AIC3254 Acoustic Echo Cancellation (AEC)

Audio Converters

ABSTRACT

This application note describes the implementation of an effective, low cost Acoustic Echo Canceller (AEC) on the Texas Instruments AIC3254. This solution is developed as a simple to use set of drag and drop components in the PurePath Studio Portable Audio development environment. A set of control properties are described that allow simple customization of the AEC implementation making it easily adaptable to a variety of communication environments. It is designed to operate at a sample rate of 8 kHz and can incorporate either a 12ms or a 32ms adaptive NLMS filter. This application note also provides an overview of PurePath Studio Portable Audio's features and capabilities applicable to the customization of this AEC solution as well as instructions for tuning the AEC parameters for hand set and hands free applications.

Contents

Figures

Tables

1 Introduction

This application note describes the implementation of an effective Acoustic Echo Canceller (AEC) on the Texas Instruments AIC3254. This solution was developed as a set of drag and drop components in the PurePath Studio Portable Audio development environment. It is designed to operate at a sample rate of 8 kHz and can incorporate either a 12ms or a 32ms adaptive NLMS filter for echo cancellation and also provides a variety of control properties to customize the implementation for various communication environments.

This application note also provides an overview of PurePath Studio Portable Audio's features and capabilities applicable to the customization of this AEC solution as well as instructions for tuning the AEC parameters for hand set and hands free applications.

Echo cancellation is a requirement for many communication applications. Echo can be a very annoying problem to users of communication systems including conventional telephones, cell phones, speakerphones or mobile (automotive) phone products. Echo occurs when a signal is input at one end of a communication system, is output via a speaker at the other end and then returns as an echo via the near end microphone. See Figure 1 below. The echo is caused by coupling of the near end speaker and microphone. This can occur due to a variety of reasons such as poor physical layout of the speaker and microphone components, user positioning of a cell phone too far away from the head or excessive amplification of the received signal. Echo can be annoying to all participants impeding the intelligibility of a person speaking or by creating confusion in regards to who is talking at a given moment in time. There have, however, been effective acoustic echo cancellation (AEC) techniques developed over the years that incorporate adaptive filtering along with residual signal removal methods.

2 AEC Overview

AEC technology minimizes the effects of echo by using cancellation, suppression and masking techniques. There are three main technologies applied to perform these tasks as shown in Figure 1. The Normalized Least Mean Squared (NLMS) algorithm is an adaptive filtering technique designed to perform echo cancellation. Non-Linear Processing (NLP) and is applied to provide residual echo suppression where NLMS is not sufficient alone. A Comfort Noise Generator (CNG) is implemented to mask the effects of NLP mode changes (described below) and to eliminate any "dead air" effect. This effect, if not addressed by the CNG, can confuse a listener into thinking the line has temporarily been dropped. The following describes each of these AEC modules in more detail:

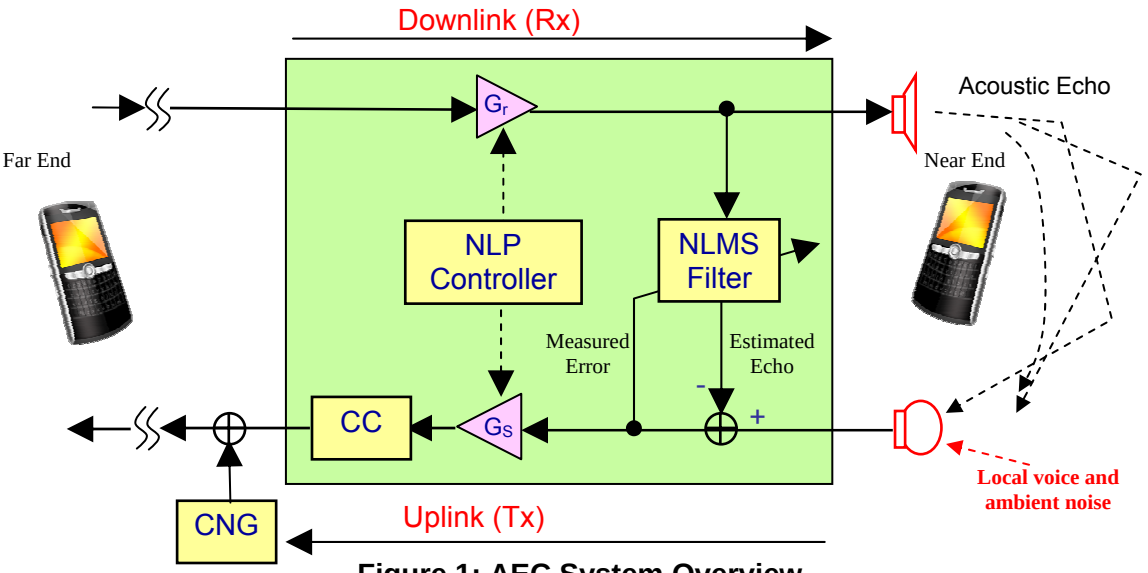

Figure 1: AEC System Overview

2.1 NLMS Adaptive filter

The NLMS Adaptive filter is based on the Normalized Least Mean Square (NLMS) algorithm. The NLMS algorithm incorporates an iterative procedure designed to make successive corrections to its weighted filter coefficients based on the measured error. This procedure eventually leads to the minimization of the mean squared error. The error in this application is the difference in the echo at the uplink (Tx) input and the estimated echo calculated by the NLMS algorithm.

As shown in Figure 1, the input signal from the far-end is directed to the NLMS module via the downlink (Rx) where the echo is estimated and passed to the uplink. The estimated echo signal is then subtracted from the uplink signal. The uplink signal consists of both the near-end transmit signal as well as the downlink echo. Ideally, the echo is completely removed and only the near-end transmit signal is passed to the far-end. However, due to dynamic changes in the uplink signal, nonlinear distortion caused by the speaker amplifier and loud speaker as well as changes in the acoustic environment (moving people, shades, chairs or doors as well as the microphone moving within the room, etc…) at the near-end, the NLMS estimate may not exactly match the

echo. The difference signal however is used to adapt the NLMS filter coefficients to converge the estimated echo to better match the real echo.

The rate at which the filter convergence occurs is controlled by two configurable parameters. There is a “faster step size” parameter that is automatically applied if the near-end microphone signal has only acoustic echo, and a “slower step size” parameter that is automatically applied if the near-end microphone signal has both acoustic echo and near-end speech (double-talk). If acoustic echo is not detected in the microphone input at all then no filter adaptation will be made.

