US-Wave.dll
User Guide

REVISION	DATE	COMMENTS	WRITTEN BY
R1	10/10/13	Updated Document	Aimeric DELAGE

TABLE OF CONTENTS

REVISIONS

REVISION	DATE	DESCRIPTION	AUTHOR
R1	10/10/13	“Sampling Frequency” parameters	AD
R0	03/09/13	Initial document	AD

PRESENTATION

This document presents the different functions of the DLL US-Wave.dll in order to facilite its call during the creation of programs. Each function is detailed with its different inputs/outputs, the values of each parameter and also examples. This DLL must be used in a Windows environment under the C calling convention.
According to the version of your appealing software (32 or 64 bits), you will be able to use the 32 bits version or 64 bits of our DLL (see the part devoted to the 64 bits version)
The tools and files used by this one are in a repertory called US-Wave being on your computer (C:\US-Wave)
You will also find in this repertory, a folder containing the drivers of the device (to be indicated to the computer if the device would be not be detected automatically) and user examples.

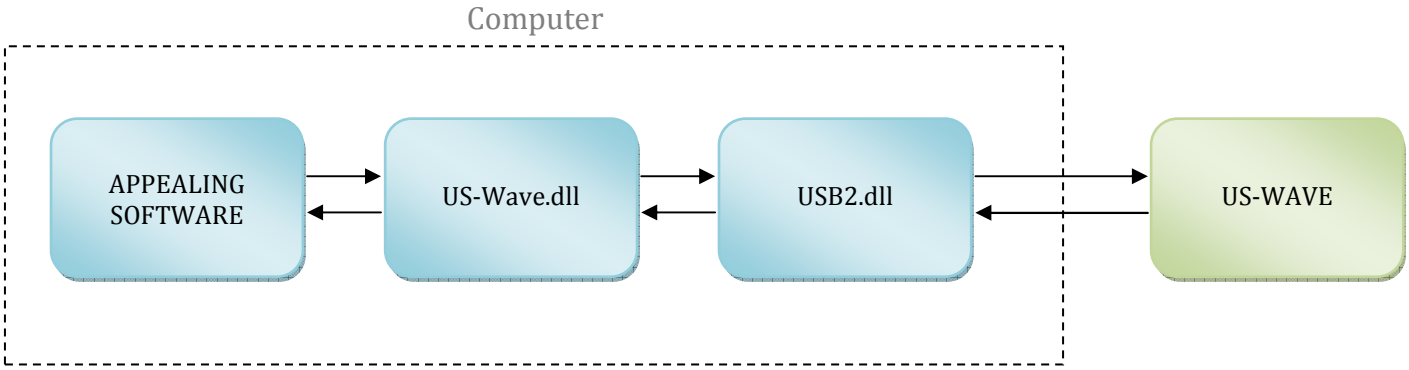
In this document, you will find 2 distinct parts for the DLL's functions. These two parts correspond to the 2 acquisition modes suggested by the device.
They are detailed below :

- Standard Acquisition Mode : the device is configured with different functions to set up all parameters before making an acquisition with the dedicated function (Data RF). A call of this function has for effect to make an acquisition. That means that the call frequency of this function will fix the acquisitions repetition rate.

- Sequencer Acquisition Mode : allows to program a number of acquisitions with different parameters. This mode uses only three functions with in more Transmitting Curve function which is common to the 2 modes. The user can define a number of sequences corresponding to a series of acquisitions which will be stored in the SDRAM memory of the device. It is advised to control the previous mode before using this one.

US-Wave uses the USB2 link to transfer its measures and receive its parameters. The DLL US-Wave.dll established USB connection by calling the DLL USB2.dll being in the same repertory. This DLL calling for the management of USB is transparent for the user, only US-Wave.dll must be called.

Synoptic :



It is preferable that the program of the user calls the DLL starting from the folder C:\US-Wave, this in order to avoid possible errors.

FORMAT

The library US-Wave.dll contains only one function which has the same name as the DLL (the prototype of the function is described in the file US-Wave.h). This function includes a series of sub-functions which thereafter in this document will be regarded fully as functions in order to facilitate the use of the DLL.

