AN2384
Rev. 1, 12/2004

Generic Tone Detection Using Teager-Kaiser Energy Operators on the StarCore SC140 Core

By Valentin Emiya, Lúcio F.C. Pessoa, Delphine Vallot, and David Melles

This application note presents the use of the Teager-Kaiser (TK) energy operator [1] for generic tone detection. The TK operator can be used to build fast and efficient multi-frequency tone detectors with a high degree of accuracy and low cost, making it ideal for infrastructure applications such as media gateways. The generic tone detector is both versatile and flexible. The Freescale theoretical study is backed up by Matlab simulations and real-time implementations on Freescale DSPs based on the StarCore™ SC140 core. This study demonstrates that the TK operator can detect tones composed of many frequencies. The implementation discussed in this application note runs on the MSC8101 device, which is the first member of the StarCore family of digital signal processors (DSPs). The MSC8101 device uses the four ALU SC140 core, 512 KB of internal SRAM, and the popular communications processor module (CPM) of the Freescale Power QUICC II™ (MPC8260) device.

CONTENTS

1 Tone Detection Basics

The design of the generic tone detector discussed in this document is based on the fact that any single-frequency tone $x(n) = A\ cos(\Omega n + \phi)$ is mapped to a constant value via the modified TK energy operator [1], as follows:

$$\Psi_k\ (x(n)) = x^2\ (n-k) - x(n)x(n-2k) = A^2\ sin^2\ (k\Omega)$$

The modified TK energy operator is a special case of a Volterra filter, which depends both on the magnitude A and the normalized frequency Ω of the tone; $\Omega = 2\pi\ f\ /\ f_s$, where f is the tone frequency and f_s is the sampling frequency. Observe that $\Psi_k\ (x(n))$ does not depend on the phase ϕ. The parameter k defines the underlying sub-rate processing ($k = 1$ in the original definition of the TK algorithm); notice that the effect of applying $\Psi_k\ (.)$ at a sampling rate f_s is equivalent to applying $\Psi_1\ (.)$ at a sampling rate $f_s\ /\ k$. Depending on the range of frequencies of interest, sub-rate processing is a preferred approach because it reduces computational requirements.

Depending on the power of the signal (that is, the magnitude A of the tone), the energy operator generates different levels for the same normalized frequency Ω Therefore, to estimate Ω, you must efficiently remove this magnitude dependency by, for example, processing $x(n)$ through an FIR filter of the form $B(z) = (z^{-l} + z^{-m})\ /\ 2$ and then applying the energy operator to the result. Once the dependency is removed, Ω is indirectly estimated by computing the following ratio (ρ_Ω) of energy operators[2]:

$$\rho_\Omega = \frac{\Psi_k\left(\frac{1}{2}\ (x(n-l) + x(n-m))\right)}{\Psi_k\ (x(n))} = cos^2\left(\left(\frac{l-m}{2}\right)\Omega\right)$$

This expression derives from the definition of energy operators and the following trigonometric identity:

$$\frac{1}{2}\left[cos\ (\alpha+\beta)\ + cos\ (\alpha+\gamma)\ \right] = cos\left(\frac{\beta-\gamma}{2}\right) cos\left(\alpha + \frac{\beta+\gamma}{2}\right)$$

Therefore, selecting $k = (l - m)\ /\ 2$, the tone magnitude A is estimated by computing the following ratio (ρ_A) [2]. Notice that $\rho_A\ /\ 2$ can be used to estimate the standard average power of $x(n)$:

$$\rho_A = \frac{\Psi_k\ (x(n))}{1 - \rho_\Omega} = A^2$$

An important practical issue is how to compute the two divisions that estimate ρ_A and ρ_Ω efficiently. A polynomial approximation is employed according to the following approach for computing a ratio between a numerator N and a denominator D:

$$q = \frac{N}{D} = \frac{2N}{2D} = \frac{2N}{2D^b 2^{-b}} = N\left(\frac{1}{2D2^b}\right) 2^{b+1} = Np(D')2^{b+1}$$

$D' = D2^b$ is the denominator normalized to the range between 1/2 and 1, b is the corresponding number of leading bits of the normalization (efficiently computed on most DSPs), and $p(.)$ is a polynomial approximation of the following function:

$$f(x) = \frac{1}{2x}, \frac{1}{2} \leq x \leq 1.$$

A third order polynomial is selected for this task, so that

$$q = N(a_3 + D'(a_2 + D'(a_1 + D'a_0))) 2^{b+4}$$

The coefficients are normalized to the range between –1 and +1, thus resulting in an additional three shifts (that is, 2^{b+1} changes to 2^{b+4}). The normalized coefficients are listed in the following table:

a_0	–0.22482299804688
a_1	0.66952514648438
a_2	–0.73565673828125
a_3	0.35321044921875

In practice, the *x(n)* signal is corrupted by noise. To overcome this practical constraint, a low pass filter (LPF) is used to smoothen the outputs from the ρ_A and ρ_Ω estimates. This noise reduction stage occurs efficiently through a single pole LPF of the following form:

$$A(z) = \frac{1-a}{1-az^{-1}}, \quad 0 < a < 1$$

If the detector is required to handle high noise levels, the outputs from the TK energy operators may also need to be filtered.

2 Multi-Component Tone Detection

When the *x(n)* signal is composed of more than one frequency, extending the preceding method is not so simple. In this case,

$$x(n) = \sum_{i=1}^{N} A_i \cos(n\Omega_i + \phi_i)$$

so that

$$\Psi_k(x(n)) = \sum_{i=1}^{N} A_i^2 \sin^2(k\Omega_i) + \varphi(n, \Omega_1, \ldots \Omega_N)$$

The function $\varphi(.)$ makes the energy operator time-varying, thus imposing additional difficulty in separating the *N* components. It can be shown that $\varphi(.)$ is given by

$$\varphi(n,\Omega_1, \ldots \Omega_N) = 2 \sum_{t<s} A_t A_s \{\sin^2(k(\Omega_t - \Omega_s)/2) \cos((\Omega_t + \Omega_s)(n-k) + \phi_t + \phi_s) + \sin^2(k(\Omega_t + \Omega_s)/2) \cos((\Omega_t - \Omega_s)(n-k) + \phi_t + \phi_s)\}$$

An important case occurs when $\Omega_i = \Omega$ and $A_i = g_i A$ for all components, which then makes $\varphi(.)$ independent of n. Now the energy operator of $x(n)$ becomes

$$\Psi_k(x(n)) = \frac{1}{g_c}\left(\sum_{i=1}^{N} g_i^2 + 2\sum_{t<s} g_t g_s \cos(\phi_t - \phi_s)\right)\Psi_k(x_c(n)),\ c = 1, \ldots, N$$

Therefore, if multiple components have the same frequency but different gain and delay with respect to a given reference component c, the energy operator of the $x(n)$ signal is simply a scaled version of the energy operator of a reference component. This result implies that the energy operator can handle multiple echoes of a single-frequency tone.

$$x_c(n) = A_c \cos(n\Omega_c + \phi_c),\ c = 1, \ldots, N$$

The preferred approach to detecting multiple components using energy operators is first to filter the $x(n)$ signal with N independent filters so that every component $x_c(n)$ can be extracted efficiently. Once the N components are extracted, their magnitude and frequency estimates (ρ_{Ai} and $\rho_{\Omega i}$) are computed. This problem is viewed as a special case of a multi-component AM-FM demodulation [3], in which the magnitude (AM) and frequency (FM) of every component is estimated, so that a tone is detected if these estimates are close enough to pre-defined reference values. In general, the set of possible frequency combinations to be detected is given as follows:

$$F = \{(\Omega_1^{(m)}, \ldots, \Omega_N^{(m)}),\ m = 1, \ldots, M\},$$

This set then defines the following set of reference points for use by the detector:

$$R = \{(\rho_{\Omega 1}^{(m)}, \ldots, \rho_{\Omega N}^{(m)}),\ m = 1, \ldots, M\},$$

Comb filters can efficiently decompose the signal so that a given filtering path (c) removes all the undesirable components from the other paths ($i \neq c$). The preferred filter structure has the general form:

$$H_c^{(m)}(z) = \Gamma_c^{(m)} \prod_{i \neq c} \frac{1 - b_i^{(m)} z^{-1} + z^{-2}}{1 - r b_i^{(m)} z^{-1} + r^2 z^{-2}},\ c = 1, \ldots, N$$

Where:

$$b_i^{(m)} = 2\cos(\Omega_i^{(m)}),\ 0 < r < 1$$

The following value is chosen:

$$\Gamma_c^{(m)}$$

So that this expression applies:

$$|H_c^{(m)}(e^{j\Omega_c^{(m)}})| = 1$$

It can be shown that the following expression applies:

$$\Gamma_c^{(m)} = \sqrt{\prod_{i \neq c} \frac{\left(\frac{1-r^2}{2}\right) - r\,[r\cos(\Omega_c^{(m)}) - \cos(\Omega_i^{(m)})]\,[r\cos(\Omega_i^{(m)}) - \cos(\Omega_c^{(m)})]}{[\cos(\Omega_c^{(m)}) - \cos(\Omega_i^{(m)})]^2}}$$

For the two-component case ($N = 2$), it is important to notice the following:

$$\Gamma_1^{(m)} = \Gamma_2^{(m)} \; \forall m$$

The values of the following are pre-computed and stored in a look-up table; if magnitude estimations are not needed:

$$\rho_{Ac}^{(m)}$$

The gains may be ignored:

$$\Gamma_c^{(m)}$$

The following coefficients are pre-computed and stored in a look-up table.

$$b_c^{(m)}, \; c = 1, \ldots, N; \; m = 1, \ldots, M$$

To detect a given multi-component tone successfully, a self-tuning mechanism is required for searching the optimal set of frequencies that *minimizes variability* of the frequency detector outputs. This self-tuning mechanism is based on the fact that a single frequency tone generates a constant value after the TK energy operator processes it. The self-tuning mechanism works as follows:

1. The process starts with an initial guess $m = m_{old}$, which defines the coefficients of the comb filters, as follows:

$$H_c^{(m)}(z), \; c = 1, \ldots, N$$

2. When the next sample is processed, magnitude and frequency estimates are computed, and the closest pair of reference frequencies is selected, thus defining a new guess $m = m_{new}$. New filter coefficients are then defined. In this process, the parameter r of the comb filters may also be adjusted.
3. This self-tuning process repeats until the optimal symbol (tone) $m = m_{opt}$ is found. From that point on, the filter coefficients are fixed, resulting in frequency lock. If the signaling tone is removed, the frequency is unlocked until a new tone is detected. The bandwidth of the LPFs can be adjusted (that is, gear-shifted) by choosing different values of α before and after frequency lock.

