To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

H8/300L SLP Series

Direction Finder (DirFind)

Introduction

This application note demonstrates how to interface the H8/38024 SLP series to an analog hall-effect sensor. The sensor outputs a sine-cosine curve pair, which is converted into directional information by the MCU and output to an alphanumeric dot-matrix display. This simple system can be used as a compass or direction finder. Advantages of using the H8/38024 SLP MCU are the many built-in peripherals such as the A/D converter, Serial Communication Interface (SCI), timers, etc.

Target Device

H8/38024

Contents

1. System Overview

Figure 1 shows the block diagram of the compass described in this application note. The system is comprised of the following components:

- H8/38024 SLP MCU
- Dinsmore Analog Hall-Effect Sensor (No. 1525)
- Alphanumeric dot-matrix display
- RS-232C Interface

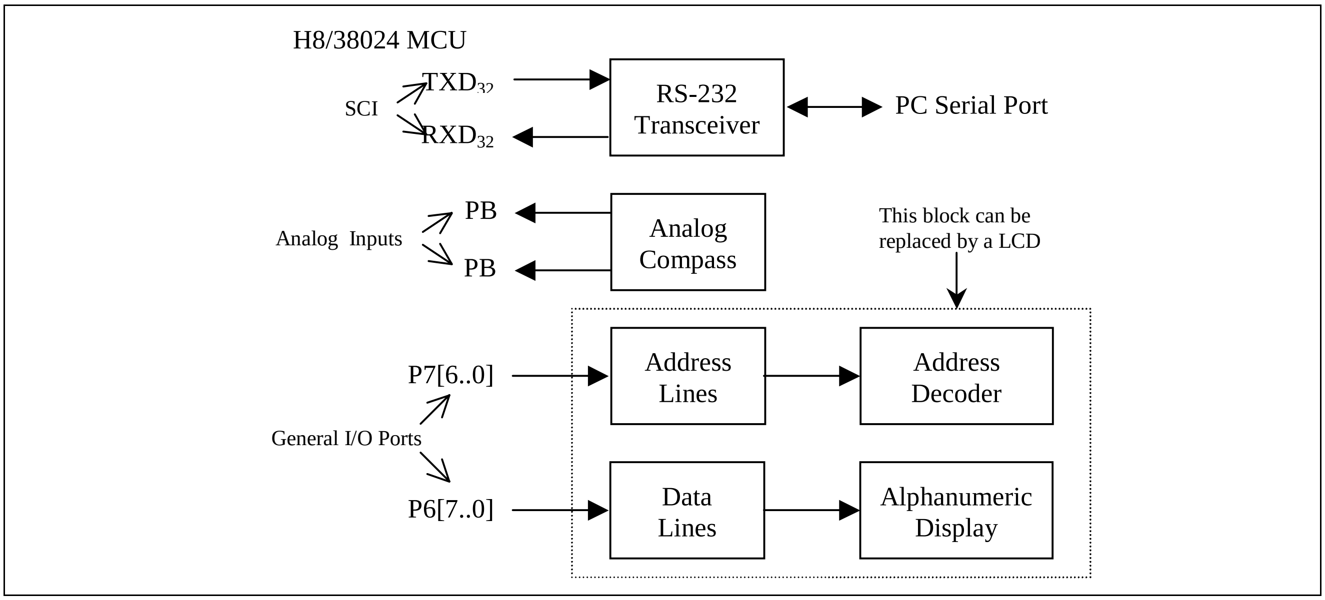


Figure 1 System Block Diagram

In this application note, the roles of the MCU are as follows:

- Convert the outputs from the analog sensor into digital equivalent. Two out of eight available analog input channels with 10-bit resolution are used to interface to the analog sensor.
- Determine the angle
- Display directional information
- Send directional information to the RS-232C serial port (debugging)

2. Hardware Implementation

The peripheral usage is as follows:

Table 1 Peripheral Usage

Peripheral	Function
Port 6[7..0]	“Data” bus
Port 7[6..0]	“Address” bus
P77	WRITE/READ_N
SCI (TXD_{32} and RXD_{32})	Communication with host PC
P43	Calibration pin for compass, low when done
PB0 and PB1	Outputs 1 and 2 of the analog sensor, respectively

2.1 Power Supplies

Three separate supplies are required in this application example:

- 6 V input voltage
- 5 V for 74HCT138, level shifter, alphanumeric display
- 3.3 V for the MCU, RS-232C transceiver, level shifter

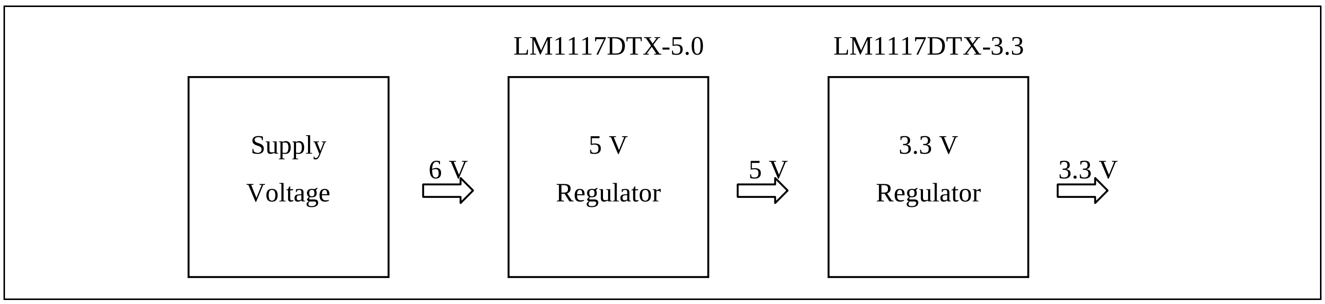


Figure 2 Power Supplies

2.2 Address and Data Buses

To allow the H8/38024 SLP MCU to access memory-mapped external devices/memory/peripherals, separate address and data buses with control signal WRITE/READ_N are constructed using general I/O ports as shown in figure 3. The MCU is operating at 3.3 V while some devices operate at 5 V. The function of the level shifters is to interface the MCU to the 5 V devices.

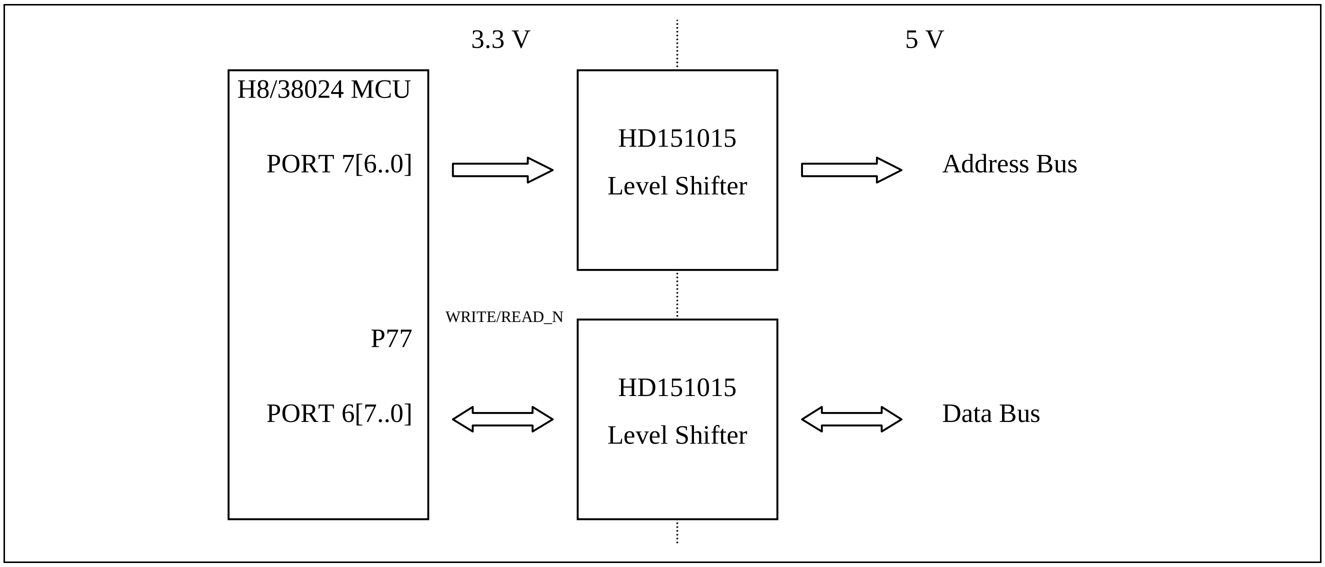


Figure 3 Address and Data Buses

2.3 Address Decoder

A 74HCT138 3 to 8 line decoder is used here to select the memory-mapped alphanumeric display (DLR1414). Table 2 lists the addresses.

