AISIN
We Touch the Future

初めてのCAN

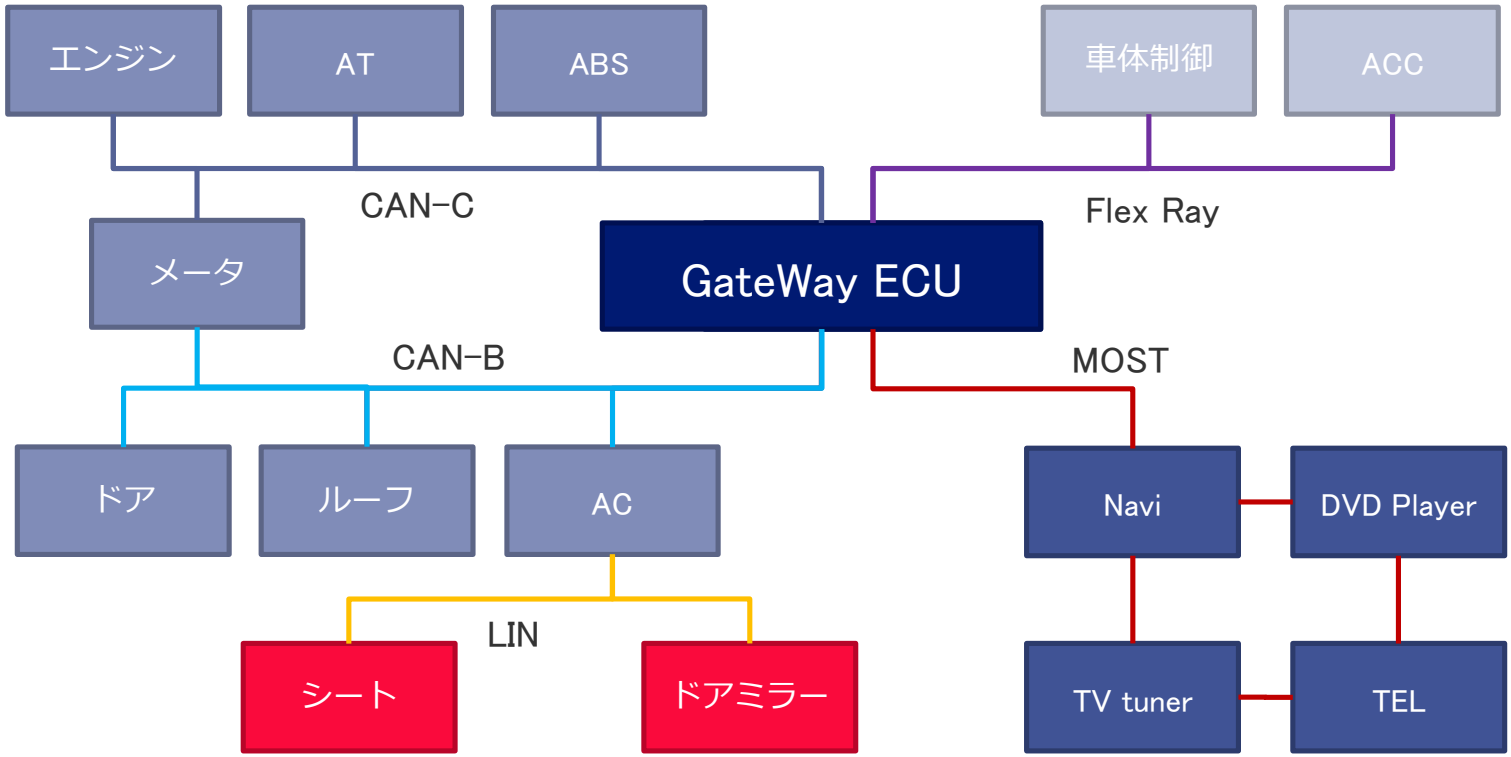
株式会社　アイシン
電子センター
ソフトウェア基盤技術部　　古河　晃

CONTENTS

1.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



（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(Classical CAN) ~5MBps(CAN FD)	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

（3）各車載ネットワークにおけるCAN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。

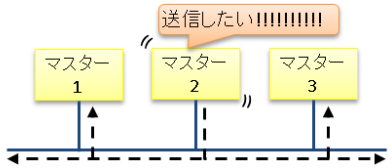
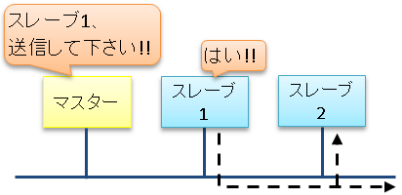
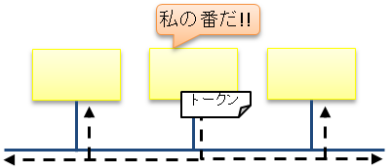
2.CANの概要

（１） 通信形態

種類 :ノード :バス	バス型	スター型	リング型
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

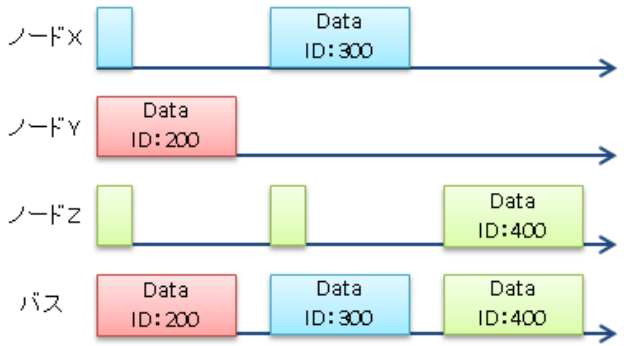
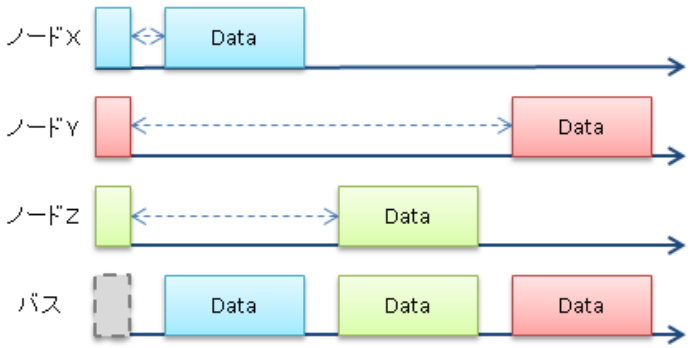
“複雑な配線の解消”という目的があったことから、CANでは「バス型」を採用している。

（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。

（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

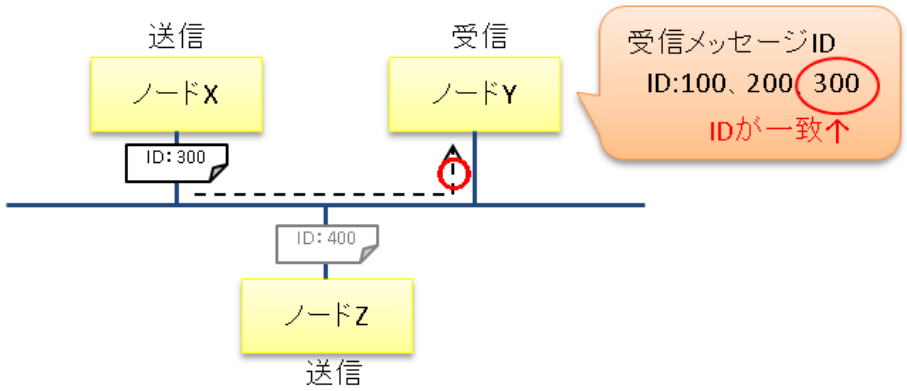
CANはCSMA/CA方式を採用している。
具体的な優先順位の判定方法はCANプロトコル詳細で解説する。

