|  |
| --- |
| UKMARSBOT I2C Sensor Controller Library Documentation |

Table of Contents

[1.0 LIBRARY OVERVIEW 1](#_Toc68369886)

[2.0 INSTALLATION 1](#_Toc68369887)

[3.0 FUNCTION DESCRIPTIONS 2](#_Toc68369888)

[4.0 EXAMPLES 16](#_Toc68369889)

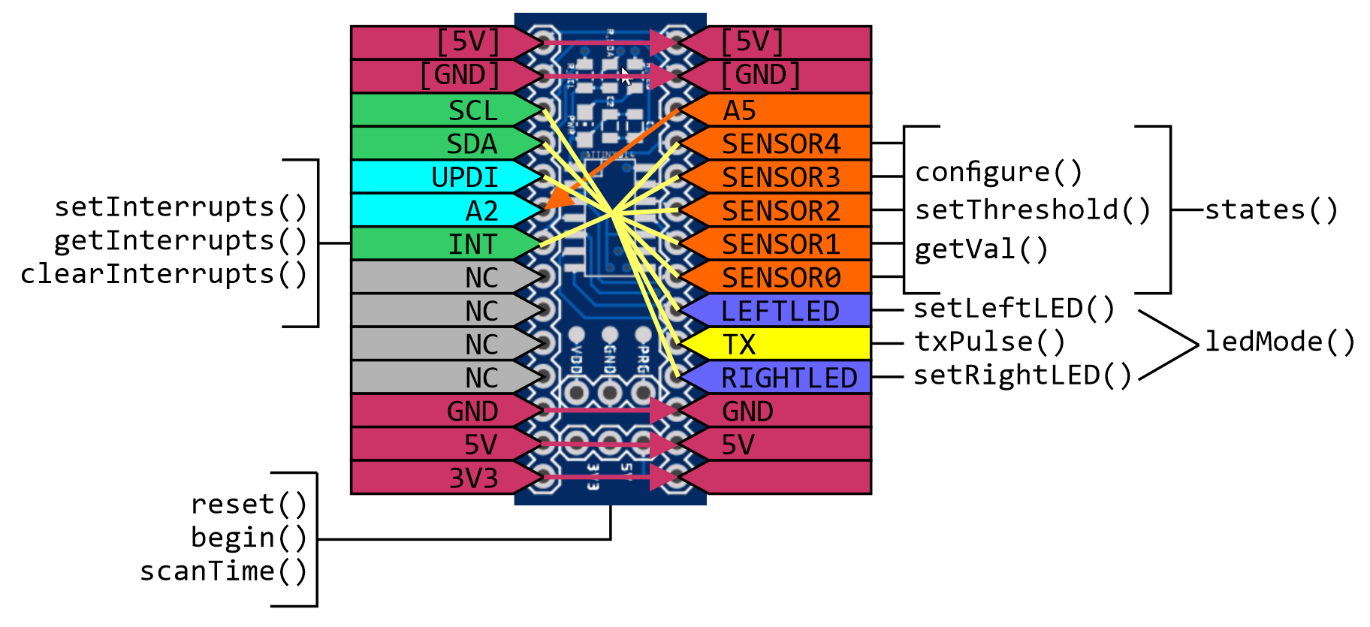
[5.0 Application Note – Custom Sensor Board Configuration: 19](#_Toc68369890)

The ISC has a supporting Arduino library to simplify the integration. This document provides an overview of the Arduino ISC library, describing the functions, their parameters and return values along with examples. Finally, example sketches are provided to give practical examples of how the device may be used.

It is recommended, for better understanding of the capabilities of the device, that the ISC Hardware Datasheet be read in conjunction with this document.

# LIBRARY OVERVIEW

The below diagram indicates schematically how the functions within the Arduino ISC Library may be used in conjunction with an ISC board.

