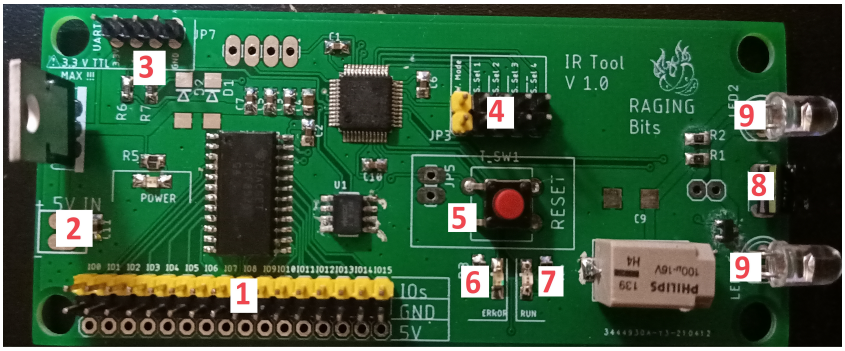
Raging Bits IR Tool v1.0

Top level specs

64MHz 32bit arm core.
3.3v level protected UART lines.
4KB UART input buffer.
Dual work mode, tool/transparent serial port.
Multi UART speed selection.
50m distance HALF-DUPLEX IR Blaster/Receiver.
5v +/- 0.25V 500mA max.
IR time precision +/- 100uS. (Directly depends on the demodulator Specs.)

Device interface



1 – IOs pins. Consists in 16 sets of IO pin, GND and 5V power, organized from 0 to 15.
2 – 5v power input.
3 – UART port. Organized as 3.3v, TX, RX and GND, from left to right respectively.
4 – UART speed and work mode selections. See jumper configuration for further information.
5 – Reset button.
6 – Error indication LED. Fast blinking. (Indicates a problem with EEPROM, IOs or internal Flash.)
7 – Device running LED. (This LED is a simple heart beat that will vary with device load so it properly reflects the work load. A fixed behaviour will show that the device is waiting for the user to finalize a command input. Usually happens when the command is incomplete, or in same cases of wrong command syntax).
8 – IR receiver demodulator. (940nm)
9 – IR emitting LEDs. (940nm)

Time interpretation

The times are always interpretated/sent in alternations of carrier ON and OFF respectively, being the first time always carrier ON, first time is sent with carrier active, the second with carrier inactive, and so on.
Timeout is meant as the last time used with carrier inactive, in this example, the carrier off time after T5.

Power

Power consumption in idle is approximately 40mA.
Power consumption during full load IR transmission is approximately 450mA.
Power supply can be done through 3.3v UART interface (pin one) OR through 5v input connector.

!!! IMPORTANTE! NEVER POWER BOTH SIMULTANEOUSLY !!!
This could damage the device!
Either provide power through UART OR through 5v input connector.

Inputs and Outputs (IOs)

Multiple IOs can be set as outputs triggered by the same data set.
This is not valid for multiple IOs as inputs, they still can be configured for the same data set, but only one inpit change is handle per turn. If multiple IOs change simultaneously, only the highest id value of them gets attended to due to the simple fact that only one set of data can be send at a time.

For protection of the system and easy user access, the IOs are in fact quasi-IOs.
This means they can act as inputs and outputs, only the outputs can go be of high impedance when high and open drain when low.
When configured as output and set high, will only be able to supply an output positive voltage over a 4mA current limited path. When an output is set low, it will be able to drive up to 50mA as an open drain. This is why an IO configured as output CAN NOT be connected to any current source directly!
As mentioned for easy user access and as an example, a switch can be connect to GND and an IO set as input, in order to read user key inputs. An LED with the respective current limiting resistor can be connected from the 5V to the IO set as output so it can be used to visual aid.

Jumper configuration

Multiple UART speeds can be configured at boot time, using the jumper setup.

The device has a secundary simple transparent bridge work mode. This mode allows multiple

devices communicating in the same space at speed that are approximatly 250Bps in half duplex mode with the same distance capability as in tool mode. Specially useful for noisy radiation areas where simple low speed communications are needed.

