# 丰田栽了的原因，嵌入式软件工程师都该看看

【第一部分】背景简介

前几年闹得沸沸扬扬的丰田刹不住事件最近又有新进展。十月底俄克拉荷马的一次庭审，2007年一辆2005年凯美瑞暴冲（Unintended Acceleration，UA）致一死一伤事件中丰田被判有责。引起广泛关注的是庭审中主要证人Michael Barr的证词让陪审团同意丰田的动力系统软件存在巨大漏洞可能导致此类事件。这是丰田在同类事件中第一次被判有责。庭审过后丰田马上同意支付300万美元进入调解程序。

出于好奇，我漫不经心地下载了Barr的286页证词，却一下子被吸引住了。几天内读完，算是对这次事件进行了一次深入了解。下面就从外行角度总结一下这份证词并尝试以更简单的语言解释里面提到的暴冲原因以及丰田犯下的错误。

Barr的证词下载自他的个人博客Barr Code，但现在该文已经被删除。稍后找地方上传。

Michael Barr是谁？他是一位拥有20年以上行业经验的嵌入式系统工程师。在十八个月中，有12位嵌入式系统专家，包Barr，受原告诉讼团所托，被关在马里兰州一间高度保安的房间内对丰田动力控制系统软件（主要是2005年的凯美瑞）源代码进行深度审查。这房间没有英特网，没有手机信号，他们进出不能携带任何纸张、记录甚至皮带。最后的调查结果被写入一份800页，13章的详细报告。而鉴于保密协议，调查内容一直没有公布，直至俄克拉荷马这次庭审才首度部分公开（报告本身似乎还没公开）。

回到正题。丰田的软件有没有缺陷？根据Barr的调查，答案是有。这其实是废话，任何软件都会有缺陷，关键在于是什么样的缺陷。丰田的软件缺陷分为三类：

非常业余的结构设计。
软件设计的基本要求是模块尽量简单化，因为这样可以一来更易于阅读二来更易于维护。但丰田的工程师显然没有遵循这原则。Barr使用一种工具自动根据代码的可能分支数量评估函数的复杂度，结果是丰田的软件中至少有67条函数复杂度超过50，意味着运行这个函数可能出现超过50种不同的执行结果，属于“非可测”级别。因为为了测试这50个不同的结果，必须准备至少50条不同的测试用例以及相应的文档，在生产环境中一般是不现实的。作为比较，Barr表示他自己的公司严格执行的其中一条规定就是任何代码复杂度不能超过30，否则不合格。而在这67条函数中还有12条复杂度超过100，达到“非可维护”级别，意味着一旦发现缺陷（Bug）也无法修复，因为实在太复杂，修复缺陷的过程中会产生新的缺陷。其中最复杂的一条函数有超过1300行代码，146个可能执行路径——正好用于根据各传感器数值计算节气门开关角度。
如果你不知道什么是节气门，那么这里我稍微解释一下。为了让内燃机运行，有三大要素：燃油、空气和点火时机。空气和燃油的混合物进入气缸，被火花塞在正确的时间点燃推动活塞并最终推动曲轴和车轮前进。在电喷技术发明以后直到2002年以前，三大要素的燃油和点火时间是由电子设备控制，节气门机械连接加速踏板，由司机直接控制。节气门大致是一个连接加速踏板的“空气龙头”——踩下去越多，“龙头”打开得越大，允许越多的空气进入发动机输出更大的动力。2002年以后，丰田引入的“电子油门”让电子系统掌管了最后一个要素：空气。加速踏板不再机械连接节气门，而是连接一些传感器，由行车电脑将这些传感器数值计算成节气门开启角度再由马达控制节气门。我们稍后会再讨论节气门开合。
极复杂的代码带来的是极复杂的功能。下面说一下被称为“厨房洗涤盆”的Task X。这里先解释一下，丰田的软件系统和很多别的软件系统一样，都是由一个统领程序（称之为“操作系统”）和若干小程序（称之为Task）组成。就好比电脑上跑的Windows是统领全局的操作系统，网络浏览器和记事本是跑在操作系统上的小程序。丰田的系统里每个Task都有自己的名字，但这些名字非常敏感，敏感到每次被提及的时候律师都要求法庭内的没有阅读代码权限的人全部清场。为了减少清场次数，Barr将这个最重要的小程序称为Task X。这个Task X有多重要呢？跟厨房里的洗涤盆一样重要。它负责非常多的事情，包括计算节气门开启角度、速度监测和保持、定速巡航监测等等。Task X的不正常运行被认为是暴冲事件的元凶。稍后会再继续讨论

