TOPPERS/OSEK Communication 外部仕様書

Ver.1.04

2007/11/09

株式会社ヴィッツ　組込みソフトウェア開発部

TOPPERS/OSEK COM

Toyohashi Open Platform for Embedded Real-Time Systems/
OSEK COM

＜目次＞

1. 概要

本仕様書は、TOPPERS/OSEK Communication(今後は OSEK COM と記載)における外部仕様書である。
OSEK COM は、自動車制御装置における画一的な通信環境を確立させるものである。
OSEK COM は、以下の事を保証する。

- １ECU 内の振舞い
- 内部通信(電子制御装置間通信)
- 外部通信(ネットワーク内伝達ノード間通信)

※OSEK /VDX OS 仕様準拠カーネル（以下 OSEK OS）が未実装の場合も OSEK COM は動作可能である。もし OSEK OS が実装されない場合は、「タスク・イベント・ISR カテゴリ 2」に相当する機能を有するソフトウェアを用意する事が必要である。

OSEK OS の詳細内容は別紙『TOPPERS_OSEK カーネル外部仕様書』を参照とする。

1.1. 参考文献

次の文書は、本書の一部をなすものではないが、本書の内容を明確にするもの又は本文に関連して参考になるものである。

1. OSEK/VDX Operation System Specification 2.2.1
2. OSEK/VDX OSEK Implementation Language Specification 2.4.1
3. OSEK/VDX OSEK Communication Specification 3.0.2
4. OSEK/VDX Network Management Concept and Application Programming Interface Version 2.5.2

1.2. 要求仕様

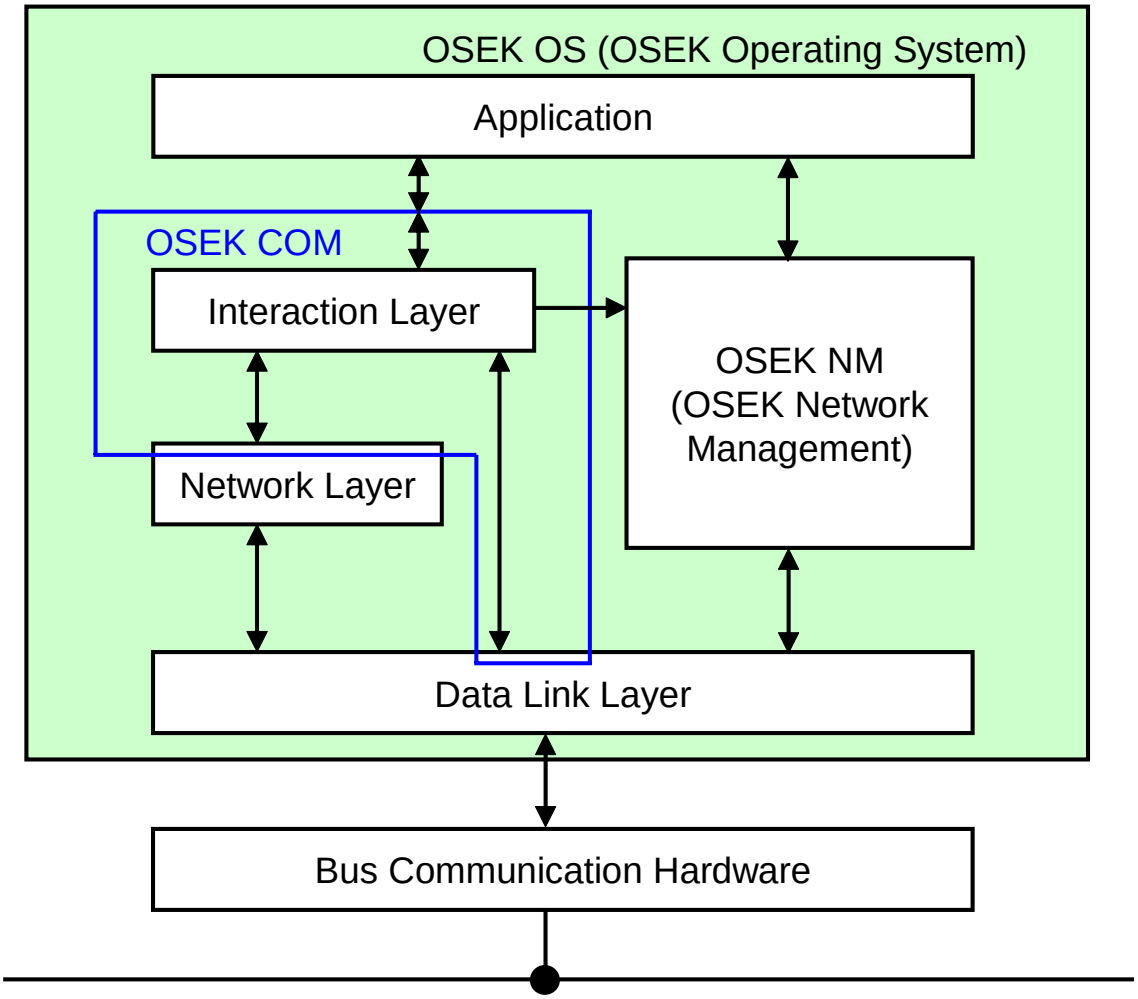
OSEK COM 仕様は、主に 4 つの要求仕様が存在する。

要求仕様	詳細内容
一般通信機能性	タスク/割込みサービスルーチン間のデータ転送を提供 (内部・外部通信を保証)[1]
アプリケーションの可搬性、再利用性、相互運用性	API によって異なる通信プロトコル等を網羅
スケーラビリティー	汎用的なハードウェアプラットフォーム上で動作可能
ネットワーク管理(NM)のサポート	Indirect NM[2]のサポートを行う

[1] サービスへのアクセスはAPIでのみ可能
[2] NMとはNetwork Management の略称であり、NMについて本仕様書内の詳細な解説は行わない

1.3. 通信概念

ここでは OSEK OS 構造における OSEK COM の位置付け及び、OSEK COM に含まれる各層の説明を記載する。



名称	意味
Interaction Layer	相互作用層のことであり、メッセージ転送(送受信操作)のためのサービスを含む API を提供 ※詳細は「2. Interaction Layer」で記載
Network Layer[3]	ネットワークの中から経路を選択しデータを中継
Data Link Layer[4]	物理的に直接結ばれた 2 点間でデータを誤りなく伝送するための制御

