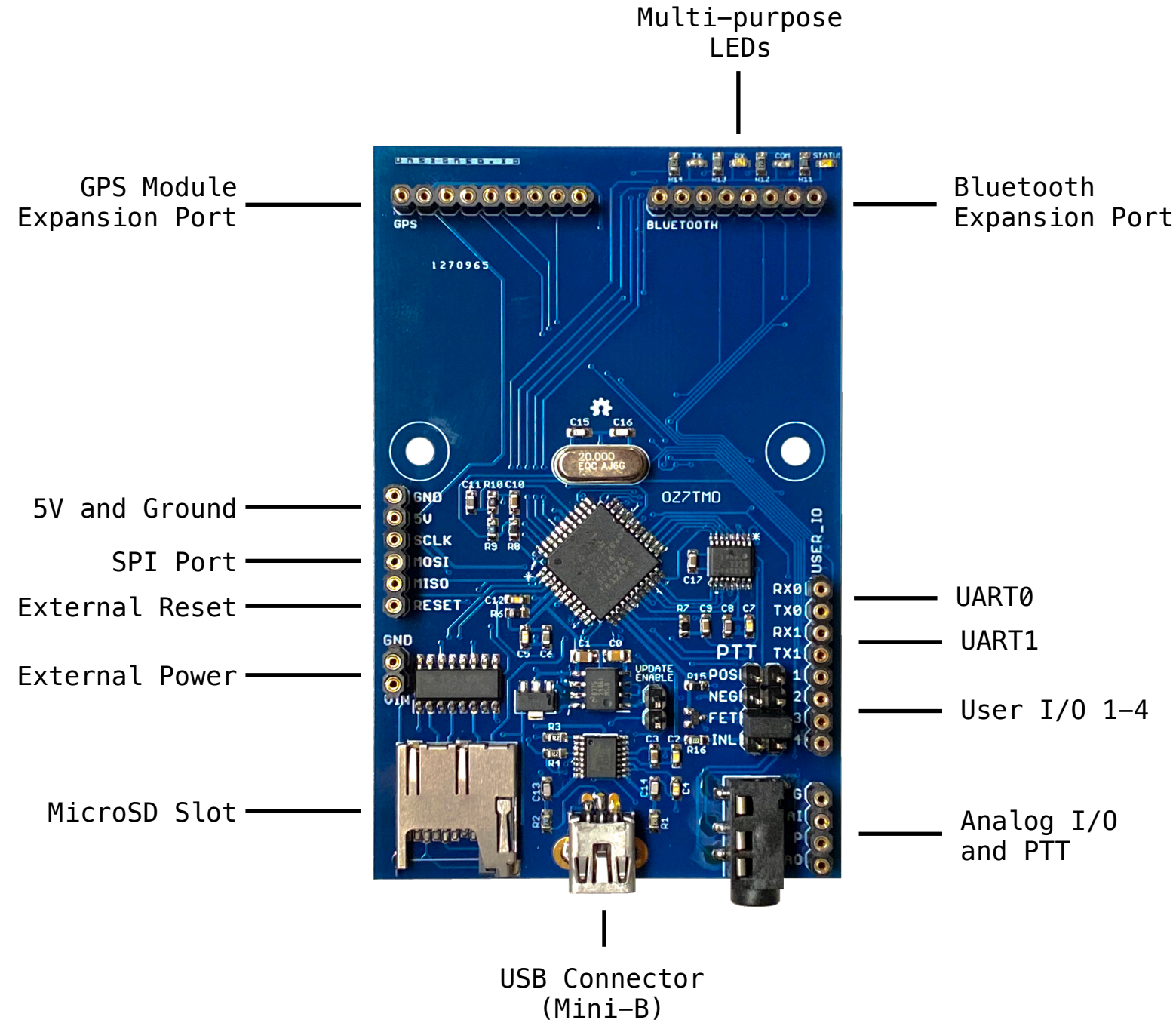
OpenModem User Manual

Thank you very much for buying this product! If you have any questions, suggestions, criticisms or good ideas, I'd love to hear from you! This guide provides a few pointers on getting started with OpenModem, how to connect a radio or other device to the modem, and how to configure it.

Device Overview

Please have a look at the following chart of the internals of OpenModem and familiarise yourself with the different ports, connectors and indicators.



