Totally Logical

# DETECT DTMF TONES USING THE Z02201/Z02922

## INTRODUCTION

The Z02201/Z02922 datapump does not perform automatic detection of DTMF tones. However, it is possible to detect DTMF tones in a host processor using the tone detection facilities provided by the Z02201/Z02922 datapumps. This Application Note explains the basic strategy employed to robustly detect DTMF tones.

## DTMF DETECTION

DTMF stands for Dual Tone Multi-Frequency. As such, DTMF detection involves detecting tone pairs. While this Application Note describes how to detect the standard DTMF tone pairs used by the telephone system, the strategy employed here can also be used to detect any tone pair.

DTMF tones are formed from 8 basic tones. Each tone is comprised of a pair of tones, offering 16 different combinations of frequencies, or 16 tones, as outlined in Table 1.

The requirements for DTMF tone detection are outlined in the ITU-T recommendation Q.24. Though the tone detectors in the Z02201/Z02922 datapumps return a value approximately every 27 ms, it is not possible to implement all of the timing constraints outlined in Q.24. While it should be possible to detect tones as short as 40 ms, it is not possible to meet the criteria for momentary interruption of the tone. Q.23 calls for interruptions shorter than 24 ms to be ignored. These interruptions cannot be timed.

**Table 1. DTMF Tone Pair Frequencies**

| High Band/ Low Band | 1209 Hz | 1336 Hz | 1477 Hz | 1633 Hz |
|---|---|---|---|---|
| 697 Hz | 1 | 2 | 3 | A |
| 770 Hz | 4 | 5 | 6 | B |
| 852 Hz | 7 | 8 | 9 | C |
| 941 Hz | * | 0 | # | D |

## DETECTION STRATEGY

The basic strategy involved uses the 16 tone detectors provided by the Z02201/Z02922 to detect not only the tone pairs but also the second harmonics of the tones. While standard DTMF tones are pure tones with no harmonics, normal speech is rich in harmonics. Therefore, to reliably detect DTMF tones without confusing the tones with normal speech requires the presence of the primary tones and an absence of any second harmonic tones.

In addition, standard DTMF tones exhibit a “twist” in their power levels. That is, one tone is transmitted at a higher power level than the other. One must therefore compare the power levels of the two detected tones to make sure that the twist is within acceptable bounds.

An inter-digit silence period must also be detected before moving onto the next tone. The following steps outline the basic procedure for DTMF tone-pair detection.

### Step 1

Set up the 16 tone detectors, as outlined in Table 2. Set up the bottom eight detectors for the primary tones, set up the top eight to detect the second harmonics, assign the threshold, and set CONFIG.mode to 2 (tone detect mode).

### Step 2

Look for a tone. Check the tone detector status output (RAM location 0x0) for presence of tones. If there are any tones, their value will be non-zero.

**Step 3**
Read all 16 relative power levels. These levels are used to check the twist.

**Step 4**
Check the bottom 8 tones for pairs. View the bottom 8 bits of the tone detector status, view the levels, and select the two dominant tones. One of the dominant tones should be from the high band (top 4 bits) while the other should be from the low band (bottom 8 bits). If 2 tones are not detected, or both tones arise from the same band, the test has failed. Return to step 2.

**Step 5**
Check the twist. Of the two dominant tones, the high band should be a higher power level than the low band. Additionally, the tone levels should be checked relative to each other for the proper twist. A ratio of 1-to-2 is correct for North America (a twist of 2-4 dB). If the dominant tone is from the low band (negative twist), or the ratio is wrong (<1 or >2), the test has failed. Return to step 2.

**Step 6**
Check for second harmonics. Compare the primary tone power levels to those of the second harmonics. There should not be any harmonics. If harmonics are present, the power level of the primary tone, with respect to any second harmonic, should be very low (a ratio of >10 is acceptable). If these conditions are not met, the test has failed. Return to step 2.