Table 2 Address Mapping

Address (hexadecimal)	Device
C0	Display (first digit)
C1	Display (second digit)
C2	Display (third digit)
C3	Display (fourth digit)

2.4 RS-232C Transceiver

The Serial Communication Interface (SCI) pins TXD_{32} and RXD_{32} are connected to the Sipex SP3232 RS-232C transceiver. This allows the MCU to communicate with the Host PC.

2.5 Analog Hall-effect Sensor

The Dinsmore analog Hall-effect sensor 1525 requires closely regulated 5.00-V DC input and furnishes a ratiometric DC output. Power consumption is approximately 18 to 19 mA. The output closely resembles a sine-cosine set of curves shown in figure 4, which cross at 2.5 V and peak at 2.9 V and floor at about 2.1 V. Since the MCU is operating with AVCC = 3.3 V, these outputs can be directly connected to the inputs of the A/D converter.

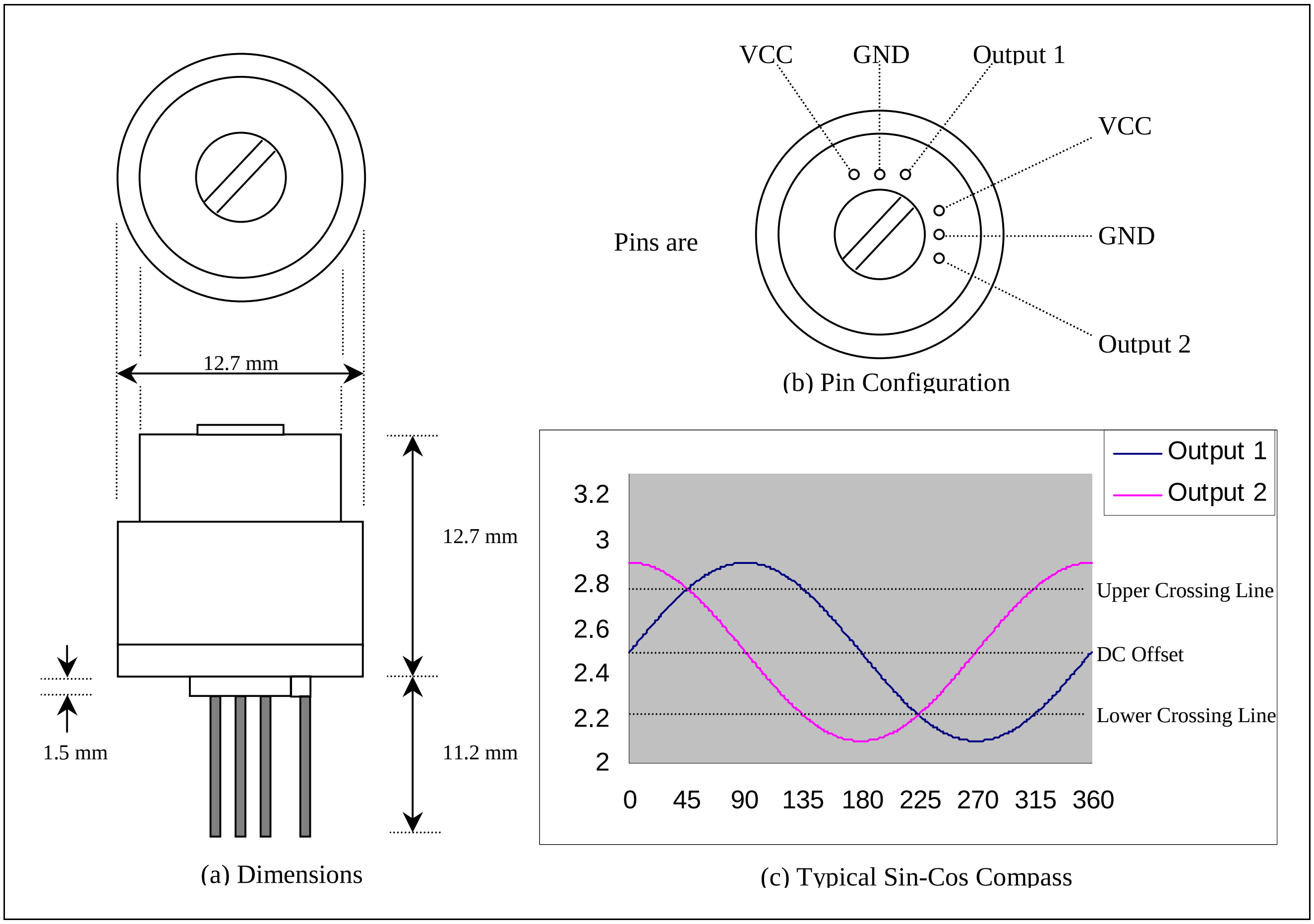


Figure 4 Dinsmore Analog Hall-effect Sensor (#1525)

- Output 1 = Amplitude·sin (ωt) + DC Offset
- Output 2 = Amplitude·cos (ωt) + DC Offset

Compare the portions of the curves above and below their crossing points where the sine and cosine curves are equal. Note that the curves are relatively straight and steep between the upper and lower crossing lines. For example, the relatively straight portions of the sine curve are from 0° to 45°, 135° to 225° and 315° to 360°. For the cosine curve, they are from 45° to 135° and 225° to 315°.

Range		**Curve with relatively straight portions**
From	**To**	
0°	45°	Sine
45°	135°	Cosine
135°	225°	Sine
225°	315°	Cosine
315°	360°	Sine

Since the arc sine and cosine of a particular value corresponds to two or more angles, the second output from the sensor is used to differentiate between the angles. Before the arc functions can be used, the sine and cosine curves must be normalised i.e., the DC offsets are subtracted from the curves.

- $\text{DC Offset} = \left(\dfrac{\text{Maximum} + \text{Minimum}}{2}\right)$

- $\text{Amplitude} = \left(\dfrac{\text{Maximum} - \text{Minimum}}{2}\right)$

- $\text{Angle(radians)} = \sin^{-1}\left(\dfrac{\text{Output 1} - \text{DC Offset}}{\text{Amplitude}}\right) \text{ or } \cos^{-1}\left(\dfrac{\text{Output 2} - \text{DC Offset}}{\text{Amplitude}}\right)$

- $\text{Angle(degrees)} = \left[\text{Angle(radians)} \times \dfrac{180}{\pi}\right]$

2.6 A/D Converter Interfacing Techniques

Some recommended A/D converter interfacing guidelines described in the "Application Note on Detailed Usage of A/D converter" are adopted here.

- A simple method for controlling noise is to have separate supplies for the (i) slower low-current analog functions and (ii) faster medium-power digital functions as shown in figure 5. The ferrite beads together with the bypass capacitors form a low-pass filter network, reducing the high-frequency noise. It resists varying current and also provides a low-impedance AC short to ground on either side.

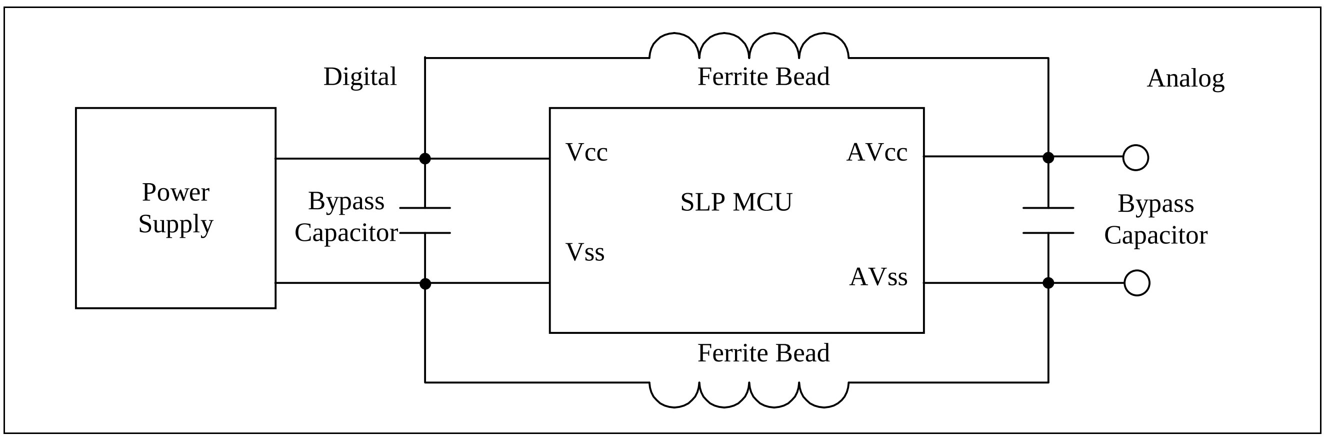


Figure 5 Split Supplies

- A low-impedance buffer should be inserted when converting a high-speed analog signal as shown in figure 6. The buffer provides a high-impedance for the sensor and a low-impedance drive for the A/D converter.