DLL name :

US-Wave.dll

Prototype of the US-Wave function :

U16 Return_Code = USWave(U16 Device_Number, Char [] Function_Name, U16* Input_Array, U16* Output_Array)

with : U16 = Unsigned short int (2 bytes ; 0 to 65535)
Char [] = String (unsigned char, 1 byte ; 0 to 255)
U16* = Unsigned short int pointer

Inputs/outputs parameters for the US-Wave function :

Device Number : number given by the USB port (0 in the case of only one device is connected)

Function Name : name of the function which will be executed by the DLL

Input Array : input table (size = 1Mega) containing parameters which will be sent to the device

Output Array : output table (size = 1Mega) containing measures sent by the device

Return Code : code to inform the appealing software of a possible error during a call

Input/Output Array formats :

Element	Description	Value
0	Size of the table	➢ 0 : no input parameter ➢ 1 : one dimensional table ➢ 2 : two dimensional table
1	Number of elements by line	➢ X : number of elements (1 to …)
2	Number of lines	➢ Y : number of lines (1 to …)
3	Data/Parameter	➢ 0 to 65535
…	Data/Parameter	➢ 0 to 65535
X	Data/Parameter	➢ 0 to 65535

FORMAT

Error codes sent by the DLL :

Code	Description
0	No error
1	Error in the USB2.dll loading
2	Init USB function has not been called before using other functions
3	Device isn't connected or the driver isn't installed
4	The requested function doesn't exist or the name is incorrect

64 Bits version

This device can be used in 64 Bits environment. If you have a 64 Bits Windows and a 32 Bits software for programming, you don't need to use the 64 Bits DLL. In the case where your appealing software is 64 Bits, it is essential to use the 64 Bits version of this DLL and to follow the procedure described below.

Calling procedure for 64 Bits DLL :

1) Launch the executable called ExeUSB2X86 and being at the location C:\US-Wave
2) Wait few seconds before to load the DLL
3) Load 64 Bits version of the DLL, called US-WaveX64.dll
4) Use the functions of the DLL in the same way as the 32 Bits version

You can check if the executable is present by opening the Task Manager (Ctrl + Alt + Suppr)
Warning : only start once ExeUSB2X86, otherwise you will appear more executable with the same name. In this case, select all executables having the same name to remove with the command « End Process »
Then you can restart just once ExeUSB2X86 and resume normal operation.

MATLAB®

The DLL of this device is compatible with different versions of Matlab® software in 32 or 64 bits version. A script example called Starter_US-Wave is provided (french and english versions) in the repertory C:\US-Wave with all functions available. Before to launch this script, you must indicate to Matlab® software the C compiler wich will be associated for the DLL callings. To do this, you must type in Matlab® command prompt the following command : mex -setup
The following message appears :

```
Welcome to mex -setup.  This utility will help you set up
a default compiler.  For a list of supported compilers, see
http://www.mathworks.com/support/compilers/R2012a/win32.html

Please choose your compiler for building MEX-files:

Would you like mex to locate installed compilers [y]/n?
```

Type y

```
Select a compiler:
[1] Lcc-win32 C 2.4.1 in C:\PROGRA~2\MATLAB\R2012a\sys\lcc
[2] Microsoft Software Development Kit (SDK) 7.1 in C:\Program Files (x86)\Microsoft Visual Studio

[0] None

Compiler:
```

Select the LCC compiler provided by Matlab® by typing 1

```
Please verify your choices:

Compiler: Lcc-win32 C 2.4.1
Location: C:\PROGRA~2\MATLAB\R2012a\sys\lcc

Are these correct [y]/n?
```

Valid your choice by typing y
Your Matlab® software is now configured for the DLL loading in C calling convention.

Then just open the file, add the directory C:\US-Wave at current directory (by the Set Path) and press F5 to launch a series of acquisitions with predefined parameters.

FUNCTIONS (STANDARD)

Init USB

Description :

Function must be called first in the beginning of the program, to establish the USB connection with the device. After that, all other functions can be used.
It isn't necessary to recall this function except in the case of a USB disconnection.

