Application Report
SLAA405B–February 2009–Revised March 2018

Low-Cost Speech With MSP430™ MCUs

Peter Forstner *MSP430 Applications*

ABSTRACT

This application report describes the implementation of a low-cost solution to store and play audio or polyphone ringtones with an MSP430™ MCU. An example solution for flexible and low-cost audio playback is chosen using a demonstration board that is based on the MSP430 USB Stick Development Tool and the MSP430FG4618 Experimenter Board. An additional example of low-cost speech with MSP430 MCUs can be found in the Voice Band Audio Playback Using a PWM DAC reference design.

Source code and sample files are available for download from http://www.ti.com/lit/zip/slaa405.

Contents

Trademarks

1 Introduction

Audio playback capabilities differentiate a product from a competitor's solution. A talking smoke detector can directly give detailed instruction on how to leave the building, and this feature can save lives in case of fire. Polyphone ring tones replacing simple beeps in an application make the final product more convenient. But usually speech or polyphone ring tones require either external speech or audio circuits or an expensive microcontroller with lots of flash memory and a DAC output.

This application report discusses possible technical solutions for speech output with a low-cost microcontroller. The final solution that is selected uses an external SPI flash memory, which enables flexibility of storage memory size based on the amount of playback time needed. The low-cost MSP430F2002 is suitable for this ultra-low-power speech playback solution. Because of the small code size and the minimal requirements for peripheral hardware, speech output can be added to any application using almost any other MSP430 microcontroller.

2 Storage of Audio Data

Audio data must be stored in nonvolatile memory, and flash memory is widely used in this case. There are two choices for storage of audio data in flash: MCU internal flash memory or external flash memory.

Audio quality and length are key parameters for selection of the correct flash memory size. An audio sampling rate of 8 ksps theoretically allows, according to Nyquist, an analog audio frequency of up to 4 kHz, which is good enough for understandable speech. A resolution of 8 bit (1 byte) is often used for speech playback, and this leads to a quick calculation that gives the required memory space for uncompressed audio data:

8 ksps × 1 byte = 8 kbytes per second

2.1 Storing Audio Data in Internal Flash Memory

The advantage of storing audio data in internal flash is clearly the simplicity of the system and the single-chip solution. One single MSP430 MCU can store software and audio data and can process and output the audio signal. Also, access to the audio data in flash is very simple for the CPU.

The main problem of this solution is the required memory space. At 8 ksps / 8-bit, an MSP430 MCU with 60-KB flash can store up to only seven seconds of audio data in 56 KB and then has just 4 KB left for code. In some simple applications with only a few seconds of audio playback, this might be a good solution. However every MSP430 MCU with 60 KB flash comes with an extensive set of peripherals, such as two USCI modules, ADC12, and two timers, which might not be needed in the application. Even the MSP430 derivatives with up to 256 KB of flash can store up to only 32 seconds of uncompressed audio data, which in many cases is not enough time for playback.

2.2 Storing Audio Data in External Flash Memory

External flash memory for the storage of audio data offers more flexibility. The selection of the flash size of the MSP430 MCU and set of peripherals is dependant only on the requirements of the application and is not dependent on the amount of audio data. If the application does not require many other tasks in addition to audio playback, a very small and low-cost MSP430 MCU, such as the MSP430F2002, can be used. Also, memory size of the external flash for audio data storage can be freely adapted to the required playback time. If more audio seconds or minutes are necessary, a memory upgrade is very easy and only the external flash memory chip must be replaced with a version with more memory.

The disadvantage of external flash memory is the required board space of this two-chip solution and the more complicated access of audio data stored in the external flash memory.

Easy interfaces to external flash are SPI and I2C and these two serial interface types are supported by nearly all new MSP430 derivatives with an USI or USCI interface. If SPI is used, all MSP430 derivatives with the USART interface can be selected to implement speech playback with external flash memory. This makes the solution with the SPI interface the most flexible, because it is useable on nearly all MSP430 derivatives.

3 Audio Playback

For audio playback, a digital-to-analog conversion is necessary. An alternative to using the MSP430's DAC12 is using PWM output with a timer. Both solutions have advantages and disadvantages.

3.1 Audio Playback With DAC12

Using the DAC12 is the straight-forward solution. The audio samples simply must be applied to the DAC12 with the correct timing, and the DAC12 directly generates the desired analog waveform. DAC12 is a peripheral that is available in only a few, and typically the full-featured, MSP430 derivatives. But if the application already requires an MSP430 MCU that is equipped with DAC12, this is the best way to convert digital audio data into an analog waveform.

DAC12 is a 12-bit DAC that can also operate in an 8-bit mode. This 8-bit mode allows the use of the DMA, if available. Figure 1 shows the signal flow for audio playback with internal flash and DMA use. In this case, the correct trigger sequence is very important:

Timer_A generates the sampling rate of 8 kHz and triggers the double-buffered DAC12 to output the next analog value. This means that the value in register DAC12_xDAT is copied at the correct time into register DAC12_xLATCH. DAC12 now detects an empty register DAC12_xDAT and triggers the DMA to load DAC12_xDAT with the next value from flash memory. It is important to build the trigger sequence in this way, because this is the only method that ensures correct timing when outputting new values using DAC12. Intuitively, many programmers would select a different trigger sequence, in which Timer_A triggers the DMA to copy the next digital audio sample to the DAC. This method generates uncertainties in the timing, because the DMA can only copy the next audio value to DAC12 when it has control of the internal data and address bus of the MSP430 MCU. This bus might be occupied by the CPU or other DMA channels and a delay of this data transport is likely. Timing delays when reloading values to DAC12 generate distortions in the analog wave form, which can be avoided if the correct trigger sequence is used.

