Application Note

Standardized UART Protocol (with OSRAM extensions)

Table of Contents

1. Introduction

The UART interface, whose hardware details are described in the application note “5-Pin Control Interface”, offers the possibility to establish a communication via two wires between projector and OSRAM lamp driver. Every meaningful communication needs a protocol to be clarified: the interface software requirements for the user are specified in this application note. The UART communication offers a lot of beneficial functions and enables a smart control of the OSRAM lamp system.

In particular, the major section of this document defines the software communication protocol, which is based on the instruction set (containing commands and queries); the communication timing is specified too, and a detailed description of the UART commands is given. The last paragraph, presenting an overview about the hardware protocol (with its communication interface, parameters and timing), acts like a bridge to the above mentioned application note “5-Pin Control Interface”.

2. UART Software Protocol

2.1. Communication Protocol

The communication takes place as the user first sends an **instruction** via the receive data signal (RxD), according to what is described in the paragraphs 2.2.1 and 2.2.2 below. The OSRAM lamp driver determines if the instruction is a **command or query**. For a command, the OSRAM lamp driver replies back to the user with an echo via the transmit data signal (TxD). For a query, the OSRAM lamp driver replies back to the user with echo and response via the transmit data signal (TxD). Figure 1 shows a flow diagram of the communication protocol.

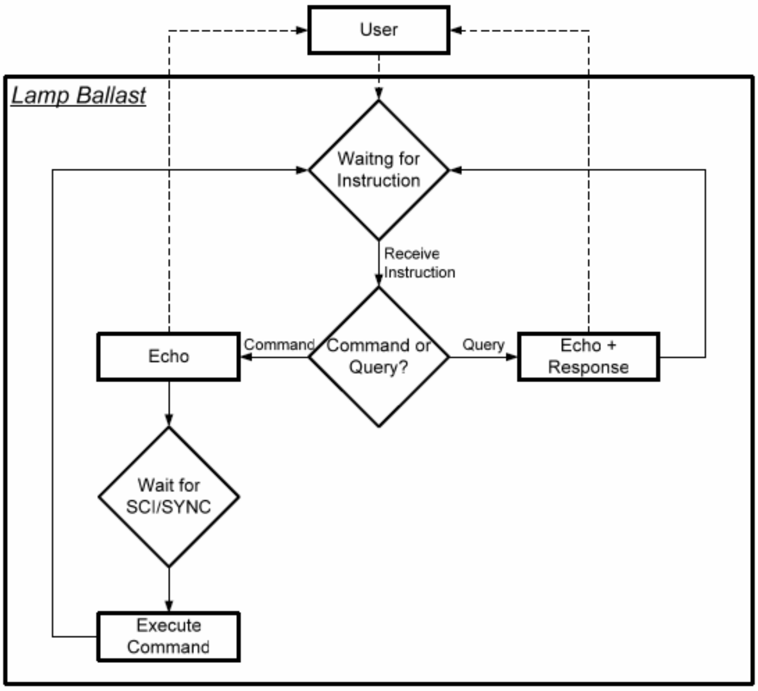


Figure 1: Communication Protocol Flow

2.2. Standardized Instruction Set

As above mentioned, there are two types of instructions: commands and queries. For a command, the OSRAM lamp driver replies with an echo. For a query, the OSRAM lamp driver replies with echo and response. Table 1 shows the instruction response structure.

Instruction	Response Structure
Command	Echo
Query	Echo + Response

Table 1: Instruction Response Structure

2.2.1. Command Structure

Each command consists of a command key and, if necessary, of additional 1 to 3 argument bytes. The lamp driver answers with a complete echo (except on the “Reset” or “Disable Communication” commands). The echo consists at least of the command key and, depending on the command, of the additional argument bytes, too.

Command keys are within the range of 00h to 7Fh (they are listed in Table 2 below).

Command Name	Key	Argument	Execution
Lamp On	**25h**	**none**	immediately (from standby)
Lamp Off	**26h**	**none**	immediately
Reset lamp driver	**3Ah**	**none**	immediately
Enable Communication	**70h**	**none**	immediately
Select Waveform	**71h**	**waveform number**	next SCI/SYNC
Set Gain	**72h**	**value**	next SCI/SYNC
Write Byte	**73h**	**data byte**	Immediately
Set Mailbox	**74h**	**16bit start address + control byte**	immediately
Disable Communication [(i)]	**75h**	**none**	immediately

Table 2: Command Set.

For some commands, one or more additional parameter bytes have to be added to the command byte in order to specify values, data, or addresses (see third column). More details about the exact meaning of the arguments are explained in the next paragraphs.

[i] Only kernel GB02 or higher

2.2.1.1. Lamp On (25h)

As the SCI signal has a "higher turning on priority" than the UART software command, the command "Lamp On" will take effect only if the SCI signal is valid at the same time.

Furthermore, after having turned off the lamp it will be possible to turn it on again earliest after a waiting time**.** This minimum OFF time is always performed and corresponds to the lamp cooling down period. Hereafter, if the SCI signal is valid and no operation error occurred, the lamp driver will try to ignite the lamp again.

If the lamp has already been turned on via SCI signal, it is not necessary to send an additional "Lamp On" command. However, if the ballast is already in UART mode before the SCI signal goes high, the lamp won't be started until a "Lamp On" command is received.

