**Verasonics序列编程手册**

# 1简介

## 1.1 Verasonics 研究系统概述

Verasonics Vantage 研究系统由连接到主机控制器计算机的 Verasonics 数据采集硬件组成。采集硬件包含用于多通道发射波形生成、模拟接收信号放大和滤波、数字信号处理和扫描排序的电子模块。计算机包含 Matlab 编程环境中执行的软件模块，允许用户在硬件和软件环境中编程和运行动作序列（称为事件）。通过编写Matlab脚本（称为设置脚本）对事件序列进行编程，该脚本生成可加载到系统中的对象集合。序列中的对象是使用指定属性集的Matlab结构定义的。当安装脚本执行时，它会创建一个二进制数据文件，其中包含所有对象结构和编程信息，并以Matlab .mat文件格式存储。

这个.mat 文件可以通过加载程序（命名为“Verasonics Script eXecution”的“VSX”）加载到系统中，该程序在 Matlab 环境中作为典型脚本运行，并通过硬件抽象层 (HAL) 与硬件通信.加载程序对从 .mat 文件读取的结构执行一些检查，添加一些对 Vantage 硬件进行编程所需的属性，并打开 GUI 和显示窗口（如果指定）。然后将控制和处理交给一个用 C 编程语言编写的 Matlab 外部函数，名为“runAcq”。然后将事件序列加载到硬件和软件定序器中并开始执行。

在序列执行期间，几乎所有的处理和显示函数都由 runAcq 函数处理。此函数包含软件序列器并在运行时处理与硬件的所有交互。在事件序列结束时，或在跳转回序列开头后，或在序列中指定的中间事件时，runAcq 将控制权返回给 Matlab 环境中的 VSX 程序，以允许执行用户界面功能或用户程式。如果在 Matlab 环境中没有要执行的操作，VSX 会再次调用 runAcq，序列从头开始重复，或者从返回到 Matlab 的地方继续。

在执行事件序列时，可以使用与 VSX 和 runAcq 交互的 Matlab GUI 控件对采集和处理进行更改。时间增益控制滑块、发射功率、平移、缩放和冻结等基本控件会自动添加到 GUI 窗口中。用户可以添加可以在 Matlab 编程脚本中指定的其他 GUI 控件，以及在更改 GUI 控件时执行的回调函数。许多示例脚本有助于说明如何实现新的控制函数。

在序列事件的执行过程中，也可以让系统在序列的特定点调用用户提供的Matlab函数。用户功能可以访问获取的和处理过的数据，执行一些独特的功能，创建自己的数据显示，或将处理过的数据返回给处理软件。这提供了一种改变系统处理或提取信息进行脱机处理的简单方法。

当用户希望退出一个加载并运行的序列时，GUI窗口将被关闭，并在下一次runAcq返回Matlab时控制返回到Matlab环境。显示窗口未关闭，可用于测量、注释和归档。系统用于存储采集数据、重构像素幅值和复杂信号数据以及图像数据的缓冲区均可在Matlab环境中使用用户提供的Matlab脚本进行检查或进一步分析。

下面的部分描述了用于Verasonics系统编程的各种结构和对象。学习如何指定编写Sequence所需的所有结构的最好方法是学习其中一个示例脚本，例如' SetUpL11\_4vFlash.m '。通过分析示例脚本和学习对象是如何指定的，可以更好地理解编程对象，以及如何使用它们。

强烈建议新用户从学习序列编程教程开始，可以在文档文件夹中找到序列编程教程。本教程将带您逐步构建序列程序，并包含一些编程技术背后的原理的更多信息。本序列编程手册通常旨在提供关于系统和编程结构的更详细的参考信息，但其他章节介绍了硬件和软件的更具体的使用，如序列控制技术、信号平均和高频采集。

最好的开始方法是开发一个简单的序列程序，如教程中提供的，以熟悉系统操作的基础。然后可以开始检查和分析examplesscripts文件夹中的各种示例脚本，以熟悉更高级的编程技术。当您要开发一个新程序时，从一个接近您想要做的事情并修改它的现有示例脚本开始几乎总是更容易的。

有超声信号处理的经验和Matlab的基本编程技能是很有帮助的。如果你需要帮助在这些领域,有很多书可以参考,如“诊断超声成像:内外,”托马斯·l·萨博第二版,爱思唯尔学术出版社,和“Matlab:实用介绍编程和解决问题,“第3版(2016年7月发布的第四版)通过暴风雨Attaway,爱思唯尔。

## 1.2 Matlab设置脚本的结构

典型的Matlab Setup脚本包含以下元件，通常应该按照给定的顺序提供这些元件。

**Global System Objects (Non-Sequence Objects)全局系统对象(非序列对象)**

Script Info: Comments for script name, program information, software revision compatibility, etc.(脚本信息：脚本名称、程序信息、软件版本兼容性等的注释)。

Unique Script Parameters: Parameters for defining other structures, attributes to as presets.(唯一脚本参数:参数定义其他结构，属性保存为预设。)

Global Resource Parameters: Hardware resources, speed of sound, media attenuation, simulation mode, error reporting.(全局资源参数:硬件资源、声音速度、媒介衰减、仿真模式、错误报告。)

Transducer Specification: Trans structure definition.(换能器规格:反式结构定义。)

Pixel Data Specification (optional): PData and PData.Region structures (Only required if performing image reconstruction) (像素数据规格(可选):PData、PData。区域结构(仅当执行图像重建时需要))

Resource Buffers and Display Window (if needed): Resource.RcvBuffer,

Resource.InterBuffer, Resource.ImageBuffer, Resource.DisplayWindow.

Media Specification (optional): Media structure.

**Sequence Objects**

Transmit Specifications: TW and TX structure, TPC profile (optional).(发射规格:TW和TX结构，TPC配置(可选)。)

Time Gain Control Specifications: TGC structure.(时间增益控制规格:TGC结构。)

Receive Specifications: Receive and RcvProfile (optional) structures.接收规格:Receive和RcvProfile(可选)结构

Reconstruction Specifications (optional): Recon and ReconInfo structure, (needed only if performing image reconstruction).(重建规格（可选）：Recon和ReconInfo结构（仅在执行图像重建时需要）。)

Processing Specifications (optional): Process structure (needed if using internal or external processing functions).(处理规范(可选):处理结构(如果使用内部或外部处理功能，则需要)。)

Sequence Control Specifications: SeqControl structure.(序列控制规范:SeqControl结构。)

Sequence Event Specifications: Event structure. (Sequence事件规格:事件结构。)

GUI Control Specifications: UI structure.(GUI控制规范:界面结构。)

Save to .mat Specification: .mat file name for saving structures(保存到.mat规范:用于保存结构的.mat文件名)

GUI Callback Function Definitions: functions to encode with text2cell utility.(GUI回调函数定义:用于text2cell实用程序编码的函数。)

强烈建议用户在编程脚本中遵循这个顺序，因为错误检查和结构验证是在程序加载时执行的，其中一些函数依赖于结构定义的默认顺序。其基本原理是在后面的结构定义中引用对象之前定义对象，尽管这条规则没有严格执行。

不需要提供所有的脚本对象，通常只需要定义运行它们的序列所需的对象。例如，如果只想获取射频数据进行离线处理，则可以省略像素数据、显示窗口、重构和处理对象。同样，也不需要定义各种对象的所有属性，因为缺少的属性会自动用默认参数填充。如果缺少所需的属性，软件通常会退出并显示一条错误消息。

# 2 全局系统对象(非序列对象)

系统对象是那些定义独立于序列事件编程的对象，例如那些定义传感器和系统特征的对象。这些对象应该在Setup编程脚本的顶部指定，因为通常需要它们来定义Sequence中包含的其他对象的属性。这也提供了一种更自然的方式来生成用户安装脚本，允许在以前定义的结构上进行构建。

通常，定义的第一个系统对象属性是定义Setup脚本的特定特征所需的变量。例如，可能需要定义变量来指示超声采集的起始和结束深度，之后的结构将使用这些变量而不是硬编码的值。这允许通过改变一两个变量来轻松修改获取范围。如果以这种方式定义的变量应该保存作为预设的一部分(一组参数,可以保存到一个二进制文件中供以后恢复程序的状态),变量可以是前缀“P。”,这使得它预设结构的一部分,P,保存到一个文件时创建一个预置。

接下来，通常定义Resource对象的Parameter属性。Resource对象用于指定执行用户序列所需的硬件和软件资源。资源。参数属性确定将被用户脚本和/或系统软件使用的全局因素。

在下面的定义中，突出显示了需要指定的属性。通常，如果这些属性缺失或为空，VSX将生成一条错误消息。以纯文本列出的其他属性，例如“speeddofsound”，如果没有显式指定，将假定为默认值。

## 2.1Resource.Parameters Attributes（参数属性）

Resource =

Parameters =

Connector (3.2+) double connector # (1(dflt), 2, or [n,m,...])

connector (3.0x) double (deprecated) cnctr # (0(both),1(dflt),2)

numTransmit double number of available transmitters

numRcvChannels double number of available receive channels

speedOfSound double speed of sound in meters/sec (deflt 1540)

attenuation double media attenuation in -dB (deflt=0dB)

speedCorrectionFactor double corrects speed of sound (default=1.0)

startEvent double event no. to start after freeze (dflt=1)

simulateMode double 0='use VDAS'(default), 1='simulate'

initializeOnly double 0 = false (default), 1 = ' true'

fakeScanhead double (dflt=0) if 1, allow to run without probe

UpdateFunction string VsUpdate' [default] or user specified

GUI string 'vsx\_gui' [default] or'<user specified>'

verbose double 0:3 specifying level of warnings/errors

Resource.Parameters.Connector –这个属性是在版本3.2中引入的，它取代了以前版本中使用的连接器属性(小写的' c ')。该值是可选的，如果没有提供，默认值为1。只有具有多个连接器的通用传感器适配器模块才需要它，例如包含双连接器传感器适配器模块 (UTA 260-D) 的 256 T/R Vantage 系统。对于这个UTA，值1和2分别选择左或右连接器，任一连接器都可以容纳一个128单元(128发射/128接收)换能器。[1,2]的值可以与连接到两个连接器的自定义传感器一起使用，它可以支持多达256个元件(256发射/256接收)。对于具有多个连接器的uta, Trans的值。Connector是要激活的连接器编号的数组。3.2之前的软件版本应该使用connector属性，它只提供**三个选项—0(两个连接器)、1(左连接器)或2(右连接器)。**

Resource.Parameters.numTransmit/numRcvChannels –这些值指定通过所选连接器在硬件中可用的发射和接收通道的数量。这些需要提供，并在使用Vantage硬件运行Setup脚本时进行检查。如果实际的硬件资源与那些指定的不匹配，序列程序可能无法执行，或者执行时发出警告消息。

Resource.Parameters.speedOfSound –此值指定要被放大的媒介中的声速。如果没有提供，则默认值为1540 m/s。由于序列对象的许多长度参数是在波长中指定的，因此，如果声速与默认值不同，那么在脚本的顶部指定声速是很重要的。

Resource.Parameters.attenuation(衰减) – 这是以dB/cm/ MHz为单位的介质衰减。典型值为-0.5 dB/cm/MHz，默认值为0db。如果没有设置任何值，但**有一个Media提供了衰减值**，该值也用于Resource.Parameters.attenuation)。仿真软件和图像重建软件都使用衰减值来补偿发射时的衰减损失。接收时的衰减损失由时间增益控制曲线(TGC)补偿。

Resource.Parameters.speedCorrectionFactor – 此值用于微调包含略有变化的声速的媒介的声速。它通常只在序列运行时进行调整，最初使用默认值1.0。该参数只修改图像重建中使用的声速，不修改其他长度参数。

Resource.Parameters.startEvent – 此值指定序列中的起始事件。如果没有提供，则默认的开始事件将是事件编号1。startEvent用于序列的初始运行，以及退出“冻结”状态时。startEvent可以在运行时使用GUI控件和control动态更改。runAcq处理的命令输入(参见4.0节中的方法)。这允许在运行时选择不同的序列事件段执行。

Resource.Parameters.simulateMode – 此参数用于打开模拟模式，在此模式中，序列由软件运行，而不是由硬件运行。如果VSX加载程序没有检测到硬件，则自动进入此模式。设置该值为1将模拟采集，并使用Media模型来描述模拟的媒介。对来自每个脉冲发生器的指定TX脉冲进行模拟，并将来自媒介目标的结果返回积累在ReceiveBuffer中。将该值设置为2将关闭媒介模拟，以便可以连续处理之前获取或加载的ReceiveBuffer。对于带有许多帧的ReceiveBuffers，模式2有效地提供了捕获的RF cinloop的回放。

Resource.Parameters.initializeOnly – 当设置此属性为1时，将导致软件在初始化后退出。初始化会添加所有缺失的默认属性并验证结构。还创建了其他隐藏结构，比如用于指定向主机内存发射数据的dmaccontrol。这对于调试在运行时使Matlab崩溃的脚本非常有用。初始化之后，可以检查添加的属性和正确的排序的所有结构。

Resource.Parameters.fakeScanhead – 如果希望在系统中没有插入扫描头时使用Verasonics硬件运行脚本，则将此可选参数(默认为0)设置为1。当想要探测传感器连接器的I/O引脚上的信号时，这个选项是有用的。

Resource.Parameters.updateFunction – 这个可选参数(默认是' VsUpdate ')可以指定来调用一个不同于系统默认值的更新函数。更新函数用于在初始化和运行时更改期间更新结构并向其添加各种属性。只有高级用户才应该尝试指定不同的更新功能。

Resource.Parameters.GUI – 可以指定此可选参数(默认为' vsx\_gui ')来调用与系统默认值不同的图形用户界面函数。GUI功能用于绘制GUI窗口和默认系统控件。它还为用户提供了创建自己的控件的方法。不同GUI功能的规范可以用于为特定的应用程序创建自定义用户界面，但这项任务很有挑战性，应该只由高级用户尝试。

Resource.Parameters.verbose – verbose属性设置VSX和其他函数在Matlab命令行上报告错误、警告和状态消息的级别。支持的值是:

! 0: 只显示错误信息。

! 1: 显示错误和警告提示。

! 2:显示错误、警告和状态信息。(默认)

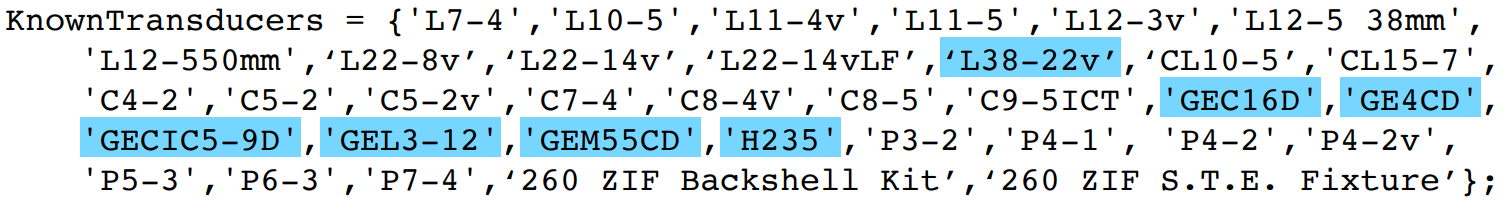
! 3. 显示错误、警告、状态和调试状态消息。

## 2.2 Transducer Object

下一个通常在新脚本中定义的系统对象是Trans对象，它描述将使用的换能器的特征。传感器的类型(线性、曲线线性、相控阵等)通常决定了扫描区域的形状，并在一定程度上决定了扫描的方法。Trans结构的距离单位可以指定为中心频率的波长(由Trans.frequency提供)或毫米单位，通过Trans.units参数选择。Resource.Parameters.speedOfSound属性中给出的声音速度决定了从mm到波长的转换。例如，一个5MHz中心频率换能器的波长为0.308毫米，声速为1540米/秒。

换能器特性可以根据已知的换能器类型自动设置，使用实用函数` computeTrans.m `。首先指定名称，然后调用实用函数。如果希望使用非默认的中心频率，请在调用computeTrans.m之前指定它。(只有某些离散的中心频率是可用的，因为前端的模数转换器必须设置为中心频率的4倍采样，采样频率来自一个250MHz的主时钟。有关支持的中心频率列表，请参见3.3.1.1节。)最后，如果要成像的媒介中的声速与默认的1540m/s不同，还请确保在调用computeTrans之前将Resource.Parameters.speedOfSound设置为正确的值。

目前已知传感器名称的名称列表包括以下Verasonics传感器、GE传感器以及飞利浦HDI-1000、3000和5000兼容探头:



突出显示（已知，Verasonics自带）的换能器仅在软件版本3.2及更高版本中可用。在可选地指定一个修改过的中心频率或声速后，可以如下所示调用computeTrans.m效用函数来填充Trans结构的其他属性。

Trans = computeTrans(Trans);

对于未知的传感器，用户必须生成自己的反式结构，并应至少提供下面反式结构中突出显示的属性。

Trans =

name string % Scanhd name (‘L7-4’, ‘C4-2’, ‘P4-2’, etc.)

id double % id read from scanhead personality eprom

frequency double % center frequency in megahertz

type double % 0=Lin(y=z=0), 1=CurvedLin(y=0), 2=2D(z=0)

units string % ‘mm’ or ‘wavelengths’

Numelementsdouble % number of transducer elements

elementWidthdouble % width in mm or wavelengths (spacing–kerf)

spacingMm double % element spacing in mm

spacing double % element spacing in wavelengths

radiusMm double % curved array radius in mm (n/a for lin)

radius double % curved array radius in wls (n/a for lin)

ElementPos [numelex5 dble]% [position(wvlngths),az,el(radians)]

ElementSens [101x1 double]% sens. curve for sngl ele –pi/2to+pi/2

lensCorrectiondouble % 1 way delay in wvlngths thru lens (dflt=0)

maxHighVoltagedouble% max. high voltage limit (optional).

Bandwidth [1x2 double]% -6db lower and upper cutoff pts in MHz

impedance complex % (optional) complex impedance at set freq

IR1wy [nx1 double]% (opt) transducer 1 way impulse response

IR2wy [nx1 double]% (opt) transducer 2 way impulse response

HVMux structure% needed only for HV mux probes (see below)

Connector [numele,1 double]% (optional) defines connector channel

% to element map for non-multiplexed probes.

connType double % specifies type of UTA connector (see text)

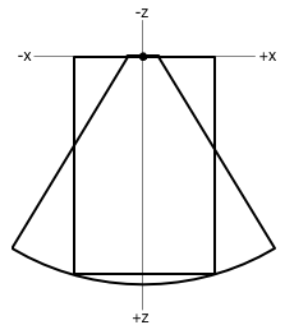
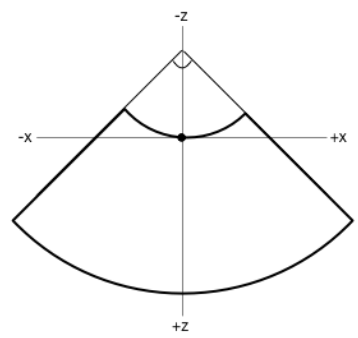
elementWidth, ElementPos, and lensCorrection 属性的单位由Trans.units指定。对于某些属性，如Trans.spacing(Mm)，波长和Mm单位总是指定的。当单位设置为“波长”时，使用波长来指定换能器的尺寸可能会在成像介质的不同声速时有些混乱，因为显然探头的尺寸必须保持固定。speedcorrectionfactor可用于改变介质中使用的波长值，但不能用于探针。**根据探头的中心频率和声速来定义探头的尺寸，例如1540米/秒。如果媒介中声音的媒介速度恰好比标称速度慢，例如1500米/秒，那么设置speedCorrectionFactor为1540/1500 = 1.0267**(到媒介目标的时间比这个因子长)。然后，介质中的波长计算将被这个相同的因素修改(在相同频率下较慢的速度会导致较短的波长)，但探针的尺寸将保持不变。

Trans.type 属性需要指定换能器的几何形状。线性数组是类型0，其中元件位置的中心只能用x坐标指定(y = z = 0)。曲线阵列，元件的中心必须用x、z坐标(y = 0)和方位角(从法线到正z轴的角度)来指定，这是一种类型。对于曲线阵列，原点在中心位置的元件表面，曲率在x,z平面。可以用x、y、z坐标指定元件中心的二维数组是类型2。对于2D数组，原点通常定义在2D数组的中心，元件位于x、y平面。然而，一些阵列是非平面的（碗阵列），并且可以在任意 3D 位置定义元件，尽管一些位置是创建重建区域的首选。(不同扫描格式下原点位置及坐标轴如图2.2.2所示)

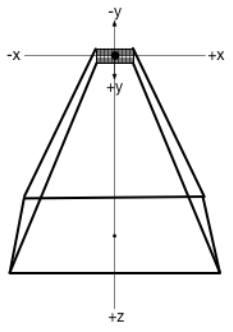
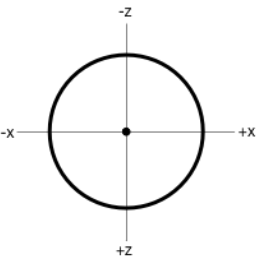


图2.2.1线阵单元位置坐标系统

Trans.numelements属性简单地指定换能器中元件的总数。这个数字，连同Trans.spacingMm(毫米)或Trans.spacing(用于波长)设置了线性阵列传感器的总孔径。Trans.elementWidth属性定义了一个换能器元件的实际宽度，它通常比切割分离元件的切口间距小。元件宽度用于元件的方向灵敏度函数的默认计算。

线阵和相控阵元件只有x个值 曲线线性阵列元件有x,z，方位角值(方位角是元件垂直于z正轴的角度)

矩形矩阵数组元件具有值 环形阵列（向内或向外）元件具有方位角值

对于非平面阵列，原点通

常位于中心元件，元件具

有方位角、仰角值

图2.2.2各种阵列类型的原点和坐标系

换能器中每个元件的位置和方向在Trans.ElementPos数组中定义。Trans.ElementPos数组有一个与元件编号对应的行索引，最多5个列值指定元件中心的x、y和z坐标(以毫米或波长为单位)，以及两个角度参数，方位角和仰角。方位角是元件法线与正z轴之间的角，在指定曲面阵列的元件位置时需要它。仰角是元件垂直于x、z平面的角度，仅用于指定特定2D数组的元件位置。如果不需要，则不需要指定方位角和仰角，如果所有元件为零，则不需要指定。参见下面的图2.2.3确定一个元件的方位角和仰角。

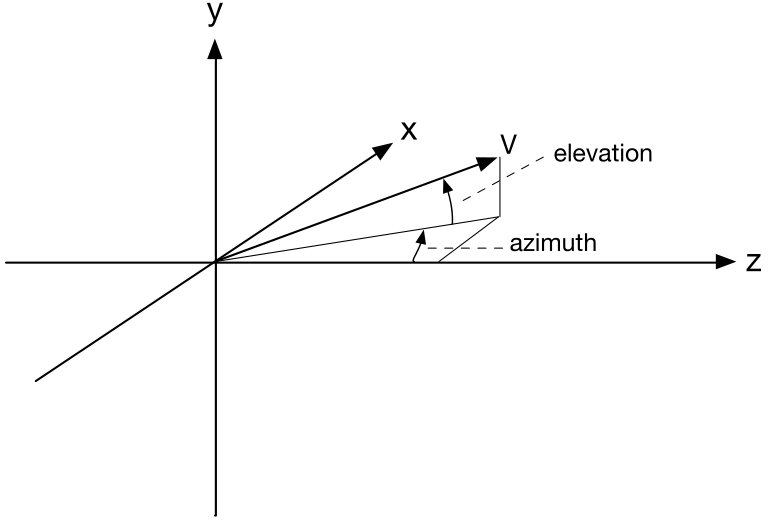


图2.2.3方位角和仰角。这些角度被指定为x,y,z轴的原点在元件中心的位置

单个换能器元件的指向性模式在Trans.ElementSens数组中指定。这个数组提供了元件对从法线到元件各个角度的目标的相对灵敏度。Trans.ElementSens数组包含101个从-pi/2到+pi/2等间隔角度的灵敏度值。角度0处的值归一化为1.0。Trans.ElementSens数组的默认设置使用Matlab公式

X = Trans.ElementWidth \* pi \* sin(Theta) ;

Trans.ElementSens=abs(cos(Theta) .\* (sin(X) ./ X)) ;

其中

Theta = (-pi/2:pi/100:pi/2);

在0处的值，给定一个小的值，以避免除以0。该公式适用于杆型振动的线性阵列单元。对于其他元件几何形状，只要值的是数量保持在 101，就可以计算或凭经验输入 Trans.ElementSens 值。对于 2D 阵列，假定元件的灵敏度函数关于元件法线轴对称。 灵敏度值的当前使用是在图像重建期间，如果元件对像素位置的灵敏度低于指定阈值，则可以消除元件的贡献。

在图像重建过程中使用Trans.lensCorrection参数来补偿通过换能器镜头的旅行时间。透镜通常是由一种比介质的声速更慢的材料制成的，因此它们可以假定有一个凸曲率。Trans.lensCorrection参数应该设置为波长的单向值，透镜添加到声学路径，将采取没有透镜。例如，假设透镜中的声速是介质中声速的3/4，透镜的平均厚度大约是12个波长(12个波长代表基于介质中声速的波长)。在这种情况下，透镜校正参数将是(12 \* 4/3)- 12，或4波长，这是声音通过透镜单向传播的有效路径长度的增加。此属性的默认值为零。

可以选择使用属性Trans.maxHighVoltage指定该换能器的变送器的高电压限制，最高到computeTrans实用函数中设置的限制。峰峰发射电压将是这个值的两倍，因为发射机有一个双极性驱动波形。指定一个限制是一个好主意，因为默认设置的最高高压限制(通常为50伏特)可能允许过多的声功率传递到换能器，导致探头和介质过度加热。对于较新的Vantage系统，最高高压限制可以设置高于默认的50伏特，最高到Transmit Power Controller（发射功率控制器）的硬件限制，目前是96伏特;但是，如果换能器在 computeTrans 中设置了功率限制以防止损坏换能器，则将强制执行此限制。(注意:最高硬件高压限制能够破坏一些传感器和/或在生物组织中产生有害的声功率水平。为防止损坏，请勿在computeTrans中修改高压限值。在扫描时，只使用特定应用所需的高电压水平。)

Trans.Bandwidth属性替换旧的Trans.bandwidth属性，并为传感器的低和高-6dB响应点(往返响应)提供MHz的值。此属性是可选的，默认为Trans.frequency值的60%。此属性目前用于设置模拟前端Analog Front End (AFE)放大器的抗混叠滤波器截止值-默认的抗混叠截止值将是高于Trans.Bandwidth(2)的最低可用滤波器截止值。带宽值也用于模拟传感器的发射波形从脉冲参数指定。当Verasonics系统在模拟模式下工作时，使用这个模拟脉冲波形。

Trans.impedance属性是一个可选规格(默认值为20欧姆)，由Verasonics系统的软件使用，使用可以分析用户序列脚本的效用函数来估计各种发射序列的高压功率需求。可以以不同的方式为未知的探测指定该值。如果中心频率只知道实分量，则该值可以用欧姆表示。如果中心频率处的复阻抗已知，则可以将其指定为单个复数值，例如。 31.2-2.05i。如果已知不同频率下的阻抗，则Trans.impedance可以指定为一个N × 2的复数数组，其中第一列指定频率兆赫，第二列是这些频率的复数阻抗欧姆，例如 [4.5 32.8-72.6i; 4.75 39.2-66.2i; 5 46.1-69.6i; 5.25 46.5-72.4i; 5.5 41.9-71.6i]。任何频率的数量(值N)可以指定在任何频率间隔，但它们必须以递增的顺序列出。对于 HIFU 应用程序，准确指定 Trans.impedance 对于从 TXEventCheck 获得适当的发射限制至关重要，因为默认值非常低。

Trans.Connector是一个可选属性，可以指定连接器I/O通道分配到传感器中的元件。(关于传感器连接器中连接器引脚的连接器I/O通道的分配，请参见下面的2.1.2节。) Trans.Connector数组的大小是一列由Trans.Numelements行，行索引对应于元件号。给定行索引的值就是该元件所连接的连接器I/O通道。默认的Trans.Connector分配假设一个128元件的换能器，与元件分配到I/O通道1对1的基础上，即元件1连接到IO 1等(元件编号可以任意分配到换能器元件，但必须与Trans.Connector数组的索引对应。指定它们的物理位置。)对于定制探测，可能需要为IO通道分配不同的元件号，在本例中是Trans。可以使用连接器阵列来设置映射。例如，对于一个128元件的探针，其中元件以相反的顺序连接到I/O通道，这个Trans.Connector数组将在Matlab中被定义为Trans.Connector = (128:-1:1)’。(请注意最后的标记转换为列数组。)对于在256通道Vantage系统中使用这两个连接器的定制探测，第二个连接器上的连接器I/O通道编号从129到256。

如果换能器(非多路复用)使用的连接器I/O通道比可用的少，那么Trans.Connector数组应该被定义，指定要使用的通道。例如，连接到连接器I/O通道1到64的64元件探头应设置为Trans.Connector = (1:64)’。Trans.Connector数组只能定义为非多路转换器(无Trans.HVMux 结构存在)，因为它的功能与下面描述的Trans.HVMux.Aperture 数组类似。

在早期的SW版本中，系统只支持一种探测连接器，因此不需要在用户脚本中识别它。较新的Vantage系统利用用户可移动传感器连接器模块，称为UTA(通用传感器适配器)。对于不同的UTA模块，可以使用其他连接器选项，它们与标准Cannon连接器具有不同的元件映射。从2.11软件版本开始，可以在Tools/ElementToChannelMapping目录中使用一些实用工具来显示不同连接器的元件和通道映射。

新字段 Trans.connType 标识用户脚本所需的连接器类型。 从 2.11 系统软件开始需要 Trans.connType； 它是一个整数值，用作标识通用传感器适配器系统支持的一种连接器类型的索引。 可识别以下连接器类型索引值：

█ Trans.connType = 1（如果未提供默认值）标识 HDI 格式连接器，与在 UTA Vantage 之前的系统上使用的探头接口相同。 为了向后兼容使用早期 SW 版本开发的脚本，如果用户设置脚本中未定义 Trans.connType（但仅当 Resource.Parameters.numTransmit 的值符合 HDI 连接器，即 128 或 256，=Resource.Parameters.connector = 0)。

█ Trans.connType = 2 标识接口来自“分线板”UTA 适配器（提供对元件信号的直接访问，根本没有特定的探头连接器）或用户构建的定制UTA 适配器。

█ Trans.connType = 3 标识 MS 系列连接器接口和引脚（Cannon DL-360 连接器，具有 256 个元件信号）。

█ Trans.connType = 4 指定 Verasonics UTA-408 连接器接口。

█ Trans.connType = 6 指定带有 LEMO 连接器的“NDT”Hypertronix UTA-160-DH/32 接口（仅适用于软件版本 3.2 或更高版本）。

█ Trans.connType = 7 指定 GE D 系列适配器 UTA-408-GE（仅适用于软件版本 3.2 或更高版本）。

还有两个“特殊”的 Trans.connType 值可以在用户的SetUp 脚本中定义：

█ Trans.connType = 0 标识完全不打算在硬件系统上使用的仅模拟脚本，因此无需标识探头连接器接口。当脚本中存在此值时，将不应用任何系统硬件兼容性约束（例如，发送和接收通道的数量可以设置为 1 到 1024 范围内的任何所需值）。将 Trans.connType 设置为 0 与将 Resource.Parameters.simulateMode 设置为 1 具有相同的效果（如果用户脚本已将 Trans.connType 设置为 0 并且 Resource.Parameters.simulateMode 未定义，VSX 将为模拟模式设置默认值 1 ）。在使用纯模拟脚本时，这个新字段还提供了额外的灵活性：通过将模拟模式设置为 2，它将允许使用 RF cineloop 回放预定义接收数据缓冲区的内容，用于纯模拟脚本。（在早期的 Vantage SW 版本中，将模拟模式设置为 2 会导致系统初始化与硬件系统一起使用的脚本，因此如果仅模拟脚本与硬件系统约束不兼容，则会阻止其运行。）VSX 将分配一个默认的 Trans.connType对于尚未定义但已将 Resource.Parameters.numTransmit 设置为与 HDI 连接器不兼容的值的任何脚本，值为零；结果将是脚本将仅在模拟模式下运行。要将 UTA 适配器与原始 HDI 以外的任何连接器类型一起使用，SetUp 脚本必须显式设置 connType 值。

█ Trans.connType = -1 用于旨在与多个UTA 适配器兼容的测试脚本。当 VSX 使用此值初始化脚本以在硬件系统上使用时，-1 将替换为索引值，该索引值表示连接到系统的 UTA 适配器上实际存在的连接器类型。这允许测试脚本或其他实用程序以“通用”格式编写，当它们运行时将自动适应实际的系统硬件配置。

**2.2.1 Connecting a custom transducer(自定义传感器) to the system**

探头元件与系统发射器和接收器的连接是自动指定的“已知”传感器，这主要是Verasonics提供的传感器和ATL/Philips HDI1000-3000-5000商用探头，在二手超声设备市场上可用。对于连接一个定制的探头，必须购买正确的传感器匹配连接器，这对于佳能ZIF连接器是第1部分。dl5 - 260 pw6a。然后探头元件必须连接到连接器上适当的IO引脚，其名称和引脚名称见下面T able 2.2.1.1。对于带有Cannon连接器UTA的Vantage 64通道系统，IO通道与发射器和接收器的连接如图2.2.1.1所示。Vantage LE 64通道和Vantage 128通道系统(带有一个佳能UTA连接器)的连接分别如图2.2.1.2和2.2.1.3所示。

对于带有Cannon连接器UTA的256通道Vantage系统，有256个发射和256个接收通道，连接到两个扫描头连接器，如图2.2.1.4所示。一个128个元件的换能器可以连接到任何一个连接器上，并且SetUp脚本可以将系统视为拥有128个发射和128个接收通道，并在Resource.Parameters.connector属性中指定使用哪个连接器。加载程序VSX将决定使用哪一组128个发射器和接收器来选择合适的连接器，并且可以指定发送和接收通道，就好像只有一个可用的连接器一样。通过在SetUp脚本中指定连接器编号0，最多256个元件的自定义探针可以同时使用两个连接器。在这种情况下，可以使用所有发送和接收通道，第二个连接器将发送和接收通道计数从129扩展到256。

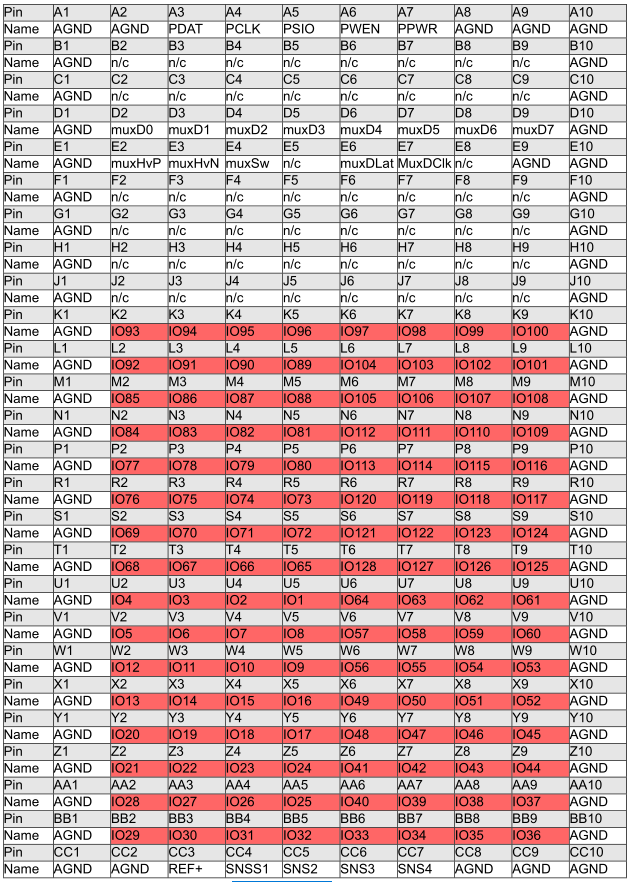
没有探测ID EPROM的自定义探测应该将Trans.name属性设置为' custom '。这将导致VSX跳过探针ID检查连接的传感器，从而允许脚本运行。Vantage系统监视扫描头是否连接，为了使这种监视正常工作，定制探针连接器上的引脚A1必须接地。

表2.2.1.1、传感器连接器I/O通道到引脚编号的映射

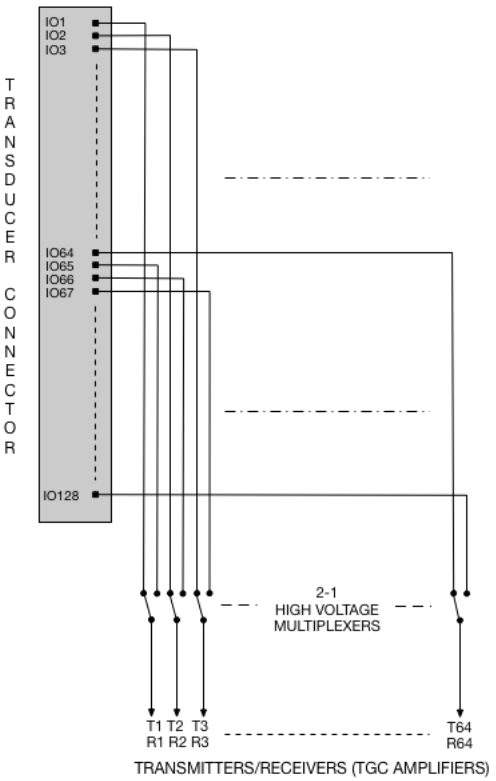


图2.2.1.1、64通道Vantage系统前端连接

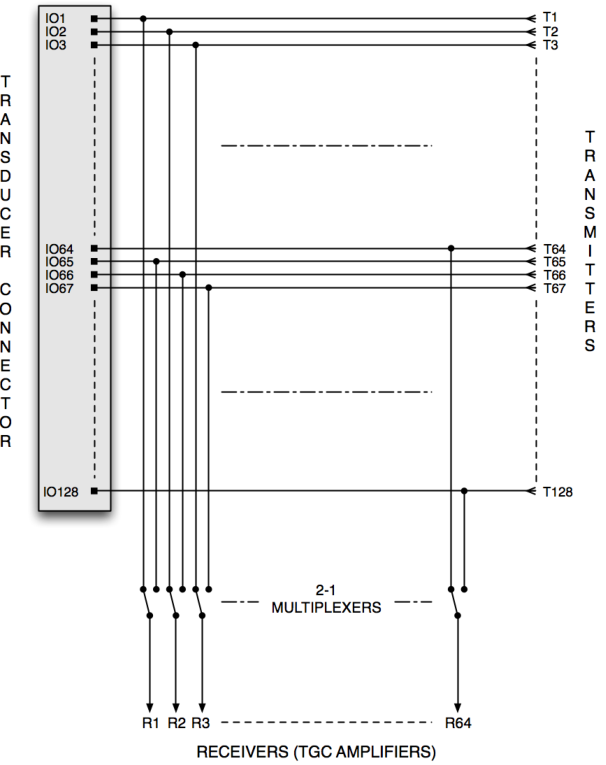


图2.2.1.2、64通道Vantage LE系统前端连接

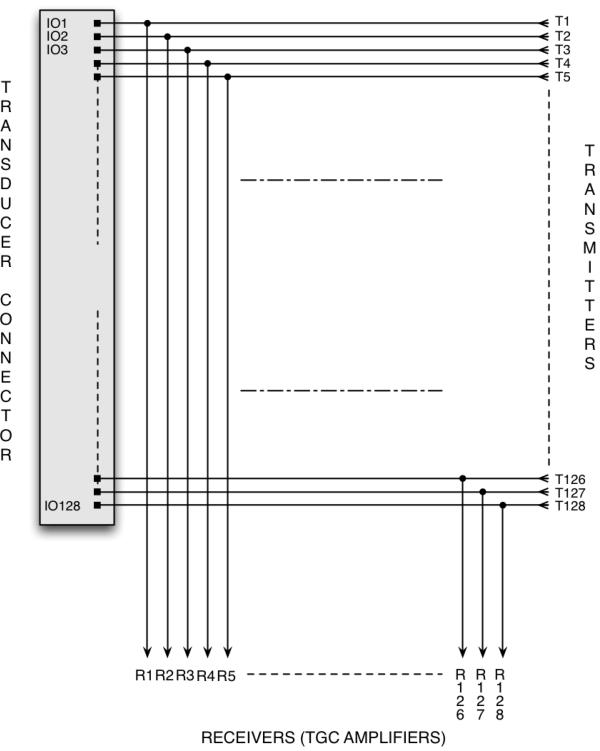


图2.2.1.3、128通道Vantage系统前端连接

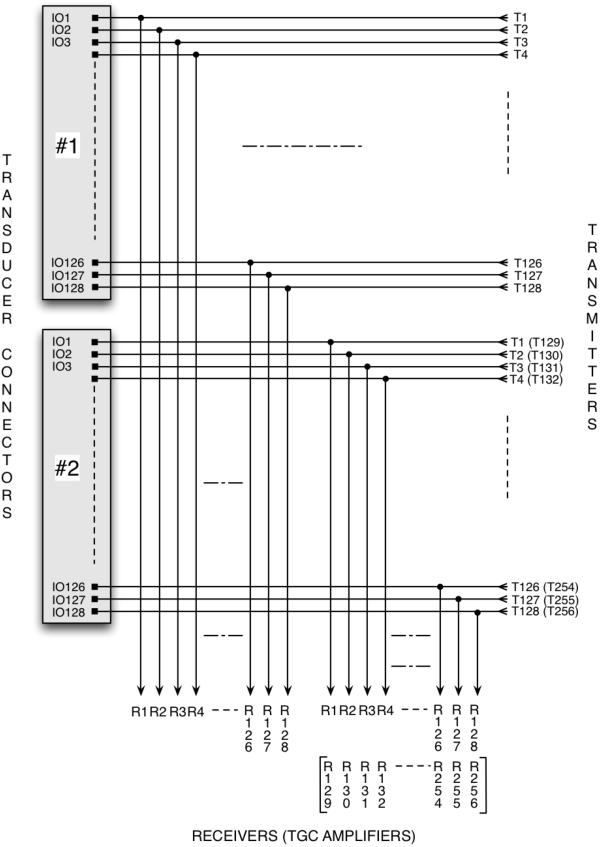


图2.2.1.4 256通道Vantage系统前端连接

**2.2.2高压多路转换器**

Vantage系统发射功率控制器为探头中带有高压多路复用器的传感器提供硬件和固件支持。这种类型的探头需要多个电源级别的多路复用器组件和数字逻辑来设置开关模式。可以支持比可用I/O通道更大的元件数的传感器，如192元件L12-3的38mm孔径，192元件L12-5的38mm孔径，和256元件L12-5的50mm孔径。对多路复用探测的支持包括以下功能:

1. 扫描头连接检测-该系统可以检测何时扫描头连接到双板系统上的单个传感器连接器，或连接到四板系统上的两个连接器中的任何一个。扫描头的存在是通过检测ZIF连接器上的引脚A1被接地来确定的。
2. 扫描头人格读取-该系统提供从Verasonics传感器和大多数飞利浦HDI-5000兼容传感器探头连接器中发现的EPROM读取扫描头人格信息的方法。这允许识别需要HV多路复用支持的扫描头。
3. 高压多路复用器特性的编程-根据探头中高压多路复用器的类型和版本，电源级别和其他接口要求可能会有所不同。computeTrans.M函数为“已知的”传感器提供了正确的正/负高压电源和传感器的逻辑供电轨的规格，以及将比特模式转换到多路复用器的参数。
4. HV多路复用位模式的序列加载—多路复用探针的理想孔径选择可以在分配给采集事件的发送和接收对象中指定。然后，通过在采集事件开始之前将相应的开关模式加载到多路复用器中来选择孔径。
5. 扫描头断开检测-当序列运行时，系统有能力检测扫描头断开，以便安全下电高压轨道。

在当前实现中，施加了以下条件:

1. 仅支持“已知”多路复用转换器，这意味着在 computeTrans.m 实用程序函数中提供了 HV 多路复用器属性的定义。

2. HV 多路复用器孔径选择仅限于由 computeTrans.m 函数和传感器孔径数据文件定义的选择。这些当前是将 128 个连接器 IO 通道放置在整个传感器孔径上的连续一个元件增量位置的设置。换言之，可以从换能器元件阵列中选择任何连续的128个元件孔径。

3. 在 3.2 软件版本之前，发送孔径和接收孔径必须使用相同的 HV mux 设置。在 3.2 版本中，发射和接收的孔径可以不同。在最后一个发射通道发射后，一个新的孔径模式被转移到 HV 多路复用器电路中。更改多路复用器孔径需要几微秒才能在新模式中移动，并且在移动完成之前不会有有效的接收数据。

4. 用于设置高压多路复用器孔径的位模式的每个编程设置所有可用的多路复用器开关;换句话说，不支持通过一次切换几个元件的数组来转换模式。

在用户脚本中指定多路转换器名称时，computeTrans.m 函数将以下结构添加到 Trans 对象。

Trans.HVMux =

highVoltageRails double % positive and negative high voltage rails

logicRail double % logic supply rail

clock double % shift clock rate in MHz

clockInvert double % 0 (default) = false, 1 = true

polarity double % polarity of shift bit for on setting (1 or 0)

latchInvert double % 0 (default) = false, 1 = true

Aperture(nele,n) double % trans aperture n to con. chnl mapping

VDASAperture(65,n)[uint8] % bit patterns corresponding to Aperture n

Trans.HVMux 的前六个属性指定了扫描头中高压多路复用器电路的电源电平和工作特性。 These parameters will be programmed into the Transmit Power Controller (TPC) and (Scanhead Interface) SHI hardware modules when the scanhead is selected.

