Incorporating the MSC1210 into Electronic Weight Scale Systems

Terence Chiu *Microsystems Group*

ABSTRACT

The MSC1210 is an integrated system for single-chip high-resolution measurements. Inside the device, there is a 24-bit ΔΣ (delta-sigma) analog core and an enhanced 8051 microprocessor. This application note discusses how the MSC1210 can be incorporated into an electronic weight scale system. It was found that the system had a display (count) resolution of 1 count in 65000 (with software averaging) using a 5kg capacity, 2mV/V Tedea-Huntleigh TH1042 load cell.

Contents

Figures

Tables

1 Load Cell Fundamentals

Nearly every electronic weighing system uses load cells. These cells convert a force or weight into an electrical signal. A strain gauge is the heart of a load cell. The resistance of a strain gauge changes when it is stressed. Strain gauges are typically made from ultra-thin heat-treated metallic foil and chemically bonded to a thin dielectric layer, known as a gauge patch. These patches are then adhered to the surface of the strain element. As the surface becomes strained, the gauge stretches or compresses, changing its resistance in proportion to the applied load. One strain gauge forms one leg of the four-legged Wheatstone bridge. The other three legs have constant resistance. When the excitation voltage (difference between the IN pins) is applied to the circuit, the output (difference between the OUT pins) becomes a differential voltage that is proportional to the force on the cell.

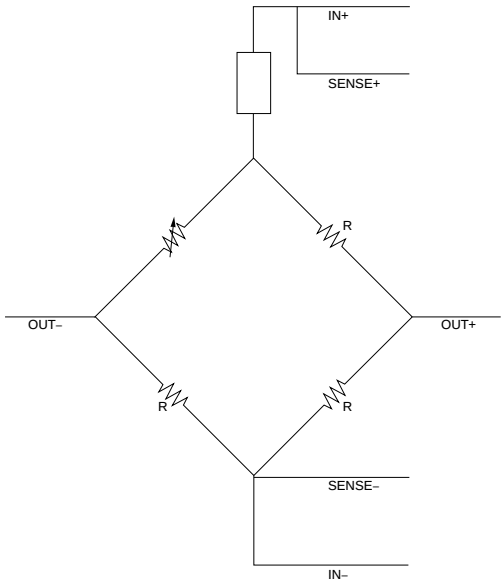

Figure 1. Schematic of TH1042 High Precision Load Cell

Most load cells are specified in terms of maximum capacity and output rating. Maximum capacity defines the maximum load (in kg) to which the load cell can be stressed. After this value is reached, the output voltage no longer changes. The output rating is given in mV/V. For example, a 2mV/V rating indicates that the full-scale output (at maximum capacity) is 10mV if the excitation voltage is 5V.

Since the differential output voltage varies in a linear relation to the weight applied, we can determine the weight from the output voltage ratiometrically. That is, we divide the observed output voltage by the rated output at the excitation voltage. For example, the observed output voltage is 1mV, the rated output is 2mV/V, the excitation voltage is 5V, and the capacity is 5kg; we can then calculate the measured mass (in kg):

Mass (kg) = (observed voltage (in volts)) / (voltage at capacity (in volts)) * capacity (in kg)

= (observed voltage) / (rated output * excitation) * capacity

= 1mV / (2mV/V * 5V) * 5kg = 0.5kg

2 Weight Measurements Using a Load Cell and the MSC1210

The primary goal of a weight measurement system is to convert the differential output signal from the load cell into a weight reading that can be displayed. The performance of a weight scale is based on its accuracy and display resolution (also known as count). Count is defined as the maximum number of bits on the display that are noise-free. To have a high count, we want the noise to be as small as possible. Display resolution (count) is key to determining the performance of a weight scale. A higher count is a strong indicator of better performance.

As a rule of thumb, to have a weight scale with a display resolution of 1 count in 60000, we need to use an analog-to-digital (A/D) converter having an internal resolution at least 4 times greater than the display resolution. That is, the internal resolution should be better than 1 count in 240000, or equivalently, the ADC must have an effective number of bits (ENOB) value of greater than 17.9 (since $2^{17.9}$ = 244590).

In contemporary weight scale systems, multiple devices are needed:

- an A/D converter to convert the analog signal from the load cell output to a digital signal;
- an external reference source for the ADC;
- a microcontroller to process these digital signals (for example, convert to weight);
- peripherals to display weights and drivers to communicate with these peripherals;
- an excitation source to power the load cell.

In using the MSC1210, we replace the ADC, microcontroller, and drivers with a single chip. The only external components needed are the load cell, excitation source, external reference source (this should be ratiometric to the excitation source), and the display. Figure 2 shows how the MSC1210 is used in our weight scale application.

For excitation of the load cell, we use a 5V power supply. The internal reference is not used; instead, the power supply is also used as the ADC reference. A 10µF capacitor tied to ground is used to filter out the noise in the power supply. This pin is tied to input pin REF IN+ to provide the ADC with a reference. The pin REF IN– is grounded.

The outputs of the load cell (OUT+, OUT–) are filtered first to remove any noise that could have been generated in the load cell itself. We found that the arrangement shown here (see Figure 2) of two resistors and one capacitor provided the best results in terms of noise performance.