2.2.1.2. Lamp Off (26h)

With this command the lamp is switched off and the "Error Status" byte (see item number 06h in Table 13) is set to zero. If an operation failure has occurred, it is necessary to send the "Lamp Off" command in order to reset the "Error Status" byte and then to be able to start the lamp again.

2.2.1.3. Reset lamp driver (3Ah)

The "Reset lamp driver" command has the same effect like an OFF-ON input power cycle. All parameters will be set to power up conditions. Please notice, the ballast won't send an echo on this command. After command is received (similar to a power on condition) the ballast needs some time to reboot and can't react on further UART commands in this period. UART communication will stay disabled until a new "Enable Communication" command is received.

WARNING:

If a valid SCI signal is applied, the ballast will immediately try to start the lamp after the reset.

The use of the reset command also clears the delay time before a new ignition, and the execution of successive ignition cycles without delay time can cause permanent damage to the lamp driver (Danger of overheating the resonance choke and full bridge transistors!).

2.2.1.4. Enable Communication (70h)

This command **opens** up the communication between the user interface and the lamp driver. After command is received gain factor is set to 100%. If this command is to be used during a different gain setting a set gain command (see below) must be issued directly after communication is enabled to restore the old gain factor. Once the access is enabled, it remains enabled until power is removed from the ballast or a disable communication command (see below) is received.

2.2.1.5. Disable Communication (75h) (Only GB02 or higher)

This command **closes** the UART communication between the user interface and the lamp driver. Once communication is disabled the ballast is only controlled by synchronization signal (SCI / SYNC) and the TxD / FLAG signal acts as a status flag again. No echo is send, as UART communication will be disabled immediately.

2.2.1.6. Select Waveform (71h)

This command shall be used to **select** one waveform among the different ones programmed into the ballast. The number of the waveform to be selected is a single byte defined in Table 3. If the selected waveform number is not available, the command returns an error. The select waveform command refers to the waveform number, not the ID.

The command will take effect on the next synchronization signal (SCI / SYNC).

Number	Waveform name (example)	ID (example)
00h (default)	RGBRGB Standard	07h
01h	2X@60Hz, 9bit	FEh
02h	3X@50Hz, 10bit	7Fh
…	…	…

Table 3: Enumeration of the waveforms stored in a lamp driver

The waveform number is independent on the waveform ID. The waveform number just enumerates the waveforms stored in the lamp driver. The waveform ID identifies the unique waveform independent on where it is stored.

2.2.1.7. Set Gain (72h)

This command shall be used to **set** the output **power** level of the lamp driver. The value must be within the allowed range between min gain and max gain. Min and max and actual gain can be read out by the "Min Gain", "Max Gain" and "Act. Gain" queries, see queries' sections 2.2.2.9 and 2.2.2.11. If the value is outside the specified range, the command returns an error. The gain can be calculated from the single byte argument "value" with the formula below:

$$Gain = \frac{Value}{128} \ [\%]$$

Example1: command "72h 80h" ⟶ *Value* =80h=128d ⟶ $Gain = \frac{128}{128} = 1 = 100\%$ output power level.

Example2: command "72h A6h" ⟶ *Value* =A6h=166d ⟶ $Gain = \frac{166}{128} \approx 1{,}3 = 130\%$ output power level.

The command takes effect on the next synchronization signal (SCI / SYNC).

2.2.1.8. Write Byte (73h)

This command is used to **write data** into specific memory addresses of the lamp driver's microcontroller. As target address, the current mailbox address registers will be taken (see 2.2.1.9). The memory address is automatically incremented after execution of the "Write Byte" command (**post increment**).

Command Key	Data Byte
73h	Data

Table 4: Syntax of the Write Byte command

2.2.1.9. Set Mailbox (74h)

This command is sent to **enable memory write access and set the target address** for the “Write Byte” command (73h) or the “Byte” query (F9h). The “Set Mailbox” command is a 4 byte command. Its syntax is defined in Table 5. Table 6 defines the mailbox control register. The mailbox control register enable bit is defined in Table 7.

Address range 0000h to 7FFFh is used for addressing EEPROM; 8000h to FFFFh address the SRAM. The real size of the EEPROM and SRAM may be shorter and depends on the control board used. For generation 3, 4 and 5 control boards the EEPROM is 512 bytes long (0000h to 01FFh). For control boards of generation 3 and 5 the SRAM is 1024 bytes long (8000h to 83FFh), for generation 4 it’s 512 bytes long (8000h to 81FFh).

Command Key	Byte 1	Byte2	Byte 3
	Mailbox Address Registers		Control Register
74h	High Byte	Low Byte	Control Byte

Table 5: Mailbox address and control registers

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved for future use							**Enable**

Table 6: Control Register Definition

Value	Definition
0	Disable memory write (default)
1	Enable memory write access

Table 7: Control register’s **Enable** bit definition

2.2.2. Query Structure

The key range F0h to FFh is reserved for queries. Table 8 lists the query set. The third column (“Response”) indicates the structure of the response given from the lamp driver after having received the corresponding query, which is indicated in the column “Key”. More detailed information about every single query is given later on in this chapter.

