Alango Hearing Enhancement Package (HEP) for CSR BlueCore5MM and CSR 8670

Integration Manual

HEP rev. 38+
Document revision 1.00 (May 11, 2015)

Table of contents

1. Purpose of this document

This document describes integration of Alango Hearing Enhancement Package (HEP) library for Kalimba DSP into CSR 8670 software project using ADK 3.5, ADK 3.0, ADK 2.5, ADK 2.0, ADK 1.1, ADK 1.0.

2. HEP delivery package

Alango HEP DSP library delivery package for CSR 8670 platform consists of:

- Library file containing HEP binary code compiled for Kalimba DSP and ready for linking (KHEP.KAP)[1].
- HEP Configurator – special GUI utility for easy HEP parameters tuning
- This integration manual
- HEP Technology Guide – a document describing HEP technology in details (TBD)
- HEP PSKEY structure specification – a document, providing details on HEP parameters structure in Persistent Store of CSR 8670.
- Alango Logger – GUI software tool allowing to record HEP input and output signals on PC directly from the 8670 Kalimba DSP (requires special hardware module)

Alango HEP library contains the following DSP and functional blocks operating at 16 KHz sampling frequency:

- Acoustic Feedback Canceller (AFC) with Adaptive Filter (AF) and Frequency Shifter (FSH)
- Adaptive Directional Microphone (ADM) for acoustic beam-forming (option)
- Binaural processing (option)
- Noise Suppressor (NS)
- Automatic Gain Control (AGC)
- Frequency equalizer (EQ)
- Noise Gate (NG)
- Dynamic Range Compressor (DRC)
- additional features:
    - real-time digital signals logger (refer to section 8)
    - special debugging modes with auditory messaging

HEP library may also contain additional audio processing blocks not mentioned above. Refer to Figure 1 illustrating a flowchart of HEP inside CSR 8670.

[1] KAP file should correspond to CSR SDK/ADK used.