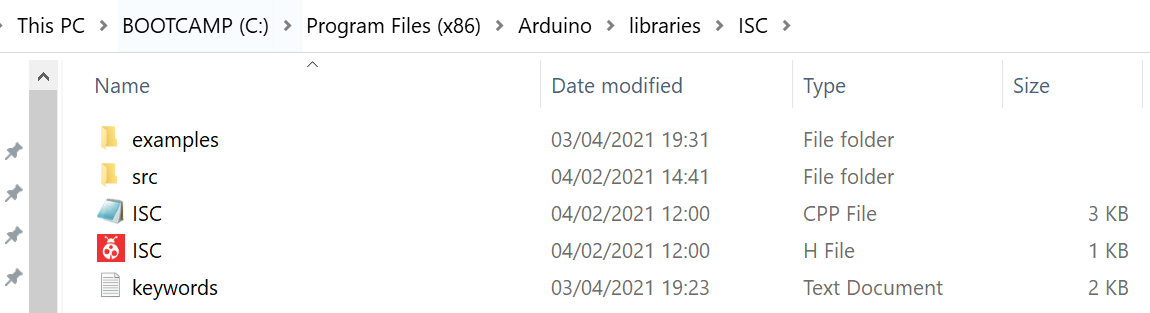


# INSTALLATION

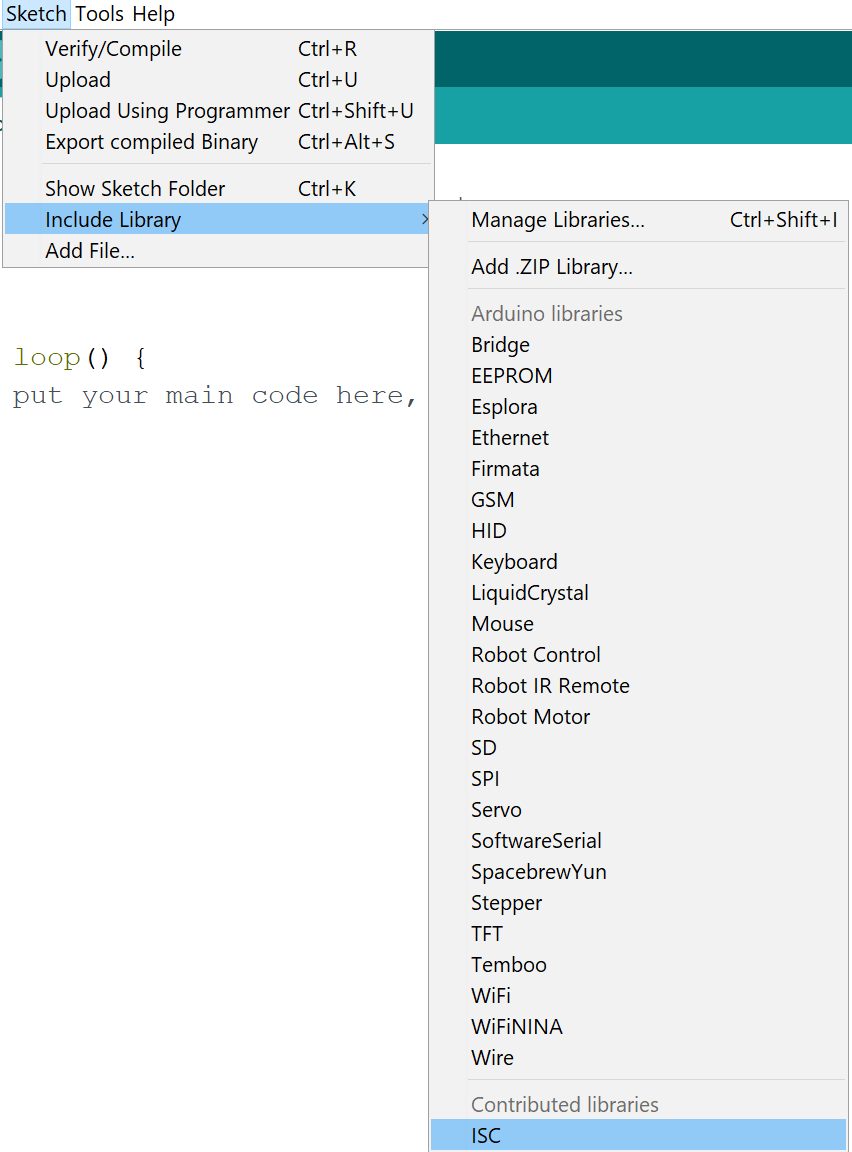
