Advanced OEM Solutions

MATLAB stub

Version 1.1

REVISION HISTORY

Date	Rev. No.	Description
2016/10/01	1.0	General Edit – Replaces Matlab Driver Documents
2016/11/18	1.1	Updated §5 Examples: (ExampleHWDlg.m 1.1.5.4d and newer, ExampleHWFifo.m and ExampleHWNoFifo.M 1.1.5.4e and newer). §3 FIFO vs. Standard Buffer: Additional FIFO explanation.
2017/01/13		Minor fix

CONTENTS

1. Introduction

The MATLAB stub is an open source library (except for the low-lever driver API) that can be used for three different purposes:

- Connecting to an OEM-PA device and managing it
- Displaying the main items of the high level calculator
- Programmatically controlling UTToolBox (for generating OEMPA files)

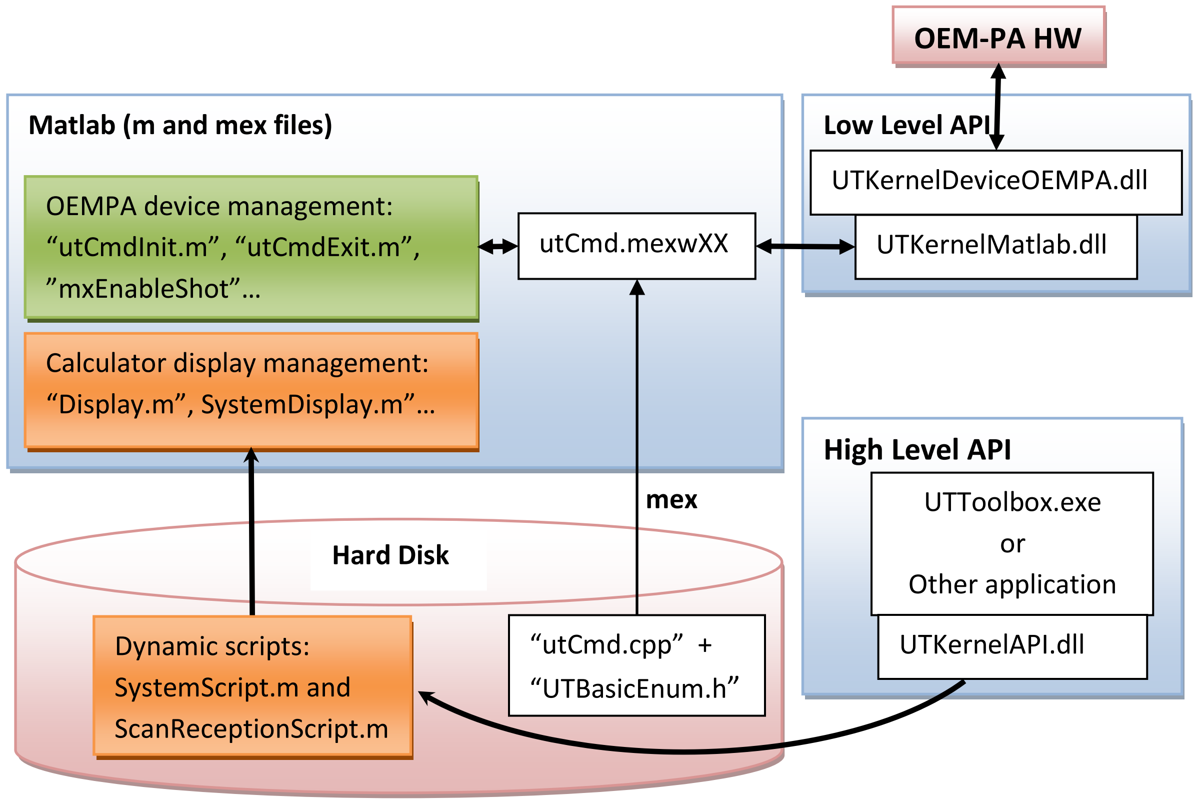
This document explains the use of the MATLAB stub. There is no link between these modes of functionality. You can build an OEMPA file and display the specimen, the probe, and the beam. Independently, you can instantiate an OEM-PA device and manage it (load the OEMPA file, change settings, retrieve data) but MATLAB cannot link those two operations.

