

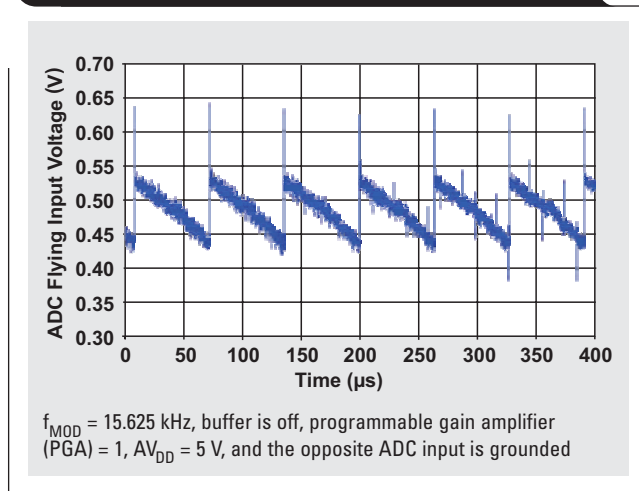
Supply voltage measurement and ADC PSRR improvement in MSC12xx devices

By Michael Gurevich (Email: gurevich_michael@ti.com)
Systems Engineer

Introduction

This article describes a simple method to measure the analog supply voltage in Texas Instruments MSC12xx devices with one of the unconnected ADC inputs of the $\Delta\Sigma$ ADCs. The ability to measure the supply's voltage change permits the ADC gain to be adjusted, which leads to a better ADC power supply rejection ratio (PSRR). This method is applicable to the MSC1210, MSC1211/1212/1213/1214, MSC1200/1201/1202, and ADS1216 families.

Figure 1. ADC flying input voltage taken with 10-M Ω , 8-pF probe



During the $\Delta\Sigma$ ADC conversion cycle, the input signal is sampled on the internal capacitors at the modulation frequency (f_{MOD}). Switching capacitor techniques are then used to compare the input voltage to the integrator voltage, eventually leading to the ADC result. Reference 1 provides a detailed description of how the ADC input circuits work. What is important here is that when the next sampling occurs, the sampling capacitor is not fully discharged but has a voltage proportional to the analog supply voltage, $A_{V_{DD}}$. This weak voltage can be seen with a high-impedance (>10 M Ω) voltmeter or oscilloscope. When the oscilloscope probe is connected to the ADC input selected by the internal program and the ADC is running with the internal buffer disabled, the oscilloscope shows a voltage in the range of 0.3 to 0.7 V as shown in Figure 1.

Measuring the flying ADC inputs

If both ADC differential inputs are disconnected from the external circuits, then both nodes will have the same potential and the measured differential voltage will be zero. But if one of the inputs is grounded, the conversion result will no longer be zero and will remain stable until the analog power supply is changed. These measurements can be done only if the ADC input buffer is disabled. If the buffer is on, the high-impedance buffer inputs fluctuate due to small-pin leakages, which force the buffer outputs to follow and suppress the voltage coming from the ADC inputs. The next natural question is: What affects the ADC readings in this condition?

What affects the flying input readings?

Experiments were performed to determine whether the ADC results are affected by any of the following factors:

- ADC modulator frequency
- ADC input sampling frequency (sampling frequency changes when the programmable gain amplifier [PGA] setting is changed)
- Decimation frequency
- Chosen ADC inputs
- Device clocking frequency
- Reference voltage and reference voltage common mode
- Device temperature
- Proximity of clocks or other noise sources

In all cases it was found that if one ADC input is grounded and the opposite input is flying, the ADC readings are stable and don't show any dependence on the preceding parameters. But if the flying input has some resistance or capacitance connected to it, the ADC readings change because the additional dc and ac paths affect the flying input voltage (see Figure 1). The ADC reading also has a minor dependence on the digital supply voltage. When the digital supply changes from 5 V to 3 V, the ADC readings change less than 1%; therefore, the ADC readings mainly depend on the analog supply voltage, $A_{V_{DD}}$. A stable ADC reading does not mean that flying input voltage measured with an external voltmeter is constant; it means that all these changes are self-compensated during conversion and lead to the stable ADC result.

