VECTOR >

はじめての LIN

ビギナー向け資料「はじめてシリーズ」

vector.com

目次

本稿はベクター・ジャパン執筆による原稿に基づき、@IT MONOist に連載された記事「車載ネットワーク『LIN』入門」より転載したものです。

@IT MONOist
http://monoist.atmarkit.co.jp

発行元：ベクター・ジャパン株式会社

1. 知っておきたい LIN の基礎知識 その 1

自動車のパワーウィンドウやミラー調整、電動シート、ドアロックなどのボディ制御に使われる通信プロトコル「LIN（Local Interconnect Network）」。本稿ではその基礎から、フレーム構造やネットワークマネジメントといった LIN の仕様までを詳しく解説します。

1.1 サブネットワークとしての LIN

1.1.1 LIN 策定の背景

近年、自動車に対する安全性や利便性の向上と環境規制への対応に伴い、電子制御部品が増加しています。この自動車の「電子制御化」は、エンジンなどを制御するパワートレイン制御やステアリングなどを制御するシャシー制御だけではなく、パワーウィンドウやミラー調整、電動シート、ドアロックなどのボディ制御に使われる「サブネットワーク」にも広がっています。

また一方で、こうしたサブネットワークの電子制御化の流れは、センサ、アクチュエータ、それらを制御する ECU（Electronic Control Unit）などの部品、そして、配線（ハーネス）の増加をもたらしています（図 1）。当然、電子制御化に伴う配線数の増加は、材料費や開発費、組み立て工数などのコストにはね返りますし、重量や配索スペース、接触不良などの電気的トラブルの増加など、自動車の品質と信頼性に大きな影響を及ぼすこともあり得ます。

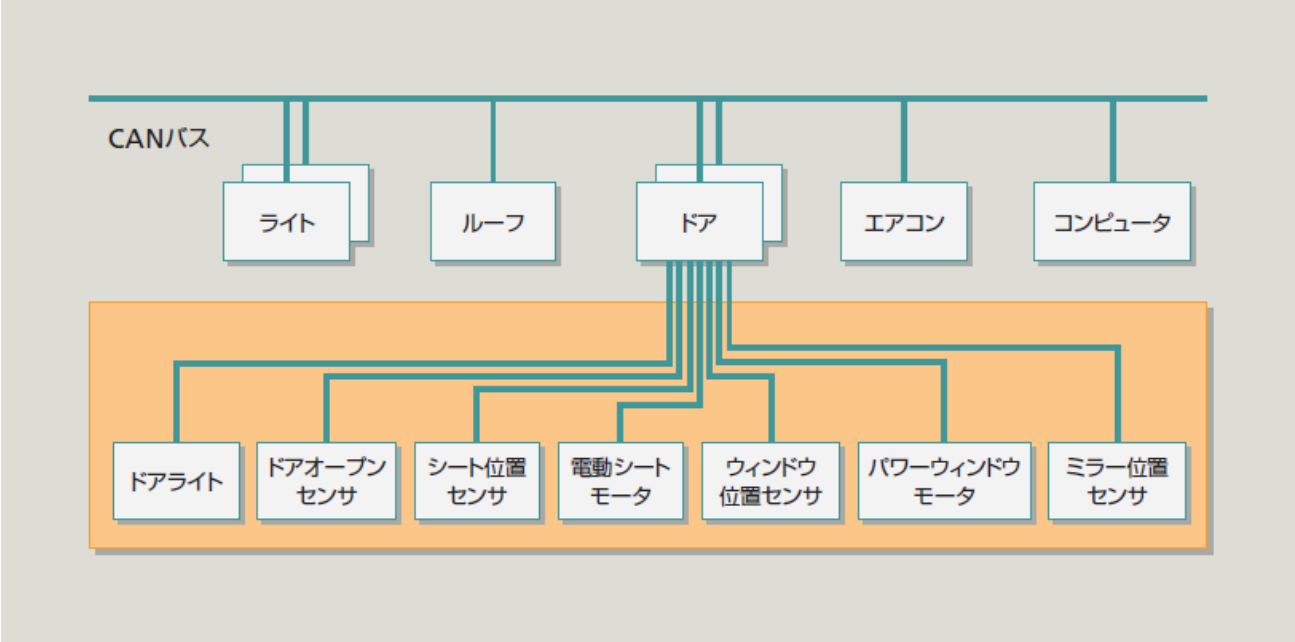


図 1: サブネットワークの電子制御化による部品・配線の増加

こうした課題の解決策の 1 つに「多重通信プロトコル」の導入があります。多重通信プロトコルを導入することで、配線数の増加を抑えながらも、センサとアクチュエータの増加に対応することが可能になります。自動車における多重通信プロトコルといえば、真っ先に「CAN（Controller Area Network）」を イメージする方が多いのではないでしょうか。CAN は、すでに多くの自動車に搭載されている実績があり、採用しやすい手段といえます。

しかし、センサやアクチュエータなどのサブネットワーク通信に、パワートレイン制御やシャシー制御に求められる通信速度や信頼性は必要ではなく、CAN を採用することはコスト面から見ても必ずしも最適な設計とはいえません。

そこで策定された通信プロトコルが、本稿の主役である「LIN（Local Interconnect Network）」です。パワートレイン制御やシャシー制御ほど通信速度、信頼性を必要としないセンサやアクチュエータなどの制御、すなわちボディ制御に採用され、シンプルかつ安価な車載向けサブネットワークシステムを構築できます（図 2）。

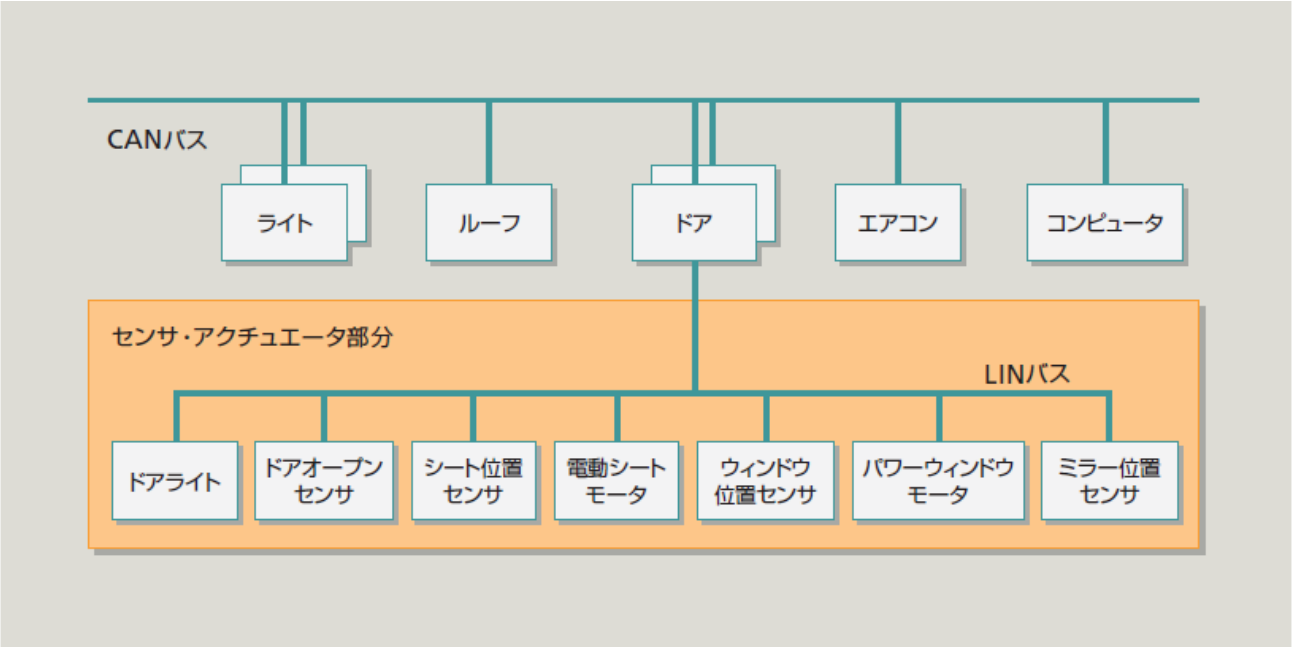


図 2：シンプルかつ安価な車載向けサブネットワークシステムを構築できる LIN

本稿では LIN の基礎知識から、フレーム構造、ネットワークマネジメントといった LIN の仕様までを詳しく解説していきます。また、新しいプロトコルバージョンである LIN2.0、2.1 で追加された仕様についても併せて紹介します。

「知っておきたい LIN の基礎知識　その 1」では、LIN の基礎知識として、LIN プロトコルの特徴やハードウェア、通信方式について説明します。

1.1.2 適用分野

LIN の適用分野は、ドアミラー、パワーシート、サンルーフ、ドアロック、エアコン、照明など、特に快適性の機能分野で使用されています（図 3）。

今後も、自動車は快適装備品の増加や品質向上、コスト削減といった要求に対応していく必要があります。また、「HV（Hybrid Vehicle）」「EV（Electric Vehicle）」などの開発では、従来とは異なる制御、通信も増加していくことが考えられます。これらの要求に対応するために、LIN が使われるケースはさらに増えると推測されます。

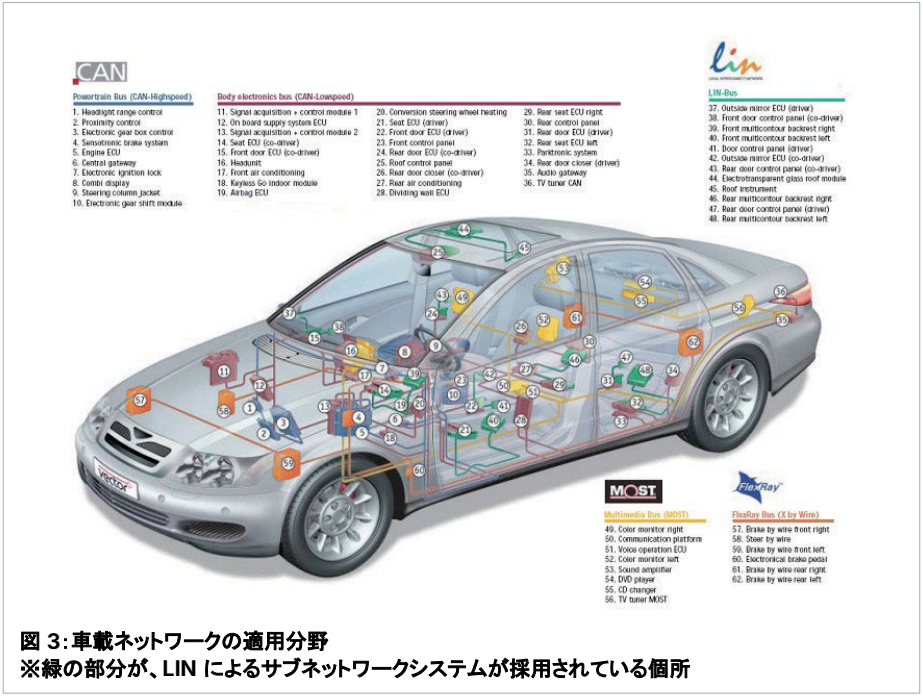


図 3：車載ネットワークの適用分野
※緑の部分が、LIN によるサブネットワークシステムが採用されている個所

1.1.3 LIN コンソーシアムとは?

LIN プロトコルを策定した「LIN コンソーシアム」は、センサ／アクチュエータにおける経済的かつ規格化された通信仕様を作成するために、2000 年に設立された欧州の標準化団体です。LIN コンソーシアムには自動車メーカー、半導体ベンダ、ツールベンダなどが参加しています。

LIN プロトコルは、1999 年に最初のバージョンである LIN1.0 が公開されました。ちなみに、2010 年 7 月時点の最新バージョンは 2006 年に発表された LIN2.1 となります（図 4）。

LIN 仕様書は、LIN コンソーシアムの Web サイトから無料でダウンロードできます。

LIN コンソーシアムでは、LIN プロトコルに適合しているかどうかを確認するためのテスト仕様「コンフォーマンステスト仕様」も策定しています。コンフォーマンステスト仕様書は、LIN コンソーシアムのメンバーのみが入手でき、自動車メーカーによっては、LIN コンフォーマンステストの認証試験の実施を義務付けているところもあります。

