CanLine-1
.NET ライブラリ
プログラマーズ・マニュアル

版数	発行日	改訂履歴
第１版	2024 年３月	初版発行
第２版	2024 年７月	・プロパティに下記を追加 ・Can1ScheduleEnd ・Can2ScheduleEnd ・LINScheduleEnd ・CAN 送信 BOX を 10 個→20 個に拡張 ・CanStartScheduleSend 関数の追加 ・関数名の変更 ・CanStopIntervalSend→CanStopSend ・LinStartMasterSimulation→ LinStartMasterSimulation1 ・LinStartMasterSimulation2 関数の追加 ・LIN 送信 BOX を 10 個→20 個に拡張 **※これらの機能を使用するにはファームウェアバージョン 1.1.0 以上である必要があります。要件に満たない場合、ユーザーズ・マニュアルのファームウェアのアップデートの手順に従い、ファームウェアのアップデートを実行してください。**
第３版	2024 年９月	・OpenEx 関数を追加 ・GetProductIdList 関数を追加

目次

1. 概要

CanLine-1.NET ライブラリは CanLine-1 本体を制御するための.NET 用ダイナミックリンクライブラリ（DLL）です。これにより開発環境 Visual Studio の VB、C#などで CanLine-1 を活用したオリジナルアプリケーションの製作が可能です。

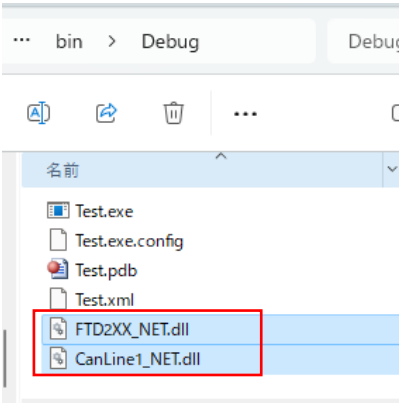
2. DLL の参照追加

前準備）

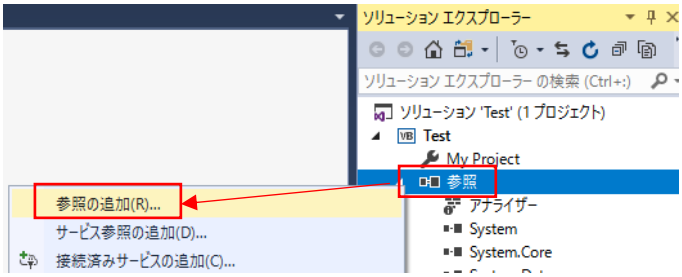
- 下記から.Net 用ライブラリのダウンロード
  https://hsdev.co.jp/canline-1/
- CanLine-1 付属 CD の USB ドライバーをインストール

