Internet: www.pcultrasound.com
Information: info@pcultrasound.com
Support: support@pcultrasound.com

TELEMED Real-time imaging for the research

User Manual

Contents

Introduction

TELEMED releases a dynamic link library (DLL) which allows to call TELEMED SDK functions and to perform real-time ultrasound image data acquisition and imaging from the other programming platforms such as MATLAB, Python etc. This opens new possibilities for researchers developing image processing algorithms for biomedical ultrasound or combining ultrasonic imaging with other modalities.

- Receive real-time ultrasound images in platform which is convenient to you (MATLAB, Python, LabVIEW or other which is capable to call C++ DLL library which links to TELEMED SDK)
- Sample programs for MATLAB, Python and LabVIEW are provided
- Source codes of C++ wrapper DLL are provided
- Possibilities to implement majority scanning controls and to control ultrasound imaging parameters from the convenient platform
- Possibilities to combine or superimpose ultrasound images with images from another modality in real-time
- Possibilities for developments and evaluation of image processing techniques and algorithms working in real-time for:
    - Speckle reduction
    - Image enhancement
    - Image restoration
    - Segmentation of regions of interest
    - Parametric analysis of images (texture quantitative measurements)

1. DLL usgfw2wrapper.dll purpose and description

DLL usgfw2wrapper.dll is a C++ program which operates as a wrapper between TELEMED SDK written in C++ and other programming platforms which are capable to call DLLs, such as MATLAB, Python or LabView. The library allows to initialize ultrasound scanner and to acquire ultrasound image data in real-time. Currently the library contains following functions, which could be called from external software:

Function	Description	Input/Output
on_init()	Function initializes variables, objects and handles used in further calls.	None/None (Void)
init_ultrasound_usgfw2()	Function creates main Usgfw2 library object.	None/returns int32, value 3 in case of success for error handling
find_connected_probe()	Function detects connected ultrasound probe.	None/return int32, value 101 in case of success (if probe detected)
data_view_function()	Function creates main ultrasound scanning object for detected probe.	None/return int32, value 100 in case of success (if probe detected)
mixer_control_function(int32, int32, int32, int32, int32, int32, int32)	Function creates B mixer control. Image width and height are passed into a function by first 4 parameters.	Input parameters are: int32 – left coordinate of imaging rectangle (i.e. 0) int32 – top coordinate of imaging rectangle (i.e. 0) int32 – right coordinate of imaging rectangle (i.e. 512) int32 – bottom coordinate of imaging rectangle (i.e. 512) int32 – R int32 – G int32 – B – RGB values for image background, (0,0,0) - black Output Return int32, value 89 in case of success
get_resolution(singlePtr, singlePtr)	Function returns pixel size (resolution) in x and y dimensions in mm for imaging axis.	Input: Pointer to float value (resolution X) Pointer to float value (resolution Y)
return_pixel_values(uint32Ptr)	Function returns buffer with pixel values of current frame in BGRA order, for further imaging purposes array must be reshaped.	Input: Pointer to initialized uint32 array of size (w*h*4), where w – image width, h – image height
Run_ultrasound_scanning()	Function to run scanning	None/None (Void)
Freeze_ultrasound_scanning()	Function to freeze ultrasound scanning	None/None (Void)
Stop_ultrasound_scanning()	Function to stop ultrasound scanning	None/None (Void)

Close_and_release()	Function releases all initialized variables, objects, controls and variables.	None/None (Void)

Please note: the function ***return_pixel_values*** calls SDK function ***GetCurrentBitmap*** and that means that there is a possibility of missing or duplicate frames which were originally obtained during scanning if imaging speed in external calling program do not matches scanning speed of ultrasound scanner. We recommend to set pauses in imaging loop to match frame grabbing speed with scanning speed (it should not exceed frame rate shown in Echo Wave 2 software to avoid duplicate frames).

It is possible to control majority imaging parameters (such as imaging depth, scanning frequency, gain and others) running the main TELEMED Echo Wave 2 software in parallel to external program which receives real-time ultrasound images. The functions for controlling scanning parameters (controls) in your custom application could be implemented as well by modifying wrapper DLL code because we provide source codes of it (Source codes of the wrapper project could be found in the directory *..\usgfw2wrapper_C++_sources\usgfw2wrapper*). So, you can supplement existing functions listed in table above by your functions which could be called from external application. To familiarize with TELEMED SDK possibilities we invite you to read Ultrasonography for Windows II Software Development Kit (Usgfw2 SDK) Programmer's Guide.

