Application Note

DTMF Signal Detection Using Z8 Encore! XP F64xx Series MCUs

AN033501-1011

Abstract

This application note demonstrates Dual-Tone Multi-Frequency (DTMF) signal detection using Zilog's Z8F64xx Series microcontrollers. The firmware for these Z8F64xx Series microcontrollers is based on the Goertzel Algorithm; three (3) testing formats are used to ensure that this firmware is working in compliance with the International Telecommunication Union (ITU) Q.24 Recommendation. This ITU Recommendation states that the minimum valid DTMF tone duration is 40ms and that the minimum pause duration is also 40ms.

Note: The source code file associated with this application note, AN0335-SC01.zip, is available for download on zilog.com. This source code has been tested with version 5.0.0 of ZDSII for Z8 Encore! XP-powered MCUs. Subsequent releases of ZDSII may require you to modify the code supplied with this application note.

Additional source code for this application is contained in the *Z8 Encore! Applications Code Library*, available for download from the Application Sample Libraries page on the Zilog website.

Caution: This application note has not been tested for incoming calls to a standard telephone.

Features

The main features described in this application note are:

- Valid tone detection within the 40ms specification
- The 8-bit Z8F64xx Series MCU used as a digital signal processor
- Firmware based on the Goertzel Algorithm

Discussion

DTMF signaling is commonly used in telephone equipment. A DTMF signal is a combination of two (2) pure sinusoidal waves, each with differing frequencies. Based on the ITU Recommendation, there are eight (8) frequencies involved in DTMF signaling. These frequencies are grouped as Low Frequency and High Frequency, as shown in Table 1.

Table 1. DTMF Frequency Grouping

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

Note: The * symbol represents the Asterisk key on a phoneset; the # symbol represents the Pound key.

Included in the Table 1 are frequencies for each group and the digits represented by these frequencies. Each DTMF digit is a combination of two frequencies: one from the Low Frequency Group and one from the High Frequency Group. For example, DTMF digit 1 is a combination of 697Hz and 1209Hz.

The ITU also recommends that both DTMF signal duration and pause duration should be 40ms. Figure 1 illustrates a standard DTMF signal.

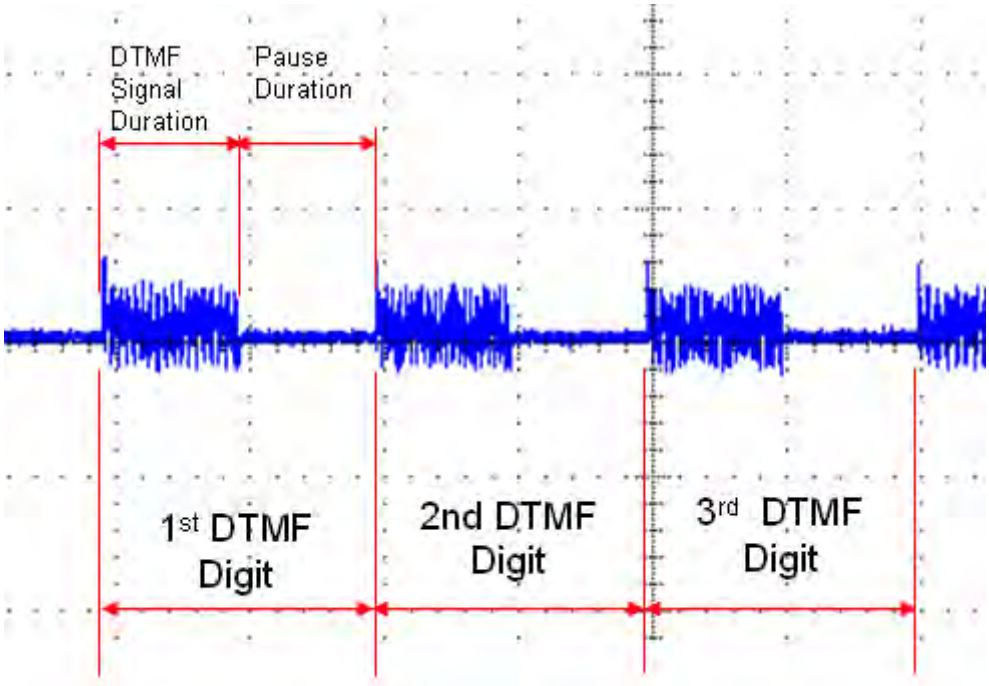


Figure 1. DTMF Signal

Theory of Operation

The Goertzel Algorithm is used in the implementation of DTMF detection. It is a Discrete Fourier Transform (DFT) method that can detect tones using less CPU power than the Fast Fourier Transform (FFT) method, and therefore is a faster method for detecting frequencies such as DTMF tones.

The Goertzel Algorithm is based on the DFT equation shown in Equation 1.

$$X(k) = \sum_{m=0}^{N-1} x(m)\, e^{-j\frac{2\pi}{N}km}$$

Equation 1

Finding its sequence from a linear filter process leads to Equation 2.

$$y_k(n) = \sum_{m=0}^{N-1} x(m)\, e^{j\frac{2\pi}{N}k(n-m)}$$

Equation 2

In Equation 2, the sample value n = N. Transforming Equation 2 into a linear difference equation yields Equation 3.

$$y_k(n) = e^{j\frac{2\pi}{N}k}\, y_k(n-1) + x(n)$$

Equation 3

In Equation 3, y(–1) = 0. Getting the z-domain transfer function of the Goertzel filter yields Equation 4.

$$H_1(z) = \frac{S(z)}{X(z)} = \frac{1 - e^{-j\frac{2\pi}{N}k} z^{-1}}{1 - 2\cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}}$$

Equation 4

Transforming Equation 4 into a difference equation leads to Equation 5.

$$s(n) = x(n) + 2\cos\left(\frac{2\pi k}{N}\right)s(n-1) - s(n-2)$$

Equation 5

Because there are only two frequencies present in a DTMF tone, we are no longer interested in acquiring the real and imaginary frequency components of a given signal. The focus of computations is now applied to the power parameters of the signal, which can be computed using the formula shown in Equation 6.

$$|X(k)|^2 = s(N-1)^2 + s(N-2)^2 - s(N-1)\,s(N-2)\cdot 2\cos\frac{2\pi}{N}k$$

Equation 6

Hardware Architecture

The high-pass filter, in Figure 2, is used to demonstrate detection of DTMF signals from telephone unit. Input to the high-pass filter is from the tip and ring of the telephone line; its output is connected to the ANA0 analog input of the Z8F64xx Series microcontroller.

