

Commands

for

STR1xxxxxx – RS-485

smart programmable controllers

and

STUR1xxxx – Transmitter

STBR1xxxx - Transmitter

Ver. 2.2.2015

Table of Contents

1. Principle of operation of the controller

5

2. Commands for controllers

2.1. System commands

0x32	Program MA0, MA1, MAE, SLO, SL1, SLE	8
0x33	Program baud rate	8
0x34	Add / Remove CN in answer	8

2.2. Commands with number of controller – first set

0x01	Program controller number	9
0x02	Get number of outputs and inputs	9
0x03	Load default parameters	9
0x04	Program output/s number	9
0x05	Program input/s number	9
0x06	Program smart lock data for input/s	10
0x08	Program analog I/O number	10
0x09	Program mode for analog inputs	11
0x0A	Program mode for analog input	11
0x10	Get status of all outputs and inputs for 0-15	12
0x11	Get status of all outputs and inputs for 16-31	12
0x12	Get status of all outputs and inputs for 32-47	12
0x13	Get status of all outputs and inputs for 48-63	13
0x14	Get output/s status	13
0x15	Get input/s status	13
0x16	Get analog I/O status	13
0x17	Set output/s	13
0x18	Set all outputs	14
0x19	Get output number	14
0x1A	Get input number	14
0x1B	Get analog I/O number	14
0x20	Set User Data RAM and Runtime Counter	15
0x21	Get User Data RAM and Runtime Counter	15
0x22	Set User Data EEPROM	15
0x23	Get User Data EEPROM	16
0x24	Set analog output/s	16
0x25	Save/Load EEPROM for analog outputs	17
0x26	Set outputs (from firmware 02)	18

2.3. Direct access commands – second set

0x84	Program output/s number	20
0x85	Program input/s number	20
0x86	Program smart lock data for input/s	20
0x88	Program analog I/O number	21
0x89	Program mode for analog input/s	21
0x94	Get output/s status	22
0x95	Get input/s status	22
0x96	Get analog I/O status	23

0x97	Set output/s	23
0x99	Get number of output	23
0x9A	Get number of input	23
0x9B	Get number of analog I/O	24
0xA4	Set analog output/s	24
0xA5	Save/Load EEPROM for analog output/s	24

3. Transmitter

25

3.1. USB Transmitter

3.1.1. Principle of operation of the transmitter

3.1.2. Commands for transmitter in COM Port mode

26

0x60/0x70	Program MA0, MA1, MAE, SL0, SL1, SLE	28
0x61/0x71	Program baud rate	28
0x62	Set User Data RAM and Runtime Counter	28
0x63	Get User Data RAM and Runtime Counter	28
0x64	Set User Data EEPROM	29
0x65	Get User Data EEPROM	29
0x66	Set USB time reset	29
0x67	Set Restart time	29
0x68/0x78	Set Mode for I/O	31
0x69/0x79	Set Pull-up for I/O	31
0x6A/0x7A	Set Outputs for I/O	31
0x6B	Get Status for I/O	31
0x6C	Temp commands with 0x66, 0x67	31

3.1.3. Commands for transmitter in HID Device mode

33

0x60/0x70	Program MA0, MA1, MAE, SL0, SL1, SLE	36
0x61/0x71	Program baud rate	36
0x62	Set User Data RAM and Runtime Counter	36
0x63	Get User Data RAM and Runtime Counter	36
0x64	Set User Data EEPROM	37
0x65	Get User Data EEPROM	37
0x66	Set USB time reset	37
0x67	Set Restart time	37
0x68/0x78	Set Mode for I/O	38
0x69/0x79	Set Pull-up for I/O	39
0x6A/0x7A	Set Outputs for I/O	39
0x6B	Get Status for I/O	39

Examples

40

3.2. Bluetooth Transmitter

3.2.1. Principle of operation of the transmitter

42

3.2.2. Commands

43

0x60/0x70	Program MA0, MA1, MAE, SL0, SL1, SLE	28
0x61/0x71	Program baud rate	28
0x62	Set User Data RAM and Runtime Counter	28

0x63	Get User Data RAM and Runtime Counter	28
0x64	Set User Data EEPROM	29
0x65	Get User Data EEPROM	29
0x68/0x78	Set Mode for I/O	31
0x69/0x79	Set Pull-up for I/O	31
0x6A/0x7A	Set Outputs for I/O	31
0x6B	Get Status for I/O	31
0x6D	Delete all authenticated devices in the pair list	45
0x6E	Set NEW Pin Code	45
0x6F	Set New Transmitter Name	45

4. Using our test tool – STR1xxxx Assistant 47

4.1. Windows 47

4.2. MacOS 50

*** Serial format is always 1(start), 8(data), 1(stop), no parity . ***

All outputs and inputs start count from 0(zero). First output/input is 0.

1. Principle of operation of the controller

Each controller has its own number, 1-255. This number is stored in one byte. This number can be programmed at any time.

Each controller has multiple inputs and outputs. Each of them has two numbers.

1. A constant number, starting from 0 to as much as is available in current controller. This number is stored in one byte.

Example: If there are 8 outputs, their numbers will be from 0 to 7.

This number is used when applying the commands (**first set**) for accessing specific controller.

Example: Controller 5, output 4.

2. The second number can be programmed and changed at any time. It is used for direct access (**second set**), regardless of controller. This number is stored in two bytes and can be from 1 to 65535(0xFFFF). This number can be duplicated on several controllers. The same number can be assigned to the output and input, as the commands are different.

Example: Output 123.

Care must be taken. When sending a command, that does not require repayment of the data, it can be directly commanded inputs and outputs of multiple controllers. If you need to return a result, it must be only from one controller.

How to connect the controller to network.

1. Set (on/off) jumpers for resistor for RS-485 line
2. Power the controller
3. Connect controller to RS-485 line
4. Program new controller number
5. Program new outputs number for direct access
6. Program new inputs number for direct access
7. Program new analog inputs/outputs number for direct access (if exist)
8. Set, if needed, data for smart lock inputs
9. Set mode (V/A) for analog inputs (if exist)
- 10. Controller is ready for use**
11. You can connect next controller

The format of each command TO controller is as follows:

MA0(master 0/start/), **MA1**(master 1), **BC**(byte count from here to end), **CC**(command), **CN**(controller number), **Data**,..., **Data**, **CS**(check sum), **MAE**(master end)

Or

MA0(master 0/start/), **MA1**(master 1), **BC**(byte count from here to end), **CC**(command), **SNL**(start number LOW byte), **SNH**(start number HIGH byte), **Data**,..., **Data**, **CS**(check sum), **MAE**(master end)

Cleaned

MA0, MA1, BC, CC, CN, Data,..., **Data, CS, MAE** – **first set**

or

MA0, MA1, BC, CC, SNL, SNH, Data,..., **Data, CS, MAE** – **second set**

The format of respond FROM controller is as follows:

SL0(slave/controller 0/start/), **SL1**(slave/controller 1), **BC**(byte count from here to end), **Data**,..., **Data**, **CS**(check sum), **SLE**(slave/controller end)

Cleaned

SL0, SL1, BC, Data,..., **Data, CS, SLE**

Each controller is programmed at manufacture with the following data.

MA0 = 0x55, MA1 = 0xAA, MAE = 0x77

SL0 = 0x56, SL1 = 0xAB, SLE = 0x78

Controller number (CN) = 0xFE.

Outputs number (ON) from 0xFF00.

Inputs number (IN) from 0xFF40. Smart Lock Input = 0x00, disabled.

Analog inputs number (AIN) from 0xFF80. Mode for analog inputs = Voltage.

Analog outputs number from 0xFFC0.

Baud rate FC = 0x05, means 9600.

User Data EEPROM = 0xFF FF FF FF FF FF FF FF

All parameters can be reprogrammed. Come into effect from the next command.

BC (byte count from here to end) – 1 means 1 byte

Example: 55,AA,5,1,5,B,77 – (in HEX)

BC = 5, 5 bytes (5,1,5,B,77)

CS (check sum) – The sum of bytes form BC to before CS.

Example: 55,AA,5,1,5,B,77 – (in HEX)

$CS = 5 + 1 + 5 = 0x0B$, (5,1,5).

If $CS > 0xFF$, only LOW byte is used.

Example: $CS = 0x1234 \rightarrow CS = 0x34$.

When operating, the controller uses a buffer of 40 bytes maximum for each command. While performing a command, it may receive next. Before receiving the last byte of the second command, the first must be finished. If not, then the second will be ignored.

When a command is sent to reprogram something, must be borne in mind, that one byte programming take time approximately 4 milliseconds.

There are built-in protections for the pause between two bytes. The time between two commands can be arbitrary. After receiving the first byte of the command, the maximum time to the next byte must be less than 300 ms. If this time is greater, all received bytes are rejected and the receiving buffer is cleared.

USER DATA EEPROM – 8 bytes

You can use it as you wish.

At power up, controller clears USER DATA RAM and RUNTIME COUNTER.

USER DATA RAM – 4 bytes

RUNTIME COUNTER – 4 bytes

Counter start from 0x00000000 at power up. Every 40mS increments with +1. Real runtime can be calculated by multiply with 40.

2. Commands for controllers

2.1. System commands

0x32 - Program MA0, MA1, MAE, SL0, SL1, SLE

MA0, MA1, 0x0D, 0x32, CN, 0xAA, 0x55, newMA0, newMA1, newMAE, newSL0, newSL1, newSLE, CS, MAE

If CN = 0, that means all controllers. This is very useful, if you want to change the parameters in the network. You do not need to restart controller.

0x33 - Program baud rate

MA0, MA1, 0x08, 0x33, CN, 0xAA, 0x55, newFC, CS, MAE

newFC (0 – 9)

0 – 300

1 – 600

2 – 1200

3 – 2400

4 – 4800

5 – 9600

6 – 19200

7 – 38400

8 – 57600

9 – 115200

If CN = 0, that means all controllers. This is very useful, if you want to change the parameters in the network. You do not need to restart controller.

0x34 – Add / remove CN in answer

MA0, MA1, 0x08, 0x34, CN, 0xAA, 0x55, **Mode**, CS, MAE

This command changes (ANSWER ONLY !) the commands that return an answer from the controller. **Mode** is saved in FLASH. No need to reboot the controller.

/Commands: 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x19, 0x1A, 0x1B, 0x21, 0x23, 0c94, 0c95, 0x96/

CN=0, means all controllers

Mode

<>0 – Old style, no CN in answer

=0 – New style, CN added to answer as fourth byte

Exmple:

Mode <>0, Old style

SL0, SL1, BC, Data,..., Data, CS, SLE

Mode=0, New style

SL0, SL1, BC, **CN**, Data,..., Data, CS, SLE

2.2. Commands with number of controller – first set

0x01 - Program controller number

MA0, MA1, 0x06, 0x01, CN, newCN, CS, MAE

CN = 1 – 0xFF

newCN = 1 – 0xFF

Example: 55,AA,6,1,12,34,4D,77 – Controller 0x12, set new controller number = 0x34.

0x02 - Get number of outputs and inputs – check command 0x34

MA0, MA1, 0x05, 0x02, CN, CS, MAE

CN = 1 – 0xFF

Controller return:

SL0, SL1, 0x09, number of outputs , number of inputs, number of analog inputs, number of analog outputs, 0, 0, CS, SLE

Example: 56,AB,9,8,4,0,0,0,15,78 – 8 outputs, 4 inputs, 0 analog inputs

0x03 - Load default parameters

MA0, MA1, 0x05, 0x03, CN, CS, MAE

CN = 1 – 0xFF

Controller restored to factory settings except CN and USER DATA EEPROM.

