令和元年度 秋期 応用情報技術者試験「問7：赤外線リモコンの設計」徹底復習レポート

1. はじめに：学習機能付き赤外線リモコン設計の戦略的意義

現代のIoTデバイス開発において、物理層の信号をデジタルデータへと変換し、解析・制御する技術はシステム設計の根幹をなす。本レポートで扱う「統合リモコン」の「学習機能」とは、あらかじめ登録されたメーカ情報に加え、未登録機器の赤外線信号を解析・登録することで、未知のデバイス操作を可能にする機能である。この設計における中心課題は、単なる波形の複製ではない。外部ノイズが混在する物理環境下で、未知のパルス信号をいかに正確に「解析・登録」し、再現性の高いデータ構造へと落とし込むかという設計思想にある。また、利用者が「特定のボタンを2秒以上押し続ける」という意図的な操作をトリガーとする仕様は、誤動作を防止しつつ学習モードへ安全に遷移させるための重要なUX設計（UIガード）である。物理現象を確実なソフトウェア制御へと繋げるこのプロセスこそ、次章で述べる「物理的特性の理解」を前提とした組込み設計の醍醐味である。

2. 赤外線信号の物理的特性とデータ構造の解析

赤外線通信の設計品質を担保するには、物理層の仕様を定量的制約として捉える必要がある。本システムが扱う信号は38～40kHzの周期で点滅し、最短350マイクロ秒のパルス幅を持つ。また、制御パターン全体の最大完了時間は60ミリ秒（60,000マイクロ秒）と定義されている。ここで、システム設計上のリソース見積もり（So What?）を検証する。設問1(1)における状態数の合計最大値は $60,000/350 \fallingdotseq 171.42$ と算出されるが、物理的に不完全なパルスは論理的に無効であるため、端数を切り捨てた「171個」が論理的な上限となる。 この数値は、制御パターンを格納する配列 T の要素数を決定する直接的な根拠となる。リソースが限られた組込み環境では、メモリ断片化を避けるために **静的メモリ確保** （Static Memory Allocation）を行うのが定石である。最大172要素（Nを含む）を静的に確保することで、決定論的なメモリフットプリントを維持し、実行時の安定性を高めることができる。

3. ハードウェア構成と信号抽出アルゴリズムの整合性

物理信号のデジタル処理を効率化するには、ハードウェアによる適切な前処理が不可欠である。赤外線センサに採用された「バンドパスフィルタ」は、38～40kHzの周波数帯域のみを透過させることで、太陽光や蛍光灯などの外部ノイズを物理層で遮断し、ソフトウェア側の処理負荷を大幅に軽減している。時間測定の精度に関しては、1マイクロ秒刻みの16ビットカウンタを用いることで、最小単位350マイクロ秒に対して十分な分解能を確保している。また、ソフトウェア制御における waitEvent() 関数の実装は、省電力設計の観点から極めて重要である。イベント待ちの間、CPUを低電力状態（スリープモード）に移行させることで、電池駆動が前提のリモコンにおける稼働時間を最大化できる。設問2(1)にある「5,000ミリ秒（5秒）」のタイムアウト処理は、このイベント待ちからの復帰とシステム保護を両立させる論理的な設計である。

4. 制御パターン抽出プログラムの論理構造分析

本プログラムは、物理的な信号変化を起点とするイベント駆動型アルゴリズムを採用している。特筆すべきは、赤外線センサからの通知を格納する **FIFO動作のキュー** の存在である。これにより、物理信号の割り込み処理と、メインの解析ロジックを時間的に分離（ **デカップリング** ）でき、350マイクロ秒という短パルスの連続においてもイベントの取りこぼしを防いでいる。ロジックの堅牢性については、制御パターンが「必ずON状態から始まる」という仕様に基づき、最初の分岐 c で「ONイベント」以外（OFFイベントやタイマイベント）をバリデーションエラー（rst ← False）として排除している。 配列 T への格納ルールは以下の通りである。

- **T2×k（添字が偶数）：** a＝ON状態の長さ
- **T2×k+1（添字が奇数）：** b＝OFF状態の長さ 変数 N が0から始まり、ループの最後でインクリメントされるため、最初のON状態が T0（偶数）に正確に格納される。 また、d（今回値）を 前回値 に更新する処理は、常に最新の信号エッジを時間測定の基点として引き継ぐための必須の更新処理である。

5. カウンタオーバーフローの検知と補正ロジックの評価

有限リソースである16ビットカウンタを扱う際、必ず直面するのが「桁溢れ（オーバーフロー）」の問題である。本システムのカウンタは65,535を超えると0に戻るため、測定中に周回が発生すると、単純な差分計算（今回値 - 前回値）の結果が負の値（TN < 0）となる。

この事象に対し、本設計では「TNに65,536（ 2^{16} ）を加算する」補正ロジックを実装している。ここで重要なのはデータ型の整合性である。getCount() は16ビットのハードウェア値を **ゼロ拡張** して32ビット符号付き整数として返しており、上位16ビットは常に0である。このため、負の値が発生した際に 2^{16} を加算する処理は、32ビットの空間上で16ビットの周回分を算術的に復元し、実時間を正しく再構成するための高度な設計的配慮といえる。これはハードウェアの物理的限界をソフトウェアのデータ構造で補完する、組込み設計の定石である。

6. 総括：高度な組込みシステム設計に向けた教訓

本問題の設計事例は、ハード・ソフト協調設計における以下のベストプラクティスを提示している。

- **物理仕様に基づくリソース設計：** 最短・最長時間の制約から、静的メモリ確保の妥当性を定量的に導出する。
- **ハードウェアによる負荷分散：** バンドパスフィルタやFIFOキューを活用し、ソフトウェアのリアルタイム性と信頼性を担保する。
- **例外処理の厳密な実装：** カウンタオーバーフローやバリデーションエラーを予測し、データ型の特性を活かした補正を行う。本設計は、未知の信号に対応する「汎用性」と、物理的制約・例外を克服する「確実性」を高度な次元で両立させており、組込みシステムエンジニアが習得すべき規範的なモデルである。**注目すべき技術キーワード**
- **バンドパスフィルタ**
- **オーバーフロー**

- イベント駆動
- 静的メモリ確保
- 割り込み処理とのデカップリング
- ゼロ拡張