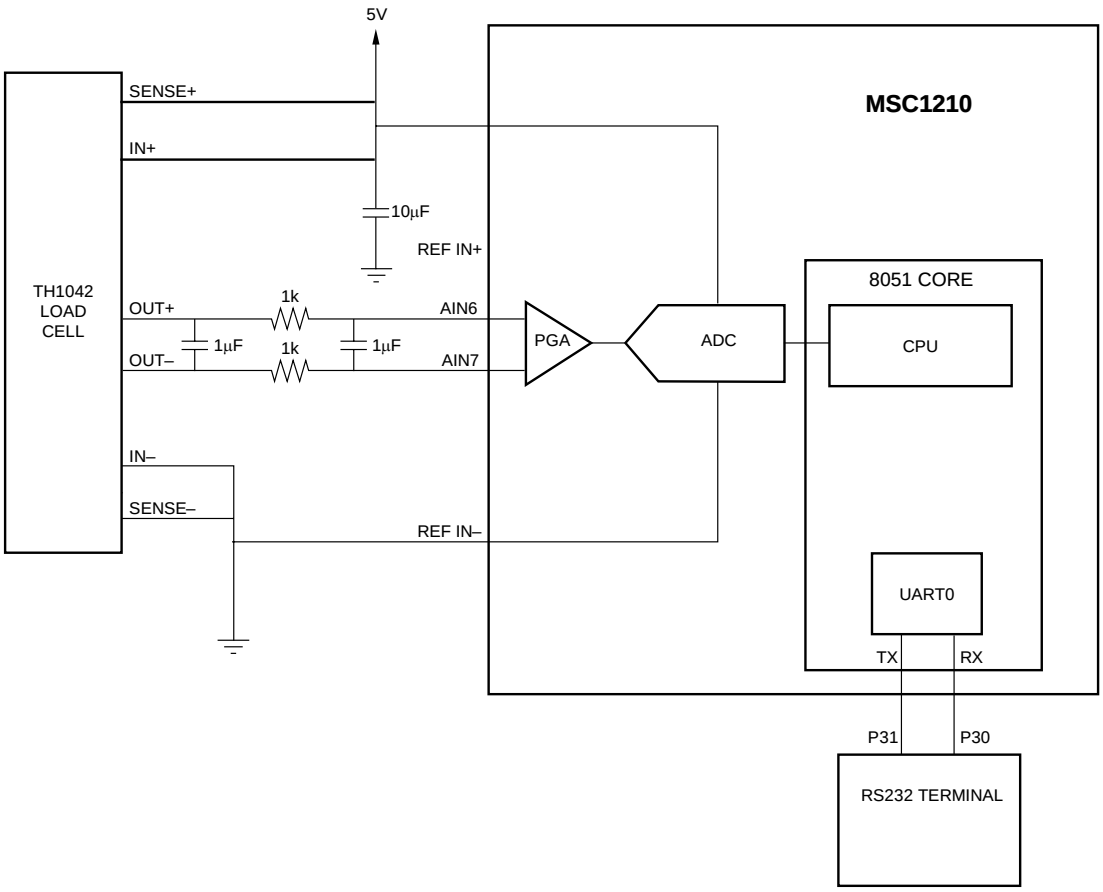

Figure 2. Load Cell Setup

3 Theory

3.1 Decimation Ratio

The ADC output data or acquisition rate is defined as:

Data Rate = f_{MOD} / (Decimation Ratio)

Where f_{MOD} = (External crystal frequency) / ((ACLK + 1) * 64)

The external crystal frequency is 11.0592MHz; thus, the ACLK register is set to 10. The decimation ratio is set to 1571 in order to get an ADC data output rate of 10Hz. That is, the ADC outputs a 24-bit number every 0.1s. The output rate of 10Hz is chosen because 60Hz is a multiple of 10Hz at which line frequency is a significant problem for low-level measurements. There also exists noise generated by the mechanical vibration of the load cell. This vibration is typically at a frequency of 20Hz. By setting the data rate to 10Hz, we effectively filter out any line frequency interference and mechanical noise from the load cell.

In countries where 50Hz is the standard line frequency, an output rate of 10Hz can also filter out any line interference. The highest output rate that can filter out the interference is 50Hz. To generate a rate of 50Hz, ACLK is set to 1 and the decimation ratio is set to 1728. Other ACLK and decimation ratio combinations exist that could generate a rate of 50Hz, but this is the setting for which the decimation ratio is highest. According to the datasheet, we can achieve a higher A/D effective number of bits (ENOB) at higher decimation ratios.

3.2 PGA and Buffer

PGA is set to 128 because we know the output rating of load cells is typically 2mV/V. For an excitation of 5V, there would be a maximum differential voltage on the order of 10mV across the output pins of the load cell. By amplifying this signal by a factor of 128, the maximum differential voltage is 1.28V. We therefore achieve smaller least-significant bit (LSB) resolution without saturating the ADC input. Here we assume the ADC reference is the same as the excitation voltage (reference voltages are discussed further on in this section). The LSB resolution in bipolar mode is defined as:

$$\text{LSB resolution} = FSR/2^{24}.$$

Where FSR is the full-scale range of the ADC.

$$FSR = (2 * \text{reference voltage}) / \text{PGA setting}$$

For a PGA of 1 and external reference = 5V, we have an FSR of 10V and LSB resolution of 600nV. For a PGA of 128, we have an FSR of 78.125mV and LSB resolution of 4.67nV. The buffer is off; according to the MSC1210 datasheet, we achieve better noise performance than when the buffer is on.

One drawback of amplifying the input signal is that the undesired noise is also amplified. Thus, the noise-performance is reduced at high PGA settings. According to the MSC1210 datasheet, we should expect an ENOB value of about 19 to 20 bits for a PGA setting of 128 and a decimation ratio of 1571. An ENOB of 19 to 20 (1 count in 524288 to 1048576) is acceptable because this value is more than four times the display resolution (count).

3.3 Offset and Gain Calibration of Load Cell

Typical load cells have rated outputs of about 2mV/V. To find an exact and accurate rated output value for this particular load cell, we need to measure the differential output when a load of known weight is applied to it. These points are then plotted on a graph and curve fitting is performed in order to find a best-fit curve or line through those points. First, we have to measure the zero offset output voltage of the load cell; that is, the differential output when no load is applied. Known weights are applied to the load cell and the output voltages are then recorded. The zero offset is subtracted from these weight readings (known as *zero offset calibration*). Figure 3 shows the plot of the results. These output values have already been zero-offset compensated.

