TI Designs

Speech Recognition Reference Design on the C5535 eZdsp™

TI Designs

This TI design highlights the voice recognition capabilities of the C5535 and C5545 devices using the TI embedded speech recognition (TIesr) library and instructs how to run a voice triggering example that prints a preprogrammed keyword on the C5535eZdsp OLED screen, based on a successful keyword capture. This design also describes the steps to customize the trigger phrase.

The speech recognition capability of the C5535 and C5545 devices showcases the performance and versatility of this low-power, low-cost DSP in embedded applications.

Design Resources

TIDEP0066	TI Design Files
TMDX5535eZdsp	Tool Folder
TMS320C5535	Product Folder
TMS320C5545	Product Folder
TLV320AIC3204	Product Folder

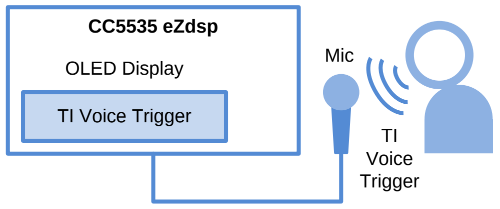
TI E2E™ Community — ASK Our E2E Experts

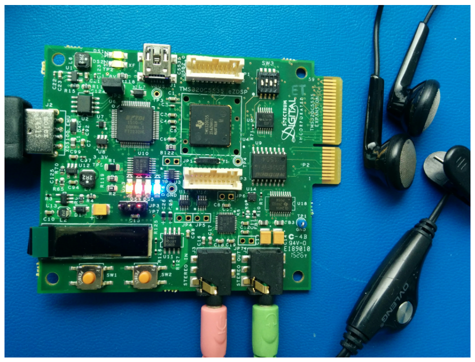
Design Features

- Includes Software Source, Schematics, Bill of Materials, and Design Files.
- Applicable to Any Application Requiring Voice Triggering.
- Offers Customizable Trigger Words.

Featured Applications

- Voice Triggering for Embedded Devices
- Acoustic Analysis
- Wireless Audio Devices (for example, Headsets Microphones, and Speaker phones)
- Industrial Controls





An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 System Description

The C5535/C5545 fixed-point DSP is based on the TMS320C55x DSP core. The C55x DSP architecture achieves high performance and low power through increased parallelism and focus on power savings. This low-cost DSP works well in embedded speech recognition applications and this reference design showcases this capability. The design contains reference C code and binaries to run a voice trigger demonstration that detects a keyword phrase spoken into the microphone of the C5535 eZdsp™ EVM.

1.1 C5535 and C5545 Fixed-Pint Digital Signal Processors

The C5535 and C5545 DSPs sample the incoming audio data and make decisions on whether the keyword phrase *TI voice trigger* has been recognized with the assistance of the TI embedded speech recognition (TIesr) software library. Section 3.2 describes the algorithm and the various threads the DSP handles. For more information, see Figure 1.

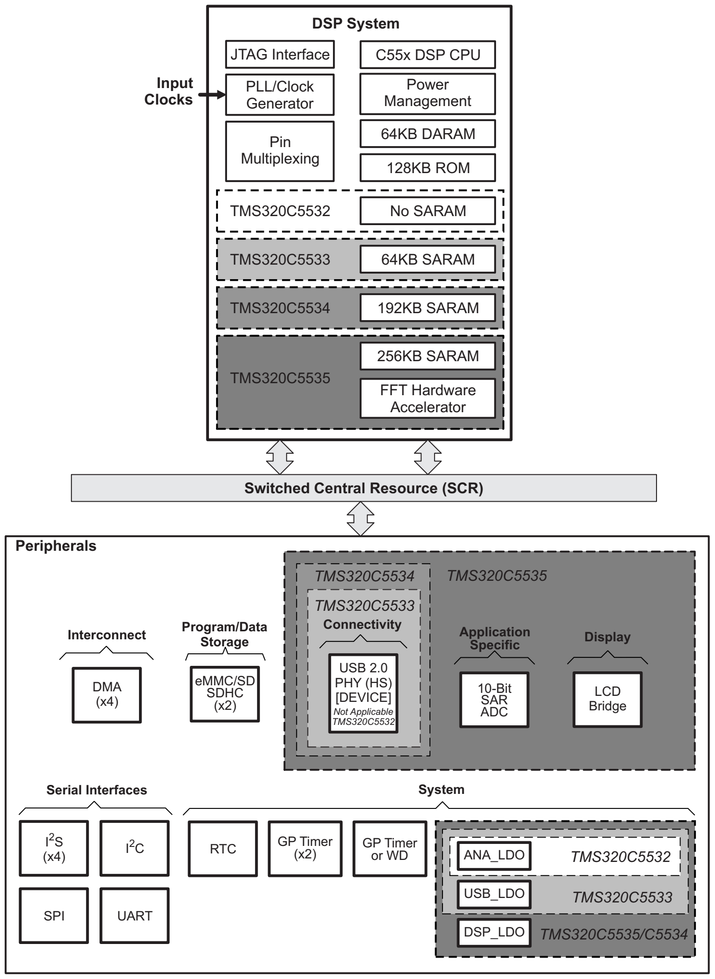


Figure 1. C5535 and C5545 SoC Architecture

2 Block Diagram

Figure 2 shows the block diagram of the demonstration.

