ARDUINO EXAMPLE SOFTWARE GUIDE - IQS7211A

A guide on how to use the Arduino example software functions

Contents

1 Introduction

This document is a supplement to the provided example code and provides additional information regarding the hardware and software required.

The example code is intended for the Arduino UNO revision 3 PCB. The code is free software, and a user can redistribute it or modify it under the terms of the latest GNU General Public License as published by the Free Software Foundation. The source code can easily be ported to a different platform if required, as it consists mostly of I^2C communication with the IQS7211A device.

The example project implements a master I^2C protocol and interacts with the IQS7211A device on this bus. The relevant data is read from the IQS7211A device and displays this on the serial monitor in the Arduino development environment when user interaction occurs.

The comments in the code are very thorough and serve as additional documentation.

Please visit the Arduino website for information related to the Arduino.

2 Hardware Connections

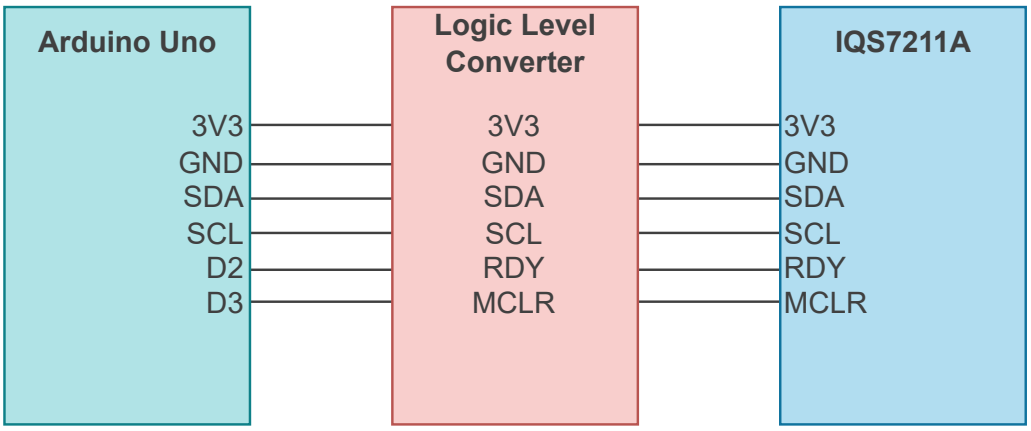

Figure 2.1: Connection Diagram between Arduino and IQS7211A

Pull-up resistors are required on the SDA, SCL and RDY lines (if not present on the IQS7211A application PCB). The recommended value for the pull-up resistors is 4.7kΩ. Digital I/O Pin 2 of the Arduino is used for the RDY input and Pin 3 for the MCLR output. Both are located under the *Digital* group. If Pin 2/3 is not available, the user can use any other pin (to connect to RDY/MCLR) on the Arduino board by changing the definitions in the *.ino* project file. Do not use the 5V rail to power the IQS7211A as the operational voltage range of the IQS7211A is from 1.71V to 3.5V. A logic level converter must be used to resolve issues with using I^2C devices that operate on different voltage levels.

