

平成 21 年度 秋期 応用情報技術者試験

【問 7】 復習レポート：組込みシステムのタスク制御とリアルタイム性の最適化

1. はじめに：本問題の目的と実務的意義 現代の組込みシステム開発において、割込み処理、タスク間通信、そしてリアルタイム性の確保は、製品の信頼性と市場競争力を決定づける極めて戦略的な要素です。特に、限られた計算資源の中で **決定的 (Deterministic) な振る舞い** を保証し、ユーザーの操作に対して遅延のない応答を返す設計能力は、シニアエンジニアにとって必須の素養といえます。本問は「デジタルフォトフレーム」という具体的な製品を題材に、以下の技術的スキルを多角的に問う構成となっています。

- * **タスク優先度設計**：実行順序の最適化による応答性の最大化。
- * **イベントドリブン制御**：非同期処理におけるタスク間の安全な連携。
- * **計算資源の制約管理**：キューの設計限界と、ワーストケース（最悪実行時間：WCET）を想定した定量的評価。これらは単なる試験対策を超え、システムの堅牢性やユーザー体験（UX）に直結する概念です。本レポートでは、アーキテクチャの解剖から定量的なパフォーマンス設計まで、実務的な視点で深く掘り下げていきます。まずは、システムの骨格となるハードウェアとソフトウェアの役割分担を整理します。

2. システムアーキテクチャの解剖：役割と責務の明確化 マルチタスクシステムにおいて、各コンポーネントの「責務の分離」はシステム全体の効率と保守性を左右します。各タスクが特定の役割に特化し、イベントを介して情報をやり取りすることで、複雑な状態遷移を整然と管理可能にします。

2.1 ソフトウェア構成要素の構造化 ソースコンテキストに基づき、本システムの主要要素とデータフローの依存関係を整理します。

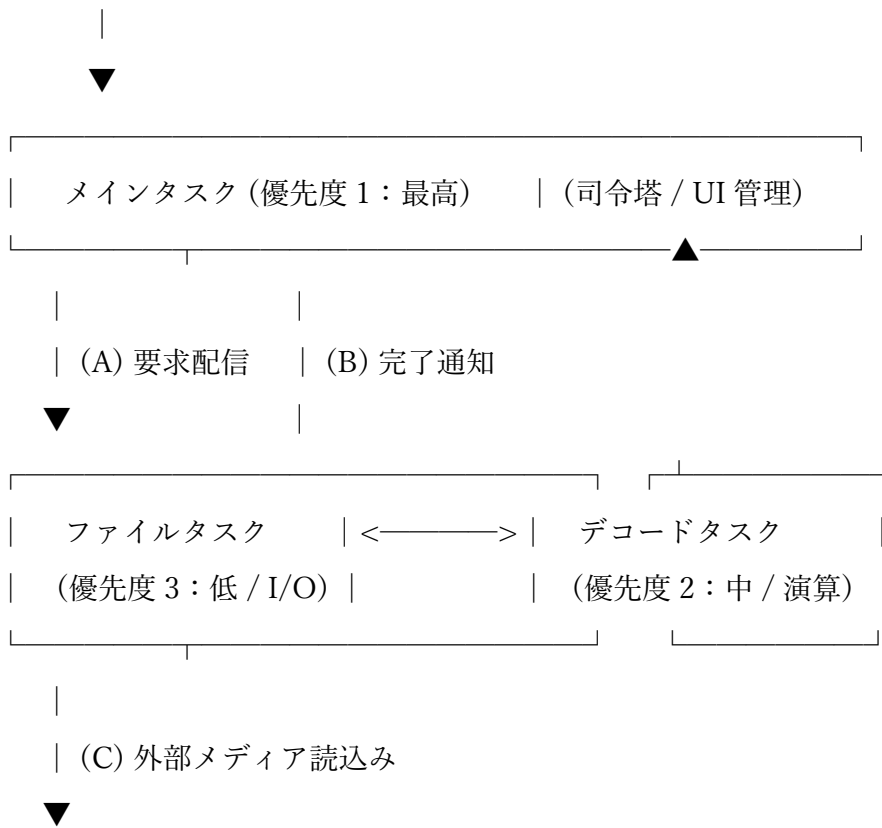
表 1：タスクおよび割込みハンドラの詳細定義

分類	名称	優先度	主な機能・責務	送受信イベント（データフロー詳細）
-----	-----	-----	-----	-----
割込み	キー	-	左/右キー押下検知	送信： キー （左・右の情報）
割込み	タイマ	-	設定時間経過の検知	送信： タイマ
タスク	メイン	1 (高)	画像表示順序の管理、全体制御	受信：キー、タイマ、読み込み完了、デコード完了 送信： 読み込み要求 (パス)、 デコード要求 (パス, ポインタ, サイズ)
タスク	デコード	2	画像データの表示変換	受信：デコード要求 送信： デコード完了 (パス, ポインタ, サイズ)
タスク	ファイル	3 (低)	外部メディアからのデータ取得	受信：読み込み要求 送信： 読み込み完了 (パス, ポインタ, サイズ)

