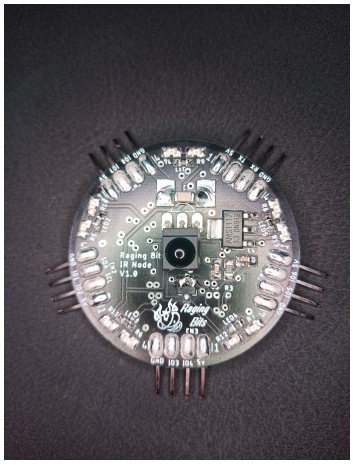
Raging Bits IR node network v1.0

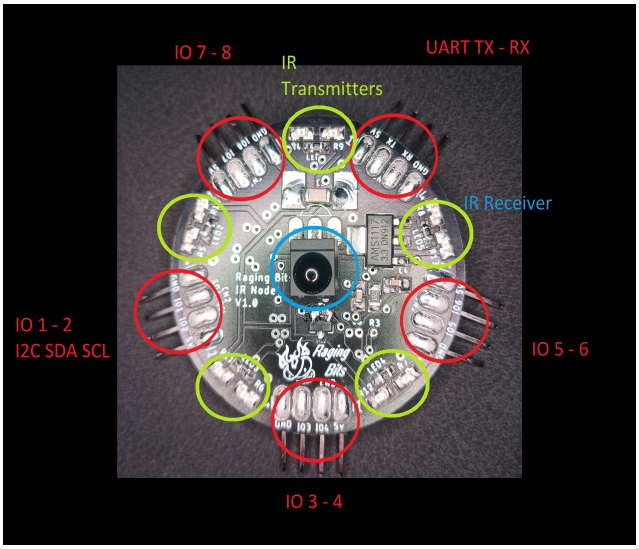


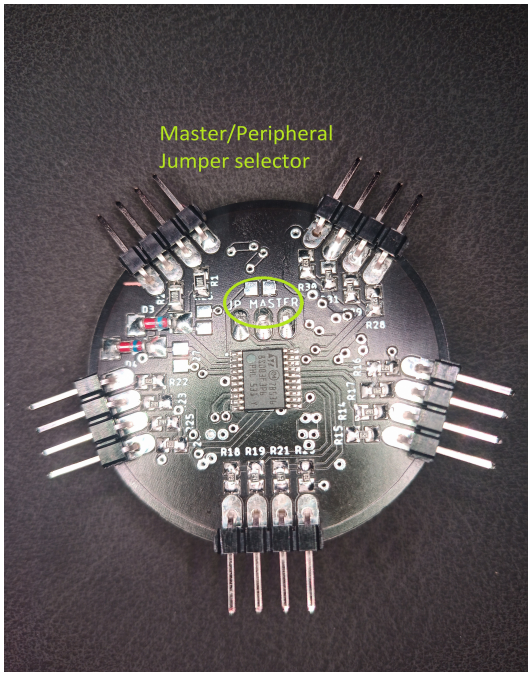
Top level specs

Uart
*I2C
*8 IOs
250bps of raw data
5V power supply.
Half duplex Star Network Topology
Every node can be configured as either Master or Peripheral
Master is always address 0x00 and Peripherals are settable from address 0x01 to 0xFE.

*(The I2C is an alternative bus that uses 2 IOs, leaving only 6 IOs available when I2C is used)

Device interface





IOs

The IOs are quasi-ios, meaning they work as both input and output without needing to change work type.

As output they set to GND with a 240ohm resistor, set to VCC with a 12kohm resistor pull-up.

To use IOs as inputs, they MUST to be set high.

Power

Voltage input is 5V.
RX all inputs high, consumption is approximately 8mA.
TX consumption is an average of 600mA.

UART

The UART speed for the master is 9600 bps fixed 1start bit, 1 stop bits and no parity.
The UART speed for the peripheral node is settable using command 0x01 with 1start bit, 1 stop bits and no parity.

I2C

Only the peripheral nodes can use the alternative I2C expansion BUS. Because this BUS is an alternative function, the PINs IO1 and IO2 of the respective port will now stop working as IO, instead working as SDA and SCL.

The BUS can be configured with MSB or LSB orientation. The speed is fixed at 100khz and the word size is 8bits+1 N/ACK bit.

Commands

The system is designed to have the commands synchronized by length of input data.
This way there will always be a synchronization reply by the driver to any input.
The Master-Peripheral interface can handle up to 10 messages with respective reply, per second.

(This speed depends directly of the type of command, as come are faster than others.)

The command composition:

[Node Address - 1B][Command+DataLength – 1B][Data - 0B to 6B]

Where Command+Data Length is a single composed byte, with the 5 LSBs being the command and the 3MSBs being the length.

The data length value represents half the data number of bytes, therefore the number of data bytes is 2x Data Length.

This way the command will have 1 Address, 1 Length+Command and 0, 2, 4 or 6 data bytes.

