

Analog - Standard - DTMF Detector

Author: Victor Kremin

Associated Project: Yes

Associated Part Family: CY8C27443

Software Version: PSoC® Designer™ 5.4

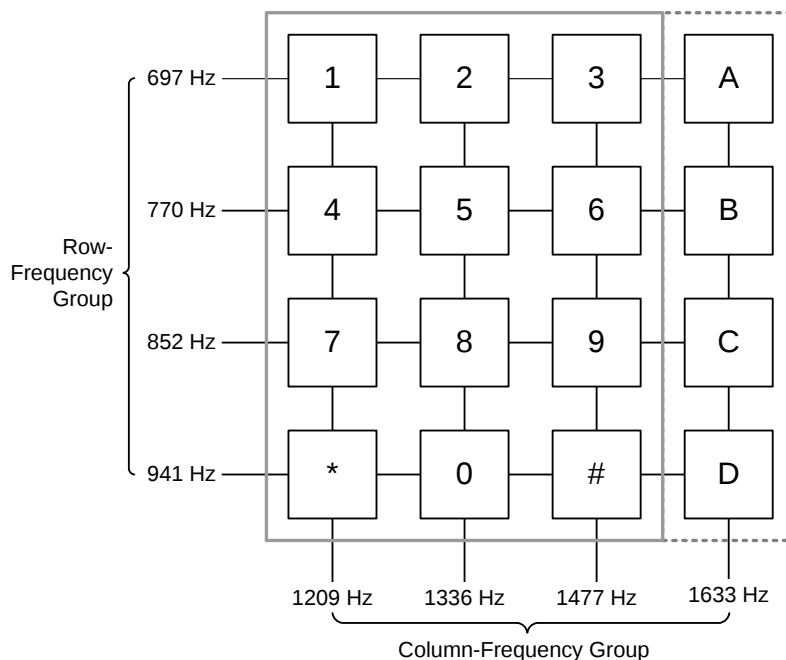
AN2122 describes the implementation of a dual-tone multiple frequency (DTMF) detector. This application note shows how to build an inexpensive decoder that satisfies most ITU requirements and consumes relatively low CPU resources to be operational in the background, while being able to distinguish speech and music from a valid DTMF signal.

Introduction

DTMF signaling is widely used in analog telephone dialing, data entry, voice mail systems, remote control of various consumer electronics (auto answering machines, home automation devices, bank information services, etc.).

The DTMF signal is the sum of two sinusoidal waveforms with predefined frequencies, which were selected according to the ITU Recommendations Q.23 [1] and Q.24 [2] ([References](#)). A user interacts with the DTMF dialer using a keypad with 16 characters. [Figure 1](#) illustrates the keypad layout and corresponding signal frequencies.

Figure 1. The Touch-Tone Phone Keypad



The conventional phones contain keys '0'-'9', '*', '#', and the keys 'A'-'D' can be found in the industrial controllers that are used especially for remote control. There are eight different frequencies, grouped into rows and columns. Each character is associated with pair frequencies from row and column groups.

The DTMF decoder determines the frequencies are present in the incoming signal and correctly decode the corresponding character. Table 1 illustrates the key decoder ITU specifications. The full DTMF signaling specifications is shown in [1].

Table 1. DTMF Decoder ITU Specifications

Parameter	Valid Limits of Parameter
Frequency Tolerance	The frequency tolerance about of $\pm 1.5\%$ is allowed for valid DTMF tone. The tone with offset $\pm 3.5\%$ must be rejected.
Signal Duration	A valid signal with duration of 40 ms must be considered as valid. The tones with duration 23 ms or less must be rejected.
Signal Interruption	A valid DTMF signal interrupted for 10 ms or less should not be detected as two distinct tones.
Signal Pause	A valid DTMF signal separated by 40 ms pause or more must be detected as two distinct characters.
Signal-to-Noise Ratio (SNR)	The DTMF detector must correctly process signal with SNR 15 dB.
Tones Twist	The detector must correctly decode DTMF codes when row frequency signal is 8 dB larger than column frequency signal. The detector must also operate correctly when column frequency signal is 4 dB larger than row signal.
Talk-Off	The detector should operate in the presence of speech and music without incorrectly identifying the speech signal as valid DTMF signals. To evaluate the decoder speech resistance, the decoder is tested using BellCore test tapes.

There are numerous design approaches to build DTMF decoders. The typical decoder consists of a front-end signal processing system and a back-end logic system to make conclusions concerning presence or absence of DTMF characters.

The most popular algorithms for front-end signal processing are based on Discrete Fourier Transform (DFT) [7] modifications. To reduce the number of calculations, the coefficients are calculated by using modified Goertzel algorithm [8], which requires only $N+2$ multiplications and $2N+2$ additions to process data for every N samples. The modified Goertzel algorithm calculates the squared absolute value of DFT coefficients and extracts row/column frequencies via second-order infinite impulse response (IIR) filter:

$$s_k[n] = 2 \cos\left(2\pi \frac{f_k}{f_s}\right) \cdot s_k[n-1] - s_k[n-2] + x[n] \quad \text{Equation 1}$$

$s_k[n]$ is the k^{th} filter response for $x[n]$ input sample, f_k and f_s are the k^{th} filter corner and sample frequencies, $s_k[-1] = s_k[-2] = 0$, and $n = 0..N$.

The squared absolute value of DFT coefficient $|Y(k)|^2$ can be calculated from the two last filter responses:

$$|Y(k)|^2 = s_k^2[N] + s_k^2[N-1] - 2 \cos\left(2\pi \frac{f_k}{f_s}\right) s_k^2[N] \cdot s_k^2[N-1] \quad \text{Equation 2}$$

In spite of the fact that this algorithm is widely recommended in application notes from various digital signal processing (DSP) chipmakers, the work [3] shows the DFT-based detection algorithms are not strictly ITU compliant because the passband for row frequencies does not satisfy the ITU requirements. The algorithm can be improved by using non-uniform DFT and dedicated back-end logic [4]. This implementation is computation intensive and DSP orientated. To meet ITU specifications, some DTMF detectors use DFT coefficient calculation for shifted row/column frequencies along with sophisticated decision logic.

