VECTOR >

はじめての CAN / CAN FD

ビギナー向け資料「はじめてシリーズ」

vector.com

目次

はじめに 4

1. CAN とは？ 4
   - **1.1 歴史 4**
   - **1.2 適用範囲 4**
   - **1.3 自動車における CAN の使用例 6**

2. CAN の基礎知識 8
   - **2.1 特長 8**
     - 2.1.1 ライン型構造 8
     - 2.1.2 マルチ・マスター方式 9
     - 2.1.3 CSMA/CA 9
     - 2.1.4 ID を使用したメッセージ・アドレッシング 10
     - 2.1.5 耐ノイズ性に優れた物理層 10
     - 2.1.6 優れたエラー検出機構 11
     - 2.1.7 データの一貫性 11
   - **2.2 キーワード 11**
     - 2.2.1 ドミナントとレセシブ 12
     - 2.2.2 シグナルコーディング 12
     - 2.2.3 通信速度 13
     - 2.2.4 同期 13
     - 2.2.5 ビットスタッフィングルール 13

3. CAN プロトコル 15
   - **3.1 フレームタイプ 15**
     - 3.1.1 データフレーム 15
     - 3.1.2 リモートフレーム 19
     - 3.1.3 オーバーロードフレーム 21
     - 3.1.4 エラーフレーム 22
   - **3.2 通信調停 23**
   - **3.3 エラー処理 25**
     - 3.3.1 エラー検出 25
     - 3.3.2 エラー検出時の動作（1） 27
     - 3.3.3 エラー検出時の動作（2） 28
     - 3.3.4 受信エラーカウンター 29
     - 3.3.5 送信エラーカウンター 30
     - 3.3.6 「パッシブ」ノードにおける送信待機 31

3.3.7 エラー発生時の CAN コントローラの動作 32

4. CAN の物理層 33
4.1 バスへの接続 33
4.2 High Speed CAN / Low Speed CAN 34
4.2.1 High Speed CAN 34
4.2.2 Low Speed CAN 35

5. CAN FD とは？ 36
5.1 CAN FD 導入の背景と概要 36

6. CAN FD プロトコル 37
6.1 フレームタイプ 37
6.2 CAN FD フレーム領域 38
6.2.1 SOF（Start of Frame） 38
6.2.2 アービトレーション領域 39
6.2.3 コントロール領域 40
6.2.4 データ領域 41
6.2.5 CRC 領域 42
6.2.6 ACK 領域 44
6.2.7 EOF（End of Frame） 45
6.3 CAN FD フレームの波形 45

7. CAN FD の標準化 46

8. FAQ 47
Q1: 同一ネットワーク上に CAN ノードと CAN FD ノードを混在させることは可能ですか? 47
Q2: CAN FD の最大転送速度は? 47
Q3: CAN FD を導入するために必要なことは? 47

あとがき 47

9. お問い合わせ窓口 48

発行元：ベクター・ジャパン株式会社

はじめに

近年の自動車の制御は、複数の ECU の協調制御によって実現されています。ECU が協調動作するための技術基盤の一つが車載ネットワークです。車載ネットワークは、パワートレイン系、ボディー系、快適装備系など自動車を幅広く制御するシステムであるため、安全性、信頼性、外的要因への耐性、開発工数、コストなど、さまざまな面で高い要求を満たさなければなりません。現在、自動車で使用されている車載ネットワークは、「CAN（Controller Area Network）」、「LIN（Local Interconnect Network）」、「MOST（Media Oriented System Transport）」、「FlexRay」、「Ethernet」といったものがあります。

CAN は、Robert Bosch 社によって 1986 年に仕様が公開され、1994 年には国際規格（ISO 11898）となり、現在もっとも普及している車載ネットワークプロトコルといわれています。また近年では、通信の需要が急増し、CAN の拡張版である CAN FD（CAN with Flexible Data rate）プロトコルも登場しました。CAN FD は、2012 年に Robert Bosch 社によって発表され、2015 年に ISO 11898-1 の改訂版として仕様が反映されました。

本資料では、最初に、CAN の基本を学びたいという方々のために CAN を理解するための基礎知識や仕組みを一から解説していきます。そして次に、CAN FD について知りたいという方々のために CAN と CAN FD プロトコルとの違いについて解説していきます。

それでは、まず CAN から見ていきましょう。

1. CAN とは？

1.1 歴史

自動車の電子化は、1970 年代後半から 1980 年代にかけて登場した自動車に対する排出ガス規制と燃料規制を背景として始まりました。エンジンの最適な点火時期や、空気と燃料の混合比の精密な制御が必要となり、自動車は機械的メカ制御から電子制御へと進化していきました。しかし、自動車の制御機能が進歩する一方で電子部品は増加し、配線の重量やスペースも増大していきました。そこで登場したのが、車載ネットワークです。ワイヤーハーネスの代わりに、バス通信を用いて複数の電子制御装置（ECU：Electronic Control Unit）間の通信を行うことで、配線コストの削減と自動車の軽量化を実現しました。CAN は、車載ネットワーク用に開発されたシリアルバスシステムであり、冒頭でも述べたように Robert Bosch 社によって 1986 年に仕様が公開され、1994 年には国際規格（ISO 11898）となり、現在もっとも普及している車載ネットワークプロトコルといわれています。

1.2 適用範囲

現在、CAN は車載ネットワーク以外にも、産業機器、農業機械、医療機器、鉄道、船舶、航空宇宙など幅広い分野で採用されています。CAN の適用範囲には主に、標準フォーマットで使用される普通乗用車向けの「CAN」と、拡張フォーマットで使用される大型車向けの「J1939」があります。また CAN をベースとした「CANopen」というプロトコルもあり、これは主に産業機器向けとして使用されます（図 1、図 2）。

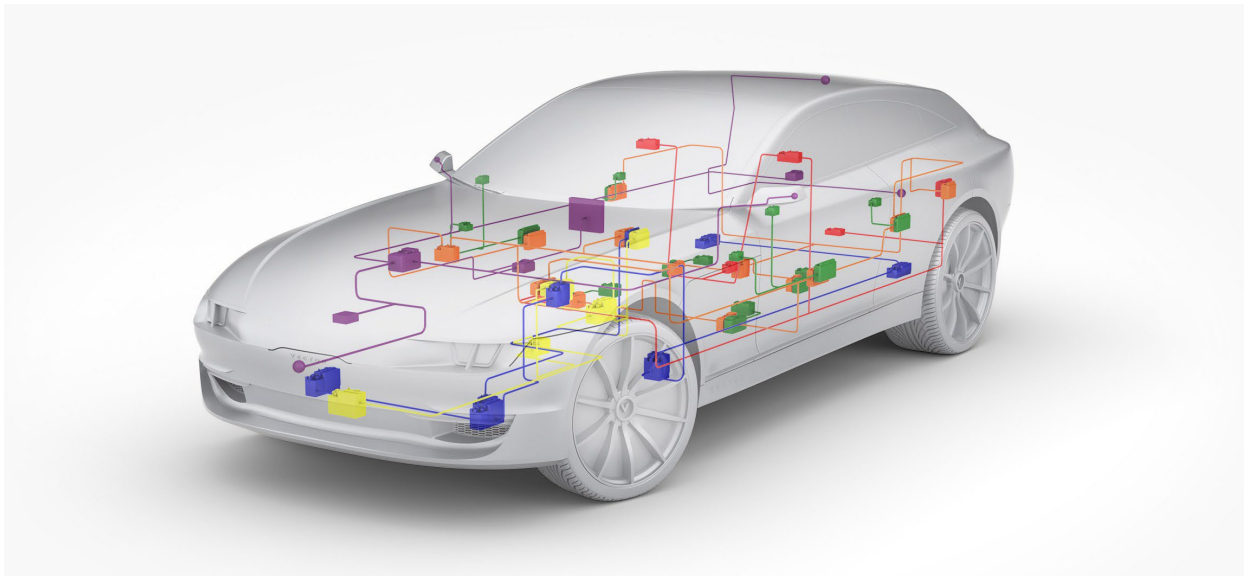


図 1:普通乗用車向け

図 2:普通乗用車以外(J1939/CANopen など)向け

1.3 自動車における CAN の使用例

CAN は自動車内のパワートレイン系やボディー系など、さまざまな部分で使われています。これまで主に ECU 間通信に使用されてきた CAN ですが、今では車載ネットワークの基本ともいえるほどに普及し、ECU 間のデータ交換だけではなく、他の用途で使用されるケースも増えています。

たとえば、これまで各 ECU の故障診断には専用の通信線（以下、バス）[1]を使用してきましたが、CAN を使って統合した「Diag On CAN（診断通信）[2]」（図 3）を使用するようになったり、また、CAN を用いて ECU 内部のデータを書き換えたり（キャリブレーション）、内部値を読み込んだりする「XCP（または CCP）[3]」（図 4）なども多く使われています。

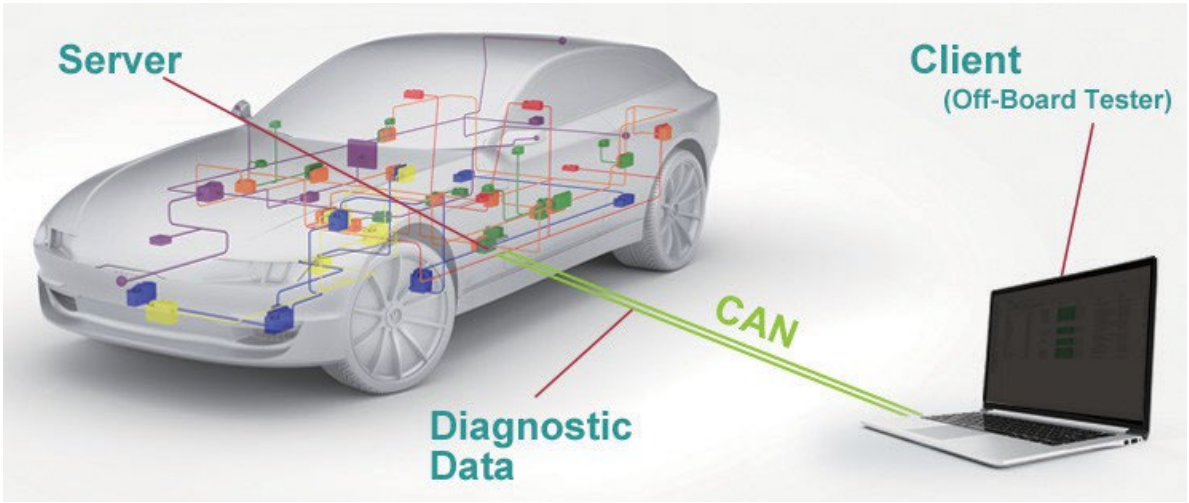


図 3：Diag On CAN

[1] 通信線はバス（bus）ともいい、各ノードを接続するもの。

[2] Diag On CAN とは、各 ECU の診断情報交換を CAN 通信で行うもの。

[3] XCP とは、CCP（CAN Calibration Protocol）を改善・統合したもので、CAN 以外のプロトコルでも使用可能。CAN 通信を使用して計測やキャリブレーションを行うことができます。

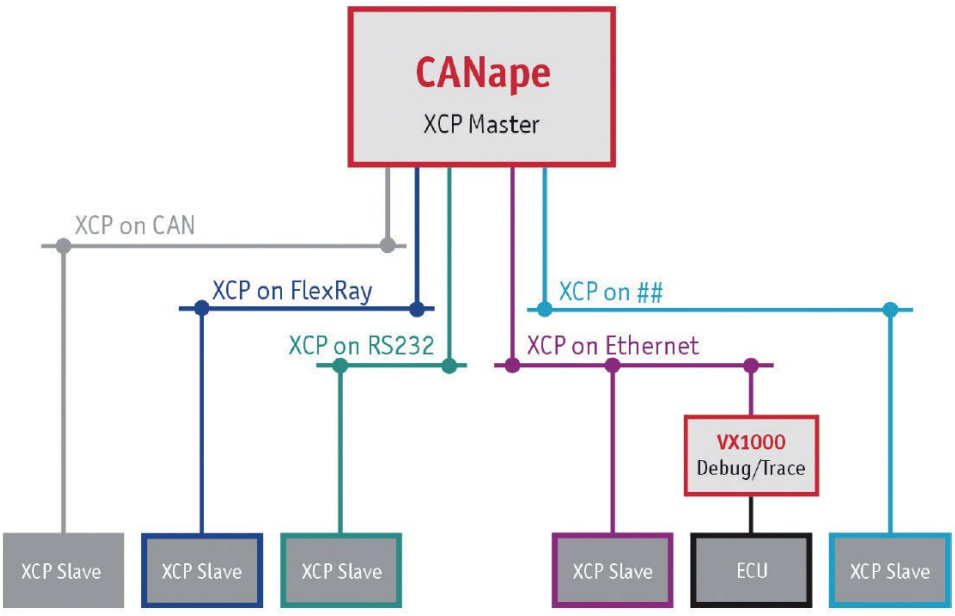


図 4：XCP

しかし、こうしたさまざまな用途での使用や ECU の増加は、バス負荷に影響を与えることが考えられます。そこで現在、CAN では 1 チャンネルを使用するのではなく、2 チャンネル以上を使用して多チャンネル化し、必要に応じた通信を別々のチャンネルに流す方式を採用しています。

また、CAN の多チャンネル化以外に、他の車載ネットワーク用プロトコルを同一車両内で使用するというケースも増えてきています。冒頭でも触れたとおり、他の車載ネットワークプロトコルには「LIN 」や

「FlexRay」などがあります。LIN は、CAN コントローラを使用するとコストへの影響が大きいセンサー・アクチュエーター分野に使用されます。FlexRay は、イベント型通信ではなくリアルタイム性を重視する分野に使用されます。

このように、同一車両内に異なるプロトコルで構成された車載ネットワークが存在していますが、それぞれのネットワークが別々に通信を行うケースはほとんどなく、多くの場合はプロトコルを変換して必要な情報を他のネットワークにデータ伝送するためのゲートウェイを設けています。

2. CAN の基礎知識

2.1 特長

CAN には、次のような特長があります。これらはどのような目的で採用され、どのようなことを行っているのでしょうか。以下に、その特長を一つずつ見ていくことにしましょう。

> ライン型構造
> マルチ・マスター方式
> CSMA/CA
> ID を使用したメッセージ・アドレッシング
> 耐ノイズ性に優れた物理層
> 優れたエラー検出機構
> データの一貫性

2.1.1 ライン型構造

世の中には車載用だけでなく、さまざまなネットワークが存在しています。一般的に、ネットワークに接続される通信機器のことを“ノード”といい、車載ネットワークの場合は電子制御ユニット（ECU：Electric Control Unit）がノードとなります。複数のノードを接続してネットワークを実現する方式には、スター型、リング型、ライン型などさまざまなものがありますが、CAN で採用されているのは「ライン型」と呼ばれる方式です（図 5）。

