
Ethernet / RS232

13

Through the use of a normal PC network, or an RS232 connection, it is possible to remote control OBJ INKdraw. The principle and the protocol is described in the following. HS Automatic has developed sample applications that demonstrate this communication, you can download these from our home page.

Although the protocol (language) is practically the same, there are some small differences between the two methods of communicating. These will be introduced, followed by the complete language specification.

TCP/IP (Ethernet)

From the server, you can connect arbitrary many computers in a PC network, and address each one through either an IP address or a DNS name. Each of the printer controllers has from 1 to 16 print heads connected to it. These must be of the same type per controller, but can be of a different type in the same network. (i.e. below, ipc1 and ipc2 can have different head types)

In order to issue commands for objects, you must be connected to a *message*, which is defined as *an open file in OBJ INKdraw*.

You can address all open messages, even the non-active messages.

ALL communication must end with "#" (pound sign).

Example: To update content of text object *T1* you send:

```
OBJECT:T1;TEX;This is the new text#
```

Allowing the communication

Before you can communicate to OBJ INKdraw, you must allow the communication. Do this for each PC under *File->Preferences->Network*; select a port and "Allow connection".

The connection to the IPC happens by TCP/IP on a user-defined port, using either the IP number or the DNS address to reach the machine. You can change the parameters for the connection for each IPC, the possible settings are:

- Port to connect to (**default is port 2000**)
- Password for connection (default blank)

Connecting to OBJ and a message

As mentioned, it is vital that you connect to a message after connecting to OBJ INKdraw, otherwise you can't send commands. Below is a complete example of a communication, which enables you start updating objects.

SERVER

OBJ INKdraw answer

(connection to ipcl.abc.net)

```
TEXT:OBJ INKdraw 2.00#  
TEXT:by HS Automatic ApS#  
TEXT:http://www.hsautomatic.com#  
REQUEST:password#
```

*****# (transmits the
password), terminated by #

```
RESULT:Password OK (10)#
```

```
REQUEST:messages#
```

```
DATA:c:\x.ink#  
RESULT:Transmission OK (0)#
```

```
REQUEST:connect;c:\x.ink#
```

```
RESULT:Transmission OK (0)#
```

```
REQUEST:object list#
```

```
DATA:Text;Text1#  
RESULT:Transmission OK (0)#
```

RS232 Serial connection

In serial communication, you are addressing the message that is designed for the board you connect to. (Board is selected in "canvas size" or "new" menu). If two open messages are designed for the same board, you address the active message (i.e. the message on top). Files are loaded in the active window.

ALL communication must start with <ESC> (ASCII 27) and end with #<EOT> (ASCII 04)

Example: To update content of text object *T1* you send:

```
<ESC>OBJ:T1;TEX;This is the new text#<EOT>
```

OBJ INKdraw will respond with ACK or NAK before the data

It is possible to concatenate several commands, like

```
<ESC>OBJ:T1;TEX;This is the new text#OBJ:T2;TEX;This  
is the other text#<EOT>
```

Notice that there are some commands that do not apply to the serial communication, especially those that deal with multiple messages

Language / Protocol introduction

Before the introduction to the language, a few words on the syntax used in this documentation:

- Everything in < > symbolizes a placeholder... for example <username> means that you should replace everything including the brackets with a username.
- Everything in () is optional, and often depends on the preceding command.
- You should include : and ; in the command line. There is no ; before #.
- You must terminal *all* commands by "#", likewise OBJ INKdraw will terminate *all* replies with the "#" character.
- Line feeds are *not* used, they are only shown in this guide for visualization and readability.

The communication works by sending a string to OBJ INKdraw, which is then interpreted and replied to. You have a number of different basic statements you can send to the IPC:

| Type of statement | DESCRIPTION |
|--|--|
| COMMAND:<string># | Basic commands for the IPC, for example for stopping, starting, loading a message, and shutting down the machine |
| OBJECT:<object name>;<command>;<data># | Manipulation of an object on the canvas. |
| REQUEST:<variable>(;<other data>)# | Makes OBJ INKdraw return information to you |
| PARAMETER:<type>;<value># | Change parameters in OBJ INKdraw. |

Answers and results returned by OBJ INKdraw

When you have sent a command to OBJ INKdraw, you will get a result back. In order for you to correctly interpret the answer from OBJ INKdraw, these answers always follow a set of rules.

