AISIN GROUP

初めての CAN/LIN

アイシン・ソフトウェア株式会社
基盤技術開発部　要素開発G
田代　真司

AISIN
AISIN SOFTWARE Co.,Ltd.

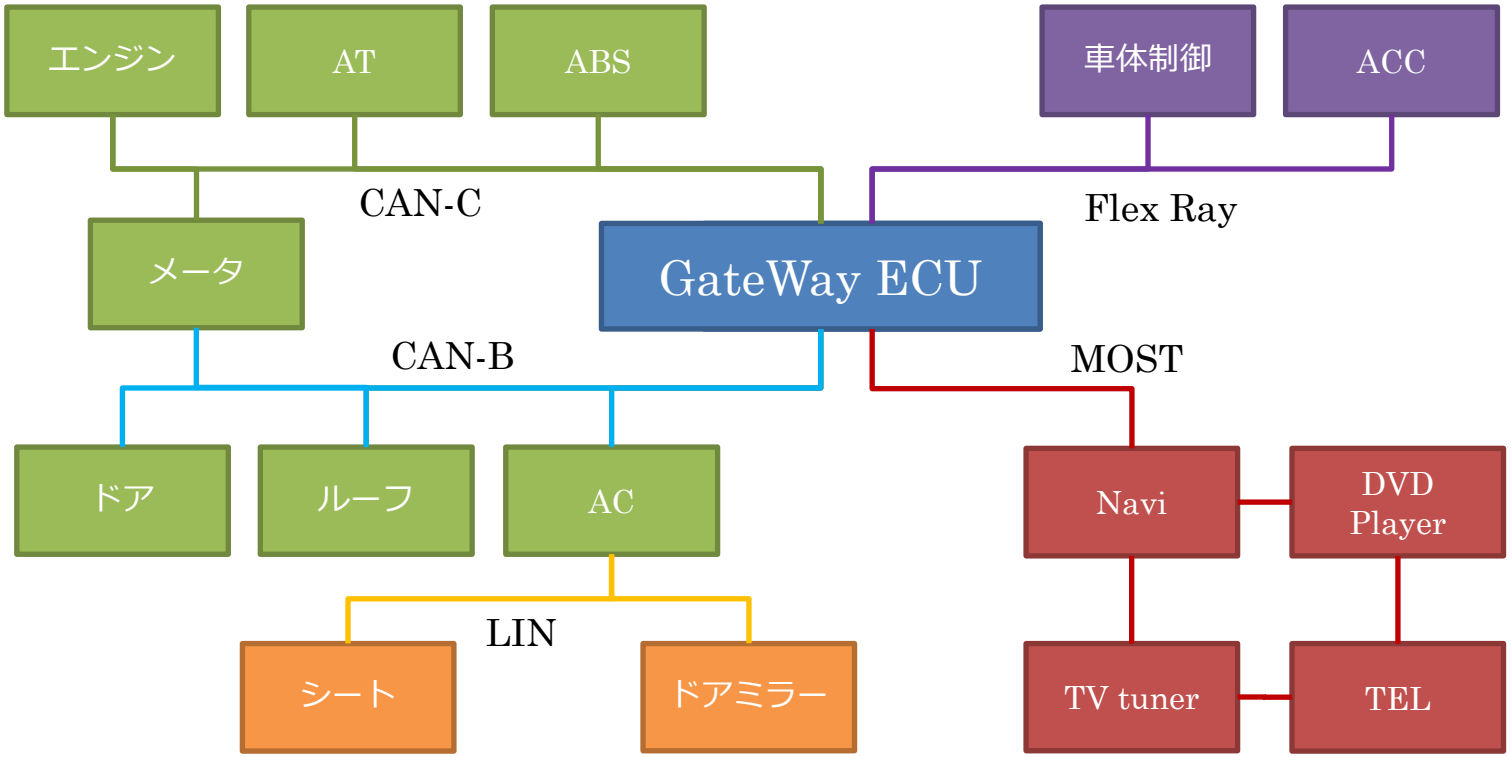
目次

1.車載ネットワークの概要

車載ネットワークの概要

（1）車載ネットワークとは？

自動車制御用コンピュータ(ECU(Electronic Control Unit))同士が情報交換をするための通信のこと



車載ネットワークの概要

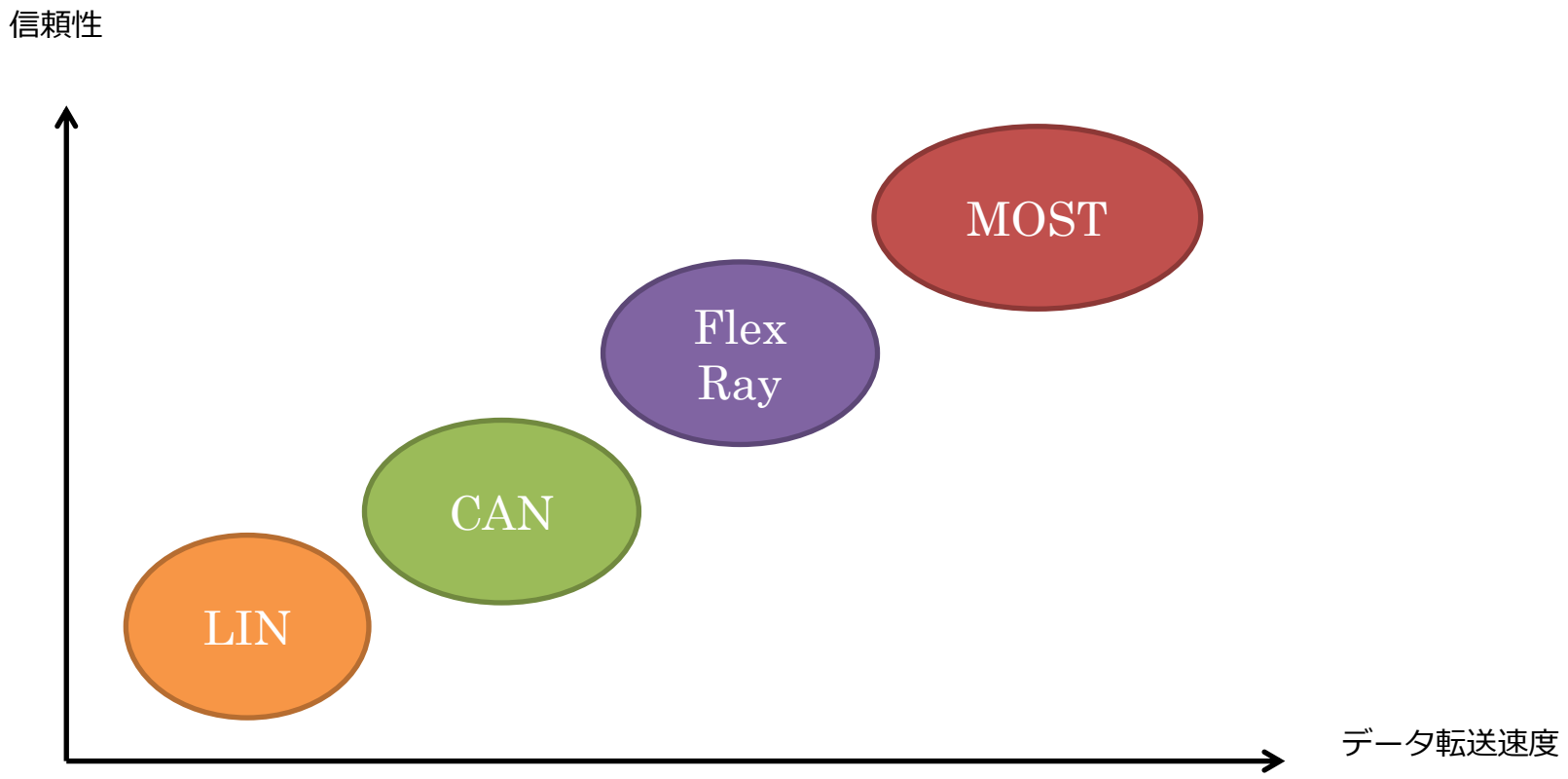
（2）各車載ネットワークの特徴

バス規格	LIN	CAN	FlexRay	MOST
伝送速度	20kbps	1Mbps	10Mbps	150Mbps
コスト	低い	比較的低い	高い	高い
ワイヤの本数	1本	2本	2本または4本	2本
主なアプリケーション	車体制御用電装品(ミラー、電動シート、各種アクセサリ)	パワートレイン(エンジン、トランスミッション、ABS)	高性能パワートレイン、安全機能(ドライブバイワイヤ、アクティブ・サスペンション、アダプティブ・クルーズコントロール)	カーオディオ、カーナビ

車載ネットワークの概要

（3）各車載ネットワークの位置づけ

データ転送速度と信頼性をキーとした各ネットワークの位置づけを示すと以下のようになります。



車載ネットワークの概要

（４）各車載ネットワークにおけるCAN/LINの位置づけ

１．CANやLINは、車載システムの高機能化が進んだ現在において、求められる高速な通信要求を満たしている通信規格とは言い難い。

２．しかしCANについては古くからある規格であり、既にCANを基幹にしてネットワーク構築された車が多い。CAN機能自体が一般化してコストが低下した現在、リスクの高いネットワークの大変更を避ける目的で、車載ネットワークにCANを採用する例は多い。

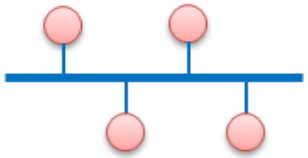
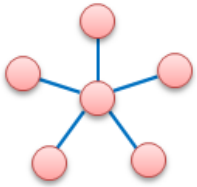
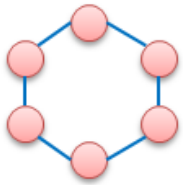
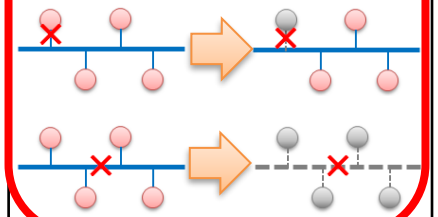
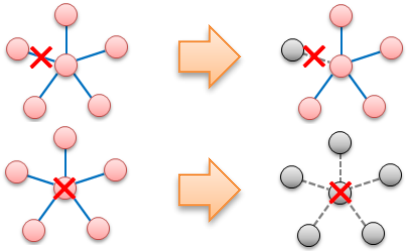
３．現在ではCANから大きなソフト変更が必要なく、かつ高速な通信が可能となるCAN-FDを選択するケースが出ている。

４．LINは部品点数が少なく、廉価部品を採用していることで、非常に低コストでネットワークを構築できることから、CANのサブネットワークとしてドアスイッチ、ドアミラー、アクチュエータ制御等、大量の情報や高速な通信速度を利用しない場所で現在も採用されている。

2.CANの概要

CANの特徴

（１） 通信形態

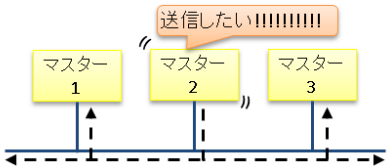
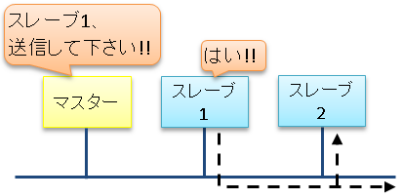
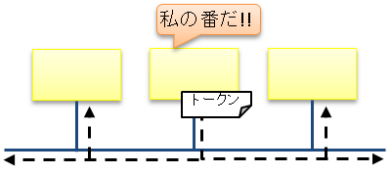
種類	バス型	スター型	リング型
:ノード :バス			
特徴	・構造が単純 ・比較的安価	・ノードの追加/削除が容易 ・トラブルの発生箇所の特定が容易	・通信の流れが一方方向 ・信号の衝突がない
障害耐性 ※1つのバス、または、ノードに障害が発生した場合	障害が発生するバス、または、ノードによって、他のノードへの影響は変わる 	障害が発生するバス、または、ノードによって、他のノードへの影響は変わる 	障害が発生するバス、または、ノード関わらず、他のノードへ影響する 

“複雑な配線の解消”という目的があったことから、CANでは「バス型」を採用している。

CANの特徴

AISIN アイシン・ソフトウェア株式会社

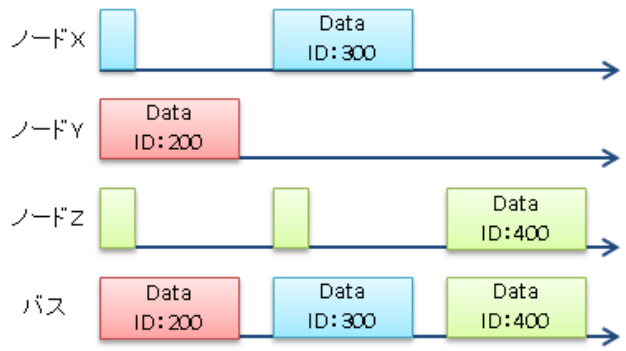
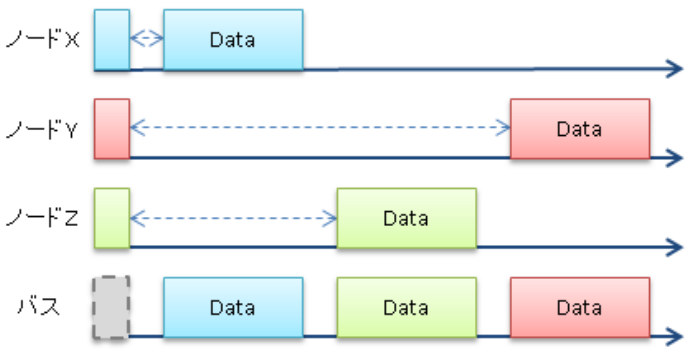
（2） 通信方式

種類	マルチ・マスター方式	マスター・スレーブ方式	トークンパッシング方式
通信方法			
	送信順番は早い者勝ち	通信順番はマスターが決定	通信順番はトークンの回る順番
特徴	・各ノードが均一仕様で設計できる ・各ノードに優劣がないため、イベント指向通信に向く ・ノードの追加接続が容易	・データの衝突がない	・データの衝突がない

CANではノードの追加、削除が容易で、同一バス内の全ノードへ同時送信可能なマルチ・マスター方式を採用している。

CANの特徴

（3） 調停方式

種類	CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance)	CSMA/CD (Carrier Sense Multiple Access with Collision Detection)
調停方法		
	複数のノードから同時に通信線へデータが送信されても、その中の “優先順位が高いものを送信する”ようになっている。	データの衝突があった場合は、 バスの空きを待ち、 ランダムな待機時間後に再送信する

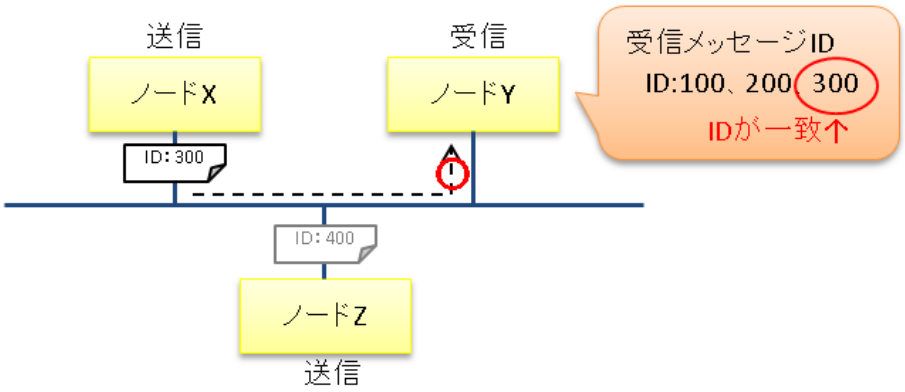
CANはCSMA/CA方式を採用している。
具体的な優先順位の判定方法はCANプロトコル詳細で解説します。

