# Release Note

2018-Apr-13

Yanchen Lu

## armv7a

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Release date | Library archive name | Ver. | Note |
| 1 | 2017-Oct-20 | libtutuClear\_152\_20171020.a | 1.5.2 | First release for R16 integration |
| 2 | 2017-Nov-06 | libtutuClear\_161\_20171106.a | 1.6.1 | - Support stereo AEC  - Small-scale algorithm update  - Release DOA API |
| 3 | 2017-Nov-17 | libtutuClear\_162\_20171117.a | 1.6.2 | - Fix issue that no wakeup after 3 hours  - Speed optimization for armv7a |
| 4 | 2017-Nov-27 | libtutuClear\_163\_R16\_20171127.a | 1.6.3 | Remove Small-scale algorithm update on Ver.1.6.1 |
| 5 | 2017-Nov-29 | libtutuClear\_163\_R16\_20171129\_20b.a | 1.6.3 | Accept 32-bit PCM input; previously accept 16-bit PCM input |
| 6 | 2017-Dec-05 | libtutuClear\_163\_R16\_20171205\_W32.a | 1.6.3 | Algo update for 90 dBC AEC test |
| 7 | 2017-Dec-06 | libtutuClear\_164\_R16\_20171206.a | 1.6.4 | Support 16-bit and 32-bit input/output port simultaneously |
| 8 | 2017-Dec-07 | libtutuClear\_165\_R16\_20171207.a | 1.6.5 | AEC performance upgrade with the penalty of additional computational complexity, not fully optmized |
| 9 | 2018-Jan-25 | libtutuClear\_166\_R16\_20180125.a | 1.6.6 | - Support ASR/VOIP dual output  - Speed optimization for armv7a |
| 10 | 2018-Feb-09 | libtutuClear\_166\_R18\_armv7a\_20180209.a | 1.6.6 | - Display debug mesg  - Used at R18 |
|  |  |  |  |  |

## arm64-v8a

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Release date | Library archive name | Ver. | Note |
| 1 | 2017-Nov-18 | libtutuClear\_162\_armv8a\_20171118.a | 1.6.2 | Frist release for R18 integration |
| 2 | 2017-Nov-28 | libtutuClear\_162\_R18\_20171128.a | 1.6.3 | Sync to R16 Ver.1.6.3 |
| 3 | 2017-Nov-30 | libtutuClear\_163\_R18\_20171130\_20b.a | 1.6.3 | Accept 32-bit PCM input; previously accept 16-bit PCM input |
| 4 | 2017-Dec-06 | libtutuClear\_164\_R18\_20171206.a | 1.6.4 | Support 16-bit and 32-bit input/output port simultaneously |
| 5 | 2017-Dec-07 | libtutuClear\_165\_R18\_20171207.a | 1.6.5 | AEC performance upgrade with the penalty of additional computational complexity |
| 6 | 2018-Jan-31 | libtutuClear\_166\_R18\_20180131.a | 1.6.6 | - Support ASR/VOIP dual output  - Speed optimization for armv7a |
| 7 | 2018-Apr-11 | libtutuClear\_167\_R18\_20180411.a | 1.6.7 | Support Amazon ESP |
| 8 | 2018-Apr-13 | libtutuClear\_168\_R18\_20180413.a | 1.6.8 | Support Alexa wakeup function |
|  |  |  |  |  |

# Tuning guide

## TUTU\_PARAM\_SYS.uw32OpMode

|  |  |
| --- | --- |
| Bit | Mode description |
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 | ASR assist |
| 5 |  |
| 6 |  |
| 7 | Multiple channel I/O data export |
| 8 |  |
| 9 |  |
| 10 | Stereo AEC |
| 11 |  |
| 12 |  |
| 13 | Generating two channel line-out signal. First channel is for speech recognition assist application. Second channel is for two-way communication application |
| 14 | Generating spatial perception signal at third line-out channel for sound energy based auditory scene analysis |
| 15 |  |
| 16~31 | Reserved |