ライン型は、単純にバスに各ノードを接続していくことでネットワークが構成できるため、ネットワークがシンプルで設計も容易に行えるというメリットがあります。それに、冒頭でも述べたように、もともと車載ネットワークには複雑な配線を解消するという目的があるため、その点においてもライン型構造は目的に合致しているといえます。このように CAN は、ライン型バス構造によって、配線設計を簡単に行うことができ、後からのノード追加も比較的容易に行うことができます。

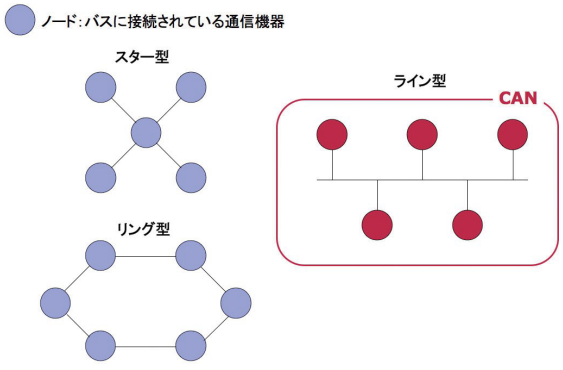


図 5：ネットワークの構造

2.1.2 マルチ・マスター方式

CAN では、ライン型で接続される各ノードに平等なバスアクセス 4が可能な「マルチ・マスター方式」（図 6）を採用しています。マルチ・マスター方式を採用するメリットとしては、以下の 3 つが挙げられます。

① 各ノードが均一仕様で設計できる

② 各ノードに優劣が無いため、イベント指向通信に向いている

③ ノードの追加接続が容易である

開発時に各ノードを同一仕様で設計でき、データ伝達を必要な場合に送信することが可能で、さらに各ノードに優劣が無いため自由度の高いネットワークを構成することができます。車載ネットワークの開発では、さまざまな要求に応じて設計を変更することが想定されるため、自由度の高いネットワーク構成は非常に重要な特長であるといえます。

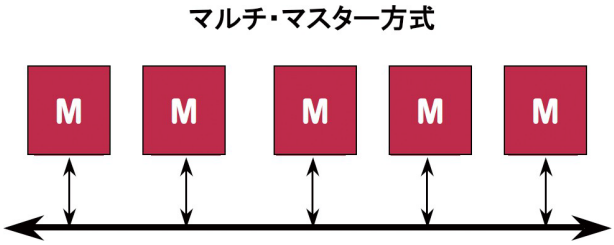


図 6：マルチ・マスター方式

2.1.3 CSMA/CA

通常は、複数のノードから自由にデータが送信されてしまうとデータが衝突してしまうため、バスが使用中の場合には送信できないようにする仕組みが採られています。しかし、複数のノードから同時にバスにデータが送信される場合には、これを防ぐことができません。

一方、CAN では「CSMA/CA（Carrier Sense Multiple Access with Collision Avoidance）」を採用し、複数のノードから同時にバスにデータが送信される場合には、優先順位が高いものを衝突させずに送信するような仕組みになっています。これにより、衝突が起きてしまった場合でも、重要なデータに高い優先順位を設定しておけばデータ伝達は確実に行われます（図 7）。

たとえば、エンジン ECU からのデータが重要であればエンジン ECU 側にあらかじめ高い優先順位を設定しておくことで、エンジン ECU からのデータとエアコン ECU からのデータが同時に送信された場合はエンジン ECU からのデータを破壊せずにバスに送信することが可能になります。

[4] バスアクセスとは、バスへ情報を送信したりすること。

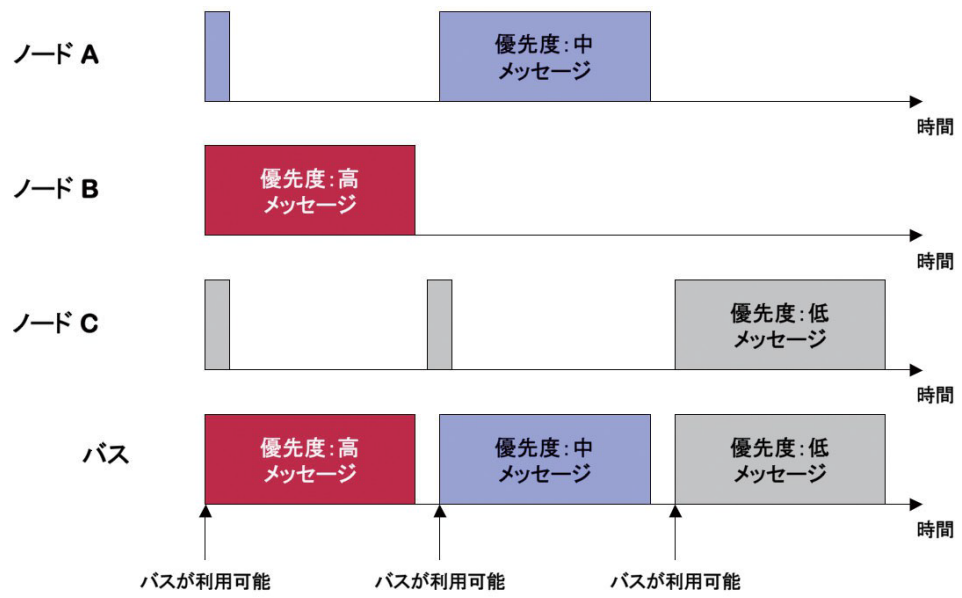


図 7：CSMA/CA

2.1.4 ID を使用したメッセージ・アドレッシング

前述した“重要なデータに高い優先順位を設定する”ことを実現するために、CAN では「ID（Identifier：識別子）を使用したメッセージ・アドレッシング」を採用しています。ID を使用したメッセージ・アドレッシングは、各ノード間でデータをやり取りする際に使用する方法で、データの中に ID を付加して送信する方式です。データを受信する側は、この ID によって“どのようなデータか”、“自分が使用するデータか”を判断することができます。この方式を採ることで、1 つの送信ノードからのデータを複数の受信ノードが同時に受信することが可能となります。

たとえば、エンジン ECU からのデータを、メーターECU とエアコン ECU が同時に受信して使用できるということになります。これにより、複数の ECU が同時に協調して制御を行うことが容易になるのです。

2.1.5 耐ノイズ性に優れた物理層

車載機器では外部ノイズによる影響は避けて通れませんが、CAN ではそれらの影響を受けない方式を採用しており、信頼性の高いネットワークを構築することが可能です。パワートレイン系に使用される High Speed CAN（CAN-C）では、「2 線式作動電圧方式」で通信を行っています。2 本線（ツイストペア）を使用し、それぞれの線に流れる電圧の差の有無によってデータを送信する方式です。この方式により、外部からノイズが混入した場合でも、それぞれの線に混入するノイズの電圧はほぼ同一となるため、電圧の差が有るか無いかに影響を受けにくい仕組みになっています（図 8）。

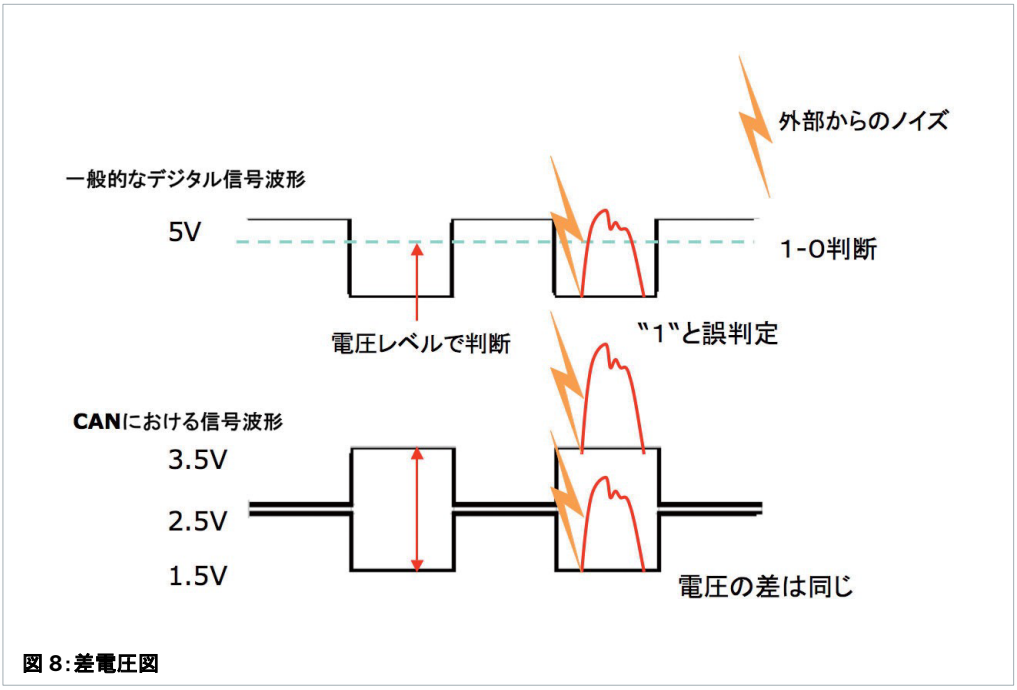


図 8:差電圧図

2.1.6 優れたエラー検出機構

安全面を重視する車載ネットワークにおいて、各ノードで使用されるデータが何らかの異常により間違ったまま届いて使われてしまい、正常な制御が行えない状態になることは避けなければなりません。CAN では、さまざまなエラー検出メカニズムを実装しており、ほぼ 100%に近い確率で各種エラーを検出することが可能となっています。たとえば、送信ノード自身が、送信するデータとバス上に流れたデータが合っているかを確認し、違いがある場合にはエラー検出することができます。

2.1.7 データの一貫性

車両制御において、あるタイミングで各ノードが協調し、同時に制御を行う場合があります。現在のエンジン回転数を使用して複数のノードが制御を行う場合、エンジン ECU から各ノードにエンジン回転数のデータが送信されます。通常、すべての受信側で正常に受信できれば問題はありませんが、もし 1 台のノードがエラーとなり、データを受信できなかった場合はどうなるのでしょうか。

たとえば、受信に失敗したノードのみ、エンジン ECU からデータを再送信してもらう場合を考えてみましょう。しかし、その場合はエンジン回転数の変化が起きた後のデータを受け取る可能性があるため、各ノードにて使用するエンジン回転数データが違ってしまう恐れがあります。

一方、CAN では、1 台のノードが受信に失敗した場合でも、データを受信した全ノードがデータを破棄し、全ノードが受信に成功するまでこれを繰り返します。つまり、ある制御を行う場合、全ノードにおいて使用するデータは同一ということになるので、CAN ではデータの一貫性を保つことができます。

2.2 キーワード

次に、CAN の基本を理解するために欠かせないキーワードについて、解説していきます。

2.2.1 ドミナントとレセシブ

CAN でのデータ伝達は、デジタル方式で行われます。送信されるデータは“0”と“1”の 2 進数に変換され、バスに送信されます。CAN では、送信される 2 進数データの“0”を「ドミナント」、“1”を「レセシブ」と表します。ドミナントは“優性”、レセシブは“劣性”の意味で、ドミナントとレセシブが別のノードから同時に送信された場合は、ドミナントが優先されます。CAN におけるさまざまな仕組みは、このドミナントとレセシブの関係により実現されていますので、これらの用語をよく理解しておくことが重要です。

2.2.2 シグナルコーディング

CAN では NRZ（Non-Return-to-Zero）方式で、送信したいデータを変換して送信しています（図 9）。この方式は比較的シンプルに変換が行えるため、さまざまな通信プロトコルで使用されています。NRZ 方式では、たとえば送信するデータが“0001100”の場合、連続する“0”の部分は“0”のまま、連続する“1”の部分は“1”のままとなり、ドミナント状態やレセシブ状態が連続することになります。状態が連続することによるデメリットもありますが、これを解消するための仕組みが用意されています。

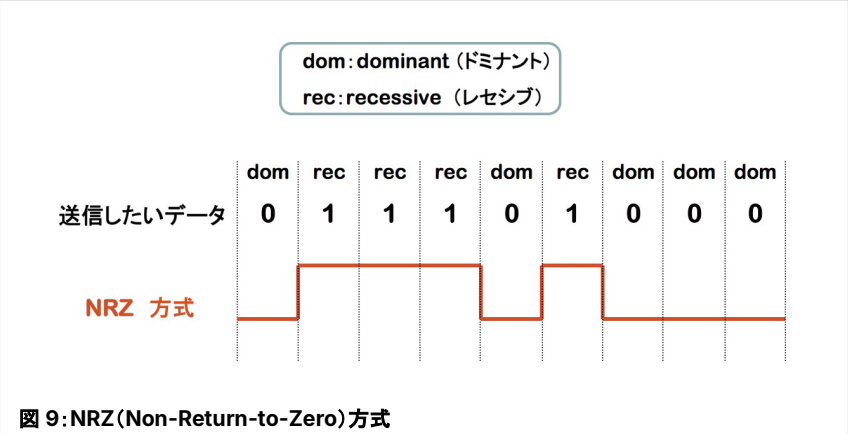


図 9：NRZ（Non-Return-to-Zero）方式

2.2.3 通信速度

通常、通信速度は bps（Bit Per Second）で表され、1 秒間に何ビットのデータが送信可能かを表します。たとえば 2bps の通信速度の場合は、1 秒間に 2 ビットのデータ伝達が可能です。この数値が高ければ高いほど、短い時間に大量のデータを伝えることができます。CAN では、1 回に送信できるデータ量は最大 8bytes、最大通信速度は 1Mbps となっており、車載ネットワークのほとんどの分野で使用可能です。

しかし、たとえば図 10 のようにノード A が 1bps、ノード B が 2bps で動作してしまうと、ノード A から"1"を 1 個送信するとノード B では"1"が 2 個受信されてしまい、ノード A－B 間のデータ伝達は正常に行えなくなってしまいます。通信を正常に行う上で、各ノード間の通信速度を同一に保つことは重要だといえます。

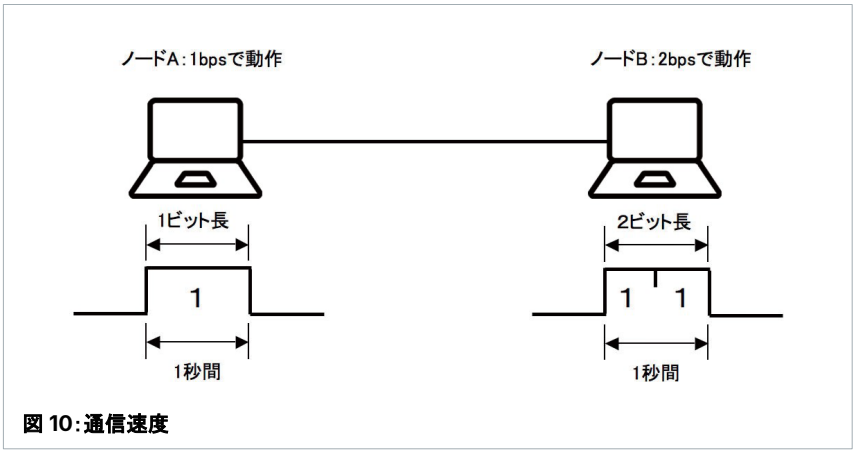


図 10：通信速度

2.2.4 同期

車載ネットワークでは多種多様なノードが接続されており、それぞれのノードは内部にプログラム処理などを行うための基準時間（以下、システムクロック）を作り出す"水晶発振子"を持っています。この水晶発振子は比較的正確に時間を作り出すことができますが、車載ネットワークの各ノードはさまざまな場所に設置されているため外気温などの影響を受けやすく、また、電源投入時からの時間変化などにより、結果として各ノードのシステムクロックに違いが生じてしまいます。