CANの特徴

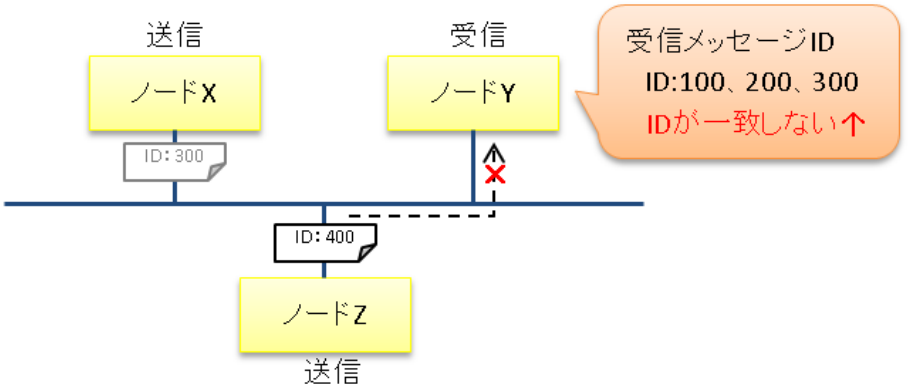
AISIN アイシン・ソフトウェア株式会社

（4） Acceptance Filter

〈アクセプタンスフィルタと一致したIDの場合〉



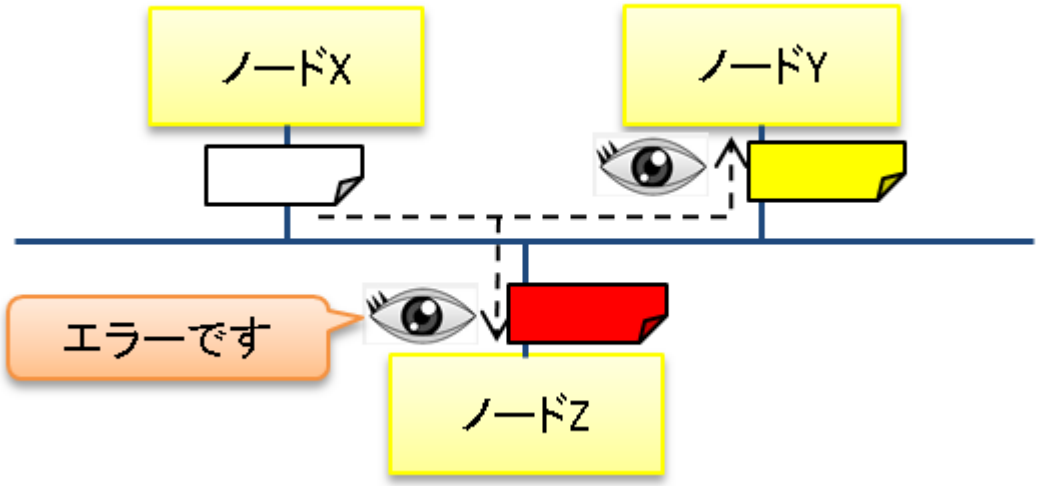
〈アクセプタンスフィルタと一致したIDでない場合〉



CANでは、各ノードにAcceptance Filterを保持可能である。
Acceptance Filterに設定したIDと一致したメッセージのみ、
受信バッファに保存できる。

CANの特徴

（5） エラー検出

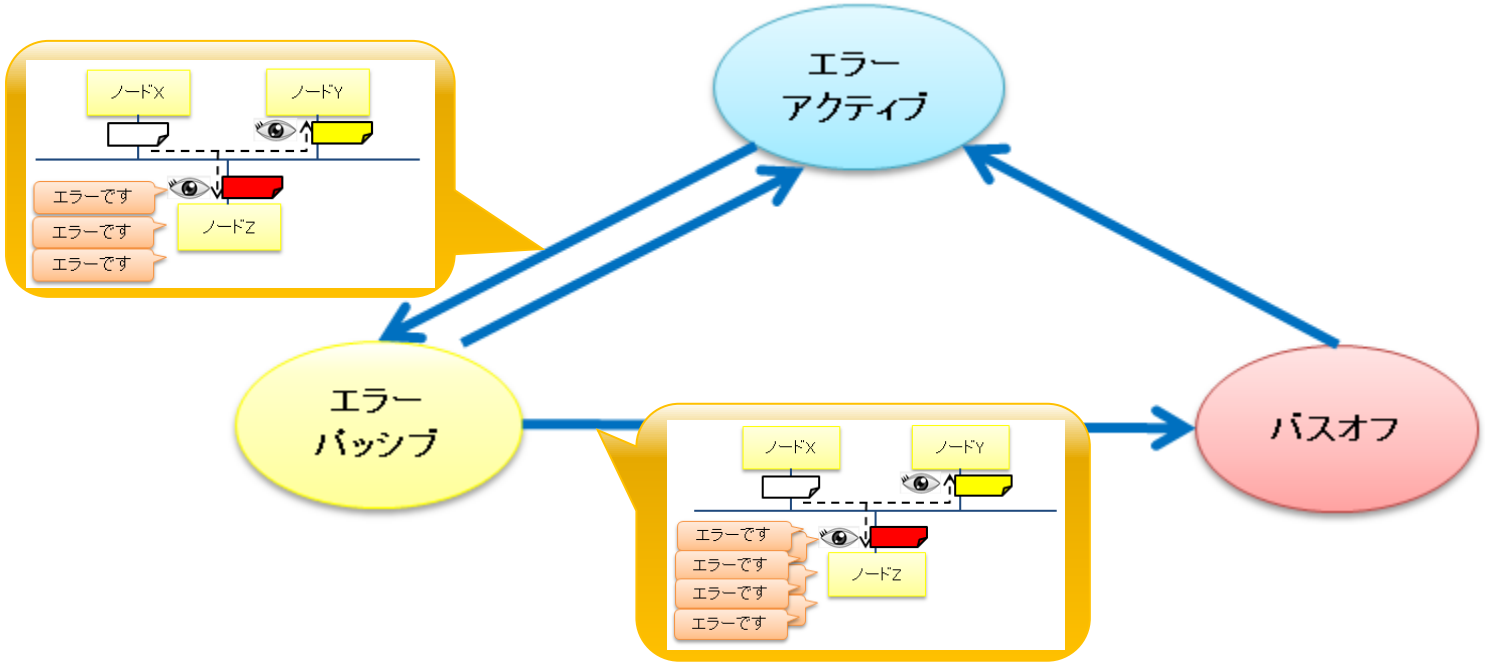


バス上に存在する全ノードは送信されているメッセージに異常がないかを監視している。
エラーを検出したノードは即、エラーフレームを送信し、エラーが発生したことを全ノードに通知する。

※エラーフレームの詳細はCANプロトコル詳細で解説します。

CANの特徴

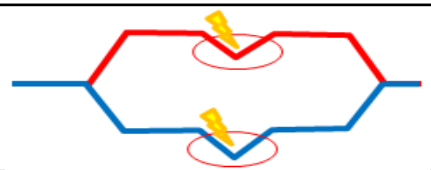
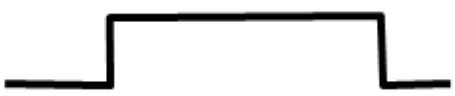
（６） 故障拡散防止



エラーを頻発するノードは送信をしにくくなる。
(エラーアクティブ⇒エラーパッシブ)
さらにエラーを頻発するノードはバス自体にアクセスできなくなる。
(エラーパッシブ⇒バスオフ)

CANの特徴

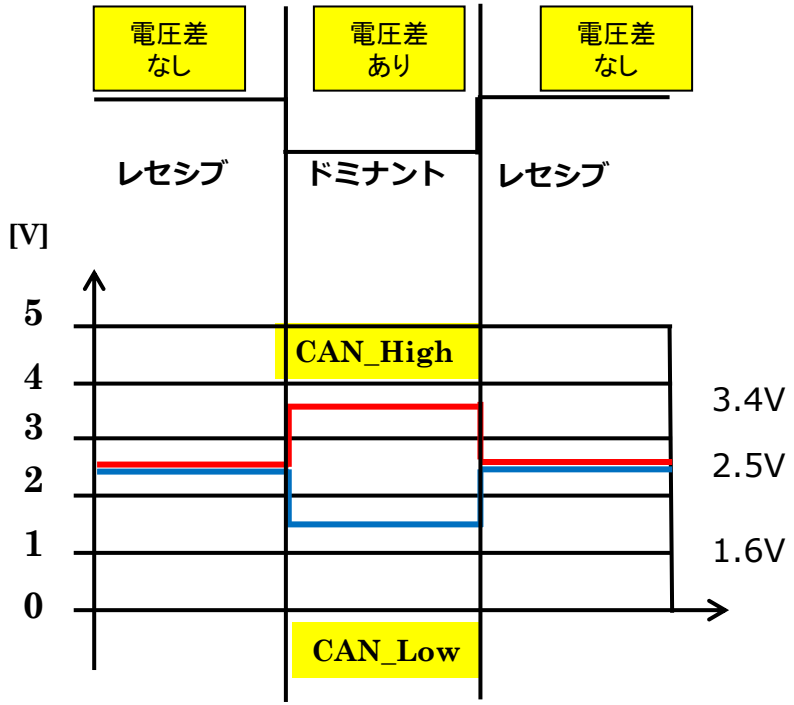
（7） ノイズ対策

種類	単線	2本線	ツイストペアケーブル
実波形			
差分電圧			
入力信号			

CANの信号線にはツイストペアケーブルを用いている。
各線に流れる電圧の差でデータを送信している。

CANの特徴

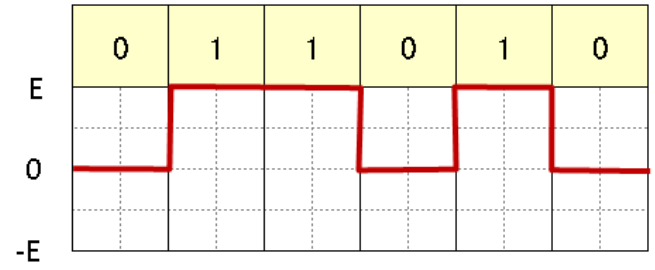
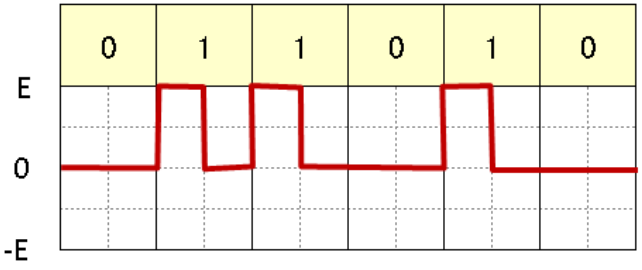
（8） ドミナントとレセシブ



CANでは電位差でHighかLowかの判断をします。
電位差がある方をLow(0)・ドミナント(優勢)
電位差がない方をHigh(1)・レセシブ(劣勢)としています。
名前の通り、バス上の優先度は名前の通りドミナントの方が高くなります。

CANの特徴

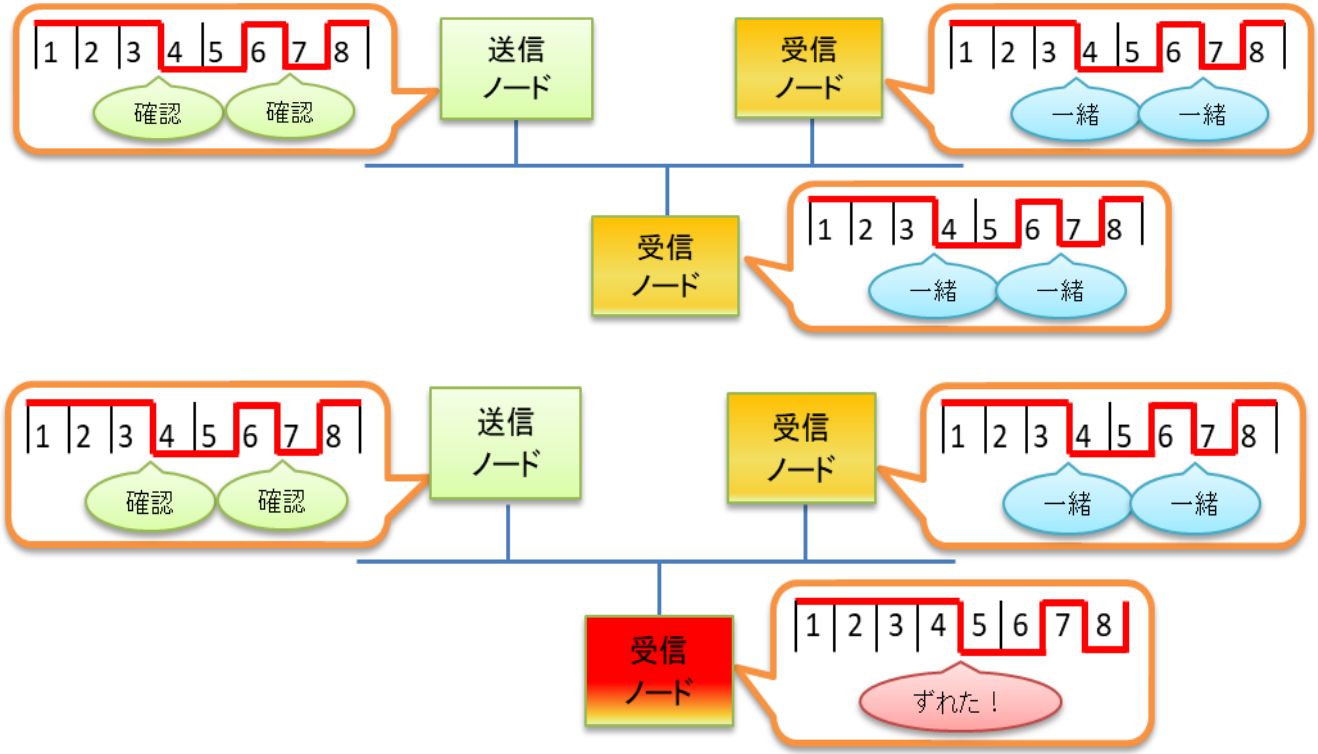
（9）伝送路符号

種類	NRZ方式 (Non Return to Zero)	RZ方式 (Return to Zero)
方式		
	ビット転送ごとに、電位を0に戻さない	ビットとビットの間に電位0を挿入する