**Step 7**
The digit is detected. Decode the digit from the bottom 8 bits of the tone detector status word.

**Step 8**
Look for the silence period and for the overall power level to diminish. This indicator must be debounced. Also look for the tone detector status bits to go to zero. When these conditions are met, return to step 2 to detect the next tone.

These steps must be followed to detect DTMF tones. While these steps may appear straightforward and simple, the implementation of the code in the external controller is more complex. The next section describes an implementation example.

**Table 2. Tone Detector Frequencies and Coefficients**

| Detector | Frequency | Coefficient |
|---|---|---|
| 0 | 697 Hz | 0x72E8 |
| 1 | 770 Hz | 0x7015 |
| 2 | 852 Hz | 0x6C9B |
| 3 | 941 Hz | 0x687A |
| 4 | 1209 Hz | 0x59F8 |
| 5 | 1336 Hz | 0x521A |
| 6 | 1447 Hz | 0x48B4 |
| 7 | 1633 Hz | 0x3D96 |
| 8 | 1394 Hz | 0x4E51 |
| 9 | 1540 Hz | 0x444C |
| 10 | 1704 Hz | 0x384F |
| 11 | 1882 Hz | 0x2A91 |
| 12 | 2418 Hz | 0xFE7E |
| 13 | 2672 Hz | 0xE956 |
| 14 | 2894 Hz | 0xD29A |
| 15 | 3266 Hz | 0xBB47 |

## IMPLEMENTATION EXAMPLE

The Example Code on page 4 is an implementation of the algorithm described above. The ideal place to perform DTMF detection is in the DSP at every sample. Because the Z02201/Z02922 currently does not support DTMF detection, the user is forced to use the external controller and the facilities provided by the Z02201/Z02922 datapumps to detect tones. There are some trade-offs that must be made as a result of this circumstance. The first version of this code was implemented according to the steps outlined above. A number of problems were uncovered during the testing phase and some compromises were made.

1. It was discovered that, for some digits, the second harmonic of the low tone was close to the fundamental frequency of the high tone. The digit "2" provides a good example. The fundamental frequency of the low tone is 697 Hz. The second harmonic of this tone is 1394 Hz. The fundamental frequency of the high tone is 1336 Hz. The difference between the high tone and the second harmonic of the low tone is only 58 Hz. Because of the requirement for a fast detection time, the accuracy of the tone detectors must be sacrificed. Testing revealed that the digit "2" could not be detected reliably, because the second harmonic test for the low tone often failed (the detected power levels were less than 20 dB apart). As a result, the check for the second harmonics in the low band had to be sacrificed.

2. The developed code only works for receive levels of –28dB and stronger. At lower receive power levels, the tone detector power levels are so low that it is not practical to distinguish the twist. Lower receive levels can be achieved, but at the cost of robustness. The re-

ceive level is lowered by relaxing the twist test and allowing a “negative” twist (negative twist is reached when the low band appears more powerful than the high band). However, allowing a negative twist also increases the chances of voice being mistaken for valid DTMF tones. Relaxing the twist test for lower receive levels is best employed when the target application does not anticipate receiving a voice signal.

3. At lower receive levels, the “tone on” time works better if the tone is on for a longer period of time. When the transmitter sends the tone, it takes some time for the power level to reach its maximum. At the other end, if attenuation is low, the power level passes the threshold sooner than if the attenuation is high.

4. The tone and silence detection thresholds were chosen to work well for the current application. However, changing the thresholds is recommended to suit a particular environment. For example, if the physical connection is a high-noise connection, the thresholds must be raised to only detect stronger signals. The result, however, unavoidably affects receive sensitivity.

Table 3 presents measurement characteristics.

**Table 3. Requires a title**

| | |
|---|---|
| Minimum Receive Power Level | -28dBm |
| Minimum Tone On Time | 40ms |
| Minimum Tone Off Time | 40ms |
| Maximum Noise Level | -40dBm |

ZiLOG offers a software utility that simplifies design of biquad filters, called Biquad.exe. This software is available for download from ZiLOG’s website at www.zilog.com.

