192-CH-DMX-Command-Interpreter

What is it?

Inexpensive lighting mixer consoles that can individually control up to 192 DMX channels of a lighting system (e.g. brightness or colour values in a range between 0 and 255) are often used to control small spotlight and lighting systems. The control is divided into 12 x 16 channels. This division of the channels into 12 groups, which can be enabled via keys individually or together for adjustment by sliders, is favourable when the fixtures can be controlled by 16 channels each (possibly also multiples of 16).

The values for the DMX channels that are set via the sliders can then be stored in scenes, each of which corresponds to a single state of all the required channels of the lighting system. These scenes can be combined as sequences, which can then also be stored and recalled with the lighting console.

Lighting systems for illuminating small rooms or model installations usually have coloured LEDs. RGB-LEDs, which are connected in series and controlled via one or a few signal lines, are very flexible. A corresponding designation of such LEDs is "NEOPIXEL", a frequently used type is WS2812.

To control such RGB-LEDs (e.g. in a strip or ring of LEDs), for each LED three channels are needed for the brightness of the colours red, green and blue. With the 192 DMX channels of a lighting control console, 64 LEDs can thus be controlled; with the 512 channels of a DMX universe, 170 LEDs can be controlled.

The division into 12 x 16 channels that can be switched to the control system is not favourable here, as 5 LEDs can be controlled with the first channel group and the 16th channel controls the red light brightness of the 6th LED. Due to the successive control of the LEDs, one must set this 16th channel of the first group, then switch to the next group and control the remaining two channels of the 6th LED there.

Therefore, depending on the lighting system to be controlled, it is easier to use another way of setting the values of the DMX channels.

With the help of the ESP32 microcontroller, an interpreter for a simple control language was realised, with the help of which the channel values of scenes and sequences of scenes can be set. Operation is via a web interface, for which the corresponding server is also provided by the ESP32. This allows a command sequence to be entered and edited. If this command sequence is then executed, the ESP32 generates DMX packets with 192 channels, which are used by wired DMX receivers of a lighting system for lighting control.

