Raging Bits Symple Link

v1.0

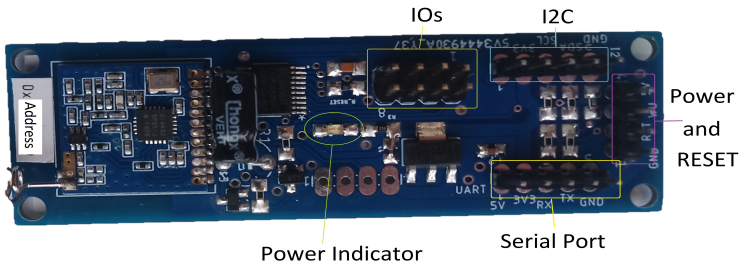
Top level specs

1x UART
1x I2C
4x IOs
1x Onboard 256Bytes EEPROM (*1)
433mHz ISM band radio with more than 500m range (LOS)
25mA/70mA radio Rx/Tx
5V input power supply

This module is a prototype device intended to speed up product development.
Although it has been developed according to OFCOM rules for ISM low power devices band, it is currently NOT LICENSED NOR CERTIFIED.

(*1) – Although the EEPROM is available for use, there is no driver interface for it, the user will need to provide the proper driver over the I2C interface.

Device interface



Radio detail pinout

IOs pins. Header numbered from 1 to 8. (Peripheral device only).

1 – GND		2- IO1	
3 – GND		4- IO2	
5 – GND		6- IO3	
7 – GND		8- IO4	

I2C header. (Peripheral device only).

1- GND		2 - SDA	
3- SCL		4 - 3V3	
5- 5V			

Power and Reset header.

1- GND		2 - RESET	
3- WakeUP		4 - 5V	

UART header.

1- 5V		2 - 3V3	
3- TX		4 - RX	
5- GND			

Power Indication LED:

When liten, indicates that the device is connected.
The color will be **Blue** for a **Central** device and **Green** for a **Peripheral** device.

Device Labels

Each device has a unique address and default key.
The device has 3 labels, 1 in the top side (Shown in the device interface picture), and 2 in the bottom as shown in the Device Labels picture.
The top side label will be near the antenna and will contain the hexadecimal address of the device.
The bottom side will have a label witl the default device key and, for easy access purposes, a 2D bar code, containing both, the address and the key.

Radio Characteristics (based on LPD433 IR2030/1/(12/51))

Base frequency (channel 0)- 433.075 Mhz

Channel deviation – 5Khz

Channel distance is 20khz.

Channel hop is 40khz (This just means that the device only use even channel numbers to minimize interfeerence.)

Modulation GFSK

Maximum emission – 20dBm (adaptative)

Power and Electrical characteristics

25mA/70mA radio Rx/Tx.

5V input power supply.

All IOs and Communication pins have a maximum input voltage of 3.3V.

Inputs and Outputs (IOs) Characteristics (Peripheral Module ONLY)

Multiple IOs can be set as outputs triggered by the same data set.

Peripheral module UART Characteristics

The peripheral UART has a basic generic configuration to suit most applications.

Fixed: 8Bit 1Start bit 1Stop bit no parity, no flow control.

Setteable: Can be configured for any speed up to 0.5Mbps.

Central module UART Characteristics

The central UART characteristics are:

19200bps, 8bits, 1 start bit, 1 stop bit, no flow control, no parity.

I2C Characteristics (Peripheral Module ONLY)

The I2C, like the UART, has the basic generic configuration to suit most applications.

Fixed: 100kHz clock speed.

Setteable: MSB or LSB first.

EEPROM Characteristics (Peripheral Module ONLY)

The EEPROM is an AT24C04 with 512Bytes of space. It is used primarily by the device for self management, for which purpose, the device has exclusive write access to the first 256B.

The second 256B are available to the user.

The EEPROM sits in the I2C bus and has no dedicated access layer. In order to interface the EEPROM, the user needs to use the I2C BUS and provide the memory driver to interface it.

An example is provided in the I2C command description.

Radio Interface Commands (Central Module ONLY)

The Peripheral module can only be controlled and instructed by a handshaked Central Module.