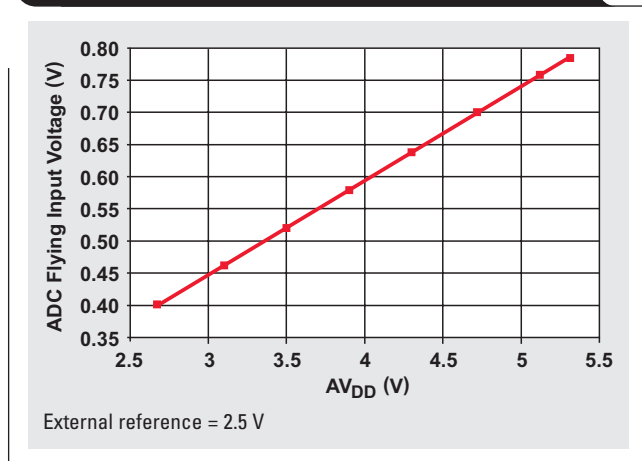
ADC flying input readings and analog power supply voltage

When the dependence of the ADC flying input readings on the analog supply voltage was investigated, it was found that in all MSC12xx devices, the ADC readings have a strictly linear relationship to the analog supply voltage:

$$V = \frac{AV_{DD}}{KS}, \quad (1)$$

where V is the ADC flying input reading and KS is a proportional coefficient. KS varies slightly from device to device and between device types and packages. The average KS value is around 6.9. Figure 2 shows the ADC flying input readings for the MSC1210.

Figure 2. MSC1210 ADC flying input readings from AV_{DD} supply



Equation 2 shows the correlation of the flying voltage on an unused ADC input to the analog supply voltage, AV_{DD} :

$$AV_{DD} = KS \times V \quad (2)$$

In Table 1, the KS value and its variations are determined for different MSC12xx devices. As you can see, the KS value slightly depends on the device type, as some devices have different internal layouts, different packages, and pins with different parasitic capacitances. Different device packages are also forced to use different types of testing boards.

It is necessary to understand that flying input measurements are absolute-value measurements, and therefore the precision of V_{REF} , ADC offset and gain calibration plays some role in the results.

Table 1. Value of coefficient KS and its variations for different devices*

DEVICE	NUMBER OF DEVICES TESTED	KS (AVERAGE SLOPE)	KS VARIATION ($\pm\%$)
MSC1200	35	6.818	3.6
MSC1201	5	7.014	1.1
MSC1210	30	6.911	2.4
MSC1211	10	6.843	2.1

* KS is determined with external reference voltage.

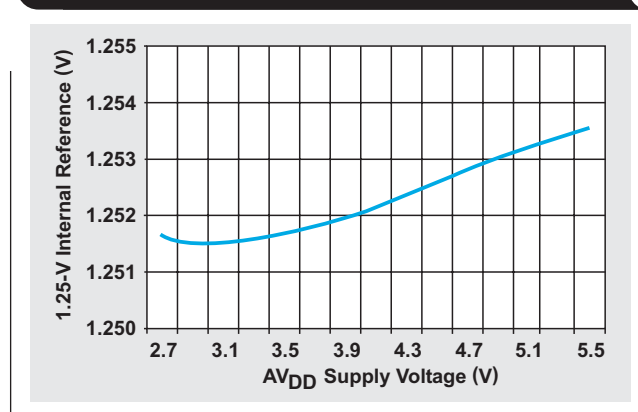
Table 1 also shows that KS variation between devices is around $\pm 2.5\%$. Therefore, the KS value that determines AV_{DD} for one user board can apply to all other similar boards with $\pm 2.5\%$ precision. Usually there is no need to track the supply voltage with a precision better than $\pm 5\%$, and 2.5% is usually sufficient. The KS values shown in Table 1 are a good starting point to investigate the KS magnitude for the user board and settings. If an internal reference is used, remember that it has some supply dependence, which can affect the KS value.

