# DAFTAR PUSTAKA


1. Earns Haw, S., "On the Nature of the Molecular Forces which Regulate the Constitution of The Luminiferous Ether", Trans. Camb., Phil. Spc., 7, pp. 97-112, 1842
2. Naumovic, Milica B., "Magnetic Levitation System in Control Engineering Education", Department of Automatic Control, University of Nis, Faculty of Electronic Engineering, Serbia, 2008
3. Yunianto, Bambang, "Kajian Permasalahan Lingkungan dan Sosial Ekonomi Rencana Penambangan dan Pengolahan Pasir Besi di Pantai Selatan Kulon Progo – Yogyakarta", Puslitbang Teknologi Mineral dan Batubara, Bandung, Januari 2009
4. Young Hugh D. dan Freedman Roger A., University Physics. Massachusetts: Addison-Wesley, 9th edition, 1998.
5. Eng. Khalid Abdelhafiz Ali. *Modelling, Identification of Amagnetic Levitation CE152*, Al-Aqsa University, 2010.
6. Shafayet Hossain. *Design of a Robust Controller for a Magnetic Levitation System*, Wichita State University.
7. Stephen C. Paschall II. *Design, Fabrication, and Control of A Single Actuator Magnetic Levitation System*, Texas A&M University, 2002.
8. Sintanyehu Challa. *Magnetic Levitation on A Steel Ball*, Addis Ababa University, 2007.
9. Katsuhiko Ogata. *Teknik Kontrol Automatik*, jilid I, Edisi kedua, Erlangga.
10. http://rodjoelgroup.blogspot.com/2012/03/daftar-kemampuan-kawat-email dilalui.html diakses 10 Mei 2012.
11. Lance Williams. *Electromagnetic Levitation Thesis*, 2005.
12. http://www.scribd.com/doc/94538561/Levitasi-Magnetik-Adalah-Proses-Melayang-Objek-Dengan-Memanfaatkan-Magnet diakses 4 Juli 2012.
13. http://www.electronics-tutorials.ws/electromagnetism/hall-effect.html diakses 4 Juli 2012.
14. Yulastri. Aplikasi Sensor UGN3505 Sebagai Pendeteksi Medan Magnet, Juni 2009

15. Korps asisten Lab Kendali dan Robotika. 2009. *Modul Praktikum Dasar Sistem kendali*. Indralaya: Laboratorium Kendali dan Robotika Teknik Elektro Universitas Sriwijaya.
16. Katsuhiko Ogata. Modern Control Engineering, 5th Edition, Prentice Hall.
17. Modul Training Mikrokontroler AVR. Bidang Keprofesian Badan Pengurus HME ITB, 2009-2010.

# LAMPIRAN

# A1301 and A1302

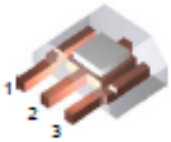
## Continuous-Time Ratiometric Linear Hall Effect Sensors

Package LH, 3-pin Surface Mount

1. VCC
2. VOUT
3. GND


Package UA, 3-pin SIP

1. VCC
2. GND
3. VOUT




**ABSOLUTE MAXIMUM RATINGS**

| | |
|---|---|
| Supply Voltage, $V_{CC}$ | 8 V |
| Output Voltage, $V_{OUT}$ | 8 V |
| Reverse-Supply Voltage, $V_{RCC}$ | –0.1 V |
| Reverse-Output Voltage, $V_{ROUT}$ | –0.1 V |
| Output Sink Current, $I_{OUT}$ | 10 mA |
| Operating Temperature | |
| Ambient, $T_A$, Range E | –40°C to 85°C |
| Ambient, $T_A$, Range K | –40°C to 125°C |
| Maximum Junction, $T_{J(max)}$ | 165°C |
| Storage Temperature, $T_S$ | –65°C to 170°C |

The A1301 and A1302 are continuous-time, ratiometric, linear Hall-effect sensors. They are optimized to accurately provide a voltage output that is proportional to an applied magnetic field. These devices have a quiescent output voltage that is 50% of the supply voltage. Two output sensitivity options are provided: 2.5 mV/G typical for the A1301, and 1.3 mV/G typical for the A1302.

The Hall-effect integrated circuit included in each device includes a Hall sensing element, a linear amplifier, and a CMOS Class A output structure. Integrating the Hall sensing element and the amplifier on a single chip minimizes many of the problems normally associated with low voltage level analog signals.

High precision in output levels is obtained by internal gain and offset trim adjustments made at end-of-line during the manufacturing process.

These features make the A1301 and A1302 ideal for use in position sensing systems, for both linear target motion and rotational target motion. They are well-suited for industrial applications over extended temperature ranges, from –40°C to 125°C.

Two device package types are available: LH, a 3-pin SOT23W type for surface mount, and UA, a 3-pin ultramini SIP for through-hole mount. Each package is available in a lead (Pb) free version (suffix, –T) with 100% matte tin plated lead-frame.

