Table of Contents

1 Protocol

The ModBox's remote control uses **TCP/IP socket communication**. When turning on the embedded Human Machine Interface, a TCP server is opened and starts listening for incoming connections. Several clients can be connected at the same time.

The listening port is **25000**.

For a client to connect, it must be on the **same network**.

2 Quick Code examples

2.1 C++/Qt

The following code sample was tested with **Qt 5.9.6 MinGW 32bit** on Windows 10, and **Qt 5.9.6 GCC 64bit** on Manjaro Linux 18.0.3:

```
#include <QtNetwork/QTcpSocket>
#include <QDebug>
```

```
    int timeout_msecs = 1000;
    quint16 port = 25000;
    QString hostname = "127.0.0.1";

    QTcpSocket* socket = new QTcpSocket();

    socket->connectToHost(hostname, port);

    if (socket->waitForConnected(timeout_msecs) == true)
    {
        // Do not forget the carriage return char (\r)

        socket->write(QString("laser1:state?\r").toUtf8());
        socket->waitForReadyRead(timeout_msecs);
        qDebug() << socket->readAll(); // Should print 'ON\r' or 'OFF\r'

        socket->write(QString("LASER2:POWER 97\r").toUtf8());
        socket->waitForReadyRead(timeout_msecs);
        qDebug() << socket->readAll(); // Should print '97.0\r'

    }
    else
    {
        // Error (wrong IP/Port or the remote control server was not started
    }

    socket->deleteLater();
    socket = nullptr;
```

2.2 C

For **UNIX** systems (tested with **gcc 8.2.1** on Manjaro Linux 18.0.3):

```c
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#define BUFFER_SIZE 128

int main(int argc, char const *argv[]) {
    uint16_t port = 25000;
    char* hostname = "192.168.56.1";

    struct sockaddr_in address;
    int sock = 0, valread;
    struct sockaddr_in serv_addr;
    char* command;
    char buffer[BUFFER_SIZE] = {0};

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("\n Socket creation error \n");
        return -1;
    }

    memset(&serv_addr, '0', sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(port);

    // Convert IPv4 and IPv6 addresses from text to binary form
    if(inet_pton(AF_INET, hostname, &serv_addr.sin_addr) <= 0) {
        printf("\nInvalid address/ Address not supported \n");
        return -1;
    }

    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        printf("\nConnection Failed \n");
        return -1;
    }

    command = "laser1:state?\r";
    send(sock , command , strlen(command) , 0 );
    valread = read( sock , buffer, BUFFER_SIZE);
    printf("%s\n", buffer); // Should print 'ON\r' or 'OFF\r'

    command = "LASER2:power 97\r";
    send(sock , command , strlen(command) , 0 );
    valread = read( sock , buffer, BUFFER_SIZE);
    printf("%s\n", buffer); // Should print '97.0\r'

    return 0;
}
```

2.3 Python 3.X

```python
#!/usr/bin/env python3

import socket
import sys

remote_ip = '192.168.56.1'
port = 25000

# create socket
socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to remote server
socket.connect((remote_ip , port))

# Send command
socket.sendall('LASER1 : state? \r'.encode('utf-8'))

# Receive reply
reply = socket.recv(128)
print(reply.decode('utf-8')) # Should print 'ON\r' or 'OFF\r'

socket.sendall('laser2:power 97\r'.encode('utf-8'))
print(socket.recv(128).decode('utf-8')) # Should print '97.0\r'
```

(Tested with **Python 3.7.2** on Manjaro Linux 18.0.3, and **Python 3.7.0** on Windows 10)

3 General Command Information

All commands are UTF-8 encoded strings.

Each command (sent and received) ends with a **carriage return** (**`\r`**).

Each **setter** command generally follows the pattern **`DEVICE:SETTING <space> VALUE`**. The reply should be **`VALUE`**.

Each **getter** command generally follows the pattern **`DEVICE:SETTING?`**. The reply should be the corresponding value.

Some commands are replied to with **`OK`**.

If an **incorrect** command is sent and not recognized, the reply will be **`ERROR`**.

If a sent value is out of the setting's bounds, the **closest bound** will be used instead (the reply will also be that bound). The same goes for sending a value with more decimals than needed (`5.4789` ➔ `5.5`).

Negative values must not have any space between the minus sign and the value: `-15` ✓ `- 15` ✗

There must **not** be any space between **`DEVICE`** and the colon **`:`** or the colon **`:`** and **`SETTING`**.

All commands are **case insensitive**. However, uppercase is mainly used for the sake of readability (e.g. the following three commands have the same effect:

- `LASER:power 45.6`
- `laser:POWER 45.6`
- `LAser:pOwER 45.6`).

The **decimal separator** is **`.`** (dot).

4 ModBox General Settings - MODBOX

4.1 [R] Laser Count - LaserCount

This is a **Read Only** value.