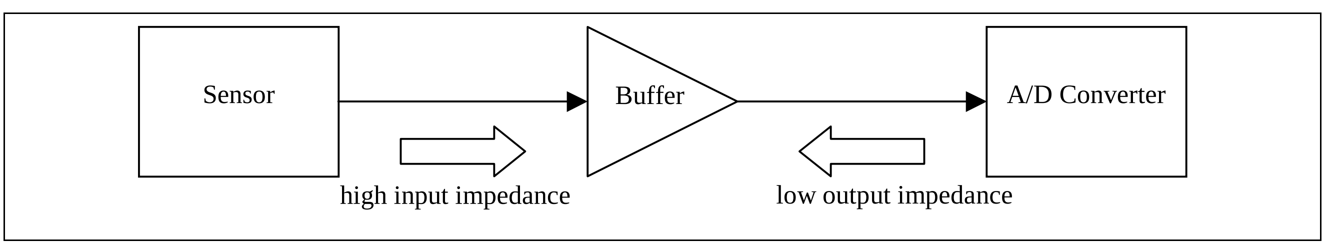


Figure 6 Signal Isolation

- All unused analog input pins are also pulled to AVCC.

3. Software Implementation

The source code is written in the C language for easy implementation and compiled using the free H8 Tiny/SLP toolchain (version 5.0.0) for HEW version 2.2 (Release 15).

The functions of the source codes are as follows:

- Initialization of I/O ports, A/D converter and serial ports.
- Calibration
  - The average DC offset and amplitude of the output waveforms have been stored in the program as "offset_value" and "amplitude_value", respectively.
  - If calibration is required, remove the jumper connecting pins 2 and 3 of J3. The analog sensor must be rotated at least one complete revolution to get the new maximum and minimum output values. When calibration is done, put jumper across pins 2 and 3 of J3. From these values, the amplitude is calculated.
- Conversion of analog inputs to angle representations. These angle representations will be output to the alphanumeric display and transmitted to the PC via the SCI. Use terminal emulating software such as the HyperTerminal or Tera Term Pro to configure the settings for the PC serial port as 38400bps, 8 data bits, 1 stop bit and parity disabled.

The flowchart of the main() function is described in figure 7.

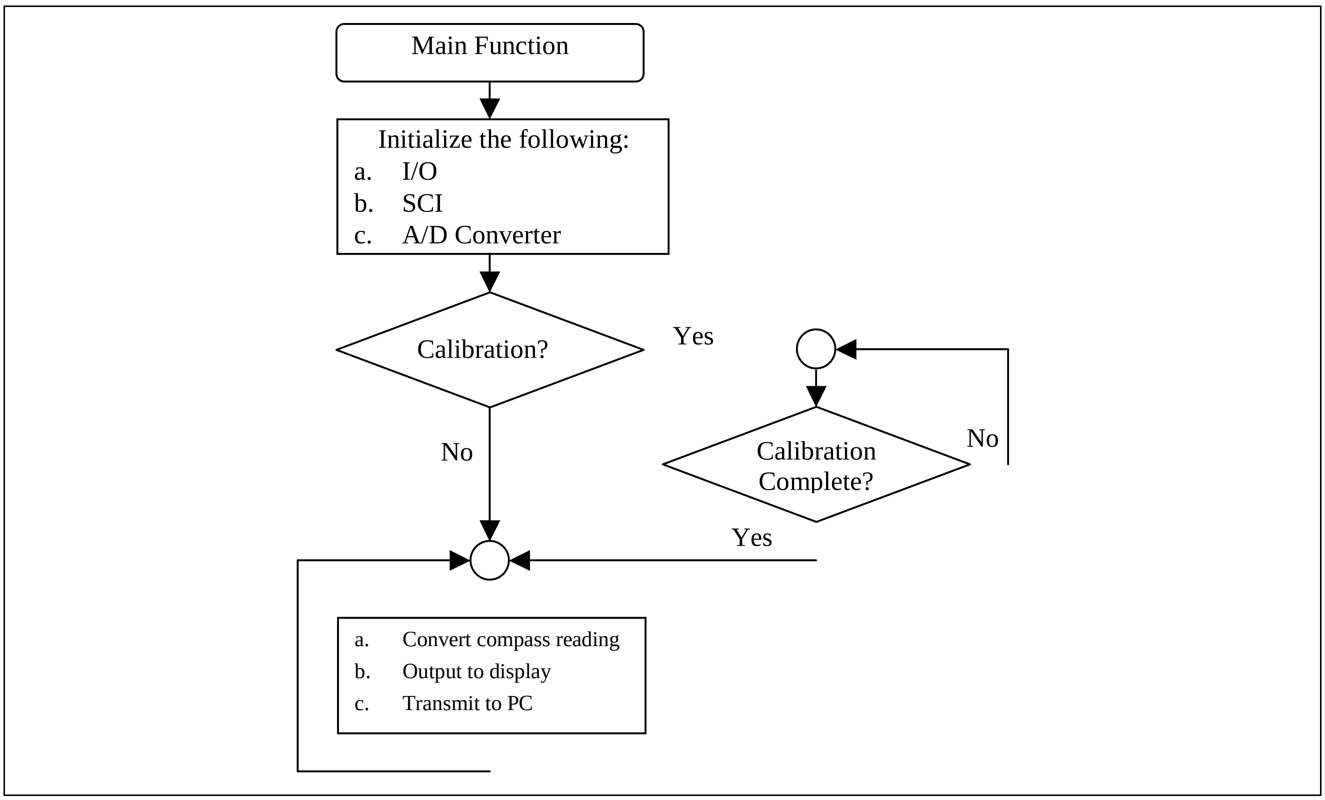


Figure 7 Flowchart

The prototype board is connected to the ALE300L-H8/3800 low-cost emulator as shown in figure 8.

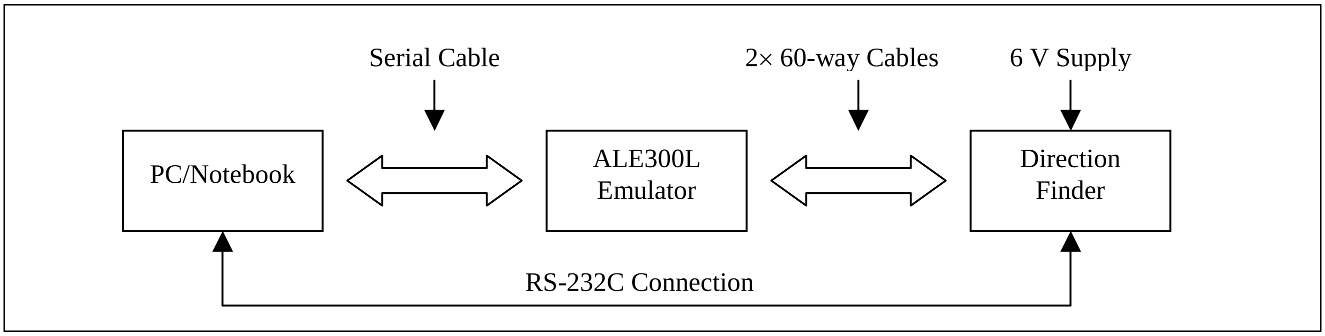


Figure 8 System Setup

3.1 Source Codes

The source codes are mainly in DirFind.c.

