---
* Introduction

"AutoInt1.dll" plug-in (32-bit) implements COM automation programming interface (ProgId "EchoWave2.CmdInt1") that allows to access "Echo Wave II" ultrasound images from other applications and adjust "Echo Wave II" ultrasound scanning parameters.
In order to use this interface, in "Echo Wave II" file "...\Config\Plugins\list.txt" must be present the following line:
AutoInt1.dll
It means that during software startup will be loaded plug-in dll.
In order to use automation interface, "Echo Wave II" software must be running.
"Echo Wave II" software is run using Admnistrator rights. This means that another application that will be connecting to "Echo Wave II" also must be run As Administrator.

"AutoInt1Client.dll" is an "Echo Wave II" automation interface client .Net assembly that can be used from 32/64-bit applications that support .Net.
"AutoInt1Client.cs" - C# source code of "AutoInt1Client.dll" that can be used as a reference for implementing automation interface clients in other programming languages.
"AutoInt1Client_test.m" - MATLAB (R2010b or more recent version) script that demonstrates how to use .Net client dll and access "Echo Wave II" ultrasound images. In order to test the script, please remove/add comments (%) from/to desired script lines.

Programming interface allows to do the following things: freeze/run or record/stop ultrasound (when is connected ultrasound scanner), open TVD or other file that is supported by "Echo Wave II", get the number of cine frames, select desired frame, get frame time in milliseconds (the time of first frame is 0.0), get frame in grayscale or RGB format, change ultrasound scanning mode, set ultrasound scanning parameters.

IMPORTANT. If some command opens "Echo Wave II" modal dialog (window) like save, open, print, patient window, options window or any message box, this means that called software will be blocked until the user will close that dialog in "Echo Wave II" software. The same applies to all other commands.
If you want to send commands to "Echo Wave II" and do not block user interface of your software, then connection to "Echo Wave II" must be done and commands must be sent from a single separate thread.

---
* AutoInt1Client.dll functions

Namespace: AutoInt1Client
Class: CmdInt1
In order to use below listed functions, must be created CmdInt1 object.
Since these functions call appropriate "Echo Wave II" functions, their operation depends on what is enabled in "Echo Wave II" user interface.

---
int ConnectToRunningProgram()

Connects to "Echo Wave II" automation interface with ProgId "EchoWave2.CmdInt1".
If connection fails, this function returns "-1". In other case it returns "0".
Connection is possible only if "Echo Wave II" is running and both connected
applications were run "As Administrator".

---
int OpenFile(string fn)

Open file by "Echo Wave II".
fn - full path to file, for example "C:\\Echo Images\\test1.tvd"
In case of unexpected errors this function returns "-1".

---
int GetFramesCount()

Get the number of frames in opened file or scanned cine.
In case of unexpected errors this function returns "-1".

---
int GoToFrame1n(int frm_idx_1n, bool load_frame_data)

Go to frame with index frm_idx_1n. Indexing starts from "1".
load_frame_data must be true if we will want to get image (bytes) of selected
frame using functions GetLoadedFrame*.
Before using this function, must be stopped cine playback, stopped recording,
stopped ultrasound scanning.
If, although not recommended, this function will be used without stopping
ultrasound playback/recording/scanning in order to load current frame, then
parameter frm_idx_1n must be "-1".
In case of unexpected errors this function returns "-1".

---
int GetCurrentFrameIdx1n()

Get index of current (visible in "Echo Wave II") frame.
In case of unexpected errors this function returns "-1".

---
double GetCurrentFrameTime()

Get time (in milliseconds) of current frame.
The time of first frame is 0.0.
In case of unexpected errors this function returns "-1".

---
int GetLoadedFrameWidth()

Get loaded frame width in pixels.
In case of unexpected errors this function returns "-1".

---
int GetLoadedFrameHeight()

Get loaded frame height in pixels.
In case of unexpected errors this function returns "-1".

```
---------------------------------------------------------
int[,,] GetLoadedFrameRGB()
---------------------------------------------------------
Get three-dimensional array of loaded frame pixel data.
Returned array contains red (R), green (G), blue(B) planes of image pixels.
R, G, B values are in the interval [0;255].
In case of unexpected errors this function returns null.

---------------------------------------------------------
int[,] GetLoadedFrameGray()
---------------------------------------------------------
Get two-dimensional array of loaded frame grayscale pixel data.
Returned grayscale ((R+G+B)/3) values are in the interval [0;255].
In case of unexpected errors this function returns null.

---------------------------------------------------------
int[,] GetLoadedFrameRed()
---------------------------------------------------------
Get two-dimensional array of Red image (24-bit RGB) plane.
Returned values are in the interval [0;255].
In case of unexpected errors this function returns null.

---------------------------------------------------------
int[,] GetLoadedFrameGreen()
---------------------------------------------------------
Get two-dimensional array of Green image (24-bit RGB) plane.
Returned values are in the interval [0;255].
In case of unexpected errors this function returns null.

---------------------------------------------------------
int[,] GetLoadedFrameBlue()
---------------------------------------------------------
Get two-dimensional array of Blue image (24-bit RGB) plane.
Returned values are in the interval [0;255].
In case of unexpected errors this function returns null.

---------------------------------------------------------
int GetUltrasoundX1(int img_id)
---------------------------------------------------------
Get X1 coordinate in pixels of ultrasound image region with passed img_id.
Here img_id: 1-B; 2-B2; 3-B3; 4-B4; 7-M; 8-PW; 9-CW.
In case of unexpected errors this function returns "-1".

---------------------------------------------------------
int GetUltrasoundX2(int img_id)
---------------------------------------------------------
Get X2 coordinate in pixels of ultrasound image region with passed img_id.
Here img_id: 1-B; 2-B2; 3-B3; 4-B4; 7-M; 8-PW; 9-CW.
In case of unexpected errors this function returns "-1".

---------------------------------------------------------
int GetUltrasoundY1(int img_id)
---------------------------------------------------------
Get Y1 coordinate in pixels of ultrasound image region with passed img_id.
Here img_id: 1-B; 2-B2; 3-B3; 4-B4; 7-M; 8-PW; 9-CW.
In case of unexpected errors this function returns "-1".
```