Figure 1 illustrates the structure of the multi-component detector, where a detected tone m_{det} is reported by a decision logic unit.

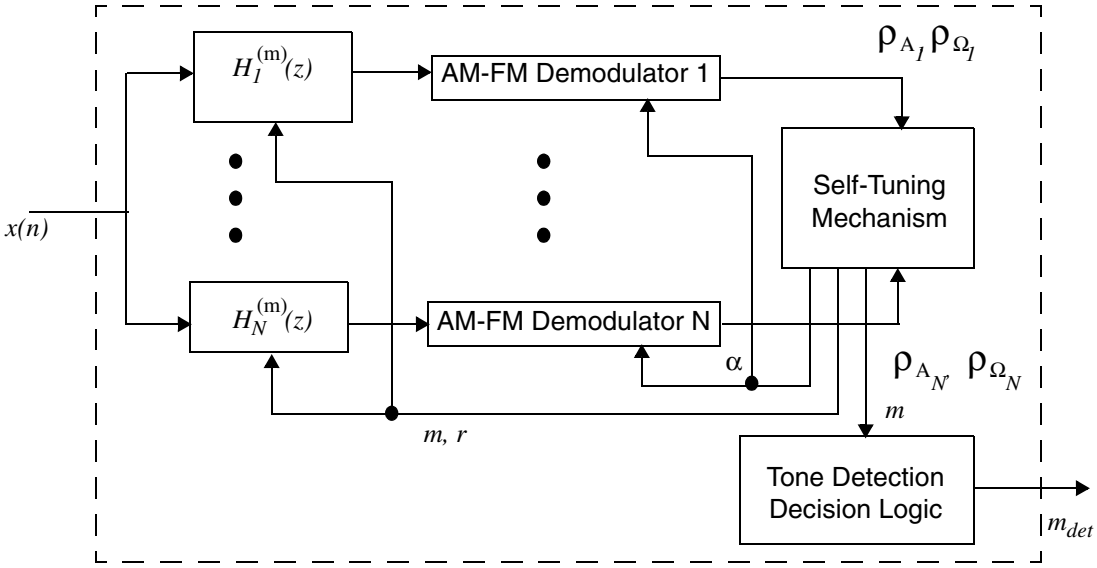

Figure 1. Structure of the Multi-Component Detector

Our design offers the following advantages over traditional detectors based on Fourier transforms:

- Easy to reuse, fine-tune, and maintain.
- Efficiently implemented on parallel architectures, such as the SC140 core.
- Requires a relatively small look-up table.
- Can be used under noisy conditions.
- Provides a low false-detection rate—for example, when a voice signal is incorrectly detected as a signaling tone.
- Very good time and frequency resolutions.

3 Two-Component Tone Detector for DTMF

Dual tone multi-frequency (DTMF) is a set of standard signaling tones adopted by telephone companies. A pressed key on telephone equipment generates a dual-frequency tone composed of a low frequency component and a high frequency component. The frequencies used for DTMF are not harmonically related, and it is difficult for a voice signal to imitate the DTMF tone. This choice minimizes the likelihood of spurious signals being accepted as valid DTMF tones. Frequency separation between consecutive tones inside each group of frequencies is typically 10 percent. DTMF signaling is a popular example of a multi-frequency tone. A DTMF key is defined by two frequencies, one for the low group (f_L) and one for the high group (f_H). The principle of the detection algorithm is to estimate the two frequencies that are present in the signal:

$$\hat{f}_L, \hat{f}_H$$

Next, find the closest reference point of a valid key (f_L, f_H). This identification process proceeds on a per-sample basis. **Figure 2** illustrates this concept; each cross represents the location of a reference key.

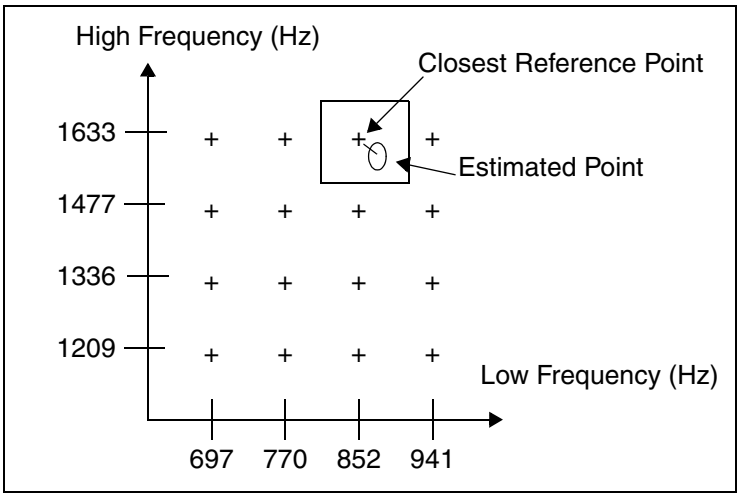

Figure 2. DTMF Digits Represented on a Bi-Dimensional Graph

Figure 3 and the following pseudo-code show the application of the TK energy operator within a DTMF detection context. The SC140 core and its parallel ALUs are ideally suited to this application and can take advantage of the inherent parallelism in this DTMF tone detector.

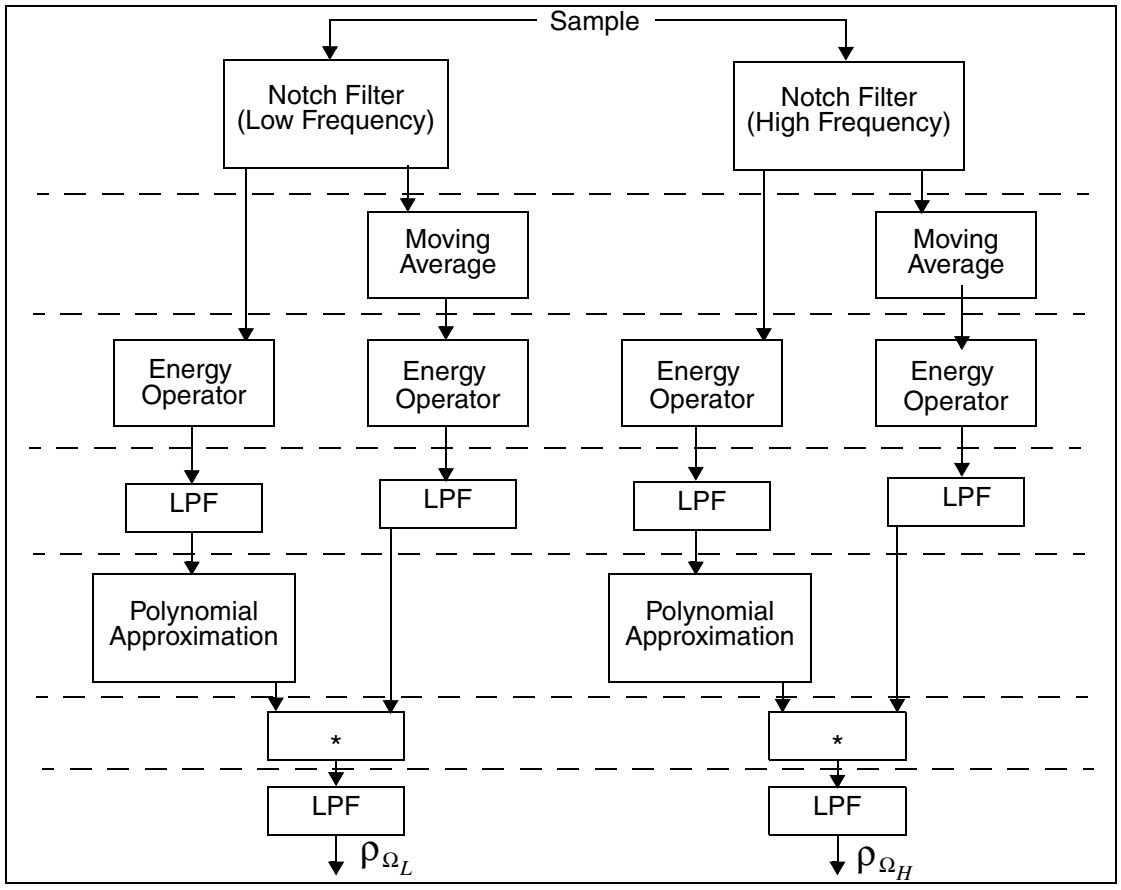

Figure 3. Four-Branch Parallelism of the DTMF Detector

The following pseudo-code details the proposed method:

Figure 3

```
Select m = m_old as the first guess and initialize all variables;
For every sample of the signal x(n), do:
     Compute the energy operator of x(n);
     Smoothen the result with an LPF of parameter α to obtain P_x;
     If the magnitude of P_x is larger than P_1
          For every component c, do in parallel:
             Filter x(n) with a comb filter with parameters c, r and generate x_c(n);
             Filter x_c(n) with a moving average of lag 2;
             Take the result of the moving average and compute its energy operator;
             Compute the energy operator of x_c(n);
             Smoothen the two energy operators with an LPF of parameter α;
             Use polynomial approximation to estimate ρ_Ωc^(m) and ρ_Ac^(m) ;
               Smoothen ρ_Ωc^(m) and ρ_Ac^(m) with an LPF of parameter α;
          End
          If the magnitude of P_x is larger than P_2
               Find the closest reference point to the current frequency estimates
               and define a new guess m = m_new;
               If m_new is equal to m_old
                    Increment detection counter;
                    If detection counter is large enough
                        choose α = α_1;
                    End
               Else
                    Clear detection counter and choose α = α_0;
               End
               m_old = m_new;
          End
          Apply decision logic and report a valid detection event on m_det;
     Else
          Set m = 0 and clear detection counter;
     End
End
```

4 DTMF Detector on StarCore

This section describes how the DTMF detector is implemented on the StarCore SC140 core. It highlights both the advantages of the SC140 core for this application and the mechanisms added to make it compliant with DTMF standards.

