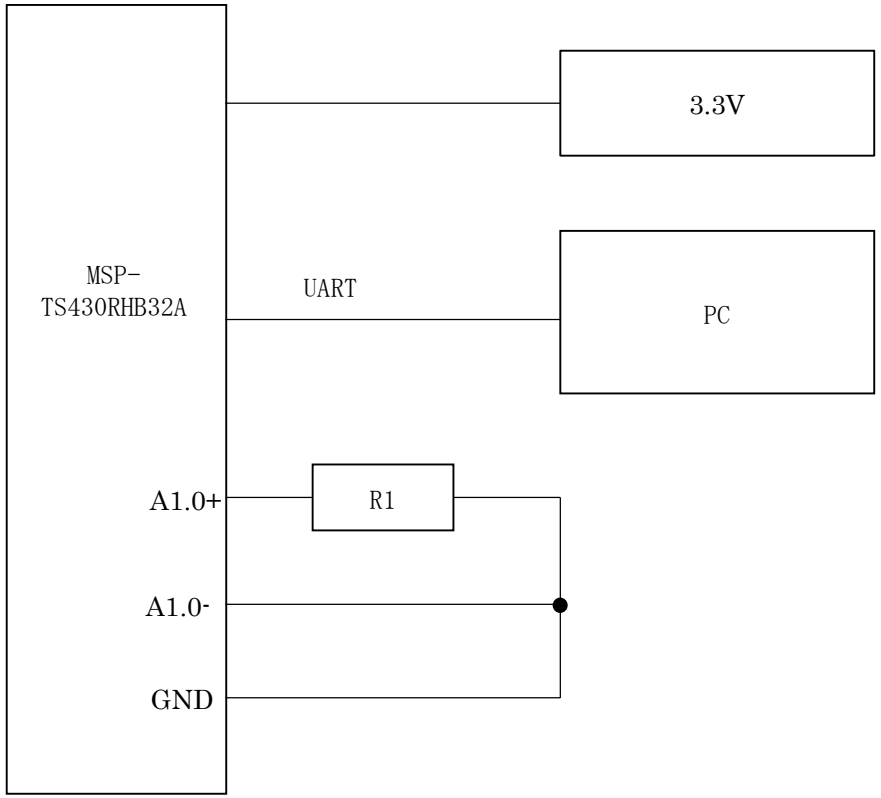
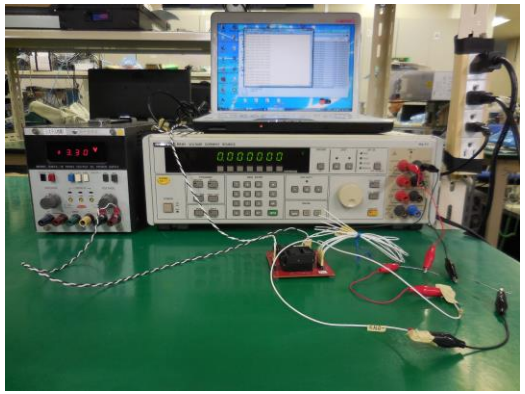
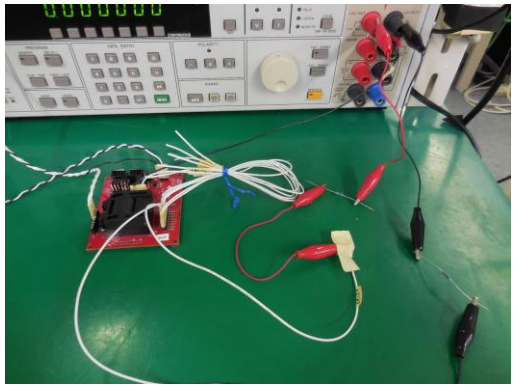
A

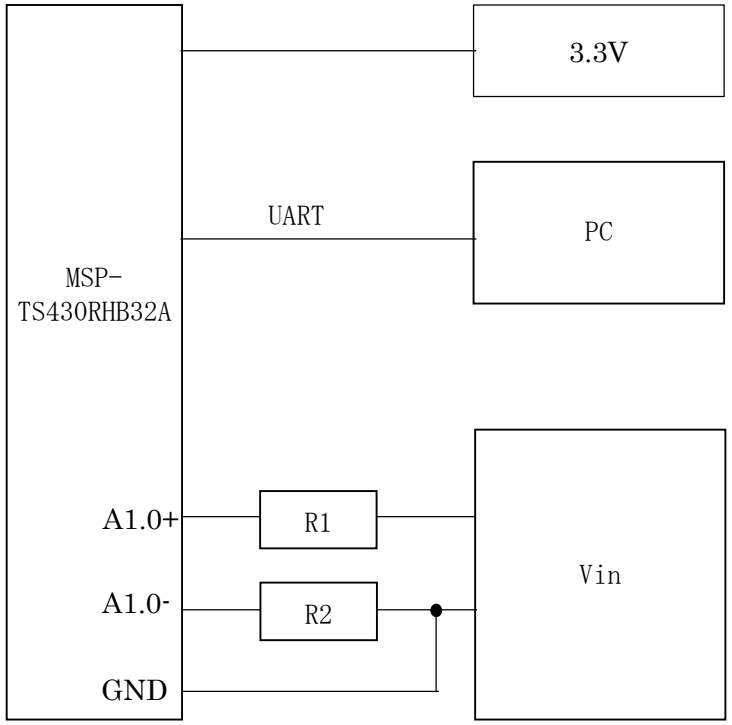


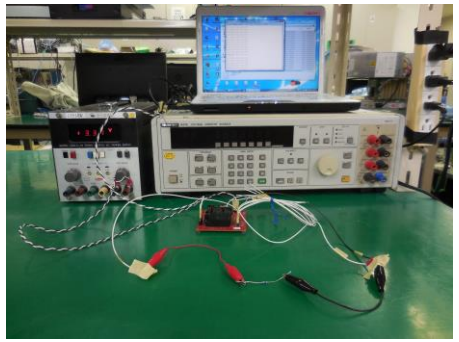


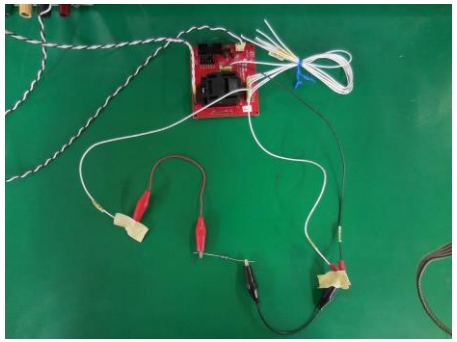


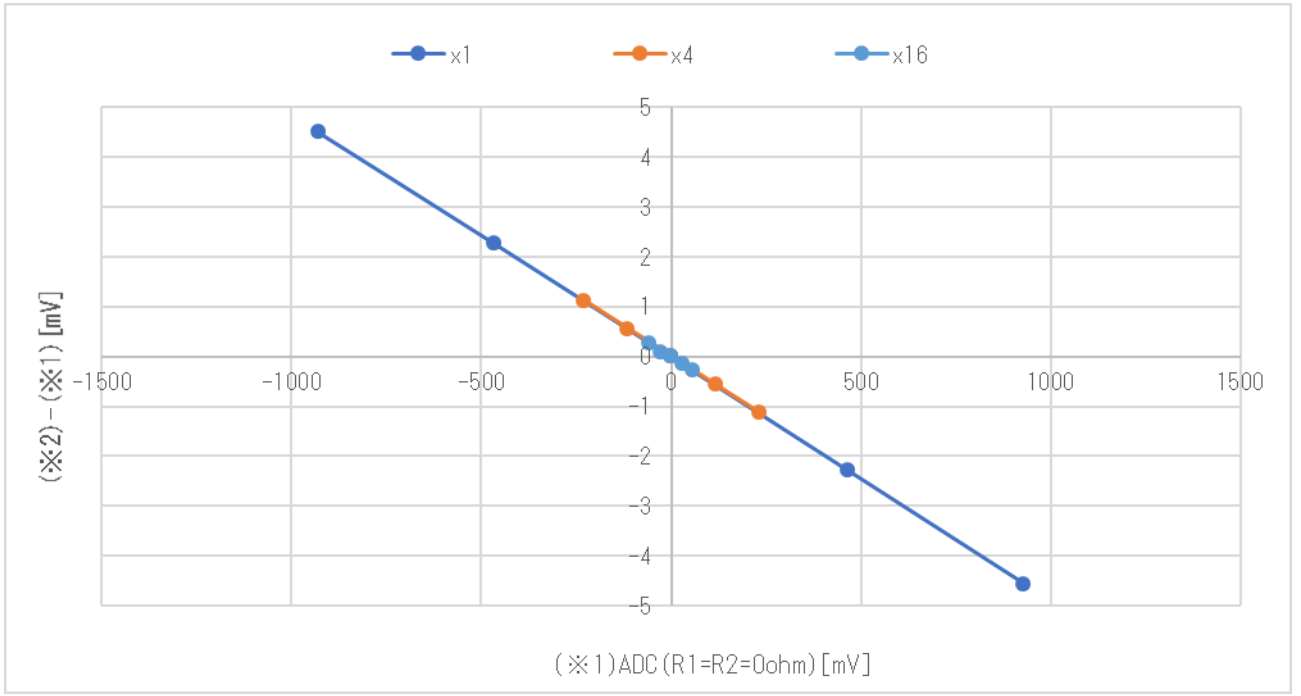
Gain	ADC (R1=0ohm)	ADC (R1=1kohm)	ADC (R1=10kohm)
x1	-1.411mV	-0.328mV	9.009mV
x2	-0.704mV	-0.165mV	4.483mV
x4	-0.649mV	3.603mV	40.115mV
x8	-0.689mV	3.588mV	40.111mV
x16	-0.682mV	3.563mV	40.033mV

B









Gain : x1			