By default (no jumpers connected) the jumper setting is Tool mode with 115200 baudrate for the UART speed.

Work mode

Jumper on – Transparent binary bridge mode.
Jumper off – IR Tool mode.

UART speed

S_Sel(n) configure the UART boot speed.

Speed (bps)	SEL 0	SEL 1	SEL 2	SEL 3
300	on	on	on	on
600	off	on	on	on
1200	on	off	on	on
4800	off	off	on	on
7200	on	on	off	on
9600	off	on	off	on
14400	on	off	off	on
19200	off	off	off	on
38400	on	on	on	off
57600	off	on	on	off
15200	on off ... off	off off ... off	on on ... off	off off ... off

Internal memory

All the programable information will be residing in an EPROM capable of over 100K write cycles per solt.
A maximum of 127 slots are available, numbered from 0 to 126, for the uer to save data to.
It can only hold RAW times. Binary data is not allowed.

Sending/Receiving RAW times

The maximum amount of raw data that can be sent/received/saved is 250 time units.
Being sent/received, this means before a timeout event happens.

Sending binary data

In tool mode, the maximum amount of binary data that can be sent per request is 500 bytes.
In bridge mode, there is no limit, although the device has a maximum of 4KB of UART input buffer. This buffer never overflows, as it always overrides the older data to store new one.

Times and timeouts

The maximum time at work must be the timeout time.
The maximum time the device supports is 60000us.
There are no lower limits for times imposed but it is STRONGLY ADVISABLE to use a minimum time of 500us with the original receiver demodulator unit.
The maximum frequency supported while sending is 40KHz, and the minimum 30KHz.
The maximum frequency supported by the original receiver demodulator is 34KHz and the maximum 38KHz. (The receiver would be able to work from 32KHz to 40KHz but already outside of specs where functionality cannot be properly garanteed.)

Commands

The system is designed to have the commands syncronized using Line Feed character termination or Carriage Return followed by Line Feed ternination. The only exception is the Serial Send command which is terminated as soon as the length of data requested is transmited.
This is in place to help the user have commands re-synchronized as fast as possible in a case of the input buffer overflow.
A receive command will be temporarely paused by a send command.
A send command cannot be paused or stoped by any other send or received, only by a stop command. The stop command will stop any current device task.
If a command is already being sent, another send will wait until the previous one has finished.
An erroneous command will almost always be detected, but like an incomplete command, a wrong command syntax may lock the device until a command timeout error happens. In this case, the Run LED indicator show a "fix" light, instead a slow blinking behaviour.

AT+CHECK\r\n

Returns the current state of the device.
IDLE – If the device is not performing any work.
SENDING SERIAL – The device is sending IR data in binary encoding.
GETTING SERIAL – The device is cycling waiting/receiving any binary encoded data.
SENDING RAW - The device is sending IR data in time encoding.

GETTING RAW– The device is cycling waiting/receiving any time encoded data.

Ex:
>AT+INFO
<CHECK:IDLE
<OK

AT+INFO\r\n

Returns the current information about the device.
The return fomat is as follow:
INFO:xxxxxxxxxx,aa.bbb.ccc,dd.eee.fff
x – The device serial number.
aa.bbb.ccc – Firmware version number; major, minor, iteration.
dd.eee.fff – Hardware version number; major, minor, iteration.

Ex:
>AT+INFO
<INFO:0000000001,00.000.001,00.000.001
<OK

AT+IR_RAW_SEND:[CARRIER_FREQUENCY],[TIMEOUT], [NUMBER_TIMES],[DATA1],...[DATAn]\r\n

Command to send an IR trail with the given times, at a given carrier frequency.
CARRIER_FREQUENCY – Frequency of the carrier to be used.
TIMEOUT – The time of null carrier after the trail is sent.
NUMBER_TIMES – Length of the trail.
DATA – The list of times modulating the IR data.