Query Name	Key	Argument	Response
Company ID	F0h	none	F0h + **Osram ID**
Lamp driver ID	F1h	none	F1h + **hardware ID + software ID**
Waveform ID	F2h	none	F2h + **waveform ID**

Query Name	Key	Argument	Response
Selected Waveform	F3h	none	F3h + **number**
Act Gain	F4h	none	F4h + **gain value**
Status	F5h	none	F5h + **status byte**
Waveform Count	F6h	none	F6h + **count**
Mailbox Control	F7h	none	F7h + **16 bit address + control byte**
Query Byte	F9h	none	F9h + **data**
Min Gain	FAh	none	FAh + **min gain value**
Max Gain	FBh	none	FBh + **max gain value**
Item Address	FFh	**item**	FFh + **item + 16 bit address**

Table 8: Query Set

2.2.2.1. Company ID (F0h)

This query returns the OSRAM ID as a single byte, defined as **01h**.

2.2.2.2. Lamp driver ID (F1h)

This query returns the ballast **hardware and software ID**. The first response byte returned is the ballast's hardware ID (it characterises the hardware platform) and the second byte is the ballast software ID (it characterises the software kernel); Table 9 defines the ballast ID response structure. Table 11 shows the platform names to the hardware IDs and Table 10 displays the kernel names to the software IDs.

Query Key	Response Byte 1	Response Byte 2
F1h	hardware ID	software ID

Table 9: Lamp driver ID response structure

Kernel Name	Feature Description	Software ID
DB03	UNISHAPE alpha	05h
DB04	UNISHAPE beta	06h
DB05	UNISHAPE release	07h
DB06	UNISHAPE 1.1	08h
DB07	UNISHAPE 1.2	09h
DB08	UNISHAPE 1.5	0Ah
DB09	UNISHAPE 2	0Bh
EA03	Gen4 Beta UNISHAPE/EASYNC	0Eh
EA04	Gen4 UNISHAPE/EASYNC	0Fh
EA05	Gen4 UNISHAPE/EASYNC	10h
EA06	Gen4 UNISHAPE	11h
EB06	Gen5 E series compatible UNISHAPE	12h
GB00	Gen5 GB series UNISHAPE beta	13h
GB01	Gen5 GB series UNISHAPE RC1	14h
GB02	Gen5 GB series UNISHAPE first release	15h
GB02P	Gen5 GB series UNISHAPE with PPR	17h
GB03	Gen5 GB series UNISHAPE	18h
GB04	Gen5 GB series UNISHAPE	19h
GB05	Gen5 GB series UNISHAPE	1Ah
GB04P	Gen5 GB series UNISHAPE with PPR	1Bh
GI01	Gen5 MULTISYNC series	1Ch
GI02	Gen5 MULTISYNC series	1Dh
GI03	Gen5 MULTISYNC series	1Eh

Table 10: OSRAM software kernel IDs

OSRAM Lamp driver Name (PT-VIP)	Hardware ID	OSRAM Lamp driver Name (PT-VIP)	Hardware ID
132AC/100-240 Q	00h	5AC/380 O1 Rev.0	11h
150AC/100-240 H1	01h	4.3AC/380 O3 Rev.0 (HC)	12h
150AC/100-240 P2	02h	O1 RP 132W	13h
150AC/100-240 Q	03h	O1 RP 180W	14h
180AC/100-240 H1	04h	O3 MID 200W	16h
180AC/100-240 H2	05h	O3 MID 230W	17h
180AC/100-240 P2	06h	O6 MIC 200W	18h
200AC/100-240 H	07h	O3 TOP 280W	19h
200AC/100-240 P	08h	O4 MEGA 350W	1Ah
2AC/380 O1 Rev.0	09h	165AC/100-240 H4	1Bh
2AC/380 O1 18kV	0Ah	O3 MID DL 230W (Gen 4)	1Ch
3AC/380 O1 Rev.0	0Bh	O1 RP RES	1Dh
3AC/380 O3 Rev.2	0Ch	O3 MID DL 230W (Gen 5)	1Eh
4AC/380 O1 Rev.1	0Dh	O6 MIC 180W	1Fh
4AC/380 O1 Rev.2	0Eh	O3 MID DL 180W	20h
4AC/380 O3 Rev.0	0Fh	O3 MID SL 180W	21h
5AC/380 O4 Rev.3	10h		

Table 11: OSRAM hardware platform IDs

2.2.2.3. Waveform ID (F2h)

This query returns the single byte **ID** of the waveform which has been selected by the “Select Waveform” command (71h, see section 2.2.1.6). This is a unique ID that is tied to the waveform.

2.2.2.4. Selected Waveform (F3h)

This query returns the single byte **number** of the actual selected **waveform** (see section 2.2.1.6 “Select Waveform”).

2.2.2.5. Act Gain (F4h)

This query returns the single byte value of the actual waveform **gain**, which has been set with the command “Set Gain” (for gain calculation formula, see section 2.2.1.7 “Set Gain (72h)”).

2.2.2.6. Status (F5h)

This query returns a single **byte**, **status of the ballast**. Table 12 defines the bits of the ballast status register.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved for future status						Over Temp	Lamp ON

Table 12: Ballast Status Byte Definition

2.2.2.7. Waveform Count (F6h)

This query returns a single byte, representing the total number of waveforms stored in the lamp driver.

2.2.2.8. Mailbox Control (F7h)

This query returns the **mailbox 16 bit address and control register,** see section 2.2.1.9 for detailed description.

2.2.2.9. Query Byte (F9h)

This query returns the **8 bit data of** a **memory location** in the lamp driver's memory. The memory location address is specified by the **mailbox address registers**, see Table 5. This address is automatically incremented after execution of "Query Byte" (**post increment**).