## TUTU\_PARAM\_SYS.uw32FuncMode

|  |  |
| --- | --- |
| Bit | Mode description |
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 | Acoustic echo cancellation (AEC) |
| 7 | Array noise suppression |
| 8 |  |
| 9 |  |
| 10 | Auto gain control (AGC). Design for two-way communication |
| 11 | Dynamic range control (DRC) |
| 12 | High-pass filter (HPF) |
| 13 |  |
| 14 | Directional of arrival (DOA) |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 | Alexa keyword wakeup |
| 21 |  |
| 22 |  |
| 23 |  |
| 24 |  |
| 25 |  |
| 26 |  |
| 27 |  |
| 28 |  |
| 29 |  |
| 30 |  |
| 31 |  |

## How to enable the support for Amazon ESP (Echo Spatial Perception)?

2018-Apr-11

Amazon ESP requires signal with acoustic echo removed and no gain effect applied.

tutuClear provides such ESP signal at third output channel, where first channel contains signal for speech recognition and second channel contains signal for two-way communication. These three channel signals are non-interleaved aligned in the buffer (please ensure enough memory is allocated to this buffer!) assigned as the fourth input argument of TUTUClear\_OneFrame(…, W16 \*ptLOut, …) or TUTUClear\_OneFrame\_32b(…, W32 \*pw32Lout, …).

Two configuration steps are required to enable ESP signal output:

1. While tutuClear initialization, HSB 8 of TUTUClearConfig\_t.uw16MaxNumOfMic needs to be set as 3, E.g. TUTUClearConfig\_t.uw16MaxNumOfMic = 0x0303, meaning three channels of microphone input and three channels of signal output. The third output channel contains signal for Amazon ESP.
2. While tutuClear operation mode setup, the 15-th bit TUTU\_PARAM\_SYS.uw32OpMode needs to be asserted. E.g. TUTU\_PARAM\_SYS.uw32OpMode = 0x00006410, meaning enable ASR assist mode, stereo AEC mode, generating two-way communication signal at 2nd output channel and generating Amazon ESP signal at 2nd output channel.

## How to make Alexa wakeup function work?

2018-Apr-13

Alexa wakeup voice model is embedded in the delivered library to avoid additional integration effort. If one wants to enable wakeup function, assert the 21th bit of TUTU\_PARAM\_SYS.uw32FuncMode to make it happen.

If Alexa keyword is detected, TUTUClearStat\_t.w16VtrgrFlg would be become unzero for one processing frame at the end of Alexa keyword. TUTUClearStat\_t is the fifth argument of TUTUClear\_OneFrame(…, TUTUClearStat\_t \*) or TUTUClear\_OneFrame\_32b(…, TUTUClearStat\_t \*). It is an output variable. Following pseudo code can be an example about how to signal keyword detection.

if (tTUTUClearStat.w16VtrgrFlg != 0) printf("wakeup\_word:\"alexa\" detected.\n");

## How to make localization (DOA) function work?

2018-Apr-13

You have to do three things to complete a localization task:

1. Assert the 15th bit of TUTU\_PARAM\_SYS.uw32FuncMode

2. Setup DOA parameters **TUTUClearParam\_t.tTUTUDOAParam**. Please refer to following tuning guide

3. TUTUClearStat\_t.w16Resrv5 gives immediate sound source localization in radian of signed Q3.7 format (-pi to pi). TUTUClearStat\_t is the fifth argument of TUTUClear\_OneFrame(…, TUTUClearStat\_t \*) or TUTUClear\_OneFrame\_32b(…, TUTUClearStat\_t \*). It is an output variable. Following pseudo code can be an example about how to signal localization result from 0 to 359 degree.

double dbDOA = ((double)tTUTUClearStat.w16Resrv5\*180.0) / (128.0\*3.1415926);

if (dbDOA < 0.0) dbDOA += 360.0;

printf("wakeup\_angle: %6.2f\n", dbDOA);

Following paragraphs are the tuning guide of sound source localization using DOA engine and setting example for R18 EVM. Array microphone geometry needs to be configured before obtaining accurate voice source localization result from data structure TUTUClearStat\_t. The geometry configuration is done in two stages. First stage defines the relationship between microphones and horizontal plane. Second stage defines the relationship among microphones.

