文書番号：

TOPPERS/OSEK
System Generation
通信ミドルウェア対応取扱説明書

Ver.1.01

2007/04/16

株式会社ヴィッツ　開発第３部

TOPPERS/OSEK System Generation
Toyohashi Open Platform for Embedded Real-Time Systems/
OSEK System Generation

＜目次＞

1. 概要

本通信ミドルウェア対応取扱説明書では、OSEK System Generator（以下 SG）の通信ミドルウェアについてのみ使用方法を記述する。

本 SG では、OIL（OSEK Implemantaion Language）と呼ばれる専門の記述言語によって書かれたソーステキストを入力し、通信ミドルウェアが必要とする C 言語のソースファイル及びその関連ファイルを出力するツールである。使用する OIL ファイルは「OSEK/VDX Implementation Language (OIL) Ver2.5」の仕様に基づいている必要がある。

1.1. 処理フロー

SG を用いて生成ファイルが出力される処理を図 1-1 に示す。

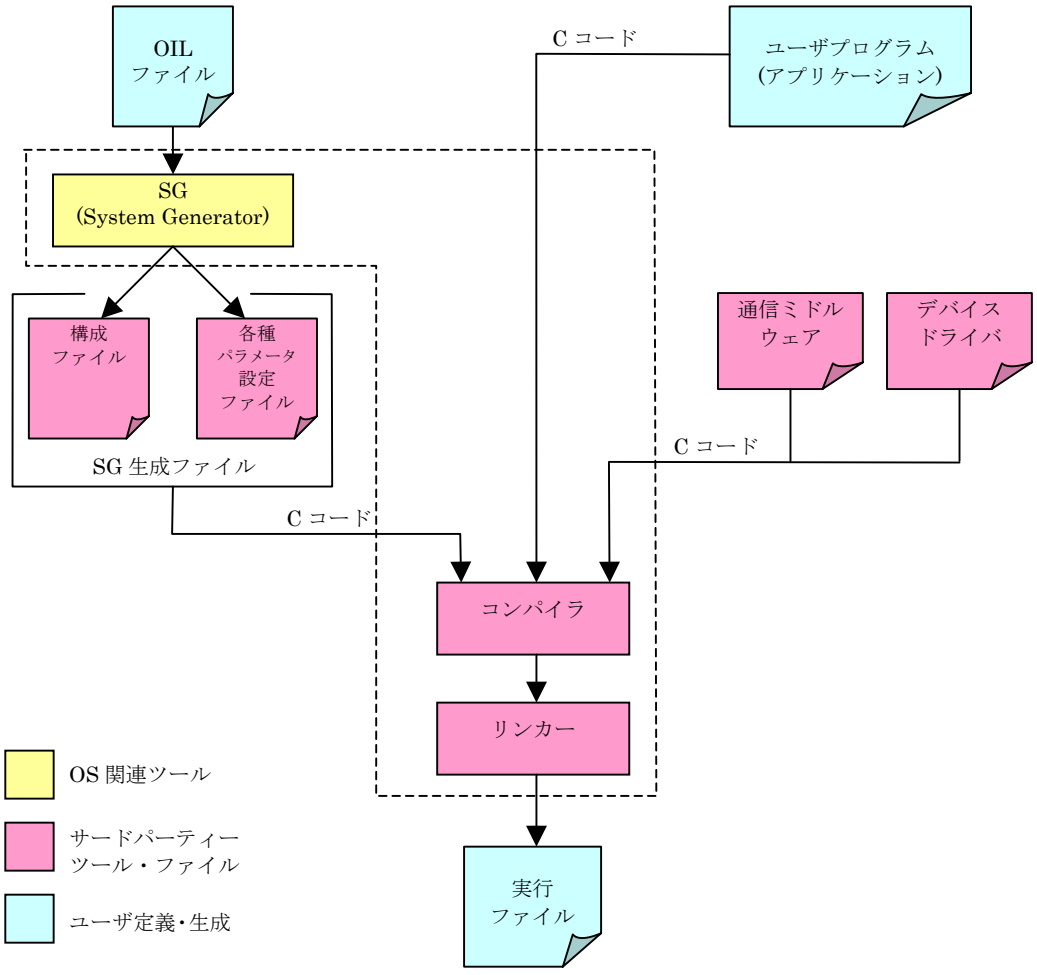


図 1　処理フロー

SG 生成ファイル、ユーザプログラム、通信ミドルウェアをコンパイルし、ライブラリをリンクして最終的にオブジェクトファイルを生成する流れになっている。

OILファイルには、SGが必要する情報（カーネルの一部を生成するといった動作定義）が記載されている。OILファイルについては、3 OILファイルを参照のこと。

1.2. 参考文献

・OSEK/VDX System Generation OIL:OSEK Implementation Language Version 2.5
・OSEK/VDX Communication Version 3.0.3
・SG 取扱説明書 v3.00

2. 使用方法

SG の実行は、コマンドプロンプト上から行う。
基本的な操作としてコマンドラインにより、SG 実行ファイル名、OIL ファイル名、及びコマンドラインオプションを指定する。

次に使用例を記載する。

■ Lin 使用時(OS 有/OS 無)

例）OIL ファイル：sample.oil の場合

sg.exe sample.oil –linmw –lj –I./../impl_oil -debug

指定内容：

LIN ミドルウェア、出力メッセージを日本語表示、OIL ファイルのインクルードパスを一階層上の impl_oil フォルダに指定、デバッグメッセージの出力。

■ DirectNM 使用時

例）OIL ファイル：directNM_canmw.oil の場合

sg.exe directNM_canmw.oil -I ../impl_oil -lj -directnm_canmw

■ InDirectNM 使用時

例）OIL ファイル：indirectNM_canmw.oil の場合

sg.exe indirectNM_canmw.oil -I ../impl_oil -lj -indirectnm_canmw

■ OS 有 COM 使用時

例）OIL ファイル：com_message.oil の場合

sg.exe com_message.oil -I ../impl_oil -lj -oscanmw

上記例を参考に SG を実行すると、カーネル構成ファイル及び ID 自動割付ファイルを出力する。

SGコマンドラインオプション一覧

オプション	別指定方法	内容
-h	--help	コマンドラインヘルプを表示します。
-debug		デバッグ用メッセージやツリーダンプを出力します。
-lj	--japanese	出力メッセージを日本語表示します。
-le	--english	出力メッセージを英語表示します。
-I <directory>		OIL ファイルのインクルードパスを指定します。
-oscanmw		OS 対応 CAN ミドルウェアセットを使用します。

-oscancom		OS 対応 CAN　COM ソフトウェアを使用します。
-oscannm		OS 対応 CAN　NM ソフトウェアを使用します。
-linmw		LIN ミドルウェアを使用します。
-linmwcfg=<file>		出力ファイル linmw_cfg.c を別名<file>に置換えます。
-linmwid=<file>		出力ファイル linmw_id.h を別名<file>に置換えます。
-directnm_candrv		非 OS 対応（Direct）CAN ドライバソフトウェアを使用します。
-directnm_cancom		非 OS 対応（Direct）CAN COM ソフトウェアを使用します。
-directnm_cannm		非 OS 対応（Direct）CAN NM ソフトウェアを使用します。
-directnm_canmw		非 OS 対応（Direct）CAN ミドルウェアセットを使用します。
-indirectnm_candrv		非 OS 対応（Indirect）CAN ドライバソフトウェアを使用します。
-indirectnm_cancom		非 OS 対応（Indirect）CAN COM ソフトウェアを使用します。
-indirectnm_cannm		非 OS 対応（Indirect）CAN NM ソフトウェアを使用します。
-indirectnm_canmw		非 OS 対応（Indirect）CAN ミドルウェアセットを使用します。
-tool=<tool>		<tool>に依存した処理を要求する。省略した場合、依存処理の出力を行いません。
-sjis		OIL ファイル内のシフト JIS を許可します。
-n <directory>		ファイルの出力先として<directory>を指定します。
-withouttime		出力ファイルのヘッダに生成日時を出力しないようにします。

3. OIL ファイル

本 SG がサポートしている OIL ファイルは、「OSEK/VDX Implementation Language (OIL) Version 2.5」に準拠したものになる。
本項目では、サポートしている OIL オブジェクト概要についてのみ記載する。OIL オブジェクトについての詳細な仕様は通信ミドルウェア対応 OIL 拡張仕様書を参照のこと。

3.1. OIL ファイル構造

OIL の記述は大きく分けて 3 部構成となっている。初めに準拠している OIL 仕様のバージョン情報を定義（OIL バージョン部）、次に標準的な実装仕様を示す定義（実装部）、最後に特定の CPU に配置されたアプリケーションの構造定義（アプリケーション部）である。

OIL ファイルの文法ルールは BNF 表記を使用した文章で表される。
全てのキーワード、属性、オブジェクト名称、他の識別子は、大文字と小文字を識別する。
BNF 表記のコメントは、C++スタイルのコメントとしても書かれる。
また、インクルードファイルを参照可能な構成になっている。

3.2. OIL 記述について

OIL 記述は、OIL オブジェクトの集合で構成されており、OIL オブジェクトは OIL より明示的に定義されている。

OS 対応 CAN 通信ミドルウェアで使用する OIL オブジェクトを記載する。

オブジェクト名称	説明
CANCOMMON	複数ネットワーク対応用に CAN ドライバ共通情報を設定する。
CANDRV	CAN ドライバ情報を設定する。
COM	COM 基本機能を設定する。
MESSAGE	アプリケーションが送信するデータ情報を設定する。
NETWORKMESSAGE	メッセージを IPDU に格納するための中間情報を設定する。
IPDU	外部送受信のデータ情報を設定する。
NM	CAN 通信ミドルウェアに対応した情報を設定する。
NMNODE	ネットワーク内のノード情報を設定する。

非 OS 対応 CAN 通信ミドルウェアで使用する OIL オブジェクトを記載する。

オブジェクト名称	説明
CANCOM	ネットワーク内のノード情報を設定する。
MESSAGE	アプリケーションが送信するデータ情報を設定する。

オブジェクト名称	説明
NETWORKMESSAGE	メッセージを IPDU に格納するための中間情報を設定する。
IPDU	外部送受信のデータ情報を設定する。
CANNM	Direct／IndirectNM 情報を設定する。
CANDRV	CAN ドライバ情報を設定する。
CANNODE	複数ネットワーク対応用に COM、CAN ドライバ共通情報を設定する。
CANNODE	COM、NM における共通情報を設定する。

LIN 通信ミドルウェアで使用する OIL オブジェクトを記載する。

オブジェクト名称	説明
LINNODE	LIN のノードに関する情報を持つオブジェクトである。
LINSIGNAL	LIN で使用するシグナル(送受信データ)の情報を持つオブジェクトである。
LINFRAME	LIN で使用するフレームの情報を持つオブジェクトである。
LINSCHEDULE	LIN のスケジュールテーブルを定義するオブジェクトである。

また、本 SG がサポートする OIL で使用できる型は次の通りである。

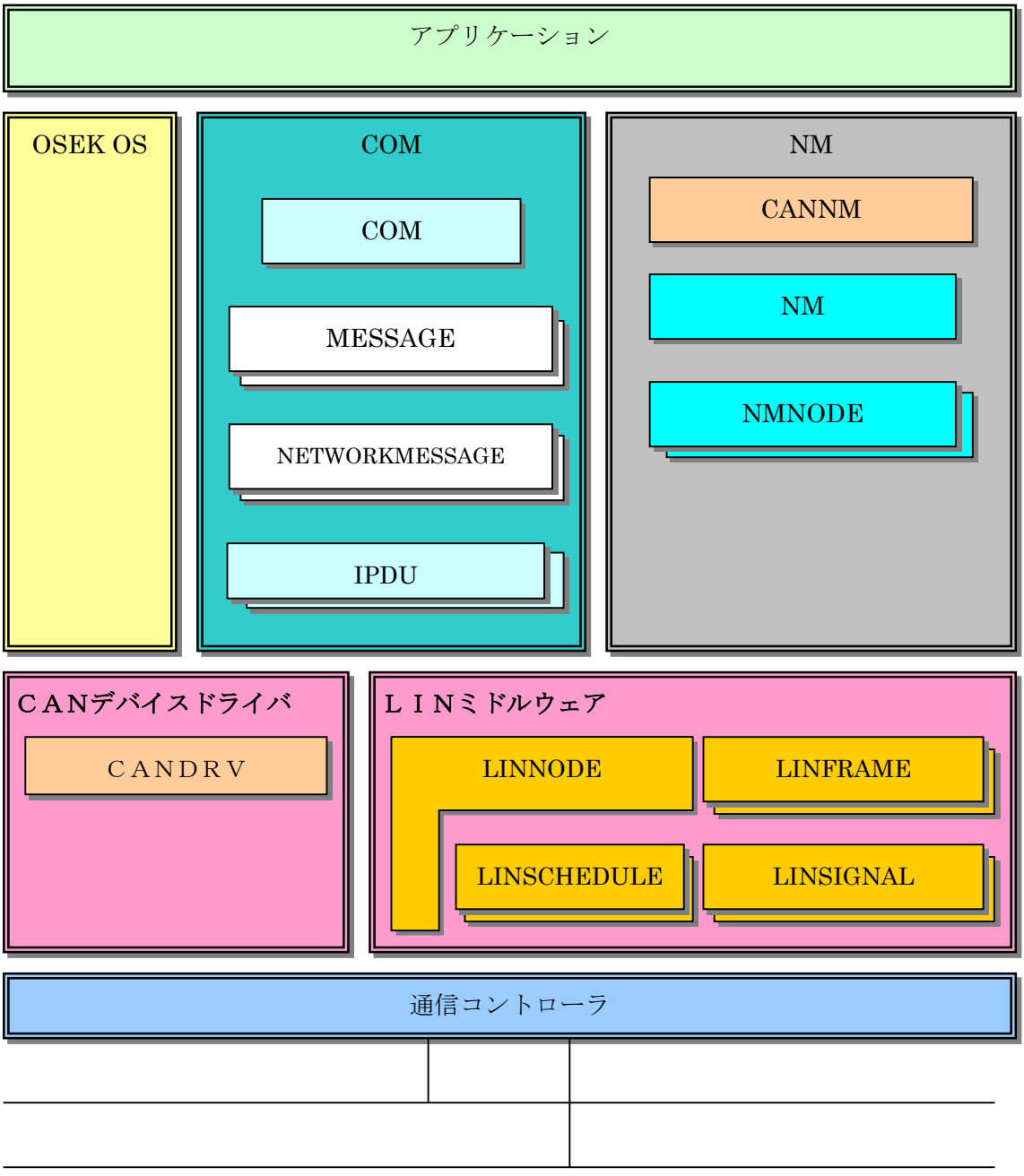
型	説明
UINT32	32 ビット長符号無し整数
INT32	32 ビット長符号付き整数
UINT64	64 ビット長符号無し整数
INT64	64 ビット長符号付き整数
FLOAT	浮動小数
ENUM	離散的な集合
BOOLEAN	TRUE,FALSE の指定
STRING	文字列（OIL 標準）
SYMBOLNAME	文字列（本 SG 独自拡張、SG にてシンボル名として有効かのチェック有り）
DATATYPE	文字列（本 SG 独自拡張、SG にて型名として有効かのチェック有り）

上記以外の型を指定した場合、OIL ファイルを認識することが出来ず出力ファイル生成エラーとなる。

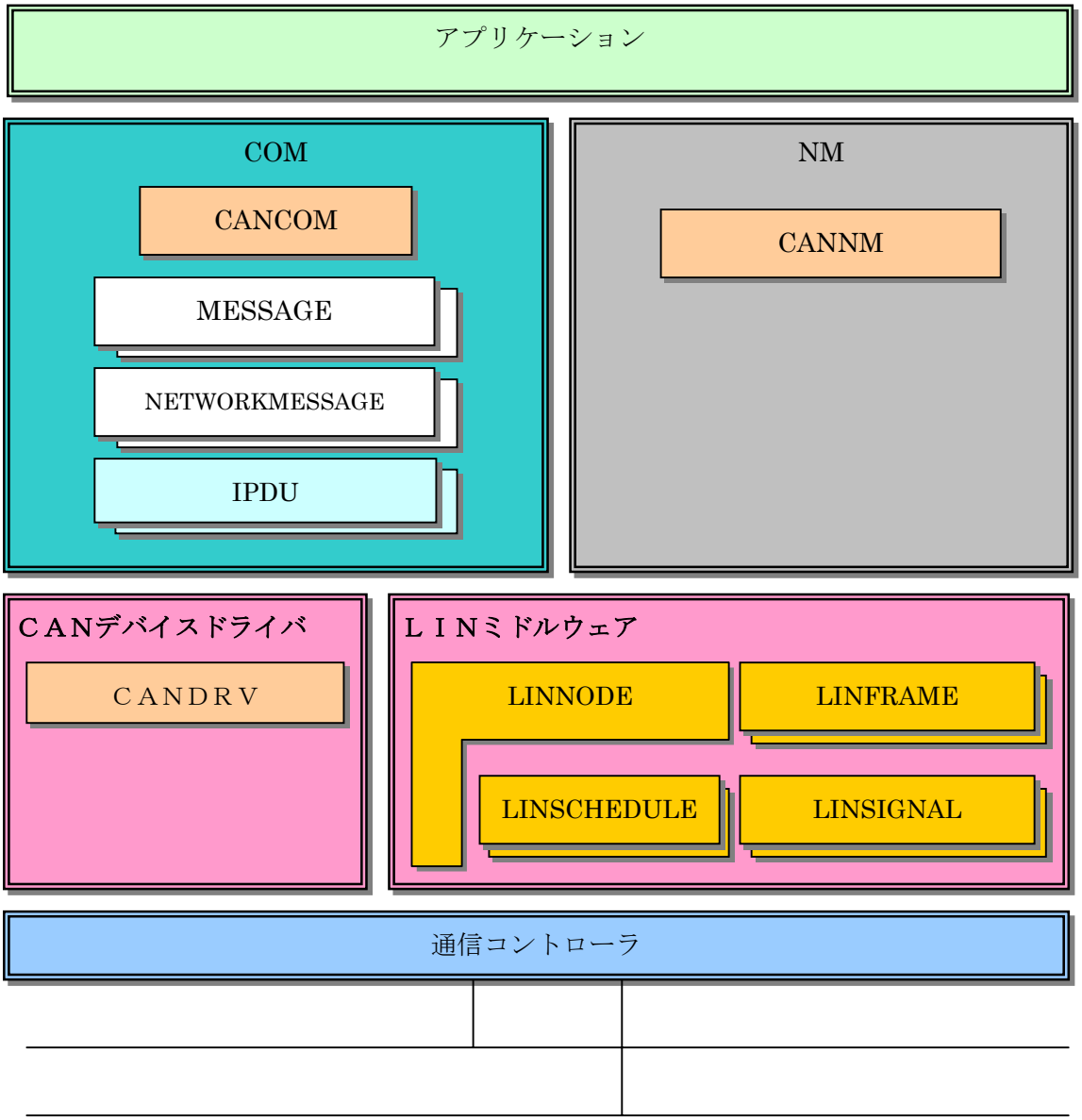
3.2.1. オブジェクト構成

オブジェクト構成図を下記に記載する。

3.2.1.1. 通信ミドルウェア(OS 対応版)



3.2.1.2. 通信ミドルウェア(非 OS 対応版)



3.3. OIL バージョン部

本 SG では、OIL バージョン 2.5 に対応している。
そのため使用する OIL ファイルの先頭に必ず次の一文を記載する必要がある。

```
OIL_VERSION = "2.5";
```

3.4. 実装部

実装部では、OIL オブジェクトに対して OIL 仕様に準拠している全ての属性と本 SG 独自拡張の属性を定義する。

実装部は、

```
IMPLEMENTATION　実装部名称　{　...　}
```

によって記述される。

以下に、本SGがサポートするOIL仕様準拠であるオブジェクトと属性について記載する。下記位階の実装部記述でも解釈は可能であるが、エラー処理が実行されない場合がある。標準属性の詳細に関しては、「OSEK/VDX Implementation Language (OIL) Version 2.5」を参照のこと。また、５出力処理に各オブジェクトの属性について説明が記載されているので、そちらを参照のこと。

各モジュールの実装部の記述形式については通信ミドルウェア対応 OIL 拡張仕様書を参照のこと。

3.5. アプリケーション部

アプリケーション部では、ユーザアプリケーションの実装に合わせたオブジェクトの状態を記述する。

SG ではアプリケーション部の記述が次の場合エラーにし、処理を停止する。

- 実装部に存在しないオブジェクトや属性を指定した。
- 実装部に記述した属性の型に合わない初期値を指定した。
- 実装部に記述した属性の範囲外の初期値を指定した。
- 実装部に NO_DEFAULT と指定があるのに、初期値を指定しなかった。
- 参照型属性に、存在しないオブジェクトを指定した。

また、SG にて値の整合性のチェックや、出力時に関連のあるオブジェクトを参照し合うが、その処理にて不整合な設定が発覚した場合もエラーとしている。

4. 構文解析処理とエラー検出処理

SG では OIL ファイルを入力とし構文解析を行う。OIL ファイルは、OIL バージョン部(OILVERSION)、実装部(IMPLEMENTATION)、アプリケーション部(CPU)で構成されている。OIL バージョン部、実装部、アプリケーション部の順に構文解析を行う。

4.1. OIL バージョン部

OIL バージョン部の解析では、OIL のバージョンが”2.5”となっていることをチェックする。これ以外のバージョンや文字など不適切な指定、また OIL_VERSION の指定が無かった場合は、エラーとし処理を中断する。

4.2. 実装部

実装部の解析では、オブジェクト毎に属性情報の管理を行う。

まず、

オブジェクト種別｛

の記述を検出したら、それが OIL でサポートされているオブジェクト(LINNODE, LINSIGNAL, LINFRAME, LINSCHEDULE)か否かを判別し、適合した場合その属性の解析に入る。

