# Data Sheet

*FC301 USB-HEAR Module*
*USB High-end Audio Receiver*
*Version 1.0.0*

*February 10, 2022*
*Build 2*

# Contents

# 1 Overview

The FC301 USB-HEAR Module is a microcontroller-based USB audio receiver solution for HiFi DAC and amplifier applications. It is implemented as a system on module (SoM) which enables easy and rapid integration.
The FC301 USB-HEAR Module implements a USB audio class 2.0 compliant PCM and DSD playback interface compatible with Windows 11/10, macOS and Linux hosts. For Windows, a bit-perfect, high-res audio driver is available.
For proper integration with a DAC, the module supports an external master clock input. The receiver can be also controlled and monitored through an I2C register interface and a set of control and status lines.

## 1.1 Block diagram

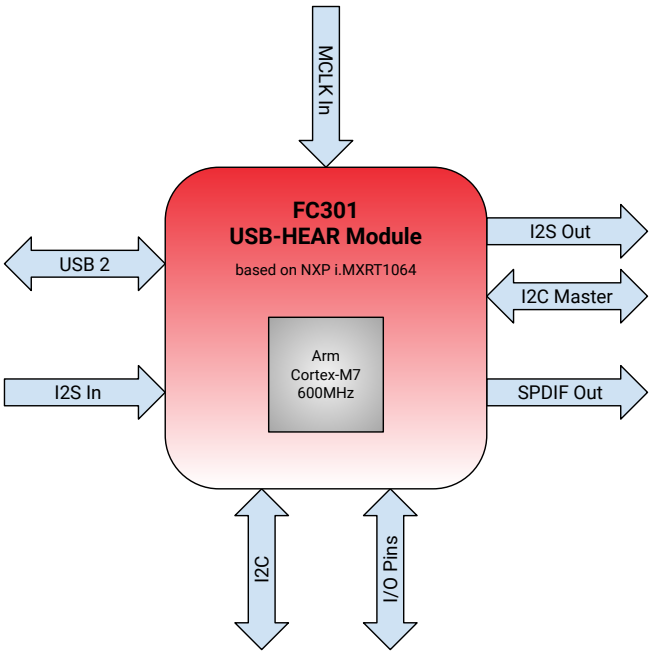


Figure 1: Block diagram FC301 USB-HEAR Module

## 1.2 Key Features

USB Audio:

- USB audio class 2.0 compliant isochronous streaming, two audio channels (stereo)
- Asynchronous clocking model, USB playback stream is synchronized to master oscillator

- 32-bit PCM sample format up to 768 kHz sampling rate
- Native DSD up to DSD1024, compatible with Windows driver and Linux
- DSD over PCM (DoP) up to DSD256

DAC Output:

- I2S compliant digital interface for connecting a digital-to-analog converter
- 32-bit PCM format up to 768 kHz sampling rate
- DSD format up to DSD1024

SPDIF Output:

- 24-bit PCM format up to 192 kHz sampling rate

Digital Input:

- I2S compliant digital interface for connecting external digital sources such as a streaming client or tuner
- 32-bit PCM format up to 192 kHz sampling rate

Control Interface:

- Control and status register set accessible via I2C
- Interrupt status line for notifications on status register changes
- Status lines for master clock selection, PCM/DSD format selection and DAC mute

Optional UART Debug Interface:

- For debugging/trace purpose, FC301 USB-HEAR Module has 3-pin Dbg-Interface (CN2) which can be assembled with 3-pin Mini-Pitch connector. (Is already assembled using FC920 USB-HEAR DevKit)

## 1.3 Additional Features

USB HID Consumer Control:

- Multimedia key events can be sent to the host to control a player application
- For example: Volume Up, Volume Down, Play, Pause, Fast Forward, Fast Backward etc.

Firmware Update:

- Firmware update via USB supported on Windows, macOS, and Linux
- Firmware update via I2C from main controller

## 1.4 Supported Operating Systems

USB-HEAR works with built-in drivers on Windows 11, Windows 10, macOS and Linux. To enable the full feature set, specifically native DSD playback, a custom driver is available for Windows 11/10.

