_work2	UW	work
B	b1	H	h1	W	w1
UB	ub1	UH	uh1	UW	uw1
VB	vb1	VH	vh1	VW	vw1
VP	vp	FP	fp	INT	int1
UINT	uint1	BOOL	bool1	FN	fn1
ER	er1	ID	id1	ATR	atr1
STAT	stat1	MODE	mode1	PRI	pri1
SIZE	size1	TMO	tmo1	RELTIM	reltim1
VP_INT	vp_int1	ER_BOOL	er_bool1	ER_ID	er_id1
ER_UINT	er_unit1				

第2節　プログラムによる確認の手順

COM1：ITRON仕様共通規定のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照

1	#include　<kernel.h>
	#include　<kernel.h>
	B　c;
	main(　)
	{
	B　a=1,b=2;
	
	　　c = a + b;
	}
	
	※ 上記ファイルをコンパイルして、エラーが発生しないことを確認する。
		APIの構成要素-5

COM2：ITRON仕様共通規定のテスト手順　その2

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID16,{TA_HLNG|TA_ACT,EXINF_16,TASK16,ITSKPRI_16,STKSZ,NULL})	
S2	CRE_TSK(TASK_ID15,{TA_HLNG|TA_ACT,EXINF_15,TASK15,ITSKPRI_15,STKSZ,NULL})	
S3	CRE_TSK(TASK_ID14,{TA_HLNG|TA_ACT,EXINF_14,TASK14,ITSKPRI_14,STKSZ,NULL})	
S4	CRE_TSK(TASK_ID13,{TA_HLNG|TA_ACT,EXINF_13,TASK13,ITSKPRI_13,STKSZ,NULL})	
S5	CRE_TSK(TASK_ID12,{TA_HLNG|TA_ACT,EXINF_12,TASK12,ITSKPRI_12,STKSZ,NULL})	
S6	CRE_TSK(TASK_ID11,{TA_HLNG|TA_ACT,EXINF_11,TASK11,ITSKPRI_11,STKSZ,NULL})	
S7	CRE_TSK(TASK_ID10,{TA_HLNG|TA_ACT,EXINF_10,TASK10,ITSKPRI_10,STKSZ,NULL})	
S8	CRE_TSK(TASK_ID9,{TA_HLNG|TA_ACT,EXINF_9,TASK9,ITSKPRI_9,STKSZ,NULL})	
S9	CRE_TSK(TASK_ID8,{TA_HLNG|TA_ACT,EXINF_8,TASK8,ITSKPRI_8,STKSZ,NULL})	
S10	CRE_TSK(TASK_ID7,{TA_HLNG|TA_ACT,EXINF_7,TASK7,ITSKPRI_7,STKSZ,NULL})	
S11	CRE_TSK(TASK_ID6,{TA_HLNG|TA_ACT,EXINF_6,TASK6,ITSKPRI_6,STKSZ,NULL})	
S12	CRE_TSK(TASK_ID5,{TA_HLNG|TA_ACT,EXINF_5,TASK5,ITSKPRI_5,STKSZ,NULL})	
S13	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_4,STKSZ,NULL})	
S14	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})	
S15	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})	
S16	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S17	CRE_MBX(MBX_ID1,{TA_TPRI|TA_MPRI,MAXMPRI_16,NULL})	

―	**<TASK_ID1:READY>**	**<TASK_ID2～TASK_ID15: READY> TASK_ID2　・・・・・・　TASK_ID15**	**<TASK_ID16: READY>**	―
1	メッセージ優先度付きで送信するメッセージmsg1,msg8,msg16を用意する			
2	msg16のmsgpriにMSGPRI_16を設定			
3	ercd = snd_mbx(MBX_ID1,&msg16)			
4	*MERCD(ercd) == E_OK*			
5	msg8のmsgpriにMSGPRI_8を設定			
6	ercd = snd_mbx(MBX_ID1,&msg8)			
7	*MERCD(ercd) == E_OK*			
8	msg1のmsgpriにMSGPRI_1を設定			
9	ercd = snd_mbx(MBX_ID1,&msg1)			
10	*MERCD(ercd) == E_OK*			
11	ext_tsk(　)			
12 〉 25		ext_tsk(　) ・ ・ ・ ・ ・ ext_tsk(　)		
26			ercd = prcv_mbx(MBX_ID1,&pk_msg)	優先度-1
27			*MERCD(ercd) == E_OK*	
28			*msg1を受信*	
29			ercd = prcv_mbx(MBX_ID1,&pk_msg)	
30			*MERCD(ercd) == E_OK*	
31			*msg8を受信*	
32			ercd = prcv_mbx(MBX_ID1,&pk_msg)	
33			*MERCD(ercd) == E_OK*	
34			*msg16を受信*	優先度-2
35			ext_tsk(　)	

COM3：ITRON仕様共通規定のテスト手順　その３

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK1,ITSKPRI_2,STKSZ,NULL})		
―	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**	―
1	ercd = get_tid(ERR_ADDR)		
2	*MERCD(ercd) == E_MACV*		サービスコールの返値とエラーコード-3
3	ercd = act_tsk(TASK_ID2)		
4	*MERCD(ercd) == E_OK*		
5	systim1 = SYSTIM_1000		
6	ercd = set_tim(&systim1)		
7	*MERCD(ercd) == E_OK*		
8	ercd = tslp_tsk(TMO_100)		
9		systim1 = SYSTIM_0	
10		ercd = set_tim(&systim1)	
11		*MERCD(ercd) == E_OK*	
12		ercd = dly_tsk(DLYTIM_300)	
13	*MERCD(ercd) == E_TMOUT*		
14	ercd = get_tim(&systim2)		
15	*MERCD(ercd) == E_OK*		
16	*systim2 < SYSTIM_1000*		相対時間とシステム時刻-1
17	ext_tsk(　)		
18		*MERCD(ercd) == E_OK*	
19		ext_tsk(　)	

(注)
No.2　E_MACVエラーを検出できるERR_ADDR番地がシステムにない場合は、テストを省略することができる。

COM4：ITRON仕様共通規定のテスト手順　その４

No	テ　ス　ト　手　順　内　容	テスト 項目参照
1	以下の機能を持つプログラムを作成し、それをソフトウェア部品のコンフィギュレータとして、カーネルのコンフィギュレーション手順に組み込む。 (1) プログラムに渡されたファイルをそのまま表示する機能 (2) プログラムに渡されたファイルに、「CRE_TSK」で始まる行があれば、その直前に「#define TA_HLNG 0x12345678」を追加してファイルとして出力する機能	
2	以下のシステムコンフィギュレーションファイルを入力して、カーネルのコンフィギュレーション手順を実行する。 (システムコンフィギュレーションファイルの内容) INCLUDE(“<kernel.h>”); INCLUDE(“\”task1.h\””); #define　EXINF_1　0x00000001 #define　ITSKPRI_1　0x0001 #define　STKSZ　0x0100 CRE_TSK(TASK_ID1,{TA_HLNG|0x00,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL}); ただし、以下の内容のヘッダファイル「task1.h」を用意しておくこと。 (task1.hの内容) extern　void　TASK1(　);	静的APIの文法とパラメータ-1,2,4,5
3	プリプロセッサからエラーメッセージが出力されないこと。	システムコンフィギュレーションファイル-1,2
4	ソフトウェア部品のコンフィギュレータが、以下の内容を表示すること。 (コンフィギュレータによる表示内容) INCLUDE(“<kernel.h>”); INCLUDE(“\”task1.h\””); CRE_TSK(TASK_ID1,{TA_HLNG|0x00,0x00000001,TASK1,0x0001,0x0100,NULL}); なお、この他に#で始まる行が含まれていてもよい。	システムコンフィギュレーションファイル-2
5	カーネルのコンフィギュレータからエラーメッセージが出力されないこと。	システムコンフィギュレーションファイル-1,2,5 ITRON仕様共通性的API-1
6	カーネルのコンフィギュレータが、以下の内容のカーネル構成・初期化ファイルを生成し、それが正しくコンパイルできること。 (カーネル構成・初期化ファイルの内容) #include <kernel.h> #include “task1.h” タスクの初期化データ	システムコンフィギュレーションファイル-3 ITRON仕様共通性的API-1
7	カーネルのコンフィギュレータが、以下の内容のID自動割付け結果ヘッダファイル「kernel_id.h」を生成すること。 (Id自動割付け結果ヘッダファイル「kernel_id.h」の内容) #define　TASK_ID1　1	APIの構成要素-4 システムコンフィギュレーションファイル-3 静的APIの文法とパラメータ-6

COM5：ITRON仕様共通規定のテスト手順　その５

No	テ　ス　ト　手　順　内　容	テスト 項目参照
1	システムコンフィギュレーションファイルの内容 INCLUDE(“<kernel.h>”); INCLUDE(“\”task1.h\””); #define EXINF_1　　0x00000001 #define ITSKPRI_1　0x0001 #define STKSZ　　0x0100 ABC_DEF(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL}) ; *静的APIの名称が異なるので、カーネルのコンフィギュレータがエラーを報告すること。*	システムコンフィギュレーションファイル-4 静的APIの文法とパラメータ-8
2	システムコンフィギュレーションファイルの内容 INCLUDE(“<kernel.h>”); INCLUDE(“\”task1.h\””); #define EXINF_1　　0x00000001 #define ITSKPRI_1　0x0001 #define STKSZ　　0x0100 CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL}; *静的APIの文法が異なるので、カーネルのコンフィギュレータがエラーを報告すること。*	システムコンフィギュレーションファイル-4 静的APIの文法とパラメータ-8
3	システムコンフィギュレーションファイルの内容 INCLUDE(“<kernel.h>”); INCLUDE(“\”task1.h\””); #define EXINF_1　　0x00000001 #define ITSKPRI_1　0x0001 #define STKSZ　　0x0100 CRE_TSK(TASK_ID1,{TA_HLNG,NULL}) ; *静的APIのパラメータ数が少ないので、カーネルのコンフィギュレータがエラーを報告すること。* ただし、上記記述を許可するようにコンフィギュレータが拡張されていた場合は、テストを省略することができる。	システムコンフィギュレーションファイル-4 静的APIの文法とパラメータ-8
4	システムコンフィギュレーションファイルの内容 INCLUDE(“<kernel.h>”); INCLUDE(“\”task1.h\””); #define EXINF_1　　0x00000001 #define ITSKPRI_1　0x0001 #define STKSZ　　0x0100 CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,NULL,NULL}); *静的APIのパラメータ数が多いので、カーネルのコンフィギュレータがエラーを報告すること。* ただし、上記記述を許可するようにコンフィギュレータが拡張されていた場合は、テストを省略することができる。	システムコンフィギュレーションファイル-4 静的APIの文法とパラメータ-8

COM6：ITRON仕様共通規定のテスト手順　その６

No	テ　ス　ト　手　順　内　容	テスト 項目参照
1	アプリケーションプログラムとリンクしてカーネルを用いる場合、カーネルオブジェクトのシンボルをダンプし、C言語レベルで内部識別子が“ _kernel_ ”または“ _KERNEL_ ”で始まる名称であること。	カーネルとソフトウェア部品の内部識別子-1

COM7：ITRON仕様共通規定のテスト手順　その７

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
—	**<TASK_ID1:READY>**	—
1	*1 == sizeof(B)*	ITRON仕様共通データ型-1
2	b1 = -1	
3	*b1 < 0*	ITRON仕様共通データ型-1
4	*1 == sizeof(UB)*	ITRON仕様共通データ型-4
5	ub1 = -1	
6	*ub1 > 0*	ITRON仕様共通データ型-4
7	*2 == sizeof(H)*	ITRON仕様共通データ型-2
8	h1 = -1	
9	*h1 < 0*	ITRON仕様共通データ型-2
10	*2 == sizeof(UH)*	ITRON仕様共通データ型-5
11	uh1 = -1	
12	*uh1 > 0*	ITRON仕様共通データ型-5
13	*4 == sizeof(W)*	ITRON仕様共通データ型-3
14	w1 = -1	
15	*w1 < 0*	ITRON仕様共通データ型-3
16	*4 == sizeof(UW)*	ITRON仕様共通データ型-6
17	uw1 = -1	
18	*uw1 > 0*	ITRON仕様共通データ型-6
19	*1 == sizeof(VB)*	ITRON仕様共通データ型-7
20	*2 == sizeof(VH)*	ITRON仕様共通データ型-8
21	*4 == sizeof(VW)*	ITRON仕様共通データ型-9
22	*sizeof(void *) == sizeof(VP)*	ITRON仕様共通データ型-10
23	*sizeof(void *) == sizeof(FP)*	ITRON仕様共通データ型-11
24	*2 ≦ sizeof(INT)*	ITRON仕様共通データ型-12

25	int1 = -1	
26	*int1 < 0*	ITRON仕様共通データ型-12
27	*2 ≦ sizeof(UINT)*	ITRON仕様共通データ型-13
28	uint1 = -1	
29	*uint1 > 0*	ITRON仕様共通データ型-13
30	bool1 = TRUE	
31	*bool1 == TRUE*	ITRON仕様共通データ型-14
32	bool1 = FALSE	
33	*bool1 == FALSE*	ITRON仕様共通データ型-14
34	*2 ≦ sizeof(FN)*	ITRON仕様共通データ型-15
35	fn1 = -1	
36	*fn1 < 0*	ITRON仕様共通データ型-15
37	*1 ≦ sizeof(ER)*	ITRON仕様共通データ型-16
38	er1 = -1	
39	*er1 < 0*	ITRON仕様共通データ型-16
40	*2 ≦ sizeof(ID)*	ITRON仕様共通データ型-17
41	id1 = -1	
42	*id1 < 0*	ITRON仕様共通データ型-17
43	*1 ≦ sizeof(ATR)*	ITRON仕様共通データ型-18
44	atr1 = -1	
45	*atr1 > 0*	ITRON仕様共通データ型-18
46	*2 ≦ sizeof(STAT)*	ITRON仕様共通データ型-19
47	stat1 = -1	
48	*stat1 > 0*	ITRON仕様共通データ型-19
49	*1 ≦ sizeof(MODE)*	ITRON仕様共通データ型-20
50	mode1 = -1	
51	*mode1 > 0*	ITRON仕様共通データ型-20
52	*2 ≦ sizeof(PRI)*	ITRON仕様共通データ型-21
53	pri1 = -1	
54	*pri1 < 0*	ITRON仕様共通データ型-21
55	*sizeof(SIZE) == sizeof(void *)*	ITRON仕様共通データ型-22
56	size1 = -1	
57	*size1 > 0*	ITRON仕様共通データ型-22
58	*2 ≦ sizeof(TMO)*	ITRON仕様共通データ型-23
59	tmo1 = -1	
60	*tmo1 < 0*	ITRON仕様共通データ型-23
61	*2 ≦ sizeof(RELTIM)*	ITRON仕様共通データ型-24
62	reltim1 = -1	
63	*reltim1 > 0*	ITRON仕様共通データ型-24
64	*2 ≦ sizeof(SYSTIM)*	ITRON仕様共通データ型-25
65	systim1 = -1	
66	*systim1 > 0*	ITRON仕様共通データ型-25
67	*sizeof(VP_INT) ≧ sizeof(INT) && sizeof(VP_INT) ≧ sizeof(VP)*	ITRON仕様共通データ型-26
68	vp_int1 = -1	
69	*vp_int1 < 0*	ITRON仕様共通データ型-26
70	*1 ≦ sizeof(ER_BOOL)*	ITRON仕様共通データ型-27
71	er_bool1 = TRUE	
72	*er_bool1 == TRUE*	ITRON仕様共通データ型-27
73	er_bool1 = FALSE	
74	*er_bool1 == FALSE*	ITRON仕様共通データ型-27
75	er_bool1 = -1	
76	*er_bool1 < 0*	ITRON仕様共通データ型-27
77	*2 ≦ sizeof(ER_ID)*	ITRON仕様共通データ型-28
78	er_id1 = -1	
79	*er_id1 < 0*	ITRON仕様共通データ型-28
80	*2 ≦ sizeof(ER_UINT)*	ITRON仕様共通データ型-29
81	er_uint1 = -1	
82	*er_uint1 < 0*	ITRON仕様共通データ型-29
83	ext_tsk( )	

COM8：ITRON仕様共通規定のテスト手順　その８

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_0,MAXSEM_FFFF})	
—	**<TASK_ID1:READY>**	—
1	ercd = pol_sem(SEM_ID1)	
2	*(ER)(B)ercd == E_TMOUT*	サービスコールの返値とエラーコード-2
3	*ercd >> 8 < 0*	サービスコールの返値とエラーコード-2
4	*MERCD(ercd) == (ER)(B)ercd*	ITRON仕様共通マクロ-1
5	*SERCD(ercd) == ercd >> 8*	ITRON仕様共通マクロ-2
6	ext_tsk(　)	

SPC1：μITRON4.0仕様の概念と共通のテスト手順　その1

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})			
S4	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_0,MAXSEM_1})			
—	**<TASK_ID1:DORMANT>**	**<TASK_ID2: DORMANT>**	**<TASK_ID3: READY>**	—
1			ercd = act_tsk(TASK_ID2)	
2		ercd = wai_sem(SEM_ID1)		
3			*MERCD(ercd) == E_OK*	
4			ercd = act_tsk(TASK_ID1)	
5	ercd = wai_sem(SEM_ID1)			
6			*MERCD(ercd) == E_OK*	
7			ercd = sus_tsk(TASK_ID2)	
8			*MERCD(ercd) == E_OK*	
9			ercd = sig_sem(SEM_ID1)	
10			*MERCD(ercd) == E_OK*	
11			ercd = rel_wai(TASK_ID2)	
12			*MERCD(ercd) == E_OBJ*	タスク状態-1
13			ercd = sig_sem(SEM_ID1)	
14	*MERCD(ercd) == E_OK*			タスク状態-1
15	ercd = slp_tsk(　)			
16			*MERCD(ercd) == E_OK*	
17			ercd = rsm_tsk(TASK_ID2)	
18		*MERCD(ercd) == E_OK*		タスク状態-1
19		ercd = wai_sem(SEM_ID1)		
20			*MERCD(ercd) == E_OK*	
21			ercd = sus_tsk(TASK_ID1)	
22			*MERCD(ercd) == E_OK*	
23			ercd = sus_tsk(TASK_ID2)	
24			*MERCD(ercd) == E_OK*	
25			ercd = wup_tsk(TASK_ID1)	
26			*ercd = E_OK*	
27			ercd = rsm_tsk(TASK_ID1)	
28	*MERCD(ercd) == E_OK*			タスク状態-2
29	ext_tsk(　)			
30			*MERCD(ercd) == E_OK*	
31			ercd = rsm_tsk(TASK_ID2)	
32			*MERCD(ercd) == E_OK*	
33			ercd = sig_sem(SEM_ID1)	
34		*MERCD(ercd) == E_OK*		タスク状態-2
35		ext_tsk(　)		
36			*MERCD(ercd) == E_OK*	
37			ext_tsk(　)	

SPC2：μITRON4.0仕様の概念と共通のテスト手順　その2

No	テ　ス　ト　手　順　内　容					テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})					
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})					
S3	CRE_TSK(TASK_ID2_1,{TA_HLNG,EXINF_2,TASK2_1,ITSKPRI_2,STKSZ,NULL})					
S4	CRE_TSK(TASK_ID2_2,{TA_HLNG,EXINF_2,TASK2_2,ITSKPRI_2,STKSZ,NULL})					
S5	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})					
—	**<TASK_ID1:DORMANT>**	**<TASK_ID2: DORMANT>**	**<TASK_ID2_1: DORMANT>**	**<TASK_ID2_2: DORMANT>**	**<TASK_ID3:READY>**	—
1					ercd = act_tsk(TASK_ID2)	
2		ercd = act_tsk(TASK_ID2_1)				
3		*ercd = E_OK*				
4		ercd = act_tsk(TASK_ID2_2)				
5		*ercd = E_OK*				
6		ercd = act_tsk(TASK_ID1)				
7	*exinf == EXINF_1*					タスクのスケジューリング規則-1,2

8	ercd = slp_tsk( )					
9		*MERCD(ercd) == E_OK*				タスクのスケジューリング規則-4
10		ercd = slp_tsk( )				
11			ercd = wup_tsk(TASK_ID2)			
12			*MERCD(ercd) == E_OK*			タスクのスケジューリング規則-3
13			ercd = wup_tsk(TASK_ID1)			
14	*MERCD(ercd) == E_OK*					
15	ext_tsk( )					
16			*MERCD(ercd) == E_OK*			タスクのスケジューリング規則-7
17			ext_tsk( )			
18				ext_tsk( )		
19		*MERCD(ercd) == E_OK*				タスクのスケジューリング規則-8
20		ext_tsk( )				
21					*MERCD(ercd) == E_OK*	
22					ext_tsk( )	

SPC3： μITRON4.0仕様の概念と共通のテスト手順　その3

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2: READY>**	－
1		ercd = dis_dsp( )	
2		*MERCD(ercd) == E_OK*	
3		ercd = act_tsk(TASK_ID1)	
4		*MERCD(ercd) == E_OK*	
5		ercd = ena_dsp( )	
6	ext_tsk( )		
7		*MERCD(ercd) == E_OK*	タスクのスケジューリング規則-5
8		ext_tsk( )	

SPC4： μITRON4.0仕様の概念と共通のテスト手順　その4

No	テ　ス　ト　手　順　内　容						テスト 項目参照
S1	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_4,STKSZ,NULL})						
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})						
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})						
S4	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})						
S5	DEF_INH(INHNO_1,{INHATR,INTHDR_1})						
－	**<TASK_ID1:READY>**		**<TASK_ID2:READY>**	**<TASK_ID3:READY>**	**<TASK_ID4:READY>**	**<INTHDR_1:割込みハンドラ>**	－
1	１回目	２回目					
2	ext_tsk( )						
3			<INTHDR_1の割込み発生>				
4						ercd = iact_tsk(TASK_ID1)	
5						*MERCD(ercd) == E_OK*	
6						return	
7		ext_tsk( )					
8			ercd = slp_tsk( )				
9				ercd = wup_tsk(TASK_ID2)			
10			*MERCD(ercd) == E_OK*				
11			ext_tsk( )				
12				*MERCD(ercd) == E_OK*			
13				ext_tsk( )			
14					ext_tsk( )		タスクのスケジューリング規則-6

SPC5： μITRON4.0仕様の概念と共通のテスト手順　その5

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	DEF_EXC(EXCNO_1,{EXCATR,EXCHDR_1})		
S3	DEF_EXC(EXCNO_2,{EXCATR,EXCHDR_2})		例外処理の枠組み-2
－	**<TASK_ID1:READY>**	**<EXCHDR_1:CPU例外ハンドラ>**	－
1	EXCNO_1のCPU例外を発生させる		
2		*return*	例外処理の枠組み-1
3	ext_tsk( )		

SPC6： μITRON4.0仕様の概念と共通のテスト手順　その6

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	DEF_TEX(TASK_ID1,{TA_HLNG,TEXRTN_1})				
S3	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S4	DEF_TEX(TASK_ID2,{TA_HLNG,TEXRTN_2})				
－	**<TASK_ID1:READY>**	**<TASK_ID1のタスク例外処理ルーチン>**	**<TASK_ID2:READY>**	**<TASK_ID2のタスク例外処理ルーチン>**	－
1	ercd = ras_tex(TASK_ID1,RASPTN_1)				
2		*exinf == EXINF_1*			
3		return			
4	ext_tsk( )				例外処理の枠組み-3

5			ercd = ras_tex(TASK_ID2,RASPTN_2)		
6				*exinf == EXINF_2*	例外処理の枠組み-4
7				return	
8			ext_tsk( )		

SPC7： μITRON4.0仕様の概念と共通のテスト手順　その7

No	テ ス ト 手 順 内 容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	DEF_INH(INHNO_1,{INHATR,INTHDR_1})				
S3	CRE_CYC(CYC_ID1,{TA_HLNG,EXINF_2,CYCHDR_1,CYCTIM_10,CYCPHS_5})				
S4	DEF_EXC(EXCNO_1,EXCATR,EXCHDR_1)				
－	**<TASK_ID1:READY>**	**<INHNO_1:割込みハンドラ>**	**<EXCHDR_1:CPU例外ハンドラ>**	**<CYC_ID1:周期ハンドラ>**	－
1	state = sns_ctx( )				
2	*state == FALSE*				タスクコンテキストと 非タスクコンテキスト-1
3	INHNO_1の割込みを発生させる				
4		state = sns_ctx( )			
5		*state == TRUE*			タスクコンテキストと 非タスクコンテキスト-2
6		ercd = iact_tsk(TSK_SELF)			
7		*MERCD(ercd) == E_ID*			タスクコンテキストと 非タスクコンテキスト-3
8		EXCNO_1のCPU例外発生を発生させる			
9			state = sns_ctx( )		
10			*state == TRUE*		タスクコンテキストと 非タスクコンテキスト-5
11			return		
12		return			
13	ercd = sta_cyc(CYC_ID1)				
14	*MERCD(ercd) == E_OK*				
15	ercd = dly_tsk(DLYTIM_20)				
16				state = sns_ctx( )	
17				*state == TRUE*	タスクコンテキストと 非タスクコンテキスト-2
18				return	
19	*MERCD(ercd) == E_OK*				
20	ercd = stp_cyc(CYC_ID1)				
21	*MERCD(ercd) == E_OK*				
22	ext_tsk( )				

SPC8： μITRON4.0仕様の概念と共通のテスト手順　その8

No	テ ス ト 手 順 内 容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	DEF_INH(INHNO_1,{INHATR,INTHDR_1})			
S3	CRE_CYC(CYC_ID1,{TA_HLNG,EXINF_2,CYCHDR_1,CYCTIM_10,CYCPHS_5})			
－	**<TASK_ID1:READY>**	**<INHNO_1:割込みハンドラ>**	**<CYC_ID1:周期起動ハンドラ>**	－
1	state = sns_loc( )			
2	*state == FALSE*			CPUロック状態-15
3	ercd = loc_cpu( )			
4	*MERCD(ercd) == E_OK*			
5	state = sns_loc( )			
6	*state == TRUE*			
7	INHNO_1の割込みを発生させる			
8	*CPUロック状態なので、割込みハンドラは起動されない*			CPUロック状態-1
9	ercd = loc_cpu( )			
10	*MERCD(ercd) == E_OK*			CPUロック状態-2
11	state = sns_ctx( )			
12	*state == FALSE*			CPUロック状態-6
13	state = sns_loc( )			
14	*state == TRUE*			CPUロック状態-7
15	state = sns_dsp( )			
16	*state == FALSE*			CPUロック状態-8
17	state = sns_dpn( )			
18	*state == TRUE*			CPUロック状態-9
19	state = sns_tex( )			
20	*state == TRUE*			CPUロック状態-10
21	ercd = unl_cpu( )			
22		ercd = iloc_cpu( )		
23		*MERCD(ercd) == E_OK*		
24		ercd = iloc_cpu( )		
25		*MERCD(ercd) == E_OK*		CPUロック状態-3
26		state = sns_loc( )		
27		*state == TRUE*		
28		ercd = iunl_cpu( )		
29		*MERCD(ercd) == E_OK*		CPUロック状態-5

30		state = sns_loc( )		
31		*state == FALSE*		
32		return		
33	*MERCD(ercd) == E_OK*			CPUロック状態-4
34	ercd = sta_cyc(CYC_ID1)			
35	*MERCD(ercd) == E_OK*			
36	ercd = loc_cpu( )			
37	*MERCD(ercd) == E_OK*			
38	50msec以上のソフトウェアディレーを入れる			
39	*CPUロック状態なので周期ハンドラは起動されない*			CPUロック状態-1
40	ercd = unl_cpu( )			
41	*MERCD(ercd) == E_OK*			
42	50msecのソフトウェアディレーを入れる			
43			state = sns_loc( )	
44			*state == FALSE*	CPUロック状態-13
45			return	
46	ercd = stp_cyc(CYC_ID1)			
47	*MERCD(ercd) == E_OK*			
48	ext_tsk( )			

（注）

S2　DEF_INHをサポートしていないシステムでは、以下のように置換し、SPC8のテストを実施すること。

S2	ATT_ISR({TA_HLNG,EXINF_1,INTNO_1,ISR_1})	

SPC9： μITRON4.0仕様の概念と共通のテスト手順　その9

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	ATT_ISR({TA_HLNG,EXINF_1,INTNO_1,ISR_1})		
－	**<TASK_ID1:READY>**	**<INTNO_1:割込みサービスルーチン>**	－
1	INTNO_1の割込み番号に対応する割込みを発生させる		
2		state = sns_loc( )	
3		*MERCD(ercd) == FALSE*	CPUロック状態-13
4		return	
5	ext_tsk( )		

（注）

ATT_ISRをシステムがサポートしていない場合は、SPC9のテストを省略することができる。

SPC10： μITRON4.0仕様の概念と共通のテスト手順　その10

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	DEF_EXC(EXCNO_1,{EXCATR,EXCHDR_1})			
S3	DEF_TEX(TASK_ID1,{TA_HLNG,TEXRTN_1})			
－	**<TASK_ID1:READY>**	**<EXCNO_1:CPU例外ハンドラ>**	**<TASK_ID1:タスク例外処理ルーチン>**	－
		1回目 / 2回目		
1	ercd = loc_cpu( )			
2	*MERCD(ercd) == E_OK*			
3	EXCNO_1のCPU例外を発生させる			
4		state = sns_loc( )		
5		*state == TRUE*		CPUロック状態-14
6		return		
7	state = sns_loc( )			
8	*state == TRUE*			CPUロック状態-14
9	ercd = unl_cpu( )			
10	*MERCD(ercd) == E_OK*			
11	EXCNO_1のCPU例外を発生させる			
12		state = sns_loc( )		
13		*state == FALSE*		CPUロック状態-14
14		return		
15	state = sns_loc( )			
16	*state == FALSE*			CPUロック状態-14
17	ercd = ena_tex( )			
18	*MERCD(ercd) == E_OK*			
19	ercd = loc_cpu( )			
20	*MERCD(ercd) == E_OK*			
21	state = sns_loc( )			
22	*state == TRUE*			CPUロック状態-17
23	ercd = unl_cpu( )			
24	*MERCD(ercd) == E_OK*			
25	ercd = ras_tex(TASK_ID1,RASPTN_1)			
26			state = sns_loc( )	
27			*state == FALSE*	CPUロック状態-16
28			return	
29	*MERCD(ercd) == E_OK*			
30	state = sns_loc( )			
31	*state == FALSE*			CPUロック状態-18
32	ext_tsk( )			

SPC11：μITRON4.0仕様の概念と共通のテスト手順　その11

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})	
S3	DEF_TEX(TASK_ID2,{TA_HLNG,TEXRTN_2})	
S4	DEF_INH(INHNO_1,{INHATR,INTHDR_1})	
S5	CRE_CYC(CYC_ID1,{TA_HLNG,EXINF_2,CYCHDR_1,CYCTIM_10,CYCPHS_5})	
S6	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_1,MAXSEM_2})	

—	<TASK_ID1:DORMANT>	<TASK_2:READY>	<TEXRTN_2: ﾀｽｸ例外処理ﾙｰﾁﾝ>		<INHNO_1: 割込みハンドラ>		<CYC_ID1:周期起動ハンドラ>		—
			1回目	2回目	1回目	2回目	1回目	2回目	
1		state = sns_dsp( )							
2		*state == FALSE*							ディスパッチ禁止状態-7
3		ercd = dis_dsp( )							
4		*MERCD(ercd) == E_OK*							
5		ercd = act_tsk(TASK_ID1)							
6		*MERCD(ercd) == E_OK*							ディスパッチ禁止状態-1
7		ercd = pol_sem(SEM_ID1)							
8		*MERCD(ercd) == E_OK*							ディスパッチ禁止状態-9
9		ercd = get_tid(p_tskid)							
10		*MERCD(ercd) == E_OK*							ディスパッチ禁止状態-2
11		*tskid == TASK_ID2*							ディスパッチ禁止状態-2
12		ercd = loc_cpu( )							
13		*MERCD(ercd) == E_OK*							
14		state = sns_dsp( )							
15		*state == TRUE*							ディスパッチ禁止状態-11
16		ercd = unl_cpu( )							
17		*MERCD(ercd) == E_OK*							
18		state = sns_dsp( )							
19		*state == TRUE*							ディスパッチ禁止状態-10
20		INTNO_1の割込み発生							
21					return				
22		state = sns_dsp( )							
23		*state == TRUE*							ディスパッチ禁止状態-4
24		ercd = sta_cyc(CYC_ID1)							
25		*MERCD(ercd) == E_OK*							
26		ソフトウェアでDLYTIM_20以上のディレーを作成							
27							ercd = iact_tsk(TASK_ID1)		
28							*MERCD(ercd) == E_OK*		ディスパッチ禁止状態-3
29							return		
30		ercd = stp_cyc(CYC_ID1)							
31		*MERCD(ercd) == E_OK*							
32		state = sns_dsp( )							
33		*state == TRUE*							ディスパッチ禁止状態-6
34		ercd = ena_tex( )							
35		*MERCD(ercd) == E_OK*							
36		ercd = ras_tex(TASK_ID2,RASPTN_2)							
37			return						
38		state = sns_dsp( )							
39		*state == TRUE*							ディスパッチ禁止状態-8
40		ercd = ena_dsp( )							
41	actcnt = can_act(TSK_SELF)								
42	*actcnt == 1*								
43	ext_tsk( )								
44		*MERCD(ercd) == E_OK*							
45		INTNO_1の割込み発生							
46						return			
47		state = sns_dsp( )							
48		*state = FALSE*							ディスパッチ禁止状態-4
49		ercd = sta_cyc(CYC_ID1)							
50		*MERCD(ercd) == E_OK*						return	
51		ercd = dly_tsk(DLYTIM_20)							
52									
53		*MERCD(ercd) == E_OK*							
54		state = sns_dsp( )							
55		*state == FALSE*							ディスパッチ禁止状態-6
56		ercd = ras_tex(TASK_ID2,RASPTN_2)							
57				return					
58		*MERCD(ercd) == E_OK*							
59		state = sns_dsp( )							
60		*state == FALSE*							ディスパッチ禁止状態-8
61		ercd = loc_cpu( )							
62		*MERCD(ercd) == E_OK*							
63		state = sns_dsp( )							
64		*state == FALSE*							ディスパッチ禁止状態-11
65		ercd = unl_cpu( )							
66		*MERCD(ercd) == E_OK*							

67		ext_tsk( )							

(注)

S4　DEF_INHをサポートしていないシステムでは、以下のように置換し、SPC11のテストを実施すること。

S4	ATT_ISR({TA_HLNG,EXINF_1,INTNO_1,ISR_1})	

なお、DEF_INHとATT_ISRの両方をサポートしているシステムでは、SPC11記載のテストとATT_ISRに置換したテストの２通りを実施すること。

SPC12： μITRON4.0仕様の概念と共通のテスト手順　その12

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	DEF_INH(INHNO_1,{INHATR,INTHDR_1})			
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2:READY>**	**<INHNO_1:割込みハンドラ>**	－
1		INHNO_1の割込み発生		
2			ercd = iact_tsk(TASK_ID1)	
3			*MERCD(ercd) == E_OK*	
4			ercd = iget_tid(p_tskid)	
5			*MERCD(ercd) == E_OK*	
6			*tskid == TASK_ID2*	ディスパッチ保留状態の間のタスク状態-1
7			return	
8	ext_tsk( )			ディスパッチ保留状態の間のタスク状態-1
9		ext_tsk( )		

SPC13： μITRON4.0仕様の概念と共通のテスト手順　その13

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	DEF_INH(INHNO_1,{INHATR,INTHDR_1})		
－	**<TASK_ID1:READY>**	**<INHNO_1:割込みハンドラ>**	－
1	state = sns_ctx( )		
2	*state == FALSE*		非タスクコンテキストから呼び出せるサービスコール-14
3	state = sns_loc( )		
4	*state == FALSE*		非タスクコンテキストから呼び出せるサービスコール-15
5	state = sns_dsp( )		
6	*state == FALSE*		非タスクコンテキストから呼び出せるサービスコール-16
7	state = sns_dpn( )		
8	*state == FALSE*		非タスクコンテキストから呼び出せるサービスコール-17
9	state = sns_tex( )		
10	*state == TRUE*		非タスクコンテキストから呼び出せるサービスコール-18
11	INHNO_1の割込み発生		
12		state = sns_ctx( )	
13		*state == TRUE*	非タスクコンテキストから呼び出せるサービスコール-14
14		state = sns_loc( )	
15		*state == FALSE*	非タスクコンテキストから呼び出せるサービスコール-15
16		state = sns_dsp( )	
17		*state == FALSE*	非タスクコンテキストから呼び出せるサービスコール-16
18		state = sns_dpn( )	
19		*state == TRUE*	非タスクコンテキストから呼び出せるサービスコール-17
20		state = sns_tex( )	
21		*state == TRUE*	非タスクコンテキストから呼び出せるサービスコール-18
22		ercd = isig_tim( )	
23		*MERCD(ercd) == E_OK*	非タスクコンテキストから呼び出せるサービスコール-9
24		return	
25	ext_tsk( )		

（注）

No.19　isig_timサービスコールをサポートしていないシステムでは、テストを省略できる。

SPC14：μITRON4.0仕様の概念と共通のテスト手順　その14

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_1,STKSZ,NULL})			
S3	DEF_INH(INHNO_1,{INHATR,INTHDR_1})			
－	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**	**<INHNO_1:割込みハンドラ>**	－
1	INHNO_1の割込み発生			
2			ercd = iact_tsk(TASK_ID2)	
3			*MERCD(ercd) == E_OK*	
4			ercd = irot_rdq(TSKPRI_1)	
5			*MERCD(ercd) == E_OK*	
6			return	
7		ext_tsk(　)		サービスコールの遅延実行-1
8	ext_tsk(　)			

SPC15：μITRON4.0仕様の概念と共通のテスト手順　その15

No	テ　ス　ト　手　順　内　容					テスト 項目参照
S1	ATT_INI({INIATR,EXINF_1,INIRTN_1})					
S2	ATT_INI({INIATR,EXINF_2,INIRTN_2})					
S3	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})					
S4	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_1,STKSZ,NULL})					
S5	DEF_INH(INHNO_1,{INHATR,INTHDR_1})					
－	**<INIRTN_1:初期化ルーチン>** カーネルの動作開始前に実行	**<INIRTN_2:初期化ルーチン>** カーネルの動作開始前に実行	**<INHNO_1:割込みハンドラ>**	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**	－
1	INHNO_1の割込み発生					
2	return					
3		return				システム初期化手順-2,8
4			ercd = iact_tsk(TASK_ID2)			システム初期化手順-9
5			*MERCD(ercd) == E_OK*			
6			return			
7				ext_tsk(　)		
8					ext_tsk(　)	システム初期化手順-1

SPC16：μITRON4.0仕様の概念と共通のテスト手順　その16

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_1,MAXSEM_2})	
S3 ・ ・ S255	CRE_SEM(SEM_ID2,{TA_TFIFO,ISEMCNT_1,MAXSEM_2}) ・ ・ CRE_SEM(SEM_ID254,{TA_TFIFO,ISEMCNT_1,MAXSEM_2})	
S256	CRE_SEM(SEM_ID255,{TA_TFIFO,ISEMCNT_1,MAXSEM_2})	オブジェクトのID番号と オブジェクト番号-1
－	**<TASK_ID1:READY>**	－
1	ercd = pol_sem(SEM_ID1)	
2	*MERCD(ercd) == E_OK*	
3	ercd = pol_sem(SEM_ID255)	
4	*MERCD(ercd) == E_OK*	オブジェクトの登録とその削除-1
5	ext_tsk(　)	

TSK1：タスク管理機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	CRE_TSK(ERR_TASK_ID,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S4	*E_IDを検出できること*			CRE_TSK-1
S5	CRE_TSK(TASK_ID3,{ERR_TSKATR,TASK1,ITSKPRI_1,STKSZ,NULL})			
S6	*E_RSATRを検出できること*			CRE_TSK-2
S7	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_1,ERR_TASK,ITSKPRI_3,STKSZ,NULL})			
S8	*E_PARを検出できること*			CRE_TSK-3
S9	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_1,TASK1,ERR_ITSKPRI,STKSZ,NULL})			
S10	*E_PARを検出できること*			CRE_TSK-4
S11	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_1,TASK3,ITSKPRI_3,ERR_STKSZ,NULL})			
S12	*E_PARを検出できること*			CRE_TSK-5
S13	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S14	*E_OBJを検出すること*			CRE_TSK-6
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2: READY>**		－
		1回目	2回目	
1		*exinf == EXINF_2*		CRE_TSK-9,11
2		actcnt = can_act(TSK_SELF)		
3		*actcnt == 0*		タスク管理機能-1
4		ercd = ras_tex(TSK_SELF,RASPTN_1)		
5		*MERCD(ercd) == E_OBJ*		タスク管理機能-2
6		ercd = act_tsk(TASK_ID1)		
7	ercd = chg_pri(TASK_ID2,TSKPRI_3)			CRE_TSK-8
8	*MERCD(ercd) == E_OK*			
9	ercd = wup_tsk(TASK_ID2)			
10	*MERCD(ercd) == E_OK*			
11	ercd = sus_tsk(TASK_ID2)			
12	*MERCD(ercd) == E_OK*			
13	ercd = ter_tsk(TASK_ID2)			
14	*MERCD(ercd) == E_OK*			
15	ercd = act_tsk(TASK_ID2)			
16	*MERCD(ercd) == E_OK*			
17	ext_tsk(　)			
18			ercd = get_pri(TSK_SELF,p_tskpri)	
19			*MERCD(ercd) == E_OK*	
20			*tskpri == ITSKPRI_2*	タスク管理機能-3
21			wupcnt = can_wup(TSK_SELF)	
22			*wupcnt == 0*	タスク管理機能-4
24			ext_tsk(　)	

(注)

No.S8　　E_PARエラーを検出できるERR_TASKがシステムに存在しない場合は、テストを省略することができる。

No.S12　E_PARエラーを検出できるERR_STKSZがシステムに存在しない場合は、テストを省略することができる。

TSK2：タスク管理機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2:READY>**	－
1		ercd = act_tsk(ERR_TASK_ID)	
2		*MERCD(ercd) ==E_ID*	act_tsk-1
3		ercd = act_tsk(TSK_SELF)	
4		*MERCD(ercd) == E_OK*	act_tsk-7
5		actcnt = can_act(TSK_SELF)	
6		*actcnt == 1*	act_tsk-7 can_act-4
7		actcnt = can_act(TSK_SELF)	
8		*actcnt == 0*	can_act-2
9		actcnt = can_act(ERR_TASK_ID)	
10		*actcnt == E_ID*	can_act-1
11		ercd = act_tsk(TASK_ID1)	
12	*exinf == EXINF_1*		act_tsk-3,5
13	ercd = act_tsk(TASK_ID2)		
14	*MERCD(ercd) == E_OK*		
15	actcnt = can_act(TASK_ID2)		
16	*actcnt == 1*		can_act-3
17	ercd = act_tsk(TASK_ID2)		
18	*MERCD(ercd) == E_OK*		
19	ercd = act_tsk(TASK_ID2)		
20	TMAX_ACTCNT = 1　　: *MERCD(ercd) == E_QOVR* TMAX_ACTCNT > 1　　: *MERCD(ercd) == E_OK*		act_tsk-2
21	ext_tsk(　)		
22		*MERCD(ercd) == E_OK*	
23		actcnt = can_act(TSK_SELF)	

24		TMAX_ACTCNT = 1　　：*actcnt == 1* TMAX_ACTCNT > 1　　：*actcnt == 2*	act_tsk-6
25		ext_tsk( )	

TSK3：タスク管理機能のテスト手順　その3

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	DEF_INH(INHNO,{INHATR,INTHDR_1})			
―	**<INTHDR_1:割込みハンドラ>**	**<TASK_ID1:DORMANT>**	**<TASK_ID2:READY>**	―
1			INTHDR_1の割込みを発生させる	
2	ercd = iact_tsk(ERR_TASK_ID)			
3	*MERCD(ercd) == E_ID*			iact_tsk-1
4	ercd = iact_tsk(TSK_SELF)			
5	*MERCD(ercd) == E_ID*			iact_tsk-7
6	ercd = iact_tsk(TASK_ID2)			
7	*MERCD(ercd) == E_OK*			非タスクコンテキストから呼び出せるサービスコール-1
8	ercd = iact_tsk(TASK_ID2)			
9	TMAX_ACTCNT = 1　：*MERCD(ercd) == E_QOVR* TMAX_ACTCNT > 1　：*MERCD(ercd) == E_OK*			iact_tsk-2
10	ercd = iact_tsk(TASK_ID1)			
11	*MERCD(ercd) == E_OK*			処理の優先順位とサービスコールの不可分性-1
12	return			
13		*exinf == EXINF_1*		iact_tsk-3,5
14		ext_tsk( )		
15			actcnt == can_act(TSK_SELF)	
16			TMAX_ACTCNT = 1　　：*actcnt == 1* TMAX_ACTCNT > 1　　：*actcnt == 2*	iact_tsk-6
17			ext_tsk( )	

(注)

No.S3　DEF_INHサービスコールをサポートしていないシステムでは、ATT_ISRサービスコールに置換に、テストを実施すること。

N0.9　E_QOVRエラーの検出を省略することが製品マニュアルに記載されている場合：MERCD(ercd) == E_OK または E_QOVR

TSK4：タスク管理機能のテスト手順　その4

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL}			
―	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**		―
		1回目	2回目	
1	ercd = act_tsk(TASK_ID2)			
2	*MERCD(ercd) == E_OK*			
3	ercd = act_tsk(TASK_ID2)			
4	*MERCD(ercd) == E_OK*			
5	ext_tsk( )			
6		ercd = get_pri(TASK_ID1,p_tskpri)		
7		*MERCD(ercd) == E_OBJ*		ext_tsk-1
8		ext_tsk( )		
9			*exinf == EXINF_2*	ext_tsk-4,5
10			actcnt = can_act(TSK_SELF)	
11			*actcnt == 0*	ext_tsk-2
12			ext_tsk( )	

TSK5：タスク管理機能のテスト手順　その5

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
―	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**		―
		1回目	2回目	
1	ercd = act_tsk(TASK_ID2)			
2	*MERCD(ercd) == E_OK*			
3	ercd = act_tsk(TASK_ID2)			
4	*MERCD(ercd) == E_OK*			
5	return			
6		ercd = get_pri(TASK_ID1,p_tskpri)		
7		*MERCD(ercd) == E_OBJ*		return-1
8		return		
9			*exinf == EXINF_2*	return-4
10			actcnt = can_act(TSK_SELF)	
11			*actcnt == 0*	return-2
12			return	

