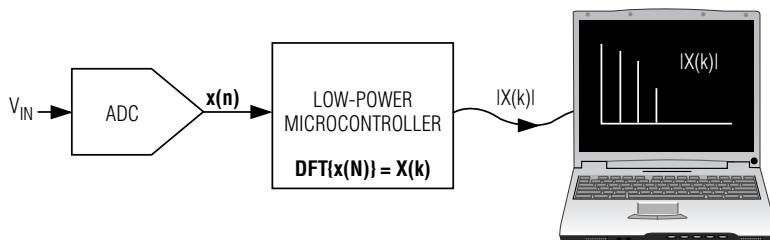
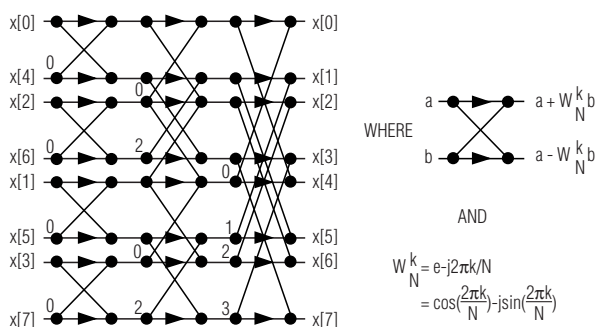


新闻简报		2
探讨文章	保护您的研发成果——双向认证及软件功能保护	3
	利用低功耗微控制器开发FFT应用	9
设计实例	精密监视负电源电流的电路	14



(1)



(2)

图1. 利用FFT应用计算输入电压的频谱。(见内文, 第13页.)

图2. 利用蝶型运算实现N=8的FFT。(见内文, 第14页.)

News Brief

Maxim公布2006财年第二季度创纪录的收入和10%的环比季度订货涨幅

Maxim Integrated Products, Inc. (MXIM) 公布截至2005年12月24日的财务第二季度净收入达到创纪录的445.9百万美元，比2006财年第一季度公布的424.4百万美元增长了5.1%。如果不考虑基于股票的补偿开支，本季度的pro forma净利润为140.0百万美元，或摊薄后每股收益0.42美元；考虑基于股票的补偿后的GAAP净利润为112.6百万美元，或摊薄后每股收益0.33美元。相比2006财年第一季度，对应的pro forma净利润为133.2百万美元，或摊薄后每股收益0.39美元；考虑基于股票的补偿后的GAAP净利润为105.4百万美元，或摊薄后每股收益0.31美元。

第二季度的订货总额大约为506百万美元，比第一季度的459百万美元增加了10%。本季度收到的流通订单总额约为230百万美元，相比前一季度的208百万美元增长了10%。所有地区的订货均有增长。截至第二季度末，未来12个月内可发货的未交货订单约为370百万美元，其中约有329百万美元要求在2006财年第三季度发货。相比第一季度末，公司未来12个月内可发货的未交货订单约为330百万美元，其中约有296百万美元要求在2006财年第二季度发货。

第二季度的pro forma研发费用(未考虑基于股票的补偿支出)为92.6百万美元，或净收入的20.8%，包括了24.3百万美元基于股票的补偿后的GAAP研发费用为116.9百万美元，或净收入的26.2%。第二季度的pro forma销售、日常行政开支(未考虑基于股票的补偿支出)为23.8百万美元，或净收入的5.3%，而包括了7.2百万美元基于股票的补偿后的GAAP销售、日常行政开支为31.1百万美元，或净收入的7.0%。

本季度，公司以334.6百万美元回购了约9.2百万股自己的普通股，支付了40.0百万美元的股息，并采购了37.4百万美元的固定设备。由于净收入的增加，第二季度应收账款增加8.1百万美元，至221.0百万美元。本季度的pro forma库存(未考虑基于股票的补偿支出)增加至186.5百万美元。第二季度包括了11.3百万美元基于股票的补偿后的GAAP库存增加至197.8百万美元。

公司期望实施一项计划，通过该计划，雇员(不包括高级职员)可以将其所持有的、行使价格在35美元以上的已授权股票期权置换为有限权限的股票单位(RSU)，置换比率由Black-Scholes模型导出，并在未来12个月内逐季授予。有些情况下，雇员也可以选择将其符合条件的股票期权以一个特定的置换率转换为RSU，该置换比率高于Black-Scholes模型给出的置换比率，但这些RSU将在未来18个月内逐季授予。该计划的细则不久将提交证券交易委员会(SEC)备案，并通知那些符合置换条件的相关人员。该计划的实施是为了增强我们雇员的凝聚力，并使其利益与我们投资人的利益保持一致。这项置换计划的实施会减少Maxim雇员股票期权的数量，并向那些选择置换的雇员提供对于Maxim股票的所有权。该置换计划涵盖了总量大约20万股的已授权期权，如果对所有这些期权履行该计划，将会发放大约4百万RSU。Maxim始终认为，股票形式的补偿是激励员工并使其目标与投资人利益相一致的最有效的激励机制。

持有符合该计划置换条件的股票期权的雇员应仔细阅读公司关于置换股票期权为RSU的提案，以及公司关于传递和股权收购材料的信函，因为其含有重要信息，包括但不限于有关该计划的各种术语和条件。持有符合该计划置换条件的股票期权的雇员不久会通过邮寄方式收到一份关于置换股票期权为RSU的公司提案，以及公司关于传递和股权收购材料的信函的拷贝，并且一经SEC备案，即可从SEC网站www.sec.gov免费获得。

Gifford先生评论道：“第二季度的业绩是我们远期战略的一个正面反映，我们的远期战略是继续服务于诸多模拟市场领域，并赢取更多的市场份额。我们相信模拟工业的前景是振奋人心的，并且我们已占据了很好的位置来迎接高盈利增长。”

Gifford先生总结道：“公司董事会已宣布2006财年第三季度的现金股息为0.125美元每股。将于2006年2月28日向2006年2月13日登记的股东付讫。”

完整的Q206新闻公报，包括安全港信息，请访问：www.maxim-ic.com.cn/NewsBrief

Maxim标志是Maxim Integrated Products, Inc.的注册商标。Dallas Semiconductor标志是Dallas Semiconductor Corp.的注册商标。
©2006 Maxim Integrated Products, Inc. All rights reserved.

保护您的研发成果——双向认证及软件功能保护

在冒名顶替、伪造证件行为猖獗的年代，保证正确的身份识别至关重要。这不仅对个人如此，对电子产品也是如此。系统供应商需要在外有黑客攻击这样的“外患”，内有克隆硬件这样的“内忧”的环境中保护其产品的安全性。实现这些安全需求的关键是认证。

什么是认证？

认证是指两个或多个实体之间建立身份认可的过程。单向认证情况下，一方需向另一方证明其身份的合法性。对于双向认证，双方需要彼此向对方证明自己的身份。最常用的认证方法是利用口令实现。使用口令的主要问题是应用中口令是暴露的，极易被探测。

我们先来回顾一下加密的历史应用，1883年弗兰德斯语言学家Auguste Kerckhoffs发表了一篇关于军事加密的文章，震惊了整个世界。Kerckhoffs讲道，安全不应依靠隐匿性(例如非公开的保密算法)，而应依靠算法及其密钥的力量。如果安全受到破坏，Kerckhoffs认为，只需替换密钥，而不是替换整个系统。