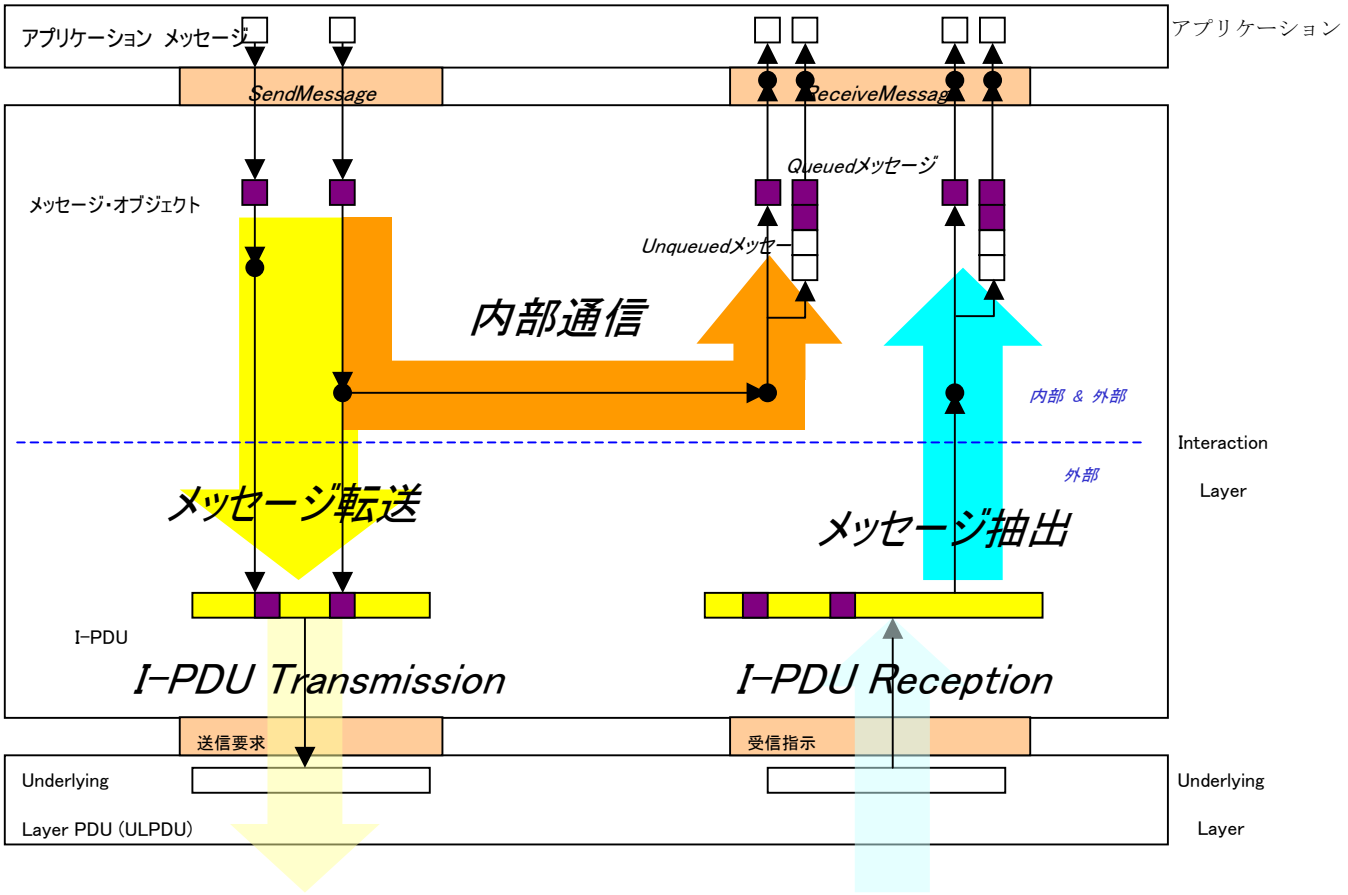
[3] Network Layerは『通信用語の基礎知識』より参照
[4] Data Link Layerは『通信用語の基礎知識』より参照

2. Interaction Layer

2.1. Interaction Layer の概要

2.1.1. 概要

Interaction Layer(以後 IL と記載)の全体の動作状況を下記に示す。



＜ＣＯＭの通信特徴＞

※OIL に関しては別紙『TOPPERS_OSEK カーネル SG 取扱説明書』を参照

内部通信	メッセージとメッセージ特性は OIL(OSEK Implementation Language)に記載し、静的に決定される。
	メッセージの内容と使用方法は OSEK COM に依存しない。
	アプリケーション側がデータを送信後、ILは即座にメッセージを受信バッファへ格納し、レシーバーが受信可能な状態にする。[5]
外部通信	多対多の通信構成が可能である。
	通信媒体（プロトコルスタック）に依存しない。[6]
	バイトオーダー、ビットオーダーが異なる相手との通信が可能である。[7]

[5] フィルタリングなどデータの補正は行わない
[6] 本バージョンではCAN（Control Area Network）のみをサポートする
[7] 本バージョンではビットオーダの変換をサポートしない

2.1.2. 通信概念

ここでは、通信規約の概念を記載する。

メッセージ操作	OSEK OS のタスク・割込みサービスルーチン(ISR)・通知のコールバック関数のみ可能
メッセージの認識方法	各メッセージにメッセージ識別子を割り振る
提供されている通信方法	m：n 通信(m＞0、n＞0)[8]
送信方法	Unqueued のみ
受信方法(データ保証回数)	Queued(1 回のみ)/Unqueued(何度でも)[9]
転送属性	トリガ転送属性 ペンディング転送属性
転送モード	直接転送モード 周期転送モード 混合転送モード
メッセージのサポート条件	静的なメッセージアドレスのみサポート[10][11]
データ長	固定データ長、可変データ長をサポート[12]
その他	デッドラインモニタリング機能[13]

2.1.3. メッセージ構成

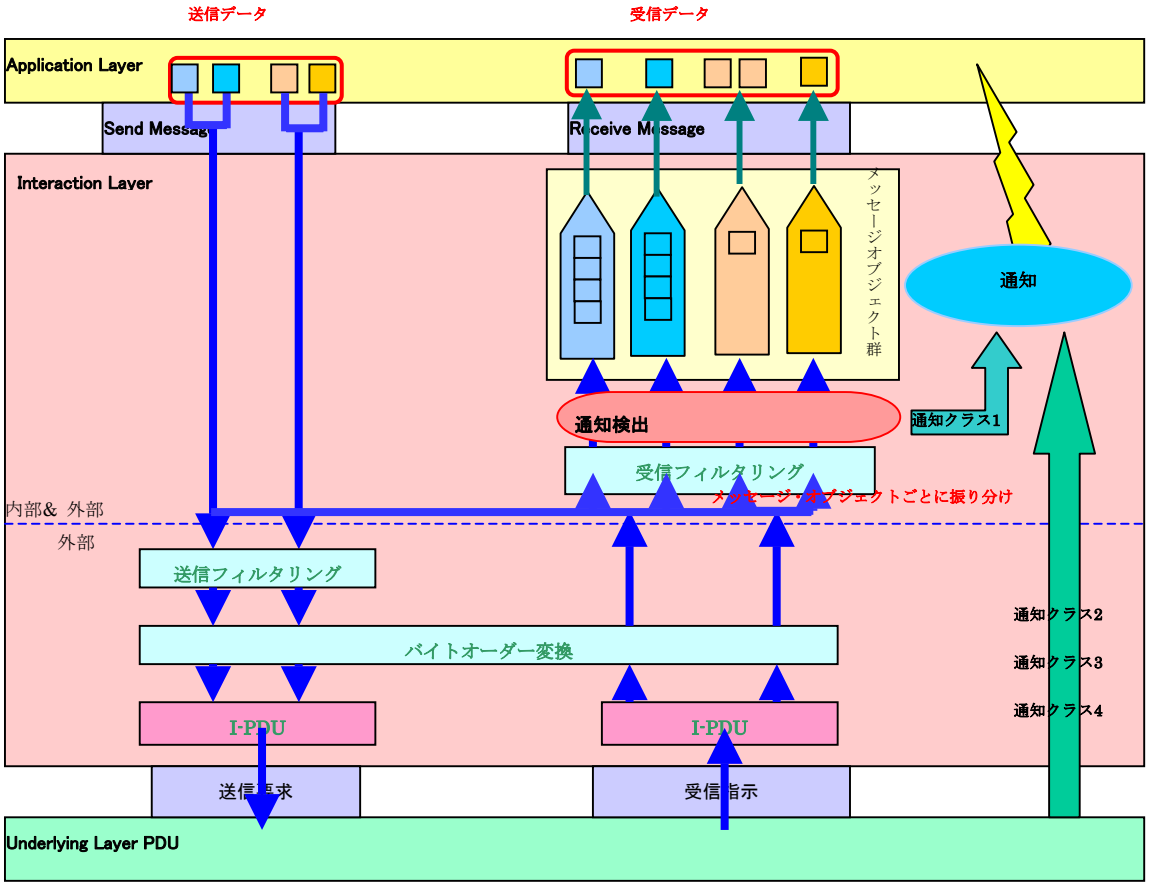
メッセージは、OILを元にシステムジェネレータ[14]で構成される。

[8] m:n通信に関しては【付録A】にて詳細内容記載
[9] 詳細は「2.2.2. Queued/Unqueued Message」で記載
[10] 無効なメッセージアドレスはサポートしない
[11] メッセージアドレスはOILにて静的に記載
[12] 本バージョンでは同一IPDU内の固定データ長メッセージと可変データ長メッセージの混在を許可しない。
[13] ペンディング転送属性は対象としない
[14] システムジェネレータに関しては別紙『TOPPERS_OSEKカーネルSG取扱説明書』を参照