A telephone keypad is represented in **Figure 4** with the corresponding DTMF frequencies (in Hertz). The values in the top left corner are the actual digits, whereas those in the bottom right corner are the encoded digits within the detection algorithm. In this internal code, the 0 value corresponds to invalid DTMF tones, such as noise or voice signals.

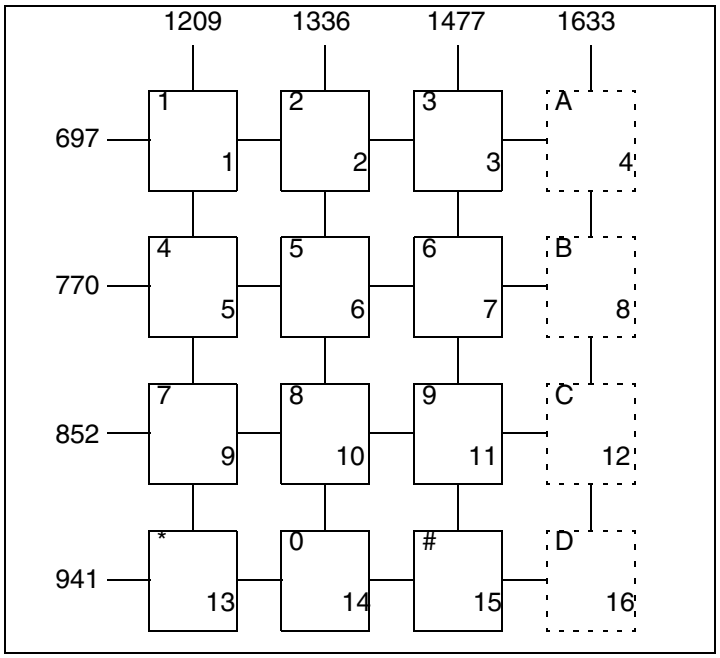

Figure 4. Telephone Keypad

4.1 ITU-T Recommendation Q.24

Table 1 shows the AT&T requirements of this recommendation [5].

Table 1. AT&T Requirements for ITU-T Recommendation

Parameters		Values
Signal frequencies	Low group	697, 770, 852, 941 Hz
	High group	1209, 1336, 1477, 1633 Hz
Frequency tolerance | Δ f |	Operation	≤1.5%
	Non-operation	≥ 3.5%
Power levels per frequency	Operation	0 to –25 dBm0
	Non-operation	Max. –55 dBm0
Power level difference (twist) between high and low frequencies		DTMF digits must be detected with twists between –8 dB and 4 dB
Signal reception timing	Signal duration:	
	Operation	Min. 40 ms
	Non-operation	Max. 23 ms
	Pause duration	Min. 40 ms
	Signal interruption	Max. 10 ms
	Signaling velocity	Min. 93 ms/key
Signal simulation by speech		For the codes 0–9, 1 false/3000 calls For the codes 0–9,*,# 1 false/2000 calls For the codes 0–9,*,#,A–D, 1 false/1500 calls
Interference by echoes		Should tolerate echoes delayed up to 20 ms and at least 10 dB down

The implementation of the TK algorithm for DTMF detection (see **Figure 5**) requires additional processing to pass Q.24 requirements. The algorithm is based upon the following principles:

- *Automatic gain control (AGC)*. A device added to improve dynamic range. It finds the peak value of the signal and amplifies the input samples by shifting variables (the gain varies between 1 and 2^8) before entering the sample processing section.
- *Sample processing section*. Explained in **Section 1**, *Tone Detection Basics*, so the corresponding block is not represented in detail in **Figure 5**. This figure emphasizes that the calculation is bypassed when signal power is too weak. ρ_{Ω_L} and ρ_{Ω_H} are the frequency detectors of the low and high frequencies, respectively, that must be compared with reference values.
- *Difference between a digit and a key*. An intermediate value is found after the energy detector and before the decision logic. This “digit” results from the analysis of a sample. In this document, “digits” are the outputs of the sample processing section, and “keys” are the outputs of the digit processing section. Thus, a digit is produced each time a sample is processed. A key is the result obtained by the analysis of a set of consecutive and coherent digits with specific conditions of duration and interruption that make it compliant with Q.24 norm. Every signal not interpreted as a key is a “non-valid” signal. A non-valid signal may be a high-energy sound with characteristics not Q.24 compliant (noise, too short signal, etc.). A “pause” is a non-valid signal that lasts at least 40 ms (Q.24 norm). Signals of very short or negligible duration are “interruptions.”

4.2 Adaptations to Recommendation Q.24

This section covers issues pertaining to frequency, including power level per frequency, tolerance, power level differences between frequencies, signal reception timing, and signal and pause duration.

4.2.1 Power Level Per Frequency

The ITU-T Q.24 standard recommends that a key with a power level per frequency between 0 dBm0 and –25dBm0 must be recognized. The power given in dBm0 is obtained in the following equation [6]:

- A-Law encoding: $Power(dBm0) = 10log_{10}(P_t) + 6.15$
- μ-Law encoding: $Power(dBm0) = 10log_{10}(P_t) + 6.18$

P_t is the average power of a linear PCM signal computed in fractional format (that is, with normalized values in the range [–1, 1]).

The AGC is calculated at the beginning of sample processing by finding the maximum sample value to be processed and then selecting the proper value for the gain. The amplitude modulation introduced by this process between consecutive sample blocks causes a transient period that is negligible when the sample block is large enough (about 40 samples, which correspond to 5 ms of data at a sampling rate of 8 KHz). Without the AGC, 16-bit rounded sample values limit the detector to a power level per frequency of about –18 dBm0; with the AGC, the power level per frequency can be smaller than –25 dBm0 (value recommended in the Q.24 norm).

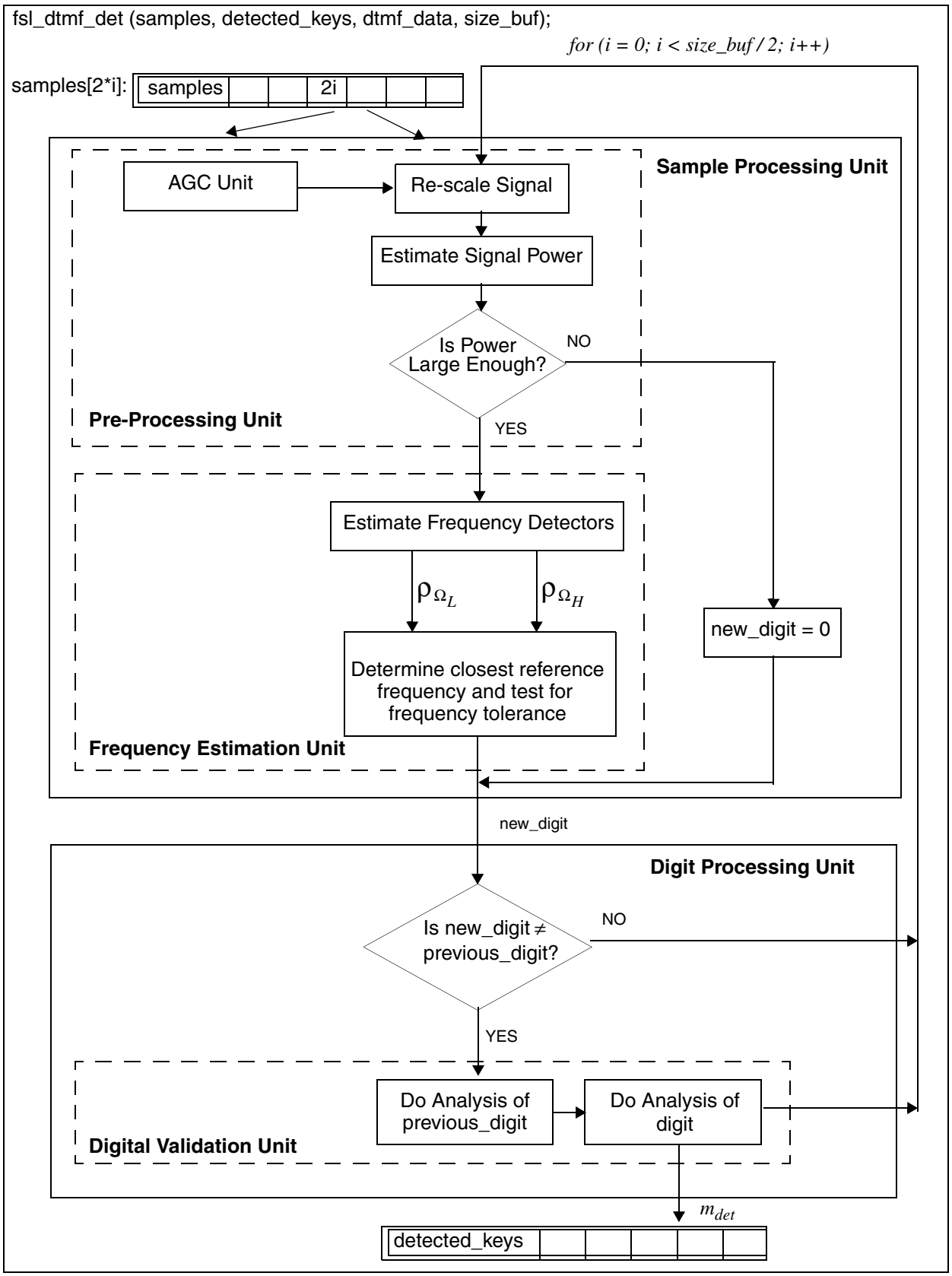

Figure 5. Structure of the TK Algorithm for DTMF Detection

4.2.2 Frequency Tolerance

According to the Q.24 standard, a frequency tone must be accepted if it deviates less than 1.5 percent from the nominal frequency, and it must be rejected if it deviates more than 3.5 percent. These constraints bound the area around the ideal values within which the frequency should be interpreted as valid, represented by a cross in **Figure 2**. **Table 2** shows the required acceptance and rejection bands for the various frequencies.