2. MATLAB, Python and LabView samples

We provide sample programs for MATLAB, Python and LabVIEW. All the samples written for different languages works about the same. The workflow is to initialize ultrasound scanner, to acquire real-time ultrasound images and display them. MATLAB and LabVIEW samples contains minimalistic graphical user interfaces (GUI), which has buttons “Run” and “Freeze” and FPS scanning speed indicator. For Python we provide only the script which acts about the same. You must start the software for these samples by using Administrator rights to run them successfully. You can find the following files for sample programs in a package:

Filename	Description
usgfw2MATLAB.m	MATLAB script for calling DLL and displaying ultrasound images in-real time.
GUI_MATLAB_usgfw2.m GUI_MATLAB_usgfw2.fig	MATLAB GUI with “Run” and “Freeze” buttons for ultrasound imaging control.
usgfw2python.py	Python script for calling DLL and displaying ultrasound images in real-time.
usgfw2labview.vi	LabVIEW sample GUI for calling DLL and displaying ultrasound images in-real time with “Run” and “Freeze” buttons for ultrasound imaging control.

We will analyse the first MATLAB script in the User Manual, meanwhile others are the repetition of it just with specific functions for imaging of each programming language.

```
% image parameters (w - width, h - height)
w = 512; % image width in pixels
h = 512; % image height in pixels
path =
'D:\...\usgfw2matlab_sources\usgfw2MATLAB\x64\Release\usgfw2MATLAB_wrapper.dll';
% path to dll library which operates between MATLAB and usgfw2 SDK
hfile = 'D:\...\usgfw2matlab_sources\usgfw2MATLAB\usgfw2matlab.h'; % path to
header (*.h) file of library which operates between MATLAB and usgfw2 SDK

% Checking if library is loaded and loading if not
if not(libisloaded('USGFW2MATLABWRAPPER'))
   loadlibrary(path,hfile,'alias', 'USGFW2MATLABWRAPPER')
end

% List of available functions
libfunctions USGFW2MATLABWRAPPER -full

calllib('USGFW2MATLABWRAPPER','on_init') % function initializes variables,
objects and handles used in later calls

[ERR_CODE1] = calllib('USGFW2MATLABWRAPPER','init_ultrasound_usgfw2'); %
function creates main Usgfw2 library object

if (ERR_CODE1 == 2)
    errordlg('Main Usgfw2 library object not created');
```

```
    calllib('USGFW2MATLABWRAPPER','Close_and_release')% function releases all
initialized controls and variables if error occurs
    unloadlibrary USGFW2MATLABWRAPPER
    return;
end

[ERR_CODE2] = calllib('USGFW2MATLABWRAPPER','find_connected_probe'); % function
detects connected probe

if (ERR_CODE2 ~= 101)
    errordlg('Probe not detected');
    calllib('USGFW2MATLABWRAPPER','Close_and_release') % function releases all
initialized controls and variables if error occurs
    unloadlibrary USGFW2MATLABWRAPPER
    return;
end

[ERR_CODE3] = calllib('USGFW2MATLABWRAPPER','data_view_function'); % function
creates main ultrasound scanning object for selected probe
if (ERR_CODE3 < 0)
    errordlg('Main ultrasound scanning object for selected probe not created');
    calllib('USGFW2MATLABWRAPPER','Close_and_release') % function releases all
initialized controls and variables if error occurs
    unloadlibrary USGFW2MATLABWRAPPER
    return;
end

[ERR_CODE4] =
calllib('USGFW2MATLABWRAPPER','mixer_control_function',0,0,w,h,0,0,0); %
function creates B mixer control, passed parameters
% are image width and height, then RGB values for background (0,0,0)-black
if (ERR_CODE4 < 0)
    errordlg('B mixer control not returned');
    calllib('USGFW2MATLABWRAPPER','Close_and_release') % function releases all
initialized controls and variables if error occurs
    unloadlibrary USGFW2MATLABWRAPPER
    return;
end

%% Get pixel size (resolution) and make the axis of ultrasound image in
%% metric units
res_X = libpointer('singlePtr', zeros(1,1));
res_Y = libpointer('singlePtr', zeros(1,1));
calllib('USGFW2MATLABWRAPPER','get_resolution',res_X,res_Y) % function returns
pixel size in x and y dimensions in mm

%% Calculations of X and Y axis for image
if (mod(w,2) == 0)
X_axis = (-w/2+0.5:w/2-0.5).*res_X.Value;
else
X_axis = (-w/2:w/2).*res_X.Value;
end
Y_axis = (0:h-1).*res_Y.Value;

old_resolution_x = res_X.Value;
old_resolution_y = res_X.Value;
```