2.2. メッセージ受信

ここではメッセージ受信について記載する。

概略を下図に示す。



2.2.1. 受信フィルタリングと通知

受信動作において、メッセージ・オブジェクトごとに振り分けるためにフィルタリングが存在する。

また通信クラス[15] 1 (メッセージが格納されたという情報)を通知する。

メッセージとメッセージオブジェクトの関係	1つのメッセージは 1 つ以上のメッセージ･オブジェクトにコピーされる[16]
アプリケーション側のメッセージ取得方法	提供されているAPI(ReceiveMessage/ReceiveDynamicMessage)[17]で取得

15 通知クラスの詳細説明は「2.7.1 通知クラス」で記載する

16 上記図参照

17 ReceiveMessage/ReceiveDynamicMessageの詳細内容は「4.システムサービス」で記載

2.2.2. Queued/Unqueued Message

メッセージ・オブジェクトごとに設定できるQueued/Unqueued[18]について記載する。

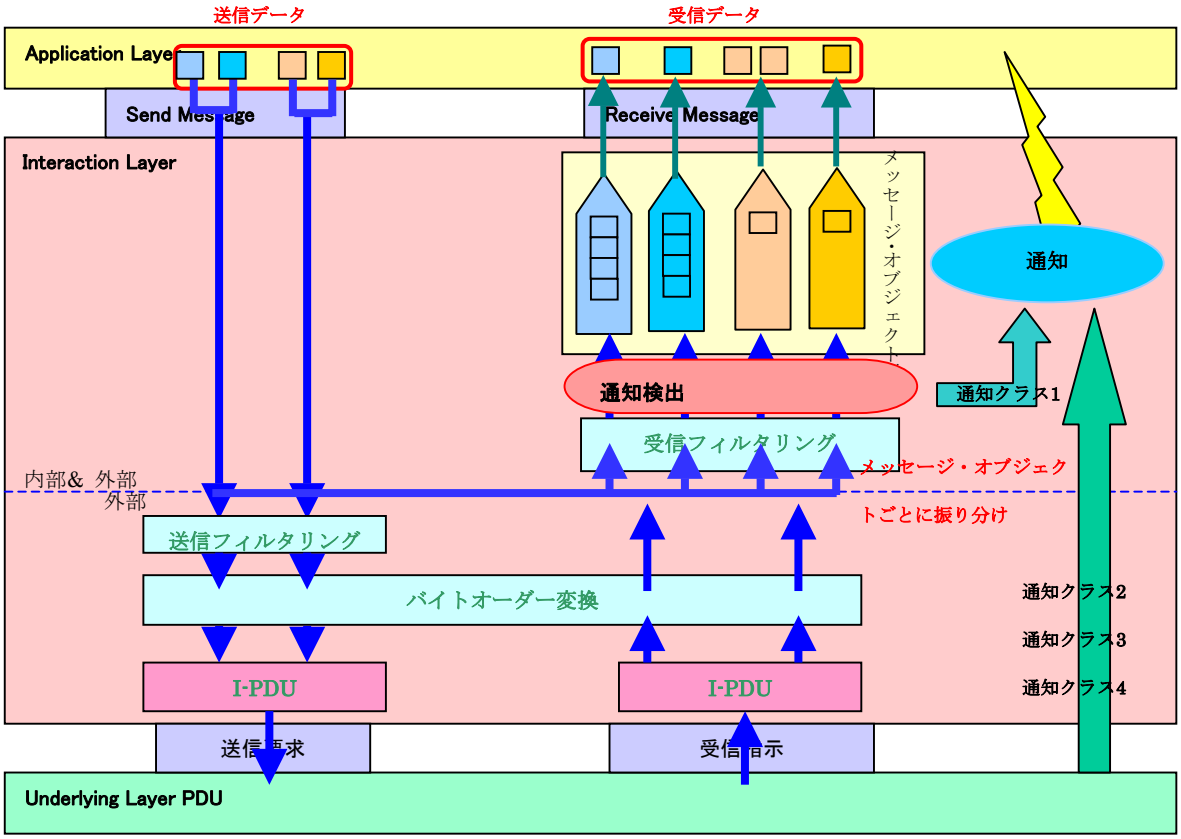
種類 / 項目	Queued	Unqueued
メッセージの扱い方法	FIFO 方式で扱う。	上書きで扱う。
空である時の振舞い	戻り値でエラーを返す。[19]	初期値を提供する。
満杯になった時の振舞い	キューに積まれない。[20]	上書きされる。
満杯でない時の振舞い	キューにストックされる。	上書きされる。[21]
M：N通信[22]	１回読むごとにメッセージを消費する。	メッセージは保持される。[23]
初期化処理	InitMessageで受信メッセージの数/データを０に設定[24]	OIL ファイルの中でメッセージの初期値を指定しない場合、StartCOM で値を０に初期化
		OILファイルの中でメッセージの初期値を指定する場合、StartCOMによって初期化[25]
		InitMessageで初期化を行う場合、StartCOMとStopCOMの間で初期化[26]

18 ここではReceiveMessage(API名)の振舞いを記載している
19 詳細は「4.システムサービス」で記載
20 最新のオブジェクトが削除
21 最新のオブジェクトに更新
22 m:n通信に関しては【付録A】にて詳細内容記載
23 重複読み込み可能
24 詳細は「4.システムサービス」で記載
25 符号なし整数値に限定
26 通常はStartCOMExtensionで使用

2.3. メッセージ送信

ここではメッセージ送信について記載する。

概略図を下図に示す。



2.3.1. 転送属性

外部通信では、送信しようとするメッセージを直ちに通信デバイスへ送るか、IPDUまでの設定とするかを使い分けることができる[27]。詳細を下記に示す。

名称	詳細内容
トリガ転送属性	IPDU に設定したメッセージを直ちに通信デバイスへ送る。
ペンディング転送属性	メッセージを一旦 IPDU に設定しておき、次にトリガ転送属性 のメッセージが IPDU に設定されるか、同じ IPDU 内の他の周期メッセージが送信機会となるまで保留される。

[27] 転送属性はIPDU以降の処理を規定しているため内部通信には存在しない

2.3.2. 転送モード

送信するメッセージをイベントとして送信するか、周期的に送信する、または両方の使い方を混在させることが可能である。詳細を下記に示す。

名称	詳細内容
直接転送モード	トリガ転送属性 のメッセージにより直ちに送信するモード。
周期転送モード	予め IPDU に設定しておいたメッセージを、周期送信の設定により一定間隔で送信するモード。
混合転送モード	直接転送モード と 周期転送モード の両方に対応するモード。両方の送信機会で IPDU の情報が送られる。

2.4. メッセージフィルタリング

外部通信の送信・受信及び、内部通信の受信の場合にメッセージのフィルタリングが実行される。フィルタリング対象のメッセージは、符号なしの整数型として解釈することができるメッセージのみである。ゼロ・レングスメッセージと可変長のメッセージでは、フィルタリングは行われない。

フィルタリングのアルゴリズム例を下表に示す。表中の値に関する説明を以下に示す。

new_value： メッセージの現在値
old_value： メッセージの最後(前回) の値
mask, x, min, max, period, offset： 定数(OIL に指定する)
occurrence： メッセージの発生回数