2.2 Nonlinear Processor (NLP)

When the NLMS algorithm cannot completely remove echo, the NLP controller will suppress any remaining residual echo. The NLP controller initially classifies the communication scenario into one of the following four modes, and then applies an appropriate amount of attenuation to the Uplink (Tx) and Downlink (Rx) signals to make the residual echo less audible.

Idle	– Neither far-end or near-end is talking.
Double Talk	– Far-end and near-end are talking at the same time.
Far-end	– Only far-end is talking.
Near-end	– Only near-end is talking.

The aggressiveness of the NLP algorithm may be controlled by several build time properties. If the NLP algorithm is too aggressive, the terminal will provide half-duplex communication and will not allow a near-end talker to cut in during the conversation while a far-end user is talking. Under aggressiveness may not provide enough suppression. The Rx/Tx attenuation levels for each mode are therefore configurable.

2.3 Center Clipper (CC)

A Center Clipper (CC) module is applied to replace small-amplitude audio samples with zeros to make low-level residual echo completely inaudible. The CC threshold is configurable.

2.4 Comfort Noise Generator (CNG)

The CNG generates pink noise to make background noise sound smoother in the uplink (Tx). The CNG automatically estimates the near-end background noise level and generates pink noise that matches the same level as the estimated background noise. The comfort noise generator may mask low level transients generated by the NLP module when changing modes as well as avoid the effects of “dead air” caused by the elimination of all residual noise on the line. The Pink noise filter is a time-invariant 2^{nd} order all-pole filter and its filter coefficients are configurable.

3 AIC3254 AEC Implementation

The realization of the AEC algorithm on the AIC3254 is shown in Figure 2 as implemented in the PurePath Studio Portable Audio environment. See section 4 for additional information on the AIC3254 and PurePath Studio. This figure shows the process flow for implementation of AEC in a typical cell phone application. The far end input (shown in blue) is applied via the I2S port represented by the MonoI2S_AEC component and transferred from the miniDSP_D processor to the miniDSP_A processor. This is followed by a transferal back to the miniDSP_D, where the signal is processed by the AEC_32 component. The far end signal is then output via the Interpolator (AEC_Interp) and associated DAC. The near end input (shown in red) is applied via the ADC and decimator as represented by the AEC_Decim component. The signal is input to the miniDSP_A processor and passed to the miniDSP_D processor where AEC is applied by the AEC_32 component. The signal is then sent back to the miniDSP_A processor through the feedback component where the signal is then output via the I2S peripheral. The feedback component is necessary to allow a loopback cycle from the output of the AEC_32 component to the AEC_CNG component.

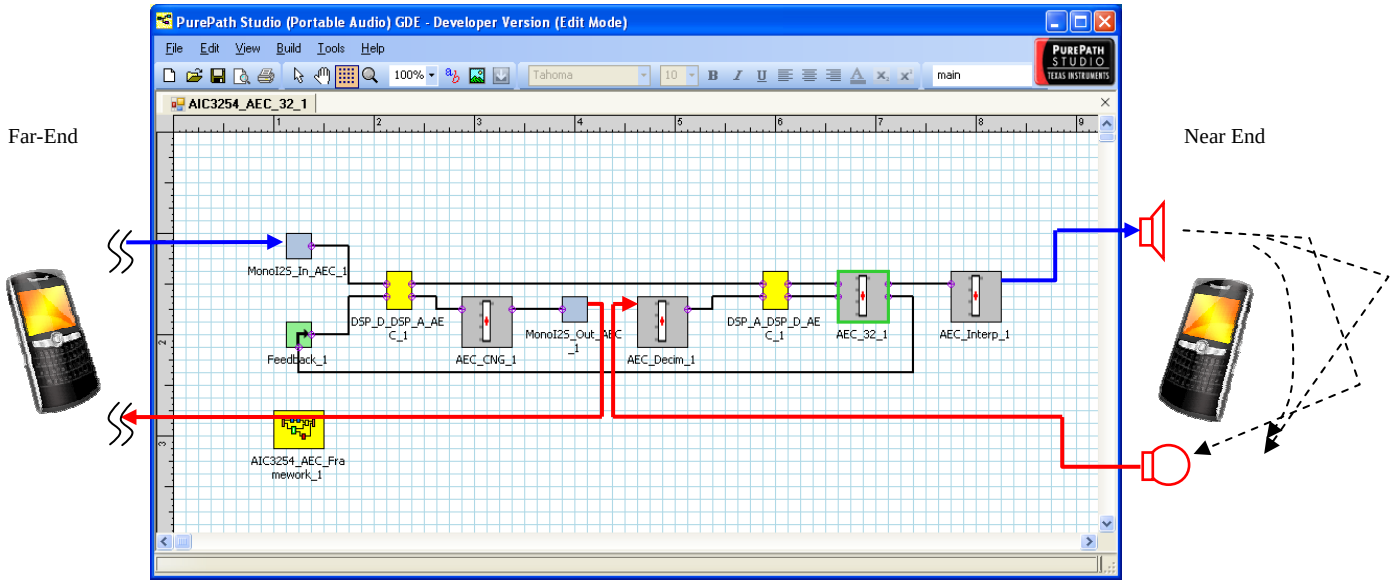

Figure 2: PurePath Studio AEC implementation

The following sections describe the AEC implementation in more detail.

3.1 AIC3254 Overview

The AEC algorithm is implemented on the TLV320AIC3254 dual core processor. The TLV320AIC3254 (sometimes referred to as the AIC3254) is a flexible, low-power, low-voltage stereo audio codec with programmable inputs and outputs, fixed/predefined and parameterizable signal processing blocks, integrated PLL, integrated LDOs and flexible digital interfaces. The AIC3254 incorporates dual miniDSPs (miniDSP_A and miniDSP_D) capable of meeting the needs of a variety of audio and communication applications. The miniDSP cores are fully software programmable and may be

controlled via I2C or SPI interfaces. See the SLAS549A Technical Data Sheet for more details on this processor (2).

The AIC3254 development environment is based on the PurePath Studio (PPS) tool set which incorporates a drag and drop style Graphical Development Environment (GDE) for simple development of process flows (i.e. systems) based on standard and custom components as described in the following sections.

3.2 PPS Environment

PurePath Studio incorporates a Graphical Development Environment (GDE) as shown in the figure below. This environment is used to create systems by dragging components from the palette onto the drawing area (environment) and connecting with lines to create system solutions.