### Features and Benefits

- Low-noise output
- Fast power-on time
- Ratiometric rail-to-rail output
- 4.5 to 6.0 V operation
- Solid-state reliability
- Factory-programmed at end-of-line for optimum performance
- Robust ESD performance



Allegro MicroSystems, Inc.
115 Northeast Cutoff, Box 15036
Worcester, Massachusetts 01615-0036 (508) 853-5000
www.allegromicro.com

# A1301 and A1302

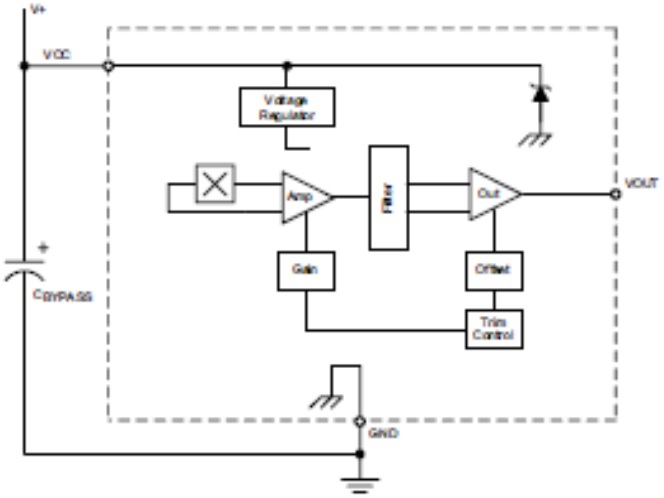
## Continuous-Time Ratiometric Linear Hall Effect Sensors

### Product Selection Guide

| Part Number | Pb-free | Packing* | Package | Ambient, $T_A$ | Sensitivity (Typical) |
|---|---|---|---|---|---|
| A1301ELHLT | – | 7-in. tape and reel, 3000 pieces/reel | Surface Mount | –40°C to 85°C | 2.5 mV/G |
| A1301ELHLT-T | Yes | | | | |
| A1301EUA | – | Bulk, 500 pieces/bag | SIP | | |
| A1301EUA-T | Yes | | | | |
| A1301KLHLT | – | 7-in. tape and reel, 3000 pieces/reel | Surface Mount | –40°C to 125°C | |
| A1301KLHLT-T | Yes | | | | |
| A1301KUA | – | Bulk, 500 pieces/bag | SIP | | |
| A1301KUA-T | Yes | | | | |
| A1302ELHLT | – | 7-in. tape and reel, 3000 pieces/reel | Surface Mount | –40°C to 85°C | 1.3 mV/G |
| A1302ELHLT-T | Yes | | | | |
| A1302EUA | – | Bulk, 500 pieces/bag | SIP | | |
| A1302EUA-T | Yes | | | | |
| A1302KLHLT | – | 7-in. tape and reel, 3000 pieces/reel | Surface Mount | –40°C to 125°C | |
| A1302KLHLT-T | Yes | | | | |
| A1302KUA | – | Bulk, 500 pieces/bag | SIP | | |
| A1302KUA-T | Yes | | | | |

*Contact Allegro for additional packing options.

### Functional Block Diagram




Terminal List

| Symbol | Description | Number | |
|---|---|---|---|
| | | Package LH | Package UA |
| VCC | Connects power supply to chip | 1 | 1 |
| VOUT | Output from circuit | 2 | 3 |
| GND | Ground | 3 | 2 |