基于密钥的认证过程如图1所示：密钥(私密)和需要认证的数据(“信息”)作为输入，来计算信息认证码，

即MAC。MAC然后附加到信息上。信息接收方进行相同的运算，将MAC计算结果与随信息一起收到的MAC比较。如果二者相同，则信息是合法的。

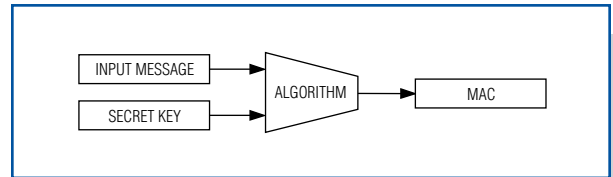


图1. 该MAC计算模型演示了基于密钥的认证过程。

但是，这种基本MAC计算模型也有一个弱点。非法者如果截取到信息，可随后回放此信息，以假冒合法身份。为克服这种固有的MAC弱点和证明MAC发送方的合法身份，接收方可产生一个随机数，作为质询码回送给发送方。MAC发送方必须根据密钥、信息和质询码重新计算新的MAC，并返回给接收方。如果对任何质询码发送方都可产生有效的MAC，则可以确信发送方是知道密钥的，其身份是合法的(图2)。该过程就是质询-响应认证。

在加密学中，由信息产生固定长度MAC的算法称为“单向”散列函数。单向表示从固定长度MAC输出推演出较长的原始信息极为困难。相反，通过加密，加密的信息与原始信息是成正比的。

1-Wire是Dallas Semiconductor Corp.的注册商标。

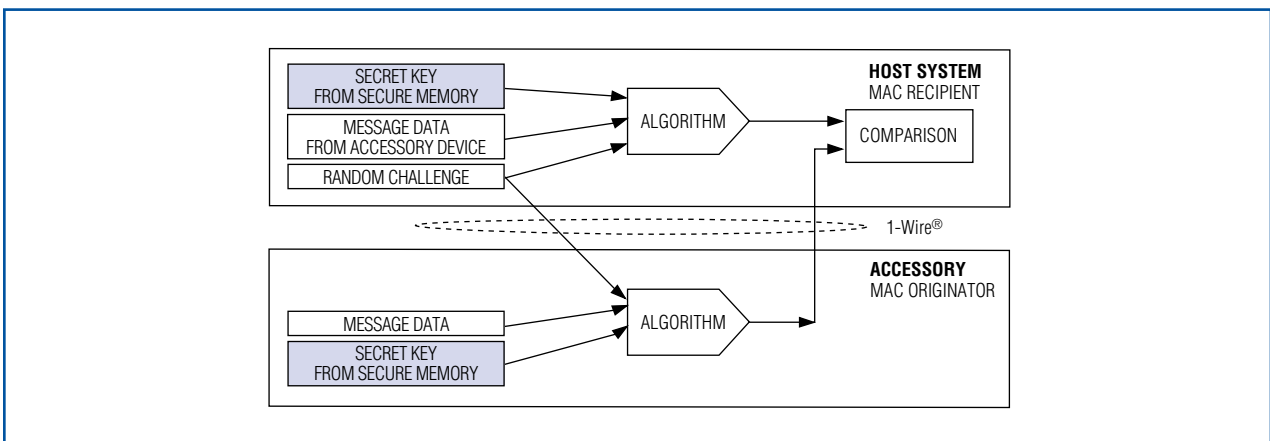


图2. MAC模型的弱点——会误认为截取的信息是合法的，质询-响应认证过程可解决这一问题。

SHA-1是经过深入研究和国际认可的单向散列算法，由National Institute of Standards and Technology (NIST) (www.itl.nist.gov/fipspubs/fip180-1.htm)开发。SHA-1已经发展成为国际标准ISO/IEC 10118-3:2004，算法的数学基础是公开的，并可从NIST网站获取。SHA-1算法的主要特点包括以下几点：1)不可逆性—从计算角度讲，不可能从MAC推演出输入信息；2)抗冲突性—对于特定MAC，找到多于一种输入信息是不现实的；3)高雪崩效应—输入的任何变化都会使MAC结果产生巨大的变化。基于这些原因以及对该算法的国际性研究，Maxim/Dallas Semiconductor选择SHA-1作为其安全存储器的质询-响应认证算法。

低成本安全认证—功能实现

DS2432 EEPROM内置SHA-1引擎，借助1-Wire接口，可以方便地加入到任何带有数字处理能力的电路中，例如带微控制器(μ C)的电路。最简单的情况下，仅仅需要一个空闲I/O引脚以及一个上拉电阻即可构成1-Wire接口，如图3所示。如果板上的计算能力或者剩余的程序存储空间不足以完成SHA-1 MAC计算，设计者可以采用DS2460 SHA-1协处理器，或将计算任务转交给系统或网络中最近的主机。协处理器还

有另一个好处，可将系统密钥存放在安全存储器中，而不必存放在程序代码内。

嵌入式硬件/软件授权管理

参考设计需要授权，并可能由第三方进行生产，需要防范非法使用程序代码。考虑到收入原因，还有必要跟踪和确认参考设计的使用次数。DS2432经过预先编程(在供给第三方制造商之前先装入密钥和存储器设定值)，可以轻松地满足这些需求和提供更多功能。上电自检时，参考设计(图4)通过DS2432执行认证过程。只有具备有效密钥的DS2432才能成功地返回有效MAC。如果检测到无效MAC，处理器将采取相应的特定操作。这种方法还带来另外一个好处，即可以通过DS2432安全存储器的设定值有选择地授权和使能参考设计的功能。(有关此概念的更多信息，见软件功能管理一节。)

具有64位有效密钥的DS2432可通过以下两种安全方法提供给被授权者或第三方制造商：1)由参考设计授权公司预先编程；或2)由Maxim/Dallas Semiconductor根据授权公司的输入信息预先编程并供货给第三方制造商。无论采用何种方法，供给被授权者或制造商的器件数量都是已知的，可据此收取授权许可费用。

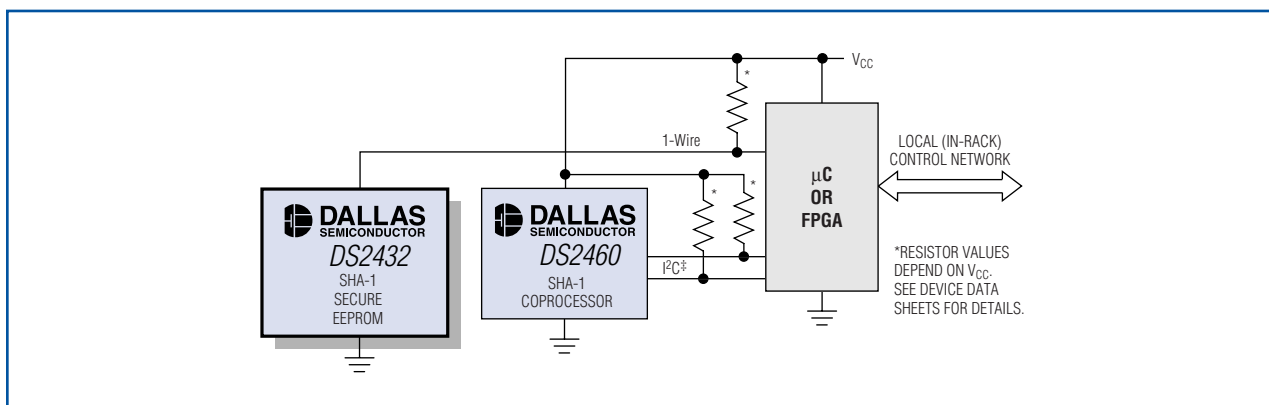


图3. DS2432 EEPROM在典型电路板环境下通过一个空闲I/O引脚和一个上拉电阻进行配置。

‡购买Maxim Integrated Products, Inc.或其从属授权关联公司的I²C产品，即得到了Philips I²C的专利许可，将这些产品用于符合Philips定义的I²C标准规范的系统。