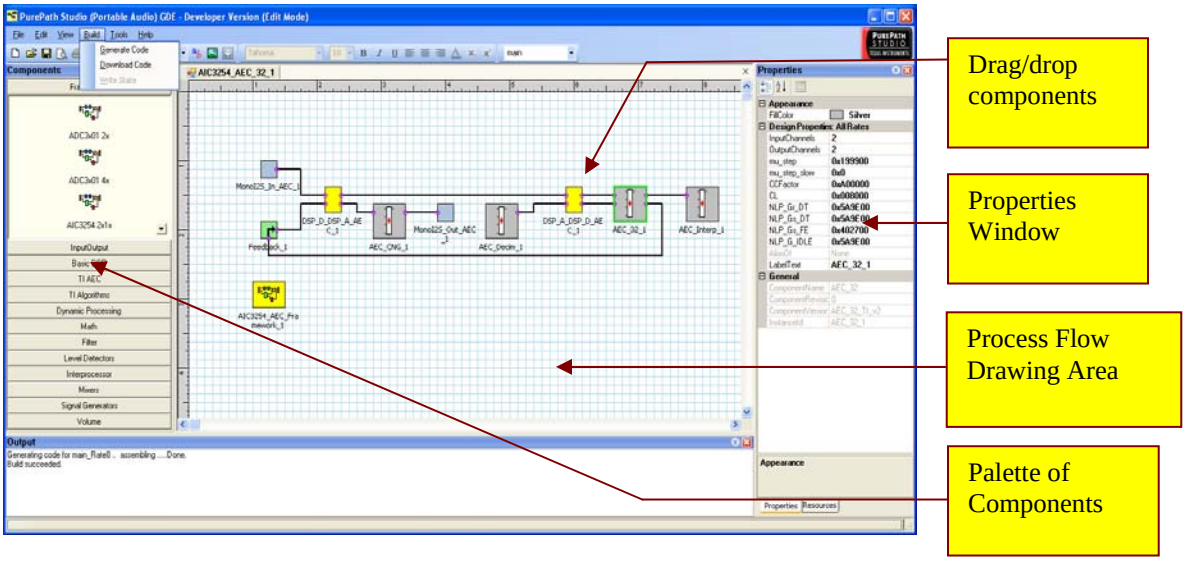

Figure 3: PurePath Studio AEC features

The environment defines various system parameters such as the sample rate and target type (i.e. AIC3254). These can be viewed in the configuration window by clicking on the drawing area.

3.2.1 PPS Components

By right clicking on a component the user can access either a help file describing the operation of the component or view available run time properties and/or design properties which are displayed on the right side of the drawing area. The AEC component does not have any run time properties, although it does have multiple design properties. Run time properties can be changed while the processor is running, but design properties can only be changed before building the process flow. To change properties, right click on the component and choose Properties option. The component properties are displayed on the right of the drawing area, and those properties that can be changed are displayed in bold type. Click on the property to be changed and type in a

desired value. Components are linked together to build systems such as an AEC-enabled device.

3.2.2 Build and Download Process Flow

A build is performed when the process flow graph has been completed by pulling down the “Build” menu as shown in the figure above. Choose “Generate Code” to build the system and then follow up with ‘Download code’ to load the software to the target board via the USB connection. The user can also choose ”download code” initially, in which case both build and download will be performed. The code is loaded and automatically starts running. The user is requested to save the process flow. This creates a process flow file with the extension *.pfw. Process flow files may also be loaded from the “File” menu using the “Open” option. See *Section 4.1 Hand Set (HS) Operation* for an example.

3.2.3 Loading Scripts

Once the code has been loaded and is running, changes to the code or registers may be made dynamically using scripts. Scripts may be loaded by pulling down the “Tools” menu and choosing the “I2C Commands” File option (as shown in the following figure). Scripts are text files formatted to be read by the AIC3254. Script files may be edited using any text editor.

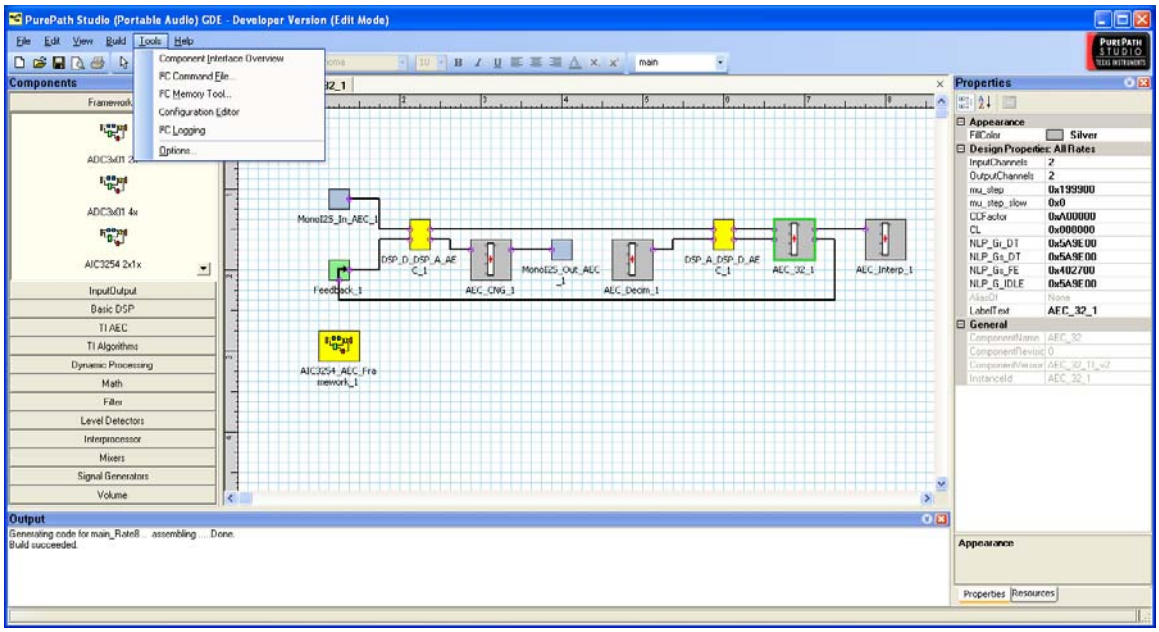

Figure 4: PurePath Studio Tools menu

The following window opens and the user can browse to locate the desired script file and click on “Execute now”. This will load the script to the AIC3254.