The flying input voltage can also be used to detect whether the ADC inputs are still connected to the external signal source.

Compensating for changes in internal reference voltage

The typical ADC PSRR value for MSC12xx devices is around 85 dB. However, the internal reference PSRR is much lower, around 60 dB, which will restrict the whole system PSRR if an internal reference is used. Usually the internal reference has a nearly linear dependence on the analog supply voltage (see Figure 3).

Figure 3. Typical MSC1210 1.25-V internal reference dependence on analog supply voltage



This linear dependence allows us to estimate the internal reference voltage change from the supply voltage change. Since the ADC flying input gives us a good tool to measure the supply voltage change, we can calculate the new ADC gain value to compensate the ADC reference shift:

$$GAIN_n = GAIN_c [1 + (SPLY_n - SPLY_c) \times KR], \quad (3)$$

where

$GAIN_c$ is the data value loaded in the ADC gain calibration register (GCR) during ADC self-calibration with a stable supply voltage ($AV_{DD} = V_c$);

$SPLY_c$ is the ADC flying input data value after ADC self-calibration with $AV_{DD} = V_c$;

$SPLY_n$ is the ADC flying input data value with the new supply voltage, V_n (the GCR still has the $GAIN_c$ value);

$GAIN_n$ is a new ADC gain for the GCR for supply voltage V_n ; and

KR is a proportional coefficient, defined later in Equation 4.

The sequence of actions is this: After a reset, when the power supply is stabilized at V_c and the device is warmed up, the program makes the ADC self-calibrate and copies the GCR value to the variable $GAIN_c$. Then the ADC flying input, $SPLY_c$, is measured; and the user program starts its routines. When the supply voltage has been changed to V_n , the program measures the new flying input voltage, $SPLY_n$. If it is different from the stored $SPLY_c$ value, the program calculates the new $GAIN_n$ value with Equation 3 and loads it to the GCR. Since the program stores the original $GAIN_c$ and $SPLY_c$ values, this procedure can be repeated many times.

There is an unknown constant, KR, which can be derived from Equation 3:

$$KR = \frac{\frac{GAIN_n - 1}{GAIN_c}}{SPLY_n - SPLY_c} \quad (4)$$

Even with the plot in Figure 3, it is possible to make only a coarse estimation for the KR value because of variations from device to device. Also affecting the KR value is the ADC PSRR; and very often the same supply used for AV_{DD} is used to power the signal external buffer, which also has some PSRR, and so on. Much better results can be achieved if a stable external control signal, independent of AV_{DD} supply changes, is used to determine the working KR value. Under these conditions, Equation 4 can be converted to Equation 5:

$$KR = \frac{\frac{CNTR_c - 1}{CNTR_t}}{SPLY_t - SPLY_c}, \quad (5)$$

where

$CNTR_c$ is the ADC control signal data value after ADC calibration with $AV_{DD} = V_c$;

$CNTR_t$ is the control signal data value after AV_{DD} has been changed to some new voltage V_t ; and

$SPLY_t$ is the ADC flying input data value for V_t .

This approach makes it fast and easy to find the precise value for the KR coefficient, which itself includes all device features and environment variations. For better KR precision, it is recommended that $V_c - V_t$ be at least 1 V and that the control signal value be bigger than $REF/2$. For MSC1200 parts, the typical KR value is 1.076×10^{-9} .

In addition to ADC supply compensation, this approach can be used to compensate ADC errors due to other factors if there is a linear dependence between ADC readings and the other factors. System temperature is a good example of such ADC gain compensation.

Table 2 shows the ADC PSRR with a 1-V input signal and the 1.25-V internal reference for the MSC1200 and MSC1210. AV_{DD} was changed from 5 V to 4 V during the PSRR measurements. To find the exact KR value with Equation 5, a 1-V control signal was used, and AV_{DD} was changed from 5 V to 3 V.