**DEVICE CHARACTERISTICS over operating temperature range, $T_A$, and $V_{CC}$ = 6 V, unless otherwise noted**

| Characteristic | Symbol | Test Conditions | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|---|
| **Electrical Characteristics** | | | | | | |
| Supply Voltage | $V_{CC}$ | Running, $T_J$ < 165°C | 4.5 | – | 6 | V |
| Supply Current | $I_{CC}$ | Output open | – | – | 11 | mA |
| Output Voltage | $V_{OUT(High)}$ | $I_{SOURCE}$ = –1 mA, Sens = nominal | 4.65 | 4.7 | – | V |
| | $V_{OUT(Low)}$ | $I_{SINK}$ = 1 mA, Sens = nominal | – | 0.2 | 0.25 | V |
| Output Bandwidth | BW | | – | 20 | – | kHz |
| Power-On Time | $t_{PO}$ | $V_{CC(min)}$ to 0.95 $V_{OUT}$ B = ±1400 G; Slew rate = 4.5 V/µs to 4.5 V/100 ns | – | 3 | 5 | µs |
| Output Resistance | $R_{OUT}$ | $I_{SINK}$ ≤ 1 mA, $I_{SOURCE}$ ≥ –1 mA | – | 2 | 5 | Ω |
| Wide Band Output Noise, rms | $V_{OUTN}$ | External output low pass filter ≤ 10 kHz; Sens = nominal | – | 150 | – | µV |
| **Ratiometry** | | | | | | |
| Quiescent Output Voltage Error with respect to $\Delta V_{CC}$[1] | $\Delta V_{OUTQ(V)}$ | $T_A$ = 25°C | – | – | ±3.0 | % |
| Magnetic Sensitivity Error with respect to $\Delta V_{CC}$[2] | $\Delta Sens_{(V)}$ | $T_A$ = 25°C | – | – | ±3.0 | % |
| **Output** | | | | | | |
| Linearity | Lin | $T_A$ = 25°C | – | – | ±2.5 | % |
| Symmetry | Sym | $T_A$ = 25°C | – | – | ±3.0 | % |
| **Magnetic Characteristics** | | | | | | |
| Quiescent Output Voltage | $V_{OUTQ}$ | B = 0 G; $T_A$ = 25°C | 2.4 | 2.5 | 2.6 | V |
| Quiescent Output Voltage over Operating Temperature Range | $V_{OUTQ(\Delta T_A)}$ | B = 0 G | 2.2 | – | 2.8 | V |
| Magnetic Sensitivity | Sens | A1301; $T_A$ = 25°C | 2.0 | 2.5 | 3.0 | mV/G |
| | | A1302; $T_A$ = 25°C | 1.0 | 1.3 | 1.6 | mV/G |
| Magnetic Sensitivity over Operating Temperature Range | $Sens_{(\Delta T_A)}$ | A1301 | 1.8 | – | 3.2 | mV/G |
| | | A1302 | 0.85 | – | 1.75 | mV/G |

[1]Refer to equation (4) in Ratiometric section on page 4.
[2]Refer to equation (5) in Ratiometric section on page 4.

## Characteristic Definitions

**Quiescent Output Voltage.** In the quiescent state (no significant magnetic field: B = 0), the output, $V_{OUTQ}$, equals one half of the supply voltage, $V_{CC}$, throughout the entire operating ranges of $V_{CC}$ and ambient temperature, $T_A$. Due to internal component tolerances and thermal considerations, however, there is a tolerance on the quiescent output voltage, $\Delta V_{OUTQ}$, which is a function of both $\Delta V_{CC}$ and $\Delta T_A$. For purposes of specification, the quiescent output voltage as a function of temperature, $\Delta V_{OUTQ(\Delta T_A)}$, is defined as:

$$\Delta V_{OUTQ(\Delta T_A)} = \frac{V_{OUTQ(T_A)} - V_{OUTQ(25°C)}}{Sens_{(25°C)}} \qquad (1)$$

where Sens is in mV/G, and the result is the device equivalent accuracy, in gauss (G), applicable over the entire operating temperature range.

**Sensitivity.** The presence of a south-polarity (+B) magnetic field, perpendicular to the branded face of the device package, increases the output voltage, $V_{OUT}$, in proportion to the magnetic field applied, from $V_{OUTQ}$ toward the $V_{CC}$ rail. Conversely, the application of a north polarity (−B) magnetic field, in the same orientation, proportionally decreases the output voltage from its quiescent value. This proportionality is specified as the magnetic sensitivity of the device and is defined as:

$$Sens = \frac{V_{OUT(-B)} - V_{OUT(+B)}}{2B} \qquad (2)$$

The stability of the device magnetic sensitivity as a function of ambient temperature, $\Delta Sens_{(\Delta T_A)}$ (%) is defined as:

$$\Delta Sens_{(\Delta T_A)} = \frac{Sens_{(T_A)} - Sens_{(25°C)}}{Sens_{(25°C)}} \times 100\% \qquad (3)$$

**Ratiometric.** The A1301 and A1302 feature a ratiometric output. This means that the quiescent voltage output, $V_{OUTQ}$, and the magnetic sensitivity, Sens, are proportional to the supply voltage, $V_{CC}$.

The ratiometric change (%) in the quiescent voltage output is defined as:

$$\Delta V_{OUTQ(\Delta V)} = \frac{\Delta V_{OUTQ(V_{CC})} / \Delta V_{OUTQ(5V)}}{V_{CC} / 5\,V} \times 100\% \qquad (4)$$

and the ratiometric change (%) in sensitivity is defined as:

$$\Delta Sens_{(\Delta V)} = \frac{Sens_{(V_{CC})} / Sens_{(5V)}}{V_{CC} / 5\,V} \times 100\% \qquad (5)$$

**Linearity and Symmetry.** The on-chip output stage is designed to provide linear output at a supply voltage of 5 V. Although the application of very high magnetic fields does not damage these devices, it does force their output into a nonlinear region. Linearity in percent is measured and defined as:

$$Lin+ = \frac{V_{OUT(+B)} - V_{OUTQ}}{2\,(V_{OUT(+B/2)} - V_{VOUTQ})} \times 100\% \qquad (6)$$

$$Lin- = \frac{V_{OUT(-B)} - V_{OUTQ}}{2(V_{OUT(-B/2)} - V_{OUTQ})} \times 100\% \qquad (7)$$

