ASerial Ver1.0.0

規格仕様書

バージョン規定

“Semantic Versioning 2.0.0”(semver)に準拠

改版履歴 (Revisions)

Ver	更新日	内容
1.00	2024/11/25	・新規作成
1.01	2025/07/07	・ デバイス情報の ASerial バージョンの表現方法の変更 (ASerial バージョンを 100 倍した数値に変更) ・ 用語に単位”データ”についての説明を追加 ・ 企画書の目次の作成とインデントの整理
1.02	2025/07/13	・ デバイス情報リクエストが来た場合はたとえデバイス ID が違ってもデバイス情報を返答する仕様に変更 ・ リセットコマンド(0x00)、デバイス情報リクエストコマンド(0x01)はデバイス ID とターゲットデバイス ID の照合が取れなくても実行することを明記
1.0.0	2025/07/18	・ ブロックフロー図の誤表記の修正 ・ 返答時間を 200ms から 50ms に変更 ・ 自動探査接続のフローチャートを追加 ・ バージョン表記を”Semantic Versioning 2.0.0”に準拠へ変更 ・ バージョン表記変更に伴い ASerial のバージョンと規格書のバージョンを統合し、1.0.0 に変更 ・ ASerial のバージョン表記方法変更により、デバイス情報リクエストのデバイス情報の ASerial のバージョン領域を拡張領域に変更

目次

バージョン規定 1

改版履歴 (Revisions) 1

目次 2

1 仕様書について 4

2 用語 5

- ASerial 5
- コントローラ(Controller) 5
- デバイス(Device) 5
- データ(単位) 5
- デバイス ID 6
- デバイス Ver 6
- ターゲットデバイス ID 6

3 通信仕様 7

3-1 通信プロトコル 7

3-2 通信設定 7

4 通信規定 8

4-1 データの定義とデータの送信方法 8

4-2 パケット毎のデータ数の上限の定義 8

4-3 スタートフラグ 8

4-4 デバイス ID とデバイス Ver の扱いについて 8

4-5 デバイスによるターゲットデバイス ID の確認 9

4-6 コントローラによるデバイス ID とデバイス Ver の確認 9

4-7 デバイス情報リクエスト 9

4-8 デバイス情報の拡張データ 9

4-9 予約コマンドの定義 10

4-10 規格拡張用コマンド領域 10

4-11 ユーザー定義コマンド領域 10

4-12 アドバンスドコマンド領域.....10
4-13 通信の原則.....10
4-14 チェックデータ.....10
4-15 チェックデータの加算方法.....10
4-16 デバイスの応答速度.....10
5 データパケット.....11
5-1 コントローラ(Controller) -> デバイス(Device).....11
5-2 デバイス(Device) -> コントローラ(Controller).....12
6 フロー図解.....13
6-1 デバイス情報照合フロー.....13
6-2 通信フロー(情報照合後).....14
6-3 デバイスパケット受信フロー例.....15
6-4 コントローラパケット受信フロー例.....16
6-5 デバイス自動探査フロー例.....17
7 配線規定.....19

1仕様書について

この仕様書は NextAmusement(以下 NextAmu)がアミューズメント機器開発時に機器内シリアル通信の容易性、確実性と汎用性を持たせるために規格した”ASerial”の規格仕様をまとめたものである。

この規格は AM 機器などクローズド環境での運用を想定しています。

2用語

・ ASerial

NextAmu がアミューズメント機器開発時に機器内シリアル通信の容易性、確実性と汎用性を持たせるために規格したシリアル通信規格。

・ コントローラ(Controller)

通信する機器の関係性において、指示を出す機器のこと。
つまりはコマンドを送信する機器のことを指す。

・ デバイス(Device)

通信する機器の関係性において、指示を受けて動作をする機器のこと。
つまりはコマンドを受信する機器のことを指す。

(例)
ゲームプログラムが動作している PC とモータなどを制御するマイコンボード(以下マイコン)があった場合、PC のゲーム内容に連動してモータを動かす場合は一般的には PC からマイコンボードにコマンドを送信してマイコンがモータを制御します。
この関係の時、PC が“コントローラ(Controller)”、マイコンが“デバイス(Device)”になります。

・ データ(単位)

データは ASerial の処理のサイズ単位のことです。1 データは 1Byte に相当します。
ASerial は基本的に 1 データずつ処理を行います。
(データの送受信はパケット単位)

