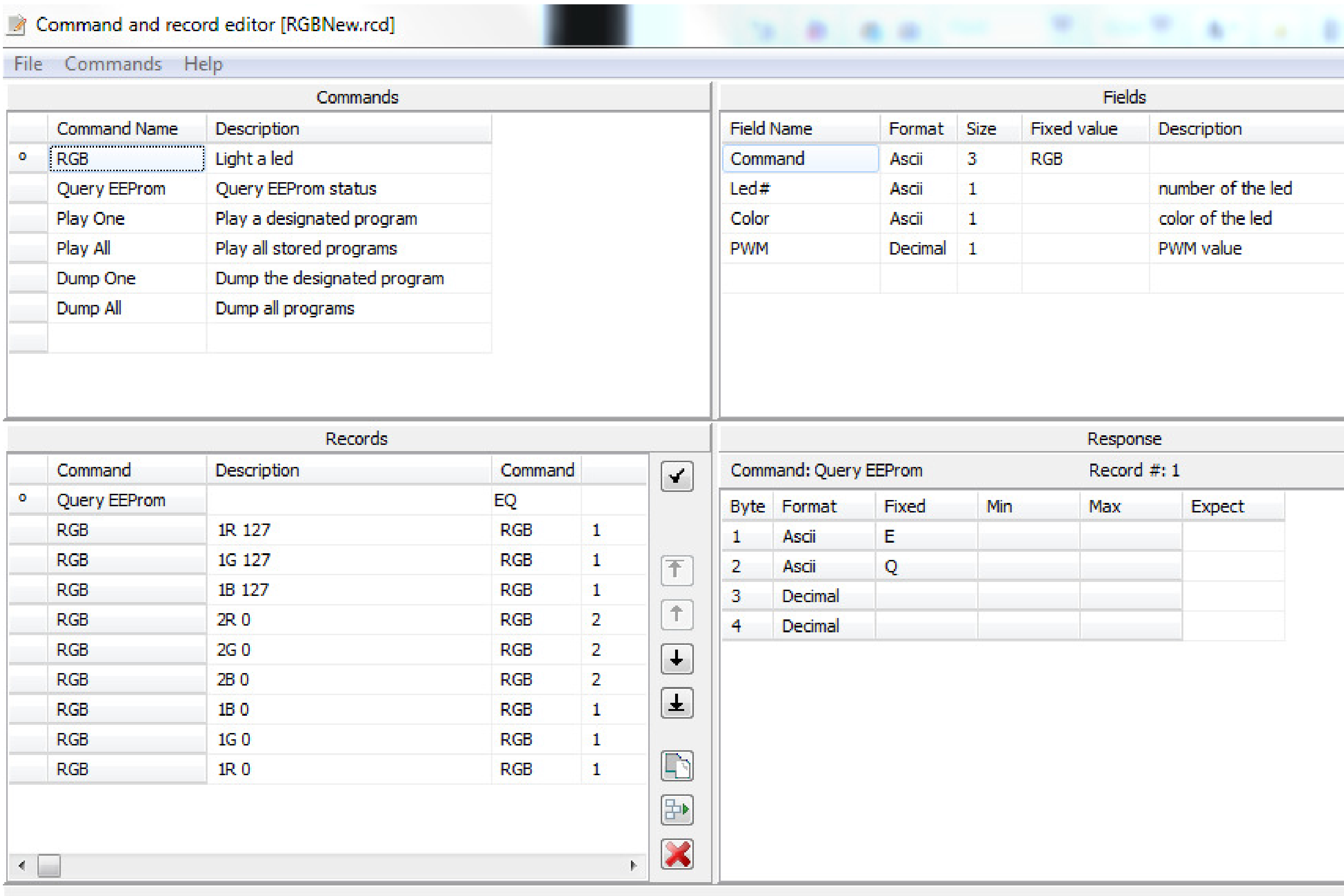
Command and Record editor.