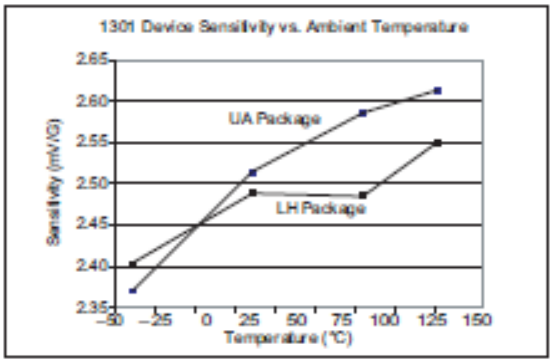
and output symmetry as:

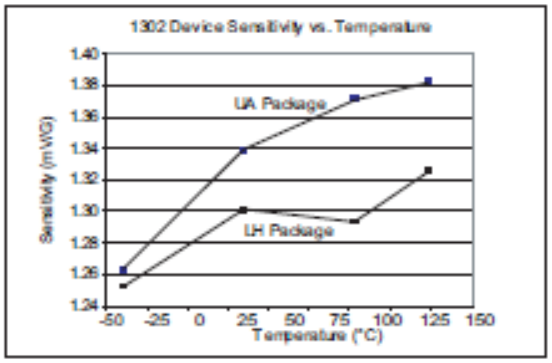
$$Sym = \frac{V_{OUT(+B)} - V_{OUTQ}}{V_{OUTQ} - V_{OUT(-B)}} \times 100\% \qquad (8)$$

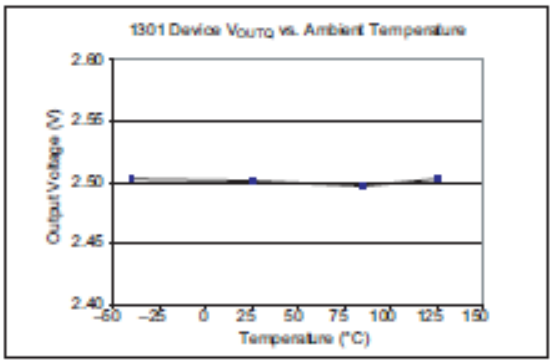


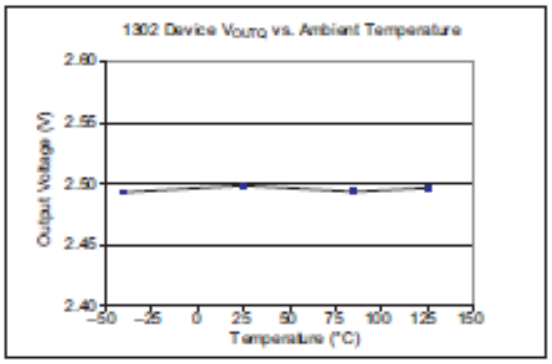
Allegro MicroSystems, Inc.
115 Northeast Cutoff, Box 15036
Worcester, Massachusetts 01615-0036 (508) 853-5000
www.allegromicro.com

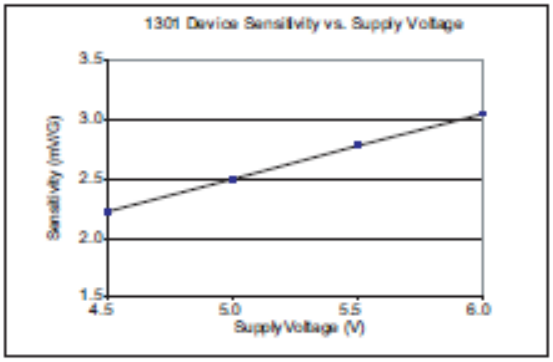
## Typical Characteristics
(30 pieces, 3 fabrication lots)