① Visual Studio で生成した実行ファイルと同じフォルダに下記２個の DLL を格納します。

- FTD2XX_NET.dll（USB 用ライブラリ）
- CanLine1_NET.dll（CanLine-1 用ライブラリ）



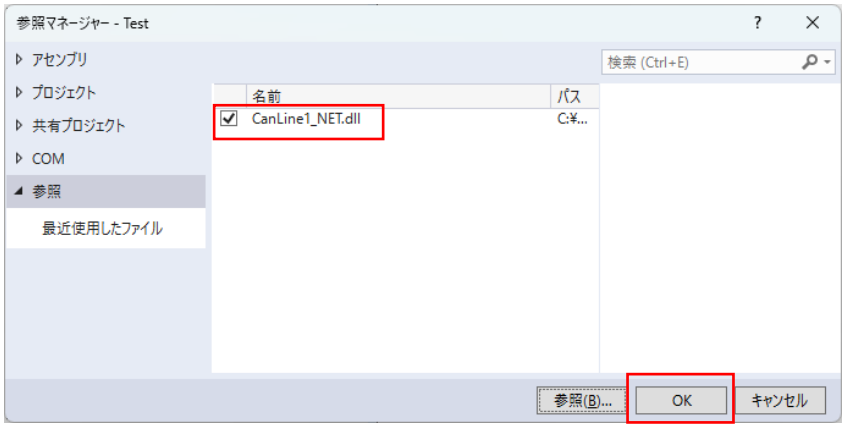
② ソリューションエクスプローラから参照を右クリックして、参照の追加を押す



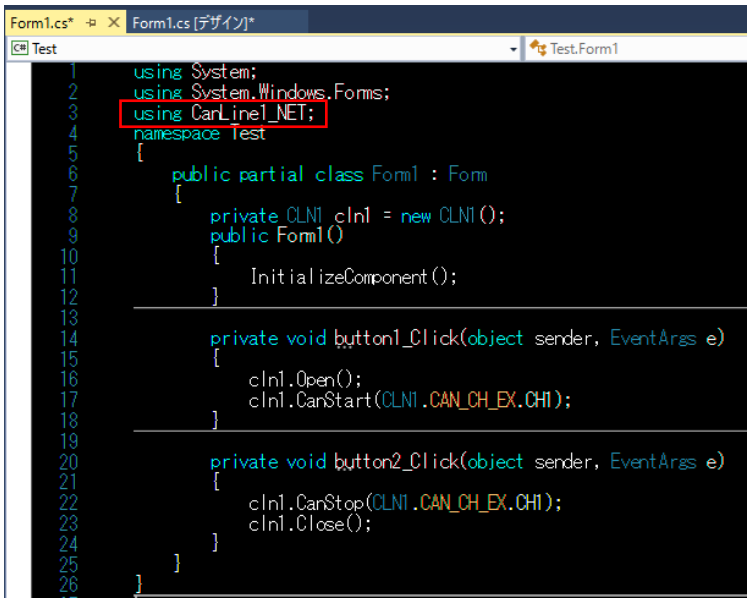
③ 参照ボタンを押す

④ さきほど格納した CanLine1_NET.dll を選択する。

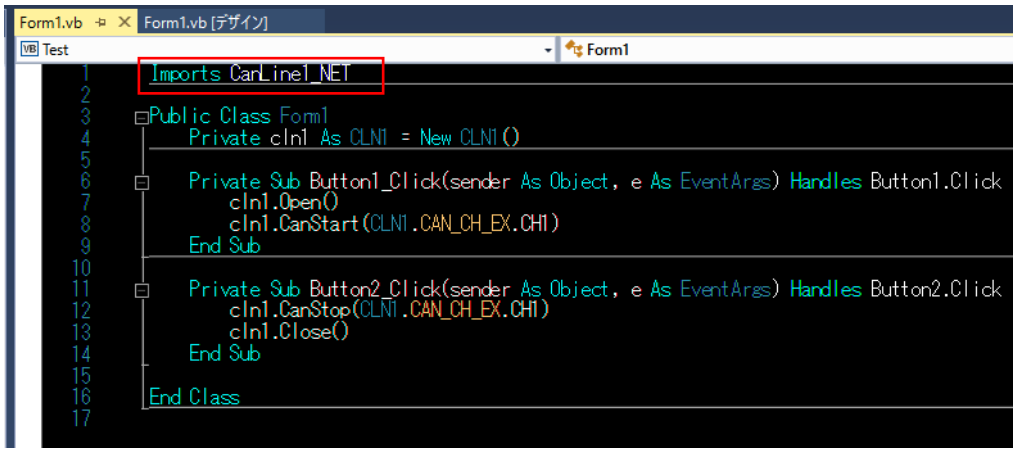
⑤ CanLine1_NET.dll にチェックを入れて OK を押す



⑥ C#であれば、ソースコードヘッダー部分に「using CanLine1_NET」の追加、



Visual Basic の場合は「Imports CanLine1_NET」を追加



3. プロパティ

[書式]

```
public bool IsOpen { get; }
public CAN_STATUS Can1Status { get; }
public CAN_STATUS Can2Status { get; }
public CAN_MODE Can1Mode { get; }
public CAN_MODE Can2Mode { get; }
public int CanRxMsgCount { get; }
public int CanRxMsgBufferSize { get; set; }
public LIN_STATUS LinStatus { get; }
public int LinRxMsgCount { get; }
public int LinRxMsgBufferSize { get; set; }
public string OtherErrorMsg { get; }
public bool Can1ScheduleEnd{ get; }
public bool Can2ScheduleEnd{ get; }
public bool LinScheduleEnd{ get; }
```

[詳細]

プロパティ名	Read/ Write	内容	値
IsOpen	Read	USB ドライバー オープン状態	false：オープン状態 true：クローズ状態
Can1Status	Read	CAN1 バス ステータス	・CAN_STATUS.BUSOFF：オフバス ・CAN_STATUS.BUSON：オンバス ・CAN_STATUS.ERR_PASIVE：エラーパッシブ ・CAN_STATUS.ERR_BUSOFF：エラーによるバスオフ
Can2Status	Read	CAN2 バス ステータス	・CAN_STATUS.BUSOFF：オフバス ・CAN_STATUS.BUSON：オンバス ・CAN_STATUS.ERR_PASIVE：エラーパッシブ ・CAN_STATUS.ERR_BUSOFF：エラーによるバスオフ
Can1Mode	Read	CAN1 モード	・CAN_MODE.SILENT：サイレントモード ・CAN_MODE.NORMAL：ノーマルモード

Can2Mode	Read	CAN2 モード	・CAN_MODE.SILENT：サイレントモード ・CAN_MODE.NORMAL：ノーマルモード
CanRxMsgCount	Read	CAN 通知 メッセージ数	0～CAN 通知メッセージバッファーサイズ CanGetMessage 関数でここで取得した数のメッセージが取得可能
CanRxMsgBufferSize	Read/ Write	CAN 通知 メッセージ バッファーサイズ	初期値は 50000 超えた場合、古いメッセージから削除
LinStatus	Read	LIN バス ステータス	・LIN_STATUS.BUSOFF：オフバス ・LIN_STATUS.BUSON：オンバス ・LIN_STATUS.TX_ERROR：送信エラー
LinRxMsgCount	Read	LIN 通知 メッセージ数	0～LIN 通知メッセージバッファーサイズ LinGetMessage 関数でここで取得した数のメッセージが取得可能
LinRxMsgBufferSize	Read/ Write	LIN 通知 メッセージ バッファーサイズ	初期値は 50000 超えた場合、古いメッセージから削除
OtherErrorMsg	Read	エラー メッセージ	各関数の戻り値で CL_RETURN.OTHER_ERROR（その他エラー）が発生時した時の詳細メッセージ
Can1ScheduleEnd ※ファームウェアバージョン 1.1.0 以上で使用可	Read	CAN1 スケジュール送信完了	true 条件：CAN1 スケジュール送信完了時 ※繰り返し回数が 0 の無限繰り返しの場合、true になりません。 false 条件：スケジュール送信開始時
Can2ScheduleEnd ※ファームウェアバージョン 1.1.0 以上で使用可	Read	CAN2 スケジュール送信完了	true 条件：CAN2 スケジュール送信完了時 ※繰り返し回数が 0 の無限繰り返しの場合、true になりません。 false 条件：スケジュール送信開始時
LinScheduleEnd ※ファームウェアバージョン 1.1.0 以上で使用可	Read	LIN スケジュール送信完了	true 条件：LIN スケジュール送信完了時 ※繰り返し回数が 0 の無限繰り返しの場合、true になりません。 false 条件：スケジュール送信開始時