To improve the speech resistance, DTMF decoders are often equipped with second-harmonic estimators. Speech and music are characterized by rich even harmonics; low levels of these harmonics characterize the DTMF signal. The speech estimator analyzes the second harmonics level and compares the results with fundamental frequency levels to detect speech/music, and predict false DTMF-signal detection.

The other effective decoder operation principles are based on resonant filtering using digital wave filters [5]. These filters are characterized by low sensitivity to rounding filter coefficients, but each sample requires several 16-bit multiplications and additions with output normalization to provide reliable filter operation. So, this decoder is better oriented for 16-bit CPUs or DSPs.

A very attractive decoder on a low-cost 8-bit processor is discussed in a recent paper [6]. The decoder uses pair-adaptive notch filters to suppress one tone in the incoming signal and a direct frequency estimator to measure the frequency of the other tone. This decoder meets all ITU specifications (both in time and frequency domains) and is characterized by very low speech resistance. The BellCore Talk-Off test results are at the upper limit of allowable bounds and are much worse than a good DFT-based decoder with second harmonics analysis. The decoder consumes about 4.5 MISP on an 8-bit CPU.

In the author's opinion, the optimal way to build a DTMF decoder lies in using adaptive resonant/band-pass filters to suppress unwanted noise signals together with direct or indirect filter-output frequency estimation. The filters (both for fundamental DTMF frequencies and second harmonics) output level analysis meets all ITU signal level specifications. These levels have reliable speech detection and direct frequency measurement provides satisfactory ITU frequency specifications. The decoder, which utilizes this principle, will be described in a separate Application Note.

DTMF Decoder Operation

During the development of this decoder, the goal is to build an inexpensive decoder that satisfies most ITU requirements and consumes relatively low CPU resources to be operational in the background. It must be stable to speech signals and incorporate second harmonics analysis to distinguish speech and music from a valid DTMF signal. The plan is to employ the modified Goertzel algorithm. The MATLAB simulations showed that 16*16-bit multiplication with output normalization is used to get stable IIR filter operation together with accurate filter corner frequency settings. Each Goertzel filter consumes about 400 CPU cycles, or 17 us at 24 MHz, for each sample process using the macros for multi-byte multiplication, PSoC® MAC [9]. To incorporate second-order harmonics analysis, 16 filter values must be calculated per sample and the sample rate must not be lower than 7.8 kHz to satisfy the Nyquist criterion. A particular solution can use a decimator for fundamental frequency filters or reject the second harmonics analysis but that is rejected. So, the other approach must be used.

The proposed decoder incorporates the quadrature correlator for fundamental DTMF frequencies and their second harmonics. The correlator correlates the incoming signal with two quadrature signals, which correspond to the sign function of sine and cosine signals of one fundamental DTMF frequency or its second harmonic. The advantage of this correlator is the absence of multiplication operations; the correlator performs addition when the reference signal is positive and subtraction when the reference signal is negative.

After processing the N samples, the quadrature components are squared to eliminate incoming signal-phase dependence and are ready to follow processing by back-end logic.

$$vs_k[n] = vs_k[n-1] + \text{sign} \left[\sin \left(2\pi \frac{f_k}{f_s} n \right) \right] \cdot x[n]$$

Equation 3

$$vc_k[n] = vc_k[n-1] + \text{sign} \left[\cos \left(2\pi \frac{f_k}{f_s} n \right) \right] \cdot x[n]$$

$$Z_k^2 = vs_k^2[N] + vc_k^2[N]$$

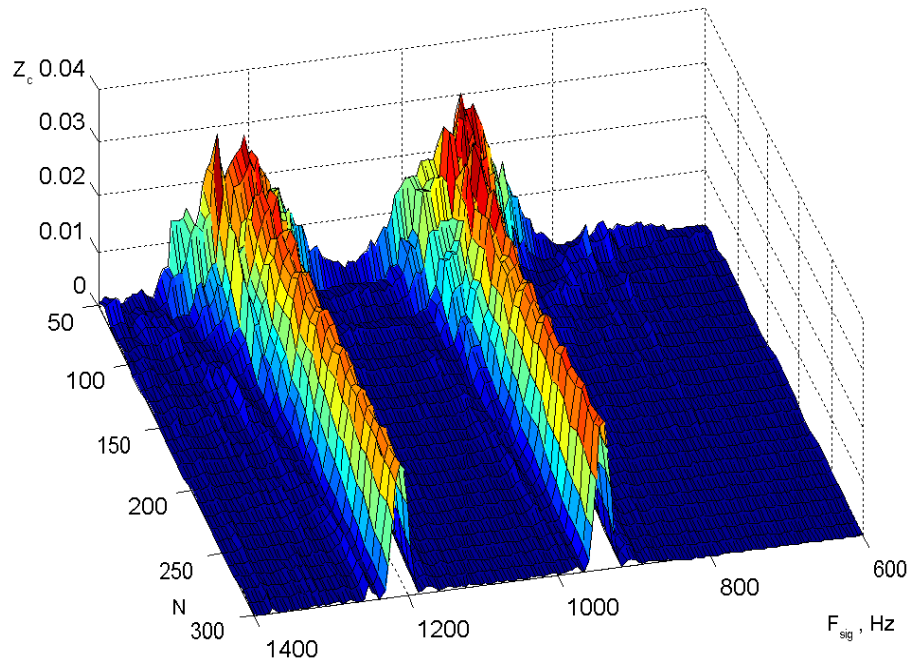
Equation 4

$vs_k[n]$ and $vc_k[n]$ are sine and cosine correlator components for k^{th} frequency, f_k is the DTMF or second harmonic k^{th} frequency, and f_s is the sampling frequency. $vs_k[0] = vc_k[0] = 0$, $n = 1..N$, and $k = 0..15$. Z_k^2 is the squared correlator output.

A simulation model is constructed using MATLAB.