※IDとは　Identifierの略。
　データ内容や送信を識別するとともに、フレーム調停する役割もある

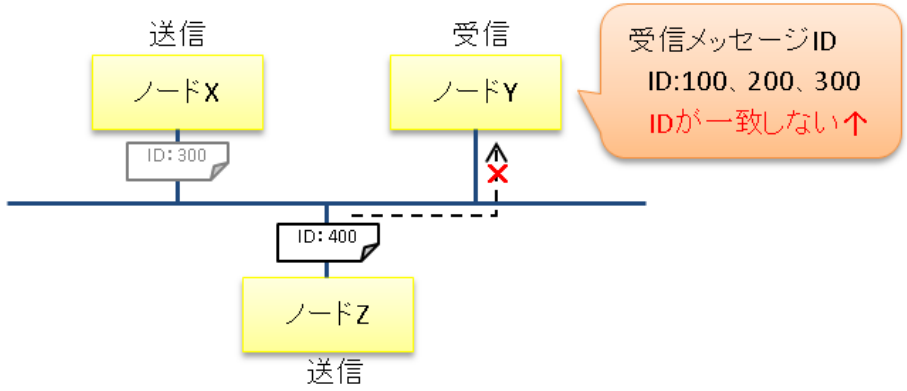
（4） Acceptance Filter

Acceptance Filter：
各メッセージスロットに対してIDをフィルタリングする機能
→　マイコン内のプログラム上で不要な受信処理を減らす

< Acceptance Filterと一致したIDの場合>

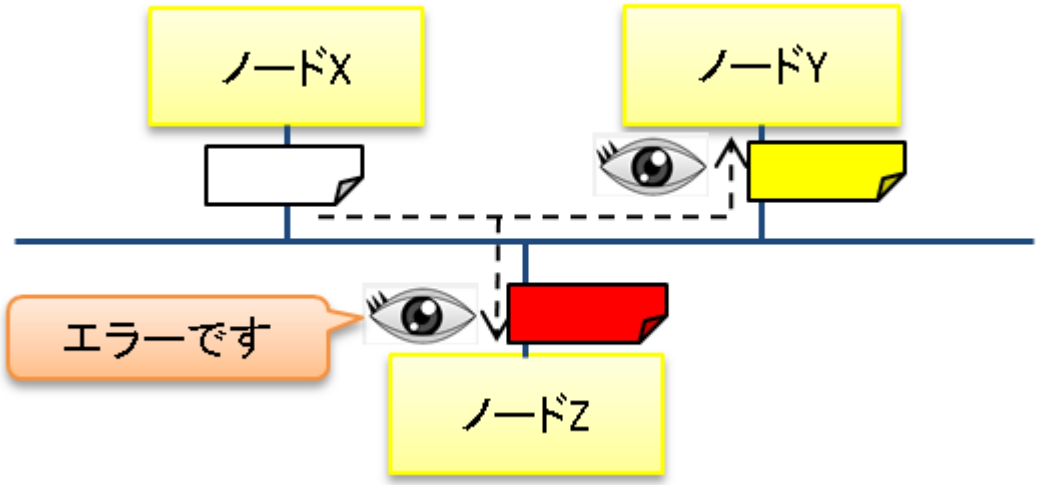


< Acceptance Filterと一致したIDでない場合>



マイコンとしては受信するがCANコントローラーで
受信データを破棄する。

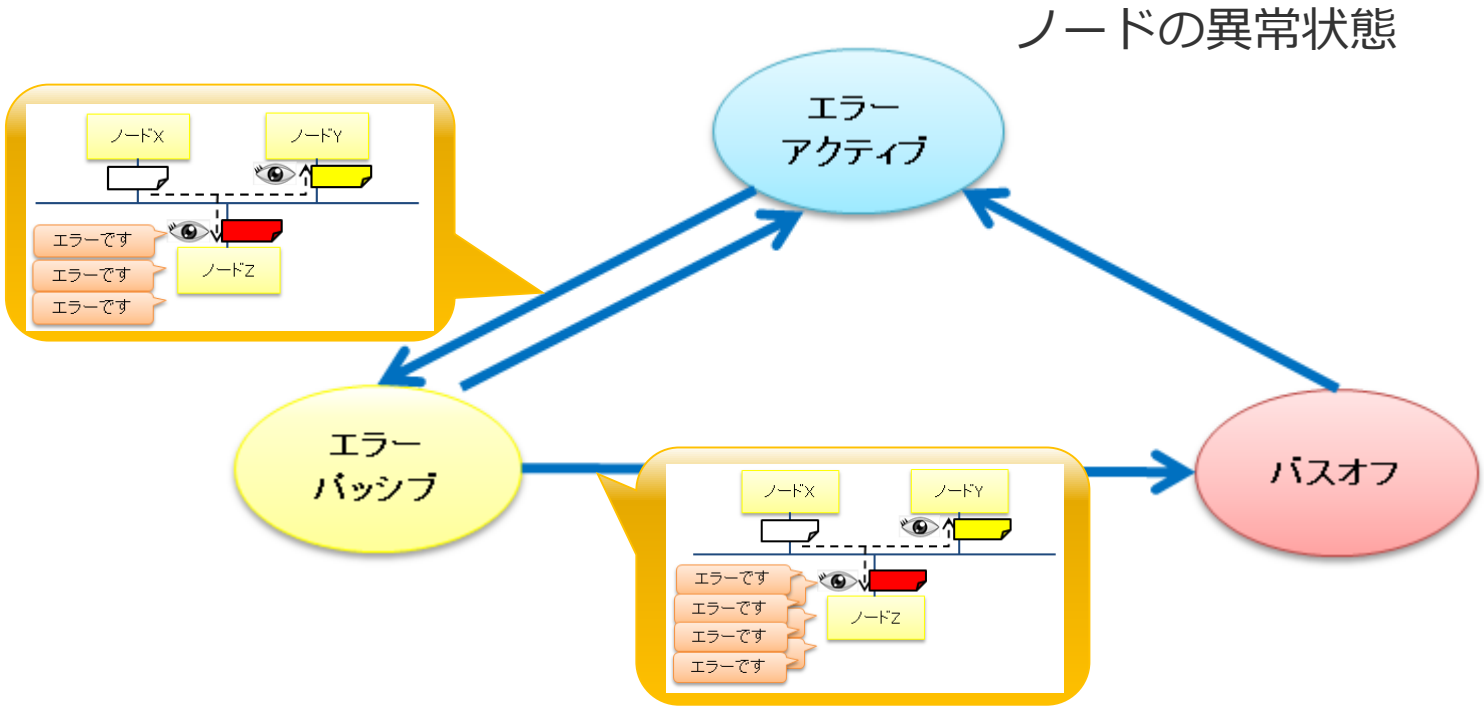
（５） エラー検出



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。

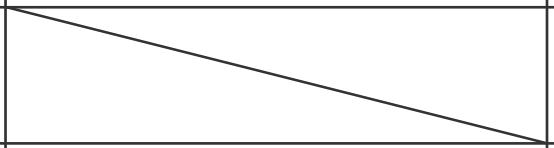
※エラーフレームの詳細はCANプロトコル詳細で解説

（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)

（7） ノイズ対策

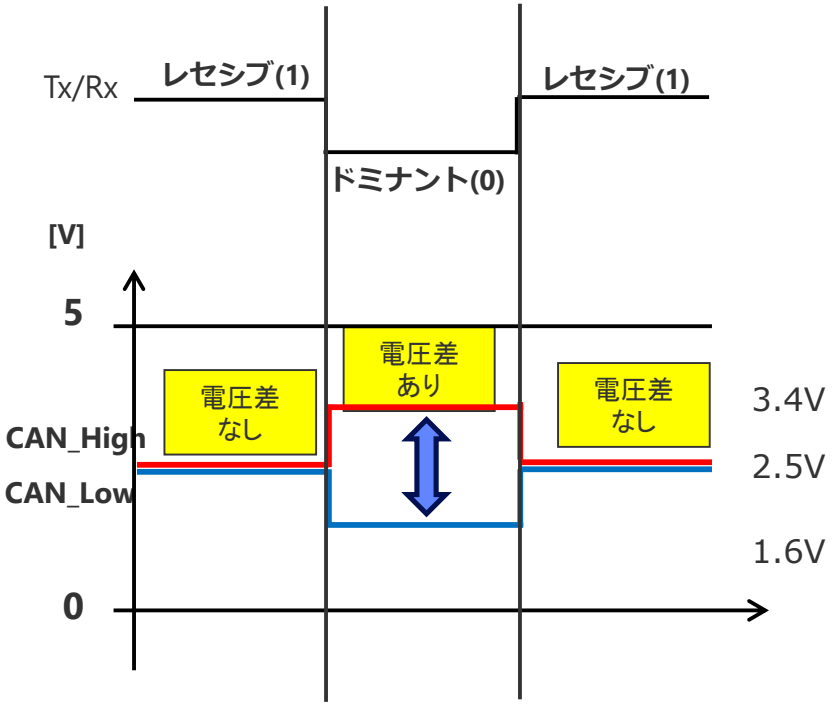
種類	単線	2本線	ツイストペアケーブル
実波形			
差分電圧			
入力信号			

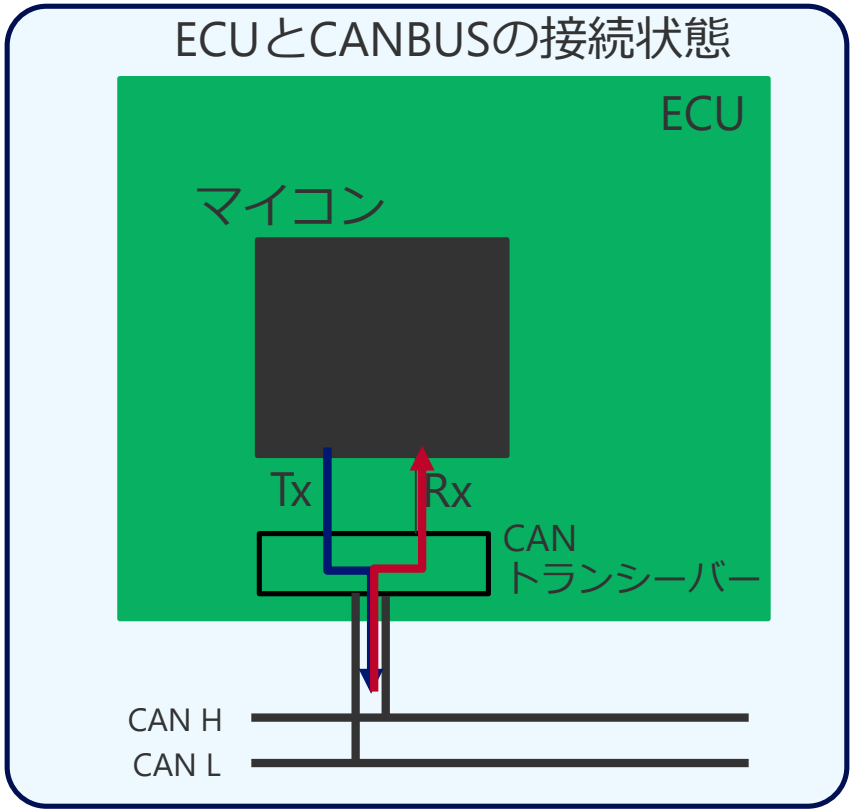
CANの信号線にはツイストペアケーブルを用いている。
各線に流れる電圧の差でデータを送信している。