4. 関数一覧

CAN/LIN 共通関数

関数名	機能	説明
Open	オープン	USB ドライバーをオープンします。
Close	クローズ	USB ドライバーをクローズします。
GetFirmVersion	ファームウェアバージョン読出し	CanLine-1 本体のファームウェアのバージョンを読み出します。
OpenEx	オープン(製品 ID 指定)	製品 ID を指定してＵＳＢドライバーをオープンします。１台のＰＣで複数台の本体接続に対応します。
GetProductIdList	製品 ID リスト取得	PC 接続中の製品 ID リストを取得します。

CAN 関数

関数名	機能	説明
CanSetConfig	CAN 通信設定	CAN の通信設定を更新します。
CanSetIdFilter	CAN　ID フィルタ設定	CAN ID にフィルターをかけます。
CanStart	CAN 通信開始	CAN 通信を開始します。
CanStop	CAN 通信停止	CAN 通信を停止します。
CanSingleSend	CAN 単発送信	CAN 単発送信を行います。
CanSetSendBox	CAN 送信 BOX 更新	CAN 定期送信用の送信 BOX を更新します。
CanStartIntervalSend	CAN 定期送信開始	CAN 定期送信を開始します。
CanStartScheduleSend	CAN スケジュール送信開始	CAN スケジュール送信を開始します。 ※ファームウェアバージョン 1.1.0 以上で使用可
CanStopSend	CAN 定期送信停止	CAN 定期送信／スケジュール送信を停止します。
CanGetMessage	CAN 通知メッセージ取得	CAN 通知メッセージを取得します。

LIN 関数

関数名	機能	説明
LinSetConfig	LIN 通信設定	LIN 通信設定を更新します。
LinStart	LIN 通信開始	LIN 通信を開始します。
LinStop	LIN 通信停止	LIN 通信を終了します。
LinSingleSend	LIN 単発送信	LIN 単発送信を行います。
LinSetMasterSendBox	LIN マスターシミュレーション用送信 BOX 更新	LIN マスターシミュレーション(定期送信／スケジュール送信）用の送信 BOX を更新します。
LinStartMasterSimulation1	LIN マスターシミュレーション（定期送信）開始	LIN マスターシミュレーション（定期送信）を開始します。
LinStartMasterSimulation2	LIN マスターシミュレーション（スケジュール送信）開始	LIN マスターシミュレーション（スケジュール送信）を開始します。 ※ファームウェアバージョン 1.1.0 以上で使用可
LinSetSlaveSendBox	LIN スレーブシミュレーション用送信 BOX 更新	LIN スレーブシミュレーション（応答）用の送信 BOX を更新します。
LinStartSlaveSimulation	LIN スレーブシミュレーション開始	LIN スレーブシミュレーションを開始します。
LinStopSimulation	LIN シミュレーション停止	LIN シミュレーション(マスター／スレーブ)を停止します。
LinGetMessage	LIN 通知メッセージ取得	LIN 通知メッセージを取得します。

5. 関数の説明

5.1.　Open：オープン

[説明]

USB ドライバーをオープンします。CanLine-1 の制御を開始するにはまずこの関数を実行します。この関数が正常終了すると、その他の関数が実行可能になります。

[書式]

CL_RETURN Open()

[引数]

なし

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_FOUND	デバイス(CanLine-1 本体)が見つかりません。
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.2.　Close：クローズ

[説明]

USB ドライバーをクローズします。

CanLine-1 の制御を終了する場合は、本関数を実行します。

[書式]

CL_RETURN Close()

[引数]

なし

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.3.　GetFirmVersion：ファームウェアバージョン読出し

[説明]

CanLine-1 本体のファームウェアのバージョンを読み出します。

[書式]

CL_RETURN GetFirmVersion(ref string version)

[引数]

変数	I/O	内容	値
version	OUT	ファームウェアバージョン	[例]1.0.1

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.DEVICE_RESPONSE_ERROR	CanLine-1 本体から応答が異常
CL_RETURN.DEVICE_NO_RESPONSE	CanLine-1 本体から応答がない
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.4.　OpenEx：オープン(製品 ID 指定)

[説明]

製品 ID を指定して USB ドライバーをオープンします。CanLine-1 の制御を開始するにはまずこの関数を実行します。この関数が正常終了すると、その他の関数が実行可能になります。

※1 台の PC で CanLine-1 本体を複数台、同時に使用する場合、本関数を使用します。製品 ID は CanLine-1 標準アプリのメニューの設定->システム設定から調べることが可能です。

[書式]

CL_RETURN OpenEx(string productID)

[引数]

変数	I/O	内容	値
productID	IN	製品 ID	[例] AB0NZ3P8

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_FOUND	指定した製品 ID のデバイス(CanLine-1 本体)が見つかりません。
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.5.　GetProductIdList：製品 ID リスト取得

[説明]

PC 接続中の本体の製品 ID リストを取得します。本関数は Open 関数、OpenEx 関数実行前にコールします。

既に Open 関数又は OpenEx 関数を実行している本体は製品 ID リストに含まれません。

[書式]

CL_RETURN GetProductIdList (ref List<string> lstProductId)

[引数]

変数	I/O	内容	値
lstProductId	OUT	製品 ID リスト	[例] AB0NZ3P8、AB0NZ3P7…

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_FOUND	デバイス(CanLine-1 本体)が見つかりません。
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.6. CanSetConfig：CAN 通信設定

[説明]

CAN の通信設定を更新します。CAN 通信開始前に行います。CanStop（CAN 通信停止）を実行するとリセットされるので、CanStart （CAN 通信開始）前に毎回実行してください。

[書式]

CL_RETURN CanSetConfig(CAN_CH chNo, CAN_CONFIG config)