Figure 2 shows the simulated correlator response Z_k^2 for the DTMF signal, which is also a function of the correlation frequency f_k and sample length N . The correlator model uses 16-bit, fixed-point arithmetic and the output response is divided by $2^{16} \cdot N^2$ for normalization and scaling. The DTMF signal consists of the signal's sum with frequencies at 941 Hz and 1209 Hz that corresponds to '*' character; the sample rate is 8518 Hz. The sample rate is larger than the rate commonly used in telephone systems, 8000 Hz, but is utilized to simplify the anti-aliasing filter.

Figure 2. Correlator Response Simulation



The simulation results show that the correlator satisfies most ITU frequency specifications for column frequency group of $N=127$, which corresponds to the accumulation time of 15 ms and satisfies the minimum DTMF signal-level detection demands.

For row frequencies, the allowable boundary is larger. Because the correlator uses output result division on 2^{16} for output data normalization, the low limit of the input signal must be evaluated as well. Figure 3 shows the simulated normalized correlator transfer characteristics as a function of signal amplitude and N .

Figure 3. Correlator Transfer Characteristics

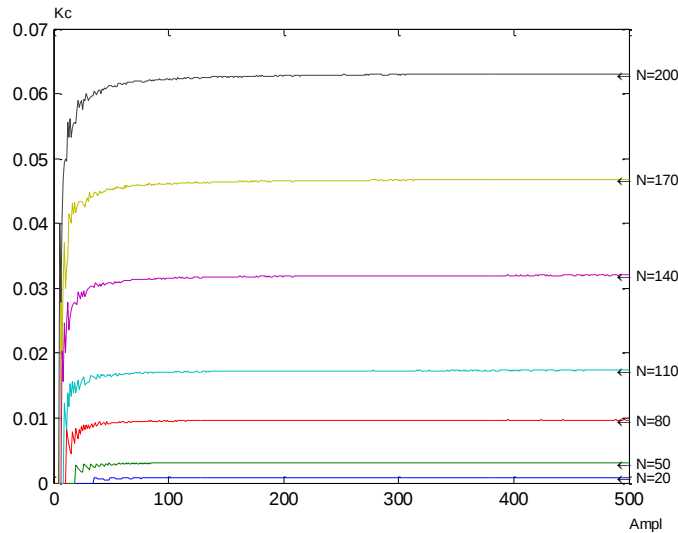


Figure 3 shows that the transfer characteristic is virtually linear when the DTMF signal code's amplitude is greater than 15. If an 8-bit ADC is used, the input signal should be multiplied with a gain of 4 to increase the dynamic range of the input signal. Under this condition, the estimated correlator dynamic range for the input signal will be more than 25 dB, which meets ITU specifications.

Decoder Implementation

Signal Processing Software Front-End

The DTMF decoder is implemented largely in software using both interrupt and main techniques. The signal processing front-end subsystem is located in an 8-bit sigma-delta ADC Interrupt Service Routine (ISR) and the back-end logic is placed in the main program loop.

The ADC ISR performs the following tasks:

- Correlation sum updates for fundamental DTMF frequencies and their second harmonics;
- Incoming signal energy calculation using PSoC MAC;
- Correlation sum squaring and normalization using PSoC MAC;
- Input amplifier gain adjustment;
- Synchronization with back-end decision logic.

Figure 4. ADC ISR Structure

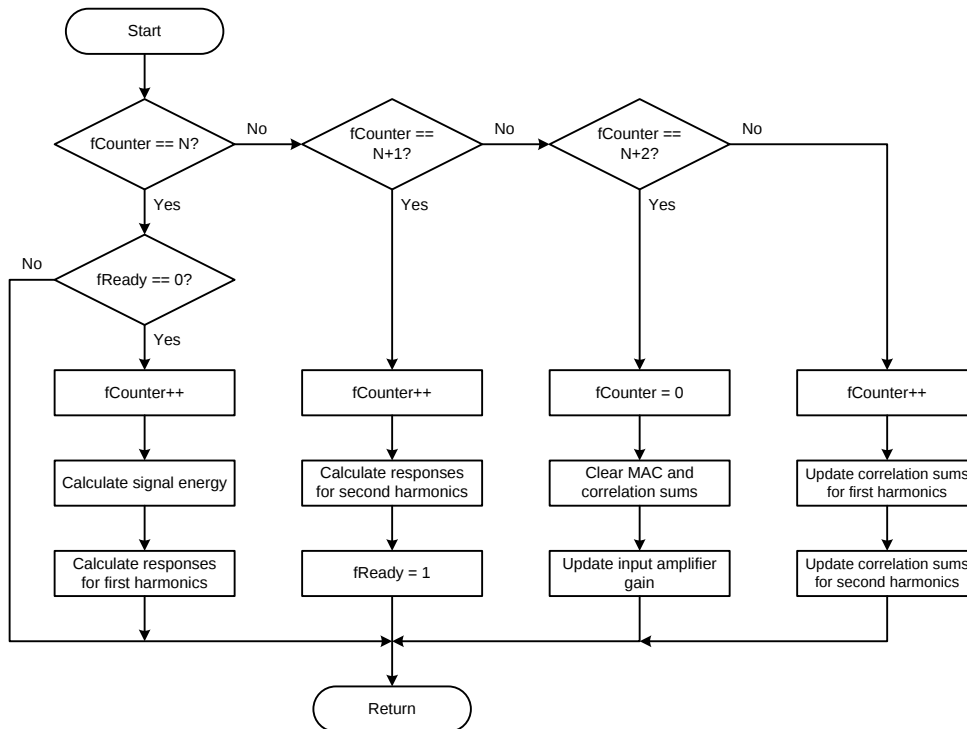


Figure 4 illustrates the ADC ISR structure. A simple state machine is implemented in the ADC ISR to process these tasks. When this routine is called, the *fCounter* variable is checked first. When the correlator has processed the *N* (where *N* is sample length) input samples, the *fCounter* is equal to *N*. Then the *fReady* flag is checked. This flag marks the previous calculation results which were read by the back-end logic and can now be overwritten.