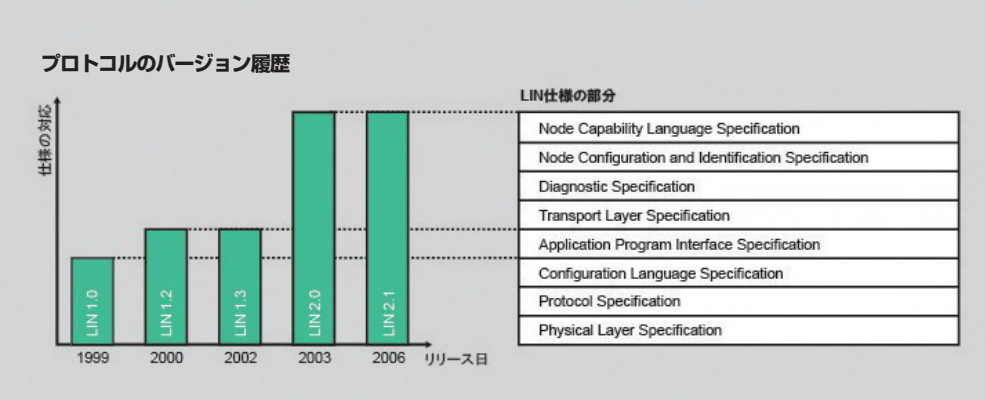


図 4：LIN プロトコルと OSI 参照モデルの比較

1.1.4 OSI 参照モデルに基づいた LIN ノード構造

LIN プロトコルを OSI 参照モデルと比較すると、図 5 のようになります。

LIN プロトコルでは物理層、データリンク層だけでなくアプリケーションと LIN ネットワークとのインターフェイス（API）も規定されています。また、バージョン 2.0 以降ではトランスポートプロトコル（TP）および診断も規定されています。

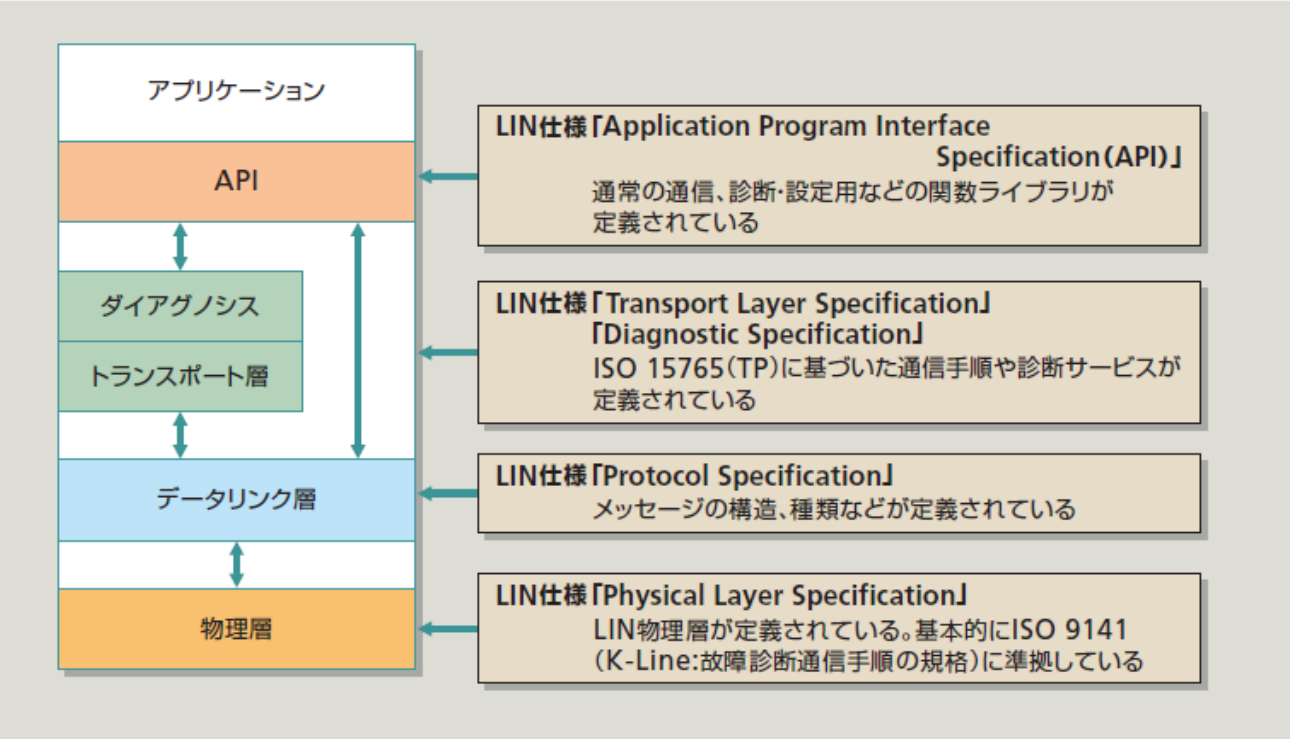


図 5：LIN プロトコルのバージョン履歴

1.1.5 LIN プロトコルの特徴

LIN プロトコルの主な特徴は、以下のとおりです。

- ✓ ライン型バス構造
- ✓ マスター・スレーブ方式（1 つのマスターと複数のスレーブ）
- ✓ トークン方式
- ✓ メッセージアドレッシング方式（すべてのノードが LIN メッセージを受信可能）
- ✓ スケジュールに基づく通信（タイムトリガー方式）
- ✓ データ転送速度 最大 20kbit/sec
- ✓ UART インターフェイス（多くのマイコンに実装されているシリアル通信装置）
- ✓ 水晶発振子、セラミックレゾネータを必要としない自己同期（スレーブノード）
- ✓ 短いメッセージ（最大 8 バイト）
- ✓ 簡単な送信データ保護（パリティ、チェックサム）

以降、上記の特徴について詳しく説明していきます。

1.2 LIN ハードウェア

LIN ノードは、マイクロコントローラ（マイコン）と LIN トランシーバによって構成されています。LIN は、簡単・安価にセンサとアクチュエータとを接続するために、多くのマイコンが搭載しているシリアル通信装置

「UART（Universal Asynchronous Receiver / Transmitter）」を使用して送受信を行います（図 6）。

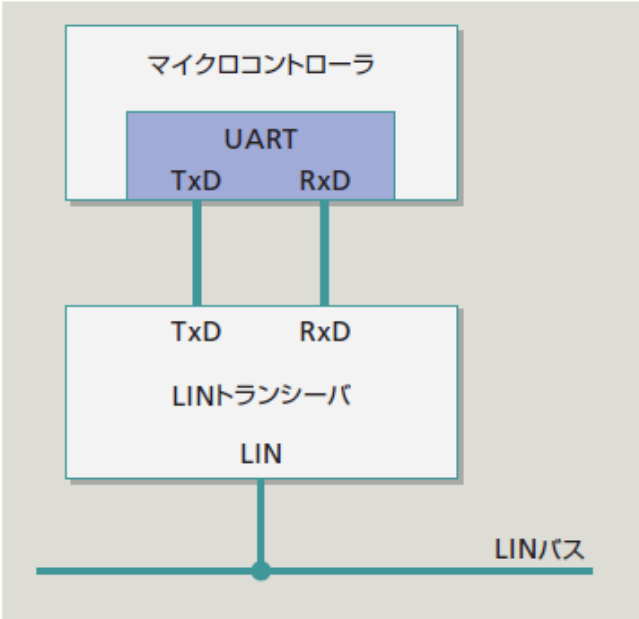


図 6：LIN ノードの構成
※マイコンは、LIN トランシーバを介して LIN バスと接続

UART の通信方式を LIN で使用する場合、8 ビットのデータの前後に「スタートビット」と「ストップビット」を付与した 10 ビット単位で送信します。また、8 ビットのデータは、最下位ビット（LSB）から送信します。スタートビットとストップビットは、データの開始と終了を判断するために使われます。スタートビットは 論理値「0」となり、電圧レベルはグランドです。ストップビットは論理値「1」となり、電圧レベルはバッテリです。論理値「1」を「リセッシブ」、論理値「0」を「ドミナント」と呼びます（図 7）。

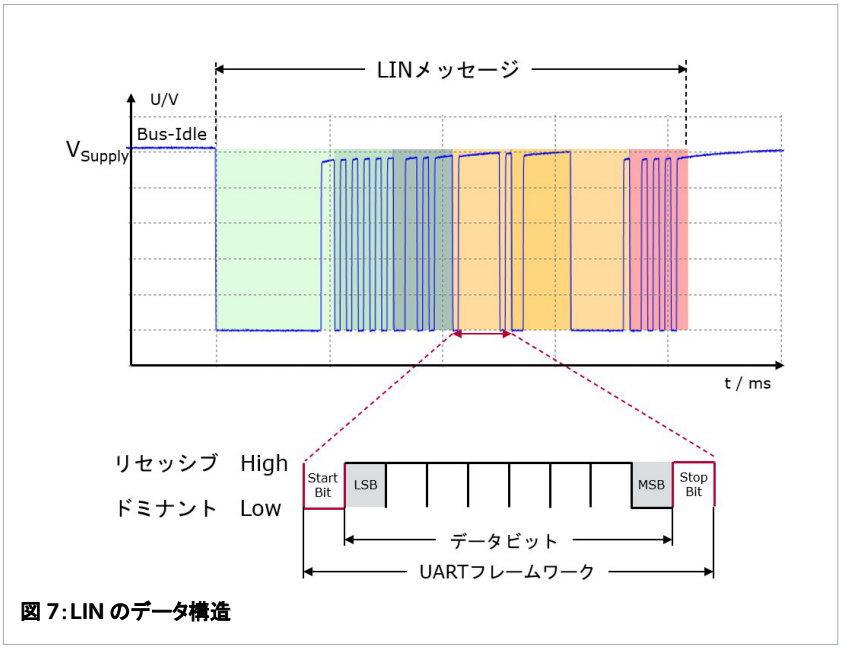


図 7：LIN のデータ構造

LIN トランシーバは、基本的に「ISO 9141」に準拠したシングルワイヤを使用します。LIN トランシーバでは入力電圧、信号の振幅、LIN ノードの省電力のためのスリープ・ウェイクアップ機能を制御します。最近では、LIN プロトコル用にカスタマイズされた「LIN UART」なども増えています。

最後に通信速度ですが、LIN では EMC 対策やクロック同期のために、最大 20kbit/sec と規定されています。一般的に 2.4kbit/sec、9.6kbit/sec、19.2kbit/sec がよく使われています。

このように、CAN と比較すると通信コントローラを必要とせず、マイコンに搭載されている UART で通信できるので、ハードウェアのコストを抑えることができます。また、LIN バスはシングルワイヤを使用するため、配線数を抑え、コストを削減することができます。

次の項目では、LIN の通信方式について説明します。

1.3 通信方式

1.3.1 概要

LIN の通信は前述のとおり、「マスター・スレーブ方式」「スケジュールに基づく通信（タイムトリガー方式）」で行われます。これは CAN とは異なる通信方式で、LIN の大きな特徴といえます。

ここでは「電車」の例を挙げ、LIN の通信方式の特徴を簡単に説明します（図 8）。

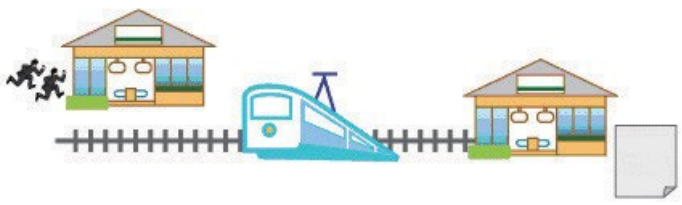


図 8：電車を例に LIN の通信方式を説明

電車は、始発駅から時刻表どおりに出発します。この時刻表には、電車が衝突しないように出発時刻が決められています。また、駅では乗客が電車を待ち、乗車対象の電車が到着すると乗車し、目的の駅まで移動します。LIN の通信も、この例と同様の動作をします。LIN では、事前に定義された「送信タイミング」に従って送 信を行います。そのため、メッセージの衝突は発生せず、各ノードは一定の間隔で確実にメッセージの送信および受信ができます。また、バスも過負荷にならないので安定した通信ができるといえます。