16 bit values can be read by two successive executions of "Query Byte". The byte order in the microcontroller's memory is first low byte, then high byte. You should send the two commands with the lowest possible gap to prevent that the 16 bit value is changed between the two read operations.

2.2.2.10. Min Gain (FAh)

This query returns the minimum allowed gain value for the lamp. (for gain calculation formula, see section 2.2.1.7 "Set Gain (72h)")

2.2.2.11. Max Gain (FBh)

This query returns the maximum allowed gain value for the lamp. (for gain calculation formula, see section 2.2.1.7 "Set Gain (72h)")

2.2.2.12. Item Address (FFh)

Query "Item Address" sets the Mailbox' address registers to the 16 bit memory address of the selected item. The purpose of this query is to be kernel independent. Memory addresses can change between kernels, but the item numbers are fixed.

The query consists of two bytes: the query key (FFh) followed by the item number. Item numbers are defined in Table 13. After the query execution the mailbox address registers are automatically set to the item's address. Therefore no successive Set Mailbox command is needed. Only the Write Enable Bit needs to be set once before being able to *write* data to the lamp driver, see section 2.2.1.9 for detailed information.

Item Name	Item Number
Ballast Temperature	01h
Lamp Voltage	02h
Lamp Current	03h
Lamp Power	04h
Operation Status	05h
Error Status	06h
Write Access Password	08h
I_{max} [ii]	0Ch
U_{max} [ii]	0Dh
Pulse-Plateau Ratio [iii]	7Eh
Waveform data in SRAM [iv]	80h
Waveform data in EEPROM	81h
Electronic Label	82h

Table 13: Item name and number

The following subsections describe the single items listed in the table above.

The items with numbers from 01h to 7Fh are 16 bit variables. They can be read out by two successive Query Byte instructions. Their byte order is low byte first, then high byte. In some rare cases, when the content of the 16 bit value is changed between the two read operations, the result may be wrong. To prevent this, you should send the two commands with lowest possible gap.

The items with numbers beginning from 80h store data without a fixed length. The first byte contains the total number of bytes used by the item (including the first byte itself). For reading the whole item first read the data length. Then perform [data length - 1] successive Query Byte instructions.

Please see also chapter Examples for detailed information.

[ii] Only Kernel GB00 or higher
[iii] Only Kernel GB-“P“ or GI series
[iv] Only Kernel DB03 to DB09

a) Ballast Temperature

The microcontroller measures the temperature in order to avoid an overheating of the lamp driver. The temperature is stored in the variable “Ballast Temperature”. Its value in microcontroller units can be converted to a temperature in degree centigrade. The relationship between microcontroller units and centigrade depends on the temperature sensor used. Please refer to the lamp driver specification for this information.

b) Lamp Voltage

This item contains the measured lamp voltage. The 16bit value 0...65535 corresponds to lamp voltages between 0 and U_{max}. The voltage in Volts can be calculated as follows:

$$U[V] = U_{\mu C} \cdot \frac{U_{max}}{65535} = U_{\mu C} \cdot f_U$$

f_U can be read from the system specification. With kernel GB00 or higher you can also read out the U_{max} Item.

See also Example 4: Reading out the lamp voltage.

c) Lamp Current

The lamp driver controls the lamp current. The average lamp current can be read out from these registers. With the actual current and lamp voltage it is possible to calculate the actual power, e.g. during the run-up phase. The following equation defines the relationship between physical current in Amps and its value in microcontroller units:

$$I[A] = I_{\mu C} \cdot \frac{I_{max}}{1023} = I_{\mu C} \cdot f_I$$

Again, f_I is defined in the system specification. With kernel GB00 or higher you can also read out the I_{max} Item.

d) Lamp Power

This item contains the present target power. The relationship between the microcontroller units and the physical power in Watts is defined by the following equation, where f_P is defined by the system specification. In most cases the factor is just 1, so one can read out the lamp power directly:

$$P[W] = P_{\mu C} \cdot f_P$$

Note that the lamp power does not change with the SetGain command. It refers to the wattage of the used lamp type.

See Example 2: Reading out the actual power in Watt.

e) Operation Status

The Operation Status is given out by the lamp driver as 16bit-Value.

Bits 15..8	Bits 7..0
Reserved	Operation Status, see Table 15 below

Table 14: Operation Status response structure

Operation Status	Description
00h	Standby
01h	Ignition
02h	Ignition
03h	Ignition
04h	Lamp run-up
05h	Cool down
06h	Normal lamp operation
07h	Reserved
08h	Shut down due to an unrecoverable error
09h	Pre-heating phase
0Ah	Reserved
0Bh	Reserved
0Ch	Pre-heating phase

Table 15: List of Operation Statuses

f) Error Status

The Error Status is given out by the lamp driver as 16bit-Value.

Bits 15..8	Bits 7..0
Reserved	Error Status, see Table 17 below

Table 16: Error Status response structure

Error Status	Description
00h	No error
01h	Temperature shut down
02h	Short circuit in output detected
03h	End of lamp life detected
04h	Lamp did not ignite

Error Status	Description
05h	Lamp extinguished during normal operation
06h	Lamp extinguished during run-up phase
07h	EEPROM write error
08h	EEPROM write buffer overflow
09h	UART buffer overflow
0Ah	Lamp current calculation error
0Bh	Corrupted software configuration
0Ch	Lamp voltage too low
0Dh	Reserved
0Eh	Reserved
0Fh	EEPROM configuration does not match FLASH kernel
10h	Maximum pre-heating time elapsed

Table 17: List of Error Statuses

g) Write Access Password

In order to protect the microcontroller's non volatile memory from accidentally overwriting, the write access is controlled by these registers. It is a precondition for writing to EEPROM memory that their value equals to 56AEh. Otherwise Write Byte commands have no effect on the EEPROM and the lamp driver will return an error code. The Enable bit in the mailbox control register must be set, too.