Trans.HVMux.Aperture 阵列指定换能器元件编号到特定孔径的连接器通道的映射。第一个索引是元件编号索引。元件编号索引对应于 Trans.ElementPos 数组中定义的元件编号索引。与元件编号索引关联的值是元件连接到特定 HVMux 设置的连接器通道。如果没有连接器通道连接到元件，则该值设置为 0。最后一个索引用于指定为换能器定义的多个孔径。对于线性阵列，这些孔将允许将 128 个连接器通道在阵列表面上一次移动一个元件。

例如，考虑 L12-5 38mm 扫描头，它的前 64 个连接器通道连接多路复用到元件 1 到 64 或元件 129 到 192。要将孔径定义为 192 元件阵列中的前 128 个元件， Trans.HVMux.Aperture 数组将定义为：

Trans.HVMux.Aperture(1:192,1) = [1:128,zeros(1,64)];

在这种情况下，前64个连接器通道连接到前64个换能器元件，最后64个连接器通道连接到元件65-128。如果将孔径定义为192个单元阵列中最后128个单元，则Trans.HVMux.Aperture阵列定义为:

Trans.HVMux.Aperture(1:192,65) = [zeros(1,64),65:128,1:64];

在这种情况下，前64个通道连接连接到192元件数组中的最后64个元件，而最后64个通道连接连接到中间的64个元件。

L12-3 38mm 扫描头等已知传感器将具有由 computeTrans.m 定义的孔径阵列。 这些定义的孔将允许将 128 个连接器通道一次移动一个元件穿过换能器的表面。 对于 L12-3 38mm 的 192 个元件，将有 65 个 Trans.HVMux.Aperture 阵列。 为了说明映射，前 10 个 Aperture 阵列如下所示。 （上面给出了最后一个编号65的孔径。）

Trans.HVMux.Aperture(:,1) = [1:128,zeros(1,64)];

Trans.HVMux.Aperture(:,2) = [0,2:128,1];

Trans.HVMux.Aperture(:,3) = [0,0,3:128,1:2];

Trans.HVMux.Aperture(:,4) = [zeros(1,3),4:128,1:3];

Trans.HVMux.Aperture(:,5) = [zeros(1,4),5:128,1:4];

Trans.HVMux.Aperture(:,6) = [zeros(1,5),6:128,1:5];

Trans.HVMux.Aperture(:,7) = [zeros(1,6),7:128,1:6];

Trans.HVMux.Aperture(:,8) = [zeros(1,7),8:128,1:7];

Trans.HVMux.Aperture(:,9) = [zeros(1,8),9:128,1:8];

Trans.HVMux.Aperture(:,10) = [zeros(1,9),10:128,1:9];

要指定孔径阵列从192个元件阵列中选择中心的128个元件，我们有以下定义:

Trans.HVMux.Aperture(:,33) = [zeros(1,32),33:128,1:32,zeros(1,32)];

从用户的角度来看，选择与多路转换器一起使用的孔径只涉及指定 Trans.HVMux.Aperture 阵列的第二个索引的值，它通常对应于所选孔径中的第一个元件（对于线性阵列）。在用户脚本中，用于发送/接收事件的孔径由 TX 和 Receive 对象中的以下结构属性指定：

TX.aperture = n;

R e c e i v e . a p e r t u r e = n ;

对于线性阵列，n 的值将等于大小等于连接器通道数（通常为 128）的连续孔径开头的第一个元件（从 1 开始计数）的数量。例如，对于L12-3 38mm 线性阵列，n = 33 的值将指定从元件 33 开始并在元件 160 结束的孔径。此孔径选择 192 元件阵列的中心 128 个元件。

在 3.2 版软件中，允许在发送和接收中使用不同的孔径，因此可以为具有发送和接收规范的事件指定不同的 Receive.aperture 编号而不是 TX.aperture 编号。在这种情况下，系统将在事件开始之前对发射孔径进行编程，然后以编程的延迟执行发射。在最后一个发射器完成后，HV 多路复用器电路将被编程为 Receive.aperture 数字，并且此重新编程可能需要几微秒才能完成。在此转换期间，接收数据采样正在进行，因为采样是在事件开始时开始的。这意味着在重新编程 HV 多路复用器电路之前，不会有有效的接收数据。通常，通过换能器透镜的传播时间延迟足以允许从透镜表面开始采集有效的接收数据。如果属性 Receive.aperture 与 TX.aperture 相同，则在 Receive 结构中仍然需要它，因为需要光圈来确定如何将 Receive.Apod 值置换到各个接收通道。

用户设置脚本不应修改由 computeTrans.m 函数为多路转换器设置的 Trans.HVMux 属性和 Trans.maxHighVoltage 限制。 使用不同的电源设置和/或更高的发射电压可能会对换能器中的多路复用器芯片造成永久性损坏。

## 2.3 Scan Format Object (deprecated)(扫描格式对象（已弃用）)

在 3.x 软件版本中，不推荐使用扫描格式对象，不建议在新的设置脚本中使用。 它的支持可能会在以后的版本中被删除。 关于换能器的扫描区域格式的信息现在包含在 PData 结构的 PData.Region 属性中。 如果在 3.x 设置脚本中提供了 SFormat，则它用于创建相应的 PData.Region 结构（参见第 2.4 节）。

扫描格式对象在下面定义，仅供参考较早的设置脚本。

SFormat =

transducer string Scnhd name (should be copied from Trans)

scanFormat string 4 char token for type of scan (RLIN,CLIN,etc)

radius double shortest distance to scan apex if CLIN or VAPX

theta double Scan angle (see figure 2.4.1).

numRays double no. of Rays (1 for Flat Focus)

FirstRayLoc [1x3 double] Location(x,y,z) from origin.

rayDelta double spacing in rads(CLIN,VAPX) or dist.(RLIN,SLIN)

startDepth double starting acquisition depth in wavelengths

endDepth double ending acquisition depth in wavelengths

SFormat.transducer 属性应设置为与 Trans.name 属性相同。其在 SFormat 对象中重复的原因仅供参考，Trans.name 优先于 SFormat.transducer。扫描的类型由一个四字符的标记指定，目前可以是以下之一：

RLIN – 直线扫描格式，传感器元件位于 x 轴。增加深度是沿着正 z 轴。此格式用于线性阵列。

SLIN – 转向线性扫描格式，传感器元件位于 x 轴。

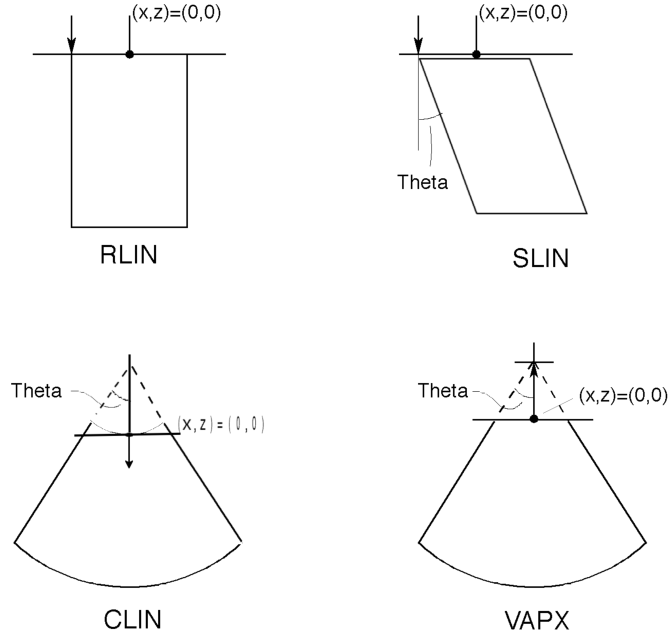
CLIN – 曲线线性扫描格式，换能器元件位于半径 R 的圆弧上，从 x=0、z=0 的原点开始。

VAPX – 虚拟 Apex 扫描格式，换能器元件位于 x 轴，扫描线从换能器后面距离 R 的原点辐射。这种格式用于相控阵和细间距线性阵列。

PYRM – 金字塔形 3D 扫描格式，其中换能器元件包含在位于 x,y 平面中且以 x=0、y=0、z=0 为中心的正方形中。扫描在正 z 轴上以扩展的锥体形状延伸。

CONE – 锥形 3D 扫描格式，其中换能器元件包含在位于 x,y 平面中且以 x=0、y=0、z=0 为中心的圆圈中。圆锥的顶点位于 z 轴上，距离 SFormat.radius 波长。

各种扫描格式如下图 2.4.1 所示。



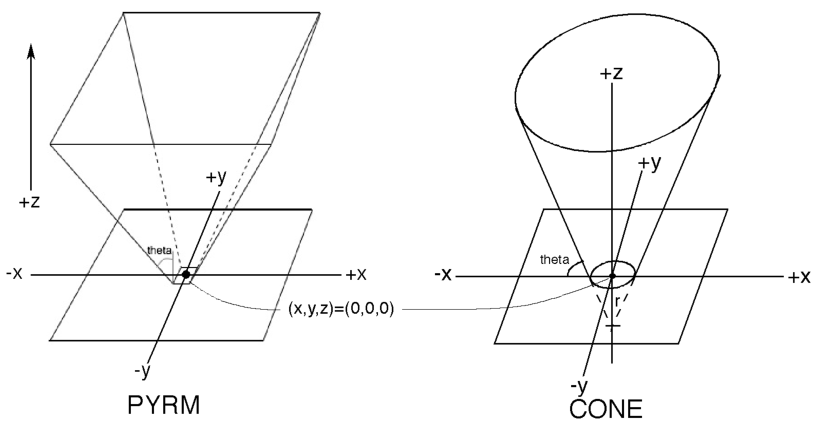


图2.3.1、扫描格式

SFormat.radius 属性用于 CLIN、VAPX、PYRM 和 CONE 格式。对于 CLIN，它指定换能器元件的曲率半径。对于 VPAX、PYRM 和 CONE 格式，它指定从扫描的虚拟顶点到换能器元件平面的距离。不需要为 RLIN 和 SLIN 格式指定此属性。

SFormat.numRays 属性指定用于组成扫描的光线数。射线将扫描格式划分为可由特定发射光束模式照射的区域（示例射线区域由图 2.4.1 中的浅灰色线显示）。射线的数量可以少至一条，对于闪光发射类型的扫描，其中整个场都被单次发射照亮。通常，扫描区域完全充满均匀的光线间距。

对于 2D 扫描格式，属性 SFormat.firstRayLoc 和 SFormat.rayDelta 对于 2D 扫描格式，属性 SFormat.firstRayLoc 和 SFormat.rayDelta 指定扫描区域如何填充射线，前者指定相对于第一条射线线原点坐标中心的位置。后者以波长（RLIN、SLIN 格式）或弧度（CLIN 和 VAPX 格式）为单位指定到扫描中下一条和连续射线线的原点的线性距离。例如，对于用单个脉冲照亮整个场的 RLIN 闪光发射扫描，firstRayLoc 值将是 [0,0,0]，而 rayDelta 值将是换能器的宽度（以波长为单位）。

对于 3D 扫描格式（PYRM 和 CONE），目前使用 SFormat 时唯一支持的扫描方法是闪光发射方法（用单个发射脉冲照亮整个体积），这需要将 SFormat.firstRayLoc 设置为 [0,0,0 ] 和 SFormat.rayDelta 设置为 2\*SFormat.theta。要在 3D 体积内指定区域，请使用 PData.Region.Shape 定义而不是 SFormat。

最后，SFormat.startDepth 和 SFormat.endDepth 属性指定从换能器平面到扫描开始和结束深度的距离，以波长为单位。

**2.4 PData Object（PData对象）**

PData对象(简称“像素数据”)描述图像重建软件要处理的像素区域。如果不需要图像重建，则可以从安装脚本中省略此对象。当进行图像重构时，PData对象指定一个较大的图像空间中的一个或多个像素区域，该区域是相对于传感器坐标系定义的。仅在指定区域中包含的像素或体素位置执行图像重建。结构定义如下:

PData =

sFormat double % (deprecated) no. of SFormat structure

Coord string % [‘rectangular’],‘polar’,‘spherical’

PDelta [1x3 double] % spacing between pixels in all dimensions

Size [1x3 double] % rows, cols and sections

Origin [1x3 double] % % x,y,z of top lft corner (for 2D) or top

lft far corner (3D view from z axis).

Region [numRegions Structs] % Region structures

如前所述，sFormat 字段已被弃用，因为不再需要 SFormat 来定义 PData 空间内的区域。下面解释的新方法使用 PData.Region.Shape 定义来定义重建区域，并在指定感兴趣区域时提供更大的灵活性。

PData.Size 属性在行、列和部分中定义（对于 3D 体积）。对于直角坐标中的典型二维扫描，行维度对应于 Z 轴（增加的行对应于增加的 Z 值），而列维度对应于 X 轴。截面大小设置为 1。在极坐标中，行对应于 R 维度，而列对应于 Theta 维度。对于 3D 体积扫描，轴排列发生了变化，行尺寸对应于 Y 轴，截面尺寸对应于 Z 轴。在这种情况下，截面是平行于 X-Y 轴平面的平面。

PData.Origin 属性定义 PData 区域在传感器坐标系中的位置。对于 2D 扫描，原点始终位于 X-Z 平面中，y 坐标为零，对于直角坐标，当在 X-Z 平面上方查看扫描时，原点标识扫描的左上角，X坐标表示每个元件的中心位置（例如128个元件，第64和65元件的中间位置x=0，最左边是（-元件尺寸\*63.5），最右边是（元件尺寸\*63.5）），Z 从上到下增加。对于极坐标，原点是极坐标原点在 R=0 处的位置。对于 3D 体积扫描，原点通常位于 X-Y 平面，但也可以具有 z 分量。如果从正 Z 的角度回头看 X-Y 轴，PData“盒子”的原点是体积的最左上角。

对于 PData.Coord =‘rectangular’（默认值），PData 数组始终是矩形区域（用于 2D 扫描）或长方体（用于 3D 扫描）。

PData.PDelta 属性中提供了 x、y 和 z 维度中像素之间的增量。 [为了向后兼容，增量仍然可以由单个属性 PData.pdelta 或 pdeltaX、pdeltaY 和 pdeltaZ 的单个属性设置，尽管不再建议使用这些属性。] 在线性阵列扫描中，其中 深度分辨率通常优于横向分辨率，可以指定适合深度分辨率的 PDelta(3) 或 z 增量，以及适合横向分辨率的 PDelta(1) 或 x 增量。 当一个非方形像素 PData 区域被渲染到一个显示窗口时，这些像素将被内插到由显示窗口规范设置的像素增量处的方形像素。

线性阵列换能器(128\*Trans.spacing)

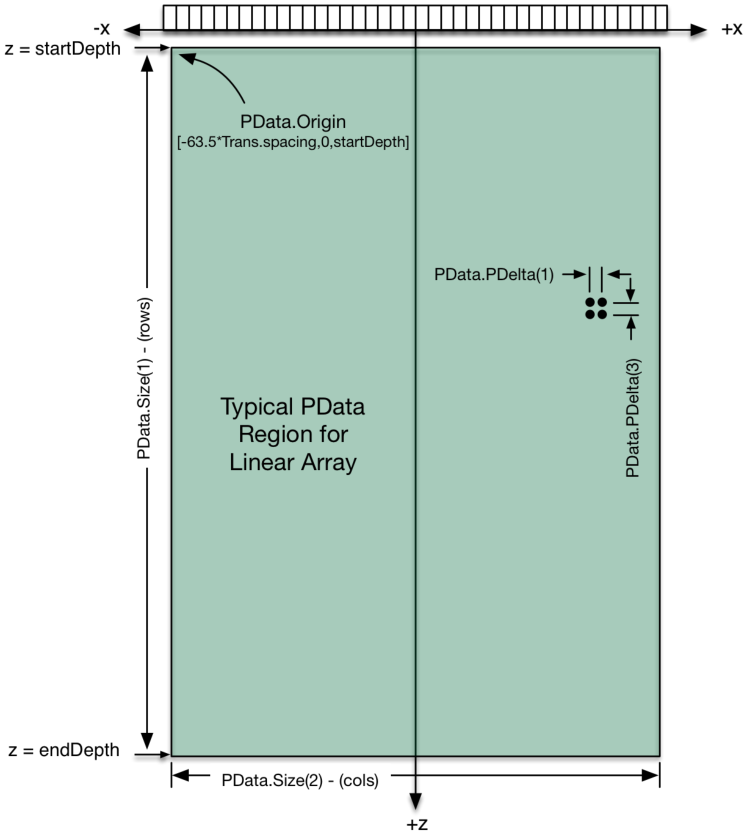


图2.4.1、线性阵列的直角坐标 PData 规范

对于 PData.Coord =‘polar’,PDelta 值被解释为 theta、r 和 z（z 仅用于圆柱坐标，尚不支持）。 在这种情况下，PData 区域是圆的一部分，顶点位于 PData.Origin 坐标处，这些坐标在换能器的坐标系（通常为矩形）中指定。 角度增量 theta 指定从垂直线到顶点的逆时针角度增量。 θ 线始终关于通过顶点且平行于 z 轴的线对称定位（参见下面的图 2.4.2）。

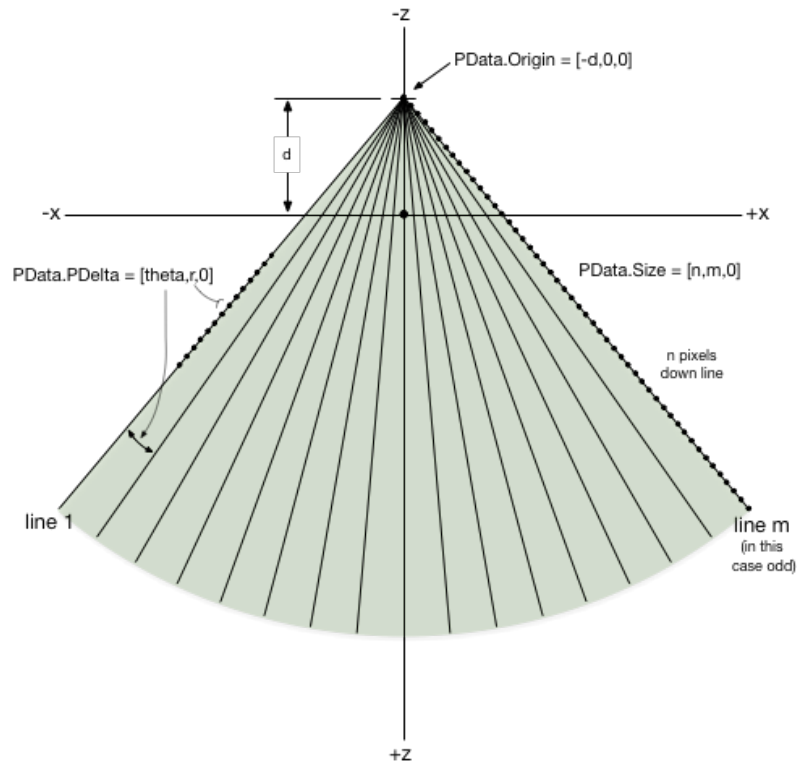


图2.4.2、极坐标PData规范

PData.Size 规范设置像素区域的大小（以像素为单位）用于 2D 扫描或用于 3D 扫描的体素。对于直角坐标，行维度对应于 z 方向，列维度对应于 x 方向，二维扫描的第三维度指定为 1。对于 3D 扫描，行和列索引分别对应于 x 和 y 维度，第三个或部分索引等于 z 维度。对于极坐标，行索引对应于 r 坐标，列索引对应于 theta 维度。由于极区关于通过顶点的垂直线对称，因此 theta = 0 方向始终位于 PData 列的中心。例如，如果 theta 大小 PData.Size(2) = 100，则 theta = 0 线将位于第 50 列和第 51 列之间。如果 theta 像素增量 PData.PDelta(1) = 0.01 弧度，则 PData 区域将是圆的一部分，其顶角为 0.01\*(100-1) 弧度或 56.72 度。第一列的角度为 -49.5 \* .01 弧度或 -28.36 度，而第 100 列的角度为 +28.36 度。

在 PData 数组的区域内可以定义多个具有各种几何形状的区域。仅对定义区域中包含的像素或体素执行图像重建。

**2.4.1 PData.Region Objects(PData.Region对象)**

Region 对象指定属于构成 PData 对象的各种像素区域的像素。 PData.Region 结构描述了 PData 阵列中将要处理的像素子集，该子集通常由换能器的类型和介质扫描的格式确定。 例如，相控阵换能器的扇区格式可以在 PData 矩形中指定，方法是创建一个 PData.Region 结构，该结构指定要处理的像素的扇区形子集。 随着 3.x 软件的发布，要处理的像素由 PData 数组中像素的线性索引列表定义。 这允许创建任意形状甚至不连续的区域。

最简单的区域规范是构成区域的 PData 数组中像素的线性地址的规范。 属性如下：

PData.Region =

numPixels double % no. of pixels in region (or voxels in 3D section)

PixelsLA [1xnumPixels]int32 % linear address of each pixel (from 0)

PData.Region 中的 PixelsLA 数组包含该区域中包含的像素的 PData 数组的线性地址列表，其中索引从 0 而不是 1 开始（从 Matlab 索引中减去 1）。在这种情况下，不需要额外的属性，因为定义足以定义区域。可以在具有不同 PData.Region() 索引的同一 PData 空间中定义多个 Region 结构。用于图像重建的 ReconInfo 结构将引用 PData.Region 索引号来指定要处理的区域。

或者，用户可以使用 Region.Shape 属性仅指定区域结构的形状。在这种情况下，shape 属性指定了一个具有附加属性的结构，这些属性需要设置形状的位置、大小和方向。指定 Shape 时，numPixels 和 PixelsLA 属性可以为空或缺失，因为它们通常会通过调用“computeRegions”实用程序函数来添加或替换。例如，矩形定义如下。

PData.Region.Shape =

struct('Name','Rectangle','Position',[50,0,25],'width',32,'height',65));

直角坐标 PData 数组支持其他 2D 形状名称，包括“平行四边形”、“梯形”、“圆形”、“环形”、“三角形”、“扇形”和“扇形”（平顶）。支持的 3D 形状是“锥形”、“金字塔”、“截面”和“切片”。每个形状都有自己独特的属性来定义它的特征。目前，极坐标 PData 数组支持的唯一形状是“Sector”和“SectorFT”。

如果 Region 结构包含受支持的 Shape 属性，则实用函数“computeRegions.m”可用于计算 Region 的 numPixels 和 PixelsLA 属性。如果在用户的设置脚本中找不到 numPixels 和 PixelsLA 属性，或者 PData.Region 结构丢失或为空，VSX 会自动调用该实用程序。 computeRegions.m 实用程序（注意以前版本的名称更改）不再将区域划分为多个部分以进行重建线程处理，因为现在由重建例程自动完成。如果未指定 PData.Regions，则由 computeRegions 实用程序添加单个 PData.Region.Shape，名为“PData”，它指定整个 PData 空间。

可以组合形状以创建更复杂的区域。在 Shape 结构中，如果提供了属性“andWithPrev”，其值等于先前定义的形状的索引，则新形状与前一个形状“与”，给出两个形状之间的交集。例如，人们可能想要引导线性阵列的波束以进行多普勒传感，但将波束的处理保持在波束没有转向的成像区域内。 2D 图像区域的形状将首先定义为与 2D 扫描格式匹配的矩形。然后，转向光束的区域可以定义为平行四边形，并与第一个区域“与”，如下图所示。

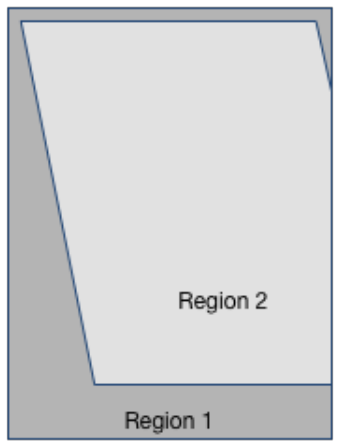


图2.4.1.1、使用‘andWithPrev’来相交两个Region

下面的文本提供了支持的PData.Region.Shape定义。

Rectangle形状结构定义如下：

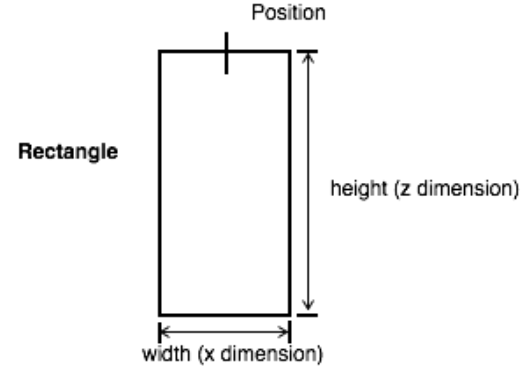
Shape =

Name string % ’Rectangle’

Position [1x3 double] % x,y,z coordinate of center of top segment

width [double] % x dimension

height [double] % z dimension



Parallelogram (平行四边形)只是添加了一个转向角度的矩形属性。

Shape =

Name string % ‘Parallelogram’

Position [1x3 double] % x,y,z coordinate of center of top segment

width [double] % x dimension

height [double] % z dimension

angle [double] % angle between vertical and left side



梯形或三角形的 Region.Shape 结构定义如下（对于三角形，顶部或底部长度应设置为零）：

Shape =

Name string % ‘Trapezoid’ or ‘Triangle’

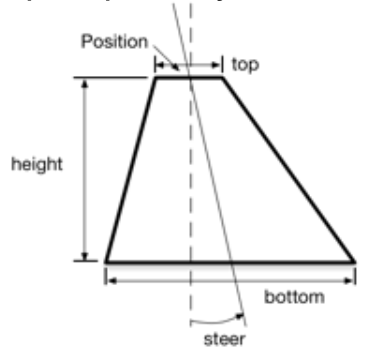
Position [1x3 double] % x,y,z coordinate of center of top segment

top [double] % length of top segment

bottom [double] % length of bottom base

height [double] % height (z dimension)

steer [double] % angle between z axis and line thru midpoints



“Circle”、“Annulus”和“Sector”的 Region.Shape 结构都可以使用以下属性定义（如果不需要属性，则可能会丢失）：

Shape =

Name string % ‘Circle’, ‘Annulus’ or ‘Sector’

Position [1x3 double] % x,y,z coordinate of center of circle or apex

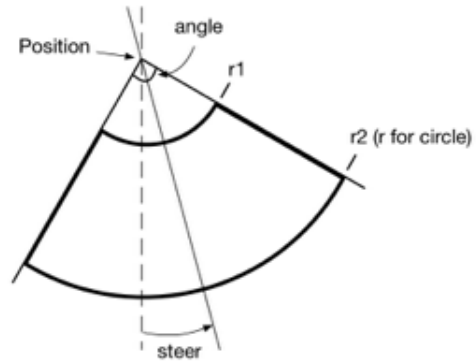
r [double] % radius (‘Circle’ shape only)

r1 [double] % inner radius (‘Sector’ or ‘Annulus’ only)

r2 [double] % outer radius (‘Sector’ or ‘Annulus’ only)

angle [double] % angle between lft and rt side of ‘Sector’

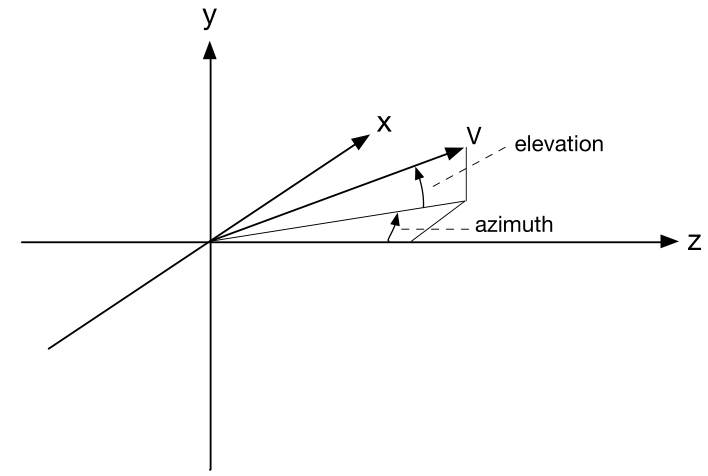
steer [double] % angle between z axis and centrln of ‘Sector’



“SectorFT”形状定义类似于“Sector”定义，不同之处在于“Sector”定义的 r1 和 r2 属性被 z 和 r 属性替换。 z 属性定义了切断扇区顶部的水平线的位置。如果将 PData 坐标系定义为“极坐标”，则整个 PData 区域不是矩形，而是圆的扇形，关于 通过 PData.Origin 的垂直线。 在这种情况下，扇区形状必须具有与 PData.Origin 相同的 Shape.Position 属性。

对于 3D 体积扫描，PData.Size 属性包含 3 个值 - 行、列和页面（页面有时称为部分）。行平行于 x 轴并沿负 y 方向增加，而列平行于 y 轴并沿正 x 方向增加。然后页面或部分平行于 x,y 平面，当从正 z 轴回看原点时，行和列处于其正常方向。 2D 换能器阵列元件通常布置在 x,y 平面中，以 z 轴的原点为中心，但这种布置不再是强制性的，换能器元件可以放置在 3D 空间中的任何位置。

3D 扫描的坐标系如下所示。换能器元件法线和矢量的方向使用两个角度指定 - 方位角和仰角。这些角度是从正 z 轴到矢量在 x、z 平面（方位角）中的投影以及从 x、z 平面到矢量（仰角）的投影。在换能器元件法线的情况下，指定方位角和仰角，就好像元件中心位于原点一样。



“Cone”(锥体)Region.shape 可以定义为在 PData 体积中指定一个圆锥形状。 圆锥的顶点必须位于负 z 轴上，角度和转向属性指定圆锥形状。 转向时，锥体锚定在其顶点，并且仍与平行于 x,y 平面的平面保持圆形相交。 锥体也可以在距顶点两个距离处被选择性地截断（圆锥截头体），其平面平行于 x,y 平面。 'Cone' Region.Shape 结构定义如下：

Shape =

Name string % ‘Cone’

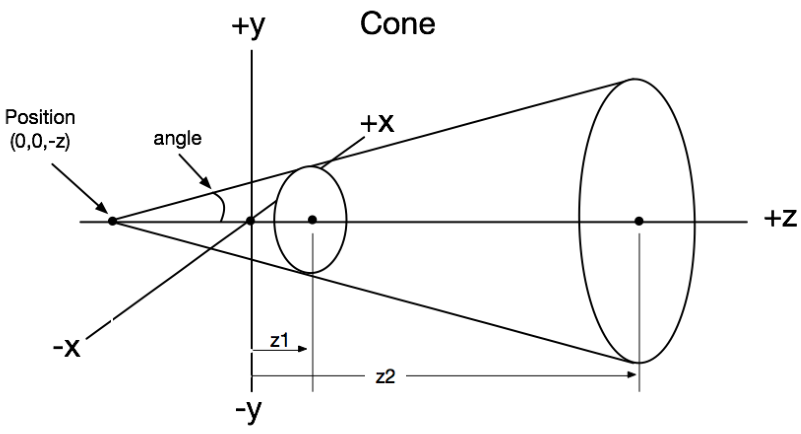
Position [1x3 double] % x,y,z coordinate of apex of cone

angle [double] % angle - centerline to outer surface

z1 [double] % optional - distance to 1st cross section.

z2 [double] % optional - distance to 2nd cross section.

Steer [1x2 double] % optional - centerline azimuth and elevation



可以定义“Pyramid”Region.shape 以指定 PData 卷中的金字塔形状。 金字塔的顶点必须位于负 z 轴上，角度和转向属性指定了金字塔的形状。 转向时，金字塔锚定在其顶点，但仍与平行于 x,y 平面的平面保持正方形相交。 金字塔也可以选择在距顶点两个距离处被截断，其平面平行于 x,y 平面。 'Pyramid' Region.Shape 结构定义如下：

Shape =

Name string % ‘Pyramid’

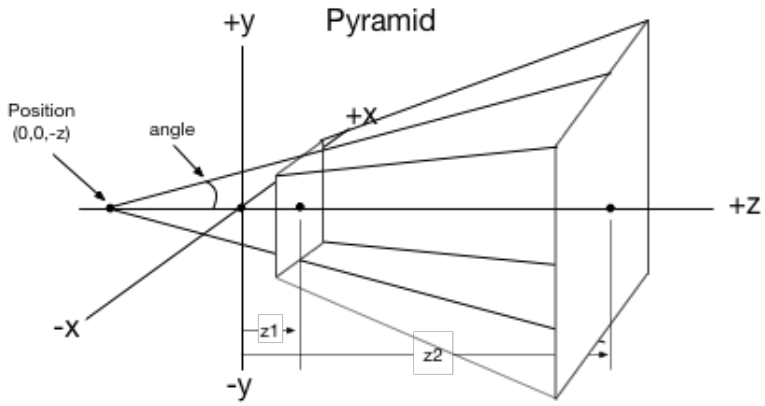
Position [1x3 double] % x,y,z coordinate of apex of pyramid

angle [double] % angle - centerline to outer surface

z1 [double] % optional - distance to 1st cross section.

z2 [double] % optional - distance to 2nd cross section.

Steer [1x2 double] % optional - centerline azimuth and elevation



三维 PData 数组可以具有由一系列页面或部分组成的区域。 “截面”形状定义如下：

Shape =

Name string % ‘Section’

Sections [1x2 double] % start section, number of sections

起始节加上节数必须小于 PData 数组的总节数。

一个三维 PData 数组也可以被一个二维平面切割，该二维平面平行于由两个轴形成的平面之一。‘Slice’(切片)区域定义如下：

Shape =

Name string % ‘Slice’

Orientation string % slice plane - ‘xz’,’yz’, or ‘xy’

oPAIntersect double % intersect coordinate with out of plane axis

**2.5 Media Object(媒介对象)**

Media Object仅用于在仿真模式下操作系统，描述仿真目标在传感器坐标系几何空间中的位置和特征。该空间中的所有对象都由一个或多个点目标定义，它们的位置和反射率可以单独指定。媒介对象的结构有以下属性:

Media =

model string % (optional) name of media model

function string % (optional) Matlab function to execute

for moving or modifying media points.

FlowObj struct % one or more FlowObjs for Doppler acq.

attenuation double % attenuation in dB/cm/MHz

numPoints double % no. of media pts (incl. FlowObjs)

MP [numPointsx4 double] % x,y,z,reflectivity(0-1.0)

Media.function 属性用于动态媒介模型，其中目标在连续采集之间改变其位置或特征。 如果存在，并且 Receive.callMediaFunc 属性为真 (=1)，则在模拟接收数据之前调用字符串指定的 Matlab 函数（m 文件）。 如果默认目录中不存在该文件，则应包含该文件的路径。

**2.6 Resource Object(资源对象)**

资源对象包含系统和存储缓冲区的配置信息。表单如下所示(必需的属性突出显示)。

Resource =

RcvBuffer =

datatype string ‘int16’ (default)

rowsPerFrame double number of rows in frame

colsPerFrame double number of columns in frame

numFrames double number of frames (1 or even number)

lastFrame double last frame transferred into buffer

InterBuffer =

datatype string ‘complex double(default)/single’

rowsPerFrame double no. of rows (optional if PData defined)

colsPerFrame double no. of cols (optional if PData defined)

sectionsPerFrame double sections (optional: 1 or PData defined)

pagesPerFrame double pages (optional: default=1)

numFrames double number of frames

ImageBuffer =

datatype string ’double’ (default)

rowsPerFrame double no. of rows (optional if PData defined)

colsPerFrame double no. of cols (optional if PData defined)

sectionsPerFrame double sections (optional: 1 or PData defined)

numFrames double number of frames

firstFrame double first image frame of multi-frame buffer

lastFrame double last frame of multi-frame buffer

DisplayWindow =

Title string ‘Window title’

mode double 2D=0 (default), 3D=1

Orientation string ‘xz’(default),’yz’, or ‘xy’

AxesUnits string ‘wavelengths’(default) or ‘mm’

Position double[1x5] [left, bottom, width, height,[depth]]

ReferencePt double[1x3] upper lft corner(x,y,z) in trans coor. sys

pdelta double pixel delta in wavelengths (default=0.25)

Colormap double[256x3] default Colormap is ‘grey’

numFrames double no. of frames in display cineloop (dflt 1)

firstFrame double first frm in cineloop (system provided)

lastFrame double last frm in cineloop (system provided)!

clrWindow int if !0, clrs the dsply wndw b4 nxt update

Parameters =

Connector (3.2+) double connector # (1(dflt),2,or [n,m,...])

connector (3.0x) double connector # (0(both),1(dflt),2)

numTransmit double number of available transmitters

numRcvChannels double number of available receive channels

speedOfSound double speed of sound in meters/sec (deflt 1540)

speedCorrectionFactor double corrects speed of sound (default=1.0)

startEvent double event no. to start at after freeze (dflt=1)

simulateMode double 0=’use VDAS’(default), 1 = ‘simulate’

fakeScanhead double (dflt=0) if 1, allow running without probe

GUI string ‘vsx\_gui’ [default] or ‘vsx\_guiOld’

UpdateFunction string ‘VsUpdate’ [default] or user specified

verbose double 0:3 int specifying level of warnings/errors

initializeOnly double 0(default), set to 1 to init. and exit

ProbeConnectorLED double see User Manual for description

ProbeThermistor double see User Manual for description

SystemLED cellarray[1x4]see User Manual for description

VDAS =

numTransmit double hardware supported numTransmit

numRcvChannels double hardware supported numRcvChannels

testPattern double if 1, use test pattern instead of A/D

dmaTimeout double timeout for DMA trnsfrs in msec, dflt 1000

sysClk double frequency of system clock

halDebugLevel double value for enabling debug features

watchDogTimeout double timeout for refreshing watchDog timer

HIFU =

externalHifuPwr double use of External HIFU Option

verbose double enable command line output

extPwrComPortID string ID of power supply control port

voltageTrackP5 double imaging profile tracks profile 5

TXEventCheckFunction string name of limit check function

extPwrConnection string ‘series’ or ‘parallel’

SysConfig (read only) =

System string type of system, e.g. “Vantage 256 UTA”

Option cell array of strings identifying options present

SWconfigFault double non-zero if SW configuration faults present

HWconfigFault double non-zero if HW configuration faults present

empty if no hardware present.

FPGAconfigFault double non-zero if any FPGA revision incorrect,

empty if no hardware present.

UTA double one of four possible values:

empty - no HW present, or SHI/baseboard

fault condition if HW system is present.

0 - HW system with no UTA baseboard or SHI.

1 - UTA baseboard module is present.

-1 - pre-UTA SHI module is present.

UTAtype 1x3 double identifies the specific UTA adapter module

installed, or empty if not a UTA system

UTAname string identifies the UTA adapter module installed

SWversion 1x3 double identifies the system SW revision level

HALversion string identifies the HAL revision level

DriverVersion sting identifies the WinDriver revision level

VDAS double non-zero if acquisition module(s) present.

AcqSlots 1x4 double array with each entry set to one or zero

based on the presence / absence of an

acquisition module in the associated slot.

TXindex double integer index value identifying the

transmit signal path configuration of the

acquisition modules.

RXindex double integer index value identifying the receive

signal path configuration of the

acquisition modules

simLicense,hwLicense,

extendedTXLicense,arbTxLicense,

arbToolboxLicense,triggerLicense,

reconLicense double License status: 0 or 1

在更完整地描述Resource属性之前，有必要简单介绍一下Verasonics系统中的数据流。Verasonics系统可以最好地理解为实现从一种内存缓冲区类型到另一种类型的数据转换。各流程的数据流顺序如下:

Acquisition(采集)——在 Vantage 硬件模块的本地通道存储器中采集的 RF 数据被发射到计算机主机存储器中的 RcvBuffer。

Pixel reconstruction(像素重建)——处理从RcvBuffer获取输入RF数据，然后输出到InterBuffer(用于 I、Q 像素)和/或 ImageBuffer（用于强度像素）。

Processing(处理)——从一个 InterBuffer 或 ImageBuffer 到另一个（或者可能是同一个缓冲区）。

Display——从一个或多个 ImageBuffers 到 DisplayWindow 缓冲区。

为了允许以仅受声速限制的速率捕获射频超声数据，数字化通道数据临时存储在系统采集模块上的高速本地存储器中。在采集了一定数量的数据（通常是一帧的所有采集数据）后，通道数据被发射到主机的内存中。用于接收发射到主机的数据的内存缓冲区是 RcvBuffer，它为每个接收通道、每个帧组织成样本。可以分配多个 RcvBuffer 来保存不同类型的采集数据，但一般情况下，最好将处理一个图像帧所需的所有采集数据（例如 2D 和多普勒采集）存储在同一个缓冲区中。在 RcvBuffer–Resource.RcvBuffer(m) 之后使用数组索引定义多个缓冲区。 RcvBuffer（实际上是所有缓冲区类型）的一个特点是，每个定义的缓冲区都由包含相似内容的帧组成，并且在行、列（可能还有节和/或页）中大小相同。定义的帧数通常取决于扫描方法以及在 RF 或图像数据 cineloop 中回放处理所需的帧数。

基于像素的图像重建方法直接处理一个或多个 RcvBuffer 中的数据，以计算用于合成孔径和多普勒应用的基带 I、Q 数据或用于正常 2D 成像的强度数据。这些计算的结果存储在 InterBuffer（用于复杂数据）或 ImageBuffer 中。重要的是要记住，重建的输出由 PData 对象指定的矩形网格中像素点处的 I、Q 数据或强度数据组成，这些数据以与换能器的特定关系定义。没有射线线的概念，其中重建输出是相对于换能器处于某个方向的数据点线；尽管对于某些多普勒应用，可能需要沿引导线定义“像素”点。

当需要额外的处理时（例如从 I、Q 像素数据计算多普勒频移数据），Interbuffer 或 ImageBuffer 可以用作源缓冲区，而 ImageBuffer 用作目标。再次注意，处理是针对像素点进行的，输入输出缓冲区中的元件通常是一一对应的。最后，当需要将图像写入显示器时，ImageBuffer 中的数据被转换为像素数据，最终进入 DisplayWindow 缓冲区。 DisplayWindow 缓冲区由直接复制到输出窗口的单帧像素数据组成。 DisplayWindow 通常具有比相应 ImageBuffer 中指定的像素更多的像素，因为通常需要将像素插值到更高的密度以在高分辨率显示器上提供有用的图像尺寸。

各种资源缓冲区定义创建了保存相应数据的实际 Matlab 数组。创建的相应 Matlab 数组具有以下名称：

RcvBuffer(:) – Matlab 元胞数组 RcvData{n}(i,j,k)，其中 n 是缓冲区编号，i 是行索引，j 是列索引，k 是帧编号。

InterBuffer(:) – Matlab 元胞数组 IQData{n}(i,j,k,l,m)，其中 n 是缓冲区编号，i 是行索引，j 是列索引，k 是节编号，l 是页码，m 是帧数。

ImageBuffer(:) – Matlab 元胞数组 ImgData{n}(i,j,k,m) 和 ImgDataP{n}(i,j,k,m)，其中 n 是缓冲区编号，i 是行索引，j 是列索引，k 是节号，m 是帧号。

**2.6.1 RcvBuffer Attributes**

RcvBuffer 接收从 VDAS 模块上的本地存储器发射的射频数据。缓冲区的组织结构如下图 2.6.1.1 所示。 RcvBuffer 可以包含一帧或偶数个相同的帧（在此示例中为 6 帧）。每个帧由许多列组成，这些列对应于用于获取该帧的通道数。通道通常是物理采集通道，它连接到换能器的单个元件以用于发送/接收采集事件（连接可能会随着多路复用换能器的后续采集而改变）。沿着列或通道向下，每一行代表一组 RF 数据样本，随着时间的推移沿着列向下增加。如果组成帧需要多个发送/接收采集事件，则每个事件的样本垂直向下堆叠在列中，如图 2.6.1.1 中的不同颜色所示。在这种情况下，8 个采集事件用于帧。多采集帧中的采集事件可以具有不同的样本数量和不同的样本模式属性。 RcvBuffer 中的总行数 N 必须大于所有采集的样本总数（在这种情况下，大于 8\*M）。如果 RcvBuffer.rowsPerFrame 值超过 524288 个样本，则某些计算平台上的 DMA 缓冲区大小限制可能会生效。

当退出使用 VDAS 硬件运行的序列或使用 UI 按钮进入冻结状态时，系统会设置 RcvBuffer.lastFrame 属性。它设置为发射到缓冲区的最后一帧（请注意，这不一定是处理的最后一帧）。

这允许外部处理函数“解包”RcvData 数组，以便第一帧是缓冲区中最旧的帧。RcvBuffer.datatype 指定样本的位大小，当前 Vantage 硬件默认为“int16” .这是目前唯一支持的数据类型。

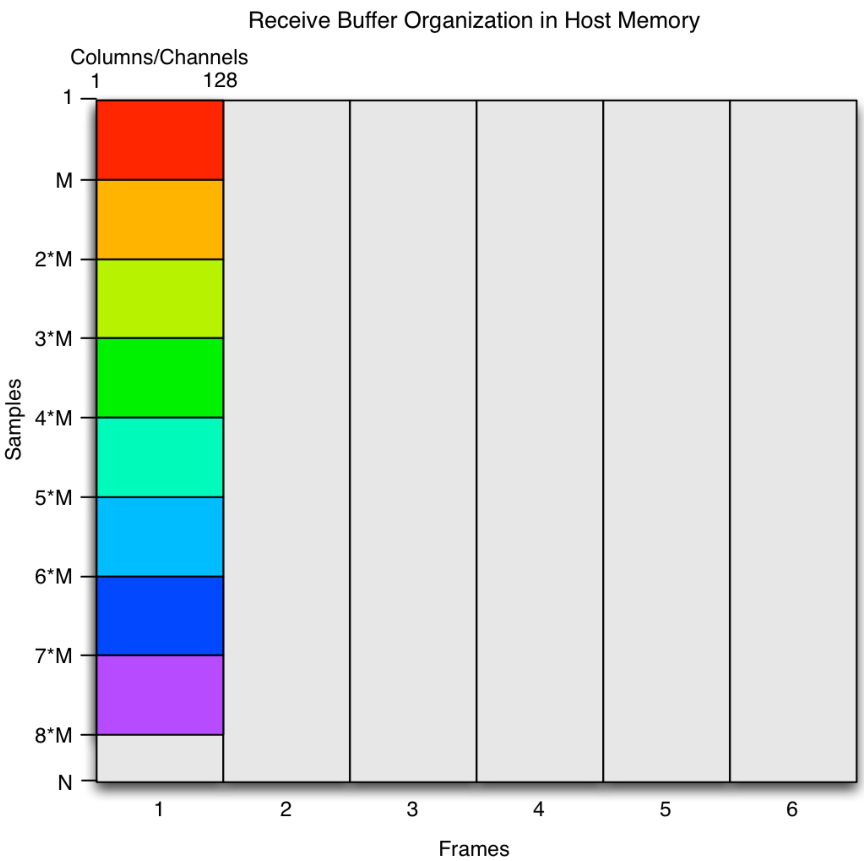


图2.6.1.1、具有6帧和每帧8个采集事件的RcvBuffer

**2.6.2 InterBuffer属性**

InterBuffer 接收像素重建处理的复数输出，因此应始终具有“复数”数据类型。数据类型可以是“复数双精度”或“复数单精度”。 请注意，目前，复杂的单一数据类型不能与内部图像处理例程一起使用，而是用于可能在 GPU 上执行的外部处理功能。InterBuffer 的元件与 PData（像素数据）数组的元件一一对应，后者定义了重建点相对于换能器的位置。 InterBuffer 可以具有比 PData 数组更大的大小（更多的行和/或列），但不能更小。如果定义了更多行或列，InterBuffer 帧的线性索引将与 PData 数组的线性索引对齐。通常，最好不要在 Resource.InterBuffer 定义中指定 InterBuffer 帧的行和列，在这种情况下，这些属性将默认为与 PData 数组相同的大小。换句话说，如果没有定义：

Resource.InterBuffer.rowsPerFrame = PData.Size(1);

Resource.InterBuffer.colsPerFrame = PData.Size(2);

Resource.InterBuffer.sectionsPerFrame = PData.Size(3); % for 3D scans

InterBuffer可以用多个帧定义，在一个帧中，除了行和列之外，还可以有两个额外的索引维度。对于三维体积扫描，行和列维度之后的下一个维度是分段。在三维扫描中，截面是通过扫描体的xy切片，该扫描体通常与二维换能器阵列的表面在距离换能器z处平行。当有多个截面时，它们通常以与截面内像素相同的空间增量间隔。第二个附加索引是页面索引。该指标通常用于多普勒采集，如彩色血流成像。页面索引将用于存储用于获取帧或感兴趣区域的多普勒信号的多个多普勒采集(通常称为集合)的结果。

**2.6.3 ImageBuffer属性**

ImageBuffer 接收像素重建方法的强度输出，也可以作为信号处理方法的目标输出缓冲区。当前支持的唯一数据类型是“double”。 ImageBuffer 中的列数始终与定义的矩形像素网格的一行上的像素数相同，但行数可以大于网格中的行数。如果为 ImageBuffer 中的数据定义了 PData 数组，则可以省略 rowsPerFrame、colsPerFrame 和可能的 sectionPerFrame 属性，并将默认为 PData.Size 规范中的值。

在处理 ImageBuffer 以显示图像数据时，会创建一个与 ImageBuffer 大小相同的二级缓冲区，称为 ImgDataP ，用于空间过滤和持久化处理。然后使用 ImageP 缓冲区将像素渲染到 DisplayWindow。这允许 ImageBuffer (ImgData) 保留原始重建的输出数据以进行可能的重新处理。

ImageBuffer 资源对象包含可用于多帧 ImageBuffer 的 firstFrame 和 lastFrame 属性。 firstFrame 编号（从 1 开始）是缓冲区中最旧帧的编号，而 lastFrame 编号是最新的。如果处理已填满缓冲区，则 lastFrame 编号将比 firstFrame 编号落后一帧，并且两者都将以新帧前进，环绕缓冲区的末尾。将 lastFrame 属性设置为 0 表示缓冲区为空，并在任何写入缓冲区的渲染操作之前触发清除缓冲区。指定要写入的 ImageBuffer 帧时，允许值为 -1，这表示应写入 lastFrame 的下一帧。大多数处理例程将在写入完成时自动增加 lastFrame 和 firstFrame 指针。当指定要读取的 ImageBuffer 的帧时，-1 的值表示应该使用最近写入缓冲区的帧。这将是 lastFrame 属性的帧号。