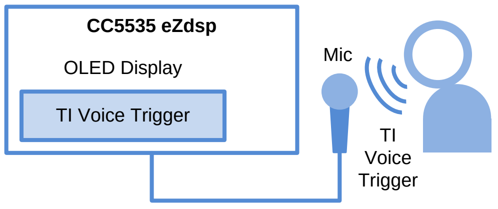


Figure 2. Demonstration Block Diagram

2.1 Highlighted Products

The reference design features the following devices:

2.1.1 C5535 and C5545 DSP

- Core:
  - High-performance, low-power, TMS320C55x fixed-point digital signal processor
    - 20-, 10-ns instruction cycle time 50- (60- on C5545), 100-MHz clock rate
    - One or two instructions executed per cycle
    - Dual multiply-and-accumulate units (up to 200 million multiply-accumulates per second [MMACS])
    - Two arithmetic and logic units (ALUs)
    - Three internal data and operand read buses and two internal data and operand write buses
    - Software-compatible with C55x devices
    - Industrial temperature devices available
  - 320KB of zero-wait state on-chip RAM composed of the following:
    - 64KB of dual-access RAM (DARAM), 8 blocks of 4K × 16 bit
    - 256KB of single-access RAM (SARAM), 32 blocks of 4K × 16 bit
    - 128KB of zero wait-state on-chip ROM (4 blocks of 16K × 16 bit)
  - Tightly coupled FFT hardware accelerator
- Peripheral:
  - Direct memory access (DMA) controller
    - Four DMA with 4 channels each (16 channels total)
  - Three 32-bit general-purpose (GP) timers
    - One selectable as a watchdog or GP
  - Two embedded multimedia cards (eMMCs) or secure digital (SD) interfaces
  - Universal asynchronous receiver/transmitter (UART)
  - Serial port interface (SPI) with four chip selects
  - Master and slave inter-intergrated circuit (I^2C bus)
  - Four inter-IC sound (I2S bus) for data transport
  - Device USB port with integrated 2.0 high-speed PHY that supports the following:
    - USB 2.0 full- and high-speed devices
  - LCD bridge with asynchronous interface
  - 10-bit, 4-input successive approximation (SAR) analog-to-digital converter (ADC)
  - IEEE-1149.1 (JTAG) boundary-scan-compatible
  - 32 general-purpose I/O (GPIO) pins (multiplexed with other device functions)

    - Configures up to 20 GPIO pins simultaneously
- Power:
  - Four core isolated power supply domains:
    - Analog
    - RTC
    - CPU and Peripherals
    - USB
  - Three I/O isolated power supply domains:
    - RTC I/O
    - USB PHY
    - DVDDIO—Three integrated LDOs (DSP_LDO, ANA_LDO, and USB_LDO) to power the isolated domains: DSP core, Analog, and USB core, respectively
  - 1.05-V core (50 MHz), 1.8-, 2.5-, 2.75-, or 3.3-V I/Os
  - 1.3-V core (100 MHz), 1.8-, 2.5-, 2.75-, or 3.3-V I/Os
- Clock:
  - Real-time clock (RTC) with crystal input, separate clock domain, and separate power supply
  - Low-power software programmable phase-locked loop (PLL) clock generator
- Bootloader:
  - On-chip ROM bootloader (RBL) to boot from SPI EEPROM, SPI serial flash or I2C EEPROM eMMC, SD, SDHC, UART, and USB
- Package:
  - 144-terminal Pb-free plastic ball grid array (BGA) (ZHH Suffix)

For more information on each of these devices, see the respective product folders at www.TI.com.

2.1.2 C5535 eZdsp USB Development Kit

The TMDX5535eZdsp is a small form factor, low-cost USB-powered DSP development kit that includes hardware and software required to evaluate the C553x generation, which is the lowest-cost and lowest-power 16-bit DSP of the industry. This low-cost kit allows quick and easy evaluation of the advanced capabilities of the C5532, C5533, C5534, C5535 and C5545 processors. The kit has an on-board XDS100 emulator for full source-level debug capability and supports Code Composer Studio™ (CCS).

Key features:

- Small form factor DSP development kit for the C5535 and C5545 processors
- TMS320C5535 fixed-point, ultra-low-power DSP
- Embedded XDS100 emulator
- 8-MB serial flash memory
- TLV320AIC3204 programmable low-power stereo audio codec
- USB 2.0 high speed
- microSD™ card slot with 2-GB micro SD card
- Line-in/Mic-in and headphone-out audio jacks
- Earphone with mic
- 60-pin expansion connector
- 96 × 16-pixel OLED display
- Two push buttons

Figure 3 shows the C5535 eZdsp kit.

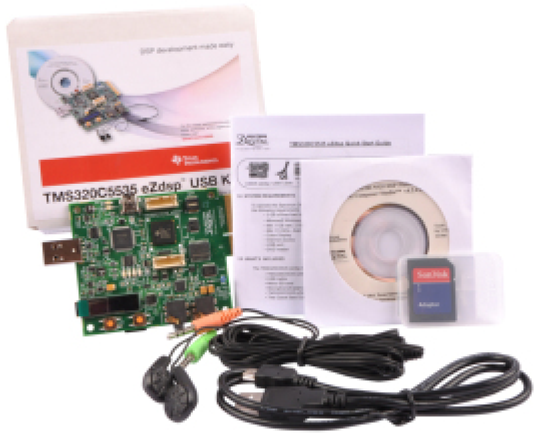


Figure 3. C5535 eZdsp Kit

2.1.3 TLV320AIC3204 Ultra-Low-Power Stereo Audio Codec

The TLV320AIC3204 device (also called the AIC3204) is a flexible, low-power, low-voltage stereo audio codec with programmable inputs and outputs, PowerTune™ capabilities, fixed predefined and parameterizable signal-processing blocks, integrated PLL, integrated LDOs, and flexible digital interfaces.