---
int GetUltrasoundY2(int img_id)

Get Y2 coordinate in pixels of ultrasound image region with passed img_id.
Here img_id: 1-B; 2-B2; 3-B3; 4-B4; 7-M; 8-PW; 9-CW.
In case of unexpected errors this function returns "-1".

---
double GetUltrasoundPhysicalDeltaX(int img_id)

Get horizontal resolution of ultrasound image region with passed img_id.
Here img_id: 1-B; 2-B2; 3-B3; 4-B4; 7-M; 8-PW; 9-CW.
Returned value is in cm/pt (centimeters per pixel) or s/pt (seconds per pixel)
depending on image type.
In case of unexpected errors this function returns "0".

---
double GetUltrasoundPhysicalDeltaY(int img_id)

Get vertical resolution of ultrasound image region with passed img_id.
Here img_id: 1-B; 2-B2; 3-B3; 4-B4; 7-M; 8-PW; 9-CW.
Returned value is in cm/pt or cm/s/pt depending on image type.
In case of unexpected errors this function returns "0".

---
int FreezeRun()

Freeze/Run ultrasound scanning.
In case of unexpected errors this function returns "-1".

---
int IsRunState()

Returns "1" if ultrasound scanning is in progress.
In case of unexpected errors this function returns "-1".

---
int PlayPause()

Play/Pause scanned or opened cine.
This function works when is enabled "Echo Wave II" Play/Pause button.
In case of unexpected errors this function returns "-1".

---
int IsPlayState()

Returns "1" if cine playback is in progress.
In case of unexpected errors this function returns "-1".

---
int RecordStop()

Record/Stop ultrasound cine (when ultrasound scanner is connected).
When recording is started, software clears cine buffer.
When recording is stopped, software freezes ultrasound scanning.

This function works only when is enabled "Echo Wave II" Record/Stop button.
That is, at first ultrasound scanning must be run using FreezeRun(), then found
anatomy of interest, then called RecordStop() in order to clear cine and do
recording and then called another RecordStop() in order to stop recording.
In case of unexpected errors this function returns "-1".

---
int IsRecordingState()

Returns "1" if cine recording is in progress.
In case of unexpected errors this function returns "-1".

---
int WmCopyDataCmd(string cmd_str)

Invoke WM_COPYDATA command cmd_str. For the list of supported WM_COPYDATA
commands please check "Echo Wave II" user manual section "Software control
using command line".
In case of unexpected errors this function returns "-1".

---
int ParamSet(int param_id, int val)

Shift parameter value by specified index val or set parameter
value val depending on passed param_id. The list of defines of param_id
is presented below (#define id_...).
If we have B Gain values 10%, 20%, 30%, 40%, 50%, current value is 40%,
then param_id=id_b_gain_shift and val=-1 will set Gain value to 30%.
If some command does not need parameters, then should be passed val=0.
In case of unexpected errors this function returns "-1".

For cases when it is not enough to pass single int value or when
defined command returns some value, are used special functions that
will be mentioned near each define.
These functions can be used only in cases where for concrete param_id is
documented
that must be used such functions. In other cases you will get either exceptions
or
crashes because of attempt to pass/return incorrect data type(s).
int ParamSet2(int param_id, int val, int val2)
int ParamSetString(int param_id, string val)
object ParamGet(int param_id)
int ParamGetInt(int param_id)
int ParamGetInt2(int param_id, int val)
bool ParamGetBool(int param_id)
bool ParamGetBool2(int param_id, int val)
double ParamGetDouble(int param_id)
float ParamGetFloat(int param_id)
string ParamGetString(int param_id)
string ParamGetString2(int param_id, int val)
string ParamGetString3(int param_id, int val, val2)

Below is the list of supported param_id defines and their description:

#define id_get_current_beamformer_code 915   // GET current beamformer code
as int. ParamGetInt(id_get_current_beamformer_code)

```
#define id_get_current_beamformer_name  916     // GET current beamformer name
as string. ParamGetString(id_get_current_beamformer_name)
#define id_get_current_probe_code       917     // GET current probe code as
int. ParamGetInt(id_get_current_probe_code)
#define id_get_current_probe_name       918     // GET current probe name as
string. ParamGetString(id_get_current_probe_name)
#define id_probe                104     // Probe button; val = 0;
#define id_is_no_probe          223     // Get if the probe is not detected.
ParamGetBool(id_is_no_probe)
#define id_is_unsupported_probe 224     // Get if is connected unsupported
probe. ParamGetBool(id_is_unsupported_probe)
#define id_is_probe_active      103     // Get if the probe is active (the probe
becomes inactive when is opened tpd/tvd file). ParamGetBool(id_is_probe_active)
#define id_get_probes_count     225     // Get the number of connected probes.
ParamGetInt(id_get_probes_count)
#define id_freeze_run           100     // Freeze/Run command; val = 0;
#define id_is_freeze            101     // GET if ultrasound scanning is frozen.
ParamGetBool(id_is_freeze)
#define id_cine_play_pause      119     // Cine play/pause; val = 0;
#define id_cine_pause           141     // Cine pause button; val = 0;
#define id_cine_record_stop     142     // Cine record/stop button; The image
must be unfrozen. First invocation clears cine buffers, second invocation does
freeze. val = 0;
#define id_cine_frame_shift     120     // Cine previous or next frame
#define id_is_cine_playing              122     // GET from software if cine is
playing. ParamGetBool(id_is_cine_playing)
#define id_is_cine_record_in_progress   143     // Get if cine is in recording
mode (not simple scanning, but recording).
ParamGetBool(id_is_cine_record_in_progress)
#define id_is_usg_file_opened           191     // Returns if we have opened
tpd/tvd file. ParamGetBool(id_is_usg_file_opened)
#define id_auto_adjust          116     // Automatic image adjustment (Auto
button); val = 0;

// Scanning modes
#define id_button_b                     174     // B button's command; val = 0;
#define id_button_dual_quad             168     // Dual/Quad button's last used
command; val = 0;
#define id_button_dual                  898     // B+B (Dual) button; val = 0;
#define id_button_quad                  899     // 4B (Quad) button; val = 0;
#define id_button_m                     900     // B+M button; val = 0;
#define id_button_cd                    901     // CD button's last used
command; val = 0;
#define id_button_pw                    625     // PW button (mode); val = 0;
#define id_button_cw                    662     // CW button (mode); val = 0;
#define id_button_pw_cw                 897     // PW/CW button's last used
command; val = 0;
#define id_button_update                902     // "Update" (F4) button; val =
0;
#define id_use_duplex_triplex_shift     871     // Option "Use Duplex/Triplex
modes"; -1 - off; 1 - on;
#define id_use_duplex_triplex           895     // Option "Use Duplex/Triplex
modes"; toggle; val = 0;
#define id_is_triplex_supported         227     // GET if is supported Triplex
(simultaneous running B+CFM+PW). ParamGetBool(id_is_triplex_supported)
#define id_get_use_duplex_triplex       228     // GET the value of option "Use
```

```
Duplex/Triplex modes". ParamGetBool(id_get_use_duplex_triplex)
#define id_is_color_m_supported         247     // GET if is supported Color M
mode. ParamGetBool(id_is_color_m_supported)

// Set scanning state. This command can be used for cases that are not available
as
// commands of scanning mode buttons.
// As parameter is passed scanning state id (e.g., id_state_b_b1r).
// In many cases the sequence of states is important, so before programming it
is recommended
// to check in "Echo Wave II" how "Echo Wave II" buttons change scanning states.
// If direct change from one state to another is unsupported by "Echo Wave II",
new state
// will be set by at first entering id_state_b_b1r.
// In state identifier names "r" means running, "f" means frozen.
#define id_scanning_state_set   201     // SET scanning state.
ParamSet(id_scanning_state_set, id_state_b_b1r)
#define id_scanning_state_get   200     // GET current scanning state.
ParamGetInt(id_kb_cmd_scanning_state_get)

// B mode states
#define id_state_b_b1r          1
#define id_state_b_b1f          2

// B+M, M states
#define id_state_bm_b1mr        3
#define id_state_bm_b1mf        4
#define id_state_bm_m1r         5
#define id_state_bm_m1f         6
#define id_state_bm_b1mr_m1r    7
#define id_state_bm_b1mf_m1f    8

// Dual states (B+B states)
#define id_state_bb_b1r_b2f     11
#define id_state_bb_b1f_b2r     13
#define id_state_bb_b1f         14
#define id_state_bb_b2f         15
#define id_state_bb_b1f_b2f_sel_b1      22
#define id_state_bb_b1f_b2f_sel_b2      23