- Contains the main function
- Performs initialization of the I/O, Serial Communication Interface (SCI) and A/D converter
- Calibration for compass
- Conversion of compass reading

```
/***********************************************************************/
/*                                                                     */
/*  FILE        :DirFind.c                                             */
/*  DATE        :Mon, Feb 09, 2004                                     */
/*  DESCRIPTION :Main Program                                          */
/*  CPU TYPE    :H8/3802                                               */
/*                                                                     */
/*  This file is generated by Renesas Project Generator (Ver.2.1).     */
/*                                                                     */
/***********************************************************************/

#ifdef __cplusplus
extern "C" {
#endif
void abort(void);
#ifdef __cplusplus
}
#endif

#include "iodefine.h"
#include <machine.h>
#include <math.h>

//----------------------------------------------------------------------
//Constant Declarations
//----------------------------------------------------------------------

//Constants for Address Decoder
#define first_digit         0xC0
#define second_digit        0xC1
#define third_digit         0xC2
#define fourth_digit        0xC3

#define  de_select          0x3F

//ASCII Constants
#define  blank              0x20

//Others
#define  address_bus        P_IO.PDR7.BYTE          //Address Bus
#define  data_bus           P_IO.PDR6.BYTE          //Data Bus
#define  calibration_input  P_IO.PDR4.BIT.P43       //Calibration Input

#define pi                  3.141592

//Average sensor values computed
//If calibration is not performed, then these values will be used
#define  offset_value       786.25
#define  amplitude_value    132.8
```

```
//----------------------------------------------------------------------
//Function Prototypes
//----------------------------------------------------------------------
void init_sci(void);
void char_put(char);
void PutStr(char *);

void init_port(void);

void display_char(unsigned char, unsigned char);

void display_word(unsigned int);

void serial_transmit(unsigned int);

void init_adc(unsigned char);
void start_adc(unsigned char);

unsigned int ADC_value (void);
void find_value(void);

void delay(void);

//----------------------------------------------------------------------
//global variables
//----------------------------------------------------------------------

unsigned int   max, min;

//----------------------------------------------------------------------

void main(void)
{
   //-----------------------------------------------------------

   unsigned int   f1, f2, f3, direction, directionS, directionC;
   unsigned int   sin_value, cos_value;

   float          output1, output2;
   float          degreeS, degreeC;
   float          dc_offset, amplitude;

   //-----------------------------------------------------------

   init_port();

   init_sci();

   //-----------------------------------------------------------

   dc_offset = offset_value;
   amplitude = amplitude_value;
```

```
    display_char(fourth_digit, blank);          //Blank display
    display_char(third_digit, blank);           //Blank display
    display_char(second_digit, blank);          //Blank display
    display_char(first_digit, blank);           //Blank display

    //-----------------------------------------------------------
    //This loop is required for calibration.
    //Need to rotate the compass first for at least 1 full revolution
    //Calibration is considered "done" when the jumper is shorted

    if (calibration_input)
    {
       display_char(fourth_digit, 0x43);  //Display 'C'

       min = 1024;
       max = 0;

       do
       {
          find_value();
       } while (calibration_input == 1);

       dc_offset = (max + min)/2;
       amplitude = (max - dc_offset);

       delay();
       display_char(fourth_digit, blank); //End of calibration : blank
    }

    //-----------------------------------------------------------

    while(1)
    {
       init_adc(0);                       //Initialize ADC

       start_adc(1);
       while (P_AD.ADSR.BYTE & 0x80);     //If ADSR = 1, A/D conversion in
 progress
       sin_value = ADC_value();

       init_adc(1);                       //Initialize ADC

       start_adc(1);

       while (P_AD.ADSR.BYTE & 0x80);     //If ADSR = 1, A/D conv in progress
       cos_value = ADC_value();

       //Calculation of Degree
       //range for asin() is -90 to 90
       //range for acos() is 0 to 180
       output1 = (sin_value - dc_offset) / amplitude;
       degreeS = asin(output1) * 180.0 / pi;
```

```
output2 = (cos_value - dc_offset) / amplitude;
degreeC = acos(output2) * 180.0 / pi;

//----------------------------------------------------------

if (cos_value >= dc_offset)       //1st or 4th quadrant
{
   if (degreeS > 0)               //1st quadrant : 0 to 90
   {
      directionS = degreeS;
   }
   else
   {
      directionS = 360 + degreeS; //4th quadrant : 270 to 360
   }
}
else
{
   directionS = 180 - degreeS;    //2nd & 3rd quadrants : 90 to 270
}

//----------------------------------------------------------

if (sin_value >= dc_offset)       //1st & 2nd quadrants : 0 to 180
{
   directionC = degreeC;
}
else
{
   if (degreeC < 0)
   {
      directionC = 180 - degreeC; //3rd quadrant : 180 to 270
   }
   else
   {
      directionC = 360 - degreeC; //4th quadrant : 270 to 360
   }
}

//----------------------------------------------------------

if (((directionC > 45) && (directionC < 135)) ||
   ((directionC > 225) && (directionC < 315)))
   direction = directionC;
else if (((directionS >= 135) && (directionS <= 225)) ||
         ((directionS >= 315) || (directionS <= 45)))
   direction = directionS;

//----------------------------------------------------------

f1 = floor(direction / 100);
f2 = floor((direction - (100 * f1)) / 10);
f3 = floor(direction - (f1 * 100) - (f2 * 10));
```

```
        display_char(third_digit, f1 + 0x30);
        char_put(f1 + 0x30);
        display_char(second_digit, f2 + 0x30);
        char_put(f2 + 0x30);
        display_char(first_digit, f3 + 0x30);
        char_put(f3 + 0x30);
        PutStr("\r\n");
        delay();
    }
}

//------------------------------------------------------------------------

/*
   init_port() : Set up the I/O ports

   a. Port 6[7..0] -> Data[7..0]
   b. Port 7[7..0] -> Address[7..0]
      Note that Port 7_7 also functions as the WRITE/READ_N signal
*/

void init_port(void)
{
   P_LCD.LPCR.BYTE = 0x00;              //SEG[32..1] as I/O Port

   P_IO.PCR6.BYTE  = 0xFF;              //Set Port 6 as all output

   P_IO.PCR7.BYTE  = 0xFF;              //Set Port 7 as all output

   data_bus       = 0xFF;               //Set Data Bus to all '1'

   address_bus    = 0xFF;               //Set Address Bus to all '1'
}

//------------------------------------------------------------------------

/*
   display_char()

   a. Port 6[7..0] -> Data[7..0]
   b. Port 7[7..0] -> Address[7..0]
      Note that Port 7_7 also functions as the WRITE/READ_N signal
*/

void display_char(unsigned char digit_position, unsigned char digit_info)
{
   P_IO.PCR6.BYTE  = 0xFF;              //Set Port 6[7..0] as output

   address_bus &= de_select;

   data_bus = digit_info;               //Data

   address_bus = digit_position;        //Address
```

```
   address_bus &= de_select;
}

//-----------------------------------------------------------------------

/*
   display_word()
*/

void display_word(unsigned int display_data)
{
   unsigned char  position, digit_info, digit_position;

   P_IO.PCR6.BYTE  = 0xFF;               //Set Port 6[7..0] as output

   for (position = 4 ; position != 0 ; position--)
   {
      switch (position)
      {
         case 1:
            digit_position = first_digit;
            digit_info = (unsigned char)(display_data & 0x000F);
            break;

         case 2:
            digit_position = second_digit;
            digit_info = (unsigned char)((display_data & 0x00F0) >> 4);
            break;

         case 3:
            digit_position = third_digit;
            digit_info = (unsigned char)((display_data & 0x0F00) >> 8);
            break;

         default:
            digit_position = fourth_digit;
            digit_info = (unsigned char)((display_data & 0xF000) >> 12);
            break;
      }

      if ((digit_info >= 0) && (digit_info <= 9))
         digit_info += 0x30;
      else
      {
         if ((digit_info >= 0xA) && (digit_info <= 0xF))
         {
            digit_info -= 0xA;
            digit_info += 0x41;
         }
      }

      address_bus &= de_select;
```

```
      data_bus = digit_info;              //Data

      address_bus = digit_position;       //Address

      address_bus &= de_select;
   }
}

//------------------------------------------------------------------------

/*
   serial_transmit()
*/

void serial_transmit(unsigned int display_data)
{
   unsigned char  position, digit_info;

   for (position = 4 ; position != 0 ; position--)
   {
      switch (position)
      {
         case 1:
            digit_info = (unsigned char)(display_data & 0x000F);
            break;

         case 2:
            digit_info = (unsigned char)((display_data & 0x00F0) >> 4);
            break;

         case 3:
            digit_info = (unsigned char)((display_data & 0x0F00) >> 8);
            break;

         default:
            digit_info = (unsigned char)((display_data & 0xF000) >> 12);
            break;
      }

      if ((digit_info >= 0) && (digit_info <= 9))
         digit_info += 0x30;
      else
      {
         if ((digit_info >= 0xA) && (digit_info <= 0xF))
         {
            digit_info -= 0xA;
            digit_info += 0x41;
         }
      }

      char_put(digit_info);
   }
}
```