CAN では、システムクロックを使ってドミナントやレセシブ 1 ビット分の長さを構成していますが、もし、システムクロックが各ノードで違ってしまうと、それぞれの 1 ビットの長さが違ってしまうことになります。1 ビットの長さによって前述の通信速度に違いが生じるため、正常なデータのやり取りができなくなってしまいます。

これを防ぐ制御が「同期」と呼ばれるものです。CAN では、各ノード間のシステムクロックの違いを補正しており、信号がレセシブからドミナントへ変化する時（"1"－"0"変化時）に同期を行っています。

2.2.5 ビットスタッフィングルール

「シグナルコーディング」のところで述べたように、NRZ では"0000"のように"0"が連続した場合は、同じ状態が連続することになります。同期は"1"－"0"の変化時に行われるので、もし"0"が長時間連続した場合は同期ができない状態になってしまいます。そして、同期ができないと各ノードの通信速度にずれが生じ、正常に通信ができなくなります。そこで、CAN では「ビットスタッフィングルール」を採用しています。

ビットスタッフィングルールとは、バス上で同じ状態が 5 回連続した場合、それまで送信されていた状態と反対の状態のビット（"スタッフビット"）を 1 つ挿入する仕組みのことです（図 11）。

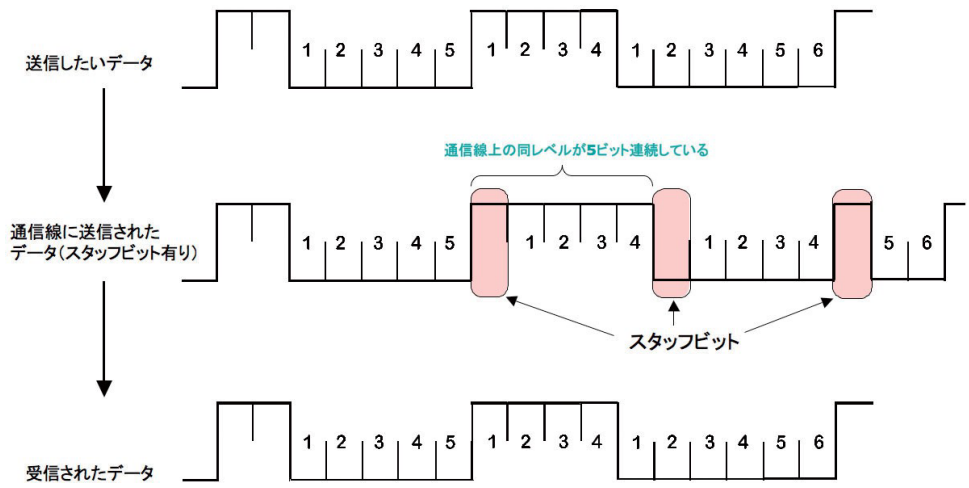


図 11:ビットスタッフィングルール

たとえば、“000000111111”と連続する場合、実際には“000001011111101”と送信されることになります(※赤文字は挿入されたスタッフビット)。“0000011111”と連続する場合は、“000001111101”です。挿入されたスタッフビットもバス上の状態としてカウントされます。このようにビットスタッフィングルールを採用することで、バス上で同じ状態が延々と連続することを防止しています。

しかし、実際に送信するはずのデータがスタッフビットによってバス上で変化してしまって大丈夫なのかと、心配される方もいらっしゃるかもしれません。しかし、送信側・受信側ともに同じルールで動作しているので、受信側は同じ状態が 5 回連続したら、次に受信されるビットはスタッフビットであると容易に判断でき、使用するデータにおいてはスタッフビットを除いたものを使用することができるのです。

さて、ここまでは CAN の基礎知識として、その特長やキーワードを見てきました。次は、CAN のデータ送信の具体的な仕組みについて見ていくことにしましょう。

3. CAN プロトコル

3.1 フレームタイプ

CAN には、以下 4 つの「フレーム [5]タイプ」があります。それぞれのフレームの構造と使われ方について、順に見ていきます。

- データフレーム
- リモートフレーム
- オーバーロードフレーム
- エラーフレーム

3.1.1 データフレーム

CAN において、実際にいろいろなデータを含むものを「データフレーム」といい、送信ノードから制御パラメーター等のデータを送信する役割を果たします。データフレームには「標準フォーマット（11 ビット ID）」と「拡張フォーマット（29 ビット ID）」という細部で異なる 2 種類の形式があります。

3.1.1.1 標準フォーマット

まずは、基本的な「標準フォーマット」について、図 12 を見ながら、順を追って説明していきます。

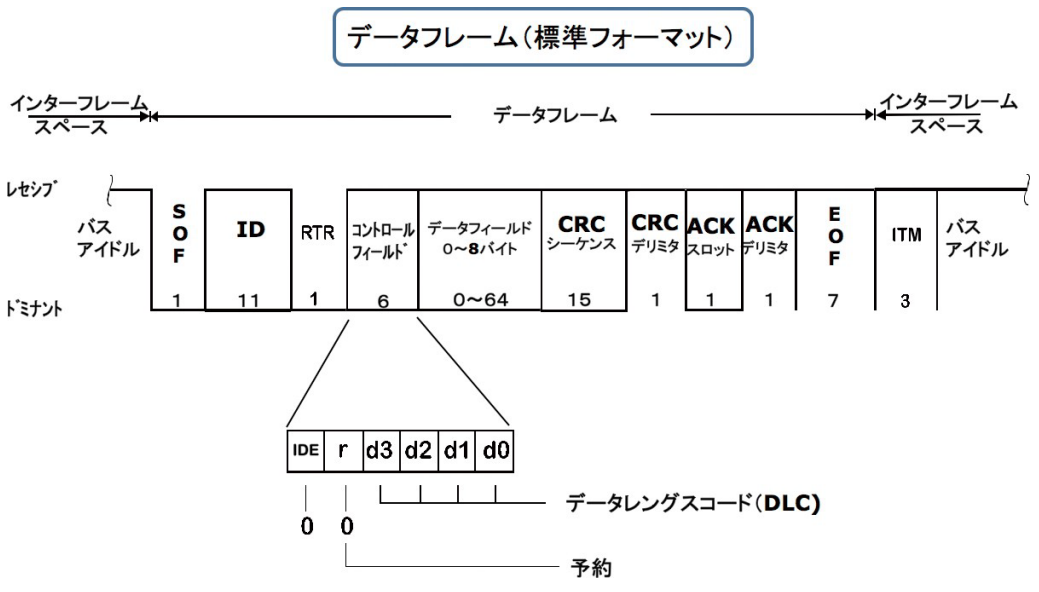


図 12：「標準フォーマット」のデータフレーム構造

[5] 通信されるデジタル信号のまとまりを「フレーム」といいます。

➤ レセシブとドミナント

上の線はレセシブ、下の線はドミナントを表しています。ドミナントにしか線がない部分はドミナント固定、レセシブにしか線がない部分はレセシブ固定となっています。両方に線があるものは、送信されるデータによって、ドミナントかレセシブかに変化します。

➤ ビット長

各部の数値は、何ビット分使用するかの長さ(ビット長)を表しています。

➤ バスアイドル

CAN では、通信が行われていない時、バスはレセシブとなっています。これを「バスアイドル」と呼びます。

➤ SOF

ノードからデータフレームが送信される時、最初に送信される部分は開始を表すためにドミナントとします。この部分を「SOF(Start Of Frame)」と呼びます。データフレームの開始を表し、バスアイドルのレセシブからドミナントへ変化することで、受信側ノードは同期を行うことができます。

➤ ID

SOF に続いて送信されるのは、「ID」です。この ID は、データ内容や送信ノードを識別するために使用され、通信調停の優先順位を決定する働きもしています。標準フォーマットでは、この ID は 11 ビット長で ID を構成していますが、拡張フォーマットでは、この ID(ベース ID)と拡張 ID の 18 ビット長で 29 ビット長の ID を構成しています。11 ビット長の場合、ID の範囲は 0x0～0x7FF となり、2048 種類の識別が可能です。

➤ RTR

ID の後は、「RTR(Remote Transmission Request)」です。データフレームとリモートフレームを識別するために使用されます。データフレームの場合には RTR はドミナントとなっています。ID と同様に、RTR も通信調停に使用されます。

➤ コントロールフィールド

RTR に続くのは、「コントロールフィールド」です。1 ビット長の IDE(Identifier Extension)、予約ビット r と 4 ビット長のデータレングスコード(DLC:Data Length Code)から構成されます。標準フォーマットの場合、IDE と予約ビット r はともにドミナントになっています。

➤ データレングスコード

「データレングスコード(DLC)」は、コントロールフィールドに続くデータフィールドにおいて、何バイトのデータが送信されるかを表しています。DLC の設定範囲は 0～8 までとなっており、結果としてデータフィールドで送信されるデータは 1 バイト単位で 0～8 バイトまで送信できます。

➤ データフィールド

「データフィールド」は送信されるデータの部分で、前述のように DLC によって設定されたデータ長となります。また、データフィールド内では全バイトは最上位ビット(MSB)より送信されます。データフィールドは 0～8 バイト長(0～64 ビット長)となっており、1 バイトごとに長さを設定できますが、どのような形式でデータを割り当てるかについては設計者が決定できるようになっています。

“たとえば、1 バイトのデータを割り当てる場合で”は、そのまま 1 バイトとして使用、“4 ビットをそれぞれ 1 ビットずつ使用”、“残り 4 ビットをまとめて使用”、“8 ビットをそれぞれ 1 ビットずつ使用”などが可能です。

➤ CRC シーケンス

データフィールドの後は、「CRC[6] シーケンス」が送信されます。CRC シーケンスは 15 ビット長であり、送信ノードが SOF、ID、コントロールフィールド、データフィールドの送信値より演算して CRC シーケンスで演算結果を送信します。これは受信ノードが送信ノードと同様に SOF、ID、コントロールフィールド、データフィールドの受信値から演算して、その結果を比較することで正常に受信できたかの判断を行うことができます。

➤ CRC デリミタ

CRC シーケンスの後には、「CRC デリミタ」が送信されます。これは CRC シーケンスの終了を表す区切り記号で、1 ビット長のレセシブとなっています。なお、CRC シーケンスと CRC デリミタを併せて「CRC フィールド」と呼びます。

➤ ACK スロット

続いて、「ACK（Acknowledgement）スロット」が送信されます。この ACK スロットは 1 ビット長で、送信ノードはこの部分でレセシブの送信を行います。ただし、受信ノードが CRC フィールド部分まで正常に受信できた場合は、ACK スロットのタイミングでドミナントを確認応答として送信することになっています。CAN では、ドミナントとレセシブが同時に送信された場合はドミナントが優先されドミナントとなるため、正常に通信を行っている CAN ネットワークにおいて ACK スロットはドミナントとなっています。しかし、この ACK スロットは 1 ビット長しかもたないために、CAN ネットワーク上の受信ノードがすべて正常に受信できたかの判断には使用できません。あくまで、送信ノードが送信したデータフレーム（CRC フィールドまでの部分）を正常に受信できたノードが存在することしか分かりません。

➤ ACK デリミタ

ACK スロットの後には、「ACK デリミタ」が送信されます。これは ACK スロットの終了を表す区切り記号で、1 ビット長のレセシブとなっています。CRC フィールドと同様に、ACK スロットと ACK デリミタを併せて“ACK フィールド”と呼びます。

➤ EOF

データフレームの終わりには、「EOF（End Of Frame）」が送信されます。EOF は 7 ビット長のレセシブとなっています。ビットスタッフィングルールは、SOF～CRC シーケンスの終わりまでの範囲で適用することになっているため、バスアイドル中や EOF は適用外となります。

➤ ITM

データフレームの範囲は SOF～EOF までですが、図 12 では EOF 終了後に「ITM（Intermission）」という表記があります。この ITM はデータフレームには含まれません。ITM は 3 ビット長のレセシブとなっており、この ITM 終了後にバスアイドルとなります。CAN が採用する CSMA/CA では、バスアイドルでないと各ノードは送信することができませんが、後ほど解説する「オーバーロードフレーム」では、ITM において唯一送信可能となっています。

[6] CRC（ Cyclic Redundancy Check ）: 巡回冗長検査といい、データ送信時のデータ破壊をチェックしています。SOF からデータフィールドまでのビットスタッフされていないすべてのビットについてある種の計算処理をして得られる符号を送信ノードと受信ノードで検出し、比較します。

3.1.1.2 拡張フォーマット

次に、「拡張フォーマット」についてです。図 13 から分かるように、標準フォーマットと異なるのは ID～RTR の部分です。以下に、その差異部分を説明します。

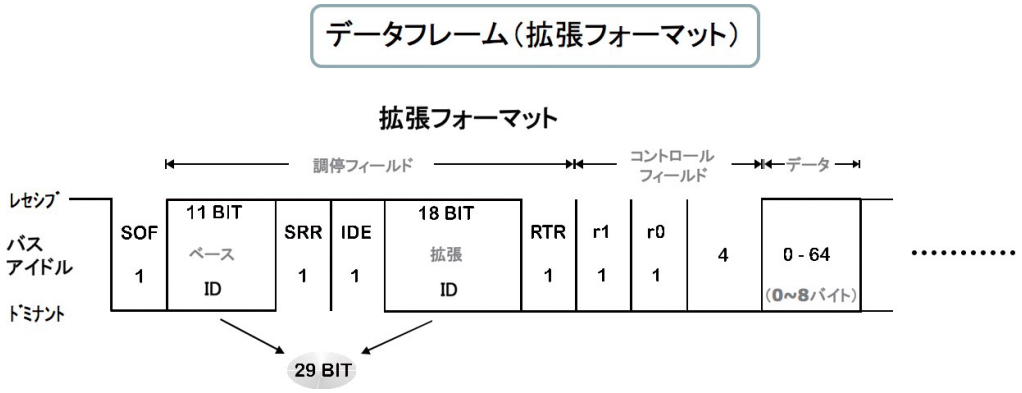


図 13：「拡張フォーマット」のデータフレーム構造

➤ ベース ID

標準フォーマットにおける ID は、拡張フォーマットでは「ベース ID」と呼ばれます。11 ビット長で、この部分は標準フォーマットと同一です。

➤ SRR

ベース ID に続くのは「SRR（Substitute Remote Request Bit）」で、1 ビット長のレセシブとなっています。

➤ IDE

SRR の後には「IDE（Identifier Extension Bit）」が送信されます。これも 1 ビット長のレセシブです。

➤ 拡張 ID

IDE に続いて送信されるのは「拡張 ID」で、18 ビット長です。ベース ID と拡張 ID を併せて 29 ビット長となり、これにより ID を表します。拡張フォーマットの 29 ビット長 ID の範囲は 0x0～0x1FFFFFF となり、536870912 種類（約 5 億 4 千万種）を使用することが可能となります。主に SAE J1939[7]で使用されています。