⚠ Cautions

- Always ensure a proper power supply to the device. The device can be powered from the USB-connector, or from an external power supply on the `External Power` port. The device can be powered from a DC voltage between **5V** and **16V**. Always ensure that the polarity and voltage is correct! An incorrect supply voltage or polarity **will damage the device**.

- Always observe your **local laws and regulations** regarding RF emissions. If this device is connected to a radio, it may be capable of generating emission modes that are illegal in your jurisdiction. It is **your** responsibility to ensure that you are operating radio equipment in a lawful way.

Specifications

- AFSK modem supporting **300**, **1200** or **2400** baud operation.
- Default mode is **1200 baud AFSK**.
- Powered by an **ATmega1284p** MCU clocked at **20 MHz**
- **128 kilobytes** of **flash**
- **16 kilobytes** of **RAM**
- Large packet **MTU of 576 bytes**
- **Mini-USB** connector
- Fully **programmable**
- **Arduino** compatible
- **Open source** firmware and configuration program
- **4** multi-purpose **IO ports** available
- Operating temperature range: **-20°C to 60°C** (in non-condensing humidity)

Connecting OpenModem to Host Equipment

OpenModem can be connected to any host supporting USB or serial UART connections. The board uses an FTDI USB-to-serial converter, so drivers should already be included in most operating systems. You can use either the **USB port**, or connect directly to the **UART0 RX** and **UART0 TX** pins.

By default, the serial settings are **115200 baud**, **8N1**. No UART flow control is used. The serial settings can be changed with the configuration program.

If the `UPDATE_ENABLE` jumper on the board is connected, the RTS line is connected to **External Reset**. In that case, toggling RTS will trigger a reboot of the modem.

The `UPDATE_ENABLE` jumper must be connected in order to update the firmware of the device, or to program the device from the Arduino IDE. It is recommended to keep the jumper disconnected in all other cases, and during normal operation.

Operating Modes

OpenModem can operate in a variety of modes, and since the modulator and demodulator is completely software-defined, new modes can be added via firmware updates in the future. The default mode is 1200-baud AFSK, which is the most widely used on standard packet radio networks. If you wish to use 300 or 2400 baud modes, please download the corresponding firmware from https://unsigned.io/openmodem and install it on the modem.

Configuring OpenModem

The device is useable out of the box, but can be further configured to your needs by using the OpenModem Configuration Utility. Please visit the OpenModem page at https://unsigned.io/openmodem to download the latest version of the configuration program for Mac, Linux or Windows.

OpenModem also allows host applications to directly configure the modem via the standard KISS command set for changing modem and CSMA parameters. In addition, OpenModem implements extended KISS commands to control additional functionality, such as input and output gain, packet logging and peripherals.

Multi-purpose LEDs

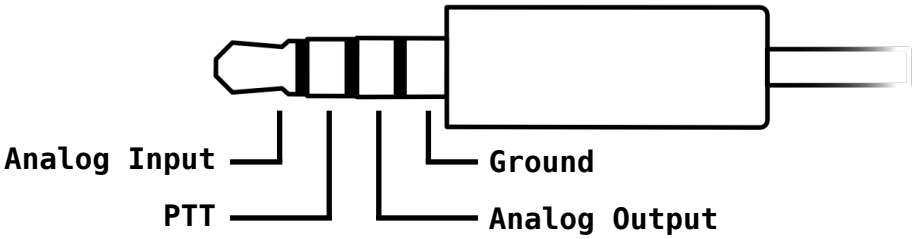
The modem is equipped with four multi-purpose LEDs at the front of the device. The LEDs are used to signify a variety of device states and events.

Pattern	Description
After powering the device: Green LED is constantly lit	Device is ready and all health checks and device verification succeeded.
When device is on: Green LED blinks quickly 4 times	SD-card was inserted and mounted correctly.
When device is on: Green LED blinks slowly 2 times	SD-card was removed and unmounted.
When device is on: Red LED lights up	Device is transmitting data to the host device or receiving data from the host device.
When device is on: Blue LED lights up	Modem has detected a valid signal, and is receiving and demodulating data.
When device is on: Orange LED lights up	Modem is modulating and transmitting data. PTT is keyed.
All LEDs flash in sequence 2 times	Encryption key loaded, and AES-128 encryption is now enabled.
Green and red LEDs blink slowly in an alternating pattern	Warning indicator, modem operation halted. Check error codes sent over serial connection.