Table 2. ADC PSRR with and without ADC gain correction

DEVICE NUMBER	ADC PSRR WITHOUT GAIN CORRECTION (dB)	ADC PSRR AFTER CORRECTION (dB)	PSRR DIFFERENCE (dB)
MSC1200 #1	51.83	72.76	20.93
MSC1200 #2	52.56	72.58	20.02
MSC1200 #3	53.88	81.42	27.54
MSC1210 #1	63.53	80.91	17.38
MSC1210 #2	68.63	93.97	25.34
MSC1210 #3	66.74	79.16	12.43

USupply program

The "C" program beginning on page 9 provides a way to verify the precision of the method discussed and to find the constants KS and KR. Communication with the program is via the device and computer serial port. By typing one-character commands and comparing the program response with real voltages applied to the device, the user can estimate the degree of precision. If necessary the program can be easily adjusted to the desired clock frequency, ADC decimation value, average number of ADC samples, reference voltage used, and input pins used. To work with the USupply program it is necessary to have the adjustable power supply (in the 2.5- to 6-V range), the source of stable dc voltage (in the 1- to 2.5-V range), and a dc voltmeter.

The ADC input pin AINCOM should be grounded, AIN4 should be flying (not connected), and AIN5 should be connected to the control signal (dc voltage, 1 V recommended). The default program clocking frequency is 1.8432 MHz. The user can switch the program to another clocking frequency but should remember that, for $DV_{DD} = 2.7$ V, the maximum clock frequency is less than 12 MHz. The program uses a 1.25-V internal reference to give the user more room to change the supply voltage.

The default version of the program works with MSC1200/1201 devices. For the MSC1210/1211 devices, the code in the `wait()` and `ADC_Avrg()` routines should be altered according to the comments in the routines.

After reset, the program uses the default values for the KR and KS constants. By manipulating program commands and the power supply, the user will be able to find the KR and KS values that best match the device environment.

For the ADC result averaging, the program uses the summation register. Constants are defined to simplify the summation register setup. The number of averages is chosen by defining the `AVRG` constant to be equivalent to either `AVRG4`, `AVRG8`, `AVRG16`, or `AVRG256`, which represents averaging of 4, 8, 16, or 256 points, respectively.

The program code size is slightly less than 8 Kbytes, which means that devices with Flash memory version Y3, Y4, or Y5 should be used.

Routines

- `wait(cycles)`: Delays the number of ADC data cycles.
- `ADC_Avrg()`: Averages the ADC result as specified in the `AVRG` constant with the ADC summation register. The subroutine returns the measured voltage as a float variable. The averaged ADC code is located in the global `u.lng` variable. The ADC is used in unipolar mode.
- `ADCsupply()`: Measures the flying input `AIN4` voltage and calculates the AV_{DD} supply voltage with the `KS` constant routine. Reports both voltages to the PC and returns to the main program ADC code averaging for flying input measurements.
- `ADCcntrl()`: Measures the control voltage between inputs `AINCOM` and `AIN5`. Converts it to volts and reports to the PC. Returns to the main program averaged ADC code.

Main program

After reset, the `USupply` program expects the “CR” character from the PC serial port and, after receiving it, performs ADC self-offset and gain calibration. It then shows a greeting message on the screen. When the user types “H” for “Help,” the program displays the following options:

```
Enter
'S' To check Supply voltage
'M' To Measure control signal
'A' To Adjust ADC gain
'C' ADC self offset and gain Calibration
'F' Find ADC gain cor.coef.
'V' Find supply Voltage coef.
```

Entering “S” will cause the program to measure the supply voltage using the flying input method and the `KS` coefficient. The user can compare the result with the real AV_{DD} voltage. If option “V” was not used before, the program uses Equation 2 with a default KS_z value.

Entering “M” will cause the program to measure the control signal on pin `AIN5` and report its current voltage to the PC.