Point

ノイズの影響を受けても電圧の差は変わらないため信号への影響はない

（8） ドミナントとレセシブ

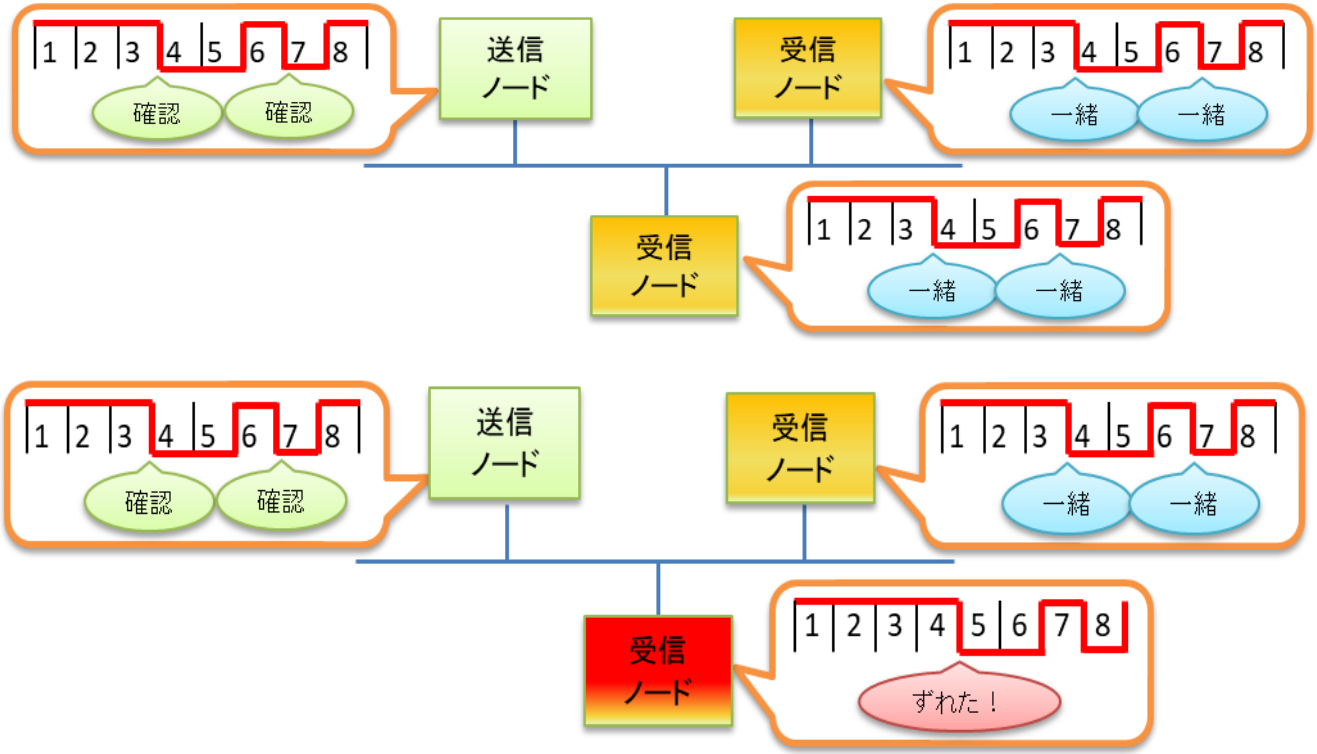




CANでは電位差でHighかLowかの判断をする。
　電位差がある方をLow(0)・ドミナント(優勢)
　電位差がない方をHigh(1)・レセシブ(劣勢)
名前の通り、バス上の優先度は名前の通りドミナントの方が高い。

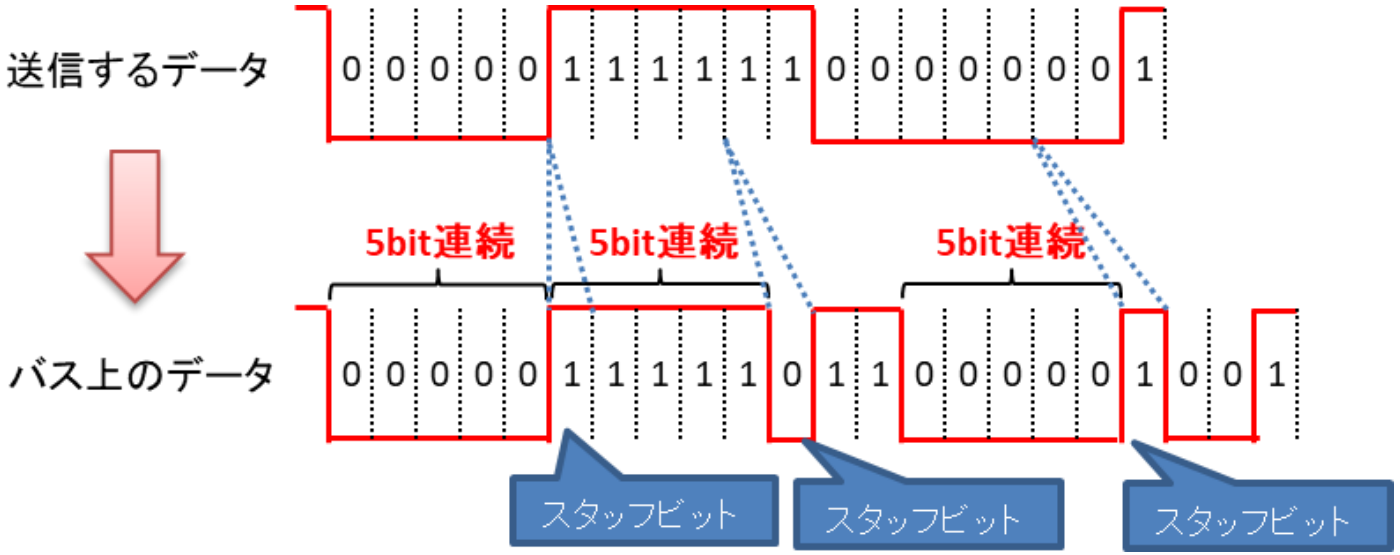
CANトランシーバーで　マイコン⇔BUSのデータ変換が行われる

（９） 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。
※NRZ方式を採用しているため、データの切り替わり時にチェックする

（１０） ビットスタッフィング



Point

・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　(フレームフォーマットについては後述)

3.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。

※リモートフレームやオーバーロードフレームは
ほとんど使われない機能なので本日説明は割愛します。

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。

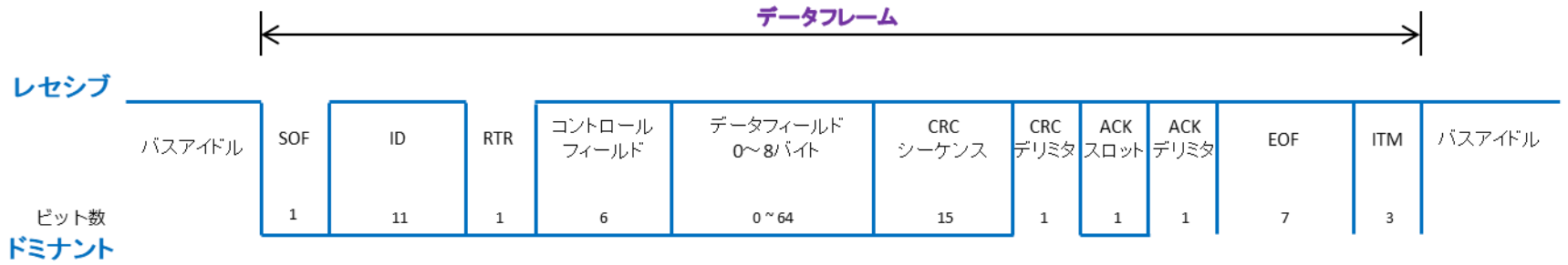
・データフレームの形式には“標準フォーマット”と
“拡張フォーマット”の細部で異なる2種類のフォーマットが存在する。

・“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。

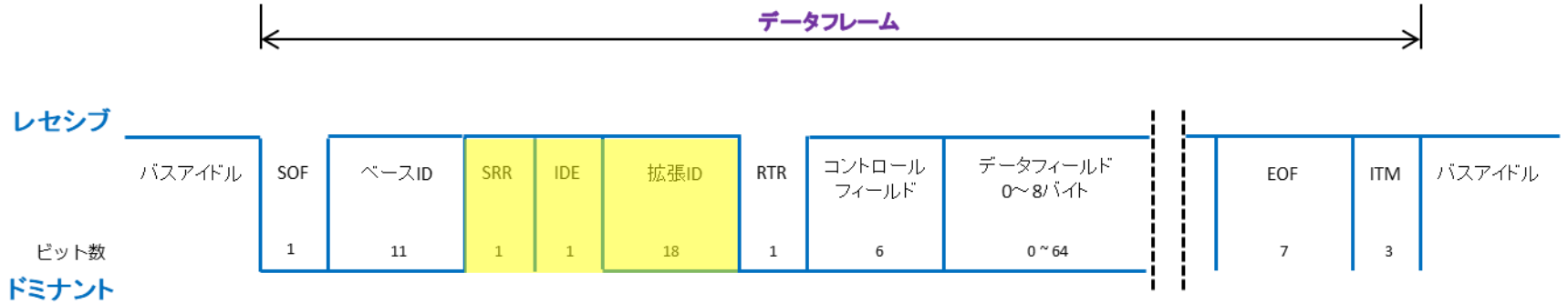
ID / プロトコル	標準フォーマット		拡張フォーマット	
	送信	受信	送信	受信
2.0A	○	○	× (送信不可)	**×** **(エラー)**
2.0B Passive	○	○	× (送信不可)	△ (無視)
2.0B Active	○	○	○	○