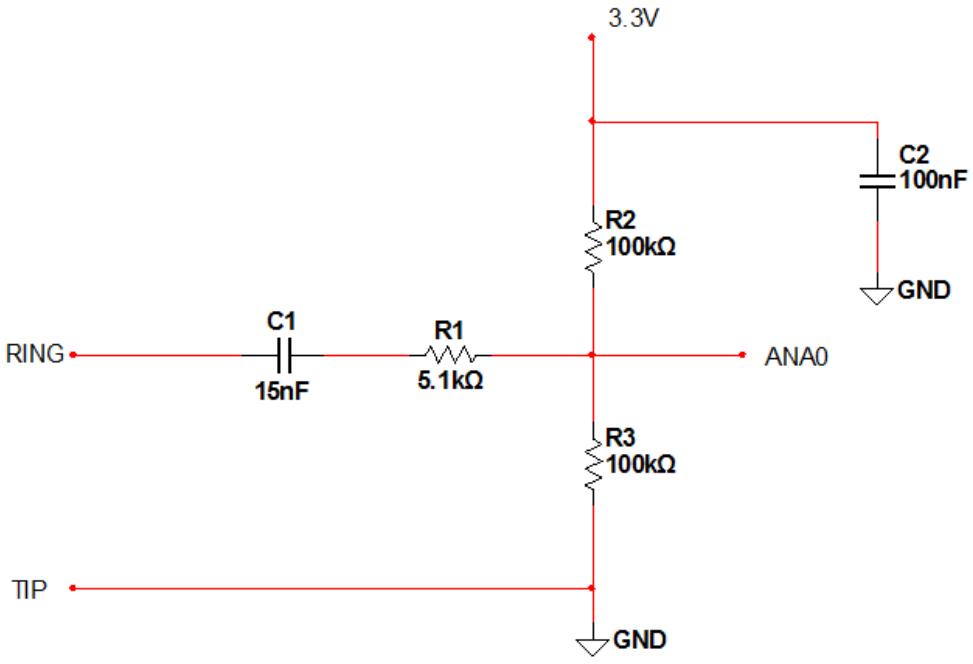


Figure 2. High-Pass Filter

The high-pass filter has a cut-off frequency of approximately 200 Hz; its primary function is to filter out the DC component from the telephone line and to suppress low-frequency line interference. The cut-off frequency is determined using the formula:

$$fc = \frac{1}{2\pi RC}$$

In the above equation, the variables are defined as:

$R = R_1 + (R_2 \parallel R_3)$

$C = C_1$

In telephony, the *tip* is the ground (positive) and the *ring* is the battery (negative). The voltage level upon ring, with respect to tip, is –6 Volts DC upon off-hook; the DTMF signal from the telephone line is superimposed on this negative DC voltage. Passing to the high-pass filter, this –6 V DC is filtered out. Because this –6 V is removed on the output going to ANA0, the DTMF signal subsequently *rides* on a 1.65 V DC bias voltage; this DC bias voltage is computed using the voltage divider formula:

$$\text{DC Bias} = 3.3\,\text{V}\left(\frac{R_3}{R_2 + R_3}\right)$$

In addition, capacitor C2 is used to filter out interference to and from the 3.3 V reference voltage.

Z8F64xx Development Board Setup

The ADC block of the Z8F64xx Series MCU is configured to operate using an external V_{REF}. Figure 3 shows how to set this up on the Z8F64xx Development Board by connecting Pin 1 of J2 and Pin 59 of JP2. These pins represent the external ADC V_{REF} of the MCU and one of the V_{DD} pins available on the development board, respectively. The terminal of ANA0, which is Pin 21 of J2, is shown to indicate the connection point of the filters described above.

