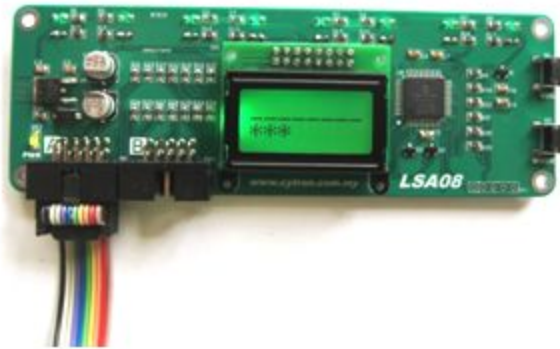




# **Advance Line Following Sensor Bar LSA08**



## **C Library Manual**

**V1.1**

**Jan 2013**

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Cytron Technologies Incorporated with respect to the accuracy or use of such information or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Cytron Technologies's products as critical components in life support systems is not authorized except with express written approval by Cytron Technologies. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

# Index

<a href="#"><u>1.</u></a>	<a href="#"><u>Introduction and Overview</u></a>	<a href="#"><u>3</u></a>
<a href="#"><u>2.</u></a>	<a href="#"><u>Getting Started</u></a>	<a href="#"><u>4</u></a>
	<a href="#"><u>2.1</u></a> <a href="#"><u>UART Mode</u></a>	<a href="#"><u>4</u></a>
	<a href="#"><u>2.2</u></a> <a href="#"><u>Analog Mode</u></a>	<a href="#"><u>7</u></a>
<a href="#"><u>3.</u></a>	<a href="#"><u>Library Functions</u></a>	<a href="#"><u>10</u></a>
<a href="#"><u>4.</u></a>	<a href="#"><u>DISCLAIMER</u></a>	<a href="#"><u>18</u></a>

## 1. INTRODUCTION AND OVERVIEW

LSA08 C library is provided for the convenience of LSA08's user. This library consists of a single file, LSA08.h which user can use, distribute and modify freely. The library includes functions for mainly two types of LSA08's output which are the UART output and Analog. However, for example of using LSA08's digital port (PORT B), user can refer to example code of IFC system. The sample code can be downloaded from Cytron's official website ([www.cytron.com.my](http://www.cytron.com.my)). The library functions are compiled based on some important definitions of user. These definitions will be discussed in "Getting Started" section. All the functions available for the use of the user are listed in the table below. Usage Mode shows that in which LSA08 mode the function can be used. The description and sample code for each function will be discussed in "Library Functions" section.

	Functions	Usage Mode (UART/ANALOG/DIGITAL)
1.	<a href="#">char LSA08_Init(void)</a>	UART, Analog
2.	<a href="#">signed char LSA08_GetPosition(void)</a>	UART, Analog
3.	<a href="#">unsigned char LSA08_GetSensor(void)</a>	UART
4.	<a href="#">unsigned char LSA08_GetJunction(void)</a>	UART
5.	<a href="#">char LSA08_ClearJunction(void)</a>	UART
6.	<a href="#">char LSA08_Calibrate(void)</a>	UART
7.	<a href="#">char LSA08_UARTMode(unsigned char mode)</a>	UART
8.	<a href="#">char LSA08_JunctionWidth(unsigned char jw)</a>	UART
9.	<a href="#">char LSA08_Threshold(unsigned char threshold)</a>	UART
10.	<a href="#">char LSA08_LineMode(unsigned char mode)</a>	UART
11.	<a href="#">char LSA08_Contrast(unsigned char contrast)</a>	UART
12.	<a href="#">char LSA08_BL(unsigned char BL)</a>	UART

## 2. GETTING STARTED

User needs to define the following parameter for the use of LSA08 library **before** the `#include "LSA08.h"`. The definitions for UART output mode and Analog output mode are different as listed below.

