完整指南

测试系统构建基础知识

文章

自动化测试系统总体拥有成本模型计算

仪器选择

自动化测试系统供电设施

开关和多路复用

测试执行软件

硬件和测量抽象层

机架布局和热分析

大规模互连和连接件

软件部署

系统维护

测试系统构建基础知识

自动化测试系统
总体拥有成本模型计算

目录

引言

大多数组织并不认为生产测试是最重要的，但为了防止产品在客户使用产品过程中出现重大质量问题而影响公司品牌，生产测试是必要的。然而，生产测试成本可能很高，并且常常被误解，特别是在没有简单的方法来量化高质量产品或缩短上市时间对业务的积极影响时。但是领先的企业对这种“必要之恶(necessary evil)”的观点并不以为意，因为他们思想超前，总是努力了解开发、部署和维护测试系统的总成本。而且实际上，自动化测试的成本远不止是测试机架的资本成本和操作员的人力成本。

本指南将介绍有关评估测试组织所需的工具和信息，可大幅节省成本的更改建议，以及如何做出更明智的投资决策来提高贵公司的盈利能力。

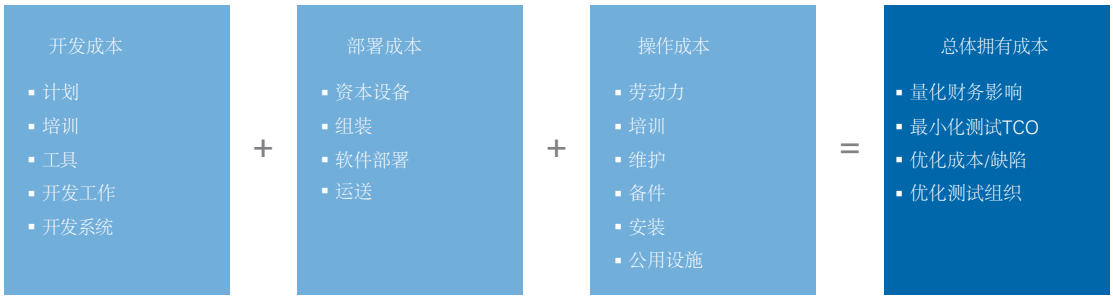


图1. 对总体拥有成本进行正确的建模有助于了解特定测试资产整个生命周期的成本，并提供了适应未来战略投资的财务框架。

开发成本

对于大多数应用，相比部署和操作及维护成本，构建自定义自动测试系统相关的开发成本是最低的。 这是因为我们通常仅构建一个系统来作为进行概念验证、性能基准测试和测试覆盖评估。 然而，开发测试系统的总成本根据最终目标的不同可能会有很大差异。 开发新产品的组织通常使用不同的架构和仪器来开发和比较多个测试系统，以确定最佳的方法。

研发（工程）团队负责产品设计和构建大部分开发系统，因此，成本通常算到其预算或成本中心中。 成熟的测试部门与他们的研发团队合作来影响产品的设计，通常称为测试设计或DFT（Design for Test），同时也致力于开发测试系统。 这是最佳做法之一，但并不总是所有测试组织都能够这样做。

如果构建的测试系统是用于测试单个设备或组件的功能，那么需求收集、仪器选择、连接固定和软件开发相关的工作较为简单。 然而，如果测试部门设计的是多用途的标准化测试系统来验证多个设备或组件的功能，则开发成本可能更高。 您必须花更多时间确定系统必须完成的所有功能，被测设备(DUT)连接件必须灵活，并且软件必须更具可扩展性，以便在添加新设备时轻松进行更改。

其他工作，例如编写硬件或测量抽象层或大规模互连系统，需要更多的前期开发成本，但对于技术快速更新换代或长生命周期系统面临仪器使用周期结束问题等情况，这些前期开发成本将给测试组织带来投资回报。

与开发自动化测试系统相关的主要成本是：

- **计划**- 正确估算确定测试系统所有可行选项所需的时间和费用，包括在供应商网站、产品演示会、评估、贸易展览和论坛上花费的时间。
- **开发人员培训** - 包括与学习一组新的软件开发工具（集成开发环境[IDE]或测试执行软件）和硬件平台（例如，基于SCSI或PXI的机架堆叠式平台）相关的时间和培训课程费用。
- **开发工具** - 购买测试软件（IDE或测试执行软件）开发许可证相关的成本。
- **开发工作** - 概念验证测试系统的硬件和软件开发时间。
- **开发系统** - 最初购买用于针对当前或其他新系统进行基准测试的概念验证或示范测试系统相关的资本成本。

部署成本

产品投入生产后，必须扩展概念验证或示范测试系统，以满足产品的产量需求。 测试系统的吞吐量（每个时间单元内测试的设备台数）直接影响满足需求所需的系统数量，产品管理和销售渠道决定了产量预测。 除了测试功能的覆盖率外，所需测试系统的数量也是在开发阶段最应该考虑的因素，因为这直接影响总部署成本。

提高测试系统部署成本的另一个因素是发货。对于小型组织，这并不是问题，因为制造测试和研发部门可以设在同一栋楼，或至少在地理位置上紧挨着。然而，即使是一些较小的公司，如果他们没有足够的能力或专业技术来自己制造和测试设备或组件，也会选择将产品的制造和测试外包。而大型公司可能将制造测试和研发部门设在同一个国家的不同地区，甚至在两个不同的国家。这会显着增加部署成本，特别是对于大型和/或重型制造测试系统的情况。较慢的货运方式有助于降低该成本，但前提是有足够的时间可以等待。最佳做法是在开发阶段考虑所有测试系统的物理尺寸和重量，特别是在比较两个选项时，因为这可显著降低下游成本。

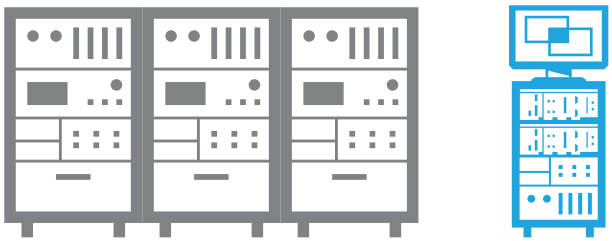


图2.在比较两个具有类似性能的测试系统时，应选择更小型、更轻质的测试系统来降低部署成本。

与部署自动化测试系统相关的主要成本是：

- **资本设备** - 所需的测试系统数量直接影响这一成本，而所需的测试系统数量由产品需求和测试系统吞吐量决定。
- **系统组装** - 将各个组件组装到测试系统中所需的时间，包括建造一个19英寸或21英寸的仪器支架或其他机械外壳、安装所有测试仪器、连接电缆和接线、安装开关和大规模互连以及固定。
- **软件部署** - 这些成本涉及编译软件组件，然后将这些组件从开发计算机导出到目标计算机以供执行。
- **运输和物流** - 测试系统的尺寸和重量以及生产或制造设施所需的测试系统数量将直接影响这一成本。运送的距离和收货所需的时间窗口也会影响成本。取决于系统的坚固性，可能需要特殊包装。

操作和维护成本

与测试系统相关的最后一项也是经常被忽视或低估的成本是运行和维护成本。 这些通常不归到最初设计产品或设备的研发团队，而是基本上由制造或生产团队承担；这种成本中心的分离使得跨部门协作成为共同的痛点。 如果公司选择将其产品的制造和测试外包出去，则外包制造商承担各项成本，并按统一费用或小时费率收取服务费。

与操作和维护自动测试系统相关的成本是：

- **每小时运行成本** - 测试系统操作员和技术支持人员的劳动力成本，以确保系统在制造期间正常运行。测试系统的数量和操作系统所需的技能水平直接影响该成本。
- **操作员培训** - 每个操作员学习如何使用测试系统所需的时间。成本通常仅限于每个操作员必须参加培训的时间量，而不管形式如何（动手培训、在线培训或面对面培训）。拥有多种测试系统的公司必须决定其人员配置策略，是每个操作员均可以操作每个测试系统，还是每个测试系统由专人操作。
- **维护** - 保持测试系统和仪器工作正常相关的成本。它通常包括每年校准设备的成本、以及在故障时更换仪器的预测成本。系统是否易于维护也会影响这一点。
- **备件库存** - 在计划外停机（例如仪器故障）或计划内停机（例如校准）时保留备用仪器所需的成本。每个测试系统都需要备用仪器；采用多个不同测试系统的公司，由于产品高度混合，需要更大量的备用仪器和零件，以确保较长的正常运行时间。
- **安装** – 功耗高或产生大量热量的测试系统需要安装特殊的大功率电气装置或冷却塔以确保正常的性能。
- **公用设施** - 与供电、冷却和占地面积相关的成本测试系统。制造工厂每平方英尺的价格和电费因地理位置不同可能会有较大差异。

财务分析方法

由于开发和部署成本会分摊到多个年份，并且未来会出现操作和维护成本，因此必须使用财务模型来确定测试系统的总体拥有成本。对于传统投资场景，项目需要产生收入和利润。 然而，对于测试系统来说，不存在收入或利润，而是哪个测试系统更具经济效益。类似的情况包括高效照明或建筑保温的投资，这需要前期成本，但成本会通过长远地减少公用设施费用来实现效益。

- **回收期（PP）** - 这是指回收项目投资成本所需的时间。计算包含两部分。首先，必须确定前期成本，看看开发部署新测试系统和部署更多旧系统之间的成本差。由于旧系统已经开发好了，因此没有相关成本。第二，前期成本除以新系统的效率（吞吐量）可节省的每年运营成本。

$$回收期(PP)\ [yr.] = \frac{前期成本[\$]}{每年节省[\$/yr.]}$$

- **投资回报率（ROI）** - 这是项目生命周期内利润与投资金额的比率，以百分比表示。这个计算较为复杂，因为它需要计算旧选项和新选项的预计总拥有成本，然后计算两者的差。接着将此结果除以更具成本效益的选项的总成本，所得结果再减去1（100%）即可得到投资回报率。

$$投资回报率(ROI)\ [\%] = \frac{净节省总额\ [\$]}{总成本[\$]} - 1$$

- **其他模型** - 要确定项目或财务投资的可行性，还可以使用许多其他财务模型，例如内部收益率(IRR)、净现值(NPV)和修正后内部收益率(MIRR)。但是，当将两个选项相互比较时，大多数高级模型都不适用，可以将分析简化为PP和ROI。

实际场景

以下实际场景有助于演示如何使用财务分析来确定总体拥有成本，从而有据可依地做出关于购买新测试系统架构的决策，而不是仍然采用旧方法。

概述

B公司是一家价值2亿美元的IP卫星通信系统制造商。 他们目前的生产测试系统是使用传统机架堆叠式仪器设计的。 B公司开发了这些测试系统，并将其部署到合同制造商的设备上进行产品测试，合同制造商以每小时30美元的统一费率收取费用。

以下是当前测试系统的主要特点：

- 完整的功能和全面的测试覆盖范围
- 中等资本投入
- 组织员工都接受过操作培训
- 吞吐量没有达到最佳状态

因为B公司最近投资了一个更大的销售渠道，使其雷达产品能够进入新市场，其生产能力必须从每年10,000台增加到25,000台。

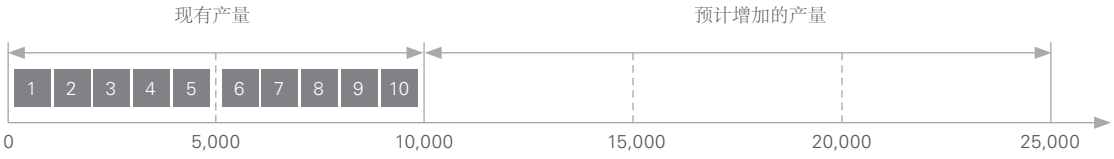


图3. 每年需要增产15,000个产品

他们的工程团队与NI合作，指定一个新的基于PXI的测试系统，该系统必须能够将每个DUT的测试时间缩短3倍。 然而，新的解决方案需要前期开发和部署成本，因此，在作出决定之前，必须建模来比较迁移到新系统和基于现有架构购买更多测试设备这两个选择对业务的影响。

现有机架堆叠式系统

NRE资本投资：	N/A
NRE开发时间：	N/A
资本支出：	每个系统$100,000
# 现有测试系统：	10台
测试时间：	每台设备40分钟
产量/吞吐量：	每年1,000台设备

基于PXI的新系统

NRE资本投资：	$90,000
NRE开发时间：	$150,000
资本支出：	每个系统$120,000
# 现有测试系统：	N/A
测试时间：	每台设备13分钟
产量/吞吐量：	每年3,000台设备

其他财务变量

摊销时间表：	5年
更换现有系统：	不需要，继续运行
每小时运行成本：	$30（合同制造商）
所需的吞吐量	每年25,000台设备

开发和部署成本

在评估过程中最常见的假设是，基于现有架构购买额外的测试系统更经济，因为组织员工已经受过充分培训，并且无需开发成本。 系统的架构是现成的，只需要复制即可。而新系统需要在开发期间会发生规划、架构设计、培训和其他非现金工程(NRE)成本。

但是，新系统的吞吐量优势不能忽视；吞吐量直接决定了达到预计产量增长必须购买的额外或新测试系统数量。在本例中，扩大现有测试系统的数量需要15个额外的系统，而新的基于PXI的系统只需要5个即可满足生产量。

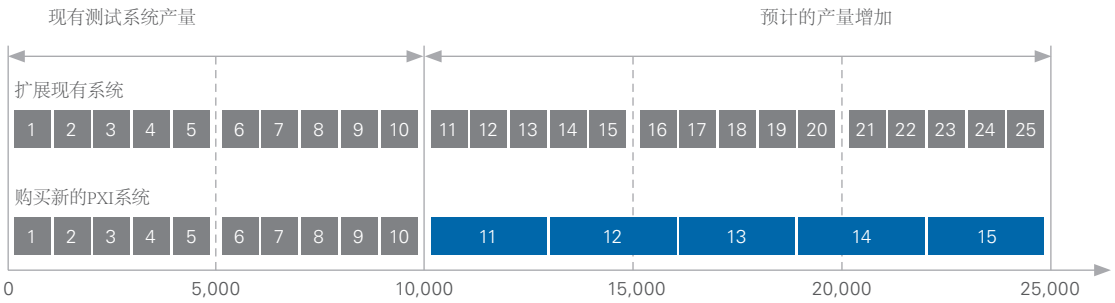


图4. 新PXI测试系统提供了高出3倍的吞吐量，大大减少了满足产品增产需求所需的系统数量。

在确定每种方法所需的测试系统数量后，您可以比较开发和部署相关的总成本，并直接理解吞吐量、资本支出和NRE的影响。

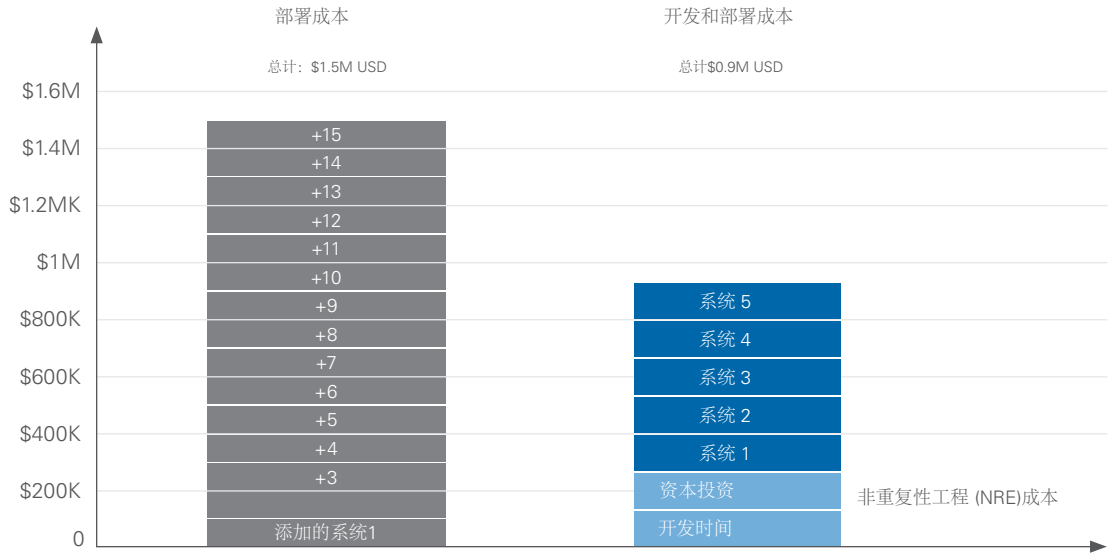


图5. 尽管新的基于PXI的测试会带来NRE开发成本，但是新系统的开发和部署总成本却降低了60万美元。

在本例中，比较开发和部署成本后，购买新解决方案比扩展现有测试系统更具成本效益。扩大现有系统的虚增成本主要源于系统的低吞吐量。仅仅从吞吐量的角度来看，就需要三倍的测试系统才能满足所需的产量，从而增加了部署成本。

但是如果变量发生了变化，会怎么样呢？对不同的假设情景进行建模，确保即使在最坏的情况下也有其优势。

要建模的一些假设情景包括：

- 如果新系统的开发时间翻倍，成本是否也翻倍？
- 如果资本支出由于通货膨胀而增加10%怎么办？
- 如果吞吐量仅提高了1.5倍，而不是3倍，会怎么样？
- 如果销售量从25,000台减少为仅20,000台，会怎么样？
- 如果额外的占地空间有限，该怎么办？
- 如果必须在测试设施中安装额外的电源或冷却装置，该怎么办？
- 如果之前的仪器使用周期结束，该怎么办？

操作和维护成本

在开发和部署所需数量的测试系统后，您必须在项目或产品整个生命周期中对其进行操作和维护。 与操作和维护测试系统相关的成本通常由公司的制造团队承担，而测试系统的开发和部署则由研发（工程）团队承担。 在没有领导指导的情况下，工程团队可能会对开发和部署进行成本优化，而不考虑对操作和维护成本的影响。

在上述仅考虑开发和部署成本的情况中，新测试系统比基于先前架构购买额外测试系统更经济。 现在分析项目前五年两个选项的运营和维护成本，以了解其对总测试成本的影响。

在本例中，B公司外包了产品的制造和测试。 合同制造商向公司B收取每小时30美金的测试费用。

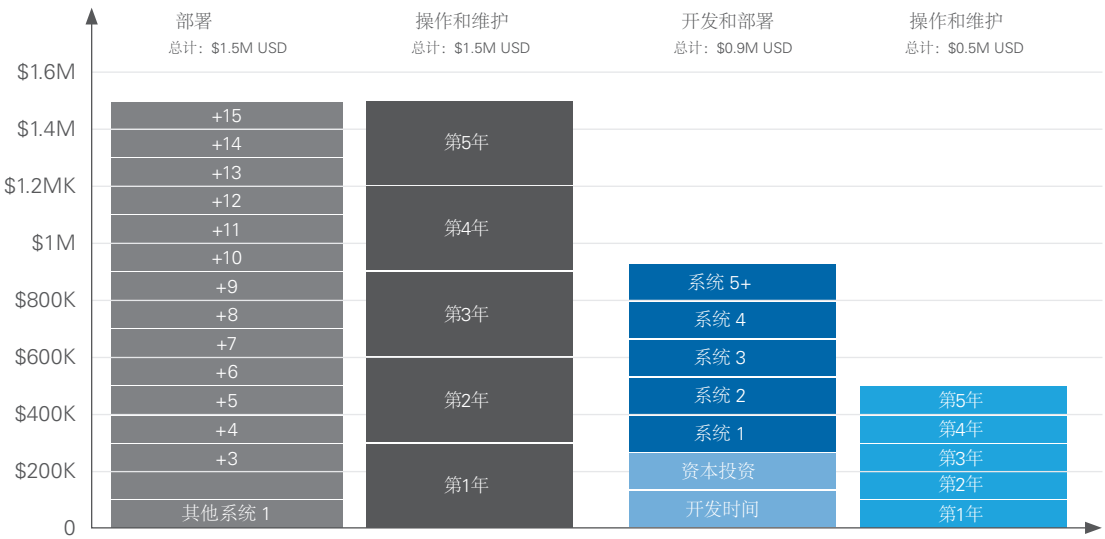


图6. 基于PXI的测试系统不仅具有很低的开发和部署成本，而且其操作和维护成本远低于之前的系统。

总拥有成本

虽然在这种情况下PXI选项是最好的选择，但确定总体拥有成本以有效地创建新系统的经济利益模型仍然很重要。 该五年分析包含了对PP、ROI、总节约成本和每个部分的测试成本降低等变量的分析。 在分析中，开发和部署成本平均摊销到五年内。

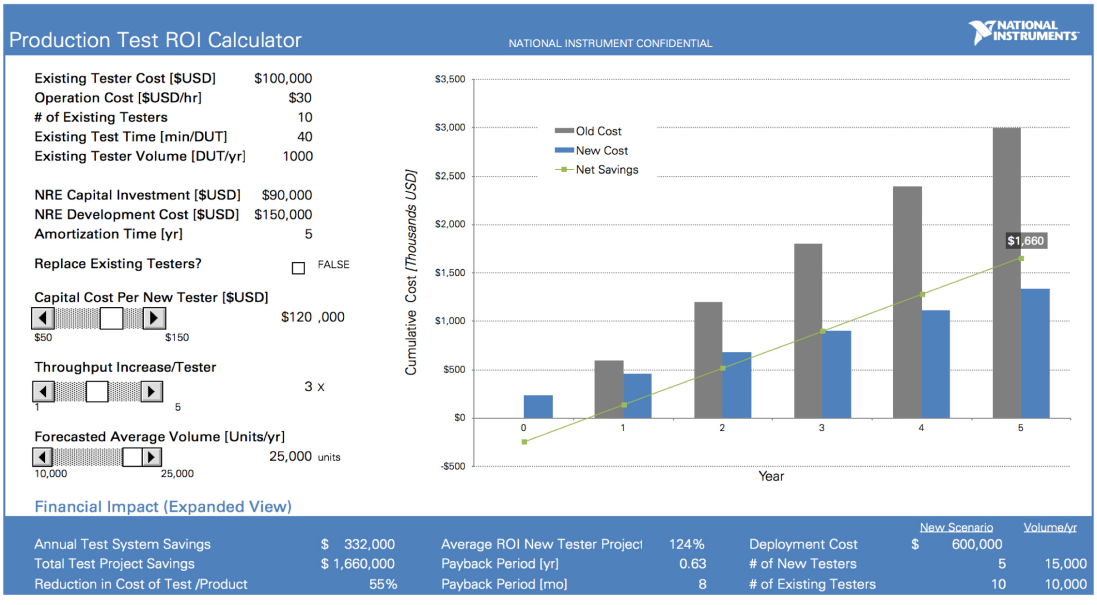


图7.与扩大现有解决方案相比，新测试系统在五年内总共节省了166万美元，投资回收期为11个月。

场景总结

在这两个选项中做出选择时，有许多因素需要考虑。 人们通常认为，扩展旧解决方案更容易且更便宜，但进一步分析显示，投资新的高性能系统才是一个明智的财务决策。 PXI系统的财务优势主要源于吞吐量提高了3倍，这使得B公司只需购买三分之一的测试系统即可完成同样的任务，从而节省了资金投入。 在这五年期间，这一决策也显着降低了他们向合同制造商支付的操作和维护成本，从而实现了11个月的PP和124%的投资回报率。

结论

随着设备复杂性和上市时间压力持续上升，自动化测试系统的总体拥有成本将继续在公司的盈利能力中发挥重要作用。 要实现这一目标，不能仅看测试系统的初始资本成本，应确保所有相关成本都纳入采购决策中。 本指南主要针对自动化生产测试，但您可以举一反三，将相同的概念应用于将产品从初始概念带到终端用户手里的其他阶段，包括研发、特性分析、验证和确认。

作为PXI平台、LabVIEW图形系统设计软件和TestStand测试管理软件的开发者以及PXI系统联盟的创始成员之一，NI 40多年来一直致力于帮助公司开发自动化测试系统，涉及的行业从半导体生产涵盖到航空航天和国防。 我们在全球50多个国家的区域现场工程师团队也可随时为各种规模的企业提供服务，确保最高的产品质量，同时降低测试成本。如有其它需求，请联系当地NI销售代表。

测试系统构建基础知识

仪器选择

目录

引言

工程师通常对 “为工作选择正确的工具” 这一名言的重要性不置可否。使用错误的工具可能会浪费时间并影响质量，而正确的工具则可在短时间内提供正确的结果。

在构建自动化测试系统时，使用的主要工具是测量仪器。 这些仪器包括数字万用表（DMM）、示波器和波形发生器等大家耳熟能详的仪器，以及各种新的和不断变化的产品类别，如矢量信号收发器和多功能一体式示波器。 如果要选择正确仪器，资深的测试工程师必须对以下方面有深入的了解:

- 被测设备的技术测量要求(DUT)
- 会影响应用的重要仪器规格
- 可用的各种类型的仪器，以及功能、尺寸、价格等方面的权衡
- 特定仪器类别中产品型号之间的细微差异

为工作选择正确的工具说起来容易，做起来难，特别是当涉及到评估许多权衡时。本指南介绍了可用仪器的主要类别，以及常见的选择标准，可帮助您缩小应用的最佳选择范围。

模拟和射频仪器

模拟和射频测试仪器的市场非常广泛，数以百计的产品类别包含数千个型号。 而且，这些仪器同样可根据物理定律来选择，具体来说，噪声和带宽的基本原理体现在放大器技术和用于创建仪器的模数转换器(ADC)上。 这些基本的物理限制使得工程师经常需要在测量精度和数据采集速度之间进行取舍。 下图显示的是随着传统和模块化仪器的技术进步，速度与分辨率之间的关系如何随着时间的推移而变化。

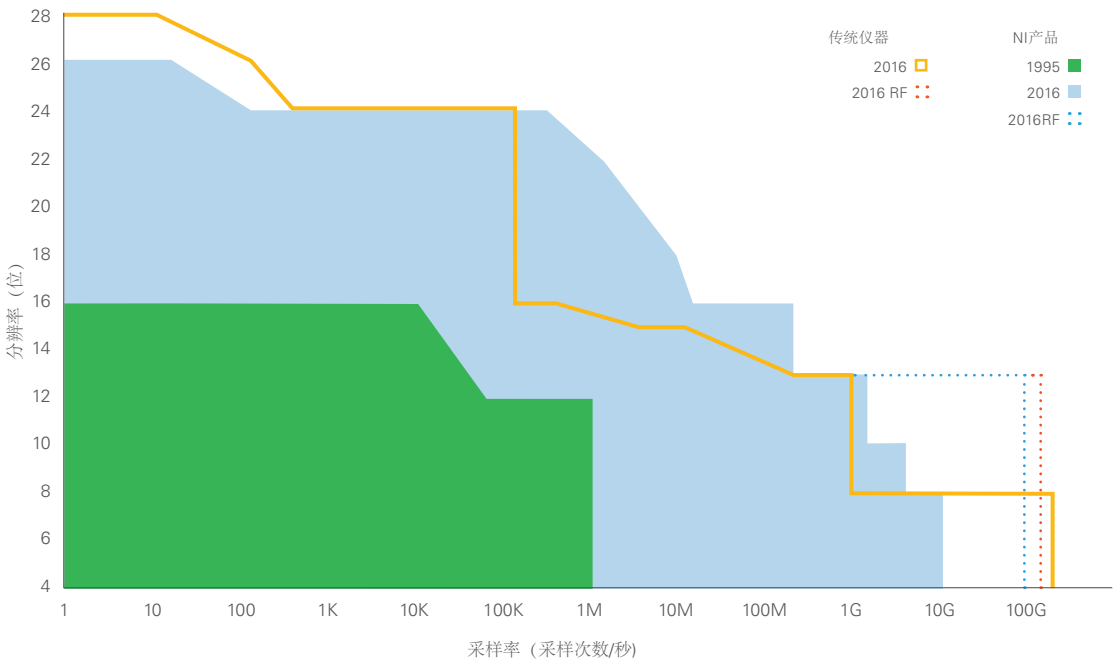


图1.分辨率与仪器的采样率

模拟和射频仪器类别

图1中的曲线代表了各种不同仪器类别的示例。图表左上角的DMM在低速下提供高精度，图表右下角的示波器在较低分辨率下提供高频采集，而左下角的DAQ产品则提供更高的通道密度和更低的成本。

如果要确定从哪个类型的仪器开始研究，首先要考虑几个关于您测量任务的关键问题：

- **信号的方向是什么？**（输入、输出或两者皆有）
- **信号的频率是多少？**（DC、kHz、MHz或GHz）

在确定上面关于方向性和速度这两个关键问题的答案后，便可根据表1来确定从哪种类型的仪器开始着手。

	DC和电源	低速模拟	高速模拟	RF和无线
输入，测量	数字万用表	模拟输入， 数据采集 (DAQ)	示波器， 频率计数器	RF分析仪 功率计 （频谱分析仪， 矢量信号分析仪）
输出，生成	可编程电源	模拟输出	函数/任意波形发生器 （FGEN，AWG）	RF信号发生器（矢量信号发生器，连续波源）
同一个设备上 输入和输出	DC电源分析仪	多功能数据采集卡 （多功能DAQ）	多功能一体式示波器	矢量信号收发仪 (VST)
同一个引脚上 输入和输出	源测量单元 (SMU)	电感电容电阻测量计	阻抗分析仪	矢量网络分析仪 (VNA)

表1. 模拟仪器类别

这个图表虽然有用，但包含的仪器类型非常少，特别是缺少垂直或特定用途的仪器。 该表未提及的一些值得注意的领域包括：

- 专用直流仪表，如静电计、微欧姆计、纳伏表等
- 音频频带分析和生成（也称为动态信号分析仪）
- 专业模拟产品，包括脉冲发生器、脉冲源/接收器等

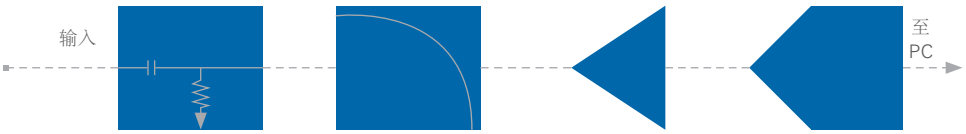
需考虑的关键规格

在将测量任务缩小到特定仪器类别之后，下一步是对该类别的产品进行比较和权衡，需考虑的规格包括：

- **信号范围、隔离和阻抗** - 首先，确保仪器的输入信号范围足够大，可捕获感兴趣的信号。 此外需考虑仪器的输入阻抗（影响测量装置的负载和频率性能）以及仪器与地面的隔离（影响抗噪声性和安全性）。
- **模拟带宽和采样率** - 接下来，确保仪器的模拟带宽（以kHz、MHz或GHz为单位）能够传递感兴趣的信号，并且ADC具有足够快的采样率来捕获感兴趣的信号（以每秒样本数单位，例如每秒千个样本、每秒百万个样本或每秒十亿个样本）。
- **测量分辨率和精度** - 最后，评估仪器垂直规格中影响测量质量的多个参数，如ADC分辨率（模拟信号的数字量化，通常在8位至24位之间）、测量精度（最大测量误差随时间和温度的变化，一般以百万或百万分之一表示）和测量灵敏度（最小可检测变化，通常以绝对单位表示，例如mV）

在量程、精度和速度等功能维度具有出色性能的仪器将可能需要在价格、尺寸、功耗和通道密度方面进行取舍，所有这些都会影响仪器的实用性。

图2显示了通用测量仪器的模拟输入路径简图，包含四级主要输入、每一级影响的仪器规格，以及典型DMM和典型示波器的仪器规格受每一级影响的示例。



	输入隔离	输入耦合和滤波	输入放大器	模数转换器(ADC)
确定的规格	隔离输入阻抗	AC/DC耦合模拟带宽	最大电压范围 最低电压灵敏度	采样率分辨率
DMM示例：	隔离电压高达300 V Cat II 10 MΩ（可选择）	DC耦合 200 kHz 带宽	高达300 V输入 低至10 nV灵敏度	10k Hz 读取速率 6.5数字（24位）分辨率
示波器示例：	接地参考50 Ω 或 1 MΩ（可选择）	DC或AC耦合（可选择）） 350 MHz带宽	高达40 Vpp输入 低至1 mV灵敏度	高达5 GS/s采样率 8位分辨率

图2. 模拟仪器的四级输入

上面的表格概括了筛选仪器规格时的一些思路，通常使用各种仪器类别和仪器供应商的各种不同命名法来表示。 这些级在影响关键规范时通常是相互依赖的。 例如，输入放大器还可以影响仪器的输入带宽和有效分辨率。 类似地，仪器的输入阻抗可能对带宽具有显著影响。

模拟和射频仪器类别

在比较DUT的测量要求以及仪器测试DUT的能力时，请记住以下关键比率。

测试准确度比= 4:1

当测试组件（例如电压参考）时，请确保测量设备的准确度远远大于被测组件的准确度。 如果不满足该标准，则测量误差可能同时来源于DUT和测试设备，这样便不可能知道真实的误差源。 因此，测试准确度比(TAR)这一概念可用于描述测量设备和被测组件的相对准确度。

TAR的可接受值为4及以上，取决于所执行的测试和所需的测试确定度。

$$TAR = \frac{\text{待测组件所需的准确度}}{\text{测量设备的准确度}}$$

宽带比 = 5:1

上升时间和带宽直接相关，可以通过一个值算出另一个值。 上升时间定义信号从满量程值的10%上升到90%所需的时间。 可根据上升时间使用以下公式计算出信号的带宽：

$$\text{带宽} = \frac{0.35}{\text{上升时间}}$$

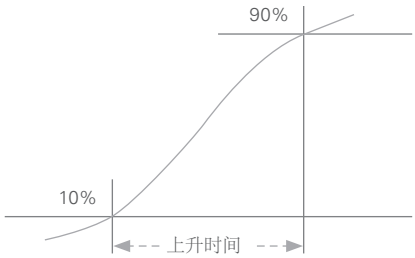


图3. 模拟信号上升时间

理想情况下，数字化仪的带宽应该是信号带宽的三到五倍，如上式所计算。 换句话说，数字化仪的上升时间应该是信号上升时间的1/5到1/3，才能以最小的误差采集信号。根据以下公式可随时逆推来确定信号的实际带宽：

$$T_m = \sqrt{T_5^2 + T_d^2}$$

T_m = 测量的上升时间，T_5 = 实际信号上升时间，T_d = 数字化仪的上升时间

时域采样比 = 10:1

尽管带宽描述了可以以最小衰减进行数字化的最高频率正弦波，但采样率仅仅是数字化仪或示波器中的ADC提供时钟以对输入信号进行数字化的速率。 采样率和带宽不直接相关; 然而，这两个重要参数之间存在一个我们希望的关系：

数字化仪实时采样率=输入信号带宽的10倍

奈奎斯特定理指出，为了避免混叠，数字化仪的采样率需要至少是被测信号中最高频率分量的两倍。 然而，仅仅是最高频率分量的两倍并不足以精确地再现时域信号。 为了准确地数字化输入信号，数字化仪的实时采样率应至少为数字化仪带宽的三到四倍。具体原因请看下图，想想你希望在示波器上看到的数字化信号。

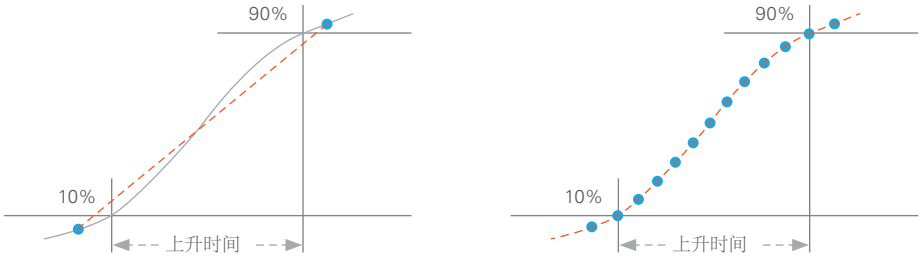


图4. 右图显示的是具有足够高采样率的数字化仪，可精确地重建信号，从而实现更精确的测量。

尽管在两种情况下通过前端模拟电路的实际信号都是相同的，但是左侧的图像属于欠采样，会使数字化信号失真。相反，右侧的图像具有足够的采样点来精确地重建信号，这可实现更精确的测量。因为信号的清晰表示对于上升时间、过冲或其它脉冲测量的时域应用非常重要，所以具有更高采样的数字化仪有益于这些应用。

数字仪器

在电子功能测试环境中，数字仪表用于与数字协议连接并测试协议的电特性和通信链路特性。 影响特定任务的可用仪器的一个最关键方面是并行与串行数字通信比较。

并行与串行标准比较

由于1GHz到2GHz附近的并行总线的时钟速率存在物理限制，串行标准越来越受到欢迎。 这是因为由每个时钟和数据线引入的偏移会以更快的速率引起误码。高速串行总线会发送包含单个差分信号数据和时钟信息的编码数据，从而避免了并行总线的速度限制。串行化数据以及以更快的速率发送有助于减少集成电路(IC)的引脚数，从而减小了尺寸。此外，因为串行通道可以以快得多的时钟速度运行，所以它们还可以实现比并行总线更高的数据吞吐量。

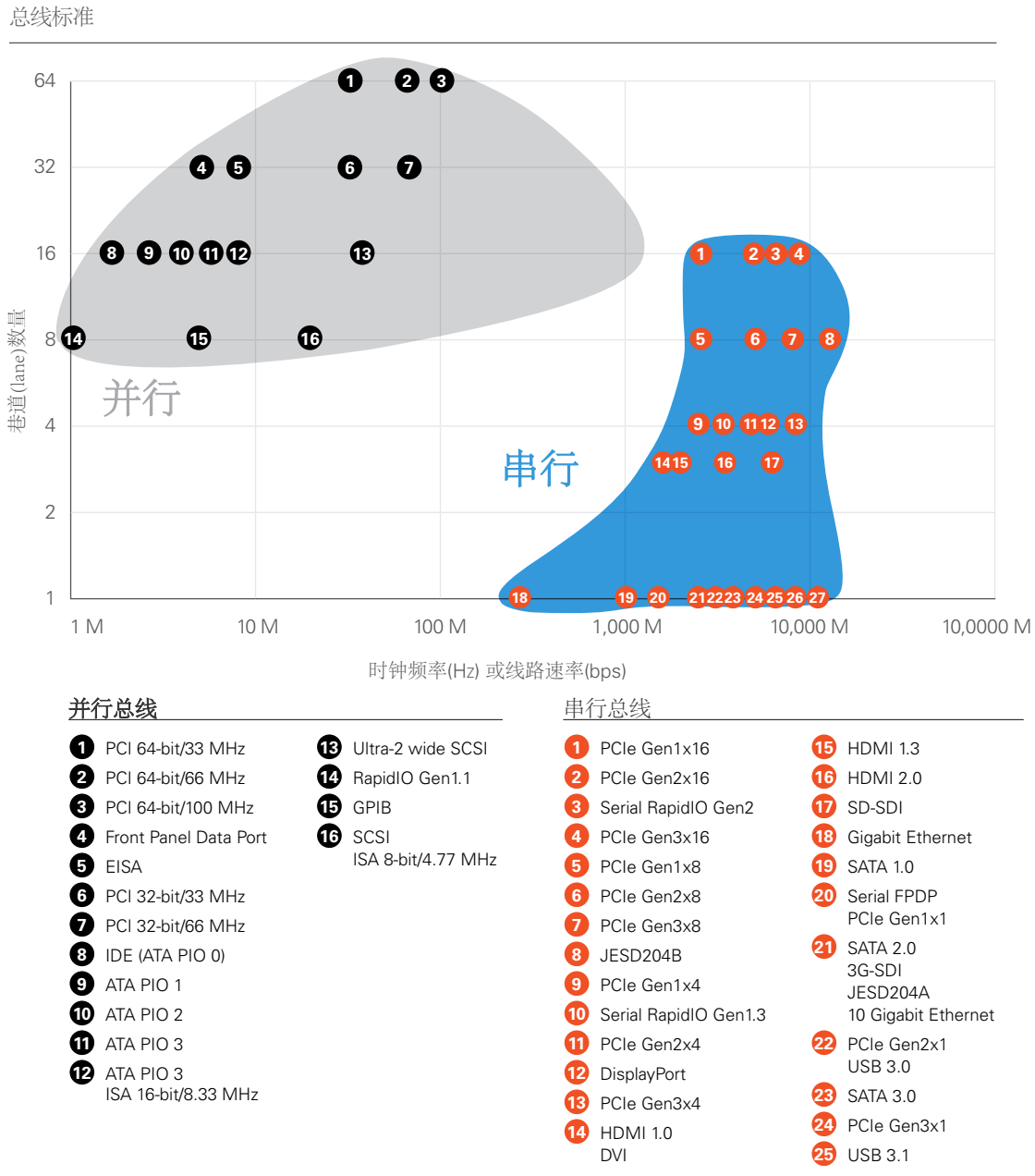


图5. 该图显示了一些常见的总线标准及其巷道数量与线路速率的关系图。串行标准具有比并行标准高得多的线路速率，从而可实现更高吞吐量。

数字仪器类别

与模拟仪器一样，您可以使用以下几个关键问题快速缩小数字仪器的选项范围：

- **需要完成哪些任务？** （数字连接、自定义数字连接或电气和定时测试）
- **链路的速率有多快？** （静态和kbit/s、Mbit/s或Gbit/s）

	静态低速	同步和高速并行 (100 MBit/s)	高速串行 (100 GBit/s)
接口（标准）	低速标准接口卡(I2C, C) 同步协议接口 （ARINC 429，CAN，GPIB，I2C，SPI）		接口卡 （10 千兆以太网，光纤通道， PCI Express等）
接口（自定义）	数字I/O (GPIO)	数字波形发生器/ 分析仪分析仪，测试图案发生器	基于FPGA的高速串行接口 Aurora, Serial Rapid I/O, JESD204b
(电气测试和定时测试 （基本接口）	引脚电子数字， 每引脚参数测量单元(PPMU)		BERT，示波器

表2. 数字仪器类别

硬件定时与软件定时

数字通信方案主要通过两种方法实现：软件定时和硬件定时。 软件定时应用的输入输出不需要使用任何类型的时钟。 软件负责控制I / O，而编程语言则通过软件来控制定时。 这种编程语言通常在操作系统上运行，可能需要几毫秒来执行软件调用。 对于软件定时，可以使用操作系统定时器来确定定时操作的速率。 通常，监测和控制报警、电机和报警器等低速应用使用软件定时。

有两种类型的软件定时通信可供选择：确定性控制和非确定性控制。 使用实时操作系统，可以实现高达1μs的精度；但是，实时操作系统不会提高通信速率，仅仅是增加确定性。 非实时系统，如Microsoft Windows，是非确定性的。 在这些系统中，软件命令在硬件中执行所花费的时间是不一致的，并且可能要花费几毫秒。 计算机内存、处理器速度和在操作系统上运行的其他应用程序等因素都可能会影响执行时间。

相比之下，硬件定时设备使用时钟的上升沿或下降沿进行确定性生成或采集。 您可以使用此类定时，以非常高的确定性在千兆位每秒的速率下采集或生成数字数据，并且可以在预确定的位置可靠地输出数据。

使用硬件定时的应用包括：

- 芯片测试
- 协议仿真和测试
- 数字视频和音频测试
- 数字电子测试

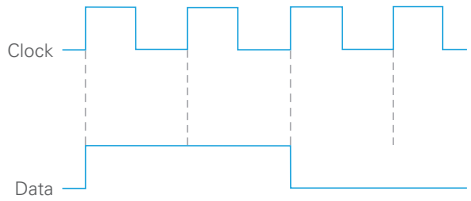


图6. 通过硬件定时操作，您可以利用实时确定的数字信号输出。

时钟速率

硬件定时数字应用的一个重要考量因素是时钟速率。 如果设备的最大时钟速率不够块，将难以补偿。 您可以使用NI高速数字I/O设备实现高达200 MHz的单端信号采样速率和高达200 MHz的差分信号，从而实现协议、数字音频和视频以及数字电子等的测试。 对于设备可能不满足串行数据流所需的时钟速率要求的情况，可以使用串行器/解串器（SERDES）来采集更高频率的数字信号。 但是，根据使用的SERDES类型，采用SERDES可能会减少可用线路的数量。

组成结构

除了理解从物理角度进行正确测量所需的模拟前端外，您还需要稳定、可重复、快速且连接PC仪器- 这是工作的一部分。 这有助于您根据环境做出决定：

- **对于台式和实验室** —准确度、重复性、底层控制、易于安装和重复测试自动化的能力
- 对于制造车间 - 速度、吞吐量、准确度、通过编程接口优化以及调试

显然，在如何选择实验室与制造车间所需的仪器上有相似之处和不同之处。通常需要针对终端部署对仪器的组成结构进行一系列主要成功标准评估。以下是您可能在制造环境中看到的一组典型的评估标准。

功能需求	测试工程说明
仪器，需要I/O？	
处理，需要计算？	
数据吞吐量，存储？	
同步？	
未来需求？	
需要在几年内部署几个系统？	
计划使用的年限？	
全球站点复制的数量？	
部署场景的环境稳定性？	
初始设置、配置和维修如何管理？	
机架安装式？	
尺寸、重量和功耗？	
连接件和连接？	

表3. 硬件部署检查清单

选择总线类型

今天，USB、PCI Express和以太网/LAN作为仪器控制的有效通信选项而备受关注。 一些测试和测量供应商和行业权威人士声称，其中这些总线本身就代表了所有仪器需求的解决方案。实际上，未来的测试和测量系统很可能仍然同时采用多种总线技术，因为每个总线都有自己的优势。

带宽

在考虑其他总线的技术优点时，带宽和延迟是两个最重要的总线特性。带宽衡量的是通过总线发送数据的速率，通常以兆字节每秒为单位。高带宽总线可以在给定周期内比低带宽总线传输更多数据。大多数用户认识到带宽的重要性，因为带宽会影响数据是否能够以与采集和生成一样快的速率通过总线输入或输出共享主机处理器，以及仪器需要多少板载内存。带宽对于复杂波形生成和采集以及RF和通信等应用十分重要。高速数据传输对于虚拟和合成仪器架构尤其重要。虚拟或合成仪器的功能和特性由软件定义；在大多数情况下，这意味着数据必须移动到主机PC进行处理和分析。图7显示了本指南中介绍的所有仪表总线的带宽（和延迟）。

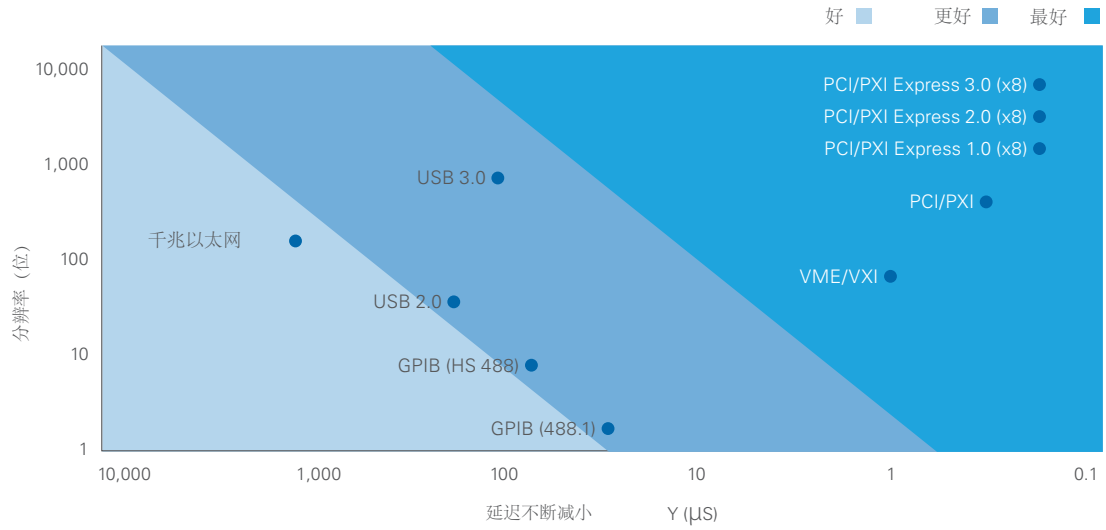


图7. 不同仪器总线的带宽和延迟关系图

延迟

延迟衡量的是通过总线的数据传输延迟。 打个比方，将仪表总线看成公路，带宽则对应于车道数和行驶速度，而延迟则对应于入口匝道和使出匝道引入的延迟。 低（越低越好）延迟总线在一端发送数据和另一端处理数据的时间之间的延迟较少。 延迟虽然比带宽更难观察到，但是会直接影响需要通过总线快速连续地发送间断性短命令的应用，例如DMM和开关之间的握手以及仪器配置。

GPIB

IEEE 488总线（通常称为GPIB）是专为仪器控制应用而设计且经过验证的总线。 GPIB作为一种强大可靠的通信总线已经应用了30年，仍然是仪器控制最受欢迎的选择，因为它具有低延迟和可接受的带宽。 GPIB目前拥有最广泛的行业领先地位，超过10,000台仪器型号采用GPIB连接。

GPIB的最大带宽约为1.8 Mbytes/s，最适合用于独立仪器的通信和控制。 最新的高速版本HS488将带宽增加至高达8 Mbytes/s。 数据的传输基于消息，通常采用ASCII字符的形式。 多个GPIB仪器可以通过电缆连接，使总距离长达20 m，并且总线上所有仪器之间共享带宽。 尽管带宽相对较低，GPIB延迟明显低于（更好）比USB，特别是以太网。 GPIB仪器在连接到系统后不会自动检测或自动配置；但是GPIB软件仍是最佳选择之一，而且坚固的电缆和连接器也是恶劣物理环境的理想选择。 GPIB是自动化现有设备或需要高度专业化仪器的系统的理想选择。

USB

USB近年来广泛用于连接计算机外设。 随着越来越多的仪器供应商在仪器中添加了USB设备控制器功能，这种普及已经扩展到测试和测量领域。 虽然大多数笔记本电脑、台式机和服务器可能配有几个USB端口，但这些端口通常都连接到同一个主机控制器，因此USB带宽在所有端口之间共享。

USB的延迟相对较好（介于慢速的以太网和快速PCI和PCI Express之间），并且电缆长度最长为5m。 USB设备可自动检测，这意味着与其他acargteyhfrjutgk技术（如LAN或GPIB）不同，USB设备在连接PC时会立即被PC识别和配置。 USB连接器是所有介绍的总线中最不可靠和最不安全的。 可能需要外部电缆扎带将其固定到位。

USB设备非常适合于需要便携式测量、笔记本电脑或台式数据记录以及车载数据采集的应用。该总线已经成为独立仪器的常见通信选择，因为它在PC上普遍存在，特别是其即插即用的易用性。 USB测试和测量类（USBTMC）规范满足各种测试和测量设备的通信要求。

PCI

PCI和PCI Express是本文介绍的所有仪器总线中带宽和延迟性能最好的总线。 PCI带宽为132 Mbytes/s，该带宽由总线上的所有设备之间共享。 PCI延迟以700 ns为基准，与以太网的1 ms相比，性能非常出色。 PCI使用基于寄存器的通信。与此处提到的其他总线不同，PCI不能连接到外部仪器。 相反，它是用于PC插入卡和模块化仪器系统（如PXI）的内部PC总线，因此无法直接进行距离测量。 尽管如此，当连接到PXI系统时，通过使用NI光纤MXI接口，PCI总线可以扩展到长达200 m。 因为PCI连接在计算机内部，所以我们可以将连接器的鲁棒性看成受其所驻留的PC的稳定性和耐用性的约束。

PXI模块化仪器系统基于PCI信号，通过高性能背板连接器和多个螺栓端子增强了连接性，保持连接到位。插入PCI或PXI模块启动后，Windows就会自动检测并安装模块的驱动程序。 通常，PCI仪器可以实现更低的成本，因为它们依赖于托管它们的PC电源、处理器、显示器和存储器，而不是将该硬件并入仪器本身中。

PCI Express

PCI Express类似于PCI。 它是PCI标准的最新版本。 因此，PCI的上述大部分评估也适用于PCI Express。

PCI和PCI Express性能的主要区别在于PCI Express具有更高的带宽，并为每个设备提供专用带宽。 在本指南涵盖的所有总线中，只有PCI Express为总线上的每个外设提供专用带宽。 GPIB、USB和LAN在所连接外设上共享带宽。 数据通过点对点连接传输，称为通道(lane)，第1代链路每个方向的带宽为250 Mbytes/s。 每个PCI Express链路可以由多个通道组成，因此PCI Express总线的带宽取决于在插槽和设备中的实现方式。 x1链路提供了250 Mbytes/s，x4链路提供1 Gbyte/s，x16链路提供4 Gbytes/s的专用带宽。 PCI Express实现了软件向后兼容性，意味着迁移到PCI Express标准的用户可以保留其在PCI的软件投资。 PCI Express也可通过外部布线扩展。

高速内部PC总线专为快速通信而设计。 因此，PCI Express是需要高带宽的高性能数据密集型系统以及集成和同步多种类型仪器的理想总线选择。

以太网/ LAN / LXI

以太网一直是一个仪器控制选项。 它是一种成熟的总线技术，已被广泛应用于外部测试和测量的许多应用领域。 100BASE-T以太网的理论最大带宽为12.5 Mbytes/s。 千兆以太网或1000BASE-T将最大带宽增加到125 Mbytes/s。 在所有情况下，以太网带宽在网络上共享。 125 Mbytes/s千兆以太网在理论上比高速USB快，但是当多个仪器和其他设备共享网络带宽时，该性能快速下降。 以太网总线通信是基于消息的，其中通信分组显着增加了数据传输的开销。 为此，以太网的延迟在本指南所述的总线技术中是最差的。

尽管如此，以太网仍然是创建分布式系统网络的强大选择。在有中继器的情况下，以太网的工作距离长达85米到100米，没有中继器时不存在距离限制。其他总线无法与控制PC或平台的分离范围。 与GPIB一样，以太网/ LAN无法进行自动配置。 您必须手动为仪器分配IP地址和子网配置。与USB和PCI一样，以太网/ LAN连接也普遍存在于现代PC中。 这使得以太网非常适合分布式系统和远程监测。 它通常与其他总线和平台技术结合使用来连接测量系统节点。 这些本地节点本身可以由基于GPIB、USB和PCI的测量系统组成。物理以太网连接的鲁棒性比USB连接高，但比GPIB或PXI低。