[引数]

変数名	I/O	内容	値
chNo	IN	チャンネル No	・CAN_CH.CH1：CAN1 ・CAN_CH.CH2：CAN2
config	IN	CAN 通信設定情報	CAN_CONFIG メンバーは下記参照

CAN_CONFIG メンバー

```
public class CAN_CONFIG{
        public CAN_TERM Terminator;
        public CAN_MODE Mode;
        public CAN_RATE BaudRate;
        public CAN_SP SamplePoint;
}
```

変数	内容	値
Terminator	終端抵抗（120Ω）	・CAN_TERM.OFF：終端抵抗無し ・CAN_TERM.ON：終端抵抗有り
Mode	CAN モード	・CAN_MODE.SILENT：サイレントモード ・CAN_MODE.NORMAL：ノーマルモード
BaudRate	通信速度[bps]	・CAN_RATE._125：125Kbps ・CAN_RATE._250：250Kbps ・CAN_RATE._500：500Kbps ・CAN_RATE._1000K：1000Kbps
SamplePoint	サンプルポイント	・CAN_SP._60Per：60% ・CAN_SP._70Per：70% ・CAN_SP._80Per：80%

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.CAN_STARTED	CAN 通信開始関数が既に実行されている。※本関数は CAN 通信開始前に実行してください。
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.DEVICE_RESPONSE_ERROR	CanLine-1 本体から応答が異常
CL_RETURN.DEVICE_NO_RESPONSE	CanLine-1 本体から応答がない
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.7. CanSetIdFilter：CAN ID フィルタ設定

[説明]

受信したいCAN IDの範囲を指定することで、ID にフィルターをかけます。最大 10 種類の範囲を指定できます。CAN 通信開始前に行います。CanStop（CAN 通信停止）を実行するとリセットされるので、フィルターを有効にする場合は、CanStart （CAN 通信開始）前に毎回実行してください。

[書式]

CL_RETURN CanSetIdFilter(CAN_CH chNo, CAN_ID_FILTER filter)

[引数]

変数	I/O	内容	値
chNo	IN	チャンネル No	・CAN_CH.CH1：CAN1 ・CAN_CH.CH2：CAN2
filter	IN	CAN ID フィルタ情報	CAN_ID_FILTER メンバーは下記参照

CAN_ID_FILTER メンバー

```
public class CAN_ID_FILTER{
        public int FilterNo;
        public CAN_ID_TYPE IdType;
        public UInt32 FromID;
        public UInt32 ToID;
}
```

変数	内容	値
FilterNo	フィルターNo	0～9 最大 10 種類設定可能。
IdType	CAN ID タイプ	・CAN_ID_TYPE.STD：標準 ID（11bit） ・CAN_ID_TYPE.EXT：拡張 ID（29bit）
FromID	取得範囲開始 ID	CAN ID タイプが標準 ID の場合 0～0x7FF、 拡張 ID の場合 0～0x1FFFFFFF
ToID	取得範囲終了 ID	CAN ID タイプが標準 ID の場合 0～0x7FF、 拡張 ID の場合 0～0x1FFFFFFF

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.CAN_STARTED	CAN 通信開始関数が既に実行されている。※本関数は CAN 通信開始前に実行してください。
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.DEVICE_RESPONSE_ERROR	CanLine-1 本体から応答が異常
CL_RETURN.DEVICE_NO_RESPONSE	CanLine-1 本体から応答がない
PARAMETER_ERROR	引数が範囲外
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.8.　CanStart：CAN 通信開始

[説明]

CAN 通信を開始します。本関数実行後に CAN の送信、受信が可能になります。

[書式]

CL_RETURN CanStart(CAN_CH_EX chNo)

[引数]

変数	I/O	内容	値
chNo	IN	チャンネル No	• CAN_CH_EX.CH1：CAN1 • CAN_CH_EX.CH2：CAN2 • CAN_CH_EX.ALL：CAN1、CAN2 両方

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.DEVICE_RESPONSE_ERROR	CanLine-1 本体から応答が異常
CL_RETURN.DEVICE_NO_RESPONSE	CanLine-1 本体から応答がない
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.9.　CanStop：CAN 通信停止

[説明]

CAN 通信を停止します。

[書式]

CL_RETURN CanStop(CAN_CHANNEL_EX chNo)

[引数]

変数	I/O	内容	値
chNo	IN	チャンネル No	• CAN_CH_EX.CH1：CAN1 • CAN_CH_EX.CH2：CAN2 • CAN_CH_EX.ALL：CAN1、CAN2 両方

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.10. CanSingleSend：CAN 単発送信

[説明]

CAN 単発送信を行います。

[書式]

CL_RETURN CanSingleSend(CAN_CH chNo, CAN_TX_MSG txMsg)

[引数]

変数	I/O	内容	値
chNo	IN	チャンネル No	・CAN_CH.CH1：CAN1 ・CAN_CH.CH2：CAN2
txMsg	IN	CAN 送信メッセージ情報	CAN_TX_MSG メンバーは下記参照

CAN_TX_MSG メンバー

```
public class CAN_TX_MSG{
        public CAN_ID_TYPE IdType;
        public UInt32 Id;
        public byte Dlc;
        public byte[] Data = new byte[8];
}
```

変数	内容	値
IdType	CAN ID タイプ	・CAN_ID_TYPE.STD：標準 ID（11bit） ・CAN_ID_TYPE.EXT：拡張 ID（29bit）
Id	CAN ID	CAN ID タイプが標準 ID の場合 0～0x7FF、 拡張 ID の場合 0～0x1FFFFFFF
Dlc	データ長	1～8
Data[0]～[8]	データ	0～0xFF