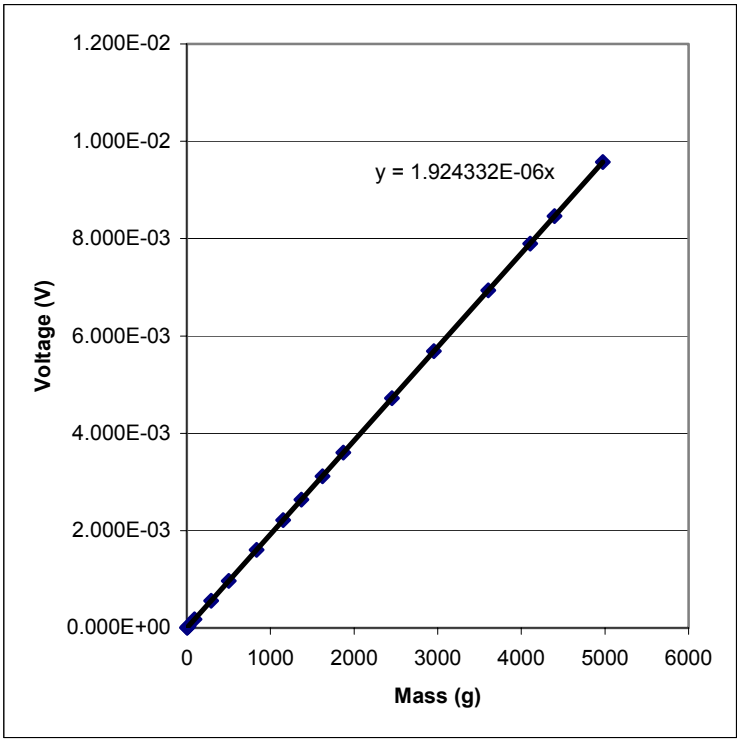

Figure 3. Output Voltage vs. Load

Curve fitting with a straight line produces a line with slope 1.9243µV/g. As a sanity check, curve fitting using a polynomial curve of degree 2 is also done. This comparison produces the curve:

$$y = -1.40e\text{-}13(x^2) + 1.925e\text{-}6(x).$$

Since the coefficient for the $\boldsymbol{x^2}$ term is seven orders of magnitude smaller than the $\boldsymbol{x}$ term, the effect of the squared term is insignificant. We conclude the load cell output behavior is linear. The rated output is (1.9243µV/g * 5000g) / 5V = 1.9243 mV/V.

3.4 Calculating the Display Resolution (Count)

The purpose of this section is to develop the theory used in determining the count or flicker of the system shown in Figure 2. Assuming an external 5V power supply is used as the ADC reference, the PGA is set to 128, and the decimation ratio is 1571, we measured an ENOB of 19.3 bits. The inputs AIN6 and AIN7 are shorted to ground.

An ENOB of 19.3 bits means 4.7 bits are lost due to undesired noise. In other words, there is $2^{4.7}$ = 26 LSB of RMS noise.

$$\text{RMS noise} = 2^{4.7} * 4.67\text{nV} = 121.38\text{nV}$$

This number represents one standard deviation (σ) from the mean of a Gaussian distribution. 66% of the voltage output measurements fall within the interval [mean – 121.38nV, mean + 121.38nV]. We would like to know the full peak-to-peak noise because that represents the maximum difference (or flicker) that one could likely see between any two output voltages. This flicker in volts is therefore proportional to the flicker in kilograms. For a Gaussian distribution, over 99.9% of the readings fall within the interval [mean – 3.3σ, mean + 3.3σ]; thus, over 99.9% of the readings are distributed over 6.6 standard deviations.

Peak-to-peak noise (volts) = 6.6 * σ = 6.6 * RMS noise = 801.108nV.

Therefore, 801.108nV is the maximum flicker one can observe between any two voltage readings. Next, we must convert the flicker in volts to the flicker in kilograms. From the fundamentals section, we know how to convert the voltage to weight:

Mass (kg) = (observed voltage) / (rated output * excitation) * capacity

So:

Flicker (kg) = (peak-to-peak noise (volts)) / (rated output * excitation) * capacity

= 801.108nV/ (1.9243mV/V * 5V) * 5kg

= 0.42 grams

There should be at most 0.42 grams of flicker between any two readings. The display resolution of the system is half of the flicker, which is +/- 0.21 grams (or 1 count in 25000). For example, if the average of a sample of weight readings is 500g, then there is a confidence level of over 99.9% that no reading is more than 500.21g and no reading is less than 499.79g.

To determine the count, we have to revisit the ENOB value. The RMS count is $2^{19.3}$ = 645474. The peak-to-peak count is therefore 645474 / 6.6 = 97799. This peak-to-peak count represents the internal count. As a rule of thumb, the display count should be four times greater than the internal count. Thus, the display count is 97799 / 4 = 25000. The display resolution is therefore +/- 0.21g, or 1 count in 25000.

3.5 Reference Voltage

This section discusses the trade-offs between using an external and internal reference. One of the advantages of using the power supply as the excitation and reference source is that the load cell output and the ADC reference vary ratiometrically as the power supply varies. Noise at the power supply is common to load cell output (or ADC input) and the reference signal. If an internal reference is used, noise at the load cell output (due to noise in the power supply) is no longer common to the reference signal. We can see that noise performance is worse compared to using the power supply as the external reference.

On the other hand, using the internal reference reduces the LSB step size. For example, for a 2.5V reference and a PGA set to 128, the LSB step size is 2.32nV, which is twice as small as using a 5V reference. The decrease in LSB step size might be more than enough to offset the difference in noise performance. To find out which choice is superior over the other, we need to go through the steps outlined in the previous section in order to calculate the resolution of this system (internal reference = 2.5V, PGA = 128). To do that, we need to measure the ENOB of the system under these conditions.

We found the ENOB of the system (inputs shorted to ground) to be about 18.3 bits (or 1 count in 323000). After performing the steps outlined in the previous section, the display resolution was +/- 0.2g or 1 count in 25000. This number is almost identical to the set up of Figure 2 (external reference = 5V power supply, PGA = 128). Thus, neither reference configuration is superior over the other.