Task X。
还有一些别的匪夷所思的发现。比如丰田的软件包含了超过一万一千个全局变量。如果你不知道什么是全局变量，那么只需要知道软件设计的一般原则是要尽量少使用全局变量，因为有可能带来无法预测的结果。这里的“少”的意思是“尽量接近零”，绝对不会是一万一千个。
不符合软件开发规范。
如同很多行业一样，汽车行业也有自己的规范。更具体一点，由于汽车的危险性质，汽车控制系统被划分为“安全关键性系统（Safety Critical System）”——说白了就是安全性非常重要，弄不好会死人的。为了达到这一特殊要求，汽车相关软件开发人员定期举行会议讨论并发布编程规范，称为MISRA C。该规范的2004年版的感谢列表里还能看到丰田员工的名字，至少让外界认为丰田确实也在遵循这些规范。
真的吗？根据源代码来看，答案是否定的。对此之前的NASA报告也有所提及，丰田辩称他们遵循的不是行业规范，而是丰田内部编程规范。这一规范与行业规范的吻合程度达到50%。但是Barr认为根据他的调查，两个规范之间吻合度小于10%，甚至有若干规范条目相互冲突。后来发现丰田的代码甚至没有遵循丰田内部规范，当然比起别的问题这个已经无关紧要了。
MISRA C拥有超过100条规范，NASA的调查只使用了到其中35条进行校对，发现超过7000处违规代码。Barr使用全部条目，对照结果是丰田的程序拥有超过80000处违规代码。
这些数字说明了什么？根据统计，违规数量可以用于预测代码中暗藏的缺陷（Bug）数量。在之前提到的汽车相关软件开发人员会议中，有人就这一主题发表过专题演讲，提出每30处违规代码可能包含一个重大缺陷和十个轻微缺陷。讽刺的是这人是丰田员工。
特别需要指出MISRA C其中一个规则的内容是不得使用递归。
如果你不知道什么是递归，那么这里我稍微解释一下。递归是一种计算方法。但一般计算方法要么是自己算，要么询问别的计算模块索要结果。而递归则是通过问一层层问自己的方法完成计算。好处是代码简单，坏处是计算层数不固定。可能会2层就出结果了，也可能会是10000层，在设计程序的时候无从得知。
软件计算需要消耗存储器。越繁琐、越长的计算自然需要占用越多的存储器。递归的问题在于其嵌套层数无法预测，从而导致可能消耗的存储器容量无法控制。稍后会再讨论存储器。
对关键变量缺乏保护。

什么是变量？变量就是存在一段存储器的0和1的集合。这些变量既可以是一些函数的处理结果，也可以是另一些函数的处理原材料。比方说前面提到有一条程序专门计算节气门开合角度，比如说是20度，这个20就是一个变量，存在存储器的一个指定位置。另一个程序专门负责开合节气门，它知道去那个指定位置读取这个20，然后把节气门开启20度。

什么是保护？嵌入式系统，或者任何系统，都会在一定条件下发生硬件或者软件错误。客观上这是无法避免的。而且汽车作为一个时常在震动、发热、位移的系统，它的内部环境不能说不恶劣，发生硬件错误的可能性甚至更高。什么样的硬件错误呢？别忘了变量都是0和1的组合，这些0和1由存储器上的高低电平代表。由于某些不可抗原因，一个电平从高变成低，或者反过来，那么这个变量就被更改了。这被称为“位反转（Bit Flip）”。为了对抗这样的事情发生，需要对变量进行保护。保护的方法一般是镜像法。简单来说就是在两个不同的地方写入同一个变量，读取的时候两边都读，比较是不是一致。如果不一致，那么可以得知这个变量已经不可靠，需要进行容错处理。