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : "Init USB"

Input_Array : no input parameters

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 0;                % No element
Input_Array[2] = 0;                % No line

USWave(0, "Init USB", &Input_Array, &Output_Array);
```

FUNCTIONS (STANDARD)

Mode

Description :

Function to select the operating mode of the device. There are two modes : Pitch and Catch (transmitter and receiver separated => 2 probes) and Pulse Echo (transmitter and receiver connected => only one probe connected to the receiver)

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : “Mode”

Input_Array[3] : select mode (0 => Pulse Echo or 1 => Pitch and Catch)

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 1;                % One element
Input_Array[2] = 1;                % One line

Input_Array[3] = 0;                % Pulse Echo Mode

USWave(0, “Mode”, &Input_Array, &Output_Array);
```

FUNCTIONS (STANDARD)

Input Impedance

Description :

Function to select receiver input impedance. Four values are available : 50 Ohms, 100 Ohms, 200 Ohms and 6 KOhms.

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : “Input Impedance”

Input_Array[3] : impedance selection (0 => 50 Ohms, 1 => 100 Ohms, 2 => 200 Ohms or 3 => 6 KOhms)

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 1;                % One element
Input_Array[2] = 1;                % One line

Input_Array[3] = 1;                % Input impedance = 100 Ohms

USWave(0, “Input Impedance”, &Input_Array, &Output_Array);
```

FUNCTIONS (STANDARD)

Transmitting Mode

Description :

Function that allows you to choose the desired transmitting type. The device has 2 types of transmitter, one called Pulse and the other Analog.
Pulse transmitter sends a transmission signal having the form of a high voltage negative pulse at adjustable width.
Analog transmitter is used to create its own transmission signal defining its shape, frequency and amplitude. There may be two possibilities either internally creating the transmission signal by a DLL function or by connecting a signal generator on the input provided for this purpose, taking care that the maximum input levels (+/- 2.5 Volts)

External analog mode : to use this mode, the device must be in external trigger (see the function Trigger for more information) to synchronize the transmission signal with the start of the acquisition.

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : “Transmitting Mode”

Input_Array[3] : select the transmitting mode (0 => Internal Analog, 1 => Pulse or 2 => External Analog)

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 1;                % One element
Input_Array[2] = 1;                % One line

Input_Array[3] = 0;                % Internal analog transmission

USWave(0, “Transmitting Mode”, &Input_Array, &Output_Array);
```

FUNCTIONS (STANDARD)

Width

Description :

Function to adjust the width of the transmission pulse in the case of the pulse transmitter . The minimum width is 36 ns in step of 8 ns (from the decimal value 5) The maximum value is set to 4 µs (in the DLL, the decimal value is limited to 500)

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : "Width"

Input_Array[3] : adjust the pulse width (1 => 36 ns, 2 => 54 ns, 3=> 68 ns, 4=> 78 ns, 5 => 87 ns, 6 => 95 ns, …) on 50 Ohms load

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 1;                % No element
Input_Array[2] = 1;                % No line

Input_Array[3] = 1;                % Transmission pulse of 36 ns

USWave(0, "Width", &Input_Array, &Output_Array);
```

FUNCTIONS (STANDARD)

Transmitting Delay

Description :

Function to add a delay between the synchro signal and the beginning of the transmission signal. This synchro signal can be software, internal ou external (see the function Trigger for more information). This means that if the delay is different of 0, the device will start to digitize before having sent its transmission signal.

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : "Transmitting Delay"

Input_Array[3] : select the delay by step of 8 ns (0 => no delay added, 1 => +8 ns, 2 => +16 ns
3 => +24 ns, …) The maximum decimal value is fixed at 65535

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 1;                % One element
Input_Array[2] = 1;                % One line

Input_Array[3] = 0;                % No delay added (the delay of this device is about 28 ns)

USWave(0, "Transmitting Delay", &Input_Array, &Output_Array);
```

FUNCTIONS (STANDARD)

Sampling Delay

Description :

Function to add a delay between the synchro signal and the beginning of the sampling window. This synchro signal can be software, internal ou external (see the function Trigger for more information). This means that if the delay is different of 0, the device will start to digitize with an offset of X ns relative to the synchro signal.