LAN对仪器的扩展(LXI)是一种基于LAN的新兴标准。 LXI标准定义了采用以太网连接的独立仪器的规格，增加了触发和同步特性。

尽管从概念上说，将某个总线或通信标准指定为最终或理想技术较为方便，但历史经验表明，几种替代标准可能继续共存，因为每种总线技术都有其独特的优点和缺点。 表4汇总了各种总线的性能标准。 应该清楚的是，没有一种总线在所有性能指标方面都是优越的。

	带宽 (MBYTES/S)	延迟(µS)	距离（米）（无扩展器）	设置和安装	连接器坚固性
GPIB	1.8 (488.1) 8 (HS488)	30	20	好	最好
USB	60 (USB 2.0)	模拟输出	5	最好	好
PCI (PXI)	132	0.7	内部PC总线	更好	更好 最好(用于PXI)
PCI EXPRESS (PXI EXPRESS)	250 (x1) 4,000 (x16)	0.7 (x1) 0.7 (x4)	内部PC总线	更好	更好 最好(用于PXI)
以太网/ LAN/LXI	12.5（快） 125 （千兆）	1,000（快） 1,000（千兆）	100米	好	好

表4. 总线性能比较

您可以通过创建混合测试和测量系统，将来自模块化仪器平台的组件（如PXI和独立仪器）连接到GPIB、USB和以太网/LAN，从而充分利用多个总线和平台的优势。 创建和维护混合系统的一个关键是实现一个能够透明地识别多种总线技术并利用开放的多供应商计算平台（例如PXI）的系统架构，以实现I/O连接。

成功开发混合系统的另一个关键是确保在驱动程序、应用程序和测试系统管理级别选择的软件是模块化的。 虽然一些供应商可能为特定仪器提供垂直软件解决方案，但最有用的系统架构是将软件功能分解为可互换的模块层，从而使系统不与特定硬件或特定供应商绑定。 这种分层方法提供了最佳的代码复用、模块化和使用寿命。 例如，虚拟仪器软件架构（VISA）是一种不依赖于供应商的软件标准，用于配置和编程基于GPIB、串行（RS232/485）、以太网、USB和/或IEEE 1394接口的仪器系统以及进行故障排除。 VISA是一个非常有用的工具，因为用于编程VISA函数的API对于各种通信接口都是类似的。

借助混合系统，您可以结合许多类型仪器的优势，包括传统设备和专用设备。 尽管为仪器找到一个通用型解决方案非常有吸引力，但现实需要您采用合适的仪器和相关总线技术来满足您的特定应用需求的。

定时和同步

PXI平台提供了仪器之间集成定时和同步的良好示例，PXI平台是用于测试和测量的模块化标准。PXI Express可维持10 MHz背板时钟以及原始PXI规范提供的单端PXI触发总线和长度匹配的PXI星形触发信号。PXI Express还为背板添加了一个100 MHz的差分时钟和差分星形触发器，以提供更高的抗噪能力和行业领先的同步精度（分别为250 ps和500 ps的模块间偏移）。 NI定时和同步模块旨在利用PXI和PXI Express机箱中的高级定时和触发技术。

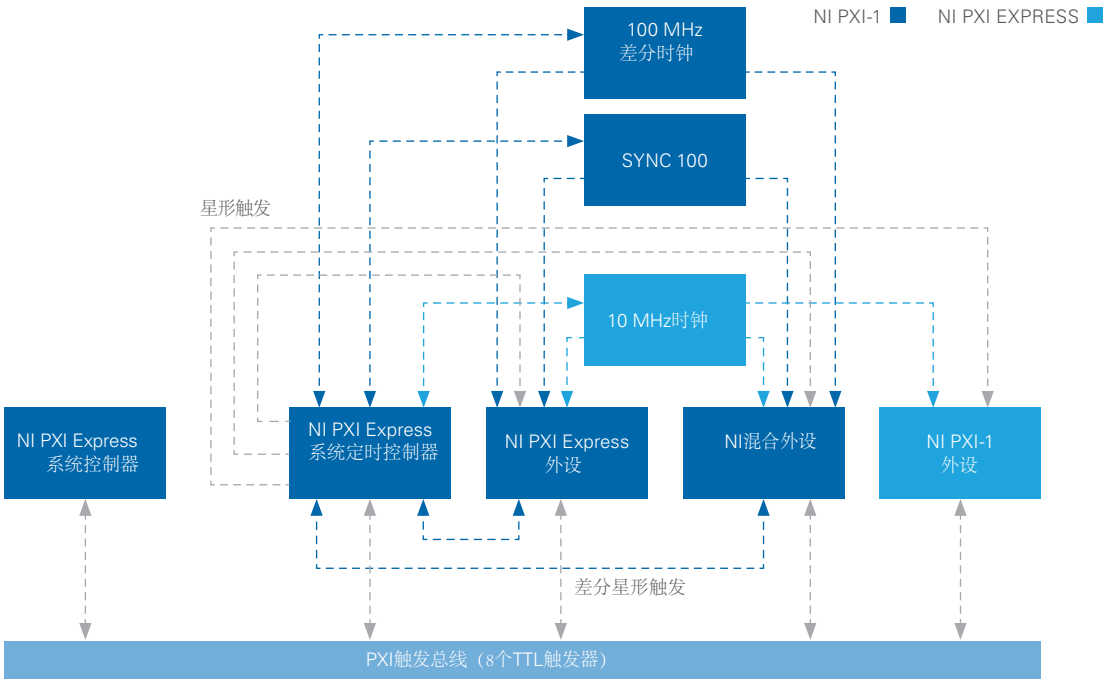


图8. PXI机箱定时和同步特性示例

下一步

通过阅读《仪器基础》白皮书系列，了解更多关于使用测试和测量仪器的基础知识。该系列包括从模拟采样理论到接地考虑因素等主题，旨在提高测量质量。

测试系统构建基础指南

自动化测试系统的电源

目录

引言

为自动测试系统或自动测试设备(ATE)供电不同于为桌面上的PC和灯供电。 测试系统由许多异构的内部组件组成，其中一些需要大电流和功率，并且这些系统通常需要部署到全球具有不同功率电源和质量的设施中。 许多测试系统组件来自多个供应商，并且必须由测试工程师集成，这使得问题更加复杂。 如果遵循电源布局和设备选择的最佳做法，选择正确的组件并做出正确的设计决策要简单得多。

电源布局可避免组件需要的功率可能要比电源可提供的功率要高这一瓶颈，从而确保所有组件正常运行。 这对于如果功率不足可能危及整个系统运行的组件来说尤其重要。 本指南通过列出创建电源布局的步骤和注意事项来介绍测试系统的电源规划。

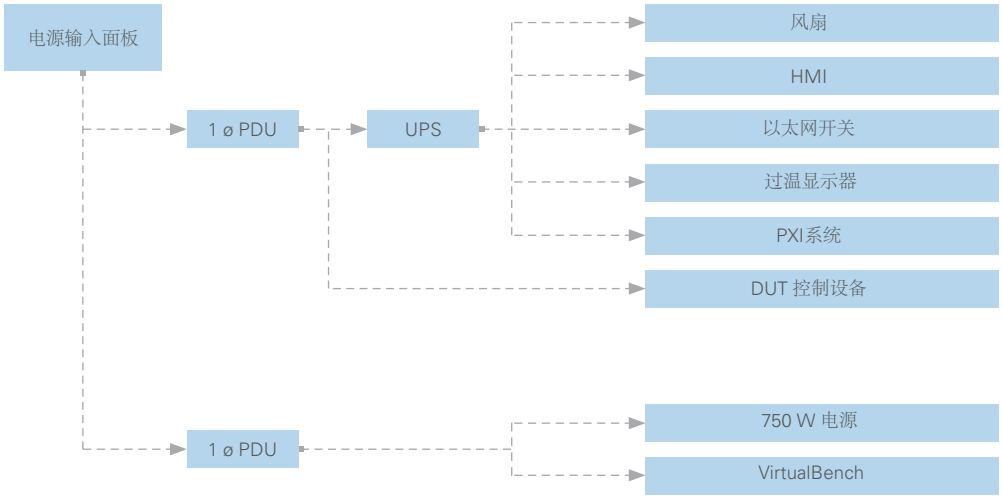


图1.电源布局包括测试系统中的所有设备，并绘制从电源到测试系统再到最终用户的电流流向图。

为系统添加电源

将电源添加到ATE系统的最佳做法是使用电源输入面板。 这允许您将内部电源电缆从施加主电压的点隔离出来。 使用电源输入面板，您可以为测试系统配备额定电压和电流与系统供电电压和电流一致的电源连接器。 NI电源输入面板使用多种连接器类型和额定功率来满足各种电源要求和地理位置要求。 图1显示了一个电源板连接器示例。 好的电源面板还应具有内置电路保护，包括断路器和保险丝，可保护系统免受电源浪涌或不正确电源的损害。 大功率面板具有内置电磁干扰(EMI)滤波器、浪涌抑制和其他将信号传递到系统的连接。

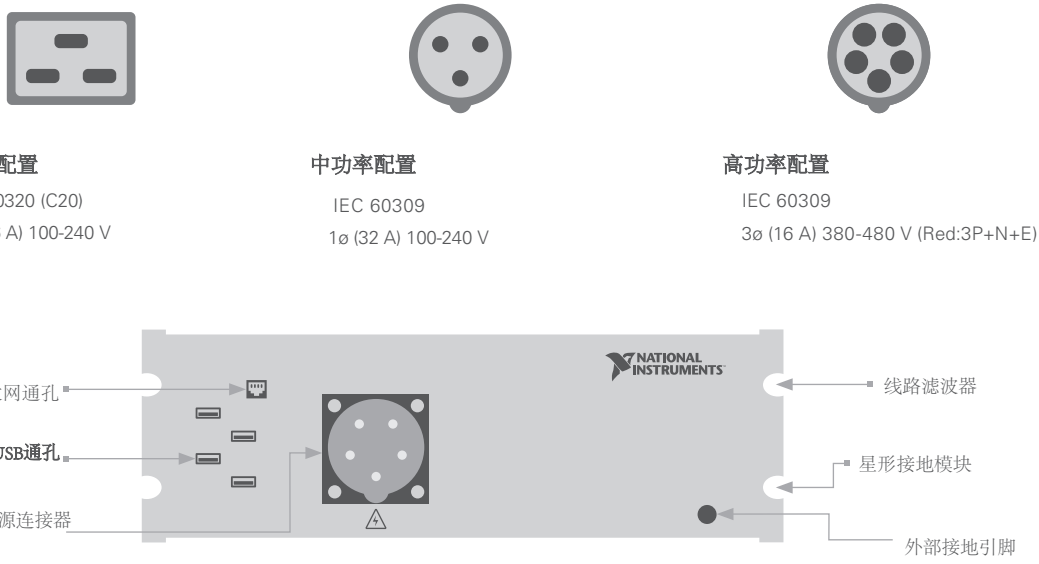


图2.电源输入面板为系统的电源输入提供连接。 电源输入面板可以仅提供其中一种标准电源连接器类型，并且好的电源面板应具有滤波或锁死开关继电器等额外特性。

地理位置注意事项

测试仪或测试设备的地理位置是为测试系统选择电源面板的关键细节。 此外，在规划新测试系统时，应考虑电力标准和电网基础设施、安全要求和易部署性，这些都是影响位置的因素

电网标准

从公共电网可获得的线路功率因国而异。 世界各国已经在其电网中对不同的有效电压等级、交流电频率、连接器和电流范围进行了标准化。 公共电网提供多种类型的电源配置：

- **单相电源**使用单根有源线来传导交流电和中性线。 这些线路的公共电压电平在100V到240V之间变化。例如，日本的线路功率为100V，而输送功率为在220V到240V之间。美国和加拿大的公共电网传输功率为110V-120 V。
- **双相或分相电源**由两条有源线组成，这两条有源线在给定的正负偏移电压和一条中性线下提供电源。在美国，双相功率通常采用120V，两条有源线之间相位差为180度。 使用两条电线来传输120 V和-120 V电压电平时，两条有源线可分别与一条中性线结合，组成两个120 V电位的单相电源，或同时使用两条有源线，组成一个240 V电位的单相电源。
- **三相电源**由三条相位相差120度的有源线和一条中性线组成。 大多数美国建筑物使用208Y/120V电源，通过三条线路来传导120V电和208V恒定功率电路输出。许多工业建筑使用480Y/277V，为大型机器提供480V电压。

全球部署

测试系统通常设计和部署在独立或多个位置上。将一个系统部署到多个位置会给系统带来一系列新要求。将系统部署到马来西亚不同于将系统部署到同一个国家甚至同一栋大楼的工厂。例如，底特律的研发部门为汽车发动机控制单元开发了测试系统后，需要将其部署到墨西哥的工厂。在设计系统时考虑电网标准和质量，并确保在运送系统之前满足在墨西哥部署所需的所有安全和监管认证。 以下是设计将在全球部署的测试系统时要考虑的事项列表：

- 电网电压标准和配置
- 电网质量和可靠性
- 材料合规性，如RoHS
- 能源合规性，如CE、PSE或KC
- 贸易合规性和进出口管制

如果计划将测试系统部署到测试系统原始开发国之外的国家或地区，请了解测试系统要部署的位置的可用功率，以及是否需要转换该功率才能运行测试系统的设备。 在上面的例子中，测试系统要部署到马来西亚和墨西哥。 幸运的是，美国和墨西哥的电网均提供110V-120V和60Hz的功率。 如果测试系统是在德国设计，要部署到主电压完全不同的墨西哥，情况可能会更复杂。

电源转换器和不间断电源(UPS)可帮助您调节标准电源以满足系统的需要。 例如，内含设备仅接受120V电压的测试系统可能需要纳入功率转换器，将230V单相功率转换为仪器所需的单相120V电源。更好的做法是，评估和选择具有全球输入电压的设备，以避免这些麻烦。

认证

许多国家都有特定的电气安全标准，如欧洲CE、日本PSE或韩国KC。 电气测试设备的合规性测试通常包括排放水平和频率、触摸安全和浪涌保护。 获得这些认证标志的最重要理由是能够将系统部署到其他国家或者证明设备能够在工厂运行。 进行必要的研究，以了解测试系统将运行的每个国家所需的认证。 忽略认证可能会导致将来使用测试系统时出现问题。 只有拥有这些认证标记的组件才能进口，因此如果没有进行适当的认证，零件将很难更换或维修。

图3.在多个国家设计和部署测试系统需要灵活的系统设计。 在开发需要在不同地理位置使用的测试系统时，请务必考虑电源标准和认证。

电磁干扰或线路滤波器

电网的高能量信号会辐射电磁噪声。大多数电源线的噪声相对一致，因此您可以提前计划。然而，没有电网是完美的，在电源信号中很可能存在一些非标准噪声。非标准噪声可能影响系统中的仪器测量，或导致系统违反认证要求。 为了保护测试系统免受来自于输电线路的意外噪声源的影响，最常见的方法是EMI和线路滤波器。线路滤波器必须在一定电压电平和电流电平下工作，滤波的信号也有一定的频率范围。例如，线路滤波器的最大电压电平和电流电平为250V和10A，频率范围为150kHz到1MHz。确保根据测试系统的功率选择正确的线路滤波器来滤除不需要的噪声频率。 NI电源输入面板包括用于保护敏感测量设备的EMI /线路滤波器。

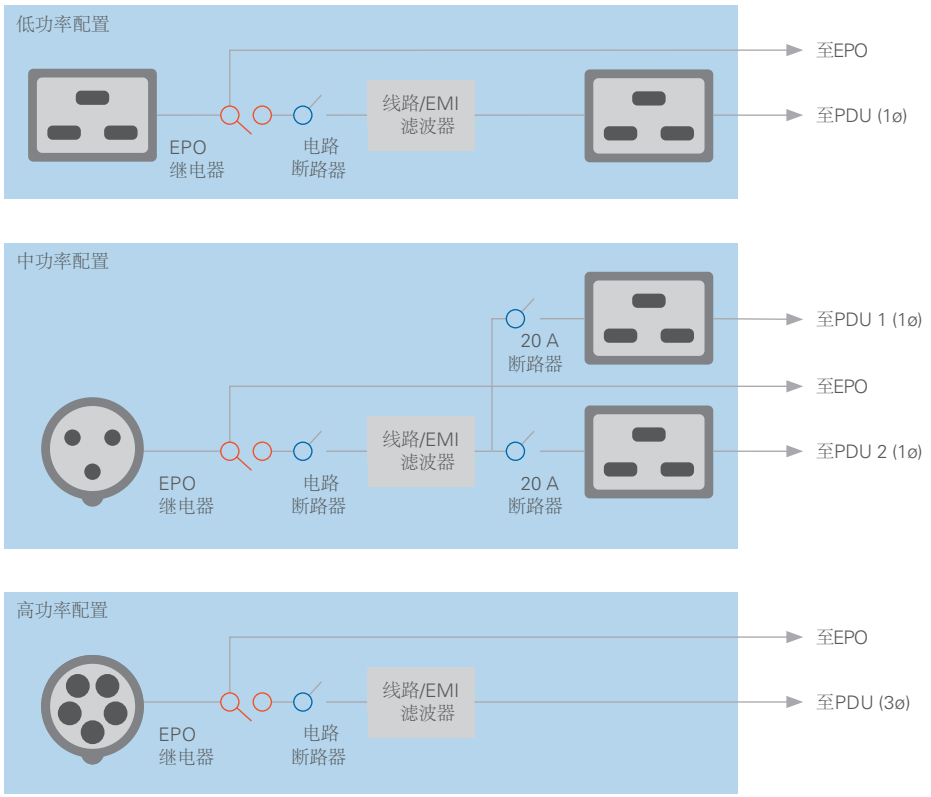


图4.电路断路器和线路/EMI滤波器对于保护测试系统中的设备以及确保仪器的正确和准确性能至关重要。图中显示了低功率、中功率和高功率配置的电源输入面板示例。

功率预算

功率预算是为测试系统规划资源和组件的关键部分。 给定的设备必须能够以正确的电压电平访问一定量的电流。 必须为整个系统以及系统内配电的每个点进行功率预算。 在确定所需的功率量后，可以对计算出的值应用几个标准规则，以正确调整测试系统中的功率分配。

系统功率预算

系统功率预算的第一个要素是确定测试系统中所有设备的最大功率需求。 总和应包含测试系统中所有组件的预期功耗，包括电压，电流和功率。 在许多情况下，功率预算的最重要部分是电流。 只有一定量的电流可以流过系统中的特定传输线，因此通常必须使用配电单元(PDU)来仔细地分配整个系统的电流。

给定设备的功耗通常在用户手册中写明，并且有时提供了在不同条件下的多个功率需求。 有时，设备标注了典型的功耗和最大或最坏情况下的功耗规范。 为了安全起见，最好的做法是使用最大功率要求，然后减去给定的百分比，通常为30%至40%，以更符合实际情况。 图5显示了独立式仪器集成到测试系统的最大功率需求。

电源要求

⚠ **注意：**如果没有按照《NI VB-8034 安全、环境、法规信息》文档描述的方法使用， VirtualBench 硬件提供的保护功能可能会受损。

电压输入范围	100 VAC - 240 VAC，50/60 Hz
功耗	最高150 W
电源输入连接器	IEC C13电源连接器
电源断开	交流电源线提供主电源断开。放置设备时应避免电源线不易断开。按下前面板电源按钮不会阻断内部电源。

图5. VirtualBench多功能一体式仪器提供了高功率以及典型或平均功率下需要的最大功率。

举一个快速简单的例子，如表1的测试系统。首先了解测试系统中每台设备的最大功耗。 确保考虑子系统和瓶颈。 PDU具有最大电流限制 - 在本例中为16 A –因此您应该据此进行规划。下一步是针对所需的典型功率（而不是最大功率）校正这些值。这意味着大约最大功率的60%到70%。 在本例中下，保险起见我们使用70%，则该测试系统提供了约1,920 W的功率。增加约20%的最大功率值以便将来扩展系统或添加新功能之用，而不必再添加电源。

设备	最大功耗	平均功耗利用	电流@110 V
PDU 1			
风扇	50 W	35 W	.03 A
HMI	100 W	70 W	.06 A
以太网开关	25 W	17.5 W	.02 A
过热显示器	10 W	7 W	.01 A
PXI系统	526.9 W	369 W	3.4 A
DUT控制泵	1,000 W	700 W	6.4 A
PDU 1 总计		**1,198.5 W**	**11.0 A**
PDU 2			
VirtualBench	150 W	105 W	1.0 A
750 W电源	1,100 W	770 W	7.0 A
PDU 2 Total		**875 W**	**8.0 A**
系统总计		**2,073.5 W**	**19.0 A**

表1.一开始先了解所有系统组件的最大功耗，乘以平均功率利用系数并将它们相加在一起计算功率预算。 记住考虑瓶颈和子系统。

三个简单的最佳实践可以显着简化功耗预算：

- 1. 系统的功率需求为每个组件最大规定功率的大约60%至70%计算。
- 2. 作为安全缓冲，在第一点的基础上增加约20%作为最终功率计算，以考虑高活动时段和测试系统任何必要的未来扩展。
- 3. 记住一些组件通过PDU和UPS进行连接，因此大型系统需要考虑电源子系统。

子系统功率预算

上述功率预算计算中不包含的步骤是如何考虑大型测试机架内的子系统。 子系统可以是更大型测试系统中的任何设备子集，全部共用一个公共电源，比如使用单组PDU的多个仪器或PXI等模块化仪器系统。

模块化仪器的一个优势是它可以简化电源管理。如果PXI机箱中包含的所有仪器在测试系统中都是独立的，则必须单独考虑每个仪器。PXI机箱为机箱中的所有仪器提供高品质和安全的电源，并提供多个电源和仪器插槽选项。在将PXI系统添加到功率预算时，有以下两个选择：

- 1. 使用PXI机箱规定的整个PXI系统的最大功耗。 例如，PXIe-1085 PXI机箱的最大功耗为791 W，在乘以70%的平均利用率后便得到554 W的功耗预算。
- 2. 添加PXI系统中所有模块的最大功耗，以获得非常精确的功耗预算值。 有关执行详细PXI系统功率预算的示例，请参见图6。

另外，模块化仪器系统明显地比传统的仪器组合更高效，因为它避免了测试系统内需要安装冗余的监视器和冷却系统等由测试系统内部供电的共用组件。

作为PXI系统精确功率预算的一个例子，可考虑具有24 GB/s系统吞吐量的PXIe-1085 PXI机箱，包括PXIe-8880 PXI控制器、六个PXIe-4139精密系统源测量单元(SMU)、两个 PXIe-5162 PXI示波器、PXIe-6570基于向量的数字通道板卡、两个PXIe-4081 7½位数字万用表(DMM)和四个PXIe-2527多路开关模块。 请参见图6中PXI系统功率预算的计算方法。

图6. PXI机箱的总功耗是机箱中所有模块的总和。可以看到在最坏情况下，机箱中的所有仪器将消耗526.9 W.

配电单元

PDU的主要用途是获取输入功率信号并将其分配给为系统组件供电的多个输出。 PDU的内部电源插座具有一定的额定电压和电流，通常可用于交流和直流电。 最好的PDU选项具有多个特性：

- 远程开/关使操作员能够使用动力机构和EPO更改电源状态。借助这种方式，操作者可以从方便的位置完全控制系统状态。 操作员还可以通过本地和全局EPO机制禁用系统的电源。
- 内置电路保护，如保险丝，可以保护昂贵但易坏的设备免受意外电力事件的影响，节省数万甚至数十万美元
- 设备组排序可以确保特定设备在其他设备组上电之前先上电。 例如，连接到外部控制器或从另一个主PXI机箱扩展出来的PXI机箱需要在主机控制器之前先启动。 在这种情况下，在启动包括PXI主机箱的设备组之前，PDU应启用包含PXI从机箱的插座组。
- 通过多个组来处理一定量的功率有助于平衡PDU上的功率负载，防止出现可能损坏测试系统设备的过流条件。 例如，具有三个电源插座组的PDU可以为每个组输送16 A的电流，防止连接到PDU的任何一台设备的电流超过16A。 这也意味着您必须知道在多个组之间分配给设备的电流。
- 直流电源可以为状态LED或冷却系统之类的组件供电，这些系统通过远程开/关和组排序消耗直流电。 一些组件甚至用于系统的“电源启动”状态，从而需要远程供电功能。

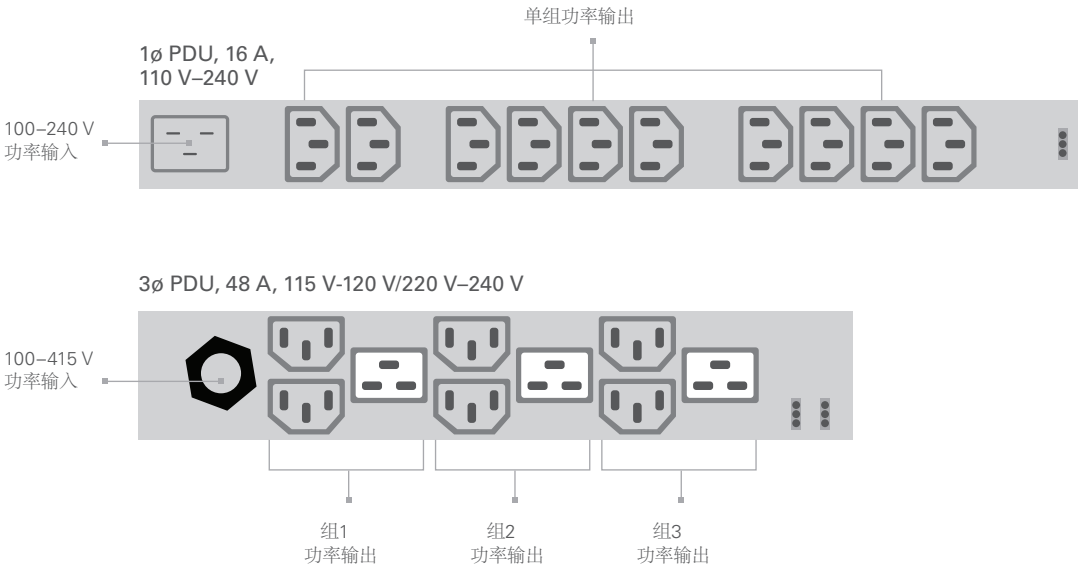


图7. PDU可采用不同的接线方式和架构。 上面的PDU只有一个组，可以为设备提供高达16 A的电流，下面的PDU具有三个组，每个组可以提供16A电流，最高可达48A。

为测试系统中的关键组件供电

请确保关键组件（如主机控制器和测试系统中的敏感仪表）通过UPS供电。 测试系统的一些组件乍看起来没那么重要，但其实很重要。 如果冷却系统在断电或掉电后没有继续运行，则主机控制器可能会过热。 如果测试系统的触摸屏显示器没电，则技术人员无法进行故障排除或对电源故障时间进行数据记录。想一下即使在停电或紧急情况也要继续操作的事项。

系统供电开销和支持

在分配电源时记得考虑测试系统的开销和基础设施，如温度控制、网络连接和用户界面元素。 过热或网络连接断开可能会导致生产中断，这与测试仪器故障同样具有危害性。

不间断电源

好的测试系统设计工程师会考虑电网的质量并设计系统，以避免在掉电和停电期间出现未定义行为。 您可以在这些事件期间以及有时在正常操作期间使用UPS为测试系统中的关键组件供电。

UPS可以提供具有可靠电压和电流的电力供应。 它也可以在断电或严重掉电后充当电池电源。 UPS是构成坚固耐用测试系统的关键部件，特别是对于电网不可靠的位置。

UPS主要有两种类型：

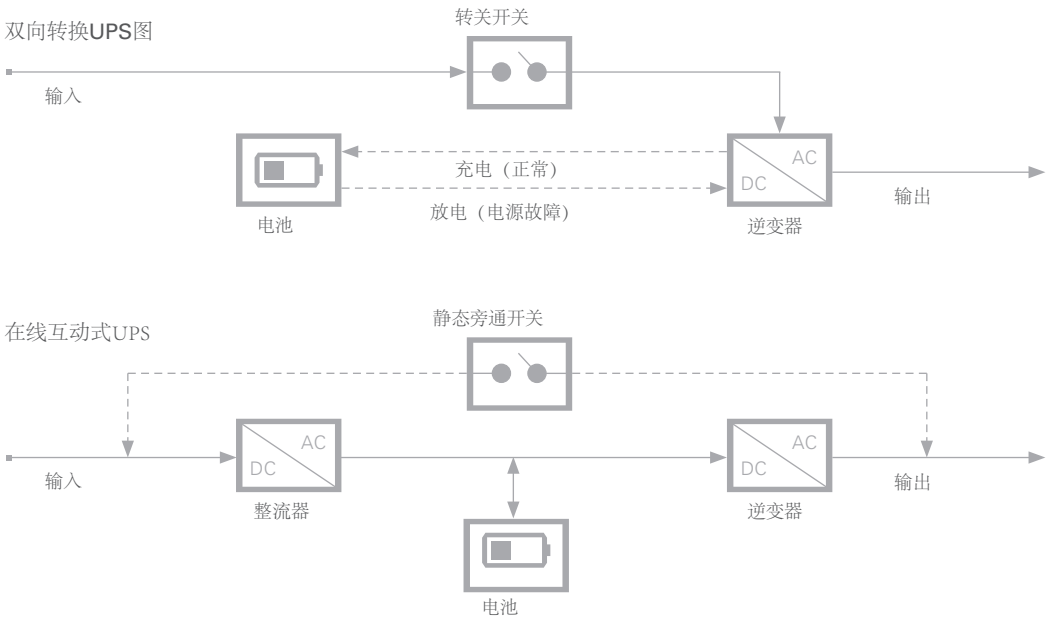


图8. UPS用于提供清洁、可靠的电源，也允许在停电或掉电的情况下正常关机。 双向转换UPS始终为电池充电，为系统提供连续的电源。

- **在线互动式UPS** – 在在线互动式UPS中，有功电源线输入直接连接到电源输出。然后，UPS监测输入功率，以确保功率不会低于给定阈值。如果电源线电压下降过低，就会切换到通过反向运行UPS为输出信号供电的充电电池。 在这种情况下，测试系统在运行期间接收线路功率而没有经过任何调理，并且在电源故障的情况下由UPS继续供电。
- **双向转换UPS** - 双向转换UPS将输入电源线连接到连续充电的电池，然后为UPS的输出线路提供电源。 双向转换UPS的供电非常连续，因为它通过板载电池供电。 双向转换UPS的另一个好处是，电池总是充满电，可以随时用作为备用电源，以允许关键系统在停电或显着掉电时正常关闭，无需切换电源。 尽管这些UPS的效率略低，但它们却能够在测试系统内部始终提供稳定和准确的电源，这使得双向转换UPS成为ATE应用的不错选择。

电源质量和可靠性

没有电网是完美的，但大多数电气设备设计为在理想的电源条件下工作。 当电网的功率与系统使用的功率不同时，系统就会不正常运行。 仪器可能进行质量很差的测量或输出错误信号。 设备和系统可能会打开后关闭，导致丢失重要设置或默认设置不正确。 这种意外的行为可能导致测试结果不理想、待测设备(DUT)受损甚至更糟糕。 双向转换UPS的一个额外优点是通过给内部电池充电来提供滤波，并提供高度可靠、清洁的电源。

停电和断电

当电网供电完全关闭时，就会发生停电。设计良好的电网很少会出现停电，当停电时，通过两种方式管理系统的行为：（1）使系统的一些或所有组件短时间使用电池供电，以便能够正常关闭或（2）由于停电而关闭。

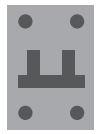
掉电和电涌在电网中更为常见，特别是在高功耗的工厂设施中，并且更难以处理，因为它们会在系统中引起不确定的行为。 掉电可以是电网中的电压或电流骤降或毛刺，会导致输入到测试系统的功率降低。 浪涌是电网的电压或电流高于正常水平的瞬时干扰。

UPS的内部电池可允许在停电或严重降压之后新电源（比如发电机）联机之前的那段时间里，为测试系统的重要设备提供足够的电力。 重要设备包括主机控制器和用户界面以及任何其他关键设备。 电池提供的时间可允许系统维护基本数据，避免系统损坏或不安全的软件状态。

电源状态

测试系统通常需要具有多个运行状态来允许调试和维护、省电和安全。好的测试系统设计方法应该能够实现四种运行状态：

- **关闭** - 系统完全禁用，没有电源通过线路滤波器或任何内部测试系统组件。
- **启用** – 电力通过线路滤波器并进入任何直接供电的设备。 通常，所有设备都通过PDU供电。 在启用状态下，只有PDU的主输出可能被激活。 在某些情况下，PDU的直流电源也会被激活，为系统支持和其他组件供电。 例如，在启用状态下，以太网路由器和实时系统控制器可以通电，以便技术人员可以监测测试系统的健康。
- **打开** - 该状态的改变从测试系统的主电源接通序列开始。 所有PDU接收电力并输出到其他系统设备。 在许多情况下，当某些系统组件必须在其他组件启动后才能开始运行时，有必要对电源序列分级。阅读“电源布局”部分，了解有关PDU的更多信息。
- **紧急关机(EPO)** - 当用户或系统显示器识别到不可接受的运行条件时，EPO会立即切断测试系统的电源。



- 系统电源禁用
- 没有设施为线路波波器供电

- 系统电源启用
- PDU主交流出口启用
- PDU直流插座启用（可独立控制）
- PDU标准插座启用
- UPS禁用

- 系统电源启用
- PDU主插座启用（直流电源—风扇、系统扩展器和ENET路由器）
  –各个DC插座启用（2组上电序列）
- PDU准插座启用（2组上电序列）
- UPS启用

- 系统电源禁用
- 没有设施为线路波波器供电

图9.测试系统需要多种电源状态，包括关闭、启用、打开和EPO，以确保高效的运行。

紧急关机

当测试系统遇到严重问题或设施发生紧急情况时，操作员需要能够快速利落地关闭测试系统。EPO机制包含在测试系统中，以简化连接并禁止功率切换。操作员可在错误状态下使用EPO来重置系统，防止对DUT造成损坏，甚至防止对自己造成伤害。EPO功能也是IEC和UL等安全标准机构的要求。

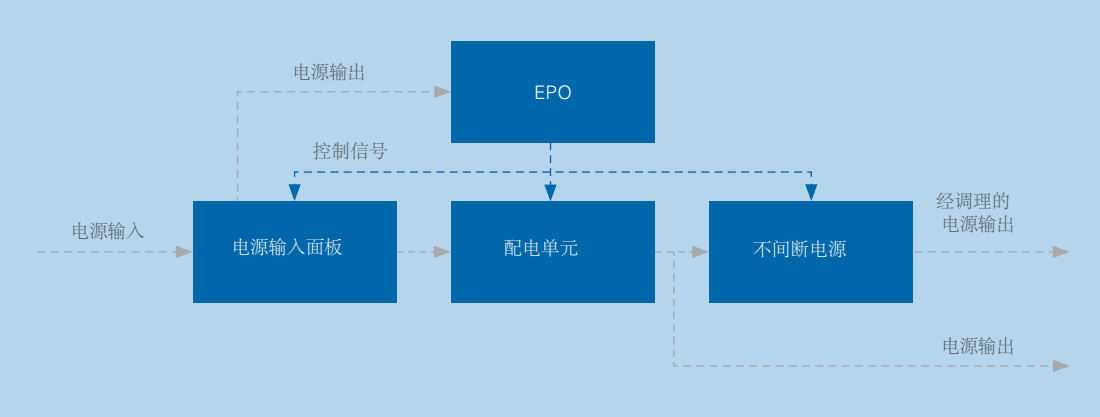


图10. EPO连接到测试系统中的所有设备，并在必要时可以禁用所有连接的设备以保持安全。

EPO通常是操作员容易操作的的物理机制，比如按钮或开关，并且在按下时切断对所有测试系统设备的供电。 理想情况下，EPO面板与测试系统的所有设备都连接，以确保所有设备能够快速关闭。 大多数EPO将系统置于关闭状态后，需要将它们重置为启用状态，然后才能重新激活并启动所有设备。 这可防止系统在出现不安全情况时断电后又意外重启。

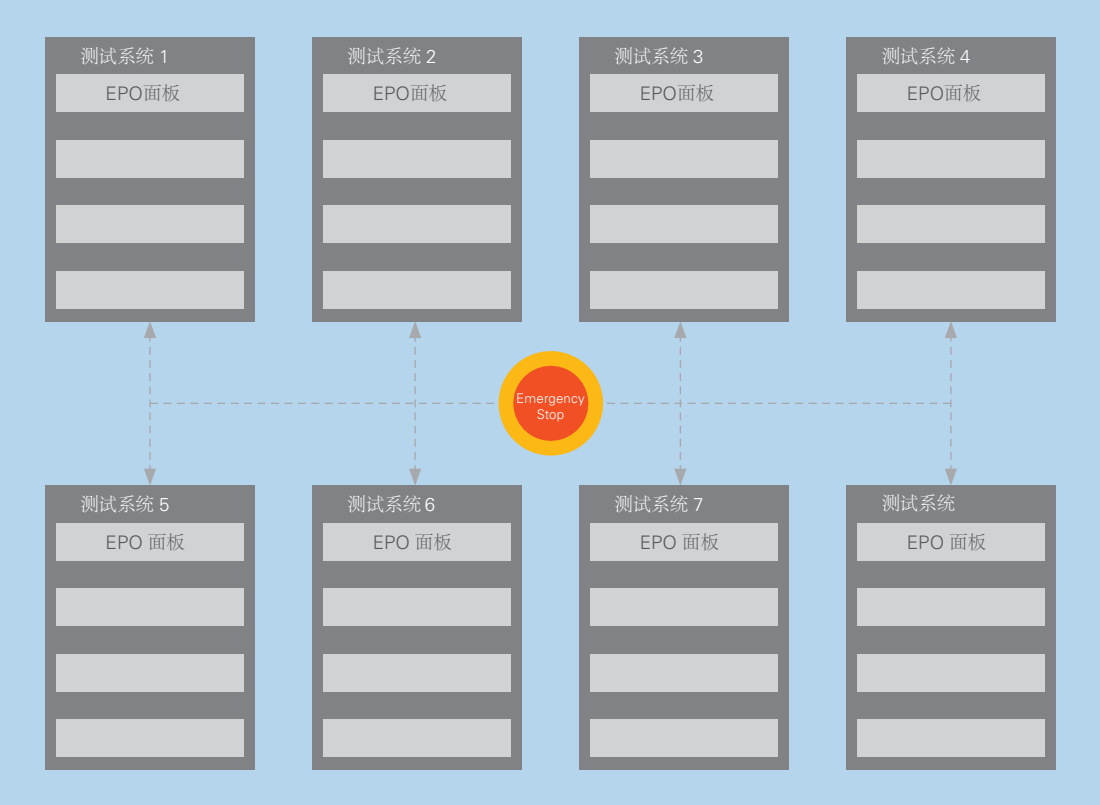


图11.在某些情况下，需要全局EPO来禁用设施中的所有测试系统和设备。 全局EPO是一个单一的关机机制，可启用各个系统的本地EPO。

接地

接地是测试系统设计的关键部分，主要原因有两个：安全性和测量质量。

确保测试系统具有适当的接地以保证安全性，这意味着给测试系统的所有设备提供适当的路径，以便电流流向实际接地或地面接地。 电源输入面板必须连接到具有正确接地的电源。 然后可以选择测试系统中的任意一台设备作为终端电源，并沿着自身的路径将其接地回电源输入面板。 以太网开关的接地电流路径应遵循图1示例测试系统的电源布局。以太网开关接地连接到UPS地，USP地应连接到PDU的地，PUD的地应连接到电源输入面板的地。接地回路会形成一条电流路径流向地面，可帮助您避免在系统中创建可能产生放电的危险充电，导致DUT损坏或操作员触电。

尽管所描述的接地路径通常足够，但是测试系统中的每个设备单独接地可以保证安全。 NI的电源输入面板有一个星形接地模块，如图2所示，连接到整个机架上的其他接地模块。 然后，电源输入面板外部的接地螺柱可以连接到机箱外部的真实接地线。 另外，每件设备通常具有可以直接连接到地面的接地螺柱。 您可以看到图12中的NI PXI机箱的接地螺钉。将每个设备连接到整个机箱中的分布式接地模块，确保每个设备都安全接地，并且所有接地线都非常短，这有助于减少电磁噪声 。

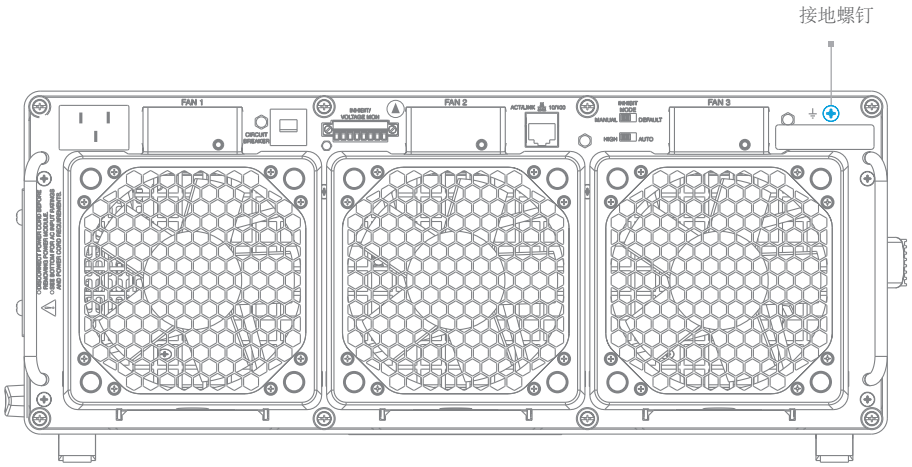


图12. PXIe-1085 PXI机箱有一个接地螺钉，可允许您将机箱和所有仪器直接接地到外部接地模块。 将机架中的每台设备接地是确保安全的最佳做法。

确保接地平面的电气连接短路。 长的接地回路可能会造成驻波，导致系统内出现射频辐射。 如果需要长传输线连接到接地平面，则需将信号与双绞线配置中的接地信号耦合，以减少电磁噪声。 如果是浮地或不参考地，则应纳入信号的正负参考。

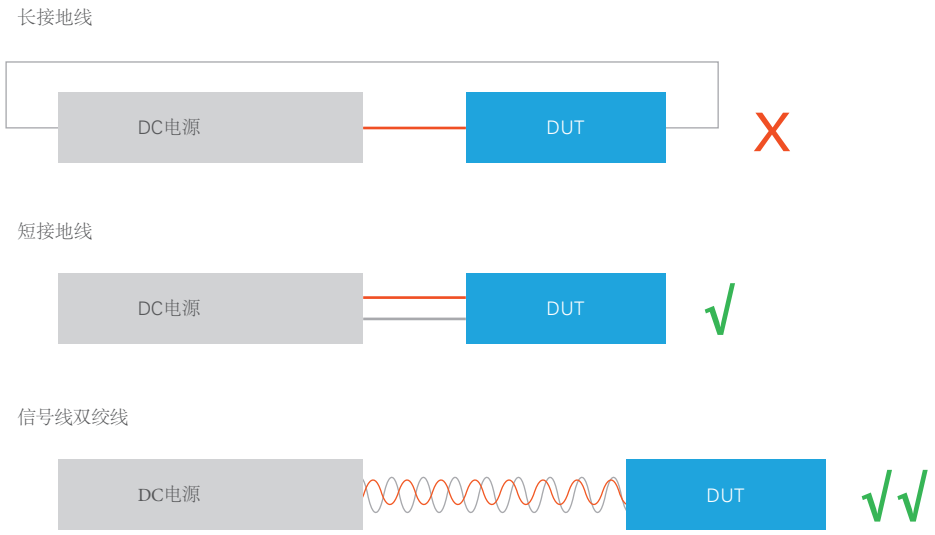


图13.系统中的长接地线如果不匹配，可能会导致明显的接地回路，并作为噪声信号的天线。 使用短接地线更好，但仍有可能接收不需要的噪声。 为了获得最佳性能，请在系统中使用双绞线的信号线和接地线。

通过阅读白皮书“现场接线和噪声注意事项综合指南“，了解为测量做出正确连接所需的所有信息。

组件的最佳实践

采购材料来构建测试系统几乎有无限种方式。 当构建需要维护的系统时，请考虑长期支持和系统的可扩展性以便在未来添加需求。 为了实现这些结果，最好向具有长期产品供应策略的商业供应商采购系统组件。 对于PDU、UPS、系统控制器和仪表等项目，与供应商合作似乎是常识，但同样的策略应用在较小的项目（如互连和电缆）可以获得长期回报。与专注于提供连接器的供应商和提供测试仪器的供应商合作，可帮助您以经济高效的方式让系统运行十年之久。

在极少情况下，由于特殊要求或特殊情况而无法使用商用产品，因而有许多公司专注于为测试系统定制设备和解决方案。 请记住，这些解决方案通常针对个别消费者，并且随着时间的推移更有可能改变或过时。

测试系统构建基础知识

开关和多路复用

目录

引言

许多自动测试应用需要将信号路由到各种待测仪器和待测设备(DUT)。通常，解决这些应用的最佳方式是部署开关网络来实现仪器和设备之间的信号路由。开关不仅可实现这种信号路由，而且也是一种经济有效的方式来增加昂贵仪器的通道数，同时增加测量的灵活性和重复性。

为自动测试系统添加开关有三种主要方法：自行设计和构建开关网络；使用基于GPIB或以太网控制的独立开关盒；或使用包含一个或多个仪器（例如数字万用表(DMM)）的模块化平台。开关基本上必须与其他仪器结合使用，因此通常需要与这些仪器的紧密集成。现成的模块化方法可以满足大多数常见测试系统中固有的这些集成挑战。本指南概述了将开关和多路复用器集成到测试系统的最佳做法。

开关架构

开关是扩展仪器通道数的一种经济高效的选项，但并不总是最佳选择。 开关架构有四种主要类型：

1. 无开关
2. 仅在测试机架中采用开关
3. 仅在测试连接件中采用开关
4. 在测试机架和测试连接件中采用开关

下表列出了所有四种开关架构的优点和缺点。

低 ○ 平均 ◒ 高 ●

	灵活性	吞吐量	成本	基本测量 （mV、μA、mΩ）
无开关	○	●	○	●
仅在测试机架中采用开关	●	◒	●	○
仅在测试连接件中采用开关	○	◒	◒	◒
在测试机架和测试连接件中采用开关	●	◒	◒	◒

表1.各种开关架构的优点和缺点

无开关

在第一种架构中，待测设备(DUT)与测试系统仪器之间的信号路由没有使用任何开关。 这样的系统通常具有专用于每个测试点的单个仪器通道。

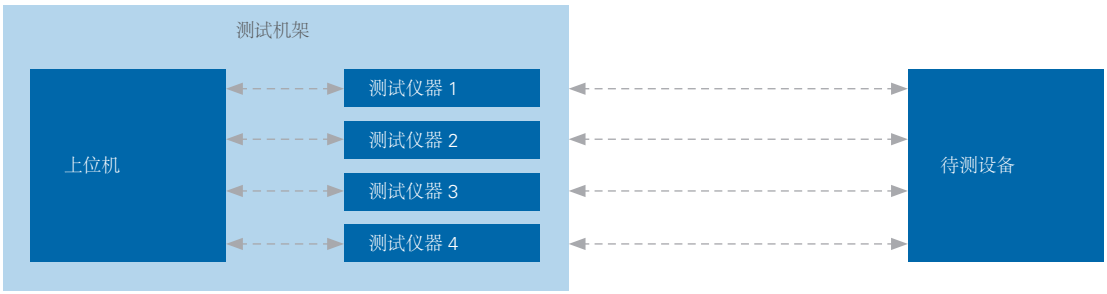


图1. 无开关测试系统直接将每个仪器连接到待测设备。

无开关架构的优点

电缆和开关经常会降低信号的完整性。 如果不使用开关，则可以为测量仪器提供更直接的路径，从而提高测量准确性。 除了提高测量准确性外，您还可以实现更快的测试速度。 通过为每个测试点提供专用仪器，您可以进行并行测量而不是顺序测量，从而提高测试吞吐量。

无开关架构的缺点

为每个测试点提供专用仪器成本非常高。 另一个缺点是可扩展性。 如果在不使用开关的情况下构建测试系统，测试机架的空间很容易就耗尽。 这可能需要完全重新设计测试系统，导致额外的硬件变更、软件更新和重新验证成本。 例如，假设测试系统使用20个PXIe-4081数字万用表(DMM)并行测试20个电阻温度检测器(RTD)传感器。 而且系统需要扩展到测试40个RTD传感器。 这时需要添加20个DMM，从而需要20个额外PXI插槽。 另一种方法是，可以使用单个PXIe-40817½ 位DMM以及一个开关，按顺序测试所有40个RTD传感器，这只需要两个PXI插槽。

什么情况下采用无开关架构

对于添加电缆和开关会导致信号失真的超级敏感型测量，或者需要将测试时间降至最低的应用，则通常建议采用无开关架构来构建测试系统。 例如，一些半导体测试应用具有专用于芯片上每个引脚的单个参数测量单元或源测量单元，因为半导体是大产量行业，测试成本通常占芯片总制造成本的很大一部分 。 使用专用仪器通道进行并行测量，可以最大程度地减少测试时间，从而显着降低测试成本。 此外，在半导体工业中，测试装置通常针对特定的芯片组或芯片组系列而开发，因此在整个生命周期中通常无法进行扩展。

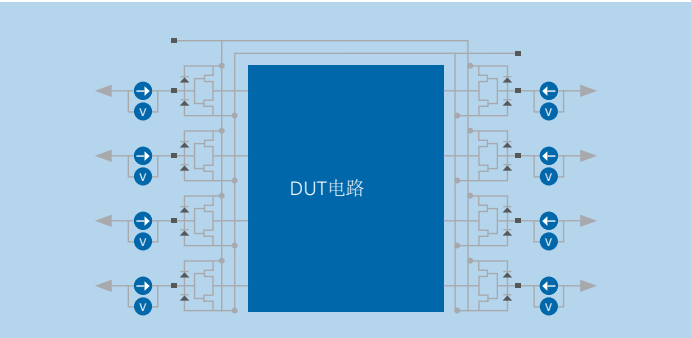


图2. 部分半导体应用使用专用仪器来并行测试给定芯片的所有引脚。

仅在测试机架中采用开关

第二种开关架构仅使用商用现成(COTS)开关在测量仪器和测量仪之间路由信号。 在测试机架中采用开关提供了一种方式来利用现成开关产品，并提供了最简单的扩展路径。 选择一个可提供广泛功能且易于扩展的COTS开关平台非常重要。否则会由于需要重新设计测试系统而导致更高的支出。

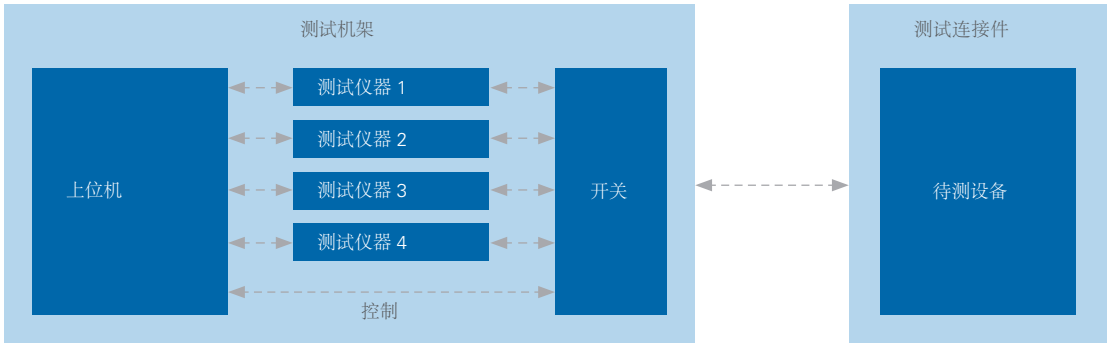


图3.部分测试系统在测试机架内集成了开关，便于扩展。

例如，PXI平台提供超过600种不同类型的开关模块，可以路由高达600 V、40 A和40 GHz的信号。 NI还提供100种不同的PXI开关模块，使您可以在200多个不同的开关拓扑中进行配置。

在测试机架中采用开关的优点

使用COTS开关解决方案，可以大大节省开发时间，包括印刷电路板(PCB)设计和驱动程序开发。 此外，COTS开关提高了测试系统的可扩展性，因为只需向开关供应商购买模块即可添加更多开关，而不需要重新设计整个测试连接件。

此外，每个开关供应商的解决方案都具有其独特的优势。例如，NI开关具有板载EEPROM，可跟踪模块上每个继电器被激活的例程数量，同时还提供其他功能来监测继电器的健康，例如功能和电阻继电器测试。 通过这些功能，您可以预测特定继电器何时达到其机械寿命的终点，从而进行预测性维护。 这些功能非常适用于维护难以手动调试的高通道数开关系统，或者在意外停机可能导致延迟成本高昂的制造生产车间。

使用NI开关模块时，可以将开关连接列表下载到开关模块上的内存，并通过开关和任意仪器之间的双向触发在列表中循环，而不会中断主机处理器，从而提高测试应用程序的吞吐量。

在测试机架中采用开关的缺点

正如前面所说的，使用开关可能会减慢测试进程，因为这需要按顺序测量任何给定DUT的测试点，而不是并行测量。 将所有开关放置在测试机架内还会增加布线的总量。 除了在开关和测量仪器之间使用电缆，DUT和开关之间也需要电缆。 这可能导致敏感型测量发生误差，例如漏电流或低电阻测量。