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.CAN_NOT_STARTED	CAN 通信開始関数が実行されていない。
CL_RETURN.CAN_MODE_SILENT	CAN モードがサイレントモードである。本関数はノーマルモードで実行して下さい。
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.11. CanSetSendBox：CAN 送信 BOX の更新

[説明]

CAN 送信 BOX を更新します。送信 BOX は最大 20 個まで設定可能です。

本関数実行後、CanStartIntervalSend（CAN 定期送信開始）関数を実行することで定期送信、CanStartSucheduleSend（CAN スケジュール送信開始）関数を実行することでスケジュール送信が開始されます。定期送信／スケジュール送信中でも本関数は実行可能で、その場合、送信中のデータが更新されます。

※送信 BOX11～20 はファームウェアバージョン 1.1.0 以上で使用可

[書式]

CL_RETURN CanSetSendBox(CAN_CH chNo, int txBoxNo, CAN_TX_MSG txMsg, UInt32 time)

[引数]

変数	I/O	内容	値
chNo	IN	CAN チャンネル No	・CAN_CH.CH1：CAN1 ・CAN_CH.CH2：CAN2
txBoxNo	IN	送信 BOX No	0～19（0=送信 BOX1…19=送信 BOX20）
txMsg	IN	CAN 送信メッセージ情報	CAN_TX_MSG メンバーは下記参照
time	IN	定期送信の周期[ms]／スケジュール送信の時間[ms]	1～99999

CAN_TX_MSG メンバー

```
public class CAN_TX_MSG{
        public CAN_ID_TYPE IdType;
        public UInt32 Id;
        public byte Dlc;
        public byte[] Data = new byte[8];
}
```

変数	内容	値
IdType	CAN ID タイプ	・CAN_ID_TYPE. STD：標準 ID（11bit） ・CAN_ID_TYPE. EXT：拡張 ID（29bit）
Id	CAN ID	CAN ID タイプが標準 ID の場合 0～0x7FF、 拡張 ID の場合 0～0x1FFFFFFF
Dlc	データ長	1～8
Data[0]～[8]	データ	0～0xFF

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
PARAMETER_ERROR	引数が範囲外
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.12. CanStartIntervalSend：CAN 定期送信開始

[説明]

CanSetSendBox（CAN 送信 BOX の更新）関数実行後、本関数を実行することで CAN 定期送信を開始します。送信 BOX No1～20 の有効／無効を設定します。

※送信 BOX11～20 はファームウェアバージョン 1.1.0 以上で使用可

[書式]

CL_RETURN CanStartIntervalSend(CAN_CH chNo, bool[] txBoxNoEnable)

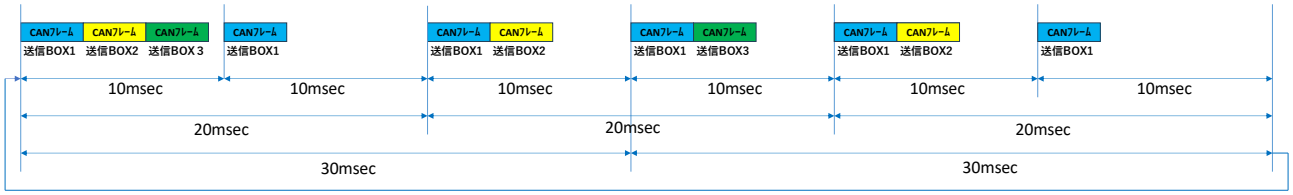
[引数]

変数	I/O	内容	値
chNo	IN	CAN チャンネル No	• CAN_CH.CH1：CAN1 • CAN_CH.CH2：CAN2
txBoxNoEnable [0]～[19]	IN	txBoxNoEnable[0]：送信 BOX No1 有効／無効 txBoxNoEnable[1]：送信 BOX No2 有効／無効 ・・・ txBoxNoEnable[19]：送信 BOX No20 有効／無効	• false：無効 • true：有効

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.CAN_NOT_STARTED	CAN 通信開始関数が実行されていない。
CL_RETURN.CAN_MODE_SILENT	CAN モードがサイレントモードである。本関数はノーマルモードで実行して下さい。
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

BOX1=10msec
BOX2=20msec
BOX3=30msec
定期送信は上記設定の場合、下記タイミングチャートのようにBOX1～BOX3まで一斉に送信が開始される。



5.13. CanStartScheduleSend：CAN スケジュール送信開始

[説明]

CanSetSendBox（CAN 送信 BOX の更新）関数実行後、本関数を実行することで CAN スケジュール送信を開始します。送信 BOX No1～20 の有効／無効を設定します。繰り返し回数に 0 を指定することで CanStopSend 関数実行まで繰り返します。1 以上を設定した場合、指定した回数まで繰り返します。繰り返しの完了は Can1ScheduleEnd プロパティ、Can2ScheduleEnd プロパティで判別可能です。

※本関数はファームウェアバージョン 1.1.0 以上で使用可

[書式]

CL_RETURN CanStartScheduleSend(CAN_CH chNo, bool[] txBoxNoEnable, UInt16 repeatCount)

[引数]

変数	I/O	内容	値
chNo	IN	CAN チャンネル No	・CAN_CH.CH1：CAN1 ・CAN_CH.CH2：CAN2
txBoxNoEnable [0]～[19]	IN	txBoxNoEnable[0]：送信 BOX No1 有効／無効 txBoxNoEnable[1]：送信 BOX No2 有効／無効 ・・・ txBoxNoEnable[19]：送信 BOX No20 有効／無効	・false：無効 ・true：有効
repeatCount	IN	繰り返し回数	0=無限繰り返し 1～999 回

[戻り値]