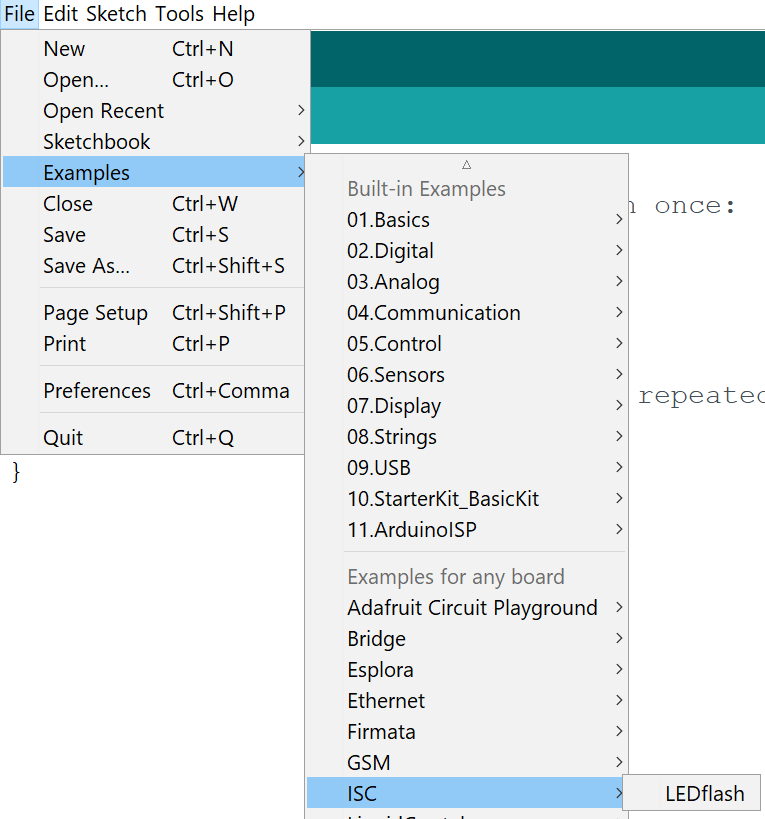
1. The ISC library and all other supporting files are provided at the following link:

<https://github.com/slinkyfish/ISC-Project>

1. Download the project as a compressed file.
2. Extract the downloaded file and copy the ISC folder into the libraries folder within the Arduino installation. The directory structure should look like the below image.



1. Once the Arduino software is restarted, the library will be available to include, and example code detailed in Section 4.0 will be available in the Examples menu.

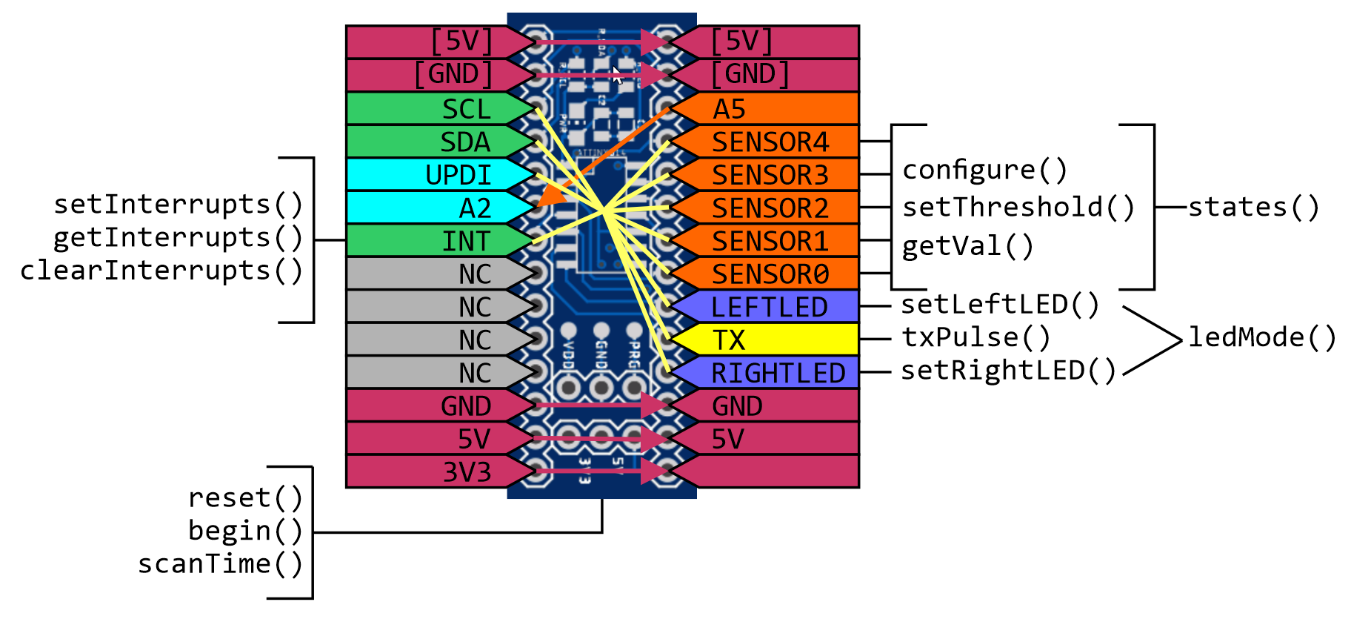
 

1. To use this library in any sketch, include the following at the start of the code:

#include <**ISC**.h>

# FUNCTION DESCRIPTIONS

The functions included in the Arduino ISC Library are shown schematically in the graphic below, then listed with brief description and link to detailed description after.



Global

[reset()](#_reset()) //Forces the ISC to a reset state, starting initial setup

[begin()](#_begin()) //Starts the ISC in run mode, defines sensor board

[scanTime()](#_scanTime()) //Returns the time taken for the ISC to complete a full cycle

LED Indicators

[ledMode()](#_ledMode()) //Controls the Indicator LED function

[setLeftLED()](#_setLeftLED()) //Direct control of Left LED state

[setRightLED()](#_setRightLED()) //Direct control of Right LED state

Individual sensors

[configure()](#_configure()) //Individual sensor setup configuration

[setThreshold()](#_setThreshold()) //Specifies the value, above which, the state is ON

[getVal()](#_getVal()) //Returns the value of specified sensor

Overview of sensors

[states()](#_states()) //Returns a byte, with bits representing sensor states

[txPulse()](#_txPulse()) //Sets time that Sensor Tx. is on before first sample is taken

Interrupts

[setInterrupts()](#_setInterrupts()) //Set components that can generate an interrupt

[getInterrupts()](#_getInterrupts()) //Determine which component(s) caused the interrupt

[clearInterrupts()](#_clearInterrupts()) //Acknowledges interrupts, clearing them down for next time

## reset()

Description

Forces the ISC to a reset state, allowing the initial setup to be performed.

Syntax

void reset(uint8\_t address);

Parameters

address 7-bit I2C address for ISC device – printed on device.

Returns

N/A

Example

|  |
| --- |
| Isc.reset(0x50); //Reset ISC with I2C (7-bit) Address of 0x50 |

## begin()

Description

This function starts the ISC in run mode, confirming that setup is complete. It defines the sensor board that is attached.

Syntax

void begin(uint8\_t boardType);

Parameters

boardType Type of sensor board connected to the ISC:

basicLineSensor UKMARS Basic Line Sensor Board

basicWallSensor UKMARS Basic Wall follower Board

spLineSensor Line Sensor board by S. Pithouse

customBoard Undefined board

Returns

N/A

Example

|  |
| --- |
| Isc.begin(basicLineSensor); //Start with Basic Line sensor attached |

## scanTime()

Description

This function returns the most recent time taken for the ISC to complete a full cycle. The units are µs.

Syntax

int scanTime();

Parameters

N/A

Returns

Most recent time taken for the ISC to complete a full cycle (µs).

Example

|  |
| --- |
| int myTime = Isc.scanTime(); //Store most recent scan time in myTime |

## setThreshold()

Description

This function allows the sensor threshold to be set for a specific sensor.