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : “Sampling Delay”

Input_Array[3] : select the delay by sampling frequency step (0 => no delay, 1 => 1*(1/SampFreq), 2 => 2*(1/SampFreq), 3 => 3*(1/SampFreq), …) Maximum decimal value is fixed at 65535

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 1;                % One element
Input_Array[2] = 1;                % One line

Input_Array[3] = 1;                % Delay of 8 ns with a sampling frequency of 125 MHz
                                   % (8 ns = 1/125MHz)

USWave(0, “Sampling Delay”, &Input_Array, &Output_Array);
```

FUNCTIONS (STANDARD)

Gain

Description :

Function to select the constant gain value over the entire duration of the sampling window. The gain can range from 0 to 80 dB with a variation step of 1 dB. This function can't be used in the same time that the Gain Curve function (either one or the other)

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : "Gain"

Input_Array[3] : select the gain value of 0 to 80 dB by 1 dB step value

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 1;                % One element
Input_Array[2] = 1;                % One line

Input_Array[3] = 30;               % Gain value 30 dB

USWave(0, "Gain", &Input_Array, &Output_Array);
```

FUNCTIONS (STANDARD)

Gain Curve

Description :

Function to create a gain curve with 2048 points. Gain value can be selected between 0 and 80 dB by 1 dB step value. The time difference between each point of this curve depends on the selected sampling frequency (1/SampFreq)
This function can't be used in the same time that the Gain function (either one or the other)

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : "Gain Curve"

Input_Array[3] : gain before the beginning of this curve (0 to 80 dB)
Input_Array[4] : first point of the gain curve (0 to 80 dB)
…
Input_Array[2050] : last point of the gain table (to fix a constant gain after the gain curve, it is advisable to use the last 6 points of this table and put the desired gain value in these 6 elements ; in the worst case)

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 2048;             % 2048 points (don't modify this value)
Input_Array[2] = 1;                % One line

Input_Array[3] = 0;                % Gain value 0 dB before the curve

Input_Array[4] = 11;               % First point of this curve equal to 11 dB
…                                  % Intermediate  values of the curve
Input_Array[2044] = 80;            % Last point of this curve equal to 80 dB

Input_Array[2045] = 0;             % Programming the last 6 points of this table
…                                  % to have a gain of 0 dB after the curve (worst case when
                                   % we have a strong variation)

USWave(0, "Gain Curve", &Input_Array, &Output_Array);
```

FUNCTIONS (STANDARD)

Gain Curve Delay

Description :

Function to add a delay between the synchro signal and the beginning of the gain curve (see the function Gain Curve for more information). This synchro signal can be software, internal ou external (see the function Trigger for more information). This delay determines the moment where the gain curve will start.

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : “Gain Curve Delay”

Input_Array[3] : select the delay by sampling frequency step (0 => no delay, 1 => 1*(1/SampFreq), 2 => 2*(1/SampFreq), 3 => 3*(1/SampFreq), …) Maximum decimal value is fixed at 65535

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 1;                % One element
Input_Array[2] = 1;                % One line

Input_Array[3] = 1;                % 8 ns delay for a sampling frequency of 125 MHz
                                   % (8 ns = 1/125MHz)

USWave(0, “Gain Curve Delay”, &Input_Array, &Output_Array);
```

FUNCTIONS (STANDARD)

Sampling Frequency

Description :

Function to select the sampling frequency of the measured signals. Seven choices : 125 MHz, 75 MHz, 50 MHz, 25 MHz, 10 MHz, 5 MHz or on external frequency (see the connector for that).
Sampling frequency value and the number of points required will determine the size of the sampling window and thus the total acquisition time.

