■割り込み処理はなるべく使わない

●割り込み処理とは
割り込み処理はCPUの外の世界の出来事をCPUに知らせる機能の一つです。
特定のアプリケーションが処理している最中でも割り込みが発生すれば
強制的に処理を中断され、割り込み処理が実行されます。

割り込み処理を直接実行させるのではなく、通常の処理で割り込み発生を
知らせるための「割り込み要求フラグ」をループして監視して、
そのイベントに応じた処理を行うこともできます。

●割り込み処理はコンパクトにする
割り込み処理は通常の処理よりも優先的に処理され、その処理が終わると
割り込み前の通常処理に戻ります。
この仕組みからあたかも割り込み処理の方が優先度が高いという錯覚を
覚えますが、システム全体としてみると、単なる仕組みとして優先度という
概念があるだけであり、機能そのものの優先度が高いわけではありません。

例えば、データ通信を行う場合に受信割り込みという機能があります。
データを受信した場合にCPUに通知する機能です。
CPUの都合で一定時間内に受信データレジスタからデータを取り出さないと
データが喪失してしまう、という制限があります。
そのため、優先的に処理しなければならないことは事実ですが、受信した
データを必ずしも急いで使う必要はありません。

1バイト単位の受信で割り込みが発生してもアプリケーションとして
欲しいデータはある一定の約束に従ったひとかたまりのデータ群である場合が
多いことでしょう。
そのひとかたまりのデータを受信してはじめて、アプリケーションとして
必要なデータが揃います。

このような場合、受信割り込み処理でひとかたまりのデータを構築する処理と
アプリケーションとしてひとかたまりのデータを処理することは分けなければ
なりません。
割り込み処理内で全ての処理を記述することも可能ですが、優先度の
低い他の割り込み処理が実行できなくなりますので、このような実装は
避けなければなりません。

●関数コールの落とし穴
割り込み処理とアプリケーション処理を分割して開発すると
C言語では関数単位などの機能に分割されます。

受信割り込み処理でひとかたまりのデータを構築し、そのデータを
アプリケーションで使用するという場合、関数が分割されただけでは
割り込み処理と通常処理としてのアプリケーションを分割したことには
なりません。
単純に割り込み処理からアプリケーション処理を記述した関数を
コールしても機能は実現できてしまうからです。

●機能拡張でトラブルが発生する
組込みシステムの開発は「コピーして派生」させることが多いのですが、
割り込み処理経由でアプリケーションが記述されていても動作している
システムに別の担当者が機能を追加することは良くある話です。

その場合、割り込み処理経由でアプリケーション処理が実行されている
ことはそのままにしておき、アプリケーション部分に機能を追加したと
します。
そして、悪いことに別の優先度の低い割り込み処理も追加しました。
ある程度は正常に動いているように見えていたシステムですが、
たまに処理がおかしくなります。

もしくは受信処理アプリケーション部分で使用していた関数を
後から追加した優先度の低い割り込み処理でも使用するような場合、
排他制御の問題が発生することもあります。

共通の関数がスタックだけを使うような処理ならば問題はありませんが、
グローバルな共通の変数を読み書きするような処理の場合は不正な処理を
引き起こすことになるからです。

つまり、「これまで正常に動作していたから」といって、
「正しい実装がされていた」ことにはなりません。

●割り込み処理はなるべく使わない
受信割り込み処理でひとかたまりのデータを構築できた場合に
アプリケーションに通知する手段としては以下のものがあります。

(1) 直接関数コールする
(2) グローバルな変数をフラグとして使ってアプリケーションで参照する
(3) RTOS使用の場合はタスク間通信機能を使ってイベントを通知する

(1)は動作しますが、設計としては欠陥システムを生む原因となります。
(2)または(3)を選択すべきです。
但し、(2)の場合は参照しているフラグを見終わったというフラグの更新の
ための排他制御問題を解決する必要があります。

したがって、割り込み処理を使わないで済む方法があるのであれば、
割り込み処理としての実装は最初から行わない、という解決方法も
十分に検討すべきでしょう。

ほぼ一定周期で処理を行えば良いという程度のタイマ監視処理などは
割り込み処理ではなく割り込み要求フラグを見て通常処理としての
メインループで処理を行うという実装の方が、問題は少なく出来ます。

以上