



はじめてのCAN / CAN FD

ビギナー向け資料「はじめてシリーズ」

発行元：ベクター・ジャパン株式会社

ベクター・ジャパン株式会社の書面による許可なしに、本書の内容を転載、複製、複写することを禁じます。

記述されている内容は予告なく変更されることがあります。

目次

はじめに	03
1. CANとは?	03
歴史	03
適用範囲	04
自動車におけるCANの使用例	05
2. CANの基礎知識	07
特長	07
ライン型構造	07
マルチ・マスター方式	08
CSMA/CA	09
IDを使用したメッセージ・アドレッシング	09
耐ノイズ性に優れた物理層	10
優れたエラー検出機構	10
データの一貫性	10
キーワード	11
ドミナントとレセシブ	11
シグナルコーディング	11
通信速度	12
同期	12
ビットスタッフィングルール	13
3. CANプロトコル	14
フレームタイプ	14
(1) データフレーム	14
標準フォーマット	14
拡張フォーマット	18
(2) リモートフレーム	19
データフレームとリモートフレーム	19
(3) オーバーロードフレーム	21
オーバーロードフラグ	21
オーバーロードデリミタ	21
(4) エラーフレーム	22
エラーフラグ	22
エラーデリミタ	22
通信調停	23
エラー処理	25
エラー検出	25
エラー検出時の動作(1)	27
エラー検出時の動作(2)	28
受信エラーカウンター	29
送信エラーカウンター	30
「パッシブ」ノードにおける送信待機	31
エラー発生時のCANコントローラの動作	32

目次

4. CANの物理層	33
バスへの接続	33
High Speed CAN / Low Speed CAN	34
High Speed CAN	34
Low Speed CAN	35
5. CAN FDとは？	36
CAN FD導入の背景と概要	36
6. CAN FDプロトコル	37
フレームタイプ	37
CAN FDフレーム領域	38
SOF(Start of Frame)	38
アービトレーション領域	39
コントロール領域	40
データ領域	41
CRC領域	42
ACK領域	44
EOF(End of Frame)	45
CAN FDフレームの波形	45
7. CAN FDの標準化	46
8. FAQ	46
あとがき	47

はじめてのCAN/CAN FD

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？

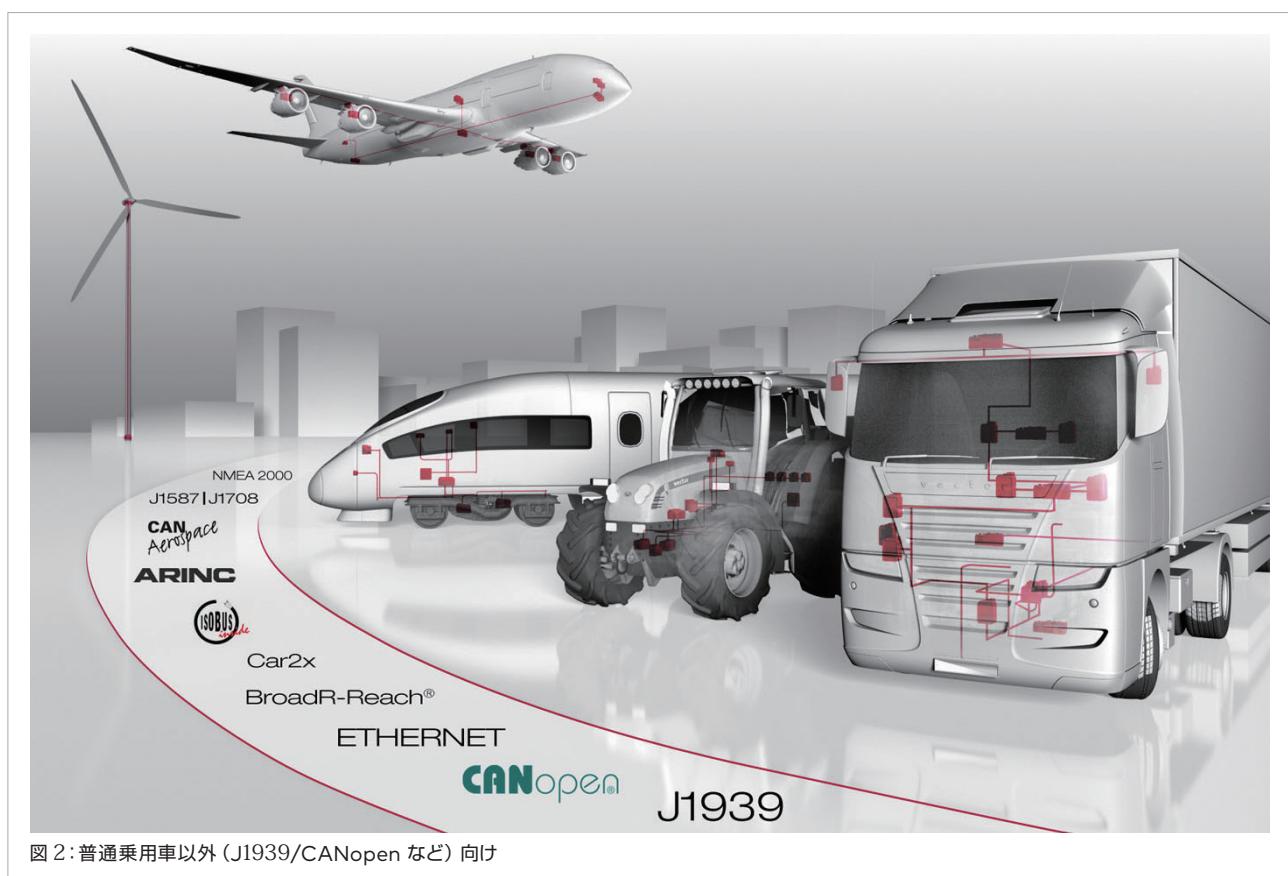
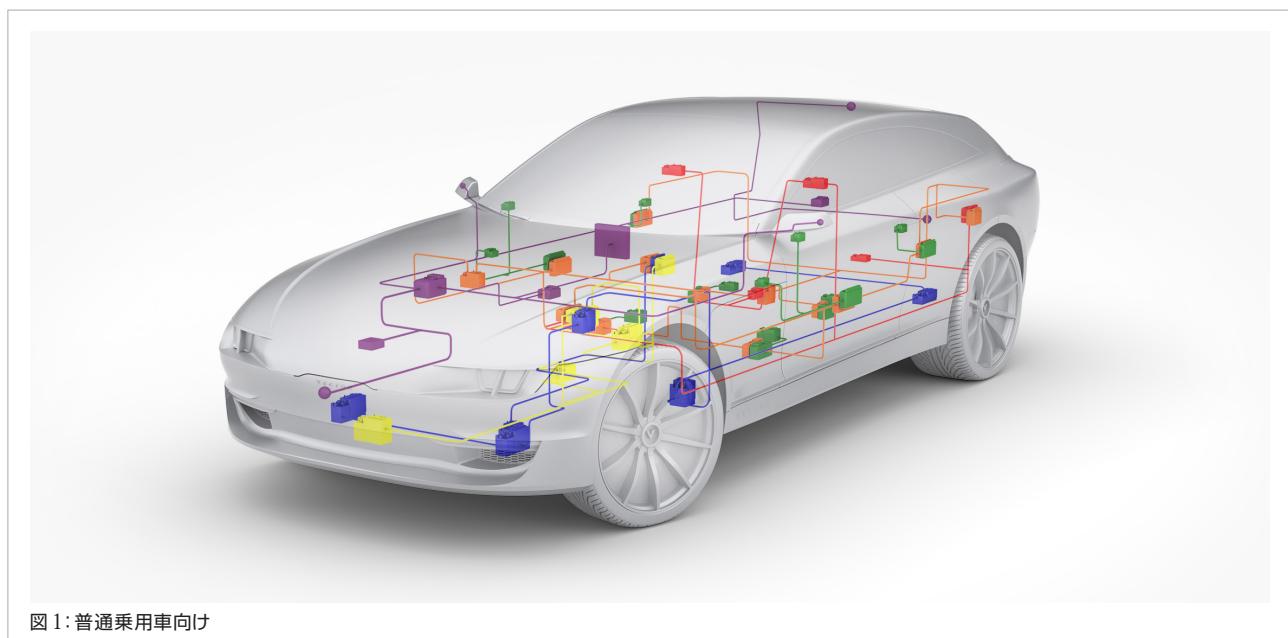
歴史

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。

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)。



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}を使用していましたが、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。

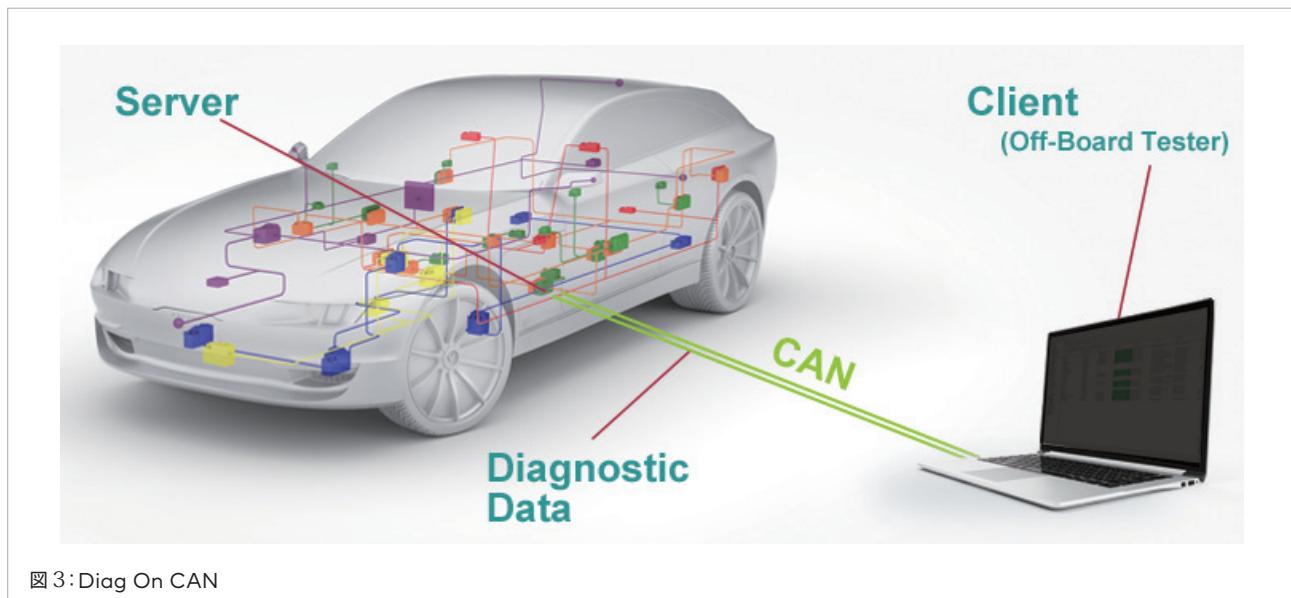
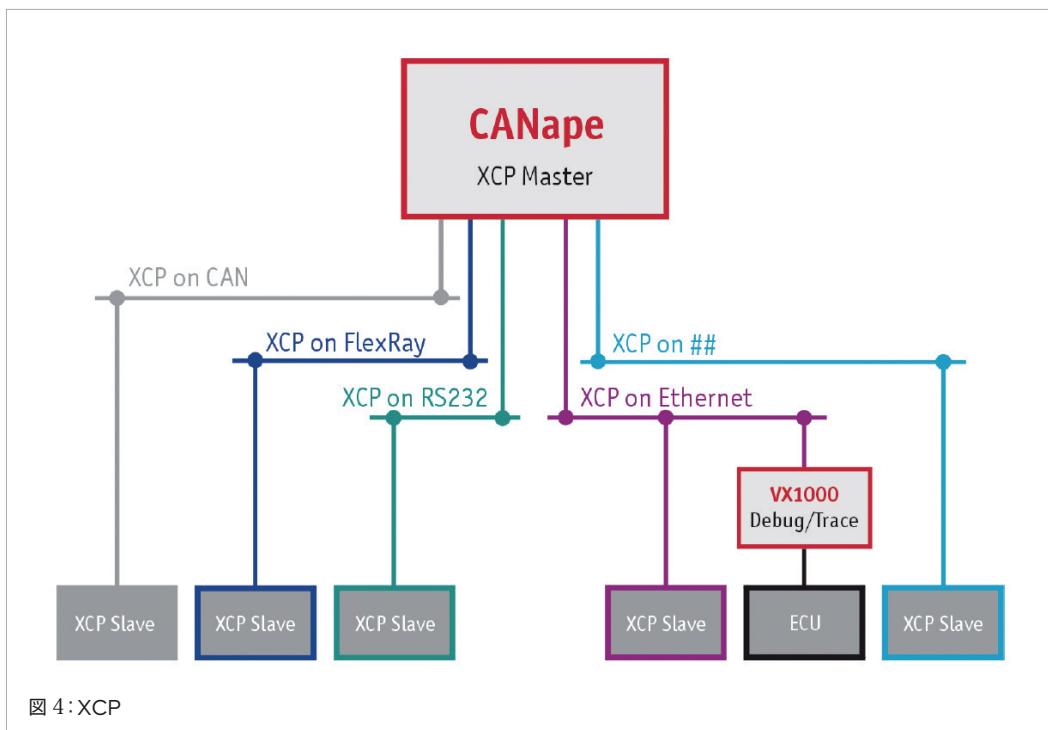