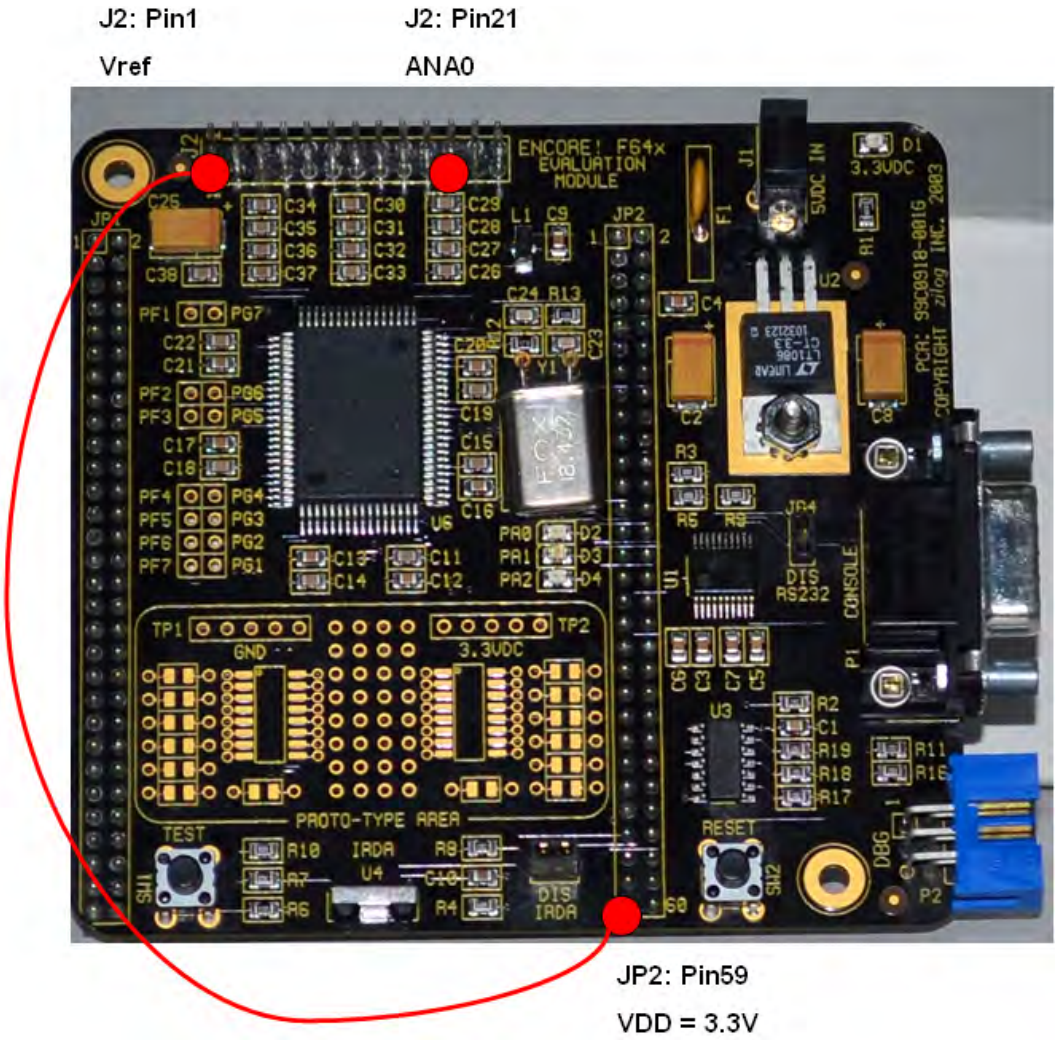


Figure 3. ADC External V_{REF} Connection and ANA0 Terminal

Software Implementation

This section presents the firmware developed for this application note. Topics are divided into: DTMF Detection and Display; ADC and Timer; and the Main routine.

DTMF Detection and Digit Display

The DTMF detection application focuses on `void Goertzel_Algorithm (void)`. With this function, there are 3 formulas used for processing samples from the ADC. These equations, which follow, are derived from Equation 5 on page 4.

$$Q_0 = \text{sample} + (\text{coefficient} \cdot Q_1) - Q_2 \quad \rightarrow \text{formula [1]}$$
$$Q_2 = Q_1 \quad \rightarrow \text{formula [2]}$$
$$Q_1 = Q_0 \quad \rightarrow \text{formula [3]}$$

These three equations are implemented for all samples; the coefficient is equal to:

$$2\cos\left(\frac{2\pi k}{N}\right)$$

In the equation above, the values are defined as:

N = number of samples

$k = N * (f_{TONE}/f_{SAMPLE\_RATE})$

To compute k, the values are defined as:

f_{TONE} = frequency of DTMF tones

f_{SAMPLE_RATE} = sampling rate frequency

After using the three formulas for a N number of samples, the relative magnitude squared based on Equation 6 on page 4, is directly computed as:

$$\text{magnitude}^2 = (Q_1^2) + (Q_2^2) - (Q_1 \cdot Q_2 \cdot \text{coefficient})$$

The process of executing the three formulas for all samples and obtaining the relative magnitude squared are performed for all 8 DTMF frequencies.

On the code, coefficients are precomputed and stored in the `iaCoefficient` array. To determine the coefficients, the defined number of samples used is 50, and the sampling rate is 3600 Hz. Table 2 presents the coefficients for all DTMF tone frequencies.

Table 2. Coefficients

Sampling Rate	3600	3600	3600	3600	3600	3600	3600	3600
N (# of samples)	50	50	50	50	50	50	50	50
DTMF Tone Freq	697	770	852	941	1209	1336	1477	1633
k	0.19361	0.21389	0.23667	0.26139	0.33583	0.37111	0.41028	0.453611
coefficient	0.69503	0.45123	0.16886	–0.14133	–1.0252	–1.3775	–1.68913	–1.91481

To make MCU processing faster, coefficients are converted into integers by multiplying them by 16. Table 3 lists these integer coefficients as used on the code.

Table 3. Integer Coefficients

coefficient	11	7	3	–2	–16	–22	–27	–31

Magnitudes for 8 DTMF tone frequencies are stored in the `iaEnergy_Magnitude` array. Elements 0 to 3 of this array hold the magnitudes for the tested low-frequency group; elements 4 to 7 correspond to the high-frequency group. This array is used by `void DTMF_DigitCodeID (void)` to identify the highest magnitude on the low-frequency group and the highest magnitude on the high-frequency group. This function is performed by comparing all of the magnitudes on each group. The resulting array index of the highest magnitude for the low-frequency group and the high-frequency group is saved in `ucLowFreq_HighestMagIdx` and `ucHighFreq_HighestMagIdx`, respectively. These indices are combined in `ucDTMF_DigitCode` to form the DTMF Digit Code shown in Table 4. `ucDTMF_DigitCode` is used by `void DTMF_DigitDisplay (void)` to identify the code and display the appropriate DTMF Digit on the host PC's HyperTerminal window.

Table 4. DTMF Digit Codes

DTMF Digit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	*** or E**	**# or F**
DTMF Digit Code (hex)	28	11	21	41	12	22	42	14	24	44	81	82	84	88	18	48

The `void Goertzel_Algorithm (void)` statement obtains N number of samples before the mathematical operations are computed; the storing of samples to the `ucSample` array occurs in intervals of 1/3600 sec. These samples from the ADC are scaled down with a factor defined by the `SCALE_DOWN` constant. Aside from these ADC samples, terms with coefficients are also scaled down by a factor of `SCALE_DOWN_COEFF`. Because the variables used are 16-bit integers, the scaling down of the ADC samples and the mathematical terms with coefficients must be performed to avoid overflow.

ADC and Timer

Upon `void ADC_Init(void)`, the ADC is configured to generate an interrupt after conversion is completed. It is set to run in Continuous Mode, with an external V_{REF} and with

ANA0 as the analog input for the DTMF signal. Upon every ADC interrupt, `void interrupt isrADC (void)` is executed to obtain a sample from the ADC Data registers, extract the 10 bit ADC data, and store this data to `uiADC_SampleData`.

Timer0 is set to generate an interrupt every 1/3600 sec. Upon every Timer0 interrupt, `void interrupt isrTimer0 (void)` is executed to set the `ucTimeToGetSample`. In turn, `void Goertzel_Algorithm (void)` is informed that 1/3600 sec has expired, and the ADC sample data can be stored to `ucSample` array.

Main

The main routine calls the functions to initialize the ADC, Timer0 and UART. After all initializations are performed, the main routine constantly checks for the presence of valid DTMF signal amplitudes through `void Debounce_DigitStart (void)`. If a valid DTMF signal amplitude is present, `void Goertzel_Algorithm (void)` is executed. After the detected signal is processed and identified, a DTMF digit is displayed on the host PC's HyperTerminal screen. Upon this display, `void main (void)` continues to execute `void Debounce_DigitEnd (void)` to check the analog input until the magnitude of the currently-detected DTMF signal is relatively low.

`void Debounce_DigitStart (void)` is used to check whether a valid DTMF magnitude is present. If the ADC sample data is below the threshold defined by `THRES_AMPLITUDE`, it means there is no DTMF signal on the line. For the testing setups (refer to the Testing/Demonstrating the Application section on page 10), these thresholds are listed Table 5.