次ページへ

- **デバイス ID**

  デバイス(Device)に割り当てる固有の ID。

  全く同じデバイスの場合は同じ ID を使えます。

  ID は 1~255 (0 はデフォルト値なので禁止)の範囲の数値で設定します。

- **デバイス Ver**

  デバイス(Device)のバージョンを管理するための数値。

  デバイスバージョンはバージョンの違いによって互換性を持つ必要はありません

  1~255(0 はデフォルト値なので禁止)の範囲で設定します。

- **ターゲットデバイス ID**

  コントローラが通信をしたいデバイスのデバイス ID です。

  コントローラはターゲットデバイス ID を元にデバイスを探査します。

  ターゲットデバイス ID とデバイスが持つデバイス ID が照合することで通信をすることができます。

3通信仕様

3 - 1 通信プロトコル

UART

3 - 2 通信設定

ボーレート	115200 bps *1
データビット	8bit
パリティ	無し
ストップビット	1bit
フロー制御	無し

*1 基本的には変更はなしですが特別な事情があり、変更しなくてはいけなくなったときはどこか(ソースコードのコメントや別ドキュメント)に明記したうえで変更が可能です。

4通信規定

4 - 1 データの定義とデータの送信方法

1 データのサイズは符号なし整数型 8bit(1Byte)です。

数値は、0x00 ～ 0xFF です。

例えば 16bit 分のデータを送りたいときは上位バイトと下位バイトに分けて送る必要があります。

つまりは 2 データ分を使って送るということです。

4 - 2 パケット毎のデータ数の上限の定義

データ数の上限は 32 データ分(256bit)です。

4 - 3 スタートフラグ

パケットの先頭にはスタートフラグ値 0xD0 を付けてください。

もし値として 0xD0(208_{DEC})を利用するときは加算フラグ値 0xAD(173_{DEC})を利用してください。

直前に加算フラグがあった場合は次の値は＋1 します。

加算フラグ値はチェックデータにもデータ数にも含みません。加算以外では無視します。

例:

データとして 0xD0　　<0xAD><0xCF> = 0xCF + 1 = 0xD0

データとして 0xAD　　<0xAD><0xAC> = 0xAC + 1 = 0xAD

＊各フラグ値が連続することは禁止です。　　<0xD0><0xD0>　　<0xAD><0xAD>

4 - 4 デバイス ID とデバイス Ver の扱いについて

通信を行う際、デバイス側はデバイス ID の照合とチェックデータが正しく受信できていることを確認してからデータの処理を行ってください。

チェックデータの確認はコントローラ側でも行ってください。

(デバイス ID が違う場合は問題ですが、デバイス Ver に関しては互換性が保てる範囲であれば相違があっても問題はありません。)

例: PC 側での開発時の想定デバイス Ver が“3”だとして実際通信した際にデバイス Ver が“4”だったとします。しかしデバイス Ver が“4”だったとしても“3”と同様のコマンドが使えて、結果も同じであれば Ver“4”は Ver“3”に対して互換性があると言えます。

“5”以上は互換性があることを記載してないので互換性は無いものとなります

4 - 5 デバイスによるターゲットデバイス ID の確認

デバイスは受信したパケットのターゲットデバイス ID が自身のデバイス ID と違う時、そのパケットは無視します。(リセットコマンド、デバイス情報リクエスコマンドのときは例外)

4 - 6 コントローラによるデバイス ID とデバイス Ver の確認

コントローラはデバイス ID とデバイス Ver の照合が取れた場合のみ接続を確立してください。

つまりは、コントローラ側のターゲットデバイス ID とデバイスのデバイス ID が同一かつ、デバイス Ver が対応バージョン内であれば通信を行うことができます。

4 - 7 デバイス情報リクエスト

コントローラはデバイスに対してデバイス情報リクエストを送信してデバイス情報を取得してください。

デバイス情報は[デバイス ID、デバイス Ver、拡張データ]の 3 つの情報です。

4 - 8 デバイス情報の拡張データ

ユーザーが自由に使えるデータ領域です。

例えば、両手ずつのデバイスで左手用と右手用を判別するデータを入れたりすることができます。

データサイズは 2 データ固定。ただし、必ずしも 2 データ分として扱う必要はありません。

1 データずつとして扱うこともできます。

必ずデバイス情報のサイズが計 4 データ分になれば問題ありません。

デバイス ID 0x01~0xFF	デバイス Ver 0x01~0xFF	拡張領域 0x0000~0xFFFF
1 データ分	1 データ分	2 データ分
計 4 データ分		

4-9 予約コマンドの定義

コマンドの“0”と“1”は予約済みコマンドなので使うことは禁止です。

(ライブラリを作る際は必ずこの 2 つは実装してください)

*このコマンドは ID の照合を無視し、実行されます。

0x00 : デバイスの初期化(リセット)