Connecting a Radio

To connect the modem to your radio, or other transmission device, use the 4-pole 3.5mm TRRS jack connector on the back of the modem. Any 4-pole 3.5mm jack with a normal cable will do. The pinout of the port is as follows:

Tip	Input to modem
Ring 1	PTT
Ring 2	Output from modem
Shield	Ground



Or alternatively, use the pin connectors broken out directly on the modem board, right next to the jack connector. The board pin connectors are labeled `G`, `AI`, `P` and `AO` for `Ground`, `Analog Input`, `PTT` and `Analog Output`.

If your radio does not have a dedicated `DATA` , `TNC` or `MODEM` connector, you can often get away with connecting the modem directly to the radio's microphone input, speaker output, and PTT line.

The input and output gain levels of OpenModem are set to sensible defaults, but radios vary a lot, so you might need to change the levels using the configuration program. When the configuration program is connected to the modem, go to the *Audio* part of the program, and tune your radio to a frequency with AFSK packet activity. You should be able to see the incoming packets as peaks on the rolling audio meter. A blue bar means that an incoming packet was successfully decoded. Red means the audio is clipping. Tune the audio levels until you get satisfactory results.

By connecting one of the `PTT` jumpers on the OpenModem board, various PTT styles can be configured. The available options are: `Positive`, `Negative`, `MOSFET` and `In-line`. Using `Positive` will set the PTT line to 5V when PTT should be keyed. Using `Negative` will set it to 0V. Using `MOSFET` will pull the PTT line from the radio to the radio's ground using a transistor. Using `In-line` will work for radios where the PTT signal is sent in-line with the microphone signal. Some handhelds, specifically some Yaesu models, use this style. The most common option would be to use the `Negative` or `MOSFET` style, but your radio may need one of the other options.

Updating the Firmware

The firmware of the device can be updated with standard AVR utilities. If you prefer to work from a command-line, the standard `avrdude` program will work fine for updating the firmware. If you'd rather use a graphical interface, the `XLoader` or `AVRDUDESS` programs are two possibilities.

Programming

The easiest way to program OpenModem is probably from within the Arduino IDE. For adding OpenModem to your Arduino IDE, please see the page at https://unsigned.io/board-support-in-arduino-ide/. It is also possible to work with any other toolchain that supports the ATmega1284p that powers OpenModem, like avr-gcc or similar.

If you want to make your own firmware for OpenModem, it can be a good idea to use the default firmware as a starting point, since a lot of the setup required to use OpenModem, and all the hardware drivers, are already written for you. The source code for the firmware is available under GPL-3.0 and can be found at https://github.com/markqvist/OpenModem.

Interfacing with OpenModem

OpenModem communicates with host devices using the KISS protocol, which is standard for most packet radio hardware. Practically any relevant software should be able to connect to the modem via KISS.

If you want to interface directly with the modem from your own programs or a serial console, please refer to the protocol reference specified later in this manual.

Flow Control and Packet Buffer

OpenModem employs a large packet buffer, that should be sufficient for even the most complex packet applications, and also to run IP-based programs using `SLIP` or `kissattach` on Linux. When the device receives data frames for transmission from the host, they will be placed in the packet buffer, and transmitted in the order they were received, as soon as the channel is clear, and according to the configured CSMA parameters.

Packet Format and MTU

OpenModem allows large packets with a size of up to 576 bytes. It uses the standard HDLC framing that is commonly in use on packet radio networks. Any kind of data can be carried inside the HDLC frames, including AX.25, which most amateur radio packet programs utilise.

As such, OpenModem is directly compatible with most packet radio software in existence, but can also be used to transport any other kind of data, such as Ethernet or Reticulum frames.

Getting Help

If you have any questions regarding OpenModem, please do not hesitate to send me an email at mark@unsigned.io. You can also register for the forums at unsigned.io where other users might offer pointers and advice.

USB and Serial Protocol

Communications to and from the device uses KISS framing with the default KISS command set, plus some extra commands to control things that were not in the original KISS command set.

OpenModem also does **not** use *HDLC ports* in the command byte, and as such uses the full 8 bits of the command byte is available for command specification. Please see table below for supported commands.