Table 5. Threshold Amplitude

Setup Number	THRES_AMPLITUDE	ANA0 V_{IN} at Idle	ADC Value
1	100	180mV	56
2	600	1.46V	452
3	900	1.78V	551

`THRES_AMPLITUDE` depends on the ADC value when the line is idle or when the line has no DTMF signal. For example, the value of `THRES_AMPLITUDE`, as used in Setup 1, is 100 because the ADC value, on idle, is 56. If this idle voltage is different from the idle voltage value listed in Table 5, use the formula below to compute its ADC value as the basis for `THRES_AMPLITUDE`.

$$\text{ADC Value} = 1023 \cdot \frac{V_{in}}{V_{Ref}}$$

`void Debounce_DigitEnd (void)` is used to check the line until it becomes idle again. Part of this function is `void Goertzel_Algorithm (void)`, in which all mag-

nitudes are added. If the total magnitude is above THRES_ENERGY_MAG, the currently-detected DTMF signal is still present on the analog input. Otherwise, the line is idle again and the code is ready for the next detection process.

Please refer to Appendix A. Flowcharts on page 18 to review the DTMF Decoder algorithms.

Testing/Demonstrating the Application

To demonstrate this DTMF detection application, three different setups are used. These three setups are:

Setup 1. The AN0248-SC01 source code as the DTMF generator.

Setup 2. Two telephone units powered together and used as a DTMF generator.

Setup 3. A live telephone line and a telephone unit as a DTMF generator.

Setup 1 demonstrates tone detection under a noise-free environment. In this setup, the DTMF generator described in AN0248 is used. DTMF tone generated by DevBoard1, as shown in Figure 4, should be correctly identified by DevBoard2 and displayed on Computer 2.

Setup 2 and Setup 3 demonstrate tone detection within a noisy environment, which is present in the actual telephone line. For these setups, pressed key on the telephone unit will be detected by the development board and this will be displayed on PC's HyperTerminal.

Equipment Used

Here is the list of equipment used in the development and testing of this application note:

- Two (2) Z8 Encore! XP Z8F64xx Series Development Kits
- 18V power supply
- Telephone units
- Oscilloscope
- Personal computer

Setup 1

Two personal computers (Computer 1 and Computer 2) and two Z8F64xx Series development kits are used in this setup. The interface between each kit’s development board and its host PC is an RS-232 cable (see Figure 4). The PC1 and PC7 terminals of DevBoard1 are connected to the input of the low-pass filter. The output of this low-pass filter is connected to the ANA0 analog input of DevBoard2. Refer to Figure 3 on page 6 for the DevBoard2 setup.

Note: Please refer to the Zilog Application Note titled Generating DTMF Tones Using Z8 Encore! MCUs (AN0248) for details about the low-pass filter design.

DevBoard1 includes the firmware referenced in the Generating DTMF Tones Using Z8 Encore! MCUs Application Note (AN0248) document. This firmware is contained in the *Z8 Encore! Applications Code Library,* which is available for download from the Application Sample Libraries page on the Zilog website. Upon extracting the contents of the library, you'll find the DTMF Generator in the following filepath on your PC:

```
C:\Program Files\ZiLOG\Applications_Library\Z8 Encore! Applications Library 1.2.0\Z8 Encore! DTMF Generator
```

The companion source code to this application note, AN0335-SC01.zip, should be loaded to DevBoard2.

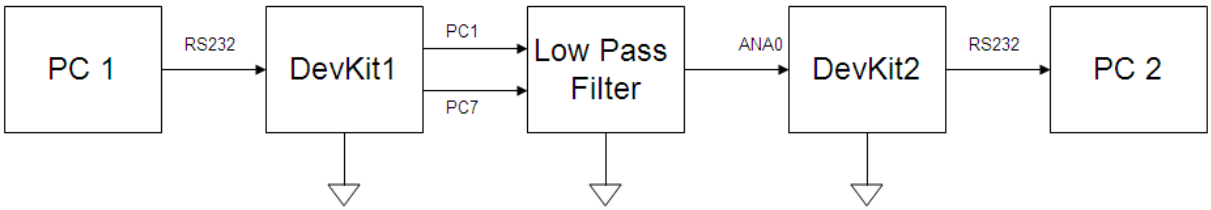


Figure 4. AN0248 and AN0335 Connections

For the display, configure HyperTerminal on both computers to the port settings shown in Figure 5.

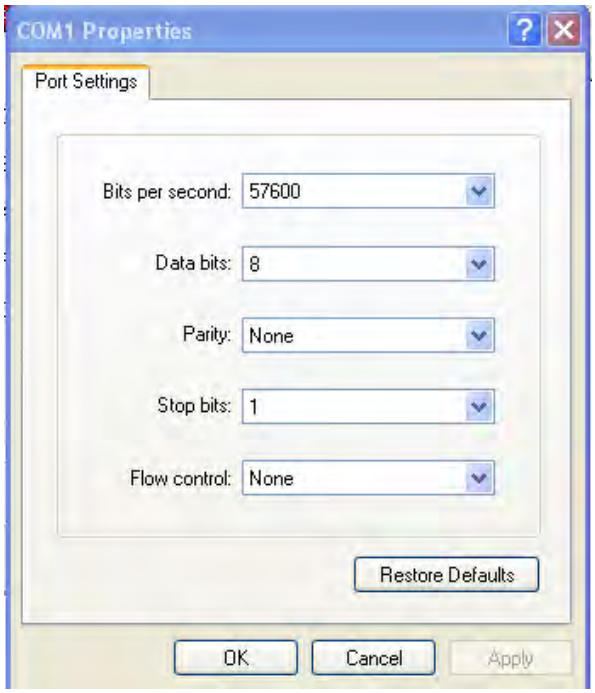


Figure 5. HyperTerminal Port Settings

In this setup, there are two types of demonstration modes that can be tested: Sequential Mode and Random Mode.

Sequential Mode Procedure

Observe the following procedure to demonstrate Sequential Mode.

1. Download the DTMF Generator code to DevBoard1 (see the beginning of this Setup 1 section to obtain this code).
2. Open the AN0335-SC01 source code in ZDSII v5.0 for Z8 Encore!.
3. In the `main.c` file, comment out the `#define THRES_AMPLITUDE` statement, as follows:

```
//#define THRES_AMPLITUDE 900    // DTMF tone amplitude threshold
                                 // value (live telephone line)
//#define THRES_AMPLITUDE 600    // DTMF tone amplitude threshold
                                 // value (telephone to telephone
                                 // setup)
#define THRES_AMPLITUDE 100      // DTMF tone amplitude threshold
                                 // value (AN0248 setup)
```

4. Compile and download the code to DevBoard2.
5. Configure HyperTerminal on both computers with the settings shown in Figure 5.
6. Reset both development boards.
7. Figure 6 shows the resulting HyperTerminal display on Computer 1 upon reset.

