MICROCHIP

AN257

DTMF Detection Using PIC18 Microcontrollers

Author: *Gaurang Kavaiya*
Microchip Technology Inc.

INTRODUCTION

This application note describes a new method for decoding Dual Tone Multifrequency (DTMF) signals using the PIC18 family of PICmicro® microcontrollers. DTMF signals, popularly known as "Touch Tone", are used for a wide variety of applications in modern telephony. DTMF provides a fast and reliable method for dialing.

This application note explains the concept of decoding DTMF signals using a PICmicro microcontroller (MCU) by providing a reusable library module for DTMF detection, which can be used as a building block for a variety of Telecom applications.

The objectives of this application note are:

- Meet all or most of the DTMF requirements
- Use minimal resources
- Create a simple and flexible system

DTMF OVERVIEW

Initially, the Pulse method was used for telephone dialing. This method works by actually disconnecting or hanging up the telephone line at specific intervals. Because dialing information cannot travel across the telephone line, Pulse dialing is quite slow and dialing information is limited between the switching office and the telephone equipment. To overcome these limitations, another dialing method using push button dialing was developed. This method is known as DTMF, or more popularly, "Touch Tone".

DTMF uses audio frequencies to transfer dialing information. The DTMF signal consists of two simultaneous sinusoidal signals. The audio frequencies are divided into two groups: Low Group and High Group. The DTMF signal is created by taking one frequency signal from each group and combining them.

DTMF tone frequencies have been chosen so that harmonics are avoided (frequencies do not fall in another DTMF frequency range), no frequency is the multiple of another, and in no case, does the sum or difference of two frequencies result in another DTMF frequency. Table 1 shows DTMF frequencies for each digit.

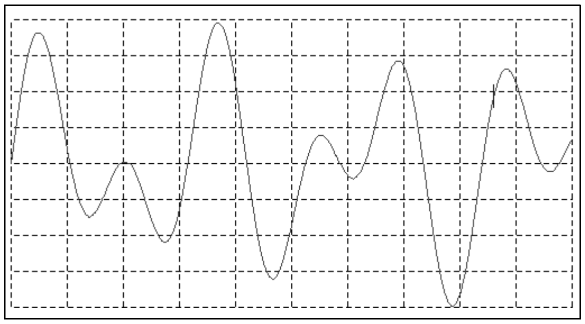
TABLE 1: DTMF FREQUENCIES

		High Group Frequencies Nominal Frequency in Hz			
		1209	1336	1477	1633
Low Group Frequencies Nominal Frequency in Hz	697	1	2	3	A
	770	4	5	6	B
	852	7	8	9	C
	941	*	0	#	D

Note: The shaded column (A, B, C, D) is not normally found on a telephone.

The actual DTMF signal may look something like the waveform shown in Figure 1 and can vary greatly depending on the frequency, phase and amplitude of each component frequency.

FIGURE 1: DTMF WAVEFORM



DTMF SPECIFICATIONS

The International Telecom Union (ITU) has defined some specifications and requirements for the DTMF detector. If a detector meets all ITU requirements, it is considered an excellent detector that can work with a variety of real world conditions. The following are some of the major requirements for a DTMF detector.

Frequency Requirements

The DTMF frequency requirements are specified as:

- Maximum accepted frequency offset: 3.5%
- Minimum rejected frequency offset: 1.5%

If one or both of the DTMF frequencies are outside the ±3.5% range of the nominal frequency, then the detector should reject it. If both of the DTMF frequencies are within the ±1.5% range of the nominal frequency, then the detector should accept them.

Power Level Requirements

The DTMF detector should accept the signal if per frequency power is ≥ -25 dBm and ≤ 9 dBm. It should reject the DTMF signal if any frequency power is ≤ 55 dBm.

Twist Requirements

The DTMF Twist requirements are specified as:

- Forward: 8 dB
- Reverse: 4 dB

The difference in the power of the two DTMF frequencies is called Twist. The DTMF receiver should accept signals when the power of the High Group frequencies is between $(L + 4)$ dBm and $(L - 8)$ dBm, where L is the power of the frequency from the Low Group frequencies.

Timing Requirements

The DTMF signal timing requirements are specified as:

- Minimum accepted tone length: 23 mS
- Maximum rejected tone length: 40 mS
- Minimum pause time: 40 mS

The DTMF detector should accept DTMF signals when the signal duration is ≥ 40 mS. It should reject the DTMF signals when the signal duration is ≤ 23 mS. The DTMF detector may either accept or reject the DTMF signals when the signal duration is between 23 mS and 40 mS. A DTMF receiver should accept the DTMF signals with inter-digit intervals ≥ 40 mS.

Signal-to-Noise Ratio Requirements

The DTMF detector should work with a Signal-to-Noise Ratio (SNR) as low as 23 dB; however, there are other parameters that can determine the performance of the DTMF detector.

Guard-Time Test

A Guard-Time test is defined as the minimal length, or the shortest tone bursts, that can be reliably detected by the detector.

Talk-Off Test

It is possible that a voice signal can occur with DTMF signals. The ability of the detector to reject normal voice signals and accept the DTMF signal is tested in the Talk-Off test.

THEORY

There are a variety of methods available to decode DTMF signals. The most basic method involves the use of eight narrow band-pass filters tuned at each DTMF frequency.

Because of the complex nature of DTMF signals, decoding them in software involves converting them into a frequency domain for analysis. This process requires considerable computational power to convert a signal into a frequency domain. The most common method uses Discrete Fourier Transform (DFT) for analysis.

When frequency spectrum analysis is required only at certain frequencies, DFT provides a better solution compared to other software algorithms. At minimal performance, the DFT method requires considerable computational power. Therefore, DTMF decoding is considered a Digital Signal Processor (DSP) domain application. Typically, 8-bit microcontrollers do not have considerable computing capability, so implementing this kind of algorithm on an 8-bit microcontroller will use all of the available computational resources.

One of the main objectives of this application note is to find an alternative for achieving frequency spectrum conversion without sacrificing too many resources. Before discussing alternate methods, it is important to understand the behavior of traditional methods.

Fourier Transform Theory

Jean Baptiste Fourier showed that any signal or waveform could be made up by adding together a series of pure tones (sine waves) with appropriate amplitude and phase. Fourier's theorem assumes that sine waves of infinite duration are added.

The frequency spectrum has two axes: frequency and energy. The energy at a particular frequency is represented by an energy bar at that point. The Fourier Transform (FT) provides details on amplitude, phase and frequency of pure tones required to create a particular signal. Because each pure tone represents one frequency in spectrum and its amplitude provides the energy details, the Fourier Transform method (see Equation 1) provides the spectrum of the signal.

EQUATION 1: FOURIER TRANSFORM

$$X(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} x(t)e^{-j\omega t}\,dt$$

Where Time Domain definitions are:

- $x(t)$ is complex, continuous and non-periodic
- t runs from $-\infty$ to $+\infty$

The Frequency Domain definitions are:

- $X(\omega)$ is complex, continuous and non-periodic
- ω runs from $-\infty$ to $+\infty$

The Fourier Transform is a mathematical formula using integrals, while the Discrete Fourier Transform (DFT) shown in Equation 2, is a discrete numerical equivalent using sums instead of integrals.

EQUATION 2: DISCRETE FOURIER TRANSFORM

$$X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-j2\pi k\frac{n}{N}}$$

Where Time Domain definitions are:

- $x[n]$ is complex, discrete and periodic
- n runs over one period from 0 to $N-1$

The Frequency Domain definitions are:

- $X[k]$ is complex, discrete and periodic
- k runs over one period from 0 to $N-1$

To better understand how DFT works, the formula can be simplified using Euler's formula:

$W = e^{-j2\pi k n/N}$ with $\cos(2\pi k\, n/N) + j\sin(2\pi k\, n/N)$

where W is the kernel function. The resulting formula is shown in Equation 3.

EQUATION 3: SIMPLIFIED DFT

$$X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]*\left(\cos\left(2\pi k\,{}^{n}\!/_{N}\right) + j\sin\left(2\pi k\,{}^{n}\!/_{N}\right)\right)$$

With the formula simplified, it is apparent that if the N number of periodic samples is multiplied by sine waves and cosine waves, a single complex number is returned as the result. The magnitude of this number will provide the energy at analysis frequency. Because DFT is discrete in nature, it is used to analyze the energy at analysis frequency. Therefore, to calculate DFT at one frequency, multiply the signal with the sine and cosine waves at the analysis frequency. The magnitude of the result provides the energy at this frequency.

It is obvious from the formula that a tremendous amount of computational power is required, and because of that reason, it is typically considered a DSP domain application. Attempting to implement this logic on an 8-bit microcontroller may end up using all the resources of the 8-bit microcontroller for basic functionality. As previously mentioned, one of the goals of this application note is to find a better alternative to calculate DFT, which will require very little computational power. Therefore, let's review the fundamentals of DFT to find an alternate solution.

DFT Fundamentals

DFT uses one property of sinusoidal called orthogonality. If sinusoidals are orthogonal, which means if two sinusoidals of the same frequency are multiplied, this can result in high amplitude. If they are of different frequencies, this results in a lower or zero amplitude. Now the question is, can the same functionality be obtained with a square wave? Square waves are digital signals, while sinusoidals are analog signals. Because sinusoidals are analog, an A/D Converter is needed to convert them to a digital signal, which based on the resolution of the A/D Converter, can result in a very large computation. Square waves are a 1-bit value so the multiplication result will be a smaller number.

If we again look at the DFT formula in Equation 3, it is important to understand the effect of using a square wave in practical applications. Finding a DFT, or the energy at a particular frequency, requires multiplying the signal using the sine and cosine reference signal of analysis frequency. The signal will be sampled for $n = 0$ to $n = N - 1$ and the instantaneous value of the signal and kernel are multiplied to find the instantaneous result. The summation (Σ) of each result provides the total signal energy. In this instance, we need to convert the incoming signal into a square wave, which is equivalent to a 1-bit A/D conversion. Therefore, we will use a square wave reference signal and sample.

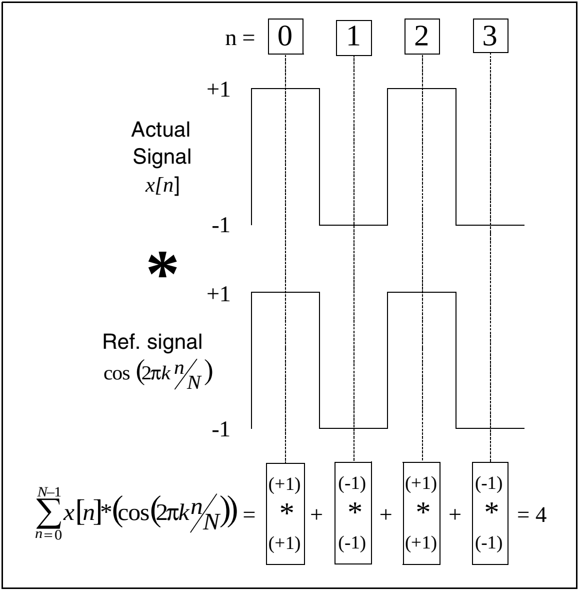
Let's analyze the DFT formula with a square wave. For simplicity, only the real part will be considered.

According to the formula shown in Equation 4, we need to multiply N points of the signal with the cosine reference signal of the analysis frequency to calculate the real part of DFT.

EQUATION 4: DFT REAL PART CALCULATION FORMULA

$$\operatorname{Re} X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n] * \left(\cos\left(2\pi k \, {}^{n}\!/_{N}\right)\right)$$

FIGURE 2: DFT USING SQUARE WAVE



n = 0 1 2 3
+1
Actual Signal x[n]
-1
*
+1
Ref. signal cos (2πk n/N)
-1
Σ x[n]*(cos(2πk n/N)) = (+1)*(+1) + (-1)*(-1) + (+1)*(+1) + (-1)*(-1) = 4

Assuming a case of $N = 4$, Figure 2 shows the DFT calculation process for a square wave.

As seen in Figure 2, the DFT calculation using a square wave with a 1-bit resolution is much simpler than using sine waves.

If both square waves have a positive value, the multiplication result will be positive. If one of the signals is negative, the result will be negative, and if both are negative, then the result will be positive. To summarize, if both square waves have the same value, then the result will be a larger number; otherwise, it will be a smaller number. However, multiplication can be replaced by a simpler digital function for a square wave. For simplicity of digital analysis, a square wave can be represented by '0' and '1' instead of '1' and '-1'. The multiplication can be achieved by a simpler exclusive nor function.

Simplified DFT Implementation

The DFT process shown in Figure 2 can be modified to replace one bit signed multiplication to simplify the implementation, while Figure 3 best explains the method of finding a DFT.

FIGURE 3: CASE 1: ACTUAL SIGNAL MATCHES THE REFERENCE SIGNAL

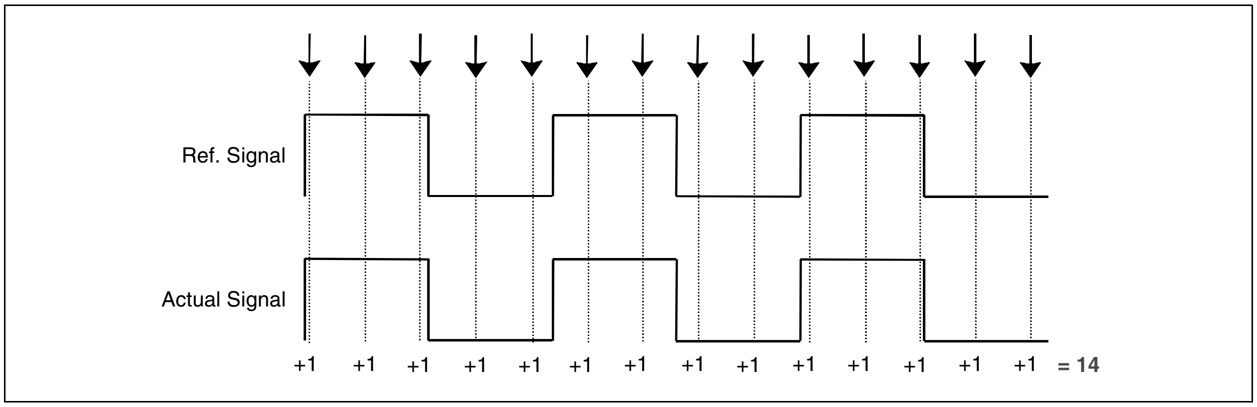
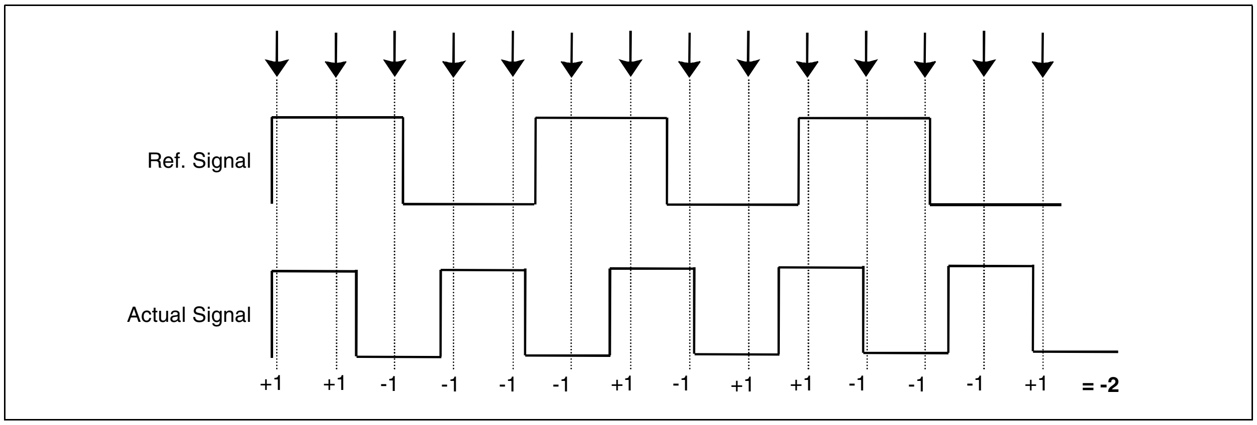


FIGURE 4: CASE 2: ACTUAL SIGNAL DOES NOT MATCH THE REFERENCE SIGNAL



The actual signal (after 1-bit A/D conversion) is sampled at a fixed sampling frequency. At the sampling point, the actual signal is compared with the reference signal of the analysis frequency. If both of the square waves match, the energy match counter is incremented. If there is a mismatch, then the same counter is decremented. This gives the effect of multiplying the square wave signal with two states of ‘1’ and ‘-1’. Implementing this on an 8-bit microcontroller is very simple. The timer resource can provide the interrupt at the sampling frequency.