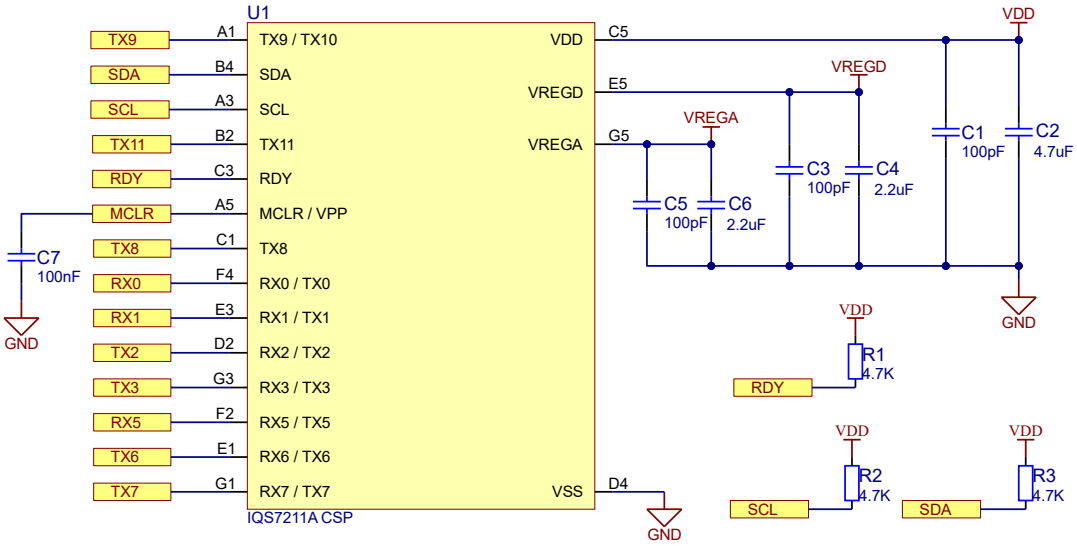

Figure 2.2: IQS7211A Reference Schematic

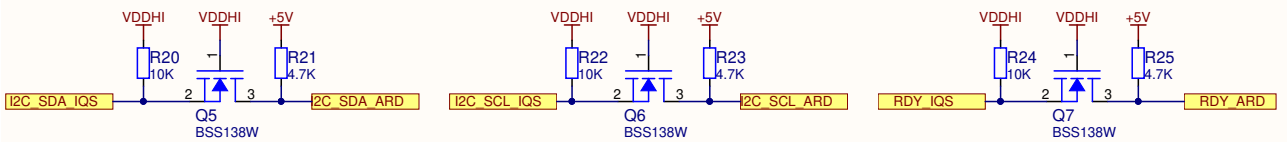

Figure 2.3: Example Circuit of Logic Level Converters

2.1 RDY/IRQ

The communication has an open-drain active-low RDY signal to inform the master that updated data is available. It is optimal for the master to use this as an interrupt input and obtain the data accordingly. It is also useful to allow the master MCU to enter low-power/sleep allowing wake-up from the touch device when user presence is detected.

2.2 Hardware Reset (MCLR)

The MCLR pin (active-LOW) can be used to reset the device.

3 Quick Start Guide

Follow these steps to get started:

1. Connect the Arduino to the application PCB as described in Section 2.
2. Connect the Arduino to a PC via USB
3. Open the example project in Arduino IDE (by opening *'IQS7211A_Example_Code_V1.0.ino'*)
4. Verify/Compile the project/sketch (click the icon or press Ctrl+R), and confirm there are no errors
5. Upload to the device (click the icon or press Ctrl+U)
6. Open the serial monitor (go to Tools->Serial Monitor). It might be necessary to change the port setup to the correct port (go to Tools->Port: *'COMxx'*). In addition, the line ending, and baud rate (115200) must be configured, see the screenshot below for details.
7. If the previous steps are successful, then the serial monitor should display the version information of the IQS7211A from the connected device. Here is an example of the version displayed from a connected IQS7211A PCB:

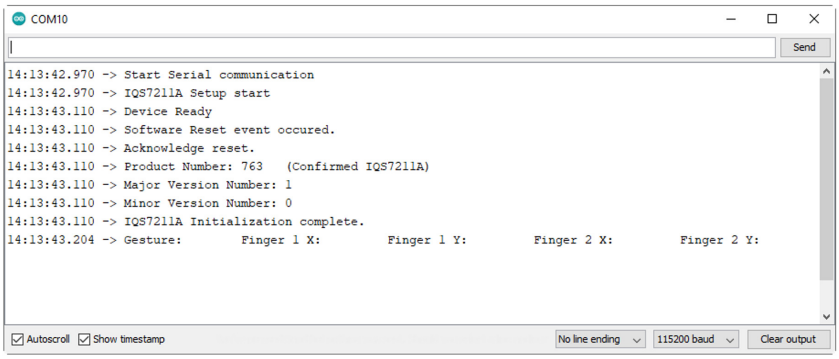

Figure 3.1: Successful Initialisation Serial Output

8. Now when interacting with the trackpad, data will display on the serial monitor as follows:

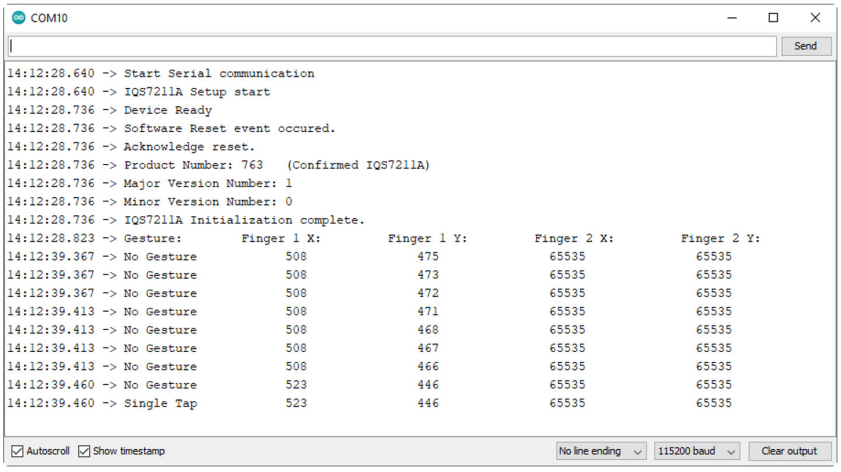

Figure 3.2: Serial Output when Interacting with Trackpad

4 Software Guide

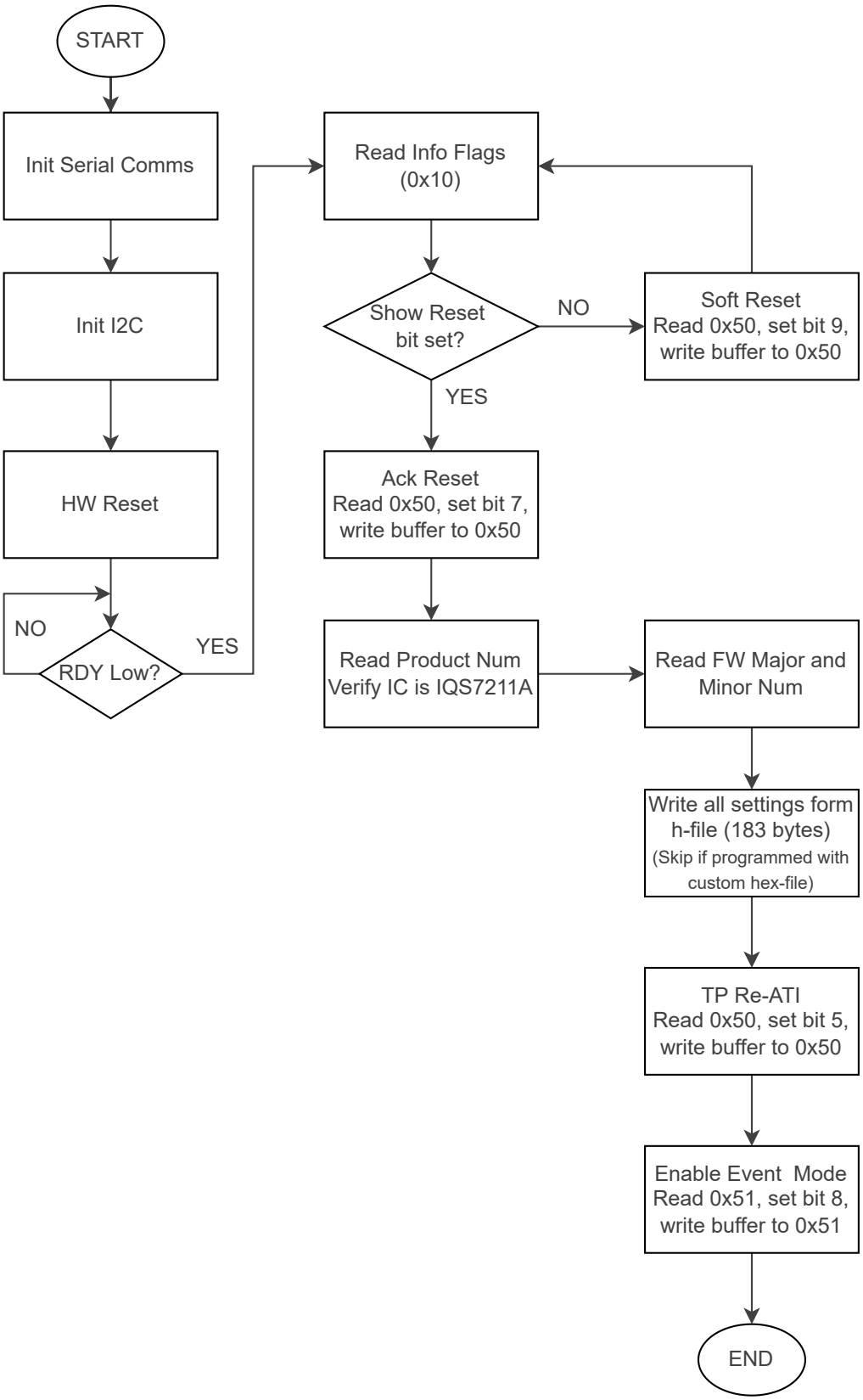

Figure 4.1: Initialisation Software Flow Diagram

4.1 I²C Module Specification

The device supports a standard two wire I²C interface with the addition of an RDY (ready interrupt) line. The provided interrupt line (RDY) is an open-drain active low implementation and indicates a communication window. IQS7211A has a fixed I²C address of 0x56. Most of the memory map implements an 8-bit addressing scheme for the required user data. For all application requirements this should be adequate.

The data is 16-bit words, meaning that each address obtains 2 bytes of data. For example, address 0x50 will provide two bytes, then the next two bytes read will be from address 0x51. The 16-bit data is sent in little endian byte order (least significant byte first).

A standard I²C STOP ends the current communication window. If multiple I²C transactions need to be done, then they should be strung together using repeated-start conditions instead of giving a STOP. This will allow the communication to occur in the same session.

4.2 Initialization

The software flow diagram shows typical steps to follow for initialization. Note that before each write transaction the register is first read. This is done to ensure that you only change the bit(s) you want to and leave the rest like it was. These steps can be followed when writing to a register: read, modify, write. See the logic analyser capture below for an example of the transactions that take place when enabling event mode.