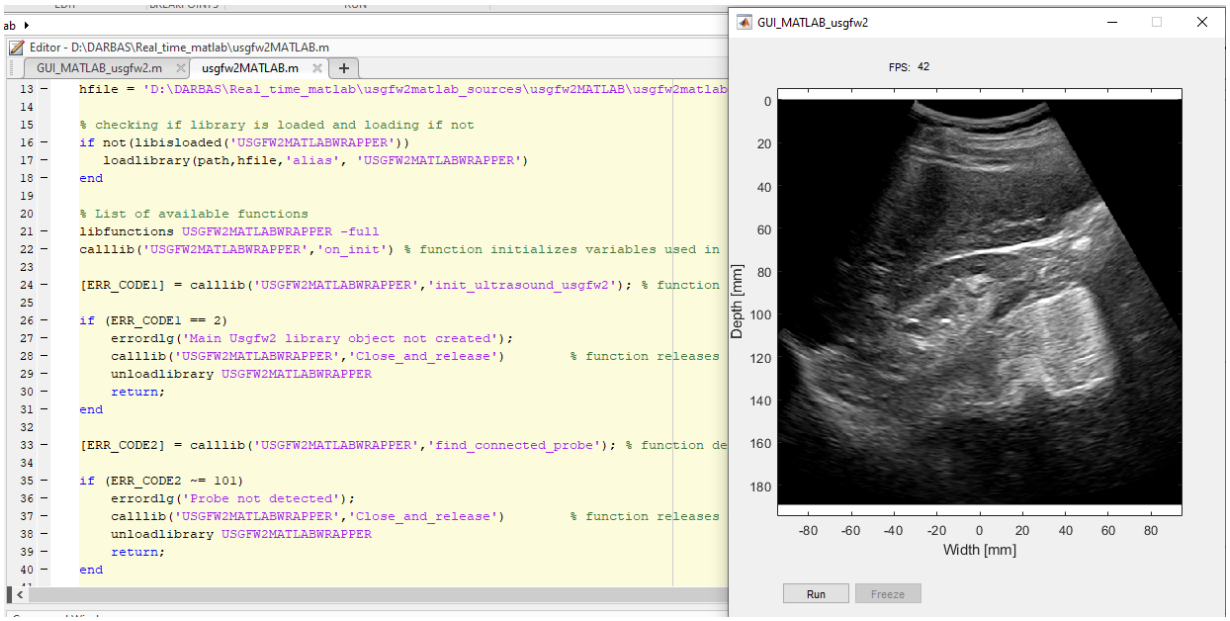
```
iteration = 0;
run_loop = true;
threshold = 1000; %% number of iterations to break imaging loop (currently there are two ways how to stop ultrasound scanning in this script, by setting number of iterations in other words frames to be displayed and manually by Ctrl+C, in case of manual termination you have to call:
% calllib('USGFW2MATLABWRAPPER','Stop_ultrasound_scanning') % function stops ultrasound scanning
% calllib('USGFW2MATLABWRAPPER','Close_and_release')        % function releases all initialized controls and variables
% unloadlibrary USGFW2MATLABWRAPPER                         % DLL library unload after, to realize all initialized variables, otherwise you will receive MATLAB crash on re-run;)

%% initializes pointer to uint32 array of w*h*4 size, 4 - because its BGRA format with alpha channel
p = libpointer('uint32Ptr', zeros(1,w*h*4));
calllib('USGFW2MATLABWRAPPER','return_pixel_values',p) % % function returns buffer with pixel values of current frame in BGRA format
Blue_component = p.Value(1:4:end); % extraction of one of the components (red in example)
img_gsc = reshape(Blue_component',[w h]); % 1D buffer reshaping into image matrix

figure(1)
update_figure = image(X_axis,Y_axis,img_gsc(:,end:-1:1)')
colormap gray
caxis([0 255])
xlabel('Width [mm]')
ylabel('Depth [mm]')
axis equal

while (run_loop)
tic

calllib('USGFW2MATLABWRAPPER','get_resolution',res_X,res_Y)

% % If resolution changes (i.e. depth control click in EW2) calculates new axes X, Y
if (res_X.Value~=old_resolution_x || res_Y.Value~=old_resolution_y)
if (mod(w,2) == 0)
X_axis = (-w/2+0.5:w/2-0.5).*res_X.Value;
else
X_axis = (-w/2:w/2).*res_X.Value;
end
Y_axis = (0:h-1).*res_Y.Value;
old_resolution_x = res_X.Value;
old_resolution_y = res_X.Value;
end

calllib('USGFW2MATLABWRAPPER','return_pixel_values',p) % % function returns buffer with pixel values of current frame in RGBA format
Blue_component = p.Value(1:4:end); % extraction of one of the components (blue in example)
img_gsc = reshape(Blue_component',[w h]); % 1D buffer reshaping into gray scale image matrix

figure(1)
```

```
set(update_figure, 'XData',X_axis)
set(update_figure, 'YData',Y_axis)
set(update_figure, 'CData',img_gsc(:,end:-1:1)') % Note for the correct imaging
the image must be rotates 180 degrees: end:-1:1 implements that
axis equal

time = toc;
FPS = int32(1/time) % Frames per second counter

iteration = iteration + 1; % iterations to stop imaging loop
if (iteration > threshold)
   run_loop = false;
   calllib('USGFW2MATLABWRAPPER','Freeze_ultrasound_scanning') %% function to
freeze scanning
   calllib('USGFW2MATLABWRAPPER','Stop_ultrasound_scanning') % function stops
ultrasound scanning
   calllib('USGFW2MATLABWRAPPER','Close_and_release')        % function releases
all initialized controls and variables
   unloadlibrary USGFW2MATLABWRAPPER                         % DLL library
unload
   return;
end
end
```