2.2 アーキテクチャ構造図（テキストモデル） タスク間の階層構造とイベントの伝播経路を以下に可視化します。

修正後のアーキテクチャ構造図

[割込み層：ハードウェア通知] （キー押下 / 時間経過）



[SD カード等の外部メディア]

構造とイベントの流れの整理

このシステムは、役割と優先度の異なる **3つのタスク**が協調して動作しています。

1. 各タスクの優先度と役割

- **メインタスク（優先度 1：最高）**
 - **役割:** システム全体の「司令塔」および「UI（画面表示・操作）管理」です。
 - **動き:** 割込み層（キーが押された、または一定時間が経過したというハードウェアの通知）を直接受け取り、処理を開始します。
- **デコードタスク（優先度 2：中）**
 - **役割:** 画像のデコードなどの「重い演算処理」を担当します。
- **ファイルタスク（優先度 3：低）**

- 。 **役割:** SD カードなどの外部メディアからデータを読み込む「I/O（入出力）処理」を担当します。

2. イベント伝播（A～C）の流れ

1. **割込み通知:** ユーザーがキーを押すなどのアクションがあると、**メインタスク**に通知されます。
2. **要求配信 (A):** メインタスクは自分で重い処理をせず、**ファイルタスク**と**デコードタスク**に処理を依頼（要求配信）します。
3. **データ読み込み (C):** ファイルタスクが、SD カード等から画像データを読み込みます。
4. **協調動作:** ファイルタスクとデコードタスクが連携し、読み込んだデータをデコード（復元）します。
5. **完了通知 (B):** デコードが完了すると、**メインタスク**に完了したことが通知され、画面が更新されます。

2.3 「レストラン比喻」による直感的理解 この構造は、一流レストランのオペレーションに例えると非常に明快です。

* **メインタスク（ウェイター）：** 客（ユーザー/タイマ）の要望を聞き、厨房に指示を出し、出来上がった料理をテーブル（画面）に並べる司令塔。

* **ファイルタスク（厨房スタッフ）：** 冷蔵庫（外部メディア）から指定された食材（画像データ）を取り出す役割。

* **デコードタスク（シェフ）：** 食材を調理し、食べられる状態（表示データ）に仕上げる専門職。

* **砂時計画像（「準備中」の札）：** 料理が届くまでの間、客の不安を解消するためのフィードバック。メインタスクが最高優先度を持つのは、ユーザーへの即時応答（UI 制御）と、背後の「厨房」への適切な指示出しを最優先すべきだからです。

3. 動作シーケンスの論理構造：自動切替えと手動切替えのメカニズム 組込みシステムにおいて、イベントは状態遷移のトリガーです。定常的な「自動切替え」と、ユーザーによる「手動介入」がどう排他的に、あるいは連続的に制御されるかを解析します。

3.1 自動切替えのライフサイクル（図 1 解析） タイマを起点とした標準フローは、**非同期処理** を巧みに活用しています。

1. **タイマイイベント受信：** 表示時間の終了を検知。

2. a 砂時計画像を表示：次の準備が整うまでの視覚的フィードバック（仕様(3)）。
3. 表示準備要求：ファイルタスクとデコードタスクに順次依頼。
4. b 画面表示用メモリ領域に格納：完成した表示データを可視化。
5. c タイマ（5 秒間）をセット：ここで現在の画像の表示計測を開始。
6. バックグラウンド先行準備：仕様(2)に基づき、メインタスクはタイマセット直後に**「次」の画像の表示準備（読み込み・デコード）を即座に開始します。これにより、5 秒間の表示時間を有効活用します。