Ex:
Send a trail of times with a 38Khz carrier frequency and a final time of 10ms.
8.5us Carrier active, 2100us carrier inactive, 550us carrier active and 10000us carrier inactive for the end of the transmission.
>AT+IR_RAW_SEND:38000,10000,3,8500,2100,550
<OK

AT+IR_RAW_GET:[CARRIER_FREQUENCY],[TIMEOUT]\r\n

Sets the device in raw data times receiving mode.
This mode can receive up to 256 times in a row before a timeout happens.
After a timeout happens, the array of times is printed as follow:
T:a,b,c,d,e,f...z

Once entered, this mode maintains active after any raw/serial data send command is requested meanwhile.

Ex:
>AT+IR_RAW_GET:38000,10000
<OK
(Time passes until a NEC protocol command is received. Being the large bulk of times the command, the small one is the command instruction for key oressed repetition.)

NEC Command received.
<T:8602,4243,525,550,525,525,550,521,529,546,529,521,554,521,525,550,524,525,551,
1587,529,1587,525,1592,550,1592,525,1591,525,1567,575,1592,525,1592,525,550,525,
1591,525,1567,575,525,525,1591,551,524,526,550,524,521,555,1591,525,525,550,525,
525,1567,575,525,526,1566,576,1566,550,1567,550

NEC Repetition command of previous NEC command.
<T:8588,2110,550

AT+IR_SERIAL_GET\r\n

Sets the device to receive IR binary data, such as a transparent serial communication. In this mode the IOs are disabled, maintaining only pure IO values, without any action taken.
Once entered, this mode maintains active after any raw/serial data send command is requested meanwhile.

Ex:
Device set to receive binary data. Another device transmites a binary data array with the value "Hello World!".
>AT+IR_SERIAL_GET
<OK
<Hello World!

AT+IR_SERIAL_SEND:[DATA_LENGTH],[DATA1]...[DATAn]

To notice that this command has no "Carriage return" nor "Line feed" termination.
This is because as soon as the length is set, the data is saved as it is sent, byte by byte.
During the data collection and until the totallity of data length indicated is collected, any data is accepted without exception.
DATA_LENGTH – Length in bytes of the data to be sent. Maximum value of 500 bytes.
DATA – Bytes of data to be sent.

Ex:
Send an array of bytes that contains the string "Hello back!".
>AT+IR_SERIAL_SEND:11,Hello back!
<OK

AT+STOP\r\n

Stops any current activity of the device, returning it to idle.
If the device was performing any work, after this commad, it will be stoped.

Ex:
>AT+CHECK
<CHECK:GETTING RAW
<OK

>AT+STOP
<OK

>AT+CHECK
<CHECK:IDLE
<OK

AT+MEM_SET:[SLOT_ID],[CARRIER_FREQUENCY],[TIMEOUT], [RAW_DATA_LENGTH],[RAW_DATA]\r\n

Command to save raw data to an EPROM slot.

Ex:
>AT+MEM_SET:1,38000,10000,3,8500,2100,550
<OK

AT+MEM_GET:[SLOT_ID]\r\n

Returns the data contained in a given EPROM data slot.
The reply consists as follow:
MEM_GET:[CARRIER_FREQUENCY],[TIMEOUT],[DATA1],...,[DATAn]
In case the slot is empty, it simply states it.
MEM_GET:EMPTY

Ex:
>AT+MEM_GET:1
<MEM_GET:38000,10000,8500,2100,550
<OK

>AT+MEM_GET:0
<MEM_GET:EMPTY
<OK

AT+MEM_DEL:[ID]\r\n

Deletes a memory slot data.

Running a AT+MEM_GET after this command will return EMPTY in case the slot erased had any data on it.

Ex:
>AT+MEM_DEL:0
<OK

>AT+MEM_GET:0
<AT+MEM_GET:EMPTY
<OK

AT+MEM_SEND:[ID]\r\n

Sends the data of a memory slot if valid.