フィルタ名	アルゴリズム	解説
F_Always	TRUE	フィルターしない。常にメッセージを通す
F_Never	FALSE	常にメッセージを通さない
F_MaskedNewEqualsX	(new_value&mask) == x	new_value とマスク値の論理積が x ならば通す
F_MaskedNewDiffersX	(new_value&mask) != x	new_value とマスク値の論理積が x でなければ通す
F_NewIsEqual	new_value == old_value	new_value と old_value が同一ならば通す
F_NewIsDifferent	new_value != old_value	new_value と old_value が異なれば通す
F_MaskedNewEqualsMaskedOld	(new_value&mask) == (old_value&mask)	マスクした値が old_value とマスクした値と同一ならば通す
F_ MaskedNewDiffersMaskedOld	(new_value&mask) != (old_value&mask)	マスクした値が old_value とマスクした値と異なれば通す
F_NewIsWithin	min <= new_value <= max	範囲内ならば通す
F_NewIsOutside	(min > new_value) OR (new_value > max)	範囲外ならば通す
F_NewIsGreater	new_value > old_value	new_value が old_value より大きければ通す
F_NewIsLessOrEqual	new_value <= old_value	new_value が old_value 以下ならば通す
F_NewIsLess	new_value < old_value	new_value が old_value より小さいならば通す
F_NewIsGreaterOrEqual	new_value >= old_value	new_value が old_value 以上ならば通す
F_OneEveryN	occurrence % period == offset	指定回数に至ったら通す

2.5. バイトオーダー

IL はローカル CPU と UL 間でのバイトオーダー変換(ビッグエンディアンからリトルエンディアン・リトルエンディアンからビッグエンディアンへの変換)を行う。センダー側ではメッセージが IPDU 内に格納される前に、レシーバー側では IPDU からメッセージが取り出される際にバイトオーダー変換が行われる。

内部通信及び、可変データ長メッセージではバイトオーダー変換は行われない。

バイトオーダーの図を以下に示す。

リトルエンディアン

	7	6	5	4	3	2	1	0
0								
1	←	LSB						
2	←							
3						MSB	←	
4								

ビッグエンディアン

	7	6	5	4	3	2	1	0
0								
1						MSB	←	
2	←							
3	←	LSB						
4								

← メッセージの格納の向き
（低ビットから高ビットへ）

2.6. デッドラインモニタリング

送受信するメッセージに時間制約を設けて監視する機能である。尚、これらのメッセージは外部通信にのみ存在し、内部通信には適用されない。詳細を下記に示す。

名称	詳細内容
受信デッドラインモニタリング	主に周期メッセージのメッセージ到着時間を監視する。 これに用いられるタイマは、時間内に UL 層から入力される次のメッセージの到着により、IL 層によりリセットされる。 規定時間以内にメッセージの到着がない場合、タイムアウトが発生し、監視タイマは強制的にリセットされる。 規定時間以内にメッセージが到着した場合には、次のメッセージ到着を監視するために監視タイマが自動的に設定される。 混合転送モード や 直接転送モード のメッセージにも適用される。
送信デッドラインモニタリング	IL 層の IPDU が、規定時間以内に UL 層へ送られるかどうかを監視する。 通信デバイスが、時刻に同期してメッセージを送信できているかどうかを監視することを目的としている。

2.7. 通知

ここでは通知・通知クラスに関して記載する。送受信処理の状態により４つの通知クラスを用いてアプリケーション側に通知をするメカニズムである。通知を行う単位としては、メッセージ・オブジェクト単位である。

2.7.1. 通知クラス

通知する方法として４つの通知クラスがあるので下記に記載する。

名称	処理内容	詳細内容
通知クラス１	受信処理	バッファに正常に格納された直後通知をする。[28]
通知クラス３		受信エラー、デッドラインモニタリングなどで異常を検知された時通知する。
通知クラス２	送信処理	外部通信が正常に送信した直後通知をする。
通知クラス４		送信エラー、デッドラインモニタリングなどで異常を検知された時通知する。

[28] キューがあふれている時は通知クラス４が発生する

2.7.2. 通知メカニズム

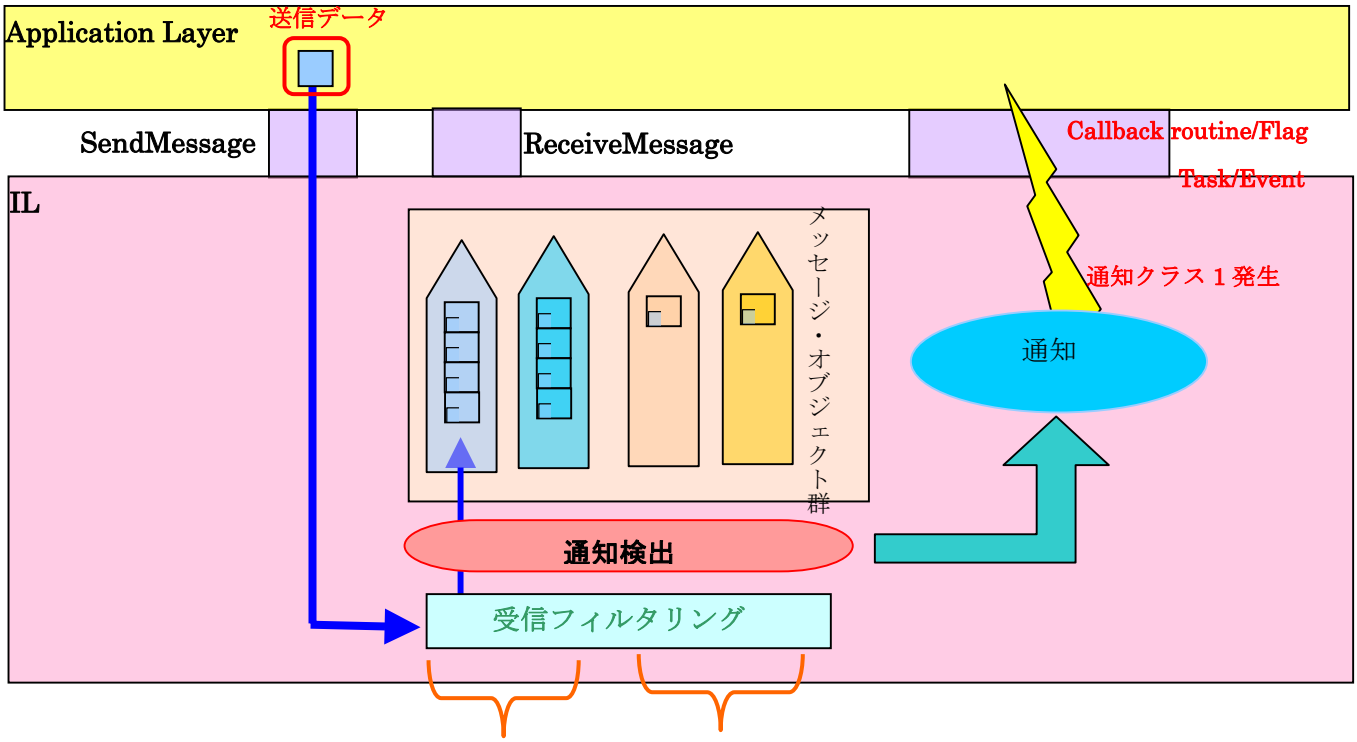
ここでは通知クラスでアプリケーション側に提供するメカニズムを下記に記載する。

通知メカニズム	内容
Callback routine[29]	ILでアプリケーション側が用意するCallback routineを呼ぶ。[30][31]
Flag	API : ReadFlag_<Flag>でアプリケーション側がチェックできるフラグをONする。[32]
Task	IL で、アプリケーション側のタスクを起動する。
Event	IL でイベントを設定する。