#define id_state_bb_b1f_b2f     26
#define id_state_bb_b1r_b2r     27

// Color Doppler states (CFM, PDI, DPDI)
#define id_state_cfm_cfm1r      16
#define id_state_cfm_cfm1f      17
#define id_state_pdi_pdi1r      18
#define id_state_pdi_pdi1f      19
#define id_state_dpdi_dpdi1r    20
#define id_state_dpdi_dpdi1f    21

// Quad states (4B states)
#define id_state_4b_b1r_b2f_b3f_b4f             30
#define id_state_4b_b1f_b2r_b3f_b4f             31
#define id_state_4b_b1f_b2f_b3r_b4f             32
#define id_state_4b_b1f_b2f_b3f_b4r             33
#define id_state_4b_b1f_b2f_b3f_b4f_sel_b1      34
```

```
#define id_state_4b_b1f_b2f_b3f_b4f_sel_b2      35
#define id_state_4b_b1f_b2f_b3f_b4f_sel_b3      36
#define id_state_4b_b1f_b2f_b3f_b4f_sel_b4      37

// Duplex states (B+PW)
#define id_state_bpw_b1pwr                     40      // B+(PW line) running
#define id_state_bpw_b1pwf                     41
#define id_state_bpw_pw1r                      42      // PW running
#define id_state_bpw_pw1f                      43
#define id_state_bpw_b1pwr_pw1r                44      // B+PW runing
#define id_state_bpw_b1pwf_pw1f                45

// Triplex states (CFM+PW)
#define id_state_cfmpw_cfm1pwr                 46
#define id_state_cfmpw_cfm1pwf                 47
#define id_state_cfmpw_pw1r                    48
#define id_state_cfmpw_pw1f                    49
#define id_state_cfmpw_cfm1pwr_pw1r            50
#define id_state_cfmpw_cfm1pwf_pw1f            51

// Triplex states (PDI+PW)
#define id_state_pdipw_pdi1pwr                 52
#define id_state_pdipw_pdi1pwf                 53
#define id_state_pdipw_pw1r                    54
#define id_state_pdipw_pw1f                    55
#define id_state_pdipw_pdi1pwr_pw1r            56
#define id_state_pdipw_pdi1pwf_pw1f            57

// Triplex states (DPDI+PW)
#define id_state_dpdipw_dpdi1pwr               58
#define id_state_dpdipw_dpdi1pwf               59
#define id_state_dpdipw_pw1r                   60
#define id_state_dpdipw_pw1f                   61
#define id_state_dpdipw_dpdi1pwr_pw1r          62
#define id_state_dpdipw_dpdi1pwf_pw1f          63

// Duplex Update states (B+PW)
#define id_state_bpw_b1pwr_pw1f         64  // B running + PW frozen
#define id_state_bpw_b1pwf_pw1r         65  // B frozen + PW running
#define id_state_bpw_b1pwf_pw1f_sel_b   66  // B frozen + PW frozen and selected
is B
#define id_state_bpw_b1pwf_pw1f_sel_pw  67  // B frozen + PW frozen and selected
is PW

// "Triplex" Update states (CFM+PW)
#define id_state_cfmpw_cfm1pwr_pw1f            68  // CFM running + PW frozen
#define id_state_cfmpw_cfm1pwf_pw1r            69  // CFM frozen + PW running
#define id_state_cfmpw_cfm1pwf_pw1f_sel_cfm    70  // CFM frozen + PW frozen
and selected is CFM
#define id_state_cfmpw_cfm1pwf_pw1f_sel_pw     71  // CFM frozen + PW frozen
and selected is PW

// "Triplex" Update states (PDI+PW)
#define id_state_pdipw_pdi1pwr_pw1f            72  // PDI running + PW frozen
#define id_state_pdipw_pdi1pwf_pw1r            73  // PDI frozen + PW running
#define id_state_pdipw_pdi1pwf_pw1f_sel_pdi    74  // PDI frozen + PW frozen
```

```
and selected is PDI
#define id_state_pdipw_pdi1pwf_pw1f_sel_pw      75  // PDI frozen + PW frozen
and selected is PW

// "Triplex" Update states (DPDI+PW)
#define id_state_dpdipw_dpdi1pwr_pw1f           76  // DPDI running + PW frozen
#define id_state_dpdipw_dpdi1pwf_pw1r           77  // DPDI frozen + PW running
#define id_state_dpdipw_dpdi1pwf_pw1f_sel_dpdi  78  // DPDI frozen + PW frozen
and selected is DPDI
#define id_state_dpdipw_dpdi1pwf_pw1f_sel_pw    79  // DPDI frozen + PW frozen
and selected is PW

// B+CW (Continuous Wave Doppler)
#define id_state_bcw_b1cwr                      97  // B+(CW line) running
#define id_state_bcw_b1cwf                      98
#define id_state_bcw_cw1r                       99  // CW running
#define id_state_bcw_cw1f                       100
#define id_state_bcw_b1cwr_cw1f                 103 // B running + CW frozen
#define id_state_bcw_b1cwf_cw1r                 104 // B frozen + CW running
#define id_state_bcw_b1cwf_cw1f_sel_b           105 // B frozen + CW frozen and
selected is B
#define id_state_bcw_b1cwf_cw1f_sel_cw          106 // B frozen + CW frozen and
selected is CW

// CFM+CW
#define id_state_cfmcw_cfm1cwr                  107
#define id_state_cfmcw_cfm1cwf                  108
#define id_state_cfmcw_cw1r                     109
#define id_state_cfmcw_cw1f                     110
#define id_state_cfmcw_cfm1cwr_cw1f             113 // CFM running + CW frozen
#define id_state_cfmcw_cfm1cwf_cw1r             114 // CFM frozen + CW running
#define id_state_cfmcw_cfm1cwf_cw1f_sel_cfm     115 // CFM frozen + CW frozen
and selected is CFM
#define id_state_cfmcw_cfm1cwf_cw1f_sel_cw      116 // CFM frozen + CW frozen
and selected is CW

// PDI+CW
#define id_state_pdicw_pdi1cwr                  117
#define id_state_pdicw_pdi1cwf                  118