验证硬件的合法性

验证硬件的合法性时，需要考虑两种情况(图5)：

- 1) 克隆电路板完全拷贝固件/FPGA配置信息；以及
- 2) 克隆系统主机。

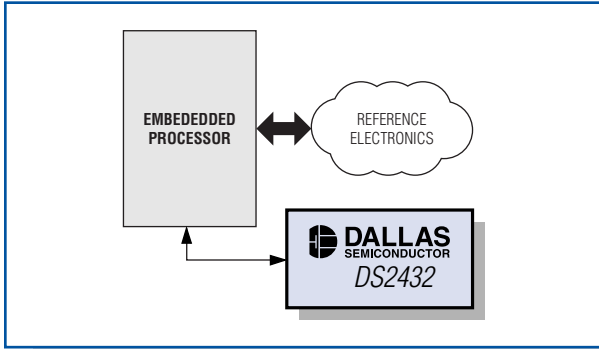


图4. 使用DS2432认证参考设计。

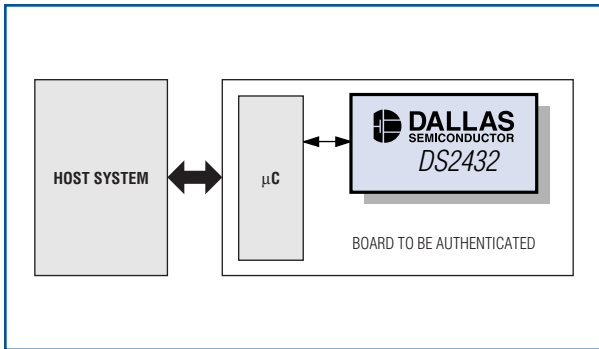


图5. 硬件认证实例，克隆电路板完全拷贝固件/FPGA配置信息或克隆系统主机。

第一种情况下，固件/FPGA试图认证克隆的电路板。克隆制造商要向用户EEPROM写入数据，必须向DS2432装入密钥。尽管这使数据看似正确，而密钥在系统中却是无效的。由于改动固件/FPGA极为复杂，为保持和主机兼容，必须精确拷贝原始固件/配置信息。如果在上电阶段电路板执行DS2432质询-响应认证过程，则DS2432产生的MAC与微控制器/FPGA计算出的MAC不同。MAC不匹配充分证明电路板是不合法的。系统与电路板之间执行质询/响应过程可以检测到该失配现象，并据此采取相应的特定操作。

第二种情况下，电路板试图认证主机系统。电路板通过以下步骤验证主机的身份：1) 产生质询码，由DS2432计算质询-响应认证码MAC；2) 向网络主机发送计算MAC的相同输入数据(当然不包括密钥)，主机根据这些数据和自己掌握的密钥计算并返回质询-响应认证码MAC。如果二者产生的MAC相同，则电路板断定主机是合法的。

软件功能管理

电子产品涵盖手持式产品到安装于机架上的单元。单元的尺寸越大，开发的成本越昂贵。为使成本得到有效控制，利用一些较小的子系统(电路板)来构建大型系统是非常有益的。通常，应用中并不需要子系统的所有功能。最具成本效益的作法不是去除这些功能，而是保持电路板不变，仅在控制软件中禁用某些功能。但这种方法又会产生新的问题：如果聪明的客户需要一些功能完备的系统，他可以只购买一套功能完备的单元和一些功能较少的单元。通过软件拷贝，功能较少的单元就可以提供完备的功能，而价格却更低，因此欺骗了系统供应商。

每个子系统电路板上的DS2432可以保护系统供应商免受这类欺骗。除了进行质询-响应认证外，DS2432还可以在其用户EEPROM内存存储独立的配置信息。配置数据可防止非法篡改，系统供应商具有完全的控制权，这一点将在数据安全一节进一步阐述。配置信息可以存储为位图形式或代码字形式，完全由系统设计者决定。根据实际需要，应尽可能简单地设置配置信息。由于DS2432提供方便的1-Wire接口，设计者只需增加一个晶体管和一个探测点，如图6所示。可以在电路板其它部分不上电的情况下，通过探测点向DS2432写入配置信息。MOSFET将DS2432与其它电路隔离，当子系统正常工作时，也不会妨碍DS2432的正常访问。

该配置写入方法还带来另一个好处，系统在用户现场安装完毕后，允许进行远程更新/更改。任何未用于配置/功能管理的用户EEPROM均可采用电子标牌的形式实现电路板标识功能。该功能在应用笔记178——利用1-Wire产品标识印刷电路板中进行了详细说明，此应用笔记可从Maxim网站 www.maxim-ic.com.cn/an178 下载。

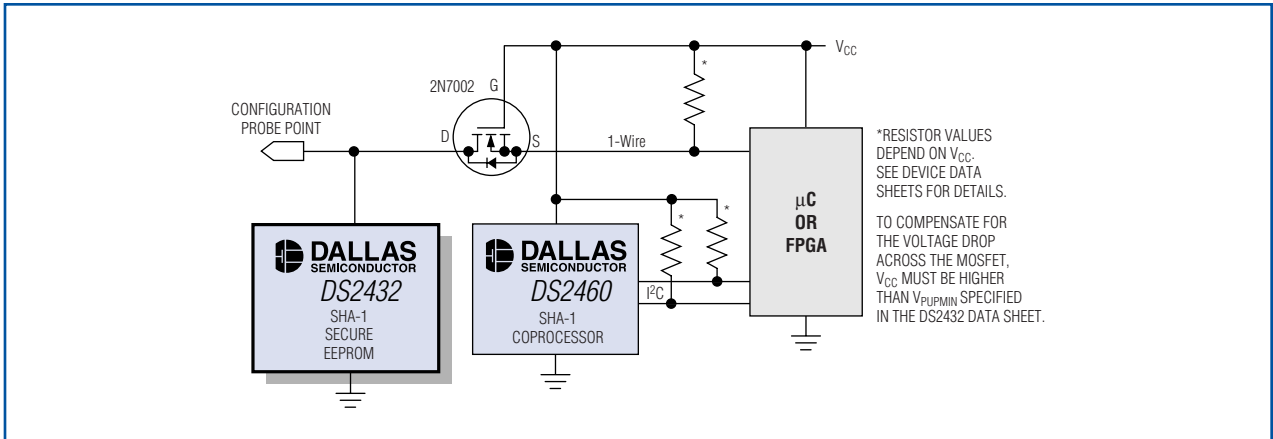


图6. 通过增加一个晶体管和配置探测点，可向DS2432写入配置信息。

DS2432认证功能详细说明

器件总体架构

DS2432 1-Wire接口、1kb SHA-1安全存储器的主要数据单元和数据流路径如图7所示。可以看到8字节密钥和临时存储质询码的缓冲存储器(暂存器)。前面未曾提及的数据单元包括独一无二的器件ID号(标准1-Wire特性)、四个用户EEPROM页面、控制寄存器和系统常数。

器件ID用作1-Wire网络中的节点地址，同时还用于

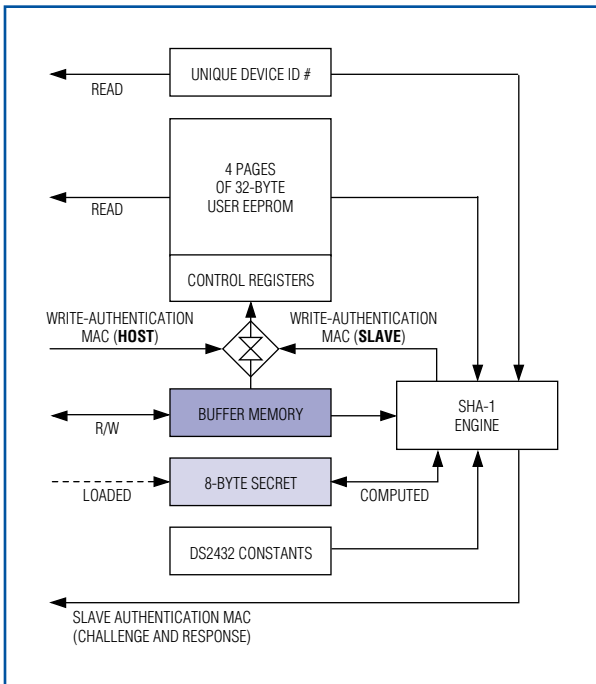


图7. DS2432 SHA-1安全存储器数据流模型的所有主要数据单元和数据流路径。

认证过程。用户存储器存放待认证“信息”的主要部分。系统常数有助于满足格式需求和完成填充功能，从而构成SHA-1计算的64字节输入数据块。控制寄存器执行特定的器件功能，例如可选的密钥写保护或EEPROM仿真模式；控制寄存器通常不参与认证过程。