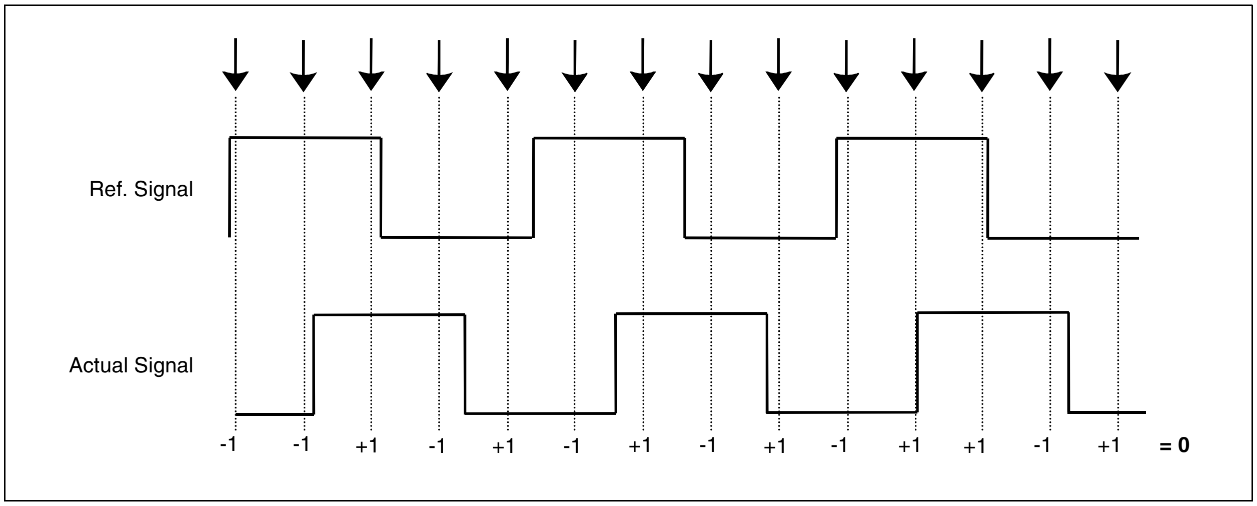
In the interrupt handler, we need to compare the bit value of the reference signal and the actual signal. Depending on the status of the two signals, we need to either increment or decrement the counter. Based on these requirements, it is very easy to implement DTMF using a 1-bit DFT on an 8-bit PICmicro microcontroller without using excessive resources.

Orthogonality of Simplified DFT

Figure 3 shows the first test case for DFT, where the actual signal matches the reference signal. The basic requirement for orthogonality is that we should get a higher magnitude when the actual signal matches the reference signal and we should get a lower amplitude for a mismatch. Figure 4 shows the second test case for mismatch. This proves that square waves are orthogonal. If you look at the Fourier theorem, it states that a square wave can be constructed using a sine wave, so this result is to be expected.

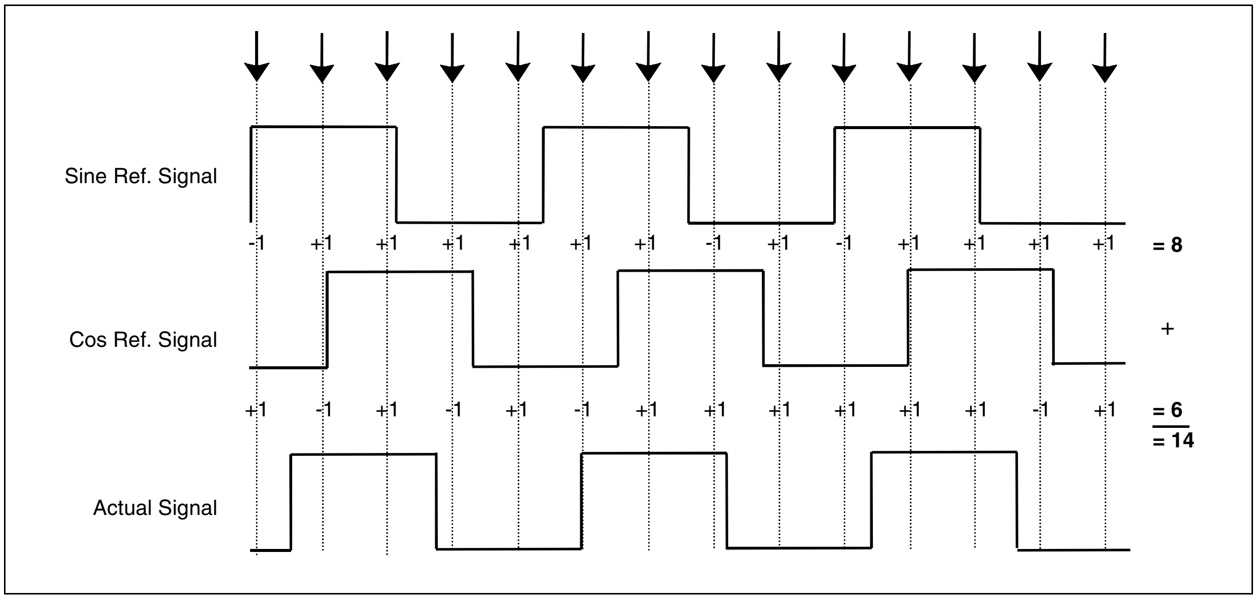
It would seem that the system described so far satisfies the requirements for DFT; however, there is still one major drawback. If the actual signal is out-of-phase compared to the reference signal, then the existing system will provide lower amplitude for some phase mismatch conditions, as illustrated in the third test case (Figure 5). Because the phase of the incoming signal can not be guaranteed, the DFT system must be insensitive to phase change.

FIGURE 5: CASE 3: ACTUAL SIGNAL IS OUT-OF-PHASE TO REFERENCE SIGNAL



Again, the solution for this problem lies in the DFT formula. The DFT formula uses both sine and cosine waves for multiplication. The signal is multiplied with both sine and cosine signals and the resultant sum value is used for analysis. Therefore, the fourth test case shown in Figure 6, analyzes the out-of-phase actual signal with the sine and cosine references.

FIGURE 6: CASE 4: USING SINE AND COSINE REFERENCE



In this case, the actual signal is sampled at a fixed sampling frequency. At the sampling point, the actual signal is compared with both of the reference signals of the analysis frequency. If the reference signals and sample frequency match, the particular energy match counter is incremented; if there is a mismatch, the same counter is decremented. At the end of the sampling period, unsigned addition of the two reference match counter values is performed to find the final match counter value. This computation provides the energy at analysis frequency, which gives the effect of multiplying the actual signal with the sine and cosine reference signals (where each signal has two states of ‘1’ and ‘-1’). It is apparent from Figure 6 that this method is insensitive to phase error.

Simplified DFT Limitations?

We now have an alternative to DFT that does not require too much computation power; however, this method also suffers from some limitations similar to normal computed DFT, as well as a few additional limitations.

Like a normal DFT, the sampling frequency is important for this method. The best result can be obtained only with a sampling frequency that is based on a specific input frequency. As in the case of a DFT, where all component frequencies are predefined, it is possible to find an optimized sampling frequency that will give minimum error for all component frequencies.

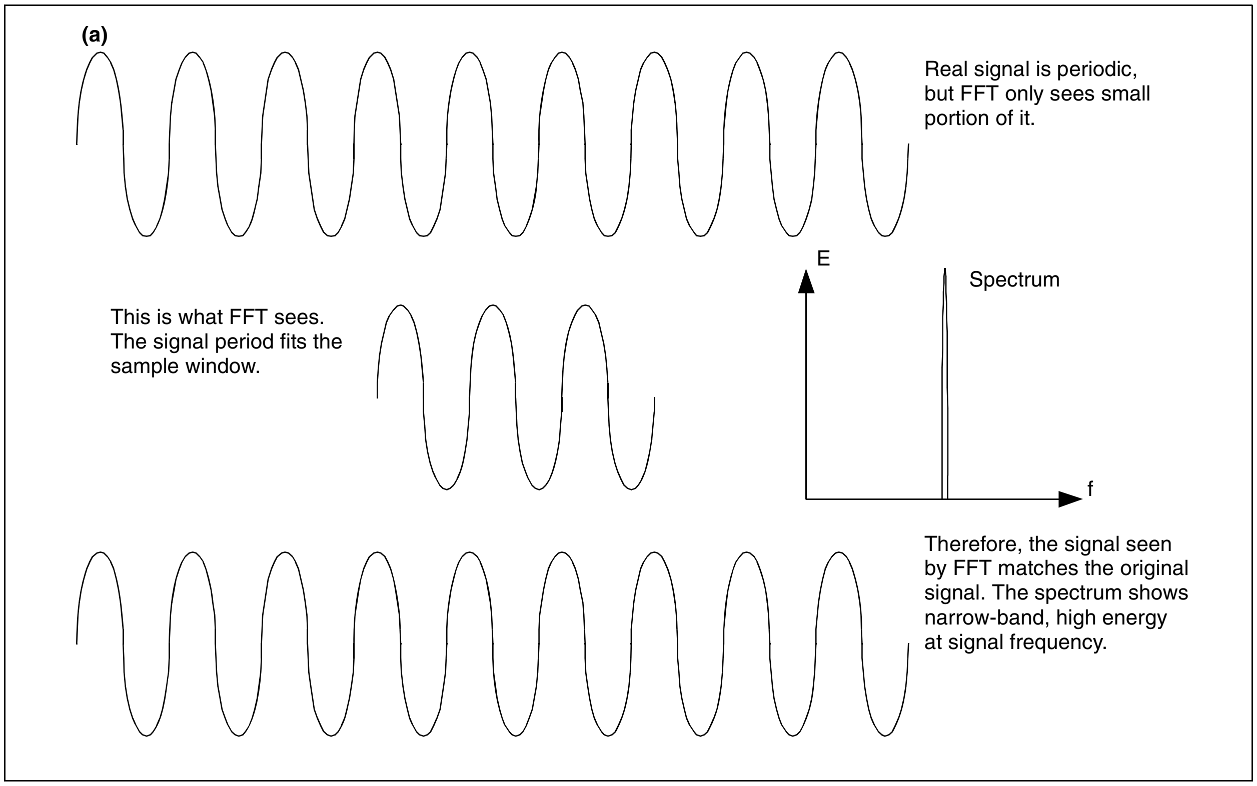
An FT assumes that the signal is periodic and of infinite duration. In a practical case, we do not sample the signal for an infinite duration. The signal is sampled for a finite duration called a "window period". The FT assumes that this signal is periodic all of the time. This can create a problem in certain cases.

Upcoming sections use the following terminology to explain DFT:

- Sampling Frequency = Fs = How often the waveform is sampled
- Sampling Period = 1/Fs
- Sample Window = Window Period = Time duration for the waveform sampling

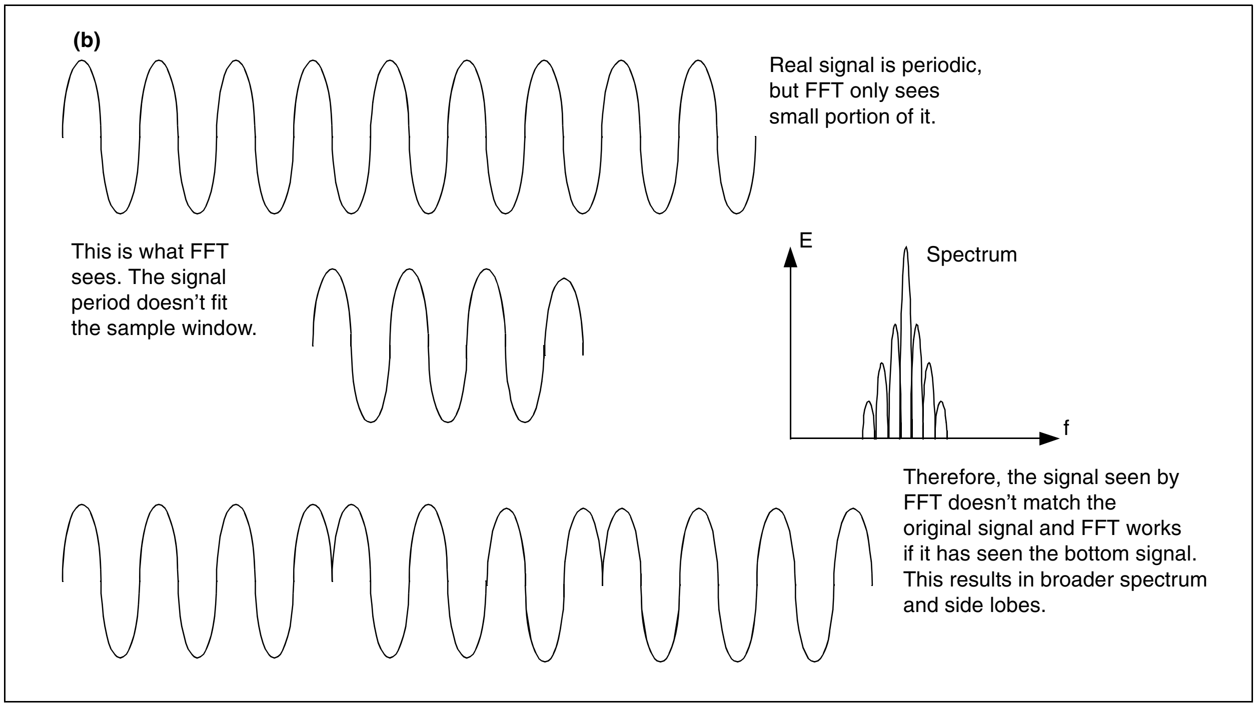
If a window period is an integer multiple of a signal period, then the integer number of cycles fits in the window period and the signal seen by FT matches the original signal. This test condition is illustrated in Figure 7.