丰田的程序总体上对其上万个变量进行了有效保护，但问题出在操作系统上。前面提到丰田的软件本质上分为操作系统和Task。Task是由丰田自己开发，但是操作系统则是由芯片供应商提供，固化在芯片里的。丰田在这里的过失是没有对供应商提供的代码进行深度审核，拿到什么用什么。

另一个保护措施是错误校验码（Error Detective and Correction Codes，EDAC）。这是一个硬件层面的数据保护措施。简而言之就是给内存中每一个字节（8比特）后面物理地增加几比特校验码。这样万一变量出错了，可以通过校验码得知，甚至可以通过校验码修复一些轻微错误。这个措施十分简单有效，但是在2005年款凯美瑞的系统中完全没有使用，丰田却告诉NASA他们用了。而在2008年款凯美瑞中使用了3比特长的EDAC。Barr认为是为了节省成本，否则应该使用5比特长。

还有值得一提的是，汽车相关的软件行业有那么几家操作系统供应商，早已形成了一套成熟标准称为OSEK。各供应商开发的符合OSEK认证的操作系统至少都能达到一定的质量。但丰田选用的操作系统却没有通过认证，让人不解。

现在我们知道丰田在编写软件的时候至少有三种缺陷。那么重点问题：丰田的这些软件缺陷是否会导致车辆暴冲？根据Barr的调查，答案是有可能。暴

冲的起因需要结合上述三种缺陷来说明。

汽车正常运行需要倚靠若干程序（这里叫Task）的同时运作。Task有很多，CPU只有一块，在任何时刻只能处理一个Task，怎么办呢？这需要操作系统的统筹规划，合理分配CPU的任务，让每个Task都能按时执行。如果出现某种意外，让某个Task突然不执行了，那么就称为这个Task“死亡”。Task死了，自然不能执行它的任务。根据Barr的测试，在某些特定情况下，Task X的死亡可以导致节气门敞开——因为Task X的其中一个任务就是根据司机的操作计算节气门开合角度，它死了也就没法重新计算这个角度，即使司机把脚挪开加速踏板，节气门也无法关闭。此为暴冲的直接原因。更糟糕的是，节气门的开合角度这个数值，被Task X算出来以后保存在一个变量中。这个特定的变量正好没有被保护（缺陷3）。意味着万一Task X死亡并且停止计算，这个数值有可能因为不可抗原因被改变，而程序无从得知。

那么Task X为何会死亡呢？一般是因为内存出错。这个出错可能是一个小小的位反转，也可能是内存里的数值被别的程序错误覆盖。同一系统内同时运行了若干程序，这些程序需要共享一块内存，内存内部必然要被划分成若干块。比如第一块给程序1，第二块给程序2，等等。如果程序1因为某些原因（比如Bug）写到第二块内存上去，就会导致程序2读取了错误的信息。这就是所谓的内存出错。丰田的系统里，正好有这么两块相邻的内存块。第一块被称为“堆栈（Stack）”，这是所有Task存储它们运行状态的地方，大小为4KB。与之相邻的地方储存了操作系统进行任务分配的记录。那么可以想象，如果很多Task给堆栈里写入太多东西，超过4KB，那么就会错误地写入与之相邻的任务分配表。这种错误被称为“堆栈溢出”。操作系统拿到了错误的任务分配表，就会错误地分配任务，造成某些Task多执行几次，某些Task少执行几次，某些Task甚至就再也不执行——死了！必须指出的是，程序死亡并不罕见，甚至可以认为是正常现象。稍后解释处理方法。

那么堆栈为什么会溢出呢？显然是因为要写入的数据超过了堆栈的容量。在设计程序的时候要计算最坏的情况并且允许冗余。即使作出了正确的设计，这种错误也相对常见，所以NASA在他们的调查中重点排查堆栈溢出的可能性。于是NASA问丰田，丰田的回复是最坏的情况下4KB堆栈只写入了41%的数据，换句话说发生溢出的可能性非常低。NASA直接取信了这个数字并没有再深入调查。但Barr他们发现丰田的回答有严重低估，实际上最坏的情