下記に内部通信で関連する通知の状態図例を記載する。

[29] CallBack routineに関しては【5.【付録Ａ】】にて詳細内容記載
[30] 引数・戻り値が存在しない
[31] 優先度レベルは割込み・タスクレベルと同等
[32] フラグのリセットはAPI：ResetFlag_<Flag>でアプリケーション側に提供

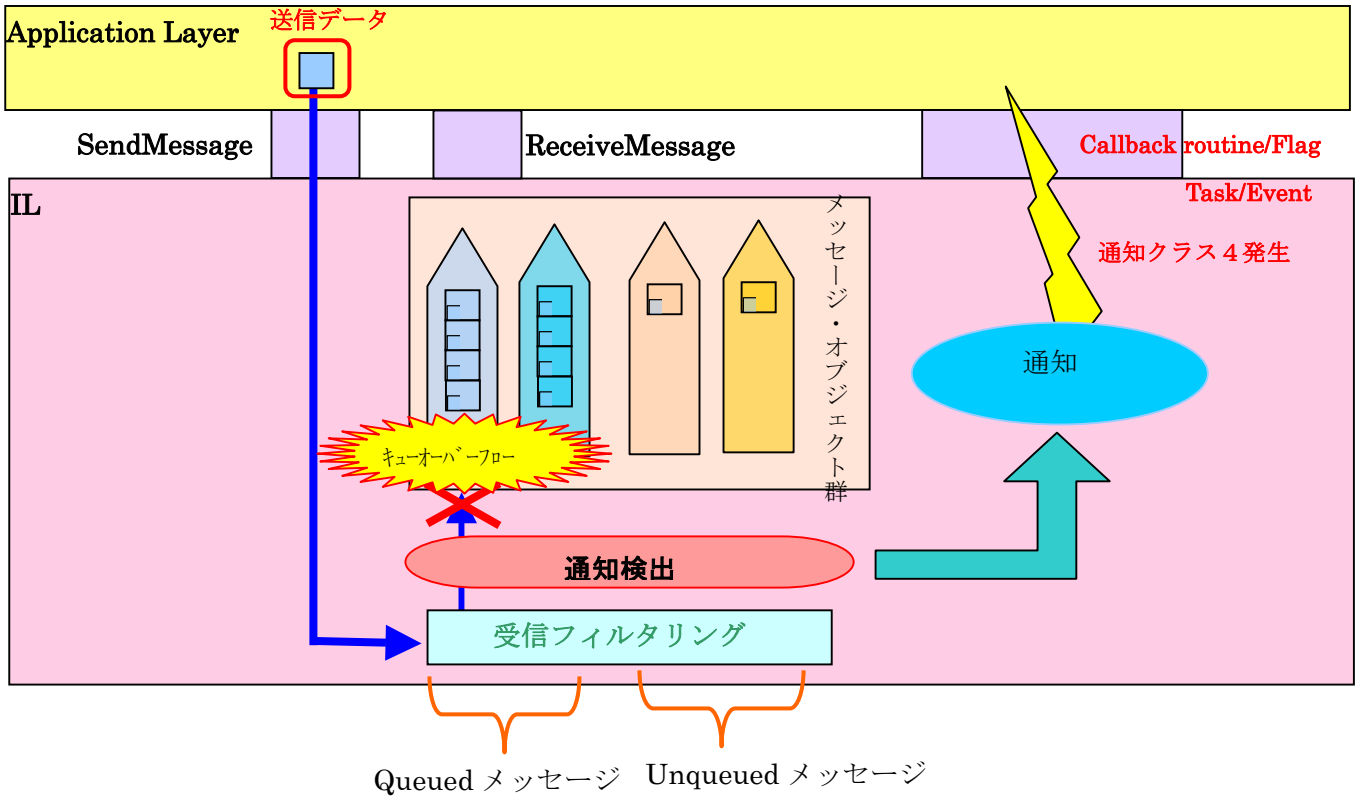
＜通知クラス１：正常に受信できた場合＞



Queued メッセージ　Unqueued メッセージ

※Unqueued メッセージに関してデータは上書きなので必ず通知クラス１が発生する
→通知クラス４は発生しない

＜通知クラス４：キューイングがあふれた場合＞



2.8. 通信システム管理

2.8.1. 初期化/停止

OSEK COM の通信を開始／終了させるためには、いくつかのシステムサービスを利用しなければならない。そのシステムサービスを下記に示す。

システムサービス名	内容
StartCOM	データ領域を初期化
	メッセージを初期化
	OSEK COM モジュールの起動
StopCOM	リソースの解放
	通信を不整合が起こらないように停止
StartPeriodic	周期メッセージの送受信の開始
StopPeriodic	周期メッセージの送受信の停止
InitMessage	任意の値でメッセージを初期化[33][34]

各システムサービスの詳細は、「4. システムサービス」で記載する。

[33] OILファイルで値を規定(符号なし整数値のみ有効)

[34] OILに記載されていなければデフォルトで(キューイング：メッセージ数を０／アンキューイング：値を０)初期化

2.8.2. エラー処理

2.8.2.1. エラーの種類

OSEK COM で使用しているエラーには大きく分けて 2 種類あり、下記に示す。

エラーの種類	内容
Application Errors	動作制御には影響を与えず、復旧可能なエラーである。
	ステータス情報でエラーを返すので、エラー種別によりユーザが復帰処理等を行う。
Fatal Errors	動作制御に影響を与え、復旧不可能なエラーである。
	COM 内部でシステムシャットダウンを行う。

また Application Error には 2 種類のレベルが存在する。

エラーの種類[35]	内容	長所	短所
標準エラー	リリース時にサポートするエラー情報	少ない時間・メモリで動作可能	情報が少ない
拡張エラー	開発段階(デバッグ)時にサポートするエラー情報	細かくチェックが可能	より多くの実行時間とメモリが必要

[35] どちらのエラーを出力するかは静的(OIL)に設定する事が可能

2.8.2.2. フック処理

特徴	OSEK COM のサービスルーチンで【E_OK】以外を返す時に呼ばれる
	エラーフック内で再度エラーが発生してもエラーフックは呼ばれない →再帰呼出は起きない
	IL によって呼び出される
	詳細設定は実装依存
	OIL によって構成が可能
備考	カテゴリ 2 の割込みはフック割込みを禁止した状態で実行[36]

[36] 詳細は別紙『TOPPERS/OSEK カーネル外部仕様書』を参照

2.8.2.3. エラー管理

COMErrorHook での効果的なエラー処理を許すため、ユーザは補足的な情報にアクセスできる。

マクロ名	内容[37]
COMErrorGetServiceId_xxx[38]	エラーを起こしたサービスを示す識別子(COMServiceIdType 型)を取得する事が可能
COMError_Name1[39]_Name2[40]	エラーを起こしたサービスのパラメータにアクセス

[37] 詳細は「4.システムサービス」に記載
[38] xxx:サービスの識別子
[39] Name1:サービス名
[40] Name2:パラメータ名

2.9. コールアウト

コールアウトは、IL の振舞いをカスタマイズする目的で、アプリケーションで実装する関数である。

本機能は静的に構成する必要があり、実行時に動作を変更することはできない。

コールアウトは、メッセージまたは IPDU の受信時と送信時に呼ばれ、データの操作が可能な機会を得る。その際に戻り値を返すことができ、戻り値はメッセージまたは I-PDU の処理を継続するか、放棄するかどうかを示す。