FIGURE 7: CASE 1: SIGNAL PERIOD IS INTEGER MULTIPLE OF SAMPLE PERIOD


(a)
Real signal is periodic, but FFT only sees small portion of it.
This is what FFT sees. The signal period fits the sample window.
E
Spectrum
f
Therefore, the signal seen by FFT matches the original signal. The spectrum shows narrow-band, high energy at signal frequency.

If the window period is not an integer multiple of the signal period, then the integer number of cycles will not fit in the window period. In this case, FT assumes that the signal is repeating at the end of the window period, which creates discontinuity in the signal seen by FT and it will not match the original signal. This test condition is illustrated in Figure 8.

FIGURE 8: CASE 2: SIGNAL PERIOD IS NOT AN INTEGER MULTIPLE OF SAMPLE PERIOD


(b)
Real signal is periodic, but FFT only sees small portion of it.
This is what FFT sees. The signal period doesn't fit the sample window.
E
Spectrum
f
Therefore, the signal seen by FFT doesn't match the original signal and FFT works if it has seen the bottom signal. This results in broader spectrum and side lobes.

This creates a problem in spectrum analysis. There is a direct relationship between the duration of signal and its frequency spectrum. Short signals have broad frequency spectra and long signals have narrow frequency spectra. The glitch in signal creates a broader spectrum. These 'glitches' are short signals; therefore, they have a broad frequency spectrum. Moreover, this broadening is superimposed on the frequency spectrum of the actual signal, which creates a secondary energy peak, or side lobes in frequency spectrum.

The solution for this problem is to ensure the smooth matching of the beginning and end section of the window period. The simplest way to do this is to make them zero to end in smooth matching. DSP uses a method called "windowing", where the input signal is multiplied by the window function, which ensures smooth matching.

Use of the window function involves complex computation. Since one of the main goals of this application note is to reduce the resources, and we have discussed an alternative for compute intensive DFT, it makes sense to find an alternative to the window function.

In the case of DTMF, frequency values are already known, so it is possible to find an optimum sampling period that gives minimum error with most of the frequencies. Another advantage occurs from 1-bit DFT. Because the resolution of the incoming signal is 1 bit, the maximum possible truncation is only 1 bit. The 1-bit truncation results in fewer errors than higher truncation in the case of sinusoidals. Therefore, we have a cheap solution for a 1-bit DFT-based system. Now we need to find the optimized window period, which will give minimum error for 1-bit DFT-based analysis of DTMF frequencies. The DTMF detection simulator PC software provides a tool to find the optimized frequency and window period.

Now that we have a solution for DFT and windowing, the question of how good 1-bit DFT really is arises. Can this solution meet the frequency band specification for DTMF? Unfortunately, the answer is no. Because 1-bit DFT provides a wideband response, it is not possible to meet this requirement without some modification in the algorithm.

Let's analyze the method to meet the frequency band requirement by creating some test cases to understand the effect of the frequency change on match counters (energy value).

All test cases use the following DTMF decoder system:

- Sampling frequency: 8 kHz
- Sampling period: 16 mS
- SNR: 40 dB

TEST CASE 1:

Test frequencies are the same as DTMF nominal frequencies.

Test conditions:

```
LowBand= 770.0     HighBand= 1209.0
PhaseL = 0.0       PhaseH  = 0.0
AmpL   = 100.0     AmpH    = 100.0
```

Match (Energy) counter values:

```
697.0 = 26         1209.0 = 66
770.0 = 68         1336.0 = 10
852.0 = 04         1477.0 = 00
941.0 = 12         1633.0 = 42
```

TEST CASE 2:

One test frequency is lower (by 5%) than the DTMF nominal frequency.

Test conditions:

```
LowBand= 731.5     HighBand= 1209.0
PhaseL = 0.0       PhaseH  = 0.0
AmpL   = 100.0     AmpH    = 100.0
```

Match (Energy) counter values:

```
697.0 = 46         1209.0 = 62
770.0 = 44         1336.0 = 06
852.0 = 08         1477.0 = 08
941.0 = 12         1633.0 = 18
```

TEST CASE 3:

One test frequency is higher (by 5%) than the DTMF nominal frequency.

Test conditions:

```
LowBand= 808.5     HighBand= 1209.0
PhaseL = 0.0       PhaseH  = 0.0
AmpL   = 100.0     AmpH    = 100.0
```

Match (Energy) counter values:

```
697.0 = 06         1209.0 = 78
770.0 = 44         1336.0 = 02
852.0 = 24         1477.0 = 04
941.0 = 20         1633.0 = 26
```

For all test cases, the match counter value provides energy information of the analysis frequency at nominal DTMF frequencies. A typical DTMF decoding algorithm will find the dominant frequency (nominal frequency with maximum counter value) in each group to find the DTMF character. The dominant frequency energy value needs to be higher than some threshold (e.g., 20) to reject the invalid frequencies.

By looking at the data from all three test cases (mainly Low Group), we can see that for two test cases, the dominant frequency is 770 Hz, and for another case, it has almost the same value as 697 Hz (Case 2). For all test cases, the dominant frequency amplitude is much higher than threshold. For Test Cases 2 and 3, the Low Group frequency is out of specification, but we are still detecting it as a valid character.

Based on the above analysis, we can say that the energy counter value follows normal DFT band. For example, if the test frequency is the same as the analysis frequency, higher energy will result. As the test frequency moves away from the analysis frequency, the energy value decreases. The rate of decrease depends on the bandwidth of the DFT response. If the response is wideband, then the energy value will decrease at a lower rate. The other parameter that affects the energy value is Twist. In the case of forward Twist, the High Group energy will be more than that of the Low Group.

Simplified DFT Band Definition

These test cases prove that a 1-bit DFT provides a wideband response and it cannot be used for DTMF decoding unless we have a method to create a narrow-band. The simplest solution is to compare the energy with some threshold and reject it if the value is less then the threshold.

If we want to meet the DTMF frequency requirement, then we should be able to reject the DTMF signal for Test Cases 2 and 3. If we plan to use the threshold method, then we will need a threshold value of around 50 for 770 Hz (we may need a little higher value if we plan to reject the frequency with an error of 3.6%). However, this creates a problem for the DTMF Twist specification.

Let's examine a test case for this.

TEST CASE 4:

Test frequencies are the same as the nominal DTMF frequency with 4 dB Reverse Twist.

Test conditions:

```
LowBand= 770.0     HighBand= 1209.0
PhaseL = 0.0       PhaseH  = 0.0
AmpL   = 100.0     AmpH    = 160.0
```

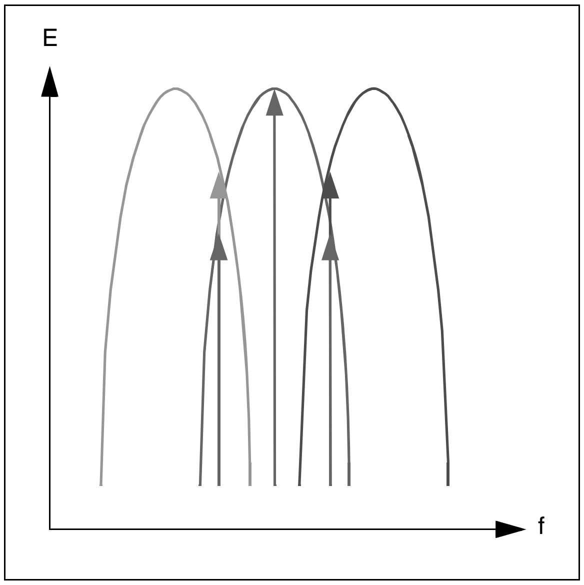
Match (Energy) counter values:

```
697.0 = 10         1209.0 = 94
770.0 = 40         1336.0 = 10
852.0 = 00         1477.0 = 08
941.0 = 04         1633.0 = 46
```

If we use a threshold rejection method, then we may use a threshold value of around 50 for 770 Hz. This will result in rejection of the above signal. If we look at the frequencies of both of the signals, then they are a valid frequency with a valid Twist. According to the DTMF specification, the detector should accept the above signal. Therefore, we may not be able to meet all of the DTMF requirements with this threshold detection method and we need to find an alternate method. If we try to plot the spectrum behavior of a 1-bit DFT, then it might look something like Figure 9.

The 1-bit DFT provides a wideband spectrum that may look something like a bell curve. The DTMF frequencies are separated at approximately 10% range. This will create an overlapped bell curve spectrum as shown in Figure 9.

FIGURE 9: 1-BIT DFT SPECTRUM



If the input frequency is in the center of this curve, we may get a higher amplitude. As the frequency moves away from center, the energy for that band decreases; however, the energy level of that particular sideband increases. If the test frequency is in the center of two analysis frequencies, then the energy level at both analysis frequencies will have almost the identical value. As the test frequency moves towards one of the analysis frequencies, the energy level for that frequency increases and it decreases for the other analysis frequency. If we find the difference between a nominal frequency and it's sideband, then we can determine the shift from center, which helps us in achieving the narrow-band response.

It is difficult to find a simple generic formula to calculate the shift from center that can meet all of the DTMF requirements. The easiest solution is to add two sideband analysis frequencies for each nominal frequency in such a way that if a frequency moves away from pass band, then the sideband frequency amplitude will be higher than the nominal frequency. The spectrum shown in Figure 9 still holds true, but both sideband frequencies are not valid DTMF frequencies. In this case, if the test frequency shows a higher amplitude than any of the sideband frequency amplitudes, then it should be rejected as it is outside the pass band. This helps in defining the narrow-band response. The sideband frequency selection is critical to achieving the desired band response. The band definition logic works in the following way:

- Add the value of the maximum detection range (Pass Band) and minimum rejection range (Stop Band). For example, for DTMF signals, it will be 1.5% + 3.5% = 5.0%. We need to add two sideband analysis frequencies at ±5.0% of the nominal DTMF frequency.
- Analyze both of the sideband frequencies for each center frequency. If one of the sideband frequencies has a higher amplitude, then reject the test frequency.

For DTMF analysis, we need to analyze eight center frequencies. This will require analysis of 16 sideband frequencies. Therefore, we need to find a 1-bit DFT for 24 analysis frequencies. This 1-bit DFT takes considerably less computation time than a conventional DFT. However, to achieve a very efficient system, we need to reduce this consumption. The actual algorithm is implemented in the following way.

- Find the dominant frequency (Nominal Frequency with Maximum Energy) in each group. These are the dominant Low Group and High Group frequencies in the DTMF signal.
- Analyze each dominant frequency with their sidebands. If any of the sideband amplitudes is higher than the dominant frequency, then reject it. It is not meeting the DTMF frequency requirements.

Let's revisit Test Case 2 and analyze its performance with the new algorithm and check it with a tighter specification.

TEST CASE 5:

One test frequency is deviated by only -3.6% from the nominal DTMF frequency.

Test conditions:

```
LowBand= 742.3     HighBand= 1209.0
PhaseL = 0.0       PhaseH  = 0.0
AmpL   = 100.0     AmpH    = 100.0
```

Match (Energy) counter values:

```
697.0 = 22         1209.0 = 62
770.0 = 64         1336.0 = 02
852.0 = 08         1477.0 = 04
941.0 = 04         1633.0 = 18
```

The Low Group dominant frequency is 770 Hz and the High Group dominant frequency is 1209 Hz. Now, we need to analyze the sidebands for these two frequencies:

```
732.0 = 102        1149.0 = 08
809.0 = 24         1269.0 = 12
```

For High Group, both sideband values are lower than the nominal frequency. Therefore, we have a valid High Group frequency. For Low Group, the -5% sideband at 732 energy (102) is higher then nominal frequency 770 energy (64). This means the Low Group frequency is out of specification and we need to reject this DTMF signal because it does not meet frequency requirements. This explains how the sideband logic helps in precisely defining the pass band. This logic works very well for band definition, regardless of the Twist. Let's revisit Test Case 4 to prove this point.

TEST CASE 6:

Test frequencies are the same as the nominal DTMF frequency with a 4 dB Reverse Twist.

Test conditions:

```
LowBand= 770.0     HighBand= 1209.0
PhaseL = 0.0       PhaseH  = 0.0
AmpL   = 100.0     AmpH    = 160.0
```

Match (Energy) counter values:

```
697.0 = 10         1209.0 = 94
770.0 = 40         1336.0 = 10
852.0 = 00         1477.0 = 08
941.0 = 04         1633.0 = 46
```

The Low Group dominant frequency is 770 Hz and the High Group dominant frequency is 1209 Hz. Now, we need to analyze the sidebands for these two frequencies.

```
732.0 = 34         1149.0 = 12
809.0 = 16         1269.0 = 04
```

Both sideband values are lower than their nominal frequency. Therefore, they will be accepted as a valid DTMF tone.

It is obvious from the previous examples that the above logic will considerably reduce the 1-bit DFT computation requirement. Now, we need to analyze 8 nominal frequencies and only 4 sideband frequencies. This makes a total of 12 analysis frequencies.

Another major issue in implementation of 1-bit DFT is the reference signal for all analysis frequencies. According to our logic, we need to calculate a 1-bit DFT for all center frequencies and analyze the sidebands of the dominant frequency. In this case, we need to create a reference signal for all nominal DTMF frequencies and their sidebands. One solution is to create a reference signal dynamically. This again, will require considerable computation power. Another solution is to precalculate the reference signal value and store it in program memory. Because we need only a 1-bit DFT, the reference signal needs to have a 1-bit value for all of the analysis points. The sampling frequency and period will decide the number of analysis points or total number of bit values for the reference signal. If we use a sampling frequency of 8 kHz and a period of 16 mS, then it will require 128 points or a 128-bit reference signal (reference table). The minimum resolution in program memory is one byte, so 16 bytes of program memory are required for one reference signal.

The advantage of precalculating the reference signal is that it considerably reduces the resource requirements. The disadvantage is that it will take more program memory space. Another major disadvantage is, since the sampling frequency is precalculated, the system must accurately maintain the sampling frequency. Any major deviation from the precalculated sampling frequency may produce catastrophic results. However, as most of the microcontroller-based systems use a crystal type oscillator, it is not difficult to achieve an accurate sampling frequency using a timer.

Now, we have a solution to achieve a 1-bit DFT without intense computation that can meet the DTMF frequency requirements. If we look at the implementation point, then we need to store the reference table for 8 nominal frequencies and 16 sideband frequencies. With this in mind, can we reduce the analysis frequency tables? Any savings will result in reduced consumption of program memory. We need sideband frequencies at ±5.0% of nominal DTMF frequencies to achieve the DTMF frequency requirements. If we look at the DTMF frequencies, we can see that they are separated by approximately a 10% difference. If we calculate the required upper and lower sideband frequencies for all nominal DTMF frequencies, then we can see that the upper sideband frequency of one DTMF frequency (731.85 Hz for 697 Hz) is almost the same as the lower sideband frequency of another DTMF frequency (731.5 Hz for 770 Hz). The difference between these two sideband frequencies is negligible. We can afford to have only one sideband frequency reference table to decrease the code size. Column D in Table 2 shows all of the nominal and sideband frequencies used for analysis, which requires eighteen reference tables in program memory.