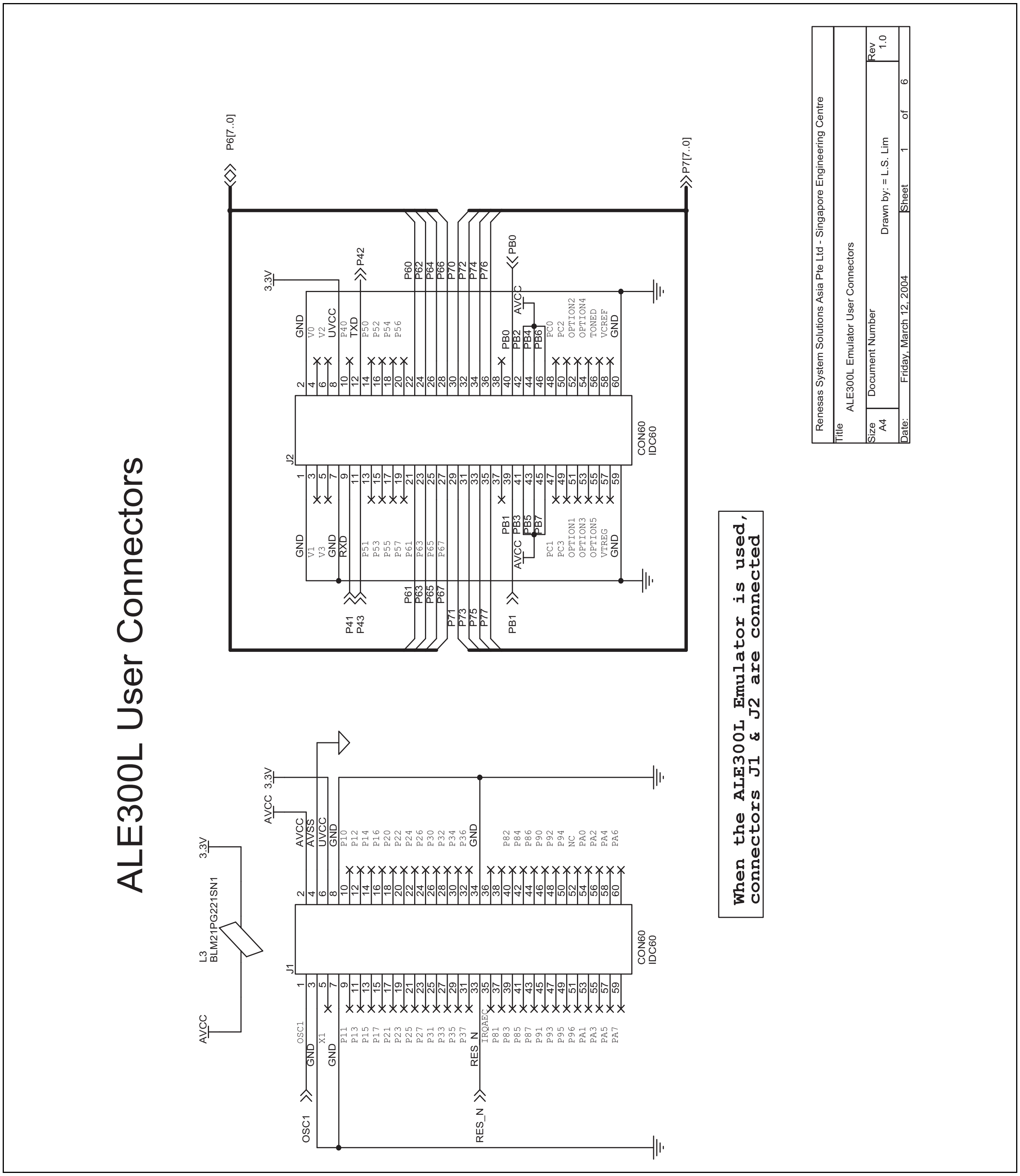
```
//------------------------------------------------------------------------

/*
   init_sci() : Sets up the Serial Communication Interface for debugging
                purposes.
*/

void init_sci(void)
{
   //SCR3 : |TIE|RIE|TE|RE|MPIE|TEIE|CKE1|CKE0|
   //TIE : Transmit interrupt enable
   //RIE : Receive interrupt enable
   //TE  : Transmit enable
   //RE  : Receive enable
   //MPIE : Multiprocessor interrupt enable
   //TEIE : Transmit end interrupt enable
   //CKE1 : Clock enable 1
   //CKE0 : Clock enable 0

   //CKE1 = CKE0 = 0
   //asynchronous mode, internal clock source, SCK32 functions as I/O port
   P_SCI3.SCR3.BYTE &= 0x00; //clear TE & RE

   //SMR : |COM|CHR|PE|PM|STOP|MP|CKS1|CKS0| : |0|0|0|0|0|0|0|0|
   //COM : Communication Mode  : 0 : asynchronous mode
   //CHR : Character Length    : 0 : character length = 8 bits
   //PE  : Parity Enable       : 0 : parity bit addition and checking disabled
   //PM  : Parity Mode         : 0 : even parity (no effect since parity is
already disabled)
   //STOP: Stop Bit Length     : 0 : 1 stop bit
   //MP  : Multiprocessor Mode : 0 : multiprocessor communication function
disabled
   //|CKS1|CKS0| : Clock Select: |0|0| : clock source for baud rate generator
= clk
   P_SCI3.SMR.BYTE = 0x00;

   //For clk = 10MHz, bit rate = 2400 bps, n = 0, N = 64
   //P_SCI3.BRR = 64;

   //For clk = 10MHz, bit rate = 38400 bps, n = 0, N = 3
   P_SCI3.BRR = 3;

   //minimum of 1-bit delay = 417ns
   nop();
   nop();
   nop();

   //SPCR : |---|---|SPC32|---|SCINV3|SCINV2|---|---| : |1|1|1|0|0|0|0|0|
   //SPC32 = 1 : P42 functions as TXD32 output pin
   //need to set TE bit in SCR3 after setting this bit to 1
   //SCINV3 = 0 : TXD32 output data is not inverted
   //SCINV2= 0 : RXD32 input data is not inverted
   //Bits 7 and 6 are reserved and always read as 1
```