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : "Sampling Frequency"

Input_Array[3] : select the sampling frequency (0 => 125 MHz, 1 => 75 MHz, 2 => 50 MHz, 3 => 25 MHz, 4 => 10 MHz, 5 => 5 MHz or 6 => External frequency)

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 1;                % One element
Input_Array[2] = 1;                % One line

Input_Array[3] = 0;                % Sampling frequency at 125 MHz (1/125 MHz = 8
                                   % ns, time between two samples)

USWave(0, "Sampling Frequency", &Input_Array, &Output_Array);
```

FUNCTIONS (STANDARD)

PRF

Description :

Function to fix the frequency (or the time) between two transmissions in internal trigger mode (see the function Trigger for more information). This means, the device will send transmission signals at this repetition frequency. Choosing this value in function of the sampling window time and delays parameters (Transmitting and/or Sampling delays)
This value is adjustable by step of 200 ns.

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : "PRF"

Input_Array[3] : fix the repetition frequency (Δ 1 => Δ 200 ns) It is recommended that this frequency doesn't exceed 20 KHz.

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 1;                % One element
Input_Array[2] = 1;                % One line

Input_Array[3] = 5000;             % Internal repetition frequency of 1 KHz
                                   % (5000 * 200 ns = 1 ms ; 1 ms => 1 KHz)

USWave(0, "PRF", &Input_Array, &Output_Array);
```

FUNCTIONS (STANDARD)

Trigger

Description :

Function to select the trigger mode for the transmission signals.
Three modes are availables :

- Soft : the calling frequency of the RF Data function will set the time between each transmission
- Internal : a programmed counter with the PRF function fix the time between each transmission and it turns in infinitely loop
- External : a synchronization signal defined by the user and connected to the input of the device determines the time interval between each transmission.

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : “Trigger”

Input_Array[3] : select the trigger mode (0 => Soft, 1 => Internal or 2 => External)

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 1;                % One element
Input_Array[2] = 1;                % One line

Input_Array[3] = 0;                % Trigger mode by software (the software will define
                                   % the time between each transmission)

USWave(0, “Trigger”, &Input_Array, &Output_Array);
```

FUNCTIONS (STANDARD)

Data RF

Description :

Function used to acquire a number of points defined by the user. The number of points and the sampling frequency determine an acquisition time also know as sampling window. The maximum number of points is limited to 16 384.

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : “Data RF”

Input_Array[3] : requested number of samples (until 16 384 max)

Output_Array : samples will be stored into this table

Example :

```
U16 Input_Array[1000000];                 % Input table declaration 16 bits
U16 Output_Array[1000000];                % Output table declaration 16 bits

Input_Array[0] = 1;                       % Input table 1D
Input_Array[1] = 1;                       % One element
Input_Array[2] = 1;                       % One line

Input_Array[3] = 16000;                   % Acquisition request of 16 000 points

USWave(0, “Data RF”, &Input_Array, &Output_Array);

Output_Array[0];                          % Return the table type
Output_Array[1];                          % Return the number of data by line (here 16 000)
Output_Array[2];                          % Return the number of lines (here 1)

Output_Array[3];                          % first point of the sampling window
…
Output_Array[3+15999];                    % last point of the sampling window

%% Convert digital data to volts %%

(Output_Array[3] – 32768) * 0.00003052
…
(Output_Array[3+15999] – 32768) * 0.00003052
```

FUNCTIONS (STANDARD/SEQUENCER)

Transmitting Curve

Description :

Function to create the transmission signals which will be sent by the device. The principle is as follow : the user creates the signal point per point in function of its frequency, amplitude and shape. This signal is then scaled to get points with numerical values for the digital analog converter. These points are then stored into a table that is sent to the SRAM memory of the device.
This function can be used in either Standard Acquisition Mode or Sequencer Acquisition Mode.
In the Standard Acquisition Mode, when there is a transmission request, the entire SRAM memory is read and it creates the programmed signal sent to the analog transmitter. You must be careful to fill this memory properly with the desired signal.
In Sequencer Mode, there may be different patterns stored in the memory as the interest of this mode is to send different transmission signals for each acquisition.
The sampling step of the digital analog converter is 8 ns (1/125 MHz), which means that between each point of the pattern, there is a period of 8 ns.
This SRAM memory can store 256 000 points, the user is free to organize it in function of the number of patterns and their sizes. The beginning address of each pattern is linked to its size and that of the precedent patterns (see the example below for more details)

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : “Transmitting Curve”

Input_Array[3] : first point of the table (digital value between 0 and 4095, 0 => - Vmax and 4095 => + Vmax)
…
Input_Array[256002] : last point of the table (digital value between 0 and 4095)

Output_Array : not used

Example (Matlab® oriented) :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 64000;            % Number of elements by line (don’t modify this value)
Input_Array[2] = 4;                % Number of lines (don’t modify this value)
                                   % 4 * 64 000 = 256 000 points
```