属性の解析では

- 属性名
- 型
- 範囲
- デフォルト値
- 説明文

を管理する。

4.3. アプリケーション部

アプリケーション部の解析では、生成するオブジェクトの情報をチェックする。

オブジェクト種別 オブジェクト名称｛

を解析したら、以降そのオブジェクトの属性について解析する。

存在する属性か否か、又は属性の値が範囲内か否かを実装部の解析結果と照らし合わせてチェックする。それが OK であれば、その属性の値を確定する。

ただし、値が AUTO であれば、実装部に WITH_AUTO の記述があることをチェックした後、SG 内で自動的に値を割り付けます。値の割り付けは属性によって異なる。

SG を実行すると、構文解釈処理により、OIL ファイルの記述内容によっては、エラーが出力される場合がある。この場合、SG はC言語コードファイルの生成処理を中断し、中断した原因を示すエラーメッセージが表示される。以下にエラーメッセージの一例と表示された場合の対処方法を記載する。

エラーメッセージ	説明、対応
OIL_VERSION exposes 2.5 as version	OIL_VERSION はバージョンとして 2.5 を期待しています。
`XXX' is out of range	`XXX' が範囲を超えています。OIL 仕様書を確認して有効属性値を設定して下さい。
undefined attribute `XXX'	未定義の属性`XXX' によりエラーが発生しています。属性名が異なっていないか確認して下さい。
invalid attribute value `XXX'	属性値`XXX' が不正です。OIL 仕様書を確認して有効属性値を設定して下さい。
The template pattern of `XXX' doesen't exit	XXX' というテンプレートパターンは存在しません。テンプレート文が異なっていないかテンプレートファイルを確認して下さい。
too many XXX objects	１つしか定義できないオブジェクトが複数定義されている可能性があります。または同名のオブジェクトが存在しています。
attribute `XXX' in XXX and `XXX' in XXX are `XXX'	XXX オブジェクトの XXX と XXX オブジェクトの XXX は XXX で一致しています。同じ値を設定してはいけないものに対し、同じ値の設定をしています。
`XXX' of `XXX' object `XXX' and `XXX' is overlapping	XXX と XXX は XXX オブジェクトの XXX の値が重複しています。重複しないよう設定してください。
attribute `XXX' of `XXX'(XXX) should be a multiple at the `XXX'	XXX の属性値 XXX(XXX) は XXX の倍数でなければなりません。倍数を設定しなおしてください。

5. 出力処理

本コンフィギュレータは、システムコンフィギュレーションファイルを処理して、通信ミドルウェア構成ファイルと各種パラメータ設定ファイルを生成する。各ファイルは、OIL 記述をもとにコンフィギュレータが通信ミドルウェアで必要な各種配列・値を C 言語ベースで出力したファイルである。

SG が出力するファイルを記載する。

モジュール名	ファイル名	説明

COM	can_com_api.c	非 OS 対応(Direct/Indirect)CANCOM ソフトウェアで使用する各エンティティの宣言を行う。
	can_com_api.h	非 OS 対応 CANCOM ソフトウェアで使用する構造体及びマクロの定義を行う。
	can_com_def.h	非 OS 対応 CANCOM ソフトウェアで使用する各種ハンドラの識別値を定義する。
	can_com_table.c	非 OS 対応 CANCOM ソフトウェアで使用する送受信データ関連テーブル及び使用 ID テーブルを定義する。
	com_cfg.c	OS 対応 CANCOM ソフトウェアで使用する各エンティティの宣言と初期設定を行う。
	com_cfg.h	OS 対応 CANCOM 卒とウェアで使用する各種マクロ及びハンドラの識別値を定義する。
CANDRV	can_drv_def.h	非 OS 対応(Direct/Indirect)CAN ドライバソフトウェアで使用する、各種ハンドラの識別値を定義する。
	can_drv_table.c	非 OS 対応(Direct/Indirect)CAN ドライバソフトウェアで各種データテーブルを定義する。
NM	can_nm_def.h	非 OS 対応(Direct/Indirect)CANNM ソフトウェアで使用する各種ハンドラの識別値を定義する。
	can_nm_table.c	非 OS 対応(Indirect)CANNM ソフトウェアで使用するノード関連テーブルを定義する。
	nm_cfg.c	OS 対応 CANNM ソフトウェアで使用する各エンティティの宣言と初期設定を行う。
	nm_cfg.h	OS 対応 CANNM ソフトウェアで使用する各種マクロ及びハンドラの識別値を定義する。
LINMW	linmw_id.h	各種ハンドラの識別値を定義する。
	linmw_cfg.c	LIN で使用する各エンティティの宣言と初期設定を行なうコードである。

5.1. OS 対応 CAN 通信ミドルウェア

5.1.1. メッセージ管理情報（MESSAGE）

5.1.1.1. メッセージ ID

OIL		設定値	出力ファイル
オブジェクト	属性		
MESSAGE	MESSAGEPROPERTY	SEND_STATIC_INTERNAL, SEND_STATIC_EXTERNAL,	com_cfg.h

		SEND_DYNAMIC_EXTERNAL, SEND_ZERO_INTERNAL, SEND_ZERO_EXTERNAL, RECEIVE_ZERO_INTERNAL, RECEIVE_ZERO_EXTERNAL, RECEIVE_UNQUEUED_INTERNAL, RECEIVE_QUEUED_INTERNAL, RECEIVE_UNQUEUED_EXTERNAL, RECEIVE_QUEUED_EXTERNAL, RECEIVE_DYNAMIC_EXTERNAL, RECEIVE_ZERO_SENDERS	

説明

メッセージ ID は 16bit 値とし、8bit ヘッダ部＋8bitID で構成しています。

8bit ヘッダの定義は下記のように分類分けしています。

メッセージIDヘッダ部定義ルール

受信メッセージ ID はヘッダの下位 4bit のいずれかのビットが ON になっている。
送信メッセージ ID はヘッダの上位 4bit のいずれかのビットが ON になっている。
・受信メッセージ定義
　受信メッセージ ID の定義は以下のように定義される。
　bit0　Special：[0]メッセージ，[1]IPDU からの受信
　bit1　Queuing：[1]キュー有りメッセージ，[0]キュー無しメッセージ
　bit2-3　Kind：[01]通常メッセージ，[10]ゼロレングス，[11]ダイナミック，[00]定義禁止
・送信メッセージ定義
　送信メッセージの ID 定義は以下のように定義される。
　bit4　Port：[0]内部，[1]外部
　bit5　Schedule：[0]トリガ，[1]ペンディング（内部通信の場合は無視されるが、0 にすること）
　bit6-7　Kind：[01]通常メッセージ，[10]ゼロレングス，[11]ダイナミック，[00]定義禁止

出力内容

#define メッセージオブジェクト名称　(MessageIdentifier)(メッセージ ID)

5.1.1.2. メッセージ数

OIL		設定値	出力ファイル
オブジェクト	属性		
MESSAGE	MESSAGEPROPERTY	SEND_STATIC_INTERNAL SEND_STATIC_EXTERNAL, SEND_DYNAMIC_EXTERNAL, SEND_ZERO_INTERNAL, SEND_ZERO_EXTERNAL,	com_cfg.h

		RECEIVE_ZERO_INTERNAL, RECEIVE_ZERO_EXTERNAL, RECEIVE_UNQUEUED_INTERNAL, RECEIVE_QUEUED_INTERNAL, RECEIVE_UNQUEUED_EXTERNAL, RECEIVE_QUEUED_EXTERNAL, RECEIVE_DYNAMIC_EXTERNAL, RECEIVE_ZERO_SENDERS	
説明			
メッセージを種別ごとにわけて出力します。			
出力内容			

・内部受信メッセージ

```
#define TNUM_RCV_UNQ_NORM     メッセージ数
#define TNUM_RCV_UNQ_ZERO     メッセージ数
#define TNUM_RCV_UNQ_DYNA     メッセージ数
#define TNUM_RCV_QUE_NORM     メッセージ数
const UINT8 tnum_rcv_unq_norm = (UINT8)TNUM_RCV_UNQ_NORM;
const UINT8 tnum_rcv_unq_zero = (UINT8)TNUM_RCV_UNQ_ZERO;
const UINT8 tnum_rcv_unq_dyna = (UINT8)TNUM_RCV_UNQ_DYNA;
const UINT8 tnum_rcv_que_norm = (UINT8)TNUM_RCV_QUE_NORM;
```

・IPDU受信受信メッセージ

```
#define TNUM_RCV_IPDU_NORM    メッセージ数
#define TNUM_RCV_IPDU_ZERO    メッセージ数
#define TNUM_RCV_IPDU_DYNA    メッセージ数
const UINT8 tnum_rcv_ipdu_norm = (UINT8)TNUM_RCV_IPDU_NORM;
const UINT8 tnum_rcv_ipdu_zero = (UINT8)TNUM_RCV_IPDU_ZERO;
const UINT8 tnum_rcv_ipdu_dyna = (UINT8)TNUM_RCV_IPDU_DYNA;
```

・内部送信メッセージ

```
#define TNUM_SND_IN_NORM      メッセージ数
#define TNUM_SND_IN_ZERO      メッセージ数
const UINT8 tnum_snd_in_norm = (UINT8)TNUM_SND_IN_NORM;
const UINT8 tnum_snd_in_zero = (UINT8)TNUM_SND_IN_ZERO;
```

・外部送信メッセージ

```
#define TNUM_SND_EX_TRIG_NORM     メッセージ数
#define TNUM_SND_EX_PEND_NORM     メッセージ数
#define TNUM_SND_EX_TRIG_ZERO     メッセージ数
#define TNUM_SND_EX_TRIG_DYNA     メッセージ数
#define TNUM_SND_EX_PEND_DYNA     メッセージ数
const UINT8 tnum_snd_ex_trig_norm = (UINT8)TNUM_SND_EX_TRIG_NORM;
const UINT8 tnum_snd_ex_pend_norm = (UINT8)TNUM_SND_EX_PEND_NORM;
const UINT8 tnum_snd_ex_trig_zero = (UINT8)TNUM_SND_EX_TRIG_ZERO;
const UINT8 tnum_snd_ex_trig_dyna = (UINT8)TNUM_SND_EX_TRIG_DYNA;
const UINT8 tnum_snd_ex_pend_dyna = (UINT8)TNUM_SND_EX_PEND_DYNA;
```

5.1.1.3. キューメッセージのデータ格納数

OIL		設定値	出力ファイル
オブジェクト	属性	RECEIVE_QUEUED_INTERNAL, RECEIVE_QUEUED_EXTERNAL	com_cfg.c
MESSAGE	MESSAGEPROPERTY		
説明			
次のメッセージ種別が指定されてあるオブジェクトは、データ格納数を com_cfg.c に出力します。 メッセージ種別： RECEIVE_QUEUED_INTERNAL、RECEIVE_QUEUED_EXTERNAL			
出力内容			
const UINT8 tnum_que_elem[TNUM_RCV_QUE_NORM] = {..};			

5.1.1.4. メッセージデータ格納バッファの定義

OIL		設定値	出力ファイル
オブジェクト	属性	RECEIVE_UNQUEUED_INTERNAL, RECEIVE_QUEUED_INTERNAL, RECEIVE_UNQUEUED_EXTERNAL, RECEIVE_QUEUED_EXTERNAL	com_cfg.c
MESSAGE	MESSAGEPROPERTY		
	CDATATYPE	型	
説明			
次のメッセージ種別が指定されてあるオブジェクトは、CDATATYPE を基にデータ格納バッファを com_cfg.c に出力します。尚、RECEIVE_UNQUEUED_INTERNAL の場合は、SENDINGMESSAGE で指定されているメッセージの CDATATYPE を参照します。 メッセージ種別： RECEIVE_QUEUED_INTERNAL、RECEIVE_QUEUED_EXTERNAL RECEIVE_UNQUEUED_INTERNAL、RECEIVE_UNQUEUED_EXTERNAL			
出力内容			

・受信-キューなし-通常メッセージデータ格納バッファ

UINT32 msgbuf_メッセージ名称;
const ApplicationDataRef msgbuf_rcv_unq_norm[TNUM_RCV_UNQ_NORM] ={…};
const UINT32 msgsize_rcv_unq_norm[TNUM_RCV_UNQ_DYNA] ={…};

・受信-キュー有り-通常メッセージデータ格納バッファ

UINT16　msgbuf_メッセージ名称[TNUM_QUE_ELEM_メッセージ名称];
const ApplicationDataRef msgbuf_head_que_norm[TNUM_RCV_QUE_NORM] = {…};
const ApplicationDataRef msgbuf_end_que_norm[TNUM_RCV_QUE_NORM] = {…};
const UINT32 msgsize_que_norm[TNUM_RCV_UNQ_DYNA] = {…};

UINT8 msgque_cnt[TNUM_RCV_QUE_NORM];
ApplicationDataRef msgque_get_addr[TNUM_RCV_QUE_NORM];
ApplicationDataRef msgque_put_addr[TNUM_RCV_QUE_NORM];

・受信-キューなし-ダイナミックメッセージデータ格納バッファ

UINT8 msgbuf_メッセージ名称[型];
const ApplicationDataRef msgbuf_rcv_unq_dyna[TNUM_RCV_UNQ_DYNA] = {…};
const UINT32 msgsize_rcv_unq_dyna[TNUM_RCV_UNQ_DYNA] = {…};
COMLengthType msglength_メッセージ名称;

<table>
<tr><th colspan="2">OIL</th><th>設定値</th><th>出力ファイル</th></tr>
<tr><td>オブジェクト</td><td>属性</td><td rowspan="2">SEND_DYNAMIC_EXTERNAL,
RECEIVE_DYNAMIC_EXTERNAL</td><td rowspan="2">com_cfg.c</td></tr>
<tr><td>MESSAGE</td><td>MESSAGEPROPERTY</td></tr>
<tr><td>NETWORKMESSAGE</td><td>MAXIMUNSIZEINBITS</td><td>数値</td><td></td></tr>
<tr><td colspan="4">説明</td></tr>
<tr><td colspan="4">次のメッセージ種別が指定されている場合、関連している NETWORKMESSAGE 管理情報の MAXIMUNSIZEINBITS の値をバイト値で出力します。
メッセージ種別：
RECEIVE_DYNAMIC_EXTERNAL、SEND_DYNAMIC_EXTERNAL</td></tr>
<tr><td colspan="4">出力内容</td></tr>
</table>

・送信-ダイナミック-トリガメッセージのサイズ

```
const UINT32 msgsize_snd_ex_trig_dyna[TNUM_SND_EX_TRIG_DYNA] = {...};
```

・送信-ダイナミック-ペンディングメッセージ

```
const UINT32 msgsize_snd_ex_pend_dyna[TNUM_SND_EX_PEND_DYNA] = {...};
```

5.1.1.5. データコピー関数の定義

OIL		設定値	出力ファイル
オブジェクト	属性	RECEIVE_UNQUEUED_INTERNAL, RECEIVE_QUEUED_INTERNAL, RECEIVE_UNQUEUED_EXTERNAL, RECEIVE_QUEUED_EXTERNAL	com_cfg.c
MESSAGE	MESSAGEPROPERTY		
	CDATATYPTE	型	

説明

次のメッセージ種別が指定されている場合、データコピー関数及び関数配列を com_cfg.c に出力します。
尚、RECEIVE_UNQUEUED_INTERNAL、RECEIVE_UNQUEUED_EXTERNAL の場合は、
参照先のメッセージから CDATATYPE を取得して出力します。
メッセージ種別：
　RECEIVE_UNQUEUED_EXTERNAL、RECEIVE_QUEUED_EXTERNAL
　RECEIVE_UNQUEUED_INTERNAL、RECEIVE_UNQUEUED_EXTERNAL

出力内容

```
static void cpy_UINT16(ApplicationDataRef p_dst, ApplicationDataRef p_src );
void cpy_UINT16(ApplicationDataRef p_dst, ApplicationDataRef p_src )
{
    *((UINT16*)p_dst) = *((const UINT16*)p_src);
}

const COPYFP copyfunc_rcv_unq_norm[TNUM_RCV_UNQ_NORM] = {..};
const COPYFP copyfunc_rcv_que_norm[TNUM_RCV_QUE_NORM] = {..};

const COPYFP copyfunc_snd_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..};
const COPYFP copyfunc_snd_pend_norm[TNUM_SND_EX_PEND_NORM] = {..};
```

5.1.1.6. 通常送信メッセージのサイズ

OIL	設定値	出力ファイル

オブジェクト	属性	SEND_STATIC_EXTERNAL	com_cfg.c
MESSAGE	MESSAGEPROPERTY		

説明
次のメッセージ種別が指定されている場合、メッセージサイズごとのバッファ配列を com_cfg.c へ出力します。 メッセージ種別； SEND_STATIC_EXTERNAL

出力内容
UINT8 const msgsize_snd_ex_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..}; UINT8 const msgsize_snd_ex_pend_norm[TNUM_SND_EX_PEND_NORM] = {..};

5.1.1.7. UNNQUEUE メッセージバッファの初期化処理

OIL		設定値	出力ファイル
オブジェクト	属性	RECEIVE_UNQUEUED_INTERNAL, RECEIVE_UNQUEUED_EXTERNAL	com_cfg.c
MESSAGE	MESSAGEPROPERTY		
	INITIALVALUE	数値	

説明
次のメッセージ種別が指定されている場合、UNNQUEUE メッセージバッファの初期化関数を com_cfg.c へ出力します。 メッセージ種別； RECEIVE_UNQUEUED_INTERNAL、RECEIVE_UNQUEUED_EXTERNAL

出力内容

void unque_init(void);
void unque_init(void)
{
(void)com_local_memset((VP)(&buf_メッセージ名称), (INT8)0, (UINT32)(sizeof(UINT32)));
msgbuf_メッセージ名称 = (UINT32)初期値;
}

5.1.1.8. フラグ初期化関数の生成

OIL		設定値	出力ファイル
オブジェクト	属性	FLAG	com_cfg.c
MESSAGE	NOTIFICATION		
	NOTIFICATIONERROR		
	FLAGNAME	オブジェクト名称	

説明
NOTIFICATION 属性または NOTIFICATIONERROR 属性に FLAG が指定されている場合、フラグの初期化関数を com_cfg.c に出力します。
出力内容

void init_flags(void);
void init_flags(void)
{
　フラグ名称 = COM_FALSE;
}

5.1.1.9. フラグアクセス関数の生成

OIL		設定値	出力ファイル
オブジェクト	属性	FLAG	com_cfg.c
MESSAGE	NOTIFICATION		
	NOTIFICATIONERROR		
	FLAGNAME	オブジェクト名称	

説明
NOTIFICATION または NOTIFICATIONERROR に FLAG が指定されている場合、フラグアクセス関数を com_cfg.c に出力します。
出力内容

static FlagValue notice_flag;
FlagValue ReadFlag_フラグ名称(void)
{
　FlagValue　 tempdata;
　ENTER_CRITICALSECTION();
　tempdata = フラグ名称;
　LEAVE_CRITICALSECTION();
　return tempdata;
}
void ResetFlag_フラグ名称(void)
{
　ENTER_CRITICALSECTION();
　フラグ名称 = COM_FALSE;
　LEAVE_CRITICALSECTION();
}

5.1.1.10. 受信フィルタリング

OIL		設定値	出力ファイル
オブジェクト	属性	RECEIVE_UNQUEUED_INTERNAL, RECEIVE_QUEUED_INTERNAL, RECEIVE_UNQUEUED_EXTERNAL, RECEIVE_QUEUED_EXTERNAL	com_cfg.c
MESSAGE	MESSAGEPROPERTY		
	CDATATYPE	型	

説明

次のメッセージ種別が指定されている場合に、フィルタ関数配列とフィルタ初期化関数配列を com_cfg.c に出力します。尚、関数が存在しない場合は、FILTFP_NULL、INIFILTFP_NULL を格納します。

メッセージ種別：

RECEIVE_UNQUEUED_INTERNAL、RECEIVE_QUEUED_INTERNAL
RECEIVE_UNQUEUED_EXTERNAL、RECEIVE_QUEUED_EXTERNAL

出力内容

・受信-キューなし-通常メッセージ

```
const FILTFP filter_rcv_unq_norm[TNUM_RCV_UNQ_NORM] = {..};
const FILTFP filter_init_rcv_unq_norm[TNUM_RCV_UNQ_NORM] = {..};
```

・受信-キュー有り-通常メッセージ

```
const FILTFP filter_rcv_que_norm[TNUM_RCV_QUE_NORM] = {..};
const FILTFP filter_init_rcv_que_norm[TNUM_RCV_QUE_NORM] = {..};
```

5.1.1.11. 送信フィルタリング

OIL		設定値	出力ファイル
オブジェクト	属性	SEND_STATIC_EXTERNAL	com_cfg.c
MESSAGE	MESSAGEPROPERTY		
	CDATATYPE	型	

説明

次のメッセージ種別が指定されている場合に、フィルタ関数配列とフィルタ初期化関数配列を com_cfg.c に出力します。尚、関数が存在しない場合は、FILTFP_NULL を格納します。

メッセージ種別：

SEND_STATIC_EXTERNAL

出力内容

・送信-外部-トリガ-通常メッセージ

```
const FILTFP filter_snd_ex_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..};
const FILTFP filter_init_snd_ex_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..};
```

・送信-外部-ペンディング-通常メッセージ

```
const FILTFP filter_snd_ex_pend_norm[TNUM_SND_EX_PEND_NORM] = {..};
const FILTFP filter_init_snd_ex_pend_norm[TNUM_SND_EX_PEND_NORM]= {..};
```

5.1.1.12. フィルタ関数の生成

OIL		設定値	出力ファイル
オブジェクト	属性	RECEIVE_UNQUEUED_INTERNAL, RECEIVE_QUEUED_INTERNAL, RECEIVE_UNQUEUED_EXTERNAL, RECEIVE_QUEUED_EXTERNAL, SEND_STATIC_EXTERNAL	com_cfg.c
MESSAGE	MESSAGEPROPERTY		
	CDATATYPE	型	