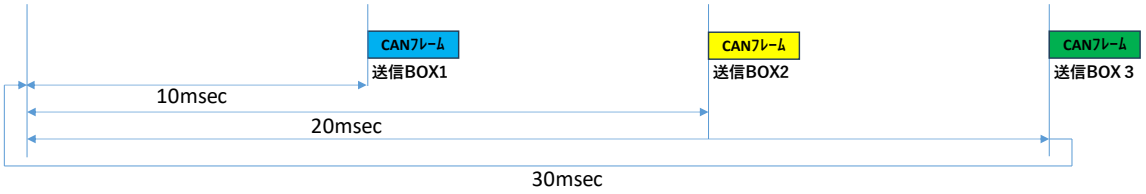
値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.CAN_NOT_STARTED	CAN 通信開始関数が実行されていない。
CL_RETURN.CAN_MODE_SILENT	CAN モードがサイレントモードである。本関数はノーマルモードで実行して下さい。
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

BOX1=10msec

BOX2=20msec

BOX3=30msec

スケジュール送信は上記設定の場合、下記タイミングチャートのようにBOX1から順番に送信されBOX3送信後、再度BOX1から送信される。



5.14. CanStopSend：CAN 定期送信／スケジュール送信停止

[説明]

CAN 定期送信／スケジュール送信を停止します。

[書式]

CL_RETURN CanStopSend(CAN_CH chNo)

[引数]

変数	I/O	内容	値
chNo	IN	CAN チャンネル No	・CAN_CH.CH1：CAN1 ・CAN_CH.CH2：CAN2

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.15. CanGetMessage：CAN 通知メッセージ取得

[説明]

CAN 送信完了、受信発生の通知メッセージを取得します。

[書式]

Int CanGetMessage(ref List<CAN_RX_MSG> lstRxMsg, int count)

[引数]

変数	I/O	内容	値
lstRxMsg	OUT	CAN 通知メッセージ情報リスト	CAN_RX_MSG メンバーは下記参照
count	IN	取得するメッセージ数	CanRxMsgCount プロパティで受信メッセージ数を取得して、それ以下を指定してください。

CAN_RX_MSG メンバー

```
public class CAN_RX_MSG{
        public UInt32 TimeStamp;
        public CAN_RX_MSG_TYPE Type;
        public CAN_ID_TYPE IdType;
        public UInt32 Id;
        public byte Dlc;
        public byte[] Data = new byte[8];
}
```

変数	内容	値
TimeStamp	タイムスタンプ	0～4294967295 最小単位：0.1msec [例]123=12.3msec
Type	CAN 通知メッセージタイプ	• CAN_RX_MSG_TYPE.CAN1_RECIEVD：CAN1 受信 • CAN_RX_MSG_TYPE.CAN1_SENT：CAN1 送信完了 • CAN_RX_MSG_TYPE.CAN2_RECIEVD：CAN2 受信 • CAN_RX_MSG_TYPE.CAN2_SENT：CAN2 送信完了
IdType	CAN ID タイプ	• CAN_ID_TYPE.STD：標準 ID（11bit） • CAN_ID_TYPE.EXT：拡張 ID（29bit）
Id	CAN ID	CAN ID タイプが標準 ID の場合 0～0x7FF、 拡張 ID の場合 0～0x1FFFFFFF
Dlc	データ長	1～8
Data[0]～[8]	データ	0～0xFF

[戻り値]

値	内容
int	取得したメッセージ数

5.16. LinSetConfig：LIN 通信設定

[説明]

LIN 通信設定を更新します。

[書式]

LinSetConfig(LIN_CONFIG config)

[引数]

変数	I/O	内容	値
config	IN	LIN 通信設定情報	LIN_CONFIG メンバーは下記を参照

LIN_CONFIG メンバー

```
public class LIN_CONFIG{
        public LIN_RATE BaudRate;
        public LIN_PULLUP PullUpp;
}
```

変数	内容	値
BaudRate	LIN 通信速度[bps]	• LIN_RATE._2400：2400bps • LIN_RATE._4800：4800bps • LIN_RATE._9600：9600bps • LIN_RATE._19200：19200bps
PullUpp	マスタープルアップ抵抗（1kΩ）	• LIN_PULLUP.OFF：プルアップ抵抗無し • LIN_PULLUP.ON：プルアップ抵抗有り

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.LIN_STARTED	LIN 通信開始関数が既に実行されている。※本関数は LIN 通信開始前に実行してください。
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.DEVICE_RESPONSE_ERROR	CanLine-1 本体から応答が異常
CL_RETURN.DEVICE_NO_RESPONSE	CanLine-1 本体から応答がない
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.17. LinStart：LIN 通信開始

[説明]

LIN 通信を開始します。本関数実行後に LIN の送信、受信が可能になります。

[書式]

CL_RETURN LinStart()

[引数]

なし

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.18. LinStop：LIN 通信終了

[説明]

LIN 通信を終了します。

[書式]

CL_RETURN LinStop()

[引数]

なし

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.19. LinSingleSend：LIN 単発送信

[説明]

LIN 単発送信を行います。

[書式]

CL_RETURN LinSingleSend(LIN_TX_MSG txMsg)

[引数]

変数	I/O	内容	値
txMsg	IN	LIN 送信メッセージ情報	LIN_TX_MSG メンバーは下記を参照

LIN_TX_MSG メンバー

```
public class LIN_TX_MSG{
        public byte Id;
        public byte Dlc;
        public byte[] Data = new byte[8];
        public LIN_SUM_TYPE SumType;
}
```

変数	内容	
Id	LIN ID	0～0x3F
Dlc	データ長	0～8
Data[0] ～ [8]	データ	0～0xFF
SumType	チェックサムタイプ	・LIN_SUM_TYPE.CLASSIC：標準チェックサム ・LIN_SUM_TYPE.ENHANCED：拡張チェックサム