TSK6：タスク管理機能のテスト手順　その6

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1_1,{TA_HLNG|TA_ACT,EXINF_1_1,TASK1_1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID1_2,{TA_HLNG,EXINF_1_2,TASK1_2,ITSKPRI_1,STKSZ,NULL})			
S3	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
—	**<TASK_ID1_1:READY>**	**<TASK_ID1_2:DORMANT>**	**<TASK_ID2:READY>**	—
1	ercd = ter_tsk(ERR_TASK_ID)		このタスクへの状態遷移はない。	
2	*MERCD(ercd) == E_ID*			ter_tsk-1
3	ercd = ter_tsk(TASK_ID1_1)			
4	*MERCD(ercd) == E_ILUSE*			ter_tsk-2
5	ercd = ter_tsk(TASK_ID1_2)			
6	*MERCD(ercd) == E_OBJ*			ter_tsk-3
7	ercd = ter_tsk(TASK_ID2)			
8	*MERCD(ercd) == E_OK*			
9	ercd = get_pri(TASK_ID2,p_tskpri)			
10	*MERCD(ercd) == E_OBJ*			ter_tsk-4
11	ercd = act_tsk(TASK_ID1_2)			
12	*MERCD(ercd) == E_OK*			
13	ercd = act_tsk(TASK_ID1_2)			
14	*MERCD(ercd) == E_OK*			
15	ercd = ter_tsk(TASK_ID1_2)			
16	*MERCD(ercd) == E_OK*			
17	ext_tsk(　)			
18		*exinf == EXINF_1_2*		ter_tsk-6,8
19		ercd = can_act(TSK_SELF)		
20		*actcnt == 0*		ter_tsk-5
21		ext_tsk(　)		

TSK7：タスク管理機能のテスト手順　その7

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})			
—	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<TASK_ID3:DORMANT>**	—
1	ercd = chg_pri(ERR_TASK_ID,TSKPRI_1)			
2	*MERCD(ercd) == E_ID*			chg_pri-1
3	ercd = chg_pri(TASK_ID2,ERR_TSKPRI)			
4	*MERCD(ercd) == E_PAR*			chg_pri-2
5	ercd = chg_pri(TASK_ID3,TSKPRI_1)			
6	*MERCD(ercd) == E_OBJ*			chg_pri-3
7	ercd = get_pri(ERR_TASK_ID,p_tskpri)			
8	*MERCD(ercd) == E_ID*			get_pri-1
9	ercd = get_pri(TASK_ID3,p_tskpri)			
10	*MERCD(ercd) == E_OBJ*			get_pri-2
11	ercd = get_pri(TASK_ID2,p_tskpri)			
12	*MERCD(ercd) == E_OK*			
13	*tskpri == ITSKPRI_2*			get_pri-3
14	ercd = get_pri(TSK_SELF,p_tskpri)			
15	*MERCD(ercd) == E_OK*			
16	*tskpri == ITSKPRI_1*			get_pri-4
17	ercd = act_tsk(TASK_ID3)			
18	*MERCD(ercd) == E_OK*			
19	ercd = chg_pri(TASK_ID2,TSKPRI_3)			
20	*MERCD(ercd) == E_OK*			
21	ercd = chg_pri(TSK_SELF,TSKPRI_3)			
22			ercd = get_pri(TASK_ID2,p_tskpri)	chg_pri-7
23			*MERCD(ercd) == E_OK*	
24			*tskpri == TSKPRI_3*	chg_pri-4
25			ercd = chg_pri(TASK_ID2,TPRI_INI)	
26		ercd = get_pri(TSK_SELF,p_tskpri)		
27		*MERCD(ercd) == E_OK*		
28		*tskpri == ITSKPRI_2*		
29		ext_tsk(　)		
30			*MERCD(ercd) == E_OK*	
31			ext_tsk(　)	
32	*MERCD(ercd) == E_OK*			chg_pri-5,8
33	ercd = chg_pri(TSK_SELF,TPRI_INI)			
34	*MERCD(ercd) == E_OK*			
35	ercd = get_pri(TSK_SELF,p_tskpri)			
36	*MERCD(ercd) == E_OK*			
37	*tskpri == ITSKPRI_1*			chg_pri-6
38	ext_tsk(　)			

SYN1：タスク付属同期機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
—	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**	—
1	ercd = wup_tsk(ERR_TASK_ID)		
2	*MERCD(ercd) == E_ID*		wup_tsk-1
3	ercd = wup_tsk(TASK_ID2)		
4	*ercd = E_OBJ*		wup_tsk-2
5	ercd = wup_tsk(TSK_SELF)		
6	*MERCD(ercd) == E_OK*		
7	ercd = slp_tsk(　)		
8	*MERCD(ercd) == E_OK*		slp_tsk-4 wup_tsk-7
9	ercd = act_tsk(TASK_ID2)		
10	*MERCD(ercd) == E_OK*		
11	ercd = wup_tsk(TASK_ID2)		
12	*MERCD(ercd) == E_OK*		
13	ercd = wup_tsk(TASK_ID2)		
14	TMAX_WUPCNT = 1　　：*MERCD(ercd) == E_QOVR* TMAX_WUPCNT > 1　　：*MERCD(ercd) == E_OK*		wup_tsk-3
15	ercd = slp_tsk(　)		
16		ercd = rel_wai(TASK_ID1)	
17	*MERCD(ercd) == E_RLWAI*		slp_tsk-1
18	ercd = slp_tsk(　)		
19		*MERCD(ercd) == E_OK*	
20		ercd = slp_tsk(　)	
21		*MERCD(ercd) == E_OK*	wup_tsk-6
22		wupcnt = can_wup(TSK_SELF)	
23		TMAX_WUPCNT = 1　　：*wupcnt == 0* TMAX_WUPCNT > 1　　：*wupcnt == 1*	slp_tsk-3
24		ercd = wup_tsk(TASK_ID1)	
25	*MERCD(ercd) == E_OK*		slp_tsk-2 wup_tsk-4,5
26	ext_tsk(　)		
27		*MERCD(ercd) == E_OK*	
28		ext_tsk(　)	

SYN2：タスク付属同期機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
—	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	—
1	ercd = tslp_tsk(ERR_TMO)		
2	*MERCD(ercd) == E_PAR*		tslp_tsk-1
3	ercd = tslp_tsk(TMO_POL)		
4	*MERCD(ercd) == E_TMOUT*		tslp_tsk-3
5	ercd = wup_tsk(TSK_SELF)		
6	*MERCD(ercd) == E_OK*		
7	ercd = tslp_tsk(TMO_POL)		
8	*MERCD(ercd) == E_OK*		tslp_tsk-6,8
9	wupcnt = can_wup(TSK_SELF)		
10	*wupcnt == 0*		tslp_tsk-5
11	ercd = tslp_tsk(TMO_10)		
12		ercd = rel_wai(TASK_ID1)	
13	*MERCD(ercd) == E_RLWAI*		tslp_tsk-2
14	ercd = tslp_tsk(TMO_2)		
15		*MERCD(ercd) == E_OK*	
16		ercd = dly_tsk(DLYTIM_2)	
17	*MERCD(ercd) == E_TMOUT*		tslp_tsk-3,7
18	ercd = tslp_tsk(TMO_FEVR)		
19		*MERCD(ercd) == E_OK*	
20		ercd = dly_tsk(DLYTIM_1000)	
21		*MERCD(ercd) == E_OK*	
22		ercd = wup_tsk(TASK_ID1)	
23	*MERCD(ercd) == E_OK*		tslp_tsk-4,9
24	ext_tsk(　)		
25		*MERCD(ercd) == E_OK*	
26		ext_tsk(　)	

(注)

No.2　E_PARエラーを検出できるERR_TMOがシステムに存在しない場合は、テストを省略することができる。

SYN3：タスク付属同期機能のテスト手順　その3

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S4	DEF_INH(INHNO,{INHATR,INTHDR_1})				
－	**<INTHDR_1:割込みハンドラ>**	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<TASK_ID3:DORMANT>**	－
1		ercd = slp_tsk(　)			
2			INTHDR_1の割込み発生		
3	ercd = iwup_tsk(TSK_SELF)				
4	*MERCD(ercd) == E_ID*				iwup_tsk-7
5	ercd = iwup_tsk(ERR_TASK_ID)				
6	*MERCD(ercd) == E_ID*				iwup_tsk-1
7	ercd = iwup_tsk(TASK_ID3)				
8	*MERCD(ercd) == E_OBJ*　(注)				iwup_tsk-2
9	ercd = iwup_tsk(TASK_ID1)				
10	*MERCD(ercd) == E_OK*				非タスクコンテキストから呼び出せるサービスコール-2
11	ercd = iwup_tsk(TASK_ID2)				
12	*MERCD(ercd) == E_OK*				
13	ercd = iwup_tsk(TASK_ID2)				
14	TMAX_WUPCNT == 1： *MERCD(ercd) == E_QOVR*(注) TMAX_WUPCNT > 1： *MERCD(ercd) == E_OK*				iwup_tsk-3
15	return				
16		*MERCD(ercd) == E_OK*			iwup_tsk-4,5
17		ext_tsk(　)			
18			ercd = tslp_tsk(TMO_POL)		
19			*MERCD(ercd) == E_OK*		iwup_tsk-6
20			ext_tsk(　)		
21				このタスクへの状態遷移はない	

(注)

No.6　E_OBJエラーの検出を省略することが製品マニュアルに記載されている場合　：MERCD(ercd) == E_OK

No.12　E_QOVRエラーの検出を省略することが製品マニュアルに記載されている場合：MERCD(ercd) == E_OK

SYN4：タスク付属同期機能のテスト手順　その4

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
－	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**	－
1	ercd = can_wup(ERR_TASK_ID)		
2	*MERCD(ercd) == E_ID*		can_wup-1
3	ercd = can_wup(TASK_ID2)		
4	*MERCD(ercd) == E_OBJ*		can_wup-2
5	ercd = wup_tsk(TSK_SELF)		
6	*MERCD(ercd) == E_OK*		
7	wupcnt = can_wup(TSK_SELF)		
8	*wupcnt == 1*		can_wup-5
9	ercd = act_tsk(TASK_ID2)		
10	*MERCD(ercd) == E_OK*		
11	ercd = wup_tsk(TASK_ID2)		
12	*MERCD(ercd) == E_OK*		
13	wupcnt = can_wup(TASK_ID2)		
14	*wupcnt == 1*		can_wup-4
15	wupcnt = can_wup(TASK_ID2)		
16	*wupcnt == 0*		can_wup-3
17	ext_tsk(　)		
18		ext_tsk(　)	

SYN5：タスク付属同期機能のテスト手順　その5

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	－
1	ercd = rel_wai(ERR_TASK_ID)		
2	*MERCD(ercd) == E_ID*		rel_wai-1
3	ercd = rel_wai(TASK_ID2)		
4	*MERCD(ercd) == E_OBJ*		rel_wai-2
5	ercd = slp_tsk(　)		
6		ercd = rel_wai(TASK_ID1)	
7	*MERCD(ercd) == E_RLWAI*		rel_wai-3,5
8	ercd = slp_tsk(　)		
9		*MERCD(ercd) == E_OK*	

10		ercd = sus_tsk(TASK_ID1)	
11		*MERCD(ercd) == E_OK*	
12		ercd = rel_wai(TASK_ID1)	
13		*MERCD(ercd) == E_OK*	
14		ercd = rsm_tsk(TASK_ID1)	
15	*MERCD(ercd) == E_RLWAI*		rel_wai-4
16	ext_tsk( )		
17		*MERCD(ercd) == E_OK*	
18		ext_tsk( )	

SYN6：タスク付属同期機能のテスト手順　その6

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S4	DEF_INH(INHNO,{INHATR,INTHDR_1})　【何らかの割込みを発生できるシステムとすること】				
－	**<INTHDR_1:割込みハンドラ>**	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<TASK_ID3:READY>**	－
1		ercd = slp_tsk( )			
2			ercd = sus_tsk(TASK_ID1)		
3			*MERCD(ercd) == E_OK*		
4			ercd = slp_tsk( )		
5				INTHDR_1の割込みを発生	
6	ercd = irel_wai(ERR_TASK_ID)				
7	*MERCD(ercd) == E_ID*				irel_wai-1
8	ercd = irel_wai(TASK_ID3)				
9	*MERCD(ercd) == E_OBJ*(注)				irel_wai-2
10	ercd = irel_wai(TASK_ID1)				
11	*MERCD(ercd) == E_OK*				非タスクコンテキストから呼び出せるサービスコール-3
12	ercd = irel_wai(TASK_ID2)				
13	*MERCD(ercd) == E_OK*				
14	return				
15			*MERCD(ercd) == E_RLWAI*		irel_wai-3,5
16			ercd = rsm_tsk(TASK_ID1)		
17		*MERCD(ercd) == E_RLWAI*			irel_wai-4
18		ext_tsk( )			
19			*MERCD(ercd) == E_OK*		
20			ext_tsk( )		
21				ext_tsk( )	

(注)

No.9　E_OBJエラーの検出を省略することが製品マニュアルに記載されている場合：MERCD(ercd) == E_OK

SYN7：タスク付属同期機能のテスト手順　その7

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})			
－	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**	**<TASK_ID3:READY>**	－
1	*exinf == EXINF_1*			
2	ercd = sus_tsk(ERR_TASK_ID)			
3	*MERCD(ercd) == E_ID*			sus_tsk-1
4	ercd = dis_dsp( )			
5	*MERCD(ercd) == E_OK*			
6	ercd = sus_tsk(TSK_SELF)			
7	*MERCD(ercd) == E_CTX*			sus_tsk-2
8	ercd = ena_dsp( )			
9	*MERCD(ercd) == E_OK*			
10	ercd = sus_tsk(TASK_ID2)			
11	*MERCD(ercd) == E_OBJ*			sus_tsk-3
12	ercd = rsm_tsk(ERR_TASK_ID)			
13	*MERCD(ercd) == E_ID*			rsm_tsk-1
14	ercd = rsm_tsk(TASK_ID2)			
15	*MERCD(ercd) == E_OBJ*			rsm_tsk-2
16	ercd = act_tsk(TASK_ID2)			
17	*MERCD(ercd) == E_OK*			
18	ercd = sus_tsk(TASK_ID2)			
19	*MERCD(ercd) == E_OK*			
20	ercd = slp_tsk( )			
21			ercd = sus_tsk(TASK_ID1)	
22			*MERCD(ercd) == E_OK*	
23			ercd = rsm_tsk(TASK_ID2)	
24		ercd = rsm_tsk(TASK_ID1)		タスク管理機能-5 sus_tsk-5 rsm_tsk-4
25		*MERCD(ercd) == E_OK*		rsm_tsk-5
26		ercd = sus_tsk(TSK_SELF)		

27			*MERCD(ercd) == E_OK*	
28			ercd = sus_tsk(TASK_ID2)	
29			TMAX_SUSCNT = 1 ：*MERCD(ercd) == E_QOVR* TMAX_SUSCNT > 1 ：*MERCD(ercd) == E_OK*	sus_tsk-4,7
30			ercd = wup_tsk(TASK_ID1)	
31	*MERCD(ercd) == E_OK*			sus_tsk-6 rsm_tsk-5
32	ercd = rsm_tsk(TASK_ID2)			
33	*MERCD(ercd) == E_OK*			rsm_tsk-3
34	ercd = rsm_tsk(TASK_ID2)			
35	TMAX_SUSCNT == 1：*MERCD(ercd) == E_OBJ* TMAX_SUSCNT > 1 ：*MERCD(ercd) == E_OK*			
36	ext_tsk( )			
37		*MERCD(ercd) == E_OK*		sus_tsk-8
38		ext_tsk( )		
39			*MERCD(ercd) == E_OK*	
40			ext_tsk( )	

SYN8：タスク付属同期機能のテスト手順　その8

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	－
1	ercd = frsm_tsk(ERR_TASK_ID)		
2	*MERCD(ercd) == E_ID*		frsm_tsk-1
3	ercd = frsm_tsk(TASK_ID2)		
4	*MERCD(ercd) == E_OBJ*		frsm_tsk-2
5	ercd = sus_tsk(TSK_SELF)		
6		ercd = frsm_tsk(TASK_ID1)	
7	*MERCD(ercd) == E_OK*		frsm_tsk-3,4
8	ercd = slp_tsk( )		
9		*MERCD(ercd) == E_OK*	
10		ercd = sus_tsk(TASK_ID1)	
11		*MERCD(ercd) == E_OK*	
12		ercd = frsm_tsk(TASK_ID1)	
13		*MERCD(ercd) == E_OK*	frsm_tsk-5
14		ercd = wup_tsk(TASK_ID1)	
15	*MERCD(ercd) == E_OK*		frsm_tsk-5
16	ext_tsk( )		
17		*MERCD(ercd) == E_OK*	
18		ext_tsk( )	

SYN9：タスク付属同期機能のテスト手順　その9

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
－	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**	－
1	ercd = dly_tsk(ERR_DLYTIM)		
2	*MERCD(ercd) == E_PAR*		dly_tsk-1
3	ercd = dly_tsk(DLYTIM_10)		
4	*MERCD(ercd) == E_OK*		dly_tsk-4 ｼｽﾃﾑ時刻管理－5
5	ercd = act_tsk(TASK_ID2)		
6	*MERCD(ercd) == E_OK*		
7	ercd = dly_tsk(DLYTIM_100)		
8		ercd = rel_wai(TASK_ID1)	
9	*MERCD(ercd) == E_RLWAI*		dly_tsk-2,3
10	ext_tsk( )		
11		*MERCD(ercd) == E_OK*	
12		ext_tsk( )	

(注)

No.2　E_PARエラーを検出できるERR_DLYTIMがシステムに存在しない場合は、テストを省略することができる。

TEX1：タスク例外処理機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	DEF_TEX(ERR_TASK_ID,{TA_HLNG,TEXRTN_1})	
S2	*E_IDを検出すること*	DEF_TEX-1
S3	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S4	DEF_TEX(TASK_ID1,{ERR_TEXATR,TEXRTN_1})	静的APIの文法と ﾊﾟﾗﾒｰﾀ-7
S5	*E_RSATRを検出すること*	DEF_TEX-2
S6	DEF_TEX(TASK_ID1,{TA_HLNG,ERR_TEXRTN})	
S7	*E_PARを検出すること*	DEF_TEX-3

TEX2：タスク例外処理機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	DEF_TEX(TASK_ID1,{TA_HLNG,TEXRTN_1})	
S3	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})	

—	**＜TASK_ID1:READY>**	**＜TASK_ID1:タスク例外処理ルーチン＞**		**＜TASK_ID2:READY>**	—
		１回目の実行	**２回目の実行**		
1	ercd = ras_tex (ERR_TASK_ID,RASPTN_1)				
2	*MERCD(ercd) == E_ID*				ras_tex-1
3	ercd = ras_tex (TASK_ID1,ERR_RASPTN)				
4	*MERCD(ercd) == E_PAR*				ras_tex-2
5	ercd = ras_tex (TASK_ID2,RASPTN_2)				
6	*MERCD(ercd) == E_OBJ*				ras_tex-3
7	ercd = ena_tex(　)				
8	*MERCD(ercd) == E_OK*				
9	ercd = dis_dsp(　)				
10	*MERCD(ercd) == E_OK*				
11	ercd = ras_tex (TSK_SELF,RASPTN_1111)				
12		*texptn == RASPTN_1111*			タスク例外処理機能-1,8,9 DEF_TEX-4 ras_tex-5
13		*exinf == EXINF_1*			タスク例外処理機能-1
14		ercd = ena_dsp(　)			
15		*MERCD(ercd) == E_OK*			
16		state = sns_ctx(　)			
17		*state == FALSE*			タスク例外処理機能-10
18		state = sns_tex(　)			
19		*state == TRUE*			sns_tex-1
20		ercd = ena_tex(　)			
21		*MERCD(ercd) == E_OK*			
22		*タスク例外処理ルーチンが多重起動されない*			タスク例外処理機能-2
23		ercd = dis_tex(　)			
24		*MERCD(ercd) == E_OK*			
25		ercd = slp_tsk(　)			
26				ercd = ena_dsp(　)	
27				*MERCD(ercd) == E_OK*	
28				ercd = ras_tex (TASK_ID1,RASPTN_AAAA)	
29				*MERCD(ercd) == E_OK*	
30				ercd = ras_tex (TASK_ID1,RASPTN_4)	
31				*MERCD(ercd) == E_OK*	
32				ercd = wup_tsk(TASK_ID1)	
33		*MERCD(ercd) == E_OK*			タスク例外処理機能-3,4
34		return			
35			*texptn == RASPTN_AAAE*		タスク例外処理機能--7 ras_tex-4
36			*exinf == EXINF_1*		
37			return		
38	*MERCD(ercd) == E_OK*				タスク例外処理機能-5
39	state = sns_tex(　)				
40	*state == FALSE*				タスク例外処理機能-6 sns_tex-2
41	ext_tsk(　)				
42				*MERCD(ercd) == E_OK*	
43				ercd = ras_tex (TASK_ID1,RASPTN_1)	
44				*MERCD(ercd) == E_OBJ*	ras_tex-3
45				ext_tsk(　)	

TEX3：タスク例外処理機能のテスト手順　その3

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	DEF_TEX(TASK_ID1,{TA_HLNG,TEXRTN_1})	
S3	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})	
S4	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})	
S5	DEF_INH(INHNO,{INHATR,INTHDR_1})	

—	**＜TASK_ID1:READY>**	**＜TASK_ID1:タスク例外処理ルーチン＞**	**＜INTHDR_1:割込みハンドラ＞**	**＜TASK_ID2:READY>**	**＜TASK_ID3:DORMANT>**	—
1	state = sns_tex(　)					
2	*state == TRUE*					タスク管理機能-7
3	ercd = ena_tex(　)					
4	*MERCD(ercd) == E_OK*					タスク管理機能-6

5	ercd = slp_tsk( )					
6				INTHDR_1の割込み発生		
7			ercd = iras_tex (ERR_TASK_ID,RASPTN_1)			
8			*MERCD(ercd) == E_ID*			iras_tex-1
9			ercd = iras_tex (TASK_ID1,ERR_RASPTN)			
10			*MERCD(ercd) == E_PAR*			iras_tex-2
11			ercd = iras_tex (TASK_ID2,RASPTN_2)			
12			*MERCD(ercd) == E_OBJ*			iras_tex-3
13			ercd = iras_tex (TSK_SELF,RASPTN_1)			
14			*MERCD(ercd) == E_ID*			iras_tex-5
15			ercd = iras_tex (TASK_ID1,RASPTN_55AA)			
16			*MERCD(ercd) == E_OK*			非タスクコンテキストから呼び出せるサービスコール-4
17			ercd = iras_tex (TASK_ID1,RASPTN_AA55)			
18			*MERCD(ercd) == E_OK*			
19			ercd = iras_tex (TASK_ID3,RASPTN_2)			
20			*MERCD(ercd) == E_OBJ*			iras_tex-3
21			ercd = iwup_tsk(TASK_ID1)			
22			*MERCD(ercd) == E_OK*			
23			return			
24		*texptn == RASPTN_FFFF*				iras_tex-4
25		*exinf == EXINF_1*				
26		return				
27	*MERCD(ercd) == E_OK*					
28	ext_tsk( )					
29				ext_tsk( )		

TEX4：タスク例外処理機能のテスト手順　その4

No	テ　ス　ト　手　順　内　容				テスト項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	DEF_TEX(TASK_ID1,{TA_HLNG,TEXRTN_1})				
S3	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S4	DEF_INH(INHNO,{INHATR,INTHDR_1}) 【何らかの割込みを発生できるシステムとすること】				
―	**<TASK_ID1:READY>**	**<TASK_ID1:タスク例外処理ルーチン>**	**<INTHDR_1:割込みハンドラ>**	**<TASK_ID2:READY>**	―
1	ercd = ena_tex( )				
2	*MERCD(ercd) == E_OK*				ena_tex-2
3	state = sns_tex( )				
4	*state == FALSE*				
5	ercd = dis_tex( )				
6	*MERCD(ercd) == E_OK*				dis_tex-2
7	state = sns_tex( )				
8	*state == TRUE*				
9	ercd = ras_tex (TASK_ID1,RASPTN_1111)				
10	*MERCD(ercd) == E_OK*				
11	ercd = ena_tex( )				
12		return			
13	*MERCD(ercd) == E_OK*				
14	ercd = slp_tsk( )				
15				ercd = dis_tex( )	
16				*MERCD(ercd) == E_OBJ*	dis_tex-1
17				ercd = ena_tex( )	
18				*MERCD(ercd) == E_OBJ*	ena_tex-1
19				10msec以上経過後にINTHDR_1の割込みが発生するよう設定する	
20				ext_tsk( )	
21			state = sns_tex( )		
22			*state == TRUE*		sns_tex-3
23			ercd = iwup_tsk(TASK_ID1)		
24			*MERCD(ercd) == E_OK*		
25			return		
26	*MERCD(ercd) == E_OK*				
27	ext_tsk( )				

SEM1：セマフォ機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_SEM(ERR_SEM_ID,{TA_TFIFO,ISEMCNT_1,MAXSEM_FFFF})	
S2	*E_IDを検出すること*	CRE_SEM-1
S3	CRE_SEM(SEM_ID3,{ERR_SEMATR,ISEMCNT_1,MAXSEM_FFFF})	
S4	*E_RSATRを検出すること*	CRE_SEM-2
S5	CRE_SEM(SEM_ID3,{TA_TPRI,ISEMCNT_2,MAXSEM_1})	
S6	*E_PARを検出すること*	CRE_SEM-3
S7	CRE_SEM(SEM_ID3,{TA_TPRI,ISEMCNT_2,MAXSEM_OVER})	
S8	*E_PARを検出すること*	CRE_SEM-4
S9	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_1,MAXSEM_FFFF})	
S10	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_1,MAXSEM_FFFF})	
S11	*E_OBJを検出すること*	CRE_SEM-5

(注)
No.S8　　E_PARエラーを検出できるMAXSEM_OVERがシステムに存在しない場合は、テストを省略することができる。

SEM2：セマフォ機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S4	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_4,STKSZ,NULL})				
S5	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_2,MAXSEM_2})				
S6	CRE_SEM(SEM_ID2,{TA_TFIFO,ISEMCNT_FFFE,MAXSEM_FFFF})				
—	**<TASK_ID1:DORMANT>**	**<TASK_ID2:DORMANT>**	**<TASK_ID3:DORMANT>**	**<TASK_ID4:READY>**	—
1				ercd = sig_sem(ERR_SEM_ID)	
2				*MERCD(ercd) == E_ID*	sig_sem-1
3				ercd = wai_sem(ERR_SEM_ID)	
4				*MERCD(ercd) == E_ID*	wai_sem-1
5				ercd = pol_sem(ERR_SEM_ID)	
6				*MERCD(ercd) == E_ID*	pol_sem-1
7				ercd = sig_sem(SEM_ID1)	
8				*MERCD(ercd) == E_QOVR*	CRE_SEM-9 sig_sem-2
9				ercd = wai_sem(SEM_ID1)	
10				*MERCD(ercd) == E_OK*	wai_sem-3,4
11				ercd = pol_sem(SEM_ID1)	
12				*MERCD(ercd) == E_OK*	pol_sem-3,4
13				ercd = pol_sem(SEM_ID1)	
14				*MERCD(ercd) == E_TMOUT*	CRE_SEM-8 pol_sem-2
15				ercd = sig_sem(SEM_ID1)	
16				*MERCD(ercd) == E_OK*	sig_sem-6
17				ercd = pol_sem(SEM_ID1)	
18				*MERCD(ercd) == E_OK*	
19				ercd = act_tsk(TASK_ID3)	
20			ercd = wai_sem(SEM_ID1)		
21				*MERCD(ercd) == E_OK*	
22				ercd = act_tsk(TASK_ID2)	
23		ercd = wai_sem(SEM_ID1)			
24				*MERCD(ercd) == E_OK*	
25				ercd = act_tsk(TASK_ID1)	
26	ercd = wai_sem(SEM_ID1)				
27				*MERCD(ercd) == E_OK*	
28				ercd = sig_sem(SEM_ID1)	
29			*MERCD(ercd) == E_OK*		CRE_SEM-6 sig_sem-3,5
30			ext_tsk(　)		
31				*MERCD(ercd) == E_OK*	sig_sem-4
32				ercd = sig_sem(SEM_ID1)	
33		*MERCD(ercd) == E_OK*			wai_sem-5,6
34		ext_tsk(　)			
35				*MERCD(ercd) == E_OK*	
36				ercd = rel_wai(TASK_ID1)	
37	*MERCD(ercd) == E_RLWAI*				wai_sem-2
38	ext_tsk(　)				
39				*MERCD(ercd) == E_OK*	
40				ercd = sig_sem(SEM_ID2)	
41				*MERCD(ercd) == E_OK*	
42				ext_tsk(　)	

SEM3：セマフォ機能のテスト手順　その3

No	テ　ス　ト　手　順　内　容					テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})					
S2	CRE_TSK(TASK_ID2_1,{TA_HLNG,EXINF_2_1,TASK2_1,ITSKPRI_2,STKSZ,NULL})					
S3	CRE_TSK(TASK_ID2_2,{TA_HLNG,EXINF_2_2,TASK2_2,ITSKPRI_2,STKSZ,NULL})					
S4	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})					
S5	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_1,MAXSEM_2})					
S6	CRE_SEM(SEM_ID2,{TA_TPRI,ISEMCNT_1,MAXSEM_1})					
S7	DEF_INH(INHNO,{INHATR,INTHDR_1}　【何らかの割込みを発生できるシステムとすること】					
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2_1:DORMANT>**	**<TASK_ID2_2:DORMANT>**	**<TASK_ID3:READY>**	**<INTHDT_1:割込みハンドラ>**	－
1				ercd = pol_sem(SEM_ID2)		
2				*MERCD(ercd) == E_OK*		
3				ercd = act_tsk(TASK_ID2_1)		
4		ercd = wai_sem(SEM_ID2)				
5				*MERCD(ercd) == E_OK*		
6				ercd = act_tsk(TASK_ID2_2)		
7			ercd = wai_sem(SEM_ID2)			
8				*MERCD(ercd) == E_OK*		
9				ercd = act_tsk(TASK_ID1)		
10	ercd = wai_sem(SEM_ID2)					
11				*MERCD(ercd) == E_OK*		
12				INTHDR_1の割込み発生		
13					ercd = isig_sem(ERR_SEM_ID)	
14					*MERCD(ercd) == E_ID*	isig_sem-1
15					ercd = isig_sem(SEM_ID1)	
16					*MERCD(ercd) == E_OK*	非タスクコンテキストから呼び出せるサービスコール-5
17					ercd = isig_sem(SEM_ID1)	
18					*MERCD(ercd) == E_QOVR*(注)	isig_sem-2
19					ercd = isig_sem(SEM_ID2)	
20					*MERCD(ercd) == E_OK*	
21					return	
22	*MERCD(ercd) == E_OK*					isig_sem-3,5 wai_sem-7
23	ext_tsk(　)					
24				ercd = sig_sem(SEM_ID2)		
25		*MERCD(ercd) == E_OK*				isig_sem-4
26		ext_tsk(　)				
27				*MERCD(ercd) == E_OK*		
28				ercd = sig_sem(SEM_ID2)		
29			*MERCD(ercd) == E_OK*			wai_sem-8
30			ext_tsk(　)			
31				*MERCD(ercd) == E_OK*		
32				ercd = pol_sem(SEM_ID1)		
33				*MERCD(ercd) == E_OK*		
34				ercd = pol_sem(SEM_ID1)		
35				*MERCD(ercd) == E_OK*		isig_sem-6
36				ercd = pol_sem(SEM_ID1)		
37				*MERCD(ercd) == E_TMOUT*		
38				ext_tsk(　)		

(注)
No.18　サービスコールを遅延実行する場合には、E_QOVRエラーを返すことを省略できる。省略する場合は、製品マニュアルにその旨が記載されていて、E_OKを返すこと。

SEM4：セマフォ機能のテスト手順　その4

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S4	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_4,STKSZ,NULL})				
S5	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_1,MAXSEM_2})				
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2:DORMANT>**	**<TASK_ID3:DORMANT>**	**<TASK_ID4:READY>**	－
1				ercd = twai_sem (ERR_SEM_ID,TMO_10)	
2				*MERCD(ercd) == E_ID*	twai_sem-1
3				ercd = twai_sem (SEM_ID1,ERR_TMO)	
4				*MERCD(ercd) == E_PAR*	twai_sem-2
5				ercd = twai_sem(SEM_ID1,TMO_POL)	
6				*MERCD(ercd) == E_OK*	twai_sem-5,6
7				ercd = twai_sem(SEM_ID1,TMO_POL)	
8				*MERCD(ercd) == E_TMOUT*	タイムアウトとノンブロッキング-1 twai_sem-12

9				ercd = act_tsk(TASK_ID3)	
10			ercd = twai_sem(SEM_ID1,TMO_100)		
11				*MERCD(ercd) == E_OK*	
12				ercd = act_tsk(TASK_ID2)	
13		ercd = twai_sem(SEM_ID1,TMO_200)			
14				*MERCD(ercd) == E_OK*	
15				ercd = act_tsk(TASK_ID1)	
16	ercd = twai_sem (SEM_ID1,TMO_FEVR)				
17				*MERCD(ercd) == E_OK*	
18				ercd = dly_tsk(DLYTIM_50)	
19				*MERCD(ercd) == E_OK*	
20				ercd = sig_sem(SEM_ID1)	
21			*MERCD(ercd) == E_OK*		twai_sem-7,11
22			ext_tsk( )		
23				*MERCD(ercd) == E_OK*	
24				ercd = dly_tsk(DLYTIM_1000)	
25		*MERCD(ercd) == E_TMOUT*			タイムアウトと ノンブロッキング-2 twai_sem-4
26		ext_tsk( )			
27				*MERCD(ercd) == E_OK*	
28				ercd = rel_wai(TASK_ID1)	
29	*MERCD(ercd) == E_RLWAI*				タイムアウトと ノンブロッキング-3 twai_sem-3,8,13
30	ext_tsk( )				
31				*MERCD(ercd) == E_OK*	
32				ext_tsk( )	

(注)

No.4　E_PARエラーを検出できるERR_TMOがシステムに存在しない場合は、テストを省略することができる。

SEM5：セマフォ機能のテスト手順　その5

No	テ　ス　ト　手　順　内　容	テスト項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_TSK(TASK_ID2_1,{TA_HLNG,EXINF_2_1,TASK2_1,ITSKPRI_2,STKSZ,NULL})	
S3	CRE_TSK(TASK_ID2_2,{TA_HLNG,EXINF_2_2,TASK2_2,ITSKPRI_2,STKSZ,NULL})	
S4	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})	
S5	CRE_SEM(SEM_ID2,{TA_TPRI,ISEMCNT_0,MAXSEM_1})	

—	**<TASK_ID1:DORMANT>**	**<TASK_ID2_1:DORMANT>**	**<TASK_ID2_2:DORMANT>**	**<TASK_ID3:READY>**	—
1				ercd = act_tsk(TASK_ID2_1)	
2		ercd = twai_sem(SEM_ID2,TMO_100)			
3				*MERCD(ercd) == E_OK*	
4				ercd = act_tsk(TASK_ID2_2)	
5			ercd = twai_sem(SEM_ID2,TMO_FEVR)		
6				*MERCD(ercd) == E_OK*	
7				ercd = act_tsk(TASK_ID1)	
8	ercd = twai_sem(SEM_ID2,TMO_200)				
9				*MERCD(ercd) == E_OK*	
10				ercd = sig_sem(SEM_ID2)	
11	*MERCD(ercd) == E_OK*				CRE_SEM-7 twai_sem-9
12	ext_tsk( )				
13				*MERCD(ercd) == E_OK*	
14				ercd = sig_sem(SEM_ID2)	
15		*MERCD(ercd) == E_OK*			
16		ext_tsk( )			
17				*MERCD(ercd) == E_OK*	
18				ercd = sig_sem(SEM_ID2)	
19			*MERCD(ercd) == E_OK*		twai_sem-10
20			ext_tsk( )		
21				*MERCD(ercd) == E_OK*	
22				ercd = act_tsk(TASK_ID1)	
23	ercd = twai_sem(SEM_ID2,TMO_FEVR)				
24				*MERCD(ercd) == E_OK*	
25				ercd = act_tsk(TASK_ID2_1)	
26		ercd = twai_sem(SEM_ID2,TMO_FEVR)			
27				*MERCD(ercd) == E_OK*	
28				ercd = act_tsk(TASK_ID2_2)	
29			ercd = twai_sem(SEM_ID2,TMO_FEVR)		
30				*MERCD(ercd) == E_OK*	
31				ercd = chg_pri(TASK_ID1,TSKPRI_4)	
32				*MERCD(ercd) == E_OK*	
33				ercd = chg_pri(TASK_ID2_2,TSKPRI_3)	

34				*MERCD(ercd) == E_OK*	
35				ercd = chg_pri(TASK_ID2_1,TSKPRI_3)	
36				*MERCD(ercd) == E_OK*	
37				ercd = sig_sem(SEM_ID2)	
38				ext_tsk( )	
39			*MERCD(ercd) == E_OK*		
40			ercd = sig_sem(SEM_ID2)		
41			ext_tsk( )		
42		*MERCD(ercd) == E_OK*			chg_pri-10
43		ercd = sig_sem(SEM_ID2)			
44		ext_tsk( )			
45	*MERCD(ercd) == E_OK*				chg_pri-9
46	ext_tsk( )				

FLG1：イベントフラグ機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_FLG(ERR_FLG_ID,{TA_TFIFO,IFLGPTN_1})	
S2	E_IDを検出できること	CRE_FLG-1
S3	CRE_FLG(FLG_ID1,{ERR_FLGATR,IFLGPTN_1})	
S4	E_RSATRを検出できること	CRE_FLG-2
S5	CRE_FLG(FLG_ID1,{TA_TFIFO,ERR_IFLGPTN})	
S6	E_PARを検出できること	CRE_FLG-3
S7	CRE_FLG(FLG_ID1,{TA_FIFO,IFLGPTN_1})	
S8	CRE_FLG(FLG_ID1,{TA_FIFO,IFLGPTN_1})	
S9	E_OBJを検出できること	CRE_FLG-4

(注)
No.S6　E_PARエラーを検出できるERR_IFLGPTNがシステムに存在しない場合は、テストを省略することができる。

FLG2：イベントフラグ機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
S3	CRE_FLG(FLG_ID1,{TA_TFIFO|TA_WSGL,IFLGPTN_0})		
S4	CRE_FLG(FLG_ID2,{TA_TPRI|TA_WSGL,IFLGPTN_0})		
S5	CRE_FLG(FLG_ID3,{TA_TPRI|TA_WSGL|TA_CLR,IFLGPTN_1})		
—	**<TASK_ID1:DORMANT>**	**<TASK_ID2:READY>**	—
1		ercd = set_flg(ERR_FLG_ID,SETPTN_1)	
2		*MERCD(ercd) == E_ID*	set_flg-1
3		ercd = set_flg(FLG_ID1,ERR_SETPTN)	
4		*MERCD(ercd) == E_PAR*	set_flg-2
5		ercd = clr_flg(ERR_FLG_ID,CLRPTN_1)	
6		*MERCD(ercd) == E_ID*	clr_flg-1
7		ercd = clr_flg(FLG_ID1,ERR_CLRPTN)	
8		*MERCD(ercd) == E_PAR*	clr_flg-2
9		ercd = wai_flg(ERR_FLG_ID,WAIPTN_1,TWF_ANDW,p_flgptn)	
10		*MERCD(ercd) == E_ID*	wai_flg-1
11		ercd = wai_flg(FLG_ID1,ERR_WAIPTN,TWF_ORW,p_flgptn)	
12		*MERCD(ercd) == E_PAR*	wai_flg-2
13		ercd = wai_flg(FLG_ID1,WAIPTN_1,ERR_WFMODE,p_flgptn)	
14		*MERCD(ercd) == E_PAR*	wai_flg-3
15		ercd = wai_flg(FLG_ID1,WAIPTN_1,TWF_ANDW,ERR_pointer)	
16		*MERCD(ercd) == E_PAR*	wai_flg-4
17		ercd = pol_flg(ERR_FLG_ID,WAIPTN_1,TWF_ANDW,p_flgptn)	
18		*MERCD(ercd) == E_ID*	pol_flg-1
19		ercd = pol_flg(FLG_ID1,ERR_WAIPTN,TWF_ORW,p_flgptn)	
20		*MERCD(ercd) == E_PAR*	pol_flg-2
21		ercd = pol_flg(FLG_ID1,WAIPTN_1,ERR_WFMODE,p_flgptn)	
22		*MERCD(ercd) == E_PAR*	pol_flg-3
23		ercd = pol_flg(FLG_ID1,WAIPTN_1,TWF_ANDW,ERR_pointer)	
24		*MERCD(ercd) == E_PAR*	pol_flg-4
25		ercd = set_flg(FLG_ID1,SETPTN_5555)	
26		*MERCD(ercd) == E_OK*	CRE_FLG-5,7
27		ercd = set_flg(FLG_ID2,SETPTN_AAAA)	
28		*MERCD(ercd) == E_OK*	CRE_FLG-6,8
29		ercd = wai_flg(FLG_ID2,WAIPTN_0002,TWF_ORW,p_flgptn)	
30		*MERCD(ercd) == E_OK*	wai_flg-9
31		*flgptn == SETPTN_AAAA*	wai_flg-10
32		ercd = pol_flg(FLG_ID2,WAIPTN_0001,TWF_ORW,p_flgptn)	
33		*MERCD(ercd) == E_TMOUT*	pol_flg-6
34		ercd = clr_flg(FLG_ID1,CLRPTN_0000)	
35		*MERCD(ercd) == E_OK*	
36		ercd = pol_flg(FLG_ID1,WAIPTN_0001,TWF_ANDW,p_flgptn)	
37		*MERCD(ercd) == E_TMOUT*	pol_flg- 7 clr_flg-3
38		ercd = pol_flg(FLG_ID1,WAIPTN_0000,TWF_ANDW,p_flgptn)	
39		*MERCD(ercd) == E_PAR*	pol_flg-11
40		ercd = set_flg(FLG_ID1,SETPTN_0055)	

41		*MERCD(ercd) == E_OK*	
42		ercd = pol_flg(FLG_ID1,WAIPTN_0050,TWF_ORW,p_flgptn)	
43		*MERCD(ercd) == E_OK*	pol_flg- 7
44		*flgptn == FLGPTN_0055*	pol_flg-9
45		ercd = set_flg(FLG_ID1,SETPTN_5500)	
46		*MERCD(ercd) == E_OK*	
47		ercd = pol_flg(FLG_ID1,WAIPTN_0055,TWF_ORW,p_flgptn)	
48		*MERCD(ercd) == E_OK*	
49		*flgptn == FLGPTN_5555*	set_flg-3
50		ercd = clr_flg(FLG_ID1,CLRPTN_0000)	
51		*MERCD(ercd) == E_OK*	
52		ercd = act_tsk(TASK_ID1)	
53	ercd = wai_flg(FLG_ID1,WAIPTN_0001,TWF_ANDW,p_flgptn)		
54		*MERCD(ercd) == E_OK*	
55		ercd = wai_flg(FLG_ID1,WAIPTN_0002,TWF_ANDW,p_flgptn)	
56		*MERCD(ercd) == E_ILUSE*	wai_flg-5,7
57		ercd = pol_flg(FLG_ID1,WAIPTN_0002,TWF_ORW,p_flgptn)	
58		*MERCD(ercd) == E_ILUSE*	pol_flg-5
59		ercd = rel_wai(TASK_ID1)	
60	*MERCD(ercd) == E_RLWAI*		wai_flg-6,12
61	ercd = wai_flg(FLG_ID1,WAIPTN_0001,TWF_ORW,p_flgptn)		
62		*MERCD(ercd) == E_OK*	
63		ercd = set_flg(FLG_ID1,SETPTN_1111)	
64	*MERCD(ercd) == E_OK*		set_flg-4,5 wai_flg-8
65	*flgptn == SETPTN_1111*		set_flg-6
66	ercd = wai_flg(FLG_ID1,WAIPTN_0000,TWF_ANDW,p_flgptn)		
67	*MERCD(ercd) == E_PAR*		wai_flg-13
68	ercd = wai_flg(FLG_ID2,WAIPTN_AAAA,TWF_ANDW,p_flgptn)		
69	*MERCD(ercd) == E_OK*		wai_flg-7,9
70	*flgptn == SETPTN_AAAA*		wai_flg-10
71	ercd = wai_flg(FLG_ID3,WAIPTN_0001,TWF_ORW,p_flgptn)		
72	*MERCD(ercd) == E_OK*		CRE_FLG-10,12 wai_flg-8
73	*flgptn == SETPTN_0001*		CRE_FLG-13
74	ercd = set_flg(FLG_ID3,SETPTN_0002)		
75	*MERCD(ercd) == E_OK*		
76	ercd = pol_flg(FLG_ID3,WAIPTN_0002,TWF_ANDW,p_flgptn)		
77	*MERCD(ercd) == E_OK*		pol_flg-8
78	*flgptn == SETPTN_0002*		
79	ercd = pol_flg(FLG_ID3,WAIPTN_0002,TWF_ANDW,p_flgptn)		
80	*MERCD(ercd) == E_TMOUT*		pol_flg-10
81	ercd = wai_flg(FLG_ID3,WAIPTN_8000,TWF_ORW,p_flgptn)		
82		*MERCD(ercd) == E_OK*	
83		ercd = set_flg(FLG_ID3,SETPTN_8000)	
84	*MERCD(ercd) == E_OK*		set_flg-5 pol_flg-8
85	*flgptn == SETPTN_8000*		set_flg-6
86	ercd = pol_flg(FLG_ID3,WAIPTN_8000,TWF_ORW,p_flgptn)		
87	*MERCD(ercd) == E_TMOUT*		set_flg-7
88	ercd = set_flg(FLG_ID3,SETPTN_5555)		
89	*MERCD(ercd) == E_OK*		
90	ercd = wai_flg(FLG_ID3,WAIPTN_5555,TWF_ORW,p_flgptn)		
91	*MERCD(ercd) == E_OK*		
92	*flgptn == SETPTN_5555*		
93	ercd = pol_flg(FLG_ID3,WAIPTN_5555,TWF_ORW,p_flgptn)		
94	*MERCD(ercd) == E_TMOUT*		wai_flg-11
95	ext_tsk( )		
96		*MERCD(ercd) == E_OK*	
97		ext_tsk( )	

(注)

No.4　E_PARエラーを検出できるERR_SETPTNがシステムに存在しない場合は、テストを省略することができる。
No.8　E_PARエラーを検出できるERR_CLRPTNがシステムに存在しない場合は、テストを省略することができる。
No.12　E_PARエラーを検出できるERR_WAIPTNがシステムに存在しない場合は、テストを省略することができる。
No.16　E_PARエラーを検出できるERR_pointerがシステムに存在しない場合は、テストを省略することができる。
No.20　E_PARエラーを検出できるERR_WAIPTNがシステムに存在しない場合は、テストを省略することができる。
No.24　E_PARエラーを検出できるERR_pointerがシステムに存在しない場合は、テストを省略することができる。