説明
次のメッセージ種別が指定されている場合に、フィルタ関数配列とフィルタ初期化関数配列を com_cfg.c に出力します。尚、関数が存在しない場合は、FILTFP_NULL を格納します。 メッセージ種別： SEND_STATIC_EXTERNAL
出力内容
フィルタ関数テンプレートを巻末付録（フィルタ関数のテンプレート）に記載します。

5.1.1.13. 初期値テーブルの生成

OIL		設定値	出力ファイル
オブジェクト	属性	SEND_STATIC_EXTERNAL, SEND_DYNAMIC_EXTERNAL, RECEIVE_UNQUEUED_INTERNAL, RECEIVE_QUEUED_INTERNAL, RECEIVE_UNQUEUED_EXTERNAL, RECEIVE_QUEUED_EXTERNAL, RECEIVE_DYNAMIC_EXTERNAL	com_cfg.h
MESSAGE	MESSAGEPROPERTY		

	INITIALVALUE	数値	
NETWORKMESSAGE	INITIALVALUE	数値	

説明
次のメッセージ種別が指定されている場合に、INITIALVALUE 属性の初期化テーブルを com_cfg.c に出力します。INITIALVALUE 属性に AUTO 指定がされていた場合は、関連する NETWORKMESSAGE 管理情報の INITIALVALUE を参照します。 メッセージ種別： RECEIVE_UNQUEUED_INTERNAL RECEIVE_QUEUED_INTERNAL RECEIVE_UNQUEUED_EXTERNAL RECEIVE_QUEUED_EXTERNAL RECEIVE_DYNAMIC_EXTERNAL SEND_STATIC_EXTERNAL SEND_DYNAMIC_EXTERNAL
出力内容
const UINT64 initialvalue_rcv_unq_norm[TNUM_RCV_UNQ_NORM] = {..}; const UINT64 initialvalue_rcv_unq_dyna[TNUM_RCV_UNQ_DYNA] = {..}; const UINT64 initialvalue_rcv_que_norm[TNUM_RCV_QUE_NORM] = {..}; const UINT64 initialvalue_snd_ex_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..}; const UINT64 initialvalue_snd_ex_pend_norm[TNUM_SND_EX_PEND_NORM] ={..}; const UINT64 initialvalue_snd_ex_trig_dyna[TNUM_SND_EX_TRIG_DYNA] = {..}; const UINT64 initialvalue_snd_ex_pend_dyna[TNUM_SND_EX_PEND_DYNA] = {..};

5.1.1.14. バイトオーダ

OIL		設定値	出力ファイル
オブジェクト	属性		
MESSAGE	MESSAGEPROPERTY	RECEIVE_UNQUEUED_EXTERNAL, RECEIVE_QUEUED_EXTERNAL, SEND_STATIC_EXTERNAL	com_cfg.c
NETWORKMESSAGE	DATAINTERPRETATION	UNSIGNEDINTEGER	
説明			

関連する NETWORKMESSAGE オブジェクトの DATAINTERPRETATION 属性を参照して、バイトオーダの実行有無が記載されたテーブルを com_cfg.c に出力します。 メッセージ種別： RECEIVE_UNQUEUED_EXTERNAL RECEIVE_QUEUED_EXTERNAL SEND_STATIC_EXTERNAL
出力内容
const UINT8 byteorder_snd_ex_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..}; const UINT8 byteorder_snd_ex_pend_norm[TNUM_SND_EX_PEND_NORM] = {..};

5.1.1.15. NETWORKMESSAGE との関連

OIL		設定値	出力ファイル
オブジェクト	属性		
MESSAGE	MESSAGEPROPERTY	SEND_STATIC_EXTERNAL, SEND_DYNAMIC_EXTERNAL, SEND_ZERO_EXTERNAL	com_cfg.c
	TRANSFERPROPERTY	TRIGGERD／PENDING／AUTO	
NETWORKMESSAGE	IPDU	オブジェクト参照	
	SIZEINBITS	数値	
	BITPOSITION	数値	
	TRANSFERPROPERTY	TRIGGERD／PENDING／AUTO	
	MESSAGEPROPERTY	－	
	DIRECTION	SENT／RECEIVED	
説明			
関連する NETWORKMESSAGE オブジェクトの IPDU 名、サイズインビッツ、ビットポジションの値を com_cfg.c に出力します。TRANSFERPROPERTY 属性が AUTO の場合は、参照している NETWORKMESSAGE オブジェクトの TRANSFERPROPERTY を参照します。この場合、両属性とも AUTO、両属性とも異なる値の場合はエラーとします。その他にも、NETWORKMESSAGE オブジェクトの MESSAGEPROPERTY 属性、DIRECTION 属性が MESSAGE オブジェクトの MESSAGEPROPERTY]と一致しているか確認します。 メッセージ種別： SEND_STATIC_EXTERNAL SEND_DYNAMIC_EXTERNAL SEND_ZERO_EXTERNAL			
出力内容			

```
const IPDUType ipduid_snd_ex_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..};
const UINT32 bitsize_snd_ex_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..};
const UINT32 offset_snd_ex_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..};
const IPDUType ipduid_snd_ex_pend_norm[TNUM_SND_EX_PEND_NORM] = {..};
const UINT32 bitsize_snd_ex_pend_norm[TNUM_SND_EX_PEND_NORM] = {..};
const UINT32 offset_snd_ex_pend_norm[TNUM_SND_EX_PEND_NORM] = {..};
const IPDUType ipduid_snd_ex_trig_zero[TNUM_SND_EX_TRIG_ZERO] = {..};
const IPDUType ipduid_snd_ex_trig_dyna[TNUM_SND_EX_TRIG_DYNA] = {..};
const UINT32 offset_snd_ex_trig_dyna[TNUM_SND_EX_TRIG_DYNA] = {..};
const IPDUType ipduid_snd_ex_pend_dyna[TNUM_SND_EX_ PEND_DYNA] = {..};
const UINT32 offset_snd_ex_pend_dyna[TNUM_SND_EX_ PEND_DYNA] = {..};
```

5.1.1.16. 受信メッセージの Notification 情報

<table>
<tr><th colspan="2">OIL</th><th>設定値</th><th>出力ファイル</th></tr>
<tr><td>オブジェクト</td><td>属性</td><td rowspan="2">RECEIVE_ZERO_INTERNAL,
RECEIVE_ZERO_EXTERNAL,
RECEIVE_UNQUEUED_INTERNAL,
RECEIVE_QUEUED_INTERNAL,
RECEIVE_UNQUEUED_EXTERNAL,
RECEIVE_QUEUED_EXTERNAL,
RECEIVE_DYNAMIC_EXTERNAL</td><td rowspan="3">com_cfg.c</td></tr>
<tr><td>MESSAGE</td><td>MESSAGEPROPERTY</td></tr>
<tr><td></td><td>NOTIFICATION</td><td>ACTIVATETASK,
COMCALLBACK,
FLAG,
INMCALLBACK</td></tr>
<tr><td colspan="4">説明</td></tr>
</table>

受信メッセージの Notification 情報を com_cfg.c に出力します。 尚、NOTIFICATION を使用しない場合には、FP_NULL を設定します。 メッセージ種別： RECEIVE_ZERO_INTERNAL RECEIVE_ZERO_EXTERNAL RECEIVE_UNQUEUED_INTERNAL RECEIVE_QUEUED_INTERNAL RECEIVE_UNQUEUED_EXTERNAL RECEIVE_QUEUED_EXTERNAL RECEIVE_DYNAMIC_EXTERNAL
出力内容
const FP notification1_rcv_unq_norm[TNUM_RCV_UNQ_NORM] = {..}; const FP notification1_rcv_unq_zero[TNUM_RCV_UNQ_ZERO] = {..}; const FP notification1_rcv_unq_dyna[TNUM_RCV_UNQ_DYNA] = {..}; const FP notification1_rcv_que_norm[TNUM_RCV_QUE_NORM] = {..};

5.1.1.17. 外部送信時 Notification 情報

OIL		設定値	出力ファイル
オブジェクト	属性	SEND_ZERO_EXTERNAL, SEND_STATIC_EXTERNAL, SEND_DYNAMIC_EXTERNAL	com_cfg.c
MESSAGE	MESSAGEPROPERTY		
	NOTIFICATION	ACTIVATETASK, COMCALLBACK, FLAG, INMCALLBACK	
説明			
外部送信メッセージの Notification 情報を com_cfg.c に出力します。 尚、NOTIFICATION を使用しない場合には、FP_NULL を設定します。 メッセージ種別： SEND_ZERO_EXTERNAL SEND_STATIC_EXTERNAL SEND_DYNAMIC_EXTERNAL			
出力内容			

```
const FP notification2_snd_ex_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..};
const FP notification2_snd_ex_pend_norm[TNUM_SND_EX_PEND_NORM] = {..};
const FP notification2_snd_ex_trig_zero[TNUM_SND_EX_TRIG_ZERO] = {..};
const FP notification2_snd_ex_trig_dyna[TNUM_SND_EX_TRIG_DYNA] = {..};
const FP notification2_snd_ex_pend_dyna[TNUM_SND_EX_PEND_DYNA] = {..};
```

5.1.1.18. IPDU における Notification 情報

OIL		設定値	出力ファイル
オブジェクト	属性		
MESSAGE	MESSAGEPROPERTY	RECEIVE_ZERO_INTERNAL, RECEIVE_ZERO_EXTERNAL, RECEIVE_UNQUEUED_INTERNAL, RECEIVE_QUEUED_INTERNAL, RECEIVE_UNQUEUED_EXTERNAL, RECEIVE_QUEUED_EXTERNAL, RECEIVE_DYNAMIC_EXTERNAL, SEND_ZERO_EXTERNAL, SEND_STATIC_EXTERNAL, SEND_DYNAMIC_EXTERNAL	com_cfg.c
	NOTIFICATION	ACTIVATETASK, COMCALLBACK, FLAG, INMCALLBACK	
説明			

通知クラス CLASS3,CLASS4 に該当するコールバックルーチン名を com_cfg.c に出力します。尚、NOTIFICATION を使用しない場合には、FP_NULL を設定します。
メッセージ種別：
　RECEIVE_ZERO_INTERNAL
　RECEIVE_ZERO_EXTERNAL
　RECEIVE_UNQUEUED_INTERNAL
　RECEIVE_QUEUED_INTERNAL
　RECEIVE_UNQUEUED_EXTERNAL
　RECEIVE_QUEUED_EXTERNAL
　RECEIVE_DYNAMIC_EXTERNAL
　SEND_ZERO_EXTERNAL
　SEND_STATIC_EXTERNAL
　SEND_DYNAMIC_EXTERNAL

出力内容

```
const FP notification3_rcv_unq_norm[TNUM_RCV_UNQ_NORM] = {..};
const FP notification3_rcv_unq_zero[TNUM_RCV_UNQ_ZERO] = {..};
const FP notification3_rcv_unq_dyna[TNUM_RCV_UNQ_DYNA] = {..};
const FP notification3_rcv_que_norm[TNUM_RCV_QUE_NORM] = {..};

const FP notification4_snd_ex_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..};
const FP notification4_snd_ex_pend_norm[TNUM_SND_EX_PEND_NORM] = {..};
const FP notification4_snd_ex_trig_zero[TNUM_SND_EX_TRIG_ZERO] = {..};
const FP notification4_snd_ex_trig_dyna[TNUM_SND_EX_TRIG_DYNA] = {..};
const FP notification4_snd_ex_pend_dyna[TNUM_SND_EX_PEND_DYNA] = {..};
```

5.1.1.19. CALLOUT

OIL		設定値	出力ファイル
オブジェクト	属性		
MESSAGE	MESSAGEPROPERTY	RECEIVE_ZERO_EXTERNAL, RECEIVE_UNQUEUED_EXTERNAL, RECEIVE_QUEUD_EXTERNAL, RECEIVE_DYNAMIC_EXTERNAL, SEND_ZERO_EXTERNAL, SEND_STATIC_EXTERNAL	com_cfg.c

LINK	FALSE
NETWORKORDERCALLOUT	シンボル名称
CPUORDERCALLOUT	シンボル名称

説明

ネットワークオーダコールアウト、バイトオーダコールアウト名を com_cfg.c に出力します。
メッセージ種別：

RECEIVE_ZERO_EXTERNAL
RECEIVE_UNQUEUED_EXTERNAL
RECEIVE_QUEUED_EXTERNAL
RECEIVE_DYNAMIC_EXTERNAL
SEND_ZERO_EXTERNAL
SEND_STATIC_EXTERNAL

出力内容

```
const CALLOUTFP callout_cpuorder_rcv_ipdu_norm[TNUM_RCV_IPDU_NORM] =   {..};
const CALLOUTFP callout_netorder_rcv_ipdu_norm[TNUM_RCV_IPDU_NORM] = {..};

const CALLOUTFP callout_cpuorder_rcv_ipdu_zero[TNUM_RCV_IPDU_ZERO] = {..};
const CALLOUTFP callout_netorder_rcv_ipdu_zero[TNUM_RCV_IPDU_ZERO] = {..};

const CALLOUTFP callout_cpuorder_rcv_ipdu_dyna[TNUM_RCV_IPDU_DYNA] = {..};
const CALLOUTFP callout_netorder_rcv_ipdu_dyna[TNUM_RCV_IPDU_DYNA] = {..}

const CALLOUTFP callout_cpuorder_snd_ex_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..};
const CALLOUTFP callout_netorder_snd_ex_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..};

const CALLOUTFP callout_cpuorder_snd_ex_pend_norm[TNUM_SND_EX_PEND_NORM] = {..};
const CALLOUTFP callout_netorder_snd_ex_pend_norm[TNUM_SND_EX_PEND_NORM] = {..};

const CALLOUTFP callout_cpuorder_snd_ex_trig_zero[TNUM_SND_EX_TRIG_ZERO] = {..};
const CALLOUTFP callout_netorder_snd_ex_trig_zero[TNUM_SND_EX_TRIG_ZERO] = {..};

const CALLOUTFP callout_cpuorder_snd_ex_trig_dyna[TNUM_SND_EX_TRIG_DYNA] = {..};
const CALLOUTFP callout_netorder_snd_ex_trig_dyna[TNUM_SND__EX_TRIG_DYNA] = {..};

const CALLOUTFP callout_cpuorder_snd_ex_pend_dyna[TNUM_SND_EX_PEND_DYNA] = {..};
const CALLOUTFP callout_netorder_snd_ex_pend_dyna[TNUM_SND_EX_PEND_DYNA] = {..};
```

5.1.1.20. 配信先テーブル情報

OIL		設定値	出力ファイル
オブジェクト	属性	SEND_STATIC_INTERNAL, SEND_STATIC_ZERO	com_cfg.c
MESSAGE	MESSAGEPROPERTY		
説明			
メッセージ種別ごとのメッセージテーブルを com_cfg.c に出力します。 MESSAGEPROPERTY 属性が SEND_STATIC_INTERNAL,SEND_STATIC_ZERO の場合に MESSAGE オブジェクト名称を出力します。また、配列の終端には MSG_NULL を出力します。			
出力内容			
const MessageIdentifier sndlist _オブジェクト名称[] = {..}; const MessageIdentifier *sndlist_snd_in_norm[TNUM_SND_IN_NORM] = {..}; const MessageIdentifier sndlist_snd_in_zero_オブジェクト名称[] = {..}; const MessageIdentifier *sndlist_snd_in_zero[TNUM_SND_IN_ZERO] = {..};			

5.1.2. ネットワークメッセージ管理情報（NETWORKMESSAGE）

5.1.2.1. ネットワークメッセージ ID

OIL		設定値	出力ファイル
オブジェクト	属性	－	com_cfg.h
NETWORKMESSAGE	－		
説明			
メッセージ管理情報、ネットワークメッセージ管理情報、IPDU 管理情報を跨いでメッセージ ID を割り付けます。基本的にメッセージ ID ヘッダ部定義ルールに基づいてネットワークメッセージ ID を割り振っています。			
出力内容			
#define ネットワークメッセージオブジェクト名称 (MessageIdentifier)(数値) #define ネットワークメッセージオブジェクト名称 (MessageIdentifier)(数値)			

5.1.2.2. 配信先テーブル情報

OIL		設定値	出力ファイル
オブジェクト	属性	STATIC, DYNAMIC,	com_cfg.c
NETWORKMESSAGE	MESSAGEPROPERTY		

		ZERO,	
	DIRECTION	RECEIVED	
説明			
メッセージ種別ごとのメッセージテーブルを com_cfg.c に出力します。 DIRECTION 属性が RECEIVED で且つ、MESSAGEPROPRETY が STATIC,ZERO,DYNAMIC の場合に MESSAGE オブジェクト名称を出力します。また、配列の終端には MSG_NULL を出力します。			
出力内容			
const MessageIdentifier sndlist_オブジェクト名称[] = {..}; const MessageIdentifier *sndlist_rcv_ipdu_norm[TNUM_RCV_IPDU_NORM] = {..}; const MessageIdentifier　sndlist_rcv_ipdu_zero[TNUM_RCV_IPDU_ZERO] = {..}; const MessageIdentifier sndlist_オブジェクト名称[] = {..}; const MessageIdentifier *sndlist_rcv_ipdu_dyna[TNUM_RCV_IPDU_DYNA] = {..};			

5.1.2.3. 送受信ネットワークメッセージのビットオーダ

OIL		設定値	出力ファイル
オブジェクト	属性	TRIGGERD, PENDING	com_cfg.c
NETWORKMESSAGE	TRANSFERPROPERTY		
	DIRECTION	SENT	
説明			
メッセージごとのビットオーダ情報を com_cfg.c に出力します。 DIRECTION 属性が SENT で且つ、TRANSFERPROPERTY が TRIGGERD,PENDING の場合に出力先配列を指定して出力します。			
出力内容			
const UINT8 bitorder_snd_ex_trig_norm[TNUM_SND_EX_TRIG_NORM] = {..}; const UINT8 bitorder_snd_ex_pend_norm[TNUM_SND_EX_PEND_NORM] = {..}; const UINT8 bitorder_snd_ex_trig_dyna[TNUM_SND_EX_TRIG_DYNA] = {..}; const UINT8 bitorder_snd_ex_pend_dyna[TNUM_SND_EX_PEND_DYNA] = {..};			

OIL		設定値	出力ファイル
オブジェクト	属性	STATIC, DYNAMIC	com_cfg.c
NETWORKMESSAGE	MESSAGEPROPRETY		

	DIRECTION	RECEIVED	
説明			
DIRECTION 属性が RECEIVED で且つ、MESSAGEPROPRETY が STATIC,DYNAMIC の場合に出力先配列を指定して出力します。			
出力内容			
const UINT8 bitorder_rcv_ipdu_norm[TNUM_RCV_IPDU_NORM] = {..}; const UINT8 bitorder_rcr_ipdu_dyna[TNUM_RCV_IPDU_DYNA] = {..};			

5.1.2.4. バイトオーダ

OIL		設定値	出力ファイル
オブジェクト	属性	STATIC	com_cfg.c
NETWORKMESSAGE	MESSAGEPROPRETY		
	DIRECTION	RECEIVED	
説明			
バイトオーダの実行有無が記載されたテーブルを com_cfg.c に出力します。 NETWORKMESSAGE オブジェクトの MESSAGEPROEPRTY 属性が STATIC の場合、有効な値が出力されます。			
出力内容			
const UINT8 byteorder_rcv_ipdu_norm[TNUM_RCV_IPDU_NORM] = {..};			

5.1.2.5. SIZEINBITS の値

OIL		設定値	出力ファイル
オブジェクト	属性	STATIC, DYNAMIC	com_cfg.c
NETWORKMESSAGE	MESSAGEPROPRETY		
	SIZEINBITS	数値	
	MAXIMUMSIZEINBITS	数値	
説明			
SIZEINBITS の値を com_cfg.c に出力します。 MESSAGEPROPERTY 属性の値を元に出力先配列と出力対象を以下のように区別します。 MESSAGEPROPERTY 属性が STATIC の場合、SIZEINBITS の値を出力します。 MESSAGEPROPERTY 属性が DYNAMIC の場合、MAXIMUMSIZEINBITS の値を出力します。 また、バイトサイズの配列も同様に出力します。			
出力内容			

const UINT32 bitsize_rcv_ipdu_norm[TNUM_RCV_IPDU_NORM] = {..};
const UINT32 bitsize_rcv_ipdu_dyna[TNUM_RCV_IPDU_DYNA] = {..};
const UINT32 bytesize_rcv_ipdu_norm[TNUM_RCV_IPDU_NORM] = {..};

5.1.2.6. BITPOSITION の値

OIL		設定値	出力ファイル
オブジェクト	属性	数値	com_cfg.c
NETWORKMESSAGE	BITPOSITION		
説明			
BITPOSITION の値を com_cfg.c に出力します。			
出力内容			
const UINT32 offset_rcv_ipdu_norm[TNUM_RCV_IPDU_NORM] = {..} const UINT32 offset_rcv_ipdu_dyna[TNUM_RCV_IPDU_DYNA] = {..}			

5.1.3. COM 管理情報（COM）

5.1.3.1. COM アプリケーションモード

OIL		設定値	出力ファイル
オブジェクト	属性	名称	com_cfg.h
COM	COMAPPMODE		
説明			
COMAPPMODE で指定された名前を順に com_cfg.h に出力します。 また、出力タイミングで ID を振り分けます。			
出力内容			
#define COM アプリケーションモード名称　((COMApplicationModeType)(1u << 0 からの数値)) #define COMAPPMODE (COM アプリケーションモード名称) #define TNUM_COMAPPMODE　COM アプリケーションモード数 #if (TNUM_COMAPPMODE > MAX_COMAPPMODE) #error "Total number of COMApplicationMode exceed the maximal value." #endif			

5.1.3.2. フックルーチンのチェック

OIL		設定値	出力ファイル
オブジェクト	属性	TURE／FALSE	com_cfg.c
COM	COMERRORHOOK		

説明
COMERRORHOOK が TURE のときに以下の定義を出力します。 COMERRORHOOK が FALSE の時はインラインアセンブラコードを com_cfg.c に出力します。
出力内容

