|  |
| --- |
| UKMARSBOT I2C Sensor Controller Notes |

Table of Contents

[1.0 FIRMWARE REVISION HISTORY 4](#_Toc63937504)

[2.0 ADDING A NEW REGISTER 5](#_Toc63937505)

[3.0 COMMISSIONING NOTES 7](#_Toc63937506)

[4.0 APPLICATION NOTES 8](#_Toc63937507)

**Nomenclature**

Below are definitions of how the following are used in the context of this datasheet:

ISC I2C Sensor Controller

MCU Microcontroller connected to the ‘Arduino Nano’ Style headers on the UKMARSBOT

ToDo:

* Should PULSELENGTH be renamed to pre… something
* Determine which polarity diagram is correct – double check with code and basic line board and my line board
* Check what value capacitors should be on power supply on ISC schematic
* How to distribute Arduino library – can it go on Arduino website? Or if not, then github…

The manual format and layout is based loosely on the following manual:

1. MCP23017 Datasheet from here: [https://ww1.microchip.com/downloads/en/DeviceDoc/20001952C.pdfhttps://ww1.microchip.com/downloads/en/DeviceDoc/20001952C.pdf](https://ww1.microchip.com/downloads/en/DeviceDoc/20001952C.pdfhttps:/ww1.microchip.com/downloads/en/DeviceDoc/20001952C.pdf)

From AVR manual:

Add below to ‘functional description’ or application notes

An interrupt source is enabled or disabled by writing to the corresponding enable bit in the peripheral's Interrupt Control register (peripheral.INTCTRL).

An interrupt request is generated when the corresponding interrupt source is enabled and the Interrupt Flag is set. The interrupt request remains active until the Interrupt Flag is cleared. See the peripheral's INTFLAGS register for details on how to clear Interrupt Flags.

Blog Post

Beginners

Plug and play – no other hardware required – just solder on own headers to suit configuration.

Simplify UKMARSBOT code with library:

Example:

Set LEDs blinking:

#include <isc.h>

Isc isc;

isc.LED(FASTBLINK); //Start Indicator LEDs blinking

myLineError = isc.error(); //Value from -512 to 512 where 0 is centred on line

leftSensor = isc.Read(LEFT\_SENSOR); //Read value from left sensor where LEFT\_SENSOR is defined as constant (0-4)

Advanced

Application with other MCU – RPi Pico flashing LEDs over I2C example

Other I2C device on the same bus

# Application Notes

Initialise ISC:

1. Reset strobe
2. Set registers
3. Enable

Read Single Sensor:

1. Set sensor state rising / falling interrupt
2. Set Tx. Enable:
   1. ON: takes two readings and returns the difference
   2. OFF: returns the raw sensor reading with LED off
3. Set Polarity – only necessary when boardtype = 0
4. Set resolution bit high if only 8 bit result is required
5. Enable interrupts in INTCTRL
6. Enable sensor

# Library Documentation

Function Descriptions

Read

This function reads the value from the given address. The 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).

Function Definition

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

Example:

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

Write

This function writes an 8 bit or 16bit value to the given address. The numVals may be 1 or 2, for the number of consecutive registers the value is to overwrite (ie. 2 when a 2-byte value is to be written).

Function Definition

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

Example:

Isc.write(SENS0THRSH, 500, 2); //Write the value 500 across two bytes for the Sensor 0 Threshold

Reset

This function returns the ISC to the reset state, allowing the setup to be started again

Function Definition

void reset(uint8\_t address);

Example:

Reset(0x50); //Reset ISC with I2C (7-bit) Address 0x50

Begin

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

Function Definition

void begin(uint8\_t boardType);

Example:

Isc.begin(1); //Start run mode, Basic Line sensor board is attached

ScanTime

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

Function Definition

int scanTime();

Example:

int myTime = isc.scanTime(); //Store most recent scan time in myTime

SetThreshold

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

Function Definition

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

Example

isc.setThreshold(SENSOR0, 500);

Sensor

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

Function Definition

int sensor(uint8\_t sensor);

Example

leftSensor = isc.sensor(SENSOR0);

LED Mode

This function controls the Indicator LED function

Function Definition:

void ledMode(uint8\_t mode);

Example:

Isc.ledMode(2);

Setup

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

Function Definition

void setup(uint8\_t sensor, uint8\_t bits);

Keywords

Enb Enable sensor – ISC will read the value

lowRes Reduce value to 8-bit

txEnb Enable Transmitter to help remove ambient light

invert [See Polarity]

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

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

Example:

Isc.setup(SENSOR0, enb+txEnb+fallingInterrupt);

txPulse

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

Function Definition

void txPulse(uint8\_t length);

Example

Isc.txPulse(100);

States

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

Function Definition

uint8\_t states();

Example

leftState = isc.states() & 0b1;

# ADDING A NEW REGISTER

The below points should be considered / implemented when adding a new register or modifying an existing one.