# 2 Module Pins

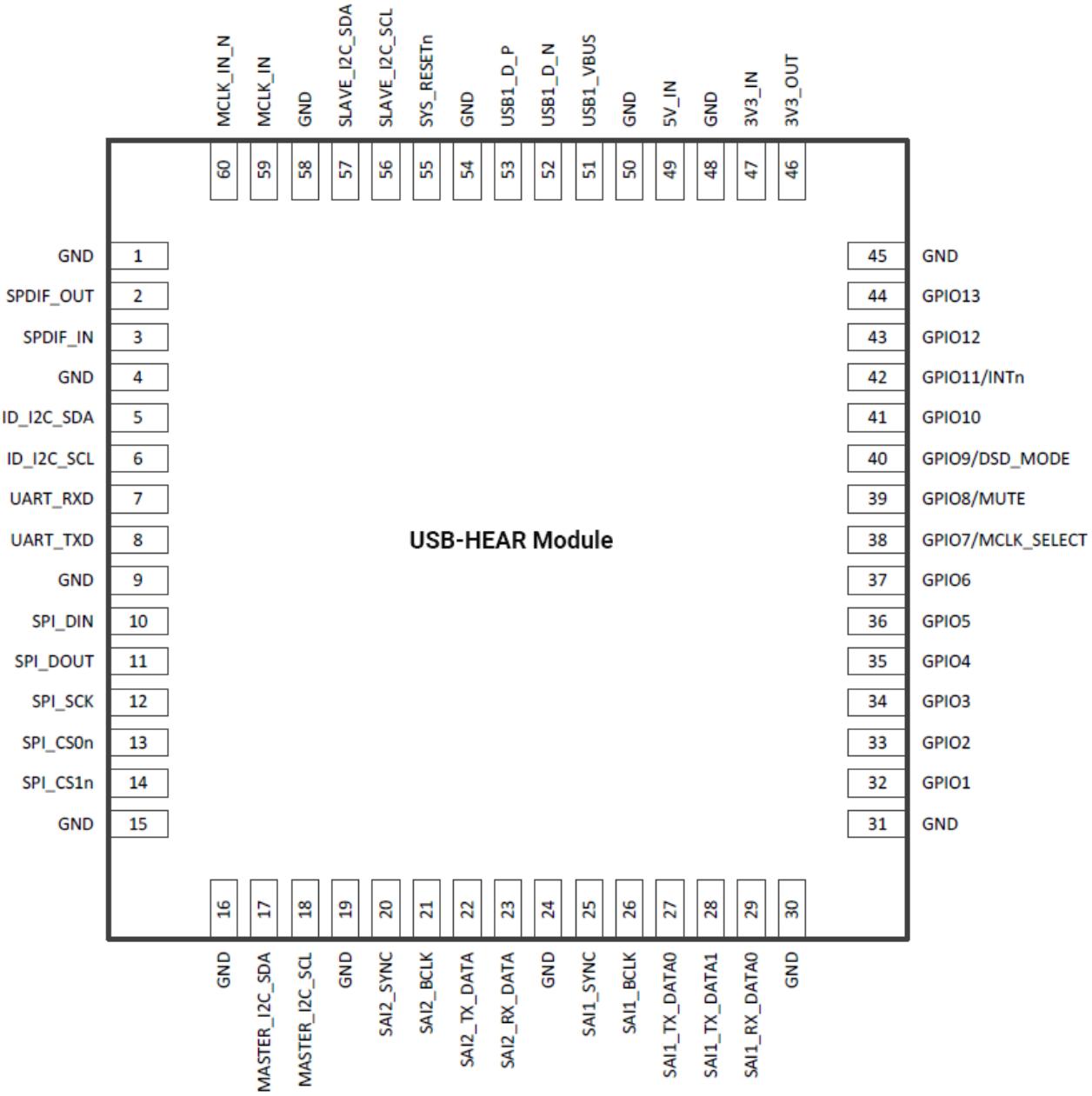


Figure 2: USB-HEAR Module Pinout

NOTE: All digital I/O pins use 3.3V logic.