伝送路符号はNRZ方式を利用しています。

CANの特徴

（１０） 同期

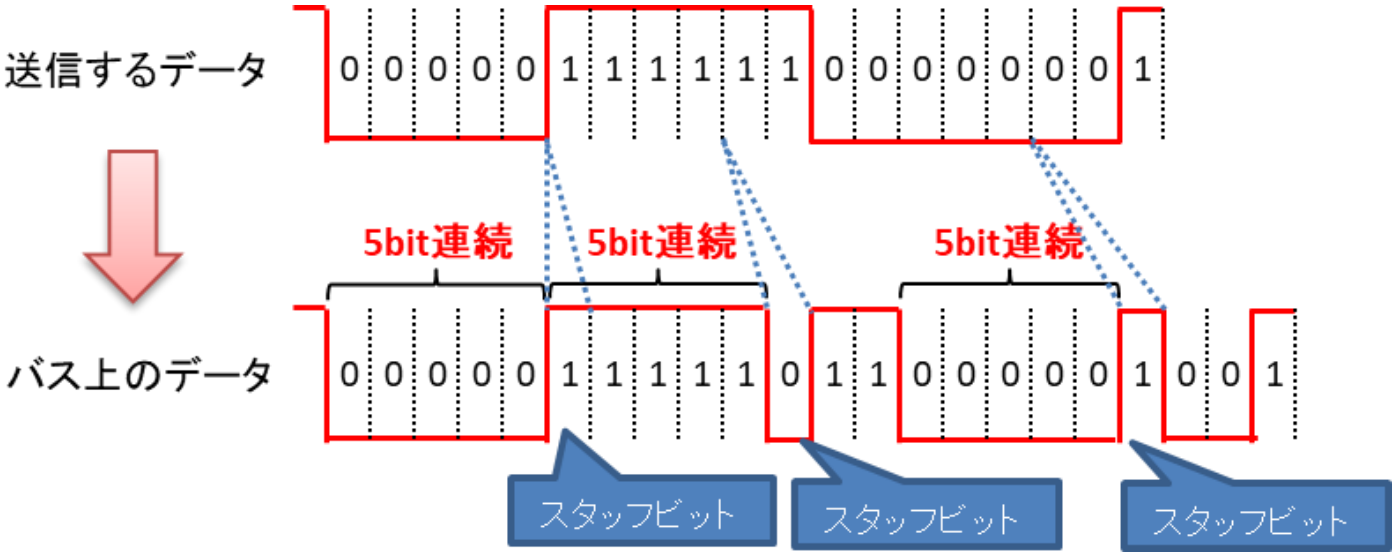


各ノード間でクロック誤差によるデータ誤認を防止するため、補正を定期的に行っている。

補正を行うタイミングは信号がレセシブ(1)⇒ドミナント(0)へ変化する時である。

CANの特徴

（１１） ビットスタッフィング



・各ノードのクロック誤差に起因するタイミング誤差が累積しないようにするための機能がCANには設けられている。同じレベルが5bit連続した場合に、レベルを反転したビットを挿入する。
・対象はデータフレーム/リモートフレームのStartからCRCシーケンスの間まで。それ以外の区間、エラーフレーム、オーバーロードフレームは対象外

3.CAN通信プロトコルの詳細

CAN通信プロトコルの詳細

（１） フレーム種別

CANのフレームは、以下の4種類が定義されている。

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

次からは各フレームの詳細について概説する。

CAN通信プロトコルの詳細

（2） データフレーム①

・CAN通信において、何かしらのデータを送信したい場合、
データフレームの形式で送信する。

・データフレームの形式には“標準フォーマット”と
“拡張フォーマット”の細部で異なる2種類のフォーマットが存在する。

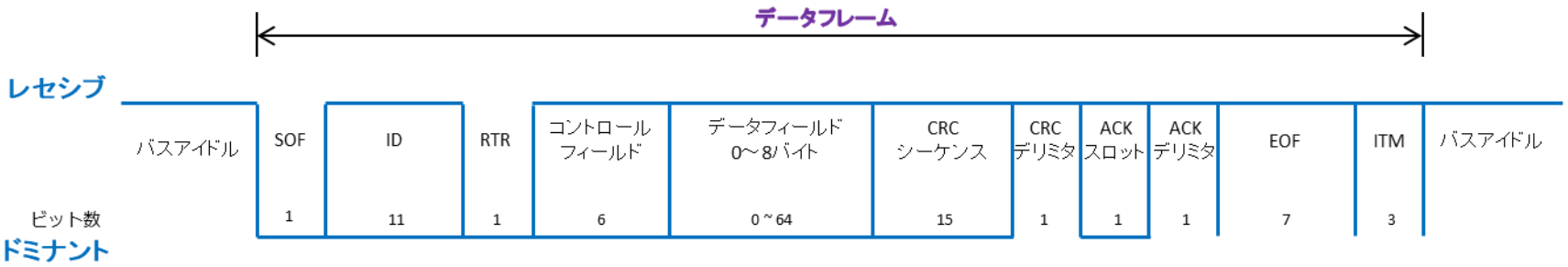
・“拡張フォーマット”の受信に対応しているバージョンは2.0Bから
であり、2.0Aとの混在はできない。

ID / プロトコル	標準フォーマット		拡張フォーマット	
	送信	受信	送信	受信
2.0A	○	○	× (送信不可)	**×** **(エラー)**
2.0B Passive	○	○	× (送信不可)	△ (無視)
2.0B Active	○	○	○	○

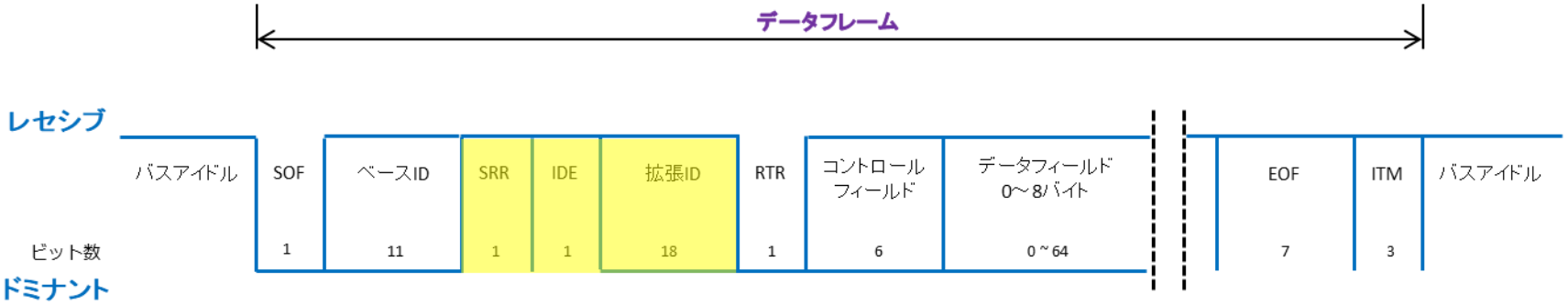
CAN通信プロトコルの詳細

（2） データフレーム②

★標準フォーマット

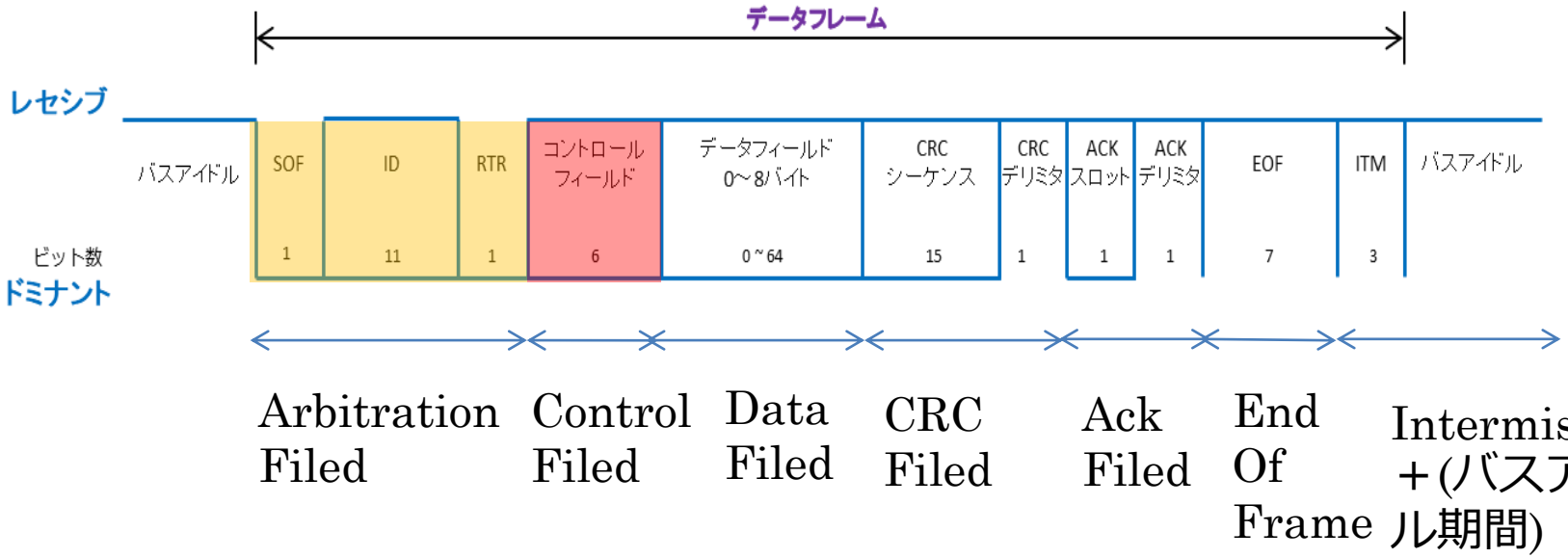


★拡張フォーマット



CAN通信プロトコルの詳細

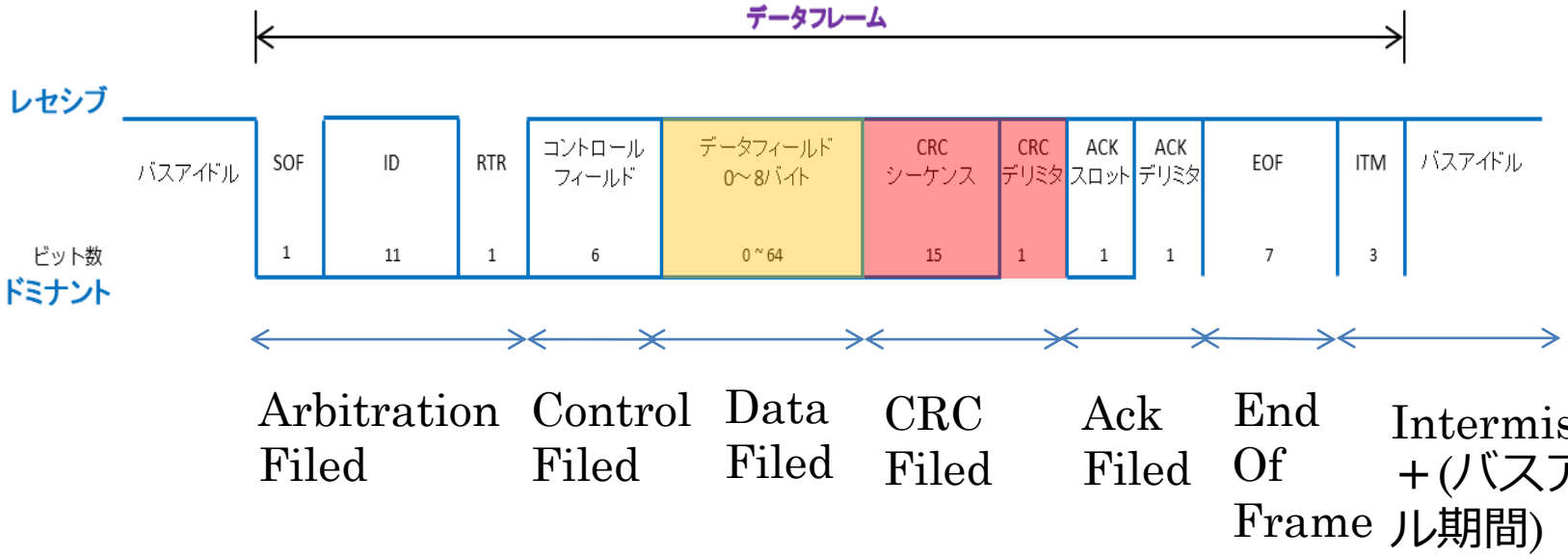
（2） データフレーム③(標準フォーマット①)



- Arbitration Filed
  →フレームの開始の指定と、CANのメッセージIDの指定、及び
  データフレームとリモートフレームかの区別をするFiled
  名前の由来は、ここでメッセージの優先度の調停(Arbitration)を
  行っていることから(調停の仕組みについては後述)
- Control Filed
  →データフィールドの長さを指定。0Byte～8Byteが設定可能

CAN通信プロトコルの詳細

（2） データフレーム④(標準フォーマット②)

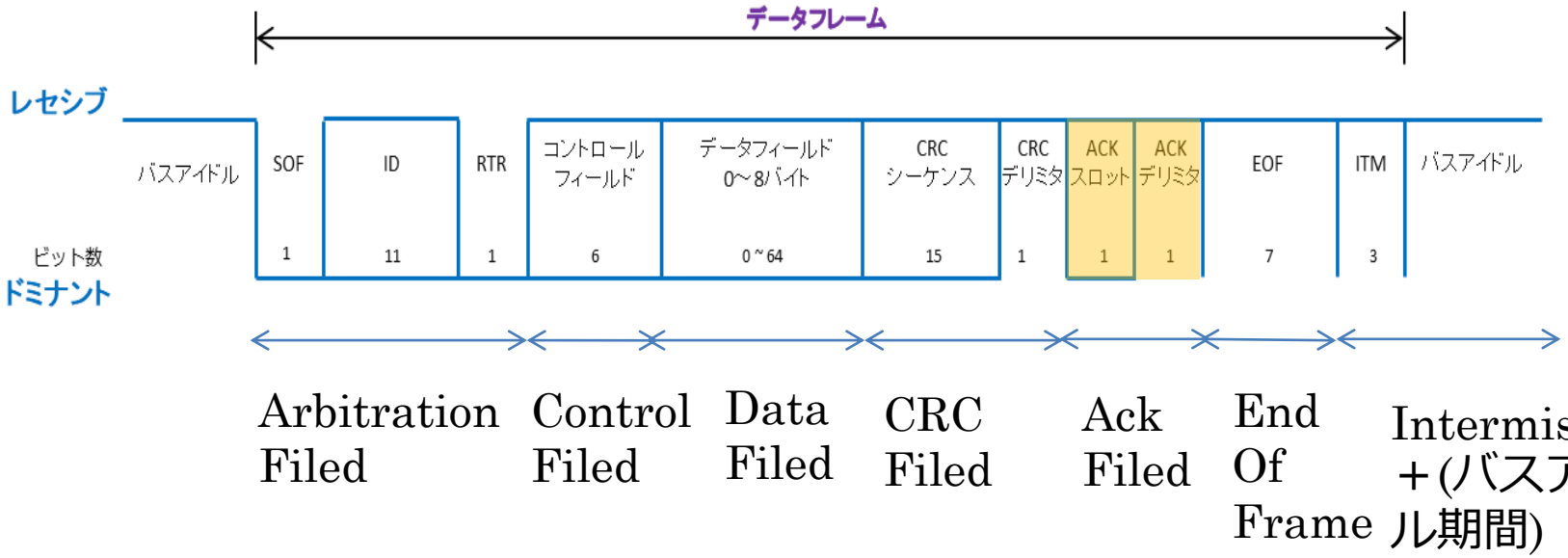


・Data Filed
→送受信させるデータを指定。MSBから送信する。

・CRC Filed
→Arbitration Filed ～ Data Filedの値より演算したCRC値を格納する。
受信したノードはここの値を見て、受信が正常にできたかを判断する。