Table 2. Required Acceptance and Rejection Bands

Tone Frequency (Hz)	Acceptance Bandwidth (Hz) (< 1.5%)	Acceptance Bandwidth (Hz)	Rejection Bandwidth (Hz) (> 3.5%)	Rejection Bandwidth (Hz)
697	±10.5	[686.5;707.5]	±24.4	< 672.6; > 721.4
770	±11.6	[758.4;781.6]	±27.0	< 743; > 797
852	±12.8	[839.2;864.8]	±29.8	< 822.2; > 881.8
941	±14.1	[926.9;955.1]	±32.9	< 908.1; > 973.9
1209	±18.1	[1190.9;1227.1]	±42.3	< 1166.7; > 1251.3
1336	±20.0	[1316;1356]	±46.8	< 1289.2; > 1382.8
1477	±22.2	[1454.8;1499.2]	±51.7	< 1425.3; > 1528.7
1633	±24.5	[1608.5;1957.5]	±57.2	< 1575.8; > 1690.2

As described in **Section 1**, *Tone Detection Basics*, the frequencies are implicitly defined through $\rho_\Omega = cos^2(\Omega)$, where the sampling rate is 4 KHz (that is, half of the original sampling rate because every other sample from the 8 KHz input vector is processed). The reference values of ρ_Ω are tabulated in **Table 3**. Δ_{acc} represents the maximum distance between the tone frequency and the accepted one, and Δ_{rej} represents the minimum distance between the tone frequency and the rejected one.

Table 3. Reference Values of ρ_W

ρ_Ω	Acceptance Bandwidth	Δ_{acc}	Rejection Bandwidth	Δ_{rej}
0.20993041992188	0.2235;0.1967	0.0136	0.242;0.1796	0.0303
0.12493896484375	0.1372;0.1131	0.0123	0.1543;0.0983	0.0266
0.05307006835938	0.0625;0.0444	0.0094	0.076;0.0341	0.019
0.00857543945313	0.0131;0.005	0.0045	0.0207;0.0017	0.0069
0.10397338867188	0.0873;0.1219	0.0179	0.067;0.1476	0.037
0.25363159179688	0.2268;0.2814	0.0278	0.1926;0.32	0.061
0.46389770507813	0.4292;0.4987	0.0348	0.3837;0.545	0.0802
0.70288085937500	0.6672;0.7374	0.0357	0.6179;0.7813	0.0784

The decisive values of the potential keys must be taken into the interval $[\Delta_{acc}, \Delta_{rej}]$, which is reported in **Table 4**. A key is declared valid if its distance to the closest point for the low frequency and for the high frequency is smaller than their corresponding decisive values.

Table 4. Interval $[\Delta_{acc}, \Delta_{rej}]$

Tone Frequency (Hz)	Decisive Value	Decimal Value
697	0x327	0.024627685546875
770	0x2A9	0.020782470703125
852	0x22A	0.01690673828125
941	0xE0	0.0068359375
1209	0x400	0.03125
1336	0x600	0.046875
1477	0x800	0.0625
1633	0x900	0.0703125

4.2.3 Power Level Difference Between Frequencies

The ITU-T Q.24 standard recommends that the high group frequency power level may be up to 4 dB larger or 8 dB smaller than the low group frequency power level. These conditions can be represented as follows:

$$\begin{cases} P_H\,(dB) < P_L\,(dB) + 4dB \\ P_H\,(dB) > P_L\,(dB) - 8dB \end{cases} \longleftrightarrow \begin{cases} P_H < P_L \times 2.512 \\ P_H > P_L \times 0.158 \end{cases} \longleftrightarrow \begin{cases} \dfrac{P_L}{P_H} > 0.398 \\ \dfrac{P_H}{P_L} > 0.158 \end{cases}$$

or in a single inequality:

$$0.158 < \frac{P_H}{P_L} < 2.512$$

P_H is the high group frequency power level and P_L is the low group frequency power level. In other words, any DTMF key with power ratios (also referred to as twist) within that range must be detected.

In the algorithm based on energy operators, the envelope estimate of the signal is calculated for the low and for the high frequencies. They correspond to the variables *dlf2* and *dhf2*, respectively, in the code. The ratio *dhf 2/ dlf2*, if *dhf2* < *dlf2*, (*dlf2/dhf2*, if *dlf2* < *dhf2*) is compared to a proper threshold so that the Q.24 recommendation is respected. To take into account noise, limited precision, and polynomial approximation error, the actual thresholds used in the code are determined experimentally, yielding the following:

$$\begin{cases} \dfrac{dhf2}{dlf2} > 0.158477783203125 & (0x1449) \\ \dfrac{dlf2}{dhf2} > 0.398101806640625 & (0x32F5) \end{cases}$$

The plots shown in **Figure 6** illustrate some of the test results. Beneath the first graph, the two extracted curves are shown with a more precise scale to find those two bounds. However, the Q.24 standard does not specify a rule for twists outside the recommended range, which allows for different implementations. Q.24 specifies only that any DTMF signal with a twist inside that range must be detected. Therefore, the twist test is disabled in the optimized implementation of the DTMF detection.

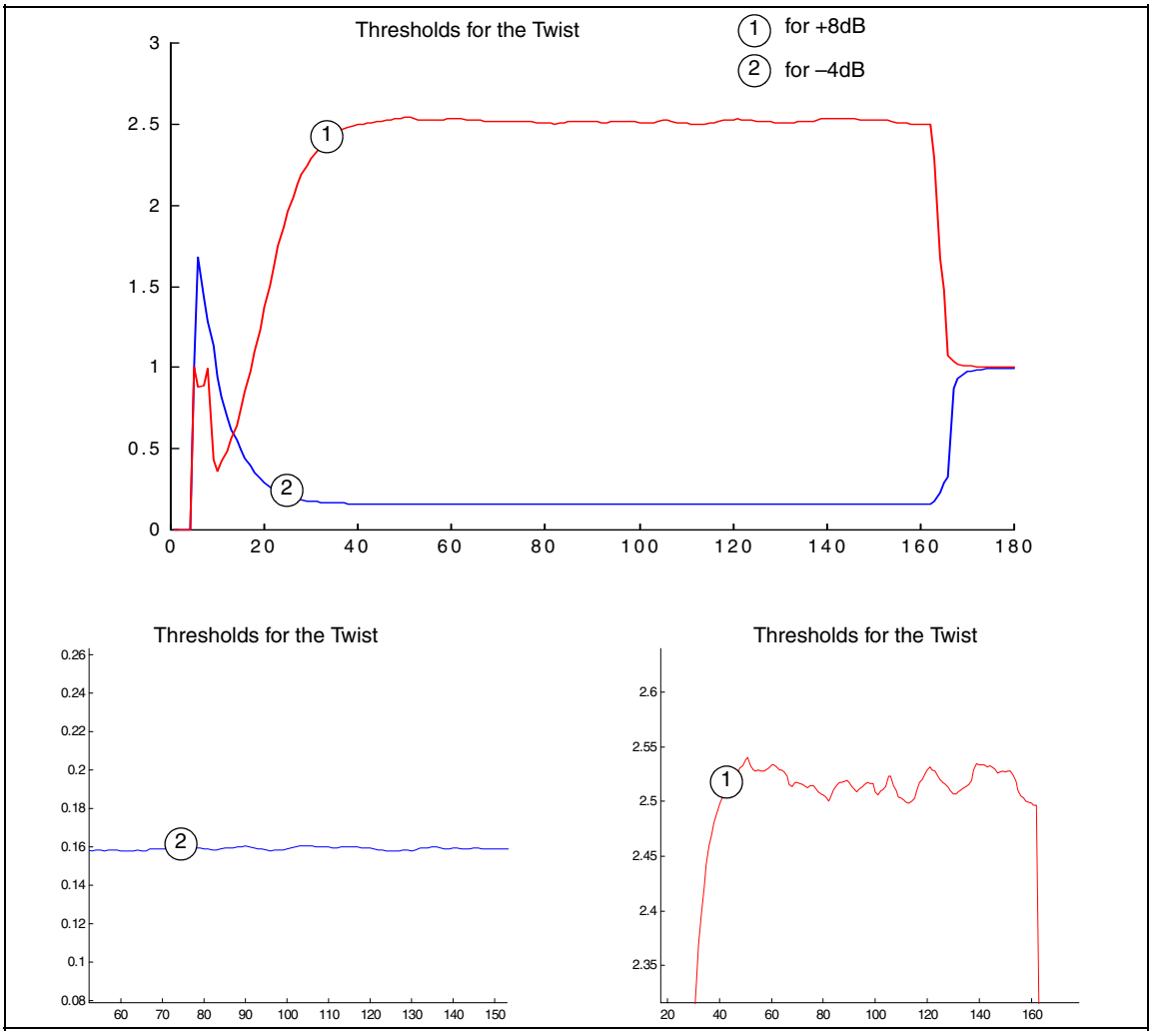

Figure 6. Plots of Test Results

4.2.4 Signal Reception Timing

- To respect the signal timing condition, we must know the duration of the signal and the pause. To get this information, we must detect an edge in the signal, which can correspond to the beginning or the end of a key, as depicted in **Figure 7**.

