

狀態機編程的優點

STM32嵌入式開發 昨天

收錄於話題

#狀態機 3 #編程 23



本文總結下狀態機編程的優點。

提高CPU使用效率

話說我只要見到滿篇都是`delay_ms()`的程序就會頭疼，動輒十幾個ms幾十個ms的軟件延時是對CPU資源的巨大浪費，寶貴的CPU時間都浪費在了NOP指令上。那種為了等待一個管腳電平跳變或者一個串口數據，讓整個程序都不動的情況也讓我非常糾結，如果事件一直不發生電平跳變，你要等到世界末日麼？關於CPU的理解，

如果應用狀態機編程思想，程序只需要用全局變量記錄下工作狀態，就可以轉頭去幹別的工作了，當然忙完那些活兒之後要再看看工作狀態有沒有變化。只要目標事件(定時未到、電平沒跳變、串口數據沒收完)還沒發生，工作狀態就不會改變，程序就一直重複著“查詢—幹別的一查詢—幹別的”這樣的循環，這樣CPU就閒不下來了。

這種處理方法的實質就是在程序等待事件的過程中間隔性地插入一些有意義的工作，好讓CPU不是一直無謂地等待。

邏輯完備性

邏輯完備性是狀態機編程最大的優點。

不知道大家有沒有用C語言寫過計算器的小程序，我很早以前寫過，寫出來一測試，那個慘不忍睹啊！當我規規矩矩的輸入算式的時候，程序可以得到正確的計算結果，但要是故意輸入數字和運算符號的隨意組合，程序總是得出莫名其妙的結果。

後來我試著思維模擬一下程序的工作過程，正確的算式思路清晰，流程順暢，可要碰上了不規矩的式子，走著走著我就暈菜了，那麼多的標誌位，那麼多的變量，變來變去，最後直接分析不下去了。

很久之后我认识了状态机，才恍然明白，当时的程序是有逻辑漏洞的。如果把这个计算器程序当做是一个反应式系统，那么一个数字或者运算符就可以看做一个事件，一个算式就是一组事件组合。对于一个逻辑完备的反应式系统，不管什么样的事件组合，系统都能正确处理事件，而且系统自身的工作状态也一直处在可知可控的状态中。反过来，如果一个系统的逻辑功能不完备，在某些特定事件组合的驱动下，系统就会进入一个不可知不可控的状态，与设计者的意图相悖。

状态机就能解决逻辑完备性的问题。

状态机是一种以系统状态为中心，以事件为变量的设计方法，它专注于各个状态的特点以及状态之间相互转换的关系。状态的转换恰恰是事件引起的，那么在研究某个具体状态的时候，我们自然而然地会考虑任何一个事件对这个状态有什么样的影响。这样，每一个状态中发生的每一个事件都会在我们的考虑之中，也就不会留下逻辑漏洞。

这样说也许大家会觉得太空洞，实践出真知，某天如果你真的要设计一个逻辑复杂的程序，会觉得状态机真香！

程序结构清晰




用状态机写出来的程序的结构是非常清晰的。


程序员最痛苦的事儿莫过于读别人写的代码。关于文档、注释的重要性以及如何去写，推荐看看此文：工程师最讨厌写技术文档？

如果代码不是很规范，而且手里还没有流程图，读代码会让人晕了又晕，只有顺着程序一遍又一遍的看，很多遍之后才能隐约地明白程序大体的工作过程。有流程图会好一点，但是如果程序比较大，流程图也不会画得多详细，很多细节上的过程还是要从代码中理解。

相比之下，用状态机写的程序要好很多，拿一张标准的UML状态转换图，再配上一些简明的文字说明，程序中的各个要素一览无余。程序中有哪些状态，会发生哪些事件，状态机如何响应，响应之后跳转到哪个状态，这些都十分明朗，甚至许多动作细节都能从状态转换图中找到。可以毫不夸张的说，有了UML状态转换图，程序流程图写都不用写。

推荐关注



 嵌入式開發中的C語言 [5] —— 編程思想





ARM與嵌入式

STM32、嵌入式、單片機、PCB、硬件電路、C語言

3篇原創內容



公眾號

喜歡此內容的人還喜歡



STM32嵌入式開發



STM32嵌入式開發



STM32嵌入式開發