図3:Diag On CAN

※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。



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

特長

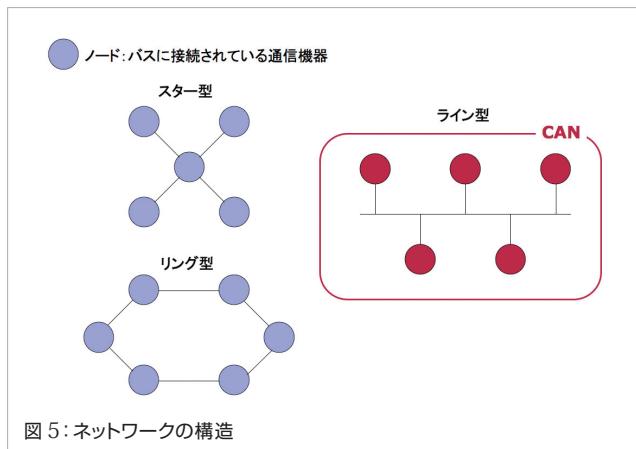
CANには、次のような特長があります。これらはどのような目的で採用され、どのようなことを行っているのでしょうか。以下に、その特長を一つずつ見ていくことにしましょう。

- > 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

●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。

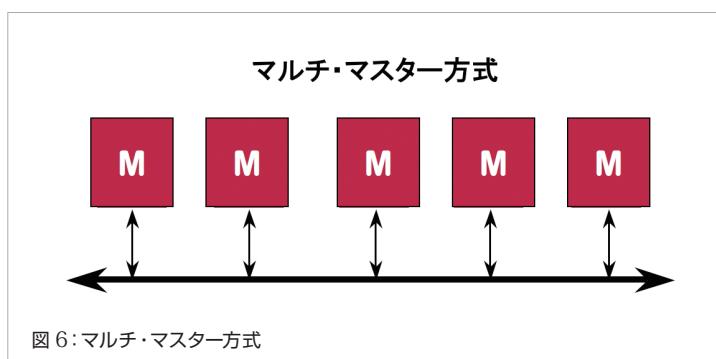


●マルチ・マスター方式

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事が可能で、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。



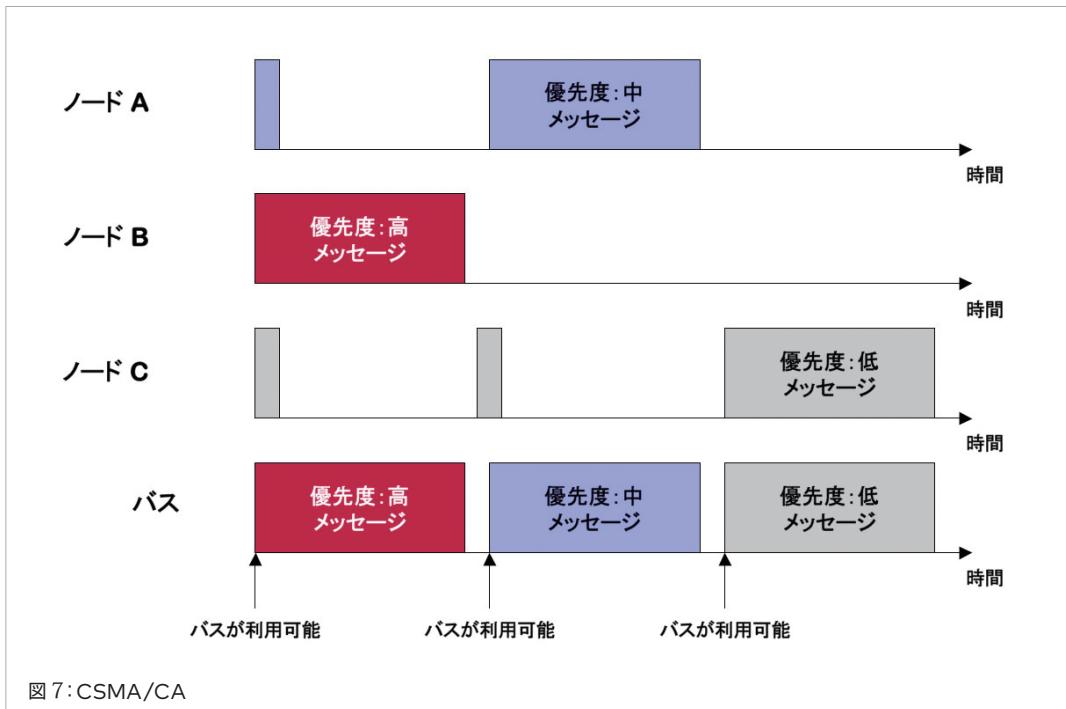
※4 バスアクセスとは、バスへ情報を送信したりすること。

● 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。



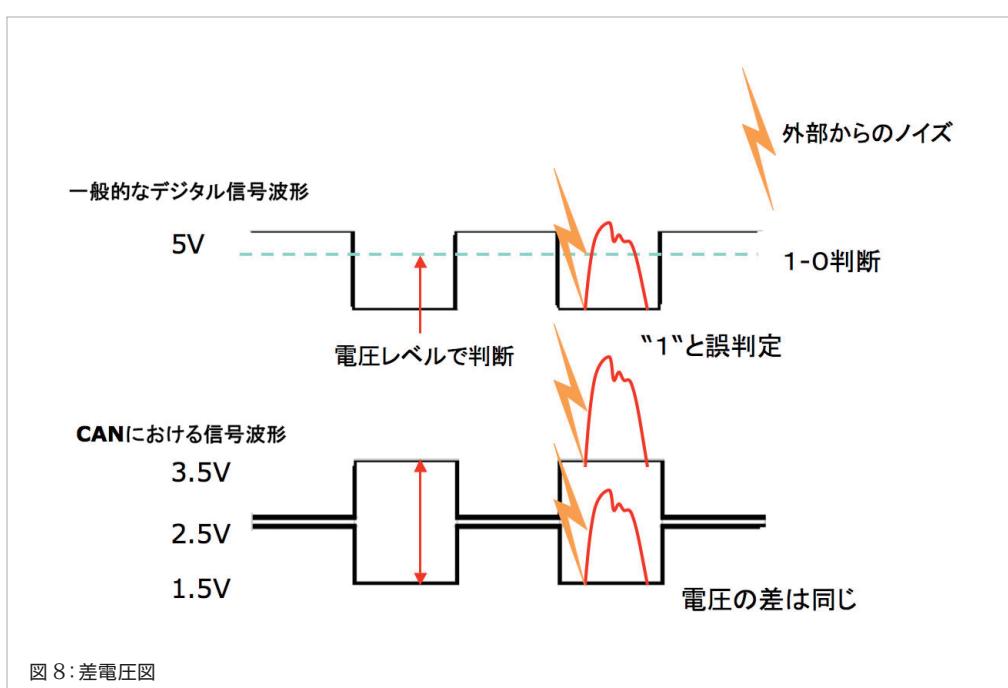
● IDを使用したメッセージ・アドレッシング

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を附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。この方式を探ることで、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。

●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できます。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、電圧の差があるかないかに影響を受けにくい仕組みになっています（図8）。



●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、違った場合にはエラー検出することができます。

●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。

キーワード

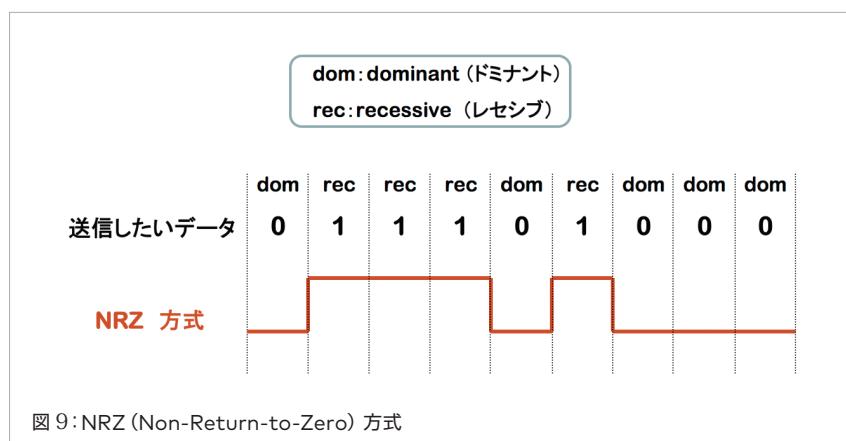
次に、CANの基本を理解するために欠かせないキーワードについて、解説していきます。

●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”的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”的意味で、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。

●シグナルコーディング

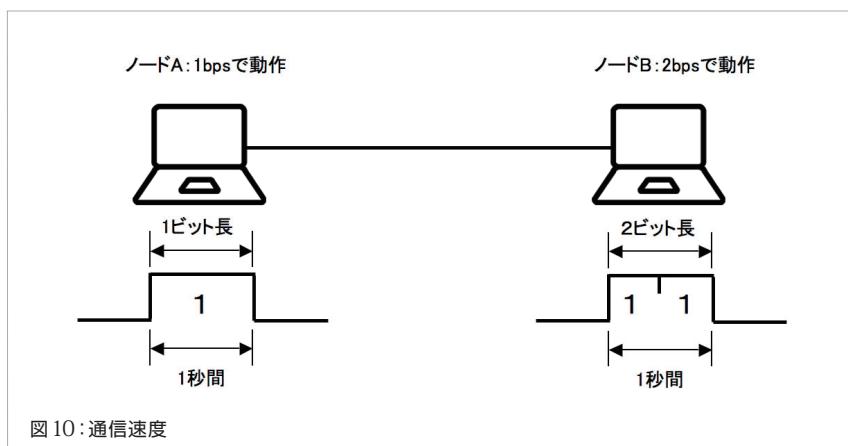
CANではNRZ(Non-Return-to-Zero)方式で、送信したいデータを変換して送信しています(図9)。この方式は比較的シンプルに変換が行えるため、さまざまな通信プロトコルで使用されています。NRZ方式では、たとえば送信するデータが“0001100”的場合、連続する“0”的部分は“0”的まま、連続する“1”的部分は“1”的まとなり、ドミナント状態やレセシブ状態が連続することになります。状態が連続することによるデメリットもありますが、これを解消するための仕組みが用意されています。



