Micromint, Inc	**AN702**
	Micro64A/128A
Optional 2-Channel, 12-bit ADC	12/3/04

Introduction: This application note demonstrates how to read the optional 12-bit ADC for the Micro64A/128A.

Background: Micro64A/128A has a built in 2-channel 12-bit ADC. ADC0 (pin 30) and ADC1 (pin 31) is where the analog signals should be connected.

How it works: There are two ways the 12-bit ADC can be read, single ended and differential. The +5V (pin 40) is the reference for the 12-bit ADC. Micro64A/128A has four utility calls for the 12-bit ADC. They are single ended ADC0, single ended ADC1, differential ADC0 subtracted by ADC1, and differential ADC1 subtracted by ADC0. Single ended ADC0 will send back the digital count for the voltage on ADC0. Single ended ADC1 will send back the digital count for the voltage on ADC1. Differential ADC0 subtracted by ADC1 will send back the digital count for ADC0-ADC1. Differential ADC1 subtracted by ADC0 will send back the digital count for ADC1-ADC0. When the differential is read and the result should be negative then a digital count of zero will be sent back. This is because the 12-bit ADC does not have a sign bit and can only send positive numbers. The CodeVision AVR program listed below is for the Micro64A. Please download the zip file that contains the program for the Micro128A.

Program Listing:

```
/*****************************************
Program : MCP3202 Example for Micro64
Company :Micromint, Inc.
*********************************************/

#include <mega64.h>
#include <MMRS485.h>     // Micromints Library for using both USARTs
// Standard Input/Output functions
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <delay.h>                              // Library for delays
// Declare your global variables here
#define MAX_ENTRY_LENGTH 10
void(*SingleADC1)(void)=0x7BFD;
void(*SingleADC0)(void)=0x7C04;
void(*DiffADC1)(void)=0x7C12;
void(*DiffADC0)(void)=0x7C0B;
float getVref(void);
int COM;        // if COM = 0 then use USART0 if it = 1 then use USART1
unsigned int Data @0xFFE;
void main(void)
{
// Declare your local variables here
float Volt;
float Vref;


// Set up USART1's Baud rate at 9600 bps with a 11.0592 MHz Crystal
UCSR1A=0x00;     // RX EN, TX EN
UCSR1B=0x18;    // RX EN, TX EN
UCSR1C=0x06;    // 8N1
UBRR1H=0x00;   // Baud rate high - 9600
UBRR1L=0x47;   // Baud rate low
```

```
         Vref = getVref();
while (1)
      {
      // Place your code here
        COM = 1;                        // Use USART1
        DDRD.6 = 0;                                       // Make PORTD.6 an output
        PORTD.6 = 1;                                      // Enable the RS485 control line
        (*SingleADC0)();
        Volt = Vref/4095.0 * Data;
        printf("CH0= %02.02f", Volt);
        printf(" V ");
        (*SingleADC1)();
        Volt = Vref/4095.0 * Data;
        printf("CH1= %02.02f", Volt);
        printf(" V ");
        (*DiffADC0)();
        Volt = Vref/4095.0 * Data;
        printf("CH0-CH1= %02.02f", Volt);
        printf(" V ");
        (*DiffADC1)();
        Volt = Vref/4095.0 * Data;
        printf("CH1-CH0= %02.02f", Volt);
        printf(" V\r\r");
        delay_ms(100);
                PORTD.6 = 0;                                          // Disable the RS485 control line
      };
}

float getVref(void)
{
char mystr[MAX_ENTRY_LENGTH + 1]; //Declare a string
char c;
        COM = 1;
        DDRD.6 = 0;                                       // Make PORTD.6 an output
        PORTD.6 = 1;                                      // Enable the RS485 control line
        c = 0;
        printf("Please enter a decimal value for the reference voltage?\n\r");
        while( c < MAX_ENTRY_LENGTH)
        {
          mystr[c++] = getchar();
          if (mystr[c-1] == 13)
             break;
        }
        mystr[c] = '\0';
        return  atof(mystr);
}
```