しかし、送信するタイミングが決められているということは、各ノードは任意のタイミングでメッセージを送信できません。送信するタイミングが来るまで待つ必要があります。

また、ネットワーク内に「送信するタイミングを制御する」という特別な役割を実行するノードが必要となるため、LIN では「マスターノード」と「スレーブノード」といった 2 種類のノードを使用して通信を行います。ここまでの説明を簡単にまとめますと、CAN では、各ノードは任意のタイミングで送信できますが、メッ セージの衝突の調停によっては必ずしも周期性が保証できないなど、あらかじめ期待されたタイミングで確実にメッセージのやりとり（通信）ができないことがあります。しかし、LIN では送信するタイミングをあらかじめ定義することにより、期待されたタイミングで確実にメッセージの通信が行えます。

1.3.2 ライン型バス構造

LIN のネットワーク構造（トポロジ）は、「ライン型バス構造」です。1 つの LIN バスラインに 1 つのマスターノードと複数のスレーブノードが接続できます。また、LIN ネットワークの推奨最大ノード数は 16 ノード、最大配線長は 40 メートルです（図 9）。

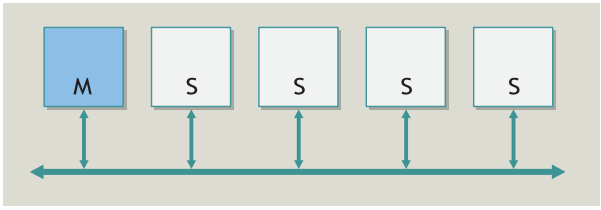


図 9：LIN のネットワーク構造

1.3.3 マスター・スレーブ方式

LIN ネットワークは、マスターノードが全体の通信を制御するマスター・スレーブ方式を採用しています（図 10）。

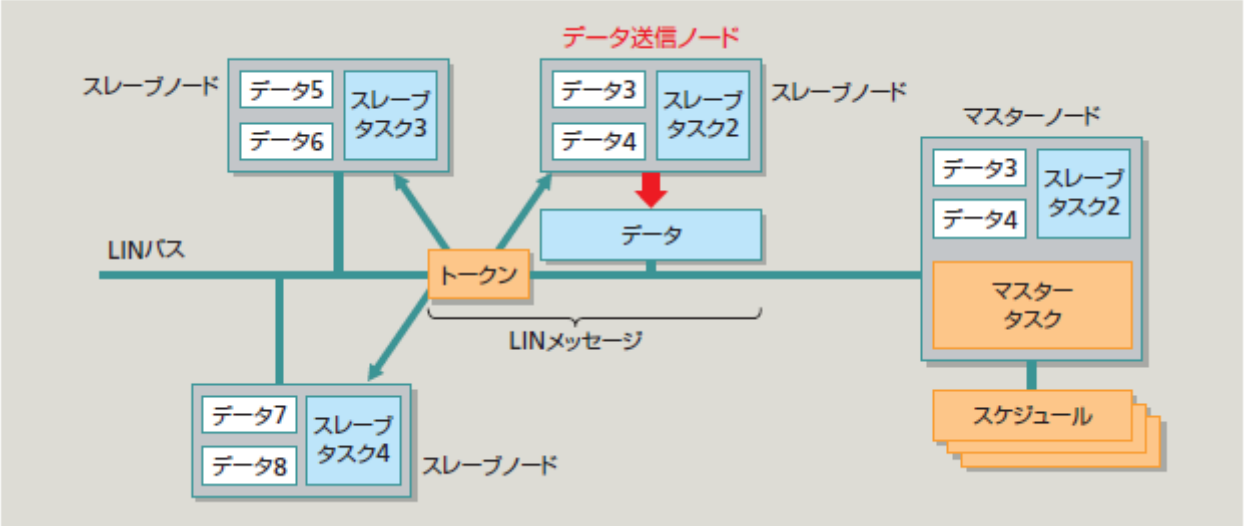


図 10：マスター・スレーブ方式
※各ノードは、マスターノードから送信されるトークンを監視し、データ送信ノードがトークンの後にデータを送信。また、マスターノードはスレーブタスクを持っているため、データの送信もできる

スレーブノードは、マスターノードに従って通信を行います。

LIN ノードの役割には、「マスタータスク」と「スレーブタスク」の 2 種類があります。

✓ **マスタータスク**

マスターノードのみが持っている役割で、トークンの送信とスケジュールの管理を行います。これは、決められたタイミングで LIN バスに送信要求「トークン」を送信します。

✓ **スレーブタスク**

マスターノードとスレーブノードの両方が持っている役割で「データ」を送信します。データの送信は、マスタータスクから送信されるトークンをスレーブタスクが監視し、データを送信するノードがトークンの後にデータを送信します。つまり、マスターノードからトークンが送信されない限り、各ノードはデータを送信できません。LIN では、このトークンとデータで 1 つの「メッセージ」を構成しています。LIN ではトークンを「ヘッダー」、データを「レスポンス」、メッセージを「フレーム」と呼んでいます。ヘッダーには、フレームの意味を表す「ID」と呼ばれる情報を持っており、各ノードはこの ID を監視することでレスポンスを送信するかどうかを識別します。また、LIN フレームは「メッセージアドレッシング方式」で送信されるため、1 つ、複数、すべての LIN ノードが LIN フレームを受信できます。

1.3.4 スケジューリング

マスタータスクは、ヘッダーを送信するタイミングを「LIN スケジュール」で定義します（図 11）。LIN スケジュールには送信する ID や送信の順番、送信する時間間隔が定義されています。

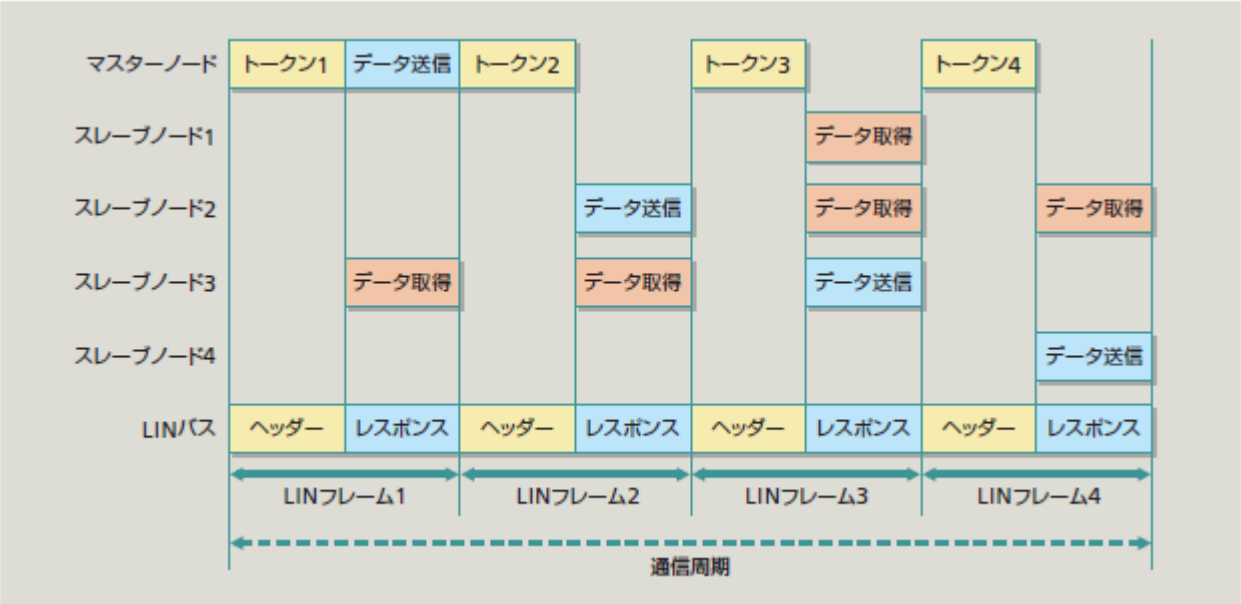


図 11:LIN スケジュール
※LIN フレーム 1 では、マスターノードがデータを送信し、スレーブノード 3 がデータを取得。LIN フレーム 3 では、スレーブノード 3 がデータを送信し、スレーブノード 1、2 がデータを受信。LIN スケジュールの最後のフレーム（LIN フレーム 4）を送信すると、最初（LIN フレーム 1）の送信に戻る

つまり、マスターノードはこの LIN スケジュールを使用して、LIN ネットワーク全体の通信の制御を行います。そのため、LIN スケジュールは LIN バス上で通信の衝突が発生しないように送信タイミングを定義する必要があります。

また、LIN スケジュールは複数のスケジュールテーブルの定義ができます。つまり、マスターノードは「起動時の初期化モード」「通常モード」「診断モード」など、車両状態の変化に応じてスケジュールの変更を行い、送信 ID や送信周期の変更ができます。

1.3.5 同期方法

LIN ノードはコスト削減のため、各信号の通信速度を調整する専用の配線などを使用していません。また、LIN スレーブノードのクロック回路もコストを抑えるために、誤差が大きい CR 発振やリングオシレーター発振などの使用が許されています。つまり、そのままでは、LIN ノード間の内部クロックに誤差が生じる可能性があります。そこで、LIN ではこの"クロックの誤差"を補正するために、マスターノードから送信されるヘッダーを使用します。 LIN では、マスターノードは水晶発振子などの高精度な受動素子の使用が規定されており、その許容誤 差は±0.5%です。これにより、マスターノードは正確なクロックの作成ができます。この高性能な内部クロックを持つマスターノードから送信されるヘッダーに、クロック誤差を補正するための「同期信号」を入れることにより、各ノード間のクロック誤差を補正します。なお、LIN フレーム（ヘッダー）の構造については「、知っておきたい LIN の基礎知識 その 2 」で説明します。

ちなみに、LIN ではスレーブノードのクロックの許容誤差は±14%（LIN1.3 では、±15%）と規定されています。ただし、高精度のクロック回路を使用した場合は、許容誤差±1.5%と規定されています。

ここまでは、LIN プロトコルの策定の背景やハードウェア、通信方式について説明しました。CAN と比較すると簡易的であり、さまざまな部分でコスト削減が図られていることがお分かりいただけたかと思います。

次に、「知っておきたい LIN の基礎知識 その 2」として、LIN フレームの構造やネットワークマネジメントなどについて説明します。

2. 知っておきたい LIN の基礎知識　その 2

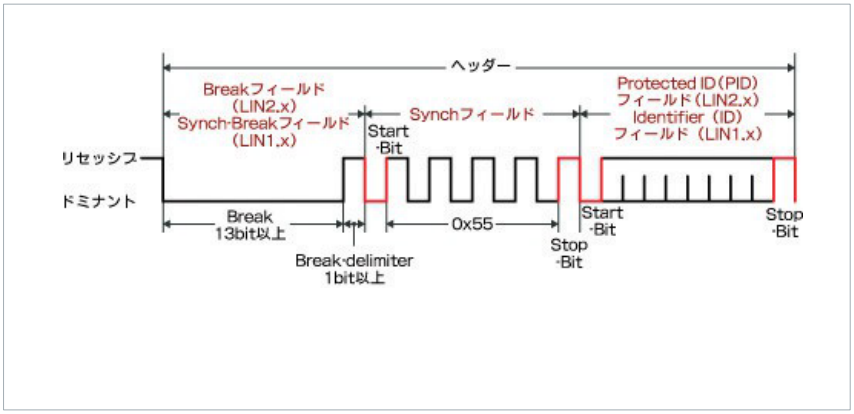
「知っておきたい LIN の基礎知識 その 1」では、LIN の基礎知識として LIN プロトコルの特徴やハードウェア、通信方式について解説しました。次は、「LIN フレームの構造」「ネットワークマネジメント」「LIN 記述ファイル（LDF:LIN Description File）」について説明します。

2.1 フレーム構造

前述で説明したとおり、LIN は「マスター・スレーブ方式」で通信を行います。1 つの LIN フレームは「ヘッダー」と「レスポンス」で構成されており、ヘッダーは「マスタータスク（マスターノード）」から送信され、レスポンスは「スレーブタスク（マスターノードまたはスレーブノード）」から送信されます。ここでは、ヘッダーとレスポンスのフレーム構成を説明します。

2.1.1 ヘッダー

ヘッダーは「Break」「Synch」「Protected ID（PID）」の 3 つのフィールドで構成されています（図 12）。



✓ **Break フィールド**

Break フィールドは、ほかのフィールドと異なり、意図的にフレーミングエラー（スタートビットから数えて 10 ビット目にストップビットが検出されない場合に発生）を起こすことにより、すべてのスレーブノードに LIN フレームの開始を通知します。Break フィールドは Break と Break-delimiter で構成され、Break は 13 ビット以上のドミナント、Break-delimiter は 1 ビット以上のリセッシブとなります。Break- delimiter は、Break の終わりを表します。