For writing to the SRAM setting the Enable bit in the mailbox control register is sufficient.

In Kernel GB00 or higher, the password doesn't need to be set when writing to the waveform area of the EEPROM, but when other configuration data is written.

WARNING:

Be careful when writing to the EEPROM out of the waveform area. Only write data approved by OSRAM to the lamp driver. Wrong configuration data may lead to permanent damage of the lamp driver or the lamp. It could also prevent security features like temperature protection from working correctly.

h) I_{max} (Only Kernel GB00 or higher)

This item contains the maximum current of the lamp driver in mill amperes. It can be used for conversion between the microcontroller units returned by the Lamp Current item and physical units. It is not used by the lamp driver itself.

i) U_{max} (Only Kernel GB00 or higher)

This item contains the maximum voltage of the lamp driver in volts. It can be used for conversion between the microcontroller units returned by the Lamp Voltage item and physical units. It is not used by the lamp driver itself.

j) Pulse-Plateau Ratio (Only Kernel GB-“P“ or GI series)

This item contains a 16 bit value for the current Pulse-to-Plateau Ratio (PPR). The PPR is only valid for standard waveforms with a single pulse.

On GB-“P” Kernels the value is recalculated only at the corresponding QueryAddress command.

On GI Kernels, the value is recalculated each waveform. It may take up to two waveforms after lamp enters normal operation mode until there is a valid PPR. In this time, the return value is 0.

You can calculate the actual Ratio from the microcontroller values by the following formula:

$\frac{I_{Pulse}}{I_{Plateau}} = \frac{PPR_{\mu C}}{16384}$. The value of $\frac{I_{Pulse}}{I_{Plateau}}$ can therefore be 0 to 4.

k) Waveform data in SRAM (Only kernel DB03 to DB09)

The active UNISHAPE waveform data is stored in this item. The item is copied from EEPROM on every start-up of the lamp driver. Afterwards it can be changed even on-the-fly without affecting the permanently stored waveform data in EEPROM.

l) Waveform data in EEPROM

This item contains the permanently stored waveform data of the lamp driver. In kernel DB03 to DB09 it is copied to SRAM on every start-up of the lamp driver. On newer kernels, the waveform is always read from the EEPROM.

m)Electronic Label

The electronic label is a string of ASCII characters which individually identifies the lamp driver.

2.2.3. Error Response Structure

If the OSRAM lamp driver receives an undefined or unrecognisable instruction, it replies with an error byte (AAh) followed by the echo. Table 18 shows the error response structure.

Parameter	Response Structure
Echo	AAh + Echo

Table 18: Error response structure

If the lamp driver’s UART input buffer is full and the ballast still receives new characters, it sends ABh as error code for each character. If a communication timeout occurs, the error code ABh is sent, too (see section 3.2)

In case the lamp driver detects parity or framing errors during receiving, it sends ACh as response. All data received in 50 ms after the error will be ignored.

3. Hardware Protocol

3.1. Communication Interface

The implemented UART interface is a serial two wire opto-isolated interface. It consists of a transmit data line (TxD) and a receive data line (RxD). RxD and TxD lines are used for two-way (full-duplex) communication. There is no handshake used for flow control.

For the logic level definition of transmit and receive line as well as for the electrical interface specification refer to the application note “5-Pin Interface”.

3.2. UART Parameters and Timing

The UART serial communication parameters comply with EIA-RS-232-C Interface Standard. The frame properties are specified in Table 19. Every frame starts with the **start bit** (which is always **0**), followed by the least significant data bit (indicated with D0). Then the next data bits are succeeding, ending with the most significant bit (indicated with D7).

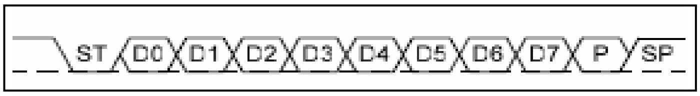


Figure 2: Frame Structure (ST=Start Bit, Dn=Data Bits, P=Parity Bit, SP=Stop Bit)

The parity bit is inserted after the data bits, before the **stop bit** (which is always **1**). When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. Each **frame** corresponds to a **single byte of information**.

Each command or query can consist of one or several successive **RxD frames**. The lamp driver will answer accordingly with one or several successive **TxD frames**. Figure 3 shows the communication flow and defines the timing limit for receiving and transmitting $\mathbf{T}_{\textbf{MAX Command}}$ and $\mathbf{T}_{\textbf{MAX Response}}$.

Following timing limits in the communication flow have to be kept:

- if after the time $\mathbf{T}_{\textbf{MAX Command}}$ = **15 ms** a whole command is not transmitted completely to the lamp driver, the command queue is cleared (this is necessary to synchronize the command decoding algorithm with the projector electronics) and the error byte ABh is returned;
- the response will start earliest after having received the first frame of a command and latest after having received the last frame of a command plus a processing time, which can vary between zero and 5 ms. The whole response time is guaranteed to be finished after $\mathbf{T}_{\textbf{MAX Response}}$ = **10 ms**.