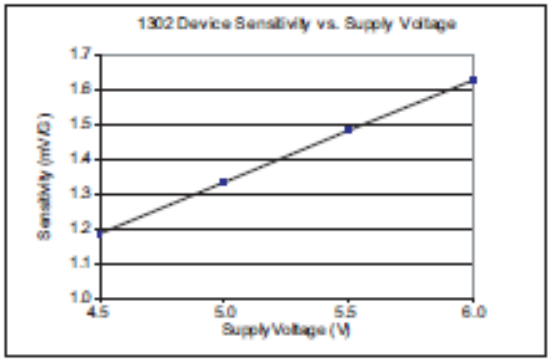


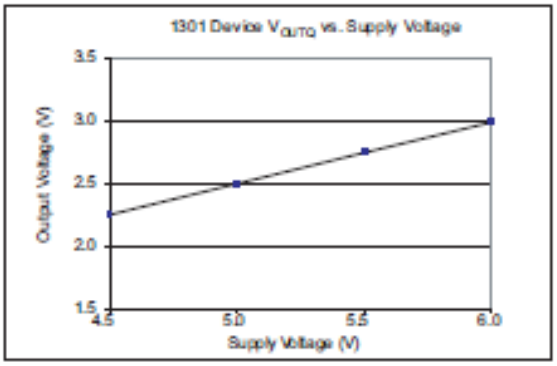



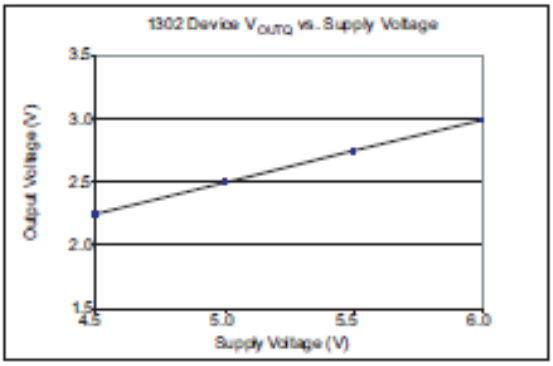



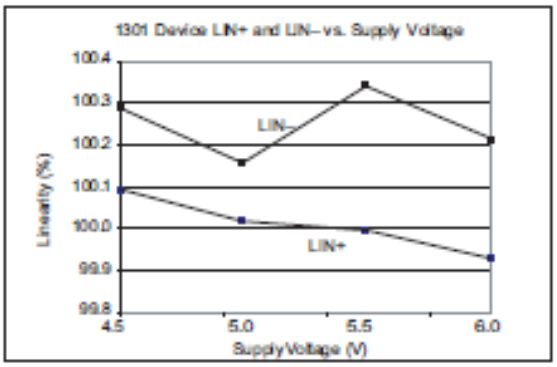



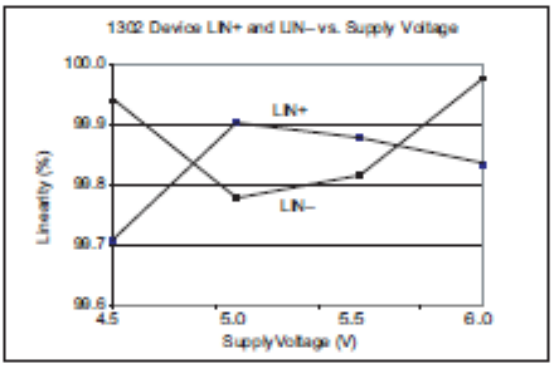

*Continued on the next page...*

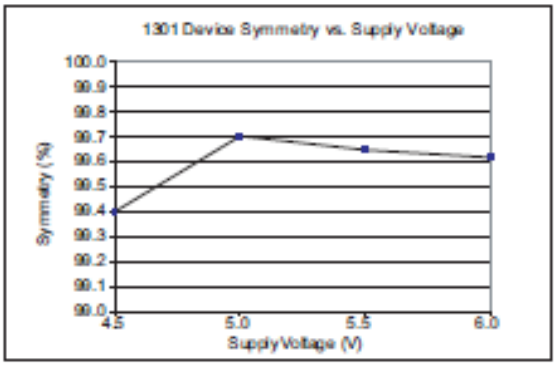
## Typical Characteristics, continued
(30 pieces, 3 fabrication lots)