## 2.1 Supply and Reset Pins

| Pin | Name | Description |
|---|---|---|
| 1, 4, 9, 15, 16, 19, 24, 30, 31, 45, 48, 50, 54, 58 | GND | System ground. |
| 49 | 5V_IN | 5V supply input to internal DC/DC regulator. |
| 46 | 3V3_OUT | 3.3V output of internal DC/DC regulator. |
| 47 | 3V3_IN | 3.3V supply input.<br>To use a 5V supply and the internal DC/DC regulator, connect this pin to 3V3_OUT. Alternatively connect it to an external 3.3V supply. |
| 55 | SYS_RESETn | Reset input, active LOW. |

## 2.2 USB Pins

| Pin | Name | Description |
|---|---|---|
| 51 | USB1_VBUS | USB VBUS sensing.<br>To be connected to VBUS provided by the host. |
| 52 | USB1_D_N | USB 2.0 high speed D- data line. |
| 53 | USB1_D_P | USB 2.0 high speed D+ data line. |

## 2.3 Clock Pins

| Pin | Name | Description |
|---|---|---|
| 59 | MCLK_IN | Master clock input. |
| 60 | MCLK_IN_N | Master clock input for differential clock signal. Reserved for future use. |
| 38 | GPIO7/MCLK_SELECT | Output that indicates which clock is to be provided to MCLK_IN.<br>LOW: MCLK_IN = 44100 Hz * 1024 = 45.1584 MHz<br>HIGH: MCLK_IN = 48000 Hz * 1024 = 49.152 MHz |

## 2.4 DAC Output Pins

| Pin | Name | Description |
|---|---|---|
| 25 | SAI1_SYNC | I2S L/R clock output. |
| 26 | SAI1_BCLK | I2S bit clock output. |
| 27 | SAI1_TX_DATA0 | I2S data output. |
| 28 | SAI1_TX_DATA1 | R channel output in DSD mode. |
| 29 | SAI1_RX_DATA0 | I2S data input. Reserved for future use. |
| 40 | GPIO9/DSD_MODE | Output that indicates in which mode SAI1 is currently operating.<br>LOW:<br>PCM mode, SAI1_TX_SYNC, SAI1_TX_BCLK and SAI1_TX_DATA0 active.<br>HIGH:<br>DSD mode, SAI1_TX_BCLK, SAI1_TX_DATA0 and SAI1_TX_DATA1 active. |
| 39 | GPIO8/MUTE | Output that indicates whether or not SAI1 signals are valid and stable.<br>LOW:<br>SAI1 and DSD_MODE outputs stable.<br>HIGH:<br>SAI1 and DSD_MODE outputs not stable because a clock or mode switch is in progress. |

## 2.5 SPDIF Pins

| Pin | Name | Description |
|---|---|---|
| 2 | SPDIF_OUT | SPDIF output. |
| 3 | SPDIF_IN | SPDIF input. Reserved for future use. |

## 2.6 I2S Input Pins

| Pin | Name | Description |
|---|---|---|
| 20 | SAI2_SYNC | I2S L/R clock input. |
| 21 | SAI2_BCLK | I2S bit clock input. |
| 23 | SAI2_RX_DATA | I2S data input. |
| 22 | SAI2_TX_DATA | I2S data output. Unused. |

## 2.7 I2C Slave Pins

| Pin | Name | Description |
|---|---|---|
| 56 | SLAVE_I2C_SCL | Slave I2C bus for internal register access. |
| 57 | SLAVE_I2C_SDA | |
| 42 | GPIO11/INTn | Interrupt output. LOW active, open drain. |

## 2.8 I2C Master Pins

| Pin | Name | Description |
|---|---|---|
| 18 | MASTER_I2C_SCL | I2C bus for external CODEC control. |
| 17 | MASTER_I2C_SDA | |
| 6 | ID_I2C_SCL | I2C bus for ID PROM access. |
| 5 | ID_I2C_SDA | |