・送受信時のコールアウト種別

CPU	アプリケーションに属するメッセージ単位
Network	ネットワークに属するメッセージ単位
I-PDU	I-PDU 単位

3. コンフォーマンスクラス

COM では様々なニーズに合うようにいくつかのレベル(Communication Conformance Classes(CCC))を提供している。

下記に各コンフォーマンスクラス[41] [42]と特徴を記載する。

コンフォーマンスクラス名	特徴
CCCA	内部通信のみ提供(外部通信は提供しない)
	キューイングはサポートしない
	メッセージステータス情報はサポートしない
	通知クラス 1 をサポート(フラグ通知機構を除く)
CCCB	CCCA の内容をサポート
	キューイングをサポートする
	メッセージステータス情報をサポートする
CCC0	内部通信と外部通信を提供
	キューイングはサポートしない
	メッセージステータス情報はサポートしない
	通知クラス 2 をサポート
	バイトオーダ変換と Direct Transmission Mode をサポート
CCC1	COM で提供されている全ての機能をサポート

[41] 本バージョンではコンフォーマンスクラスのCCC 1 をサポートする
[42] 本バージョンではCCC1 以外のクラスへの切り替えをサポートしていない

各コンフォーマンスクラスが提供している機能一覧を記載する。

機能	CCCA	CCCB	CCC0	CCC1
Unqueued messages	○	○	○	○
Notification Class 1	○[43]	○	○	○
Queued messages		○		○
Message status information		○		○
External communication			○	○
Triggered Transfer Property			○	○
Notification Class 2			○	○
Byte ordering			○	○
Direct Transmission Mode			○	○
Filtering				○
Pending Transfer Property				○
Zero-length messages				○
Dynamic-length messages				○
Periodic Transmission Mode				○
Mixed Transmission Mode				○
Minimum delay time				○
Deadline Monitoring				○
Notification Class 3				○
Notification Class 4				○
Callouts				○

[43] CCCAではNotificationのフラグ機構をサポートしていない

4. システムサービス

ステータスコードの一覧を下記に示す。

ステータスコード[44]	内容
E_OK	正常終了
E_COM_ID	識別子が無効(範囲外)
E_COM_LENGTH	データ長の範囲をオーバー
E_COM_LIMIT	メッセージキューのオーバーフロー
E_COM_NOMSG	メッセージキューが空
E_COM_SYS_STARTCOMEXT	StartCOMExtension でエラー
E_COM_SYS_CALLEVEL	COM コールレベルの不具合
E_COM_SYS_INTERRUPT	割込禁止中に CALL されたときのエラー
E_COM_SYS_ERR	内部エラー
E_IL_FILTERED	フィルタによって除外
E_IL_CALLOUTFALSE	コールアウトの戻りが不正

[44] 独自のステータスコードには「**E_COM_SYS_*****」として定義することを規定している

4.1. Interface to OSEK Indirect NM

TOPPERS/OSEK NM における indirect メッセージを実装するため、COM から通知をする機構である。ただし、本バージョンでは Indirect NM に関する内容は対応していない。

4.2. 型情報

COM における型情報を下記の表で記載する。

型名	サイズ	備考
StatusType	UINT8	ステータス情報
MessageIdentifier	UINT16	メッセージID
ApplicationDataRef	-	データ格納場所へのポインタ
COMLengthType	UINT8	データ長
LengthRef	-	COMLengthType型ポインタ
FlagValue	UINT8	フラグの値
COMApplicationModeType	UINT8	COMアプリケーションモード
COMShutdownModeType	UINT8	COMシャットダウンモード
CalloutReturnType	UINT8	Calloutからの返り値
COMServiceIdType	UINT8	COMのサービスAPIのID

4.3. COM モジュールの起動/停止

4.3.1. StartCOM

構文		StatusType StartCOM (COMApplicationModeType <Mode>)
パラメータ(in)		Mode：COM アプリケーションモード <使用可能なモード：OIL にて指定されたモード>
パラメータ(out)		なし
記述		指定したアプリケーションモードで COM の初期化と COM の制御を開始する。
特性		TOPPERS/OSEK カーネルでは、StartCOM はタスクから呼ばれる。StartCOM は、アプリケーション関数の StartCOMExtension を呼ぶ。StartCOM では、メッセージの周期的な送信ができないため、必要な場合は StartCOMExtension から StartPeriodic を呼ぶようにする。StartCOMExtension によって返されたステータスコードが E_OK でない場合は、StartCOM の戻り値として返す。
戻り値	標準	E_OK：正常
		E_COM_SYS_INTERRUPT：割込み禁止中に呼び出し(※1)
		E_COM_SYS_CALLEVEL：COM 起動済み
		E_COM_SYS_ERR：内部エラー
	拡張	E_COM_ID：<Mode>が範囲外
備考		COM によって使用されるハードウェアおよび下位レベルのリソースが初期化される処理を行う前に StartCOM を呼ぶと不確定の振舞い結果となる。 ※1) 割込みチェック機能はユーザによる拡張実装を想定

4.3.2. StopCOM

構文		StatusType StopCOM (COMShutdownModeType <Mode>)
パラメータ(in)		Mode：COM シャットダウンモード <使用可能なモード：COM_SHUTDOWN>
パラメータ(out)		なし
記述		直ちに全ての COM の制御を中止させる。
特性		途中の処理があっても、完了するのを待たずにシャットダウン処理（UL の停止、周期送信停止）が直ちに実行される。
戻り値	標準	E_OK：正常
		E_COM_SYS_INTERRUPT：割込み禁止中に呼び出し(※1)
		E_COM_SYS_CALLEVEL：COM 起動前
		E_COM_SYS_ERR：内部エラー
	拡張	E_COM_ID：<Mode>が範囲外
備考		バッファのクリア、管理情報の初期化は行わない。 StopCOM を呼ぶ前にリソースを解放しなかった時の振舞いは不確定であり、データを損失する可能性がある。 ※1）割込みチェック機能はユーザによる拡張実装を想定

4.4. COM アプリケーションモードの取得

4.4.1. GetCOMApplicationMode

構文	COMApplicationModeType GetCOMApplicationMode (void)
パラメータ(in)	なし
パラメータ(out)	なし
記述	現在の COM アプリケーションモードを返す。
特性	StartCOM が呼ばれる前に、GetCOMApplicationMode が呼ばれる場合の振舞いは不確定である。
戻り値	現在の COM アプリケーションモード。
備考	主にモード依存のアプリケーション処理を書くために使用する。

4.5. メッセージ初期化

4.5.1. InitMessage

構文		StatusType InitMessage (MessageIdentifier <Message>,ApplicationDataRef <DataRef>)
パラメータ(in)		Message：メッセージ識別子(C 識別子)
		DataRef：アプリケーションのメッセージ初期化処理データを参照
パラメータ(out)		なし
記述		<DataRef>パラメータによって参照されるアプリケーションデータの<Message>で特定されたメッセージ・オブジェクトを初期化する。
特性		デフォルトの初期化を変更するための関数で、StartCOMExtension から実行可能である。 動的な長さのメッセージについては、最大メッセージ長のメッセージで初期化される。 IPDU 内の送信メッセージを初期化した場合、バイトオーダー変換、CPU-order および Network-order MessageCallouts が呼ばれる。 InitMessage の引数によって、フィルタリングの old_value も設定される。
戻り値	標準	E_OK：正常
		E_COM_SYS_ERR：内部エラー
	拡張	E_COM_ID：対象外のメッセージ種別で呼び出された
備考		-

4.6. COM 周期送信処理

4.6.1. StartPeriodic

構文		StatusType StartPeriodic (void)
パラメータ(in)		なし
パラメータ(out)		なし
記述		周期的送信モードまたは混合送信モードを使用して、メッセージの周期的な伝達を開始する。
特性		本 API が呼ばれると、周期的な送信を完全に再初期化し、再度開始する。
戻り値	標準	E_OK：正常
		E_COM_SYS_INTERRUPT：割込み禁止中に呼び出し(※1)
		E_COM_SYS_CALLEVEL：COM 起動前
		E_COM_SYS_ERR：内部エラー
	拡張	-
備考		StartCOM 後であること、データが一度は初期化されていることが必要である。 ※1) 割込みチェック機能はユーザによる拡張実装を想定