可毫无限制地读取器件ID号和用户EEPROM。并可完全读/写访问缓冲存储器。可以直接装入密钥，但永远不能读取它。改变用户存储器或寄存器的内容要求主机和从机(即DS2432)计算出相同的写操作认证MAC，才可以打开缓冲存储器至EEPROM的路径。

取决于MAC结果的不同用途，DS2432 SHA-1引擎具有三种不同的工作方式。任何情况下，SHA-1引擎均接收64字节输入数据，并计算出20字节MAC结果。不同之处在于输入数据。作为安全系统的根本需求，主机必须要么知道、要么能够计算出应用中有效/合法从器件的密钥。

质询-响应认证MAC

正如前面的应用示例所述，DS2432的主要功能是完成质询-响应认证。主机发送一个随机质询码，指示DS2432根据该质询码、密钥、主机所选存储器页的数据、以及其它数据(这些数据共同构成信息)计算出响应MAC(见图8)。

DS2432完成计算后，将MAC回送给主机进行验证。主机使用有效密钥和DS2432所使用的相同信息数据重新进行MAC计算。如果该结果和DS2432给出的MAC是匹配的，则器件是合法的，因为只有合法的

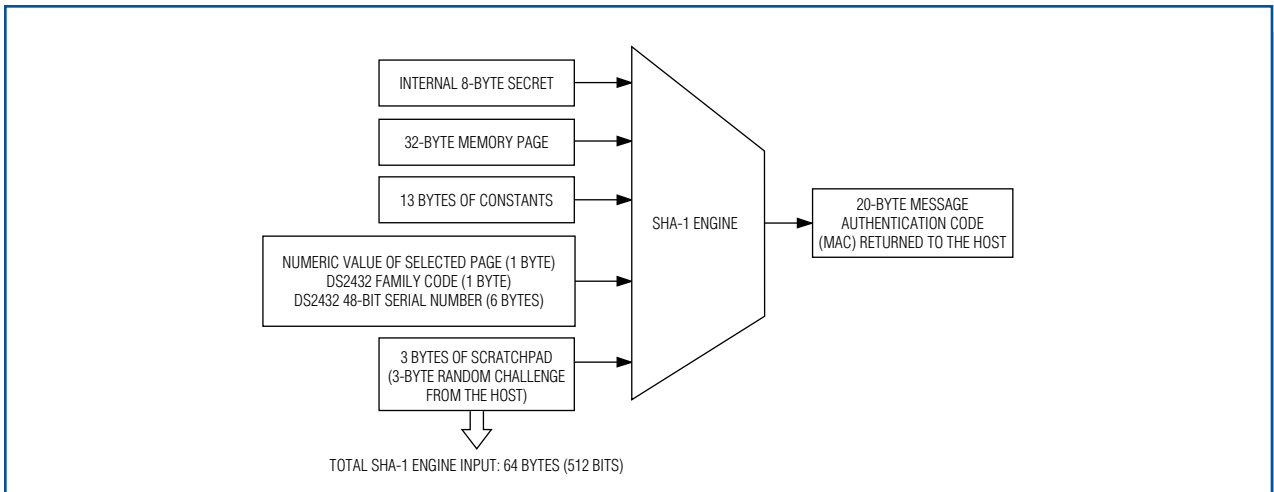


图8. 用于生成质询-响应认证MAC的DS2432 SHA-1引擎输入数据。

DS2432才能正确地响应质询-响应过程。质询码是随机数据这一点是非常重要的。如果质询码始终不变，很容易遭受一个利用有效、静态、记录和回放的MAC (不是使用认证DS2432实时算出的MAC)进行回放攻击。

数据安全

除了提供从器件的认证功能外，同时强烈要求存放在器件中的数据是可信的。为实现这一点，DS2432的写访问是安全受限的。将数据从暂存器拷贝到EEPROM或控制寄存器之前，DS2432要求主机提供写访问认证MAC来证明其合法身份。DS2432根据暂存器中的新数据、密钥、需要更新的存储器页数据、以及其它数据(图9)计算该MAC。

合法主机知道密钥并可计算出有效的写访问MAC。拷贝命令执行过程中收到主机MAC时，DS2432将其与自身计算的结果进行比较。只有当二者匹配时，数据才会从缓冲存储器传输至目标EEPROM。当然，不能修改写保护的存储器页，即使MAC是正确的。

密钥保护

DS2432的架构允许直接向器件装入密钥。可通过读保护提供密钥保护，如果需要，还可以采用写保护提供密钥保护，这将永远不能改变密钥。只要在设备制造现场访问密钥是安全和可控的，这种保护等级是很有效的。

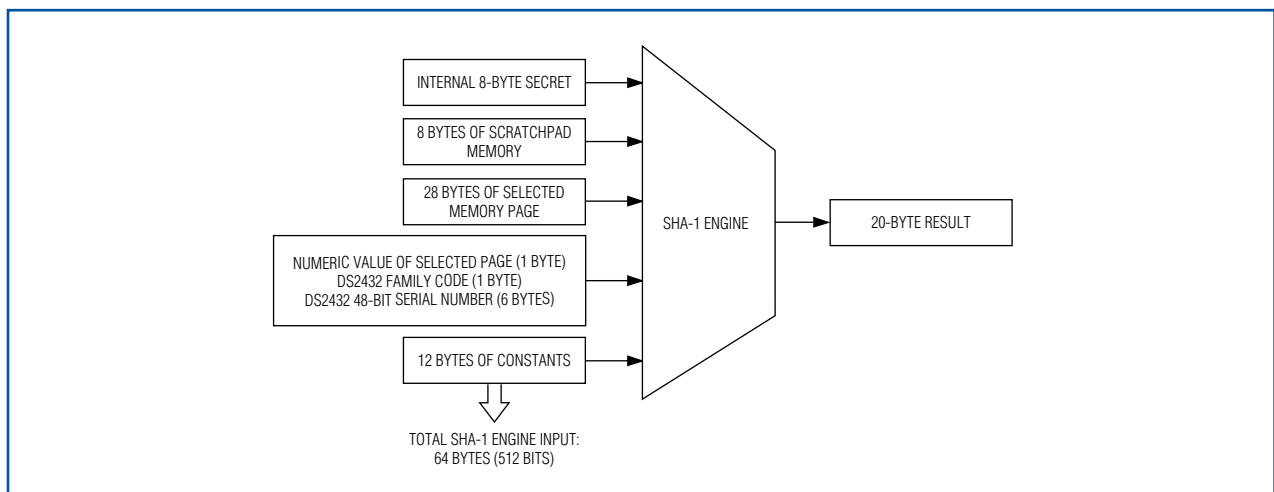


图9. SHA-1引擎输入数据用于计算写访问认证MAC。

可以采用不同方法提升密钥保护等级：1) 由DS2432计算其密钥；2) 由DS2432在不同场合分阶段计算其密钥；3) 计算密钥时包含独一无二的器件ID号，生成与器件相关的密钥；4) 组合第2和第3种方法。