Entering “A” will cause the program to adjust the ADC GCR value using the `KR` coefficient. The magnitude of the adjustment depends on how much the supply voltage AV_{DD} has been changed from the supply level V_c , which existed during ADC calibration. This option should be used after ADC calibration is done and AV_{DD} has changed to V_n . The program reports to the PC the old ADC gain, the adjustment coefficient, and the new ADC gain. If option “F” was not used before, the program performs an adjustment with a default KR_z value.

Entering “C” will cause the program to perform ADC self-offset and gain calibration. The program reports the

ADC GCR value to the PC, copies it to $GAIN_c$, and measures and stores the ADC flying input and control signal readings.

Entering “F” will cause the program to find the `KR` coefficient with supply voltage V_n . For this purpose the program uses the control signal readings with supply voltages V_c and V_n . The new `KR` value is reported to the PC, and with it the ADC gain adjustment (option “A”) can then be more precisely performed.

Entering “V” will cause the program to find the supply proportional coefficient, `KS`. The program asks the user to enter the current value of AV_{DD} in volts, then it measures the flying input voltage and calculates the precise value for `KS`. The updated `KS` value is also reported to the PC and then used with subsequent supply voltage measurements.

Conclusion

The suggested method for measuring analog supply voltage in `MSC12xx` devices without additional external parts can be important in many battery applications. When the device current power supply voltage is known, the method allows adjustment of the ADC gain, which provides better ADC `PSRR`, especially in the cases with an internal ADC reference. The same method can be used to compensate system gain change due to temperature variations. The `USupply` program enables the user to verify the method and precision of measurements for the desired boards and settings and to find the `KS` and `KR` values.

References

For more information related to this article, you can download an Acrobat Reader file at www-s.ti.com/sc/techlit/litnumber and replace “*litnumber*” with the **TI Lit. #** for the materials listed below.

Document Title	TI Lit. #
1. Joseph Wu, “Input Currents for High-Resolution ADCs,” Application Reportsbaa090
2. Michael Gurevich, “ADC Offset in <code>MSC12xx</code> Devices,” Application Reportsbaa097
3. Michael Gurevich, “ADC Gain Calibration—Extending the ADC Input Range in <code>MSC12xx</code> Devices,” Application Reportsbaa107

Related Web sites

analog.ti.com

www.ti.com/sc/device/partnumber

Replace *partnumber* with `ADS1216`, `MSC1200Y2`, `MSC1201Y2`, `MSC1202Y2`, `MSC1210Y2`, `MSC1211Y2`, `MSC1212Y2`, `MSC1213Y2` or `MSC1214Y2`

USupply program

```

/* 1.08 ##### Usupply #####
Texas Instruments, G.M. September 2004; MSC1200/MSC1210;
To work with MSC1210/1211 parts, the code should be changed in the
<ADC_Avrg> and <Wait> subroutines (see comments there).
The program demonstrates part AVdd measurement using the ADC flying input.
It also adjusts ADC gain when AVdd is changed.
Internal 1.25 V reference. ADC buffer off. Unipolar ADC mode.
Communication through UART0. After reset hit the 'Enter' key.
----- Pins -----
AINCOM      Grounded
AIN5        Control signal input, always positive, bigger then REF/2
AIN4        Flying input, used to measure AVDD
----- Commands -----
"S" Measure supply voltage with flying AIN4
"M" ADC converts the control signal at AIN5
"A" Adjust the ADC gain when AVdd is changed
"C" Calibrate ADC
"F" Find KR, the ADC correction coefficient, when AVdd is changed
"V" Find KS, the supply voltage conversion coefficient
"H" Print help
#####*/
#include <MSC1210.H>
#include <stdio.h>

#define FCLOCK 2 //External clock freq. is 1.842 MHz
#define DCMTN 781 //ADC decimation ratio, 781==> fdata = 20.0 Hz
#define LSB 1.25/ 16777216 //LSB for 1.25-V reference, ADC unipolar mode
#define KSz 7.0 //Supply voltage coefficient (6.621 for MSC1200), default value
#define KRz 1.076E-9 //ADC gain correction coefficient, default value for MSC1200/1201
// #define KRz 2.28E-10 //ADC gain correction coefficient, default value for MSC1210/1211

//----- Constants for summation register -----
#define AVRG2 0xC0 //Summator average 2 ADC samples
#define AVRG4 0xC9 //Summator average 4 ADC samples
#define AVRG8 0xD2 //Summator average 8 ADC samples
#define AVRG16 0xDB //Summator average 16 ADC samples
#define AVRG256 0xFF //Summator average 256 ADC samples
#define AVRG AVRG8 //Constant used for ADC averaging

//----- Subroutines -----
float ADC_Avrg(void);
void Wait(char );
unsigned long ADCcntrl(void);
unsigned long ADCsupply(void);
extern void autobaud(void); //Boot ROM subroutine

float KR; //Coefficient to adjust ADC gain when supply is changed
float KS; //Coefficient to calculate supply voltage when flying input is measured

#####
//u is unsigned long variable with separate byte addressing
//Long = u.lng; Byte= u.bt.b4, ... , u.bt.b1
#####
union { signed long lng;
    struct {
        unsigned char b4 ; //MSB
        unsigned char b3 ;
        unsigned char b2 ;
        unsigned char b1 ; //LSB
    } bt ;
} u;
//-----