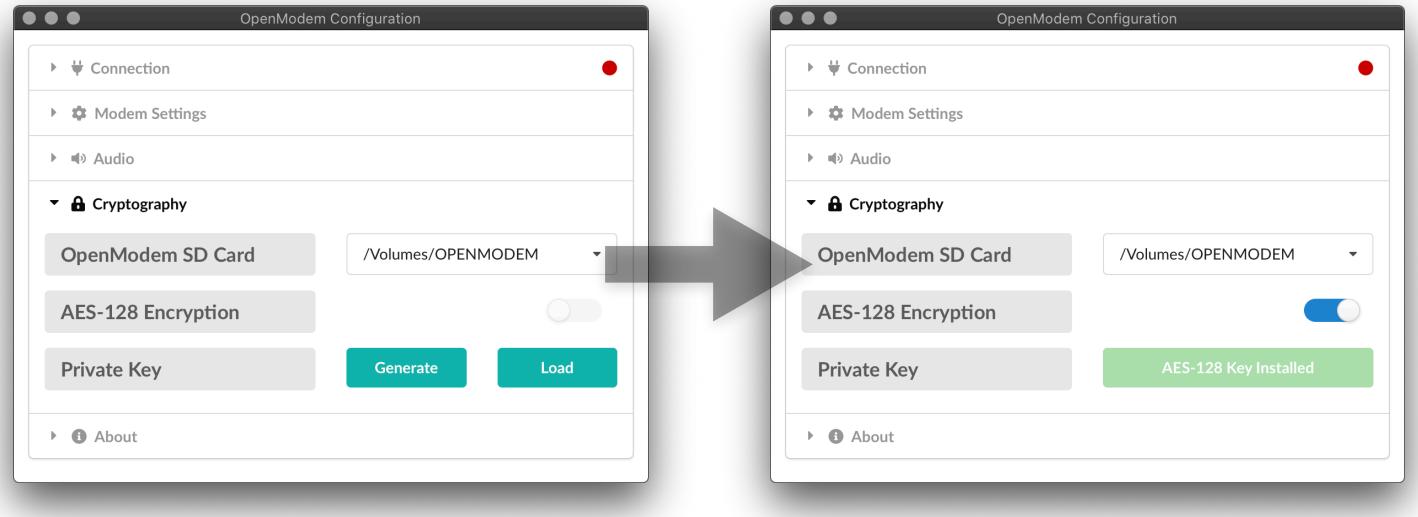
Command	Byte	Description
Data frame	0x00	A data frame to or from the device
Preamble	0x01	Get or set the transmission preamble
CSMA Persistence	0x02	Get or set the CSMA Persistence parameter
CSMA Slot Time	0x03	Get or set the CSMA Slot Time parameter
TX Tail	0x04	Get or set the transmission tail
Full-duplex	0x05	Get or set full-duplex mode
Set hardware	0x06	KISS Set-hardware command
Save configuration	0x07	Save running configuration to EEPROM
LED Intensity	0x08	Get or set LED intensity
Output gain	0x09	Get or set output gain
Input gain	0x0A	Get or set input gain
Passall	0x0B	Skip checksum on packets and pass all data
Log packets	0x0C	Enable or disable logging packets to SD card
GPS Mode	0x0D	Get or set the GPS mode
Bluetooth Mode	0x0E	Get or set the Bluetooth mode
Serial Baudrate	0x10	Get or set the serial baudrate
Reboot	0x11	Reboot the modem
Audio peak	0x12	Audio peak level data from modem
Enable diagnostics	0x13	Enable or disable diagnostic output to serial port
Mode	0x14	Control operating mode
NMEA output	0x40	Control NMEA GPS output
Output configuration	0xF0	Dump EEPROM configuration to serial port
Return	0xFF	KISS Return command

AES-128 Encryption

OpenModem supports strong AES-128 encryption directly in-modem. When enabled, the encryption layer is completely transparent to the host and any user applications. This makes it possible to run any legacy application with a layer of strong encryption, even though the application was not originally designed to encrypt any transmitted data, or to encrypt IP traffic.

To enable encryption on OpenModem, you will need a MicroSD card of at least 64 megabytes. The card should ideally be formatted as exFAT, but FAT32 is also supported.