・COMERRORHOOK = TRUEの場合
#define USE_COMERRORHOOK
“#define USE_COMSTARTCOMEXTENSION
 	
・COMERRORHOOK = FALSEの場合
asm(".glb $COMErrorHook");
asm("$COMErrorHook .equ 0");
asm(” .glb _StartComExtension”);
asm(“_StartComExtension .equ 0”);

5.1.4.　IPDU 管理情報（IPDU）

5.1.4.1.　実装ドライバオプション

OIL		設定値	出力ファイル
オブジェクト	属性	CAN／LIN	com_cfg.h
IPDU	LAYERUSED		
説明			
LAYERUSED の値を com_cfg.h に出力します。			
出力内容			
#define UL_SUPPORT_設定値			

5.1.4.2.　IPDU オブジェクト ID 出力

OIL		設定値	出力ファイル
オブジェクト	属性	－	com_cfg.h
IPDU	LAYERUSED		
	IPDUPROPERTY	－	
説明			
レイヤー、送受信種別を加味したオブジェクト ID を com_cfg.h に出力します。			
出力内容			
#define オブジェクト名称　番号			

5.1.4.3.　IPDU オブジェクト数の出力

OIL	設定値	出力ファイル

オブジェクト	属性	CAN／LIN	com_cfg.h
IPDU	LAYERUSED		
	IPDUPROPERTY	SENT／RECEIVE	
説明			
レイヤー、送受信種別を加味した IPDU オブジェクトの数を com_cfg.h に出力します。			
出力内容			
#define TNUM_LAYEUSED 設定値_IPDUPROPERTY 設定値　　数値			

5.1.4.4. IPDU オフセット ID の出力

OIL		設定値	出力ファイル
オブジェクト	属性	CAN／LIN	com_cfg.h
IPDU	LAYERUSED		
	IPDUPROPERTY	SENT／RECEIVE	
説明			
com_cfg.c で出力している ID テーブルアクセス時のオフセット値を com_cfg.h に出力します。			
出力内容			
#define OFFSET_ LAYEUSED 設定値_IPDU_ IPDUPROPERTY 設定値　　オフセット値			

5.1.4.5. CAN ID 割付オプション

OIL		設定値	出力ファイル
オブジェクト	属性	CAN	com_cfg.h
IPDU	LAYERUSED		
	IPDUPROPERTY	SENT／RECEIVE	
説明			
CAN プロトコル指定が合わせて１３個以下なら CAN_STATIC_ALL_ID を出力します。 送信 CAN プロトコル指定だけで且つ１３個以下なら CAN_STATIC_SEND_ID を出力します。 上記以外なら CAN_DYNAMIC_ALL_ID を出力します。尚、CAN_DYNAMIC_ALL_ID の場合は、バッファ領域出力用に CAN_DYNAMIC_BUFF_DIFF_NUM の定義を出力します。			
出力内容			
#define CAN_STATIC_ALL_ID #define CAN_STATIC_SEND_ID #define CAN_DYNAMIC_ALL_ID #define CAN_DYNAMIC_BUFF_DIFF_NUM　　定義数(13 個以上)			

5.1.4.6. IPDU 数

OIL		設定値	出力ファイル
オブジェクト	属性	SENT／RECEIVE	com_cfg.h
IPDU	IPDUPROPERTY		
説明			
送信 IPDU、受信 IPDU の数を com_cfg.h に出力します。			
出力内容			
#define TNUM_SND_IPDU　定義数 #define TNUM_RCV_IPDU　定義数			

5.1.4.7. タイマ ID 出力

OIL		設定値	出力ファイル
オブジェクト	属性	SENT／RECEIVE	com_cfg.h
IPDU	IPDUPROPERTY		
説明			
タイマ ID 関連、及びタイマオブジェクト数を com_cfg.h に出力します。 TIMETICK は固定 0 出力とします。			
出力内容			
#define timer_snd_ctrl_オブジェクト名称　ID #define timer_snd_periodic_オブジェクト名称　ID + TNUM_SND_IPDU #define timer_rcv_ctrl_オブジェクト名称　ID #define COM_TIMETICK　0 #define TNUM_TIMER　オブジェクト定義数			

5.1.4.8. ネットワーク ID 変換テーブル

OIL		設定値	出力ファイル
オブジェクト	属性	CAN	com_cfg.c
IPDU	LAYERUSED		
	IPDUPROPERTY	SENT／RECEIVE	
	NETWORKID	数値	
説明			

レイヤー毎にネットワーク ID のテーブルを com_cfg.c に出力します。 LAYERUSED 属性の文字列と IPDUPROPERTY 属性の SENT／RECIVED で文字列を作成します。 テーブルに入る値は、NETWORKID の値。 can_id_send[IPDUid-IPDU_CAN_SEND_OFFSET]でアクセス可能な順番で値を設定します。 次の仕様で並べ替えて ID を出力します。 　IPDUPROPERTY 属性の値(SENT/RECEIVED)を元に送信、受信を区別します。 　NETWORKID 属性値の昇順で並べます。 配列に出力する値は、NETWORKID 属性値にパリティを付加したものとします。 パリティは NETWORKID 属性値に 0x03f でマスクした値です。
出力内容
const CANType can_id_"snd or rcv" [TNUM_"SND"or"RCV"_CAN]= {NETWORKID 値}; const UINT8 can_type_"snd"or"rcv"[TNUM_"SND"or"RCV"_CAN] = { 0x00,0x00・・・ };

5.1.4.9. ULPDU バッファアドレス配列

OIL		設定値	出力ファイル
オブジェクト	属性	SENT／RECEIVE	com_cfg.h
IPDU	IPDUPROPERTY		
	SIZEINBITS	数値	
説明			
IPDU の ID 順にアクセスできるような順番で ULPDU 受信のバッファ名を com_cfg.c に出力します。 SIZEINBITS 属性の値を、8 で割って切り上げた値(バイト変換サイズ)で出力します。 また、ULPDU 送信のバッファ名を com_cfg.c に出力します。			
出力内容			
UINT8* const　ulpdu_rcv_buf_addr[TNUM_RCV_IPDU] = {ULPDU 受信バッファ名}; UINT8 ulpdu_rcv_dlc[TNUM_RCV_IPDU]; UINT8 ulpdu_snd_buf_IPDU オブジェクト名称[SIZEINBITS 属性値]; UINT8 *const ulpdu_snd_buf_addr[TNUM_SND_IPDU] = {　}; UINT8 ulpdu_snd_dlc[TNUM_SND_IPDU];			

5.1.4.10. ドライバ関数テーブル

OIL		設定値	出力ファイル
オブジェクト	属性	SENT／RECEIVE	com_cfg.h
IPDU	IPDUPROPERTY		
	LAYERUSED	CAN	

説明
ipduproperty と layrused の値を元に出力先配列、及びドライバ関数名を区別して com_cfg.c に出力します。 組み合わせ例： IPDUPROPERTY＝"SENT" and LAYERUSED＝"CAN" 送信 CAN ドライバ関数名＝can_snd 送信 CAN キャンセルドライバ関数名＝can_snd_cancel IPDUPROPERTY＝"RECEIVED" and LAYERUSED＝"CAN" 受信 CAN ドライバ関数名＝can_rcv
出力内容
const ULSndFunc snd_ul_func [TNUM_SND_IPDU] = {..}; const ULSndCancelFunc snd_cancel_ul_func [TNUM_SND_IPDU] = {..}; const ULRcvFunc　rcv_ul_func [TNUM_RCV_IPDU] = {..};

5.1.4.11. IPDU サイズ

OIL		設定値	出力ファイル
オブジェクト	属性	数値	com_cfg.c
IPDU	SIZEINBITS		
説明			
全ての IDPU オブジェクトのバッファ領域をバイトサイズで com_cfg.c に出力します。			
出力内容			
UINT8 ipdubuf_オブジェクト名称[SIZEINBITS バイト変換サイズ];			

5.1.4.12. メッセージバッファの値

OIL		設定値	出力ファイル
オブジェクト	属性	－	com_cfg.c
IPDU	－		
説明			
IPDU サイズの変数名を com_cfg.c に出力します。			
出力内容			
UINT8* const msgbuf_snd_ipdu[TNUM_SND_IPDU] = {送信 IPDU サイズ変数名}; UINT8* const msgbuf_rcv_ipdu[TNUM_RCV_IPDU] = {受信 IPDU サイズ変数名};			

5.1.4.13. IPDU 数

OIL		設定値	出力ファイル
オブジェクト	属性	－	com_cfg.c

IPDU	－		
説明			
IPDU のオブジェクト数を com_cfg.c に出力します。			
出力内容			
const UINT8 tnum_snd_ipdu = TNUM_SND_IPDU; const UINT8 tnum_rcv_ipdu = TNUM_RCV_IPDU;			

5.1.4.14. IPDU 用逆引きテーブル

OIL		設定値	出力ファイル
オブジェクト	属性	SENT／RECEIVE	com_cfg.c
IPDU	IPDUPROPERTY		
NETWORKMESSAGE	－	オブジェクト名称	
説明			
IPDU を参照している NETWORKMESSAGE オブジェクト名称を com_cfg.c に出力します。 IPDUPROPERTY の SENT、RECEIVED で出力先配列を判断します。 配列の末尾には MSG_NULL を出力します。			
出力内容			
const MessageIdentifier sndlist_snd_ipdu_NETWORKMESSEGA オブジェクト名称[] = {..}; const MessageIdentifier *sndlist_snd_ipdu[TNUM_SND_IPDU] = {..};			

5.1.4.15. IPDU 用逆引きの値

OIL		設定値	出力ファイル
オブジェクト	属性	SENT／RECEIVE	com_cfg.c
IPDU	IPDUPROPERTY		
NETWORKMESSAGE	－	オブジェクト名称	
説明			
IPDU を参照している NETWORKMESSAGE のオブジェクト名称を com_cfg.c に出力します。 IPDUPROPERTY の SENT、RECEIVED で出力先配列を判断します。 配列の末尾には MSG_NULL を出力します。			
出力内容			
const MessageIdentifier sndlist_rcv_ipdu_NETWORKMESSAGE オブジェクト名称[] = {..}; const MessageIdentifier *sndlist_rcv_ipdu[TNUM_RCV_IPDU] = {..};			

5.1.4.16. SIZEINBITS の値

OIL		設定値	出力ファイル
オブジェクト	属性	数値	com_cfg.c

IPDU	SIZEINBITS		
説明			
SIZEINBITS の値を配列にして com_cfg.c に出力します。 SIZEINBITS の値をバイト値に変換し出力します。			
出力内容			
const UINT32 bitsize_snd_ipdu[TNUM_SND_IPDU] = {設定値}; const UINT32 bitsize_rcv_ipdu[TNUM_RCV_IPDU] = {設定値}; const UINT32 bytesize_snd_ipdu[TNUM_SND_IPDU] = {バイト設定値}; const UINT32 bytesize_rcv_ipdu[TNUM_RCV_IPDU] = {バイト設定値};			

5.1.4.17. TIMEPERIOD の値

OIL		設定値	出力ファイル
オブジェクト	属性	数値	com_cfg.c
IPDU	TIMEPERIOD		
説明			
TIMEPERIOD の値を配列にして com_cfg.c に出力します。 TIMEPERIOD が条件により使用していない場合は、は 0 を出力します。			
出力内容			
const UINT64 timeperiod_snd_ipdu[TNUM_SND_IPDU] = {設定値};			

5.1.4.18. TIMEOFFSET の値

OIL		設定値	出力ファイル
オブジェクト	属性	数値	com_cfg.c
IPDU	TIMEOFFSET		
説明			
TIMEOFFSET の値を配列にして出力します。 TIMEOFFSET が条件により使用していない場合は 0 を出力します。			
出力内容			
const UINT64 timeoffset_snd_ipdu[TNUM_SND_IPDU] = {設定値};			

5.1.4.19. MINIMUMDELAYTIME の値

OIL		設定値	出力ファイル
オブジェクト	属性	数値	com_cfg.c
IPDU	MINIMUMDELAYTIME		
説明			

MINIMUMDELAYTIME の値を配列にして com_cfg.c に出力します。 MINIMUMDELAYTIME が条件により使用していない場合は 0 を出力します。
出力内容
const UINT64 delaytime_snd_ipdu[TNUM_SND_UPDU] = {設定値};

5.1.4.20. TIMEOUT の値

OIL		設定値	出力ファイル
オブジェクト	属性	数値	com_cfg.c
IPDU	TIMEOUT		
	IPDUPROPERTY	SENT／RECEIVE	
説明			
TIMEOUT の値を配列にして com_cfg.c に出力します。 IPDUPROPERTY を参照して SND/RCV を区別します。 TIMEOUT が条件により使用されていない場合は 0 を出力します。			
出力内容			
const UINT64 timeout_snd_ipdu[TNUM_SND_IPDU] = {設定値}; const UINT64 timeout_rcv_ipdu[TNUM_RCV_IPDU] = {設定値};			

5.1.4.21. FIRSTTIMEOUT サイズ

OIL		設定値	出力ファイル
オブジェクト	属性	数値	com_cfg.c
IPDU	FIRSTTIMEOUT		
	IPDUPROPERTY	SENT／RECEIVE	
説明			
FIRSTTIMEOUT の値を配列にして com_cfg.c に出力します。 IPDUPROPERTY を参照して SND/RCV を区別します。 FIRSTTIMEOUT が条件により使用されていない場合は 0 を出力します。			
出力内容			
const UINT64 firsttimeout_snd_ipdu[TNUM_SND_IPDU] = {設定値}; const UINT64 firsttimeout_rcv_ipdu[TNUM_RCV_IPDU] = {設定値};			

5.1.4.22. CALLOUT

OIL		設定値	出力ファイル
オブジェクト	属性	シンボル名称	com_cfg.c
IPDU	IPDUCALLOUT		
	IPDUPROPERTY	SENT／RECEIVE	

説明
IPDUCALLOUT の値を並べて com_cfg.c に出力します。 IPDUPROPERTY を参照して SND/RCV を区別します。 定義が無いものは、デフォルト値 CALLOUTFP_NULL を出力します。
出力内容
const CALLOUTFP callout_snd_ipdu[TNUM_SND_IPDU] = {シンボル名称}; const CALLOUTFP callout_rcv_ipdu[TNUM_RCV_IPDU] = {シンボル名称};

5.1.4.23. LAYERUSED の値

OIL		設定値	出力ファイル
オブジェクト	属性	CAN／LIN	com_cfg.c
IPDU	LAYERUSED		
説明			
LAYERUSED の値を並べて com_cfg.c に出力します。			
出力内容			
const UINT8 *layerused_snd_ipdu[TNUM_SND_IPDU] = {設定値}; const UINT8 *layerused_rcv_ipdu[TNUM_RCV_IPDU] = {設定値};			

5.1.4.24. TRANSMISSIONMODE の値

OIL		設定値	出力ファイル
オブジェクト	属性	数値	com_cfg.c
IPDU	TRANSMISSIONMODE		
説明			
TRANSMISSIONMODE の値を com_cfg.c に出力します。			
出力内容			
const UINT8 transmode_snd_ipdu[TNUM_SND_IPDU] = {..};			

5.1.4.25. ワーク変数の準備

OIL		設定値	出力ファイル
オブジェクト	属性	数値	com_cfg.c
IPDU	SIZEINBITS		
	IPDUPROPERTY	SENT／RECEIVE	
説明			
IPDUPROPERTY 属性値によって、SIZEINBITS 属性値の最大値を 4 で割って切り上げた値を com_cfg.c に出力します。			
出力内容			

UINT32 msgbuf_snd_tmp[設定値/4];
UINT32 msgbuf_snd_after_swapbytes[設定値/4];
const UINT32 bytesize_snd_max =設定値;
UINT32 ipdubuf_rcv_tmp[設定値/4];
UINT32 msgbuf_rcv_after_swapbytes[設定値/4];
const UINT32 bytesize_rcv_max =設定値;

5.1.4.26. IPDU 送信管理

OIL		設定値	出力ファイル
オブジェクト	属性	―	com_cfg.c
IPDU	―		
説明			
IPDU 送信バッファを com_cfg.c に出力します。			
出力内容			
UINT8 flg_snd_ipdu[TNUM_SND_IPDU];			

5.1.4.27. IPDU 送受信関連タイマ ID

OIL		設定値	出力ファイル
オブジェクト	属性	SENT／RECEIVE	com_cfg.c
IPDU	IPDUPROPERTY		
説明			
IPDUPROPERTY 属性値によって、オブジェクト名称を com_cfg.c に出力します。			
出力内容			
TimerType timer_snd_ctrl[TNUM_SND_IPDU] = {..}; TimerType timer_snd_periodic[TNUM_SND_IPDU] = {..}; TimerType timer_rcv_ctrl[TNUM_RCV_IPDU] = {..};			

5.1.4.28. タイマ関連コントロールブロック

OIL		設定値	出力ファイル
オブジェクト	属性	―	com_cfg.c
IPDU	―		
説明			
タイマ関連のコントロールブロックを com_cfg.c に出力します。			
出力内容			

const UINT8 tnum_timer = TNUM_TIMER; const TickCounterType timinib_tickid[TNUM_TIMER] = {COM_TIMETICK}; TimerType timcb_next[TNUM_TIMER]; TimerType timcb_prev[TNUM_TIMER]; TimerTickType timcb_timval[TNUM_TIMER]; TimerTickType timcb_cycle[TNUM_TIMER]; TIMFP timcb_cback[TNUM_TIMER]; UINT8 timcb_param[TNUM_TIMER];

5.1.4.29. CANID 数

OIL		設定値	出力ファイル
オブジェクト	属性	－	com_cfg.c
IPDU	－		
説明			
CANID の使用数を com_cfg.c に出力します。			
出力内容			
const UINT8 tnum_snd_can = TNUM_SND_CAN; const UINT8 tnum_rcv_can = TNUM_RCV_CAN;			

5.1.4.30. CAN 通信種別情報

OIL		設定値	出力ファイル
オブジェクト	属性	－	com_cfg.c
IPDU	－		
説明			
IPDU に格納されるデータが、分割データ、分割不要データどちらのデータかを com_cfg.c に出力します。			
出力内容			
const UINT8 can_snd_msgkind[TNUM_SND_CAN] = {　}; const UINT8 can_rcv_msgkind[TNUM_RCV_CAN] = {　};			

5.1.4.31. CAN ワークバッファ

OIL		設定値	出力ファイル
オブジェクト	属性	－	com_cfg.c
IPDU	－		
説明			
CAN 用ワークバッファを com_cfg.c に出力します。			

出力内容
IPDUType ulcan_snd_queue[TNUM_SND_CAN]; ULCANWorkInfo ulcan_rcv_info[TUNM_RCV_CAN];

5.1.5. NM 管理情報（NM）

5.1.5.1. NM の規模出力

OIL		設定値	出力ファイル
オブジェクト	属性	MIN_NM	nm_cfg.h
NM	NMSCALEABILITY	MAX_NM	
出力内容			
#define　NM_SCALEBILITY_TYPE　設定値			

5.1.5.2. OpCode システムマスク出力

OIL		設定値	出力ファイル
オブジェクト	属性	0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80	nm_cfg.h
NM	OPCODE_ALIVE, OPCODE_RING, OPCODE_LIMPHONE, OPCODE_SLEEPIND, OPCODE_SLEEPACK		
説明			
OPCODE_ALIVE、OPCODE_RING、OPCODE_LIMPHONE、OPCODE_SLEEPIND、OPCODE_SLEEPACK の値の論理和を出力する			
出力内容			
#define OPCODE_SYSTEM_MASK　論理和値			

5.1.5.3. OpCode ユーザマスク出力

OIL		設定値	出力ファイル
オブジェクト	属性	0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80	nm_cfg.h
NM	OPCODE_ALIVE, OPCODE_RING, OPCODE_LIMPHONE, OPCODE_SLEEPIND, OPCODE_SLEEPACK		
説明			

OPCODE_ALIVE、OPCODE_RING、OPCODE_LIMPHONE、OPCODE_SLEEPIND、OPCODE_SLEEPACK の値の否定論理和を出力する
出力内容
#define OPCODE_SYSTEM_MASK　否定論理和値

5.1.5.4. OPCODE の Alive メッセージビット出力

OIL		設定値	出力ファイル
オブジェクト	属性	0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80	nm_cfg.h
NM	OPCODE_ALIVE		
出力内容			
#define　NMMSG_OPCODE_ALIVE　(NmOpcodeType)(OPCODE_ALIVE の値)			

5.1.5.5. OPCODE の Ring メッセージビット出力

OIL		設定値	出力ファイル
オブジェクト	属性	0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80	nm_cfg.h
NM	OPCODE_RING		
出力内容			
#define　NMMSG_OPCODE_RING　(NmOpcodeType)(OPCODE_RING の値)			

5.1.5.6. OPCODE の LimpHome メッセージビット出力

OIL		設定値	出力ファイル
オブジェクト	属性	0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80	nm_cfg.h
NM	OPCODE_LIMPHONE		
出力内容			
#define　NMMSG_OPCODE_LIMPHONE　(NmOpcodeType)(OPCODE_LIMPHONE の値)			

5.1.5.7. OPCODE の Sleep.ind 有無ビット出力

OIL		設定値	出力ファイル
オブジェクト	属性	0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80	nm_cfg.h
NM	OPCODE_SLEEPIND		
出力内容			
#define　NMMSG_OPCODE_SLEEPIND　(NmOpcodeType)(OPCODE_SLEEPIND の値)			

5.1.5.8. OPCODE の Sleep.ack 有無ビット出力

OIL		設定値	出力ファイル
オブジェクト	属性	0x01, 0x02, 0x04, 0x08, 0x10,	nm_cfg.h

NM	OPCODE_SLEEPACK	0x20, 0x40, 0x80	
出力内容			
#define　NMMSG_OPCODE_SLEEPACK　(NmOpcodeType)(OPCODE_SLEEPACK の値)			