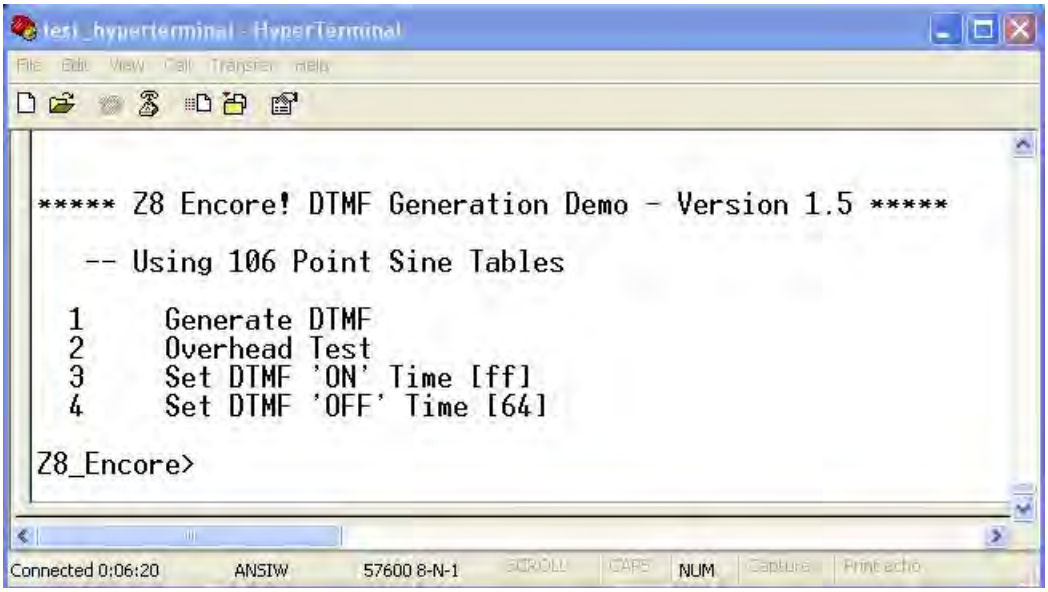


Figure 6. Initial HyperTerminal Display

8. On Computer 1, set the DTMF ON time by entering a 3, then enter the DTMF ON time in milliseconds; the required format for this entry is the hexadecimal format. Figure 7 provides an example for setting the DTMF ON time to 40 ms.

```
Z8_Encore> 3
Enter ON time in hex milli seconds (01 - FF mS): 28
```

Figure 7. The DTMF ON Time Setup in HyperTerminal

9. On Computer 1, set the DTMF OFF time by entering a 4, then entering the DTMF OFF time in milliseconds. Once again, the required format for this entry is the hexadecimal format. Figure 8 shows how to set the DTMF OFF time to 40 ms.

```
Z8_Encore> 4
Enter OFF time in hex milli seconds (01 - FF mS): 28
```

Figure 8. The DTMF OFF Time Setup in HyperTerminal

10. After setting up the DTMF ON and OFF times, the HyperTerminal program on Computer 1 displays the updated DTMF ON and OFF times; see Figure 9.

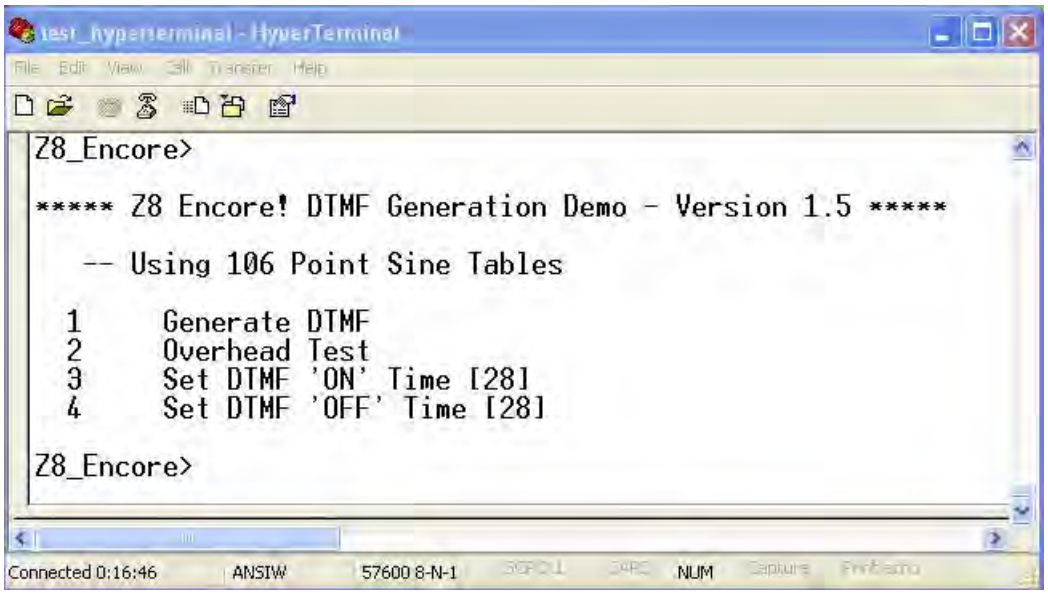


Figure 9. 40 ms DTMF ON and OFF Times

11. With the screen shown in Figure 9 displayed on Computer 1, enter a 1 on your keyboard and press the Enter key. HyperTerminal displays a result showing the 0, 1, 2, 3,

4, 5, 6, 7, 8, 9, A, B, C, D, E & F characters to indicate that they are being transmitted as DTMF tones.

12. DevBoard2 will detect the DTMF tones generated by DevBoard1, and the corresponding DTMF digits will be displayed on the HyperTerminal screen on Computer 2; see Figure 10.