Features:

- Stereo audio DAC with 100-dB SNR
- 4.1-mW stereo, 48-ksps DAC playback
- Stereo audio ADC with 93-dB SNR
- 6.1-mW stereo, 48-ksps ADC record
- PowerTune
- Extensive signal-processing options
- Six single-ended or three fully-differential analog inputs
- Stereo analog and digital microphone inputs
- Stereo headphone outputs
- Stereo line outputs
- Low-noise PGA
- Low-power analog bypass mode
- Programmable microphone bias
- Programmable PLL
- Integrated LDO
- 5 mm × 5 mm 32-pin QFN package

3 System Design Theory

The speech recognition reference demonstration uses the TI embedded speech recognition library (TIesr) and leverages the high-performance and low-power DSP core of the C5535 and C5545 devices to process the microphone input and respond to a preprogrammed phrase. The following sections present the theory of operation of the demonstration.

3.1 TI Embedded Speech Recognizer

TIesr is a Hidden Markov Model (HMM)-based automatic speech recognizer (ASR) that is speaker independent. TIesr contains an efficient fixed-point implementation of the grammar-based recognizer that requires reduced memory and processing. The algorithm contains excessive background noise suppression and compensation. It also contains confidence measuring to allow for flexibility in misrecognition. The voice activity detection decisions are based on the following:

- Energy in the signal
- Autocorrelation
- Matching to silence acoustic models

Figure 4 shows the TIesr API architecture.

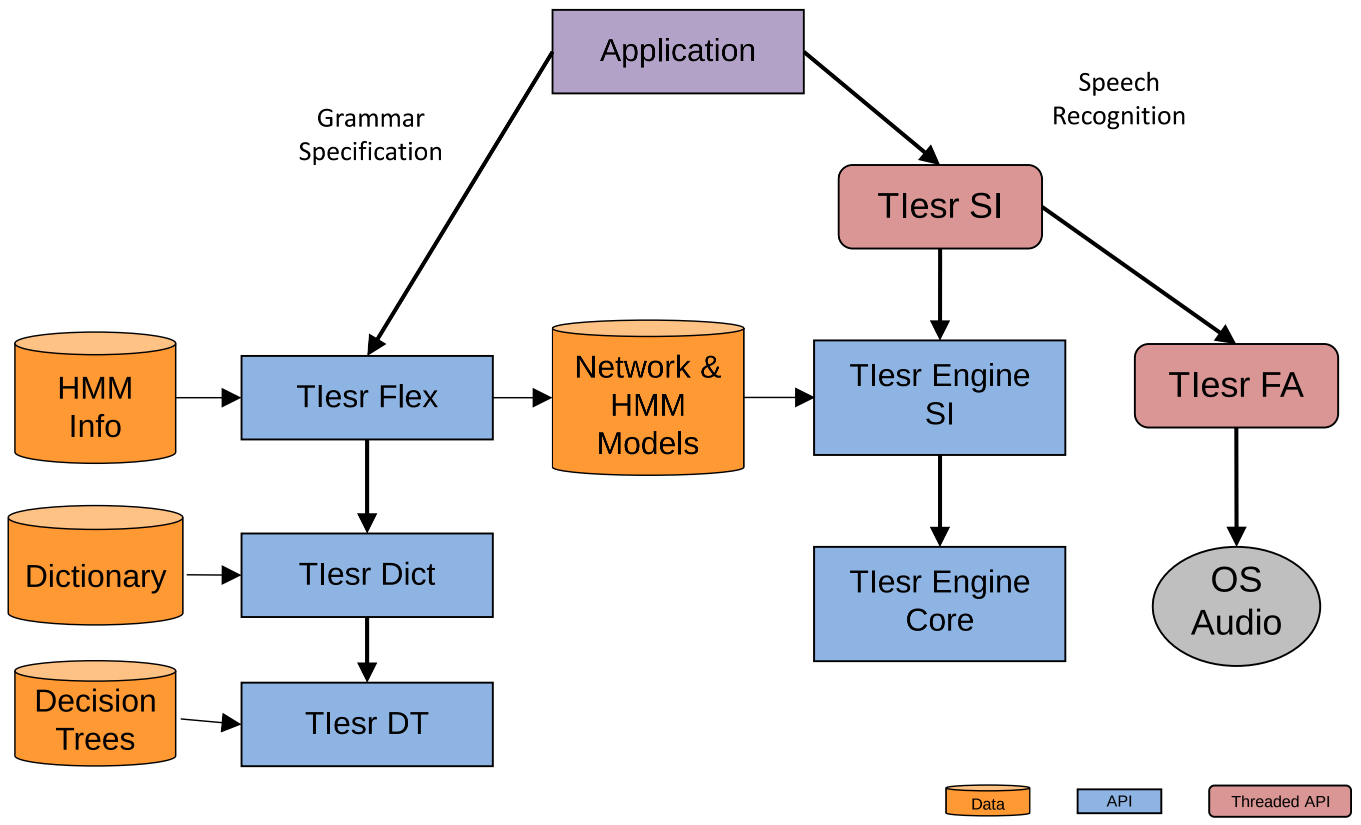


Figure 4. TIesr API Architecture

Features:

- Fixed point representation:
  – Offers10-mfcc, 10-delta, and 20-ms frames
  – Offers representation in short or byte
  – Offers dynamic byte scaling based on current active grammar

- Voice activity detection:
  - Does not send features to recognizer if data not required
  - Decides based on the following:
    - Energy in signal
    - Autocorrelation
    - Matching to silence acoustic models

3.2 Demonstration Software Architecture

The C5535 and C5545 real-time TIesr demo functions by using TI-RTOS to divide various tasks into threads. Table 1 lists the summary of the demonstration software architecture. The demonstration is split into threads as listed in Table 1,

Table 1. Demonstration Software Architecture Summary

Thread number	Priority (0 highest)	Name	Function
1	1	User Input	• START, STOP functions • Fired by timer ISR • SAR polling rate fixed by timer intr
2	0	Audio Data Collection	• Ping pong buffer → input circular buffer • Fired by DMA ISR • Rate fixed by DMA (I2S sync) intr • Fires recognizer thread (for example, SWI_inc())
3	2	Recognizer	• Read audio frame from input circular buffer • Run recognizer on input audio frame • If recognition result, run JAC_update() and get recognizer results • Fired by audio data collection
4	3	Output Display	• Output results to LCD • Fired by recognition thread (for example, SWI_post() or SEM_post())

When the demonstration is executed, there are several states that the system enters based on the task. The following is a summary on the various tasks the demonstration executes.

Idle

User Input active thread

- Waiting for user to input command from push-button (SW2) network.
- If START command is received, transition to Start Audio Data Collection state.
- If STOP command received, remain in Idle state.

Audio Data Collection active thead

- ADC output not written to input circular buffer. Input circular buffer is empty.

Recognizer not active thread

- Recognizer state is up-to-date with respect to all previous recognition results.

Output Display may be active thread

- Output of previous recognition results to LCD may be in progress.

Start Audio Data Collection

User Input active thread

- Software controls updated for the following:
  - ADC output is written to input circular buffer.
  - Thread: Recognizer is notified when a new audio frame is available.

Audio Data Collection active thread

- ADC output is not written to input circular buffer. Input circular buffer is empty.

Recognizer not active thead

- Recognizer state is up-to-date with respect to all previous recognition results.

Output Display may be active thread

- Output of previous recognition results to LCD may be in progress.

Recognizing

User Input active thread

- Waiting for user to input command from push-button network.
- If START command is received, remain in Recognizing state.
- If STOP command is received, transition to Idle state.

Audio Data Collection active thread

- ADC output is written to input circular buffer.

Recognizer active thread

- Recognizer is consuming audio frames from input circular buffer.
- The number of input audio frames received before previous recognition processing is completely transmitted to Recognizer thread through global variable or SWI mailbox (SWI_inc())
- Recognizer thread remains in outer multiple utterance recognition loop until one of following conditions are met:
  - A STOP command is received.
  - All audio frames are processed.
- Recognizer thread remains in inner single utterance recognition loop (CallSearchEngine() and SpeechEnded()) until one of following conditions are met:
  - A STOP command is received.
  - All audio frames are processed and no recognition result is found.
  - A recognition result is found.
- If the inner and outer loop terminated because a STOP command was received:
  1. Updates the software controls so the ADC output buffer is not written to circular buffer, and Thread: Recognizer is not notified when a new audio frame is available.
  2. Clears the input circular buffer.
  3. Exits the recognizer thread and transitions to an idle state.
- If the inner or outer loop terminated because all audio frames are processed and no recognition result is found, exit the recognizer thread and remain in the recognizing state.
- If the inner loop is terminated because a recognition result is found, perform recognition post-processing:
  1. Calls JAC_update().
  2. Obtains and prepare recognition results for display.
  3. Forwards the recognition results to Output Display thread.

3.3 Hardware

The C5535 and C5545 TIesr demonstration uses the C5535 ezdsp EVM. Speech is captured through the microphone connected to Stereo IN (J3) of the eZdsp. The speech is processed through the audio codec (AIC3204) and passed onto the I2S port of C5535 DSP through the Octal FET bus switch (SN74CBTLV3245A).The 5535 eZdsp is powered through the USB connector J2 that also has the XDS100 emulator for CCS debugging.

When the demonstration is executed as described in Section 5, the system waits for a button press from the user on SW2 to arm the system. LED DS2 flashes rapidly to indicate that the system is waiting for microphone input. When the keyword is recognized, the phrase is printed on the OLED.

4 Getting Started Hardware

This section lists all of the hardware required to get started. The parts used in this design can be ordered from http://www.ti.com.

4.1 C5535 eZdsp USB Stick Development Kit

To purchase the C5535 eZdsp USB Stick Development Kit, see http://www.ti.com/tool/tmdx5535ezdsp. The kit includes a microphone that is required to run the demonstration. The product page also contains links to schematics and other hardware related resources. For the C5535 eZdsp Voice Trigger demonstration setup, see Figure 5.