If this flag is cleared, the *fCounter* is incremented and the correlation sums for fundamental DTMF frequencies are squared and normalized. Note that the correlation sums squaring and normalization code is placed in the ISR to save 32 bytes of RAM. Moreover, because the ISR uses the PSoC MAC for incoming signal calculation, there is sense to use this MAC unit here to avoid resource-sharing problems. In addition, the correlator output calculation is distributed into several sequential interrupts to balance the interrupt latency.

The following interrupt performs the correlation results calculation for second harmonics and sets the *fReady* flag. Now the correlator data is completely ready and can be read by the back-end logic subsystem. This subsystem must read the correlation results and clear the *fReady* flag more often than the correlator accumulation period, or 15 ms in our case, to avoid correlator stalls. The next ADC interrupt initializes the correlation sums, MAC, clears *fCounter* variable, and updates the input PGA gain value if the automatic gain-control loop is used after the correlator is ready to process the next *N* ADC samples.

Back-End Logic

The back-end decision logic is more complicated and is implemented as a subroutine, which is periodically called by the main program loop. The back-end logic routine implements the following tasks:

- Performs various validity checks;
- Parses the incoming data stream into separate DTMF characters;
- Calls the automatic gain-loop routine, if needed.

Figure 5. Decision Logic Algorithm

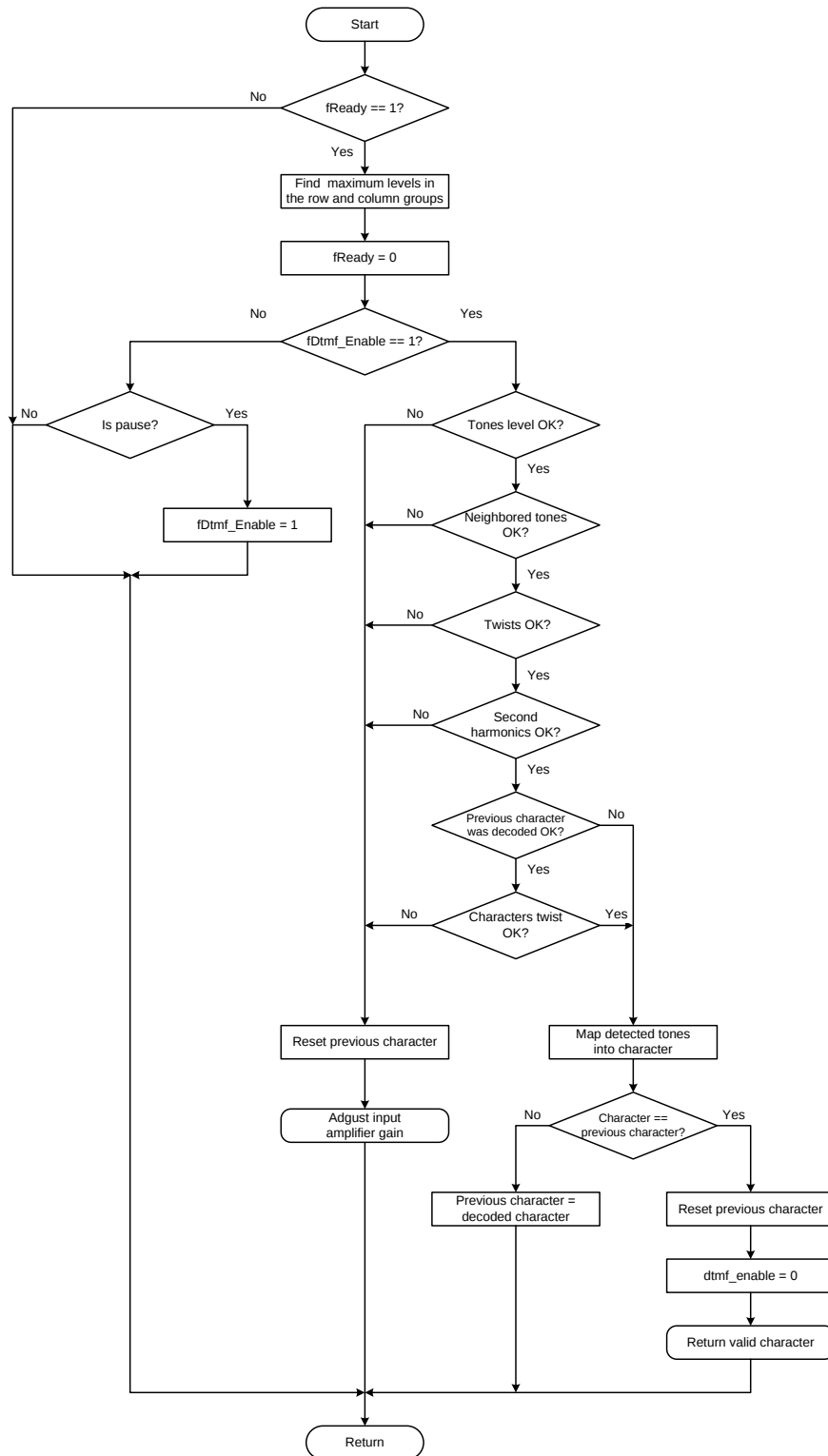


Figure 5 illustrates the decision logic operation. The fReady flag is checked. When the correlator data is ready, the tones with maximum level from row and column frequency groups are found. The fReady flag is then cleared to allow calculation of the next correlator data and avoid possible correlator stalls. The back-end logic subsystem can be set to two modes: awaiting valid DTMF tone signal and pending signal pause. The fDtmf_Enable flag is used to switch between these modes. The fDtmf_Enable flag is set to enable awaiting valid DTMF tone mode.

To detect a valid DTMF tone, a series of checks are performed. The first check is signal strength. The sum of squared amplitudes of the peak tones from row and column frequency groups must be greater than a predetermined threshold. Because there are possible twists (differences in the level between high and low tones) between row and column peak tones, the sum estimation works better than separately comparing the row and column tone levels.