```

Continued on next page

USupply program (Continued)

```

##### Main program #####
#####
void main(void)
{
    char k;
    unsigned long gainC;
    signed long sply, splyC, cntr, cntrC;
    float x;

    KR= KRz;           //Preset gain correction coefficient
    KS= KSz;           //Preset supply coefficient

//----- Set MSC -----
    PDCON = 0x37;      //Turn on ADC
    CKCON = 0x10;      //Need for ser. communication in MSC1200/1201 only /////
    TCON = 0x00;       //Need for ser. communication in MSC1200/1201 only /////
    ACLK = FCLOCK-1;   //Aclock
    USEC = FCLOCK-1;   //One usec clock
    ONEMS = 18420;     //Setting for 1.842 MHz Clock
    ADMUX = 0x48;      //Set ADC multiplexer, AIN4-AINCOM
    ADCON0 = 0x20;     //Internal Vref = 1.25 V on, buffer off, BOD off, PGA = 1
    DECIM = DCMTN;     //Set ADC decimation ratio
    autobaud();        //Set and start serial communication with PC
    ADCON1 = 0x71;     //Unipolar, Sinc3, start ADC offset and gain calibration

    printf("\nStart USupply\nAINCOM->GND\n");
    printf("\nAIN5->Cntrl signal\nAIN4->Flying\n");
    printf("ADC Decimation=%i\n", DECIM);
    k='C';             //Start with calibration
//===== MAIN LOOP =====
    while(1)
    {
        switch (k)
//----- Measure supply voltage at AIN4 -----
        case 's': case 'S':
            ADCsupply();          break;
//----- Measure control voltage at AIN5 -----
        case 'm': case 'M':
            ADCcntrl();           break;
//----- Calibrate ADC -----
//Control signal bigger then REF/2 is recommended
        case 'c': case 'C':
            ADCON1 = 0x71;        //Unipolar, Sinc3, start ADC self-offset and gain calibration
            Wait(1);              //Wait when ADC calibration is done
            u.bt.b4=0; u.bt.b3=GCH; u.bt.b2=GCM; u.bt.b1=GCL; //Copy the ADC gain
            gainC=u.lng;          //Store the ADC gain
            printf ("ADC Gain= %li\n", gainC);

            splyC = ADCsupply();  //Measure supply voltage, store ADC readings
            cntrC = ADCcntrl();   //Measure control signal, store ADC reading
            break;

//----- Find the ADC gain correction coefficient -----
//Before this, ADC calibration with valid control signal (V > REF/2)
//and valid AVdd should be performed. Then change the supply voltage and type 'F'.
        case 'f': case 'F':
            sply = ADCsupply();    //Measure supply voltage
            cntr = ADCcntrl();     //Measure control voltage at AIN5
//----- Calculate coefficient -----
            x= (float) cntrC/cntr -1 ;
            KR= x/(float) (sply-splyC); //ADC gain correction coefficient
            printf ("ADC Gain Cor.Coeff.KR= %E\n", KR );
            break;
    }
}