CAN通信プロトコルの詳細

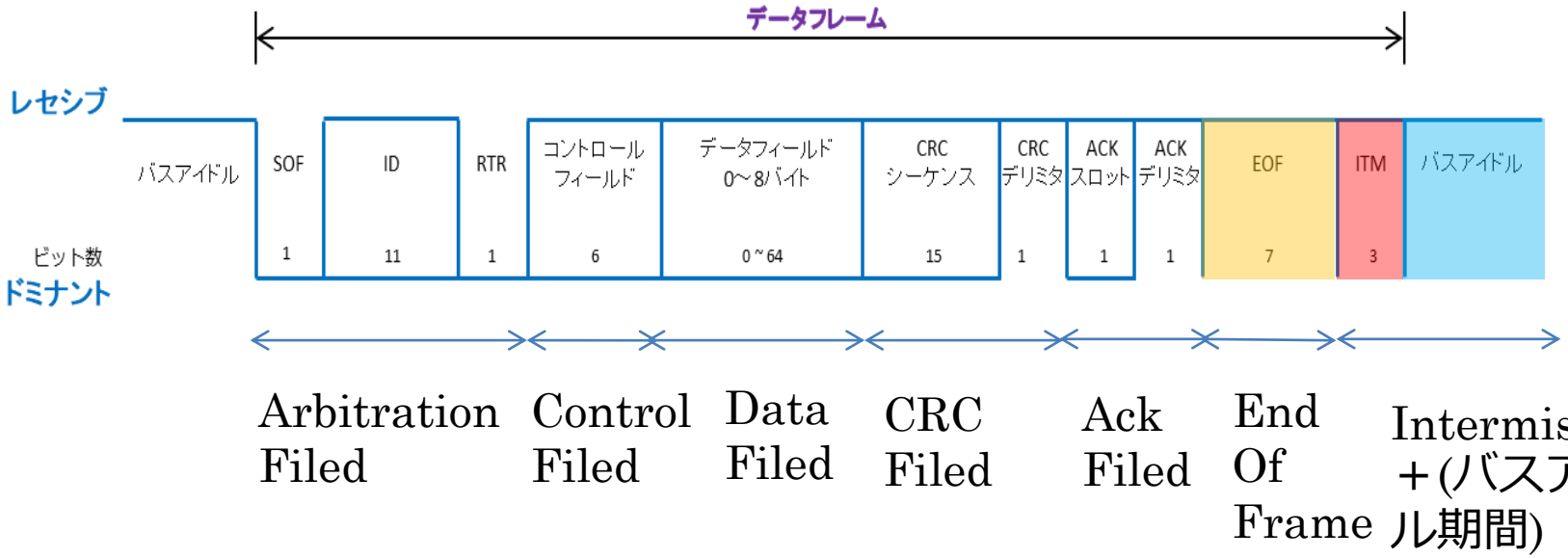
（2） データフレーム⑤(標準フォーマット③)



・Ack Filed
→受信OKかどうかを返すフィールド。送信ノードは常にレセシブを設定。
受信ノードは正常に受信できた場合は応答としてドミナントを送信する。
送信ノードはAck Filedの値をモニタして、正常に受信できたかを確認する。
※どれか一つのノードがAck応答を返した場合、送信ノードは正常送信できたと判断する。

CAN通信プロトコルの詳細

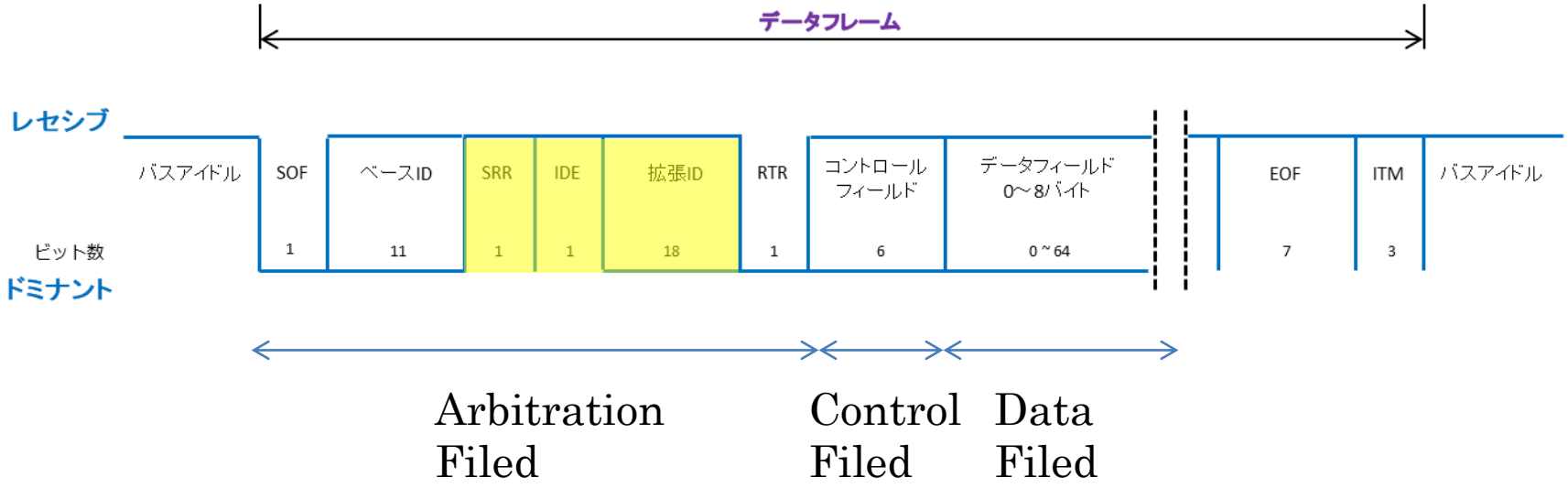
（2） データフレーム⑥(標準フォーマット④)



- End Of Frame
  →データフレームの終わりを示す。7ビットのレセシブを固定。
- ITM
  → End Of Frame後に3ビットレセシブ固定を送信する。
  データフレーム/リモートフレームはこの間、次のフレーム送信はできない。
- バスアイドル
  →何も送信するデータがない場合、CANバスはレセシブ固定となる。

CAN通信プロトコルの詳細

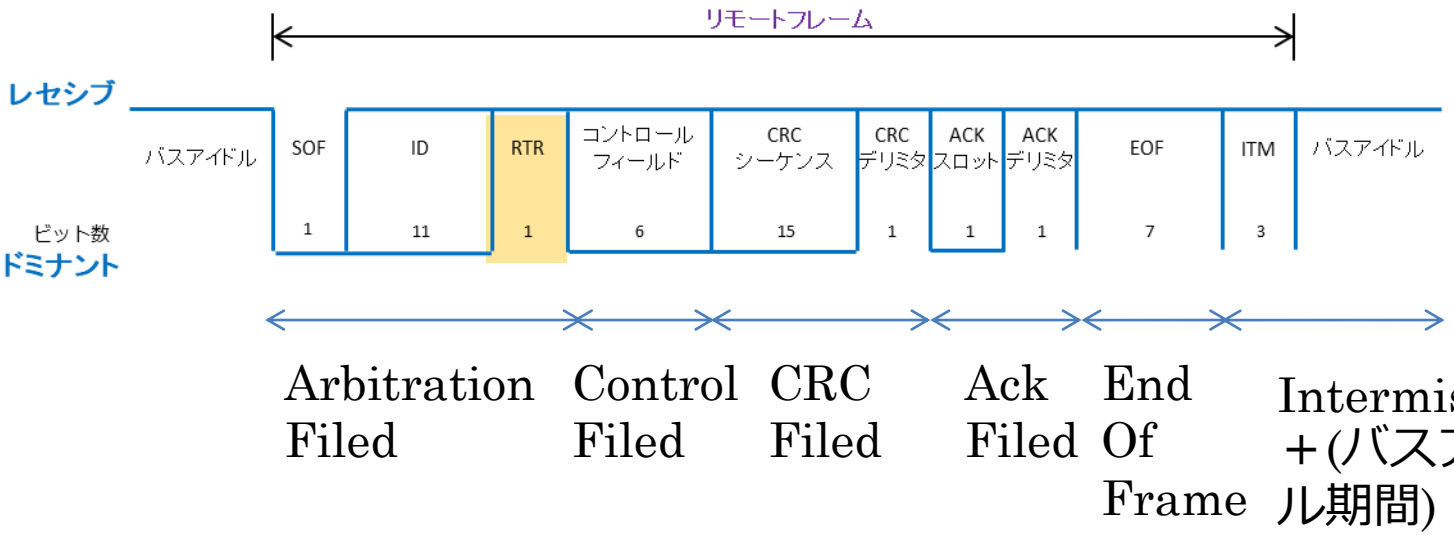
（2） データフレーム⑦(拡張フォーマット)



・標準フォーマットとの違いはArbitration Filedの違い
IDを示すことができる領域が18bit分増加
→標準フォーマットでは2048種類しかCANフレームを識別できなかったのが、
拡張フォーマットでは540万種類ほどの識別が可能。
※プロトコルバージョン2.0Bから対応。

CAN通信プロトコルの詳細

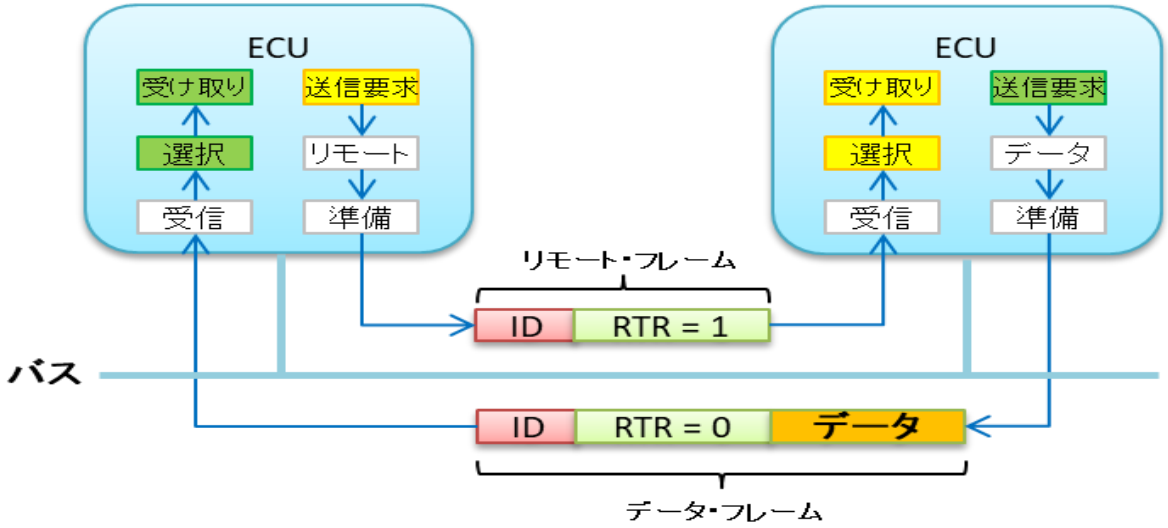
（3） リモートフレーム①



・データフレームとリモートフレームの違いは以下の2点のみ
→Arbitration Filedの最終ビット(RTR領域)がレセシブ(1)
データフレームではドミナント(0)
→ここでこのフレームがデータフレームかリモートフレーム
かを判断している。
→Data Filedがない

CAN通信プロトコルの詳細

（3） リモートフレーム②

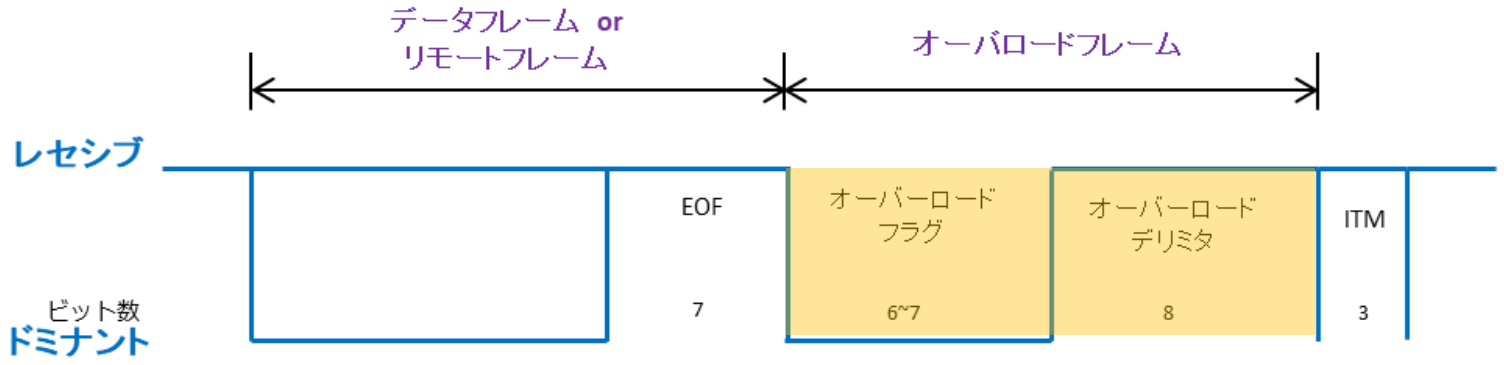


・リモートフレームの役割
→データの要求を行うためのフレーム
リモートフレームを受けたノードは、該当するIDの
データフレームを送信する。
※昨今のCANバスでは、頻繁にデータをやり取りするケースが
多いため、リモートフレームを使用する機会は少ない。
(AUTOSAR規格のCANドライバーはリモートフレームは送信禁止、
受信しても無視する仕様となっている)

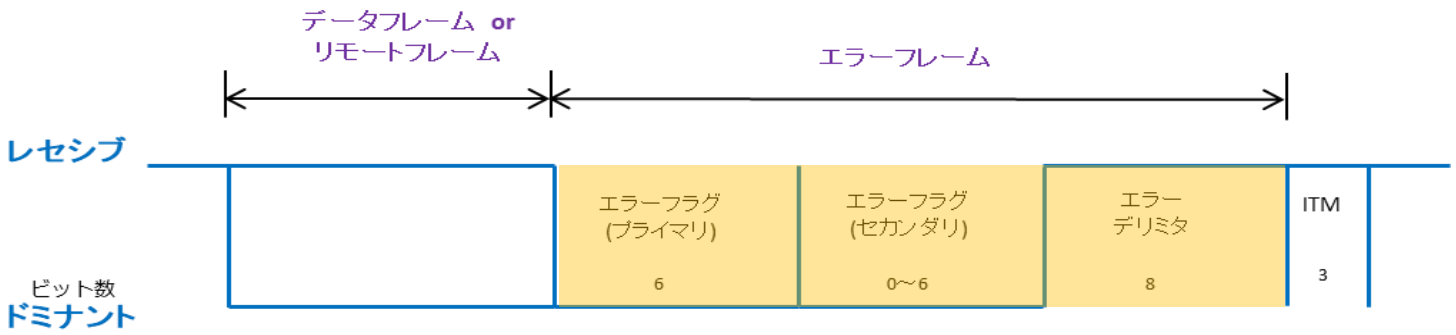
CAN通信プロトコルの詳細

（4） オーバーロードフレーム／エラーフレーム①

★オーバーロードフレーム



★エラーフレーム



CAN通信プロトコルの詳細

（4） オーバーロードフレーム／エラーフレーム②

・オーバーロードフレームの役割
→CANコントローラが、前回のフレームの処理が終わっていないことを理由に、次のフレームの開始を遅延させたい場合に送信する。
※昨今はマイコン性能が向上しているので、オーバーロードフレームが発生することはまずない。