0x01 : デバイス情報リクエスト

4-10 規格拡張用コマンド領域

0x02～0x1F は拡張用領域なので利用禁止です。

4-11 ユーザー定義コマンド領域

ユーザー定義コマンドは 0x20～0x7F の数値が使えます。

4-12 アドバンスドコマンド領域

0x80～0xFF はアドバンスド領域です。高度な制御に使うので一般に利用禁止です。

予約済み (0x00~0x01)	拡張領域 (0x02~0x1F)	ユーザー定義領域 (0x20~0x7E)	アドバンスド領域 (0x80~0xFF)

4-13 通信の原則

デバイスから自発的な通信はないものとします。必ずコントローラからのコマンドに応答する形での通信とします。(応答型通信)

4-14 チェックデータ

通信文の終端には必ずチェックデータ(読み取りデータの加算値)をつけてください。

上位バイト、下位バイトの順番で送信してください。

例: 0xAF5E ならば<AF><5E>の順番です。

4-15 チェックデータの加算方法

もし、値で 2 データ(2Byte)以上のデータを上位下位に分けて送信した場合でも加算は 1 データずつ加算してください。

4-16 デバイスの応答速度

デバイスからの返答はコントローラ送信完了から 50ms 以内に返してください。

5データパケット

5 - 1 コントローラ(Controller) -> デバイス(Device)

5 - 1 - 1 フォーマット最大サイズ

8bit(スタートフラグ)+8bit(ターゲットデバイス ID)+8bit(送信データ数値)+8bit(コマンド)+(8bit(データ)+8bit(加算フラグ値))*32(最大データ数)+16bit(チェックデータ)

= 560bit(70Byte)

5 - 1 - 2 データパケットフォーマット

スタートフラグ 0xD0 (8bit)	ターゲットデバイス ID (8bit)	送信データ数 (8bit)	コマンド (8bit)	データ (0～256bit (0～32Byte))	チェックデータ (16bit)

スタートフラグ、デバイス ID、送信データ数、コマンド、データ 1、データ 2・・・、チェックデータ(読み取りデータの加算値)

(例)*読みやすいように 1 バイトごとに<>で区切っています(0 埋め)

<D0><0E><0A><1F><12><A7><FF><00><00><BF><AE><FD><6D><00><04><8F>

先頭から

D0 : スタートフラグ

0E : ターゲットデバイス ID(14_{DEC})

0A : 送信データ数(10_{DEC})

1F : コマンド(31_{DEC})

12 ~ 00 : データ

048F : チェックデータ(1176_{DEC})

＊データを含まないことも可能です(固定の処理を行う場合やデバイス情報リクエストなどの時です)

スタートフラグ 0xD0 (8bit)	ターゲットデバイス ID (8bit)	送信データ数 (8bit)	コマンド (8bit)	チェックデータ (16bit)

この時、送信データ数は 0、チェックデータも 0 になります

5 - 2 デバイス(Device) -> コントローラ(Controller)

5 - 2 - 1 フォーマット最大サイズ

8bit(スタートフラグ)+8bit(送信データ数値)+(8bit(データ)+8bit(加算フラグ値))*32(最大データ数)+16bit(チェックデータ)

= 544bit(68Byte)

5 - 2 - 2 データパケットフォーマット

スタートフラグ 0xD0 (8bit)	送信データ数 (8bit)	データ (0～256bit (0～32Byte))	チェックデータ (16bit)

スタートフラグ、送信データ数、データ 1 、データ 2・・・、チェックデータ(読み取りデータの加算値)

(例) *読みやすいように 1 バイトごとに<>で区切っています(0 埋め)