●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。



●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。

●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所以で、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（“スタッフィット”）を1つ挿入する仕組みのことです（図11）。

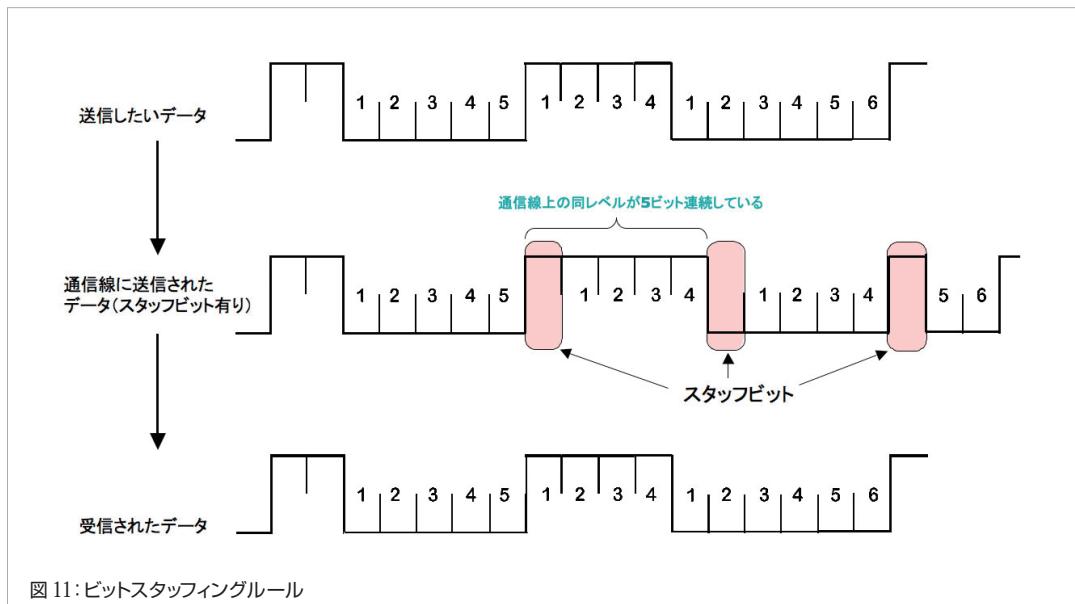


図11:ビットスタッフィングルール

たとえば、“000000111111”と連続する場合、実際には“00000101111101”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ィット）。 “0000011111”と連続する場合は、“000001111101”です。挿入されたスタッフィ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ィ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ィ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ィ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

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。

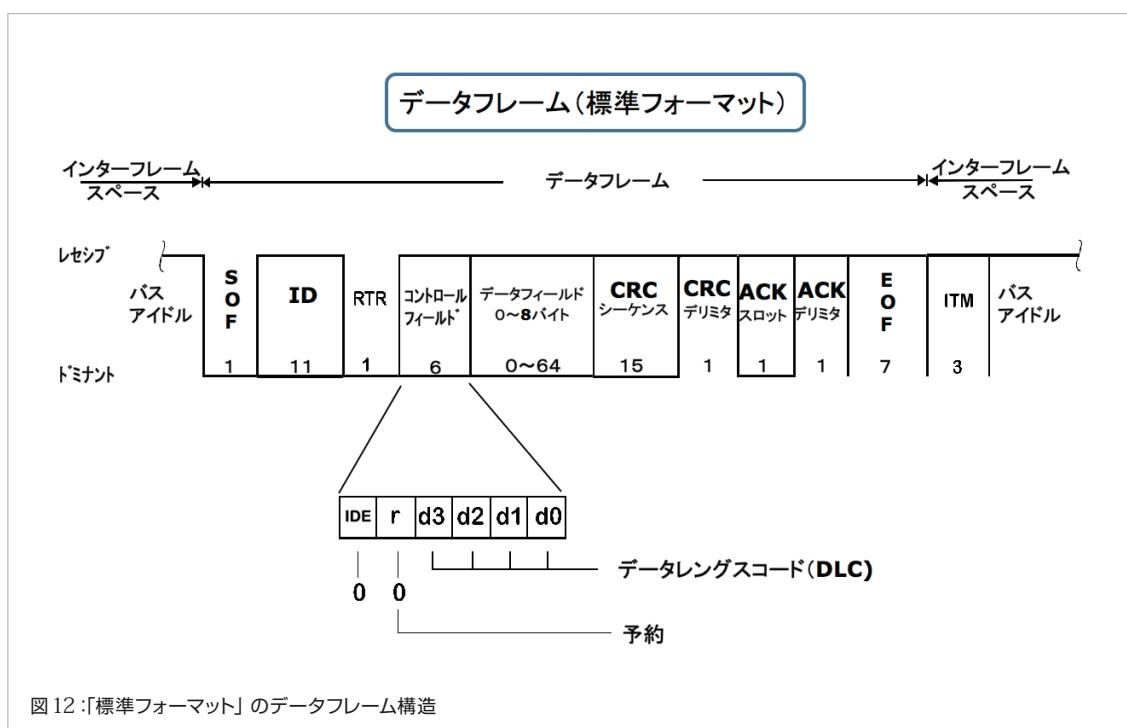
- (1) データフレーム
- (2) リモートフレーム
- (3) オーバーロードフレーム
- (4) エラーフレーム

(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。

●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。



※5 通信されるデジタル信号のまとめを「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ジングスコード (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。

▶ データレンジングスコード

「データレンジングスコード (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。

※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。

► EOF

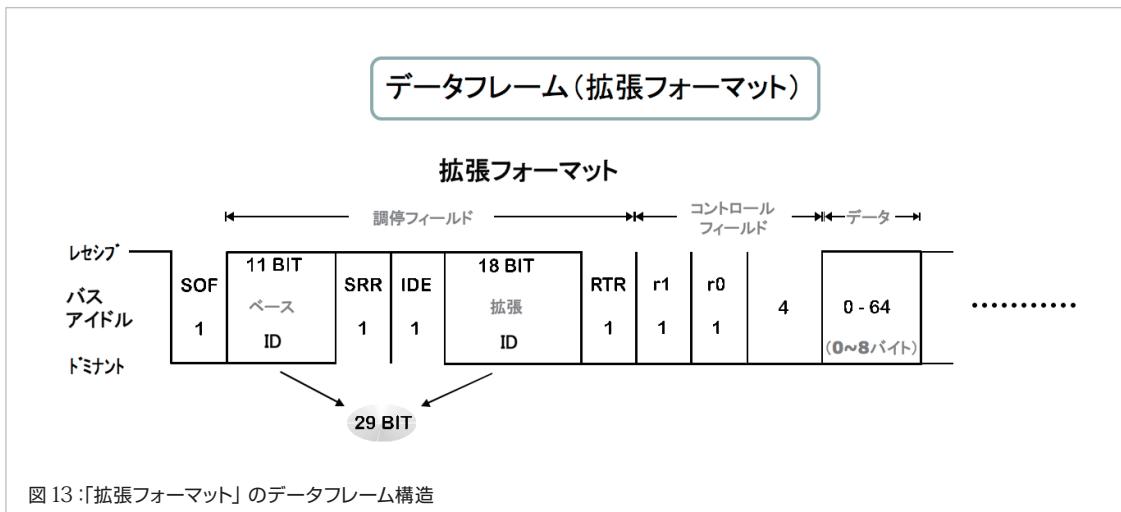
データフレームの終わりには、「EOF (End Of Frame)」が送信されます。EOFは7ビット長のレセシブとなっています。ビットスタッフイングルールは、SOF～CRCシーケンスの終わりまでの範囲で適用することになっているため、バスアイドル中やEOFは適用外となります。

► 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。

●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。



▶ 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。

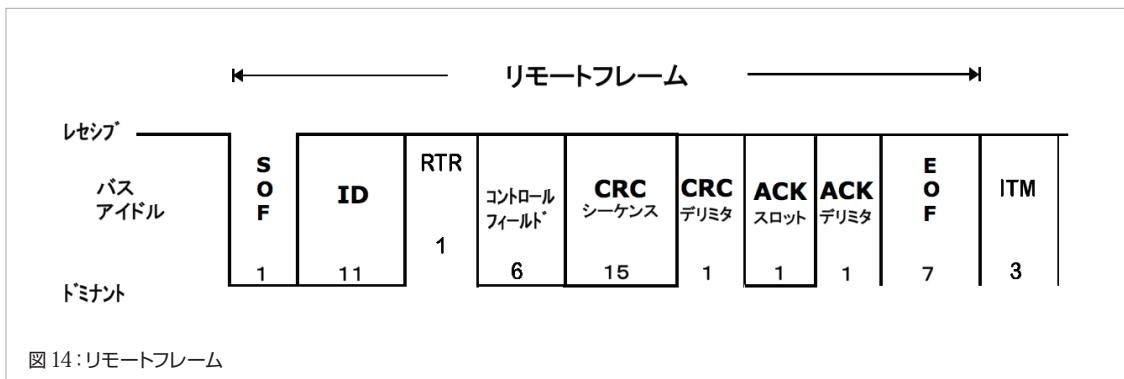
▶ 拡張ID

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 $0 \times 0 \sim 0 \times 1FFFFFFF$ 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、船用電子機器などにも幅広く利用されています。