5.1.5.9.　OPCODE のメッセージ種別マスク出力

OIL		設定値	出力ファイル
オブジェクト	属性	―	nm_cfg.h
NM	―		
出力内容			
#define NMMSG_OPCODE_MSGMASK　(NmOpcodeType)(NMMSG_OPCODE_ALIVE　| NMMSG_OPCODE_RING | NMMSG_OPCODE_LIMPHONE)			

5.1.5.10. OPCODE の Sleep ビットマスク出力

OIL		設定値	出力ファイル
オブジェクト	属性	―	nm_cfg.h
NM	―		
出力内容			
#define NMMSG_OPCODE_SLEEPMASK (NmOpcodeType)(NMMSG_OPCODE_SLEEPIND | NMMSG_OPCODE_SLEEPACK)			

5.1.5.11. Sleep/Awake 要求メッセージ受信時アプリケーション通知の有無出力

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE	nm_cfg.h
NM	NMSLEEPIND		
説明			
TRUE の場合のみ出力する			
出力内容			
#define　USE_NM_SLEEPIND			

5.1.5.12. バスイニットルーチン参照の有無出力

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE	nm_cfg.h
NM	BUSINIT_ROUTINE		
説明			
TRUE の場合のみ出力する			
出力内容			

#define USE_NM_BUS_INITIALIZE

5.1.5.13. バスシャットダウンルーチン参照の有無出力

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE	nm_cfg.h
NM	BUSSHUTDOWN_ROUTINE		
説明			
TRUE の場合のみ出力する			
出力内容			
#define USE_NM_BUS_SHUTDOWN			

5.1.5.14. バス Awake ルーチン参照の有無出力

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE	nm_cfg.h
NM	BUSAWAKE_ROUTINE		
説明			
TRUE の場合のみ出力する			
出力内容			
#define USE_NM_BUS_AWAKE			

5.1.5.15. バススリープルーチン参照の有無出力

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE	nm_cfg.h
NM	BUSSLEEP_ROUTINE		
説明			
TRUE の場合のみ出力する			
出力内容			
#define USE_NM_BUS_SLEEP			

5.1.5.16. バスリスタートルーチン参照の有無出力

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE	nm_cfg.h
NM	BUSRESTART_ROUTINE		
説明			
TRUE の場合のみ出力する			

出力内容
#define USE_NM_BUS_RESTART

5.1.5.17. IDBase 出力

OIL		設定値	出力ファイル
オブジェクト	属性	0x00000000 ～ 0x000007F8	nm_cfg.h
NM	NMMSG_ID_BASE		
出力内容			
#define NMMSG_ID_BASE　　(NodeIdType)(NMMSG_ID_BASE の値)			

5.1.5.18. Window マスク出力

OIL		設定値	出力ファイル
オブジェクト	属性	0x00000700, 0x00000780, 0x000007C0, 0x000007E0, 0x000007F0, 0x000007F8	nm_cfg.h
NM	NMMSG_WINDOW_MASK		
出力内容			
#define NMMSG_ID_BASE　　(NodeIdType)(NMMSG_WINDOW_MASK の値)			

5.1.5.19. ネットワーク管理ノード数出力

OIL		設定値	出力ファイル
オブジェクト	属性	オブジェクト参照	nm_cfg.c
NM	NMNODE		
出力内容			
#define NM_TNUM_NODE_MAX　NMNODE の数 const UINT8 nm_tnum_node_max = (UINT8)NM_TNUM_NODE_MAX;			

5.1.5.20. ネットワーク管理ノードリスト出力

OIL		設定値	出力ファイル
オブジェクト	属性	オブジェクト参照	nm_cfg.c
NM	NMNODE		
NMNODE	NODEID	数値	
説明			
NMNODE で参照関係 NMNODE オブジェクトの NODEID の値からリストを作成する			
出力内容			

const NodeIdType nm_targetconfig_node[NM_TNUM_NODE_MAX] ＝{ NM_ID_NMNODE の値 1, NM_ID_NMNODE の値 2,・・};

5.1.5.21. 自ノードＩＤ出力

OIL		設定値	出力ファイル
オブジェクト	属性	オブジェクト参照	nm_cfg.c
NM	MYNODE		
NMNODE	NODEID	数値	
説明			
NMNODE オブジェクトの NODEID の値を出力			
出力内容			
#define　NM_NODE_ID　NM_ID_MYNODE の値 const NodeIdType nm_node_id = (NodeIdType)NM_NODE_ID;			

5.1.5.22. ネットワークの ID 出力

OIL		設定値	出力ファイル
オブジェクト	属性	数値	nm_cfg.c
NM	NETWORKID		
出力内容			
#define　NM_NETWORK_ID　　NETWORKID の値 const NetIdType nm_network_id = (NetIdType)NM_NETWORK_ID;			

5.1.5.23. コンフィギュレーションリスト出力

OIL		設定値	出力ファイル
オブジェクト	属性	―	nm_cfg.c
NM	―		
出力内容			
ConfigType nm_config_table[NM_TNUM_NODE_MAX];			

5.1.5.24. コンフィグマスクリスト出力

OIL		設定値	出力ファイル
オブジェクト	属性	数値	nm_cfg.c
NM	NODEID		
出力内容			
const ConfigType nm_cmask[NM_TNUM_NODE_MAX] ＝{ NM_CMASK_NMNODE の値 1, NM_CMASK_NMNODE の値 2,・・};			

5.1.5.25. TTyp タイマ出力

OIL		設定値	出力ファイル
オブジェクト	属性	70～110	nm_cfg.c
NM	TTYP		
出力内容			
#define　NM_TTYP_TIME　　TTYP の値 const UINT32 nm_ttyp_time = (UINT32)NM_TTYP_TIME;			

5.1.5.26. TMAX タイマ出力

OIL		設定値	出力ファイル
オブジェクト	属性	220～284	nm_cfg.c
NM	TMAX		
出力内容			
#define　NM_TMAX_TIME　　TMAX の値 const UINT32 nm_tmax_time = (UINT32)NM_TMAX_TIME;			

5.1.5.27. TError タイマ出力

OIL		設定値	出力ファイル
オブジェクト	属性	1～65535	nm_cfg.c
NM	TERROR		
出力内容			
#define　NM_TERROR_TIME　　TERROR の値 const UINT32 nm_terror_time = (UINT32)NM_TERROR_TIME;			

5.1.5.28. TWaitBusSleep タイマ出力

OIL		設定値	出力ファイル
オブジェクト	属性	1～65535	nm_cfg.c
NM	TWAITBUSSLEEP		
出力内容			
#define　NM_TWAITBUSSLEEP_TIME　　TWAITBUSSLEEP の値 const UINT32 nm_twaitbussleep_time = (UINT32)NM_TWAITBUSSLEEP_TIME;			

5.1.5.29. TTx タイマ出力

OIL	設定値	出力ファイル

オブジェクト	属性	1～65535	nm_cfg.c
NM	TTX		
出力内容			
#define　NM_TTX_TIME　TTX の値 const UINT32 nm_ttx_time = (UINT32)NM_TTX_TIME;			

5.1.5.30. 受信不能カウンタ閾値出力

OIL		設定値	出力ファイル
オブジェクト	属性	1～255	nm_cfg.c
NM	NMRXLIMIT		
出力内容			
#define　NM_RX_LIMIT_COUNTNMRXLIMIT の値 const UINT8 nm_rx_limit_count = (UINT8)NM_RX_LIMIT_COUNT;			

5.1.5.31. 送信不能カウンタ閾値出力

OIL		設定値	出力ファイル
オブジェクト	属性	1～255	nm_cfg.c
NM	NMTXLIMIT		
出力内容			
#define　NM_TX_LIMIT_COUNTNMTXLIMIT の値 const UINT8 nm_tx_limit_count = (UINT8)NM_TX_LIMIT_COUNT;			

5.1.5.32. NM ステータスマスク出力

OIL		設定値	出力ファイル
オブジェクト	属性	数値	nm_cfg.c
NM	NMSMASK		
	PRESENT_NETWORK	TRUE／FALSE	
	OPERATION_MODE_IF	TRUE／FALSE	
	NMACTIVE	TRUE／FALSE	
	NMON	TRUE／FALSE	
	NMLIMPHOME	TRUE／FALSE	
	NMBUSSLEEP	TRUE／FALSE	
	NMTWBS_NOR_LIMP	TRUE／FALSE	
	RING_DATA_ALLOWED	TRUE／FALSE	
	GMODE_BUSSLEEP	TRUE／FALSE	

	NMWAIT_BUSSLEEP	TRUE／FALSE	
説明			
NMSMASK に TRUE を設定した場合、下記 OIL 属性で TRUE に設定した属性の define 値を論理和した値を出力します。NMSMASK が FALSE もしくは、下記 OIL 属性が全て FALSE の場合は define 値として NM_NST_NON を出力します。			
出力内容			
#define NM_SMASK　　(define 値 1 | define 値 2 | ‥) const NetworkStatusType nm_smask = (NetworkStatusType)NM_SMASK;			

5.1.5.33. 受信 NM メッセージリングデータフィールドサイズ出力

OIL		設定値	出力ファイル
オブジェクト	属性	数値	nm_cfg.c
NM	TNUM_RINGDATA		
出力内容			
#define NM_TNUM_RING_DATA ((TNUM_RINGDATA の値+ 7) / 8) const UINT8 nm_tnum_ring_data = (UINT8)NM_TNUM_RING_DATA;			

5.1.5.34. 受信 NM メッセージデリングデータバッファ出力

OIL		設定値	出力ファイル
オブジェクト	属性	数値	nm_cfg.c
NM	TNUM_RINGDATA		
説明			
TNUM_RINGDATA の値が 0 の場合、受信 NM メッセージデリングデータバッファの配列数を+1 します。			
出力内容			
(i) TNUM_RINGDATA > 0 の場合 RingDataType nm_ringdata_buf[NM_TNUM_RING_DATA]; (ii) TNUM_RINGDATA = 0 の場合 RingDataType nm_ringdata_buf[NM_TNUM_RING_DATA +1];			

5.1.5.35. 受信 NMPDU バッファキューサイズ出力

OIL		設定値	出力ファイル
オブジェクト	属性	2～64	nm_cfg.c
NM	NMNUMRCVQUE		
出力内容			

#define NM_TNUM_RCV_QUE (NMNUMRCVQUE の値) const UINT8 nm_tnum_rcv_que = (UINT8)NM_TNUM_RCV_QUE;

5.1.5.36. 受信 NMPDU バッファキューサイズ出力

OIL		設定値	出力ファイル
オブジェクト	属性	2～64	nm_cfg.c
NM	NMNUMRCVQUE		
出力内容			
#define NM_TNUM_RCV_QUE (NMNUMRCVQUE の値)			

5.1.5.37. 受信 NMPDU バッファ送信元出力

OIL		設定値	出力ファイル
オブジェクト	属性	－	nm_cfg.c
NM	－		
出力内容			
NodeIdType nmmsg_rcv_srcid[NM_TNUM_RCV_QUE];			

5.1.5.38. 受信 NMPDU バッファ送信先出力

OIL		設定値	出力ファイル
オブジェクト	属性	－	nm_cfg.c
NM	－		
出力内容			
NodeIdType nmmsg_rcv_destid[NM_TNUM_RCV_QUE];			

5.1.5.39. 受信 NMPDU バッファ OpCode

OIL		設定値	出力ファイル
オブジェクト	属性	－	nm_cfg.c
NM	－		
出力内容			
NmOpcodeType nmmsg_rcv_opcode[NM_TNUM_RCV_QUE];			

5.1.5.40. 受信 NMPDU バッファデータバッファ出力

OIL		設定値	出力ファイル
オブジェクト	属性	2～64	nm_cfg.c
NM	NMNUMRCVQUE		
出力内容			

static RingDataType nmmsg_rcv_ringdata1[NM_TNUM_RING_DATA]; static RingDataType nmmsg_rcv_ringdata2[NM_TNUM_RING_DATA]; 　　: static RingDataType nmmsg_rcv_ringdata「NMNUMRCVQUE の値」[NM_TNUM_RING_DATA];

5.1.5.41. 受信 NMPDU バッファデータフィールド出力

OIL		設定値	出力ファイル
オブジェクト	属性	2～64	nm_cfg.c
NM	NMNUMRCVQUE		
出力内容			
const RingDataType* nmmsg_rcv_datafield[NM_TNUM_RCV_QUE] = { nmmsg_rcv_ringdata1, nmmsg_rcv_ringdata2, ‥, nmmsg_rcv_ringdata「NMNUMRCVQUE の値」 };			

5.1.5.42. 送信 NMPDU バッファ送信元

OIL		設定値	出力ファイル
オブジェクト	属性	－	nm_cfg.c
NM	－		
出力内容			
NodeIdType nmmsg_snd_srcid;			

5.1.5.43. 送信 NMPDU バッファ送信先

OIL		設定値	出力ファイル
オブジェクト	属性	－	nm_cfg.c
NM	－		
出力内容			
NodeIdType nmmsg_snd_destid;			

5.1.5.44. 送信 NMPDU バッファ OpCode

OIL		設定値	出力ファイル
オブジェクト	属性	－	nm_cfg.c
NM	－		
出力内容			
NmOpcodeType nmmsg_snd_opcode;			

5.1.5.45. 送信 NMPDU バッファデータバッファ

OIL	設定値	出力ファイル

オブジェクト	属性	－	nm_cfg.c
NM	－		
出力内容			
static RingDataType nmmsg_snd_ringdata[NM_TNUM_RING_DATA];			

5.1.5.46. 送信 NMPDU バッファデータフィールド

OIL		設定値	出力ファイル
オブジェクト	属性	－	nm_cfg.c
NM	－		
出力内容			
const RingDataType* nmmsg_snd_datafield = nmmsg_snd_ringdata;			

5.1.5.47. NM メッセージ受信時の通知機構

OIL		設定値	出力ファイル
オブジェクト	属性	ACTIVATETASK, SETEVENT, NMCALLBACK, NONE	nm_cfg.c
NM	NMMSGNOTIFICATION		
出力内容			

(i) NMMSGNOTIFICATION = ACTIVATION の場合

```
DeclareTask(TASK の値);
static void activate_nm_msg( void );
static void activate_nm_msg( void )
{
  (void)ActivateTask(TASK の値);
}
const FP notification_nm_msg = activate_nm_msg;
```

(ii) NMMSGNOTIFICATION = EVENT の場合

```
DeclareTask(TASK の値);
DeclareEvent (EVENT の値);
static void setevent_nm_msg( void );
static void setevent_nm_msg( void )
{
  (void)SetEvent(TASK の値, EVENT の値);
}
const FP notification_nm_msg = setevent_nm_msg;
```

(iii) NMMSGNOTIFICATION = NMCALLBACK の場合

```
extern void NMCALLBACKNAME(NMCALLBACK の値 )( void );
const FP notification_nm_msg = NMCALLBACKNAME(NMCALLBACK の値);
```

(iv) NMMSGNOTIFICATION = NONE の場合

```
const FP notification_nm_msg = FP_NULL;
```

5.1.5.48. コンフィグ変化時の通知機構

OIL		設定値	出力ファイル
オブジェクト	属性	ACTIVATETASK, SETEVENT, NMCALLBACK, NONE	nm_cfg.c
NM	NMCMASKNOTIFICATION		
出力内容			

(i) NMCMASKNOTIFICATION = ACTIVATION の場合

```
DeclareTask(TASK の値);
static void activate_nm_configuration( void );
static void activate_nm_configuration( void )
{
  (void)ActivateTask(TASK の値);
}
const FP notification_nm_configuration = activate_nm_configuration;
```

(ii) NMCMASKNOTIFICATION = EVENT の場合

```
DeclareTask(TASK の値);
DeclareEvent (EVENT の値);
static void setevent_nm_configuration( void );
static void setevent_nm_configuration ( void )
{
  (void)SetEvent(TASK の値, EVENT の値);
}
const FP notification_nm_configuration = setevent_nm_configuration;
```

(iii) NMCMASKNOTIFICATION = NMCALLBACK の場合

```
extern void NMCALLBACKNAME(NMCALLBACK の値)( void );
const FP notification_nm_configuration = NMCALLBACKNAME(NMCALLBACK の値);
```

(iv) NMCMASKNOTIFICATION = NONE の場合

```
const FP notification_nm_configuration = FP_NULL;
```

5.1.5.49. NM ステータス変化時の通知機構

OIL		設定値	出力ファイル
オブジェクト	属性	ACTIVATETASK, SETEVENT, NMCALLBACK, NONE	nm_cfg.c
NM	NMSMASKNOTIFICATION		
出力内容			

(i) NMSMASKNOTIFICATION = ACTIVATION の場合

```
DeclareTask(TASK の値);
static void activate_nm_status( void );
static void activate_nm_status( void )
{
  (void)ActivateTask(TASK の値);
}
const FP notification_nm_status = activate_nm_status;
```

(ii) NMSMASKNOTIFICATION = EVENT の場合

```
DeclareTask(TASK の値);
DeclareEvent (EVENT の値);
static void setevent_nm_status( void );
static void setevent_nm_status( void )
{
  (void)SetEvent(TASK の値, EVENT の値);
}
const FP notification_nm_status = setevent_nm_status;
```

(iii) NMSMASKNOTIFICATION = NMCALLBACK の場合

```
extern void NMCALLBACKNAME(NMCALLBACK の値)( void );
const FP notification_nm_status = NMCALLBACKNAME(NMCALLBACK の値);
```

(iv) NMSMASKNOTIFICATION= NONE の場合

```
const FP notification_nm_status = FP_NULL;
```

5.1.6. NM ノード管理情報（NMNODE）

5.1.6.1. ネットワーク管理ノード

OIL		設定値	出力ファイル
オブジェクト	属性	—	nm_cfg.h
NMNODE	NODEID		
出力内容			
#define NM_ID_NMNODE オブジェクト名称　　(NodeIdType)(NODEID の値)			

5.1.6.2. コンフィグマスク

OIL	設定値	出力ファイル

オブジェクト	属性	TRUE／FALSE	nm_cfg.h
NMNODE	NMCMASK		
	ACTUAL_CONFIG		
	LIMP_HOME_CONFIG		

出力内容

(i) NMCMASK、ACTUAL_CONFIG、LIMP_HOME_CONFIG = TRUE の場合
#define　NM_CMASK_NMNODE オブジェクト名称
(ConfigType)(NM_CONF_ACT | NM_CONF_LIMP)

(ii)ACTUAL_CONFIG = FALSE、NMCMASK、LIMP_HOME_CONFIG = TRUE の場合
#define　NM_CMASK_NMNODE オブジェクト名称　(ConfigType)(NM_CONF_ACT)

(iii) NMCMASK、ACTUAL_CONFIG = TRUE、LIMP_HOME_CONFIG = FALSE の場合
#define　NM_CMASK_NMNODE オブジェクト名称　(ConfigType)(NM_CONF_LIMP)

(iv) NMCMASK = TRUE、ACTUAL_CONFIG、LIMP_HOME_CONFIG = FALSE の場合
#define　NM_CMASK_NMNODE オブジェクト名称　(ConfigType)(NM_CONF_NONE)

(v) NMCMASK = FALSE の場合
#define　NM_CMASK_NMNODE オブジェクト名称　(ConfigType)(NM_CONF_NONE)

5.2.　非 OS 対応 CAN 通信ミドルウェア(DirectNM)

5.2.1.　COM 管理情報(COM)

5.2.1.1.　電源投入後から初期化完了までの時間

OIL		設定値	出力ファイル
オブジェクト	属性	0～65535	can_com_def.h
COM	INITINTERVAL		
出力内容			
#define CAN_ECU_INIT_TIME (INITINTERVAL の値)			

5.2.1.2.　受信キューの数

OIL		設定値	出力ファイル
オブジェクト	属性	1～64	can_com_def.h
COM	RCVQUEUENUM		
CANCOMMON	COMUSERCVINT	TRUE／FALSE	
説明			

RCVQUEUENUM の値を can_com_def.h に出力します。 また、受信キュー数 1 の時は、CANCOMMON オブジェクト COMUSERCVINT 属性（＝COM 受信割込み）未使用時のみ設定が可能です。
出力内容
#define CAN_COM_NUM_RCVQUE (CAN_COM_NUM_RCVQUE の値)

5.2.1.3. 送信キューの数

OIL		設定値	出力ファイル
オブジェクト	属性	1～8	can_com_def.h
COM	SNDQUEUENUM		
出力内容			
#define CAN_COM_NUM_SNDQUE (SNDQUEUENUM の値)			

5.2.1.4. 通信途絶判定時間

OIL		設定値	出力ファイル
オブジェクト	属性	1～65535	can_com_def.h
COM	SNDSTOP		
出力内容			
#define CAN_COM_TIME_SNDSTOP (SNDSTOP の値)			

5.2.1.5. COM 送信間の確保時間

OIL		設定値	出力ファイル
オブジェクト	属性	1～65535	can_com_def.h
COM	SNDINTERVAL		
出力内容			
#define CAN_COM_SND_IVL (SNDINTERVAL の値)			

5.2.1.6. イベント-イベント間隔保障時間

OIL		設定値	出力ファイル
オブジェクト	属性	1～65535	can_com_def.h
COM	EVNTINTERVAL		
出力内容			
#define CAN_COM_EVNT_ITV_CNTDATA　(UINT16)(EVNTINTERVAL の値)			

5.2.1.7. COM のメイン周期

OIL	設定値	出力ファイル

オブジェクト	属性	1～255	can_com_def.h
COM	MAIN_CYCLE		
出力内容			
#define CAN_COM_MAIN_CYCLE (MAIN_CYCLE の値)			

5.2.1.8. 送受信 CAN-ID の数

OIL		設定値	出力ファイル
オブジェクト	属性	0～2047	can_com_def.h
COM	CANID		
説明			
送受信 CANID の数を can_com_def.h に出力します。 CANID の定義数は 2～72 の範囲で設定してください。			
出力内容			
#define CAN_COM_NUM_CANID (送受信 CANID の数)			