TABLE 2: DTMF FREQUENCY CHART

Number	Nominal DTMF Frequencies	Lower Sideband Frequencies (-5%)	Upper Sideband Frequencies (+5%)	Analysis Frequencies
	A	B	C	D
1	697	662.15	731.85	662 (B1)
2	770	731.5	808.5	697 (A1)
3	852	809.4	894.6	732 (B2, C1)
4	941	893.95	988.05	770 (A2)
5	1209	1148.55	1269.45	809 (B3, C2)
6	1336	1269.2	1402.8	852 (A3)
7	1477	1403.15	1550.85	894 (B4, C3)
8	1633	1551.35	1714.65	941 (A4)
9				988 (C4)
10				1149 (B5)
11				1209 (A5)
12				1269 (B6, C5)
13				1336 (A6)
14				1403 (B7, C6)
15				1477 (A7)
16				1551 (B8, C7)
17				1633 (A8)
18				1715 (C8)

MEETING THE DTMF SPECIFICATIONS

After discussing solutions for the major problems associated with DTMF decoding, let's take another look at the specifications to evaluate how well we are meeting them.

The proposed DTMF detection system is very flexible to meet various resource-to-performance ratios; typically, the higher the resources requirement, the better the performance. Using the tools described in this section, users can define the system that best meets their requirement.

Described in this application note is a Windows® operating system-based software program (DTMF Detection Simulator) that works as a simulator for the DTMF decoder algorithm. This software generates synthesized DTMF signals for a variety of DTMF conditions. This synthesized DTMF signal is analyzed by a DTMF decoder algorithm (discussed in theory), that provides various results for test conditions. This program also generates a sine/cosine reference table formatted according to firmware requirements (see **Appendix A: "DTMF Detection Simulator" "DTMF Detection Simulator"** for more information).

The optimized system uses the following parameters that provide the best resource-to-performance ratio.

- Sampling frequency: 8 kHz
- Window period: 16 mS

Frequency Requirements

As previously discussed, sideband frequency analysis is the method of choice for defining the detection band. This method adequately meets the DTMF frequency requirements.

Power Level Requirements

The power level requirement can be met by implementing one of the following solutions.

One solution is to have enough sample points that can generate enough energy (match counter value) for the lowest required power level.

Another solution is to use basic threshold detection. The energy counter value is compared with some threshold energy value; if it is less than threshold, reject it. This value will be much lower than the value required to achieve the band definition with threshold limit. Therefore, this will not create any problem with Twist or other requirements.

Twist Requirements

This algorithm is not providing any direct control for Twist requirements. It is indirectly met by the band definition algorithm. Higher Twist in any direction will provide less amplitude for one of the DTMF frequencies. If enough points are used in analysis, then even the worst-case energy will be sufficient for analysis.

The main requirement is to have enough energy value to meet the frequency requirements. The selection of the sampling frequency and period will determine the Twist specifications of the system.

Timing Requirements

The frequency requirements are best met in frequency domain, while timing requirements are best met in time domain.

Typically, a DSP-based system analyzes the DTMF timing requirement in frequency domain. Timing analysis in frequency domain is difficult. It is possible to implement a mechanism to analyze the timing requirements in time domain for better accuracy.

During DTMF on-time, DTMF signals are present on the telephone line and during off-time, there is no activity on the telephone line. An incoming DTMF signal is converted to square wave for analysis. This conversion will provide a pulse train during DTMF on-time and will provide no pulses during DTMF off-time, which can be considered as activity on a telephone line. We can use an interrupt mechanism, along with a timer resource, to measure the duration of activity or no activity. Because this measurement is carried out in time domain, it can provide very high accuracy based on the resolution of the timer resource.

Signal-to-Noise Ratio Requirements

The DTMF system described thus far has a very good SNR. Because we are using a 1-bit A/D Converter, the noise signal needs to have sufficient amplitude in the opposite direction of the DTMF signal to change the bit level. If this amplitude coincides with the sampling point, it can cause some error in measurement.

The probability for occurrence of all conditions being met at the same time is low. If all conditions occur at the same time, the energy value will be changed by one count. This is very insignificant compared to typical energy values and provides good SNR. Actual SNR of the system will depend on the system used and performance of the 1-bit A/D Converter.

Guard-Time Test

The detection time depends on the system used. The worst-case detection time is quite good compared to other systems. The optimized system has a detection time of 16 mS, which is much less than the minimum possible on-time (23 mS) for the DTMF signal.

Talk-Off Test

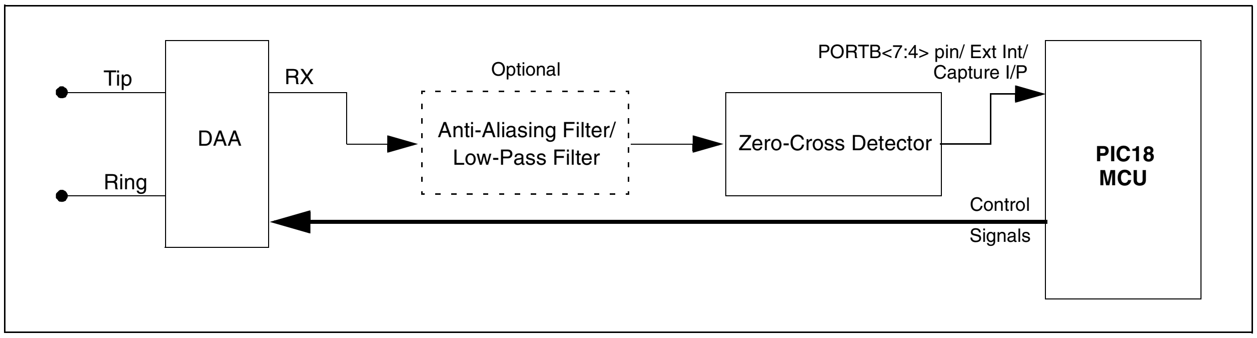
Typically, most of the voice signals do not meet the timing requirements. Therefore, an accurate timing check takes care of most of the voice interference. See the **"Possible Improvements"** section for additional information on reducing voice interference.

IMPLEMENTATION

Hardware Setup

Figure 10 shows the hardware setup required for the DTMF detector.

FIGURE 10: HARDWARE SETUP



Digital Access Arrangement (DAA) is an essential part of any circuit involving telephone line interface. It handles the telephone line interface, and meets all of the requirements of the specific country. DAA converts a balanced 2-wire signal to 4-wire for separate receive and transmit output, and detects the ring and generates a signal to indicate a ring. DAA also provides logic control for generation of an on hook/off hook condition. Any available DAA module can be used that meets a user's requirement. Users can also build their own DAA module using discrete components; however, this will not be discussed since it is beyond the scope of this application note.

Receiver (RX) output from DAA is fed to the Anti-Aliasing Filter (Low-Pass Filter). This block is not required for the operation of the DTMF detector; however, it may improve the SNR of the system. As explained in theory, the system does have good SNR, but if a user still wants to improve it, this filter can be used.

As shown in Figure 10, if an anti-aliasing filter is not used, the output of RX will be fed directly to the zero-cross detector. In this case, the zero-cross detector works as a 1-bit A/D Converter, which is a very important part of the system. We want to use a 1-bit A/D Converter that is a sine wave to square wave converter. There are various kinds of 1-bit A/D Converters available that can do the same job.

Because signal amplitude for telephone lines can vary greatly, most circuit blocks need some kind of Automatic Gain Control (AGC) circuit; however, AGC usage has some negative effects. One is that AGC is a complex circuit block, which can increase cost and complexity of the system; another is that it takes some time to set the level of the input signal which eventually increases the detection time. If we use a 1-bit A/D Converter that does not require AGC, we can lessen the detection time and reduce the cost of the system.

Let's compare the performance of the three most widely used 1-bit A/D Converters with variable amplitude.

Figure 11 shows the performance of the zero-cross detector with variable amplitude, which illustrates that the signal amplitude variation does not change the duty cycle of the signal. This provides an excellent 1-bit A/D Converter for sinusoidal signals, since the zero-cross detector is immune to amplitude change. This in turn, eliminates the requirement of an AGC circuit in this application.

FIGURE 11: ZERO-CROSS DETECTOR PERFORMANCE

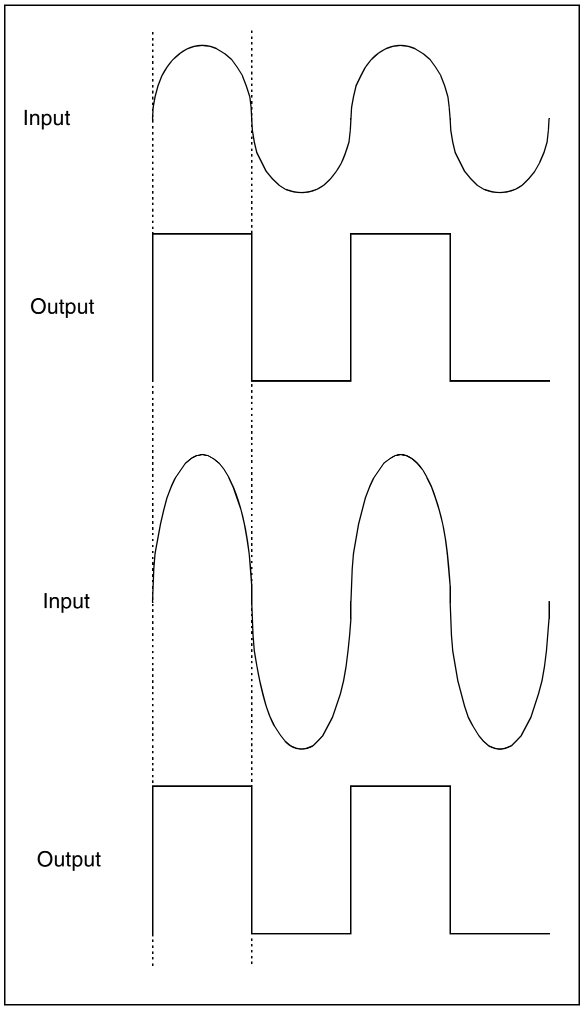


Figure 12 and Figure 13 show the performance of the Schmitt Trigger and the comparator with variable amplitude, which shows that signal amplitude variation changes the duty cycle of the output signal. Depending on the comparator level, or Upper Trip Point (UTP) and Lower Trip Point (LTP), and signal amplitude, the system may not produce any output for certain signal levels. This makes the usage of an AGC circuit a necessity.

FIGURE 12: SCHMITT TRIGGER PERFORMANCE

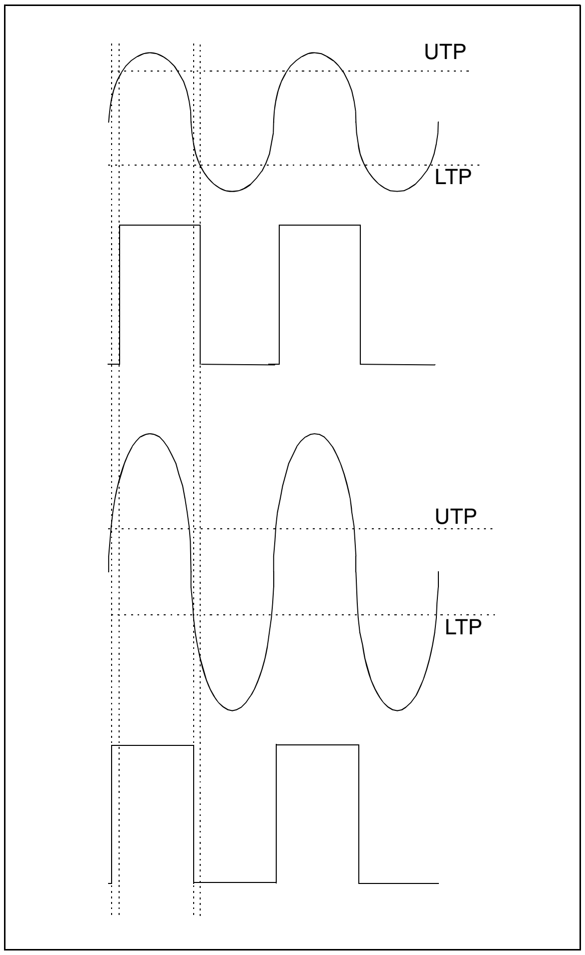
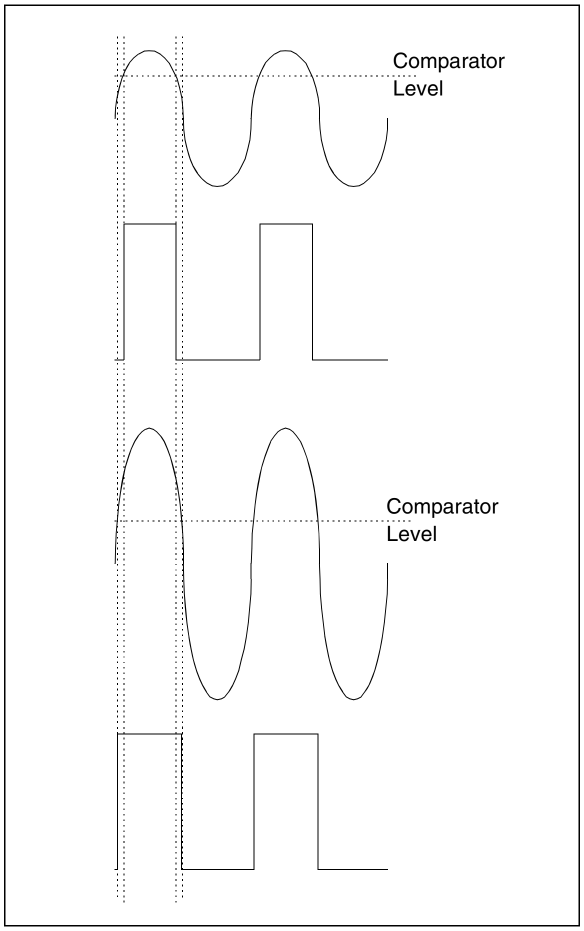


FIGURE 13: COMPARATOR PERFORMANCE



Based on the previous discussion, it seems that use of a zero-cross detector as a 1-bit A/D Converter is the best choice for this application. Users can build their own zero-cross detector circuit; however, for reference purposes, Figure B-1 in **Appendix B: "Zero-Cross Detector"** shows the circuit diagram of the zero-cross detector used for the test board. This circuit was built using the Microchip op amp, MCP604, and uses a single positive supply for the operation.

As discussed in theory, activity on a telephone line is monitored to check DTMF timing requirements. Any activity on the line is considered presence of signal, while no activity on the line indicates the pause time. The zero-cross detector generates pulses corresponding to the input signal. This means if a signal is present on a telephone line, pulses are generated as the output of the zero-cross detector. The pulses generated by the zero-cross detector are fed to the microcontroller, which can be fed to any port pin for analysis in software. If a signal is fed to a port pin, polling of that pin is required, which may result in a demand for most of the computational resources of the MCU. A better solution to this problem is to use an interrupt, which can be done by feeding this signal to one of three possible locations for analysis.