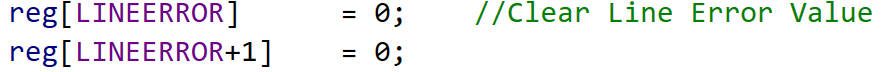
1. Define in addressMap.h



1. Update numRegisters in addressMap.h



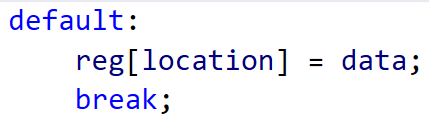
1. Update setRegDefaults() function



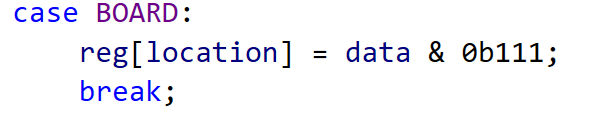
1. Update receiveData() function
   1. Readonly?



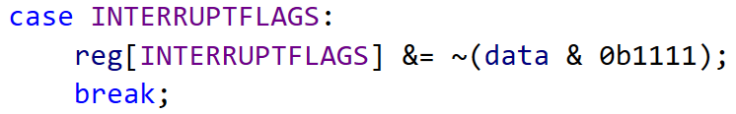
* 1. Write?
     1. If the whole register is to be replaced, ‘default’ case handles this.



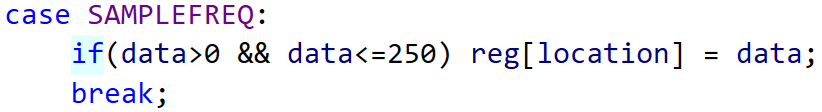
* + 1. If only some elements can be written to:



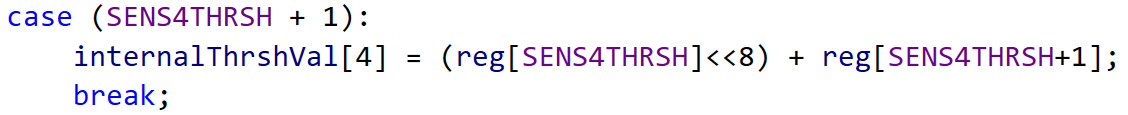
* + 1. If bits are to be cleared (ie. interrupt flags):



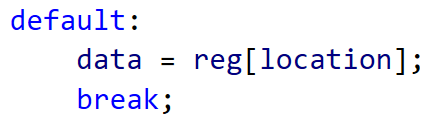
* + 1. If value is constrained:



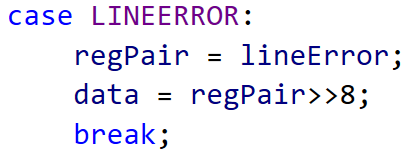
* 1. Second of two-byte pair is written, combine to create internal value.



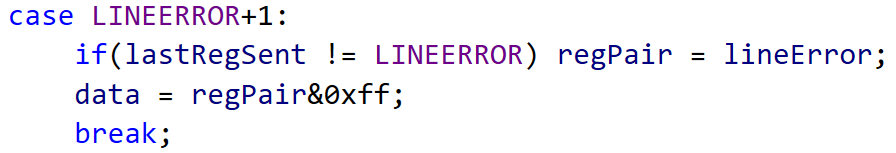
1. Update sendData() function
   1. If a single register is to be sent, ‘default’ case handles this:



* 1. If the first byte of a 2-byte pair is read:
     1. The corresponding internal value (MSB and LSB) is ‘frozen’ in regPair
     2. MSB is extracted to be sent



* 1. If the second byte of a 2-byte pair is read:
     1. If the last register sent was not the first byte of the pair, update the ‘frozen’ value
     2. LSB is extracted to be sent



# COMMISSIONING NOTES

Desired Functionality:

1. Read live sensor values
2. Ability to alter emitter pulse duration
3. Ability to read raw sensor value without emitter turned on
4. Set threshold of White/Black
5. Obtain sensor state (result of comparison on live value and threshold)
   1. ~~Option to invert polarity~~ (is this necessary with ability to choose from below?)
6. Interrupt on sensor state changing (RISING / FALLING / BOTH)
7. Add Register for control / configuration of onboard indicator LEDs:
   1. Indicating side of line
   2. Set over I2C registers
8. ~~Be able to set sensor update frequency with register~~
9. ~~Be able to set I2C address from jumper on board – 3 bits is plenty (use DIP switch).~~
   1. ~~OR is it possible to write to non-volatile memory?~~ Is this necessary?

Software notes:

* Remove ability to invert sensor state in sensor setup – rising and falling interrupts remove the need for this I think
* Make code match manual, noting that registers have changed order, and some bits have been added/removed and moved around
* Remove loop frequency stuff – just loop as fast as possible
* Make timer time cycle and send over I2C – freezing LSB if required
* Make address = 0x50 (7-bit)
* Test TCA period to see what value scanTime becomes, tweak to prevent wraparound

Considerations for next version:

* Silkscreen could indicate pin numbers to reference to from documentation
* Consider using DI as interrupt –frees up analog Input pins
* Remove UPDI from UKMARSBOT interface? – is it easy to program UPDI from Arduino?
* Does Voltage selection header need to be a header, or could it be a solder bridge?
* Fix Silkscreen for V selection header
* Use readable size Silkscreen lettering
* Indicate LED polarity on silkscreen

# APPLICATION NOTES

Basic Line Sensor:

Corner / Radius markers: Rising Interrupt, enabled with Tx

Line Sensors: enabled with Tx

Pulse duration >= 10us