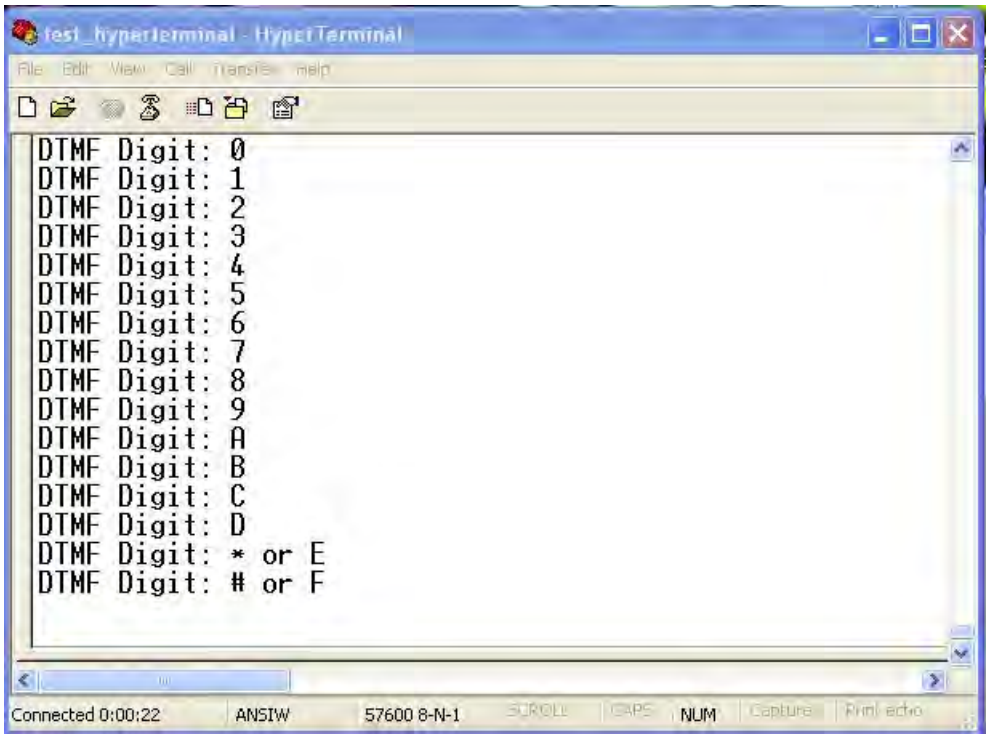


Figure 10. The DTMF Digit Display on Computer 2

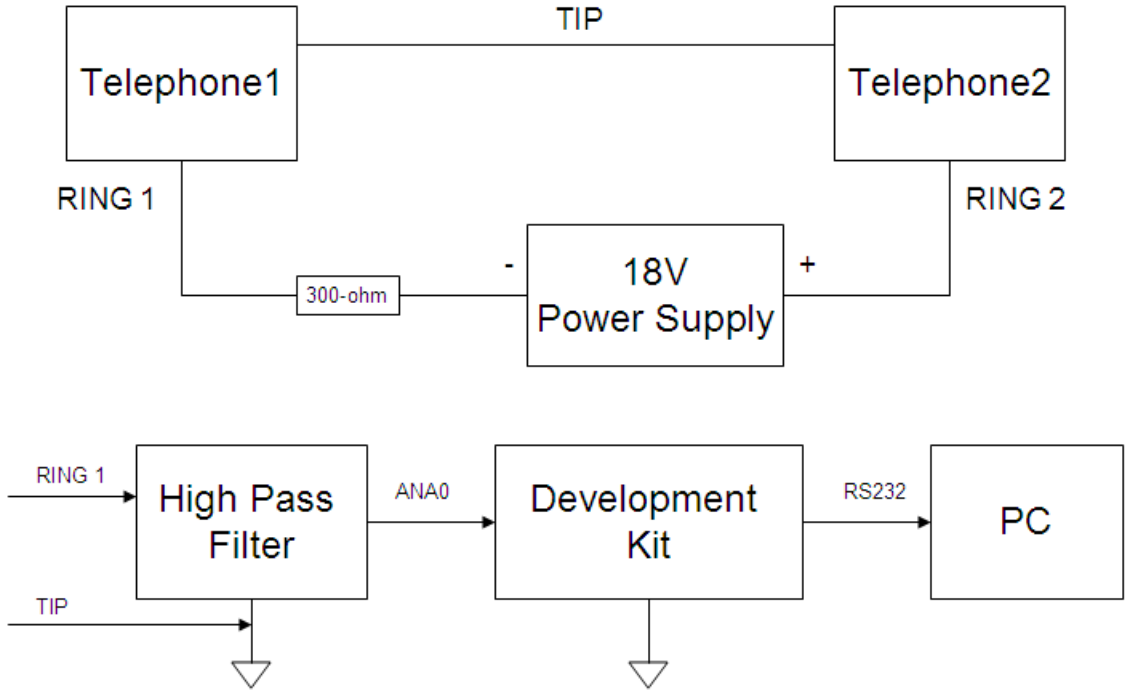
13. Repeat Steps 8 through 12 with different DTMF ON and OFF time values. The range of values that will provide valid DTMF detection ranges from 33ms (21 HEX) up to 255ms (FF HEX).

Random Mode Procedure

Follow the Sequential Mode Procedure up through Step 9. However, upon reaching Step 10, enter any combination of characters in the range 0–F and press the ENTER key. These characters should be detected by DevBoard2 and display in HyperTerminal on Computer 2.

Setup 2

In this setup, two telephone units are involved. These phones are both powered by an 18V power supply, as shown in Figure 11. Tip and Ring1 are connected on the input side of the high-pass filter. The high-pass filter output is connected to the input of the development board which is interfaced to a host PC via an RS-232 cable. Refer to Figures 2 and 3 (see page 4) for the connection details.



Note: 18V Power Supply is not connected to the common ground.

Figure 11. Two Telephone Units Functioning as a DTMF Detector

Procedure

Observe the following procedure to demonstrate Setup 2.

1. Open the AN0335-SC01 source code in ZDSII v5.0 for Z8 Encore!.
2. In the `main.c` file, comment out the `#define THRES_AMPLITUDE` statement specified for telephone-to-telephone setup, as follows:

```
//#define THRES_AMPLITUDE 900    // DTMF tone amplitude
                                 // threshold value (live telephone
                                 // line)
#define THRES_AMPLITUDE 600      // DTMF tone amplitude threshold
                                 // value (telephone to telephone
                                 // setup)
//#define THRES_AMPLITUDE 100    // DTMF tone amplitude threshold
                                 // value (AN0248 setup)
```

3. Compile and download the code to the development board.
4. Perform all connections as illustrated in Figure 11.

5. Configure HyperTerminal on the PC with the port settings shown in Figure 5 on page 11.
6. Ensure that both telephone units are off-hook.
7. Reset the development board.
8. Press any key on either of the two telephone units; the corresponding DTMF digit will be displayed in the HyperTerminal window.