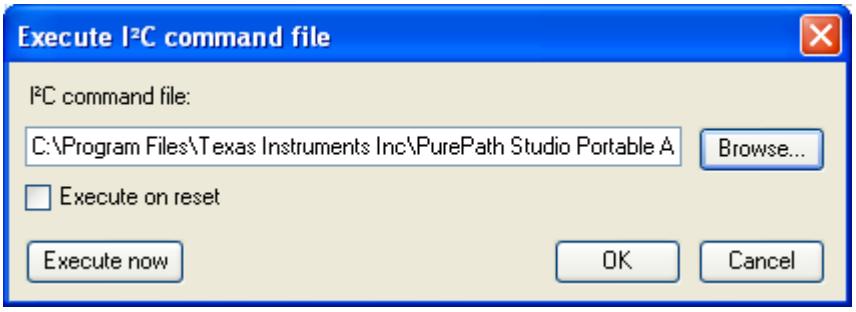

Figure 5: I2C Command Window

See ***Section 4.2 Hands Free (HF) Operation*** for an example of using scripts.

4 AEC Process flow

The AEC process flow consists of a few components that shall be described below. Processing for the AEC algorithm is performed across both DSPs of the AIC3254. The majority of the implementation is applied in miniDSP_D while much of miniDSP_A is left available for additional processing. See Section 5 for resource availability.

4.1 AEC Framework and AEC Components

4.1.1 AEC_Framework

The AEC_Framework component is required. This component determines register settings for the AIC3254 necessary for proper AEC operation. This includes registers controlling clocking, I/O, and power supply operation.

4.1.2 AEC_32 or AEC_12

Only one of these components is required. The AEC_32 component implements AEC using a 32ms, 256 tap FIR adaptive NLMS filter. The AEC_12 component implements AEC using a 12ms, 96 tap FIR adaptive NLMS filter. The AEC_32 component utilizes more resources (see Section 5 for details) but provides improved echo cancellation. Each of these components implements the NLMS, NLP and CC functions of the AEC algorithm. One of these components must be incorporated in the miniDSP_D processor and connected to the rest of the AEC system as shown in Figure 4.

4.1.3 AEC_CNG

The AEC_CNG component is required. This component implements the Comfort Noise Generation function of the AEC algorithm. This component must be incorporated in the miniDSP_A processor and connected to the rest of the AEC system as shown in Figure 4.

4.2 Inputs/Outputs Components

4.2.1 AEC_Decim

The AEC_Decimation component is required. This component implements the ADC decimation (4x) function required for the uplink input (This is also referred to as the send input (Sin)). This component must be incorporated in the miniDSP_A processor and connected to the rest of the AEC system as shown in Figure 4.

4.2.2 AEC_Interp

The AEC_Interpolation component is required. This component implements the DAC interpolation (4x) function required for the downlink output (this is also referred to as the receive output (Rout)). This component must be incorporated in the miniDSP_D processor and connected to the rest of the AEC system as shown in Figure 4.

4.2.3 MonoI2S_In_AEC

The MonoI2S_In_AEC component is required. This component receives downlink data (this is also referred to as the received input (Rin)) from the I2S port and inputs to the AEC system. This component must be incorporated in the miniDSP_D processor and connected to the rest of the AEC system as shown in Figure 4.

4.2.4 MonoI2S_Out_AEC

The MonoI2S_Out component is required. This component transmits uplink data (this is also referred to as the send output ((Sout)) to the I2S port. This component must be incorporated in the miniDSP_A processor and connected to the rest of the AEC system as shown in Figure 4.

4.3 Inter-processor Components

4.3.1 DSP_A_DSP_D_AEC

The DSP_A_DSP_D_AEC component is required. This component transfers data from miniDSP_A to miniDSP_D and must be connected in the AEC process flow as shown in Figure 4.

4.3.2 DSP_D_DSP_A_AEC

The DSP_D_DSP_A_AEC component is required. This component transfers data from miniDSP_D to miniDSP_A and must be connected in the AEC process flow as shown in Figure 4.

4.4 Integrating Custom Components

The AEC implementation on the AIC3254 supports system customization by dragging and dropping components onto the drawing area from the palette and linking into the AEC process flow. There are two areas where custom processing may be incorporated into the process flow.

1. Area 1: The received input (Rin). Following the MonoI2S_In_AEC component and before the DSP_D_DSP_A_AEC component.

2. Area 2: The send output (Sout). Following the AEC_CNG component and before MonoI2S_Out_AEC component.

See Section 5 regarding available resources. After linking in new components the available resources can be evaluated after a build. The resource window can be viewed via the View menu as shown in Figure 6 below. See Resources menu in the bottom of the right window as shown in Figure 6.