仅在测试连接件中采用开关

第三种开关架构仅在测试连接件中使用开关。 在这种情况下，您可使用靠近连接件或位于连接件内部的PCB上的单个继电器将来自测量仪器的信号路由到DUT上的各个测试点。 采用此架构需要在测试站中安装一个继电器驱动器，以控制测试程序中的各个继电器。COTS继电器驱动器的一个典型例子是PXI-2567，这是一个64通道继电器驱动器模块，允许您使用NI-SWITCH驱动程序和标准API来控制外部继电器，而无需自定义编程。 或者，您可以设计一个外部电路来驱动继电器，但这需要额外的设计工作。

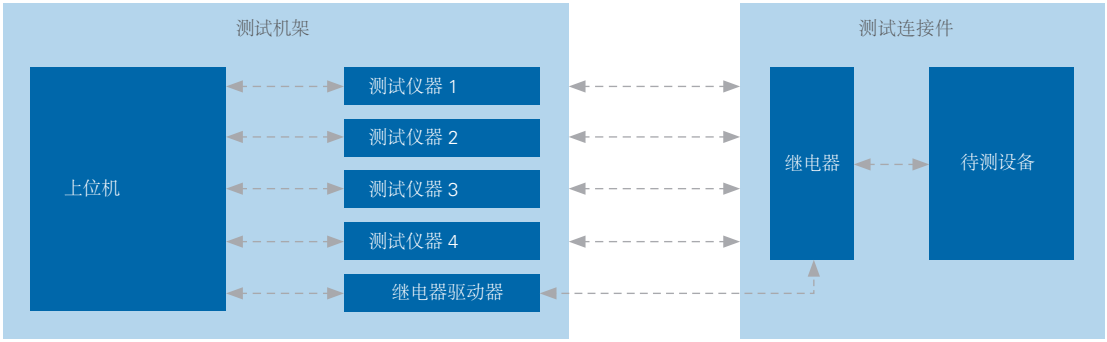


图4. 在测试连接件中采用开关的测试系统需要一个继电器驱动器。

在测试连接件中采用开关的优点

如上面所述，开关有助于降低测试成本，无论放置于哪个位置。 此外，在测试连接件中接入开关可避免在待测设备和开关之间连接电缆。 减少布线也有助于减少测量误差。

在测试连接件中采用开关的缺点

如上面所述，使用开关可能会减慢测试进程。 此外，在测试连接件中接入定制开关需要具备PCB设计经验，因此并不是任何人都可以选择这一选项。 在测试连接件中接入开关也会使测试系统难以扩展，无法满足更多测试点的需求。

此选项的另一个缺点是需要为特定安全和合规标准设计定制电路板而产生更多成本。 如果您正在测试高压设备，则可能需要定制一个符合各种规定（如UL、CE和VDE）的开关连接件。 所设计的PCB的继电器必须满足这些标准的爬电距离和间隙要求，这是一项具有挑战性的工作。 在这种情况下，使用COTS开关有助于降低成本。 许多COTS供应商都会根据各种安全标准对其模块进行认证。 例如，所有具有大于60 VDC或30 VAC和42.2 Vpk额定电压的NI开关模块被认为是高压设备，因而是根据以下安全标准进行开发的。

图 5. NI开关模块可满足各种安全和合规标准

在测试连接件和测试机架中使用开关

最后一个开关架构包含了测试站中的开关以及测试连接件。 使用这一方法，您可以利用两种COTS开关解决方案的优点，同时还可将开关放置在靠近测试连接件的位置，从而最小化特定敏感型测量的误差。 使用PXI-2567继电器驱动器以及其他基于PXI的开关，您可以使用受支持的标准驱动程序API对整个开关系统进行编程，包括测试机架中的COTS开关和测试连接件中的定制继电器。

在测试连接件和测试机架中使用开关的优点

通过将COTS开关放置在测试支架中，并将继电器安装在测试连接件中，即可构建一个可轻松扩展的开关系统，并使关键或底层测量的误差最小。使用此架构，您可以将用于路由敏感信号的开关放置在测试连接件中，并将所有其余开关放置在测试机架中。 除了可扩展性，使用COTS开关还可以帮助您利用特定供应商的功能，例如NI PXI开关模块中的继电器计数跟踪和硬件触发功能。

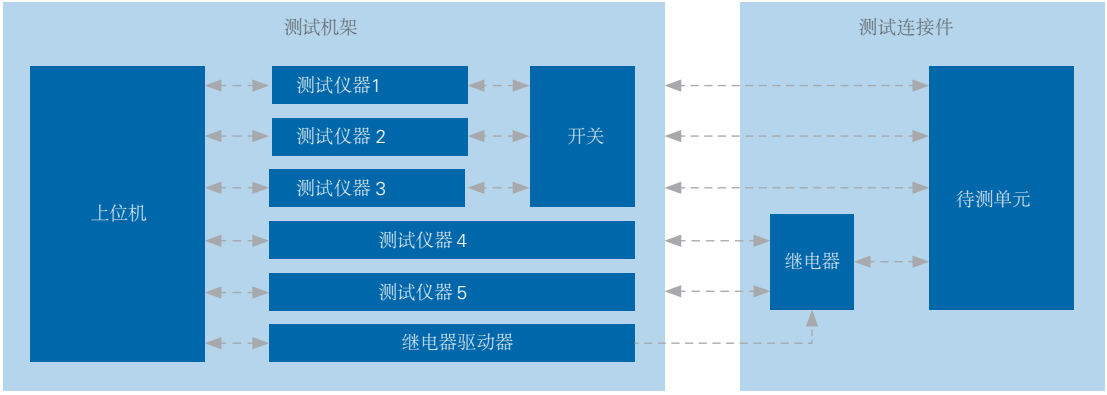


图 6. 在测试机架和测试连接件中采用开关的测试系统提供了更高灵活性，但需要额外的设计工作。

测试连接件和测试机架中采用开关的缺点
使用开关可能会减慢测试进程，因为这需要按顺序测量测试点，而不是并行测量。 将定制开关接入测试连接件也很耗时，并且需要较丰富的PCB设计知识，尤其是对于高压或高频信号。

如何为应用选择最佳开关

除了开关位置之外，您还需要比较各种开关拓扑和继电器类型，以确保开关子系统满足您的信号要求和测试目标。 对于自动测试应用，“开关”这个词通常用于描述使用继电器在多个组件和仪器之间开关信号的COTS设备。 开关以各种方式连接继电器，形成不同的开关拓扑，例如通用继电器、多路复用器和矩阵。 不同的继电器类型各有优缺点，包括尺寸、信号速率和预期寿命。 本节介绍了常见的开关拓扑、常用继电器类型、主要开关规格，以及在自动测试系统中使用开关的常用技巧和窍门。

常见开关拓扑

在确定您的应用适合采用开关架构之后，下一步是选择最佳的开关拓扑或排列继电器以构建更大型的开关网络。大多数开关供应商将开关分为三大类：通用继电器、多路复用器和矩阵。某些开关（例如PXIe-2524）具有多种拓扑结构，使您能够在软件中配置拓扑结构。 您可以选择五种不同的拓扑结构来满足不断变化的需求。 在考虑拓扑时，需要考虑所需的连接总数、同时连接的最大数量以及将来对测试系统进行更改的扩展需求。

通用继电器
通用开关由多个独立的继电器组成，彼此之间独立使用。如果只是想闭合/断开电路内的连接或两个可能的输入或输出之间的开关时，通用继电器是一个很好的选择。继电器按照其刀掷的数量进行分类。继电器的刀是各条路径的共同接线端。每个继电器的刀可以连接的位置就是继电器的掷。

单刀单掷(SPST)继电器类似于只有导通和关断状态的标准灯开关。 SPST继电器有两种形式：A型和B型。A型 SPST继电器处于常开状态，直至继电器被激活，继电器触点接通，形成完整的电路。 或者，B 型SPST处于常闭状态，直至继电器被激活，继电器触点断开其连接，从而断开电路。

图7. SPST继电器有两种类型：常开（A型）和常闭（B型）。

单刀双掷(SPDT)继电器具有单刀或称作公共连接端，可在常开触点和常闭触点之间切换。SPDT分为C型或D型继电器。C型SPDT激活时，先断开常闭信号路径，然后将继电器连接到常开触点。这一SPDT继电器操作称为“先开后合”或BBM。D型继电器激活时，先闭合常开信号路径，然后断开常闭信号路径。 此SPDT继电器操作称为“先合后开”或MBB。

图 8. SPDT继电器在两个掷/连接之间共享一个公共端（刀）。

任务	断开	操作过程	操作完成
C型	N.C. COM N.O.	N.C. COM N.O.	N.C. COM N.O.
D型	N.C. COM N.O.	N.C. COM N.O.	N.C. COM N.O.

图9. SPDT继电器也分为两类：C型和D型。

双刀单掷(DPST)继电器是两个A型 SPST继电器同时启动，通常使用相同的线圈并封装在一起。DPST适用于需要两个信号路径同时断开或闭合的情况。DPST可由两个独立控制的A型SPST继电器组成，但在两个继电器启动时可能会存在一些时间差。

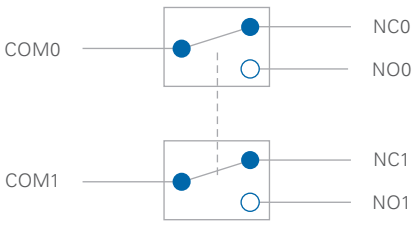


图10. DPST继电器可同步控制两个A型SPST继电器。

多路复用器

多路复用器是多个继电器的一种组合方式，可允许一个输入连接到多个输出，或一个输出连接到多个输入。 多路复用器提供了将多个DUT连接到单个仪器的有效方式。 然而，这种开关架构需要较多的前期知识来了解哪些DUT连接需要访问各种仪器。

多路复用器有时使用多个A型 SPST继电器并将继电器的末端连接在一起。 这种构建多路复用器的方法简单且有效，但是其缺点是未使用的信号路径可能会引起AC信号反射，从而降低开关的额定带宽。

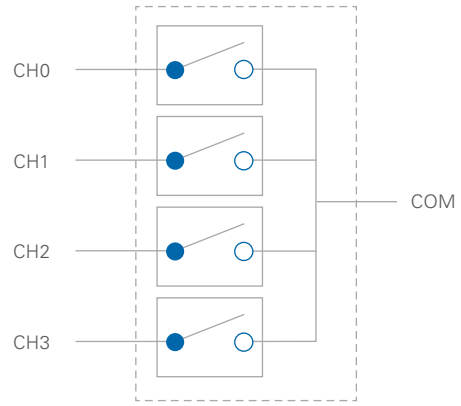


图11. 4 x 1多路复用器由多个A型SPST继电器并联组成

或者，多路复用器有时采用级联的C型SPDT继电器组合来确保AC信号的信号完整性。 这种类型的多路复用器通常需要更多的PCB空间，但它减少了可能降低开关带宽的任何桩线或额外的非端接信号路径。

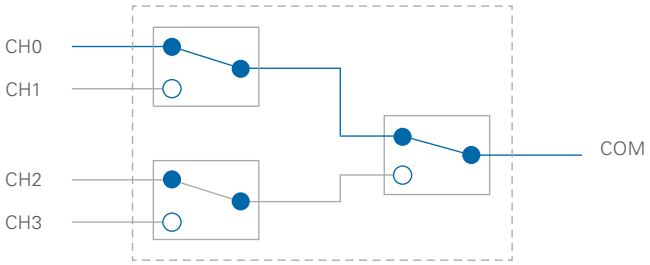


图2. 4 x 1多路复用器由多个C型SPDT继电器级联组成

矩阵

矩阵是最灵活的开关配置。 与多路复用器不同，矩阵可以同时连接多个信号路径。 矩阵在行和列的每个交叉点配有一个继电器，从而能够连接列-列、列-行和行-行信号路径。 借助矩阵的灵活性，您可以通过各种信号路径将所有开关通道彼此连接，而不需要预先定义。通常建议在硬件规划阶段规划开关路由，但是矩阵可以灵活地根据测试需求变化来更改开关路由。

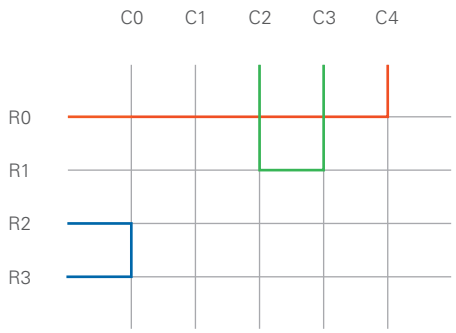


图13. 矩阵可允许最高程度的信号路由灵活性

矩阵大小通常描述为M行×N列（m×N）配置。 常见的配置包括4 x 64、8 x 32和16 x 16。但是，在大多数情况下，对行或列没有特别的要求。如果您更习惯用行而不是列来思考，例如64×4矩阵而不是4×64矩阵，则开关矩阵可以随时转置。

其他拓扑

通用继电器、矩阵和多路复用器开关构成绝大多数开关，但是也有其它专用开关拓扑存在，例如稀疏矩阵或故障插入单元（FIU）。

稀疏矩阵是混合组合，介于矩阵和复用器之间，通常用于RF应用。 通过连接两个复用器的公共端，您可以创建具有许多行和列的伪矩阵，但在任何给定时间只能连接一个可能的信号路径。 多路复用器通常比矩阵提供更多的信道密度，因为矩阵的每个行 - 列交叉需要至少一个继电器。 因此，稀疏矩阵通常可在给定空间中提供更多的通道密度，但却受到行列之间单个信号路径的限制。 稀疏矩阵也可用于AC应用，但是信号带宽可能会因为传统矩阵中未端接的行和列所形成的桩线而降低。

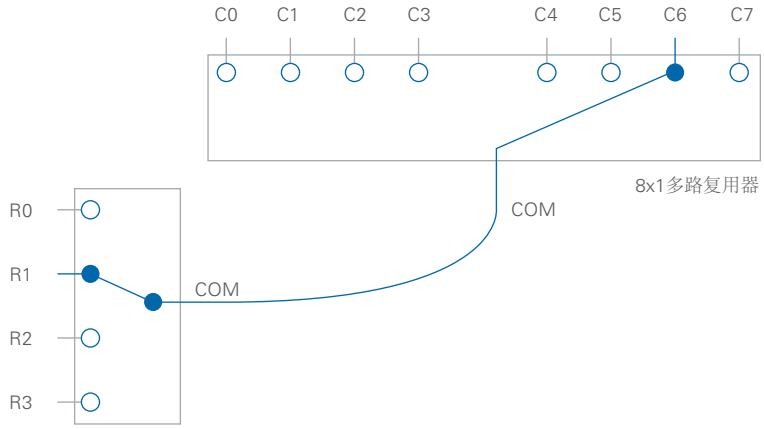


图14. 稀疏矩阵通过连接两个或更多个复用器的公共端来创建，并且通常用于切换RF信号。

另一种专用开关架构是FIU，通常用于硬件在环(HIL)测试系统中。 硬件故障插入，也称为故障植入，是测试系统的一个关键考虑因素，用于测试嵌入式控制单元的可靠性，需要同时植入已知响应和可接受响应的故障条件。 为了实现这一点，FIU插入在测试系统的I / O接口和ECU之间，使得测试系统可以在正常操作和故障条件（比如电源短路、对地短路、引脚到引脚短路或开路）之间切换。 有关FIU的更多信息，请阅读《**使用故障插入单元(FIU)进行电子测试**》白皮书。

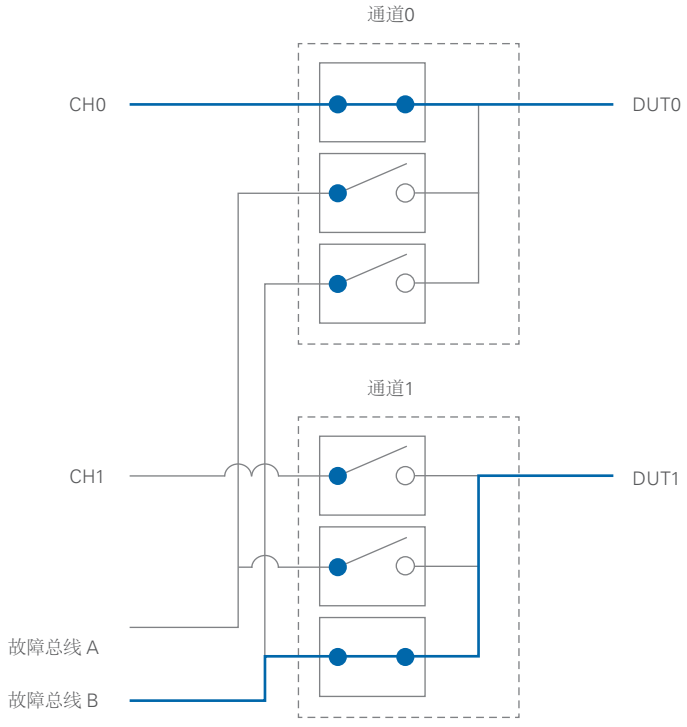


图15. FIU可允许自动化故障条件测试，通常用于测试嵌入式系统的可靠性，比如汽车ECU。

继电器类型

继电器是在电路中建立或切断连接的远程控制设备。市面上有很多类型的继电器，但常见的继电器类型有四种：机电继电器、干簧继电器、固态继电器和场效应晶体管(FET)继电器。 每种继电器都有可能影响开关系统的性能、成本、预期寿命和密度，这就是为什么选择最佳继电器类型来满足应用需求非常重要的原因。

请注意，单个继电器和成品开关产品的规格在大多数情况下是不同的。 继电器规格，例如带宽、额定功率和接触电阻，仅指单个继电器，不包括将继电器连接到开关拓扑结构的PCB路径或为开关拓扑结构提供用户接口的连接器。 例如，单个继电器可以是0.05Ω的接触电阻和300V的额定电压，但是成品开关产品可以具有更大的路径电阻（例如1Ω），包含多个继电器和PCB走线，并且单个继电器不需要具备300V下安全操作开关产品所需的PCB 漏电性能和间隙。

机电继电器

机电继电器（EMR）或电枢继电器通过流过电感线圈的电流来感应将电枢移动到断开或闭合位置的磁场，通过两个触点的接触来完成电路。EMR有不同的类型，例如锁存和非锁存，但其在操作上的差异很小。 非锁存EMR使用单个线圈，并在电流停止流动后返回到其默认位置。 而锁存EMR即使在电流停止流动时也保持在所处的位置。 一些锁存EMR使用一个线圈来使电流反向流动，以反转磁场的方向，从而将电枢推或拉到所需的位置。 其他锁存EMR使用电枢任一侧的线圈推动电枢断开或闭合。

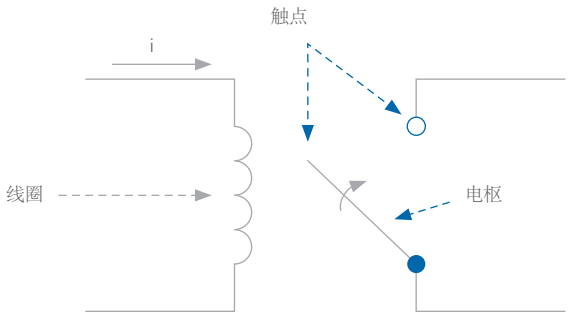


图16. 单线圈机电继电器使用磁场来断开和闭合机械开关。

EMR支持从低电压/ 电流到高电压/ 电流以及DC到GHz频率等各种信号特性。 此外，EMR具有低接触电阻，通常远小于1Ω，并且可以处理高达300 W的意外浪涌电流和高功率。因此通常可以找到符合测试系统所需信号特性的EMR 。 但是，EMR会占用非常多的PCB空间，与其他选项相比速度较慢（150次关断/秒），并且由于部件的移动性而具有较短的生命周期（最多10的六次方次关断）。

基于这些优缺点，如果需要高功率、高电流或高带宽耐用型继电器且不太关心继电器速度，而且愿意经常更换继电器（因为继电器的性能会随时间的推移而降低），EMR是不错的选择。

干簧继电器

干簧继电器通过流过电感器的电流来产生用于连接物理触点的磁场。 然而，干簧继电器具有比EMR小得多且更轻的触点。 干簧继电器的线圈缠绕在两个重叠的铁磁性叶片（称为簧片）上，簧片密封在填充有惰性气体的玻璃或陶瓷胶囊内。 当线圈通电时，两个簧片被拉在一起，触点相接触，在继电器内形成信号路径。 当电流停止流过线圈时，簧片的弹簧将触点分离。

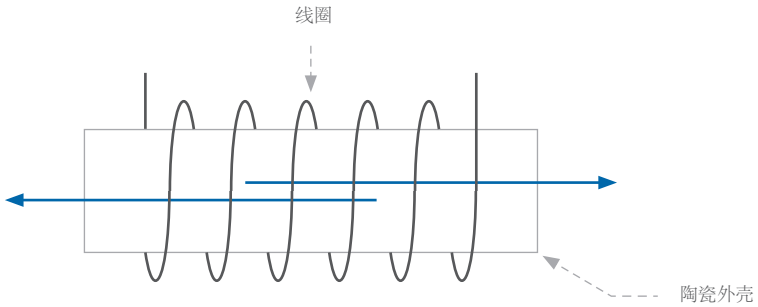


图17. 电流停止流过线圈时，簧片的弹簧力会使触点分开。

由于干簧继电器的体积更小，因而可以在更小的空间内安装更多继电器，并且它们的开关速度可高于EMR，最高可达2000次关断/秒。此外，其有限的移动机械部件和隔离的环境提供了更长的机械寿命，高达10的九次方次关断。

然而，由于干簧继电器的接触尺寸较小，干簧继电器无法处理高功率信号，并且更容易受到自热或电弧的损坏，导致簧片出现小面积烧坏。当熔化部分固化时两个簧片仍然连接，那么触点就会焊接在一起。在这种情况下，如果弹簧力足以拉开两个簧片，则继电器保持关闭，或者断开其中一个簧片。为了防止损坏，需要监测可能由于热切换容性负载而引起的大浪涌电流信号，并使用内联保护电阻来降低电流尖峰的电平和持续时间。有关保护干簧继电器的更多信息，请阅读《干簧继电器保护》白皮书。干簧继电器的小尺寸和高速度使其成为许多应用的理想选择。干簧继电器更常见于矩阵和多路复用器模块而不是通用开关模块。一款典型的产品是 PXI-2530B，它是一个COTS开关，可以通过切换各种前端接线端子来配置为13个独特的矩阵或多路复用器拓扑。

固态继电器

固态继电器(SSR)属于电子继电器，其组件包括用于响应输入的传感器、将电力开关到负载电路的固态电子开关装置以及用于在没有机械部件的情况下激活控制信号的耦合机构。 它们通常使用具有LED的光敏金属氧化物半导体场效应晶体管（MOSFET）器件来进行驱动。

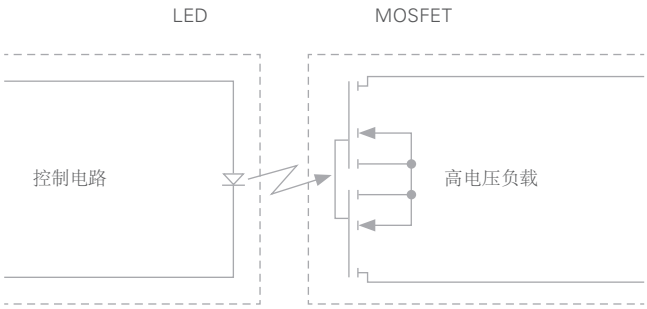


图18. SSR使用具有LED的感光MOSFET来驱动设备。

SSR的速度略快于EMR，高达300次关断/秒，因为它们的开关时间取决于为LED开启和关闭所需的时间。由于没有机械部件，SSR不易受到可能损坏继电器的物理振动的影响，因而具有无限长的机械寿命。

然而，SSR也有其缺点。 首先，它们不如EMR那么稳定，并且如果信号电平超出其额定值，则容易损坏。 第二，SSR较为昂贵，并且产生的热量比其他类型的继电器高。最后，由于SSR是通过晶体管而不是物理金属连接进行连接，因而路径电阻可从小于1Ω到100Ω甚至更大。最新的SSR对路径电阻进行了优化，以减小其影响。

当使用中小信号电平，且需要一个可以持续多个周期的继电器时，具有无限机械寿命的SSR就是一个很好的选择。COTS SSR开关的一个示例就是PXI-2533，它是一个4×64矩阵，具有55W额定开关功率，并且提供无限的机械寿命。

FET继电器

与SSR类似，FET继电器不是机械装置，而是使用晶体管来路由信号。与SSR不同的是，控制电路直接驱动晶体管的栅极而不是驱动LED。

直接驱动晶体管栅极可实现比上面所提到的任何其它类型的继电器更快的开关速度，高达60,000次关断/秒。此外，无机械部件使得FET继电器的体积比机电或干簧继电器小得多，并且不易受到冲击和振动问题的影响，这使得FET继电器具有无限的操作寿命。然而，FET继电器具有比任何其他继电器类型高得多的路径电阻，通常在8Ω至15Ω范围内，并且缺少物理隔离，因此仅适用于低电平信号。

FET继电器是低电平信号和需要快速继电器操作或无限机械寿命的应用的理想选择。COTS FET开关的一个示例是PXI-2535，它是一个4 x 136矩阵，可在少于16µs内执行继电器操作。

低○ 平均◒ 高●

功能	电枢	簧片	FET	SSR
高功率	●	◒	○	◒
高速	○	◒	●	◒
小尺寸封装	◒	●	●	●
低路径电阻	●	◒	○	◒
低电压偏置	◒	○	◒	●
更长的生命周期	○	◒	●	●

图2. 不同继电器选项比较

开关扩展

如果自行创建开关拓扑，则可以创建一个满足应用精确尺寸需求的矩阵或多路复用器。 但是，许多人选择使用COTS开关来减少开发工作，并且大多数COTS开关具有固定尺寸。 因此，知道如何组合多个矩阵或多路复用器以创建更大的矩阵或多路复用器是非常重要的。

多路复用器扩展

扩展多路复用器通道数的最简单方法是直接将多个多路复用器的公共端连接在一起。 这种方法存在输入通道同时短路并可能损坏硬件的风险。 因此，在任何给定时间需要确保只有一个通道连接到公共端。 某些开关软件，如Switch Executive，可让用户设置软件排除条件，防止在任何给定时间多个输入路径连接到公共端。 这种方法的另一个缺点是未使用和未端接的路由会导致桩线，从而增加电容并降低高频性能。

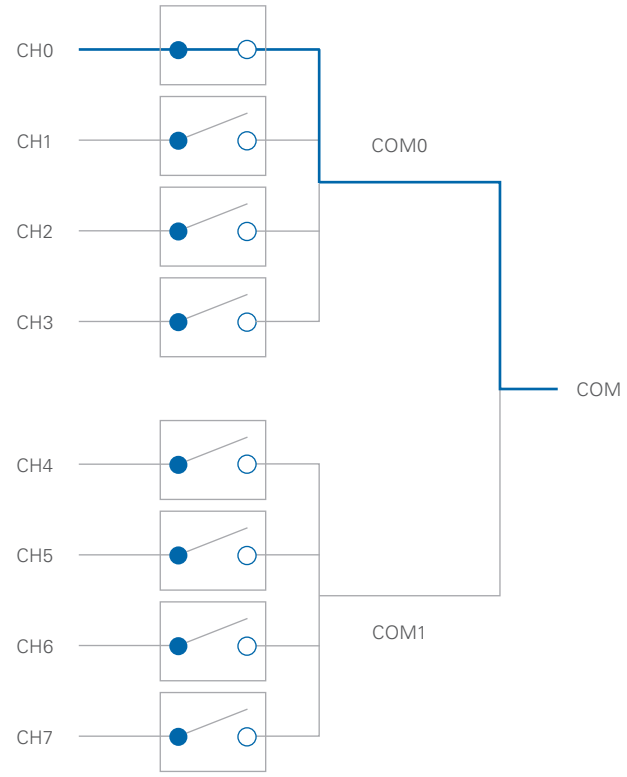


图19. 8 x 1多路复用器将两个4 x 1多路复用器的公共端连接在一起。

另一种方法是增加一个多路复用器来连接多个多路复用器的公共端，多路复用器的内部结构仅允许一个通道路径到公共端，因而需要更多的多路复用器。但是，这种方法仍然会存在PCB桩线，从而降低带宽性能。

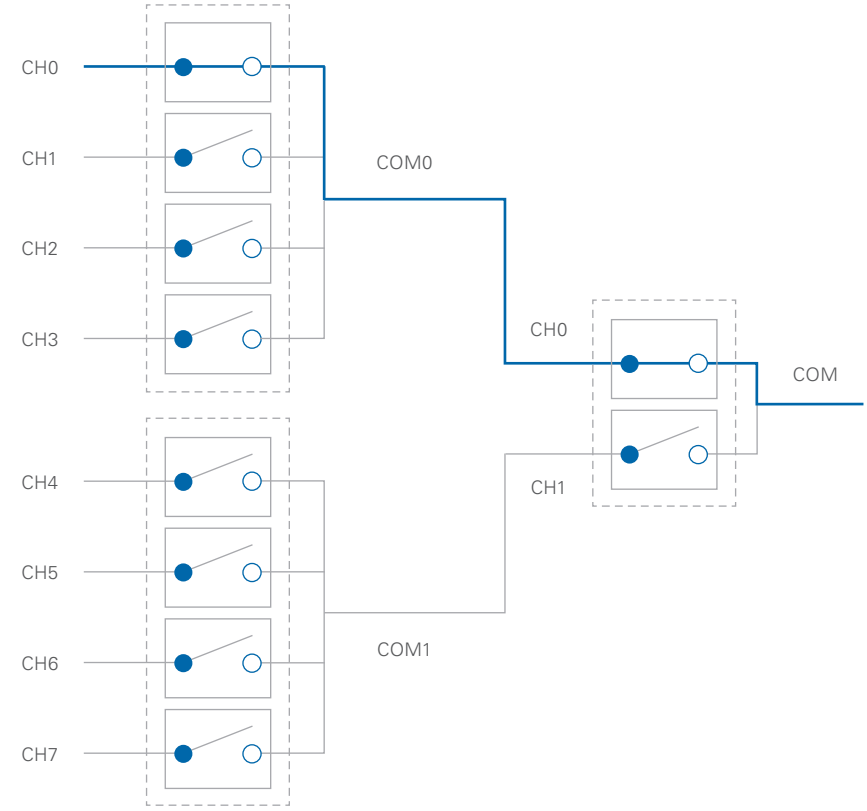


图20. 8 x 1多路复用器由两个4 x 1多路复用器通过一个额外的多路复用器来切换公共端组成。

对于高频应用，应使用C型SPDT继电器来创建大型多路复用器。此选项可确保有效信号路径上不存在任何桩线，从而有助于提高开关的带宽。

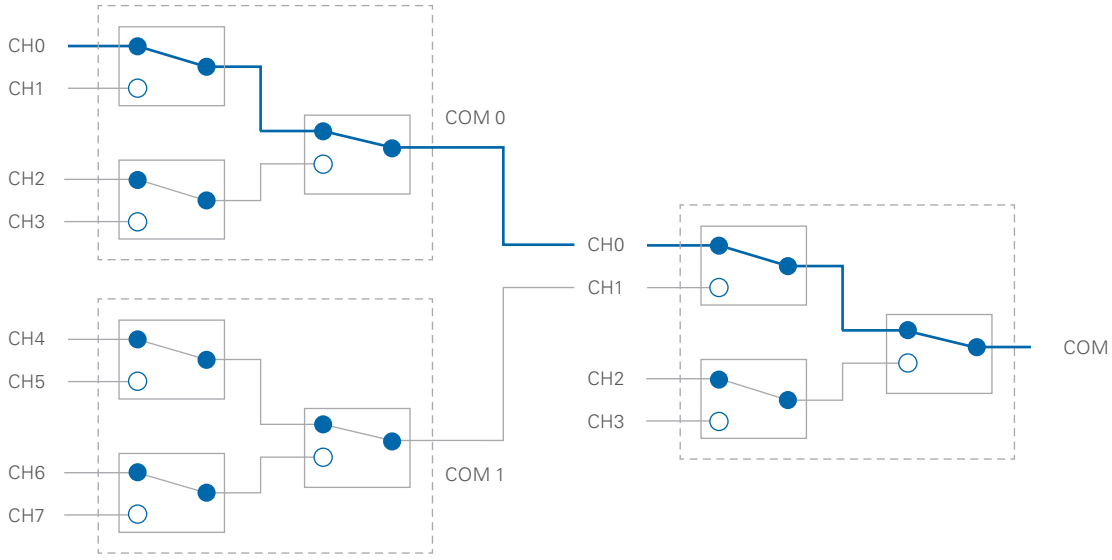


图21. 8 x 1多路复用器由三个采用C型SPDT继电器的4 x 1多路复用器级联而成。

矩阵扩展

开关矩阵还可以用于创建远大于单个COTS矩阵开关尺寸的大型矩阵。扩展矩阵的方法有两种。列扩展是指连接两个或多个矩阵模块之间的每一行，使扩展矩阵的列数加倍。行扩展是指连接两个或更多个矩阵模块的每一列，使扩展矩阵的行数加倍。

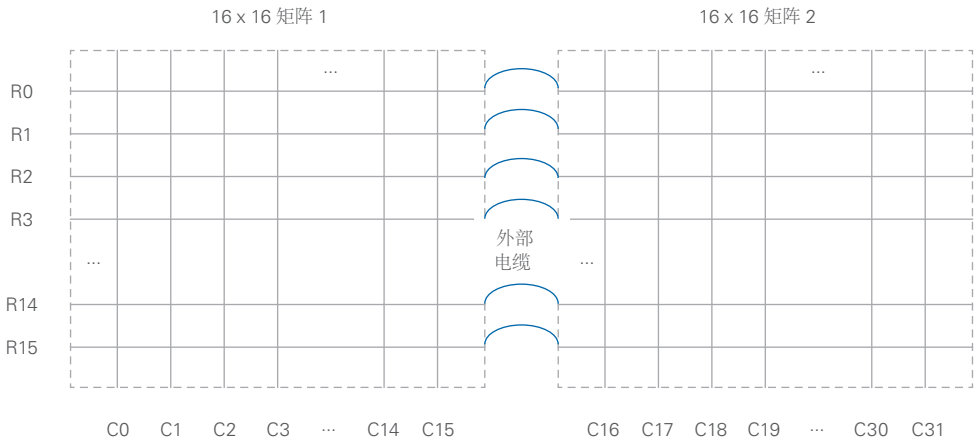


图22. 通过列扩展创建的16 x 32矩阵

为了简化矩阵扩展，一些COTS矩阵开关（如PXIe-2532B）提供专用电缆，可轻松连接多个开关模块的行来组合矩阵。不过所有矩阵都是可扩展的，即使没有预先提供附件也是如此。如果要手动扩展矩阵，可使用外部导线连接单个矩阵的行或列。有关矩阵扩展的更多信息，包括示例和常见问题，请阅读《PXI开关模块的矩阵扩展指南》。

主要开关规格

除了继电器类型和开关拓扑，还需要确保开关子系统所连接的信号的完整性。 大多数开关根据信号类型分为两类：低频/DC和RF。

低频/直流开关规格

开关通常会标出额定电压和电流，但还应注意最大开关功率规格，也就是触点可以切换的功率上限。 例如，150 V 2 A开关的最大开关功率为60 W，而且不应在2 A (300 W)时使用150 V电源。 因此，除了最大电压和电流电平之外，考虑信号的最大功率也很重要。

在处理开关时，信号频率也是一个棘手的话题。 信号经常通过其基频进行描述，这适用于简单的正弦波。 但对于方波或具有尖锐边缘的信号，则需要记住，方波具有远高于基频的谐波频率，可能会形成尖锐边缘。 如果打算切换一个方波，请选择一个额定切换频率为信号基波频率7到10倍的开关。 例如，如果使用10 MHz的开关路由10 MHz方波，输出将看起来更接近正弦波而不是方波。

有关开关带宽的更多信息，请阅读《开关带宽选择》白皮书。

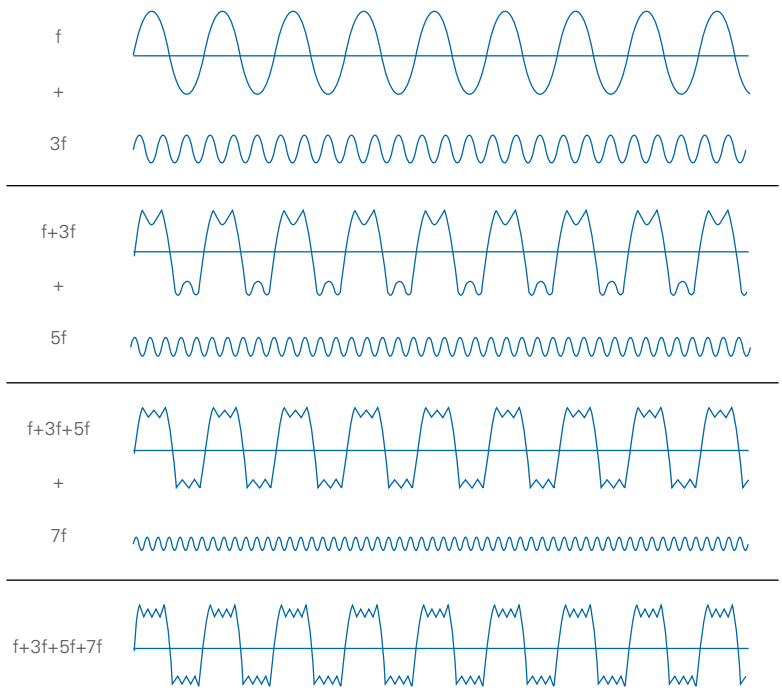


图23. 方波信号具有远高于基频的谐波频率。

开关路径电阻、热EMF和偏置电压会影响低电平信号测量，例如DMM电阻测量。 因此，应该选择一个对测量影响最小的开关，并设计一种测量技术来补偿这些误差源。 有关在切换低电平信号时如何减少误差的详细信息，请参阅以下白皮书之一：

第一部分：切换低电压信号时如何减少误差

第二部分：切换低电流信号时如何减少误差

第三部分：切换低电阻信号时如何减少误差

射频开关规格

额定频率大于10 MHz或20 MHz的开关通常称为RF开关。 RF开关通常具有较低的信道密度，以保持信号完整性，因此RF开关适用于需要较高带宽的信号路径。 但是，拓扑结构和带宽限制不足以提供足够的信息来选择一个RF开关。

所有RF开关都具有额定特性阻抗，这是用于确定传播信号在信号路径中以何种程度传输或反射的传输线参数。 组件制造商在设计设备时专门将特性阻抗设计为50 Ω或75Ω，因为RF系统中的所有组件必须阻抗匹配才能最小化信号损耗和反射。 RF市场大部分是50Ω RF系统，应用于大多数通信系统。 75Ω RF系统较为少见，主要用在视频RF系统中。 务必要确保电缆和连接器等组件以及可能安装在测试系统中的其他仪器都是阻抗匹配的。

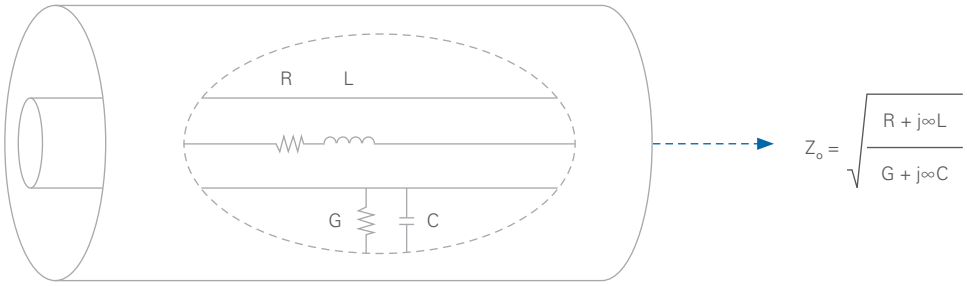


图24. 传输线的特性阻抗

除带宽和特性阻抗外，还有其他RF开关规格会直接影响信号完整性，如插入损耗、电压驻波比（VSWR）、隔离、串扰和RF功率。 插入损耗衡量的是信号通过开关后的功率损耗和信号衰减。 VSWR是反射 - 透射波之比，具体来说是“驻波”模式中最大值（当反射波同相时）与最小值（当反射波不同相时）的比率。 隔离是耦合在开路电路上的信号幅度，串扰是耦合在电路（比如独立的多路复用器组）之间的信号幅度。

RF开关一个有趣的现象是所有这些规格会根据信号频率而变化。 因此，在选择RF继电器或开关时，应在特定信号频率下比较这些规格。 否则，很容易误解RF开关的性能。

如需了解更多的关于RF开关选型，请阅读《理解主要RF开关规格》白皮书。

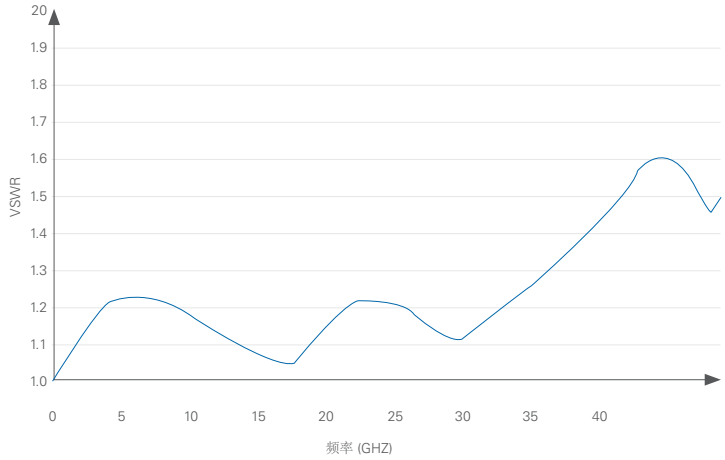


图25. 许多RF开关规格会随信号频率而变化。

开关使用技巧和窍门

在规划自动化测试系统的开关部分时，一些常规技巧可以帮助您构建一个保持信号完整性的高效开关系统。

总测试点与同时连接

使用矩阵时，请考虑可能的最大连接数量和同时连接的最大数量。 如果您仅关注可能连接的总数，则通常会导致所有行专用于每个仪器的每个I / O引脚。 但是，这种方法可能导致不必要的大型矩阵。 例如，如果您有22个仪器引脚和106个DUT测试点，那么您可能会创建一个22 x 106矩阵（2,332个继电器），其中22个I / O引脚连接到行，106个DUT测试点连接到列。

但是，如果在同一时间仅需要连接最多四个仪器引脚，则22 x 106矩阵就会过大且造成不必要的浪费。 相反，我们可以考虑将仪器放置在22个额外的列上，并使用行在列之间进行路由。 这样就可以将矩阵尺寸减少到4 x 128（512个继电器），大约为原尺寸的20%。 这不仅可以节省空间和金钱，而不会影响测试时间或质量。

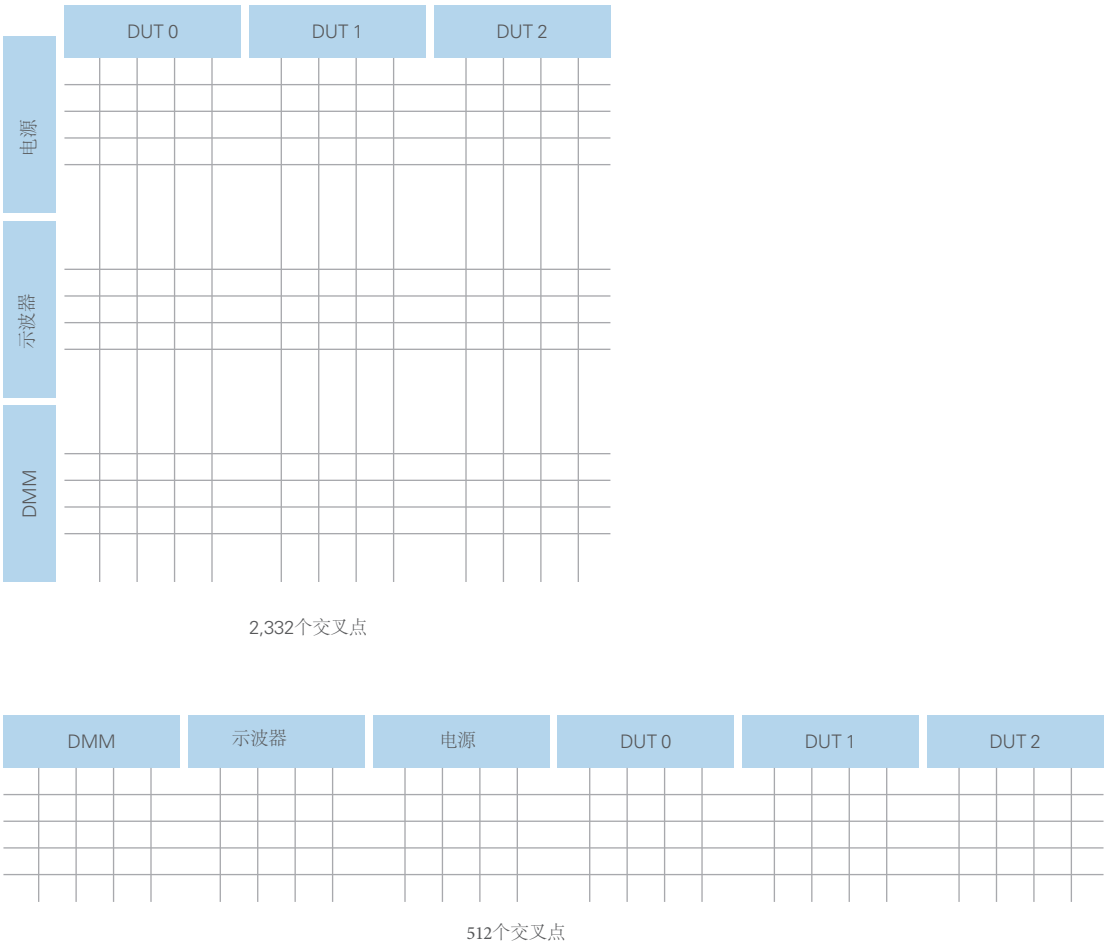


图26. 将仪器放在列上，然后使用行来路由信号，可在顺序测试执行过程中反转矩阵空间，但是将仪器放在行上可实现更快速的并行测试需求。

N线开关

许多矩阵或多路复用器开关模块可以在给定拓扑内而不是标准的1线开关模式中切换两个或四个信号路径。 在执行测量时，可以使用1线开关将各种信号路由到可能以单个信号或地为参考的仪器。

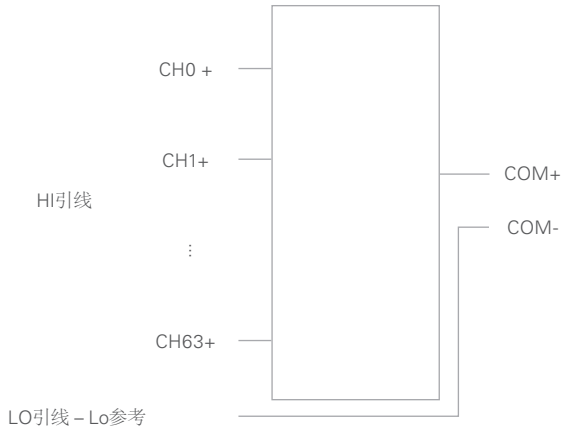


图27. 单端多路复用器非常适用于以共享信号或地为参考的测量。

有时多个信号需要同时切换。2线或差分开关提供两个信号路径，可使用一个命令进行控制。这提供了一种简单的方法来切换差分信号，同时具有出色的共模噪声抑制性能。4线开关通常用于4线电阻测量，其中两条引线用于激励，另外两条引线用于测量DUT上的电压降。

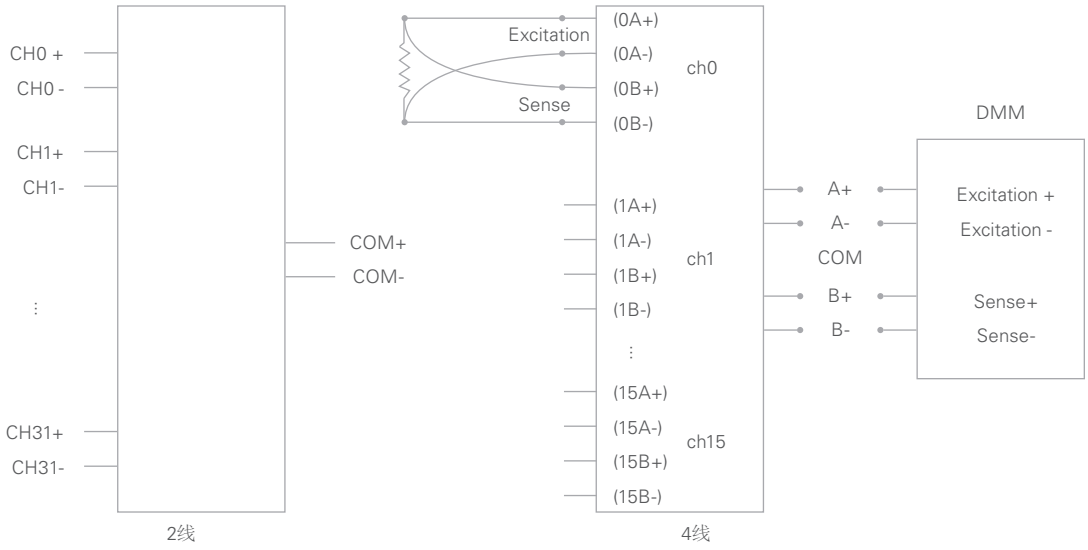


图28. 使用2线或4线开关同时切换多条信号路径。

开关功率

很多时候，测试需求计划包括最大电压和电流电平，但瞬时功率通常被忽略。 开关或继电器的额定电压和电路可能为100 V和2 A，但这并不一定意味着它可以处理200 W的信号。许多开关具有完全独立于其电压和电流额定值的最大额定功率。 例如，常见的干簧继电器的额定电压和电流可能为100 V和500 mA，但它的最大额定功率是10 W。因此，在选择开关时，应考虑最大瞬时功率。

将高电平信号从一般或低电平信号中分离

高功率或高频信号的开关通常比通用信号的开关具有更低的密度。因此，应将大功率或高频信号从开关系统中隔离出来，以维持主开关系统的通道密度。如果要创建一个适合所有信号的开关来处理高电平或高频率信号，则该开关可能非常大型且昂贵。

基于信号频率比较RF规格

比较RF开关时，应根据信号频率评估规格。许多RF规格，例如隔离、VSWR、插入损耗和RF载波功率会根据信号频率而变化。为了进行准确的比较，需要查看详细的开关规格，找到所需频率的规格。 此外，一些开关供应商针对每个类别的开关分别发布了保证和典型的规格，而其他开关供应商仅发布典型的规格，这些规格看起来比保证规格要好得多。

考虑最大开关速度的硬件触发开关

在许多自动测试场景中，时间就是金钱。 许多开关使用软件命令进行单独控制，总线延迟和软件开销计算到每个开关操作上。 某些开关提供硬件定时和触发，这使您能够将开关连接列表加载到开关内存上，并使用硬件触发器依次触发列表的连接。 每次开关操作完成后，开关会向仪器发送触发，启动下一次测量。

这种操作称为开关握手，可以消除与传统软件触发开关相关的软件开销和总线延迟。 开关握手对于高速继电器类型（例如FET或SSR）尤其重要，因为软件开销和总线延迟构成了大部分的开关操作。开关握手与簧片继电器相结合可能会使总开关时间减少10倍，而FET开关则可能缩短100倍甚至更多。 继电器速度越快，开关握手可以提高的吞吐量就越多。

下一步

NI开关产品

无论是在十几个测试点上执行高精度低速测量，还是对集成电路进行高通道高频特性分析，NI都提供基于PXI的灵活模块化开关解决方案，帮助您最大限度地重复利用设备、提高测试吞吐量和系统可扩展性。

了解更多关于NI PXI开关产品

Switch Executive

Switch Executive是一款智能开关管理与路由应用程序，能够加速开发并且简化对复杂开关系统的维护。只需轻点鼠标的图形化配置、自动路由功能以及直观的通道别名，即可帮助轻松设计和描述开关系统。

了解更多关于Switch Executive

NI Switch Health Center

为了简化高通道数系统的继电器维护和提高其可靠性，NI Switch Health Center会将信号沿开关的每个路由发送出去，以验证每个继电器的状态。它会提醒用户继电器是否发生故障、卡住断开还是卡住闭合，并报告电阻的变化来确定继电器是否即将“寿终正寝”。

了解更多关于NI Switch Health Center

测试系统构建基础知识

测试执行软件

目录

引言

大多数测试系统基本上围绕两个概念进行设计：效率和成本。无论是在消费电子行业还是在半导体生产领域，测试工程师都关心测试系统的独立测试时间和总吞吐量，以及这些参数如何影响资源。当应用不断扩展到包含多个测试、多种仪器和多个待测设备(UUT)时，便不可避免地需要监测测试执行软件，以解决成本和效率问题。

测试执行软件通常作为定制化的解决方案进行部署，或作为商用现成(COTS)产品直接购买。在典型的构建V.S.购买论证中，测试架构师必须确定自行编写测试执行程序更合理，还是投资和整合现有解决方案更具成本效益。在决定定制还是购买测试执行软件之前，有必要了解此类软件的目的和核心功能。本指南总结了测试执行软件的主要功能，并探讨其实际应用场景。

背景

测试执行软件可以自动化和简化大型测试系统。 测试执行软件处于软件堆栈的最上层，整合了测试的基本功能，如测试执行、结果采集和报告生成。 该解决方案的特点并不是针对特定的UUT，因此各种应用可以使用该测试执行软件作为框架。 这意味着使用LabVIEW图形化语言、C、.NET或其他语言编写测试代码的开发人员可以专注于测试特定设备，而所有UUT的常用功能都由最上层测试执行软件维护。 总而言之，从开发、成本和维护的角度来看，测试执行软件以有效的方式定义了此类常用功能。