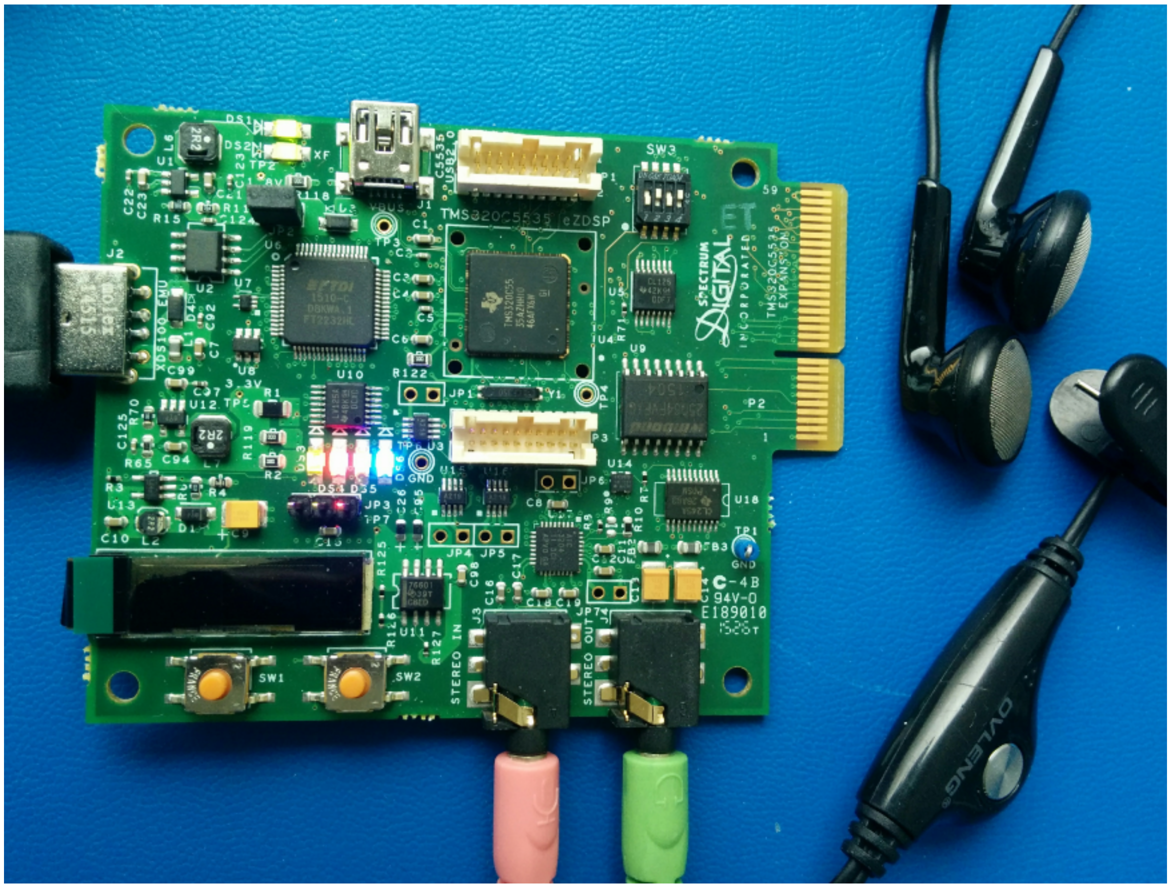


Figure 5. C5535 eZdsp Voice Trigger Demonstration Setup

5 Getting Started Firmware

5.1 Running the Prebuilt Voice Trigger Demonstration

This section describes the steps to run the voice trigger demonstration on the C5535 ezdsp, using the pre-built binary. Section 5.2 shows how to build and run this binary. Section 5.3 shows how to customize the trigger phrase.

5.1.1 Software Requirements

To get started with the demonstration, the following software is required:

- CCS (demonstration was tested with v6.1.2)
- TIesr Speech recognition software package

5.1.2 Hardware Requirements

To get started with the demonstration, the following hardware is required:

- A C5535 eZdsp device with a microphone
- A PC with a Windows® 7 operating system

5.1.3 Running the Demonstration in CCS

To build and run the keyword trigger demonstration on the C5535 ezdsp, do as follows:

1. Open CCS.
2. Proceed to Section 5.2.3, Step 14.

   The voice trigger phrase *TI voice trigger* displays on the OLED.

5.2 Building and Running the Voice Trigger Demonstration

This section describes how to build and run the voice trigger demonstration on the C5535 ezdsp and how to customize the trigger phrase.

5.2.1 Software Requirements

To get started with the demonstration, the following software components are required:

- CCS (the demonstration was tested with v6.1.2)
- DSP/BIOS™ real-time operating system v5.42.0.7
- TIesr Speech recognition software package
- C55XCSL-LOWPOWER-3.00.00.02 (This design assumes the CSL is installed at the following Windows path: C:\c55xx_csl_3.00)

Setting up and getting started with CCS is beyond the scope of this document. For more information, see the getting started guides on the CCS download page. TI assumes the user understands how to import, build, and run CCS projects.

5.2.2 Hardware Requirements

To get started with the demonstration, the following hardware components are required:

- A C5535 eZdsp device with a microphone
- A PC with a Windows 7 operating system

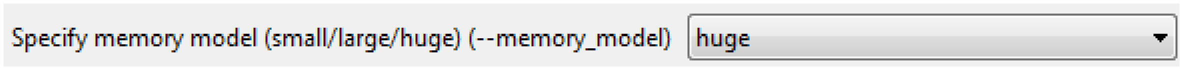
5.2.3 Building and Running in CCS

To build and run the keyword trigger demonstration on the C5535 eZdsp, do as follows:

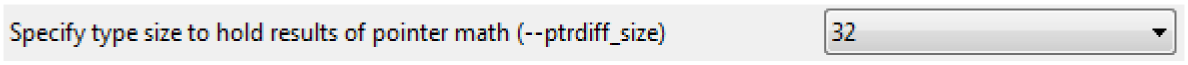
1. Open CCS.
2. Navigate to /TIesr_src/TIesr_C55_demo.
3. Import the following projects into the CCS workspace:
   - TIesrDemoC55
   - TIesrEngineC55

NOTE: Ensure that DSP/BIOS real-time operating system v5.42.0.7 is installed before building the project.