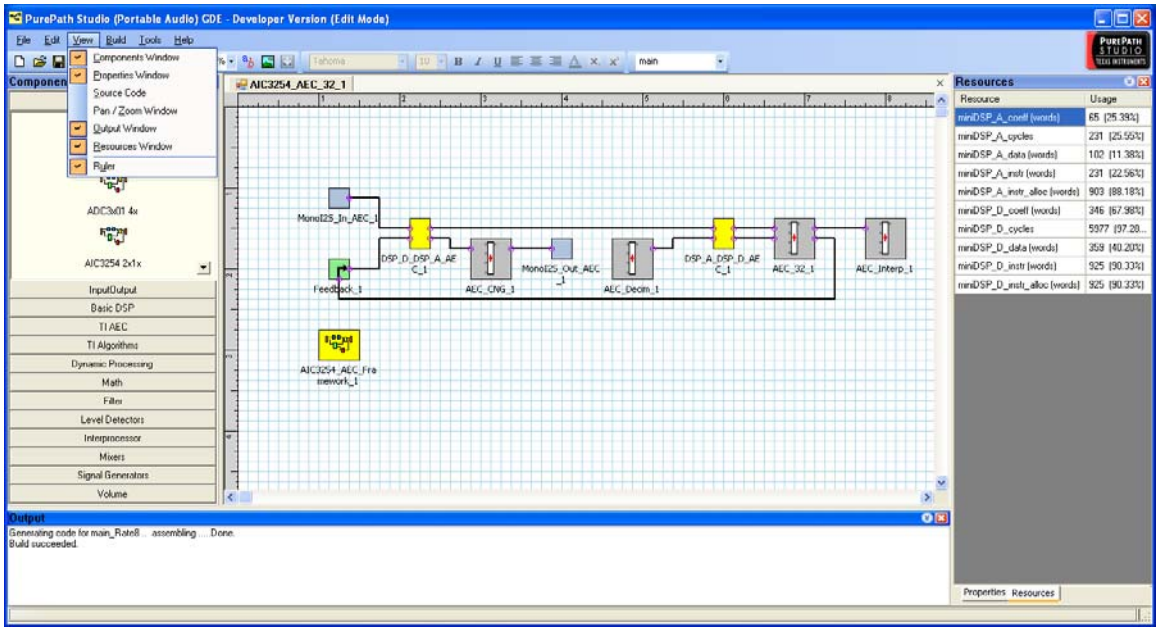

Figure 6: PurePath Studio View menu

4.5 AEC Control Properties

Each of the AEC processing components (AEC_12, AEC_32) provides a common set of programmable properties. The AEC component does not require run time properties; however, there are multiple design properties that can be customized prior to building and loading the AEC component in a PurePath Studio process flow. These design properties are located in the right window when highlighting either the AEC_32 or AEC_12 component. These properties are described below. Examples are also provided to assist in formatting the desired values in the appropriate fixed point format.

4.6 NLMS Properties

As previously noted, the NLMS algorithm has two controllable properties: *faster step size* and *slower step size*. These can be viewed and edited by clicking on the AEC_12 or AEC_32 component. These parameters are displayed in the design time properties window on the right of the PPS environment window.

4.6.1 Faster step size (mu_step)

The *faster step size* property determines how quickly the NLMS algorithm will converge and cancel the echo. This parameter is used during the far-end (FE) mode of operation when only a far-end signal is detected. A larger value makes the NLMS convergence

speed faster, but degrades the stationary convergence performance. The following shows how to convert the desired value (μ) to the required fixed-point format:

mu_step = round {μ * 2^{25}}
e.g. if μ is 0.1, tuning parameter mu_step = 0x333333

4.6.2 Slower step size (mu_step_slow)

When the NLMS algorithm detects double-talk (i.e. echo and near-end speech), the *slower step size* is used. Although a smaller value degrades convergence performance during double-talk, it also protects the adaptive filter from divergence better. The slower step size (i.e. *mu_step_slow*) must be smaller than the faster step size (i.e. *mu_step*). The following shows how to convert the desired value (μ) to the required fixed-point format:

mu_step_slow = round {μ * 2^{25}}
e.g. if μ is 0.025, tuning parameter mu_step_slow = 0x0CCCCD

4.7 NLP Properties

The NLP module also has six (6) control properties as described below: These can be viewed and edited by clicking on the AEC_12 or AEC_32 component. These parameters are displayed in the design time parameter window on the right of the PPS window.

4.7.1 Center Clipper (CCFactor) threshold

A Center Clipper operates only at very low signal levels to remove residual echo and artifacts by zeroing signals below a defined threshold. The Center Clipper threshold is bound between 0.0 and 1.0. 1.0 provides the strongest amount of center clipping, and 0.0 disables center clipping. The following shows how to convert the desired value (CC) to the required fixed-point format:

$$CCfactor = round \{CC * (-2^{23})\}$$

e.g. if CC = 0.75, CCfactor = 0xA00000

4.7.2 Coupling Loss (CL)

To make the NLP controller work properly, Coupling Loss has to be configured for the target platform terminal. CL is a ratio of the Rx digital output level to the corresponding acoustic echo level at the Tx digital input. If the speaker and microphone are acoustically de-coupled well, CL will be a larger number because the echo level will be lower with a given Rx signal. In a typical mobile phone form factor, CL = ~ 10 to 20 dB for handheld mode, and ~ -15 to 0 dB for speakerphone mode.

The following shows how to convert the desired value to the required fixed-point format:

$$CL = round \{10^{(-0.1*CL\_dB)} * 2^{17}\}$$

e.g. if CL_dB = 0 dB, Note: CL_dB is in the log domain (dB)
CL = 0x020000

4.7.3 NLP Rx Attenuation in DT (NLP_Gr_DT)

The NLP_Gr_DT property controls the downlink (Rx) attenuation level during double-talk mode. This value should be set according to the requirements outlined in Table 1. The following shows how to convert the desired value (Grxdt_dB) to the required fixed-point format:

$$NLP\_Gr\_DT = round \{10^{(0.05*Grxdt\_dB)} * 2^{23}\}$$

e.g. if Grxdt_dB = -3 dB, Note: Grxdt_dB is in the log domain (dB)
NLP_Gr_DT = 0x5A9DF8

4.7.4 NLP Tx Attenuation in DT (NLP_Gs_DT)

The NLP_Gs_DT property controls the uplink (Tx) attenuation level during double-talk mode. This value should be set according to the requirements outlined in Table 1. The following shows how to convert the desired value (Gtxdt_dB) to the required fixed-point format:

$$NLP\_Gs\_DT = round \{10^{(0.05*Gtxdt\_dB)} * 2^{23}\}$$

e.g. if Gtxdt_dB = -3 dB, Note: Gtxdt_dB is in the log domain (dB)
NLP_Gs_DT = 0x5A9DF8

4.7.5 NLP Tx Attenuation in FE (NLP_Gs_FE)

The NLP_Gs_FE property controls the uplink (Tx) attenuation level during far-end only mode. This value should be set according to the requirements outlined in Table 1. The following shows how to convert the desired value (Gtxfe_dB) to the required fixed-point format:

$$NLP\_Gs\_FE = round\{10^{(0.05*Gtxfe\_dB)} * 2^{23}\}$$

e.g. if Gtxfe_dB = -15 dB, Note: Gtxfe_dB is in log domain (dB)

NLP_Gs_FE = 0x4026E7

Table 1. Attenuation level tuning guideline

Terminal Type*	Attenuation Level in double-talk*		Attenuation Level in far-end only**
	Rx	Tx	Tx
1 (full-duplex)	≤ 3 dB	≤ 3 dB	6 ~ 15 dB
2a (partial-duplex)	≤ 5 dB	≤ 6 dB	~ 24 dB
2b (partial-duplex)	≤ 8 dB	≤ 9 dB	~ 36 dB
2c (partial-duplex)	≤ 10 dB	≤ 12 dB	~ 48 dB
3 (half-duplex)	> 10 dB	> 12 dB	~ 60 dB