When using the 2.5V internal reference, we found that the ENOB performance decreases for heavier loads (that is, when the magnitude of the voltage seen across the ADC inputs increases). When no load was applied, an ENOB of 18.3 bits was observed. When the maximum 5kg load was applied, an ENOB of 17.8 bits (or 1 count in 228000) was observed. As the load increases, then, internal reference noise becomes more significant and the internal resolution is no longer four times the display resolution. This behavior was not observed when the power supply was used as the reference. That is, the ENOB remains the same regardless of the magnitude of the load cell output voltage. Since this application note discusses maximizing the precision of the MSC1210, we will use the original configuration (external reference = 5V, PGA = 128) for the remainder of the report.

3.6 Filtering, Display Rate, and Settling Time

Better noise performance can be achieved using software filtering. A simple averaging algorithm can reduce the undesired noise by a factor of (sample size)$^{1/2}$. For example, if we are to perform averaging over four samples (a window size of 4), the noise effectively is reduced by a factor of two. The trade-off with averaging is speed; we need to wait for four samples before a new conversion result is produced. The display update time therefore is increased by a factor of four. The settling time, which is the time required for a new sample to propagate through the averaging window, is also quadrupled. Thus, a system utilizing software averaging of window size 4 has a settling time of 0.4s (instead of 0.1s with no averaging) and a display rate of 2.5Hz. Modifying the simple averaging algorithm can alleviate the slow display rate problem. Using a running average serves this purpose. The running average can be computed as follows:

$$Ave(n) = ave(n-1) + (sample – ave(n-1)) / N$$

Where *ave(n)* is the current average, *ave(n-1)* is the previous average, *sample* is the current ADC output, and *N* is the window size.

The running average is therefore a function of the current sample and the averages of the previous samples. Hence, a new average value is produced every time a new ADC output is generated. In other words, the display rate is the same as the data rate. The settling time, however, is ***independent*** of the averaging algorithm and can only be reduced by decreasing the window size.

3.7 Dynamically Controlled Window Sizes

Sometimes, it may be preferable to display readings in two steps. First, after a new weight is placed on the scale, a low-resolution reading is quickly displayed. This reading is accurate, but only has a small number of noise-free bits. This event corresponds to using a small averaging window. As time progresses, readings with higher and higher noise-free bits are displayed, corresponding to using bigger and bigger averaging window sizes. To the user, s/he sees digits settling one by one from the lowest resolution to the higher resolution.

In other words, we need to implement a state machine in software that dynamically controls the averaging window sizes. To determine which window size to use, the state machine must monitor successive weight readings. Figure 4 illustrates the operation of the state machine, using a 3-state state machine (S2, S4, and S8) as an example. Each state corresponds to using a window size of 2, 4, and 8 respectively. When a new weight is placed on the load cell initially, the difference between successive readings is larger than some predetermined threshold (**thres1**). The machine moves into state S2. As the weight begins to settle, the difference between successive readings become smaller and falls below **thres1**. At this point, the machine shifts into state S4. Finally, when the difference falls below some threshold (**thres2**, where **thres2** < **thres1**), the machine moves into state S8, and stays there as long as the weight remains on the load cell.

Suppose a different weight is now placed on the load cell. The difference initially becomes so large that it exceeds **thres1**. The machine shifts back to state S2, but eventually the machine will settle back into state S8 through state S4 (as long as the weight remains on the load cell).

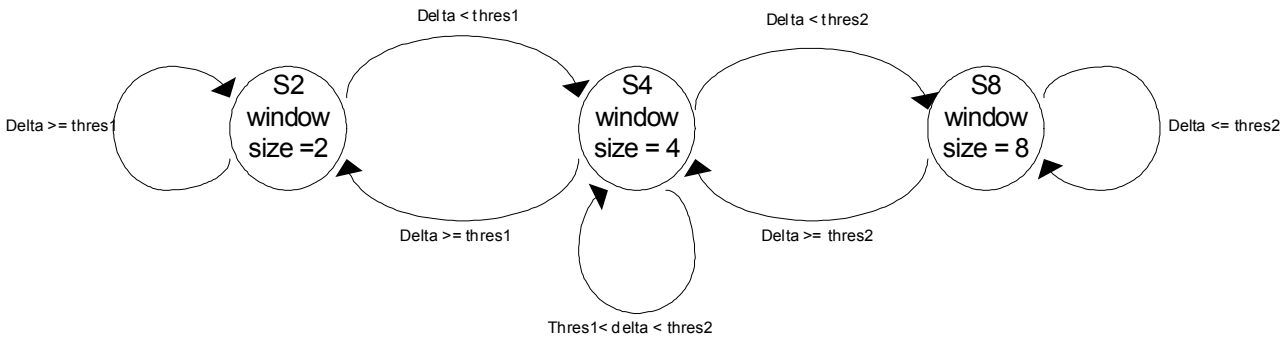

Figure 4. Example of State Machine for Dynamically Controlling Window Sizes

3.8 Temperature Drift Compensation

Temperature drift occurs in most weight scale applications because of changes in the ambient environment or because of self-heating that occurs when the bridge resistors are excited. If the load cell experiences any temperature change, it expands and induces a strain in the bridge resistors, even when no force is applied. Therefore, it is desirable to perform an offset calibration to compensate the effect caused by temperature drift when the temperature changes by more than a specified amount (even by a few degrees).

We can use the temperature sensor that is built into the MSC1210 to monitor the temperature of the environment. At the end of each offset calibration, the temperature sensor voltage is measured. This output is then converted to a temperature reading using two-point extrapolation. According to the MSC1210 datasheet, the temperature sensor voltage is 115mV at 25˚Celsius and the sensor coefficient is 375µV/C. To convert the voltage to temperature, we just perform an extrapolation of a straight line based on these two parameters. A monitor program can be set up such that at specified intervals the temperature sensor voltage is measured and converted to a temperature reading. If the current temperature differs from the post-offset calibration reading by more than a specified amount, we perform another offset calibration.