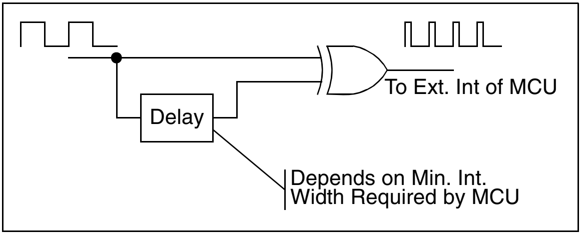
1. PORTB<7:4> pins: These pins generate an interrupt-on-change event, which can generate an interrupt on each edge of the signal.

   The only problem with the PORTB change interrupt is that read/write activity on PORTB can result in a missed interrupt. This is acceptable with the DTMF detector since this interrupt is used only for timing measurement and it can have high measurement resolution. Use of the PORTB change interrupt also reduces the resource requirement, since the remaining three PORTB pins are still available to users.

2. External Interrupt: This pin will generate an interrupt on each positive or negative direction edge.

   The only limitation to this approach is that only one type of edge (only positive or negative edges) can generate an interrupt. To generate an interrupt on each edge, a circuit similar to Figure 14 is needed, or firmware needs to modify the INT edge select bit to achieve an interrupt on every edge.

FIGURE 14: EXTERNAL INTERRUPT CIRCUIT



3. Capture mode: A Capture module can be used just like an external interrupt, or with a captured timer value. Firmware can modify the capture configuration at each interrupt to generate an interrupt on each edge, or the system can use a circuit similar to Figure 14.

The current firmware version (V1.0) supports only a PORTB change interrupt-based system. In the future, for compatibility with other similar Telecom modules (e.g., FSK detection), other methods may be supported.

FIRMWARE

The source code for the DTMF detector is designed to achieve complete background operation using two interrupts. It checks for the signal on a telephone line, and samples it upon availability. It verifies all of the timing requirements and if all timing requirements are met, it analyzes the sampled data for DTMF specifications. If all of these conditions are met, the received character is copied into a FIFO buffer.

The source code for the DTMF detector is designed in general purpose library format, and supports most devices in the PIC18 family of microcontrollers. The firmware consists of the following files.

- `DTMFDec.asm`: Main file for DTMF Decoder Firmware. This file contains all of the required source code for the DTMF detector. Users should include this file in their project.
- `DTMFDec.inc`: Include file that contains all of the definitions required to use the `DTMFDec.asm` file. Users should include this file in the source code when they want to use DTMF detector functions. This file also contains definitions of certain user-defined DTMF parameters (compile-time options). Users can change these parameters to modify DTMF detector firmware to suit their requirements.
- `MainDef.inc`: This file contains definitions of some of the system related parameters and some generalized RAM locations used for context savings.
- `P18xxx.inc`: Generic include file for the PIC18 family of microcontrollers.
- `Telecom.inc`: This file contains all of the definitions required to use different Telecom modules; for example, DTMF detector, Telecom signal generator, FSK detector and so on. Users should include this file in their main source code only after all module specific include files, and it should be included only once in user code.
- `DTMFDetT.asm`: Test file to demonstrate usage of DTMF detector module.
- `DTMFDtMT.asm`: Test file to measure MIPS usage of DTMF detector module.
- `DTMFTab.dat`: This file contains sine/cosine reference table for DTMF frequencies. This file is generated by PC software based on the selection of system parameters.
- `DTMFSBT.dat`: This file contains the sine/cosine reference table for DTMF sideband frequencies and is generated by PC software based on the selection of system parameters.

The user should copy all of these files, with the exception of `DTMFDetT.asm`, `DTMFDtMT.asm` and `DTMFdet.mcp` (project files) into their project directory. The user should create a project that utilizes a linker. Add the `DTMFDec.asm` file in the project. Include the `DTMFDec.inc` file in all source files that utilize the DTMF detector module. Please refer to the `DTMFdet.mcp` files for more information.

The user should call the `ServiceTMR1Int` function from the high priority interrupt vector and it should be the first function to provide highest priority to signal sampling. Since this is a real-time application, sampling must be done at the required interval. If any other interrupt function has higher priority and it blocks the system resources for a longer period, then the performance of the DTMF detector may be affected. Users should call the `ServicePortBInt` function from a low priority interrupt vector.

Some of the context savings code is required in user code. This is shown in the `DTMFDetT.asm` test file. Context savings will be the same for many interrupt service functions and keeping it in user code saves program memory and RAM resources.

Compile-Time Options

The firmware provides some compile-time options to modify the DTMF detector module to suit different requirements. Example 1 shows four code portions of the `DTMFDec.inc` file. Users can modify this file to change the settings for the DTMF detector system.

EXAMPLE 1: DTMFDec.inc CODE PORTION

```
;DTMF Signal timing requirement definition (Code Section 1)
DTMFMinOnTime  = .23  ;mS.
DTMFMaxOnTime  = .750 ;mS.
DTMFMinOffTime = .40  ;mS.

;system parameters (Code Section 2)
#define SamplePin           PORTB,4      ;Zero cross detector I/P pin
#define Fsamp               .8000        ;Sampling frequency in Hz
#define TotSamplePnts       .128         ;Digit should be such that
                                         ;TotSamplePnts/8 = n, where n is integer

;Define DTMF Character Storage Buffer size (Code Section 3)
#define DTMFCodeBufferSize   .20

;Define maximum allowed edges in blank period (Code Section 4)
#define MaxTotOffPeriodEdges .16
```

CODE SECTION 1

Users can modify the DTMF signal timing conditions in Code Section 1. DTMF specifications define the minimum accepted tone length (minimum on-time) of 23 mS and maximum rejected tone length (maximum on-time) of 40 mS. This means if tone length is greater than 40 mS, it must be accepted as a valid DTMF tone length, and if tone length is less than 23 mS, it must be rejected as a valid DTMF tone. Similarly, minimum pause time (intercharacter period) is defined as 40 mS.

Because of the system's excellent capability with timing measurement and faster detection time, it is possible to detect tone with length less than 23 mS. DTMF specifications do not specify the maximum on-time for the DTMF signals. However, if maximum tone duration is known, then define it. This will be helpful in rejecting other tone signals based on timing requirements. By default, the system on and off period requirements are defined to DTMF specifications. However, users can change these values by modifying Code Section 1 to meet their custom requirements.

CODE SECTION 2

Code Section 2 defines the DTMF detector system parameter. Performance of the system with a particular parameter can be realized with the included DTMF Detection Simulator PC software. The DTMF system-related parameters include sampling frequency for DTMF signals and window period for same. Depending on the actual system implementation, users can use different PORTB pins to implement the system (users can define the pin under `SamplePin`).

Note:	It is the responsibility of the user to define this pin as an input.

CODE SECTION 3

Code Section 3 provides selection of the FIFO buffer size for the storage of received DTMF characters. If the user is not expecting bulk data, minimum size should be used to save RAM. It is up to the user to select the best size to meet their requirement.

CODE SECTION 4

Code Section 4 provides the selection of maximum allowed signal edges in a pause period. Ideally, there should not be any signals or pulses during pause time. However, in a real system, because of noise and other problems, the DTMF system may get a few edges in the pause period. This will be considered an end of the pause period and depending on the occurrence of the signal, an invalid pause time may be detected which can lead to the rejection of the received signal.

The solution for this is to use an external low-pass filter to remove these noise pulses. Nevertheless, to avoid the usage of an external filter, firmware provides this facility. Users can define the maximum allowed pulses in a pause period and if the total number of pulses that occur in a pause period are less than the defined value, then firmware will consider it as a valid pause period. This reduces the requirement for an external low-pass filter. The user is the best judge to select an appropriate value.

The majority of the system timing requirements are based on the main oscillator frequency. Users need to define the main oscillator frequency in the `MainDef.inc` file (see Example 2). The accurate oscillator frequency is critical for the proper operation of the system.

Note:	The user should ensure that the oscillator frequency is stable and accurate.

EXAMPLE 2:

```
#define FOSC .20000000 ;Oscillator Frequency
```

Interrupt Handling

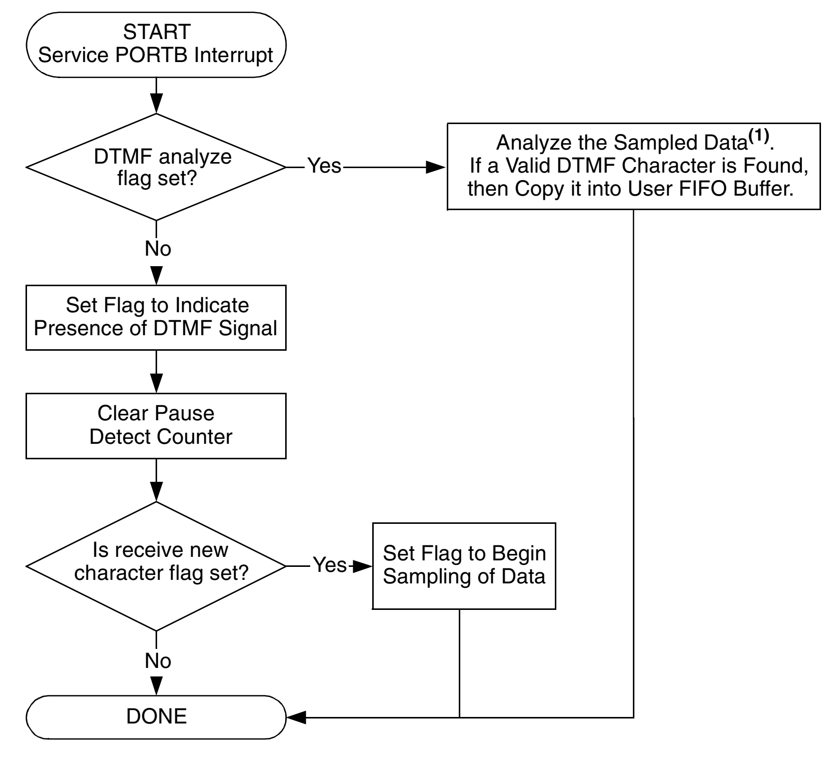
The DTMF detection system is based on two interrupts to achieve the complete background operation. Timer1 or Timer3 is used to achieve a precise sampling frequency, while a PORTB interrupt is used to monitor the activity on the telephone line for timing measurements. The timer interrupt needs to be highest priority interrupt for proper operation. The PORTB change interrupt is assigned to the low priority interrupt. Timer overflow provides an interrupt at the sampling frequency. The firmware samples the bit value of the signal and stores it in RAM. The algorithm implementation is slightly different than what was discussed in theory. The firmware first samples the data and stores the bit value in RAM. Then, it verifies the timing requirements. If all the timing requirements are met, then it performs the 1-bit DFT. This increases the RAM requirement but it considerably decreases the processor power requirement.

The PORTB interrupt sets a flag on activity. The timer interrupt uses this flag to increment the on or off-time counter for timing measurements. The timer interrupt increments one counter for off-time measurement. The PORTB interrupt clears the same counter. Therefore, if a signal is present, the off counter value will never go high. In the absence of a signal, the off-time counter will keep incrementing. One noise glitch during off-time can result in improper measurement of the off-time. Users can define the compile-time option `MaxTotOffPeriodEdges` to avoid this problem. If it is defined, then firmware will wait for the number of edges defined in `MaxTotOffPeriodEdges` before clearing the off counter. This will significantly improve the noise performance for off-time calculation.

1-bit DFT calculation requires some processing time in milliseconds (depends on FOSC). If this is performed on the timer interrupt, then it can block the system resources for a longer period. To avoid this, firmware samples all of the data on timer interrupt and checks for the timing requirement. If all of the conditions are met, a flag is set and a PORTB interrupt is generated. All of the 1-bit DFT analysis is performed on the low priority interrupt, which allows other high priority interrupts to be used.

Figure 15 shows the flowchart for the PORTB change interrupt handler and Figure 19 shows the flowchart for the Timer interrupt handler.

FIGURE 15: PORTB CHANGE INTERRUPT HANDLER


START
Service PORTB Interrupt
DTMF analyze flag set?
Yes
Analyze the Sampled Data(1). If a Valid DTMF Character is Found, then Copy it into User FIFO Buffer.
No
Set Flag to Indicate Presence of DTMF Signal
Clear Pause Detect Counter
Is receive new character flag set?
Yes
Set Flag to Begin Sampling of Data
No
DONE

Note 1: See Figure 16, DTMF Data Analysis.

FIGURE 16: DTMF DATA ANALYSIS

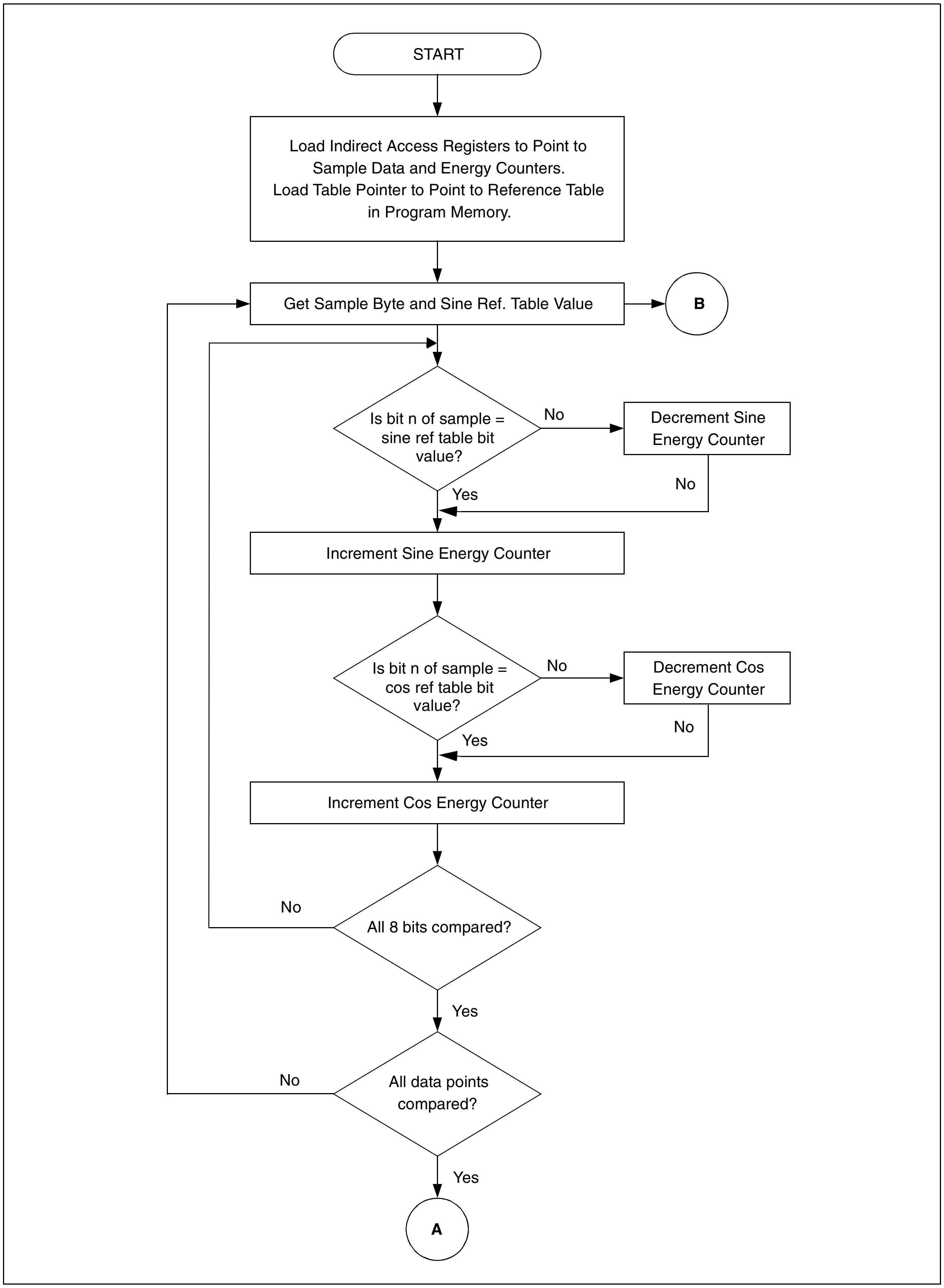

START
Load Indirect Access Registers to Point to Sample Data and Energy Counters. Load Table Pointer to Point to Reference Table in Program Memory.
Get Sample Byte and Sine Ref. Table Value
B
Is bit n of sample = sine ref table bit value?
No
Decrement Sine Energy Counter
No
Yes
Increment Sine Energy Counter
Is bit n of sample = cos ref table bit value?
No
Decrement Cos Energy Counter
No
Yes
Increment Cos Energy Counter
No
All 8 bits compared?
Yes
No
All data points compared?
Yes
A