（2） データフレーム②

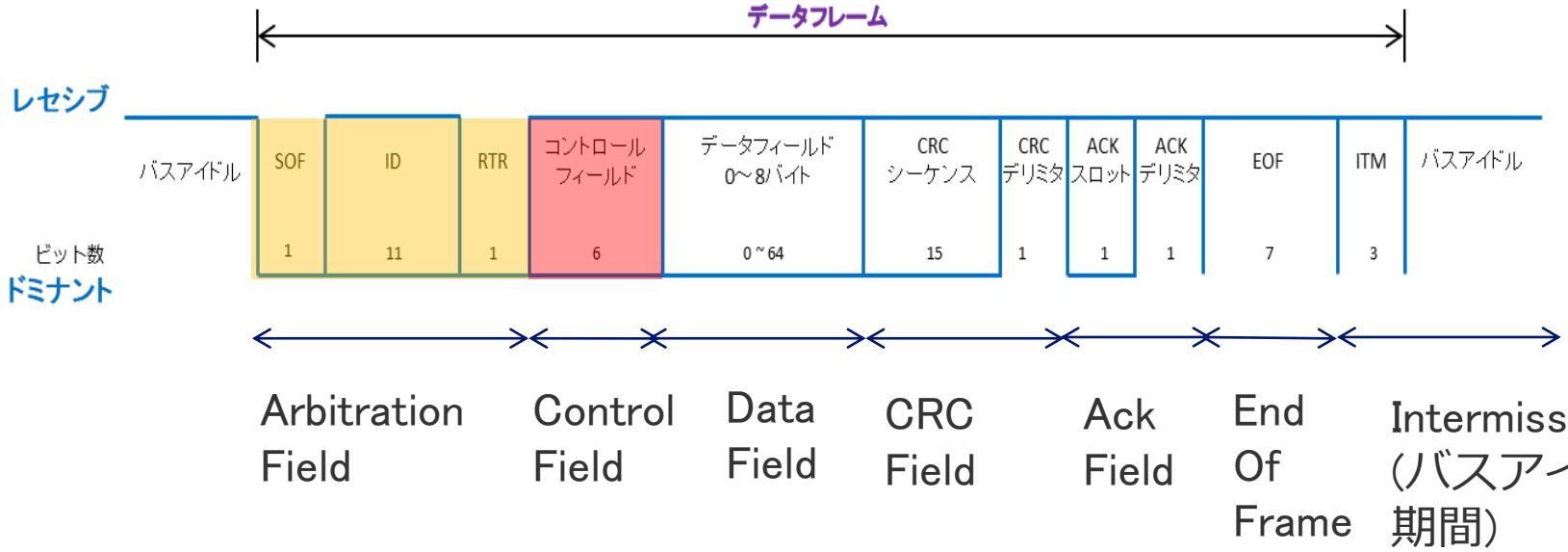
★標準フォーマット



★拡張フォーマット

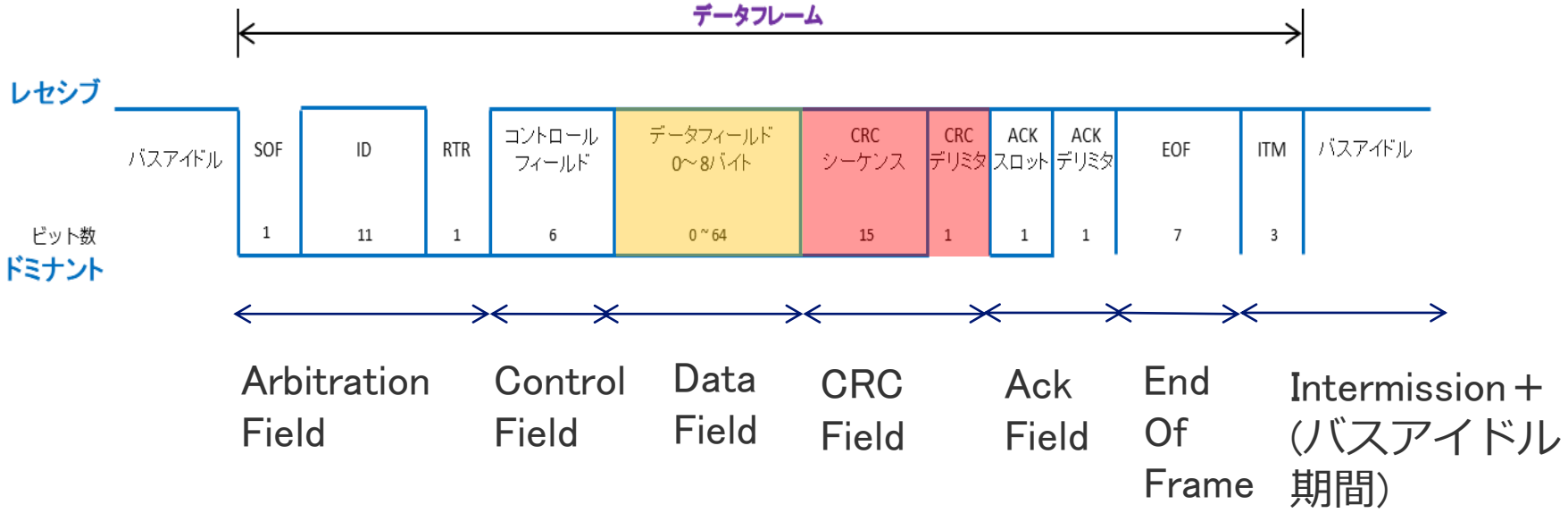


（2） データフレーム③(標準フォーマット①)



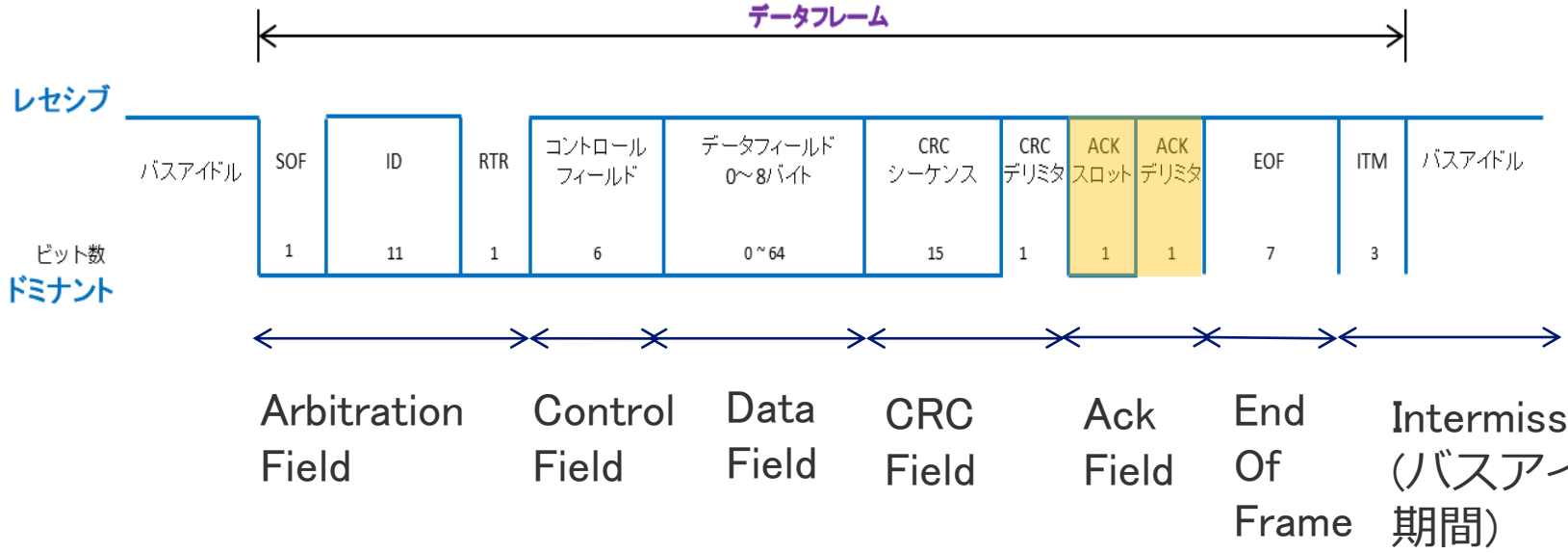
- Arbitration field
  →フレームの開始の指定と、CANのメッセージIDの指定、及びデータフレームとリモートフレームかの区別をするfield
  名前の由来は、ここでメッセージの優先度の調停(Arbitration)を行っていることから(調停の仕組みについては後述)
- Control field
  →データフィールドの長さを指定。0Byte～8Byteが設定可能

（2） データフレーム④(標準フォーマット②)



- Data Field
  →送受信させるデータを指定。MSBから送信する。
- CRC Field
  →Arbitration Field ～ Data Field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。

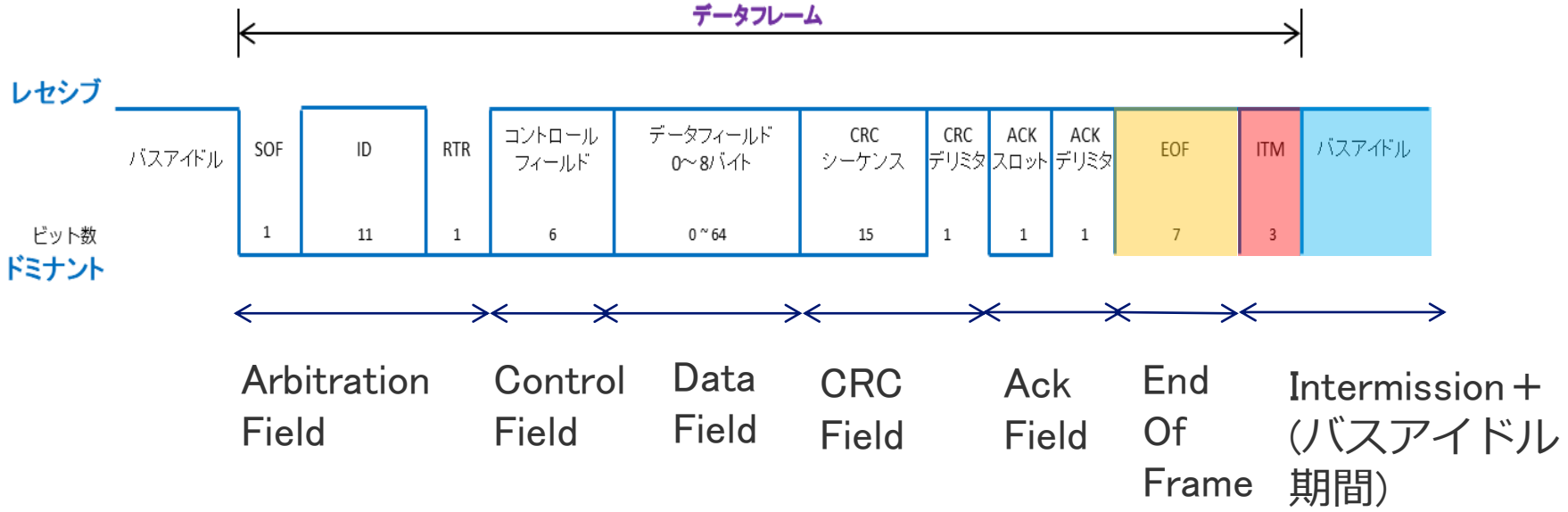
（2） データフレーム⑤(標準フォーマット③)



・Ack field
→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 field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。

Point

（2） データフレーム⑥(標準フォーマット④)