2. Architecture

There are two sets of m-files that are independent of each other:

- M-files to manage the OEM-PA device.
- M-files to display calculator results.
    - Static files that are always the same.
    - Dynamic files that are created by the high level API to define the specimen, probe, etc…

Here is the flowchart:


OEM-PA HW
Matlab (m and mex files)
OEMPA device management: “utCmdInit.m”, “utCmdExit.m”, ”mxEnableShot”…
Calculator display management: “Display.m”, SystemDisplay.m”…
utCmd.mexwXX
Low Level API
UTKernelDeviceOEMPA.dll
UTKernelMatlab.dll
High Level API
UTToolbox.exe or Other application
UTKernelAPI.dll
mex
Hard Disk
Dynamic scripts: SystemScript.m and ScanReceptionScript.m
“utCmd.cpp” + “UTBasicEnum.h”

The mex file (“utCmd.mexw32” or “utCmd.mexw64”) can be compiled with the following MATLAB command line*:

```
>> mex utCmd.cpp
```

If you run this command for the first time then MATLAB may ask you the location of the compiler. By convenience you can find “utCmd.cpp” in the project of “UTKernelMatlab” but this project doesn’t compile it. It is good practice to recompile the mex file the first time you use it.

** Prior to 1.1.5.3w, this command was >> mex utCmd.c.*

2.1 Folders

There are 3 working directories:

Working directory	Comment
“C:\Program Files\AOS\OEMPA [version]”	DLL (“UTKernelMatlab.dll” and others)
“C:\Program Files\AOS\OEMPA [version]\matlab”	M files and mex file (low level API).
“C:\ProgramData\AOS\OEMPA [version]\Sources\Matlab”	M files for the calculator display (high level API): static m files and dynamic m files (generated by the high level API).

3. FIFO vs. Standard Buffer

The next few sections concern how to connect to, manage and retrieve data from the OEM-PA through MATLAB.

Depending on if the FIFO is enabled, one of two different APIs should be used to retrieve A-scan data (the C-scan FIFO has not yet been implemented). If the FIFO is enabled then the callback is not registered and the data should be read from the FIFO (see 7.2 FIFO). If the FIFO is disabled then another buffer is instantiated in the MATLAB stub, this buffer is only able to store one sequence (see 7.1 No FIFO).

Only the 64-bit version supports use of the FIFO. It is not available in the 32-bit version.

The API without the FIFO enabled is easier to understand. However, to sustain high throughput, the FIFO should be enabled to avoid data loss. Once the FIFO is full the first sequence will never be a whole sequence (that is “mxGetAcquisitionAscanFifoIndex” will return an index list for which some items will be -2), because new data will constantly overwrite the oldest data stored in the FIFO. This may occur either because the throughput is too high or due to a SW coding bug that would need to be fixed. Usually, the only way to recover from this error is to flush the FIFO and to restart. Before enabling the pulser it is good practice to flush the FIFO.

4. DLL History

Prior to version “1.1.5.3w”, the input mex file name was “utCmd.c” in the folder “C:\Program Files\AOS\OEMPA [version]\matlab”. The first task of the mex file was to load the MATLAB stub “UTKernelMatlab.dll” that itself was loading the OEM-PA driver.

From version “1.1.5.3w” forward, the input mex file name is “utCmd.cpp” in the same folder. The first task of the mex file is now to load the first level MATLAB stub “UTStubMatlab.dll”. This DLL is responsible for loading “UTKernelMatlab.dll”. The old utCmd.c was difficult to debug, so most of its code has been moved in “UTStubMatlab.dll” which is easier to debug.

The MATLAB scripts and the mex file should be in the same folder but the MATLAB stub could be in a different folder. The driver DLL should be in the same folder as the MATLAB stub.

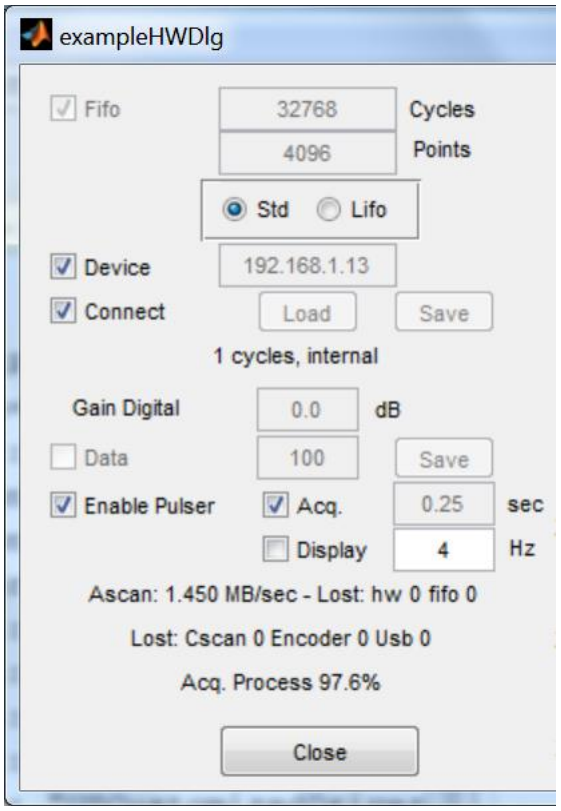
5. Examples

There are four examples available that are relevant to connecting to and managing the OEM-PA:

Name	Comment
“ExampleHWFifo.m” “ExampleHWNoFifo.m” (*Replaces “ExampleHW.m”*)	Basic example (FIFO enabled or disabled) to load an OEM-PA setup file and to acquire a few A-scans. Included in the standard AOS package. *For versions prior to 1.1.5.4e “ExampleHW.m” was included. For newer versions it has been split into “ExampleHWFifo.m” (FIFO enabled) and “ExampleHWNoFifo.m” (FIFO disabled).*
“ExampleHWDlg.m”	A dialog (MATLAB GUI) to test the main options and to display the sequence (B-scan). Included in the standard AOS package beginning with version 1.1.5.3v. This example supports FMC (display raw data). Ability to save acquisition data as a MATLAB file (internal mode or encoder mode) is included as of version 1.1.5.4d.
“AOS_FMC_example.m”	An example about FMC (FIFO enabled). This example is not included in the standard AOS package.

5.1 ExampleHWDlg

Here is an explanation of the dialog from “ExampleHWDlg.m”:



exampleHWDlg

✓ Fifo | 32768 Cycles | 4096 Points

● Std ○ Lifo

✓ Device | 192.168.1.13

✓ Connect | Load | Save

1 cycles, internal

Gain Digital | 0.0 dB

☐ Data | 100 | Save

✓ Enable Pulser | ✓ Acq. | 0.25 sec

☐ Display | 4 Hz

Ascan: 1.450 MB/sec - Lost: hw 0 fifo 0

Lost: Cscan 0 Encoder 0 Usb 0

Acq. Process 97.6%

Close

------------------- CONTROLS --------------------

“Fifo”: checkbox to enable the FIFO.

“Std”: standard function to read data in the FIFO (mode First In First Out).

“Lifo”: the last data are read first in the FIFO (mode Last In, First Out).

“Cycles”: maximum allocated cycles in the FIFO. 32768 is able to process FMC 128 (2x128x128=32768 that is 2 sequences).

“Points”: maximum number of A-scan points for each cycle.

“Device”: instantiation of a new device with the corresponding options (FIFO enabled or disabled, “Cycles” and “Points”).

“Connect”: connect/disconnect.

“Load”: load setup file (file to HW). Internal and/or encoded trigger modes are possible.

“Save”: save setup file (readback the HW to file).

“Gain Digital”: to send a new value to the HW (this value is not updated via a readback from HW or from the setup file).

“Enable Pulser”: enable/disable pulser.

“Data”: to save acquisition data as a MATLAB file. The text box controls the maximum count of sequences saved in the MATLAB file.

“Save”: this button can be pressed to save the acquisition data into your MATLAB file.

“Acq.”: enable the acquisition timer. The timer period (“sec”) specifies when to read HW data.

“Display”: enable the display timer. The frequency (“Hz”) specifies how often B-scan display is updated. The frequency is the theoretical refresh rate.

------------------- STATUS --------------------

Three lines of status are available.

The first line displays the A-scan throughput and A-scans lost in the HW (“hw”) and A-scans lost in the FIFO (“fifo”).

The second line displays other lost data: “C-scan”, “Encoder”, “Usb”.

The third line displays two items: First, the percentage of the throughput which data has been caught by the “Acq.” Timer. Second, the real “Refresh Rate”.

6. Setup API Commands

This section presents a list of available functions in MATLAB for managing the OEM-PA. See the included examples specifics on usage.

Here is the API (any version):

Function name	Comment
utCmdInit	This function should be the first call required to load the MATLAB stub “UTKernelMatlab.dll”. The full path of the dll is the single input parameter. The returned value (version “1.1.5.3w” and later) is 0 in case of error and a non null integer otherwise.
utCmdExit	This function should be called before closing MATLAB to free the MATLAB stub (any device that has been instantiated will be automatically deleted).
utCmdNewDevice	Instantiation of a new device. 3 inputs are used, one of them is to enable or disable the FIFO.
mxDeleteDevice	To delete a device previously instantiated by “utCmdNewDevice”.
mxConnect	Connect/disconnect to the device.
mxReadFileWriteHW	Load setup file (OEM-PA file for example).
mxGetSWCycleCount / mxGetHWCycleCount / (Previously mxGetCycleCount)	Get the cycle count. “GetSWCycleCount” / “GetHWCycleCount” are used with versions 1.1.5.3m and newer. “GetHWCycleCount” returns the same cycle count as the values in the setup file. “GetSWCycleCount” is the same than “GetHWCycleCount” except for FMC, in that case the value is the count of sub-cycles. “GetCycleCount” was used with versions 1.1.5.3l and older.
mxEnableShot	Enable/disable the pulser.

