



IQS5xx-B000 I²C Bootloader Example Code Description

Version 1.1

24 February 2017

1 Introduction

The IQS5xx-B000 I²C bootloader example code illustrates how to program or upgrade the firmware of an IQS5xx-B000 device.

The example code was developed using an Arduino Uno Rev3 PCB and the standard Arduino IDE. The integrated serial terminal in the Arduino environment was used to display output data to the user.

Porting this example project to a different MCU will require the low-level I²C functions to be updated for that specific MCU, but most of the code is reusable.

The IQS5xx-B000 GUI supplies a user with a firmware file that is specific to their application. The firmware file is stored in Intel Hex Format and must first be parsed before it can be programmed to the IQS5xx IC. For this purpose, we use a parser application to convert the HEX file into an equivalent C-language header file.

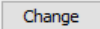
For information on using the GUI to obtain your IQS5xx-B000 custom settings, and to generate the custom HEX file, please refer to the following document available on the Azoteq website under *Design -> Application Notes*:

AZD087 - IQS5xx-B000 Setup and User Guide.pdf

The rest of this document describes the example hex file parser and how to use it, as well as the implementation of the example I²C bootloader code.

2 Hex File Parser

An application was developed to convert an IQS5xx-B000 firmware file, stored in the Intel Hex format, to a C-language header file. The header file can then be used by the firmware of a programmer. Note that the parser application requires the newest version of the Java Runtime Environment (JRE) to be installed.

Figure 2.1 shows the main GUI screen of the Azoteq HEX file parser program. The output header file is saved per default to the same directory as the Azoteq HEX file parser program. This setting can however be changed by clicking on . The output filename can also be specified.

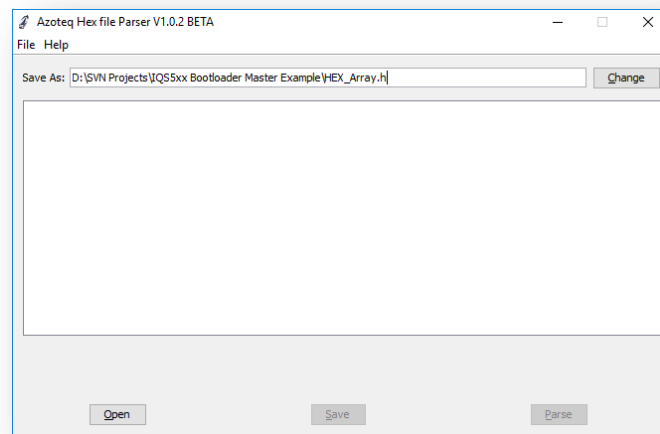






Figure 2.1 Home screen of the Azoteq HEX file parser program

The input HEX file can be selected by clicking on . A preview of the HEX file is shown in the text area. The  button becomes visible once a file has been loaded. Click the  button to parse the HEX file into a header (.h) file. If the output (save as) directory is correct, click  which saves the header file to the directory displayed in “Save as:” as shown in Figure 2.2.

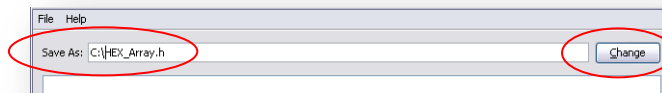


Figure 2.2 Save directory and change directory button

Figure 2.3 shows how the parser program looks after a HEX file has been parsed.

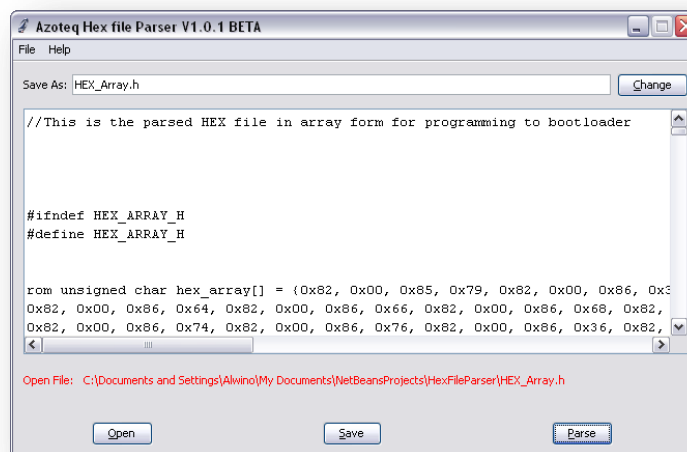


Figure 2.3 The final step in the HEX file parsing

Once the file has been saved (default filename is `HEX_Array.h`) it can be used to update the `Hex_Array.h` file found in the example project. In the example code, the correct include name is



already specified, thus the header file is automatically included. The header file contains an array of the program data which is in the correct order to be sent to the IQS5xx via I²C, to upgrade the firmware to the required version. A smaller second array contains the checksum data block.