The Central module can only be controlled and instructed by the UART serial port.

Each radio module will have a unique address of 32bits that will be different from zero and than 0xFFFFFFFF.

Each command will have a fixed size of 128 Bytes. If the command content doesn't consume all the space, irrelevant data padding shall be added till the command sent to the Central device UART comprises a total of 128bytes.

All the commands intended to a Peripheral device, will have the Peripheral access key in the command.

Command Structure and Type

There are 2 types of commands that can be sent to the Central module, data commands and instruction commands. The Peripheral module does not take any commands that are not sent via Central module.

Command reply structure

Every command issue will have a reply with the specific format:
<0xAA><Len><Orig addr><P. type><Orig sub-addr><Data><0x55>

<0xAA> - Starter character.
<Len> - 1byte indicationg the reply total length except start and end bytes.
<Orig addr> - 4bytes with the origin address (This value is zero for instruction reply).
<P. type> - 1byte indicationg the packet type (This value is zero for instruction reply).
<Orig sub-addr> - 2bytes value with the of the reply (This value is zero for instruction reply).
<Data> - N bytes containing the reply data (This field does not exist for for instruction reply).
<0x55> - End character.

Each command sent will have a reply structure for command confirmation.
This is not the command reply, simply the confirmation that the command has been queued, by the Central, to be sent.

Instruction commands

The Instruction command format starts de device address like Data command, but this address is Zero, followed byte the instruction type, the respective instruction data and the needed padding to fullfil 128bytes total of command.

Command format:
<0x00000000><0xMM><0xNN ... 0xNN><padding to 128bytes>
0x00000000 – 4byte value zero. This value will always be zero to indicate an instruction command.
M – 1byte Instruction.
N – Variable length Instruction data.
Padding – Padding with as many bytes of ignored data as needed to fullfil a total command length of 128bytes.

Again, each command sent will have a reply structure for command confirmation.
This is not the command reply, simply the confirmation that the command has been queued, by the Central, to be sent.

Retrieve Peripheral Information (0x00)

This command instructs the device to return the device native information. Name, software version, hardware version, etc.

The command is composed by the start address (0x00000000), followed by the instruction byte value 0x00, the destination Peripheral device address and respective Peripheral key(56bytes), with added padding..

4Bytes	1Byte	4Bytes	Peripheral Key	Padding till 128 Bytes
[0x00000000]	[0x00]	[0xMMMMMMMM]	[0xKK ... 0xKK]	[0x00 ... 0x00]

Handshake with Peripheral (0x01) (mandatory to be done once with every new peripheral)

This command handshakes the the Central with the Peripheral device, being after this command, the peripheral capable of sending messages back to the Central device. Unless a device is handshake, it will not reply to Central requests.

The command is composed by the start address (0x00000000), followed by the instruction byte value 0x01, the destination Peripheral device address and key(56bytes), with added paddding.

4Bytes	1Byte	4Bytes	Peripheral Key	Padding till 128 Bytes
[0x00000000]	[0x01]	[0xMMMMMMMM]	[0xKK ... 0xKK]	[0x00 ... 0x00]

Central key update (0x02)

This command replaces the current Central key with a new one. If the key is the same as the previous one, it just remains unchanged. This command changes the key permanently, with a new

one. The Central device will now constantly use the new key.
The device will only reinstate the default key when it has the settings reset to defaults.
After the Central device key update to a different key, any Peripheral devices that had previously been handshaked with the Central device, need to be handshaked again.

The command is composed by the start address (0x00000000), followed by the instruction byte value 0x02, the respective new key(56bytes) for the Central device, with added padding.

```
    4Bytes      1Byte  Central new Key  Padding till 128 Bytes
[0x00000000][0x02][0xKK  ...  0xKK][0x00        ...        0x00]
```

Peripheral key update (0x03)

This command replaces the current Peripheral key, permanently, with a new one.
The Peripheral device will now constantly use the new key.
The device will only reinstate the default key when it has the settings reset to defaults.
After the Peripheral device key update to a different key, the Central device must start using the new key for any other packets.