FIGURE 17: DTMF DATA ANALYSIS (CONTINUED)

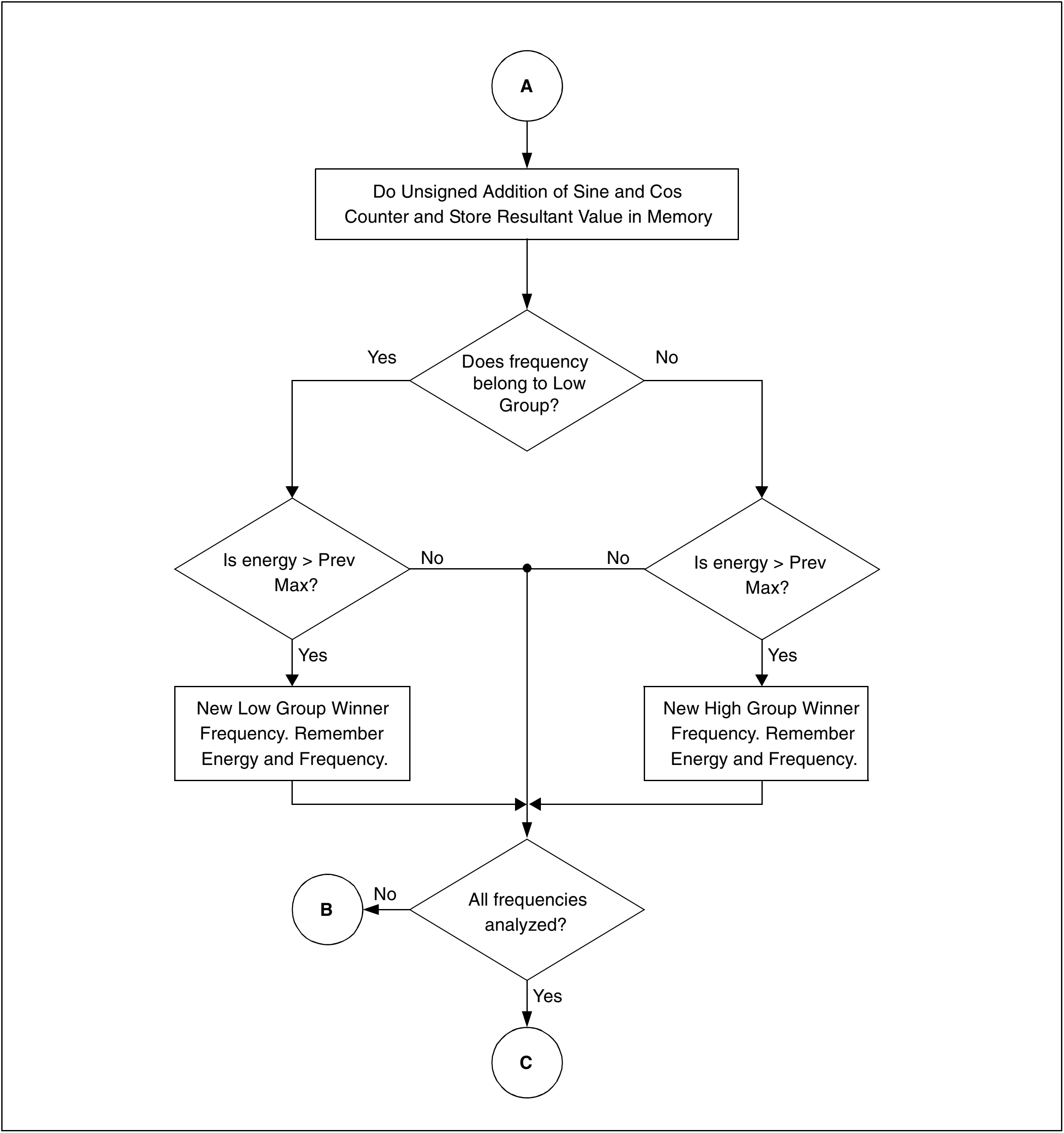

A
Do Unsigned Addition of Sine and Cos Counter and Store Resultant Value in Memory
Does frequency belong to Low Group?
Yes
No
Is energy > Prev Max?
No
No
Is energy > Prev Max?
Yes
Yes
New Low Group Winner Frequency. Remember Energy and Frequency.
New High Group Winner Frequency. Remember Energy and Frequency.
All frequencies analyzed?
No
B
Yes
C

FIGURE 18: DTMF DATA ANALYSIS (CONTINUED)

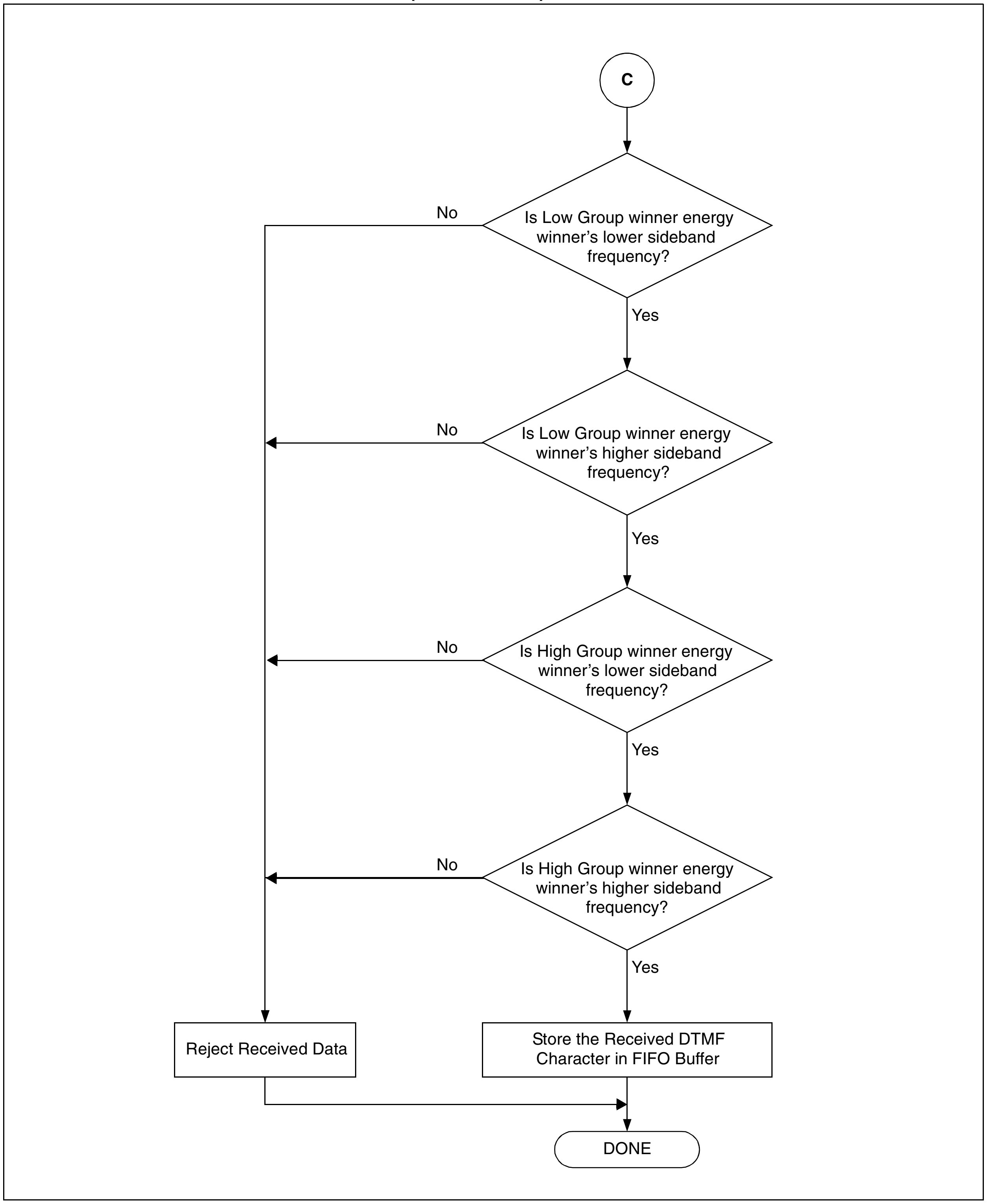
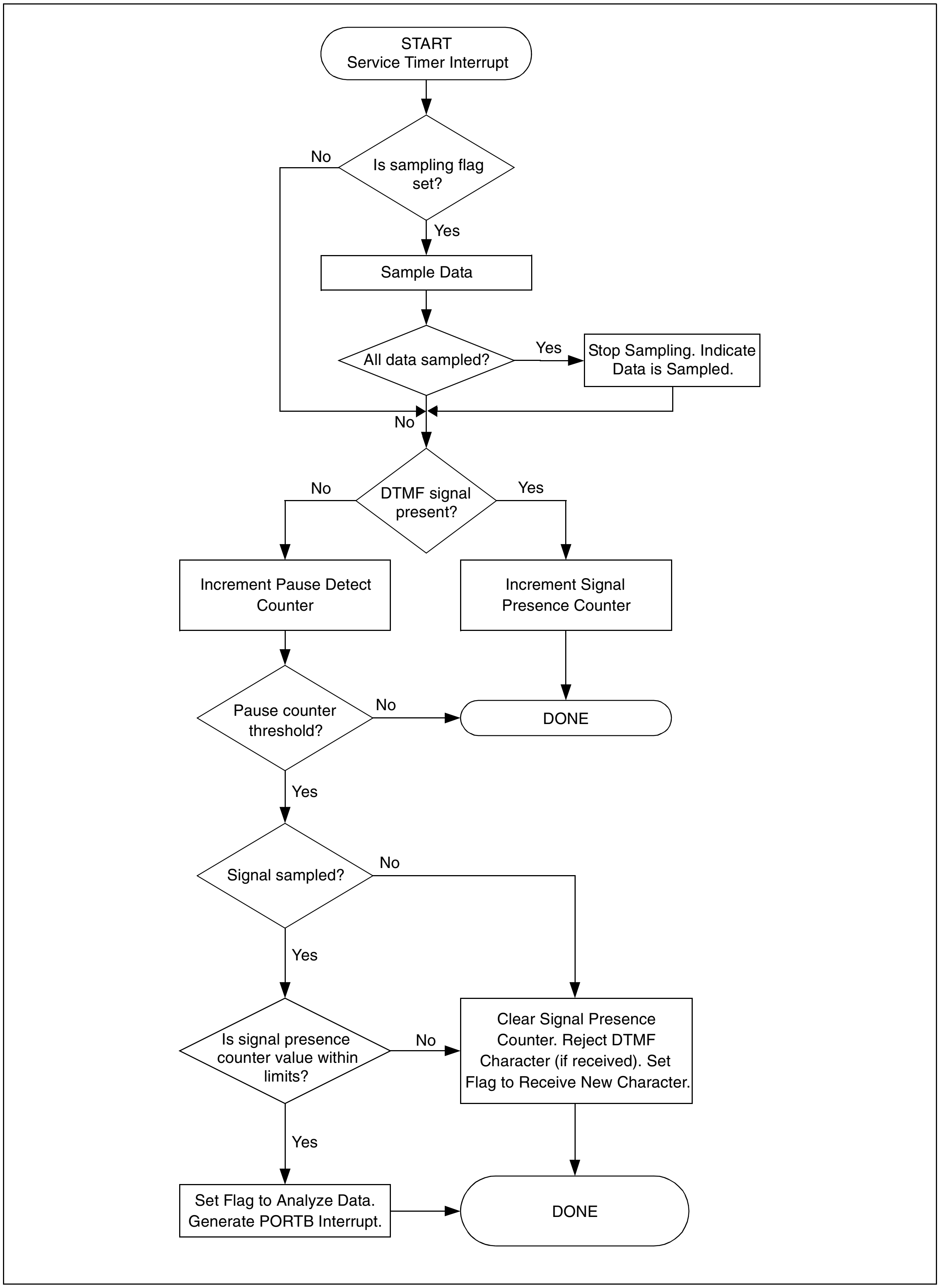

C
Is Low Group winner energy winner's lower sideband frequency?
No
Yes
Is Low Group winner energy winner's higher sideband frequency?
No
Yes
Is High Group winner energy winner's lower sideband frequency?
No
Yes
Is High Group winner energy winner's higher sideband frequency?
No
Yes
Reject Received Data
Store the Received DTMF Character in FIFO Buffer
DONE

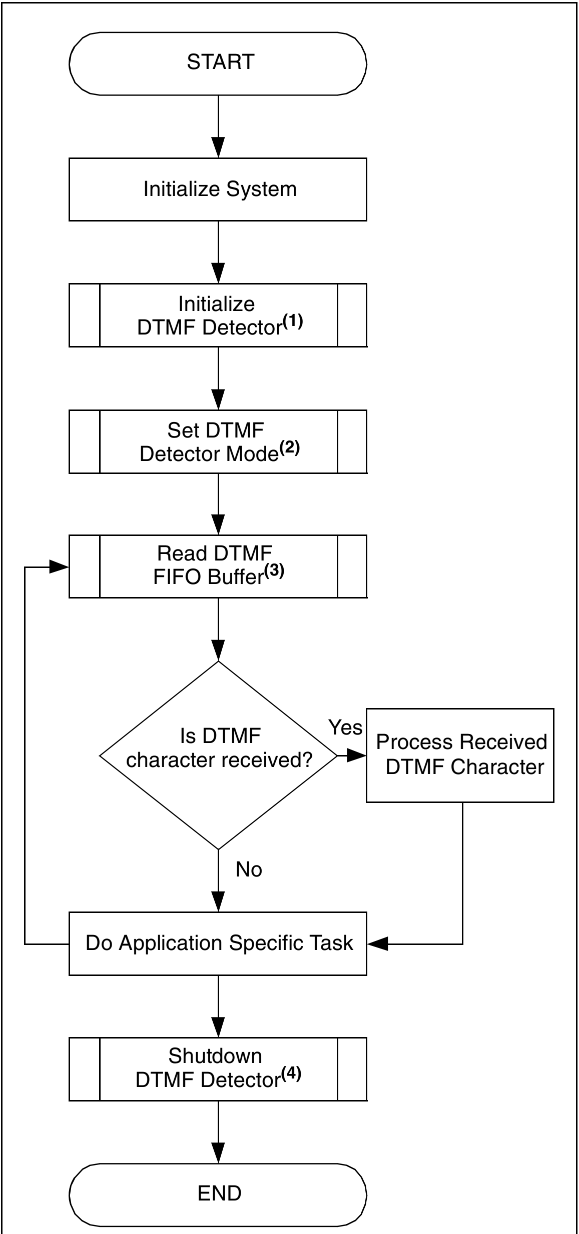
FIGURE 19: TIMER INTERRUPT HANDLER


START
Service Timer Interrupt
Is sampling flag set?
No
Yes
Sample Data
All data sampled?
Yes
Stop Sampling. Indicate Data is Sampled.
No
DTMF signal present?
No
Yes
Increment Pause Detect Counter
Increment Signal Presence Counter
Pause counter threshold?
No
DONE
Yes
Signal sampled?
No
Yes
Is signal presence counter value within limits?
No
Clear Signal Presence Counter. Reject DTMF Character (if received). Set Flag to Receive New Character.
Yes
Set Flag to Analyze Data. Generate PORTB Interrupt.
DONE

User Code

To enable the DTMF detector module, the `SetDTMFDecoderMode` macro and the following functions must be called. Figure 20 shows the simplified block diagram for the user code.