This tool is aimed at the HID terminal/monitor applications of John Barrat and myself.
It's purpose is to facilitate the creation of files to be loaded for sending data to a USB device (Tx files) and subsequently
test the response of the device.
The output generated follows the conventions of these applications, this means:
- hexadecimal, numerical and binary input will be enclosed by brackets
- numerical input will generate a leading length modifier (#, W, D).
- binary input will generate a leading 'B'
- hexadecimal input will generate a leading '$'

examples: [#12], [W1024], [D65537], [B11001111], [$0D0A]
There are more examples at the end of this document.

Here's what it looks like.



Commands

	Command Name	Description
o	RGB	Light a led
	Query EEProm	Query EEProm status
	Play One	Play a designated program
	Play All	Play all stored programs
	Dump One	Dump the designated program
	Dump All	Dump all programs

Fields

Field Name	Format	Size	Fixed value	Description
Command	Ascii	3	RGB	
Led#	Ascii	1		number of the led
Color	Ascii	1		color of the led
PWM	Decimal	1		PWM value

Records

	Command	Description	Command	
o	Query EEProm		EQ	
	RGB	1R 127	RGB	1
	RGB	1G 127	RGB	1
	RGB	1B 127	RGB	1
	RGB	2R 0	RGB	2
	RGB	2G 0	RGB	2
	RGB	2B 0	RGB	2
	RGB	1B 0	RGB	1
	RGB	1G 0	RGB	1
	RGB	1R 0	RGB	1