4.6.2. StopPeriodic

構文		StatusType StopPeriodic (void)
パラメータ(in)		なし
パラメータ(out)		なし
記述		周期的送信モードまたは混合送信モードでのメッセージの周期的な伝達を停止する。
特性		StopPeriodic が呼ばれると、周期的な送信を完全に停止する。
戻り値	標準	E_OK：正常
		E_COM_SYS_INTERRUPT：割込み禁止中に呼び出し(※1)
		E_COM_SYS_CALLEVEL：COM 起動前
		E_COM_SYS_ERR：内部エラー
	拡張	-
備考		現状の実装では、内部エラーは存在しないが、他の API と共通にするために残してある。 ※1) 割込みチェック機能はユーザによる拡張実装を想定

4.7. アプリケーションへの通知手段

4.7.1. ReadFlag

構文	FlagValue ReadFlag_<Flag>(void)
パラメータ(in)	なし
パラメータ(out)	なし
記述	<Flag>がセットされているかどうかを参照する
特性	フラグ名が「ABC」の時、フラグを参照するマクロ名は「ReadFlag_ABC()」
戻り値	COM_TRUE：<Flag>がセットされている
	COM_FALSE：<Flag>がセットされていない
備考	メッセージの送受信におけるアプリケーションへの通知手段として FLAG が指定されている場合のみ本サービスは使用できる。

4.7.2. ResetFlag

構文	void ResetFlag_<Flag>(void)
パラメータ(in)	なし
パラメータ(out)	なし
記述	指定したフラグをリセットする
特性	フラグ名が「ABC」の時、フラグをリセットするマクロ名は「ResetFlag_ABC( )」となる。
戻り値	-
備考	メッセージの送受信におけるアプリケーションへの通知手段として FLAG が指定されている場合のみ本サービスは使用できる。

4.8. メッセージの送受信

4.8.1. SendMessage

構文		StatusType SendMessage (MessageIdentifier <Message>,ApplicationDataRef <DataRef>)
パラメータ(in)		Message：メッセージ識別子(C 識別子)
		DataRef：送信されるアプリケーションメッセージデータへの参照
パラメータ(out)		なし
記述		<DataRef>パラメータによって参照されるアプリケーションメッセージで、<Message>によって特定されるメッセージ・オブジェクトを更新する。
特性		＜内部通信＞ メッセージは IL の受信部に送信される。
		＜外部通信＞ <Message>が「Triggered Transfer Property」の転送属性を持っている場合、更新してメッセージに結合したI-PDUの送信を直ちに行う。(ただし、I-PDUの中に転送モード「Periodic Transmission Mode」でメッセージを詰める時は直ちに送信は行われず、周期送信により送信される。) <Message>が「Pending Transfer Property」の転送属性を持っている場合、送信は更新によって行われない。
戻り値	標準	E_OK：正常
		E_COM_SYS_INTERRUPT：割込禁止中にコール(※1)
		E_COM_SYS_CALLEVEL：COM 起動前
		E_COM_SYS_ERR：内部エラー
	拡張	E_COM_ID：<Message>が範囲外
		E_IL_FILTERED：フィルタによって除外
		E_IL_CALLOUTFALSE：コールアウトの戻りが不正
備考		Notification(通知クラス 2 と 4)で設定されたフラグをクリアする。 ※1) 割込みチェック機能はユーザによる拡張実装を想定

4.8.2. ReceiveMessage

構文		StatusType ReceiveMessage (MessageIdentifier <Message>,ApplicationDataRef <DataRef>)
パラメータ(in)		Message：メッセージ識別子(C 識別子)
パラメータ(out)		DataRef：受信データを保存するアプリケーションのメッセージ領域の参照
記述		<Message>によって特定されたメッセージ・オブジェクト中のデータを取得し、<DataRef>の中に格納する。
特性		メッセージオーバーフローでもメッセージを受信し続ける。そのため、オーバーフロー状態をクリアする機能がある。
戻り値	標準	E_OK：メッセージデータ(キューされている・キューされていない場合のどちらかの)が利用可能
		E_COM_NOMSG：キューされているメッセージが空の場合
		E_COM_LIMIT：メッセージキューがオーバーフロー
		E_COM_SYS_INTERRUPT：割込禁止中にコール(※1)
		E_COM_SYS_CALLEVEL：COM 起動前
		E_COM_SYS_ERR：内部エラー
	拡張	E_COM_ID：<Message>が範囲外
備考		Notification(通知クラス 1 と 3)で設定されたフラグをクリアする。 ※1) 割込みチェック機能はユーザによる拡張実装を想定

4.8.3. SendDynamicMessage

構文		StatusType SendDynamicMessage (MessageIdentifier <Message>,ApplicationDataRef <DataRef>, LengthRef <LengthRef>)
パラメータ(in)		Message：メッセージ識別子(C 識別子)
		DataRef：送信されるアプリケーションメッセージデータの参照
		LengthRef：メッセージにデータの長さを含んでいる値の参照
パラメータ(out)		なし
記述		<DataRef>パラメータによって参照されるアプリケーションメッセージで、<Message>によって特定されるメッセージ・オブジェクトを更新する。 <Message>が「Triggered Transfer Property」の転送属性を持っている場合、更新してメッセージに結合したI-PDUの送信を直ちに行う。(ただし、I-PDUの中に転送モード「Periodic Transmission Mode」でメッセージを詰める時は直ちに転送は行われず、周期送信により送信される。) <Message>が「Pending Transfer Property」の転送属性を持っている場合、送信は更新によって行われない。
特性		本サービスはキューされていないメッセージだけと共に利用することができる。 また、本サービスを外部通信だけに提供する。
戻り値	標準	E_OK：正常
		E_COM_SYS_INTERRUPT：割込禁止中にコール(※1)
		E_COM_SYS_CALLEVEL：COM 起動前
		E_COM_SYS_ERR：内部エラー
	拡張	E_COM_ID：<Message>が範囲外
		E_COM_LENGTH：<LengthRef>が示す値が<Message>で定義された最大の長さ範囲内に無い場合
備考		Notification(通知クラス 2 と 4)で設定されたフラグをクリアする。 ※1) 割込みチェック機能はユーザによる拡張実装を想定

4.8.4. ReceiveDynamicMessage

構文		StatusType ReceiveDynamicMessage (MessageIdentifier <Message>,ApplicationDataRef <DataRef>, LengthRef <LengthRef>)
パラメータ(in)		Message：メッセージ識別子(C 識別子)
パラメータ(out)		DataRef：送信されるアプリケーションメッセージデータの参照
		LengthRef：メッセージにデータの長さを含んでいる値の参照
記述		<Message>によって特定されたメッセージ・オブジェクト中のデータを取得し、<DataRef>の中に格納する。 受信メッセージデータの長さは<LengthRef>によって値を参照する。
特性		本サービスは、外部通信だけに提供される。
戻り値	標準	E_OK：メッセージデータ(キューされていない場合)が利用可能
		E_COM_SYS_INTERRUPT：割込禁止中にコール(※1)
		E_COM_SYS_CALLEVEL：COM 起動前
	拡張	E_COM_ID：<Message>が範囲外
備考		Notification(通知クラス 1 と 3)で設定されたフラグをクリアする。 ※1) 割込みチェック機能はユーザによる拡張実装を想定