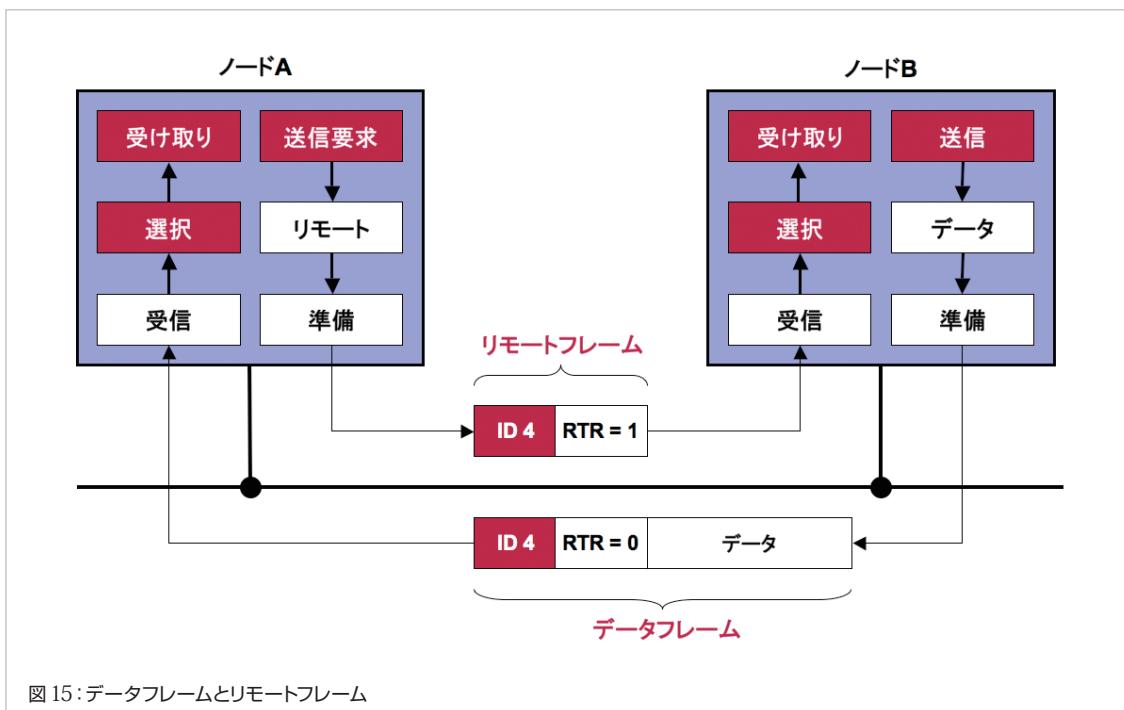
(2) リモートフレーム

「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。



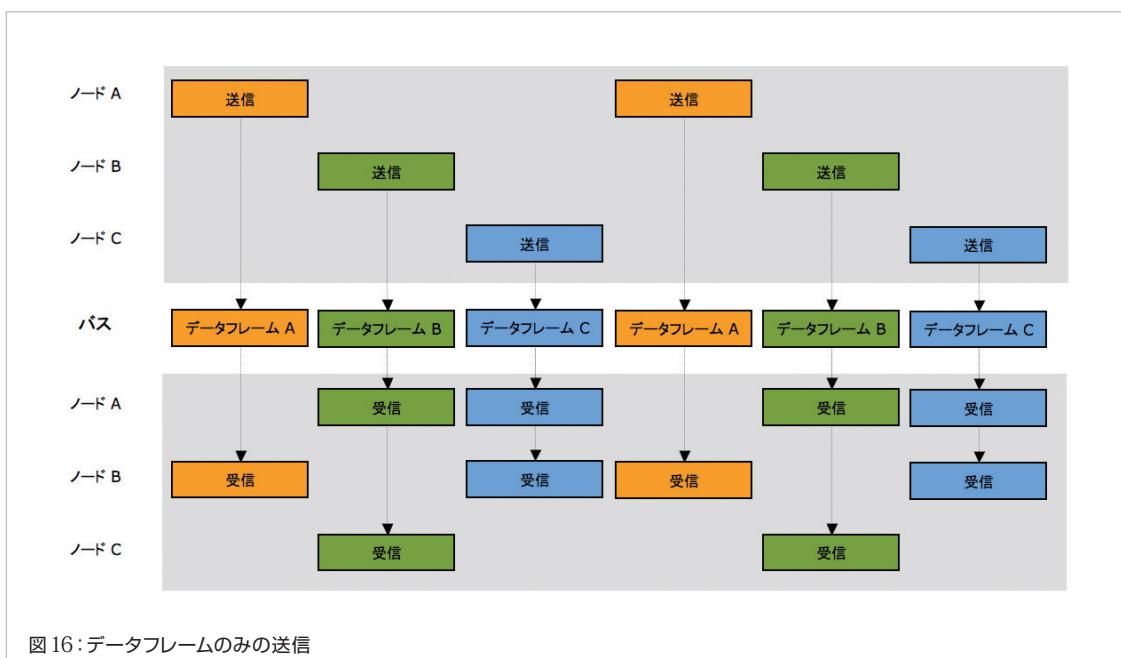
●データフレームとリモートフレーム

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)。



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。

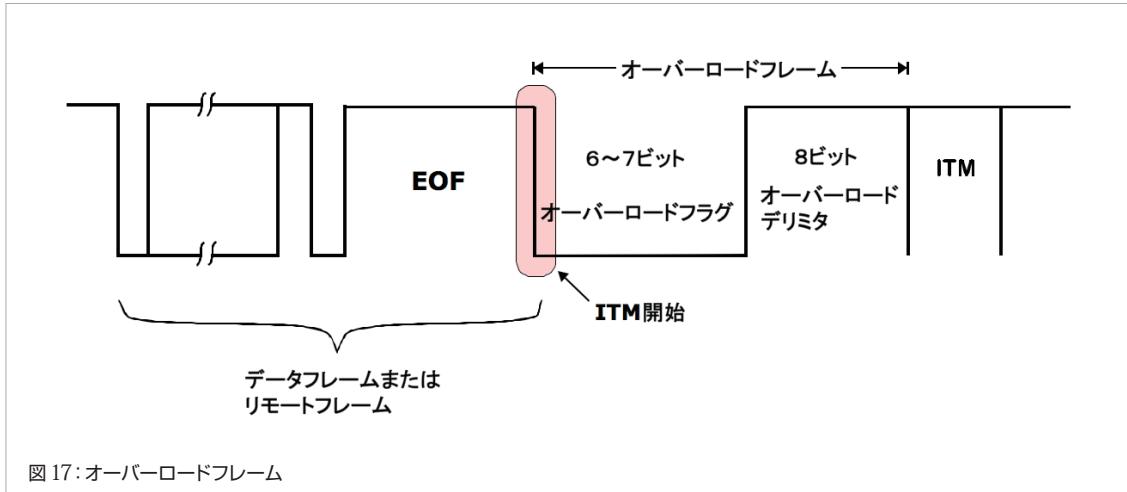
つまり、現在ではリモートフレームはほとんど使われなくなり、データフレームを定期的に各ノードからバスに送信する方式が使われるようになりました。各ノードから自由にデータフレームを送信し、そのデータフレームを必要とするノードが自由に受信する方式であれば、リモートフレームを使用せずにデータフレームのみで各ノードの情報交換が実現できます(図16)。



(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。



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。

●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グ(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。

●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。

(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。



●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、結果として通信を行えなくなることを防ぐ仕組みとなっています。

●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。

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。

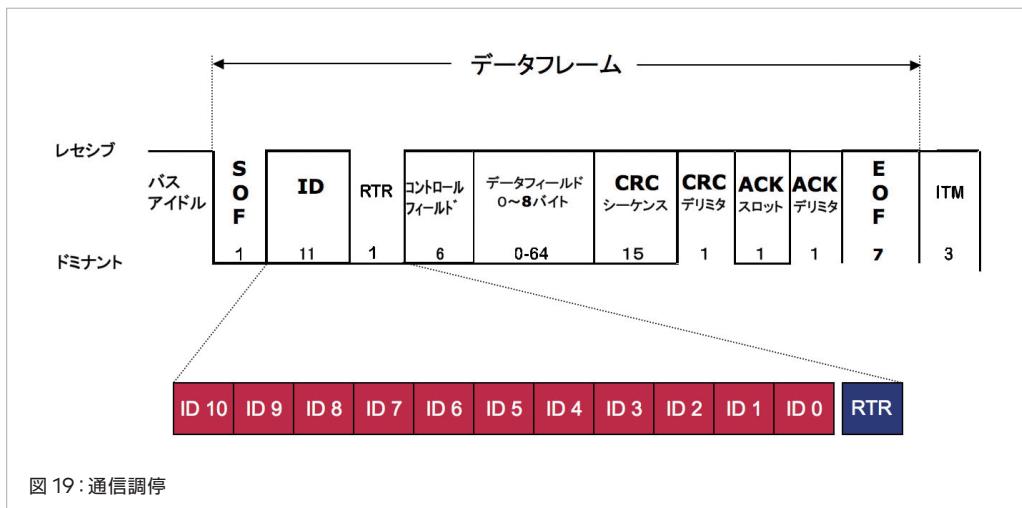


図20は、ID 0x653と0x65Bの2台のノードより同時にデータフレームが送信された場合を表しています。

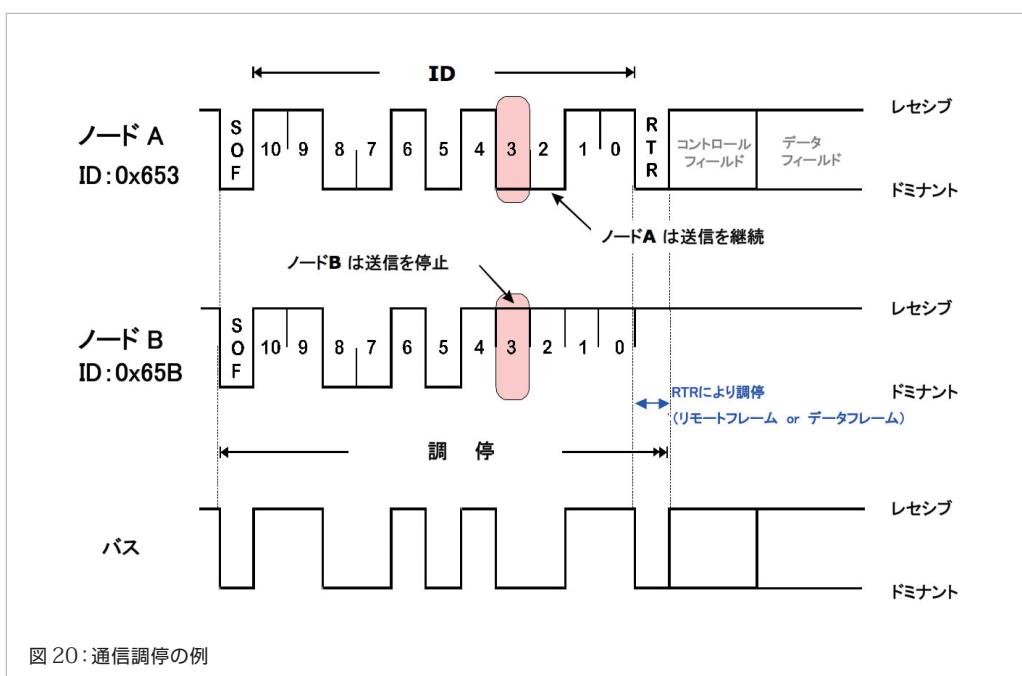


図20:通信調停の例

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 0×0 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す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。