如果采用上面第1种方法，每个DS2432自己计算其密钥，只知道计算密钥的原始数据；永远不会暴露密钥本身。如果采用第2种方法，密钥在不同场合分阶段计算，只知道密钥的“本地”原始数据。这种方法可有效控制“最终”密钥的信息。如果密钥是与器件相关的(第3种方法)，主机还需要增加一个计算步骤。但如果一个器件的密钥被意外发现，潜在危害却可降至最低。如果密钥分阶段计算，并且与具体器件相关(第4种方法)，可获得最高保护等级。但是，为确保系统保密性，主机和从机一样需要在不同地点进行设置。

计算密钥之前，必须先装入一个已知数值作为密钥。有了这个已知密钥，必须向四个存储器页之一写入

计算新密钥的32字节数据。接下来，需要向DS2432的暂存器写入一个局部密钥。局部密钥可以是用于计算的存储器页码和独一无二的器件ID号(CRC字节除外)，或任何其它与应用相关的8字节数据。

如果指示DS2432计算密钥，则DS2432启动SHA-1引擎，使用图10所示的输入数据计算MAC。20字节MAC的最低8个字节自动拷贝到密钥存储器地址，立即成为有效密钥。

结论

了解安全认证功能并巧妙实现，可提供极具竞争力的优势。认证不但保护了程序代码，而且公共硬件平台利用安全的软件功能设置有助于降低生产成本。DS2432的数据安全性甚至可以实现远程配置修改，节省了技术人员的宝贵时间。从DS2432所展示的功能可以看出，一个小小的硅晶片将对收益产生巨大的影响。

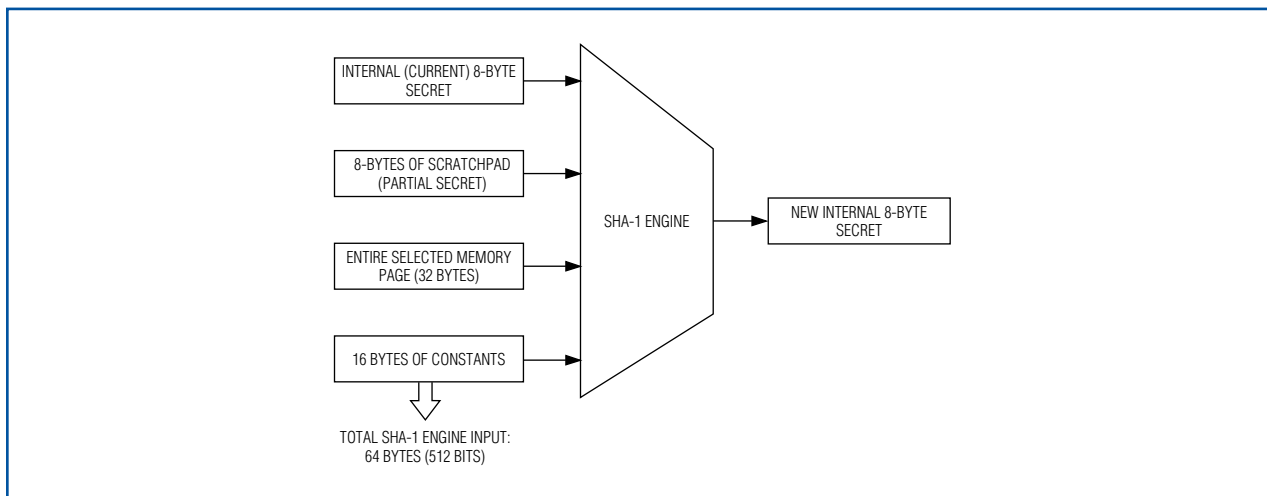


图10. 器件计算密钥时的输入数据；20字节MAC结果的最低8个字节成为新的密钥。

利用低功耗微控制器开发FFT应用

今天的低功耗微控制器(μC)也开始集成原先只存在于大型微处理器、ASIC和DSP中的外设功能,使我们有可能以很低的功耗实现复杂的算法。本文讨论一种快速傅立叶变换(FFT)应用,并在一个含有单周期硬件乘法器的低功耗 μC 上实现该应用。

这个FFT应用实时计算一路输入电压(图1中的 V_{IN})的频谱。为完成该任务,用一片模数转换器(ADC)对 V_{IN} 进行采样,获得的采样结果传送给 μC 。然后, μC 对这些采样执行256点FFT运算,获得输入电压的频谱。为便于检测, μC 将计算出的频谱数据传送给PC,由PC实时显示出来。

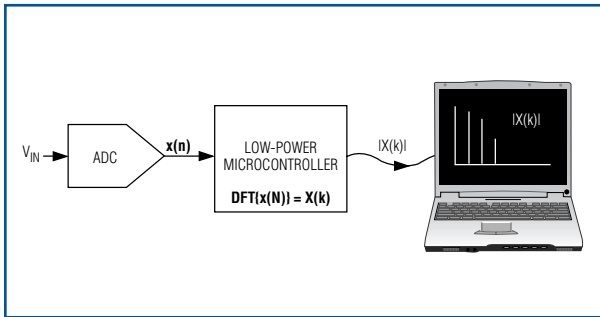


图1. 利用FFT应用计算输入电压的频谱。

该FFT应用的固件针对MAXQ2000系列中的一款16位、低功耗 μC 用C语言编写。有兴趣的读者可以通过网上的同名文章www.maxim-ic.com.cn/AN3722下载该项目的相关固件和电路原理图。

背景知识

为确定输入信号采样的频谱,我们需要对这些输入采样进行离散傅立叶变换(DFT)。DFT的定义如下:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \quad \text{for } 0 \leq k \leq N-1 \quad (\text{式1})$$

其中 N 是采样的数量, $X(k)$ 是频谱, $x(n)$ 是一组输入采样。利用欧拉公式展开求和符号,并分离输入采样和频谱的实部和虚部,得到以下等式:

$$\begin{aligned} X_{\text{Re}}(k) &= \sum_{n=0}^{N-1} \left[x_{\text{Re}}(n) \cos\left(\frac{2\pi kn}{N}\right) + x_{\text{Im}}(n) \sin\left(\frac{2\pi kn}{N}\right) \right] \\ &= \sum_{n=0}^{N-1} \left[x_{\text{Re}}(n) \cos\left(\frac{2\pi kn}{N}\right) \right] \end{aligned} \quad (\text{式2})$$

$$\begin{aligned} X_{\text{Im}}(k) &= - \sum_{n=0}^{N-1} \left[x_{\text{Re}}(n) \sin\left(\frac{2\pi kn}{N}\right) - x_{\text{Im}}(n) \cos\left(\frac{2\pi kn}{N}\right) \right] \\ &= - \sum_{n=0}^{N-1} \left[x_{\text{Re}}(n) \sin\left(\frac{2\pi kn}{N}\right) \right] \end{aligned} \quad (\text{式3})$$

式2和3中,求和符号中第二项的消失是由于输入采样全部为实数。假定我们有 N 个采样,直接计算式2和3需要 $2N^2$ 次乘法和 $2N(N-1)$ 次加法运算。这样,我们的256点输入采样DFT将需要进行131,072次乘法和130,560次加法运算。我们还是将注意力转向FFT吧!