FLG3：イベントフラグ機能のテスト手順　その3

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S4	DEF_INT(INHNO,{INHATR,INTHDR_1})				
S5	CRE_FLG(FLG_ID1,{TA_TFIFO|TA_WSGL|TA_CLR,IFLGPTN_0})				
S6	CRE_FLG(FLG_ID2,{TA_TPRI|TA_WSGL,IFLGPTN_0})				
－	**<TASK_ID1:READY>**	**<INTHDR_1:割込みハンドラ>**	**<TASK_ID2:READY>**	**<TASK_ID3:READY>**	－
1	ercd = wai_flg(FLG_ID2,WAIPTN_0505, TWF_ORW,p_flgptn)				
2			ercd = wai_flg(FLG_ID1,WAIPTN_0505, TWF_ANDW,p_flgptn)		
3				INTHDR_1の割込みを発生させる	
4		ercd = iset_flg(ERR_FLG_ID,SETPTN_0001)			
5		*MERCD(ercd) == E_ID*			iset_flg-1
6		ercd = iset_flg(FLG_ID2,ERR_SETPTN)			
7		*MERCD(ercd) == E_PAR*			iset_flg-2
8		ercd = iset_flg(FLG_ID2,SETPTN_0500)			
9		*MERCD(ercd) == E_OK*			非タスクコンテキストから呼び出せるサービスコール-6
10		ercd = iset_flg(FLG_ID1,SETPTN_0500)			
11		*MERCD(ercd) == E_OK*			
12		ercd = iset_flg(FLG_ID2,SETPTN_0005)			
13		*MERCD(ercd) == E_OK*			
14		ercd = iset_flg(FLG_ID1,SETPTN_0005)			
15		*MERCD(ercd) == E_OK*			
16		return			
17	*MERCD(ercd) == E_OK*				iset_flg-4,5
18	*flgptn == SETPTN_0500*				iset_flg-3,6
19	ext_tsk(　)				
20			*MERCD(ercd) == E_OK*		CRE_FLG-9,11
21			*flgptn == SETPTN_0505*		
22			ercd = pol_flg(FLG_ID1,WAIPTN_0001, TWF_ORW,p_flgptn)		
23			*MERCD(ercd) == E_TMOUT*		iset_flg-7
24			ext_tsk(　)		

(注)

No.7　E_PARエラーを検出できるERR_SETPTNがシステムに存在しない場合は、テストを省略することができる。

FLG4：イベントフラグ機能のテスト手順　その4

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
S3	CRE_FLG(FLG_ID1,{TA_TPRI|TA_WSGL,IFLGPTN_0})		
S4	CRE_FLG(FLG_ID2,{TA_TFIFO|TA_WSGL|TA_CLR,IFLGPTN_0})		
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	－
1	ercd = twai_flg(ERR_FLG_ID,WAIPTN_0001,TWF_ANDW,p_flgptn,TMO_POL)		
2	*MERCD(ercd) == E_ID*		twai_flg-1
3	ercd = twai_flg(FLG_ID1,ERR_WAIPTN,TWF_ORW,p_flgptn,TMO_POL)		
4	*MERCD(ercd) == E_PAR*		twai_flg-2
5	ercd = twai_flg(FLG_ID1,WAIPTN_0001,ERR_WFMODE,p_flgptn,TMO_POL)		
6	*MERCD(ercd) == E_PAR*		twai_flg-3
7	ercd = twai_flg(FLG_ID1,WAIPTN_0001,TWF_ANDW,ERR_pointer,TMO_POL)		
8	*MERCD(ercd) == E_PAR*		twai_flg-4
9	ercd = twai_flg(FLG_ID1,WAIPTN_0001,TWF_ANDW,p_flgptn,ERR_TMO)		
10	*MERCD(ercd) == E_PAR*		twai_flg-5
11	ercd = twai_flg(FLG_ID1,WAIPTN_0000,TWF_ORW,p_flgptn,TMO_POL)		
12	*MERCD(ercd) == E_PAR*		twai_flg-15
13	ercd = set_flg(FLG_ID1,SETPTN_0055)		
14	*MERCD(ercd) == E_OK*		
15	ercd = twai_flg(FLG_ID1,WAIPTN_0001,TWF_ORW,p_flgptn,TMO_POL)		
16	*MERCD(ercd) == E_OK*		twai_flg-10,11,17
17	*flgptn == SETPTN_0055*		twai_flg-12
18	ercd = twai_flg(FLF_ID1,WAIPTN_0002,TWF_ANDW,p_flgptn,TMO_POL)		
19	*MERCD(ercd) == E_TMOUT*		
20	ercd = twai_flg(FLF_ID1,WAIPTN_0020,TWF_ORW,p_flgptn,TMO_1)		
21		ercd = dly_tsk(DLYTIM_5)	
22	*MERCD(ercd) == E_TMOUT*		twai_flg-8,16
23	ercd = twai_flg(FLG_ID1,WAIPTN_5500,TWF_ANDW,p_flgptn,TMO_100)		
24		*MERCD(ercd) == E_OK*	
25		ercd = twai_flg(FLG_ID1,WAIPTN_0055,TWF_ORW,p_flgptn,TMO_100)	
26		*MERCD(ercd) == E_ILUSE*	twai_flg-6
27		ercd = set_flg(FLG_ID1,SETPTN_5500)	
28	*MERCD(ercd) == E_OK*		twai_flg-9,14
29	*flgptn =　WAIPTN_5555*		
35	ercd = twai_flg(FLG_ID1,WAIPTN_0002,TWF_ANDW,p_flgptn,TMO_FEVR)		

36		*MERCD(ercd) == E_OK*	
37		ercd = dly_tsk(DLYTIM_1000)	
38		*MERCD(ercd) == E_OK*	
39		ercd = rel_wai(TASK_ID1)	
40	*MERCD(ercd) == E_RLWAI*		twai_flg-7,18
41	ercd = twai_flg(FLG_ID2,WAIPTN_0001,TWF_ORW,p_flgptn,TMO_FEVR)		
42		*MERCD(ercd) == E_OK*	
43		ercd = set_flg(FLG_ID2,SETPTN_5555)	
44	*MERCD(ercd) == E_OK*		twai_flg-10
45	*flgptn == SETPTN_5555*		
46	ercd = pol_flg(FLG_ID2,WAIPTN_0001,TWF_ORW,p_flgptn)		
47	*MERCD(ercd) == E_TMOUT*		
48	ercd = set_flg(FLG_ID2,SETPTN_8000)		
49	*MERCD(ercd) == E_OK*		
50	ercd = twai_flg(FLG_ID2,WAIPTN_8000,TWF_ANDW,p_flgptn,TMO_FEVR)		
51	*MERCD(ercd) == E_OK*		
52	*flgptn == SETPTN_8000*		
53	ercd = pol_flg(FLG_ID2,WAIPTN_8000,TWF_ORW,p_flgptn)		
54	*MERCD(ercd) == E_TMOUT*		twai_flg-13
55	ercd = twai_flg(FLG_ID2,WAIPTN_0001,TWF_ANDW,p_flgptn,TMO_POL)		
56	*MERCD(ercd) == E_TMOUT*		twai_flg-9
57	ext_tsk( )		
58		*MERCD(ercd) == E_OK*	
59		ext_tsk( )	

(注)
No.4　E_PARエラーを検出できるERR_WAIPTNがシステムに存在しない場合は、テストを省略することができる。
No.8　E_PARエラーを検出できるERR_pointerがシステムに存在しない場合は、テストを省略することができる。
No.10　E_PARエラーを検出できるERR_TMOがシステムに存在しない場合は、テストを省略することができる。

DTQ1：データキュー機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_DTQ(ERR_DTQID,{TA_TFIFO,DTQCNT_1,NULL})	
S2	*E_IDエラーを検出できること。*	CRE_DTQ-1
S3	CRE_DTQ(DTQ_ID1,{ERR_DTQATR,DTQCNT_1,NULL})	
S4	*E_RSATRエラーを検出できること。*	CRE_DTQ-2
S5	CRE_DTQ(DTQ_ID1,{TA_FIFO,ERR_DTQCNT,NULL})	
S6	*E_PARエラーを検出できること。*	CRE_DTQ-3
S7	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_1,NULL})	
S8	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_1,NULL})	
S9	*E_OBJエラーを検出できること。*	CRE_DTQ-4

(注)
No.S6　E_PARエラーを検出できるERR_DTQCNTがシステムに存在しない場合は、テストを省略することができる。

DTQ2：データキュー機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})	
S3	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})	
S4	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_4,STKSZ,NULL})	
S5	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_1,NULL})	
S6	CRE_DTQ(DTQ_ID2,{TA_TPRI,DTQCNT_1,NULL})	
S7	CRE_DTQ(DTQ_ID3,{TA_TPRI,DTQCNT_2,NULL})	

—	**<TASK_ID1:DORMANT>**	**<TASK_ID2:DORMANT>**	**<TASK_ID3:DORMANT>**	**<TASK_ID4:READY>**	—
1				ercd = snd_dtq(ERR_DTQID,data)	
2				*MERCD(ercd) == E_ID*	snd_dtq-1
3				ercd = snd_dtq(DTQ_ID1,DATA_55)	
4				*MERCD(ercd) == E_OK*	CRE_DTQ-7
5				p_dataにdataのアドレス設定	
6				ercd = rcv_dtq(DTQ_ID1,p_data)	
7				*MERCD(ercd) == E_OK*	
8				*data == DATA_55*	snd_dtq-3,4,13
9				ercd = act_tsk(TASK_ID3)	
10			ercd = rcv_dtq(DTQ_ID1,p_data)		
11				*MERCD(ercd) == E_OK*	
12				ercd = act_tsk(TASK_ID2)	
13		ercd = rcv_dtq(DTQ_ID1,p_data)			
14				*MERCD(ercd) == E_OK*	
15				ercd = snd_dtq(DTQ_ID1,DATA_33)	
16			*MERCD(ercd) == E_OK*		CRE_DTQ-5 snd_dtq-5,6
17			*data == DATA_33*		snd_dtq-7
18			ercd = slp_tsk( )		
19				*MERCD(ercd) == E_OK*	
20				ercd = snd_dtq(DTQ_ID1,MEMORY)	
21		*MERCD(ercd) == E_OK*			
22		*data = MEMORY*			snd_dtq-14
23		ercd = slp_tsk( )			

24				*MERCD(ercd) == E_OK*	
25				ercd = snd_dtq(DTQ_ID1,DATA_00)	
26				*MERCD(ercd) == E_OK*	
27				ercd = wup_tsk(TASK_ID3)	
28			*MERCD(ercd) == E_OK*		
29			ercd = snd_dtq(DTQ_ID1,DATA_33)		
30				*MERCD(ercd) == E_OK*	
31				ercd = wup_tsk(TASK_ID2)	
32		ercd = snd_dtq(DTQ_ID1,DATA_22)			
33				*MERCD(ercd) == E_OK*	
34				ercd = act_tsk(TASK_ID1)	
35	ercd = snd_dtq(DTQ_ID1,DATA_11)				
36				*MERCD(ercd) == E_OK*	
37				ercd = snd_dtq(DTQ_ID2,DATA_FF)	
38				*MERCD(ercd) == E_OK*	
39				ercd = chg_pri(TASK_ID2,TSKPRI_3)	
40				*MERCD(ercd) == E_OK*	
41				ercd = rcv_dtq(DTQ_ID1,p_data)	
42			*MERCD(ercd) == E_OK*		snd_dtq-9
43			ercd = snd_dtq(DTQ_ID2,DATA_EE)		
44				*MERCD(ercd) == E_OK*	
45				*data == DATA_00*	
46				ercd = rcv_dtq(DTQ_ID1,p_data)	
47		*MERCD(ercd) == E_OK*			snd_dtq-10
48		ercd = snd_dtq(DTQ_ID2,DATA_DD)			
49				MERCD(ercd) == E_OK	
50				*data == DATA_33*	
51				ercd = rel_wai(TASK_ID1)	
52	*MERCD(ercd) == E_RLWAI*				snd_dtq-2
53	ercd = snd_dtq(DTQ_ID2,DATA_CC)				
54				*MERCD(ercd) == E_OK*	
55				ercd = rcv_dtq(DTQ_ID2,p_data)	
56	MERCD(ercd) == E_OK				CRE_DTQ-6 snd_dtq-11
57	ext_tsk( )				
58				*MERCD(ercd) == E_OK*	
59				*data == DATA_FF*	
60				ercd = rcv_dtq(DTQ_ID2,p_data)	
61			*MERCD(ercd) == E_OK*		
62			ext_tsk( )		
63				*MERCD(ercd) == E_OK*	
64				*data == DATA_CC*	
65				ercd = rcv_dtq(DTQ_ID2,p_data)	
66		*MERCD(ercd) == E_OK*			
67		*ext_tsk( )*			
68				*MERCD(ercd) == E_OK*	
69				*data == DATA_EE*	
70				ercd = rcv_dtq(DTQ_ID2,p_data)	
71				*MERCD(ercd) == E_OK*	
72				*data == DATA_DD*	snd_dtq-12
73				ercd = snd_dtq(DTQ_ID3,DATA_11)	
74				*MERCD(ercd) == E_OK*	
75				ercd = snd_dtq(DTQ_ID3,DATA_22)	
76				*MERCD(ercd) == E_OK*	
77				ercd = rcv_dtq(DTQ_ID3,p_data)	
78				*MERCD(ercd) == E_OK*	
79				ercd = rcv_dtq(DTQ_ID3,p_data)	
80				*MERCD(ercd) == E_OK*	
81				*data == DATA_22*	snd_dtq-8
82				ext_tsk( )	

DTQ3：データキュー機能のテスト手順　その3

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
S3	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_0,NULL})		
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	－
1	ercd = psnd_dtq(ERR_DTQID,DATA_55)		
2	*MERCD(ercd) == E_ID*		psnd_dtq-1
3	ercd = psnd_dtq(DTQ_ID1,DATA_55)		
4	*MERCD(ercd) == E_TMOUT*		psnd_dtq-2
5	ercd = rcv_dtq(DTQ_ID1,p_data)		
6		ercd = psnd_dtq(DTQ_ID1,DATA_AA)	
7	*MERCD(ercd) == E_OK*		psnd_dtq-5,6
8	*data == DATA_AA*		psnd_dtq-3,4,7,8
9	ercd = rcv_dtq(DTQ_ID1,p_data)		
10		*MERCD(ercd) == E_OK*	
11		ercd = psnd_dtq(DTQ_ID1,MEMORY)	

12	*MERCD(ercd) == E_OK*		
13	*data == MEMORY*		psnd_dtq-9
14	ext_tsk( )		
15		*MERCD(ercd) == E_OK*	
16		ext_tsk( )	

DTQ4：データキュー機能のテスト手順　その4

No	テ　ス　ト　手　順　内　容			テスト項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	DEF_INH(INHNO_1,{INHATR_1,INTHDR_1})			
S4	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_0,NULL})			
S5	CRE_DTQ(DTQ_ID2,{TA_TFIFO,DTQCNT_2,NULL})			
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<INTHDR_1:割込みハンドラ>**	－
1	ercd = rcv_dtq(DTQ_ID2,p_data)			
2		<INTHDR_1の割込み発生>		
3			ercd = ipsnd_dtq(ERR_DTQID,DATA55)	
4			*MERCD(ercd) == E_ID*	ipsnd_dtq-1
5			ercd = ipsnd_dtq(DTQ_ID1,DATA_AA)	
6			*MERCD(ercd) == E_TMOUT*(注)	ipsnd_dtq-2
7			ercd = ipsnd_dtq(DTQ_ID2,DATA_88)	
8			*MERCD(ercd) == E_OK*	
9			ercd = ipsnd_dtq(DTQ_ID2,MEMORY)	
10			*MERCD(ercd) == E_OK*	非タスクコンテキストから呼び出せるサービスコール-7
11			return	
12	*MERCD(ercd) == E_OK*			ipsnd_dtq-5,6,7
13	*data == DATA_88*			ipsnd_dtq-3,4,8
14	ercd = rcv_dtq(DTQ_ID2,p_data)			
15	*MERCD(ercd) == E_OK*			
16	*data == MEMORY*			ipsnd_dtq-9
17	ext_tsk( )			
18		ext_tsk( )		

(注)

No.6　E_TMOUTエラーの検出を省略することが製品マニュアルに記載されている場合：MERCD(ercd) == E_OK

DTQ5：データキュー機能のテスト手順　その5

No	テ　ス　ト　手　順　内　容				テスト項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S4	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_4,STKSZ,NULL})				
S5	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_1,NULL})				
S6	CRE_DTQ(DTQ_ID2,{TA_TPRI,DTQCNT_1,NULL})				
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2:DORMANT>**	**<TASK_ID3:DORMANT>**	**<TASK_ID4:READY>**	－
1				ercd = tsnd_dtq (ERR_DTQID,DATA_55,TMO_10)	
2				*MERCD(ercd) == E_ID*	tsnd_dtq-1
3				ercd = tsnd_dtq (DTQ_ID1,DATA_55,ERR_TMO)	
4				*MERCD(ercd) == E_PAR*	tsnd_dtq-2
5				ercd = act_tsk(TASK_ID3)	
6			ercd = rcv_dtq(DTQ_ID1,p_data)		
7				*MERCD(ercd) == E_OK*	
8				ercd = tsnd_dtq (DTQ_ID1,DATA_11,TMO_POL)	
9			*MERCD(ercd) == E_OK*		tsnd_dtq-7,8
10			*data == DATA_11*		tsnd_dtq-5,6,9,18
11			ercd = tsnd_dtq (DTQ_ID1,MEMORY,TMO_100)		
12			*MERCD(ercd) == E_OK*		
13			ercd = tsnd_dtq (DTQ_ID1,DATA_77,TMO_200)		
14				*MERCD(ercd) == E_OK*	tsnd_dtq-16
15				ercd = prcv_dtq(DTQ_ID1,p_data)	
16			*MERCD(ercd) == E_OK*		tsnd_dtq-11
17			ercd = slp_tsk( )		
18				*MERCD(ercd) == E_OK*	
19				*data == MEMORY*	tsnd_dtq-19
20				ercd = prcv_dtq(DTQ_ID1,p_data)	
21				*MERCD(ercd) == E_OK*	
22				*data == DATA_77*	tsnd_dtq-10
23				ercd = tsnd_dtq (DTQ_ID1,DATA_11,TMO_POL)	

24				*MERCD(ercd) == E_OK*	
25				ercd = wup_tsk(TASK_ID3)	
26			*MERCD(ercd) == E_OK*		
27			ercd = tsnd_dtq (DTQ_ID1,DATA_22,TMO_100)		
28				*MERCD(ercd) == E_OK*	
29				ercd = act_tsk(TASK_ID2)	
30		ercd = tsnd_dtq (DTQ_ID1,DATA_33,TMO_200)			
31				*MERCD(ercd) == E_OK*	
32				ercd = act_tsk(TASK_ID1)	
33	ercd = tsnd_dtq (DTQ_ID1,DATA_44,TMO_FEVR)				
34				*MERCD(ercd) == E_OK*	
35				ercd = psnd_dtq (DTQ_ID2,DATA_66)	
36				*MERCD(ercd) == E_OK*	
37				ercd = dly_tsk(DLYTIM_50)	
38				*MERCD(ercd) == E_OK*	
39				ercd = prcv_dtq(DTQ_ID1,p_data)	
40			*MERCD(ercd) == E_OK*		tsnd_dtq-15
41			ercd = slp_tsk( )		
42				*MERCD(ercd) == E_OK*	
43				*data == DATA_11*	
44				ercd = dly_tsk(DLYTIM_1000)	
45		*MERCD(ercd) == E_TMOUT*			tsnd_dtq-4,12
46		ercd = slp_tsk( )			
47				*MERCD(ercd) == E_OK*	
48				ercd = rel_wai(TASK_ID1)	
49	*MERCD(ercd) == E_RLWAI*				tsnd_dtq-3,17
50	ercd = slp_tsk( )				
51				*MERCD(ercd) == E_OK*	
52				ercd = chg_pri(TASK_ID2,TSKPRI_3)	
53				*MERCD(ercd) == E_OK*	
54				ercd = wup_tsk(TASK_ID3)	
55			*MERCD(ercd) == E_OK*		
56			ercd = tsnd_dtq (DTQ_ID2,DATA_77,TMO_FEVR)		
57				*MERCD(ercd) == E_OK*	
58				ercd = wup_tsk(TASK_ID2)	
59		*MERCD(ercd) == E_OK*			
60		ercd = tsnd_dtq (DTQ_ID2,DATA_88,TMO_100)			
61				*MERCD(ercd) == E_OK*	
62				ercd = wup_tsk(TASK_ID1)	
63	*MERCD(ercd) == E_OK*				
64	ercd = tsnd_dtq (DTQ_ID2,DATA_99,TMO_100)				
65				*MERCD(ercd) == E_OK*	
66				ercd = prcv_dtq(DTQ_ID2,p_data)	
67	*MERCD(ercd) == E_OK*				tsnd_dtq-13
68	ext_tsk( )				
69				*MERCD(ercd) == E_OK*	
70				*data == DATA_66*	
71				ercd = prcv_dtq(DTQ_ID2,p_data)	
72			*MERCD(ercd) == E_OK*		
73			ext_tsk( )		
74				*MERCD(ercd) == E_OK*	
75				*data == DATA_99*	
76				ercd = prcv_dtq(DTQ_ID2,p_data)	
77		*MERCD(ercd) == E_OK*			
78		ext_tsk( )			
79				*MERCD(ercd) == E_OK*	
80				*data == DATA_77*	
81				ercd = prcv_dtq(DTQ_ID2,p_data)	
82				*MERCD(ercd) == E_OK*	
83				*data == DATA_88*	tsnd_dtq-14
84				ext_tsk( )	

(注)

No.4　E_PARエラーを検出できるERR_TMOがシステムに存在しない場合は、テストを省略することができる。

DTQ6：データキュー機能のテスト手順　その6

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
S3	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_0,NULL})		
S4	CRE_DTQ(DTQ_ID2,{TA_TFIFO,DTQCNT_1,NULL})		
S5	CRE_DTQ(DTQ_ID3,{TA_TFIFO,DTQCNT_2,NULL})		
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	－
1	ercd = fsnd_dtq(ERR_DTQID,DATA_00)		
2	*MERCD(ercd) == E_ID*		fsnd_dtq-1
3	ercd = fsnd_dtq(DTQ_ID1,DATA_00)		
4	*MERCD(ercd) == E_ILUSE*		fsnd_dtq-2
5	ercd = fsnd_dtq(DTQ_ID2,DATA_11)		
6	*MERCD(ercd) == E_OK*		
7	ercd = fsnd_dtq(DTQ_ID2,DATA_22)		
8	*MERCD(ercd) == E_OK*		
9	ercd = prcv_dtq(DTQ_ID2,p_data)		
10	*MERCD(ercd) == E_OK*		
11	*data == DATA_22*		fsnd_dtq-7
12	ercd = rcv_dtq(DTQ_ID2,p_data)		
13		ercd = fsnd_dtq(DTQ_ID2,DATA_33)	
14	*MERCD(ercd) == E_OK*		fsnd_dtq-3,4
15	*data == DATA_33*		fsnd_dtq-5
16	ercd = fsnd_dtq(DTQ_ID3,DATA_44)		
17	*MERCD(ercd) == E_OK*		
18	ercd = fsnd_dtq(DTQ_ID3,DATA_55)		
19	*MERCD(ercd) == E_OK*		
20	ercd = rcv_dtq(DTQ_ID3,p_data)		
21	*MERCD(ercd) == E_OK*		
22	*data == DATA_44*		
23	ercd = rcv_dtq(DTQ_ID3,p_data)		
24	*MERCD(ercd) == E_OK*		
25	*data == DATA_55*		fsnd_dtq-6
26	ercd = fsnd_dtq(DTQ_ID2,DATA_66)		
27	*MERCD(ercd) == E_OK*		
28	ercd = rcv_dtq(DTQ_ID2,p_data)		
29	*MERCD(ercd) == E_OK*		
30	*data == DATA_66*		
31	ext_tsk(　)		
32		*MERCD(ercd) == E_OK*	
33		ext_tsk(　)	

DTQ7：データキュー機能のテスト手順　その7

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_0,NULL})			
S4	CRE_DTQ(DTQ_ID2,{TA_TFIFO,DTQCNT_1,NULL})			
S5	CRE_DTQ(DTQ_ID3,{TA_TFIFO,DTQCNT_1,NULL})			
S6	CRE_DTQ(DTQ_ID4,{TA_TFIFO,DTQCNT_2,NULL})			
S7	DEF_INH(INHNO_1,{INHATR_1,INTHDR_1})			
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<INTHDR_1:割込みハンドラ>**	－
1	ercd = rcv_dtq(DTQ_ID2,p_data)			
2		<INTHDR_1の割込み発生>		
3			ercd = ifsnd_dtq(ERR_DTQID,DATA_77)	
4			*MERCD(ercd) == E_ID*	ifsnd_dtq-1
5			ercd = ifsnd_dtq(DTQ_ID1,DATA_77)	
6			*MERCD(ercd) == E_ILUSE*	ifsnd_dtq-2
7			ercd = ifsnd_dtq(DTQ_ID2,DATA_88)	
8			*MERCD(ercd) == E_OK*	
9			ercd = ifsnd_dtq(DTQ_ID3,DATA_99)	
10			*MERCD(ercd) == E_OK*	
11			ercd = ifsnd_dtq(DTQ_ID3,DATA_AA)	
12			*MERCD(ercd) == E_OK*	非タスクコンテキストから呼び出せるサービスコール-8
13			ercd = ifsnd_dtq(DTQ_ID4,DATA_11)	
14			*MERCD(ercd) == E_OK*	
15			ercd = ifsnd_dtq(DTQ_ID4,DATA_22)	
16			*MERCD(ercd) == E_OK*	
17			return	
18	*MERCD(ercd) == E_OK*			ifsnd_dtq-3,4
19	*data == DATA_88*			ifsnd_dtq-5
20	ext_tsk(　)			
21		ercd = prcv_dtq(DTQ_ID3,p_data)		
22		*MERCD(ercd) == E_OK*		
23		*data == DATA_AA*		ifsnd_dtq-7
24		ercd = prcv_dtq(DTQ_ID4,p_data)		

25		*MERCD(ercd) == E_OK*		
26		*data == DATA_11*		
27		ercd = prcv_dtq(DTQ_ID4,p_data)		
28		*MERCD(ercd) == E_OK*		
29		*data == DATA_22*		ifsnd_dtq-6
30		ext_tsk( )		

DTQ8：データキュー機能のテスト手順　その8

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})			
S4	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_1,NULL})			
S5	CRE_DTQ(DTQ_ID2,{TA_TFIFO,DTQCNT_0,NULL})			
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<TASK_ID3:READY>**	－
1	ercd = rcv_dtq(ERR_DTQID,p_data)			
2	*MERCD(ercd) == E_ID*			rcv_dtq-1
3	ercd = rcv_dtq(DTQ_ID1,ERR_pointer)			
4	*MERCD(ercd) == E_PAR*			rcv_dtq-2
5	ercd = snd_dtq(DTQ_ID1,DATA_01)			
6	*MERCD(ercd) == E_OK*			
7	ercd = rcv_dtq(DTQ_ID1,p_data)			
8	*MERCD(ercd) == E_OK*			
9	*data == DATA_01*			rcv_dtq-4
10	ercd = snd_dtq(DTQ_ID1,DATA_02)			
11	*MERCD(ercd) == E_OK*			
12	ercd = snd_dtq(DTQ_ID1,DATA_03)			
13		ercd = rcv_dtq(DTQ_ID1,p_data)		
14	*MERCD(ercd) == E_OK*			rcv_dtq-5,6
15	ercd = slp_tsk( )			
16		*MERCD(ercd) == E_OK*		
17		*data == DATA_02*		
18		ercd = rcv_dtq(DTQ_ID1,p_data)		
19		*MERCD(ercd) == E_OK*		
20		*data == DATA_03*		rcv_dtq-5
21		ercd = wup_tsk(TASK_ID1)		
22	*MERCD(ercd) == E_OK*			
23	ercd = snd_dtq(DTQ_ID2,DATA_04)			
24		*MERCD(ercd) == E_OK*		
25		ercd = rcv_dtq(DTQ_ID2,p_data)		
26	*MERCD(ercd) == E_OK*			rcv_dtq-7,8
27	ercd = rcv_dtq(DTQ_ID1,p_data)			
28		*MERCD(ercd) == E_OK*		
29		*data == DATA_04*		rcv_dtq-9
30		ercd = rcv_dtq(DTQ_ID1,p_data)		
31			ercd = snd_dtq(DTQ_ID1,DATA_05)	
32	*MERCD(ercd) == E_OK*			rcv_dtq-10
33	*data == DATA_05*			
34	ercd = rel_wai(TASK_ID2)			
35	*MERCD(ercd) == E_OK*			
36	ext_tsk( )			
37		*MERCD(ercd) == E_RLWAI*		rcv_dtq-3,11
38		ext_tsk( )		
39			*MERCD(ercd) == E_OK*	
40			ext_tsk( )	

(注)

No.4　E_PARエラーを検出できるERR_pointerがシステムに存在しない場合は、テストを省略することができる。

DTQ9：データキュー機能のテスト手順　その9

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})			
S4	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_0,NULL})			
S5	CRE_DTQ(DTQ_ID2,{TA_TFIFO,DTQCNT_1,NULL})			
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<TASK_ID3:READY>**	－
1	ercd = prcv_dtq(ERR_DTQID,p_data)			
2	*MERCD(ercd) == E_ID*			prcv_dtq-1
3	p_dataにE_PARエラーを検出する値を設定			
4	ercd = prcv_dtq(DTQ_ID1,p_data)			
5	*MERCD(ercd) == E_PAR*			prcv_dtq-2
6	ercd = prcv_dtq(DTQ_ID1,p_data)			
7	*MERCD(ercd) == E_TMOUT*			prcv_dtq-3
8	ercd = snd_dtq(DTQ_ID1,DATA_99)			
9		ercd = prcv_dtq(DTQ_ID1,p_data)		
10	*MERCD(ercd) == E_OK*			
11	ercd = snd_dtq(DTQ_ID2,DATA_11)			

12	*MERCD(ercd) == E_OK*			
13	ercd = snd_dtq(DTQ_ID2,DATA_22)			
14		*MERCD(ercd) == E_OK*		
15		*data == DATA_99*		prcv_dtq-4
16		ercd = snd_dtq(DTQ_ID2,DATA_33)		
17			ercd = prcv_dtq(DTQ_ID2,p_data)	
18	*MERCD(ercd) == E_OK*			prcv_dtq-5,6
19	ext_tsk( )			
20			*MERCD(ercd) == E_OK*	
21			*data == DATA_11*	
22			ercd = prcv_dtq(DTQ_ID2,p_data)	
23		*MERCD(ercd) == E_OK*		
24		ext_tsk( )		
25			*MERCD(ercd) == E_OK*	
26			*data == DATA_22*	
27			ercd = prcv_dtq(DTQ_ID2,p_data)	
28			*MERCD(ercd) == E_OK*	
29			*data == DATA_33*	prcv_dtq-5
30			ext_tsk( )	

(注)

No.5　E_PARエラーを検出できるp_dataがシステムに存在しない場合は、テストを省略することができる。

DTQ10：データキュー機能のテスト手順　その10

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})			
S4	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_1,NULL})			
S5	CRE_DTQ(DTQ_ID2,{TA_TFIFO,DTQCNT_0,NULL})			
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2:DORMANT>**	**<TASK_ID3:READY>**	－
1			ercd = trcv_dtq(ERR_DTQ_ID,p_data,TMO_POL)	
2			*MERCD(ercd) == E_ID*	trcv_dtq-1
3			p_dataにE_PARエラーを検出する値を設定する	
4			ercd = trcv_dtq(DTQ_ID1,p_data,TMO_POL)	
5			*MERCD(ercd) == E_PAR*	trcv_dtq-2
6			ercd = trcv_dtq(DTQ_ID1,p_data,ERR_TMO)	
7			*MERCD(ercd) == E_PAR*	trcv_dtq-3
8			ercd = snd_dtq(DTQ_ID1,DATA_01)	
9			*MERCD(ercd) == E_OK*	
10			ercd = trcv_dtq(DTQ_ID1,p_data,TMO_POL)	
11			*MERCD(ercd) == E_OK*	trcv_dtq-15
12			*data == DATA_01*	trcv_dtq-6
13			ercd = act_tsk(TASK_ID2)	
14		ercd = snd_dtq(DTQ_ID1,DATA_02)		
15		*MERCD(ercd) == E_OK*		
16		ercd = snd_dtq(DTQ_ID1,DATA_03)		
17			*MERCD(ercd) == E_OK*	
18			ercd = trcv_dtq(DTQ_ID1,p_data,TMO_1)	
19		*MERCD(ercd) == E_OK*		trcv_dtq-7,8
20		ercd = slp_tsk( )		
21			*MERCD(ercd) == E_OK*	
22			*data == DATA_02*	
23			ercd = trcv_dtq(DTQ_ID1,p_data,TMO_FEVR)	
24			*MERCD(ercd) == E_OK*	
25			*data == DATA_03*	trcv_dtq-7
26			ercd = wup_tsk(TASK_ID2)	
27		*MERCD(ercd) == E_OK*		
28		ercd = snd_dtq(DTQ_ID2,DATA_04)		
29			*MERCD(ercd) == E_OK*	
30			ercd = trcv_dtq(DTQ_ID2,p_data,TMO_100)	
31		*MERCD(ercd) == E_OK*		trcv_dtq-9,10
32		ercd = trcv_dtq(DTQ_ID1,p_data,TMO_100)		
33			*MERCD(ercd) == E_OK*	
34			*data == DATA_04*	trcv_dtq-11
35			ercd = act_tsk(TASK_ID1)	
36	ercd = trcv_dtq(DTQ_ID1,p_data,TMO_200)			
37			*MERCD(ercd) == E_OK*	
38			ercd = dly_tsk(DLYTIM_50)	
39			*MERCD(ercd) == E_OK*	
40			ercd = snd_dtq(DTQ_ID1,DATA_05)	
41		*MERCD(ercd) == E_OK*		trcv_dtq-12,14
42		*data == DATA_05*		
43		ercd = trcv_dtq(DTQ_ID1,p_data,TMO_FEVR)		
44			*MERCD(ercd) == E_OK*	
45			ercd = dly_tsk(DLYTIM_1000)	
46	*MERCD(ercd) == E_TMOUT*			trcv_dtq-5
47	ext_tsk( )			
48			*MERCD(ercd) == E_OK*	

49			ercd = rel_wai(TASK_ID2)	
50		*MERCD(ercd) == E_RLWAI*		trcv_dtq-4,13,16
51		ext_tsk( )		
52			*MERCD(ercd) == E_OK*	
53			ext_tsk( )	

(注)

No.5　E_PARエラーを検出できるp_dataがシステムに存在しない場合は、テストを省略することができる。

No.7　E_PARエラーを検出できるERR_TMOがシステムに存在しない場合は、テストを省略することができる。

MBX1：メールボックス機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_MBX(ERR_MBXID,{TA_TFIFO|TA_MFIFO,MAXMPRI_5,NULL})	
S2	*E_IDエラーを検出できること。*	CRE_MBX-1
S3	CRE_MBX(MBX_ID1,{ERR_MBXATR,MAXMPRI_5,NULL})	
S4	*E_RSATRエラーを検出できること。*	CRE_MBX-2
S5	CRE_MBX(MBX_ID1_2,{TA_TFIFO|TA_MFIFO,ERR_MAXMPRI,NULL})	
S6	*E_PARエラーを検出できること。*	CRE_MBX-3
S7	CRE_MBX(MBX_ID1,{TA_TFIFO|TA_MFIFO,MAXMPRI_5,NULL})	
S8	CRE_MBX(MBX_ID1,{TA_TFIFO|TA_MFIFO,MAXMPRI_5,NULL})	
S9	*E_OBJエラーを検出できること。*	CRE_MBX-4
S10	CRE_MBX(MBX_ID5,{TA_TPRI|TA_MPRI,MAXMPRI_0,NULL})	
S11	*E_PARエラーを検出できること。*	CRE_MBX-9

MBX2：メールボックス機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S4	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_4,STKSZ,NULL})				
S5	CRE_MBX(MBX_ID1,{TA_TFIFO|TA_MFIFO,MAXMPRI_3,NULL})				
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2:DORMANT>**	**<TASK_ID3:DORMANT>**	**<TASK_ID4:READY>**	－
1				ercd = snd_mbx(ERR_MBXID,&msg1)	
2				*MERCD(ercd) == E_ID*	snd_mbx-1
3				pk_msgにE_PARエラーを検出する値を設定する	
4				ercd = snd_mbx(MBX_ID1,pk_msg)	
5				*MERCD(ercd) == E_PAR*	snd_mbx-2
6				ercd = rcv_mbx(ERR_MBXID,ppk_msg)	
7				*MERCD(ercd) == E_ID*	rcv_mbx-1
8				ppk_msgにE_PARエラーを検出する値を設定する	
9				ercd = rcv_mbx(MBX_ID1,ppk_msg)	
10				*MERCD(ercd) == E_PAR*	rcv_mbx-2
11				ercd = prcv_mbx(ERR_MBXID,ppk_msg)	
12				*MERCD(ercd) == E_ID*	prcv_msg-1
13				ercd = prcv_mbx(MBX_ID1,ppk_msg)	
14				*MERCD(ercd) == E_PAR*	prcv_msg-2
15				ppk_msgに正常値を設定	
16				ercd = prcv_mbx(MBX_ID1,ppk_msg)	
17				*MERCD(ercd) == E_TMOUT*	prcv_mbx-3
18				ercd = snd_mbx(MBX_ID1,&msg1)	
19				*MERCD(ercd) == E_OK*	CRE_MBX-5
20				ercd = rcv_mbx(MBX_ID1,ppk_msg)	
21				*MERCD(ercd) == E_OK*	
22				*ppk_msg == msg1の先頭アドレス*	rcv_mbx-4
23				ercd = snd_mbx(MBX_ID1,&msg2)	
24				*MERCD(ercd) == E_OK*	
25				ercd = prcv_mbx(MBX_ID1,ppk_msg)	
26				*MERCD(ercd) == E_OK*	
27				*ppk_msg==msg2の先頭アドレス*	prcv_mbx-4
28				ercd = snd_mbx(MBX_ID1,&msg3)	
29				*MERCD(ercd) == E_OK*	
30				ercd = snd_mbx(MBX_ID1,&msg2)	
31				*MERCD(ercd) == E_OK*	
32				ercd = snd_mbx(MBX_ID1,&msg1)	
33				*MERCD(ercd) == E_OK*	
34				ercd = prcv_mbx(MBX_ID1,ppk_msg)	
35				*MERCD(ercd) == E_OK*	
36				*ppk_msg==msg3の先頭アドレス*	snd_mbx-7
37				ercd = prcv_mbx(MBX_ID1,ppk_msg)	
38				*MERCD(ercd) == E_OK*	
39				*ppk_msg==msg2の先頭アドレス*	snd_mbx-7
40				ercd = prcv_mbx(MBX_ID1,ppk_msg)	
41				*MERCD(ercd) == E_OK*	
42				*ppk_msg==msg1の先頭アドレス*	snd_mbx-7
43				ercd = act_tsk(TASK_ID3)	

44			ercd = rcv_mbx(MBX_ID1,ppk_msg)		
45				*ercd = OK*	
46				ercd = act_tsk(TASK_ID2)	
47		ercd = rcv_mbx(MBX_ID1,ppk_msg)			
48				*MERCD(ercd) == E_OK*	
49				ercd = act_tsk(TASK_ID1)	
50	ercd = rcv_mbx(MBX_ID1,ppk_msg)				
51				*MERCD(ercd) == E_OK*	
52				ercd = snd_mbx(MBX_ID1,&msg1)	
53			*MERCD(ercd) == E_OK*		snd_mbx-5 rcv_mbx-5
54			*ppk_msg==msg1の先頭アドレス*		snd_mbx-4,6
55			ext_tsk( )		
56				*MERCD(ercd) == E_OK*	
57				ercd = snd_mbx(MBX_ID1,&msg2)	
58		*MERCD(ercd) == E_OK*			rcv_mbx-6
59		*ppk_msg==msg2の先頭アドレス*			
60		ext_tsk( )			
61				*MERCD(ercd) == E_OK*	
62				ercd = rel_wai(TASK_ID1)	
63	*MERCD(ercd) == E_RLWAI*				rcv_mbx-3
64	ext_tsk( )				
65				*MERCD(ercd) == E_OK*	
66				ext_tsk( )	

(注)
No.5　E_PARエラーを検出できるpk_msgがシステムに存在しない場合は、テストを省略することができる。
No.10　E_PARエラーを検出できるppk_msgがシステムに存在しない場合は、テストを省略することができる。
No.14　E_PARエラーを検出できるppk_msgがシステムに存在しない場合は、テストを省略することができる。

MBX3：メールボックス機能のテスト手順　その3

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL}) (優先度付きメッセージを4個用意し、msg1,msg21,msg22,msg3とすること。)	
S2	CRE_MBX(MBX_ID1,{TA_TFIFO|TA_MPRI,MAXMPRI_3,NULL})	
－	**<TASK_ID1:READY>**	－
1	msg3のmsgpriにMSGPRI_3を設定	
2	ercd = snd_mbx(MBX_ID1,&msg3)	
3	*MERCD(ercd) == E_OK*	CRE_MBX-6
4	msg21のmsgpriにMSGPRI_2を設定	
5	ercd = snd_mbx(MBX_ID1,&msg21)	
6	*MERCD(ercd) == E_OK*	
7	msg22のmsgpriにMSGPRI_2を設定	
8	ercd = snd_mbx(MBX_ID1,&msg22)	
9	*MERCD(ercd) == E_OK*	snd_mbx-9
10	msg1のmsgpriにMSGPRI_1を設定	
11	ercd = snd_mbx(MBX_ID1,&msg1)	
12	*MERCD(ercd) == E_OK*	
13	ercd = prcv_mbx(MBX_ID1,ppk_msg)	
14	*MERCD(ercd) == E_OK*	
15	*ppk_msg==msg1の先頭アドレス*	snd_mbx-8
16	ercd = prcv_mbx(MBX_ID1,ppk_msg)	
17	*MERCD(ercd) == E_OK*	
18	*ppk_msg==msg21の先頭アドレス*	
19	ercd = prcv_mbx(MBX_ID1,ppk_msg)	
20	*MERCD(ercd) == E_OK*	
21	*ppk_msg==msg22の先頭アドレス*	
22	ercd = prcv_mbx(MBX_ID1,ppk_msg)	
23	*MERCD(ercd) == E_OK*	
24	*ppk_msg==msg3の先頭アドレス*	CRE_MBX-10
25	msg1のmsgpriにERR_MSGPRIを設定	
26	ercd = snd_mbx(MBX_ID1,&msg1)	
27	*MERCD(ercd) == E_PAR*	snd_mbx-3
28	ext_tsk( )	

MBX4：メールボックス機能のテスト手順　その4

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2_1,{TA_HLNG,EXINF_2_1,TASK2_1,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID2_2,{TA_HLNG,EXINF_2_2,TASK2_2,ITSKPRI_2,STKSZ,NULL})				
S4	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL}) (優先度付きメッセージを３個用意し、msg1,msg21,msg22とすること。)				
S5	CRE_MBX(MBX_ID1,{TA_TPRI|TA_MFIFO,MAXMPRI_3,NULL})				
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2_1:DORMANT>**	**<TASK_ID2_2:DORMANT>**	**<TASK_ID3:READY>**	－
1				ercd = act_tsk(TASK_ID2_1)	
2		ercd = rcv_mbx(MBX_ID1,ppk_msg)			
3				*MERCD(ercd) == E_OK*	

4				ercd = act_tsk(TASK_ID2_2)	
5			ercd = rcv_mbx(MBX_ID1,ppk_msg)		
6				*MERCD(ercd) == E_OK*	
7				ercd = act_tsk(TASK_ID1)	
8	ercd = rcv_mbx(MBX_ID1,ppk_msg)				
9				*MERCD(ercd) == E_OK*	
10				ercd = snd_mbx(MBX_ID1,&msg1)	
11	*MERCD(ercd) == E_OK*				CRE_MBX-7 rcv_mbx-7
12	*ppk_msg==msg1の先頭アドレス*				
13	ext_tsk( )				
14				*MERCD(ercd) == E_OK*	
15				ercd = snd_mbx(MBX_ID1,&msg21)	
16		*MERCD(ercd) == E_OK*			
17		*ppk_msg==msg21の先頭アドレス*			
18		ext_tsk( )			
19				*MERCD(ercd) == E_OK*	
20				ercd = snd_mbx(MBX_ID1,&msg22)	
21			*MERCD(ercd) == E_OK*		rcv_mbx-8
22			*ppk_msg==msg22の先頭アドレス*		
23			ext_tsk( )		
24				*MERCD(ercd) == E_OK*	
25				ext_tsk( )	

MBX5：メールボックス機能のテスト手順　その5

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})	
S3	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})	
S4	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_4,STKSZ,NULL}) (優先度付きメッセージを３個用意し、msg1,msg2,msg3とすること。)	
S5	CRE_MBX(MBX_ID1,{TA_TFIFO|TA_MFIFO,MAXMPRI_3,NULL})	

－	<TASK_ID1:DORMANT>	<TASK_ID2:DORMANT>	<TASK_ID3:DORMANT>	<TASK_ID4:READY>	－
1				ercd = trcv_mbx (ERR_MBXID,ppk_msg,TMO_10)	
2				*MERCD(ercd) == E_ID*	trcv_mbx-1
3				ppk_msgにE_PARエラーを検出する値を設定する	
4				ercd = trcv_mbx (MBX_ID1,ppk_msg,TMO_10)	
5				*MERCD(ercd) == E_PAR*	trcv_mbx-2
6				ercd = trcv_mbx (MBX_ID1,ppk_msg,ERR_TMO)	
7				*MERCD(ercd) == E_PAR*	trcv_mbx-3
8				ercd = snd_mbx(MBX_ID1,&msg1)	
9				*MERCD(ercd) == E_OK*	
10				ercd = trcv_mbx (MBX_ID1,ppk_msg,TMO_POL)	
11				*MERCD(ercd) == E_OK*	trcv_mbx-12
12				*ppk_msg==msg1の先頭アドレス*	trcv_mbx-6
13				ercd = act_tsk(TASK_ID3)	
14			ercd = trcv_mbx (MBX_ID1,ppk_msg,TMO_100)		
15				*MERCD(ercd) == E_OK*	
16				ercd = act_tsk(TASK_ID2)	
17		ercd = trcv_mbx (MBX_ID1,ppk_msg,TMO_200)			
18				*MERCD(ercd) == E_OK*	
19				ercd = act_tsk(TASK_ID1)	
20	ercd = trcv_mbx (MBX_ID1,ppk_msg,TMO_FEVR)				
21				*MERCD(ercd) == E_OK*	
22				ercd = dly_tsk(DLYTIM_50)	
23				*MERCD(ercd) == E_OK*	
24				ercd = snd_mbx(MBX_ID1,&msg1)	
25			*MERCD(ercd) == E_OK*		trcv_mbx-7,11
26			*ppk_msg==msg1の先頭アドレス*		
27			ext_tsk( )		
28				*MERCD(ercd) == E_OK*	
29				ercd = dly_tsk(DLYTIM_300)	
30		*MERCD(ercd) == E_TMOUT*			trcv_mbx-5
31		ext_tsk( )			
32				*MERCD(ercd) == E_OK*	
33				ercd = dly_tsk(DLYTIM_1000)	
34				*MERCD(ercd) == E_OK*	
35				ercd = rel_wai(TASK_ID1)	
36	*MERCD(ercd) == E_RLWAI*				trcv_mbx-4,8,13
37	ext_tsk( )				

38				*MERCD(ercd) == E_OK*	
39				ext_tsk( )	

(注)