## 2.9 UART Pins

| Pin | Name | Description |
|---|---|---|
| 7 | UART_RXD | Optional Debug interface. |
| 8 | UART_TXD | |

## 2.10 SPI Pins

| Pin | Name | Description |
|---|---|---|
| 10 | SPI_DIN | Reserved for future use. |
| 11 | SPI_DOUT | |
| 12 | SPI_SCK | |
| 13 | SPI_CS0n | |
| 14 | SPI_CS1n | |

## 2.11 GPIO Pins

| Pin | Name | Description |
|---|---|---|
| 32 | GPIO1 | Reserved for future use. |
| 33 | GPIO2 | |
| 34 | GPIO3 | |
| 35 | GPIO4 | |
| 36 | GPIO5 | |
| 37 | GPIO6 | |
| 41 | GPIO10 | |
| 43 | GPIO12 | |
| 44 | GPIO13 | |

# 3 Power Supply

The USB-HEAR module requires a 3.3V power supply for its internal components. Power has to be provided through the 3V3_IN pin as shown in Figure 3 below.

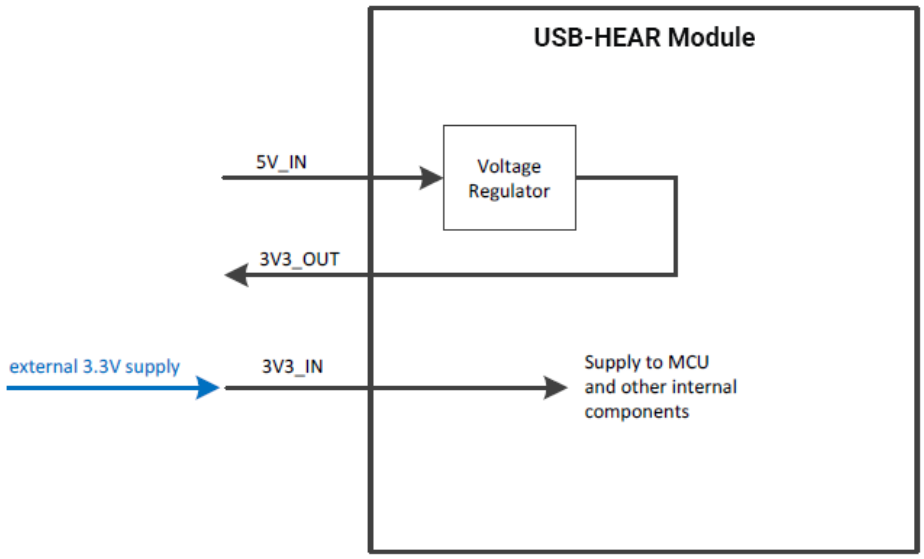


Figure 3: Application with external 3.3V supply, internal regulator unused

If there is a 5V supply available only in the given application then the module's internal regulator can be used to generate the required 3.3V supply. This requires an external connection from 3V3_OUT to 3V3_IN as shown in Figure 4 below.