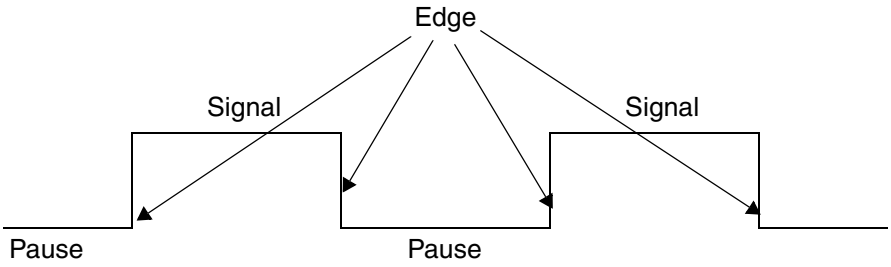

Figure 7. Transition Between Pause and DTMF Signal

A potential digit is determined from every processed sample. A comparison between two consecutive digits occurs every other sample (the detector works at 4 KHz with an 8 KHz sampled input). When a difference (potential edge) is detected, the sample number is stored so that the calculation of the duration uses the previous edge index. This duration is compared to the values defined by the Q.24 recommendation, and a decision is made. The detailed implementation of the test is explained in the next section. At a sampling rate of 4 KHz, 40 ms corresponds to 160 samples. Because of the transition period (the transition state at the beginning and end of a given digit), the signal and pause durations must be decreased. The values in the implementation are defined experimentally, as illustrated in the plots shown in **Figure 8**. After a series of keys is tested, digits (①) and corresponding samples (②) are represented. The figures focus on one particular key, where transition periods appear at the beginning of the detection of key 2. The signal characteristics of the first experiment are given by:

- Sampling rate: 4 KHz
- Power of tone: 0 dBm0
- Twist: 4 dB
- SNR: 40 dB
- Phase offset: 0
- Detected key: 2

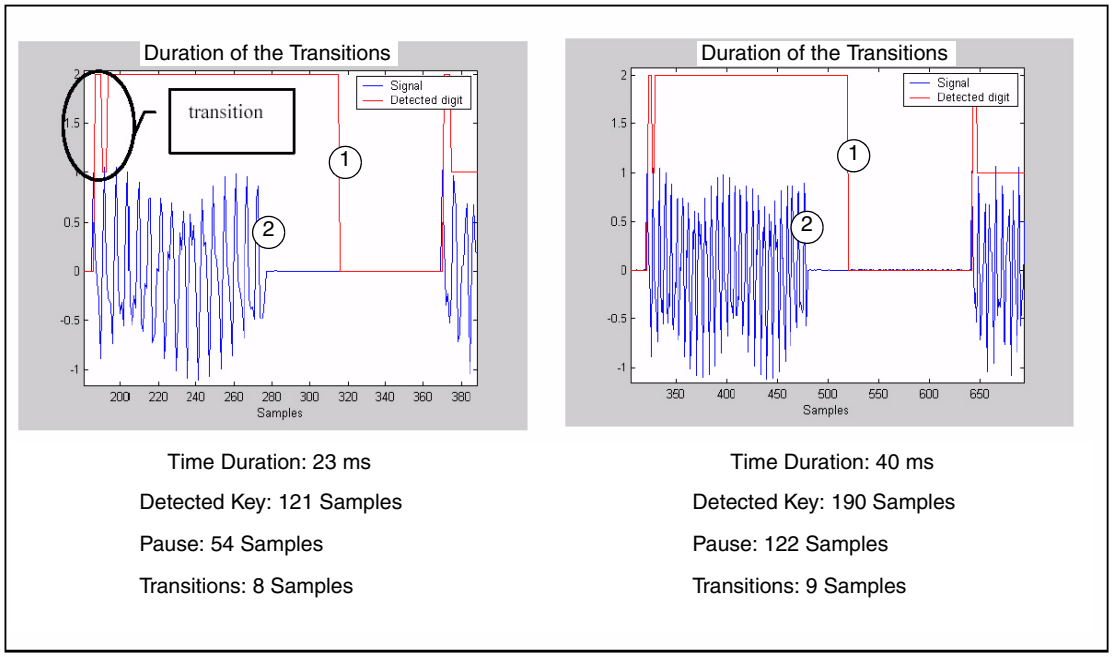

Figure 8. Plots of Experiments to Define Implementation Values

For this signal power (0 dBm0), the average of each part is:

- Detected key: 145 samples
- Pause: 88 samples
- Transitions: 9 samples

The second experiment tests the influence of the signal power and the noise (see **Figure 9**). The other characteristics (twist, power of the tone, and so on) are the same as in the previous experiment, but the signal power is now reduced to –12 dBm0, SNR = 10dB. The average of each part is:

- Detected key: 121 samples
- Pause: 119 samples
- Transitions: 14 samples

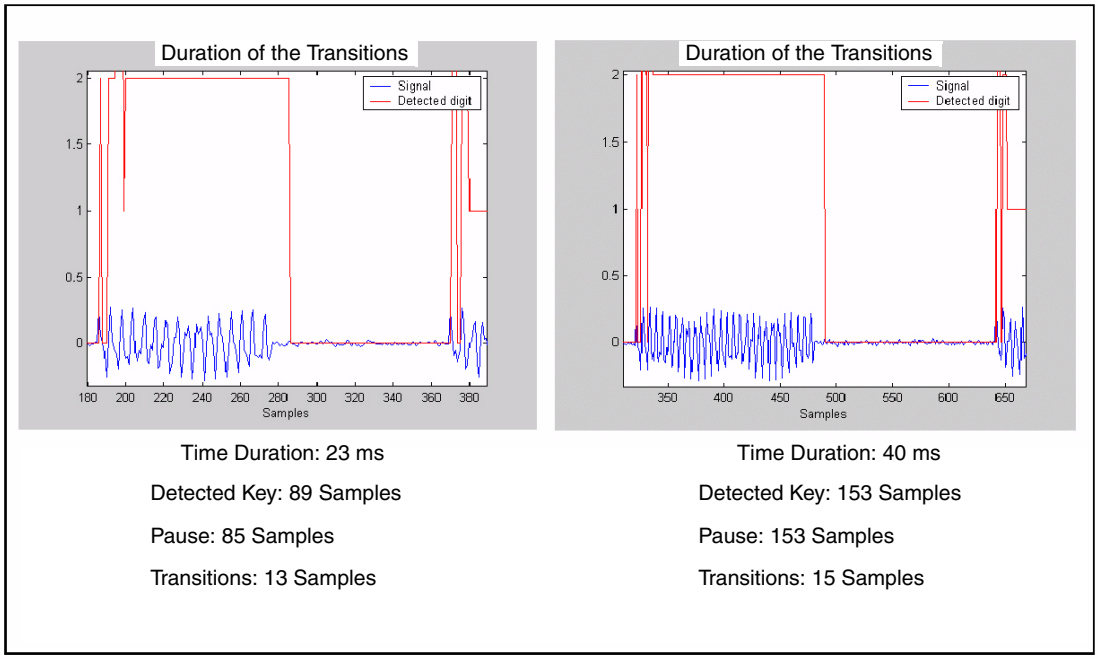

Figure 9. Influence of Signal Power and Noise

These results are approximately the same for all keys, and if there is less twist, the transitions are usually faster. These experiments demonstrate the sensitivity of duration as a function of other factors, such as noise, low power level, twist, and so on. Nevertheless, theoretical and practical duration thresholds also differ because of C implementation effects, particularly 16-bit rounding. This is why previously calculated duration thresholds must be further adjusted by looking at the detected duration of valid and non-valid digits with the final C program.

Figure 10 shows the duration of keys detected in Mitel test 6 [7]. This test consists of pulses with a duration that starts at 49 ms and gradually reduces to 10 ms. The measured duration (vertical axis) is calculated according to the number of times a digit is detected, whereas the real duration (in ms) of the sound is 49 – key index / 10. This test shows that a significant variance in duration must be accounted for in the C implementation. The duration, represented by the number of samples at 4 KHz in **Figure 7** (floating-point version of the detector), is still correct on the average. For instance, at the beginning, 200 samples per key are reported (a 49 ms pulse corresponds to 49 ms × 4 KHz = 196 samples). However, the variation (peak-to-peak deviation of the line) often reaches 20 samples, even for keys that must be detected.