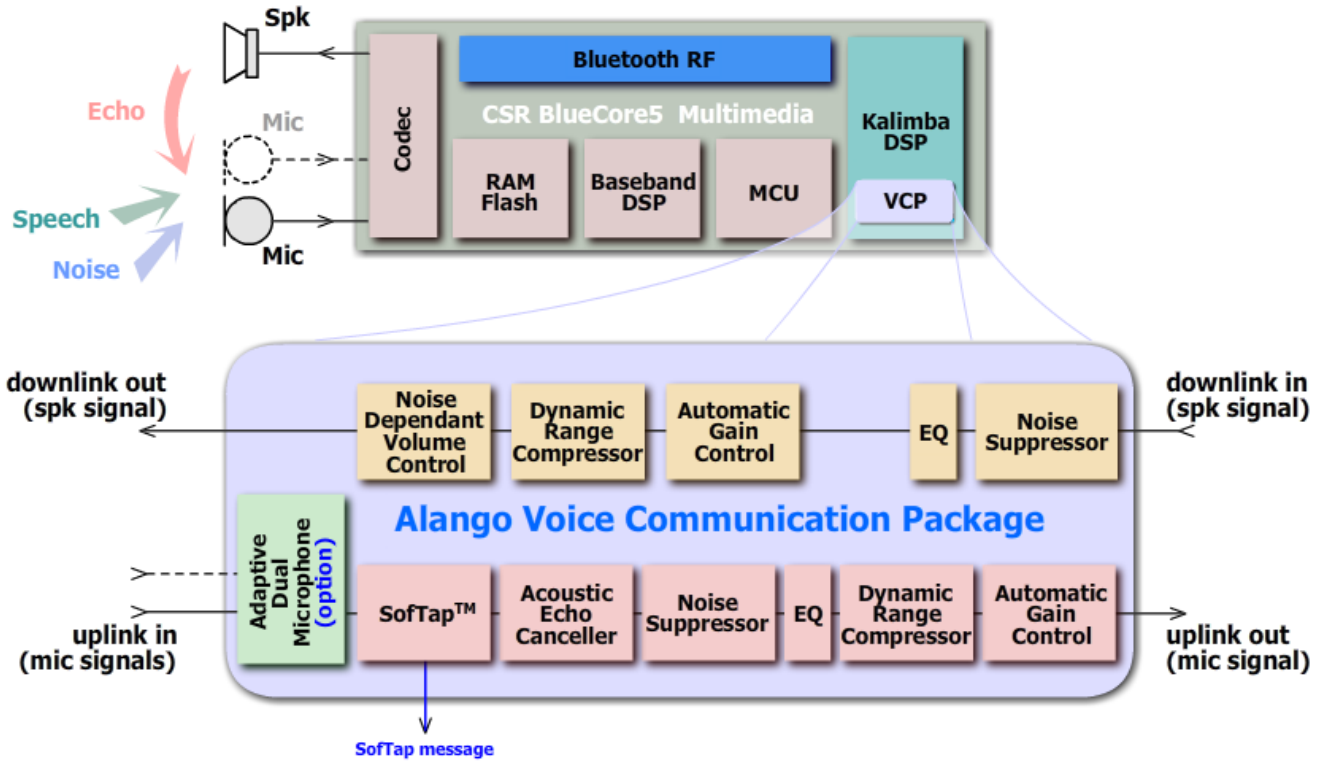

Figure 1. Alango HEP and Alango VCP inside CSR 8670

Depending on HEP library version, some of the processing blocks may be included or excluded from a particular copy of the library.

3. Types of HEP library

There are four types of HEP library available:

- Monaural HEP – for mono wearables
- Monaural HEP with ADM - for multi-microphone mono wearables
- Binaural HEP – for stereo wearables
- Binaural HEP with ADM – for multi-microphone stereo wearables

Important! Different CSR SDK may require differently compiled HEP libraries.

4. HEP license information

HEP is a commercial DSP software product. HEP licensing terms are to be negotiated with Alango. HEP library may require a special **Alango protection IC** or a **software license key**.

A) Library requiring an Alango protection IC can be "**production**" and "**demo**".

The demo library produces short beep at the start of operation, then operates normally during 7 minutes and mutes the audio afterwards.

The production library operates in a fully functional mode only when a valid protection IC is detected. If the protection IC is not detected, HEP produces constant beep immediately after the start.

B) Library protected with software license key requires that the key, stored in PSKEY, corresponds to a specific Bluetooth MAC-addresses range.

With a valid license key that corresponds to the MAC-address of the device running HEP, HEP operates in a fully functional mode.

With a special "demo" license, HEP works in demonstration mode allowing 7 (seven) minutes of normal operation without interruptions; after this period of time the audio channel is fully **muted**.

The demo license data for HEP is as follows:

0123 4567 890A BCDE F012 3456 7890 ABCD

This demo license is valid and can be used only within the following Bluetooth addresses range:

0002 5B 00A5A5 -- 0002 5B 00A5E5 (64 different addresses)

If no license is found, or license data is incorrect for the particular Bluetooth address, or if the demo license is used with inappropriate Bluetooth address, HEP produces constant beep via RX channel immediately after the start (SCO opening).

5. Integration of HEP

HEP library is intended for integration into applications developed with ADK 3.5, ADK 3.0, ADK 2.5, ADK 2.0, ADK 1.1., ADK 1.0. HEP is delivered in a form of .KAP file ready for linking.

Exemplary project for ADK 3.5, integrating HEP, is supplied with the HEP package.

6. Messages support

HEP communicates with VM using messaging mechanism.

6.1 Boot sequence

A high level description of the boot sequence is:

1. VM starts HEP using the KalimbaLoad function.
2. VM receives HEP_READY_MSG and responds by:
    a. sending HEP_LOADPARAMS_MSG*.
    b. sending the HEP_VOLUME_MSG message.
    c. connecting the streams to the DSP.