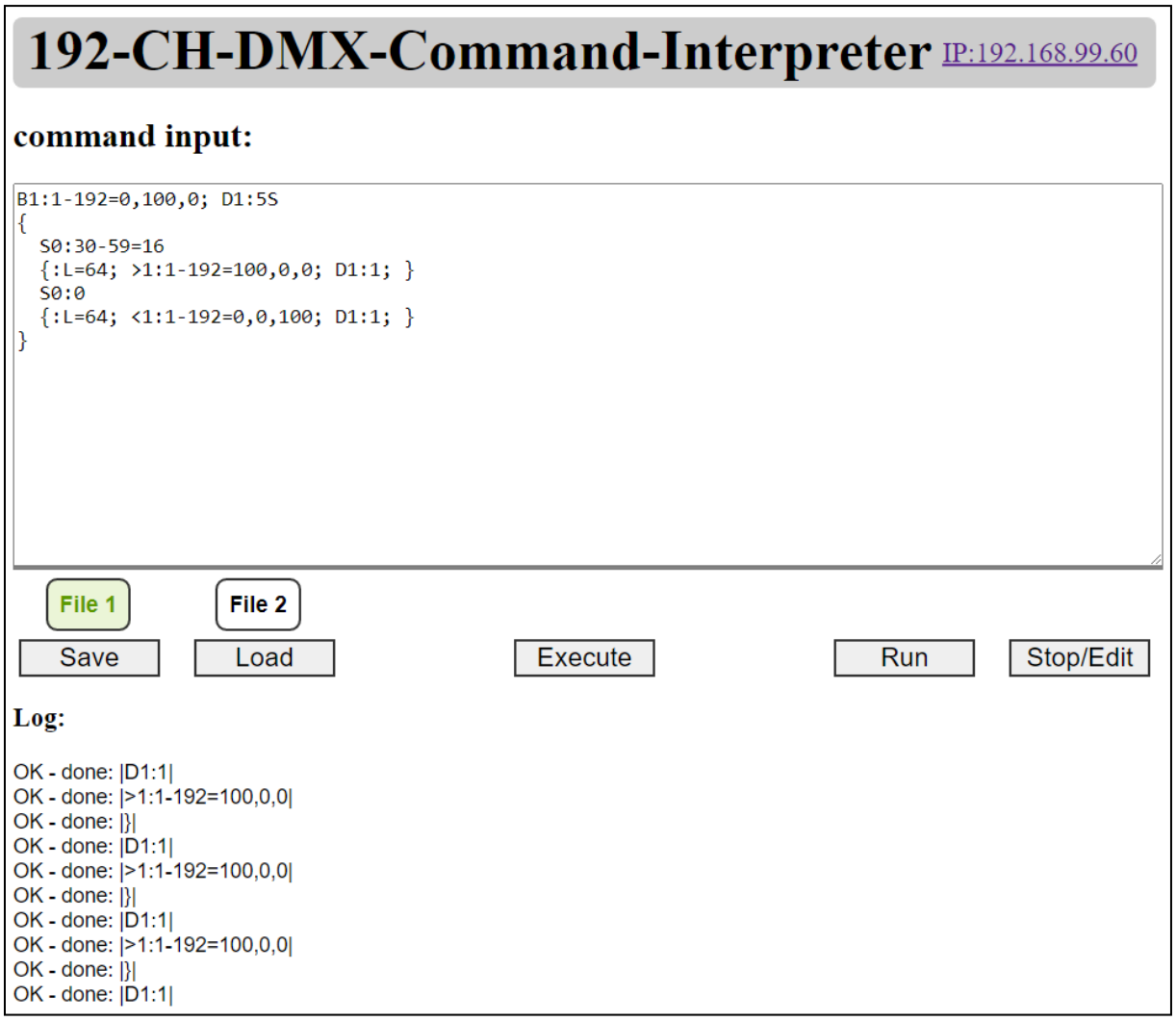
DMX-Command-Interpreter

The basic concept of the DMX-command-interpreter is based on two buffers with 192 channels each. For setting the channel values in these two buffers, there is a command that can be used flexibly to set the values of all channels or of channel ranges (fixtures).

Further commands then make it possible to output these two buffers as scenes to the DMX system in a time-controlled way statically one buffer after the other or dynamically with sliding transitions (fade) between buffers.

If the values from the buffers have been output to the DMX system and a time-controlled sliding transition is not currently taking place, the values in the buffers can be changed without direct effect on the DMX system. Complex sequences can be realised by repeatedly setting the buffers and then outputting them. As additional effects, two commands can be used to set a flash (strobe) for certain groups of LEDs and to shift the values of the channels in a predefined range (shift).

For sequence control, there is also a command to form loops that are processed permanently, for a certain time or for a certain number of runs.



The picture shows the web interface of the command interpreter. At the top right is the current web address. Below this, the commands for DMX control are entered. These can be entered line by line (no additional separator required) or within a line separated by ';'. Indenting lines with spaces to improve clarity is possible (the tab key has a different function in the web browser).

Commands entered in the input area can be stored in two files, each of which occupies half of the 4096-character EEPROM memory. Thus, each storable file can be approximately 2040 characters in size. The selection of whether file memory 1 or 2 is to be used is made via the selection buttons. Below this are the buttons to "Save" the input area into the selected file or to "Load" the program from the file into the input area.

When the voltage of the ESP32 is switched on, the command interpreter immediately processes the programm stored in file 1. It is thus possible to use the command interpreter to control a lighting system immediately after switching on the operating voltage.

Double-clicking on the input area selects its entire content. So it is easy to transfer the content to another editor, to edit it there or (with more than 2 programmes) to save it separately.

The content of the input area is automatically converted to capital letters and optimised.

The middle button "Execute" enables commands for setting and changing the channel values in the two buffer memories to be immediately output to the lighting system. Even if the setting of the channel values only affects the buffer memories, the effect of the channel values, e.g. for red, green and blue of RGB LEDs, can be immediately assessed and adjusted without first having to run a programme. For immediate execution of commands, the input cursor must be positioned somewhere under the corresponding command and then the "Execute" button must be clicked. This direct execution is not possible for all commands.

The button "Start" does just this for the programme in the input area. It’s saving to one of the two files is not necessary before. The programme is processed directly from the input area. In the "Log" area at the bottom left on the screen (scrolling down), confirmations of command processing are output and errors are displayed. If no further processing is possible due to an error, processing stops.

Since the programme is interpreted directly from the input area, the input area must not be changed after starting. Therefore, the input area is locked after the start button is pressed. If the programme has been completed or interrupted by an error, the editing option of the input area is not automatically reactivated.

The "Stop/Edit" button therefore has several functions. It interrupts a programme that is currently running (including the programme from file 1 that was automatically executed at the start). If the input cursor cannot be placed in the still locked input area by clicking, the input must be released again by clicking the button "Stop/Edit”. A double click on the button also sets all channels in the connected lighting system to the value 0 (blackout).

Commands

The commands consist of a character at the beginning, followed by the number of the buffer memory, the address information of DMX channels, values for the channels or time specifications for display duration, time for the thread or number or duration of the execution of programme loops, depending on the type of command. Separators are used within commands for unambiguous localisation. There are no spaces within a command. At the end of a command there is a semicolon or the end of line.

