RTC-Library-FUKUSHIMA

設計ノウハウ集

＜本ドキュメントの目的＞

本ドキュメントは、組み込みソフトウェア開発及びロボットコンポーネントソフトウェア開発において培った経験を基にノウハウ集としてまとめ、ロボット・テクノロジー・コンポーネント（以下 「RTC」という。 ）開発の一助になることを目的として作成しました。

ノウハウをまとめるにあたり、使用対象者の利便を考えチェック欄を設けてあります。レビュー時にご利用ください。

Ver.1.0

発行日 2017 年 12 月 15 日

RTC-Library-FUKUSHIMA

目次

1.使用対象者

本ドキュメントは、RTC-Library-FUKUSHIMA にオープンソースソフトウェア(英: Open-source software, 略称: OSS)の RTC やパッケージを登録する方を対象としています。

2.設計工程

2.1.詳細設計

No	Category	設計時の考慮点	Must /Want	OK/NG /対象外
1	初期化	RTC で使用する全ての変数を意味のある値で初期化しているか？ 例：signed char で宣言した変数を 0xFF で初期化すると“-1”を意味する。この初期化は意図通りか?	Must	
2	初期化	RTC で参照している変数を初期化せずに参照していないか？ （変数の値が不定な状態で参照することがないこと）	Must	
3	変則入力	Key 多重押しに関する振る舞いは明確になっているか？ 例：多重押しを検出した場合、Key Off 検出と同じ振る舞いとする。など	Must	
4	変則入力	RTC 起動前から押されている Key の振る舞いを明確にしているか？ 例：RTC 起動前から押されている Key は無効にする。など	Want	
5	変則入力	入力を待っている処理があった場合、その処理が中断された時に初期化や保持するデータは明確になっているか？ 例：Key の押し続け中に多重押し等により中断されたときに、押し続けのデータは無効になっているか？	Want	
6	異常入力	制御ハードウェアや接続 RTC より不定なデータを読み込んだ時にどのようになるか明確になっているか？データ破棄や初期化などを考慮しているか？（不定データを入力したときの誤動作を防止）	Must	
7	電源供給不足による誤動作	制御ハードウェアに電源が供給されていないのにスイッチの入力や制御 IC 等への出力をするような設計になっていないか？ （制御ハードウェアがどこから電源を取っているのか明確にする）	Must	
8	電源安定不足による誤動作	制御ハードウェアの電源が安定していないのにスイッチの入力や制御 IC 等への出力をするような設計になっていないか？ （制御ハードウェアが安定する時間を明確にする）	Must	
9	閉ループ	入力待ち処理では、必ずガードタイマまたはリトライカウントを設けているか？ ガードタイマ時間をオーバーした場合に、どのように処理するかは明確になっているか？ （期待した入力が無かった場合、どのような動作にならなければいけないかを考慮する）	Must	

2.2.コーディング

No	Category	設計時の考慮点	Must /Want	OK/NG /対象外
1	初期化	RTC で使用する全ての変数を意味のある値で初期化しているか？ 例：signed char で宣言した変数を 0xFF で初期化すると“-1”を意味する。この初期化は意図通りか?	Must	
2	初期化	RTC で参照している変数を初期化せずに参照していないか？ （変数の値が不定な状態で参照することがないこと） 例：AUTO 変数を初期化せずに参照していないか？	Must	
3	データ保証	データ送信中及び送信待ち中に送信データを上書しない設計になっているか？	Must	
4	データ保証	受信中のデータは、受信が完了するまで参照できないように考慮されているか？	Must	
5	独立性	コンパイルスイッチで処理内容が異なる場合、コンパイルスイッチの値が変わっても正しく処理が動作することを確認したか？	Must	
6	独立性	取得関数を削除した場合、取得関数の呼び出し元関数で取得関数を使用しなくても問題ない設計になっているか？	Must	
7	独立性	取得関数の処理内容を変更した場合、呼び出し元関数に影響が無いことを確認したか？	Must	
8	保守性	同じタイミングで実行する処理がある場合、関数化しているか？ 例：同じタイミングで初期化が必要な変数などがある場合、関数化する。	Want	
9	保守性	変数に同じ値をセットする箇所が二か所以上ある場合、値を定数化しセットしているか？	Want	
10	保守性	定数の削除、値の変更を行った場合、 定数を使用している全ての処理に影響がない事を確認したか？	Must	
11	保守性	変数のサイズに変更があった場合、プログラムに影響がないようになっているか？ （バッファサイズの変更をするときは、プログラムに影響がないように設計することを考慮する）	Must	
12	閉ループ	入力待ち処理では、必ずガードタイマまたはリトライカウントを設けているか？ ガードタイマ時間をオーバーした場合に、どのように処理するかは明確になっているか？ （期待した入力が無かった場合、どのような動作にならなければいけないかを考慮する）	Must	

13	変則入力	入力を待っている処理があった場合、その処理が中断された時に初期化や保持するデータは明確になっているか？ 例：Key の押し続け中に多重押し等により中断されたときに、押し続けのデータは無効になっているか？	Want	
14	異常検出	処理が失敗し、致命的な事象が発生する場合、ガード処理は入れているか？ 例：メモリーを確保できなかった場合、通常の処理を実行せずにユーザーに異常状態を通知するなど	Must	
15	リソース管理	RTC で確保したメモリーを適切なタイミングで開放しているか？（メモリーリークを起こしていないか？を確認する）	Must	
16	ケアレスミス	加算/減算処理がある場合、加算時のオーバーフロー、減算時のアンダーフローを考慮しているか？	Must	
17	ケアレスミス	除算処理がある場合、0 で割る事はないか？	Must	
18	ケアレスミス	テーブルデータを使用している場合、範囲外のチェックに問題はないか？また、テーブルの MAX 値は正しいか？	Must	
19	ケアレスミス	変数に値をセットする場合、宣言した型より大きな値をセットする事はないか？	Must	
20	ケアレスミス	コネクション処理を実行した場合、クローズ処理を実行しているか？ 例：ファイルのファイルクローズを確実に実行しているか？	Must	

２.３.単体テスト

No	Category	設計時の考慮点	Must /Want	OK/NG /対象外
1	網羅性	関数内の全分岐をテストケースで網羅しているか確認したか？	Want	
2	境界値テスト	データの境界値分析を行い、適切な値でテストを実施しているか？ 例：1-10 の範囲の項目の場合、0 と 11 は無効になり,1 と 10 は正常に処理されること	Must	
3	異常値テスト	データの異常値・特異値分析を行い、適切な値でテストを実施しているか？ 例：1-10 の範囲の項目の場合、文字データ（英字、2 バイト文字)、0 以下、11 以上は異常値になりエラー処理が実行されること	Must	
4	デグレードテスト	処理を追加／変更したことで、処理時間が変更し、他の処理に影響を与えてしまうことはないか？ （実行時間の延長により、他の処理に影響を与えることの防止）	Want	
5	デグレードテスト	処理を追加／変更したことで、デグレードが発生していないことを動作確認しているか？ 例：追加／変更部以外が単体テストの入力・出力値が変更前と変更後で同じ結果になること	Must	
6	設計内容テスト	変数の設計内容とプログラムの内容に食い違いがないことを確認しているか？ （変数の初期化ヌケ／モレ防止）	Want	

著作権