The command is composed by the start address (0x00000000), followed by the instruction byte value 0x03, the destination Peripheral address and key(56bytes), the new Peripheral key(56bytes) and added padding.

```
    4Byte       1Byte      4Bytes Address       Current Key        New Key        Padding till 128 Bytes
[0x00000000][0x03][0xMMMMMMMM][0xKK  ...  0xKK][0xLL  ...  0xLL][0x00        ...        0x00]
```

Central channel set (soft update) (0x04)

This command sets the Central device channel, temporarily, to a new one.
The device will reinstate the set channel when it power cylcles.

The command is composed by the start address (0x00000000), followed by the instruction byte value 0x04, the new Central channel (1byte) and added padding.

```
    4Byte       1Byte   Channel(1byte)   Padding till 128 Bytes
[0x00000000] [0x04]      [0xMM]        [0x00        ...        0x00]
```

Central channel update (hard update) (0x05)

This command sets the Central device channel, permanently, to a new one.
The device will only reinstate the default channel when it has the settings reset to defaults.

The command is composed by the start address (0x00000000), followed by the instruction byte value 0x05, the new Central channel (1byte) and added padding.

```
    4Byte       1Byte   Channel(1byte)   Padding till 128 Bytes
[0x00000000] [0x05]      [0xMM]        [0x00        ...        0x00]
```

Peripheral channel update (hard update) (0x06)

This command sets the Peripheral device channel, permanently, to a new one.
The device will only reinstate the default channel when it has the settings reset to defaults.

The command is composed by the start address (0x00000000), followed by the instruction byte value 0x06, the Peripheral device address, new Peripheral channel, Peripheral key(56byte), and added padding.

4Byte	1Byte	Peripheral address	Channel(1byte)	Peripheral key	Padding till 128 Bytes
[0x00000000]	[0x06]	[0xNNNNNNNN]	[0xMM]	[0xKK ... 0xKK]	[0x00 ... 0x00]

Data commands

The Data command format starts de device address, followed by the command sub-address, the need for an acknowledged packet at protocol level, the length of the command data, the command data, the Peripheral key and finally, the needed padding to fullfil 128bytes total of command.
The command work type will be given in the sub-address.

Command format:
<0xNNNNNNNN><0xMMMM><0xPP><0xQQ><0xRR ... 0xRR><0xSS .. 0xSS><padding to 128bytes>

N – 4byte Peripheral device address. This value will always be bigger than 0 and smaller than 0xFFFFFFFF.
M – 2byte Command sub-address. This value, will always be smaller than 0xFFF0.
P – 1byte Command protocol ACK. This value is either 0 or 1.
Q – 1byte Command data length in number of bytes. The data length cannot be bigger than 56.
R – Variable length Command data. This array of bytes must match have a length that matches the Command data length value.
S – 56Bytes Peripheral key.
Padding – Padding with as many bytes of ignored data as needed to fullfil a total command length of 128bytes.

Again, each command sent will have a reply structure for command confirmation.
This is not the command reply, simply the confirmation that the command has been queued, by the Central, to be sent.

Peripheral IOs

There are **4** r/w IOs in the Peripheral device. These are pseudo-IOs, which are pulled up open-drain outputs. If working as inputs, the IOs need to be set high, leaving then pulled to 3.3v by a 5kohm resistor, and then read for the respective input. If working as outputs, the IOs can be either reset or 5kohm pulled-up set. Each IO cannot drow more than 5mA when in reset.

Peripheral IOs get (0x0000)

This command requests the current state of a given Peripheral IOs.

The command follows the command structure, but has no data to be sent. This way it has null length and no data.

4Byte	SubAddr	Ack	Length	Data	Peripheral key	Padding till 128 Bytes
[0xNNNNNNNN]	[0x0000]	[0xPP]	[0x00]	(no data)	[0xSS .. 0xSS]	[0x00 ... 0x00]

The command reply following the request, will follow the same format but will have 8 bytes of data with the pairs of the IO/value.