0x04 - Program output/s number

MA0, MA1, 0x09, 0x04, CN, start number, number of outputs, new LOW byte, new HIGH byte, CS, MAE

CN = 0 – 0xFF. CN = 0, means all controllers.

start number = 0 – 0x3F

number of outputs = 1 – 0x40

Example: 55,AA,9,4,12,5,4,65,32,BF,77

Controller 0x12,

output 5 -> 0x3265 (direct number)

output 6 -> 0x3266 (direct number)

output 7 -> 0x3267 (direct number)

output 8 -> 0x3268 (direct number)

0x05 - Program input/s number

MA0, MA1, 0x09, 0x05, CN, start number, number of inputs, new LOW byte, new HIGH byte, CS, MAE

CN = 0 – 0xFF. CN = 0, means all controllers.

start number = 0 – 0x3F

number of inputs = 1 – 0x40

Example: 55,AA,9,5,15,2,1,65,3,8E,77

Controller 0x15,

input 2 -> 0x0365 (direct number)

0x06 - Program smart lock data for input/s

Normally, inputs are read at intervals of 40ms. Likely to miss the change of level in short pulses at the input. Especially in large networks. This feature allows you to keep the level of input and extend it. You have two options.

1. Remain level until the input gets read. Bit 6 = 1.
2. Remain level until the input gets read or a specified period of time. Bit 6 = 0.

With bit 5 you can select which level to extend – 0 or 1.

With bit 4-0 you enable/disable this function, or set time. When bit 4-0 = 0, function is disabled.

MA0, MA1, 0x08, 0x06, CN, start number, number of inputs, data smart lock, CS, MAE

CN = 0 – 0xFF. CN = 0, means all controllers.

start number = 0 – 0x3F

number of inputs = 1 – 0x40

data smart lock

bit 7 = 0

bit 6 -> 0 – use time and read, 1 – use read

bit 5 -> 0 – level L (0/LOW), 1 – level H (1/HIGH)

bit 4-0 -> time (1-0x1f), step **400ms**

If bit 4-0 = 0, disable smart lock function.

Example: 55,AA,8,6,15,2,3,23,4B,77

Controller 0x15, input 2,3,4, level H(1), extended for read and for 3*400ms=1.2s

/0x23(0b00100011) -> b6 = 0, b5 = 1, b4-0 = 3 /

0x08 - Program analog I/O number

MA0, MA1, 0x09, 0x08, CN, start number, number of analog I/O, new LOW byte, new HIGH byte, CS, MAE

CN = 0 – 0xFF. CN = 0, means all controllers.

start number = 0 – 0x3F

number of analog I/O = 1 – 0x40

Example: 55,AA,9,8,12,2,3,65,32,BF,77

Controller 0x12,

analog I/O 2 -> 0x3265 (direct number)

analog I/O 3 -> 0x3266 (direct number)

analog I/O 4 -> 0x3267 (direct number)

0x09 - Program mode for analog inputs

MA0, MA1, BC, 0x09, CN, start analog input number, number of analog inputs, mode data, .., CS, MAE

CN = 0 – 0xFF. CN = 0, means all controllers.

analog input number = 0 – 0x3F

number of analog inputs = 1 – 0x20

mode data

One to four byte. Every Bit means 0 – Voltage, 1 – Current

If number of analog inputs = 1 - 8, used only 1 Byte

If number of analog inputs = 9 - 16, used 2 Bytes

If number of analog inputs = 17 - 24, used 3 Bytes

.....

Bits count from low to high.

Example: 55,AA,8,9,12,2,4,3,2C,77

mode data = 3, used first 4 bits -> X,X,X,X,0,0,1,1 (0x03)

Controller 0x12,

analog input 2 -> Current

analog input 3 -> Current

analog input 4 -> Voltage

analog input 5 -> Voltage

Example: 55,AA,9,9,12,2,B,55,2,88,77

mode data = 55, 2, used first 11 bits of 2 bytes

-> 0,1,0,1,0,1,0,1 (0x55) for inputs 2-9 (start from 2)

/ 0(Input 9), 1(input 8), 0(input 7), 1(input 6), 0(input 5), 1(input 4), 0(input 3), 1(input 2) /

-> X,X,X,X,X,0,1,0 (0x02) for inputs 0x0A -0x0C

/ X,X,X,X,X,0(input C), 1(input B), 0(input A) /

analog input 2 -> Current

analog input 3 -> Voltage

analog input 4 -> Current

analog input 5 -> Voltage

analog input 6 -> Current

analog input 7 -> Voltage

analog input 8 -> Current

analog input 9 -> Voltage

analog input A -> Voltage

analog input B -> Current

analog input C -> Voltage

0x0A - Program mode for analog input

MA0, MA1, 0x07, 0x0A, CN, analog input number, mode, CS, MAE

CN = 0 – 0xFF. CN = 0, means all controllers.

analog input number = 0 – 0x3F

mode = 0 – Voltage, 1 – Current

Example: 55,AA,7,A,12,2,0,25,77

Controller 0x12,

analog input 2 -> Voltage

0x10 – Get status of all outputs and inputs for 0 - 15– check command 0x34

MA0, MA1, 0x05, 0x10, CN, CS, MAE

CN = 1 – 0xFF.

Controller return:

SL0, SL1, BC, Low 0, High 0, Low 1, High 1,, CS, SLE

Low, High (2 bytes) ->

High = b7(0/1-Voltage/Current for analog input), b6(status output), b5(status input), X, X, X, b1-b0(analog)

Low = analog

or

V/C, output, input, X, X, X, a9, a8 (high) a7, a6, a5, a4, a3, a2, a1, a0 (low)

a9-a0 = analog value from analog I/O = 0 - 0x3FF

Example (for STR10804 - /8 outputs, 4 inputs/) return:

56,AB,13, 0,40, 0,60 0,0, 0,20, 0,0, 0,0, 0,40, 0,40 53,78

0,40 -> output 0 = 1, input 0 = 0 - (0100 0000=0x40)

0,60 -> output 1 = 1, input 1 = 1 - (0110 0000=0x60)

0,0 -> output 2 = 0, input 2 = 0

0,20 -> output 3 = 0, input 3 = 1

0,0 -> output 4 = 0

0,0 -> output 5 = 0

0,40 -> output 6 = 1

0,40 -> output 7 = 1

0x11, 0x12, 0x13 – Get status of all outputs and inputs for 16 - 63– check command 0x34

Same as 0x10 commands, but return status from 16-63 outputs and inputs.

0x14 – Get output/s status– check command 0x34

MA0, MA1, 0x07, 0x14, CN, start number output, number of outputs, CS, MAE

CN = 1 – 0xFF.

start number = 0 – 0x3F

number of outputs = 1 – 0x20

Controller return:

SL0, SL1, BC, output first, output next,, CS, SLE

Example: 55,AA,7,14,12,2,4,33,77 – Get status from controller 0x12, outputs 2,3,4,5

Return: 56,AB,7,1,0,0,1,9,78

output 2 -> 1

output 3 -> 0

output 4 -> 0

output 5 -> 1

0x15 – Get input/s status– check command 0x34

MA0, MA1, 0x07, 0x15, CN, start number input, number of inputs, CS, MAE

CN = 1 – 0xFF.

start number = 0 – 0x3F

number of inputs = 1 – 0x20

Controller return:

SL0, SL1, BC, input first, input next,, CS, SLE

Example: 55,AA,7,15,12,2,4,34,77 – Get status from controller 0x12, inputs 2,3,4,5

Return: 56,AB,7,1,0,0,1,9,78

input 2 -> 1

input 3 -> 0

input 4 -> 0

input 5 -> 1

0x16 – Get analog I/O status– check command 0x34

MA0, MA1, 0x07, 0x16, CN, start number analog input, number of analog I/O, CS, MAE

CN = 1 – 0xFF.

start number = 0 – 0x3F

number of analog I/O = 1 – 0x10

Controller return:

Check command 0x10, format is the same. Bit 5, 6 set to 0(zero).

Example: 55,AA,7,16,12,2,4,34,77 – Get status from controller 0x12, analog I/O 2,3,4,5

Return: Check command 0x10, format is the same. Bit 5, 6 set to 0(zero). Result is only for selected analog I/O.

0x17 – Set output/s

MA0, MA1, 0x08, 0x17, CN, start number output, number of outputs, 0/1, CS, MAE

CN = 0 – 0xFF. CN = 0, means all controllers.

start number output = 0 – 0x3F

number of outputs = 1 – 0x40

0/1 – level of outputs (off/on)

0x18 – Set all outputs

MA0, MA1, 0x06, 0x18, CN, 0/1, CS, MAE

CN = 0 – 0xFF. CN = 0, means all controllers.

0/1 – level of outputs (off/on)

0x19 – Get output number– check command 0x34

MA0, MA1, 0x06, 0x19, CN, output number, CS, MAE

CN = 1 – 0xFF.

output number = 0 – 0x3F

Controller return:

SL0, SL1, 0x05, output low byte number, output high byte number, CS, SLE

Example: 55,AA,6,19,12,2,33,77 – Get number of 2 output, from controller 0x12

Return: 56,AB,5,34,12,4B,78

output 2 -> 0x1234

0x1A – Get input number– check command 0x34

MA0, MA1, 0x06, 0x1A, CN, input number, CS, MAE

CN = 1 – 0xFF.

input number = 0 – 0x3F

Controller return:

SL0, SL1, 0x05, input low byte number, input high byte number, CS, SLE

Example: 55,AA,6,1A,12,2,34,77 – Get number of 2 input, from controller 0x12

Return: 56,AB,5,34,12,4B,78

input 2 -> 0x1234

0x1B – Get analog I/O number– check command 0x34

MA0, MA1, 0x06, 0x1B, CN, analog I/O number, CS, MAE

CN = 1 – 0xFF.

analog I/O number = 0 – 0x3F

Controller return:

SL0, SL1, 0x05, analog I/O low byte number, analog I/O high byte number, CS, SLE

Example: 55,AA,6,1B,12,2,3577 – Get number of 2 analog I/O, from controller 0x12

Return: 56,AB,5,34,12,4B,78

analog I/O 2 -> 0x1234

0x20 – Set User Data RAM and Runtime Counter

MA0, MA1, 0x0E, 0x20, CN, type, rc0, rc1, rc2, rc3, udr0, udr1, udr2, udr3, CS, MAE

CN = 1 – 0xFF.

type

Bit 0=1 – Set Runtime Counter (RC)

Bit 1=1 – Set User Data RAM (UDR)

rc0, rc1, rc2, rc3 (LSB – MSB) – runtime counter

udr0, udr1, udr2, udr3 (LSB – MSB) – user data ram

Example:

Set RC with 0x12448892, for controller 0xFE

55, AA, 0E, 20, FE, 1, 92, 88, 44, 12, X, X, X, X, CS, 77

Set UDR with 0x23568419, for controller 0xFE

55, AA, 0E, 20, FE, 2, X, X, X, X, 19, 84, 56, 23, CS, 77

Set RC and UDR, for controller 0xFE

55, AA, 0E, 20, FE, 3, 92, 88, 44, 12, 19, 84, 56, 23, CS, 77

0x21 –Get User Data RAM and Runtime Counter– check command 0x34

MA0, MA1, 0x05, 0x21, CN, CS, MAE

CN = 1 – 0xFF.