✓ **Synch フィールド**

「知っておきたい LIN の基礎知識その 1」の同期方法で説明した「同期信号」が Synch フィールドです。Synch フィールドは、各ノード間のクロック誤差の補正に使用し、0x55 UART フレーム（スタートビット／ストップビットあり）を送信します。スレーブノードは、立ち下がりエッジの最初と最後の時間差を 8 で割ることによって算出された 1 ビットの時間間隔（TBit）を基に、各ノードのクロック誤差を調整します（図 13）。ただし、高精度のクロック回路（許容誤差±1.5%）を使用した場合は、クロック誤差の調整は必要ありません。

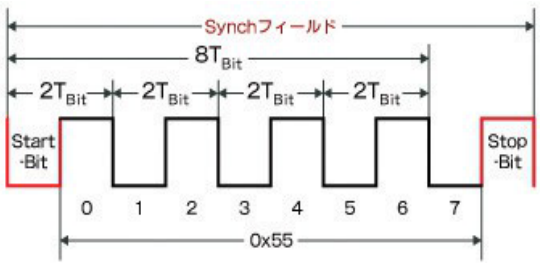


図 13：Synch フィールド
※Synch フィールドにより、各ノードの通信速度をマスターノードの通信速度に合わす。これにより各ノードのクロック誤差が調整される

✓ Protected ID（PID）フィールド

PID フィールドは、LIN フレームの識別情報を表す 6 ビット（0~5 ビット）のフレーム ID と、2 ビット（6、7 ビット）のパリティの合計 8 ビットで構成されています（図 14）。

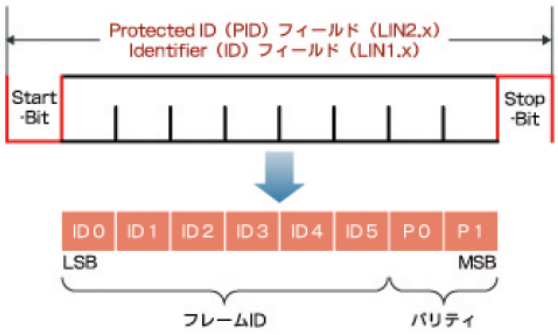


図 14：Protected ID フィールド

●フレーム ID の範囲は、6 ビットで表現できる 0~63（0x3F）です。ただし、フレーム ID60（0x3C）、61（0x3D）は診断フレームとして使用するなど、LIN 仕様上、すでに役割が定義されているフレーム ID もあります。

●パリティには、フレーム ID から演算式によって求められた値が格納されます。ヘッダーを受信したノードは、ヘッダー内のパリティビットの値とフレーム ID から演算した値を比較し、正しいフレーム ID を受信したかどうかを確認します。

※パリティビット（P0、P1）の演算式

$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4$

$P1 = \neg (ID1 \oplus ID3 \oplus ID4 \oplus ID5)$

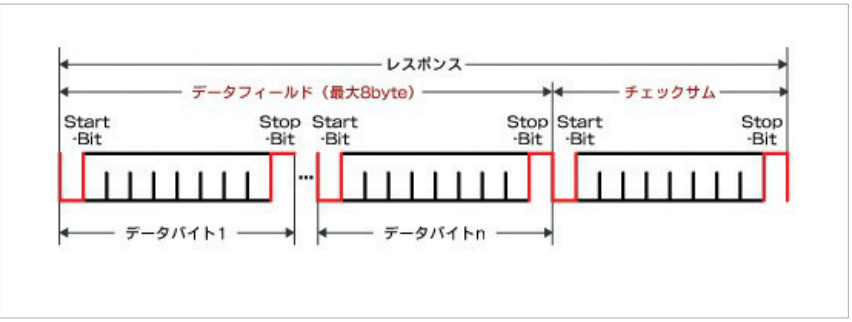
⊕：XOR
¬：INV（反転）

例
ID = 0x22 = 10 0010b
P0 = 0⊕1⊕0⊕0 = 1
P1 = ¬(1⊕0⊕0⊕1) = 1
PID = 1110 0010b = 0xE2

LIN1.x では、PID フィールドを「Identifier（ID）」フィールドと呼び、5、6 ビット目（図 14 の ID4、ID5）にレスポンスのデータ長（DLC:Data Length Code）の情報を格納する場合があります。

2.1.2 レスポンス

レスポンスは「データ」「チェックサム」の 2 つのフィールドで構成されています。データ、チェックサムともに、UART フレーム（スタートビット／ストップビットあり）で送信します（図 15）。



✓ **データフィールド**

データフィールドには、最大 8 バイトのデータが格納されます。

✓ **チェックサム**

チェックサムは、データを正確に受信できたかどうかを確認するために使用します。チェックサムには、各データ値の総和を反転した値が格納されます。ただし、総和の結果が桁あふれとなった場合、桁上がり値を演算結果に加算する必要があります（モジューロ 256 方式）。チェックサムには、「標準チェックサム（Classic Checksum）」「拡張チェックサム（Enhanced Checksum）」の 2 種類があります。

- **標準チェックサム**
  - → 演算対象は、すべてのデータバイト
  - → LIN1 のすべてのフレーム ID に使用
  - → LIN2.x では、診断フレーム（フレーム ID 60~61）のみ使用

標準チェックサムの例

Data0	A0	8ビット値	58	標準
Data1 +	B8	桁上がり値を加算 +	01	チェックサム = A6
Sum1	182	Sum2	59	
		INV	A6	

- 拡張チェックサム
  - → 演算対象は、PID およびすべてのデータバイト
  - → LIN2.x のフレーム ID 0~59 に使用

拡張チェックサムの例

PID	E2	8ビット値	82	8ビット値	3B	拡張
Data0	+ A0	桁上がり値を加算	+ 01	桁上がり値を加算	+ 01	チェックサム = C3
Sum1	182	Result1	83	Result2	3C	
		Data1	+ B8	INV	C3	
		Sum2	13B			

このように、LIN フレームの先頭部分（Synch フィールド）でクロック誤差の調整をすることで、各スレーブノードはマスターノードからフレーム ID を受信し、データの送受信を行うことができます。また、フレーム ID やデータが正しく送受信できたかどうかを各ノードが確認するために、LIN フレームにはパリティ、チェックサムが含まれています。

2.1.3 時間規定

LIN フレームは、LIN スケジュールによって各フレームの送信タイミングが重複しないように定義されます。その簡易的なハードウェア構成により、LIN フレームの送信時間に「レスポンススペース」や「インターバイトスペース」と呼ばれる時間が含まれることを許容しており、これらの時間誤差を考慮してスケジュールを設計する必要があります（図 16）。

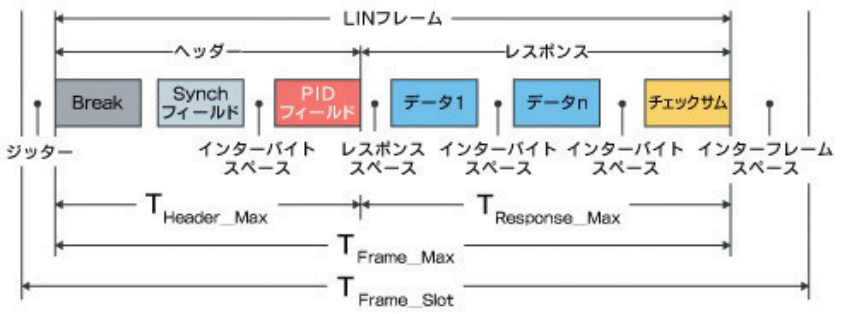


図 16：時間規定

LIN フレームの送信に必要な時間は、以下の式で求めることができます。「公称タイミング」とはレスポンススペース、インターバイトスペースを含めず、Break を 13 ビット（最小値）、Break-delimiter を 1 ビット（最小値）として求めたビット時間です。

〈公称タイミング〉

・$T_{Header\_Nominal} = 34T_{Bit}$

Break$(13T_{Bit})$ + Break-delimiter$(1T_{Bit})$ + Synch$(10T_{Bit})$ + PID$(10T_{Bit})$

・$T_{Response\_Nominal} = (N_{Data} + 1) \times 10T_{Bit}$

Data$(10T_{Bit})$ × データバイト数(N_{Data}) + チェックサム$(10T_{Bit})$

・$T_{Frame\_Nominal} = T_{Header\_Nominal} + T_{Response\_Nominal}$

レスポンススペース、インターバイトスペースなどの許容時間は、LIN フレームごとに最大 40%です。そのため、LIN フレームの最大時間は、以下の式で求めることができます。

〈最大タイミング〉

- $T_{Header\_Max} = 1.4 \times T_{Header\_Nominal}$
- $T_{Response\_Max} = 1.4 \times T_{Response\_Nominal}$
- $T_{Frame\_Max} = T_{Header\_Max} + T_{Response\_Max}$

LIN スケジュールの時間単位は、「タイムベース（Time_Base）」と呼ばれ、LIN スケジュールを処理するための最小時間単位となります。例えば、タイムベースが 5ms となっている場合、LIN フレームの送信間隔は 5ms 単位で設定ができます。一般的にタイムベースは、5ms または 10ms がよく使われています。

また、1 つの LIN フレームに必要なタイムベースの合計を「フレームスロット（Frame-Slot）」と呼びます。

2.1.4 フレームタイプ

LIN のフレームタイプは、「アンコンディショナル（Unconditional）」「イベントトリガー（Event Triggered）」「スポラディック（Sporadic）」「診断（Diagnostic）」の 4 種類となります。

> アンコンディショナルフレーム（フレーム ID 0~59）

通常使われる LIN のフレームタイプです。LIN ノード間のデータ通信に使われます（図 17）。

図 17：アンコンディショナルフレーム

- **イベントトリガーフレーム（フレーム ID 0~59）**※LIN2.0 で追加

  発生頻度の少ないイベントに使用するスレーブノード用のフレームで、1 つのヘッダーに対して複数のレスポンスを定義できます。

  ヘッダーは LIN スケジュールのとおりに送信されますが、レスポンスは必要なときにだけ送信されます。送信したノードが判断できるように、データバイトの 1 バイト目は PID が格納されます（図 18）。

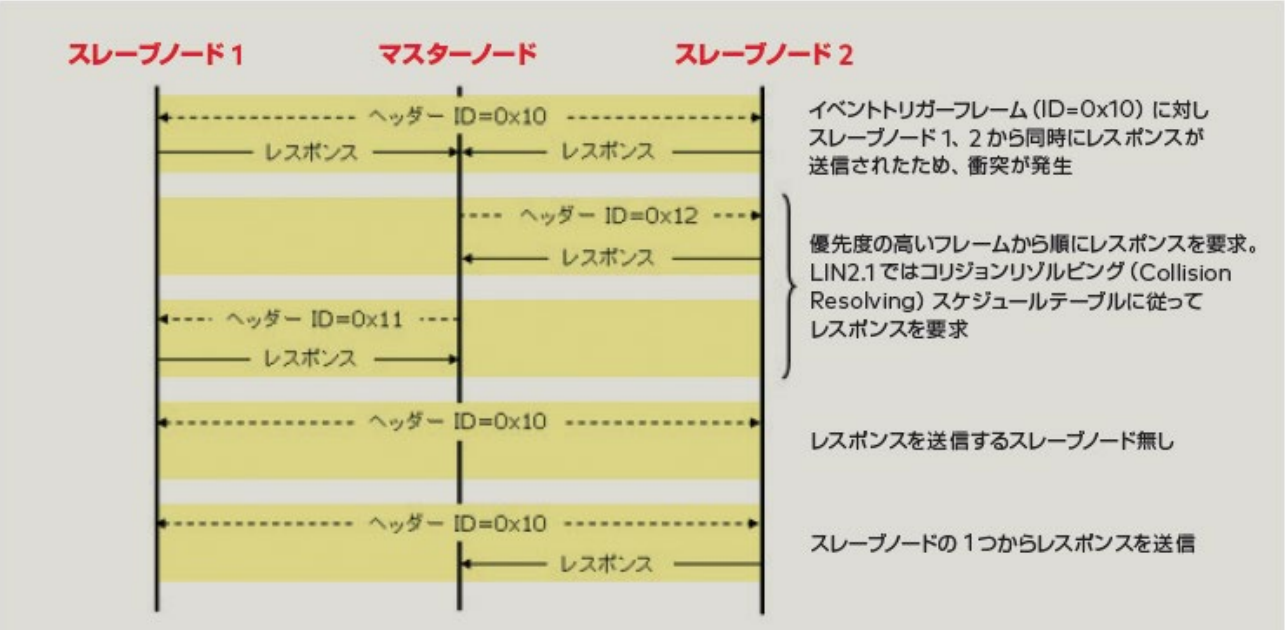


図 18：イベントトリガーフレーム

イベントトリガーフレームのフレーム ID は、関連するスレーブノードで共有しているため、同時に複数のスレーブノードがレスポンスを送信し、衝突する可能性があります。衝突が発生した場合、衝突によって失われたレスポンスデータを再送信させるために、マスターノードはアンコンディショナルフレームを使用し、スレーブノードごとに再送信要求を行います。LIN2.1 では、イベントトリガーフレームの衝突発生時に使用する「コリジョンリゾルビング（Collision Resolving）スケジュールテーブル」が追加されています。イベントトリガーフレームの使用例として、4 ドア・セントラル・ロッキング・システムでのドアロックの監視があります。イベントトリガーフレームを使用した場合、1 つのフレームスロットで 4 つのドアロックを監視できます。フレームスロットの減少は帯域の有効活用につながり、より効率的な通信を設定できます。しかし、同時に複数のドアロックが操作された場合は衝突が発生するので、再送信が必要です。もし、ドアロックの監視にアンコンディショナルフレームを使用した場合、ほとんど変化しないドアロック信号に 4 つのフレームスロットを使用するので、帯域無駄となります。このように、発生頻度の少ないイベントに対し、1 つのフレームスロットで通信することで、帯域の有効活用ができます。