* Kalimba responds with HEP_PARAMS_LOADED_MSG.

Volume is handled with HEP in the following manner:

1. VM sends the HEP_VOLUME_MSG.
2. HEP does internal calculations and sends the HEP_CODEC_MSG.
3. VM receives HEP_CODEC_MSG and sets CODEC gains.

6.2 Messages description

6.2.1 Messages to Kalimba

KalimbaSendMessage(HEP_LOADPARAMS_MSG, PSkeyNumL, PSkeyNumR, 0, 0);
This message says Kalimba to load parameters for the left ear from PSkey number PSkeyNumL.
This message says Kalimba to load parameters for the right ear from PSkey number PSkeyNumR.

KalimbaSendMessage(HEP_VOLUME_MSG, 0, 0, VolumeL, VolumeR);
Send Volume to Kalimba for internal calculation.
VolumeL – volume for the left ear, VolumeR – volume for the right ear

KalimbaSendLongMessage(HEP_PARAMS_ID_MSG, sizeof(MsgParams), (const uint16*)MsgParams);
This message tells the Kalimba to change parameters specified in the MsgParams structure.

Supported values:
<TBD>

6.2.2 Messages from Kalimba

HEP_READY_MSG

Indicates that HEP has powered up and is ready to receive messages from the VM application.

HEP_CODEC_MSG

This message tells the VM application which CODEC gain values to use with the internal CODEC.
@param DacGainL (0)
@param DacGainL (1)
@param AdcGainL (2), if bit 15 is high the mic preamp should be enabled.
@param AdcGainR (3), if bit 15 is high the mic preamp should be enabled.

HEP_PARAMS_LOADED_MSG

Indicates that the HEP parameters have been loaded.
@param Status (0), Status is set to “1” if the parameters are correct otherwise wrong.
@param Revision (1), Current KHEP revision.

7. HEP Configurator

HEP Configurator – is MS Windows (w32) application providing easy and intuitive way of configuring HEP parameters, managing HEP acoustic profiles and field acoustic tuning.

HEP Configurator can be used “on-line” (with CSR 8670-based device connected to PC via SPI interface using either LPT or USB) for “on–the-fly” acoustic tuning or “off-line” for HEP acoustic profiles management.

Important!

- HEP Configurator is a .NET application. Please make sure that .NET framework is installed in your system before running this software.
- For on-line operation, BlueSuite 2.x or any CSR SDK should be installed on computer before using the utility.

HEP Configurator main screen shows a block chart of HEP DSP library with configurable parameters for each signal-processing block. The program screen of the utility is shown in Figure 2.