FUNCTIONS (STANDARD/SEQUENCER)

Transmitting Curve

%% Parameters used to create the pattern with sine function of Matlab® %%

```
Fe = 125e+06;                    % Sampling frequency for the DAC (125 MHz)
Nb_Bits = 12;                    % Number of bits of the Digital Analog Converter (DAC)
Y = 2^Nb_Bits / 2;               % Reference 0 (2048 => 0 Volts)
Phi = 0;                         % Dephasing in degrees
Phi = Phi*pi / 180;              % Dephasing calculation
```

%% Editable parameters %%

```
% Pattern N°1 %
F0 = 1e+06;                      % Center frequency of the signal
N0 = round(Fe/F0);               % Calculation of the number of points in the signal
t0 = [0 :1/Fe:(N0 - 1)/Fe];      % Time vector
s0 = 0.5 * sin(2*pi*F0*t0 + Phi); % Signal vector (amplitude is fix to 0.5, this means that
                                 % the transmission signal will be between – Vmax /2
                                 % and +Vmax / 2)
s0 = Y + s0 * (Y - 1);           % Signal vector with an offset of 0
s0 = [s0,2048];                  % Reset signal (the signal must start and finish by 2048)
N0 = N0 + 1;                     % Update signal size

Input_Array[3] = s0[0];          % Writing points in the transfer table
…                                % The table must be completely filled by data
Input_Array[X] = s0[Max];        % Fill the not used table elements by 2048

USWave(0, “Transmitting Curve”, &Input_Array, &Output_Array);
```

FUNCTIONS (SEQUENCER)

Load Sequencer

Description :

Function to load Sequencer's parameters into the internal memory of the device. To recap, the Sequencer Acquisition Mode allows to make acquisitions with different parameters and store data into the SDRAM memory of the device. There are parameters common at all sequences (6 parameters commons) and also parameters for each sequence. A sequence is composed of one or more acquisitions. Each sequence has 9 parameters. The number of sequences is limited to 226.
A sequence is linked to a predefined pattern by the Transmitting Curve function. The sequence N°1 will be linked to the pattern N°1, … The example below, will show you the possibilities for each sequence.

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : "Load Sequencer"

Input_Array[3] : first parameter of the table (see the example for its description, parameter values are similar to those of the other functions)
…
Input_Array[3+2047] : last parameter of the table (see the example for its description)

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 2048;             % 2048 parameters (don't modify this value)
Input_Array[2] = 1;                % One line

% Common parameters to all sequences %
Input_Array[3] = 2;                % Number of patterns and so of sequences (here 2
                                   % sequences)
Input_Array[4] = 1;                % Select the trigger mode (here Internal mode). Soft trigger
                                   % can't be used in the Sequencer Acquisition Mode.
Input_Array[5] = 5000;             % PRF value (5000 => 1 KHz)
Input_Array[6] = 0;                % Not used for this moment
Input_Array[7] = 0;                % Operating mode (here Pulse Echo)
Input_Array[8] = 0;                % Not used for this moment
```

FUNCTIONS (SEQUENCER)

Load Sequencer

```
% Parameters for the first sequence (first pattern) %
Input_Array[9] = 1;                 % Number of acquisitions for this sequence (here one
                                    % with the pattern N°1)
Input_Array[10] = 126+3;            % Number of points in the pattern (here, the pattern
                                    % has 126 points that always add the value 3)
Input_Array[11] = 0;                % Transmitting delay (here zero)
Input_Array[12] = 0;                % Sampling delay (here zero)
Input_Array[13] = 15;               % Gain (here 15 dB)
Input_Array[14] = 10;               % Number of bursts in the sampling window. One
                                    % burst is equal to 1024 samples (here 10 * 1024 = 10 240
                                    % samples for this acquisition)
Input_Array[15] = 0;                % Sampling frequency (here 0 => 125 MHz)
Input_Array[16] = 1;                % Input impedance (1 => 100 Ohms)
Input_Array[17] = 0;                % Start address for the pattern N°1 in the memory (here 0)
                                    % Start address is determined by the precedent pattern
                                    % size