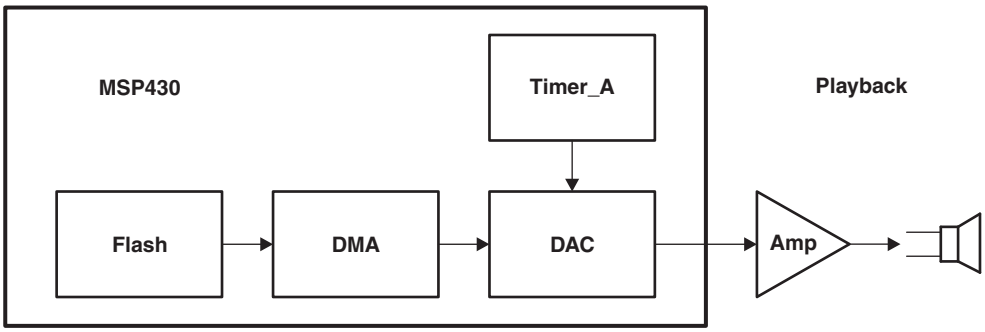

Figure 1. Speech Output With Internal Flash and DAC

If DMA is not available, it must be replaced with an interrupt service routine (ISR) triggered by an empty register DAC12_xDAT of DAC12. In that case, the CPU assumes responsibility for moving the digital audio data from flash memory to the DAC12. The ISR method also allows processing of audio data during audio output; e.g., restoring compressed audio data.

If external SPI flash memory is used, the data must be read from the serial interface USI, USCI, or USART and written to the DAC12_xDAT register of ADC12. This task also can be fulfilled by either the DMA or the CPU executing an ISR.

3.2 Audio Playback With PWM

PWM is often used as low-cost solution for a digital-to-analog conversion. Analog audio also can be generated by PWM, and Figure 2 shows the signal flow for a PWM output with audio data stored in internal flash memory. For more information about audio playback using a PWM DAC, see the Voice Band Audio Playback Using a PWM DAC reference design.

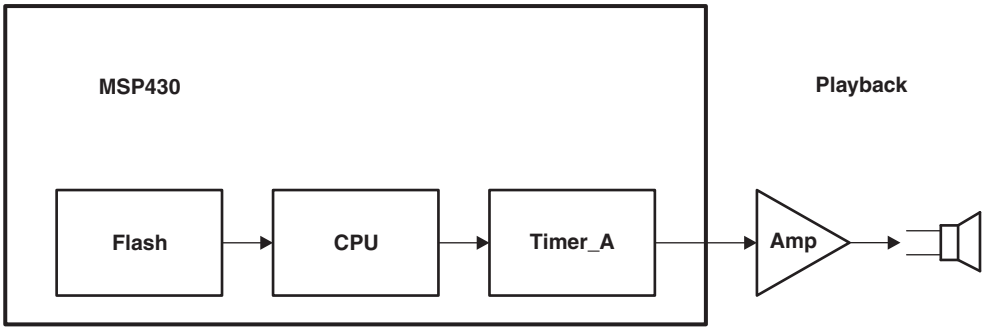

Figure 2. Speech Output With Internal Flash and PWM

The first step in this solution is the selection of the correct frequencies. For speech playback, one fixed parameter is the 8-ksps sampling rate for digital audio storage. If Timer_A generates a PWM with a frequency of 8 kHz, a very steep analog filter is necessary to suppress the 8-kHz PWM frequency and pass only the 0-Hz to 4-kHz analog audio frequency. This problem can be solved easily by selecting a higher PWM frequency. For example 32 kHz is above the audible frequency range and, if it is necessary to filter this frequency, a simple lowpass filter can be implemented, allowing 0 Hz to 4 kHz to pass and suppressing 32 kHz. In addition, 32 kHz is a quadruple of 8 kHz, which makes data calculations on a binary computer easy.

If Timer_A generates a 32-kHz PWM with an 8-bit resolution, an input clock frequency of 32 kHz × 256 = 8.192 MHz is necessary.

The MSP430F2xx family has factory-measured DCO calibration values for 1 MHz, 8 MHz, 12 MHz, and 16 MHz stored in the information flash memory. The DCO calibration value for 8 MHz gives a static systematic error of –2.4%, which is acceptable for audio playback. Using the 8-MHz calibrated DCO frequency allows the generation of audio playback without any external component, such as a crystal with external load capacitors.

The MSP430F5xx family offers an integrated 32768-Hz oscillator, the REFO. Based on this REFO frequency and the built-in frequency-locked loop (FLL), the DCO can be programmed to the desired frequency and the frequency stability is good enough for speech output. Hence, with MSP430F5xx derivatives, audio playback without any external components is also possible.

All other MSP430 MCU families require a crystal (e.g., 32768-Hz watch crystal) to ensure a clock signal with the required frequency accuracy. Figure 3 shows the clock generation for the 32-kHz PWM based on a 8.192-MHz clock signal.

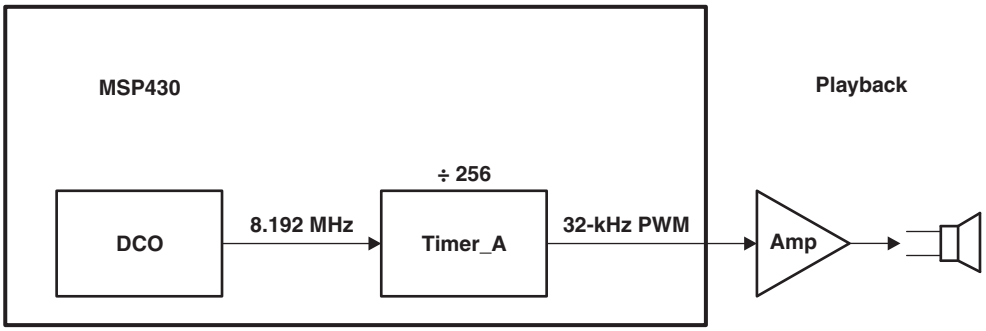

Figure 3. Clock Generation for PWM

With a PWM frequency of 32 kHz and an audio sample rate of 8 ksps, there are four PWM cycles for each audio sample. Out of two audio samples, four PWM duty-cycle values must be generated, which can be done with linear averaging. Two audio samples (s1 and s2) are read from memory, and the three missing audio samples between s1 and s2 are calculated according to Figure 4.