Command B - Setting values for the DMX channels in buffer memories 1 and 2.

Structure: **B** <p> : <start channel> , or - <end channel> = <value1> , <value2>, ... ,<value48>
<p> -> 1 or 2, followed by a colon as separator.
<start channel> <end channel> -> 1...192
Separator between channels -> alternatively comma or hyphen
Separator between channel numbers and values -> equal sign
<values> -> -1, 0, ... , 255 (1 to 48 values are possible, separated by commas).

In command B, the next character after the initial letter is the indication of buffer memory 1 or 2, followed by a colon. After that, the range of DMX channels whose values are to be changed in the buffer memory must be specified. A comma or a hyphen can be written between the start channel and the end channel as a separator. It is permissible to specify only the start channel if only the value for one channel is to be written in the buffer memory. The number of the end channel must not be smaller than that of the start channel. An equal sign follows the specification of the channel range.

The values after the equal sign are written consecutively in the specified channel range. At least one value must be specified, a maximum of 48 values are possible. How many of these values are actually written into DMX channels of the buffer memory depends exclusively on the length of the range between the start channel and the end channel (both inclusive). If only one start channel is specified, only the first value is written, further specified values are ignored. If the length of the range between the start channel and the end channel is greater than the number of specified values, the values are used cyclically a corresponding number of times until all channels of the specified range have received a value. If a string of 64 RGB LEDs is controlled by the command interpreter, then with the channel specification 1-192 and the specification of three values 0,80,0, all 64 RGB LEDs are set to green with approximately one third of the maximum brightness. In this way, it is possible to set many channels with just a few values. With the value -1, whenever this value is used, the old value in the buffer is not changed.

Command B only writes the values in the specified buffer memory and not in the output buffer to the DMX lighting system. When the command is processed in a programme, there is only the confirmation in the log, but no recognisable reaction of the system. To find suitable colour values for the channels during command input, use the "Execute" button to be able to immediately evaluate the effect of the values in the respective channels. Usually, several commands B for different colours in parts of the lighting system (fixtures) are used in succession.

Commands > and < - Shifting the values of a range in the buffer memory.

Structure: **<** or **>** <p> : <start channel> , or - <end channel> = <value1>,<value2>, ... ,<value48>
<p> -> 1 or 2, followed by a colon as separator
<start channel> <end channel> -> 1...192
Separator between channels -> alternatively comma or hyphen
Separator between channel number and values -> equal sign
<values> -> -1, 0, ... , 255 (1 to 48 values are possible, separated by commas)

The command **>** shifts the values of the channels towards the end channel. The command **<** shifts the values in the specified range in the direction of the start channel. The steps of the shift affect (how many hops to neighbour channels) depend on the number of specified values behind the equal sign. The range for the shift, which is specified by means of the start channel and end channel, must be at least as large as the new values specified behind the equals sign. If three values are specified, the range between start channel and end channel must also have at least the size three. If the specified range is larger, all values are copied to places 3 channels away from their old position in the command direction. The new values specified in the command are always written to the freed positions in the order from the lower to the higher channel, i.e. for the three channels of an RGB LED, the value for the red colour is always the first specified value, regardless of the direction in which the freed area was shifted. If -1 is specified as a value, the old value, which was only duplicated when moving, is retained.

The effect of these commands can also be checked directly with the "Execute" button during input. For different parts of the lighting system, several move commands can be specified one after the other.

Command D - Display buffer memory (output to DMX system)

Structure: **D** <p> : <display duration>
<p> -> 1 or 2, followed by a colon as a separator
<display duration> -> number (then 10th of a second,
alternatively with a letter behind - 'S' for second or 'M' for minute)

The command D displays the specified buffer memory for a period of time. For this purpose, the interpretation of the programme is stopped for this duration. If only a number is specified as the display duration, this number is evaluated as 10ths of a second. If the letter 'S' is appended directly to the number, the duration is in seconds, if the letter 'M' is appended to the number, the duration is in minutes.

Command F - Crossfade between the two buffer memories.

Structure: **F** <p> : <display duration>
<p> -> 1 or 2, followed by a colon as a separator
<display duration> -> number (x 0.25 seconds,
alternatively with a letter behind - 'S' for second or 'M' for minute)

The command F1 starts the crossfade from buffer memory 1 to 2, the command F2 does the reverse. Approximately 250 steps are required for the crossfade, which are executed in a minimum of one millisecond each, i.e. the entire crossfade takes a minimum of approx. 0.25 seconds. This minimum crossfade time can be extended by specifying a number as a factor.