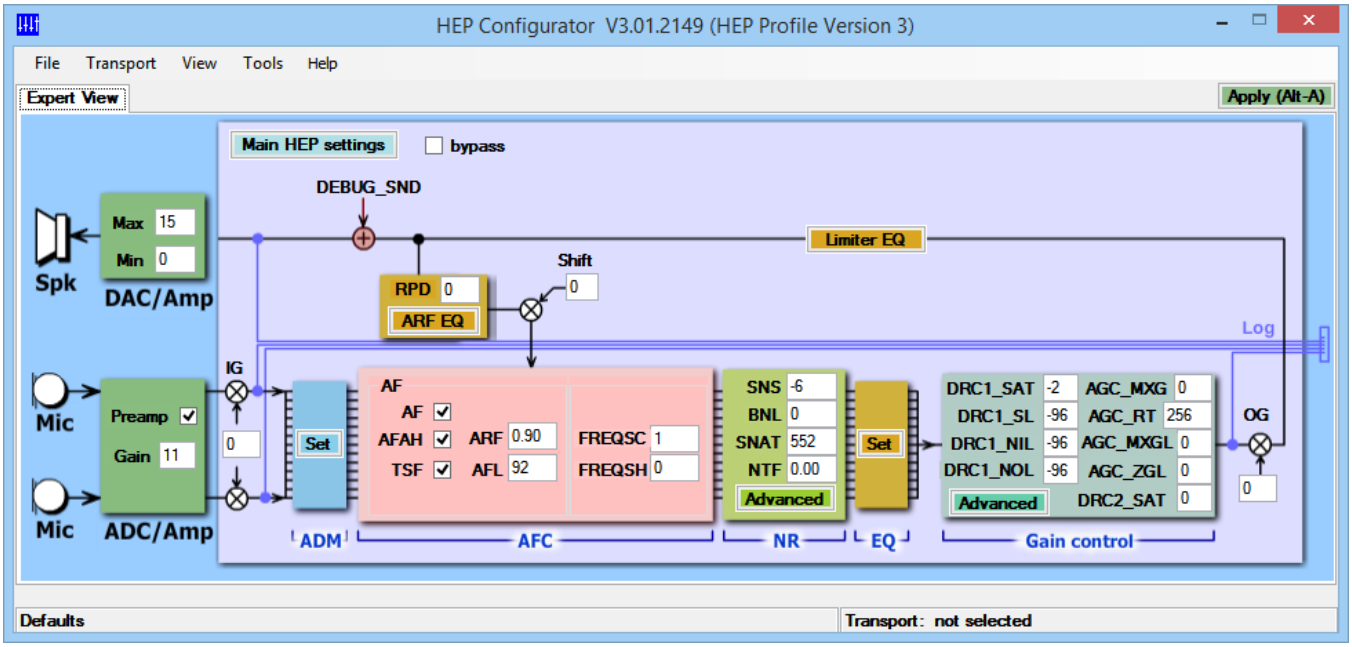

Figure 2. HEP Configurator screen in “Expert View”

The utility converts specified physical values of the parameters into their fixed-point representation form and stores them inside CSR 867x flash memory PSKEY (directly in the device in “on-line” mode) or in a form of .HEP disk files (“HEP acoustic profiles”).

User can read information on every HEP parameter presented in HEP Configurator window by clicking in the parameter edit box with right mouse button.

With the help of this configuration utility, system integrator can change HEP parameters “on-the-fly” i.e. during an active call. The new parameters stored in the PSKEYs will take effect almost instantly (in about 5 seconds or less after applying the parameters) thereby significantly simplifying the tuning procedure.

The following parameters are controlled by the utility:

- HEP DSP parameters
- HEP internal input and output digital gains
- bypass switch allowing to bypass any signal processing on appropriate channels
- Special HEP debugging modes such as “beeps” (mode forcing HEP to produce periodical beeps via RX indicating that HEP is up and running), “sweep-tone” generation, etc.
- HEP equalization (EQ) coefficients
- Loudspeaker minimal and maximal digital (DAC) gain
- Microphone boost switch (+20 dB boost) and ADC gain
- HEP license data

After pressing the “Apply” button, the parameters are stored in PSKEY which number is taken from the “Main HEP params PSKEY” field. The parameters are applied “on-the-fly” with no need to restart or reconnect the Bluetooth device. A short beep produced by the Bluetooth device indicates that the parameters where applied.

8. Digital signals logging to/from Kalimba DSP

Alango has developed a unique software plus hardware feature allowing monitoring and recording live HEP input and output signals directly from the CSR 8670 Kalimba DSP. The hardware interface utilizes just one user defined PIO line to transfer the audio input and output streams to PC for further analysis (Figure 3).