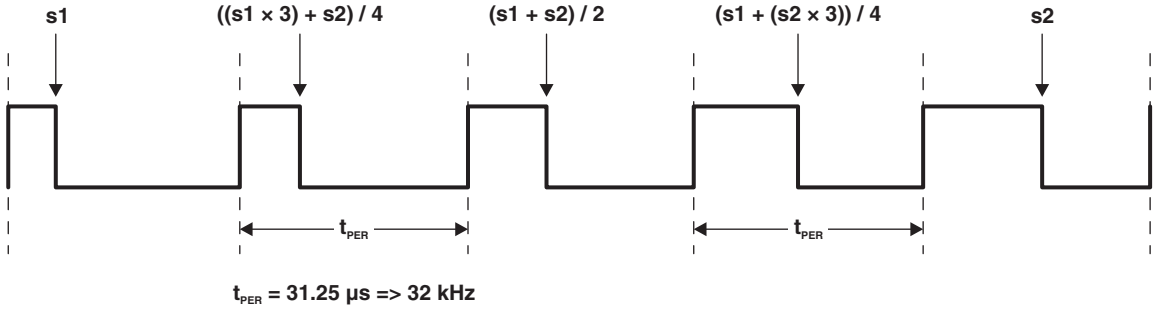

Figure 4. Clock Generation for PWM

This calculation can be implemented in software without the use of any mathematical libraries. Because the PWM frequency 32 kHz is four times the sampling rate, only adding and shifting instructions are necessary. The code in C-language can be done as shown in the following example using the variable uAvgCntr as a modulo-4 counter to count the calculated missing samples:

```
switch (uAvgCntr)
{
  case 0: TACCR1 = uAudioSample1;
          break;
  case 1: TACCR1 = (uAudioSample1 + uAudioSample1 + uAudioSample1 + uAudioSample2) >> 2;
          break;
  case 2: TACCR1 = (uAudioSample1 + uAudioSample2) >> 1;
          break;
  case 3: TACCR1 = (uAudioSample1 + uAudioSample2 + uAudioSample2 + uAudioSample2) >> 2;
          _BIC_SR_IRQ(LPM4_bits); // Clear all LPM bits from 0(SR)
                                  // this restarts audio output loop in main()
                                  // and reads next audio sample from
                                  // SPI-Flash memory
          break;
}
uAvgCntr++;                       // increment averaging counter
uAvgCntr = uAvgCntr & 3;          // averaging counter range: 0 ... 3
```

load next audio sample from SPI Flash in main()

At first, the audio sample s1 is an output via PWM. The next three samples are then calculated from the audio samples s1 and s2. After this process, the next audio sample s3 is loaded from flash into the microcontroller for the next averaging calculation and the cycle repeats from the beginning.

4 eZ430-Speech, A Demonstration

A low-cost and ultra-low-power demonstration has been designed: eZ430-Speech. This demo fits the eZ430 development tool in memory stick form factor as shown in Figure 5.

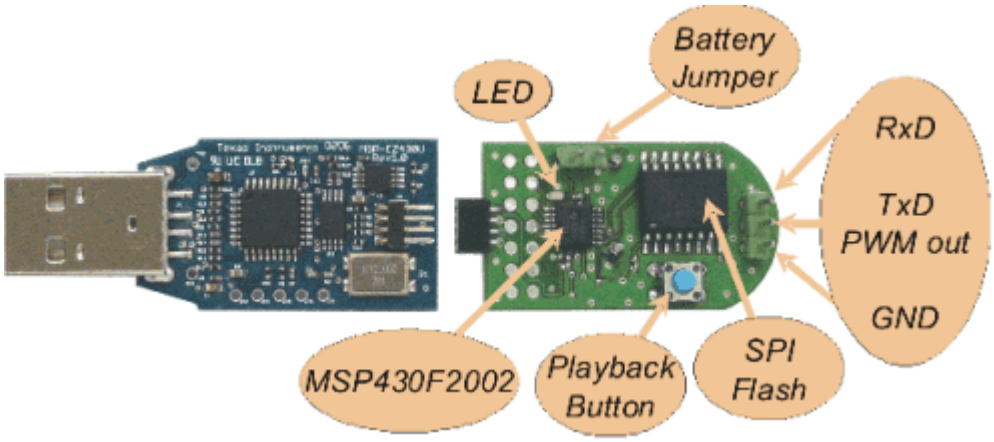

Figure 5. eZ430-Speech With eZ430 Development Interface

4.1 eZ430-Speech Hardware

Because audio playback with external SPI flash and PWM can be done easily with a low-cost MSP430 MCU, the MSP430F2002 has been chosen. To implement a fast serial communication, the USI serial interface is necessary, which is available in the MSP430F2002. The integrated Timer_A2 can be used to generate the PWM output signal for audio output. Because of the calibrated DCO in the MSP430F2xx family, no external crystal is necessary for this application. Figure 6 shows that, in addition to the MSP430F2002 and the SPI flash, the only external components on the board are:

- A switch to start speech output
- An LED to show active operation
- A decoupling capacitor

No other external components necessary.