Controller return:

SLO, SL1, 0x0B, rc0, rc1, rc2, rc3, udr0, udr1, udr2, udr3, CS, SLE

rc0, rc1, rc2, rc3 (LSB – MSB) – runtime counter

udr0, udr1, udr2, udr3 (LSB – MSB) – user data ram

0x22 – Set User Data EEPROM

MA0, MA1, 0x0D, 0x22, CN, ude0, ude1, ude2, ude3, ude4, ude5, ude6, ude7, CS, MAE

CN = 1 – 0xFF.

ude0,... ude7 – user data eeprom

0x23 – Get User Data EEPROM– check command 0x34

MA0, MA1, 0x05, 0x23, CN, CS, MAE

CN = 1 – 0xFF.

Controller return:

SL0, SL1, 0x0B, ude0, ude1, ude2, ude3, ude4, ude5, ude6, ude7, CS, SLE

ude0,.. ude7 – user data eeprom

0x24 – Set Analog Output/s

MA0, MA1, BC, 0x24, CN, Type, Low, High, Speed, LB, HB, CNx, ..., CS, MAE

CN = 0 – 0xFF. CN = 0, means all controllers.

Speed = 1 - 255

LB, HB = 16 bits for every analog outputs. To select output, bit must be set = 1.

Example:

To select output: 1, 3, 4 -> 0000 0000 0001 1010 -> LB=0x1A, HB=0x00.

To select output: 2, 4, 7, 9 -> 0000 0010 1001 0100 -> LB=0x94, HB=0x02.

(Outputs count from 0(zero)!)

Analog Outputs use 10bit DAC. Value must be between 0 – 0x3FF.

Low, High (2 bytes) ->

High = b7, b6, X, X, X, X, b1-b0 (analog), Low = analog

or

b7, b6, X, X, X, X, a9, a8 (high) a7, a6, a5, a4, a3, a2, a1, a0 (low)

a9-a0 = analog value for analog output = 0 - 0x3FF

b7, b6

> 0, 0 – Set new value

> 0, 1 – Set only new speed

> 1, 0 – Set new value with old Speed

> 1, 1 – Set new value with new Speed

Analog Outputs are refreshed every 40mS. Speed mean how much to add every interval to the current value. If the current value is greater than the new value, this number will be **subtracted**.

For Example:

Speed = 4 -> every 40mS, 4 will be added to current value of analog output.

If Current value is 0x000, and you set new value to 0x3FF, new value would be reached after

(0x3FF - 0x000) / 4 = 0x100(intervals) * 40mS = 10.24 sec

So, the new value will be reached after 10.24 seconds.

Type – different format of the command.

Type=0

MA0, MA1, BC, 0x24, CN, 0, Low, High, Speed, LB, HB, CS, MAE

Use this format to select single controller with CN.

Type =1

MA0, MA1, BC, 0x24, CN, 1, Low, High, Speed, LB, HB, CN(end), CS, MAE

Use this format to select controllers with CN to CN(end).

CN(end)>CN!

For example 2, 3, 4, 5, 6 -> CN = 2, CN(end) = 6.

Type =2-16

MA0, MA1, BC, 0x24, CN, Type, Low, High, Speed, LB, HB, CN1, ..CNx, CS, MAE

Use this format to select up to 16 controllers with CN, CN1, CN2,.....

For example 2, 5, 6, 34, 67, 12 -> CN = 2, CN1 = 5, CN2 = 6, CN3 = 34, CN4 = 67, CN5 = 12. Type =6.

* * * * *

Example 1: Set outputs 2, 3, 4 to 0x123, without speed for controller with CN=0x12.

MA0, MA1, BC, 24, 12, 0, 23, 1, X, 1C, 0, CS, 77

Example 2: Set outputs 5, 6 to 0x230, with new speed 8 for controller with CN=0x12.

MA0, MA1, BC, 24, 12, 30, C2, 8, 60, 0, CS, 77

30, C2 -> C2(high) 1, 1, 0, 0, 0, 0, 1, 0 , 30(low) 0x30

1,1 - b7, b6

1,0 , 30(low) 0x30 -> 0x230

Example 3: Set outputs 1, 6, 9 to 0x230, with new speed 8 for controllers with CN=0x12, 0x23, 0x34.

MA0, MA1, BC, 24, 12, 3, 30, C2, 8, 42, 2, 23, 34, CS, 77

Example 4: Set outputs 3, 4, 7 to 0x256, with old speed for controllers with CN=0x12 ...0x56.

MA0, MA1, BC, 24, 12, 1, 56, 82, X, 98, 0, 56, CS, 77

56, 82 -> 82 (high) 1, 0, 0, 0, 0, 0, 1, 0 , 56(low) 0x56

1, 0 - b7, b6

1, 0 , 56(low) 0x56 -> 0x256

0x25 - Save/Load EEPROM for Analog Outputs

MA0, MA1, BC, 0x25, CN, Type, LB, HB, Command, CNx,....., CS, MAE

CN = 0 - 0xFF. CN = 0, means all controllers.

LB, HB = 16 bits for every analog outputs. To select output, bit must be set = 1.

Example:

To select output: 1, 3, 4 -> 0000 0000 0001 1010 -> LB=0x1A, HB=0x00.

To select output: 2, 4, 7, 9 -> 0000 0010 1001 0100 -> LB=0x94, HB=0x02.

(Outputs count from 0(zero)!)

Command:

b7 - 0 = without speed, 1 = with speed <- ONLY for SAVE

b0 - 0 = Save, 1 = Load

With a speed means, that the set value will be reached gradually, with the set speed at power up.

Type – different format of the command.

Type=0

MA0, MA1, BC, 0x25, CN, 0, LB, HB, Command, CS, MAE

Use this format to select single controller with CN.

Type=1

MA0, MA1, BC, 0x25, CN, 1, LB, HB, Command, CN(end), CS, MAE

Use this format to select controllers with CN to CN(end).

CN(end)>CN!

For example 2, 3, 4, 5, 6 -> CN = 2, CN(end) = 6.

Type=2-16

MA0, MA1, BC, 0x25, CN, Type, LB, HB, Command, CN1, ..CNx, CS, MAE

Use this format to select up to 16 controllers with CN, CN1, CN2,.....

For example 2, 5, 6, 34, 67, 12 -> CN = 2, CN1 = 5, CN2 = 6, CN3 = 34, CN4 = 67, CN5 = 12. Type = 6.

At power up, controller loads the initial values of analog outputs from EEPROM. With this command you can save the current values of as initial. At any time, you can load the values recorded as current.

When a command is sent to program something, must be borne in mind, that one byte programming take time approximately 4 milliseconds. Every analog output use 3 byte for this command.

Example: Save current value for outputs 3, 5, 9 with speed for controller with CN=0x22 ... 0x33.

MA0, MA1, BC, 25, 22, 1, 24, 2, 80, 33, CS, 77

80 -> b7=1, b0 = 0. Type = 1.

0x26 – Set outputs

MA0, MA1, BC, 0x26, CN, start output number, number of outputs, data, .., CS, MAE

CN = 0 – 0xFF. CN = 0, means all controllers.

output number = 0 – 0x3F

number of outputs = 1 – 0x20

data

One to four byte. Every Bit means 0 –OFF, 1 – ON

If number of outputs = 1 - 8, used only 1 Byte

If number of outputs = 9 - 16, used 2 Bytes

If number of outputs = 17 - 24, used 3 Bytes

If number of outputs = 25 - 32, used 4 Bytes

Bits count from low to high.

Example: Set outputs 3-ON, 4-OFF, 5-OFF, 6-ON, 7-ON for controller with CN=0x22. So 3..7-> 5 outputs
>55,AA,08,26,22,03,05,19,71,77 0x19(hex)=0b00011001(bin) -> 0bXXX 1(7) 1(6) 0(5) 0(4) 1(3)

Example: Set outputs 0...3-ON, 4, 5-OFF, 6, 7-ON, 8...10-OFF, 11-ON for controller with CN=0x22.

So 0...11-> 12 outputs

>55,AA,09,26,22,0,0C,CF,8,30,77

0xCF(hex)=0b11001111(bin) -> 0b1(7) 1(6) 0(5) 0(4) 1(3) 1(2) 1(1) 1(0)

0x08(hex)=0b00001000(bin) -> 0bXXXX 1(11) 0(10) 0(9) 0(8)

2.3. Direct access commands – second set

0x84 - Program output/s number

MA0, MA1, 0x0A, 0x84, SNL, SNH, number of outputs L, H, start new outputs number L, H, CS, MAE

SNL, SNH = 1 – 0xFFFF.

number of outputs L, H = 1 – 0xFFFF. /SNL, SNH + number of outputs L, H < 0x10000/

start new outputs number L, H = 1 – 0xFFFF. /start new outputs number L, H + number of outputs L, H < 0x10000/

Example: 55,AA,A,84,34,12,3,0,42,36,4F,77

3,0 -> 3 outputs

(old)output 0x1234 -> (new)0x3642

(old)output 0x1235 -> (new)0x3643

(old)output 0x1236 -> (new)0x3644

0x85 - Program input/s number

MA0, MA1, 0x0A, 0x85, SNL, SNH, number of inputs L, H, start new inputs number L, H, CS, MAE

SNL, SNH = 1 – 0xFFFF.

number of inputs L, H = 1 – 0xFFFF. /SNL, SNH + number of inputs L, H < 0x10000/

start new inputs number L, H = 1 – 0xFFFF. /start new inputs number L, H + number of inputs L, H < 0x10000/

Example: 55,AA,A,85,34,12,2,0,42,36,4F,77

2,0 -> 2 inputs

(old)input 0x1234 -> (new)0x3642

(old)input 0x1235 -> (new)0x3643

0x86 - Program smart lock data for input/s

Normally, inputs are read at intervals of 40ms. Likely to miss the change of level in short pulses at the input. Especially in large networks. This feature allows you to keep the level of input and extend it. You have two options.

3. Remain level until the input gets read. Bit 6 = 1.
4. Remain level until the input gets read or a specified period of time. Bit 6 = 0.

With bit 5 you can select which level to extend – 0 or 1.

With bit 4-0 you enable/disable this function, or set time. When bit 4-0 = 0, function is disabled.

MA0, MA1, 0x09, 0x86, SNL, SNH, number of inputs L, H, data smart lock, CS, MAE

SNL, SNH = 1 – 0xFFFF.

number of inputs L, H = 1 – 0xFFFF. /SNL, SNH + number of inputs L, H < 0x10000/

data smart lock

bit 7 = 0

bit 6 -> 0 – use time and read, 1 – use read

bit 5 -> 0 – level L (0/LOW), 1 – level H (1/HIGH)

bit 4-0 -> time (1-0x1f), step **400ms**

If bit 4-0 = 0, disable smart lock function.

Example: 55,AA,9,86,34,12,2,0,23,FA,77

input 0x1234 and input 0x1235 = level H(1), extended for read and for 3***400ms**=1.2s

/0x23(0b00100011) -> b6 = 0, b5 = 1, b4-0 = 3 /

0x88 - Program analog I/O number

MA0, MA1, 0x0A, 0x88, SNL, SNH, number of analog I/O L, H, start new analog I/O number L, H, CS, MAE

SNL, SNH = 1 – 0xFFFF.

number of analog I/O L, H = 1 – 0xFFFF. /SNL, SNH + number of analog I/O L, H < 0x10000/

start new analog I/O number L, H = 1 – 0xFFFF.