Ex:
Sned the slot data.
>AT+MEM_SEND:0
<OK
Delete the slot.
>AT+MEM_DEL:0
<OK
Try again and fails.
<AT+MEM_SEND:0
>AT+MEM_SEND:EMPTY
<OK

AT+IO_CONF:[PIN_ID],[INPUT/OUTPUT],[MEM_SLOT],[PIN_VALUE], [EDGE/PULSE_LENGTH]\r\n

IO pin configuration command.
It can configure a pin to act as input or ouput, as well as associate it with an EPROM memory data action. The Ios work and respective actions are only active whyle the system is idle or receiving raw data. In case the device is set to send/receive IR binary data, the IOs are inactive.

PIN_ID – IO Pin number. 0 to 15.
INPUT/ OUTPUT – Value 0 sets the pin as input. Value 1, as ouput.
MEM_SLOT – Memory slot number to be linked to the pin action. Value 0 to 127 means a valid memory location. Anything above is invalid, and avoids a link to be created.
PIN_VALUE – If the pin is configured as output, this sets the normal value of the pin, 1 as set value and 0 as reset value. The pulse on action is the inverted normal status.
If the pin is configured as input, this has no meaning.
EDGE/PULSE_LENGTH – If the pin is set as input and a valid memory location is linked, this indicates the edge of action for the data to be sent out.
If the pin is set as output and a valid memory location is linked, this indicates the length of the action pulse for the valid data received.

If the IO is configured as input and associated with a valid EPROM slot data, when the input changes, the data from the memory slot is sent through the IR blaster, as when the input crosses the configured edge. If this edge is set to 1, the crossing of pin set to reset will trigger a data send.
If the edge is set to 2, the crossing of pin reset to set will trigger a data send.
If the edge is set to 3, the crossing of either edge will trigger a data send.
If the edge is set to zero, then no memory data linked is sent.

If the IO is configured as output and associated with a valid EPROM slot data, then the pin will pulse for a given time when the data associated with the respetive configured memory slot, is received. The pulse time, depends on the edge configuration value. An edge value of zero equates to a pulse of 25ms, edge value of 1 to 50ms, edge value of 2 to 75ms and edge value of 3 to 100ms.

The pulse value should be configured smaller than to the respective protocol timeout in use, such that no IR data is lost while the pulse is active.

If the slot id value is above the maximum number of slots, the io configuration will only change the pin value and respective information. No memory link action is set.
If the slot id has a valid value, then the slot MUST me set with correct valid data.

Ex:
Configuring io 14 as output only without any memory linked, set to value 1.
>AT+IO_CONF:14,1,128,1,0
<OK

Configuring io 14 as output with memory slot number 2 linked, set to value 0 and an action pulse of 50ms on receiving data match.
>AT+IO_CONF:14,1,2,0,1
<OK

Configuring io 12 as input with memory slot number 2 linked, set to send the data when the pin value only goes from set to reset.
>AT+IO_CONF:12,0,2,0,1
<OK

AT+CONF_SAVE\r\n

Saves the current device configuration, Ios setup including respective memory link actions and current device work mode.
This configuration will be loaded EVERY time the device restarts in Tool mode.

Ex:
>AT+CONF_SAVE
<OK

AT+CONF_LOAD\r\n

Loads the device saved configuration in case a valid one has been previously saved and not deleted, overwriting the current setup of the device.
If no valid configuration exists, nothing is done.

Ex:
>AT+CONF_LOAD
<OK

AT+CONF_DEL\r\n

Deletes any previously saved configuration. The nixt time the device boots up, the default configuration is loaded.

Ex:

>AT+CONF_DEL
<OK

Notes

The IR blaster is capable of 750mA/LED current pulses, taking the maximum ability of the LED transmitting power. The maximum work duty cycle will at a worst case reach a 20%. During long time transmissions, the IR drive will warm up as the LED current limiting resistences take just over 1W hit during the pulse.

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 UART input buffer system is protected from overflow at a cost of the loss of the older data, just as a FIFO circular buffer.