有多种FFT算法可供使用。本应用采用普通的radix-2算法,继续将DFT分解为两个更小的DFT。为此, N 必须是2的幂。Radix-2 FFT算法的步骤可归纳为如图2所示的蝶型运算。观察这些蝶型运算我们可以发现,radix-2算法仅需 $(N/2)\log_2(N)$ 次乘法和 $N\log_2(N)$ 次加法。图2中用到的参数 W_N 就是通常所谓的“旋转因子”,可以在执行算法前预先计算出来。

在图2中,FFT的输入显示为一种特殊的排列顺序,这种序列是对原始序列索引号的二进制位反转后得到的。因此,当我们对 $N=8$ 个采样执行radix-2 FFT算法时,需要将输入数据的原始序列:

0(000b), 1(001b), 2(010b), 3(011b), 4(100b), 5(101b), 6(110b), 7(111b)

重新排列为:

0(000b), 4(100b), 2(010b), 6(110b), 1(001b), 5(101b), 3(011b), 7(111b)。

FFT输出则以正确的顺序排列。图2还说明,每个单独的蝶型运算得到的结果,是下一级FFT运算所需的唯一数据。由于运算过程可“即位”完成,新值可替代旧值,这样,计算 N 个采样的FFT只需要 $2N$ 个变量(因为每个数据都包括实部和虚部两部分)。

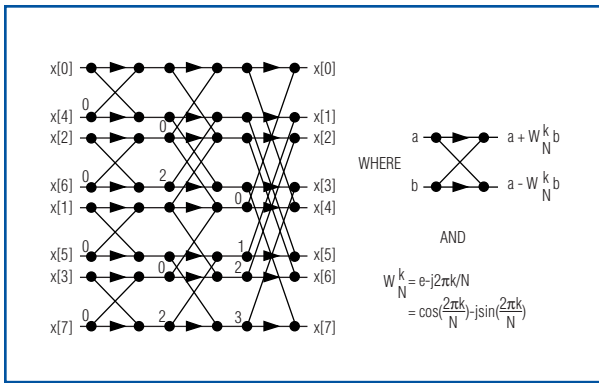


图2. 利用蝶型运算实现N=8的FFT。

FFT完成后，结果为复数形式。式4和5将结果转换为极坐标方式：

$$X_{MAGN}(k) = \sqrt{X_{Re}^2(k) + X_{Im}^2(k)} \quad (\text{式4})$$

$$X_{PHASE}(k) = \arctan\left(\frac{X_{Im}(k)}{X_{Re}(k)}\right) \quad (\text{式5})$$

有关DSP的文献中可以找到很多优化方法，可使上述DFT/FFT算法更小或更快。其中最重要的一种优化方法(可能也是最容易实现的)源于这样一个事实，那就是作为一个实数信号，其DFT幅度是相对 $X(N/2)$ 对称的，因此：

$$X(k) = X^*(N/k) \quad (\text{式6})$$

编写FFT代码绝非易事。低功耗 μC 的一些局限性又使该任务进一步复杂化。

存储器：我们所选的 μC 有2kB的RAM。已经知道该算法需要用到 $2N$ 个16位变量来存储FFT数据，这样，我们的 μC 可以执行 N 最高为512的FFT。然而，固件的其他部分也要用到一些RAM。因此在此项目中，我们将 N 限制为256。若采用16位变量来表示每个值的实部和虚部，FFT数据总共需要1024字节的RAM。

速度：低功耗 μC 尽管具有高MIPS/mA性能，仍然需要一些优化手段以最大程度减少运行FFT所需的指令数。好在本应用所用的C编译器(IAR的Embedded Workbench for MAXQ, 见www.iar.com)可提供多种级别的优化和设置。高效地使用硬件乘法器可使代码优化到可以接受的水平。

无浮点运算能力：通常的低功耗 μC 都不具备浮点运算能力。因此，所有运算都必须采用定点算法。为了表示小数，固件采用带符号的Q8.7表示法。这样，在固件中假定：

- 第0位至第6位代表小数部分
- 第7位至第14位代表整数部分
- 第15位代表符号位(二的补码)

这样的安排对于加法和减法没有影响，但在做乘法时必须注意将数据按照Q8.7格式对齐。

所选的数据表示法还要适应FFT算法可能遇到的最大数值，同时又要提供足够的精度。例如，我们的ADC可提供带符号的8位采样，以二的补码表示。如果输入为最大幅度(带符号的8位采样结果为127)的直流电压，则其频谱全部包含于 $X(0)$ 中，用Q8.7表示为32512。这个数值能够由单个带符号的16位数据表示。

固件

以下部分讨论在低功耗 μC 上计算radix-2 FFT的固件实现。从ADC读出采样结果后，将其存储在 $\mathbf{x_n_re}$ 数组中。这个数组代表 $x(n)$ 的实部。虚部存储在 $\mathbf{x_n_im}$ 数组中，在开始运行FFT前初始化为零。完成FFT后，计算结果会取代原始采样数据，被存储在 $\mathbf{x_n_re}$ 和 $\mathbf{x_n_im}$ 中。

获取采样

FFT算法假定采样是以固定的取样频率获得的。在获取FFT采样数据时如果不加小心会产生一些问题。例如，采样间隔的抖动就会给FFT结果引入误差，应尽力减小之。

ADC采样循环中的判断语句会造成采样间隔的抖动。例如，我们的系统从ADC读取带符号的8位采样数据，并将其存储在一个16位变量的数组中。在下面的程序清单1中给出了两种伪代码算法，以执行这种ADC读取-存储功能。算法1给出的方法会造成采样间隔的抖动，因为负采样比正采样需要花更多的时间来读取并存储。

清单1. 两种ADC采样伪代码算法。第二种算法避免了第一种产生的问题——采样间隔抖动。

```
// ALGORITHM 1: INCONSISTENT SAMPLING
FREQUENCY - BAD!
```

```

// sample[] is an array of 16-bit variables
for i = 0 to (N-1)
begin
    doADCSampleConversion()           //
    Instruct ADC to sample Vin
        sample[i] = read8BitSampleFromADC() //
    Read 8-bit sample from ADC
        if (sample[i] & 0x0080)           // If
    the 8-bit sample was negative
            sample[i] = sample[i] + 0xFF00 //
    Make the 16-bit word negative
end
// ALGORITHM 2: FIXED SAMPLING FREQUENCY -
GOOD!
// sample[] is an array of 16-bit variables
for i = 0 to (N-1)
begin
    doADCSampleConversion()           //
    Instruct ADC to sample Vin
        sample[i] = read8BitSampleFromADC() //
    Read 8-bit sample from ADC
end
for i = 0 to (N-1)
begin
    if (sample[i] & 0x0080)           // If
    the 8-bit sample was negative
        sample[i] = sample[i] + 0xFF00 //
    Make the 16-bit word negative
end

```

三角函数查找表

本FFT算法通过查表(LUT)而非计算得到正弦或余弦函数值。程序清单2给出了正弦和余弦LUT的声明。实际固件的注释中包含了自动生成这些LUT的源代码，可由程序调用。两个LUT均含有 $N/2$ 个分量，因为旋转因子的索引号变化范围为0至 $(N/2) - 1$ (见图2)。

清单2. 正弦和余弦函数LUT。

```

const int cosLUT[N/2] = {+128,+127,+127, ...
,-127,-127,-127};
const int sinLUT[N/2] = {+0 ,+3 , +6, ...
,+9 , +6, +3};

```

这些LUT数组声明为const，强制编译器将它们存储于代码空间而非数据空间。由于LUT数值须采用Q8.7表示法，它们对应于正弦和余弦实际值与 2^7 的乘积。

位反转

位反转排序(N已知)可在运行时通过计算、查表或直接利用展开循环编写。所有这些方法都需要在源代码的尺寸和运行速度间进行折衷。本FFT应用利用展开循环进行位反转，其源代码较长，但运行速度较快。程序清单3给出了该展开循环的实现过程。本应用固件的注释中包含了用于程序自动生成展开循环的源代码。

清单3. 用于实现 $N = 256$ 的位反转展开循环。

```

i=x_n_re[ 1]; x_n_re[ 1]=x_n_re[128];
x_n_re[128]=i;
i=x_n_re[ 2]; x_n_re[ 2]=x_n_re[ 64];
x_n_re[ 64]=i;
i=x_n_re[ 3]; x_n_re[ 3]=x_n_re[192];
x_n_re[192]=i;
i=x_n_re[ 4]; x_n_re[ 4]=x_n_re[ 32];
x_n_re[ 32]=i;
...
i=x_n_re[207]; x_n_re[207]=x_n_re[243];
x_n_re[243]=i;
i=x_n_re[215]; x_n_re[215]=x_n_re[235];
x_n_re[235]=i;
i=x_n_re[223]; x_n_re[223]=x_n_re[251];
x_n_re[251]=i;
i=x_n_re[239]; x_n_re[239]=x_n_re[247];
x_n_re[247]=i;