[7] SAE J1939 は、トラック、バスの制御とネットワーク通信のために開発されたプロトコルで、現在ではトラック、バスに限らず、建設機械や農業機械などのディーゼルエンジン搭載車両や、舶用電子機器などにも幅広く利用されています。

3.1.2 リモートフレーム

「リモートフレーム」は、データフレームの要求に使用され、基本構造はデータフレームからデータフィールドを除いたもの（もしくはデータフレームで DLC を 0、データフィールドが 0 バイトのもの）と同一になっています（図 14）。リモートフレームの ID は要求するデータフレームの ID を設定し、リモートフレームの DLC は要求するデータフレームの DLC を設定します。データフレームと違うのは、RTR がレセシブとなることです。この違いにより、RTR を使用してデータフレームとリモートフレームの識別を行うことが可能となっています。

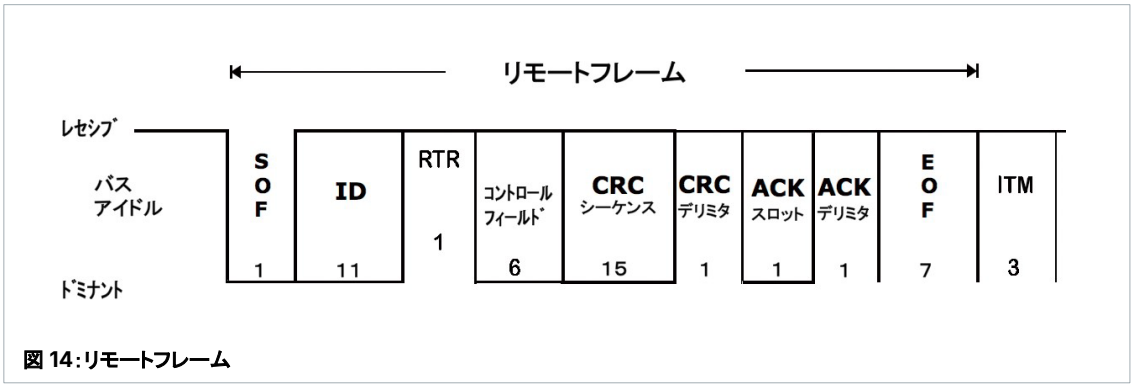


図 14：リモートフレーム

3.1.2.1 データフレームとリモートフレーム

CAN の基本は、データを必要としているノードからリモートフレームを送信し、それに対して該当するノードからデータフレームを返すという形になっています。リモートフレームはデータの要求を行い、データフレームは要求に対してデータの返信を行います（図 15）。

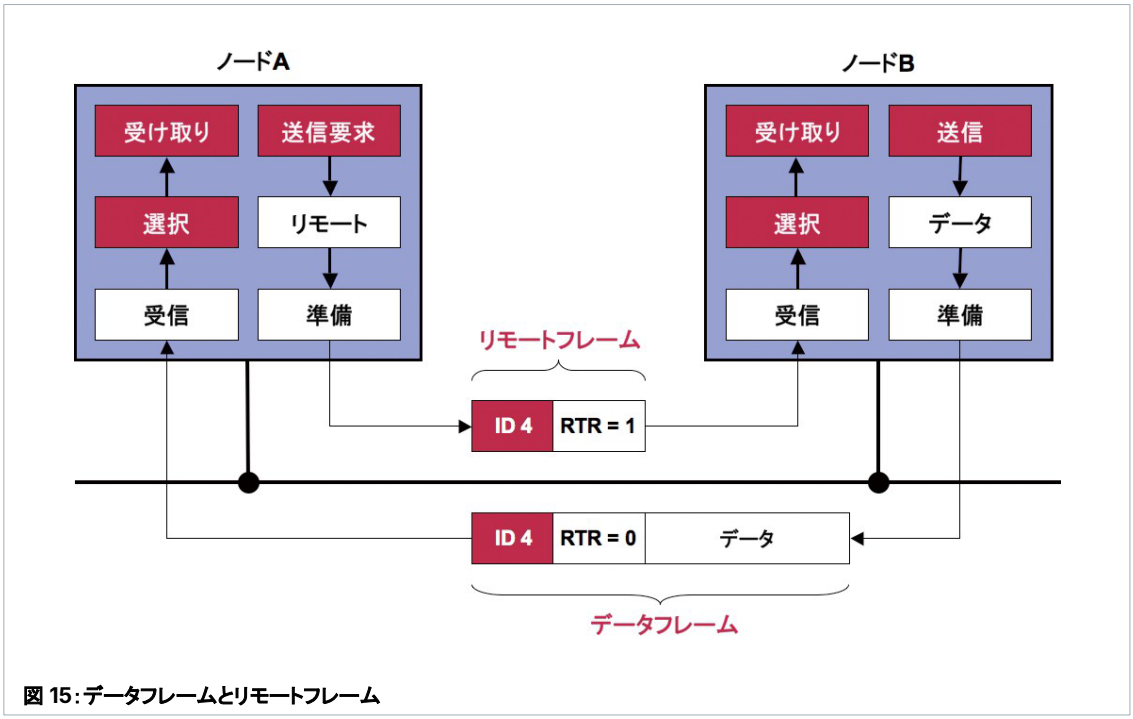


図 15：データフレームとリモートフレーム

この方式を使用すると、ノード内マイコンのリソースを送受信に占有されずに必要な時だけリソースを使用でき、バス占有率も下がります。しかし、頻繁なデータのやり取りにおいては必要なデータフレーム数と同じリモートフレームが必要になるため、かえってバス占有率が上がってしまうということもあります。このため、制御が高度化し、各ノード間の情報交換が非常に多くなってくるにつれ、従来の方式とは違った方法で通信を行うことが考え出されました。

つまり、現在ではリモートフレームはほとんど使われなくなり、データフレームを定期的に各ノードからバスに送信する方式が使われるようになったのです。各ノードから自由にデータフレームを送信し、そのデータフレームを必要とするノードが自由に受信する方式であれば、リモートフレームを使用せずにデータフレームのみで各ノードの情報交換が実現できます（図 16）。

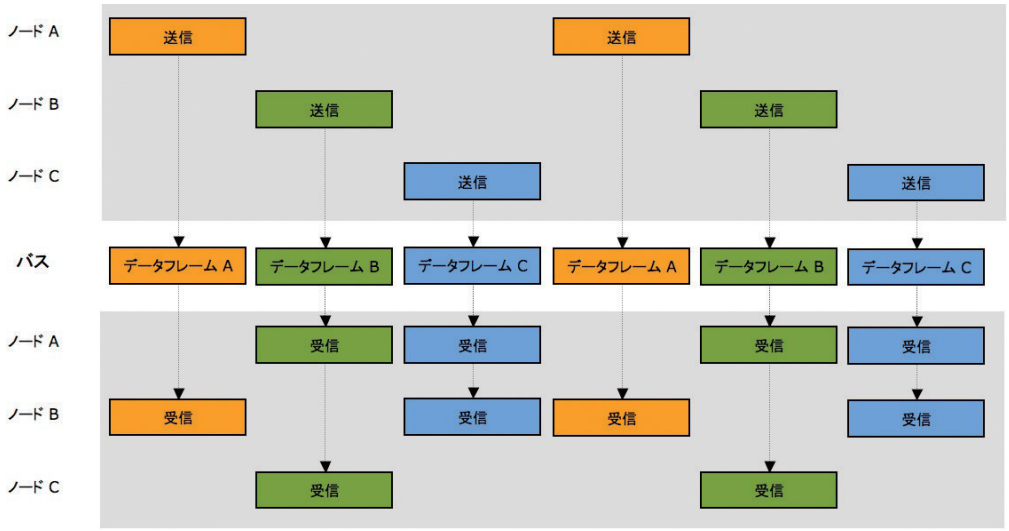


図 16：データフレームのみの送信

3.1.3 オーバーロードフレーム

CAN が開発された頃は、マイコンチップや CAN コントローラの処理能力が低く、処理が正常に行えないことがありました。それを防止するために考え出されたのが、「オーバーロードフレーム」（図 17）です。オーバーロードフレームは、CAN コントローラが前回のフレームの処理をまだ完了していない時に、次 のフレームの開始を遅延させるために用いられます。

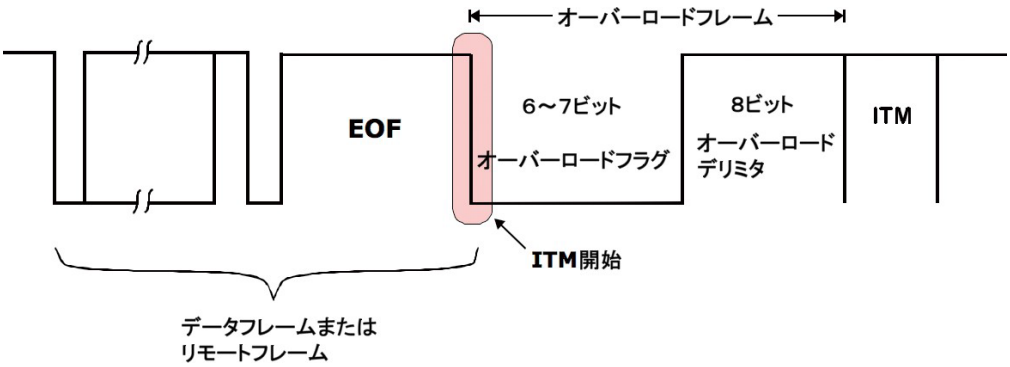


図 17：オーバーロードフレーム

データフレーム受信中に処理が間に合わないノードは、オーバーロードフレームを送信し、次に送信されるデータフレームの送信開始を遅らせることができます。

オーバーロードフレームは、「オーバーロードフラグ」と、「オーバーロードデリミタ」から構成されます。

3.1.3.1 オーバーロードフラグ

「オーバーロードフラグ」は、6 ビットのドミナントから構成され、インターミッション（ITM）の最初の 2 ビット以内から送信が開始されます。オーバーロードフラグを受信したノードは、即座にオーバーロードフラグを送り返します。これにより、最初のオーバーロードフラグとそれにより送信されるオーバーロードフラグが重なる部分が生じ、結果としてオーバーロードフラグのビット長は 7 ビットとなります。

なお、全ノードが同時にオーバーロードフラグを送信した場合は、オーバーロードフラグのビット長は 6 ビットとなります。

3.1.3.2 オーバーロードデリミタ

「オーバーロードデリミタ」は、8 ビットのレセシブから構成され、オーバーロードフレームの区切りを示します。オーバーロードフレームが送信されることにより、バスアイドル開始位置を遅らせることができ、その間に処理が間に合わないノードは処理を行うことができます。

3.1.4 エラーフレーム

「エラーフレーム」（図 18）は、通信中に各種エラーが発生した時に送信されるフレームで、ネットワークに接続されているノードに異常を知らせる役割を果たします。

エラーフレームは、エラーフラグとエラーデリミタから構成されます。これらは、ビットスタッフィングルールに違反、または固定フォーム部分を破壊する形で、直近の送信を中断させます。

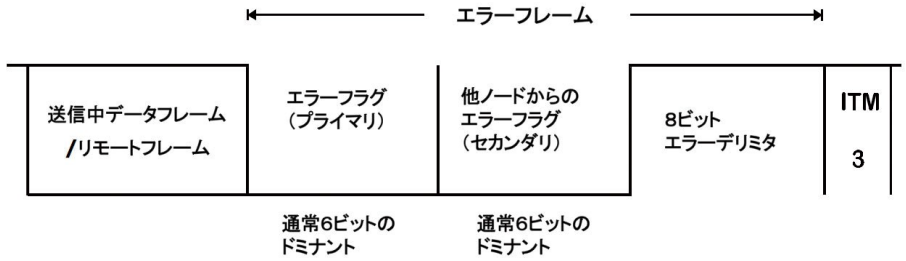


図 18：エラーフレーム

3.1.4.1 エラーフラグ

「エラーフラグ」は、エラー発生を他ノードに知らせる目的で使用されます。エラーフラグは通常 6 ビットのドミナントから構成され、ビットスタッフィングルール違反を発生させます。

このビットスタッフィングルール違反によって、他ノードもエラーフラグを送信します。結果として、プライマリとセカンダリを合わせて 6～12 ビットのドミナントがエラーフラグとしてバスに現れることになります。ドミナント長が 6～12 ビットと変化する理由は、他ノードがどのタイミングでエラーを検出するかによってプライマリとセカンダリが重なる場合があり、それによりセカンダリ部分は 0～6 ビットという扱いになります。

すべての CAN コントローラは、エラー発生状態により内部のエラーカウンターの値を増加させます。これにより、現在ノードがどのような状態であるかを知ることができ、ハードウェア故障など深刻な状態のノードからエラーフラグが連続して送信され、結果として通信が行えなくなることを防ぐ仕組みとなっています。

3.1.4.2 エラーデリミタ

「エラーデリミタ」は、8 ビットのレセシブからなり、エラーフレームを終了させます。データ、またはリモートフレームの終端における 1 ビットの ACK デリミタと 7 ビットの EOF の部分と同じ、8 ビットの連続したレセシブから構成されます。

3.2 通信調停

CAN で採用している CSMA/CA では、バス使用中に他ノードがデータフレームやリモートフレームを送信することはできませんが、実際には、複数ノードから同時に送信されてしまうのを防ぐことはできません。このように、複数ノードから同時にデータフレーム、リモートフレームが送信された場合には「通信調停」を行う必要があります。その時の優先順位は ID（識別子）によって決定され、ID の値が小さい方が優先順位は高くなります。この ID についてはネットワーク設計時に決定することができます。

標準フォーマットを例に、実際の通信調停がどのように行われているか見てみましょう。CAN において通信調停に使用されるのは図 19 の ID と RTR となります。