#define id_state_pdicw_cw1r                     119
#define id_state_pdicw_cw1f                     120
#define id_state_pdicw_pdi1cwr_cw1f             123 // PDI running + CW frozen
#define id_state_pdicw_pdi1cwf_cw1r             124 // PDI frozen + CW running
#define id_state_pdicw_pdi1cwf_cw1f_sel_pdi     125 // PDI frozen + CW frozen
and selected is PDI
#define id_state_pdicw_pdi1cwf_cw1f_sel_cw      126 // PDI frozen + CW frozen
and selected is CW

// DPDI+CW
#define id_state_dpdicw_dpdi1cwr                127
#define id_state_dpdicw_dpdi1cwf                128
#define id_state_dpdicw_cw1r                    129
#define id_state_dpdicw_cw1f                    130
#define id_state_dpdicw_dpdi1cwr_cw1f           133 // DPDI running + CW frozen
#define id_state_dpdicw_dpdi1cwf_cw1r           134 // DPDI frozen + CW running
#define id_state_dpdicw_dpdi1cwf_cw1f_sel_dpdi  135 // DPDI frozen + CW frozen
```

```
and selected is DPDI
#define id_state_dpdicw_dpdi1cwf_cw1f_sel_cw    136 // DPDI frozen + CW frozen
and selected is CW

// B+CM, CM states  (Color M; M Color Flow Mode)
#define id_state_bcm_cfm1mr             137     // M line on CFM image
#define id_state_bcm_cfm1mf             138
#define id_state_bcm_cm1r               139     // pure Color M
#define id_state_bcm_cm1f               140
#define id_state_bcm_b1mr_cm1r          141     // B + Color M
#define id_state_bcm_b1mf_cm1f          142

// B mode controls
#define id_b_frequency_shift                    300     // B Frequency
#define id_b_frequency_shift_loop               320     // B Frequency; go to
first when is reached last value
#define id_b_change_thi                         178     // B Switch THI on/off;
val = 0;
#define id_b_is_thi_frequency                   177     // GET if current
frequency is THI. ParamGetBool(id_b_is_thi_frequency)
#define id_b_focus_shift                        302     // B Focus depth
#define id_b_focus_shift_loop                   321     // B Focus depth; go to
first when is reached last value
#define id_b_focuses_number_shift               334     // B Focuses Number
#define id_b_focus_set_shift                    335     // B Focus Set
#define id_b_dynamic_focus                      304     // B Dynamic Focus
on/off; val = 0;
#define id_b_dynamic_focus_shift                323     // B Dynamic Focus; pass
negative or positive value in order to turn on or off
#define id_b_is_dynamic_focus                   171     // GET if is turned on
Dynamic Focus. ParamGetBool(id_b_is_dynamic_focus)
#define id_b_depth_shift                        305     // B Depth
#define id_b_depth_shift_asynch                 306     // B Depth asynchronous
shift
#define id_b_power_shift                        307     // B Power
#define id_b_gain_shift                         309     // B Gain
#define id_b_dynamic_range_shift                311     // B Dynamic Range
#define id_b_palette_shift                      338     // B Palette (B Color
Map)
#define id_b_palette_gamma_shift                313     // B Palette Gamma
#define id_b_palette_brightness_shift           315     // B Palette Brightness
#define id_b_palette_contrast_shift             317     // B Palette Contrast
#define id_b_palette_negative                   319     // B Palette Negative;
val = 0;
#define id_b_dynamic_range_shift_loop           322     // B Dynamic Range; go
to first when is reached last value
#define id_b_view_area_shift                    324     // B View Area
#define id_b_view_area_shift_loop               325     // B View Area; go to
first when is reached last value
#define id_b_frame_averaging_shift              326     // B Frame Averaging
#define id_b_rejection_shift                    327     // B Rejection
#define id_b_image_enhancement_enabled_shift    328     // B Image Enhancement
Enabled; negative value - turn off; positive - turn on
#define id_b_image_enhancement_method_shift     329     // B Image Enhancement
Method
#define id_b_image_enhancement_method_shift2    336     // B Image Enhancement;
```

```
Off, 1, 2, ...
#define id_b_speckle_reduction_enabled_shift    330     // B Speckle Reduction
Enabled; negative value - turn off; positive - turn on
#define id_b_speckle_reduction_level_shift      331     // B Speckle Reduction
Level
#define id_b_speckle_reduction_level_shift2     337     // B Speckle Reduction:
Off, 1, 2, ...
#define id_b_lines_density_shift                332     // B Lines Density
#define id_b_change_scan_direction              106     // B Change scan
direction; val = 0;
#define id_b_is_scan_direction_changed          133     // GET bool scan
direction value of current image. ParamGetBool(id_b_is_scan_direction_changed)
#define id_b_flip_up_down                       105     // B Flip image up/down;
val = 0;
#define id_b_rotate_shift                       333     // B Rotate
#define id_b_rotate_get                         132     // GET int rotation
value of current image. ParamGetInt(id_b_rotate_get)