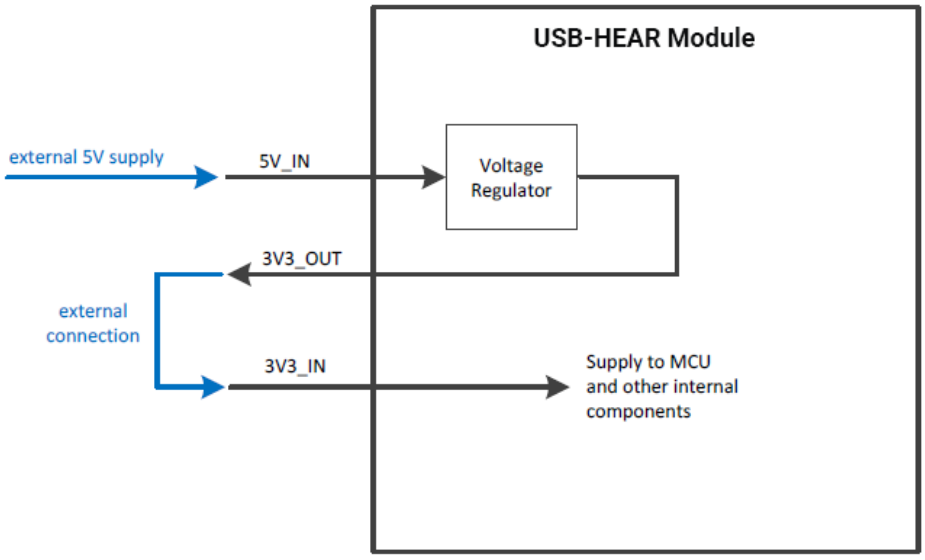


Figure 4: Application with external 5V power supply and internal voltage regulator

# 4 DAC Output

The DAC output supports two modes, PCM and DSD.

## 4.1 PCM Mode

In PCM mode an I2S compatible protocol is used. A single data line carries left and right channels interleaved. The most significant bit of each sample word is transmitted first.

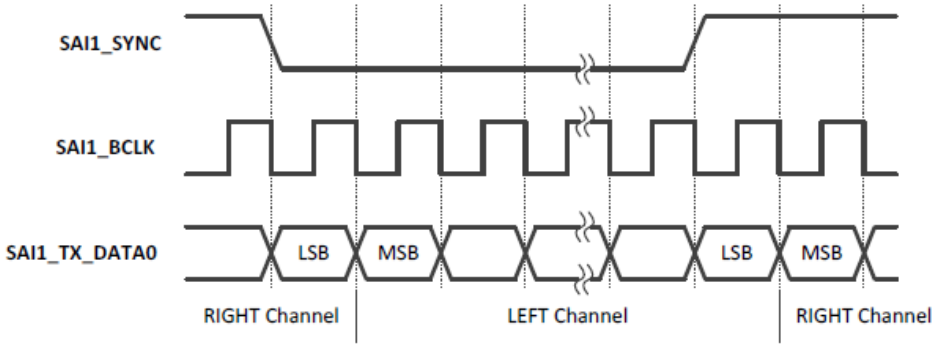


Figure 5: I2S compatible protocol in PCM mode

## 4.2 DSD Mode

DSD mode uses two data lines to carry left and right channel bit streams.

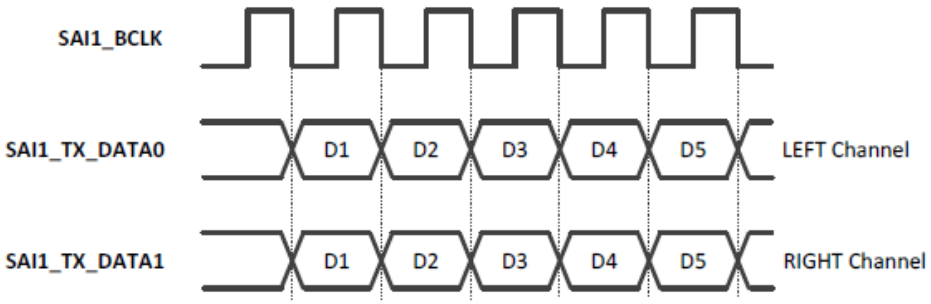


Figure 6: Two data lines and bit clock in DSD mode

# 5 I2C Register Access

The USB-HEAR module implements a set of control and status registers which are accessible via I2C. This enables runtime control and monitoring of the module's operation for an application processor.

## 5.1 I2C Device Address

The 7-bit I2C slave address is 0x59. The layout of the slave address byte is as follows:

| **Bit:** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|---|---|---|---|---|---|---|---|---|
| **Value:** | 1 | 0 | 1 | 1 | 0 | 0 | 1 | R/W |

## 5.2 Register Address and Size

Each register has a unique address in the range 0..255. The register address is 8 bits (one byte) in size. Register size varies. There are 1-byte, 2-byte and 4-byte registers. For multi-byte registers, bytes are transferred in little-endian byte order: least significant byte first.