No.5　E_PARエラーを検出できるppk_msgがシステムに存在しない場合は、テストを省略することができる。

No.7　E_PARエラーを検出できるppk_msgがシステムに存在しない場合は、テストを省略することができる。

MBX6：メールボックス機能のテスト手順　その6

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2_1,{TA_HLNG,EXINF_2_1,TASK2_1,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID2_2,{TA_HLNG,EXINF_2_2,TASK2_2,ITSKPRI_2,STKSZ,NULL})				
S4	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL}) (優先度付きメッセージを３個用意し、msg1,msg21,msg22とすること。)				
S5	CRE_MBX(MBX_ID1,{TA_TPRI|TA_MPRI,MAXMPRI_3,NULL})				
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2_1:DORMANT>**	**<TASK_ID2_2:DORMANT>**	**<TASK_ID3:READY>**	－
1				ercd = act_tsk(TASK_ID2_1)	
2		ercd = trcv_mbx (MBX_ID1,ppk_msg,TMO_100)			
3				*MERCD(ercd) == E_OK*	
4				ercd = act_tsk(TASK_ID2_2)	
5			ercd = trcv_mbx (MBX_ID1,ppk_msg,TMO_FEVR)		
6				*MERCD(ercd) == E_OK*	
7				ercd = act_tsk(TASK_ID1)	
8	ercd = trcv_mbx (MBX_ID1,ppk_msg,TMO_200)				
9				*MERCD(ercd) == E_OK*	
10				msg1のmsgpriにMSGPRI_1を設定	
11				ercd = snd_mbx(MBX_ID1,&msg1)	
12	*MERCD(ercd) == E_OK*				trcv_mbx-9
13	*ppk_msg==msg1の先頭アドレス*				
14	ext_tsk( )				
15				*MERCD(ercd) == E_OK*	
16				msg21のmsgpriにMSGPRI_2を設定	
17				ercd = snd_mbx(MBX_ID1,&msg21)	
18		*MERCD(ercd) == E_OK*			CRE_MBX-8
19		*ppk_msg==msg21の先頭アドレス*			
20		ext_tsk( )			
21				*MERCD(ercd) == E_OK*	
22				msg22のmsgpriにMSGPRI_2を設定	
23				ercd = snd_mbx(MBX_ID1,&msg22)	
24			*MERCD(ercd) == E_OK*		trcv_mbx-10
25			*ppk_msg==msg22の先頭アドレス*		
26			ext_tsk( )		
27				*MERCD(ercd) == E_OK*	
28				ext_tsk( )	

MPF1：固定長メモリプール機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_MPF(ERR_MPFID,{TA_TFIFO,BLKCNT_2048,BLKSZ_256,NULL})	
S2	*E_IDエラーを検出できること*	CRE_MPF-1
S3	CRE_MPF(MPF_ID1,{ERR_MPFATR,BLKCNT_10,BLKSZ_10,NULL})	
S4	*E_RSATRエラーを検出できること。*	CRE_MPF-2
S5	CRE_MPF(MPF_ID1,{TA_TFIFO,BLKCNT_0,BLKSZ_10,NULL})	
S6	*E_PARエラーを検出できること。*	CRE_MPF-3
S7	CRE_MPF(MPF_ID1,{TA_TFIFO,BLKCNT_10,BLKSZ_0,NULL})	
S8	*E_PARエラーを検出できること。*	CRE_MPF-4
S9	CRE_MPF(MPF_ID1,{TA_TFIFO,BLKCNT_1,BLKSZ_10,NULL})	
S10	CRE_MPF(MPF_ID1,{TA_TFIFO,BLKCNT_1,BLKSZ_10,NULL})	
S11	*E_OBJエラーを検出できること*	CRE_MPF-5
S12	CRE_MPF(MPF_ID2,{TA_TPRI,BLKCNT_1,BLKSZ_10,NULL})	CRE_MPF-7

MPF2：固定長メモリプール機能のテスト手順　その2（タスク単位で変数blkを設けること）

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S4	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_4,STKSZ,NULL})				
S5	CRE_MPF(MPF_ID1,{TA_TFIFO,BLKCNT_1,BLKSZ_10,NULL})				
S6	CRE_MPF(MPF_ID2,{TA_TFIFO,BLKCNT_1,BLKSZ_10,NULL})				
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2:DORMANT>**	**<TASK_ID3:DORMANT>**	**<TASK_ID4:READY>**	－
1				ercd = get_mpf(ERR_MPFID,p_blk)	
2				*MERCD(ercd) == E_ID*	get_mpf-1
3				ercd = pget_mpf(ERR_MPFID,p_blk)	
4				*MERCD(ercd) == E_ID*	pget_mpf-1

5				p_blkにE_PARエラーを検出する値を設定する	
6				ercd = get_mpf(MPF_ID1,p_blk)	
7				*MERCD(ercd) == E_PAR*	get_mpf-2
8				ercd = pget_mpf(MPF_ID1,p_blk)	
9				*MERCD(ercd) == E_PAR*	pget_mpf-2
10				ercd = rel_mpf(ERR_MPFID,blk)	
11				*MERCD(ercd) == E_ID*	rel_mpf-1
12				blkにメモリブロックの先頭アドレス以外を設定する	
13				ercd = rel_mpf(MPF_ID1,blk)	
14				*MERCD(ercd) == E_PAR*	rel_mpf-3
15				ercd = pget_mpf(MPF_ID1,&blk)	
16				*MERCD(ercd) == E_OK*	CRE_MPF-6 pget_mpf-4
17				*blk==メモリブロックの先頭アドレス*	pget_mpf-4
18				ercd = rel_mpf(MPF_ID2,blk)	
19				*MERCD(ercd) == E_PAR*	rel_mpf-2
20				ercd = pget_mpf(MPF_ID1,p_blk)	
21				*MERCD(ercd) == E_TMOUT*	pget_mpf-3,5
22				ercd = rel_mpf(MPF_ID1,blk)	
23				*MERCD(ercd) == E_OK*	
24				ercd = get_mpf(MPF_ID1,p_blk)	
25				*MERCD(ercd) == E_OK*	get_mpf-4
26				*blk==メモリブロックの先頭アドレス*	get_mpf-4
27				ercd = act_tsk(TASK_ID3)	
28			ercd = get_mpf(MPF_ID1,&blk)		
29				*MERCD(ercd) == E_OK*	
30				ercd = act_tsk(TASK_ID2)	
31		ercd = get_mpf(MPF_ID1,&blk)			
32				*MERCD(ercd) == E_OK*	
33				ercd = act_tsk(TASK_ID1)	
34	ercd = get_mpf(MPF_ID1,&blk)				
35				*MERCD(ercd) == E_OK*	
36				ercd = rel_mpf(MPF_ID1,blk)	
37			*MERCD(ercd) == E_OK*		CRE_MPF-8,9 get_mpf-5
38			*blk==メモリブロック先頭アドレス*		
39			ercd = rel_mpf(MPF_ID1,blk)		
40		*MERCD(ercd) == E_OK*			get_mpf-6 rel_mpf-6
41		*blk==メモリブロック先頭アドレス* *(blkはNo.38で獲得したアドレスと同一)*			rel_mpf-4,7
42		ercd = rel_wai(TASK_ID1)			
43	*MERCD(ercd) == E_RLWAI*				get_mpf-3
44	ext_tsk( )				
45		*MERCD(ercd) == E_OK*			
46		ercd = rel_mpf(MPF_ID1,blk)			
47		*MERCD(ercd) == E_OK*			
48		ext_tsk( )			
49			*MERCD(ercd) == E_OK*		rel_mpf-5
50			ext_tsk( )		
51				*MERCD(ercd) == E_OK*	
52				ext_tsk( )	

(注)
No.7　E_PARエラーを検出できるp_blkがシステムに存在しない場合は、テストを省略することができる。
No.9　E_PARエラーを検出できるp_blkがシステムに存在しない場合は、テストを省略することができる。

MPF3：固定長メモリプール機能のテスト手順　その3（タスク単位で変数blkを設けること）

No	テ　ス　ト　手　順　内　容	テスト項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_TSK(TASK_ID2_1,{TA_HLNG,EXINF_2_1,TASK2_1,ITSKPRI_2,STKSZ,NULL})	
S3	CRE_TSK(TASK_ID2_2,{TA_HLNG,EXINF_2_2,TASK2_2,ITSKPRI_2,STKSZ,NULL})	
S4	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})	
S5	CRE_MPF(MPF_ID1,{TA_TPRI,BLKCNT_1,BLKSZ_10,NULL})	

—	**<TASK_ID1:DORMANT>**	**<TASK_ID2_1:DORMANT>**	**<TASK_ID2_2:DORMANT>**	**<TASK_ID3:READY>**	—
1				ercd = pget_mpf(MPF_ID1,&blk)	
2				*MERCD(ercd) == E_OK*	CRE_MBX-7
3				ercd = act_tsk(TASK_ID2_1)	
4		ercd = get_mpf(MPF_ID1,&blk)			
5				*MERCD(ercd) == E_OK*	
6				ercd = act_tsk(TASK_ID2_2)	
7			ercd = get_mpf(MPF_ID1,&blk)		
8				*MERCD(ercd) == E_OK*	
9				ercd = act_tsk(TASK_ID1)	
10	ercd = get_mpf(MPF_ID1,&blk)				
11				*MERCD(ercd) == E_OK*	
12				ercd = rel_mpf(MPF_ID1,blk)	

13	*MERCD(ercd) == E_OK*				get_mpf-7
14	ercd = rel_mpf(MPF_ID1,blk)				
15	*MERCD(ercd) == E_OK*				
16	ext_tsk( )				
17		*MERCD(ercd) == E_OK*			
18		ercd = rel_mpf(MPF_ID1,blk)			
19		*MERCD(ercd) == E_OK*			
20		ext_tsk( )			
21			*MERCD(ercd) == E_OK*		get_mpf-8
22			ercd = rel_mpf(MPF_ID1,blk)		
23			*MERCD(ercd) == E_OK*		
24			ext_tsk( )		
25				*MERCD(ercd) == E_OK*	
26				ext_tsk( )	

MPF4：固定長メモリプール機能のテスト手順　その4（タスク単位で変数blkを設けること）

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S4	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_4,STKSZ,NULL})				
S5	CRE_MPF(MPF_ID1,{TA_TFIFO,BLKCNT_1,BLKSZ_10,NULL})				
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2:DORMANT>**	**<TASK_ID3:DORMANT>**	**<TASK_ID4:READY>**	－
1				ercd = tget_mpf (ERR_MPFID,p_blk,TMO_POL)	
2				*MERCD(ercd) == E_ID*	tget_mpf-1
3				p_blkにE_PARエラーを検出する値を設定する	
4				ercd = tget_mpf (MPF_ID1,p_blk,TMO_POL)	
5				*MERCD(ercd) == E_PAR*	tget_mpf-2
6				ercd = tget_mpf (MPF_ID1,&blk,ERR_TMO)	
7				*MERCD(ercd) == E_PAR*	tget_mpf-3
8				ercd = tget_mpf (MPF_ID1,p_blk,TMO_POL)	
9				*MERCD(ercd) == E_OK*	tget_mpf-6,12
10				*blk==メモリブロックの先頭アドレス*	tget_mpf-6
11				ercd = act_tsk(TASK_ID3)	
12			ercd = tget_mpf (MPF_ID1,&blk,TMO_100)		
13				*MERCD(ercd) == E_OK*	
14				ercd = act_tsk(TASK_ID2)	
15		ercd = tget_mpf (MPF_ID1,&blk,TMO_200)			
16				*MERCD(ercd) == E_OK*	
17				ercd = act_tsk(TASK_ID1)	
18	ercd = tget_mpf (MPF_ID1,&blk,TMO_FEVR)				
19				*MERCD(ercd) == E_OK*	
20				ercd = rel_mpf(MPF_ID1,blk)	
21			*MERCD(ercd) == E_OK*		tget_mpf-7,11
22			*blk==メモリブロック先頭アドレス*		
23			ercd = dly_tsk(DLYTIM_500)		
24				*MERCD(ercd) == E_OK*	
25				ext_tsk( )	
26		*MERCD(ercd) == E_TMOUT*			tget_mpf-5 システム時刻管理-4
27		ext_tsk( )			
28			*MERCD(ercd) == E_OK*		
29			ercd = rel_wai(TASK_ID1)		
30	*MERCD(ercd) == E_RLWAI*				tget_mpf-4,8,13
31	ext_tsk( )				
32			*MERCD(ercd) == E_OK*		
33			ext_tsk( )		

(注)

No.5　E_PARエラーを検出できるp_blkがシステムに存在しない場合は、テストを省略することができる。

No.7　E_PARエラーを検出できるERR_TMOがシステムに存在しない場合は、テストを省略することができる。

MPF5：固定長メモリプール機能のテスト手順　その5（タスク単位で変数blkを設けること）

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2_1,{TA_HLNG,EXINF_2_1,TASK2_1,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID2_2,{TA_HLNG,EXINF_2_2,TASK2_2,ITSKPRI_2,STKSZ,NULL})				
S4	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S5	CRE_MPF(MPF_ID1,{TA_TPRI,BLKCNT_1,BLKSZ_10,NULL})				
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2_1:DORMANT>**	**<TASK_ID2_2:DORMANT>**	**<TASK_ID3:READY>**	－
1				ercd = tget_mpf (MPF_ID1,&blk,TMO_POL)	
2				*MERCD(ercd) == E_OK*	
3				ercd = act_tsk(TASK_ID2_1)	
4		ercd = tget_mpf (MPF_ID1,&blk,TMO_100)			
5				*MERCD(ercd) == E_OK*	
6				ercd = act_tsk(TASK_ID2_2)	
7			ercd = tget_mpf (MPF_ID1,&blk,TMO_FEVR)		
8				*MERCD(ercd) == E_OK*	
9				ercd = act_tsk(TASK_ID1)	
10	ercd = tget_mpf (MPF_ID1,&blk,TMO_200)				
11				*MERCD(ercd) == E_OK*	
12				ercd = rel_mpf(MPF_ID1,blk)	
13	*MERCD(ercd) == E_OK*				tget_mpf-9
14	*blk==メモリブロックの先頭アドレス*				
15	ercd = rel_mpf(MPF_ID1,blk)				
16	*MERCD(ercd) == E_OK*				
17	ext_tsk(　)				
18		*MERCD(ercd) == E_OK*			
19		*blk==メモリブロックの先頭アドレス*			
20		ercd = rel_mpf(MPF_ID1,blk)			
21		*MERCD(ercd) == E_OK*			
22		ext_tsk(　)			
23			*MERCD(ercd) == E_OK*		tget_mpf-10
24			*blk==メモリブロックの先頭アドレス*		
25			ercd = rel_mpf(MPF_ID1,blk)		
26			*MERCD(ercd) == E_OK*		
27			ext_tsk(　)		
28				*MERCD(ercd) == E_OK*	
29				ext_tsk(　)	

TIM：システム時刻管理機能のテスト手順**（本テストは、システム初期化直後に実行すること。）**

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
－	**<TASK_ID1:READY>**	－
1	ercd = get_tim(&systim2)	
2	*MERCD(ercd) == E_OK*	
3	*systim2 == 0*	システム時刻管理-1 システム初期化手順-3
4	systimにE_PARエラーを返す値を設定	
5	ercd = get_tim(&systim)	
6	*MERCD(ercd) == E_PAR*	get_tim-1
7	ercd = set_tim(&systim)	
8	*MERCD(ercd) == E_PAR*	set_tim-1
9	systimにE_PARエラーを返す値を設定	
10	ercd = set_tim(&systim)	
11	*MERCD(ercd) == E_PAR*	set_tim-2
12	systim1にSYSTIM_50を設定	
13	ercd = set_tim(&systim1)	
14	*MERCD(ercd) == E_OK*	
15	ercd = get_tim(&systim2)	
16	*MERCD(ercd) == E_OK*	
17	*systim2 ≧ SYSTIM_50*	set_tim-3 get_tim-2
18	ercd = dly_tsk(DLYTIM_16)	
19	*MERCD(ercd) == E_OK*	
20	ercd = get_tim(p_systim)	
21	*MERCD(ercd) == E_OK*	
22	*systim ≧ systim2 + DLYTIM_16*	システム時刻管理-2 isig_tim-1
23	ext_tsk(　)	

(注)

No.6　E_PARエラーを検出できるp_systimがシステムに存在しない場合は、テストを省略することができる。
No.8　E_PARエラーを検出できるp_systimがシステムに存在しない場合は、テストを省略することができる。
No.11　E_PARエラーを検出できるp_systimがシステムに存在しない場合は、テストを省略することができる。

CYC1：周期ハンドラ機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_CYC(CYC_ID1,{ERR_CYCATR,EXINF_CYC1,CYCHDR_1,CYCTIM_10,CYCPHS_5})	
S2	*E_RSATRエラーを検出できること*	CRE_CYC-1
S3	CRE_CYC(CYC_ID1,{TA_HLNG,EXINF_CYC1,ERR_CYCHDR,CYCTIM_10,CYCPHS_5})	
S4	*E_PARエラーを検出できること*	CRE_CYC-2
S5	CRE_CYC(CYC_ID1,{TA_HLNG|TA_STA,EXINF_CYC1,CYCHDR_1,CYCTIM_0,CYCPHS_5})	
S6	*E_PARエラーを検出できること*	CRE_CYC-3
S7	CRE_CYC(CYC_ID1,{TA_HLNG|TA_STA,EXINF_CYC1,CYCHDR_1,CYCTIM_0,ERR_CYCPHS})	
S8	*E_PARエラーを検出できること*	CRE_CYC-4

(注)

No.S4　E_PARエラーを検出できるERR_CYCHDRがシステムに存在しない場合は、テストを省略することができる。

CYC2：周期ハンドラ機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_CYC(CYC_ID1,{TA_HLNG|TA_STA,EXINF_CYC1,CYCHDR_1,CYCTIM_10,CYCPHS_5})			
S2	CRE_CYC(CYC_ID2,{TA_HLNG,EXINF_CYC2,CYCHDR_2,CYCTIM_55,CYCPHS_0})			
S3	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
－	**<TASK_ID1:READY>**	**<CYCHDR_1:周期ハンドラ>**	**<CYCHDR_2:周期ハンドラ>**	－
1	共通変数cyc_work1,cyc_work2を0クリア			
2	ercd = sta_cyc(ERR_CYCID)			
3	*MERCD(ercd) == E_ID*			sta_cyc-1
4	ercd = stp_cyc(ERR_CYCID)			
5	*MERCD(ercd) == E_ID*			stp_cyc-1
6	ercd = stp_cyc(CYC_ID2)			
7	*MERCD(ercd) == E_OK*			stp_cyc-3
8	ercd = dly_tsk(DLYTIM_100)			
9	*MERCD(ercd) == E_OK*			
10	*cyc_work2 == 0*			周期ハンドラ-2 CRE_CYC-6
11	*cyc_work1 ≧ 9*			CRE_CYC-5,7
12	ercd = stp_cyc(CYC_ID1)			
13	*MERCD(ercd) == E_OK*			
14	work = cyc_work1(カウンタのセーブ)			
15	ercd = dly_tsk(DLYTIM_100)			
16	*MERCD(ercd) == E_OK*			
17	*cyc_work1 == work*			stp_cyc-2
18	ercd = sta_cyc(CYC_ID2)			
19	*MERCD(ercd) == E_OK*			
20	ercd = dly_tsk(DLYTIM_100)			
21	*MERCD(ercd) == E_OK*			
22	*cyc_work2 == 1*			sta_cyc-2
23	ercd = sta_cyc(CYC_ID2)			
24	*MERCD(ercd) == E_OK*			
25	ercd = dly_tsk(DLYTIM_50)			
26	*MERCD(ercd) == E_OK*			
27	*cyc_work2 == 1*			sta_cyc-3
28	ercd = stp_cyc(CYC_ID2)			
29	*MERCD(ercd) == E_OK*			
30	ext_tsk(　)			
31		<CYCTIM_10周期に起動される＞		
32		*exinf == EXINF_CYC1*		システム時刻管理-6 周期ハンドラ-1
33		cyc_work1 ++		
34		return		
35			＜CYCTIM_55周期に起動される＞	
36			*exinf == EXINF_CYC2*	
37			cyc_work2 ++	
38			return	

SYS1：システム状態管理機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_1,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_1,STKSZ,NULL})				
S4	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_1,STKSZ,NULL})				
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<TASK_ID3:READY>**	**<TASK_ID4:READY>**	－
1	ercd = rot_rdq(ERR_TSKPRI)				
2	*MERCD(ercd) == E_PAR*				rot_rdq-1
3	ercd = get_tid(p_tskid)				
4	*MERCD(ercd) == E_OK*				
5	*tskid == TASK_ID1*				get_tid-1
6	state = sns_dpn(　)				
7	*state == FALSE*				sns_dpn-4
8	ercd = loc_cpu(　)				

9	*MERCD(ercd) == E_OK*				
10	state = sns_loc( )				
11	*state == TRUE*				loc_cpu-1 sns_loc-1
12	state = sns_dpn( )				
13	*state == TRUE*				sns_dpn-2
14	ercd = loc_cpu( )				
15	*MERCD(ercd) == E_OK*				
16	state = sns_loc( )				
17	*state == TRUE*				loc_cpu-2
18	ercd = unl_cpu( )				
19	*MERCD(ercd) == E_OK*				
20	state = sns_loc( )				
21	*state == FALSE*				unl_cpu-1 sns_loc-2
22	ercd = unl_cpu( )				
23	*MERCD(ercd) == E_OK*				
24	state = sns_loc( )				
25	*state == FALSE*				unl_cpu-2
26	state = sns_ctx( )				
27	*state == FALSE*				sns_ctx-1
28	ercd = dis_dsp( )				
29	*MERCD(ercd) == E_OK*				
30	state = sns_dsp( )				
31	*state == TRUE*				dis_dsp-1 sns_dsp-1
32	state = sns_dpn( )				
33	*state == TRUE*				sns_dpn-3
34	ercd = dis_dsp( )				
35	*MERCD(ercd) == E_OK*				
36	state = sns_dsp( )				
37	*state == TRUE*				dis_dsp-2
38	ercd = ena_dsp( )				
39	*MERCD(ercd) == E_OK*				
40	state = sns_dsp( )				
41	*state == FALSE*				ena_dsp-1 sns_dsp-2
42	ercd = ena_dsp( )				
43	*MERCD(ercd) == E_OK*				
44	state = sns_dsp( )				
45	*state == FALSE*				ena_dsp-2
46	ercd = rot_rdq(TPRI_SELF)				
47		ercd = rot_rdq(TSKPRI_1)			
48			ercd = rot_rdq(TPRI_SELF)		
49				ercd = rot_rdq(TSKPRI_1)	
50	*MERCD(ercd) == E_OK*				rot_rdq-3
51	ercd = chg_pri(TASK_ID2,TSKPRI_2)				
52	*MERCD(ercd) == E_OK*				
53	ercd = chg_pri(TASK_ID3,TSKPRI_2)				
54	*MERCD(ercd) == E_OK*				
55	ercd = chg_pri(TASK_ID4,TSKPRI_2)				
56	*MERCD(ercd) == E_OK*				
57	ercd = rot_rdq(TSKPRI_2)				
58	*MERCD(ercd) == E_OK*				
59	ercd = rot_rdq(TSKPRI_2)				
60	*MERCD(ercd) == E_OK*				
61	ercd = dly_tsk(DLYTIM_20)				
62				*MERCD(ercd) == E_OK*	
63				ext_tsk( )	
64		*MERCD(ercd) == E_OK*			
65		ext_tsk( )			
66			*MERCD(ercd) == E_OK*		rot_rdq-2
67			ext_tsk( )		
68	*MERCD(ercd) == E_OK*				
69	ext_tsk( )				

SYS2：システム状態管理機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_1,STKSZ,NULL})	
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_1,STKSZ,NULL})	
S4	DEF_INH(INHNO,{INHATR,INTHDR_1})	

—	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<TASK_ID3:READY>**	**<INTHDR_1:割込みハンドラ>**		—
				<1回目>	<2回目>	
1	INTHDR_1の割込み発生					
2				ercd = irot_rdq (ERR_TSKPRI)		
3				MERCD(ercd) == E_PAR		irot_rdq-1

4				ercd = irot_rdq (TPRI_SELF)		
5				MERCD(ercd) == E_PAR		irot_rdq-3
6				ercd = iget_tid (p_tskid)		
7				*tskid== TASK_ID1*		非タスクコンテキストから呼び出せるサービスコール-11 iget_tid-1
8				ercd = iloc_cpu( )		
9				*MERCD(ercd) == E_OK*		非タスクコンテキストから呼び出せるサービスコール-12
10				state = sns_loc( )		
11				*state == TRUE*		iloc_cpu-1
12				ercd = iloc_cpu( )		
13				MERCD(ercd) == E_OK		
14				state = sns_loc( )		
15				*state == TRUE*		iloc_cpu-2
16				ercd = iunl_cpu( )		
17				*MERCD(ercd) == E_OK*		非タスクコンテキストから呼び出せるサービスコール-13
18				state = sns_loc( )		
19				*state == FALSE*		iunl_cpu-1
20				ercd = iunl_cpu( )		
21				MERCD(ercd) == E_OK		
22				state = sns_loc( )		
23				*state == FALSE*		iunl_cpu-2
24				state = sns_ctx( )		
25				*state == TRUE*		sns_ctx-2
26				ercd = irot_rdq (TSKPRI_1)		
27				*MERCD(ercd) == E_OK*		非タスクコンテキストから呼び出せるサービスコール-10
28				return		
29		ext_tsk( )				
30			ext_tsk( )			
31	10msec後に割込み発生					irot_rdq-2
32	ext_tsk( )					
33					ercd = iget_tid (p_tskid)	
34					tskid == TSK_NONE	iget_tid-2
35					state = sns_dpn( )	
36					*state = TRUE*	sns_dpn-1
37					return	

INH1：割込み管理機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容		テスト項目参照
S1	DEF_INH(INHNO_1,{ERR_INHATR,INTHDR_1})		
S2	*E_RSATRエラーを検出できること。*		DEF_INH-1
S3	DEF_INH(ERR_INHNO,{INHATR,INTHDR_1})		
S4	*E_PARエラーを検出できること。*		DEF_INH-2
S5	DEF_INH(INHNO_1,{INHATR,ERR_INTHDR})		
S6	*E_PARエラーを検出できること。*		DEF_INH-3
S7	DEF_INH(INHNO_1,{INHATR,INTHDR_1})		静的APIの文法とパラメータ-3
S8	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
－	**<TASK_ID1:READY>**	**<INTHDR_1:割込みハンドラ>**	－
1	INHNO_1の割込み発生		
2		return	DEF_INH-4
3	ext_tsk( )		

(注)
No.S6　E_PARエラーを検出できるERR_INTHDRがシステムに存在しない場合は、テストを省略することができる。

ISR1：割込みサービスルーチンのテスト手順　その1

No	テ　ス　ト　手　順　内　容		テスト項目参照
S1	ATT_ISR(ERR_ISRATR,{EXINF_1,INTNO_1,ISR_1})		
S2	*E_RSATRエラーを検出できること。*		ATT_ISR-1
S3	ATT_ISR(TA_HLNG,{EXINF_1,ERR_INTNO,ISR_1})		
S4	*E_PARエラーを検出できること。*		ATT_ISR-2
S5	ATT_ISR({TA_HLNG,EXINF_1,INTNO_1,ISR_ADR_0})		
S6	*E_PARエラーを検出できること。*		ATT_ISR-3
S7	ATT_ISR({TA_HLNG,EXINF_1,INTNO_1,ISR_1})		
S8	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
－	**<TASK_ID1:READY>**	**<ISR_1:割込みサービスルーチン>**	
1	INTNO_1の割込み番号に対応する割込み発生		
2		*exinf == EXINF_1*	割込み管理機能-4 ATT_ISR-4

3		return	
4	ext_tsk( )		

(注)
No.S6　E_PARエラーを検出できるISR_ADR_0がシステムに存在しない場合は、テストを省略することができる。

EXC1：CPU例外ハンドラのテスト手順　その1

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	DEF_EXC(EXCNO_1,{ERR_EXCATR,EXCHDR_1})		
S2	*E_RSATRエラーを検出できること。*		DEF_EXC-1
S3	DEF_EXC(ERR_EXCNO,{EXCATR,EXCHDR_1})		
S4	*E_PARエラーを検出できること。*		DEF_EXC-2
S5	DEF_EXC(EXCNO_1,{EXCATR,ERR_EXCHDR})		
S6	*E_PARエラーを検出できること。*		DEF_EXC-3
S7	DEF_EXC(EXCNO_1,{EXCATR,EXCHDR_1})		
S8	DEF_EXC(EXCNO_1,{EXCATR,EXCHDR_2})		
S9	CRE_TSK(TASK_ID1,TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL)		
－	**<TASK_ID1:READY>**	**<EXCHDR_2:CPU例外ハンドラ>**	－
1	EXCHDR_2の割込み発生		
2		return	DEF_EXC-5
3	ext_tsk( )		

(注)
No.S6　E_PARエラーを検出できるERR_EXCHDRがシステムに存在しない場合は、テストを省略することができる。

INI1：初期化ルーチンのテスト手順　その1

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	ATT_INI({TA_HLNG,EXINF_1,INIRTN_1})		
S2	CRE_TSK({TASK_ID1,TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
－	**<TASK_ID1:READY>**	**<INIRTN_1:初期化ルーチン>**	－
1		TASK_ID1より先に初期化ルーチンが実行される	ATT_INI-1
2		*exinf == EXINF_1*	ｼｽﾃﾑ構成管理機能-3
3		return	
4	ext_tsk( )		
5			

第3節　製品マニュアルによる確認の見方

以下に、製品マニュアル確認の見方を説明する。

UMA

No.	区　　分	テ　ス　ト　項　　目	テスト項目参照
1	APIの構成要素	すべてのサービスコールの名称、パラメータとリターンパラメータの種類・順序・名称・データ型がμITRON4.0仕様書通りに記載されていること。ただし、実装が独自に拡張しているサービスコールは除く。	APIの構成要素-1 サービスコールの返値とエラーコード-1
2		静的APIの名称、パラメータとリターンパラメータの種類・順序・名称・データ型がμITRON4.0仕様書通りに記載されていること。ただし、実装が独自に拡張している静的APIは除く。	APIの構成要素-2
3	サービスコールの返値とエラーコード	サブエラーコードをサポートする場合は、その意味が記載されていること。	サービスコールの返値とエラーコード-3
4	静的APIの文法とパラメータ	静的APIの処理中にエラーが発生した場合の扱いが記載されていること。	静的APIの文法とパラメータ-8
5	APIの名称に関する原則	実装が独自に用意するサービスコールの名称が記載されていること。	APIの名称に関する原則-1
6		実装が独自に定義するメインエラーコードの名称とその内容が記載されていること。	APIの名称に関する原則-2

(1)　No.
製品マニュアルの確認番号を示す。

(2)　区分
製品マニュアルの確認で、確認項目を内容毎に分類したものである。

(3)　テスト項目
μITRON4.0仕様書に記載されている内容であり、テスト手順(プログラム)では確認できない項目である。したがって、検定を受けるカーネルメーカが作成する製品マニュアルに、振舞いや意味が明確に記載されていることを確認する。

(4) テスト項目参照
製品マニュアルで確認すべき事項が、テスト項目のどの部分に記載されているかを示す。
なお、No.3で示すように、カーネルが当該機能をサポートしていない場合は、検定項目から除くことを明記している。

第4節　製品マニュアルによる確認の手順

UMA：

No.	区　　分	テ　　ス　　ト　　項　　目	テスト項目参照
1	APIの構成要素	スタンダードプロファイルに含まれるサービスコールの名称、パラメータとリターンパラメータの種類・順序・名称・データ型がμITRON4.0仕様書通りに記載されていること。	APIの構成要素-1 サービスコールの返値とエラーコード-1
2		スタンダードプロファイルに含まれる静的APIの名称、パラメータの種類・順序・名称・データ型がμITRON4.0仕様書通りに記載されていること。	APIの構成要素-2
3	サービスコールの返値とエラーコード	サブエラーコードをサポートする場合は、その意味が記載されていること。	サービスコールの返値とエラーコード-2
4	静的APIの文法とパラメータ	静的APIの処理中にエラーが発生した場合の扱いが記載されていること。	静的APIの文法とパラメータ-9
5	サービスコール	実装が独自に用意するサービスコールと静的APIの名称がμITRON4.0仕様書通りに記載されていること。	サービスコール-1 静的API-1
6	定　　数	実装が独自に定義するメインエラーコードの名称とその内容がμITRON4.0仕様書通りに記載されていること。	定数-1
7	割込み処理モデル	割込みハンドラの記述方法（カーネルが用意する出入口処理やアプリケーションが登録する割込みハンドラで行うべき処理）が記載されていること。	割込み処理ハンドラと割込みサービスルーチン-1
8		DEF_INTとATT_ISRの両方をサポートした場合の振舞いが記載されていること。	割込み処理ハンドラと割込みサービスルーチン-2
9		どの優先度より高い優先度を持つものをカーネルの管理外の割込みにするかが、製品マニュアルに記載されていること。	割込み処理ハンドラと割込みサービスルーチン-3
10	CPU例外ハンドラで行える操作	CPU例外ハンドラの記述方法が記載されていること。	CPU例外ハンドラで行える操作-1
11		CPU例外ハンドラ内で呼出し可能なサービスコールが記載されていること。	CPU例外ハンドラで行える操作-2
12		CPU例外が発生したコンテキストや状態の読出し方法が記載されていること。	CPU例外ハンドラで行える操作-3
13		CPU例外が発生したタスクのID番号の読出し方法が記載されていること。	CPU例外ハンドラで行える操作-4
14		CPU例外ハンドラ内でras_texと同等の操作ができ、方法が記載されていること。	CPU例外ハンドラで行える操作-5
15	タスクコンテキストと非タスクコンテキスト	タスクコンテキストを実行中にCPU例外が発生した場合、CPU例外ハンドラが実行されるコンテキストが製品マニュアルに記載されていること。	タスクコンテキストと非タスクコンテキスト-4
16	処理の優先順位とサービスコールの不可分性	割込みハンドラおよび割込みサービスルーチン相互間の優先順位の関係が記載されていること。	処理の優先順位とサービスコールの不可分性-2
17		周期ハンドラの優先順位と、isig_timを呼び出した割込みハンドラの優先順位との関係が記載されていること。	処理の優先順位とサービスコールの不可分性-3
18		CPU例外ハンドラの優先順位が、CPU例外が発生した処理の優先順位と、ディスパッチャの優先順位のいずれよりも高いことが製品マニュアルに記載されていること。	処理の優先順位とサービスコールの不可分性-4

19		CPU例外ハンドラと、割込みハンドラやタイムイベントハンドラとの間の優先順位が記載されていること。	処理の優先順位とサービスコールの不可分性-5
20	CPUロック状態	割込みハンドラ内でCPUロック解除状態にするための方法が記載されていること。	CPUロック状態-11
21		割込みハンドラから正しくリターンするための方法が記載されていること。	CPUロック状態-12
22	システム初期化手順	カーネルの初期化処理を呼び出す方法が記載されていること。	システム初期化手順-4
23		静的APIの処理中にエラーを検出した場合の扱いが記載されていること。	システム初期化手順-5
24		カーネルの管理外の割込みを禁止するかどうかが記載されていること。	システム初期化手順-6
25		初期化ルーチン内でサービスコールを呼び出せるか否か、呼び出せる場合にはどのようなサービスコールが呼び出せるか記載されていること。	システム初期化手順-7
26	オブジェクトの登録とその削除	カーネルに登録できるオブジェクトの最大数やID番号の範囲の指定方法が記載されていること。	オブジェクトの登録とその削除-2
27		自動割付けされるID番号の範囲の指定方法が記載されていること。	オブジェクトの登録とその削除-3
28	ext_tsk	エラーが発生した場合の処理が記載されていること。	ext_tsk-6
29	システム時刻管理	システム時刻更新機能がカーネル内部にある場合は、システム時刻更新方法が記載されていること。	システム時刻管理-3 isig_tim-2
30		記載通りに実行するとシステム時刻が正しく更新されること。	システム時刻管理-3
31	割込み管理機能	割込みハンドラの記述方法が記載されていて、正しく動作すること。	割込み管理機能-3
32		割込みハンドラ内でCPUロック解除状態にする方法が記載されていること。	割込み管理機能-5
33		CPUロックを解除した後に、割込みハンドラからリターンする方法が記載されていること。	割込み管理機能-6
34	DEF_INH	inhnoの具体的な意味が記載されていること。	DEF_INH-5
35	システム構成管理機能	CPU例外ハンドラの記述方法が記載されていて、正しく動作すること。	システム構成管理機能-2
36	DEF_EXC	excnoの具体的な意味が記載されていること。	DEF_EXC-4

第5節　ヘッダファイルによる確認の見方

以下に、μITRON4.0仕様で定義するヘッダファイルの内容確認の見方について説明する。

HED：

No.	区　分	テ　ス　ト　項　目		テスト項目参照
1	APIの構成要素	“kernel.h”から“itron.h”をインクルードしていること。		APIの構成要素-3
2	APIの名称に関する原則	ITRON仕様共通定義で規定されるデータ型、定数、マクロの定義などを含むヘッダファイルの名称が、“itron.h”であること。		ヘッダファイル-1
3		カーネル仕様で定められるサービスコールの宣言と、データ型、定数、マクロの定義などを含むヘッダファイルの名称が、“kernel.h”であること。		サービスコールの返値とエラーコード-1 ヘッダファイル-2
4		カーネルのコンフィギュレータが生成する自動割付け結果ヘッダファイルの名称を“ kernel_id.h ”とすること。		ヘッダファイル-3
－	ITRON仕様共通データ型（itron.h）	型	整数／ポインタ／構造体	－
5		B	整数	ITRON仕様共通データ型-1
6		H	整数	ITRON仕様共通データ型-2
7		W	整数	ITRON仕様共通データ型-3
～				～
34	ITRON仕様共通定数（itron.h）	NULL	0で定義されていること	ITRON仕様共通定数-1
35		TRUE	1で定義されていること	ITRON仕様共通定数-2
36		FALSE	0で定義されていること	ITRON仕様共通定数-3

(1)　No.
ヘッダファイルの確認番号を示す。

(2)　区分
ヘッダファイルの確認で、確認項目をμITRON4.0仕様書の内容に添って分類したものである。

(3)　テスト項目
μITRON4.0仕様書に記載されている内容であり、テスト手順(プログラム)では確認できない項目である。したがって、検定を受けるカーネルメーカは、μITRON4.0仕様書が規定するヘッダファイルの名称、および当該ヘッダファイル内で記述すべき内容が明確に記載されていることを確認する。
データ型や定数についても、μITRON4.0仕様書が規定している通りに定義されていることを確認する。なお、検定対象のカーネルがTA_NULLをサポートしていなければ、省略することが可能である。
網掛けされてあるエラーコードは、実装依存や実装定義でエラーの検出を省略できることを示す。ただし、これらのエラーコードの検出を省略する場合は、省略する旨が製品マニュアルに記載されていること。

(4)　テスト項目参照
ヘッダファイルで確認すべき事項が、テスト項目のどの部分に記載されているかを示す。

第6節　ヘッダファイルによる確認の手順

HED：

No.	区　　分	テ　　ス　　ト　　項　　目		テスト項目参照
1	APIの構成要素	“kernel.h”から“itron.h”をインクルードしていること。		APIの構成要素-3
2	APIの名称 に関する原則	ITRON仕様共通定義で規定されるデータ型、定数、マクロの定義などを含むヘッダファイルの名称が、“itron.h”であること。		ヘッダファイル-1
3		カーネル仕様で定められるサービスコールの宣言と、データ型、定数、マクロの定義などを含むヘッダファイルの名称が、“kernel.h”であること。		サービスコールの返値とエラーコード-1 ヘッダファイル-2
4		カーネルのコンフィギュレータが生成する自動割付け結果ヘッダファイルの名称を “ kernel_id.h ” とすること。		ヘッダファイル-3
—	ITRON仕様 共通データ型 （itron.h）	型	整数／ポインタ／構造体	—
5		B	整数	ITRON仕様共通データ型-1
6		H	整数	ITRON仕様共通データ型-2
7		W	整数	ITRON仕様共通データ型-3
8		UB	整数	ITRON仕様共通データ型-4
9		UH	整数	ITRON仕様共通データ型-5
10		UW	整数	ITRON仕様共通データ型-6
11		VB	定義されていること	ITRON仕様共通データ型-7
12		VH	定義されていること	ITRON仕様共通データ型-8
13		VW	定義されていること	ITRON仕様共通データ型-9
14		VP	ポインタ	ITRON仕様共通データ型-10
15		FP	ポインタ	ITRON仕様共通データ型-11
16		INT	整数	ITRON仕様共通データ型-12
17		UINT	整数	ITRON仕様共通データ型-13
18		BOOL	整数	ITRON仕様共通データ型-14
19		FN	整数	ITRON仕様共通データ型-15
20		ER	整数	ITRON仕様共通データ型-16
21		ID	整数	ITRON仕様共通データ型-17
22		ATR	整数	ITRON仕様共通データ型-18
23		STAT	整数	ITRON仕様共通データ型-19
24		MODE	整数	ITRON仕様共通データ型-20
25		PRI	整数	ITRON仕様共通データ型-21
26		SIZE	整数	ITRON仕様共通データ型-22
27		TMO	整数	ITRON仕様共通データ型-23
28		RELTIM	整数	ITRON仕様共通データ型-24
29		SYSTIM	整数	ITRON仕様共通データ型-25
30		VP_INT	整数またはポインタ	ITRON仕様共通データ型-26
31		ER_BOOL	整数	ITRON仕様共通データ型-27
32		ER_ID	整数	ITRON仕様共通データ型-28
33		ER_UINT	整数	ITRON仕様共通データ型-29
34	ITRON仕様共通 定数 （itron.h）	NULL	0で定義されていること	ITRON仕様共通定数-1
35		TRUE	1で定義されていること	ITRON仕様共通定数-2
36		FALSE	0で定義されていること	ITRON仕様共通定数-3
37		E_OK	0で定義されていること	ITRON仕様共通定数-4
38		E_SYS	-5で定義されていること	ITRON仕様共通定数-5
39		E_NOSPT	-9で定義されていること	ITRON仕様共通定数-6
40		E_RSFN	-10で定義されていること	ITRON仕様共通定数-7
41		E_RSATR	-11で定義されていること	ITRON仕様共通定数-8

42		E_PAR	-17で定義されていること	ITRON仕様共通定数-9
43		E_ID	-18で定義されていること	ITRON仕様共通定数-10
44		E_CTX	-25で定義されていること	ITRON仕様共通定数-11
45		E_MACV	-26で定義されていること	ITRON仕様共通定数-12
46		E_ILUSE	-28で定義されていること	ITRON仕様共通定数-13
47		E_NOMEM	-33で定義されていること	ITRON仕様共通定数-14
48		E_OBJ	-41で定義されていること	ITRON仕様共通定数-15
49		E_QOVR	-43で定義されていること	ITRON仕様共通定数-16
50		E_RLWAI	-49で定義されていること	ITRON仕様共通定数-17
51		E_TMOUT	-50で定義されていること	ITRON仕様共通定数-18
52		TA_NULL	0で定義されていること	ITRON仕様共通定数-19
53		TMO_POL	0で定義されていること	ITRON仕様共通定数-20
54		TMO_FEVR	-1で定義されていること	ITRON仕様共通定数-21
55	ITRON仕様 共通マクロ	“itron.h”にMERCDマクロが定義されていること。		ITRON仕様共通マクロ-1
56		“itron.h”にSERCDマクロが定義されていること。		ITRON仕様共通マクロ-2
57		TA_HLNG	0x00 で定義されていること	カーネル共通定義-1
58		TA_TFIFO	0x00 で定義されていること	カーネル共通定義-2
59		TA_TPRI	0x01 で定義されていること	カーネル共通定義-3
60	カーネル 共通定義	TA_MFIFO	0x00 で定義されていること	カーネル共通定義-4
61		TA_MPRI	0x02 で定義されていること	カーネル共通定義-5
62		TSK_SELF	0 で定義されていること	カーネル共通定義-6
63		TSK_NONE	0 で定義されていること	カーネル共通定義-7
64		TMIN_TPRI	1 で定義されていること	カーネル共通構成定数-1
65		TMAX_TPRI	16以上 で定義されていること	カーネル共通構成定数-2
66		TMIN_MPRI	0x01 で定義されていること	カーネル共通構成定数-3
67	カーネル共通 構成定数	TMAX_MPRI	TMAX_TPRI以上 で定義されていること	カーネル共通構成定数-4
68		TKERNEL_MAKER	カーネルのメーカーコード	カーネル共通構成定数-5
69		TKERNEL_PRID	カーネルの識別番号	カーネル共通構成定数-6
70		TKERNEL_SPVER	ITRON仕様のバージョン番号	カーネル共通構成定数-7
71		TKERNEL_PRVER	カーネルのバージョン番号	カーネル共通構成定数-8
72	タスク管理機能	TMAX_ACTCNT	1以上で定義されていること	タスク管理機能-9
73	タスク付属 同期機能	TMAX_WUPCNT	1以上で定義されていること	タスク付属同期機能-1
74		TMAX_SUSCNT	1以上で定義されていること	タスク付属同期機能-2
75	タスク例外 処理機能	TBIT_TEXPTN	16ビット以上で定義されていること	タスク例外処理機能-11
76		TEXPTN	16ビット以上で定義されていること	タスク例外処理機能-12
77	セマフォ	TMAX_MAXSEM	65535以上で定義されていること	セマフォ-1
78	イベントフラグ	FLGPRN	16ビット以上で定義されていること	イベントフラグ-1
79		TBIT_FLGPTN	16ビット以上で定義されていること	イベントフラグ-2
80		T_MSG	実装毎に定義されていること	メールボックス-1
81	メールボックス	T_MSG_PRI	メールボックスの優先度付きメッセージヘッダ typedef struct t_msg_pri{ T_MSG msgque ; PRI msgpri ; } T_MSG_PRI ;	メールボックス-2
82	システム 時刻管理	TIC_NUME	実装毎に定義されていること	システム時刻管理-7
83		TIC_DENO	実装毎に定義されていること	システム時刻管理-8
84	割込み管理機能	INHNO	実装毎に定義されていること	割込み管理機能-1
85		INTNO	実装毎に定義されていること	割込み管理機能-2
86	システム構成管理機能	EXCNO	実装毎に定義されていること	システム構成管理機能-1

第4章　自動車制御用プロファイルのテスト項目

第１節　テスト項目の見方

テスト項目は、μITRON4.0 仕様書に記載されている内容と対応しており、確認すべき項目を洗い出したものである。また、「4.1　タスク管理機能」などタイトルの前の番号は、μITRON4.0 仕様書に記載されてあるものと一致させた。

以下に、act_tsk サービスコールを例にテスト項目の見方を説明する。

No.	区　分	テ　ス　ト　項　目		テスト手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK2-2
2		E_QOVR	キューイングオーバーフロー(起動要求キューイング数のオーバーフロー)	TSK2-20
3	サービスコール処　理	対象タスクが休止状態である場合には、tskid で指定されるタスクを、休止状態から実行可能状態に移行させること。		TSK2-12
4		タスクの起動時に行うべき処理を行うこと。		(Ref)タスク管理機能-2,3
5		タスクを起動する際のパラメータは、生成時に指定したタスクの拡張情報(exinf)と一致すること。また、起動されたタスクでは、拡張情報が正しく受け取れること。		TSK2-12
6		対象タスクが休止状態でない場合には、tskid で指定されるタスクに対する起動要求をキューイング(タスクの起動要求キューイング数に 1 を加える)すること。		TSK2-24
7		tskid に TSK_SELF が指定されると、自タスクを対象とすること。		TSK2-4,6