✓ **スポラディックフレーム（ID 0~59）**※LIN2.0 で追加

特定のシグナルが更新された場合などに使用されるマスターノード用のフレームです。1 つのフレームスロットに複数のスポラディックフレームを定義できます。ただし、同時に複数のシグナルが更新された場合は、優先順位の高いフレームを最初に送信します。シグナル更新がない場合、マスターノードはスポラディックフレームのレスポンスだけでなくヘッダーも送信しません（図 19）。

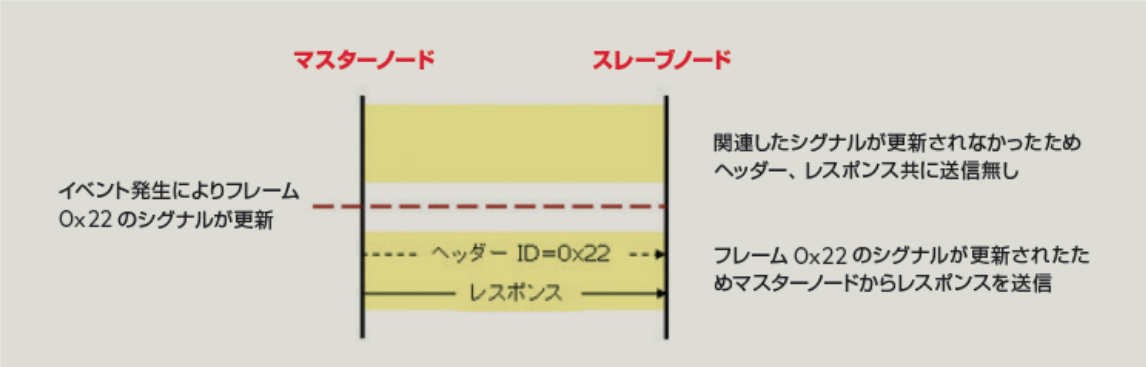


図 19：スポラディックフレーム

✓ **診断フレーム（フレーム ID 60~61）**

LIN ネットワークの診断、ノードコンフィグレーション、スリープモードコマンドに使用するフレームです。フレーム ID 60（0x3C）は、「マスターリクエスト」フレームと呼ばれ、マスターノードがレスポンスを送信します。フレーム ID 61（0x3D）は、「スレーブレスポンス」フレームと呼ばれ、スレーブノードがレスポンスを送信します（図 20）。

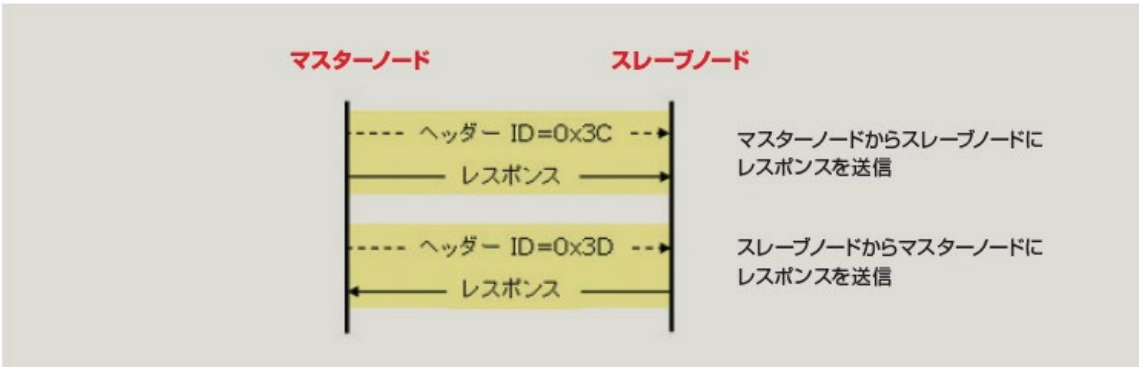


図 20：診断フレーム

✓ **予約フレーム（フレーム ID 62~63）**

フレーム ID 62（0x3E）と 63（0x3F）は予約フレームとなっているため、LIN2.x では使用できません。

2.1.5 エラー処理

LIN フレームでエラーが検出された場合は、マスタータスク、スレーブタスクによりデータを破棄します。LIN プロトコルでは、エラー処理の定義がないのでアプリケーションで定義する必要があります。LIN1.3 の仕様書では、下記のエラーが定義されています。

- Bit Error
- Checksum Error
- Identifier Parity Error
- Slave Not Responding Error
- Inconsistent Synch Field Error
- Physical Bus Error

LIN2.x では上記に加え、エラーを検知したスレーブノードがマスターノードに通知する「ステータスマネジメント」が追加されました。ステータスマネジメントについては、後ほど「LIN2.0/LIN2.1 の新仕様」で説明する予定です。

2.2 ネットワークマネジメント

2.2.1 ウェイクアップシグナルフレーム／スリープモードフレーム

「知っておきたい LIN の基礎知識 その 1」の適用分野で説明したとおり、LIN は主に快適装備品などに使用されるため、車両の状態によって通信を必要としない場合があります。例えば、キーレスエントリーでドアを施錠した場合、リモコンからのドアロック要求を受光するセンサ ECU は、次の開錠命令を受けるまで LIN 通信は不要です。

この“通信が不要な状態”となったときに「省電力モード（ウェイクアップ／スリープモード）」に切り替えることで、LIN ノードの消費電力を抑えることができます。

この省電力モードの切り替えを行うのが、ネットワークマネジメントです（図 21）。

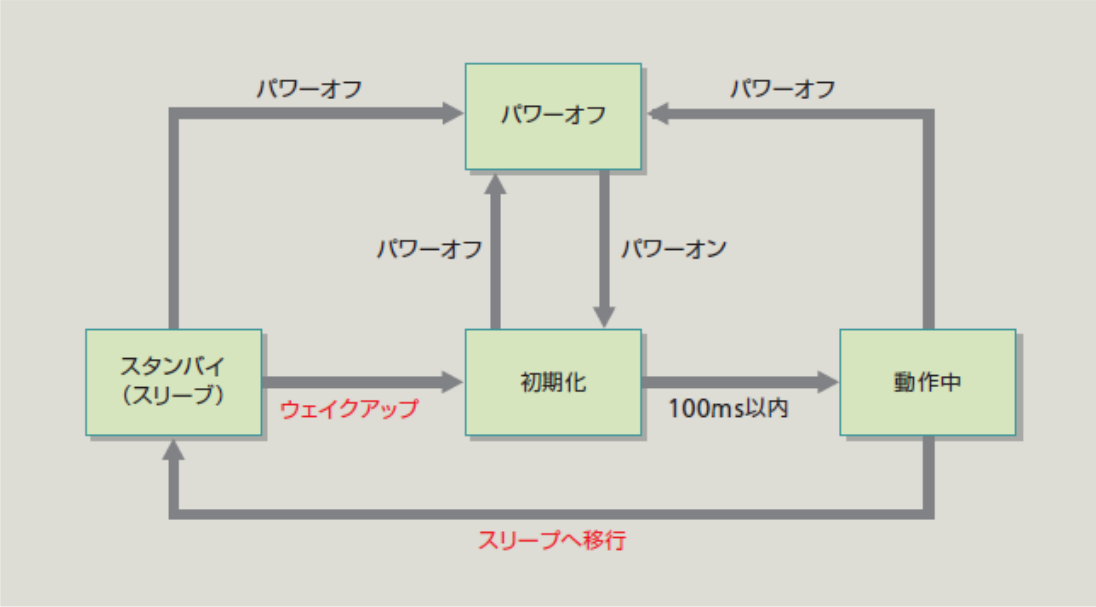


図 21：ネットワークマネジメント
※「動作中」にバスアイドル時間が 4~10 秒（LIN1.x では 2 万 5000 ビット）、または「スリープモード」フレーム送受信で「スタンバイ（スリープ）」状態に移行。スリープ中に「ウェイクアップシグナル」フレーム送受信で「初期化」状態に移行。「初期化」状態は 100ms 以内に動作中に移行

✓ **スリープモードフレーム（Go-to-Sleep コマンド）**

スリープモードフレームとは、マスタータスク（マスターノード）がスレーブノードにスリープモードへの移行を要求するコマンドです。マスターノードは、マスターリクエストフレーム（フレーム ID 60）のデータバイトの 1 バイト目に 0x00、2~8 バイトに 0xFF を格納して送信します（LIN1.x では、2~8 バイトに 0xFF を格納することは規定されていません）。このフレームを検知したスレーブノードは、通信を停止（スリープモードに移行）します（図 22）。

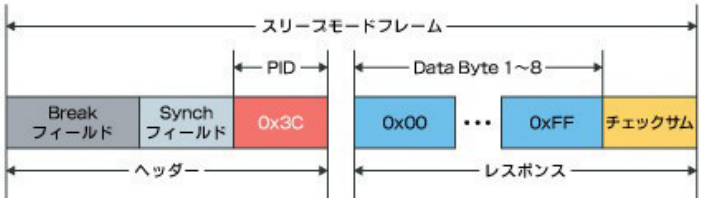


図 22：スリープモードフレーム
※スリープモードフレームはマスタータスク（マスタノード）のみ送信可能

✓ ウェイクアップシグナルフレーム

ウェイクアップシグナルフレームとは、スレーブタスク（マスターノードまたはスレーブノード）が各ノードにウェイクアップモード（初期化／動作中）への移行を要求するシグナルです。ウェイクアップシグナルは、ショートドミナントバスレベルです。ドミナント状態は、0.25~5ms で 0xF0 バイト（LIN1.x では、0x80 バイト）となります（図 23）。マスターノードの場合は、Break フィールドの送信でもウェイクアップモードへの移行を要求できます。

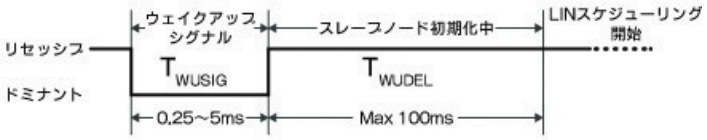


図 23：ウェイクアップシグナルフレーム
※ウェイクアップシグナルフレームは、スレーブタスク（マスターノードまたはスレーブノード）から送信可能

2.2.2 ウェイクアップシグナルのタイミング

ウェイクアップシグナルを送信してもスケジューリングが開始されない（Break が検出されない）場合、スレーブタスクはウェイクアップ要求を繰り返すことができます。ウェイクアップシグナルの間隔（リセッシブ状態）は、150~250ms（LIN1.x では、最大 128 ビット）です。ウェイクアップシグナルが合計 3 回送信されてもスケジューリングが開始されない場合、1.5 秒（LIN1.x では、1 万 5000 ビット）以上の休止状態の後に、4 回目のウェイクアップシグナル送信が可能になります（図 24）。