4. Navigate to the CSL source folder installed in Section 5.2.1.
5. Import the following projects into the CCS workspace:
   - cslVC5505
   - atafs_bios_drv_lib
6. Navigate to Properties→Build→C5500 Compiler→Processor Options in atafs_bios_drv_lib.
7. Change Specify memory model to *huge*.


Specify memory model (small/large/huge) (--memory_model) huge

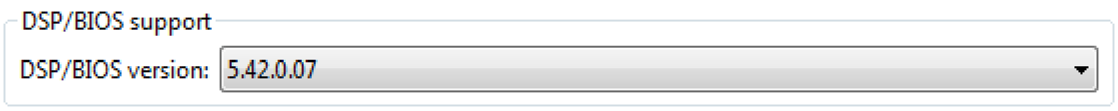
8. Change Specify type size to hold results of pointer math to *32* if not specified inProperties→Build→C5500 Compiler→Advanced Options→Runtime Model Options.


Specify type size to hold results of pointer math (--ptrdiff_size) 32

9. Uncheck *use large memory model* if checked.
10. Repeat Steps 6 through 8 for cslVC5505.
11. Navigate to C:\c55xx_csl_3.00\inc\csl_general.h change #define CHIP_C5517 to //#define CHIP_C5517.
12. Right-click on TIesrDemoC55.
13. Select Build Project.

    This action creates TIesrDemoC55.out in the Debug folder.

NOTE: If there are build errors because of missing DSP/BIOS versions in atafs_bios_drv_lib or any of the other projects, ensure that CCS registers the previously installed DSP/BIOS real-time operating system v5.42.0.7 when it discovers newly installed products at start-up. Figure 6 shows the DSP/BIOS version used by atafs_bios_drv_lib under Properties→General.


DSP/BIOS support

DSP/BIOS version: 5.42.0.07

Figure 6. Screenshot of DSP/BIOS Version in Project Properties

14. Create a target configuration for the C5535ezdsp.
15. Launch the configuration with the microphone connected to STEREO IN J3.
16. Connect to the DSP core with the C5535 or C5545 GEL file initialized.
17. Navigate to Run→Load→Load Program.
18. Load /TIesr_src/TIesr_C55_demo/build/ccsv5/Debug/TIesrDemoC55.out on to the core.
19. Click Resume.

The demonstration executes on the EVM.

20. Press the SW2 button.

    The DS2 LED flashes rapidly indicating the system is ready to accept the keyword (see Figure 7). Press the SW2 button twice if, not registered the first time.

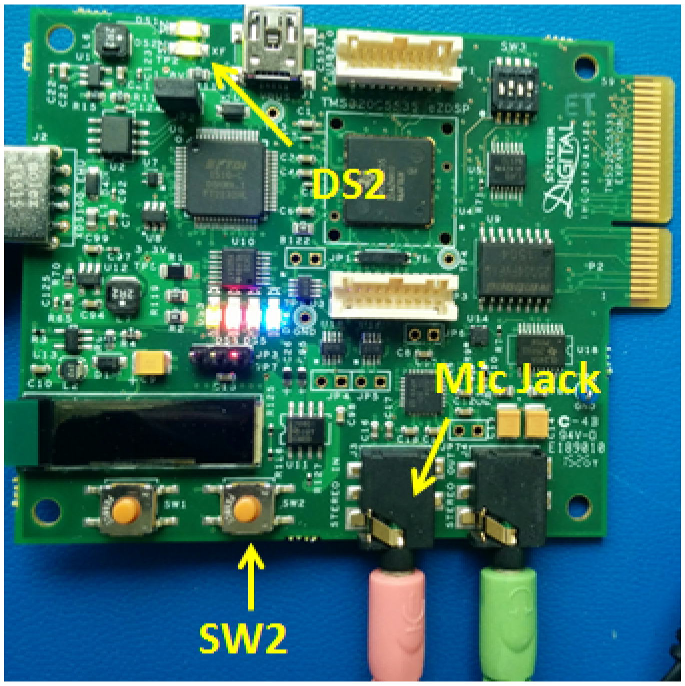


Figure 7. LED, Button, and Microphone Jack

21. Say *TI voice trigger* into the microphone.

The OLED shows this phrase indicating that the system recognizes the keyword phrase. The system enters listening mode again after a few seconds (For the OLED display, see Figure 8).

NOTE: Enunciate the trigger word, especially the T and I with short pauses in between. _FILL might also appear on the screen indicating that the _FILL word is recognized. This happens when a word that the recognizer does not understand is spoken or can be due to background noise.