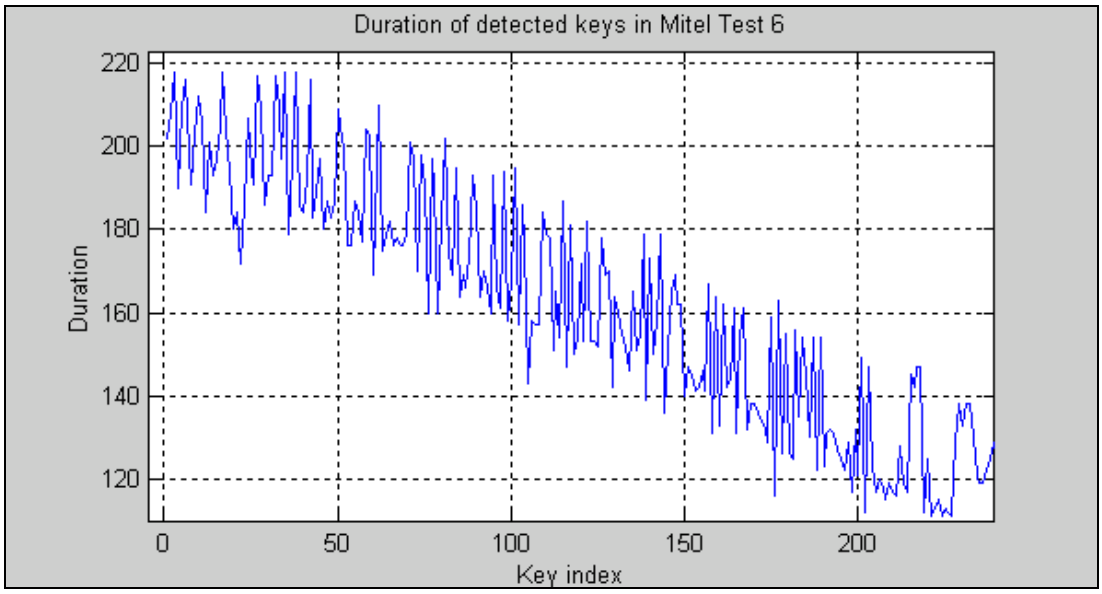

Figure 10. Duration of Detected Keys

Considering that a shorter signal duration threshold may decrease the performance in talk-off tests, the final value is calculated to keep the right margin below the upper duration limit (40 ms, 160 samples) and above the lower one (23 ms, 92 samples). The interruption duration is adjusted in the same way. However, the pause duration can be more tolerant since its reduction does not have undesirable influence on other parts of the detector (see **Table 5**).

Table 5. Theoretical and Practical Durations

Number of Samples (4 KHz)	Pause_duration	Signal_duration	Interruption_duration
Theoretical	160	160	40
Practical	100	110	50

4.2.5 Digit Processing Unit

The ITU-T Q.24 recommendations for signal and pause duration are as follows:

- The signal must be at least 40 ms long to be detected as a key.
- Between two detected keys, there must be a non-valid signal that lasts at least 40 ms (that is, a pause).
- Signals with a total of 10 ms or less interrupt duration (silence, for example) should be detected.

To implement these requirements, the following variables are used:

- *new_digit*. The digit just found by the sample processing unit.
- *previous_digit*. Previous loop digit.
- *digit*. Digit recorded before previous_digit and having a significant duration.
- *i_min*. Index corresponding to the beginning of the digit.
- *i_edge*. Index corresponding to the end of the digit, the beginning of previous_digit, and the last detected edge.

- *start*. Simulated time that corresponds to the instant when the function is called. It is incremented at the end of each call by the number of processed samples. Since it is a 16-bit value, the program initializes it to 0 when it becomes large enough and during an appropriate non-valid signal.
- *i+start*. Current loop index.
- *pause_state*. Set to 1 when a pause is detected. It is a necessary condition to detect a key.

All indexes simulate time by processing the samples at 4 KHz (every increment for each of the indexes is equivalent to 250 ms).**Figure 11** shows the overall digit processing flow chart.

This decision logic section first processes previous_digit by analyzing its duration (Step 1). It may be ignored (case (A)) or may replace the digit (case (B)). Based on the result of Step 1, a decision is made about the digit (Step 2). The digit processing part of the algorithm is performed only when an edge is detected, that is, when new_digit differs from previous_digit.

4.2.6 Adjustments of the Sample Processing Unit

The parameter α, which is the IIR low pass filter (LPF) gain, influences the bandwidth of the low pass filters. If its value is close to 1, it decreases the filter bandwidth. The parameter *r* is the IIR notch filter gain that controls roll-off around the notch location. Optimized values of α and *r*, which generate good detection results, are determined experimentally. All 16 DTMF digits must be considered. Extensive tests were conducted for different values of power level (0 dBm0, –1 dBm0, –18 dBm0, and –22 dBm0), with a duration of 40 ms. The experiments show that the best values for α and *r* are as follows:

$$\begin{cases} \alpha = 0.88 \\ r = 0.35 \end{cases}$$

To reduce noise effects, two bandwidths are used in low pass filters. When the same digit is detected several times, the LPF with the lower bandwidth is used (filter with coefficient α_locked) to focus on the locked frequency. When the frequencies are unlocked, the LPF with the larger bandwidth is used (filter with coefficient α_unlocked) to allow faster response to any new tone. Each value is further optimized, resulting in the following final values:

$$\begin{cases} alpha\_locked = 0.91552734375 & (0x7530) \\ alpha\_unlocked = 0.701904296875 & (0x59D8) \\ r = 0.36773681640625 & (0x2F12) \end{cases}$$

As differences among theoretical values, simulation results, and actual (real-time) detector results appear, all constants may be optimized, either empirically or theoretically. The following parameters were adjusted in relation with various Q.24 tests (Mitel, CSELT, Bellcore): frequency tolerance tables, significant signal duration, pause duration, signal duration, and interruption duration. If the detector performance needs to change, these values can be tuned again. Other variables can also be optimized, such as coefficients of notch filters, energy thresholds, minimum duration for frequency lock, and thresholds for sample amplification.

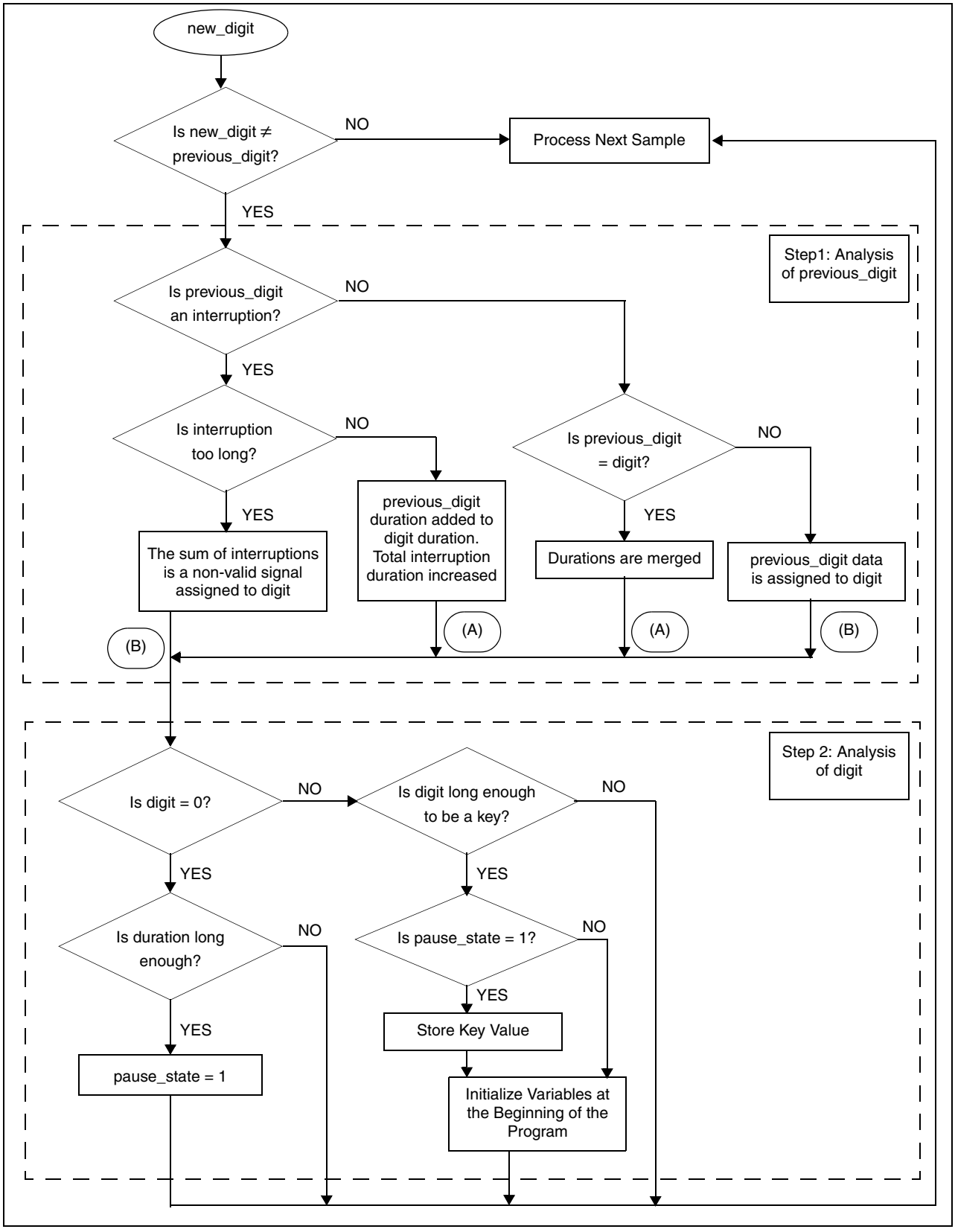

Figure 11. Digit Processing Block Diagram

5 Functional Interface

The DTMF detector consists of two C-callable functions, `fsl_dtmf_det_init`() for DTMF detector initialization and `fsl_dtmf_det()` for running the DTMF detector process.

5.1 fsl_dtmf_det_init

The `fsl_dtmf_det_init` function is called when a voice channel is opened to initialize all channel-dependent data. The interface is provided as:

```
void fsl_dtmf_det_init (DTMF_detector_data *dtmf_det_chan)
```

- `DTMF_detector_data` is an 8-byte aligned channel-dependent data structure.
- `*dtmf_det_chan` is a pointer to the channel-dependent data.

5.2 fsl_dtmf_det

The `fsl_dtmf_det` function is called for per channel DTMF detection with the following interface, and it returns the number of keys contained in the detected_keys vector:

```
INT8 fsl_dtmf_det (FRACTION16 *samples, INT8 *detected_keys, DTMF_detector_data
      *dtmf_det_chan, INT16 buffer_size);
```

- `FRACTION16` is a data type for an 8-byte aligned 16-bit signed fraction.
- `DTMF_detector_data` is an 8-byte aligned channel-dependent data structure.
- `INT16` is a data type for a 16-bit signed integer.
- `*samples` is a pointer to the linear data input.
- `*detected_keys` is a pointer to the output where keys are coded according to the following table.
- `*dtmf_det_chan` is a pointer to the channel-dependent data.
- `buffer_size` is the number of samples to be processed in the input buffer.