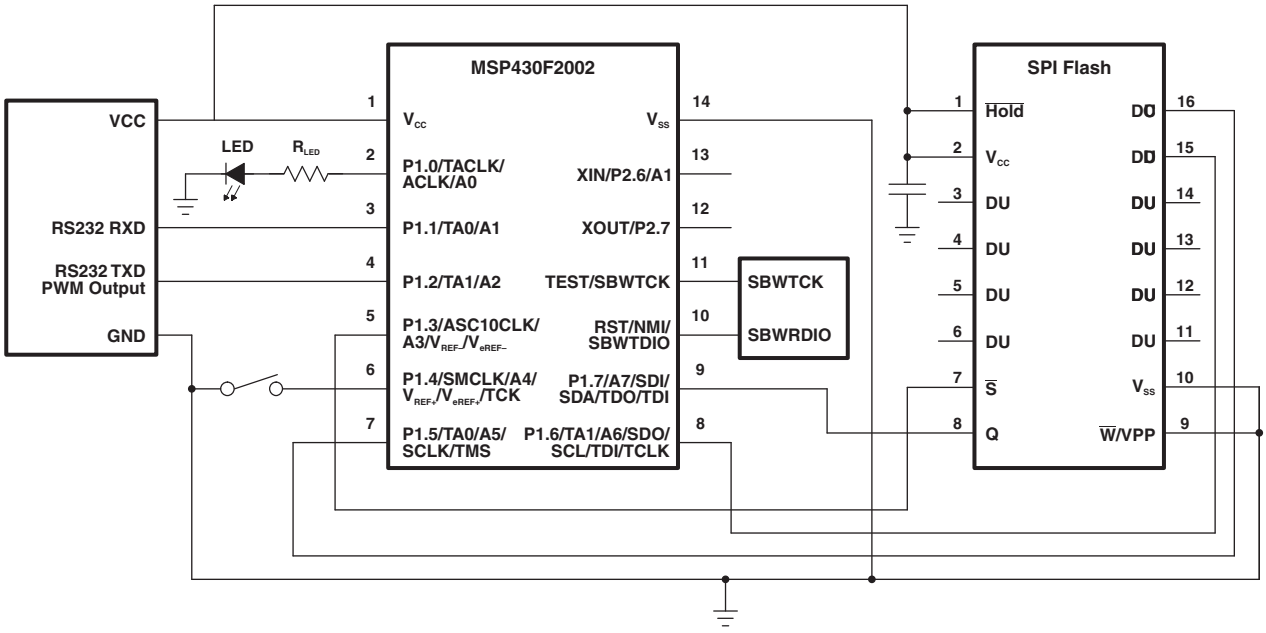

Figure 6. Schematic of eZ430-Speech

The supply current I_{CC} of the SPI flash memory during erasure, programming, and read operations is too high to power it directly from an MSP430 GPIO. An external switch to turn off I_{CC} of the SPI flash could be added, and an MSP430 GPIO pin could control this switch to further reduce power consumption. If no speech is generated, the MSP430 MCU and SPI flash are in deep sleep mode. In this mode, the MSP430 MCU consumes less than 1-µA supply current. Pressing the button wakes the MSP430 MCU, and speech output is performed. Only during speech output, supply current of approximately 3.5 mA is flowing. During the long quiescent time, the system current consumption is extremely low, and a battery life of several years can be expected.

Naturally, any other MSP430 MCU with hardware SPI interface can also generate PWM speech output as an additional function, in addition to the desired functions for the application.

4.2 eZ430-Speech Software

A Microsoft Excel® macro and two pieces of MSP430 MCU software to run the demo are supplied. The first software programs the SPI flash with audio data (MSP430F2012WriteWAV.c). Once the audio data is stored in the SPI flash, the software to play the audio data (MSP430F2012PlaybackWAV.c) is loaded into the MSP430 MCU, and the demo is ready to run.

4.2.1 Generating Audio Data

The simplest method to generate audio data is the Sound Recorder application (sndrec32.exe), which is installed with some versions of Microsoft Windows®. Any other software capable of recording or converting WAV sound files can be used instead.

NOTE: sndrec32.exe may not be available in newer versions of Microsoft Windows. If so, alternative methods of getting sound file data must be used.

Existing WAV files should be checked for correct audio format: PCM 8.000 kHz, 8-bit, mono. If the audio format is different, the file must be converted into the required audio format.

Another option is to record WAV files from a microphone or any other sound source. Again Sound Recorder is sufficient to do the recording, and the correct audio format (PCM 8.000 kHz, 8-bit, mono) still must be selected.

The conversion from the binary WAV file to a file with ASCII-represented hex values can be done with the included Excel macro ReadWavFile, which is stored in the file ReadWavFile.xls.

1. Open ReadWavFile.xls in Excel.
2. Press Alt+F8 to start the macro.
3. Double-click on ReadWavFile.
4. Select the WAV file to convert. Ensure that the audio is format is PCM 8 kHz, 8-bit, mono.
5. Select and copy the first column of the Excel worksheet to a text file. Go to cell A1 by typing Ctrl+Home and then press Ctrl+Shift+↓ to select the converted data.
6. Paste the data to a text file. Example text files with data generated by this macro can be found in the directory SPI-FlashData.

4.2.2 Loading Digital Audio Data Into SPI Flash

To load digital audio data into the SPI flash, the MSP430 MCU software MSP430F2012WriteWAV.c is used. This software converts between asynchronous RS232 UART and synchronous SPI. Using this software, a standard terminal emulation program on a PC, such as Microsoft HyperTerminal, can be used to download new audio data. The terminal emulation program must be configured in the following way:

- Baud rate: 38400
- Data bits: 8
- Parity: None
- Stop bits: 1
- Flow control: None

- Character delay: 10 milliseconds (Properties > Settings > ASCII Setup)

The UART communication between the host computer and the MSP430 MCU has been implemented with full-duplex software UART using both CCRs of Timer_A2 on the MSP430F2002.

Before trying to communicate with the SPI flash memory, the example code (SW-FullDuplexUART38400Baud-16MHzSMCLK.c) contains a full duplex 38400 baud UART echo software, which can test the UART communication between the host computer and the MSP430.

With this method of programming the SPI flash, the audio data files on the host computer are ASCII files and can be generated and modified with any ASCII editor. These ASCII files contain, in addition to the pure audio data, the necessary instructions for the SPI flash memory. All values are ASCII-represented hex values.

If no communication is ongoing, the select signal $\overline{S}$ of the SPI flash must be at high level. Before any instruction is sent to the SPI flash memory, $\overline{S}$ must switch to a low level. After the instruction is sent completely, $\overline{S}$ must go high again. The sequence of $\overline{S}$ and SPI instructions (SPI TX) is shown in Figure 7. This feature has been implemented with a very simple method: as long as the MSP430 MCU software reads from the host UART ASCII values that are non-hex values, the $\overline{S}$ signal is set to high and the received characters are ignored. As soon as a 2-digit ASCII hex value is detected, $\overline{S}$ is switched to low level, the 2-digit ASCII hex value is converted to one binary byte and sent through SPI to the SPI flash memory. $\overline{S}$ stays low as long as only valid 2-digit ASCII hex values are received by the UART. Any non-hex ASCII character received by the UART, such as a '.', switches the signal $\overline{S}$ to high level and ends the instruction.