**2.6.4 DisplayWindow属性**

DisplayWindow 结构包含 Matlab 和系统软件用来生成输出显示窗口的各种属性。重建图像的像素分辨率通常设置为充分采样超声扫描的空间分辨率，而不是更大，因为这将花费更多的图像重建处理时间，并减慢帧速率。为了将重建的图像渲染到高分辨率显示器，需要根据显示器上所需图像的大小将像素内插到更高的密度。 DisplayWindow 处理执行此插值，将重建的像素分辨率转换为由 Resource.DisplayWindow.pdelta 指定的新像素增量。 DisplayWindow 也可以在计算机屏幕上具有指定的位置，并且相对于换能器的大小和位置与重建窗口不同。计算机屏幕上窗口的位置和大小（以像素为单位）由 Resource.DisplayWindow.Position 给出，它的值是 [x, y, width, height]，其中 x 和 y 是左下角的坐标。相对于换能器的位置由属性 Resource.DisplayWindow.ReferencePt 给出，其值以波长为单位，并指定 DisplayWindow 左上角在换能器坐标系中的位置。

DisplayWindow 区域将显示 PData 数组区域中属于 DisplayWindow 的部分。集成 3D 显示例程时将需要 Resource.DisplayWindow.mode 属性。现在它默认为 2D。在 2D 模式下扫描和显示 3D 体积时，Orientation 属性确定 DisplayWindow 如何与 PData 体积区域对齐。例如，当 Orientation = ‘xz’ 时，DisplayWindow 平面平行于 x,z 平面。在这种情况下，ReferencePt y 值确定 x,z 平面的 y 位置。对于 C 扫描，方向将设置为“xy”，参考点 z 值将确定平面在 z 方向上的深度。各种方向选择的参考点位置如下图所示。 ReferencePt 始终位于两个轴的最负方向，行方向始终沿轴对的第一个位置的轴。

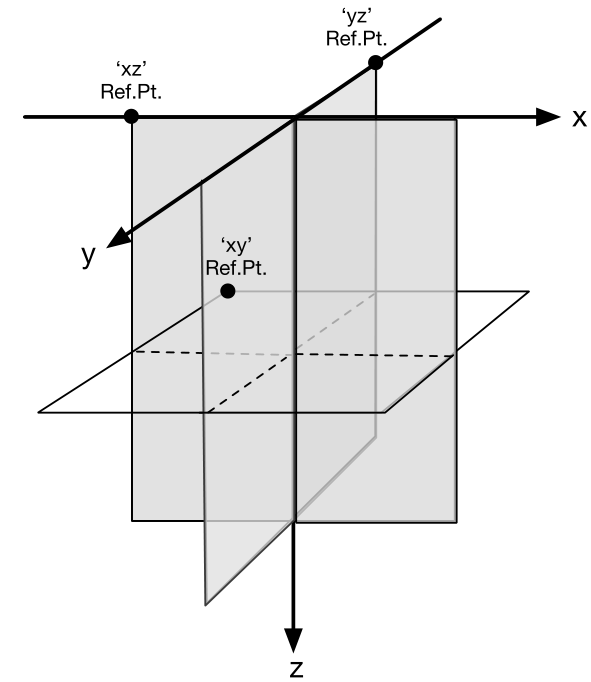


图2.6.4.1、三维体显示窗口的2D平面

显示的 3D 体积切片始终是 DisplayWindow 平面与 PData 体积的交点。目前，平面外的轴值将用于在PData 3D体积中查找最近的体素平面(体素平面之间不进行插值)。用户有责任确保 DisplayWindow 平面与正在积极重建的 PData 体积的一部分相交，否则将不会显示图像数据。

Units属性指定了在DisplayWindow的水平轴和垂直轴上标注的单位('波长'或'毫米')。如果没有提供，Units属性默认为“波长”。

numFrames属性指定DisplayWindow cineloop缓冲区中的帧数。存储在cinloop中的帧与运行时显示在屏幕上的帧是相同的。在3.0之前的版本中，cineloop帧是通过使用Process/imageDisplay属性对ImageBuffer中的帧进行重新处理生成的，并且cineloop帧的数量与ImageBuffer中的帧数量相同。虽然这种旧方法仍可用于某些自定义编程，但 GUI 窗口上提供的 cineloop 控件现在控制显示帧的历史缓冲区，其中帧数与 ImageBuffer 帧数无关。 现在可以将 InterBuffer 和 ImageBuffer 中的帧数设置为双缓冲或唯一处理所需的数量。 在冻结或退出时，DisplayWindow 的 firstFrame 和 lastFrame 属性被复制到 Matlab 工作区，以便可以确定显示的最旧和最近的帧。 一个名为“cineDisplay”（参见第 4.2 节）的新 runAcq 命令，它具有“displayWindow”和“frameNumber”参数，可用于显示 cineloop 缓冲区中的特定帧。 可以从冻结状态或外部函数调用此 runAcq 命令。 cineloop UIControl 的工作方式与以前相同，仅当为 DisplayWindow 定义的 numFrames 大于 1 时才会呈现。

如果定义了 DisplayWindow 结构，VSX 现在只会创建 DisplayWindow。 这与之前的行为形成对比，如果定义了 ImageBuffer，则会创建默认的 DisplayWindow。 这允许创建 DisplayWindow 并将其用于自定义数据（如果需要），或者自定义显示例程与用户 Imagebuffer 一起使用。

Resource.DisplayWindow.Colormap 属性指定 Matlab 环境中的标准颜色图。 颜色图是一个 256x3 的 R、G、B 值数组，范围从 0 到 1.0。 来自图像重建的强度值的典型输出图像处理是使用幂函数或对数函数压缩值，并使用线性灰度图显示压缩值。 然后可以通过修改灰度图来获得额外的压缩或扩展，这可以通过使用 colorMap 工具或使用 Matlab 工具栏提供的菜单直接操作 DisplayWindow 的图来执行。

**2.6.5 Parameter attributes of Resource Object（资源对象参数属性）**

这些属性在前面的2.1节中已经描述过。

**2.6.6 VDAS attributes of Resource Object(资源对象VDAS属性)**

资源对象的 VDAS 属性通常由 VSX 加载程序自动插入。 如果用户提供了属性，则加载程序不会自动生成，而是使用用户属性。 这使用户可以覆盖用于复杂脚本或调试和测试的属性。 建议用户在修改大多数 Resource.VDAS 属性之前咨询 Verasonics 支持。

**2.6.7 HIFU attributes of Resource Object(资源对象HIFU属性)**

系统使用资源对象的 HIFU 属性来管理使用 TPC 配置文件 5 以及系统上安装了扩展突发选项或外部 HIFU 选项的操作。 对于不使用配置文件 5 的用户脚本，不使用 Resource.HIFU 结构，并且不会由脚本加载程序 VSX 创建或解析。 有关这些选项和 Resource.HIFU 结构中各个参数的使用的更多信息，请参阅本文档的第 3.2.4 节“高功率发射 (HIFU)”。

1. **Sequence Objects序列对象**

序列对象只是用于定义采集和处理一系列超声采集所需的所有事件和属性的对象集合。

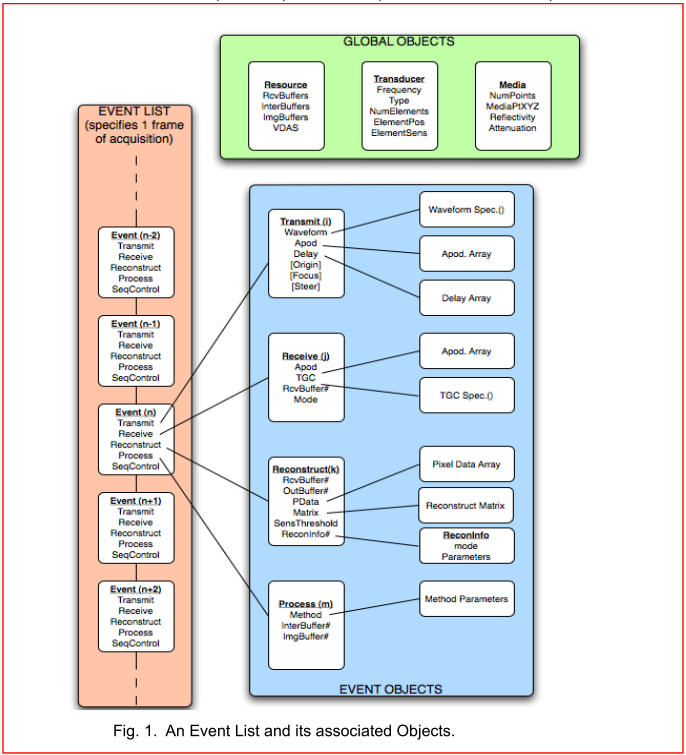


图1、事件列表及其关联对象

顺序读取序列对象的事件以确定采集和处理系统要执行的动作。 每个事件都包含其他对象，这些对象详细描述了超声信息采集和处理的各种组件。 通常，会生成一系列事件以指定一个或多个超声数据帧的采集。 序列的结尾通常会跳回第一个事件，因此序列会无限重复。

Sequence Object 包含的格式是 Matlab 编程语言的 .mat 文件结构。 这种格式是 Matlab 结构和变量的集合，它们捆绑在一起并保存为二进制文件。 Matlab 编程语言用于在离线设置脚本中定义事件及其包含的对象，该脚本将信息保存到 .mat 文件中。 然后，该文件由加载程序 VSX 加载到运行时编程环境中，该程序验证脚本的结构并添加缺失的属性。 VSX 然后在硬件和软件定序器中启动脚本的执行，执行硬件采集和处理，或采集模拟和处理。

以下部分描述了序列对象中使用的各种对象及其关联的事件列表。 对象被定义为 Matlab 编程语言格式的结构。 如前所述，并非需要定义各种结构的所有属性，因为 VSX 将为缺少的属性提供适当的默认值。 对于某些结构，有以字符“VDAS----”开头的属性仅由 VSX 定义。 在使用 Vantage Research System 进行超声采集时，这些属性是对硬件进行编程所必需的，而在模拟模式下运行时会被忽略。 退出已加载并运行的脚本后，可以检查 VSX 添加的属性。 如果 Resource.Parameters.initializeOnly 设置为 1，也可以在加载后查看它们并且无需运行脚本。

* 1. **Event Objects(事件对象)**

为获取一帧或多帧超声数据而执行的事件序列由事件对象的集合定义。 通常，事件对象是设置文件中定义的最后一个结构，但要定义事件使用的对象，必须有一个行动计划。 在定义事件引用的序列对象之前，先勾勒出事件列表中的动作序列是很方便的。 事件对象的属性如下所示。

Event =

info: string 'optional descriptive text for this event'

tx: double # of TX structure array to use for transmit

rcv: double # of Receive structure array to use for rcv.

recon: double # of Recon structure array for reconstruct.

process: double # of Process structure array to use.

seqControl: double # of Sequence Control structure array to use

除第一个属性（info）外，所有属性都指定了与属性同名的特定结构数组的索引，该索引应事先定义（软件会检查引用的结构数组索引是否在定义的索引范围。）。这些结构体数组是可能包含其他对象指针或索引的对象，将在下面的部分中进行全面描述。每个事件都不需要上述所有属性。例如，事件可能只指定重构，在这种情况下，Event.recon 属性将具有 Recon 索引，而 Event.tx、Event.rcv、Event.process 和 Event.seqControl 结构编号将设置为零. （在 Matlab 中，结构体数组索引从 1 开始，因此 Event 属性索引值为零表示不需要操作。） Process 结构体数组用于指定缓冲区到缓冲区的进程，并指定所需的任何处理动作在事件的采集和/或重建组件之后。 Event.seqControl 属性用于指定序列控制操作，例如分支或暂停（可以是有条件的，取决于控制操作）。该对象还可以包含用于控制序列流的硬件特定项，包括启动数据发射的中断操作。

事件结构定义了通常按顺序执行的动作。第一个动作是采集——一个发送，然后是一个接收周期。侦察或重建动作可以对新获取的数据进行操作以产生输出，而处理动作可以对重建输出进行操作。 seqControl 动作通常发生在其他动作之后，尽管有例外，例如 'extTrigger' seqControl 命令的'pause'，它将暂停序列以等待外部触发器，然后再针对它所在的事件执行任何动作指定的。虽然在同一个事件中组合多个动作通常很有用，但这不是必需的；无需任何重建或处理即可执行采集（tx 和 rcv）。事件也可以是仅发送事件，没有指定 rcv，但 rcv 必须始终伴随有 tx 参考。 （可以通过关闭引用的 TX 结构的 TX.Apod 数组中的所有发送器来创建仅接收事件。）

为了定义事件序列，创建了多个事件结构，具有升序索引 - Event(1)、Event(2)、Event(3)，...。 事件（n）。 然后将 n 个 Event 结构按顺序加载到硬件和软件序列器中执行； 或在模拟模式下，依次读取以确定下一个模拟器动作。 有关 Event 对象的更多信息，请参见第 3.7 节。

* 1. **Transmit Objects(发射对象)**

本节介绍指定超声脉冲发射所需的对象。 在为发送脉冲编程系统时需要考虑三个方面：

1) 使用 TW 结构阵列为系统中可用的每个发射器指定发射波形（类型、频率、占空比、持续时间和极性）。

2) 使用 TX 结构阵列，指定要在一个序列中使用的每个发射动作的波束特性，包括哪些发射器在孔径（变迹）中处于活动状态，以及每个活动发射器在发射前的延迟时间。

3) 用于特定发射事件的发射功率电平规范，使用高压设置和发射功率控制器 (TPC) 配置文件设置。

**3.2.1 TW Objects(TW对象)**

可以使用 TW 对象指定系统中每个发射器的发射波形。 通过索引 TW 结构定义可以定义多个 TW 规范。 TW 结构中始终需要的唯一属性是 TW.type，用于指定五个波形定义选项之一，如下所示。 如果波形由 Vantage 硬件生成，类型必须指定为“parametric”、“envelope”或“pulseCode”。 可以使用“参数”类型定义简单的单频突发，而随时间变化的波形需要“包络”或“脉冲代码”类型。

TW(i) =

type string ‘parametric’, ‘envelope’, ‘pulseCode’,

‘function’ or ‘sampled’

对于每个受支持的 TW.type 值，需要该类型特有的附加 TW 属性来表征它。 这些在下面的小节中针对每种单独的类型列出。 对于所有 TW 类型，既可以定义单个波形，然后将其应用于所有发射通道，也可以为每个单独的通道指定唯一的波形。 通过指定具有不同相对脉冲宽度的每通道波形，可以在整个发射孔径上应用幅度加权窗函数。

如下所列的附加 TW 属性将由系统软件在处理用户脚本以在仿真中或在硬件系统上执行时生成。 不应在 SetUp 脚本中指定这些属性，因为它们的值将被系统生成的值覆盖。 下面列出它们以供参考。

TW(i).PulseCode [nx5 double] Waveform definition as series of pulses.

TW(i).Wvfm1Wy [nx1 double] one way simulation wvform (sampled at 250MHz)

TW(i).Wvfm2Wy [nx1 double] two way simulation wvform (sampled at 250MHz)

TW(i).peak [double] distance in wavelengths to peak of wave

TW(i).numsamples[double] no. of samples in Waveform

TW(i).estimatedAvgFreq [double] average frequency in MHz of waveform

TW(i).Bdur [double] waveform overall duration in usec

TW(i).sysExtendBL [double] set to 0 for imaging waveforms less than

25 cycles long; 1 for longer waveforms.

TW(i).CumOnTime[double] sum of the durations in usec of all

active pulses in the waveform.

TW(i).Numpulses [double] total number of active pulses in the waveform.

TW(i).integralPkUsec [1x2 double] peaks of waveform integral (see text).

TW(i).fluxHVlimit [double] max allowed TX voltage for this waveform

TW(i).arbwave [uint16] waveform table for programming the HW

TW.PulseCode 是一个数组，用于以“通用”格式定义发送波形，供系统使用。每个用户提供的 TW.type 波形规范都被转换为 TW.PulseCode 数组并添加到 TW 结构中。然后系统的其余部分使用 TW.PulseCode 对发送器进行编程以进行硬件操作，并合成发送仿真波形以在仿真模式下操作。 TW.Wvfm2Wy 是用于生成仿真波形的发送波形。它是通过将发射三态波通过二阶带通滤波器产生的，其中心频率和带宽由 Trans.frequency 和 Trans.bandwidth 设置。请注意，TW.Wvfm2Wy 将基于仅代表前 20 个周期的实际发送波形的截断副本，因此如果实际发送波形比这更长，则不能代表实际发送波形（这是为了允许仿真运行，用于调试使用长 HIFU 或“推送”发射波形的脚本，但要了解实际模拟结果将不准确）。

TW.peak 表示从波形开始到表示 TW.Waveform 瞬时波形幅度的包络峰值的时间（以 Trans.frequency 的波长为单位）。 TW.peak 与发射延迟时间相加，提供发射脉冲离开发射元件的有效时间，这是提高图像重建精度所必需的。

TW.estimatedAvgFreq 是整个波形的估计“平均”频率，通过计算波形中的极性反转得到，加一得到半周期总数，除以 2 得到总周期数，然后除以通过总波形持续时间来获得以 MHz 为单位的平均频率。

TW.sysExtendBL 是一个逻辑标志，用于通知系统的其余部分是否必须根据其持续时间将此波形使用 TPC 配置文件 5。

TW.CumOnTime 表示波形中所有活动脉冲的累积持续时间（以 usec 为单位）； TW.Numpulses 只是波形中所有活动脉冲的计数。 TXEventCheck 算法使用这些值来评估 HIFU 发射事件期间发射组件的安全操作限制。

TW.integralPkUsec：发射波形积分的正负峰值。这些值表示在波形过程中将出现在发射变压器中的峰值磁通密度；它们用于确定 TW.fluxHVlimit，即必须对发射电压施加的限制，以防止变压器饱和。

TW.arbwave：这是一个 uint16 值数组，用于对硬件系统中的波形生成 FPGA 逻辑进行编程，以合成 TW.PulseCode 定义的实际发送输出。

**3.2.1.1´parametric´ Transmit Waveforms(“参数化”发射波形)**

参数类型定义任意持续时间的对称周期性波形。 对于这种类型，每个发射机的波形由必需的 TW.Parameters 属性中的四个描述符和下面列出的可选 TW.equalize 定义。

TW(i).Parameters [Nx4 double] parametric parameters A,B,C,D (see text)

TW(i).equalize [double] 0 or 1(default) equalization pulses

off/on control (see text)

对于这种类型 如果 TW.Parameters 是一个 1 x 4 数组，则由下面解释的四个值定义的波形将应用于相关发射事件中的所有活动通道。或者，可以将 TW.Parameters 定义为 N x 4 数组，其中 N 行中的每一行指定每个通道的唯一波形。在这种情况下，N 的值必须与换能器使用的有效孔径大小相匹配（通常是 Resource.Parameters.numTransmit 或 Trans.numElements 中的较小者）。

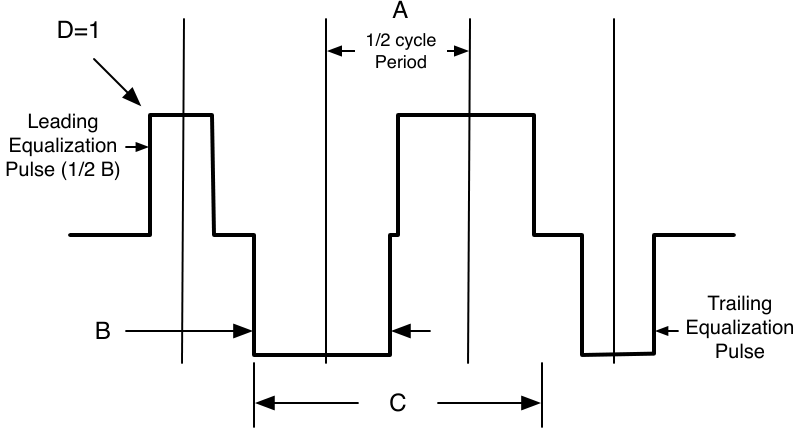
如果我们将 TW.Parameters 数组的一行中的参数值称为 A、B、C 和 D，这四个值的含义如下：

A = 发射突发频率。

B = 半周期的开启时间，以分数表示（范围 0-1）。 （见下面的注释） C = 否。 ½ 周期周期的波形（不计算可选的均衡脉冲）。

D = 设置波形第一个脉冲的极性（-1 为负，1 为正）。

所有后续脉冲的极性总是与第一个脉冲交替出现。



A 参数是一个双标量，表示 0.5 到 41.67 MHz 范围内的发射中心频率。请注意，有用的发射频率范围可能小于此值，具体取决于随 Vantage 系统购买的频率选项。实际发送半周期周期将是 250 MHz 时钟周期的整数，通过四舍五入 (125 / A) 到最接近的整数得出。因此，由于四舍五入，精确的中心频率可能与用户设置的值略有不同；系统 SW 会自动将用户指定的 A 值修改为最接近的可设置频率。没有高频配置选项的 Vantage 系统支持的可设置发射频率如表 3.2.1.1 所示。 High Frequency Vantage 系统支持并使用三个附加发射频率，25、31.25、41.67 MHz（参见第 7.1.1 节）。

B 参数指定 ½ 周期期间正驱动或负驱动的持续时间。在 ½ 周期的剩余时间内，发送驱动器输出零电平。这减少了发射到换能器的总能量，并可用于发射变迹或修改声输出以改变模式。开启持续时间表示为 ½ 周期周期的一小部分，其值范围为 0 到 1.0。例如，使用值 0.5 设置 50% 的准时。 [注意：零值不应用于关闭发送器，因为最小占空比是两个时钟。使用 TX.Apod 阵列禁用发射器。] 对于 Vantage 高频系统发射，B 参数使用中存在系统约束 - 对于 41.67 MHz 的发射波形，B 只能设置为 1.0（参见第 7.1 节。 1）。

在 V1 系统上，在通道到通道的基础上使用脉冲宽度缩放会导致发射焦点的小失真，因为在 V1 上，波形的第一个有效脉冲的前沿总是在发射延迟结束时开始- 因此，当脉冲宽度改变时，整个波形的相对相位会发生变化。对于 Vantage，我们通过始终在第一个半周期的前沿开始波形而消除了这个错误，而不管脉冲宽度如何，因此如果相对脉冲宽度小于 1.0，波形发生器将保持在零输出状态波形的开始，直到到达第一个有效脉冲的前沿。

C 参数通过指定 1/2 周期周期数来设置脉冲长度，范围从 0 到 1e07。当设置为 0 时，发送器输出被禁用。(如果发射器的 TX.Apod 值设置为 0，则无论 TW.Parameters 数组中的值如何，C 参数都会自动设置为 0。）对于长于约 32 个半周期的脉冲持续时间，系统必须具有安装的高功率选项，以及使用发射配置文件 5 生成的发射波形。其他硬件限制和散热可能会限制发射的持续时间，发射器电路的安全操作条件将由软件强制执行。

D 参数用于指定初始跃迁的极性，为 1（第一个跃迁正）或 -1（第一个跃迁负）。 TW.equalize = 1（默认值）的第一个转换是前导均衡脉冲的转换。

**支持的发射频率**

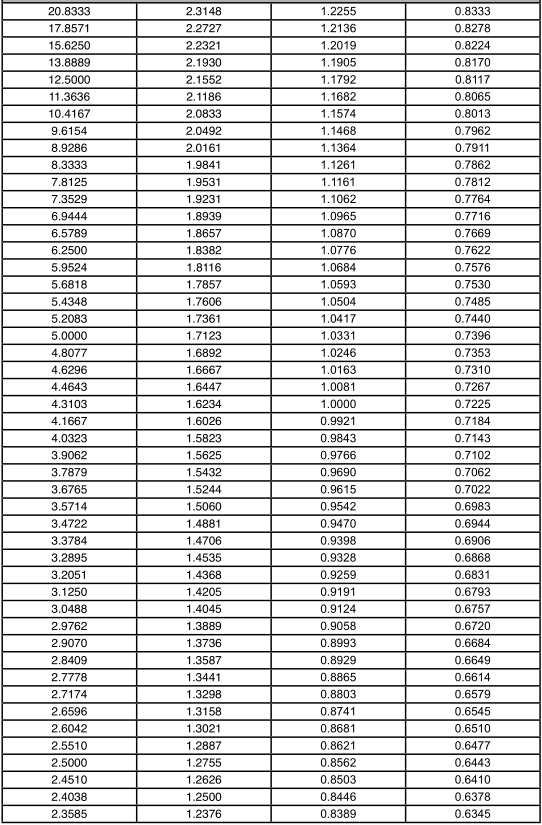


表3.2.1.1 Vantage 发射频率支持 250MHz 时钟（标准系统)。 可以使用以下Matlab 语句计算这些值：

A = (6:197)'; TxFreq = 250 ./ (2.\*A); [A, TxFreq]

TW.equalize 属性可被选择用于控制均衡脉冲，如图所示。如果未指定 TW.equalize，系统将自动分配默认值 1。TW.equalize 有三个允许值： TW.equalize = 0 禁用均衡脉冲。在这种情况下，C 的值设置了波形的整个周期，因此 C 应该是均匀的，以产生没有直流成分的波形。

TW.equalize = 1 在波形的开头和结尾附加一个均衡脉冲，如图所示。均衡脉冲的持续时间是由 B 设置的完整脉冲的 1/2。如果 C 为奇数，则两个均衡脉冲将具有相同的极性，因此对于任何 C 值，整个波形的直流含量都为零。从每个均衡脉冲的中心到相邻全脉冲的中心的时间将设置为 A，即整个波形的半周期周期，如图所示。

当 TW.equalize = 1 时，TW.equalize = 2 以与上述相同的方式附加均衡脉冲，除了在这种情况下，均衡脉冲将移动到更接近相邻的完整脉冲，以便它们之间的“关闭时间”将匹配完整脉冲之间的关闭时间，即 (A - B)。

不建议禁用均衡脉冲，因为这可能会导致发射波形带有直流偏置。带有直流分量的发射脉冲可导致时间增益接收器在发射后的恢复期较长，从而有效地掩盖了传感器附近的回波。

对于参数波形，典型用法是在设置脚本中定义单个波形，VSX 会将其复制到所有可用的发射机规格中。如果需要，可以为每个发射器定义一个单独的参数波形，但是在不同的换能器元件上以不同的频率或脉冲持续时间进行发射会导致不相干的发射波前。通常，人们会考虑改变发射器范围的唯一参数是 B 参数，它可用于发射变迹。通过减小孔径中外部发射器的半周脉冲宽度，这些元件的功率输出就会降低。正如我们将在下面的部分中看到的，TX.Apod 阵列可用于为“参数”波形执行此功能。 TX.Apod 数组值的范围可以从 0.0 到 1.0，其中 0 会关闭发射器，而与 TW.Parameters 设置无关。对于任何非零 TX.Apod 数组值，将使用关联的 TW.Parameters A、C 和 D 值，关联的 B 值将乘以 TX.Apod 值。应该注意的是，TW.Parameters 行索引与连接器 I/0 通道编号一一对应，而 TX.Apod 阵列索引则与换能器逻辑孔径中的元件相对应。这意味着TX.Apod的第n个索引中的权重可能不会修改TW.Parameters的第n行的B参数。

（另外一个原因是只为 TW.Parameters 定义单行并让系统执行任何发射变迹。）在指定参数波形类型时，唯一需要设置的 TW 属性是 TW.type 和 TW.Parameters（以及可能是可选参数 equalize 和/或 extendBL）。然后通过调用实用函数“computeTWWaveform.m”来设置其他属性，该函数还根据参数值计算数组 TW.Wvfm2Wy64Fc 中的仿真波形。事实上，如果 TW.Wvfm2Wy64Fc 或 TW.peak 属性不存在，序列加载程序 VSX 会自动调用 computeTWaveform。

'parametric' 发射波形，V1 向后兼容模式 为了向后兼容 V1 系统上使用的脚本，TW 'parametric' 波形类型还可以使用指定发射时钟周期数的 A 和 B 参数。如果软件看到 B 参数为整数值，它会将 A 和 B 参数解释为多个发送时钟周期，默认发送时钟为 180MHz。可以使用属性设置发送时钟的其他值

TW(i).twClock [double] xmit master clk {45,90,180MHz(default)}

A、B、C、D 值的解释和允许范围总结如下：

A：发射波形的一个半周期的周期，以TW.twClock设置的发射时钟频率的整数周期为单位。 A 的允许范围是 5 到 63 个时钟周期。

B：发射波形每个半周期内有效发射脉冲的周期，单位为TW.twClock设置的发射时钟频率的整数周期。 B 的允许范围是 2 到 A 时钟周期；如果 A 大于 31，则 B 缩放 2，因此 B 变量值永远不会超过 31。

C：波形持续时间，指定为构成波形的半周期的整数。允许的范围是 0:31。零值完全禁用发送输出。对于高功率选项允许的更长持续时间的发射，必须设置以下 TW 属性：

TW(i).extendBL [double] 0(default) or 1=multiply burst duration by 64

D：设置波形中第一个有效脉冲的极性。所有后续脉冲都会从该起点自动交替出现符号。允许的值只有 1 和 -1。

当 Vantage 系统在 V1 向后兼容模式下响应“参数”TW.type 时，将根据需要重新调整 V1 格式 A 和 B 参数值，以使用 250 MHz 的实际 Vantage 时钟速率产生等效波形。因此，在许多情况下，精确的发射载波频率和脉冲持续时间与 V1 上的不同，但差异通常只有几个百分点。这种转换产生的精确频率将使用新变量 TW.frequency 列在 TW 结构中，TW.frequency 是一个标量双倍，表示以 MHz 为单位的发射中心频率。通过保留每个半周期具有相同持续时间的 V1 约束，允许的载波频率被限制为 125 MHz 的整数约数 - 但此约束的好处是波形具有半波对称性，因此偶次谐波含量为零。

**3.2.1.2‘envelope’ Transmit Waveforms(' envelope '发送波形)**

“包络”发射波形类型是 Vantage 系统的一种新类型，它允许生成具有时变幅度调制的周期性波形，该调制采用脉冲波调制或时变频率来生成“啁啾”波形。

包络波形定义需要三个变量来指定波形：

TW.envNumCycles 是整个周期中整个波形的持续时间。波形的每个单独周期将是一对对称的正负半周期脉冲，其持续时间完全相同，确保生成的波形中没有直流甚至谐波成分。 TW.envNumCycles 是一个双精度标量，其整数值必须至少为 1 且不大于 10,000。

TW.envFrequency 是一个长度必须等于 TW.envNumCycles 的行向量。每个条目都是一个双精度值，以 MHz 为单位表示相关周期的频率，允许的范围为 0.5 到 32 MHz。频率值将被修改以表示相关波形的最接近的整数 250 MHz 时钟周期。

TW.envPulseWidth 是一个长度必须等于 TW.envNumCycles 的行向量。每个条目都是一个双精度值，表示相关发送周期的两个半周期的相对脉冲宽度。每个条目的允许范围是从 0.0 到 1.0。实际脉冲宽度将根据为该周期指定的相对脉冲宽度和频率四舍五入到最接近的 250 MHz 时钟周期整数。可以指定负值来反转相关循环的极性（如果 TW.envPulseWidth 值为正，则前半个周期将为正，如果 TW.envPulseWidth 值为负，则前半个周期将为负）。

**3.2.1.3‘pulseCode’ Transmit Waveforms( ' pulseCode '发送波形)**

“pulseCode”类型允许用户通过直接指定 TW.PulseCode 数组来明确定义任意发射波形。这是 Vantage TW 结构中的一个新变量，用于提供用于指定发射波形的通用机制。对于这种类型，TW.PulseCode 是唯一需要的输入变量，系统将按原样使用。

TW.PulseCode 数组由 computeTWaveform 函数生成，以响应用户提供的任何 TW.type 定义。 TW.PulseCode 然后被 update 和 arbwaveCompile 函数用于生成将提供给硬件的实际波形描述符表，并且也被 computeTWaveform 用于生成用于仿真的 TW.Wvfm2Wy64Fc 数组。 TW.PulseCode 还旨在提供一种易于访问且直接的格式，可以根据用户的需要轻松用作其他波形分析功能的输入。

TW.PulseCode 是一个双精度值的 Matlab 数组，大小 (N x 5) 用于将应用于所有通道的单个波形定义，或者可选大小 (N x 5 x numTXchannels) 用于独特的波形定义每个频道。阵列中的 N 行中的每一行都指定了一个有效的发射“模式”或波形段。行中定义每个段的五个值，此处称为 [Z1, P1, Z2, P2, R]，定义如下：

**Z1** 是一个非负整数值（范围 0-1023），表示在有效脉冲 **P1** 之前波形保持在零状态的 250 MHz 时钟周期的数量。

**P1** 是一个有符号整数（范围 0-175），定义了 **Z1** 零状态间隔之后的有效输出脉冲。 脉冲的极性由 P1 的符号设置，250 MHz 时钟周期的持续时间由 **P1** 的绝对值设置。

**Z2** 定义了有效脉冲 **P1** 和 **P2** 之间波形的零输出状态的持续时间，使用上面对 **Z1** 给出的相同规范。

**P2** 定义了一个跟随 **Z2** 的有效输出脉冲，使用上面为 **P1** 给出的相同规范。

**R** 是一个正整数（范围 1-10e07），指定由 [**Z1 P1 Z2 P2**] 定义的波形段重复的次数。将 **R** 设置为零会以 [**Z1 P1 Z2 P2**] 来终止波形输出，该行将被丢弃。

请注意，任何 **Z** 或 **P** 值都可以设置为零，从而导致该波形元件的持续时间为零。 如果 **R** 值设置为零，则表示波形结束，TW.PulseCode 数组中的所有后续条目都将被忽略。 如果数组第一行的 **R** 值为零，则定义了一个完全不产生输出的波形。 在终止波形的任何行中，**R** 的值必须为零。 如果 **R** 设置为零波形终止行，则可以省略它是定义数组的最后一行。 **P1** 或 **P2** 值为零不产生波形输出（对于均衡脉冲隔离和重复段内的一种波形模式很有用）。

**3.2.1.4 Simulation Transmit Waveforms(仿真发射波形)**

使用模拟模式 (Resource.Parameters.simulateMode = 1) 运行序列时，允许波形类型为“parametric”、“pulseCode”、“function”或“sampled”。 基于周期数和总持续时间，模拟模式下的发送波形长度存在一些限制。 通常，仿真波形应限制在 25 个周期或更少。

对于“函数”波形类型，波形是根据以下语句计算的加窗正弦曲线：

n = TW.numsamples;

k = TW.samplesPerWL = 64;

T = (0:n-1)’;

TW.Wvfm1Wy64Fc(1:n) = D\*128\*Window.\*sin(2\*pi\*T/k);

其中D为极性描述符(+1或-1)，Window为广义余弦窗，定义为:

Window = A – B\*cos(T\*2\*pi/n) + C\*cos(2\*T\*2\*pi/n);

在这种情况下，脉冲长度由 TW.numsamples 属性设置（例如，对于三周期脉冲，将 TW.numsamples 设置为 192）。根据模拟代码的要求，TW.samplesPerWL 值应为 64。

流行的窗函数有 Hamming (A=0.54, B=0.46, C=0)、Hann (A=0.5, B=0.5, C=0) 和 Blackman (A=0.42, B=0.5, C=0.08)。对于“function”类型，设置任何描述符值都会调用“computeTWaveform”计算方法来生成 TW.Wvfm1Wy64Fc 和 TW.peak 数据。 256 的幅度值对于模拟器来说有些随意，但如果使用 16 位射频数据，则该值不应更高，因为可能会出现饱和。

对于与仿真模式一起使用的“函数”波形类型，只需要一组描述符，因为当前版本的仿真不支持单个发射机的不同频率或突发长度规范。在这种情况下，TX.Apod 阵列可以通过使用范围从 0 到 1.0 的各个发射器的加权值来提供变迹功能。

最后，模拟器也可以使用“采样”波形类型。在这种情况下，TW.Wvfm1Wy64Fc、TW.Wvfm2Wy64Fc、TW.numsamples、TW.samplesPerWL 和 TW.peak 属性必须由用户在设置脚本中计算。根据模拟代码的要求，TW.samplesPerWL 属性应始终设置为 64。

对于所有波形类型，TW.peak 属性提供从发射波形起点到峰值幅度点的波长距离。这需要指定发射脉冲离开发射元件的有效时间，这是通过将 TW.peak 添加到元件的延迟值（以波长为单位）来给出的。对于对称发射波形，TW.peak 通常等于 ½ 脉冲串长度。

**3.2.2 TX Object(TX对象)**

TX对象包含定义传感器孔径中每个有源元件的发射波形和定时所需的所有信息。在非多路复用传感器的情况下，假设变送器的数量M大于或等于传感器元件的数量N，并且传感器元件以一对一的方式连接到变送器（元件1连接到变送器1等）。对于元件未与连接器输入/输出通道一一连接的情况，变速器。连接器阵列指定映射，允许将元件编号视为与发射器编号一一映射。发射周期从采集事件开始时开始，表示时间0。各个发射机开始倒计时其延迟时间，延迟结束时，生成发射波形。

TX对象的主要属性如下所示。在调用实用程序函数“computeTXDelays”计算发射延迟值之前，需要突出显示的属性。

TX =

waveform: double number of a transmit waveform structure.

Apod [1xM double] (see notes)

aperture double (see notes - only needed for muxed xducers)

Origin [1x3 double] effective origin of transmit beam at xducer

focus double Focal distance in wavelengths (0=flat).

Steer [1x2 double] Beam steering angles in radians (θ,α).

Delay [1xM double] delay times for xmitters in wavelengths

TXPD [PData.Sizex3] transmit pixel data

peakCutOff double used to qualify pixel intensities dflt=1.25

peakBLMax double used to qualify pixel burst lengths

VDASApod [1xN double] translated TX.Apod array for hardware

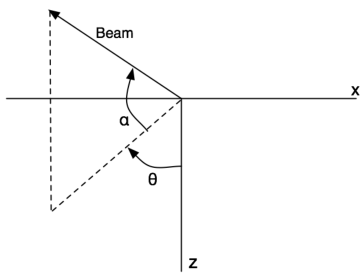
VDASDelay [1xN double] translated TX.Delay array for hardware

TX.waveform 属性是 TW 对象数组的索引（参见上一节），这些对象定义了每个活动发射器要生成的发射波形。发射波形通常是换能器元件中心频率的几个周期，这些周期在延迟周期之后被施加到元件。

如果发射事件不需要所有发射器，则可以使用 TX.Apod 阵列通过为未使用的发射器指定零值来选择可用发射器的子集。通过使用范围从 -1.0 到 1.0 的 TX.Apod 值，还可以指定跨发射孔径的切趾加权函数，负值反转波形极性。使用 VDAS 硬件时，TX.Apod 阵列既可用作开/关指示器，0 值关闭发射器，非零值启用它，也可用于使用发射波形占空比控制的发射变迹。当 TX.Apod 阵列的绝对值介于 0.0 和 1.0 之间时，使用脉冲宽度调制来修改发射波形以降低发射的效应功率。例如，对于参数波形，TW.Parameters B 值在加载到硬件时会被修改，以设置最接近的可设置 ½ 周期占空比。由于最小 B 值为 2 个发送时钟，因此将 TX.Apod 值设置为小于 2/T （其中 T 是发送波形的半周期内的发送时钟数）对最小切趾没有影响。如果 TW 结构定义了每通道波形的数组，那么这些波形将按原样使用，并且它们的相对脉冲宽度不会由 TX.Apod 缩放。相反，TX.Apod 将仅被解释为逻辑值 - 零将禁用通道的发送输出，但任何非零值将导致使用该通道 TW.PulseCode 完全按原样。

TX.aperture 属性仅适用于在探头主体中包含高压多路复用器的传感器，该多路复用器用于选择全阵列孔径的子孔径。 TX.aperture 的值指定在 Trans.HVMux.Aperture 数组中定义的孔径。当加载程序 VSX 加载脚本时，使用没有相应 Trans.HVMux.Aperture 定义的 TX.aperture 属性将导致错误。 TX.aperture 的值是指 Trans.HVMux.Aperture 数组的第二个索引。还应该注意的是，当使用 TX.aperture 属性选择子孔径时，TX.Apod 值线性应用于子孔径的元件，即使实际换能器元件可以随着每个 TX.aperture 选择而改变。

对于典型的发射波束，属性 Origin、focus 和 Steer 足以定义波束特性。原点对于聚焦光束 (TX.focus > 0) 和虚拟顶点光束 (TX.focus < 0) 是必需的，并指定了光束似乎源自的换能器元件表面上的点。对于线性阵列，这通常是发射光束的中心与换能器元件的 x、z 平面相交的地方，这通常是一维阵列的发射孔径的中心（发射光束靠近换能器孔径的外边缘）。焦距是从换能器上的原点到光束到达焦点的距离（以波长为单位）。正焦点值定义了多元件发射波束的会聚点。焦点值 0 指定穿过发射孔径的平坦波前，而负焦点值指定发散光束，其有效源在负焦点值处沿 -z 轴（无转向）。转向角指定光束与原点处换能器平面的角度。 θ 指定光束从正 z 轴（方位角）投影到 x,z 平面的角度，α 指定光束相对于 x,z 平面的角度（仰角）。



如果给定了 TX.Apod、TX.focus 和 TX.Steer（还有 TX.Origin 用于 TX.focus~=0），则可以调用“computeTXDelays”方法来计算各个发射延迟。此功能还需要访问 Trans，即指定传感器几何形状的全局对象（之前定义）。该方法计算所有具有非零 TX.Apod 值的元件的 TX.Delay 值。

重申一下，如果 TX.focus 设置为 0，这意味着一个平坦的聚焦光束，对于线性阵列和零度转向，所有发射器将同时启动。在这种情况下，可以将光束的原点设置为光圈的中心。非零转向角会增加发射延迟的线性斜率。如果正确指定了换能器上的原点和转向角，“computeTXDelays”方法也将与虚拟顶点类型的发射函数一起使用。另一种类型的发射模式是用于闪光发射成像的发散光束，这可以通过使用 TX.focus 的负值来指定。

对于所有其他发射延迟模式，延迟必须由用户计算并以波长写入 TX.Delay 阵列。当转换为微秒时，TX.Delay 条目的最大值为 45.5 微秒。例如，5 MHz 中心频率传感器的波长周期为 200 纳秒。在这种情况下，TX.Delay 的最大波长设置为 45.5e-06/200e-09 = 227.5 个波长。此外，还必须计算换能器面处发射波束的有效原点。在不使用“computeTXDelays”方法的任意延迟模式的情况下，TX.focus 和 TX.Steer 的属性将被忽略。

**3.2.2.1 TX.TXPD Data(TX.TXPD数据)**

TX 对象的 TXPD 属性保存有关 PData 数组中定义的像素位置处的模拟发射场的信息。提供了一个用于计算字段参数的函数，称为 computeTXPD。其用法如下：

TX.TXPD = computeTXPD(TX, PData);

输出是一个 uint16 值数组，其中三个索引用于 2D，四个索引用于 3D 体积。前两个或三个索引与 PData.Size 的前两个或三个索引相同，而最后一个索引可以是 1、2 或 3，并指定计算的参数。计算的参数是 1) 像素位置的峰值发射强度（单通道发射等效值）； 2) 峰值透射出现在像素位置的时间（等效波长）； 3) 发射突发的长度，如在像素位置所见（以波长为单位）。

如果添加到 TX 结构，则引用 TX 结构的重建（通过 ReconInfo.txnum 属性）将使用 TXPD 数据处理重建，并执行以下操作：1）重建输出强度将在每个像素位置进行归一化根据像素处的峰值透射强度。

2) 从发射突发源到像素位置的峰值强度时间的传播时间将使用 TXPD 计算而不是算法近似。

3) 仅重建峰值强度大于 TX.peakCutOff（发射等效值）和像素位置处的突发长度小于 TX.peakBLMax 的像素。如果未提供，这些新的 TX 属性分别默认为 1.25 和（最小发射突发长度 + 0.25（波长））。

可以添加 TXPD 属性以促进某些类型的扫描序列的图像重建，特别是那些涉及多个重叠发射波束的扫描序列。 在这些序列中，重叠波束内的像素可以被重建多次，即，对于每个使像素发出声波的发射/接收事件，重建一次。 可以组合多个重建，每个像素重建通过其发射光束强度归一化，以在每个像素处产生合成发射焦点。 更多信息将在第 3.4 节中提供，包括图像重建。

**‘showTXPD’**——此功能将自动将 TXPD 属性添加到工作场所的 TX 结构中，并打开一个窗口以可视化字段。 在 Matlab 工作场所执行示例脚本以计算脚本的结构（例如“SetUpL7\_4FlashAngles.m”）脚本后，可以键入

>>showTXPD

在Matlab命令行中。发射字段将被计算，并将创建如下图所示的窗口。



图3.2.2.1.1、使用showTXPD对TX的发射域进行可视化

**3.2.2.2 TX VDAS parameters(TX VDAS参数)**

TX.VDASApod 和 TX.VDASDelay 属性由加载函数 VSX 或处理函数 runAcq 设置，不需要用户设置。 它们用于将逻辑用户 TX 属性映射到对 VDAS 系统进行编程所需的硬件特定参数。 例如，TX.Apod 阵列值适用于换能器孔径中元件的线性范围，无论该孔径如何映射到连接器 I/O 通道。 TX.VDASApod 数组值派生自 TX.Apod 值，但置换后与 I/O 通道具有直接的一对一关系。

VDAS 参数在结构定义中可见，以帮助调试和用于硬件的自定义配置。

**3.2.3 Transmit Power Controller (TPC,发射功率控制器)**

发射功率控制器或 TPC 为 VDAS 模块上的发射器提供高压电源。其范围为 1 - 96 伏，可使用 VSX GUI 窗口上的高压滑块对象进行设置。由于发射器是双极输出设备，因此实际发射的峰峰值电压是设置电平的两倍。也可以通过在滑块下方的文本框中键入一个值来设置高电压电平。出于安全原因，在加载新的序列脚本时，高压电平始终设置为最小值，并且必须在运行期间调整到所需的电平。

TPC 提供高压输出的快速变化，允许为不同的采集模式使用不同的功率水平。每个功率级别都可以有自己的一组独特属性，包括发射驱动电压级别的规格和最大功率输出。给定功率级别的属性集合称为配置文件，当前 TPC 固件支持多达四个配置文件，第五个可用作“高功率”选项。

配置文件使用 TPC 结构指定，结构索引对应于配置文件编号。请注意，配置文件 1 到 4 的所有 TPC 属性都是可选的，如果未设置，则使用默认值填充或设置为空。如果未提供 TPC 结构定义，则为所有使用的配置文件编号提供默认定义。这意味着对于大多数使用模型，根本不需要定义 TPC 结构，并且可以简单地请求在事件列表中的适当时间使用不同的配置文件编号。 TPC结构有如下定义：

TPC =

name string (optional) name identifier for profile

maxHighVoltage double (optional) max high voltage for this profile

highVoltageLimit double (optional) high voltage limit based on use model

xmitDuration double (optional) longest transmit duration (usec)

TPC.name 是一个可选字符串，可用于标识配置文件。例如，TPC(1) 可能用于 2D 成像，而 TPC(2) 可能用于多普勒。在这种情况下，我们可以设置 TPC(1).name = ‘2D’ 和 TPC(2).name = ‘Doppler’ 用于识别目的。默认情况下，此字段设置为空数组。

TPC.maxHighVoltage 值设置可为配置文件设置的最大高压电平，并可用于设置低于 Trans.maxHighVoltage 中设置的电平的电压。如果 TPC.maxHighVoltage 电平高于 Trans.maxHighVoltage 电平，则 Trans.maxHighVoltage 电平优先。使用 TPC 配置文件 5 时，必须为 TPC 结构提供用户设置的 TPC.maxHighVoltage 级别。 TPC.maxHighVoltage 通常设置为固定的绝对最大允许电压，这取决于系统硬件和换能器的能力，或者换能器的最大声输出强度限制，并且在序列运行期间不会改变。