Key	1	2	3	4	5	6	7	8	9	0	*	#	A	B	C	D
Code	1	2	3	5	6	7	9	10	11	14	13	15	4	8	12	16

5.3 Code Size and Performance

No reference input files exist, so files from Mitel tests are used to determine how many millions of cycles per second (MCPS) the program requires per channel. **Table 6** shows the results obtained with the hand-optimized assembly code.

Table 6. Assembly Code Results

Input File	Minimum MCPS	Average MCPS	Maximum MCPS
Mitel Test 1	0.1540	0.3409	0.3755
Mitel Test 2	0.1540	0.1914	0.3945
Mitel Test 3	0.1540	0.1775	0.4385
Mitel Test 4	0.1540	0.2066	0.4310

Table 6. Assembly Code Results (Continued)

Input File	Minimum MCPS	Average MCPS	Maximum MCPS
Mitel Test 5	0.1540	0.1718	0.3785
Mitel Test 6	0.1540	0.1971	0.3860
Mitel Test 7a	0.1540	0.2602	0.3900
Mitel Test 7b	0.1540	0.2640	0.3875
Mitel Test 7c	0.1540	0.2649	0.3815
Mitel Test 8	0.1540	0.1982	0.4235
Test average	**0.1540**	**0.2273**	**0.3987**

Memory requirements are as follows:

- Stack size: 80 bytes for the detector, 0 bytes for the initialization function.
- Tables: 100 bytes.
- Program: 2620 bytes.
- Data: 64 bytes/channel.

6 Using the DTMF Detector

The Freescale Starcore DSPs based on the SC140 core and supporting development tools provide a rich set of features to optimize performance in demanding signal processing applications. Different code optimization techniques are used to boost performance in MCPS and reduce code size. The following method was chosen to implement the tone detector:

- Adapt original C code to take advantage of DSP architecture, that is, four parallel DALUs (data arithmetic logic units).
- Use the Metrowerks Compiler (Version 1.5 here) to compile this C code with automated optimization.
- Finally, perform hand optimization on the assembler to take further advantage of efficient assembly instructions, such as multi-word move instructions.

For example, this method reduced the peak search in the AGC from 20 cycles per sample. The method resulted in an overall savings of 0.5 cycles per sample, yielding a savings of approximately 97 percent in program speed and offering significant reduction in memory usage. Easy to integrate on a multi-channel system, the detector [11] requires only the following elementary tasks, which are illustrated by the driver code on the next page:

1. After memory is assigned to a channel data structure, initialize it by calling the `fsl_dtmf_det_init` function.
2. Ensure that the input vector size is equal to or larger than 80 samples. Otherwise, performance may degrade due to amplitude modulation introduced by the AGC.
3. The `fsl_dtmf_det` function fills the output vector with the number of elements given by the 16-bit unsigned integer in `dtmf_det_chan` (`dtmf_det_chan` is the data structure associated to a channel) with offset 46.

You can empty this vector after several function calls by clearing the 16-bit value. Clearing the output vector is optional. The output vector size must be large enough to store the maximum number of keys between two checks of these keys. One interval of silence followed by a key may last down to 93 ms (Q.24 requirements for signaling

velocity), so that the size is equal to the time between two checks / 93 × 10–3. Alternatively, you can check whether a key has been detected after each call, using an input buffer that is less than 93 ms long, and clear the variables in case of success. In both cases, you should add some margin to the output vector size since the 93 ms value is recommended by the Q.24 norm, but it may be lower.

```
#include <stdio.h>
#include <stdlib.h>
#include "fsl_dtmf_det.h"
#define  SIZE_BUF         800
void load_samples(fract16 samples, int channel); //external function
int i, j, channel, len;
FILE *txid;
short samples[SIZE_BUF];
#pragma align samples 8
UINT8 buffer[SIZE_BUF * 2];
DTMF_detector_data dtmf_channel[2];
#pragma align dtmf_channel 8
UINT8 detected_keys[4];
UINT16 *p_num_dtmf_digits;
char dtmf_key[16] = {
     '1', '2', '3', 'A',
     '4', '5', '6', 'B',
     '7', '8', '9', 'C',
     '*', '0', '#', 'D'
};
char s[] = "Detected DTMF key _.\n";
void
main ()
{
     printf ("\n\n*** Generic Tone Detection on StarCore SC140 ***\n\n");
     printf ("DTMF test: \n\n");

     fsl_dtmf_det_init (&dtmf_channel[0]);
     fsl_dtmf_det_init (&dtmf_channel[1]);

  for (i = 0; i < SIZE_BUF * 2; i++) {
    buffer[i] = 0;
    samples[i >> 1] = 0;
  }

  while (1) {
    for (channel = 0; channel < 2; channel++) {
        load_samples(samples,channel);
        fsl_dtmf_det(samples,detected_keys,&dtmf_channel[channel], SIZE_BUF);
        p_num_dtmf_digits = (UINT16 *) (&dtmf_channel[channel] + 46);
        if (*p_num_dtmf_digits) {
          for (j = 0; j < *p_num_dtmf_digits; j++) {
            s[18] = dtmf_key[0xf & (detected_keys[j] - 1)];
            printf ("%s", s);
          }
          *p_num_dtmf_digits = 0;
      }
    }
  }
}
```

7 DTMF Q.24 Compliance Tests

The DTMF detection algorithm is written in a mixture of both ANSI C and MSC8101 assembly code. The complete module, along with drivers for peripherals (that is, ethernet and codec), is a Metrowerks project using version 1.5 of Metrowerks IDE for Starcore SC140. This code runs in real time on the MSC8101ADS [4].

Mitel tests [7], Centro Studi E Laboratori Telecomunicazionis (CSELT) talk-off tests, and Bellcore talk-off tests [8] were performed. On the hardware set-ups illustrated in **Figure 12**, the three following methods of testing were used:

- *Set-up 1*. Ethernet digital PCM linear data. Audio files are sent to the detector via the 100 Mbit Ethernet port. Sample quality remains high. Detected keys are stored in an array and written into a file at the end of the process.
- *Set-up 2*. Ethernet digital A/μ-Law encoded data with the same device as Set-up 1.
- *Set-up 3*. Linear codec analog linear data. The MSC8101ADS codec is described in [9]. Before testing of the detector, the noise added by the codec was analyzed to ensure that it was not excessive. Audio files are played by the PC sound card, which is then plugged in to the codec input.

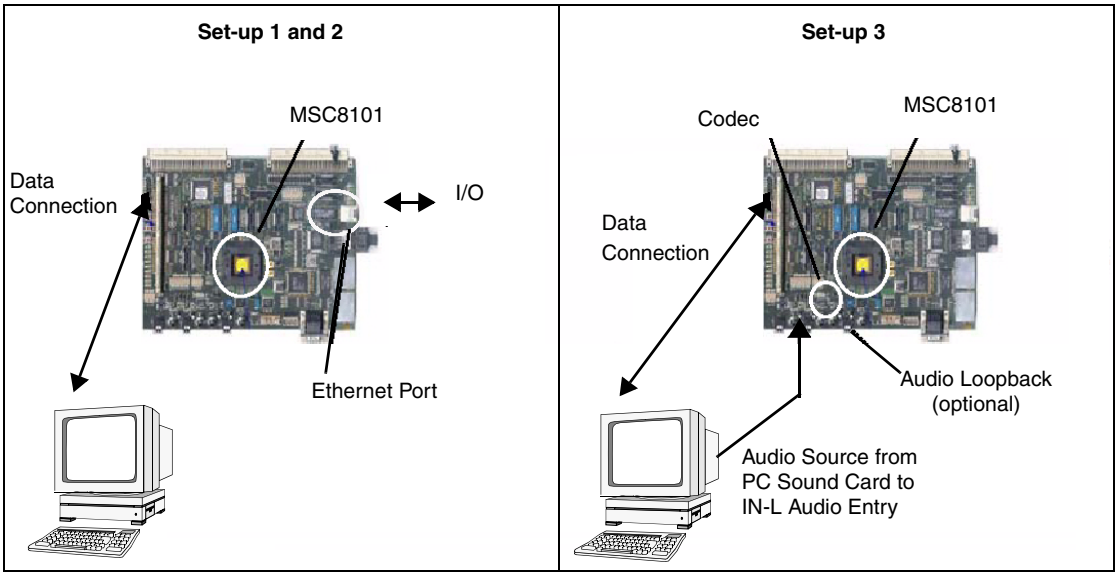

Figure 12. Testing Set-ups

Testing results are listed in detail as follows with (P) indicating Q.24 compliance (see **Table 7**):

- *Mitel Side 1 Test 2*: *Decode Test Digits*. Decode test digits 1 to 16 (10 pulses each). In all cases, all digits are detected. The test passed.
- *Mitel Side 1 Test 3: Bandwidth and center frequency check*. This test determines the receiver recognition bandwidth (RRB) and receiver channel center frequency offset (RCFO) for both the H and L frequency for the keys 1, 5, 9, and D.

  ITU-T Q.24 recommends that a frequency deviation of less than or equal to 1.5 percent must be recognized as a valid digit and a deviation of more than 3.5 percent must be considered as a non-valid digit. Thus, a passing score on a given item must be between 15 and 34 detects. This test includes the result of the calibration tone in Side 1 Test 1, which was measured to be 1000 Hz. This test shows results consistent with the talk-off test results presented later. The test passed.