Vin	(※1) ADC (R1=R2=0ohm)	(※2) ADC (R1=R2=1kohm)	(※2)-(※1)
926mV	927.736mV	923.189mV	-4.547mV
463mV	463.137mV	460.867mV	-2.270 mV
0mV	-1.417mV	-1.412mV	0.005mV
-463mV	-465.911mV	-463.637mV	2.274mV
-926mV	-930.334mV	-925.821mV	4.513mV

Gain : x4			
Vin	(※1) ADC (R1=R2=0ohm)	(※2) ADC (R1=R2=1kohm)	(※2)-(※1)
230mV	229.802mV	228.693mV	-1.109mV
115.8mV	115.366mV	114.810mV	-0.556mV
0mV	-0.669mV	-0.657mV	0.011mV
-115.8mV	-116.701mV	-116.132mV	0.569mV
-230mV	-231.129mV	-230.005mV	1.124mV

Gain : x16			
Vin	(※1) ADC (R1=R2=0ohm)	(※2) ADC (R1=R2=1kohm)	(※2)-(※1)
57.9mV	57.133mV	56.861mV	-0.272mV
28.95mV	28.214mV	28.083mV	-0.131mV
0mV	-0.704mV	-0.694mV	0.010mV
-28.95mV	-29.624mV	-29.520mV	0.104mV
-57.9mV	-58.543mV	-58.255mV	0.288mV

```
#include <msp430.h>

int gain_value[4];
unsigned long int adc_data[5];
unsigned long int adc_data_tmp[4];


void led_on()
{
    P1OUT |= BIT4;            // Toggle LED
}

void led_off()
{
    P1OUT &= ~BIT4;           // Toggle LED
}

void putc_uart(char c)
{
    while (!(UCA0IFG&UCTXIFG)); // USCI_A0 TX buffer ready?