### 2. 1 UART Mode

User needs to define several variables at lines before the `#include "LSA08.h"` as shown below.

```
//Choose the mode used-----
#define LSA08_USE_UART

#define _XTAL_FREQ 20000000          //definition of oscillator crystal frequency
#define BAUDRATE 57600              //UART Baudrate
#define LSA08_UE PORTCbits.RC5      //any digital output pin for UE pin of LSA08
#define LSA08_UE_TRIS TRISCbits.TRISC5 //corresponding direction tristate .

//define the LSA08 address-----
unsigned char LSA08_ADD= 0x01;

//define the UART variable-----
unsigned char ERR_FLAG=0;

//Choose the UART Mode used-----
//#define LSA08_UARTMODE 0
//#define LSA08_UARTMODE 1
#define LSA08_UARTMODE 2

#include <pic.h>
#include "LSA08.h"

void main (void)
{

//user main program here....-----

while(1)
```

```
{  
  
    }//while  
}//main
```

User needs to **create** the following **functions** for the use of LSA08 library.

```
void UART_INIT(void);  
void UART_SEND(char data);  
unsigned char UART_REC(void);  
void UART_DUMP(void);
```

Example code for PIC16F887 or any compatibles.

```
void UART_INIT(void)  
{  
    //set port direction  
    TRISCbits.TRISC7=1;  //RX  
    TRISCbits.TRISC6=0;  //TX  
  
#ifdef LSA08_UE_TRIS  
    LSA08_UE_TRIS=0;  
#endif  
    // Initialize UART.  
    TXSTAbits.BRGH = 1;      // Select high speed baud rate.  
    BAUDCTLbits.BRG16=1;     // Baud 16bits  
  
    SPBRG =0x56; SPBRGH=0x00; //57600  
    RCSTAbits.SPEN = 1;      // Enable serial port.  
    TXSTAbits.TXEN = 1;  
    RCSTAbits.CREN = 1;  
}
```

```

void UART_SEND(char data)
{
    while(!TRMT) ; //wait for previous transmit completion
    TXREG=data;
}

unsigned char UART_REC(void)
{
    unsigned long waitcount=0;
    unsigned char rec_data;

    if(RCSTAbits.OERR){
        RCSTAbits.CREN=0;
        RCSTAbits.CREN=1;
        ERR_FLAG=1;
        return(255);
    }
    // Read the received data.
    while (RCIF == 0) //wait for data
    {
        waitcount++;
        if (waitcount > 15000){ //      break if wait too long, no incoming data
            ERR_FLAG=1;
            return (255); //no line
        }
    }

    rec_data = RCREG;

    if (FERR == 1) {
        while(RCIF) rec_data=RCREG;
        ERR_FLAG=1;
        return (255);
    }
    else{
        ERR_FLAG=0;
        return rec_data;
    }
}
//return the data received

```

```

}

void UART_DUMP(void)
{
    unsigned char dump;

    while (RCIF == 1) //wait for data
    {
        dump=UART_REC();
    }
}

```

## 2.2 Analog Mode

User needs to define the following parameters for the use of LSA08 library at lines **before** the `#include "LSA08.h"`.

```

//Choose the mode used-----
#define LSA08_USE_ANALOG

//define the LSA08 address-----
#define LSA08_ADD 0x01

//define analog middle value or ADC value when voltage is 2.25V
#define ANALOG_MIDVAL 115 //please adjust this value accordingly (8 bits value)

#include <pic.h>
#include "LSA08.h"

void main (void)
{

    //user main program here....-----
}

```

```

while(1)
{

} //while
} //main

```

User needs to create the following functions for the use of LSA08 library.

```

void ADC_INIT(void);
unsigned short GET_ADC(void);

```

Example code for PIC16F877/PIC16F887 or any compatibles

```

void ADC_INIT(void)
{
    ADCON0bits.ADCS=0b10;
    ADCON0bits.CHS=0;

    //ANSELbits.ANS0=1;           //PIC16F887 has ANSEL for analog setting

    ADCON1bits.ADFM=1;           //right justified
    ADCON1bits.ADCS2=1;
    ADCON1bits.PCFG=0b1110;      //set channel AN0 analog
    TRISA0=1;                    //set TRISA0 input

    ADCON0bits.ADON=1;

    __delay_ms(5);
}

unsigned short GET_ADC(void)
{
    unsigned short adc;
    __delay_ms(1);               //wait for cap charging

    ADCON0bits.GO_DONE = 1;      // Start the conversion and wait for it to complete.
    while (ADCON0bits.GO_DONE == 1);
}

```



```
// Return the ADC result.  
adc=ADRESH<<8;  
adc|=ADRESL;  
return (adc);  
}
```

### 3. LIBRARY FUNCTION

Every LSA08's library function will be described in the following section and example to use the function will be included for reference.