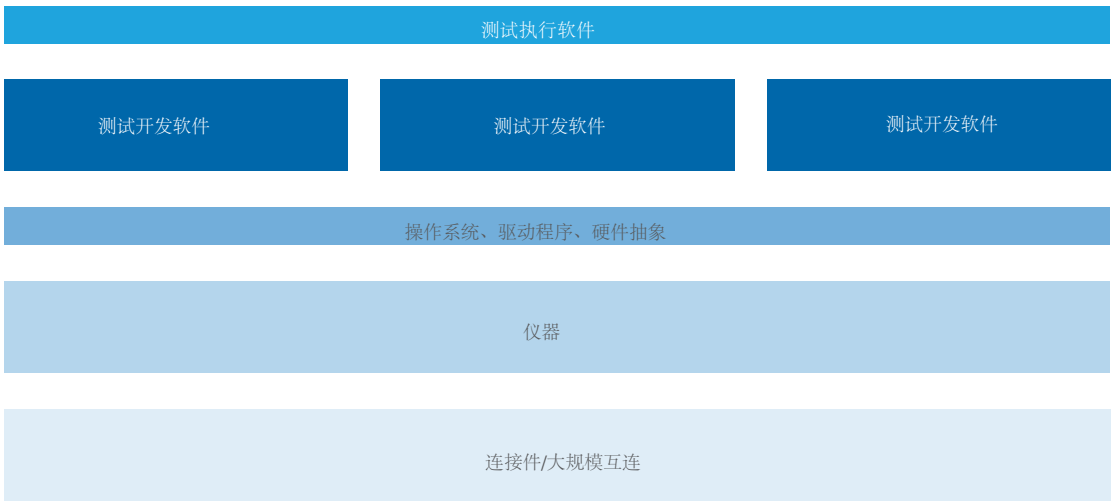


图1.测试管理人员通过完成在更高层次抽象的所有测试中通用的任务，允许单个测试开发与整个测试系统的架构需求分离。

测试执行软件的特性

取决于公司的规模、特定测试仪的规模以及待测设备的种类，测试执行软件可以从简单到复杂，不一而足。本指南概述测试执行软件可能包含的常见特性。一些特性对于测试执行软件的所有实现至关重要，而其他特性则属于附加功能，并不一定必要。每个特性列出了完成开发的估算时间。这些估算是基于数百个自动化测试客户的经验，详情请参与《测试执行软件—构建还是购买？ 基于NI TestStand的财务分析比较》。

测试序列开发环境

测试执行软件提供了构建测试序列的开发环境。这是最基本的特性，也是一个复杂的特性，为整个执行了提供开发接口。序列架构包括实现分支或循环逻辑的能力、导入测试限制值的方法以及独立测试代码的规范和整理。与测试代码的交互需要具有使用各种编译格式的灵活性，例如DLL、VI和脚本，以及与不同开发环境集成的能力。测试管理软件还可以使用源自源代码控制提供程序的测试代码。

在定制的测试执行软件中实现测试序列开发环境可能需要大约100人天才能完成，而商业解决方案提供了现成的环境。由于开发环境提供的功能范围，此特性如果作为内部解决方案，需要最长的开发时间。而且，该特性是序列架构体验的基础，不能被忽略。

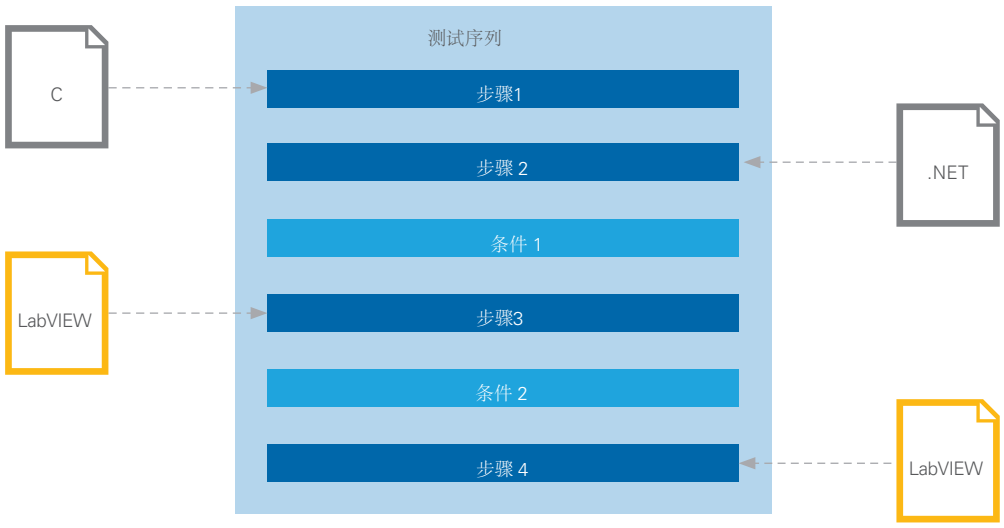


图2. 高效的序列开发环境可帮助测试工程师开发和调试复杂序列，并将序列调用道现有测试代码中。

自定义操作界面

操作员界面是操作员通过其与测试系统交互的显示器。它通常允许选择关键输入参数，例如UUT标识号、要执行的测试序列或报告路径。它还包含一个“运行”或“开始”按钮来控制执行。许多大型测试系统如今需要因应用或公司而异的专业化图形化用户界面，并且可让开发者灵活选择编程语言。除了定制之外，这种高度功能性接口包括加载、显示和运行测试序列的能力，以及交互式用户提示、执行进度指示器、测试数据的可视化和本地化。

设计自定义操作界面需要花费8到32个人日的开发时间。COTS解决方案提供了现成的库和UI控件，可以减少该时间。开发自定义操作界面可能是一个重要的时间投资，且不论测试执行解决方案是自行开发还是购买现成的。认为操作界面对其系统不那么重要的测试工程师会指示操作员在开发环境中工作。

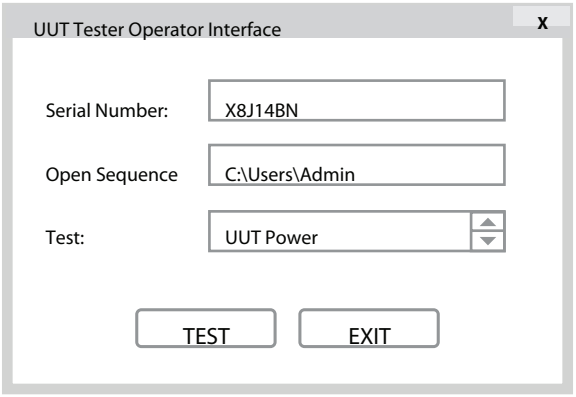


图3.用户操作员界面能够识别给定测试序列的的唯一UUT、公司、应用程序、测试和操作员角色。

序列执行引擎

测试执行软件的一个核心组成是序列生成引擎。序列执行引擎负责评估UUT所需的所有操作。这包括调用独立的测试代码、创建测试之间执行的流程，以及在测试之间管理数据。序列生成引擎是负责执行给定测试序列的程序，无论是在开发环境中、通过自定义操作界面，还是在部署的测试程序上。

内部实现序列执行引擎需要至少15个人天。但是，它是所有测试执行软件的必备功能。

结果报告

鉴于测试执行软件的抽象角色，这部分软件功能负责整合独立的测试数据、临时存储到内存中以及发布综合测试结果。报告可以有各种格式，包括XML、text、HTML和ATML。数据也可以在执行之后存储到数据库。测试执行软件通过可扩展的报告选项实现多样化的报告格式。结果报告是许多测试系统的必要组成部分。

从零开始开发结果采集和报告生成器可能需要大约15人天，取决于所需的具体报告。由于COTS解决方案中内置了报告生成器，因此可以定制结果报告，只需一人天甚至更少的时间就可以提供应用所需的报告。

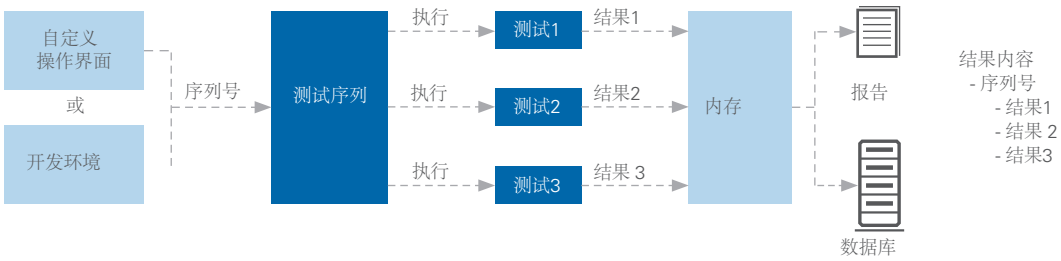


图4. 测试执行软件在测试系统的一部分作用是整合执行过程中生成的结果，并生成报告发送给数据库。

用户管理

根据使用人的角色和级别将测试执行级别区分开来是有必要的，用户管理工具可有效地划分总测试架构师、编写和调试测试代码的每个测试开发人员以及运行测试的操作员或生产经理之间的责任。 赋予特定用户的功能甚至会有密码保护，以防止误用测试序列。

在定制测试执行软件中实现用户管理系统需要大约5个人日的开发时间。 虽然对于使用测试执行软件不是必需的，但是用户管理工具不需要大量的人力投入即可实现，而且可以简化测试执行软件职责的执行。

权限	架构师	开发人员	操作人员
编辑	√	—	—
保存	√	—	—
部署	√	√	—
循环	√	√	—
运行	√	√	√
退出	√	√	√

用户	级别I
Mark	操作员
Larry	操作员
Julie	开发人员
Scott	开发人员
Lauren	架构师

表1. 类似于Windows文件权限，用户管理器可将测试执行软件的角色和责任区分开来。

并行测试功能

并行测试涉及同时测试多个待测件，并能够维持正确的代码模块性能、结果采集和UUT跟踪。并行测试方法的范围从流水线执行（按测试顺序执行，但测试执行软件可以同时测试多个socket）一直涵盖到动态优化、批处理或其他复杂的执行操作。

对于测试执行开发人员来说，实施并行测试通常是非常耗时，从头开始开发需要花费100人日。尽管并行测试可能需要大量时间来开发，但是对于大型测试系统，通过扩展执行来降低对吞吐量的要求通常至关重要。许多公司在首次部署测试执行软件时并没有考虑并行测试，后来才意识到他们最终需要这个功能，不过已经为时已晚。

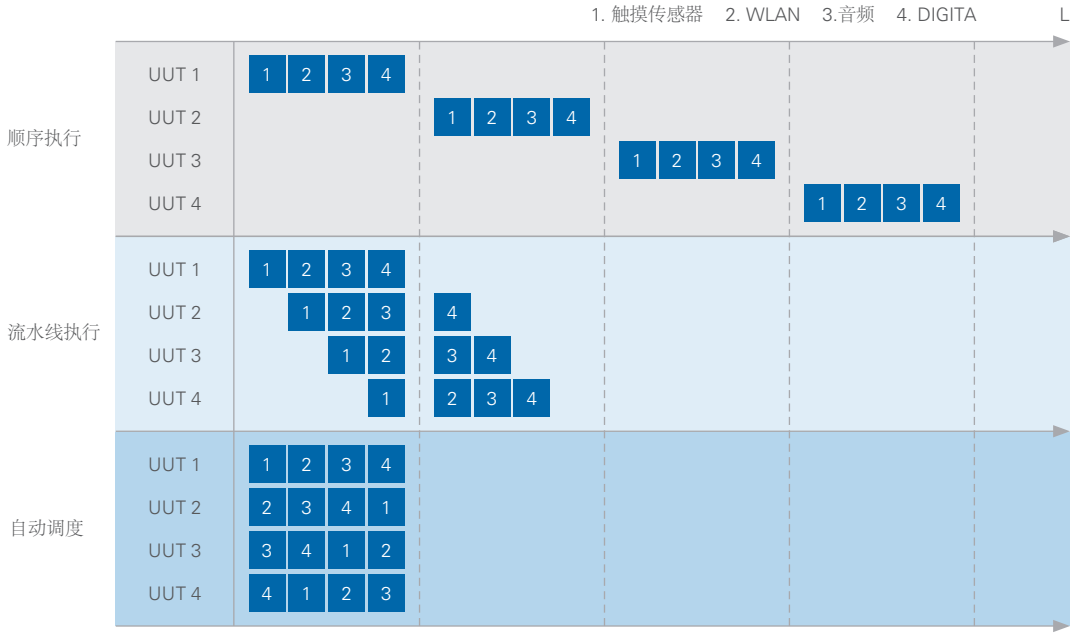


图5. 并行测试功能可显著增加系统吞吐量，而且无需重新构建测试执行软件的架构。

单元/设备跟踪和序列号扫描

同时测试多个UUT时，可能需要对每个被测设备进行唯一的标识和跟踪。 该信息可以与单元或批量级的特定分析的测试结果存储在一起，或者在出现问题时精确定位错误的来源。 设备跟踪可以由操作员通过键盘手动输入，也可以在读取条形码之后由全自动扫描仪加载UUT信息。

从头开始开发此类功能可能需要五个人日，如果采用COTS解决方案，大约需要一个人日进行自定义。 并不一定每个测试系统都需要UUT跟踪。但该功能适用于需要高容量、高吞吐量测试的应用，例如半导体或消费电子行业。

测试部署工具

大多数大型测试系统不是孤立构建的；通常是多个测试站点或整个生产车间的整体解决方案。测试执行软件提供了一种机制或实用程序来将整个软件堆栈打包到一个内置可发布单元中，从而在系统部署中发挥关键作用。测试系统可以以各种方式分布 - 架构师可能希望部署测试系统的映像或包含所有必需的依赖性和运行时间的完全功能的安装程序。有关此主题的更多信息，请参阅《构建测试系统基础知识》系列之《软件部署》白皮书。

部署是一项非常重要的任务，从头开始可能需要一个开发团队花费20人日的时间来完成。借助商用测试执行软件的现成部署实用程序，可能仍然需要三个人日的时间来成功部署。考虑到该特性对于多个测试站点的适用性，通常需要具有测试执行解决方案。

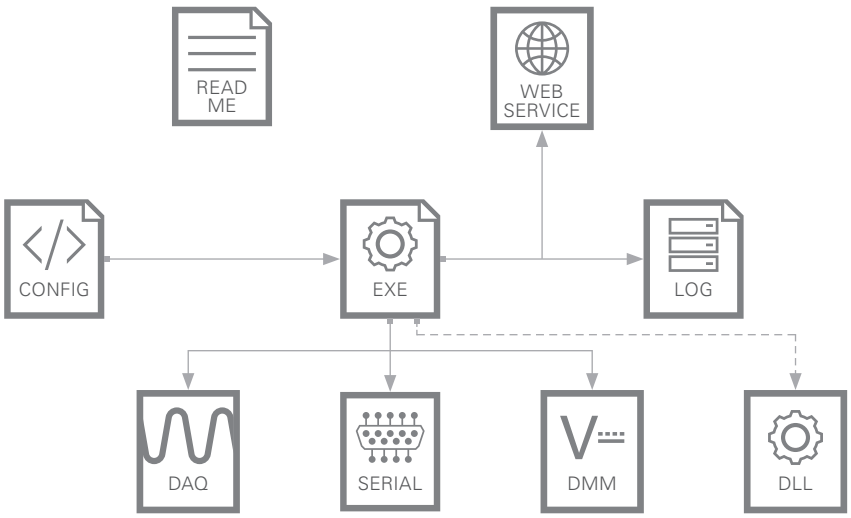


图6. 部署涉及使用部署工具或编译服务器将测试系统的所有必要组件封装起来，然后发布到所需的测试站。

维护

与大型测试系统中的任何其他组件一样，测试执行软件必须得到适当的维护，以确保其性能能够经受时间的考验。 这包括扩展以纳入新测试、升级以维持软件或操作系统升级的兼容性，并修复检测到的任何错误。 维护测试执行解决方案甚至延伸到文档领域。 文档是操作人员、开发人员和架构师在使用测试执行软件时所依赖的重要资源。

虽然很难预测特定测试人员的需求，但是每年开发自定义测试执行程序时需要在初期将15%的时间用于维护。 总开发时间包括生成足够文件所需的时间（估计20天）。测试执行软件支持的时间间隔可能会有所变化，这会大大改变成本的估算值。但是不建议在实施任何测试执行解决方案时减少维护工作。

实际场景 1

Jonathan是一家小公司设计实验室的测试工程师，该公司致力于提供低成本的消费电子产品。每个设备通过一个远程控制器进行控制。 Jonathan负责编写测试代码，用于在设备进入生产之前验证远程控制器和原型之间的发射器 - 接收器通信。 随着公司生产规模不断扩大，Jonathan可用于每台设备执行必要测试的时间减少了。 他需要自动执行现有的验证代码，以便可以将更多的时间花在为新设备编写代码上。 因此，他决定使用测试执行软件来执行测试代码序列。

下表列出了Jonathan对测试执行软件的要求。

功能	实现
测试序列开发环境	Jonathan需要一个开发环境来构建序列。他编写的测试代码已经相当模块化，所以他只需要在这个环境中调用和循环测试代码。
自定义操作界面	Jonathan想要能够以最小的交互执行一组测试代码。 他希望仅指定一些相关参数来识别设备、需要的测试和报告路径。 但是，由于他是终端操作者，因此将接口与开发环境分离并不重要。
序列执行引擎	这是Jonathan测试执行解决方案的核心需求。 每个测试由几个独立的LabVIEW VI组成，这些VI必须按顺序执行。
结果报告	现有的测试代码目前会提示用户为给定的原型生成新的技术数据管理流(TDMS)文件。 随后的每个VI将最少的测试结果存储到这个文件中。 测试执行软件只需要自动创建此TDMS文件，然后执行测试代码的其余部分，并以根据约定生成结果。
用户管理	用户管理不是一个主要事项，因为Jonathan将负责设计、开发和操作这个测试执行软件。
并行测试功能	Jonathan每次仅对一个原型进行测试，因此不需要担心UUT的数量。
单元/设备跟踪和序列号扫描	由于测试是在设计原型上进行的，因此没有分配的序列号需要跟踪。 相反，Jonathan会通过操作员在运行时输入的唯一名称来跟踪每个UUT。
测试部署工具	Jonathan不打算将这个代码部署到其他测试仪。 他的测试台是设计实验室独有的，与制造设施独立开来。
维护	这个项目完全属于Jonathan。 他将实施和维护任何被选中的测试执行软件。 他不打算文档记录他的工作，因为他将是唯一一个操作和使用这个测试执行软件的人。

表2. Jonathan对测试执行软件解决方案的需求主要是现有验证代码的简单自动化。

Jonathan决定自行开发一个测试执行软件。 他没有复杂的序列执行或报告需求，也没有计划将此系统部署到其他用户或测试站。 如果购买商业解决方案，是不会看到投资回报的，因为大多数功能都派不上用场。 相反，他只需依靠所掌握的软件知识和以前购买的应用软件，就可以开发一个序列执行器来满足其需求，而且只需短短10天即可完成。

Jonathan使用LabVIEW软件开发测试执行程序。 他通过一个简单的界面构建一个解决方案，使操作员能够调用一组已确定的测试步骤并选择TDMS日志的路径。 当为新原型引入了额外的测试时，Jonathan偶尔会对序列执行器进行小的改动。 总的来说，由于部署了这个序列执行器，设计实验室的工作效率提高了。

在本例中，开发测试执行软件Jonathan满足其需求的最佳选择。 通常，内部开发解决方案是需要序列执行或完全自动化时采取的第一步，并且与生产环境的需求相比，可能更适合于测试台应用。

如果...

- 几个月后，设计实验室聘用了一位新的测试工程到，并开始协助测试过程。 这个工程师将如何学习序列执行工具的操作方法，或有效地解决显示的任何错误或漏洞？
- Jonathan调到另一个部门或离开公司。 那么更新或维护序列执行软件所需的知识又该如何维护？
- 当序列执行器有必要执行整个原型的功能评估。 它如何将不同工程师用其他语言编写的额外测试代码、不同的编程方法和报告技术整合在一起？
- 测试执行软件需要移植到生产环境中，以确保测试的一致性。 这些解决方案能否满足这些需求？

实际场景 2

Dave的公司正在设计一个新的功能测试仪，部署到生产线的最后。目前，UUT测试是通过手动执行一系列现有的零散的代码片段来实现。这个过程显着限制了生产线的产量，而且Dave希望在该过程的自动化中使用测试执行软件。该公司没有对测试执行软件进行标准化，每个小组通常都是从有限的商业解决方案和无数的定制解决方案中选择自己所需的。

下表列出了Dave对测试设备提出的要求。

功能	实现
测试序列开发环境	必须具有能够支持测试执行软件主要功能的高效开发环境。 环境必须能够支持LabVIEW、.NET和Python代码的排序。
自定义操作界面	Dave最终需要一个为公司定制的操作界面。 除了"运行“按钮，他还想删除大多数功能。
序列执行引擎	这对于系统解决产量问题是 不可避免的。
结果报告	目前，每次测试单独将数据记录到SQL数据库。 因此，需要测试执行软件能够整合所有结果，将总结果传送到数据库并用序列号进行标识。
用户管理	大多数与测试仪的交互由生产环节的操作者进行。 Dave希望通过用户管理工具或可自定义的界面，删除操作员的开发权限。
并行测试功能	只要测试仪的吞吐量与生产吞吐量相匹配，Dave就不需要同时测试多个UUT。
单元/设备跟踪和序列号扫描	序列号用于标识每个组件和组装的UUT。 条形码扫描器用于跟踪此类信息。 测试执行程序必须能够在执行的多个测试之间传输此类信息。
测试部署工具	Dave需要将最终产品部署到另外10个测试仪中。
维护	测试工程部门将维护测试执行软件，无论是内部解决方案的全部功能还是COTS选项的相关功能。

表3. Dave对测试执行软件的评估主要基于对功能测试仪的吞吐量需求。

为了做出决定，Dave还从财务角度对测试装置进行权衡。他估计，新的测试仪将包括一个大型高性能PXI机箱和嵌入式控制器的组合。基于评估UUT所需的测试，机箱将包含从数据采集卡和模块化仪器（如数字化仪和任意波形发生器）到射频测试设备等多个模块。不论测试执行解决方案如何，每个测试台的成本将在10万美元左右。

在评估软件堆栈时，Dave指出，购买COTS解决方案会增加项目成本。测试执行软件的开发许可证需要几千美元，每增加一个测试仪的部署许可证将增加了500美元的成本。

Dave相信他可以通过在Python中构建自定义解决方案来节省测试执行软件的成本。该语言是开源的，而且开发环境是免费的 - 他们相信这两种优势将抵消自行开发测试执行程序所需的额外时间。

测试工程团队精通Python，在要求的时间截点内提供核心功能 - 顺序序列引擎、数据库连接和复用现有测试代码。测试执行软件已成功部署到生产线。测试工程师偶尔会被调用来修复一个或多个测试仪的漏洞。

如果……

- 生产线上的生产需求增加，导致现有测试执行软件无法满足产量需求。 这时需要增加并行测试功能。
  - 尝试实现此功能需要多少额外的开发时间？ 这会如何影响自定义与COTS解决方案的成本比较？
  - 假设由于Python语言的多处理限制，测试仪的吞吐量无法满足需求。Dave的团队需要购买额外的硬件来复用当前的解决方案，或者从头重新开发另一个测试执行软件。 这将如何进一步影响自定义与COTS解决方案的成本比较？
- 测试工程团队不能总是为其他重要事项维护或升级测试执行软件。
  - 当这种需求出现并且团队无法提供帮助时，生产将如何受到影响？ 因此导致的停机如何影响系统的维护成本？
  - 测试工程团队花在维护测试仪的时间应如何量化？ 这个因素如何影响系统维护成本？

实际场景 3

Karen在一家设计和生产小型医疗设备的公司工作。 每个产品都有自己的全自动化生产线。虽然每个小组都采用测试执行软件进行最上层的系统管理，但该公司还没有对解决方案进行标准化。 最近，一个新的测试经理就职，对测试执行软件标准化很感兴趣。Karen的任务是选择商业解决方案、现有内部产品还是进行新的开发来实现测试执行软件的标准化。

Karen为负责每个产品的各个小组制定了以下要求列表。

功能	实现
测试序列开发环境	测试开发人员需要一个灵活的开发环境，特别是可以与其LabVIEW和VB.NET代码连接。Tortoise SVN用于源代码控制，因此开发环境需要与此工具集成。
自定义操作界面	测试经理希望根据正在开发或测试的产品自定义操作界面。 操作员表示他们希望在监督测试仪时有一个进度指示器来指示测试状态。
序列执行引擎	这是所有测试仪必需的功能。
结果报告	所有生产系统必需符合公司的HTML报告标准。
用户管理	测试工程团队由几个系统架构师和较多的测试开发人员组成。 测试经理想要分离这两个角色之间的职责。
并行测试功能	在对组装单元进行功能测试时，生产线会一次评估一个UUT。 但是，板卡级测试应该优化提高执行速度。 为了满足所有测试仪的需求，需要进行并行测试。
单元/设备跟踪和序列号扫描	公司每个产品和板卡的UUT信息通过操作员输入进行跟踪。
测试部署工具	公司有一个专门的测试工程师团队，负责编写测试代码。 这个团队必须能够将实验室的开发环境部署到所需的生产环境。 目前，这是手动完成的。
维护	测试经理需要一个正式的维护计划作为标准化工作的一部分。这个计划的一部分需要适应公司今年晚些时候当前系统生命周期结束时进行的操作系统迁移。

表4. Karen对测试执行软件的兴趣源于对各种测试仪进行标准化的需求。

考虑到这一标准，Karen排除了所有现有的内部解决方案。因为这些解决方案大多数是针对某个功能进行开发的。架构的一致性很低，无法扩展到其他生产线，特别是在序列生成需求、操作界面自定义和高效部署实践方面。此外，当软件出现问题或者对设备进行修改时，难以跟踪哪些测试工程师负责特定测试执行程序。

因此，Karen向她的经理提议采用商业解决方案。测试执行软件由知名的供应商提供，该供应商的其他硬件和软件工具应该已经应用在测试仪中。此测试管理软件的现成即用功能可以满足测试仪所需的序列执行要求，并采用要求的特定报告格式。测试执行软件包括一组可满足其他测试仪需求的工具，包括用户管理工具和部署实用程序。鉴于由商业供应商对其进行维护，Karen的经理不必担心软件无法兼容之后的操作系统迁移。

Karen的公司最终成功地做出了决定。总的来说，测试执行软件提供了一个灵活的框架，适用于不同的生产线。购买的测试执行软件进行标准化给公司带来了额外的好处。此外，由供应商负责提供培训，以便于测试工程师能够适应新软件。测试执行软件的购买还包括维护合同，由供应商提供常规补丁和升级。公司还可以获得技术支持资源，可以帮助解决其测试序列问题。

商业解决方案仍然是Karen公司的标准。当测试工程师由于晋升、退休或离职需要换人时，测试经理可以聘请具有测试执行程序开发经验的人员。该公司成功地从过时的操作系统迁移了两个完整版本，同时仍然保留所选择的测试执行软件。随着新产品的开发，可扩展架构可以持续满足生产需求。

结论

无论公司规模、行业或个人测试标准如何，都必须部署测试执行软件来实现最上层的系统管理。 这意味着需要引入一定程度的抽象，将系统的常见功能与测试代码的特定功能分开。 在构建最终解决方案之前，必须对测试执行软件的需求进行完整的评估。 许多测试工程师都在构建还是购买测试执行软件之间犹豫不定。 做出决定需要从成本、功能和维护角度仔细考虑每个解决方案的优势。

下一步

TestStand是业界标准的测试管理软件，可帮助测试和验证工程师更快速地构建和部署自动化测试系统。 TestStand包括一个可立即运行的测试序列引擎，支持多种测试代码语言、灵活的结果报告和并行/多线程测试。

虽然TestStand包含许多现成功能，但仍设计为高度可扩展。 因此，全球成千上万的用户选择了TestStand来构建和部署自定义自动化测试系统。NI提供培训和认证计划，每年培养和认证了超过1000名TestStand用户。

了解更多关于TestStand

测试系统构建实用指南

硬件和测量抽象层

Grant Gothing，ATE研发经理，Bloomy Controls

目录

引言

从初始规划到硬件和软件开发再到最终集成，自动化测试设备(ATE)的设计和开发提出了诸多挑战。 在该过程的每个阶段，更改都变得日益困难，且实现成本日益增加。此外，由于在开发周期中软件通常在硬件之后，因此许多开放式项目都留给软件工程师处理。良好的规划有助于降低常见的风险，但无法防止所有问题，特别是在快速的测试开发周期中，许多问题都出现在最终集成阶段。软件比硬件的适应性更强这一观点经常导致“用软件解决就好了”这一误解。 但是，硬件和软件是紧密耦合的，大部分的问题通常需要对两者同时进行更新。 这不会随着初始部署而结束，而是会持续出现在系统的整个生命周期中。

随着产品越来越复杂，测试它们所需的系统也越来越复杂。 由于ATE仪器成本日益重要，将仪器复用于多个产品的能力就显得非常必要。 而且，不断缩短的开发时间要求工程师并行开发软件和硬件，但开发要求通常没有明确的定义。 另外，测试系统部署之后，长产品生命周期意味着仪器故障或淘汰以及产品和测试需求的变更可能会给测试设备带来诸多挑战。因此，模块化、灵活性和可扩展性对于成功开发自动化功能测试系统至关重要。

从硬件的角度来看，这通常通过使用模块化仪器和具有可互换测试连接件的互连系统来实现。 但如何让测试软件适应各种硬件呢？ 硬件抽象层(HAl)和测量抽象层(MAl)是实现该任务的其中两种有效设计方法。 与测试序列采用针对特定设备的代码模块不同，抽象层可将测量类型和针对特定仪器的驱动程序从测试序列中分离出来。 由于测试程序通常根据所使用的仪器类型（例如电源、数字万用表[DMM]、模拟输出和继电器）来定义，而不是根据特定仪器，因此采用抽象层会使得测试序列更快开发，更易于维护，更适应新的仪器和要求。 使用硬件抽象来将软件和硬件分开可赋予硬件和软件工程师以并行工作的能力，从而缩短开发时间。 开发用于序列和底层代码实现的通用API可帮助系统架构师维护常用函数库，进而实现标准化和可复用性。 这使得测试开发人员可以专注于每个被测单元(UUT)序列的开发，减少花在编写底层代码的时间。

ATE软件挑战

开发	维护
紧迫的开发周期 需求定义不清晰 测试步骤不断变化 硬件设计完成之前就要开始软件开发 软件和硬件工程师分离	长产品生命周期 ▪ 仪器故障或过时 ▪ 仪器变更 产品更新 ▪ 测试步骤更改 ▪ 需要新硬件 制造工程师通常不是原始的测试开发人员

软件抽象的好处

开发	维护
软硬件分离 序列开发与代码（驱动程序）开发分离 为仪器提供通用API 优化代码复用 缩短开发时间 将架构师与测试开发人员的角色分离	降低被淘汰或硬件更换的风险 ▪ 降低对特定仪器的依赖性 ▪ 无需修改测试序列即可更换硬件 降低代码复杂性，方便未来测试支持/更改 提高代码跨平台兼容性

了解HAL和MAL之间的区别非常重要。 HAL属于代码接口，可使应用软件能够在通用层次上而不是针对特定设备的层次上与仪器进行交互。 通常，HAL定义了仪器类或者仪器必须符合的类型和标准参数和功能。 换句话说，从仪器的角度来看，HAL提供了与仪器通信的通用接口。 MAL属于软件接口，提供了可以在一组抽象硬件上执行的高级操作。 从UUT的角度来看，这些操作是使用多种工具来执行某个任务的一种方式。 这些共同组成了一个硬件抽象框架。

图1. 抽象框架的上层盖图

打印机对话框是HAL/MAL的一个典型的日常用例。 当您通过计算机进行打印时，不必打开终端，只需将原始串口、USB或TCP命令发送到打印机进行初始化和配置，然后发送要打印的数据。 硬件驱动器负责实现配置和打印方法。 所有打印机制造商均遵循特定标准，将这些方法部署到其驱动器中，使得打印机方便易用。 在一个硬件上执行多个任务所使用的通用接口就是HAL。 那么我们需要编写代码来调用HAL的抽象方法，以便配置和打印文档吗？ 不用，当您选择打印时，就会显示打印对话框。 此对话框提供了一个通用接口，用于调整配置参数，并将可打印数据发送到设备上。 这就是MAL，它使您能够直观地运行所有打印机，而无需了解打印机设备的底层功能。 与打印文档一样，ATE HAL定义了每种仪器类型必须遵循的公共底层任务集，而MAL则提供了执行高级操作来驱动仪器的一种通用方法。

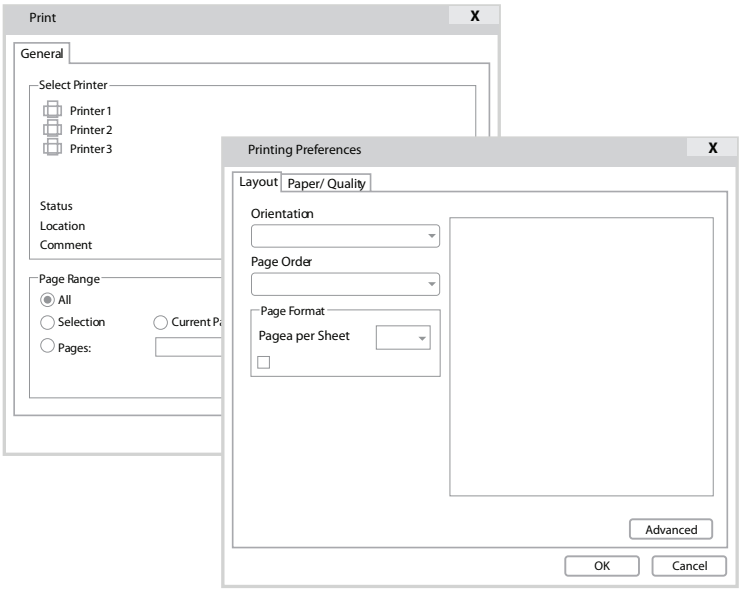


图2.打印机对话框是HAL/MAL的典型日常用例。

现有的HAL/MAL

测试和测量工程师采用了许多方法来解决HIL和MAL。其中大部分方法可以立即使用，或者集成到更大型的自定义HAL/MAL方法中，以便以最少的投入来实现功能扩展。 以下是几个最常见的例子。

抽象	描述	类型	优点	缺点
特定厂商的驱动系程系列 驱动程序（NI-DAQmx、模块化仪器、Pickering PILPXI）	HAL	厂商特定的驱动程序系列为某些供应商的通用仪器组提供通用接口。 这些驱动程序集可以连接每个特定系列的几十到几百个仪器。 示例包括NI驱动程序（例如NI-DAQmx、NI-DCPower、NI-DMM、NI-Scope、NI-SWITCH和NI-FGEN）和Pickering PILPXI。	▪ 为所支持的仪器提供通用的直观接口 ▪ 有详细文档记述且经过测试 ▪ 提供所有可用的功能 ▪ 学习周期短— 同一个驱动程序可控制该系列的所有仪器	▪ 仅对每个厂商的特定驱动程序有效 ▪ 并非所有仪器都支持所有功能
业界标准的接口	HAL	IVI是仪器驱动程序软件的标准，可提高仪器互换性，并在与符合IVI标准的仪器连接时提供灵活性。 该标准定义了13中仪器类型的规范，目前许多制造商都遵循的这一标准，这使得一个驱动程序控制多种类型的仪器。 仪器类型包括DMM、示波器、任意波形/函数发生器、直流电源、开关、功率计、频谱分析仪、RF信号发生器、计数器、数字化仪、下变频器、上变频器和交流电源。	▪ 适用于从USB到PXI等各种仪器 ▪ 兼容众多 GPIB、串行和LXI台式仪器 ▪ 即插即用 ▪ 所有驱动程序采用标准编程模式 ▪ 上层仪器 API ▪ 支持模拟设备	▪ 只指定API，而非实现 - 两个“可互换”实现可能为同一测量返回不同的结果 ▪ 无法用于不兼容的仪器 ▪ 可能无法实现所有功能 ▪ 可能会出现仪器无法支持的功能
Switch Executive	MAL	Switch Executive是一款开关管理和路由应用程序，可允许将兼容的开关矩阵和多路复用器仪器组合到一个虚拟开关设备中。 这个虚拟开关可以使用命名的通道和路由直观地进行配置和激活。	▪ 直观的开关路由设置和操作 ▪ 基于UUT或测试为中心的名称定义通道和路由 ▪ 定义无连接路由以增加安全性	▪ 要求开关兼容NI或IVI标准 ▪ 无法支持通过NI-DAQmx控制的继电器

表1.现成软件抽象层

现成的抽象能够帮助我们只需少量的自定义即可获得诸多功能。但是，这些抽象无法统一起来。IVI驱动程序和NI产品驱动程序是兼容仪器的最佳HAL，但是从以仪器为中心的角度，它们仍然需要开发测试序列。从以测试为中心的角度来看，Switch Executive非常适合抽象开关路由，但它只能用于符合NI或IVI标准的开关连接（无模拟或数字I/O、DMM、示波器、电源等）。通过使用统一的HAL/MAL，您可以更有效地开发以UUT为中心的序列，这些序列可与各种仪器连接，更好地处理仪器通道和连接的变更。

虽然HAL和MAL提供了诸多好处，但它们通常需要工程师基于过去的经验进行大量预测。 我们需要考虑不同层次的抽象。 有些抽象是软件和时间密集型，有些则是现成即用。 一般来说，对特定仪器和测量进行抽象的程度越高，需要的框架规划和软件开发越高级。 构建大型抽象框架非常耗时，并且如果没有正确规划，可能风险非常高。 不恰当的初期假设或实施可能会产生积极和消极的持久后果。 重要的是找到满足您的特定需求的硬件替换产品。 如果您不确定如何进行，那么简单开始，保持可扩展性，并尽可能使用内置的抽象。

背景

为了更好地理解HAL/MAL的实现方式，您必须了解自动化测试软件的结构。 从最高层次看，自动化测试软件采用测试执行程序（或测试执行软件），例如TestStand。 执行程序会调用一系列测试步骤，通常是代码模块或函数，这些步骤使用LabVIEW软件的图形化语言、.NET或Activex进行开发。结合针对特定仪器的自定义方法，这些代码模块便具有特定用途，例如使用DMM和开关的开关式DMM；或使用电源和示波器可进行纹波测量的电源。 这样做有一定的好处，可让每个开发人员能够编写所需的特定功能，但同时需要大量的交叉功能，并且可能难以开发、部署和管理。 而且，它要求每个测试开发人员都精通底层软件（如LabVIEW）。

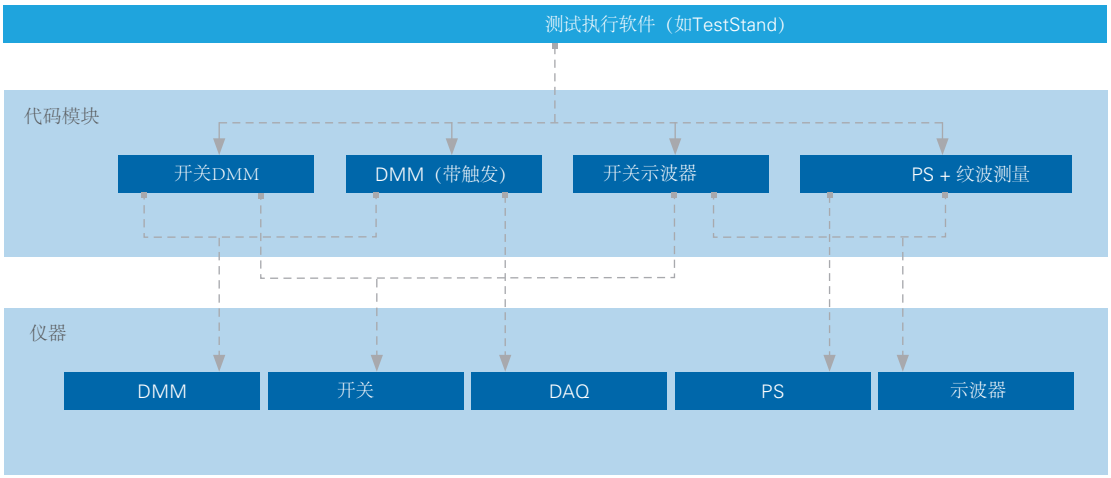


图3.非抽象自动化测试软件的解剖

不使用抽象

如果不使用硬件或测量抽象，就必须使用代码模块，直接引用驱动程序来与代码仪器连接。这导致测试序列与特定仪器和特定驱动程序代码紧密耦合。 不采用HAL/MAL框架时，通常会发生四个不可避免的问题：

- **由于过时或需求变化，仪器需要更改** - 如果不采用抽象，就需要更改每次调用该仪器的驱动程序，在典型的测试序列中这可能涉及几十个步骤。 每次仪器更换都会导致一系列软件更改。
- **由于新需求的出现而导致的驱动程序功能更改** - 如果驱动程序需要更新，则可能需要更新该驱动程序的每个例程以匹配新代码，尤其是在更改输入或输出时。 而且，直接调用驱动程序代码模块需要每个测试开发人员理解所使用的每个驱动程序的内部工作原理，特别是在使用多功能操作引擎的情况下。 要使用所有这些功能，测试工程师还必须是精通软件的工程师。

- **测试序列是从仪器的角度进行开发** -通过使用针对特定仪器的驱动程序，所有测试序列使用以仪器为中心的通道名称（例如使用以仪器为中心的名称开发测试序列）而不是以UUT或测试为中心的名称（例如 5v_Rail、LED_Control、VDD）进行开发。由于我们从UUT的角度开发测试程序，这使得开发和调试变得非常困难。 而且，任何测试变化都需要对仪器、连线和互连系统有非常深入的了解。
- **测试序列的开发通常与硬件开发同步进行**—为了满足紧迫的时间期限，软件和硬件开发通常同步进行。 因此，在开发测试序列时，通常不知道仪器和通道的详细信息。 如果不采用抽象，就需要为驱动程序、通道编号和连接保留占位符。 任何硬件信号更改都需要更新测试序列。

例如，使用自定义方法，多路复用DMM测量代码模块看起来会类似于下图，下图显示的是通用开关式DMM labVIEW VI。 代码模块包含针对特定仪器类型的特定调用集合。 在本例中，这些集合是NI MUX和NI DMM。 该代码模块基于输入通道和开关拓扑结构来连接开关，基于一些输入参数使用DMM进行测量，然后断开开关。 在测试执行程序中，您必须知道要填充哪些字段，以及从仪器的角度来看，需要什么通道、拓扑和配置。 您还必须确保将开关和DMM测量数据正确地传递到代码模块。

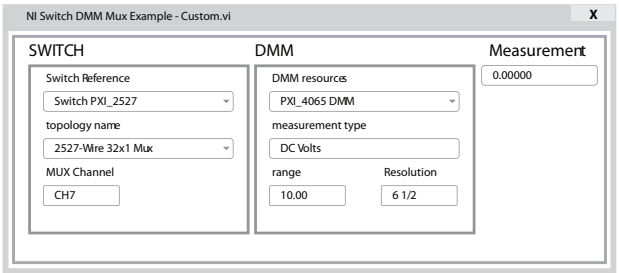


图4.LabVIEW中典型的多路复用DMM测量应用程序前面板

从测试执行程序的角度来看，我们可以调用代码模块来执行特定的功能（复用DMM）。 该函数可对开发该函数的仪器进行特定调用。 以下程序框图显示的是命令调用的嵌套。 在图中，测试执行程序包含调用代码模块的步骤。 代码模块使用驱动程序来与特定仪器进行通信。 每个外部项都取决于其内部调用。

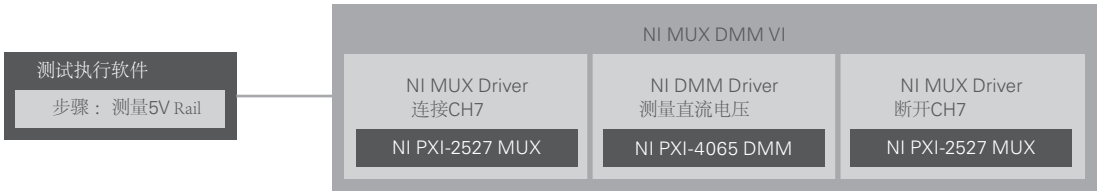
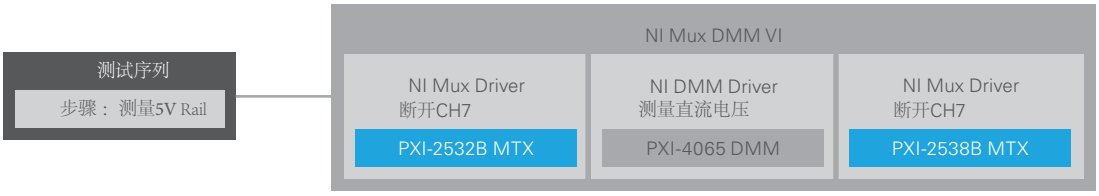


图5.执行多路复用DMM测量的嵌套式命令调用

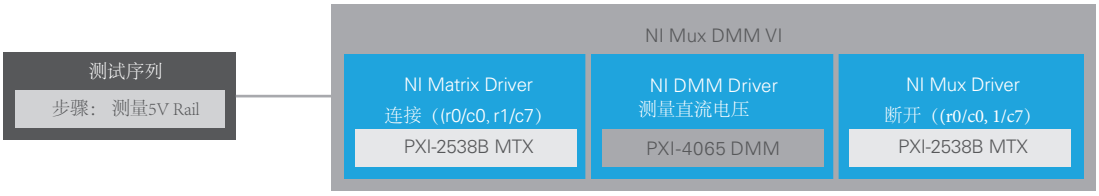
如果必须更改仪器，则依赖关系中的每个函数也必须更改。 例如，如果初始多路复用器的通道不够用，并且需要变换为较高通道数的矩阵，则由于依赖关系链，我们需要进行一系列更改：

1. **仪器**——PXI-2527多路复用器更改为PXI-2532B矩阵
2. **驱动程序**——NI多路复用器驱动程序更改为NI矩阵（行/列而不是通道）
3. **代码模块** - NI Mux DMM VI必须更改为NI Matrix DMM VI
4. **函数调用** - 代码模块的测试执行调用必须更新
5. **序列** - 测试序列必须针对该代码模块的每次调用进行更新

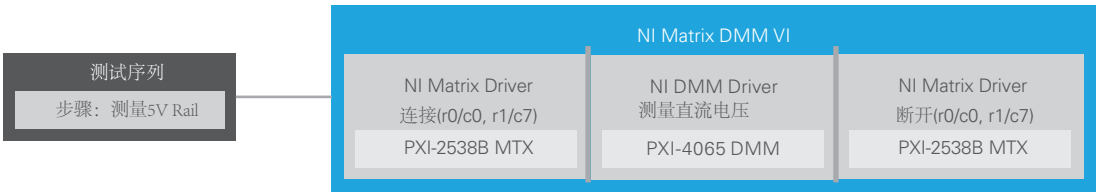
步骤1：更改仪器



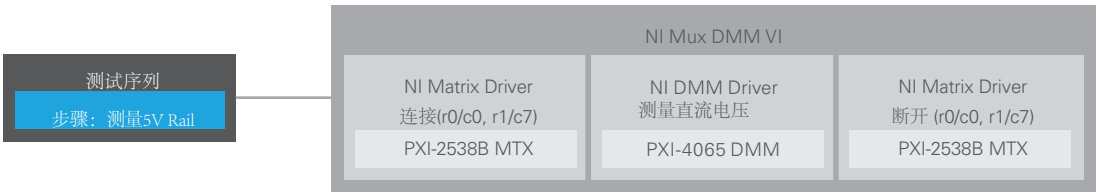
步骤2：更改驱动仪器



步骤3：更改代码模块



步骤4：更改函数调用



步骤5:更改测试序列

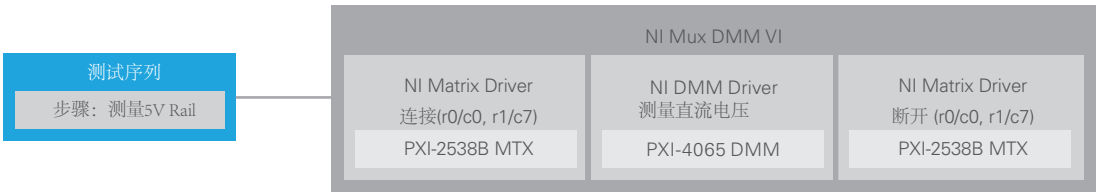


图6. 依赖关系链所需的非抽象更改

采用抽象

硬件和测量抽象打破了测试执行程序和与仪器交互的代码模块之间的耦合。 测试执行程序没有调用直接与特定仪器交互的代码模块，而是与MAL进行交互。 这定义了基于通用仪器类型执行常见任务的操作或步骤类型。 这些操作是仪器通用的，通常具有高级别名称，如“信号输入”、“信号输出”、“连接”、“电源”和“负载”。 它们还会接收针对特定测试的参数（而不是针对特定仪器的参数），如信号名称、连接名称、电源别名、电压/电流和负载方法（CV、CC、CP）。 映射框架使用配置文件将通用操作的特定测试参数转换为特定仪器参数，如仪器参考、通道编号、矩阵行和列、GPIB地址和仪器配置约束。 框架与HAL连接，与配置文件定义的特定仪器进行通信。 它基于MAL操作类型调用每个特定仪器对应的方法，并从配置文件中提取针对特定仪器的参数。

如果将每个步骤视为一份烹饪食谱（煎饼），则配置文件中的信息将是成分（鸡蛋、牛奶、黄油、面粉），操作就是烹饪功能（混合、搅拌、拍打），驱动程序就是厨房工具（碗、搅拌机、扒炉），框架就是将所这些整合在一起的指令。

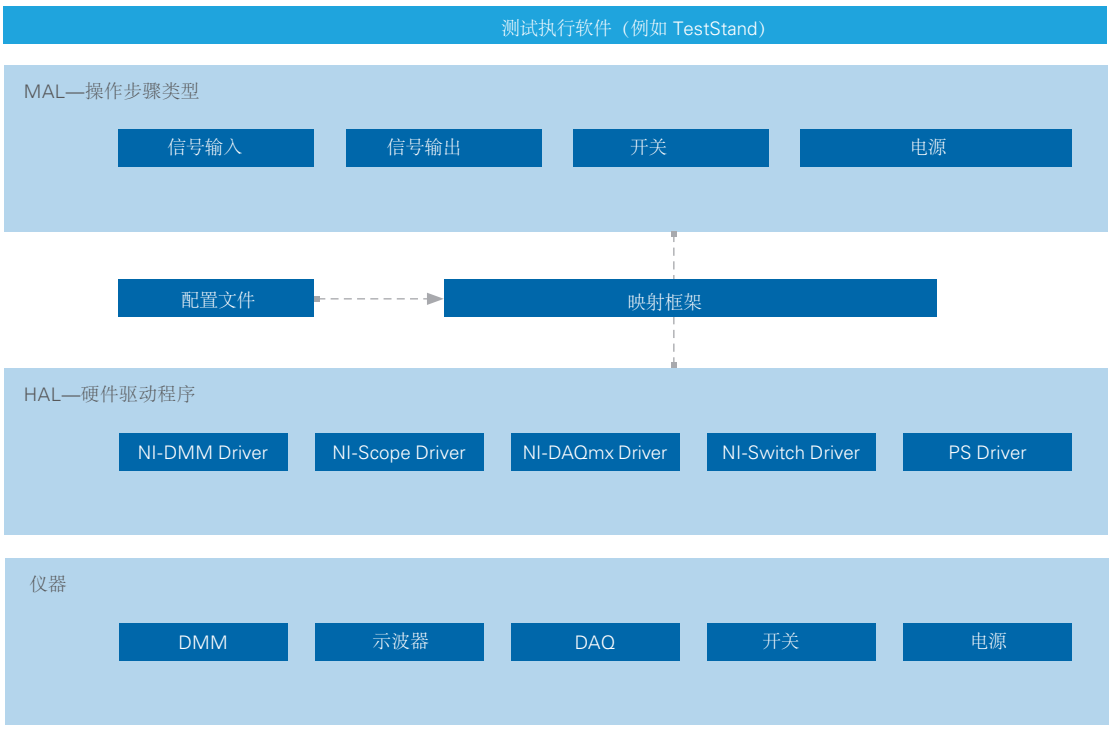


图7.抽象自动化测试软件的解析图

本节将继续介绍基于抽象的多路复用DMM示例。 在本示例中，测试执行程序使用针对特定步骤的输入参数5V Rail来调用通用步骤类型“信号输入”。 在本例的框架中，信号输入定义为三个设备操作： 连接信号路由、读取测量设备数据、断开信号路由。 这些操作使用5V Rail参数传递到映射框架。 映射框架读取配置文件以查找5V Rail的仪器和通道详细信息。 这些对应于PXI-2527多路复用通道7的连接，以及DC电压模式下的PXI-4065 DMM测量。 然后，框架调用适当的抽象驱动程序NI-Switch和NI-DMM，以与配置文件定义的特定仪器进行通信。

