Application Note

Tools Summary

Yao-Sheng Tung

Verasonics Inc. | www.verasonics.com | support@verasonics.com

11335 NE 122nd Way, Suite 100, Kirkland, WA 98034 USA

Table of Contents

Table of Contents **2**
1. Introduction **3**
2. ArbWaveToolbox **3**
3. ColorMapTool **3**
4. ElementToChannelMapping **3**
4.1 Defining the Trans.Connector array for a user-supplied transducer 4
5. EventAnalysisTool **5**
6. PreSetTool **5**
6.1 Important Notes 6
7. PTool **6**
8. RF_FilterDesignTool **7**
9. RFDataViewer **7**
10. SaveRF **8**
11. ShearWaveVisualizationTool **8**
12. ShowTimeTag **8**
13. ShowTXPD **10**

Introduction

A number of useful tools and utilities can be found in the *Tools* subfolder within the Vantage project folder. An overview of each of these tools is listed here and some demonstration videos are provided on Verasonics website: http://verasonics.com/tools-demonstration/

ArbWaveToolbox

A collection of utilities and examples to help with defining and analyzing waveforms using the Vantage system's arbitrary waveform transmit features. Documentation on how to use these utilities is included within the *ArbWaveToolbox* subfolder of the *Tools* directory. A video is provided on Verasonics website.

ColorMapTool

From the clinical standpoint, a proper colormap allows eyes to distinguish the subtle difference in the tissue texture.

Verasonics provides a ColorMapTool for the user to adjust the gamma curve of the display window, as well as visualize the effect on image in real-time. The user can also save up to 5 curves for different tissues. A video is provided on Verasonics website.

ElementToChannelMapping

ElementToChannelMapping is a useful tool to help understand the actual mapping between transducer element numbers, system channel numbers, entries in the TX/Receive Apod arrays, and pin numbers at the transducer connector. The *Trans.Connector* array (or the selected column of *Trans.HVMux.Aperture* for HV Mux probes) defines the mapping from individual transducer elements to the associated channel in the Vantage HW system for the selected connector or connectors. These channels are identified in the *Sequence Programming* manual as "I/O channels", to distinguish them from physical hardware channels (in some cases, such as when using a single HDI connector on a Vantage 256 system, the number of I/O channels may be less than the number of physical HW channels). For all Vantage software releases since 2.11 there will always be an exact 1-to-1 mapping between I/O channels and columns of the Receive Data buffer in the Matlab workspace; the number of columns in the Receive buffer definition (*Resource.RcvBuffer.colsPerFrame*) will always be equal to the number of I/O channels for the selected connector(s).

In summary, the length of the Trans.Connector array will always be equal to the number of transducer elements identified in the Trans structure. The Nth entry in that array will define the connection for element N of the transducer definition, and the value of that entry will identify the I/O channel number the element is connected to. A value of zero means that element is not

connected at all, and cannot be used by the system. All of this remains exactly the same for the UTA system configuration, for all UTA adapter modules.

Defining the Trans.Connector array for a user-supplied transducer

If you are defining a script for a probe recognized by the Vantage software and supported through the *computeTrans* function, all of the element-to-channel mapping is done for you and most of the time you don't even have to think about it. But if you need to connect test equipment to a particular individual element signal at the probe connector, you will need to find the associated connector pin number (the Trans.Connector array defines the element-to-channel mapping but does not specify which physical connector pin is associated with each element). Even worse, if you are connecting a transducer that is not supported by *computeTrans* to the system, you will have to figure out how to define the Trans.Connector array. These issues were complicated enough on the original system; in the UTA system they get even more confusing since there are multiple UTA adapters supporting multiple connector types to choose from, and each UTA adapter and connector has a unique pin assignment at the connector and a unique mapping of those pins to system channels.

The Vantage SW includes some utilities to make the element-to-channel-to-connector pin mapping easier to understand:

showEL After running any script with VSX, and then exiting VSX, the Matlab workspace will still contain all of the structures that were used while the script was running. Type the command showEL at the command prompt, and a listing of all transducer elements will be displayed in the Matlab Command Window, showing the system I/O channel number, physical connector pin number, and Apod array entry number (applies to both Receive.Apod and TX.Apod) associated with that element. Enter the command with an argument, i.e. showEL(7), and the mapping only for the identified element will be displayed. While working with a script, this utility makes it quick and easy to find the connector pin number and I/O channel number (which is also the column number in the Receive Buffer for the associated receive data) for that element.

showCH and showAPOD These utilities are identical to showEL as described above and will display the same information, except that the listing will be sorted by I/O channel numbers or Apod array entries instead of by elements.

To define the Trans.Connector array for a probe that is not recognized by computeTrans, you must first determine from the probe design documentation which physical connector pin each element connects to. Then you can use one of the utilities described above to find the mapping from each connector pin to system I/O channel, so you can put the correct channel number in each entry of the Trans.Connector array. Refer to the Sequence Programming manual for additional information on integrating custom transducers on the system.