Please note: currently wrapper DLL was compiled for x64 bit version, so TELEMED SDK x64 must be installed as well. For successfully running MATLAB samples Microsoft Windows SDK 7 must be installed (x64 version), Download Microsoft Windows SDK for Windows 7 and .NET Framework 4 from Official Microsoft Download Center, because some header files are needed for running MATLAB samples. You must have an installation in the following path "C:\Program Files\Microsoft SDKs\Windows\v7.1\Include", or if the installation path slightly differs you have to recompile the usgfw2wrapper project including the DIR to your Microsoft Windows SDK (x64). MATLAB scripts were prepared by using 2020a version (for older versions it might be necessary to install MinGW-w64 C/C++ compiler).

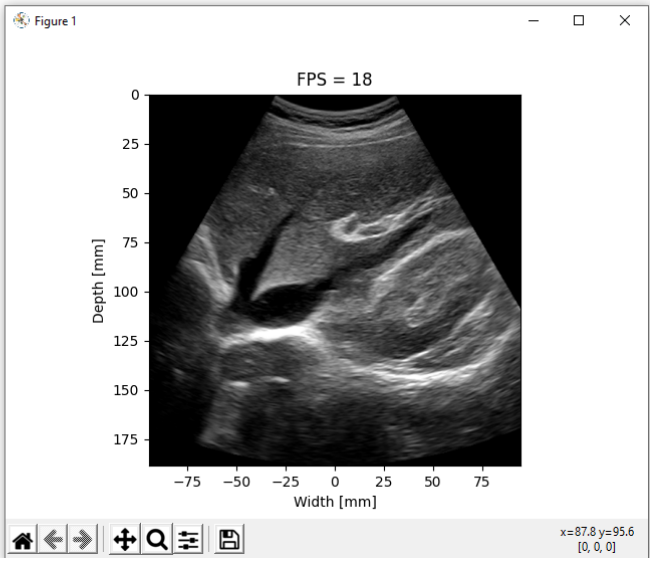
2.1 Figures for all samples

MATLAB sample GUI:

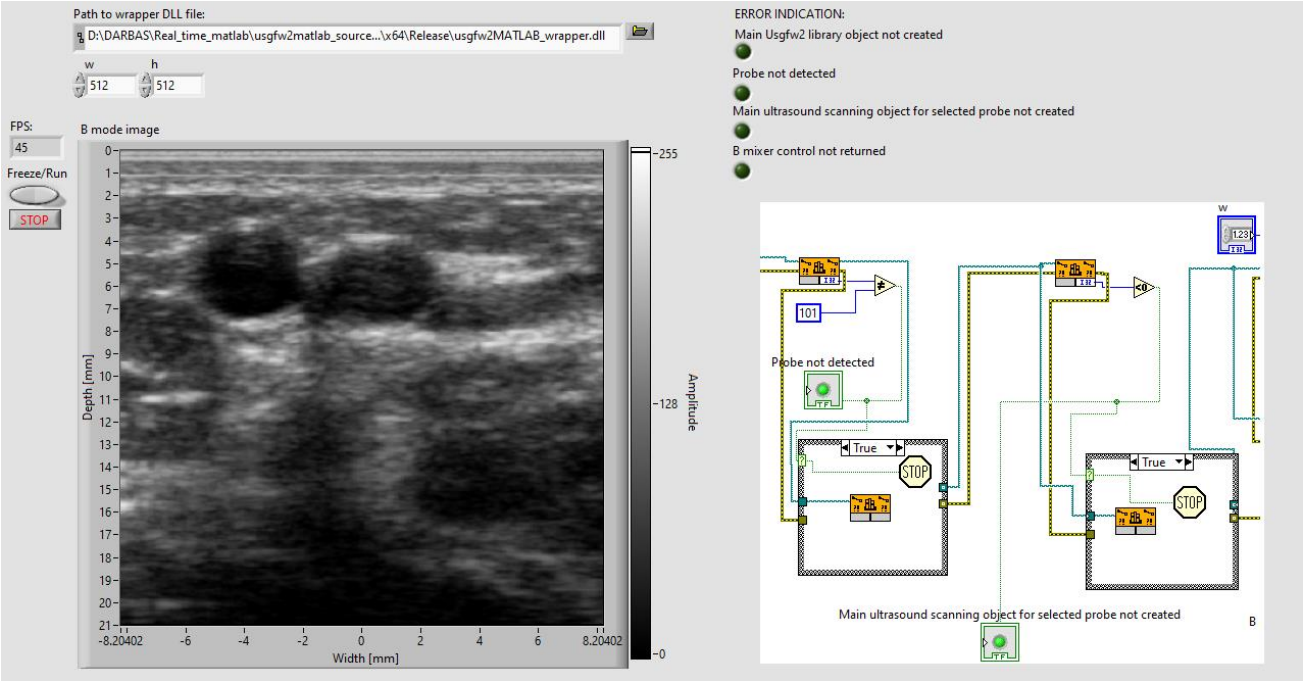


Python sample script:

```
if (ERR == 2):
     print('Main Usgfw2 library object not created');
     usgfw2.Close_and_release()
     sys.exit()

ERR = usgfw2.find_connected_probe()

if (ERR != 101):
     print('Probe not detected')
     usgfw2.Close_and_release()
     sys.exit()

ERR = usgfw2.data_view_function()

if (ERR < 0):
    print('Main ultrasound scanning object for selected probe not created')
    sys.exit()


ERR = usgfw2.mixer_control_function(0,0,w,h,0,0,0)
if (ERR < 0):
    print('B mixer control not returned');
    sys.exit()

res_X = ctypes.c_float(0.0)
res_Y = ctypes.c_float(0.0)
usgfw2.get_resolution(ctypes.pointer(res_X),ctypes.pointer(res_Y))


X_axis = np.zeros(shape=(w));
Y_axis = np.zeros(shape=(h));
if (w % 2 == 0):

   k = 0;
   for i in range (-w//2, w//2+1):
      if (i<0):
          j=i+0.5
```



LabVIEW sample GUI:



Revision History

Version	Date	Description of Revision	Revision author
1.0.0	18/05/2022	Initial Release	A. Sakalauskas