・エラーフレームの役割
→各種エラーが発生時に送信され、間近の送信を中断させる役割がある。

両者ともビットスタッフィングのルールに違反する形でフレームを出す！！！
→固定フォームを破壊する形となり、間近の送信が中断される仕組みとなっている。

CAN通信プロトコルの詳細

（５）ＣＡＮのエラーチェック①

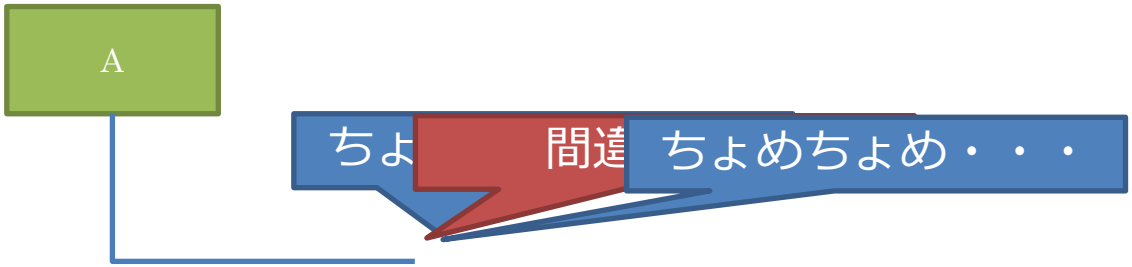
エラー種別	監視するノード	チェック内容
ビットモニタリング	Tx	送信したデータとバス上のデータが異なっていないかをチェック
アクナレッジチェック	Tx	Ack応答があるかをチェック(正常受信したノードのチェック)
CRCチェック	Rx	受信ノードが演算したCRC値と受信したCRC値の一致をチェック
フォームチェック	Rx	固定フォーマットに違反していないかをチェック
スタッフチェック	Rx	ビットスタッフィングに違反していないかをチェック

・上記チェックに引っかかった時、即時にエラー検出したノードがエラーフレームを送信する。

CAN通信プロトコルの詳細

AISIN アイシン・ソフトウェア株式会社

（５）ＣＡＮのエラーチェック②



・間違えるとエラーフレームが出る。その後送信ノードはデータ再送をする。

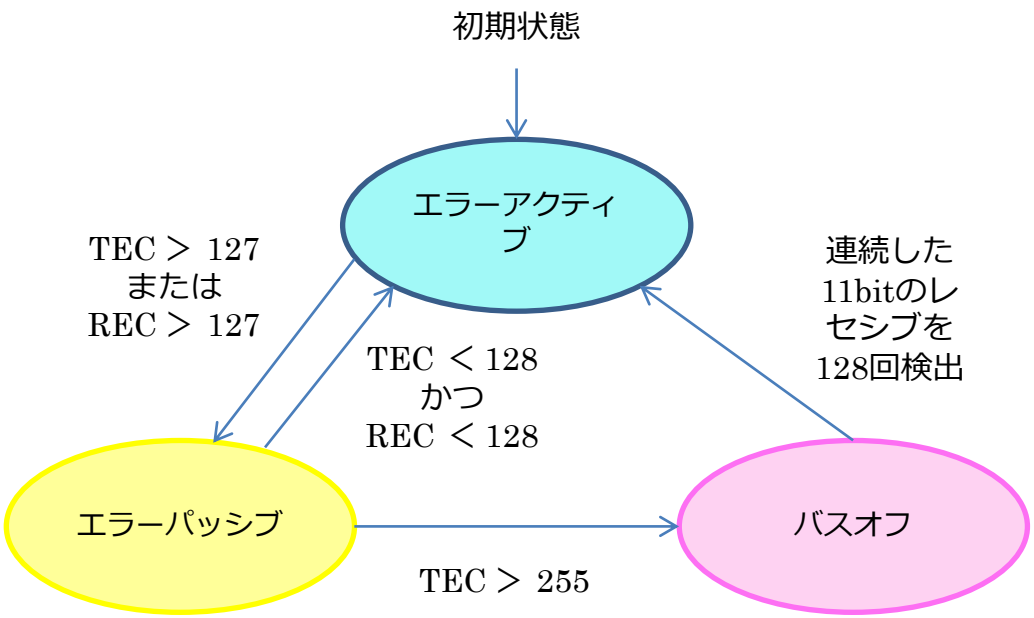
→何度もエラーが発生するノードがあると、同じ送信が繰り返されてしまい、バス負荷があがってしまう・・・

エラーカウンタ

→自身のポンコツ具合をカウントする。

CAN通信プロトコルの詳細

（５）ＣＡＮのエラーチェック③



・各ノードはエラー発生頻度をエラーカウンタという形で保持している
　エラーカウンタが一定値を超えると送信を遠慮するような制御を行う
・エラーカウンタは2種類が存在
　TEC：送信エラーカウンタ
　REC：受信エラーカウンタ
・バスオフへ遷移する条件はTECが規定値を超えた場合のみ

CAN通信プロトコルの詳細

（５）ＣＡＮのエラーチェック④

状態	振る舞い
エラーアクティブ	・エラーを検出した場合、**アクティブエラーフラグを送信**する ・アクティブエラーフラグが送信されると、**送信フレーム処理が強制的に終了させられる**
エラーパッシブ	・エラーを検出した場合、**パッシブエラーフラグを送信**する ・パッシブエラーフラグはアクティブエラーフラグのように**送信フレーム処理を強制的に終了させることはできない** ・通常はITM後に送信可能だが、エラーパッシブ状態ではITM後、**8bitのレセシブを経ないと送信できない。** **→エラーアクティブのノードより送信できる可能性が低い** ・Ackエラーを検出しても送信エラーカウンタをインクリメントしない。
バスオフ	・バスには影響を与えない（ネットワーク遮断）

ポイント

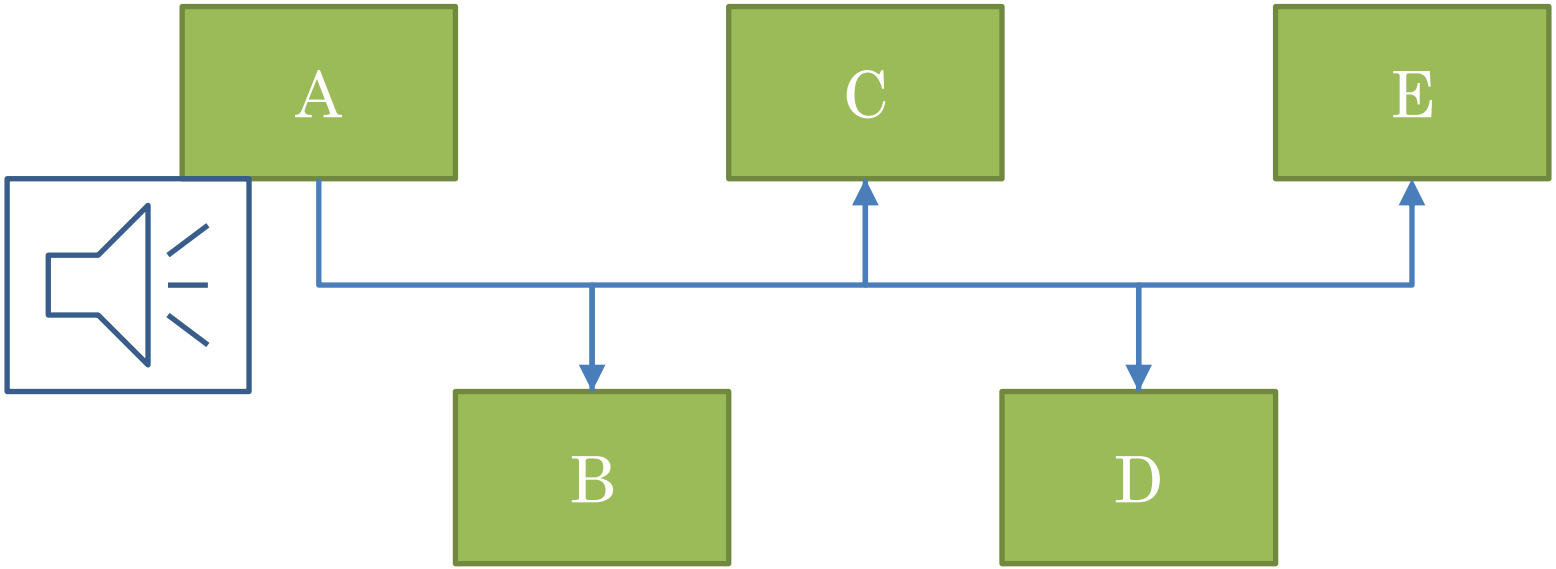
①他に割り込んで送信を止める権利があるのはエラーアクティブのノードのみ

②Ackエラーが繰り返されただけではバスオフまで遷移しない

③エラーパッシブになると送信がしにくくなる、バスオフだと一切の送信が不可となる。

CAN通信プロトコルの詳細

（6）通信調停①

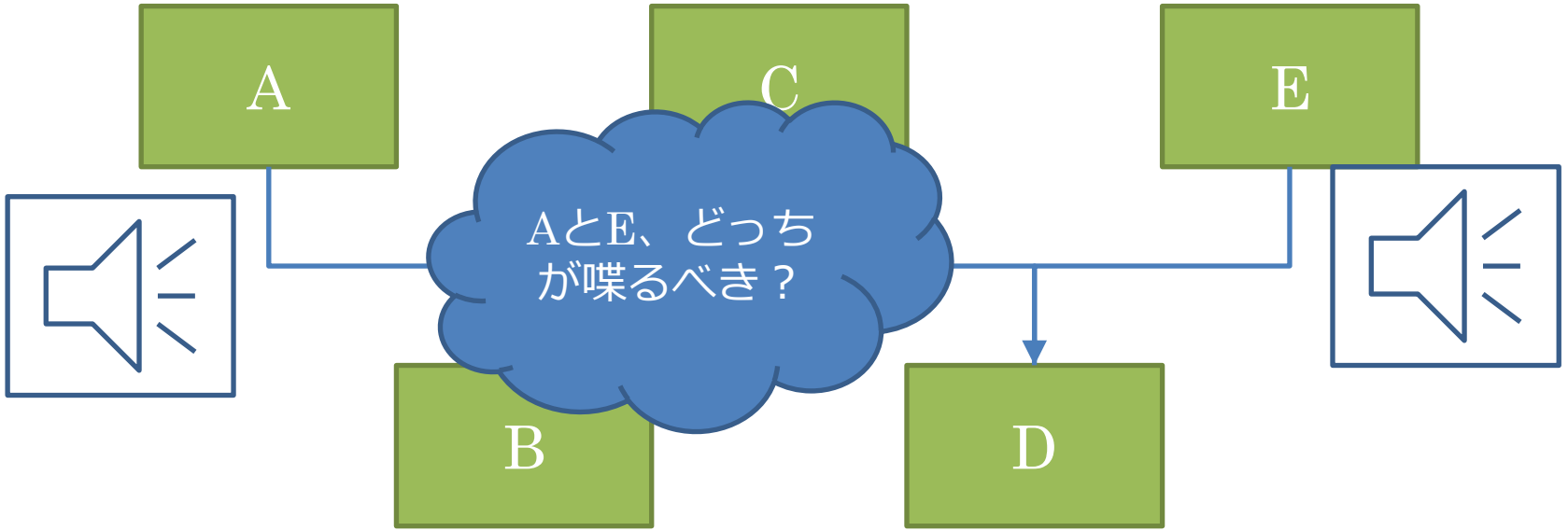


・CANで採用しているCSMA/CA方式では、
あるノードが送信を行っている最中、他のノードは送信できない

・上記図では、ノードAがしゃべっている間は、ノードB~Eは
受信ノードとして振舞わなければならない。

CAN通信プロトコルの詳細

（６）通信調停②



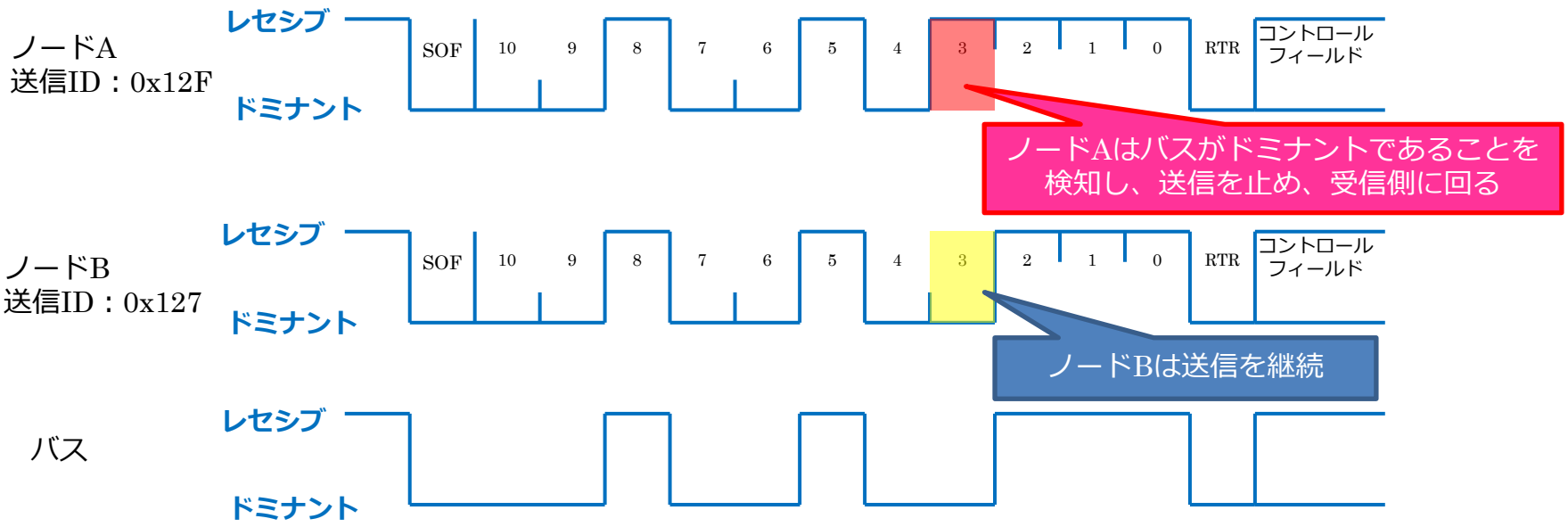
・CANはマルチマスタ方式であるため、同時に複数のノードが送信を始める可能性がある。

複数ノードが同時に喋り始めた場合、誰が喋るかを決める仕組みが「調停(Arbitration)」です。

CAN通信プロトコルの詳細

AISIN アイシン・ソフトウェア株式会社

（6）通信調停③



- CANバスに別々のノードがドミナントとレセシブを同時に送信した場合、バス上はドミナントとなる。この時レセシブを出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する。

CAN通信プロトコルの詳細

（6）通信調停④

- 複数ノードから同時にフレームが送信された場合、ドミナントレベルが優先されるため、最も優先順位が高い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る。**

- IDの割り当てについては、ネットワーク設計者が自由に割り当てできるが、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**ことを考慮するべき。