FIGURE 20: USER CODE


START
Initialize System
Initialize DTMF Detector(1)
Set DTMF Detector Mode(2)
Read DTMF FIFO Buffer(3)
Is DTMF character received?
Yes
Process Received DTMF Character
No
Do Application Specific Task
Shutdown DTMF Detector(4)
END

Note 1: Call `InitDTMFDecoder` function.
2: Call `SetDTMFDecoderMode` macro.
3: Call `ReadDTMFDecBuf` function.
4: Call `ShutDownDTMFDecoder` macro.

USER CODE FUNCTIONS

`InitDTMFDecoder`

This routine is used to initialize the DTMF decoder. Users should call this function to start using the DTMF decoder module

Pre-Condition

None

Input

None

Output

None

Stack Requirement

1 level deep

Side Effects

Databank, STATTUS, W changed

`ServiceTMRInt`

This routine is used to sample the o/p of the zero-cross detector. Once all the data is captured, it generates a PORTB interrupt to analyze data. It also checks for some of the timing requirements. This function should be called from a high priority interrupt vector and should have highest priority.

Pre-Condition

Should be highest priority interrupt for proper operation.

Input

None

Output

None

Stack Requirement

1 level deep

Side Effects

Databank, STATTUS, W changed

`ServicePortBInt`

This routine is used to monitor the telephone line activity and generates data to find the DTMF on-time and off-time. This function also analyzes the sampled data to find a valid DTMF character. This function should be called from a low priority interrupt vector.

Pre-Condition

None

Input

None

Output

If a valid DTMF character is found, it is stored in the DTMF FIFO buffer.

Stack Requirement

1 level deep

Side Effects

Databank, STATTUS, W changed

`ReadDTMFDecBuf`

This routine is used to read the DTMF decoder FIFO buffer. If this buffer does not contain any data, then it returns '`0`'; otherwise, it returns the ASCII equivalent of the character.

Pre-Condition

None

Input

None

Output

W reg = ASCII value of DTMF character (if available); otherwise, W reg = `0`.

Stack Requirement

2 levels deep

Side Effects

Databank, W, STATUS, FSR0 and Table Access registers changed. Overlaid RAM locations TempD1, TempD2 and TempD3 used.

USER MACROS

`SetDTMFDecoderMode`

This macro is used to enable the DTMF detector module.

Pre-Condition

None

Input

None

Output

None

Stack Requirement

None

Side Effects

None

`ShutDownDTMFDecoder`

This macro is used to shut down the DTMF detector module.

Pre-Condition

None

Input

`DisablePORTBInt` (if PORTB change interrupt is not used by any other resource).

`KeepPORTBInt` (if PORTB change interrupt is being used by any other resource).

Output

None

Stack Requirement

None

Side Effects

None

Status information

`DTMFDecBufSize` – Contains the information of number of bytes available in the FIFO buffer. Do not modify this parameter.

Error messages

Message: `"Maximum data buffer size is limited to 127"`

Detail: Maximum allowed size of FIFO buffer is 127 bytes; if the user tries to define a larger FIFO buffer, the above message is generated.

Workaround: Use FIFO buffer size of less than 127 bytes.

DTMF DETECTOR TESTING

The International Telecommunications Union (ITU) and Bellcore (Telcordia Technologies) have set recommendations for DTMF detector performance. There are some gray areas in the DTMF specifications and recommendations, and a lack of clarification on some specifications that can lead to various interpretations for test methods. For example, ITU Frequency Tolerance recommendations are:

- Maximum accepted frequency offset: 3.5%
- Minimum rejected frequency offset: 1.5%

ITU does not specify whether one or both frequencies of the DTMF signal must be able to handle the 1.5% error simultaneously. This can create the following test conditions:

- Nominal Low Group, Nominal High Group
- +FR error on Low Group, Nominal High Group
- -FR error on Low Group, Nominal High Group
- Nominal Low Group, +FR error on High Group
- Nominal Low Group, -FR error on High Group
- +FR error on Low Group, +FR error on High Group
- +FR error on Low Group, -FR error on High Group
- -FR error on Low Group, +FR error on High Group
- -FR error on Low Group, -FR error on High Group

where FR is the selected frequency range.

If we consider 1.5% as the range, an infinite number of possible frequency steps are created for the total possible test cases. For the ITU specification, the required precision in both timing and frequency tolerances are very difficult to achieve with most of the software-based DTMF decoders since they use some form of DFT. The goal of the detector performance test was to test at all possible conditions to meet all DTMF requirements; however, this can lead to a multitude of test cases.

Considering the wide variety of possible test cases, it is very important to know how the DTMF detector is tested. The detector was tested in the following ways:

- *Mathematical Modeling*: The mathematical simulator software (DTMF Detector Simulator) was created to model the proposed system (see **Appendix A: "DTMF Detection Simulator" "DTMF Detection Simulator"**). The firmware algorithm was checked with a variety of synthesized DTMF test conditions. This made it possible to test firmware algorithm with thousands of DTMF test conditions quickly.
- *Using an Arbitrary Waveform Generator:* An Agilent Technologies arbitrary waveform generator was used to create a variety of DTMF test conditions. The resulting information was input to the DTMF detector system to verify the performance of the system.
- *Telephone Line Simulator:* The Sage Instruments 930A Communication Test Set telephone line simulator was used to check the performance of the DTMF detector.
- *Actual Telephone Line:* The DTMF detector system was finally verified with an actual POTS line.

The MIPS requirements for the DTMF detector can be calculated using certain test files. See **Appendix C: "Test Files" "Test Files"** for details on these test files.

RESOURCE REQUIREMENTS

The following are the resource requirements for the optimized DTMF detector system:

- Processor power – approximately 0.8 MIPS (F_{SAMP} = 8 kHz, window period = 16 mS, 10 characters/second)
- RAM
  - Access – 16 bytes
  - Banked – 30 bytes + FIFO buffer size
  - Banked – 17 bytes (Context savings)
  - Banked – 3 bytes (Overlaid)
- ROM – 1580 bytes
- Peripherals
  - Timer1 or Timer3
  - One PORTB Change Interrupt pin

The use of different sampling frequency, window period and character throughput rate, can substantially change the resource requirements and performance. For example, use of a 24 mS window period, with the same sampling frequency of 8 kHz, will reduce the total failures but will increase the RAM requirement. In addition, it will take slightly more MIPS and have a higher detection time.

POSSIBLE IMPROVEMENTS

A different sampling frequency and period can considerably change the performance. The user can use PC software to find the best system that suits their requirements.

It is still possible to get better performance with some modifications in the algorithm. Some of the possible improvements are:

- Use two levels of analysis. The performance of DFT is dependent on the sampling frequency and period. The basic 1-bit DFT provides wideband response; however, it can quickly detect the dominant frequencies. Therefore, the solution is to perform two levels of analysis.

  To do this, first do a quick sampling (e.g., 8 mS) to find the dominant frequencies. Now, change the sampling frequency and period in such a way that it provides better DFT for the two dominant frequencies. Sample it again for a longer period (e.g., 16 mS). This will provide better DFT results. Then, use an enhanced algorithm to achieve better performance.
- It is possible to improve power level and Twist performance. Analysis of the dominant frequency energy can provide the information on Twist. In the case of forward Twist, the Low Group dominant energy will be higher than the High Group dominant energy and vice versa. The difference in energy can provide the Twist value. This information can be used to create various threshold values for power level and Twist requirements.
- Talk-off test improvement. Human voice signals are rich in harmonics, while DTMF signals are pure tone signals. This property can be used to improve the talk-off performance of the DTMF detector by analyzing the second harmonic of the DTMF nominal frequency. It is a voice signal if they contain significant energy and needs to be rejected; otherwise, it is a DTMF tone. This will require 8 more reference tables and analysis of two dominant frequency harmonic energies.

CONCLUSION

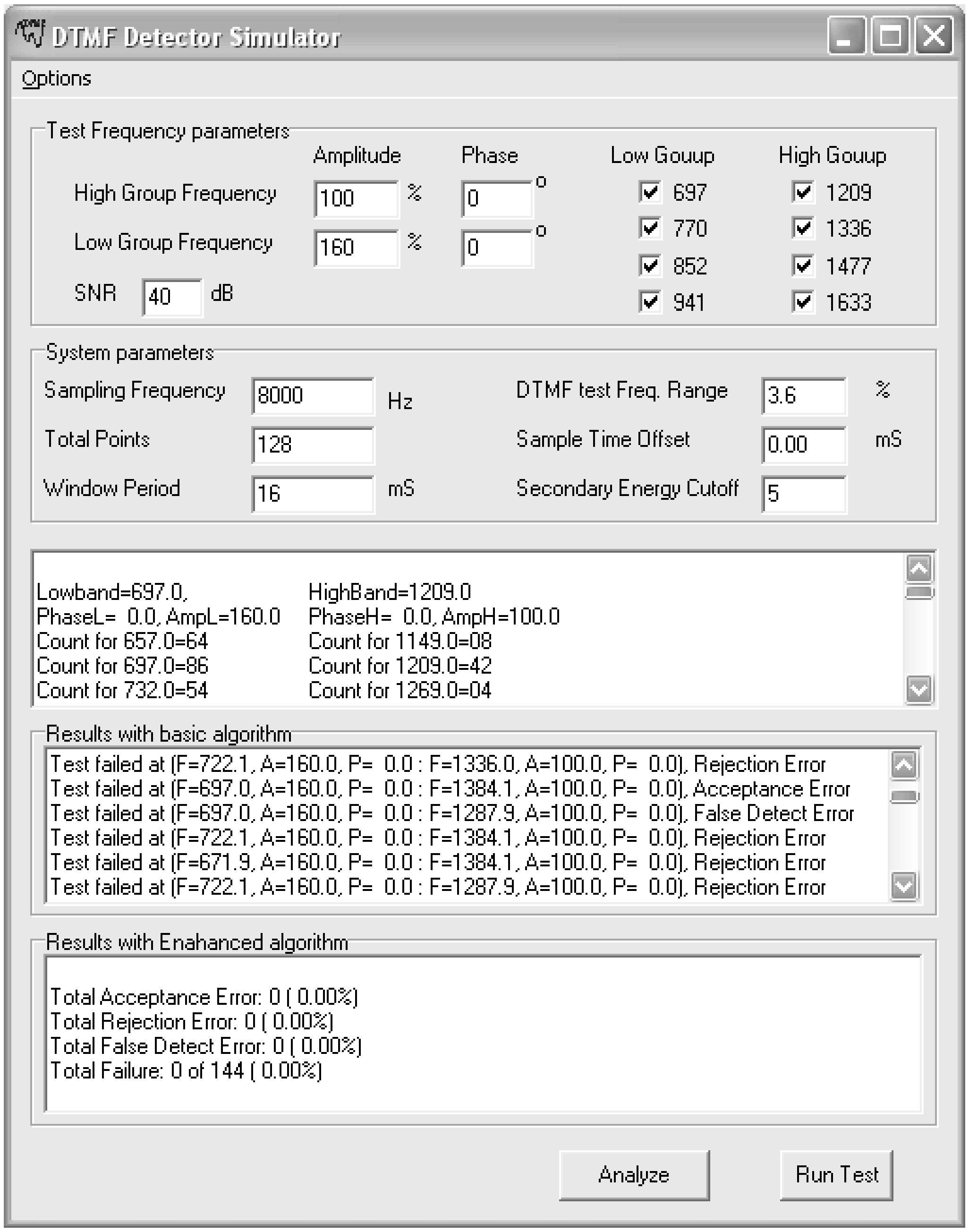
We discussed the new concept of decoding DTMF signals using a PIC18 microcontroller. We have seen that the DTMF decoder system is flexible enough to achieve various performance-to-resource ratios. The system meets all or most of the DTMF requirements depending on the implementation. In summary, we discussed the following:

- DTMF decoder specifications and requirements.
- Implemented a 1-bit DFT without intense computation.
- The method to define the bands to meet DTMF frequency requirements.
- The method to accurately check the timing requirements.
- The required hardware setup that uses very little external resources and avoids the necessity of AGC.
- The firmware details. The firmware is based on two interrupts to achieve complete background operation. The firmware implements a FIFO buffer that eases the reception of bulk data. Firmware provides some compile-time options to customize the module for individual requirements.
- The resource requirements of the system. The system uses very little internal and external resources and is very efficient on MIPS usage.
- The software that works as a simulator and implements a mathematical algorithm model, and also works as a development tool exporting data to firmware.

APPENDIX A: DTMF DETECTION SIMULATOR

Figure A-1 shows the user interface window for the DTMF Detection Simulator software.