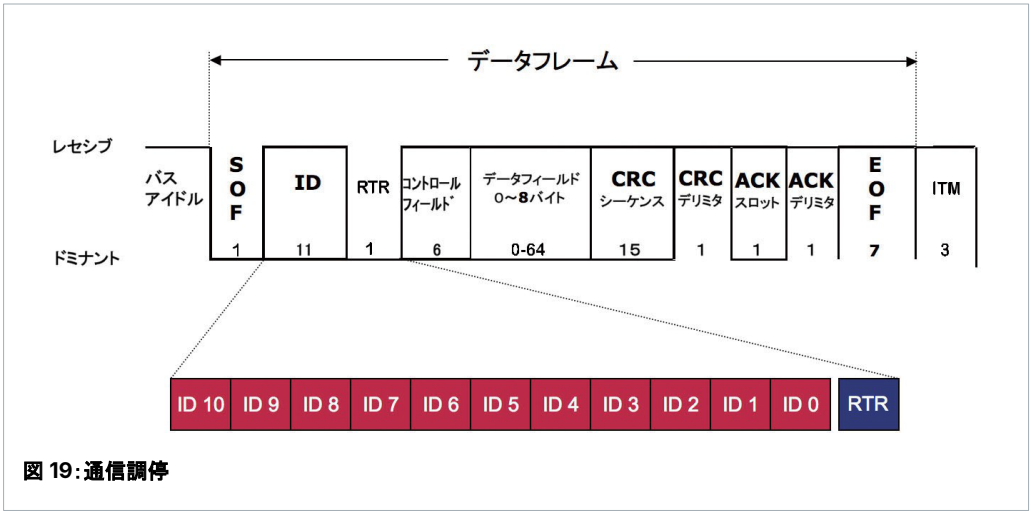


図 19：通信調停

図 20 は、ID 0x653 と 0x65B の 2 台のノードより同時にデータフレームが送信された場合を表しています。

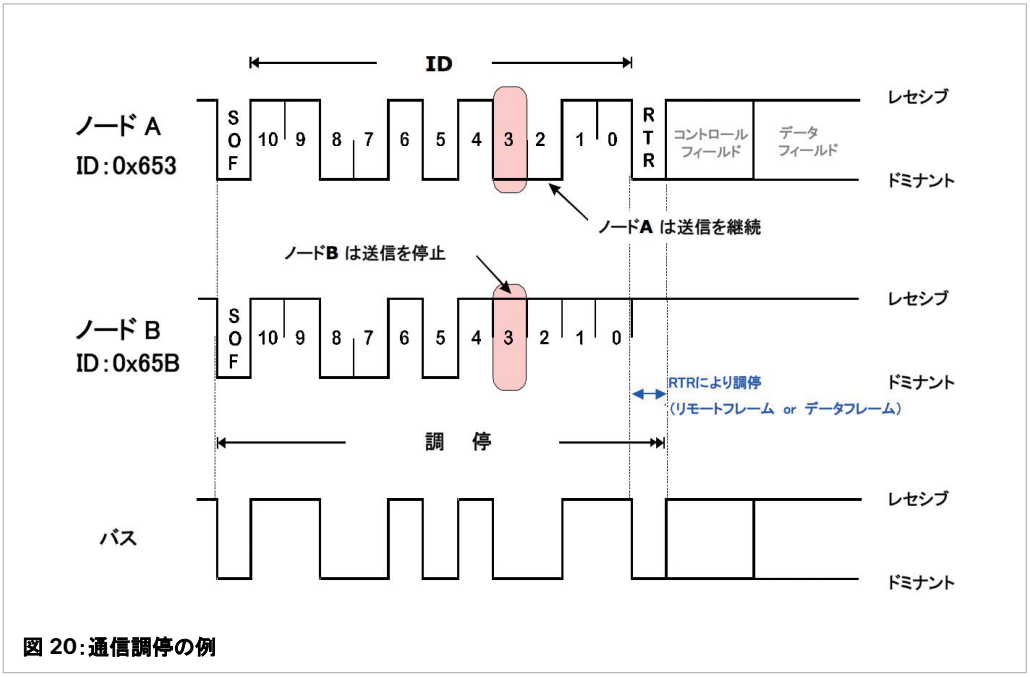


図 20：通信調停の例

同時にデータフレームが送信される場合、開始位置は同一となります。

まず、SOF が送信されますが、SOF は 1 ビット長のドミナントとなっており、バスの状態はドミナントとなります。各ノードは自身が送信したものとバス状態をモニターして比較しますが、SOF においては各ノード自身が送信した内容のままになっていると判断し、送信を継続します。

続いて、ID が 1 ビットずつ送信されますが、送信中の複数ノードの送信ビットが同一（たとえば、複数ノードからレセシブが送信されればバスはレセシブ）となり、送信した内容そのままとなります。

レセシブとドミナントが別々のノードより同時に送信された場合は、ドミナントが優先され、バスの状態はドミナントとなります。この時、レセシブを送信したノードは自身が送信したものとバス状態の違いにより、通信調停に負けたことを検出し、送信を停止します。

複数ノードから同時送信が起こってしまった場合は、他ノードがレセシブ送信を行っている時にドミナントを送信したノードが通信調停に勝つので、優先順位が高い ID は 0x0 であり、ID の値が小さいものほど優先順位が高くなります。

ID の割り当てについては、データフレームのデータフィールドの割り当てと同様に、設計者が自由に割り当てることができますが、通信調停時の優先順位の関係で重要度の高いものは ID の値を小さくするなど、ネットワーク全体を考慮する必要があります。

基本的には ID のみで通信調停が行えますが、なぜ RTR も通信調停に使用するのでしょうか。

それは、同じ ID のデータフレームとリモートフレームが同時に送信された場合、ID のみでは調停を行えないためです。このようなことが起きた場合、ID だけではなく RTR も使用して通信調停を行いますが、データフレームでは RTR はドミナント、リモートフレームでは RTR はレセシブであるため、データフレームが優先されることになります。

3.3 エラー処理

3.3.1 エラー検出

さて、CAN の 4 つのフレームタイプ（「データフレーム」、「リモートフレーム」、「オーバーロードフレーム」、「エラーフレーム」）の構造と使われ方、そしてこれらに関連する通信調停が分かったところで、今度はネットワーク上で発生したエラーが、どのように検出されているかについて見ていきましょう。

まずは図 21 を見ながら、エラー処理に関する用語を解説していきます。

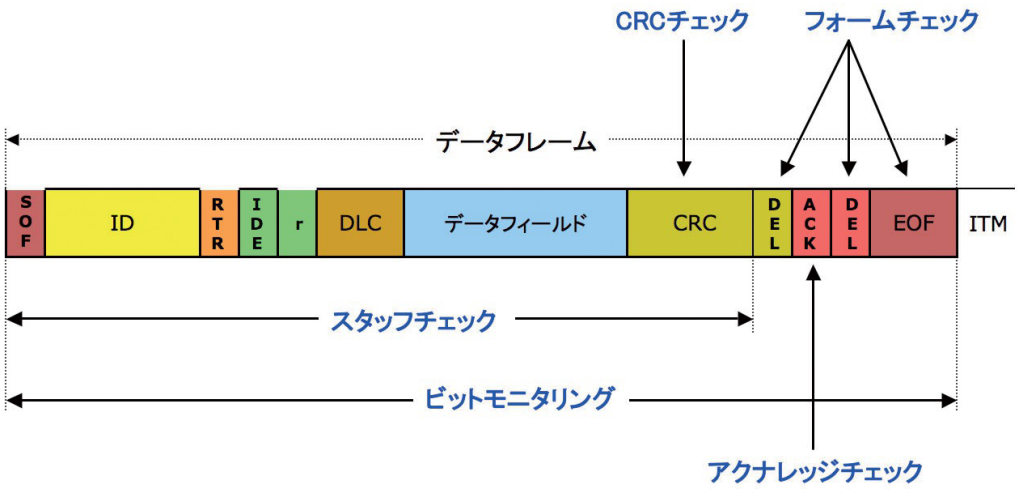


図 21: エラー検出

送信ノードでの監視

➤ ビットモニタリング

送信データとバス上をサンプリングしたデータとの相違をチェックし、相違があればビットエラーとして扱います。ただし、調停フィールドとアクナレッジスロットでは、判定を行いません。

➤ フォームチェック

CRC デリミタ、アクナレッジデリミタ、または EOF は、通常レセシブと決められています。ここでドミナントが検出された時は、フォームエラーとして扱います。

➤ アクナレッジチェック

ネットワークに接続されたノードはデータフレームが流れてくると、受信内容と CRC とを比較します。正常な場合は「アクナレッジ」を送信して、送信ノードに対して正常に通信できているかを知らせます。アクナレッジスロットにおいて該当箇所のバス状態がレセシブのままであった場合、アクナレッジエラーとして扱います。これは、全受信ノードがドミナントを返さなかった時、すなわち全受信ノードが誤ったメッセージを受信したか、ネットワークに他ノードが接続されていない場合に発生します。

受信ノードでの監視

➤ CRC チェック

受信ノードが演算した CRC と、フレームに含まれる CRC の値が合致しなかった時に、CRC エラーとして扱います。

➤ フォームチェック

CRC デリミタ、アクナレッジデリミタ、または EOF は、通常レセシブと決められています。ここでドミナントが検出された時は、フォームエラーとして扱います。

➤ スタッフチェック

ビットスタッフィングルールが守られているかを監視します。ビットスタッフィングエリア内で同一レベルが 6 ビット以上継続した場合は、スタッフエラーとして扱います。

以上 5 種類のエラー検出機構において、ネットワーク上で発生したエラーを発見できるようになっています。これらのエラーが発見された場合、エラーフラグ（6 ビットのドミナント）が即座に送信されます。ただし、CRC エラーの場合は、EOF の最初のビットまでエラーフラグは送信されません。

3.3.2 エラー検出時の動作(1)

図 22 は、送信ノードが送信中にエラーを発見した場合、どのような処理を行うかを表したものです。

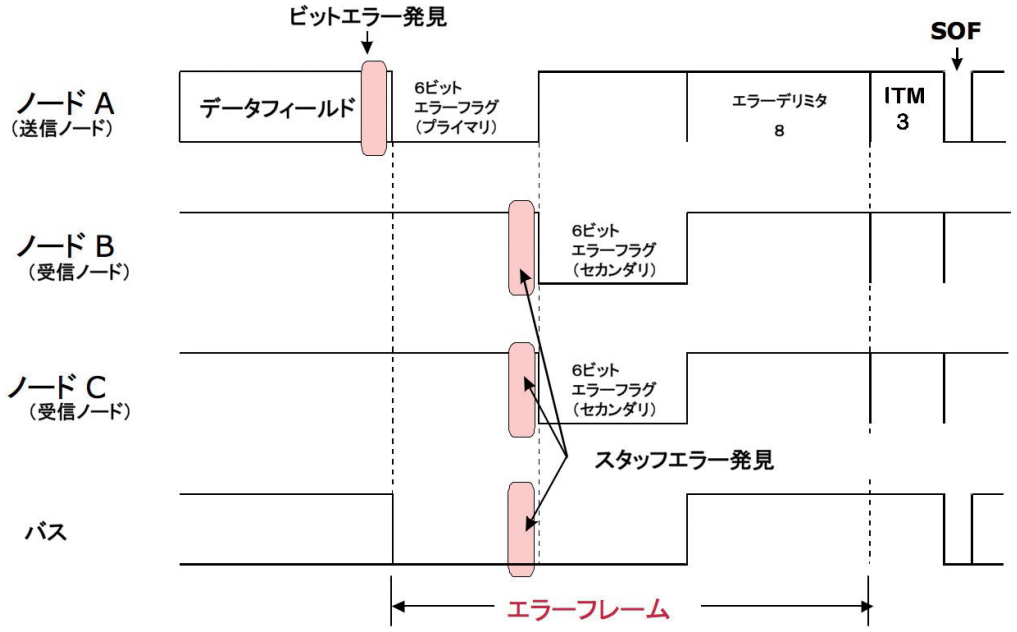


図 22:エラー検出時の動作(1)

送信ノードがデータフィールドにてビットエラーを検出すると、即 6 ビット長ドミナントのエラーフラグ(プライマリ)を送信します。そして、他の ECU はこの 6 ビット長ドミナントのエラーフラグをビットスタッフィングルール違反として検出し、6 ビット長ドミナントのエラーフラグ(セカンダリ)を送信します。

このビットスタッフィングルール違反ですが、エラーフラグ(プライマリ)送信前の状態がドミナントであれば、エラーフラグ(セカンダリ)が送信されるタイミングが前に移動するため、エラーフラグ(プライマリ)とエラーフラグ(セカンダリ)が重なる部分が出てきます。これにより、前述のセカンダリ部分の長さが変化する状態となります。

この後、エラーデリミタが送信され、ITM 終了後に送信ノードは再送信を行います。この場合のエラーフレームは 20 ビットとなり、ITM 分を含めて再送信は 23 ビット後となります。CRC エラーを除き、エラー検出時の動作はこれと同様です。

3.3.3エラー検出時の動作(2)

図 23 は、受信ノードが受信中に CRC エラーを発見した場合、どのような処理を行うかを表したものです。

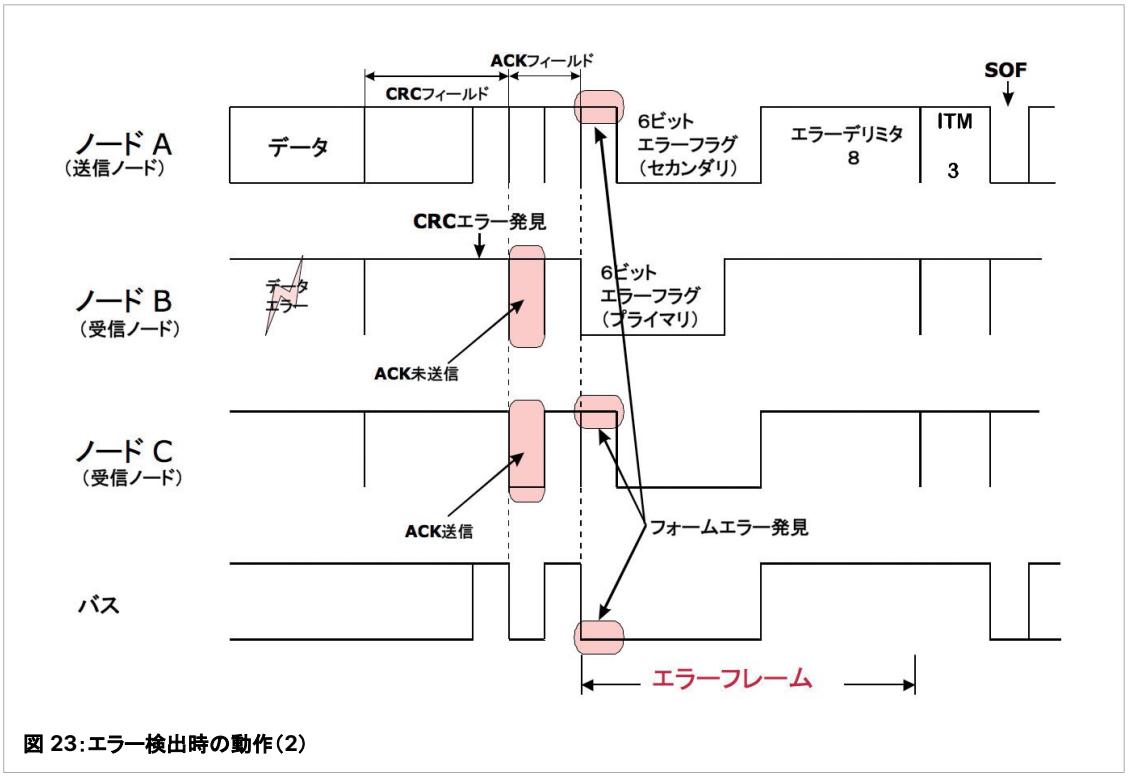


図 23:エラー検出時の動作(2)