Syntax

void setThreshold(uint8\_t sens, uint16\_t threshVal);

Parameters

Sensor

SENSOR0 See diagram in Section ?? – A0 on Sensor Board

SENSOR1 See diagram in Section ?? – A0 on Sensor Board

SENSOR2 See diagram in Section ?? – A0 on Sensor Board

SENSOR3 See diagram in Section ?? – A0 on Sensor Board

SENSOR4 See diagram in Section ?? – A0 on Sensor Board

threshVal

Value between 0 – 1024

Returns

N/A

Example

|  |
| --- |
| Isc.setThreshold(SENSOR0, 500); |

## getVal()

Description

This function returns the most recent sensor value for the specified sensor.

Syntax

int getVal(uint8\_t sensor);

Parameters

Sensor

SENSOR0 See diagram in Section ?? – ‘A0’ on Sensor Board

SENSOR1 See diagram in Section ?? – ‘A1’ on Sensor Board

SENSOR2 See diagram in Section ?? – ‘A2’ on Sensor Board

SENSOR3 See diagram in Section ?? – ‘A3’ on Sensor Board

SENSOR4 See diagram in Section ?? – ‘A4’ on Sensor Board

Returns

Value of specified sensor

Example

|  |
| --- |
| leftSensor = Isc.getVal(SENSOR0); |

## ledMode()

Description

This function controls the Indicator LED function

Syntax

void ledMode(uint8\_t mode);

Parameters

mode

MASTERCTRL MCU has control

FREQOUT Scan frequency output on LEDs

LEDOFF Turn LEDs off

LEDBRDCTRL Allow board specific LED control

FASTBLINK Start LEDS blinking fast

SLOWBLINK Start LEDs blinking slowly

Returns

N/A

Example

|  |
| --- |
| Isc.ledMode(FASTBLINK); |

## setLeftLED()

Description

This function controls the Indicator LED function. Reads current led state, sets into Master control mode and turns Left LED on.

Syntax

void setLeftLED(uint8\_t state);

Parameters

state

HIGH Turn LED on

LOW Turn LED off

Returns

N/A

Example

|  |
| --- |
| Isc.setLeftLED(HIGH); |

## setRightLED()

Description

This function controls the Indicator LED function. Reads current led state, sets into Master control mode and turns Left LED on.

Syntax

void setRightLED(uint8\_t state);

Parameters

state

HIGH Turn LED on

LOW Turn LED off

Returns

N/A

Example

|  |
| --- |
| Isc.setRightLED(HIGH); |

## configure()

Description

This function allows individual sensor setup to be configured. There are keywords that may be combined with ‘+’ as shown.

Syntax

Isc.configure(uint8\_t sensor, uint8\_t config);