EventAnalysisTool

A very useful tool to assist with developing and debugging a SetUp script. After running a SetUp script, so its output is present in the Matlab workspace, just enter *EventAnalysisTool* at the Matlab command prompt and you will see a visual display of all events in the event sequence. Clicking on any Event will display the contents of structures indexed by that Event, making it easy to find and fix errors in indexing logic as well as helping you understand what the system will actually be doing as you step through each event in the event sequence. Please refer to the Application Note – EventAnalysisTool.pdf for more details.

Sometimes, debugging is very frustrating, so debugging capability is added into the EventAnalysisTool. In the EventAnalysisTol/Examples subfolder, six error scripts are provided to illustrate how to use EventAnalysisTool to find bugs in the script. The corresponding corrected scripts are also provided.

- 1) ErrorScript1 - Warning message if any of event has an empty field
- 2) ErrorScript2 - Each TX, Rcv, and Seq should be used
- 3) ErrorScript3 - Consistency of frameNum or BufferNum in Recon/ReconInfo
- 4) ErrorScript4 - A bug in the callback! The endDpeth of all receive used for one Recon should be identical
- 5) ErrorScript5 - Each Receive should not have the same bufnum, framenum, acqNum, and mode; otherwise the data will be overwritten to the same memory location.
- 6) ErrorScript6 - transferToHost must be unique in the event sequence.

Usually, the Recon and ReconInfo is complicated and, if the script crashes Matlab, you should use EventAnalysisTool to inspect the Recon/ReconInfo setting. A video is provided on Verasonics website.

PreSetTool

The preSetTool is used to save the configuration of the UI controls shown in the VSX GUI for most example scripts, including default setting (TGC, TGC All Gain, Digital Gain, High Voltage P1, High Voltage P2, Speed of Sound), PTool, ColorMapTool, filterTool and some predefined UI controls using VsSlider and VsButtonGroup definition. The intent is to allow user to save all system settings after optimization for a particular application, so the user can easily recall them later without having to manually readjust the controls.

Since 3.0, the source code of the preSet function has been embedded in the vsx_gui.m as nested functions, so no addition folder is provided in the Tools folder. As shown in the Script modification from 2_11 to 3_0.pdf, if the user-defined variables will be modified by the callbacks of any UI controls, these variables should be placed in the P structure, that will be saved in the preSet file, to make preSetTool function correctly. After clicking “Save”, the default

filename is defined by the title of display window. The user is able to rename and save all preSets in a desired folder. All configurations will be saved in a variable named "preSet", within the settings file you have defined. The user can use "Load" to restore the setting back to what's saved in the preSet file. A video is provided on Verasonics website.

Important Notes

- DO NOT load preSet file before the image is displayed. The Matlab may crash if the preSet file is trying to modify the variables in the workspace while the system is being initialized.
- The preSetTool is only used to save some variables in the Matlab workspace and the status of the UI controls of the SetUp scripts provided by Verasonics. All settings with custom GUI controls may not be reconfigured properly, depending on the complexity of the callback functions.
- The preSetTool does not support other custom UI controls. In addition, the Zoom and Pan in the Display panel are not saved, either.
- Since the SetUp script and UI controls are different among probes and applications, correct probe name and application should be indicated in the filename. Using a preSet file from a different script may cause unexpected errors or cause a software crash.
- If the callback function of a UI control will change the value or status of other UI controls, the preSetTool will not restore all parameters correctly.
- The callback functions of the Doppler Mode, Compression, and Reject will change the color map of the display window. The callback function of the Doppler Mode selection will overwrite the Map change caused by Compression and Reject. Therefore, after loading preSet, the execution of the callback function of Compression or Reject is required to display correct image. For instance, clicking the Reject slider and have it back to the correct value will produce the correct image associated with the preSet.

PTool

To assist in modifying image processing parameters, a new tool has been created, called PTool. This tool is invoked within a dropdown menu in the main GUI window, and brings up a new figure window with controls for modifying all of the Process image attributes. The Process structure, with any adjustments made to the processing attributes, can then be written to the Matlab command line, where it can be copied into the setup script.

RF_FilterDesignTool

A utility to assist with designing and reviewing the digital filters provided in the receive data path for each channel of the Vantage HW system. Please refer to Vantage Sequence Programming Manual, 3.3 Receive Object, and http://verasonics.com/resources-downloads/ - Bandwidth sampling white paper- for more information about the meaning of different sampleMode and how Verasonics implement the Bandwidth Sampling method (NS200BWI and BS67BW) for high frequency transducer. In addition to the documentation, A video is provided on Verasonics website.

RFDataViewer

showRcvData is a stand-alone utility to view the RF data in the RcvData buffer using a grayscale image and line plots.

When no arguments are provided, the data displayed uses default settings, listed below.