The second check is adjacent tones strength analysis. The strongest row/column tone must stand out from the adjacent tones within its group by more than a pre-set threshold ratio. To avoid overflow, the peak tone levels are scaled and compared with the adjacent tones.

The third check is tone twist estimation. There are two possible twists types in the incoming DTMF signal. Assuming the telephone channel is a low-pass filter, the high frequency tone is usually received with lower amplitude than low frequency tone, which is called “normal twist.” To overcome this twist, the DTMF generators amplify the high frequency tone level. If the telephone channel gain-frequency characteristic is near to linear, the high-frequency column tone can be larger than the low frequency row tone. This phenomenon is called “reverse twist.” The decoder must accept DTMF signals with 8 dB row/column amplitude relations for “normal twist” and 4 dB for “reverse twist.” The decoder checks the relationship between peak row/column tones to detect both “normal” and “reverse” twists.

The fourth check is the second harmonics strength check. It is assumed that the DTMF signal contains only fundamental tones of its odd harmonics (if telephone channel amplifiers were saturated under incorrect signal levels). Speech and music have significant levels of even harmonics added to fundamental frequencies. This check makes sure that the ratios between peak row/column tones and their second harmonics are larger than a predefined threshold.

For reliable DTMF decoder operation, each character must be decoded twice, consecutively. If the previous character was correctly decoded, the summarized peak tone's level between two sequentially decoded characters will not differ substantially. This check also prevents false detection of short DTMF signals.

When all tone checks have been completed, the peak tones are mapped in the corresponding character. If the previously detected character is the same as the current character detected, the previous character is cleared, the decoder is switched from the “pause” to “await” mode by clearing the fDtmf_Enable flag, and a valid character is returned. This switch is required to prevent the misdetection of one durable DTMF signal as several separate identical characters.

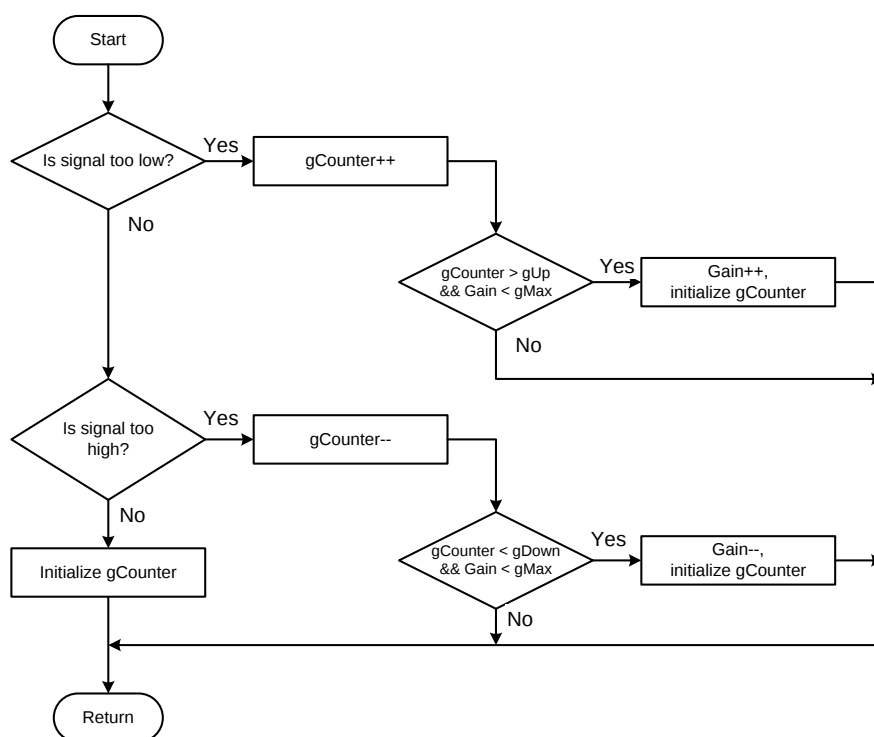
If any of the checks fail, the previously detected character is cleared and a routine to calculate the input amplifier gain is called. If a valid DTMF detection sequence is in progress, the gain adjusting system is completely blocked. Note that the gain is updated in the ADC ISR to synchronize the gain update with the new measurement cycle and eliminate unwanted correlator data fluctuations.

In “pause” mode (fDtmf_Enable is equal to 0); the strength of peak row/column tones is checked. The sum of the squared row and column peak tones detects the pause. If this sum is smaller than the predefined threshold value, the fDtmf_Enable flag is set and the decoder exits pause mode.

Automatic Gain-Control Subsystem

The automatic gain control (AGC) subsystem expands the decoder input signal's dynamic range. The AGC adjusts the input PGA gain according to the estimated signal energy. Figure 6 illustrates the AGC operation.