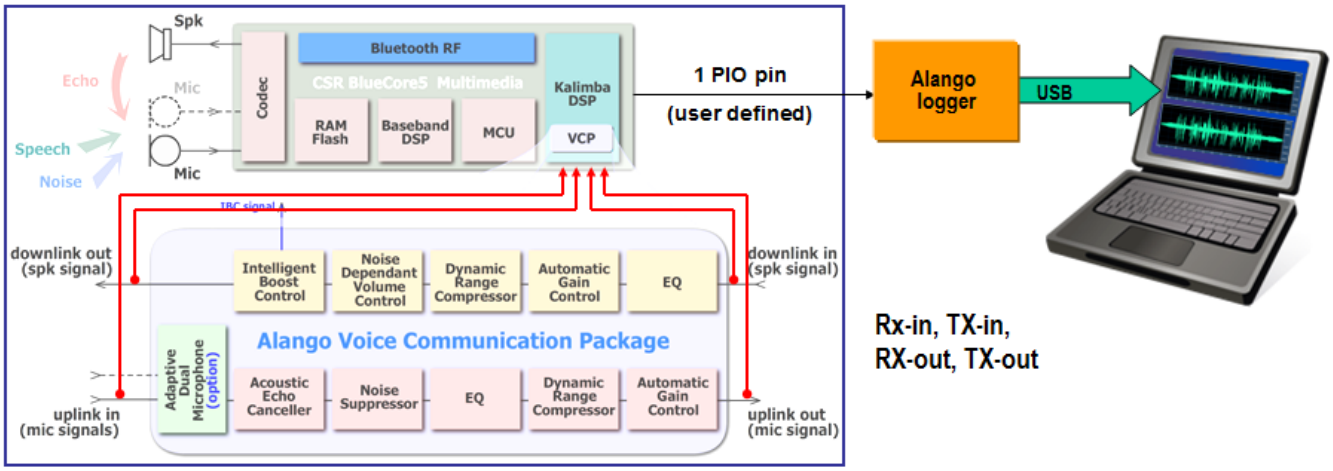

Figure 3. Digital signals logging (capturing) from BC05MM / 8670

CSR 867x PIO line number used for sending digital signals is a parameter of HEP library (HEP.KAP) and is controllable using HEP Configurator. The PIO signal is connected to Alango Logger Module (ALM), which represents standard FTDI high-speed USB-UART mini-module FT2232H with simple connections on it, as shown in a figure below.

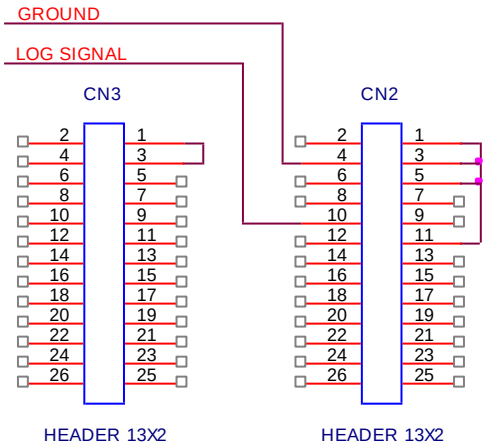

Figure 4. Alango Logger based on FTDI FT2232H mini-module

ALM receives digital audio signals via only one signal wire, so that only two wires connect between the voice communication device and the Logger: Log signal (BC05MM / 8670 PIO) and Ground.

ALM connects to PC running Alango Logger software utility using standard USB cable.