```

Radix-2 FFT算法

采样按照位反转方式重新排序后就可进行FFT运算了。本radix-2 FFT应用的固件通过三个主循环执行图2所示的蝶型运算。外循环计数 $\log_2(N)$ 级FFT运算。内循环执行每一级的蝶型运算。

FFT算法的核心部分是执行每次蝶型运算的一小块代码。程序清单4给出了这一块代码，遗憾的是，它是本应用中唯一“不可移植”的固件。宏MUL_1和MUL_2利用 μC 的硬件乘法器执行单指令周期乘法运算。这些宏的内容专用于MAXQ2000，可在实际固件中全部看到。

清单4. 用C编写的蝶型运算。

```

/* (1) Macro MUL_1(A,B,C): C=A*B      (result
in Q8.7)*/
/* (2) Macro MUL_2(A,C) : C=A*last_B (result
in Q8.7)*/
MUL_1(cosLUT[tf],x_n_re[b],resultMulReCos);
MUL_2(sinLUT[tf],resultMulReSin);

```

```

MUL_1(cosLUT[tf],x_n_im[b],resultMulImCos);
MUL_2(sinLUT[tf],resultMulImSin);
x_n_re[b] = x_n_re[a]-
resultMulReCos+resultMulImSin;
x_n_im[b] = x_n_im[a]-resultMulReSin-
resultMulImCos;
x_n_re[a] = x_n_re[a]+resultMulReCos-
resultMulImSin;
x_n_im[a] =
x_n_im[a]+resultMulReSin+resultMulImCos;

```

复数的极坐标转换

为了便于确定 V_{IN} 频谱的幅度，我们必须将复数形式的 $X(k)$ 转换为极坐标形式。实现该转换的固件见程序清单5。幅度值取代了原始的FFT结果，因为固件不再需要这些数据。

清单5。FFT结果从复数形式转换为极坐标形式。

```

const unsigned char magnLUT[16][16] =
{
{0x00,0x10,0x20, ... ,0xd0,0xe0,0xf0},
{0x10,0x16,0x23, ... ,0xd0,0xe0,0xf0},
...
{0xe0,0xe0,0xe2, ... ,0xff,0xff,0xff},
{0xf0,0xf0,0xf2, ... ,0xff,0xff,0xff}
};

...
...

/* Compute x_n_re=abs(x_n_re) and
x_n_im=abs(x_n_im) */
...
...

x_n_re[0] = magnLUT[x_n_re[0]>>11][0];

for(i=1; i<N_DIV_2; i++)
x_n_re[i] =
magnLUT[x_n_re[i]>>11][x_n_im[i]>>11];

x_n_re[N_DIV_2] =
magnLUT[x_n_re[N_DIV_2]>>11][0];

```

频谱幅度并非根据式4计算得到，而是通过一个二维LUT查表得到。第一索引号为频谱实部的高4位(MSB)，第二索引号为频谱虚部的高4位。为得到高4位索引号，可将带符号的16位数据右移11次。在使用频谱的实部和虚部作为索引号之前，需首先将它们转换为绝对值。因此，符号位为零。

从式6我们已经知道，频谱的幅度是对称于 $X(N/2)$ 的，因此我们只需将前 $(N/2)+1$ 个频谱数据转换为极坐标形式。还有，我们可以看到，对于实数输入采样， $X(0)$ 和 $X(N/2)$ 的虚部总为零。因此单独计算这两条谱线的幅度。本项目实际固件的注释中包含了用于自动生成该LUT的源代码，可由程序调用来计算 $X(k)$ 的幅度。

Hamming或Hann窗

此项目固件还包括了对输入采样加Hamming或Hann窗的LUT(Q8.7格式)。加窗函数可有效降低时域采样中 $x(n)$ 的舍入操作所引起的频谱泄漏。Hamming和Hann窗函数分别如式7和8所示。

$$h(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \quad (\text{式7})$$

$$h(n) = 0.5 \left[1 - \cos\left(\frac{2\pi n}{N-1}\right) \right] \quad (\text{式8})$$

程序清单6给出了实现这些函数的代码。同样，本项目实际固件的注释中包含了用于自动生成这些LUT的源代码，可由程序调用来实现这些窗函数。

清单6。用来实现Hamming和Hann窗函数的LUT。

```

const char hammingLUT[N] = {+10, +10, +10, ...
,+10, +10, +10};
const char hannLUT[N] = { +0, +0, +0, ...
, +0, +0, +0};
...
...
for(i=0; i<256; i++)
{
#ifdef WINDOWING_HAMMING

MUL_1(x_n_re[i],hammingLUT[i],x_n_re[i]); //
x(n)*=hamming(n);

#endif

#ifdef WINDOWING_HANN

MUL_1(x_n_re[i],hannLUT[i],x_n_re[i]);
// x(n)*=hann(n);

#endif
}

```

测试结果

为了测试该FFT应用的性能，固件将 $X(k)$ 的幅度通过 μC 的UART端口上传给PC。专门编写的 *FFT Graph* 软件用于从PC串口读取这些幅值，并以图形方式实时显示频谱(该项目提供此软件和固件)。图3所示为 μC 以200ksp/s采样四种不同输入信号并处理后，由 *FFT Graph* 所显示出来的结果：

- a) 4.3V 直流信号
- b) 50kHz 正弦信号
- c) 70kHz 正弦信号
- d) 6.25kHz 方波

接下来干什么？

有兴趣的读者还可以花费大量的时间来继续优化和重新配置该FFT应用。尽管在本文中我们选择了radix-2算法，还有很多其他算法可以显著降低加法和乘法运算量。有很多本文未提及的优化可以提升FFT的速度。例如，作为纯实数的输入采样，其虚部总为零，频谱中只有前半部分有实际意义。利用这一

点，第一级和最后一级FFT的执行速度可进一步优化，但需要付出更多的程序空间。

总之，本文所讨论的算法对于在低功耗 μC 上实现FFT应用而言，提供了一个很好的出发点。如果了解更多信息和具体实现的细节，请查阅我们为本应用所提供的、带有详细注释的固件。

参考文献

Cooley, J. W.和J. W. Tukey, "An Algorithm for the Machine Computation of Complex Fourier Series," *Mathematics Computation*, 第19期, 第297-301页, 1965。

Lemieux, Joe, "Fixed-point math in C," *Embedded Systems Programming*, 2003年10月。

Proakis, John G.和Dimitris G. Manolakis, *Digital Signal Processing Principles, Algorithms, and Applications*, 第3版, Prentice Hall, 1996。

Smith, Steven W., *The Scientist and Engineer's Guide to Digital Signal Processing*, 第2版, California Technical Publishing, 1999。