エラー処理

●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。

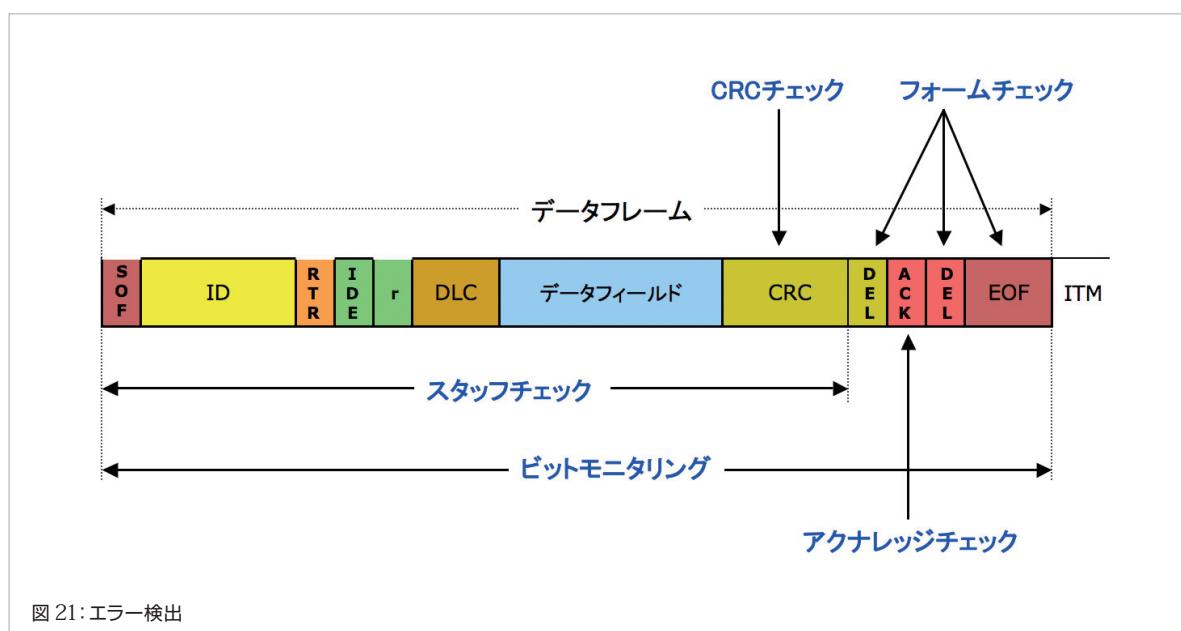


図 21: エラー検出

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。

●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。

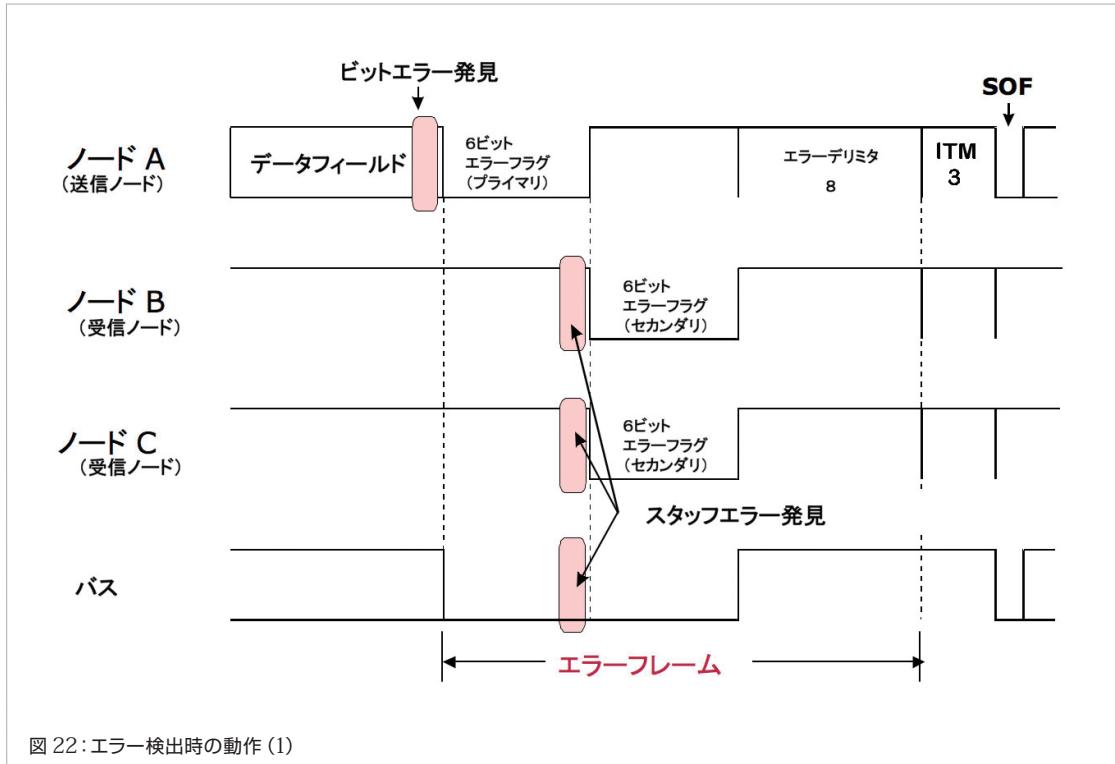


図 22: エラー検出時の動作(1)

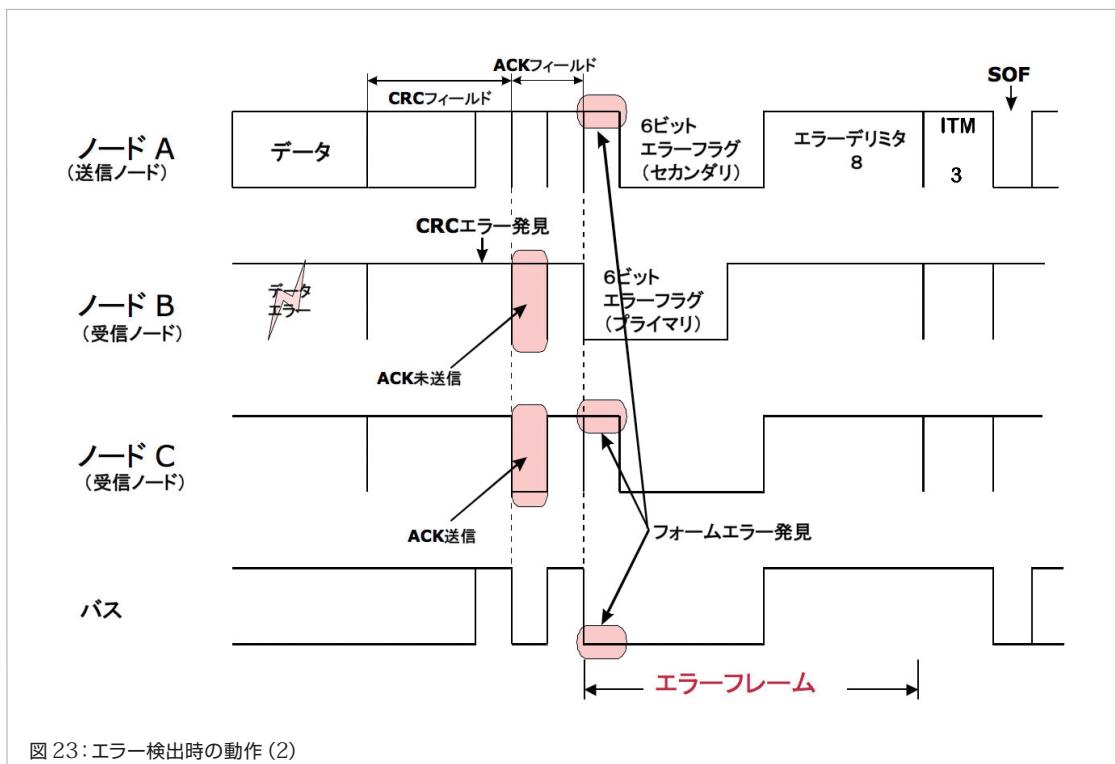
送信ノードがデータフィールドにてピットエラーを検出すると、即6ビット長ドミナントのエラーフラグ(プライマリ)を送信します。そして、他のECUはこの6ビット長ドミナントのエラーフラグをピットスタッフィングルール違反として検出し、6ビット長ドミナントのエラーフラグ(セカンダリ)を送信します。

このピ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。

●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。



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。

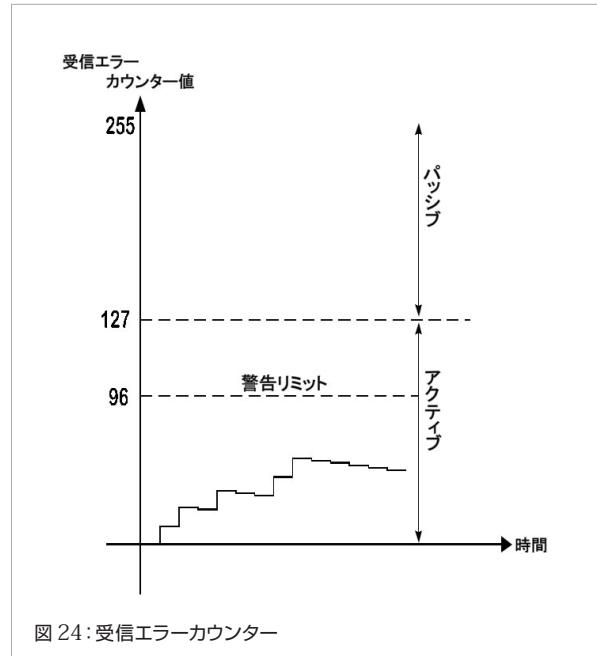
CRCエラー検出時に即エラーフラグを送信できなかった受信ノードは、EOF開始時にエラーフラグ(プライマリ)を送信します(通常時より3ビット分遅延)。このEOF内のエラーフラグの1ビット目がドミナントであることにより、他ノード(送信ノードを含む)はフォームエラーを検出し、エラーフラグ(セカンダリ)を送信します。