Each of the following variables can be set in the function call using the Matlab convention: showRcvData('variableName', numericalValue, ...)

variableNames	Default	Description
bufferNum	1	Receive Buffer index
frameNum	1	Frame number
acqNum	0	Acquisition number: =0 All =N Nth acquisition
sampleRange	[start:end]	range of samples to be plotted in image and line plots (default is entire frame)
chNums	[32 96]	channel numbers to plot as lines (=0 skips all line plotting) (values for 128 channels)
interpF	4	interpolation upsampling factor for the RF data line plots (=1 for no interpolation)
plotSamplePoints	0	=1 will plot '+' at each true data point, and 'o' at each interpolated sample, w/o line
cMax	500	absolute value of data above which color will be saturated black or white in the 2D image and scale line plots to [-2*cMax 2*cMax]
ordering	'channel'	'channel': plot data in channel order, exactly as stored in RcvData structure 'element': plot data in element order, using Trans.Connector

		vector to unscramble the RF data

By default, the program plots an entire frame, and marks the boundaries between acquisition events in the frame. The zoom feature is turned on by default, so one can use the figure's rubber band box zoom control to examine data details for a smaller region within a frame. If the sampleRange is specified explicitly, the frameNum is ignored, and the plotted range corresponds to the samples in the receive buffer. In this case, the acquisition boundaries are not marked. If an acquisition number is provided, that acquisition (for frame=frameNum) is plotted. The data is also interpolated by interpF to make the image smoother. If ordering is by element, then chNums is used to represent element number(s) to plot. This version of the function is intended to be used from the Matlab interface after running VSX. A version suitable for use as an external function, which requires that the only input argument be a local RF data buffer variable name, can be developed from this function.

SaveRF

The “saveRF” function in this folder will be invoked while it’s selected while executing VSX. This function allows user to save RFdata to a desired location for post processing.

ShearWaveVisualizationTool

The example script of Shear wave imaging is located in the ExampleScripts/Vantage 128 and 256/Misc/RadiationForce_ShearWaveVisualization, not the Tools folder. However, it could be considered as a useful tool for shear wave application. Both documentation and video are provided for more details.

ShowTimeTag

This is a new user-programmable feature added in the 3.1 release. When enabled, the time tag feature will overwrite the first two samples of each receive data acquisition event with a 32 bit "time tag" value taken from a free-running counter in the hardware system. As a result, the RF data stored in memory from every acquisition event will include a time tag identifying the precise time at which it was acquired relative to other acquisition events with a resolution of 25 usec and a range greater than 24 hours. If desired, you can also associate the relative time tag values with an absolute time and date taken from the host computer’s time / date functions.

When a script starts running the time tag feature will be disabled by default. It can be enabled or disabled whenever desired by using the mex command "enableAcqTimeTagging" with the following syntax:

result = enableAcqTimeTagging(enable);

The argument "enable" must be a Matlab string, set to the value ’0’ to disable time tags or the value ’1’ to enable them. The returned value "result" will be a Matlab string variable, set to the value ’Success’ if the command completed successfully, or some other string identifying the problem if there was an error.

The enable status is "global", in the sense that while time tags are enabled they will be added to every RF data acquisition event executed by the HW system, and they will be applied to every receive channel included in the DMA transfer to the Receive Buffer in the host computer. There is no mechanism for applying the time tag selectively to individual channels. The Recon function is not aware of the presence of time tags, so if the Recon processing makes access to the first two samples from a receive event, the time tag values will be used as if they were legitimate RF data. In most acquisition scripts this is not a problem since the first two samples will come from before or during the transmit activity, and thus the actual sample values are typically meaningless anyway. If it was necessary to prevent the time tags from corrupting the receive data samples you could either arrange the acquisition timing so the first two samples would not be used for RF processing, or you could use an external processing function to force them to zero after the time tag value had been read.

Enabling and disabling the time tag feature will not affect the state of the 32-bit time tag counter. It is reset to the zero state at system power-up and then free runs continuously as long as the system is powered up. Since it increments every 25 usec, the 32-bit value will wrap around back to zero after approximately 107,000 seconds or 29.8 hours. If desired, you can also program the counter to wrap back to zero after exactly 24 hours or 86,400 seconds. You can also reset the counter back to zero whenever desired. Both of these options are controlled through a separate mex function, "setTimeTaggingAttributes", with the following syntax:

result = setTimeTaggingAttributes(wrap24, reset);

The argument "wrap24" must be a Matlab string, set to either ’1’ or ’true’ to enable the 24-hour wraparound, or ’0’ or ’false’ to allow the full 32 bit count wraparound. The "reset" argument will force the counter value back to zero if it is set to ’1’ or ’true’, at the point in time when the command is executed. The "reset" argument will do nothing if it is set to ’0’ or ’false’.

To map the time tag counter value to an absolute time and date, simply record the time and date from the host computer using an appropriate Matlab function, immediately after sending the "reset" command to the time tag counter. Then subsequent time tag values can be interpreted as an offset from that recorded time.

The example script "SetUpL11_4vFlashTimeTag" provides an example of enabling and disabling the time tag feature as well as extracting the time tag value in an external processing function and displaying it. The utility function "ShowTimeTag" provides an example of extracting time tag values from the receive data buffer in the Matlab workspace.

ShowTXPD

When a TX structure is present in the Matlab workspace, invoking showTXPD will produce and display beam plots of the transmit beam that will be produced by each individual TX structure. A detailed documentation is provided in the ShowTXPD folder.

Verasonics, Inc.
11335 NE 122nd Way, Suite 100,
Kirkland, WA 98034
USA