4.8.5. SendZeroMessage

構文		StatusType SendZeroMessage (MessageIdentifier <Message>)
パラメータ(in)		Message：ゼロレングスメッセージ(C 識別子)のメッセージ識別子
パラメータ(out)		なし
記述		＜外部通信＞ SendZeroMessage サービスは、ゼロレングスメッセージ<Message>と結合した I-PDU の送信を直ちに行う。 （ただし、I-PDU の中に転送モード「Periodic Transmission Mode」でメッセージを詰める時は直ちに転送は行われず、周期送信により送信される。）
		＜内部通信＞ メッセージ<Message>は通知のために IL の受信部に送信される。
特性		-
戻り値	標準	E_OK：正常
		E_COM_SYS_INTERRUPT：割込禁止中にコール(※1)
		E_COM_SYS_CALLEVEL：COM 起動前
		E_COM_SYS_ERR：内部エラー
	拡張	E_COM_ID：<Message>が範囲外
備考		Notification(通知クラス 2 と 4)で設定されたフラグをクリアする。 ※1) 割込みチェック機能はユーザによる拡張実装を想定

4.9. メッセージのステータス取得

4.9.1. GetMessageStatus

構文		StatusType GetMessageStatus (MessageIdentifier <Message>)
パラメータ(in)		Message：メッセージ識別子(C 識別子)
パラメータ(out)		なし
記述		メッセージ・オブジェクト<Message>の現在のステータスを返す。
特性		-
戻り値	標準	E_COM_NOMSG：メッセージキューが空の場合
		E_COM_LIMIT：メッセージキューがオーバーフロー
		E_OK：上記エラーが発生した場合や、既にエラーが発生している場合以外
	拡張	E_COM_ID：<Message>が範囲外
備考		-

4.10. マクロ

4.10.1. COMErrorGetServiceId マクロ

構文	COMErrorGetServiceId ()
パラメータ(in)	なし
パラメータ(out)	なし
記述	COMErrorGetServiceId は、COMErrorHook から呼ぶことを想定しており、エラーが発生した COM サービスの識別子を返す。 COMServiceId_<ServiceName> ※<ServiceName>：サービスの名前
特性	-
戻り値	エラーが発生した COM サービスの識別子。 COMServiceId_StartCOM COMServiceId_StopCOM COMServiceId_GetCOMApplicationMode COMServiceId_InitMessage COMServiceId_StartPeriodic COMServiceId_StopPeriodic COMServiceId_ReadFlag COMServiceId_ResetFlag COMServiceId_SendMessage COMServiceId_ReceiveMessage COMServiceId_SendDynamicMessage COMServiceId_ReceiveDynamicMessage COMServiceId_SendZeroMessage COMServiceId_GetMessageStatus COMServiceId_COMErrorGetServiceId COMServiceId_StartCOMExtension COMServiceId_COMCallout COMServiceId_CallCOMSndMsgDRV COMServiceId_CallCOMRcvMsgDRV
備考	注意：COMErrorHook から呼ばれない場合、値は未定義である。

4.10.2. COMError_Name1_Name2 マクロ

構文	COMError_Name1_Name2() ※Name1：サービス(例えば SendMessage)の名前 Name2：パラメーター(例えば DataRef)の名前
パラメータ(in)	なし
パラメータ(out)	なし
記述	COMErrorHook を呼んだ COM サービスのパラメータにアクセスすることに使用されるマクロの名前のためのパターンである。 COMErrorHook 内で使用する。
特性	-
戻り値	COM サービスのパラメータ
備考	以下のマクロが存在する。 COMError_StartCOM_Mode() COMError_StopCOM_Mode() COMError_InitMessage_Message() COMError_InitMessage_DataRef() COMError_SendMessage_Message() COMError_SendMessage_DataRef() COMError_ReceiveMessage_Message() COMError_ReceiveMessage_DataRef() COMError_SendDynamicMessage_Message() COMError_SendDynamicMessage_DataRef() COMError_SendDynamicMessage_LengthRef() COMError_ReceiveDynamicMessage_Message() COMError_ReceiveDynamicMessage_DataRef() COMError_ReceiveDynamicMessage_LengthRef() COMError_SendZeroMessage_Message() COMError_GetMessageStatus_Message()

4.10.3. COMCallout_SendDataPointer マクロ

構文	COMCallout_SendDataPointer
パラメータ(in)	なし
パラメータ(out)	なし
記述	コールアウト時に送信データへのポインタをユーザに公開する。
特性	-
戻り値	送信データへのポインタ
備考	-

4.10.4. COMCallout_ReceiveDataPointer マクロ

構文	COMCallout_ReceiveDataPointer
パラメータ(in)	なし
パラメータ(out)	なし
記述	コールアウト時に受信データへのポインタをユーザに公開する。
特性	-
戻り値	受信データへのポインタ
備考	-

4.11. アプリケーションサービス

4.11.1. StartCOMExtension

構文		StatusType StartCOMExtension (void)
パラメータ(in)		なし
パラメータ(out)		なし
記述		StartCOMExtension はアプリケーションで提供され、StartCOM ルーチンの終わりに呼ばれる。
特性		初期化機能(例えば InitMessage)や、追加スタートアップ機能(例えば StartPeriodic)など、スタートアップルーチンを拡張するために使用する。
戻り値	標準	E_OK：正常
		エラー：ユーザによる実装独自のステータスコード
	拡張	-
備考		StartCOM の終わりに COM によって呼ばれる。

4.11.2. COMCallout

構文	COMCallout(CalloutRoutineName)
パラメータ(in)	なし
パラメータ(out)	なし
記述	CalloutRoutineName はアプリケーションによって提供され、COM の実行で呼ばれる。
特性	アプリケーション関連の機能(例えば gatewaying)を備えた COM 機能性を拡張するために使用することができる。 戻り値は、callout が戻った後のメッセージか I-PDU の処理を、IL が継続する(COM_TRUE)か、放棄する(COM_FALSE)かどうかを示す。
戻り値	COM_TRUE：処理を継続
	COM_FALSE：処理を放棄
備考	-

4.11.3. COMErrorHook

構文	void COMErrorHook (StatusType <Error>)
パラメータ(in)	発生したエラーの識別子
パラメータ(out)	なし
記述	COMErrorHook はアプリケーションによって提供され、E_OK 以外のステータスコードを返す COM のシステムサービスの終わりに COM によって呼ばれる。
特性	-
戻り値	-
備考	-

5. 【付録Ａ】

ここでは定義名と概念を付録として記載する

5.1. CallBack Routine について

通知メカニズムの１つで、コールバックが呼ばれた時のコンテキストの振舞いは決まっている。

例. タスク　→タスクプライオリティとして振舞う

ISR　　→割込みプライオリティとして振舞う

長所：最も速いレスポンス時間を与える事が可能

短所：処理が重いと他のシステムに影響を与える可能性がある

5.2. m:n 通信について

＜レシーバー(受信側)＞

・メッセージは、各 ECU で複数のレシーバーを持つ事が可能

・アプリケーションは、多数のタスク・ISR を備えたどんなメッセージ・オブジェクトにもアクセスが可能

＜センダー(送信側)＞

・メッセージは、1ECU 内で、いくつもの送信者を持つことが可能

・メッセージは、１つのメッセージ・オブジェクトに１つのみ格納可能

・外部通信については、１つのメッセージ・オブジェクトは１つの I-PDU 内にメッセージを１つだけ含むことが可能

6. 【付録B】

参考にした文献を下記に示す

『通信用語の基礎知識』・・・http://www.wdic.org/d/COMM

変更履歴

Version	Date	Detail	Editor
1.0.0	2004/09/17	・ 初版作成	安田
1.0.1	2006/05/25	・ 外部通信に対応	鵜飼
1.0.2	2006/07/10	・ エラーコード追記	鵜飼
1.0.3	2007/02/28	・ 個別仕様記載、誤記修正	徳安/泉
1.0.4	2007/11/09	・ 誤記修正	泉
		・	