## DETECTING DTMF: A “C” CODE EXAMPLE

Operation of the final circuit in its layout and environmental conditions should be carefully verified with an accurate and properly-guardbanded frequency counter. Verifying in this manner assures system compliance with ITU-T specs (100 ppm). In order to avoid loading the XTAL or EXTAL terminals with the parasitic input capacitance of the frequency counter, the EYECLK (1/16 of the master clock rate) can be monitored instead of the crystal oscillator directly.

## CONCLUSION

By following the suggestions outlined in this Application Note, any user can implement a solid strategy for DTMF (or any tone pair) using features provided by the Z02201/Z02922 datapumps.

## EXAMPLE CODE

```
/* The tone detector coefficients */
const UINT16 tones[16] = {0x72E8, 0x7015, 0x6C9B, 0x687B, 0x59F9, 0x521A,
    0x48B4, 0x3D97, 0x4E51, 0x444D, 0x384F, 0x2A91, 0xFE7E, 0xE955, 0xD29A, 0xBB46};

/* The digit masks */
const UINT8 digits[] = {0x28, 0x11, 0x21, 0x41, 0x12, 0x22, 0x42, 0x14,
    0x24, 0x44, 0x18, 0x48, 0x81, 0x82, 0x84, 0x88};

/* Mask array to convert from bit position to bit value */
const UINT8 index_vals[] = {0x1, 0x2, 0x4, 0x8, 0x10, 0x20, 0x40, 0x80};

/* The biquad filter coefficients for silence detection */
const UINT16 biqdvals[] = {0xca2e, 0x6868, 0x123d, 0xdb94, 0x129c,
    0xd152, 0x3ae0, 0x0f13, 0x116d, 0x129c};

/* Storage for the 16 relative tone levels */
UINT16 tone_levels[16];

/* DSP RAM location addresses */
#define DTDTHRESH 0x3           // Tone detector threshold
#define MINUS30DB 1036          // value for -30db threshold
#define MINUS39DB 368           // value for -39db threshold
#define MINUS42DB 260           // value for -42db threshold
#define DTDSTATUS 0x0           // Tone detector status
#define CONFIG    0x1FF
#define IDLE      0             // CONFIG modes—idle
#define DTONES    2             // Detect tones
#define DPCTRL    0x1FA         // DP Control
#define AGC_FRZ   0x4000        // AGC freeze

#define BQONTHRESH 0x51         // Biquad filter on threshold
#define BQOFFTHRESH 0x52        // Biquad filter off threshold
#define TONESTATUS 0x1FC        // Biquad tone status
#define TONEA     0x8000        // Tone A status bit
#define DTD0      0x145         // Discrete tone detector coefficients
#define BQACOEFFS 0x155         // Biquad Filter A coefficients
#define DTD0LEV   0x26          // Discrete tone detector levels

/* Return the magnitude of the passed parameter */
SINT16 mag(SINT16 i) {
    if (i < 0)              /* if value is < 0 */
```

```
        return 0;           /* we call it 0 */
    return i;
}

/*
 * AT#P—perform dtmf recognition. Test command to check the DTMF
 * detection code.
 */
void detect_DTMF(void) {
    UINT16 temp, tone_status;
    UINT16 hitone, lotone;
    SINT16 ptone, stone;
    UINT8 i, hindex, lindex, digit, have_digit;

    // setup the tone detectors for the DTMF freqs + their second harmonics
    temp = DTD0;
    for (i=0; i<16; i++) {
        write_DSP_RAM(temp, tones[i]);
        temp++;
    }
    // setup biquad A for a flat line energy detector
    for (i=0; i<10; i++)
        write_DSP_RAM(BQACOEFFS+i, biqdvals[i]);
    write_DSP_RAM(DPCTRL, AGC_FRZ);         // freeze the AGC
    write_DSP_RAM(CONFIG, DTONES);
    LWrite_DSP_RAM(BQONTHRESH, MINUS30DB);
    LWrite_DSP_RAM(BQOFFTHRESH, MINUS42DB);
    for (;;) {
        // look for energy in the DTMF frequency band
        for (T_ms = 5; T_ms && (read_DSP_RAM(BQDETECTED) & TONEA););
        if (T_ms)
            continue;
        write_DSP_RAM(CONFIG, IDLE);        // sync-up the tone detectors
        write_DSP_RAM(CONFIG, DTONES);      // to this point.
        LWrite_DSP_RAM(DTDTHRESH, MINUS39DB);// set threshold to -39 db
        LWrite_DSP_RAM(BQONTHRESH, MINUS30DB);
        LWrite_DSP_RAM(BQOFFTHRESH, MINUS42DB);
        have_digit = FALSE;
        for (;;) {
            /* if we detect one or more tones */
            if ((tone_status = LRead_DSP_RAM(DTDSTATUS)) != 0) {
                /* get the relative levels of the received tones */
```