Figure 6. Automatic Gain Control Algorithm



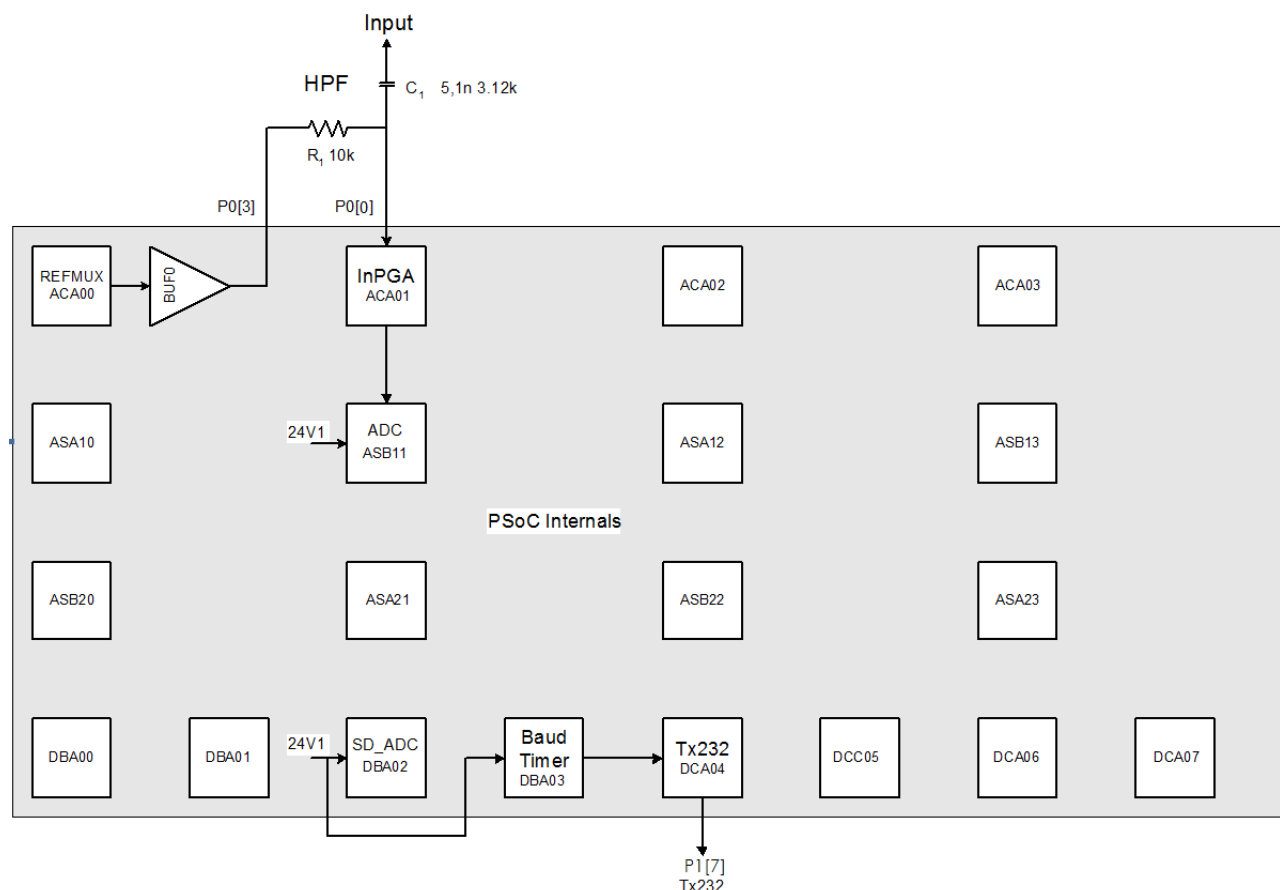
AGC operates as follows; when the energy signal is too low, the variable gCounter is incremented. If this variable reaches the upper predefined limit, the gain is increased and gCounter is reinitialized. If the incoming energy signal is too high, gCounter is decremented. When this variable reaches the lower limit, the PGA gain is decreased. If the incoming energy signal falls into the allowed range, the gain counter variable gCounter is reinitialized to an immediate value. Upper and lower PGA gain level checks are implemented as well.

Note that the automatic gain subsystem has asymmetric transient characteristics; the gain rising time is much shorter than the gain falling time. This allows for quick recovery from over-range situations, but preserves the existing gain level during speech/signal pauses. These AGC transient characteristics were obtained by proper selection of gain counter gCounter upper, lower limits and intermediate initialization values. The AGC can be turned off at the time of compilation by setting AGCUSE condition variables in dtmfcalc.asm and globdefs.h to 0 if AGC is not used.

Decoder Hardware

Figure 7 illustrates the internal PSoC connections and several external components that are needed to make the DTMF detector operational.

Figure 7. PSoC Internals



The high-pass filter R_1C_1 cuts the low-frequency signals as the AC mains. The input programmable gain amplifier scales the incoming signal. The amplifier output is connected with input from the 8-bit sigma-delta ADC. The baud-rate timer forms the clock signal for the serial transmitter, which can be connected via level shifter (MAX3221E, for example) to a PC to analyze the decoded characters. The default data rate is 115200 bauds.

The optional component is crystal oscillator, that clock the processor. The external oscillator (for example, DS32KHz from Maxim, and SG-3030JF from Epson, etc.), connected with the P1[1] port via small decoupling capacitor is recommended. Although, the 32.768 kHz crystal, connected accordingly to P1[0], is also used.

If optional watch crystal is used, the high-frequency PSoC generator output must be analyzed to verify that the phase-locked loop (PLL) is working properly and no visible jitter is present on the generator signal when the serial transmitter is active. To do this, the baud-rate timer output can be passed to an external pin and observed via an oscilloscope. The possible jitter can degrade the ADC signal-to-noise ratio and reduce overall performance of the DTMF. The external clock oscillator is strongly recommended for debugging purposes. This is because the emulation board has dense traces and other digital signals can break/degrade the PLL and oscillation stability. For end products, the watch crystal is completely sufficient.

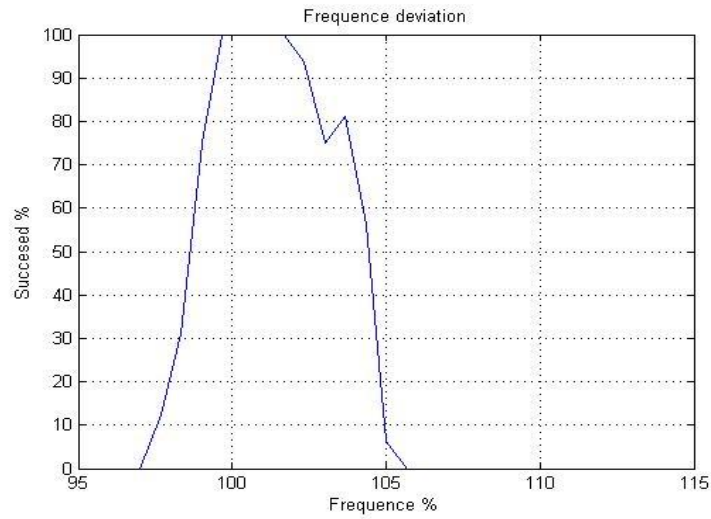
Decoder Test Results

Table 2 illustrates test results for the proposed decoder. The decoder is tested using generated waveforms from sound editor Cool Edit Pro 2.0; you can download the full-featured evaluation copy at [11]. Note that the text marked in *red italics* illustrates the properties that are outside of ITU specifications.