TPC.highVoltageLimit 参数还限制了可以使用的最大高压设置，但它的用途与 TPC.maxHighVoltage 略有不同。 TPC.highVoltageLimit 旨在支持更动态的限制，该限制可能会随着系统操作状态的变化而变化，并且可能比 TPC.maxHighVoltage 更具限制性。例如，在 PW 多普勒脚本中，用户可以控制多普勒 PRF（pulse repetition frequency，[脉冲](https://baike.baidu.com/item/%E8%84%89%E5%86%B2/1938481?fromModule=lemma_inlink" \t "_blank)重复频率）和发射脉冲串持续时间，TPC.highVoltageLimit 可以由声学输出控制函数设置，以保持对脉冲串持续时间和任意组合的恒定输出功率限制。公积金。在某些情况下，这个与状态相关的限制可能比 TPC.maxhighVoltage 更严格，而在其他情况下可能不会。在系统中实际设置发射电压的低级 SW 函数将始终强制执行来自 Trans.maxHighVoltage、TPC.maxHighVoltage 和 TPC.highVoltageLimit 的最严格的值。

TPC.xmitDuration 属性指示发射周期的持续时间，并且应该设置为包括最长的 TX.Delay 值加上发射突发的持续时间。这向控制器指示何时可以安全地开始转换到下一个配置文件，这可能发生在采集事件的接收期间。如果未提供，则该属性默认为采集周期的整个长度。

**3.2.3.1 Managing TPC Profiles in a Sequence(顺序管理TPC配置文件)**

默认情况下，如果在用户序列中未指定配置文件，则使用的默认配置文件为配置文件 1。当序列中使用两个或多个 TPC 配置文件时，会在 VSX GUI 窗口中创建一个额外的高压滑块，用于控制第二个配置文件.如果具有高功率选项的配置文件 5 是附加配置文件之一，则第二个滑块将控制其级别；否则，第二个滑块将控制在配置文件 1（默认配置文件）之后指定的第一个附加配置文件。如果用户想要拥有第三个配置文件级别的滑块控制，他们将不得不提供自己的 GUI 控制元件。

从一个配置文件切换到另一个配置文件在事件序列中通过使用 SeqControl 命令进行控制（SeqControl 对象在第 3.6 节中描述）。用户负责指示新配置文件何时生效，并为从当前配置文件过渡到新配置文件提供足够的时间。此外，用户必须考虑在发送事件期间补充消耗功率的时间，以便指定配置文件的发送总线在下一次发送事件之前完全恢复。

用户可以通过将“setTPCProfile”的 SeqControl 命令置于采集事件（同时具有发送和接收规范或仅发送规范的事件）中指定从一个配置文件到另一个配置文件的更改，条件为“下一个” ' 用参数中的配置文件编号指定。在 TPC(n).xmitDuration 周期结束时，如果指定，或者如果没有指定，则在采集周期结束时，TPC 将开始更改为新配置文件的电压电平。

从一个配置文件切换到另一个配置文件可能需要 800 微秒到 6 到 8 毫秒，具体取决于新配置文件的高压电平与旧配置文件的电平相差多远。此时间段应计入使用不同配置文件的采集事件之间的时间（参见“timeToNextAcq”SeqControl 命令的用法）。如果序列中的下一个发射事件被编程为在 TPC 完成从一个配置文件到下一个配置文件的转换之前发生，则该事件将延迟到转换完成。如果使用“timeToNextAcq”SeqControl 命令安排下一个事件，则会发出错过“timeToNextAcq”周期的警告，表明为转换分配的时间不足。此警告可用于微调序列以允许有足够的时间进行转换。

某些序列可能发生的另一种类型的错误是 TPC 未能达到配置文件的设置级别。这可能发生在某些高 PRF 序列中，其中从 TPC 汲取的功率大于 TPC 恢复设定电平的能力。在这种情况下，会生成一个停止序列的 TPC 错误。

在某些情况下，希望 TPC 配置文件在序列中的某个点立即生效，对于此操作，可以使用“setTPCProfile”命令的“immediate”条件。此选项应在非采集事件中使用，这意味着 Event.tx 和 Event.rcv 都为 0。在这种情况下，将立即开始转换到参数字段中指定的配置文件。再次，用户负责在下一个发射事件之前为配置文件转换留出足够的时间。 （在这种情况下，必须使用“noop”命令，因为“timeToNextAcq”命令只能在采集事件中使用。）在带有“next”的非采集事件中使用“setTPCProfile”命令指定的条件将延迟配置文件转换的开始，直到序列中的下一个采集事件之后。

**3.2.4 High Power Transmit (HIFU,高功率发射)**

VDAS 系统有两种可选配置，允许在非常高的发射功率水平下使用该系统，以支持使用长发射序列 (“HIFU”) 的治疗应用，以及用于刺激的短持续时间“推送”发射事件用于其他目的的组织。对于这些选项中的任何一个，都使用一个专用的独立电源进行大功率发射。通过选择 TPC 配置文件 5 来选择此电源，而配置文件 1 到 4 始终使用“成像”发射电源。这两个选项都可用于两板（128 个发射通道）或四板（256 个发射通道）配置。下面分别介绍这两个选项。

“扩展发射选项”使用容量为 48 瓦的专用内部电源为 HIFU 发射突发供电。系统内提供的大型储能电容器（称为“推式电容器”）允许以更高的功率水平发射短脉冲，前提是脉冲之间有足够的时间从内部电源为电容器充电。可用发射功率的大小在很大程度上取决于发射突发持续时间、活动发射通道的数量以及换能器提供给系统的负载阻抗。通常，在长达几毫秒的突发持续时间内，可以实现从几百瓦到两千瓦的发射功率水平。

“HIFU 选项”增加了一个连接到系统的外部 HIFU 发射电源。对于突发持续时间和重复间隔的任何所需组合，此选项支持高达 1200 瓦的发射输出功率水平，在 1 至 5 MHz 范围内的任何发射频率下。安装此选件后，典型传感器负载阻抗为 50 欧姆，每个发射通道可在 1 - 5 MHz 频率范围内提供高达 8 瓦的连续输出功率水平。

成像模式可以根据需要在 HIFU 发射序列之间交错，利用 VDAS 系统为成像和多普勒模式提供的全部功能和性能。对于 Extended Burst 和 External HIFU 选项，TPC Profile 5 用于选择高功率发射电源，而 TPC Profile 1 到 4 仍可以正常方式用于成像。这允许 HIFU 突发和成像的发射电压电平完全相互独立地控制。发射脉冲串频率、持续时间、有效孔径、焦点等也可以为每个发射脉冲串独立编程。

**3.2.4.1** **Controlling the External Power Supply（控制外部电源）**

**注意:**本节仅适用于配置了HIFU Option的系统。

作为HIFU选项的一部分，该系统提供了一个1200瓦容量的外部OEM电源。外部电源通过增加到系统后面板的连接器连接到VDAS系统发射电路。另外，SW远程控制链路连接VDAS系统的上位机电源。通过这个远程控制链接和支持系统SW，用户可以从Profile 5高压滑块GUI控制外部电源，与使用Profile 5内部辅助电源的Extended Burst选项的系统完全相同。

以下是设置和使用外部电源时用户应注意的事项:

连接到系统:电源实际上提供两个完全独立的600瓦输出。为了达到我们的目的，我们将这两个输出并联，并在专门用于并联连接的“电压跟踪-电流共享”模式下操作电源。从系统到电源的电缆组件有四根线，两根红两根黑。这些应该直接连接到电源后面的四个输出端子，匹配红对红，黑对黑(两根相同颜色的电线连接到两个端子中的哪一个并不重要)。两个输出的直接并行连接发生在VDAS系统内部，而不是在电源上。这确保了通过电缆和后面板连接器引脚更均匀的电流共享。确保电源连接是紧的(你可以手工做的最紧)。

**source.HIFU. extpwrcomportid:**与external HIFU选项一起使用的外部电源的SW控制是通过“虚拟串口”实现的，当电源的驱动程序安装在用户的计算机上时，该“虚拟串口”是由Windows操作系统分配的;物理接口通过系统自带的USB数据线连接。一旦安装了电源驱动程序(参考电源附带的说明书和CD)，它将在计算机上被标识为端口号，例如“COM3”。在用户的设置脚本中，这个字符串值必须分配给变量"Resource.HIFU.extPwrComPortID"。当VSX执行脚本时，这个变量的值将被传递给实际与电源通信的例程。

**System Startup（系统启动）:**在尝试运行将使用配置文件5的脚本之前，请确保电源已连接并打开。如果电源未实际连接，或者系统由于任何原因无法成功与其通信，则脚本的执行将终止，并将报告致命错误。如果发生这种情况，请检查所有连接并重试。您还可以在主机上使用“设备管理器”，以确保它将电源识别为已连接的设备，并且com端口ID号与您在Matlab脚本中指定的端口号匹配，如上所述。请注意，如果您要运行一个根本不调用配置文件5的成像脚本，则无需打开或连接电源。

**前面板控制:**如果远程控制接口已初始化且功能正常，则电源前面板控制将被禁用。不要忽略这一点，并试图从其前面板控件手动控制电源;这样做很容易导致系统处于自毁状态。

**3.2.4.2 Using the High Power TPC Profile（使用高功率TPC配置文件）**

在大多数情况下，使用扩展突发或外部HIFU选项为“HIFU”发射事件对系统进行编程与为成像采集进行编程没有区别，这将在本文档的其余部分中进行解释。同样的' setTPCProfile ' SeqControl命令用于从事件序列中选择高功率配置文件，参数设置为5。当高功率配置文件与用户脚本中的成像配置文件编号1 - 4一起使用时，在GUI中打开的第二个高压滑块控制高功率电压水平。在本例中，对于配置文件5滑块，在指示设置高电压水平的正常文本框的右侧创建了一个额外的文本框。此附加文本框报告储能“推”电容器上的平均测量电压水平。如果正在运行的序列消耗的存储电容超过充电的补充，所测电压将下降。当电压下降到设置水平的80%时，文本背景变为蓝色，提请注意问题。如果发生这种情况，则必须修改序列以发射更少的功率，和/或允许更长时间的电容再充电。

在使用配置文件5进行高功率发射时，用户必须了解其他几个系统约束和控制参数，如下段所述:

**P5ena：**这个参数不应该在用户的安装脚本中指定;它是由VSX在Matlab工作区中自动定义和设置的，当一个脚本被执行时，它表示系统的实际HW配置和运行状态。运行VSX后，用户可以在工作空间中查看P5ena的值，以确认系统的实际运行状态:—如果P5ena值为0，表示配置文件5没有被用户的安装脚本调用。

—取值为1时，表示系统已安装并正在使用Extended Transmit选项。

—取值为2时，表示HIFU Option已安装并正在使用。

如果一个脚本试图选择TPC配置文件5，但VSX确定HW配置不支持它，P5ena将被设置为0，脚本的操作将被终止，并报告一个致命错误。

**Resource.HIFU.externalHifuPwr**该参数必须在打算使用外部HIFU选项的用户脚本中指定并设置为值1。如果系统被配置为External HIFU选项，并且脚本调用配置文件5，如果'Resource.HIFU. Option '，操作将被终止，并发出致命错误消息。externalHifuPwr'缺失或未设置为1。对于不使用配置文件5的成像脚本以及使用Extended Burst选项的任何脚本，根本不需要指定此参数。此参数的唯一目的是帮助确保用户意识到他们的脚本将使用1200瓦外部HIFU电源。如果一个使用配置文件5的脚本在模拟模式下执行，没有HW存在，这个参数的状态将被VSX用来设置profile5ena

为1或2(参见上面的'profile5ena')，因此在模拟过程中，极限检查算法将仍然知道脚本是打算使用扩展爆发还是外部HIFU选项。

**timeToNextEB:**这是一个专门的序列控制(SeqControl)命令，它只能用于利用配置文件5和扩展突发长度的发射事件。这使得在定义配置文件5“推送”发射事件与成像数据采集事件之间的事件序列时具有更大的灵活性。它还可以用于对5型发射的占空比施加最大限制，从而允许在突发期间在更高的电平上运行。'timeToNextEB'命令的指定和使用方式与'timeToNextAcq'命令相同，不同的是'timeToNextEB'只能在配置文件5扩展突发事件中使用，并且它将延迟时间设置为下一个配置文件5扩展突发事件，忽略中间出现的任何成像突发。例如，假设在您的脚本中有一个Extended Burst发射事件，接下来是一系列成像事件，然后是一个空闲时间，然后再跳回去重复这个序列。您可以将timeToNextAcq和timeToNextEB命令放在配置文件5扩展突发长度发射的事件中。timeToNextAcq将把延迟设置为下面的成像事件，而timeToNextEB将把总体延迟设置为概要5发射的重复(即，它将设置为成像采集时间加上重复序列之前所需的额外延迟的总和)。

**Resource.HIFU.voltageTrackP5:**系统允许使用来自Profile 5电源的正常短时间成像发射突发，这在某些情况下是方便的，因为它避免了系统在一个发射事件和下一个发射事件之间执行概要转换所需的开销时间。但为了获得最佳的成像性能(特别是多普勒等应用，需要非常低的噪声和非常好的发射电源稳定性)，建议用户切换回成像配置文件。在优化成像性能和最小化剖面过渡时间这两个相互冲突的目标之间，成像剖面电压可以保持与配置文件5相同。这将profile切换时间减少到小于500 usec。为了方便用户实现这一概念，可选的Resource.HIFU.voltageTrackP5参数已定义。如果在用户的设置脚本中指定了此参数，并将其设置为成像配置文件的索引值（即值1、2、3或4），则将禁用该配置文件的GUI控制，而是由配置文件5的GUI自动控制。因此，两个配置文件将始终具有相同的电压设置，从而最小化配置文件转换时间。如果未指定此参数或将其设置为零值，则将禁用跟踪功能。

**转换到配置文件5:**正如3.2.3.1小节所解释的，用户有责任确保在事件序列中允许有足够的时间来完成概要转换，通常是通过包含'noop'、'timeToNextAcq'或'timeToNextEB'序列控制命令。过渡到轮廓5比其他轮廓过渡稍微复杂一些，因为在这种情况下，HW必须物理上关闭连接专用发射电源到发射电路的开关。当这个开关仍然处于转换状态时，系统执行一个发射事件可能是自毁的。正因为如此，如果它发生在概要文件5转换完成之前，系统HW实际上会延迟发射的开始。不会报告错误，但如果发射延迟导致错过了' timeToNextAcq '，则会发出相应的警告。(如果发射活动在转换到除5之外的任何概要文件之前开始，则在Matlab命令提示符中显示一条警告消息，允许系统执行继续)。

TPC配置文件初始化:当配置文件5将在脚本中激活时，系统还需要在第一个发射事件之前执行'setTPCProfile'命令，以避免任何可能的关于系统状态的模糊性。为了满足这一要求，通常可以用一个非采集事件和' setTPCProfile '命令启动一个序列，参数设置为所需的初始配置文件和'立即'条件，然后是一个很长的' noop '(几毫秒)，以允许达到高电压水平。然后可以进入序列的正常操作，在序列的末尾跳转到后面的起点。

**3.2.4.3 Transmit Circuit Limit Checking（发射电路限制检查）**

在与长发射突发持续时间和/或高发射占空比相关联的高功率电平下，系统内发射组件的功耗能力在某些情况下可能会被超过，从而导致对系统的永久损坏。 为了避免这种情况，系统中提供了限制检查功能。 每次执行用户设置脚本时，以及每次在脚本运行时修改发射参数时，都会自动调用此函数。 如果检测到可能对系统造成自毁的操作状态，则会向用户显示警告，并且将发射功率降低到安全水平或终止脚本的操作。

**注意:** 本节中描述的限制检查算法和功能的唯一目的是保护 VDAS 系统本身免受组件损坏。 用户必须意识到，对系统来说“安全”的操作状态可能仍然会输出足够的功率来破坏他们的传感器！ 用户有责任根据需要定义和施加额外的限制，以保护换能器免受损坏，并应用预期应用可能需要的任何声输出功率限制。 本节中描述的一些功能和控制参数可用于促进可能需要的任何额外输出控制的实施。 用户必须注意的另一个因素是 Vantage 系统变送器的功能本质上是一个源阻抗非常低的电压源； 因此，实际输出功率和发射电流在很大程度上取决于换能器提供给系统的实际负载阻抗。 该值必须由用户在参数 Trans.impedance 中指定。 不正确的跨阻抗值可能会使系统和换能器遭受潜在的严重损坏。

限制检查功能通过单独的 Matlab 函数“TXEventCheck.m”提供。 此功能位于 Vantage 发布目录中的“HIFU”子文件夹中。 整个 TXEventCheck.m 文件中的注释解释了每个单独的限制以及如何评估它们。

TXEventCheck 使用许多系统参数与系统的其余部分进行通信。 在使用配置文件 5 执行脚本后，所有这些参数都将保留在 Matlab 工作区中，用户可以在其中查看它们以帮助了解限制以及如何调整发射操作状态以获得最佳性能，同时保持在这些限制内。 用户可以设置其他参数来控制系统操作的某些方面。 其中一些参数和相关功能定义如下：

**TPC(5).highVoltageLimit 和TPC(5).currentLimit：**TXEventCheck 使用这两个参数来传达为维持系统硬件的安全操作状态而必须施加的发射输出限制。 实际控制配置文件 5 发射电源的功能从 TPC 结构中读取这些值，并在发射电压设置开始更改时强制执行它们。 电流限制值 TPC(5).currentLimit 仅适用于与 External HIFU Option 一起使用的外部电源； 对于扩展突发选项，内部辅助电源自我限制在大约 0.5 安培的最大输出电流，独立于 TXEventCheck。

**Resource.HIFU.verbose：**当它处于活动状态时，限制检查算法会通过 matlab 命令窗口显示大量状态和警告消息。 在开发和测试设置脚本时，这些对用户来说非常有价值，但在运行已经经过全面测试的脚本时，它们可能会非常烦人和具有破坏性。 如果需要，用户可以在其设置脚本中为 Resource.HIFU.verbose 分配零值以抑制警告消息。 （但是，将始终显示错误消息报告）。

**TXEvent 和 TXthermal：**这些是由 TXEventCheck 创建并写入 Matlab 工作区的数据结构。 它们包含事件序列中标识的每个配置文件 5 扩展突发长度发射事件的条目。 对于这些事件中的每一个，它列出了已由限制检查算法识别和评估的所有操作状态参数的值。 查看这些结构可以帮助用户深入了解哪些参数对他们在脚本中定义的操作状态的限制最大。 TXEvent结构体中列出的参数的定义和单位参见TXEventCheck.m文件中的代码和注释。

**Resource.HIFU.TXEventCheckFunction：**为了针对特定应用和换能器调整系统以实现最佳操作，在极少数情况下，用户可能会发现需要修改或增强限制检查功能“TXEventCHeck.m”的某些方面。 为了促进这一点，系统提供了一种方法来替换不同的文件名来代替作为 Verasonics 软件版本的一部分提供的 TXEventCheck.m 函数。 要利用此功能，请复制 TXEventCheck 并为其指定一个不同的名称。 然后将所需的更改编辑到重命名的副本中。 最后，将新名称作为字符串分配给需要使用修改函数的设置脚本中的变量“Resource.HIFU.TXEventCheckFunction”。 VSX 将使用分配给此变量的值作为调用限制检查时要调用的函数的名称。 但是请注意，只有在咨询 Verasonics 后才能对 TXEventCheck 进行任何更改或添加。 如果用户损坏了他们的系统，并且确定对限制算法进行了未经授权的修改，Verasonics 将不承担任何维修费用，即使系统仍在保修期内或服务合同期内。

* 1. **Receive Objects（接收对象）**

接收对象定义了获取事件接收阶段的所有特征。 在 Vantage 硬件和仿真软件中，TX 和 Receive 周期从同一时间点或时间 0 开始。这意味着随着发射器对其延迟周期进行倒计时，接收通道已经在对接收信号进行采样， 因为接收采样也从时间 0 开始。

有许多参数需要为接收周期指定，包括 A/D 转换器采样率、后 A/D 处理功能以及 Vantage 模块和主机内存上的本地内存中的存储位置。 在大多数情况下，系统能够从用户提供的最少一组参数中自动选择默认操作参数，从而使用户不必将大量参数添加到他们的设置脚本中。 VSX 加载程序会自动将参数添加到用户结构中，用户可以在运行脚本后检查这些参数。

Vantage 硬件中接收处理的目标是：

1) 提供足够的接收频谱 A/D 采样，包括前端的噪声频谱，如果可能的话。 为选定的采样率在 A/D 之前适当地设置抗混叠低通滤波器截止。

2) 低通滤波器并在必要时抽取 A/D 采样率，以提供大约四倍于换能器频谱中心频率的采样率。 4xFc 采样率提高了下游滤波的效率，并进一步降低了窄带信号的采样率。 任何图像重建处理也需要这个速率。

3) 对 4xFc 采样信号进行带通滤波以匹配换能器带宽，或缩小用于多普勒处理的信号带宽。

4) 将 4xFc 采样率抽取为对已过滤接收信号的带宽进行充分采样所需的速率。 这里抽取是在样本对上执行的，对于 4xFc 采样率，它代表基带 I 和 Q 信号的非并发样本。 对于 200% 带宽，保留所有 4xFc 样本； 对于 100% 带宽，每隔一个样本对将被丢弃； 对于 50% 的带宽，保留四分之一的样本对。 可以从这些抽取样本对中精确地重建原始滤波接收 RF 信号，前提是抽取样本率足以表示滤波 RF 信号的带宽。

5) 独立缩放每个接收通道的输出。 缩放用于接收变迹，或可用于补偿换能器元件中的增益变化。 它还可以用于标准化累积采集。

6) 如果需要，允许将新获取的接收数据与先前获取的数据进行平均。

为了帮助理解硬件中发生的信号处理，可以参考下图 3.3.1。 该图显示了 Vantage 系统中单个接收通道的信号路径。 在图中，红色的属性是通常由系统自动编程的 Vantage 硬件相关属性。

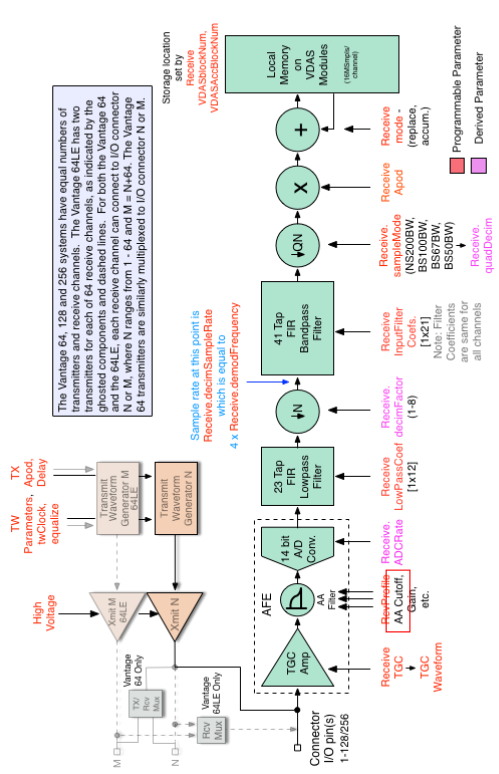


图3.3.1 Vantage硬件模块单接收通道信号处理路径

**信号路径说明（针对技术倾向）**

——时间增益控制放大器 (TGC) 增益使用参考 TGC 结构中指定的波形进行编程。 增益通常被编程为随着采集深度的增加而增加，以补偿信号衰减。 TGC 增益控制范围为 40 dB。 在 TGC 增益级之前有一个固定增益低噪声前置放大器，在它之后，在 A/D 之前有另一个固定增益缓冲器。 这些固定增益级中的每一个都有几个增益设置可用，以允许用户调整从传感器输入到 A/D 的总增益。 这些增益设置可以使用 RcvProfile 结构进行编程（参见第 3.3.3 节）。

——在A/D之前有一个可编程的抗混叠低通滤波器，截止频率由系统根据指定的换能器频率和带宽自动选择。 用户可以使用 RcvProfile 结构覆盖这个自动选择。

——A/D 转换器是一个 14 位转换器，可以从 10 MHz 到 62.5 MHz 的速率进行采样（采样率必须是 250 MHz 系统时钟速率的整数约数）。

——低通滤波器：在 A/D 之后的数字信号路径中，这个对称的 23 抽头滤波器使用 12 个 16 位系数进行编程，范围为 +/- 1.0。 系数数组中的第一个条目是与 23 阶 FIR 结构中最外面的一对阶一起使用的系数； 第十二项是用于中心抽头的系数。 19 位在每个乘法器的输出端进行，并通过求和（一个额外的 MSB 以防止满量程输入数据和系数溢出，以及四个额外的 LSB）。在输出端执行溢出检测和限幅，并且 2 lsbs 被丢弃以提供 16 位输出字宽。

——Decimator（抽取器）：Decimator可以编程为在将低通滤波器输出样本传递到带通滤波器时从 0 跳到 7 个样本。 这个想法是允许 A/D 以高采样率运行以对整个噪声频谱进行采样，但将样本以中心频率的四倍传送到带通滤波器。

——带通滤波器：这个对称的 41 抽头滤波器编程有 21 个 16 位系数，范围为 +/- 1.0。 系数数组中的第一项是用于 41 抽头 FIR 结构中最外面一对抽头的系数（抽头 1 和 41）； 第二个系数用于抽头 2 和 40，依此类推； 第 21 个系数用于抽头 21。在内部，滤波器求和提供两个额外的 LSB 和一个额外的 MSB，总共 19 位。 与低通滤波器一样，输出端的溢出检测和限幅逻辑允许丢弃 MSB，并截断两个额外的 LSB 以产生 16 位输出宽度。

——正交抽取器：该抽取器用于减少窄带信号在存储器中存储的输出样本数。 抽取器选项是：a) 通过所有输出样本（200% 带宽），b) 通过 2 个样本，跳过两个样本（100% 带宽），c) 通过 2 个样本，跳过 6 个样本（50% 带宽）。

——缩放乘法器：为这个 16x16 位乘法器提供了一个 16 位系数宽度，缩放后允许最大系数值为 +/- 4.0。 允许绝对值大于 1 的系数允许对以标称值 1.0 为中心的通道间增益误差进行增益补偿。 输出端需要另一个溢出检测和限幅电路，以保持 16 位的输出字宽。

——累加器：允许将传入的采集数据添加到先前采集的数据中。对超过 16 位的值进行溢出检测和裁剪。

**3.3.1 The Receive Object(接收对象)**

设置接收获取特性的主要对象是接收对象。 Receive 对象的属性详述如下。 必需的属性以黄色突出显示。 还有一些依赖属性是从其他属性派生的，由运行时软件添加以供参考。 这些以蓝色突出显示。

**Receive =**

mode double 0=replace data, 1=accumulate

Apod [1xnele double] Apodization value (see below) for rcv aper.

aperture double only used with HVMux probes to specify sub-aperture.

startDepth double starting depth of acquisition in wavelengths endDepth double ending depth of acquisition in wavelengths.

TGC double number of TGC waveform object to use.

bufnum double number of rcv buffer to use (defined in Resource) framenum double number of frame in rcv buffer to use.

acqNum double no. of acquisition in frame sequence.

sampleMode string ‘ NS200BW(I) ' , ' BS100BW ', ' BS67BW ', 'BS50BW’,’custom’

decimSampleRate double sample rate after decimation in MHz.

demodFrequency double frequency that will be translated to 0.

decimFactor double decimation factor after Low Pass Filter

ADCRate double A/D sample rate in MHz

quadDecim double decimation factor after input filter.

samplesPerWave double samples/wavelength of Trans.frequency

startSample double start row for this acqNum (added by VSX).

endSample double ending row for this acqNum (added by VSX).

LowPassCoef [1x12]double sym. coefs for 23 tap FIR following A/D

InputFilter [1x21]double sym. coefs of 41 tap input fltr (4\*Fc smpls)

callMediaFunc double 1=call Media Function (Simulator only)

**Receive.mode** 属性是一个数字，指定存储采集数据的方法。 对于 Receive.mode = 0，新获取的数据被写入存储内存，取代任何以前的数据。 对于Receive.mode = 1，将新获取的数据累加到存储内存中，方法是首先读取存储在内存中VDASblockNum指定地址的数据，将其添加到输入数据中，并将总和写回VDASblockNum。 （有关如何实现累积的更详尽解释，请参见第 6.1 节。）Receive.Apod 阵列与活动孔径中的换能器元件具有一对一的对应关系。 参与的接收元件由非零值指示，而零值关闭与该元件关联的接收通道并从任何重建处理中消除该元件。 Receive.Apod 值的范围从 -4.0 到 3.9997，用于设置信号处理链末端乘法器级的增益。 当使用负值时，接收样本将进行符号反转和缩放。

**Receive.Apod** 的标称值 1.0 将过滤后的 14 位 A/D 数据缩放为 16 位接收数据输出样本值的一半。 换句话说，14 位 A/D 的 -8192/+8191 计数输出变为 -16384/+16382。 因此，保留了输入滤波器生成的动态范围的额外最低有效位。 选择在 16 位存储器字中缩放到小于满量程以留下 1 位字增长用于增益补偿或信号平均。 例如，为了补偿接收器增益变化或换能器元件灵敏度变化的目的，用户可能想要将接收通道输出缩放得更高或更低。

对于 Vantage 64 或 64LE 系统，Receive.Apod 数组的非零值不能超过 64 个。 此外，接收通道多路复用施加了一个附加条件，即数组索引 N 和 N+64 不能同时为非零，其中 N 的范围为 1 - 64。VSX 将检查这些条件，如果不满足则报告错误。

**Receive.aperture** 属性仅用于探头手柄或主体中带有多路复用器的传感器。 Receive.aperture 的值指定在 Trans.HVMux.Aperture 数组中定义的孔径。 在没有相应的 Trans.HVMux.Aperture 定义的情况下使用 Receive.aperture 属性将在加载程序 VSX 加载脚本时导致错误。 Receive.aperture 的值引用 Trans.HVMux.Aperture 数组的第二个索引，并选择整个数组中定义的子孔径位置之一。 当使用 Receive.aperture 属性选择子孔径时，Receive.Apod 值将应用于子孔径的线性非零元件。 Trans.HVMux.Aperture 中的列（由 Receive.aperture 选择）为系统上选定的换能器连接器指定从换能器元件到连接器通道的映射。 [对于没有多路复用器的传感器（因此在接收结构中没有 Receive.aperture 值），Trans.Connector 数组提供相同的元件到连接器通道映射功能。]如果 Receive.aperture 值不同于 TX.aperture 在同一事件中的值，换能器或换能器适配器中的高压多路复用器必须等待切换，直到最后一个发射器已经发射并且切换时间可能达到几微秒。 这可能会延迟接收来自 RcvBuffer 中所有通道的有效接收数据。

**Receive.startDepth** 属性指定开始获取接收频谱中心频率波长样本的深度。 Receive.endDepth 属性指定停止的深度。 如果要执行图像重建，则支持的最大波长范围为 1024。使用 Vantage 硬件运行时，Receive.endDepth 参数控制采集周期的运行时长。 Receive.endDepth 的值或单次采集中可能需要的最大样本数没有固定的最大限制。 然而，极长的采集事件最终会遇到一些其他系统约束（例如 HW 系统中可用的内存量，或最大 DMA 发射大小，或主机中用于 matlab 工作区的可用内存等。 ) 由于 Vantage 硬件将输入样本存储在 128 个样本块中，波长深度将由加载程序 VSX 转换为样本（使用 Receive.samplesPerWave 属性来确定每个波长的样本）并四舍五入到下一个 128 样本边界。 换句话说，如果 Receive.endDepth - Receive.startDepth 不是 128 个样本的倍数，Receive.endDepth 将自动扩展以获取 128 个样本的倍数。

**Receive.TGC** 属性包含 TGC 波形结构集合的索引，定义如下。 接收模拟器支持媒介模型中的衰减，因此在模拟模式下运行时可以使用 TGC 波形。 Vantage 硬件可以在采集模块上的本地内存中存储多个预定义波形，VSX 目前为 8 条曲线分配了内存。 本地内存中的波形编号对应TGC波形结构索引编号。 当 Receive 对象加载到 Vantage 硬件中时，Receive.TGC 编号用于确定从中加载波形的本地内存区域中的位置。

**Receive.bufNum** 属性是一个整数，它定义了 CPU 内存中的接收缓冲区编号，采集数据应发射到该缓冲区。 这个缓冲区的特征在上面定义的 Resource 对象中定义。 每个缓冲区通常被定义为接受多帧采集数据，其中每帧可能包含多次采集。

**Receive.frameNum** 属性是一个整数，用于定义采集数据应发射到的 CPU 内存中 RcvBuffer（先前指定）的帧号。 要求 RcvBuffer 帧的大小和结构相同，并且每个帧都应由一组类似的 Receive 结构定义，但帧号除外。

**Receive.acqNum** 属性是一个整数，它定义了接收数据的多采集帧的采集编号。 当 RcvBuffer 帧包含来自多个采集的数据时，给定通道的单独采集事件样本将按顺序存储在内存中。 [对于Matlab数组，数据存储在顺序内存中的列中]例如，包含五个采集事件的 RcvBuffer 帧会将数据按顺序存储在内存中，如下所示：

Column 1 Column 2 Column 3

Ch.1-Acq1, Acq2, Acq3, Acq4, Acq5, ..., Ch.2-Acq1, Acq2, Acq3, Acq4, Acq5, ..., Ch.3-Acq1, …

样本中每个采集段的大小由 Receive(j) 给出：

n = 2\*(Receive(j).endDepth – Receive(j).startDepth) \* Receive(j).samplesPerWave

结果 n 向上舍入到下一个 128 个样本边界。 为一帧采集定义 Receive 结构时，Receive.acqNum 值应按升序定义，从一个开始作为该帧的第一个采集。 然后，加载程序 VSX 将为 Vantage 采集模块上的本地内存中的采集分配顺序存储位置，以便所有采集都可以在单个 DMA 操作中发射到主机内存。 在某些帧采集序列中，例如当使用不同焦区通过多个发射采集射线时（参见第 6.3 节关于多个发射区的编码），可能需要部分覆盖本地存储器中先前采集的数据。 这可以通过在为帧定义的接收结构序列中重新使用先前指定的 acqNum 来实现。 当指定的 Receive.acqNum 等于或小于先前定义的最高 Receive.acqNum 时，VSX 会尝试查找关联的先前定义的 Receive 结构，并为新的 Receive 结构分配本地内存中相同的存储位置。 如果新 Receive 所需的存储量大于先前定义的结构所用的存储量，则会发生错误。

**Receive.sampleMode** 属性用于指定对存储在 RcvBuffer 中的接收数据进行采样的方法。 Receive.sampleMode 的选择是：

‘NS200B W ’ Nyquist sampling (200% bandwidth ) of demodFrequency.

‘NS200BWI‘ 2-1 interleaved sampling (requires 2 acquisitions) .

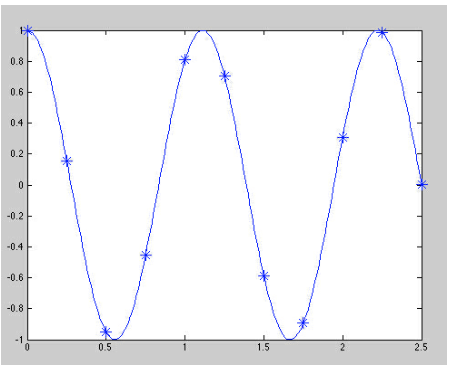
‘BS100BW ’ 100% bandwidth sampling of demodFrequency.

‘BS67BW‘ 67 % bandwidth sampling of demodFrequency .

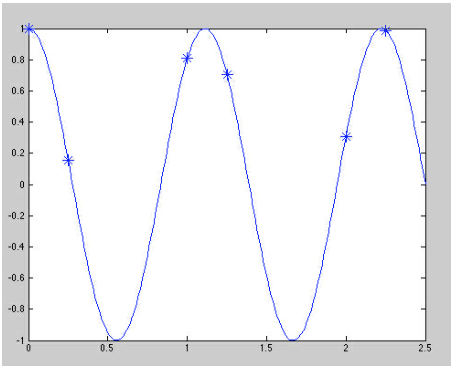
‘BS50BW‘ 50% bandwidth sampling of demodFrequency.

‘custom‘ sample rate set by decimSampleRate

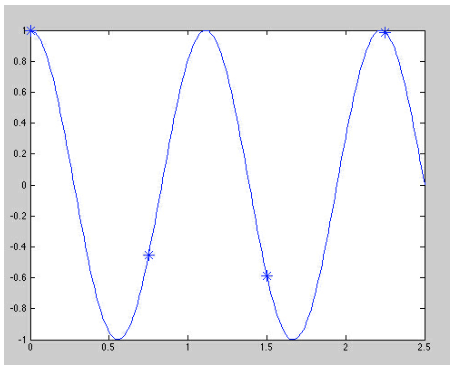
为了说明如何实施这些不同的采样方案，下图显示了可实现中心频率（有效中心频率或 demodFrequency）90% 处的余弦波样本。



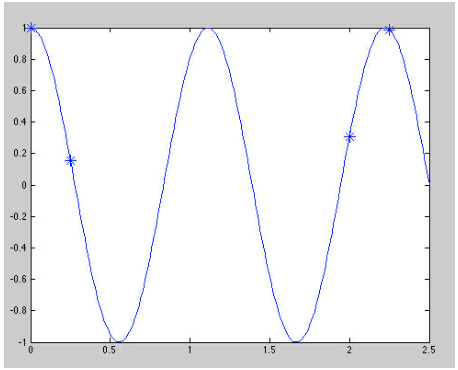
‘NS200BW’ 每个有效中心频率周期 4 个样本



‘BS100BW’ 在每个周期的有效中心频率周期的1/4处采样2个



‘BS67BW’ 每 3/4 个有效中心频率周期采样 1 次。



“BS50BW”每 2 个周期在有效中心频率周期的 1/4 处采样 2 个。

图3.3.1.1各种SampleModes的样本点

如果接收信号满足带宽要求，这些 sampleMode 方法中的每一个都允许以任何所需的采样率精确地重建原始接收信号。 Verasonics 图像重建软件可以使用任何提供的采样模式。 对窄带信号使用较低的采样模式的优点是需要较少的内存来存储数据，并减少了数据发射到主机内存的时间和总线带宽。 大多数超声换能器在 60 - 100% 的范围内具有典型的 -6dB 带宽，而多普勒测量通常使用大约 20 - 30% 的发射带宽进行。

在设置脚本中定义 Receives 时，用户应根据采样和带宽要求选择 Receive.sampleMode。 如果没有定义 sampleMode，系统软件会设置默认属性‘NS200BW’。 通过前五个 sampleMode 选择，系统软件随后将使用提供的 Trans.frequency 和 Receive.sampleMode 值选择适当的 ADCRate、decimSampleRate、decimFactor、

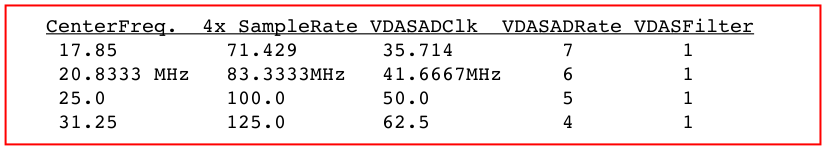
和 quadDecim 因子。 例如，对于给定的 Trans.frequency 和“NS200BW”sampleMode，软件将选择最高的 ADCRate，它可以在低通滤波器之后被抽取以产生 4\*F 速率，其中 F 尽可能接近Trans.frequency，基于 Vantage 的可用采样率。 低通滤波器后的抽取因子将在 decimFactor 中设置，生成的采样率在 decimSampleRate 中设置。 demodFrequency 将设置为 decimSampleRate 的 1/4，quadDecim 因子将设置为 1（对于 200% 带宽）。 样本存储在本地内存中的采样率始终是 decimSampleRate 除以 quadDecim 因子。 然后将设置 samplesPerWave 属性以指定接收缓冲区中每个发射频率波长的实际样本数。 如果 Trans.frequency 恰好是所选采样率的 1/4，则 Receive.samplesPerWave 将等于 4.0，但如果不是这种情况，则 Receive.samplesPerWave 可能大于或小于 4.0，具体取决于可实现的最接近位置 4\*F 采样率下降。 例如，如果 Trans.frequency = 5.0，并且 sampleMode 为“NS200BW”，则可以实现的最接近的 4\*5.0 采样率为 19.2308 MHz (250/13)，产生的 Receive.samplesPerWave 值为 3.8462。 在这种情况下，Receive.demodFrequency 将等于 19.2308 MHz/4 或 4.8077，并且与 Trans.frequency 值不匹配。 如果用户计划进行多普勒处理，他们将希望将事件的发射频率（在 TX(Event.tx) 中定义）与事件的 Receive.demodFrequency（在 Receive(Event.rcv) 中定义）相匹配，以确保基带 I 和 Q 样品。 对于未与 Trans.frequency 值对齐的派生 demodFrequency，不会发出警告。

通常，Receive.sampleMode 和 Trans.frequency 是指定接收采样率所需的全部。 如果用户希望设置独立于 Trans.frequency 的采样率，则允许指定 Receive.sampleMode 并同时指定 Receive.decimSampleRate 或 Receive.demodFrequency（如果指定两者，则它们必须一致 ). 在这种情况下，运行时软件不会使用 Trans.frequency 值，而是会尝试找到一个可实现的 ADCRate，该 ADCRate 在抽取高达 8 倍后产生用户提供的 decimSampleRate 或 demodFrequency。 然后软件将设置其他相关属性，包括 Receive.samplesPerWave 属性。 例如，如果用户将 sampleMode 设置为“NS200BW”并将 decimSampleRate 设置为 20.8MHz，则运行时软件将选择 62.5 的 ADCRate 和 3 的 decimFactor。decimSampleRate 将修改为 20.8333 MHz，并使用 5MHz Trans.frequency , Receive.samplesPerWave 将设置为 4.1667。 然后 Receive.demodFrequency 将计算为 20.8333/4 或 5.2083 MHz。 同样，对于多普勒采集，用户可能希望以 5.2083 MHz 发射以确保基带 I 和 Q 样本。

如果用户为 sampleMode 选择“自定义”选项，则允许直接为 decimSampleRate 或 ADCRate 设置一个值。 如果他们这样做，该值将被调整为硬件支持的最接近的可实现率。 如果未提供 decimSampleRate，则将其设置为与 ADCRate 相同的值，将 decimFactor 编程为 1。如果用户提供 decimSampleRate 值但未指定 ADCRate，则软件将选择最接近的可实现 ADRate，其 decimFactor 最高，可产生用户的 decimSampleRate 值 . 如果同时提供 ADCRate 和 decimSampleRate，则它们必须与 1 - 8 的整数 decimFactors 一致。在“自定义”sampleMode 的所有情况下，demodFrequency 由软件设置为 decimSampleRate 的 1/4，并且 quadDecim 因子集 到 1. Receive.sampleMode = 'custom' 不能用于提供给重建操作的 Receive 中，这将导致错误。

请注意，Receive.sampleMode 可以在任何单独的 Receive 事件中独立设置。 这允许在同一 RcvBuffer 帧内指定不同的采集采样率，这在使用混合模式采集时可能是可取的，例如 2D 和多普勒，其中 2D 采集需要 200% 的带宽，而多普勒采集可以使用 50% 的带宽。 始终希望使用适合采集的最低带宽，以减少将采集数据移动到主机所需的 DMA 发射带宽。

Receive.interleave 属性已被删除，交织现在由“NS200BWI”sampleMode 属性指示。 如果运行时软件针对 4\*Trans.frequency 的 decimSampleRate 计算出高于 62.5MHz 的 ADCRate，则假设将使用 2-1 交错采集，ADCRate 将简单地减半。 支持交错采样的中心频率为：



对于200% 带宽采样，Trans.frequency 值不必是上面列出的精确频率之一（Trans.frequency 应在可实现频率的 10-20% 以内以进行充分采样），但应该注意 demodFrequency 必须始终是列出的中心频率之一。 用户负责在两个交错采集中使用适当的 TX 延迟对适当的双采集进行编程。 第一次获取交织对的 TX 应该有额外的 TX.Delay 时间。 重建将自动为具有‘NS200BWI’sampleMode 属性的 Receives 调用 ReconInfo.Pre =‘interleaveRF’函数。 此函数将进行两次连续采集并将它们交错。 Receive 的 decimSampleRate 和 demodFrequency 属性被编程为匹配交错数据采样率。 有关交错采样的更多信息，请参阅第 7.2.2.2 节。

**Receive.samplesPerWave** 属性现在是从其他属性派生的依赖属性。 为了向后兼容，只要未提供 sampleMode 属性，就允许用户指定 Receive.samplesPerWave 属性。 在这种情况下，将应用对 samplesPerWave 的旧限制，并且 4、2、4/3 和 1 的值将被转换为相应的 sampleMode 属性，并且 Trans.frequency 值将被要求为“允许”值 基于可实现的 Vantage 采样率。 对于没有 sampleMode 和 4、2、4/3 或 1 以外的 samplesPerWave，软件将为 sampleMode 属性设置“自定义”值，并相应地计算 ADCRate。 如果提供了 sampleMode 属性，则 samplesPerWave 属性成为派生参数，其计算值将覆盖任何用户提供的值。

**Receive.startSample** 和 **Receive.endSample** 值由运行时软件（当执行序列设置脚本时）根据 Receive.startDepth 和 Receive.endDepth 值使用 Receive.samplesPerWave 和 Receive.acqNum 值计算得出。 计算这些属性是为了方便用户在从 RcvBuffer 访问 RF 采集数据时使用。 例如，在执行设置脚本后，可以使用 Matlab 语句在 RcvBuffer one 的第一帧中为第三次采集 (Receive(3).acqNum = 3) 绘制通道 32 的 RF 数据：

**> plot(RcvData{1}(Receive(3).startSample:Receive(3).endSample,32,1))**

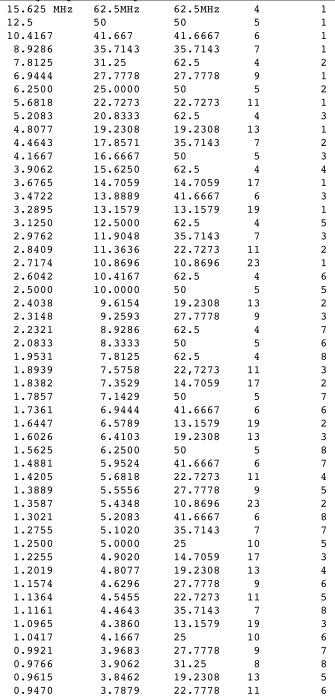
Receive.LowPassCoef 属性是一个包含 12 个值的行向量，表示 23 个抽头 FIR 滤波器的对称系数，该滤波器直接应用于每个接收通道的 A/D 转换器。 此时，采样率与 A/D 采样率相同。 前 11 个值 (C1 - C11) 是第一个和最后 11 个滤波器抽头的对称系数，而最后一个值 (C12) 代表滤波器中心抽头的系数。 系数的范围是 -1.0 到 +1.0。 Receive.LowPassCoef 属性的设置是可选的，如果用户未定义，VSX 会在执行用户脚本时提供一组合适的默认值。 该滤波器级的目的是从接收输入频谱中去除大于换能器中心频率两倍的频率，从而允许将采样率（如有必要）抽取至 4\*Fc。 对于 A/D 速率不能设置为 4\*Fc（最小 A/D 速率为 10 MSPS）的低频传感器，需要这种抽取。 4\*Fc 采样率是正确操作随后的输入滤波器阶段以及图像重建例程所必需的。 如果 A/D 采样率不需要抽取（它已经是 4\*Fc），低通 FIR 滤波器可以用来执行其他滤波器操作，并且可以编程为高通、带通，甚至 带阻滤波器。

Receive.InputFilter 属性是一个包含 21 个值的行向量，表示 41 个抽头 FIR 滤波器的非零系数，该滤波器应用于低通滤波器输出抽取器之后的每个接收通道。 此时假设采样率大约是接收中心频率的四倍。 前 20 个值 (C1 – C20) 是 41 抽头带通 FIR 滤波器的对称系数，而最后一个值 (C21) 是中心抽头的系数。 系数的范围可以是 -1.0 到 +1.0。 输入滤波器的目的是进一步塑造接收器输入频谱，以提高信噪比并消除接收器和 A/D 转换器输出中的直流偏移。 对于典型的采集，最好使用与所用换能器的发射频谱相匹配的带通滤波器。 如果输入滤波器之后的正交抽取滤波器用于进一步降低采样率，则应相应地设置输入滤波器带宽以防止混叠。 如果 InputFilter 属性缺失或设置为空 ([ ])，VSX 将为带通滤波器提供一组与 samplesPerWave 属性设置一致的默认系数。