## EXAMPLE CODE (Continued)

```
        temp = DTD0Lev;
        for (i=0; i<16; i++) {
            tone_levels[i] = LRead_DSP_RAM(temp);
            temp++;
        }
        hitone = lotone = 0;
        hindex = lindex = 0;
        temp = tone_status;
        /* now look at the levels of the bottom 8 tones */
        for (i=0; i<8; i++) {
            /* find the two dominant tones */
            if (temp & 0x1) {
                if (tone_levels[i] >= hitone) {
                    lotone = hitone;/* bump the old high to the low */
                    lindex = hindex;
                    hitone = tone_levels[i];
                    hindex = i;
                } else {
                    if (tone_levels[i] > lotone) {
                        lotone = tone_levels[i];
                        lindex = i;
                    }
                }
            }
            temp >>= 1;
        }
        /*
         * look at the twist—it must be in the right direction and
         * the right magnitude.
         */
        temp = hitone/lotone;
        if (hitone && lotone && (hindex > 3) && (lindex < 4)
          && (temp >= 1) && (temp <= 2)) {
            /* Now look at the 2nd harmonics...if there aren't any */
            /* or they are *real* low power, we are OK */
            stone = mag(tone_levels[hindex+8]);  // high band first
            ptone = mag(tone_levels[hindex]);
            if (stone && ((stone > ptone) || (ptone/stone < 10))) {
                continue;                        // failed second harmonics test
            }
            /*
```

```
                     * We only test the second harmonics in the high band
                     * because some low-band second harmonics are close
                     * to a high-band frequency.  The result is false failures.
                     * To test the low-band second harmonics, remove the
                     * comment symbols from the following lines.
                     */
//                  stone = mag(tone_levels[lindex+8]);  // then the low band
//                  ptone = mag(tone_levels[lindex]);
//                  if (stone && ((stone > ptone) || (ptone/stone < 10))) {
//                      continue;                        // failed second harmonics test
//                  }
                    digit = index_vals[hindex] | index_vals[lindex];
                    for (i=0; i<16; i++) {
                        if (digits[i] == digit) {
                            have_digit = TRUE;
                            putdec(i);  // found it!  display it
                            putcrlf();
                            for (;;) {  // wait for end
                                /* if the discrete tone detectors go away */
                                if (!((UINT8)LRead_DSP_RAM(DTDSTATUS) & digit))
                                    break;
                                /* or the total received power goes away */
                              for (T_ms=10; T_ms && !(read_DSP_RAM(BQDETECTED) &TONEA););
                                if (!T_ms)
                                    break;
                            }
                            break;
                        }
                    }
                } else {
//                  puts("failed twist ");
                }
            }
            if (have_digit)
                break;
        }
    }
}
```

ZiLOG, Inc.
910 East Hamilton Avenue, Suite 110
Campbell, CA 95008
Telephone (408) 558-8500
FAX 408 558-8300
Internet: http://www.zilog.com