Advanced OEM Solutions

Software Functions & Parameter List

Version 1.1.4.1 / 1.1.5.0 / 1.1.5.1

Contents

Revision

2012/03/13	Analog Gain
2012/03/18	Text Editing
2013/01/21	Added Filters; Updated Global Diagram
2013/04/03	Corrections
2013/12/24	Corrections

1 Introduction

This document explains FW parameters that are associated with the function list of the OEM-PA driver (see Section “Low Level API” of *Software_API.pdf*). For more detailed information please see the header file “UTKernelDriverOEMPA.h”.

The description of these parameters is documented by order of their nature and where they apply.

2 Board

Three types of HW are available:

- PA: Phased Array.
- MC: Multiple-Channel.
- FMC: Full Matrix Capture.

Depending on the HW type, some parameters may not be applicable. The column “HW” indicates supported platforms. For some parameters, a copy of the FW parameter is saved on the computer, and there is a special function to read this value instead of requesting the FW parameter over the communication link.

All functions are based on the international system of units (unless specified elsewhere).

The following is a list of parameters and associated functions that are not cycle dependent (i.e. cycle index is not an input parameter):

API Description	**HW**	**Function Name**	**Data type (min, step, max)**
FW version	All	GetFWId	WORD
Connector element count Provides the maximum element count that the system can handle	All	GetElementCountMax	int
Aperture maximum element count Provides the maximum aperture size that the system can handle	All	GetApertureCountMax	int
Temperature (unit is in degrees Celsius)	All	GetTemperatureSensor	
A-scan acquisition enable Must be enabled for each cycle. See in next table: A-scan maximum enable, minimum enable, and saturation enable	All	EnableAscan / GetEnableAscan	bool

(All means “PA”, “MC” and “FMC”)

API Description	**HW**	**Function Name**	**Data type (min, step, max)**
C-scan time of flight acquisition Enables the Time of Flight results in the C-scan data (To enable the C-scan acquisition, you have to enable C-scan acquisition for each cycle, see next table)	PA / MC	EnableCscanTof / GetEnableCscanTof	bool
A-scan bit size Provide the information of how many bits the A-scan data is encoded with.	All	SetAscanBitSize / GetAscanBitSize	enumBitSize (8bits, 12bits, 16bits, 8bits log)
Cycle count Defines the number of cycles (pulses) in a sequence.	All	SetCycleCount / GetCycleCount	int Min=1, step=1, max=2047 (or 4096)
TriggerMode To select the synchronization of the pulse	All	SetTriggerMode / GetTriggerMode	enumOEMPATrigger (internal, encoder, external cycle, external sequence)
TriggerEncoderStep This is the encoder step used for acquisition when “TriggerMode” is set to “encoder mode”, the encoder resolution is not taken into account (raw encoder value).	All	SetTriggerEncoderStep / GetTriggerEncoderStep	DWORD Min=0, step=1, max=0xffffffff
FMC ElementStart To define the first receiver element (only for FMC).	FMC	SetFMCElementStart / GetFMCElementStart	int
FMC ElementStop To define the last receiver element (only for FMC).	FMC	SetFMCElementStop / GetFMCElementStop	int
Reset encoder To reset the raw encoder value in the HW.	All	ResetEncoder	void
RequestIO Request IO streams that contain the encoder counts and digital input states. Encoder and digital inputs are sent by the FW in the same stream. You can select how this stream will be sent by the FW.	All	Set RequestIO / Get RequestIO	enumOEMPARequestIO (NotRequired, OnCycleOnly, OnSequenceOnly, OnDigitalInputOnly, OnDigitalInputAndCycle,OnDigitalInputAndSequence)
RequestIODigitalInputMask If RequestIO is set to some variation of OnDigitalInputX, then this masks sets which input is used and whether it is triggering on a falling or rising edge.	All	SetRequestIODigitalInputMask	int

API Description	HW	Function Name	Data type (min, step, max)
DigitalDebouncer Time to debounce digital input (not used by encoders). You can use the toolbox (device configuration) to use this feature easily.	All	SetDigitalDebouncer / GetDigitalDebouncer	
MappingInput Selection of digital inputs used for encoders and external signals for pulse synchronization (see “TriggerMode”, external cycle signal and external sequence cycle). There are 6 digital inputs. You can use the toolbox (device configuration) to use this feature easily.	All	SetMappingInput / GetMappingInput	enumOEMPAMappingDigitalInput
MappingOutput Selection of digital outputs You can use the toolbox (device configuration) to use this feature easily.	All	SetMappingOutput / GetMappingOutput	enumOEMPAMappingDigitalOutput (low, high, signal cycle, signal sequence)
Digital input Digital input of the IO board that are sent to the computer. Digital inputs that are used for encoder or trigger are not sent.	All	GetDigitalInput / swGetDigitalInput	DWORD Min=0, step=1, max=0x3F
HWKeepAlive KeepALive of the HW. On idle state (pulser is disabled), the HW occasionally sends streams and awaits an answer from the computer. If the answer is not received, the socket is automatically disconnected.	All	SetHWKeepAlive / GetHWKeepAlive	bool

Filter Coefficients/ScaleNote [1]
15 sets of coefficients can be used. Each set has 35 coefficients (the filters architecture is a FIR, symmetrical, order 64)
The scale gives the index of the 41 bits-output to take as LSB to have the needed 18 bits signal.
	All	SetFilter /
		GetFilter
			enumOEMPAFilterIndexfilterIndex
			WORD scale;
			WORD coeff[34];

Table 1: board parameter list

Note [1]: The filter coefficients and scale parameters are given by the SW "DefaultConfiguration.exe" (*Hardware default configuration and custom filters*). To use coefficients calculated by another mean, please contact AOS.

3 Cycle

Here is a list of parameters and associated functions for which cycle index is an input parameter (International System of Units):

API Description	HW	Function Name	Data type (min, step, max)
Time Slot Note [1] Defines the time duration of a cycle (from its pulse to the pulse of the next cycle)	All	SetTimeSlot /GetTimeSlot	double Min=50.0 us, step=10.0 ns, max>1s
Analog Gain To optimize the signal/noise ratio it is better to increase this analog gain before the digital gain.	All (Note [2])	SetGainAnalog / GetGainAnalog / CheckGainAnalog	float Min=0.0 dB, step=0.1 dB, max=47.7 dB
Digital Gain Digital gain for each acquisition channel (common to all cycles of this channel).	All	SetGainDigital / GetGainDigital / CheckGainDigital	double Min=0.0 dB, step=0.1 dB, max=80.0 dB
Beam correction Digital gain that is unique to each cycle (correction for each beam after calibration). After calibration, it is better to use the Digital Gain.	All	SetBeamCorrection / GetBeamCorrection / CheckBeamCorrection	float Min=-80.0 dB, step=0.1 dB, max=+80.0 dB
DAC enable	All	EnableDAC / GetEnableDAC	bool
DAC count Note [2] Defines the number of points for DAC.	All	SetDACSlope / GetDACSlope / CheckDACSlope SetDACGain / GetDACGain / CheckDACGain	int min=0, step=1, max=64
DAC time of flight Note [2] Defines the position of the points for DAC.	All	SetDACSlope / GetDACSlope / CheckDACSlope	double Min=0.0 s, step=10.0 ns, max>1s
DAC slope Note [3] Defines the gain slope between each of the points for DAC.	All	SetDACSlope / GetDACSlope / CheckDACSlope	float Min=-320.0 dB/us, step=10/1024 dB/us, max=319.99 dB/us
Filter index Selects the number of the filter coefficients set to use. Value “eOEMPAFilterOff” means "no filter", which disable also needed features like DC removing. You should always set a value between 1 and 15.	All	SetFilterIndex / GetFilterIndex	enumOEMPAFilterIndex

Multiple Channel acquisition Selects the cycle type (either acquisition either replay).	MC	EnableMultiHWChannelAcquisition / GetEnableMultiHWChannelAcquisition	bool (true for acquisition cycle and false for replay cycle).