况会达到94%，而且还不算递归。丰田在代码中使用了递归（缺陷2）。因而实际数字有可能超过94%而且无法预计上限，因为递归计算的嵌套层数是无法预测的。所以实际情况下堆栈溢出的可能性相当可观。一旦溢出，相邻的任务分配表不可避免就会遭到破坏。此为暴冲的根本原因其中之一。之所以说“其中之一”，是因为堆栈溢出仅仅是损坏任务分配表的其中一个原因，别的还有许多可能性并没有被Barr在法庭上深入解释。而且任务分配表的损坏也仅仅是导致Task死亡的原因之一。
顺便提一句，2005年的凯美瑞的这部分供应商是电装，没有搭载堆栈实时监测功能——溢出了也不知道。同年的卡罗拉却搭载了，因为供应商是通用。

到这里我小结一下，串链子。左边是原因，右边是后果。
堆栈溢出→（可能导致）→任务分配表被改写→（可能导致）→Task X死亡→（可能导致）→节气门敞开→（导致）→汽车暴冲

必须指出的是，这条链子从最左边一直到Task X死亡，都还算是嵌入式系统的常见故障。虽然程序代码写得不好也许导致更容易出错，客观上丰田并没有特别大的过错。只要处理得当，这些故障都不会导致暴冲。

到此为止还只是前奏而已，接下来我们来看看丰田到底做错了什么。

【第二部分】丰田之罪

上面反复提到，嵌入式系统中内存出错或者程序死亡其实是一种正常现象——至少从Barr的证词可以得出这个结论——现在连我们都知道了，嵌入式工程师肯定比我们更清楚才对。确实，为了使系统正常运行不被错误干扰，一般的做法是设置若干层防护措施（Failsafe），让运行中出现的错误无法轻易突破，得到妥善处理。丰田的工程师自然也懂得这一点。很可惜，他们搞砸了。

上面那条链子应该修改成这样：
（防护措施0）→堆栈溢出→（防护措施1）→（可能导致）→任务分配表被改写→（防护措施2）→（可能导致）→Task X死亡→（防护措施3）→（可

能导致）→节气门敞开→（防护措施4）→（导致）→汽车暴冲

可以看到，防护措施不可谓不多。只要处理得当，这链条应该是走不通的。现在让我们从左到右看一个小小的内存错误是如何一层层突破防护最终导致汽车暴冲的。

首先防护措施0。这个其实上面提到了，因为设计缺陷低估了最大占用的存储容量，并且不符合规范地使用了递归，最终可能导致堆栈溢出。

然后到防护措施1。上面也提到了，任务分配表紧邻堆栈。作为外行我都觉得这是个十分危险的设计——既然堆栈这么容易溢出，好歹应该将任务分配表放远一点啊。当然我是外行，可能实际上比想象中复杂很多。这段Barr的证词中并未特别提到，属于我的个人理解。

防护措施2。从这里开始丰田的错误越发严重。任务表被改写导致某些Task运行异常，在软件层应该有若干检测措施，比方说用特殊的监视Task来监视别的Task是否正常。但丰田是怎么做的呢？还记得上面的“厨房洗涤盆”Task X吗？它是如此复杂（缺陷1），除了控制汽车运行的任务之外竟然还兼任大部分的监视任务，比如生成DTC。
DTC（diagnostic trouble codes），是汽车电脑系统会根据情况生成的错误代码。有的车主可能会遇到汽车某报警灯常亮，修车师傅拿个仪器插在行车电脑上得出一个代码，再查表就知道哪个元件坏了——这就是DTC。除了用于修车，DTC还被用于检测行车电脑和各传感器的异常状态。
可以想象，这个既是运动员又是裁判的Task X一旦死亡，软件层的检测措施大部分就失效了。

防护措施3。在这里丰田的错误开始到达顶峰。即使设置正确无误，上面提到的监视Task也只不过是另一个Task而已，与它的监视对象算是平级——监视Task自己同样有可能出现故障。嵌入式系统的一般做法是在所有程序之上再设置一道屏障，被称为“看门狗（Watchdog）”。所谓看门狗，是一个优先级很高的倒计时程序。别的程序需要在运行的时候特意去重置一下这个计时器让它重新开始倒计时，这个动作被称为“喂狗”。如果因为程序出问题太长时间不喂狗，倒计时完成，看门狗知道什么地方卡住了，马上采取措施，比如重启整个系统。重启系统听起来似乎很严重，实际上却是一件相