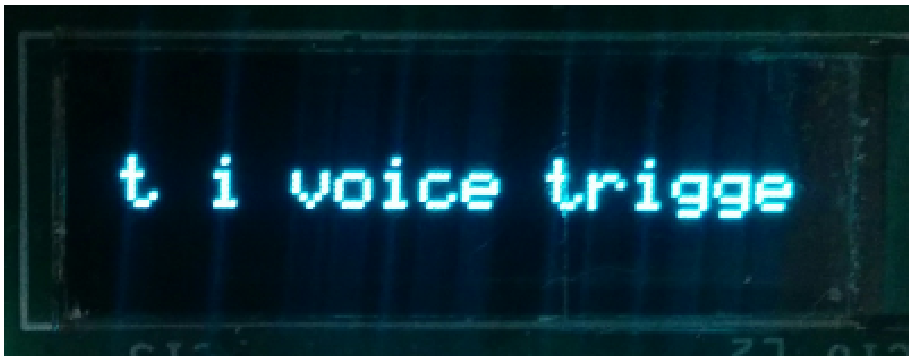


Figure 8. OLED on C5535 eZdsp After the Keyword Phrase is Recognized

5.3 Customizing the Trigger Phrase

This section describes how to customize the trigger phrase.

5.3.1 Software Requirements

The TIesr speech recognition software package is required.

5.3.2 Hardware Requirements

This design requires the following hardware:

- Linux® Desktop PC with Ubuntu 12.04 LTS
- A PC with Windows 7 operating system

5.3.3 Building the Modified TIesr Engine

To build a modified TIesr engine, do as follows:

1. Copy the source file (/TIesr_src/) to your Ubuntu PC
2. Navigate to /TIesr_src/TIesr_model_build/ on the console.
3. Execute the following command: $make LinuxDebugGnu

   This action creates a directory called Dist structured as shown in Figure 9. The model files are used in the following step to generate the trigger phrase .bin files.

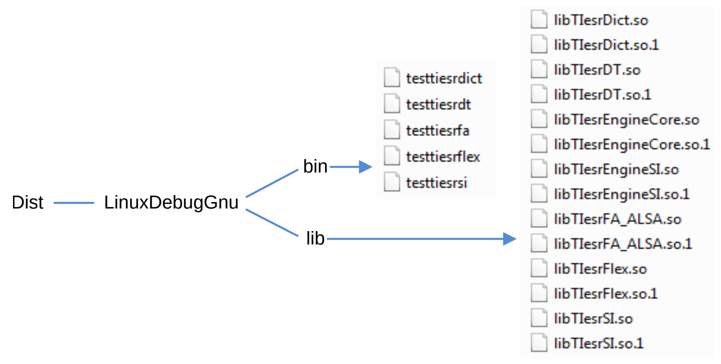
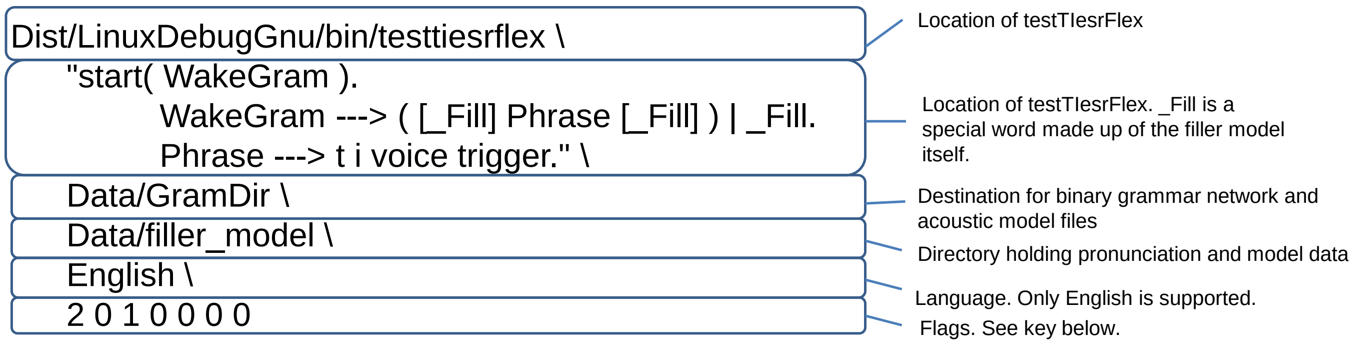


Figure 9. Directory Structure of Dist

4. Open /TIesr_src/TIesr_model_build/build_files.sh for editing.

   This file contains the grammar callers to feed into testTIesrflex to generate the .bin recognition files. Choose keywords or phrases that are long and contain rich acoustic content.Figure 10 shows a description of the build_files.sh contents.



```
Dist/LinuxDebugGnu/bin/testtiesrflex \
    "start( WakeGram ).
            WakeGram ---> ( [_Fill] Phrase [_Fill] ) | _Fill.
            Phrase ---> t i voice trigger." \
    Data/GramDir \
    Data/filler_model \
    English \
    2 0 1 0 0 0 0
```

Location of testTIesrFlex
Location of testTIesrFlex. _Fill is a special word made up of the filler model itself.
Destination for binary grammar network and acoustic model files
Directory holding pronunciation and model data
Language. Only English is supported.
Flags. See key below.

Flag Key:

max_pron: Maximum pronunciations per word to include in output grammar network
inc_rule: Flag indicating to include decision tree rule pronunciation
auto_sil: Flag indicating to include optional silence between words
lit_end: Flag output files in little endian format
byte_mean: Output acoustic probability mean vectors as byte data
byte_var: Output acoustic probability variance vectors as byte data
add_close: (optional; default enabled) Add closure phones prior to stop consonants

Figure 10. build_files.sh Structure

Several phrases can also be used instead of a single phrase. The following is an example using multiple phrases as it would be written in build_files.sh.

NOTE: TIesr engine is currently limited to a maximum of two keyword phrases.

"start(WakeGram).

WakeGram ---> ([_Fill] Phrase [_Fill]) | _Fill.