Set new node address - 0x00

Sets the device address to a new one. The command length is 1 that equals to 2 bytes of data.
The command data byte zero is the new device address and the data byte one is the inversion of the new device address.
[Device address][0x40|0x00][new address][~(new address)] (where 0x40 = (Len of 0x01) << 6)
This command has a fixed length of 2 bytes.
(where 0x40 = (Len of 0x01) << 6, therefore 2 bytes)

Ex: Query for device information, change device address from 1 to 2 and then query for device information.
>[0x01][0x0A]
<[0x00][0xCA][1][0][1][2]['R']['B']
>[0x01][0x40][0x02][0xFD]
<[0x00][0x40]['O']['K']
>[0x01][0x0A]
>(No answer. The device in now address 0x02)
>[0x02][0x0A]
<[0x00][0xCA][1][0][1][2]['R']['B']

Set node UART speed - 0x01

Sets the device UART speed. The command length is 2 that equals to 4 bytes of data.
Of these 4 bytes only the first 3 are used, so byte 4 is ignored.
The command data bytes will have the new UART speed from MSB to LSB.
The nodes UART default speed is 9600.
This command has a fixed length of 4 bytes.

[Device address][0x80|0x01 == 0x81][Speed MSB][Speed byte 2][Speed LSB][N/A]
(where 0x80 = (Len of 0x02) << 6, therefore 4 bytes)

Ex: Set node, with address 0x01, UART to 115200 bps (115200 = 0x01C200)
>[0x01][0x81][0x01][0xC2][0x00][0x00]
<[0x00][0x41]['O']['K']

Send data to node UART – 0x02

Sends a set of 2,4 or 6 bytes to the node UART at the configured speed.

This command has a variable length depending on the amount of data to be sent, of 2,4 or 6 bytes.

[Device address][0xN0|0x02 == 0xN2][data1]...[data(N*2)]
(where 0xN0 = (Len of 0x0N) << 6, therefore N*2 bytes)

Ex: Send 6 bytes to the UART of node address 0x03 (0xC = 0x03<<6)
>[0x03][0xC2]['H']['e']['l']['l']['o']['!']
<[0x00][0x42]['O']['K']

Read data from node UART – 0x03

Reads data from node UART. This data will have a length of 2,4 or 6 bytes.

The data read will always be repeated until it's acknowledged.

This command has a fixed length of zero bytes.

[Device address][0x00|0x03 == 0x03]
(where 0x00 = (Len of 0x00) << 6, therefore 0 bytes)

Ex: Reading data from UART of node address 0x02
>[0x02][0x03]
<[0x00][0x83]['H']['i']['!'][0x00]
(0x80 = 0x02<<6, therefore 4 bytes of data)

Read data with acknowledge, from node UART – 0x04

Acknowledges last UART read data, and requests reading for more if any. This new data will have a length of 2,4 or 6 bytes.

The data read will always be repeated until it's acknowledged. After this command, the previous data is discarded and fresh data is read, if any available.

This command has a fixed length of zero bytes.

[Device address][0x00|0x04 == 0x04]
(where 0x00 = (Len of 0x00) << 6, therefore 0 bytes)

Ex: Full reading data from UART of node address 0x02

>[0x02][0x03] (Read any data available)
<[0x00][0xC3]['H']['i'][' ']['t']['h']['e'] (0xC0 = 0x03<<6, therefore 6 bytes of data)
>[0x02][0x04] (Ack last and request more)
<[0x00][0xC4]['r']['3'][' ']['b']['u']['d'] (0xC0 = 0x03<<6, therefore 6 bytes of data)
>[0x02][0x04] (Ack last and request more)
(For this example, we're pretending that next packet never reaches master.)
<[0x00][0x84]['d']['y']['!'][0x00] (0x80 = 0x02<<6, therefore 4 bytes of data)
... (Come time later after other work...)
>[0x02][0x03] (Read any data available)
(The data that wasn't acknowledged is now repeated.)
<[0x00][0x83]['d']['y']['!'][0x00] (0x80 = 0x02<<6, therefore 4 bytes of data)
>[0x02][0x04] (Ack last and request more)
<[0x00][0x03] (0x00 = 0x00<<6, therefore 0 bytes of data)

(After the whole message received we have "Hi there buddy!" as the total data that was received at the node UART.)

Read node IOs – 0x05

Reads the IO values on the node.

To Note that IOs can only work as inputs IF they are set high!

The read data will have a length of 2 bytes.

The first byte represents the 8 IOs values, from IO1 being the LSBit of the value, and IO8 being the MSBit value.

The second byte not used.

This command has a fixed length of zero bytes.

[Device address][0x00|0x05 == 0x05]
(where 0x00 = (Len of 0x00) << 6, therefore 0 bytes)

Ex: Read the IO values of node address 0x03

>[0x03][0x05]
<[0x00][0x45][253][253] (0x40 = 0x01<<6, therefore 2 bytes of data)
(253 = 0b11111101, IO2 is reset, IO1,IO3 to IO8 are set.)

Set node IOs – 0x06

Sets the IO values on the node.

The first byte represents the 8 IOs values to be set, from IO1 being the LSBit of the value, and IO8 being the MSBit value.