当普通的事情。嵌入式系统的重启非常快，时速100公里的汽车中动力系统可以在半米之内完成重启——车上的人根本觉察不到。

通过阅读代码和拟真实验，Barr惊讶地发现上述嵌入式系统的常识性做法竟然在丰田软件系统内不存在！丰田的软件确实有一只看门狗，但它竟然不是用于监视Task异常，而是用于防止CPU过载。首先这个做法不能说后无来者至少算是前无古人。还记得上面提到的800页13章的报告吗？目瞪口呆的Barr将丰田看门狗的分析结果写入了报告的第一章，因为他实在太震惊。其次，丰田看门狗的防止CPU过载功能也相当蹩脚，在拟真测试发现即使它正常工作，还是会允许CPU过载时间长达1.5秒——时速100公里的车能跑40米以上。CPU一旦过载，就会导致所有的Task进入一种“假死”状态，无法处理信息，这时司机无法控制汽车动力，十分危险

另外，丰田的工程师还犯了一个嵌入式课堂上被反复提到的经典错误：使用硬件时钟中断喂狗。硬件中断拥有非常高的优先级，即使Task卡住（比如出现死循环）也不能阻止硬件中断——可想而知这样一来看门狗就等于完全白瞎了。

这里也提一句，同年的普锐斯却令人意外地搭载了一只运作正常的看门狗，反而让人摸不着头脑。

还没完。这一层防护是嵌入式系统的关键阵地。前面都是电子系统，后面马上进入机械运作，足以造成灾难了。所以仅仅拥有软件级别的防护还不足够，丰田的做法是在主CPU之外单独设置了一块监视芯片，从硬件级别对系统的运作进行监视。监视芯片有两个任务。第一，它运行一种叫做系统卫士（System Guard）的程序，原理上来说是专门用于防止暴冲。主CPU和监视芯片上都会运行系统卫士，可是研究发现Task X一旦死亡，这些系统卫士统统都不起作用了。第二，它运行一个被称为“刹车回声检查（Brake Echo Check）”的程序。这个程序从代码上来看似乎可以检测出Task X的死亡，并且采取相应措施：关闭节气门。听起来像是好消息，但是同样有问题：首先这个程序不太可靠，即使正常运行，理论上也有失效的可能。最关键的是该程序不会自动运行，需要司机先对刹车踏板有“动作”才会触发。注意这里我特意没用“踩刹车”这个词，因为根据分析“触发动作”十分令人困惑。它分两种情况：如果Task X死亡的那一刻司机的脚不在刹车踏板上，那么触发动作是踩刹车。还算可以理解。另一种情况，如果Task X死亡那一刻司机的脚踩在刹车踏板上，那么触发动作是完全释放刹车踏板。没错，察觉车子在不正常加速的司机需要停止踩刹车才能让控制系统关闭节气门！这种违背人类认知的行为应该不是丰田工程师特意设计的。如果是，他

们到底在想什么啊?
到此为止，上面提到的都算是“战术层面”的错误，都是“小错”。在讲解这块监视芯片的时候，可以发现丰田犯下最严重的“战略层面”错误——基础设计。Barr认为，如果基础设计正确，上述那些小错都完全不会导致汽车暴冲——不管代码写得多业余，不管内存错误多严重，不管Task死得多频繁，统统不会致命。让我们回到2002年以前，没有电子油门的时候。那时候的拉线油门是由油门踏板机械连接的。当驾驶员的右脚踩下刹车，他的右脚必然不在油门踏板上，节气门自然而然地被关闭。这个动作如此自然，甚至算不上安全措施，仅仅因为每人只有一只右脚，不可能同时踩油门和刹车。当丰田设计电子油门的时候，只要稍微有点常识，都应该从设计阶段就将这一“自然而然”发生的动作考虑进去。但是很显然，他们没这么做。监视芯片上运行的代码是用汇编语言（一种更加接近机器执行代码，远离人类语言，更加难懂的编程语言）编写的，运行层次比主CPU的C语言更低。Barr认为如果设计得当，现有的监视芯片完全有能力胜任上述功能，需要的仅仅是几百行代码，别的什么都不用更改——不会提高任何生产成本。很遗憾，他们没有做到。