5.2.1.9. 送信データ ID の数

OIL		設定値	出力ファイル
オブジェクト	属性	SENT	can_com_def.h
COM	IPDUPROPERTY		
	CANID	0～2047	
出力内容			
#define CAN_COM_NUM_SNDDATA (送信データ ID の数)			

5.2.1.10. 受信データ ID の数

OIL		設定値	出力ファイル
オブジェクト	属性	RECEIVED	can_com_def.h
COM	IPDUPROPERTY		
	CANID	0～2047	
出力内容			
#define CAN_COM_NUM_RCVDATA (受信データ ID の数)			

5.2.1.11. COM 受信割り込み使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE／FALSE	can_com_def.h
COM	COMUSERCVINT		
出力内容			

#define CAN_COM_USE_RCVINT (COMUSERCVINT の値)

5.2.1.12. CANID とデータ ID 関連テーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～2047	can_com_table.c
COM	CANID		
	DATAID	0～255	
出力内容			

const CANIDType CANID_TABLE　CANRCVDATA_TABLE [CAN_COM_NUM_RCVDATA] = {
{CANID の値, データ ID の値}
};			

5.2.1.13. 受信データ関連テーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～255	can_com_table.c
COM	CANID		
	TIMEOUT	数値	
	DLCSIZEOVER	GETDATA8BYE, GETDATACANCEL	
	DLCSIZEUNDER	GETDATAONLY, GETDATAAND0, GETDATACANCEL	
	SIZEINBITS	1～64	
	MESSAGEFORMAT	CANID, DATAID	
	CANID	0～2047	
	IPDUPROPERTY	RECEIVED	
MESSAGE	INITIALVALUE	0~255	
NETWORKMESSAGE	BITPOSTION	0～63	
説明			

・デフォルト値について デフォルト値 ={ データ ID の値, ITIALVALUE の値, INITIALVALUE の値, ...} 最初の 8bit はデータ ID が出力されます。 それ以降の bit については、MESSAGEFORMAT に DATA を設定した時のみ、 BITPOSTION で指定された位置に対応するメッセージの INITIALVALUE の値が出力します 何も設定されていない場合は 0 が出力されます。 例：データ ID が 0x02 かつ、BITPOSTION = 8 で INITIALVALUE が 0x01 を設定した場合 デフォルト値 ={0x02, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00 } 先頭にはデータ ID が設定されます。 次に BITPOSTION が 8 なので、INITIALVALUE の値を配列の 3 つめの要素に設定します。 （つまり、BITPOSTION が 0 の場合は配列の 2 つめの要素に設定されます）
出力内容
const CANRcvDataType CANRCVDATA_TABLE[CAN_COM_NUM_RCVDATA] = { {データ ID の値, SIZEINBITS の値 / 8,TIMEOUT の値 , TIMIOUT の値, DLCSIZEOVER の値, DLCSIZEUNDER の値, デフォルト値}

5.2.1.14. 送信データ関連テーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～255	can_com_table.c
COM	DATAID		
	CANID	0～2047	
	TRANSMISSIONMODE	DIRECT, PERIODIC, MIXED, MIXEDEXT	
	USESNDINTERVAL	TRUE／FALSE	
	TIMEPERIOD	数値	
	TIMEOFFSET	数値	
	MESSAGEFORMAT	CANID, DATA	
MESSAGE	SIZEINBITS	1～64	
	INITIALVALUE	0~255	
NETWORKMESSAGE	BITPOSTION	0～63	
説明			

・TRANSMISSIOMMODE の OIL 設定値について
“DIRECT“が設定されていた場合、”CAN_COM_EVENT_MODE”が出力されます。
“PERIODIC”が設定されていた場合、”CAN_COM_EVENT_MODE”が出力されます。
“MIXED”が設定されていた場合、”CAN_COM_EVENT_MODE”が出力されます。
“MIXEDEXT”が設定されていた場合、”CAN_COM_EVENT_MODE”が出力されます。

・デフォルト値について
デフォルト値 ={ データ ID の値, ITIALVALUE の値, INITIALVALUE の値, …}
最初の 8bit はデータ ID が出力されます。
それ以降の bit については、MESSAGEFORMAT に DATA を設定した時のみ、
BITPOSTION で指定された位置に対応するメッセージの INITIALVALUE の値が出力します
何も設定されていない場合は 0 が出力されます。

例：データ ID が 0x02 かつ、BITPOSTION = 8 で INITIALVALUE が 0x01 を設定した場合
デフォルト値 ={0x02, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00 }
先頭にはデータ ID が設定されます。
次に BITPOSTION が 8 なので、INITIALVALUE の値を配列の 3 つめの要素に設定します。
（つまり、BITPOSTION が 0 の場合は配列の 2 つめの要素に設定されます）

出力内容

const CANSndDataType CANSNDDATA_TABLE[CAN_COM_NUM_SNDDATA] = {
{データ ID の値, CANID の値, TRANSMISSIONMODE に対応する値, CAN_USESNDINTERVAL の値, SIZEINBITS の値 / 8, TIMEOFFSET の値, TIMEPERIOD の値, デフォルト値}
};

5.2.1.15. ユーザアクセス用デファイン定義

OIL		設定値	出力ファイル
オブジェクト	属性	－	can_com_api.h
COM	－		
出力内容			
#define message_001 api_snddata[0].ipdu_001_def.message_001			

5.2.2. NM 管理情報(NM)

5.2.2.1. ネットワーク上のノード最大数

OIL		設定値	出力ファイル
オブジェクト	属性	1～255	can_nm_def.h

NM	NODENUM		
CANCOMMON	MAINCYCLE	1～255	
出力内容			
#define CAN_ECU_NUM_NODE (NODENUM の値)			

5.2.2.2. 受信キューの数

OIL		設定値	出力ファイル
オブジェクト	属性	2～64	can_nm_def.h
NM	NMNUMRCVQUE		
出力内容			
#define CAN_ECU_NUM_NODE (NMNUMRCVQUE の値)			

5.2.2.3. 送信キューの数

OIL		設定値	出力ファイル
オブジェクト	属性	2～8	can_nm_def.h
NM	NMNUMSNDQUE		
出力内容			
#define CAN_NM_NUM_SNDQUE (NMNUMSNDQUE の値)			

5.2.2.4. データ長

OIL		設定値	出力ファイル
オブジェクト	属性	2～8	can_nm_def.h
NM	DLC		
出力内容			
#define CAN_NM_NUM_DLC (UINT8)(DLC の値)			

5.2.2.5. 受信不能カウンタ閾値

OIL		設定値	出力ファイル
オブジェクト	属性	1～255	can_nm_def.h
NM	NMRXLIMIT		
出力内容			
#define CAN_NM_RX_LIMIT (UINT8)(NMRXLIMIT の値)			

5.2.2.6. 送信不能カウンタ閾値

OIL		設定値	出力ファイル
オブジェクト	属性	1～255	can_nm_def.h

NM	NMTXLIMIT		
出力内容			
#define CAN_NM_TX_LIMIT (UINT8)(NMTXLIMIT の値)			

5.2.2.7. LimpHome 送信周期

OIL		設定値	出力ファイル
オブジェクト	属性	1～65535	can_nm_def.h
NM	TERROR		
CANCOMMON	MAINCYCLE	1～255	
説明			
TERROR の値を can_nm_def.h に出力します。 また、CANCOMMON オブジェクトの MAINCYCLE 属性値の倍数になるように設定されていない場合はエラーとします。			
出力内容			
#define CAN_NM_TERR_LIMIT (UINT16)(TERROR の値)			

5.2.2.8. トークン保持時間

OIL		設定値	出力ファイル
オブジェクト	属性	70～110	can_nm_def.h
NM	TTYP		
CANCOMMON	MAINCYCLE	1～255	
説明			
TTYP の値を can_nm_def.h に出力します。 また、CANCOMMON オブジェクトの MAINCYCLE 属性値の倍数になるように設定されていない場合はエラーとします。			
出力内容			
#define CAN_NM_TTYP_LIMIT (UINT8)(TTYP の値)			

5.2.2.9. メッセージが存在しない場合の認識時間

OIL		設定値	出力ファイル
オブジェクト	属性	220～284	can_nm_def.h
NM	TMAX		
CANCOMMON	MAINCYCLE	1～255	
説明			
TMAX の値を can_nm_def.h に出力します。 また、CANCOMMON オブジェクトの MAINCYCLE 属性値の倍数になるように設定されていない場合はエラーとします。			

出力内容
#define CAN_NM_TMAX_LIMIT (UINT16)(TMAX の値)

5.2.2.10. BusSleep 前待機時間

OIL		設定値	出力ファイル
オブジェクト	属性	1～65535	can_nm_def.h
NM	TWAITBUSSLEEP		
CANCOMMON	MAINCYCLE	1～255	
説明			
TWAITBUSSLEEP の値を can_nm_def.h に出力します。 また、CANCOMMON オブジェクトの MAINCYCLE 属性値の倍数になるように設定されていない場合はエラーとします。			
出力内容			
#define CAN_NM_TWBS_LIMIT (UINT16)(TWAITBUSSLEEP の値)			

5.2.2.11. NM のメイン周期

OIL		設定値	出力ファイル
オブジェクト	属性	1～255	can_nm_def.h
NM	MAINCYCLE		
出力内容			
#define CAN_NM_MAIN_CYCLE (MAINCYCLE の値)			

5.2.2.12. NM で使用する CAN-ID

OIL		設定値	出力ファイル
オブジェクト	属性	0～255	can_nm_def.h
NM	IDBASE		
	WINDOWMASK	0～2040	
出力内容			
#define CAN_NM_MY_CAN_ID (UINT16)(NM で使用する CAN_ID の値)			

5.2.2.13. IDBASE のマスク値

OIL		設定値	出力ファイル
オブジェクト	属性	0～2040	can_nm_def.h
NM	WINDOWMASK		
出力内容			
#define CAN_NM_WINDOW_MASK (UINT16)(WINDOWMASK の値)			

5.2.3. CANDRV 管理情報(CANDRV)

5.2.3.1. ボーレート設定

OIL		設定値	出力ファイル
オブジェクト	属性	250, 500	can_drv_def.h
CANDRV	BAUDRATE		
出力内容			

・BAUDRATE = 250 の場合

```
#define CAN_BAUDRATE CAN_DRV_250KBPS
```

・BAUDRATE = 500 の場合

```
#define CAN_BAUDRATE CAN_DRV_500KBPS
```

5.2.3.2. チャンネル設定

OIL		設定値	出力ファイル
オブジェクト	属性	0, 1	can_drv_def.h
CANDRV	CHANNEL		
出力内容			

・CHANNEL = 0 の場合

```
#define CAN_CH CAN_DRV_CH0
```

・CHANNEL = 1 の場合

```
#define CAN_CH CAN_DRV_CH1
```

5.2.3.3. CPU クロック周波数設定

OIL		設定値	出力ファイル
オブジェクト	属性	20, 30, 32	can_drv_def.h
CANDRV	CLOCK		
出力内容			

・CLOCK = 20 の場合

#define CAN_CLK CAN_DRV_CLK_20MHZ

・CLOCK = 30 の場合

#define CAN_CLK CAN_DRV_CLK_30MHZ

・CLOCK = 32 の場合

#define CAN_CLK CAN_DRV_CLK_32MHZ

5.2.3.4. ビットタイミング設定

OIL		設定値	出力ファイル
オブジェクト	属性	16, 20	can_drv_def.h
CANDRV	NUMTQ		

出力内容

・NUMTQ = 16 の場合

#define CAN_NUM_TQ CAN_DRV_16TQ

・NUMTQ = 20 の場合

#define CAN_NUM_TQ CAN_DRV_20TQ

5.2.3.5. 使用入力ポート設定

OIL		設定値	出力ファイル
オブジェクト	属性	CAN_DRV_USE_P77, CAN_DRV_USE_P83, CAN_DRV_USE_P95	can_drv_def.h
CANDRV	PORTIN		

出力内容

・PORTIN = 77 の場合

#define CAN_PORT_IN CAN_DRV_USE_P77

・PORTIN = 83 の場合

#define CAN_PORT_IN CAN_DRV_USE_P83

・PORTIN = 95 の場合

#define CAN_PORT_IN CAN_DRV_USE_P95

5.2.3.6. 使用出力ポート設定

OIL		設定値	出力ファイル
オブジェクト	属性	CAN_DRV_USE_P76, CAN_DRV_USE_P82, CAN_DRV_USE_P96	can_drv_def.h
CANDRV	PORTOUT		
出力内容			
・PORTIN = 76 の場合 #define CAN_PORT_OUT CAN_DRV_USE_P76 ・PORTIN = 82 の場合 #define CAN_PORT_OUT CAN_DRV_USE_P82 ・PORTIN = 96 の場合 #define CAN_PORT_OUT CAN_DRV_USE_P96			

5.2.3.7. CAN の Sleep 許可/禁止

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE	can_drv_def.h
CANDRV	USESLEEPMODE		
説明			
DirectNM かつ COM,NM 使用時の設定値は TRUE 固定とし、FALSE が指定された場合はエラーとします。			
出力内容			
・USESLEEPMODE = TRUEの場合 #define CAN_DRV_SLEEPMODE CAN_TRUE			

5.2.3.8. CAN 受信割り込み 使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE	can_drv_def.h
CANDRV	CANRCVINT		
説明			
DirectNM かつ COM,NM 使用時の設定値は TRUE 固定とし、FALSE が指定された場合はエラーとします。			
出力内容			
・CANRCVINT = TRUEの場合 #define CAN_DRV_RCVINT CAN_TRUE			

5.2.3.9. NM 送信設定 使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE	can_drv_def.h
CANDRV	USENMSND		
説明			
DirectNM かつ COM,NM 使用時の設定値は TRUE 固定とし、FALSE が指定された場合はエラーとします。			
出力内容			
・USENMSND = TRUEの場合 #define CAN_DRV_NM_SND CAN_TRUE			

5.2.3.10. COM 送信設定 使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE	can_drv_def.h
CANDRV	USECOMSND		
説明			
TRUE 固定とし、FALSE が指定された場合はエラーとします。			
出力内容			
・USECOMSND = TRUEの場合 #define CAN_DRV_COM_SND CAN_TRUE			

5.2.3.11. NM 受信設定 使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE	can_drv_def.h
CANDRV	USENMRCV		
説明			
DirectNM かつ COM,NM 使用時の設定値は TRUE 固定とし、FALSE が指定された場合はエラーとします。			
出力内容			
・USENMRCV = TRUEの場合 #define CAN_DRV_NM_RCV CAN_TRUE ・USENMRCV = FALSEの場合 #define CAN_DRV_NM_RCV CAN_FALSE			

5.2.3.12. COM 受信設定 使用/未仕様

OIL	設定値	出力ファイル

オブジェクト	属性	TRUE／FALSE	can_drv_def.h
CANDRV	USECOMRCV		
CANCOMMON	COMUSERCVINT	TRUE／FALSE	
説明			
COM 受信割り込み(CANCOMMON オブジェクトの COMUSERCVINT 属性)が TRUE の場合は TRUE、FALSE の場合は FALSE を指定する。			
出力内容			
・USECOMRCV = TRUEの場合 #define CAN_DRV_COM_RCV CAN_TRUE ・USECOMRCV = FALSEの場合 #define CAN_DRV_COM_RCV CAN_FALSE			

5.2.3.13. CAN ポーリング受信 使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE／FALSE	can_drv_def.h
CANDRV	USEPOLRCV		
説明			
COM 受信割り込み(CANCOMMON オブジェクトの COMUSERCVINT 属性)が TRUE の場合は TRUE、FALSE の場合は FALSE を指定する。			
出力内容			
・USEPOLRCV = TRUEの場合 #define CAN_DRV_POL_RCV CAN_TRUE ・USEPOLRCV = FALSEの場合 #define CAN_DRV_POL_RCV CAN_FALSE			

5.2.3.14. エラー通知設定 使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE／FALSE	can_drv_def.h
CANDRV	USEERRCALL		
出力内容			
・USEERRCALL = TRUEの場合 #define CAN_DRV_ERR_CALL CAN_FALSE ・USEERRCALL = FALSEの場合 #define CAN_DRV_ERR_CALL CAN_FALSE			

5.2.3.15. ウェイクアップ割り込みレベル設定

OIL		設定値	出力ファイル
オブジェクト	属性	1～7	can_drv_def.h
CANDRV	WKUPINTLEVEL		
出力内容			
#define CAN_WKUPINT_LEVEL WKUPINTLEVEL の値			

5.2.3.16. 受信完了割り込みレベル設定

OIL		設定値	出力ファイル
オブジェクト	属性	1～7	can_drv_def.h
CANDRV	RCVINTLEVEL		
出力内容			
#define CAN_RCVINT_LEVEL RCVINTLEVEL の値			

5.2.3.17. 送信完了割り込みレベル設定

OIL		設定値	出力ファイル
オブジェクト	属性	1～7	can_drv_def.h
CANDRV	SNDINTLEVEL		
出力内容			
#define CAN_SNDINT_LEVEL SNDINTLEVEL の値			

5.2.3.18. エラー割り込みレベル設定

OIL		設定値	出力ファイル
オブジェクト	属性	1～7	can_drv_def.h
CANDRV	ERRINTLEVEL		
出力内容			
#define CAN_ERRINT_LEVEL ERRINTLEVEL の値			

5.2.3.19. 送受信用の CAN スロット ID テーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～2047	can_drv_def.h
CANDRV	CANID		
	USEPOLRCV	TRUE／FALSE	
説明			

送受信用の CAN スロット ID テーブルを can_drv_def.h に出力します。 CAN ポーリングが FALSE の時は、スロット 0～10 を全て CAN_DRV_NO_USE として出力します。
出力内容

const far UINT16 CAN_SLOTID_TABLE[CAN_DRV_MAX_SLOTNUM] = {
　　CANID の値,
　　…
};

5.2.3.20. CAN スロットマスクテーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～2040	can_drv_def.h
CANDRV	WINDOWMASK		
CANCOMMON	COMUSERCVINT	TRUE／FALSE	
説明			
CAN スロットマスクテーブルを can_drv_def.h に出力します。 COM 受信割り込み使用ありの場合は、ローカルマスクレジスタ A,B を 0x000 に設定し、それ以外の場合は、ローカルマスクレジスタ A,B を WINDOW_MASK に設定します。 グローバルマスクレジスタは 0x07FF に設定する。			
出力内容			

const far UINT16 CAN_SLOTMASK_TABLE [CAN_DRV_MASKREG_NUM]= {
　　グローバルマスクレジスタ
　　,ローカルマスクレジスタ A
　　,ローカルマスクレジスタ B
};			

5.2.3.21. NM 用 ASU テーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～255	can_drv_def.h
CANDRV	IDBASE		
	WINDOWMASK	0～2040	
説明			

NM 用 ASU テーブルを can_drv_def.h に出力します。
DirectNM で使用する（受信する）CAN-ID のビットを 1 に設定し、また、受信しない CAN-ID のビットは 0 に設定すること。
各配列の要素ごとに 8 個の CAN-ID を管理している。
■配列の 1 つめの要素：CAN-ID 000h～007h までを管理

ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

←上位ビット　　下位ビット→

000h の CAN-ID を使用する場合は、対応するビット(この場合は最下位ビット)を 1 とし
1 つ目の要素の出力結果は”0x01”となる。

出力内容

・DirectNMで使用するCAN-IDが 000h, 00Fhだった場合

```
const far UINT8 CAN_ASU_TABLE_NM[ CAN_DRV_ASU_MAX ]= {
    0x01, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,      /* CAN-ID: 000h - 03Fh */
    …
};
```

5.2.3.22. COM 用 ASU テーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～2047	can_drv_def.h
CANDRV	CANID		
	USEPOLRCV	TRUE/FALSE	

説明

CAN ポーリングが FALSE の場合は、OIL にて指定された全ての ID を本テーブルに出力します。
TRUE の場合は送受信用の CAN スロットテーブル(CAN_SLOTID_TABLE)に配置できなかった ID のみ本テーブルに出力する。
また、本テーブルは OIL の設定(COM 受信割り込み)によって出力の制限は行われない。
各配列の要素ごとに 8 個の CAN-ID を管理している。
■配列の 1 つめの要素：CAN-ID 000h～007h までを管理

ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

←上位ビット　　下位ビット→

000h の CAN-ID を使用する場合は、対応するビット(この場合は最下位ビット)を 1 とし
1 つ目の要素の出力結果は”0x01”となる。

出力内容

・定義されたCAN-IDが 000h, 00Fhだった場合

```
const far UINT8 CAN_ASU_TABLE_COM[ CAN_DRV_ASU_MAX ]= {
    0x01, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,      /* CAN-ID: 000h - 03Fh */
    …
};
```

5.3. 非 OS 対応 CAN 通信ミドルウェア(InDirectNM)

5.3.1. COM 管理情報(COM)

5.3.1.1. 電源投入後から初期化完了までの時間

OIL		設定値	出力ファイル
オブジェクト	属性	0～65535	can_com_def.h
COM	INITINTERVAL		
出力内容			
#define CAN_ECU_INIT_TIME (INITINTERVAL の値)			

5.3.1.2. 受信キューの数

OIL		設定値	出力ファイル
オブジェクト	属性	1～64	can_com_def.h
COM	RCVQUEUENUM		
CANCOMMON	COMUSERCVINT	TRUE／FALSE	
説明			
RCVQUEUENUM の値を can_com_def.h に出力します。 また、受信キュー数 1 の時は、CANCOMMON オブジェクト COMUSERCVINT 属性（＝COM 受信割込み）未使用時のみ設定が可能です。			
出力内容			
#define CAN_COM_NUM_RCVQUE (CAN_COM_NUM_RCVQUE の値)			

5.3.1.3. 送信キューの数

OIL		設定値	出力ファイル
オブジェクト	属性	1～8	can_com_def.h
COM	SNDQUEUENUM		
出力内容			
#define CAN_COM_NUM_SNDQUE (SNDQUEUENUM の値)			