New API commands to update hardware UT parameters beginning with version 1.1.5.3f (“mxSetFilter”, “mxSetFilterIndex” available beginning with version 1.1.5.3l). The commands “mxLockDevice” (before) and “mxUnlockDevice” (after) should be called anytime a HW setting is updated (see *Software _API.pdf* 4.3 “Low Level API”):

Function name	Comment
mxSetAscanStart	Update the A-scan start.
mxSetAscanRange	Update the A-scan range.
mxSetGainAnalog	Update the analog gain.
mxSetGainDigital	Update the digital gain.
mxSetFilterIndex	Update the filter index.
mxSetFilter	Update the filter coefficients.

These functions are linked to the low level functions of the Low Level API, which allows for UT parameters to be quickly changed with having to load a new OEMPA file. As the MATLAB stub is open source, you can add your own Low Level function to this list.

Special functions to check FMC setup (available since version 1.1.5.3):

Function name	Comment
mxGetFullMatrixCapture	Get the FMC element range: start, stop and step (step only available from version 1.1.5.3t).
mxGetFMCTimeLimitation	Get FMC TimeSlot limitations.

7. Acquisition API Commands

This section presents a list of available functions in MATLAB for retrieving data from the OEM-PA. See the included examples for specifics on usage.

Common API commands to monitor lost data:

Function name	Comment
mxGetLostCountAscan	Get the count of lost A-scans in the HW.
mxGetLostCountCscan	Get the count of lost C-scans in the HW.
mxGetLostCountEncoder	Get the count of lost encoder triggers in the HW.
mxGetLostCountUSB3	Get the count of lost USB3 data in the HW.
mxGetAcquisitionAscanFifoStatus	Get the count of lost data in the FIFO.
mxResetCounters	Reset those counters and also the FIFO counters if the FIFO has been enabled.

7.1 No FIFO

API commands for A-scans:

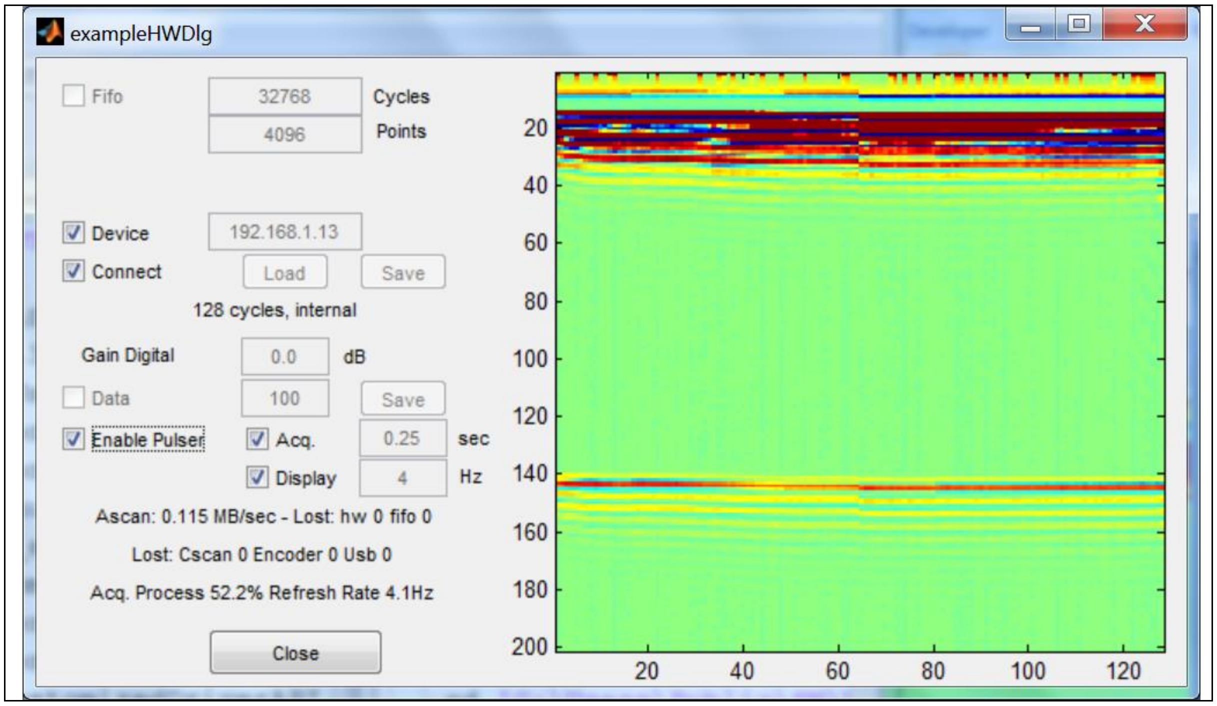
Function name	Comment
mxGetAcquisitionAscanCount	Get the A-scan count set in the callback.
mxGetAcquisitionAscanData	Get one and only one A-scan for the specified cycle.
mxGetAcquisitionAscanBitSize	Get the A-scan bit size and sign. Values returned are strings; “8Bits”, “12Bits”, “16Bits” for bit size and “Signed” or “Unsigned” for sign. This data is returned by the callback.
mxGetAcquisitionAscanSize	Return the byte size of the A-scan. This data is returned by the callback.
mxGetAcquisitionAscanEncoder	Return the encoder positions.