/start new analog I/O number L, H + number of analog I/O L, H < 0x10000/

Example: 55,AA,A,88,34,12,2,0,42,36,52,77

2,0 -> 2 analog I/O

(old)analog I/O 0x1234 -> (new)0x3642

(old)analog I/O 0x1235 -> (new)0x3643

0x89 - Program mode for analog inputs

MA0, MA1, BC, 0x89, SNL, SNH, number of analog inputs, mode data, ..., CS, MAE

SNL, SNH = 1 – 0xFFFF.

number of analog inputs = 1 – 0x20

mode data

One to four byte. Every Bit means 0 – Voltage, 1 – Current

If number of analog inputs = 1 - 8, used only 1 Byte

If number of analog inputs = 9 - 16, used 2 Bytes

If number of analog inputs = 17 - 24, used 3 Bytes

.....

Bits count from low to high.

Example: 55,AA,8,89,34,12,4,3,DE,77

mode data = 3, used first 4 bits -> X,X,X,X,0,0,1,1 (0x03)

analog input 0x1234 -> Current

analog input 0x1235 -> Current

analog input 0x1236 -> Voltage

analog input 0x1237 -> Voltage

Example: 55,AA,9,89,34,12,B,55,2,3A,77

mode data = 55, 2, used first 0x0B bits of 2 bytes

-> 0,1,0,1,0,1,0,1 (0x55) for inputs 0x1234-0x123B
/ 0(Input 0x123B), 1, 0, 1, 0, 1, 0, 1(input 0x1234) /

-> X,X,X,X,X,0,1,0 (0x02) for inputs 0x123C -0x123E
/ X,X,X,X,X,0(input 0x123E), 1, 0(input 0x123C)/

analog input 0x1234 -> Current
analog input 0x1235 -> Voltage
analog input 0x1236 -> Current
analog input 0x1237 -> Voltage
analog input 0x1238 -> Current
analog input 0x1239 -> Voltage
analog input 0x123A -> Current
analog input 0x123B -> Voltage
analog input 0x123C -> Voltage
analog input 0x123D -> Current
analog input 0x123E -> Voltage

0x94 – Get output/s status– check command 0x34

MA0, MA1, 0x07, 0x94, SNL, SNH, number of outputs, CS, MAE

SNL, SNH = 1 – 0xFFFF.

number of outputs = 1 – 0x20 /SNL, SNH + number of outputs < 0x10000/

Controller return:

SL0, SL1, BC, output first ,output next,, CS, SLE

Example: 55,AA,7,94,34,12,4,E5,77 – Get status for outputs 0x1234 – 0x1237

Return: 56,AB,7,1,0,0,1,9,78

output 0x1234 -> 1

output 0x1235 -> 0

output 0x1236 -> 0

output 0x1237 -> 1

0x95 – Get input/s status– check command 0x34

MA0, MA1, 0x07, 0x95, SNL, SNH, number of inputs, CS, MAE

SNL, SNH = 1 – 0xFFFF.

number of inputs = 1 – 0x20 /SNL, SNH + number of inputs < 0x10000/

Controller return:

SL0, SL1, BC, input first, input next,, CS, SLE

Example: 55,AA,7,95,34,12,4,E6,77 – Get status for inputs 0x1234 – 0x1237

Return: 56,AB,7,1,0,0,1,9,78

input 0x1234 -> 1

input 0x1235 -> 0

input 0x1236 -> 0

input 0x1237 -> 1

0x96 – Get analog I/O status– check command 0x34

MA0, MA1, 0x07, 0x96, SNL, SNH, number of analog I/O, CS, MAE

SNL, SNH = 1 – 0xFFFF.

number of analog I/O = 1 – 0x10 / SNL, SNH + number of analog I/O < 0x10000/

Controller return:

Check command 0x10, format is the same. Bit 5, 6 set to 0(zero).

Example: 55,AA,7,96,34,12,4,E7,77 – Get status for analog I/O 0x1234 – 0x1237

Return: Check command 0x10, format is the same. Bit 5, 6 set to 0(zero). Result is only for selected analog I/O.

0x97 – Set output/s

MA0, MA1, 0x09, 0x97, SNL, SNH, number of outputs L, H, 0/1, CS, MAE

SNL, SNH = 1 – 0xFFFF.

number of outputs L, H = 1 – 0xFFFF / SNL, SNH + number of outputs L, H < 0x10000/

0/1 – level of outputs (off/on)

0x99 – Get output number

MA0, MA1, 0x06, 0x99, SNL, SNH, CS, MAE

SNL, SNH = 1 – 0xFFFF.

Controller return:

SL0, SL1, 0x05, controller number, output number, CS, SLE

Example: 55,AA,6,99,34,12,E5,77 – Get controller number and output number for output 0x1234

Return: 56,AB,5,8,2,F,78

Controller number = 8, output number = 2.

0x9A – Get input number

MA0, MA1, 0x06, 0x9A, SNL, SNH, CS, MAE

SNL, SNH = 1 – 0xFFFF.

Controller return:

SL0, SL1, 0x05, controller number, input number, CS, SLE

Example: 55,AA,6,9A,34,12,E6,77 – Get controller number and input number for input 0x1234

Return: 56,AB,5,8,2,F,78

Controller number = 8, input number = 2.

0x9B – Get analog I/O number

MA0, MA1, 0x06, 0x9B, SNL, SNH, CS, MAE

SNL, SNH = 1 – 0xFFFF.

Controller return:

SL0, SL1, 0x05, controller number, I/O number, CS, SLE

Example: 55,AA,6,9B,34,12,E7,77 – Get controller number and analog I/O number for analog I/O 0x1234

Return: 56,AB,5,8,2,F,78

Controller number = 8, analog I/O number = 2.

0xA4 – Set analog output/s

MA0, MA1, 0x09, 0xA4, SNL, SNH, number of outputs L, H, Low, High, Speed, CS, MAE

SNL, SNH = 1 – 0xFFFF.

number of outputs L, H = 1 – 0xFFFF / SNL, SNH + number of outputs L, H < 0x10000/

Please refer to the description of the command **0x24** for explanation of the other parameters (Low, High, Speed).

0xA5 – Save/Load EEPROM for analog output/s

MA0, MA1, 0x09, 0xA5, SNL, SNH, number of outputs L, H, Command, CS, MAE

SNL, SNH = 1 – 0xFFFF.

number of outputs L, H = 1 – 0xFFFF / SNL, SNH + number of outputs L, H < 0x10000/

Please refer to the description of the command **0x25** for explanation of the Command parameter.

3. Transmitter

3.1. USB Transmitter

3.1.1. Principle of operation of the transmitter

Please note in this file is described only software application on the transmitter. For specific hardware configuration, please download the PDF file for the hardware of transmitter.

http://smarthardware.eu/manual/stur108_doc.pdf

The transmitter is with parallel processing. While receiving / transmitting through the serial port, it can receive and transmit other commands through USB port. Supports full duplex for serial port, if you do not use part with SN75176 RS-485 transmitter. There are two independent buffers (256 bytes each) for transmission / reception via the serial port. When the transmitter accepts command to write to EEPROM, the following commands are blocked until the current one. Must be borne in mind, that one byte programming take time approximately 4 milliseconds. During writing, it continues to process the serial port and can send/receive through the USB port.

If you disable all the commands (JC0-off, JC1..JC5 – on), transmitter works as a simple transmitter (USB to Serial to RS-485), similar to those with FTDI and etc.

JM – select mode **COM** (set) /**HID** (removed).

JC0 .. JC5 – enable/disable commands.

JD – Clear all config, except EEPROM.

Transmitter can operate in two modes (COM / HID), selected with a jumper **JM**. This jumper is checked only at the start of the transmitter (Power Up or Restart).

JC0..JC5 checked dynamically upon receiving the command. Please see the detail in the description of the two modes.

JD jumper is checked only at the start of the transmitter (Power Up or Restart).

How to use:

Disconnect power. Put jumper. Connect power (or put Jumper and restart transmitter).The blue LED will light constantly. Now transmitter waits to remove jumper. Remove jumper. The transmitter will continue in selected with **JM** mode. The blue LED will flash briefly.

3.1.2. Commands for transmitter in COM Port mode

		JC	ON	OFF
	Load baud rate upon opening port from computer	0	Disable	Enable
0x60/0x70	Program MA0, MA1, MAE, SL0, SL1, SLE	1	Disable	Enable
0x61/0x71	Program baud rate	2	Disable	Enable
0x62	Set User Data RAM and Runtime Counter	3	Disable	Enable
0x63	Get User Data RAM and Runtime Counter		Disable	Enable
0x64	Set User Data EEPROM		Disable	Enable
0x65	Get User Data EEPROM		Disable	Enable
0x66	Set USB time reset	4	Disable	Enable
0x67	Set Restart time		Disable	Enable
0x68/0x78	Set Mode for I/O	5	Disable	Enable
0x69/0x79	Set Pull-up for I/O		Disable	Enable
0x6A/0x7A	Set Outputs for I/O		Disable	Enable
0x6B	Get Status for I/O		Disable	Enable
0x6C	Temp commands with 0x66, 0x67			

Some commands have two versions. Second means EEPROM version.

Example: 0x60 / 0x70

0x60 – **RAM** version. The result is valid until the transmitter is restarted. At startup are loaded the parameters from EEPROM.

0x70 – **EEPROM** version. Same as above, but is saved in EEPROM. At startup will be loaded same parameters.

In this mode, the transmitter uses commands similar to the other controllers. The length of the command is set to **14** bytes. The result which returns is always **13** bytes.

When you send a command that returns a result, it is always added last to the internal buffer data received from the serial port.

Example:

- If you receive 5 bytes, this means that the result has not been obtained in the computer and need to do an additional one reading.
- If you get 13 bytes, it means that you have read exactly the result.
- If you receive 20 bytes, it means that you have read 7 bytes from the serial port (the first) and 13 bytes (the last) the result of the command.

Jumpers JC0..JC5 are checked dynamically upon receiving the command.

In order to recognize transmitter a command, exactly **14** bytes must to be sent at once. If you send 2 times with 7 bytes for example, they will not be recognized as a command, and will be sent through the serial port. Or if you send more than 14 bytes.

If a command is recognized, but it is disabled, then these bytes will be ignored and will not be sent to the serial port.

If you disable ALL commands (JC0-off, JC1..JC5 - on), then everything, including the received commands will be sent to the serial port.

The format of each command TO transmitter is as follows:

MA0(master 0/start/), **MA1**(master 1), **0x0C**, **CC**(command), **Data0**,..., **Data7**, **CS**(check sum), **MAE**(master end)

Cleaned

MA0, MA1, 0x0C, CC, Data0,..., Data7, CS, MAE

The format of respond FROM transmitter is as follows:

SL0(slave/controller 0/start/), **SL1**(slave/controller 1), **0x0B**, **Data0**,..., **Data7**, **CS**(check sum), **SLE**(slave/controller end)

Cleaned

SL0, SL1, 0x0B, Data,..., Data, CS, SLE

For **CS** check page 6.

Each controller is programmed at manufacture with the following data.

MA0 = 0x55, MA1 = 0xAA, MAE = 0x77

SL0 = 0x56, SL1 = 0xAB, SLE = 0x78

Baud rate FC = 0x05, means 9600.

I/O – Inputs with Pull-up enabled.

User Data EEPROM = 0xFF FF FF FF FF FF FF FF

All parameters can be reprogrammed. Come into effect from the next command.

USER DATA EEPROM – 8 bytes

You can use it as you wish.

At power up, controller clears USER DATA RAM and RUNTIME COUNTER.