For longer fade times, however, it is more convenient to specify the duration by entering a number with the unit of measurement 'S' for seconds or 'M' for minutes. The factor is then determined accordingly.

Command S - Control flash effect (strobe)

Structure: **S** <p> : <start channel> , or - <end channel> = <control value>
<p> -> 0, 1 or 2, followed by a colon as a separator
<start channel> <end channel> -> 1...192
Separator between channels -> alternatively comma or hyphen
Separator between channel numbers and values -> equal sign
<control value> -> 0, 1 ... 10, 11 ... 19 (only one value, controls the flash behaviour)

The command S only prepares the flash effect which is executed in conjunction with the commands D and F. The definition of when and with which DMX channels flashing takes place by specifying the buffer memory and the range between start channel and end channel. If the number 1 is entered after the command letter S, the flash effect only occurs if a command D1 or F1 is currently being processed. The same applies to S2 with regard to commands D2 and F2. However, if the command S0 is executed, the flashing is not limited to one of the two buffer memories. Compatible with an older programme version where the flash effect could not yet be restricted to one buffer memory, the 0 can also be omitted - S: corresponds to S0:

Only one S-command can be active at a time, each new S-command (e.g. S1:10-12=15) terminates an S-command that may have been set earlier (e.g. S0:17-21=3).

For the specification of start channel and end channel, the same specifications apply as for command B, a range or only one channel can be specified. However, for a white flash to be generated, the channel selection for RGB LEDs must contain at least the three colours red, green and blue. A single channel that controls a red LED can also only flash red.

After the channel specification and the equal sign there is a control value. If the control value is 0, all flash effects are switched off (regardless of whether S:0, S0:0, S1:0, S2:0).

If the control value is in the range from 1 to 10, a regular flash effect is generated. The intervals between the flashes are shortest for value 1 and longest for value 10.

With control values from 11 to 19, random intervals are generated, whereby the total number of flashes per second is greatest with value 11 and lowest with value 19.

Commands { - loop start and } - loop end

Structure: **{** -> start of an endless loop

{ : T = <time duration> -> start of a timed loop

{ : L = <run count> -> start of a loop with number of runs.

} -> end of each loop

Loops are executed at least once. In a time-controlled loop (T after the colon), the duration is specified either as a number (greater than 0) without a unit of measurement in tenths of a second or, as already with other commands, with 'S' for seconds or with 'M' for minutes directly after the number. In the case of a loop with a specified number of runs (L behind the colon, number also greater than 0), the loop is runs a given number of times.

Whether a loop will run again or will be exited is only checked when the command } for the end of the loop is reached. If a loop runs for a long time because of the commands it contains, it can be, that in a time-controlled loop the time has already expired, but the loop processing has not yet reached the end of the loop again.

Because of the check at the end of the loop whether a loop is executed again, only the end of the loop and then the command following the start of the loop are shown in the log as the processed command.

Making

The hardware of the command interpreter consists minimally of an ESP32 board (the "DOIT ESP32 DevKit" and "LOLIN 32" were tested) and a MAX485 circuit for controlling the DMX bus. The MAX485 is supplied with 5 volts. For safety reasons, the DO output of the MAX485 was lowered in level to approx. 3.3 volts via two resistors before it was connected to the RX2 input of the ESP32. There are many suggestions on the Internet for setting up DMX control systems regarding the use of optocouplers, a separate DC/DC voltage supply for the MAX485, the use of BIAS resistors and a terminating resistor.

The software was translated using the Arduino IDE with the corresponding ESP32 plug-in. The adaptation of the command interpreter for access to the WIFI and the specification of a suitable IP address or a DHCP use are to be defined in the source code. A simple OTA administration is available, which must also be configured in the source code. For the web interface, two language files are available in German and English, which are selected via a variable definition.

The adapted and compiled software can be loaded onto the ESP32 via OTA (if already done one time) or serially.

Functional adjustments are also easily possible via variables, although these have not yet been tested. For example, it is possible to increase the size of the transmitted DMX packets to more than

192 channels. In this case, the time interval for sending DMX packets, which is set to 10 ms for the specially used DMX hardware, may have to be increased.

The commands are based on the comparison with a simple mixing console of a lighting system. In order not to make the project too complex, a more extensive language, variables and the use of the SPIFFS file system were omitted. Since the unchanged used functions for DMX packet transmission that I took from / https://github.com/someweisguy/esp_dmx /are based on RTOS, it might also be more elegant to realise the time control of the command interpreter not via the millis() function but also directly via RTOS functions. Here I have no corresponding experience.

With regard to the web interface, the control of the web client by the ESP32 server using AJAX functions would possibly enable a more comfortable operation. This has also fallen victim to an effort/time optimisation.

22.12.2021