To get the full sequence a loop is required like the following (see “ExampleHWDlg.m”, the assumption is that all A-scan have the same size):

```
acqAscanCount = mxGetAcquisitionAscanCount(deviceId);
if acqAscanCount>=acquisitionTotalAscan+CycleCount
   A-scan = [];
```

```
        for iCycle=0:CycleCount-1
           data = mxGetAcquisitionAscanData(deviceId, iCycle);
           A-scan = [A-scan data'];
           end
        acquisitionTotalA-scan = acqAscanCount
   end
```

CycleCount is the cycle count in the sequence. Example with a linear scan 128 elements:



32768 Cycles
4096 Points
192.168.1.13
128 cycles, internal
Gain Digital 0.0 dB
Data 100
Acq. 0.25 sec
Display 4 Hz
Ascan: 0.115 MB/sec - Lost: hw 0 fifo 0
Lost: Cscan 0 Encoder 0 Usb 0
Acq. Process 52.2% Refresh Rate 4.1Hz

API commands for C-scans:

Function name	Comment
mxGetAcquisitionCscanCount	Get the C-scan stream count sent in the callback.
mxGetAcquisitionCscanAmplitude	Get amplitudes of one sequence (all cycles) for the specified gate number.
mxGetAcquisitionCscanTimeOfFlight	Get times of flight of one sequence (all cycles) for the specified gate number.
mxGetAcquisitionCscanEncoder	Return the encoder positions.

The buffer is only able to store one sequence. Old data are overwritten by new data, this is why at any time usually two different sequences are saved in the buffer. Here is an example:

Cycle	0	1	2	3	4	5	6	7
Sequence	5	5	4	4	4	4	4	4

After saving the next A-scan, the buffer looks as follows:

Cycle	0	1	2	3	4	5	6	7
sequence	5	5	**5**	4	4	4	4	4

The sequence number and the cycle number are not accessible, so the encoder value could be for a different sequence/cycle than the one you expect.

7.2 FIFO

By using the FIFO, it is possible to store more than one sequence. Also, sequence number and cycle number becomes accessible. A good practice is to allocate at least 2 sequences in the FIFO so that one full sequence can be read from it.

The following old API for A-scans is still available:

Function name	Comment
mxGetAcquisitionAscanCount	Get the A-scan count sent to the FIFO. Note: It is better to call the new function "mxGetAcquisitionAscanFifoStatus"
mxGetAcquisitionAscanBitSize	Get the A-scan bit size, 0 for 8 bits, 1 for 12 bits, 2 for 16 bits and 3 for 8 bits logarithmic data. The sign of the data is also returned. This data is loaded from the setup file.
mxGetAcquisitionAscanSize	Return the byte size of the A-scan. This data is loaded from the setup file.

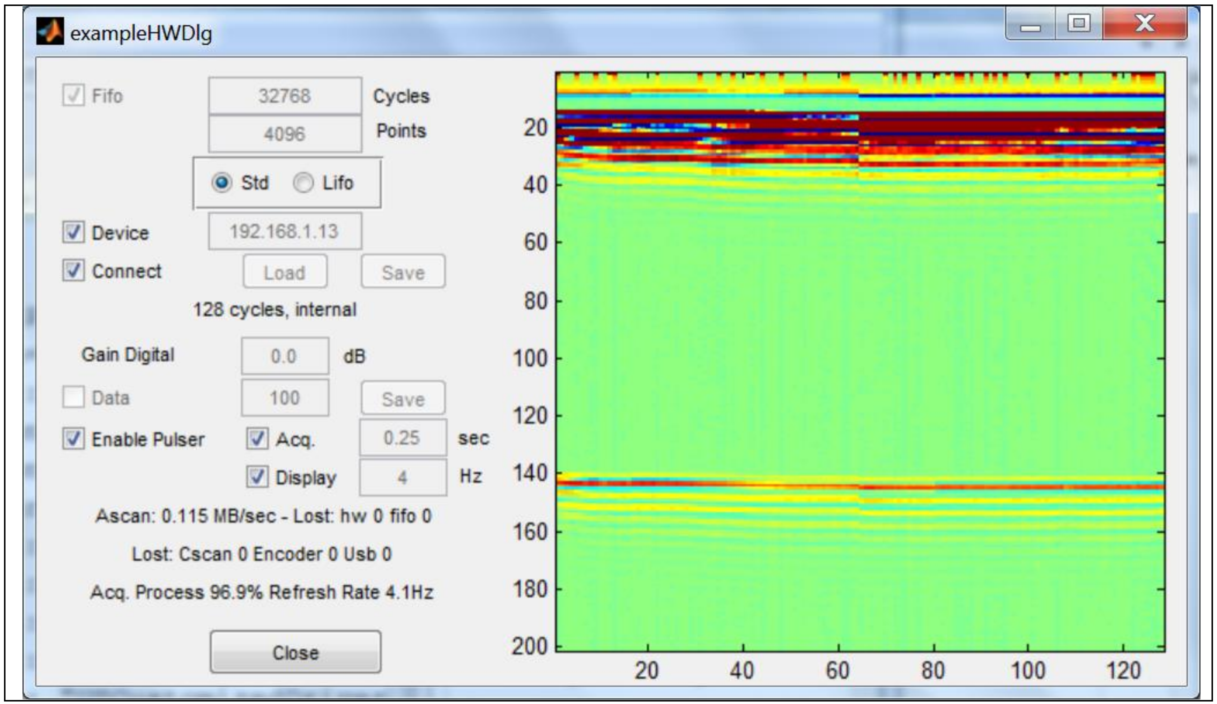
The new A-scan API is the following (from version "1.1.5.3w"):