4 Results

The measurements discussed in this section are taken under the following configuration:

MSC1210 set up

- External crystal frequency: 11.0592MHz
- ACLK register = 10
- Decimation ratio = 1571 (or equivalently 10Hz)
- Buffer off
- PGA = 128
- External reference = 5V
- Offset DAC = 0
- Bipolar inputs
- System offset calibration only

Load cell parameters

- Rated output: 1.92293mV/V
- Excitation: 5V (minimum recommended voltage)
- Capacity: 5kg

4.1 No Averaging

A 4105.5g weight is placed on the load cell and 200 consecutive weight readings are recorded. The following figure is a plot of the results.

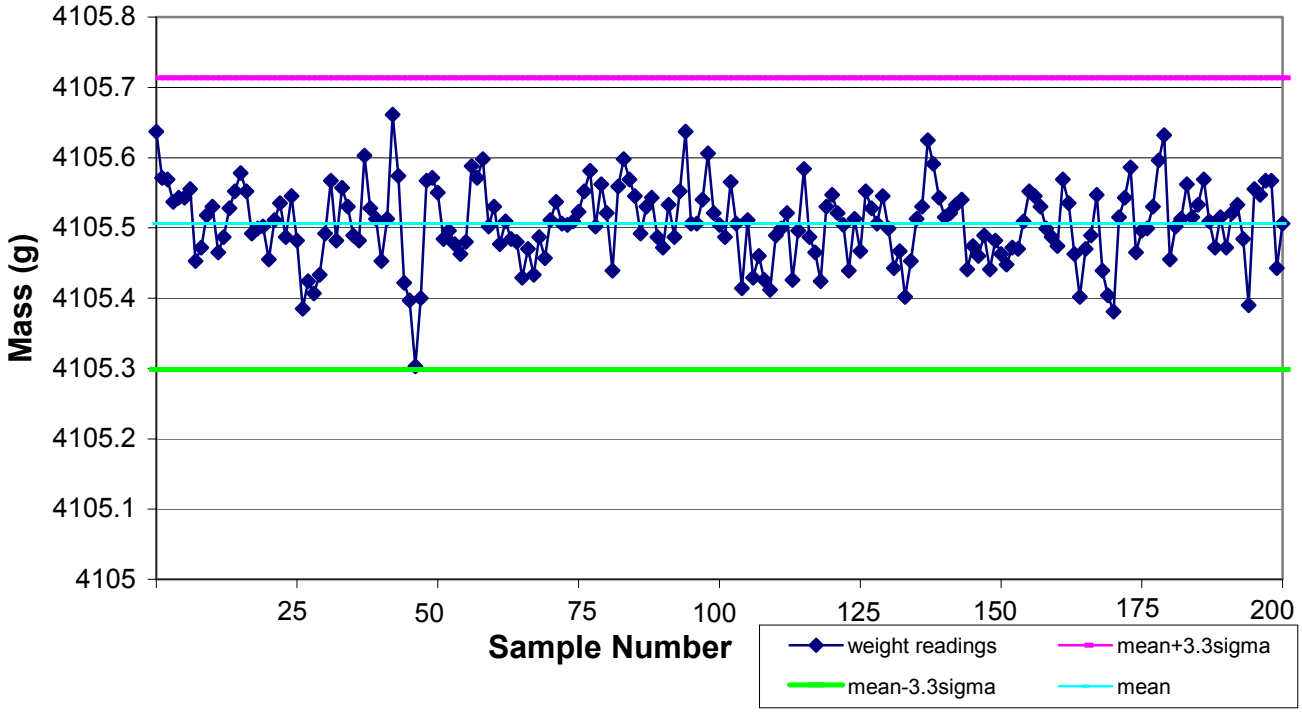

Figure 5. 200 Consecutive Readings (mean = 4105.506g, ENOB = 19.3)

The ENOB for these 200 samples is 19.3. This translates to a peak-to-peak noise of 799.83nV and a flicker of 0.42g. The display resolution is therefore +/- 0.21g (or 1 count in 25000). These numbers are consistent with those calculated in Section 3.

The middle horizontal line is the mean of these 200 measurements. The top horizontal line is the upper limit; that is, there is a confidence level of over 99.9% that no reading is more than 4105.506 + 0.21g. The bottom horizontal line is the lower limit; that is, there is a confidence level of over 99.9% that no reading is less than 4105.506 – 0.21g. The above figure confirms that out of these 200 measurements, all of them fall within the upper and lower limits.

Figure 6 is a histogram plot of the same data in Figure 5 (bin width = 0.0358g).

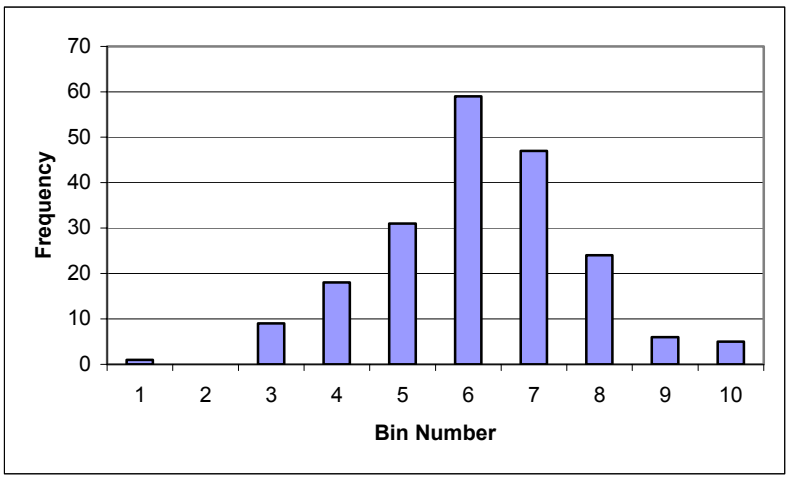

Figure 6. Histogram for Data in Figure 5

The maximum value is 4105.661g and the minimum value is 4105.303g. These 200 samples are distributed over 10 bins. The bin width is (4105.661 – 4105.303)/10 = 0.0358g. The intervals for each bin are listed in Table 1.