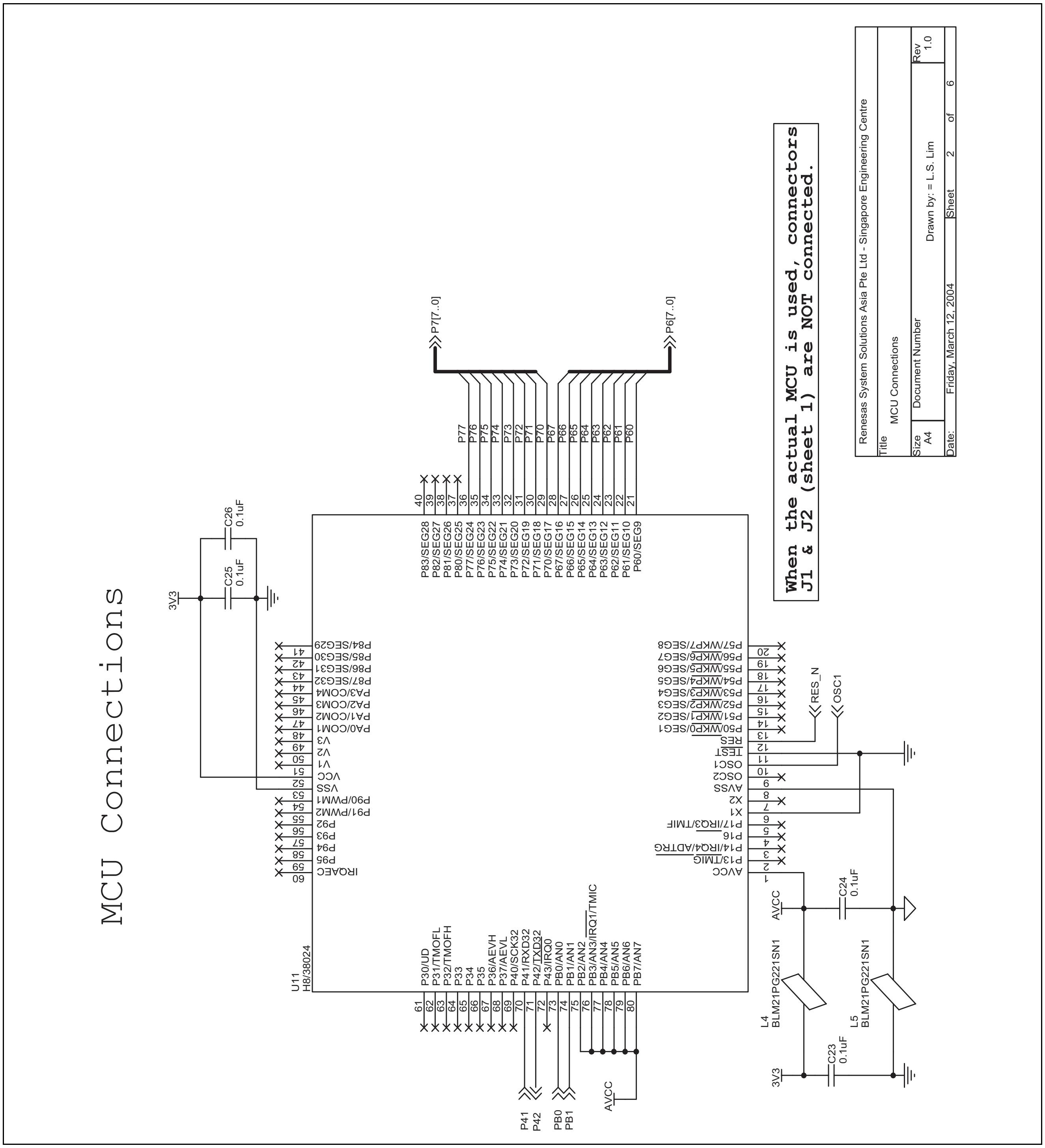
```
   //Bits 4, 1 and 0 are reserved and only 0 can be written to these bits
   P_SCI3.SPCR.BYTE = 0xE0;

   P_SCI3.SCR3.BYTE |= 0x30; //Set TE & RE
}

//-----------------------------------------------------------------------

/*
   char_put() : Transmits a character to the PC for debugging purposes.
*/

void char_put(char OutputChar)              //Serial Port
{
    //SSR  : |TDRE|RDRF|OER|FER|PER|TEND|MPBR|MPBT|
   //TDRE : transmit data register empty
   //RDRF : receive data register full
   //OER  : overrun error
   //FER  : framing error
   //PER  : parity error
   //TEND : transmit end
   //MPBR : Multiprocessor bit receive
   //MPBT : Multiprocessor bit transfer

   while ((P_SCI3.SSR.BIT.TDRE) == 0);        //Wait for TDRE = 1

   P_SCI3.TDR = OutputChar;

   while ((P_SCI3.SSR.BIT.TEND) == 0);        //Wait for TEND = 1

   P_SCI3.SSR.BIT.TEND = 0;
}

//-----------------------------------------------------------------------

/*
   PutStr() : Transmits a string of characters to the PC for debugging
purposes.
*/

void PutStr(char *str)
{
    while (*str != 0)
   {
      char_put(*str++);
   }
}

//-----------------------------------------------------------------------

/*
   init_adc()
*/
```

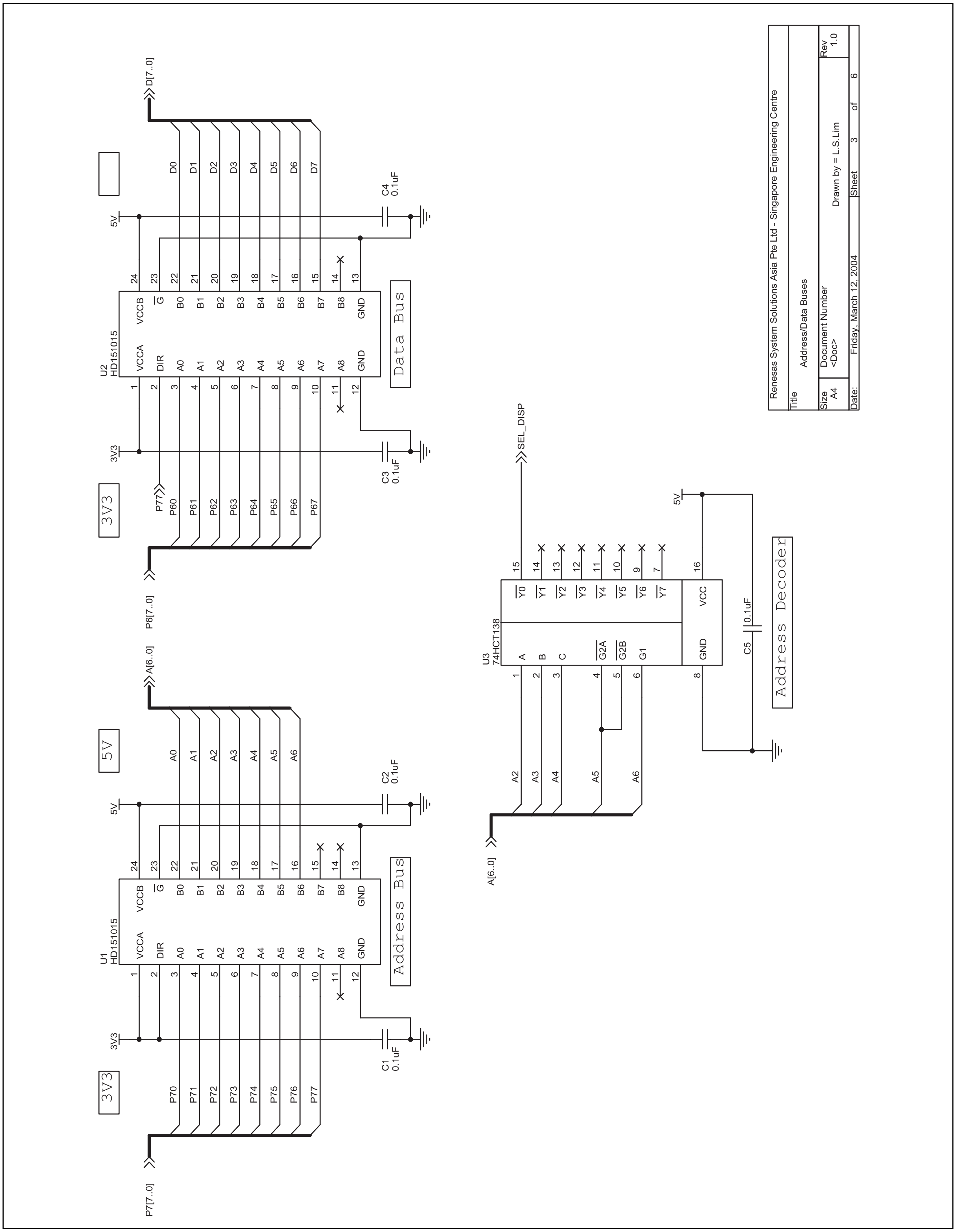
```
void init_adc(unsigned char Input_CH)
{
   //CKS = 0 -> A/D Conversion period = 62/phi
   P_AD.AMR.BIT.CKS= 0;

   //TRGE = 0 -> Disable start of A/D conversion by external trigger
   P_AD.AMR.BIT.TRGE = 0;

   //Input_CH = 0-7 => Select ADC input channel
   P_AD.AMR.BIT.CH = (Input_CH + 4);

   //ADC Module standby mode is cleared
   P_SYSCR.CKSTPR1.BIT.ADCKSTP = 1;
}

//-----------------------------------------------------------------------

/*
   start_adc(start)

*/

void start_adc(unsigned char start)
{
   if (start == 1)
      P_AD.ADSR.BYTE |= 0x80;           //Set ADSF : start A/D conversion
   else
      P_AD.ADSR.BYTE &= 0x7F;           //Set ADSF : stop A/D conversion
}

//-----------------------------------------------------------------------

/*
   ADC_value()
*/

unsigned int ADC_value(void)
{
   unsigned int   adrrL, adrrH;
   unsigned int   valueL,valueH;
   unsigned int   D_value;

   adrrH = P_AD.ADRR >> 8;              //Capture the ADC value from AN#
   adrrL = P_AD.ADRR << 8;
   valueH = adrrH << 2;
   valueL = adrrL >> 14;
   D_value = valueL | valueH;
   return (D_value);
}

//-----------------------------------------------------------------------

/*
```