Response

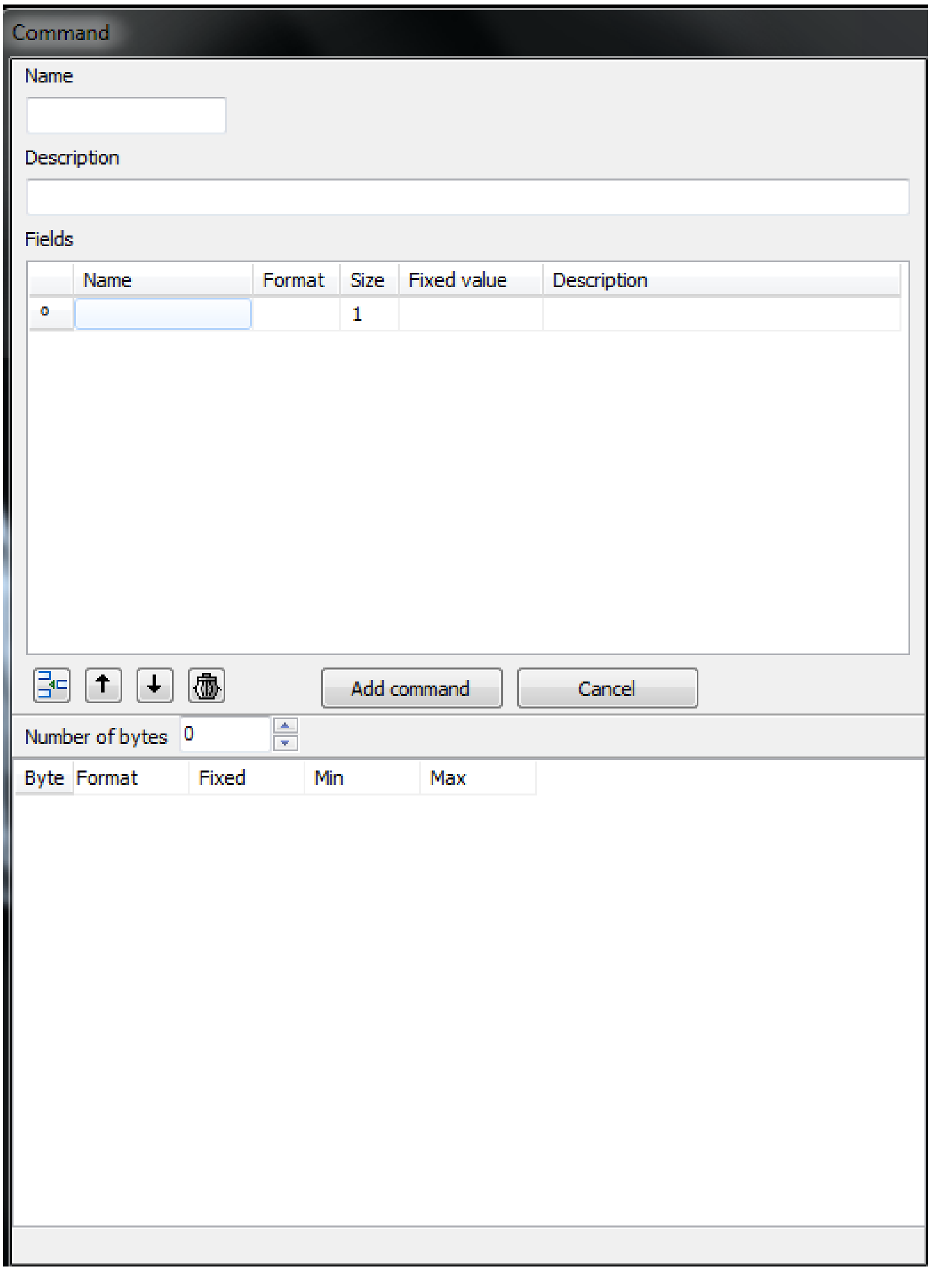
Command: Query EEProm Record #: 1

Byte	Format	Fixed	Min	Max	Expect
1	Ascii	E			
2	Ascii	Q			
3	Decimal				
4	Decimal				

The upper left hand panel holds commands. Commands can be compared to functions.
The upper right hand panel holds fields, these are the parameters for the selected command.

Creating a command.

Select Command>New command.
A window comes up allowing you to enter:
- The name of the command
- A description of the command, this is optional.
- The fields.