Function name	Comment
mxGetAcquisitionAscanFifoStatus	To get the different FIFO counters, that is: current data count in the FIFO, data lost in the FIFO, total data count inputed in the FIFO, total data byte size input to the FIFO.
mxGetAcquisitionAscanFifoIndex	To get the FIFO item list (First In First Out). If any item of the first returned output is -2, it means that the first sequence is not a whole sequence. For example if the FIFO is full then most of the time the first sequence will not be a whole sequence (because the new data from HW will continuously overwrite the oldest data saved in the FIFO). See the FIFO documentation in *Software_API.pdf* to know what the item list is. You can also take a look to the script of this function.
mxGetAcquisitionAscanLifoIndex	To get the FIFO item list (Last In First Out). This function is better than "mxGetAcquisitionAscanFifoIndex" if you don't want to

	process all data. For example if you want to display the B-scan, it is better to display the most recent acquisition data. Assumption: all A-scans in the item list should be the same length.
mxGetAcquisitionAscanFifoData	To get an A-scan for the specified item list (usually one full sequence is returned). Sequence number, cycle number and encoder are also returned. The input parameter is delivered by calling “mxGetAcquisitionAscanFifoIndex” or “mxGetAcquisitionAscanLifoIndex”. Assumption: all cycles should have the same point count otherwise no data is returned.
mxFlushAcquisitionAscanFifo	To remove all items in the FIFO (the pulser should be disabled before calling this function).

“mxGetAcquisitionAscanFifoIndex” should be used if you want to acquire all data without any data loss. If you just want to display the last acquisition data then “mxGetAcquisitionAscanLifoIndex” is simpler to call (because the fifo is automatically flushed).

Example with a linear scan 128 elements:



128 cycles, internal

Ascan: 0.115 MB/sec - Lost: hw 0 fifo 0

Lost: Cscan 0 Encoder 0 Usb 0

Acq. Process 96.9% Refresh Rate 4.1Hz

The new C-scan API is the following (from version “1.1.5.3w”):

Function name	Comment
mxGetAcquisitionCscanFifoStatus	To get the different FIFO counters, that is: current data count in the FIFO, data lost in the FIFO, total data count inputed in the FIFO, total data byte size inputed in the FIFO.

mxGetAcquisitionCscanFifoIndex	To get the FIFO item list (First In First Out). See the FIFO documentation to know what the item list is.
mxGetAcquisitionCscanLifoIndex	To get the FIFO item list (Last In First Out).
mxGetAcquisitionCscanFifoData	To get amplitude and time of flight for the specified item list (usually one full sequence is returned). Outputs are 2D arrays because 4 gates are returned. Sequence number, cycle number and encoder are also returned.
mxFlushAcquisitionCscanFifo	To remove all items in the FIFO (the pulser should be disabled before calling this function).

8. Calculator display

This section covers the secondary function of the MATLAB stub, which is useful for displaying results of the high level calculator. It can help you ensure that input parameters (probe, specimen, etc…) are correct.

Only m-files are used for this feature - you don't need to load any other dll or mex file.