Insert the card into your computer, and run the OpenModem Configuration program. Select the *Cryptography* tab, and select the SD-card from the drop-down box. Press the *Generate* button in the *Private Key* section to generate a new AES-128 key and install it onto the card.

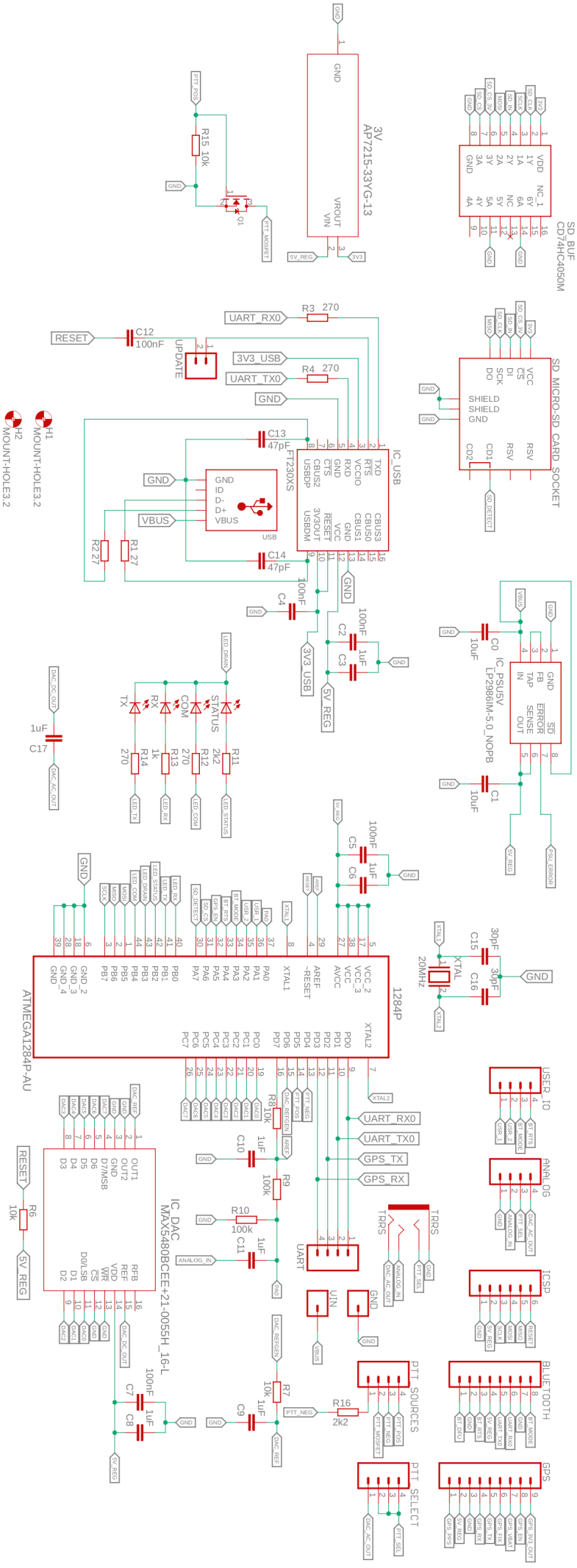


If the key is successfully installed, the *AES-128 Encryption* toggle will switch on, and the *Private Key* section will show *AES-128 Key Installed*.

You can now insert the SD card into the modem. If the modem can read the SD card, it will immediately load the key and start encrypting all outgoing traffic, and automatically decrypt all incoming traffic. To signal that encryption is enabled, the modem will flash all LEDs sequentially two times. This will happen every time the modem is started, to verify encryption is on.

When encryption has been enabled, an encryption lock will be set in the EEPROM of the modem. If the modem is not able to load the encryption key from the SD card at startup, or if the SD card is removed during operation, the modem will stop operating, and flash an error signal.

To disable encryption, you must insert the SD card into a computer, run the configuration program and manually disable the encryption checkbox. You can then insert the SD card back into the modem, which will unlock and disable encryption.



Sincere thanks to

You, for supporting OpenModem, and through that all of my open-source work

ChaN for the FatFS driver

Andrew Carter for his 8-bit AES implementation (MIT License)

Daniel Otte for the AVR-Crypto-Lib MD5 and MD5-HMAC implementations

Everyone who has supported me through the years, and made OpenModem possible.