Table2: cycle parameters list

Note [1]: See the following timing diagram that describes the "TimeSlot" parameter:

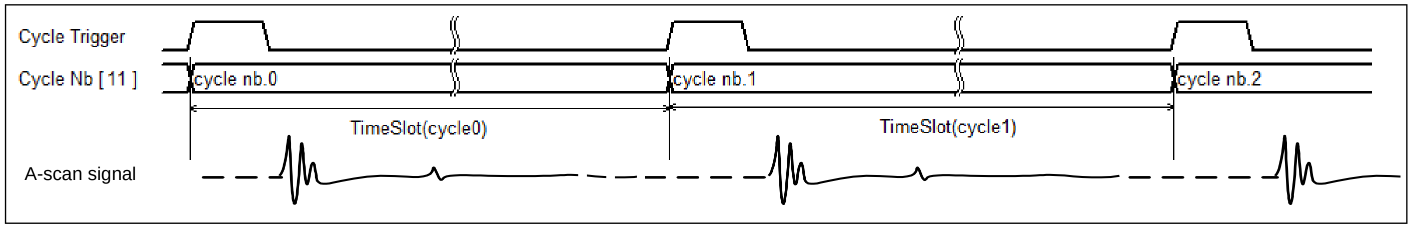


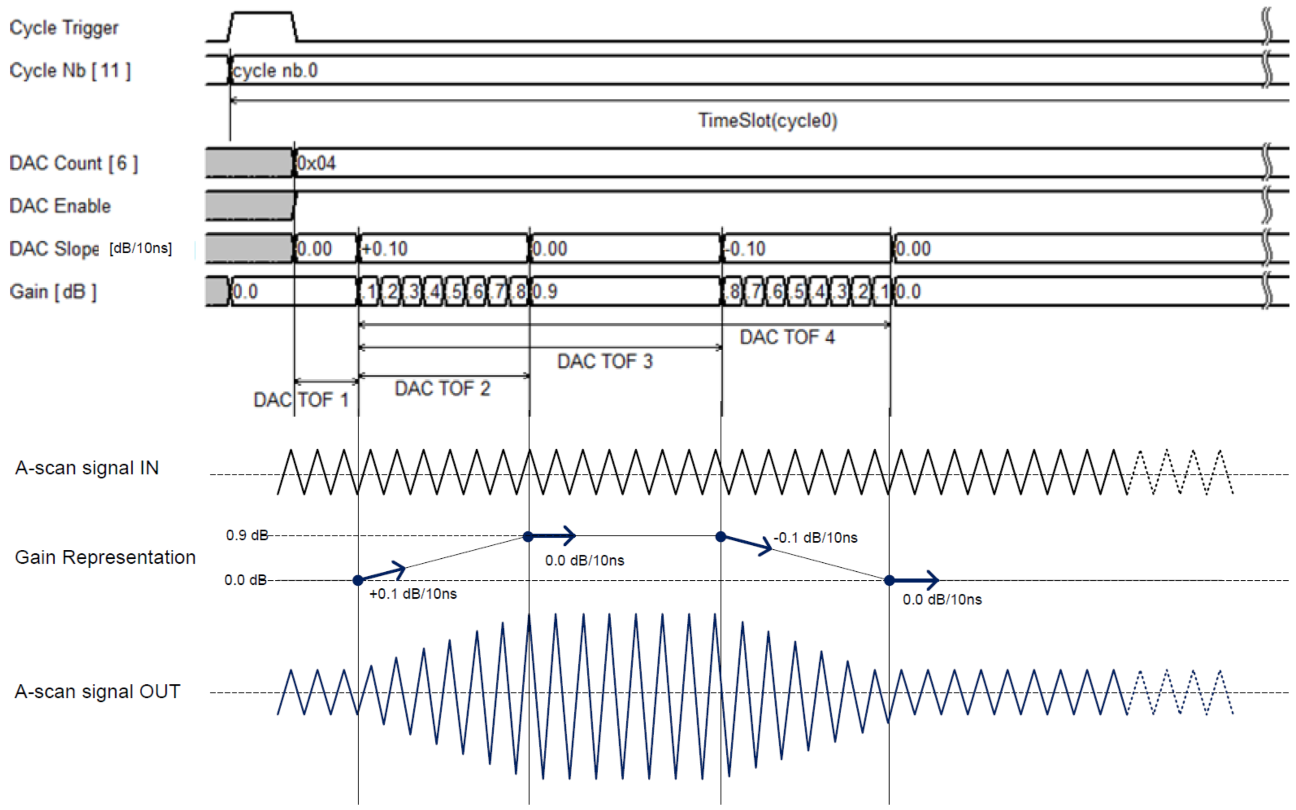
figure.1 TimeSlot parameter description

In the case of FMC, the TimeSlot is specified for all acquired elements.

Note [2]: In the case of MC, AnalogGain is only valid for acquisition cycles.

Note [3]: The "slope" defines how much gain the signal will be increased by, within one CLK period (which here is 10ns). The "smoothest" slope usable is +/- $0.097.10^{-3}$ dB/10ns (=0.1/1024), which is also **0.009765625 dB/us**. The "sharpest" slope is +/- 32767 times this value, which is **+/- 319.99 dB/us.**

In the following example, "+0.1 dB/10ns" means that the applied gain takes +0.1 dB each 10ns. Within 9 CLK periods, the gain reaches +0.9dB. The gain limit reached is bound to Beam Gain and Beam Correction since we have the following restriction: **0 dB ≤ Beam Gain + Beam Correction + DAC Gain reached≤ +80 dB**.
The following figure shows an example of DAC with the following setup:



DAC Count	4			
DAC TOF (ns)	60	140	220	300
DAC Slope (dB/10ns)	0.00	+1.00	-1.00	0.00

figure.3 Example of DAC
(The A-scan signal is represented as a periodic signal for ease of explanation)

3.1 A-scan

The following is a listing of available A-scan parameters:

API	**HW**	**Functions**	**Data type (min, step, max)**
Rectification Defines whether the signal is RF or rectified.	All	SetAscanRectification / GetAscanRectification	enumRectification (see header "UTBasicEnum.h")
Start Defines when the A-scan starts from its synchronization reference time.	PA / FMC	SetAscanStart / GetAscanStart / Check	double min=0.0 us, step=10.0 ns, max>1s (step is 20.0 ns in the case of FMC)
Range Note [1] Defines the A-scan Range.	All	SetAscanRangeWithCount / GetAscanRangeWithCount	double min=10.0 ns, step=10.0 ns, max>1s
Point count Note [1] Defines the quantity of points that constitute the A-scan.	All	SetAscanRangeWithCount / GetAscanRangeWithCount	Long
Compression type Note [1] Defines if the compression is obtained by dropping some points ("eDecimation"), or using a peak holder between 2 resulting point positions ("eCompresion").	All	SetAscanRangeWithCount / GetAscanRangeWithCount	enumCompressionType
Maximum enable Note [2] Provides the maximized component of the compressed A-scan	All	EnableAscanMaximum / GetEnableAscanMaximum	Bool
Minimum enable (**coming soon**) Note [2] Provides the Minimized component of the compressed A-scan	All	EnableAscanMinimum / GetEnableAscanMinimum	Bool
Acquisition channel identifier #1 (optional) Note [3] Marker for the Probe (or other as you want) data of each cycle	All	SetAscanAcqIdChannelProbe / GetAscanAcqIdChannelProbe	WORD Min=0,step=1,max=65534, default=65535
Acquisition channel identifier #2 (optional) Note [3] Marker for the Scan (or other as you want) data of each cycle	All	SetAscanAcqIdChannelScan / GetAscanAcqIdChannelScan	WORD Min=0,step=1,max=65534, default=65535
Acquisition channel identifier #3 (optional) Note [3] Marker for the Cycle (or other as you want) data of each cycle	All	SetAscanAcqIdChannelCycle / GetAscanAcqIdChannelCycle	WORD Min=0,step=1,max=65534, default=65535

Table3: A-scan parameters list

Note [1]: The number of points in the A-scan (Point Count), and the A-scan Range, define the point factor according to the sampling frequency. Instead of using “SetAscanRangeWithCount”, other functions are also available: “SetAscanRange” and “SetAscanRangeWithFactor”. For more information, see the document *Software_API.pdf* paragraph “A-scan”/”FW parameters”.

Note [2]: If “eCompression” is selected, the A-scan is put through a maximization process and/or a minimization process. The maximization or minimization will have no effect if no compression is selected or if the Compression Type chosen is "eDecimation". If "maximum enable" is true, A-scan data resulting from the maximization process will be available in the output of the driver, through the member ***pBufferMax*** of the A-scan Callback function (***UINT WINAPI ProcessAcquisitionAscan_0x00010103***). And if "minimum enable" is true, A-scan data resulting from the minimization process will be available in the output of the driver, through the member ***pBufferMin***. See figure 6 for an example of MAX and MIN of the A-scan data.
Since maximization and minimization can't be applied when the type is "eDecimation" or when there is no compression (i.e. range (in ns) / 10 = point quantity), there is no need to put "minimum enable" to true. You can use only "maximum enable" = true and take the data in the member ***pBufferMax.***

Note [3]: When reading the A-scan data in the output of the Driver, there are two ways to know what the data you are reading is for. The first way is to look in ***pAscanHeader*** at the cycle and sequence numbers, the data size, the point quantity, etc. The other is to attach an "acquisition channel identifier" to each A-scan data. Three identifiers are available. Identifier values are assigned to each cycle at startup, and can be used to identify A-scan data when it is received with the A-scan Call-back function. For more explanation, see the document *Software_API.pdf*.

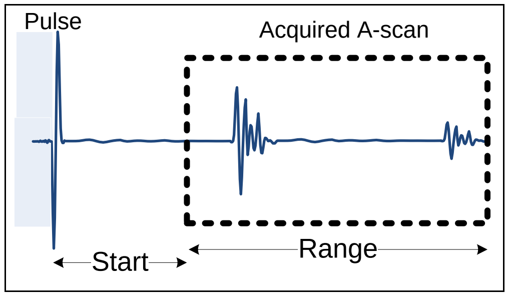


figure.4 A-scan "Start" and "Range" example

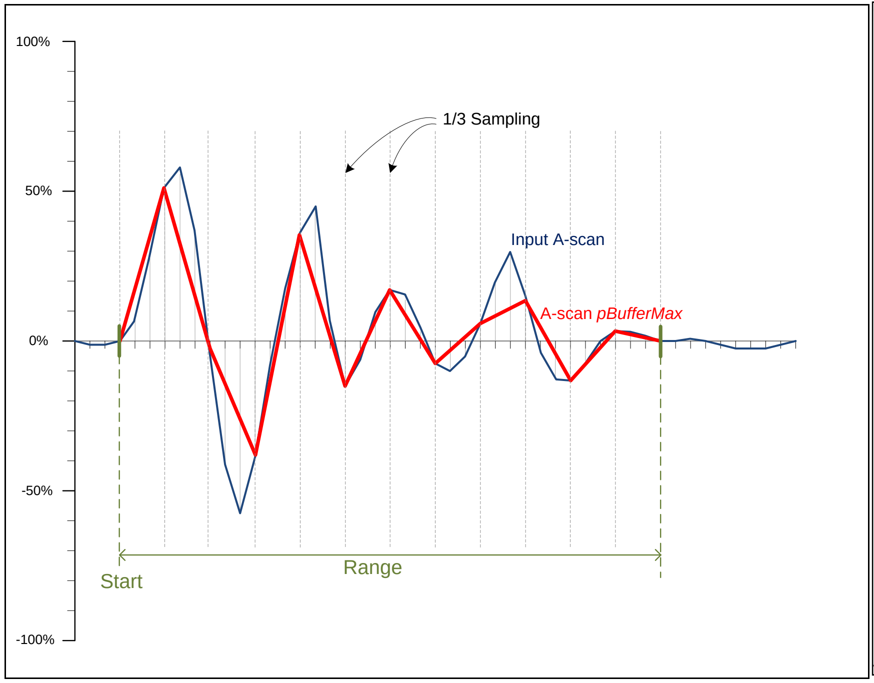


figure.5 Decimation example : **"Compression type = eDecimation"**

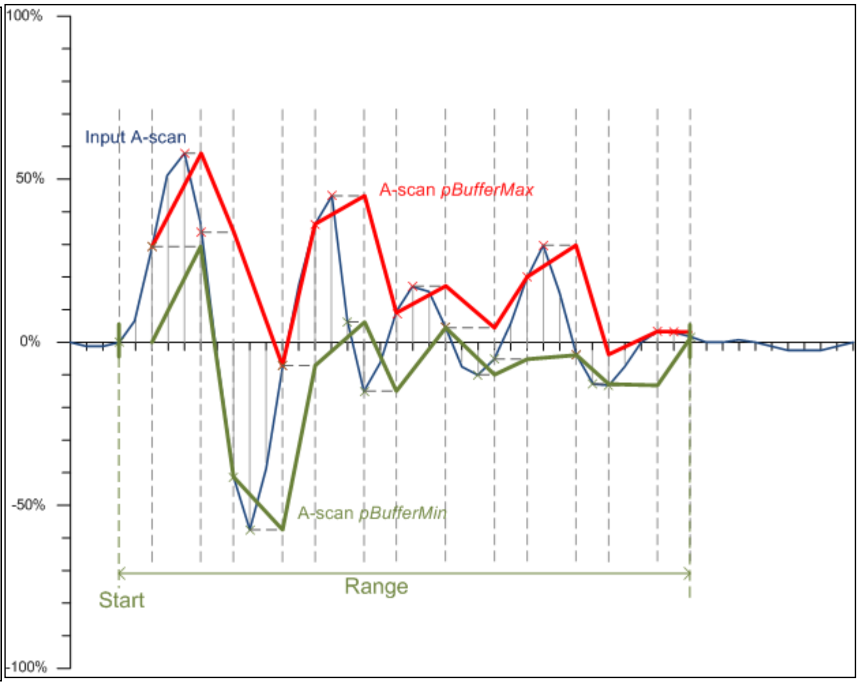


figure.6 Compression example : **"Compression type = eCompression"**

In "**decimation**", the frequency structure of the signal is kept,
but the maximum and minimum of the signal can be lost.
The resulting sampling frequency is 100/n MHz, with n=[1;256]