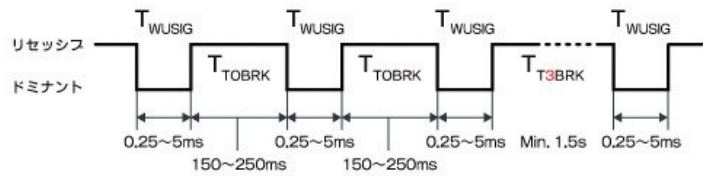


図 24：ウェイクアップシグナルのタイミング
※ 3 回目のウェイクアップ送信後もスリープ状態が継続している場合、その後の処理はアプリケーションに依存

ネットワークマネジメントのタイミングパラメーターは、LIN プロトコルバージョンによって異なります。LIN プロトコルバージョンごとのタイミングパラメーターをまとめると以下になります（表 1）。

パラメーター	略語	LIN1.3	LIN2.0	LIN2.1
バスアイドルタイムアウト	TTIME_OUT	25,000TBit	4s	4～10s
ウェイクアップシグナル（ドミナント）	TWUSIG	Nom.8TBit	0.25～5ms	0.25～5ms
ウェイクアップシグナル デリミタ（リセッシブ）	TWUDEL	4～64TBit	100ms	100ms
ウェイクアップシグナル送信後のタイムアウト（リセッシブ）	TT0BRK	Max. 128TBit	150ms	150～250ms
ウェイクアップシグナル3回送信後のタイムアウト（リセッシブ）	TT3BRK	Min. 15,000TBit	1.5s	1.5s

表 1：LIN プロトコルバージョンごとのタイミングパラメーター

LIN1.x では、タイミングパラメーターはビット時間（TBit）で定義されています。つまり、LIN1.x では、通信速度によってタイミングパラメーターが異なります（表 2）。

通信速度 [bit/s]	1,000（Min）	9,600	19,200	20,000（Max）
TBit [μ s]	1,000	104	52	50

表 2：LIN1.x は通信速度によりタイミングパラメーターが異なる

2.3 LIN 記述ファイル（LDF）

LIN 記述ファイルは、LIN ネットワークを定義する標準形式で、LIN の仕様書に規定されています。LIN 記述ファイルには通信速度、ノード、フレーム、シグナル、通信スケジュールなどの LIN 通信に必要な情報を定義します。

LIN 記述ファイルのフォーマットは、テキストベースとなっており LIN の通信規格と同様、非常に簡易的です（図 25）。

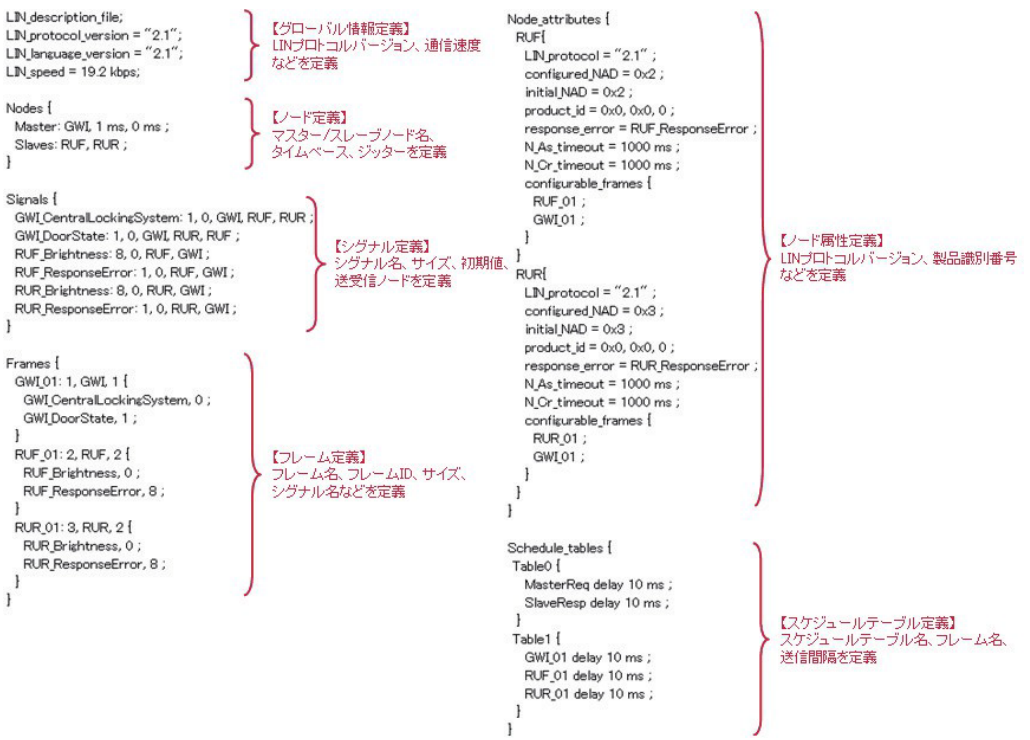


図 25：LIN 記述ファイルの例
※ほかに、イベントトリガーフレームやコリジョンリゾルビングスケジュールテーブルなどの定義がある

LIN 記述ファイルは、各開発工程間や自動車メーカーとサプライヤ間の共通インターフェイスとして機能し、非互換性などの問題を軽減できます。またシステムジェネレータによる LIN 通信ドライバの生成、解析・シミュレーションツールの設定に使用します。解析・シミュレーションツールでは、スケジュールに従ったヘッダー送信、スケジュールテーブルの切り替え、シグナル単位での解析や値の変更などが簡単に行えます（図 26）。このように、LIN 記述ファイルを使用することで効率的な開発が可能となります。

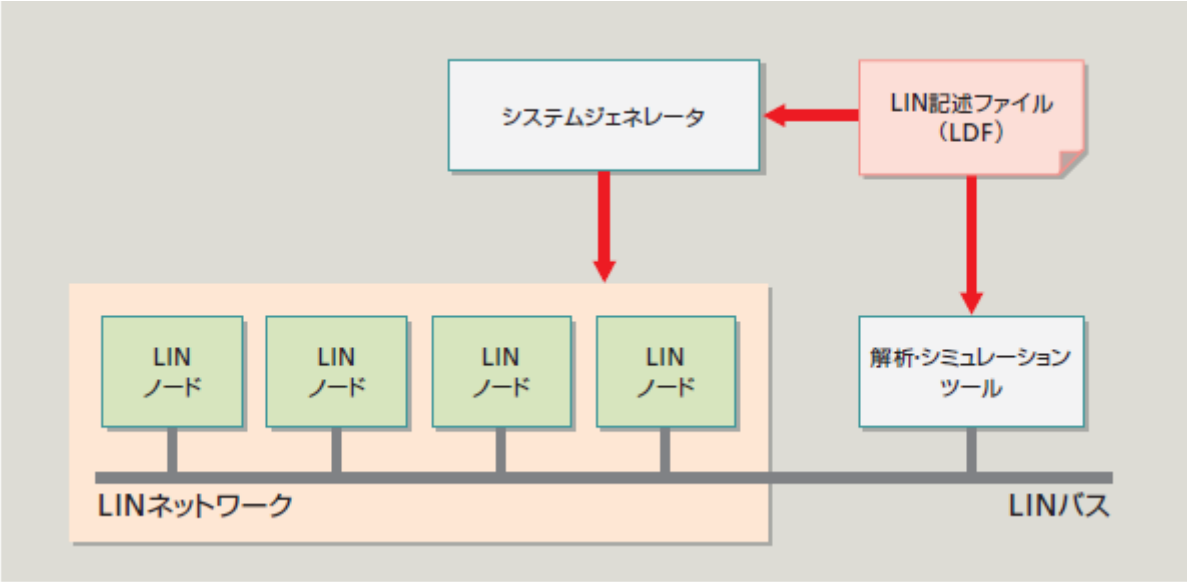


図 26：LIN 記述ファイルによる効率的な開発

「知っておきたい LIN の基礎知識 その 2」として、LIN フレームの構造、ネットワークマネジメント、LIN 記述ファイルについて説明しました。LIN フレームの各フィールドの役割や LIN フレームタイプの違い、LIN 記述ファイル使用のメリットなど理解いただけましたでしょうか。

次は、「ステータスマネジメント」「ノードコンフィグレーションと識別」など、LIN2.0、2.1 で追加された仕様について解説します。

3. LIN2.0/LIN2.1 の新仕様

ここまでは、LIN プロトコルの特徴やフレーム構造、フレームタイプといった LIN プロトコルの概要について説明しました。ここからは、LIN2.0/LIN2.1 で追加された「ステータスマネジメント」「診断」「ノードコンフィグレーションと識別」「ノード機能ファイル」について詳しく説明します。

3.1 ステータスマネジメント

LIN2.0 では、各スレーブノードがエラーを検知した際に、マスターノードに報告するステータスマネジメントが追加されました。

これにより、マスターノードはエラーの発生や原因となるノードの特定ができるので、より適切なリカバリ制御や処理が可能となります。ただし、「知っておきたい LIN の基礎知識 その 2」でも説明したとおり、LIN の仕様にはエラー処理が定義されていないため、別途アプリケーションで定義する必要があります。各スレーブノードは、マスターノードにエラー状況を報告するために、送信するアンコンディショナルフレームの 1 つ（ステータス管理フレーム）に、1 ビットのレスポンスエラー（response_error）シグナルを定義します。

このレスポンスエラーシグナルは、フレームのレスポンスでエラーが検出された場合にスレーブノードのドライバにより自動的に設定されます。また、ステータス管理フレームの送信が完了すると、このシグナルは自動的にクリアされます。

3.2 診断

車載ネットワークにおける“診断”とは、ECU が自身の状態を判断すること、または外部から ECU の状態を読み出し判断することにより、ECU の故障を診断するものです。

外部から診断を行う場合、診断テスターなどの「情報を読み出す側」から「対象となる ECU」に対し、「診断要求（リクエスト）」を送信します。診断要求を受信した ECU は要求に応じた処理を行い、「診断応答（レスポンス）」によって結果を返信します。

この診断応答には、受信した診断要求を正常に処理できた場合に送信される「肯定応答（ポジティブレスポンス）」と、エラーなどで処理できなかった場合に送信される「否定応答（ネガティブレスポンス）」の 2 種類があります。

LIN2.0 では、この診断の仕様が追加されました。マスターノードとスレーブノードでは、以下のように診断の方法が異なります（図 27）。

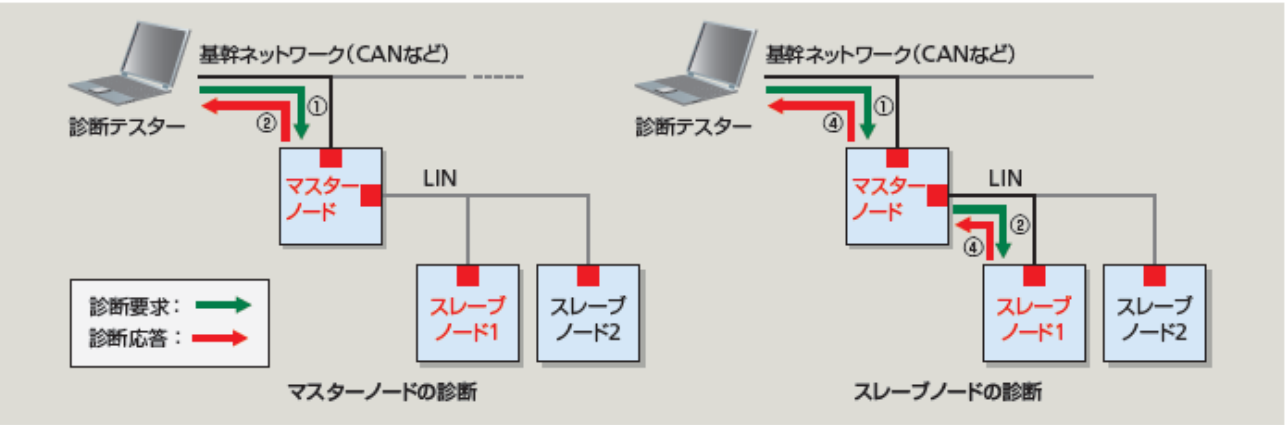


図 27：LIN2.0 で追加された「診断」の仕様
※マスターノードの診断（左図）は、CAN などの基幹ネットワークを使用する。スレーブノードの診断（右図）は、マスターノードを経由（ゲートウェイ）して、LIN ネットワークを使用する