8.1 Dynamic M files

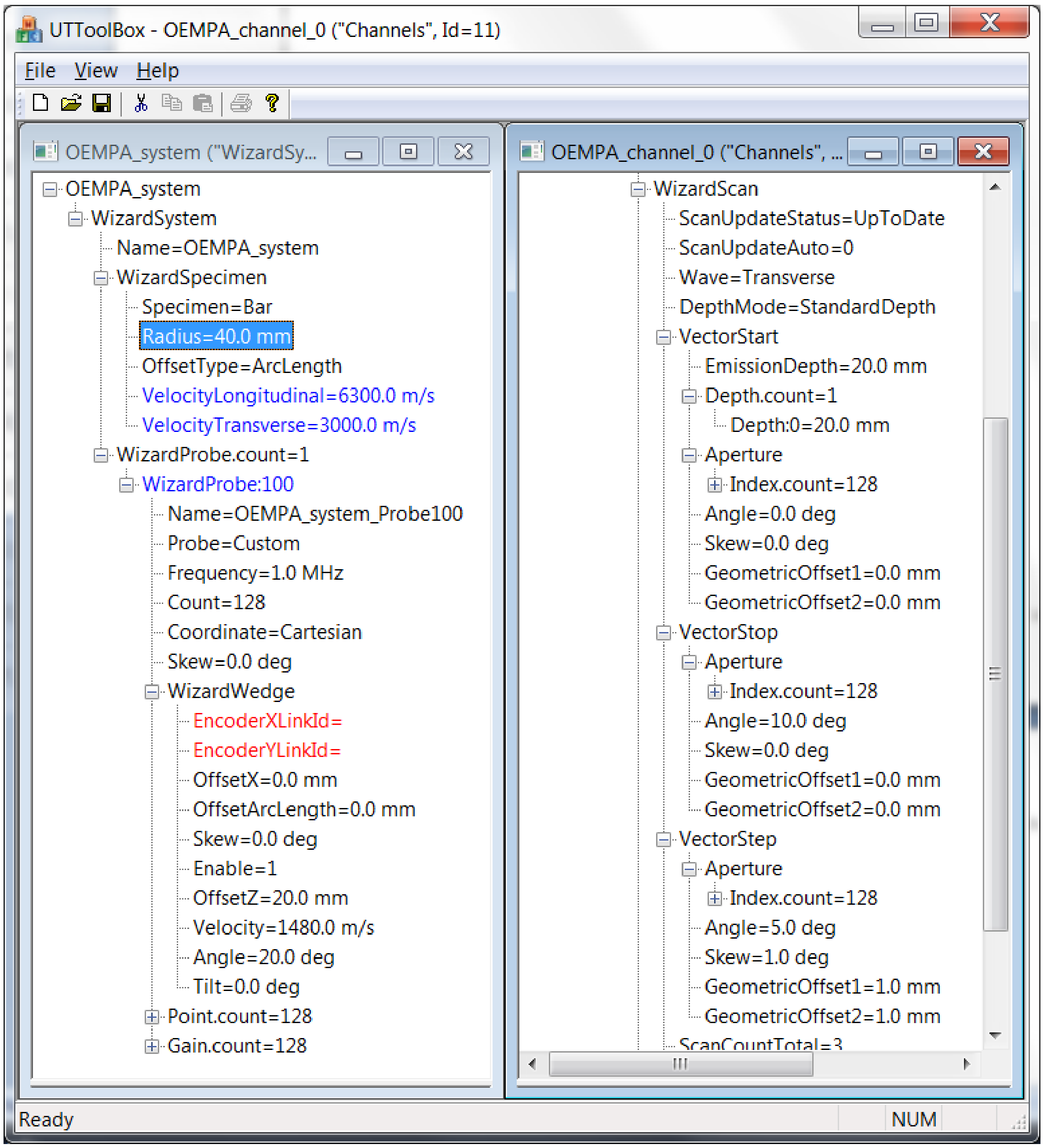
The high level "UTKernelAPI.dll" is able to output the two calculator files ("SystemScript.m" and "ScanReceptionScript.m") in the folder "C:\ProgramData\AOS\OEMPA [version]\Sources\Matlab. These .m files can be interpreted by MATLAB to display a figure with the probe, specimen, and the beams.

To enable "UTKernelAPI.dll" to output the two files, you have to run the following MATLAB command:

```
>> mxSetCfgKernelFile('DisplaySystemEnable',1)
>> mxSetCfgKernelFile('DisplayScanReceptionEnable',1)
```