## 5.3 I2C Transactions

Details on supported transactions are depicted below. Symbols used in the description are defined as follows.

| | |
|---|---|
| START | Start condition |
| STOP | Stop condition |
| SLA+W | Slave address + R/W bit cleared |
| SLA+R | Slave address + R/W bit set |
| RA | Register address (0x00...0xFF) |
| DataX | Data bytes, Data0 = least significant byte, DataN = most significant byte |

## 5.4 Register Write Transaction

To write a register, a master initiates a single I2C transaction which has the following layout.

| START | SLA+W | RA | Data0 | Data1 | … | DataN | STOP |
|---|---|---|---|---|---|---|---|

Figure 7: Register write access

The number of data bytes transferred in a write transaction must correspond exactly to the size of the addressed register. A partial register write is not supported.

## 5.5 Register Read Transaction

To read from a register, a write transaction followed by a read transaction can be used as shown below.

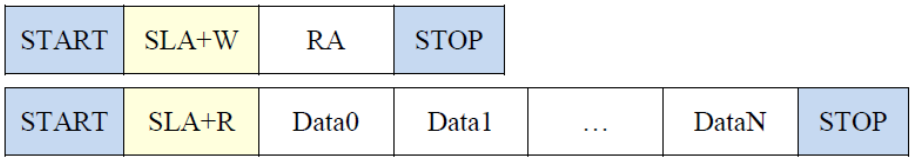


Figure 8: Register read access using two separate transactions

Alternatively, a combined I2C transaction can be used by concatenating a write and a read transaction with a repeated start condition. This is shown below.

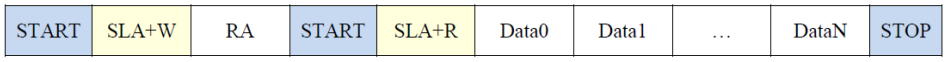


Figure 9: Register read access using a repeated start condition

A partial register read operation is supported, i.e. the number of data bytes requested by the master in a read operation can be less than the size of the addressed register.

# 6 Control and Status Registers

| Address | Register | Size (bytes) | Access | Description |
|---|---|---|---|---|
| 0x00 | INTERFACE_VERSION | 2 | RO | Version of the register layout. |
| 0x01 | FIRMWARE_VERSION | 2 | RO | Current firmware version. |
| 0x04 | MAGIC_PATTERN | 4 | RO | Contains the constant value 0xB3B2B1B0. Useful for testing register access, byte order, etc. |
| 0x05 | SCRATCH | 4 | RW | An arbitrary value can be written and then read back. Useful for testing register access. |
| 0x08 | LAST_WR_STATUS | 4 | RO | Completion status of the last register write transaction. |
| 0x09 | LAST_RD_STATUS | 4 | RO | Completion status of the last register read transaction. |
| 0x10 | INT_STATUS | 2 | RW | Interrupt status bits. A one must be written to clear a status bit. |
| 0x11 | INT_STATUS_AC | 2 | RO | Same as INT_STATUS but with auto-clear as a side effect on read access. |
| 0x12 | INT_ENABLE | 2 | RW | Interrupt enable bit mask. |
| 0x21 | DACOUT_CTRL | 1 | RW | Control bits for the DAC output. |
| 0x22 | DACOUT_STATUS | 2 | RO | Bits that reflect the current operating mode and status of the DAC output. |
| 0x28 | INPUT_SEL | 1 | RW | Switches between USB and I2S input stream. |
| 0x30 | USB_CTRL | 1 | RW | Control bits for the USB device. |
| 0x31 | USB_STATUS | 1 | RO | Bits that reflect the current status of the USB device. |
| 0x32 | USBAUD_STATUS | 2 | RO | Bits that reflect the current operating mode and status of the USB audio device. |
| 0x40 | DIGI_IN_CTRL | 1 | RW | Control bits for the I2S input. |
| 0x41 | DIGI_IN_STATUS | 2 | RO | Bits that reflect the current status of the I2S input. |
| 0x70 | HIDCC_EVENT | 2 | WO | A write to this register posts a HID Consumer Control event that will be forwarded to the host. |
| 0x71 | HIDCC_KEY | 2 | WO | A write to this register posts a HID Consumer Control "key pressed" plus "key released" event. |

Detailed register reference documentation is available separately.