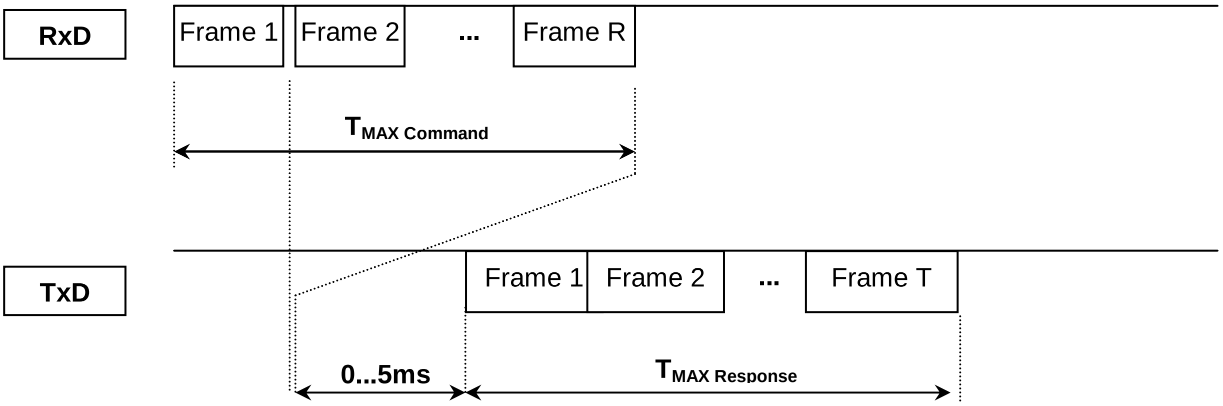


Figure 3: Communication flow chart

NOTE: Because no flow control is used, we highly recommend sending a command only after having received the answer of the previous command. This is to avoid an overrun of the receiving buffer and to ensure the proper execution of the commands!

Frame Parameter	Value	Tolerance
Baud Rate	9600	±1%
Number of Start Bits	1	n/a
Number of Data Bits	8	n/a
Number of Stop Bits	1	n/a
Parity Bit	Even	n/a

Table 19: UART Parameters

The **parity bit** is calculated by doing an **exclusive-or** of all the data bits. As the defined parity is "even", the relation between the parity bit and the data bits (D_0 ... D_7) is as follows:

$$Parity = D_0 \oplus D_1 \oplus D_2 \oplus \ldots \oplus D_7$$

In other words, the parity bit is 0 if the number of "1" bits is even.

3.3. Example of a single Byte command

Following oscilloscope image is an example of transmit and receive signal for a one byte transmission:

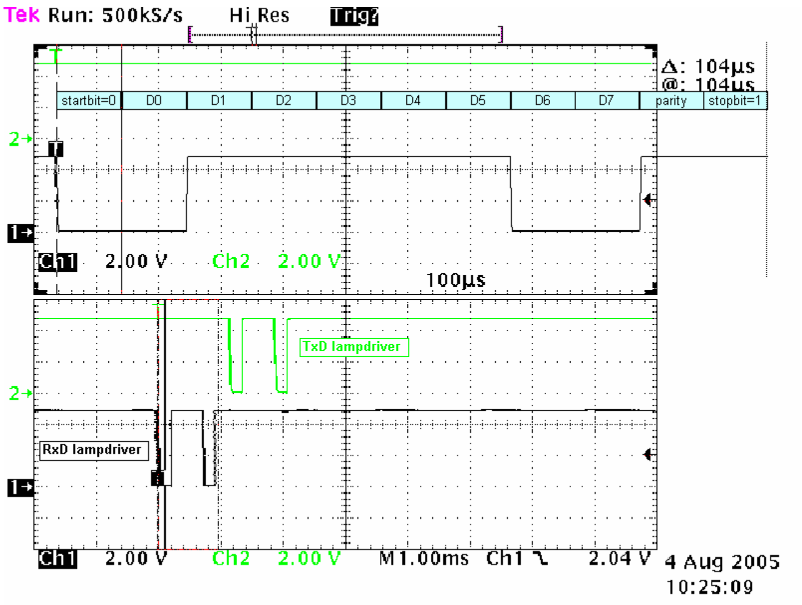


Figure 4: Oscilloscope plot of 3Eh command

The command transmitted from the projector is an “Enable Communication” command, in this example by means of the command byte “3Eh”. The binary conversion of “3Eh” is 00111110, represented by the last signal in the figure above and with an enlarged scale in the first position: please note that the first bit, D0, is the least significant one.

The green signal in the middle is the lamp driver’s answer (command echo). The measurement was done between signal pins and projector GND, using 5V supply voltage.

4. Examples

4.1. Example 1: Setting the lamp driver to maximum power

In this example we want to set the output power to the lamp driver’s maximum gain. Let us assume the maximum gain was 93h (115%). If we do not know the maximum gain in advance we first need to read it out. Then we set the waveform gain to this value.

	Send to lamp driver	**Receive**
Enable Communication	70h	70h
Reading out maximum gain	FBh	FBh 93h
Setting waveform gain to maximum	72h 93h	72h 93h

4.2. Example 2: Reading out the actual power in Watt

As the lamp driver's power is not given as an absolute value in Watt we need to read out the nominal power first and then multiply it with the actual waveform gain. Let us assume the lamp driver has 300W as nominal power (012Ch) and the actual gain was 90% (73h).