(1)　No.
当該サービスコールのテスト番号を示す。

(2)　区分
テスト項目をテスト内容毎に分類したものである。

(3)　テスト項目
μITRON4.0 仕様書に記載されている内容であり、機能を確認する項目である。エラーコードに網掛けされてあるものは、実装定義でエラーの検出を省略できることを示す。ただし、これらのエラーコードの検出を省略する場合は、省略する旨が製品マニュアルに記載されていること。

(4)　テスト手順参照
テスト項目を確認する具体的手段が、テスト手順のどの部分に記載されているかを示す。
(Ref)と記述されているものは、別のテスト項目で確認済であるので、ここでの確認項目ではないことを示す。
また、複数の番号が記述されている場合は、当該テスト項目の確認に複数のテスト手順が必要であることを示している。

第 2 節　テスト項目

2.　ITRON 仕様共通定義

2.1　ITRON 仕様共通概念

2.1.1　用語の意味

確認項目なし

2.1.2　API の構成要素

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	自動車制御用プロファイルに含まれるサービスコールの名称、パラメータとリターンパラメータの種類・順序・名称・データ型が、μITRON4.0 仕様書と一致していること。	UMA-1
2	自動車制御用プロファイルに含まれる静的 API の名称、パラメータの種類・順序・名称・データ型がμITRON4.0 仕様書と一致していること。	UMA-2
3	“kernel.h”から“itron.h”をインクルードしていること。	HED-1
4	オブジェクトの ID 番号などの自動割付けを行うコンフィギュレータは、自動割付けを行った結果を、自動割付け結果ヘッダファイル“ kernel_id.h ”に生成すること。	COM4-7
5	ITRON 仕様で標準化された同じヘッダファイルを複数回インクルードしてもエラーとならないこと。	COM1-1

2.1.3　オブジェクトの ID 番号とオブジェクト番号

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	ID 番号は 1～255 の範囲の正の値をサポートしていること。	SPC15-S256

2.1.4　優先度

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	1～16 の 16 段階のタスク優先度をサポートすること。	COM2-16

2.1.5　機能コード

確認項目なし

2.1.6　サービスコールの返値とエラーコード

項番	確　　認　　事　　項			ﾃｽﾄ手順参照
1	エラーコードは、下位 8 ビットのメインエラーコードと、それを除いた上位ビットのサブエラーコードで構成される。			COM8-2,3
2	サブエラーコードをサポートする場合は、実装毎に定義しサブエラーコードに関する記述が製品マニュアルに記載されていること。			UMA-3
3	ﾒｲﾝｴﾗｰｺｰﾄﾞ	E_MACV	メモリアクセス違反	COM3-2

2.1.7　オブジェクト属性と拡張情報

確認項目なし

2.1.8　タイムアウトとノンブロッキング

確認項目なし

2.1.9　相対時間とシステム時刻

確認項目なし

2.1.10 システムコンフィギュレーションファイル

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	システムコンフィギュレーションファイルには、カーネルやソフトウェア部品の静的 API と ITRON 仕様共通静的 API に加えて、C 言語処理系のプリプロセッサディレクティブを記述できること。	COM4-3,5
2	システムコンフィギュレーションファイルは、まず、C言語のプリプロセッサに通される。次にソフトウェア部品のコンフィギュレータによって順に処理され、最後にカーネルのコンフィギュレータによって処理されること。	COM4-3,4,5
3	カーネルコンフィギュレータでは、カーネル構成・初期化ファイルと ID 自動割付け結果ヘッダファイル(kernel_id.h)を生成すること。	COM4-6,7
4	自分自身に対する静的 API または共通静的 API として解釈できない記述が含まれている場合には、エラーを報告すること。	COM5-1,2,3,4
5	カーネルのコンフィギュレータは、“＃”で始まる行を無視すること。	COM4-5

2.1.11 静的 API の文法とパラメータ

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	サービスコールに対応する静的 API のパラメータは、対応するサービスコールのC言語 API に準ずること。ただし、パラメータにパケットへのポインタがある場合には、パケット中の各要素を“，”で区切り、“｛”と“｝”で囲んだ形で記述すること。	COM4-2
―	静的 API のパラメータは、記述に使える式によって分類される。	―
2	自動割付け対応整数値パラメータ	COM4-2
3	自動割付け非対応整数値パラメータ	INH1-S7
4	プリプロセッサ定数式パラメータ	COM4-2
5	一般定数式パラメータ	COM4-2
6	自動割付け対応整数値パラメータに単一の識別子が記述された場合、その静的 API を処理するコンフィギュレータが、識別子に整数値を割付けること。	COM4-7
7	静的 API に文法エラーやパラメータ数の過不足があった場合には、それを検出したコンフィギュレータがエラーを報告すること。	COM5-1,2,3,4
8	静的 API の処理中にエラーを検出した場合の扱いについては、実装毎に定義すること。エラー発生時の扱いは製品マニュアルに記載されていること。	UMA-4

2.2. API の名称に関する原則

2.2.1 ソフトウェア部品識別名

確認項目なし

2.2.2 サービスコール

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	実装が独自に用意するサービスコールの名称が、μITRON4.0 仕様書と合致していること。また、その名称が製品マニュアルに記載されていること。	UMA-5

2.2.3 コールバック

確認項目なし

2.2.4 静的 API

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	実装が独自に用意する静的 API の名称が、μITRON4.0 仕様書と合致していること。また、その名称が製品マニュアルに記載されていること。	UMA-5

2.2.5 パラメータとリターンパラメータ

確認項目なし

2.2.6 データ型

確認項目なし

2.2.7 定数

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	実装が独自に定義するメインエラーコードの名称が、μITRON4.0 仕様書と一致していること。また、その名称が製品マニュアルに記載されていること。	UMA-6

2.2.8 マクロ

確認項目なし

2.2.9 ヘッダファイル

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	ITRON 仕様共通定義で規定されるデータ型、定数、マクロの定義などを含むヘッダファイルの名称を“ itron.h ”とすること。	HED-2
2	カーネル仕様で定められるすべてのサービスコールの宣言と、データ型、定数、マクロの定義などを含むヘッダファイルの名称を“ kernel.h ”とすること。	HED-3
3	カーネルのコンフィギュレータが生成する自動割付け結果ヘッダファイルの名称を“ kernel_id.h ”とすること。	HED-4

2.2.10 カーネルとソフトウェア部品の内部識別子

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	カーネルの内部識別子は、C 言語レベルで、_kernel_または_KERNEL_で始まる名称とすることを原則とする。	COM6-1

2.3 ITRON 仕様共通定義

2.3.1 ITRON 仕様共通データ型

項番	確　　認　　事　　項		ﾃｽﾄ手順参照
1	B	符号付き 8 ビット整数	HED-5 COM7-1,3
2	H	符号付き 16 ビット整数	HED-6 COM7-7,9
3	W	符号付き 32 ビット整数	HED-7 COM7-13,15
4	UB	符号無し 8 ビット整数	HED-8 COM7-4,6
5	UH	符号無し 16 ビット整数	HED-9 COM7-10,12
6	UW	符号無し 32 ビット整数	HED-10 COM7-16,18
7	VB	データタイプが定ならない 8 ビットの値　(実装毎に定義)	HED-11 COM7-19
8	VH	データタイプが定ならない 16 ビットの値　(実装毎に定義)	HED-12 COM7-20
9	VW	データタイプが定ならない 32 ビットの値　(実装毎に定義)	HED-13 COM7-21
10	VP	データタイプの定まらないものへのポインタ	HED-14 COM7-22
11	FP	プログラムの起動番地(ポインタ)	HED-15 COM7-23
12	INT	16 ビット以上の符号付き整数	HED-16 COM7-24,26
13	UINT	16 ビット以上の符号無し整数	HED-17 COM7-27,29

14	BOOL	真偽値(TRUE または FALSE)	HED-18 COM7-31,33
15	FN	機能コード(16 ビット以上の符号付き整数)	HED-19 COM7-34,36
16	ER	エラーコード(8 ビット以上の符号付き整数)	HED-20 COM7-37,39
17	ID	オブジェクトの ID 番号(16 ビット以上の符号付き整数)	HED-21 COM7-40,42
18	ATR	オブジェクト属性(8 ビット以上の符号無し整数)	HED-22 COM7-43,45
19	STAT	オブジェクトの状態(16 ビット以上の符号無し整数)	HED-23 COM7-46,48
20	MODE	サービスコールの動作モード(8 ビット以上の符号無し整数)	HED-24 COM7-49,51
21	PRI	優先度(16 ビット以上の符号付き整数)	HED-25 COM7-52,54
22	SIZE	メモリ領域のサイズ(ポインタと同じビット幅の符号無し整数)	HED-26 COM7-55,57
23	TMO	タイムアウト指定(16 ビット以上の符号付き整数、時間単位は 1 ミリ秒)	HED-27 COM7-58,60
24	RELTIM	相対時間(16 ビット以上の符号無し整数、時間単位は 1 ミリ秒)	HED-28 COM7-61,63
25	SYSTIM	システム時刻(16 ビット以上の符号無し整数、時間単位は 1 ミリ秒)	HED-29 COM7-64,66
26	VP_INT	データタイプが定まらないものへのポインタまたはプロセッサに自然なサイズの符号付き整数　(実装毎に定義)	HED-30 COM7-67,69
27	ER_BOOL	エラーコードまたは真偽値(符号付き整数)	HED-31 COM7-70,72,74,76
28	ER_ID	エラーコードまたは ID 番号(符号付き整数、負の ID 番号は表現できない)	HED-32 COM7-77,79
29	ER_UINT	エラーコードまたは符号無し整数(符号付き整数、符号無し整数を表現する場合の有効ビット数は UINT より 1 ビット短い)	HED-33 COM7-80,82

2.3.2 ITRON 仕様共通定数

項番	区　　分	確　認　事　項			ﾃｽﾄ手順参照
1	一　　般	NULL	無効ポインタ	0	HED-34
2		TRUE	真	1	HED-35
3		FALSE	偽	0	HED-36
4		E_OK	正常終了	0	HED-37
5	メイン エラーコード	E_SYS	システムエラー	-5	HED-38
6		E_NOSPT	未サポート機能	-9	HED-39
7		E_RSFN	予約機能コード	-10	HED-40
8		E_RSATR	予約属性	-11	HED-41
9		E_PAR	パラメータエラー	-17	HED-42
10		E_ID	不正 ID 番号	-18	HED-43
11		E_CTX	コンテキストエラー	-25	HED-44
12		E_MACV	メモリアクセス違反	-26	HED-45
13		E_ILUSE	サービスコール不正	-28	HED-46
14		E_NOMEM	メモリ不足	-33	HED-47
15		E_OBJ	オブジェクト状態エラー	-41	HED-48
16		E_QOVR	キューイングオーバーフロー	-43	HED-49
17		E_RLWAI	待ち状態の強制解除	-49	HED-50
18		E_TMOUT	ポーリング失敗またはタイムアウト	-50	HED-51
19	オブジェクト属性	TA_NULL	オブジェクト属性を指定しない	0	HED-52
20	タイムアウト 指　　定	TMO_POL	ポーリング	0	HED-53
21		TMO_FEVR	永久待ち	-1	HED-54

2.3.3 ITRON 仕様共通マクロ

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	ER mercd=MERCD(ER ercd)がヘッダファイル“ itron.h ”でマクロ定義され、エラーコードからメインエラーコードが取り出せること。	COM8-4 HED-55
2	ER sercd=SERCD(ER ercd) がヘッダファイル“ itron.h ”でマクロ定義され、エラーコードからサブエラーコードが取り出せること。	COM8-5 HED-56

2.3.4 ITRON 仕様共通静的 API

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	システムコンフィギュレーションファイルから、プリプロセッサ定数式パラメータおよび一般定数式パラメータの解釈に必要なＣ言語の宣言・定数とプリプロセッサマクロの定数などを含んだファイルをインクルードするための指定「INCLUDE(文字列);」ができること。	COM4-5,6

3. μITRON4.0 仕様の概念と共通定義

3.1 基本的な用語の意味

確認項目なし

3.2 タスク状態とスケジューリング規則

3.2.1 タスク状態

項番	確　　認　　事　　項		ﾃｽﾄ手順参照
1	ﾀｽｸの待ち	タスクの待ち解除とは実行可能状態に移行させること。	SPC1-5

3.2.2 タスクのスケジューリング規則

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	実行できるタスクが複数ある場合には、その中で最も優先順位の高いタスクが実行状態となり、他は実行可能状態となること。	SPC2-7
2	タスク間の優先順位は、異なる優先度を持つタスク間では、高い優先度を持つタスクの方が高い優先順位を持つこと。	SPC2-7
3	同じ優先度を持つタスク間では、先に実行できる状態(実行状態または実行可能状態)になったタスクの方が高い優先順位を持つこと。	SPC2-12
4	最も高い優先順位を持つタスクが変化した場合には、ただちにディスパッチが起こり、実行状態のタスクが切り替わること。	SPC2-9
5	ディスパッチが起こらない状態になっている場合には、実行状態のタスクの切替は、ディスパッチが起こる状態となるまで保留されること。	SPC3-7
6	優先順位の高いタスクが実行できる状態にある限り、それより優先順位の低いタスクは全く実行されないこと。すなわち、最も高い優先順位を持つタスクが待ち状態に入るなどの理由で実行できない状態とならない限り、他のタスクは全く実行されないこと。	SPC4-14
7	実行可能状態のタスクが実行状態になった後に実行可能状態に戻った直後には、同じ優先度を持つタスクの中で最高の優先順位を持っている。	SPC2-16
8	実行状態のタスクが待ち状態になった後に待ち解除されて実行できる状態になった直後には、同じ優先度を持つタスクの中で最低の優先順位となること。	SPC2-19

3.3 割込み処理モデル

3.3.1 割込みハンドラと割込みサービスルーチン

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	割込みハンドラの記述方法は実装毎に定義し、製品マニュアルに記載されていること。	UMA-7
2	DEF_INH と ATT_ISR の２つをサポートした場合は、その振舞いが製品マニュアルに記載されていること。	UMA-8
3	カーネルは、ある優先度より高い優先度を持つ割込みを、管理しないものとすることができる。どの優先度より高い優先度を持つものをカーネルの管理外の割込みとするかが製品マニュアルに記載されていること。	UMA-9

3.3.2 割込みの指定方法と割込みサービスルーチンの起動

確認項目なし

3.4 例外処理モデル

3.4.1 例外処理の枠組み

項番	確　　認　　事　　項	ﾃｽﾄ手順参照
1	CPU 例外ハンドラは、プロセッサが検出する CPU 例外によって起動されること。	SPC5-2
2	CPU 例外ハンドラは、CPU 例外の種類毎に、アプリケーションで登録できること。	SPC5-S3

3.4.2 CPU 例外ハンドラで行える操作

項番	確　　認　　事　　項	テスト手順参照
1	CPU 例外ハンドラの記述方法は実装毎に定義し、製品マニュアルに記載されていること。	UMA-10
2	CPU 例外ハンドラ内で呼出し可能なサービスコールは実装毎に定義し、製品マニュアルに記載されていること。	UMA-11
—	CPU 例外ハンドラ内で次の操作を行う方法を用意すること。また、その方法は製品マニュアルに記載されていること。	—
3	CPU 例外が発生したコンテキストや状態の読出しができること。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。	UMA-12
4	CPU 例外が発生したタスクの ID 番号が読み出せること(例外を発生させたのがタスクである場合のみ)。	UMA-13

(注)　CPU ロック状態で CPU 例外が発生した場合には、項番 4 の操作ができることは要求されない。

3.5 コンテキストとシステム状態

3.5.1 処理単位とコンテキスト

確認項目なし

3.5.2 タスクコンテキストと非タスクコンテキスト

項番	確　　認　　事　　項	テスト手順参照
1	タスクの実行されるコンテキストは、タスクコンテキストであること。	SPC6-2
2	割込みハンドラおよび周期ハンドラが実行されるコンテキストは、非タスクコンテキストである。	SPC6-5,17
3	非タスクコンテキストからサービスコールに渡すパラメータとして、自タスクを指定する TSK_SELF を用いることはできない。渡された場合には、E_ID エラーとなること。	SPC6-7
4	タスクコンテキストを実行中に CPU 例外が発生した場合、CPU 例外ハンドラが実行されるコンテキストが製品マニュアルに記載されていること。	UMA-14
5	非タスクコンテキストを実行中に CPU 例外が発生した場合には、CPU 例外ハンドラが実行されるコンテキストは非タスクコンテキストであること。	SPC6-10

3.5.3 処理の優先順位とサービスコールの不可分性

項番	確　　認　　事　　項	テスト手順参照
1	割込みハンドラの優先順位は、ディスパッチャの優先順位よりも高いこと。	TSK3-11
2	割込みハンドラおよび割込みサービスルーチン相互間の優先順位は、それらを起動する外部割込みの優先度に対応して定めることを基本に、実装毎に定義すること。	UMA-15
3	周期ハンドラの優先順位は、isig_tim を呼び出した割込みハンドラの優先順位以下で、ディスパッチャの優先順位よりも高いという範囲内で、実装毎に定義すること。	UMA-16
4	CPU 例外ハンドラの優先順位は、CPU 例外が発生した処理の優先順位と、ディスパッチャの優先順位のいずれよりも高いこと。	UMA-17
5	CPU 例外ハンドラと、割込みハンドラやタイムイベントハンドラとの間の優先順位は実装毎に定義すること。	UMA-18

3.5.4 CPU ロック状態

項番	確認事項			ﾃｽﾄ手順参照
1	CPU ロック状態では、割込みハンドラ(カーネルの管理外の割込みによって起動されるものを除く)や周期ハンドラは起動されず、ディスパッチも起こらないこと。すなわち、割込みが発生しても各ハンドラは起動されないこと。			SPC7-8,37
—	CPU ロック状態では、以下のサービスコールを呼び出すことができること。			—
2		loc_cpu	CPU ロック状態への移行	SPC7-10
3		iloc_cpu	CPU ロック状態への移行	SPC7-23
4		unl_cpu	CPU ロック状態の解除	SPC7-31
5		iunl_cpu	CPU ロック状態の解除	SPC7-27
6		sns_ctx	コンテキストの参照	SPC7-12
7		sns_loc	CPU ロック状態の参照	SPC7-14
8		sns_dsp	ディスパッチ禁止状態の参照	SPC7-16
9		sns_dpn	ディスパッチ保留状態の参照	SPC7-18
10	割込みハンドラ内で CPU ロック解除状態にするための方法が用意されていること。			UMA-19
11	割込みハンドラから正しくリターンするための方法が用意されていること。			UMA-20
12	割込みサービスルーチンと周期ハンドラの実行開始直後は、CPU ロック解除状態になっていること。			SPC7-42 SPC8-3
13	CPU 例外ハンドラの起動と、ハンドラからのリターンによって、CPU ロック／ロック解除状態が変化しないこと。			SPC9-5,8,13,16
14	タスクの実行開始直後は、CPU ロック解除状態になっていること。			SPC7-2

3.5.5 ディスパッチ禁止状態

項番	確認事項	ﾃｽﾄ手順参照
1	ディスパッチ禁止状態では、ディスパッチが起こらないこと。	SPC10-6
2	ディスパッチ禁止状態でも、自タスクを広義の待ち状態にする可能性がないサービスコールは呼び出せること。	SPC10-10,11
3	非タスクコンテキストから呼び出せるサービスコールは、ディスパッチ禁止状態でも制限を受けないこと。	SPC10-28
4	割込みハンドラの起動と、そこからのリターンによってディスパッチ禁止／許可状態は変化しないこと。	SPC10-23,42
5	割込みサービスルーチンの起動と、そこからのリータンによってディスパッチ禁止／許可状態は変化しないこと。	SPC10-23,42
6	周期ハンドラの起動と、そこからのリータンによってディスパッチ禁止／許可状態は変化しないこと。	SPC10-33,47
7	タスクの実行開始直後は、ディスパッチ許可状態になっていること。	SPC10-2
8	ポーリングを行うサービスコールは、自タスクを待ち状態にする可能性がないため、ディスパッチ禁止状態でも呼び出すことができること。	SPC10-8
9	ディスパッチ禁止状態で CPU ロック状態に移行し、その後 CPU ロックを解除しても、ディスパッチ禁止状態は保持されていること。	SPC10-19
10	CPU ロック状態でも、ディスパッチ禁止状態かディスパッチ許可状態かを参照できること。	SPC10-15,51

（注）
項番 4：割込みハンドラをサポートしていないシステムでは、テストを省略できる。
項番 5：割込みサービスルーチンをサポートしていないシステムでは、テストを省略できる。

3.5.6 ディスパッチ保留状態の間のタスク状態

項番	確認事項	ﾃｽﾄ手順参照
1	ディスパッチ保留状態では、実行中のタスクがプリエンプトされるべき状況になっても、新たに実行すべき状態となったタスクにはディスパッチされないこと。実行すべきタスクへのディスパッチは、ディスパッチが起こる状態となるまで保留されること。	SPC11-6

3.6 非タスクコンテキストからのサービスコール呼出し

3.6.1 非タスクコンテキストから呼び出せるサービスコール

項番	確認事項			ﾃｽﾄ手順参照
－	非タスクコンテキスト専用のサービスコール			－
1		iact_tsk	タスクの起動	TSK3-7
2		iwup_tsk	タスクの起床	SYN2-10
3		irel_wai	待ち状態の強制解除	SYN5-9
4		isig_sem	セマフォ資源の返却	SEM3-16
5		iset_flg	イベントフラグのセット	FLG3-9
6		ipsnd_dtq	データキューへの送信(ポーリング)	DTQ3-10
7		ifsnd_dtq	データキューへの強制送信	DTQ5-12
8		isig_tim	タイムティックの供給	SPC12-19
9		iget_tid	実行状態のタスク ID の参照	SYS2-3
10		iloc_cpu	CPU ロック状態への移行	SYS2-5
11		iunl_cpu	CPU ロック状態の解除	SYS2-13
－	どのコンテキストからでも呼び出せるサービスコール			－
12		sns_ctx	コンテキストの参照	SPC12-2,11
13		sns_loc	CPU ロック状態の参照	SPC12-4,13
14		sns_dsp	ディスパッチ禁止状態の参照	SPC12-6,15
15		sns_dpn	ディスパッチ保留状態の参照	SPC12-8,17

3.6.2 サービスコールの遅延実行

項番	確認事項	ﾃｽﾄ手順参照
1	サービスコールを遅延実行する場合にも、遅延実行することを許されないものを除いて、サービスコールの実行順序がサービスコールの呼び出された順序に一致しなければならないこと。	SPC13-15

3.6.3 呼び出せるサービスコールの追加

確認項目なし

3.7 システム初期化手順

項番	確認事項	ﾃｽﾄ手順参照
1	ATT_INI 以外の静的 API の処理は、システムコンフィギュレーションファイル中に記述された順序でおこなうこと。	SPC14-8
2	初期化ルーチンはアプリケーションで用意され、ATT_INI を使って登録され、実行されること。	SPC14-3
3	カーネルの初期化処理を呼び出す方法は、実装毎に定義すること。	UMA-21
4	静的 API の処理中にエラーを検出した場合の扱いは、実装毎に定義すること。	UMA-22
5	カーネルの管理外の割込みを禁止するかどうかは、実装毎に定義すること。	UMA-23
6	初期化ルーチン内でサービスコールを呼び出せるか否か、呼び出せる場合にはどのようなサービスコールが呼び出せるかは、実装毎に定義すること。	UMA-24
7	初期化ルーチンは、システムコンフィギュレーションファイル中に ATT_INI の記述のある順序で実行すること。	SPC14-3
8	カーネルの動作開始時点で初めて割込みを許可すること。	SPC14-4

3.8 オブジェクトの登録とその解除

項番	確認事項	ﾃｽﾄ手順参照
1	ID 番号で識別されるオブジェクトについては、少なくとも 255 個のオブジェクトが登録できること。	SPC15-4
2	カーネルに登録できるオブジェクトの最大数や ID 番号の範囲は、実装毎に定義すること。	UMA-25
3	自動割付けされる ID 番号の範囲の指定方法は、実装毎に定義すること。	UMA-26

3.9 処理単位の記述形式

確認項目なし

3.10 カーネル構成定数／マクロ

確認項目なし

3.11 カーネル共通定義

3.11.1 カーネル共通定義

項番	区　　分	確　認　事　項			ﾃｽﾄ手順参照
1	オブジェクト属　性	TA_HLNG	高級言語用のインタフェースで処理単位で起動	0x00	HED-57
2		TA_TFIFO	タスクの待ち行列を FIFO 順に	0x00	HED-58
3	その他のカーネル共通定義	TSK_SELF	自タスク指定	0	HED-59
4		TSK_NONE	該当するタスクが無い	0	HED-60

3.11.2 カーネル共通構成定数

項番	区　　分	確　認　事　項			ﾃｽﾄ手順参照
1	優先度の範囲	TMIN_TPRI	タスク優先度の最小値	1	HED-61
2		TMAX_TPRI	タスク優先度の最大値	16 以上	HED-62
4	バージョン情報	TKERNEL_MAKER	カーネルのメーカーコード		HED-63
5		TKERNEL_PRID	カーネルの識別番号		HED-64
6		TKERNEL_SPVER	ITRON 仕様のバージョン番号		HED-65
7		TKERNEL_PRVER	カーネルのバージョン番号		HED-66

4. μITRON4.0 仕様の機能

4.1 タスク管理機能

No.	区　　分	テ　ス　ト　項　目			ﾃｽﾄ手順参照
1	タスクの生成時に行うべき処理	タスクの起動要求キューイング数をクリアすること。			TSK1-3
2	タスクの起動時に行うべき処理	タスクの優先度を初期化すること。			TSK1-16
3		起床要求キューイング数をクリアすること。			TSK1-18
4	タスクの終了	タスクのメインルーチンからリターンした場合にはタスクを終了すること。			(Ref) return-1
5	定　　数	TMAX_ACTCNT	タスクの起動要求キューイング数の最大値	1 以上	HED-67

CRE_TSK：タスクの生成（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号	TSK1-S4
2		E_RSATR	予約属性(tskatr が不正あるいは使用できない)	TSK1-S6
3		E_PAR	パラメータエラー(task が不正)	TSK1-S8
4		E_PAR	パラメータエラー(itskpri が不正)	TSK1-S10
5		E_PAR	パラメータエラー(stksz が不正)	TSK1-S12
6		E_OBJ	オブジェクト状態エラー(対象タスクが登録済)	TSK1-S14
7	静的 API 処　　理	タスクの生成時に行うべき処理を行うこと。		(Ref)タスク管理機能-1
―		tskatr に TA_ACT の指定がない場合		―
8		対象タスクを未登録状態から休止状態に移行させること。		TSK1-5
―		tskatr に TA_ACT の指定がある場合		―
9		対象タスクを未登録状態から実行可能状態に移行させること。		TSK1-1
10		タスクの起動時に行うべき処理を行うこと。		(Ref)タスク管理機能-2,3
11		拡張情報(exinf)がパラメータとして渡ること。また、起動されたタスクでは、起動情報が正しく受取れること。		TSK1-1

(注)

No.3　E_PAR エラーを検出できる task がシステムに存在しない場合は、テストを省略することができる。

No.5　E_PAR エラーを検出できる stksz がシステムに存在しない場合は、テストを省略することができる。

act_tsk：タスクの起動

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK2-2
2		E_QOVR	キューイングオーバーフロー(起動要求キューイング数のオーバーフロー)	TSK2-20
3	サービスコール 処　　理	対象タスクが休止状態である場合には、tskid で指定されるタスクを、休止状態から実行可能状態に移行させること。		TSK2-12
4		タスクの起動時に行うべき処理を行うこと。		(Ref)タスク管理機能-2,3
5		タスクを起動する際のパラメータは、生成時に指定したタスクの拡張情報(exinf)と一致すること。また、起動されたタスクでは、拡張情報が正しく受け取れること。		TSK2-12
6		対象タスクが休止状態でない場合には、tskid で指定されるタスクに対する起動要求をキューイング(タスクの起動要求キューイング数に１を加える)すること。		TSK2-24
7		tskid に TSK_SELF が指定されると、自タスクを対象とすること。		TSK2-4,6

iact_tsk：タスクの起動

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK3-3
2		E_QOVR	キューイングオーバーフロー(起動要求キューイング数のオーバーフロー) サービスコールを遅延実行する場合で、E_QOVR エラーを返すことを実装定義で省略できる。省略する場合は、E_OK を返すこと。(注)	TSK3-9
3	サービスコール処　　理	対象タスクが休止状態である場合には、tskid で指定されるタスクを、休止状態から実行可能状態に移行させること。		TSK3-13
4		タスクの起動時に行うべき処理を行うこと。		(Ref)タスク管理機能-2,3
5		タスクを起動する際のパラメータは、生成時に指定したタスクの拡張情報(exinf)と一致すること。また、起動されたタスクでは、拡張情報が正しく受け取れること。		TSK3-13
6		対象タスクが休止状態でない場合には、tskid で指定されるタスクに対する起動要求をキューイング(タスクの起動要求キューイング数に１を加える)すること。		TSK3-16
7		tskid に TSK_SELF が指定された場合には、E_ID エラーを返すこと。		TSK3-5

(注)

No.2　サービスコールを遅延実行する場合で、E_QOVR エラーを省略することが製品マニュアルに記載されていれば、E_OK を返すことができる。

can_act：タスク起動要求のキャンセル

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK2-10
2	サービスコール処　　理	タスクの起動要求キューイング数を０にクリアすること。		TSK2-8
3		０クリアする前の起動要求キューイング数を返値として actcnt に返すこと。		TSK2-16
4		tskid に TSK_SELF が指定されると、自タスクを対象とすること。		TSK2-6

ext_tsk：自タスクの終了

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	サービスコール処　　理	自タスクを実行状態から休止状態に移行させること。		TSK4-7
―		自タスクに対する起動要求がキューイングされている場合		―
2			起動要求キューイング数が１以上の場合には、起動要求キューイング数から１を減じること。	TSK4-11
3			タスクの起動時に行うべき処理を行うこと。	(Ref)タスク管理機能-2,3
4			実行可能状態に移行させること。	TSK4-9
5			タスクを起動する際のパラメータとして、タスクの拡張情報(exinf)を渡すこと。また、起動されたタスクでは、拡張情報が正しく受け取れること。	TSK4-9
6		サービスコール内でエラーが発生した場合の処理。		UMA-27

return：自タスクの終了

No.	区　　分	テ　ス　ト　項　目		テスト手順参照
1	サービスコール処　　理	自タスクを実行状態から休止状態に移行させること。		TSK5-7
—		自タスクに対する起動要求がキューイングされている場合		—
2		起動要求キューイング数が１以上の場合には、起動要求キューイング数から１を減じること。		TSK5-11
3		タスクの起動時に行うべき処理を行い、実行可能状態に移行させること。		(Ref)タスク管理機能-2,3
4		タスクを起動する際のパラメータとして、タスクの拡張情報(exinf)を渡すこと。また、起動されたタスクでは、拡張情報が正しく受け取れること。		TSK5-9

ter_tsk：タスクの強制終了

No.	区　　分	テ　ス　ト　項　目		テスト手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK6-2
2		E_ILUSE	サービスコール不正使用(対象タスクが自タスク)	TSK6-4
3		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態)	TSK6-6
4	サービスコール処　　理	tskid で指定されるタスクを、休止状態に移行させること。		TSK6-10
—		対象タスクに対する起動要求がキューイングされている場合		—
5		起動要求キューイング数から１を減じること。		TSK6-20
6		対象タスクを実行可能状態に移行させること。		TSK6-18
7		タスクの起動すべき処理を行うこと。		(Ref)タスク管理機能-2,3
8		タスクを起動する際のパラメータとして、タスクの拡張情報を渡すこと。また、起動されたタスクでは、拡張情報だ正しく受け取れること。		TSK6-18

chg_pri：タスク優先度の変更

No.	区　　分	テ　ス　ト　項　目		テスト手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK7-2
2		E_PAR	パラメータエラー(tskpri が不正)	TSK7-4
3		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態)	TSK7-6
4	サービスコール処　　理	tskid で指定されるタスクの優先度を、tskpri で指定される値に変更すること。		TSK7-24
5		tskid に TSK_SELF が指定されると、自タスクを対象とすること。		TSK7-32
6		tskpri に TPRI_INI が指定されると、タスクの起動時優先度に変更すること。		TSK7-37
—		対象タスクが実行できる状態である場合		—
7		タスクの優先順位を、変更後の優先度にしたがって変化させること。		TSK7-22
8		変更後の優先度と同じ優先度を持つタスクの間では、対象タスクの優先順位を最低とすること。		TSK7-32

get_pri：タスク優先度の参照

No.	区　　分	テ　ス　ト　項　目		テスト手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	TSK7-8
2		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態)	TSK7-10
3	サービスコール処　　理	tskid で指定されるタスクの優先度を参照し、tskpri に返すこと。		TSK7-13
4		tskid に TSK_SELF が指定されると、自タスクを対象とすること。		TSK7-16

4.2　タスク付属同期機能

No.	区　　分	テ　ス　ト　項　目			テスト手順参照
1	定　　数	TMAX_WUPCNT	タスクの起床要求キューイング数の最大値	１以上	HED-68

slp_tsk：起床待ち

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	SYN1-17
－	サービスコール処　理	自タスクに対する起床要求がキューイングされていない場合		－
2		自タスクを起床待ち状態に移行させること。		SYN1-25
－		自タスクに対する起床要求がキューイングされている場合		－
3		起床要求キューイング数から１を減じること。		SYN1-23
4		自タスクを待ち状態に移行させずに、そのまま実行を継続すること。		SYN1-8

wup_tsk：タスクの起床

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	SYN1-2
2		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態)	SYN1-4
3		E_QOVR	キューイングオーバーフロー(起床要求キューイング数のオーバーフロー)	SYN1-14
4	サービスコール処　理	tskid で指定されるタスクを、起床待ち状態から待ち解除すること。		SYN1-25
5		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		SYN1-25
6		対象タスクが起床待ち状態でない場合には、タスクの起床要求キューイング数に１を加えること。		SYN1-21
7		tskid に TSK_SELF が指定されると、自タスクを対象とすること。		SYN1-8

iwup_tsk：タスクの起床

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	SYN2-6
2		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態) サービスコールを遅延実行する場合で、E_OBJ エラーを返すことを実装定義で省略できる。省略する場合は E_OK を返すこと。(注)	SYN2-8
3		E_QOVR	キューイングオーバーフロー(起床要求キューイング数のオーバーフロー) サービスコールを遅延実行する場合で、E_QOVR エラーを返すことを実装定義で省略できる。省略する場合は E_OK を返すこと。(注)	SYN2-14
4	サービスコール処　理	tskid で指定されるタスクを、起床待ち状態から待ち解除すること。		SYN2-16
5		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		SYN2-16
6		対象タスクが起床待ち状態でない場合には、タスクの起床要求キューイング数に１を加えること。		SYN2-19
7		tskid に TSK_SELF が指定されると、E_ID エラーを返すこと。		SYN2-4

(注)

No.2　サービスコールを遅延実行する場合で、E_OBJ エラーを省略することが製品マニュアルに記載されていれば、E_OK を返すことができる。

No.3　サービスコールを遅延実行する場合で、E_QOVR エラーを省略することが製品マニュアルに記載されていれば、E_OK を返すことができる。

can_wup：タスク起床要求のキャンセル

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	SYN3-2
2		E_OBJ	オブジェクト状態エラー(対象タスクが休止状態)	SYN3-4
3	サービスコール処　　理	タスクの起床要求キューイング数を 0 にクリアすること。		SYN3-16
4		クリアする前の起床要求キューイング数を wupcnt に返すこと。		SYN3-14
5		tskid に TSK_SELF が指定されると、自タスクを対象タスクとすること。		SYN3-8

rel_wai：待ち状態の強制解除

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	SYN4-2
2		E_OBJ	オブジェクト状態エラー(対象タスクが待ち状態でない)	SYN4-4
3	サービスコール処理	対象タスクが待ち状態の時は実行可能状態に移行させること。		SYN4-7
4		このサービスコールにより待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_RLWAI を返すこと。		SYN4-7

irel_wai：待ち状態の強制解除

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(tskid が不正あるいは使用できない)	SYN5-5
2		E_OBJ	オブジェクト状態エラー(対象タスクが待ち状態でない) サービスコールを遅延実行する場合で、E_OBJ エラーを返すことを実装定義で省略できる。省略する場合は、E_OK を返すこと。(注)	SYN5-7
3	サービスコール処理	対象タスクが待ち状態の時は実行可能状態に移行させること。		SYN5-13
4		このサービスコールにより待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_RLWAI を返すこと。		SYN5-13

(注)

No.2　サービスコールを遅延実行する場合で、E_OBJ エラーを省略することが製品マニュアルに記載されていれば、E_OK を返すことができる。

4.3　タスク例外処理機能

確認項目なし

4.4　同期・通信機能

4.4.1　セマフォ

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	最大資源数	セマフォの最大資源数の最大値	65535 以上	HED-69

CRE_SEM：セマフォの生成（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(semid が不正あるいは使用できない)	SEM1-S2
2		E_RSATR	予約属性(sematr が不正あるいは使用できない)	SEM1-S4
3		E_PAR	パラメータエラー(isemcnt>maxsem)	SEM1-S6
4		E_PAR	パラメータエラー(maxsem が最大資源数を越えた)	SEM1-S8
5		E_OBJ	オブジェクト状態エラー(対象セマフォが登録済)	SEM1-S11
—	静的 API 処　理	sematr には以下の指定ができ、オブジェクトが生成できること。		—
6		TA_TFIFO		SEM2-29
7		isemcnt で示す初期値が正しいこと。		SEM2-14
8		maxsem で示す最大資源数が正しいこと。		SEM2-8

(注)

No.4　E_PAR エラーを検出できる maxsem がシステムに存在しない場合は、テストを省略することができる。

sig_sem：セマフォ資源の返却

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(semid が不正あるいは使用できない)	SEM2-2
2		E_QOVR	キューイングオーバーフロー(最大資源数を越える返却)	SEM2-8
—	サービスコール 処　理	資源の獲得を待っているタスクがある場合		—
3		待ち行列の先頭のタスクを待ち解除すること。		SEM2-29
4		対象セマフォの資源数は変化してはならないこと。		SEM2-31
5		待ち状態に入ったサービスコールの返値として E_OK を返すこと。		SEM2-29
—		資源の獲得を待っているタスクがない場合		—
6		対象セマフォの資源数に１を加えること。		SEM2-16

isig_sem：セマフォ資源の返却

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(semid が不正あるいは使用できない)	SEM3-14
2		E_QOVR	キューイングオーバーフロー(最大資源数を越える返却) サービスコールを遅延実行する場合で、E_QOVR エラーを返すことこを実装定義で省略できる。省略する場合は、E_OK を返すこと。(注)	SEM3-18
—	サービスコール 処　理	資源の獲得を待っているタスクがある場合		—
3		待ち行列の先頭のタスクを待ち解除すること。		SEM3-22
4		対象セマフォの資源数は変化してはならないこと。		SEM3-25
5		待ち状態に入ったサービスコールの返値として E_OK を返すこと。		SEM3-22
—		資源の獲得を待っているタスクがない場合		—
6		対象セマフォの資源数に１を加えること。		SEM3-35

(注)

No.2　サービスコールを遅延実行する場合で、E_QOVR エラーを省略することが製品マニュアルに記載されていれば、E_OK を返すことができる。

wai_sem：セマフォ資源の獲得

No.	区　　分	テ　ス　ト　項　目		テスト手順参照
1	エラーコード	E_ID	不正 ID 番号(semid が不正あるいは使用できない)	SEM2-4
2		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	SEM2-37
―	サービスコール処　　理	対象セマフォの資源数が１以上の場合		―
3		セマフォの資源数から１を減じること。		SEM2-10
4		待ち状態とせずにサービスコールの処理を終了すること。		SEM2-10
―		対象セマフォの資源数が０の場合		―
5		対象セマフォの資源数は０のまま変化してはならないこと。		SEM2-33
―		セマフォ属性が TA_TFIFO		―
6		自タスクを待ち行列の末尾につなぐこと。		SEM2-33

pol_sem：セマフォ資源の獲得(ポーリング)

No.	区　　分	テ　ス　ト　項　目		テスト手順参照
1	エラーコード	E_ID	不正 ID 番号(semid が不正あるいは使用できない)	SEM2-6
2		E_TMOUT	ポーリング失敗	SEM2-14
―	サービスコール処　　理	対象セマフォの資源数が１以上の場合		―
3		セマフォの資源数から１を減じること。		SEM2-12
4		待ち状態とせずにサービスコールの処理を終了すること。		SEM2-12

4.4.2　イベントフラグ

No.	区　　分	テ　ス　ト　項　目			テスト手順参照
1	定　　数	FLGPTN	イベントフラグのビットパターン(符号無し整数)	16 ビット以上	HED-70
2		TBIT_FLGPTN	イベントフラグのビット数	16 ビット以上	HED-71

CRE_FLG：イベントフラグの生成（静的 API）

No.	区　　分	テ　ス　ト　項　目		テスト手順参照
1	エラーコード	E_ID	不正 ID 番号(flgid が不正あるいは使用できない)	FLG1-S2
2		E_RSATR	予約属性(flgatr が不正あるいは使用できない)	FLG1-S4
3		E_PAR	パラメータエラー(iflgptn が不正)	FLG1-S6
4		E_OBJ	オブジェクト状態エラー(対象イベントフラグが登録済)	FLG1-S9
―	静的 API 処　　理	flgatr には以下の指定ができ、オブジェクトが生成できること。		―
5		TA_TFIFO		FLG2-26
6		TA_TFIFO｜TA_WSGL		FLG2-26
7		TA_TFIFO｜TA_CLR		FLG3-20
8		TA_TFIFO｜TA_WSGL｜TA_CLR		FLG3-20
9		iflgptn で示すビットパターンの初期値が正しいこと。		FLG2-73

(注)

No.3　E_PAR エラーを検出できる iflgptn がシステムに存在しない場合は、テストを省略することができる。

set_flg：イベントフラグのセット

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(flgid が不正あるいは使用できない)	FLG2-2
2		E_PAR	パラメータエラー(setptn が不正)	FLG2-4
3	サービスコール処　理	対象イベントフラグのビットパターンを、サービスコール呼出し前のビットパターンと setptn の値のビット毎の論理和に更新すること。		FLG2-49
4		対象イベントフラグのビットパターンが更新された結果、そのイベントフラグで待っているタスクの待ち解除条件を満たした場合には、該当するタスクを待ち解除すること。		FLG2-64
―		待ち解除されたタスクに対しての処理		―
5		待ち状態に入ったサービスコールの返値として E_OK を返すこと。		FLG2-64,84
6		待ち解除時のビットパターンとして、この時の(待ち解除条件を満たした)ビットパターンを返すこと。		FLG2-65,85
7		TA_CLR が指定されている場合には、イベントフラグのビットパターンのすべてのビットをクリアし、サービスコールの処理を終了すること。		FLG2-87

（注）

No.2　E_PAR エラーを検出できる setptn がシステムに存在しない場合は、テストを省略することができる。

iset_flg：イベントフラグのセット

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(flgid が不正あるいは使用できない)	FLG3-5
2		E_PAR	パラメータエラー(setptn が不正)	FLG3-7
3	サービスコール処　理	対象イベントフラグのビットパターンを、サービスコール呼出し前のビットパターンと setptn の値のビット毎の論理和に更新すること。		FLG3-18
4		対象イベントフラグのビットパターンが更新された結果、そのイベントフラグで待っているタスクの待ち解除条件を満たした場合には、該当するタスクを待ち解除すること。		FLG3-17
―		待ち解除されたタスクに対しての処理		―
5		待ち状態に入ったサービスコールの返値として E_OK を返すこと。		FLG3-17
6		待ち解除時のビットパターンとして、この時の(待ち解除条件を満たした)ビットパターンを返すこと。		FLG3-18
7		TA_CLR が指定されている場合には、イベントフラグのビットパターンのすべてのビットをクリアし、サービスコールの処理を終了すること。		FLG3-23

（注）

No.2　E_PAR エラーを検出できる setptn がシステムに存在しない場合は、テストを省略することができる。

clr_flg：イベントフラグのクリア

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(flgid が不正あるいは使用できない)	FLG2-6
2		E_PAR	パラメータエラー(clrptn が不正)	FLG2-8
3	サービスコール処　理	対象イベントフラグのビットパターンを、サービスコール呼出し前のビットパターンと clrptn の値のビット毎の論理積に更新すること。		FLG2-37

（注）

No.2　E_PAR エラーを検出できる clrptn がシステムに存在しない場合は、テストを省略することができる。

wai_flg：イベントフラグ待ち

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(flgid が不正あるいは使用できない)	FLG2-10
2		E_PAR	パラメータエラー(waiptn が不正)	FLG2-12
3		E_PAR	パラメータエラー(wfmode が不正)	FLG2-14
4		E_PAR	パラメータエラー(p_flgptn が不正)	FLG2-16
5		E_ILUSE	サービスコール不正使用(TA_WSGL 属性が指定されたイベントフラグで待ちタスクあり)	FLG2-56
6		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	FLG2-60
—	サービスコール処　理	wfmode には、以下が指定できること。		—
7		TWF_ANDW		FLG2-56,69
8		TWF_ORW		FLG2-64,72
—		対象イベントフラグのビットパターンが waiptn と wfmode で指定される待ち条件を満たしている場合		—
9		自タスクを待ち状態とせずにサービスコールの処理を終了すること。		FLG2-30,69
10		flgptn には、この時の(待ち解除条件を満たした)ビットパターンを返すこと。		FLG2-31,70
11		TA_CLR が指定されている場合には、イベントフラグのビットパターンのすべてのビットをクリアすること。		FLG2-94
—		対象イベントフラグのビットパターンが waiptn と wfmode で指定される解除条件を満たしていない場合		—
12		自タスクを待ち行列につなぎ、イベントフラグ待ち状態に移行させること。		FLG2-60
13		waiptn に 0 を指定した場合には、E_PAR エラーを返すこと。		FLG2-67

(注)

No.2　E_PAR エラーを検出できる waiptn がシステムに存在しない場合は、テストを省略することができる。

No.4　E_PAR エラーを検出できる p_flgptn がシステムに存在しない場合は、テストを省略することができる。

pol_flg：イベントフラグ待ち(ポーリング)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(flgid が不正あるいは使用できない)	FLG2-18
2		E_PAR	パラメータエラー(waiptn が不正)	FLG2-20
3		E_PAR	パラメータエラー(wfmode が不正)	FLG2-22
4		E_PAR	パラメータエラー(p_flgptn が不正)	FLG2-24
5		E_ILUSE	サービスコール不正使用(TA_WSGL 属性が指定されたイベントフラグで待ちタスクあり)	FLG2-58
6		E_TMOUT	ポーリング失敗	FLG2-33
—	サービスコール処　理	wfmode には、以下が指定できること。		—
7		TWF_ANDW		FLG2-37,77
8		TWF_ORW		FLG2-43,84
—		対象イベントフラグのビットパターンが waiptn と wfmode で指定される待ち条件を満たしている場合		—
9		flgptn には、この時の(待ち解除条件を満たした)ビットパターンを返すこと。		FLG2-44
10		属性に TA_CLR が指定されている場合には、イベントフラグのビットパターンのすべてのビットをクリアすること。		FLG2-80
11		waiptn に 0 を指定した場合には、E_PAR エラーを返すこと。		FLG2-39