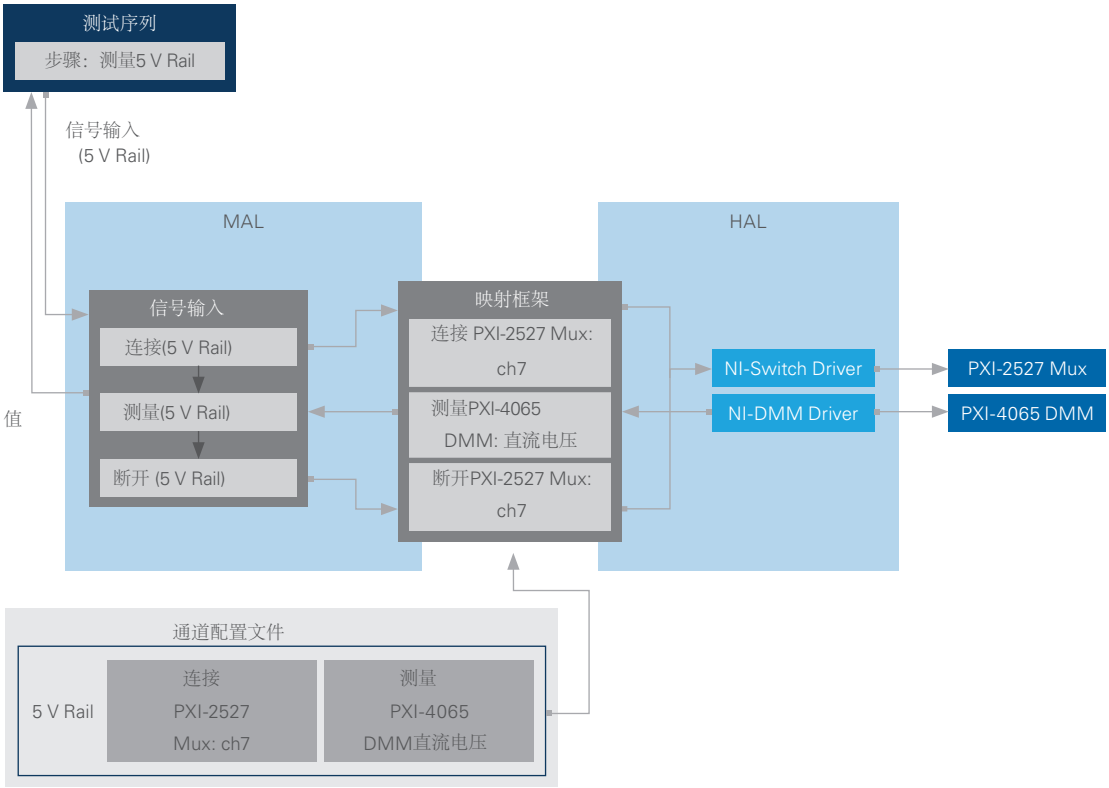


图8. 使用抽象框架对DMM测量进行函数调用

在不采用抽象情况下，PXI-2532B矩阵由PXI-2527多路复用器替代，这时使用HAL/MAL框架来执行相同的更改会更为容易。 由于所有仪器细节存储在配置文件中，并且HAL提供了一个与类似仪器交互的公共接口，所以只需要改变配置文件即可。 通过将PXI-2527 Mux： Ch7替换为PXI-2532B Mtx: r0/c0, r1, c7， 映射框架可自动提取更新的详细信息，并使用新参数调用新矩阵， 完全无需更改测试顺序或代码模块。

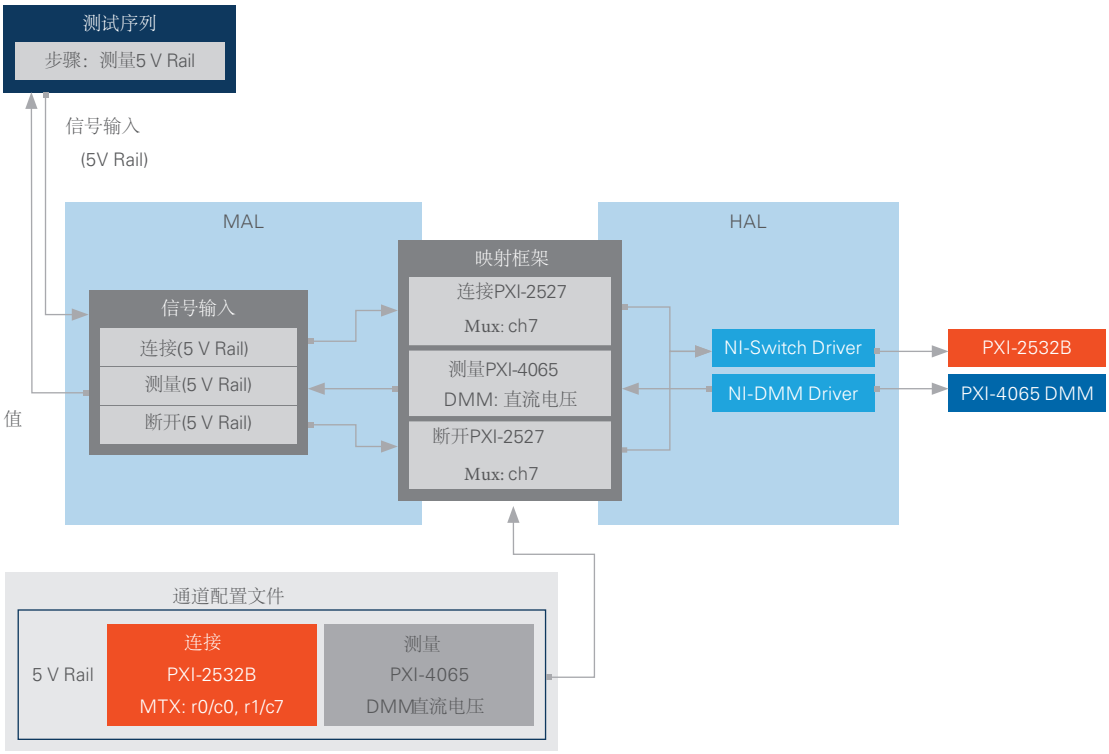


图9.抽象可帮助您以最少的软件更新来轻松更新硬件 - 只需更新配置文件即可。

方法

在决定抽象框架时要考虑的最重要问题是所有其他决策所基于的抽象范围。 一种极端是在没有抽象的情况，每个硬件接口就可直接调用仪器驱动程序。 另一个极端是完全采用抽象，组件、通信协议、测量和配置格式之间的每一个可能接口都有一个抽象定义。 本节探讨了这些可能接口的一些选项。

选项1: 仪器驱动程序

仪器驱动程序方法是自动化测试中最常用的方法，主要是因为该方法只需很少的编码、预测和规划。这种方法需要开发底层代码模块来与特定仪器连接。 这些通常称为底层驱动程序或仪器驱动程序，然后由高级代码模块或直接由测试执行程序调用。 下面的框图显示了针对特定仪器开发的每个仪器驱动程序。 在这种情况下，如果更换仪器，则必须同时更改驱动程序和更高级别的调用。

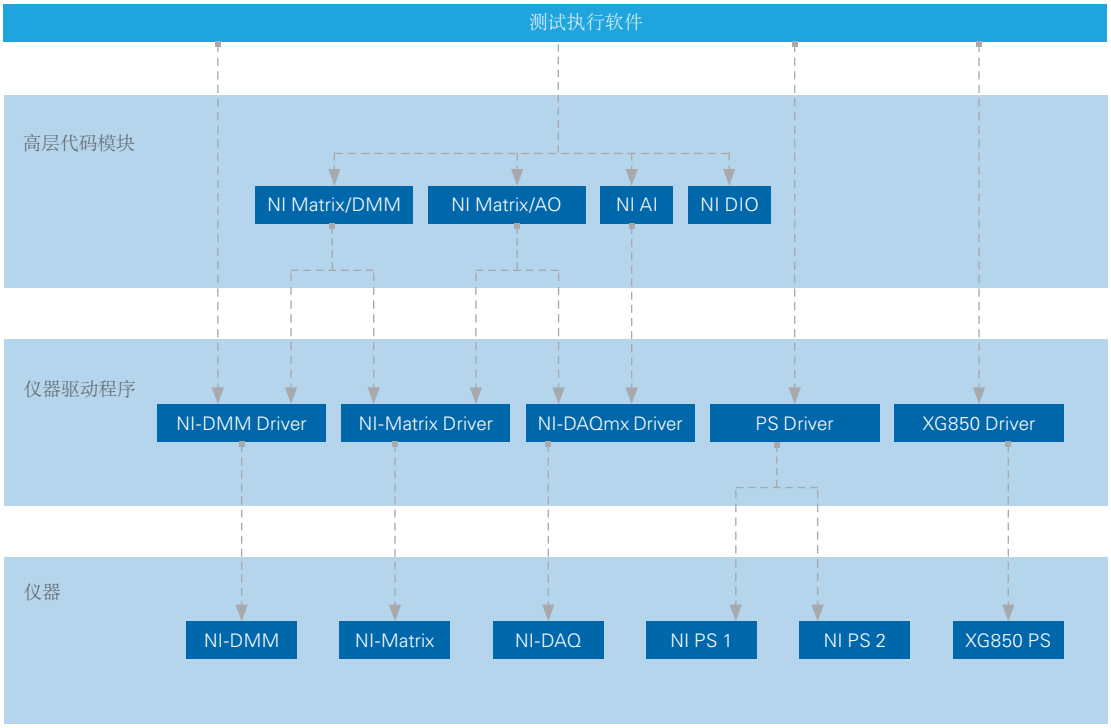


图10.无抽象自动测试软件的仪器特定驱动程序方法概述

虽然此方法不包括任何抽象，但在提高驱动程序开发和交互的鲁棒性方面，仍然有值得我们借鉴的地方：

- 开发或使用仪器驱动程序包来与每个仪器连接。
  - 底层驱动程序包实现了初始化、交互以及关闭与仪器的连接所需的所有功能。
  - 功能应简单、单一。
  - 驱动程序应该能够处理同一仪器类型的多个例程（例如同一系统中的两个相同电源）。
- 开发包装仪器驱动程序，简化仪器界面。
  - 已有的驱动程序可能会包含几个难以理解的函数。 您可以将已有的全功能仪器驱动程序包装到更简单的包装仪器驱动程序中，以提高易用性。
- 确保所有仪器连接都经过仪器驱动程序。
  - 这为所有仪器通信提供了单一进入点，从而简化了调试，减少了竞争条件，并允许通过一个统一的位置管理仪器状态。
  - 包装仪器驱动程序（如果开发的话）应该只有一个入口点。
  - 驱动程序可以由测试执行程序直接调用，或者由更高级别的代码模块调用。
- 不在驱动程序级别实现针对特定测试的功能。
  - 针对特定测试的算法应该由更高级别的代码模块或在测试执行程序中实现。

- 确保仪器驱动程序不会相互识别。
  - 高级代码模块或测试执行程序调用单个仪器驱动程序时应执行多仪器交互。

选项2: 现成即用HAL/MAL

将抽象整合到仪器驱动程序架构内的最快方法是使用已有的HAL和MAL。 虽然目前市面上仅提供有限的完全集成的HAL/MAL抽象框架购买选项，但是许多硬件供应商已经在其仪器中实现了一定级别的硬件抽象；Switch Executive是专门针对开关连接和路由而开发的MAL。 基于这些已有的抽象程序来构建代码模块，您就能够以最小的开发工作量提高ATE软件的适应性和抽象。

现成即用的硬件抽象

已有的硬件抽象会使用常见的底层接口来连接各种仪器。 这减少了所需仪器驱动程序的数量，同时降低了系统中仪器更换的影响。 测试执行程序和更高级别的代码模块可以引用常规的驱动程序，从而减少开发工作量和仪器更改的影响。当实现下面定义的抽象类型之一时，特定接口的I/O是固定的。 因此，仪器更改通常不会导致代码模块更改。

您可以使用两种方式来利用已有的硬件抽象： 仪器系列驱动程序和通信标准。 仪器系列驱动程序往往是针对特定供应商的驱动程序，可以控制该供应商目录中特定仪器类型的许多仪器。 通信标准提供了一种行业标准的方法来与多个供应商的某些仪器类型连接。 您可以使用这些标准来开发仪器驱动程序，以便控制各种类似的仪器。

通过仪器系列驱动程序实现硬件抽象

仪器系列驱动程序是针对特定供应商的驱动程序，可与仪器常用的产品系列进行通信。 与IVI驱动程序类似，仪器系列驱动程序使用通用驱动程序实现与多个不同仪器的通信。 常见的示例包括NI模块化仪器（NI-DMM、NI-Switch、NI-DCPower和NI-Scope）和Pickering PILPXI。 仪器系列驱动程序实现了相关产品系列之间的互换性。 虽然它们不支持跨供应商或跨产品系列复用，但这些驱动程序通常直观，易于实现，并且包含每个仪器的大多数功能（就算不是全部功能）。

基于通信标准的硬件抽象

许多仪器制造商都遵循设备通信的行业标准。 通过遵循行业标准，制造商就可以使其仪器与其他类似仪器进行互操作。 两个最常见的标准是可编程仪器的标准命令（SCPI，通常发音为“skippy”）和可互换虚拟仪器(IVI)。

SCPI

SCPI定义了测试和测量行业中控制可编程仪器的语法和命令标准。 通过这些命令，用户可以设置和查询仪器的常用参数。 SCPI命令可以通过各种通信协议实现，包括GPIB和LAN以及串行协议。 通过开发单个符合SCPI标准的驱动程序，您可以与多个相同类型的仪器（DC电源、电子负载等）进行通信，而无需开发特定于仪器的驱动程序。在开发SCPI驱动程序时，请注意，虽然SCPI定义了一个常用的命令和语法标准，但是不同的供应商有时在实现标准时会有一些微小的差别，使得开发100%标准的驱动程序变得困难。 在选择符合SCPI标准的仪器和开发驱动程序时，请务必要密切注意每个仪器的命令细节。

IVI

IVI仪器驱动程序软件的标准，可提高仪器互换性，并为连接到符合IVI标准的仪器提供灵活性。 该标准使用VISA定义了I/O抽象层。 由于SCPI已经整合到IVI中，因此许多符合SCPI标准的仪器都符合IVI标准。 IVI标准定义了13个仪器类型的规范，许多制造商都在遵循这一标准，这使得每种类型的单个驱动程序能够控制来自不同供应商的多个专用仪器。 仪器类型包括DMM、示波器、任意波形/函数发生器、DC电源、开关、功率计、频谱分析仪、RF信号发生器、计数器、数字化仪、下变频器、上变频器和交流电源。 许多PXI和台式仪器都遵循IVI标准，而且许多编程语言和测试执行程序已经有一些现成的驱动程序。

通过使用IVI驱动程序为IVI兼容的仪器开发测试序列和代码模块，不同供应商的仪器看起来就会一样。 您可以针对每种类型使用一个驱动程序集，以连接各种可互换仪器。 相比使用仪器特定的驱动程序，如果使用类似功能来替换兼容IVI的仪器，则代码和序列更新将会大大减少。 然而，尽管IVI驱动程序可以实现兼容仪器的大多数功能，但是一些仪器可能仍需要特定代码来执行自定义功能。 而有些仪器则可能无法处理所有符合IVI标准的功能。最后，尽管两个仪器可以执行相同的IVI函数，但得到的结果可能不一定相同。因此，每做出一次更改，都需要验证和测试仪器的功能。

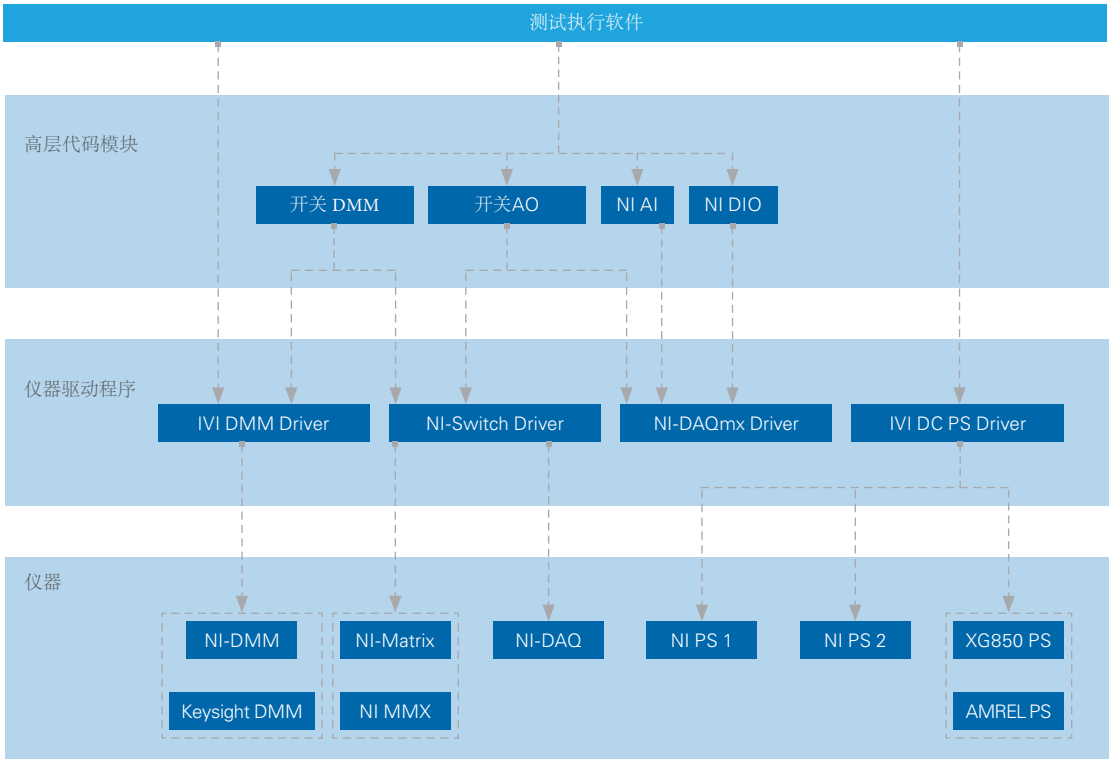


图11.采用现成抽象的自动化测试软件概述

现成即用的测量抽象

虽然仪器普遍已经具有硬件抽象，但它仅从仪器的角度来实现抽象。 测量抽象是非常有限的。由于测试系统具有高度自定义性，因而很难为测量操作定义一个标准。 最常用的现成即用测量抽象层是Switch Executive，这是一款开关管理和路由应用程序，可允许将兼容的开关矩阵和多路复用器仪器组合到一个虚拟开关设备中。 这个虚拟开关可以使用用户命名的通道和路由直观地进行配置和激活。虽然Switch Executive仅适用于符合NI-Switch和IVI标准的设备，但却提供了一种极佳的方法来从UUT或测试的角度定义开关路由。

首先，Switch Executive提供了一个图形化配置实用程序，用于在单个仪器和多个仪器之间设置开关通道名称和路由。 行、列、通道和路由组都可以进行配置和命名，以便直观地设置开关机制。

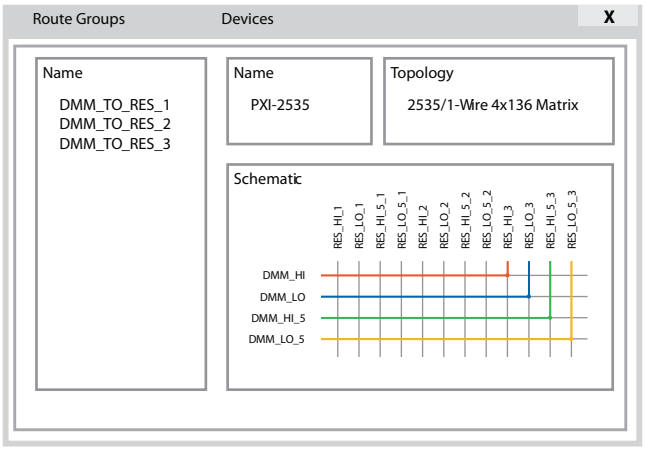


图12.Switch Executive MAL配置接口

其次，Switch Executive可集成到LabVIEW和TestStand中，提供了强大的接口，可按名称设置和查询预配置的路由。当与TestStand测试执行软件一起使用时，Switch Executive可以逐个步骤执行，以便在执行步骤的代码模块之前为开关仪器提供命名接口。

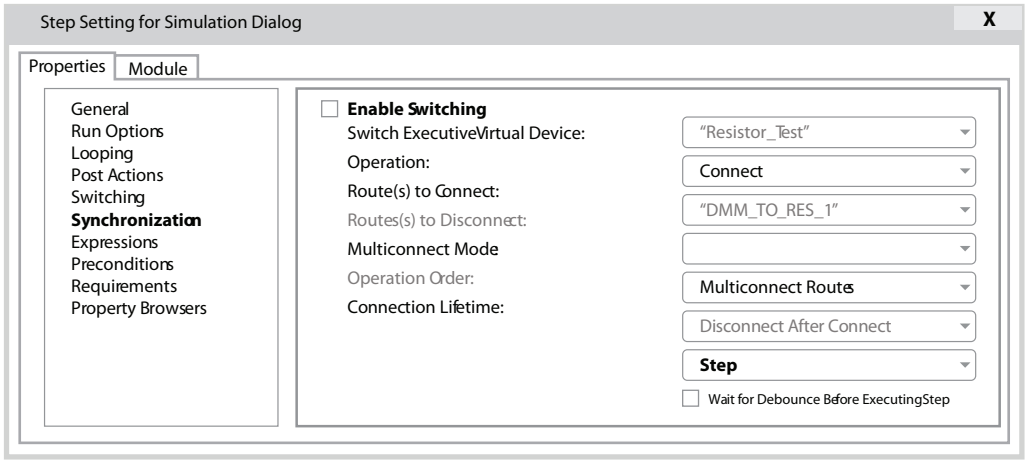


图13.Switch Executive MAL测试设置

Switch Executive是一个有用的MAL，可将开关连接抽象为特定测试名称，而不是特定仪器名称。 当与IVI开关硬件抽象结合使用时，就形成了一个非常出色的集成HAL/MAL框架。 但是，Switch Executive不适合用于使用非IVI开关或外部数字输出控制继电器的情况。 此外，Switch Executive仅适用于开关路由，无法扩展到其他测量类型。 如果需要开关之外的集成HAL/MAL框架，就需要自定义开发代码。

选项3: 集成式HAL/MAL框架

集成式HAL/MAL框架提供了一个结构来实现由测试执行程序(MAL)调用的高级动作，与底层驱动程序接口相连接，并在两者之间映射细节。 此框架由三种主要类型的代码模块实现： 动作、映射框架和硬件驱动程序。 这些代码模块类型都是由一组API定义。 API是一组用于软件应用程序的工具（函数、协议、参数、语法），定义了代码模块的运行方式及其与周围软件的交互方式。 在基本的HAL/MAL框架中，有四种常见的API： 测量API、配置API、硬件驱动API和仪器API。 代码模块、API及其之间的交互如下所示。

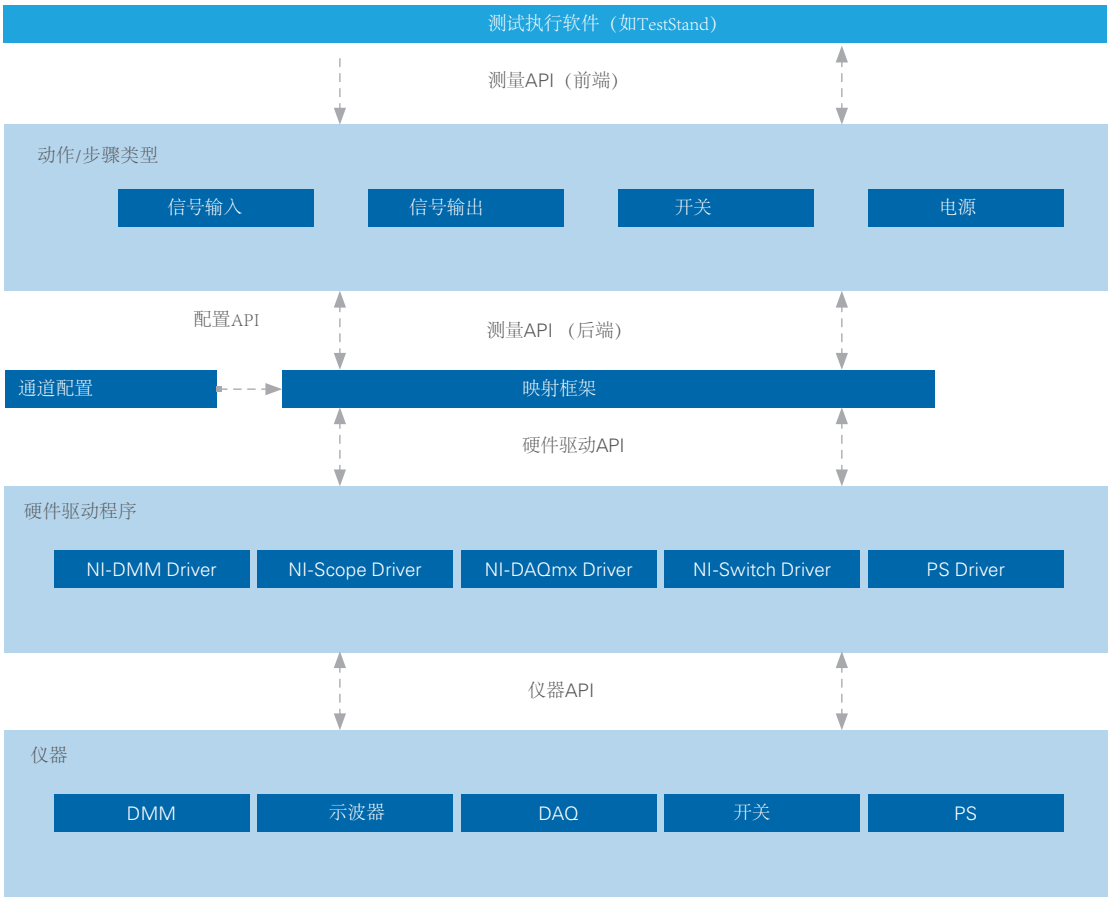


图14.采用集成式MAL和HAL的自动化测试软件概述

三种类型的代码模块是：

- **动作/步骤类型** - 动作定义了MAL的功能。 特定动作定义了每个测量类型（输入或输出）。 动作可以简单到调用一种仪器类型的一个函数，例如进行开关连接； 也可以复杂到调用多个仪器的多个函数，例如将开关连接与设置电源电压、电流和启用状态组合。 这些代码模块通过测量API来定义它们的方法和参数。
- **映射框架** - 映射框架是使用配置文件中的默认值将高级动作链接到底层仪器设备的内部代码。映射框架代码模块通过硬件驱动程序API与硬件驱动程序交互，并通过测量API与操作交互。
- **硬件驱动程序** - 硬件驱动程序代码模块将通用设备类型函数调用（DMM、电源、开关等）转换为特定仪器通信（SCPI、IVI、NI-DCPower和专用通信）。 因此，硬件驱动程序在一端实现硬件驱动程序API，在另一端实现仪器特定的API。

HAL/MAL抽象框架至少包含以下四个API的其中一个：

- **测量API** - 测量API定义高级动作及其特定参数。 这是MAL的定义。 测量API定义了所有动作必须遵循的公共框架，然后允许每个操作定义执行特定功能所需的API（参数和方法）。每个动作必须至少实现后端测量API，映射框架使用该后端测量API将人类可读别名链接到特定的开关和测量仪器以及适当的信道。或者，可以开发API的前端，为每个操作提供更直观的接口。这个前端通常是一个配置对话框/向导。 信号输入的测量API示例可定义信号输入别名和返回值输出。 API还可为别名定义连接、测量，然后断开连接。

- **配置API** - 映射框架使用配置API填写有关如何从测量API转换为硬件API的详细信息。 配置API用于定义配置文件或数据库的参数、语法和内容。 只有映射框架使用此API。 例如，配置API可以规定配置文件为Microsoft Excel文件，并且每个信号别名应该具有以下属性： 名称、类型、连接详细信息、仪器、仪器配置和换算关系。

- **硬件API** - 硬件API是定义特定类型仪器必须实现的通用参数和方法的抽象API。 此API定义了HAL。 例如，DMM Hardware API可指示所有DMM必须能够初始化、配置（电压、电流、电阻、量程、分辨率）、测量（返回值）和关闭。

- **仪器API** - 仪器API由每个单独的仪器定义，因此不是抽象层。 每个仪器特定的硬件驱动程序负责实现用于控制其特定仪器的必要函数和命令。 这是将在仪器特定代码接口中使用的同一个API，并且负责实现用于该特定仪器的特定通信协议和命令。

为了更好地理解代码模块和API之间的交互，请查看多路复用DMM示例， 其中详细解释了每个代码模块的输入和输出。

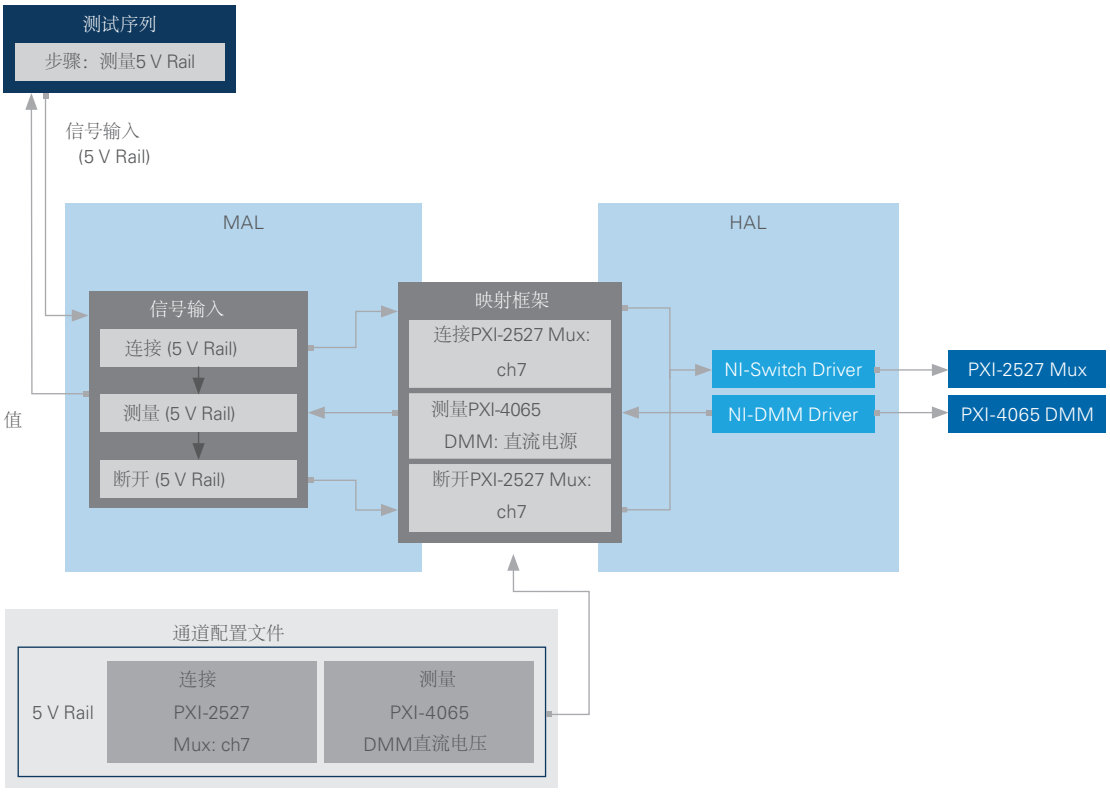


图15.使用抽象框架进行复用DMM测量

在该示例中，信号输入块是操作代码模块，定义了信号输入应当执行Switch Device Connect函数、Measurement Device Measure函数，然后执行Switch Device Disconnect函数。 此函数的测量API定义了代码模块需要从测试执行程序接收的别名，并将其传递到映射框架，然后从映射框架获取返回值传递回测试执行软件。

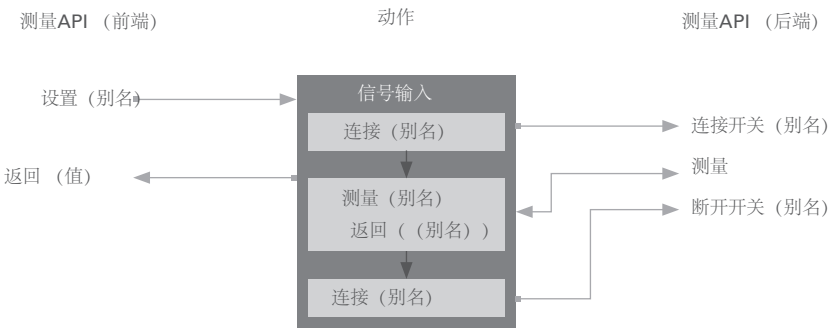


图16. 信号输入的MAL Action API示例

映射框架通过测量API从操作接收命令。 然后通过配置API解析配置文件中的别名数据，以获取正确的仪器ID和参数。 配置API定义了系统配置的文件格式、语法和字段。 然后，映射框架通过硬件驱动API将特定仪器的信息传递到适当的驱动程序。

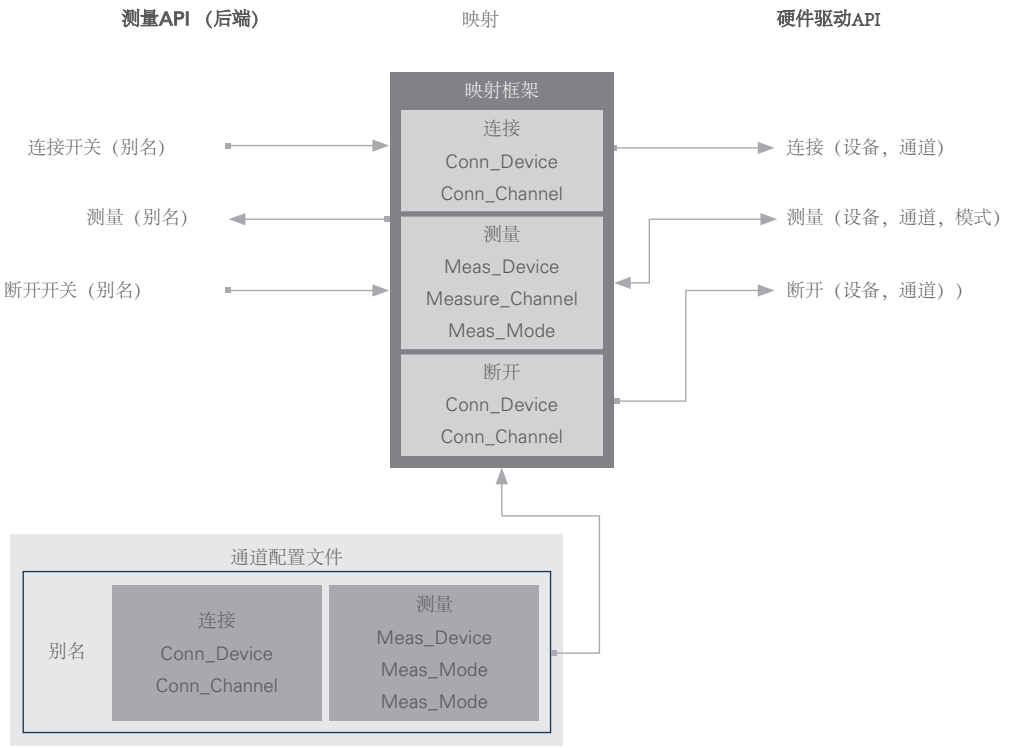


图17.信号输入的映射框架API示例

映射框架使用通用的硬件驱动API调用各个硬件驱动程序。 然后每个驱动程序解释通用设置的细节，并使用现成即用的方法和参数与特定仪器进行通信。

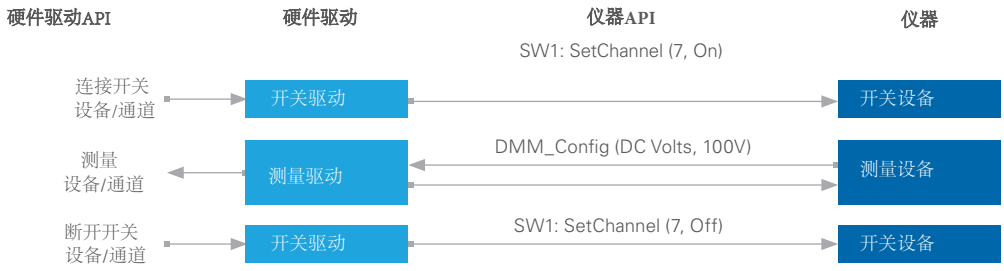


图18.信号输入的硬件驱动程序API示例

选项4: HAL/MAL插件架构

插件是集成框架的有用补充。 一个真正的插件仅是一个软件组件，可以在部署后修改而不需要重新部署整个应用程序。 插件与主应用程序和/或框架分开存储在磁盘上，并在运行时动态加载。

虽然开发插件架构会带来一些挑战，但同时也通过明确地限制添加或修改功能的范围和风险来简化软件回归测试。无插件框架在每次需要新测量类型、仪器驱动程序或配置格式时都必须进行重新构建。由于没有插件，整个源内置到单个EXE中，所以即使对仪器库进行一个微小的改动，也不能保证其不会无意中影响其它应用特征。由于很难知道源修改的所有可能影响，测试必须非常彻底。

插件架构使得开发人员能够添加或修复插件代码，而无需修改或重新部署底层框架，因而提供了最高级别的软件模块化。 这通过编写一个仅依赖于抽象类或模块的框架来实现，该框架可动态加载所需的具体插件，通常仅在需要时才加载。 成功的插件架构依赖于完善的接口设计。 换句话说，为了在测试框架中使用插件，框架必须知道如何调用任何可能的插件组件。如果所有插件都采用一致的软件接口，那么在运行时加载这些插件只需要框架或测试应用程序知道在哪里找到这些插件即可。

尽管这些是抽象框架一些常见的过程、API和代码模块，但它们当然不是唯一的。 每个框架都是独一无二的，并且有自己的要求、过程和实现。 对于一些团队，这种抽象层次可能超过需要。 然而在其他情况下，系统架构师可能需要注入额外的抽象层。 基于框架架构师和用户的需求和能力，这些API的具体实现的解释因人而异。 一些工程师使用简单的操作引擎来实现所有抽象，一些工程师则使用更高级的面向对象编程，一些工程师使用插件，还有一些工程师更倾向于使用单个代码库。 关键是找到适当的抽象和实现程度来适应您的特定需求和能力。 此外需要理解的是，不是所有问题都可以通过抽象来解决，有时仍然需要针对特定仪器的代码。 因此，在开发抽象层时，务必允许开发自定义代码来实现高级功能。这可通过允许仪器参考由更高级别的代码模块或测试执行程序获得来实现。 高级开发人员不应被框架所束缚。

无 ○ 部分 ◒ 全部 ●

抽象选项	选项1 无抽象	选项2 现成抽象	选项3 基本自定义	选项4 采用插件
允许使用以下组件替换各个仪器:				
采用相同通信协议的仪器	●	●	●	●
兼容IVI或产品系列的仪器	○	●	●	●
采用不同通信协议的仪器	○	◒	●	●
无需修改测试序列即可更改仪器通道/连线（只需修改配置文件）	○	◒	●	●
从测试/UUT的角度执行测量/任务	○	◒	●	●
无需修改框架即可添加新仪器或测量	○	○	○	●

表2.抽象层选项的功能比较

实际场景1

您是一名来自商用产品公司的测试工程师，负责开发新产品的电子组件的功能测试。 您需要测试三个PCBA和最终组装。 现在有一个通用ATE仪器平台，但它已经过时，并且先前的测试程序由于设备故障和过时而存在许多问题。 幸运的是，作为该项目的一部分，一个新的ATE平台已经开发出来，通过可互换的测试头来适应不同组件的仪器测量需求。

您的任务是开发测试序列和代码模块软件来与硬件工程师开发的仪器和连接件进行交互。您具备使用测试执行软件（与先前平台使用的测试执行软件一样）相关的经验，并且有几年的软件应用程序开发经验。 之前已经有多次关于如何使用抽象来减缓现有系统过时这一问题的讨论。您必须决定这是否是正确的解决方法以及实现方法。

是否采用抽象……

您必须做的第一个决定是是否要开发一个抽象框架，且不管抽象的级别如何。考虑现成抽象的优势，如IVI，这个决定的答案几乎是肯定的。 不采用抽象的唯一情况是100%已知项目的寿命，并且永远不需要进行任何更改，但这几乎是不可能的。

您是否需要HAL?

下一个决定是使用什么级别的硬件抽象。 这个决定较为复杂，因为这涉及诸多因素。 硬件抽象通常更容易理解，因此比MAL的实现成本更低。 如果可以合理地使用预抽象驱动程序（例如IVI和产品系列驱动程序），则成本将更低。 但是，一旦必须使用不属于某类驱动程序的仪器，则可能需要为每种仪器类型开发一个通用接口。 例如，如果系统具有一些符合IVI标准的电源以及不符合IVI标准的电源，则可能需要开发适用于两种类型的抽象电源定义。 定义抽象硬件定义通常需要了解该特定类型的大多数仪器的工作原理。然后可以使用该信息来定义系统内每种仪器类型的常用方法和参数。

目标是覆盖约80%合理预期的功能。与团队成员谈话，确定每个仪器类型必须通过每个抽象仪器驱动程序实现的核心功能和参数。例如，团队可以规定所有电源的核心功能应该先初始化，然后设置电压/电流/启用状态、回读电压/电流/启用状态，最后关闭。虽然某些其他功能可能以后会被电源使用到，但这些功能并不一定值得成为系统标准化的一部分。如果您对某种特定仪器类型不够了解，或者不确定需要什么功能，请从最小的架构开始。您可以随时添加新功能到标准中，但是在多个驱动程序使用该标准后，很难更改功能的参数或详细信息。

下面的流程图可以帮助您决定什么级别的硬件抽象适合您。如果你不确定答案，可以假设一个更抽象的解决方案，或者比较不抽象的解决方案。更抽象的解决方案需要更多的前期设计，但可以节省长期的时间，而较不抽象的解决方案可以让您更快地运行，但未来可能会出现问题。 要注意的一点是，需要先问一下自己：我是否需要MAL。这是因为如果没有精心设计的HAL，就不能有效地实现MAL。

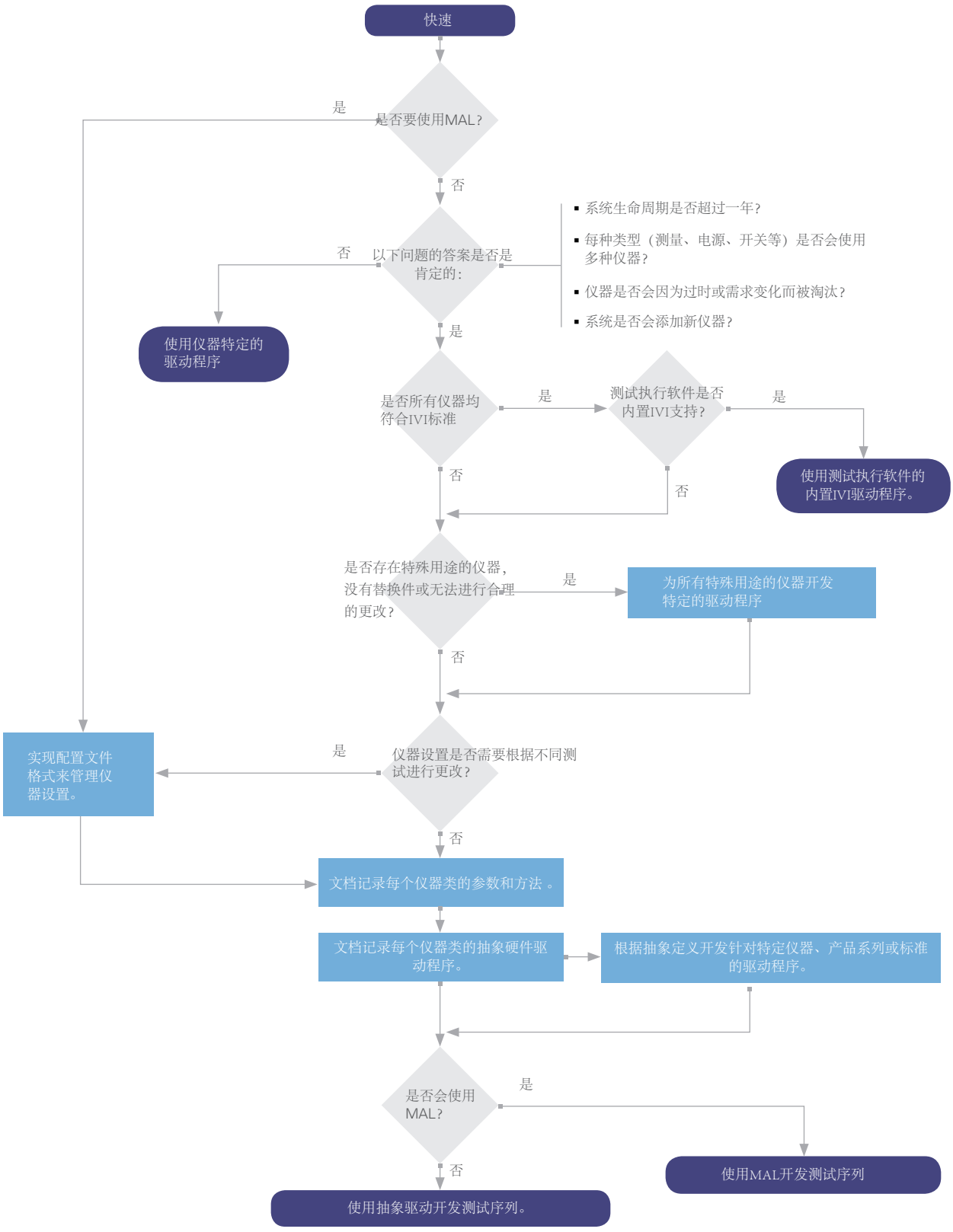


图19.决定要实现的抽象级别的流程图

您是否需要MAL?

HAL的第一个决定是是否需要MAL。 这是因为在不依赖于硬件抽象的情况下，MAL几乎是不可能实现的。 因此，这个问题实质上是在问你是否需要集成的抽象框架。 当团队中有多名测试开发者没有底层软件开发经验时，HAL/MAL是理想的选择。 以下一些主要问题可以帮助您决定是否开发MAL:

- 是否有一个软件架构师可以计划和支持该框架？ HAL/MAL难以在没有架构师/所有者的情况下获得有效的支持。
- 是否有多个测试开发人员只有很少的软件开发经验吗？ 抽象框架的一个最大好处是它缩短了测试开发的学习曲线。
- 系统是否具有长使用寿命，需要支持多种产品？ 这可能需要较大的前期投资，但后期收益将越大，使用频率也越高。
- 您是否愿意开发和支持MAL？ 没有抽象比定义不明确和执行不佳的抽象更好。 简单易用的HAL/MAL可以节省大量的时间；但是，如果过于复杂或设计不当，可能会很麻烦，反而会增加开发和调试时间。

如果你对大多数这些问题回答“是”，那么开发一个集成的抽象框架将会得到长期的回报。

实际场景2

即使抽象的所有好处已知，仍然存在如何衡量成本与回报的关键问题（其中单位通常是时间）。虽然抽象决策的第一部分通常从技术角度考虑，但是成本/效益决策必须在更高的业务层面做出。

成本是多少钱?