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。

●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。

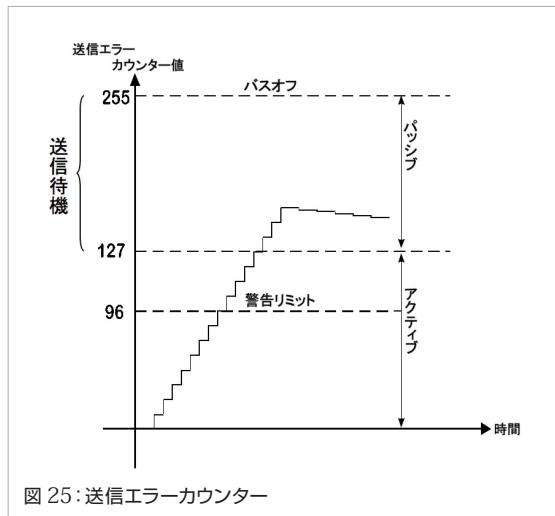
【受信エラーカウンターの値※9】

- > 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。

●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ます

●「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。

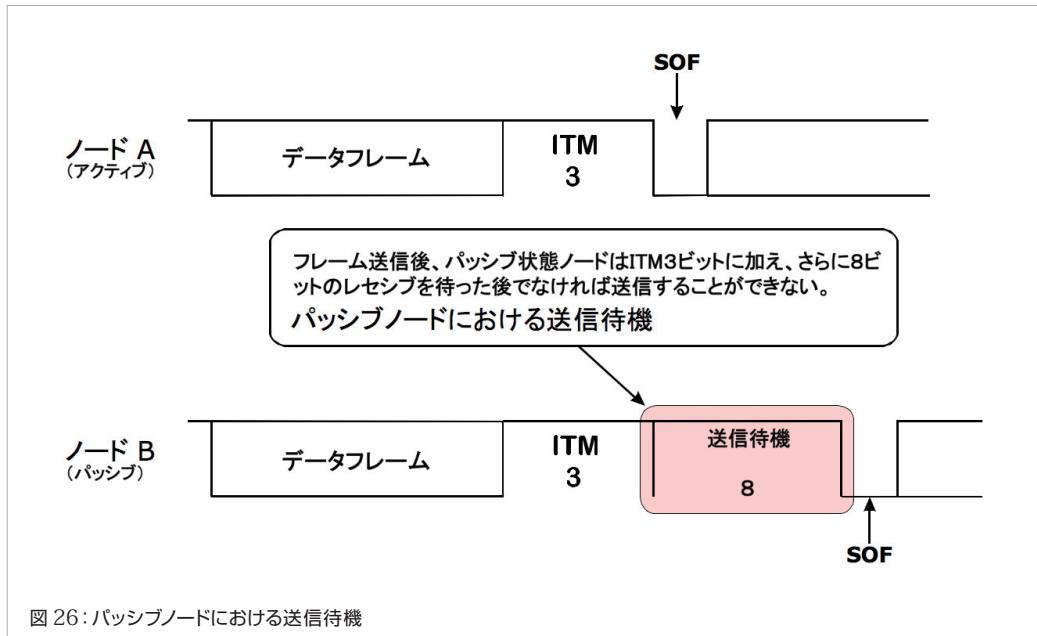


図 26: パッシブノードにおける送信待機

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。

●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。

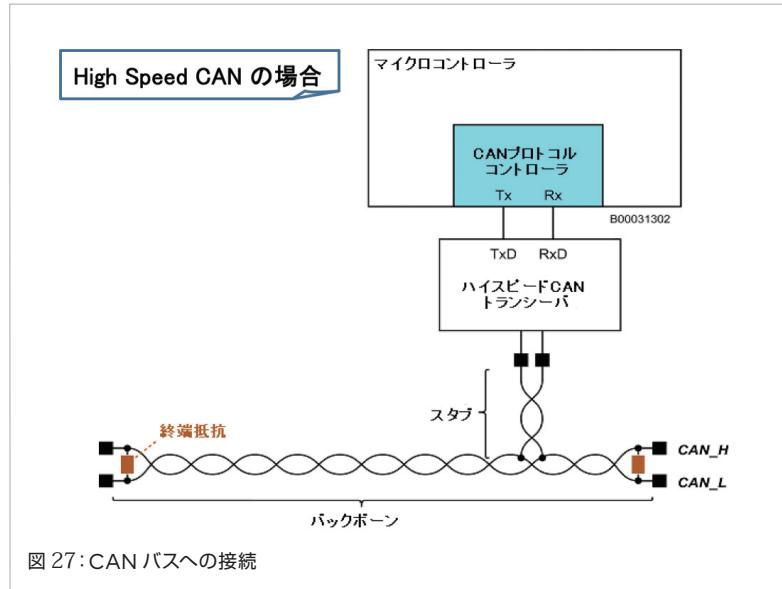
4. CANの物理層

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。

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。



【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。

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。

● 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）。

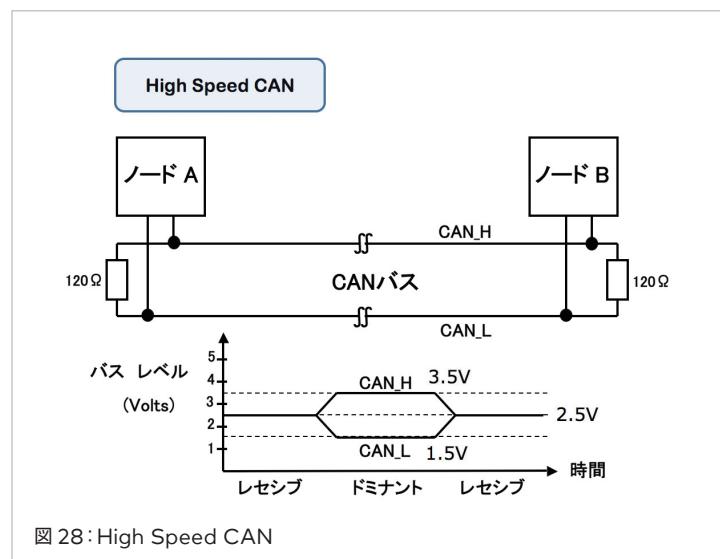


図 28: High Speed CAN

● 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)。

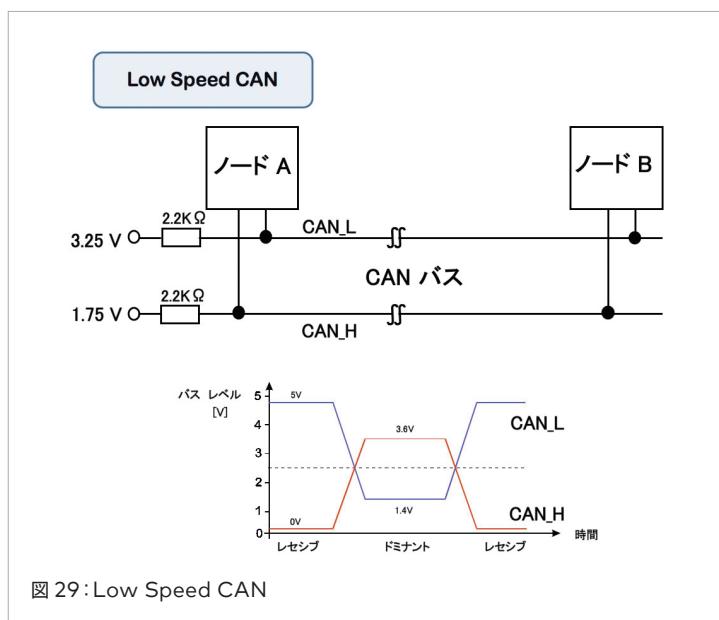


図 29: Low Speed CAN

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?

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、セキュリティー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た中で登場したのが、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、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に置き換えることによって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

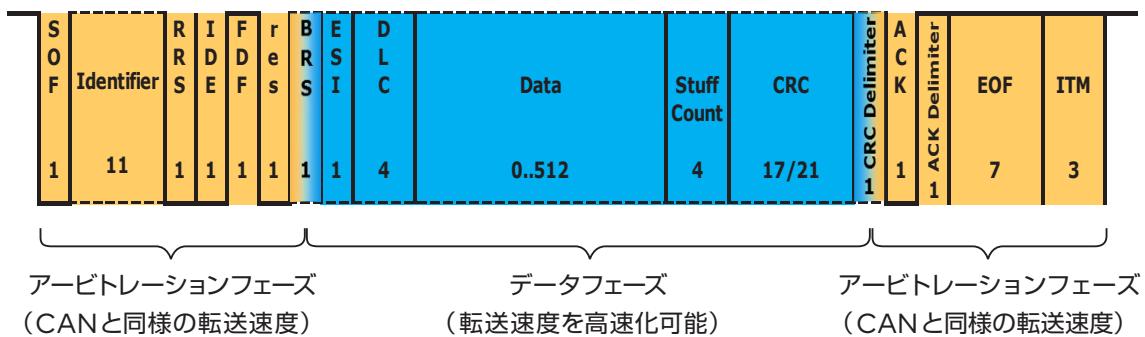
では、CAN FDで使用されるフレームタイプとCAN FDフレームの領域、各フレームの領域のフィールド構造についてCANとの違いを説明していきます。

フレームタイプ

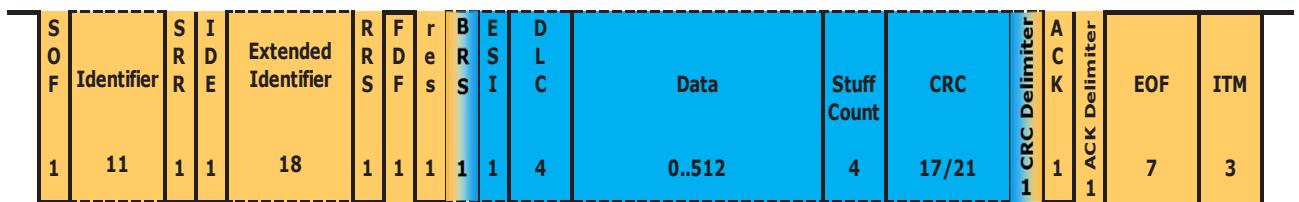
CAN FDはデータフレームのみ定義されており、CANで定義されているリモートフレームはありません。データフィールドが存在しないリモートフレームはビットレートの切替えが不要となるため、CAN FD用にリモートフレームを定義する必要がありません。データフレームはCANと同様に「標準フォーマット(11ビットID)」と「拡張フォーマット(29ビットID)」の2種類の形式があります。転送速度の高速化はBRSビットからCRC Delimiterビット間のみ可能です。

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

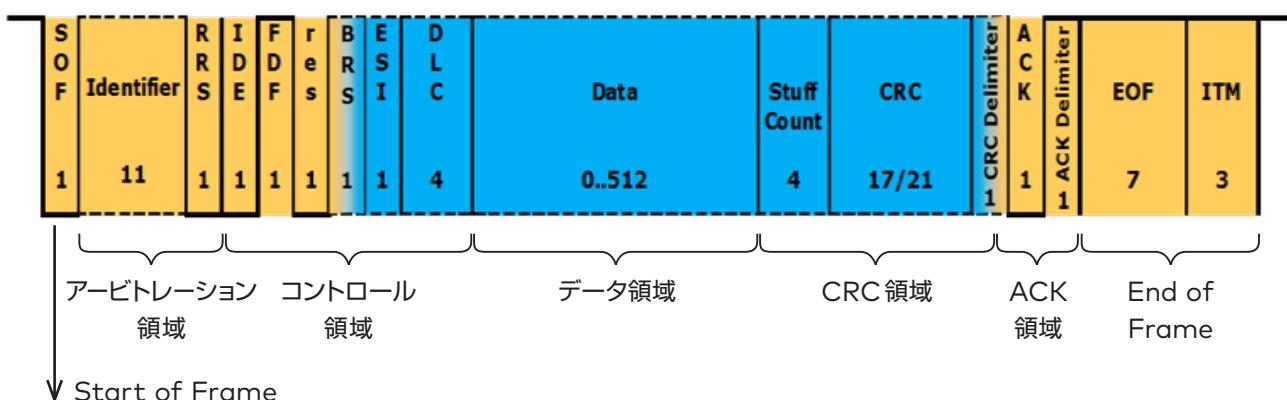


拡張フォーマットのCAN FDフレーム



CAN FD フレーム領域

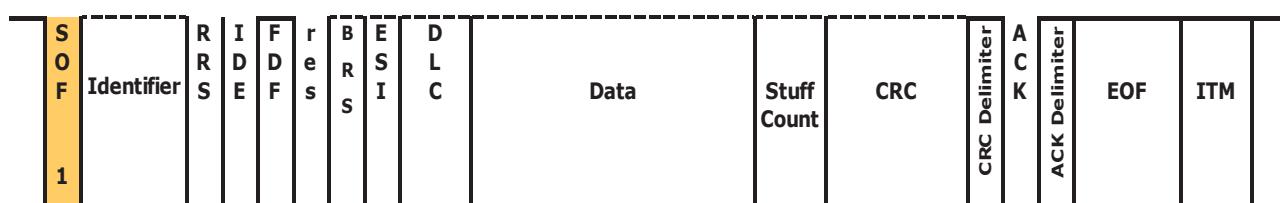
CAN FD フレームは Start of Frame (SOF) と、アービトレーション領域、コントロール領域、データ領域、CRC 領域、ACK 領域、End of Frame (EOF) の 7 つのビット領域で構成されています。



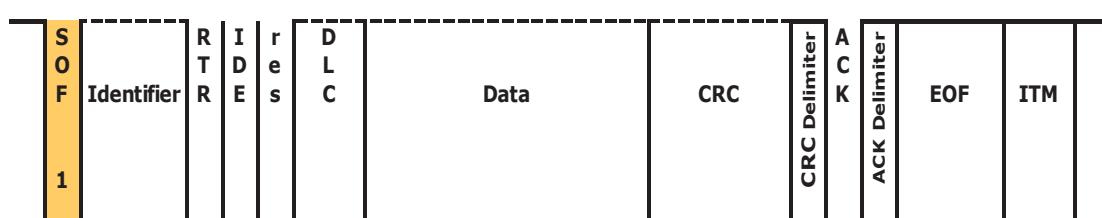
● SOF (Start of Frame)

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



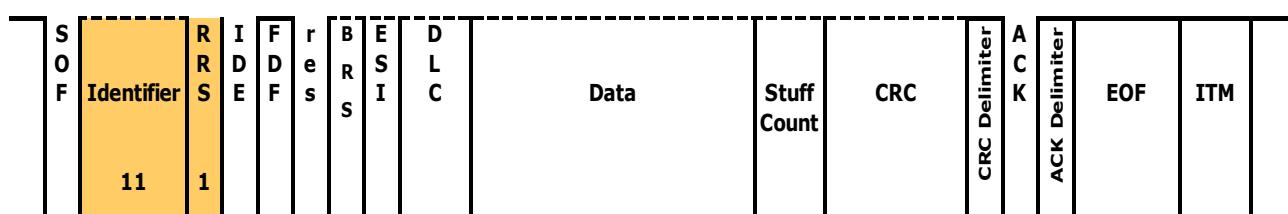
CAN フレーム



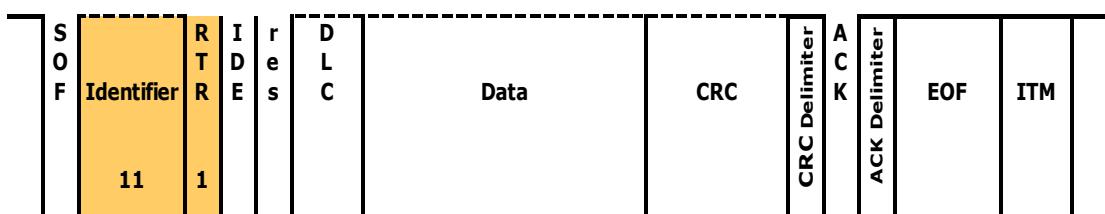
●アービトレーション領域

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)