Setup 3

In this setup, a live telephone unit is involved. Input side of the high-pass filter is connected to the tip and ring of the telephone line as shown in Figure 12. Output of the high-pass filter is connected to the ANA0 analog input of the development board which is interfaced to PC via RS232 cable. Refer to Figures 2 and 3 for the appropriate connections.

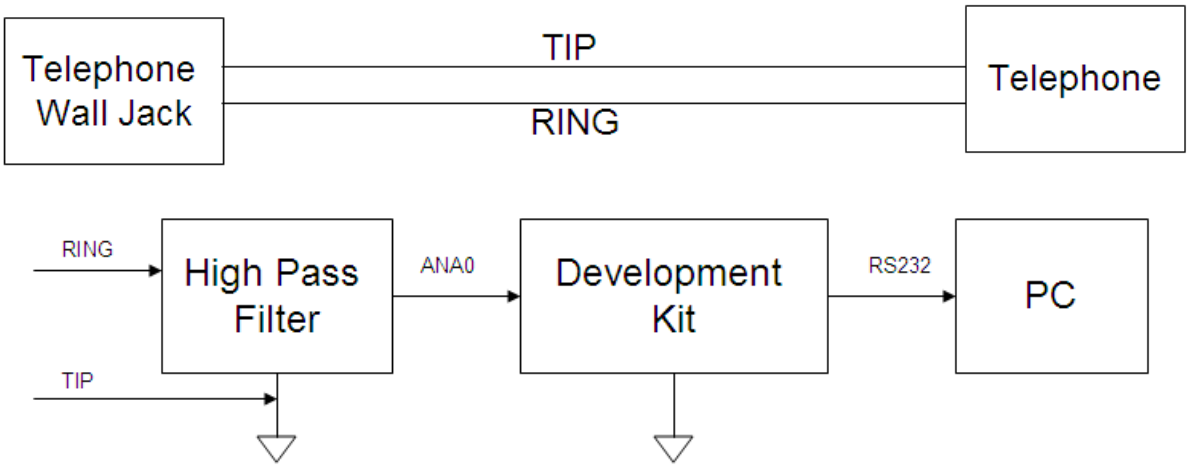


Figure 12. A Live Telephone as a DTMF Detector

Procedure

Observe the following procedure to demonstrate Setup 3.

1. Open the AN0335-SC01 source code in ZDSII v5.0 for Z8 Encore!.
2. In the `main.c` file, comment out the `#define THRES_AMPLITUDE` line that specifies the live telephone line, as follows:

```
#define THRES_AMPLITUDE 900     // DTMF tone amplitude
                                // threshold value (live telephone
                                // line)
//#define THRES_AMPLITUDE 600   // DTMF tone amplitude threshold
                                // value (telephone to telephone
                                // setup)
//#define THRES_AMPLITUDE 100   // DTMF tone amplitude threshold
```

```
                                        // value (AN0248 setup)
```

3. Compile and download the code to the development board.
4. Perform all connections as illustrated in Figure 12.
5. Configure HyperTerminal on the PC with the settings shown in Figure 5 on page 11.
6. Ensure that the telephone unit is off-hook.
7. Reset the development board.
8. Press any key on the telephone unit; the corresponding DTMF digit will be displayed in the HyperTerminal window.

Results

The results from the three testing setups indicate that the code developed for this application note can detect a minimum 40ms duration of the DTMF tone and a 40ms pause duration. It can even detect a DTMF tone duration of 33ms and a pause duration of 33ms.

Summary

This application note implements the Goertzel algorithm for DTMF tone detection using Zilog's Z8F64xx Series Microcontrollers. The application passes three testing methods to generate and detect DTMF tones. The firmware described herein can be used in applications that involve DTMF detection, such as caller ID and remote switching for home automation using a phone line.

References

The following documents describe functional specifications and/or otherwise support this application note. With the exception of the first two documents, each is available for download from the Zilog website.

- Digital Signal Processing (4th edition) by John G. Proakis and Dimitris K. Manolakis
- EE times Design Article: The Goertzel Algorithm by Kevin Banks (http://www.eetimes.com/design/embedded/4024443/The-Goertzel-Algorithm)
- Generating DTMF Tones Using Z8 Encore! MCU Application Note (AN0248)
- Z8 Encore! XP Z8F64xx Series Product Specification (PS0199)
- Z8 Encore! XP Z8F64xx Series Development Kit User Manual (UM0151)

Appendix A. Flowcharts

Figures 13 through 16 illustrate the flow of the DTMF decoder code described in the the Software Implementation section on page 6.