受信ノードが CRC フィールドにて CRC エラーを検出すると、エラーフラグの送信は行わず、また ACK スロットでもドミナント送信を行いません(ネガティブ ACK:レセシブ送信 [8])。

しかし、他の受信ノードが CRC を比較して正常に受信を行えた場合、ACK(ドミナント)を送信し、図 23 のような処理となり、送信ノードが送信を継続します。この処理は、CRC エラーとなったノードからのエラーフラグにより、ACK スロットがどのような状態であるかを認識できなくなることを回避するようになっており、これにより送信ノードは全ノードが受信に失敗したかどうかを知ることができます。

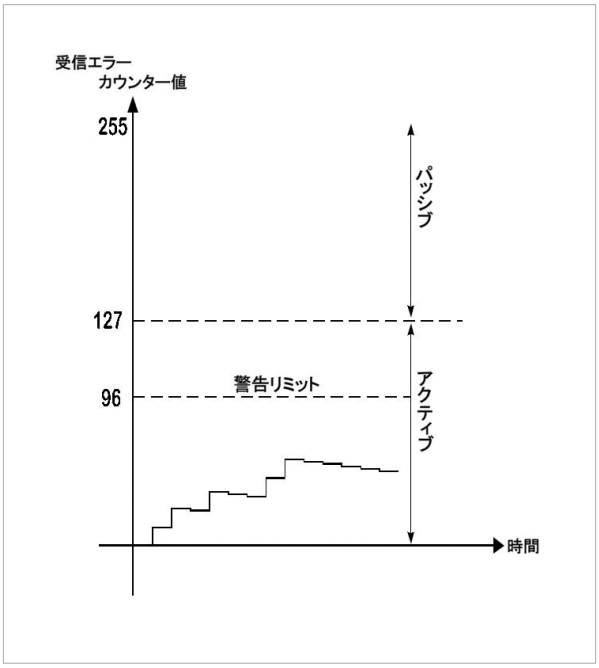
CRC エラー検出時に即エラーフラグを送信できなかった受信ノードは、EOF 開始時にエラーフラグ(プライマリ)を送信します(通常時より 3 ビット分遅延)。この EOF 内のエラーフラグの 1 ビット目がドミナントであることにより、他ノード(送信ノードを含む)はフォームエラーを検出し、エラーフラグ(セカンダリ)を送信します。

この場合のエラーフレームは 15 ビット長となり、ITM 分を含めて再送信は 18 ビット後となります。

[8] すべての受信ノードが ACK スロットでドミナント送信を行わない場合、ACK スロットはレセシブのままとなり、アクナレッジエラーとなります。

3.3.4 受信エラーカウンター

各ノードの CAN コントローラは、それぞれ「受信エラーカウンター」を持っています（図 24）。



受信エラーカウンターは、以下のような動きをします。

> 受信ノードがエラーフラグ（プライマリ）を送信した場合
> ⇒ 受信エラーカウンターに 8 を加算
> 受信ノードがエラーフラグ（セカンダリ）を送信した場合
> ⇒ 受信エラーカウンターに 1 を加算
> 受信ノードがエラーなしで受信を完了した場合
> ⇒ 受信エラーカウンターから 1 を減算

また、各 CAN コントローラは、それぞれの受信エラーカウンターの値によって状態が定義されています。

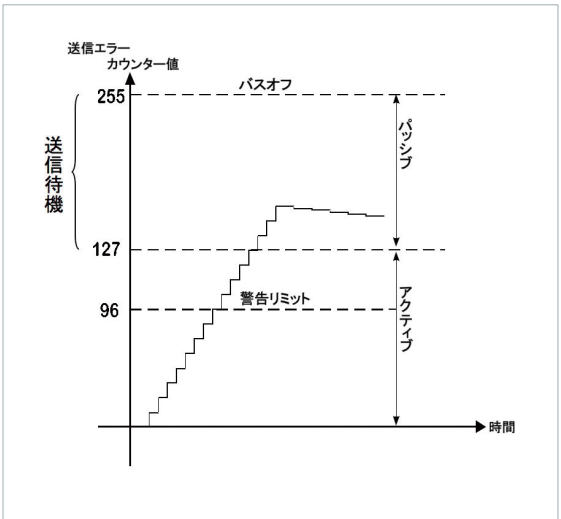
【受信エラーカウンターの値 [9]】

> 0～127：“127”までは、ノードは「 アクティブ」
> 96： ”96”は、バスに重度の障害があることを示す警告リミット
> 127：“127”を超えると、ノードは「パッシブ」に移行

[9] 受信エラーカウンターは、ハードウェアの設定により上限“128”に設定できます。

3.3.5 送信エラーカウンター

各ノードの CAN コントローラは、それぞれ「送信エラーカウンター」を持っています（図 25）。



送信エラーカウンターは、以下のような動きをします。

> 送信ノードがエラーフラグを送信した場合（ビットエラーまたはアクナレッジエラーの場合）

⇒ 送信エラーカウンターに 8 を加算

> 送信ノードからフレームが正常に送信された場合

⇒ 送信エラーカウンターから 1 を減算

> 例外規定の場合

⇒ CAN コントローラの状態が「パッシブ」となった送信ノードがアクナレッジエラーを検出しても、送信エラーカウンターは加算されない

また、各 CAN コントローラは、それぞれの送信エラーカウンターの値によって状態が定義されています。

【送信エラーカウンターの値】

> 0～96：警告リミット以下の「アクティブ」。エラーカウンター値が"96"を超えると、CAN コントローラは警告（エラーフラグのセット、割り込み）を発する

> 97～127：ノードは「アクティブ」。エラーカウンター値がこのエリアにある場合、バスは重度の障害を持っていることになる

> 128～255：ノードは「パッシブ」（ほとんどの CAN コントローラはこの領域への変化についてマイコンに報告しない）。また、送信待機を行う

> 255：エラーカウンターがこの値への到達後、ノードは「バスオフ」となりバスから切り離される（バスへの送信アクセスが不可となる）。マイコンはこの領域への移行を知ることができる

3.3.6「パッシブ」ノードにおける送信待機

エラーカウンターが「パッシブ」となることは、何らかの問題がノードに発生したことを示します。この状態のノードからエラーフラグが繰り返し送信され、バスの占有が行われてしまうと、他ノードの通信が正常に行えなくなってしまいます。これを防止するため、パッシブノードでは送信待機が行われます（図 26）。これは、次のようなメカニズムによって実現されています。

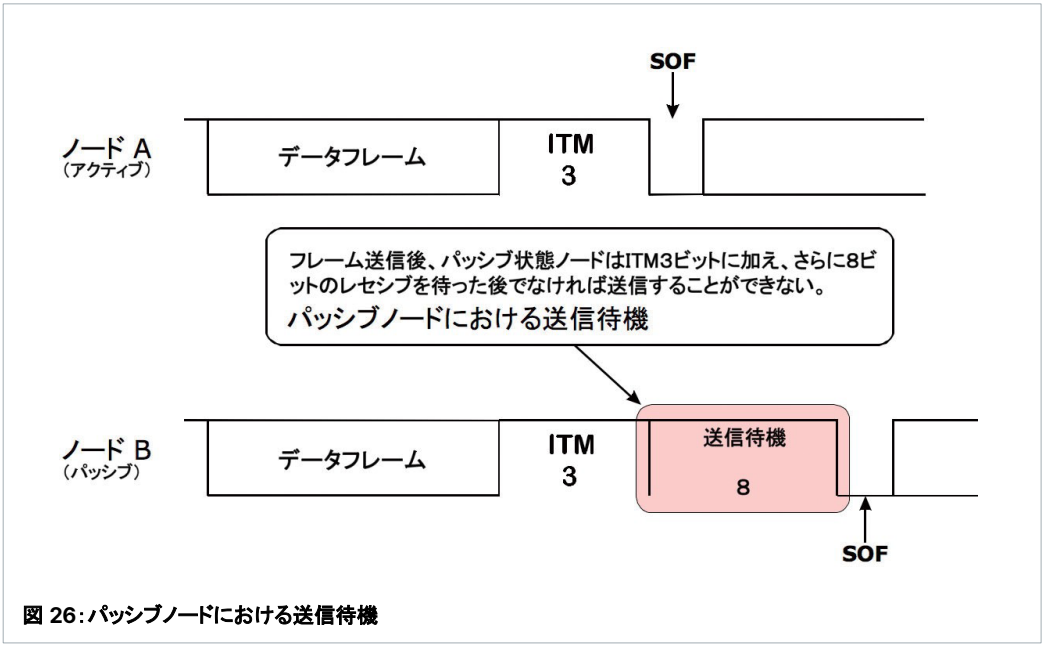


図 26：パッシブノードにおける送信待機

通常では、ITM 終了後にバスアイドル扱いとなってノードは送信を開始できますが、パッシブノードは ITM 終了後に 8 ビットのレセシブを受信した後でないとバスアイドルにはなりません。これにより、正常な他ノードはパッシブノードからの送信に妨害されずにバスへのアクセスを行うことができます。

3.3.7 エラー発生時の CAN コントローラの動作

ここで、エラー発生時の CAN コントローラの動作について解説します。

①ノードがエラーを検出する。
↓
②エラーフラグが送信される（プライマリ）。ただし、CRC エラーの場合は実質 3 ビット遅れ、EOF 開始時より送信。また、このエラーフラグは、ノードが「アクティブ」の場合は 6 ビットのドミナント。ノードが「 パッシブ」の場合は 6 ビットのレセシブとなる。
↓
③正常に送信または受信動作を行っていた他ノードは、このエラーフラグ（プライマリ）をスタッフエラー、またはフォームエラーと認識し、エラーフラグ（セカンダリ）を送信する。
↓
④エラーフレームが送信されることにより、データフレーム（またはリモートフレーム）は EOF まで正常に送信することができず中断されてしまう。また、全ノードはエラーとなるので受信情報は破棄される。
↓
⑤エラーカウンターの値が規定どおり加算される。
↓
⑥送信ノードが再送信を行う。

4. CAN の物理層

さて、ここまで CAN の基礎知識とプロトコルについて解説してきました。次は、CAN 通信における物理層について簡単に解説していきます。これまでの制御解説と違い、ハードウェア関連の解説が主になりますが、CAN の実装や試験を行う際に開発作業をスムーズに行うためにも、ぜひ把握しておきましょう。

4.1 バスへの接続

まず、各ノード（CAN プロトコルコントローラ、CAN コントローラ）がどのように CAN バスへ接続されるかについて解説します。

図 27 は High Speed CAN の CAN バスへの接続例です。このように、CAN コントローラは CAN トランシーバを介して CAN バスと接続されます。High Speed CAN / Low Speed CAN などの物理層での規格の差異は、CAN トランシーバにより吸収されます。これは、どのような物理層であっても CAN コントローラは影響を受けず、CAN プロトコルの仕組みを共通で使用できるということを意味します。

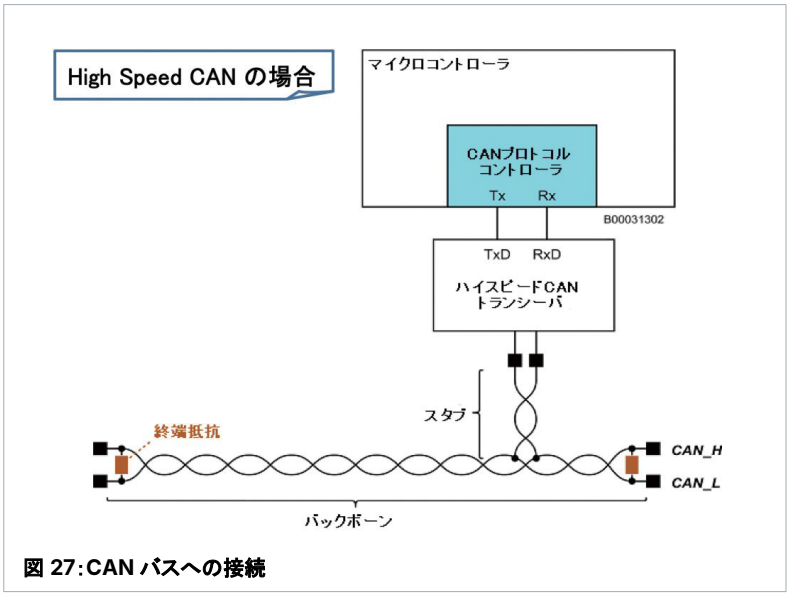


図 27：CAN バスへの接続

【用語解説】

- マイクロコントローラ（マイコン）：

  送信データ、受信データの処理を行う。ノードの他の処理も行っている。

- CAN プロトコルコントローラ：

  CAN プロトコルの機能（ビットスタッフィング、通信調停、エラーハンドリング、CRC チェックなど）を実現。フィルター搭載。

- CAN トランシーバ：

  バス送信電圧の発生、調整、動作電流の確保、配線の保護を行う。

- 終端抵抗：

  反射電圧の抑制、バスレベルの調整用途で使用される。

4.2 High Speed CAN / Low Speed CAN

CAN は、通信スピード（ボーレート）125Kbps を境に「High Speed CAN」と「Low Speed CAN」の 2 つに分かれます。これらは用途に応じて使い分けられ、最大通信速度と物理層が異なります。

4.2.1 High Speed CAN

「High Speed CAN」は、通信速度が最大 1Mbps まで可能なため、主に、高速通信が要求されるパワートレイン系に使用されます。通信には 2 本線を使用し、電圧差にてドミナント／レセシブを判断することで外来ノイズの影響を受けにくくしています。また、CAN_H と CAN_L の電圧差は、規格では 2V となっており、120Ω の終端抵抗 ×2 により 30mA の電流で CAN トランシーバを駆動しています（図 28）。

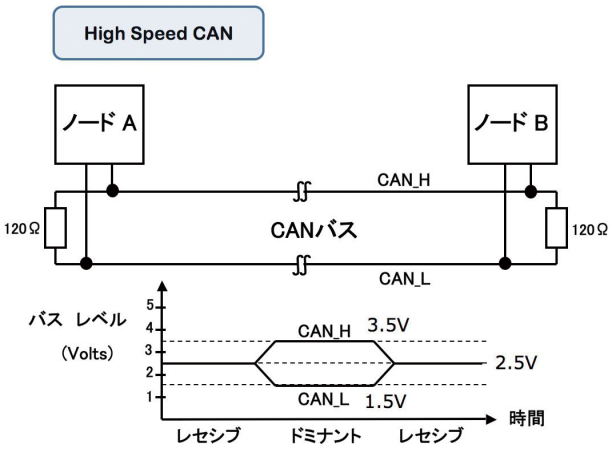


図 28：High Speed CAN

4.2.2 Low Speed CAN

「Low Speed CAN」は、通信速度が最大 125Kbps まで可能で、主に、それほど高速通信が要求されないボディー系や快適装備系に使用されます。通信には 2 本線を使用し、通常は High Speed CAN のように電圧差にて通信を行います。ただし、バスに問題が発生した場合はシングルワイヤーモードに切り替わり、単線で通信を行うことが可能です。これは、バスが自動車内でも外側部分に配置される場合があるため、フォールトトレランスが考慮されているからです（図 29）。