Parameters

sensor

SENSOR0 See diagram in Section ?? – ‘A0’ on Sensor Board

SENSOR1 See diagram in Section ?? – ‘A1’ on Sensor Board

SENSOR2 See diagram in Section ?? – ‘A2’ on Sensor Board

SENSOR3 See diagram in Section ?? – ‘A3’ on Sensor Board

SENSOR4 See diagram in Section ?? – ‘A4’ on Sensor Board

config

enb Enable sensor – ISC will read the value

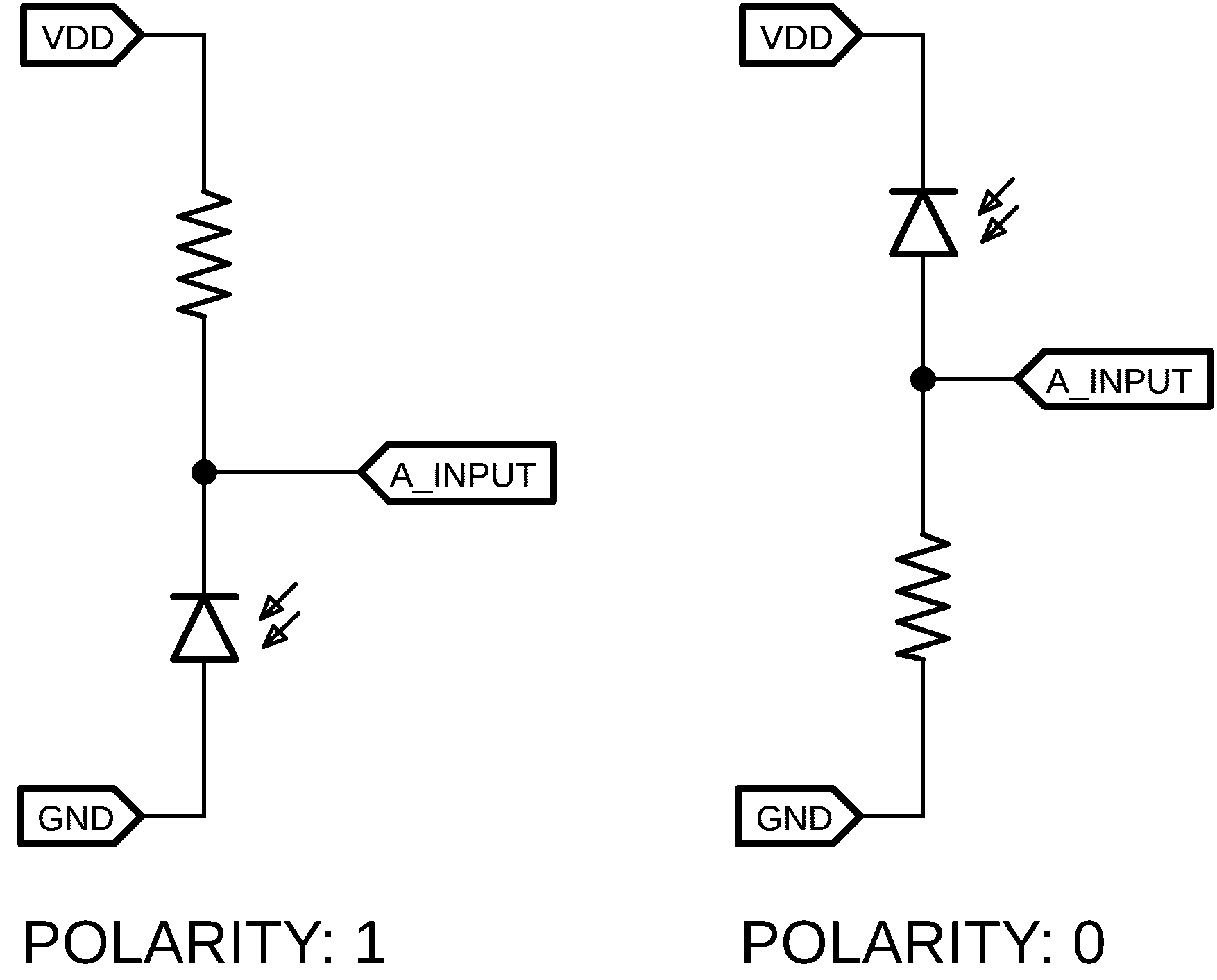
lowRes Reduce value to 8-bit

txEnb Enable Transmitter to help remove ambient light

fallingInterrupt Sensor will cause interrupt on 1->0 state change

risingInterrupt Sensor will cause interrupt on 0->1 state change

flip [See Polarity]



Returns

N/A

Example

|  |
| --- |
| Isc.configure(SENSOR0, enb + txEnb + risingInterrupt); //Marker Sensor |

## txPulse()

Description

This function allows the time in µs that the Tx emitter is on before the first sample is taken to be adjusted.

Syntax

void txPulse(uint8\_t length);

Parameters

Length

Amount of time in us…??

Returns

N/A

Example

|  |
| --- |
| Isc.txPulse(100); |

## states()

Description

This function gets the current states of the sensors and returns a byte with 1 representing HIGH sensor state.

Syntax

uint8\_t states();

Parameters

N/A

Returns

Byte with bits indicating current sensor state

Example

|  |
| --- |
| leftState = Isc.states() & 0b1; |

## read()

Description

Reads a byte (or pair of bytes) from a specified register within the ISC.

Syntax

int read(uint8\_t regAddr, uint16\_t numVals);

Parameters

regAddr

Register address to read from

numVals

May be 1 or 2, for the number of consecutive registers to read (ie. 2 when a 2-byte

value is to be read).

Returns

The value stored in that location (int)

Example

|  |
| --- |
| int timeTaken = Isc.read(SCANTIME, 1);//Read 1 byte at address SCANTIME |

## write()

Description

Writes a byte (or pair of bytes) to a specified register within the ISC.