    UCA0TXBUF = c;
}

void put_adc_data(unsigned long int d)
{
    static char hexchar[]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
    putc_uart(hexchar[(d>>20)&0xf]);
    putc_uart(hexchar[(d>>16)&0xf]);
    putc_uart(hexchar[(d>>12)&0xf]);
    putc_uart(hexchar[(d>> 8)&0xf]);
    putc_uart(hexchar[(d>> 4)&0xf]);
    putc_uart(hexchar[(d    )&0xf]);
}

void main(void) {
    WDTCTL = WDTPW | WDTHOLD;                     // Stop WDT

    P1DIR |= BIT4;              // LED OutputA
    led_on();

    SD24CTL = SD24REFS;                           // Internal ref
    SD24CCTL0 = 0;
    SD24CCTL1 = 0;
    SD24CCTL2 = 0;
    SD24CCTL3 = 0;

    //  Timer_A0, PWM, Up Mode, DCO SMCLK/3
    P1SEL1 |= BIT5;                   // P1.5 CCRx Function
    P1DIR  |= BIT5;                   // Set P1.5 as output

    TA0CCR0 = 9;
    TA0CCTL1 = OUTMOD_3;              // CCR1 Toggle/set
    TA0CCR1 = 5;
    TA0CTL = TASSEL_2 | MC_1 | ID_0;    // SMCLK/1, Up/Down Mode

    _delay_cycles(320000);                        // Delay ~200us for 1.2V ref to settle

//  eUSCI_A, 115200 UART Echo ISR, DCO SMCLK
    P1SEL0 |=  BIT2 | BIT3;           // P1.2/3 eUSCI_A Function
    P1SEL1 &= ~(BIT2 | BIT3);

    UCA0CTL1 |= UCSWRST;              // Hold eUSCI in reset
    UCA0CTL1 |= UCSSEL_2;             // SMCLK
    UCA0BR0   = 142;                  // 115200 baud
    UCA0BR1   = 0;
    UCA0MCTLW = 0x2200;               // 16.384MHz/115200 = 142.22 (See UG)
    UCA0CTL1 &= ~UCSWRST;             // Release from reset
    UCA0IE   |= UCRXIE;               // Enable RX interrupt

    while (1) {
        _bis_SR_register(LPM0_bits | GIE);     // Enter LPM0 w/ interrupts
        _no_operation();

        SD24CCTL0 |= SD24SNGL | SD24GRP;
        SD24CCTL1 |= SD24SNGL | SD24GRP;
        SD24CCTL2 |= SD24SNGL | SD24GRP;
        SD24CCTL3 |= SD24SNGL | SD24IE;

        SD24INCTL0 &= 0xC0;
        SD24INCTL0 |= gain_value[0];
        SD24INCTL1 &= 0xC0;
        SD24INCTL1 |= gain_value[1];
        SD24INCTL2 &= 0xC0;
        SD24INCTL2 |= gain_value[2];
```

```
        SD24INCTL3 &= 0xC0;
        SD24INCTL3 |= gain_value[3];
        SD24CCTL3 |= SD24SC;
        _bis_SR_register(LPM0_bits | GIE);     // Enter LPM0 w/ interrupts
        _no_operation();

        adc_data[0]=adc_data_tmp[0];
        adc_data[1]=adc_data_tmp[1];
        adc_data[2]=adc_data_tmp[2];
        adc_data[3]=adc_data_tmp[3];
        SD24INCTL0 |= SD24INCH_6;
        SD24CCTL3 |= SD24SC;
        _bis_SR_register(LPM0_bits | GIE);     // Enter LPM0 w/ interrupts
        _no_operation();

        adc_data[4]=adc_data_tmp[0];

        put_adc_data(adc_data[0]);
        putc_uart(',');
        put_adc_data(adc_data[1]);
        putc_uart(',');
        put_adc_data(adc_data[2]);
        putc_uart(',');
        put_adc_data(adc_data[3]);
        putc_uart(',');
        put_adc_data(adc_data[4]);
        putc_uart(0x0a);
        putc_uart(0x00);

        _no_operation();                  // For debugger
    }
}

#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=SD24_VECTOR
__interrupt void SD24_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(SD24_VECTOR))) SD24_ISR (void)
#else
#error Compiler not supported!