In "**compression**", the maximum values of the signal are kept in the Max Buffer, and the minimum values in the Min Buffer,
but there is distortion in the resulting signals.
The resulting sampling frequency is 100*n/256 MHz, with n=[1;256]

3.1.1 Tracking

Here is a list of A-scan tracking parameters:

API	Functions	Data type (min, step, max)
TrackingStartEnable Defines whether the start of the gate is synchronized by another gate detection.	SetTrackingGateStart / GetTrackingGateStart	bool
TrackingStartIndexGate Defines the gate number on which the gate is synchronized.	SetTrackingGateStart / GetTrackingGateStart	int
TrackingStartIndexCycle Defines the cycle number on which the gate is synchronized.	SetTrackingGateStart / GetTrackingGateStart	int
TrackingStopEnable Defines whether the stop of the gate is synchronized by another gate detection.	SetTrackingGateStop / GetTrackingGateStop	bool
TrackingStopIndexGate Defines the gate number on which the stop is synchronized.	SetTrackingGateStop / GetTrackingGateStop	int
TrackingStopIndexCycle Defines the cycle number on which the stop is synchronized.	SetTrackingGateStop / GetTrackingGateStop	int

The tracking feature is explained in the driver API.

3.2 C-scan

Here is a list of C-scan parameters (there are 4 gates available):

API	HW	Functions	Data type (min, step, max)
Enable Defines whether C-scan stream will be available.	PA / MC	SetGateModeThreshold / GetGateModeThreshold / CheckGateModeThreshold	bool
Mode (amplitude) Note [1] Defines which amplitude is provided from the gate detection. Not used when mode (rectification) is not "eSigned".	PA / MC	SetGateModeThreshold / GetGateModeThreshold / CheckGateModeThreshold	enumGateModeAmp
Mode (time of flight) Note [2] Defines which Time Of Flight is provided from the gate detection.	PA / MC	SetGateModeThreshold / GetGateModeThreshold / CheckGateModeThreshold	enumGateModeTof
Mode (rectification) Note [2] Defines whether the gate detection is done from the RF signal or from the rectified signal.	PA / MC	SetGateModeThreshold / GetGateModeThreshold / CheckGateModeThreshold	enumRectification
Threshold Note [2] Defines the threshold that is used for the gate detection.	PA / MC	SetGateModeThreshold / GetGateModeThreshold / CheckGateModeThreshold	double Min=0 %, step=100/256 %, max=100 %
Start Note [1 & 2] Defines the time when the gate starts from its synchronization time reference.	PA / MC	SetGateStart / GetGateStart / CheckGateStart	double Min=0.0 s, step=10.0 ns, max>1s
Stop Note [1 & 2] Defines the time when the gate ends from its synchronization time reference.	PA / MC	SetGateStop / GetGateStop / CheckGateStop / CheckGateStartStop	double Min=0.0 s, step=10.0 ns, max>1s
Amplitude acquisition channel identifier (optional) Note [3] Marker for the Amplituderesult of each cycle and each gate	PA / MC	SetGateAcqIDAmp / GetGateAcqIDAmp	WORD Min=0,step=1,max=65534, default=65535
Time of flight acquisition channel identifier (optional) Note [3] Marker for the TOF result of each cycle and each gate	PA / MC	SetGateAcqIDTof / GetGateAcqIDTof	WORD Min=0,step=1,max=65534, default=65535

Table4: C-scan parameters list

Note [1]: Refer to the following drawings that display different examples of Gate amplitude results:

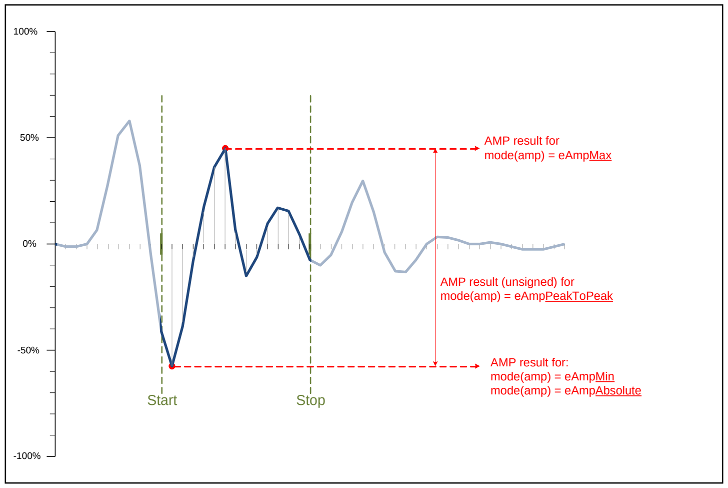


figure.7 Gate Amplitude Results example 1
*mode(rectification) is "**eSigned**"*

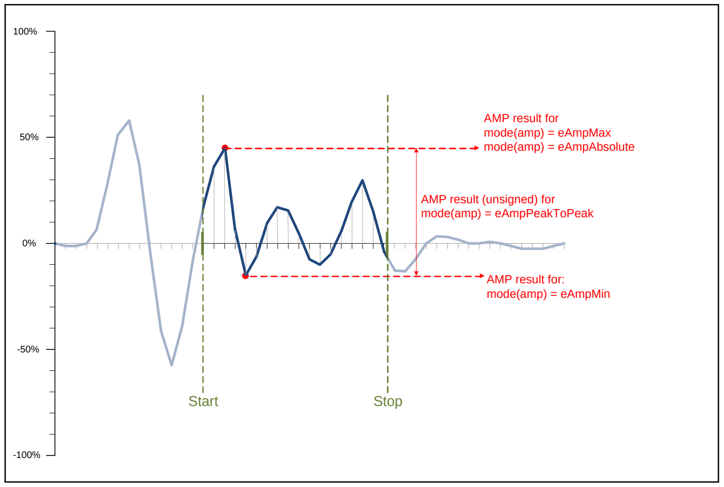


figure.8 Gate Amplitude Results example 2
*mode(rectification) is "**eSigned**"*

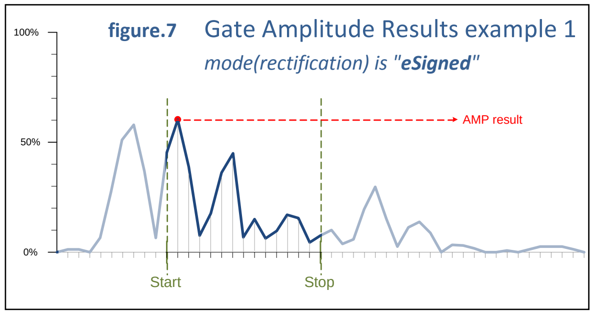


figure.9 Gate Amplitude Results example 3
*mode(rectification) is "**eUnsigned**"*