4Byte	SubAddr	Ack	Length	Data	Peripheral key	Padding till 128 Bytes
[0xNNNNNNNN]	[0x0000]	[0xPP]	[0x08]	[0x01 0xNN 0x02 0xNN 0x03 0xNN 0x04 0xNN]	[0xSS .. 0xSS]	[0x00 ... 0x00]

Where 0xNN is the respective IO state, with 2 possible values, '1' or '0'.

Peripheral IOs set (0x0001)

This command instructs the current output state of a given Peripheral IO, or set of IOs.
The length will have a maximum number of 5 pairs of values, indicating the IO and respective value.
This command will also provide the ability to set the automatic update mode in the Peripheral.
If active, the Automatic IO update will send a **Pripheral IOs get (0x0000)** message with the most up to date IOs values every time they change with a minimum of 500ms in between messages.

The IO name will be from '1' to '4', or '5' if (de)activating the auto update.
The setting value will either be '1' to set or '0' to reset.

The command follows the command structure and will always have the data in pairs of values, up to 5 pairs (10 values).

For example, if setting IO 1, reseting IO 3 and activationg auto update, the message data would be: [113051] with a length of [6]

4Byte	SubAddr	Ack	Length	Data	Peripheral key	Padding till 128 Bytes
[0xNNNNNNNN]	[0x0001]	[0xPP]	[0x06]	[0x01 0x01 0x03 0x00 0x05 0x01]	[0xSS .. 0xSS]	[0x00 ... 0x00]

Another example, if setting IO 2, the message data would be:
[21] with a length of [2]

4Byte	SubAddr	Ack	Length	Data	Peripheral key	Padding till 128 Bytes
[0xNNNNNNNN]	[0x0001]	[0xPP]	[0x02]	[0x02 0x01]	[0xSS .. 0xSS]	[0x00 ... 0x00]

Peripheral UART

There is one UART in the Peripheral that can be used to bridge data trough the Peripheral to the Central device. The UART has a default speed of 9600bps but can be set from 200 to 500kbaud.

The UART will always have no flow control, 1 start bit and 1 stop bit.

The maximum of 1KB of input buffer, after this the data will be overwritten and lost.

Peripheral UART setup (0x0002)

This command sets the UART speed and can also set the UART auto update.
When the UART auto update feature is enable, the UART will send any data accumulated in it's RX buffer automatically, thought a **Peripheral UART get (0x0004)** message.

This command data will be composed of either 4 or 5 bytes. The first 4 bytes are an unsigned 32 value representing the UART speed. The 5th byte, when sent, will represent the uart auto update featur enable/disable, with a value of either '1' or '0' respectively.

For example, setting uart to 115200, with auto update enabled:

4Byte	SubAddr	Ack	Length	Data	Peripheral key	Padding till 128 Bytes
[0xNNNNNNNN]	[0x0002]	[0xPP]	[0x05]	[0x00 0xC2 0x01 0x00 0x01]	[0xSS .. 0xSS]	[0x00 ... 0x00]

Simply setting the UART to 9600:

4Byte	SubAddr	Ack	Length	Data	Peripheral key	Padding till 128 Bytes
[0xNNNNNNNN]	[0x0002]	[0xPP]	[0x04]	[0x25 0x80 0x00 0x00]	[0xSS .. 0xSS]	[0x00 ... 0x00]

Peripheral UART send (0x0003)

This command requests the peripheral to send a given amount of data through the UART.
This command data will have from 0, up to 56 bytes of data to be sent.
For example, sending "Hello World!" through the uart :

4Byte	SubAddr	Ack	Length	Data	Peripheral key	Padding till 128 Bytes
[0xNNNNNNNN]	[0x0003]	[0xPP]	[0x0C]	["Hello World!"]	[0xSS .. 0xSS]	[0x00 ... 0x00]

Peripheral UART get (0x0004)

This command requests the peripheral to retrieve any data accumulated in the UART RX buffer. This command data will have a null length and no data.