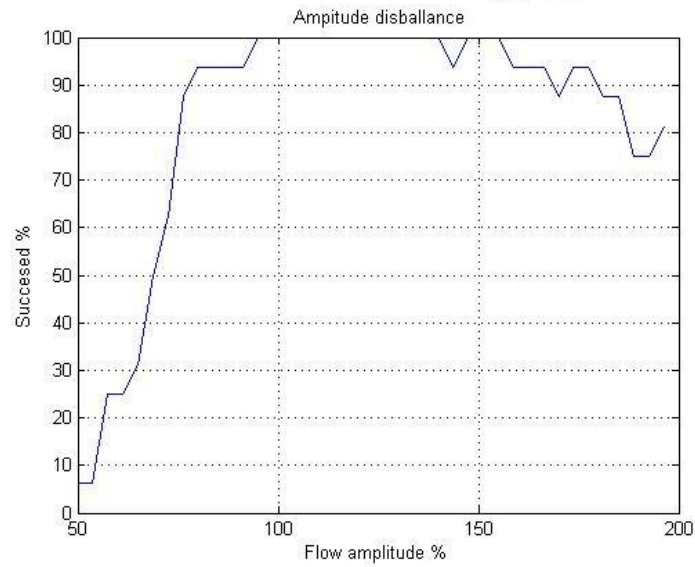
Table 2. Decoder Test Results

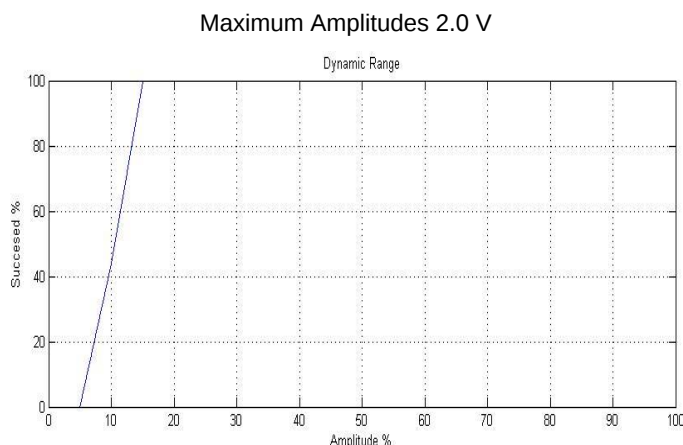
Test Item	Parameter	Comment
Frequency Tests		
Frequency	Allowed Deviation	The larger frequency tolerance for lower limit of 697 Hz, upper limit of 941 Hz, and lower limit for 1209 Hz is caused by the absence of proximity tone level checks for these frequencies. If you require a decoder strict in ITU-compliance, you can add two intermediate frequency bins. For example, take 660 Hz and 1075 Hz and add additional checks in the back-end logic routine.
697 Hz	-5.5%... +3.1%	
770 Hz	-2.8%...+3.4%	
852 Hz	-3.4%...+3.5%	
941 Hz	-3.7%...+4.5%	
1209 Hz	-4.2%...+2.8%	
1336 Hz	-3.2%...+2.7%	
1447 Hz	-2.8%...+2.6%	
1633 Hz	-2.6%...+2.5%	
Twist Tests		
Normal Twist Limit	9.1 dB	The tests were performed for digits 1, 5, 9, D.
Reverse Twist Limit	4.7 dB	
Time Tests		
Minimum Tone Time	45 ms	The test is performed for digits 1, 2, 3, 4.
Min Pause Time	13 ms	
Dynamic Range Tests		
Dynamic Range (AGC Off)	17 dB	The dynamic range of 25 dB can be obtained by lowering the DTMF signal-strength check threshold.
Dynamic Range (AGC On)	38 dB	The AGC must have time to stabilize, so if it is used, a continuous tone must precede the information tones.
Talk-Off Resistance Tests		
False Character Detection Rate	3-7 responses per hour when speech/music signal was present.	The talk-off test gets complete using the BellCore reference tape.

Frequency Amplitudes 1.5 V



High Frequency Amplitudes 1.5 V





Possible Decoder Modifications

The proposed application provides the basic DTMF decoder, which satisfies most designs. Many variations can be suggested. The decoder can be equipped with an anti-aliasing LPF for some applications. This LPF can be built using the PSoC switched-capacitor filters or adjustable Sallen and Key low-pass filters, [AN2031](#).

If the user cannot dynamically allocate the sigma-delta ADC for detection, it can be done using PSoC dynamic re-configuration. A 6-bit SAR ADC or single-bit comparator can replace the ADC. The comparator-based DTMF detector has some performance issues, but is still suitable for many remote-control applications.

The correlation memory consumes $4*N$ bytes of Flash memory. This is not suitable for some applications. In this case, the modified Goertzel algorithm can be implemented instead of a quadrature correlator and the second harmonics analysis can be omitted. Secondly, the user adjusts the sample length and sample frequency according to application demands. The command-line utility, *dtmf.exe* (included in the downloadable files), automatically generates the assembler include file, *corrtables.inc*, for given N and sample frequency.

The total time of execution for the ADC ISR during the incoming signal is near 70 us with a sample period of 117 us and PSoC clock frequency at 24 MHz. If the second harmonic analysis is not required, the ISR can be simplified by omitting the 16-quadrature sum calculation (saving 32 bytes of RAM). The CPU clock can be decreased to 12 or 6 MHz. In this case, the sample rate can be reduced as well, and an anti-aliasing filter is suppresses possible speech/music signals.