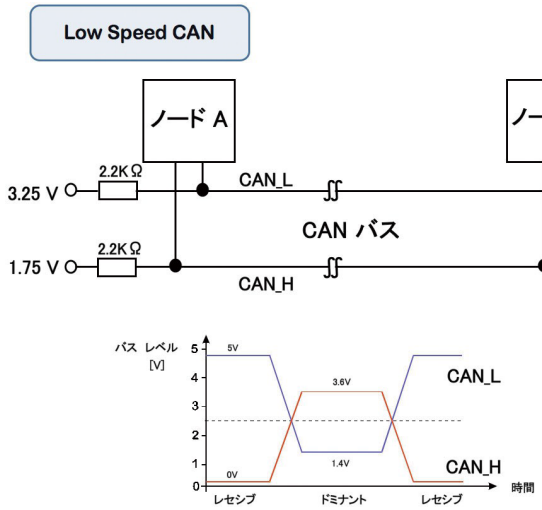


図 29：Low Speed CAN

ここまで、CAN を知るための基本的知識として、「CAN の特長」、「仕組み」、「物理層」について解説してきました。これらの内容が CAN についてのすべてを網羅している訳ではありませんが、少なくとも CAN の基本を理解いただけたかと思います（CAN について、さらに詳細をお知りになりたい方は、ISO 11898 として公開されている CAN プロトコル仕様などをご参照ください）。

では次に、CAN をベースに拡張された CAN FD プロトコルについて解説していきます。早速、その背景から見てまいりましょう。

5. CAN FD とは？

5.1 CAN FD 導入の背景と概要

「CAN FD」は CAN のプロトコルを拡張して、より多くのデータを高速に送るための通信プロトコルです。

自動車の安全運転システムのさらなる高度化や自動運転機能、セキュリティ対策などを実現するには、通信速度が CAN では十分ではないことが予想されています。また、ECU のソフトウェアの肥大化により、ECU の ROM 容量が増え、CAN 通信によるリプログラミングでは ECU の書換えに非常に時間が掛かってしまいます。

より高速な車載ネットワークが必要となってきているなかで登場したのが、CAN よりも高速な車載ネットワークの一つ、CAN FD です。これまでは、機能追加によって CAN バスのバス負荷が高くなり遅延時間の増大や帯域不足になった場合、ネットワークを分割することによって帯域不足を解消していましたが、CAN FD に置き換えれば、ネットワークを分割する必要がなくなります。

CAN FD（CAN with Flexible Data rate）はその名のとおり、データレートが可変です。また、CAN をベースに、1 つのフレームに載せられるデータ長を従来の 8 バイトから 64 バイトに拡張しており、すなわち 64 バイトまでのデータを送信できます（プロトコルの詳細はこの後で説明します）。診断・リプログラミングの用途では、データレート 5Mbps 程度での使用が検討され、制御系などでは 2Mbps 程度での使用が検討されています。CAN FD のトポロジーはスター型、バス型、ポイントツーポイントなどが検討されています。

6. CAN FD プロトコル

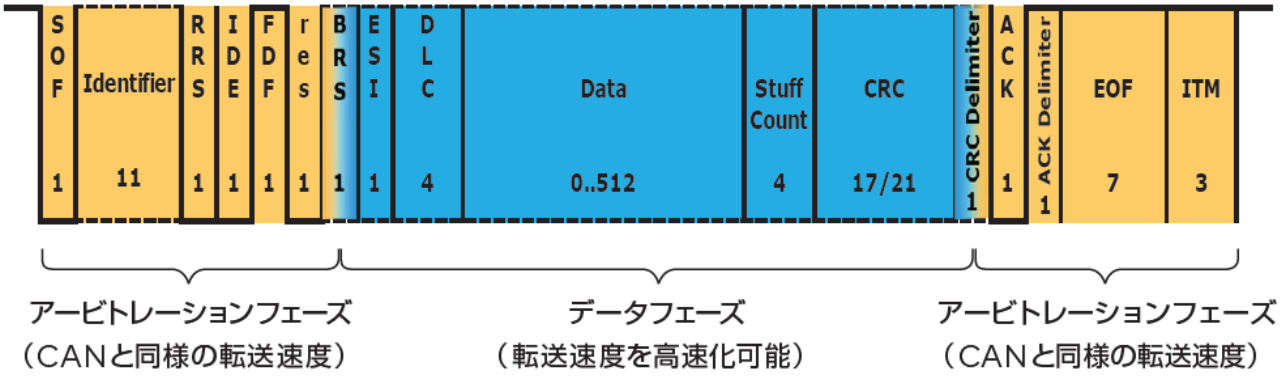
では、CAN FD で使用されるフレームタイプと CAN FD フレームの領域、各フレームの領域のフィールド構造について CAN との違いを説明していきます。

6.1 フレームタイプ

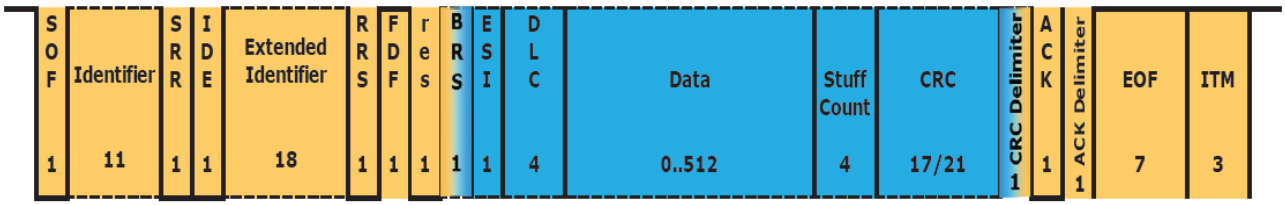
CAN FD はデータフレームのみ定義されており、CAN で定義されているリモートフレームはありません。データフィールドが存在しないリモートフレームはビットレートの切替えが不要となるため、CAN FD 用にリモートフレームを定義する必要がありません。データフレームは CAN と同様に「標準フォーマット（11 ビット ID）」と「拡張フォーマット（29 ビット ID）」の 2 種類の形式があります。転送速度の高速化は BRS ビットから CRC Delimiter ビット間のみ可能です。

以降の図では、CAN と同様の転送速度の部分をオレンジ色、転送速度を高速化可能な部分を青色で表示します。

標準フォーマットの CAN FD フレーム

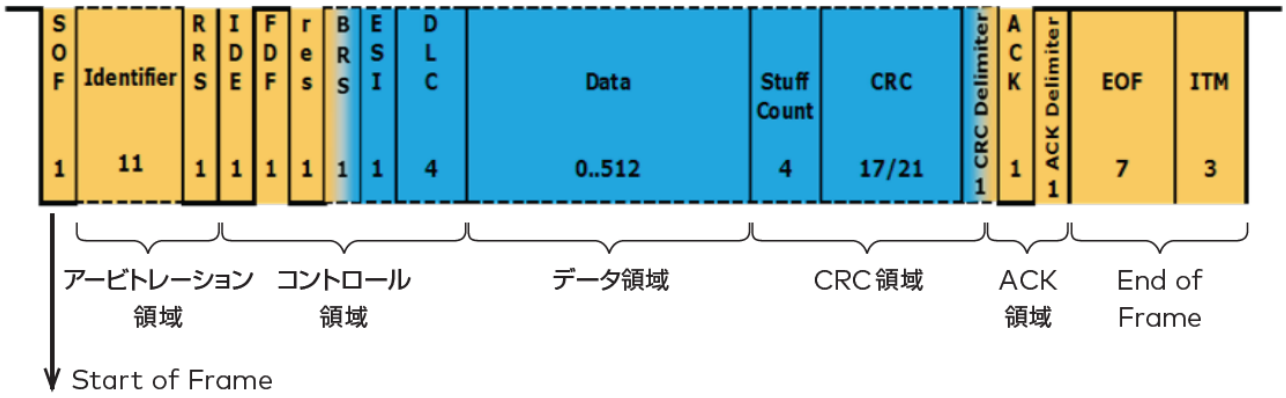


拡張フォーマットの CAN FD フレーム



6.2 CAN FD フレーム領域

CAN FD フレームは Start of Frame（SOF）と、アービトレーション領域、コントロール領域、データ領域、CRC 領域、ACK 領域、End of Frame（EOF）の 7 つのビット領域で構成されています。



6.2.1 SOF（Start of Frame）

ノードからフレームが送信されるとき、最初に送信される部分はフレームの“開始”を表すためにドミナント状態とします。この部分を「SOF（ Start Of Frame）」と呼びます。SOF は CAN と同様に 1 ビットの“ドミナント”ビットです。SOF がバスアイドルのレセシブ（1）からドミナント（0）へ変化することにより、受信側ノードは同期を行うことができます。

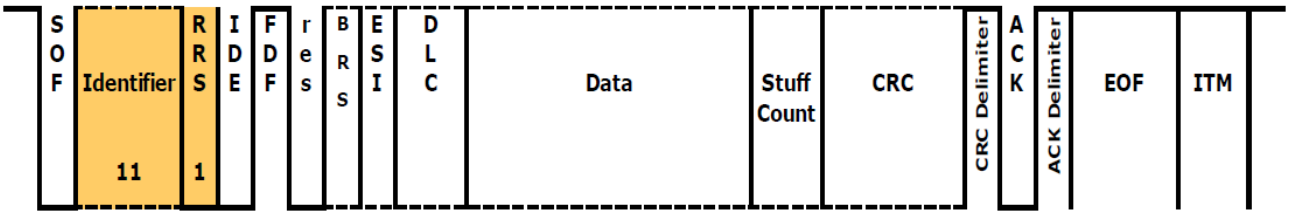
CAN FD フレーム

CAN フレーム

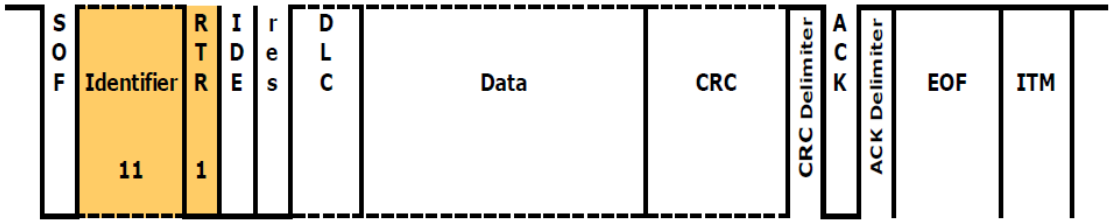
6.2.2 アービトレーション領域

CAN FD のアービトレーション領域は Identifier と RRS ビットで構成されています。Identifier は CAN と同様に、データ内容や送信ノードを識別するために使用され、通信調停の優先順位を決定する働きもしています。

CAN FD フレーム



CAN フレーム



> RRS ビット（Remote Request Substitution）

CAN FD ではリモートフレームが無いので、CAN で使用されていた RTR ビットは RRS ビットに置き換えられ、ドミナントに固定されます。

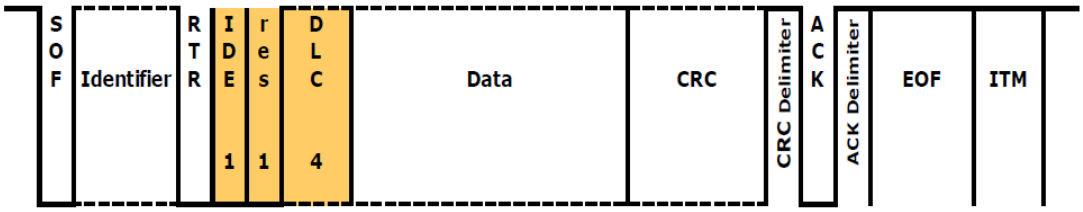
6.2.3コントロール領域

CAN FD のコントロール領域は IDE ビット、FDF ビット、res ビット、BRS ビット、ESI ビットで構成されています。

CAN FD では、新たに FDF ビット、BRS ビット、ESI ビットが追加されました。IDE ビットは CAN と同様であり、res ビットは予約ビットになります。

CAN FD フレーム

CAN フレーム



> **FDF（FD Format Indicator）**

FDFビットがドミナントの場合は CAN のデータフレームであり、レセシブの場合は CAN FD のフレームになります。

> **BRS（Bit Rate Switch）**

BRS ビットがレセシブの場合、送信ノードが BRS ビットのサンプリングポイントで高速な転送速度のクロックモードに切り替えることを意味し、応答する全受信ノードも同様にクロックのモードの切替えを行う必要があります。CRC Delimiter のサンプリングポイントで全ノードが調停ボーレートに戻ります。つまり、全 CAN FD ノードは 2 種類のボーレートを持つことになります。

> **ESI（Error State Indicator）**

CAN フレームでは送信ノード用に自身のエラー状態をレポートする手段がありませんでしたが、CAN FD の ESI ビットによって全ノードは現在の送信ノードのエラー状態を知ることができます。

送信ノードの状態が Error Passive の場合はレセシブになり、Error Active の場合はドミナントになります。

> データ長コード（DLC）

DLC は何バイトのデータが送信されるかを表します。CAN、CAN FD ともに 4 ビット構成です。CAN の DLC の設定範囲は 0～8 バイトですが、CAN FD では DLC=8 以上の場合、以下のようにデー タバイト数が定義されます。

DLC	CAN（バイト）	CAN FD（バイト）
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7

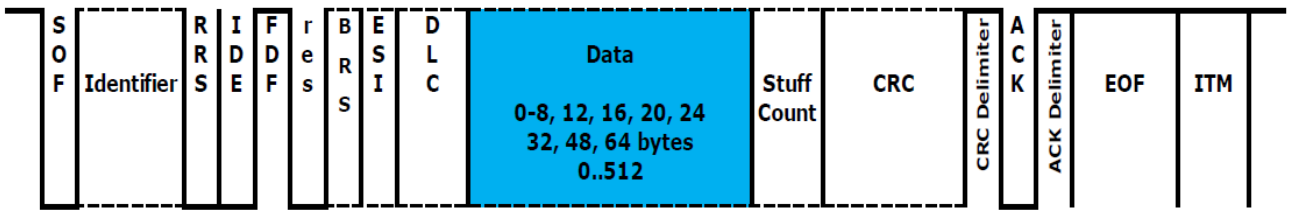
DLC	CAN（バイト）	CAN FD（バイト）
8	8	8
9	8	12
10	8	16
11	8	20
12	8	24
13	8	32
14	8	48
15	8	64

6.2.4 データ領域

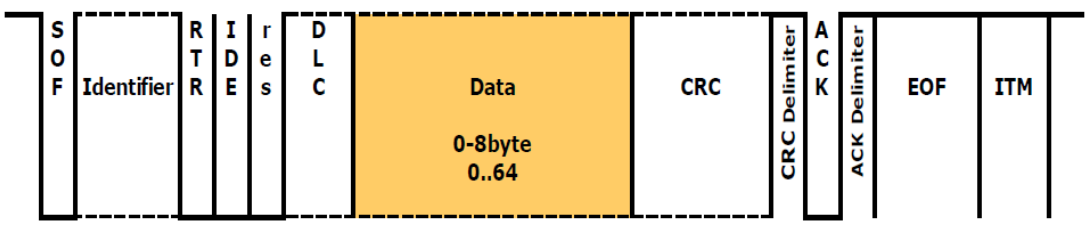
データ領域は Data のみで構成されています。データ長は CAN の場合、0-8 バイトになりますが、CAN FD では 0-8、12、16、20、24、32、48、64 バイトとなります。