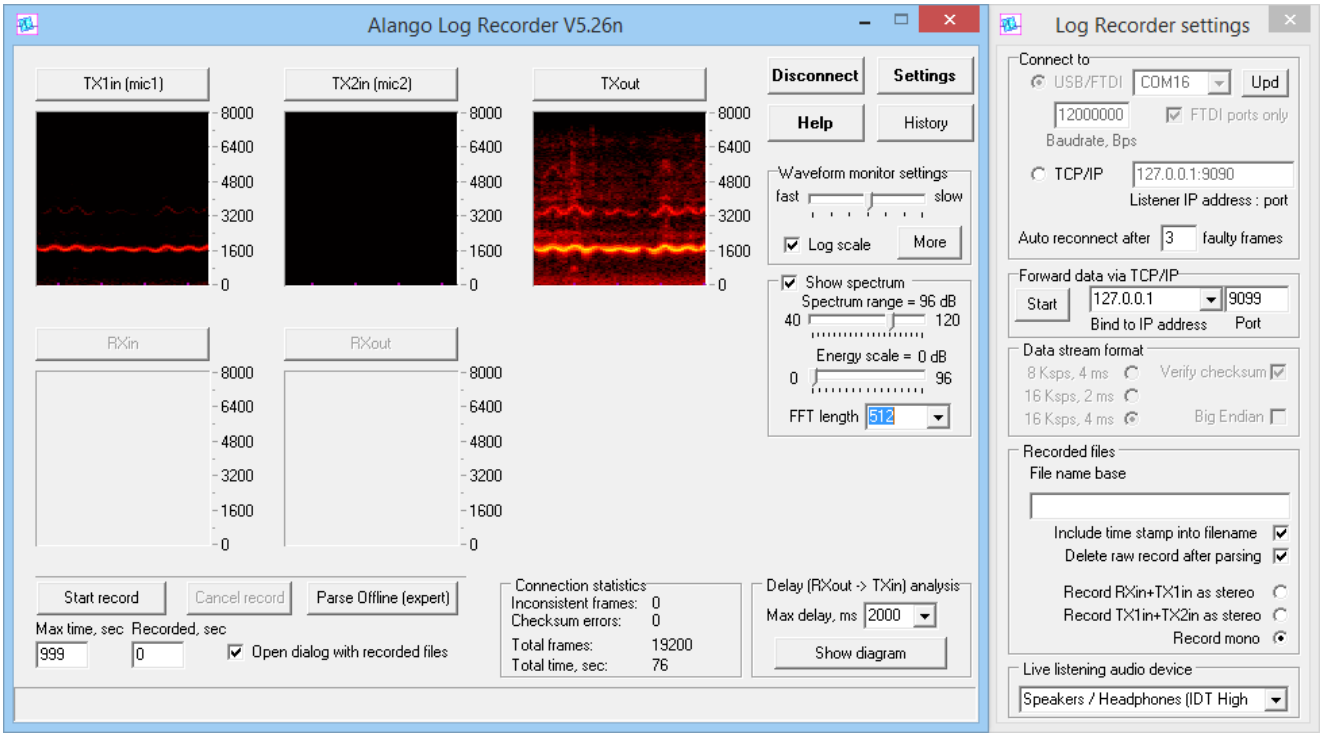

Figure 5. Alango Logger software utility in action

For proper operation, ALM requires a special driver to be installed in the system. The drivers are available for free at FTDI web-site (http://www.ftdichip.com/Drivers/D2XX.htm).

One simple change must be introduced inside VM application (in CVC_plugin) to enable the logging feature. It is necessary to set PIO direction and delegate the control by calling the following code that should be added **before** KalimbaLoad( ) function:

```
#include <pio.h>
.
.
.
#define LOG_PIO     <PIN_NUMBER_USED_FOR_TRANSMITTION>
#define LOG_MASK    (    (1 << LOG_PIO) & 0xFFFF    )
.
.
.
PioSetDir(LOG_MASK, LOG_MASK);
PioSetKalimba(LOG_MASK, LOG_MASK);
KalimbaLoad()
.
```

.

Note:
1. in SDK 2010 and further, use PioSetDir32() instead of PioSetDir(), and PioSetKalimbaControl32() instead of PioSetKalimba()
2. For CSR 8670 using ADK 1.1 only PioSetKalimbaControl32() must be called

9. Contact information

Alango Technologies Ltd.,

2 Etgar St., POB 62,
Tirat Carmel 39100, Israel

Tel: +972-4-8580743
Fax: +972-4-8580621

Web site: http://www.alango.com
E-mail: tech-il@alango.com