The data of each array corresponds to the IQS5xx-B000 memory as follows:

Table 2.1 Parser header file arrays

Array	Size	Corresponding address on IQS5xx	Description
hex_array[]	15 360 bytes (240 blocks of 64 bytes)	0x8400 – 0xBFFF	This array contains the program and custom settings data for the IQS5xx-B000 trackpad
crc_array[]	64 bytes (1 block)	0x83C0 – 0x83FF	This array contains the checksum data

The contents of the output HEX_Array.h file need to be modified to adhere to the syntax of the language used by the firmware programmer. The large program array must be stored in the program memory due to space constraints. For the example code, the firmware programmer uses the Arduino language (a C variant).

The declaration of the array that contains the program data and the smaller crc array must be changed as shown in the table below.

Table 2.2 Updates to parser output file

Array	Old declaration	New declaration
hex_array	rom unsigned char hex_array[]	const uint8_t hex_array[] PROGMEM
crc_array	rom unsigned char crc_array[]	uint8_t crc_array[]

Now with the Hex_Array.h updated correctly according to Table 2.2 for the Arduino environment, it can be used to replace the Hex_array.h file in the example implementation. Now the example code has a customised firmware image to be loaded onto the IQS5xx IC.

3 IQS5xx I²C Example Bootloader Code

The files required for the example code are as follows:

```
IQS5xx_Example_Code.ino  
defs.h  
IQS5xx.cpp  
IQS5xx.h  
I2C.cpp  
I2C.h  
HEX_Array.h
```

The example bootloader implementation follows the suggested procedure described in the following document: *IQS5xx I2C Bootloader v2.x Technical User Guide v0.04.pdf*.

The bootloader-specific section is found in the file *IQS5xx.cpp* in the function *ProgramIQS5xx()*. The rest of the surrounding code is an example master implementation for the IQS5xx-B000 slave device, where the trackpad data is displayed on the serial terminal. For further information on the



example implementation, you can find documentation on the Azoteq website under Software & Tools -> IQS5XX B000 Example Code.

3.1 Programming summary

The steps for programming firmware to the bootloader, as implemented in the example code, are as follows.

1. Enter bootloader.
2. Read and verify the bootloader version number to verify bootloader entry and successful communication with correct version of bootloader.
3. Write the new application firmware and settings (0x8400-0xBFFF) to the device.
4. Write the checksum descriptor section (0x83C0-0x83FF).
5. Verify all programming was successful:
 - Perform a CRC check to verify the Application code section.
 - Read back the non-volatile custom settings section (0xBE00-0xBFFF) which is not included in the CRC calculation. Compare this to the data in the hex_array[] to verify that each byte matches.
6. Exit bootloader mode either by command or by resetting the IQS5xx (toggle NRST low).

3.2 Successful implementation

For details on compiling and uploading the project to the Arduino, and also configuring the serial monitor for IQS5xx-B000 data output, please refer to the readme document available in the master example project for the IQS5xx-B000 (Azoteq website under Software & Tools -> IQS5XX B000 Example Code).

If the project was correctly compiled, and the firmware bootloader programming process was successful, the following messages will be displayed on the serial terminal.

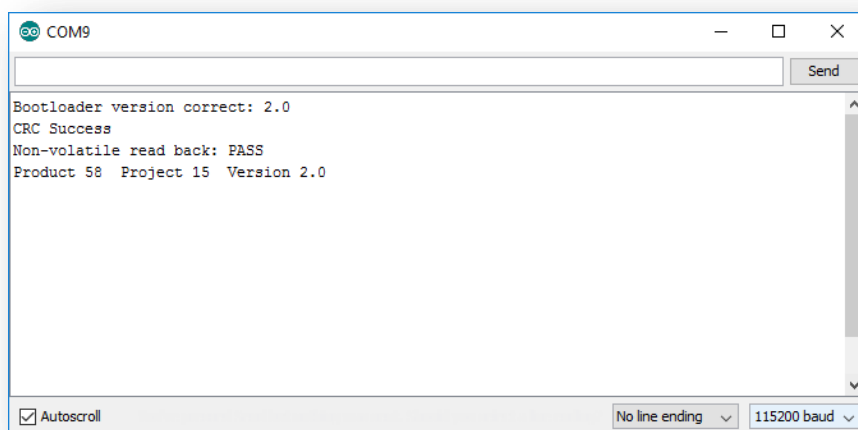


Figure 3.1 Successful bootloader programming

Once the firmware programming has completed, the trackpad should be functioning correctly. This can be verified by interacting with the trackpad and observing the output data in the terminal.