# 7 Electrical Characteristics

## 7.1 DC Supply Characteristics

| Description | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| 5 V power supply to internal regulator | 5V_IN | tbd. | 5 | tbd. | V |
| 3.3 V output from internal regulator | 3V3_OUT | tbd. | 3.3 | tbd. | V |
| 3.3 V power supply to internal components | 3V3_IN | tbd. | 3.3 | tbd. | V |
| | | | | | |
| 5 V power supply current | $I_{5V\_IN}$ | – | tbd. | tbd. | mA |
| 3.3 V sink current (internal regulator) | $I_{3V3\_OUT}$ | – | tbd. | tbd. | mA |
| 3.3 V power supply current | $I_{3V3\_IN}$ | – | tbd. | tbd. | mA |

## 7.2 Digital Pin Characteristics

| Description | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| HIGH input voltage | $V_{IH}$ | tbd. | – | tbd. | V |
| LOW input voltage | $V_{IL}$ | tbd. | – | tbd. | V |
| HIGH output voltage | $V_{OH}$ | tbd. | – | tbd. | V |
| LOW output voltage | $V_{OL}$ | tbd. | – | tbd. | V |
| | | | | | |
| HIGH output current | $I_{OH}$ | – | – | tbd. | mA |
| LOW output sink current | $I_{OL}$ | – | – | tbd. | mA |