* Terminal type and attenuation level in double-talk are defined in ITU-T P.340

** There is no standard guideline for attenuation in far-end only mode. Values in the table are informational only.

4.7.6 NLP Rx/Tx Attenuation in IDLE (NLP_G_IDLE)

The NLP_G_IDLE property controls both the downlink (Rx) and uplink (Tx) attenuation level during idle mode. The following shows how to convert the desired value (Gidle_dB) to the required fixed-point format:

$$NLP\_G\_IDLE = round\{10^{(0.05*Gidle\_dB)} * 2^{23}\}$$

e.g. if Gidle_dB = -3 dB, Note: Gidle_dB is in log domain (dB)

NLP_G_IDLE = 0x5A9DF8

4.8 Comfort Noise Properties

The Comfort Noise generator filter can be controlled by the following control properties (*CNG_b0, CNG_a1*, and *CNG_a2*). These are coefficients that define the shape of the noise generated to mask NLP transients and eliminate "dead air" effect. They are controlled by the run time properties associated with the AEC_CNG component. Highlight the AEC_CNG component and see the design time properties in the properties window on the right. This filter is a 2^{nd} order all-pass filter described by the following equation:

$$y(n) = B0*x(n) + A1 * y(n-1) + A2 * y(n-2)$$

4.8.1 CNG filter coefficients (CNG_b0, CNG_a1, CNG_a2)

The following is an example comfort noise filter. This example implements coefficients that generate pink noise with a peak frequency of 500 Hz and pole radius of 0.9 at a sampling rate of 8 kHz:

A1 = 2*0.9*cos(2*pi*500/8000)
A2 = -0.9^2
B0 =0.2315 to adjust filter gain to 0 dB.

The following shows how to convert the desired coefficient value to the required fixed-point format:

B0_fix = round {B0 * 2^{23}} = 0x1DA1CA
A1_fix = round {A1 * 2^{22}} = 0x6a6e51
A2_fix = round {A2 * 2^{23}} = 0x9851EC

5 AEC Application Examples

The AEC automatically runs on the AIC3254 EVM using the provided components, process flows and scripts. The EVM must initially be set up as described in Section 3 of the EVM User's Guide, TLV320AIC3254EVM-K, SLAU264-October 2008. Connect the USB cable from the PC to the EVM. Open PurePath Studio to continue running the example.

5.1 Hand Set (HS) Operation

The AEC implementation on the AIC3254 defaults to support hand set operation. Hand Set operation can be implemented using the following procedure:

1. From PPS, Load the process flow "ExampleProcessFlow/AEC /AIC3254_AEC_32_1.pfw". See ***Section 3.1.3 Build and Download Process flow***, for details on loading a process flow.

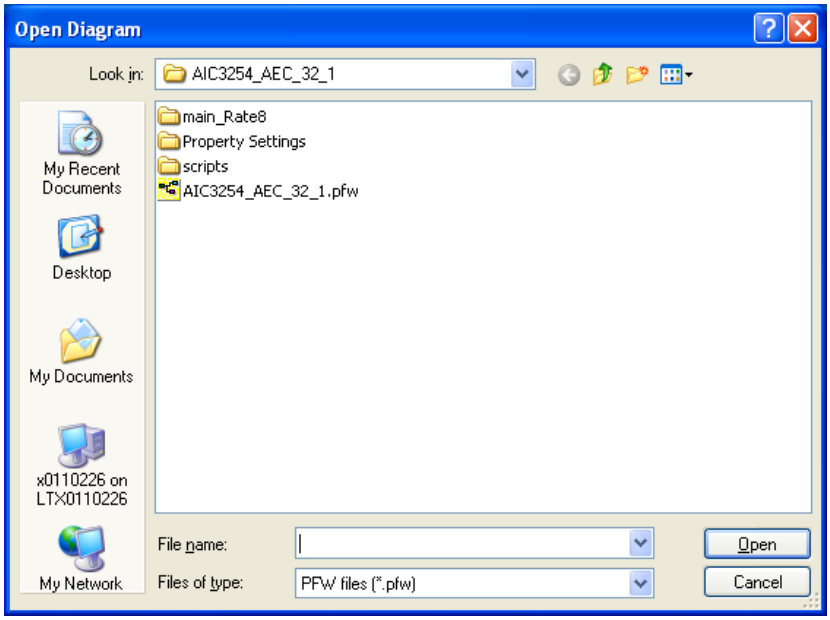

Figure 7: Download Window

2. All AEC properties must be modified prior to build if a desired change is to be made to the AEC properties. Set up the design properties to the desired values. See ***Section 3.1.2, PPS Components*** for details on modifying design properties.

3. From the Build menu, choose the "Download Code: option. See ***Section 3.1.3 Build and Download Process flow.***

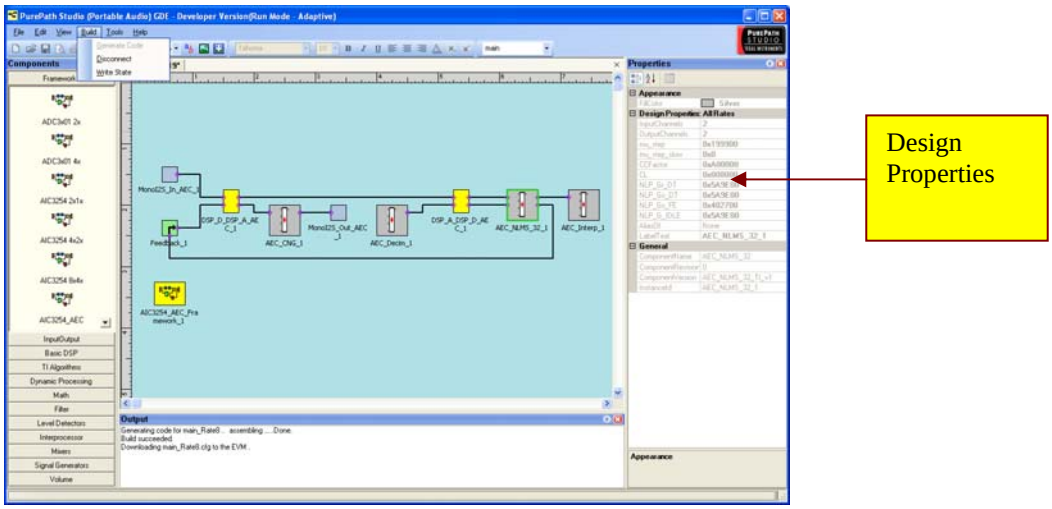

Figure 8: Download Window

This will build and load the process flow. The AEC should automatically be running on the target processor and initially configured for use in a hand set. The uplink output is configured for a headphone peripheral and the microphone level is set lower than the hands free operation.