3.2 手動切替えの割込み制御（図 2 解析） ユーザー操作による例外処理では、スケジュールの再構築が必要です。

* d タイマのセットを解除：進行中の「5 秒後に自動更新」という予約を一旦破棄し、手動操作を優先します。

* 即時処理：直ちに a 砂時計を表示し、指定された画像の準備フローを開始します。

3.3 設問 1 の論理解明

* a = キ（砂時計画像を表示）：準備未完了時の代替表示として仕様(3)に合致。

* b = エ（画面表示用メモリ領域に格納）：デコード完了後、VRAM 等へ転送するフェーズ。

* c = ク（タイマをセット）：表 4 の動作②に基づき、表示開始と同時に 5 秒を計測。

* d = ケ（タイマのセットを解除）：表 4 の動作①に基づき、既存のスケジュールをリセット。

4. システムの限界と信頼性：イベントキューのオーバーフロー解析 キューイング理論において、要求速度が処理速度を上回ると、未処理イベントの蓄積が発生します。本問のケースは、システム設計上の致命的なボトルネックを露呈させています。

4.1 現象：右キー連続押下時の画像スキップ ユーザーが「右キー」を連打した際、一部の画像が表示されない現象は、イベントの破棄という形で顕在化します。

4.2 ボトルネックの特定：高優先度生産者と低優先度消費者の対立 本質的な原因は、メインタスク（優先度 1）とファイルタスク（優先度 3）の優先度差と、外部メディアの物理的な遅延の組み合わせにあります。

1. 高頻度の生成：高優先度のメインタスクは、キーイベントのたびに即座に「読み込み要求」をファイルタスクへ送信します。

2. 低速な消費：ファイルタスクは優先度が低いうえ、外部メディアからの読み込みという時間のかかる処理を実行中のため、要求を消化しきれません。

3. イベント破棄：ファイルタスクのイベントキューが上限に達し、メインタスクからの「読み込み要求」が仕様に基づき破棄されます。* 解答：読み込み要求イベントが破棄されたから

4.3 影響度評価とアーキテクトの視点 これは「要求が無視される」というユーザー体験上最悪のシナリオです。この「スターベーション（飢餓）」に近い状態を避けるには、キューの拡張だけでなく、ファイルタスクがビジー状態のときはメインタスク側でキー入力を抑制する、あるいは古い要求をキャンセルして最新の要求を優先する「LIFO 型」の制御検討が必要です。

5. 定量的パフォーマンス設計：ファイルサイズと処理時間の限界値「勘」を排除し、数理モデルによってデッドライン（表示期限 5 秒）**を保証するための定量評価を行います。

5.1 数理モデルの構築 ファイルサイズを A （M バイト）とし、ソースの仕様に基づき変数を導出します。

* 読み込み時間 e ：速度 2M バイト/秒 より、 $A/2$ 秒。

* デコード時間 f ：1M バイト当たり 20 ミリ秒 より、 $20A/1000$ 秒。

5.2 制約条件の適用と計算 「読み込み + デコード ≤ 5 秒」を満たす不等式を解きます。

$$\frac{A}{2} + \frac{20A}{1000} \leq 5$$

両辺を 1000 倍して整理：

$$500A + 20A \leq 5000$$

$$520A \leq 5000$$

$$A \leq \frac{5000}{520} = \frac{125}{13} \approx 9.61538\dots$$

5.3 結論としての閾値 g 設問の指示（小数第 2 位以下切り捨て）に従い、上限値は **9.6M バイト** となります。ここで「切り捨て」が指示されるのは、9.61MB を許容して 5 秒を超過するリスクを排除するためであり、組込み設計における **安全側の設計思想** を反映しています。実務では、メディアの劣化によるリトライ等で WCET が変動する「コーナーケース」を考慮し、さらなるマージンを設けるのが通例です。

6. 総括：高度組込みエンジニアに求められる視点 本解析を通じて、堅牢な組込み製品を支える「三位一体の設計思想」が浮き彫りになりました。

1. **タスク構造の最適化**：UI の応答性を司る高優先度タスクと、重い計算を担う低優先度タスクを、イベントドリブンで疎結合に保つ。
2. **シーケンスの厳密性**：自動と手動の切り替え、割込みによる状態のリセットなど、あらゆる外部イベントに対して状態の整合性を保つ。
3. **定量的根拠の提示**：物理的なデバイス速度に基づき、ファイルサイズ等のパラメータ上限を数理的に定義し、デッドラインを死守する。

実務への応用ポイント

- * **非同期処理の落とし穴**：高優先度タスクからの「過剰な要求」が低優先度タスクのキューを溢れさせるリスクを常に予見せよ。
- * **UX の維持**：砂時計表示のような「中間状態」のフィードバックこそが、低価格なハードウェア資源で高い UX を実現する鍵となる。
- * **決定的設計の追求**：数理モデルで導き出した 9.6MB という数値は、システムが信頼性を担保できる物理的な境界線である。本レポートが、論理的かつ数値的な根拠に基づいた高度なシステム設計への一助となることを願います。