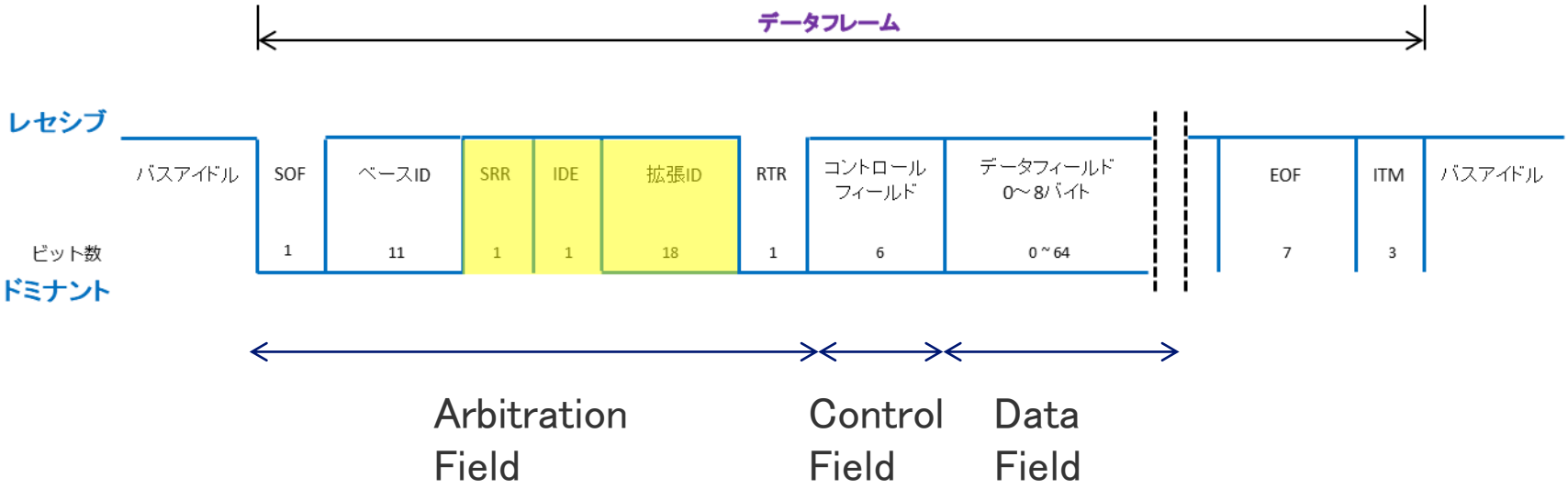
・End Of Frame
→データフレームの終わりを示す。7ビットのレセシブを固定。

・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

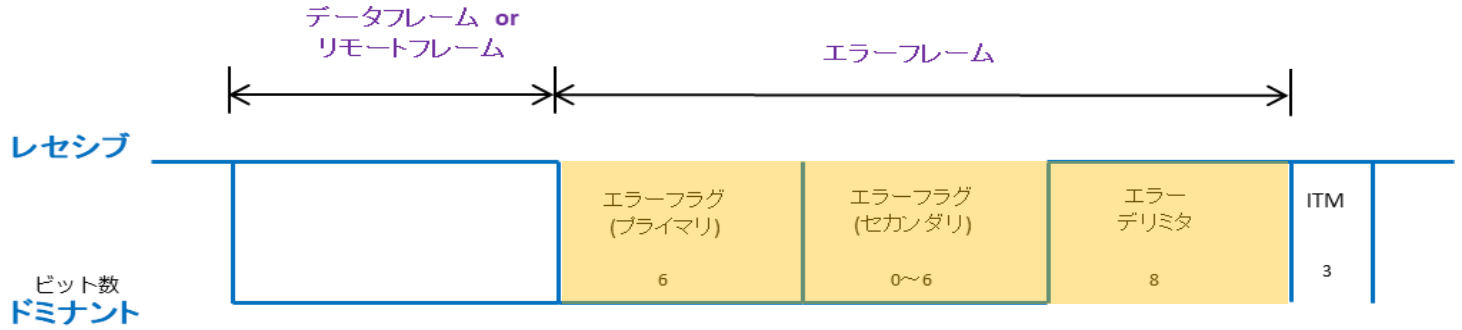
（2） データフレーム⑦(拡張フォーマット)



・標準フォーマットとの違いはArbitration field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。

（2） エラーフレーム①

・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。



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。

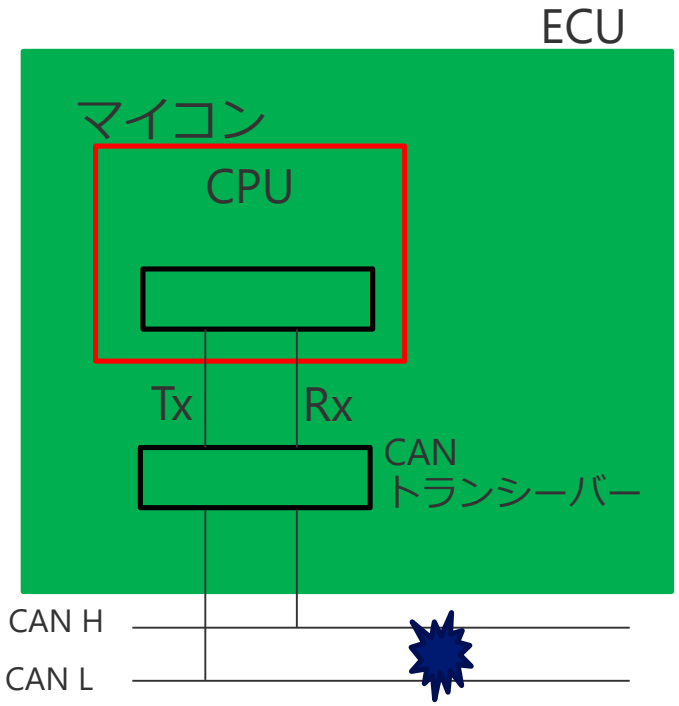
（３）CANのエラーチェック①

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

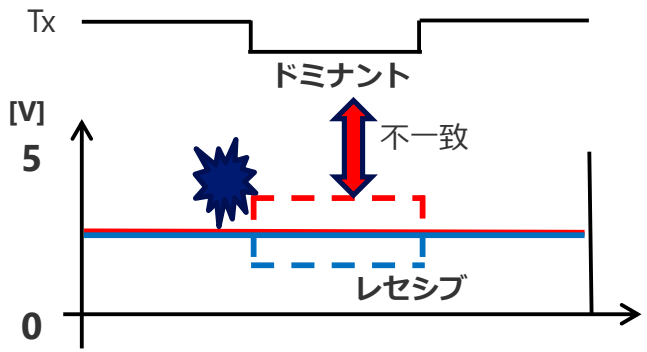
・上記チェックに引っかかった時、即時にエラー検出したノードがエラーフレームを送信する。