	Send to lamp driver	**Receive**
Enable Communication	70h	70h
Setting mailbox address to item "power"	FFh 04h	FFh 04h 23h 45h
Read item's low byte	F9h	F9h 2Ch
Read item's high byte	F9h	F9h 01h
Query actual gain	F4h	F4h 73h

With this information we can now calculate the actual power in Watt:

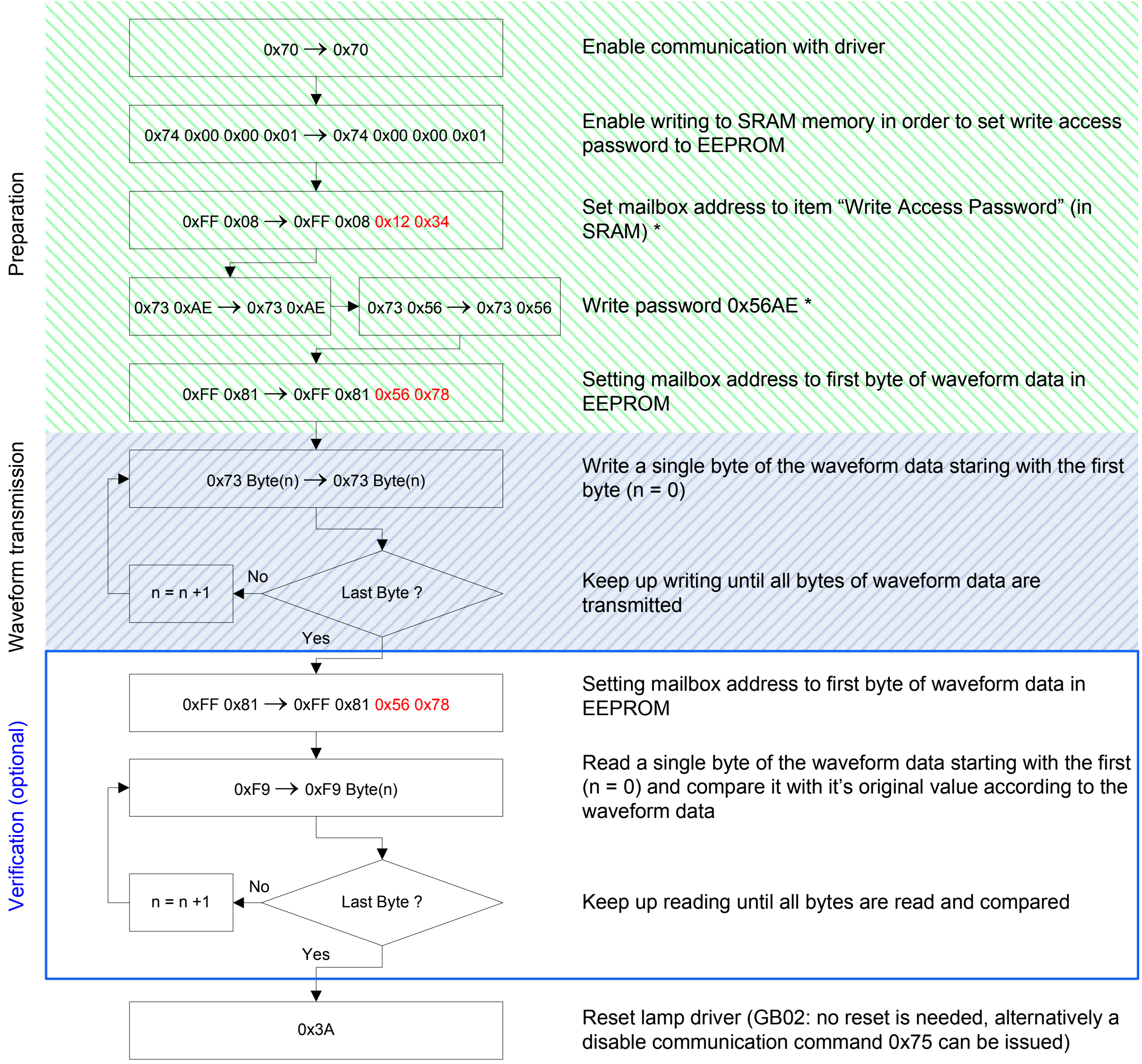
$$P = P_{\text{nominal}} \cdot \frac{\text{act gain}}{128} = 300\text{W} \cdot \frac{115}{128} = 270\text{W}$$

4.3. Example 3: Uploading a new waveform

For uploading a new waveform to EEPROM we first need to enable write access to EEPROM. To do so the item "Write Access Password" must be set to 56AEh. Then we can write the new waveform data to EEPROM.

First we set the mailbox' write enable bit to true. Then we select the item "Write Access Password". Now we can write the EEPROM write access password. Then the new waveform data can be written to the EEPROM.