% Parameters for the second sequence (second pattern) %
Input_Array[18] = 3;                % Number of acquisitions for this sequence (here three
                                    % with the pattern N°2)
Input_Array[19] = 200+3;            % Number of points in the pattern (here, the pattern
                                    % has 200 points that always add the value 3)
Input_Array[20] = 0;                % Transmitting delay (here zero)
Input_Array[21] = 0;                % Sampling delay (here zero)
Input_Array[22] = 22;               % Gain (here 22 dB)
Input_Array[23] = 40;               % Number of bursts in the sampling window. One
                                    % burst is equal to 1024 samples (here 40 * 1024 = 40 960
                                    % samples for each acquisition)
Input_Array[24] = 1;                % Sampling frequency (here 1 => 75 MHz)
Input_Array[25] = 0;                % Input impedance (0 => 50 Ohms)
Input_Array[26] = 126;              % Start address for the pattern N°2 in the memory (here
                                    % 126 because pattern N°1 has 126 points)

Input_Array[27] = 0;                % Reset all other parameters
…
Input_Array[2050] = 0;              %

USWave(0, “Load Sequencer”, &Input_Array, &Output_Array);
```

FUNCTIONS (SEQUENCER)

Exe Sequencer

Description :

Function to valid the Sequencer Acquisition Mode and launch a series of acquisitions defined by the Load Sequencer function. These acquisitions are stored one after the other into the SRAM memory. Before to execute this function, make sure that the device is configured in analog transmitter mode (see the Transmitting Mode function for more information)

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : “Exe Sequencer”

Input_Array[3] : not used

Output_Array : not used

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 1;                % One element
Input_Array[2] = 1;                % One line

USWave(0, “Exe Sequencer”, &Input_Array, &Output_Array);
```

FUNCTIONS (SEQUENCER)

Read Memory

Description :

Function to read the SDRAM memory which contains the different acquisitions programmed by the Load Sequencer function. The storage order of the data is the same as that of the sequences.
The user can select the start address of reading and also the number of data to read.
This SDRAM memory is organized in rows and in columns. It has 16384 rows and 4 columns, providing the ability to store 65535 bursts. As a reminder, a burst consists of 1024 samples.

SDRAM organization :

	C0	C1	C2	C3
L0	Burst N°0	Burst N°16384	Burst N°32768	Burst N°49152
L1	Burst N°1	Burst N°16385	Burst N°32769	Burst N°49153
…	**…**	**…**	…	**…**
L16383	Burst N°16383	Burst N°32767	Burst N°49151	Burst N°65535

Parameters :

Device_Number : US-Wave number (equal to 0 if only one device is connected)

Function_Name : “Read Memory”

Input_Array[3] : number of bursts which will be read
Input_Array[4] : row number to choose the first burst to read
Input_Array[5] : column number to choose the first burst to read

Output_Array : samples will be stored in this table

Example :

```
U16 Input_Array[1000000];          % Input table declaration 16 bits
U16 Output_Array[1000000];         % Output table declaration 16 bits

Input_Array[0] = 1;                % Input table 1D
Input_Array[1] = 3;                % 3 input parameters
Input_Array[2] = 1;                % One line
```

FUNCTIONS (SEQUENCER)

Read Memory

```
Input_Array[3] = 300;                    % Number of bursts to read (here 300)
Input_Array[4] = 0;                      % Row number for the first burst to read (here N°0)
Input_Array[5] = 0;                      % Column number for the first burst to read (here N°0)

USWave(0, “Read Memory”, &Input_Array, &Output_Array);

Output_Array[0];                         % Not used
Output_Array[1];                         % Not used
Output_Array[2];                         % Non used

Output_Array[3];                         % first point of the first burst read
…
Output_Array[3+X];                       % last point of the last burst read

%% Convert digital data to volts %%

(Output_Array[3] – 32768) * 0.00003052
…
(Output_Array[3+X] – 32768) * 0.00003052
```