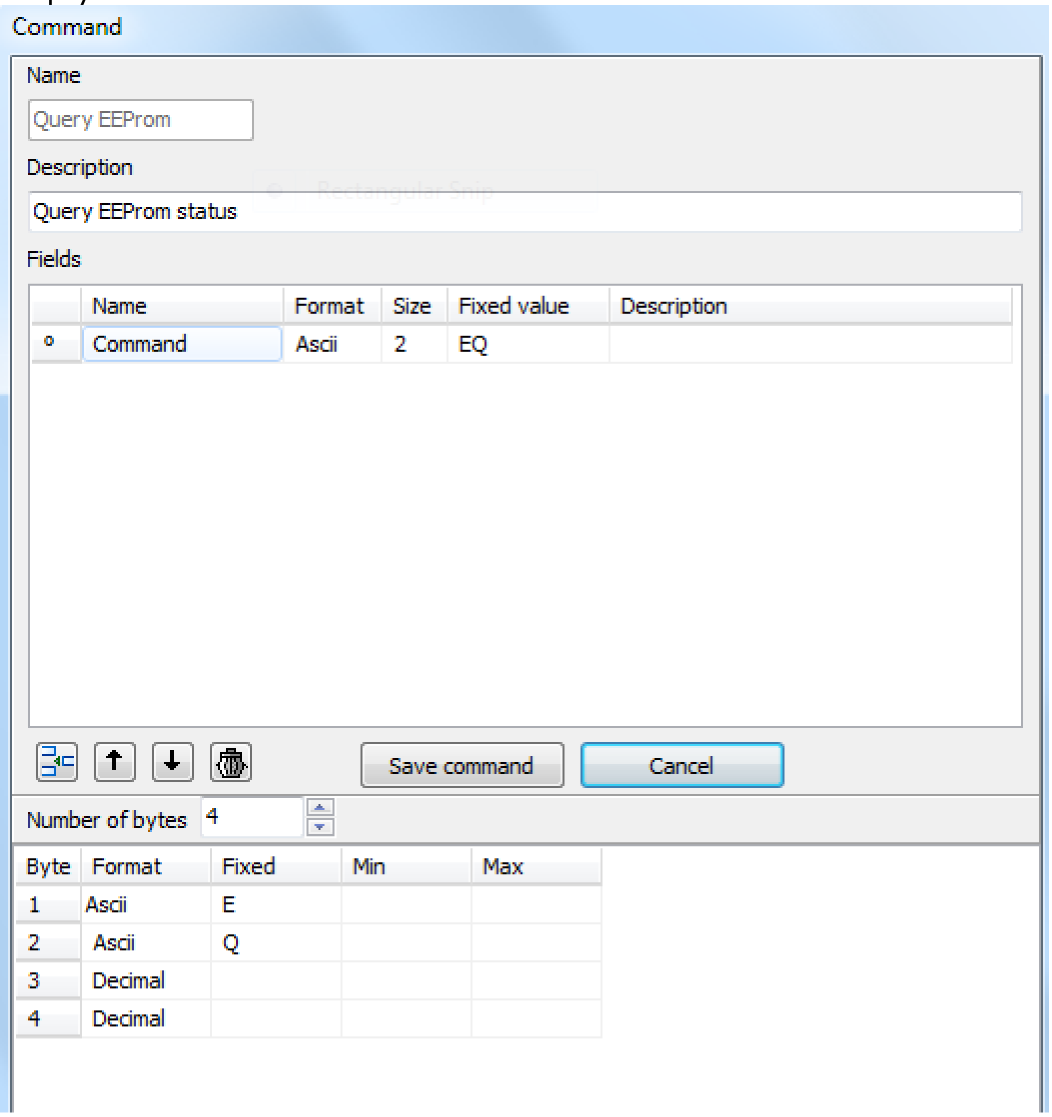


Example 1: I have a USB device containing 3 RGB led's. To light up one of those I have the following command:
RGB1Cxxx. Where RGB is the command, 1 is the led, C the color (R, G or B) and xxx the PWM value (0..255).
The name of the command would be: RGB
The description: Set the RGB values for a specific LED and color
Example 2: On this device, an EEProm stores programs to be played. There's a command that plays all programs

stored.
Name: Play All
Description: too obvious

Each field has:
- a name
- a format, you can choose from a number of formats. The format deals with the way you supply input and how that will
be translated to output. See below.
- a size. For decimal values size is 1, 2 or 4. The size determines the number of bytes used in the output. Minimum is 1.
- an optional fixed value. This will override any value you try to supply in the records.
- a description, this is optional

Pressing enter in the description column will check the values and create a new empty field.



Name: Query EEProm

Description: Query EEProm status

Fields

	Name	Format	Size	Fixed value	Description
○	Command	Ascii	2	EQ	

Number of bytes: 4

Byte	Format	Fixed	Min	Max
1	Ascii	E		
2	Ascii	Q		
3	Decimal			
4	Decimal			

On Format and Size.

Format: determines the way you supply data in a record and the way output is generated.

Available formats are Ascii, Binary, Decimal and Hexadecimal. Just type an A, B, D or H.

Size: the number of corresponding bytes in the output.

Ascii: you just type in the value, no conversion what so ever. The string entered will be stored in the output. If it is
shorter than the size parameter, it will be right padded with spaces.
Hexadecimal: a hex value may be entered as xx, multi byte is ok. If the number of characters (excluding the $) is odd,
a '0' will be inserted as the first character. '$' as first character is allowed, but not necessary.
If the output size is larger than the number of actual bytes from the input it will be right padded with [$00].
Decimal: on generation, the numerical value you type in will be enclosed in brackets.
Size determines the number of bytes in the output and for decimal values this may be 1, 2 or 4 bytes.
If size is 1 the number will be preceeded by a '#', if it is 2 it will be preceeded by a 'W' and for 4 it will be preceeded by
a 'D'.
Binary: a binary value may be entered as 001001. A binary field is always size 1. If the number of 0's and 1' is not a
multiple of 8, '0's will be inserted at the left. '%', 'b', 'B' as first character are allowed, but not necessary.