#define id_b_steering_trapezoid_angle_shift     339     // B Angle
(Steering/Trapezoid)
#define id_b_steering_trapezoid_angle_shift_loop        341     // B Angle
(Steering/Trapezoid); go to first when is reached last value
#define id_b_scan_type_shift                    340     // B Scan Type:
Standard, Trapezoid, Compound
#define id_b_scan_type_shift_loop               344     // B Scan Type:
Standard, Trapezoid, Compound; go to first when is reached last value
#define id_b_compound_on_off                    342     // B Compound on/off;
val = 0;
#define id_zoom_controls                        110     // Show Zoom ControlBar;
val = 0;
#define id_b_zoom_default                       111     // B Set zoom 1:1
(default); val = 0;
#define id_b_zoom_factor_shift                  112     // B Zoom
#define id_b_zoom_factor_shift_asynch           113     // B Zoom asynchronous
#define id_b_tgc_shift                          114     // B TGC; As parameter
is passed the number (index) of TGC control [0;4] and shift of value.
ParamSet2(id_b_tgc_shift, 0, 1);
#define id_b_tgc_get_min                        179     // GET minimal possible
TGC value. ParamGetInt(id_b_tgc_get_min)
#define id_b_tgc_get_max                        180     // GET maximal possible
TGC value. ParamGetInt(id_b_tgc_get_max)
#define id_b_tgc_set                            181     // Set TGC value. As
parameter is passed the number (index) of TGC control [0;4] and new value.
ParamSet2(id_b_tgc_set, 0, 10);

// M mode controls
#define id_m_line_position_shift                400     // M Line Position
#define id_m_line_position_shift_asynch         401     // M Line Position
asynchronous shift
#define id_m_sweep_speed_shift                  402     // M Sweep Speed
#define id_m_palette_gamma_shift                404     // M Palette Gamma
#define id_m_palette_brightness_shift           406     // M Palette Brightness
#define id_m_palette_contrast_shift             408     // M Palette Contrast
#define id_m_palette_negative                   410     // M Palette Negative;
val = 0;
#define id_m_palette_negative_shift             411     // M Palette Negative;
-1 - off, +1 - on
```

```
#define id_m_rejection_shift                    412     // M Rejection
#define id_m_zoom_default                       413     // M Zoom 1:1 (default);
val = 0;
#define id_m_zoom_factor_shift                  414     // M Zoom factor
#define id_m_zoom_position_shift                415     // M Zoom vertical shift

// Color Doppler (CD) mode controls (CFM, PDI, DPDI)
#define id_cd_frequency_shift                   500     // CD Frequency
#define id_cd_frequency_shift_loop              501     // CD Frequency; go to
first when is reached last value
#define id_cd_steering_angle_shift              502     // CD Steering Angle (CD
Angle)
#define id_cd_steering_angle_shift_loop         536     // CD Steering Angle (CD
Angle); go to first when is reached last value
#define id_cd_prf_shift                         504     // CD PRF
#define id_cd_power_shift                       506     // CD Power
#define id_cd_gain_shift                        508     // CD Gain (changes CFM
or PDI/DPDI Gain depending on current scanning mode)
#define id_cd_window_horiz_position_shift       510     // CD Window Horizontal
Position
#define id_cd_window_horiz_position_shift_asynch        538     // CD Window
Horizontal Position asynchronous shift
#define id_cd_window_vert_position_shift        512     // CD Window Vertical
Position
#define id_cd_window_vert_position_shift_asynch         539     // CD Window
Vertical Position asynchronous shift
#define id_cd_window_size_shift                 537     // CD Window Size
#define id_cd_window_size_shift_loop            513     // CD Window Size; go to
first when is reached last value
#define id_cd_window_horiz_size_shift           514     // CD Window Horizontal
Size
#define id_cd_window_vert_size_shift            516     // CD Window Vertical
Size
#define id_cd_window_horiz_vert_size_shift              540     // CD Window
Horizontal and Vertical Size
#define id_cd_window_horiz_vert_size_shift_asynch       541     // CD Window
Horizontal and Vertical Size asynchronous shift
#define id_cd_color_map_shift                   518     // CD Color Map (CFM,
PDI, DPDI)
#define id_cd_palette_invert_shift              531     // CD Palette Invert
(CFM or DPDI depending on current scanning mode); -1 - invert on; 1 - invert
off;
#define id_cd_palette_invert                    535     // CD Palette Invert
(CFM or DPDI depending on current scanning mode); Toggle on/off; val = 0;
#define id_cd_color_priority_shift              520     // CD B/Color Priority
#define id_cd_color_threshold_shift             522     // CD Color Threshold
#define id_cd_lines_density_shift               524     // CD Lines Density
#define id_cd_wall_filter_shift                 526     // CD Wall Filter
#define id_cd_baseline_shift                    527     // CD Baseline