这是一个难以回答的问题，因为它主要取决于开发人员的经验、编程能力和所需的抽象级别。 但是，您可以估计各种组件的大致数量级，如下表所示。

类别	任务	描述	估计小时数（低）	估计小时数（高）
规划	架构定义	记录动作类型、设备类型以及动作和设备之间的常用接口类型	24	48
	每种设备类型的HAL定义	记录每种类型设备的输入和输出和方法	8（每个设备）	16（每个设备）
	每个动作的MAL定义	记录每种类型的测量/动作的输入和输出和方法	8（每个操作）	16（每个操作）
	配置定义	定义配置文件或数据库的格式、语法和内容	24	48
实现	映射框架开发	实现所有的软件，将配置文件映射到动作和抽象驱动程序 - 大多数底层框架在此处开发	60	120
	抽象设备驱动开发	为每个设备类型进行抽象设备接口代码的软件开发 - 实质上是在构建仪器	4（每个设备）	24（每个设备）
	仪器驱动开发	为使用HAL的每个仪器特定驱动程序进行软件开发 -填写到每个特定确定的模板中	4（每个仪器）	24（每个仪器）
	动作开发	为MAL定义的每个动作进行软件开发 - 实现前端和后端API来与测试执行程序和映射框架连接	4（每个动作）	24（每个动作）
总计		开发框架（不包括单个仪器驱动程序）的总时间 - 假设五种设备类型和五个动作，每个设备一个仪器特定的驱动程序	248	776

表3.抽象框架任务和成本

这表明完全集成的HAL/MAL抽象层所需的开发时间可以低至250小时，也可以超过750小时。 取决于抽象的级别，该时间甚至可以超过1000小时。

如何降低成本？

当涉及到软件开发时，成本与复杂性密切相关。 复杂性有好也有坏，取决于其性质。 目标是增加良好的复杂性，同时避免不良的复杂性。 可以增加功能的复杂性是好的。 每个特性通常会增加功能。 使用可扩展、灵活的模块化代码来实现这些目标可能较为复杂。 但是当以简练的方式实现时，这种复杂性是有益的。 由于规划不当、冗余功能和繁琐的意大利面条式代码而产生的复杂性是不好的，因为它增加了开发成本，但没有增加功能。

您可以通过四种方式降低ATE抽象框架的复杂性：

- **预先规划您的架构。**与大多数开发过程一样，前期规划和文档可以节省大量的时间和避免开发过程中的麻烦。 事先规划和文档记述API和代码模块有助于减少交叉功能和不必要的相互依赖，这将使得代码更加强大，并减少不必要的复杂性。 您不必计划每个API和代码模块的每个细微差别，而是定义软件的主要交互、参数和基本功能。
- **不要想得太超前**。 在开发大型架构时，工程师往往会过度设计并尝试规划所有可能的情况。 虽然前瞻性的方法可能是好的，但最好是针对已知的情况进行设计。 工程师在设计系统时经常会考虑最坏但通常不会发生的情况。 这只有20%的概率，但却需要80%的时间。 你将最终花费更多的时间来尝试处理假定的边缘情况，而不是专注于将被大多数时间使用的软件。
- **认清一个事实，抽象并不能解决一切问题。** 抽象非常有用，但试图抽象出每一个可能的接口并不值得。 相反，可考虑将自定义硬件交互作为框架的一部分，来解决通用接口无法实现的情况。 为您的系统制定切实可行的规则，以减少抽象层。 例如，将配置文件限制为一种格式（ini、xls、database）以减少映射框架的复杂性，或将操作限制为三个独立的硬件调用，以防止需要实现递归硬件驱动程序API调用。
- **保持灵活、可扩展和模块化。**虽然灵活性、可扩展性和模块化会增加复杂性，但它们是开发大型架构的最佳工具。 此时非常适合采用插件架构，因为插件架构可定义低级框架，而细节通过唯一的代码库实现。 这意味着新功能可以在旧功能的基础上进行扩展，而不会干扰已经存在的功能。 精心规划的插件架构不仅可满足已知情况的需求，还可在未来根据新挑战进行扩展。

值得吗？

即使实现得很好，抽象框架的开发可能非常耗时，但因为其收益往往大于付出而得到广泛应用。 几个关键因素可以提高回报率，使您的框架更成功。 其中许多回报可以通过节省的时间或精力来量化。 下表列出了与任务相关的一些典型成本，并比较了非抽象系统和使用HAL/MAL抽象框架的系统之间的差异。

任务	估计的时间（标准）	估计的时间（抽象）	WHY THE PAYOFF?
为新测试工程师提供测试软件平台培训	60小时 每个工程师	40小时 每个工程师	掌握如何使用抽象框架通常需要了解测试执行程序以及框架。在任一情况下，开发人员必须了解如何与测试系统硬件交互。在使用仪器特定驱动程序时，工程师必须知道每个驱动程序的详细信息以及如何使用它们。而在学习抽象框架时，工程师只需要理解要执行的高级动作，因为仪器细节留给框架。通常，这些高级动作比各种仪器特定驱动程序更直观、更容易实现。
基本功能测试序列的开发和调试（由有经验的工程师进行）	每个序列 80 小时	每个序列 40 小时	测试序列开发变得更快，因为硬件的细节被保存到序列中的一个固定位置，而不是每个驱动程序调用。测试从UUT的角度与硬件交互，可允许序列更直观、更好地匹配测试步骤。一般来说，直观易懂的框架可以将开发和调试时间缩短一半。
由新工程师进行测试开发和调试	每个序列 120 小时	每个序列 60 小时	如果是新的或经验不足的工程师开发测试序列，开发时间的回报将会放大。由于框架规定了一组规则和功能，与使用仪器特定的驱动程序和代码相比，经验不足的工程师可以更好地使用已存在的步骤来开发序列。此外，直观的框架可让产品测试工程师无需掌握底层软件语言即可开发序列。
由于故障/过时仪器或新仪器需求而需要更新测试序列	驱动程序开发 8 小时 加上每个序列 4 ~ 20 小时	驱动程序开发 8 小时 8 小时 加上每个序列 <1小时	当系统中的仪器需要更换时，测试也必须随之进行更改。对于非抽象平台，这意味着必须为新仪器更新驱动程序调用的每个实例。仪器引用得越多，更新的时间越长。而采用抽象框架时，工程师虽然需要开发一个新的仪器驱动程序，但在开发完成之后，只需修改配置文件/数据库即可。
将测试序列转移到新ATE硬件平台	每个序列 40 ~ 80 小时	每个序列 <8 小时	有时，整个系统都会升级，所有测试都必须迁移到新系统。通常这些新系统会采用完全不同的仪器。这种情况下，无论是否使用抽象框架，都必须开发新的驱动程序，但是这些驱动程序开发出来之后，必须更新测试序列才能使用它们。如果使用非抽象序列，这一过程非常麻烦，有时还不如重新编写序列来得简单。而抽象序列通常可以在不到一天的时间内更新，所有这一切都通过配置文件来完成，而不必触及测试序列软件。

表4.非抽象和抽象系统中不同任务的相关成本

对于前面假设的场景，您可以使用这些数字来看看是否或何时开发集成的抽象框架。

首先，假设您自己开发所有四个测试序列。您必须从开发框架开始。在标准的非抽象情况下，您必须开发针对特定仪器的驱动程序。第二个场景可使用现成的抽象来开发驱动程序。对于第三个场景，您需要开发集成式HAL/MAL。

任务	开发时间（标准）	开发时间（现成抽象）	开发时间（集成的HAL/MAL）
框架/驱动程序开发	80 小时	100 小时	500 小时
测试开发（4个测试）	80 x 4 = 320 小时	80 x 4 = 320 小时	40 x 4 = 160 小时
总计	400 小时	420 小时	660 小时

表5.集成的HAL/MAL需要最多的前期开发工作。

在完成初始开发时，集成的HAL/MAL方法比标准方法多出大约240小时，而现成抽象仅多出20个小时。 然而，初始开发结束并不意味着测试程序的结束。

六个月后，研发部门发现需要进行更多的测量，系统中的32通道多路复用器已经不够用了，需要使用4 x 128矩阵代替。 现在您必须开发一个新的驱动程序，并更新每个测试序列以使用矩阵而不是多路复用器。如果您使用现成的抽象驱动程序，则不需要进行任何驱动程序开发来处理新矩阵，并且序列中的函数调用不需要更改 - 只有详细信息需要更改。而如果使用集成的HAL/MAL，序列更新将仅需要在通道配置文件中完成。

任务	开发时间（标准）	开发时间（现成抽象）	开发时间（集成的HAL/MAL）
新驱动程序开发	4 小时	0 小时	0 小时
新矩阵需更新2个简单的测试序列	2 x 4 = 8 小时	2 x 2 = 4 小时	2 x 1 = 1 hour
新矩阵需更新2个复杂的测试序列	2 x 16 = 32 小时	2 x 12 = 24 小时	2 x 2 = 2 小时
额外时间	44 小时	28 小时	7 小时
总计	444 小时	448 小时	667 小时

表6.集成的HAL/MAL方法更易于更新，但仍需要更多的开发工作。

即使到现在，集成抽象层的投入还没有开始得到回报，而现成即用的硬件抽象几乎持平了。现在想象一下，一个新的测试程序出现，需要测试四个程序集。但很遗憾，你太忙了，不能自己开发这些序列，招聘了两名新测试工程师。 你必须对他们进行系统培训，然后让他们开发序列。

任务	开发时间（标准）	开发时间（现成抽象）	开发时间（集成的HAL/MAL）
培训/学习时间	60 x 2 = 120 小时	50 x 2 = 100 小时	40 x 2 = 80 小时
测试开发（4个测试）	120 x 4 = 480 小时	100 x 4 = 400 小时	60 x 4 = 160 小时
额外时间	600 小时	500 小时	240 小时
总计	1,044 小时	948 小时	907 小时

表7.当开发更多测试或需要重大更改时，集成的HAL/MAL方法将得到长远的回报。

此时，最初开发框架的500小时投入已经开始得到回报，比标准开发方法少了100小时。 随着新测试的开发、产品的改变以及产品的生命周期持续，初始投资将持续得到回报。

使用抽象还有许多更主观的回报，无法仅通过一个数字来呈现。 开发测试的时间也大大减少，因为HAL/MAL使得硬件尚未完全定义之前的软件工作更易于展开。 通过维护标准框架，您可以确保一个统一的存储库，并添加新的驱动程序和测量、管理错误以及减少工程师之间的代码分歧。 标准化有助于使每个人（测试工程师、制造工程师和技术人员）保持一致，从而更好地支持系统。 虽然本文档中还详细描述了无数其他优势，但还是让您的能力和ROI计算来帮助您了解什么级别的抽象适合您。

下一步

TestStand

TestStand是一款行业标准的测试管理软件，可帮助测试和验证工程师更快速构建和部署自动化测试系统。 TestStand包含可立即运行的测试序列引擎，可支持多种测试代码语言、灵活的报表生成和并行/多线程测试。 尽管TestStand包含了许多现成即用的功能，但是它也具有高度可扩展性。 因此，全球数以万计的用户均选择TestStand来构建和部署自动化测试系统。 NI提供了培训和认证计划，每年培养和认证了一千多名TestStand用户。

了解有关TestStand的更多信息

关于Bloomy

Bloomy致力于电子功能测试；航空电子设备、电池和BS硬件在环(HIL)测试；航天系统集成实验室(SIL)数据系统提供产品和服务，同时提供一流的labVIEW、TestStand和VeriStand应用程序开发服务。 Bloomy是NI 24年的联盟合作伙伴，自联盟合作伙伴成立以来就加入，是NI最重要的白金级和精选级合作伙伴。

了解更多关于Bloomy UTS软件套件（套件包含了集成的HAL/MAL)

测试系统构建基础知识

机架布局和热分布

目录

介绍

每时每刻，新的项目都放在测试工程师的桌前，期望他们开发出不仅能够满足规格要求和发布期限，而且能够提供高质量和可靠性的测量系统。最理想的情况是，工程师有充足的时间和资源进行深入的研究、建模和仿真，以创建完美的系统。不幸的是，在实际应用中，项目进度通常无法允许足够的时间和资源来开发完美系统。Control Engineering在2014年8月进行的系统集成研究表明，只有67%的系统项目能够在预算内按时完成。面对着紧迫的发布时间和苛刻的项目时间要求，重要的是要考虑测量系统中可能影响测量质量的各个因素，这反过来更有可能增加时间进度、成本和性能的风险。这些方面包括选择的仪器、连接和电缆的质量以及测量方法的实现。然而，我们经常会忽视热量对测量质量和测量系统可靠性的影响。

本文提供了更多关于如何设计来避免风险的知识。了解热量如何影响测量质量，查看基本设计方法，以及探索用于设计机架测量系统的热建模工具。

机架设计中热量的重要性

在通用测量系统中，热量的来源有很多个；然而，在机架安装测量系统中，机架内部产生的热量以及与机架周围环境的热交换是影响测量的热变化的主要来源。因此，你需要关注热量的原因有以下几个：

- **良好的设计实现** – 了解热量的影响并在设计系统时将其考虑在内是一个很好的做法。在了解热量如何影响系统后，您就不会让热量成为影响系统性能的主要变数之一。请记住，在指定的温度范围之外运行仪器可能会影响仪器的质量和预期寿命，这也是设备需要具有良好热设计的一个原因。
- **系统不确定性** - 热量将始终存在并且难以完全消除。因此，通过更深入地理解热量是什么，您可以更好地考虑系统不确定性，并在测量误差和测量结果中更准确地解释它们。
- **系统稳定性** - 稳定性对于良好的测量系统非常重要。如果观察到易变性，通常很难确定根本原因和/或如何解决它。由于这种易变性，系统中的热变化可能导致错误的测试结果。控制系统中的热量有助于最小化这种风险。
- **产品质量** - 产品需要特定的热环境来确保最佳性能，特别是在调整过程中。将系统热量对产品性能的影响降低到最小有助于提高整体产品质量。

热量和仪器

对于仪器，热量是一个不可忽视的考虑因素。 仪器只有遵循特定温度要求才能满足规格。 大多数仪器都会存在温度漂移，如果温度不稳定或超出规定范围，则测量结果会有所不同。如果要正确理解和信任测试解决方案的测量结果，就必须了解这种影响。

例如，看看一些相关行业，如电信和IT都已经按照推荐的或者允许范围内的温度范围采取了最佳的实践方式。大多数设备制造商都会遵循这些最佳实践。 一些设备也有其自己的规格，因此设计目标是满足特定设备的规格以及行业的最佳实践。这些行业的主要关注点包括长期可靠性、系统正常运行时间和较低的总拥有成本(TCO)，这与自动化测试非常相关。类似地，自动测试工程师还应考虑与机架系统和热量相关的潜在影响。

这些行业如果没有对热量进行良好的管理，可能会导致更高的运营成本。 例如，如果冷却系统故障，温度上升会给系统的其余部分增加负荷，导致设备寿命缩短。 如果温度过高，IT系统可能会遇到CPU级的计算错误，从而导致应用程序错误。 我们可以部署备用的冷却系统，但这会提高TCO。 最重要的是，由于自动关机导致的停机会导致服务中断，任何停机都会导致资金损失。

国家和国际标准

关于国家标准，网络设备建造系统(NEBS)和美国采暖、制冷和空调工程师协会(ASHRAE)分别为电信和IT设备制定了指南和最佳实践。

ASHRAE是一个侧重于多个领域实践的组织，而NEBS是专门针对电信设备。ASHRAE可能会参考NEBS中关于机箱和机架安装设备的一些指导，但ASHRAE的最佳实践似乎更全面地涉及外壳设计的所有方面。

虽然这些国家标准不直接适用于测试和测量认证，但它们遵循许多与自动化测试相关的机架设计和性能原则和准则。

国际电工委员会(IEC)制定的国际标准涉及外壳的热方面。 外壳制造商主要参考这些标准来设计或测试外壳或为客户提供使用指南。

- IEC 61587-1规定了在室内条件下空机箱（即机柜、机架、机框和机箱）的环境、测试和安全要求。
- IEC 62194-1提供了在室内和室外条件下评估空机箱热性能的方法。

电信和IT行业的设计目标与测试和测量行业类似，但主要的重点领域和挑战有些不同。 测试和测量的设计目标更多地侧重于满足单个特定设备的规格，因为目前还不存在用于测试和测量设备机架的通用标准，但是该行业的各个公司各有自己的最佳实践。 主要的关注点是确保每个仪器以及被测设备(DUT)保持与其规格一致。 这一点甚至比长期可靠性或正常运行时间更重要，因为我们主要在较高温下关注长期可靠性或正常运行时间，而测量准确性在相对较低的温度下也有可能降低。

在挑战方面，自动化测试系统存在更多的限制。其中一个是，测试机架通常处在人进人出的环境或不受控制的生产厂房环境中。这增加了房间/位置的总体热分布随机性，这一点与只有静止物体的服务器室不同。

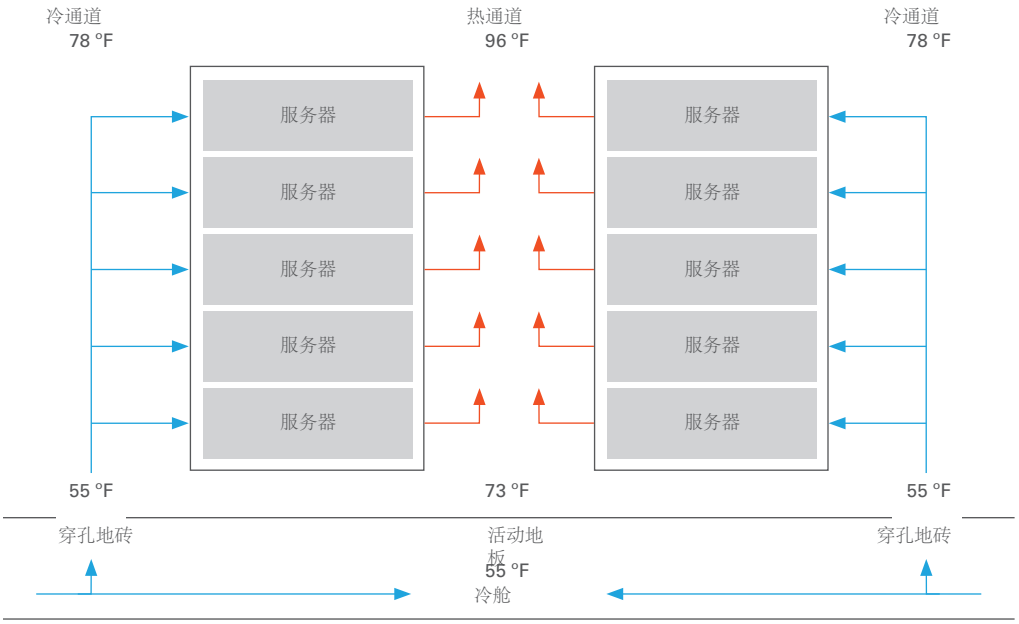


图1. 服务器机房/设施具有受控环境，与放置在不受控制的吵杂环境中的测试系统不同。

热对DUT、测试系统或测试结果的影响

第一个也是最根本的影响在于仪器和DUT的精度。 如果您无法确保仪器处在正确的环境温度下，则精度将降低。

例如，如果环境温度在调整阶段和验证阶段之间的变化足以改变仪器或DUT的精度，则校准可能是无效的。 这也可能导致制造中的通过/不合格结果不准确。 管理此风险的最佳方法是在计算测量不确定度时考虑热偏移。

即使温度在指定的操作范围内，来自不同测试站的数据也会有一些差异。开发环境中采集的数据与生产环境中采集的数据也是如此，其中测试站周围的环境温度变化是偏差的主要原因。

虽然存在偏移，但大多数仪器都是在环境温度下工作。这意味着，即使环境温度发生微小变化，也可能会转化为仪器温度的变化，从而增加不同测试站的数据存在差异的可能性。

由于温度差异可能导致开发和生产的数据存在偏差，因此这也可能发生在验证和确认以及测试开发之间。验证和确认通常是在办公室环境中的台式设备上执行，而测试开发则通常在受控环境中使用机架安装式测试台来进行。 这导致仪器所处的完全不同，即使仪器和随后的测试系统完全相同。 某些仪器甚至在某些温度范围内具有不同的测量规格，因此在设计和测量计算中使用正确的规格很重要。

此外，前面讨论的环境都无法完美地模拟生产制造环境，因此如果环境中存在未知的信息，建议在设计中采取保守态度，以解决这些环境可能出现的问题。 例如，下图对办公区和受控环境中测试台前端几个小时内的室内环境温度数据进行了比较。

受控环境是一个配有专用空调的小房间，可以更加清晰地看到温控器开启和关闭，温度变化更加明显；该房间温度保持在约23 ℃至25 ℃。 办公区温度较为稳定，但稍微高一点。

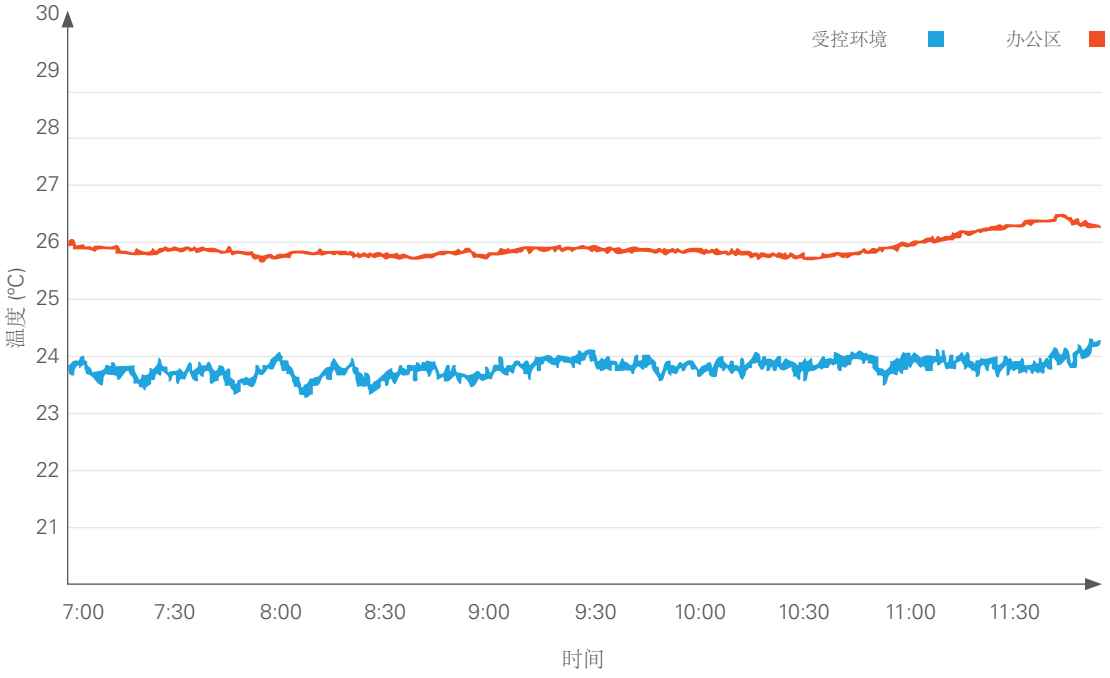


图2. 室内环境温度

一天刚开始时两个温度均轻微升高（图表的右侧）；当人们到达时，由于体热和开门，室内温度有所升高。 请注意，办公区域的温度可能会随着时间、位置、楼层和其他因素而变化。 相比之下，受控环境温度由于配有专用空调而在一整年中都相当稳定。 基于所有这些事实，在进行验证和确认或开发测试和采集数据时必须始终跟踪环境温度。 如果验证和确认的数据和测试数据之间存在差异，跟踪环境温度有助于进行数据分析。

机架安装系统的热分布

在考虑系统中的热分布时，很多人会对其进行过度简化。 最常见的简化莫过于“底部冷，顶部热”的观点，还有人认为温度梯度在机架上是均匀分布的，但在大多数情况下，这些简化并不完全正确。

图3. 通常假设上热下冷，热量从下岛上均匀分布，这一假设会导致结果不理想。

在实际系统中，热分布受到多个变量的影响，因此，热分布也会不断变化。 如果使用红外测温枪或热电偶适当地测量系统，您将看到机架系统存在局部热区，且水平或垂直轴上的温度梯度不均匀等特性。 这是因为我们并不是隔离其他一切来处理冷热空气; 热分布取决于机架布局、单个设备的风扇速度、入口和出口通风口的位置、系统中每个设备的功耗，以及系统中所有风扇的组合所形成的气流。

这一点非常重要，因为这意味着机架顶部不一定是最需要注意的，这直接与热量最常见的过度简化观点之一不一致。了解每个系统独特的热分布特性以及解决相关的问题需要进行全面的评估。

以下示例有助于解释在典型的机架安装式测试系统中，热能是如何工作。

图4. 局部热区或导致整个测试机架的不同位置具有不同的温度。

首先注意的是局部热区，这取决于系统的机架布局和设备规格以及使用情况。 在本测试站示例中，电源周围的热量最高，其次是PXI机箱。 测试站其他部分的温度似乎比这些局部热区低，因此根据每个系统，可能需要以不同的方式来处理这些局部热区。 另一个要注意的是，底部热区在x轴上的分布并不均匀。

热不均匀性是怎么产生的？

通常，这种不均匀性源于设备的气流模式。 例如，电源由单独的供电单元和电源机箱组成。从图中可以看到，机箱的气流从左流到右。 这意味着大部分暖空气在仪器的右侧。

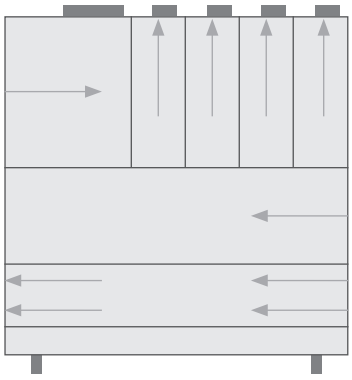


图 5. 模块化电源的气流流向顶视图

后端的排气来自各个模块；不是所有的模块都在同一时间运行，所以后端的温度通常不会像右侧那样高。此外，机架系统的热分布根据使用情况而变化，因此需要进行特性分析，而不是简单地观察特定情况的温度。

系统的热分布应该是什么样的？

您需要了解机架式系统的几个方面，以确定热分布的情况。 首先，您需要了解每个系统的独特需求：

- 所需的系统级规范是什么？
- 系统将在什么环境下运行？
- 需要使用哪些仪器，这些仪器的温度要求是什么？
- 只需将温度维持在一定的温度范围，还是应用需要温度保持稳定？

例如，如果您的DUT是PXI模块，并且需要在切换DUT时启动和关闭DUT PXI机箱，则机架的热分布会以重复的方式变化。 了解这些重复的变化需要了解机架热分布中的任何不稳定性。

最后，并非机架内部的所有点都要求保持相同的温度。通常有一些区域的温度比其他区域更高，但是要求所有仪器进气口的气流保持在指定温度范围内。

设计方法

下一节将重点介绍从设计到推出开发机架式系统的最佳实践。

在开始机架设计之前，您应该了解正在使用的仪器的几个关键元素，这将对设计产生全面影响：

- **评估设备的进气口和出气口**

  首先，研究一下您的仪器，了解模块上进气口和出气口的位置。为设备提供温度要求的能力很大程度取决于仪器进气口处的温度以及散热的位置。了解这将有助于您成功地画出机架设计的热分布图。

- **了解规格和温度要求**

  通常，设备会有特定的存放、运行和校准温度要求。你关心的是哪一部分？每一个温度代表什么意思？ 了解每种仪器规定温度的方式以及温度对仪器性能或规格的影响，特别是那些会影响保证性能的规格。例如，3458A规定环境温度必须保持在23°C±5°C才能保证设备的规格。 此外，如果在上次自动校准后温度相对变化±1 °C，便需要再次执行自动校准。 第一个规格是环境的绝对温度，第二个是相对于上一次校准的温度。 需要理解这些差异及其对解决方案的可能影响。

- **了解器件的环境定义**

  大多数传统的台式仪器将环境温度定义为环境的空气温度。 通常，对于PXI产品和机箱，环境温度定义为紧接在机箱风扇入风口外部的空气温度。 因为像PXI-1045这样的机箱需要约1.75英寸的间隙来允许正确的空气流量，您可以安全地假设这个间隙内靠近进气风扇的温度即是环境温度。

  常见的错误是假定仪器的环境温度就是仪器所处的房间的温度。 一般来说，如果您在桌面上使用仪器，附近没有会产生影响的热源，这个观点可能是正确的； 然而，在机架安装设计中，必须将机架内部的局部空气温度视为仪器的环境温度。 机架设计中的仪器更容易受到热问题的影响。

  根据具体应用和仪器的使用，务必准确了解每个设备的环境温度。

在选择机架或进入机架设计的任何其他细节之前，请了解机架可能遇到的预期热负载。 这可以通过为系统要使用的所有电子设备进行功率预算来轻松完成。 了解功耗有助于了解热负载。

所有电子产品的功率预算

在设计时考虑所有仪器和外设，包括测量设备、PC、显示器、电池备份以及机架内任何可能的发热源。 对于这些设备，请参考产品规格以确定每个设备的功耗。 一般来说，产品规格列出了最坏情况下的功耗（在完全使用或满负载下），这通常不代表设备的一般或平均性能。 通常，在设计中一般以60%的额定最大功耗为基准。 话虽如此，将来您可能会将您的机架设计用于其他目的，这可能会导致热负荷增加，因此在计算功率预算时应留有余量。

理想情况下，如果您可以提前测量设备的实际功耗，则可以获得最佳的整体功耗预测； 然而，这在规划系统阶段基本是不可能的。 建议在系统设计完成后再返回查看，并进行这些测量和文档记录。

温度要求和气流分布

基于仪器温度要求和气流分布，绘制仪器所需的一般位置。热量由下往上流通，因此，机架通常越往底部越凉，越往顶部越热。 设计时将最敏感的仪器放在机架的底部。您可以使用技术为其他机架位置中的仪器创建可接受的热环境，但这通常需要一定的代价。

可用性可能会限制某些仪器的放置，需要评估和了解该仪器放置的影响。 也许它需要额外考虑如何处理气流或如何为不太常规的设备进气口提供冷空气。 在设计时需实时记住这一点。 同时，还要考虑仪器规定的任何间隙限制。 通常，仪器会规定与设备或设备的出入口保持一定距离。 请确保满足这些要求。

基于布局的机架尺寸

布局可让您了解安装仪器所需的机架尺寸。在机架选择时应记住考虑外部约束，例如占地面积和房间高度。

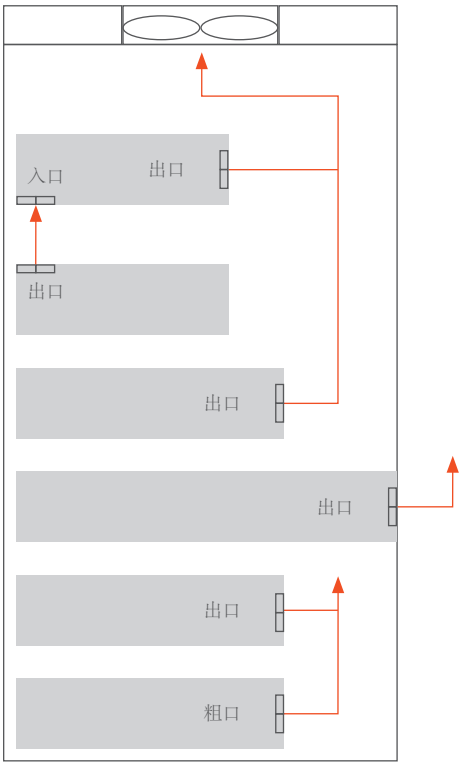


图6. 机架系统中仪器阻止热气流出的示例。

所有仪器的废气应通过一条通畅的通道来排出机架。 在此布局示例中，您可以看到整个仪器阻塞其下方仪器的排气。 另一个错误的做法是让热空气和冷空气相遇。 您可以通过更加巧妙地布置机架来解决这些问题，但首先要了解仪器的入口和出口以及预期的气流。

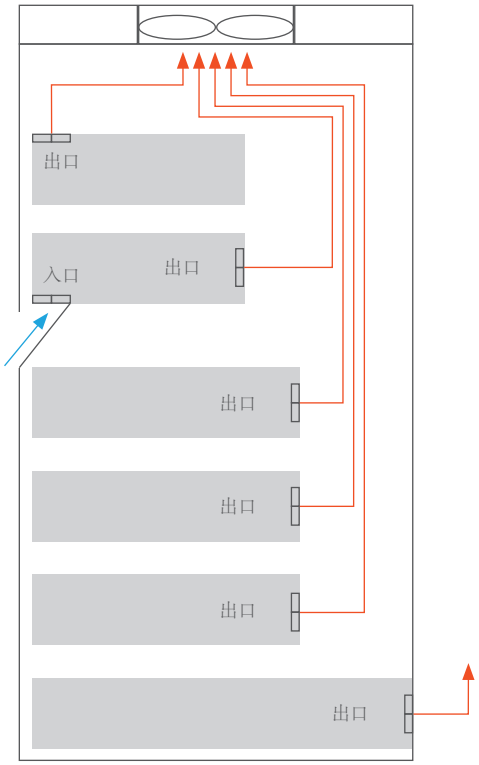


图7. 机箱系统正确排气布局示例

重新放置仪器，为所有仪器的热气提供连续的气流路径。 此外，通过机械分离来确保所有仪器尽可能吸进来自外部的空气。

对于仪器的进气口位于机架内部的情况，基于仪器的安装方式，一个仪器的废气可以再循环到另一个仪器的入口。多个仪器的废气可以进入彼此的进气口。 这可能会显着增加机架内的环境温度，并且越朝上，温度越高。

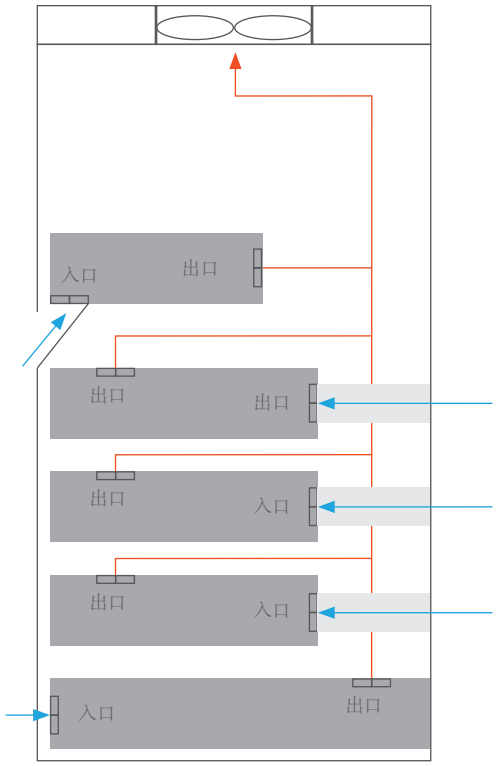


图8. 机架设计中自定义入口通风示例

在这些情况下，理想的方法是提供从机架外部到仪器入口的隔离路径。 这可确保正确理解和控制仪器的环境温度。

记住，仅仅观察进气口温度是不够的。确保根据仪器的规格为每台仪器提供足够的间隙，以确保其周围有适当的绝缘和气流。忽略这些约束虽然较为简单，但常常会导致大量空间未使用或浪费。因此，为了获得仪器规定的性能，必须遵循这些间隙规格。

从图片来看，仪器之间正在形成局部热气流。 记住，红色的热箭头只是为了表示该热量将被阻止进入进气口。 为设备进气口设计隔离路径可使得空气在机架周围和上方流动。 大多数机架组件为所有仪器的侧面提供足够的间距，以确保可以形成适当的“烟囱效应”。 仪器释放的热量也应该允许在进气口隔离屏障周围流动。 有许多方法可确保热量从机架中正确排出而不会影响仪器。这些只是其中几个例子。

热传递和气流

如前所述，大多数仪器的出气口温度等于环境温度。自热和空气加热会导致偏差：

- 自热 – 由于来自其他组件的热气以及自热效应，电子设备上的任何部件都会升温到环境温度以上。 我们无法控制自热。
- 空气加热 – 巧妙的机架系统布局可以最大程度减少空气加热，正确设计的机架冷却系统或室内冷却系统可以解决环境加热的问题。 因此，在设计系统时我们可以控制此偏差。

在这种情况下，由于安装了不同的PXI卡并且位于机架的不同位置，两个机箱的自热和空气加热稍微有所不同。

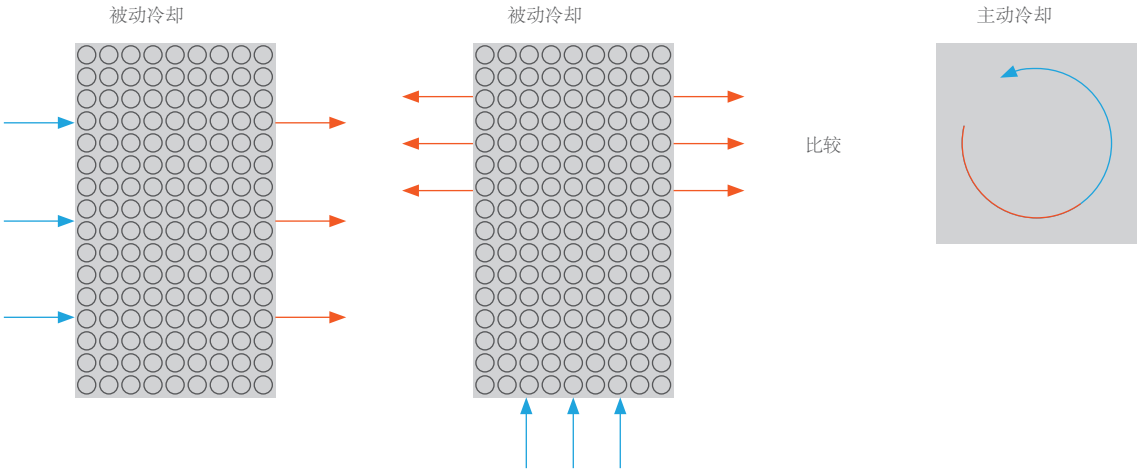


图9. 被动冷却依赖于内部安装仪器的风扇，而主动冷却则使用辅助风扇和安装在机架中的鼓风机。

被动冷却

被动冷却柜的目的是最大限度地发挥内部安装设备通过其风扇冷却自身的能力。 在这种方法中，设备会产生气流，并且机架中的表面和通风口交换热量。.

主动冷却

被动冷却只需依靠设备风扇和热传递，而主动冷却柜需要使用额外的风扇和/或鼓风机进行战略性部署，以补充空气，增加散热。

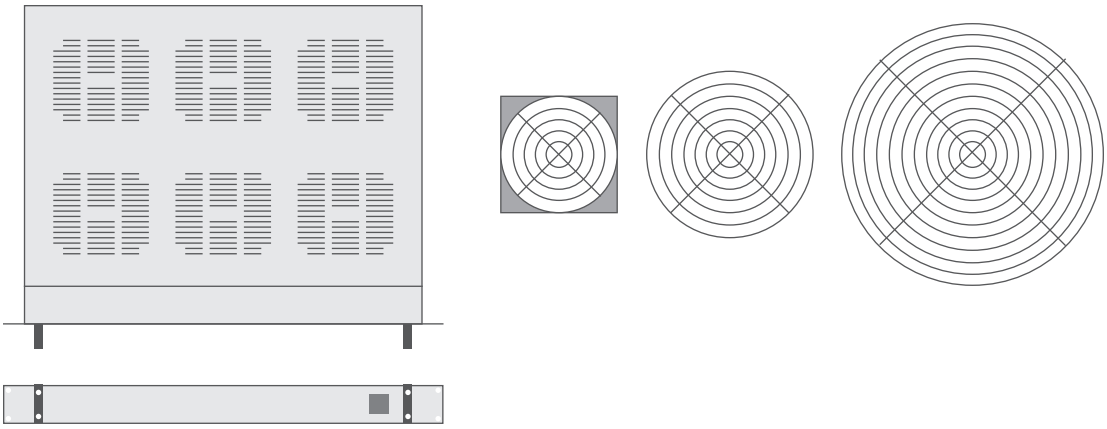


图10. 主动冷却选项的范围从单个托盘涵盖到侧边或顶部安装的每个风扇。

通常情况下，机架式测试系统需要强制风冷。通常在机架顶部安装一个尺寸合适的排风扇；然而，通风口和气流需要根据排气扇位置来计划。 如果空气路径发生弯曲、遇到障碍物、或者在某个区域集中了多个大功率仪器，则建议在机架中间放置风扇盘以帮助通风。 如果机架具有需要冷却的热点，则还可以考虑局部散热。 您可以使用单个风扇或风扇盘来实现此目的。

风扇容量

为了更好地了解合适的风扇大小，我们需要计算风扇每分钟能够移动的空气量，以立方英尺/分钟(CFM)为单位，同时需要考虑机架中所有设备的总功率瓦数和机架内外部空气之间的温差。

获得高CFM通常需要一些代价，如成本、振动和声学噪声。 此外，无论风扇的CFM多高，机架的冷却温度不能低于房间的室温。 因此，这个方程的目的是在所有这些参数之间实现良好的平衡。 空气阻力也是风扇冷却能力的一部分。 空气阻力会随着气流路径中物体的横截面积的增大而增加，无论其是进气区域的开口面积还是气流路径中的装置面积。 因此，风扇额定CFM应给予一定的余量，以确保其能够克服空气阻力并仍然提供必要的气流运动。

在评估总功率瓦数时，请不要使用设备的额定功率; 这通常是设备可以耗散的最大可能功率，但是设备很少情况下需要使用这么大的功率。 建议使用50%到60%的额定功率。 或者更好的做法是，使用机架系统的PDU来计算出实际功率，并使用该值。

Delta T (ΔTc)等于需要从机架中带走的热量。 这需要查看设备要求或模拟结果。 该数据根据机架内的不同位置可能会有所不同 - 通常在仪器出口或机架顶部较高，因此请务必了解适合您系统的ΔT，以及如何确认是否实现了该 ΔT 。

建模和验证

如果系统的成本高，包括组件的交货周期长，而且是关键或战略应用的一部分，或系统包含许多未知参数，则建模应该是设计过程的一部分。

机架设计建模

除了理论计算，机架系统的建模和仿真可以大大加快设计优化，并能够提供有效的反馈来帮助您了解哪里需要改进。

在软件中输入尽可能多的设计规范以获得最准确的建模。 如果您找不到必要的规范，请评估类似的组件/仪器，并询问团队的资深组员是否有使用这些设备的经验，以便获得估算值。 未知事项越少，建模就越准确。

在对设计进行了详细的研究后，无论通过评估仪器的温度要求、通过计算来优化气流和温度，还是进行设计仿真，您已经相当于执行了真正的合格性测试。 此时，您可以根据设计进行制造并验证设计的性能。

可以使用温度传感器或热成像相机来分析性能特性。关注机架内重要的关键区域，例如机架的进气口和仪器的进气口。收集机架设计中的温度数据，同时为机架供电并以常规的方式运行仪器，因为它们通常用于要测试的产品。 这可让你最真实地了解温度的行为。

有时可能还需要运行某些最坏情况的负载条件，例如热负载处于最高或最低时，以确保设计可以适应这些条件。 考虑测试时间点、测试持续时间和测试条件（有多少操作人员，可能与测试站有哪些正常交互等）作为可能影响结果的因素。 仔细分析结果，找出任何以前未识别的异常或您可能需要解决的问题。

验证方法

首先，在设计初期使用标准图形、方程式和仿真，以熟悉系统。 第二，使用温度传感器或热成像来执行系统特性分析，以验证设计并进行设计迭代，直到满足要求。 最后，执行Gage R&R研究来验证稳定性和性能，以确保站到站的性能与生产测试和验证、确认阶段的性能一致。

基于系统监测的可维护性

健康和监测系统使您能够实时评估系统，以确保系统仍能满足预期的性能。通过获得对系统性能的反馈，您至少可以在测试期间做出有据可依的决策，同时也可以更好地了解测量数据和更好地预测系统的维护活动。

重点领域包括：

- 用于监测系统性能的独立式系统
- 用于报告维护问题的系统看门狗
- 用于验证测试条件的测试反馈
- 用于评估和趋势分析的历史记录

将设计标准应用于产品

制定规格/限值时考虑热容量

使用机架设计采集数据时，在考虑对制造测试限制值进行规格验证时，不要忽视热量的影响。 如果预期结果和实际结果之间存在差异，热量可能是这些差异的原因。

规格验证

在推导规格参数时，必须清楚假设条件。 如果采集实际数据时的环境条件与推导的假设条件不同，请务必记得考虑进去。 确保在所提供的温度条件下获得预期的性能。

限制值计算

与规格验证类似，限制值推到应考虑温度差异和温度变化。 如果产品规格规定了某个温度范围，而您在不同温度范围的环境中进行测试，则应适当考虑设备的温度系数来补偿差异。 例如，NI开关模块的工作温度通常规定为0至55°C，但是测试这些模块的一般温度环境是标准制造测试车间，温度通常维持在24 ℃ ±4 ℃。 在确定测试限值时，应从规格和测量不确定度中减去最坏情况下的等效温度系数。

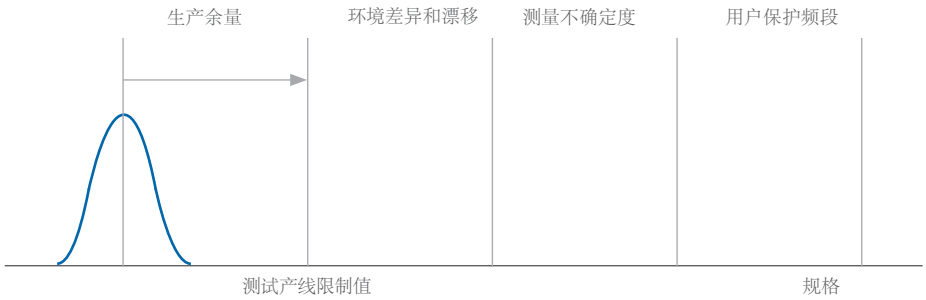


图11. 通用规格模型示例

独立系统监测

实时监测温度

实时监测温度使您能够在测试期间做出动态决策。 比如确定违反了某个操作要求后立即停止测试； 或者确定需要先执行自校准或施加延迟，然后再继续执行稳定性测试。 最后，历史数据可以在出现性能或测量结果相关问题时提供所需的信息。

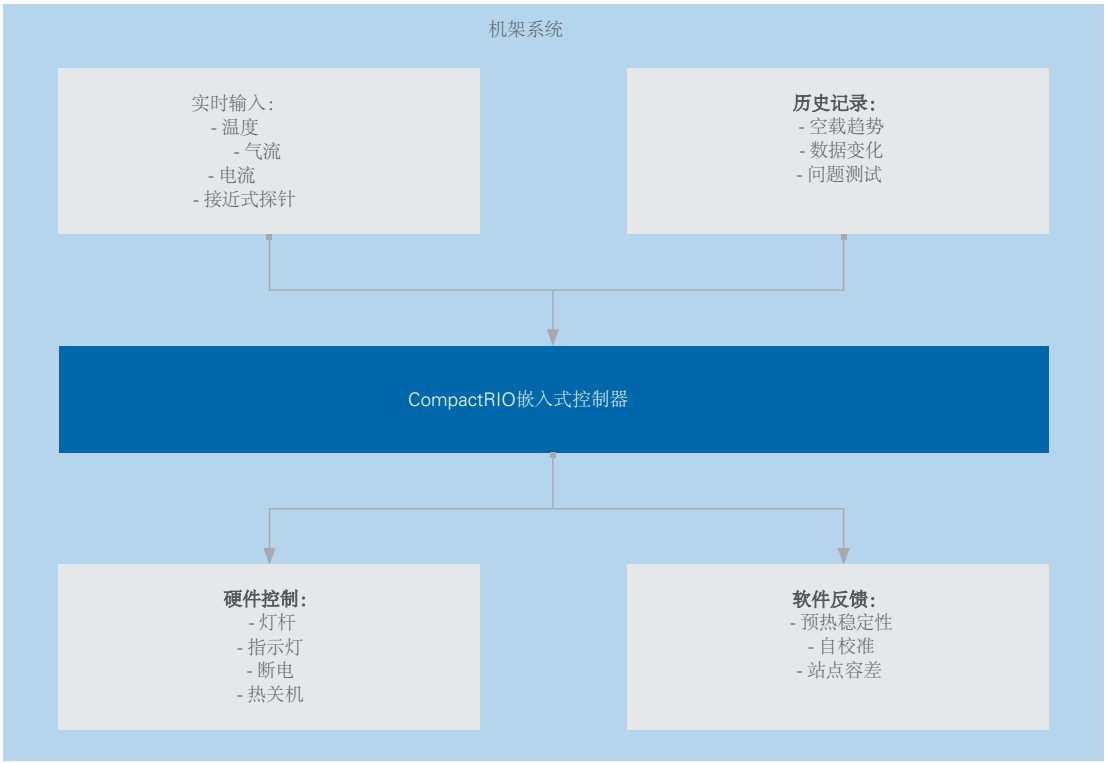


图12. 基于CompactRIO的独立监测系统示例

例如，您可以使用嵌入式独立系统来监视和控制测试站设计的某些方面。 您可以监测温度，并在测试执行中根据温度数据做出决策，也可监测和管理测试站资源，以支持并行测试。 一般来说，您可以对多个任务使用独立监测系统，不仅有助于机架系统的设计和验证，而且也有利于部署和长期使用。

虽然存在许多选项，您可以使用此类方法来：

- 监测
  - 整个机架的环境温度
  - 仪器的气流、电流消耗和内部温度
  - 仪器健康以便进行维护
  - 机架门，使用接近式传感器检测系统是否被打开
  - 温度状态，实行热停机机制以保护系统
- 获得数据，为测试应用做出实时决策
- 记录历史数据以供将来分析
- 通过灯杆、指示灯或显示器向测试站用户提供测试站状态反馈，并报告任何超出容差的情况

健康和监测系统可以帮助您实时评估系统，在测试期间做出有据可依的决策，以及更好地了解测量数据和预测系统的维护活动。

下一步

NI 联盟合作伙伴网络

NI联盟伙伴网络项目囊括了全球950多家独立的第三方公司，旨在为工程师提供基于图形化系统设计的完整解决方案和高品质产品。从产品和系统到集成、咨询和培训服务，NI联盟伙伴都通过独一无二的产品和技术来帮助用户应对当前一些最为严峻的工程挑战。

查找联盟伙伴

NI PXI机箱冷却

NI机箱经设计和验证，可满足或超过大多数高功率PXI模块的冷却要求。 NI设计的机箱远远高于PXI和PXI Express要求，分别为PXI和PXI Express机箱的每个外设插槽提供30 W和38.25 W的电源和冷却。 这种高标准的功率和冷却使得高性能模块（例如数字化仪、高速数字I/O和RF模块）的高级功能能够满足需要连续采集或高速测试的应用的需求。

了解更多关于NI PXI机箱设计优势

立即构建专属于您的PXI测试系统

NI是PXI的创造者和领先供应商，这一模块化仪器标准已经拥有70多家供应商，提供超过1500种产品。为您的应用选择合适的机箱、控制器和模块，并使用在线配置指南为您推荐系统所需的组件和附件。

配置您的PXI系统

测试系统构建基础知识

大规模互连系统和连接件

目录

引言

搭建一个测试系统，但却没有考虑如何将仪器连接到被测设备(DUT)，这就类似于开着没有车轮的汽车。 你的汽车可能有一流的马力和意大利皮革座椅，但是没有车轮，你就无法到达目的地。 大规模互连和测试连接件就是与自动化测试系统道路连接的车轮。 在确定所需的仪器、开关数量以及开关在测试系统的安装位置后，下一步是选择一个合适的大规模互连系统，并设计适当的连接件，将DUT无缝地连接到系统的其余部分。

大规模互连系统概述

大规模互连系统是专为简化来自或送往DUT或DUT连接件的大量信号的连接而设计的机械结构。 大规模互连系统不是逐一连接每个信号，而是可以同时连接和断开所有信号。 对于自动化测试系统，大规模互连系统通常需要一些可互换的机械外壳，所有信号通过该机械外壳从仪器（通常位于测试机架中）连接到DUT，从而帮助用户轻松地快速更换DUT，或是保护仪器前端的电缆连接，避免重复连接和电路断路。

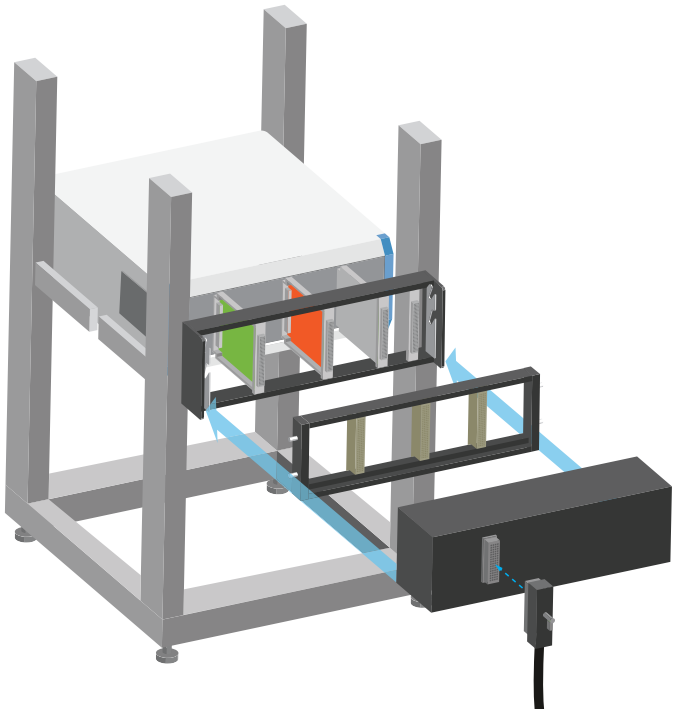


图1.大规模互连系统可同时连接测试系统和DUT之间的大量信号，提供了一种简单的方法，可以使用多个测试连接件在不同的DUT之间复用常用测试设备。

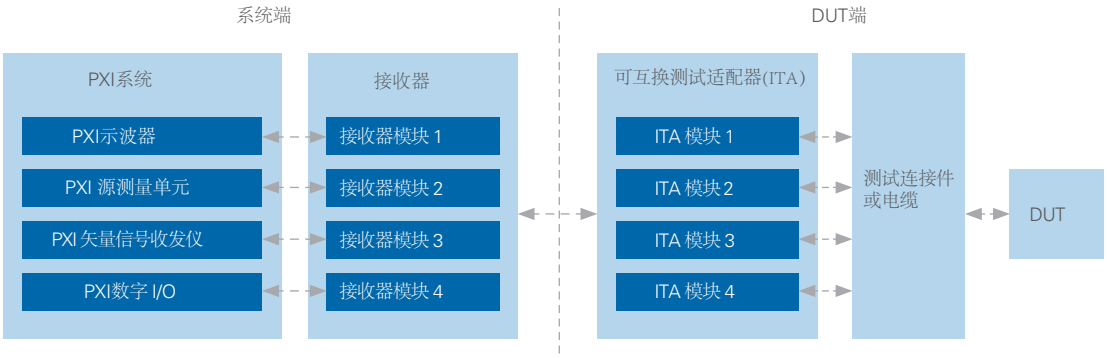


图2.大规模互连系统可以简化为两个部分。系统端组件（通常称为接收器）用于将仪器连接到大规模互连系统，用作为ITA的“插座”。或者，DUT端组件（通常称为ITA）用于将DUT连接到大规模互连系统，用作为接收器的“插头”。接收器和ITA在单个机械操作中相互配合，提供了一种简单的方式在不同DUT之间复用常用的测试硬件。

系统端组件

系统端组件包含仪器和大规模互连系统之间的一切。即使您更改可互换测试适配器(ITA)和连接件来连接各种DUT，系统端组件仍是测试系统中一部分常用的组件。下面是每个系统端组件的简要说明。

接收器

接收器是大规模互连系统中测试系统端的核心组件。它提供一个机制来将多个仪器同时连接到DUT。接收器系统包括框架、安装硬件、接收器模块以及从接收器模块到仪器的连接。

安装硬件

安装硬件将接收器固定在机架或PXI机箱的前部。这些硬件通常安装在19英寸机架的前侧，以方便测试人员进行操作。有些安装硬件带有铰链，以便于接触接收器后面的仪器或电缆。

接收器模块

接收器模块安装在接收器中，使得从仪器到接收器的主连接器（一类标准连接器，可连接到大规模互连系统的DUT端）以及所有其他接收器模块的连接通过一次机械操作即可完成。连接从仪器和其他辅助设备路由到接收器模块内的相应触点。这些连接通常根据所通过的信号的密度、带宽、电流或其它特殊要求来确定。

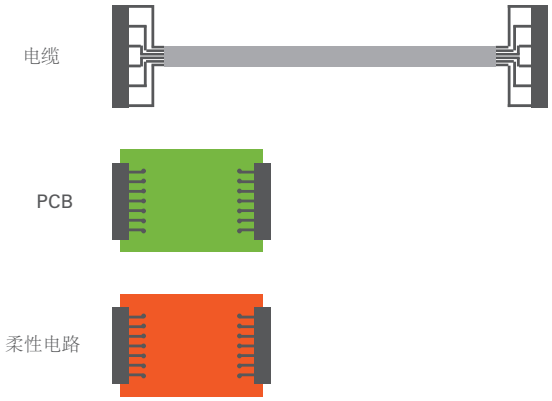


图3.电缆组件（最上面）提供了灵活的安装和接收器模块位置，但通常具有较长的信号路径。 PCB（中间）和柔性电路（最下面）可最小化信号长度，保持信号质量，但灵活性次于电缆组件。

为了将仪器和其他辅助设备连接到接收器模块，可以单独使用或组合使用两种方法：

- **电缆总成** - 使用电缆组件，通过标准或定制电缆从仪器直接连接到接收器中的触点。 电缆组件提供了灵活的安装和接收器模块放置，但是接收器模块和仪器之间的信号路径（24英寸或更长） 通常较长，这可能会影响性能。

- **接口适配器** - 接口适配器通常用于将仪器连接器（例如DIN、D-SUB、SCSI等）的所有I/O连接到接收器模块。 适配器始终与仪器直接匹配，并使用印刷电路板(PCB)、柔性电路或电缆为特定仪器提供最有效的连接方法。 接口适配器的优势是接收器模块和仪器之间的信号长度（通常为6英寸）较短，且接收器模块和仪器之间的信号性能可变，但这种方法灵活性较差，因此需要更周密的前期规划，安装位置必须精确，不能灵活更改。

低○ 平均◒ 高●

	电缆	采用电缆的大规模互连	采用PCB或柔性电路的大规模互连
信号质量	◒	◒	●
避免串扰	◒	◒	●
性能连续性（系统到系统）	◒	◒	●
DUT之间频繁互换	○	●	●
适用于设计和特性分析	●	○	○
适用于验证和确认(V&V)	◒	◒	◒
适用于生产测试	○	●	●
系统易于维护和升级	○	○	●
系统可重配置性（即可扩展性）	○	◒	●
易于复制（例如全球部署）	○	◒	●

表1. 电缆适用于器件的设计和特性分析，而大规模互连则适用于生产测试环境。

DUT端组件

DUT端组件包括大规模互连系统和连接件或DUT之间的一切。 DUT端组件组装在一个装置中，通常称为ITA，通过一组公共的仪器可以轻松地互换，以测试不同的DUT。下面简要说明DUT端的每个组件。

ITA

可互换测试适配器(ITA)是大规模互连的DUT端核心组件，包含机柜或机械框架以及与接收器配对的ITA模块和触点，并将系统输入和输出传输到DUT 。 如果接收器是插座，那么ITA就是插头。 许多测试系统可在不更换测试系统和接收器的情况下，通过更换不同的ITA来测试各种DUT。

ITA模块

ITA模块安装在ITA内部，其方式与接收器模块安装在接收器内部的方式大致相同。 它们为主互连系统提供了经由接收器传输的各种信号，并通过电缆、PCB或ITA外壳内的其他连接来实现这些连接。 我们需要选择正确的ITA模块和触点来匹配之前指定的接收器模块和触点。 然后将正确的信号从机箱内路由到连接件或DUT连接器。

外壳

外壳是容纳ITA和相应ITA电缆/模块的机械外壳。 常见的做法是将ITA外壳和测试连接件集成到单个框架或物理平台上，并在其上放置DUT，比如在消费电子或半导体测试中；或者在ITA和DUT之间连接一条电缆。 虽然我们可以选择标准机箱，但实际上几乎每个外壳都进行了一定程度的定制，以满足DUT的要求。

测试连接件

每个DUT都不相同， 因而需要独特的连接方法来实现最高效的测试。 例如，一些DUT更适合在ITA和DUT之间连接单条电缆，而其他DUT则更适合采用集成式测试连接件（例如钉子夹具床），从而提供了无需电缆的直接连接方法。

如何选择大规模互连系统

选择最佳的大规模互连方法可确保测试系统能够发挥其全部潜力，充分利用仪器的所有功能。然而，性能只是决策时需要考虑的一个方面，应该与成本结合考虑。为了找到满足高性价比解决方案，应该将大规模互连系统的资本支出与标准电缆的总成本（包括设计、验证、定制件的生命周期管理、维护成本和文档描述）进行比较。完整大规模互连配置(包含高速模拟、高速数字、开关和射频信号）的平均价格在$ 10,000至$ 15,000范围内。 但是，估算应用成本时最好应要结合大规模互连系统供应商提供的报价进行考虑。