1. Configure aw16MicPosX and aw16MicPosY in **TUTUClearParam\_t.tTUTUDOAParam**

A cartesian coordinate is put onto the microphone array to define the horizontal space. Fig. 1 is an example of microphone array with six microphones which are uniformly distributed onto a circle with a radius of 40 millimeters. The x-axis crosses the y-axis at circle origin. X-axis positive points to the zero degree of horizontal space (azimuth dimension).



**Figure 1. An array with six microphones which are uniformly distributed on a circle**

After coordinate system for the microphone array is defined, position of each microphone in this coordinate system becomes the value to parameters, aw16MicPosX and aw16MicPosY. aw16MicPosX is microphone position at x-axis in the unit of millimeter; aw16MicPosY is microphone position at y-axis in the unit of millimeter. Taking Fig. 1 as an example, aw16MicPosX and aw16MicPosY should be input with,

|  |  |
| --- | --- |
| aw16MicPosX[0] | 0 |
| aw16MicPosX[1] | 35 |
| aw16MicPosX[2] | 35 |
| aw16MicPosX[3] | 0 |
| aw16MicPosX[4] | -35 |
| aw16MicPosX[5] | -35 |
| aw16MicPosX[6] | 0 |
| aw16MicPosX[7] | 0 |
| aw16MicPosY[0] | 40 |
| aw16MicPosY[1] | 20 |
| aw16MicPosY[2] | -20 |
| aw16MicPosY[3] | -40 |
| aw16MicPosY[4] | -20 |
| aw16MicPosY[5] | 20 |
| aw16MicPosY[6] | 0 |
| aw16MicPosY[7] | 0 |

**Table 1. Parameter values of aw16MicPosX and aw16MicPosY for the array in Fig. 1.**

1. Configure auw16MicPairSlct in **TUTUClearParam\_t.tTUTUDOAParam**

Sound source localization requires microphones with matched characteristics. Matched microphones should be paired in a way for achieving optimal balance between localization accuracy and computational complexity. In Fig. 2, 1st, 3rd and 5th microphones are considered matched to each other and selected for localization processing.



**Figure 2. Microphone pairs, two pairs (left) or three pairs (right), for localization process**

At least two microphone pairs are required for sound localization. Three microphones can generate at most three different microphone pairs. The two pairing options described in Fig. 2 generate parameter values,

|  |  |  |
| --- | --- | --- |
|  | Two pairs | Three pairs |
| auw16MicPairSlct[0] | 0x0020 | 0x0020 |
| auw16MicPairSlct[1] | 0x0040 | 0x0040 |
| auw16MicPairSlct[2] | 0x0000 | 0x0042 |
| auw16MicPairSlct[3] | 0x0000 | 0x0000 |
| auw16MicPairSlct[4] | 0x0000 | 0x0000 |
| auw16MicPairSlct[5] | 0x0000 | 0x0000 |
| auw16MicPairSlct[6] | 0x0000 | 0x0000 |
| auw16MicPairSlct[7] | 0x0200 | 0x0300 |

**Table 2. auw16MicPairSlct configuration for Fig. 2**

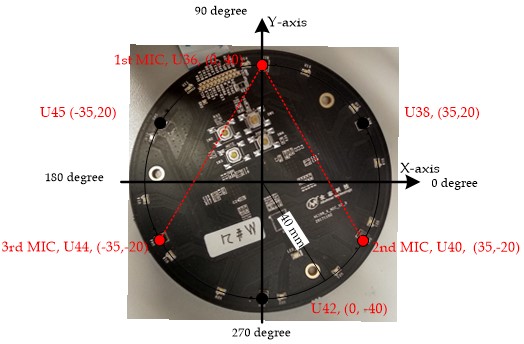
It does not matter that which microphone within a pair to be reference microphone (first 4 LSB) or paired microphone (second 4 LSB). Noticeably, microphone pair number need to be specified by b8~11 of auw16MicPairSlct[7]. Two-pair option consumes less power than three-pair option with slightly worse localization performance.