（３）CANのエラーチェック①-２

実際、どんな時にエラーが発生するか



CAN-H/CAN-Lがショート！

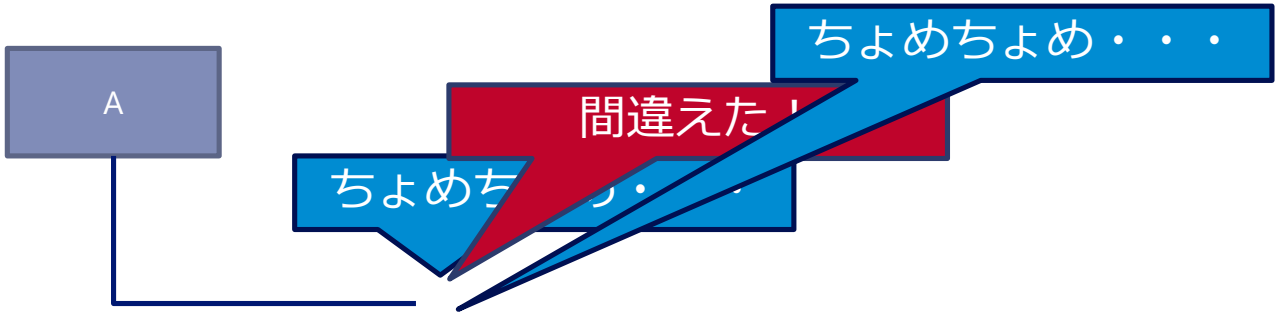


CAN-H/CAN-Lがショートすると BUS上は CAN-H/CAN-Lで同一電位となるため信号としてはレセシブとなりTxデータと不一致

→　ビットモニタリングによるエラー検出
　　SOFで基本的に引っかかる

CAN通信プロトコルの詳細

（３）CANのエラーチェック②



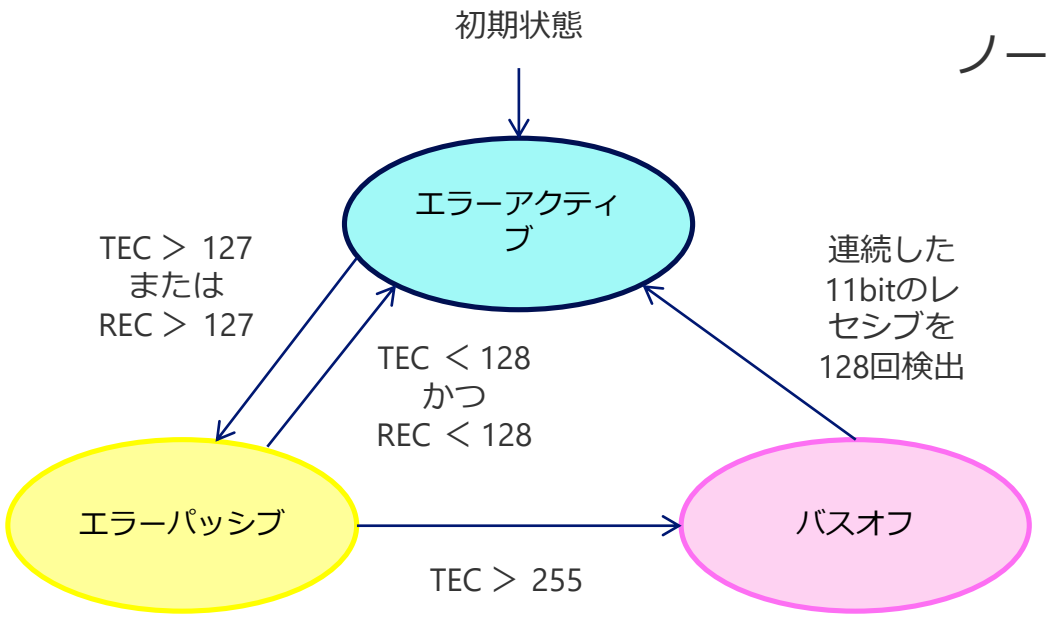
・間違えるとエラーフレームが出る。その後送信ノードはデータ再送をする。

→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・・・

送信エラーカウンタ

→自身の送信成功/失敗回数をカウントする。

（３）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**

Point

CAN通信プロトコルの詳細

（３）CANのエラーチェック④

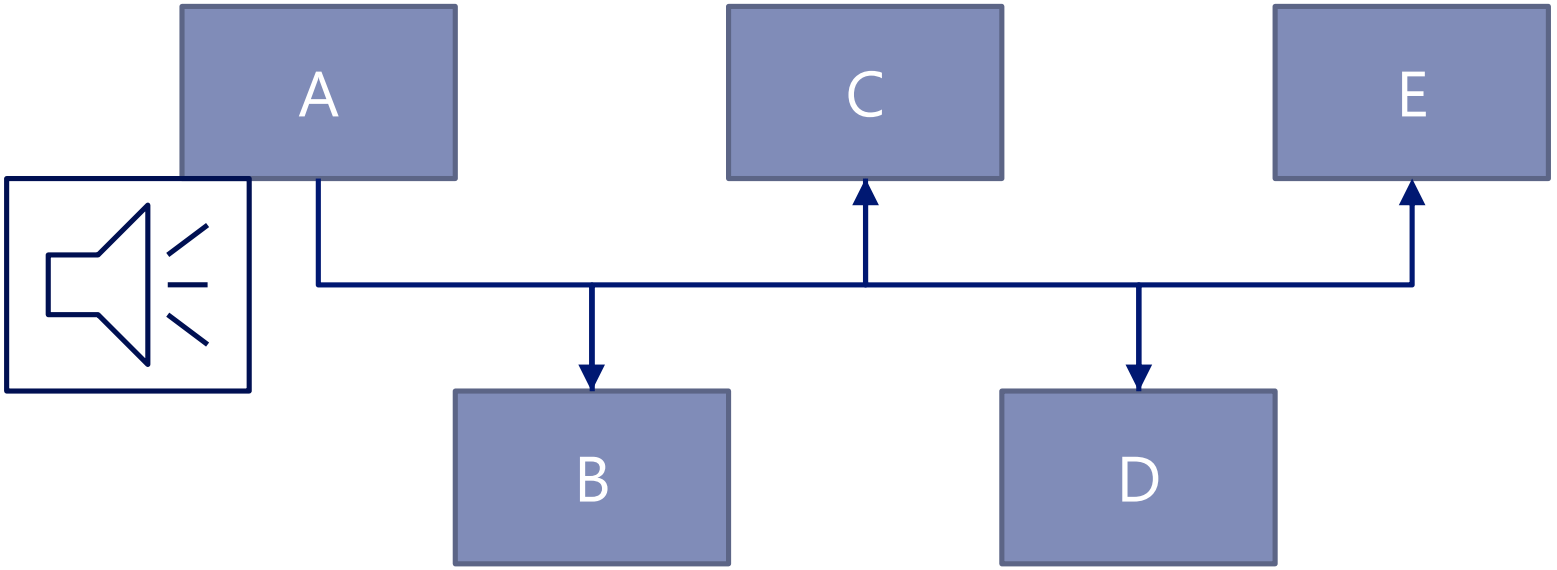
状態	振る舞い
エラーアクティブ	・エラーを検出した場合、**アクティブエラーフラグを送信**する ・アクティブエラーフラグが送信されると、**送信フレーム処理が強制的に終了させられる**
エラーパッシブ	・エラーを検出した場合、**パッシブエラーフラグを送信**する ・パッシブエラーフラグはアクティブエラーフラグのように**送信フレーム処理を強制的に終了させることはできない** ・通常はITM後に送信可能だが、エラーパッシブ状態ではITM後、**8bitのレセシブを経ないと送信できない。** **→エラーアクティブのノードより送信できる可能性が低い** ・Ackエラーを検出しても送信エラーカ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）

Point

①Ack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
→　対抗機がいない場合はBUSOFFは発生しない。

CAN通信プロトコルの詳細

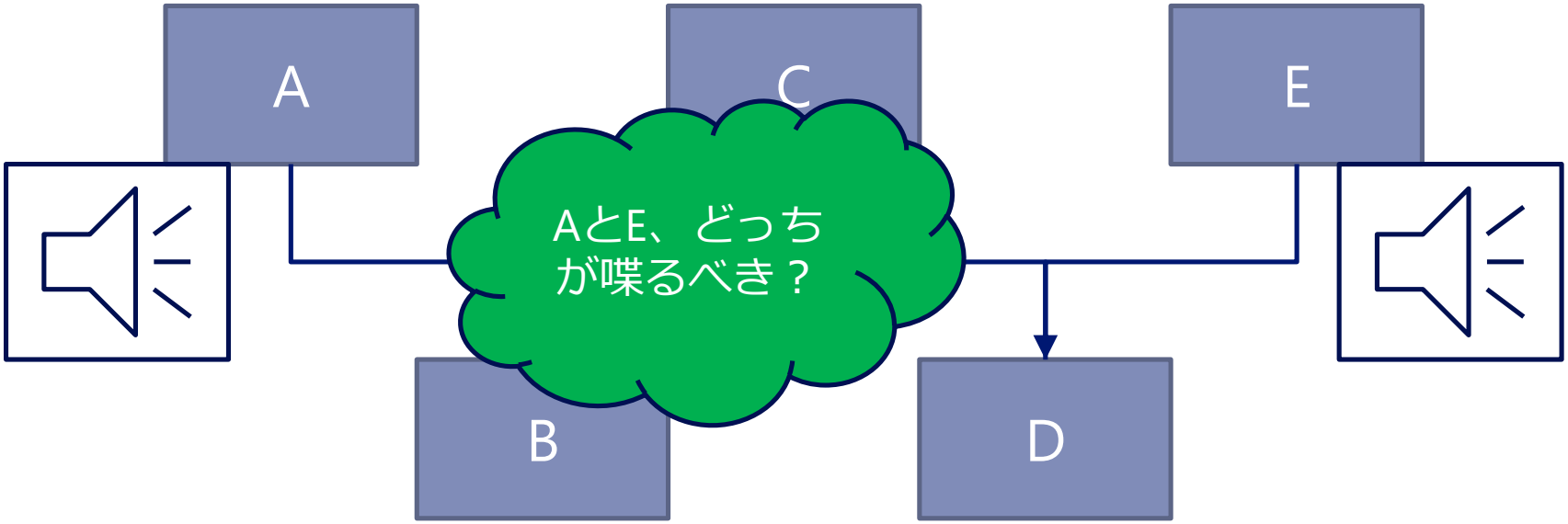
（４）通信調停①



・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。

（４）通信調停②



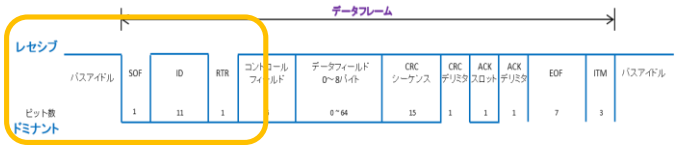
・CANはマルチマスタ方式であるため、同時に複数のノードが送信を始める可能性がある。