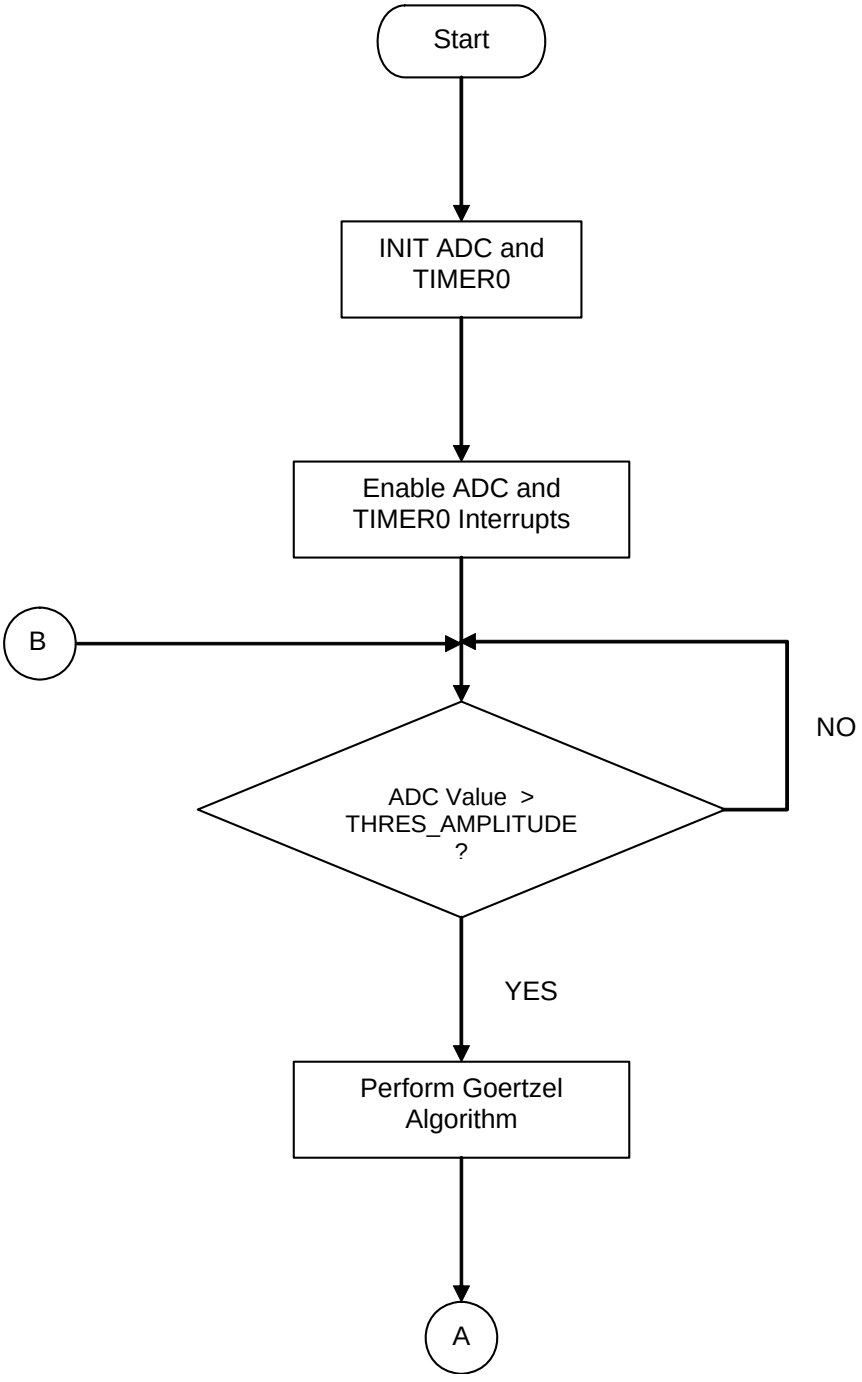


Figure 13. DTMF Decoder Flowchart, #1 of 4

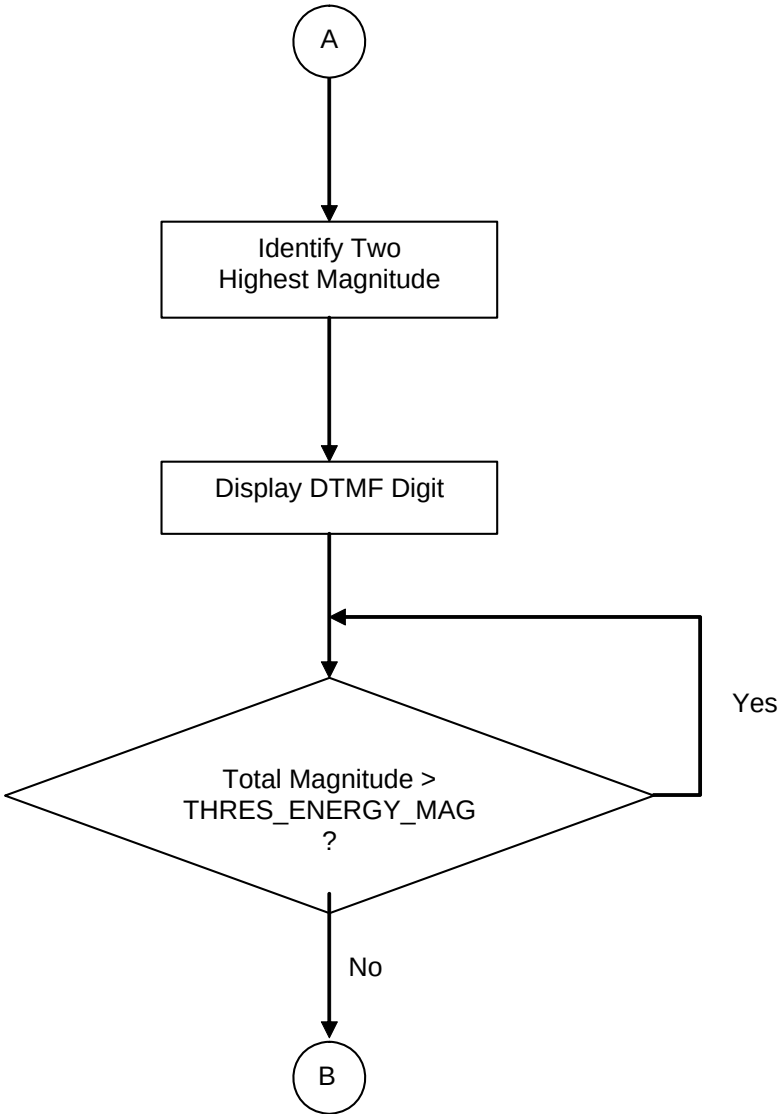


Figure 14. DTMF Decoder Flowchart, #2 of 4

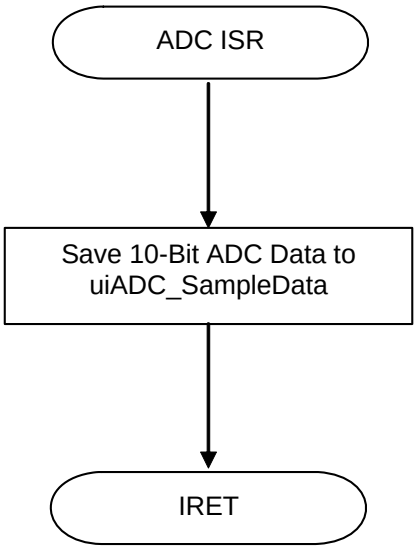


Figure 15. DTMF Decoder Flowchart, #3 of 4

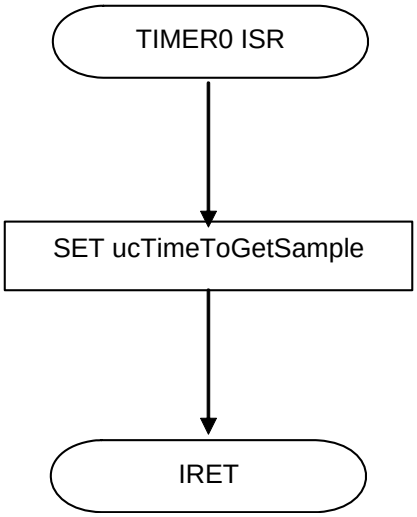


Figure 16. DTMF Decoder Flowchart, #4 of 4

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

 Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.