	Send to lamp driver	**Receive**
Enable Communication	70h	70h
Enable writing to memory	74h 00h 00h 01h	74h 00h 00h 01h
Setting mailbox address to item "Write Access Password"	FFh 08h	FFh 08h 12h 34h
Write password's low byte	73h AEh	73h AEh
Write password's high byte	73h 56h	73h 56h
Setting mailbox address to item "Waveform data in EEPROM"	FFh 81h	FFh 81h 56h 78h
Write Byte1	73h 60h	73h 60h
Write Byte2	73h 01h	73h 01h
...	...	...
Write ByteN	73h FFh	73h FFh


Preparation
Waveform transmission
Verification (optional)
0x70 → 0x70
Enable communication with driver
0x74 0x00 0x00 0x01 → 0x74 0x00 0x00 0x01
Enable writing to SRAM memory in order to set write access password to EEPROM
0xFF 0x08 → 0xFF 0x08 0x12 0x34
Set mailbox address to item "Write Access Password" (in SRAM) *
0x73 0xAE → 0x73 0xAE
0x73 0x56 → 0x73 0x56
Write password 0x56AE *
0xFF 0x81 → 0xFF 0x81 0x56 0x78
Setting mailbox address to first byte of waveform data in EEPROM
0x73 Byte(n) → 0x73 Byte(n)
Write a single byte of the waveform data staring with the first byte (n = 0)
n = n +1
No
Last Byte ?
Yes
Keep up writing until all bytes of waveform data are transmitted
0xFF 0x81 → 0xFF 0x81 0x56 0x78
Setting mailbox address to first byte of waveform data in EEPROM
0xF9 → 0xF9 Byte(n)
Read a single byte of the waveform data starting with the first (n = 0) and compare it with it's original value according to the waveform data
n = n +1
No
Last Byte ?
Yes
Keep up reading until all bytes are read and compared
0x3A
Reset lamp driver (GB02: no reset is needed, alternatively a disable communication command 0x75 can be issued)

Remarks:
- The two steps marked with * can be left out in case of a GB02 kernel
- Red values are kernel dependent and are only given as example
- Byte(n) denotes the n-th byte of the waveform information
- The comparison during verifcation must be done by the external device and is not carried out by the ballast itself

Figure 5: **Flow chart of the waveform upload process.**

How the waveform data is organised in the EEPROM of the lamp driver depends on the kernel and is described in corresponding firmware application note of each kernel. The waveform data is stored in Intel-Hex format in the EEP file but it must be extracted from this format **before** it can be transmitted to the lamp driver. A description of the Intel hex format is available as separate Application Note or can easily be found on the web.

4.4. Example 4: Reading out the lamp voltage

After the communication is enabled, we first set the mailbox' address pointer to the item "Lamp Voltage". This is done by QueryAddress. This query returns the item's memory address (dependent of firmware version, in this example 8107h) and automatically sets the mailbox register to that address. Now we can read out the low and high byte of the 16bit value that corresponds to the lamp voltage. This is done by two successive QueryByte calls.

In our example, we received the value 7BDEh=31710 for the lamp voltage.

The relation between this value and the actual voltage depends on the hardware used. Refer to section 2.2.2.12.b) on how to convert the values. In this example we assume $f_U = \frac{155\,V}{65535} = 0{,}002365\,V$, so we have a lamp voltage of $U = 31710 \cdot 0{,}002365\,V = 75\,V$.

	Send to lamp driver	**Receive**
Enable Communication	70h	70h
Setting mailbox address to the lamp voltage's item address	FFh 02h	FFh 02h 81h 07h
Read item's low byte	F9h	F9h DEh
Read item's high byte	F9h	F9h 7Bh

5. Application Note Change Log

2008-06-13 Version 3.3:

- Updated Section 2.2.12.j) "Pulse-Plateau Ratio"

2008-03-28 Version 3.2:

- Updated Example 4

2008-03-05 Version 3.1:

- Corrected Pulse-Plateau item number in table 13

2008-03-04 Version 3.0:

- Updated Commands:
  Lamp On, Reset Lamp Driver, Enable Communication, Disable Communication, Set Mailbox
- Updated Tables:
  Table 10: OSRAM software kernel IDs; Table 11: OSRAM hardware platform IDs; Table 13: Item name and number; Table 15: List of Operation Statuses; Table 17: List of Error Statuses
- Updated Item Descriptions:
  Lamp Voltage, Lamp Current, Write Access Password, I_{max}, U_{max}, Pulse-Plateau-Ratio, Waveform data in EEPROM
- Added flow chart of waveform upload process
- Section 2.2.3. "Error Response Structure" updated
- Section 3.1 "Communication Interface"
- Minor typographic and layout corrections

2007-08-13 Version 2.5:

- Added errors during receive
- Updated table of kernels
- Updated table of lamp driver

2007-06-13 Version 2.4:

- Added new command "Disable communication"

- Updated table of kernels

2006-12-21 Version 2.2:

- Added Example 4: Reading out the lamp voltage and updated the Query ItemAddress description

2006-11-13 Version 2.1:

- Updated Table 11: OSRAM hardware platform IDs
- Updated Table 10: OSRAM software kernel IDs

2006-09-19 Version 2.0:

- Updated Table 11: OSRAM hardware platform IDs

2006-09-15 Version 1.9:

- Corrected kernel table: EA03 and following kernels share same software ID for Unishape and Easync operation modes

2006-08-30 Version 1.8:

- Updated kernel table (EA04)
- Corrected description of Queries Error/Operation Status

2006-04-01 Version 1.6:

- Updated kernel table (EA03)

2006-01-29 Version 1.5:

- Added Examples chapter
- Updated kernel and lamp driver table

2005-10-15 Version 1.4:

- Corrected command keys of “Query Byte”

2005-10-01 Version 1.3:

- Updated chapter “Hardware Protocol”
- Changed key for “Item Address” Query
- Changed order of error structure

2005-08-01 Version 1.2:

- New Query “Item Address”

Authors: B. Dobler (b.dobler@osram.com), M. Baier, C. Breuer, A. Huber