5.3.1.4. 通信途絶判定時間

OIL	設定値	出力ファイル

オブジェクト	属性	1～65535	can_com_def.h
COM	SNDSTOP		
出力内容			
#define CAN_COM_TIME_SNDSTOP (SNDSTOP の値)			

5.3.1.5. COM 送信間の確保時間

OIL		設定値	出力ファイル
オブジェクト	属性	1～65535	can_com_def.h
COM	SNDINTERVAL		
出力内容			
#define CAN_COM_SND_IVL (SNDINTERVAL の値)			

5.3.1.6. イベント-イベント間隔保障時間

OIL		設定値	出力ファイル
オブジェクト	属性	1～65535	can_com_def.h
COM	EVNTINTERVAL		
出力内容			
#define CAN_COM_EVNT_ITV_CNTDATA　(UINT16)(EVNTINTERVAL の値)			

5.3.1.7. COM のメイン周期

OIL		設定値	出力ファイル
オブジェクト	属性	1～255	can_com_def.h
COM	MAIN_CYCLE		
出力内容			
#define CAN_COM_MAIN_CYCLE (MAIN_CYCLE の値)			

5.3.1.8. 送受信 CAN-ID の数

OIL		設定値	出力ファイル
オブジェクト	属性	0～2047	can_com_def.h
COM	CANID		
説明			
送受信 CANID の数を can_com_def.h に出力します。 CANID の定義数は 2～72 の範囲で設定してください。			
出力内容			
#define CAN_COM_NUM_CANID (送受信 CANID の数)			

5.3.1.9. 送信データ ID の数

OIL		設定値	出力ファイル
オブジェクト	属性	SENT	can_com_def.h
COM	IPDUPROPERTY		
	CANID	0～2047	
出力内容			
#define CAN_COM_NUM_SNDDATA (送信データ ID の数)			

5.3.1.10. 受信データ ID の数

OIL		設定値	出力ファイル
オブジェクト	属性	RECEIVED	can_com_def.h
COM	IPDUPROPERTY		
	CANID	0～2047	
出力内容			
#define CAN_COM_NUM_RCVDATA (受信データ ID の数)			

5.3.1.11. COM 受信割り込み使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE／FALSE	can_com_def.h
COM	COMUSERCVINT		
出力内容			
#define CAN_COM_USE_RCVINT (COMUSERCVINT の値)			

5.3.1.12. ノード定義

OIL		設定値
オブジェクト	属性	0～255
COM	－	
説明		
ユーザ定義ノード名を can_com_def.h に出力します。 OIL で設定したノード名が監視ノード分出力されます。		
出力内容		
#define ユーザ定義ノード名 (連続した値 (0 から昇順))		

5.3.1.13. COM 自ノード CAN-ID

OIL		設定値	出力ファイル
オブジェクト	属性	0～2047	can_com_def.h

COM	OWNCANID		
出力内容			
#define CAN_COM_OWN_NODE_CANID ((UINT16)(OWNCANID の値))			

5.3.1.14. COM 自ノードコールバックインデックス

OIL		設定値
オブジェクト	属性	ノード定義のうち１つ
COM	－	
説明		
COM 自ノードコールバックインデックスを can_com_def.h に出力します。 OIL で設定したユーザ定義ノード名を設定します。		
出力内容		
#define CAN_COM_OWN_NODE_IDX (ユーザ定義ノード名)		

5.3.1.15. InDirectNM 定義

OIL		設定値
オブジェクト	属性	値なし
COM	－	
説明		
InDirectNM 定義を can_com_def.h に出力します。 InDirectNM 時のみ宣言し、宣言無しで DirectNM と同等の COM になります。		
出力内容		
#define CAN_COM_NM_INDIRECT		

5.3.1.16. CANID とデータ ID 関連テーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～2047	can_com_table.c
COM	CANID		
	DATAID	0～255	
出力内容			

const CANIDType CANID_TABLE　CANRCVDATA_TABLE [CAN_COM_NUM_RCVDATA] = {
　{CANID の値, データ ID の値}
};			

5.3.1.17. 受信データ関連テーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～255	can_com_table.c
COM	CANID		
	TIMEOUT	数値	
	DLCSIZEOVER	GETDATA8BYE, GETDATACANCEL	
	DLCSIZEUNDER	GETDATAONLY, GETDATAAND0, GETDATACANCEL	
	SIZEINBITS	1～64	
	MESSAGEFORMAT	CANID, DATAID	
	CANID	0～2047	
	IPDUPROPERTY	RECEIVED	
MESSAGE	INITIALVALUE	0~255	
NETWORKMESSAGE	BITPOSTION	0～63	

説明

・デフォルト値について
デフォルト値 ={ データ ID の値, ITIALVALUE の値, INITIALVALUE の値, …}
最初の 8bit はデータ ID が出力されます。
それ以降の bit については、MESSAGEFORMAT に DATA を設定した時のみ、
BITPOSTION で指定された位置に対応するメッセージの INITIALVALUE の値が出力します
何も設定されていない場合は 0 が出力されます。

例：データ ID が 0x02 かつ、BITPOSTION = 8 で INITIALVALUE が 0x01 を設定した場合
デフォルト値 ={0x02, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00 }
先頭にはデータ ID が設定されます。
次に BITPOSTION が 8 なので、INITIALVALUE の値を配列の 3 つめの要素に設定します。
（つまり、BITPOSTION が 0 の場合は配列の 2 つめの要素に設定されます）

出力内容

const CANRcvDataType CANRCVDATA_TABLE[CAN_COM_NUM_RCVDATA] = {
　　{データ ID の値, SIZEINBITS の値 / 8,TIMEOUT の値 , TIMIOUT の値, DLCSIZEOVER の値, DLCSIZEUNDER の値, デフォルト値}

5.3.1.18. 送信データ関連テーブル

OIL	設定値	出力ファイル

オブジェクト	属性		
COM	DATAID	0～255	can_com_table.c
	CANID	0～2047	
	TRANSMISSIONMODE	DIRECT, PERIODIC, MIXED, MIXEDEXT	
	USESNDINTERVAL	TRUE／FALSE	
	TIMEPERIOD	数値	
	TIMEOFFSET	数値	
	MESSAGEFORMAT	CANID, DATA	
MESSAGE	SIZEINBITS	1～64	
	INITIALVALUE	0~255	
NETWORKMESSAGE	BITPOSTION	0～63	

説明

・TRANSMISSIOMMODE の OIL 設定値について

“DIRECT“が設定されていた場合、”CAN_COM_EVENT_MODE”が出力されます。

“PERIODIC”が設定されていた場合、”CAN_COM_EVENT_MODE”が出力されます。

“MIXED”が設定されていた場合、”CAN_COM_EVENT_MODE”が出力されます。

“MIXEDEXT”が設定されていた場合、”CAN_COM_EVENT_MODE”が出力されます。

・デフォルト値について

デフォルト値 ={ データ ID の値, ITIALVALUE の値, INITIALVALUE の値, …}

最初の 8bit はデータ ID が出力されます。

それ以降の bit については、MESSAGEFORMAT に DATA を設定した時のみ、

BITPOSTION で指定された位置に対応するメッセージの INITIALVALUE の値が出力します

何も設定されていない場合は 0 が出力されます。

例：データ ID が 0x02 かつ、BITPOSTION = 8 で INITIALVALUE が 0x01 を設定した場合

デフォルト値 ={0x02, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00 }

先頭にはデータ ID が設定されます。

次に BITPOSTION が 8 なので、INITIALVALUE の値を配列の 3 つめの要素に設定します。

（つまり、BITPOSTION が 0 の場合は配列の 2 つめの要素に設定されます）

出力内容

```
const CANSndDataType CANSNDDATA_TABLE[CAN_COM_NUM_SNDDATA] = {
    {データ ID の値, CANID の値, TRANSMISSIONMODE に対応する値,
CAN_USESNDINTERVAL の値, SIZEINBITS の値 / 8, TIMEOFFSET の値, TIMEPERIOD
の値, デフォルト値}
};
```

5.3.1.19. ノード監視インデックステーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～2047	can_com_table.c
COM	MAINUSECANID		
IPDU	IPDUPROPERTY	RECEIVED	

説明
送信データ関連テーブルで設定した受信 ID を監視対象とする場合は設定したノード名を、監視対象としない場合は CAN_COM_NO_USE を設定する。 設定した ID が RECEIVED 設定の IPDU オブジェクトの CANID に割当てられていない場合はエラーとする。 IPDU オブジェクトの受信設定された数分のテーブルの要素数ができる。 IPDU オブジェクトで設定した CANID と COM オブジェクトで設定した MAINUSECANID が一致する場合に設定したノード名を出力する。一致するものがなかった場合は CAN_COM_NO_USE が出力される。

出力内容

```
#ifdef CAN_COM_NM_INDIRECT
const UINT16 CAN_RCV_IDX_TBL[CAN_COM_NUM_RCVDATA] = {
    ノード名 / CAN_COM_NO_USE
};
#endif
```

5.3.1.20. ユーザアクセス用デファイン定義

OIL		設定値	出力ファイル
オブジェクト	属性	－	can_com_api.c
COM	－		

出力内容
#define message_001 api snddata[0],ipdu_001 def.message_001

5.3.2. NM 管理情報(NM)

5.3.2.1. LimpHome 送信周期

OIL		設定値	出力ファイル
オブジェクト	属性	1～65535	can_nm_def.h
NM	TERROR		
CANCOMMON	MAINCYCLE	1～255	
説明			
TERROR の値を can_nm_def.h に出力します。 また、CANCOMMON オブジェクトの MAINCYCLE 属性値の倍数になるように設定されていない場合はエラーとします。			
出力内容			
#define CAN_NM_TERR_LIMIT (UINT16)(TERROR の値)			

5.3.2.2. BusSleep 前待機時間

OIL		設定値	出力ファイル
オブジェクト	属性	1～65535	can_nm_def.h
NM	TWAITBUSSLEEP		
CANCOMMON	MAINCYCLE	1～255	
説明			
TWAITBUSSLEEP の値を can_nm_def.h に出力します。 また、CANCOMMON オブジェクトの MAINCYCLE 属性値の倍数になるように設定されていない場合はエラーとします。			
出力内容			
#define CAN_NM_TWBS_LIMIT (UINT16)(TWAITBUSSLEEP の値)			

5.3.2.3. TOB 選択時タイムアウト時間

OIL		設定値	出力ファイル
オブジェクト	属性	1～65535	can_nm_def.h
NM	TOB		
	USEMONITOR	TOB	
説明			
TOB の値を can_nm_def.h に出力します。 モニタにて TOB 選択時(USEMONITEOR = TOB)のみ設定可能です。			
出力内容			
#define CAN_NM_TOB_LIMIT (UINT16)(TOB の値)			

5.3.2.4. NM のメイン周期

OIL		設定値	出力ファイル
オブジェクト	属性	1～255	can_nm_def.h
NM	MAINCYCLE		
出力内容			
#define CAN_NM_MAIN_CYCLE (MAIN_CYCLE の値)			

5.3.2.5. NM Sleep 使用/未使用

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE／FALSE	can_nm_def.h
NM	NMUSESLEEP		
出力内容			
・NMUSESLEEP = TRUEの場合 #define CAN_NM_USE_SLEEP (CAN_TRUE) ・NMUSESLEEP = FALSEの場合 #define CAN_NM_USE_SLEEP (CAN_FALSE)			

5.3.2.6. ノード監視選択

OIL		設定値	出力ファイル
オブジェクト	属性	TOB, OMT	can_nm_def.h
NM	USEMONITOR		
出力内容			
・USEMONITOR = TOBの場合 #define CAN_NM_USE_MONITOR (CAN_NM_TOB) ・USEMONITOR = OMTの場合 #define CAN_NM_USE_MONITOR (CAN_NM_OMT)			

5.3.2.7. 自ノード ID インデックス

OIL		設定値	出力ファイル
オブジェクト	属性	0～255	can_nm_def.h
NM	MYNODEID		
出力内容			
#define CAN_NM_MY_NODE_ID (MYNODEID の値)			

5.3.2.8. 拡張ネットワークステータス用カウンタ閾値

OIL		設定値	出力ファイル
オブジェクト	属性	1～255	can_nm_def.h
NM	NETTHRESHOLD		
出力内容			
#define CAN_NM_NET_THRESHOLD (NETTHRESHOLD の値)			

5.3.2.9. 拡張コンフィグ用カウンタ閾値

OIL		設定値	出力ファイル
オブジェクト	属性	1～255	can_nm_def.h
NM	CFGTHRESHOLD		
出力内容			
#define CAN_NM_CFG_THRESHOLD (CFGTHRESHOLD の値)			

5.3.2.10. 拡張ネットワークステータス用カウンタ⊿INC

OIL		設定値	出力ファイル
オブジェクト	属性	1～255	can_nm_def.h
NM	NETDELTAINC		
出力内容			
#define CAN_NM_NETDELTA_INC (NETDELTAINC の値)			

5.3.2.11. 拡張ネットワークステータス用カウンタ⊿DEC

OIL		設定値	出力ファイル
オブジェクト	属性	1～255	can_nm_def.h
NM	NETDELTADEC		
出力内容			
#define CAN_NM_NETDELTA_DEC (NETDELTADEC の値)			

5.3.2.12. 監視ノード最大数

OIL		設定値	出力ファイル
オブジェクト	属性	ユーザ指定数	can_nm_def.h
NM	－		
説明			
ユーザ定義（CANNODE オブジェクト定義数）分出力されます。			
出力内容			

#define CAN_NM_MON_NODE_MAX (CANNODE オブジェクト定義数)

5.3.2.13. INC,DEC,NodeID テーブル

OIL		設定値	出力ファイル
オブジェクト	属性	1～255	can_nm_def.h
NM	DELTAINC		
	DELTADEC	1～255	
	NODEID	1～255	
説明			
ユーザ定義（CANNODE オブジェクト定義数）が 255 以上の場合はエラーとする。			
出力内容			

const NodeCfgStsType CAN_NM_NODE_CFG_TBL[CAN_NM_MON_NODE_MAX] = {
{INC, DEC, ノードＩＤ}
}			

5.3.3.　CANDRV 管理情報(CANDRV)

5.3.3.1.　ボーレート設定

OIL		設定値	出力ファイル
オブジェクト	属性	250, 500	can_drv_def.h
CANDRV	BAUDRATE		
出力内容			
・BAUDRATE = 250 の場合 #define CAN_BAUDRATE CAN_DRV_250KBPS ・BAUDRATE = 500 の場合 #define CAN_BAUDRATE CAN_DRV_500KBPS			

5.3.3.2.　チャンネル設定

OIL		設定値	出力ファイル
オブジェクト	属性	0, 1	can_drv_def.h
CANDRV	CHANNEL		
出力内容			

・CHANNEL = 0 の場合

#define CAN_CH CAN_DRV_CH0

・CHANNEL = 1 の場合

#define CAN_CH CAN_DRV_CH1

5.3.3.3. CPU クロック周波数設定

OIL		設定値	出力ファイル
オブジェクト	属性	20, 30, 32	can_drv_def.h
CANDRV	CLOCK		
出力内容			

・CLOCK = 20 の場合

#define CAN_CLK CAN_DRV_CLK_20MHZ

・CLOCK = 30 の場合

#define CAN_CLK CAN_DRV_CLK_30MHZ

・CLOCK = 32 の場合

#define CAN_CLK CAN_DRV_CLK_32MHZ

5.3.3.4. ビットタイミング設定

OIL		設定値	出力ファイル
オブジェクト	属性	16, 20	can_drv_def.h
CANDRV	NUMTQ		
出力内容			

・NUMTQ = 16 の場合

#define CAN_NUM_TQ CAN_DRV_16TQ

・NUMTQ = 20 の場合

#define CAN_NUM_TQ CAN_DRV_20TQ

5.3.3.5. 使用入力ポート設定

OIL		設定値	出力ファイル
オブジェクト	属性	CAN_DRV_USE_P77, CAN_DRV_USE_P83, CAN_DRV_USE_P95	can_drv_def.h
CANDRV	PORTIN		

出力内容
・PORTIN = 77 の場合 #define CAN_PORT_IN CAN_DRV_USE_P77 ・PORTIN = 83 の場合 **#define CAN_PORT_IN CAN_DRV_USE_P83** ・PORTIN = 95 の場合 #define CAN_PORT_IN CAN_DRV_USE_P95

5.3.3.6. 使用出力ポート設定

OIL		設定値	出力ファイル
オブジェクト	属性	CAN_DRV_USE_P76, CAN_DRV_USE_P82, CAN_DRV_USE_P96	can_drv_def.h
CANDRV	PORTOUT		
出力内容			
・PORTIN = 76 の場合 #define CAN_PORT_OUT CAN_DRV_USE_P76 ・PORTIN = 82 の場合 #define CAN_PORT_OUT CAN_DRV_USE_P82 ・PORTIN = 96 の場合 #define CAN_PORT_OUT CAN_DRV_USE_P96			

5.3.3.7. CAN の Sleep 許可/禁止

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE／FALSE	can_drv_def.h
CANDRV	USESLEEPMODE		
NM	NMUSERSLEEP	TRUE／FALSE	
説明			
NM オブジェクトの NMUSERSLEEP 属性の値と同期させて設定する。			
出力内容			

・USESLEEPMODE = TRUEの場合 #define CAN_DRV_SLEEPMODE CAN_TRUE ・USESLEEPMODE = FALSEの場合 #define CAN_DRV_SLEEPMODE CAN_FALSE

5.3.3.8. CAN 受信割り込み 使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE／FALSE	can_drv_def.h
CANDRV	CANRCVINT		
CANCOMMON	COMUSERCVINT	TRUE／FALSE	
説明			
NM オブジェクトの COMUSERCVINT 属性の値と同期させて設定する。			
出力内容			
・CANRCVINT = TRUEの場合 #define CAN_DRV_RCVINT CAN_TRUE ・CANRCVINT = FALSEの場合 #define CAN_DRV_RCVINT CAN_FALSE			

5.3.3.9. NM 送信設定 使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	FALSE	can_drv_def.h
CANDRV	USENMSND		
説明			
FALSE 固定とし、TRUE が指定された場合はエラーとします。			
出力内容			
・USENMSND = FALSEの場合 #define CAN_DRV_NM_SND CAN_FALSE			

5.3.3.10. COM 送信設定 使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE	can_drv_def.h
CANDRV	USECOMSND		
説明			
TRUE 固定とし、FALSE が指定された場合はエラーとします。			
出力内容			

・USECOMSND = TRUEの場合 #define CAN_DRV_COM_SND CAN_TRUE

5.3.3.11. NM 受信設定 使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	FALSE	can_drv_def.h
CANDRV	USENMRCV		
説明			
FALSE 固定とし、TRUE が指定された場合はエラーとします。			
出力内容			
・USENMRCV = FALSEの場合 #define CAN_DRV_NM_RCV CAN_FALSE			

5.3.3.12. COM 受信設定 使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE／FALSE	can_drv_def.h
CANDRV	USECOMRCV		
CANCOMMON	COMUSERCVINT	TRUE／FALSE	
説明			
COM 受信割り込み(CANCOMMON オブジェクトの COMUSERCVINT 属性)が TRUE の場合は TRUE、FALSE の場合は FALSE を指定する。			
出力内容			
・USECOMRCV = TRUEの場合 #define CAN_DRV_COM_RCV CAN_TRUE ・USECOMRCV = FALSEの場合 #define CAN_DRV_COM_RCV CAN_FALSE			

5.3.3.13. CAN ポーリング受信 使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE／FALSE	can_drv_def.h
CANDRV	USEPOLRCV		
説明			
COM 受信割り込み(CANCOMMON オブジェクトの COMUSERCVINT 属性)が TRUE の場合は TRUE、FALSE の場合は FALSE を指定する。			
出力内容			

・USEPOLRCV = TRUEの場合 #define CAN_DRV_POL_RCV CAN_TRUE ・USEPOLRCV = FALSEの場合 #define CAN_DRV_POL_RCV CAN_FALSE

5.3.3.14. エラー通知設定 使用/未仕様

OIL		設定値	出力ファイル
オブジェクト	属性	TRUE／FALSE	can_drv_def.h
CANDRV	USEERRCALL		
出力内容			
・USEERRCALL = TRUEの場合 #define CAN_DRV_ERR_CALL CAN_FALSE ・USEERRCALL = FALSEの場合 #define CAN_DRV_ERR_CALL CAN_FALSE			

5.3.3.15. ウェイクアップ割り込みレベル設定

OIL		設定値	出力ファイル
オブジェクト	属性	1～7	can_drv_def.h
CANDRV	WKUPINTLEVEL		
出力内容			
#define CAN_WKUPINT_LEVEL WKUPINTLEVEL の値			

5.3.3.16. 受信完了割り込みレベル設定

OIL		設定値	出力ファイル
オブジェクト	属性	1～7	can_drv_def.h
CANDRV	RCVINTLEVEL		
出力内容			
#define CAN_RCVINT_LEVEL RCVINTLEVEL の値			

5.3.3.17. 送信完了割り込みレベル設定

OIL		設定値	出力ファイル
オブジェクト	属性	1～7	can_drv_def.h
CANDRV	SNDINTLEVEL		
出力内容			

#define CAN_SNDINT_LEVEL SNDINTLEVEL の値

5.3.3.18. エラー割り込みレベル設定

OIL		設定値	出力ファイル
オブジェクト	属性	1～7	can_drv_def.h
CANDRV	ERRINTLEVEL		
出力内容			
#define CAN_ERRINT_LEVEL ERRINTLEVEL の値			

5.3.3.19. 送受信用の CAN スロット ID テーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～2047	can_drv_def.h
CANDRV	CANID		
	USEPOLRCV	TRUE/FALSE	
説明			
送受信用の CAN スロット ID テーブルを can_drv_def.h に出力します。 COM ポーリングが FALSE の時は、スロット 0～10 を全て CAN_DRV_NO_USE とする。			
出力内容			

const far UINT16 CAN_SLOTID_TABLE[CAN_DRV_MAX_SLOTNUM] = {
　CANID,
　…
};			