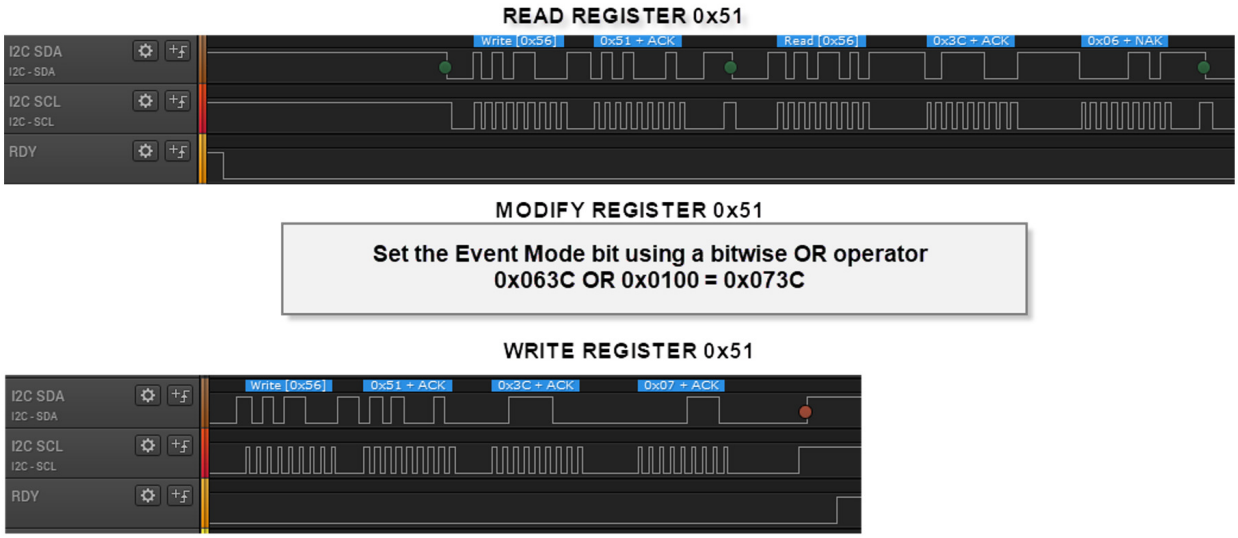

Figure 4.2: Logic Analyser Capture

The IQS7211A is programmed with the generic trackpad firmware. The example code writes all the settings defined in the h-file exported by the IQS7211A PC GUI.

4.2.1 Important Bits

Below are some important bits used during the initialization process of the example code. Detailed information is available in the IQS7211A datasheet.

Info Flags (Register 0x10):

- Bit 7: **Show Reset**- Indicates a reset
  - 0: Reset indication has been cleared by host, writing to *Ack Reset*
  - 1: Reset has occurred and indication has not been cleared by host

System Control (Register 0x50):

- Bit 9: **SW Reset**- Reset the device
  - 0: No action
  - 1: Reset device after communication window terminates
- Bit 7: **Ack Reset**- Acknowledge a reset
  - 0: No action
  - 1: Acknowledge the reset by clearing *Show Reset* flag
- Bit 5: **TP Re-ATI**- Queue a re-ATI on trackpad channels
  - 0: No action
  - 1: Perform re-ATI when trackpad channels are sensed again

Config Settings (Register 0x51):

- Bit 8: **Event Mode**- Enable event mode communication
  - 0: I^2C is presented each cycle (except auto-prox cycles)
  - 1: I^2C is only initiated when an enabled event occurs

4.3 Arduino IQS7211A Driver Functions

The comments in the code describe the functions thoroughly and can be used as an example to expand the IQS7211A driver. Navigate to *'\src\IQS7211A\IQS7211A.cpp'* to see the functions used in the Arduino example. Also see all the available unions already defined in the IQS7211A.h file.

The user can replace the *'IQS7211A_init_AZP1149A1.h'* settings file with their own h-file exported from the IQS7211A GUI. Remember to update the reference in *'IQS7211A.cpp'*.

```
// Path to settings header file
#include "src/IQS7211A/IQS7211A_init_AZP1149A1.h"
```

All the memory map addresses can be found in *'\src\IQS7211A\inc\iqs7211a_addresses.h'*

Contact Information

	USA	Asia	South Africa
Physical Address	11940 Jollyville Rd Suite 120-S Austin TX–78759 USA	Room501A, Block A T-Share International Centre Taoyuan Road Nanshan District Shenzhen Guangdong Province PRC	1 Bergsig Avenue Paarl 7646 South Africa
Postal Address	11940 Jollyville Rd Suite 120-S Austin TX–78759 USA	Room501A, Block A T-Share International Centre Taoyuan Road Nanshan District Shenzhen Guangdong Province PRC	PO Box 3534 Paarl 7620 South Africa
Tel	+1 512 538 1995	+86 755 8303 5294 ext 808	+27 21 863 0033
Email	info@azoteq.com	info@azoteq.com	info@azoteq.com

Visit www.azoteq.com
for a list of distributors and worldwide representation.