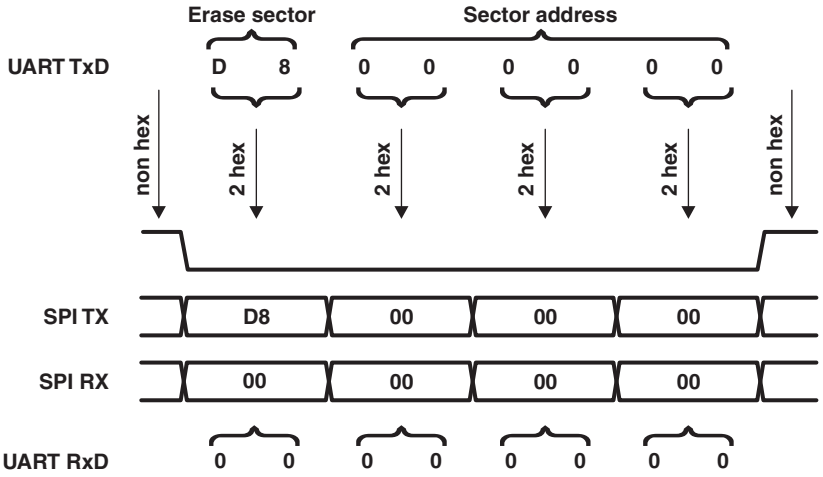

Figure 7. Conversion of UART ASCII Hex to SPI Binary Hex

The following are examples of SPI commands sent in ASCII through the USART for erasure and programming of the SPI flash:

- Erase Sector 0

.30	Clear status register fail flags
.0100	Write status register BP2=BP1=BP0=0
.06	Write enable
.D8000000	Sector erase and start with address 000000
.0500	Read status register

- Write Audio Data to Sector 0

.30	Clear status register fail flags
.06	Write enable
.0100	Write status register bp2 = bp1 = bp0 = 0
.06	Write enable

.020000007F1E0000004080C1014181... write data and start with address 000000
.06 Write enable
etc.

The instruction 'write data' is coded with the hex value 02, followed by a 3-byte memory start address 000000. All hex values after the instruction and the start address (after 02000000) are data values written to the flash. For the actual implementation of the playback demo software, it is necessary that audio data always starts at the beginning of a 64 KB sector in SPI flash. The first four hex values of the audio data represents the number of audio data bytes following. In the previous example, the number of bytes is 7F1E0000, which is represented by the hex value 0x00001E7F (low byte first). This method corresponds to the length double-word in the header of WAV files on personal computers.

To replace audio data currently in flash with new audio samples, the selected sector must be erased before the new data can be loaded. The baud rate of 38400 defines the transfer speed of the complete communication (USART and SPI). With this relatively slow speed, there is no problem with flash programming speed and, hence, no handshake is necessary. However, erasing a flash sector can take several seconds, and programming the flash should be delayed accordingly. Details about the timing for erasure and programming of a SPI flash can be found in the data sheet of the selected SPI flash memory.

The example code comes with some ASCII files to erase, write and read the SPI flash memory. In Microsoft HyperTerminal, the menu Transfer > Send Text file... can be used to send these files to the MSP430.

4.2.3 Audio Playback Software

After audio data has been stored in the SPI flash memory, the playback software (MSP430F2012PlaybackWAV.c) must be loaded into the MSP430 MCU, and the demo can be started. Figure 8 shows the software flow of audio playback.