5.3.3.20. CAN スロットマスクテーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～2040	can_drv_def.h
CANDRV	WINDOWMASK		
CANCOMMON	COMUSERCVINT	TRUE／FALSE	
説明			
CAN スロットマスクテーブルを can_drv_def.h に出力します。 ローカルマスクレジスタ A,B を 0x000 に設定し、グローバルマスクレジスタは 0x7FF に設定する。			
出力内容			

```
const far UINT16 CAN_SLOTMASK_TABLE [ CAN_DRV_MASKREG_NUM ]= {
    グローバルマスクレジスタ
    ,ローカルマスクレジスタ A
    ,ローカルマスクレジスタ B
};
```

5.3.3.21. COM 用 ASU テーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～2047	can_drv_def.h
CANDRV	CANID		
	USEPOLRCV	TRUE/FALSE	
説明			

CAN ポーリングが FALSE の場合は、OIL にて指定された全ての ID を本テーブルに出力します。
TRUE の場合は送受信用の CAN スロットテーブル(CAN_SLOTID_TABLE)に配置できなかった ID のみ本テーブルに出力する。
また、本テーブルは OIL の設定(COM 受信割り込み)によって出力の制限は行われない。
各配列の要素ごとに 8 個の CAN-ID を管理している。
■配列の 1 つめの要素：CAN-ID 000h～007h までを管理

ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

←上位ビット　　下位ビット→

000h の CAN-ID を使用する場合は、対応するビット(この場合は最下位ビット)を 1 とし
1 つ目の要素の出力結果は”0x01”となる。

出力内容

・定義されたCAN-IDが 000h, 00Fhだった場合

```
const far UINT8 CAN_ASU_TABLE_COM[ CAN_DRV_ASU_MAX ]= {
    0x01, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,      /* CAN-ID: 000h - 03Fh */
    …
};
```

5.4.　OS 対応／非 OS 対応 LIN 通信ミドルウェア

5.4.1.　ノード情報(LINNODE)

5.4.1.1. インタフェースハンドル

OIL	設定値	出力ファイル

オブジェクト	属性	オブジェクト名称	linmw_id.h
LINNODE	－		
出力内容			
#define　TLIN_IFCHND_オブジェクト名称　　連続した値（1 から）			

5.4.1.2. レスポンスハンドル

OIL		設定値	出力ファイル
オブジェクト	属性	オブジェクト名称	linmw_id.h
LINNODE	－		
出力内容			
#define　TLIN_RESHND_オブジェクト名称　　連続した値（1 から）			

5.4.1.3. スケジュールテーブル選択

OIL		設定値	出力ファイル
オブジェクト	属性	MASTER／SLAVE	linmw_cfg.c
LINNODE	NODEATTR		
	SCHEDULE	オブジェクト参照	
説明			
NODEATTR 属性を MASTER に指定した場合は、そのマスターノードが管理するスケジュールテーブルを選択する。スケジュールテーブルの出力内容については別項目にて記載する。			

5.4.1.4. ダイアグテーブル

OIL		設定値	出力ファイル
オブジェクト	属性	USE／UNUSE	linmw_cfg.c
LINNODE	DIAG		
	DIAGADDR	1～255	
	DIRECT	SEND／RECEIVE	
説明			
ダイアグ使用/未使用は DIAG 属性に USE を指定すると 1(使用)を出力、UNUSE を指定すると 0(未使用)を出力する。 送受信方向は DIRECT 属性に SEND を指定すると 1(送信)を出力、RECEIVE を指定すると 0(受信)を出力する。 尚、送信データ配列、受信データ配列は OIL 記述に関係なく全て 0 を出力する。			
出力内容			

```
const TLinDiagRcd gCTlinDiagTable[] =
{
  { ダイアグ使用/未使用, 送受信方向, アドレス値, {送信データ配列}, {受信データ配列} }
};
```

5.4.2. シグナル情報(LINSIGNAL)

OIL の LINSIGNAL オブジェクトの内容が影響する出力について記載する。

5.4.2.1. シグナルテーブル

OIL		設定値	出力ファイル
オブジェクト	属性	1～16	linmw_id.h
LINSIGNAL	SIZE		
	INITVALUE	0～65535	
説明			
更新フラグは OIL の内容に関係なく 0 を出力する。 シグナルサイズ値は SIZE 属性で指定した値から 1 引いた値を出力する。 シグナル値は INITVALUE 属性で指定した値を出力する。			
出力内容			

```
const TLinSigRcd gCTLinSignalTable[] =
{
  {更新フラグ, シグナルサイズ値, シグナル値}
};
```

5.4.2.2. シグナルハンドラ定義

OIL		設定値	出力ファイル
オブジェクト	属性	オブジェクト名称	linmw_id.h
LINSIGNAL	－		
LINNODE	－	オブジェクト名称	
説明			
ノード名称は LINNODE オブジェクト名称、シグナル名称は LINSIGNAL オブジェクト名称を出力する。LINNODE オブジェクト内の SENDFRAME 属性(このフレーム内のシグナル)、RECEIVESIGNAL 属性により、どの LINNODE オブジェクトと関連付けされているかを判断する。			
出力内容			
#define TLIN_SIGHND_ノード名称_シグナル名称　　連続した値（1 から）			

5.4.3. フレーム情報(LINFRAME)

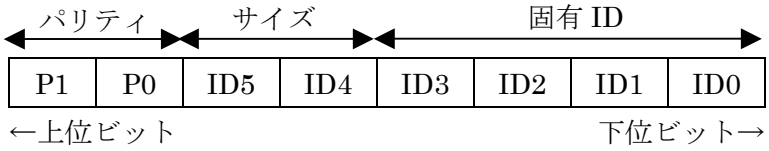
OIL の LINFRAME オブジェクトの内容が影響する出力について記載する。

5.4.3.1. フレームテーブル

OIL		設定値	出力ファイル
オブジェクト	属性	0～15	linmw_cfg.c
LINFRAME	FRAMEID		
	FRAMESIZE	2, 4, 8	
LINNODE	SENDFRAME	オブジェクト参照	
	RECEIVESIGNAL		
説明			

送受信方向、フレーム ID、シグナル数、先頭シグナルインデックス、フレームサイズの順に並んだものを 1 セットとして出力する。
送受信方向は LINNODE オブジェクトの SENDFRAME 属性に指定されている LINFRAME オブジェクトについては、送信側（出力 1）になる。RECEIVESIGNAL 属性に指定されている LINSIGNAL オブジェクトを持つ LINFRAME オブジェクトについては、受信側（出力 0）になる。但し、LINNODE オブジェクトに指定されていない LINFRAME オブジェクトの場合は、強制的に受信側とする（このケースは LINNODE がマスターの場合にあり得る）。LINNODE オブジェクトの SENDFRAME 属性、RECEIVESIGNAL 属性、および LINFRAME オブジェクトの USESIGNAL 属性の組合せにより、同じ ECU から送信も受信もするような LINFRAME オブジェクトとなる場合はエラーとなる。
フレーム ID は RAMEID 属性と FRAMESIZE 属性の指定によりフレーム ID が決まる。
詳細は下記図を参照。

■フレーム ID



■フレーム ID 選定

ビット	値
ID3～ID0 (固有 ID)	FRAMEID 属性値がそのまま入る
ID5,ID4 (サイズ)	以下の表により決まる ID5 / ID4 / FRAMESIZE 属性値 0 / 0 / 2 1 / 0 / 4 1 / 1 / 8
P0 (偶数パリティ)	ID0, ID1, ID2, ID4 の XOR
P1 (奇数パリティ)	ID1, ID3, ID4, ID5 の XOR の反転

シグナル数、先頭シグナルインデックスはフレームに含まれるシグナル数と、フレーム内の先頭のシグナルがシグナルオフセットテーブル（gCTLinSignalOffsetTable[]）の何番目かの情報を出力する。
フレームサイズは FRAMESIZE 属性を出力する。尚、AUTO を指定した場合は、フレーム内に詰込シグナルのサイズ・オフセット(SIGNAL 属性、OFFSET 属性)を基にし、サイズを自動的に決定して出力する。

出力内容

```
const TLinFrmRcd gCTLinFrameTable[] =
{
  { 送受信方向, フレーム ID, シグナル数, 先頭シグナルインデックス, フレームサイズ},
};
```

5.4.3.2. フレームハンドラ定義

OIL		設定値	出力ファイル
オブジェクト	属性	オブジェクト名称	linmw_id.h
LINFRAME	－		
LINNODE	－	オブジェクト名称	
説明			
LINNODE オブジェクト内の SENDFRAME 属性、RECEIVESIGNAL 属性(このシグナルを含むフレーム)により、どの LINNODE オブジェクトと関連付けされているかを判断する。TLIN_FRMHND_LINNODE オブジェクト名称_LINFRAME オブジェクト名称として出力する			
出力内容			
TLIN_FRMHND _ノード名称_フレーム名称　　連続した値（1 から）			

5.4.3.3. シグナルオフセットテーブル

OIL		設定値	出力ファイル
オブジェクト	属性	FORMAT	linmw_cfg.c
LINFRAME	USESIGNAL		
	SIGNAL	オブジェクト参照	
	OFFSET	0～63	
説明			
SIGNAL 属性で指定したシグナルオブジェクト名称、及び参照シグナルが属しているノード名称を出力する（シグナルハンドラ定義）、OFFSET 属性の値を出力する。左記組み合わせを一セットとして定義数分出力する。			
出力内容			

const TLinSigLnk gCTLinSignalOffsetTable[] =
{
{ TLIN_SIGHND_ノード名称_シグナル名称, オフセット値 },
};			

5.4.3.4. レスポンスレコードテーブル

OIL		設定値	出力ファイル
オブジェクト	属性	FORMAT	linmw_cfg.c

LINFRAME	USESIGNAL		
	SIGNAL	オブジェクト参照	
説明			
全フレームのシグナルハンドラ定義を出力する。			
出力内容			

const TLinResRcd gCTlinResponseRecord[] =
{
{ TLIN_FRMHND_ノード名称_フレーム名称 }
};			

5.4.3.5. レスポンステーブル

OIL		設定値	出力ファイル
オブジェクト	属性	―	linmw_cfg.c
LINFRAME	―		
説明			
各フレームが、どのノードに関係あるかの情報を知るため、ノードに関連するフレーム数と、関連するフレームの先頭の情報を持つ。			
出力内容			

const TLinResTbl gCTlinResponseTable[] =
{
{ フレーム数, 先頭フレームインデックス値},
};			

5.4.4. スケジュールテーブル情報(LINSCHEDULE)

OIL の LINSCHEDULE オブジェクトの内容が影響する出力について記載する。

5.4.4.1. スケジュールハンドル

OIL		設定値	出力ファイル
オブジェクト	属性	オブジェクト名称	linmw_id.h
LINSCHEDULE	―		
LINNODE	―		
説明			
LINSCHEDULE オブジェクトで定義した名称と関連する LINNNODE オブジェクト名称を出力する。			
出力内容			
#define TLIN_SCHHND_ノード名称_スケジュール名称　　連続した値（1 から）			

5.4.4.2. スケジュールレコードテーブル

OIL		設定値	出力ファイル
オブジェクト	属性	数値	linmw_cfg.c
LINSCHEDULE	DELAY		
	FRAME	オブジェクト参照	
説明			
SENDFRAME 属性内で指定した FRAME 属性のフレームハンドルを出力する。また、SENDFRAME 属性内で指定した DELAY 属性の値を msec 単位に変換して出力する(尚、OIL での設定単位は usec)。			
出力内容			

const TLinSchRcd gCTlinSchduleRecordTable[] =
{
{ TLIN_FRMHND_ノード名称_フレーム名称, ディレイ値},
};			

5.4.4.3. スケジュールテーブル

OIL		設定値	出力ファイル
オブジェクト	属性	オブジェクト名称	linmw_id.h
LINSCHEDULE	－		
説明			
スケジュールレコードテーブルのフレームハンドラ、ディレイ時間のセットを 1 レコードとする。各レコードがどのスケジュールに含まれるのか情報を知るため、スケジュール内のレコード数、スケジュールの先頭レコード情報を持つ。			
出力内容			

const TLinSchTbl gCTlinSchduleTable[] =
{
{ レコード数, 先頭レコードインデックス値},
};			

5.4.5. RAM 領域用エンティティテーブル

OIL		設定値	出力ファイル
オブジェクト	属性	－	linmw_cfg.c
－	－		
出力内容			

```
TlinSigRcd      gTLinSignalTable[ユーザ定義];
TlinSigLnk      gTLinSignalOffsetTable[ユーザ定義];
TlinFrmRcd      gTLinFrameTable[ユーザ定義];
TlinSchRcd      gTlinSchduleRecordTable[ユーザ定義];
TlinSchTbl      gTlinSchduleTable[ユーザ定義];
TlinResRcd      gTlinResponseRecord[ユーザ定義];
TlinResTbl      gTlinResponseTable[ユーザ定義];
TlinDiagRcd     gTlinDiagTable[ユーザ定義];
TlinEntityRcd     gTLinEntityRcd;
```

5.4.6. NULL ハンドル

OIL		設定値	出力ファイル
オブジェクト	属性	－	linmw_id.h
LINNODE	－		

説明
NULL ハンドルは定義されていないハンドルであることを識別するために使用する。各種の NULL ハンドルは OIL の記述に関係なく出力される。

出力内容
#define　TLIN_IFCHND_NULL　0
#define　TLIN_SIGHND_NULL　0
#define　TLIN_FRMHND_NULL　0
#define　TLIN_SCHHND_NULL　0
#define　TLIN_RESHND_NULL　0

6. 付録

6.1.　フィルタ関数のテンプレート

メッセージ管理情報（MESSAGE オブジェクト）のフィルタ（FILTER）属性で指定したフィルタ関数を下記に記載します。

6.1.1.　ALWAYS

ALWAYS の場合は以下のフォーマットで出力します。

[フィルタ関数]

```
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
{
    return E_OK;
}
```

[初期化関数]

なし

6.1.2.　NEVER

NEVER の場合は以下のフォーマットで出力します。

[フィルタ関数]

```
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
{
    return E_IL_FILTERED;
}
```

[初期化関数]

なし

6.1.3.　MASKEDNEWEQUALSX

MASKEDNEWEQUALSX の場合は以下のフォーマットで出力します。

[フィルタ関数]

```
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
{
    StatusType ercd = E_IL_FILTERED;
    if((*(DATATYPE *)p_in & (DATATYPE) マスク値 ) == (DATATYPE) 比較値 ){
```

```
            ercd = E_OK;
        }
        return ercd;
    }
```

[初期化関数]

なし

6.1.4. MASKEDNEWDIFFERSX

MASKEDNEWDIFFERSX の場合以下のフォーマットで出力します。

[フィルタ関数]

```
    static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
    static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
    {
        StatusType ercd = E_IL_FILTERED;
        if((*(DATATYPE *)p_in & (DATATYPE) マスク値 ) != (DATATYPE) 比較値 ){
            ercd = E_OK;
        }
        return ercd;
    }
```

[初期化関数]

なし

6.1.5. NEWISEQUAL

NEWISEQUAL の場合、以下のフォーマットで出力します。

[フィルタ関数]

```
    static DATATYPE old_value_オブジェクト名称;
    static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
    static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
    {
        StatusType ercd = E_IL_FILTERED;
        if(*(DATATYPE *)p_in == old_value_オブジェクト名称){
            old_value_オブジェクト名称 = *(DATATYPE *)p_in;
            ercd = E_OK;
        }
        return ercd;
    }
```

[初期化関数]

```
static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in);
static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in)
{
    old_value_オブジェクト名称 = *(DATATYPE *)p_in;
}
```

6.1.6. NEWISDIFFERENT

NEWISDIFFERENT の場合、以下のフォーマットで出力します。

[フィルタ関数]

```
static DATATYPE old_value_オブジェクト名称;
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
{
    StatusType ercd = E_IL_FILTERED;
    if(*(DATATYPE *)p_in != old_value_オブジェクト名称){
        old_value_オブジェクト名称 = *(DATATYPE *)p_in;
        ercd = E_OK;
    }
    return ercd;
}
```

[初期化関数]

```
static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in);
static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in)
{
    old_value_オブジェクト名称 = *(DATATYPE *)p_in;
}
```

6.1.7. MASKEDNEWEQUALMASKEDOLD

MASKEDNEWEQUALMASKEDOLD の場合、以下のフォーマットで出力します。

[フィルタ関数]

```
static DATATYPE old_value_オブジェクト名称;
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
{
    StatusType ercd = E_IL_FILTERED;
    if((*(DATATYPE *)p_in & (DATATYPE)マスク値) == (old_value_オブジェクト名称 &
(DATATYPE)マスク値)){
```

```
            old_value_オブジェクト名称 = *(DATATYPE *)p_in;
            ercd = E_OK;
        }
        return ercd;
    }
```

[初期化関数]

```
    static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in);
    static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in)
    {
        old_value_オブジェクト名称 = *(DATATYPE *)p_in;
    }
```

6.1.8. MASKEDNEWDIFFERSMASKEDOLD

MASKEDNEWDIFFERSMASKEDOLD の場合、以下のフォーマットで出力します。

[フィルタ関数]

```
    static DATATYPE old_value_オブジェクト名称;
    static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
    static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
    {
        StatusType ercd = E_IL_FILTERED;
        if((*(DATATYPE *)p_in & (DATATYPE)マスク値) != (old_value_オブジェクト名称 &
(DATATYPE)マスク値)){
            old_value_オブジェクト名称 = *(DATATYPE *)p_in;
            ercd = E_OK;
        }
        return ercd;
    }
```

[初期化関数]

```
    static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in);
    static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in)
    {
        old_value_オブジェクト名称 = *(DATATYPE *)p_in;
    }
```

6.1.9. NEWISWITHIN

NEWISWITHIN の場合、以下のフォーマットで出力します。

[フィルタ関数]

```
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
{
    StatusType ercd = E_IL_FILTERED;
    if(最低値 < *(DATATYPE *)p_in && *(DATATYPE *)p_in < 最大値){
        ercd = E_OK;
    }
    return ercd;
}
```

[初期化関数]

なし

6.1.10. NEWISOUTSIDE

NEWISOUTSIDE の場合、以下のフォーマットで出力します。

[フィルタ関数]

```
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
{
    StatusType ercd = E_IL_FILTERED;
    if(最低値 > *(DATATYPE *)p_in || *(DATATYPE *)p_in > 最大値){
        ercd = E_OK;
    }
    return ercd;
}
```

[初期化関数]

なし

6.1.11. NEWISGREATER

NEWISGREATER の場合、以下のフォーマットで出力します。

[フィルタ関数]

```
static DATATYPE old_value_オブジェクト名称;
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
{
    StatusType ercd = E_IL_FILTERED;
    if(*(DATATYPE *)p_in > old_value_オブジェクト名称){
        old_value_オブジェクト名称 = *(DATATYPE *)p_in;
```

```
            ercd = E_OK;
        }
        return ercd;
    }
```

[初期化関数]

```
    static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in);
    static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in)
    {
        old_value_オブジェクト名称 = *(DATATYPE *)p_in;
    }
```

6.1.12. NEWISLESSOREQUAL

NEWISLESSOREQUAL の場合、以下のフォーマットで出力します。

[フィルタ関数]

```
    static DATATYPE old_value_オブジェクト名称;
    static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
    static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
    {
        StatusType ercd = E_IL_FILTERED;
        if(*(DATATYPE *)p_in <= old_value_オブジェクト名称){
            old_value_オブジェクト名称 = *(DATATYPE *)p_in;
            ercd = E_OK;
        }
        return ercd;
    }
```

[初期化関数]

```
    static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in);
    static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in)
    {
        old_value_オブジェクト名称 = *(DATATYPE *)p_in;
    }
```

6.1.13. NEWISLESS

NEWISLESS の場合、以下のフォーマットで出力します。

[フィルタ関数]

```
    static DATATYPE old_value_オブジェクト名称;
    static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
```

```
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
{
    StatusType ercd = E_IL_FILTERED;
    if(*(DATATYPE *)p_in < old_value_オブジェクト名称){
        old_value_オブジェクト名称 = *(DATATYPE *)p_in;
        ercd = E_OK;
    }
    return ercd;
}
```

[初期化関数]

```
static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in);
static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in)
{
    old_value_オブジェクト名称 = *(DATATYPE *)p_in;
}
```

6.1.14. NEWISGREATEROREQUAL

NEWISGREATEROREQUAL の場合、以下のフォーマットで出力します。

[フィルタ関数]

```
static DATATYPE old_value_オブジェクト名称;
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
{
    StatusType ercd = E_IL_FILTERED;
    if(*(DATATYPE *)p_in >= old_value_オブジェクト名称){
        old_value_オブジェクト名称 = *(DATATYPE *)p_in;
        ercd = E_OK;
    }
    return ercd;
}
```

[初期化関数]

```
static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in);
static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in)
{
    old_value_オブジェクト名称 = *(DATATYPE *)p_in;
}
```

6.1.15. ONEEVERYN

ONEEVERYN の場合、以下のフォーマットで出力します。

[フィルタ関数]

```
static int occurrence_オブジェクト名称;
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in);
static StatusType filter_func_オブジェクト名称 (ApplicationDataRef p_in)
{
    StatusType ercd = E_IL_FILTERED;
    if(occurrence_オブジェクト名称 % 周期 == 設定値){
        occurrence_オブジェクト名称 -= 周期;
        ercd = E_OK;
    }
    return ercd;
}
```

[初期化関数]

```
static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in);
static void filter_init_func_オブジェクト名称(ApplicationDataRef p_in)
{
    occurrence_オブジェクト名称 = 0;
}
```

変更履歴

Version	Date	Detail	Editor
1.00	2007/03/20	・　新規作成	ヴィッツ
1.01	2007/04/16	・　非 OS 対応 CAN 通信ミドルウェア(DirectNM)の項を追加 ・　非 OS 対応 CAN 通信ミドルウェア(InDirectNM)の項を追加	ヴィッツ
		・	