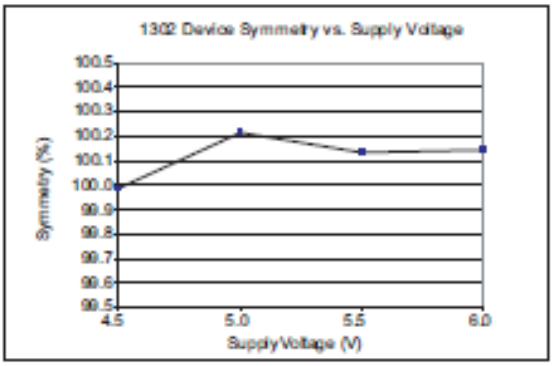


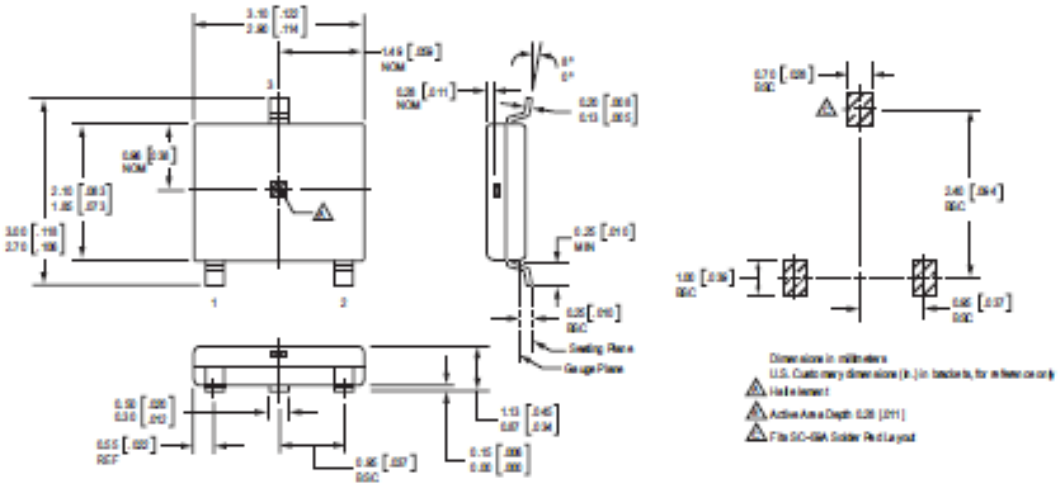



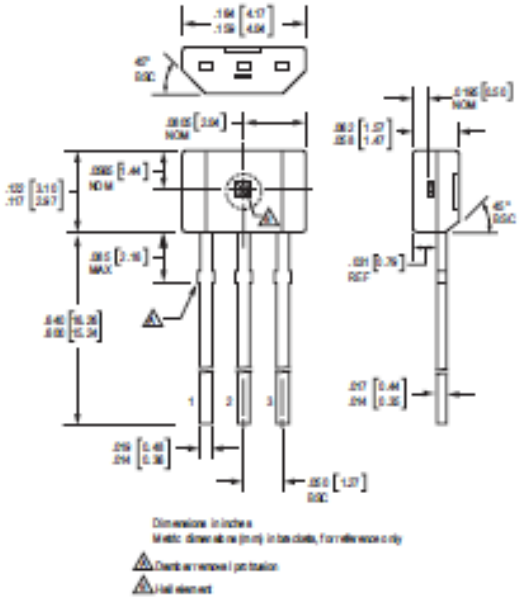



## Package LH, 3-Pin; (SOT-23W)



## Package UA, 3-Pin; (TO-92)



```
/*****************************************************
Project : Magnetic Levitation
Date    : 7/18/2012
Author  : No Name

Chip type           : ATmega8
AVR Core Clock frequency: 16.000000 MHz
*****************************************************/

//#define _mega88     // chip ATmega88
                // configure chip

#ifdef _mega88
#include <mega88.h>
#else
#include <mega8.h>
#endif

#include <delay.h>
#include <stdio.h>

#define ADC_VREF_TYPE 0x40

// Read the AD conversion result
#pragma used+
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
```

```
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
#pragma used-

#include <manipulator.c>
#include <display.c>

#define pot1       3
#define pot2       4
#define hall       5
#define SW         PINC.0

#define get_state   0
#define run        1
#define set_SP      2
#define set_PD      3
#define kalibrasi   4

#pragma used+
eeprom signed int Emin_flux,Emax_flux;
eeprom signed int Ecenter,Erange;
eeprom signed int Eambiant[128];
signed int ambiant[128],amb;
```

```
signed int min_flux,max_flux;
signed int center,range;
signed int raw_flux;
unsigned int led_map,index;
unsigned char state;
unsigned char count,i;
unsigned char justifunction,point_type;
#pragma used-

#define _P  0
#define _D  1
#define _L  2
#define _M  3
#define _S  4

#define Pscale  0.002
#define Dscale  0.1
#define Psmoot  0.6
#define Dsmoot  0.4

#define windupI 20.0

#pragma used+
//float error,last_error;
float error,error2;
float set,ref,vel,duty,out;
float D_smt,abs_ofset;
eeprom signed int Econtroller_const[5];
eeprom float Eofset;
```

```
float controller_const[5],ofset;
#pragma used-

void set_coil(unsigned char PWM)
{
#ifdef _mega88
OCR2A = PWM;
#else
OCR2 = PWM;
#endif
}

void main(void)
{
#ifdef _mega88
// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif
#endif

// Input/Output Ports initialization
// Port B initialization
// 5=0 4=0 3=0 2=0 1=0 0=0
DDRB=0x3F;
```

```
// Port C initialization
// 6=P 5=T 4=T 3=T 2=0 1=P 0=P
PORTC=0x43;
DDRC=0x04;

// Port D initialization
// 7=0 6=0 5=0 4=0 3=0 2=0 1=0 0=P
PORTD=0x01;
DDRD=0xFE;

// Clock value: 250.000 kHz
// Mode: Phase correct PWM top=0xFF
ASSR=0x00;
#ifdef _mega88
TCCR2A=0x81;
TCCR2B=0x04;
#else
TCCR2=0x64;     // Clock value: 250.000 kHz
#endif

// Analog Comparator: Off
ACSR=0x80;
#ifdef _mega88
ADCSRB=0x00;
DIDR1=0x00;
#else
SFIOR=0x00;
#endif
```

```
// ADC Clock frequency: 1000.000 kHz
// ADC Voltage Reference: AVCC pin
#ifdef _mega88
DIDR0=0x00;
#endif
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;


for(i=0;i<5;i++)    {
   for(count=6;count<12;count++)    {
      led_map = (get_map(count,1,1,1) & get_map(count,1,0,1));
      set_led(led_map);
      delay_ms(50);
      }
   }
while(SW)  delay_ms(10);

count = 0;
while(state==get_state)            //------------------------>>> tekan singkat --> runing
   {                          //------------------------>>> tekan lama 50% --> set_PD,    100%
--> kalibrasi
   delay_ms(25);
   if(count<105)   count++;
   set_led(get_map(count,10,1,0));
   if(SW)  {
      if((count>50)&(count<100))  {state = set_PD;    for(i=0;i<3;i++)
{set_led(0x1F);     delay_ms(200); set_led(0x3FF); delay_ms(200);}}
```

```
        else                {state = run;      for(i=0;i<3;i++)    {set_led(0);
delay_ms(200); set_led(0x3FF); delay_ms(200);}}
      }
   if(count>=100)  {for(i=0;i<3;i++)    {set_led(0x3E0);    delay_ms(200);
set_led(0x3FF); delay_ms(200);}     state = kalibrasi;}
   }
while(!SW)  delay_ms(10);
count = 0;
i = 0;