（注）

No.2　E_PAR エラーを検出できる waiptn がシステムに存在しない場合は、テストを省略することができる。

No.4　E_PAR エラーを検出できる p_flgptn がシステムに存在しない場合は、テストを省略することができる。

4.4.3 データキュー

CRE_DTQ：データキューの生成（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ1-S2
2		E_RSATR	予約属性(dtqatr が不正あるいは使用できない)	DTQ1-S4
3		E_PAR	パラメータエラー(dtqcnt が不正)	DTQ1-S6
4		E_OBJ	オブジェクト状態エラー(対象データキューが登録済)	DTQ1-S9
－	静的 API 処　　理	dtqatr には以下の指定ができ、オブジェクトが生成できること。		－
5		TA_TFIFO		DTQ2-7
6		データキュー領域に格納できるデータの個数が dtqcnt で指定できること。		DTQ2-10

(注)

No.3　E_PAR エラーを検出できる dtqcnt がシステムに存在しない場合は、テストを省略することができる。

psnd_dtq：データキューへの送信(ポーリング)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ2-2
2		E_TMOUT	ポーリング失敗(対象データキューで受信を待っているタスクがなく、データキュー領域に空きがない場合には、E_TMOUT エラーを返すこと。)	DTQ2-4
3	サービスコール 処　　理	dtqid で指定されるデータキューに、data で指定されるデータを送信できること。		DTQ2-8
4		送信されるデータは、送信側から受信側にコピーされること。		DTQ2-8
－		対象データキューで受信を待っているタスクがある場合		－
5		受信待ち行列の先頭のタスクに送信するデータを渡し、そのタスクの待ちを解除すること。		DTQ2-7
6		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ2-7
7		データキューから受信したデータとして data の値を返すこと。		DTQ2-8
－		データキューで送信されるデータ(１ワードのメッセージ)		－
8		整数値であること。		DTQ2-8
9		送信側と受信側で共有しているメモリ上に置かれたメッセージの先頭番地であること。		DTQ2-13

ipsnd_dtq：データキューへの送信(ポーリング)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ3-4
2		E_TMOUT	ポーリング失敗(サービスコールを遅延実行する場合で、E_TMOUT エラーを返すことを実装定義で省略できる。省略する場合は E_OK を返すこと。)	DTQ3-6
3	サービスコール処　　理	dtqid で指定されるデータキューに、data で指定されるデータを送信できること。		DTQ3-13
4		送信されるデータは、送信側から受信側にコピーされること。		DTQ3-13
—		対象データキューで受信を待っているタスクがある場合		—
5		受信待ち行列の先頭のタスクに送信するデータを渡し、そのタスクの待ちを解除すること。		DTQ3-12
6		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ3-12
7		データキューから受信したデータとして VP_INT data の値を返すこと。		DTQ3-12
—		データキューで送信されるデータ(１ワードのメッセージ)		—
8		整数値であること。		DTQ3-13
9		送信側と受信側で共有しているメモリ上に置かれたメッセージの先頭番地であること。		DTQ3-16

(注)

No.2　サービスコールを遅延実行する場合で、E_TMOUT エラーを省略することが製品マニュアルに記載されていれば、E_OK を返すことができる。

fsnd_dtq：データキューへの強制送信

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ4-2
2		E_ILUSE	サービスコール不正使用(対象データキューのデータキュー領域の容量が０)	DTQ4-4
—	サービスコール処　　理	対象データキューで受信を待っているタスクがある場合		—
3		受信待ち行列の先頭のタスクに送信するデータを渡し、そのタスクの待ちを解除すること。		DTQ4-14
4		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ4-14
5		データキューから受信したデータとして data の値を返すこと。		DTQ4-15
—		受信を待っているタスクがない場合		—
6		送信するデータをデータキューの末尾に入れること。		DTQ4-25
7		データキュー領域に空きがない場合には、データキューの先頭のデータを抹消し、データキュー領域に必要な領域を確保すること。		DTQ4-11

ifsnd_dtq：データキューへの強制送信

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ5-4
2		E_ILUSE	サービスコール不正使用(対象データキューのデータキュー領域の容量が 0)	DTQ5-6
—	サービスコール処　理	対象データキューで受信を待っているタスクがある場合		—
3		受信待ち行列の先頭のタスクに送信するデータを渡し、そのタスクの待ちを解除すること。		DTQ5-18
4		待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの返値として E_OK を返すこと。		DTQ5-18
5		データキューから受信したデータとして VP_INT data の値を返すこと。		DTQ5-19
—		受信を待っているタスクがない場合		—
6		送信するデータをデータキューの末尾に入れること。		DTQ5-29
7		データキュー領域に空きがない場合には、データキューの先頭のデータを抹消し、データキュー領域に必要な領域を確保すること。		DTQ5-23

rcv_dtq：データキューからの受信

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ6-2
2		E_PAR	パラメータエラー(p_data が不正)	DTQ6-4
3		E_RLWAI	待ち状態の強制解除(待ち状態の間に rel_wai を受付)	DTQ6-37
—	サービスコール処　理	対象データキューにデータが入っている場合		—
4		先頭のデータを取り出し、data に返すこと。		DTQ6-9
—		データが入っていない場合		—
5		自タスクを受信待ち行列につなぎ、データキューからの受信待ち状態に移行させること。		DTQ6-13
6		他のタスクがすでに受信待ち行列につながっている場合には、自タスクを受信待ち行列の末尾につなぐこと。		DTQ6-18

（注）

No.2　E_PAR エラーを検出できる p_data がシステムに存在しない場合は、テストを省略することができる。

prcv_dtq：データキューからの受信(ポーリング)

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号(dtqid が不正あるいは使用できない)	DTQ7-2
2		E_PAR	パラメータエラー(p_data が不正)	DTQ7-5
3		E_TMOUT	ポーリング失敗	DTQ7-7
—	サービスコール処　理	対象データキューにデータが入っている場合		—
4		先頭のデータを取り出し、data に返すこと。		DTQ7-13

（注）

No.2　E_PAR エラーを検出できる p_data がシステムに存在しない場合は、テストを省略することができる。

4.7 時間管理機能

4.7.1 システム時刻管理

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	時刻の更新	アプリケーションが定められた周期で isig_tim を呼出し、システム時刻が更新されること。		CYC2-10
2		システム時刻更新機能がカーネル内部にある場合は、isig_tim を呼び出さなくてもシステム時刻が更新されること。ただし、システム時刻更新方法が製品マニュアルに記載されていること。		UMA-28,29
3	ｼｽﾃﾑ時刻に依存して行う処理	周期ハンドラの起動		CYC2-29
4	定　　数	TIC_NUME	タイムティックの周期の分子	HED-72
5		TIC_DENO	タイムティックの周期の分母	HED-73

isig_tim：タイムティックの供給

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　　理	システム時刻を更新できること。	CYC2-10
2		システム時刻を更新する機構をカーネル内部に持つ場合には、このサービスコールをサポートする必要はない。	UMA-28

4.7.2 周期ハンドラ

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	拡張情報	周期ハンドラを起動すべき時刻になると、その周期ハンドラの拡張情報(exinf)をパラメータとして、周期ハンドラを起動すること。	CYC2-29
2	停止状態	周期ハンドラを起動すべき時刻になっても周期ハンドラを起動せず、次に起動すべき時刻の決定のみを行うこと。	CYC2-9

CRE_CYC：周期ハンドラの生成（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_RSATR	予約属性(cycatr が不正あるいは使用できない)	CYC1-S2
2		E_PAR	パラメータエラー(cychdr が不正)	CYC1-S4
3		E_PAR	パラメータエラー(cyctim が 0)	CYC1-S6
4		E_PAR	パラメータエラー(cycphs が不正)	CYC1-S8
—	静的 API 処　　理	TA_STA 指定の場合		—
5		周期ハンドラを生成した後に、周期ハンドラを動作している状態とすること。		CYC2-10
—		TA_STA 指定なしの場合		—
6		周期ハンドラを動作していない状態とすること。		CYC2-9
7		周期ハンドラを最初に起動すべき時刻は、これらのサービスコールが呼び出された時刻(静的 API の場合にはシステム初期化の時刻)に、指定された起動位相を加えた時刻とすること。		CYC2-10

(注)

No.2　E_PAR エラーを検出できる cychdr がシステムに存在しない場合は、テストを省略することができる。

sta_cyc：周期ハンドラの動作開始

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号 (cycid が不正あるいは使用できない)	CYC2-3
2	サービスコール処　　理	cycid で指定される周期ハンドラを、動作している状態に移行させること。		CYC2-19
3		対象周期ハンドラに、周期ハンドラ属性に TA_PHS が指定されていないで動作している状態の周期ハンドラが指定された場合には、周期ハンドラを次に起動すべき時刻の再設定のみを行うこと。		CYC2-24

stp_cyc：周期ハンドラの動作停止

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_ID	不正 ID 番号 (cycid が不正あるいは使用できない)	CYC2-5
2	サービスコール処　　理	cycid で指定される周期ハンドラを、動作していない状態に移行させること。		CYC2-15
3		動作していない状態の周期ハンドラが指定された場合には、何もしないこと。		CYC2-7

4.8　システム状態管理機能

get_tid：実行状態のタスク ID の参照

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処理	実行状態のタスクの ID 番号を参照し、tskid に返すこと。	SYS1-3

iget_tid：実行状態のタスク ID の参照

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処理	実行状態のタスクの ID 番号を参照し、tskid に返すこと。	SYS2-3
2		実行状態のタスクがない時は、tskid に TSK_NONE を返すこと。	SYS2-28

loc_cpu：CPU ロック状態への移行

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　　理	CPU ロック状態に移行すること。	SYS1-9
2		CPU ロック状態で呼び出された場合には何もしないこと。	SYS1-15

iloc_cpu：CPU ロック状態への移行

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　　理	CPU ロック状態に移行すること。	SYS2-7
2		CPU ロック状態で呼び出された場合には何もしないこと。	SYS2-11

unl_cpu：CPU ロック状態の解除

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　　理	CPU ロック解除状態に移行すること。	SYS1-19
2		CPU ロック解除状態で呼び出された場合には何もしないこと。	SYS1-23

iunl_cpu：CPU ロック状態の解除

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　　理	CPU ロック状態解除に移行すること。	SYS2-15
2		CPU ロック解除状態で呼び出された場合には何もしないこと。	SYS2-19

dis_dsp：ディスパッチの禁止

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　　理	ディスパッチ禁止状態に移行すること。	SYS1-29
2		ディスパッチ禁止状態で呼び出された場合には何もしないこと。	SYS1-35

ena_dsp：ディスパッチの許可

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　　理	ディスパッチ許可状態に移行すること。	SYS1-39
2		ディスパッチ許可状態で呼び出された場合には何もしないこと。	SYS1-43

sns_ctx：コンテキストの参照

No.	区　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　理	タスクコンテキストから呼び出された場合に FALSE を返すこと。	SYS1-25
2		非タスクコンテキストから呼び出された場合に TRUE を返すこと。	SYS2-21

sns_loc：CPU ロック状態の参照

No.	区　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　理	システムが、CPU ロック状態の場合に TRUE を返すこと。	SYS1-9
2		システムが、CPU ロック解除状態の場合に FALSE を返すこと。	SYS1-19

sns_dsp：ディスパッチ禁止状態の参照

No.	区　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	サービスコール処　理	システムが、ディスパッチ禁止状態の場合に TRUE を返すこと。	SYS1-29
2		システムが、ディスパッチ許可状態の場合に FALSE を返すこと。	SYS1-39

sns_dpn：ディスパッチ保留状態の参照

No.	区　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
―	サービスコール処　理	システムが、以下に示すディスパッチ保留状態の場合には TRUE を返すこと。	―
1		ディスパッチャよりも優先順位の高い処理が実行されている間	SYS2-30
2		CPU ロック状態の間	SYS1-11
3		ディスパッチ禁止状態の間	SYS1-31
4		システムが、ディスパッチ保留状態でない場合に FALSE を返すこと。	SYS1-5

4.9　割込み管理機能

No.	区　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	データ型	INHNO	割込みハンドラ番号	HED-74
2		INTNO	割込み番号	HED-75
3	記述形式	製品マニュアル記載の通りに割込みハンドラを記述し、期待する動作を行うこと。		UMA-30
4	拡張情報	割込みサービスルーチンを起動する際、その割込みサービスルーチンの拡張情報(exinf)をパラメータとして渡すこと。		ISR1-2
5	CPU ロック解除	割込みハンドラ内で CPU ロック解除状態にする方法が、製品マニュアルに明示されていること。		UMA-31
6	リターン	CPU ロック解除した後に、割込みハンドラから正しくリターンするための方法が製品マニュアルに明示されていること。		UMA-32

DEF_INH：割込みハンドラの定義（静的 API）

No.	区　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_RSATR	予約属性(inhatr が不正あるいは使用できない)	INH1-S2
2		E_PAR	パラメータエラー(inhno が不正)	INH1-S4
3		E_PAR	パラメータエラー(inthdr が不正)	INH1-S6
4	静的 API 処　理	inhno で指定される割込みハンドラ番号に対して、定義情報に基づいて割込みハンドラを定義できること。		INH1-2
5		inhno の具体的な意味が実装で定義されていること。		UMA-30

(注)

No.3　E_PAR エラーを検出できる inthdr がシステムに存在しない場合は、テストを省略することができる。

ATT_ISR：割込みサービスルーチンの追加（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_RSATR	予約属性(isratr が不正あるいは使用できない)	ISR1-S2
2		E_PAR	パラメータエラー(intno が不正)	ISR1-S4
3		E_PAR	パラメータエラー(isr が不正)	ISR1-S6
4	静的 API 処　　理	割込みサービスルーチンを起動する時に、パラメータとして exinf を渡すこと。		ISR1-2

(注)

No.3　E_PAR エラーを検出できる isr がシステムに存在しない場合は、テストを省略することができる。

4.11　システム構成管理機能

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	データ型	EXCNO	CPU 例外ハンドラ番号	HED-76
2	CPU 例外ハンドラの記述形式	製品マニュアル記載の通りに CPU 例外ハンドラを記述し、期待する動作を行うこと。		UMA-34
3	拡張情報	初期化ルーチンを起動する際には、その初期化ルーチンの拡張情報(exinf)をパラメータとして渡すこと。		INI1-2

DEF_EXC：CPU 例外ハンドラの定義（静的 API）

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_RSATR	予約属性(excatr が不正あるいは使用できない)	EXC1-S2
2		E_PAR	パラメータエラー(excno が不正)	EXC1-S4
3		E_PAR	パラメータエラー(exchdr が不正)	EXC1-S6
4	静的 API 処　　理	excno の具体的な意味は実装定義であるが、一般的な実装では、プロセッサで区別される例外の種類に対応する。。		UMA-36
5		すでに CPU 例外ハンドラが定義されている CPU 例外ハンドラ番号に対して、再度 CPU 例外ハンドラを定義した場合には、以前の定義を解除し、新しい定義に置き換えること。		EXC1-2

(注)

No.3　E_PAR エラーを検出できる exchdr がシステムに存在しない場合は、テストを省略することができる。

ATT_INI：初期化ルーチンの追加（静的 API）

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	静的 API 処理	追加された初期化ルーチンは、システム初期化時に、静的 API の処理の一環として実行すること。	INI1-1

5.　付属規定

5.1　μITRON4.0 仕様準拠の条件

確認項目なし

5.2　自動車制御用プロファイル

5.2.1　制約タスク

No.	区　　分	テ　ス　ト　項　目	ﾃｽﾄ手順参照
1	タスクの終了	タスクのメインルーチンからのリターンによってのみ、タスクを終了することができること。	RST-17
2	タスク属性	タスク属性に TA_RSTR を指定してタスクを生成した場合に、そのタスクは制約タスクとなる。	RST-6

5.2.2 自動車制御用プロファイルに含まれる機能

No.	区　　分	テ　ス　ト　項　目		ﾃｽﾄ手順参照
1	エラーコード	E_NOSPT	制約タスクから ext_tsk サービスコールを呼び出した	RST-4
2			制約タスクから ter_tsk サービスコールを呼び出した	RST-6
3			制約タスクから chg_pri サービスコールを呼び出した	RST-8
4			制約タスクから slp_tsk サービスコールを呼び出した	RST-10
5			制約タスクから wai_sem サービスコールを呼び出した	RST-12
6			制約タスクから wai_flg サービスコールを呼び出した	RST-14
7			制約タスクから rcv_dtq サービスコールを呼び出した	RST-16

第5章　自動車制御用プロファイルのテスト手順

第１節　プログラムによる確認の見方

1.1　テストに必要なシステム条件

自動車制御用プロファイルのテスト手順を実施する場合のシステム条件を以下に示す。

(1) ソフトウェアから何らかの CPU 例外が発生できること。
(2) ソフトウェアから何らか割込みが発生できること。
(3) 1msec 周期で isig_tim サービスコールを呼び出し、タイマを更新すること。

1.2　静的 API のテストを行う場合

以下に、静的 API のテストを行う場合のテスト手順の見方を、タスク管理機能のテスト手順その１で説明する。また、「4.1　タスク管理機能」などタイトルの前の番号は、μITRON4.0 仕様書に記載されてあるものと一致させた。

TSK1：タスク管理機能のテスト手順　その１

No	テ ス ト 手 順 内 容	テスト項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})	
S3	CRE_TSK(ERR_TASK_ID,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S4	*E_ID を検出できること*	CRE_TSK-1
S5	CRE_TSK(TASK_ID3,{ERR_TSKATR,TASK1,ITSKPRI_1,STKSZ,NULL})	
S6	*E_RSATR を検出できること*	CRE_TSK-2
S7	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_1,ERR_TASK,ITSKPRI_3,STKSZ,NULL})	
S8	*E_PAR を検出できること*	CRE_TSK-3
S9	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_1,TASK1,ERR_ITSKPRI,STKSZ,NULL})	
S10	*E_PAR を検出できること*	CRE_TSK-4
S11	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_1,TASK3,ITSKPRI_3,ERR_STKSZ,NULL})	
S12	*E_PAR を検出できること*	CRE_TSK-5
S13	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S14	*E_OBJ を検出すること*	CRE_TSK-6

(1)　No.
静的 API の記述順序を示す。

(2)　テスト手順内容
システムコンフィギュレーションファイルに記述する内容である。
「TASK_ID」等は、μITRON 検定仕様書で使用する定数一覧でデータ型とその値を定義している。
網掛けされてあるものは、実装定義でエラーの検出を省略できることを示す。ただし、これらのエラーコードの検出を省略する場合は、省略する旨が製品マニュアルに記載されていること。
No.S4 でエラーを検出するとコンフィギュレータが停止するシステムの場合は、No.S4 のテストが完了したら、No.S3 と No.S4 をエディタ等で削除し、再度コンフィギュレータを実行する。エラーを検出して停止したら当該行を削除する作業を最後まで繰り返す。

(3)　テスト項目参照
テスト手順で確認すべき事項が、テスト項目のどの部分に記載されているかを示す。

1.3 静的 API 以外のテストを行う場合

以下に、タスク管理機能のテスト手順　その２を例にテスト手順の見方を説明する。

TSK2：タスク管理機能のテスト手順　その２

No	テ ス ト 手 順 内 容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK1,ITSKPRI_2,STKSZ,NULL})		
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2:READY>**	－
1		ercd = act_tsk(ERR_TASK_ID)	
2		*MERCD(ercd) ==E_ID*	act_tsk_1
3		ercd = act_tsk(TSK_SELF)	
4		*MERCD(ercd) == E_OK*	act_tsk-7
5		act_cnt = can_act(TSK_SELF)	
6		*actcnt == 1*	act_tsk-7 can_act-4
7		act_cnt = can_act(TSK_SELF)	
8		*actcnt == 0*	can_act-2
9		act_cnt = can_act(ERR_TASK_ID)	
10		*actcnt == E_ID*	can_act-1
11		ercd = act_tsk(TASK_ID1)	
12	*exinf == EXINF_1*		act_tsk-3,5
13	ercd = act_tsk(TASK_ID2)		
14	*MERCD(ercd) == E_OK*		
15	actcnt = can_act(TASK_ID2)		

(1) No.
プログラムの実行順序を示す。必ずこの順序で実行されていることを確認しなければならない。
番号の前にＳが付加されているものは、静的 API の記述順序を示す。
番号のみの場合は、タスクやハンドラの実行順序であることを示す。したがって、タスクやハンドラは、必ず昇順で実行されることを確認すること。
「－」は、補足的説明を加えている行を示し、プログラム化を行う必要はない。

(2) テスト手順内容
静的 API でテスト環境を整え、No.1 から順次実行していく。また、No.1 で定数として使用している「ERR_TASK_ID」は、μITRON 検定仕様書で使用する定数一覧でデータ型とその値を定義している。さらに、No.5 で変数と使用している「actcnt」は、μITRON 検定仕様書で使用する変数一覧でデータ型を定義している。
*ercd == E_OK*など、*イタリック体*で記述している行では、値を比較し必ず一致しなければならないことを示す。
テスト項目と同様に、エラーコードに網掛けされてあるものは、実装定義でエラーの検出を省略できることを示す。ただし、これらのエラーコードの検出を省略する場合は、省略する旨が製品マニュアルに記載されていること。したがって、製品マニュアルに省略する旨が記載されている場合に限り、当該テストを省略できる。
No.12 で示すように、記述位置が変わっている場合は、状態が遷移しなければならないことを示す。
カーネル構成定数によって結果が異なる場合がある。
例えば、
　TMAX_ACTCNT == 1　：*actcnt == 1*
　TMAX_ACTCNT ＞ 1　：*actcnt == 2*　　などである。
定義している値によって、どちらかを選択すること。
静的 API 記述の次の行の**<TASK_ID1:DORMANT>**等、太字で記述しているものはタスクやハンドラの初期状態を示している。

(3) テスト項目参照
テスト手順で確認すべき事項が、テスト項目のどの部分に記載されているかを示す。

1.4 タスク・ハンドラの記述形式

テスト手順は、テスト項目確認のための具体的手段を記載している。テストするためのプログラムがより自然に書けるよう、C 言語を意識した記載とした。
タスクおよびハンドラは、μITRON4.0 仕様書の規定の沿って、以下の通り記述する。

(1)タスクの C 言語による記述形式

```
void task(VP_INT exinf)
{
        タスク本体
        ext_tsk( )
}
```

(2)タスクの C 言語による記述形式(return で終了する場合)

```
void task(VP_INT exinf)
{
        タスク本体
        return
}
```

(3)タスク例外処理ルーチンの C 言語による記述形式

```
void texrtn(TEXPTN texptn, VP_INT exinf)
{
        タスク例外処理ルーチン本体
}
```

(4)周期ハンドラの C 言語による記述形式

```
void cychdr(VP_INT exinf)
{
        周期ハンドラ本体
}
```

(5)割込みサービスルーチンの C 言語による記述形式

```
void isr(VP_INT exinf)
{
        割込みサービスルーチン本体
}
```

(6)初期化ルーチンの C 言語による記述形式

```
void inirtn(VP_INT exinf)
{
        初期化ルーチン本体
}
```

1.5 自動車制御用プロファイルのテスト手順で使用する定数一覧

自動車制御用プロファイルのテスト手順では、以下に示す定数を定義しテストを実施すること。

ID			起動番地		
ID	TASK_ID1	コンフィギュレータで ID の自動割付けを行う	FP	TASK1	TASK_ID1 起動番地
ID	TASK_ID1_1		FP	TASK1_1	TASK_ID1_1 起動番地
ID	TASK_ID1_2		FP	TASK1_2	TASK_ID1_2 起動番地
ID	TASK_ID2		FP	TASK2	TASK_ID2 起動番地
ID	TASK_ID2_1		FP	TASK2_1	TASK_ID2_1 起動番地
ID	TASK_ID2_2		FP	TASK2_2	TASK_ID2_2 起動番地
ID	TASK_ID3		FP	TASK3	TASK_ID3 起動番地
ID	TASK_ID4		FP	TASK4	TASK_ID4 起動番地
ID	TASK_ID5~TASK_ID16		FP	TASK5～TASK16	TASK_ID5～TASK_ID16 起動番地
ID	SEM_ID1~SEM_ID255		FP	INTHDR_1	割込みハンドラ起動番地
ID	FLG_ID1		FP	ISR_ADR_0	0
ID	FLG_ID2		FP	ISR_1	割込みサービスルーチン起動番地
ID	FLG_ID3		FP	EXCHDR_1	CPU 例外ハンドラ起動番地
ID	DTQ_ID1		FP	EXCHDR_2	CPU 例外ハンドラ起動番地
ID	DTQ_ID2		FP	INIRTN_1	初期化ルーチン起動番地
ID	DTQ_ID3		FP	INIRTN_2	初期化ルーチン起動番地
ID	DTQ_ID4		FP	CYCHDR_1	周期ハンドラ起動番地
ID	CYC_ID1		FP	CYCHDR_2	周期ハンドラ起動番地
ID	CYC_ID2		INHNO	INHNO	何らかの割込みハンドラを実装毎に定義する
拡張情報			INHNO	INHNO_1	実装で定義
VP_INT	EXINF_1	(VP_INT)0x00000001	EXCNO	EXCNO_1	実装で定義
VP_INT	EXINF_1_1	(VP_INT)0x00000011	INTNO	INTNO_1	実装で定義
VP_INT	EXINF_1_2	(VP_INT)0x00000012	優先度		
VP_INT	EXINF_2	(VP_INT)0x00000002	PRI	ITSKPRI_1~ITSKPRI_16	0x0001～0x0010
VP_INT	EXINF_2_1	(VP_INT)0x00000021	PRI	TSKPRI_3	0x0003
VP_INT	EXINF_2_2	(VP_INT)0x00000022	属性		
VP_INT	EXINF_3	(VP_INT)0x00000003	ATR	INHATR	実装毎に値を定義する
VP_INT	EXINF_4	(VP_INT)0x00000004	ATR	INHATR_1	実装毎に値を定義する
VP_INT	EXINF_CYC1	(VP_INT)0x00000001	ATR	EXCATR	実装毎に値を定義する
VP_INT	EXINF_CYC2	(VP_INT)0x00000002	ATR	INIATR	実装毎に値を定義する
初期値			サイズ		
FLGPTN	IFLGPTN_0	0x0000	SIZE	STKSZ	0x0100
FLGPTN	IFLGPTN_1	0x0001			
UINT	ISEMCNT_0	0			
UINT	ISEMCNT_1	1			
UINT	ISEMCNT_2	2			
UINT	ISEMCNT_FFFE	0xfffe			

周期ハンドラ		
RELTIM	CYCTIM_0	0
RELTIM	CYCTIM_10	10
RELTIM	CYCTIM_55	55
RELTIM	CYCPHS_0	0
RELTIM	CYCPHS_5	5
最　大　値		
UINT	MAXSEM_1	1
UINT	MAXSEM_2	2
UINT	MAXSEM_FFFF	0xffff
フラグパターン		
FLGPTN	SETPTN_1	0x0001
FLGPTN	CLRPTN_0000	0x0000
FLGPTN	CLRPTN_1	0x0001
FLGPTN	SETPTN_0001	0x0001
FLGPTN	SETPTN_0002	0x0002
FLGPTN	SETPTN_0005	0x0005
FLGPTN	SETPTN_0055	0x0055
FLGPTN	SETPTN_0500	0x0500
FLGPTN	SETPTN_0505	0x0505
FLGPTN	SETPTN_5500	0x5500
FLGPTN	SETPTN_1111	0x1111
FLGPTN	SETPTN_5555	0x5555
FLGPTN	SETPTN_8000	0x8000
FLGPTN	SETPTN_AAAA	0xAAAA
FLGPTN	WAIPTN_0000	0x0000
FLGPTN	WAIPTN_1	0x0001
FLGPTN	WAIPTN_0001	0x0001
FLGPTN	WAIPTN_0002	0x0002
FLGPTN	WAIPTN_0050	0x0050
FLGPTN	WAIPTN_0055	0x0055
FLGPTN	WAIPTN_0505	0x0505
FLGPTN	WAIPTN_8000	0x8000
FLGPTN	WAIPTN_AAAA	0xAAAA
FLGPTN	FLGPTN_0055	0x0055
FLGPTN	FLGPTN_5555	0x5555
そ　の　他		
VP_INT	DATA_00	(VP_INT)0x00000000
VP_INT	DATA_01	(VP_INT)0x00000001
VP_INT	DATA_05	(VP_INT)0x00000005
VP_INT	DATA_11	(VP_INT)0x00000011
VP_INT	DATA_22	(VP_INT)0x00000022
VP_INT	DATA_33	(VP_INT)0x00000033
VP_INT	DATA_55	(VP_INT)0x00000055
VP_INT	DATA_77	(VP_INT)0x00000077
VP_INT	DATA_88	(VP_INT)0x00000088
VP_INT	DATA_99	(VP_INT)0x00000099
VP_INT	DATA_AA	(VP_INT)0x000000AA
UW	MEMORY	共有メモリアドレス

カウンタ		
UINT	DTQCNT_0	0x00000000
UINT	DTQCNT_1	0x00000001
UINT	DTQCNT_2	0x00000002
エラー定数		
FP	ERR_TASK	E_PARエラーを返すアドレスを実装毎に定義する
ID	ERR_TASK_ID	-1
ATR	ERR_TSKATR	0xff
PRI	ERR_ITSKPRI	-1
PRI	ERR_TSKPRI	-2
SIZE	ERR_STKSZ	E_PARエラーを返す値を実装毎に定義する
INHNO	ERR_INHNO	E_PARエラーを返す値を実装毎に定義する
ATR	ERR_INHATR	E_PARエラーを返す値を実装毎に定義する
ATR	ERR_ISRATR	2
INHNO	ERR_INTNO	未サポート割込み番号
EXCNO	ERR_EXCNO	E_PARエラーを返す値を実装毎に定義する
ATR	ERR_EXCATR	E_PARエラーを返す値を実装毎に定義する
ATR	ERR_SEMATR	0x55
ID	ERR_SEM_ID	-2
UINT	MAXSEM_OVER	E_PARエラーを返す値を実装毎に定義する
ID	ERR_FLG_ID	-3
ATR	ERR_FLGATR	0xaa
FLGPTN	ERR_SETPTN	0xffffffff
FLGPTN	ERR_CLRPTN	0xffffffff
FLGPTN	ERR_WAIPTN	0xffffffff
MODE	ERR_WFMODE	0x02
ID	ERR_DTQID	-2
ATR	ERR_DTQATR	0xaa
UINT	ERR_DTQCNT	0xffffffff
ID	ERR_CYCID	0
ATR	ERR_CYCATR	0xdd
RELTIM	ERR_CYCPHS	E_PARエラーを返す値を実装毎に定義する
FP	ERR_INTHDR	E_PARエラーを返す値を実装毎に定義する
FP	ERR_EXCHDR	E_PARエラーを返す値を実装毎に定義する
FP	ERR_CYCHDR	E_PARエラーを返す値を実装毎に定義する

1.6 自動車制御用プロファイルのテスト手順で使用する変数一覧

自動車制御用プロファイルのテスト手順では、以下に示す変数を定義しテストを実施すること。

型	変数名	型	変数名	型	変数名
ER	ercd	VP_INT	exinf	BOOL	state
ER_UINT	actcnt	PRI	tskpri	ID	*p_tskid
PRI	*p_tskpri	ER_UNIT	wupcnt	FLGPTN	*p_flgptn
FLGPTN	flgptn	VP_INT	*p_data	VP_INT	data
UW	cyc_work1	UW	cyc_work2	UW	work
B	b1	H	h1	W	w1
UB	ub1	UH	uh1	UW	uw1
VB	vb1	VH	vh1	VW	vw1
VP	vp1	FP	fp1	INT	int1
UINT	uint1	BOOL	bool1	FN	fn1
ER	er1	ID	id1	ATR	atr1
STAT	stat1	MODE	mode1	PRI	pri1
SIZE	size1	TMO	tmo1	RELTIM	reltim1
VP_INT	vp_int1	ER_BOOL	er_bool1	ER_ID	er_id1
ER_UINT	er_uint1				

第２節　プログラムによる確認の手順

COM1：ITRON仕様共通規定のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照

1	#include　<kernel.h>
	#include　<kernel.h>
	B　c;
	main(　)
	{
	B　a=1,b=2;
	
	　　c = a + b;
	}
	
	※ 上記ファイルをコンパイルして、エラーが発生しないことを確認する。
		APIの構成要素-5

COM2：ITRON仕様共通規定のテスト手順　その2

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID16,{TA_HLNG|TA_ACT,EXINF_16,TASK16,ITSKPRI_16,STKSZ,NULL})	
S2	CRE_TSK(TASK_ID15,{TA_HLNG|TA_ACT,EXINF_15,TASK15,ITSKPRI_15,STKSZ,NULL})	
S3	CRE_TSK(TASK_ID14,{TA_HLNG|TA_ACT,EXINF_14,TASK14,ITSKPRI_14,STKSZ,NULL})	
S4	CRE_TSK(TASK_ID13,{TA_HLNG|TA_ACT,EXINF_13,TASK13,ITSKPRI_13,STKSZ,NULL})	
S5	CRE_TSK(TASK_ID12,{TA_HLNG|TA_ACT,EXINF_12,TASK12,ITSKPRI_12,STKSZ,NULL})	
S6	CRE_TSK(TASK_ID11,{TA_HLNG|TA_ACT,EXINF_11,TASK11,ITSKPRI_11,STKSZ,NULL})	
S7	CRE_TSK(TASK_ID10,{TA_HLNG|TA_ACT,EXINF_10,TASK10,ITSKPRI_10,STKSZ,NULL})	
S8	CRE_TSK(TASK_ID9,{TA_HLNG|TA_ACT,EXINF_9,TASK9,ITSKPRI_9,STKSZ,NULL})	
S9	CRE_TSK(TASK_ID8,{TA_HLNG|TA_ACT,EXINF_8,TASK8,ITSKPRI_8,STKSZ,NULL})	
S10	CRE_TSK(TASK_ID7,{TA_HLNG|TA_ACT,EXINF_7,TASK7,ITSKPRI_7,STKSZ,NULL})	
S11	CRE_TSK(TASK_ID6,{TA_HLNG|TA_ACT,EXINF_6,TASK6,ITSKPRI_6,STKSZ,NULL})	
S12	CRE_TSK(TASK_ID5,{TA_HLNG|TA_ACT,EXINF_5,TASK5,ITSKPRI_5,STKSZ,NULL})	
S13	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_4,STKSZ,NULL})	
S14	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})	
S15	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})	
S16	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	

―	**<TASK_ID1:READY>**	**<TASK_ID2～TASK_ID15: READY> TASK_ID2　・・・・・　TASK_ID15**	**<TASK_ID16: READY>**	―
1	ext_tsk(　)			
2 〃 15		ext_tsk(　) ・ ・ ・ ・ ・ ext_tsk(　)		
16			ext_tsk(　)	優先度-1

COM3：ITRON仕様共通規定のテスト手順　その３

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK1,ITSKPRI_2,STKSZ,NULL})		
―	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**	―
1	ercd = get_tid(ERR_ADDR)		
2	*MERCD(ercd) == E_MACV*		サービスコールの返値とエラーコード-3
3	ext_tsk(　)		

(注)

No.2　E_MACVエラーを検出できるERR_ADDR番地がシステムにない場合は、テストを省略することができる。

COM4：ITRON仕様共通規定のテスト手順　その４

No	テ　ス　ト　手　順　内　容	テスト 項目参照
1	以下の機能を持つプログラムを作成し、それをソフトウェア部品のコンフィギュレータとして、カーネルのコンフィギュレーション手順に組み込む。 (1) プログラムに渡されたファイルをそのまま表示する機能 (2) プログラムに渡されたファイルに、「CRE_TSK」で始まる行があれば、その直前に「#define TA_HLNG 0x12345678」を追加してファイルとして出力する機能	
2	以下のシステムコンフィギュレーションファイルを入力して、カーネルのコンフィギュレーション手順を実行する。 (システムコンフィギュレーションファイルの内容) INCLUDE(“<kernel.h>”); INCLUDE(“\”task1.h\””); #define　EXINF_1　0x00000001 #define　ITSKPRI_1　0x0001 #define　STKSZ　0x0100 CRE_TSK(TASK_ID1,{TA_HLNG|0x00,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL}); ただし、以下の内容のヘッダファイル「task1.h」を用意しておくこと。 (task1.hの内容) extern　void　TASK1(　);	静的APIの文法とパラメータ-1,2,4,5
3	プリプロセッサからエラーメッセージが出力されないこと。	システムコンフィギュレーションファイル-1,2
4	ソフトウェア部品のコンフィギュレータが、以下の内容を表示すること。 (コンフィギュレータによる表示内容) INCLUDE(“<kernel.h>”); INCLUDE(“\”task1.h\””); CRE_TSK(TASK_ID1,{TA_HLNG|0x00,0x00000001,TASK1,0x0001,0x0100,NULL}); なお、この他に#で始まる行が含まれていてもよい。	システムコンフィギュレーションファイル-2
5	カーネルのコンフィギュレータからエラーメッセージが出力されないこと。	システムコンフィギュレーションファイル-1,2,5 ITRON仕様共通性的API-1
6	カーネルのコンフィギュレータが、以下の内容のカーネル構成・初期化ファイルを生成し、それが正しくコンパイルできること。 (カーネル構成・初期化ファイルの内容) #include <kernel.h> #include “task1.h” タスクの初期化データ	システムコンフィギュレーションファイル-3 ITRON仕様共通性的API-1
7	カーネルのコンフィギュレータが、以下の内容のID自動割付け結果ヘッダファイル「kernel_id.h」を生成すること。 (Id自動割付け結果ヘッダファイル「kernel_id.h」の内容) #define　TASK_ID1　1	APIの構成要素-4 システムコンフィギュレーションファイル-3 静的APIの文法とパラメータ-6

COM5：ITRON仕様共通規定のテスト手順　その５

No	テ　ス　ト　手　順　内　容	テスト 項目参照
1	システムコンフィギュレーションファイルの内容 INCLUDE(“<kernel.h>”); INCLUDE(“\”task1.h\””); #define EXINF_1　0x00000001 #define ITSKPRI_1　0x0001 #define STKSZ　0x0100 ABC_DEF(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL}) ; *静的APIの名称が異なるので、カーネルのコンフィギュレータがエラーを報告すること。*	システムコンフィギュレーションファイル-4 静的APIの文法とパラメータ-7
2	システムコンフィギュレーションファイルの内容 INCLUDE(“<kernel.h>”); INCLUDE(“\”task1.h\””); #define EXINF_1　0x00000001 #define ITSKPRI_1　0x0001 #define STKSZ　0x0100 CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL}; *静的APIの文法が異なるので、カーネルのコンフィギュレータがエラーを報告すること。*	システムコンフィギュレーションファイル-4 静的APIの文法とパラメータ-7
3	システムコンフィギュレーションファイルの内容 INCLUDE(“<kernel.h>”); INCLUDE(“\”task1.h\””); #define EXINF_1　0x00000001 #define ITSKPRI_1　0x0001 #define STKSZ　0x0100 CRE_TSK(TASK_ID1,{TA_HLNG,NULL}) ; *静的APIのパラメータ数が少ないので、カーネルのコンフィギュレータがエラーを報告すること。* ただし、上記記述を許可するようにコンフィギュレータが拡張されていた場合は、テストを省略することができる。	システムコンフィギュレーションファイル-4 静的APIの文法とパラメータ-7

4	システムコンフィギュレーションファイルの内容 INCLUDE(“<kernel.h>”); INCLUDE(“\”task1.h\””); #define EXINF_1　　0x00000001 #define ITSKPRI_1　0x0001 #define STKSZ　　0x0100 CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,NULL,NULL}); *静的APIのパラメータ数が多いので、カーネルのコンフィギュレータがエラーを報告すること。* ただし、上記記述を許可するようにコンフィギュレータが拡張されていた場合は、テストを省略することができる。	システムコンフィギュレーションファイル-4 静的APIの文法とパラメータ-7

COM6：ITRON仕様共通規定のテスト手順　その６

No	テ　ス　ト　手　順　内　容	テスト項目参照
1	アプリケーションプログラムとリンクしてカーネルを用いる場合、カーネルオブジェクトのシンボルをダンプし、C言語レベルで内部識別子が“ _kernel_ ”または“ _KERNEL_ ”で始まる名称であること。	カーネルとソフトウェア内部識別子-1

COM7：ITRON仕様共通規定のテスト手順　その７

No	テ　ス　ト　手　順　内　容	テスト項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
―	**<TASK_ID1:REDAY>**	―
1	*1 == sizeof(B)*	ITRON仕様共通データ型-1
2	b1 = -1	
3	*b1 < 0*	ITRON仕様共通データ型-1
4	*1 == sizeof(UB)*	ITRON仕様共通データ型-4
5	ub1 = -1	
6	*ub1 > 0*	ITRON仕様共通データ型-4
7	*2 == sizeof(H)*	ITRON仕様共通データ型-2
8	h1 = -1	
9	*h1 < 0*	ITRON仕様共通データ型-2
10	*2 == sizeof(UH)*	ITRON仕様共通データ型-5
11	uh1 = -1	
12	*uh1 > 0*	ITRON仕様共通データ型-5
13	*4 == sizeof(W)*	ITRON仕様共通データ型-3
14	w1 = -1	
15	*w1 < 0*	ITRON仕様共通データ型-3
16	*4 == sizeof(UW)*	ITRON仕様共通データ型-6
17	uw1 = -1	
18	*uw1 > 0*	ITRON仕様共通データ型-6
19	*1 == sizeof(VB)*	ITRON仕様共通データ型-7
20	*2 == sizeof(VH)*	ITRON仕様共通データ型-8
21	*4 == sizeof(VW)*	ITRON仕様共通データ型-9
22	*sizeof(void *) == sizeof(VP)*	ITRON仕様共通データ型-10
23	*sizeof(void *) == sizeof(FP)*	ITRON仕様共通データ型-11
24	*2 ≦ sizeof(INT)*	ITRON仕様共通データ型-12
25	int1 = -1	
26	*int1 < 0*	ITRON仕様共通データ型-12
27	*2 ≦ sizeof(UINT)*	ITRON仕様共通データ型-13
28	uint1 = -1	
29	*uint1 > 0*	ITRON仕様共通データ型-13
30	bool1 = TRUE	
31	*bool1 == TRUE*	ITRON仕様共通データ型-14
32	bool1 = FALSE	
33	*bool1 == FALSE*	ITRON仕様共通データ型-14
34	*2 ≦ sizeof(FN)*	ITRON仕様共通データ型-15
35	fn1 = -1	
36	*fn1 < 0*	ITRON仕様共通データ型-15
37	*1 ≦ sizeof(ER)*	ITRON仕様共通データ型-16
38	er1 = -1	
39	*er1 < 0*	ITRON仕様共通データ型-16
40	*2 ≦ sizeof(ID)*	ITRON仕様共通データ型-17
41	id1 = -1	
42	*id1 < 0*	ITRON仕様共通データ型-17
43	*1 ≦ sizeof(ATR)*	ITRON仕様共通データ型-18
44	atr1 = -1	
45	*atr1 > 0*	ITRON仕様共通データ型-18
46	*2 ≦ sizeof(STAT)*	ITRON仕様共通データ型-19
47	stat1 = -1	
48	*stat1 > 0*	ITRON仕様共通データ型-19
49	*1 ≦ sizeof(MODE)*	ITRON仕様共通データ型-20
50	mode1 = -1	
51	*mode1 > 0*	ITRON仕様共通データ型-20

52	*2 ≦ sizeof(PRI)*	ITRON仕様共通データ型-21
53	pri1 = -1	
54	*pri1 < 0*	ITRON仕様共通データ型-21
55	*sizeof(SIZE) == sizeof(void *)*	ITRON仕様共通データ型-22
56	size1 = -1	
57	*size1 > 0*	ITRON仕様共通データ型-22
58	*2 ≦ sizeof(TMO)*	ITRON仕様共通データ型-23
59	tmo1 = -1	
60	*tmo1 < 0*	ITRON仕様共通データ型-23
61	*2 ≦ sizeof(RELTIM)*	ITRON仕様共通データ型-24
62	reltim1 = -1	
63	*reltim1 > 0*	ITRON仕様共通データ型-24
64	*2 ≦ sizeof(SYSTIM)*	ITRON仕様共通データ型-25
65	systim1 = -1	
66	*systim1 > 0*	ITRON仕様共通データ型-25
67	*sizeof(VP_INT) ≧ sizeof(INT) && sizeof(VP_INT) ≧ sizeof(VP)*	ITRON仕様共通データ型-26
68	vp_int1 = -1	
69	*vp_int1 < 0*	ITRON仕様共通データ型-26
70	*1 ≦ sizeof(ER_BOOL)*	ITRON仕様共通データ型-27
71	er_bool1 = TRUE	
72	*er_bool1 == TRUE*	ITRON仕様共通データ型-27
73	er_bool1 = FALSE	
74	*er_bool1 == FALSE*	ITRON仕様共通データ型-27
75	er_bool1 = -1	
76	*er_bool1 < 0*	ITRON仕様共通データ型-27
77	*2 ≦ sizeof(ER_ID)*	ITRON仕様共通データ型-28
78	er_id1 = -1	
79	*er_id1 < 0*	ITRON仕様共通データ型-28
80	*2 ≦ sizeof(ER_UINT)*	ITRON仕様共通データ型-29
81	er_uint1 = -1	
82	*er_uint1 < 0*	ITRON仕様共通データ型-29
83	ext_tsk( )	

COM8：ITRON仕様共通規定のテスト手順　その８

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_0,MAXSEM_FFFF})	
—	**<TASK_ID1:READY>**	—
1	ercd = pol_sem(SEM_ID1)	
2	*(ER)(B)ercd == E_TMOUT*	サービスコールの返値とエラーコード-2
3	*ercd >> 8 < 0*	サービスコールの返値とエラーコード-2
4	*MERCD(ercd) == (ER)(B)ercd*	ITRON仕様共通マクロ-1
5	*SERCD(ercd) == ercd >> 8*	ITRON仕様共通マクロ-2
6	ext_tsk( )	

SPC1：μITRON4.0仕様の概念と共通のテスト手順　その1

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})		
S4	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_0,MAXSEM_1})		
—	**<TASK_ID1:DORMANT>**	**<TASK_ID3: READY>**	—
1		ercd = act_tsk(TASK_ID1)	
2	ercd = wai_sem(SEM_ID1)		
3		*MERCD(ercd) == E_OK*	
4		ercd = sig_sem(SEM_ID1)	
5	*MERCD(ercd) == E_OK*		タスク状態-1
6	ext_tsk( )		
7		*MERCD(ercd) == E_OK*	
8		ext_tsk( )	

SPC2： μITRON4.0仕様の概念と共通のテスト手順　その2

No	テ　ス　ト　手　順　内　容					テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})					
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})					
S3	CRE_TSK(TASK_ID2_1,{TA_HLNG,EXINF_2,TASK2_1,ITSKPRI_2,STKSZ,NULL})					
S4	CRE_TSK(TASK_ID2_2,{TA_HLNG,EXINF_2,TASK2_2,ITSKPRI_2,STKSZ,NULL})					
S5	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})					
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2: DORMANT>**	**<TASK_ID2_1: DORMANT>**	**<TASK_ID2_2: DORMANT>**	**<TASK_ID3:READY>**	－
1					ercd = act_tsk(TASK_ID2)	
2		ercd = act_tsk(TASK_ID2_1)				
3		*ercd = E_OK*				
4		ercd = act_tsk(TASK_ID2_2)				
5		*ercd = E_OK*				
6		ercd = act_tsk(TASK_ID1)				
7	*exinf == EXINF_1*					タスクのスケジューリング規則-1,2
8	ercd = slp_tsk(　)					
9		*MERCD(ercd) == E_OK*				タスクのスケジューリング規則-4
10		ercd = slp_tsk(　)				
11			ercd = wup_tsk(TASK_ID2)			
12			*MERCD(ercd) == E_OK*			タスクのスケジューリング規則-3
13			ercd = wup_tsk(TASK_ID1)			
14	*MERCD(ercd) == E_OK*					
15	ext_tsk(　)					
16			*MERCD(ercd) == E_OK*			タスクのスケジューリング規則-7
17			ext_tsk(　)			
18				ext_tsk(　)		
19		*MERCD(ercd) == E_OK*				タスクのスケジューリング規則-8
20		ext_tsk(　)				
21					*MERCD(ercd) == E_OK*	
22					ext_tsk(　)	