Back to example 1:
I need 4 fields:
Field 1 : name = Command, Format = Ascii, Size = 3, Fixed value = RGB
Field 2 : name = Led, Format = Ascii, Size = 1
Field 3 : name = Color, Format = Ascii, Size = 1
Field 4 : name = PWM, Format = Decimal, Size = 1
Of course it is valid for any field to use Binary or Ascii, the only difference is the way you enter data in a record and the
output of the generator.
Example 2: 1 Field : name = Command, Format = Ascii, Size = 2, Fixed value = EQ

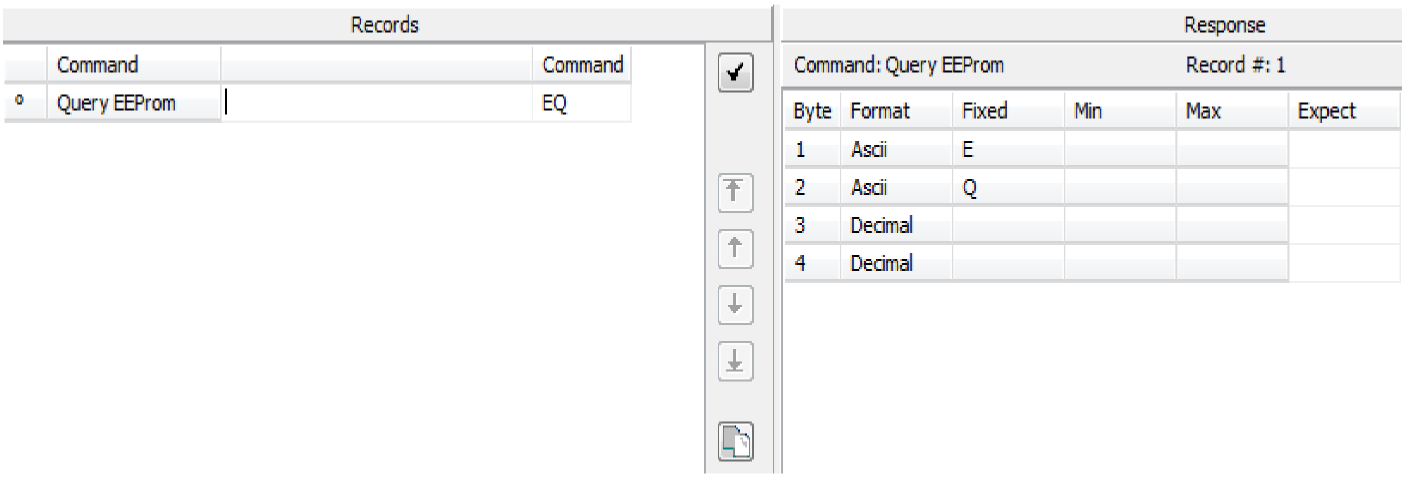
When you're done adding fields to a command, press 'Add command'.

You'll see the newly created command in the two upper panels.
On the left side you'll see Name, Description and Command.
On the right side you'll see a list of the fields and their attributes.

You can create as many commands as you like and up to 64 fields per command. If the total size of fields exceeds 64
you'll get a warning.

Creating a record.

Double click the command (left upper) that you want to create a record for.



Records

	Command		Command
○	Query EEProm		EQ

Response

Command: Query EEProm Record #: 1

Byte	Format	Fixed	Min	Max	Expect
1	Ascii	E			
2	Ascii	Q			
3	Decimal				
4	Decimal				

The lower panel now will show you the following:
- The name of the command
- A field for a description, this is optional. It is used in the HID monitors to clarify the record
- A number of columns for the fields for this command

You now can go through the fields and enter the data you want.
Please check you're entering data that complies with the format of the field.

Note there's a field that's new: 'Expect'.
This field only has a meaning if neither 'Fixed', 'Min' or 'Max' have been supplied from the Command.
Now why would you have an Expected value when you don't have Fixed, Min and/or Max?
The reason is this:
Suppose your device has an internal counter and there are commands to change that counter.

Command: Reset counter, no response expected
Command: Increment counter, no response expected
Command: Query counter: one byte expected.

The records that you want to send to the device:

Reset counter
Query counter, response 0
Increment counter
Query counter, response 1

A fixed value wouldn't help you here, and neither would Min and Max.

Here are some guidelines:
- if you specifiy Ascii as format and 12 as size, the generated field will be 12 bytes long even if you entered only 2
- same for Hexadecimal
- Numerical values should fit the size
- Negative decimal values are converted to two's complement, i.e. '-12' becomes '244'

When you click 'Add record', or 'Save record' in editing mode, all fields are checked.
Checking stops at the first error encountered, a message will be shown in the bottom part of the window.
The buttons on the right let you manipulate the records.