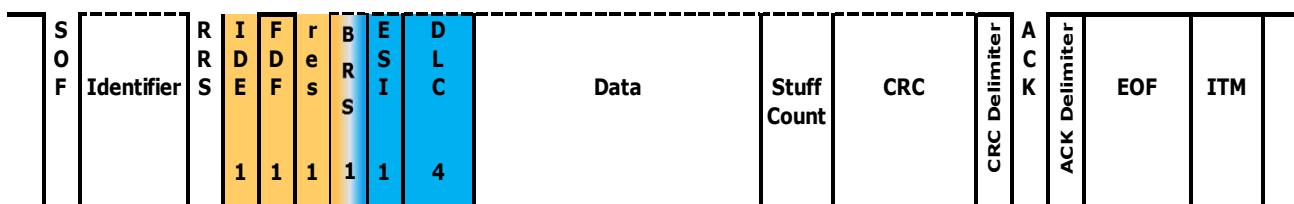
CAN FDではリモートフレームが無いので、CANで使用されていたRTRビットはRRSビットに置き換えられ、ドミナントに固定されます。

●コントロール領域

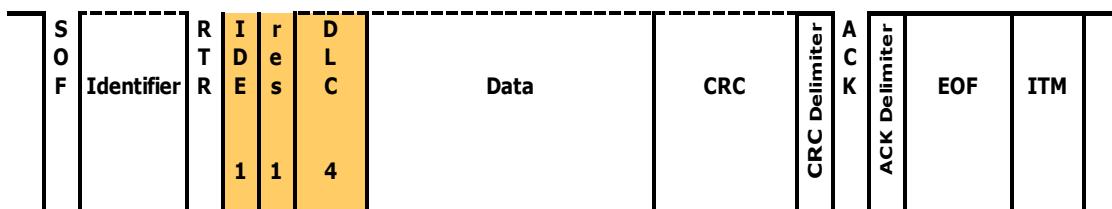
CAN FDのコントロール領域はIDEビット、FDFビット、resビット、BRSビット、ESIビットで構成されています。

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。

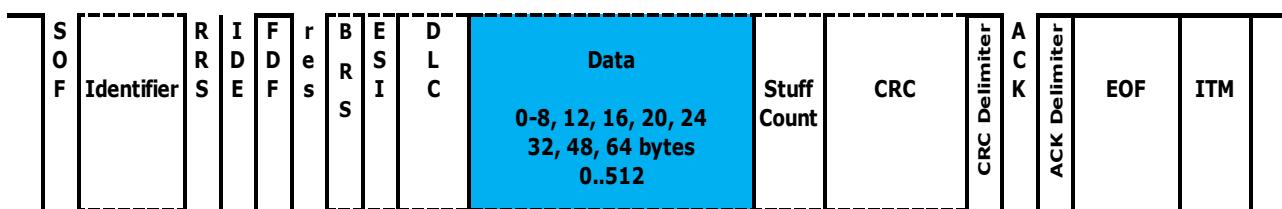
DLC	CAN (バイト)	CAN FD (バイト)
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7

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

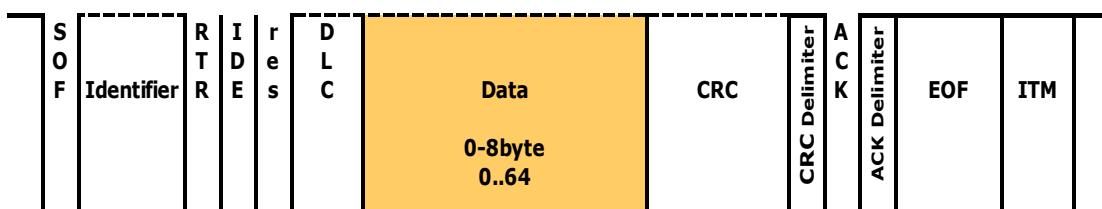
●データ領域

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



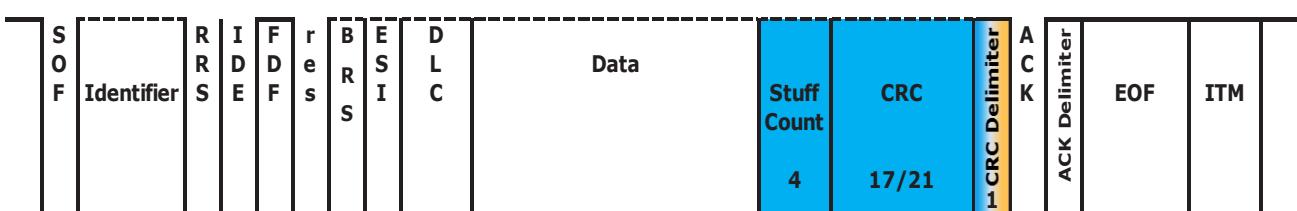
CANフレーム



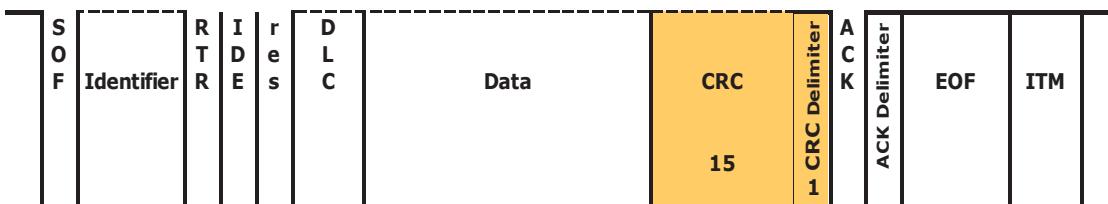
●CRC領域

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。

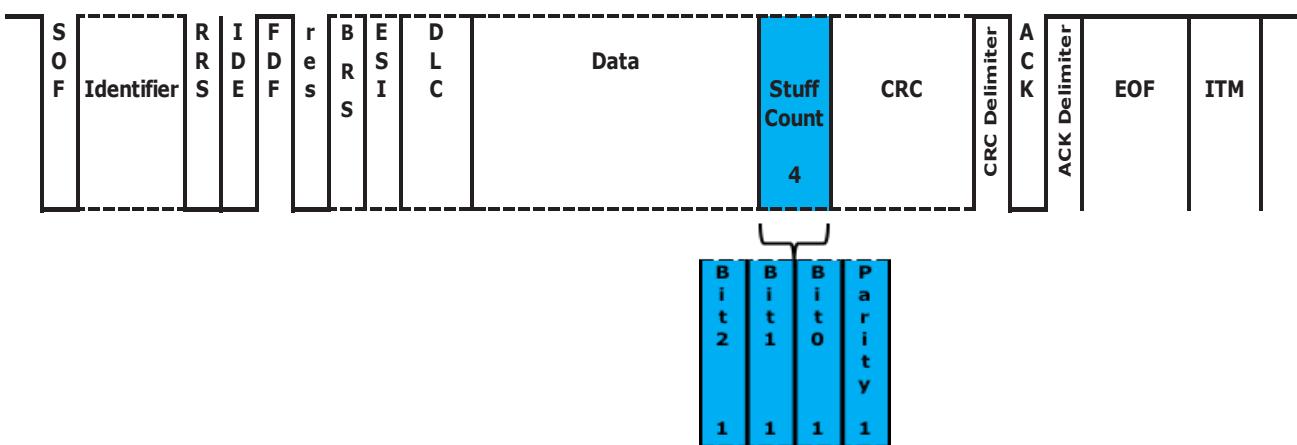
CAN FDフレーム



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传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。

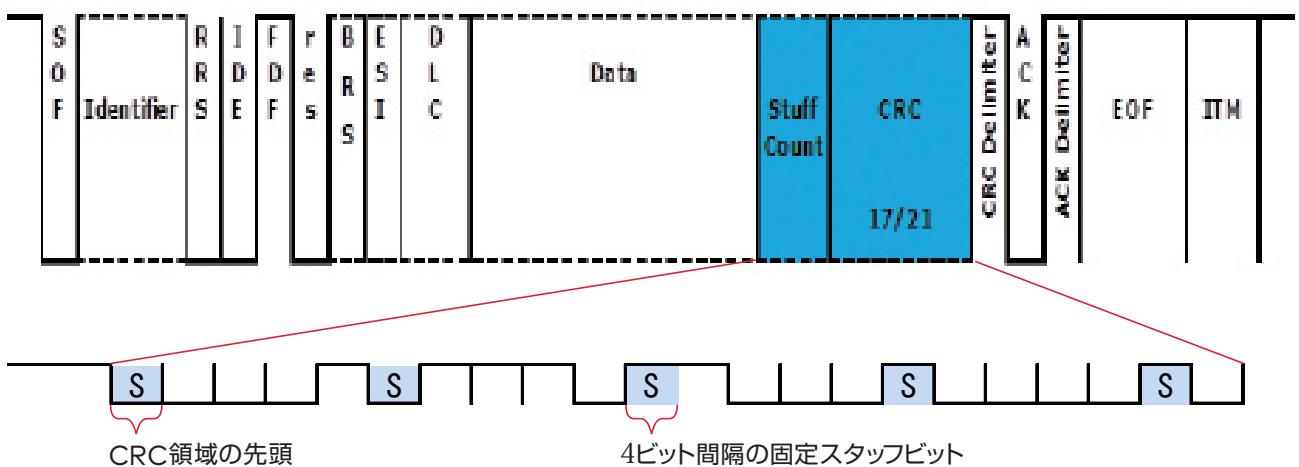
> ビットスタッフィングルール

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

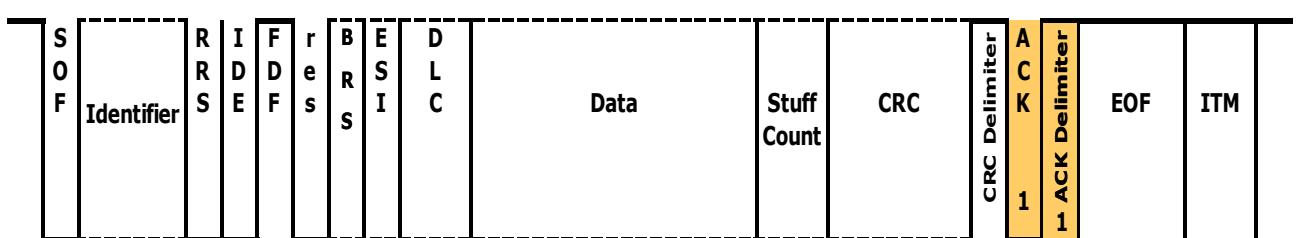
CRC領域ビットスタッフィング



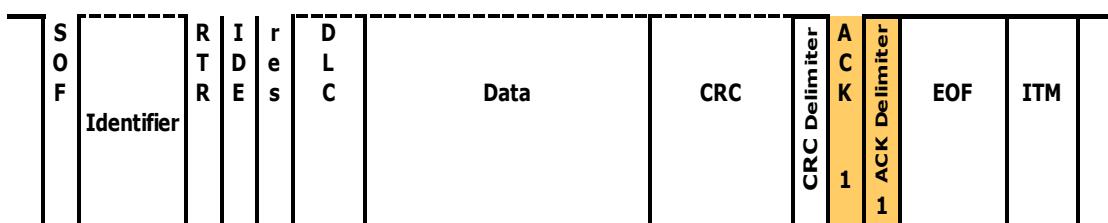
● ACK 領域

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ー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