#### 3.1 char LSA08\_Init(void)

##### Function:

User calls this function to initialize the main controller to use LSA08. This function will initialize main controller to use LSA08 in different modes depending on the defined parameter by the user either **LSA08\_USE\_UART** or **LSA08\_USE\_ANALOG**.

if defined **LSA08\_USE\_UART**, function will

1. Initialize the main controller's UART module
2. Set the UART mode of LSA08 to either mode 1, 2 or 3 according to defined LSA08\_UARTMODE
3. Clear the junction counter of LSA08 through UART command.

if defined **LSA08\_USE\_ANALOG**, function will

1. Initialize the main controller's ADC module

##### Requirements:

Due to varieties of controller that may be used by the user. There is no standard predefined UART initialization function, UART data retrieval and sending function, ADC initialization function and ADC data retrieval function defined in the library. Thus, user needs to define the function for the use of the library.

If defined **LSA08\_USE\_UART**, the following function needs to be defined in main.c or user's main program file.

1. void **UART\_INIT**(void);
2. void **UART\_SEND**(char data);
3. unsigned char **UART\_REC**(void);
4. void **UART\_DUMP**(void);

Example code for the above functions are listed as below. Function may vary dependent on the main controller used, especially for the controller peripheral initialization. Please refer to respective controller's datasheet for correct and complete initialization.

If defined **LSA08\_USE\_ANALOG**, the following function needs to be defined in main.c or

user's main program file.

1. void **ADC\_INIT**(void){;
2. unsigned short **GET\_ADC**(void);

Example code for the above mentioned functions are listed as in the previous section .Function may vary depending on the main controller used, especially for the controller peripheral initialization. Please refer to respective controller's datasheet for correct and complete initialization.

**Return:** initialization status (1-succesful 0- fail)

#### Sample Code:

```
if( LSA08_Init()==1)
    lcd_putstr("init done!");
Else
    Lcd_putstr("init failed");
```

### 3.2 signed char LSA08\_GetPosition(void)

#### Function:

To get the line position detected by LSA08. The return line position depends on the mode defined. This function is only for the use of **UART mode 2** and **Analog Mode**.

#### Requirements:

if defined **LSA08\_USE\_UART**, the following function is needed.

1. void **UART\_SEND**(char data);
2. unsigned char **UART\_REC**(void);

Define LSA08\_UE and initialize that pin and output pin.

**#define LSA08\_UE RC5**

if defined **LSA08\_USE\_ANALOG**, the following function is needed.

1. unsigned short **GET\_ADC**(void);

Define ANALOG\_MIDVAL in the user main program file or main.c. Typically this value is 115  
**#define ANALOG\_MIDVAL 115**

**Return:**

if defined **LSA08\_USE\_UART** and **LSA08\_UARTMODE=2**  
1. Return position of the line detected, value **-35 to 35**, middle as 0  
2. Return 127 if no line detected

if defined **LSA08\_USE\_ANALOG**  
1. Return an 8 bits values dependent on defined ANALOG\_MIDVAL (-values to +values), typically between **-115 to 115**  
2. Return 127 if no line detected

**Sample Code:**

```
signed char Position;  
  
Position =LSA08_GetPosition();
```

### **3.3 unsigned char LSA08\_GetSensor(void)**

**Function:**

To get sensors digital values, for **LSA08\_UARTMODE=1** only. This function sends commands to LSA08 to get the digital values of the sensors.

**Requirements:**

LSA08 in UART mode 1.

**Return:**

Return all 8 sensors digital values. The sensor values are digitized according to the THRESHOLD values of the setting of LSA08. MSB represent sensor 7 and LSB represent sensor 0.