相似文章出现于2005年10月发行的 *Embedded Systems Design* 上。

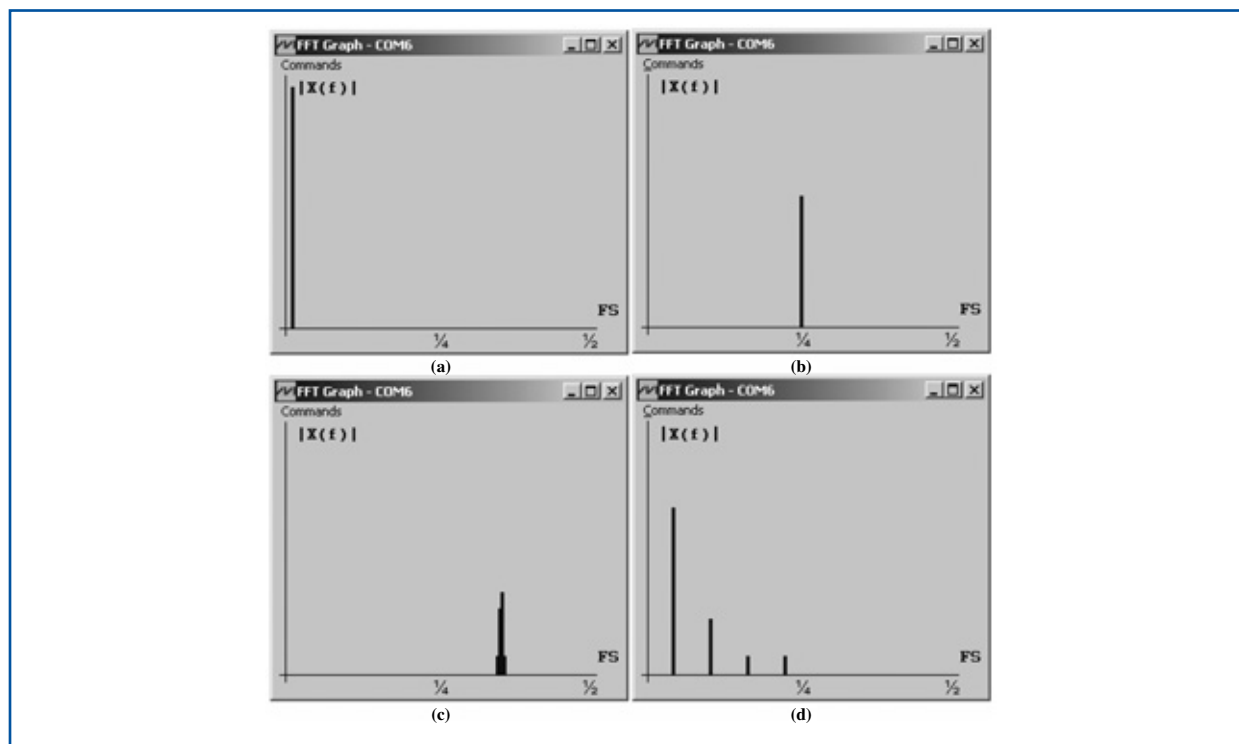


图3. FFT Graph软件显示的由低功耗 μC 计算出的频谱。

DESIGN SHOWCASE

精密监视负电源电流的电路

在一个高可靠系统中，电源电流监视通常是一项必备功能，因为电流过大会给系统造成损害，或危及其安全性。这些系统通过监视电源电流，在发生故障前将其关断，可以避免过载故障造成损害。然而，大多数电流监视IC被设计用于监视正电源。对于负电源，图1所示电路可以用来监视其负载电流并提供一个成正比的输出电压。

运放(IC1A)的同相端和反相端电压被有源反馈电流镜强制拉平(相等)。这样 $V_{R1} = V_{SENSE}$ ，因此：

$$I_{R1} = I_O \frac{R_{SENSE}}{R_1}$$

现在，你有三种选择。你可以通过连接一个电阻 R_O 到地、到 V_{CC} 或到一个反相放大器，来将输出

电流(I_{R1})转换为电压。 R_O 连接到地(GND)可省掉正电源。此时，输出电压为负且正比于负载电流：

$$V_O = -I_O \frac{R_{SENSE}}{R_1} R_O \quad (R_O \text{ 连接 GND})$$

你可以将 R_O 连接到 V_{CC} 以便得到正输出电压，但输出是以 V_{CC} 为参考电压的：

$$V_O = V_{CC} - I_O \frac{R_{SENSE}}{R_1} R_O \quad (R_O \text{ 连接 } V_{CC})$$

为得到以地为参考点的正输出电压，你需要采用一个反相放大器(IC1B)，如图1所示：

$$V_O = I_O R_{SENSE} \frac{R_2}{R_1} \quad (R_O \text{ 连接至反相放大器})$$

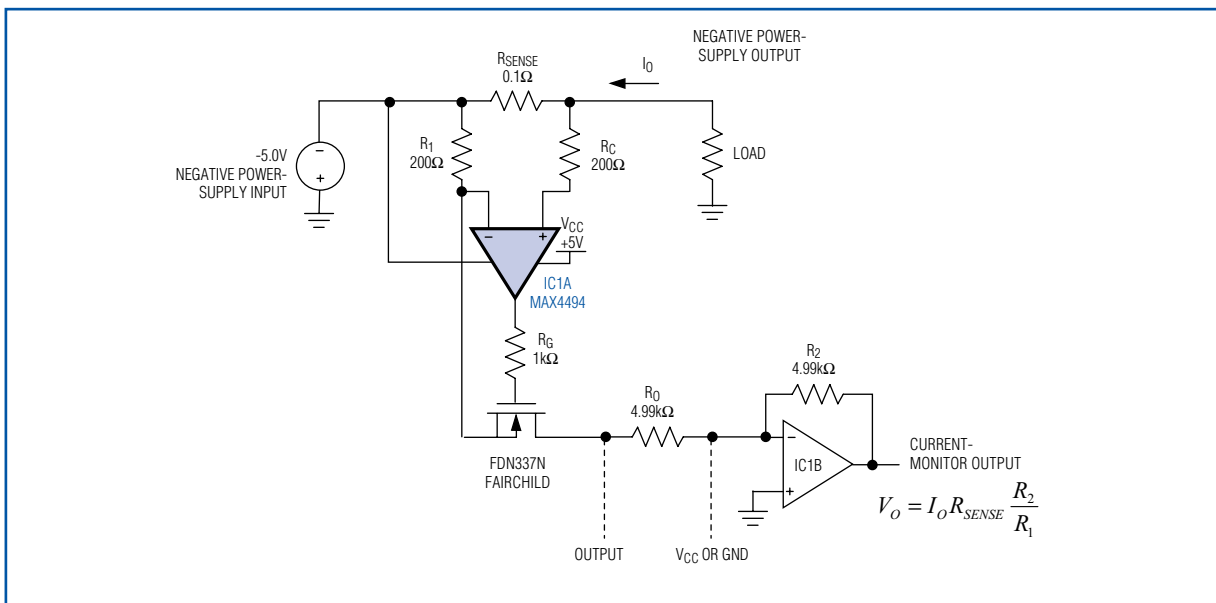


图1. 这个电流检测电路监视负电源，并提供一个正比于负载电流的正输出电压。

DESIGN SHOWCASE

注意图中的 R_O 并不影响反相放大器的输出电压，通常这个电阻是出于稳定性考虑而设的。 R_G 可选，也是用于保持稳定，因其可以隔离运放和MOSFET栅极容性负载。最后， R_C 用于补偿运放的输入偏置电流。

图2显示了图1电路的测量误差与负载电流的关系。为保证电流测量的精度，电阻(除 R_G 和 R_C 外)应具有优于1%的精度。 R_{SENSE} 的额定耗散功率必须高于负载电流在它上面产生的功耗。

类似文章出现在2005年9月出版的*Power Electronics Technology*上。

OUTPUT ERROR vs. LOAD CURRENT

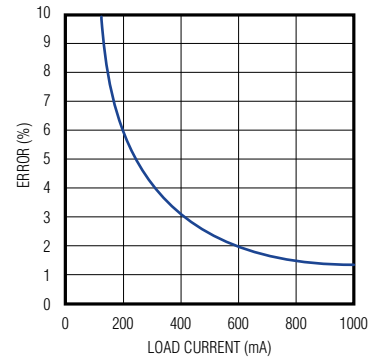


图2. 图1电路的电流测量误差在满量程时小于2%，但在较低电流下，运放固有的输入失调电压降低了测量精度。