SPC3： μITRON4.0仕様の概念と共通のテスト手順　その3

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2: READY>**	－
1		ercd = dis_dsp(　)	
2		*MERCD(ercd) == E_OK*	
3		ercd = act_tsk(TASK_ID1)	
4		*MERCD(ercd) == E_OK*	
5		ercd = ena_dsp(　)	
6	ext_tsk(　)		
7		*MERCD(ercd) == E_OK*	タスクのスケジューリング規則-5
8		ext_tsk(　)	

SPC4： μITRON4.0仕様の概念と共通のテスト手順　その4

No	テ　ス　ト　手　順　内　容						テスト 項目参照
S1	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_4,STKSZ,NULL})						
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})						
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})						
S4	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})						
S5	DEF_INH(INHNO_1,{INHATR,INTHDR_1})						
－	**<TASK_ID1:READY>**		**<TASK_ID2:READY>**	**<TASK_ID3:READY>**	**<TASK_ID4:READY>**	**<INTHDR_1:割込みハンドラ>**	－
1	1回目	2回目					
2	ext_tsk(　)						
3			<INTHDR_1の割込み発生>				
4						ercd = iact_tsk(TASK_ID1)	
5						*MERCD(ercd) == E_OK*	
6						return	
7		ext_tsk(　)					
8			ercd = slp_tsk(　)				
9				ercd = wup_tsk(TASK_ID2)			
10			*MERCD(ercd) == E_OK*				
11			ext_tsk(　)				
12				*MERCD(ercd) == E_OK*			
13				ext_tsk(　)			
14					ext_tsk(　)		タスクのスケジューリング規則-6

SPC5： μITRON4.0仕様の概念と共通のテスト手順　その5

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	DEF_EXC(EXCNO_1,{EXCATR,EXCHDR_1})		
S3	DEF_EXC(EXCNO_2,{EXCATR,EXCHDR_2})		例外処理の枠組み-2
－	**<TASK_ID1:READY>**	**<EXCHDR_1:CPU例外ハンドラ>**	－
1	EXCNO_1のCPU例外を発生させる		
2		*return*	例外処理の枠組み-1
3	ext_tsk(　)		

SPC6： μITRON4.0仕様の概念と共通のテスト手順　その6

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	DEF_INH(INHNO_1,{INHATR,INTHDR_1})				
S3	CRE_CYC(CYC_ID1,{TA_HLNG,EXINF_2,CYCHDR_1,CYCTIM_10,CYCPHS_5})				
S4	DEF_EXC(EXCNO_1,EXCATR,EXCHDR_1)				
－	**<TASK_ID1:READY>**	**<INHNO_1:割込みハンドラ>**	**<EXCHDR_1:CPU例外ハンドラ>**	**<CYC_ID1:周期ハンドラ>**	－
1	state = sns_ctx(　)				
2	*state == FALSE*				タスクコンテキストと 非タスクコンテキスト-1
3	INHNO_1の割込みを発生させる				
4		state = sns_ctx(　)			
5		*state == TRUE*			タスクコンテキストと 非タスクコンテキスト-2
6		ercd = iact_tsk(TSK_SELF)			
7		*MERCD(ercd) == E_ID*			タスクコンテキストと 非タスクコンテキスト-3
8		EXCNO_1のCPU例外発生を発生させる			
9			state = sns_ctx(　)		
10			*state == TRUE*		タスクコンテキストと 非タスクコンテキスト-5
11			return		
12		return			
13	ercd = sta_cyc(CYC_ID1)				
14	*MERCD(ercd) == E_OK*				
15	20msecのソフトウェアディレー				
16				state = sns_ctx(　)	
17				*state == TRUE*	タスクコンテキストと 非タスクコンテキスト-2
18				return	
19	ercd = stp_cyc(CYC_ID1)				
20	*MERCD(ercd) == E_OK*				
21	ext_tsk(　)				

SPC7： μITRON4.0仕様の概念と共通のテスト手順　その7

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	DEF_INH(INHNO_1,{INHATR,INTHDR_1})			
S3	CRE_CYC(CYC_ID1,{TA_HLNG,EXINF_2,CYCHDR_1,CYCTIM_10,CYCPHS_5})			
－	**<TASK_ID1:READY>**	**<INHNO_1:割込みハンドラ>**	**<CYC_ID1:周期起動ハンドラ>**	－
1	state = sns_loc(　)			
2	*state == FALSE*			CPUロック状態-14
3	ercd = loc_cpu(　)			
4	*MERCD(ercd) == E_OK*			
5	state = sns_loc(　)			
6	*state == TRUE*			
7	INHNO_1の割込みを発生させる			
8	*CPUロック状態なので､割込みハンドラは起動されない*			CPUロック状態-1
9	ercd = loc_cpu(　)			
10	*MERCD(ercd) == E_OK*			CPUロック状態-2
11	state = sns_ctx(　)			
12	*state == FALSE*			CPUロック状態-6
13	state = sns_loc(　)			
14	*state == TRUE*			CPUロック状態-7
15	state = sns_dsp(　)			
16	*state == FALSE*			CPUロック状態-8
17	state = sns_dpn(　)			
18	*state == TRUE*			CPUロック状態-9
19	ercd = unl_cpu(　)			
20		ercd = iloc_cpu(　)		
21		*MERCD(ercd) == E_OK*		
22		ercd = iloc_cpu(　)		
23		*MERCD(ercd) == E_OK*		CPUロック状態-3

24		state = sns_loc( )		
25		*state == TRUE*		
26		ercd = iunl_cpu( )		
27		*MERCD(ercd) == E_OK*		CPUロック状態-5
28		state = sns_loc( )		
29		*state == FALSE*		
30		return		
31	*MERCD(ercd) == E_OK*			CPUロック状態-4
32	ercd = sta_cyc(CYC_ID1)			
33	*MERCD(ercd) == E_OK*			
34	ercd = loc_cpu( )			
35	*MERCD(ercd) == E_OK*			
36	50msec以上のソフトウェアディレーを入れる			
37	*CPUロック状態なので周期ハンドラは起動されない*			CPUロック状態-1
38	ercd = unl_cpu( )			
39	*MERCD(ercd) == E_OK*			
40	50msecのソフトウェアディレーを入れる			
41			state = sns_loc( )	
42			*state == FALSE*	CPUロック状態-12
43			return	
44	ercd = stp_cyc(CYC_ID1)			
45	*MERCD(ercd) == E_OK*			
46	ext_tsk( )			

（注）

S2　DEF_INHをサポートしていないシステムでは、以下のように置換し、SPC7のテストを実施すること。

S2	ATT_ISR({TA_HLNG,EXINF_1,INTNO_1,ISR_1})

SPC8：μITRON4.0仕様の概念と共通のテスト手順　その8

No	テ　ス　ト　手　順　内　容		テスト項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	ATT_ISR({TA_HLNG,EXINF_1,INTNO_1,ISR_1})		
－	**<TASK_ID1:READY>**	**<INTNO_1:割込みサービスルーチン>**	－
1	INTNO_1の割込み番号に対応する割込みを発生させる		
2		state = sns_loc( )	
3		*MERCD(ercd) == FALSE*	CPUロック状態-12
4		return	
5	ext_tsk( )		

（注）

ATT_ISRをシステムがサポートしていない場合は、SPC8のテストを省略することができる。

SPC9：μITRON4.0仕様の概念と共通のテスト手順　その9

No	テ　ス　ト　手　順　内　容			テスト項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	DEF_EXC(EXCNO_1,{EXCATR,EXCHDR_1})			
－	**<TASK_ID1:READY>**	**<EXCNO_1:CPU例外ハンドラ>**		－
		1回目	2回目	
1	ercd = loc_cpu( )			
2	*MERCD(ercd) == E_OK*			
3	EXCNO_1のCPU例外を発生させる			
4		state = sns_loc( )		
5		*state == TRUE*		CPUロック状態-13
6		return		
7	state = sns_loc( )			
8	*state == TRUE*			CPUロック状態-13
9	ercd = unl_cpu( )			
10	*MERCD(ercd) == E_OK*			
11	EXCNO_1のCPU例外を発生させる			
12			state = sns_loc( )	
13			*state == FALSE*	CPUロック状態-13
14			return	
15	state = sns_loc( )			
16	*state == FALSE*			CPUロック状態-13
17	ext_tsk( )			

SPC10： μITRON4.0仕様の概念と共通のテスト手順　その10

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})	
S3	DEF_INH(INHNO_1,{INHATR,INTHDR_1})	
S4	CRE_CYC(CYC_ID1,{TA_HLNG,EXINF_2,CYCHDR_1,CYCTIM_10,CYCPHS_5})	
S5	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_1,MAXSEM_2})	

－	**<TASK_ID1:DORMANT>**	**<TASK_ID2:READY>**	**<INHNO_1:割込みハンドラ>**		**<CYC_ID1:周期起動ハンドラ>**		－
			1回目	2回目	1回目	2回目	
1		state = sns_dsp(　)					
2		*state == FALSE*					ディスパッチ禁止状態-7
3		ercd = dis_dsp(　)					
4		*MERCD(ercd) == E_OK*					
5		ercd = act_tsk(TASK_ID1)					
6		*MERCD(ercd) == E_OK*					ディスパッチ禁止状態-1
7		ercd = pol_sem(SEM_ID1)					
8		*MERCD(ercd) == E_OK*					ディスパッチ禁止状態-8
9		ercd = get_tid(&tskid)					
10		*MERCD(ercd) == E_OK*					ディスパッチ禁止状態-2
11		*tskid == TASK_ID2*					ディスパッチ禁止状態-2
12		ercd = loc_cpu(　)					
13		*MERCD(ercd) == E_OK*					
14		state = sns_dsp(　)					
15		*state == TRUE*					ディスパッチ禁止状態-10
16		ercd = unl_cpu(　)					
17		*MERCD(ercd) == E_OK*					
18		state = sns_dsp(　)					
19		*state == TRUE*					ディスパッチ禁止状態-9
20		INTNO_1の割込み発生					
21			return				
22		state = sns_dsp(　)					
23		*state == TRUE*					ディスパッチ禁止状態-4
24		ercd = sta_cyc(CYC_ID1)					
25		*MERCD(ercd) == E_OK*					
26		ソフトウェアで20msec以上のディレーを作成					
27					ercd = iact_tsk(TASK_ID1)		
28					*MERCD(ercd) == E_OK*		ディスパッチ禁止状態-3
29					return		
30		ercd = stp_cyc(CYC_ID1)					
31		*MERCD(ercd) == E_OK*					
32		state = sns_dsp(　)					
33		*state == TRUE*					ディスパッチ禁止状態-6
34		ercd = ena_dsp(　)					
35	actcnt = can_act(TSK_SELF)						
36	*actcnt == 1*						
37	ext_tsk(　)						
38		*MERCD(ercd) == E_OK*					
39		INTNO_1の割込み発生					
40				return			
41		state = sns_dsp(　)					
42		*state = FALSE*					ディスパッチ禁止状態-4
43		ercd = sta_cyc(CYC_ID1)					
44		*MERCD(ercd) == E_OK*				return	
45		20msecのソフトウェアディレー					
46		state = sns_dsp(　)					
47		*state == FALSE*					ディスパッチ禁止状態-6
48		ercd = loc_cpu(　)					
49		*MERCD(ercd) == E_OK*					
50		state = sns_dsp(　)					
51		*state == FALSE*					ディスパッチ禁止状態-10
52		ercd = unl_cpu(　)					
53		*MERCD(ercd) == E_OK*					
54		ext_tsk(　)					

(注)

S3　DEF_INHをサポートしていないシステムでは、以下のように置換し、SPC10のテストを実施すること。

S3	ATT_ISR({TA_HLNG,EXINF_1,INTNO_1,ISR_1})	

なお、DEF_INHとATT_ISRの両方をサポートしているシステムでは、SPC10記載のテストとATT_ISRに置換したテストの２通りを実施すること。

SPC11：μITRON4.0仕様の概念と共通のテスト手順　その11

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	DEF_INH(INHNO_1,{INHATR,INTHDR_1})			
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2:READY>**	**<INHNO_1:割込みハンドラ>**	－
1		INHNO_1の割込み発生		
2			ercd = iact_tsk(TASK_ID1)	
3			*MERCD(ercd) == E_OK*	
4			ercd = iget_tid(&tskid)	
5			*MERCD(ercd) == E_OK*	
6			*tskid == TASK_ID2*	ディスパッチ保留状態の間のタスク状態-1
7			return	
8	ext_tsk(　)			
9		ext_tsk(　)		

SPC12：μITRON4.0仕様の概念と共通のテスト手順　その12

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	DEF_INH(INHNO_1,{INHATR,INTHDR_1})		
－	**<TASK_ID1:READY>**	**<INHNO_1:割込みハンドラ>**	－
1	state = sns_ctx(　)		
2	*state == FALSE*		非タスクコンテキストから呼び出せるサービスコール-12
3	state = sns_loc(　)		
4	*state == FALSE*		非タスクコンテキストから呼び出せるサービスコール-13
5	state = sns_dsp(　)		
6	*state == FALSE*		非タスクコンテキストから呼び出せるサービスコール-14
7	state = sns_dpn(　)		
8	*state == FALSE*		非タスクコンテキストから呼び出せるサービスコール-15
9	INHNO_1の割込み発生		
10		state = sns_ctx(　)	
11		*state == TRUE*	非タスクコンテキストから呼び出せるサービスコール-12
12		state = sns_loc(　)	
13		*state == FALSE*	非タスクコンテキストから呼び出せるサービスコール-13
14		state = sns_dsp(　)	
15		*state == FALSE*	非タスクコンテキストから呼び出せるサービスコール-14
16		state = sns_dpn(　)	
17		*state == TRUE*	非タスクコンテキストから呼び出せるサービスコール-15
18		ercd = isig_tim(　)	
19		*MERCD(ercd) == E_OK*	非タスクコンテキストから呼び出せるサービスコール-8
20		return	
21	ext_tsk(　)		

（注）
　No.17　isig_timサービスコールをサポートしていないシステムでは、テストを省略できる。

SPC13：μITRON4.0仕様の概念と共通のテスト手順　その13

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_1,STKSZ,NULL})			
S3	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_1,MAXSEM_FFFF})			
－	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**	**<INHNO_1:割込みハンドラ>**	－
1	ercd = wai_sem(SEM_ID1)			
2	*MERCD(ercd) == E_OK*			
3	INHNO_1の割込み発生			
4			ercd = iact_tsk(TASK_ID2)	
5			*MERCD(ercd) == E_OK*	
6			ercd = isig_sem(SEM_ID1)	
7			*MERCD(ercd) == E_OK*	
8			ercd = isig_sem(SEM_ID1)	
9			*MERCD(ercd) == E_OK*	
10			return	
11	ercd = wai_sem(SEM_ID1)			
12	*MERCD(ercd) == E_OK*			
13	ext_tsk(　)			
14		ercd = wai_sem(SEM_ID1)		
15		*MERCD(ercd) == E_OK*		ｻｰﾋﾞｽｺｰﾙの遅延実行-1
16		ext_tsk(　)		

SPC14：μITRON4.0仕様の概念と共通のテスト手順　その14

No	テ　ス　ト　手　順　内　容					テスト 項目参照
S1	ATT_INI({INIATR,EXINF_1,INIRTN_1})					
S2	ATT_INI({INIATR,EXINF_2,INIRTN_2})					
S3	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})					
S4	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_1,STKSZ,NULL})					
S5	DEF_INH(INHNO_1,{INHATR,INTHDR_1})					
－	**<INIRTN_1:初期化ルーチン>** カーネルの動作開始前に実行	**<INIRTN_2:初期化ルーチン>** カーネルの動作開始前に実行	**<INHNO_1:割込みハンドラ>**	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**	－
1	INHNO_1の割込み発生					
2	return					
3		return				ｼｽﾃﾑ初期化手順-2,7
4			ercd = iact_tsk(TASK_ID2)			ｼｽﾃﾑ初期化手順-8
5			*MERCD(ercd) == E_OK*			
6			return			
7				ext_tsk(　)		
8					ext_tsk(　)	ｼｽﾃﾑ初期化手順-1

SPC15：μITRON4.0仕様の概念と共通のテスト手順　その15

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_1,MAXSEM_2})	
S3 ・ ・ S255	CRE_SEM(SEM_ID2,{TA_TFIFO,ISEMCNT_1,MAXSEM_2}) ・ ・ CRE_SEM(SEM_ID254,{TA_TFIFO,ISEMCNT_1,MAXSEM_2})	
S256	CRE_SEM(SEM_ID255,{TA_TFIFO,ISEMCNT_1,MAXSEM_2})	ｵﾌﾞｼﾞｪｸﾄのID番号とｵﾌﾞｼﾞｪｸﾄ番号-1
－	**<TASK_ID1:READY>**	－
1	ercd = pol_sem(SEM_ID1)	
2	*MERCD(ercd) == E_OK*	
3	ercd = pol_sem(SEM_ID255)	
4	*MERCD(ercd) == E_OK*	ｵﾌﾞｼﾞｪｸﾄの登録とその削除-1
5	ext_tsk(　)	

TSK1：タスク管理機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})	
S3	CRE_TSK(ERR_TASK_ID,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S4	*E_IDを検出できること*	CRE_TSK-1
S5	CRE_TSK(TASK_ID3,{ERR_TSKATR,TASK1,ITSKPRI_1,STKSZ,NULL})	
S6	*E_RSATRを検出できること*	CRE_TSK-2
S7	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_1,ERR_TASK,ITSKPRI_3,STKSZ,NULL})	
S8	*E_PARを検出できること*	CRE_TSK-3
S9	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_1,TASK1,ERR_ITSKPRI,STKSZ,NULL})	
S10	*E_PARを検出できること*	CRE_TSK-4
S11	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_1,TASK3,ITSKPRI_3,ERR_STKSZ,NULL})	
S12	*E_PARを検出できること*	CRE_TSK-5
S13	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S14	*E_OBJを検出すること*	CRE_TSK-6

—	<TASK_ID1:DORMANT>	<TASK_ID2: READY>		—
		1回目	2回目	
1		*exinf == EXINF_2*		CRE_TSK-9,11
2		actcnt = can_act(TSK_SELF)		
3		*actcnt == 0*		タスク管理機能-1
4		ercd = act_tsk(TASK_ID1)		
5	ercd = chg_pri(TASK_ID2,TSKPRI_3)			CRE_TSK-8
6	*MERCD(ercd) == E_OK*			
7	ercd = wup_tsk(TASK_ID2)			
8	*MERCD(ercd) == E_OK*			
9	ercd = ter_tsk(TASK_ID2)			
10	*MERCD(ercd) == E_OK*			
11	ercd = act_tsk(TASK_ID2)			
12	*MERCD(ercd) == E_OK*			
13	ext_tsk( )			
14			ercd = get_pri(TSK_SELF,p_tskpri)	
15			*MERCD(ercd) == E_OK*	
16			*tskpri == ITSKPRI_2*	タスク管理機能-2
17			wupcnt = can_wup(TSK_SELF)	
18			*wupcnt == 0*	タスク管理機能-3
19			ext_tsk( )	

(注)

No.S8　E_PARエラーを検出できるERR_TASKがシステムに存在しない場合は、テストを省略することができる。

No.S12　E_PARエラーを検出できるERR_STKSZがシステムに存在しない場合は、テストを省略することができる。

TSK2：タスク管理機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
—	**<TASK_ID1:DORMANT>**	**<TASK_ID2:READY>**	—
1		ercd = act_tsk(ERR_TASK_ID)	
2		*MERCD(ercd) ==E_ID*	act_tsk-1
3		ercd = act_tsk(TSK_SELF)	
4		*MERCD(ercd) == E_OK*	act_tsk-7
5		actcnt = can_act(TSK_SELF)	
6		*actcnt == 1*	act_tsk-7 can_act-4
7		actcnt = can_act(TSK_SELF)	
8		*actcnt == 0*	can_act-2
9		actcnt = can_act(ERR_TASK_ID)	
10		*actcnt == E_ID*	can_act-1
11		ercd = act_tsk(TASK_ID1)	
12	*exinf == EXINF_1*		act_tsk-3,5
13	ercd = act_tsk(TASK_ID2)		
14	*MERCD(ercd) == E_OK*		
15	actcnt = can_act(TASK_ID2)		
16	*actcnt == 1*		can_act-3
17	ercd = act_tsk(TASK_ID2)		
18	*MERCD(ercd) == E_OK*		
19	ercd = act_tsk(TASK_ID2)		
20	TMAX_ACTCNT = 1　：*MERCD(ercd) == E_QOVR* TMAX_ACTCNT > 1　：*MERCD(ercd) == E_OK*		act_tsk-2
21	ext_tsk( )		
22		*MERCD(ercd) == E_OK*	
23		actcnt = can_act(TSK_SELF)	
24		TMAX_ACTCNT = 1　：*actcnt == 1* TMAX_ACTCNT > 1　：*actcnt == 2*	act_tsk-6
25		ext_tsk( )	

TSK3：タスク管理機能のテスト手順　その3

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	DEF_INH(INHNO,{INHATR,INTHDR_1})			
—	**<INTHDR_1:割込みハンドラ>**	**<TASK_ID1:DORMANT>**	**<TASK_ID2:READY>**	—
1			INTHDR_1の割込みを発生させる	
2	ercd = iact_tsk(ERR_TASK_ID)			
3	*MERCD(ercd) == E_ID*			iact_tsk-1
4	ercd = iact_tsk(TSK_SELF)			
5	*MERCD(ercd) == E_ID*			iact_tsk-7
6	ercd = iact_tsk(TASK_ID2)			
7	*MERCD(ercd) == E_OK*			非タスクコンテキストから呼び出せるサービスコール-1
8	ercd = iact_tsk(TASK_ID2)			
9	TMAX_ACTCNT = 1　：*MERCD(ercd) == E_QOVR* TMAX_ACTCNT > 1　：*MERCD(ercd) == E_OK*			iact_tsk-2

10	ercd = iact_tsk(TASK_ID1)			
11	*MERCD(ercd) == E_OK*			処理の優先順位とサービスコールの不可分性-1
12	return			
13		*exinf == EXINF_1*		iact_tsk-3,5
14		ext_tsk( )		
15			actcnt == can_act(TSK_SELF)	
16			TMAX_ACTCNT = 1 ：*actcnt == 1* TMAX_ACTCNT > 1 ：*actcnt == 2*	iact_tsk-6
17			ext_tsk( )	

(注)

No.S3　DEF_INHサービスコールをサポートしていないシステムでは、ATT_ISRサービスコールに置換に、テストを実施すること。

N0.9　E_QOVRエラーの検出を省略することが製品マニュアルに記載されている場合：MERCD(ercd) == E_OK または E_QOVR

TSK4：タスク管理機能のテスト手順　その4

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL}			
―	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**		―
		1回目	2回目	
1	ercd = act_tsk(TASK_ID2)			
2	*MERCD(ercd) == E_OK*			
3	ercd = act_tsk(TASK_ID2)			
4	*MERCD(ercd) == E_OK*			
5	ext_tsk( )			
6		ercd = get_pri(TASK_ID1,p_tskpri)		
7		*MERCD(ercd) == E_OBJ*		ext_tsk-1
8		ext_tsk( )		
9			*exinf == EXINF_2*	ext_tsk-4,5
10			actcnt = can_act(TSK_SELF)	
11			*actcnt == 0*	ext_tsk-2
12			ext_tsk( )	

TSK5：タスク管理機能のテスト手順　その5

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
―	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**		―
		1回目	2回目	
1	ercd = act_tsk(TASK_ID2)			
2	*MERCD(ercd) == E_OK*			
3	ercd = act_tsk(TASK_ID2)			
4	*MERCD(ercd) == E_OK*			
5	return			
6		ercd = get_pri(TASK_ID1,p_tskpri)		
7		*MERCD(ercd) == E_OBJ*		return-1
8		return		
9			*exinf == EXINF_2*	return-4
10			actcnt = can_act(TSK_SELF)	
11			*actcnt == 0*	return-2
12			return	

TSK6：タスク管理機能のテスト手順　その6

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1_1,{TA_HLNG|TA_ACT,EXINF_1_1,TASK1_1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID1_2,{TA_HLNG,EXINF_1_2,TASK1_2,ITSKPRI_1,STKSZ,NULL})			
S3	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
―	**<TASK_ID1_1:READY>**	**<TASK_ID1_2:DORMANT>**	**<TASK_ID2:READY>**	―
1	ercd = ter_tsk(ERR_TASK_ID)		このタスクへの状態遷移はない。	
2	*MERCD(ercd) == E_ID*			ter_tsk-1
3	ercd = ter_tsk(TASK_ID1_1)			
4	*MERCD(ercd) == E_ILUSE*			ter_tsk-2
5	ercd = ter_tsk(TASK_ID1_2)			
6	*MERCD(ercd) == E_OBJ*			ter_tsk-3
7	ercd = ter_tsk(TASK_ID2)			
8	*MERCD(ercd) == E_OK*			
9	ercd = get_pri(TASK_ID2,p_tskpri)			
10	*MERCD(ercd) == E_OBJ*			ter_tsk-4
11	ercd = act_tsk(TASK_ID1_2)			
12	*MERCD(ercd) == E_OK*			
13	ercd = act_tsk(TASK_ID1_2)			
14	*MERCD(ercd) == E_OK*			
15	ercd = ter_tsk(TASK_ID1_2)			
16	*MERCD(ercd) == E_OK*			

17	ext_tsk( )			
18		*exinf == EXINF_1_2*		ter_tsk-6,8
19		ercd = can_act(TSK_SELF)		
20		*actcnt == 0*		ter_tsk-5
21		ext_tsk( )		

TSK7：タスク管理機能のテスト手順　その7

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})			
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<TASK_ID3:DORMANT>**	－
1	ercd = chg_pri(ERR_TASK_ID,TSKPRI_1)			
2	*MERCD(ercd) == E_ID*			chg_pri-1
3	ercd = chg_pri(TASK_ID2,ERR_TSKPRI)			
4	*MERCD(ercd) == E_PAR*			chg_pri-2
5	ercd = chg_pri(TASK_ID3,TSKPRI_1)			
6	*MERCD(ercd) == E_OBJ*			chg_pri-3
7	ercd = get_pri(ERR_TASK_ID,p_tskpri)			
8	*MERCD(ercd) == E_ID*			get_pri-1
9	ercd = get_pri(TASK_ID3,p_tskpri)			
10	*MERCD(ercd) == E_OBJ*			get_pri-2
11	ercd = get_pri(TASK_ID2,p_tskpri)			
12	*MERCD(ercd) == E_OK*			
13	*tskpri == ITSKPRI_2*			get_pri-3
14	ercd = get_pri(TSK_SELF,p_tskpri)			
15	*MERCD(ercd) == E_OK*			
16	*tskpri == ITSKPRI_1*			get_pri-4
17	ercd = act_tsk(TASK_ID3)			
18	*MERCD(ercd) == E_OK*			
19	ercd = chg_pri(TASK_ID2,TSKPRI_3)			
20	*MERCD(ercd) == E_OK*			
21	ercd = chg_pri(TSK_SELF,TSKPRI_3)			
22			ercd = get_pri(TASK_ID2,p_tskpri)	chg_pri-7
23			*MERCD(ercd) == E_OK*	
24			*tskpri == TSKPRI_3*	chg_pri-4
25			ercd = chg_pri(TASK_ID2,TPRI_INI)	
26		ercd = get_pri(TSK_SELF,p_tskpri)		
27		*MERCD(ercd) == E_OK*		
28		*tskpri == ITSKPRI_2*		
29		ext_tsk( )		
30			*MERCD(ercd) == E_OK*	
31			ext_tsk( )	
32	*MERCD(ercd) == E_OK*			chg_pri-5,8
33	ercd = chg_pri(TSK_SELF,TPRI_INI)			
34	*MERCD(ercd) == E_OK*			
35	ercd = get_pri(TSK_SELF,p_tskpri)			
36	*MERCD(ercd) == E_OK*			
37	*tskpri == ITSKPRI_1*			chg_pri-6
38	ext_tsk( )			

SYN1：タスク付属同期機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
－	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**	－
1	ercd = wup_tsk(ERR_TASK_ID)		
2	*MERCD(ercd) == E_ID*		wup_tsk-1
3	ercd = wup_tsk(TASK_ID2)		
4	*ercd = E_OBJ*		wup_tsk-2
5	ercd = wup_tsk(TSK_SELF)		
6	*MERCD(ercd) == E_OK*		
7	ercd = slp_tsk( )		
8	*MERCD(ercd) == E_OK*		slp_tsk-4 wup_tsk-7
9	ercd = act_tsk(TASK_ID2)		
10	*MERCD(ercd) == E_OK*		
11	ercd = wup_tsk(TASK_ID2)		
12	*MERCD(ercd) == E_OK*		
13	ercd = wup_tsk(TASK_ID2)		
14	TMAX_WUPCNT = 1　　: *MERCD(ercd) == E_QOVR* TMAX_WUPCNT > 1　　: *MERCD(ercd) == E_OK*		wup_tsk-3
15	ercd = slp_tsk( )		
16		ercd = rel_wai(TASK_ID1)	
17	*MERCD(ercd) == E_RLWAI*		slp_tsk-1
18	ercd = slp_tsk( )		
19		*MERCD(ercd) == E_OK*	

20		ercd = slp_tsk( )	
21		*MERCD(ercd) == E_OK*	wup_tsk-6
22		wupcnt = can_wup(TSK_SELF)	
23		TMAX_WUPCNT = 1 ： *wupcnt == 0* TMAX_WUPCNT > 1 ： *wupcnt == 1*	slp_tsk-3
24		ercd = wup_tsk(TASK_ID1)	
25	*MERCD(ercd) == E_OK*		slp_tsk-2 wup_tsk-4,5
26	ext_tsk( )		
27		*MERCD(ercd) == E_OK*	
28		ext_tsk( )	

SYN2：タスク付属同期機能のテスト手順　その2

No	テ ス ト 手 順 内 容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S4	DEF_INH(INHNO,{INHATR,INTHDR_1})				
－	**<INTHDR_1:割込みハンドラ>**	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<TASK_ID3:DORMANT>**	－
1		ercd = slp_tsk( )			
2			INTHDR_1の割込み発生		
3	ercd = iwup_tsk(TSK_SELF)				
4	*MERCD(ercd) == E_ID*				iwup_tsk-7
5	ercd = iwup_tsk(ERR_TASK_ID)				
6	*MERCD(ercd) == E_ID*				iwup_tsk-1
7	ercd = iwup_tsk(TASK_ID3)				
8	*MERCD(ercd) == E_OBJ* （注）				iwup_tsk-2
9	ercd = iwup_tsk(TASK_ID1)				
10	*MERCD(ercd) == E_OK*				非タスクコンテキストから呼び出せるサービスコール-2
11	ercd = iwup_tsk(TASK_ID2)				
12	*MERCD(ercd) == E_OK*				
13	ercd = iwup_tsk(TASK_ID2)				
14	TMAX_WUPCNT == 1： *MERCD(ercd) == E_QOVR*（注） TMAX_WUPCNT > 1： *MERCD(ercd) == E_OK*				iwup_tsk-3
15	return				
16		*MERCD(ercd) == E_OK*			iwup_tsk-4,5
17		ext_tsk( )			
18			ercd = slp_tsk( )		
19			*MERCD(ercd) == E_OK*		iwup_tsk-6
20			ext_tsk( )		
21				このタスクへの状態遷移はない	

(注)

No.6　E_OBJエラーの検出を省略することが製品マニュアルに記載されている場合　：MERCD(ercd) == E_OK

No.12　E_QOVRエラーの検出を省略することが製品マニュアルに記載されている場合：MERCD(ercd) == E_OK

SYN3：タスク付属同期機能のテスト手順　その3

No	テ ス ト 手 順 内 容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
－	**<TASK_ID1:READY>**	**<TASK_ID2:DORMANT>**	－
1	ercd = can_wup(ERR_TASK_ID)		
2	*MERCD(ercd) == E_ID*		can_wup-1
3	ercd = can_wup(TASK_ID2)		
4	*MERCD(ercd) == E_OBJ*		can_wup-2
5	ercd = wup_tsk(TSK_SELF)		
6	*MERCD(ercd) == E_OK*		
7	wupcnt = can_wup(TSK_SELF)		
8	*wupcnt == 1*		can_wup-5
9	ercd = act_tsk(TASK_ID2)		
10	*MERCD(ercd) == E_OK*		
11	ercd = wup_tsk(TASK_ID2)		
12	*MERCD(ercd) == E_OK*		
13	wupcnt = can_wup(TASK_ID2)		
14	*wupcnt == 1*		can_wup-4
15	wupcnt = can_wup(TASK_ID2)		
16	*wupcnt == 0*		can_wup-3
17	ext_tsk( )		
18		ext_tsk( )	

SYN4：タスク付属同期機能のテスト手順　その4

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
—	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	—
1	ercd = rel_wai(ERR_TASK_ID)		
2	*MERCD(ercd) == E_ID*		rel_wai-1
3	ercd = rel_wai(TASK_ID2)		
4	*MERCD(ercd) == E_OBJ*		rel_wai-2
5	ercd = slp_tsk(　)		
6		ercd = rel_wai(TASK_ID1)	
7	*MERCD(ercd) == E_RLWAI*		rel_wai-3,4
8	ext_tsk(　)		
9		*MERCD(ercd) == E_OK*	
10		ext_tsk(　)	

SYN5：タスク付属同期機能のテスト手順　その5

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S4	DEF_INH(INHNO,{INHATR,INTHDR_1})　【何らかの割込みを発生できるシステムとすること】				
—	**<INTHDR_1:割込みハンドラ>**	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<TASK_ID3:READY>**	—
1		ercd = slp_tsk(　)			
2			ercd = slp_tsk(　)		
3				INTHDR_1の割込みを発生	
4	ercd = irel_wai(ERR_TASK_ID)				
5	*MERCD(ercd) == E_ID*				irel_wai-1
6	ercd = irel_wai(TASK_ID3)				
7	*MERCD(ercd) == E_OBJ*(注)				irel_wai-2
8	ercd = irel_wai(TASK_ID1)				
9	*MERCD(ercd) == E_OK*				非タスクコンテキストから呼び出せるサービスコール-3
10	ercd = irel_wai(TASK_ID2)				
11	*MERCD(ercd) == E_OK*				
12	return				
13		*MERCD(ercd) == E_RLWAI*			irel_wai-3,4
14		ext_tsk(　)			
15			*MERCD(ercd) == E_RLWAI*		
16			ext_tsk(　)		
17				ext_tsk(　)	

(注)

No.7　E_OBJエラーの検出を省略することが製品マニュアルに記載されている場合：MERCD(ercd) == E_OK

SEM1：セマフォ機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_SEM(ERR_SEM_ID,{TA_TFIFO,ISEMCNT_1,MAXSEM_FFFF})	
S2	*E_IDを検出すること*	CRE_SEM-1
S3	CRE_SEM(SEM_ID3,{ERR_SEMATR,ISEMCNT_1,MAXSEM_FFFF})	
S4	*E_RSATRを検出すること*	CRE_SEM-2
S5	CRE_SEM(SEM_ID3,{TA_TPRI,ISEMCNT_2,MAXSEM_1})	
S6	*E_PARを検出すること*	CRE_SEM-3
S7	CRE_SEM(SEM_ID3,{TA_TPRI,ISEMCNT_2,MAXSEM_OVER})	
S8	*E_PARを検出すること*	CRE_SEM-4
S9	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_1,MAXSEM_FFFF})	
S10	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_1,MAXSEM_FFFF})	
S11	*E_OBJを検出すること*	CRE_SEM-5

(注)

No.S8　　E_PARエラーを検出できるMAXSEM_OVERがシステムに存在しない場合は、テストを省略することができる。

SEM2：セマフォ機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S4	CRE_TSK(TASK_ID4,{TA_HLNG|TA_ACT,EXINF_4,TASK4,ITSKPRI_4,STKSZ,NULL})				
S5	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_2,MAXSEM_2})				
S6	CRE_SEM(SEM_ID2,{TA_TFIFO,ISEMCNT_FFFE,MAXSEM_FFFF})				
—	**<TASK_ID1:DORMANT>**	**<TASK_ID2:DORMANT>**	**<TASK_ID3:DORMANT>**	**<TASK_ID4:READY>**	—
1				ercd = sig_sem(ERR_SEM_ID)	
2				*MERCD(ercd) == E_ID*	sig_sem-1
3				ercd = wai_sem(ERR_SEM_ID)	
4				*MERCD(ercd) == E_ID*	wai_sem-1
5				ercd = pol_sem(ERR_SEM_ID)	

6				*MERCD(ercd) == E_ID*	pol_sem-1
7				ercd = sig_sem(SEM_ID1)	
8				*MERCD(ercd) == E_QOVR*	CRE_SEM-8 sig_sem-2
9				ercd = wai_sem(SEM_ID1)	
10				*MERCD(ercd) == E_OK*	wai_sem-3,4
11				ercd = pol_sem(SEM_ID1)	
12				*MERCD(ercd) == E_OK*	pol_sem-3,4
13				ercd = pol_sem(SEM_ID1)	
14				*MERCD(ercd) == E_TMOUT*	CRE_SEM-7 pol_sem-2
15				ercd = sig_sem(SEM_ID1)	
16				*MERCD(ercd) == E_OK*	sig_sem-6
17				ercd = pol_sem(SEM_ID1)	
18				*MERCD(ercd) == E_OK*	
19				ercd = act_tsk(TASK_ID3)	
20			ercd = wai_sem(SEM_ID1)		
21				*MERCD(ercd) == E_OK*	
22				ercd = act_tsk(TASK_ID2)	
23		ercd = wai_sem(SEM_ID1)			
24				*MERCD(ercd) == E_OK*	
25				ercd = act_tsk(TASK_ID1)	
26	ercd = wai_sem(SEM_ID1)				
27				*MERCD(ercd) == E_OK*	
28				ercd = sig_sem(SEM_ID1)	
29			*MERCD(ercd) == E_OK*		CRE_SEM-6 sig_sem-3,5
30			ext_tsk( )		
31				*MERCD(ercd) == E_OK*	sig_sem-4
32				ercd = sig_sem(SEM_ID1)	
33		*MERCD(ercd) == E_OK*			wai_sem-5,6
34		ext_tsk( )			
35				*MERCD(ercd) == E_OK*	
36				ercd = rel_wai(TASK_ID1)	
37	*MERCD(ercd) == E_RLWAI*				wai_sem-2
38	ext_tsk( )				
39				*MERCD(ercd) == E_OK*	
40				ercd = sig_sem(SEM_ID2)	
41				*MERCD(ercd) == E_OK*	
42				ext_tsk( )	

SEM3：セマフォ機能のテスト手順　その3

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
S2	CRE_TSK(TASK_ID2_1,{TA_HLNG,EXINF_2_1,TASK2_1,ITSKPRI_2,STKSZ,NULL})	
S3	CRE_TSK(TASK_ID2_2,{TA_HLNG,EXINF_2_2,TASK2_2,ITSKPRI_2,STKSZ,NULL})	
S4	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})	
S5	CRE_SEM(SEM_ID1,{TA_TFIFO,ISEMCNT_1,MAXSEM_2})	
S6	CRE_SEM(SEM_ID2,{TA_TFIFO,ISEMCNT_1,MAXSEM_1})	
S7	DEF_INH(INHNO,{INHATR,INTHDR_1}　【何らかの割込みを発生できるシステムとすること】	

－	**<TASK_ID1:DORMANT>**	**<TASK_ID2_1:DORMANT>**	**<TASK_ID2_2:DORMANT>**	**<TASK_ID3:READY>**	**<INTHDT_1:割込みハンドラ>**	－
1				ercd = pol_sem(SEM_ID2)		
2				*MERCD(ercd) == E_OK*		
3				ercd = act_tsk(TASK_ID2_1)		
4		ercd = wai_sem(SEM_ID2)				
5				*MERCD(ercd) == E_OK*		
6				ercd = act_tsk(TASK_ID2_2)		
7			ercd = wai_sem(SEM_ID2)			
8				*MERCD(ercd) == E_OK*		
9				ercd = act_tsk(TASK_ID1)		
10	ercd = wai_sem(SEM_ID2)					
11				*MERCD(ercd) == E_OK*		
12				INTHDR_1の割込み発生		
13					ercd = isig_sem(ERR_SEM_ID)	
14					*MERCD(ercd) == E_ID*	isig_sem-1
15					ercd = isig_sem(SEM_ID1)	
16					*MERCD(ercd) == E_OK*	非タスクコンテキストから呼び出せるサービスコール-4
17					ercd = isig_sem(SEM_ID1)	
18					*MERCD(ercd) == E_QOVR*(注)	isig_sem-2
19					ercd = isig_sem(SEM_ID2)	
20					*MERCD(ercd) == E_OK*	
21					return	
22		*MERCD(ercd) == E_OK*				isig_sem-3,5
23		ext_tsk( )				
24				ercd = sig_sem(SEM_ID2)		
25			*MERCD(ercd) == E_OK*			isig_sem-4
26			ext_tsk( )			

27				*MERCD(ercd) == E_OK*		
28				ercd = sig_sem(SEM_ID2)		
29	*MERCD(ercd) == E_OK*					
30	ext_tsk( )					
31				*MERCD(ercd) == E_OK*		
32				ercd = pol_sem(SEM_ID1)		
33				*MERCD(ercd) == E_OK*		
34				ercd = pol_sem(SEM_ID1)		
35				*MERCD(ercd) == E_OK*		isig_sem-6
36				ercd = pol_sem(SEM_ID1)		
37				*MERCD(ercd) == E_TMOUT*		
38				ext_tsk( )		

(注)

No.18　サービスコールを遅延実行する場合には、E_QOVRエラーを返すことを省略できる。省略する場合は、製品マニュアルにその旨が記載されていて、E_OKを返すこと。

FLG1：イベントフラグ機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_FLG(ERR_FLG_ID,{TA_TFIFO,IFLGPTN_1})	
S2	E_IDを検出できること	CRE_FLG-1
S3	CRE_FLG(FLG_ID1,{ERR_FLGATR,IFLGPTN_1})	
S4	E_RSATRを検出できること	CRE_FLG-2
S5	CRE_FLG(FLG_ID1,{TA_TFIFO,ERR_IFLGPTN})	
S6	E_PARを検出できること	CRE_FLG-3
S7	CRE_FLG(FLG_ID1,{TA_FIFO,IFLGPTN_1})	
S8	CRE_FLG(FLG_ID1,{TA_FIFO,IFLGPTN_1})	
S9	E_OBJを検出できること	CRE_FLG-4

(注)

No.S6　E_PARエラーを検出できるERR_IFLGPTNがシステムに存在しない場合は、テストを省略することができる。

FLG2：イベントフラグ機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
S3	CRE_FLG(FLG_ID1,{TA_TFIFO|TA_WSGL,IFLGPTN_0})		
S4	CRE_FLG(FLG_ID2,{TA_TFIFO|TA_WSGL,IFLGPTN_0})		
S5	CRE_FLG(FLG_ID3,{TA_TFIFO|TA_WSGL|TA_CLR,IFLGPTN_1})		
—	**<TASK_ID1:DORMANT>**	**<TASK_ID2:READY>**	—
1		ercd = set_flg(ERR_FLG_ID,SETPTN_1)	
2		*MERCD(ercd) == E_ID*	set_flg-1
3		ercd = set_flg(FLG_ID1,ERR_SETPTN)	
4		*MERCD(ercd) == E_PAR*	set_flg-2
5		ercd = clr_flg(ERR_FLG_ID,CLRPTN_1)	
6		*MERCD(ercd) == E_ID*	clr_flg-1
7		ercd = clr_flg(FLG_ID1,ERR_CLRPTN)	
8		*MERCD(ercd) == E_PAR*	clr_flg-2
9		ercd = wai_flg(ERR_FLG_ID,WAIPTN_1,TWF_ANDW,p_flgptn)	
10		*MERCD(ercd) == E_ID*	wai_flg-1
11		ercd = wai_flg(FLG_ID1,ERR_WAIPTN,TWF_ORW,p_flgptn)	
12		*MERCD(ercd) == E_PAR*	wai_flg-2
13		ercd = wai_flg(FLG_ID1,WAIPTN_1,ERR_WFMODE,p_flgptn)	
14		*MERCD(ercd) == E_PAR*	wai_flg-3
15		ercd = wai_flg(FLG_ID1,WAIPTN_1,TWF_ANDW,ERR_pointer)	
16		*MERCD(ercd) == E_PAR*	wai_flg-4
17		ercd = pol_flg(ERR_FLG_ID,WAIPTN_1,TWF_ANDW,p_flgptn)	
18		*MERCD(ercd) == E_ID*	pol_flg-1
19		ercd = pol_flg(FLG_ID1,ERR_WAIPTN,TWF_ORW,p_flgptn)	
20		*MERCD(ercd) == E_PAR*	pol_flg-2
21		ercd = pol_flg(FLG_ID1,WAIPTN_1,ERR_WFMODE,p_flgptn)	
22		*MERCD(ercd) == E_PAR*	pol_flg-3
23		ercd = pol_flg(FLG_ID1,WAIPTN_1,TWF_ANDW,ERR_pointer)	
24		*MERCD(ercd) == E_PAR*	pol_flg-4
25		ercd = set_flg(FLG_ID1,SETPTN_5555)	
26		*MERCD(ercd) == E_OK*	CRE_FLG-5,6
27		ercd = set_flg(FLG_ID2,SETPTN_AAAA)	
28		*MERCD(ercd) == E_OK*	
29		ercd = wai_flg(FLG_ID2,WAIPTN_0002,TWF_ORW,p_flgptn)	
30		*MERCD(ercd) == E_OK*	wai_flg-9
31		*flgptn == SETPTN_AAAA*	wai_flg-10
32		ercd = pol_flg(FLG_ID2,WAIPTN_0001,TWF_ORW,p_flgptn)	
33		*MERCD(ercd) == E_TMOUT*	pol_flg-6
34		ercd = clr_flg(FLG_ID1,CLRPTN_0000)	
35		*MERCD(ercd) == E_OK*	
36		ercd = pol_flg(FLG_ID1,WAIPTN_0001,TWF_ANDW,p_flgptn)	
37		*MERCD(ercd) == E_TMOUT*	pol_flg- 7 clr_flg-3
38		ercd = pol_flg(FLG_ID1,WAIPTN_0000,TWF_ANDW,p_flgptn)	