For example, get any data in the uart receiving buffer:

```
      4Byte          SubAddr Ack   Length  Data   Peripheral key  Padding till 128 Bytes
[0xNNNNNNNN][0x0004][0xPP] [0x00](no data)[0xSS .. 0xSS]       [0x00 ... 0x00]
```

The Peripheral replies with (reply format as previously explained):

```
[0xAA]  [0x13]  [0xNNNNNNNN]     [0x04]        [0x0004]     ["Hello World!"][0x55]
<0xAA><Len>       <Orig addr>        <P. Type>  <Orig sub-addr>       <Data>      <0x55>
```

Peripheral I2C setup (0x0005)

The I2C port in the Peripheral device will always have 100kHz clock speed but may have the data format configured to either LSB or MSB (default).

This command requests the peripheral to set the I2C data as either MSB or LSB format.

For example, set the I2C data format as LSB:

```
      4Byte          SubAddr Ack   Length  Data   Peripheral key  Padding till 128 Bytes
[0xNNNNNNNN][0x0004][0xPP] [0x01] [0x00] [0xSS .. 0xSS]       [0x00 ... 0x00]
```

Peripheral I2C send (0x0006)

This command requests the peripheral to send a given amount of data through the I2C.
The data to be sent is composed by 1 + bytes, where the first byte is always the device address (depending on the device, may be device address and or command).
This byte will have the Read/Write bit forced by the instruction.

For example, send 5 bytes of data to device 0x05 through the I2C:

The address being 4, the byte value should be 0x0A(0x5<<1, being the first bit reserved for Write bit indication).

```
      4Byte          SubAddr Ack   Length                Data                      Peripheral key  Padding till 128 Bytes
[0xNNNNNNNN][0x0005][0xPP] [0x06] [0x0A 0x01 0x02 0x03 0x04 0x05] [0xSS .. 0xSS]       [0x00 ... 0x00]
```

In the I2C output, we will have:

```
Master:  [start][0x0A]       [0x01       0x02       0x03       0x04       0x05]       [stop]
Slave:                  [ack]       [ack]       [ack]       [ack]       [ack]       [ack]
```

Peripheral I2C get (0x0006)

This command requests the peripheral to read a given amount of data through the I2C.
The data to be read is composed by 1 + bytes, where the first byte is always the device address (depending on the device, may be device address and or command).
This byte will have the Read/Write bit forced by the instruction. (0b00000000 for write and 0b00000001 for read)

For example, read 5 bytes of data from device 0x05 through the I2C:

The address being 4, the byte value should be 0x0A(0x5<<1, being the first bit reserved for Read bit indication).

4Byte	SubAddr	Ack	Length	Data	Peripheral key	Padding till 128 Bytes
[0xNNNNNNNN]	[0x0006]	[0xPP]	[0x06]	[0x0A 0x00 0x00 0x00 0x00 0x00]	[0xSS .. 0xSS]	[0x00 ... 0x00]

In the I2C output, we will have: (0x0B = 0x0A|0x01 = (0x05<<1)|0x01 where 0x01 is a Read indication.)
Master: [start][0x0B] [ack] [ack] [ack] [ack] [nack][stop]
Slave: [ack][0x01 0x02 0x03 0x04 0x05]

Resetting the device to defaults

If the device needs to be reset to a default state, follow the steps:

1)Remove all power from the device.

2)In the I2C header, clace a jumper shorting SDA to GND.

3)Power up the device.

4)Remove the jumper while the device is powered.

5)Wait 1 or 2 seconds and power cycle the device.

After this, the device will now have the default access key and cleared any automatic behaviour as well as removed any handshaked Central device, effectively back to default shipping condition.

Notes

!!! IMPORTANTE! THIS DEVICE IS NOT A REPROGRAMABLE MODULE !!!
Attempt to reprogram any part of the device, will result in possible permanent damage and certain erase of all the internal code, leading to a permanent loss of the device functionality and therefore voiding the device support!

!!! IMPORTANTE! THIS DEVICE IS A PROTOTYPE, NOT A FINAL PRODUCT !!!
Although this product has been configured to follow Ofcom **LPD433** regulation for the lowe power devices 433 ISM band, it has not been approved nor certified.

The device shall ONLY be used for its intended purpuse.

References

https://www.silabs.com/documents/public/data-sheets/Si4430-31-32.pdf
https://www.ofcom.org.uk/__data/assets/pdf_file/0028/84970/ir-2030.pdf

Device pictures of interest