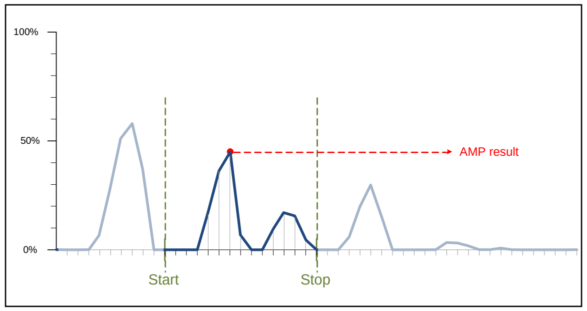


figure.10 Gate Amplitude Results example 4
*mode(rectification) is "**eUnsignedPositive**"*

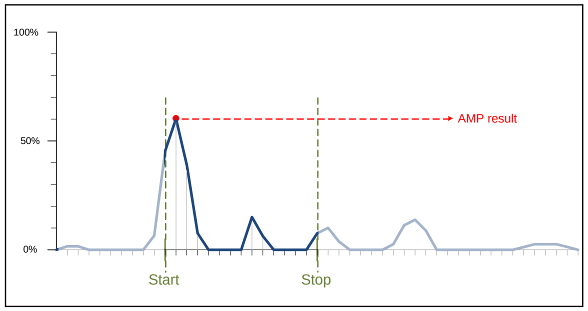


figure.11 Gate Amplitude Results example 5
*mode(rectification) is "**eUnsignedNegative**"*

Note [2]: Refer to the following drawings that display different examples of Gate time-of-flight results:
In the mode (time of flight) "eTofAmp", the TOF result is the time when the AMP result has been found (see drawings above)
The following drawings show the other 3 TOF modes.
(Due to sampling, the TOF result cannot be exactly the time where the signal crosses the threshold or the "0 axis", the result is taken on the first CLK period after the cross is detected.)

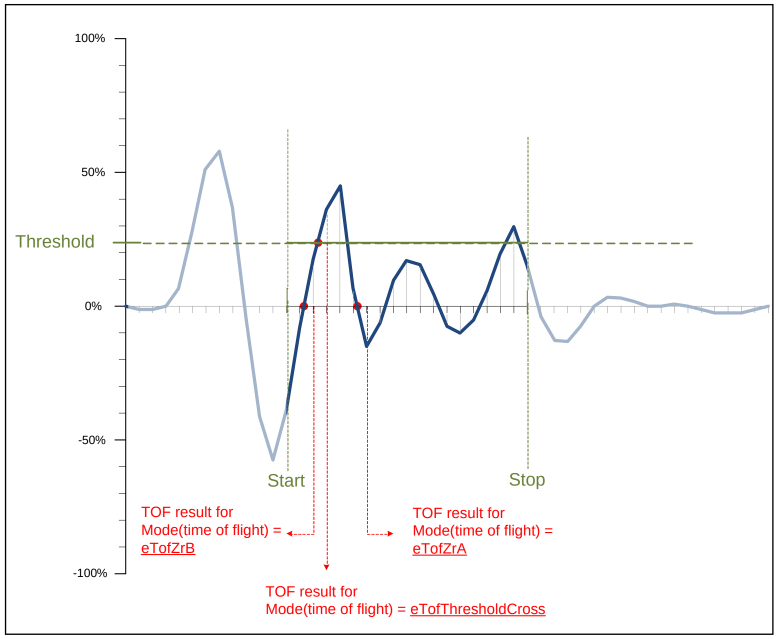


figure.12 Gate Time Of Flight Results example 1
*mode(rectification) is "**eSigned**"*

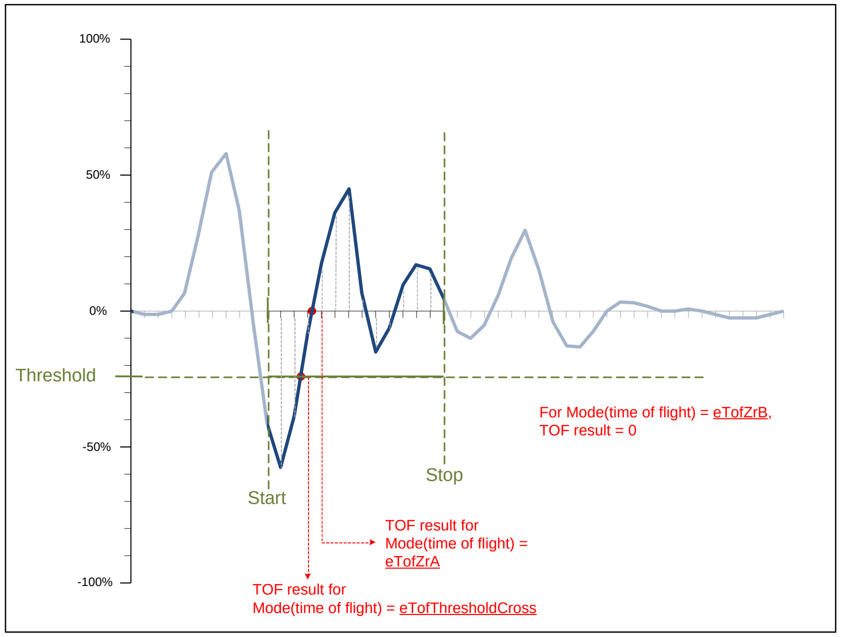


figure.13 Gate Time Of Flight Results example 2
*mode(rectification) is "**eSigned**"*

Note [3]: When reading the C-scan data in the output of the driver, there are two ways to know what the data you are reading is for. The first way is to look in the C-scan header, at the gate/cycle/sequence numbers, the data size, etc. The other way is to attach an "acquisition channel identifier" to each gate of each cycle. You can put one identifier for the Amplitude result, and another for the TOF result (if "C-scan time of flight acquisition"=true, see table1). Using "Amplitude acquisition channel identifier", you can attach a special value to the AMP result of the concerned gate of the cycle (and if "C-scan time of flight acquisition"=true, you can also use "time of flight acquisition channel identifier" to mark the TOF result of each Gate of each cycle). Identifier values are assigned to each gate and cycle at startup, and can be used to identify A-scan data when it is received with the C-scan Call-back function.

Note [4]: The Gate-tracking feature allows you to synchronize the Start and Stop of one gate with the TOF result of another. Without this feature, the reference (0) of the Start and Stop parameters is considered to be the Cycle Trigger (same reference as for the pulses and Emission/Reception delays), but with Gate tracking, the reference becomes the TOF result of another gate from the previous cycle. See figure 14 for an example of this feature. The Start and Stop of Gate 2 are referenced to the TOF result of Gate 1 (of the previous cycle).