8.2 Example

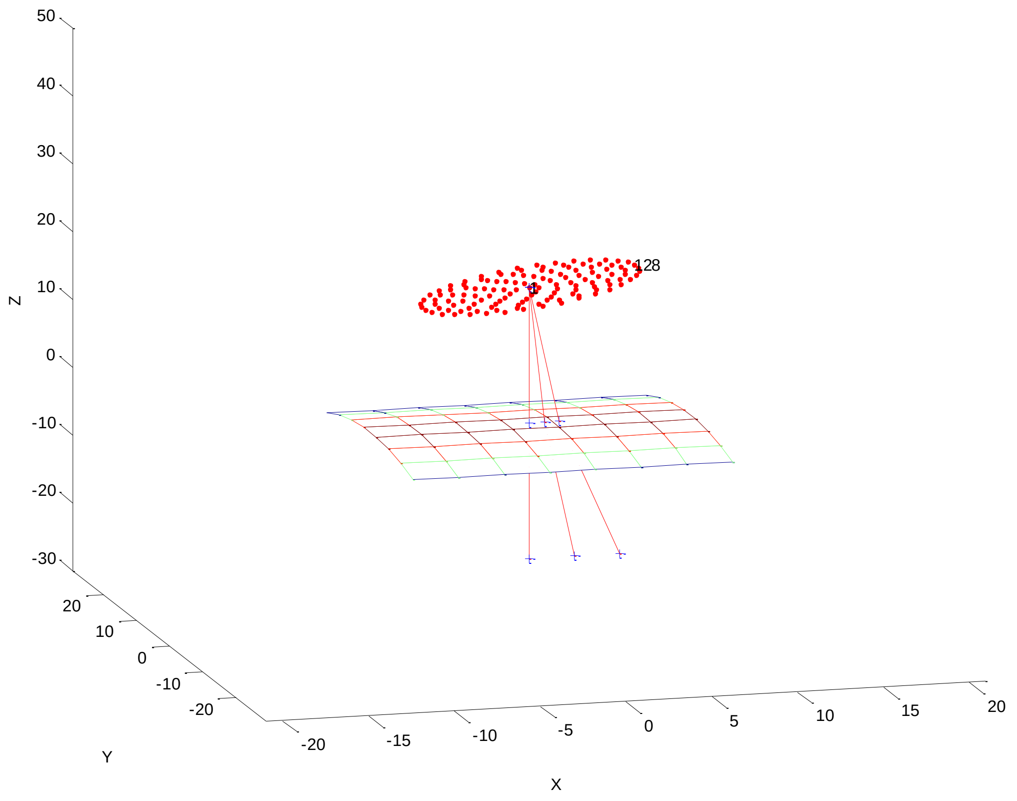
Let's take an example (matrix is supported only with version 1.1.5.0).


UTToolBox - OEMPA_channel_0 ("Channels", Id=11)
File
View
Help
OEMPA_system ("WizardSy...
OEMPA_channel_0 ("Channels", ...
Ready
NUM

```
OEMPA_system
  WizardSystem
    Name=OEMPA_system
    WizardSpecimen
      Specimen=Bar
      Radius=40.0 mm
      OffsetType=ArcLength
      VelocityLongitudinal=6300.0 m/s
      VelocityTransverse=3000.0 m/s
    WizardProbe.count=1
      WizardProbe:100
        Name=OEMPA_system_Probe100
        Probe=Custom
        Frequency=1.0 MHz
        Count=128
        Coordinate=Cartesian
        Skew=0.0 deg
        WizardWedge
          EncoderXLinkId=
          EncoderYLinkId=
          OffsetX=0.0 mm
          OffsetArcLength=0.0 mm
          Skew=0.0 deg
          Enable=1
          OffsetZ=20.0 mm
          Velocity=1480.0 m/s
          Angle=20.0 deg
          Tilt=0.0 deg
        Point.count=128
        Gain.count=128
```

```
WizardScan
  ScanUpdateStatus=UpToDate
  ScanUpdateAuto=0
  Wave=Transverse
  DepthMode=StandardDepth
  VectorStart
    EmissionDepth=20.0 mm
    Depth.count=1
      Depth:0=20.0 mm
    Aperture
      Index.count=128
    Angle=0.0 deg
    Skew=0.0 deg
    GeometricOffset1=0.0 mm
    GeometricOffset2=0.0 mm
  VectorStop
    Aperture
      Index.count=128
    Angle=10.0 deg
    Skew=0.0 deg
    GeometricOffset1=0.0 mm
    GeometricOffset2=0.0 mm
  VectorStep
    Aperture
      Index.count=128
    Angle=5.0 deg
    Skew=1.0 deg
    GeometricOffset1=1.0 mm
    GeometricOffset2=1.0 mm
  ScanCountTotal=3
```

The result of the command line “Display(0)” is the following:



9. Creating an OEMPA file (ExampleWizard.m)

The final function of the MATLAB Stub is the ability to programmatically control UTToolBox. The included example file, “ExampleWizard.m” in the “C:\Program Files\AOS\OEMPA [version]\MATLAB” folder shows how to pass commands to the UTToolBox program for the purpose of creating a scan sequence, which can then be saved to an OEMPA file. The OEMPA file can then be loaded and managed using the APIs from sections 6 (Setup API) and 7 (Acquisition) or any other program based on the AOS SDK, such as OEMPATool (See *Software_Utilities.pdf*). The example contains comments to guide the user through the commands. The primary commands used are “mxGetToolboxParameter” and “mxSetToolboxParameter” to interact with UTToolbox. The file can be edited to create different types of scans. An OEM-PA device must be connected in order to save an OEMPA file by using the command “mxToolboxSaveSetup”.