#define id_cd_color_averaging_shift             529     // CD Color Averaging
#define id_cd_dynamic_range_shift               530     // PDI/DPDI Dynamic
Range
#define id_cd_pulses_number_shift               532     // CD Pulses Number (CD
Pulse Length)
#define id_cd_packet_size_shift                 533     // CD Packet Size (CFM
or PDI/DPDI depending on current scanning mode)
```

```
#define id_cd_spatial_filtering_shift2          534     // CD Spatial Filtering:
Off, 1, 2, 3, 4
#define id_cd_scale_shift                       542     // PDI/DPDI Scale
#define id_cd_transparency_shift2               543     // CD Transparency: Off,
0.1, 0.2, ...

// Pulsed Wave (PW) and Continuous Wave (CW) Doppler mode controls
#define id_pw_correction_angle_shift    600     // PW Correction Angle
#define id_pw_steering_angle_shift      602     // PW Steering Angle
#define id_pw_steering_angle_shift_loop 641     // PW Steering Angle go to first
when is reached last value
#define id_pw_prf_shift                 604     // PW PRF
#define id_pw_color_map_shift           605     // PW Color Map
#define id_pw_palette_invert_shift      603     // PW Palette Invert; -1 - off;
1 - on;
#define id_pw_palette_invert            681     // PW Palette Invert; toggle;
val = 0;
#define id_pw_power_shift               606     // PW Power
#define id_pw_dynamic_range_shift       607     // PW Dynamic Range
#define id_pw_gain_shift                608     // PW Gain
#define id_pw_sample_volume_horiz_position_shift        610     // PW Sample
Volume Horizontal Position
#define id_pw_sample_volume_horiz_position_shift_asynch2        642     // PW
Sample Volume Horizontal Position
#define id_pw_sample_volume_vert_position_shift         612     // PW Sample
Volume Vertical Position
#define id_pw_sample_volume_vert_position_shift_asynch2         643     // PW
Sample Volume Vertical Position
#define id_pw_sample_volume_size_shift                  614     // PW Sample
Volume Size
#define id_pw_sweep_speed_shift         616     // PW Sweep Speed
#define id_pw_baseline_shift            618     // PW Baseline
#define id_kb_cmd_pw_baseline_shift2    682     // PW Baseline shift taking into
account PW Invert
#define id_pw_invert                    620     // PW Invert checkbox; toggle;
val = 0;
#define id_pw_is_pw_invert              134     // GET bool PW invert value of
current image. ParamGetBool(id_pw_is_pw_invert)
#define id_pw_sound_volume_shift        621     // PW Sound Volume
#define id_pw_wall_filter_shift         623     // PW Wall Filter
#define id_pw_invert_shift              626     // PW Invert; -1 - off, +1 - on
#define id_pw_scale_shift               627     // PW Scale
#define id_pw_frequency_shift           628     // PW Frequency
#define id_pw_frequency_shift_loop      629     // PW Frequency; go to first
when is reached last value
#define id_pw_smooth_shift              639     // PW Smoothing
#define id_pw_spectral_averaging        640     // PW Spectral Averaging;
toggle; val = 0;
#define id_pw_auto_trace                680     // Turn PW/CW auto trace on/off;
toggle; val = 0;

#define id_are_controls_enabled_b       202     // Return true if B controls are
enabled. ParamGetBool(id_are_controls_enabled_b)
#define id_are_controls_enabled_m       203     // Return true if M controls are
enabled. ParamGetBool(id_are_controls_enabled_m)
#define id_are_controls_enabled_pw      204     // Return true if PW controls
```

```
are enabled. ParamGetBool(id_are_controls_enabled_pw)
#define id_are_controls_enabled_cfm     205     // Return true if CFM controls
are enabled. ParamGetBool(id_are_controls_enabled_cfm)
#define id_are_controls_enabled_pdi     206     // Return true if PDI controls
are enabled. ParamGetBool(id_are_controls_enabled_pdi)
#define id_are_controls_enabled_dpdi    207     // Return true if DPDI controls
are enabled. ParamGetBool(id_are_controls_enabled_dpdi)

#define id_is_control_enabled           208     // Return true if control with
passed integer identifier is enabled. ParamGetBool2(id_is_control_enabled,
b_focus_shift)
// Supported id_is_control_enabled identifiers: m_line_position_shift,
cd_window_position_shift, pw_sample_volume_position_shift,
// b_focus_shift, zoom_factor_shift, pw_sweep_speed_shift, m_sweep_speed_shift,
b_steering_trapezoid_angle_shift
// cd_steering_angle_shift, pw_steering_angle_shift, cd_prf_shift,
cd_window_size_shift,
// pw_prf_shift, pw_scale_shift, use_duplex_triplex_shift, cd_gain_shift,
pw_gain_shift,
// pw_baseline_shift, thumbnails_show, cd_baseline_shift,
pw_correction_angle_shift, pw_sample_volume_position_shift.

#define id_get_exam_id                  236     // GET identifier of current
exam (from "Patient" window). ParamGetInt(id_get_exam_id)
#define id_new_patient                  930     // New patient command; val = 0;
#define id_new_exam                     238     // Start new exam for current
patient; val = 0;