以下常规步骤可以帮助您确定哪些大规模互连选项适用于您的系统，但是您也应该咨询大规模互连专家和NI联盟合作伙伴（例如MAC Panel或Virginia Panel Corporation (VPC)），他们可以指导您做出正确的决定。 除了提供建议外，一些大规模互连供应商还提供现成的大规模互连套件，包含接收器模块和用于常用仪器的匹配ITA模块，从而减少了配置大规模互连系统所需的时间和精力。 您只需提供系统所使用仪器的列表，他们可以提供相应的套件，包括接收器模块和匹配的ITA模块。

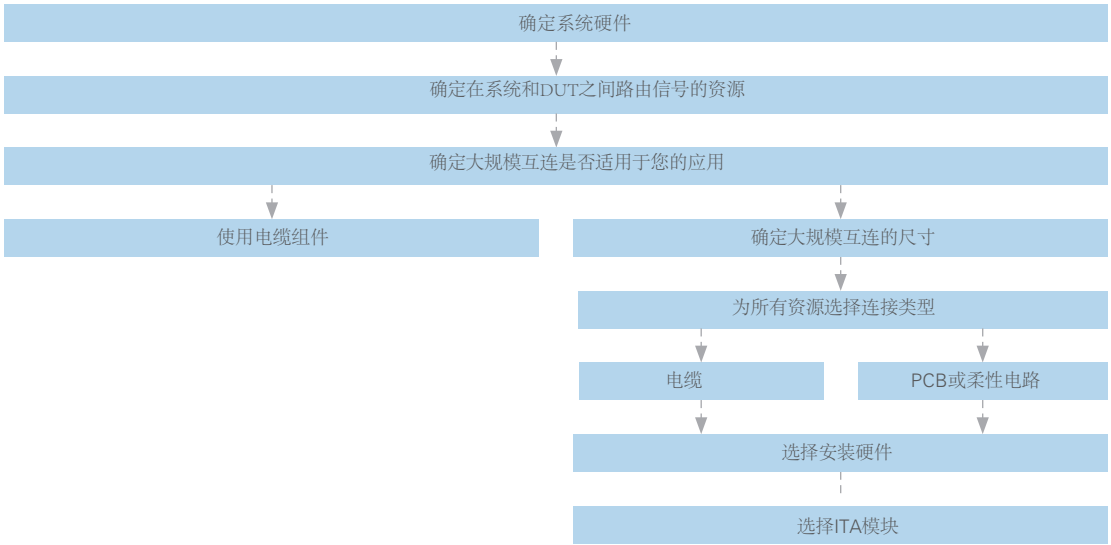


图4. 大规模互连系统的考量因素

1. 确定系统硬件

首先，确定测试DUT所需的系统硬件。 这包括仪器、开关、路由到DUT所需的任何辅助硬件（例如电源）以及测试机架的尺寸和样式。

2.确定测试系统路由到DUT所需的资源。

根据测试系统和被测设备的复杂程度，您很可能会需要通过大规模互连系统来路由所有必需的仪器和辅助组件连接。 有些测试组件不需要通过大规模互连系统进行路由，例如PXI嵌入式控制器或RAID存储系统。

在选择通过大规模互连系统进行路由的资源时，必须考虑测试需求可能的未来变化，以便构建一个灵活的测试系统来满足未来的需求。 例如，如果您有额外的资源，比如额外的仪器或电源通道，可能当前DUT并不需要，但未来的DUT可能会需要，那么您就可以基于这些未来变化预期规划好这些资源的路由，从而节省时间和金钱。

3.确定应用最佳接口类型 - 电缆或大规模互连。

此选择取决于多个因素，包括系统的复杂性、技术性能要求、灵活性和总体拥有成本。决策时请使用表1作为指导，同时也需要咨询大规模互连专家，以确保您为测试系统做出正确的选择。 假设您决定使用大规模互连，以下步骤可以帮助您确定接收器和ITA的尺寸以及选择正确的接收器模块、ITA模块和安装硬件。

4.确定接收器和ITA的尺寸。

接收器有许多不同的尺寸和样式可供选择，具体取决于前面几个问题的答案。 接收器最好保留备用插槽，以方便将来进行扩展。 类似于PXI系统仪器的规划，一般的指导原则是初期设计时至少保留20%的插槽未被占用。 请注意，ITA的尺寸将始终与接收器的尺寸匹配。

5.选择接收器模块和连接方法（电缆或接口适配器），以满足所有必需资源的路由。

所选择的接收器模块和连接方法取决于测试目标、所选仪器和任何其他辅助要求。同样地，您可以使用表1作为指导，但请务必咨询大规模互连专家，以确保您为测试系统做出正确的选择。您可以选择各种各样的模块和触点形式来满足所有信号类型的要求，包括：

- 低频交流信号
- 电源
- 射频信号
- 微波信号
- 热电偶
- 光纤

6.选择安装硬件。

在选择接收器模块和连接方法后，下一步是选择安装硬件，这取决于几个因素。大多数使用大规模互连的系统都会安装在19英寸的机架组件。如果采用电缆系统，建议将接收器安装在铰接框架上，以便于接触仪器和机箱。如果使用接口适配器，则建议使用标准安装法兰将接收器安装到PXI机箱中，然后将接收器和机箱安装到机架内的滑架上。

7.选择匹配的ITA模块和触点。

所配置的ITA模块和触点应匹配步骤五的选择。ITA模块中的触点可能不需要完全填充也可完全匹配接收器模块中的触点。在大多数情况下，来自特定仪器的所有资源都会传递到接收器模块。然而，ITA端并不需要所有这些资源才能测试特定的DUT;因此，并非所有ITA模块都需要完全填充触点。

测试连接件概述

测试连接件是在测试系统和DUT之间提供可重复连接的器件，通常采用定制设计来满足特定DUT的需要。在连接件背面使用大规模互连时，您可以使用一个通用仪器机架来互换各种测试连接件；还可以通过轻松地互换不同DUT的测试连接件来重复利用测试设备，这是通用测试仪或高混合测试仪的理想选择。

在设计测试连接件时，请提前了解您要执行的测试类型，因为器件设计、DUT特性、验证和确认(V＆V)以及生产测试的需求是完全不同的，而且需要的测试连接件功能也各不相同。

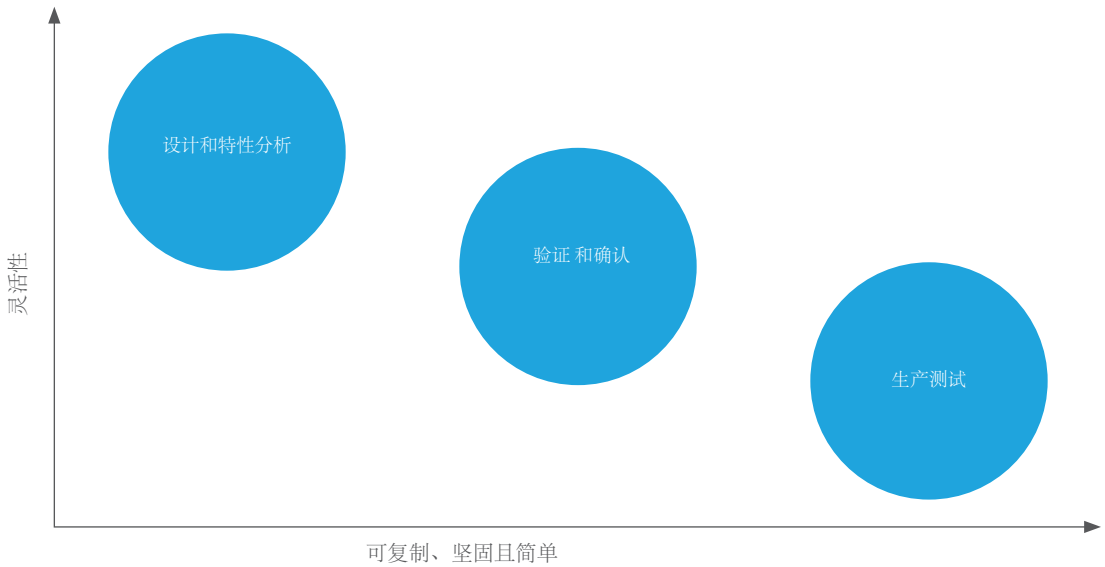


图5.生产测试系统需要坚固、易用且易于复制的测试连接件，而仪器系统设计和特性分析需要电缆、探头和Kelvin夹具的灵活性。.

设计和特性分析

在产品设计阶段，我们必须能够灵活地连接或断开DUT到测量仪器之间的电缆或引脚，以便对设计中的任何和所有元件进行测试或故障排除。这时就需要一种高度灵活的方式来与仪器连接，而不是采用大规模互连和连接件的简单或坚固型测试系统。许多设计工程师采用具有电缆、探头和Kelvin夹具的仪器，这样可轻松地更改连接。除了故障排除，设计工程师通常需要对器件的实际行为进行特性分析或记述，以便确定器件的规格参数，为开发V&V测试台或生产测试系统的测试工程师提供指导。

验证和确认

V&V是通过对具有统计代表性的一组产品进行测试来检查特定产品是否满足设计规格的过程。

验证

验证是检查器件是否满足相应法规和规范的客观过程。验证通常在设计和开发阶段以及开发阶段完成后执行。在开发期间，验证测试有时需要模拟系统其余部分的行为或进行建模，以预测或预了解器件开发完成后的行为。开发完成后，验证测试可以采用回归测试，重复进项多项测试设计，以确保器件随着时间的推移持续满足设计要求。

确认

确认是一个比较主观的过程，涉及主观评估器件是否满足最终用户的操作需求，这通过为新产品创建需求问题描述来实现。确认测试的要求来自于用户需求、规格和/或行业法规。在确认规范时，我们的目标是确保规范能够涵盖用户的需求，而不是确保给定器件满足其规范要求。

在V&V过程中采集的测试数据也可用于确定生产测试系统的测试限制。虽然只有一小部分给定的产品进行了V&V，但几乎所有最终产品都需要通过生产测试。因此，V&V过程设计的连接件通常有助于减少DUT和仪器之间的连接，使得这些连接件相比用于生产测试的连接件较不坚固。

生产测试

生产或功能测试通常在制造过程结束时进行，目的是测试产品是否满足公布的规格和质量标准。 许多时候，功能测试是自动化进行的，以便提高生产力以及减少人为交互导致的错误。有时，生产测试包括仿真或模拟产品的实际工作环境。 最重要的是，功能测试使用的是客户最终使用的连接器，而不是PCB上的各种测试点。

由于每个设备的制造和运输都经过生产测试，连接件的设计必须坚固耐用， 最大程度延长正常运行时间，且必须易于使用且符合人体工程学。同时应该尽量减少测试操作员所需的交互。将DUT与连接件或仪器到连接件的连接电缆对准所花费的额外时间会降低工作效率并增加测试成本，并且增加由于人机交互而导致错误的风险。最后，生产测试系统（包括测试连接件）应易于复用于其他部署。

连接件考虑事项

当设计测试连接件时，确保连接件使用正确的接线类型和技术，尽可能使用PCB而不是电缆，并尽可能多地自动连接。 此外，我们还需要为连接件制定预防性维护计划，以确保长期的成功部署。

正确接线

电线通常是噪声和误差的来源，因此我们应该为测试连接件选择最佳类型的电线。 为了确保信号完整性，一些电线提供了绝缘、屏蔽、保护或双绞线等特性。 一些仪器手册建议使用特定电缆，但这通常取决于正在执行的测量类型。 例如，双绞线是执行差分测量时抑制噪声的理想选择。 使用屏蔽电缆也是抑制噪声的一种技术，但重要的是根据信号源和输入配置的接地使用正确的接地方案。最后，保护通常用于消除数字万用表(DMM)或源测量单元(SMU)的HI和IO端子之间漏电流和寄生电容的影响。

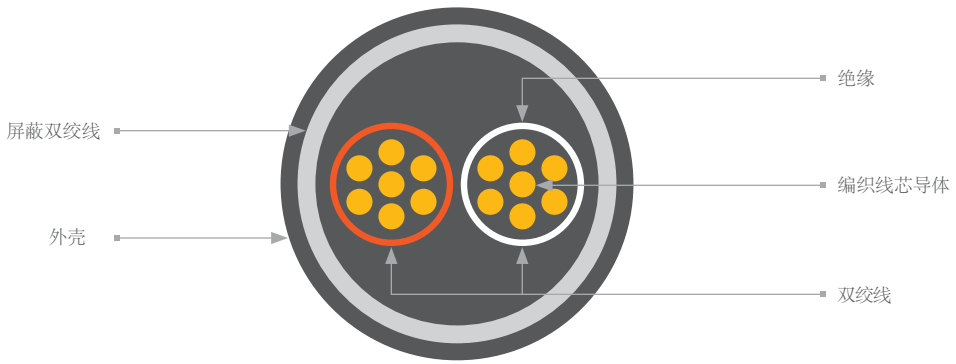


图6.为了保持信号完整性，不同的电缆提供屏蔽、绝缘、双绞线或保护等不同功能。

最大程度减少操作员交互

大规模互连系统的目标是提高测试设备复用性，以及减少可能导致错误、降低产量和增加测试总成本的操作员交互。 除了使用大规模互连系统来减少用户交互之外，良好的测试连接件应最大化测试操作者通过单次交互实现的测试连接件和DUT之间的连接数量。 例如，一些测试连接件利用由单个操作手柄驱动的连接方法或者利用电动或气动马达自动地完成多个连接。 生产测试时，通常使用客户最终用于连接设备的连接器来进行连接。

构建可复制且易于扩展的连接件

某些接线技术可以保持信号完整性，但建议在测试连接件中使用PCB来提高信号完整性，以及减少测试操作员或技术人员在安装阶段的接线工作。 使用可同时进行多个连接的测试连接件有助于提高生产能力、测试重复性和用户人体工程学，但是同时必须减少测试连接件内部的布线量并且在可能的情况下用PCB替换电线，因为这可以进一步提高信号完整性，减少安装和连线相同测试系统所需的时间。 需要权衡的是，设计定制PCB需要更多的前期成本和工作量，但这些付出将在第一个系统以及随后复制的测试系统部署后得到回报。

为测试连接件制定预防性维护计划

为了确保测试系统能够长期提供支持，总测试系统维护计划中应包括测试连接件维护计划。 这应包括整个测试系统生命周期中进行定期检查和/或更换连接器、电缆、弹簧针、继电器和其他组件。 在选择检查周期时，需要考虑特定部件的故障率。 判断故障率时，请参考供应商提供的平均故障间隔时间(MTBF)或公司特定的服务故障率。 尽管在比较或分析仪器时两者都是有用的，但是服务故障率可能更有用，因为它可指示仪器实际上如何用于特定应用。 如果不知道服务故障率数据，可通过MTBF数据导出理论故障率，从而计划符合成本和风险容限的检查周期。

下一步

NI PXI配置指南

PXI配置指南可帮助您比较仪器选项并配置基于PXI的测试系统，包括PXI机箱、控制器、模块、软件、服务和辅助项目。
使用PXI配置指南，配置专属于您的测试系统

MAC Panel

MaC Panel公司为希望在测试或测量环境中实现可靠、经济高效的电气连接的公司提供了解决方案源。除了全系列的大规模互连产品，包括为PXI设计的MaC Panel SCOUT大规模互连系统，MaC Panel公司还提供了定制布线服务、钣金加工和定制设计帮助以及一系列选项来支持全球的自动化测试设备。

了解更多关于MAC Panel

Virginia Panel Corporation

弗吉尼亚面板公司， 简称VPC，致力于为商业、军事、电信、航空航天、医疗、汽车和消费电子应用设计、制造和销售大规模互连产品。

了解更多关于VPC

NI联盟伙伴目录

NI联盟合作伙伴网络是一个由全球1,000多家独立第三方公司组成的项目，提供完整的产品以及集成、咨询和培训服务。 其中一些公司，如MAC Panel和VPC，提供定制布线解决方案、大规模互连系统和完整的连接件解决方案。

浏览NI联盟合作伙伴目录

测试系统构建实用指南

软件部署

目录

引言

尽管设备日益复杂化，测试工程师却通常需要在日益紧迫的时间期限和缩减的预算下创建更复杂、更高级的混合测试系统。创建这些测试系统的一个最重要步骤是将测试系统软件部署到目标机器。这也是最繁琐、最令人沮丧的一个步骤。如今，对于那些只是简单寻求最便宜和最快速解决方案的工程师，琳琅满目的部署方法更加让他们无从选择。此外，测试系统开发人员也面临着其系统特有的各种考虑因素和敏感事项。

在本指南中，我们将部署定义为编译一系列软件组件并为将这些组件从开发计算机导出到目标机器上执行的过程。测试工程师不直接在开发环境运行测试系统软件，反而采用部署方法，主要是因为成本、性能、可移植性和保护等原因。以下几个示例是测试工程师需要从执行开发环境转变为二进制部署的常见情况：

- 每个测试系统的应用软件开发许可证成本开始超出预算限制。每个系统使用部署许可证是一种更有吸引力且更加高效的解决方案。
- 由于内存限制或依赖关系问题，测试系统的源代码变得难以传输。
- 测试系统开发人员不希望终端用户能够编辑或接触系统的源代码。
- 通过开发环境运行时，测试系统的执行速度会变慢或内存消耗增加。编译执行代码可提高性能，减小占用的内存。

本指南建议并比较了不同的考量因素和工具，以解决测试系统部署过程中的困难和困惑。尽管本指南可以解决测试系统部署的多个不同主题，例如源代码控制最佳做法或安装程序创建，但所选的主题应涵盖大多数通用部署问题。每个部分的结尾都提供了一个基本用例和高级用例的最佳范例建议：

- 基本用例是一个由可执行程序组成的简单测试系统，可按顺序运行测试步骤，并调用几个硬件驱动程序。这种类型的系统包含的测试功能通常少于200个。
- 每个基本用例的最佳实践最后都会阐述可能需要考虑高级用例的情况。

高级用例是指大型生产测试系统，使用可执行文件、模块、驱动程序、Web服务或第三方应用程序的组合来执行高度混合的测试序列。这种类型的系统通常包含数百或甚至数千个测试功能。

管理和确定系统组件

组件定义

在软件开发中，组件是指系统中使用的任何物理信息，例如二进制可执行文件、数据库表、文档、库或驱动程序。完成成功部署的第一步是确定与测试系统相关的组件，并确保每个组件都有相应的部署方法。该步骤可以非常简单，也可非常复杂。例如，简单测试系统的组件可能是单个可执行程序和必要的硬件驱动程序。

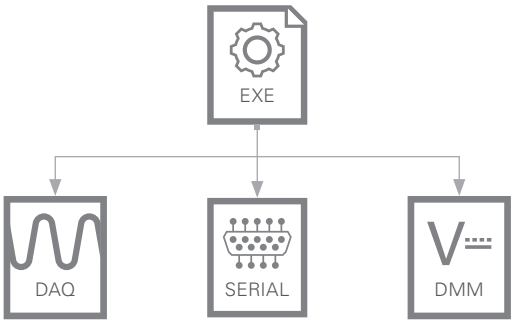


图1. 简单的测试系统可执行程序示例，包含DAQ、串行和DMM驱动程序

复杂的系统组件

然而，在复杂的测试系统中，这些组件通常是XML配置文件、数据库表、自述文本文件或Web服务。 较复杂的系统提供了更高级的部署选项。 例如，配置文件需要频繁地更新，以便针对季节性天气变化对采集的数据进行校准，而主要可执行程序很少需要更新。 每次配置文件需要更新时都不必重新部署可执行程序，因此配置文件可以使用与可执行程序不同的部署方法。

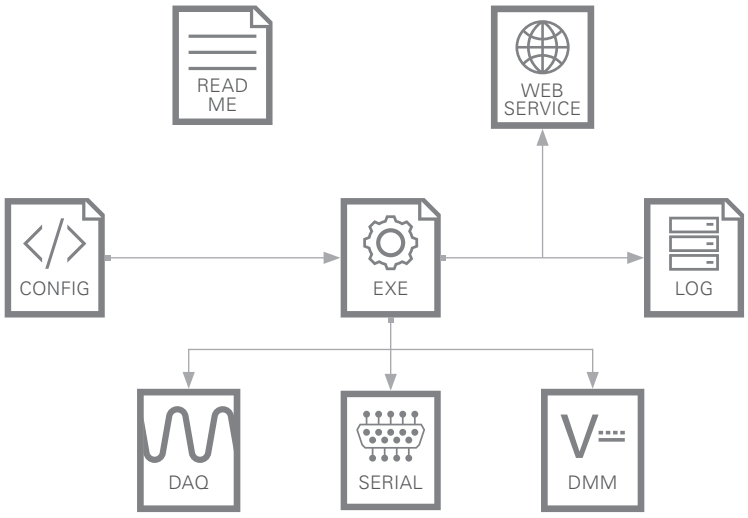


图2. 具有复杂依赖关系的测试系统示例

除了确定每个系统组件和设计其部署方法之外，确定系统组件之间的关系并确保部署方法不会中断这些关系是非常重要的。对于需要频繁更新的配置文件，工程师可能必须将配置文件安装到每个部署系统上的相同位置，以便可执行程序在运行时找到它。

依赖关系跟踪

维护依赖项之间的关系涉及安装一个依赖关系跟踪程序，以确保每个组件的依赖关系组件均已部署。 虽然在手动确定每个系统组件之后这一点看起来很明显，但是依赖关系通常可以深度嵌套，而且当系统扩展时需要能够自动识别。 例如，如果系统B中的可执行程序依赖于一个.dll文件来正确执行，制定部署计划的工程师可能忘记将.dll文件标识为必要组件或者不知道这个依赖关系。 在这些情况下，开发工具就派上用场了，可自动识别所编译的应用程序的大部分（即使不是全部）依赖关系。

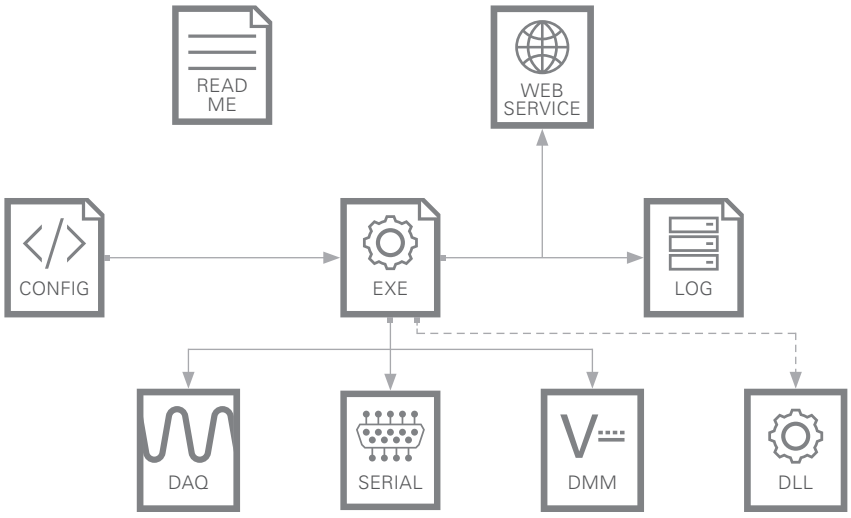


图 3. 复杂测试系统出现意外依赖关系

以下是几个开发软件应用程序示例：

- **LabVIEW Application Builder—**识别特定上层VI的依赖关系（子VI），并在编译的应用程序中纳入这些子VI
- **TestStand Deployment Utility (TSDU)—**以TestStand工作区文件或路径为输入，并识别系统的依赖关系代码模块; 自动编译这些模块并将模块纳入到编译的安装程序中
- **ClickOnce—**可帮助开发人员轻松地为其.NET应用程序创建安装程序、应用程序甚至Web服务的一项微软技术；可以配置为将安装程序纳入依赖关系，或者提示用户在部署后安装依赖关系
- **JarAnalyzer—**Java应用程序的依赖关系管理工具；可以遍历目录、解析该目录中的每个jar文件，并确定它们之间的依赖关系

关系管理

通常情况下，关系不仅存在于主测试程序可执行程序及其相关组件之间，而且存在于每个单独组件之间。 这使得不同组件或软件模块之间的关系的性质成为一个问题。 随着系统不断扩展，解析不同库、驱动程序或文件之间的依赖关系可能变得非常复杂。 例如，测试系统可以使用具有以下关系的三个不同代码库，如下图所示。

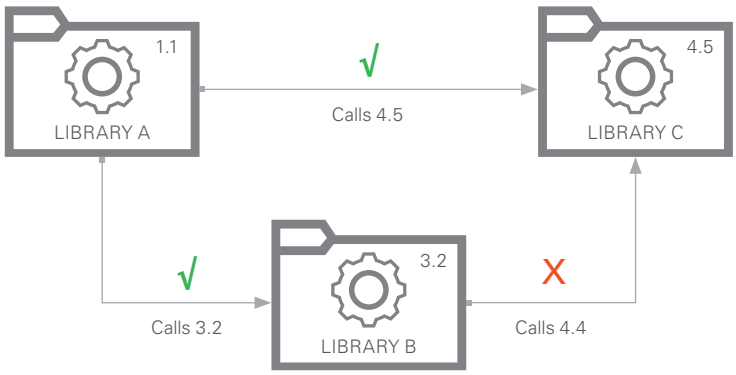


图4. 由于库A依赖于库C版本4.5，因而库B对库C版本4.4的依赖导致了一个不可解决的依赖性问题。

对于这些复杂的系统，通常需要使用依赖解析器来识别依赖冲突以及管理不可解决的问题。虽然我们可以自行编写依赖解析器，但工程师可以使用程序包管理系统来管理依赖关系。程序包管理器的一个例子就是NuGet，这是一个专为.NET框架程序包设计的免费开源程序包管理器。 另一个例子是用于LabVIEW软件的VI程序包管理器，可允许用户发布代码库，并通过API提供自定义代码库管理工具。

最佳实践

基本用例：对于基本或简单的系统，通常可以手动跟踪所有必需的组件。 使用软件应用程序或程序包管理器来管理依赖性并不是很必要，而且需要的前期成本过高。 然而，如果碰到警告信号，例如一直碰到依赖关系缺失问题或依赖关系不断增多，通常意味着需要更高级的依赖关系管理。

高级用例: 采用可扩展的依赖关系管理系统可让复杂的系统变得更易于维护和升级。 这可能意味着使用程序包管理器来诊断程序包之间或程序包与软件应用程序之间的关系，以了解和识别各种组件的依赖关系，维护这样的系统对于长期成功至关重要。

硬件检测

硬件断言

需要特定硬件设置的测试系统需要确定该硬件存在于系统上，并且当硬件不包含在部署计划中或不兼容时可执行应急计划。 虽然开发人员通常通过视觉检查测试机器并将硬件组件与原始开发系统匹配来手动完成硬件断言，但是更好的做法是假定测试系统是为第三方创建的。 测试系统的客户如何知道是否有不兼容的硬件？ 如果插槽或端口不正确，系统能否适应正确的模块？ 当硬件丢失时，系统是否可以解决或进行调整？ 在开发初期回答这些问题可以简化测试系统的扩展和发布。

硬件标准化

硬件断言的最终目标是确定预期系统和实际物理硬件系统之间完全一样。 为此，通常最有效的方法是先将每个测试系统上要用的硬件组件标准化：

- **文档描述** - 标准硬件中的组件列表应该可供每个新系统访问。 文档描述应该包含有关提供商、产品编号、订单号、组件数量、可更换组件、保修、支持政策、产品生命周期等信息。
- **可维护** - 硬件标准化最困难的问题之一是确保每个测试系统使用的硬件组件在将来仍然可用。通常旧硬件是指制造商标示为寿命终止(EOL)，并且需要刷新测试系统硬件的标准组件。从硬件升级和测试系统停机的角度来看，这种刷新通常非常昂贵。与硬件制造商共同探讨硬件组件的生命周期策略有助于减少未来面临的挑战。大多数硬件制造商，比如NI，提供生命周期咨询以及每个硬件组件生命周期的逐步终结。
- **可复制** – 应考虑是否需要全球性甚至区域性地部署硬件。确保有合适的硬件部署方法，以便在偏远位置快速搭建新系统。对于许多系统，维护备用硬件组件以便进行维护或紧急替换也是非常重要的。

开机自检(POST)

即使测试系统的硬件正确且正确连接，也必须对硬件进行简单测试，以确保系统在运行时其性能与预期一致。 幸运的是，大多数硬件组件都具有制造商配置的自检功能，以对设备的通道、端口和内部电路板进行简单检查。 在为每个测试系统供电时，应对所有连接的设备进行自检，以便在早期检查出故障硬件。 例如，每个NI设备都具有自检功能，可以通过设备的驱动程序API以编程方式进行调用。 为测试系统供电的第一步可以是调用每个设备的自检功能，并向操作员警告任何故障硬件。

别名配置

不幸的是，对硬件进行标准化不能完全确保相同的配置。 通常，需要使用硬件配置软件（例如Measurement & Automation Explorer (MAX)）来将硬件设备重新映射到别名。 例如，在安装所有硬件组件并为系统供电后，工程师可以使用MAX来检测系统上存在的NI硬件，并使用Windows设备管理器来查找非NI硬件。之后就可以通过编辑.ini配置文件来将硬件设备正确映射到别名。下图显示了此过程的可能输出。

别名	设备名称
PXI NI-4139	PXI 1 插槽1
PXI NI-3245	PXI 1 插槽2
PXI NI-2239	PXI 2 插槽1

表1.用于将物理硬件映射到测试系统别名的hw_config.ini文件

程序配置

LabVIEW软件中用于NI硬件的System Configuration API等库可以以编程方式生成所有可用实时硬件的列表并进行别名映射配置。 例如，测试系统可执行程序可以调用System Configuration API的查找硬件功能来生成可用NI硬件列表。 在该列表中，每个设备的别名属性可以通过硬件节点设置为预定义的名称。 这可能会导致系统出现问题，例如硬件设备映射到不正确的别名。 因此，工程师应将其与另一种保护措施（如手动确认映射列表或标准化硬件设置）结合使用。

最佳实践

基本做法：对于基本或简单的系统，重要的是确保预期的硬件存在于系统上。 将硬件标准化对于所有系统都是不错的做法，尤其是随着硬件系统数量的增加，硬件标准化尤为重要。 正确执行测试系统所需的机箱、模块和外设应进行文档记述并定期更新。 然而，验证正确的设备是否存在于系统中通常可以使用MAX之类的工具进行手动检查，而不是使用程序化或可重配置解决方案。 随着硬件系统随着模块和设备数量的增加而扩展，可能需要采用更高级的解决方案来防止硬件丢失问题。

高级做法： 对于复杂的系统，跟踪哪些硬件对于系统是必要的或者哪些硬件存在于系统中应该使用多种解决方案的组合。 与上述基本做法一样，不同系统的硬件应该进行标准化并进行文档记录。 如果要检测故障硬件，应通过开机自检(POST)来确保所连接的硬件能正常工作。 此外，当硬件标准化失败时，应当使用程序化或手动别名映射系统来自动将期望的设备重新映射到系统的别名。

依赖关系解析

依赖关系断言

我们通常需要制定一个计划来解决对部署系统的现有依赖关系和缺失依赖关系。 通常，部署的测试机器已经安装了测试系统镜像的部分依赖关系。 对于较小型的系统，我们可以简单地重新安装所有依赖关系来确保它们的存在。 然而，对于较大型的系统，我们需要先检查系统上是否存在这些依赖关系，这样有助于避免重新安装所有依赖关系。 这种做法称为依赖关系断言，有助于减少部署时间，但前提是要为不同的依赖关系制定计划。 “组件化 ”部分将进一步讨论如何通过组件化实现更快速的部署。

例如，测试系统可兼容14.0和15.0版本的NI-DAQmx驱动程序。 虽然测试系统可能要求安装NI-DAQmx 15.0，但它可能允许14.0版本作为依赖项。 尽管可以兼容15.0版本，但允许14.0版本而非15.0版本可能会改变测试系统的行为，比如跳过某些测试步骤或调用不同的函数。 所有这些更改都需要记录和测试。

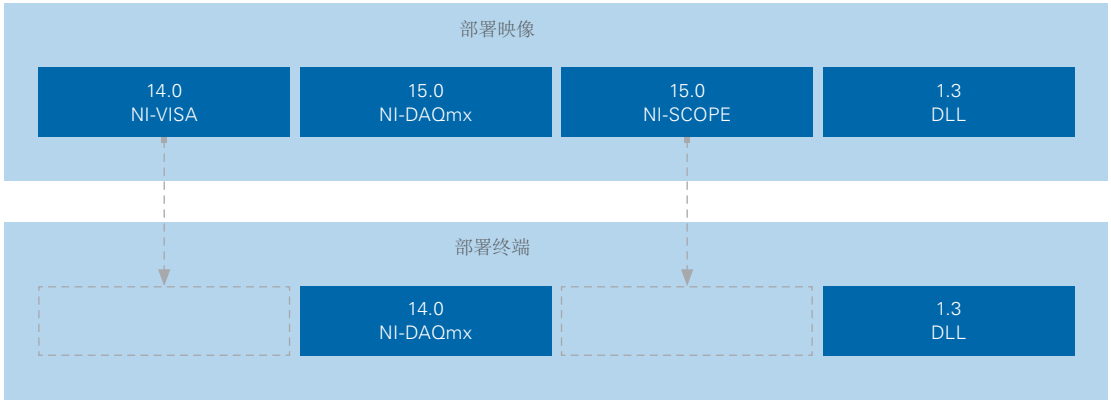


图5. 依赖关系断言

依赖关系断言的第二个要素是决定如何处理丢失的依赖关系。 如前所述，一个较好的做法是操作的时候想象测试系统部署到客户机器上的场景。 进行部署的工程师是否应该被通知缺少的依赖关系？ 缺少的依赖关系是否应该在后台悄无声息地安装还是应该由用户手动找到并安装？在初期回答这些问题有助于实现更快速的部署以及对缺失的依赖关系进行适当的处理。

最佳实践：

基本做法：对于基本测试系统，依赖关系解析和断言通常是不必要的。 安装所有测试系统的依赖关系（无论它们是否存在于系统中）通常比尝试识别并仅安装缺失的依赖关系更简单。随着测试系统的扩展，系统总安装时间可能会不断增加，增加到了一定程度，就需要开发依赖关系断言和解析工具。

高级做法：即使具有稳定的网络连接或采用压缩图像，部署时间也有可能会迅速增加到不合理的程度。 对于大多数高级测试系统，需要一定数量的依赖关系断言来避免重新安装所有组件。部署过程可以集成System Configuration API等用于查找系统中安装的NI软件的工具或用于生成Windows机器上所有程序的列表的wmic命令集。 这可以允许安装程序跳过特定组件或允许版本差异。

版本管理

通常，工程师需要知道当前部署到测试系统的软件镜像是哪个版本，或者能够提供发布部署历史。 如果这些是必要的需求，应该有一个版本管理系统来解决以下每个问题：

- 哪个版本目前部署到系统A？
- 系统B的最近部署状态是什么？
- 版本1、2和3部署到哪里？
- 系统A的发布历史是什么？

在大多数测试环境中，工程师使用铅笔和剪贴板系统来回答这些问题，但是，也有一些工具可以自动记录版本指标，并提供特定系统的版本历史记录。这些用于版本管理的工具可以集成到集成开发环境（IDE）中，或作为独立的版本管理工具存在。 其中一些例子包括：

- **Visual Studio Release Management—**Visual Studio IDE附带了用于自动化部署、版本历史跟踪和版本安全性管理的工具
- **Jenkins Release Plugin—**借助这个专用于Jenkins持续集成(CI)服务的插件，开发人员可以指定编译前和编译后操作来管理Jenkins集成开发的版本。
- **XL Deploy—**此应用程序版本自动化(ARA)软件可扩展到企业级，并提供可视化状态仪表板、安全性和分析功能来管理版本。

虽然上述几个工具可作为IDE和独立部署解决方案，但版本管理工具更经常与CI服务器和端到端部署过程结合使用。 这一点非常容易理解，因为我们更经常会问某个机器使用的是哪些代码，而不是哪个版本。 对于编译的系统镜像，这可能难以通过手动观察来确定。 跟踪从开发到部署的代码对于有效的版本管理是必要的。

端到端系统自动化

高效的版本管理是为测试系统部署开发复杂的端到端过程的必要组件。 从开发到部署，每个过程都依赖前一个版本；如果源代码管理良好，就能够很好地管理测试和版本。 有了良好的测试和版本处理，版本管理就只是原系统的简单扩展。 下图显示了一个典型的端到端系统。

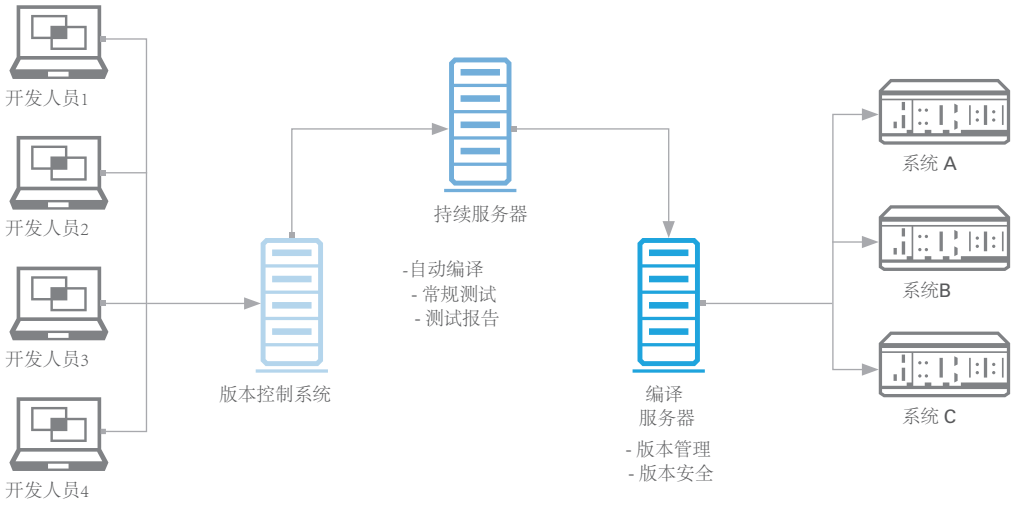


图6.开发人员将代码提交到版本控制存储库，然后可以CI服务器中进行编译和测试。 在服务器中，版本可以存储在编译服务器中并进行管理。

在此架构中，测试系统开发人员定期开发源代码并将其提交到版本控制存储库。 接着CI服务会将源代码拉入其自己的存储库，并相应地编译和测试代码。 此时，无论是自动还是手动执行，开发人员都可以将通过CI测试的版本移动并存储到编译服务器或存储库中。编译服务器通过报告和跟踪每个软件版本与特定测试机器的链接来进行版本管理。 通常，测试机器通过请求安装特定版本的测试系统来启动部署过程；然而，开发人员还可以配置编译服务器，以将镜像推送到所选择的机器上。

即使是在基本系统需要一定程度版本管理的情下，最实际的解决方案应反映版本需求的固有复杂性。 如果需求是跟踪哪个版本部署到系统上，则通过配置文件或对作为编译可执行文件的组件进行手动版本管理可能就足够了。 如果需求范围扩大，测试系统数量增加，或者应用程序版本数量增加，则必须对版本管理系统进行定义。

最佳实践:

高级做法: 通常，需要版本管理的复杂测试系统最适合采用一定形式的端到端自动化。 这可以通过CI服务（如Jenkins或Bamboo）来完成，这些服务会将版本管理与版本测试和源代码控制相结合。

版本测试

回归测试

在软件工程中，回归测试是指在对系统进行改动之后测试先前开发的系统的过程。 回归测试的目的是保持每个版本的完整性，并跟踪系统中特定更新或修补程序的错误。 对于组件化系统，回归测试对于确定升级到模块A是否会导致模块B中出现意外行为尤其重要。例如，升级系统中的NI-DAQmx硬件驱动程序可能会导致硬件抽象库在调用较旧NI-DAQmx版本但被较新版本弃用的函数时会出现问题。回归测试有两种类型：功能测试和单元测试。

功能测试

在测试系统中，最重要的问题是软件进行了哪些更新？这些更新是否会破坏系统的功能？系统是否仍然按照预期的方式运行？功能测试是通过一组已知输入验证系统是否会产生预期输出，可以帮助回答关于整个系统的这些问题。这种类型的测试通常采用“黑箱”方法；不对系统的内部机制进行分析，仅仅验证系统的输出是否与预期一致。对于测试系统，功能测试可能是验证硬件配置更新、驱动程序更改或测试步骤添加不会更改原始测试功能。工程师可以在模拟待测设备(DUT)的测试系统上执行功能测试，验证这些系统经校准后是否通过特定测试。例如，用于测试某个对象是否是圆形的系统由四个部分组成：相机控制器、圆周传感器、直径传感器和体积传感器。如果系统从版本1.0更新到1.1，并且对直径传感器进行了更改，则在下面的图中，正在测试的第二个圆形一开始会通过圆形测试器的测试，然后在系统更新后无法通过测试。

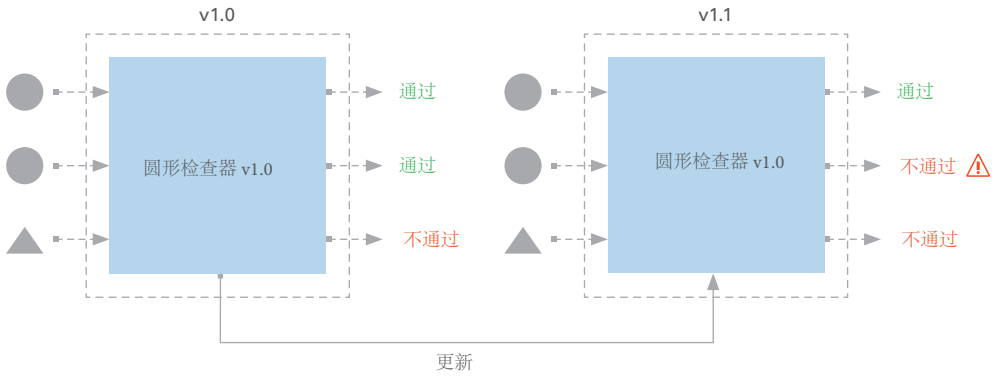


图7.对测试系统中的模块的小更新可能导致功能测试发生故障，这是一种伪故障。

单元测试

功能测试是针对整个系统，而单元测试则是针对特定的模块、组件或功能。这种类型的测试旨在跟踪测试系统特定部分的质量，而不仅仅是正确性。例如，测试结果记录到数据库后，可以在数据库控制器上进行单元测试以测量数据吞吐量。这样，不仅可以对数据库控制器的任何更改进行分析来对功能进行适当的记录，而且还可以回答软件更改是加快还是减慢系统的记录能力这一问题。除了帮助发现错误，单元测试可以将观察到的性能增强或降低与特定更改相关联。前面介绍的圆形测试器示例可以用来说明单元测试和功能测试之间的差异。假设圆形测试器的直径传感器软件部件进行了升级，则可以对直径传感器进行单元测试，而不需要对整个系统进行功能测试。对于单元测试，可以向特定组件提供表示具有特定直径的圆的二进制图像数据，并且测试输出是否与圆的已知直径匹配。通过这种方式，就可以验证和定量测量模块的正确性，例如测量模块的执行时间。

在这种特定情况下，升级显着降低了模块的速度。另外，由于升级后系统未通过功能测试但通过单元测试，我们还可以推断，软件漏洞很可能存在于相机控制器和直径传感器之间的通信中。 这种验证系统正确性和单个模块功能的能力可以确保只有合格的版本才会部署到测试机器上。

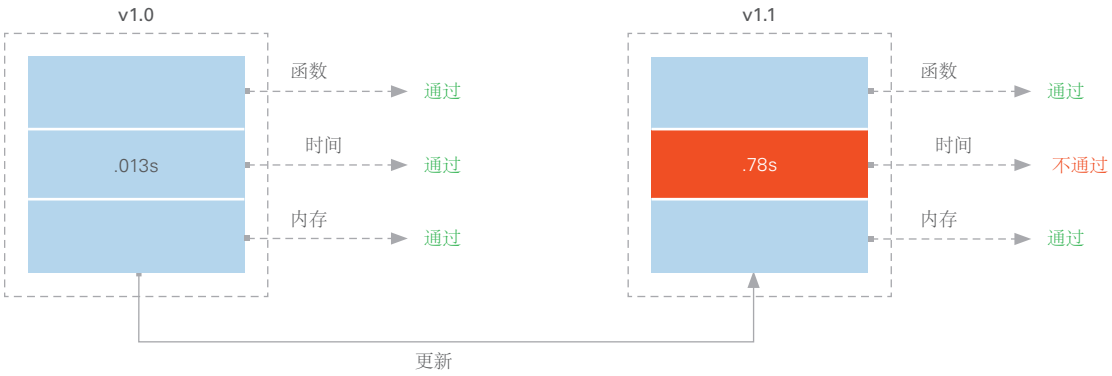


图8.对v1.0和v1.1执行单元测试后，更新后的处理时间被判定为一个问题，导致过程功能测试出现伪故障。

测试过程

为了节省开发时间，大多数测试系统中的回归测试与源代码控制、编译或版本管理同时进行。 这可允许重用测试代码，从而需要更频繁的更新。 然而，计划测试代码编译所需的开发时间和预算也很重要。 通常，回归测试是CI服务或IDE的组件，其中源代码控制、编译和测试按顺序执行。

最佳实践：

高级做法：对于所有测试系统，在将系统部署到新机器之前，应该进行一定程度的功能测试。功能测试的范围从使用模拟硬件在开发环境中手动运行应用程序等简单场景到基于配置文件运行一系列功能测试的复杂场景。 单元测试对于简单、单一的应用可能不必要，但是随着测试系统复杂性的增加，可能就需要进行单元测试。 随着添加的模块越来越多，特定的自定义测试对于跟踪漏洞或确保系统满足某些规格是必要的。

最佳实践：

高级做法: 复杂的测试系统不仅应该对每个新版本的测试系统进行各种输入功能测试，而且还要为系统的每个单独模块开发单元测试。 这两种回归测试方法都应该在部署过程中最有效的时间点上完成。 例如，功能测试应该在编译每个版本后进行，单元测试应该在每个源代码控制提交点执行。 通常，这些测试对于系统是强制性的或无需说明的，特别是在航空航天和国防工业中。

组件化

因为部署时间太长是大型测试系统的一个常见问题，所以最佳的做法是只更新测试系统中需要更改的单个组件而不是重新构建整个系统。 本指南的“依赖关系解析”一节部分解决了这一问题，但我们仍需要单独讨论如何开发模块化架构或基于插件的架构来实现更高效的部署。 无论工程师选择哪种架构，最佳做法都是定期更新外设模块，而更核心的模块则保持相对恒定，无需重新编译。 这种做法自然会产生有关更新频率的问题，稍后将在本节中探讨。

部署插件架构

对于软件部署，插件是指代码模块，其安装独立于主应用程序的安装，在功能上独立于其他插件，遵守全局插件接口，并且当用于编译的应用程序时可避免名称冲突。 主应用程序应该能够动态加载每个插件，通过标准接口调用每个插件，并使用每个插件作为扩展，而不需要重新编译。 成功开发后，插件框架可允许组件化部署、更新或安装特定或缺少的插件，而不需要重新编译主应用程序或任何未受影响的插件。

例如，为简单应用程序开发的插件框架可能包含主要可执行文件，会在加载时搜索插件目录，或在运行时执行定期搜索，并通过标准接口执行该插件。这样，插件可以连续部署到系统的插件目录中，而无需编辑主应用程序。

硬盘驱动器复制

通常，代码库、硬件驱动程序或特定文件是测试系统核心的一部分，不需要像其他模块化外设组件一样频繁更新。 在这些情况下，硬盘驱动器复制是将环境标准化为后续开发基准的一个好方法。 工程师可以将开发机器或初始测试机器的硬盘驱动器复制并克隆到其他测试机器上。 复制驱动器后，测试机器便拥有一个共同的起点，通常包括主测试应用程序或程序、必要的硬件驱动程序、系统驱动程序集和关键外设应用程序，如MAX硬件配置。 然而，重要的是要认识到，硬盘驱动器复制也有相应的注意事项，如测试机器之间需要相同的计算机硬件，或者占用大量内存的镜像，这使其不适用于软件频繁更新的情况。

使用硬盘驱动器复制为进一步测试开发奠定基础的一个例子是使用Symantec Ghost（一种常见的硬盘驱动器复制工具）和TestStand Deployment实用程序(TSDU)。在下图(A)的第一帧中，开发机器将其核心软件堆栈（红色）复制到目标机器上。这个核心软件堆栈是Windows操作系统、硬件设备驱动程序、运行引擎和MAX的组合。目标机器生成镜像之后，便可在开发机器上进行开发(B)，通过使用TestStand和LabVIEW（绿色）创建测试序列。然后，开发人员可以使用TSDU将测试序列移动到目标机器。对于测试序列的频繁更新，开发人员可以继续使用TSDU来节省开发时间，因为核心软件堆栈不需要更改。有时，开发可能是在未部署到目标机器(C)的开发机器上进行。系统不匹配可能会导致依赖关系缺失的问题。在这种情况下，开发人员可以选择重新为开发机器创建镜像，并将其复制到目标机器上，而不是使用TSDU更新目标机器，以重新对齐两个机器(D)。接下来，开发人员可以继续使用TSDU进行频繁更新，并且当将来出现系统不匹配时，可以使用Ghost重新为目标机器的硬盘创建镜像。

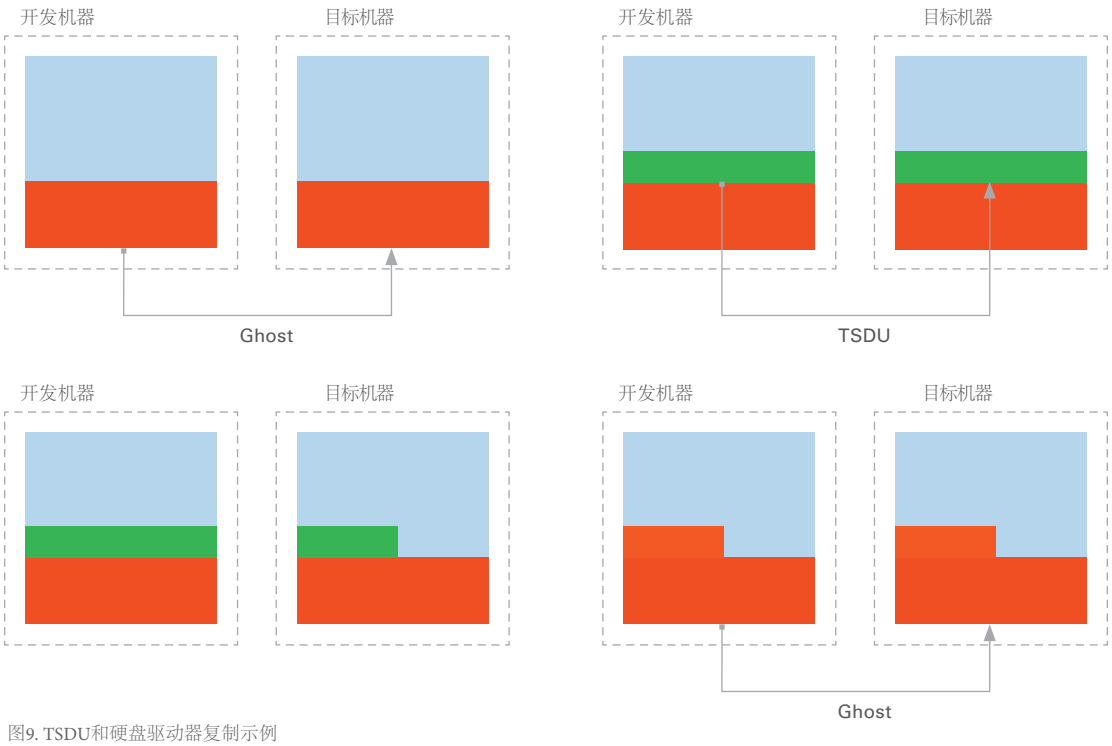


图9. TSDU和硬盘驱动器复制示例

持续集成和持续部署

持续集成(CI）是指通常在单独的CI服务器上持续提交、编译和测试代码。 在大多数测试系统中，CI服务用于为编译、测试和部署系统软件提供必要的框架。 这些服务提供各种配置选项，在CI服务器上定期自动运行，以创建编译计划、自动测试规则和版本部署等。 使用CI服务器最明显的优势之一是跟踪和管理不同编译文件和部署文件的能力。

编译

版本	状态	最后一次编译
1.2	未通过	2016年5月24日
1.1	通过	2016年4月2日
1.0	通过	2015年12月7日

部署

版本	状态	最后一次编译	机器
1.0	通过	2015年6月7日	A
1.1	未通过	2015年6月9日	B
1.1	通过	2016年3月16日	C

表2. CI服务提供用于跟踪应用程序编译和部署的仪表板。

不同的CI工具在功能上的差异很大。开源开发者和软件公司都有开发CI工具。 后者具有为系统设置提供支持的额外好处。

- **Jenkins—**Jenkins誉为“领先的开源自动化服务器”，是当今最受欢迎的CI服务之一，因为它可允许轻松的安装和配置。 Jenkins也可以使用几乎所有的编程语言，因为它可以通过它们的命令行界面或通过广泛的Jenkins插件与程序连接。
- **Bamboo—** 由软件公司Atlassian开发，是领先的专用CI服务。除了Bamboo提供的测试、编译和集成功能外，Atlassian提供了Jenkins所没有的“一流部署支持”。
- **Travis CI and Circle CI—**这两个开源的CI服务提供了极大的扩展能力，但仅与位于GitHub存储库中的项目集成。