USER DATA RAM – 4 bytes

RUNTIME COUNTER – 4 bytes

Counter start from 0x00000000 at power up. Every 40mS increments with +1. Real runtime can be calculated by multiply with 40.

0x60, 0x70 - Program MA0, MA1, MAE, SLO, SL1, SLE

MA0, MA1, 0x0C, 0x60, newMA0, newMA1, newMAE, newSLO, newSL1, newSLE, 0,0,CS, MAE

You do not need to restart controller.

0x61, 0x71 - Program baud rate

MA0, MA1, 0x0C, 0x61, newFC, 0,0,0,0,0,0, CS, MAE

newFC (0 – 9)

0 – 300

1 – 600

2 – 1200

3 – 2400

4 – 4800

5 – 9600

6 – 19200

7 – 38400

8 – 57600

9 – 115200

You do not need to restart controller.

0x62 – Set User Data RAM and Runtime Counter

MA0, MA1, 0x0C, 0x62, rc0, rc1, rc2, rc3, udr0, udr1, udr2, udr3, CS, MAE

rc0, rc1, rc2, rc3 (LSB – MSB) – runtime counter

udr0, udr1, udr2, udr3 (LSB – MSB) – user data ram

If you set all 4 bytes (0xFF, 0xFF, 0xFF, 0xFF) for RC and/or UDR, then this part will not be changed.

Example:

Set RC with 0x12448892

55, AA, 0C, 62, 92, 88, 44, 12, FF, FF, FF, FF, DA, 77

0x63 –Get User Data RAM and Runtime Counter

MA0, MA1, 0x0C, 0x63, 0, 0, 0, 0, 0, 0, 0, 0, CS, MAE

Controller return:

SLO, SL1, 0x0B, rc0, rc1, rc2, rc3, udr0, udr1, udr2, udr3, CS, SLE

rc0, rc1, rc2, rc3 (LSB – MSB) – runtime counter

udr0, udr1, udr2, udr3 (LSB – MSB) – user data ram

0x64 – Set User Data EEPROM

MA0, MA1, 0x0C, 0x64,, ude0, ude1, ude2, ude3, ude4, ude5, ude6, ude7, CS, MAE

ude0,.. ude7 – user data eeprom

0x65 –Get User Data EEPROM

MA0, MA1, 0x0C, 0x65, 0, 0, 0, 0, 0, 0, 0, 0, CS, MAE

Controller return:

SL0, SL1, 0x0B, ude0, ude1, ude2, ude3, ude4, ude5, ude6, ude7, CS, SLE

ude0,.. ude7 – user data eeprom

0x66 – Set USB time reset, 0x67 – Set Restart time, 0x6C

Both commands work similarly.

The first, restart USB connection after **Time_usb**, the second - restart controller after **Time_reset**.

- Upon receipt/transmit over USB both times are restarts.
- Both commands and times are independent of each other.
- At anytime you can send a command to set new time.
- To **disable** function set 0x00 to **Time_usb** or **Time_reset**.

There is a peculiarity that is associated with USB (CDC device) standard and not to the particular transmitter.

Send to serial port.

When you send something from your program to the serial port, this happens consistently.

Program->USB buffer in computer->USB port->Transmitter USB port->Internal buffer->Serial port

In this case both timers will be restarted. Everything is OK.

Read from serial port.

When you read from serial port, in fact you only read from the internal buffer into the computer. Transfer between the transmitter and the computer takes place in the background. Once you have taken something from the serial port in the internal buffer of the transmitter, it is immediately sent to the computer as soon as possible AND both timers will be restarted. BUT if there's nothing to send to the computer, will NOT be a transfer and timers will NOT be reset. If the buffer is empty in the computer, no matter how often you read it, there will be no transfer through USB cable. If there is nothing to receive, but you want to reset the timers, you need to send something to the transmitter. In this case, the command **0x6C** is used. This command is empty, and only appear when at least one of the timers is different from zero. In all other cases, this command does not exist. This command cannot be disabled in any way.

MA0, MA1, 0x0C, 0x66, Time_usb, Wait_usb , 0, 0, 0, 0, 0, 0, CS, MAE

Time_USB - After how much time to restart the USB connection, In seconds.

1 - means 1 second.

0 - disables this function. This is set at power up controller. After rebooting, also set to 0.

0xFF - perform immediately restart connection.

Wait_usb - How long to wait before attach it back to USB. In the 40ms interval.

$0x19 = 25 * 40\text{mS} = 1\text{sek}$

Generally not good for set time under one second as the computer cannot detect disconnecting and take it as interference. Optimum is 1.4-1.5 seconds.

Example: 55, AA, 0C, 66, 0x2, 0x23, 0, 0, 0, 0, 0, 97, 78 – After 2 sek disconnect, wait 1.4 sek and **connect** again.

MA0, MA1, 0x0C, 0x67, Time_reset, Wait_reset , 0, 0, 0, 0, 0, 0, CS, MAE

Time_reset - After how much time to reset controller, In seconds.

1 - means 1 second.

0 - disables this function. This is set at power up controller. After rebooting, also set to 0.

0xFF - perform immediately reboot.

Wait_reset - How long to wait before to reset controller. In the 40ms interval.

$0x19 = 25 * 40\text{mS} = 1\text{sek}$

Generally not good for set time under one second as the computer cannot detect disconnecting and take it as interference. Optimum is 1.4-1.5 seconds.

Example: 55, AA, 0C, 66, 0x2, 0x23, 0, 0, 0, 0, 0, 97, 78 – After 2 sek disconnect, wait 1.4 sek and **reset** controller.

MA0, MA1, 0x0C, 0x6C, 0xC6, 0, 0, 0, 0, 0, 0, 9, CS, MAE – Empty commands

Tips

It makes no sense to set **Time_USB > Time_reset**, because after reboot both times were set to 0 and functions are disabled.

Generally no sense to use both commands simultaneously. It is not desirable both commands have a similar time, because after the first execution will immediately execute the second and the computer may be confused.

Before closing the port, finally, turn off these commands by sending zero times.

Recommended method of use:

1. Open port.
2. Send 0x66 command.
3. Work usual.
4. If an error occurs and the transfer block over USB, **immediately close** the port and wait for the preset time to expires. It is important to break the connection when the port is closed.
5. Go to 1.

0x68, 0x78 - Set Mode for I/O

MA0, MA1, 0x0C, 0x68, data, 0, 0, 0, 0, 0, 0, 0, CS, MAE

With this command selects each line that will be input or output.

1- Input, 0 – Output.

The controller STUR108 has 8 lines. It is used only one byte of the command. Each bit corresponds to the line. Bit 0 for line 0, ... bit 7 for line 7.

Example:

55, AA, 0C, 68, 0x33, 0,0,0,0,0,0,A7,78

Data – 0x33 (0b00110011), Line 0,1,4,5 – Input, Line 2,3,6,7 – Output.

0x69, 0x79 - Set Pull-Up for I/O

MA0, MA1, 0x0C, 0x69, data, 0, 0, 0, 0, 0, 0, 0, CS, MAE

With this command you select Pull-Up for each line, which is set as input.

If line is set as output, and pull-up is enabled, when line is switched to input, pull-up will be automatically enabled.

1- Enable, 0 – Disable.

The controller STUR108 has 8 lines. It is used only one byte of the command. Each bit corresponds to the line. Bit 0 for line 0, ... bit 7 for line 7.

0x6A, 0x7A - Set Outputs H/L for I/O

MA0, MA1, 0x0C, 0x6A, data, 0, 0, 0, 0, 0, 0, 9, CS, MAE

With this command you select level for each line, which is set as output.

If line is set as input, when line is switched to output, level will be chosen from this info.

1- High, 0 – Low.

The controller STUR108 has 8 lines. It is used only one byte of the command. Each bit corresponds to the line. Bit 0 for line 0, ... bit 7 for line 7.

0x6B –Get Status for I/O

MA0, MA1, 0x0C, 0x6B, 0, 0, 0, 0, 0, 0, 0, CS, MAE

With this command you get level for each line.

Controller return:

SL0, SL1, 0x0B, Data, 0, 0, 0, 0, 0, 0, 0, CS, SLE

The controller STUR108 has 8 lines. It is used only one byte of the command. Each bit corresponds to the line. Bit 0 for line 0, ... bit 7 for line 7.

3.1.3. Commands for transmitter in HID device mode

		JC	ON	OFF
	Echo mode	0	Enable	Disable
/0x70	Program MA0, MA1, MAE, SL0, SL1, SLE	1	Disable	Enable
0x61/0x71	Program baud rate	2	Disable	Enable
0x62	Set User Data RAM and Runtime Counter	3	Disable	Enable
0x63	Get User Data RAM and Runtime Counter		Disable	Enable
0x64	Set User Data EEPROM		Disable	Enable
0x65	Get User Data EEPROM		Disable	Enable
0x66	Set USB time reset	4	Disable	Enable
0x67	Set Restart time		Disable	Enable
0x68/0x78	Set Mode for I/O	5	Disable	Enable
0x69/0x79	Set Pull-up for I/O		Disable	Enable
0x6A/0x7A	Set Outputs for I/O		Disable	Enable
0x6B	Get Status for I/O		Disable	Enable

Some commands have two versions. Second means EEPROM version.

Example: 0x61 / 0x71

0x61– **RAM** version. The result is valid until the transmitter is restarted. At startup are loaded the parameters from EEPROM.

0x71 – **EEPROM** version. Same as above, but is saved in EEPROM. At startup will be loaded same parameters.

This mode is different from COM Port and is related to the operation of HID Devices.

1. Computer is Host, and transmitter is Client. All transactions are initiated from the Host. You always send something to transmitter and always get something. The controller cannot transfer data to the computer itself, there must be a request from the computer.

2. Another important thing in the standard is that nowhere does it indicate how many bytes are transferred. The controller cannot determine how many bytes were received. The same goes for the computer, how many bytes were received.

For this is used the following thing:

1. It is used a buffer of 64 bytes for transfer between the computer and the transmitter.
2. The first byte is the number of bytes transferred.
3. The transmitter always returns 64 bytes.
4. If you send a command to the controller, **first byte is 0x50 - fixed**. You will receive a response in which the first byte is the number of transmitted command.
 - 4.1. If first byte of response is **0xFD**, this command is disabled.
 - 4.2. If first byte of response is **0xFF**, invalid command.
 - 4.3. If first byte of response is **0xFE**, transmitter is busy (writing to EEPROM).

5. If you send bytes to the serial port, **first byte is 0x01 – 0x3F**. You will receive an answer with bytes received from the serial port.
6. If you only need to read bytes from the serial port, **first byte is 0x00**. You will receive an answer with bytes received from the serial port.

VERY IMPORTANT

For different operating systems, the buffer you are using in your software is treated differently.

In **MacOS**, simply reserve buffer with length of 64 bytes for transfer. Strictly speaking two independent buffers, with 64 bytes for send / receive.

In **Windows** things are otherwise. If you use **WinApi**, you must add one more byte in the beginning, which is always **0**/zero/. I.e. the size of the buffers should be 65 bytes. The first byte indicates which endpoint is used for transfer for **WinApi**. Please check whether you need to add a start byte, according to the language which is used to create your software.

Always is used endpoint 0.

Transmitter uses:

Vid=0x04d9

Pid=0x0005

Search for HID devices with these vid/pid . Then send a command. If you receive a valid response, this is your transmitter. Our software uses the command 0x65 – get User data EEPROM. It is displayed in the field between Rescan and Connect buttons.