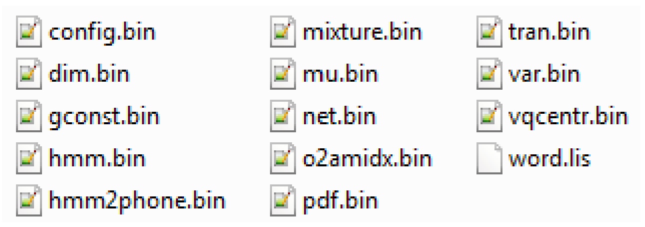
Phrase ---> t i voice trigger | ti technology."\

5. Run the build_files.sh script.

   The file permissions may need to be changed to run this script. The following code shows the console output if the script executes correctly.

```
$ ./build_files.sh
Loading language, grammar, and model data
Parsing input grammar
Writing out binary grammar network and model files
Grammar network and model files output successfully
```

   This script outputs a file set like Figure 11 shows in /TIesr_src/TIesr_model_build/Data/GramDir.



config.bin	mixture.bin	tran.bin
dim.bin	mu.bin	var.bin
gconst.bin	net.bin	vqcentr.bin
hmm.bin	o2amidx.bin	word.lis
hmm2phone.bin	pdf.bin	

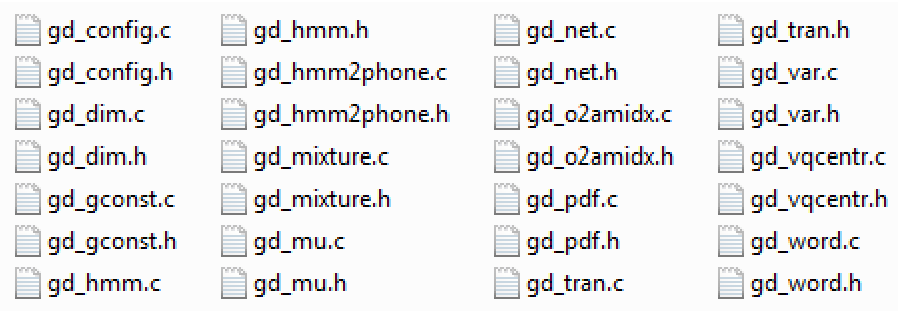
Figure 11. Files Generated After Running build_files.sh

6. Navigate to /TIesr_src/TIesr_C55_demo/TIesrBintoASCII.

   This directory contains the TIesrBintoASCII utility that convertd the .bin files generated in Step 5, to .c and .h files that CCS later uses to build the demonstration.

7. Run convert.sh (permission changes to run this script may be required).

   Running this script executed the TIserBintoASCII utility to convert the files and copy the generated .c and .h files into /TIesr_src/TIesr_C55_demo/C55xTIesrData/GramKWS.



gd_config.c	gd_hmm.h	gd_net.c	gd_tran.h
gd_config.h	gd_hmm2phone.c	gd_net.h	gd_var.c
gd_dim.c	gd_hmm2phone.h	gd_o2amidx.c	gd_var.h
gd_dim.h	gd_mixture.c	gd_o2amidx.h	gd_vqcentr.c
gd_gconst.c	gd_mixture.h	gd_pdf.c	gd_vqcentr.h
gd_gconst.h	gd_mu.c	gd_pdf.h	gd_word.c
gd_hmm.c	gd_mu.h	gd_tran.c	gd_word.h

Figure 12. Files Generated at /TIesr_src/TIesr_C55_demo/C55xTIesrData/GramKWS after running convert.sh.

8. If on a Windows PC, copy the files generated in the previous step to /TIesr_src/TIesr_C55_demo/C55xTIesrData/GramKWS.

   The following step uses this new file set in the CCS build.

9. Repeat Steps 12 through 24 in Section 5.2.3, *Building and Running in CCS*.

   The modified trigger word files are taken from GramKWS that was updated in the previous step.

10. Use this phrase in the demonstration.

5.3.4 Software Disclaimer

The software provided is for demonstration purposes only. This software is not intended for production systems. For licensing information, see the manifest Software_manifest.html in the source root folder at /TIesr_src/.

6 Design Files

6.1 Schematics

To download the schematics for the C5535 eZdsp, see http://www.ti.com/tool/TIDEP0066.

6.2 Bill of Materials

To download the bill of materials for the C5535 eZdsp, see http://www.ti.com/tool/TIDEP0066.

6.3 Gerber Files

To download the schematics for the C5535 eZdsp, see http://www.ti.com/tool/TIDEP0066.

7 References

MathWorks® Introduction to Hidden Markov Models, http://www.mathworks.com/help/stats/hidden-markov-models-hmm.html

8 Terminology

- TIesr: Texas Instruments Embedded Speech Recognition
- Filler_model: Speech model used by testTIesrflex to generate non-vocabulary speech data.
- EVM: Evaluation module

9 About the Author

LALINDRA JAYATILLEKE is an embedded processing applications engineer at TI.

10 Acknowledgements

LORIN NETSCH is currently with MSP430 embedded software development at TI. Lorin was part of the former speech and audio technologies laboratory that built the TIesr engine and it’s API.

FRANK LIVINGSTON is an embedded processing DSP engineer at TI. He is the original author of the TIesr C55x demonstration.

RAN KATZUR is an embedded processing applications engineer at TI He authored the BintoASCII utility that converts the TIesr generated bin files to source that CCS can use.

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (March 2016) to A Revision	Page
• Added Note.	16
• Updated Flag Key Phrase.	17