中編

# 組み込みシステム設計者のための LIN 2.0マイコン実装術

## ――使用するCPU性能に応じたオプション選び

システムの記事

舘 伸幸

**前回（本誌2005年11月号，pp.92-97）に引き続き，プロトコル仕様について解説します．今回は，フレーム構造の詳細やスケジューリング，エラー処理を説明します（LIN Protocol Specification の2.1.2章以降の後半部分に相当）．また，診断に関する規定が記述されている仕様書である“Diagnostic and Configuration Specification”の内容も紹介します．（筆者）**

フレームの先頭にはブレーク（break）と呼ばれるロー・パルス（“L”レベル）が存在します（**図1**）．LIN 1.3では 同期ブレーク（sync break）という名称でしたが，LIN 2.0では単にブレークとなっています．

## 1．プロトコル詳細――フレームの設定方法

ブレークはロー・パルスが少なくとも13ビット以上であること，またスレーブは11ビット時間[注1]をしきい値として検出することが規定されています．ここでの13ビットは，マスタが±0.5％の精度で保証しなければなりません（この精度値は，“LIN Physical Layer Specification”で規定されている）．

一方，スレーブの検出しきい値は，スレーブ・ノードの動作クロックの周波数偏差を最大14％まで許容することから，11ビットで判別するようになっています．なお，プロトコル仕様書には，水晶などの精度の良いクロックを利用できるときは9.5ビットでの判定でもよい，という注意書きがあります．これは，フレーミング・エラーを利用してブレークを検出できる可能性を示唆しています．UARTの仕様にもよりますが，フレーミング・エラー発生時にUARTの受信バッファがゼロであれば，少なくとも9.5ビット以上のロー・パルスを受信したと言えるためです．

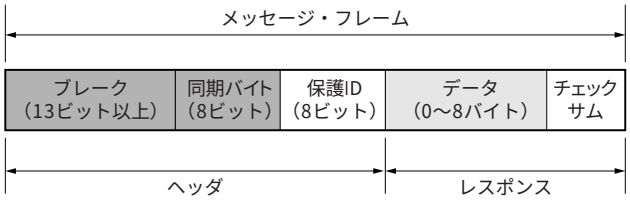



**図1　フレームの構成**

**LINのメッセージ・フレームは大きくヘッダとレスポンスからなる．ヘッダはフレームの最初の部分（ブレーク，同期バイト，保護ID），レスポンスはデータとチェックサムで構成される．**

注1：1ビット時間とは，あるネットワーク伝送速度における1ビットの伝送時間のこと．例えば，ネットワークの伝送速度が10kbpsの場合，1ビット時間＝$1 \div (10 \times 10^3) = 100\mu s$となる．

### ● フレーミング・エラーを使ってブレーク検出を簡便に

**図2**のUARTフォーマットを見てください．フレーミング・エラーとは，ストップ・ビットにおけるサンプリングで，“L”レベルが検出された場合に発生するエラーです（ス

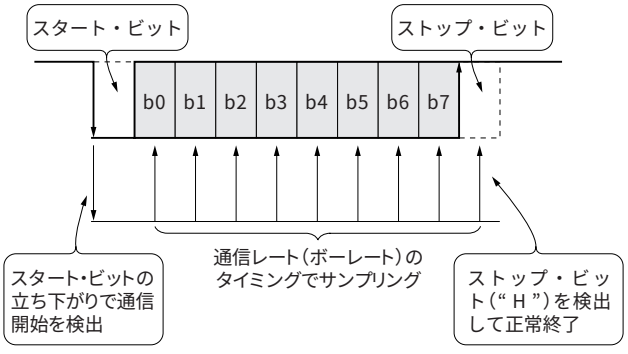



**図2　UARTフォーマット**

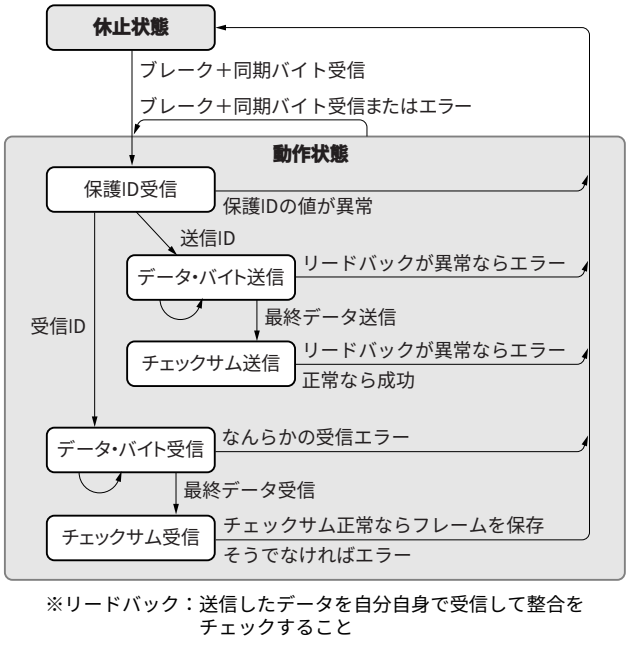
**UARTのフレームでは，ストップ・ビットは1ビットの“H”レベルである．このストップ・ビットの位置で“L”レベルが検出されることをフレーミング・エラーという．**