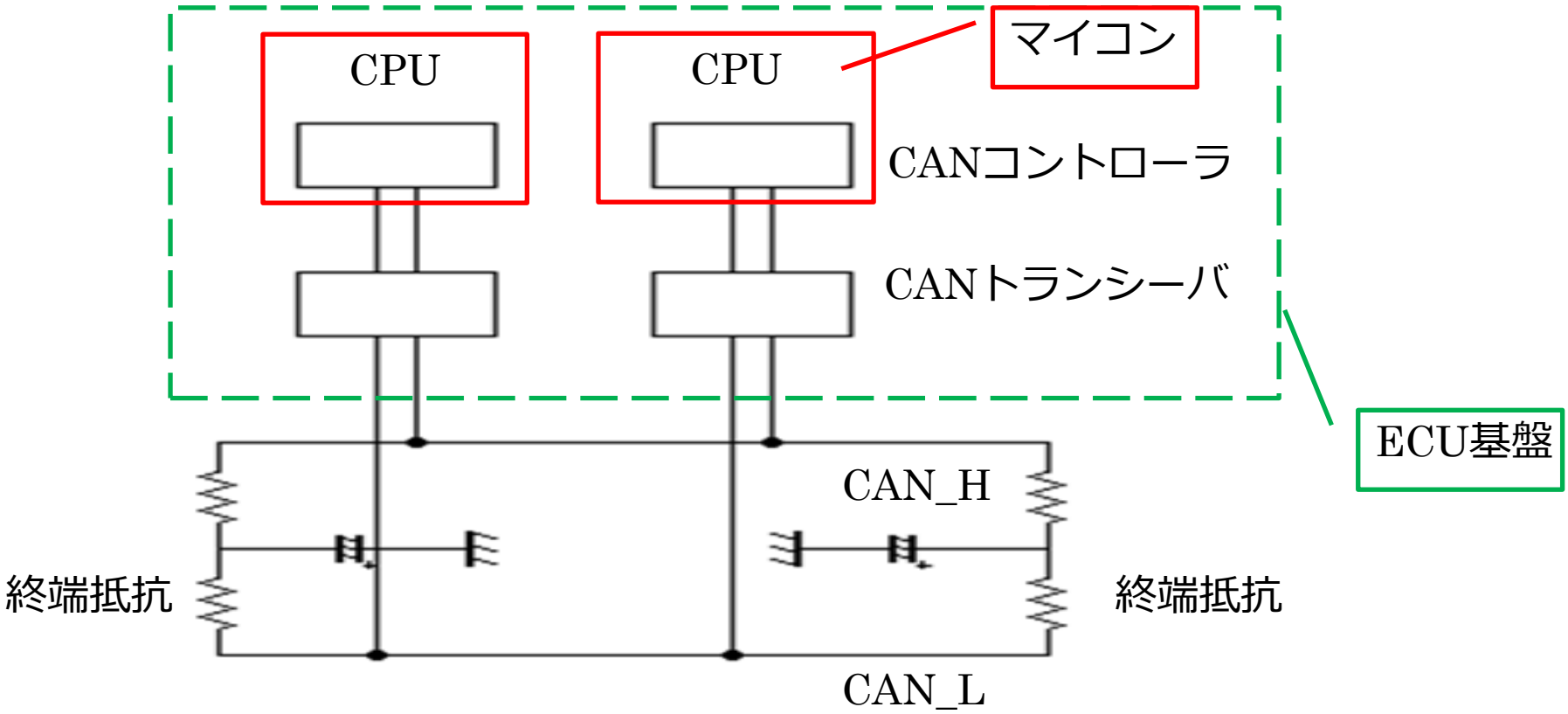
- 同じIDのデータフレームとリモートフレームが同時に送信された場合、IDのみでは調停は行えないため、RTRも使用して通信調停の判定を行う。データフレームではRTRはドミナント、リモートフレームではRTRはレセシブであるため、**データフレームが優先される**ことになる。

- 標準フレームと拡張フレームで、最初の11bitのID領域が全く一緒だった場合は、IDEがドミナントの**標準フォーマットのフレームが優先**される。

4.CANプラットフォーム

CANプラットフォーム

（1）ＣＡＮノード/バスの構成①



・上記はCAN-Cの構成
①CANコントローラ
②CPU
③CANトランシーバ
④CANバス(CAN_H、CAN_L、終端抵抗で構成)

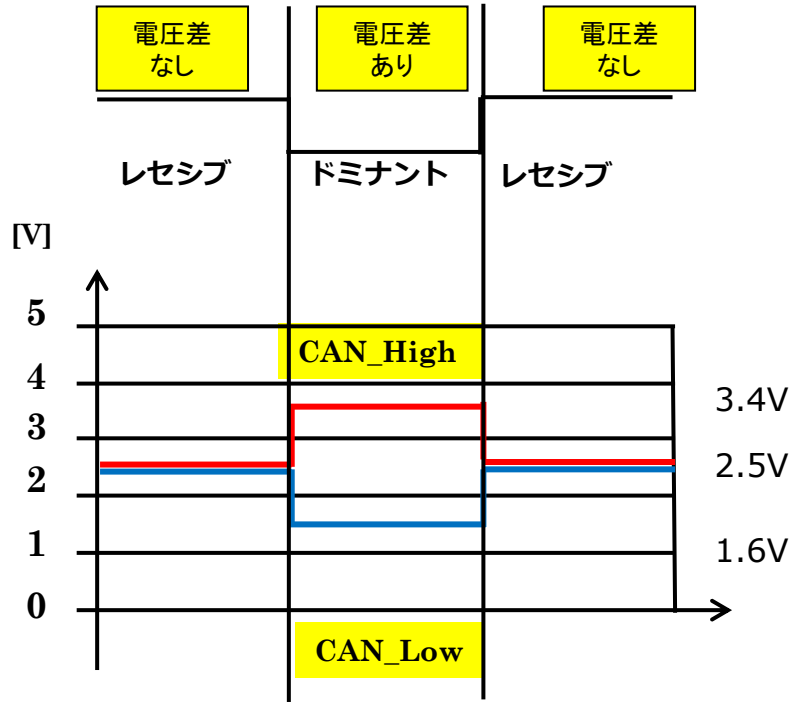
CANプラットフォーム

（1）ＣＡＮノード/バスの構成②

名称	対応内容
CANコントローラ	・通信調停 ・スタッフビットの付加 ・CRCチェック ・エラー検出、通知 ・フィルタリング
CPU	・どのメッセージを受信したかの判断 ・どのメッセージを送信するかの判断
CANトランシーバ	・CAN_H、CAN_Lの物理的電位差からバスレベルを判断、受信データに変換 ・送信データを物理的なバスレベルに変換
CAN_H	・ドミナント時に高電圧へプルされるライン
CAN_L	・ドミナント時に低電圧にプルされるライン
終端抵抗	・反射電圧の抑制

CANプラットフォーム

（1）ＣＡＮノード/バスの構成③



ドミナント時にレセシブより高い電圧が測定されるラインがCAN_H
ドミナント時にレセシブより低い電圧が測定されるラインがCAN_L
※CANの特徴 (8)の再掲

CANプラットフォーム

（２）ＣＡＮコントローラ①

・「外付け」と「内蔵」の2つのパターンがある。
※但し、現在車載用マイコンとして販売されているほとんどのマイコンは
CANコントローラを内蔵している。

・CANコントローラで送受信するメッセージをマネージメントするための
方法に「BASIC-CAN」と「FULL-CAN」の2種類がある。

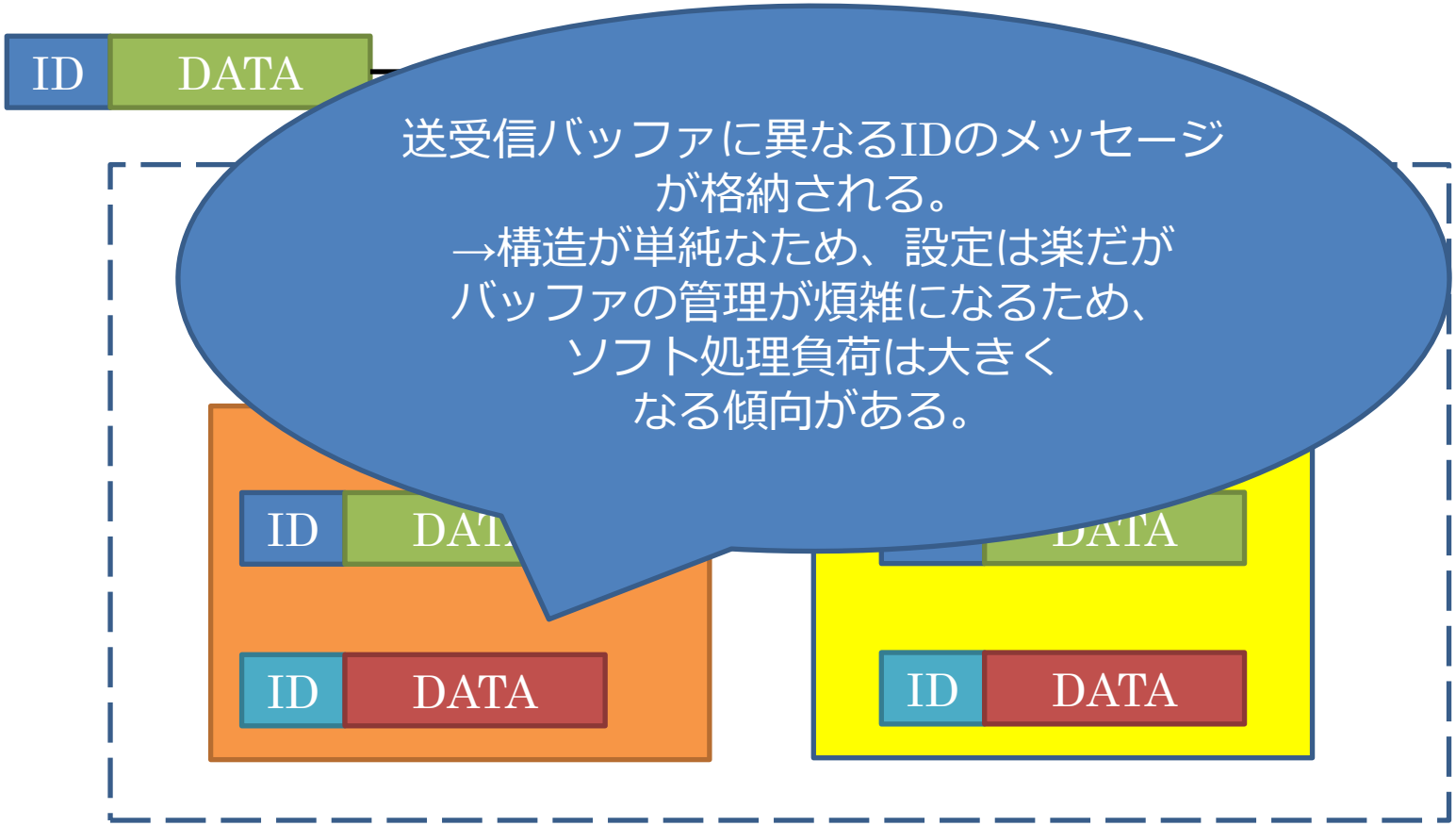
「BASIC-CAN」と「FULL-CAN」の大きな違いは、メールボックスを
一つで対応するか、メッセージごとにメールボックスを設けるかの違いがある。

・通信速度の設定はここに対して行う。
一般的にはネットワークで統一する必要があるため、ネットワークの仕様を
確認の上、設定が必要。

CANプラットフォーム

（2）ＣＡＮコントローラ②

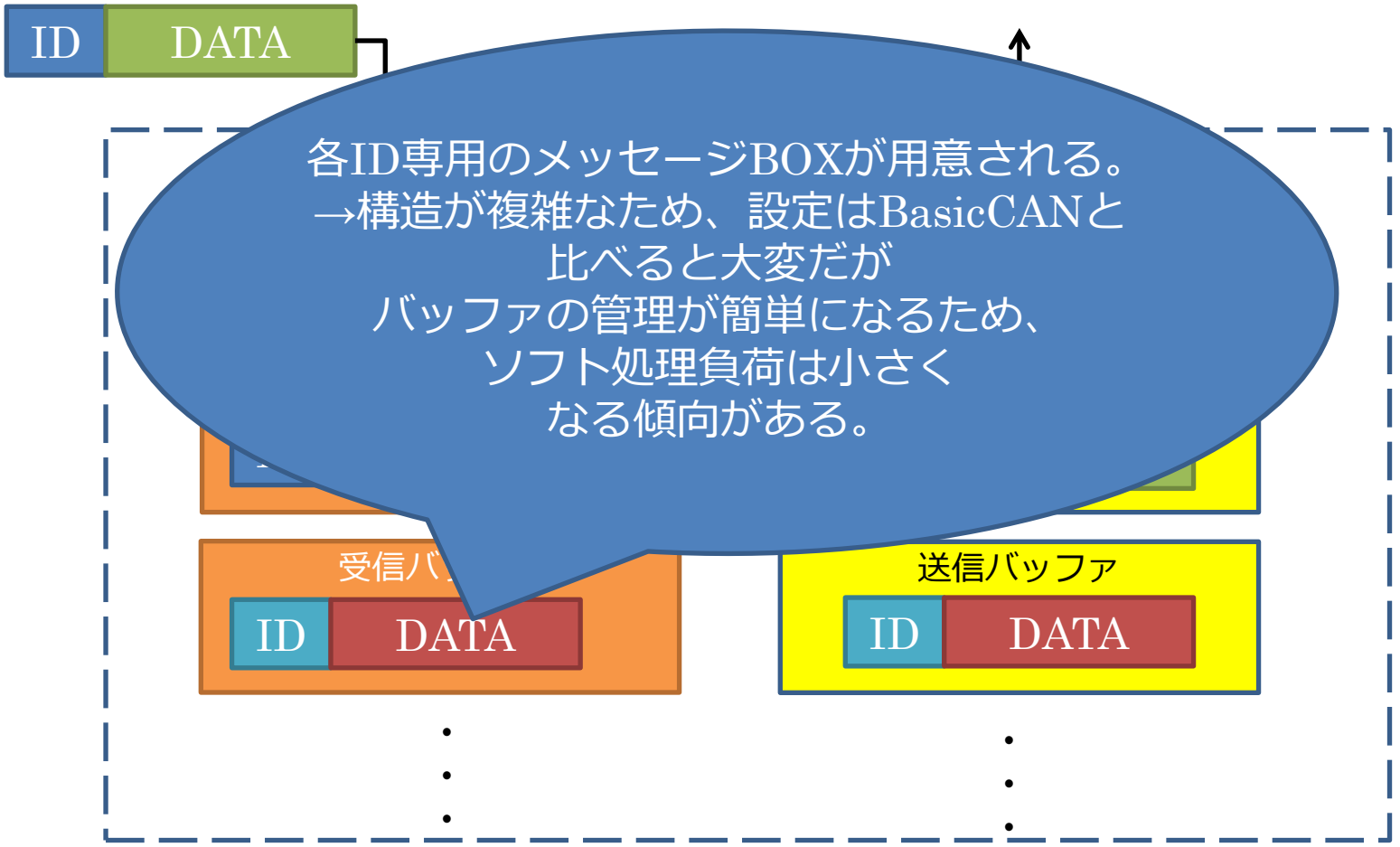
・「BASIC_CAN」のイメージ



CANプラットフォーム

（2）ＣＡＮコントローラ③

・「FULL_CAN」のイメージ



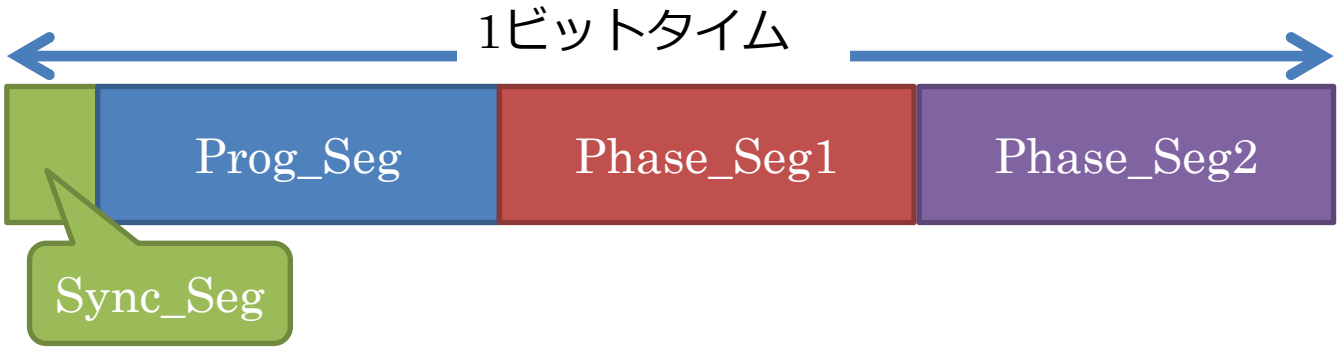
CANプラットフォーム

（2）ＣＡＮコントローラ④

・通信速度
→データ転送速度(bps：ビット毎秒)とビットタイムの設定が必要

・ビットタイム
→1ビットの信号を各セグメントに細分化するための設定。
ビットタイムを設定することによりCANバスの状態に応じて
ドミナント/レセシブを認識するポイントを操作可能となり、
ビットの認識の正確性とバス遅延等に対する障害耐性を高めている。