<D0><0A><12><A7><FF><00><00><BF><0A><E0><B3><00><04><14>

先頭から

D0 : スタートフラグ値

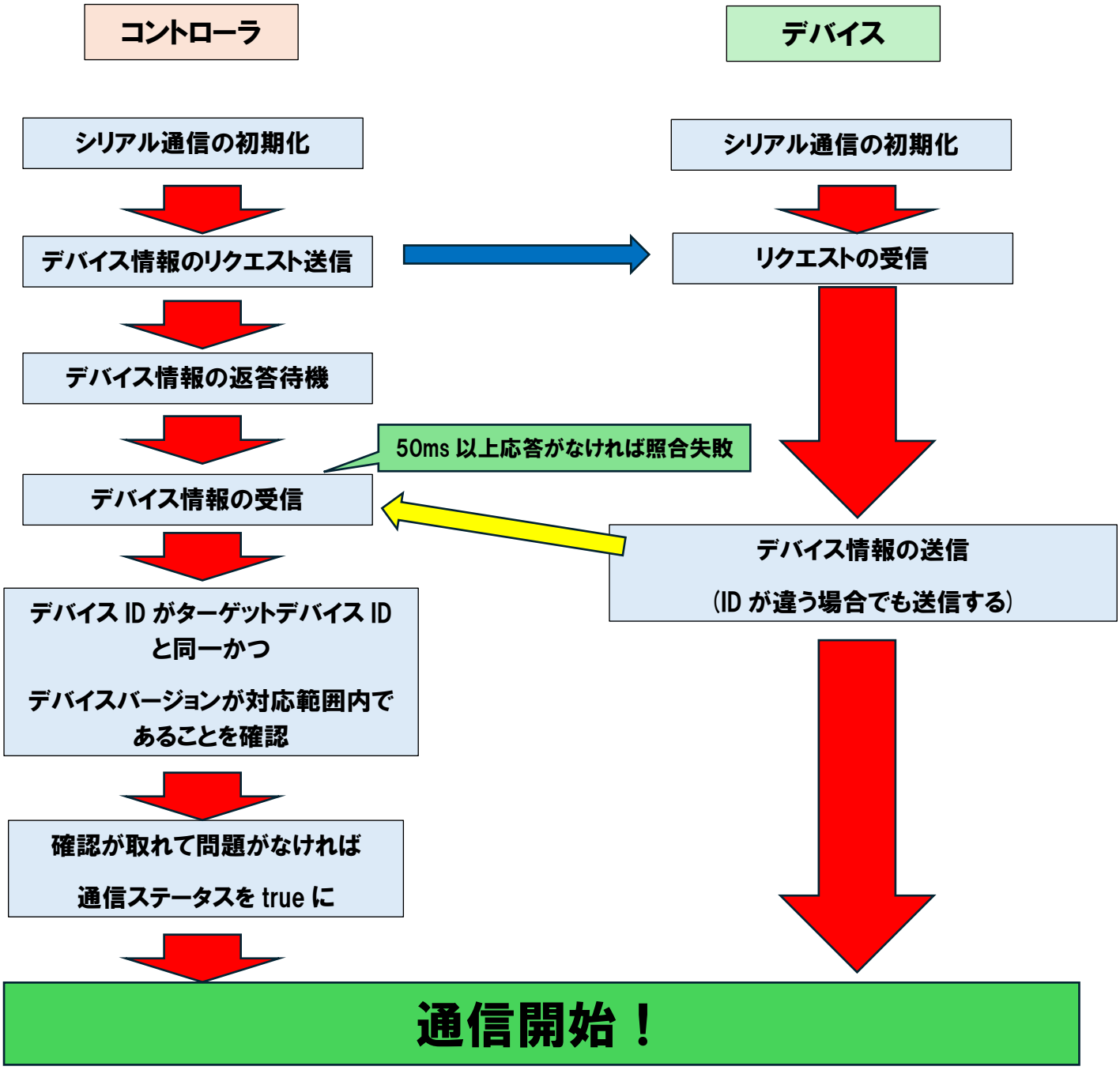
A : 送信データ数(10_{DEC})