Echo mode – If this mode is selected, all transfer to/from serial port is disabled. All commands are disabled. Everything that received in the controller from USB port is returned in the same form, without any modification and processing. This is useful to setup the software. So you can check what receives transmitter.

Please see the examples given at the end for easier understanding of the entire transfer.

Each controller is programmed at manufacture with the following data.

MA0 = **0x55**, MA1 = **0xAA**, MAE = **0x77** – used for COM PORT Mode

SL0 = **0x56**, SL1 = **0xAB**, SLE = **0x78** – used for COM PORT Mode

Baud rate FC = **0x05**, means 9600.

I/O – Inputs with Pull-up enabled.

User Data EEPROM = 0xFF FF FF FF FF FF FF FF

All parameters can be reprogrammed. Come into effect from the **next** command.

USER DATA EEPROM – 8 bytes

You can use it as you wish.

At power up, controller clears USER DATA RAM and RUNTIME COUNTER.

USER DATA RAM – 4 bytes

RUNTIME COUNTER – 4 bytes

Counter start from 0x00000000 at power up. Every 40mS increments with +1. Real runtime can be calculated by multiply with 40.

0x70 - Program MA0, MA1, MAE, SL0, SL1, SLE – Used in COM PORT only

0x50, 0x70, newMA0, newMA1, newMAE, newSL0, newSL1, newSLE

You do not need to restart controller.

0x61, 0x71 - Program baud rate

0x50, 0x61, newFC

newFC (0 – 9)

0 – 300

1 – 600

2 – 1200

3 – 2400

4 – 4800

5 – 9600

6 – 19200

7 – 38400

8 – 57600

9 – 115200

You do not need to restart controller.

0x62 – Set User Data RAM and Runtime Counter

0x50, 0x62, rc0, rc1, rc2, rc3, udr0, udr1, udr2, udr3

rc0, rc1, rc2, rc3 (LSB – MSB) – runtime counter

udr0, udr1, udr2, udr3 (LSB – MSB) – user data ram

If you set all 4 bytes (0xFF, 0xFF, 0xFF, 0xFF) for RC and/or UDR, then this part will not be changed.

Example:

Set RC with 0x12448892

> 50, 62, 92, 88, 44, 12, FF, FF, FF, FF

0x63 –Get User Data RAM and Runtime Counter

0x50, 0x63

Controller return:

0x63, rc0, rc1, rc2, rc3, udr0, udr1, udr2, udr3

rc0, rc1, rc2, rc3 (LSB – MSB) – runtime counter

udr0, udr1, udr2, udr3 (LSB – MSB) – user data ram

0x64 – Set User Data EEPROM

0x50, 0x64,, ude0, ude1, ude2, ude3, ude4, ude5, ude6, ude7

ude0,... ude7 – user data eeprom

0x65 –Get User Data EEPROM

0x50, 0x65

Controller return:

0x65, ude0, ude1, ude2, ude3, ude4, ude5, ude6, ude7

ude0,... ude7 – user data eeprom

0x66 – Set USB time reset, 0x67 – Set Restart time

Both commands work similarly.

The first, restart USB connection after **Time_usb**, the second - restart controller after **Time_reset**.

- Upon receipt/transmit over USB both times are restarts.
- Both commands and times are independent of each other.
- At anytime you can send a command to set new time.
- To **disable** function set 0x00 to **Time_usb** or **Time_reset**.

Generally you do not need to use these commands. The only application is if you want to restart the controller through your software.

0x50, 0x66, Time_usb, Wait_usb

Time_USB - After how much time to restart the USB connection, In seconds.

1 - means 1 second.

0 - disables this function. This is set at power up controller. After rebooting, also set to 0.

0xFF - perform immediately restart connection.

Wait_usb - How long to wait before attach it back to USB. In the 40ms interval.

$0x19 = 25 * 40\text{mS} = 1\text{sek}$

Generally not good for set time under one second as the computer cannot detect disconnecting and take it as interference. Optimum is 1.4-1.5 seconds.

Example: 50, 66, 0x2, 0x23 – After 2 sek disconnect, wait 1.4 sek and **connect** again.

0x50, 0x67, Time_reset, Wait_reset

Time_reset - After how much time to reset controller, In seconds.

1 - means 1 second.

0 - disables this function. This is set at power up controller. After rebooting, also set to 0.

0xFF - perform immediately reboot.

Wait_reset - How long to wait before to reset controller. In the 40ms interval.

$0x19 = 25 * 40\text{mS} = 1\text{sek}$

Generally not good for set time under one second as the computer cannot detect disconnecting and take it as interference. Optimum is 1.4-1.5 seconds.

Example: 50, 66, 0x2, 0x23 – After 2 sek disconnect, wait 1.4 sek and **reset** controller.

Tips

It makes no sense to set **Time_USB** > **Time_reset**, because after reboot both times were set to 0 and functions are disabled.

Generally no sense to use both commands simultaneously. It is not desirable both commands have a similar time, because after the first execution will immediately execute the second and the computer may be confused.

Before closing the port, finally, turn off these commands by sending zero times.

Recommended method of use:

6. Connect to transmitter.
7. Send 0x66 command.
8. Work usual.
9. If an error occurs and the transfer block over USB, **immediately disconnect** and wait for the preset time to expires. It is important to break the connection when the transmitter is disconnected.
10. Go to 1.

0x68, 0x78 - Set Mode for I/O

0x50, 0x68, data

With this command selects each line that will be input or output.

1- Input, 0 – Output.

The controller STUR108 has 8 lines. It is used only one byte of the command. Each bit corresponds to the line. Bit 0 for line 0, ... bit 7 for line 7.

Example:

50, 68, 0x33

Data – 0x33 (0b00110011), Line 0,1,4,5 – Input, Line 2,3,6,7 – Output.

0x69, 0x79 - Set Pull-Up for I/O

0x50, 0x69, data

With this command you select Pull-Up for each line, which is set as input.

If line is set as output, and pull-up is enabled, when line is switched to input, pull-up will be automatically enabled.

1- Enable, 0 – Disable.

The controller STUR108 has 8 lines. It is used only one byte of the command. Each bit corresponds to the line. Bit 0 for line 0, ... bit 7 for line 7.

0x6A, 0x7A - Set Outputs H/L for I/O

0x50, 0x6A, data

With this command you select level for each line, which is set as output.

If line is set as input, when line is switched to output, level will be chosen from this info.

1- High, 0 – Low.

The controller STUR108 has 8 lines. It is used only one byte of the command. Each bit corresponds to the line. Bit 0 for line 0, ... bit 7 for line 7.

0x6B –Get Status for I/O

0x50, 0x6B

With this command you get level for each line.

Controller return:

0x6B, Data

The controller STUR108 has 8 lines. It is used only one byte of the command. Each bit corresponds to the line. Bit 0 for line 0, ... bit 7 for line 7.

Examples

Commands to the transmitter

Send command without result. (0x6A)

> 50, 6A, FF – Set all Outputs HIGH
< 6A

Send command with result. (0x6B)

> 50, 6B – Get Status
< 6B, FF – All Lines are HIGH

Commands to the serial port

Send bytes to serial port without result. (0x12,0x23,0x34,0x45 – 4 bytes)

> 4, 12, 23, 34, 45
< 0 - If nothing is received in the internal buffer from serial port
or
< 2, 12, 34 – If 0x12, 0x34 (2 bytes) has been received in internal buffer.

Read from serial port

> 0
< 0 - If nothing is received in the internal buffer from serial port
or
< 2, 12, 34 – If 0x12, 0x34 (2 bytes) has been received in internal buffer.

Send bytes to serial port and receive bytes from serial port.

For example we will assume that we have connected one controller STR10804 to the transmitter.

We will send command 0x02, 0xFE (Get number of outputs and inputs) to STR10804 with CN=0xFE.

We must receive SL0, SL1, 9, 8,4,0,0,0,0, CS, SLE – 11 bytes

First send bytes to the serial port.

>7, MA0, MA1, 0x05, 0x02, CN, CS, MAE – 7 bytes

Transmitter immediately returned result.

< 0 - If nothing is received in the internal buffer from serial port

And start transmitting received bytes to serial port. STR10804 receive commands and return answer to serial port.

Here we have two options.

1/ Wait long enough to accept the entire response in the internal buffer, and then read it in one reading.

> 0 – read again serial port

< B,56,AB,9,8,4,0,0,0,0,15,78 – received 0x0B bytes from serial port

2/ Read many times until read all answer from STR10804. The part that we read depends on the interval of reading. Four times, for example.

(1)> 0

< 0 - Still nothing.

(2)> 0

< 3, **56, AB, 9** – First 3 bytes

(3)> 0

< 2, **8, 4** – Second part, (8, 4)

(4)> 0

< 6, **0,0,0,0,15,78** – Last part (0,0,0,0,15,78)

Total: **56,AB,9,8,4,0,0,0,0,15,78**

3.2. Bluetooth Transmitter

3.2.1. Principle of operation of the transmitter

Please note in this file is described only software application on the transmitter. For specific hardware configuration, please download the PDF file for the hardware of transmitter.

http://smarthardware.eu/manual/stbr106_doc.pdf

The transmitter is with parallel processing. While receiving / transmitting through the serial port, it can receive and transmit other commands through Bluetooth port. Supports full duplex for serial port, if you do not use part with SN75176 RS-485 transmitter. There are two independent buffers (256 bytes each) for transmission / reception via the serial port. When the transmitter accepts command to write to EEPROM, the following commands are blocked until the current one. Must be borne in mind, that one byte programming take time approximately 4 milliseconds. During writing, it continues to process the serial port and can send/receive through the Bluetooth port.

You can connect several transmitters to a single network. Active will be only one that is connected to your device via the Bluetooth.

JC0 .. JC4 – enable/disable commands.

JD – Clear all config, except EEPROM. **Default Settings are the same as USB Transmitter in COM Mode.**

JB – Clear all devices, restore PIN code and Name of the transmitter to default for Bluetooth.

Default PIN: 5555, NAME: STBR-106

When you connect to the transmitter, it will appear as a serial port on your computer/mobile device.

JC0..JC4 checked dynamically upon receiving the command. Please see the detail in the description of the two modes.

JD or JB jumper is checked only at the start of the transmitter (Power Up or Restart).

How to use:

Disconnect power. Put jumper. Connect power (or put Jumper and restart transmitter).The blue LED will light constantly. Now transmitter waits to remove jumper. Remove jumper. The transmitter will continue to work normally. The blue LED will flash briefly.

3.1.2. Commands for transmitter

		JC	ON	OFF
0x60/0x70	Program MA0, MA1, MAE, SL0, SL1, SLE	0	Disable	Enable
0x61/0x71	Program baud rate	1	Disable	Enable
0x62	Set User Data RAM and Runtime Counter	2	Disable	Enable
0x63	Get User Data RAM and Runtime Counter		Disable	Enable
0x64	Set User Data EEPROM		Disable	Enable
0x65	Get User Data EEPROM	3	Disable	Enable
0x68/0x78	Set Mode for I/O		Disable	Enable
0x69/0x79	Set Pull-up for I/O		Disable	Enable
0x6A/0x7A	Set Outputs for I/O		Disable	Enable
0x6B	Get Status for I/O	4	Disable	Enable
0x6D	Delete all authenticated devices in the pair list		Disable	Enable
0x6E	Set NEW Pin Code		Disable	Enable
0x6F	Set New Transmitter Name		Disable	Enable

Commands with the same numbers for the two type (USB and Bluetooth) transmitters are identical.