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.LIN_NOT_STARTED	LIN 通信開始関数が実行されていない。
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.20. LinSetMasterSendBox：LIN マスターシミュレーション用送信 BOX 更新

[説明]

LIN 送信 BOX を更新します。送信 BOX は最大 20 個まで設定可能です。

本関数実行後、LinStartMasterSimulation1（LIN マスターシミュレーション（定期送信）開始）を実行することで定期送信、LinStartMasterSimulation2（LIN マスターシミュレーション（スケジュール送信）開始）関数を実行することでスケジュール送信が開始されます。定期送信／スケジュール送信中でも本関数は実行可能で、その場合、送信中のデータが更新されます。

※送信 BOX11～20 ファームウェアバージョン 1.1.0 以上で使用可

[書式]

CL_RETURN LinSetMasterSendBox(int txBoxNo, LIN_TX_MSG txMsg, UInt32 time)

[引数]

変数	I/O	内容	値
txBoxNo	IN	送信 BOX No	0～19（0=送信 BOX1…19=送信 BOX20）
txMsg	IN	LIN 送信メッセージ情報	LIN_TX_MSG メンバーは下記参照
time	IN	定期送信の周期[ms]／スケジュール送信の時間[ms]	1～99999

LIN_TX_MSG メンバー

```
public class LIN_TX_MSG{
        public byte Id;
        public byte Dlc;
        public byte[] Data = new byte[8];
        public LIN_SUM_TYPE SumType;
}
```

変数	内容	
Id	LIN ID	0～0x3F
Dlc	データ長	0～8 ※0 を選択の場合、スレーブからのレスポンスを受信することが可能。
Data[0]～[8]	データ	0～0xFF
SumType	チェックサムタイプ	・LIN_SUM_TYPE.CLASSIC：標準チェックサム ・LIN_SUM_TYPE.ENHANCED：拡張チェックサム

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
PARAMETER_ERROR	引数が範囲外
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.21. LinStartMasterSimulation1：LIN ﾏｽﾀｰｼﾐｭﾚｰｼｮﾝ(定期送信)開始

[説明]

LinSetMasterSendBox（LIN マスターシミュレーション用送信 BOX 更新）関数実行後、本関数を実行することで LIN マスターシミュレーション（定期送信）を開始します。送信 BOX No1～20 の有効／無効を設定します。

※送信 BOX11～20 ファームウェアバージョン 1.1.0 以上で使用可

[書式]

CL_RETURN LinStartMasterSimulation1(bool[] txBoxNoEnable)

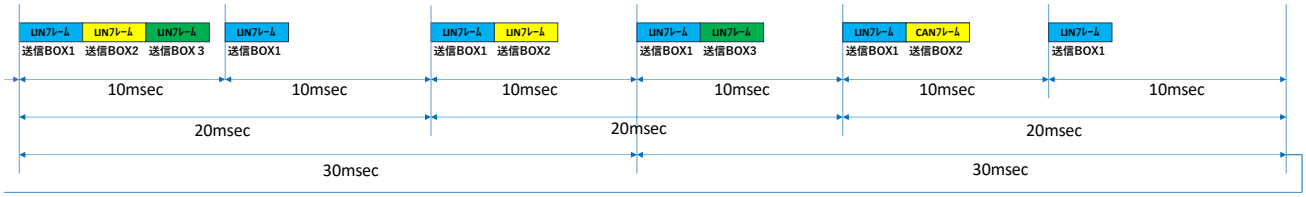
[引数]

変数	I/O	内容	値
txBoxNoEnable[0]～[19]	IN	txBoxNoEnable[0]：送信 BOX No1 有効／無効 txBoxNoEnable[1]：送信 BOX No2 有効／無効 ・・・ txBoxNoEnable[19]：送信 BOX No20 有効／無効	・false：無効 ・true：有効

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.LIN_NOT_STARTED	LIN 通信開始関数が実行されていない。
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

BOX1=10msec
BOX2=20msec
BOX3=30msec
定期送信は上記設定の場合、下記タイミングチャートのようにBOX1～BOX3まで一斉に送信が開始される。



5.22. LinStartMasterSimulation2：LIN ﾏｽﾀｰｼﾐｭﾚｰｼｮﾝ(ｽｹｼﾞｭｰﾙ送信)開始

[説明]

LinSetMasterSendBox（LIN マスターシミュレーション用送信 BOX 更新）関数実行後、本関数を実行することで LIN マスターシミュレーション（スケジュール送信）を開始します。送信 BOX No1～20 の有効／無効を設定します。繰り返し回数に 0 を指定することで CanStopSend 関数実行まで無限に繰り返します。1 以上を設定した場合、指定した回数まで繰り返します。繰り返しの完了は LinScheduleEnd プロパティで判別可能です

※本関数はファームウェアバージョン 1.1.0 以上で使用可

[書式]

CL_RETURN LinStartMasterSimulation2(bool[] txBoxNoEnable, UInt16 repeatCount)

[引数]

変数	I/O	内容	値
txBoxNoEnable[0] ～[19]	IN	txBoxNoEnable[0]：送信 BOX No1 有効／無効 txBoxNoEnable[1]：送信 BOX No2 有効／無効 ・・・ txBoxNoEnable[19]：送信 BOX No20 有効／無効	・false：無効 ・true：有効
repeatCount	IN	繰り返し回数	0=無限繰り返し 1～999 回

[戻り値]