# 8 Physical Dimensions

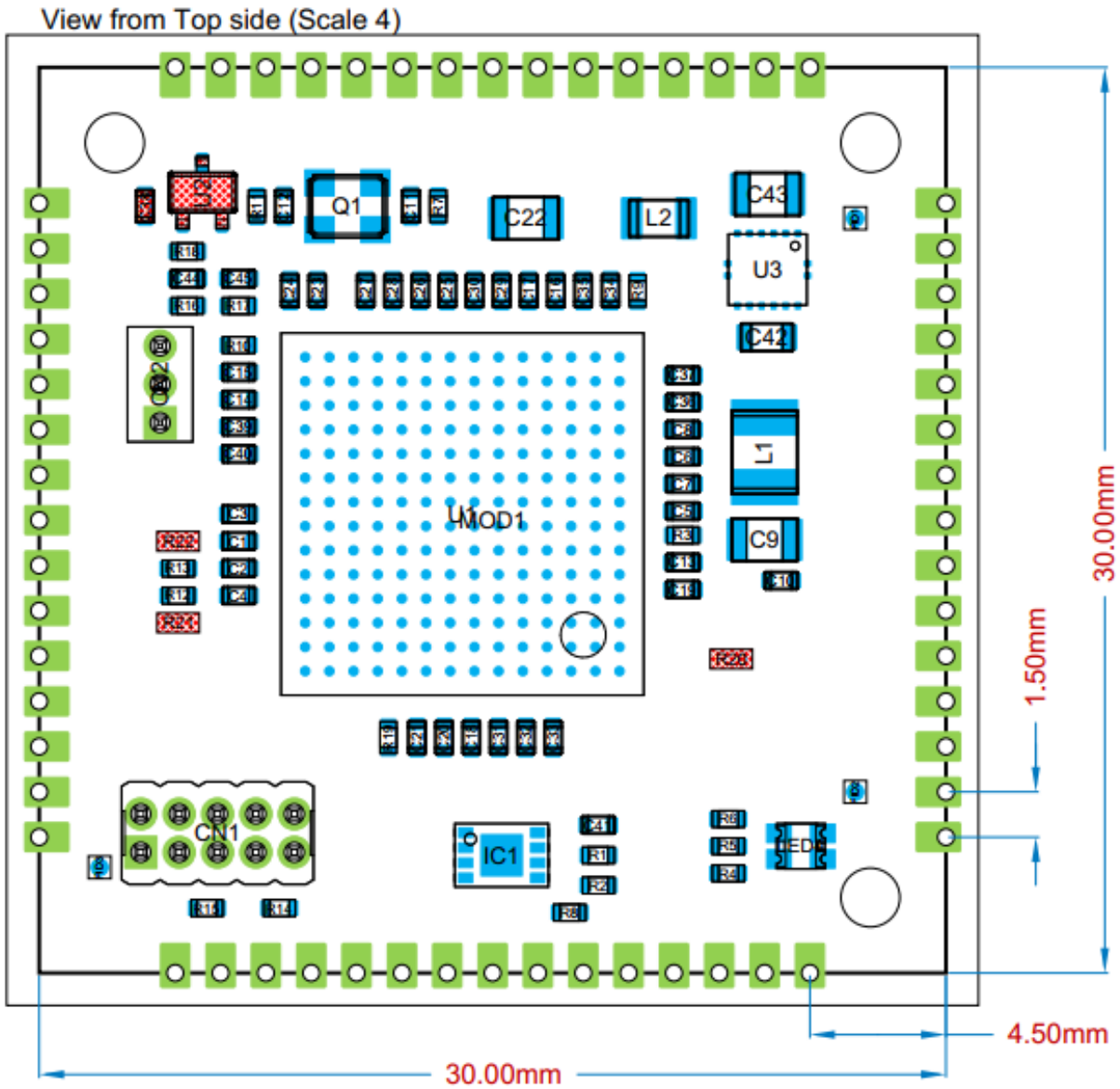


Figure 10: FC301 USB-HEAR Module top view

# 9 Debug-Tools

## 9.1 UART-Interface (CN2)

For debugging/trace purpose, FC301 USB-HEAR Module has 3-pin Dbg-Interface (CN2) which can be assembled with 3-pin Mini-Pitch connector. (Is already assembled using FC920 USB-HEAR DevKit)

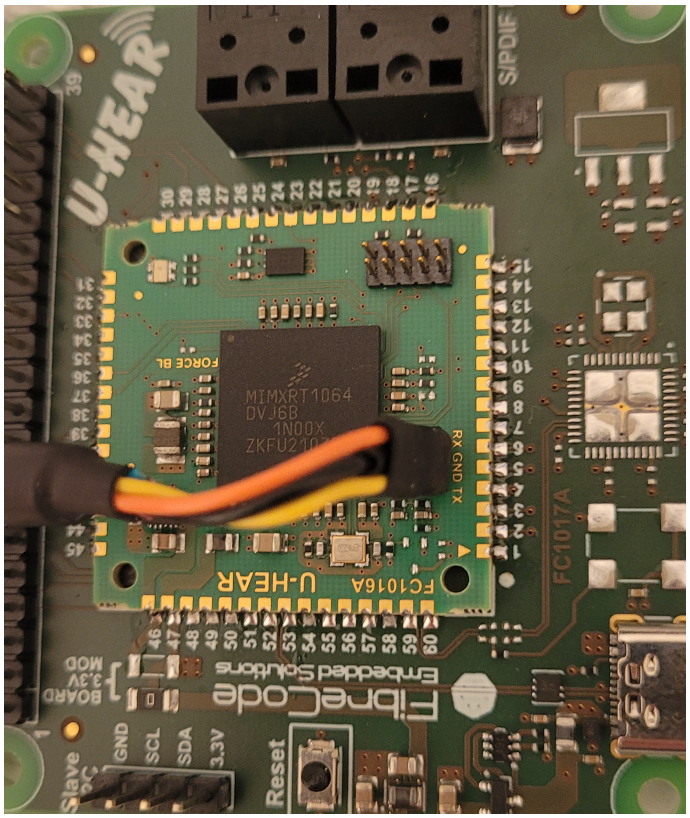


Figure 11: UART Connector on FC920 USB-HEAR DevKit

| **Signal** | **Color** | **FibreCode Board Pin** |
|---|---|---|
| TXD (output) | Orange | RX |
| GND (ground) | Black | GND |
| RXD (input) | Yellow | TX |

UART settings: 921600 baud, 8 data bits, no parity, 1 stop bit (8N1).