总的来说，CI的目标是提供自动化且可配置的工具，使开发人员能够在编译和测试其软件时继续编码。

最佳实践：

基本做法：对于简单系统，组件化通常不是什么大问题。尽管系统使用非常少的代码模块或不使用插件架构，但是每个测试系统通常可以部署为独立应用程序。然而，如果安装时间变得非常久并且开始减慢部署速度，则可能需要采用更高程度的组件化方法，而不需要重新安装所有组件。

高级做法：当测试系统变得庞大、复杂或者使用插件架构时，是时候淘汰单一部署镜像，采用模块化部署来单独更新每个组件。 使用插件架构是实现这种模块化设置的快速方式，但也可以通过配置CI服务来实现。

实际情景

高级部署框架的一个案例是一家音频设备生产公司使用TestStand和LabVIEW对其产品进行功能性电气测试。 音频设备制造商的测试部门有50多个测试系统，遍布在全球各地。 每个系统使用一个PXI机箱来容纳各种模块，包括数据采集、数字I/O、数字信号采集、数字万用表和频率计数器板卡。

负责部署的测试工程师按照列出的步骤对每个要上线的新测试系统进行操作。

1.创建基本系统镜像

每个新测试系统都会有一个必要软件的列表，包括公司自己开发的软件和第三方软件，以确保系统的安全性。该公司的IT部门需要此软件，包括防病毒软件、VPN安全应用程序和Windows组策略配置规范。 其次，每个系统需要一个基本软件集来执行其必要的测试序列。这个软件的主要组件是一组与公布的NI系统驱动程序集进行交叉检查的驱动程序。也就是说，一个版本的测试系统可能包含NI-DMM 14.0、NI-Switch 15.1、NI-FGEN 14.0.1和NI-DAQmx 14.5驱动程序。 此外，还需要LabVIEW 2014和TestStand 2014的运行引擎来运行主测试系统可执行文件。 下面的图表概述了所有必要的软件。

软件	版本
NI-DAQmx驱动程序	14.5.0
NI-DMM驱动程序	14.0.0
NI-Switch驱动程序	15.1
NI-FGEN驱动程序	14.0.0
LabVIEW运行引擎	2014
TestStand 运行引擎	2014
内部防病毒软件	3.2

表3.创建基本系统镜像时，重要的是明确列出所需的驱动程序和运行引擎版本。

部署到新测试系统的第一步是在开发机器上创建此镜像，并使用硬盘驱动器镜像软件复制该镜像。 此软件镜像可能是之前创建的，因而有可能在多个机器上重复使用。 这有助于降低部署成本，因为每批次相同的测试机器只需要进行一次安装。

在通过安装所有必需的软件生成基本系统镜像后，使用Symantec Ghost来复制硬盘驱动器并将新镜像上传到编译服务器。 编译服务器位于总部，并且只需要维护存储位于服务器上的多个系统镜像的大容量存储器。

2.部署基本镜像

将基本镜像上传到编译服务器之后，测试工程师会将新的测试系统连接到公司网络，然后使用web界面连接到镜像服务器并浏览可安装的各种基本系统镜像。在选择适当的版本后，Symantec Ghost使用镜像对新系统的硬盘驱动器进行镜像。 此时，测试系统具有执行测试序列所需的基本必需软件。

3.验证硬件

在将必要的硬件模块物理安装到PXI机箱并打开系统后，测试工程师需要在软件中将系统别名映射到实体设备。 尽管提供模块与关联插槽编号的列表，测试工程师必须使用配置系统设置来映射别名，以便模块位置可以在系统之间移动。 对于该公司，每个测试系统使用工程师编辑的.ini文件，提供实体系统硬件至测试系统别名的映射。 这通过在MAX中识别设备并手动编辑.ini文件以创建相应的映射来实现。

4.安装应用程序和组件

此时，测试系统已经安装了基本系统镜像并验证了实体硬件。现在，工程师的任务是安装最新版本的测试应用程序。在这种情况下，应用程序是由TSDU生成的TestStand安装程序，包含了所有必需的代码模块、序列文件和支持文件。为了解释这个安装程序是如何生成的，我们需要看一下生产公司采用的开发系统。每个开发人员在LabVIEW中创建一个特定的测试步骤或在TestStand中创建测试序列，并将它们提交到一个Apache Subversion源代码控制存储库。此存储库位于运行CI服务（Jenkins）的服务器上。 Jenkins服务运行所提交代码模块的测试，接着TestStand序列分析器使用命令行对序列进行验证，然后使用TSDU命令行界面将必要的测试序列编译到安装程序中。在编译每个安装程序之后，使用Jenkins Deploy Plugin将安装程序及其必需的支持文件自动部署到编译服务器上。

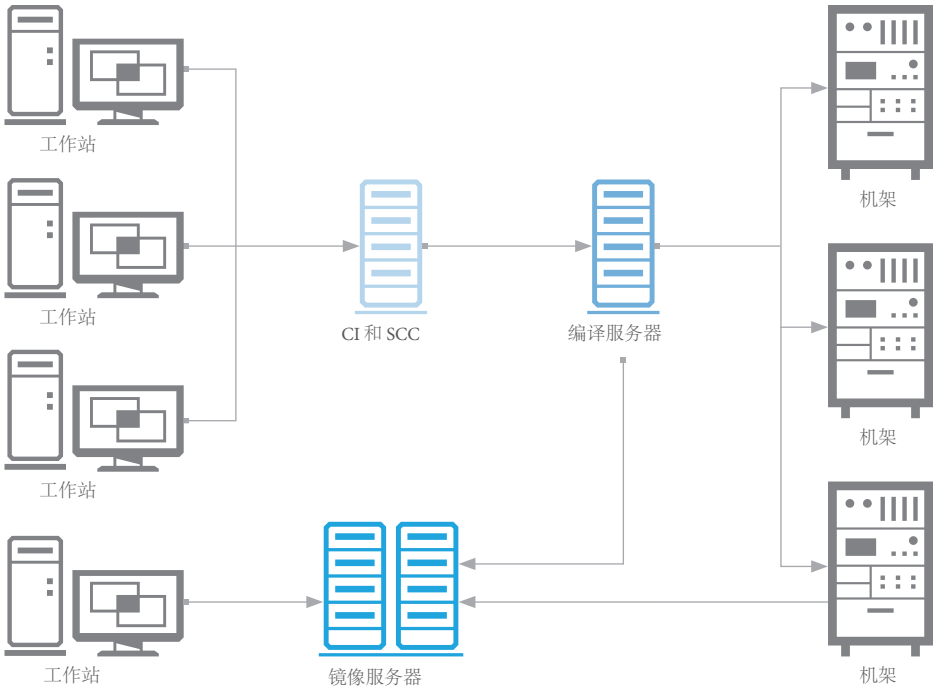


图10.该测试部署系统使用镜像服务器来存储和部署基本系统的镜像，以保持各种测试站彼此同步。然后，开发人员定期将源代码上传到持续集成和源代码控制服务器，以定期编译和测试提交的代码。一旦提交的代码通过所有必要的测试，内置镜像将添加到编译服务器，由编译服务器处理该测试软件系统镜像的大规模发布。

5.执行

将TestStand安装程序放在编译服务器上后，测试工程师可以将安装程序下载到新的测试系统上。 然后，工程师可以运行安装程序，找到主测试可执行文件，并开始运行基本测试系统。

使用这个部署系统，测试工程师可以快速轻松地对每个测试系统进行更改。硬盘驱动器镜像系统可用于大规模代码修订或驱动程序集升级，而较轻量级的编译服务器可用于部署对主测试应用程序或单个组件和插件的小更改。

总结

测试系统部署通常可能是一个复杂的过程，尤其是随着测试系统的复杂性和数量不断增加。在开发测试系统的早期建立正确的部署过程是实现可扩展的成功部署的关键。创建成功部署过程的第一步是要确定和定义所有必需的测试系统组件，并且采用正确的部署方法。动态硬件配置选项也是许多部署系统的重要考虑因素。对于更大型的高级系统，动态解析部署镜像和目标机器之间的依赖关系有助于降低部署过程的复杂性以及升级或重新镜像系统所需的时间。管理和测试部署镜像的每个版本是测试系统开发人员需要考虑的另一个重要因素。无论是采用持续集成服务还是配置文件，都必须维护一个可扩展的版本管理系统来实现分布式部署。测试系统的部署方法必须针对系统的功能和性质进行高度自定义。本指南的各个章节提供了构建可扩展解决方案所需的建议，不管使用何种工具或系统有哪些功能。

TestStand Deployment Utility

The TestStand Deployment Utility可自动化执行部署中的许多步骤，包括收集测试系统的序列文件、代码模块和支持文件，然后为这些文件创建安装程序，从而简化了部署TestStand系统的复杂过程。

了解有关TestStand Deployment Utility的更多信息

LabVIEW Application Builder最佳实践

LabVIEW Application Builder最佳实践可简化LabVIEW应用程序的管理和组织。 这些可建议帮助工程师在开始开发之前制定指南和步骤，以确保其应用程序适用于大量VI和多个开发人员，从而节省开发时间和资源。

开始使用Application Builder最佳实践开发LabVIEW项目

测试系统构建基础知识

系统维护

目录

引言

理想的系统永远不会出现故障。安装系统、打开系统，然后运行系统，之后就不需要去管它们，直到10年、15年或20年或更长时间后淘汰系统。 不幸的是现实并非如此，或者说至少还没有实现。 系统故障和突发的意外故障可能会带来高额的损失。 即使有最周全的计划，我们也不能完全消除故障的风险，但我们可以减少这些风险。 维护策略可以帮助您管理此成本并降低故障风险。

维护计划对于确保自动测试设备(ATE)系统在整个生命周期内具有最低总体拥有成本至关重要。 具有可维护性的系统与完善的维护计划相结合，有助于：

- 通过维护功能和延长使用寿命来最大化资本投资收益
- 通过管理物流、计划表和备件库存来最大限度地减少停机成本

任何维护计划的目标都是保持系统尽可能长时间地正常工作，而且在系统停机时尽快使其恢复正常工作。 另外，还要以尽可能低的成本达到这一目的。

概念和定义

维护是执行服务来维持系统正常运行并在系统故障时维修系统的活动。 维护分为三个方面：预测性维护、预防性维护和纠正性维护。

可维护性是指进行维护的容易性。 一些行业将其称为可服务性。 可维护性越高，控制维护成本就越容易。

预测性维护通过状态监测在系统故障发生之前进行检测，在工业中称为基于状态的维护。当预测到潜在故障时，就会安排维护活动来对系统进行维护。这些活动可以延长系统使用寿命，并避免意外停机。只有在检测到维护需要时，预测性维护活动才会开展，有计划地安排停机，这通常比计划外停机的成本低。计划内停机成本可以分摊到接受维护的许多其他系统。目标是在故障发生前尽可能长时间地使用系统/组件，从而最大限度地提高资本投资收益，并尽可能减少意外停机成本。随着物联网以令人难以置信的步伐前进，智能机器的概念已经深入人心，智能机器可以自我监控，并在需要维护时与其他机器组成的网络进行通信。传感器、嵌入式控制器、FPGA、网络和大模拟数据分析等领域的技术进步使得预测性维护更加容易，并且比以往更具成本效益。预测性维护的衡量指标是停机时间；这个时间称为平均预测维护时间(MPDMT)。

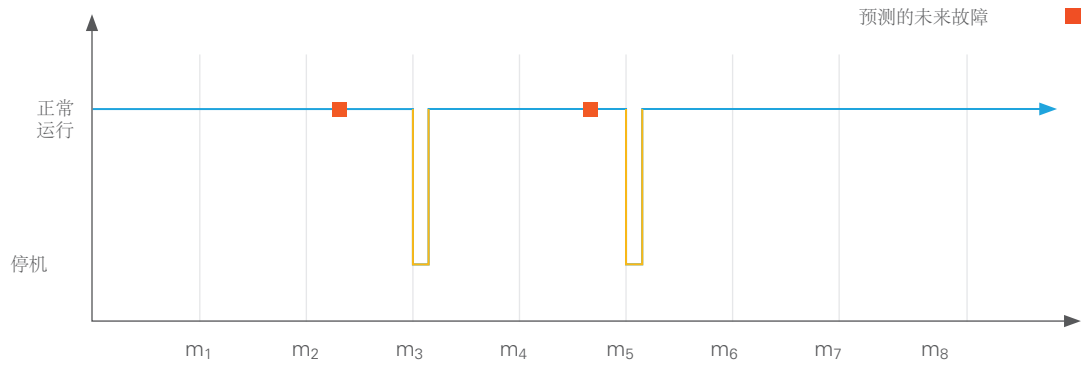


图1.查看预测性维护的正常运行时间和停机时间随时间的变化。预测性维护可最大限度地利用您的资本投资，并通过降低停机频率以及将昂贵的计划外停机时间转换为传播较低的计划内停机时间，从而最大限度地减少停机成本，但需要故障监测设备和预测软件。

预测性维护活动包括：

- **状态监测** —这可确保系统正常运行、检测故障的发生，并识别可能导致系统故障的组件隐患或性能退化。基于经济高效的嵌入式微处理器和FPGA技术，内置自检和状态监测技术已经得到广泛应用。 这有时被称为预测和健康管理(PHM)或系统健康监测。 该概念是指检测系统中的性能变化和隐藏故障，防止出现更严重的系统故障。

  今天，大多数汽车都配有发动机自动健康监测系统，用于检测问题并及时闪烁发动机检查灯，让发动机在永久损坏之前得到维修。测试系统可以监测温度、风扇速度、内存使用、测试时间、测量准确度、继电器操作计数等。

- **维修系统组件** — 这有助于减缓磨损并延长系统的使用寿命。

  一些汽车轮胎会配有检查空气压力的传感器。不正确的气压会缩短轮胎的使用寿命，影响油耗性能。如果测试系统在灰尘较多的环境中使用，则可能需要清洁空气过滤器和外壳内部的灰尘，以免其过热，导致电子设备的使用寿命缩短。监测系统的内部温度或气流可以帮助您了解何时可能需要清洁灰尘过滤器。

- **更换系统组件** - 在组件故障之前更换组件，以避免意外停机。

  测试系统可以使用继电器来开关信号，以测试待测设备(DUT)。 取决于所切换的电负载，继电器仅可持续预计的操作次数。 因此，监测操作次数并在继电器故障之前更换继电器模块通常比等到发生故障并出现计划外停机再采取措施更具成本效益。

- **校准以补偿漂移** - 测量系统的目的是提供可信的测量数据。 如果测量数据不可信，则系统处于不正常运行状态。

  大多数测试系统的电子器件需要以一定时间间隔进行校准。 但是，如果使用的是尖端技术，可能我们还不能很好地把握校准间隔。 因此，可通过监测测量漂移来了解何时适合需要安排校准维护。

- **验证**- 这可确保系统在重新联机之前正常工作。 如果在故障情况下联机，只会增加停机时间。

- **系统重新联机** - 必须始终考虑这一点，因为对于某些应用，这不是一个简单的任务。

  例如，如果测试是制造过程的一部分，则让系统重新联机可能需要停止生产线并且使测试装置与生产流程重新同步。

预防性维护是维护系统以防止系统故障和延长使用寿命的活动。 预防性维护活动通常需要进行安排，并导致计划内停机。 计划内停机成本可以在接受预防性维护的许多其他系统中共享。 目标是最大限度地减少意外停机成本。 预防性维护的措施是发生的停机时间; 这一时间称为平均预防保持时间（MPMT）。

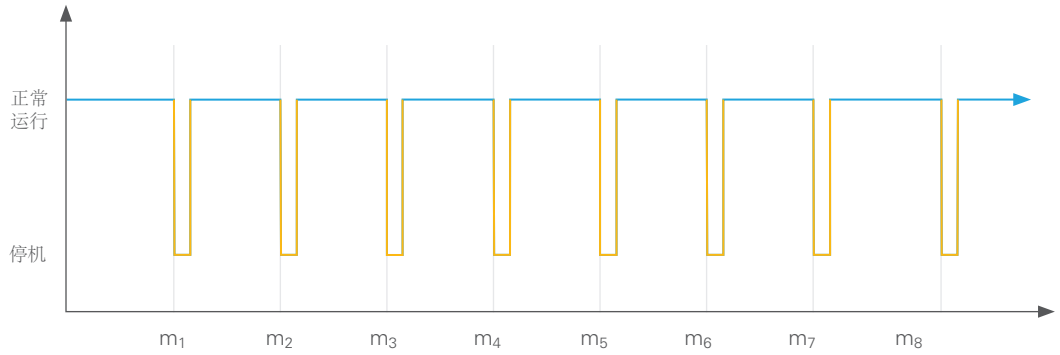


图2.预防性维护并不能总是最大限度地利用您的资本投资，但通过避免昂贵的计划外停机时间来最大限度地减少停机成本。

预防性维护活动包括：

- **维修系统组件** - 这有助于减缓磨损并延长系统的使用寿命。

  这就是为什么汽车机油需要定期更换的原因。 测试系统上运行的复杂软件程序可能存在最终导致系统故障的潜在资源泄漏和/或故障。 简单的系统重启可以将软件刷新到新状态。 如果测试系统在灰尘较多的环境中使用，则可能需要清洁空气过滤器和外壳内部的灰尘，以免其过热，导致电子设备的使用寿命缩短。 如果不能监测温度和/或气流，则可能需要安排定期维护。

- **更换系统组件** - 在组件故障之前更换组件，以避免意外停机。

  汽车上的轮胎或刹车片在汽车行驶一定里程后需要更换，以避免出现故障，导致事故发生或停在半路。 测试系统可能配有测试该设备的连接器引脚，可能在100,000次连接之后磨损。 如果每小时测试50个设备，则连接器应该在磨损发生故障之前持续工作大约2,000个小时或83天。 预防性维护应该每隔约80天安排一次，更换连接器。 在发生故障之前更换通常比等到故障发生并出现计划外停机再采取措施更具成本效益。

- **校准以补偿漂移** - 测量系统的目的是提供可靠的测量数据。 如果测量数据不可靠，则系统处于不正常运行状态。

  大多数测试系统的电子器件需要以一定时间间隔进行校准。

- **验证**- 这可确保系统在重新联机之前正常工作。 如果在故障情况下联机，只会增加停机时间。

- **系统重新联机** - 必须始终考虑这一点，因为对于某些应用，这不是一个简单的任务。

  例如，如果测试是制造过程的一部分，则让系统重新联机可能需要停止生产线并且使测试装置与生产流程重新同步。

纠正性维护是修复故障系统以让其恢复到工作状态的活动。 纠正性维护活动通常不会事先安排，因而 会导致计划外停机。目标是在故障发生前尽可能长时间地使用系统/组件，从而最大限度地提高资本投资收益，并在故障发生之后尽可能减少意外停机成本。纠正性维护的衡量指标是发生故障后的停机时间；这个时间称为平均预测维护时间(MPDMT)。

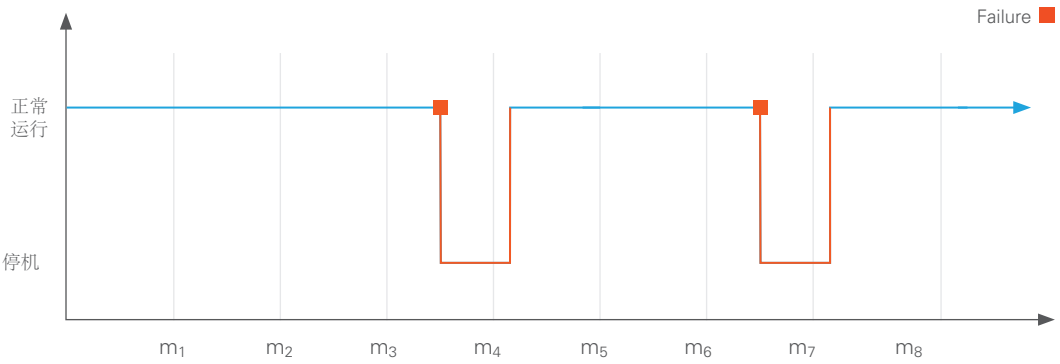


图3.查看纠正性维护的正常运行时间和停机时间随时间的变化。 纠正性维护可最大限度地利用您的资本投资，但不会将停机成本最小化，因为它是计划外的。 您可以采取措施将计划外停机时间或MTTR的持续时间最小化。

纠正性维护活动包括：

- **检测** - 尽可能快地检测系统故障，最小化昂贵的计划外停机时间，并且尽可能防止对系统中的其它组件和/或同一流程中使用的其他系统的损坏。

  汽车中的压力传感器可以尽快检测到油压下降，向驾驶员发出警告并防止对发动机的永久损坏。 如果油泵出现故障或油位由于泄漏而变得很低， 修理油泵或将泄漏处密封然后加油要比购买新的发动机要便宜得多。 对于ATE系统，电子元件可能会故障，从而影响关键测量并导致测试结果不正确。 如果故障需要时间来检测，则公司可能在不知情的情况下将不合格产品提供给客户，或者如果冷却风扇故障，机箱温度可能升高到会损坏部分电子设备的水平。

- **诊断和隔离** - 在检测到故障后进行正确诊断和故障隔离可以帮助操作人员和维护人员快速找到并维修正确的组件，从而最大限度地减少意外停机时间并节省成本。

  汽车机械系统拥有复杂的诊断设备来有效且高效地诊断问题。 这通过降低修理或更换故障部件的风险来节省时间和金钱。 对于复杂的ATE系统也是如此，如果没有适当的诊断工具，光是诊断问题，就可能花费几小时甚至几天。

- **维修** - 通过维修或更换发生故障的组件来修复系统。 计划外停机时间很大程度上受到备件可用性的影响。 取决于应用、环境和人员的技能水平，就近提供备用系统或备用部件可能不一定具有成本效益或切实可行。

  大多数人不会在没有备胎的情况下开车穿越全国，但很可能他们只需要开车去几个城市街区。

  备件战略对于控制成本至关重要。需要考虑以下问题：备件是否应存放在现场或附近的服务中心？如果要求供应商提前从工厂发出替换件，是否需要支付费用，或者只能等到设备故障再返给供应商维修？答案取决于计划外停机的成本。组件的数量、组件的平均故障间隔时间（MTBF）和补充备件所需的时间决定了所需备件的数量。一些公司提供不同级别的备件服务来帮助评估所需备件的数量、物流和备件管理成本。

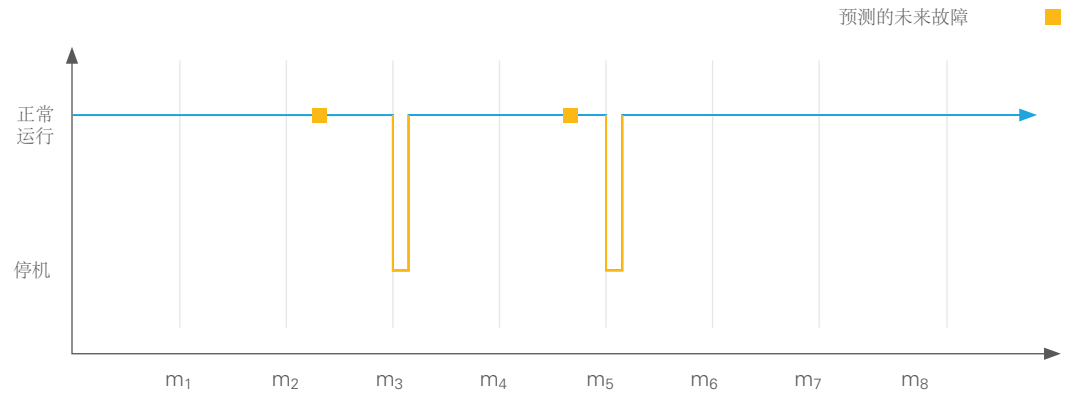
- **验证** - 这可确保系统在重新联机之前正常工作。 如果没有此步骤，系统可能仍然不正确运行，只会导致更多的计划外停机。

  这就好比汽车的刹车修好了之后，直接在高速公路上高速驾驶汽车，而没有先测试和验证刹车是否能够正常工作。

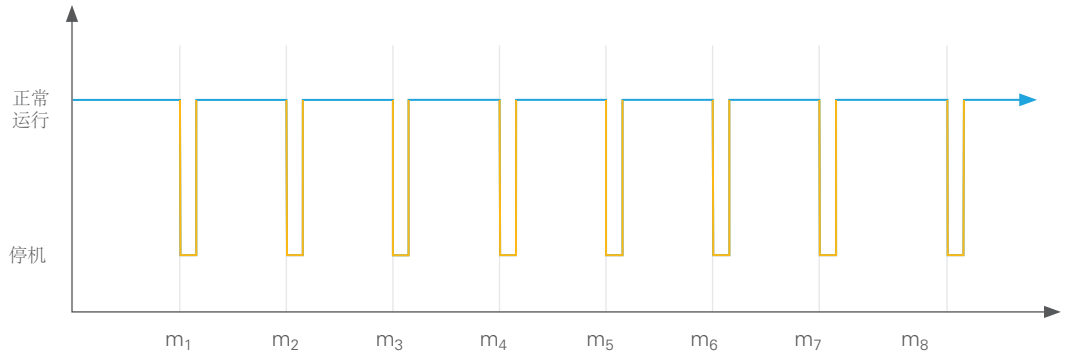
- **系统重新联机** - 必须始终考虑这一点，因为对于某些应用，这不是一个简单的任务。

  例如，如果测试是制造过程的一部分，则让系统重新联机可能需要停止生产线并且使测试装置与生产流程重新同步。

预测性



预防性



纠正性

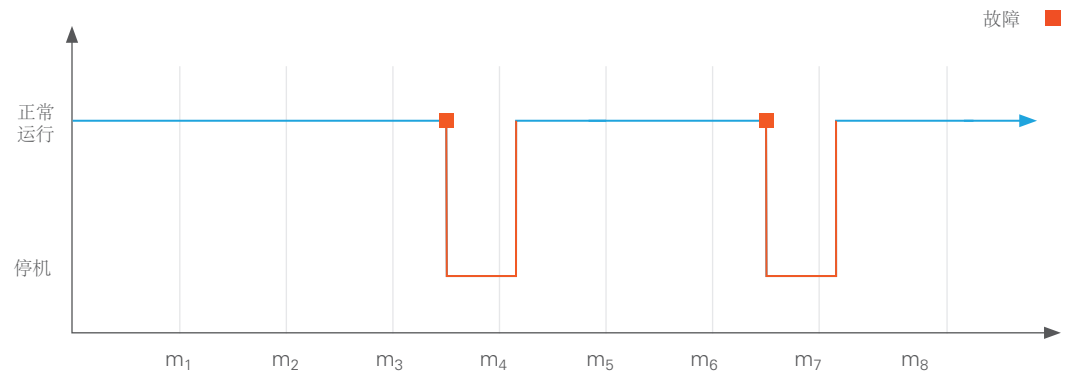


图4. 对比一下正常运行时间和停机时间，可以看出计划外停机的成本通常比计划内停机时间要高得多。

可维护性设计

系统的设计极大地影响了部署有效、高质量和可控维护计划的能力。 在设计高维护性的自动化测试系统时，建议考虑以下最佳实践和指南。

自检和监测

自检和监测对于减少计划内和计划外停机非常重要。 从头开始设计自检和监测功能对于实现高效、有效的健康监测、故障检测、故障诊断和隔离以及系统验证至关重要。

模块化设计

模块化设计可简化维护、更换、维修和校准系统组件相关的活动，减少相关的时间，同时还优化系统诊断和故障隔离，并在计划外停机期间节省宝贵的时间。 此外，模块化设计还可降低与备件相关的成本。您无需在库存中备用几台完整的系统，只需备用一些组件、子系统或模块。 组件通常具有不同的故障率 - 故障率较低的组件需要较少的备件，而故障率较高的组件需要较多的备件。

标准化

标准化可以大大降低成本，因为它简化了物流，减少了备件数量、所需的维护工具和设备数量以及培训成本。

例如，一些航空公司使用了10种甚至更多类型的飞机。然而，西南航空公司只使用一种飞机——波音737。这有助于成本节省。机械师只需针对一种类型的飞机进行培训，而且库存 也只需准备一种飞机的备件。他们可以在最后一刻换出飞机进行维修。机队的飞机是完全可互换的。所有飞行员和地勤人员都非常熟悉这种飞机。而且，在飞机的存放方式和存放位置方面完全没有任何挑战，因为所有飞机都具有相同的形状和尺寸。

标准化极大地简化了控制维护过程。良好控制的过程是可重复且可预测的，因此我们设计的系统必须采用一种统一的方式来完成各种维护任务。如果西南航空公司的维修人员使用不同的工具以及进行不同的维护任务，则每个维修人员所交付的成果质量各不相同，完成任务所花费的时间也各不相同，这使得维护成本难以控制和管理。

简易性

保持操作和维护尽可能简单。 换句话说，让正确的事情简单做。 这减少了所需的文档记述和培训量，提高了工作的一致性，并减少了进行维护所需的时间。

环境和人为因素

始终考虑环境和人为因素。 例如，如果系统在多尘环境中使用，则其可能需要在通风口上安装灰尘过滤器。 维护过滤器是否容易？ 系统是否需要脚轮，可以移动来方便维护？ 如果是，请确保它们具有合适的重量且适合所处的地形。 操作人员和维护人员的技能水平如何，需要多少培训？是否能以用户友好的方式设计硬件和软件接口吗？

设计原则	预测性	预防性	纠正性
自检和监测	▪ 状态监测 ▪ 功能验证	▪ 功能验证	▪ 检测故障 ▪ 诊断和定位故障 ▪ 功能验证
模块化设计	▪ 状态监测 ▪ 维护 ▪ 更换 ▪ 校准 ▪ 功能验证	▪ 维护 ▪ 更换 ▪ 校准 ▪ 功能验证	▪ 检测故障 ▪ 诊断和定位故障 ▪ 维修 ▪ 功能验证
标准化	▪ 状态监测 ▪ 维护 ▪ 更换 ▪ 校准 ▪ 功能验证 ▪ 优化任务一致性	▪ 维护 ▪ 更换 ▪ 校准 ▪ 功能验证 ▪ 优化任务一致性	▪ 检测故障 ▪ 诊断和定位故障 ▪ 维修 ▪ 功能验证 ▪ l优化任务一致性
简易性	▪ 降低文档记述和培训成本 ▪ 优化任务一致性	▪ 降低文档记述和培训成本 ▪ 优化任务一致性	▪ 降低文档记述和培训成本 ▪ 优化任务一致性
环境和人为因素	▪ 降低预测性维护事件的频率和/或MPdMT ▪ 减少人为失误 ▪ 提高安全性	▪ 降低预防性维护事件的频率和/或MPMT ▪ 减少人为失误 ▪ 提高安全性	▪ 降低故障率和/或MTTR ▪ 减少人为失误 ▪ 提高安全性

表1. 该表格简要概括了每个设计原则在每种维护方法上的具体体现。

维护策略

应该使用哪种方法？ 预测策略会等到检测到潜在的未来故障，然后在方便的时间安排维护或替换。 预防策略在定期计划的时间间隔主动维护、更换和/或校准系统组件，以最大限度地降低故障风险和计划外停机时间的成本。 纠正策略会等到某个组件故障来最大限度地利用资本投资，然后尽快进行维修，以最大限度地减少意外停机的成本，或最小化MTTR。 对于每个策略，您可以自己执行、与供应商制定服务协议，也可以什么也不做，在发生故障时持乐观态度，当然这是不推荐的。

以下介绍了不同的技巧组合，并解释了哪种维护策略最适合用于不同子系统或组件。 此处讨论的方法是状态监测可行性、基于可靠性的维护(RCM)和故障成本分析。 RCM基于运行时间对系统组件故障率和组件故障成本的影响。以下三个图显示了三种策略的故障率与运行时间的函数关系。 每个图描绘了不同类型的组件的特性。 实际应用当然不止这三种场景，但这些是最常见的，可帮助您理解RCM的工作原理。

图5显示了故障率随时间而增加。在这种情况下，组件的故障率可能一开始表现为恒定，但在系统达到预期使用寿命之前开始进入磨损。换句话说，部件的使用寿命明显短于系统使用的时间长度。这可能是最直观的场景，因为风扇、连接器、机电继电器、固态硬盘驱动器、电池、电子校准等机械组件都有这种趋势。在每次预防性维护事件发生之后，故障率会降低回到出厂水平，从而恢复系统的可靠性。

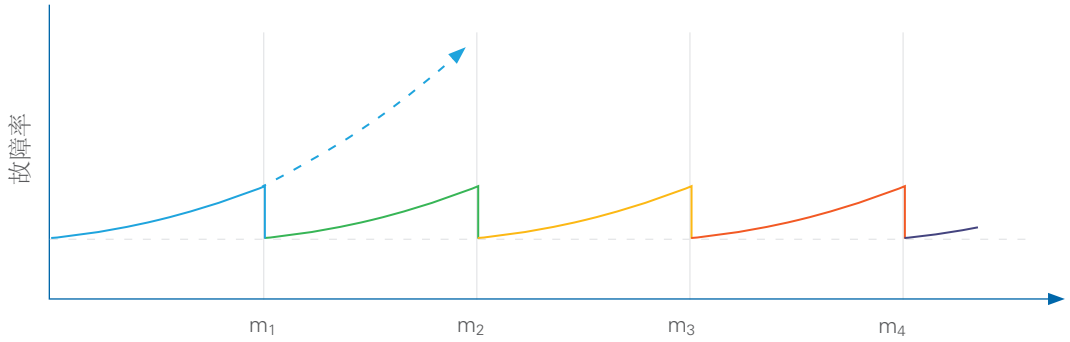


图5. 预防性维护会在故障率升高时启动维护事件，使故障率降低回出厂水平。

图6显示了故障率随时间保持恒定，有时称为稳态故障率。 在这种情况下，组件仅在超出系统的预期使用寿命（这不包括校准）才开始磨损。 换句话说，部件的使用寿命远远超过系统使用的时间长度。 这一场景常见于电子器件，比如IC、电阻器、陶瓷电容器、二极管、电感器等。 现代电子通常具有远远超过10至15年的使用寿命。 实际上它们在测试系统淘汰之前不会磨损。

在每个预测性维护事件之后，故障率不会改变，因此在组件发生故障之前更换组件没有任何好处。 从数学上来说，这种故障率被视为“随机机会”。 因此，使用新组件替换正在运行的旧组件并不能提高系统可靠性。

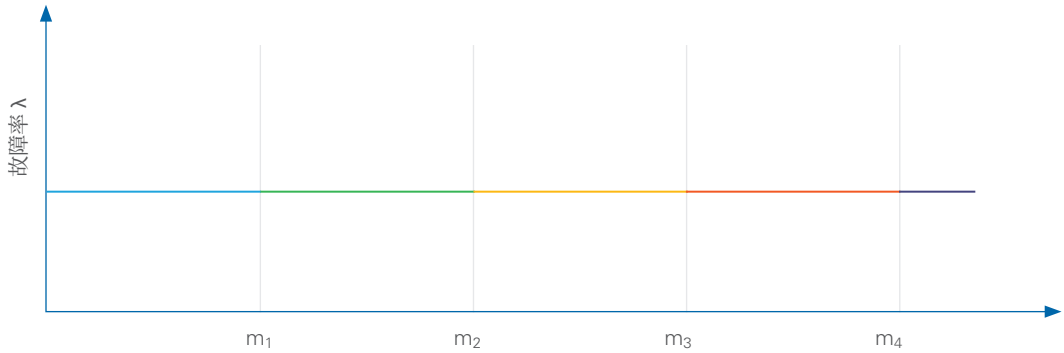


图6.对于预防性维护，由于故障率随时间保持恒定，每个维护事件对故障率几乎没有影响或完全没有影响。

图7显示了故障率随时间而降低。 这可能是最不直观的情况，常见于软件和复杂的计算机系统。 对软件和固件进行重大升级或添加新功能、新技术等可能会导致缺陷（错误），增加系统故障率。 在每次预防性维护事件之后，故障率升高到更高的水平，从而降低系统的可靠性。 但是，有些情况下我们必须升级软件，例如操作系统更新或硬件过时。

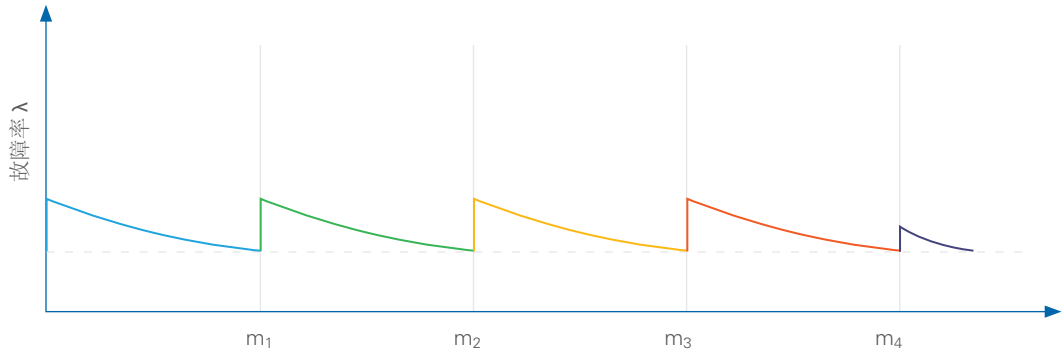


图7. 故障率随时间下降，预防性维护实际上在每次维护事件之后提高了故障率。

此外，有时候我们没有足够的数据来知道故障率是随时间增加、保持不变还是减小。 这常见于新产品、技术或设计。使用预测性维护策略来监测故障随时间的变化有助于了解组件的情况，前提是监测的成本相比故障的成本更具效益。 即使未建立趋势，预测性策略通常会最大化您的资本投资收益，同时最大程度降低停机成本。

使用此方法为整个系统开发维护策略时，可以将系统分解为多个子系统和/或组件，然后评估每个组件，确定最佳的维护策略。 以下提供了一些推荐的指导：

- 在引起系统故障之前，是否可以在一开始就检测到组件故障？

  考虑到纠正性维护事件的故障成本和预测性维护事件的额外计划停机时间，对该组件故障进行状态监测是否具有成本效益？

- 该组件的故障率是否随时间增加、保持恒定或减少，或者存在其他趋势，您知道吗？
  - 故障是否重要，故障成本是否很高？

下图显示了一个决策流程图，可帮助您为系统的每个组件和故障模式选择最佳策略。 当然，这个流程图应该根据实际需要进行调整。

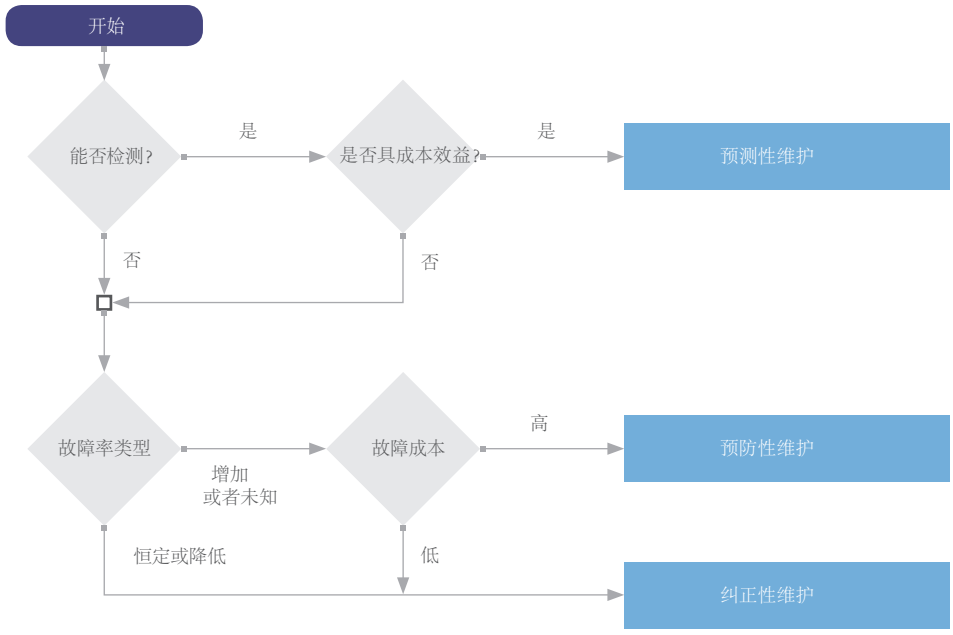


图8. 维护策略决策流程图

ATE系统示例

基于PXI Express的ATE系统由基本组件或子系统组成，每个组件或子系统可以根据其维护策略分为较小的组件。

机箱

机箱背板可能难以通过监测来发现潜在故障。 它具有恒定的故障率，使用寿命为10至15年甚至更长。 电子器件基本上都是数字的，不需要校准。 纠正性维护策略或“运行至故障”是最好的方法。

此示例中的机箱电源不提供监测功能。 电源通常使用较大的液体电容器，有些还配有冷却风扇。 根据负载和环境条件，这些部件的典型使用寿命约为7至10年。采用预测性或预防性维护策略较为合适。

我们可以监测机箱风扇速度和机箱温度。如果速度开始减慢，或者如果机箱温度开始增加，则会发送警告并且在不久的将来在合适的时间安排维护。 这一场景适合采用预测性维护策略。

控制器

控制器的集成电路和电子部件（不包括硬盘和RAM）可以提供监测工具来识别潜在故障。在本例中，实现这些功能需要大量的开发时间，并且不具有成本效益。 控制器还具有恒定的故障率，应具有10至15年或更长的典型使用寿命。 电子器件基本上都是数字的，不需要校准。纠正性维护策略或“运行至故障”通常是最好的方法。

控制器的RAM具有可自动运行的错误校正码(ECC)，可以通过监测找到并纠正错误。 如果这些错误的频率持续增加，则可能需要安排时间对RAM进行更换。RAM不需要校准，适用于采用预测性维护策略。

在此示例中，控制器的硬盘驱动器是一个固态硬盘驱动器(SSD)，可监测读写次数。SSD经过一定次数的读写后就会发生损耗。因此，当读写次数接近损耗值时，应当安排更换SSD。SSD不需要校准。 预测性维护策略是最好的方法。

软件具有一些独特的特性 - 它不会磨损，并且不受环境的影响，只会因为设计缺陷或错误而出现故障。我们可以监测资源泄漏，如内存使用和碎片化；然而，许多故障在崩溃之前是无法发现的。软件的一个优点是不会磨损，只需重启系统就会正常工作。这解决了所有问题，直至漏洞再次导致软件崩溃。因此，每周一次或每月一次重启软件的预防性维护策略可以解决许多问题。软件可靠性一个更具挑战性的方面是升级。由于需要新的功能或与其他软件包的兼容性或者可能需要补丁来修复漏洞，软件需要偶尔进行升级。问题是，每次引入新的软件就会改变整个生态系统，从而可能会引入更多的漏洞。只有这样做之后才会知道结果。这种动态变化使得软件升级后软件故障风险立即上升，然后在运行一段时间后稳定下来。最常用于软件的升级维护方法是延迟升级，需要时才升级。

仪器模块

仪器模块上的集成电路可能难以通过监测来发现潜在故障。 它们具有恒定的故障率，使用寿命为20年或更长。 模拟电子器件可能随时间发生漂移，因此需要校准。 解决漂移问题需要采取预防性维护策略来进决校准。 许多校准实验室可以在校准后对模块进行最终验证测试，以证明一切正常。 这个测试可以有效地捕获故障或濒临故障的其他电子元件。 但是没有测试是完美的，对于电子器件的部分其它故障模式，可能适合采用校正维护策略或运行到故障方法。 此时，组合策略是最好的方法。

开关模块

开关模块的基板主要由集成电路组成，通常没有合适的工具来监测电子器件的健康。开关模块具有恒定的故障率，典型使用寿命为10至15年或更长。 电子器件基本上都是数字的，不需要校准。 纠正性维护策略或运行到故障方法是最佳方法。

开关的机电继电器可使用工具来监测操作次数。 继电器经过一定次数的操作后会磨损，这取决于所切换的电负载。 您可以使用制造商提供的数据和公式来估算开关的数量。 因此，当操作的数量接近磨损值，应安排更换开关。 开关模块不需要校准。 预测性维护策略是最好的方法。

电缆

固定电缆基本上是连接的，从不断开，或者在极少情况下需要重新连接，不会产生任何影响。固定电缆几乎不会故障，除了振动或人为滥用。 其故障率是恒定且非常低的。 纠正性维护策略是最好的方法。

动态电缆经常连接和断开，且经过一定次数的重新连接后会发生磨损。 故障率随时间增加并且检测潜在故障可能没那么容易，但可以估计。 可允许的重新连接次数需要询问制造商。 如果重新连接的平均次数已知，并且知道每小时、每天、每个单元需要多少次重新连接，那么就可以安排预防性维护。 此时，预防性维护策略是最好的方法。

子组件	预测性维护	预防性维护	纠正性维护
机箱背板	—	—	√
机箱电源	—	√	—
机箱风扇	√	—	—
控制器母板	—	—	√
控制器RAM	√	—	—
控制器固态驱动器	√	—	—
控制器软件	—	√	—
仪器模块	—	校准	√
开关模块羁绊	—	—	√
开关模块继电器	√	—	—
固定电缆	—	—	√
动态电缆	—	√	—

表2. 使用此维护策略适用于基于PXI Express的ATE的每个主要组件。 请注意，每个组件的最佳策略因应用的具体情况而异。

结论

预测性、预防性和纠正性方法各有其优势、挑战和适用情况。在大多数情况下，与维护相关的最大费用是计划外停机成本（故障成本）。通过状态监测和预测将计划外停机转换为计划内停机通常是有好处的。

每年，状态监测设备、网络、服务器和大模拟数据（Big Analog Data™）分析都会不断降低成本并提高性能，因此工业趋向于采用更智能的设备和预测性能更高的维护。对于意外停机不可避免的情况，有效的备件和维修策略是最小化和管理维护成本的关键。

具有可维护性的系统与完善的维护计划相结合，将帮助您管理故障成本和降低故障风险，避免昂贵的计划外停机，从而降低了维护成本和总体拥有成本。自检、模块化设计、标准化、简单性和环境/人为因素是设计可维护性的基本组成部分。

附录：维护成本

许多公司主要根据测试设备的价格来做出采购决策，而没有考虑部署、操作和维护设备的成本。 他们甚至很少考虑设备停机的成本。测试系统整个生命周期内的停机（或故障）和维护成本远高于购买价格，通常达到两到三倍。 最大的罪魁祸首是停机或故障成本。 这就是为什么我们需要制定维护计划以及系统的可维护性变得越来越重要的原因。

本附录提供了一个简单的总维护成本(TCM)模型，可用于估计系统在其使用寿命内的潜在停机和维护成本。 计算测试系统的TCM可能非常乏味和复杂。 该模型提供了一定复杂度的详细估算，对于大多数应用来说是足够且可管理的。

总维护成本 (TCM)

$$TCM = CD + M$$

CD = 计划外和计划内停机成本

M = 维护成本

通过将投资的维护费用(M)与在系统生命周期内减少的停机成本(CD)或者TCM成本的总减少量进行比较，就可以衡量维护计划的投资回报率(ROI)。 一些公司将计划内停机的成本与维护成本相结合，并将其与计划外停机时间的成本进行比较，因为他们的主要重点是避免意外停机和故障。 每个公司可能有自己的方法来估算TCM和维护的ROI，取决于公司想要跟踪的指标。

停机成本 (CD)

停机成本有时看起来像“运气”钱，因为一些公司发现他们很难估计该成本。但是停机成本是实实在在的。停机有两种类型：计划内（安排的）和计划外（未安排的）。维护计划的目标是尽可能减少所有停机时间，并将尽可能多的计划外停机转换为计划内停机。

计划外停机的成本始终是最高的，因为它发生在您需要设备的时候。计划外停机永远不会在好时机时发生，并且可能由于生产中断、产品损失、对其他设备的附带损害、劳动力损失（劳动力可能必须坐着等待系统维修好），以及其他依情况而定的物流成本而造成巨额的利润损失。一些制造公司估计其计划外停机的成本约为每小时8,000美元。石化、电力和运输公司估计的每小时成本就更高了。这一成本因产品、情况、公司和行业是不同的。时间就是金钱；所以我们需要采用具有合适备件策略的纠正性维护计划来最小化故障系统的平均修复时间（MTTR）。

计划内停机的成本也很高，但比计划外停机要便宜，因为计划内停机对生产影响最小，可最大程度减小产量损失、对其他设备的附带损坏风险和物流成本，避免劳动力损失（因为经过培训的人员、工具和部件都在现场准备好进行维护）。计划内停机可能比计划外停机的时间短，并且可以分摊到需要维护的许多其他系统。由于计划外停机通常比计划内停机昂贵，许多公司已经实施了预测性和预防性维护计划。

$$CD = UD + PD$$

UD = 计划外停机的成本

PD = 计划内停机的成本

$$UD = \lambda \times MTTR \times T_U \times \text{每小时成本}$$

λ = *稳态故障率（每小时故障次数）*

系统的稳态故障率是指在系统的生命周期或使用寿命内预期的故障率。 该生命阶段介于使用初期（系统老化）和磨损阶段之间，到了磨损阶段，系统故障率预期将显着增加，系统是时候退休了。 以下数学关系适用于故障率达到稳定状态时的系统生命阶段。

$$\lambda = \frac{1}{MTBF_{系统}}$$

$MTBF_{系统}$ = *系统平均故障时间间隔（小时）*
T_U = *系统整个生命周期的总运行时间（小时）*

电子器件的运行时间通常包含系统在执行工作和处于空闲状态时的上电时间。

MTTR = 平均维修时间（小时）

MTTR 不仅仅是维修或更换故障组件的时间。 它还包括：

- 检测故障的时间
- 诊断系统以了解哪个/哪些组件出现故障的时间（在很大程度上受到有无备件和/或多余组件的影响）
- 靠近和维修或更换故障组件所需的时间（在很大程度上受到有无备件和/或多余组件的影响）
- 验证系统恢复正常工作的时间
- 系统重新联机的时间

显然，MTTR在很大程度上取决于备件可用性、系统位置、设计和通常发生的故障类型。

$$MTTR = \frac{\Sigma(\lambda_i t_i)}{\Sigma \lambda_i}$$

λ_i = *第i个故障模式的故障率*
t_i = *第i个故障模式的系统维修时间*
故障模式定义为发生的故障类型或故障的根本原因。

$PD = (\lambda \times MPdMT + f_{PM} \times MPMT) \times T_U \times$ *每小时的计划内停机时间*

预测性维护发生的频率与系统的故障率相关。预测性维护不是在发生故障之后执行维护，而是在检测到潜在故障条件之后但在故障发生之前的某个时间安排维护。

MPdMT = 平均预测维护时间（小时）

MPdMT包括：

- 访问时间
- 维护或更换组件的时间（在很大程度上受到有无备件和/或多余组件的影响）
- 验证系统是否正常运行的时间
- 系统重新联机的时间

$$MPdMT = \frac{\sum(\lambda_i t_i)}{\sum \lambda_i}$$

λ_i = *第i次预测性维护活动的频率*

t_i = *对系统进行第i次预测性维护活动的时间*

f_{PM} = *预测性维护的频率（每小时）*

MPMT = 平均预测性维护时间（小时）

MPMT包括：

- 访问时间
- 维护或更换组件的时间（在很大程度上受到有无备件和/或多余组件的影响）
- 验证系统是否正常运行的时间
- 系统重新联机的时间

$$MPMT = \frac{\sum(f_i t_i)}{\sum f_i}$$

f_i = *第i次预防性维护活动的频率*

t_i = *对系统进行第i次预防性维护活动的时间*

维护成本 (M)

$$M = PdM + PM + CM$$

PdM = 预测性维护的成本
PM = 预防性维护的成本
CM = 纠正性维护的成本

预测性维护的成本 (PdM)

$$PdM = \lambda \times T_U \times PdM事件 + 工具成本$$

PdM事件 = 一次PdM事件的平均成本

PdM事件= （MPdMT x 每小时计划内停机的劳动力成本）+ 维护或替换 + 备件 + 物流成本

计划内停机的劳动力成本包括对系统执行预测性或预防性维护的劳动力成本以及每小时估算的劳动力培训成本。

工具成本 = PdM所需的软硬件成本

工具成本通常为一次性支出，包括：

- 状态监测软件和硬件成本
- 拆卸和替换组件的工具成本
- 测试设备和软件验证成本（可能与纠正性维护一样）
- 设备和软件维护成本

注意：这些工具通常用于预测性、预防性和纠正性维护。如果可以使用工具，则工具成本只需要考虑一次，而不需要三种类型的维护都考虑一遍。

如上所示，MPdMT主要受到是否有合适的设备/工具以及操作人员的技能水平的影响。

预防性维护的成本 (PM)

$$PM = f_{PM} \times T_U \times PM事件 + 工具成本$$

f_{PM} = 预防性维护的频率（每小时）

PM事件 = 一次PM事件的平均成本

PM事件 =（MPMT x 每小时计划内停机的人力成本） + 校准、维护或更换 + 备件 + 物流成本

系统的MPMT越小，预测维护的成本就越低。 如上所示，MPMT主要受到是否有合适的设备/工具、操作人员技能水平以及校准策略的影响。 许多系统供应商提供了不同的校准选项。 根据情况，现场校准服务或向系统供应商购买校准服务协议可能更具成本效益。 标准的供应商校准项目一般就足够了。

纠正性维护成本 (CM)

$$CM = \lambda \times T_U \times CM事件 + 工具成本$$

CM事件 = 一次CM事件的平均成本

CM事件 =（MTTR x 每小时计划外停机的人工成本） + 维修或更换 + 备件 + 物流成本

计划外停机的人工成本包括系统维修的人工成本和按每小时估计的劳动力培训成本。

系统的MTTR越小，系统可用性就越大，计划外停机的成本也就越低。综上所述，MTTR受位置、系统设计、正确设备/工具可用性、人员技能水平和良好的备件策略的极大影响。许多系统供应商提供备件选项。 根据情况，现场备件或向系统供应商购买服务协议以提供备件或购买两者兼具的一些混合协议可能更具成本效益。 如果计划外停机的成本足够低，则可能就不需要现场备件，只需依赖标准供应商修理可能就足够了。