Some commands have two versions. Second means EEPROM version.

Example: 0x60 / 0x70

0x60 – **RAM** version. The result is valid until the transmitter is restarted. At startup are loaded the parameters from EEPROM.

0x70 – **EEPROM** version. Same as above, but is saved in EEPROM. At startup will be loaded same parameters.

In this mode, the transmitter uses commands similar to the other controllers. The length of the command is set to **14** bytes. The result which returns is always **13** bytes.

When you send a command that returns a result, it is always added last to the internal buffer data received from the serial port.

Example:

- If you receive 5 bytes, this means that the result has not been obtained in the computer and need to do an additional one reading.
- If you get 13 bytes, it means that you have read exactly the result.
- If you receive 20 bytes, it means that you have read 7 bytes from the serial port (the first) and 13 bytes (the last) the result of the command.

Jumpers JC0..JC4 are checked dynamically upon receiving the command.

In order to recognize transmitter a command, the maximum delay between the two bytes must be less than 100 milliseconds. If a command is recognized, but it is disabled, then these bytes will be ignored and will not be sent to the serial port.

If you disable **ALL** commands (JC0..JC4 – on), then everything, including the received commands will be sent to the serial port.

Transmitter uses the same format as the USB transmitter. For a detailed explanation please see the corresponding descriptions for the USB transmitter in **COM PORT Mode**. Here will be described only the three (**0x6D**, **0x6E**, **0x6F**) commands that are used only for that transmitter.

All three commands are related to the work of the Bluetooth module. The connection between the module and your device is done by entering a password, and selecting the device name. After connecting the two devices once, each subsequent connection is performed by the device address. Each of the two devices save in themselves address the other. Password and the name are no longer used.

For this controller operates as follows:

When you send a command (only these 3 commands!), it is stored and waiting for you to disconnect from the transmitter. The transmitter will continue to operate normally. When you break connection, CPU in the transmitter, will program new parameters, delete the saved devices and restart Bluetooth module. Now the new parameters are valid. To connect to the transmitter, you will need to enter your password.

0x6D - Delete all authenticated devices in the pair list

MA0, MA1, 0x0C, 0x6D, 0xC5, 0x00, 0x01, X, X, X, X, X, CS, MAE

Transmitter return:

SL0, SL1, 0x0B, 0x6D, 0,0,0,0,0,0, CS, SLE

After sending this command, the controller will erase all your devices have been connected to the transmitter.

0x6E - Set NEW Pin Code

MA0, MA1, 0x0C, 0x6E, A, B, C, D, A, B, C, D, CS, MAE

A,B,C,D – 4 numbers PIN code

Can be used only numbers (0 – 9).

Transmitter return:

SL0, SL1, 0x0B, 0x6E, Result,0,0,0,0,0, CS, SLE

Result :

0 – Command accepted.

1 – Command rejected. Usually indicates an invalid character, or difference between the first and second part of the PIN code.

Once you send the command, you can **CANCEL** it by sending a new command with an invalid PIN code.

Example:

To set new pin code to **1234**

MA0, MA1, 0x0C, 0x6E, 1, 2, 3, 4, 1, 2, 3, 4, CS, MAE

Transmitter will return

SL0, SL1, 0x0B, 0x6E, 0,0,0,0,0,0, CS, SLE

0x6F - Set New Transmitter Name

MA0, MA1, 0x0C, 0x6F, Command, B0, B1, B2, B3, B4, B5, B6, CS, MAE

The transmitter name length is between 1 and 20 characters. Can only be used the following characters:

(0 – 9), (a – z), (A – Z), (*, / , +, -, _ , space)

Since it is not possible to send all characters with one command, they are sent as the parts. For this is used parameter **Command** to indicate which part is transferred.

Command

0, 1, 2 – Part

3 – The command is now complete

The last character must be **invalid**. It is determined the length of the password.

Transmitter returns answer only when Command is 0x03.

Transmitter return:

SL0, SL1, 0x0B, 0x6F, **Result**, **Length of Name**, 0,0,0,0,0, CS, SLE

Result :

0 – Command accepted. **Length of Name – 0 - 20**

1 – Command rejected. Indicates an invalid first character – Cancel Command.

Once you send the command, you can **CANCEL** it by sending a new command with an invalid FIRST character and Command = 0x03.

Example 1:

To set new name to “Pepi”

“P”, “e”, “p”, “i” – mean ASCII code of P, e, p, i. - > 0x50, 0x65, 0x70, 0x69

> MA0, MA1, 0x0C, 0x6F, 0x00, “P”, “e”, “p”, “i”, 0x00, X, X, CS, MAE

> MA0, MA1, 0x0C, 0x6F, 0x03, X, X, X, X, X, X, CS, MAE

Answer

< SL0, SL1, 0x0B, 0x6F, 0, 4, 0,0,0,0,0, CS, SLE

4 – Length of Name

Example 2:

To set new name to “Hello My Friend 23”

> MA0, MA1, 0x0C, 0x6F, 0x00, “H”, “e”, “l”, “l”, “o”, “ ”, “M”, CS, MAE

> MA0, MA1, 0x0C, 0x6F, 0x01, “y”, “ ”, “F”, “r”, “i”, “e”, “n”, CS, MAE

> MA0, MA1, 0x0C, 0x6F, 0x02, “d”, “ ”, “2”, “3”, 0x00, X, X, CS, MAE

> MA0, MA1, 0x0C, 0x6F, 0x03, X, X, X, X, X, X, CS, MAE

Answer

< SL0, SL1, 0x0B, 0x6F, 0, 0x12, 0,0,0,0,0, CS, SLE

Example 3:

To set Cancel name

> MA0, MA1, 0x0C, 0x6F, 0x00, 0x00, X, X, X, X, X, CS, MAE

> MA0, MA1, 0x0C, 0x6F, 0x03, X, X, X, X, X, X, CS, MAE

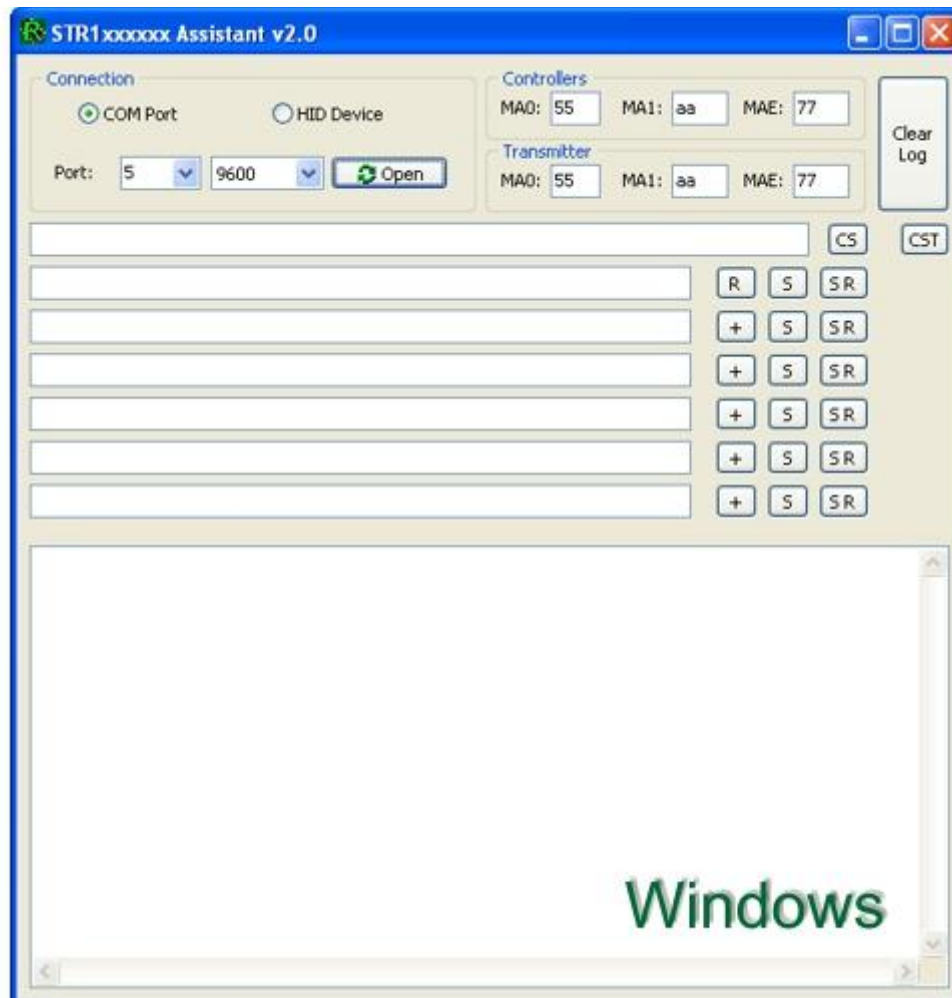
Answer

< SL0, SL1, 0x0B, 0x6F, 1, 0,0,0,0,0,0, CS, SLE

4. Using our test tool – STR1xxxx Assistant

4.1. Windows

4.1.1. COM Port mode



- Check Sum for serial port command for controllers
- Check Sum for transmitter
- Read
- Send
- Send and Read
- Copy second row to current

1. Select Port.
2. Select baud rate
3. Click button.
4. Write command in first row **without MA0, MA1, BC, CS, MAE**.

Example:

for command 0x01 – **1,fe** (CN=**0xFE** here)

for command 0x17 – **17,fe,0,8,1** (CN=**0xFE**, start number = **0**, number of outputs = **8**, on= **1**)

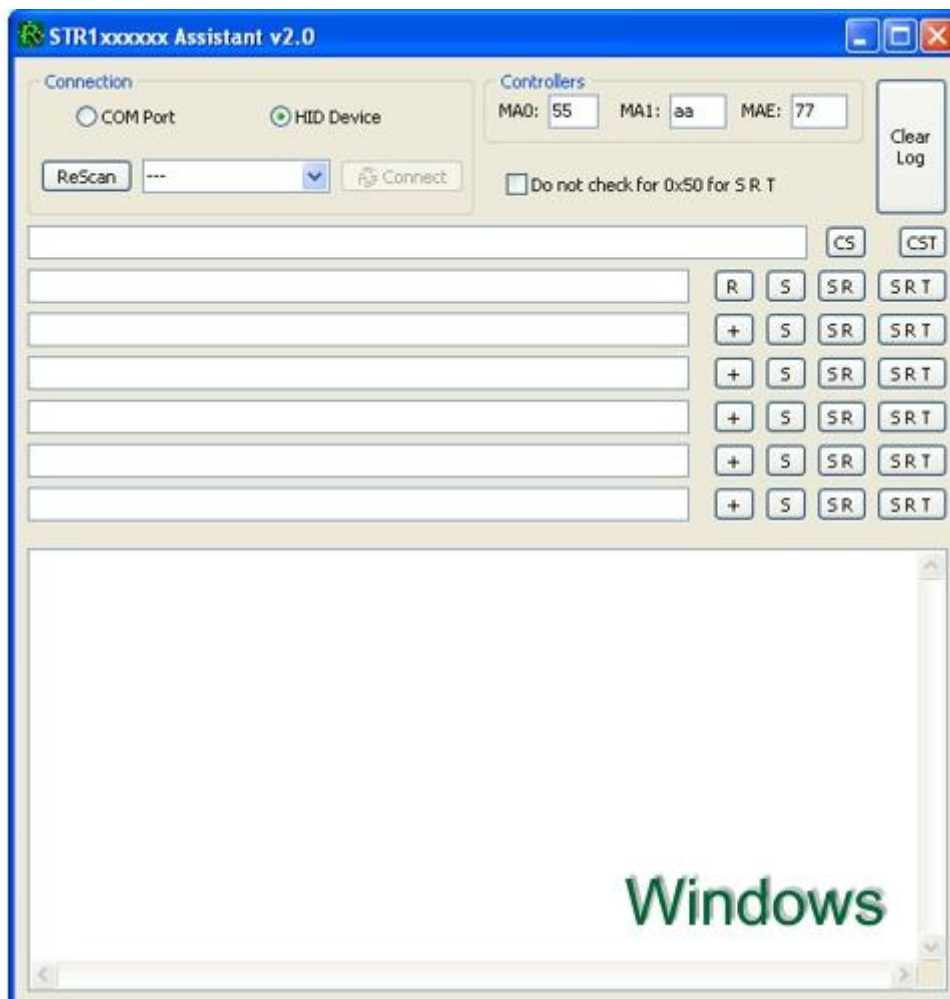
5. Click **CS** for serial port, or **CST** for transmitter.