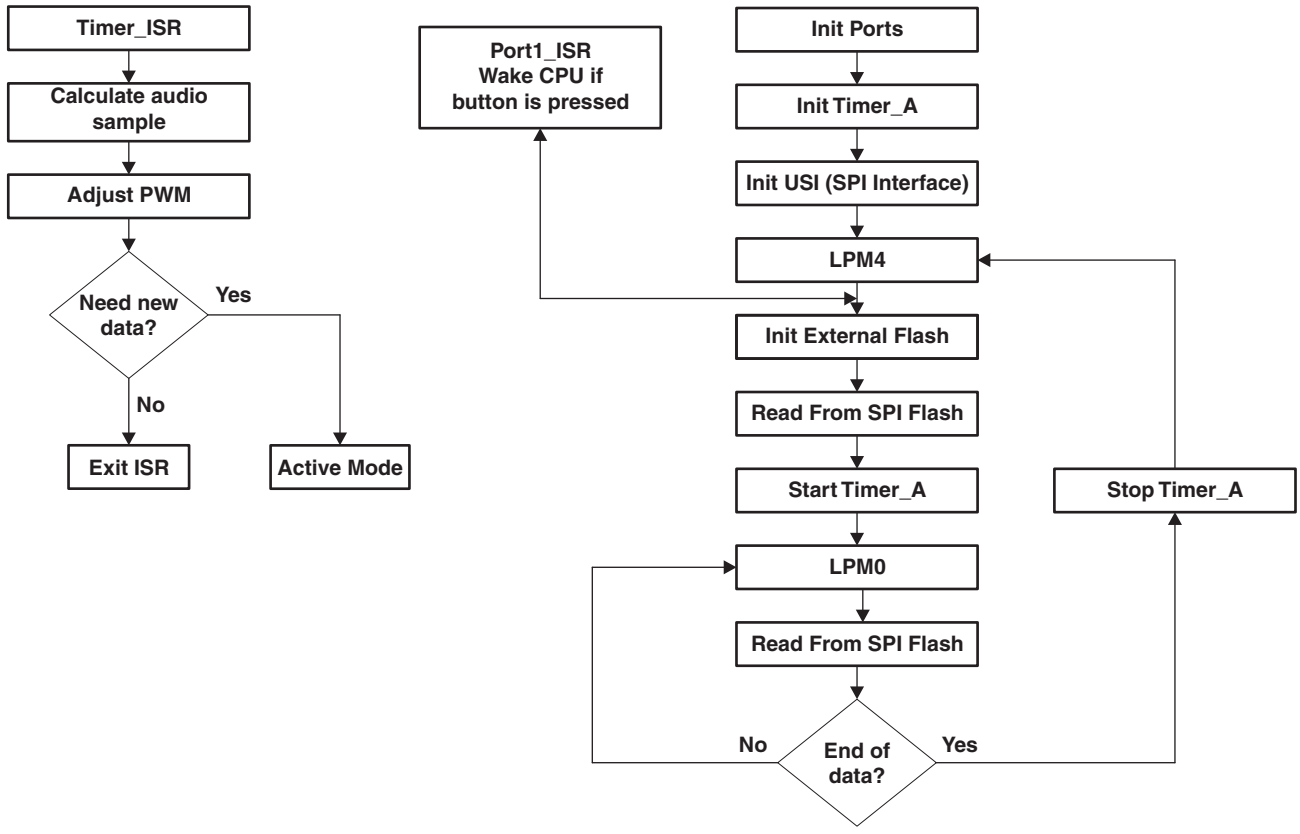

Figure 8. Software Flow of PWM Playback

After initialization of GPIOs, Timer_A, and the USI serial interface, the MSP430 MCU goes to LPM4 (low-power mode 4) and waits for a button pressed. As soon as the button has been pressed, the first two audio samples are read from the SPI flash, the first audio sample is written to Timer_A, and the MSP430 MCU goes to LPM0. In LPM0, the DCO continues to generate an 8-MHz clock signal, and Timer_A generates the PWM output signal. After each PWM period, a Timer_A interrupt is triggered. In the corresponding ISR, the next audio value is calculated and written to the Timer_A CCR to switch to a new PWM duty cycle. After four timer values, it is necessary to load the next audio sample from the external SPI flash. To start this task, the ISR exits with a modified Status Register on the stack, to keep the CPU in active mode after the RETI instruction, and in main() the CPU reads the next audio sample value from the external SPI flash. This continues until the last audio sample value has been read from the external SPI flash. Timer_A is then stopped, and the CPU goes to deep sleep mode LPM4 waiting for the button to be pressed.

PWM adjustments are done with an interrupt triggered at the falling edge of the PWM output signal. As shown in Figure 9, this method works without any problems for a decreasing duty cycles values.

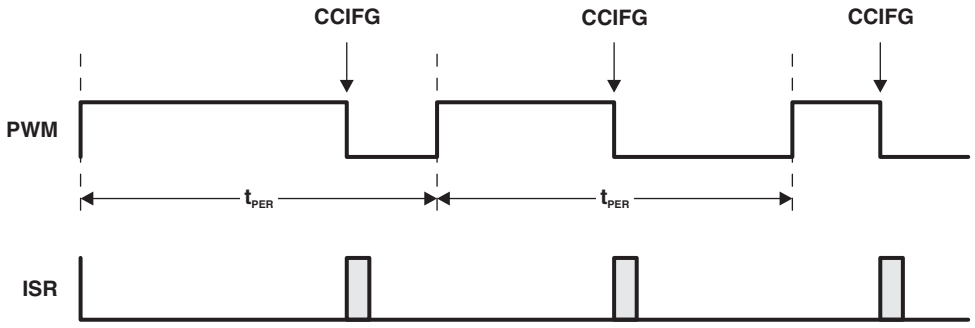

Figure 9. PWM Adjustment for Decreasing Duty Cycles

If the duty cycle value is increasing there are situations where this adjustment method might not work as expected. Figure 10 shows this scenario:

- The following audio data values is stored in the SPI flash: 20, 220.
- The software generates three additional linear averaged values between these two values and sends five values to the PWM: 20, 70, 120, 170, 220. These values represent the number of Timer_A clock cycles from the PWM rising edge until the PWM falling edge.
- The interrupt service routine (ISR) that is used needs 34 MCLK clock cycles from triggering the interrupt until the PWM is adjusted. Because MCLK runs at 16 MHz and the Timer_A clock (SMCLK) is 8 MHz, this 34 MCLK software delay time is equivalent to 17 Timer_A clock cycles.
- In this scenario, the PWM changes, for example, from 70 to 120, which is equivalent to 50 Timer_A clock cycles, but the ISR needs only 17 timer clock cycles to update the Timer_A PWM duty cycle. If this happens, the new programmed falling edge does not occur as desired in the next PWM cycle, instead it occurs within the same PWM cycle. On the oscilloscope it looks like three duty cycle values are lost.

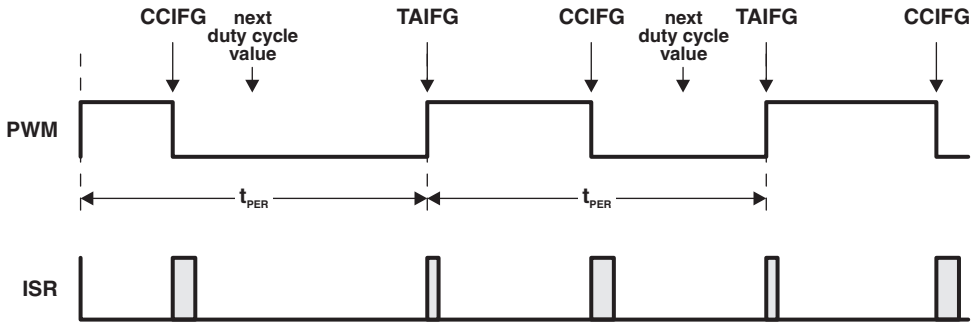

Figure 10. PWM Adjustment for Increasing Duty Cycles

To avoid this problematic scenario for increasing PWM cycles, the software compares two successive values in SPI flash with the value 68 and, if the next SPI flash value is more than 68 higher than the actual one, the interrupt for the falling PWM edge (CCIFG) is disabled while the interrupt for the rising PWM edge (TAIFG) is enabled. When the interrupt for the rising PWM edge (TAIFG) is triggered, this ISR disables the rising edge PWM interrupt and re-enables the falling edge PWM interrupt. This sequence ensures that only one falling edge PWM interrupt is triggered within one PWM cycle.

The compare value of 68 for two successive SPI flash values corresponds to 17 Timer_A clock cycles. The following is another example for this scenario:

- Successive SPI flash audio data values: 100, 168
- The linear averaging algorithm generates the values 100, 117, 134, 151, 168 for the five required PWM cycles. The difference between two successive PWM values is exactly 17, which corresponds to 34 MCLK cycles, the software execution time of the ISR until the Timer_A duty cycle change.