Syntax

void write(uint8\_t regAddr, uint16\_t data, uint8\_t numVals);

Parameters

regAddr

Register address to write to

data

Data to write

numVals

May be 1 or 2, for the number of consecutive registers to write (ie. 2 when a 2-byte

value is to be written).

Returns

N/A

Example

|  |
| --- |
| Isc.write(SENS0THRSH, 500, 2);//Write 500 (2 bytes) for Sensor 0 Threshold |

## setInterrupts()

Description

Set components that can generate an interrupt.

Syntax

void setInterrupts(uint8\_t interrupts);

Parameters

interrupts

SENSOR0INTERRUPT Interrupt according to Sensor 0 Configuration

SENSOR1INTERRUPT Interrupt according to Sensor 1 Configuration

SENSOR2INTERRUPT Interrupt according to Sensor 2 Configuration SENSOR3INTERRUPT Interrupt according to Sensor 3 Configuration

SENSOR4INTERRUPT Interrupt according to Sensor 4 Configuration

Returns

N/A

Example

|  |
| --- |
| setInterrupts(SENSOR0INTERRUPT + SENSOR1INTERRUPT); //Interrupt on Sensor 0 and Sensor 1 as per their configuration (rising/falling) |

## getInterrupts()

Description

Reads a byte (or pair of bytes) from a specified register within the ISC.

Syntax

uint8\_t getInterrupts();

Parameters

N/A

Returns

Active Interrupts which may be interrogated

Example

|  |
| --- |
| int activeInterrupts = getInterrupts(); //Store components which triggered interrupt |

## clearInterrupts()

Description

Acknowledges interrupts, clearing them down for next time.

Syntax

void clearInterrupts(uint8\_t interrupts);

Parameters

interrupts

SENSOR0INTERRUPT Interrupt according to Sensor 0 Configuration

SENSOR1INTERRUPT Interrupt according to Sensor 1 Configuration

SENSOR2INTERRUPT Interrupt according to Sensor 2 Configuration SENSOR3INTERRUPT Interrupt according to Sensor 3 Configuration

SENSOR4INTERRUPT Interrupt according to Sensor 4 Configuration

Returns

N/A

Example

|  |
| --- |
| clearInterrupts(SENSOR0INTERRUPT + SENSOR1INTERRUPT); //Clear down Sensor 0 and Sensor 1 Interrupt flags so they can trigger again |

# EXAMPLES

## Example\_0\_Blink

Initialise the ISC and set LED mode to SLOWBLINK.

#include <ISC.h>

ISC Isc;

void setup(){

Isc.reset(0x50); //Establish communication and Reset ISC (Address 0x50)

Isc.begin(basicLineSensor);

ledMode(SLOWBLINK);

}

void loop(){

}

## Example\_1\_LeftLED

Initialise the ISC and set Left LED on.

#include <ISC.h>

ISC Isc;

void setup(){

Isc.reset(0x50); //Establish communication and Reset ISC (Address 0x50)

Isc.begin(basicLineSensor);

leftLED(HIGH);

}

void loop(){

}

## Example\_2\_ReadSensor

Initialise the ISC and read sensor 0 value every 100ms.

#include <ISC.h>

ISC Isc;

void setup(){

Serial.begin(115200); //Start communications over Serial Port

Isc.reset(0x50); //Establish communication and Reset ISC (Address 0x50)

Isc.setup(SENSOR0, enb+txEnb); //Setup Left Sensor, enabled with Tx enabled

Isc.txPulse(10); //10µs Tx on time before sampling begins

Isc.begin(basicLineSensor);

}

void loop(){

Serial.println(Isc.getVal(SENSOR0));

delay(100);

}

## Example\_3\_ScanTime

Initialise the ISC and report the scan time every 100ms

#include <ISC.h>

ISC Isc;

void setup(){

Serial.begin(115200); //Start communications over Serial Port

Isc.reset(0x50); //Establish communication and Reset ISC (Address 0x50)

Isc.begin(basicLineSensor);

}

void loop(){

Serial.println(Isc.scanTime(SENSOR0));

delay(100);