When constructing microphone pairs, it is important that a single network is generated within the microphone array. In Fig. 3, 1st, 3rd, 4th and 5th microphones are selected to involve localization processing. Four microphones may arrive with various pairing combinations. If only considering two-pair, it may lead to two networks as depicted at the left part of Fig. 3. Sound localization engine requires the existence of only one network. Therefore, two-pair does not work out. Three-pair is required at least for Fig. 1 example, as depicted at the right part of Fig. 3.

**Figure 3. Two-pair choice leads to two networks (left); Three-pair choice leads to single network (right)**

Following example shows how to setup **TUTUClearParam\_t.tTUTUDOAParam** to achieve localization function with using only 3 microphones on R18 AC108\_8\_MIC\_V2\_0 daughter board.



TUTU\_PARAM\_DOA.aw16MicPosX[0] 0

TUTU\_PARAM\_DOA.aw16MicPosX[1] 35

TUTU\_PARAM\_DOA.aw16MicPosX[2] -35

TUTU\_PARAM\_DOA.aw16MicPosX[3] 0

TUTU\_PARAM\_DOA.aw16MicPosX[4] 0

TUTU\_PARAM\_DOA.aw16MicPosX[5] 0

TUTU\_PARAM\_DOA.aw16MicPosX[6] 0

TUTU\_PARAM\_DOA.aw16MicPosX[7] 0

TUTU\_PARAM\_DOA.aw16MicPosY[0] 40

TUTU\_PARAM\_DOA.aw16MicPosY[1] -20

TUTU\_PARAM\_DOA.aw16MicPosY[2] -20

TUTU\_PARAM\_DOA.aw16MicPosY[3] 0

TUTU\_PARAM\_DOA.aw16MicPosY[4] 0

TUTU\_PARAM\_DOA.aw16MicPosY[5] 0

TUTU\_PARAM\_DOA.aw16MicPosY[6] 0

TUTU\_PARAM\_DOA.aw16MicPosY[7] 0

TUTU\_PARAM\_DOA.auw16MicPairSlct[0] 0x0010

TUTU\_PARAM\_DOA.auw16MicPairSlct[1] 0x0020

TUTU\_PARAM\_DOA.auw16MicPairSlct[2] 0x0000

TUTU\_PARAM\_DOA.auw16MicPairSlct[3] 0x0000

TUTU\_PARAM\_DOA.auw16MicPairSlct[4] 0x0000

TUTU\_PARAM\_DOA.auw16MicPairSlct[5] 0x0000

TUTU\_PARAM\_DOA.auw16MicPairSlct[6] 0x0000

TUTU\_PARAM\_DOA.auw16MicPairSlct[7] 0x0200

TUTU\_PARAM\_DOA.auw16Resrv[0] 0x0000

TUTU\_PARAM\_DOA.auw16Resrv[1] 0x0000

TUTU\_PARAM\_DOA.auw16Resrv[2] 0x0000

TUTU\_PARAM\_DOA.auw16Resrv[3] 0x0000

TUTU\_PARAM\_DOA.auw16Resrv[4] 0x0000

TUTU\_PARAM\_DOA.auw16Resrv[5] 0x0000

TUTU\_PARAM\_DOA.auw16Resrv[6] 0x0000

TUTU\_PARAM\_DOA.auw16Resrv[7] 0x0000