**Sample Code:**

```
unsigned char sensor_bin;  
  
sensor_bin= LSA08_GetSensor();
```

### 3.4 unsigned char LSA08\_GetJunction(void)

**Function:**

To get junction count of LSA08 using LSA08's packet command. The internal junction counter will reset to 0 after the value reach 255.

**Requirements:**

Perform a junction counter clear on starting using **LSA08\_ClearJunction()** function.

**Return:**

Number of junctions detected so far

**Sample Code:**

```
unsigned char junction_count;  
  
junction_count= LSA08_GetJunction();  
  
if(junction_count>5)  
{  
    stop();  
    while(1);  
}
```

### 3.5 char LSA08\_ClearJunction(void)

**Function:**

To clear the internal junction counter of LSA08

**Requirements:**

**Return:**

status (1-succesful, 0-fail)

**Sample Code:**

```
LSA08_ClearJunction();
```

### 3.6 char LSA08\_Calibrate(void)

**Function:**

Call LSA08 to calibrate the sensor to the line and background

**Requirements:**

User need to move the LSA08 across the line and background surface that it will do line following at a constant distance from the surface to allow LSA08's sensor to calibrate and save the value to non-volatile memory.

**Return:**

Status (1-succesful, 0-fail)

**Sample Code:**

```
LSA08_Calibrate();

motor(180,180); //set speed
RotateRight();
__delay_ms(600);

RotateLeft();
__delay_ms(600);__delay_ms(600);

RotateRight(); motor(180,180);
__delay_ms(600);

motor(0,0);Forward();
```

### 3.7 char LSA08\_UARTMode(unsigned char mode)

**Function:**

To set the UART mode of LSA08.

**Input:**

UART mode (0-3)

**mode 0:** no data from UART of LSA08

**mode 1:** one byte digital values of LSA08 sensors

**mode 2:** Line position detected by LSA08

**mode 3:** raw analog data of every sensor

**Requirements:**

**Return:** Status (1-successful, 0-fail)

**Sample Code:**

```
LSA08_UARTMode(2);
```

### 3.8 char LSA08\_JunctionWidth(unsigned char jw)

**Function:**

To set the junction width i.e no of sensors detecting line which LSA08 consider as a junction cross. Set value to a value which is 1 or 2 sensors more than the normal line width.

**Input:**

Junction Width/No of sensors (1-8)

**Requirements:**

**Return:** Status (1-successful, 0-fail)

**Sample Code:**

```
LSA08_JunctionWidth(6);
```

### 3.9 char LSA08\_Threshold(unsigned char threshold)

**Function:**

To set the threshold value for the digitization of sensor value and also number of bars displayed on LSA08's LCD that LSA08 will assume as a line exist. Set appropriately high threshold level for a stable operation of LSA08 and to prevent noises.

**Input:** THRESHOLD value (0-7)

**Requirements:**

**Return:** Status (1-succesful, 0-fail)

**Sample Code:**

```
LSA08_Threshold(4);
```

### 3.10 char LSA08\_LineMode(unsigned char mode)

**Function:**

To set line mode of LSA08 to either Dark-On or Light-On mode. Dark-On means line following is performed on brighter background surface with darker color line. Light-On means line following is performed on darker background compared to the color of the line.

**Input:** DARK\_ON or LIGHT\_ON

**Return:** Status (1-succesful, 0-fail)

**Sample Code:**

```
LSA08_LineMode(DARK_ON);
```



### 3.11 char LSA08\_Contrast(unsigned char contrast)

**Function:**

To set LSA08's LCD contrast value

**Requirements:**

**Input:** contrast level (0-255)

**Return:** Status (1-succesful, 0-fail)

```
LSA08_Contrast(80);
```

### 3.12 char LSA08\_BL(unsigned char BL)

**Function:**

To set the Backlight brightness level of LCD

**Input:** Backlight level (0-10)

**Requirements:**

**Return:** Status (1-succesful, 0-fail)

**Sample Code:**

```
LSA08_BL(2);
```

#### **4. DISCLAIMER**

This SOFTWARE is provided by The provider."as is" and "with all faults." THE PROVIDER makes no representations or warranties of any kind concerning the safety, suitability, inaccuracies, typographical errors, or other harmful components of this SOFTWARE. CYTRON TECHNOLOGIES SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.