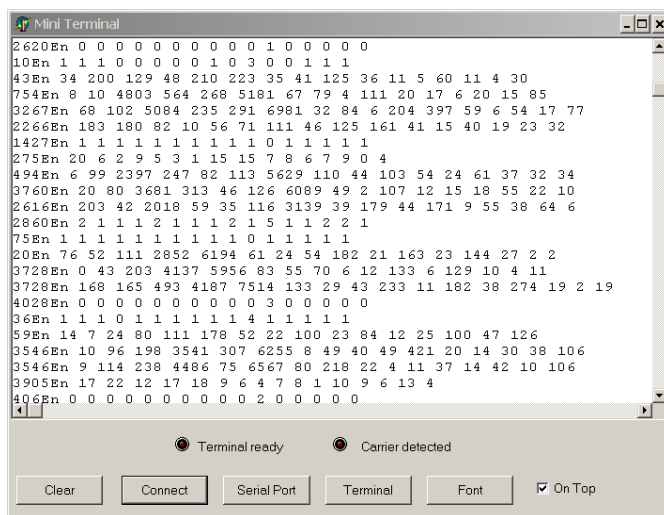
If high accuracy in the DTMF tone frequency analysis is required, the sample length can be increased by discarding the rule that each character must be consecutively detected twice.

Additional frequency bins with intermediate frequencies can be estimated and compared. To improve timing specifications and guarantee strong ITU compliance, the detector can be equipped with an incoming signal synchronizer to synchronize the correlation cycles with the DTMF start signal.

For debugging and testing purposes, a special debug mode is implemented when the detector sends, via serial port, the debug information concerning incoming signal level, correlator data, and both fundamental DTMF frequencies and their second harmonics. You can use any terminal program to see this stream. The number at the far left is the scaled energy signal and the other sixteen numbers are correlator output data. The number at the far left corresponds to 697 Hz. The number at the far right corresponds to 3266 Hz.

Figure 8 illustrates the terminal window during debug information transmission when the phone is dialed. Debug mode can be entered during compilation by setting the *DEBUG* variable to 1 in the *globdefs.h* file. In normal mode (*DEBUG* is set to 0), the decoder sends only the properly decoded characters, followed with new line symbols. Note that users should disconnect the ICE when the project is being debugged to decrease the possible influence of noise introduced by the ICE on analog part performance.

Figure 8. Terminal Window in Debug Mode



References

1. ITU Blue Book, Recommendation Q.24, Multi-Frequency Push-Button Signal Reception, Geneva, 1989.
 2. ITU Blue Book, Recommendation Q.23. Technical Features of Push-Button Telephone Sets, Geneva, 1989.
 3. G. Arslan, B. Evans, F.A Sakarya and J. Pino, "Performance evaluation and real-time implementation of subspace, adaptive and DFT algorithms for multi-tone detection", in Proc. IEEE Int. Conf. Telecommunications, (Istanbul, Turkey), pp.884-887, Apr. 1996. The paper can be downloaded in the electronic form at http://ptolemy.eecs.berkeley.edu/papers/96/dtmf_ict/ic196.pdf.
 4. M. Felder, J. Mason, B. Evals, "Efficient dual-tone multi-frequency detection using the non-uniform discrete Fourier transform," IEEE Signal Processing Letters, vol. 5, pp.160-163, July 1998.
 5. Kaiser Ulrich, Digital Processor Architecture with Reduced Instruction Set for Wave Digital Filters, Dissertation, Bochum, 1991.
 6. Amey A. Deosthali, Shawn R. McCaslin, Brian L. Evans, A Low-Complexity ITU-Compliant Dual Tone Multiple Frequency Detector, IEEE Trans. On Signal Processing, vol. 48, no. 3, pp. 911-916, Mar. 2000, the article can be download in the electronic form at <http://www.ece.utexas.edu/~bevans/papers/2000/dtmf/dtmf.pdf>.
 7. V. Oppenheim and R. W. Schaffer, Discrete-Time Signal Processing. Prentice-Hall, 1990.
 8. J. Proakis, D.G. Manolakis, Digital Signal Processing Principles, Algorithms, and Applications. NJ, Prentice Halls, 1995.
 9. Dave Van Ess, Signed Multi-Byte Multiplication, Application Note [AN2038](#), Cypress Microsystems.
 10. Jeff Dahlin, Using the PSoC Microcontroller External Crystal Oscillator, Application Note [AN2027](#), Cypress Microsystems.
- The sound editor Cool Edit Pro 2.0 can be downloaded freely at <http://www.syntrillium.com/download/>.

About the Author

Name: Victor Kremin

Title: Associate Professor

Background: Victor earned a radiophysics diploma in 1996 from Ivan Franko National Lviv University, PhD degree in Computer Aided Design Systems in 2000 and is presently working as an Associate Professor at National University "Lvivska Polytechnika" (Ukraine). His interests involve the full cycle of embedded systems design including various processors, operating systems, and target applications.

Contact: vick@cypress.com

Document History

Document Title: Analog – Standard – DTMF Detector – AN2122

Document Number: 001-38003

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1520284	YIS	10/01/2007	New Application Note
*A	2936425	VICK	05/24/2010	Updated Cross References
*B	3282455	ANBI_UKR	06/14/2011	Updated to latest PSoC Designer. Updated to new template.
*C	3315215	YARD_UKR	07/15/2011	Updated associated project.
*D	4384903	GRAA	05/20/2014	Updated to new template.
*E	4622158	ASRI	01/13/2015	Updated Abstract. Updated Software Version as “PSoC® Designer™ 5.4” in page 1. Updated attached associated project to PSoC Designer 5.4.
*F	4669719	ASRI	02/24/2015	Fixed typo in footer in page 1.
*G	5711698	AESATP12	04/26/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at [Cypress Locations](#).

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

PSoC® Solutions

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6](#)

Cypress Developer Community

[Forums](#) | [WICED IOT Forums](#) | [Projects](#) | [Videos](#) | [Blogs](#) | [Training](#) | [Components](#)

Technical Support

cypress.com/support



Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2007-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spanion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spanion, the Spanion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.