ビットタイムは「Sync_Seg」「Prog_Seg」「Phase_Seg1」
「Phase_Seg2」の4種ある。



CANプラットフォーム

（2）ＣＡＮコントローラ⑤

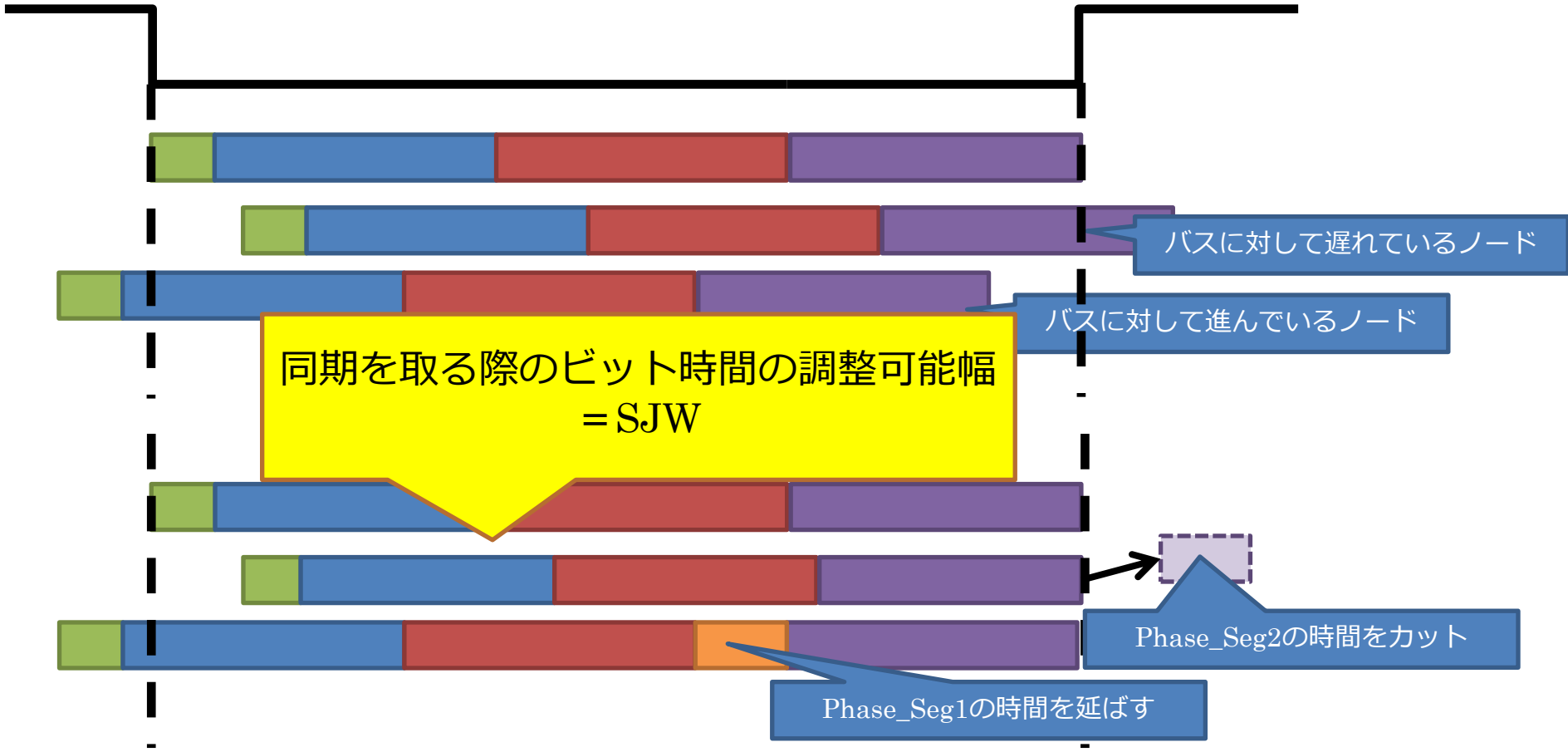
名称	役割	時間
Sync_Seg	バスの同期を取る役割 ※ドミナント→レセシブの変化タイミングとここが一致することが同期	1TQ
Prog_Seg	ネットワーク遅延を吸収するための区間	1TQ ～ 8TQ
Phase_Seg1	ネットワークに対して自ノードが進んでいる場合の調整区間 ※マイコンによってはProg_Seg＋Phase_Seg1を設定するケースもある。	1TQ ～ 8TQ
Phase_Seg2	ネットワークに対して自ノードが遅れている場合の調整区間	2TQ ～ 8TQ

※TQ：Time Quanta：セグメントごとに細分化した時間の単位
　　1 TQの時間が長いほど、バス遅延に対する耐性が増加する。

CANプラットフォーム

（2）ＣＡＮコントローラ⑥

・SJW
　→バス遅延を吸収する役割



（3）ＣＡＮトランシーバ

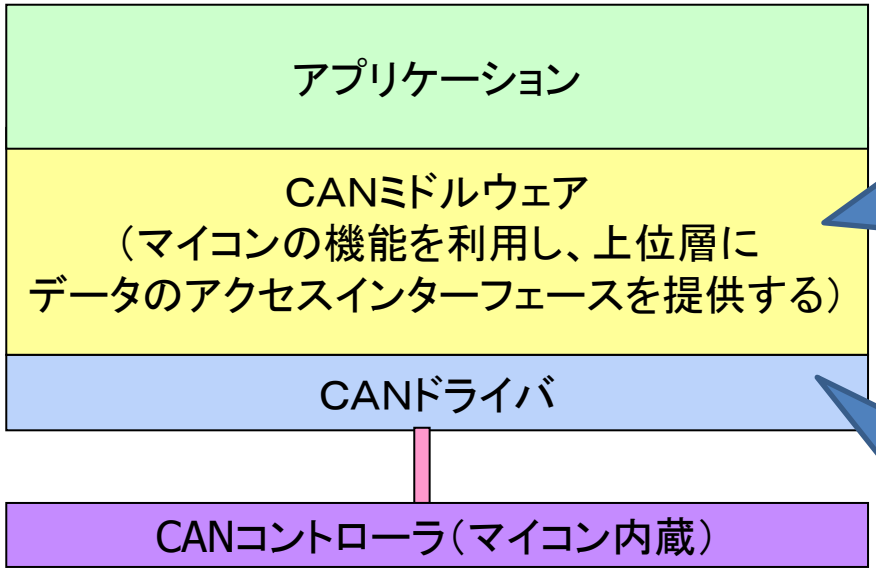
・従来のトランシーバは物理的な信号と論理信号の切り替えが主な役割で、
単純な機能のみ実装されていることが多かった。

・昨今はCANトランシーバも高機能化されている。
CANトランシーバがOFF中に特定のシグナルを受けた場合だけ
CPUへ割り込み信号を通知する機能(Selective Wakeup機能)を持ったもの
(ISO 11898-2:2016で規格化)やLINトランシーバも一緒に内蔵されている
複合トランシーバも出ている。
※但しコスト面で従来の単純なトランシーバが採用されることも多い。

CANプラットフォーム

（４）ＣＡＮソフトウェアの構成

・CANソフトウェアの構成例



CANミドルウェアの役割
- アプリケーション制御プログラムとのデータ受け渡しインターフェースの提供
- メッセージの受信及びその通知
- メッセージの送信及びそのタイミング制御
- 通信異常時の制御
- 通信開始、停止の制御

CANドライバの役割
- マイコンレジスタの設定を行い、CANコントローラを制御する。
  ※通信速度(ボーレート)、アプセプタンスフィルターの設定、バッファの設定等
- 受信スロットからデータを取り出す。
- 送信データを送信スロットにセットする。

5.CAN FDの概要

CAN FDの概要

（１）CAN FDが登場した背景

・昨今の自動車では、安全性・快適性の向上、電動化、自動運転、
コネクティッドへの対応、セキュリティへの対策が必要となっている。
このためには従来以上にネットワーク上でやり取りするデータが増加。

また車載ECUではCANを使ってソフト書き換えを実施しているが、
書き換えるソフトウェアが年々巨大化しているため、書き換えるのに
必要な時間が増加している。

→これに対応するには現在のCANの通信速度、データ長では不十分

→しかし既存のCANバスをFlaxRay等に置き換えるのはハード的にも
ソフト的にも大きな手間

→CANを拡張する形でより高速な通信、データ長に対応可能な
CAN-FD(CAN with Flexible Data Rate)の仕様が2012年に策定された

CAN FDの概要

（２）CAN FDの特徴①

特徴	CAN-FD
データ長	CANのMax 8Byte → Max **64Byte**
通信速度	アービトレーション領域とデータ領域で通信速度変更可 データ領域なら車載用途でも**2Mbps**も可能 ※CANは車載用途だと500Kbpsが限界
物理層	**トランシーバはCAN-FD対応が必要**。 バスはCANと同じ。 (但しEMC対策はCANより強化が必要)
上位層	**コントローラはCAN-FDが必要**。 ソフトウェアは大きな変更不要。 ※コントローラの設定変更、送受信データの64byte対応は必要。

※CANとCAN-FDの混在は可能

CAN FDの概要

（２）CAN FDの特徴②

プロトコル	CAN-FD
Arbitration Filed	RTR領域が0固定 **→リモートフレームは無し**
Control Filed	CANかCAN-FDかを区別するデータが追加 **通信速度変更点**の追加 DLCのデータフォーマットがCANと変更
Data Filed	CANの0~8Byte → **0~8/12/16/20/24/32/48/64**Byteが可 ※DLCは上記のみ指定可能。つまり13byteのデータ送信はできない。
CRC Filed	Stuff Countの追加 15bitのCRC → 17bit or 21bitにCRCに変更 Max 2bitのCRC Delimiter ※CRC Delimiterまでがデータ領域の通信速度対象
Ack Filed	1bitのAck → Max 2bitのAck
EOF	EOFは7ビットのレセシブで変更なし

6.LINの概要

LINの概要

（１）LINの特徴

項目	説明
通信方式	マスター/スレーブ方式 →1つのマスタと複数のスレーブで構成(推奨最大16)
伝送路	1線式
通信速度	最大20Kbps
通信コントローラ	UART(簡単・安価を実現するため、多くのマイコンに搭載しているUARTを利用)
スリープ/ウェイクアップ	スレーブはマスタからのスリープ要求、一定時間通信が無い場合にスリープする。マスタからのウェイクアップ要求、また自ノードの要因でウェイクアップする。
同期方式	マスタから送信するフレームのヘッダに同期用の基準となるSync Fieldをスレーブが受け取ることで同期 →このため、マスタノードのクロックの精度をしっかりしておけば、スレーブ側は多少精度が低い発振子を利用しても良いことになっている。

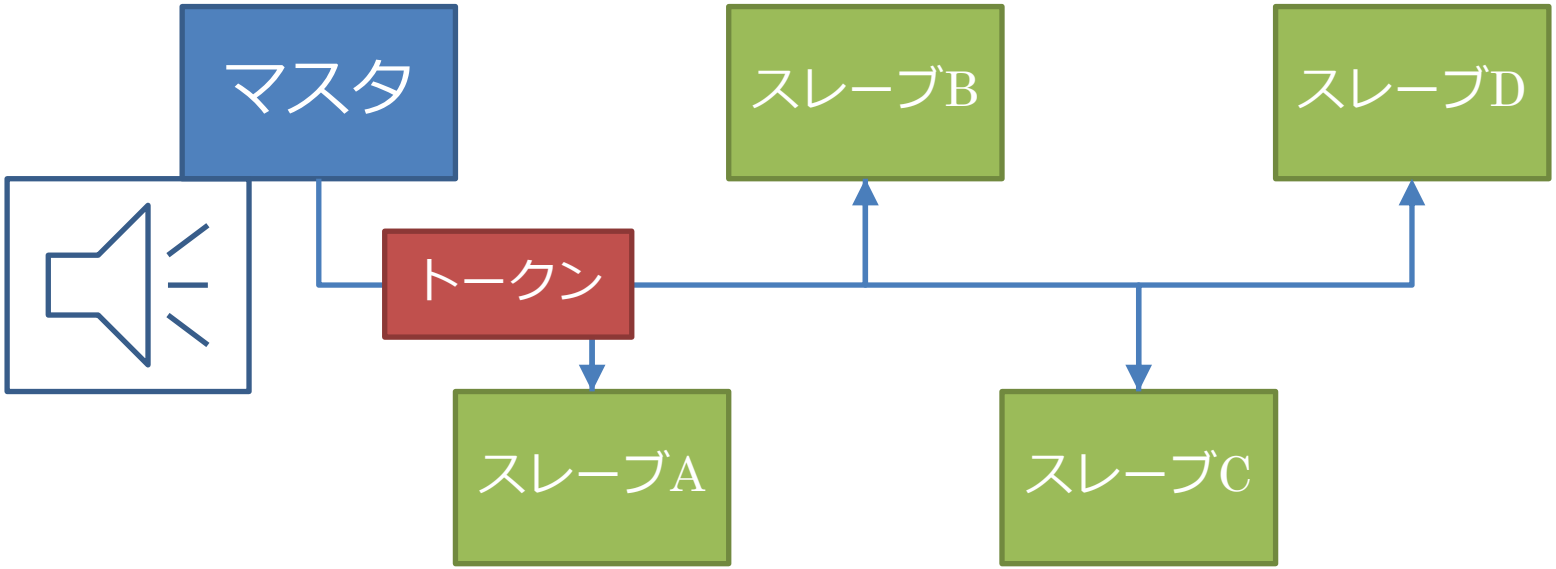
LINの概要

（２）LINとCANの違い

	CAN	LIN
通信形態	バス型	バス型
通信方式	マルチ・マスター方式	**マスター/スレーブ方式** →CANのような調停は不要
通信速度	規格上はMax 1Mbps	Max **20Kbps**
伝送路	2線式	**1線式**
Data Filed	MSBから送信	**LSBから送信**
ドミナントとレセシブ	０がドミナント １がレセシブ	０がドミナント １がレセシブ
伝送路符号	NRZ方式	NRZ方式
同期方式	レセシブ→ドミナントの立下りで同期	**マスタが送信するヘッダ受信時**
誤り補正	CRC	**パリティ(ヘッダ)・チェックサム(レスポンス)**