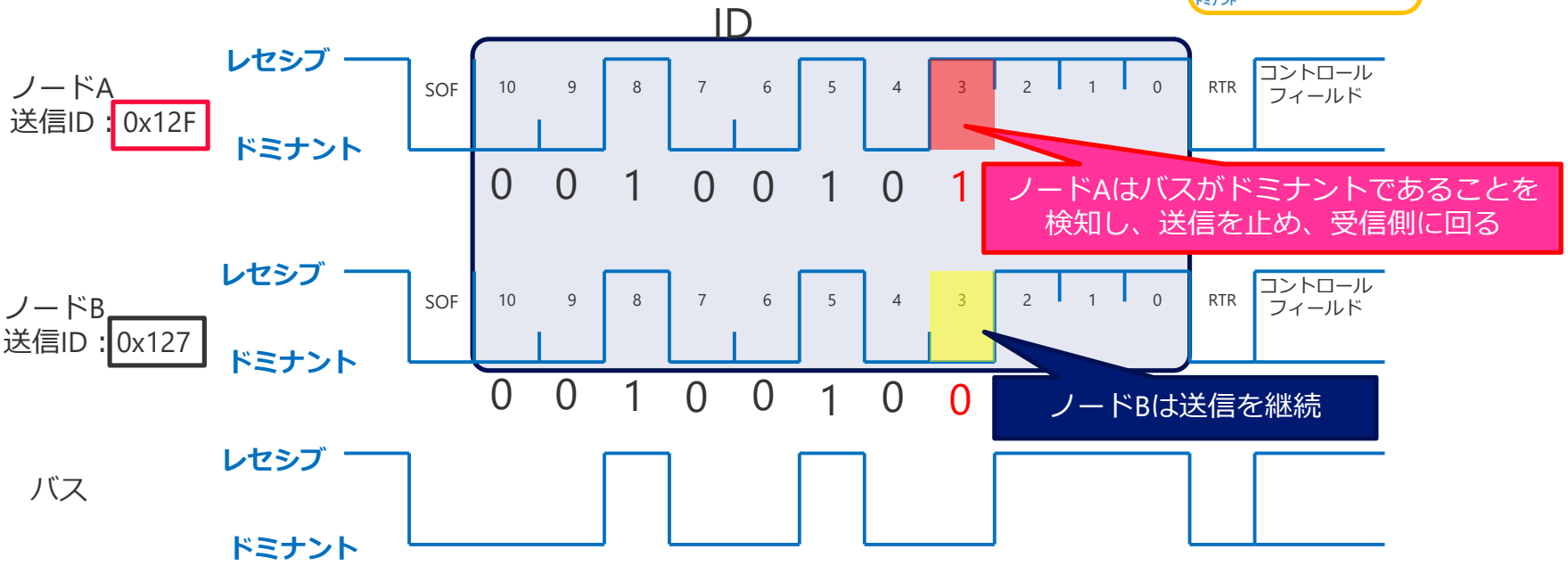
Point

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

（４）通信調停③





Point

- 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。

（４）通信調停④

重要

・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。

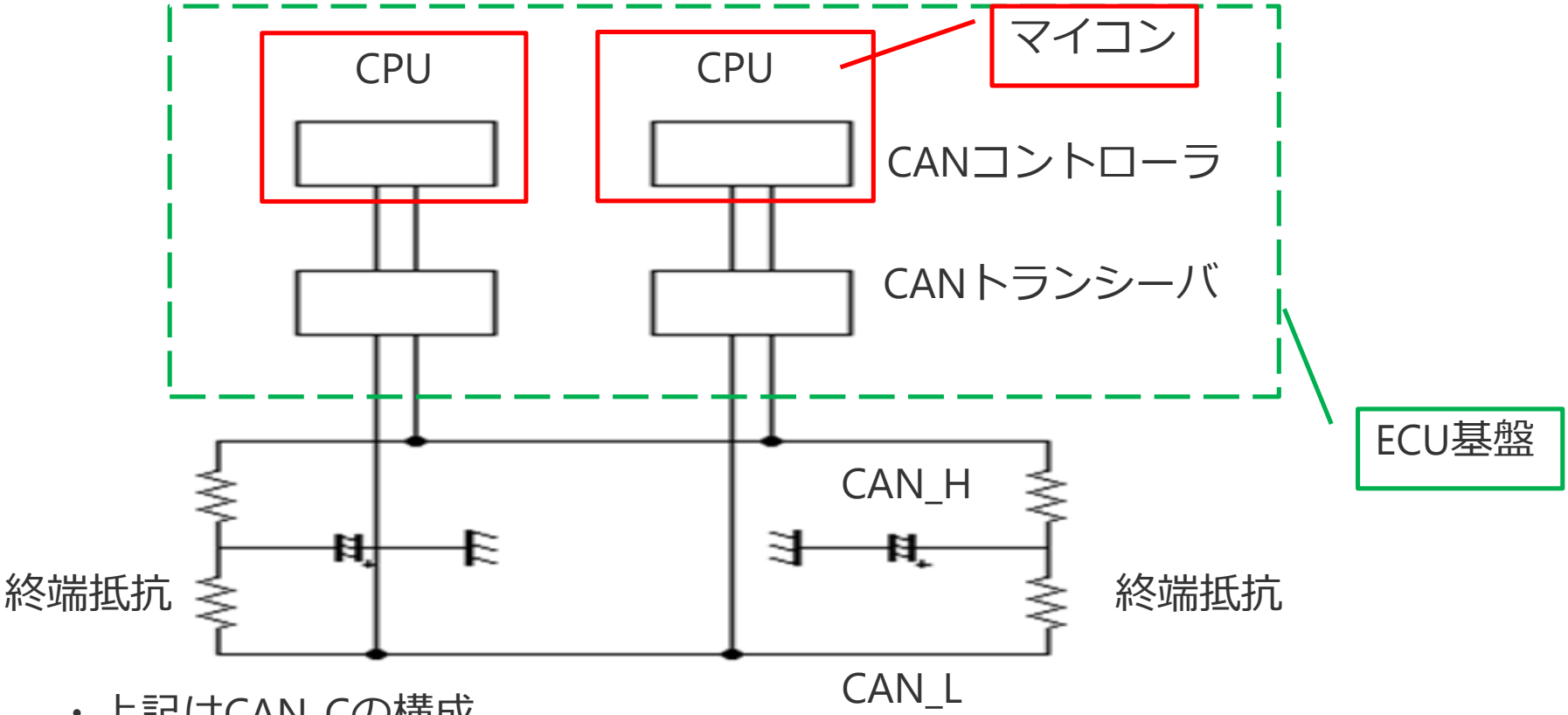
・同じIDのデータフレームとリモートフレームが同時に送信された場合、IDのみでは調停は行えないため、RTRも使用して通信調停の判定を行う。データフレームではRTRはドミナント、リモートフレームではRTRはレセシブであるため、**データフレームが優先される**ことになる。

・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)

（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

（２）CANコントローラ①

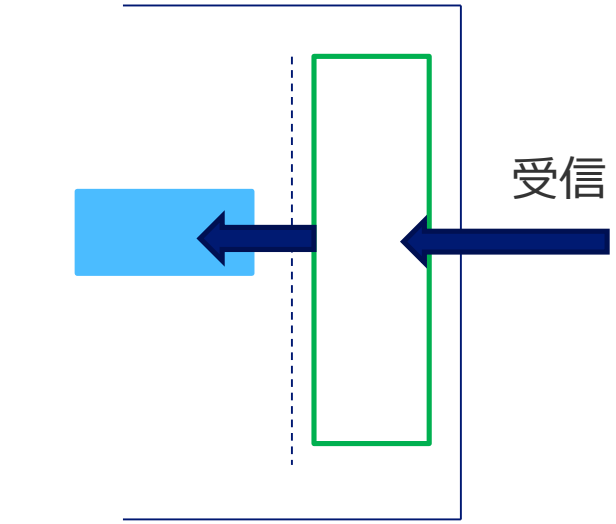
・「外付け」と「内蔵」の2つのパターンがある。
※但し、現在車載用マイコンとして販売されているほとんどのマイコンは
CANコントローラを内蔵している。

・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。

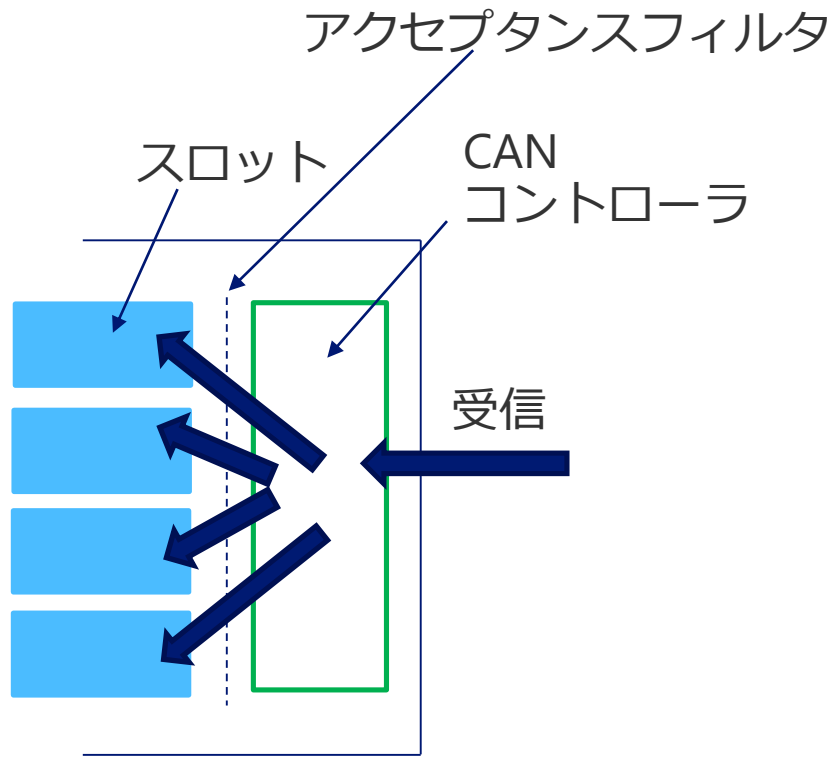
「BASIC-CAN」と「FULL-CAN」の大きな違いは、メールボックスを
一つで対応するか、メッセージごとにメールボックスを設けるかの違いがある。
今はFULL-CANで対応しているマイコンがほとんどで、一部のマイコンで
FULL-CANとBASIC-CANの併用が行われている。

・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。

（２）CANコントローラ②



アクセプタンスフィルタを通して
一つのスロットに割り当てられる
→受信周期によってはSW側の処理が
高くなる



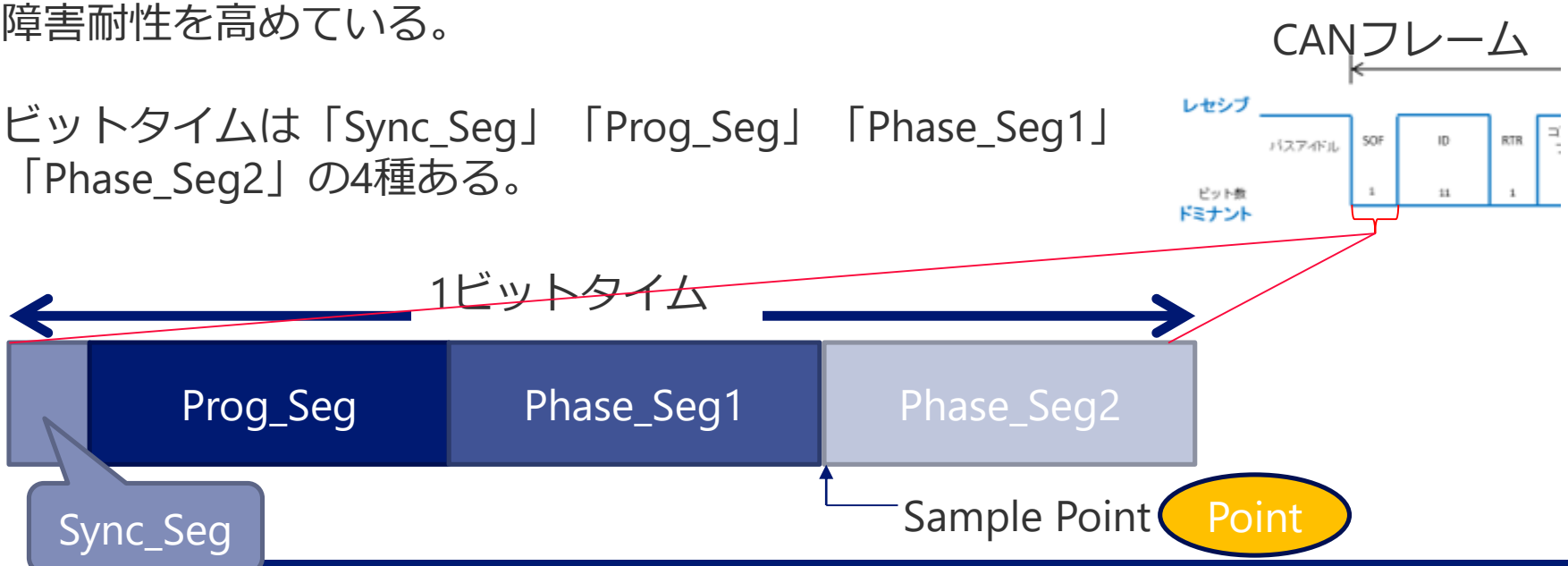
アクセプタンスフィルタを通して
フィルタを通過したスロットに設定される
→SW側の取得タイミングに余裕ができる

（２）CANコントローラ③

・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(サンプルポイント)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。