The second byte represents the mask of the IOs allowed to be modified where the 8 IOs values allowed to be changed by byte one value, from IO1 being the LSBit of the value, and IO8 being the MSBit value.

This command has a fixed length of 2 bytes.

[Device address][0x04|0x06 == 0x46][New IOs Values][Permitted mask of IO modification]
(where 0x40 = (Len of 0x01) << 6, therefore 2 bytes)

Ex: Set the IO values of node address 0x03

>[0x03][0x05] (Read the IOs)
<[0x00][0x45][253][253] (0x40 = 0x01<<6, therefore 2 bytes of data)
(253 = 0b11111101, IO2 is reset, IO1,IO3 to IO8 are set.)
>[0x03][0x06][0x00][0x03] (Write the IOs with zeroes, but mask only allows IO1 and IO2)
<[0x00][0x46]['O']['K']
>[0x03][0x05] (Read the IOs)
<[0x00][0x45][252][252] (0x40 = 0x01<<6, therefore 2 bytes of data)
(252 = 0b11111100, IO1 and IO2 are reset, IO3 to IO8 are set.)

Setup I2C – 0x07

This command enables/disables I2C work as well as sets up I2C configuration.

This command has variable data length of either 0 bytes or 2 bytes.

If the command data length is 0 bytes, the I2C is disabled and the bus PINs returned to the primary IO functionality.

If the command has a data length 2 bytes, the I2C is enabled and the BUS pins assume the secondary functionalities of SDA and SCL.

When the command enables I2C, the the first byte represents the I2C data bitwise direction, being LSB if the data byte has a value of zero, and MSB otherwise. The second byte has no meaning.

Disable I2C:

[Device address][0x00|0x07 == 0x07]
(where 0x00 = (Len of 0x00) << 6, therefore 0 bytes)

Enable I2C:

[Device address][0x04|0x07 == 0x47][MSB(!0x00) or LSB(0x00) indication][N/A]
(where 0x40 = (Len of 0x01) << 6, therefore 2 bytes)

Ex: Enable and disable I2C of node address 0x03

>[0x03][0x47] [0x01][0x00](Enable I2C set as MSB)
<[0x00][0x46]['O']['K']

Send data to node I2C BUS – 0x08

This command sends data to the node I2C BUS.

This command has variable data length of either 2 bytes, 4 or 6 bytes.

This command can only be used successfully if the I2C has been previously enabled.

[Device address][0xN0|0x28== 0xN8][data1]...[data(N*2)]
(where 0xN0 = (Len of 0x0N) << 6, therefore N*2 bytes)

Ex: Enable, send data to and disable I2C of node address 0x03
>[0x03][0x47] [0x01][0x00](Enable I2C set as MSB)
<[0x00][0x47]['O']['K']
>[0x03][0x48] [0x23][0x01](Send I2C bytes 0x23 and 0x01)
<[0x00][0x48]['O']['K']
>[0x03][0x07] (Disable I2C)
<[0x00][0x47]['O']['K']
>[0x03][0x48] [0x23][0x01](Send I2C bytes 0x23 and 0x01)
<[0x00][0x88]['N']['O']['K'][0x00](Command refused as I2C is no longer active.)

Read data from node I2C BUS – 0x09

This command reads data from the node I2C BUS.

This command has variable data length of either 2 bytes, 4 or 6 bytes, where these represent the amount of data to be read. Only the first data byte of the command is sent out of the BUS.

This command can only be used successfully if the I2C has been previously enabled.

[Device address][0xN0|0x29== 0xN9][data][N/A]...[N/A(N*2)]
(where 0xN0 = (Len of 0x0N) << 6, therefore N*2 bytes)

Ex: Enable, read 2 bytes of data from the I2C of node address 0x03

>[0x03][0x49] [0x23][0x00](Send byte 1 to I2C and read 2 bytes from it)
<[0x00][0x49]['A']['B']('AB' are read bytes from the I2C BUS in the node.)

Read node information - 0x0A

This command reads the node information, that consists on 6 bytes.

This command has a fixed length of 2 bytes.

[Device address][0xN0|0x0A== 0x0A]
(where 0xN0 = (Len of 0x0N) << 6, therefore N*2 bytes)

The reply data will be the Hardware Version, the Firmware Version and the Manufacturer Info, which in this case the latest being 'R'(aging) 'B'(bits)

Ex: Request information from node address 0x03
>[0x03][0x0A]
<[0x00][0xCA][1][1][1][0]['R']['B']
(Hardware version: 1.1 Firmware version: 1.0 Manufacturer: 'RB')

Notes

The UART interface is done with 3.3v signals.
The TX pin will only be able to output 3.3v. The RX pin is capable of 5v input signals.

The refresh rate depends on a combination of 3 factors:
Frame length.
UART speed.
Device type of command. (Information reading is faster than IO access and I2C interface, for example.)

As a reference:
IO access can deal with up to 9 replied messages per second.

All devices come from factory with Address 0x01. The user shall updated it according to need.