Table 1. Bin Number and Interval

Bin Number	Interval (g)
1	4105.303 – 4105.339
2	4105.339 – 4105.375
3	4105.375 – 4105.410
4	4105.410 – 4105.446
5	4105.446 – 4105.482
6	4105.482 – 4105.518
7	4105.518 – 4105.554
8	4105.554 – 4105.589
9	4105.589 – 4105.625
10	4105.625 – 4105.661

The average value is located in bin 6.

4.2 Running Average

The following plot shows 200 weight readings, using a running average of window size 8.

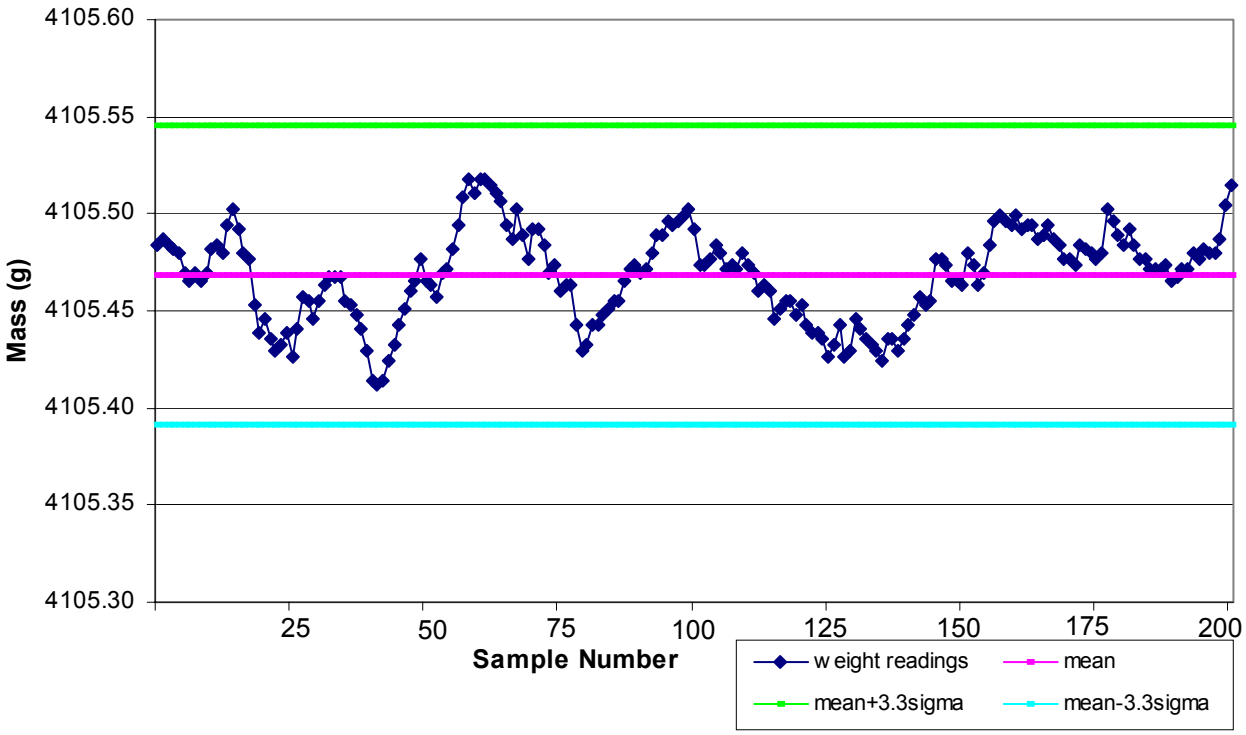

Figure 7. 200 Consecutive Readings, Window Size = 8 (mean = 4105.469g, ENOB = 20.7)

The ENOB for these 200 samples is 20.7. This translates to a peak-to-peak noise of 296.38nV and a flicker of 0.15g. The display resolution is therefore +/- 0.08g (or 1 count in 62500). The middle horizontal line is the mean of these 200 measurements. The top horizontal line is the upper limit, and the bottom horizontal line is the lower limit. We expect the data show that there is a confidence level of over 99.9% that no reading is more than 4105.469 + 0.08g or less than 4105.469 – 0.08g. Fig. 7 confirms that out of these 200 measurements, all of them fall within the upper and lower limits.

Figure 8 is a histogram plot of the same data in Figure 7 (bin width = 0.01g).

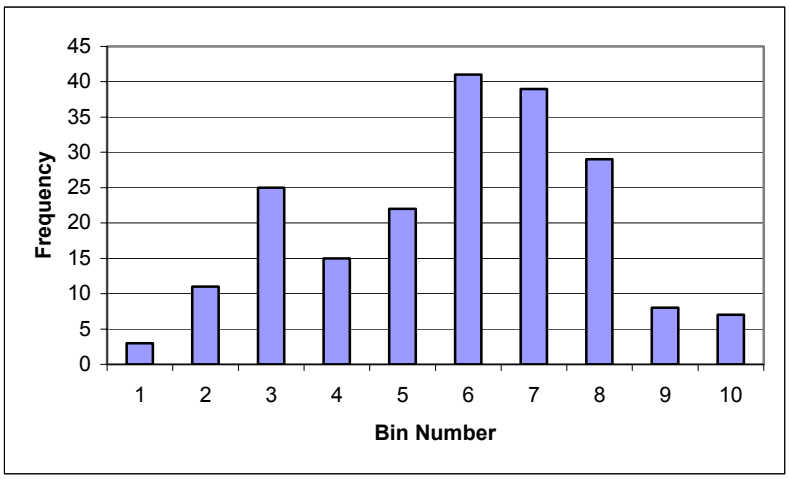

Figure 8. Histogram for Data in Figure 7

The maximum value is 4105.518g and the minimum value is 4105.412g. These 200 samples are distributed over 10 bins. The bin width is (4105.518 – 4105.412)/10 = 0.01g. The intervals for each bin are listed in Table 2.

Table 2. Bin Number and Interval

Bin Number	Interval (g)
1	4105.412 – 4105.423
2	4105.423 – 4105.433
3	4105.433 – 4105.444
4	4105.444 – 4105.454
5	4105.454 – 4105.465
6	4105.465 – 4105.476
7	4105.476 – 4105.486
8	4105.486 – 4105.497
9	4105.497 – 4105.507
10	4105.507 – 4105.518

The average value is located in bin 6.