39		*MERCD(ercd) == E_PAR*	pol_flg-11
40		ercd = set_flg(FLG_ID1,SETPTN_0055)	
41		*MERCD(ercd) == E_OK*	
42		ercd = pol_flg(FLG_ID1,WAIPTN_0050,TWF_ORW,p_flgptn)	
43		*MERCD(ercd) == E_OK*	pol_flg-8
44		*flgptn == FLGPTN_0055*	pol_flg-9
45		ercd = set_flg(FLG_ID1,SETPTN_5500)	
46		*MERCD(ercd) == E_OK*	
47		ercd = pol_flg(FLG_ID1,WAIPTN_0055,TWF_ORW,p_flgptn)	
48		*MERCD(ercd) == E_OK*	
49		*flgptn == FLGPTN_5555*	set_flg-3
50		ercd = clr_flg(FLG_ID1,CLRPTN_0000)	
51		*MERCD(ercd) == E_OK*	
52		ercd = act_tsk(TASK_ID1)	
53	ercd = wai_flg(FLG_ID1,WAIPTN_0001,TWF_ANDW,p_flgptn)		
54		*MERCD(ercd) == E_OK*	
55		ercd = wai_flg(FLG_ID1,WAIPTN_0002,TWF_ANDW,p_flgptn)	
56		*MERCD(ercd) == E_ILUSE*	wai_flg-5,7
57		ercd = pol_flg(FLG_ID1,WAIPTN_0002,TWF_ORW,p_flgptn)	
58		*MERCD(ercd) == E_ILUSE*	pol_flg-5
59		ercd = rel_wai(TASK_ID1)	
60	*MERCD(ercd) == E_RLWAI*		wai_flg-6,12
61	ercd = wai_flg(FLG_ID1,WAIPTN_0001,TWF_ORW,p_flgptn)		
62		*MERCD(ercd) == E_OK*	
63		ercd = set_flg(FLG_ID1,SETPTN_1111)	
64	*MERCD(ercd) == E_OK*		set_flg-4,5 wai_flg-8
65	*flgptn == SETPTN_1111*		set_flg-6
66	ercd = wai_flg(FLG_ID1,WAIPTN_0000,TWF_ANDW,p_flgptn)		
67	*MERCD(ercd) == E_PAR*		wai_flg-13
68	ercd = wai_flg(FLG_ID2,WAIPTN_AAAA,TWF_ANDW,p_flgptn)		
69	*MERCD(ercd) == E_OK*		wai_flg-7,9
70	*flgptn == SETPTN_AAAA*		wai_flg-10
71	ercd = wai_flg(FLG_ID3,WAIPTN_0001,TWF_ORW,p_flgptn)		
72	*MERCD(ercd) == E_OK*		wai_flg-8
73	*flgptn == SETPTN_0001*		CRE_FLG-9
74	ercd = set_flg(FLG_ID3,SETPTN_0002)		
75	*MERCD(ercd) == E_OK*		
76	ercd = pol_flg(FLG_ID3,WAIPTN_0002,TWF_ANDW,p_flgptn)		
77	*MERCD(ercd) == E_OK*		pol_flg-7
78	*flgptn == SETPTN_0002*		
79	ercd = pol_flg(FLG_ID3,WAIPTN_0002,TWF_ANDW,p_flgptn)		
80	*MERCD(ercd) == E_TMOUT*		pol_flg-10
81	ercd = wai_flg(FLG_ID3,WAIPTN_8000,TWF_ORW,p_flgptn)		
82		*MERCD(ercd) == E_OK*	
83		ercd = set_flg(FLG_ID3,SETPTN_8000)	
84	*MERCD(ercd) == E_OK*		set_flg-5 pol_flg-8
85	*flgptn == SETPTN_8000*		set_flg-6
86	ercd = pol_flg(FLG_ID3,WAIPTN_8000,TWF_ORW,p_flgptn)		
87	*MERCD(ercd) == E_TMOUT*		set_flg-7
88	ercd = set_flg(FLG_ID3,SETPTN_5555)		
89	*MERCD(ercd) == E_OK*		
90	ercd = wai_flg(FLG_ID3,WAIPTN_5555,TWF_ORW,p_flgptn)		
91	*MERCD(ercd) == E_OK*		
92	*flgptn == SETPTN_5555*		
93	ercd = pol_flg(FLG_ID3,WAIPTN_5555,TWF_ORW,p_flgptn)		
94	*MERCD(ercd) == E_TMOUT*		wai_flg-11
95	ext_tsk( )		
96		*MERCD(ercd) == E_OK*	
97		ext_tsk( )	

(注)

No.4　E_PARエラーを検出できるERR_SETPTNがシステムに存在しない場合は、テストを省略することができる。
No.8　E_PARエラーを検出できるERR_CLRPTNがシステムに存在しない場合は、テストを省略することができる。
No.12　E_PARエラーを検出できるERR_WAIPTNがシステムに存在しない場合は、テストを省略することができる。
No.16　E_PARエラーを検出できるERR_pointerがシステムに存在しない場合は、テストを省略することができる。
No.20　E_PARエラーを検出できるERR_WAIPTNがシステムに存在しない場合は、テストを省略することができる。
No.24　E_PARエラーを検出できるERR_pointerがシステムに存在しない場合は、テストを省略することができる。

FLG3：イベントフラグ機能のテスト手順　その3

No	テ　ス　ト　手　順　内　容				テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})				
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})				
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})				
S4	DEF_INT(INHNO,{INHATR,INTHDR_1})				
S5	CRE_FLG(FLG_ID1,{TA_TFIFO|TA_WSGL|TA_CLR,IFLGPTN_0})				
S6	CRE_FLG(FLG_ID2,{TA_TFIFO|TA_WSGL,IFLGPTN_0})				
－	**<TASK_ID1:READY>**	**<INTHDR_1:割込みハンドラ>**	**<TASK_ID2:READY>**	**<TASK_ID3:READY>**	－
1	ercd = wai_flg(FLG_ID2,WAIPTN_0505, TWF_ORW,p_flgptn)				
2			ercd = wai_flg(FLG_ID1,WAIPTN_0505, TWF_ANDW,p_flgptn)		
3				INTHDR_1の割込みを発生させる	
4		ercd = iset_flg(ERR_FLG_ID,SETPTN_0001)			
5		*MERCD(ercd) == E_ID*			iset_flg-1
6		ercd = iset_flg(FLG_ID2,ERR_SETPTN)			
7		*MERCD(ercd) == E_PAR*			iset_flg-2
8		ercd = iset_flg(FLG_ID2,SETPTN_0500)			
9		*MERCD(ercd) == E_OK*			非タスクコンテキストから呼び出せるサービスコール-5
10		ercd = iset_flg(FLG_ID1,SETPTN_0500)			
11		*MERCD(ercd) == E_OK*			
12		ercd = iset_flg(FLG_ID2,SETPTN_0005)			
13		*MERCD(ercd) == E_OK*			
14		ercd = iset_flg(FLG_ID1,SETPTN_0005)			
15		*MERCD(ercd) == E_OK*			
16		return			
17	*MERCD(ercd) == E_OK*				iset_flg-4,5
18	*flgptn == SETPTN_0500*				iset_flg-3,6
19	ext_tsk(　)				
20			*MERCD(ercd) == E_OK*		CRE_FLG-7,8
21			*flgptn == SETPTN_0505*		
22			ercd = pol_flg(FLG_ID1,WAIPTN_0001, TWF_ORW,p_flgptn)		
23			*MERCD(ercd) == E_TMOUT*		iset_flg-7
24			ext_tsk(　)		

(注)

No.7　E_PARエラーを検出できるERR_SETPTNがシステムに存在しない場合は、テストを省略することができる。

DTQ1：データキュー機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_DTQ(ERR_DTQID,{TA_TFIFO,DTQCNT_1,NULL})	
S2	*E_IDエラーを検出できること。*	CRE_DTQ-1
S3	CRE_DTQ(DTQ_ID1,{ERR_DTQATR,DTQCNT_1,NULL})	
S4	*E_RSATRエラーを検出できること。*	CRE_DTQ-2
S5	CRE_DTQ(DTQ_ID1,{TA_FIFO,ERR_DTQCNT,NULL})	
S6	*E_PARエラーを検出できること。*	CRE_DTQ-3
S7	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_1,NULL})	
S8	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_1,NULL})	
S9	*E_OBJエラーを検出できること。*	CRE_DTQ-4

(注)

No.S6　E_PARエラーを検出できるERR_DTQCNTがシステムに存在しない場合は、テストを省略することができる。

DTQ2：データキュー機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
S3	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_0,NULL})		
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	－
1	ercd = psnd_dtq(ERR_DTQID,DATA_55)		
2	*MERCD(ercd) == E_ID*		psnd_dtq-1
3	ercd = psnd_dtq(DTQ_ID1,DATA_55)		
4	*MERCD(ercd) == E_TMOUT*		psnd_dtq-2
5	ercd = rcv_dtq(DTQ_ID1,p_data)		
6		ercd = psnd_dtq(DTQ_ID1,DATA_AA)	
7	*MERCD(ercd) == E_OK*		CRE_DTQ-5 psnd_dtq-5,6
8	*data == DATA_AA*		psnd_dtq-3,4,7,8
9	ercd = rcv_dtq(DTQ_ID1,p_data)		
10		*MERCD(ercd) == E_OK*	CRE_DTQ-6
11		ercd = psnd_dtq(DTQ_ID1,MEMORY)	
12	*MERCD(ercd) == E_OK*		
13	*data == MEMORY*		psnd_dtq-9
14	ext_tsk(　)		
15		*MERCD(ercd) == E_OK*	
16		ext_tsk(　)	

DTQ3：データキュー機能のテスト手順　その3

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	DEF_INH(INHNO_1,{INHATR_1,INTHDR_1})			
S4	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_0,NULL})			
S5	CRE_DTQ(DTQ_ID2,{TA_TFIFO,DTQCNT_2,NULL})			
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<INTHDR_1:割込みハンドラ>**	－
1	ercd = rcv_dtq(DTQ_ID2,p_data)			
2		<INTHDR_1の割込み発生>		
3			ercd = ipsnd_dtq(ERR_DTQID,DATA55)	
4			*MERCD(ercd) == E_ID*	ipsnd_dtq-1
5			ercd = ipsnd_dtq(DTQ_ID1,DATA_AA)	
6			*MERCD(ercd) == E_TMOUT*（注）	ipsnd_dtq-2
7			ercd = ipsnd_dtq(DTQ_ID2,DATA_88)	
8			*MERCD(ercd) == E_OK*	
9			ercd = ipsnd_dtq(DTQ_ID2,MEMORY)	
10			*MERCD(ercd) == E_OK*	非タスクコンテキストから呼び出せるサービスコール-6
11			return	
12	*MERCD(ercd) == E_OK*			ipsnd_dtq-5,6,7
13	*data == DATA_88*			ipsnd_dtq-3,4,8
14	ercd = rcv_dtq(DTQ_ID2,p_data)			
15	*MERCD(ercd) == E_OK*			
16	*data == MEMORY*			ipsnd_dtq-9
17	ext_tsk(　)			
18		ext_tsk(　)		

(注)
　No.6　E_TMOUTエラーの検出を省略することが製品マニュアルに記載されている場合：MERCD(ercd) == E_OK

DTQ4：データキュー機能のテスト手順　その4

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
S3	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_0,NULL})		
S4	CRE_DTQ(DTQ_ID2,{TA_TFIFO,DTQCNT_1,NULL})		
S5	CRE_DTQ(DTQ_ID3,{TA_TFIFO,DTQCNT_2,NULL})		
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	－
1	ercd = fsnd_dtq(ERR_DTQID,DATA_00)		
2	*MERCD(ercd) == E_ID*		fsnd_dtq-1
3	ercd = fsnd_dtq(DTQ_ID1,DATA_00)		
4	*MERCD(ercd) == E_ILUSE*		fsnd_dtq-2
5	ercd = fsnd_dtq(DTQ_ID2,DATA_11)		
6	*MERCD(ercd) == E_OK*		
7	ercd = fsnd_dtq(DTQ_ID2,DATA_22)		
8	*MERCD(ercd) == E_OK*		
9	ercd = prcv_dtq(DTQ_ID2,p_data)		
10	*MERCD(ercd) == E_OK*		
11	*data == DATA_22*		fsnd_dtq-7
12	ercd = rcv_dtq(DTQ_ID2,p_data)		

13		ercd = fsnd_dtq(DTQ_ID2,DATA_33)	
14	*MERCD(ercd) == E_OK*		fsnd_dtq-3,4
15	*data == DATA_33*		fsnd_dtq-5
16	ercd = fsnd_dtq(DTQ_ID3,DATA_44)		
17	*MERCD(ercd) == E_OK*		
18	ercd = fsnd_dtq(DTQ_ID3,DATA_55)		
19	*MERCD(ercd) == E_OK*		
20	ercd = rcv_dtq(DTQ_ID3,p_data)		
21	*MERCD(ercd) == E_OK*		
22	*data == DATA_44*		
23	ercd = rcv_dtq(DTQ_ID3,p_data)		
24	*MERCD(ercd) == E_OK*		
25	*data == DATA_55*		fsnd_dtq-6
26	ercd = fsnd_dtq(DTQ_ID2,DATA_66)		
27	*MERCD(ercd) == E_OK*		
28	ercd = rcv_dtq(DTQ_ID2,p_data)		
29	*MERCD(ercd) == E_OK*		
30	*data == DATA_66*		
31	ext_tsk( )		
32		*MERCD(ercd) == E_OK*	
33		ext_tsk( )	

DTQ5：データキュー機能のテスト手順　その5

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_0,NULL})			
S4	CRE_DTQ(DTQ_ID2,{TA_TFIFO,DTQCNT_1,NULL})			
S5	CRE_DTQ(DTQ_ID3,{TA_TFIFO,DTQCNT_1,NULL})			
S6	CRE_DTQ(DTQ_ID4,{TA_TFIFO,DTQCNT_2,NULL})			
S7	DEF_INH(INHNO_1,{INHATR_1,INTHDR_1})			
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<INTHDR_1:割込みハンドラ>**	－
1	ercd = rcv_dtq(DTQ_ID2,p_data)			
2		<INTHDR_1の割込み発生>		
3			ercd = ifsnd_dtq(ERR_DTQID,DATA_77)	
4			*MERCD(ercd) == E_ID*	ifsnd_dtq-1
5			ercd = ifsnd_dtq(DTQ_ID1,DATA_77)	
6			*MERCD(ercd) == E_ILUSE*	ifsnd_dtq-2
7			ercd = ifsnd_dtq(DTQ_ID2,DATA_88)	
8			*MERCD(ercd) == E_OK*	
9			ercd = ifsnd_dtq(DTQ_ID3,DATA_99)	
10			*MERCD(ercd) == E_OK*	
11			ercd = ifsnd_dtq(DTQ_ID3,DATA_AA)	
12			*MERCD(ercd) == E_OK*	非タスクコンテキストから呼び出せるサービスコール-7
13			ercd = ifsnd_dtq(DTQ_ID4,DATA_11)	
14			*MERCD(ercd) == E_OK*	
15			ercd = ifsnd_dtq(DTQ_ID4,DATA_22)	
16			*MERCD(ercd) == E_OK*	
17			return	
18	*MERCD(ercd) == E_OK*			ifsnd_dtq-3,4
19	*data == DATA_88*			ifsnd_dtq-5
20	ext_tsk( )			
21		ercd = prcv_dtq(DTQ_ID3,p_data)		
22		*MERCD(ercd) == E_OK*		
23		*data == DATA_AA*		ifsnd_dtq-7
24		ercd = prcv_dtq(DTQ_ID4,p_data)		
25		*MERCD(ercd) == E_OK*		
26		*data == DATA_11*		
27		ercd = prcv_dtq(DTQ_ID4,p_data)		
28		*MERCD(ercd) == E_OK*		
29		*data == DATA_22*		ifsnd_dtq-6
30		ext_tsk( )		

DTQ6：データキュー機能のテスト手順　その6

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})			
S3	CRE_TSK(TASK_ID3,{TA_HLNG|TA_ACT,EXINF_3,TASK3,ITSKPRI_3,STKSZ,NULL})			
S4	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_1,NULL})			
S5	CRE_DTQ(DTQ_ID2,{TA_TFIFO,DTQCNT_0,NULL})			
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	**<TASK_ID3:READY>**	－
1	ercd = rcv_dtq(ERR_DTQID,p_data)			
2	*MERCD(ercd) == E_ID*			rcv_dtq-1
3	ercd = rcv_dtq(DTQ_ID1,ERR_pointer)			
4	*MERCD(ercd) == E_PAR*			rcv_dtq-2
5	ercd = psnd_dtq(DTQ_ID1,DATA_01)			

6	*MERCD(ercd) == E_OK*			
7	ercd = rcv_dtq(DTQ_ID1,p_data)			
8	*MERCD(ercd) == E_OK*			
9	*data == DATA_01*			rcv_dtq-4
10	ercd = rcv_dtq(DTQ_ID1,p_data)			
11		ercd = rcv_dtq(DTQ_ID1,p_data)		
12			ercd = psnd_dtq(DTQ_ID1,DATA_05)	
13	*MERCD(ercd) == E_OK*			rcv_dtq-5
14	*data == DATA_05*			
15	ercd = rel_wai(TASK_ID2)			
16	*MERCD(ercd) == E_OK*			
17	ext_tsk( )			
18		*MERCD(ercd) == E_RLWAI*		rcv_dtq-3,6
19		ext_tsk( )		
20			*MERCD(ercd) == E_OK*	
21			ext_tsk( )	

(注)
No.4　E_PARエラーを検出できるERR_pointerがシステムに存在しない場合は、テストを省略することができる。

DTQ7：データキュー機能のテスト手順　その7

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT,EXINF_2,TASK2,ITSKPRI_2,STKSZ,NULL})		
S3	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_0,NULL})		
－	**<TASK_ID1:READY>**	**<TASK_ID2:READY>**	－
1	ercd = prcv_dtq(ERR_DTQID,p_data)		
2	*MERCD(ercd) == E_ID*		prcv_dtq-1
3	p_dataにE_PARエラーを検出する値を設定		
4	ercd = prcv_dtq(DTQ_ID1,p_data)		
5	*MERCD(ercd) == E_PAR*		prcv_dtq-2
6	ercd = prcv_dtq(DTQ_ID1,p_data)		
7	*MERCD(ercd) == E_TMOUT*		prcv_dtq-3
8	ercd = psnd_dtq(DTQ_ID1,DATA_99)		
9	*MERCD(ercd) == E_OK*		
10	ext_tsk( )		
11		ercd = prcv_dtq(DTQ_ID1,p_data)	
12		*MERCD(ercd) == E_OK*	
13		*data == DATA_99*	prcv_dtq-4
14		ext_tsk( )	

(注)
No.5　E_PARエラーを検出できるp_dataがシステムに存在しない場合は、テストを省略することができる。

CYC1：周期ハンドラ機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_CYC(CYC_ID1,{ERR_CYCATR,EXINF_CYC1,CYCHDR_1,CYCTIM_10,CYCPHS_5})	
S2	*E_RSATRエラーを検出できること*	CRE_CYC-1
S3	CRE_CYC(CYC_ID1,{TA_HLNG,EXINF_CYC1,ERR_CYCHDR,CYCTIM_10,CYCPHS_5})	
S4	*E_PARエラーを検出できること*	CRE_CYC-2
S5	CRE_CYC(CYC_ID1,{TA_HLNG|TA_STA,EXINF_CYC1,CYCHDR_1,CYCTIM_0,CYCPHS_5})	
S6	*E_PARエラーを検出できること*	CRE_CYC-3
S7	CRE_CYC(CYC_ID1,{TA_HLNG|TA_STA,EXINF_CYC1,CYCHDR_1,CYCTIM_0,ERR_CYCPHS})	
S8	*E_PARエラーを検出できること*	CRE_CYC-4

(注)
No.S4　E_PARエラーを検出できるERR_CYCHDRがシステムに存在しない場合は、テストを省略することができる。

CYC2：周期ハンドラ機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_CYC(CYC_ID1,{TA_HLNG|TA_STA,EXINF_CYC1,CYCHDR_1,CYCTIM_10,CYCPHS_5})			
S2	CRE_CYC(CYC_ID2,{TA_HLNG,EXINF_CYC2,CYCHDR_2,CYCTIM_55,CYCPHS_0})			
S3	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
－	**<TASK_ID1:READY>**	**<CYCHDR_1:周期ハンドラ>**	**<CYCHDR_2:周期ハンドラ>**	－
1	共通変数cyc_work1,cyc_work2を0クリア			
2	ercd = sta_cyc(ERR_CYCID)			
3	*MERCD(ercd) == E_ID*			sta_cyc-1
4	ercd = stp_cyc(ERR_CYCID)			
5	*MERCD(ercd) == E_ID*			stp_cyc-1
6	ercd = stp_cyc(CYC_ID2)			
7	*MERCD(ercd) == E_OK*			stp_cyc-3
8	100msec以上のソフトウェアディレーを実行			
9	*cyc_work2 == 0*			周期ハンドラ-2 CRE_CYC-6
10	*cyc_work1 ≧ 9*			ｼｽﾃﾑ時刻管理-1 isig_tim-1 CRE_CYC-5,7
11	ercd = stp_cyc(CYC_ID1)			

12	*MERCD(ercd) == E_OK*			
13	work = cyc_work1(カウンタのセーブ)			
14	100msec以上のソフトウェアディレーを実行			
15	*cyc_work1 == work*			stp_cyc-2
16	ercd = sta_cyc(CYC_ID2)			
17	*MERCD(ercd) == E_OK*			
18	100msec以上のソフトウェアディレーを実行			
19	*cyc_work2 == 1*			sta_cyc-2
20	ercd = sta_cyc(CYC_ID2)			
21	*MERCD(ercd) == E_OK*			
22	50msec以上のソフトウェアディレーを実行			
23	*MERCD(ercd) == E_OK*			
24	*cyc_work2 == 1*			sta_cyc-3
25	ercd = stp_cyc(CYC_ID2)			
26	*MERCD(ercd) == E_OK*			
27	ext_tsk( )			
28		<CYCTIM_10周期に起動される＞		
29		*exinf == EXINF_CYC1*		システム時刻管理-3 周期ハンドラ-1
30		cyc_work1 ++		
31		return		
32			< CYCTIM_55周期に起動される＞	
33			*exinf == EXINF_CYC2*	
34			cyc_work2 ++	
35			return	

SYS1：システム状態管理機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容	テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})	
—	**<TASK_ID1:READY>**	—
1	ercd = get_tid(&tskid)	
2	*MERCD(ercd) == E_OK*	
3	*tskid == TASK_ID1*	get_tid-1
4	state = sns_dpn( )	
5	*state == FALSE*	sns_dpn-4
6	ercd = loc_cpu( )	
7	*MERCD(ercd) == E_OK*	
8	state = sns_loc( )	
9	*state == TRUE*	loc_cpu-1 sns_loc-1
10	state = sns_dpn( )	
11	*state == TRUE*	sns_dpn-2
12	ercd = loc_cpu( )	
13	*MERCD(ercd) == E_OK*	
14	state = sns_loc( )	
15	*state == TRUE*	loc_cpu-2
16	ercd = unl_cpu( )	
17	*MERCD(ercd) == E_OK*	
18	state = sns_loc( )	
19	*state == FALSE*	unl_cpu-1 sns_loc-2
20	ercd = unl_cpu( )	
21	*MERCD(ercd) == E_OK*	
22	state = sns_loc( )	
23	*state == FALSE*	unl_cpu-2
24	state = sns_ctx( )	
25	*state == FALSE*	sns_ctx-1
26	ercd = dis_dsp( )	
27	*MERCD(ercd) == E_OK*	
28	state = sns_dsp( )	
29	*state == TRUE*	dis_dsp-1 sns_dsp-1
30	state = sns_dpn( )	
31	*state == TRUE*	sns_dpn-3
32	ercd = dis_dsp( )	
33	*MERCD(ercd) == E_OK*	
34	state = sns_dsp( )	
35	*state == TRUE*	dis_dsp-2
36	ercd = ena_dsp( )	
37	*MERCD(ercd) == E_OK*	
38	state = sns_dsp( )	
39	*state == FALSE*	ena_dsp-1 sns_dsp-2
40	ercd = ena_dsp( )	
41	*MERCD(ercd) == E_OK*	
42	state = sns_dsp( )	
43	*state == FALSE*	ena_dsp-2
44	ext_tsk( )	

SYS2：システム状態管理機能のテスト手順　その2

No	テ　ス　ト　手　順　内　容			テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})			
S2	DEF_INH(INHNO,{INHATR,INTHDR_1})			
—	**<TASK_ID1:READY>**	**<INTHDR_1:割込みハンドラ>**		—
		<1回目>	<2回目>	
1	INTHDR_1の割込み発生			
2		ercd = iget_tid(&tskid)		
3		*tskid== TASK_ID1*		非タスクコンテキストから 呼び出せるサービスコール-9 iget_tid-1
4		ercd = iloc_cpu( )		
5		*MERCD(ercd) == E_OK*		非タスクコンテキストから呼び 出せるサービスコール-10
6		state = sns_loc( )		
7		*state == TRUE*		iloc_cpu-1
8		ercd = iloc_cpu( )		
9		MERCD(ercd) == E_OK		
10		state = sns_loc( )		
11		*state == TRUE*		iloc_cpu-2
12		ercd = iunl_cpu( )		
13		*MERCD(ercd) == E_OK*		非タスクコンテキストから呼び 出せるサービスコール-11
14		state = sns_loc( )		
15		*state == FALSE*		iunl_cpu-1
16		ercd = iunl_cpu( )		
17		MERCD(ercd) == E_OK		
18		state = sns_loc( )		
19		*state == FALSE*		iunl_cpu-2
20		state = sns_ctx( )		
21		*state == TRUE*		sns_ctx-2
22		return		
23				
24				
25	10msec後に割込み発生			
26	ext_tsk( )			
27			ercd = iget_tid(&tskid)	
28			tskid == TSK_NONE	iget_tid-2
29			state = sns_dpn( )	
30			*state = TRUE*	sns_dpn-1
31			return	

INH1：割込み管理機能のテスト手順　その1

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	DEF_INH(INHNO_1,{ERR_INHATR,INTHDR_1})		
S2	*E_RSATRエラーを検出できること。*		DEF_INH-1
S3	DEF_INH(ERR_INHNO,{INHATR,INTHDR_1})		
S4	*E_PARエラーを検出できること。*		DEF_INH-2
S5	DEF_INH(INHNO_1,{INHATR,ERR_INTHDR})		
S6	*E_PARエラーを検出できること。*		DEF_INH-3
S7	DEF_INH(INHNO_1,{INHATR,INTHDR_1})		静的APIの文法と パラメータ-3
S8	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
—	**<TASK_ID1:READY>**	**<INTHDR_1:割込みハンドラ>**	—
1	INHNO_1の割込み発生		
2		return	DEF_INH-4
3	ext_tsk( )		

(注)

No.S6　E_PARエラーを検出できるERR_INTHDRがシステムに存在しない場合は、テストを省略することができる。

ISR1：割込みサービスルーチンのテスト手順　その1

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	ATT_ISR(ERR_ISRATR,{EXINF_1,INTNO_1,ISR_1})		
S2	*E_RSATRエラーを検出できること。*		ATT_ISR-1
S3	ATT_ISR(TA_HLNG,{EXINF_1,ERR_INTNO,ISR_1})		
S4	*E_PARエラーを検出できること。*		ATT_ISR-2
S5	ATT_ISR({TA_HLNG,EXINF_1,INTNO_1,ISR_ADR_0})		
S6	*E_PARエラーを検出できること。*		ATT_ISR-3
S7	ATT_ISR({TA_HLNG,EXINF_1,INTNO_1,ISR_1})		
S8	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
－	**<TASK_ID1:READY>**	**<ISR_1:割込みサービスルーチン>**	
1	INTNO_1の割込み番号に対応する割込み発生		
2		*exinf == EXINF_1*	割込み管理機能-4 ATT_ISR-4
3		return	
4	ext_tsk(　)		

(注)
No.S6　E_PARエラーを検出できるISR_ADR_0がシステムに存在しない場合は、テストを省略することができる。

EXC1：CPU例外ハンドラのテスト手順　その1

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	DEF_EXC(EXCNO_1,{ERR_EXCATR,EXCHDR_1})		
S2	*E_RSATRエラーを検出できること。*		DEF_EXC-1
S3	DEF_EXC(ERR_EXCNO,{EXCATR,EXCHDR_1})		
S4	*E_PARエラーを検出できること。*		DEF_EXC-2
S5	DEF_EXC(EXCNO_1,{EXCATR,ERR_EXCHDR})		
S6	*E_PARエラーを検出できること。*		DEF_EXC-3
S7	DEF_EXC(EXCNO_1,{EXCATR,EXCHDR_1})		
S8	DEF_EXC(EXCNO_1,{EXCATR,EXCHDR_2})		
S9	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
－	**<TASK_ID1:READY>**	**<EXCHDR_2:CPU例外ハンドラ>**	－
1	EXCHDR_2の割込み発生		
2		return	DEF_EXC-5
3	ext_tsk(　)		

(注)
No.S6　E_PARエラーを検出できるERR_EXCHDRがシステムに存在しない場合は、テストを省略することができる。

INI1：初期化ルーチンのテスト手順　その1

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	ATT_INI({TA_HLNG,EXINF_1,INIRTN_1})		
S2	CRE_TSK(TASK_ID1,{TA_HLNG|TA_ACT,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
－	**<TASK_ID1:READY>**	**<INIRTN_1:初期化ルーチン>**	－
1		TASK_ID1より先に初期化ルーチンが実行される	ATT_INI-1
2		*exinf == EXINF_1*	ｼｽﾃﾑ構成管理機能-3
3		return	
4	ext_tsk(　)		
5			

RST：制約タスクのテスト手順　その１

No	テ　ス　ト　手　順　内　容		テスト 項目参照
S1	CRE_TSK(TASK_ID1,{TA_HLNG|TA_RSTR,EXINF_1,TASK1,ITSKPRI_1,STKSZ,NULL})		
S2	CRE_TSK(TASK_ID2,{TA_HLNG|TA_ACT|TA_RSTR,EXINF_2,TASK1,ITSKPRI_2,STKSZ,NULL})		
S3	CRE_SEM(SEM_ID1,{TA_FIFO,ISEMCNT_1,MAXSEM_FFFF})		
S4	CRE_FLG(FLG_ID1,{TA_FIFO,IFLGPTN_1})		
S5	CRE_DTQ(DTQ_ID1,{TA_TFIFO,DTQCNT_1,NULL})		
－	**<TASK_ID1:DORMANT>**	**<TASK_ID2:READY>**	－
1		ercd = act_tsk(TASK_ID1)	
2		*MERCD(ercd) ==E_OK*	
3		ercd = ext_tsk(　)	
4		*MERCD(ercd) ==E_NOSPT*	自動車制御用プロファイルに含まれる機能-1
5		ercd = ter_tsk(TASK_ID2)	
6		*MERCD(ercd) ==E_NOSPT*	制約タスク-2 自動車制御用プロファイルに含まれる機能-2
7		ercd = chg_pri(TSK_SELF,TPRI_INI)	
8		*MERCD(ercd) ==E_NOSPT*	自動車制御用プロファイルに含まれる機能-3
9		ercd = slp_tsk(　)	
10		*MERCD(ercd) ==E_NOSPT*	自動車制御用プロファイルに含まれる機能-4
11		ercd = wai_sem(SEM_ID1)	
12		*MERCD(ercd) ==E_NOSPT*	自動車制御用プロファイルに含まれる機能-5
13		ercd = wai_flg(FLG_ID1,WAIPTN_0002,TWF_ORW,p_flgptn)	
14		*MERCD(ercd) ==E_NOSPT*	自動車制御用プロファイルに含まれる機能-6
15		ercd = rcv_dtq(DTQ_ID1,p_data)	
16		*MERCD(ercd) ==E_NOSPT*	自動車制御用プロファイルに含まれる機能-7
17		return	制約タスク-1
18	return		

第３節　製品マニュアルによる確認の見方

以下に、製品マニュアル確認の見方を説明する。

UMA

No.	区　　分	テ　ス　ト　項　目	テスト項目参照
1	APIの構成要素	すべてのサービスコールの名称、パラメータとリターンパラメータの種類・順序・名称・データ型がμITRON4.0仕様書通りに記載されていること。ただし、実装が独自に拡張しているサービスコールは除く。	APIの構成要素-1 サービスコールの返値とエラーコード-1
2		静的APIの名称、パラメータとリターンパラメータの種類・順序・名称・データ型がμITRON4.0仕様書通りに記載されていること。ただし、実装が独自に拡張している静的APIは除く。	APIの構成要素-2
3	サービスコールの返値とエラーコード	サブエラーコードをサポートする場合は、その意味が記載されていること。	サービスコールの返値とエラーコード-2
4	静的APIの文法とパラメータ	静的APIの処理中にエラーが発生した場合の扱いが記載されていること。	静的APIの文法とパラメータ-8
5	サービスコール	実装が独自に用意するサービスコールの名称が記載されていること。	サービスコール-1
6	定数	実装が独自に定義するメインエラーコードの名称とその内容が記載されていること。	定数-1

(1)　No.
製品マニュアルの確認番号を示す。

(2)　区分
製品マニュアルの確認で、確認項目を内容毎に分類したものである。

(3)　テスト項目
μITRON4.0仕様書に記載されている内容であり、テスト手順(プログラム)では確認できない項目である。したがって、検定を受けるカーネルメーカが作成する製品マニュアルに、振舞いや意味が明確に記載されていることを確認する。

(4)　テスト項目参照
製品マニュアルで確認すべき事項が、テスト項目のどの部分に記載されているかを示す。
なお、No.3で示すように、カーネルが当該機能をサポートしていない場合は、検定項目から除くことを明記している。

第４節　製品マニュアルによる確認の手順

UMA：

No.	区　　分	テ　　ス　　ト　　項　　目	テスト項目参照
1	APIの構成要素	自動車制御用プロファイルに含まれるサービスコールの名称、パラメータとリターンパラメータの種類・順序・名称・データ型がμITRON4.0仕様書通りに記載されていること。	APIの構成要素-1 サービスコールの返値とエラーコード-1
2		自動車制御用プロファイルに含まれる静的APIの名称、パラメータの種類・順序・名称・データ型がμITRON4.0仕様書通りに記載されていること。	APIの構成要素-2
3	サービスコールの返値とエラーコード	サブエラーコードをサポートする場合は、その意味が記載されていること。	サービスコールの返値とエラーコード-2
4	静的APIの文法とパラメータ	静的APIの処理中にエラーが発生した場合の扱いが記載されていること。	静的APIの文法とパラメータ-8
5	サービスコール	実装が独自に用意するサービスコールと静的APIの名称がμITRON4.0仕様書通りに記載されていること。	サービスコール-1 静的API-1
6	定数	実装が独自に定義するメインエラーコードの名称とその内容がμITRON4.0仕様書通りに記載されていること。	定数-1
7	割込み処理モデル	割込みハンドラの記述方法（カーネルが用意する出入口処理やアプリケーションが登録する割込みハンドラで行うべき処理）が記載されていること。	割込み処理ハンドラと割込みサービスルーチン-1
8		DEF_INTとATT_ISRの両方をサポートした場合の振舞いが記載されていること。	割込み処理ハンドラと割込みサービスルーチン-2
9		どの優先度より高い優先度を持つものをカーネルの管理外の割込みにするかが、製品マニュアルに記載されていること。	割込み処理ハンドラと割込みサービスルーチン-3
10	CPU例外ハンドラで行える操作	CPU例外ハンドラの記述方法が記載されていること。	CPU例外ハンドラで行える操作-1
11		CPU例外ハンドラ内で呼出し可能なサービスコールが記載されていること。	CPU例外ハンドラで行える操作-2
12		CPU例外が発生したコンテキストや状態の読出し方法が記載されていること。	CPU例外ハンドラで行える操作-3
13		CPU例外が発生したタスクのID番号の読出し方法が記載されていること。	CPU例外ハンドラで行える操作-4
14	タスクコンテキストと非タスクコンテキスト	タスクコンテキストを実行中にCPU例外が発生した場合、CPU例外ハンドラが実行されるコンテキストが製品マニュアルに記載されていること。	タスクコンテキストと非タスクコンテキスト-4
15	処理の優先順位とサービスコールの不可分性	割込みハンドラおよび割込みサービスルーチン相互間の優先順位の関係が記載されていること。	処理の優先順位とサービスコールの不可分性-2
16		周期ハンドラの優先順位と、isig_timを呼び出した割込みハンドラの優先順位との関係が記載されていること。	処理の優先順位とサービスコールの不可分性-3
17		CPU例外ハンドラの優先順位が、CPU例外が発生した処理の優先順位と、ディスパッチャの優先順位のいずれよりも高いことが製品マニュアルに記載されていること。	処理の優先順位とサービスコールの不可分性-4
18		CPU例外ハンドラと、割込みハンドラやタイムイベントハンドラとの間の優先順位が記載されていること。	処理の優先順位とサービスコールの不可分性-5
19	CPUロック状態	割込みハンドラ内でCPUロック解除状態にするための方法が記載されていること。	CPUロック状態-10

20		割込みハンドラから正しくリターンするための方法が記載されていること。	CPUロック状態-11
21	システム初期化手順	カーネルの初期化処理を呼び出す方法が記載されていること。	システム初期化手順-3
22		静的APIの処理中にエラーを検出した場合の扱いが記載されていること。	システム初期化手順-4
23		カーネルの管理外の割込みを禁止するかどうかが記載されていること。	システム初期化手順-5
24		初期化ルーチン内でサービスコールを呼び出せるか否か、呼び出せる場合にはどのようなサービスコールが呼び出せるか記載されていること。	システム初期化手順-6
25	オブジェクトの登録とその削除	カーネルに登録できるオブジェクトの最大数やID番号の範囲の指定方法が記載されていること。	オブジェクトの登録とその削除-2
26		自動割付けされるID番号の範囲の指定方法が記載されていること。	オブジェクトの登録とその削除-3
27	ext_tsk	エラーが発生した場合の処理が記載されていること。	ext_tsk-6
28	システム時刻管理	システム時刻更新機能がカーネル内部にある場合は、システム時刻更新方法が記載されていること。	システム時刻管理-2 isig_tim-2
29		記載通りに実行するとシステム時刻が正しく更新されること。	システム時刻管理-2
30	割込み管理機能	割込みハンドラの記述方法が記載されていて、正しく動作すること。	割込み管理機能-3
31		割込みハンドラ内でCPUロック解除状態にする方法が記載されていること。	割込み管理機能-5
32		CPUロックを解除した後に、割込みハンドラからリターンする方法が記載されていること。	割込み管理機能-6
33	DEF_INH	inhnoの具体的な意味が記載されていること。	DEF_INH-5
34	システム構成管理機能	CPU例外ハンドラの記述方法が記載されていて、正しく動作すること。	システム構成管理機能-2
35	DEF_EXC	excnoの具体的な意味が記載されていること。	DEF_EXC-4

第５節　ヘッダファイルによる確認の見方

以下に、μITRON4.0仕様で定義するヘッダファイルの内容確認の見方について説明する。

HED：

No.	区　分	テ　ス　ト　項　目		テスト手順参照
1	APIの構成要素	“kernel.h”から“itron.h”をインクルードしていること。		APIの構成要素-3
2	ヘッダファイル	ITRON仕様共通定義で規定されるデータ型、定数、マクロの定義などを含むヘッダファイルの名称が、μITRON4.0仕様書と一致していること。		ヘッダファイル-1
3		カーネル仕様で定められるサービスコールの宣言と、データ型、定数、マクロの定義などを含むヘッダファイルの名称が、μITRON4.0仕様書と一致していること。		サービスコールの返値とエラーコード-1 ヘッダファイル-2
4		カーネルのコンフィギュレータが生成する自動割付け結果ヘッダファイルの名称を“ kernel_id.h ”とすること。		ヘッダファイル-3
－	ITRON仕様 共通データ型 （itron.h）	型	整数／ポインタ／構造体	－
5		B	整数	ITRON仕様共通データ型-1
6		H	整数	ITRON仕様共通データ型-2
7		W	整数	ITRON仕様共通データ型-3
～				～
34	ITRON仕様 共通定数 （itron.h）	NULL	0で定義されていること	ITRON仕様共通定数-1
35		TRUE	1で定義されていること	ITRON仕様共通定数-2
36		FALSE	0で定義されていること	ITRON仕様共通定数-3

(1)　No.
ヘッダファイルの確認番号を示す。

(2)　区分
ヘッダファイルの確認で、確認項目をμITRON4.0仕様書の内容に添って分類したものである。

(3)　テスト項目
μITRON4.0仕様書に記載されている内容であり、テスト手順(プログラム)では確認できない項目である。したがって、検定を受けるカーネルメーカは、μITRON4.0仕様書が規定するヘッダファイルの名称、および当該ヘッダファイル内で記述すべき内容が明確に記載されていることを確認する。
データ型や定数についても、μITRON4.0仕様書が規定している通りに定義されていることを確認する。なお、検定対象のカーネルがTA_NULLをサポートしていなければ、省略することが可能である。
網掛けされてあるエラーコードは、実装依存や実装定義でエラーの検出を省略できることを示す。ただし、これらのエラーコードの検出を省略する場合は、省略する旨が製品マニュアルに記載されていること。

(4)　テスト項目参照
ヘッダファイルで確認すべき事項が、テスト項目のどの部分に記載されているかを示す。

第６節　ヘッダファイルによる確認の手順

HED：

No.	区　　分	テ　　ス　　ト　　項　　目		テスト項目参照
1	APIの構成要素	“kernel.h”から“itron.h”をインクルードしていること。		APIの構成要素-3
2	ヘッダファイル	ITRON仕様共通定義で規定されるデータ型、定数、マクロの定義などを含むヘッダファイルの名称が、“itron.h”であること。		ヘッダファイル-1
3		カーネル仕様で定められるサービスコールの宣言と、データ型、定数、マクロの定義などを含むヘッダファイルの名称が、“kernel.h”であること。		サービスコールの返値とエラーコード-1 ヘッダファイル-2
4		カーネルのコンフィギュレータが生成する自動割付け結果ヘッダファイルの名称を “ kernel_id.h ” とすること。		ヘッダファイル-3
―	ITRON仕様共通データ型（itron.h）	型	整数／ポインタ／構造体	―
5		B	整数	ITRON仕様共通データ型-1
6		H	整数	ITRON仕様共通データ型-2
7		W	整数	ITRON仕様共通データ型-3
8		UB	整数	ITRON仕様共通データ型-4
9		UH	整数	ITRON仕様共通データ型-5
10		UW	整数	ITRON仕様共通データ型-6
11		VB	定義されていること	ITRON仕様共通データ型-7
12		VH	定義されていること	ITRON仕様共通データ型-8
13		VW	定義されていること	ITRON仕様共通データ型-9
14		VP	ポインタ	ITRON仕様共通データ型-10
15		FP	ポインタ	ITRON仕様共通データ型-11
16		INT	整数	ITRON仕様共通データ型-12
17		UINT	整数	ITRON仕様共通データ型-13
18		BOOL	整数	ITRON仕様共通データ型-14
19		FN	整数	ITRON仕様共通データ型-15
20		ER	整数	ITRON仕様共通データ型-16
21		ID	整数	ITRON仕様共通データ型-17
22		ATR	整数	ITRON仕様共通データ型-18
23		STAT	整数	ITRON仕様共通データ型-19
24		MODE	整数	ITRON仕様共通データ型-20
25		PRI	整数	ITRON仕様共通データ型-21
26		SIZE	整数	ITRON仕様共通データ型-22
27		TMO	整数	ITRON仕様共通データ型-23
28		RELTIM	整数	ITRON仕様共通データ型-24
29		SYSTIM	整数	ITRON仕様共通データ型-25
30		VP_INT	整数またはポインタ	ITRON仕様共通データ型-26
31		ER_BOOL	整数	ITRON仕様共通データ型-27
32		ER_ID	整数	ITRON仕様共通データ型-28
33		ER_UINT	整数	ITRON仕様共通データ型-29
34	ITRON仕様共通定数（itron.h）	NULL	0で定義されていること	ITRON仕様共通定数-1
35		TRUE	1で定義されていること	ITRON仕様共通定数-2
36		FALSE	0で定義されていること	ITRON仕様共通定数-3
37		E_OK	0で定義されていること	ITRON仕様共通定数-4
38		E_SYS	-5で定義されていること	ITRON仕様共通定数-5
39		E_NOSPT	-9で定義されていること	ITRON仕様共通定数-6
40		E_RSFN	-10で定義されていること	ITRON仕様共通定数-7
41		E_RSATR	-11で定義されていること	ITRON仕様共通定数-8

42		E_PAR	-17で定義されていること	ITRON仕様共通定数-9
43		E_ID	-18で定義されていること	ITRON仕様共通定数-10
44		E_CTX	-25で定義されていること	ITRON仕様共通定数-11
45		E_MACV	-26で定義されていること	ITRON仕様共通定数-12
46		E_ILUSE	-28で定義されていること	ITRON仕様共通定数-13
47		E_NOMEM	-33で定義されていること	ITRON仕様共通定数-14
48		E_OBJ	-41で定義されていること	ITRON仕様共通定数-15
49		E_QOVR	-43で定義されていること	ITRON仕様共通定数-16
50		E_RLWAI	-49で定義されていること	ITRON仕様共通定数-17
51		E_TMOUT	-50で定義されていること	ITRON仕様共通定数-18
52		TA_NULL	0で定義されていること	ITRON仕様共通定数-19
53		TMO_POL	0で定義されていること	ITRON仕様共通定数-20
54		TMO_FEVR	-1で定義されていること	ITRON仕様共通定数-21
55	ITRON仕様 共通マクロ	“itron.h”にMERCDマクロが定義されていること。		ITRON仕様共通マクロ-1
56		“itron.h”にSERCDマクロが定義されていること。		ITRON仕様共通マクロ-2
57	カーネル 共通定義	TA_HLNG	0x00 で定義されていること	カーネル共通定義-1
58		TA_TFIFO	0x00 で定義されていること	カーネル共通定義-2
59		TSK_SELF	0 で定義されていること	カーネル共通定義-3
60		TSK_NONE	0 で定義されていること	カーネル共通定義-4
61	カーネル共通 構成定数	TMIN_TPRI	1 で定義されていること	カーネル共通構成定数-1
62		TMAX_TPRI	16以上 で定義されていること	カーネル共通構成定数-2
63		TKERNEL_MAKER	カーネルのメーカーコード	カーネル共通構成定数-3
64		TKERNEL_PRID	カーネルの識別番号	カーネル共通構成定数-4
65		TKERNEL_SPVER	ITRON仕様のバージョン番号	カーネル共通構成定数-5
66		TKERNEL_PRVER	カーネルのバージョン番号	カーネル共通構成定数-6
67	タスク管理機能	TMAX_ACTCNT	1以上で定義されていること	タスク管理機能-5
68	タスク付属 同期機能	TMAX_WUPCNT	1以上で定義されていること	タスク付属同期機能-1
69	セマフォ	TMAX_MAXSEM	65535以上で定義されていること	セマフォ-1
70	イベントフラグ	FLGPRN	16ビット以上で定義されていること	イベントフラグ-1
71		TBIT_FLGPTN	16ビット以上で定義されていること	イベントフラグ-2
72	システム 時刻管理	TIC_NUME	実装毎に定義されていること	システム時刻管理-4
73		TIC_DENO	実装毎に定義されていること	システム時刻管理-5
74	割込み管理機能	INHNO	実装毎に定義されていること	割込み管理機能-1
75		INTNO	実装毎に定義されていること	割込み管理機能-2
76	システム構成管理機能	EXCNO	実装毎に定義されていること	システム構成管理機能-1