Command	Sent	Reply
Get number of embedded lasers	**MODBOX:LaserCount?**	1 or 2

4.2 [R] Software Version – VERSION

This is a **Read Only** value.

Command	Sent	Reply
Get software version	**MODBOX:VERSION**?	V1.5.1 (e.g.)

4.3 [R] MBC type – MBCTYPE

This command is available **only since version 1.4**. Prior to this version, assume the result would be AN.

The result of this command will determine which command are available for the MBC subcommands!

This is a **Read Only** value.

Command	Sent	Reply
Get type of MBC included in the ModBox	**MODBOX:MBCTYPE**?	AN or DG

AN stands for **Analogic MBC**.

DG stands for **Digital MBC**.

5 Lasers - LASER1 or LASER2

You can use **LASER** instead of **LASER1**, as it makes more sense if your ModBox has only one embedded Laser.

5.1 State (ON / OFF) - STATE

Command	Sent	Reply
Turn On First Laser	LASER:STATE ON or LASER1:STATE ON	ON
Turn Off First Laser	LASER:STATE OFF or LASER1:STATE OFF	OFF
Get First Laser State	LASER:STATE? or LASER1:STATE?	ON or OFF
Turn On Second Laser	LASER2:STATE ON	ON
Turn Off Second Laser	LASER2:STATE OFF	OFF
Get First Second State	LASER:STATE	ON or OFF

5.2 Power (0.0 to 100.0%) - POWER

Command	Sent	Reply
Set Power for First Laser	LASER:POWER 45.6 LASER1:POWER 97.0	45.6 97.0
Get Power for First Laser	LASER:POWER? or LASER1:STATE?	45.9 (e.g.)
Set Power for Second Laser	LASER2:POWER 40.5	40.5
Get Power for Second Laser	LASER2:POWER?	45.9 (e.g.)

5.3 Temperature (0.0 to 100.0%) - TEMP

Command	Sent	Reply
Set Temperature for First Laser	LASER:TEMP 19 LASER1:TEMP 105.2	19.0 100.0
Get Temperature for First Laser	LASER:TEMP? or LASER1:TEMP?	45.9 (e.g.)

5.4 **[R]** Name – NAME

This is a **Read Only** value.

Command	Sent	Reply
Get First Laser Name	LASER:NAME? or LASER1:NAME?	1310 nm (e.g.)
Get Second Laser Name	LASER2:NAME?	1550 nm (e.g.)

5.5 **[R]** Calibration Power (0.0 to 100.0%) – CalibrationPower

This is a **Read Only** value.

This setting is (approximately) the recommended value to set the Laser power to, in order to start an mbc calibration. This matches a laser power output of around 10 dBm.

Command	Sent	Reply
Get First Laser Name	LASER:CalibrationPower? or LASER1:CalibrationPower?	20.0 (e.g.)
Get Second Laser Name	LASER2:CalibrationPower?	25.0 (e.g.)

6 AN MBC - MBC

Since **v1.4** of the embedded software, the ModBox may include either a **Digital** MBC or of an **Analogic** MBC The following commands are only available if the reply of the **MODBOX:MBCTYPE** command (4.3 page 7) is **DG**. Otherwise, you should see **7 DG MBC - MBC** page **13**.

If the version of your software is prior to **v1.4**, then your ModBox does include an AN MBC and the following commands will be available.

6.1 Running Mode (AUTO / MAN) - MODE

Command	Sent	Reply
Switch to Automatic running mode	MBC:MODE **AUTO**	AUTO
Switch to Manual running mode	MBC:MODE **MAN**	MAN
Get Running Mode	MBC:MODE?	AUTO or MAN

6.2 Bias Value (−10.000 to 10.000 V) – BIAS

You may only set this value in **MANUAL** mode. If you try to set it in **AUTOMATIC** mode, the reply will be **AUTO** and nothing will happen.

Command	Sent	Reply
Set Bias Value	MBC:BIAS 5.678	5.678 (e.g.) or AUTO
Get Bias Value	MBC:BIAS?	-7.167

6.3 Polarity (+ / −) - POLARITY

Command	Sent	Reply
Set to Quad + Polarity	MBC:POLARITY +	+
Set to Quad - Polarity	MBC:POLARITY −	−

Get Polarity	MBC: POLARITY?	+ or –

6.4 Reset Feedback Loop - RESET

Command	Sent	Reply
Reset Feedback Loop	MBC:RESET	OK

6.5 Save parameters - SAVE

Command	Sent	Reply
Save parameters to MBC's internal memory (persistent through ModBox power loss)	MBC:SAVE	OK

Please note that if your ModBox contains two embedded lasers, this also saves the current Modulator and Laser Photodiode gain values for the currently turned on Laser. This means that the next time the laser is turned on, those saved values will be restored.