#define id_quick_save                   107     // Image quick save; val = 0;
#define id_last_used_action_image_save  160     // Image save (save as) button's
command; val = 0;
#define id_quick_print                  108     // Image quick print; val = 0;
#define id_last_used_action_image_print 161     // Image printing button's
command; val = 0;
#define id_report_quick_print           135     // Report quick print; val = 0;
#define id_last_used_action_report      162     // Report button's command; val
= 0;
#define id_tool_measurement_1d          136     // Select "line" tool (primary
tool) of current mode; val = 0;
#define id_tool_measurement_2d          137     // Select "ellipse" tool
(secondary tool) of current mode; val = 0;
#define id_tool_annot_label             219     // Open Annotations ControlBar
and select label tool; val = 0;
#define id_tool_annot_arrow             220     // Open Annotations ControlBar
and select arrow tool; val = 0;
#define id_tool_annot_text              237     // Open annotations ControlBar
and select text tool; val = 0;
#define id_tool_obj_delete              230     // "Delete selected" button; val
= 0;
#define id_measurements_delete_all      138     // "Delete All" button; val = 0;
#define id_thumbnails_show              635     // Show thumbnails; val = 0;
#define id_thumbnails_show_hide         638     // Show/hide thumbnails
depending on their current state; val = 0;
#define id_thumbnails_selection_shift   636     // Virtual marker for selecting
thumbnail for opening.
#define id_thumbnails_selection_open    637     // Open thumbnail that was
selected using virtual marker; val = 0;
```

```
#define id_presets                      117     // Show Presets ControlBar; val
= 0;
#define id_preset_apply1                919     // Apply preset. As command
value is passed preset name. ParamSetString(id_preset_apply1, "Default")
#define id_presets_get_count            920     // GET the number of presets of
current probe. ParamGetInt(id_presets_get_count)
#define id_preset_get_name              921     // GET the name of preset of
currrent probe by passed preset index. ParamGetString2(id_preset_get_name, 0)
#define id_presets_get_count2           922     // GET the number of presets of
probe with passed code. ParamGetInt2(id_presets_get_count2, probe_code)
#define id_preset_get_name2             923     // GET the name of preset of
probe with passed code by passed preset index.
ParamGetString3(id_preset_get_name, probe_code, 0)
#define id_presets_load_from_path       933     // Load presets from passed
folder, for example, c:\temp\presets2. ParamSetString(id_presets_load_from_path,
"c:\\temp\\presets2")
#define id_cine_controls                246     // Show Cine ControlBar; val =
0;
#define id_measurements_controls        123     // Show Measurements ControlBar;
val = 0;
#define id_calculations_controls        124     // Show Calculations ControlBar;
val = 0;
#define id_navigate_calc_field          232     // When is opened Calculations
ControlBar, select previous/next field in it.
#define id_biopsy_controls              129     // Show Biopsy ControlBar; val =
0;
#define id_annotations_controls         130     // Show Annotations ControlBar;
val = 0;
#define id_body_marks_controls          131     // Show Body Marks ControlBar;
val = 0;
#define id_body_marks_get_count         233     // GET the number of body mark
images. ParamGetInt(id_body_marks_get_count)
#define id_body_mark_set_by_idx         234     // Set Body Mark by passed its
index (0, 1, 2, ...)
#define id_body_mark_remove             235     // Remove Body Mark; val = 0;
#define id_user_manual                  144     // Open user manual; val = 0;
#define id_app_set_topmost              924     // Set application topmost
status: 1 - on, 0 - off
#define id_app_set_visible              925     // Set application visible
status: 1 - on, 0 - off
#define id_app_set_window_state         926     // Set main window state: 0 -
normal, 1 - minimized
#define id_app_exit                     927     // Exit software; val = 0;
#define id_set_patient_name             928     // Set passed string as patient
name. ParamSetString(id_set_patient_name, "Text")
#define id_set_patient_id               929     // Set passed string as patient
id. ParamSetString(id_set_patient_id, "Text")
#define id_shutdown                     186     // Initiate computer shut down;
val = 0;
#define id_patient2                     931     // Show patient window; val = 0;
#define id_options2                     932     // Show options window; val = 0;
// IMPORTANT. Cine end time is generated at the software graphical user
interface (GUI) after finishing of GUI invoked freeze command. This means that
it may differ from real time when hardware scanning was finished.
#define id_get_cine_end_date_time_str   690     // Get cine end (freeze)
absolute date and time as string. Format: "yyyy.MM.dd HH:mm:ss.ffffff".
```

```
ParamGetString(id_get_end_date_time_str).
```