```

Continued on next page

USupply program (Continued)

```

//----- Adjust the ADC gain -----
//Before this, the ADC gain correction coefficient should be found.
//Supply voltage should be changed from the calibration value to the new one.
    case 'a': case 'A':
        sply= ADCsupply(); //Measure current supply voltage
        printf ("ADC Correction Coef.= %.12lf\n", KR );//////////
        x= 1.+(float)(sply-splyC) * KR; //Calculate ADC gain correction
        printf ("ADC Adjust coef.= %lf\n", x );
        printf ("Old ADC gain= %li\n", gainC );
        u.lng= gainC * x; //New ADC gain
        printf ("New ADC gain= %li\n", u.lng );
        GCH=u.bt.b3; GCM=u.bt.b2; GCL=u.bt.b1; //Store ADC new gain
        break;

//----- Find the supply conversion coefficient -----
    case 'v': case 'V':
        ADCsupply(); //Measure supply
        printf ("Enter current supply voltage.(V)\nAVdd=");
        scanf ("%f", &x); //Read voltage as float
        KS= x/(u.lng * LSB); //New supply voltage coefficient
        printf ("\nSupply voltage coef.KS= %.3f\n", KS );
        break;

//----- Print Help text -----
    case 'h': case 'H':
        printf ("Enter\n'S' To check Supply voltage\n");
        printf (" 'M' To Measure control signal\n");
        printf (" 'A' To Adjust ADC gain\n");
        printf (" 'C' ADC self offset and gain Calibration\n");
        printf (" 'F' Find ADC gain cor.coef.\n");
        printf (" 'V' Find supply Voltage coef.\n");
        break;
    }
    printf ("-----\nFor help type 'H'\n" );
    k = getchar(); printf("\n");
}
} //main-----

##### ADC supply #####
//Measure supply voltage, using flying ADC input. AINCOM is grounded. Buffer off.
#####
unsigned long ADCsupply(void)
{
    float z;
    ADMUX= 0x48; //Switch to flying input
    z = ADC_Avrg(); //Measure flying (supply) voltage with ADC
    printf ("Flying input= %.3fv\n", z);
    z = z * KS; //Calculate AVdd
    printf ("Supply voltage= %.3fv\n", z );
    return u.lng;
} //-----

```

Continued on next page

USupply program (Continued)

```

##### ADCcntrl #####
//Measure control signal at ADC inputs AINCOM and AIN5.
//ADC unipolar mode, AINCOM is grounded. Buffer off.
#####
unsigned long ADCcntrl(void)
{
    float y;
    ADMUX= 0x58;                //AINCOM-AIN5, control voltage
    Wait(1);                    //Wait for better precision
    y = ADC_Avrg();            //Measure voltage with ADC
    printf ("Control voltage= %lfv\n", y);
    return u.lng;
}

//-----

##### Wait #####
//Make a pause for (t) ADC data cycles
#####
void Wait(char t)
{
    char n;
    n=ADRESL;                    //Dummy read to clear ADC IRQ
    n=SUMR0;                    //Dummy read to clear SUM.IRQ

    for (n=0; n<t; n++)        //Wait for t ADC conversions
    {
        while (!(AIPOL & 0x20)); //Wait for data ready MSC1200/1201
//        while (!(AIE & 0x20)); //Wait for data ready MSC1210/1211
        u.bt.b1 = ADRESL;        //Dummy read to clear ADCIRQ
    }
}

//Wait -----

/*##### ADC_Avrg #####
Get ADC average from AVRГ samples using summation register.
ADC is in unipolar mode. Result is always positive voltage. ADC code is in u.lng
#####*/
float ADC_Avrg (void)
{
    float y;
    Wait(4);                    //Wait for 4 ADC conversions, settle the filter
    SSCON = 0;                  //Clear summation registers
    SSCON = AVRG;              //Set the averaging
    while (!(AIPOL & 0x40)) {} //Wait for summator interrupt MSC1200/1201
//    while (!(AIE & 0x40)) {} //Wait for summator interrupt MSC1210/1211
    u.bt.b4=SUMR3; u.bt.b3=SUMR2; u.bt.b2=SUMR1; u.bt.b1=SUMR0; //Copy ADC data
    y= LSB * u.lng;            //ADC result in volts
//    printf ("ADC code = %ld\n",u.lng);////////////////////
    return y;
}

//ADC_Avrg -----

#####
#####

