

令和3年度春期 応用情報技術者試験 午後問7「デジタル補聴器の設計」復習レポート

1. システム概要と設計の基本方針

本設問は、デジタル補聴器という実製品の設計を題材に、組込みシステムにおける**「リアルタイム制約」「消費電力の最適化」「デジタル信号処理」**という三つの重要テーマを統合的に問う高度な問題です。補聴器は、装着者が周囲の音や自身の声を違和感なく聞き取るために極めて低いレイテンシ（遅延）が求められる一方で、ウェアラブルデバイスとして極限の低消費電力設計も要求されます。本システムでは、ソーステキストにある「特定の帯域の音が聞き取りにくい利用者」という具体的なニーズに対し、ソフトウェアによる柔軟なパラメタ調整で応えています。システム構成と処理の流れを整理すると以下の通りです。

- **ハードウェア構成**：ADC（A/Dコンバータ）、バッファ、制御部、DAC（D/Aコンバータ）で構成。サンプリングおよび変換時間は、ADC/DACともに24kHzの低速クロックで制御される。
- **信号処理フロー**：8つの帯域分割、帯域ごとの音量調整、合成、そして急激な変化を抑えるAVC（自動音量調節）の順に実行される。
- **外部連携**：利用者はスマホアプリを介し、BT（Bluetooth）モジュールを通じて帯域ごとの音量パラメタ等を動的に変更できる設計となっている。この一連の処理を破綻なく継続するためには、ハードウェアとソフトウェアの接点である「バッファ」の緻密な設計が不可欠です。次節では、その定量的な設計根拠を解説します。

2. ハードウェア構成とバッファサイズの定量的分析

組込みシステムにおいて、入出力バッファの設計はデータ転送の安定性と並行処理の効率を決定づけます。本システムでは、ADCから送られる連続的なデータを「フレーム（64サンプル）」単位で処理するため、適切なバッファサイズの算出が設計の第一歩となります。空欄aのバッファサイズは、ソーステキストの制約に基づき以下の計算で導出されます。**バッファサイズの算出プロセス**

1. **1サンプルあたりのデータ量**：ソースにある「16ビットの符号付き整数」という記述より、 $16\text{ビット} \div 8 = 2\text{バイト}$ 。
2. **1フレームあたりのデータ量**：「64サンプルを1フレームとする」ため、 $2\text{バイト} \times 64\text{サンプル} = 128\text{バイト}$ 。
3. **バッファ全体の構成**：入出力バッファは「それぞれ三つのブロックで構成」され、各ブロックに1フレーム分を格納するため、 $128\text{バイト} \times 3\text{ブロック} = 384\text{バイト}$ 。図3に示される「3ブロック構成」の採用は、単なる容量確保ではなく**「ハードウェアによる排他制御の解決」**を意図しています。「ADCが書き込んでいるブロック」「制御部が演算しているブロック」「DACが読み出しているブロック」を物理的に分けることで、同一メモリへの競合を避けつつ、パイプライン的な並行処理を実現しています。このバッファ構造が確定したことで、次はデータが各ブロックを通過する際に蓄積される「処理遅延」の評価へと進みます。

3. リアルタイム処理における遅延時間の評価

補聴器において、音の入力から出力までの遅延（レイテンシ）は、利用者の QOL（生活の質）に直結する致命的な指標です。遅延が大きすぎると、自分の声が遅れて聞こえる「不快なエコー」の原因となります。マイク入力からイヤホン出力までの遅延時間（空欄 b）を算出する際、前提としてソースの「ADC及びDACの変換時間は無視できる」という条件を考慮する必要があります。

1. **1フレームあたりの時間**：24kHz（1秒間に24,000回）サンプリングで1フレームが64サンプルであるため、 $64 \div 24,000 = 8/3,000$ 秒となります。※1フレームは約 2.66...ms ですが、計算精度を保つため分数のまま処理します。
2. **発生する遅延の論理（図3参照）**：
3. **入力フェーズ**：最初の1フレーム時間をかけてADCがブロック1へ書き込む。
4. **処理フェーズ**：次の1フレーム時間で制御部がブロック1を処理し、出力バッファへ書き込む。
5. **出力フェーズ**：さらに次の1フレーム時間でDACがブロック1から読み出し、アナログ変換して出力する。このように、あるデータがシステムを通り抜けるには **計3フレーム分** の時間が必要です。計算式は $(8/3,000) \times 3 = 8/1,000$ 秒、すなわち 8.0ms となります。本設問で「8」ではなく 8.0ms と記述することは、設計者として有効数字と精度（小数第1位までという指示）を遵守する基本姿勢を問われています。次は、この時間枠内でいかに演算を完結させるか、クロック周波数の設計に焦点を移します。