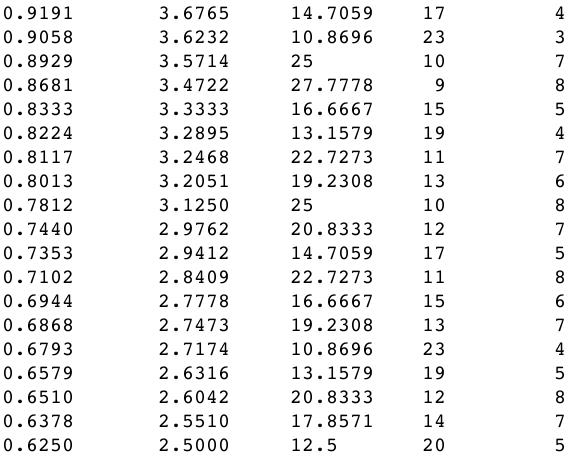
Receive.callMediaFunc 仅由模拟器使用。 将 Receive.callMediaFunc 设置为 true (1.0) 会在模拟接收数据之前调用 Media.function 中指定的 Matlab 函数。

下表列出了与4xFc采样一致的250MHz主时钟导出的中心频率（demodFrequency）。

解调频率 抽样率 ADC率 250MHz/N 十进制因子



解调频率 抽样率 ADC率 250MHz/N 十进制因子



**表3.3.1.1 支持的解调频率和采样率**

**3.3.2 Receive Profile Control Structures（接收配置文件控制结构）**

“RcvProfile”数据结构为用户提供了一种机制来控制与每个通道的接收信号路径中的 TGC 前置放大器和 A/D 转换器相关的各种设置。 用户可以根据需要在设置脚本中定义的 RcvProfile 结构中设置变量（在下面的 3.3.3.1 节中列出）。 VSX 将为用户未明确设置的任何 RcvProfile 变量分配默认值。

用户可以通过三种方式使用 RcvProfile 结构，为事件序列中的每个采集事件实现所需的接收信号路径设置：

1. 使用 VSX 分配的默认设置 如果用户认为不需要更改 VSX 分配的任何默认值，则根本不需要在用户脚本中定义 RcvProfile 结构。 当 VSX 加载脚本并发现没有 RcvProfile 结构时，它将创建一个结构并使用分配给每个 RcvProfile 变量的默认值填充它。 当 HAL 将脚本加载到 HW 系统上执行时，所有接收通道都将使用此默认 RcvProfile 结构中的值进行编程。 然后采集序列中的每个接收事件都将使用这些相同的值。

2. 应用用户指定的设置，对所有接收事件使用相同的值 如果用户想要更改一个或多个 RcvProfile 变量的默认值，而默认值将由 VSX 定义，只需在 SetUp 脚本中定义一个 RcvProfile 结构， 并根据需要分配值。 当 VSX 看到这个用户定义的 RcvProflile 结构时，它将简单地按原样使用它来将所需的值发送到 HAL 以对 HW 进行编程。 VSX 将添加用户未指定的任何变量，并为其分配相关的默认值。 当 HAL 将脚本加载到 HW 系统上执行时，所有接收通道都将使用此用户创建的 RcvProfile 结构中的值进行编程。 然后采集序列中的每个接收事件都将使用这些相同的值。

3. 为事件序列中的不同接收事件设置不同的用户定义值 对多普勒和成像事件使用不同的设置），然后可以定义多个 RcvProfile 结构的数组，在每个结构中指定不同的值。 在每个结构中，VSX 将自动为任何未指定的变量填充默认值。 然后在事件序列中，用户可以在任何需要的地方添加“setRcvProfile”序列控制命令。 setRcvProfile 命令的参数将标识 RcvProfile 结构之一，所选结构的值将在事件序列中的那个点加载到 HW 接收通道。 这些值将对所有后续接收事件保持有效，直到在事件序列中遇到另一个 setRcvProfile 命令。

**3.3.2.1 RcvProfile Variable Definitions（RcvProfile变量定义）**

下面给出了 RcvProfile 数据结构中用户可编程的每个变量的说明

• RcvProfile.DCsubtract 是一个字符串变量，必须设置为“on”或“off”。 它控制 A/D 的直流偏移减法功能，以允许从每个通道的接收 A/D 数据样本中减去每个通道常数（存储在另一个每个通道的硬件寄存器中）。 在系统上电时，自动校准序列用于确定每个通道的 A/D 转换器产生的实际直流偏移值。 该值保存在每个通道的寄存器中，因此当启用 DCsubtract 时，每个通道的 DC 偏移量将归零为零。 如果用户未指定，VSX 将为 DCsubtract 设置默认值“开”。

• RcvProfile.AntiAliasCutoff 是一个双精度值，其允许值为 5、10、15、20 和 30，表示 A/D 转换器之前的低通滤波器的截止频率（以 MHz 为单位）（5 仅适用于之后构建的系统 大约 2015 年 9 月）。 Vantage 高频系统具有 35 和 50 MHz 的附加截止频率值。 如果用户未指定，VSX 将自动设置默认值，该默认值是高于传感器带宽上限 Trans.Bandwidth(2) 的可用选项中的最低值。 请注意，如果用户未设置 Trans.Bandwidth，VSX 将根据 Trans.frequency 的 60% 分配默认值。

• RcvProfile.PgaGain 是一个double，用于设置A/D 之前的前置放大器输出缓冲器的总增益（以dB 为单位）。 唯一允许的值为 24 和 30 dB。 如果用户未设置，VSX 将设置默认值 24 dB。

• RcvProfile.PgaHPF 是控制PGA 积分器反馈路径的双精度数，用于提供高通频率响应并使直流耦合信号路径中可能存在的直流偏移为零。 此字段的唯一允许值是 80 和 0。RcvProfile.PgaHPF = 80 启用积分器反馈路径，从而导致 PGA 的一阶高通频率响应具有大约 80 KHz 的断点。 如果用户未设置此属性，则这是默认系统设置。 RcvProfile.PgaHPF = 0 禁用积分器，允许 PGA 一直具有平坦的频率响应直至直流。 这可能会导致不可预测的 DC 偏移添加到信号中。

• RcvProfile.LnaGain 是一个double，用于设置固定增益低噪声输入放大器级的总增益（以dB 为单位）。 唯一允许的值为 15、18 和 24 dB。 （最低设置为 12 dB，而不是大约 2015 年 9 月之前构建的系统的 15。）如果用户未设置，VSX 将设置默认值 18 dB。

• RcvProfile.LnaHPF 是用于LNA 的积分器反馈路径的双重控制，用于提供可编程的高通频率响应并消除直流耦合信号路径中可能存在的直流偏移。 此字段的唯一允许值是 200、150、100、50 和 0。设置为 200、150、100 和 50 的值会启用积分器反馈路径，从而导致 LNA 的一阶高通频率响应具有 断点分别在大约 200 KHz、150 KHz、100 KHz 和 50 KHz。 RcvProfile.LnaHPF = 0 禁用积分器，使 LNA 具有平坦的频率响应，一直到直流。 这可能会导致不可预测的 DC 偏移添加到信号中。

• RcvProfile.LnaZinSel 是一个双重选择反馈电阻器，用于启用控制 LNA 输入放大器输入阻抗的“有源终端”功能，如传感器连接器所示。 此变量必须设置为 0 到 31 范围内的整数值。该值本身没有直接意义（它表示一个 5 位二进制值，其中每一位切换 5 个不同电阻值中的一个进出电路 ). 值 0 给出最低输入阻抗设置，即 115 欧姆。 输入阻抗随着值的增加而增加，最高阻抗值为 31（没有主动反馈的“高阻抗”状态）。 对于给定的反馈电阻，所得输入阻抗也将是 LNA 增益设置的函数，由上述 LnaGain 值控制。 请注意，当 LnaZinSel 设置为 31 时，“高 Z”输入阻抗与 20 pf 并联时约为 8,000 欧姆。如果用户未指定 LnaZinSel，则默认值为 0。默认 LNA 增益为 18 dB， 这导致输入电阻为 115 欧姆。

对于具有高频配置的 Vantage 系统，已修改输入反馈网络的硬件设计，以在前置放大器输入端引入高通频率响应。 带宽超过25MHz的探头如果采用“4/3采样”（见7.2.3节），则必须在A/D转换前引入高通滤波器滤除DC到Fs/2的无用信号和噪声 （因为这些频率会与从 Fs/2 到 Fs 的 Nyquist 频带混叠，这将在使用 4/3 采样时由系统处理）。 此高通响应由输入阻抗网络提供，并可通过 RcvProfile.LnaZinSel 进行编程。 编程值为零会导致高通截止频率约为 20 MHz，并且截止频率会随着 RcvProfile.LnaZinSel 值的增加而逐渐降低。 值为 31 会禁用反馈网络，因此也会禁用高通滤波器，从而在低频下产生平坦的频率响应，输入阻抗为 6000 欧姆（与上述标准频率配置相同）。 如果 RcvProfile.LnaZinSel 没有在设置脚本中指定，系统将分配默认值 31 以禁用高通滤波器。 这是不使用 4/3 采样功能的较低频率探头

‘’

的首选设置。

**3.3.3 TGC Waveform Objects TGC （波形对象）**

TGC 对象为采集事件的接收部分定义了时间增益补偿曲线。 对象的结构如下：

**TGC =**

CntrlPts [1x8 double] 指定曲线的控制点。

rangeMax double 最大范围的波长数。

Waveform [1x512 double] TGC 曲线的 512 个增益值

要定义 TGC 波形，用户在设置脚本中指定 TGC.CntrlPts 数组和 TGC.rangeMax。 系统 SW 然后使用这些值来合成 TGC.Waveform 数组； 该数组用于在接收事件期间创建实际的时变 TGC 控制信号（在 3.08 之前的版本中，TGC.Waveform 数组使用 uint16 数据类型）。

TGC.CntrlPts 数组指定在接收采集间隔过程中 8 个等距点的 TGC 级别，从零开始到 TGC.rangeMax 设置的深度结束，以 Trans.frequency 的波长为单位。 因此 TGC.rangeMax 通常设置为等于 Receive.endDepth。 用于 TGC.CntrlPts 的值必须在 0 到 1023 的范围内，其中 0 表示最小 TGC 增益，1023 表示最大 TGC 增益。 Vantage 系统的 40 dB TGC 范围在控制点的 0 到 1023 范围内大致呈对数线性。 每个控制点值用于设置设置为 VSX 的 matlab GUI 控件中 TGC 控件滑块之一的初始位置。

如果需要，可以通过定义一组 TGC 结构来创建多个 TGC 波形。 然后 Receive.TGC 变量可用于选择要用于每个单独接收结构的 TGC 波形。 当定义了多个 TGC 结构时，GUI TGC 控件滑块将默认为 TGC(1)。 在 3.07 软件版本中，可以使用滑块上方的 GUI 控件选择其他波形。

3.0 版本的媒介模拟软件支持 TGC 接收采集，并将 TGC 增益曲线应用于模拟的 RF 接收数据。

**3.4 Reconstruction Objects(重构对象)**

重建对象仅由软件处理使用，并且被排除在加载到 Vantage 硬件中的序列对象结构之外。 它们描述了从获取的接收数据中形成像素信息所需的重建方法，并与 ReconInfo 对象一起，通常指定重建完整 PData 帧所需的所有属性。

**3.4.1 Recon**

Recon 结构提供重建的一般属性，包括要使用的源缓冲区和目标缓冲区。 Recon对象的属性如下：

**Recon =**

senscutoff [double] 从总和中排除通道的截止值。

pdatanum [double] 要使用的 PData 结构的编号。

rcvBufFrame [double] 如果提供，在ReconInfo覆盖帧号

newFrameTimeout [double] (dflt(default，默认）1000 msec)等待新frm的时间

IntBufDest [1x2 double] [InterBuffer编号，frame编号]

ImgBufDest [1x2 double] [InterBuffer编号，frame编号]

RINums [n double] n ReconInfo 结构编号的行向量

**Recon.senscutoff** 是一个从 0.0 到 1.0 的值，用于设置灵敏度阈值，低于该阈值的元件将从给定像素的重建求和中排除。 典型值为 0.6，对应于 4.44 dB 的信号损失。 降低 senscutoff 值会增加换能器近场接收孔径中的有效元件数量，从而增加横向分辨率。 然而，由于元件对其表面法线方向的回波最敏感，远离重建点方向的大回波信号会增加并导致混乱。

**Recon.pdatanum** 指定定义重建数据像素位置的 PData 结构的编号。

**Recon.rcvBufFrame** 是一个可选属性，在提供时会覆盖 ReconInfo 结构指定的帧编号。 如果设置为非零正值，则该值为将用于重建的帧编号。 如果设置为 -1，并与 Vantage 硬件一起运行，则处理确定发射到此 Recon 结构使用的 RcvBuffer 中的最后一个采集帧，并处理此帧。 如果最后发射的帧恰好与上次执行的重建中使用的帧相同，则处理将等待最多 1000 毫秒以发射新帧，如果没有新帧出现，则旧帧将被发射 再次处理。 如果需要，Recon.newFrameTimeout 属性可用于更改此默认超时。 使用 -1 设置允许序列实现异步采集和处理，其中硬件以一种速率采集帧，并且处理以不同的速率连续处理最近采集的帧。 当不使用 Vantage 硬件（模拟模式 1 或 2）运行时，rcvBufFrame = -1 设置将导致重建处理始终使用重建的前一帧中的下一帧。 当启动序列、退出冻结状态或从模拟模式 0（正常采集）转换到模拟模式 2（接收数据循环）时，Recon.rcvBufFrame = -1 的 Recon 中重建的第一帧将是来自 Resource.RcvBuffer.lastFrame，通常是缓冲区中最旧的帧。 此行为允许从缓冲区中最旧的帧开始播放接收数据循环。

**Recon.IntBufDest** 和 **Recon.ImgBufDest** 分别使用一个二元件数组指定目标缓冲区和将接收重构输出的帧。 （用于源 RF 数据的 RcvBuffer 在 ReconInfo 对象引用的 Receive 对象中提供。）根据 ReconInfo 对象中指定的重建模式，可能需要一个或两个缓冲区目的地。 对于 Recon.ImgBufDest 中的帧编号值，可以使用值 -1，表示应使用下一个可用的帧编号进行输出。 这可用于提供先前重建的历史记录，例如，可用于自定义处理。

**Recon.RINums** 属性是一个行向量，指定与此重建关联的 ReconInfo 结构索引。 每个 ReconInfo 对象都包含有关如何为特定像素数据区域重建像素的信息。 各个像素区域通常由“computeRegions”函数定义。 从 3.0 软件版本开始，每个区域的像素被自动划分为子集，供重建例程的多个处理线程处理。

**3.4.2 ReconInfo Objects（ReconInfo对象）**

对于每个 Recon 对象，都有一组关联的 ReconInfo 对象。 ReconInfo 对象集对于每个 Recon 应该是唯一的，即使在某些情况下，结构看起来是相同的。 （虽然可见属性可能相同，但在运行时添加的属性取决于父 Recon 结构。）上述 Recon.RINums 列表中的每个数字都是特定 ReconInfo 对象的编号。 这些对象提供了有关如何处理像素数据区域的详细信息，因此至少有与 PData.Regions 一样多的 ReconInfo 结构。 在组合多个重建以产生结果的情况下，每个区域可能有多个 ReconInfo 结构。 ReconInfo 对象的属性如下所示。

ReconInfo =

mode [string] Numeric or string representation数字或字符串表示

txnum [double] 用于获取的发射对象的编号。

rcvnum [double] 用于获取的发射对象的编号。

Aperture [1xnumchannels double]（可选）光圈规格

regionnum [double] 像素数据的区域结构数。

pagenum [double] 表示多页 InterBuffer的页码

normPower [double] 可选的归一化功率（见文本）。

normWeight [double] 可选的归一化权重（见文本）。

scaleFactor [double] Optional scale factor for all modes.

Pre [string] ’clearInterBuf’,‘clearImageBuf’,‘interleaveRF’

Post [string] ’IQ2IntensityImageBuf’,’…Add’,’…Mul’

threadSync [double] default=0, 1=>sync after recon w this RI

**ReconInfo.mode** 指定重建的模式。 它通常用字符串指定，但为了向后兼容 3.0 之前的版本，它也可以用数字指定。 在下面的描述中，InterBuffer 和 ImageBuffer 是在父 Recon 结构中指定的。 目前支持的选项有：

**Mode** **Destination Buffer(s)（目标缓存区）**

**0** ‘replaceIntensity‘ ImageBuffer,(IQ—>InterBuffer if defined)

**1** ‘addIntensity‘ ImageBuffer

**2** ‘multiplyIntensity‘ ImageBuffer

**3**  ‘replaceIQ‘ InterBuffer

**4**  ‘accumIQ‘ InterBuffer

**5**‘accumIQ\_replaceIntensity‘ InterBuffer, ImageBuffer

**6** ‘accumIQ\_addIntensity‘ InterBuffer, ImageBuffer

**7** ‘accumIQ\_multiplyIntensity‘ InterBuffer, ImageBuffer

使用**“replaceIQ”**和**“accumIQ”**模式，可以在 InterBuffer 中为多个采集事件累积重建的 I、Q 数据，并计算最后一次重建、替换、添加或乘以输出中的数据的总和的大小 ImageBuffer（模式 5、6 或 7）。 为了根据对总和做出贡献的通道数对重建数据进行归一化，使用与 ImageBuffer 大小相同的内部辅助通道计数缓冲区来存储多次重建的累积通道数。 模式 5、6 和 7 使用此累积通道计数来标准化 IQData 的幅度。

**ReconInfo.txnum** 指定用于发射的 TX 对象的索引。 需要此对象来确定发射波束的来源和特性。

**ReconInfo.rcvnum** 指定用于获取的 Receive 对象的索引。 Receive 对象包含有关如何获取 RF 数据以及数据存储位置的信息。 重建软件仅限于处理 4096 个样本的深度，因此 (Receive.endSample - Receive.startSample) 必须小于 4096（这些属性是在使用 VSX 运行设置脚本时计算的）。

**ReconInfo.Aperture** – 此可选属性是一个整数数组，指定重建通道正在使用的换能器元件。 默认情况下，此数组是在运行时从 ReconInfo.rcvnum 指定的 Receive 结构的 Receive.Apod 数组构建的。 如果由用户提供，默认值将被覆盖。 条目数应与重建使用的通道数相同。 元件编号应按通道顺序列出，并以与 RcvBuffer 帧和采集编号中的映射相同的方式对应于通道到元件的映射。 该阵列允许从更大的换能器阵列中指定任何元件子组； 然而，规范自然必须与数据采集保持一致，这通常会限制哪些元件可以连接到各个接收通道。 例如，要为 128T/64R Vantage LE 系统指定 128 通道孔径的中央 64 元件部分，ReconInfo.Aperture 规范如下：

[64,65,66,67 ... 92,93,94,95,32,33,34,35 ... 60,61,62,63]

说明按通道顺序，表示通道1连接到元件64（从0开始计数），通道2连接到元件65，依此类推。 同样，如果重建使用的孔径与 Receive(ReconInfo.rcvnum).Apod 数组指定的孔径相同，则无需指定此数组。

**ReconInfo.regionnum** – 要使用此 ReconInfo 对象重建的区域的编号。 每个 ReconInfo 对象为 Recon.pdatanum 中指定的 PData 结构的一个且仅一个区域指定重建参数。 不允许使用数字 0，在 3.0 之前的软件版本中表示“所有区域”，因为不再自动创建额外的 ReconInfo 结构。

**regionnum** 属性引用 PData.Region 结构，该结构定义了该区域中的所有像素。 如果 ReconInfo.txnum 规范引用的 TX 结构中没有 TX.TXPD 数据，则将重建所有区域像素。 如果 TXPD 数据存在，它将用于限定 PData.Region 规范中的哪些像素将被重建。 在这种情况下，只有 Region 中符合条件的像素才会接收重建输出。 不合格的像素要么不被触摸，要么将被写入零，这取决于重建模式。

**ReconInfo.pagenum** - 只有在重构为具有多个页面的 InterBuffer 时才需要此属性。 多页 InterBuffer 通常用于彩色多普勒重建，其中需要一组采集来捕获多普勒信息。 然后可以将每个采集重建到 InterBuffer 的单独页面。

**ReconInfo.normPower** 和 **ReconInfo.normWeight** - 这些属性在提供时提供用于写入 ImageBuffer 的重建模式的规范化常量。 对于算术平均值，可以使用“replaceIntensity”，然后进行多次“addIntensity”重建，或者对于基于多次采集 I、Q 数据的合成孔径，“accumIQ\_replaceIntensity”，然后进行多次“accumIQ\_addIntensity”重建。 添加到总和的每个重建都可以使用 normWeight 来归一化其贡献。 三个重建的结果，其中一个人想要取加权算术平均值然后是：

**a \*ReconInfo(1).normWeight + b\*ReconInfo(2).normWeight + ...**

**c\* ReconInfo(3).normWeight**

其中，ReconInfo(1).mode = ‘replaceIntensity’,

ReconInfo(2).mode = ‘addIntensity’,

ReconInfo(3).mode = ‘addIntensity’

对于同等权重的贡献，normWeight 值将分别等于 1/3。

**normPower** 常量用于对几何平均值的乘积进行归一化，方法是取 n 个强度值的每个乘积的 n 次方根。 （几何平均值通常比算术平均值更适合超声数据的平均，因为它会导致更暗的充满液体的区域，例如囊肿。）常数表示为分数指数值。 对于几何平均值，可以使用“replaceIntensity”模式，然后进行多模式“multiplyIntensity”重建，或者对于合成孔径，使用“accumIQ\_replaceIntensity”模式，然后进行多模式“accumIQ\_multiplyIntensity”重建。 例如，如果重建由三个相乘的强度重建组成，则集合中使用的 ReconInfo 结构可以将 normPower 设置为 1/3，以将立方根应用于每个被乘数 (a^0.333 \* b^0.333 \* c^ 0.333 = (a\*b\*c)^0.333)。

这些常量必须由用户在设置脚本中提供，因此由用户为采集和重建方法设置适当的值。 如果未提供这些属性，则应用 normWeight 和 normPower 的默认值 1.0。 有关使用乘法平均进行空间复合的示例脚本，请参阅 SetUpL11\_4vFlashAngleSC.m 脚本。

**ReconInfo.scaleFactor** - 可以提供此常量来缩放任何重建模式的输出，因为它在任何其他输出处理（例如计算幅度或累加到 InterBuffer）之前应用于 I、Q 重建输出。 它的主要用途是对将在 InterBuffer 中组合的多个合成孔径采集进行加权。 例如，对于 7 角度平面波发射重建，可以加权比直线前进方向更高的转向角，因为由于元件方向性函数，发射波前的幅度随着转向角的增加而减小。 然后，在写入 ImageBuffer 时，通过使用 ReconInfo.normWeight 或 ReconInfo.normPower 进行加权，可以将生成的 7 角度重建累加与其他重建输出相结合。 如果未提供，ReconInfo.scaleFactor 的默认值为 1.0。

**Pre** 属性用于指定在重建操作之前执行的函数，而 **Post** 属性指定在重建操作之后执行的函数。 “clearInterBuf”和“clearImageBuf”Pre 函数清除目标帧的整个 PData 像素区域。 如果重建输出仅进入作为整个 PData 数组中像素子集的区域，则需要这些函数，因为未写入的像素可能包含来自先前重建的数据。 如果重建输出到整个 PData 数组，则可以改用“replaceIQ”或“replaceIntensity”重建模式。

当 PData 数组的部分区域上的多个 IQ 重建在 InterBuffer 中组合时，如在多个重叠宽波束的情况下，使用 Post 函数“IQ2IntensityImageBuf”、“IQ2IntensityImageBufAdd”和“IQ2IntensityImageBufMul”。 在这种情况下，求和的 IQ 数据使用重建模式“accumIQ”累积在 InterBuffer 中，并且在最后一次 IQ 重建之后，使用 Post 函数检测整个 IQ 帧以在目标 ImageBuffer 中强度数据。 如果最后一次重建的区域是 PData 数组的子集，则对帧中的最后一次重建使用“accumIQ\_replaceIntensity”模式重建（重建 IQ、累积和检测）不等同于使用“IQ2IntensityImageBuf”Post 函数，因为 只有像素的子集会被处理。 **“IQ2IntensityImageBufAdd”**和**“IQ2IntensityImageBufMul”**（类似于模式“accumIQ\_addIntensity”和模式“accumIQ\_multiplyIntensity”）可用于将强度数据与输出缓冲区中已有的数据相加或相乘。 在这些 Post 例程中，“normWeight”或“normPower”的 ReconInfo 属性可用于规范化输出值。 如果不需要 Pre 或 Post 操作，则这些属性可以缺失或为空。

**“threadSync”**属性设置为 1 (true) 时，指定重建线程应在 ReconInfo 中指定的区域重建后同步。 当重建具有多个 ReconInfo 结构且具有待处理的唯一 regionnums 并且 ReconInfo 区域中的像素与一个或多个尚待处理的 ReconInfo 区域中的像素重叠时，需要线程同步。 如果没有线程同步，不同的线程可能同时处理具有共享像素的不同区域，在竞争条件下写入相同的像素。 多次 ReconInfo 重建中的最后一个 ReconInfo 始终执行线程同步，即使 threadSync 属性为 0。线程同步会减慢重建处理速度，应仅在需要时使用，这主要是在处理具有重叠区域的宽波束扫描时。 线模式重建不需要线程同步，那里有大量不重叠的Region。 当对同一区域或整个 PData 空间的所有像素执行多次重建时，也不需要它，如在 flash 和 flashAngles 脚本中。 如果用户不确定是否需要线程同步，可以检查模拟模式下静态图像的重建。 如果没有像素表现出闪烁强度行为，则不太可能需要线程同步。

**3.5 Process Objects（过程对象）**

过程对象用于描述应用于获取或重建数据的处理类型。 过程对象仅由系统软件使用 - Vantage 硬件定序器忽略事件列表中的这些项目。 Process 对象是一个结构体，它指定在事件列表中的特定点执行的处理任务。

**Process =**

classname [string] Class of processing objects.处理对象的类。

method [string] Method to use for class.用于类的方法。

Parameters [string,value,string,value,…] 定义处理对象的各种属性的属性、值对数

组。在 Matlab 中，此构造是一个元胞数

组，由可能具有不同数据类型的字符串和

值组成。

**3.5.1 Image Display Object（图像显示对象）**

一个常见的处理任务是显示图像。 虽然重建处理的输出可以是像素强度数据，它本身定义了可以在显示器上查看的图像，但在将图像数据渲染到显示器之前通常需要执行额外的处理。 例如，强度数据通常在动态范围内进行压缩，并缩放到适合输出显示的最大数值。 此外，持久化处理通常用于对新帧与先前帧数据进行平均。 如果强度数据和多普勒速度数据要在同一显示器上显示，则多普勒数据必须替换强度数据或与强度数据组合，并且选择合适的颜色图来显示信息。 与这些动作相关联的处理对象是一个Image Object，它由以下结构体定义：

Image =

imgbufnum [double] 用于源的 ImageBuffer 编号。

framenum [double] Source frame number (use -1 for last) .源帧编号（最后使用 -1）

pdatanum [double] No. of PData structure 结构编号

srcData [string] ‘intensity’(default),’signedColor’,’unsignedColor’

grainRemoval [string] ‘none’(default),‘low’,‘medium’,‘high’

persistMethod [string] ‘none’(default), ‘simple’, ‘dynamic’

persistLevel [double] Amount of persistence, range 0–99 持久化程度，范围 0–99

interpMethod [string] Interpolation method (‘4pt’ interp(default)). 插值方法

processMethod [string] ‘none’(default),‘reduceSpeckle1’,‘reduceSpeckle2’

averageMethod [string] ‘none’(default),‘runAverage2’,‘runAverage3’

pgain [double] Processing gain. 处理增益。

reject [double] Low level reject (0-100) 低级拒绝 (0-100)

compressMethod [string] ‘power’ or ‘log’

compressFactor [double] (0-100)

mappingMethod [string] ‘full’(default),‘lowerHalf’,’upperHalf’.

threshold [double] 0-255。与像素值比较以确定是否覆盖。

gamma [double] 压缩应用于灰度颜色图 （gray scale colormap）

blacklevel [double] 拒绝应用于灰度颜色图

display [double] 0-不显示，1-显示。

displayWindow [double] Dest. display window目标的显示窗口（默认为 1）。

当 Process 对象中指定的类名为‘Image’时，在 VSX 初始化期间创建一个 Image 对象来控制处理。 在 Matlab 中，Image 对象的属性将使用 Properties.Parameters 结构在属性/值对的元胞数组中给出。 下面是定义一组图像属性所需的 Matlab 代码示例。

**Process(1).classname = ‘Image’;**

**Process(1).method = ‘imageDisplay’;**

**Process(1).Parameters = {‘imgbufnum’,1,… % ImageBuffer to process.**

**‘framenum’,1,…**

**‘pdatanum’,1,…**

**‘compressMethod’,‘power’,… % power comp.**

**‘compressFactor’, 40,… ‘**

**pgain’,1.0,… % image processing gain**

**‘reject’,3,… % reduce intens. below this index**

**‘mappingMode’,’full’,…**

**‘displayWindow’,1,… % Dest. display**

**‘display’,1}; % display image**

如上所示，并非所有处理属性都需要定义，因为未定义的属性将默认为适当的值。

渲染过程包括根据 Image 结构的属性处理指定 ImageBuffer 中的源数据。 生成的像素数据被写入与 DisplayWindow 关联的 DisplayData 数组。 如果 Image.display 属性为真 (1)，则 DisplayData 数组将复制到显示窗口的 CData 结构中，并将图像呈现到屏幕上。 通过将 Image.display 属性设置为 false (0)，可以计算 DisplayData 数组，但不会将图像渲染到显示器。 当组合来自多个 ImageBuffer 的数据时通常使用此功能，如彩色多普勒成像。 第一个 ImageBuffer 被处理成 DisplayData 数组而不渲染到屏幕，然后第二个 ImageBuffer 被处理并与同一个 DisplayData 数组组合。 在 DisplayData 数组中组合所有 ImageBuffers 之后，最后的图像处理将 Image.Display 属性设置为 true (1)，从而使图像显示在屏幕上。

显示数据有四个临时缓冲区。 缓冲区 ID0 充当某些处理例程的临时缓冲区，例如插值函数的输出。 缓冲区 ID1 到 ID3 充当已处理图像数据帧的历史缓冲区，允许将连续处理的帧组合起来进行显示。

强度数据的**“grainRemoval”**属性从三个不同的 3x3 矩阵过滤器中进行选择，以去除与其周围邻居显着不同的奇异低强度值和高强度值。 过滤器分为“低”、“中”和“高”，并提供越来越高的颗粒去除水平。 这些过滤器对 ImageBuffers 中的数据进行操作，因此对这些缓冲区的像素密度很敏感。 像素密度越高，这些过滤器的影响就越小。

**“persistMethod”**有三个选项——“none”、“simple”和“dynamic”。 “动态”选项需要为四个值的“persistLevel”指定一个向量。 每个值根据帧与帧之间像素强度值的变化量来设置持久性。 第一个值设置小于满量程强度四分之一的变化的持久性。 第二个值设置在满量程强度的四分之一到二分之一之间变化的持久性，依此类推。 可以使用图像处理工具 (Processing tool ，PT ool) 实时调整这些值。

**“interpMethod”**现在被指定为一个字符串，“4pt”是当前支持的方法。 4pt 插值比以前的版本有了显着改进，现在在粗像素密度方面做得更好。 如果没有指定处理方法，则将强度数据的插值输出发送到历史缓冲区中的当前帧； 否则，它被发送到 ID0 临时缓冲区。

**“processMethod”**用于在插值的 ImageBuffer 帧上应用空间过滤器。 这些过滤器是 3x3 矩阵过滤器，根据应用于周围邻居的函数修改中心像素。 当前的过滤器是“reduceSpeckle1”和“reduceSpeckle2”。 这些过滤器对周围像素的线对进行操作，并将中心值钳位到所选线对之间的差异。 “reduceSpeckle1”找到绝对差值最小的线对，并用它来限制中心值。 “reduceSpeckle2”对所有线对的最小值和最大值进行排序，并将中心值固定在最小值的最大值和最大值的最小值之间。 processMethod 将 ID0 缓冲区作为输入并输出到历史缓冲区中的当前帧。

**“averageMethod”**有三个选项：“none”、“runAverage2”和“runAverage3”。 后两者分别计算最后两帧或三帧的运行平均值。 历史缓冲区中的原始帧不会被修改，因为平均值是计算到 DisplayData 缓冲区的。

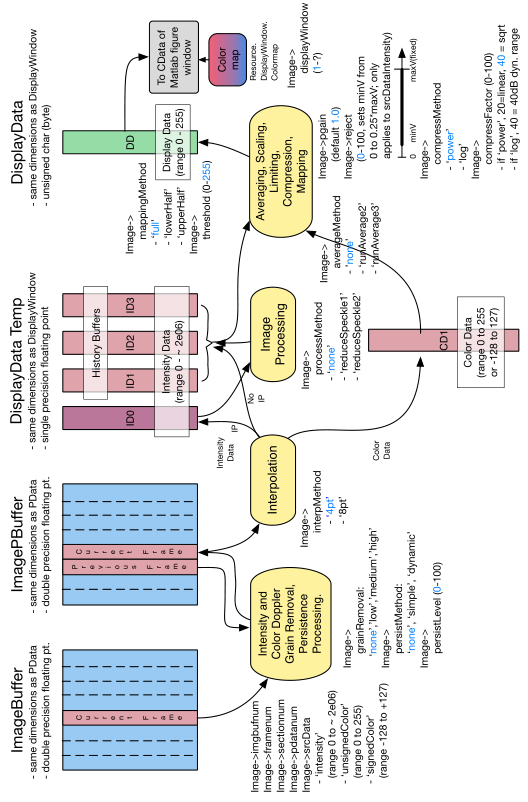
**“pgain”、“reject”**和**“compressMethod”**属性现在在数据写入 DisplayData 缓冲区之前应用于强度数据。 首先应用“pgain”比例因子，最大数据值限制为 maxV，其中 maxV 是一个固定值，它是从归一化重建强度输出中预期的最大值。 “reject”属性的范围从 0 到 100，将 minV 设置为 maxV/4 的百分比。 然后从缩放后的数据中减去 minV 值，使数据现在介于 0 和 maxV-minV 之间。

**“compressMethod”**现在有两个选项——“power”和“log”。 “power”选项计算强度提高到分次幂，由“compressFactor”设置。

20 的“compressFactor”为线性输出提供 1.0 的幂，而 40 的“compressFactor”为平方根压缩提供 0.5 的幂。 较高的“compressFactor”值提供较小的功率，从而产生更多的压缩。 “log”选项根据“compressFactor”值提供不同比例因子的对数压缩曲线。 “compressFactor”的值越高，对数曲线的上升速度越快，从而提高了低电平强度值的亮度。 Matlab 图形窗口的灰度曲线（或颜色图）不再由 vsx\_gui 中的压缩和拒绝滑块修改，因为这些函数现在由图像结构的“压缩因子”和“拒绝”属性指定。 如果图形窗口颜色图需要修改，可以通过设置相应 Resource.DisplayWindow 的“颜色图”属性或使用 Matlab 中图形窗口的颜色图编辑工具来加载新图。 已经开发出 Verasonics 色图工具，超声技师可以使用该工具对图像进行最终优化。

在某些应用程序中，用户可能希望使用自定义显示例程，而不是 Matlab 图形窗口。 通常情况下，runAcq处理代码在生成图像显示输出时会调用Utilities目录下的VsDisplay函数，将要复制的DisplayData数组传递给DisplayWindow的CData。 用户可以添加或替换 VsDisplay.m 函数中的代码来执行自己的显示功能。

如果需要，外部显示功能也可以编码为 Matlab mex 文件。 使用外部显示函数时，Image 参数引用的 Resource.DisplayWindow 的属性“displayWindow”仍必须在 Resource 定义中定义，因为传递给 UDisplay 函数的 DisplayData 将由这些属性定义。imageDisplay 数据流和处理函数如下图 3.5.1.1。



**图3.5.1.1、Post-Reconstruction Image Processing（后-重建图像处理）**

**3.5.2 Doppler Process Object（多普勒过程对象）**

对于重建 IQ 数据的多普勒处理，必须使用以下结构定义多普勒对象。

Doppler =

srcbufnum [double] 源缓冲区号

srcframenum [double] 源帧编号

Srcsectionnums [1x2double] 3D 扫描的源部分 [start,stop]。

srcpagenum [double] 起始页号。 在 numPRIs 个页面之外。

pdatanum [double] PData 结构编号

dstbufnum [double] 目标缓冲区的编号

dstframenum [double] 目标的帧数

prf [double] 以Hz为单位的 PRF。

numPRIs [double] 要处理的 PRI 数。

wallFilter [string] Method of wall filtering.（墙过滤方法）

pwrThreshold [double] 最大功率阈值的百分比

maxPower [double] 用于计算和显示的最大功率电平

postFilter [double] 过滤估算数据的方法（0=none）

当 Process 对象中指定的类名为“Doppler”时，会在运行时创建一个 Doppler 对象来控制处理。 至于 Image 对象，Doppler 对象的属性应该使用 Properties.Parameters 结构参数在属性/值对的元胞数组中给出。 下面是为彩色血流成像定义一组多普勒属性所需的 Matlab 代码示例。

Process(1).classname = ‘Doppler’;

Process(1).method = ‘computeCFIFreqEst’;

Process(1).Parameters = {‘srcbufnum’,1,… % no. 要处理的缓冲区。

‘srcframenum’,1,… % 起始帧号。

‘srcpagenum’,1,…

‘pdatanum’,2,…

‘dstbufnum’,1,…

‘dstframenum’,1,…

‘prf’,2000,… % 多普勒 prf，以赫兹为单位

‘numPRIs’,16,… %集合中的 PRI 数量

‘wallFilter’,‘regression’,… % wall filter的方法

‘pwrThreshold’,0.1,… % （**最大功率阈值 %**）

‘postFilter’,0};

对于 2D 图像格式，集合中包含的各种 I、Q 数据帧应重建为复杂格式中间缓冲区的单独页面。 示例源缓冲区定义可能如下所示：

**Resource.InterBuffer(m).datatype = 'complex';**

**Resource.InterBuffer(m).numFrames = 1;** % 只需要一帧。

**Resource.InterBuffer(m).rowsPerFrame = 2048; Resource.InterBuffer(m).colsPerFrame = 64;**

**Resource.InterBuffer(m).pagesPerFrame = numPRIs;** % 每个集合的 numPRIs 页数

在 2D 情况下，多普勒处理从“srcbufnum”的“srcframenum”中的“srcpagenum”开始，并按“numPRIs”页面的页码递增。

对于 3D 图像格式，多普勒集合包含多帧重建的 I、Q 数据，每帧包含多个页面。 在这种情况下，中间源缓冲区定义可能如下所示：

**Resource.InterBuffer(m).datatype = 'complex';**

**Resource.InterBuffer(m).numFrames = numPRIs;** %每个 PRI 需要一帧。

**Resource.InterBuffer(m).rowsPerFrame = 2048;**

**Resource.InterBuffer(m).colsPerFrame = 64;**

**Resource.InterBuffer(m).pagesPerFrame = n; %**3D体积中的页数

在 3D 情况下，多普勒处理从“srcbufnum”的“srcframenum”中的“srcpagenum”开始，并按“numPRIs”帧的页码递增，处理每页中的部分范围。 如果要处理帧中的所有部分，可以通过省略 srcsectionnums 属性或将其设置为 [0,0] 来完成。 如果要处理一系列部分，请将范围指定为开始编号和结束编号，例如 [10,20] 以处理第 10 到 20 部分。sectionnums 规范只能用于 3D 扫描格式。

3.5.2.1 选择壁式过滤器

多普勒处理的壁滤波方法指定为字符串。 当前的选择包括“无”和“回归”，以及 FIR 滤波器和两组矩阵滤波器设计的设置。

“无”——没有墙过滤。 此设置可用于组织多普勒传感。

“回归”——选择二阶（二次基）回归滤波器。 此过滤器最适合较短的合奏长度（低于 16 PRI）。

FIR 滤波器设置 - 在 3 个 FIR 滤波器选项中进行选择。 FIR 滤波器最适用于较长的集成长度（> 19 PRI），因为它们按滤波器顺序减少了集成中的滤波样本数量。 因此，一个 12-tap 32 PRI 合奏在过滤后将只有 21 个 PRI。 如果指定的 FIR 滤波器少于 20 个 PRI（不包括虚拟对象），则会生成运行时错误。 过滤器的名称和特征是：

**• ‘FIRLow’ - 13 taps, -60 dB below 0.003 PRF , -3dB @ 0.08 PRF**

**• ‘FIRMedium’ - 12 taps, -63 dB below 0.017 PRF , -3dB@ 0.15 PRF**

**• ‘FIRHigh’ - 13 taps, -65 dB below 0.035 PRF , -3dB @ 0.18 PRF**

矩阵滤波器设置（一般流动条件）——在一组受限的集合长度下，矩阵滤波器系数的六种选择提供了一系列高通衰减和通带行为。 这些实现了时变脉冲响应滤波器。 六个过滤选项中的每一个都可用于 [12,24]（净值，不包括前体“虚拟”脉冲）范围内的偶合奏长度。 过滤器名称是：

**• 'TVFiltVLow'**

**• 'TVFiltLow'**

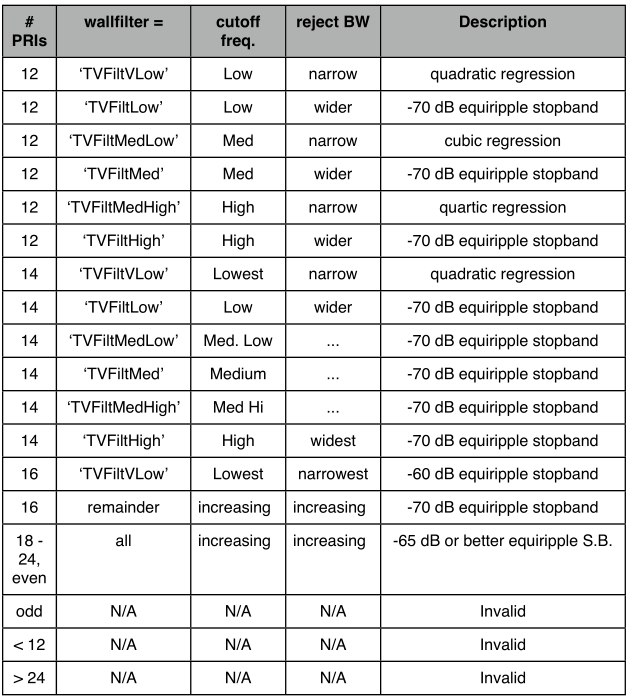
**• 'TVFiltMedLow'**

**• 'TVFiltMed'**

**• 'TVFiltMedHigh'**

**• 'TVFiltHigh'**

**下表描述了每个过滤器名称对应的特征**

****

矩阵滤波器设置（Weak/Low Flow Conditions，弱/低流量条件）- 在弱多普勒信号流条件下，20 到 60 之间的一组受限集合长度允许适当矩阵滤波器系数的六种选择。 它们具有 -95 dB 的等波纹阻带，并针对检测整体长度为 { 20、24、26、28、30、34、36、40、44、48、52、56、60 } 的微弱流量信号进行了优化， 不包括前体脉冲。 The Weak Flow/Low Flow Conditions (弱流量/低流量条件)过滤器名称是：

**• 'WeakFlowVLow'（最低阻带截止频率）**

**• 'WeakFlowLow'**

**• 'WeakFlowMedLow'**

**• 'WeakFlowMed'**

**• 'WeakFlowMedHigh'**

**• 'WeakFlowHigh'（最高阻带截止频率）**

请注意，在更高的整体长度下，弱流矩阵滤波器过渡带变得陡峭，而阻带足够宽，即使在低截止设置下也能发挥作用。 这允许低截止滤波器使用高 PRF，这通常会提高当前在多普勒过程对象中实现的自相关处理的检测性能。 弱/低流量过滤器系数性能记录在“CDIWeakFlowFilterResponses.pdf”中，存储在文档子文件夹中。

**3.5.3 External Process Object（外部过程对象）**

对于用户想要提供他们自己的处理例程的情况，运行时软件提供了一种机制来支持这一点。 在事件列表中的适当位置，用户放置一个带有进程结构的事件，该结构标识 1) 所需的源缓冲区、帧号、节号和页码（页码属性只能与类型为’inter’的 srcbuffer 一起使用'); 2) 将处理后的数据发送到的输出缓冲区帧、节和页码； 3) 外部处理例程的名称，应定义为普通的Matlab函数（或mex文件）。 下面是流程结构的格式：

**Process(1).classname = ‘External’; % 将处理标识为外部处理。**

**Process(1).method = 'processFunctionName';**

**Process(1).Parameters = {'srcbuffer','receive',...**

**'srcbufnum',1,... %要处理的缓冲区的数量。**

**‘srcframenum’,1,… % 起始帧号。**

**‘srcsectionnum’,1,…**

**‘srcpagenum’,1,…**

**‘dstbuffer’，‘image’，...**

**‘dstbufnum’,1,…**

**‘dstframenum’,1,…**

**‘dssectionnum’,1,…**

**‘dstpagenum’,1};**

srcbuffer 和 dstbuffer 的选择是‘receive’、‘inter’、‘image’和‘imageP’或‘none’。 当不需要输入和/或输出缓冲区时，选择“无”选项。 srcframenum、srcsectionnum 或 srcpagenum 缺少规范或值为零意味着输入需要所有框架、所有部分或所有页面。 目前，仅支持连续的内存发射，因此不允许从所有帧发射单个部分或单个页面。 当为 srcbuffer 指定“receive”、“image”或“imageP”时，srcframenum 允许值为 -1，它用于指定处理到缓冲区中的最后一帧。 如果为 dstbuffer 指定了‘image’或‘imageP’，则 dstframenum 也允许值为 -1 或 -2。 这两个选项都写入目标缓冲区中的最后一帧，但 -2 会在写入前递增最后一帧指针和 Receive.ImageBuffer.lastFrame 属性。 当为 dstbuffer 指定‘image’或‘imageP’时，dstpagenum 属性将被忽略。

在外部处理数据时，用于输入的缓冲区应被视为只读； 在对“runAcq”的连续调用之间修改这些数据可能会导致软件故障，因为 runAcq 在初始化时读取并存储这些缓冲区的地址一次，并且写入 Matlab 数组通常会导致 Matlab 制作原始数组的副本， 然后修改它并用不同的地址存储。 此外，如果您的外部处理函数正在使用向量指令处理多帧缓冲区的各个帧，则可能需要考虑每个帧的内存对齐，并且可能需要将输入缓冲区的帧大小定义为 32 字节的倍数 必要的。 对于输出缓冲区，不应修改原始缓冲区； 相反，要写入的新值应该作为外部函数的输出返回。

Matlab处理函数的格式应该是一个m文件函数，第一条语句格式如下：

**function Output = myProcessingFunc(Input)**

只允许一个输入和输出参数，Output 数组的格式应与目标数组的格式匹配。 输出格式可以在运行时通过使用 Matlab 工作区中的资源缓冲区定义来确定。 例如，从指定缓冲区读取接收 RF 数据并返回图像数据的函数可以具有以下定义：

**function ImageData = myProcessFunc(ReceiveData)**

在上面的函数定义中，ReceiveData 和 ImageData 都是函数的局部变量。 如果返回的 ImageData 用于替换 ImageBuffer 中的帧，则它应该具有维度（Resource.ImageBuffer(n).rowsPerFrame、Resource.ImageBuffer(n).colsPerFrame）。