#endif
{
    unsigned long int v;
    switch (__even_in_range(SD24IV,SD24IV_SD24MEM3)) {
        case SD24IV_NONE: break;
        case SD24IV_SD24OVIFG:  break;
        case SD24IV_SD24MEM0:   break;
        case SD24IV_SD24MEM1:   break;
        case SD24IV_SD24MEM2:   break;
        case SD24IV_SD24MEM3:
            SD24CCTL0 &= ~(SD24LSBACC);
            v = SD24MEM0;
            adc_data_tmp[0] = v << 8;       // Save CH0 results (clears IFG)
            SD24CCTL0 |= SD24LSBACC;
            v = SD24MEM0;
            adc_data_tmp[0] |= v;

            SD24CCTL1 &= ~(SD24LSBACC);
            v = SD24MEM1;
            adc_data_tmp[1] = v << 8;       // Save CH0 results (clears IFG)
            SD24CCTL1 |= SD24LSBACC;
            v = SD24MEM1;
            adc_data_tmp[1] |= v;

            SD24CCTL2 &= ~(SD24LSBACC);
            v = SD24MEM2;
            adc_data_tmp[2] = v << 8;       // Save CH0 results (clears IFG)
            SD24CCTL2 |= SD24LSBACC;
            v = SD24MEM2;
            adc_data_tmp[2] |= v;

            SD24CCTL3 &= ~(SD24LSBACC);
            v = SD24MEM3;
            adc_data_tmp[3] = v << 8;       // Save CH0 results (clears IFG)
            SD24CCTL3 |= SD24LSBACC;
            v = SD24MEM3;
            adc_data_tmp[3] |= v;

            __bic_SR_register_on_exit(LPM0_bits);       // Wake up from LPM0
            break;
        default: break;
    }
}
```

```
// Echo back RXed character, confirm TX buffer is ready first
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=USCI_A0_VECTOR
__interrupt void USCI_A0_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(USCI_A0_VECTOR))) USCI_A0_ISR (void)
#else
#error Compiler not supported!
#endif
{
  static int n=0,i;
  static char str[256];
  char c;

  switch(__even_in_range(UCA0IV,USCI_UART_UCTXCPTIFG)) {
    case USCI_NONE: break;
    case USCI_UART_UCRXIFG:
      c = UCA0RXBUF;
      str[n++]=c;

      if (c==0x0a) {
        n=0;
        if (str[0]=='A') {
          led_on();
          __bic_SR_register_on_exit(LPM0_bits);     // Wake up
        }
        if (str[0]=='P') {
          int p;
          p=(str[1]-'0')*10 + (str[2]-'0');
          if (p==1)   gain_value[0] = SD24GAIN_1;
          if (p==4)   gain_value[0] = SD24GAIN_4;
          if (p==8)   gain_value[0] = SD24GAIN_8;
          if (p==16)  gain_value[0] = SD24GAIN_16;

          p=(str[4]-'0')*10 + (str[5]-'0');
          if (p==1)   gain_value[1] = SD24GAIN_1;
          if (p==4)   gain_value[1] = SD24GAIN_4;
          if (p==8)   gain_value[1] = SD24GAIN_8;
          if (p==16)  gain_value[1] = SD24GAIN_16;

          p=(str[7]-'0')*10 + (str[8]-'0');
          if (p==1)   gain_value[2] = SD24GAIN_1;
          if (p==4)   gain_value[2] = SD24GAIN_4;
          if (p==8)   gain_value[2] = SD24GAIN_8;
          if (p==16)  gain_value[2] = SD24GAIN_16;

          p=(str[10]-'0')*10 + (str[11]-'0');
          if (p==1)   gain_value[3] = SD24GAIN_1;
          if (p==4)   gain_value[3] = SD24GAIN_4;
          if (p==8)   gain_value[3] = SD24GAIN_8;
          if (p==16)  gain_value[3] = SD24GAIN_16;
        }
      }
      break;
    case USCI_UART_UCTXIFG:
      break;
    case USCI_UART_UCSTTIFG:
      break;
    case USCI_UART_UCTXCPTIFG:
      break;
    default: break;
  }
}
```