✓ **マスターノードを診断する場合**

マスターノードは、通常 CAN などの基幹ネットワークと接続されます。そのため、LIN ではなく基幹ネットワークを使用します。

✓ **スレーブノードを診断する場合**

LIN 通信は、マスターノードが通信制御を行うため、スレーブノードは診断テスターと直接通信できません。そのため、マスターノードを経由して行います。このように、LIN 通信を用いた診断はスレーブノードに対してのみ使用され、マスターノードはスレーブノードに診断要求を、スレーブノードはマスターノードに診断応答を送信します。次項では、診断で使われる通信手順「トランスポートプロトコル」について説明します。

3.2.1 トランスポートプロトコルとは

トランスポートプロトコルとは、OSI 参照モデルのネットワーク層の通信手順で、LIN では 8 バイトを超えるデータを分割して送受信する際に使用します。LIN のトランスポートプロトコルは、基本的に CAN で使われている「ISO 15765-2」と同じで、データバイト内の一部に特別な情報を格納して送信されます。LIN2.x では、このトランスポートプロトコルを診断、ノードコンフィグレーションと識別に使用します。トランスポートプロトコルを用いた通信では、診断フレーム（ID:0x3C、0x3D）を使用します。マスター ノードは、対象となるスレーブノードに「マスターリクエストフレーム（ID:0x3C）」を送信し、診断要求（リクエスト）や設定の変更などを行います。また、診断要求を受信したスレーブノードは「 スレーブレスポンスフレーム（ID:0x3D）」を使用し、診断応答（レスポンス）や設定変更の結果をマスターノードに送信します。スレーブノードの識別には、スレーブノードごとに割り振られたノードアドレス（NAD）を使用します（図 28）。

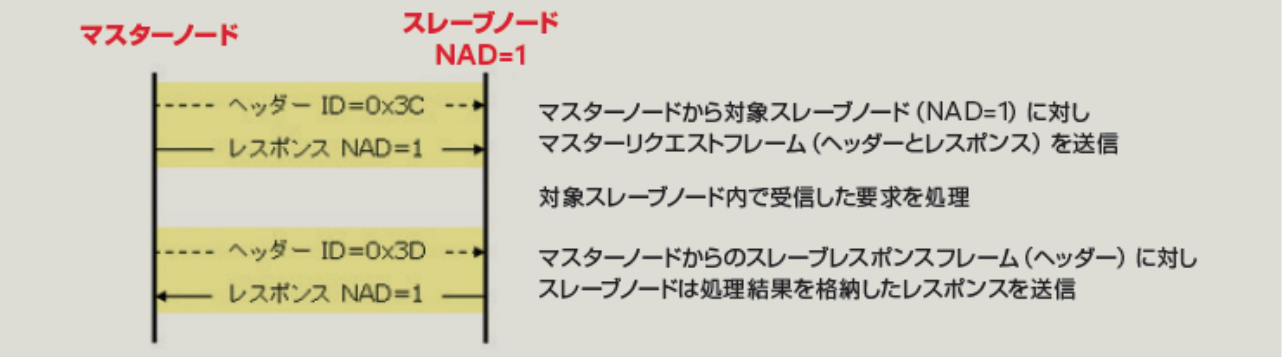


図 28：スレーブノードの識別
※診断フレームのフレーム ID は共通となるが、NAD を使用することでスレーブノードごとに個別の診断フレームを送信できる

3.2.2トランスポートプロトコルの LIN フレーム（データバイト）構造

トランスポートプロトコルを使用した場合、データバイトには下記の値が格納されます（表 3）。トランスポートプロトコルの送信方法には、下記の 2 種類があります。

- 1 つのフレームで送信する「非分割送信（シングルフレーム送信）」
- 複数のフレームを使用して送信する「分割送信（マルチフレーム送信）」

名 称	意 味
NAD（Node Address）	ノードを特定するために各スレーブノードに割り振られたアドレス。データバイトの1バイト目に格納する
PCI（Protocol Control Information）	上位4ビットで「フレームの種類（PCIタイプ）」、下位4ビットで「データ長」を表す。データバイトの2バイト目に格納する 上位4ビットが0000 : Single Frame（SF） 上位4ビットが0001 : First Frame（FF） 上位4ビットが0010 : Consecutive Frame（CF）
LEN（Length）	データ長。最大4095 バイト
SID（Service Identifier）	診断要求の意味（サービス）を表す
RSID（Response Service Identifier）	診断応答の意味（サービス）を表す。SIDに0x40 を加算した値が格納される （SID が「0x10」の場合、RSIDは「0x50」となる）
Err（Error Code）	否定応答で使用し、エラーの意味を表す
D1 ～ D6	データバイト。SID、RSIDにより格納する情報が異なる。各フレームのデータ長は 8バイト固定で、使用しないデータバイトには「0xFF」が格納される

表 3:トランスポートプロトコルを使用した場合のデータバイトの値

➢ 非分割送信（シングルフレーム送信）

送信するデータが、1 つのフレームに収まる場合に使用されます。PCI には「0x01~0x06（データ長 1 バイトから 6 バイトのシングルフレーム）」が格納されます（図 29）。

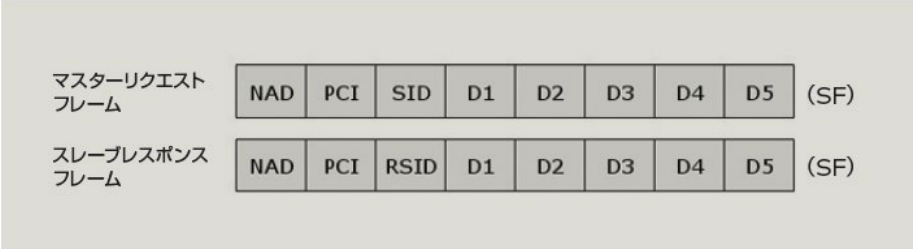


図 29：非分割送信について（1）
※データバイトの最初 3 バイトに NAD、PCI、SID（RSID）を格納するため、シングルフレームで送信できるデータは SID（RSID）を含め、最大 6 バイトとなる。つまり、SID を含めて 7 バイト以上のデータを送信する場合はマルチフレームを使用する

受信したスレーブノードが要求を処理できない場合（受信したサービス（SID）が未対応など）、否定応答となります。否定応答ではサービスに関係なく、PCI は「0x03（データ長 3 バイトのシングルフレーム）」、RSID は否定応答を表す「0x7F」になります。また、D1 は SID、D2 はエラーコード、D3~5 は使用しないため「0xFF」になります（図 30）。

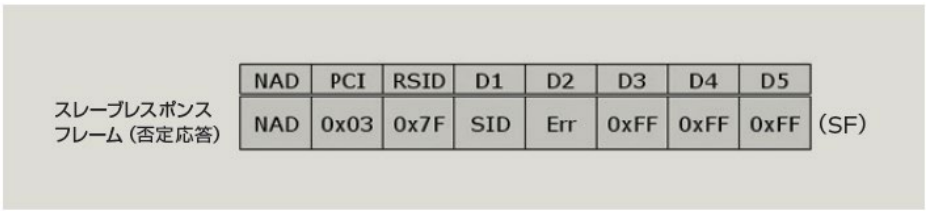


図 30：非分割送信について（2）

➢ 分割送信（マルチフレーム送信）

送信するデータが、1 つのフレームに収まらない場合に使用されます。マルチフレームには、最初のフレーム「First Frame（FF）」と 2 つ目以降のフレーム「Consecutive Frame（CF）」の 2 種類があります（図 31）。

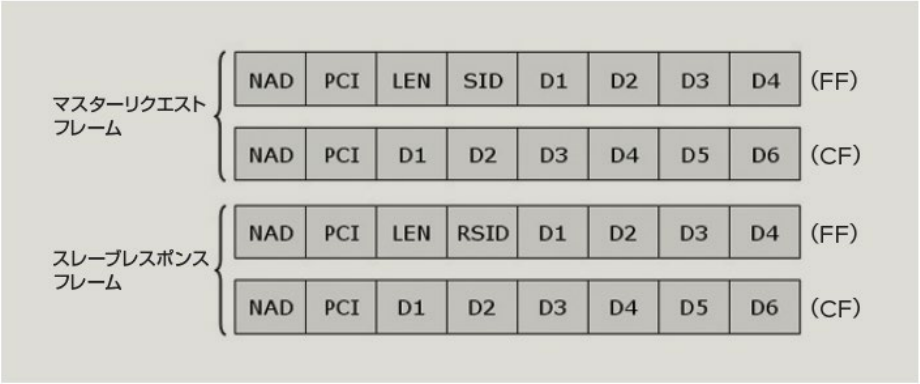


図 31：分割送信について
※マルチフレームの最初の LIN フレームは FF、2 つ目以降は CF と呼ばれる。すべてのデータが送信されるまで CF を送信する

また、LIN2.1 では 1 つのマスターリクエストフレームで全スレーブノードに診断要求を行う「ファンクショナルアドレッシング」の仕様が追加されました。ファンクショナルアドレッシングで送信する場合、ノードアドレスには「ファンクショナルノードアドレス（0x7E）」が格納されます。

3.3 ノードコンフィグレーションと識別

ノードコンフィグレーションと識別は、スレーブノードを効率的に再利用するためのサービスです。

LIN 記述ファイルには、LIN ネットワークに必要な情報が定義されており、送受信するノード、フレーム ID、シグナルなど、スレーブノード固有の情報も含まれています。つまり、スレーブノード固有の情報は、LIN ネットワーク固有の情報として定義されます。

また、LIN 記述ファイルの定義は通信によって動的に変更できません。そのため、あるスレーブノードを別の LIN ネットワークで使用する場合、フレーム ID の重複などが発生する可能性があるため容易に再利用できません（図 32）。

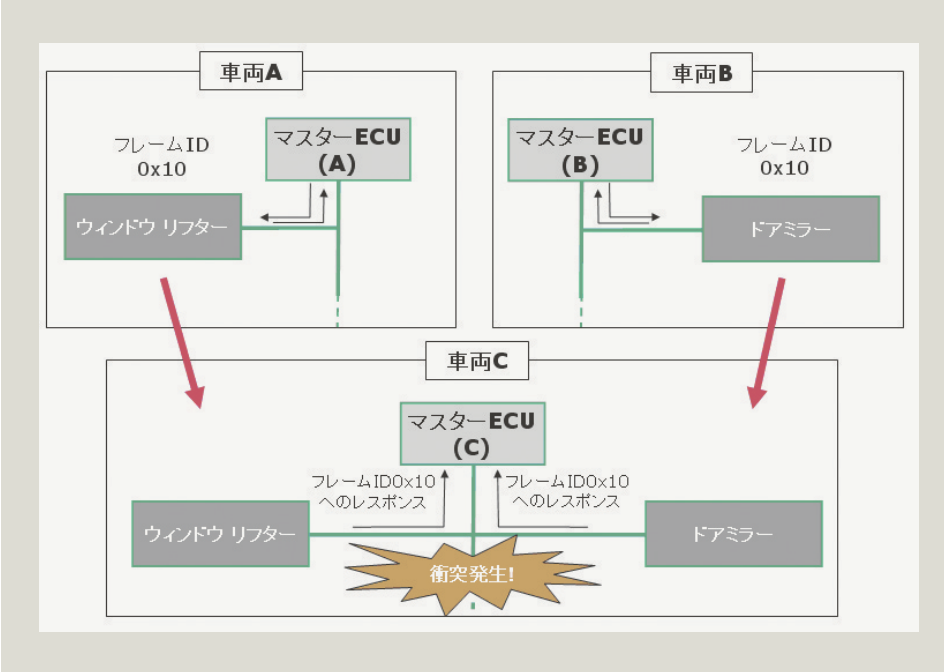


図 32:レスポンスの衝突
※車両 A、B で使用している 2 つのスレーブノードを車両 C で再利用した場合、フレーム ID0x10 が重複しているため、レスポンスが衝突する。フレーム ID は車両 A、B の各 LIN 記述ファイルに定義されているため、車両 C の LIN 記述ファイルの変更では衝突を回避できない。そのため、再利用するスレーブノードのアプリケーションなどを変更する必要がある