To summarize the I/O setup, the AEC component has four I/O and is supported on the AIC3254EVM as shown in the following table.

Table 2. Default AEC I/O – Hand Set operation

AEC I/O	AIC3254 Peripheral	AIC3254EVM connector
Uplink Input (Sin)	IN3L - ADC	Mic Input (J5/J4 connector)
Uplink Output (Sout)	I2S Output	USB
Downlink Input (Rin)	I2S Input	USB
Downlink Output (Rout)	HPL - DAC	Headphone output (J9 connector)

A microphone and speaker can be connected to the respective connectors while the USB port may be used to stream and record files.

The following describes how to set up the AEC in PPS for hand set operation.

4. The AEC parameter tuning process is described below.
   a) Disable all AEC parameters (see Table 3).
   b) Determine the DAC digital PGA gain so that it provides sufficient loudness at the speaker output. If a target device allows end-users to change the speaker volume, set the volume level to nominal and then determine the DAC PGA gain. It is suggested to start with a DAC PGA gain of about -6 dB.
   c) Determine the ADC analog PGA gain that provides sufficient signal level at the I2S output. It is suggested to start with an ADC PGA gain of about 30 dB.

d) Measure the I2S input and output signal levels. These will be used to set some tuning parameters later.

e) Enable the AEC by setting fast and slow step sizes to their default values in Table 3.

f) Apply signal to the I2S input and observe the echo signal at the I2S output in a quiet environment with no local voice. If the echo divergence is observed (echo level is increased instead of being suppressed), reduce the fast step size value. Repeat this process until the echo divergence is not observed at all. No changes should be made for the slow step size at the moment.

g) Determine a preliminary CL value as follows:

   CL (dB) = Pin (dB) + DAC PGA (dB) – Pout (dB)

   Pin (dB) and Pout (dB) are the I2S input and output levels respectively that were measured in (d). Note that the ADC PGA is not included in the CL calculation. Change the default CL value to the calculated preliminary value.

h) Enable the NLP Attenuation parameters by setting these four parameters to their default values:

   1) Rx and Tx attenuations in DT (NLP_Gr_DT/NLP_Gs_DT)
   2) Tx attenuation in FE (NLP_Gs_FE)
   3) Rx and Tx attenuation in IDLE (NLP_G_IDLE)

   Enable the NLP CC parameter by setting it to its default value.

i) Apply signal to the I2S input and observe the echo signal at the I2S output in a quiet environment with no local voice. If the residual echo level at the I2S output is still high, make the far end transmit attenuation (NLP_Gs_FE) stronger by changing it to a smaller value until the residual echo level at the I2S output is inaudible or at a desired level.

j) Apply signal to the I2S input and observe the echo signal at the I2S output with local voice (i.e. in double-talk mode).

   Here is how to fine tune the CL parameter. If the local voice is choppy during double-talk, decrease the CL value. If residual echo is too audible during double-talk, increase the CL value. Find the optimal trade-off for the CL value between local voice quality and residual echo level during double talk.

k) If the residual echo level is high during double-talk after the fine tuning of CL, make the Rx and Tx DT attenuations (NLP_Gr_DT/NLP_Gs_DT) stronger by changing them to smaller values. See the DT attenuation example in Table 4. Typically, for the hand set operation, either terminal type 1 or 2a example setting can be used. Changing the slow step size may or may not affect residual echo level during double talk. It is recommended to keep the slow step size at the default value (0x000000).

l) Enable the CNG by setting the CNG filter coefficients b0, a1 and a2. The CNG level can be controlled by increasing or decreasing the b0 value. See ***Section 3.3.3, Comfort Noise Properties*** for more detailed filter coefficients information.

Table 3. Default AEC parameters in hand set operation

	Parameter Name	Default	Disabled
1	Faster Step Size (mu_step)	0x199900	0x000000
2	Slower Step Size (mu_step_slow)	0x000000	0x000000
3	CC threshold (CC Threshold)	0xa00000	0x000000
4	Coupling Loss (CL)	0x008000	-
5	NLP Rx Attenuation in DT (NLP_Gr_DT)	0x5a9e00	0x7FFFFF
6	NLP Tx Attenuation in DT (NLP_Gs_DT)	0x5a9e00	0x7FFFFF
7	NLP Tx Attenuation in FE (NLP_Gs_FE)	0x402700	0x7FFFFF
8	NLP Rx/Tx Attenuation in IDLE (NLP_G_IDLE)	0x5a9e00	0x7FFFFF
9	CNG filter B0 (CNG_b0)	0x213f00	0x000000
10	CNG filter A1 (CNG_a1)	0x6a6e51	-
11	CNG filter A2 (CNG_a2)	0x9851ec	-

Table 4. NLP Attenuation parameter example

Terminal Type	Attenuation in double-talk		Attenuation in FE
	Rx	Tx	Tx
1 (full-duplex)	0x5a9df8 (-3dB)	0x5a9df8 (-3dB)	0x4026e7 (-6dB)
2a (partial-duplex)	0x47facd (-5 dB)	0x4026e7 (-6dB)	0x081385 (-2 dB)
2b (partial-duplex)	0x32f52d (-8dB)	0x2d6a86 (-9dB)	0x020756 (-36dB)
2c (partial-duplex)	0x287a27 (-10dB)	0x2026f3 (-12dB)	0x008274 (-48dB)
3 (half-duplex)	0x7fffff (0 dB)	0x000000 (-inf)	0x000000 (-inf)

5.2 Hands Free (HF) Operation

Hands free operation can be implemented using the following procedure:

1. Setting up hands free operation can be performed by following the first three steps in the hand set procedure in ***Section 4.1, Hand Set (HS) Operation*** above. The AIC3254 will initially be set up for hand set operation.

2. Configuring the AEC implementation for Hands Free operation requires loading a script. There is a script to do this in the “ExampleProcessFlow/AEC” directory called “AEC_HF.txt”. See ***Section 3.1.4 Loading Scripts*** for details on loading a script.