while(state==kalibrasi)
   {
   delay_ms(10);
   raw_flux = read_adc(hall);
   center = read_adc(pot1);
   range = read_adc(pot2);
   min_flux = limit(0,(center - (range/2)),1023);
   max_flux = limit(0,(center + (range/2)),1023);

   led_map =
interpolasi(0,11,min_flux,max_flux,limit(min_flux,raw_flux,max_flux));
   set_led(get_map(led_map,1,1,0));

   if(!SW) {
      if(count<100)   {          //--------------------------->>> tekan lama  simpan zoom
            count++;
            set_led(get_map(count,10,1,0));
            }
         else       {
```

```
            Ecenter = center;
            Erange = range;
            state = set_SP;
            for(i=0;i<3;i++)    {
               set_led(0); delay_ms(200); set_led(0x3FF); delay_ms(200);
               if(i==3)    {
                  set_coil(0);
                  Eambiant[0] = raw_flux;
                  }
               }
            }
      }
   else if(count>0)    {          //--------------------------->>> tekan singkat
         state = set_SP;
         count = 0;
         }
   }
while(!SW)  delay_ms(10);
count = 0;
i = 0;

center = Ecenter;
range = Erange;
min_flux = limit(0,(center - (range/2)),1023);
max_flux = limit(0,(center + (range/2)),1023);

while(state==set_SP)
   {
   delay_ms(10);
```

```
raw_flux = read_adc(hall);
index = read_adc(pot1)/350;
led_map =
interpolasi(0,11,min_flux,max_flux,limit(min_flux,raw_flux,max_flux));
set_led(get_map(led_map,1,1,1));
if(!SW) {
   if(count<120)   {          //-------------------------->>> tekan lama masuk set_PD
         count++;
         if(count>20)    set_led(get_map(count-20,10,1,0));
         }
   else if(count>=120) {
         state = set_PD;
         for(i=0;i<=128;i++)    {
            set_coil(limit(0,(signed int)i*2,255));
            led_map = interpolasi(0,10,1,128,i);
            set_led(get_map(led_map,1,1,1));
            delay_ms(10);
            set_led(0x3FF); delay_ms(10);
            Eambiant[i] = raw_flux;
            }

         set_coil(0);
         }
   }
else if(count>0)    {          //-------------------------->>> tekan singkat
   if(index==2)       {
      Econtroller_const[_L] = raw_flux;
      for(i=0;i<3;i++)    {set_led(0x3F);     delay_ms(200); set_led(0x3FF);
delay_ms(200);}     //-->> batas bawah
```

```
        }
      else if(index==1)   {
        Econtroller_const[_S] = raw_flux;
        for(i=0;i<3;i++)    {set_led(0x387);    delay_ms(200); set_led(0x3FF);
delay_ms(200);}     //-->> center point
        }
      else if(index==0)   {
        Econtroller_const[_M] = raw_flux;
        for(i=0;i<3;i++)    {set_led(0x3F0);    delay_ms(200); set_led(0x3FF);
delay_ms(200);}     //-->> atas point
        }
      count = 0;
      }
   }
while(!SW)  delay_ms(10);
count = 0;
i = 0;

//-------------------------------------------------------->>>>>> inisialisasi konstanta

// copy eeprom
for(i=0;i<128;i++)   ambiant[i] = Eambiant[i];
controller_const[_P] = (float)Econtroller_const[_P]*Pscale;
controller_const[_D] = (float)Econtroller_const[_D]*Dscale;
controller_const[_L] = (float)Econtroller_const[_L];
controller_const[_S] = (float)Econtroller_const[_S];
controller_const[_M] = (float)Econtroller_const[_M];
ofset = Eofset;
center = Ecenter;
```

```
range = Erange;
min_flux = limit(0,(center - (range/2)),1023);
max_flux = limit(0,(center + (range/2)),1023);

point_type = (state==run) ? 1 : 0;
i=0;    count=0;
duty = 0;
set_coil(0);

abs_ofset = (controller_const[_M] - controller_const[_L])/2;
//-------------------------------------------------------->>>>>>> main loop
while (1)
    {
      if(state==set_PD)   {                    //--------->>> tuning konstanta PD
         controller_const[_P] =
LPF_smoot((float)read_adc(pot1)*Pscale,controller_const[_P],0.5);