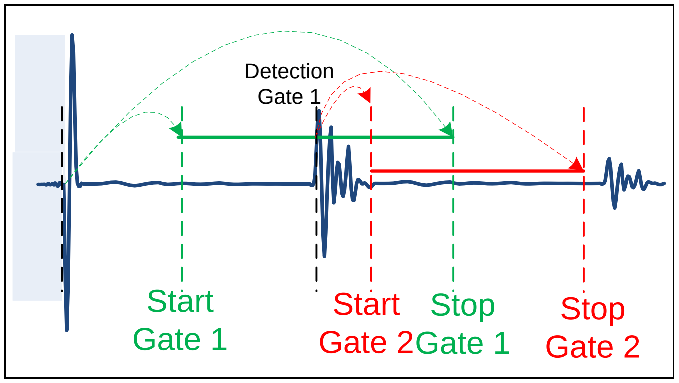


figure.14 Gate Tracking example

Start and Stop of Gate 2 are synchronized with the TOF result of the Gate 1 of the previous cycle

3.2.1 Tracking

Here is a list of gate tracking parameters:

API	Functions	Data type (min, step, max)
TrackingStartEnable Defines whether the start of the gate is synchronized by another gate detection.	SetTrackingGateStart / GetTrackingGateStart	bool
TrackingStartIndexGate Defines the gate number on which the gate is synchronized.	SetTrackingGateStart / GetTrackingGateStart	int
TrackingStartIndexCycle Defines the cycle number on which the gate is synchronized.	SetTrackingGateStart / GetTrackingGateStart	int
TrackingStopEnable Defines whether the stop of the gate is synchronized by another gate detection.	SetTrackingGateStop / GetTrackingGateStop	bool
TrackingStopIndexGate Defines the gate number on which the stop is synchronized.	SetTrackingGateStop / GetTrackingGateStop	int
TrackingStopIndexCycle Defines the cycle number on which the stop is synchronized.	SetTrackingGateStop / GetTrackingGateStop	int

The tracking feature is explained in the driver API.

3.3 Emission focusing

The following is a list of parameters and associated functions for emission focal laws (International System of Units):

API	HW	Functions	Data type (min, step, max)
Aperture Note 1 & 3 Defines which elements are enabled and which elements are disabled within the aperture.	All (for MC only for acq cycle)	SetEmissionEnable / GetEmissionEnable / SetAllElementEnable / SetElementEnable	DWORD [2] Min={0x00000000, 0x00000000} (no element), max={0xffffffff, 0xffffffff } (64 elements)
Pulse width Note 2 & 3 Defines the pulse width for all elements within the aperture, for this cycle.	All (for MC only for acq cycle)	SetEmissionWidth / SetEmissionWidths / GetEmissionWidths	float Min=0.0 s, step=4.0 ns, max>1s
Delays Note 2 & 3 Assigns the pulse delays that define the electronic focusing in transmission (emission). In other words, the electronic lens.	All (for FMC delay should be 0)	SetEmissionDelay / SetEmissionDelays / GetEmissionDelays	float Min=0.0 s, step=4.0 ns, max=32764 ns
Wedge delay Note 3 Defines the delay of the wedge (mechanic or electronic) that characterizes this aperture.	PA / FMC	SetEmissionWedgeDelay / GetEmissionWedgeDelay / CheckWedgeDelay	double Min=0.0 s, step=10.0 ns, max>1s

Table5: emission focusing parameters list

Note 1: An aperture is defined by two DWORDs. Each bit is for one probe element, so there is a maximum of 64 elements. In the case your OEM-PA device has only 16 channels, then only the first 16 elements can be used (see function "GetElementCountMax"). The LSB of the first DWORD is element 1, the MSB of the 2nd DWORD is element 64.

Note 2: To define delays (or pulse width), the aperture should be specified as an input parameter (same input parameter as for the function "SetEmissionEnable"). The main input parameter (which is an array of delays or pulse widths) is defined for each element of the aperture, not for each element of the probe.

Note 3: Refer to the following figure that shows examples of the emission parameters.

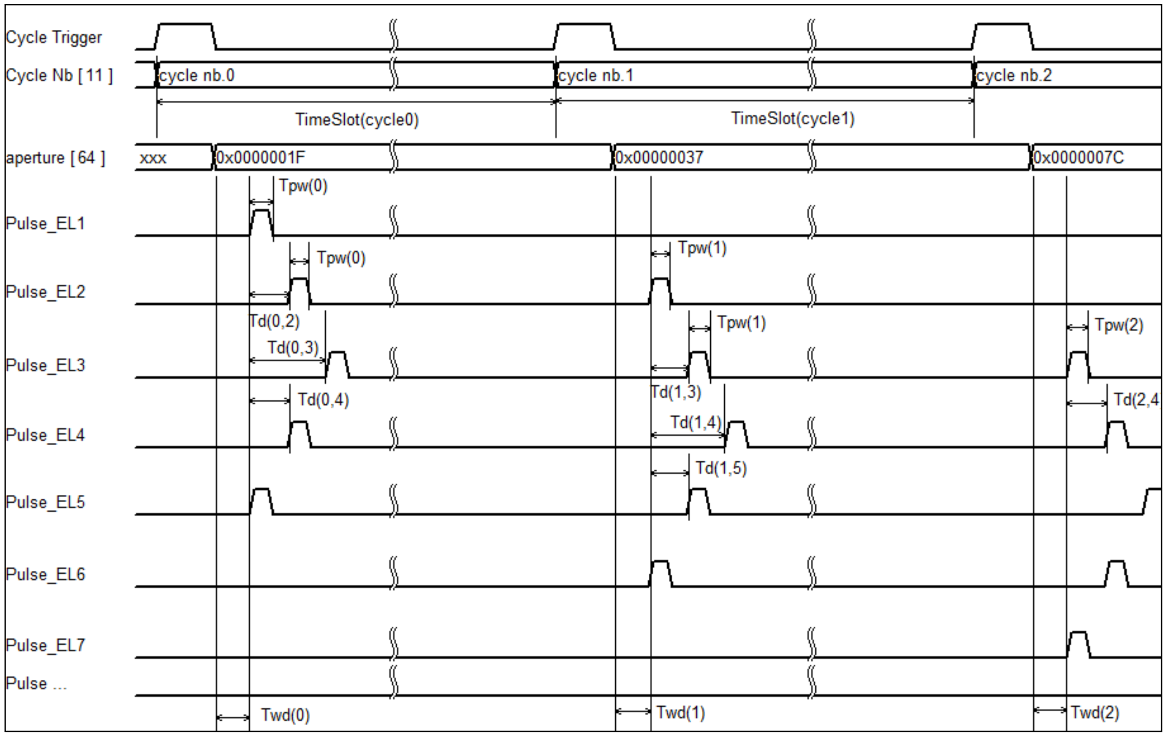


figure.15 Emission timing example.

Example of the 3 first cycles of a linear scan with an aperture of 5 elements and a step of 1 element.