LINの概要

（3）マスター/スレーブ方式①

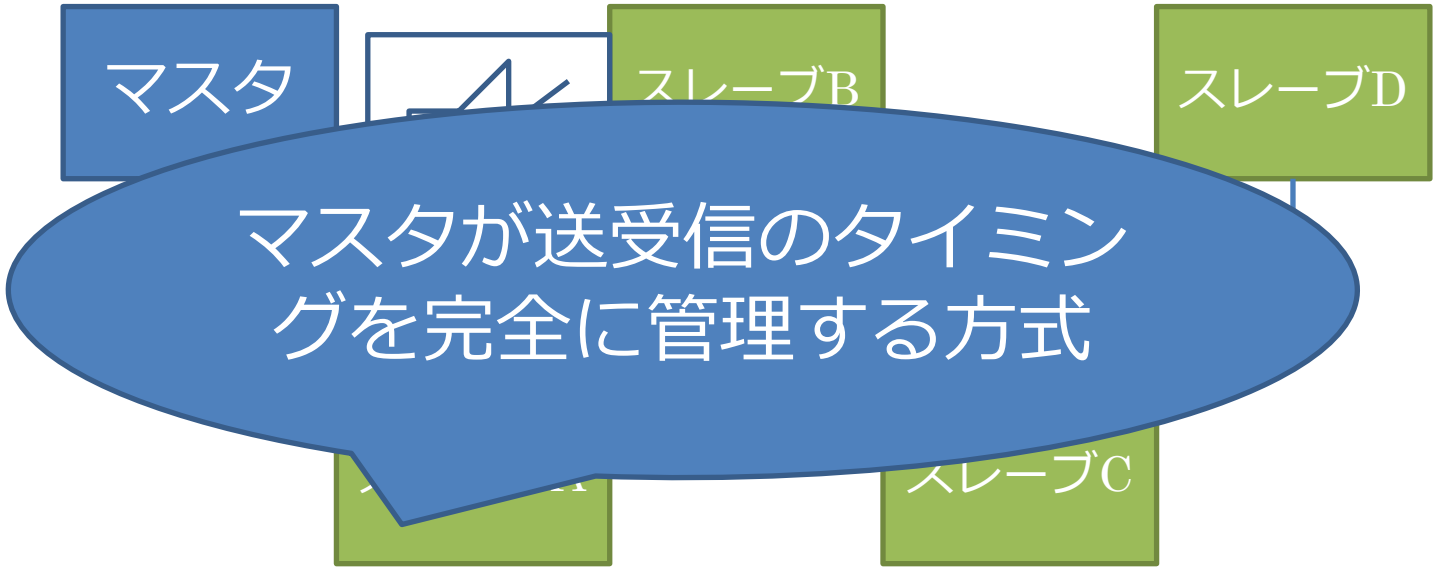


・マスタスレーブ方式では、まずはマスタがトークンと呼ばれるものを全スレーブに通知します。

・トークンには「誰が」「誰に」送信をするかが記述されています。

LINの概要

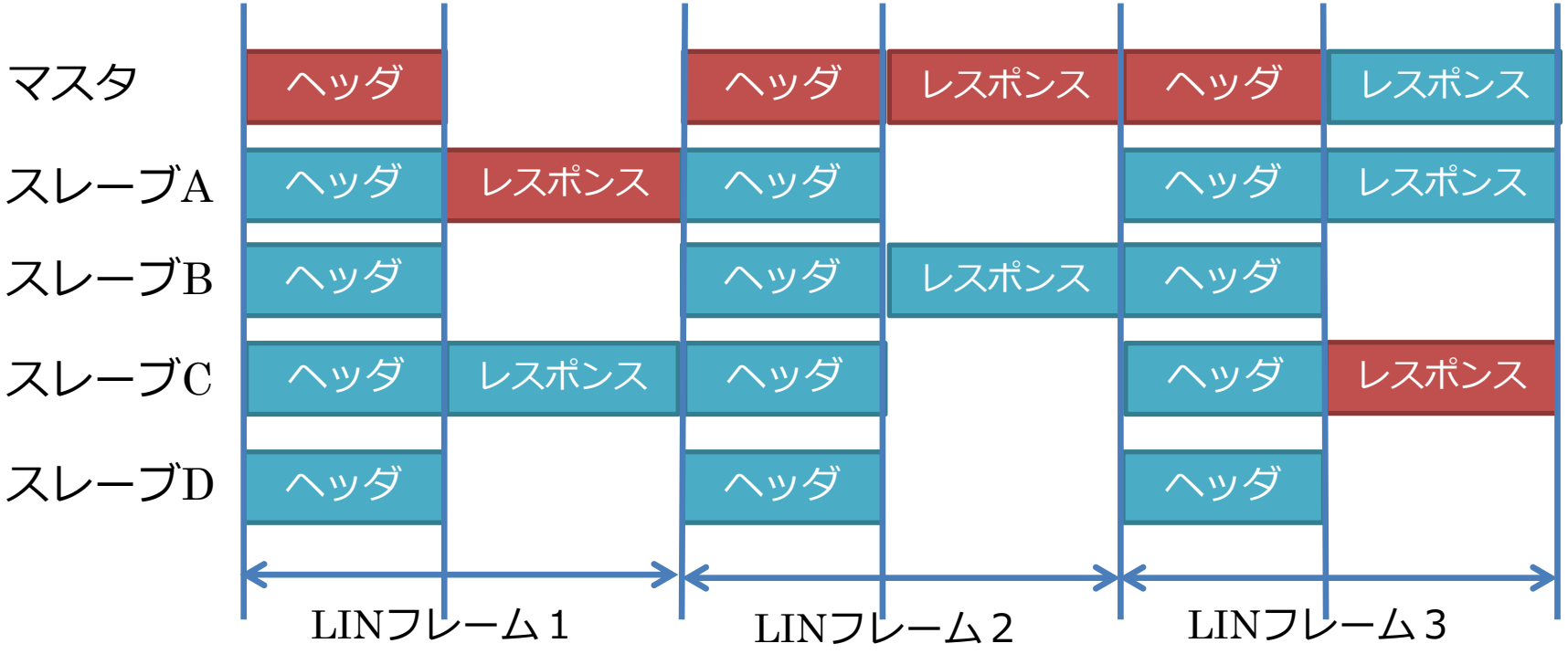
（3）マスター/スレーブ方式②



・次にトークンで送信指示されたノードがデータを送信します。
そして、トークンで受信指示されたノードがデータを受信します。
※上記ではトークンに「スレーブBが」「スレーブA」にデータを
送信をすることが記述された場合の例を示しています。

LINの概要

（４）LINプロトコル概要①

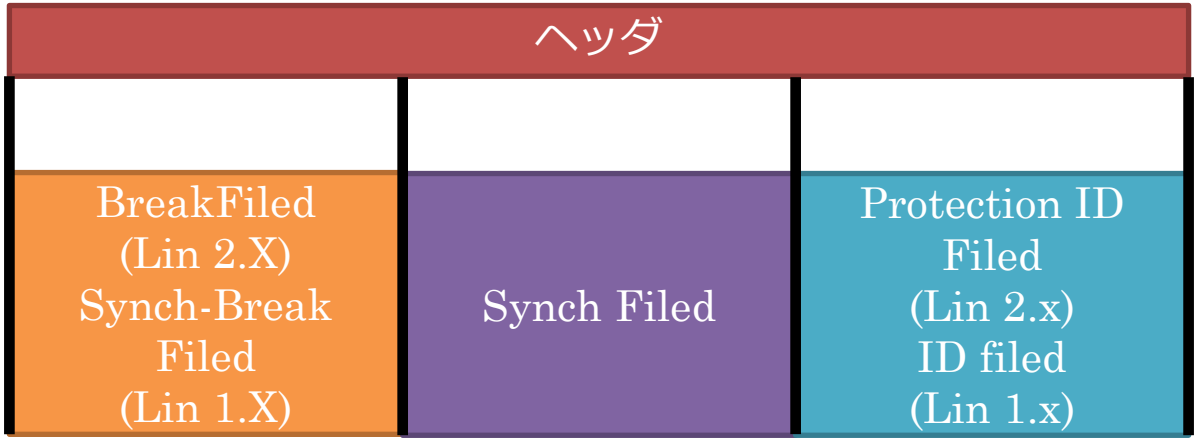


・一つのLINフレームは一つのヘッダと一つのレスポンスで構成されます。

・ヘッダは「トークン」の役割を果たします。ヘッダに書かれた内容に沿ってレスポンスでデータを送受信します。

LINの概要

（４）LINプロトコル概要②



フィールド名	説明
Break Filed	フレームの開始を明示する領域 13bitのドミナントと、1bitのレセシブで構成
Synch filed	同期のための信号を送信する領域 。0x55を送信
Protection ID field	6ビットのIDと2ビットのパリティを送信。

※break Filed以外は先頭にスタートビット(0)
終端にエンドビット(1)が付与される

LINの概要

（４）LINプロトコル概要③

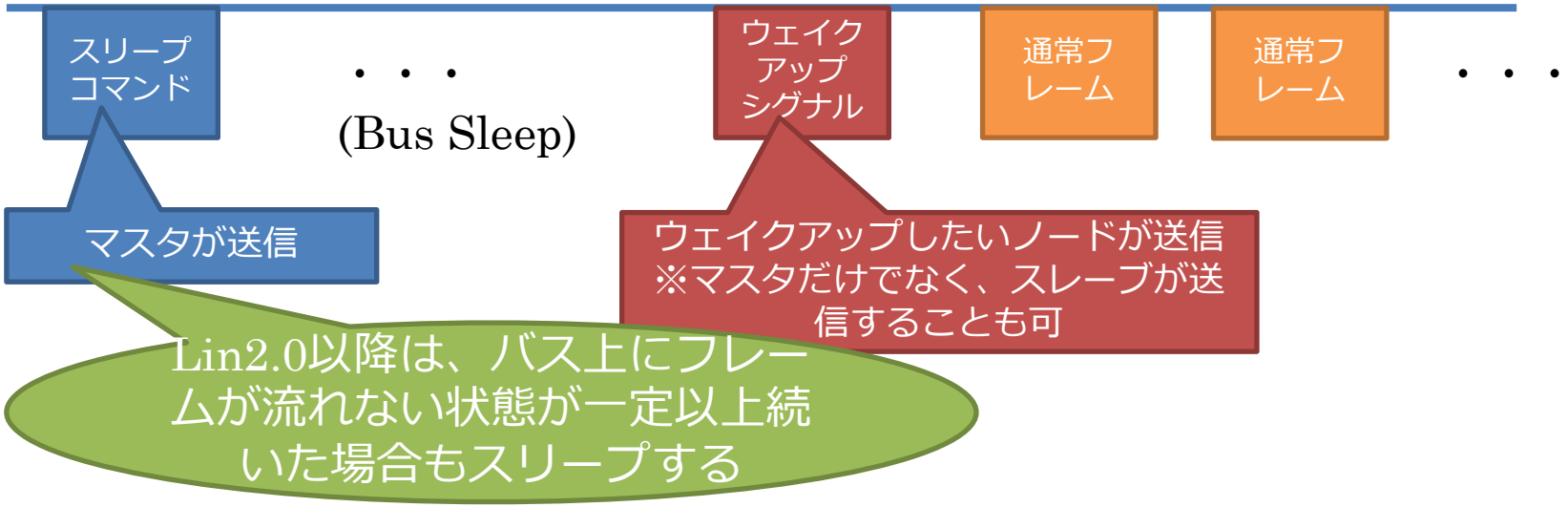
フィールド名	説明
Data Filed	データを1byteごとに送信。LSB(bit0)から送信する。
Check Sum	レスポンスデータの誤り検知用チェックサムを送信 チェックサム方式は2種類あり ・クラシック：チェックサム対象が「全データ」(Lin1.x) ・エンハンス：チェックサム対象が「保護ID＋全データ」(Lin 2.0で追加)

※全フィールドの先頭にスタートビット(0)
　終端にエンドビット(1)が付与される

LINの概要

（４）LINプロトコル概要④

・Sleep/Wakeup
システムの消費電力低減のため、LINではSleep機構をサポートしている。



LINの概要

（４）LINプロトコル概要⑤

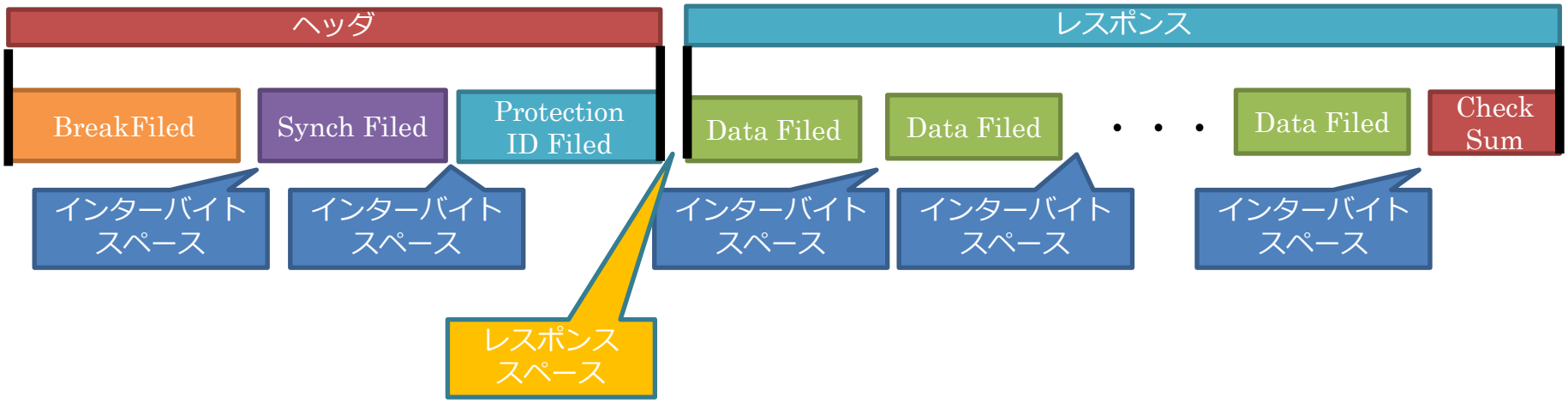
・LINのエラー

エラー名	説明
Bit Error	自身が送信したデータとバス上のデータが不一致
Checksum Error	受信したデータとチェックサムを足した結果がFFhでない
Identifier Parity Error	受信したパリティ値と受信データから算出したパリティ値が不一致
Slave Not Responding Error	ヘッダ送信後、規定以内にレスポンスが完了しない
Inconsistent Synch Filed Error	Synch Filedで受信したデータから算出した通信速度が許容以上にずれている
Physical Bus Error	LINバス上に有効なメッセージを送信できていない

LINの概要

（４）LINプロトコル概要⑥

・インターバイトスペース/レスポンススペース



・各Filedのストップビット～スタートビットの間にスペースが設けられます。
この時間のことをインターバイトスペースと呼びます。

・ヘッダとレスポンスの間にはインターバイトスペースとは異なる時間が
設けられます。この時間のことをレスポンススペースと呼びます。

ご清聴

ありがとう
ございました