5 Summary

The results of the two tests are summarized in Table 3. By averaging over eight samples, we effectively increased the resolution of the weight scale by a factor greater than 2. The settling time, however, is increased eight times, to approximately 1s. We feel this trade-off is acceptable in weight scale applications because most users do not need to apply a different weight on the scale every second.

Table 3. Summary of Results

Parameter	No Averaging	Moving Average
LSB	4.67nV	4.67nV
ENOB	19.3 bits	20.7 bits
RMS noise (one standard deviation)	121.38nV	44.75nV
Peak-to-peak noise (6.6 standard deviation)	801.108nV	295.38nV
Flicker	0.42g	0.15g
Display Resolution	+/- 0.21g	+/- 0.08g
Count	1 in 25000	1 in 65000

Example Code

In this section, we include the sample C program ***conv.c*** and assembly program ***adc_sub.a51***. The latter file contains subroutines, which are called out by the C program. This sample program generates the results that are presented in the previous section. A listing file is also included to give the user an estimate of the code and data size.

```
//conv.c
#include "MSC1210.H"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
extern void autobaud(void);
extern unsigned long unipolar(void);
extern unsigned long bipolar(void);
extern signed long deltasigma(void);
extern signed long adc_win(unsigned char);
extern char rx_byte(void);
extern void putcr(void);
#define rated_output 1.92433e-3
#define capacity 5000
#define excitation 5

void adc_mave(unsigned int n, float k, float conv_factor)
{
        unsigned char xdata *winptr;
        long int idata adcsum;
        long int sw_offset_mave=0;
        unsigned int fill_ptr=1, mod_ptr=0 ;
        adcsum=0; deltasigma(); // dispose one conversion
        //acquire the sw offset,
        sw_offset_mave = deltasigma();
        printf(" Volts\t \t grams\n");
        while (RI==0) {
                        adcsum=adcsum+adc_win(mod_ptr);
                        printf("%1.8f\t %f\n", (float)((adcsum / fill_ptr)-
sw_offset_mave)*k, (float)((adcsum / fill_ptr)-sw_offset_mave)*k*conv_factor);
                        if (fill_ptr==n) fill_ptr=n; else fill_ptr++;
                        if (mod_ptr==(n-1)) mod_ptr=0; else mod_ptr++;
        }
        rx_byte();    // Flush serial byte
}

void main(void)
{
        signed long int xdata adres[200];
        signed long int dummy;
        signed long int sw_offset;
        int i;
        int j;
        char s;
        float ave,sd,n,k, conv_factor;
        CKCON &= 0xf8; // 0 MOVX cycle stretch
        k = 5/(pow(2,23)*128); //1 LSB in volts
        //Timer Setup
        ONEUSEC= 10;               // 11.0592MHz Clock !!!!!
        ONEMSL=0x32;
        ONEMSH=0x2B;
        HUNDMS=0x63;
        PWRMGT &= 0xf7;            //turn on adc

        autobaud(); RI=0;
        printf("MSC1210 load cell Test\n");
        printf("1) Acquire 200 samples and perform stats analysis\n");
        printf("2) Moving average dump\n");
        scanf("%c",&s);
```

```
        /* Setup ADC */
        P1DDRH |= 0xc0;
        ADCLK = 10;  //11.0592MHz => modclk = 15.6kHz
        ADMUX = 0x067;              // Select AIN6-AIN7
        ADCON0 = 0x07;  // Vref off , Buff off, BOD off, PGA=128
        ADCON1 = 0X00;  // bipolar
        ADCON2 = 0x23;              // decimation ratio = 1571 => data rate = 10Hz
        ADCON3 = 0x06;
        ADCON1 = 0x04;              // bipolar, auto, system calibration, offset
        PGASHFT = 0;
        //wait for the calibration to take place
        printf ("\n\n System Calibrating....\n");
        for (i=0;i<5;i++){ // dump 5 conversions
                        while(!(PIREG & 0x20)){}
                        dummy=bipolar();
        }
        while(!(PIREG & 0x20)){}
                        sw_offset=bipolar(); //get the zero load offset after calibration
        if (s=='1') { //s=='1'
                        ADMUX = 0x67; //AIN6-AIN7
                        conv_factor = capacity/(rated_output*excitation);
                        while (1){
                                        printf("voltage -> mass(g)\n");
                                        for (i=0;i<200;i++){ // acquire 200 points
                                                        while(!(PIREG & 0x20)){}
                                                        adres[i]=bipolar()-sw_offset;
//subtract zero load offset from reading
                                                        printf ("%1.8f\t %f\n",
(float)adres[i]*k, (float)adres[i]*k*conv_factor);
                                        }
                                        ave=0; n=0; sd=0; j=0;
                                        for (i=0;i<200;i++){
                                                        ave+= adres[i];            n++;

                                        }
                                        ave= ave/n;
                                        for (i=0;i<200;i++){
                                        sd += pow(adres[i]-ave,2);
                                        }
                                        sd= sqrt(sd/n);
                                        printf("\n**** In LSB:
SD=%3.2f\tMean(volts)=%1.8f\tN=%4d\tENOB=%3.2F****\n",sd,ave*k,(int)n, 24-
log(sd)/log(2));
                        } //while(1)
        } //if s=='1'
        else {
                        adc_mave(8,k, conv_factor);
        } // else

}// main
```

```
//adc_sub.a51
$NOMOD51
$include (reg1210.inc)
PUBLIC  unipolar, bipolar, read_sum_regs
PUBLIC  deltasigma, _adc_win
adc_sub             SEGMENT  CODE
        RSEG  adc_sub

;;;;;;;;;;;;;;;;;;;;
; moving average ADC conversion and window calc
; long int adc_win(unsigned char i);
; return adc - win[i] and update win[i] with adc