トップ・ビットは1ビットの“H”レベル）．フレーミング・エラーが発生するまでにサンプリングされた情報は，UARTの受信バッファ（シフト・レジスタ）に入っているので，受信バッファがゼロでフレーミング・エラーが発生したということは，すなわち，少なくとも9.5ビット（＝スタート・ビット1ビット＋データ・ビット8ビット＋ストップ・ビットのサンプリング位置までの0.5ビット）の幅のロー・パルスを検出できたと解釈できるわけです．

LIN 2.0では，ブレークと同期バイト（synch）が処理された後は動作状態（active）となります（**図3**）．そこからは，どこでもブレークを検出できなくてはなりません．ブレーク検出にフレーミング・エラーを利用できれば，簡便にこの仕様を実現できます．*RC*発振など，精度の良くない発振器の場合，休止状態（dominant）からのブレーク検出にフレーミング・エラーは使えませんが，フレーム途中のブレーク検出であれば（ボーレート補正直後なので）利用できる可能性もあります．

ただ，注記として「とくに必須ではない」との断り付きで，ブレークが部分的にデータ・バイトと重なった場合も検出できることが希望されるとあります．

この場合，フレーミング・エラーによる検出では対応できません．かといって，フレーム全域にわたってエッジ割り込みなどでロー・パルスのビット幅を検出し続けるという処理は，少なくともLINのスレーブ・ノードに使うような非力なCPUにとっては現実的ではありません．この「希望される仕様」を実現するには，ハードウェアに特別なロー・パルス検出回路を持たせるべきでしょう．低コスト・低負荷を考慮するならば，クラスタを設計する際にあえてこの仕様に挑戦しないという判断もあるでしょう．




※リードバック：送信したデータを自分自身で受信して整合をチェックすること

**図3　状態遷移**

**ブレークと同期バイトが処理された後，動作状態となる．その後は，どこでもブレークを検出できなくてはならない（そのとき受信中のフレームはエラーとして廃棄する）．**

### ● パルス幅の測定は時間的制約やCPU性能を考えて選択

ブレークの次に来る同期バイト（synch byte）は，通信レート同期のためのフィールドです．LIN 1.3では同期フィールド（synch field）という名称でした．

マスタ・タスクは，単にUARTで0x55を送信するだけです．一方，スレーブ・タスクは複数の処理手段が考えられます．水晶などの高精度の発振を利用していて十分な精度が見込まれる場合，単にUART受信を行って結果が0x55であることを確認するというのが，処理的にはもっとも簡単な方法です．精度の高くない発振回路を使用して仕様に忠実に設計する場合，タイマなどを利用してパルス幅を計測する処理が必要となります．

このあたりの詳細は“LIN Physical Layer Specification”の“Bit TIMING REQUREMENTS AND SYNCRONIZATION PROCEDURE”という章で記述されています．（理由はわからないが）パルス幅は立ち下がりエッジ間で計測しなければならないと記述されているので，実際の設計・実装でも仕様と一致しておいたほうがよいでしょう．また，

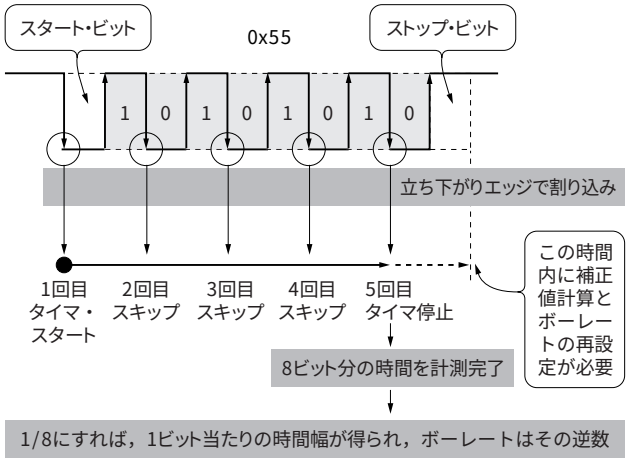



**図4　ボーレート計測**

**仕様書ではスタート・ビットの立ち下がりからビット7の立ち下がりまでの8ビット分を計測して，1/8にする方法が推奨されている．**