6.6 **[R]** Laser photodiode Voltage (0.00 to 5.00 V) - VPDL

Command	Sent	Reply
Get Laser Photodiode Voltage	MBC:VPDL?	2.56 (e.g.)

6.7 **[R]** Modulator photodiode Voltage (0.00 to 5.00 V) - VPDM

Command	Sent	Reply
Get Modulator Photodiode Voltage	MBC:VPDM?	0.98 (e.g.)

6.8 Laser Photodiode Gain - Coarse (0.0 to 100.0%) - GCPDL

Command	Sent	Reply

Set Coarse Laser Photodiode Gain	MBC:GCPDL 65.2	65.2	
Get Coarse Laser Photodiode Gain	MBC:GCPDL?	48.0	(e.g.)

6.9 Laser Photodiode Gain - Fine (0.0 to 100.0%) - GFPDL

Command	Sent	Reply	
Set Fine Laser Photodiode Gain	MBC:GFPDL 39.2	39.2	
Get Fine Laser Photodiode Gain	MBC:GFPDL?	9.3	(e.g.)

6.10 Modulator Photodiode Gain - Coarse (0.0 to 100.0%) - GCPDM

Command	Sent	Reply	
Set Coarse Modulator Photodiode Gain	MBC:GCPDM 12.3	12.3	
Get Coarse Modulator Photodiode Gain	MBC:GCPDM?	32.1	(e.g.)

6.11 Modulator Photodiode Gain - Fine (0.0 to 100.0%) - GFPDM

Command	Sent	Reply	
Set Coarse Modulator Photodiode Gain	MBC:GFPDM 98.7	98.7	
Get Coarse Modulator Photodiode Gain	MBC:GFPDM?	78.9	(e.g.)

7 DG MBC - MBC

The following commands are only available since **v1.4** of the embedded software and if the reply of the **MODBOX:MBCTYPE** command (4.3 page 7) is **DG**. Otherwise, you should see **6 AN MBC - MBC page 10**.

7.1 Running Mode (AUTO / MAN) - MODE

Command	Sent	Reply
Switch to Automatic running mode	MBC:MODE **AUTO**	AUTO
Switch to Manual running mode	MBC:MODE **MAN**	MAN
Get Running Mode	MBC:MODE?	AUTO or MAN

7.2 Bias Value (−10.000 to 10.000 V) – BIAS

You may only set this value in **MANUAL** mode. If you try to set it in **AUTOMATIC** mode, the reply will be **AUTO** and nothing will happen.

Command	Sent	Reply
Set Bias Value	MBC:BIAS 5.678	5.678 (e.g.) or AUTO
Get Bias Value	MBC:BIAS?	−7.167

7.3 Transfer Level (QUAD+ / QUAD−) - TRANSFERLEVEL

Command	Sent	Reply
Set to Quad + Polarity	MBC:TRANSFERLEVEL QUAD+	QUAD+
Set to Quad - Polarity	MBC:TRANSFERLEVEL QUAD−	QUAD−
Get Polarity	MBC:TRANSFERLEVEL?	QUAD+ or QUAD−

7.4 Photodiode Polarity (+ / −) - PHOTODIODEPOLARITY

Command	Sent	Reply
Invert Photodiode Polarity	MBC:PHOTOTIODEPOLARITY INV	INV
Restore Photodiode Polarity (set to Not Inverted)	MBC:PHOTODIODEPOLARITY NOT	NOT
Get Photodiode Polarity	MBC:PHOTODIODEPOLARITY?	INV or NOT

7.5 Rescan - RESCAN

Command	Sent	Reply
Start MBC Rescan	MBC:RESCAN	OK

7.6 Save parameters - SAVE

Command	Sent	Reply
Save parameters to MBC's internal memory (persistent through ModBox power loss)	MBC:SAVE	OK

7.7 Photodiode Gain (1 to 127) - PHOTODIODEGAIN

Command	Sent	Reply
Set Photodiode Gain	MBC:PHOTODIODEGAIN 65	65
Get Photodiode Gain	MBC:PHOTODIODEGAIN?	48 (e.g.)

7.8 Dither Amplitude (10 to 1000 mV) - DITHERAMPLITUDE

Rounded to a multiple of 10 mV.

Command	Sent	Reply
Set Dither Amplitude	MBC:DITHERAMPLITUDE 392	390
Get Dither Amplitude	MBC:DITHERAMPLITUDE?	10 (e.g.)

7.9 Dither Frequency (400 to 1400 Hz) - DITHERFREQUENCY

Rounded to a multiple of 40 Hz.

Command	Sent	Reply
Set Dither Frequency	MBC:DITHERFREQUENCY 470	480
Get Dither Frequency	MBC:DITHERFREQUENCY?	1080 (e.g.)