Table 7. Mitel Side 1 Test 3 Results Summary

Key	Frequency	Variation	Set-up 1	Set-up 2	Set-up 3
1	697 Hz	+0.1 to +4%	21 (P)	20 (P)	21 (P)
		–0.1 to –4%	28 (P)	28 (P)	28 (P)
		RRB%	5.0	4.9	4.9
		RCFO%	–0.15	–0.2	–0.2
	1209 Hz	+0.1 to +4%	22 (P)	21 (P)	22 (P)
		–0.1 to –4%	29 (P)	28 (P)	30 (P)
		RRB%	5.1	5.1	5.2
		RCFO%	–0.2	–0.2	–0.25
5	770 Hz	+0.1 to +4%	21 (P)	21 (P)	21 (P)
		-0.1 to -4%	27 (P)	26 (P)	27 (P)
		RRB%	4.7	4.8	4.8
		RCFO%	–0.2	–0.15	–0.15
	1336 Hz	+0.1 to +4%	20 (P)	20 (P)	21 (P)
		–0.1 to –4%	28 (P)	28 (P)	28 (P)
		RRB%	4.8	4.8	4.9
		RCFO%	–0.15	–0.25	–0.2
9	852 Hz	+0.1 to +4%	25 (P)	25 (P)	26 (P)
		–0.1 to –4%	27 (P)	27 (P)	27 (P)
		RRB%	5.1	5.2	5.3
		RCFO%	0.1	0.05	0.1
	1477 Hz	+0.1 to +4%	22 (P)	22 (P)	24 (P)
		–0.1 to –4%	29 (P)	28 (P)	29 (P)
		RRB%	5.1	5.1	5.3
		RCFO%	–0.2	–0.2	–0.1
D	941 Hz	+0.1 to +4%	26 (P)	27 (P)	27 (P)
		–0.1 to –4%	22 (P)	23 (P)	23 (P)
		RRB%	4.8	4.8	5.0
		RCFO%	0.35	0.35	0.35
	1633 Hz	+0.1 to +4%	25 (P)	23 (P)	27 (P)
		–0.1 to –4%	26 (P)	27 (P)	30 (P)
		RRB%	5.1	5.1	5.7
		RCFO%	0.2	0.1	0

- *Mitel Side 1 Test 4: Amplitude ratio*. This test determines the acceptable amplitude ratio in dB of both tones in the digits 1, 5, 9, and D. In the first sequence, the H tone amplitude is maintained and the L tone is attenuated so the amplitude ratio L/H varies from 0 to –20 dB (standard twist). In the second sequence, the L tone amplitude is maintained and the H tone is attenuated so the amplitude ratio varies from 0 to +20 dB (reverse twist). The AT&T requirements of the Q.24 standard require the amplitude ratio of the first sequence to be ≤–4 dB and ≥ +8 dB for the second sequence. The test passed.

Table 8. Mitel Side 1 Test 4 Results Summary

Key	Frequency	Set-up 1	Set-up 2	Set-up 3
KEY-1	0 to –20 dB	19.9 (P)	20.0 (P)	19.6 (P)
	0 to +20 dB	17.6 (P)	17.9 (P)	16.9 (P)
KEY-5	0 to -20 dB	20.0 (P)	20.0 (P)	20.0 (P)
	0 to +20 dB	13.8 (P)	14.6 (P)	16.0 (P)
KEY-9	0 to -20 dB	20.0 (P)	20.0 (P)	20.0 (P)
	0 to +20 dB	11.3 (P)	11.9 (P)	14.7 (P)
KEY-D	0 to -20 dB	20.0 (P)	20.0 (P)	19.8 (P)
	0 to +20 dB	17.9 (P)	17.3 (P)	11.8 (P)

- *Mitel Side 1 Test 5: Dynamic range*. This test determines the dynamic range in dB by gradually attenuating the tones in the key 1. The AT&T requirements of the Q.24 standard require the power level to be at least –25 dBm0. The test passed.

Table 9. Mitel Side 1 Test 5 Results Summary

Set-up 1	Set-up 2	Set-up 3
29 dB (P)	29 dB (P)	27 dB (P)

- *Mitel Side 1 Test 6: Guard time*. This test determines the minimum signal duration in milliseconds to detect the tones in the key 1. The AT&T requirements of the Q.24 standard require the minimum signal duration to fall between 40 and 23 ms. The test passed.

Table 10. Mitel Side 1 Test 6 Results Summary

Set-up 1	Set-up 2	Set-up 3
28.7 ms (P)	28.5 ms (P)	27.4 ms (P)

- *Mitel Side 1 Test 7: Signal-to-noise ratio*. This test determines whether a receiver can detect 1000 pulses of the tones in the key 1 with white noise to create three different signal to noise ratios. The test passed.

Table 11. Mitel Side 1 Test 7 Results Summary

S/N Ratio	Set-up 1	Set-up 2	Set-up 3
S/N -24 dBV	1000 (P)	1000 (P)	1000 (P)
S/N -18 dBV	1000 (P)	1000 (P)	1000 (P)
S/N -12 dBV	1000 (P)	1000 (P)	1000 (P)

- *Mitel Side 2 Test 2 Talk-off test*. This test is a 30-minute condensed recording of conversations. The Mitel document states that a receiver with an acceptable talk-off response should register less than 30 tones. The test passed.

Table 12. Mitel Side 2 Test 2 Results Summary

Set-up 1	Set-up 2	Set-up 3
1 (P)	1 (P)	1 (P)

- *CSELT Talk-off test*. This test is a 20-minute highly condensed recording of sounds from conversations. The speech bursts are 150 ms each. The CSELT documentation does not directly state what an acceptable talk-off response is. The ETSI specification for DTMF transmitters and receivers [10] allows five responses for this test. The test passed.

Table 13. CSELT Talk-off Test Summary

Set-up 1	Set-up 2	Set-up 3
0 (P)	0 (P)	0 (P)

- *Bellcore tests*. Six 30-minute recordings compose this talk-off test. Detected keys are counted in-groups, with a maximum number of detects allowed in each group. The test passed.

Table 14. Bellcore Tests Summary

Key Groups	0 to 9	0 to 9, * and #	All 16 digits
Norm	<333	<500	<666
Set-up 1	9 (P)	9 (P)	9 (P)
Set-up 2	9 (P)	9 (P)	9 (P)
Set-up 3	9 (P)	9 (P)	9 (P)

8 Conclusion

This application note presents the theory and design of a generic tone detector based on the Teager-Kaiser (TK) energy operator along with its practical application within the context of a DTMF detector. It explains the practical implementation details and describes the extensive test methodology and results.

The DTMF detector implementation on the Freescale SC140 core has two major components: initialization (fsl_dtmf_det_init) and actual detection (fsl_dtmf_det). The first component initializes the states of the channel-dependent data structure of the detector. The core of the DTMF tone detector (second component) relies on TK energy operators.

The core detection process works at 4 KHz, on a per-sample basis, with a block of input samples. One intrinsic feature of this implementation is that samples do not have to be buffered prior to the detection and decision parts of the algorithm. Consequently, the program could be adapted to run without any input buffer by changing the AGC. The algorithm is easily tuned, and extending it to other signaling standards, such as MF-R1 and MF-R2, is fairly straightforward. Because of the simplicity of the implementation and the architectural advantages of the MSC8101, the module requires low average MCPS and memory. These features make the TK energy operator implementation an attractive alternative to conventional algorithms, such as Goertzel filters, and consequently favor its use in multi-channel infrastructure applications such as media gateways. The proposed method passed all the major tests, including Mitel, BellCore, and CSELT, proving to be robust in terms of frequency tolerance, twist, power level, guard time, SNR, and talk-off. All available tests for Q.24 recommendation passed (section 2 of Q.24, with the exception of 2.7 to 2.9).

Use of the Teager-Kaiser energy operator in tone detection applications yields a number of significant advantages: faster response, versatility, improved time resolution, independence between time and frequency resolutions, ease of maintenance and fine tuning. Additionally, the decision logic has very few control parameters, which can easily be adjusted for different tone formats. The TK-based algorithm is inherently simple to reuse and support, and it can deliver generic tone detection functionality and performance comparable to or better than that of classical methods.

9 References

[1] J. F. Kaiser, “On a Simple Algorithm to Calculate the ‘Energy’ of a Signal,” *Proceedings of the IEEE ICASSP’90*, pp. 149–152.

[2] P. Maragos, J.F. Kaiser, and T.F. Quartieri, “On Amplitude and Frequency Demodulation Using Energy Operators,” I*EEE Transactions in Signal Processing*, Vol. 41, No. 4, pp. 1532–1550, 1993.

[3] B. Santhanam, et al., “Multicomponent AM-FM demodulation via Periodicity-based Algebraic Separation and Energy-based Demodulation,” *IEEE Transactions on Communications*, vol. 48, no. 3, pp. 473–489, March 2000.

[4] Freescale Semiconductor, *MSC8101 Application Development System User’s Manual*.

[5] ITU-T Recommendation Q.24, Multi-frequency Push Bottom Signal Reception, 11/98.

[6] ITU-T Recommendation G.711, Pulse code modulation (PCM) of voice signals, 1993.

[7] MITEL Semiconductor DTMF Receiver Test Cassette, CM7291.

[8] Bell Communications Research, Digit Simulation Test Tape, TR-TSR000763, July 1987.

[9] Freescale Semiconductor, *MSC8101ADS MCC/SPI/Codec demo*, Sept. 2000.

[10] ETSI Recommendation ES 201235-3 v1.1.1 (2000-09), Specification of Dual Tone Multi-Frequency (DTMF) Transmitter and Receivers; Part 3: Receivers.

[11] Freescale Semiconductor, Packet Telephony Operation, *DTMF Transmitter/Receiver for MSC810x DSPs*.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations not listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GMBH
Technical Information Center
Schatzbogen 7
81829 München, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
+800 2666 8080

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN2384
Rev. 1
12/2004

freescale™
semiconductor