CAN と同様にデータバイトは最上位ビット（MSB）から送信されます。

CAN FD フレーム



CAN フレーム



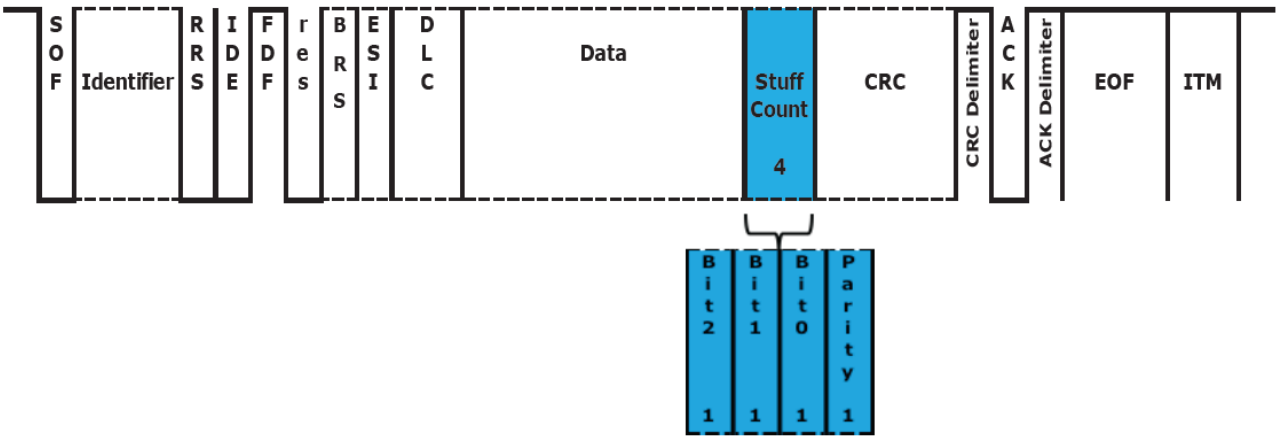
6.2.5 CRC 領域

CAN FD の CRC 領域は Stuff Count、CRC、CRC Delimiter で構成されています。CAN と同様に SOF からデータ領域の値を演算してその結果を比較することで正常に受信できたかの判断を行います。CRC の後には CRC Delimiter が送信されます。これは CRC の終了を表す区切り記号で 1 ビットのレセシブとなっています。しかし、CAN と CAN FD では演算方法が異なり、CAN FD では新しく Stuff Count が追加されています。

CAN FD フレーム

CAN フレーム

> **Stuff Count（CAN FD）**



CAN の CRC 計算方法ではスタッフビットを使用しませんが、CAN FD ではデータ領域までのスタッフビットを計算に使用します。Stuff Count は次の 2 つの要素で構成されます。

1. CRC 領域より前のスタッフビットの数を 8 で割った余り（Stuff bit count modulo 8）をグレイコード化した値（Bit0-2）
2. グレイコード化した値のパリティー（偶数パリティー）

> CRC

大きくなったデータ領域の伝送品質を維持するために、多くの CRC ビットが必要とされます。CAN では 15 ビットでしたが、CAN FD では以下のようになります。

0～16 バイトのデータ領域 → 17 ビット

17～64 バイトのデータ領域 → 21 ビット

CAN FD の CRC Delimiter は常に 1 ビットで送信されますが、ノード間の位相のずれを考慮し、受信側で最大 2 ビット時間を許容します。CAN FD フレームのデータ領域（高速化可能な領域）は CRC Delimiter の最初の 1 ビットのサンプリングポイントまでになります。

> ビットスタッフィングルール

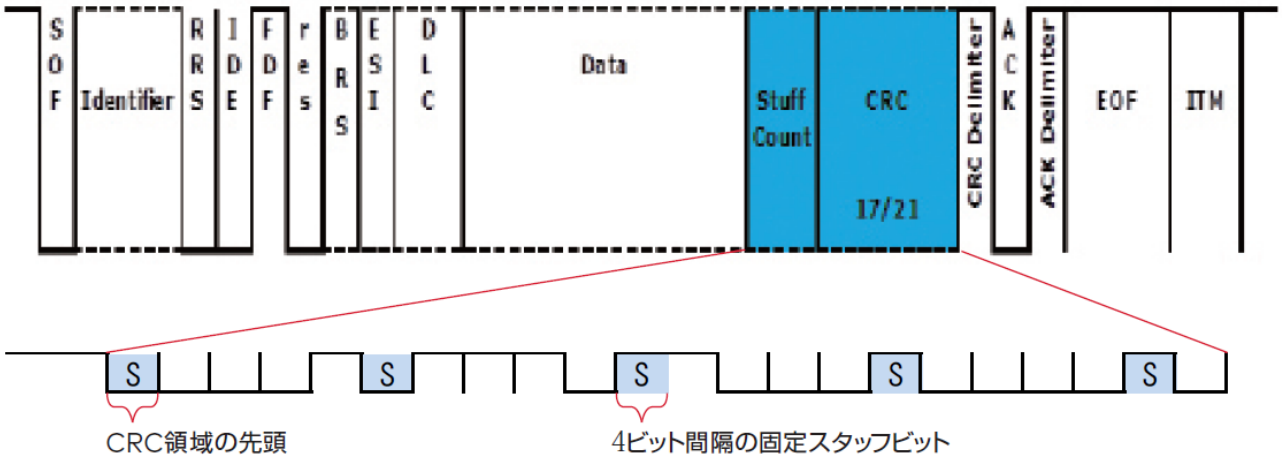
CAN と同様に SOF からデータ領域の末端までビットスタッフィングされ、配置されたスタッフビット数はグレイコード化されてパリティービットで保護されます。（Stuff Count）

CRC 領域では固定されたビット位置に固定スタッフビットを配置します。固定スタッフビットの値は、その前のビットの値の逆になります（Stuff Count の前にも固定スタッフビットが配置されます）。

・CRC 領域の先頭

・4 ビット間隔の固定スタッフビット

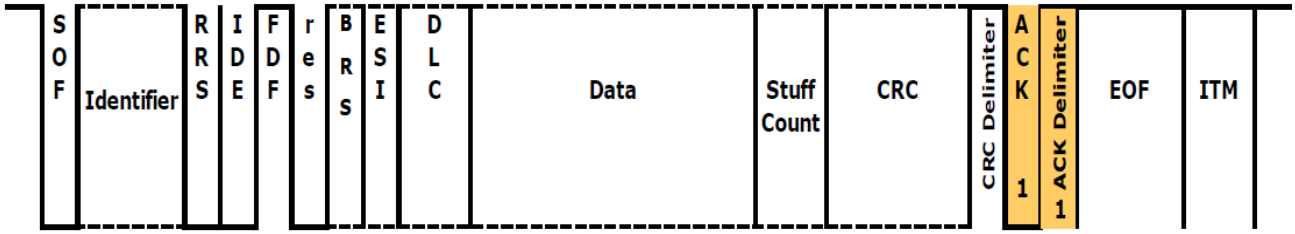
CRC 領域ビットスタッフィング



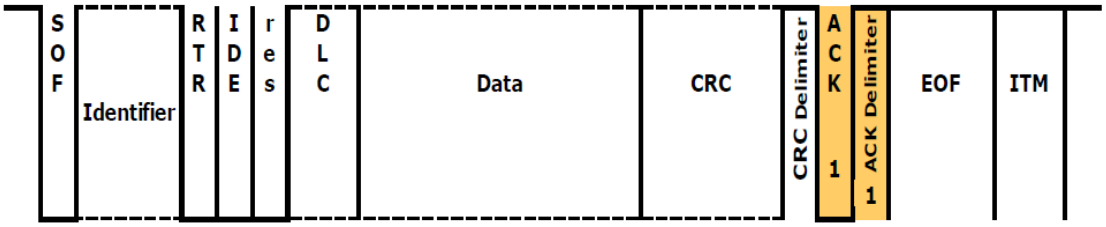
6.2.6 ACK 領域

CAN FD の ACK 領域は ACK と ACK Delimiter で構成されています。CAN では ACK は 1 ビットでしたが、CAN FD の受信ノードは最大 2 ビット時間まで有効な ACK として認識します。追加の 1 ビット時間は、高速なデータ領域から低速なアービトレーション領域へのクロック切替え時に発生する可能性があるトランシーバの位相のずれおよびバスの伝播遅延の補完に使用されます。ACK の後には、ACK Delimiter が送信されます。これは ACK の終了を表す区切り記号で、1 ビットのレセシブとなっています。

CAN FD フレーム



CAN フレーム

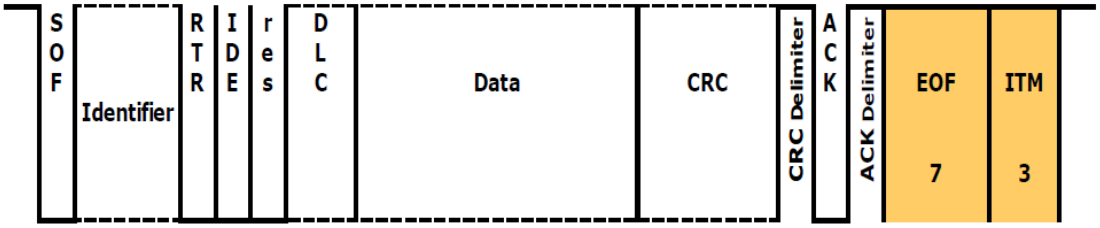


6.2.7 EOF（End of Frame）

データフレームの終わりには、「EOF（End Of Frame）」が送信されます。CAN と同様に EOF は 7 ビット長のレセシブとなっています。

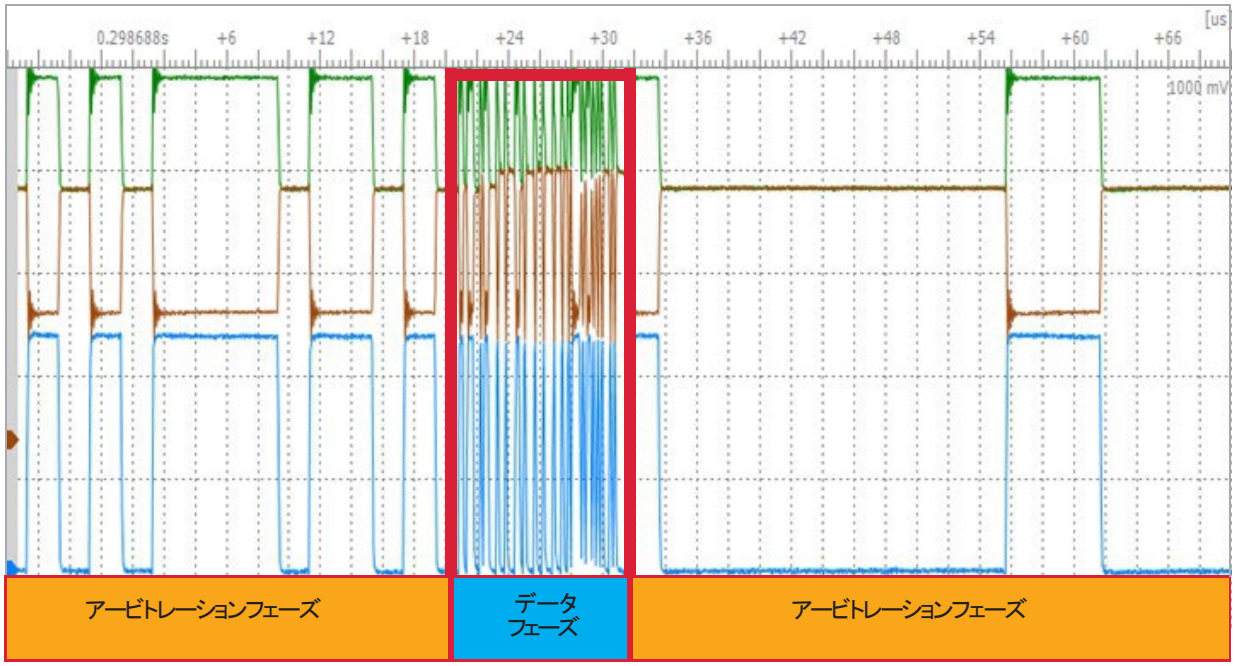
CAN FD フレーム

CAN フレーム



6.3 CAN FD フレームの波形

下図は CAN FD フレームの波形を表示させた図になります。データフェーズである赤枠の箇所が高速化されていることがわかります。



7. CAN FD の標準化

CAN FD は CAN のデータリンク層および物理層が規定されている ISO 11898 を改訂する形で標準化が進められています。

主にデータリンク層の仕様が規定されている ISO 11898-1 は 2015 年に CAN FD の仕様を含む改訂版がすでに発行されています。

また、主に物理層の仕様が規定されている ISO 11898-2 では CAN FD 要件の追加とともに ISO 11898-5/6 の統合が行われ、低電力モード、ウェイクアップ、パーシャルネットワーキングの仕様も追加されます。

8. FAQ

Q1: 同一ネットワーク上に CAN ノードと CAN FD ノードを混在させることは可能ですか?

A: CAN FD 対応ノードは CAN フレームと CAN FD フレーム両方の送受信を行うことができますが、CAN にのみ対応している CAN ノードは CAN FD フレームを受信すると受信エラーとなります(表 1)。

		受信側	
		CAN対応ノード	CAN FD対応ノード
送信側	CAN対応ノード	OK	OK
	CANFD 対応ノード	Error	OK

表 1

Q2: CAN FD の最大転送速度は?

A: CAN FD の最大転送速度は規定されていません。トランシーバ等によりますが、転送速度に関しては現状、制御系などの EMC 要件が特に厳しい領域では 2Mbps 程度、リプログラミングなどの領域では 5Mbps 程度とされています。

Q3: CAN FD を導入するために必要なことは?

A: CAN FD 対応のトランシーバおよびコントローラの導入やネットワークデータベースのアップデートなどシステム全体をアップデートする必要があります。

また、通信の高速化に伴い、課題となるリンギング対策用の回路(RSC: Ringing Suppression Circuitry)の導入などが検討されています。

あとがき

以上、CAN および CAN FD を知るための基本的知識として解説させていただきましたが、いかがでしたでしょうか。本資料で解説した基礎知識が、車載ネットワーク開発に携わる、もしくは今後携わっていくエンジニアの方々の一助となれば幸いです。[参照]

[参照] CAN Newsletter magazine「Ringing suppression in CAN FD networks」
http://can-newsletter.org/engineering/engineering-miscellaneous/160309_ringing-suppression- in-canfd-networks_denso/

9. 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