On the second row you will see the command, ready for sending.

6. Click **S** for sending, **R** for reading or **SR** for sending and receiving.

7. You can copy command from second row, by clicking **+** to other rows.





4.1.2. HID Device mode












CS - Check Sum for serial port command for controllers

CST - Check Sum for transmitter

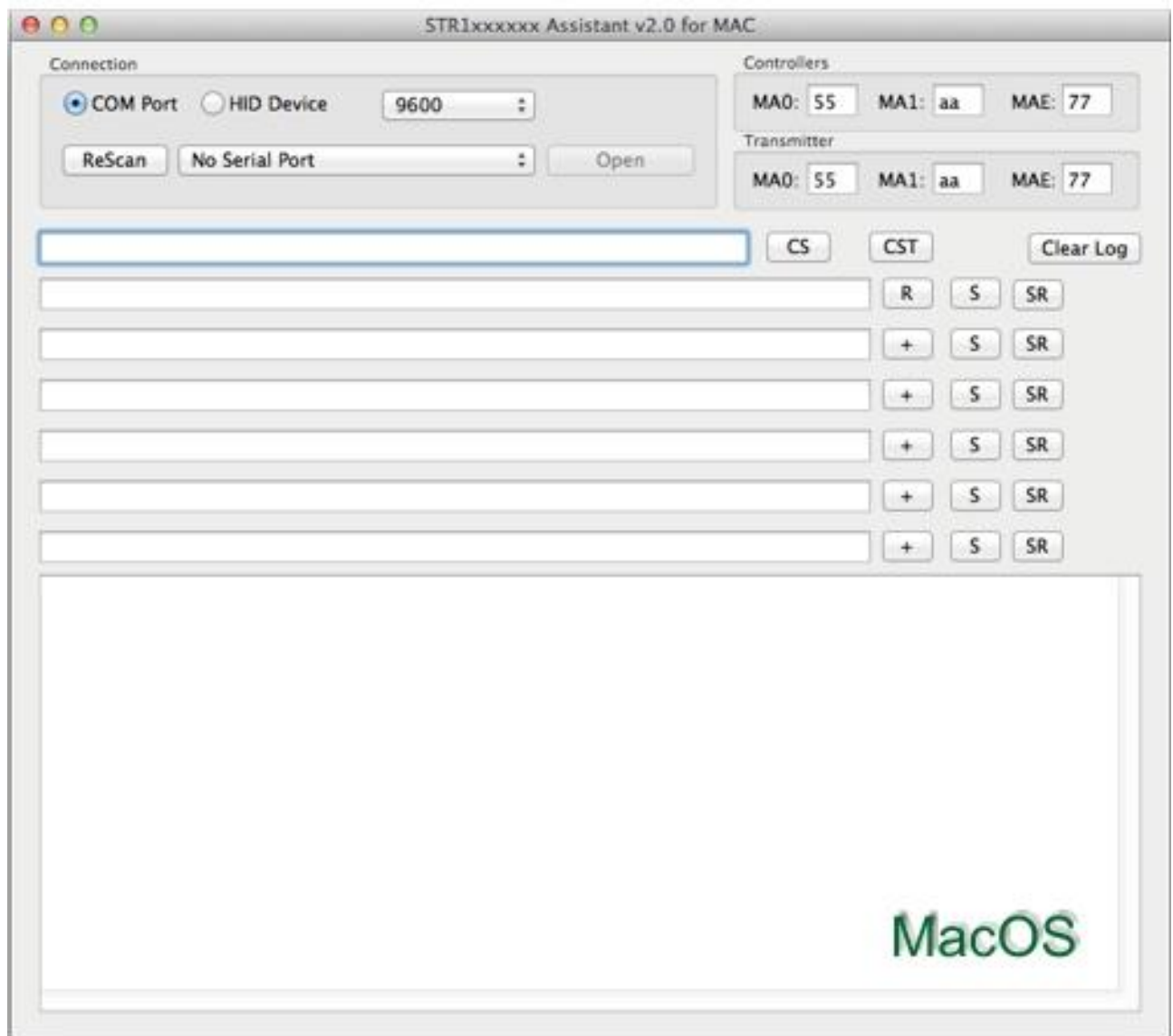
R - Read

-  - Send
 - Send and Read
 - Copy second row to current
 - Send and Read for Transmitter

1. Click .
2. Select transmitter from list (here is EEPROM data for current transmitter).
3. Click  button.
4. Write command in first row **without MA0, MA1, BC, CS, MAE**.
Example:
for command 0x01 – **1,fe** (CN=**0xFE** here)
for command 0x17 – **17,fe,0,8,1** (CN=**0xFE**, start number = **0**, number of outputs = **8**, on= **1**)
5. Click  for serial port, or  for transmitter.
On the second row you will see the command, ready for sending.
6. Click  for sending,  for reading or  for sending and receiving. For transmitter use .
7. You can copy command from second row, by clicking  to other rows.

4.2. MacOS

4.2.1. COM Port mode



- Check Sum for serial port command for controllers
- Check Sum for transmitter
- Read
- Send
- Send and Read
- Copy second row to current

1. Click .

2. Select baud rate
3. Select transmitter from list (here is EEPROM data for current transmitter).

3. Click button.

4. Write command in first row **without MA0, MA1, BC, CS, MAE**.

Example:

for command 0x01 – **1,fe** (CN=0xFE here)

for command 0x17 – **17,fe,0,8,1** (CN=0xFE, start number = 0, number of outputs = 8, on= 1)

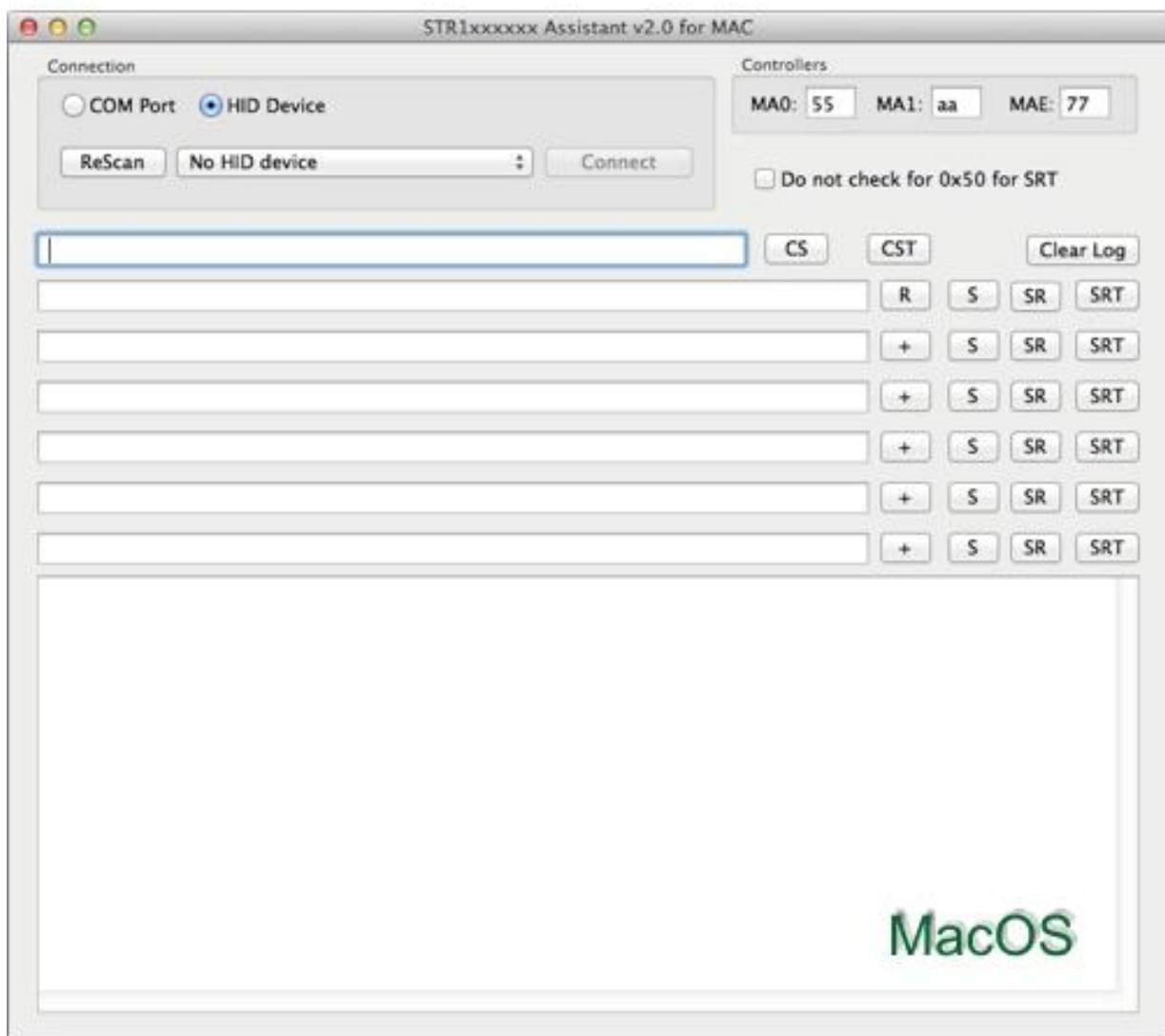
5. Click for serial port, or for transmitter.








On the second row you will see the command, ready for sending.


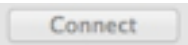







6. Click for sending, for reading or for sending and receiving.

7. You can copy command from second row, by clicking to other rows.

4.1.2. HID Device mode



-  - Check Sum for serial port command for controllers
-  - Check Sum for transmitter
-  - Read
-  - Send
-  - Send and Read
-  - Copy second row to current
-  - Send and Read for Transmitter

1. Click  .
 2. Select transmitter from list (here is EEPROM data for current transmitter).
 3. Click  button.
 4. Write command in first row **without MA0, MA1, BC, CS, MAE**.
- Example:
- for command 0x01 – **1,fe** (CN=**0xFE** here)
- for command 0x17 – **17,fe,0,8,1** (CN=**0xFE**, start number = **0**, number of outputs = **8**, on= **1**)
5. Click  for serial port, or  for transmitter.
- On the second row you will see the command, ready for sending.
6. Click  for sending,  for reading or  for sending and receiving. For transmitter use  .
 7. You can copy command from second row, by clicking  to other rows.

**Thank you for using our products. For current versions and new products
seek us at**

www.SmartHardware.eu