The playback demo software plays four different sound samples. Each time a user presses the button, one sound sample is played and, after four different sound samples, the sounds repeat from the beginning with the first sound sample. Each sound sample is in one 64-KB flash segment of the external SPI flash and, hence, each sound sample must be less than 64 KB. The limitation of 64 KB for each sound sample is a limitation of the demo software and not a limitation of the hardware. With different software, sound samples can be as large as the SPI flash memory.

The actual system uses 8-bit samples with a sampling rate of 8 kbps.

With the PWM running at a frequency of 32 kHz, sampling rates of 16 ksps and 32 ksps can easily be implemented. With different PWM frequencies, a huge variety of sampling rates is available.

If a higher resolution than 8 bits per sample is required, the limitation is the maximum system frequency. One bit more for the resolution doubles the required oscillator clock frequency. Hence, a 9-bit resolution needs a Timer_A clock frequency of 16 MHz.

The size of the demo audio playback software in C-language is:

- 786 bytes of code memory
- 68 bytes of data memory

Code and data size can be reduced with optimization and use of assembler coding.

4.2.4 Speech Compression

Simple compression algorithms are possible with MSP430 MCU, and the application report *Speech and Sound Compression and Decompression With MSP430 MCUs* describes an IMA adaptive differential pulse code modulation (ADPCM) compression and decompression algorithm and the steps to use the ADPCM library on the MSP430 MCU.

5 MSP-EXP430FG4618-Speech, A Demonstration With CPUX

5.1 MSP-EXP430FG4618-Speech Hardware

The MSP-EXP430FG4618 Experimenter's Board features an MSP430F2013 and an MSP430FG4618. Based on this board, PWM speech output is demonstrated with audio data stored in MSP430FG4618 internal flash memory. Pin TA2/P2.0 outputs the PWM audio signal, which is accessible on H4, pin 1. Surrounding pins on H4 are programmed to a logic low level and can be used as GND. To listen to the audio, use these connections to a powered speaker.

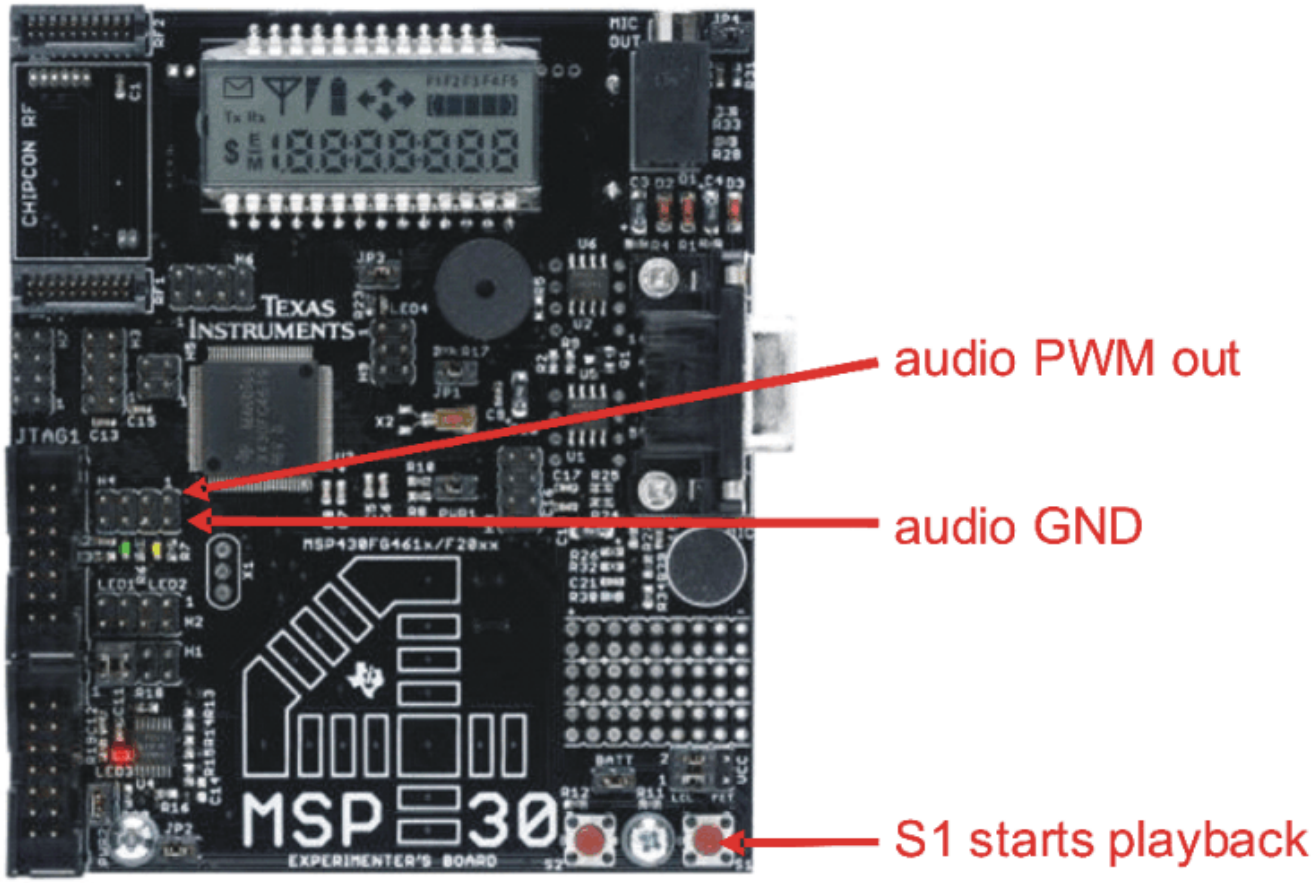

Figure 11. MSP-EXP430FG5618-Speech Hardware

5.2 MSP-EXP430FG4618-Speech Software

The eZ430-Speech software has been ported to the MSP430FG4618. There was no need to modify handling of Timer_A for PWM speech generation. CCR2 is used instead of CCR1, because the corresponding PWM output pin TA2 is accessible on a header, while TA1 is not accessible. The software section with the data handling through the external SPI flash memory has been replaced by a simple char array and pointer handling in extended memory above 64 KB. The filename is MSP430FG4618PlaybackWAV.c. This software is designed for IAR and CCE.