Tpw([c]): *Pulse Width, cycle [c]*

Td([c],[e]): *Delay, cycle [c], element [e]*

Twd([c]): *Wedge Delay, cycle [c]*

3.4 Reception focusing

The following is a list of parameters and associated functions for the reception focal law (International System of Units):

API	HW	Functions	Data type (min, step, max)

Aperture Note 1 & 5
Defines which elements are enabled and which element are disabled within the aperture.
	All
	(Note 8)
		SetReceptionEnable /
		GetReceptionEnable /
		SetAllElementEnable /
		SetElementEnable
			DWORD [2]
			Min={0x00000000, 0x00000000}
			(no element), max={0xffffffff, 0xffffffff }
			(64 elements)

Gain Note 2 & 6 Defines the normalization gain. This is an individual gain for each element that allows you to normalize the probe sensitivity.	PA / FMC (Note 8)	SetReceptionGain / SetReceptionGains / GetReceptionGains	float
Focusing focal count Note 3 & 7 Defines the number of focal points used inside the DDF delay patterns.	PA	SetReceptionFocusing/ GetReceptionFocusing	int Min=1, step=1, max=63
Focusing time of flight Note 3 & 7 Defines the Time Of Flight that triggers the switch of the delay patterns.	PA	SetReceptionFocusing / GetReceptionFocusing	double Min=0.0 s, step=20.0 ns, max=655.34 us
Dynamic Depth Focusing (DDF) enable Note 7. Defines if the DDF will be active or if only the first delay pattern will be used as a static delay pattern for this cycle.	PA	EnableDDF / GetEnableDDF	bool
Delays (note 4) Note 4 & 7 Defines the delay for each element and for each focal point.	PA / FMC (Note 8)	SetReceptionDelay / SetReceptionDelays / GetReceptionDelays	float Min=0.0 s, step=5.0 ns, max=40955ns
Wedge delay Note 7 Defines the delay of the wedge (mechanic or electronic) that characterizes this aperture.	PA / FMC	SetReceptionWedgeDelay / GetReceptionWedgeDelay / CheckWedgeDelay	double Min=0.0 s, step=10.0 ns, max>1s

Table6: reception focusing parameters list

Note 1: An Aperture is defined by two DWORDs. Each bit is for one probe element, so there is a maximum of 64 elements. In the case your OEM-PA device has only 16 channels, the first 16 elements can be used (see function “GetElementCountMax”). The LSB of the first DWORD is element 1, the MSB of the second DWORD is element 64.

Note [2]: To define gain, the aperture should be specified as an input parameter (same input parameter as for the function "SetReceptionEnable"). The main input parameter (which is an array of gain) is defined for each element of the aperture, not for each element of the probe.

Note [3]: For reception focusing, you can use a single focal law (standard focusing) or more (DDF). For standard focusing, the focal count is 1 and for DDF the focal count is greater than 1. In this case you have to specify the time of flight of each focal law with the "SetReceptionFocusing" function and you need to enable it ("EnableDDF" function). In the case of standard focusing the time of flight value of the single focal law is not significant.

Note [4]: Three parameter formats are available: scalar, array, and callback functions. For scalar input parameter you must specify the element number and focal law number. For callback input parameters, reference the example in "CustomizedAPI.cpp" and "APISamples.cpp". For array input parameters, the order is given by the table below.

Array index (input parameter "pfDelay" of function "SetReceptionDelays")	**Element number** (assumption : aperture has 4 elements)	**Focal law index** (assumption : 4 focal laws)
0	0	0
1	1	0
2	2	0
3	3	0
4	0	1
5	1	1
6	2	1
7	3	1
8	0	2
9	1	2
10	2	2
11	3	2
12	0	3
13	1	3
14	2	3
15	3	3

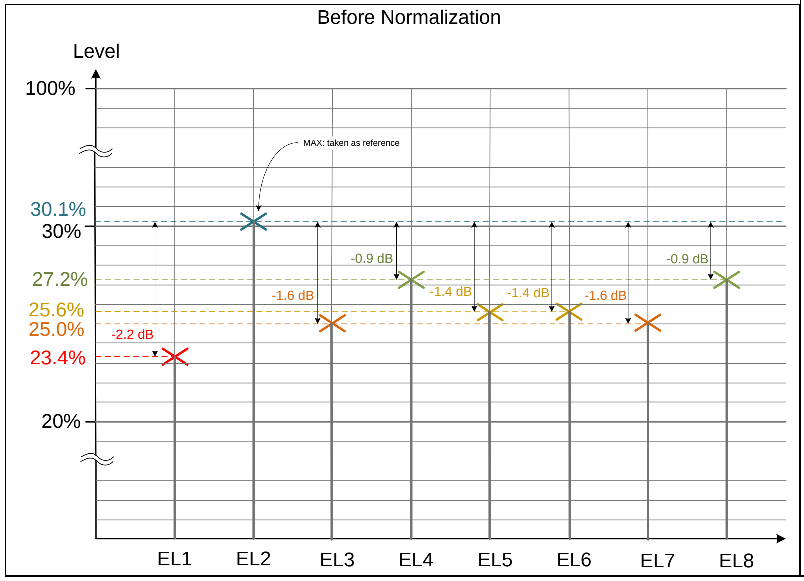
Table7: element order of array input parameter

Note [5]: Refer to the following table that describes some example of aperture definition.

Probe Element Number:	EL32	EL31	…	EL8	EL7	EL6	EL5	EL4	EL3	EL2	EL1	Aperture (first DWORD)
Cycle 0	-	-	-	-	-	-	-	✓	✓	✓	✓	0x000F
Cycle 1	-	-	-	-	-	-	✓	✓	✓	✓	-	0x001E
Cycle 2	-	-	-	-	-	✓	✓	✓	✓	-	-	0x003C
Cycle 3	-	-	-	-	✓	✓	✓	✓	-	-	-	0x0078
Cycle 4	-	-	-	✓	✓	✓	✓	-	-	-	-	0x00F0
…	…	…	…	…	…	…	…	…	…	…	…	…

Table8: reception aperture - example of a linear scan with an aperture width of 4 elements and a step of 1 element
(The system used for the example has 32 channels so only the first DWORD is used.)

Note [6]: Refer to the following drawings that describe an example of an aperture sensitivity distribution before and after normalization with the gain table.



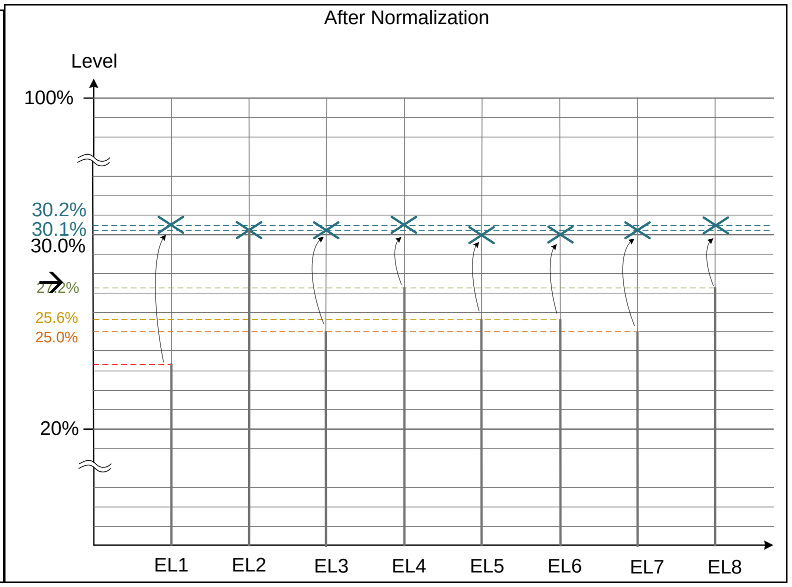


figure.16 Normalization Example – Before Normalization
Example on 8 elements of a PA probe

figure.17 Normalization Example – After Normalization
Example on 8 elements of a PA probe

Element number	**EL1**	**EL2**	**EL3**	**EL4**	**EL5**	**EL6**	**EL7**	**EL8**	**…**
Gain applied (dB)	2.2	0.0	1.6	0.9	1.4	1.4	1.6	0.9	…

Table9: Elements normalization example – Gain Table

Note [7]: Refer to the following diagram that describes the electronic lens in DDF mode (delay pattern along the aperture for each focal point of the DDF).