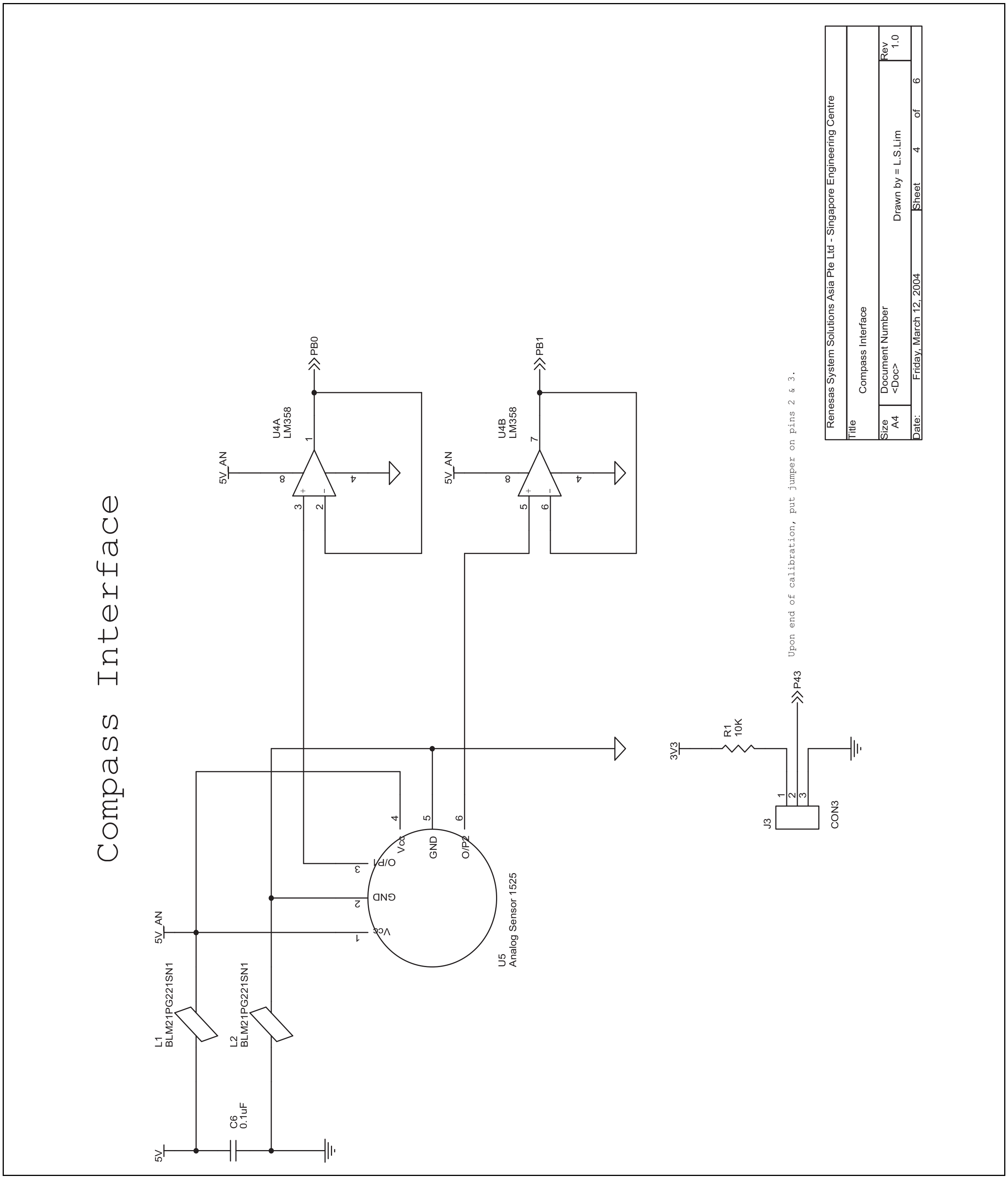
```
   find_value() - find the maximum and minimum values of output1
*/

void find_value(void)
{
   unsigned int   temp_output1;

   init_adc(0);                   //Initialize ADC channel 0

   start_adc(1);                  //Start ADC

   while (P_AD.ADSR.BYTE & 0x80); //If ADSR = 1, A/D conversion in progress

   temp_output1 = ADC_value();

   if (temp_output1 > max)
   {
      max = temp_output1;
   }

   if (ADC_value() < min)
   {
         min = temp_output1;
   }
}

//----------------------------------------------------------------------

/*
   delay() - software delay routine
*/

void delay(void)
{
   unsigned int   delay_loop;

   for (delay_loop = 0 ; delay_loop < 30000 ; delay_loop++);
}

//----------------------------------------------------------------------

void abort(void)
{

}
//----------------------------------------------------------------------
```