防护措施4。现在已经脱离电子系统，节气门已经敞开，发动机全速运转，需要使用机械运作来组织机械运作了。如何让向前冲的车子停下来?不开车的人都知道，刹车！现代汽车都装备了刹车助力，助力来自于发动机运转的时候产生的负压。我们知道发动机需要吸入空气，吸入体积等于排气量乘以转速。节气门又是用来阻挡空气的，那么节气门关闭而发动机转速相对高的时候（比如高速丢油门），发动机的实际空气吸入量比它能吸入的体积要少，那么从节气门到气缸进气口之间会形成明显低气压（所谓负压，比大气压力小）。刹车助力就是利用了这个负压推动气鼓产生更大的推力带动刹车片抓紧刹车盘。但是如果节气门敞开让空气随便进来，低气压就不存在了，这时刹车助力大大减弱，刹车效率也大大降低。这就是为什么暴冲事件当事人都说全油门的时候根本刹不住的重要原因。这个现象称为“真空损失（Vacuum Loss）”，存在于所有自然吸气的汽油发动机汽车（柴油和增压发动机没影响），不算丰田的错。但丰田迟迟不搭载刹车优先系统（Brake Override System）允许刹车的同时敞开节气门，毫无疑问是这个现象的帮凶。
所谓刹车优先系统，指的是保证同时踩下刹车油门两个踏板的时候无条件关

闭节气门的功能。这么做很显然主要是为了降低发动机输出，同时也保证刹车助力。丰田在2010年的凯美瑞上终于搭载了刹车优先系统，但是别高兴得太早。根据Barr的调查，丰田竟然将如此重要的修改“理所当然”地写入了他们的“厨房洗涤盆”——Task X。我只能“哑然失笑，扼腕叹息”。

好了，到此为止都还是Barr的一面之词，而且大部分都是在那间守卫森严的房间内进行拟真测试得出的“理论结果”。那么实车测试情况如何呢？丰田对Barr的证词如何反驳呢？

先说说实车测试。为了证明理论，他们把2008年和2005年的凯美瑞放在马力机上，固定车身架起前轮模拟车辆运行情况。他们的做法是首先让车子运行在时速68英里（110公里），启动巡航，脚离开油门踏板。然后暂停巡航，速度开始下降。下降到一定程度恢复巡航，速度开始上升。在到达68英里的设定时速以前，他们用一台连接行车电脑的笔记本“注入”错误。所谓注入错误，就是人为地翻转一个特定字节——将0改成1，或者反过来——模拟内存损坏。结果完全符合理论，时速超过68英里也不停止加速，直至时速90英里（145公里），测试人员踩下刹车。大约1秒以后节气门被关闭，Barr认为这是上述“刹车回声检查”的功劳。

实车测试证明了Barr的理论，却并不是全无破绽。丰田辩护律师就两点提出质疑：
实车测试使用人工注入错误的方法，并不能证明现实中这种错误就一定会发生。
对此Barr的回答是测试的局限性。因为测试时间、样本有限，而待测试的样本空间无穷大。如果要等待那个特定的错误自然出现，可能需要成百万上亿小时的不间断测试，显然是不现实的。更何况从科学上而言，没有办法对这个错误证伪——就好比无法证明宇宙里没有外星人，最多只能证明火星上找不到而已。但是这个测试足以证明一个小小的位翻转确实可以突破重重障碍最终导致暴冲，足以证明丰田的软件存在不能容忍的隐患。
Bookout女士（本案原告）声称，在她驾驶汽车离开高速的时候发现不受控加速，她拼命反复踩下刹车并且拉起手刹，现场留下了刹车痕迹。但并没有

迹象表明发动机动力中断——换言之“刹车回声检测”没起作用。暗指Barr的理论站不住。
对此Barr的回答是首先尽管在实车测试中每次都生效，但代码分析表明“刹车回声检查”这一功能在理论上靠不住。其次这一功能的另外一个触发动作是要让脚完完全全离开刹车踏板。试想车子正在不受控地往前冲，任何人都会不由自主地踩刹车，让人完全不踩刹车踏板根本就是违背认知的。
Bookout女士即使如同她所称反复踩刹车，很可能只是一直将脚放在踏板上往复运动，从未完全挪开。Barr还引用一位丰田自己的软件专家的证词。该专家承认，如果发生暴冲的时刻脚正好接触到刹车踏板，并且之后一直没挪开，那么汽车的暴冲距离“取决于还剩多少汽油”。