「スレーブノードの効率的な再利用」に対応するため、LIN2.x では以下の仕様が追加されました。

- スレーブノード固有の情報を定義する「ノード機能ファイル（NCF）」
- スレーブノードを通信によって再設定する「ノードコンフィグレーションと識別」

3.3.1 ノード機能ファイル（NCF）

ノード機能ファイル（NCF:Node Capability File）は、1 スレーブノードの固有情報を定義する標準形式で、LIN 記述ファイルと同様、LIN の仕様書に規定されています。ノード機能ファイルには、ネットワークに依存しない情報を定義します（図 33）。

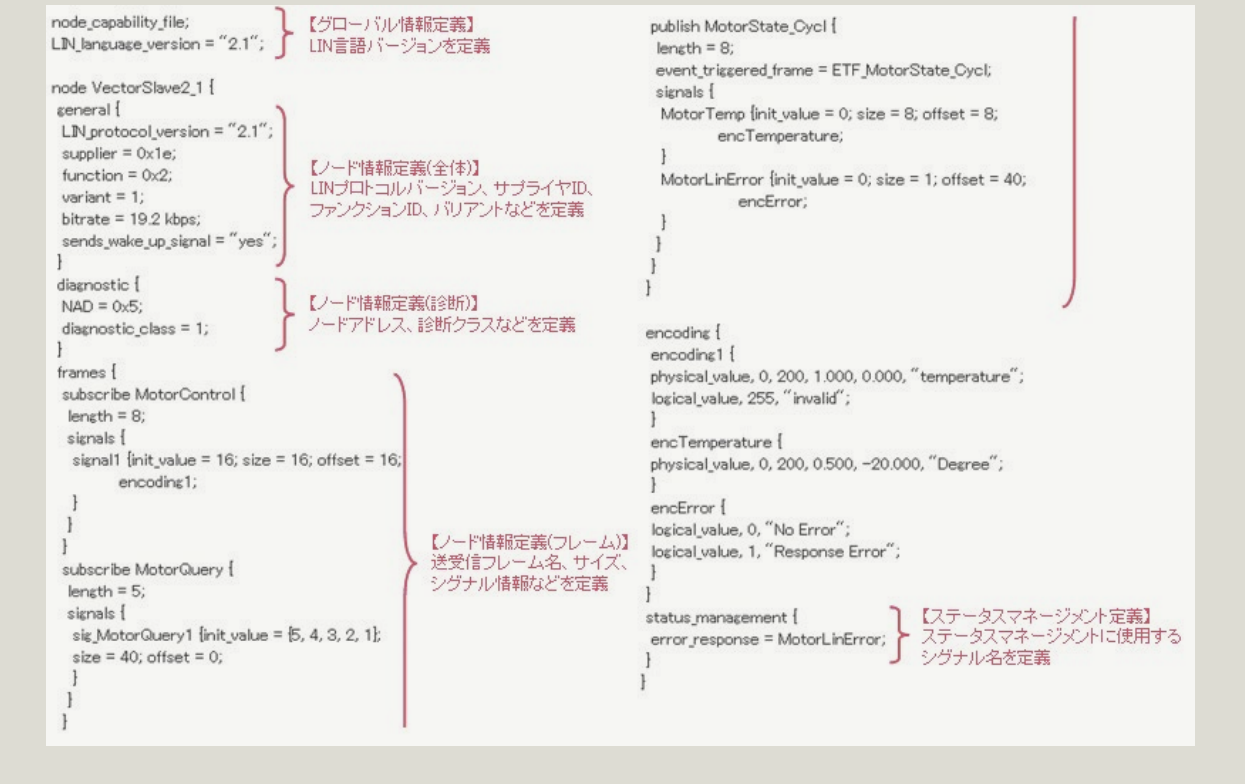


図 33:ノード機能ファイルの例
※ LIN 記述ファイルには LIN ネットワーク全体を定義するが、ノード機能ファイルにはスレーブノードの固有情報を定義する

ノード機能ファイルには、スレーブノードが送受信するフレーム ID、シグナルなどのほかにスレーブノードを特定するためのノードアドレス（NAD）も定義します。ただし、ノード機能ファイルに定義されているノードアドレスが、ほかのスレーブノードと重複している可能性もあるため、ノード機能ファイルにはスレーブノードごとに必ず重複しない「製品識別番号」と呼ばれる下記の 3 つの値を定義します。

- サプライヤ ID

  LIN コンソーシアムがサプライヤごとに指定する 16 ビットの ID
- ファンクション ID

  各サプライヤが機能ごとに指定する 16 ビットの ID
- バリアント

  各サプライヤがハードウェアやソフトウェアの修正ごとに指定する 8 ビットの ID

ノード機能ファイルの情報は、システム定義ツールを使用して LIN 記述ファイルに結合されます。これにより、スレーブノード固有の情報を簡単に LIN 記述ファイルに定義できます（図 34）。

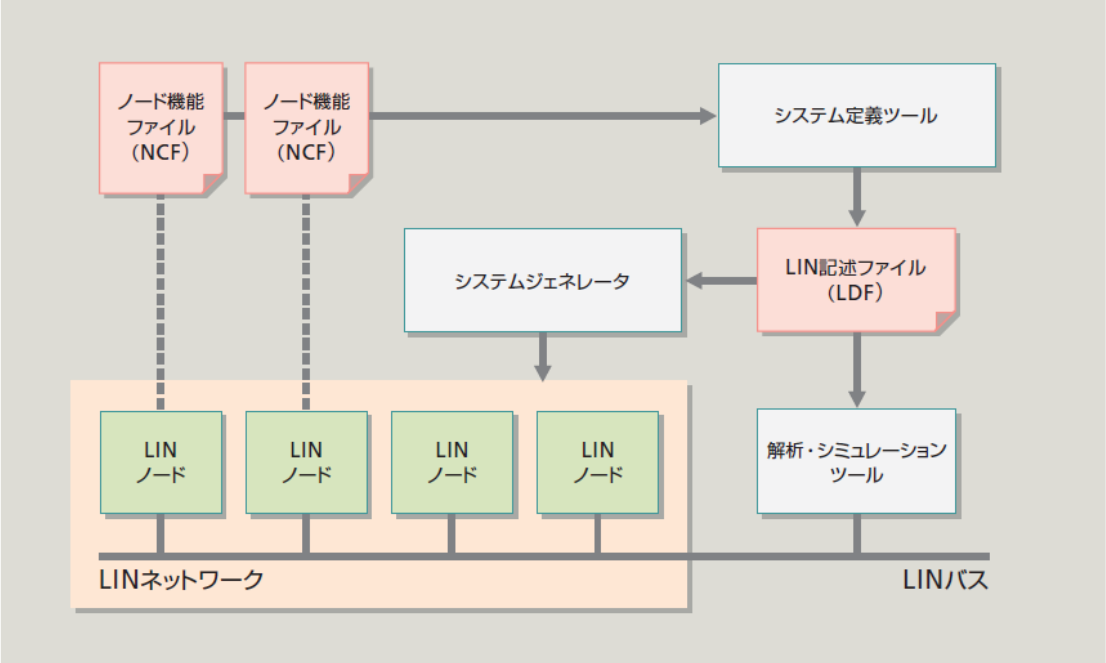


図 34：ノード機能ファイルの情報
※ LIN2.1 では、システム定義ツールを「LIN クラスタデザインツール」、システムジェネレータを「LIN クラスタジェネレータ」と呼ぶ。クラスタとは LIN バスとすべてのノードを指す

3.3.2 ノードコンフィグレーションと識別のサービス

ノード機能ファイルは、1 スレーブノードに対する定義となるため、ほかのスレーブノードのノードアドレスやフレーム ID と重複する可能性があります。

このスレーブノード間の設定の重複などを解決するために追加された仕様がノードコンフィグレーションと識別のサービスです。

ノードコンフィグレーションと識別は、トランスポートプロトコルのシングルフレームを使用し、スレーブノードを 1 つずつ設定します。ノードコンフィグレーションと識別で定義されているサービス（SID）は「0xB0」から「0xB7」です。

- Assign NAD（SID:0xB0）※オプションサービス

  サプライヤ ID、ファンクション ID、バリアントを使用し、ノードアドレスを再設定します。これにより、ノードアドレスの重複を解決します。

- Assign Frame ID（SID:0xB1）※必須サービス、LIN2.0 のみ

  ノード定義ファイルに設定されたメッセージ ID を使用し、1 フレームごとに 1 つの PID を再設定します。これにより、フレーム ID の重複を解決します。LIN2.1 ではフレーム ID の再設定に「Assign Frame ID Range（SID:0xB7）」を使用します。

- Read by Identifier（SID:B2）※必須サービス

  スレーブノードの製品識別番号などの情報を読み込みます。

- Conditional Change NAD（SID:0xB3）※オプションサービス

  LIN ネットワーク内の不正なスレーブノード（誤ったノードを接続した場合など）を検出し、NAD を変更します。

- Data Dump（SID:0xB4）※オプションサービス

  開発に必要な任意の診断を行う場合に使用します。開発時のみ有効です。

- Assign NAD via SNPD（SID:0xB5）

  SNPD（Slave Node Position Detection）仕様により予約されているサービスです。

- Save Configuration（SID:0xB6）※オプションサービス、LIN2.1 のみ

  スレーブノードが不揮発性 RAM をサポートしている場合、設定されている NAD、PID を不揮発性 RAM に保存します。

- Assign Frame ID Range（SID:0xB7）※必須サービス、LIN2.1 のみ

  1 フレームごとに最大 4 つの PID を再設定します。これにより、フレーム ID の重複を解決します。

  1 フレームで複数のフレーム ID を再設定できるため、LIN2.0 のサービス「Assign Frame ID（SID:0xB1）」よりも設定時間を短縮できます。

3.4 異なるプロトコルバージョンのノードが混在した場合の動作

最後に、LIN ネットワーク内に LIN プロトコルバージョンが異なるノードが混在した場合の動作について説明します。

LIN ネットワークでは、マスターノードが通信を制御するため、マスターノードのプロトコルバージョンにより使用できるスレーブノードのプロトコルバージョンが決まります。

- マスターノードのプロトコルバージョンが LIN1.3 の場合

  LIN2.x のスレーブノードは使用できません。

- マスターノードのプロトコルバージョンが LIN2.x の場合

  LIN1.3、LIN2.x のスレーブノードが使用できます。ただし、スレーブノードが LIN1.3 の場合、マスターノードは拡張チェックサムや診断などの LIN2.x の仕様は使用できません。また、LIN2.0/2.1 が混在した場合も同様に、ファンクショナルアドレッシングやノードコンフィグレーションと識別のサービスなど、プロトコルバージョン間で異なる仕様は使用できません。

LIN は、主にドアミラーやパワーシートなどのボディ制御に使用されるシンプルかつ安価な車載向けサブネットワークシステムとして策定されました。今後も快適装備品の増加や、さらなるコスト削減の取り組みに伴い、LIN が使われるケースが多くなると予想されます。また、複雑化する制御や通信を実現し、効率の良い開発を行うために、LIN2.0/2.1 で追加された仕様を採用するノードも増加していくと考えられます。

本稿で説明した LIN の基礎知識が、皆さんの LIN プロトコルの理解と導入の一助となれば幸いです。

4. お問い合わせ窓口

技術関連のお問い合わせは、ベクターサポートまでお寄せください。

✓ サポートリクエスト（Web サイト経由のサポート問い合わせフォーム）
www.vector.com/jp/ja/support-downloads/support/

【お問い合わせ時のお願い】

お問い合わせの際、サポート開始前に製品のバージョン（例：CANape バージョン xx.x など）、シリアルナンバーなどをお伺いいたします。お手数ではございますが、事前にご確認のうえ、お問い合わせくださいますようお願い申し上げます。

【サポート対応について】

夜間／休日に頂いたお問い合わせは翌営業日の対応となります。また、内容によりご回答までにお時間を頂く場合がございます。なお、不具合が発生した際にご使用されていた関連プログラム一式のコピーを、検証のためご提供いただくようお願いする場合がございます。何卒ご理解ご協力のほど、よろしくお願い申し上げます。

✓ **お見積もり／保守契約関連のお問い合わせ**

営業部　TEL：（東京）03-4586-1808　E-mail：sales@jp.vector.com

VECTOR >

MORE INFORMATION
Visit our website for:

> News
> Products
> Demo software
> Support
> Training and workshops
> Contact addresses

vector.com