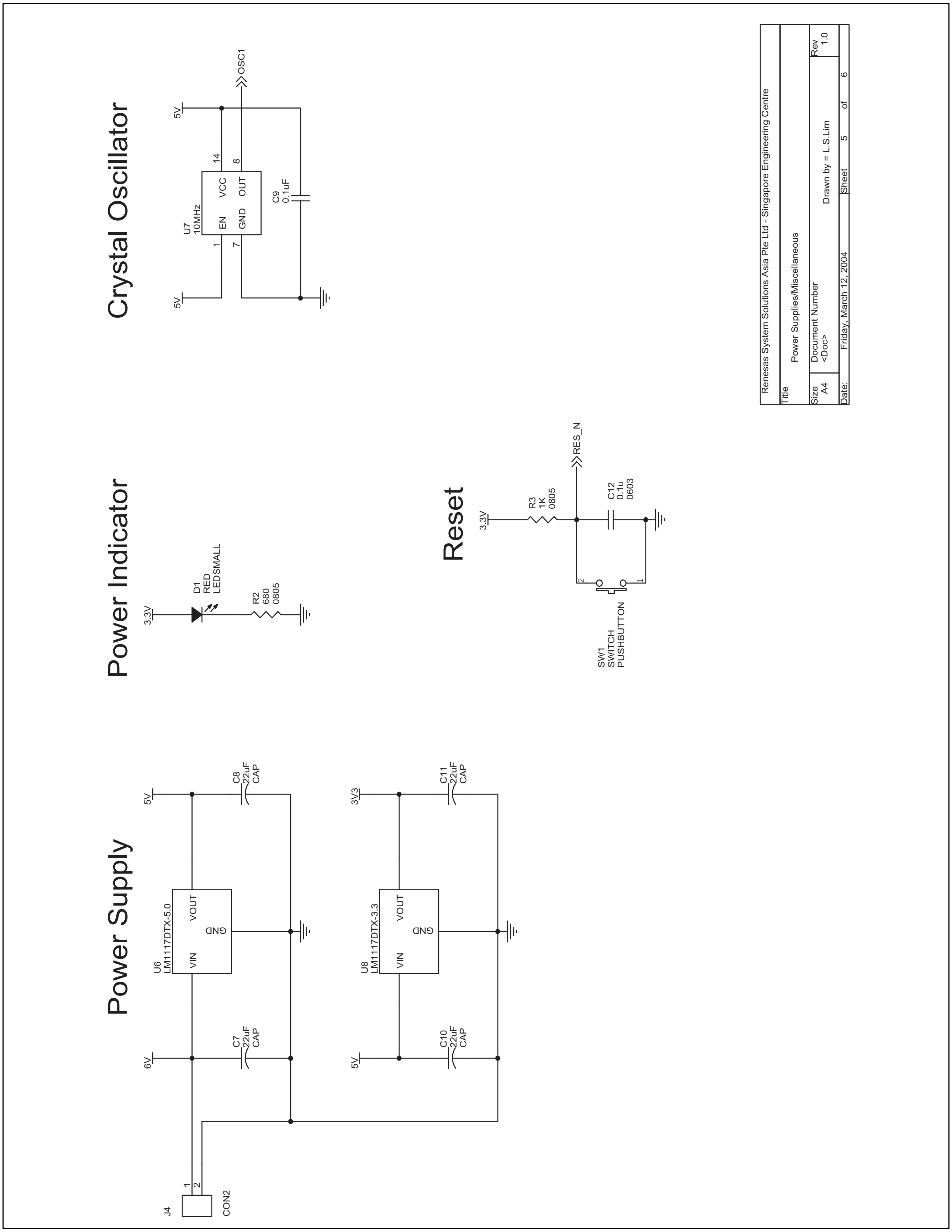
4. Hardware Schematics

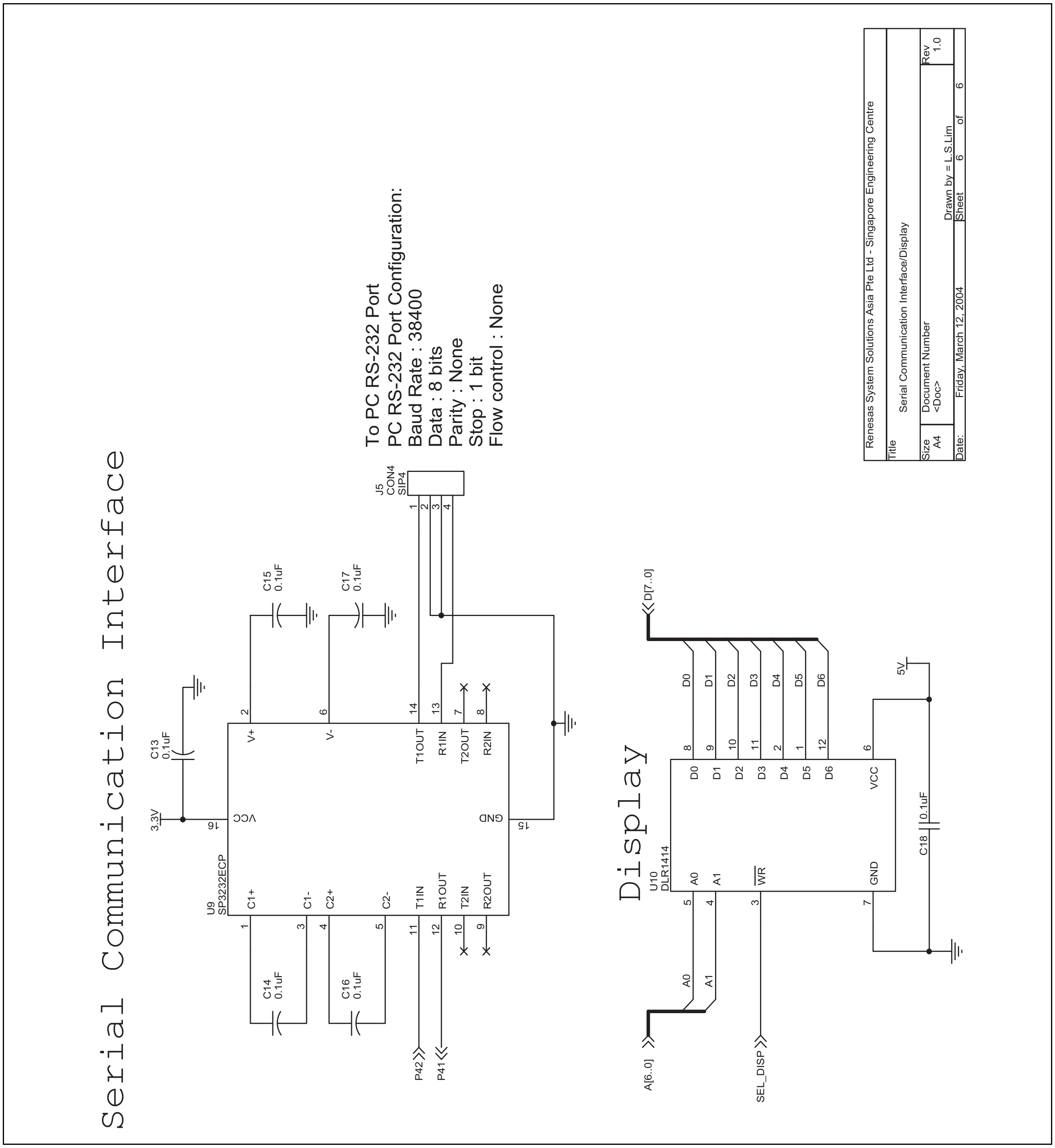



MCU Connections
When the actual MCU is used, connectors J1 & J2 (sheet 1) are NOT connected.
Renesas System Solutions Asia Pte Ltd - Singapore Engineering Centre
Title
MCU Connections
Size
A4
Document Number
Rev
1.0
Drawn by: = L.S. Lim
Date: Friday, March 12, 2004
Sheet 2 of 6
U11
H8/38024
3V3
C25
0.1uF
C26
0.1uF
C23
0.1uF
C24
0.1uF
AVCC
L4
BLM21PG221SN1
L5
BLM21PG221SN1
RES_N
OSC1
P7[7..0]
P6[7..0]
P41
P42
PB0
PB1


Data Bus
Address Bus
Address Decoder
U1
HD151015
U2
HD151015
U3
74HCT138
D[7..0]
P6[7..0]
A[6..0]
P7[7..0]
SEL_DISP
Renesas System Solutions Asia Pte Ltd - Singapore Engineering Centre
Title
Address/Data Buses
Size
A4
Document Number
<Doc>
Drawn by = L.S.Lim
Rev
1.0
Date:
Friday, March 12, 2004
Sheet
3
of
6


Compass Interface
5V
C6
0.1uF
L1
BLM21PG221SN1
L2
BLM21PG221SN1
5V_AN
U5
Analog Sensor 1525
Vcc
GND
O/P1
O/P2
U4A
LM358
U4B
LM358
PB0
PB1
3V3
R1
10K
P43
J3
CON3
Upon end of calibration, put jumper on pins 2 & 3.
Renesas System Solutions Asia Pte Ltd - Singapore Engineering Centre
Title
Compass Interface
Size
A4
Document Number
<Doc>
Drawn by = L.S.Lim
Rev
1.0
Date:
Friday, March 12, 2004
Sheet 4 of 6


Crystal Oscillator
Power Indicator
Reset
Power Supply
5V
U7
10MHz
EN
VCC
GND
OUT
1
14
7
8
OSC1
C9
0.1uF
3.3V
D1
RED
LEDSMALL
R2
680
0805
R3
1K
0805
RES_N
C12
0.1u
0603
SW1
SWITCH
PUSHBUTTON
6V
3V3
U6
LM1117DTX-5.0
VIN
VOUT
GND
C7
22uF
CAP
C8
22uF
CAP
U8
LM1117DTX-3.3
C10
22uF
CAP
C11
22uF
CAP
J4
CON2
Renesas System Solutions Asia Pte Ltd - Singapore Engineering Centre
Title
Power Supplies/Miscellaneous
Size
A4
Document Number
<Doc>
Rev
1.0
Drawn by = L.S.Lim
Date:
Friday, March 12, 2004
Sheet
5
of
6


Serial Communication Interface
To PC RS-232 Port
PC RS-232 Port Configuration:
Baud Rate : 38400
Data : 8 bits
Parity : None
Stop : 1 bit
Flow control : None
J5
CON4
SIP4
U9
SP3232ECP
C13
0.1uF
C14
0.1uF
C15
0.1uF
C16
0.1uF
C17
0.1uF
3.3V
VCC
GND
V+
V-
C1+
C1-
C2+
C2-
T1OUT
R1IN
T2OUT
R2IN
T1IN
R1OUT
T2IN
R2OUT
P42
P41
Display
U10
DLR1414
D[7..0]
D0
D1
D2
D3
D4
D5
D6
A0
A1
WR
5V
C18
0.1uF
A[6..0]
SEL_DISP
Renesas System Solutions Asia Pte Ltd - Singapore Engineering Centre
Title
Serial Communication Interface/Display
Size
A4
Document Number
<Doc>
Rev
1.0
Date:
Friday, March 12, 2004
Drawn by = L.S.Lim
Sheet
6
of
6

5. References

1. Dinsmore Sensing Systems General Information, Dinsmore Sensors (http://www.dinsmoresensors.com/).
2. LM1117/LM1117I 800mA Low-Dropout Linear Regulator, 2002, National Semiconductor Corporation.
3. LM158/LM258/LM358/LM2904 Low Power Dual Operational Amplifiers, 2002, National Semiconductor Corporation.
4. HD151015 9-bit Level Shifter/Transceiver with 3 State Outputs, 3rd Edition, June 1993, Renesas Technology Corp. (ref. no.: REJ03D0300-0400, http://renesas.com)
5. Detailed Usage of ADC, 2003, Renesas Technology Corp. (Application Note ref. no: RJS06B0026-0100 http://renesas.com,)
6. DLR1414 4-character 5 × 7 Dot Matrix Alphanumeric Intelligent Display with Memory/Decode/Driver, Infineon Technologies.

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Sep.10.04	—	First edition issued