另一个更完整的示例是以下外部处理例程，它在获取帧后绘制特定通道的 RF 数据。 处理结构定义如下：

**Process(2).classname = 'External';**

**Process(2).method = 'myFunction';**

**Process(2).Parameters = {'srcbuffer','receive',...** % 要处理的缓冲区。

**'srcbufnum',1,...**

**'dstbuffer','** **none'};** % 无输出缓冲区

在事件列表中，可以在任何事件中引用外部处理函数，而无需现有的处理引用。 在以下情况下，处理引用包含在“NOP”事件中。

**Event（n）.info = 'noop';** % 帧之间的 noop用于帧速率的控制

**Event (n).tx = 0;** % 不发射

**Event (n).rcv = 0;**  % 没有接收

**Event（n）.recon = 0;** % 不重建

**Event (n).process = 2;**  % 外部处理函数

**Event(n).seqControl = 1;** % reference for‘noop’command(“noop”命令参考)

用于绘制 RF 数据的外部处理函数在 m 文件中定义如下，应保存在 Matlab 路径中的某个位置。

**function myFunction(RData)**

**persistent myHandle**

**channel = 10; % Channel no. to plot**

**if isempty(myHandle)||~ishandle(myHandle)**

**figure;**

**myHandle = axes('XLim',[0,1500],'YLim',[-2048 2048], …**

**'NextPlot','replacechildren');**

**end**

**plot(myHandle,RData(:,channel));**

**return**

在默认目录中定义多个外部函数可能会有问题 - 这些文件需要具有与其他脚本中使用的名称不同的唯一名称，并且当脚本移动到不同的目录时，必须记住也移动关联的外部函数文件。 为了防止外部处理函数与安装脚本分离，在最近的软件版本中可以将安装脚本中的外部处理函数 m 文件定义为程序行的单元格数组。 元胞数组必须具有名称 EF(n).Function，并且第一个成员应该是函数的原型。 程序行被定义为带引号的字符串。 例如，上面列出的“myFunction.m”文件可以在安装脚本中定义如下：

**EF(1).Function = {'myfunction()', …**

**'persistent myHandle', …**

**'channel = 10; % Channel no. to plot', …**

**'if isempty(myHandle)||~ishandle(myHandle)', …**

**' figure;', …**

**'myHandle = axes(''XLim'',[0,1500],''YLim'',[-2048 2048],...' …**

**' ''NextPlot'',''replacechildren'');', …**

**'end', …**

**'plot(myHandle,RData(:,channel));', …**

**'return', …**

**};**

请注意使用双引号将带引号的值包含在字符串中。 当设置脚本中包含外部函数元胞数组定义时，加载程序函数 VSX 会将元胞数组解释为 Matlab 的“tmp”目录中的文件，并将“tmp”目录添加到路径中。

对于较长的外部函数，将函数定义为元胞数组中带引号的字符串是乏味且难以阅读的。 提供了一个辅助函数，它将两个分隔符之间的文本转换为单元格数组字符串，称为“text2cell”，其调用输入是要读取的文件名（默认情况下，调用该函数的脚本）和分隔符字符串。 这允许在可以自动解释为 EF.Function 元胞数组的安装文件末尾编写普通的 Matlab 代码。 为防止在运行 SetUp 脚本时执行此代码，请在定义 SetUp 脚本中的所有结构后放置一个“return”，然后在“return”之后，将外部函数的代码行放在两个实例之间 指定的分隔符。 仅在第一行使用函数原型，因为包含关键字“function”会产生 Matlab 错误，因为脚本中不允许定义函数。 例如，我们可以指定上面的 EF(1).Function 如下：

**return** % 将此返回放在脚本末尾以防止执行下面的代码

%EF#1

**myfunction()**

**persistent myHandle**

**channel = 10; % Channel no. to plot**

**if isempty(myHandle)||~ishandle(myHandle)**

**figure;**

**myHandle = axes('XLim',[0,1500],'YLim',[-2048 2048],...' …**

**'NextPlot','replacechildren');**

**end**

**plot(myHandle,RData(:,channel)); return**

%EF#1

然后将函数行插入 EF(1) 所需的全部内容。函数如下：

**> EF(1).Function = text2cell('%EF#1');**

请务必使用唯一标识符来定义多个外部函数，并确保不要在 SetUp 脚本主体的任何位置使用此标识符字符串，因为它必须唯一地括起您的函数行。

运行 VSX 后，可以在 Matlab 编辑器中通过键入（VSX 将 tempdir 添加到路径）查看在 tempdir 目录中创建的文件：

**> edit myFunction.m**

对于调试，请记住每次执行 VSX 时，tmp 目录中的函数都会被元胞数组定义覆盖，因此应在元胞数组定义或分隔符括号文本中而不是在函数中更改代码 tmp 目录。要在 Matlab 中调试外部函数，请在元胞数组定义中插入行，

**'keyboard', …**

或者在函数原型行之后或要开始调试的位置，只插入括号内分隔符定义（the delimiter bracketed definition）的

**Keyboard**

这将导致 Matlab 在此行中断并返回到命令提示符。此时，可以在 Matlab 编辑器中调出外部函数，使用上面的编辑命令，单步执行代码并观察执行情况。 如有必要，使用“dbquit”命令退出调试模式。

**3.6 Sequence Control Objects（序列控制对象**

序列控制对象用于提供与序列事件关联的序列流控制信息。 此外，还有专门与板载定序器操作相关的 VDAS 参数。）

command [string] 控制 seq.operation 的命令（见正文）。

condition [string] 启用命令的条件。

argument [double] 命令参数

通过在 Event.seqControl 属性中指定它们的索引值列表，可以在单个事件中指定最多三个 SeqControl 命令。 当在 Event 对象中给出 SeqControl 索引或索引列表时，列出的控制命令通常将在任何 TX/Receive 获取周期之后按指定的顺序执行。 （一个 TX/Receive 采集周期由事件结构中非零 TX 或 Receive 结构编号表示。）但是，一些 SeqControl 命令需要在事件开始时和任何 TX/Receive 采集周期之前执行 . 无论在多命令 SeqControl 列表中指定的顺序如何，以下命令都将在事件开始时执行。

**1) ‘triggerOut’** - 触发输出脉冲将在事件开始时和任何 TX/ Receive interval（接收间隔）之前立即生成。 有关详细信息，请参阅下面的“triggerOut”定义文本。

**2) 'pause'（用于触发输入）**或 **'triggerIn'** - 使用其中任何一个，HW 定序器将暂停并等待触发输入信号，紧接在它准备好开始 TX/Receive 间隔之前 相同的事件，因此该间隔将在触发信号后立即开始。

**3) ‘setRcvProfile’** - 当放置在具有活动 TX/接收间隔的事件中时，到新 RcvProfile 状态的转换将在该事件的活动间隔开始之前完成。

**4) ‘setTPCProfile’**——当放置在包含 TX/Receive 活动的 Event 中时，TPC 配置文件转换将安排在发射活动结束后发生。 在这种情况下需要默认的“下一个”条件。 如果在非获取事件中指定了“setTPCProfile”，则可以使用“立即”条件立即开始转换。

下面列出了当前可识别的 SeqControl 命令。 如果命令被硬件和软件定序器识别，则它以黑色列出。 仅由硬件定序器识别的命令为红色，仅由软件定序器识别的命令为蓝色。

**'call'** 将当前事件指针压入堆栈并跳转到参数属性中指定的事件编号。

**'cBranch'**  条件分支到参数属性中指定的新事件编号。 条件字段应设置为“bFlag”：

– bFlag 使用 Matlab 函数设置。 如果 bFlag 为 0，则不发生分支； 如果是

nz，branch.bFlag在分支走完后会自动清空。 （见正文）

**' DMA '** 启动 DMA，如 DMAControl 对象中所述，其编号在参数属性中给出。

**' jump'** 无条件分支到参数属性中指定的事件编号。 可选条件：exitAfterJump（见

正文）

**‘loopCnt’** 将循环计数器 N (1-8) 设置为参数属性中指定的值（最大值 65536）。 该

值将在测试后递减（请参阅以下命令）。 要设置的计数器由条件字段中的

文本“counterN”指定。

**‘loopTst’** 测试循环计数器N：如果为0，则继续； 如果 nz，递减循环计数并跳转到

参数属性中指定的指令。要测试的计数器由条件字段中的文本“counterN”

指定。

‘**markTransferProcessed‘** 将发射标记为已处理。 通过在参数字段中提供先前

“transferToHost”命令的 SeqControl 编号来引用对标

记的发射。 此命令通常用于从与 DMA 关联的

“waitForProcessing”条件中释放硬件定序器。

**‘multiSysSync’** 仅与使用外部时钟同步模块的多个 Vantage 系统一起使用。 条件

选项为“正常”（默认）、“BNC\_Rising”或“BNC\_Falling”。 参数

以毫秒为单位指定 SW 错误超时（默认 10000 或 10 秒）。（有关使

用信息，请参阅多系统同步的应用说明）。

**‘noop’** 执行参数属性中指定的持续时间的 NOOP（“NO OPeration”）时间延迟

（值 \* 200nsec；最大值为 2^25 - 1，持续 6.7 秒）。 软件定序器是否在模

拟模式 0（与硬件一起运行）中实现“noop”可以通过指定条件属性来设置。

‘Hw&Sim’（未指定属性时默认）表示软件仅在模拟模式 1 和 2 中执行

‘noop’，而忽略模拟模式 0 中的‘noop’，即在硬件中运行时。

“Hw&Sw”指定软件定序器也在模拟模式 0 中执行“noop”。

**‘pause’** Pause for condition (VDAS) -当前支持的条件：

‘extTrigger’——等待外部触发。 触发器选项使用参数指定：

1 - 触发输入 1 使能下降沿。

2 - 触发输入 2 使能下降沿。

3 - 启用两个触发器（下降沿必须同时发生）。

17 - 触发输入 1 使能上升沿。

18 - 触发输入 2 使能上升沿。

19 - 启用两个触发器（上升沿必须同时出现）。

将 256 添加到上述参数值指定等待具有 2.8 微秒延迟的外部触发。 这用于

锁定 DMA 内存访问以确保一致的采集时间。

**‘returnToMatlab’** 当遇到这个命令时，处理函数runAcq 暂停执行并返回到

Matlab 环境。 下次调用 runAcq（未指定 startEvent）时，将

在序列中的下一个事件处继续执行。

**‘RTN’** 从堆栈弹出事件编号指针，并在指针的下一个事件开始执行指令。

**‘setTPCProfile’**指定到 TPC 高压配置文件编号的转换，该编号在参数字段中给出。

有效配置文件编号为 1 到 4，配置文件 5 可用于 TPC 的高功率选项。

支持条件：

‘immediate’ - 立即开始转换到新配置文件。

‘next’ -（默认）在 TPC.xmitDuration 周期结束时（如

果已设置）或当前获取周期结束时开始过渡到新配置文件。

**‘setRcvProfile**’指定到 RcvProfile 的转换，其索引在参数字段中提供。 新配置文件将

一直有效，直到给出下一个 setRcvProfile 命令。

**'stop'**  停止执行指令并清除指令指针。

**'sync'**  同步硬件和软件定序器。 可以在参数属性中设置软件定序器等待硬件定

序器的超时时间（最多 2.14e09 usec）。 （见正文）

**'timeToNextAcq'** 指定下一次获取事件的时间（以微秒为单位）。 时间在参数属性中指

定，范围为 10 – 4190000 微秒。 条件字段可用于指示是否不应将错

过的 timeToNextAcq 周期报告为警告消息（“忽略”）。 软件定序器

仅在模拟模式 1 和 2 中识别“timeToNextAcq”，而在硬件（模拟模

式 0）中运行时忽略它。 （见正文）

**'timeToNextEB'** 指定下一个扩展突发发射事件的时间（以微秒为单位）。 时间在参数属

性中指定，范围为 10 – 4190000 微秒。 条件字段可用于指示是否不

应将错过的 timeToNextAcq 周期报告为警告消息（“忽略”）。 （请

参阅第 3.2.4.2 节 - 使用高功率 TPC 配置文件。）

**'transferToHost'** 此命令用于让 VSX 创建“DMA”命令并自动将其插入到序列中。 还

创建了一个 DMAControl 对象来指定 DMA 发射。 可以使用名为**“waitForProcessing”**的此命令指定一个可选的条件属性，然后需要一个参数值。 在此

条件下，将执行 DMA 发射，但如果参数中指定的 SeqControl 结

构发射的数据尚未标记为已处理或重建（通过引用发射到的

RcvBuffer 帧的 Recon 对象），则硬件定序器将暂停 在下一次

DMA 发射之前。 注意：参数字段中引用的 SeqControl 结构必须

是“transferToHost”命令。 硬件定序器等待的超时以毫秒为单位

由 Resource.VDAS.dmaTimeout 设置（默认 = 1000 毫秒）。

（见正文。）

**‘triggerIn’** 指定 TRIG IN 1&2 BNC 连接器的条件。（参见文本。）

**'triggerOut'**  在 TRIG OUT BNC 连接器上生成外部 1 微秒低电平有效输出。 可以在

参数字段中设置延迟。 （见正文。）

**‘waitForTransferComplete’**用于暂停软件定序器，直到参数字段中指定的发射完成。

参数字段应引用先前“transferToHost”命令的

SeqControl 编号。 此命令在任何处理操作之前暂停软件

定序器，并等待在引用的发射结束时设置 DMAControl

对象中的“已发射但未处理”标志。 该标志将保持设置状

态，直到给出“markTransferProcessed”的

SeqControl 命令。 等待时长有超时，由

Resource.VDAS.dmaTimeout 设置，默认为 1000 毫秒。

“jump”命令有一些特殊的特征。 当序列运行时，软件独立于硬件序列器处理事件列表。 在软件中，处理“跳转”命令返回到序列中的第一个事件实现返回到 Matlab 环境，以允许检查用户界面事件。 如果没有要处理的用户界面操作，Matlab 将再次调用 runAcq 处理例程并继续处理第一个事件。 如果“跳转”命令转到某个其他事件而不是第一个事件，则不会返回到 Matlab（这允许序列中的循环没有返回到 Matlab 的开销）。 然而，在某些情况下，人们可能希望返回 Matlab 进行不返回到第一个事件的跳转。 在这些情况下，使用“jump”命令指定“exitAfterJump”条件将强制返回到 Matlab。 对 runAcq 的下一次调用将在“跳转”事件之后的下一个事件中恢复事件处理。 如果只是想在事件序列的某个时刻返回到 Matlab，可以使用“returnT oMatlab”命令。 在任何重复的序列中应该至少有一个返回到 Matlab，因为没有返回，序列就不能退出。

'cBranch'（条件分支）命令有一个 Matlab 函数来设置导致分支被采用的 'bFlag' - 'setConditionalBranchFlag'。 此函数从 Matlab 工作区调用，可用于使硬件定序器根据来自 GUI 控件的用户输入分支到不同的序列事件。 要同时使软件定序器分支，请在 Matlab 工作区中将 Control.Command 字符串设置为“setBFlag”。 下一次返回到 Matlab 将选择此命令并在软件定序器中设置 BFlag。 需要注意的是，硬件和软件定序器不一定会同时分支，这可能会导致同步序列出现问题。 如果硬件和软件定序器需要同时从分支点开始，可以在分支的第一个事件中使用‘sync’命令，无需其他操作。

“triggerIn”命令指定后面板上 BNC 中两个触发器的预期条件。 这些条件在 SeqControl.condition 属性中提供并且是：

**'Trigger\_1\_Falling'**

**'Trigger\_2\_Falling'**

**'Trigger\_1\_2\_Falling'**

**'Trigger\_1\_Rising'**

**'Trigger\_2\_Rising'**

**'Trigger\_1\_2\_Rising'**

系统将等待指定条件发生，并在 SeqControl.argument 属性中提供超时值，范围从 0 到 255，其中每个计数代表 250 毫秒的时间增量。 默认值为 0，表示系统将永远等待触发条件发生。

**“triggerOut”**命令在非采集事件中编程时，在序列中遇到该事件时立即发送，但要符合以下指定的条件。 当在与采集（tx 或 tx/rcv）相同的事件中编程时，触发器将在发射延迟时间段开始之前发送，其中包括等待先前编程的“timeT oNextAcq”到期。 有几个条件可应用于触发输出：1) ‘syncADC\_CLK’将触发输出同步到指定 A/D 时钟的边沿。 2) ‘syncSYNC\_CLK’同步到 SYNC 时钟，它是 A/D 时钟的 1/4。 如果系统触发自身和另一个 Vantage 单元，则必须使用此选项。 3) ‘syncNone’（默认）表示触发输出在触发预定时间后尽快生成。 如果希望在 TX/接收周期之前的某个时间触发外部事件，请将“triggerOut”命令放置在采集事件之前的单独事件中，并在其后跟一个具有所需时间周期的“noop”命令。

触发输出脉冲也可以从相关发射/接收事件的开始延迟参数字段中指定的可选时间延迟。 该参数必须是 0 到 1,048,575 之间的整数值（相当于 20 位无符号整数），代表 250 MHz 时钟周期 (4 ns) 的计数，最大延迟约为 4 毫秒。 这种可编程延迟允许在采集开始时对触发输出脉冲进行精确的定时控制，包括将触发脉冲延迟到足以允许设置后续等待触发输入事件的时间，该事件将检测到该触发脉冲并继续序列。 如果未指定参数，将使用默认值零延迟。

**“DMA”**命令只能由运行时软件设置，并将替换任何用户指定的“transferToHost”命令。 ‘DMA’命令（‘DMA’代表直接内存访问）指定将采集模块上本地内存地址范围内的数据发射到主机中RcvBuffer的内存中。 在提供发射详细信息的 SeqControl.argument 属性中引用了 DMAControl 结构（也由运行时软件创建）。 DMA 编程的复杂性已经从用户那里抽象出来，用户只需要在获取的数据应发射到 RcvBuffer 时管理其序列中的时间。 这是使用“transferToHost”命令完成的。

**‘transferToHost’ programming （‘transferToHost’编程）**

在序列列表中想要将获取的数据发射到主机 RcvBuffer 的事件中（通常在一帧的末尾），用户指定一个“transferToHost”SeqControl 命令。 在加载序列之前，运行时软件按顺序遍历事件列表，当它发现带有“transferToHost”的 SeqControl 命令的事件时，它会为事件列表中**前一个采集帧**的采集数据生成一个 DMAControl 对象。 当硬件定序器收到此事件时，它会生成一个中断，以启动到主机 RcvBuffer 的 DMA 发射。 请注意，序列中的每个“transferToHost”命令都必须有一个唯一的 SeqControl 结构； 换句话说，您不能在序列中的多个点“重复使用”相同的 SeqControl(n).command=‘transferToHost’，因为每个‘transferToHost’都与唯一的 DMA 操作相关联。 从包含“transferToHost”命令的事件开始，软件向后搜索以前的事件，查看获取事件引用的接收结构。 第一个找到的 Receive 的 RcvBuffer 编号和帧编号被捕获并与之前的 Event Receives 进行比较。 然后使用具有相同 Receive.bufnum 和 Receive.framenum 编号的所有 Receives 来组成发射的 DMAControl 结构。 因此，在序列中不要在“transferToHost”操作之前混合来自不同 RcvBuffers 或 RcvBuffer 帧的采集非常重要。 如果这样做，发射的数据将仅来自具有相同 Receive.bufnum 和 Receive.framenum 属性的最后一个事件。 当“transferToHost”DMA 被编程到硬件定序器中时，会在序列的下一个 DMA 之前设置两个条件停止命令。 第一个是“等待处理”的条件停止，其功能将在下面的第 3.6.1 节中描述。 第二个是“正在进行的 DMA”的有条件停止。 此停止在“transferToHost”DMA 启动时设置，并在 DMA 完成时清除。 这可以防止在前一个 DMA 尚未完成的情况下处理新的 DMA。 如果 DMA 跟不上采集，此停止可能会导致错过“timeToNextAcq”消息（如下所述）。

**“transferToHost”**命令通常在没有“waitForProcessing”条件的情况下使用。 在这种情况下，硬件定序器将执行 DMA，而不设置“DMA 进行中”停止以外的停止条件； 如果所有“transferToHost”命令都以这种方式设置，则硬件定序器将以其自己的速率异步运行，与处理无关。 在这种操作模式下，可以对软件进行编程以重建和处理最近发射的帧（使用 Recon.rcvBufFrame = -1），这将以处理允许的速度处理帧，最高可达采集帧速率。 使用这种编程方法的设置脚本称为“异步”。 异步编程可用于一些有趣的采集序列，其中采集可以以非常高的帧速率运行，将帧捕获到多帧 RcvBuffer 中，并且在实时采集期间，软件会尽可能快地处理帧，尽可能快地跳过采集的帧 需要生成帧的实时显示。 在 RcvDataLoop 模式下，所有获取的帧都将被处理，因此 RcvBuffer 帧以处理帧速率播放，这可能看起来像慢动作播放。 有关如何使用此采集方法的示例，请参阅 SetUpL11\_4vFlash 脚本。

**“sync”（同步）**命令可用于在序列中的特定事件中同步硬件和软件定序器。 此命令为两个音序器设置条件暂停，无论哪个音序器先到达同步点，都会暂停等待另一个音序器到达。 当另一个定序器到达时，条件暂停被释放，两个定序器继续序列中的下一个事件。 用户可以设置软件音序器等待硬件音序器到达的超时时间，方法是使用“同步”命令在 SeqControl 的参数字段中指定时间（以微秒为单位）。 默认超时为 500000 usec（0.5 秒）。 硬件定序器没有超时，将无限期地等待软件定序器赶上。 “sync”命令的处理方式与其他 SeqControl 命令略有不同，因为它在具有多个 SeqControl 命令的事件中的执行顺序独立于 SeqControl 命令列表中给出的顺序强制执行。 “同步”条件暂停始终在任何采集（Tx 和 Rcv）、重建或处理操作之后以及任何分支命令之前执行。 如果在没有采集操作的事件中，触发输入或输出包含在“同步”命令中，则触发操作发生在“同步”命令之后和任何分支命令之前。 如果事件执行采集操作，则触发输入或输出将置于采集操作之前，因此也置于同步命令之前。

**“timeToNextAcq”**命令用于设置采集事件之间的精确时间。 它还可用于设置仅发射事件 (Event.rcv = 0) 之间的持续时间。 持续时间在参数字段中以微秒为单位设置 (10 – 4190000)。 当当前事件的发射/接收周期开始时，会为 timeToNextAcq 的持续时间设置一个计数器，然后该计数器在后台运行，直到发生具有 TX 引用的下一个事件。 在为下一个事件开始发送/接收周期之前，检查计数器，如果仍在运行，则发送/接收周期延迟直到计数器完成（注意此延迟适用于发送延迟倒计时的开始，因为 发射延迟计数器在发射/接收周期开始时启动）。 这提供了连续采集事件之间的精确时间差。 如果新事件也有 timeToNextAcq 命令，则计数器设置为使用参数字段中的值再次运行。 如果在启动下一个事件时计时器已经完成，则会打印一条警告消息以通知用户未满足他们请求的持续时间。 出现这种情况的原因有多种：1) 在异步脚本中，不满足时序的典型原因是 DMA 时间过长。 如果在准备启动新 DMA 时前一个 DMA 正在进行中，则序列必须暂停，直到前一个 DMA 完成。 这可能会导致错过 timeToNextAcq 持续时间。 这里的解决方案是 a) 减少发射的数据量，可能通过使用较低的信号带宽 (Receive.sampleMode)，b) 将您的采集分组并使用每个 DMA 发射更多数据，从而降低与启动新 DMA 相关的开销，或者 c) 增加 timeToNextAcq 持续时间。 2) 在同步脚本中，缺少 timeToNextAcq 持续时间的典型原因是处理延迟。 如果处理时间长于编程到采集定序器中的持续时间，timeToNextAcq 持续时间将被错过。 输出警告时，它还会显示错过时间的持续时间（以微秒为单位）。 这是计数器完成后到定序器最终到达下一个采集事件的时间。 这个错过的持续时间可以用来帮助决定将“timeToNextAcq”持续时间增加多少。 （注意：在由于处理延迟而重复错过计时的同步脚本中，错过的持续时间可能比实际时间长，因为增加了打印警告消息的时间。）

**3.6.1 Synchronizing sequencers with data transfer commands-（将定序器与数据发射命令同步-）**

如前所述，硬件和软件定序器以它们自己的速率独立运行，硬件定序器处理发射/接收采集，软件定序器处理采集的数据。 在大多数情况下，这是一个优势，允许高帧速率采集，同时以较慢的帧速率处理最近采集的帧。 切换到 RF 循环播放，然后可以看到以处理帧速率播放的所有帧。 然而，有时需要同步两个定序器的动作。 例如，人们可能想要使用不同的采集方法采集帧，并对每一帧进行独特的处理。 在这种情况下，我们希望硬件定序器与处理同步，以便我们知道我们正在处理正确的采集集。 还有一些与实时运行时的异步处理有关的问题，例如：1) 如何确保软件只处理完整的采集帧，以及 2) 如何防止 DMA 发射覆盖正在发射的帧中的数据 正在处理？ 为了解决这些问题和其他问题，我们可能需要同步获取和处理数据。 用于同步处理的 SeqControl 命令是：

**Command Condition Argument**

**‘transferToHost‘ ‘waitForProcessing‘ SeqControl index ‘waitForTransferComplete‘ - SeqControl index ‘markTransferProcessed‘ - SeqControl index**

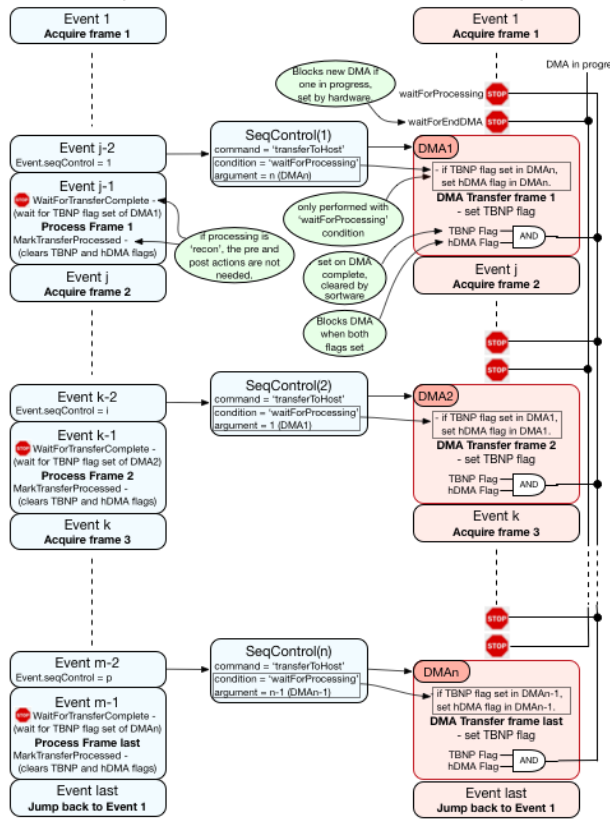
理解同步过程的一种方法是可视化属于由“transferT oHost”命令编程的 DMA 的两个标志，一个我们将命名为“transferredButNotProcessed”或“TBNP”，另一个命名为“holdDMA”或“hDMA”。 在使用硬件运行采集序列之前，所有标志都被清除。

当“transferToHost”DMA 发射完成时，“TBNP”标志被设置。 在处理来自发射的数据之前，软件可以读取 TBNP 标志并等待为特定的“transferToHost”DMA 设置 TBNP 标志，使用“waitForTransferComplete”SeqControl 命令。 该命令的参数是一个 SeqControl 索引，它指定了我们希望等待其标志的‘transferT oHost’DMA。 当处理完成时，可以使用“markTransferProcessed”SeqControl 命令重置标志，再次在参数中引用 SeqControl 索引和“transferToHost”DMA，在其中重置 TBNP 标志。 重要的是要注意，如果标志在处理后没有重置，它保持设置状态，如果相同的处理重复另一个'waitForTransferComplete'动作，它会认为新数据已经发射，而实际上旧数据仍然是 在缓冲区中。）

“transferToHost”命令中的“hDMA”标志是一个标志，当“TBNP”标志也被设置时，它将停止任何新的“transferToHost”DMA 操作。 当两个标志都设置时，硬件定序器将在任何新的“transferToHost”DMA 操作之前暂停并等待两个标志被软件清除。 当给出“waitForProcessing”SeqControl 命令时，“hDMA”标志在“transferToHost”DMA 命令中设置。 此命令的参数是 SeqControl 的索引，指定要在其中设置标志的“transferToHost”DMA。（如果引用的 SeqControl 不具有“transferToHost”的命令属性，则会生成错误。）“waitForProcessing”条件 可以将硬件定序器操作与软件处理同步。 当在序列中遇到具有此条件的“transferToHost”DMA 命令时，将检查引用的“transferToHost”DMA 的“TBNP”标志，如果设置，则设置该 DMA 中的“hDMA”标志。 （如果未设置“TBNP”标志，则不采取任何操作。）这将导致硬件定序器在到达下一个“transferToHost”DMA 时暂停。 硬件定序器将保持暂停状态，直到软件定序器处理“markTransferProcessed”SeqControl 命令，软件定序器使用包含设置的“hDMA”和“TBNP”标志的“transferToHost”引用 SeqControl 索引。 然后重置两个标志，释放硬件定序器以执行下一次发射。

例如，处理与硬件同步发生的多帧序列的操作如下图 3.6.1 所示。 控制结构将对两个或更多帧起作用，并且将最近获取和发射的帧的处理与下一帧的获取重叠。 在事件序列中，带有条件“waitForProcessing”的“transferToHost”命令被放置在每帧的最后一个采集事件中或紧随其后。 对于带有“waitForProcessing”条件的 SeqControl“transferToHost”的参数属性，我们设置了前一个“transferToHost”SeqControl 结构的编号。 【为什么是上一帧？ 我们不希望硬件定序器比软件处理提前一帧以上，所以我们让它在发射当前帧后暂停，直到处理完前一帧。]（如果这是第一个'transferToHost'SeqControl结构 序列，设置最后一个'transferToHost'SeqControl结构的编号，然后跳回到序列的开头。）使用'transferToHost'SeqControl结构的这种安排，在遇到'transferToHost'SeqControl命令时，硬件定序器是 设置为在下一次 DMA 发射之前暂停。 如果通过引用的“transferToHost”SeqControl 命令（之前的发射）发射的数据尚未通过“markTransferProcessed”操作标记为已处理，则硬件定序器将暂停并等待处理完成。 当“markTransferProcessed”动作发生时，它会取消硬件定序器暂停。

需要注意的是，对于重建处理，处理软件会自动检查‘TBNP’标志，对于发射到 Recon 正在处理的 RcvBuffer 帧的 SeqControl ‘transferToHost’，并等待标志被设置。 此外，在重建处理结束时，‘TBNP’标志被清除，如果‘hDMA’标志也被设置的话，它也会被清除。 因此，如果对获取和发射的数据帧的唯一处理是使用 Recon 对象进行图像重建，则无需将 Recon 事件与“waitForTransferComplete”和“markTransferProcessed”SeqControl 操作放在一起。 还允许将 Recon 结构的 Recon.rcvBufFrame 属性设置为 -1，这意味着“处理最近获取的帧”，因为软件将等待“TNBP”标志被设置，并且当它被设置时 ，最近获取的帧将是与设置标志的发射相关联的帧。 这消除了为处理每个帧指定不同的 Recon 结构的需要。



**3.7 Event Objects (Continued) (事件对象（续）)**

虽然在本节开头进行了介绍，但并未涵盖 Event 对象的所有功能。 下面再次给出结构：

**event =**

info: string 'optional descriptive text for this event'

'此事件的可选描述性文本'

tx: double # of TX structure array to use for transmit

用于发射的 TX 结构数组的#

rcv: double # of Receive structure array to use for rcv.

用于rcv 的接收结构数组的#

recon: [up to 16 doubles] # of Recon structure array for recons

用于重建的重建结构数组的#

process: double # of Process structure array to use.

要使用的进程结构数组的#

seqControl: [up to 3 doubles] # of SeqControl structure array to use

要使用的 SeqControl 结构数组

在上面的定义中，请注意 recon 属性可以是单值的，也可以是索引数组。 异步序列需要在同一事件中支持多次重建，其中 Recon 适用于最近获取的 RcvBuffer 数据帧。 当在同一事件中遇到多个侦察都引用最近获取的帧 (Recon.rcvBufFrame = -1) 时，最新帧的确定仅对第一个侦察执行，并重新用于其他侦察。 这可确保所有 Recons 使用相同的帧，这对于 2D 和多普勒血流成像等混合模式采集至关重要。

在同一事件中支持多个 SeqControl 索引允许需要与单个事件关联的多个控制操作。 例如，人们可能希望为特定事件同时指定“timeToNextAcq”命令和“transferToHost”命令。 当在同一个事件中给出多个 SeqControl 索引时，它们通常按给定的顺序执行，尽管顺序通常与所采取的操作无关。

事件对象应该是脚本中定义的最后一个序列对象。 这是为了在将对象加载到系统中时检查所引用对象的索引是否存在。

在示例脚本中，会注意到事件索引是用变量 n 定义的，该变量在每个事件后递增。 这允许轻松地添加或删除序列中的事件，而无需重新编号后续事件。

1. **Verasonics Script Execution (VSX)（Verasonics 脚本执行）**

一旦创建了序列对象并将其捆绑为 .mat 文件格式，就可以使用“VSX”Matlab 程序加载和执行它。 该程序执行许多功能，包括：

1）将Sequence Object加载到Matlab环境中，解析各种结构，检查是否缺少必需的参数并编程一些默认属性。 （有些属性直到初始化过程的后期才添加，在调用 mex 函数“runAcq”之后。）

2) 检测到 VDAS 硬件的存在，如果找到，硬件将“打开”并由 VSX 应用程序“拥有”。 如果没有检测到硬件，脚本将以模拟模式执行。

3) 对于大型机箱 Verasonics 系统（2 或 4 板），检测是否存在连接到脚本指定的扫描头连接器（默认为 1 或左侧连接器）的传感器，并读取其 ID。 （大多数 Verasonics 和 HDI1000 - 5000 探头在探头手柄中包含一个用于识别探头的个性化 eprom。）如果未插入传感器，或者探头 ID 与用户脚本中指定的探头名称不匹配，VSX 将产生错误并退出。 用户脚本可以为没有个性化 EPROM 的传感器使用“自定义”Trans.name 条目，在这种情况下，VSX 将跳过探头 ID 检查。

4) 打开一个图形用户界面窗口，具有一些基本的控制功能。 安装脚本 .mat 文件中定义的任何其他 GUI 控件都会添加到界面窗口中。

5) 根据设置脚本定义的 Resource.DisplayWindow 定义，打开一个或多个显示窗口。

6) 进入一个执行循环，该循环将一直持续到 GUI 窗口关闭，或序列事件列表完成。 （大多数序列在序列末尾跳回起点，因此序列无限重复。）在此执行循环中，调用名为“runAcq”的 mex（Matlab 外部）函数，它执行所有处理 顺序事件。 ‘runAcq’从序列中的指定事件（Resource.Parameters.startEvent）开始，并继续执行事件，直到满足以下条件之一：

**-** 到达序列的末尾，并且没有跳回到开始。 在这种情况下，VSX 会自动进入“冻结”状态。 通过按“冻结”GUI 切换按钮解冻系统将再次运行序列。

**-** 执行‘jump’SeqControl 命令回到序列的开始，在这种情况下，“runAcq”在跳转后将控制权返回到 Matlab 环境。 需要返回到 Matlab 环境来处理在事件处理期间可能已激活的任何 GUI 控件。 如果在 Matlab 环境中没有要采取的操作，则会再次调用“runAcq”并从序列的开头继续处理。

**-** “‘jump’SeqControl 命令出现在事件列表的主体中，并设置了“ExitAfterJump”条件。 在这种情况下，runAcq 将返回到 Matlab，并保存跳转到的事件编号，以便下次执行 runAcq 时序列将从该事件开始。 跳转命令的这种修改允许在按顺序执行子循环时返回到 Matlab。

- 在序列中遇到“returnT oMatlab”SeqControl 命令。 当序列持续时间很长时使用此命令，等到结束返回 Matlab 会导致对 GUI 控件的响应太慢。 下次调用 runAcq 时，将继续执行下一个事件。

- 执行“停止”SeqControl 命令，在这种情况下，“runAcq”立即返回到 Matlab 并进入冻结状态。

初始化后，VSX 和 runAcq 的典型运行时行为是跳回其开始事件的序列，VSX 从其“while”循环中重复调用 runAcq，每次调用处理序列中的所有事件。 runAcq函数处理从开始事件到最后一个事件的所有事件和数据处理，找到跳回，触发返回Matlab。 在 Matlab 中，唯一的操作是检查任何需要注意的 GUI 事件，然后使用适当的输入参数再次调用 runAcq。

* 1. **（Defining GUI Controls）定义 GUI 控件**

要在运行时更改序列对象参数，正确的方法是设置一个 GUI 对象，该对象在激活时执行回调例程。 GUI 对象定义和回调例程通常可以在构建序列对象 .mat 包的安装脚本中定义。 回调例程可以在 Matlab 环境中设置一个动作值，VSX 在调用 runAcq 期间读取该值，从而触发 Matlab 环境中的某些动作。 某些内置 GUI 控件使用此方法，但不建议用户使用，因为它需要修改 VSX 以合并您的 GUI 操作。 由于 VSX 将随着新软件版本的更新而更新，因此您的 GUI 修改必须不断应用于新代码。

在不修改 VSX 的情况下提供运行时控件的推荐方法是在安装脚本中设置 GUI 对象和回调函数，以便在加载脚本时将其合并到 VSX 的 GUI 窗口中。 回调函数在激活时可以在 Matlab 环境中修改脚本中指定的适当序列结构，但这些修改在重新加载到 runAcq 处理函数和硬件之前不会产生任何影响。 要执行重新加载，GUI 控件还需要在 Matlab 环境中设置一个 Control 结构，该结构将在 VSX 的下一次调用时传递给 runAcq，指定要更新的结构或属性。 Control结构传递给runAcq后会自动清除，使控制动作只执行一次。

修改运行时参数时，请务必记住 VSX 可能已将属性添加到原始脚本结构中，特别是用于对硬件进行编程的 VDAS 属性。 对于添加了 VDAS 组件的结构，例如 TW、TX 和 Receive，当 runAcq 检测到结构发生变化以更新添加的 VDAS 组件时，函数“update.m”将被调用。 几乎在所有情况下，用户都不应该直接修改 VDAS 参数。

为了在脚本中定义 GUI 控件，提供了一个名为 UI 的结构。 该结构具有以下字段。

UI =

Statement string 在加载时执行的 Matlab 语句。

Control cell Matlab uicontrol 命令的输入参数。

Callback cell Quoted statements for callback function. 回调函数的引用语句。

handle handle Handle for callback(s) created (set by VSX). 创建的回调句柄。

UI.Statement 属性定义了一个字符串，该字符串将在创建 UI 控件时作为 Matlab 语句执行。 这对于执行初始化 GUI 或其组件所需的任何命令很有用。 UI.Statement 字段还可用于在加载和运行用户序列之前执行命令。 例如，可以使用以下语句为特定发射配置文件（通常默认为最小值）设置初始高电压。

**% 将配置文件一的 TPCHighVoltage 设置为 40V**

**UI(1).Statement = '[result,hv] = setTpcProfileHighVoltage(40,1);';**

**UI(2).Statement = 'hv1Sldr = findobj(''Tag'',''hv1Sldr'');';**

**UI(3).Statement = 'set(hv1Sldr,''Value'',hv);';**

**UI(4).Statement = 'hv1Value = findobj(''Tag'',''hv1Value'');';**

**UI(5).Statement = 'set(hv1Value,''String'',num2str(hv,''%.1f''));';**

上述语句将配置文件的高压设置为 40 伏，并设置高压滑块和文本字段以反映新值。

UI.Control 属性是属性/值对的元胞数组，将在 Matlab uicontrol 命令中使用，或与自定义 Verasonics 控件一起使用（请参阅下面的下一节）。 例如，定义滑块控件通常需要的属性可以指定如下：

**UI(1).Control = {'Style','slider',...**

**'Position',[0.375,0.1,0.25,0.030],... % position on UI**

**'Max',10.0,'Min',1,'Value',5.0,...**

**'SliderStep',[0.025 0.1],...**

**'Tag','mySlider',...**

**'Callback',{@mySliderCallback}};**

请注意，位置单位是相对于整个 GUI 窗口大小以相对值定义的。 字体也应该以相对单位定义，以便文本的大小能够填充定义的控件大小。 VSX 在创建 GUI 时会将这些属性传递给 Matlab uicontrol 函数，并在 UI.handle 属性中设置控件的句柄。 该句柄可用于通过其他函数或回调找到控件，或者上面定义的控件也可以通过其标签“mySlider”找到。

最后，UI.Callback 属性提供与控件关联的回调函数的元胞数组定义。 第一个单元格指定回调函数的原型，其名称应与 UI.Control ‘Callback’ 属性中指定的名称相匹配。 以下单元格指定回调函数的行，VSX 将在系统的 tmp 目录中创建该函数。 （Matlab 为文件定义了一个临时目录，由函数“tempdir”访问。）回调函数通常修改序列对象属性，并使用 Control 结构（在第 4.2 节中描述）将其传递给序列执行程序 runAcq。 例如，如果上面定义的滑块用于控制第二个 Process 结构的处理增益（pgain）属性，则回调定义如下：

**UI(2).Callback = {'mySliderCallback(hObject,eventdata)',...**

**' ',...**

**'pgn = get(hObject,''Value'');',...**

**'Control = evalin(''base'',''Control'');',...**

**'Control.Command = ''set&Run'';',...**

**'Control.Parameters = {''Process'',2,''pgain'',pgn};',...**

**'assignin(''base'',''Control'', Control);',...**

**'return'};**

为了更容易地创建和解释长回调函数，提供了一个实用函数，它将两个分隔符之间的文本转换为元胞数组字符串，称为“text2cell”，其调用输入是要读取的文件名（可选；如果缺少默认调用 脚本文件）和分隔符字符串。 这允许在可以自动解释为 UI.Callback 元胞数组的安装文件末尾编写普通的 Matlab 代码。 为了防止在运行 SetUp 脚本时执行此代码，在定义了 SetUp 脚本中的所有结构之后放置一个“return”，然后在“return”之后，将回调函数的代码行放在两个之间 指定分隔符的实例。 仅在要编码的文本的第一行使用函数原型，因为包含关键字“function”会产生 Matlab 错误，因为脚本中不允许定义函数。 比如上面的UI(2).Callback函数可以这样写：

**return % 将此 return 放在脚本末尾以防止执行下面的代码**

**%CB#2**

**mySliderCallback(hObject,eventdata)**

**pgn = get(hObject,'Value');**

**Control = evalin('base', 'Control');**

**Control.Command = 'set&Run';**

**Control.Parameters = {'Process',2,'pgain',pgn};**

**assignin('base', 'Control');**

**return**

**%CB#2**

定界符应独占一行，并以注释字符开头，紧接着是定界符字符串。 要将上述功能行插入到 UI(2).Callback 中，请使用以下语句：

**> UI(2).Callback = text2cell('%CB#2');**

定界符可以是任何字符串，但应该确保它不会在脚本文件的其他地方使用，除了函数调用本身。 （在搜索要编码的文本的开头时，忽略用单引号包围的定界符。）例如，您可能希望在定界符字符串中包含函数名称，例如：

**'%myCallback#2'**

如果要对调用脚本以外的文件中的文本进行编码，请将文件名（及其路径，如果不在默认路径中）指定为 text2cell 中的第一个输入参数，例如：

**> UI(2).Callback = text2cell('myCodeFile.m','%CB#2');**

**4.1.1 Verasonics UI Controls（Verasonics 用户界面控件）**

为了便于在 Verasonics GUI 上创建用户控件，较新的软件版本提供了一些内置的 UI 功能，这些功能可以放置在某些预定义的位置。 目前可以选择的VsXXX控件有：VsSlider、VsPushButton、VsT oggleButton、VsButtonGroup。 这些控件由 UI.Control 单元格数组中的自定义位置和“样式”指定，如下所示：

**UI(1).Control = {'UserXX', 'Style', 'VsXXX', ... }**

当前可用于用户控件的位置如图 4 所示。 4.1.1 下面。 上面语句中用于“UserXX”的标签指定了控件的位置。 例如，要在中心列的最低位置创建一个控件，请将“UserB1”用作 UI.Control 中的第一个条目。 请注意，一些用户控件位置与 VSX 放置在 GUI 中的一些默认控件重叠。 这是因为重叠控件是有条件的——仅当脚本中存在 Recon 结构时才添加 Speed of Sound 控件，并且仅当定义了 ImageBuffer 时才添加 Zoom、Pan、Compression、Reject 和 Persistence 控件。 即使在添加了默认控件的脚本中，如果需要，也可以将用户控件放置在默认控件之上。

Verasonics 用户控件由 VSX 在序列加载时创建，并使用默认标记和回调创建。 回调基本上是代码前导码，有助于管理控制操作并为随后的用户提供的代码提供值。 用户提供的代码放在 UI.Callback 元胞数组中，并在创建回调函数时添加到前导代码中。 创建 UI.Callback 元胞数组时，仅包含所需的代码语句。 VSX 会根据 UserXX 位置将函数命名为‘UserXXCallback.m’，函数定义和调用参数自动包含在序言中。

如上一节所述，用户提供的回调函数代码可以在 SetUp 脚本末尾的指定分隔符之间编写，并使用以下语句放置在 UI.Callback 单元格数组中：

**> UI(n).Callback = text2cell('%CB#n');**

在这种情况下，分隔符是“%CB#n”，n 是 UI 结构的编号。

虽然可以在运行 VSX 时将回调函数和外部函数编码为位于 Matlab 路径中的单独 Matlab 函数文件，但建议在 SetUp 文件中对这些函数进行编码，因为它将回调函数和外部函数与 需要它们的设置脚本，并且无需管理大量需要在主目录中使用不同名称的小函数文件。

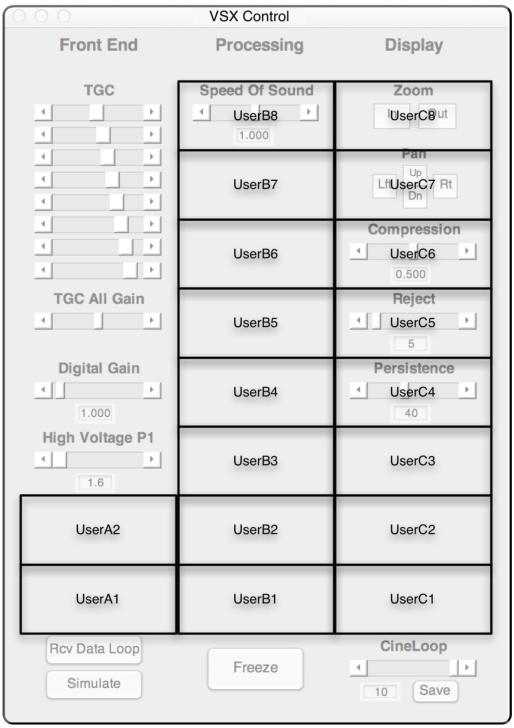


图 4.1.1 可用于 Verasonics 自定义控件的位置

**VsSlider -** 当使用“VsSlider”的“样式”指定 UserXX 位置时，VSX 将创建三个 UI 控件 - 一个文本框、一个滑块和一个链接到滑块操作的编辑窗口。 可以提供 VsSlider 的唯一属性/值字段以简化创建。 这些属性/值组合是：

**Label, Text string**

**SliderMinMaxVal, [Min,Max,initialVal]**

**SliderStep, [small, large ] default [0.01, 0.1 ]**

**ValueFormat, '%3.0i' default format for integer field整数字段的默认格式**

文本、滑块和编辑控件的句柄位于 UI.handle(1)、UI.handle(2) 和 UI.handle(3) 中。 还为滑块（“UserXXSlider”）和编辑框（“UserXXEdit”）创建了标签。 对于创建的编辑框，ValueFormat 被复制到 UIControl 的 UserData 字段，可以从更改值的回调访问它。

当滑块或编辑框的值改变时，UI(n).Callback 函数被执行，自动更新滑块和/或编辑框的值。 VSX 创建的回调名为 UserXXCallback.m 并具有下面的前导代码以同步滑块和编辑值。 在 UI(n).Callback 中指定在序言之后要执行的用户回调代码，并且可以访问名为“UIValue”的变量以从滑块或编辑框中获取更改后的值。