4. ソフトウェア処理と電力最適化のトレードオフ

組込み設計の要諦は「性能（速度）と消費電力のトレードオフ」の最適化にあります。実行時間は供給クロック周波数に反比例するため、必要最小限のクロック選定が電池寿命を延ばす鍵となります。まず、1フレームあたりの最大実行時間 T_d の計算式（空欄 c, d, e）は、以下の通りです（順不同）。 $T_d = 8 \times T_f + T_s + T_a$ ここで「①信号処理」が **8倍** されているのは、ソースにある「8つの帯域に対して繰り返す」というアルゴリズムの構造を忠実に数式化したものです。次に、基本周波数 f_0 供給時の合計実行時間を表1から計算します。

- ① 信号処理（8帯域分）： $0.30 \times 8 = 2.40 \times T_{frame}$
- ② 合成処理： $0.05 \times T_{frame}$
- ③ AVC処理： $0.20 \times T_{frame}$
- **合計**： $2.65 \times T_{frame}$ 処理を T_{frame} （受信完了通知の周期）内に完結させるためには、 $T_d \leq T_{frame}$ である必要があります。実行時間の性質 $T_{new} = T_{old} \times (f_{old}/f_{new})$ に基づくと、クロックを 2.65 倍以上に引き上げなければなりません。高速クロックは「 f_0 の整数倍」という制約があるため、条件を満たす最小の整数は **3倍** です。消費電力は周波数に比例するため、この最小値を選択することが、省電力設計における最適解となります。枠組みが決定したところで、最後は音質の快適性を守る「音量調節アルゴリズム」の論理構造を精査します。

5. 自動音量調節（AVC）アルゴリズムの論理構造

AVCは、目標音量へ「Mフレームかけて徐々に変化させる」ことで、急激な音圧変化による利用者の不快感を防ぎます。図4のフローチャートにおける空欄の役割は以下の通りです。

- 空欄 ア : $dv > 0$ and $v \geq vt$
- 音量を上げている ($dv > 0$) 過程で、現在の音量 v が目標 vt に「到達または超過」したかを判定する論理。
- 空欄 イ : vt / 空欄 ウ : 0
- 調整完了時に現在の音量を目標値に固定し、変化量 dv を初期化して次のトリガを待つ処理。ここで、網掛け部の判定 ($dv = 0$ かつ $v \neq vt$) が “Yes” となるケースは、本システムが**「整数演算」であることに起因する重要な例外処理です。
実務的視点による分析：整数演算の罫 変化量 dv は $(vt - v) \div M$ で算出されます。例えば $M = 2$ のとき、目標との差 $(vt - v)$ が 1 であれば、 $1 \div 2 = 0.5$ となりますが、整数演算では 0 に切り捨てられます。このままでは $dv = 0$ となり、永遠に目標音量に到達しません。この判定は、目標との微小な差が演算で消失するのを防ぎ、収束を保証するための「ダメ押し」の処理なのです。この判定結果が “Yes” となる状態を要約すると、「目標との微小な差が整数演算で消失し、変化量が0となったが未到達の状態」**と言えます。こうした境界条件への配慮が、実装ミスを防ぐ極めて重要な鍵となります。

6. 総括：試験対策に向けた重要ポイントの整理

本問題は、サンプリング理論、バッファ制御、リアルタイム制約、電力最適化という、組み込みエンジニアが直面する課題が凝縮された傑作問題です。試験合格、そして実務への応用に向けて、以下の3点をチェックリストとして活用してください。

1. ハードウェア構造から遅延フレーム数を読み取る能力：図3のようなバッファの動きを追跡し、入力から出力まで何フレーム要するかを可視化すること。
2. 単位換算と有効数字への執着： s から ms への換算ミスや、 8.0 といった精度指定の無視は、組み込み設計では致命的な欠陥に繋がります。
3. 整数演算の制約（切り捨て）を常に疑う： $dv = 0$ のケースのように、数式上は動くはずのロジックがコンピュータ上でどう振る舞うかを洞察すること。本問題で扱われた「低遅延と省電力の両立」というテーマは、現代のウェアラブルデバイスやIoT機器の設計思想そのものです。この論理的な思考プロセスを自分のものにすることで、単なる知識を超えた「設計の洞察力」を養ってください。