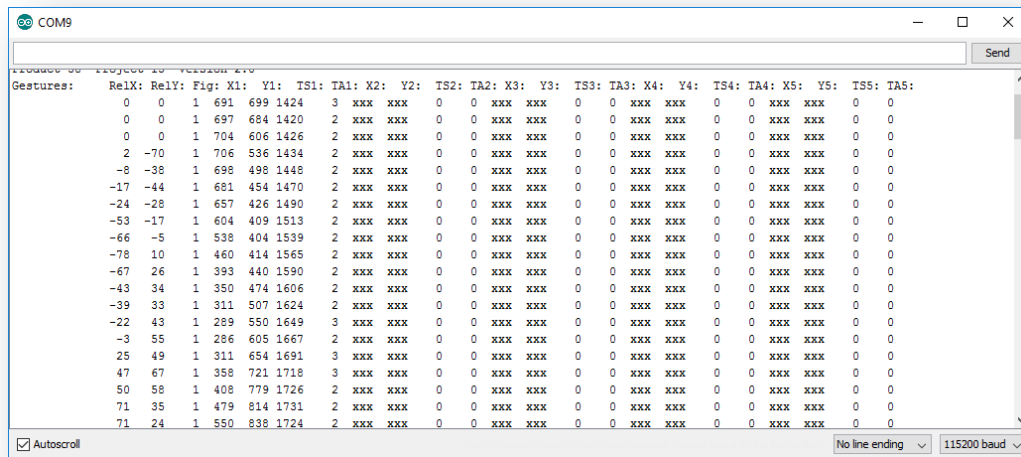


Figure 3.2 Trackpad output data

4 Summary

Using this example implementation, a firmware update for the IQS5xx-B000 can easily be implemented. Since the IQS5xx-B000 is designed to allow customers to program their custom setting together with the application firmware during the production testing stage, this is an important step in the production process.

Field updates could also be required, and then the bootloader programming also needs to be utilised to achieve this.



5 Contact Information

	USA	Asia	South Africa
Physical Address	6507 Jester Blvd Bldg 5, suite 510G Austin TX 78750 USA	Rm2125, Glittery City Shennan Rd Futian District Shenzhen, 518033 China	109 Main Street Paarl 7646 South Africa
Postal Address	6507 Jester Blvd Bldg 5, suite 510G Austin TX 78750 USA	Rm2125, Glittery City Shennan Rd Futian District Shenzhen, 518033 China	PO Box 3534 Paarl 7620 South Africa
Tel	+1 512 538 1995	+86 755 8303 5294 ext 808	+27 21 863 0033
Fax	+1 512 672 8442		+27 21 863 1512
Email	info@azoteq.com	info@azoteq.com	info@azoteq.com

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

The following patents relate to the device or usage of the device: US 6,249,089; US 6,952,084; US 6,984,900; US 7,084,526; US 7,084,531; US 8,395,395; US 8,531,120; US 8,659,306; US 8,823,273; US 9,209,803; US 9,360,510; EP 2,351,220; EP 2,559,164; EP 2,656,189; HK 1,156,120; HK 1,157,080; SA 2001/2151; SA 2006/05363; SA 2014/01541; SA 2015/023634

IQ Switch®, SwipeSwitch™, ProxSense®, LightSense™, AirButton™, ProxFusion™, Crystal Driver™ and the  logo are trademarks of Azoteq.

The information in this Datasheet is believed to be accurate at the time of publication. Azoteq uses reasonable effort to maintain the information up-to-date and accurate, but does not warrant the accuracy, completeness or reliability of the information contained herein. All content and information are provided on an "as is" basis only, without any representations or warranties, express or implied, of any kind, including representations about the suitability of these products or information for any purpose. Values in the datasheet is subject to change without notice, please ensure to always use the latest version of this document. Application specific operating conditions should be taken into account during design and verified before mass production. Azoteq disclaims all warranties and conditions with regard to these products and information, including but not limited to all implied warranties and conditions of merchantability, fitness for a particular purpose, title and non-infringement of any third party intellectual property rights. Azoteq assumes no liability for any damages or injury arising from any use of the information or the product or caused by, without limitation, failure of performance, error, omission, interruption, defect, delay in operation or transmission, even if Azoteq has been advised of the possibility of such damages. The applications mentioned herein are used solely for the purpose of illustration and Azoteq makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Azoteq products are not authorized for use as critical components in life support devices or systems. No licenses to patents are granted, implicitly, express or implied, by estoppel or otherwise, under any intellectual property rights. In the event that any of the abovementioned limitations or exclusions does not apply, it is agreed that Azoteq's total liability for all losses, damages and causes of action (in contract, tort (including without limitation, negligence) or otherwise) will not exceed the amount already paid by the customer for the products. Azoteq reserves the right to alter its products, to make corrections, deletions, modifications, enhancements, improvements and other changes to the content and information, its products, programs and services at any time or to move or discontinue any contents, products, programs or services without prior notification. For the most up-to-date information and binding Terms and Conditions please refer to www.azoteq.com

www.azoteq.com/ip

info@azoteq.com