; i point to long int xdata array
_adc_win:
        mov           a, r7         ; r7*4-> DPTR
        swap          a             ;   r7.7~6 -> dph.1~0
        rr            a
        rr            a
        anl           a,#03h
        mov           dph,a
        mov           a,r7          ;  r7.5~0,0,0 -> dpl
        rl            a
        rl            a
        anl           a,#0fch
        mov           dpl,a
        call          deltasigma    ; adc -> r4~7
        movx          a,@dptr       ; r0~3 <- win[i], win[i] <- adc
        mov           r0,a
        mov           a, r4
        movx          @dptr,a
        inc           dptr
        movx          a,@dptr
        mov           r1,a
        mov           a, r5
        movx          @dptr,a
        inc           dptr
        movx          a,@dptr
        mov           r2,a
        mov           a, r6
        movx          @dptr,a
        inc           dptr
        movx          a,@dptr
        mov           r3,a
        mov           a, r7
        movx          @dptr,a
adcsub:
        clr           c       ; r4~7 <- r4~7(adc) - r0~3 (old win[i])
        mov           a,r7
        subb          a,r3
        mov           r7,a
        mov           a,r6
        subb          a,r2
        mov           r6,a
        mov           a,r5
        subb          a,r1
        mov           r5,a
        mov           a,r4
        subb          a,r0
        mov           r4,a
        ret
```

```
;;;;;;;;;;;;;;;;;;;;
; signed long deltasigma(void)
; return the 3 byte adres to R4567 (MSB~LSB)
; return signed long int with sign-extend R4 for bipolar
;        and stuff 0 on R4 for unipolar
; ADCON0[7]=OSE
;P1.7 =0: Conversion with reversed ADMUX
;            +-------------- reverse ADMUX (Last-ADC)/2
;            |     +-------- default ADMUX
;            |     |    +--- (ADC-Last)/2
;            V     V    V
;P1.7 -----|________|--------|_______
;
;
;
;
deltasigma:
       mov          a,ADCON0
       rlc          a
       jnc          ds_once          ; if AZ=0 then ds once
       jb           P1.7, ds_nosw    ; default ADMUX setting
       mov          a,ADMUX
       swap         a
       mov          ADMUX,a
ds_nosw:
       call         ds_once
       call         ds_once
       call         ds_once
       call         ds_once
       call         ds_once          ; r4~7 <- new adc
       mov          r3,DPL1          ; r0~3 <- last adc
       mov          r2,DPH1
       mov          r1,BPL
       mov          r0,BPH
       mov          DPL1,r7          ; last <- new
       mov          DPH1,r6
       mov          BPL,r5
       mov          BPH,r4
       jb           P1.7,ds_adcsub   ; If set: (ADC-Last)/2 else (Last-ADC)/2
       mov          a,r7             ; XCH (r4~7, r0~3)
       xch          a,r3
       xch          a,r7
       mov          a,r6
       xch          a,r2
       xch          a,r6
       mov          a,r5
       xch          a,r1
       xch          a,r5
       mov          a,r4
       xch          a,r0
       xch          a,r4
ds_adcsub:
       call         adcsub           ; adcsub: r4~7 -= r0~3 ;
       mov          a,r4             ; r4~7 /= 2; 2's comp div
       rlc          a                        ;   c has sign
       mov          a,r4             ;   (c,r4)>>1->c
       rrc     a
       mov          r4,a
       mov          a,r5             ;   (c,r5)>>1->c
       rrc     a
```

```
        mov         r5,a
        mov         a,r6          ;   (c,r6)>>1->c
        rrc     a
        mov         r6,a
        mov         a,r7          ;   (c,r7)>>1->c
        rrc     a
        mov         r7,a
        jb          P1.7, ds_done
        mov         a,ADMUX
        swap        a
        mov         ADMUX,a
ds_done:
        cpl         P1.7
        ret
ds_once:
        mov         a,AIE
        jnb         acc.5,ds_once
        mov         r4,#0
        mov         r5,adresh
        mov         r6,adresm
        mov         r7,adresl
        mov         a,ADCON1
        jb          acc.6, ds_unipolar        ; stuffed zero for unipolar
        mov         a,r5
        jnb         acc.7,ds_positive         ; sign-extend for bipolar
        mov         r4,#0ffh
ds_positive:
ds_unipolar:
        cpl         P1.4
        ret


;;;;;;;;;;;;;;;;;;;;;
; unsigned long unipolar(void)
; return the 3 byte adres to R4567 (MSB~LSB)
; unsigned long int with R4=0
unipolar:
        mov         r4,#0
        mov         r5,adresh
        mov         r6,adresm
        mov         r7,adresl
        ret
;;;;;;;;;;;;;;;;;;;;;
; signed long bipolar(void)
; return the 3 byte adres to R4567 (MSB~LSB)
; return signed long int with sign extendsion on R4
bipolar:
        mov         r4,#0
        mov         a,adresh
        mov         r5,a
        mov         r6,adresm
        mov         r7,adresl
        jnb         acc.7,positive
        mov         r4,#0ffh
positive:
        ret
;;;;;;;;;;;;;;;;;;;;;
; signed long read_sum_regs(void)
; return the 4 byte sumr to R4567 (MSB~LSB)
; return signed long int, sign extension done by hardware
read_sum_regs:
```

```
        mov           r4, SUMR3;
        mov           r5, SUMR2;
        mov           r6, SUMR1;
        mov           r7, SUMR0;
        ret
        end
```

```
//conv.lst
...
<snip>
...
MODULE INFORMATION:   STATIC OVERLAYABLE
   CODE SIZE        =   1158    ----
   CONSTANT SIZE    =    240    ----
   XDATA SIZE       =   ----     851
   PDATA SIZE       =   ----    ----
   DATA SIZE        =   ----    ----
   IDATA SIZE       =   ----       4
   BIT SIZE         =   ----    ----
END OF MODULE INFORMATION.


C51 COMPILATION COMPLETE.  0 WARNING(S),  0 ERROR(S)
```

References

1. *MSC1210Yx Datasheet (SBAS203A)*

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265