This script will redirect the DAC output from the headphone peripheral to the line out peripheral on the AIC3254 and adjust the microphone PGA level to compensate for the head-phone distance associated with using a hands free phone. The EVM connection is as follows for Hands Free Operation:

Table 5. AEC I/O – Hands Free operation

AEC I/O	AIC3254 Peripheral	AIC3254EVM connector
Uplink Input (Sin)	IN3L - ADC	Mic Input
Uplink Output (Sout)	I2S Output	USB
Downlink Input (Rin)	I2S Input	USB
Downlink Output (Rout)	LOL – DAC	Line output to TPA2010D1

Note that in order to adequately drive most hands free speakers, it is usually necessary to utilize an external audio amplifier (such as the TPA2010D1). Thus, to achieve the best performance possible, attach a TPA2010D1 EVM board to the Line Output of the AIC3254 EVM as shown below.

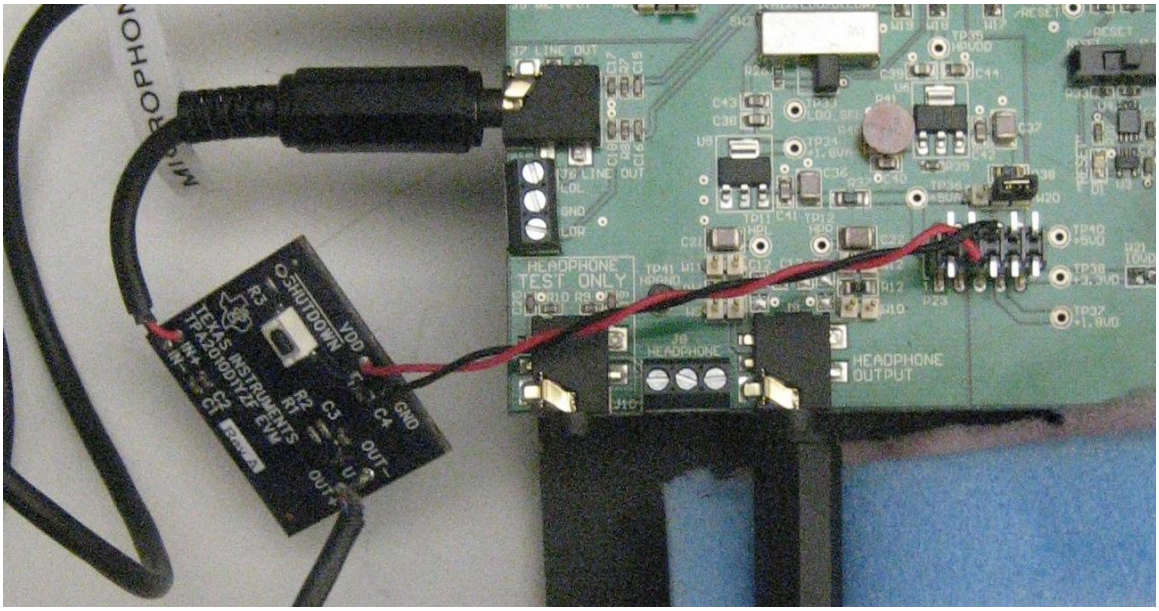

Figure 9: TPA2010D1 EVM board Line Out connection

3. The AEC tuning process for hands free operation is the same as for the hand set tuning process (see ***Section 4.1, Hand Set (HS) Operation***, step 4). See Table 6 for default Hands Free Operation settings. It is recommended to always disable the CC by setting the threshold to zero for hands free operation.

For hands free operation with good speaker-microphone separation, any of the terminal type 1, 2a or 2b settings in Table 4 can be used. For hands free operation with strong speaker-microphone coupling or severe nonlinear distortion due to small speaker or plastic vibration, any of the terminal type 2b, 2c or 3 settings in Table 4 should be used.

Table 6. Default AEC parameters in hands free operation

	Parameter Name	Default	Disabled
1	Faster Step Size (mu_step)	0x199900	0x000000
2	Slower Step Size (mu_step_slow)	0x000000	0x000000
3	CC threshold (CC Threshold)	0x000000	0x000000
4	Coupling Loss (CL)	0x008000	-
5	NLP Rx Attenuation in DT (NLP_Gr_DT)	0x5a9e00	0x7FFFFF
6	NLP Tx Attenuation in DT (NLP_Gs_DT)	0x5a9e00	0x7FFFFF
7	NLP Tx Attenuation in FE (NLP_Gs_FE)	0x402700	0x7FFFFF
8	NLP Rx/Tx Attenuation in IDLE (NLP_G_IDLE)	0x5a9e00	0x7FFFFF
9	CNG filter B0 (CNG_b0)	0x213f00	0x000000
10	CNG filter A1 (CNG_a1)	0x6a6e51	-
11	CNG filter A2 (CNG_a2)	0x9851ec	-

6 AEC System Requirements

The AEC_12 has the following resource requirements.

Table 7. AEC_12 Resource Requirements

AEC_12		Required	Available on AIC3254
miniDSP_A	Instruction Words	231	1024
	Data RAM	101	896
	Coefficient RAM	65	512
	Cycles/Frame	218	904
miniDSP_D	Instruction Words	696	1024
	Data RAM	239	896
	Coefficient RAM	182	512
	Cycles/Frame	2742	6144

The AEC_32 has the following resource requirements.

Table 8. AEC_32 Resource Requirements

AEC_32		Required	Available on AIC3254
miniDSP_A	Instruction Words	231	1024
	Data RAM	101	896
	Coefficient RAM	65	512
	Cycles/Frame	218	904
miniDSP_D	Instruction Words	925	1024
	Data RAM	358	896
	Coefficient RAM	345	512
	Cycles/Frame	5967	6144

7 Conclusion

An AEC solution has been developed for the AIC3254 DSP processor operating in the PurePath Studio environment. The implementation provides an extensive number of parameter controls to customize the AEC characteristics for a multitude of applications. Additionally, an extensive amount of additional resources are made available for inclusion of supplementary processing features.

8 References

1. PurePath Studio Integrated Development Environment (IDE), TLV320AIC3254 miniDSP Programmers Guide, V1.5 23 April 2009.
2. TLV320AIC3254 Ultra Low Power Stereo Audio Codec With Embedded miniDSP Technical Data Sheet, SLAS549A–SEPTEMBER 2008–REVISED OCTOBER 2008.
3. PurePath Studio for Portable Audio, Graphical Development Environment User's Quick Start Guide for the TLV320 AIC3254 EVM, January 6th, 2009.
4. EVM User's Guide, TLV320AIC3254EVM-K, SLAU264-October 2008