```

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

Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DSP	dsp.ti.com
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com

Applications

Audio	www.ti.com/audio
Automotive	www.ti.com/automotive
Broadband	www.ti.com/broadband
Digital control	www.ti.com/digitalcontrol
Military	www.ti.com/military
Optical Networking	www.ti.com/opticalnetwork
Security	www.ti.com/security
Telephony	www.ti.com/telephony
Video & Imaging	www.ti.com/video
Wireless	www.ti.com/wireless

TI Worldwide Technical Support

Internet

TI Semiconductor Product Information Center Home Page
support.ti.com

TI Semiconductor KnowledgeBase Home Page
support.ti.com/sc/knowledgebase

Product Information Centers

Americas

Phone	+1(972) 644-5580	Fax	+1(972) 927-6377
Internet/Email	support.ti.com/sc/pic/americas.htm		

Europe, Middle East, and Africa

Phone			
Belgium (English)	+32 (0) 27 45 54 32	Netherlands (English)	+31 (0) 546 87 95 45
Finland (English)	+358 (0) 9 25173948	Russia	+7 (0) 95 7850415
France	+33 (0) 1 30 70 11 64	Spain	+34 902 35 40 28
Germany	+49 (0) 8161 80 33 11	Sweden (English)	+46 (0) 8587 555 22
Israel (English)	1800 949 0107	United Kingdom	+44 (0) 1604 66 33 99
Italy	800 79 11 37		
Fax	+(49) (0) 8161 80 2045		
Internet	support.ti.com/sc/pic/euro.htm		

Japan

Fax			
International	+81-3-3344-5317	Domestic	0120-81-0036
Internet/Email			
International	support.ti.com/sc/pic/japan.htm		
Domestic	www.tij.co.jp/pic		

Asia

Phone			
International	+886-2-23786800		
Domestic	Toll-Free Number		
Australia	1-800-999-084	New Zealand	0800-446-934
China	800-820-8682	Philippines	1-800-765-7404
Hong Kong	800-96-5941	Singapore	800-886-1028
Indonesia	001-803-8861-1006	Taiwan	0800-006800
Korea	080-551-2804	Thailand	001-800-886-0010
Malaysia	1-800-80-3973		
Fax	886-2-2378-6808	Email	tiasia@ti.com
Internet	support.ti.com/sc/pic/asia.htm		ti-china@ti.com

C011905

Safe Harbor Statement: This publication may contain forward-looking statements that involve a number of risks and uncertainties. These "forward-looking statements" are intended to qualify for the safe harbor from liability established by the Private Securities Litigation Reform Act of 1995. These forward-looking statements generally can be identified by phrases such as "TI or its management believes," "expects," "anticipates," "foresees," "forecasts," "estimates" or other words or phrases of similar import. Similarly, such statements herein that describe the company's products, business strategy, outlook, objectives, plans, intentions or goals also are forward-looking statements. All such forward-looking statements are subject to certain risks and uncertainties that could cause actual results to differ materially from those in forward-looking statements. Please refer to TI's most recent Form 10-K for more information on the risks and uncertainties that could materially affect future results of operations. We disclaim any intention or obligation to update any forward-looking statements as a result of developments occurring after the date of this publication.

Trademarks: All other trademarks are the property of their respective owners.

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

© 2005 Texas Instruments Incorporated

SLY073