**function UserXXCallback(hObject, eventdata);**

**Cntrl = get(hObject, 'Style');**

**if (strcmp(Cntrl,'slider')**

**UIValue = get(hobject,'Value');**

**h = findobj('Tag', [UserXx 'Edit']);**

**set(h,'String', num2str(UIValue,get(h,'UserData'));**

**else**

**UIValue = str2num(get(hobject,'String'));**

**h = findobj('Tag', [Userxx 'Slider']);**

**max = get(h,'Max');**

**min = get(h,'Min');**

**if (UIValue > max)**

**UIValue = max;**

**set(hobject,'String', num2str(UIValue,get(hobject, 'UserData'));**

**end**

**if (UIValue < min)**

**UIValue = min;**

**set(hobject, 'String', num2str(UIValue,get(hobject, 'UserData'));**

**end**

**set (h, 'Value', UIValue);**

**end**

**< User provided statements >**

例如，为序列编程教程中描述的通道选择器滑块创建 UI 控件简化为以下内容：

**UI(1).Control = {'UserB1','Style','VsSlider','SliderMinMaxVal',[1,nr,32]};**

**UI(1).Callback = {'assignin(''base'',''myPlotChnl'',UIValue)'};**

**VsPushButton -** 当使用“VsPushButton”的“样式”指定 UserXX 位置时，可用属性为：

**Label, Text String 要放置在按钮控件中的标签。**

UI(n).Callback 代码在释放按钮且光标位于按钮框架中时执行。 在这种情况下，回调序言如下：

**Function(函数) UserXXCallback(hObject, eventdata);**

**VsToggleButton -** 当 UserXX 位置指定为“VsT oggleButton”的“样式”时，可用属性如下：

**Label, Text String 要放置在切换按钮控件中的标签。**

切换按钮的 UI(n).Callback 前导码在 UIState 变量中提供切换按钮的状态。

**function UserXXCallback(hObject, eventdata);**

**VsButtonGroup -** 当使用“VsButtonGroup”的“样式”指定 UserXX 位置时，可用属性为：

**Title, Text String 按钮组的标题（位于边界框的顶部）。**

**NumButtons, N 组中的按钮数 (default = 2).**

**Label, [Text1,…T extN] 按钮 N 按钮的标签。**

对于 2 个按钮，（默认）按钮组的大小与位置框的大小相同。 对于更多按钮，按钮组向下增长并与下面的用户位置重叠。 UI(n).Callback 代码在按下按钮时执行。 变量 UIState 被设置为按下的按钮的编号。 在这种情况下，回调序言如下：

**function UserXXCallback(hObject, eventdata);**

**S = get(eventdata.NewValue,'Tag');**

**UIState = str2double(S(18));**

**4.1.2 Debugging VSX Generated UI Controls (调试 VSX 生成的 UI 控件)**

调试 VSX 生成的 UI.Callback 或 EF.Function 需要一些额外的步骤，因为这些函数在 VSX 运行之前不存在，并且会被每次新运行的 VSX 覆盖。 因此，不能在运行 VSX 之前设置断点。 最简单的调试方法是在回调或外部函数中想要中断执行的位置插入“keyboard”语句。 当运行 VSX 并调用回调函数或外部函数时，键盘命令会将控制返回到 Matlab 命令行。 此时，可以使用以下命令在 Matlab 编辑器中调出该函数：

**>edit UserXXCallback.m**

该函数用于 Verasonics 回调或外部函数

**>edit myExtFuncName.m**

Matlab 将在函数的“keyboard”语句处显示执行点。 然后可以使用 Matlab 调试命令单步执行代码并观察执行情况。 如果要对代码进行更改，当然应该在 SetUp 脚本中的函数定义中进行这些更改，并在再次运行 VSX 之前重新编译脚本。 当 VSX 生成的功能正常工作时，可以删除“keyboard”语句。

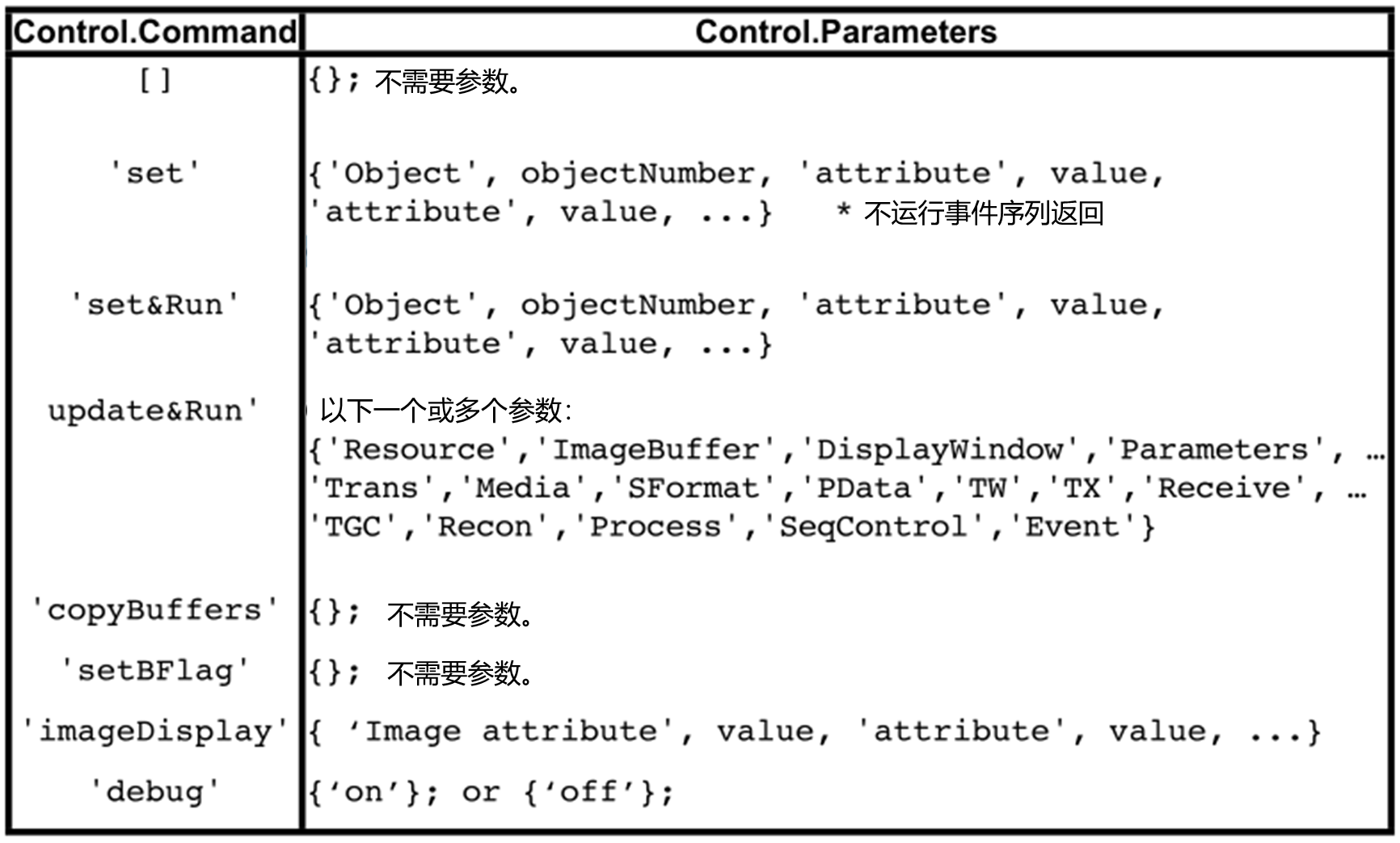
**4.2 设置控制参数**

runAcq 函数有一个输入参数，它是一个名为“Control”的结构数组。 这个结构有两个字段：

**Control.Command** – 表示命令的字符串。

**Control.Parameters** – 一个元胞数组，其中包含给定命令所需的各种参数。

有效命令及其相关参数如下：



当不需要操作时，Control.Command 设置为 [ ]（空状态）。 这也是在 runAcq 执行 Control.Command 后设置的条件，因此命令只执行一次。

一次调用 runAcq 可以执行多个 Control.Command。 这是通过定义多个带有索引的控制结构来实现的，例如

**Control(1).Command = ‘set&Run’;**

**Control(1).Parameters = {'ObjectX', objectNumber, 'attribute', value};**

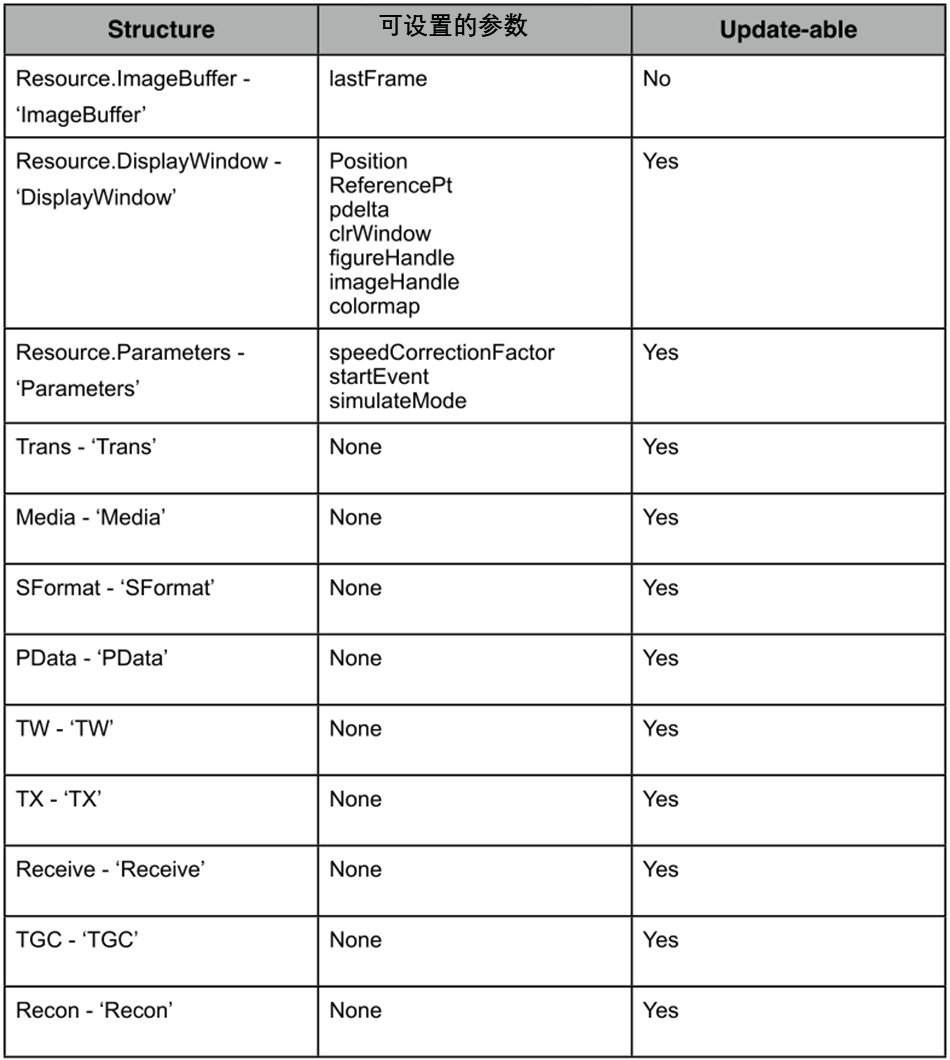
**Control(2).Command = ‘set&Run’;**

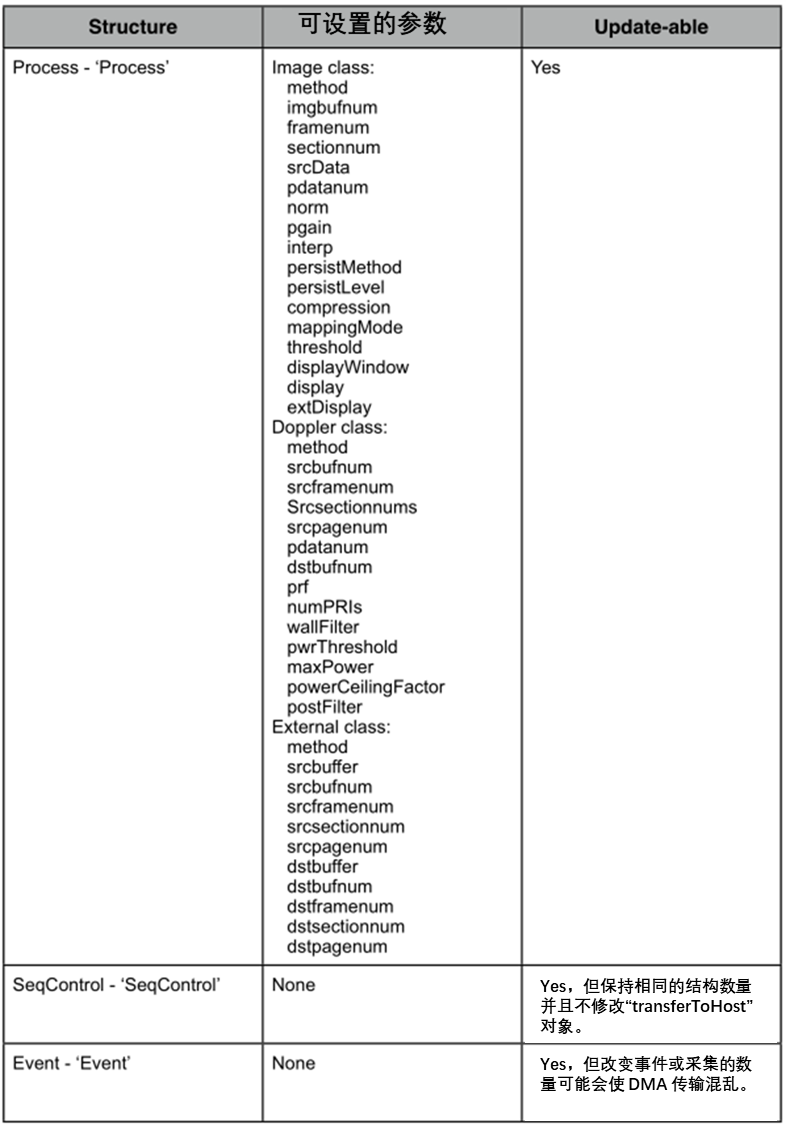
**Control(2).Parameters = {'ObjectY', objectNumber, 'attribute', value};**

在这种情况下，每个控制命令从索引 1 开始依次执行。序列事件（对于不立即返回的命令）仅在所有控制命令完成后执行。

要设置同一对象的多个属性，请注意“set”命令可以采用可变数量的属性、值对。 另请注意，如果您使用“set”命令设置属性，该属性将不会在 Matlab 环境中更新。 这可以用作一项功能，允许您在 runAcq 环境中设置各种参数，然后通过对相同对象进行“更新”来恢复原始状态。 如果您设置了一些参数并稍后尝试更新同一对象，它也可能导致意外行为。 更改对象的属性时，通常最好先在 Matlab 环境中更改属性，无论您打算执行“更新”还是“设置”操作。

目前，并非所有结构属性都可以使用“set”或“set&Run”命令更改，并且有一些结构，例如在运行时无法更新的资源结构。 下表列出了当前支持的具有可设置属性或可以完全更新的结构（Control.Parameters 中使用的结构名称用单引号括起来）。 （有关示例，请参见第 4.1 节）。





**5. 示例脚本—“SetUpL11\_4vFlash.m”**

此脚本生成获取和处理超声图像所需的所有序列对象，这些图像由典型线性阵列换能器（在本例中为 Verasonics L11-4v 换能器）的未聚焦发射脉冲生成。 换能器中的每个元件都同时发射一束超声波，产生从换能器传出的平坦波前。 在接收时，所有 128 个通道都用于从全视场采集回波数据。

此序列说明了异步采集和处理的使用，其中采集的帧速率不一定与处理的帧速率匹配。 可以以相当高的速率将帧采集到内存中，而处理速率可以达到实时观看的合理水平，例如 30fps。 只需稍作改动，即可转换序列以提供同步采集和处理。 将为获取的 RF 数据和处理后的图像数据创建一个电影循环，其中任何一个都可以在回放模式下进行查看。

由于这是一个相当简单的脚本，因此可以在没有大纲的情况下对相关结构进行编程，但通常最好先描述序列中所需的操作。

简单闪光（平面波）成像的序列动作：

1. 发射单个脉冲，所有发射延迟设置为 0，产生一个平面波，使整个图像空间无声。以

换能器的中心频率发射。

2. 使用 200% 带宽奈奎斯特采样和默认数字滤波器从所有通道接收数据。 对 Receives

进行编程，将数据发射到 100 帧的 RcvBuffer 中。

3. 对采集之间的时间进行编程，将采集帧速率设置为 100 fps。

4. 每一帧采集完成后，将采集数据的帧发射给主机。

5. 重建最近获取的图像。

6. 使用大部分默认参数处理重建图像并显示生成的图像。

7. 捕捉一个 20 帧显示的电影回放以供审查。

我们现在可以继续编写我们的安装脚本。 Courier 字体的线条是 Matlab 代码，而蓝色斜体的线条是对编程步骤的解释。

**5.1 Define system parameters（定义系统参数）**

首先，请务必使用“全部清除”语句启动您的脚本，以消除 Matlab 工作区中可能存在的变量。 这很重要，因为定义的新变量和结构将在设置脚本的末尾保存。 接下来，最好在文件的顶部定义一些全局和系统参数。 参数定义如下： 定义脚本的一些基本参数。 保存预设时，以 P 开头的参数将与其他值一起保存。

**clear all**

**P.startDepth = 5; % 以波长定义采集范围**

**P.endDepth = 192;**

**% 定义系统参数**。

定义系统中可用的发射机数量。 对于 128 通道 Vantage 系统或使用单个连接器的 256 通道系统，此数字为 128。

**Resource.Parameters.numTransmit = 128;**  %发射通道数。

定义系统中可用的接收通道数。 对于 128 Vantage 系统或使用一个连接器的 256 通道 Vantage 系统，此数字为 128。

**Resource.Parameters.numRcvChannels = 128;** % 接收通道数。

以米/秒为单位定义要成像的介质中的声速。 对于医学成像，声速是各种组织的平均值，通常为 1540 m/s。 如果此参数的定义与默认值 1540 m/s 不同，则应在计算 Trans 中的换能器特性之前进行定义。

**Resource.Parameters.speedOfSound = 1540;** % 声速，以米/秒为单位

定义错误报告的级别。 2 级将报告大多数系统错误。

**Resource.Parameters.verbose = 2;**

**simulateMode** 属性指定您是要在模拟模式下运行脚本，还是使用从 Vantage 硬件获取的数据。 值为 0 表示您想要使用 Vantage 硬件进行采集。 如果未检测到硬件，VSX 将自动切换到模拟模式，该模式设置值为 1。在模拟模式下，您可以选择是否从您可以定义的媒介模型中模拟 RF 数据（simulateMode = 1） , 或者是否简单地处理 RcvBuffer 内存中已经存在的任何数据 (simulateMode = 2)。 VSX 创建一个 GUI 切换控件以从 simulateMode = 0 或 1 切换到 simulateMode = 2（名为 RcvDataLoop）。

**Resource.Parameters.simulateMode = 0;**

% Resource.Parameters.simulateMode= 1强制模拟模式，即使存在硬件。

% Resource.Parameters.simulateMode = 2停止序列并连续处理 RcvData。

**5.2（定义传感器特性）**

Trans 结构阵列定义了连接到系统或用于模拟的换能器的特性。

**% 指定 Trans 结构数组。**

设置传感器的名称。 在这种情况下，L11-4v 是已知的传感器类型，传感器规格的其余部分可以使用实用函数“computeTrans”填写。 单位可以指定为“波长”或“毫米”。 我们还可以为高压电源设置安全限制。

**Trans.name = 'L11-4v';**

**Trans.units = 'wavelengths';**

**Trans = computeTrans(Trans);**

**Trans.maxHighVoltage = 50;**

通过在 Matlab 命令提示符下键入“Trans”运行 SetUpL11\_4vFlash 脚本后，可以查看实用函数计算的其他 Trans 属性。

为了更好地理解，下面描述了一些添加的属性。

**Trans.frequency = 6.25;**

该属性是换能器的中心频率 Fc。 该值用于确定采集的 A/D 采样率，通常将其设置为最接近的可实现的四倍 Fc 率。

Fc 还指定波长到时间和/或距离的转换（给定声速）。 由于时间和距离属性以波长指定，因此提供适当的中心频率很重要。 [注意：指定波长的时间和距离可以让一切在中心频率或声速发生变化时适当地缩放。 例如，以波长指定采集深度允许深度按比例放大或缩小，同时改变不同的扫描头频率，匹配穿透，与频率成反比。] 重要：如果您希望使用与默认值不同的 Fc 由 computeTrans 指定，请在调用 computeTrans 之前指定它，以便该函数可以使用您的 Fc 值计算波长。

**Trans.type = 0;**

换能器类型 0 指定线性阵列。 支持的其他类型是 1（曲线线性）和 2（二维数组）。 线性数组的元件可以用单个 x 坐标值指定。

**Trans.numelements = 128;**

换能器中的元件数量。

**Trans.elementWidth = 1.0958;**

元件宽度。 请注意，这可能与下面的元件间距不同

**Trans.spacingMm = .300; ％ 毫米**

**Trans.spacing = 1.2175; % 波长**

元件间距有时称为间距。 元件间距由 computeTrans.m 以毫米和波长提供。

**Trans.ElementPos = zeros(Trans.numelements,4);**

**Trans.ElementPos(:,1) = Trans.spacing \* …**

**(-((Trans.numelements-1)/2):((Trans.numelements-1)/2));**

阵列中所有换能器元件的位置。 该数组的每个元件都有一行，具有四个值：x、y、z 和 alpha。 角度 alpha 以弧度为单位，指定元件的法线与 z 轴的角度。 此参数通常仅用于弯曲线性阵列。 x坐标值是我们需要为线性数组设置的唯一参数，它设置为元件中心的位置。 z 轴在中心元件之间穿过换能器阵列的中心。

**Theta = (-pi/2:pi/100:pi/2);**

**Theta(51) = 0.0000001;**

**eleWidthWl = Trans.elementWidth \* Trans.frequency/speedOfSound;**

**Trans.ElementSens = abs(cos(Theta) \* sin(eleWidthWl\*pi\*sin(Theta))./ …**

**(eleWidthWl\*pi\*sin(Theta)));**

元件灵敏度曲线，有时称为方向图，是使用上述公式计算的。 该曲线表示元件对以与元件法线成θ角进入的回波的响应强度下降（以及发射强度随角度的下降）。 在这种情况下，我们使用一个公式来计算窄条类型元件的曲线。 如果我们知道确切的曲线，就像可以用水箱中的水听器测量的那样，我们就可以直接输入这些值。

**Trans.impedance = [53x2 double];**

对于某些换能器，元件阻抗已由 computeTrans 测量和提供。 此属性用于计算功率限制并在模拟模式下模拟发射波形。

5.3 定义用于重建的像素数据区域。

像素数据结构数组 PData 定义了要重建和可能显示的换能器视场区域。 整个 PData 像素阵列的形状始终为矩形，其大小通常适合包围线性阵列换能器的整个扫描区域。 但是，可以调整此矩形的大小并将其放置在相对于换能器的任何位置，并且可以设置为在视野内定义特定的感兴趣区域。 PData 阵列的像素密度独立于 DisplayWindow 像素密度定义，应设置为充分采样系统的图像分辨率。 该分辨率由多种因素决定，例如换能器频率、换能器孔径大小和发射脉冲串的长度。 通常，0.4 到 1.0 波长之间的像素密度足以表示大多数图像。 在指定像素的位置时，应将像素视为无穷小点。

**% 指定 PData 结构数组。**

将 PDelta(pdeltaX,0,pdeltaZ) 属性设置为每个像素的波长增量，该增量足够小以充分采样图像的分辨率。 我们对 L11-4v 线性阵列使用单独的 pdeltaX 和 pdeltaZ 值，因为深度分辨率通常比横向分辨率好得多。 请注意，值越小，给定区域的图像或信号重建所需的时间就越长。

**PData.PDelta = [Trans.spacing, 0, 0.5];**

通过指定行数设置 PData 区域的高度。

**PData.Size(1) = ceil((P.endDepth-P.startDepth) ...**

**/PData.PDelta(3));**

通过指定列数设置 PData 区域的宽度。 在这种情况下，我们希望宽度等于换能器的宽度。

**PData.Size(2) = ceil((Trans.numelements \* ...**

**Trans.spacing)/PData.PDelta(1));**

对于 3D 扫描，PData 数组将是 3 维的。 在这种情况下，它只是 2D，因此我们将 Size(3) 参数设置为 1（单页）。

**PData.Size(3) = 1;**

PData.Origin 是区域左上角的位置。 在这种情况下，我们将原点的 x 维度设置为第一个换能器元件的中心。 z 维度设置为扫描的起点，如 P.startDepth 所给。

PData.Origin = [-Trans.spacing\*(Trans.numelements-1)/2,0, ...

P.startDepth];

上面的 PData 区域定义没有 Region 属性。 当由 VSX 加载时，将调用“computeRegions”实用函数，这将创建一个区域名称“PData”，其大小与 PData 数组相同。

**5.4定义在模拟模式下使用的媒介模型（a media model）**

大多数脚本应该定义一个媒介模型，用于在使用 VDAS 硬件运行脚本之前以模拟模式验证脚本。 媒介模型定义了一组点目标，每个点目标都有自己的反射率。 还可以提供一个函数，该函数可以在执行以某种方式操纵点目标的脚本期间调用。 L11\_4vFlash 脚本中使用了以下属性：

**% 指定媒介对象**

在这个脚本中，我们调用一个单独的 Matlab 脚本来定义媒介模型，命名为“pt1”。 此脚本定义模型名称、点目标的位置及其反射率以及点目标的数量。

**pt1;**

指定一个函数，在每一帧之后移动 x 方向上的所有点。 何时调用此函数在下面定义的接收结构中指定。

**Media.function = 'movePoints';**

**解释（仅在模拟中应用）：**这段代码中，我们调用Matlab 脚本'pt1'来定义介质模型，会定义介质模型的名称、点目标的位置以及它们的反射率等信息。

**Media.function = 'movePoints';**作用是指定一个函数名称 'movePoints'，这个函数用于在每一帧之后将所有的点目标在 x 方向上移动。

**5.5 Define resources used by the sequence（定义序列使用的资源）**

Resource 对象定义了系统所需的内存资源以及一些系统参数。 有关各种缓冲区类型及其组织方式的说明，请参见第 2.5 节。

首先，我们定义一个 RcvBuffer 来存储采集的、单独的通道 RF 数据：

将缓冲区的数据类型设置为 16 位有符号整数。 这是目前唯一的选择。

**Resource.RcvBuffer(1).datatype = 'int16';**

将 rowsPerFrame 属性设置为大于帧中所有通道采集总和的值，因为采集将向下堆叠一列。 在这种情况下，每帧只有一次采集。 考虑到最大深度采集通常少于 512 个波长（1024 次往返），或每次采集 4096 个样本（每个波长 4 个样本），我们可以将最大 rowsPerFrame 设置为 4096 个样本。

**Resource.RcvBuffer(1).rowsPerFrame = 4096;**

将 colsPerFrame 属性设置为可用通道数，在本例中为 128。

**Resource.RcvBuffer(1).colsPerFrame=Resource.Parameters.numRcvChannels;**

将 numFrames 属性设置为要在 RcvBuffer 中捕获的帧数。 帧总数应大于在帧的处理时间内可以获取的帧数。 这样，我们就不会覆盖正在处理的帧中的数据。 例如，如果采集设置为每秒采集 100 帧，而我们每秒能够处理 50 帧，则采集将采集两帧，而我们正在处理一帧。 在这种情况下，我们的 RcvBuffer 中应该至少有两个帧（最好是 3 个或更多帧，因为 CPU 调度的可变性可能会导致某些处理时间更长）。 对于这个脚本，我们设置了大量的帧 (100) 以提供 1 秒的 RF cineloop（以 100 fps 的采集速率）。

**Resource.RcvBuffer(1).numFrames = 100;**

如果我们不使用合成孔径重建，则不需要 InterBuffer 来存储复杂信号像素重建数据。 我们可以直接重建 ImageBuffer 中的强度数据。 10 帧 ImageBuffer 将接收图像重建的输出

，但不会进行显示所需的任何额外处理。 如果我们还想访问 IQ 重建数据（I-同相，Q-正交）的最后一帧，我们可以定义一个单帧 InterBuffer。 如果我们不定义行和列，InterBuffer 和 ImageBuffer 的大小将与 PData.Size 相同。

**Resource.InterBuffer(1).numFrames = 1;**

**Resource.ImageBuffer(1).numFrames = 10;**

下一个要定义的资源是用于显示处理过的回波强度图像的 DisplayWindow。 该缓冲区将定义用于显示图像的 Matlab 显示窗口的属性。 DisplayWindow 通常具有比 ImageBuffer 更高的像素密度，因为 ImageBuffer 必须仅对超声图像的分辨率进行充分采样，而 DisplayWindow 必须包含足够的像素才能在高清显示器上呈现合理大小的图像。 ImageBuffer 的像素将被插值到 DisplayWindow 的像素密度。

DisplayWindow 被创建为具有 8 位像素的位图窗口。 定义颜色图以设置回波强度到灰度值的映射。

设置将出现在显示窗口框架中的标题。

**Resource.DisplayWindow(1).Title = 'L11-4vFlash';**

设置显示窗口的像素密度。 这将决定 DisplayWindow 中图像数据的比例。 pdelta 值越大，在 DisplayWindow 中呈现的图像就越小。

**Resource.DisplayWindow(1).pdelta = 0.35;**

设置显示窗口在计算机显示屏上的位置。 为了使 displayWindow 垂直居中，我们获得了 ScreenSize 数组，它提供了计算机显示器的宽度和高度。 位置参数是[leftEdge, bottomEdge, width, height]。 在这种情况下，我们设置 DisplayWindow 的大小（当转换为波长时）以匹配 PData 区域的大小（以波长为单位）。 我们还设置 DisplayWindow x 和 y 参考点以匹配 PData.Origin x 和 z 参考点。 如果我们想要在显示器上的 PData 区域周围有一个黑色边框，我们可以将 DisplayWindow 大小设置得更大一些，并将参考点设置在 PData.Origin 的略上方和左侧。 （请注意，参考点是以波长为单位的显示窗口左上角相对于换能器坐标系的位置。）

**ScrnSize = get(0,'ScreenSize');**

**DwWidth = ceil(PData.Size(2)\*PData.PDelta(1)/ ...**

**Resource.DisplayWindow(1).pdelta);**

**DwHeight = ceil(PData.Size(1)\*PData.PDelta(3)/ ...**

**Resource.DisplayWindow(1).pdelta);**

**Resource.DisplayWindow(1).Position = [250, ...** % 左边缘

**(ScrnSize(4)-(DwHeight+150))/2, ...** % 底部

**DwWidth, DwHeight];**

**Resource.DisplayWindow(1).ReferencePt = [PData.Origin(1), ...**

**0,PData.Origin(3)];** % 二维成像在 X、Z 平面

设置 DisplayWindow 历史缓冲区的帧数。 历史缓冲区是显示帧的历史。

**Resource.DisplayWindow(1).numFrames = 20;**

设置 DisplayWindow 轴的单位类型。 默认值为“波长”，但我们可以选择“mm”表示毫米。

**Resource.DisplayWindow(1).AxesUnits = ‘mm’;**

设置 DisplayWindow 的颜色图。 这里的属性是一个返回 256x3 double 值数组的函数，每一行都有一个 R、G、B 设置。

**Resource.DisplayWindow(1).Colormap = gray(256);**

有时，您可能会看到一些示例脚本中使用了其他资源属性。 这些通常用于特殊目的或调试。

这就完成了我们序列的非事件结构的定义，即那些不直接参与指定获取和处理动作的结构。 其余要定义的结构将在我们的事件序列中通过它们的索引号直接或间接引用。 显然，对于哪些事件将组成序列来定义引用的结构，用户必须已经非常清楚，因此在某些情况下，用户可能希望在尝试定义引用的项目之前粗略地列出一系列事件。 对于 L11\_4vFlash 等相当简单的序列，很容易将事件序列的外观牢记在心，从而允许预先定义引用的项目。

**5.6 Define the Transmit Waveform structure, TW.（定义发送波形结构）。**

TW 结构定义了发射脉冲的特性。 每个 TW 结构都定义了一个发射波形，因此如果我们有多个不同的波形，我们必须为每个波形创建一个单独的 TW 结构。 然而，对于许多扫描，尤其是 L11\_4vFlash 脚本定义的 2D 回波扫描，定义用于每个发射事件的单个波形就足够了。

设置要使用的发射波形定义类型。 可以由 Vantage 硬件生成的简单发射波形可以将类型定义为“parametric”。 其他更复杂的发射波形类型是“包络”和“脉冲代码”。 如果只编写用于模拟的脚本，则可以使用名为“function”和“sampled”的其他类型。 （见第 3.2.1 节）

**TW(1).type = 'parametric';**

设置参数波形的参数。 这些值表示： A，以 MHz 为单位的发射频率（它应该有一个可以用偶数个 250MHz 时钟周期实现的周期【频率设置为尽可能接近我们的传感器中心频率（40\* (1/250 MHz)=1/6.25MHz】）； B，半周期时间的占空比，表示为1.0的分数（用于发射变迹）； C，发射脉冲串的半周期数； D、前半周期的极性。 在这种情况下，我们希望将频率设置为尽可能接近我们的传感器中心频率6.25 MHz，该频率恰好具有恰好 40 个250MHz 时钟周期的周期。 我们将半周期时间设置为半周期周期的一部分，以更好地近似正弦波，在本例中为 0.67。 对于这个发射波形，我们想要一个1个周期的短脉冲串，所以我们将半周期的周期数设为2。最后，我们将第一个半周期的初始极性设置为1，即正极。 在这种情况下，我们希望所有发射器都使用相同的波形，因此我们只需指定一组参数。

**TW(1).Parameters = [Trans.frequency,0.67,2,1]; ％ A B C D**

**5.7 Define the Transmit events structure, TX.（定义发射事件结构，TX）**

TX 结构定义了扫描期间使用的每个发送事件的特性。 这包括发射波形、焦点（如果有）、转向角和切趾函数。 对于 L11\_4vFlash 扫描，我们只需要一个发射规范，我们将使用它来生成平坦的发射波前。

**% 指定 TX 结构数组。**

指定用于此发射的发射波形。

**TX(1).waveform = 1; %使用第一个 TW 结构。**

设置发射波束的原点。 在平面波前发射的情况下，这不是特别有意义，我们只是将原点设置为孔径的中心。 对于聚焦波束，原点参数将是波束出现在换能器上的点。

**TX(1).Origin = [0.0,0.0,0.0]; % flash发射原点在 (0,0,0)**

如果要使用效用**函数“computeTXDelays”来计算发射延迟**，则需要接下来的两个属性“focus”和“Steer”。

指定发射焦点。 对于未聚焦的光束，该值设置为 0。

**TX(1).focus = 0;**

指定用于此发射的波束控制。 可以用两个角度指定转向：θ 指定光束从正 z 轴（方位角）投影到 x、z 平面的角度，α 指定光束相对于 x、z 平面的角度（仰角） ).

**TX(1).Steer = [0.0,0.0]; % theta, alpha = 0.**

指定用于此发射的发射切趾函数。 对于使用 Vantage 硬件的发射事件，TX.Apod 值的范围可以从 0 到 1，其中 0 表示关闭发射器。 此 TX 引用的 TW 波形结构中 TW.Parameters 的 B 参数将被 TX.Apod 值修改（半周期的占空比分数将乘以每个发射器的 TX.Apod 值）。

**TX(1).Apod = ones(1,Trans.numelements);**

在此示例中，我们选择使用实用函数“computeTXDelays”来计算每个发射器的发射延迟时间。 这些延迟时间是根据 Trans.frequency 属性的波长计算的，它根据 Resource.Parameters.speedOfSound 属性中定义的声速转换为时间。

**TX(1).Delay = computeTXDelays(TX(1));**

**5.8定义时间增益控制（Time Gain Control, TGC.）波形结构**

TGC 结构定义了与 VDAS 硬件一起使用的初始时间增益控制曲线。 可以定义多条曲线，但通常一条曲线就足够了，供所有通道使用。 曲线由八个控制点定义，代表 GUI 面板上 TGC 滑块控件的位置。 曲线始终从 0 开始，值范围从 0 到 1023（最大增益）。

**% 指定 TGC 波形结构。**

定义控制点的初始值。

TGC.CntrlPts = [0,200,344,452,606,747,870,920];

设置 TGC 曲线缩放的最大范围。 需要缩放以使曲线适应不同的深度设置。

TGC.rangeMax = P.endDepth;

使用实用函数“computeTGCWaveform”计算曲线。

TGC.Waveform = computeTGCWaveform(TGC);

**5.9 Define the receive events structure, Receive.（定义接收事件结构体）**

Receive 结构定义了每个采集事件的接收者特征。对于到达 RcvBuffer 中唯一位置的序列中的每个采集事件，应该有一个单独的 Receive 规范。 在 L11\_4vFlash 扫描中，每帧有一个发射/接收事件。 由于可能存在大量接收结构，因此最好使用默认属性预定义结构，然后在循环中用所需的属性覆盖默认属性。 默认属性集的定义如下：

**maxAcqLength = ceil(sqrt(P.endDepth^2 + ...**

**((Trans.numelements-1)\*Trans.spacing)^2));**

**Receive=repmat(struct('Apod', ones(1,Trans.numelements), ...**

**'startDepth', P.startDepth, ...**

**'endDepth', maxAcqLength, ...**

**'TGC', 1, ...**

**'bufnum', 1, ...**

**'framenum', 1, ...**

**'acqNum', 1, ...**

**'sampleMode', 'NS200BW', ...**

**'mode', 0, ...**

**'callMediaFunc', 1), ...**

**1,Resource.RcvBuffer(1).numFrames);**

**Receive.Apod 属**性设置单独的接收通道增益。 值为 0 将关闭接收通道，将所有零写入通道内存。 此处为所有通道设置了 1 的值，这意味着我们希望通道输出具有全振幅。 Receive.Apod 数组中值的数量通常等于非多路复用传感器数组中的元件数量。

**Receive.startDepth** 属性确定 A/D 采样开始的范围，并设置为由 P.startDepth 设置的相同波长值。

**Receive.endDepth** 属性确定 A/D 采样结束的范围。 对于此属性，我们需要设置比 P.endDepth 中找到的值更大的值，因为要在 P.endDepth 处重建像素，我们需要为不在重建像素正上方的元件的较长路径长度获取 RF 数据。 可以将 L11-4v 的最坏情况最长路径长度计算为 P.endDepth 和换能器孔径的平方和的平方根。 对于 192 个波长的 P.endDepth 值和 128 个波长的孔径大小，最坏情况下的路径长度约为 230 个波长（maxPathLength）。

Receive.TGC 属性是 TGC 结构中定义的时间增益控制波形的编号（参见前面定义的对象）。 该曲线定义了接收器增益如何随着扫描深度的时间而增加。

接下来的三个属性**“bufnum”、“framenum”**和**“acqnum”**定义了 Receive 的采集数据在主机内存 RcvBuffer 中的位置。 回想一下，RcvBuffer（在这种情况下，RcvBuffer 第一个）是用多个帧定义的。 在每一帧中，每一列都包含相应通道的所有采集数据，在单独的 acqnum 段中，这些段连续向下打包。 请注意，采集数据首先存储在 VDAS 模块的本地内存中，因此这些属性指定采集数据发射到主机内存时的目标位置。

**Receive.sampleMode** 属性设置为‘NS200BW，这将转化为 RcvBuffer 的采样率是 Trans.frequency 的四倍（设置为基于 250MHz 主时钟的最接近的可实现采样率）。

Receive.mode 属性指定接收通道如何存储其 RF 数据。 对于 Receive.mode = 0，RF 数据替换本地内存中的数据。 对于 Receive.mode = 1，获取的数据将添加到本地内存中已有的数据中。 此模式允许在将累加和发射到 RcvBuffer 之前在本地内存中累加采集次数（有关用法，请参阅编码示例部分）。

我们的规范中缺少 Receive.LowPassCoef 和 Receive.InputFilter 属性，这意味着我们将使用系统提供的默认值。 使用 VSX 运行我们的脚本后，可以在 Matlab 工作区中检查这些属性。 可以使用以下 Matlab 命令绘制 Receive(1) 的滤波器响应：

**>>freqz([Receive(1).InputFilter,Receive(1).InputFilter(20:-1:1)])**

**>>freqz([Receive(1).LowPassCoef,Receive(1).LowPassCoef(11:-1:1)])**

最后一个属性**“callMediaFunction”**仅在模拟模式下使用。 当为 Receive 结构设置为 1 时，在获取模拟数据之前调用 Media 结构中提供的 Media.function 名称。 这允许在帧之间移动媒介点，以便模拟可以处理移动目标。

**以下循环设置各个Receive（接收）结构的属性：**

**% - 设置事件特定的接收属性。**

for i =1:Resource.RcvBuffer(1).numFrames

Receive(i).framenum = i;

End

在这个循环中，它修改了与每个采集帧关联的接收结构，正在设置的参数是： Receive(i).framenum – 要在 RcvBuffer 中用于获取数据的帧数。

**5.10 Define the reconstruction structure, Recon.（定义重建结构）**

Recon 结构定义了图像/信号重建过程的细节。 这是计算 PData.Region 结构中定义的像素位置处的复数信号或幅度值的过程。 Recon 结构的输入数据始终是 RcvBuffer 中的 RF 数据，输出数据被定向到复杂信号数据的 InterBuffer 和/或幅度数据的 ImageBuffer。 重建可以有多个部分； 例如，在合成孔径重建中，它结合了来自单独采集的复杂信号数据。 重建的多个部分使用 ReconInfo 结构定义，这将在下面描述。

**% 指定 Recon 结构数组。**

**Receive=struct('senscutoff', 0.6, ...**

单个 Recon 结构在结构定义中定义。 第一个属性，“senscutoff”参数表示一个从 0 到 1.0 的阈值，它确定单个元件何时对重建有贡献。 Trans.ElementSens 函数用于确定元件对正在重建的像素点的相对灵敏度——如果灵敏度高于“senscutoff”值，则元件的信号包含在重建中，否则不包含。

设置此 Recon 将使用的 PData 结构编号。 该结构定义了将被重建的像素位置。

**'pdatanum', 1, ...**

**“rcvBufFrame”**是一个可选参数，可以设置为覆盖与此 Recon 关联的 ReconInfo 结构中指定的 RcvBuffer 帧编号。 如果该值为正整数，则直接表示要使用的帧号。 如果设置为 -1，如本例所示，则表示应使用最近发射的采集帧。

**'rcvBufFrame', -1, ...**

**InterBuffer** 帧用于存储来自重建的 IQ 数据。 在这种情况下，实际上不需要 InterBuffer，因为我们每帧只有一次采集，可以直接在 ImageBuffer 中重建为强度。 在“replaceIntensity”重建中包含一个 InterBuffer 定义 [buffer 1, frame 1] 将输出 IQ 数据和强度数据，即使这个脚本没有进一步使用这些数据。

**'IntBufDest', [1,1], ...**

设置 ImageBuffer 帧以用于存储来自重建的幅度数据。 在这种情况下，我们使用第一个缓冲区，并将帧号设置为 -1，这意味着使用缓冲区中的下一个可用帧。 这允许捕获可以在“冻结”模式下检查的输出帧的历史记录。

**'ImgBufDest', [1,-1], ...**

定义要与此 Recon 一起使用的 ReconInfo 结构的索引号。 如果引用多个 ReconInfo 结构，则应将它们定义在行数组或向量中。 在这种情况下，只能指定一个 RINum。

**'RINums', 1);**

**5.11 定义 ReconInfo 结构。**

ReconInfo 结构定义了重建过程中每个步骤的细节。每个 ReconInfo 结构的索引都在 Recon 结构中引用，我们必须为每个引用定义一个 ReconInfo 结构。 多个 Recon 结构不得引用相同的 ReconInfo 结构。 ReconInfo 结构引用用于获取采集数据的发射和接收结构，并指定要重建的 PData 数组区域。 他们还定义了一种重构模式，决定是输出复数信号数据还是幅值数据，以及是替换输出缓冲区中的数据还是与缓冲区中已有的数据累加。

**% 定义单个 ReconInfo 结构。**

**ReconInfo = struct('mode', 'replaceIntensity', ...**

**'txnum', 1, ...**

**'rcvnum', 1, ...**

**'regionnum', 1);**

**ReconInfo.mode** 属性定义了重建的输出以及如何将其处理到输出缓冲区中。 此示例使用“replaceIntensity”模式（有关不同模式的完整定义，请参阅第 3.4.1 节）。 模式“replaceIntensity”输出重建的强度数据，替换输出 ImageBuffer 中的数据。

rcvnum 属性设置为 1，但在 Recon 结构中我们指定了 rcvBufFrame = -1，这意味着 1 将在运行时被替换为最近获取的帧的接收索引。

请注意，此重建的 regionnum 属性设置为 1。我们没有在 PData 结构中定义 PData.Region，但是当我们运行脚本时，VSX 将调用 computeRegions 实用函数，这将添加一个 PData。 包含整个 PData 数组的区域结构。

**5.12 Define the Process structure(s)（定义流程结构）。**

Process 结构定义了对重建像素数据执行的处理类型，并设置了该处理的各种参数。 Process 结构由类名、类中的方法和参数（属性）列表组成。 在 L11\_4vFlash 脚本中，唯一要执行的处理是在 DisplayWindow 中显示图像。

**% 指定流程结构数组。**

将类名属性设置为“图像”。 这是对 ImageBuffer 中的图像数据进行操作的方法类。

**Process(1).classname = 'Image';**

将方法设置为“imageDisplay”。 这是用于处理 ImageBuffer 中的图像并将结果发送到 DisplayWindow 的方法。

**Process(1).method = 'imageDisplay';**

用于处理的参数列表。 第一个是要使用的 ImageBuffer。

**Process(1).Parameters = {'imgbufnum',1,...**

设置要处理的帧数。 在这种情况下，-1 表示处理最后重建的帧。

**'framenum',-1,...** % (-1 => lastFrame)

指定描述帧像素位置的 PData 结构索引。

**'pdatanum',1,...** % 要使用的 PData 结构

“pgain”值将增益因子应用于重建输出。 值 1.0 会使数据保持不变。

**'pgain',1.0,...** % 处理增益

在将强度映射到显示器之前，“拒绝”值会切断低强度。 该值是要拒绝的强度范围的下四分位数的百分比。

**'reject',2,...** % 处理增益

**Reject****解释：**假设你有一个图像，其中包含许多像素，每个像素具有不同的强度值。如果你将所有像素的强度值按升序排列，强度值的分布将形成一个从低到高的曲线。而这个分布的四分之一点，就是从低强度到高强度的前 25% 部分。

现在，假设你设置 "reject" 参数的值为 2。这意味着系统会从强度分布的低端截取掉前 2% 的部分。换句话说，它会剔除掉最低的 2% 的强度值，然后将剩余的 98% 的强度值映射到显示范围内。

这样的处理可以在显示图像时帮助过滤掉一些强度非常低的噪声或无关紧要的部分，使得图像在显示时更加清晰和易于观察。参数 "reject" 的作用就是控制这种截取的程度，以便根据应用需求来调整图像的显示效果。

将持久性方法设置为“simple”，并将级别设置为 20，这意味着强度值输出由前一帧强度的 20% 加上新强度值的 80% 组成。

**'persistMethod','simple',...**

**'persistLevel',20,...**

设置插值方法。 在这种情况下，“4pt”表示使用 4 点双线性插值法。 这是目前唯一支持的方法。

**'interpMethod','4pt',...** % interp. method

设置“grainRemoval”、“processMethod”和“averageMethod”。 前两个属性用于将 3x3 空间过滤器应用于图像数据，而最后一个属性用于在显示之前平均 2 或 3 个连续帧。 在这种情况下，“无”表示未应用任何处理。

**'grainRemoval','none',...**

**'processMethod','none',...**

**'averageMethod','none',...**

设置“compressMethod”和“compressFactor”。 “compressMethod”选项是“power”或“log”。 “power”选项将强度数据提高到 n 次幂，其中 n 是由“compressFactor”值设置的分数（40 等于 0.5）。 可以通过修改显示窗口颜色图来完成额外的压缩或扩展。