Back to example 1:
I want to set Led1 to All Red, Led2 to all Green and Led3 to all Blue.
So I need three records:
Record 1: Led = 1, Color = R, PWM = 255
Record 2: Led = 2, Color = G, PWM = 255
Record 3: Led = 3, Color = B, PWM = 255
Example 2:
I only need 1 record for this and there are no fields to fill in since there is a field with a fixed value

Creating records from different commands will appear a bit strange at the first time, since the column headers for the
fields contain the names of the fields of that specific command.
Suppose you have a number of commands, each with a number of fields, not necessarily the same number.
On double clicking the command this is entered in the first column of the record, column 3 and onwards contain the
names of the fields.
Suppose you entered data in the fields and then decide you need another record for a different command.
When you double click that command, the column headers are replaced with the names of that fields of the new
command.
The number of columns visible is that of the command that has the most fields.
The content of the column headers is only there to help you, it has no effect on values entered previously.

Commands revisited.

Suppose you created a command and used it to create some records. But when you think again, there's a field missing
or it has the wrong format.
Now you can edit a command easily, so that's no problem. But the records created from the previous definition will not
necessarily contain all/the correct values.
Be very sure that records created earlier match the current definition before generating a file, since generating a file will
use the latest definition. You'll get a warning when you edit a command that already has been used.

Saving your work.

When you're creating Commands, a .cnd file will be created.
This file will contain the command definitions and the command responses.
When you're creating Records, a .rcd file will be created.
This file contains:
- the command definitions
- the command responses
- the record definitions
- the record responses
- the record response string
- the record descriptions

- the record strings

A word of warning: don't try to edit these files. You might corrupt them and then they may crash the tools that use
them.

Command definitions:

Each line holds a command:
- the name
- for all fields:
-- the name
-- the format
-- the size
-- the fixed value
-- the description
All these are separated by a $FF character.

Command responses:

These are the bytes to be expected from the device.
Each line holds a command:
- for all bytes
-- the format
-- the fixed value
-- the minimum value
-- the maximum value
All these are separated by a $FF character.

Record definitions:

Each line holds a record:
- the name of the command
- the values for all fields
All these are separated by a $FF character.

Record responses:

Each line holds the response for a record:
- the name of the command
- the values for all the fields
All these are separated by a $FF character.

Record response strings:

These lines hold translations of the record responses so they are easier to check.
The lines are the same as the record responses but all bytes are now represented by their decimal representation.
All these are separated by a $FF character.

Record response descriptions:

These lines hold the textual description of the records.
They consist of the description of the command, a semicolon and the description of the record.
These lines are used by the two USB HID tools.

Record strings:

These lines hold the conversion of each record to a string to be sent to the device.

Saving and loading Command and Record files.

Both types of files can be saved and loaded using the menu options.
However, there's a 'feature'.
Since the record files are built given a number of commands available at that time, the record files will always contain the command definitions.
This also means that when loading records, the current commands are replaced by the commands in the records file.

Loading files into the USB HID tools:

Note there's a slight difference in terminology between John's and my tool.
John's USB HID tool has a menu option 'Load Command/Response Records', the default extension is '.rcd' and this is actually what I call a Record file.
My USB HID tool has a menu option 'Load record file', which essentially does exactly the same.

Examples on Format and Size.

Fields

Name	Format	Size	Fixed value
Nr1	Ascii	12	
Nr2	Hexadecimal	6	
Nr3	Binary	1	
Nr4	Decimal	4	
Nr5	Ascii	2	A

Records
Basic example

```
#1: Monkey  0D0A    110011  124 <empty>
```

Output generated

```
Monkey      [$0D][$0A][$00][$00][$00][$00][B00110011][D124]A
```

Shows an attempt to override field nr5 which has a fixed value. It fails.

```
#2: Elephant  A     1  65588    ZZ
```

Output generated

```
Elephant    [$0A][$00][$00][$00][$00][$00][B00000001][D65588]A
```