12 ~ 0 : データ

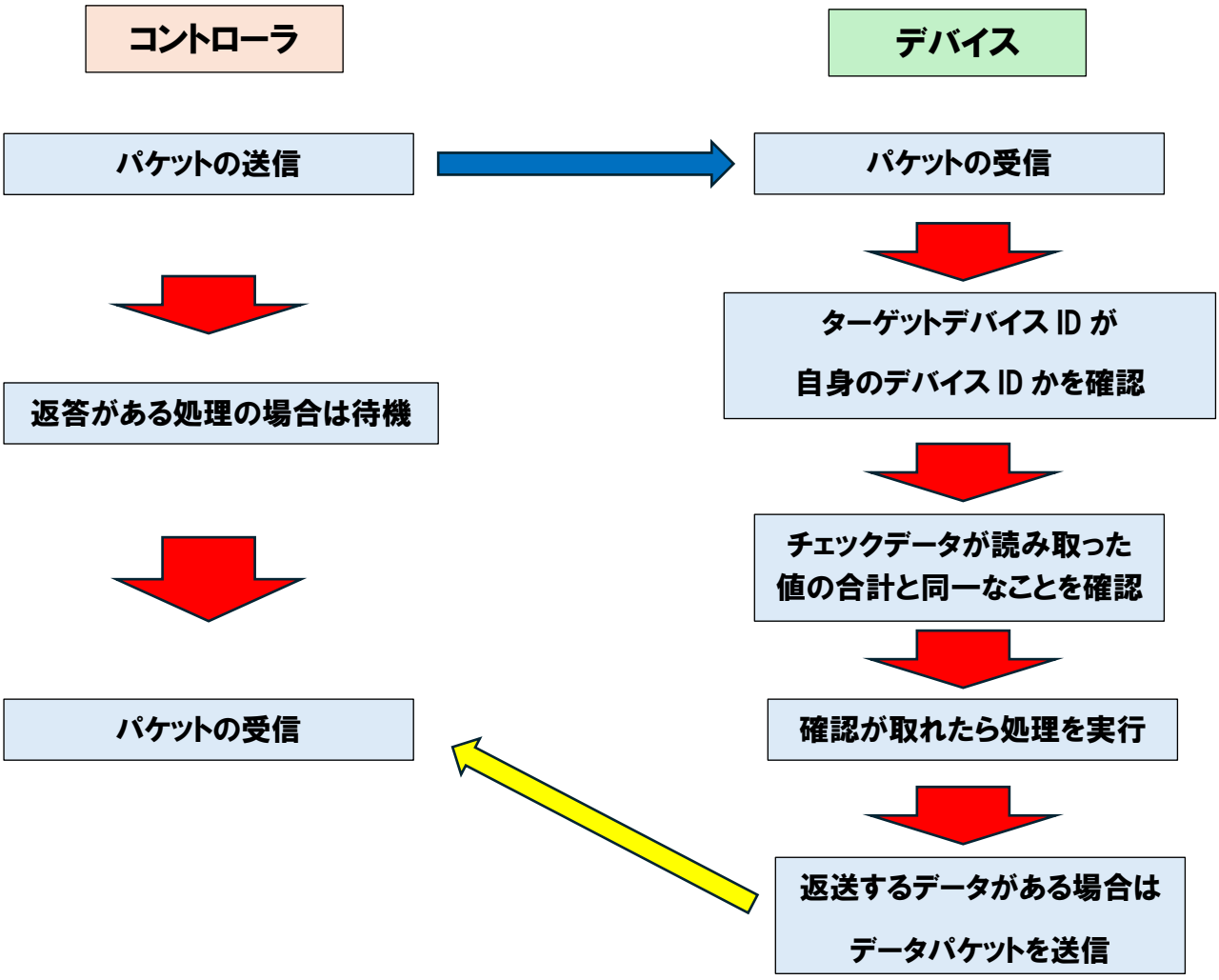
0414 : チェックデータ(1044_{DEC})