**图像压缩解释：**有些图像的强度范围可能非常广，从非常低的强度到非常高的强度。如果直接将这种广泛的强度范围映射到显示设备的范围内，可能会导致一些低强度细节在显示中难以分辨，因为显示器只能同时显示有限范围内的强度级别。通过压缩图像的强度范围，可以将细节集中在显示范围内，从而更好地呈现图像的特定区域。

压缩可以增强图像中的对比度，使细微的强度变化更加明显。通过将较低的强度值提升，并将较高的强度值降低，可以突出图像中的特定特征和细节，使其更加易于观察。

**'compressMethod','power',... %power表示使用幂函数压缩**

**'compressFactor',40,...**  % X^0.5，开根号

设置“mappingMode”以在灰度/彩色图的方式显示强度数据。彩色流图像将彩色图分为灰色和彩色两半。

**'mappingMode','full',... %**这里设置为 'full'，表示希望将图像的强度数据在灰度/彩色映射的全部范围内显示。对于彩色流动图像，这将分成灰度和彩色两个部分。

将“display”属性设置为 1 ，表示要在处理后立即显示图像。 在某些情况下，可能需要进行其他处理之后才显示图像，所以可以将这个值设置为其他数值或者 0。

**'display',1,...**  % 显示图像

设置用于指定图像应该显示在哪个显示窗口中。

**'displayWindow',1}; %**设置为 1，表示希望图像显示在第一个显示窗口中。

**5.13定义 SeqControl 和 Event 结构。**

我们现在已经准备好为我们的序列定义单独的序列事件。 每个事件都可以引用发射动作、接收动作、重建动作和处理动作。 最后，可以设置一个 SeqControl 动作来控制诸如 DMA 发射和事件流控制等因素。 对于不需要的操作，设置零索引，表示该项目没有操作。 在 tx 和 rcv 操作的情况下，这些必须都设置或仅设置 tx 索引（对于仅发射事件）。 仅接收事件可以具有对 TX 结构的 tx 引用，其中禁用所有发射器 (TX.Apod = zeros(128))。

在事件序列中需要它们的地方定义 SeqControl 结构通常更容易。 这种方法的例外是重复使用的 SeqControl 结构，在这种情况下，预先定义它们会更简单。 在 SetUpL11\_4vFlash 脚本中，我们最初定义了四个 SeqControl 结构。

**% 指定已知的 SeqControl 结构。**

**SeqControl(1).command = 'jump';** % 跳回开始。

**SeqControl(1).argument = 1;**

**SeqControl(2).command = 'timeToNextAcq';** % 帧间时间

**SeqControl(2).argument = 10000**; % 10 毫秒 100fps

**SeqControl(3).command = 'returnToMatlab';**

定义一个计数器变量 nsc，以跟踪新的 SeqControl 索引。

**nsc = 4;**  % nsc 是 SeqControl 对象的计数

定义一个计数器变量 n，以跟踪新的事件索引。 如果需要，这使得在序列中插入新事件变得容易。

**n = 1;** % n 是事件的计数

**% 获取 RcvBuffer 中定义的所有帧**

RcvBuffer 中所有帧的获取和重建定义在一个循环中。 这使得只需更改 Resource.RcvBuffer.numFrames 属性即可轻松更改 RF cineloop 中获取的帧数。

**for i = 1:Resource.RcvBuffer(1).numFrames**

**Event(n).info = 'acquisition';**

**Event(n).tx = 1; % use 1st TX structure.**

**Event(n).rcv = i; % use ith Rcv structure.**

**Event(n).recon = 0; % no reconstruction.**

**Event(n).process = 0; % no processing**

**Event(n).seqControl = [2,nsc]; % use seqControl nsc defined below**

**SeqControl(nsc).command = 'transferToHost';**

**nsc = nsc + 1;**

**n = n+1;**

采集每一帧后，设置“timeToNextAcq”SeqControl 以控制 VDAS 硬件的帧速率，并请求将采集的帧数据发射到主机内存 RcvBuffer。 对于发射到 RcvBuffer 中唯一位置的每个帧，我们需要一个单独的“transferToHost”命令，因为这些命令将被转换为唯一的“DMA”命令。

在每个获取的帧之后指定重建。 硬件定序器忽略这些事件并在软件进行重建处理时继续获取新帧。 软件处理忽略 tx 和 rcv 操作（除非在模拟模式下），并且 Recon.rcvBufFrame = -1 时，将尝试处理最近发射的数据帧。 如果切换 RcvDataLoop 控件，重建将从上一帧开始处理 RcvBuffer 中的下一帧，并将按顺序处理缓冲区中的每一帧。

**Event(n).info = 'Reconstruct';**

**Event(n).tx = 0; % no transmit**

**Event(n).rcv = 0; % no rcv**

**Event(n).recon = 1; % reconstruction**

**Event(n).process = 1; % processing**

由于 RF cineloop 可能很长，导致在到达序列末尾之前重建了很多帧，我们在每重建的第 5 帧处插入一个“returnToMatlab”。 这允许更快速地响应 GUI 控件，这只能在回到 Matlab 环境时发生。

**if floor(i/5) == i/5 % 每 5 帧退出到 Matlab**

**Event(n).seqControl = 3; % return to Matlab**

**else**

**Event(n).seqControl = 0;**

**end**

**n = n+1;**

**end**

最后一个事件只是跳回第一个事件。

**Event(n).info = 'Jump back to first event';**

**Event(n).tx = 0; % no TX**

**Event(n).rcv = 0; % no Rcv**

**Event(n).recon = 0; % no Recon**

**Event(n).process = 0;**

**Event(n).seqControl = 1; % jump command**

上述事件定义了一个异步序列，这意味着硬件序列器独立于软件序列器运行，没有任何同步。 要理解这是如何工作的，记住系统中有两个运行的定序器很有用，一个在 VDAS 模块上以硬件实现，另一个在 runAcq.c mex 程序的软件中实现。 这些定序器独立运行，并且只响应它们可以实现的事件操作。 硬件定序器操作的唯一事件操作是 tx、rcv 和一些 seqControl 操作。 硬件定序器忽略侦察和处理操作。 或者，软件定序器操作的唯一事件操作（当与 Vantage 硬件一起运行时）是侦察、处理和一些 seqControl 操作。

从硬件序列器的角度来看，我们的事件序列表示“获取 Resource.RcvBuffer(1).numFrames 帧，帧之间的时间为 10 毫秒，每帧后将采集数据发射到主机内存； 然后跳回到序列的开始，然后再做一次。” 从软件定序器的角度来看，我们的事件序列说“进行重建，使用最近发射到 RcvBuffer 的帧数据（如 Recon.rcvBufFrame = -1 所示），然后处理重建数据并将其发送到 展示。 在 5 个重建/处理帧之后，返回到 Matlab。” 在 Matlab 环境中不需要任何操作的情况下，在“returnT oMatlab”命令之后，Event 序列将重新进入 Event，软件序列器将再执行 5 个重建/处理操作。 在 Resource.RcvBuffer(1).numFrames/5 重建/处理设置后，将跳回到序列的开头，并重复处理。

只要硬件帧采集快于重建处理时间，每次重建后都会显示一个新的图像帧，并且帧速率会显得相当均匀（如果 显示的帧速率高于约 25 fps。）在本例中，我们将采集帧速率设置为大约 100 fps（在采集帧时使用“timeToNextAcq”命令）。 如果我们对单个闪光图像的重建/处理速率约为 40 fps（这将取决于计算机处理能力，但通常要高得多，超过 100 fps），我们可以每 2.5 个采集帧显示一个新图像。 由于我们只能处理和显示完整的帧，因此我们显示的帧将不均匀地通过 RcvBuffer，可能采用以下方式：1, 3, 6, 8, 11, 13, 16, 18, 21, 23, 26 ... 肉眼看不到这种不均匀的步进，因为肉眼看到的是统一的帧速率。

在 RcvDataLoop 模式下（通过按下 GUI 面板上的切换按钮进入），重建/处理操作将依次遍历所有采集的帧，采集的数据将以慢动作播放。 在这种模式下，我们正在重建并查看所有 100 fps 的采集，并且帧之间的时间是统一的。 这是应测量媒介中任何运动的模式，因为帧之间的时间由“timeToNextAcq”值精确设置。

如果想要同步序列操作，其中硬件在获取额外帧之前等待处理完成，则上述事件序列只需要进行一些小的更改。 添加的语句在下面修改后的事件序列中以红色显示（这些更改来自“SetUpL11\_4vFlashSync.m”文件）。

**% 指定 SeqControl 结构数组。**

我们可以消除之前的帧速率的 SeqControl(2) ‘timeToNextAcq’，因为帧速率现在将由处理速率决定。

**SeqControl(1).command = 'returnToMatlab';**

**SeqControl(2).command = 'jump';**  % 跳回开始。

**SeqControl(2).argument = 1;**

**nsc = 3;**  % nsc 是 SeqControl 对象的计数

创建一个新变量来保存最后一个“transferToHost”SeqControl 的索引。

**lastTTHnsc = 0;**

n = 1; % n 是事件的计数

**% 获取 RcvBuffer 中定义的所有帧**

**for i = 1:Resource.RcvBuffer(1).numFrames**

**Event(n).info = 'Acquisition';**

**Event(n).tx = 1;**  % 使用第一个 TX 结构。

**Event(n).rcv = i;**  % 使用第一个 Rcv结构。

**Event(n).recon = 0;**  % 没有重建

**Event(n).process = 0;**

将以下 seqControl 规范替换为下方的红色规范。

**Event(n).seqControl = [1,nsc];** % SeqControl structs

**SeqControl(nsc).command = 'transferToHost';**

**nsc = nsc + 1;**

将“transferToHost”命令的 SeqControl.condition 设置为“waitForProcessing”，并将 SeqControl.argument 设置为指向之前的“transferToHost”SeqControl。 这表明我们希望在执行下一个“transferToHost”SeqControl 命令之前完成对由前一个“transferToHost”SeqControl 发射的数据的处理。 不再需要“timeToNextAcq”命令，因为硬件定序器将与处理速率同步。

**Event(n).seqControl = nsc;**

**SeqControl(nsc).command = 'transferToHost';**

**SeqControl(nsc).condition = 'waitForProcessing';**

**SeqControl(nsc).argument = lastTTHnsc;**

**lastTTHnsc = nsc;**

**nsc = nsc + 1;**

**n = n+1;**

侦察事件可以保持不变，因为它会在处理之前自动等待“transferToHost”命令完成，之后它将执行“markTransferProcessed”以释放硬件定序器以执行下一个“transferToHost”。

**Event(n).info = 'Reconstruct';**

**Event(n).tx = 0; % no transmit**

**Event(n).rcv = 0; % no rcv**

**Event(n).recon = 1; % reconstruction**

**Event(n).process = 1; % processing**

**n = n+1;**

修复第一个同步 SeqControl 结构中的参数以指向最后一个 transferToHost SeqControl。 序列第一次执行时，它不会等待最后一次发射的处理，因为尚未发射任何数据。

SeqControl(3).argument = lastTTHnsc;

Event(n).info = 'Jump back to first event';

Event(n).tx = 0; % no TX

Event(n).rcv = 0; % no Rcv

Event(n).recon = 0; % no Recon

Event(n).process = 0;

Event(n).seqControl = 2; % 跳转命令

通过这些更改，硬件采集帧速率将与重建/处理速率相匹配。 我们可以将 Recon.rcvBufFrame 设置为 -1（处理最近发射的帧），因为随着处理的获取节奏，最近发射的帧将始终是刚刚获取的帧。

最后一点：如果我们的异步 L11\_4vFlash 序列在模拟模式下执行（通过设置 Resource.Parameters.simulateMode = 1），将启动从我们的媒介点集获取的模拟，并且将处理每个获取帧。 实际上，我们的序列将变成同步序列，因为软件现在正在连续处理所有事件操作。

L11\_4vFlash序列的参数说明到此结束。 脚本的其余部分定义了可选的 GUI 控件，这些控件将在加载期间由 VSX 添加到 GUI 面板。 这些控件的创建在第 4.1 节中介绍。

**6.** **Coding Techniques（编码技巧）**

**6.1 Averaging RF Data（平均 RF 数据）**

在发射到主机接收缓冲区之前，采集数据可以累积到 Vantage 模块的本地内存中。 这可用于提取隐藏在噪声中的小信号，或用于谐波成像的求和脉冲反转采集。 当使用累积模式 (Receive.mode = 1) 时，VDAS 模块上的本地内存用于累积采集数据，然后再向主机内存中的 RcvBuffer 进行任何发射。 请注意，Vantage 模块上的本地内存只有 16 位宽，因此在进行大量累加时，可能会出现饱和。 如果对隐藏在噪声中的非常小的信号进行平均，这通常不是问题，因为噪声的累积只会随着采集次数的平方根而增长。 溢出检测在累加器中实现，因此超过 16 位有符号整数范围的累加将被限制在最大正值或负值处。

为了平均两次采集（如用于脉冲反转），只需执行两次采集，第一次使用 Receive.mode = 0，第二次使用 Receive.mode = 1，使用相同的 Receive bufnum、framenum 和 acqnum 规范。 在定义帧的接收结构时，像往常一样定义模式 0 接收，其中 acqNum 从先前的模式 0 增加 1（用于每帧多次采集）。 然后在模式 0 之后立即定义模式 1 Receive，具有匹配的 bufnum、framenum 和 acqNum 属性。 模式 0 和模式 1 接收也可以定义为两个单独的组，如果更方便的话 - 为帧定义所有模式 0 接收，然后使用相同的 bufnum、framenum 和 acqNum 属性定义所有模式 1 接收。

为了对两次以上的采集进行平均，设置一个循环序列很有用，其中可以使用循环计数器设置平均采集数。 下面的 Matlab 代码是部分事件序列的示例，它使用多个流控制 SeqControl 命令来完成此操作。 第一组 Receives 的定义与使用 Receive.mode = 0 的单次采集完全相同，因此这些 Receives 将具有从 1 开始递增的 acqNum 值。VSX 将像往常一样为这些 Receives 分配 Receive.VDASblockNum 值，从 从本地内存中的第一个内存位置开始。 可以在第一组之后立即定义的下一组 Receives 将与第一组相同，但 Receive.mode = 1。第二组的 bufnums、framenums 和 acqNums 应该与 第一集。

您现在可以通过使用两组 Receives 来设置您的 Events 以累积所需数量的采集。 事件列表将有一个循环来进行累积，其数量可以在变量“numAccum”中定义。 如果 numAccum = 1，则不进行累加，第一组采集被传送到主机。

**numAccum = 11;**

**% 指定 SeqControl 结构**

**% - 设置循环计数。**

**SeqControl(1).command = 'loopCnt';**

**SeqControl(1).argument = numAccum-1;**

**% - 跳转到测试循环计数。**

**SeqControl(2).command = 'jump';**

**SeqControl(2).argument = [];**  %-跳回到累积的开始。

**SeqControl(3).command = 'loopTst';**

**SeqControl(3).argument = [];** % - 将数据帧发射到主机。

**SeqControl(4).command = 'transferToHost';**

**% Specify Event Structures % 指定事件结构**

**% Acquire the first set of acquisitions for the frame % 获取帧的第一组采集**

**for j = 1:numAcqs**

**Event(n).tx = j;** % 使用适当的 TX 结构。

**Event(n).rcv = j;**  % 使用第一组接收

**Event(n).recon = 0;**  % 没有重建

**Event(n).process = 0;**  % 0=不处理

**n = n+1;**

**end**

**Event(n).info = 'Set loop count for number of accumulates.';**

**Event(n).tx = 0;**

**Event(n).rcv = 0;**

**Event(n).recon = 0;**

**Event(n).process = 0;**

**Event(n).seqControl = 1;** % command='loopCnt',argument=numAccum-1

**n = n+1;**

**Event(n).info = 'Jump to loop count test.';**

**Event(n).tx = 0;**

**Event(n).rcv = 0;**

**Event(n).recon = 0;**

**Event(n).process = 0;**

**Event(n).seqControl = 2;**  % 'jump-to-test' SeqControl

**n = n+1;**

**SeqControl(3).argument = n;** %Sets the jump event no. for start of accums.

**For j = 1:numAcqs**

**Event(n).tx = j;** % use appropriate（合适的） TX structure.

**Event(n).rcv = j+numAcqs;** % use 2nd set of Receive

**Event(n).recon = 0;**  % no reconstruction

**Event(n).process = 0;**  % 0=no processing

**n = n+1;**

**end**

**SeqControl(2).argument = n;** % Set jmp event for the 'jmp-to-test' SeqControl.

**Event(n).info='Test loop count-if nz,jmp back to start of accumulates.';**累计

**Event(n).tx = 0;**

**Event(n).rcv = 0;**

**Event(n).recon = 0;**

**Event(n).process = 0;**

**Event(n).seqControl = 3;**

**n = n+1;**

% - Transfer data to host - 向主机发射数据

**Event(n).info = 'Transfer data to host.';**

**Event(n).tx = 0;**

**Event(n).rcv = 0;**

**Event(n).recon = 0;**

**Event(n).process = 0;**

**Event(n).seqControl = 4;**

**n = n+1;**

6.2 条件序列事件编码

很多时候，人们希望有条件地执行一个事件或一系列事件，这可能取决于 GUI 按钮按下或其他操作。 有多种编码条件事件的方法，应该选择最适合应用程序的方法。 由于硬件和软件定序器的异步执行，如果选择了不合适的方法，可能会得到意想不到的结果。 事件条件执行的方法如下：

1. **更改 Resource.Parameters.startEvent** - 这是有条件地执行事件的首选方法，这些事件可以按短序列放置在事件列表顶部附近。 请参阅下面所示的事件概要。

**一次性条件执行：**

**Event(1)**

**--**

**--**

**Event(i-1).seqControl -> ‘jump’ to Event j;**

**Event(i)**

**--**

**--**

**Event(j)**

**--**

**--**

**Event(k).seqControl -> ‘jump’ to Event j ；**

上述事件序列有条件地执行从 Event(1) 到 Event(i-2) 或从 Event(i) 到 Event(j) 的事件，具体取决于 startEvent 是否设置为 1 或 i。 在执行一次条件事件后，执行将从事件（j）到事件（k）的事件序列循环。 只要在运行时设置了 Resource.Parameter.startEvent，或者当序列被冻结和解冻时，就会执行条件事件。

**Either-Or（非此即彼）的条件执行：**

**Event(1)**

**--**

**--**

**Event(i-2).seqControl -> ‘call’ events starting at Event k;**

**Event(i-1).seqControl -> ‘jump’ to event Event 1;**

**Event(i)**

**--**

**--**

**Event(j-1).seqControl -> ‘call’ events starting at Event k;**

**Event(j).seqControl -> ‘jump’ to event Event i;**

**Event(k)**

**--**

**--**

**Event(m).seqControl -> ‘rtn’**

上面的序列可以设置为从事件 1 或事件 i 开始执行，只需将 Resource.Parameters.startEvent 设置为 1 或 i。 例如，当设置为start at 1 时，执行从1 到i-3 的事件，然后调用从k 开始的事件的子程序。 子程序返回后，执行跳回到第一个事件并重复序列。 子程序包含两种情况下要执行的事件，而1到i-3的事件仅在startEvent设置为1时执行，i到j的事件仅在startEvent设置为i时执行 .

可能会使用这种编码方法的应用程序是在脚本中更改一系列采集的发射波束的特性。 波束 1 的采集可以在第一个事件之后编码，波束 2 的采集可以在事件 i 之后编码。 重建和处理事件对于这两种情况都是相同的，并且在从事件 k 开始的子程序中进行编码。 这允许在设置脚本中预先计算两个波束的发射波束特性，并在它们之间快速切换。

执行条件事件的 startEvent 方法通常优于其他方法，因为只要 startEvent 发生变化，它就会同步硬件和软件定序器。 更改只能在序列中的“returnT oMatlab”点之后发生，因此可以控制处理发生更改的时间。 在“returnT oMatlab”点，执行任何挂起的 GUI 操作，并且 Resource.Parameters.startEvent 的“设置”会导致硬件和软件定序器在序列恢复时在 startEvent 处启动。

**b. 使用 LoopCnt/LoopTst 命令实现“开关”结构** - Matlab 或 C 中的典型开关结构允许根据控制参数的值执行不同的代码段。 可以在一系列事件中创建类似的结构，前提是在序列中只有一次使用 loopCnt/loopT st 命令。 此方法还需要使用 startEvent 来指定 loopCnt 的不同值。 以下序列大纲实现了一个 3 种情况的开关。

**Event(1).seqControl -> ‘loopCnt’ set to 0**

**Event(2).seqControl -> ‘jump’ to Event 6**

**Event(3).seqControl -> ‘loopCnt’ set to 1**

**Event(4).seqControl -> ‘jump’ to Event 6**

**Event(5).seqControl -> ‘loopCnt’ set to 2**

**Event(6)**

**--**

**--**

**--**

**(common events here)**

**--**

**--**

**--**

****

startEvent 的初始值（1、3 或 5）决定将执行哪个案例，并且该案例将在每次通过序列循环时重复。 在运行时，用户可以使用 GUI 控件更改 startEvent 以选择不同的 loopCnt，这将导致执行不同的 case。 硬件和软件定序器将与 startEvent 值的每次更改同步。

**c. 使用“cBranch”（条件分支）命令** - 虽然此命令似乎是有条件地执行事件的最明显方式，但很难正确使用。 问题是硬件和软件定序器的异步执行。 **当使用函数调用设置分支标志时，硬件和软件定序器将在下一次到达包含“cBranch”顺序控制命令的事件时采用分支**。 这就是问题所在，因为当设置分支标志时，硬件和软件定序器可能处于不同的事件中，硬件定序器可能在 cBranch 事件之前，而软件定序器在它之前。 这可能会导致难以分析的意外行为。 “cBranch”命令的最佳用途是在两个没有同步 SeqControl 命令的独立序列循环之间切换。 这允许硬件定序器或软件定序器首先分支，而不会被等待处理的硬件定序器或等待 DMA 完成的软件定序器挂断序列。

**6.3 Combining RF Data for Multi-focal Zone Transmit/Receive（组合射频数据进行多焦区发射/接收）**

一种常用的线模式扫描采集方法涉及沿每个线方向使用不同的发射焦点区域进行发射，并结合 RF 数据以实现一种“动态”发射聚焦。 这提高了更大景深的横向分辨率，但代价是更长的线采集时间（这对于短深度或非移动目标可能不会出现问题）。 通过使用以下编程步骤，可以在 Verasonics 系统中实施这种采集方法，而不会影响 DMA 发射时间或处理时间：

**1)** 多个发射聚焦区定义有不同的 TX 结构。 例如，对于 3 个发射区域，可以定义焦点，如下所示：

**TX(1) -----------------------------------------------<-----1----->-------------------**

**TX(2) -------------------------------<------2----->----------------------------------**

**TX(3) ----------------<-----3----->---------------------------------------------------**

在上图中，深度从左到右增加，每个焦区的景深显示为左右箭头。 目标是结合 3 个不同焦区的接收采集，以便可以扩展有效景深。

**2)** 定义TX结构对应结构的Receive startDepth和endDepth属性，从最深的深度开始。 所有 Receives 都应在相同的 startDepth 开始获取，并使用刚好延伸到相应 TX 焦点区域末端的 endDepth。

**Receive(1) |----------------------------------------------<-----1----->----------|-----**

**Receive(2) |------------------------------<------2----->|------------------------------**

**Receive(3) |---------------<-----3----->|----------------------------------------------**

^ ^ ^ ^

Receive(1:3). Receive(3). Receive(2). Receive(1).

startDepth endDepth endDepth endDepth

第一个 Receive 的 endDepth 设置为获取到扫描区域的深度。 其他 Receives 设置为仅获取相应焦点区域的末端。

**3)** 定义 Receive.acqNum 值，以便 Receive(2) 的采集覆盖 Receive(1) 的采集，Receive(3) 的采集覆盖 Receive(2) 的采集。

**Receive(1).acqNum = 1;**

**Receive(2).acqNum = 1;**

**Receive(3).acqNum = 1;**

**Receive(4).acqNum = 2;** % Receive for deepest zone of next ray. 接收下一射线的最深区域。

**----**

注意：下一条射线的 Receive(4) 的 acqNum 值设置为 2，并且必须比之前定义的最大 acqNum 大 1，以便为 Receive(4) 分配新的存储空间，而不是覆盖之前的 Receive。

**4)** 定义事件结构以首先获取最深区域的 RF 数据，然后是较浅区域。

**Event(n).info = 'Acquire deepest zone first.';**

**Event(n).tx = 1;**

**Event(n).rcv = 1;**

**Event(n).recon = 0;**

**Event(n).process = 0;**

**Event(n).seqControl = 1;** %在这里使用适当的“timeToNextAcq”

**n = n+1;**

**Event(n).info = 'Acquire zone 2';**

**Event(n).tx = 2;**

**Event(n).rcv = 2;**

**Event(n).recon = 0;**

**Event(n).process = 0;**

**Event(n).seqControl = 1;**

**n = n+1;**

**Event(n).info = 'Acquire zone 3';**

**Event(n).tx = 3;**

**Event(n).rcv = 3;**

**Event(n).recon = 0;**

**Event(n).process = 0;**

**Event(n).seqControl = 1;**

**n = n+1;**

对于 128T/64R VDAS 系统，人们可能想要使用合成孔径采集，这需要对我们的序列编程稍作修改。 在这种情况下，我们将为每个 TX 聚焦区域设置两个 Receives，我们将 Receives 定义如下：

**Receive(1). acqNum = 1;** % **Receive for 1st half of aperture for deepest**

**Zone 1.（**接收最深区域1的第一半孔径**）**

**Receive (2).acqNum = 2;** % 接收最深区域1的第二半孔径

**Receive (3).acqNum = 1;** % 接收区域 2 的第一半孔径。

**Receive (4).acqNum = 2;** % 接收区域 2 的第二半孔径**。**

**Receive (5).acqNum = 1;** % 接收区域 3的第一半孔径。

**Receive (6).acqNum = 2;** % 接收区域 3 的第二半孔径。

**Receive (7).acqNum = 3;** % First Receive for next ray line.下一条射线的首次接收

**------**

事件结构将定义如下：

**vent(n).info = 'Acquire 1st half of aperture for deepest zone, 1.';**

**Event(n).tx = 1;**

**Event(n).rcv = 1;**

**Event(n).recon = 0;**

**Event(n).process = 0;**

**Event(n).seqControl = 1;**  % 在这里使用适当的“timeToNextAcq”

**n = n+1;**

**Event(n).info = 'Acquire 2nd half of aperture for zone 1';**

**Event(n).tx = 1;**

**Event(n).rcv = 2;**

**Event(n).recon = 0;**

**Event(n).process = 0;**

**Event(n).seqControl = 1;**

**n = n+1;**

**Event(n).info = 'Acquire 1st half of aperture for zone 2.';**

**Event(n).tx = 2;**

**Event(n).rcv = 3;**

**Event(n).recon = 0;**

**Event(n).process = 0;**

**Event(n).seqControl = 1;**

**n = n+1;**

**Event(n).info = 'Acquire 2nd half of aperture for zone 2';**

**Event(n).tx = 2;**

**Event(n).rcv = 4;**

**Event(n).recon = 0;**

**Event(n).process = 0;**

**Event(n).seqControl = 1;**  % 在这里使用适当的“timeToNextAcq”

**n = n+1;**

**Event(n).info = 'Acquire 1st half of aperture for zone 3.';**

**Event(n).tx = 3;**

**Event(n).rcv = 5;**

**Event(n).recon = 0;**

**Event(n).process = 0;**

**Event(n).seqControl = 1;**

**n = n+1;**

**Event(n).info = 'Acquire 2nd half of aperture for zone 3';**

**Event(n).tx = 3;**

**Event(n).rcv = 6;**

**Event(n).recon = 0;**

**Event(n).process = 0;**

**Event(n).seqControl = 1;**

**n = n+1;**

通过上述编程，本地内存中生成的 RF 数据行将由 3 次采集的多个部分组成。 组合数据到主机的发射将发生在帧的末尾，并且将包含与每行（最深区域）执行一次单次采集相同数量的数据。 当对该组合数据执行图像重建时，使用最深的区域 1 作为 ReconInfo.rcvnum 值，多个采集段之间的接缝通常不会显示，因为重建组合了从区域边界处的每个区域获取的 RF 数据， 从而平滑过渡。 由于 TX.Origin 对于线路上的多个焦点区域中的每一个都是相同的，因此线路的任何 TX 结构都可用于 ReconInfo.txnum。

当前软件版本的 ExampleScripts 目录中包含使用 L7-4 换能器进行两个发射聚焦区域扫描的示例。

1. **Using the Vantage System High Frequency Configuration**

**（使用 Vantage System 高频配置）**

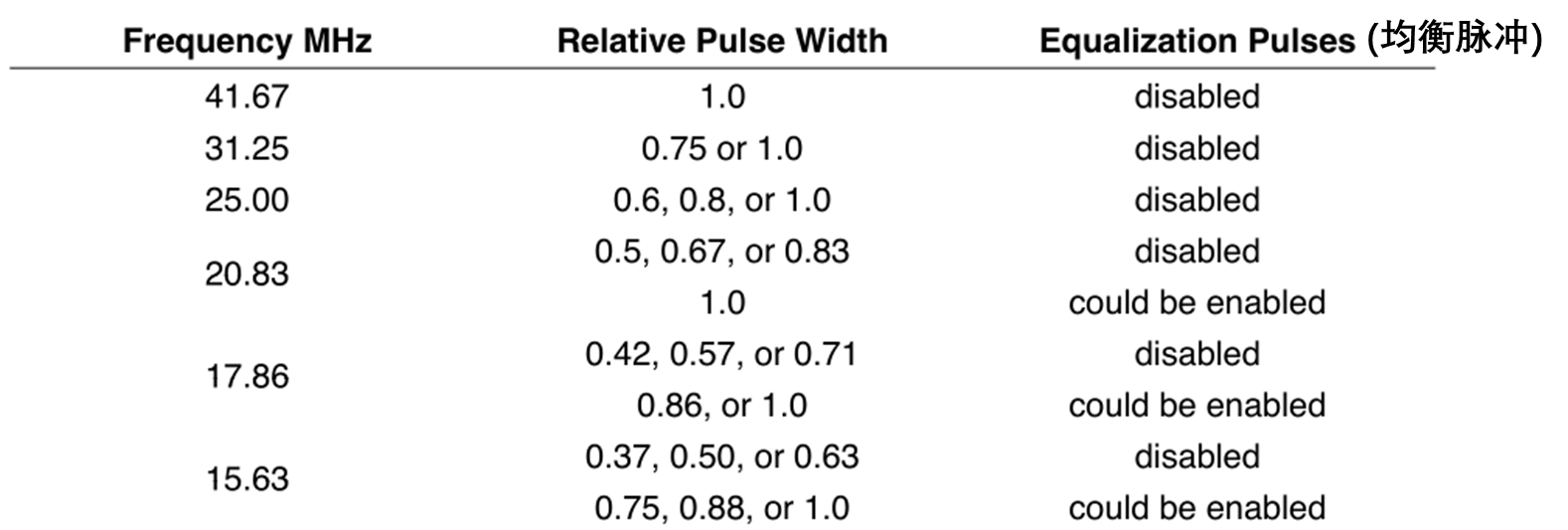
本节概述了 15 MHz 以上频率下发射和接收操作的编程技术、系统特性和限制。 除了下面提到的细节之外，为高频探头开发脚本与上面解释的“标准”频率系统配置相同。

**7.1 High Frequency Transmit（高频发射）**

对于高频配置，使用具有更宽带宽的不同变压器来支持高频操作。 虽然使用 TW 结构定义高频发射波形的功能与标准系统相同，但在更高频率下定义任意波形的灵活性降低了。这是因为必须应用两个系统发射约束：最小脉冲持续时间 和变压器饱和磁通限制。

**7.1.1 Minimum Pulse Duration (最小脉冲持续时间)**

高频发射器无法产生小于三个系统时钟周期或 12 纳秒 (3/250MHz) 的一致输出电平。 持续时间为 1 或 2 个时钟周期的脉冲对系统无害，但会产生不可预测的结果，这些结果因通道而异。 因此，可产生的最高频率周期性发射波形是具有 1.0 相对脉冲宽度的 41.67 MHz 方波。 另请注意，由于此限制，TW“参数”波形类型的均衡脉冲不应在某些更高频率下使用，因为均衡脉冲的周期仅为发射波形的一半。 下表总结了使用“参数”波形定义可在 15 MHz 以上产生的发射频率和相对脉冲宽度：



**表 7.1.1 支持的发射频率和高于 15 MHz 的脉冲宽度**

例如，以下三行代码定义了相对脉冲宽度为 0.8 的 25 MHz 双周期发射波形：

**TW.type = ‘parametric’;**

**TW.Parameters = [25.0, 0.8, 4, 1];**

**TW.equalize = 0;** % 禁用均衡脉冲

第三个参数的值为 4 表示以半周期为单位的突发持续时间； 请注意，当均衡脉冲被禁用时，这必须是一个偶数整数才能产生没有直流内容的波形。

**7.1.2 Transformer Saturation Flux Limit (变压器饱和磁通限制)**

发射变压器磁通限制限制了可在低发射频率下使用的最大发射电压，以防止变压器磁芯饱和。 对于高频配置，此限制为 6.25 伏特-秒（在大约 2015 年 9 月之前构建的系统上为 4.2 伏特-秒），比标准频率发射器允许的 25 伏特-秒限制更多。 如果脚本定义的发射波形将超过 Trans.maxHighVoltage 设置的此限制（请参阅 2.1），系统将退出并出现错误情况，指出允许保持在该波形的通量限制内的最大电压。 例如，2.5 MHz 方波的半周期持续时间为 0.2 微秒。 因此，对于 1.0 的相对脉冲宽度，高频配置的最大允许发射电压将为 21 伏特，因为：

**21 Volts \* 0.2 usec = 4.2 Volt\*usec**

对于使用均衡脉冲的相同波形，最大允许电压将增加到 42 伏。 如果除了均衡脉冲之外，相对脉冲宽度减小到 0.7，则最大允许电压将为 60 伏。 在启用均衡脉冲和 0.7 相对脉冲宽度的情况下，任何高达 96 伏系统最大值的发射电压都可用于 4 MHz 或更高的发射频率。

**7.2** **High frequency Receive Acquisition (高频接收采集)**

三种技术可用于高频接收采集，具体取决于探头的中心频率和带宽。

**7.2.1 4 samplesPerWave(每波4个样本)**

高频配置仍然可以使用通常用于标准频率系统的相同接收数据采集方案，RF 数据采样率设置为 4xTrans.frequency。 由于 HW 系统支持的最高 A/D 采样率为 62.5 MHz，因此接收数据的最大中心频率为 15.625 MHz，奈奎斯特带宽限制范围为 DC 至 31.25 MHz。 一个保守的经验法则是接收数字信号路径和 Recon 处理可以在采集中心频率 +/- 60% 的范围内提供非常好的性能，或者对于 62.5 MHz A/D， 大约 6 到 25 MHz 速度。 对于所需带宽落在该范围内的任何探头和应用，可以使用此采集方案并通过在 Receive.InputFilter 中指定带通滤波器系数以匹配所需的中心频率和带宽来实现非常好的性能。 请注意，如果您未指定 Receive.InputFilter，系统将分配一个以 Trans.frequency 为中心的默认宽带（大约 100%）滤波器，如果实际中心频率远高于 15 MHz，这显然不是最佳选择。

随系统提供的示例脚本“SetUpL22\_14vFlashAngles”就是这种运行状态的示例。 在此脚本中，Trans.frequency 设置为 15.625 MHz，因此接收 A/D 采样率为 62.5 MHz。 发射波形的标称频率为 17.86 MHz，已选择 Receive.InputFilter 系数以提供 12 至 24 MHz 的带宽，以 18 MHz 为中心。 有关更多解释性说明，请参阅脚本本身中的注释。

使用这种方法进行接收采集的脚本在高频配置和标准频率系统配置上同样有效。 但是请注意，与高频配置相比，标准频率系统在 15 MHz 以上的发射频率下的发射输出幅度和通道间均匀性将显着降低。

任何可能的接收采集方案都必须考虑的高频配置的另一个重要方面：在标准频率系统上为接收前置放大器提供可编程输入阻抗的网络已在高频系统中进行了修改，因此它提供了高 - 通滤波功能，在低于 20 MHz 的频率下滚降。 在带宽低于 20 MHz 的高频系统上使用探头时（例如上面引用的 L22-14v 示例），必须禁用此高通响应。 这是通过将输入阻抗编程为其“高阻抗”状态来实现的（请参阅示例脚本中的第 50 行，以及 3.3.3.1 了解更多信息）。

* + 1. **Interleaved sampling with** **4 samplesPerWave**

**(使用 4 个 samplesPerWave 的交错采样)**

**7.2.2.1 Overview（概述）**

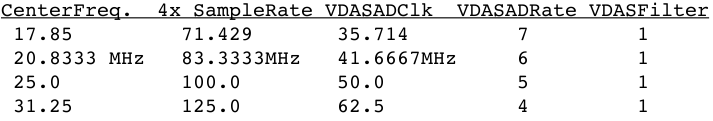
“交错采样”是一种采集方案，**用于在上述 25 MHz 带宽上限以上进行采样**，对于具有 4 个 samplesPerWave 的典型采集。 **交错采样的基本思想是通过组合来自连续两次发射-接收采集的样本，使A/D转换器的有效采样率翻倍，第二次采集的采样点比第一次采集的采样点移动了A/D采样周期的一半。 例如，可以组合以 62.5 MHz （即16ns）采样率进行的两次采集以产生 125 MHz 的有效采样率。** **所需要做的就是交错两次采集的样本，并将样本点的定时从一次采集转移到另一次采集 ，间隔8 ns。** 如果获取的数据来自 Vantage 系统上实施的往返发射-接收事件，则只需将 8 纳秒的额外发射延迟添加到其中一个事件相对于另一个事件的传输时间上，就可以轻松实现8 ns的相对移位。由于 Vantage 系统上的发射延迟由以 250 MHz 系统时钟速率运行的计数器提供，因此只需将两个计数添加到第二个事件的发射延迟值即可实现 8 纳秒的发射延迟偏移。

交错采样概念的唯一重要权衡是需要组合来自两个连续采集的数据。 这将可实现的最大采集 PRF（脉冲可重复频率） 降低了 2 倍，并且还要求在完成两次采集所需的时间间隔内，成像组织的任何运动（相对于换能器）不超过一个波长的一小部分。

**7.2.2.2 Implementation of Interleaved Sampling（交错采样的实现）**

下面列出的步骤概述了如何实施交错采样模式“NS200BWI”，每个步骤后都有一些讨论和指南。有关所有这些步骤的完整实施示例，请参阅示例脚本“SetUpL22\_14vFlashAnglesInterleave.m”。 本脚本使用**2X交错**，A/D速率为41.7 MHz，有效射频数据采集采样率为83.3MHz，允许在 20.83 MHz 的中心频率下进行正常的“每波 4 个样本”重建处理。 通读此示例脚本中包含的注释，并查看下面列出的每个实施步骤。

1. 从下表中确定最接近您的 Trans.frequency 值的支持中心频率。 在您的接收结构中，将 Receive.sampleMode 设置为“NS200BWI”，并将 Receive.demodFrequency 设置为表中的中心频率。 目标交错采样率将是中心频率的 4 倍，A/D 采样率将是交错采样率的一半。



**2.** 设置交错采集之间的发射延迟偏移。 对于来自 41.7 MHz (250 MHz / 6) A/D 速率的 2X 交错的 L22 示例，采样周期的一半是 250 MHz 系统时钟的 3 个周期，或 12 纳秒，因此应将此偏移量添加到 第一次采集的所有通道的发射延迟。 将偏移量添加到第一个发射会将此发射的 A/D 样本放在没有延迟的发射样本之前。 由于发射延迟以Trans.frequency的波长指定，我们需要将12纳秒的时间周期转换为Trans.frequency的波长。 由于一个波长代表时间的 1/Trans.frequency，因此我们希望添加的波长延迟部分由 12 nsec/(1/Trans.frequency) 给出。 由于 12 纳秒实际上是 0.5 倍 (1/41.6667 MHz)，我们可以根据以下公式计算延迟：

**delay(wavelengths) = 0.5 \* (Trans.frequency/41.6667MHz)**

注意：对于某些 4x 采样率（71.429 和 100 MHz），无法基于 250MHz 时钟为 TX.Delay 精确编程 1/2 周期 A/D 时钟延迟。 在这些情况下，重建精度可能会受到轻微影响。

**3.** 设置交错采集事件序列。 对于每个交错采集，在序列中请求两个单独的发送-接收事件。 除了增加 acqnum 值外，两个事件的接收结构必须相同，并且两个事件的发送结构也必须相同，除了从第 2 步添加到其中第一个的 TX.Delay 偏移量。

**4.** 在采集硬件系统中设置数字 RF 数据滤波器：请注意，由于这些滤波器分别对来自 2X 交错的每次采集的 RF 数据进行操作，因此它们不能用于为交错的 RF 样本实施任意定义的 FIR 滤波器。 关于交错的输出数据，这些滤波器表示 FIR 滤波器结构，其中每隔一个抽头系数设置为零。 这意味着它们提供的任何滤波都将在交错采样率的 1/4 左右进行镜像，从而产生以 Fs/4 为中心的对称频率响应。 例如，可以在 Receive.InputFilter 中定义看似高通滤波器的部分，交错输出 RF 数据的最终滤波器响应将是一个以 Fs/4 为中心的对称带通滤波器。 请注意，当前的 Vantage SW 版本不为交错操作状态提供 Receive.InputFilter 的默认设置。 因此，用户应始终在用户设置脚本中手动指定所需的 Receive.InputFilter 系数，即使对于平面“直通”滤波器也是如此。

**5.** 对于重建，Recon 和 ReconInfo 结构被定义为好像 RF 信道数据是在较高的交错采样率下采集的。 ReconInfo 应该仅引用交错接收对中的第一个接收，即具有额外发射延迟的接收。 重建软件在处理具有“NS200BWI”sampleMode 的 Receive 时，会自动交错 RcvBuffer 中的两次采集，这样 RcvBuffer 中的数据就好像是以较高采样率采集的一样。

**7.2.3 4/3 samplesPerWave**

Vantage 系统提供的另一种支持频率高于 25 MHz 的接收数据采集和处理的方法是“4/3 采样”，其中采样率 Fs 设置为 4/3 x Trans.frequency。 因此，传感器带宽集中在Fs/2和Fs之间，采样过程将使其混叠到从0到Fs/2的频率范围内。 这个概念工作得很好，但只在两个相当严格的限制范围内：首先，可用的成像分数带宽被限制在 67% 以下，因为 0.5 Fs 的奈奎斯特带宽以 0.75 Fs 为中心。 其次，与换能器相结合的系统必须采用非常有效的高通滤波器来抑制从 DC 到 Fs/2 的不需要的频谱。 该频谱中的任何噪声和检测到的信号都将混入 Fs/2 到 Fs 的所需信号中，从而降低灵敏度、分辨率和波束成形性能。

对于 4/3 采样方法的示例，可以想象一个探头具有大约 50% 的分数带宽，中心频率为 37.5 MHz（带宽从 28 MHz 扩展到 47 MHz）。 “BS67BW”采样模式会将 A/D 采样率设置为 50 MHz（37.5 乘以 4/3），从而产生 25 MHz 的奈奎斯特带宽。 37.5 MHz 的探头中心频率将混叠为 RF 数据输出样本中的 12.5 MHz。 另请注意，接收数据频谱将被混叠“折叠”：47 MHz 的上频带边缘将出现在输出数据的 3 MHz 处，而 28 MHz 的下频带边缘将出现在 22 MHz 处。 如果 Recon 处理函数识别出“BS67BW”的 Receive.sampleMode，系统会自动考虑接收数据的混叠和折叠，因此，检测到的输出数据将与以有效的150 MHz采样率进行采样所获得的结果相同。

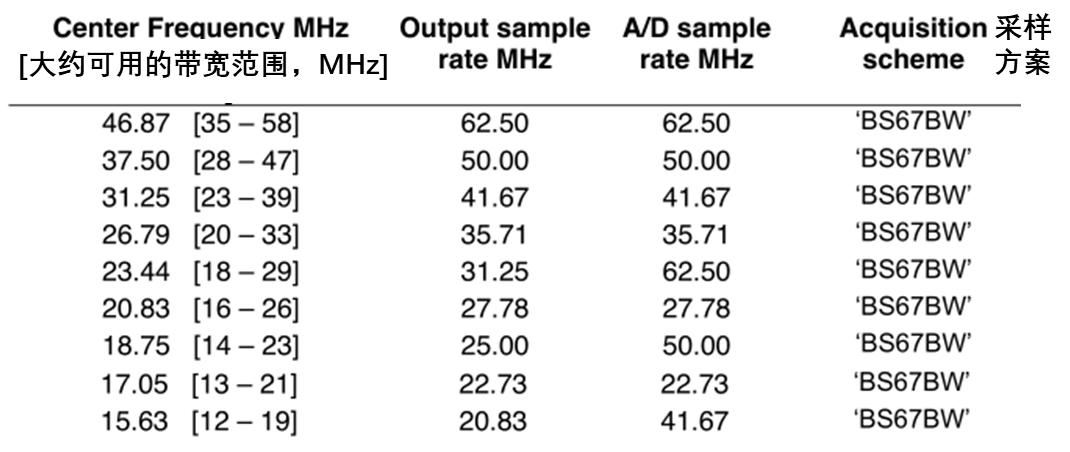
使用“BS67BW”采样模式获得良好性能的最大挑战是提供足够的高通滤波以拒绝频率低于 Fs/2 的信号和噪声，以避免污染从 Fs/2 到 Fs 的所需信号的混叠副本。 请注意，此滤波必须在 A/D 之前的模拟信号路径中完成，因为 A/D 处的采样是发生混叠的点。 在 Vantage 系统中，我们提供了两种实现高通滤波器的机制：

**- 前置放大器输入高通滤波器：**在具有高频配置的 Vantage HW 系统中，接收前置放大器输入端的反馈网络已被修改以引入一阶高通滤波器（对于标准配置，该网络用于 允许对具有平坦频率响应的接收输入阻抗进行可编程控制）。 当输入阻抗设置为最低水平时，此滤波器的截止频率设置为 20 MHz，由 RcvProfile.LnaZinSel 控制（参见 3.3.3.1）。 随着编程输入阻抗的增加，截止频率将逐渐降低，并且在最大“高阻”设置（RcvProfile.LnaZinSel = 31）下，反馈网络被禁用，因此高通滤波器不再存在。

**- “8/3”输入采样：**如果 Trans.frequency 设置的处理中心频率允许，系统将自动以所需输出采样率的两倍运行 A/D 转换器。 在 HW 系统提供的数字滤波中，Receive.LowPassCoef 属性将被编程为提供高通函数，截止值为 A/D 采样率的 ¼ (2/3 Fc)，然后向下采样到输出 4/3 采样率。 与模拟前置放大器输入高通滤波器相比，此 22 阶滤波器可在二次采样之前更好地抑制不需要的频谱，但限制是它只能在可以运行 A/D 时使用 在 8/3 Fc 采样率。

请注意，使用这些技术中的任何一种，通常都必须调整 Receive.InputFilter 提供的带通滤波。 由于它在接收数据频谱的混叠副本上运行，表观分数带宽将增加三倍（例如，如果探头具有以 Fc 为中心的 50% 分数带宽，则该频谱的以 1/3 Fc 为中心的混叠副本将 相对于混叠中心频率具有 150% 的表观分数带宽）。 当使用“BS67BW”sampleMode 时，现有的 Vantage SW 不会自动考虑这一点，因此系统提供的默认 Receive.InputFilter 系数可能具有过于严格的带宽。

下表列出了 Vantage 系统上高于 15 MHz 的“BS67BW”sampleMode 探头中心频率可实现的处理中心频率和 A/D 采样率：



示例脚本“SetUpL22\_14vFlashAngles67BW”提供了“8/3 输入采样”方法的示例。 在此脚本中，中心频率设置为 18.75 MHz，因此所需的 4/3 RF 数据采样率为 25 MHz。 A/D 在 50 MHz 下运行，在 A/D 之后的数字信号路径中提供高通滤波和二次采样 2。 有关此处介绍的概念的详细示例，请参阅该示例脚本及其中的解释性注释。

**完结！**