         controller_const[_D] =
LPF_smoot((float)read_adc(pot2)*Dscale,controller_const[_D],0.5);
         if(!SW)    {
            if(count<100)   count++;
               else       {                 //--------->> tekan lama untuk menyimpan
                  Econtroller_const[_P] = (signed int)(controller_const[_P]/Pscale);
                  Econtroller_const[_D] = (signed int)(controller_const[_D]/Dscale);
                  state = run;
                  point_type = 1;
                  }
            }
         ofset = 0;
         }
```

```
    else if(state==set_SP)  {
        if(read_adc(pot2)>800)      ofset = LPF_smoot(interpolasiF(-
abs_ofset,abs_ofset,0,1024,(float)read_adc(pot1)),ofset,0.2);
        else if(read_adc(pot2)<200) {if(!SW) {Eofset = ofset;   state = run;
point_type = 1;}}
          else                  {if(!SW) {ofset = Eofset;   state = run;    point_type =
1;}}
        }
      else    {
        if((!SW)&(read_adc(pot2)>800))  {state = set_SP;    point_type = 2;}
        }

    amb = ambiant[(unsigned char)(duty/2)] - ambiant[0];
    raw_flux = LPF_smoot((float)read_adc(hall),raw_flux,Psmoot);
    ref  = raw_flux - amb;
    set = LPF_smoot(controller_const[_S]+ofset,set,0.1);
    error = (set-ref);
    vel = LPF_smoot(error - error2,vel,Dsmoot);
    error2 = error;

    duty = 0;
    if(ref<=controller_const[_M])     {
        if(raw_flux>=controller_const[_L])  duty = controller_const[_P]*error +
controller_const[_D]*vel;
        duty = limitF(0,duty,255);
        }
      else    {
        duty = 0;
        }
```

```
#pragma used+
signed int limit(signed int bawah,signed int nilai,signed int atas)
{
if(nilai<bawah)     nilai=bawah;
else if(nilai>atas) nilai=atas;
return nilai;
}
float limitF(float bawah,float nilai,float atas)
{
if(nilai<bawah)     nilai=bawah;
else if(nilai>atas) nilai=atas;
return nilai;
}
#pragma used-

#pragma used+
float interpolasiF(float min,float max,float atas,float bawah,float nilai)
{
float data;
data = (nilai - bawah) * (max - min);
data /= (atas - bawah);
data += min;
return data;
}
signed int interpolasi(signed int min,signed int max,signed int bawah,signed int
atas,signed int nilai)
{
//signed long int data;
//data = (signed long int)(nilai - bawah) * (max - min);
```

```
//data /= (atas - bawah);
//data += min;
//return (signed int)data;
return (signed
int)interpolasiF((float)min,(float)max,(float)atas,(float)bawah,(float)nilai);
}
#pragma used-

///******************** simple Low Pass Filter *****************///
#pragma used+
float LPF_smoot(float curent_data,float prev_data,float smoot)
{
float filtered_data;
if(smoot<=0.0)      filtered_data = prev_data;
else if(smoot<1.0)  filtered_data = (smoot*curent_data) + ((1-
smoot)*prev_data);
   else           filtered_data = curent_data;
return filtered_data;
}
#pragma used-
```

```
#pragma used+
unsigned int get_map(unsigned int value,unsigned int div,char justify,char
point)
{
unsigned char swapt = value/div;
unsigned int map_out;
if(point==0)    {
       if(justify==1)  return 0x03FF >> swapt;
          else       return 0x03FF << swapt;
       }
   else       {
       if(justify==1)  map_out = (0x0400 >> swapt);
          else       map_out = (0x0001 << swapt) >> 1;
       if(point==1)    return ~map_out;
          else       return map_out;
       }
}
#pragma used-
#pragma used+
void set_led(unsigned int led_out)
{
unsigned char port_buferC;
unsigned char port_buferD;
unsigned char port_buferB;
port_buferC = (led_out << 2) & 0x04;
port_buferD = (led_out << 1) & 0xFC;
port_buferB = (led_out >> 7) & 0x07;
PORTC = (PORTC & 0xFB) | port_buferC;
PORTD = (PORTD & 0x03) | port_buferD;
```

```
PORTB = (PORTB & 0xF8) | port_buferB;
}
#pragma used-
```