6フロー図解

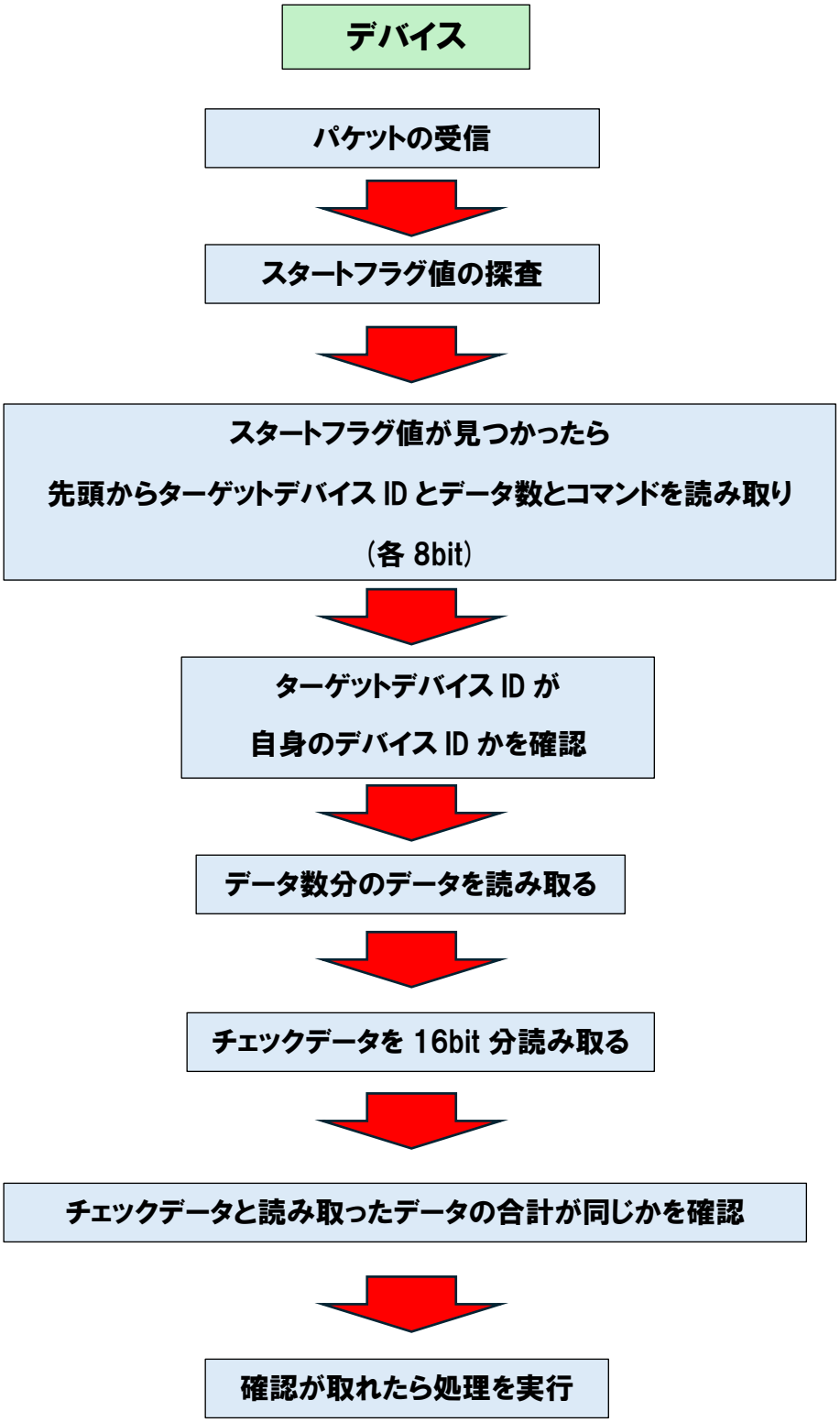
6 - 1 デバイス情報照合フロー



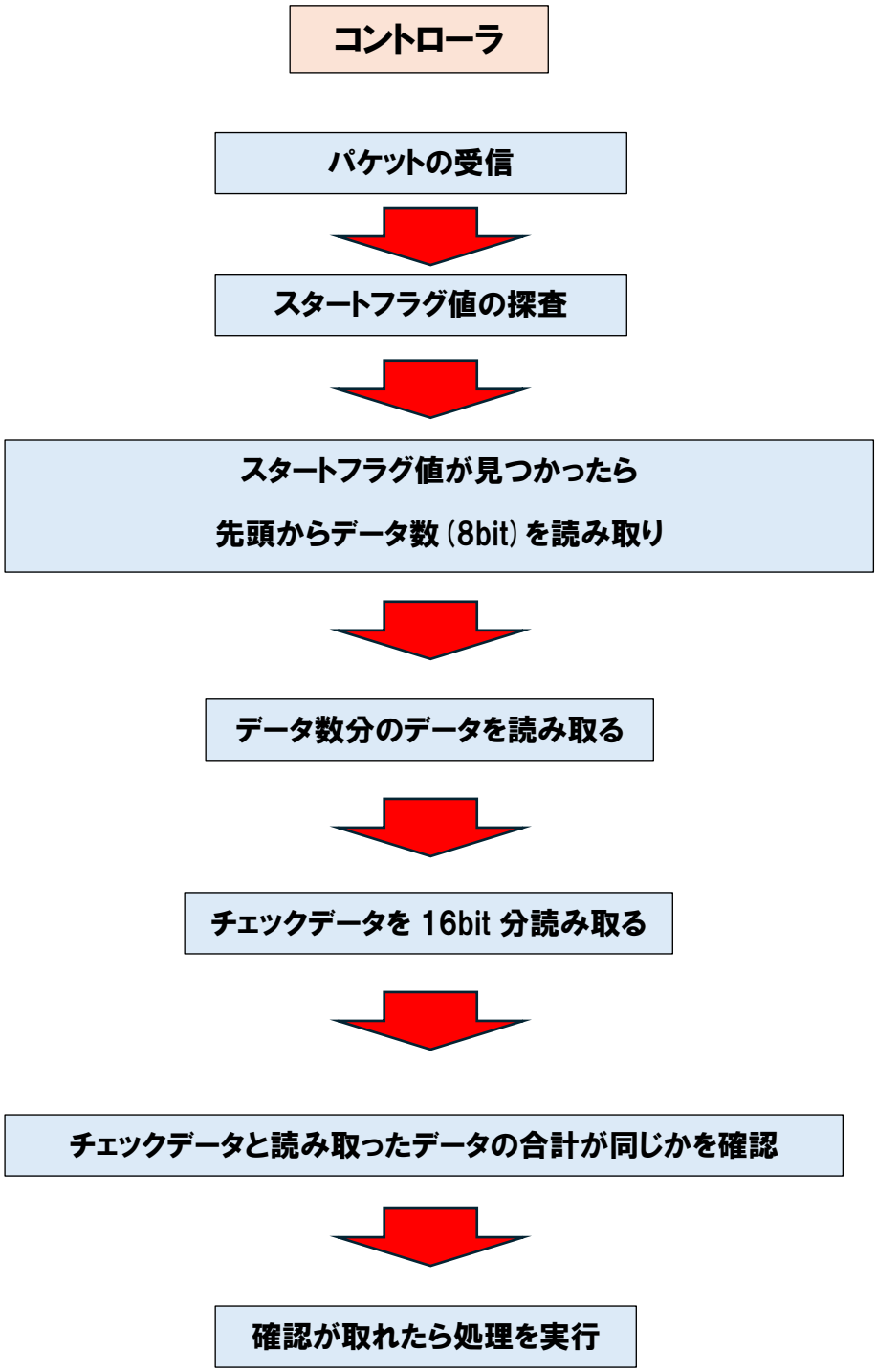
6 - 2 通信フロー(情報照合後)