}

## Example\_4\_Interrupts

Configure a sensor increment a count each time it exceeds a boundary

# TEXT – KEYWORDS

Isc.configure(SENSOR0, enb + txEnb + risingInterrupt); //Configure Marker Sensor

# FUNCTIONS

**editor.keyword1.style = #d35400,bold**

editor.data\_type.style = #d35400,bold

# METHODS

editor.keyword2.style = #D35400,plain

editor.function.style = #d35400,plain

# STRUCTURES

editor.keyword3.style = #5E6D03,plain

editor.reserved\_word.style = #5E6D03,plain

# TEXT - LITERALS

editor.literal1.style = #00979C,plain

editor.literal2.style = #00979C,plain

editor.variable.style = #00979C,plain

editor.reserved\_word\_2.style = #00979C,plain

editor.literal\_boolean.style = #00979C,plain

editor.literal\_char.style = #00979C,plain

editor.literal\_string\_double\_quote.style = #005C5F,plain

editor.preprocessor.style = #5E6D03,plain

# http://www.arduino.cc/

editor.url.style = #0000ff,underlined

# e.g. + - = /

editor.operator.style = #434f54,plain

# ?? maybe this is for words followed by a colon

# like in case statements or goto

**editor.label.style = #7e7e7e,bold**

# TEXT - COMMENTS

editor.comment1.style = #434F54,plain dk grey /\*\*/

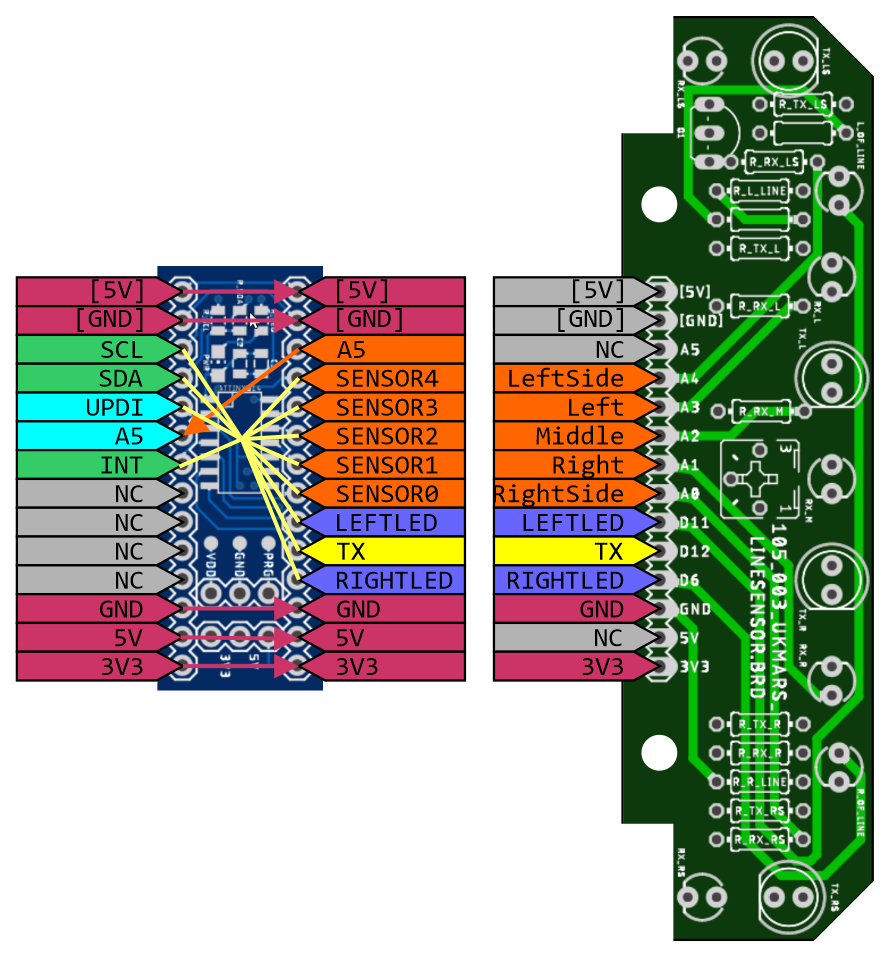
editor.comment2.style = #95a5a6,plain light grey //

# LINE STATUS - editor line number status bar at the bottom of the screen

linestatus.font = SansSerif,plain,10

linestatus.height = 20

# Application Note – Custom Sensor Board Configuration:



Basic Line Sensor Connections