最后顺带说一下那份800页，13章的详细报告完成后，Barr将其提交给了丰田的软件部门，等待他们的反驳。最终结果是“非常少（Very little）”，13章中的11章，包括堆栈溢出的部分、代码混乱的部分、违反开发规范的部分、Task X过于臃肿甚至兼任节气门控制和防护措施的部分、看门狗形同虚设的部分、无EDAC的部分、重要变量缺乏保护的部分、使用了非标准化操作系统的部分，全部没受到任何形式的反驳。

## 【第三部分】后记

写到这里，谈谈人们比较关心的几点。当然还是外行眼光。
NASA / NHTSA怎么没发现这些问题?
NHTSA本身不具备检验电子系统的能力，于是委托NASA。NASA检验的是整个电控系统，包括电控传动部分，范围比较宽，只有很少一部分资源被用于检验软件系统，也没有投入足够的人力进行逐行代码审阅。更何况在很多关键问题，比如之前提到的EDAC的使用、堆栈的设计，NASA都直接采信丰田的回复，最终被证明不正确。甚至NASA从来都没拿到过监视芯片的源代码，丰田的说法是“他们没说要啊”。NASA报告虽然没能找到软件系统导致暴冲的确切原因，但没有否定其可能性。与之相比Barr的团队全部都是嵌入式系统专家，投入上千小时，深入程度甚至超过丰田自身对这个系统的理解（比如丰田没看过供应商的OS代码，Barr看了）。
能否100%确定本案就是由软件错误造成的?

不能。并没有直接证据。诉讼团认为，软件错误造成该事故的可能性比软件错误没造成该事故的可能性大（原文：more likely than not）。
这里再提一句，2005年款的凯美瑞没有搭载行车数据记录器（俗称“黑盒子”），后来的车款渐渐开始搭载。但是Barr发现这个记录功能并不可靠，完全有可能记录错误信息。比如司机踩刹车了可能会被记录成没踩。
本案的意义
之前虽然丰田赔了不少钱，但是从未在涉及人身伤害的案件上承责。所以本案意义在于开先例。美国的法律又特别注重先例，今后丰田的法务部门要头疼了。
本案提到的有缺陷软件涉及了哪些车型?
全部是美国的车型。Barr的调查重点是2005年款凯美瑞，另外审阅过的包括雷克萨斯ES、Tacoma、卡罗拉和普锐斯等等，生产年份大致在2002年（电子油门元年）与2010年之间。其中凯美瑞、雷克萨斯ES和Tacoma使用的软件系统大致接近（原文：Substantially similar）。另外根据统计，汽车暴冲投诉中与2004年款以后的凯美瑞有关的案件数量激增400%。

最后的最后，放上本案关键证人Michael Barr的独家访谈：

我：这么看来似乎手动档汽车更安全，你怎么认为?
Barr：很多专家都这么认为，离合器至少可以物理断开动力系统。但是我翻阅卷宗，发现其中有个案例是受害者开手动档凯美瑞载着家人，突然巡航系统失灵，无法取消。他踩下离合，同时试图躲避前方慢速车辆结果失控冲出路面造成单车事故。幸运的是没死人。

我：现在我们都知道丰田的软件很糟糕。可是你对整个汽车行业的软件水平有什么看法？丰田的软件在同行内属于什么水平?
Barr：我没有接触过丰田以外的软件代码。但是请注意，这次发现的最严重问题是丰田在设计源头上没有考虑安全，软件质量反倒没有那么重要。只要一个安全为先的设计，比如刹车和关闭节气门的可靠互动、防止节气门开启降低刹车效率的机制等等，不管软件有多差劲也不会造成致命结果。只是我

真不知道软件还能怎么差。

我：终极问题，你开什么车?
Barr：我不开丰田。接触该案以来我没买过新车。老实说我现在非常害怕买新车。我倒是问过一个与车企斗争了三十多年的职业状棍同样的问题，他开宝马。

【全文完】