（２）CANコントローラ④

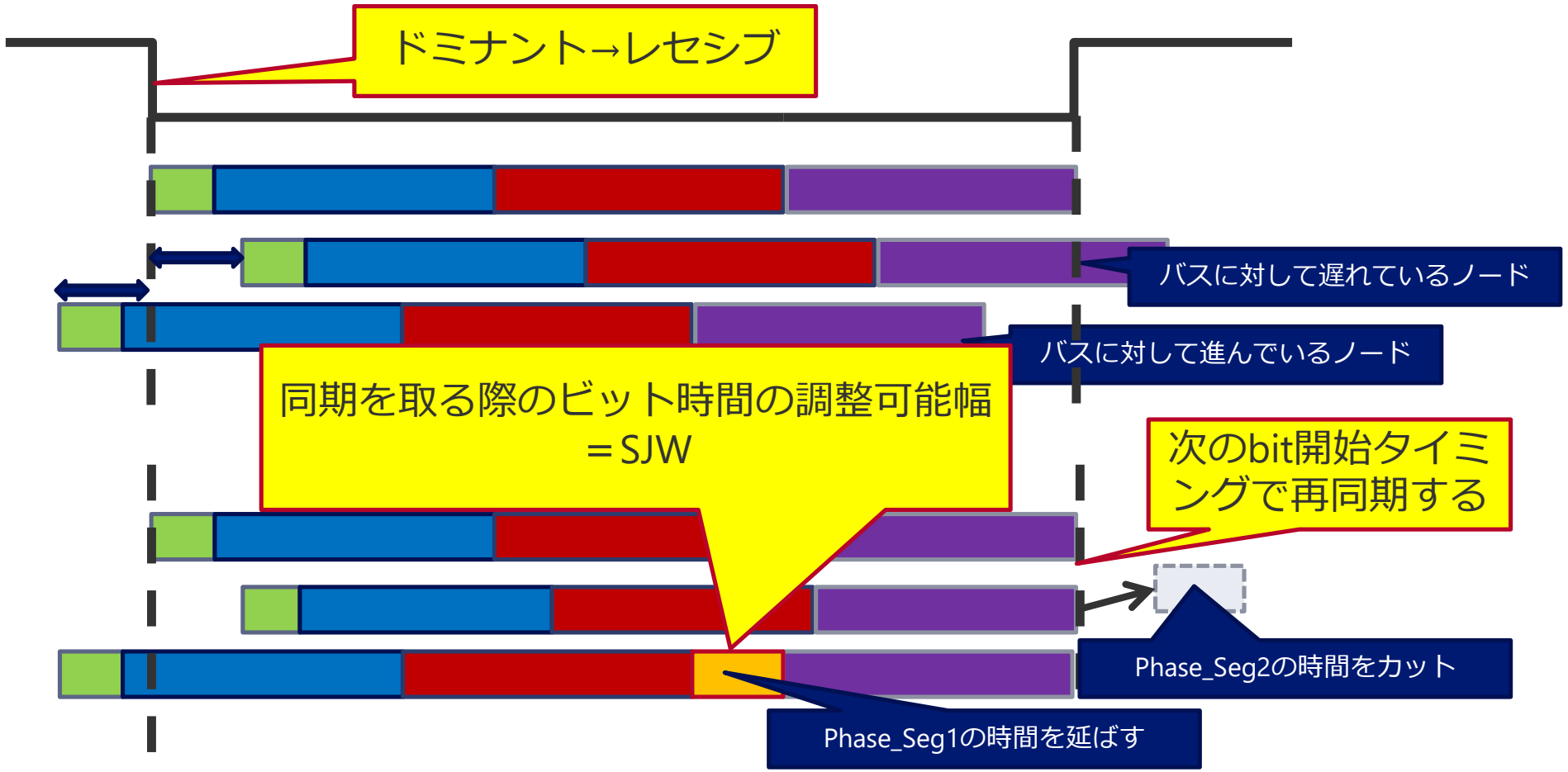
名称	役割	時間
Sync_Seg	バスの同期を取る役割	1TQ
Prog_Seg	ネットワーク遅延を吸収するための区間	1TQ ～ 8TQ
Phase_Seg1	ネットワークに対して自ノードが進んでいる場合の調整区間 ※マイコンによってはProg_Seg＋Phase_Seg1を設定するケースもある。	1TQ ～ 8TQ
Phase_Seg2	ネットワークに対して自ノードが遅れている場合の調整区間	2TQ ～ 8TQ

Point

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

（２）CANコントローラ⑤

・SJW
→バス遅延を吸収する役割



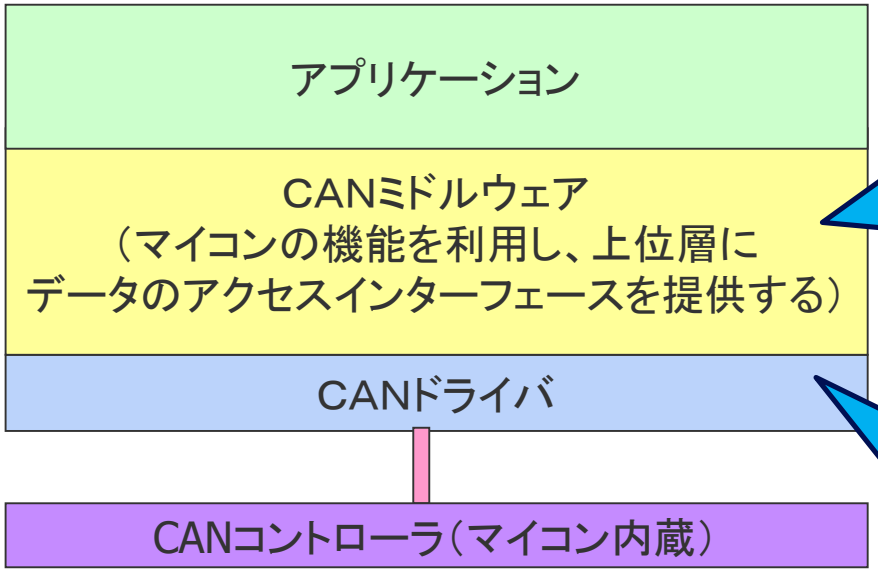
（３）CANトランシーバ

・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。

（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

（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

（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。

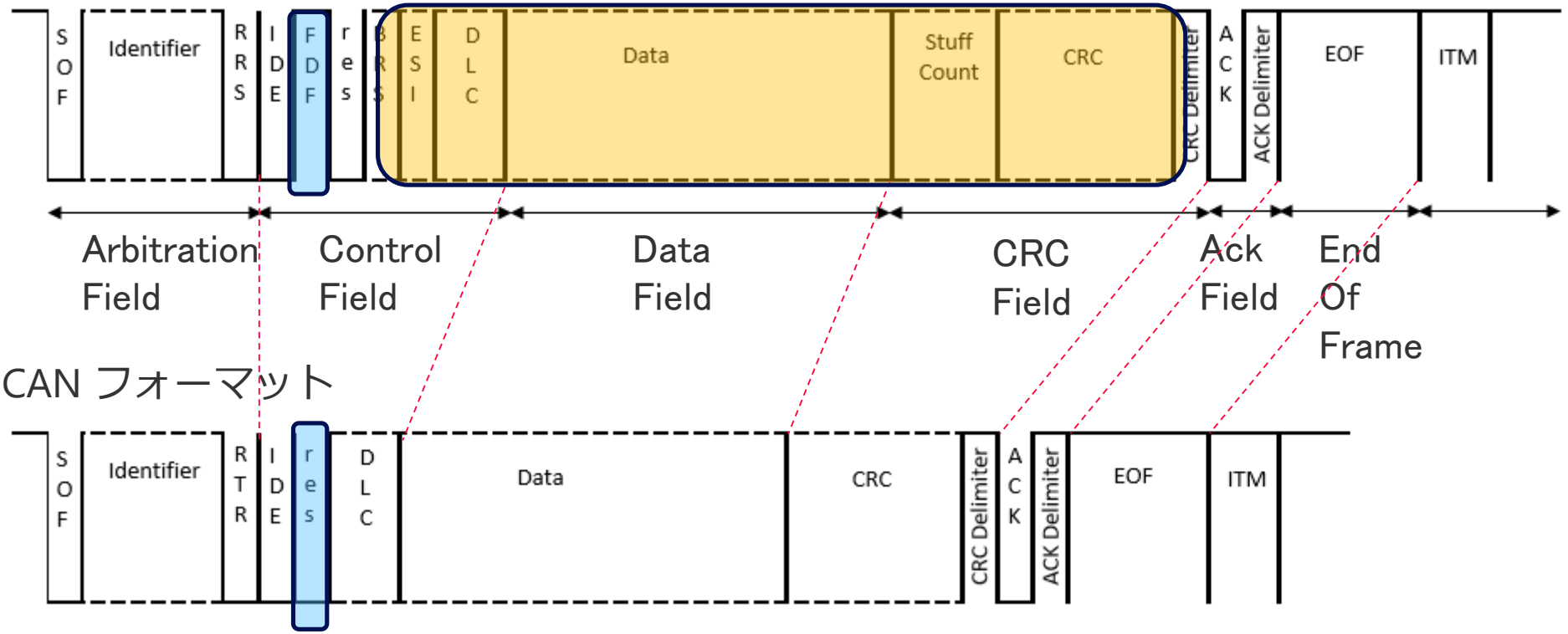
※CANとCAN-FDの混在は可能　　（４）を参照

（２）CAN FDの特徴②

プロトコル	CAN-FD
Arbitration field	RTR領域が0固定 **→リモートフレームは無し**
Control field	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 field	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 field	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 field	1bitのAck → Max 2bitのAck
EOF	EOFは7ビットのレセシブで変更なし

（3）CAN FDフォーマット

CANFD フォーマット



フォーマットの構成はCANとCANFDで変わらず

CANFDはFDFビットで判断、BRSビットで高速化（CRC　Fieldまで）
高速化する領域はデータフェーズ、それ以外をアービトレーションフェーズと呼ぶ

（４）Classical CANとCAN FDの混載について

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

- CANFDに対応しているコントローラはClassical CANとCANFDの両方を送受信することが可能
- Classical CANのみ対応しているCANコントローラにCANFDフレームを送信すると受信側がCANerrorを検出する
  したがってCANFDフレームを送信したい場合はそのネットワークにつながっているすべてのコントローラに対してCANコントローラをCANFD対応にする必要がある


AISIN
We Touch the Future