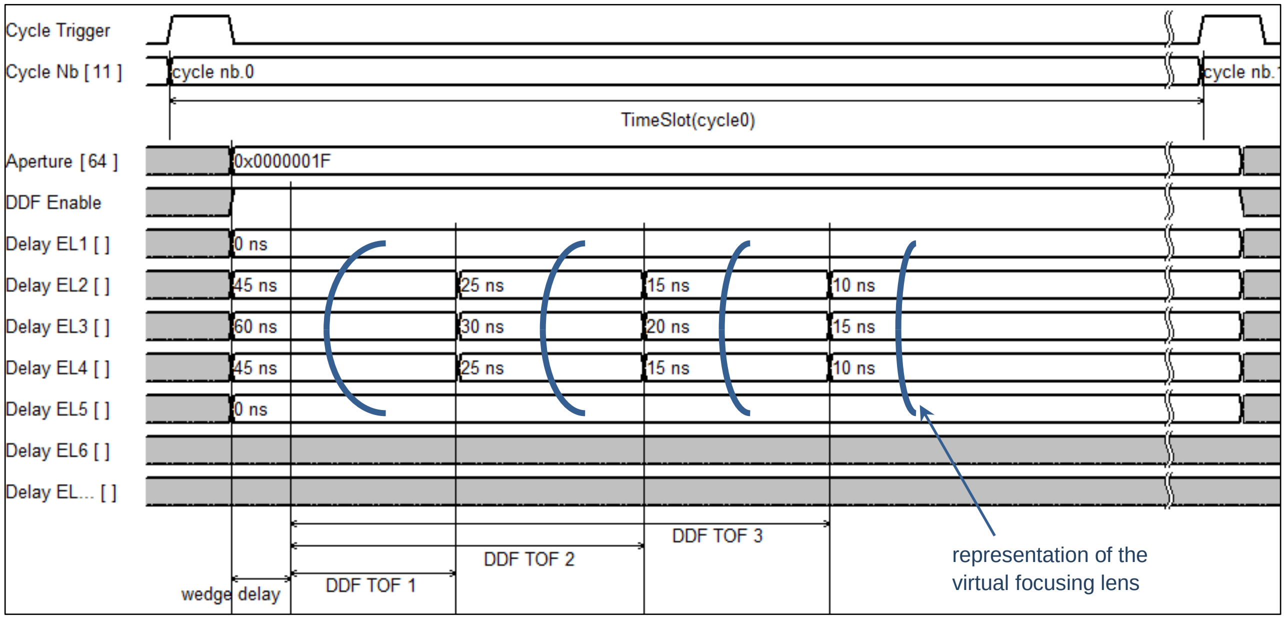


figure.18 Reception – Dynamic Depth Focusing example
*Example of the first cycle of a linear scan with an aperture of 5 elements and a **Focusing Focal Count of 4** (DDF)*
(*The focusing angle is 0°C)*

Note [8]: For FMC: it is possible to define the delay and the gain for one element only, then those values are applied to all reception elements (define by ElementStart and ElementStop). For FMC ,Delay should be 0.

4 Annex 1 - Firmware Global Diagram

See the following diagram that describes the main parts of OEM-PA Firmware. On the top and bottom of the diagram, you will find the parameters listed in this document, and on the right, the "outputs" of OEM-PA Driver.

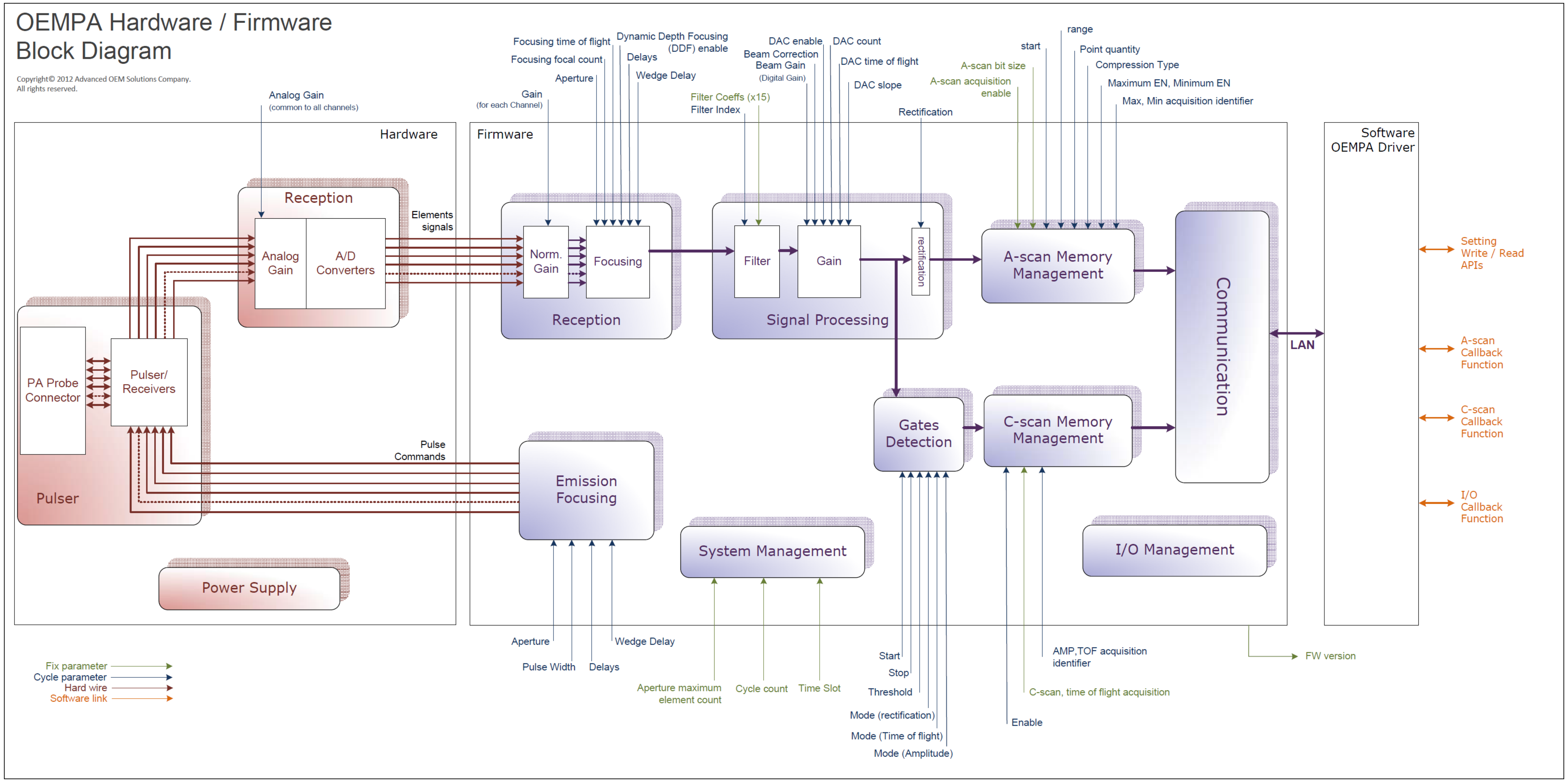


figure.19 OEM-PA Firmware – Global Diagram