The software is available as source code as well as TI-TEXT and INTEL-HEX image files. Because the speech data does not fit into the free but memory-limited versions of the MSP430 software development tools, the source code can be tested with the debugger only if a full version of the software development tool is used. Without a full version, the PWM speech still can be tested by loading the TI-TEXT or INTEL-HEX image into flash with any programmer.

6 Conclusion

Audio playback with PWM is a simple way of adding new features to a product. With an external SPI flash, nearly every MSP430 MCU can play audio without the need for an external speech circuit. Speech can be added easily to existing applications, because the necessary code size and the hardware requirements are minimal. In combination with speech compression, such as ADPCM, the memory size requirement of the external SPI flash can also be reduced.

7 References

1. Speech and Sound Compression and Decompression With MSP430 MCUs
2. Using PWM Timer_B as a DAC

Schematics

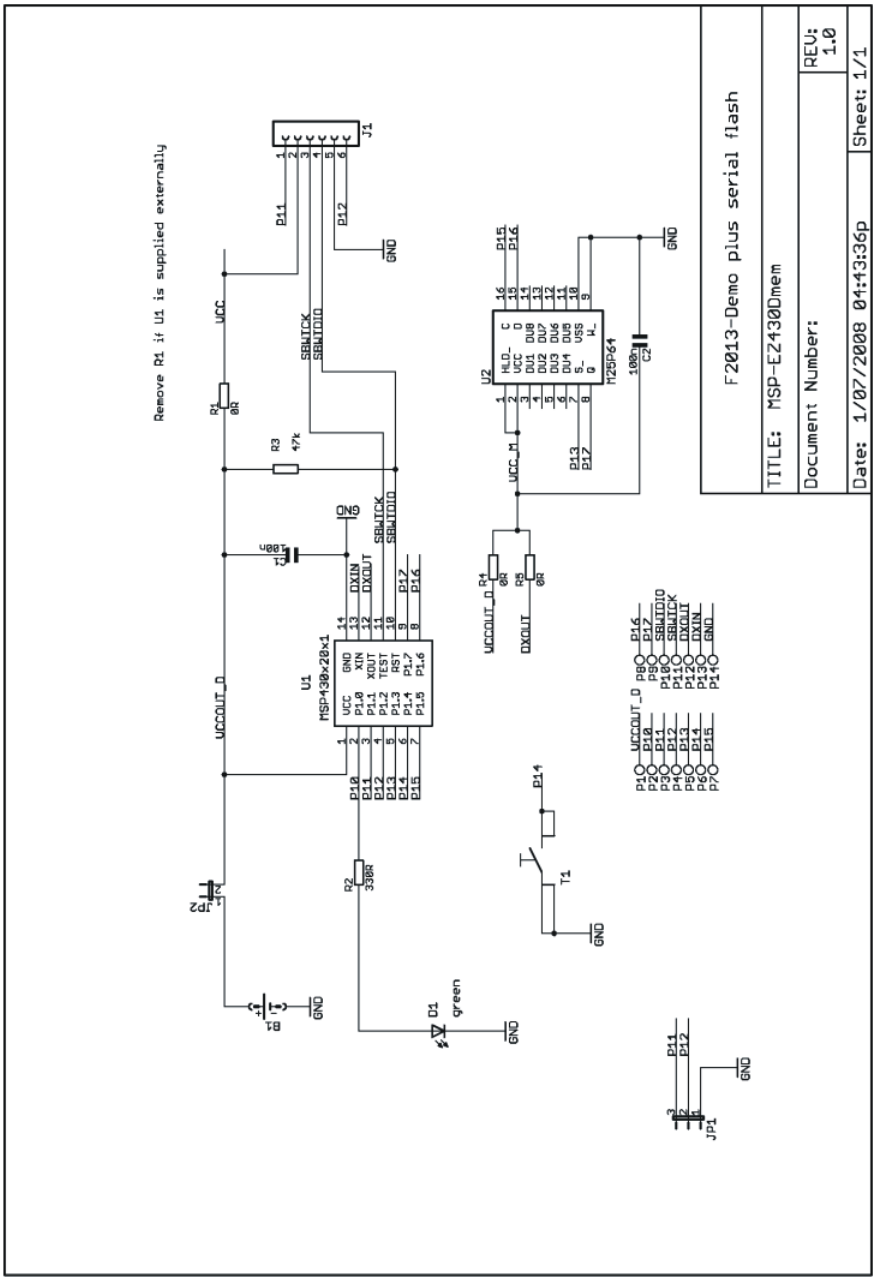

Figure 12. eZ430-Speech Schematic

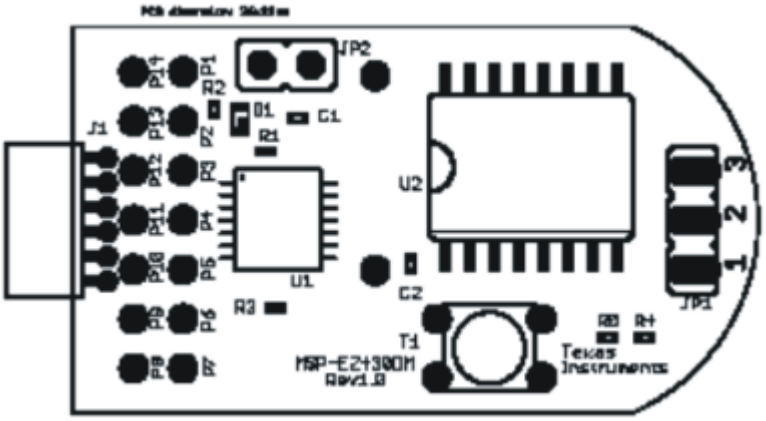

Figure 13. eZ430-Speech Board Layout

Revision History

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

Changes from September 15, 2009 to March 23, 2018 **Page**

- Added link to reference design in abstract 1
- Added link to reference design in Section 3.2, *Audio Playback With PWM* 4
- Added note about availability of Sound Recorder in Section 4.2.1, *Generating Audio Data* 7

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265