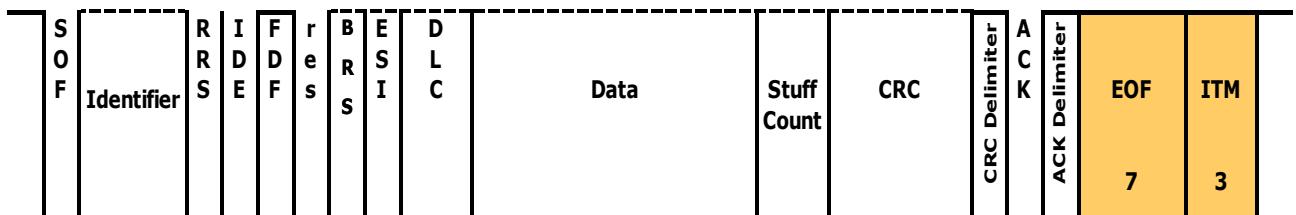
CAN フレーム



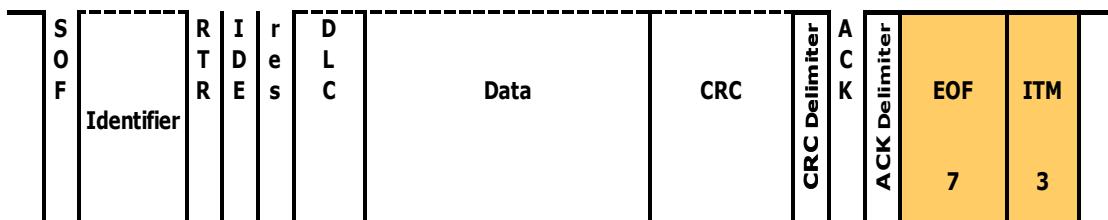
●EOF(End of Frame)

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

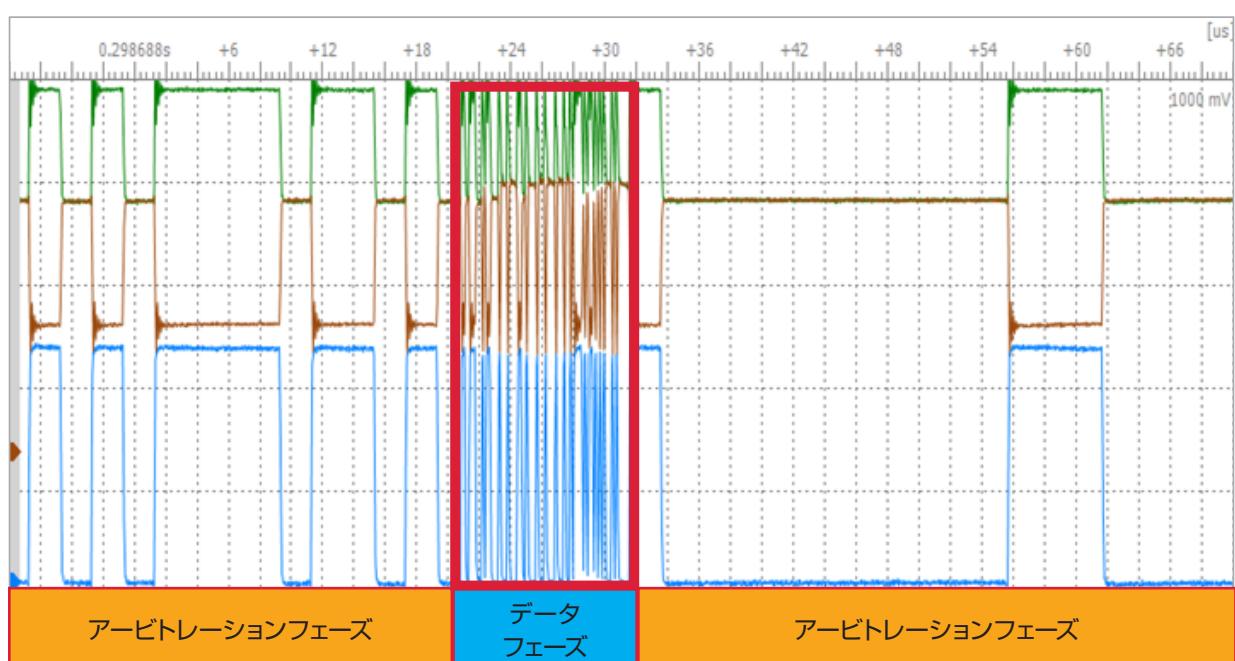


CAN フレーム



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
	CAN FD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。トランシーバー等によりますが、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ーおよびコントローラー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。

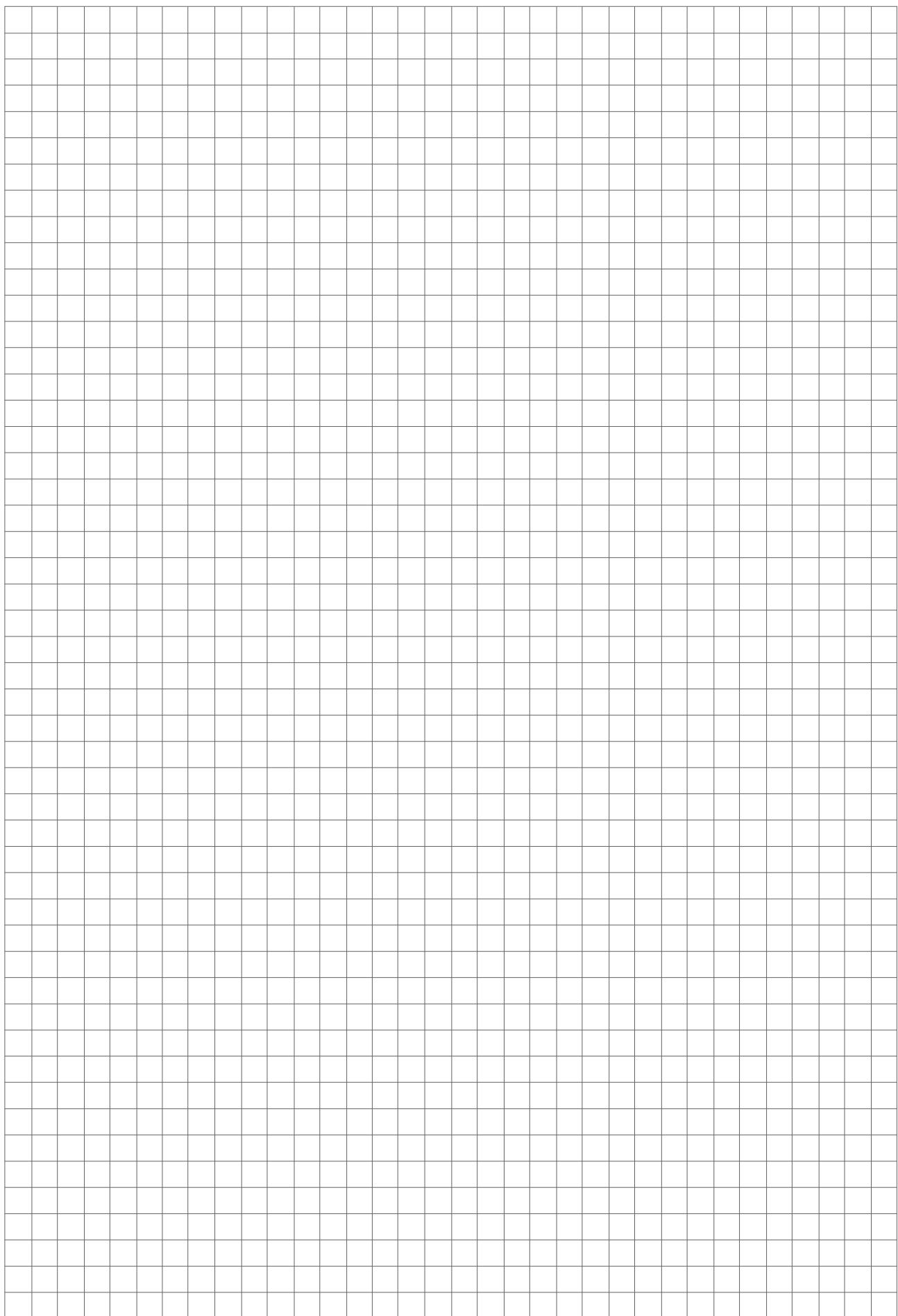
また、通信の高速化に伴い、課題となるリング対策用の回路(RSC: Ringing Suppression Circuitry)の導入などが検討されています。

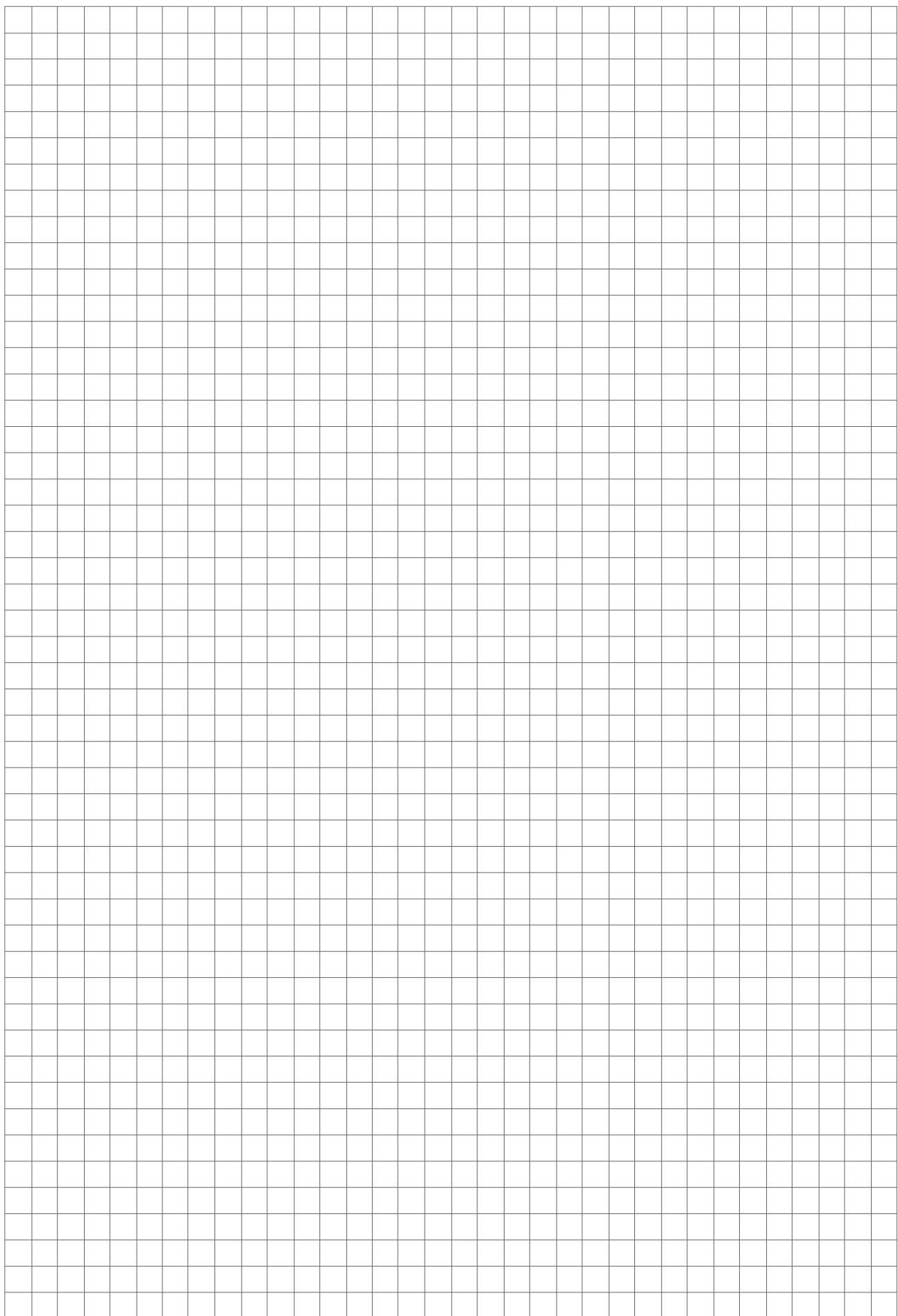
あとがき

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: CAN Newsletter magazine 「Ringing suppression in CAN FD networks」

http://can-newsletter.org/engineering/engineering-miscellaneous/160309_ringing-suppression-in-canfd-networks_denso/







www.vector.com/jp/ja/