6-3 デバイスパケット受信フロー例



6-4 コントローラパケット受信フロー例



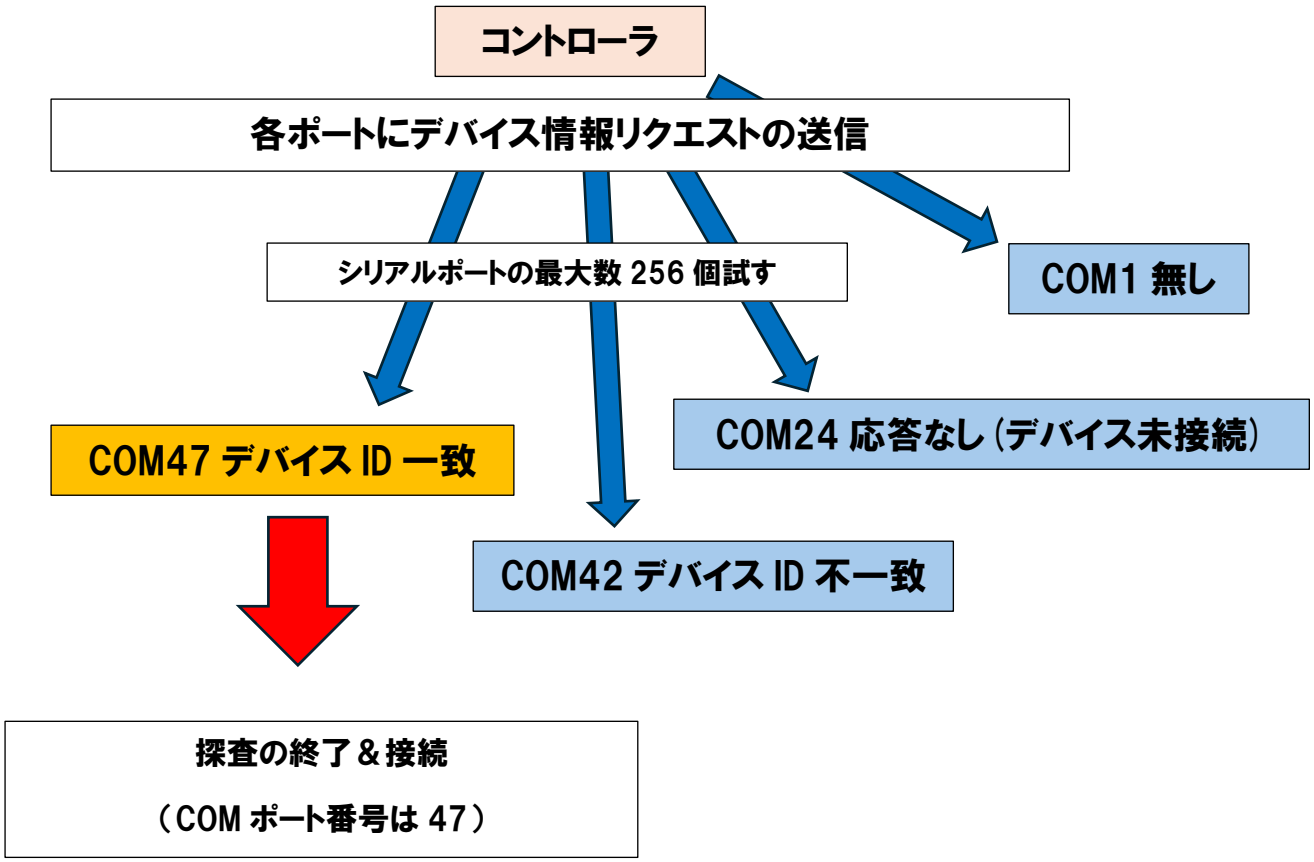
6-5 デバイス自動探査フロー例

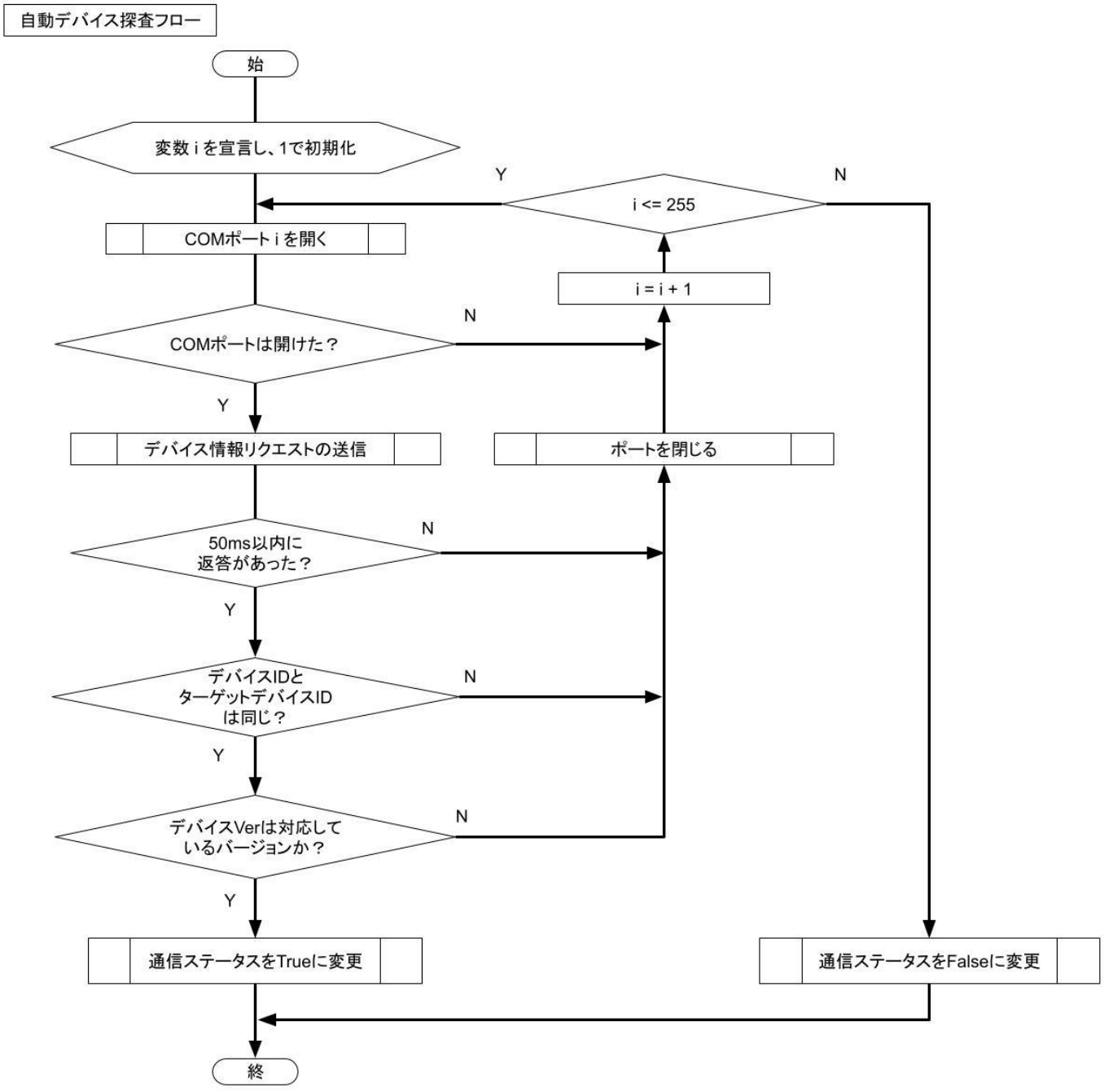
シリアル通信をする際に困ることとしてデバイスを接続する USB ポートが違っていたりデバイスに搭載されているシリアル変換 IC のドライバなどの関係で COM ポート番号が前回接続時と異なるものが割り当てられてしまう。

すると、COM ポート番号をデバイスマネージャー等を使い、調べる必要がありとても扱いづらい。

そこで ASerial ではデバイスが持つデバイス ID を利用してデバイスの自動探査処理を行うことができる。

処理の例を以下に示す。




自動デバイス探査フロー
始
変数 i を宣言し、1で初期化
i <= 255
Y
N
COMポート i を開く
i = i + 1
COMポートは開けた？
N
Y
デバイス情報リクエストの送信
ポートを閉じる
50ms以内に
返答があった？
N
Y
デバイスIDと
ターゲットデバイスID
は同じ？
N
Y
デバイスVerは対応して
いるバージョンか？
N
Y
通信ステータスをTrueに変更
通信ステータスをFalseに変更
終

7配線規定

- コントローラ側を基準にします
- 線色は

  TXD：青

  RXD：黄色

  GND：黒

(例)