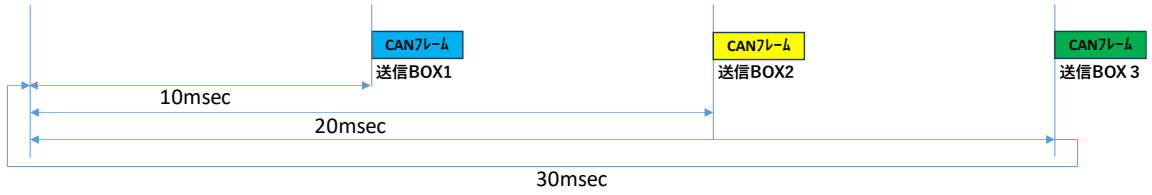
値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.LIN_NOT_STARTED	LIN 通信開始関数が実行されていない。
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

BOX1=10msec

BOX2=20msec

BOX3=30msec

スケジュール送信は上記設定の場合、下記タイミングチャートのようにBOX1から順番に送信されBOX3送信後、再度BOX1から送信される。



5.23. LinSetSlaveSendBox：LIN スレーブシミュレーション用送信 BOX 更新

[説明]

LIN スレーブシミュレーション（応答）用の送信 BOX を更新します。最大 20 種類のスレーブ応答が可能です。本関数実行後、LinStartSlaveSimulation（LIN スレーブシミュレーション開始）関数を実行することで、スレーブ応答が開始されます。スレーブシミュレーション開始後に実行した場合、スレーブ応答のデータが更新されます。

※送信 BOX11～20 ファームウェアバージョン 1.1.0 以上で使用可

[書式]

CL_RETURN LinSetSlaveSendBox(int txBoxNo, LIN_TX_MSG txMsg)

[引数]

変数	I/O	内容	値
txBoxNo	IN	送信 BOX No	0～19 最大 20 種類のメッセージを応答可能
txMsg	IN	LIN 送信メッセージ情報	LIN_TX_MSG メンバーは下記参照

LIN_TX_MSG メンバー

```
public class LIN_TX_MSG{
        public byte Id;
        public byte Dlc;
        public byte[] Data = new byte[8];
        public LIN_SUM_TYPE SumType;
}
```

変数	内容	
Id	LIN ID	0～0x3F
Dlc	データ長	1～8
Data[0] ～[8]	データ	0～0xFF
SumType	チェックサムタイプ	・LIN_SUM_TYPE.CLASSIC：標準チェックサム ・LIN_SUM_TYPE.ENHANCED：拡張チェックサム

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
PARAMETER_ERROR	引数が範囲外
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.24. LinStartSlaveSimulation：LIN スレーブシミュレーション開始

[説明]

LinSetSlaveSendBox（LIN スレーブシミュレーション用送信 BOX 更新）関数実行後、本関数を実行することで LIN スレーブシミュレーション（応答）を開始します。送信 BOX No1～20 の有効／無効を設定します。

※送信 BOX11～20 ファームウェアバージョン 1.1.0 以上で使用可

[書式]

CL_RETURN LinStartSlaveSimulation(bool[] txBoxNoEnable)

[引数]

変数	I/O	内容	値
txBoxNoEnable[0] ～[19]	IN	txBoxNoEnable[0]：送信 BOX No1 有効／無効 txBoxNoEnable[1]：送信 BOX No2 有効／無効 ・・・ txBoxNoEnable[19]：送信 BOX No20 有効／無効	• false：無効 • true：有効

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.LIN_NOT_STARTED	LIN 通信開始関数が実行されていない。
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.25. LinStopSimulation：LIN シミュレーション停止

[説明]

LIN シミュレーション(マスター／スレーブ)を停止します。

[書式]

CL_RETURN LinStopSimulation()

[引数]

なし

[戻り値]

値	内容
CL_RETURN.OK	正常
CL_RETURN.DEVICE_NOT_OPENED	オープン関数が実行されていない
CL_RETURN.DEVICE_WRITE_ERROR	CanLine-1 本体へのコマンド送信エラー
CL_RETURN.OTHER_ERROR	その他エラー ※OtherErrorMsg プロパティでエラーメッセージを確認してください。

5.26. LinGetMessage：LIN 通知メッセージ取得

[説明]

LIN 送信完了、受信発生の通知メッセージを取得します。

[書式]

int LinGetMessage(ref List<LIN_RX_MSG> lstRxMsg, int count)

[引数]

変数	I/O	内容	値
lstRxMsg	OUT	LIN 通知メッセージ情報リスト	LIN_RX_MSG メンバーは下記参照
count	IN	取得するメッセージ数	LinRxMsgCount プロパティで受信メッセージ数を取得して、それ以下を指定してください。

LIN_RX_MSG メンバー

```
public class LIN_RX_MSG{
        public UInt32 TimeStamp;
        public LIN_RX_MSG_TYPE Type;
        public byte Id;
        public byte Dlc;
        public byte[] Data = new byte[8];
        public LIN_SUM_TYPE SumType;
        public LIN_RX_MSG_STATUS Status;
}
```

変数	内容	値
TimeStamp	タイム スタンプ	0～4294967295 最小単位：0.1msec [例]123=12.3msec
Type	メッセージ タイプ	・LIN_RX_MSG_TYPE.RECIEVD：LIN 受信 ・LIN_RX_MSG_TYPE.SENT：LIN 送信完了
Id	LIN ID	0～0x3F
Dlc	データ長	0～8
Data[0]～[8]	データ	0～0xFF
SumType	チェックサム タイプ	・LIN_SUM_TYPE.CLASSIC：標準チェックサム ・LIN_SUM_TYPE.ENHANCED：拡張チェックサム

Status	ステータス	• LIN_RX_MSG_STATUS.SUCCESS：正常 • LIN_RX_MSG_STATUS.TIME_OUT：タイムアウトエラー • LIN_RX_MSG_STATUS.SUM_ERROR：チェックサムエラー • LIN_RX_MSG_STATUS.PARITY_ERROR：パリティーエラー

[戻り値]

値	内容
int	取得したメッセージ数