FIGURE A-1: DTMF DETECTION SIMULATOR USER INTERFACE



Test Frequency parameters

	Amplitude	Phase	Low Gouup	High Gouup
High Group Frequency	100 %	0 °	✓ 697	✓ 1209
Low Group Frequency	160 %	0 °	✓ 770	✓ 1336
			✓ 852	✓ 1477
SNR 40 dB			✓ 941	✓ 1633

System parameters

Sampling Frequency	8000 Hz	DTMF test Freq. Range	3.6 %
Total Points	128	Sample Time Offset	0.00 mS
Window Period	16 mS	Secondary Energy Cutoff	5

```
Lowband=697.0,              HighBand=1209.0
PhaseL=  0.0, AmpL=160.0    PhaseH=  0.0, AmpH=100.0
Count for 657.0=64          Count for 1149.0=08
Count for 697.0=86          Count for 1209.0=42
Count for 732.0=54          Count for 1269.0=04
```

Results with basic algorithm

```
Test failed at (F=722.1, A=160.0, P=  0.0 : F=1336.0, A=100.0, P=  0.0), Rejection Error
Test failed at (F=697.0, A=160.0, P=  0.0 : F=1384.1, A=100.0, P=  0.0), Acceptance Error
Test failed at (F=697.0, A=160.0, P=  0.0 : F=1287.9, A=100.0, P=  0.0), False Detect Error
Test failed at (F=722.1, A=160.0, P=  0.0 : F=1384.1, A=100.0, P=  0.0), Rejection Error
Test failed at (F=671.9, A=160.0, P=  0.0 : F=1384.1, A=100.0, P=  0.0), Rejection Error
Test failed at (F=722.1, A=160.0, P=  0.0 : F=1287.9, A=100.0, P=  0.0), Rejection Error
```

Results with Enahanced algorithm

```
Total Acceptance Error: 0 ( 0.00%)
Total Rejection Error: 0 ( 0.00%)
Total False Detect Error: 0 ( 0.00%)
Total Failure: 0 of 144 ( 0.00%)
```

TABLE A-1: DTMF DETECTION SIMULATOR PARAMETERS

Parameter	Type	Description
High Group Amplitude	Edit Value	Relative amplitude of High Group frequencies.
Low Group Amplitude	Edit Value	Relative amplitude of Low Group frequencies.
High Group Phase	Edit Value	Phase value for High Group frequencies.
Low Group Phase	Edit Value	Phase value for Low Group frequencies.
SNR (dB)	Edit Value	Signal-to-noise ratio at test point. This parameter defines the noise level based on the amplitude of two sinusoidal.
697 (Low Group)	Check Box	Analyze/Do not analyze components with 697 Hz as Low Group frequency.
770 (Low Group)	Check Box	Analyze/Do not analyze components with 770 Hz as Low Group frequency.
852 (Low Group)	Check Box	Analyze/Do not analyze components with 852 Hz as Low Group frequency.
941 (Low Group	Check Box	Analyze/Do not analyze components with 941 Hz as Low Group frequency.
1209 (High Group)	Check Box	Analyze/Do not analyze components with 1209 Hz as High Group frequency.
1336 (High Group)	Check Box	Analyze/Do not analyze components with 1336 Hz as High Group frequency.
1477 (High Group)	Check Box	Analyze/Do not analyze components with 1477 Hz as High Group frequency.
1633 (High Group)	Check Box	Analyze/Do not analyze components with 1633 Hz as High Group frequency.
System Parameters		
Sampling Frequency	Edit Value	Sampling frequency in Hz.
Total Points	Edit Value	Total points to be analyzed.
Window Period	Edit Value	Period for sampling the signal in mS. If a window period is defined, software will calculate the total points and vice versa.
DTMF Test Frequency Range	Edit Value	Defines the frequency offset in % from center to create the following test conditions (FR = frequency range value) • Nominal Low Group frequency, Nominal High Group frequency • +FR error on Low Group frequency, Nominal High Group frequency • -FR error on Low Group frequency, Nominal High Group frequency • Nominal Low Group frequency, +FR error on High Group frequency • Nominal Low Group frequency, -FR error on High Group frequency • +FR error on Low Group frequency, +FR error on High Group frequency
Generate Sample Table	Options Menu Command	This command generates a sample DTMF signal value table based on selected parameters. This command is useful to test the firmware for a variety of DTMF test conditions without actually feeding the DTMF signal. To use this feature, the user must select the `UseSampleTable` compile-time option in firmware. If this option is used, firmware will not sample the actual signal, but will use values from the sample table generated by this command.
Run Test	Command Button	This command runs the DTMF simulator test for the selected parameters and displays various results in the result section.

Note: The DTMF Detection Simulator software provides a tool to estimate system performance. The simulator does not consider the signal timing requirements. Remember that the actual system performance may vary from the predicted system.

TABLE A-1: DTMF DETECTION SIMULATOR PARAMETERS (CONTINUED)

Parameter	Type	Description
Analyze	Command Button	This command can be used for analyzing the DTMF algorithm for a range of parameters. Clicking on the button will display one dialog box. The user can select Twist requirement and the total amplitude steps for each Twist. Similarly, the user can enter the total steps in phase value and frequencies of each sinusoidal and any initial phase difference value here. For each frequency step, the nine possible combinations explained in the DTMF test frequency range will be tested. Depending on the speed of the computer and the total steps selected in each analysis, it can take a few minutes to finish the operation. The progress bar in the lower left corner displays the progress of the analysis. The process generates the following data files for user information. • `DTMFBR.dat` – This file provides the information on test cases that failed with the basic algorithm. It also provides the same information as the basic algorithm's result display. • `DTMFER.dat` – This file provides test case information that failed with enhanced algorithm. It also provides the same information as enhanced algorithm's result display. • `DTMFres.dat` – This file provides test case information on match counter values of all test cases. It also provides the same information as match counter values result display.
Results		
Match Counter Values	Result Display	Values of match counter (energy level) at all analysis frequencies and sideband are displayed here.
Result with Basic Algorithm	Result Display	Basic algorithm is applied on the DTMF signals for analysis. In this algorithm, frequency with highest match counter (energy level) value in each group is considered a dominant frequency. If the difference in value of the two highest match counters is less than threshold (secondary energy cutoff), then no dominant frequency is found. This results in rejection of DTMF character. On rejection, three error conditions are displayed: 1. Acceptance Error – If both frequencies are within the DTMF frequency range and a character was rejected, an acceptance error is generated. 2. Rejection Error – If one of the frequencies is outside the DTMF frequency range and a character was accepted, a rejection error is generated. 3. False Detect Error – If a detected DTMF character is different than the generated character, a false detect error is generated. If any of the above errors are generated, the test conditions and failure type are displayed. Statistics of total failure, failure for each error, and total test cases are displayed following the other information.

Note: The DTMF Detection Simulator software provides a tool to estimate system performance. The simulator does not consider the signal timing requirements. Remember that the actual system performance may vary from the predicted system.

TABLE A-1: DTMF DETECTION SIMULATOR PARAMETERS (CONTINUED)

Parameter	Type	Description
Results With Enhanced Algorithm	Result Display	The enhanced algorithm is applied on the DTMF signals for analysis. In this algorithm, the first algorithm is applied on the signals to find a dominant frequency pair. Once a dominant frequency pair is found, its sideband frequencies are analyzed to define the detection band. If any of the sideband frequency energy levels is higher than the center frequency, that character is rejected. On rejection, the following error conditions are displayed: 1. Acceptance Error – If both frequencies are within the DTMF frequency range and a character was rejected, an acceptance error is generated. 2. Rejection Error – If one of the frequencies is outside the DTMF frequency range and if character was accepted, a rejection error is generated. 3. False Detect Error – If a detected DTMF character is different than the generated character, a false detect error is generated. If any of the above errors are generated, the test conditions and failure type are displayed. Statistics of total failure, failure for each error and total test cases are displayed following the other information.
Generate Sine/Cosine Table	Options Menu Command	This command generates a sine/cosine table for reference frequencies and their sideband frequency. It generates `DTMFTab.dat` (sine/cosine reference frequency table) and `DTMFSBT.dat` (sine/cosine sideband frequency table) files that are used by the firmware.
Generate Sine w/f Table	Options Menu Command	This command generates a sine waveform table for use by the Telecom signal generator firmware.

Note: The DTMF Detection Simulator software provides a tool to estimate system performance. The simulator does not consider the signal timing requirements. Remember that the actual system performance may vary from the predicted system.

APPENDIX B: ZERO-CROSS DETECTOR

The following figure shows the schematic diagram for the zero-cross detector, which uses the Microchip op amp, MCP604. Capacitor C1 is used to block the DC component. Since the op amp performs best in its linear region, two resistors (R3 and R4) are used to bias the inputs at VDD/2. The AC coupled signal is fed to one input of the op amp. The positive direction signal will create a voltage difference at the op amp input. Because of high gain, this will result in a logic state change at output; however, because of op amp slew rate specifications, the circuit may not provide 50% duty cycle at output. This problem will be more significant at lower amplitude. The transistor switch at the op amp output provides a quick transition to achieve 50% duty cycle. Please refer to the schematic in Figure B-1 for more details. This circuit was tested for amplitude sensitivity and it provides constant output for signals between 50 mV to 5.0V.

The PCB layout and power supply decoupling play an important role for the performance of the zero-cross detector.

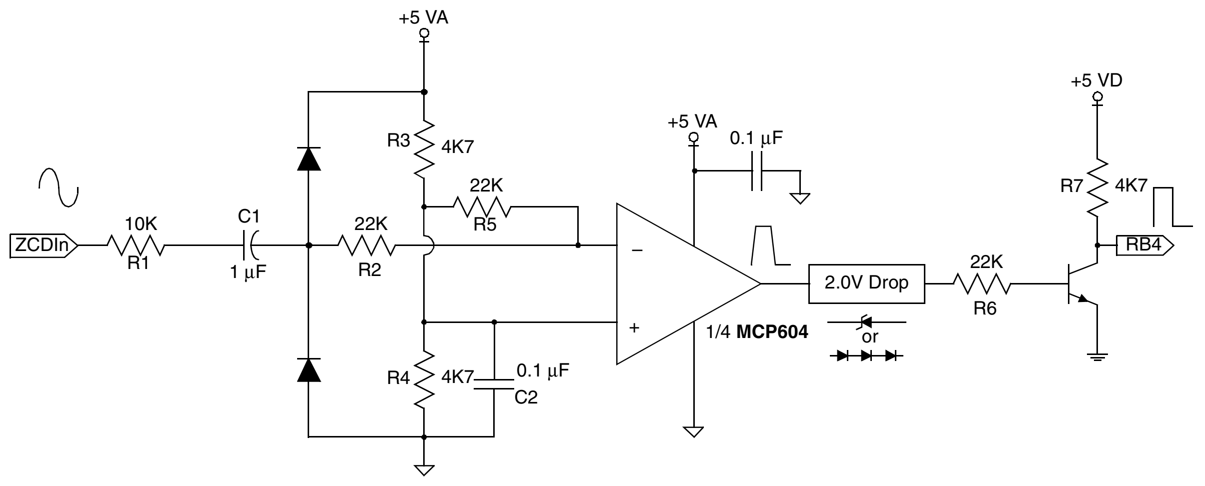
The following simple tests can help in verifying the performance of the zero-cross detector.

In the absence of signal (no input or pause period), RB4 should be held at a constant DC level. If the system is noisy, then some pulses may be observed on RB4. This may cause problems with pause time detection.

Feed a single frequency sinusoidal at input and observe the output. The output should be a 50% duty cycle square wave. Vary the amplitude of the incoming signal over a wide range (expected DTMF signal range). This should have a negligible effect on the square wave output (RB4) in terms of duty cycle, frequency, and amplitude.

This circuit was designed to create a low-cost zero-cross detector. The MCP606 kind of generic op amp was used. The other op amp on the same chip can be used for other system requirements. The VDD/2 biasing was used to improve the performance and avoid supply line noise dominance. The transistor switch is used to overcome op amp slew rate or GBWP limitations. This circuit is provided for reference only. It is possible to implement this circuit in a variety of ways. The user can select any circuit that provides comparable performance.

FIGURE B-1: ZERO-CROSS DETECTOR SCHEMATIC DIAGRAM



APPENDIX C: TEST FILES

To test the functionality and demonstrate the usage of the DTMF detector, the following test files were developed.

- `DTMFDetT.asm`: This test file explains the usage of the FSK detector module.
- `DTMFDtMT.asm`: This test file is useful to calculate the MIPS requirement of the DTMF detector module for different system implementation (see the **"Firmware"** section for more information).

The `DTMFDtMT.asm` file was developed to measure the MIPS requirement of firmware for a particular operating condition. Because the MIPS requirement depends on various factors and can change considerably, it can be approximated for certain system implementation. The basic logic used is to test the time available to the foreground application in the presence of a background task. Output of one port pin is toggled at a fixed interval in foreground. If an interrupt-based system is not present, the system receives all of the available MIPS. In this case, pulse width of the signal will be minimum. In the presence of a background task, the foreground task will not get 100% processing power, and as the foreground task is interrupted continuously, it will result in higher pulse width at test pin for the same delay. The difference in pulse width can be measured to estimate the MIPS consumption of the background task in the measurement period.

The MIPS consumption of the DTMF firmware can be measured in the following way.

- Define `MIPSTestPin` (port pin), FOSC and other system related parameters. Make sure the port pin used for the MIPS test is defined as o/p.
- Define the `DTMFDecBasicMIPSTest` parameter.
- Run the code and measure the period of square wave at MIPS test pin, referred to as "Ref width".
- Undefine the `DTMFDecBasicMIPSTest` parameter (comment the definition). Then, define the system in such a way that the background task works on a particular condition during the measurement period (e.g., for consumption at maximum throughput, provide continuous data during the measurement period, typically 10 characters/sec).
- Run the code and measure the period of the signal at the MIPS test pin (referred to as "Sample Width").
- The MIPS requirement for the above operating condition can be found using the following equation:
  MIPS Usage = Tot MIPS * ((Sample Width – Ref Width)/Sample Width), where:
  Tot MIPS = the total MIPS available to the system as the oscillator frequency.
- The MIPS requirement for other operating conditions can also be measured using the above method. Some compile-time options are provided in firmware to generate measuring conditions.

APPENDIX D: SOURCE CODE

The complete source code, including the application and necessary support files, are available under a no-cost license agreement. It is available for download as a single archive file from the Microchip corporate web site, at:

www.microchip.com

After downloading the archive, always check the file "version.log" for the current revision level and a history of changes to the software.

MICROCHIP®

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Alpharetta, GA
Tel: 770-640-0034
Fax: 770-640-0307

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

San Jose
Mountain View, CA
Tel: 650-215-1444
Fax: 650-961-0286

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8676-6200
Fax: 86-28-8676-6599

China - Fuzhou
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Shunde
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

China - Qingdao
Tel: 86-532-502-7355
Fax: 86-532-502-7205

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-2229-0061
Fax: 91-80-2229-0062

India - New Delhi
Tel: 91-11-5160-8631
Fax: 91-11-5160-8632

Japan - Kanagawa
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Penang
Tel:011-604-646-8870
Fax:011-604-646-5086

Philippines - Manila
Tel: 011-632-634-9065
Fax: 011-632-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Taiwan - Hsinchu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

EUROPE

Austria - Weis
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark - Ballerup
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Massy
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Ismaning
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

England - Berkshire
Tel: 44-118-921-5869
Fax: 44-118-921-5820

04/20/05