General rules for the answers

- All answers are preceded by a word, and a colon [:]
- All answers are terminated by the hash symbol [#]
- All actions are terminated by a result with a result code.

Types of answers from OBJ INKdraw

There are 4 types of reply from OBJ INKdraw. The table below shows an overview of the possibilities; with the text to scan for marked in **bold**:

| Reply from OBJ | Description |
|---|--|
| TEXT :text text text text# | Text information that is not directly data from the program, for example the start-up message with the HS Automatic ApS company name and program version. |
| REQUEST :type of request# | OBJ INKdraw is asking you to enter Information, terminated by the # character, for example a password. |
| DATA :value of data# <i>OR</i> DATA :field;value# | OBJ INKdraw is returning data to you. More DATA replies may follow each other, for example in case of a file listing. Each DATA reply will have one value. For items that return more than one piece of data, there is a descriptive name before the value |
| RESULT :Result description (code) # | The result of the command. This will always be the last reply , to indicate that OBJ INKdraw is ready for new commands. The result text is also followed by a code for easy identification. |

Language Reference

Commands

CMD is accepted instead of **COMMAND**

Stop print

```
COMMAND:S#  
COMMAND:stop#
```

The stop print command stops the printer. If a layout is currently printing, the current print will finish.

Start print

```
COMMAND:R#  
COMMAND:start#
```

Works on: Both versions, only in editmode. The command starts the printer.

Disconnect

```
COMMAND:D#  
COMMAND:disconnect#
```

Works on: Both versions Disconnects the client program from Obj INKDraw. Does not work on serial

Shutdown

```
COMMAND:Q#  
COMMAND:shut down#
```

Works on: Both versions Shuts down the entire computer immediately.

Load file

```
COMMAND:F;<file name>#  
COMMAND:load file;<file name>#
```

Works on: Both versions. In multitask Inkdraw this command will work if connected to an open layout, else only if less than two layouts are currently open.

Loads the file specified by <file name>. If no path is given Inkdraw will load the file from the (inkdraw)\files folder, and the .ink extension is optional.

Goto record

```
COMMAND:G;<record number>#  
COMMAND:goto;<record number>#
```

Works on: Both versions, but only in layouts where a database is loaded. The command points the database to the specified record.

Print go

```
COMMAND:P#  
COMMAND:print#  
COMMAND:go#
```

Works on: Both versions, only in printmode, and only with the fast load option activated. The print go command loads one print to the printhead. Use this command with the user-managed buffer.

Direct access of database

```
COMMAND:A;<record 1>[;<record 2>;<record 3>...]#
```

Works on: Both versions, but only if the direct access feature is enabled and a database loaded.
The command loads the record numbers used for the direct access feature.

Requests

REQ is accepted instead of REQUEST

Connect to message

```
REQUEST:connect;<message name>#
```

Works on: Multitask Inkdraw only. Connects to the specified message. File type (.ink) is mandatory. If just file name is given, file is loaded from "files directory". Does not work on serial.

List of open layouts

```
REQUEST:messages#  
REQUEST:file list#
```

Works on: Multitask Inkdraw only. Returns a list of open layouts. Format is DATA:<layout name>#. Does not work on serial.

Request list of objects

```
REQUEST:object list#
```

Works on: Both versions, in multitask Inkdraw you need to be connected to a layout

Inkdraw returns a list of objects from the current layout all having the format DATA:<object type>;<object name>#

Contents of files directory

```
REQUEST:dir#  
REQUEST:directory#
```

Works on: Both versions

Inkdraw returns a list of .ink files found in the (inkdraw)\files folder with the format DATA:<filename>#

Request data of an object

```
REQUEST:object data;<object name>#
```

Works on: Both versions. Inkdraw returns all data available for the object. The format of these data is DATA:<field>;<value>#

Request parameter data

```
REQUEST:parameters#
```

Works on: Both versions

Inkdraw returns all available parameters. The format is DATA:<field>;<value>#

Printer status

```
REQUEST:status#
```

Works on: Both versions

Inkdraw returns status of the printer (online/offline/printing) as well as hardware status (fuses, inklow). Return format is DATA:<field>;<value>#

Available fonts

```
REQUEST:font list#
```

Works on: Both versions

Inkdraw returns a list of all available fonts. Format is DATA:#

Object commands

OBJ will be accepted instead of OBJECT

Create object

```
OBJECT:<object name>;create;<object type>#
```

Works on: Both versions. The object name must not be equal to an existing object

Creates a new object with the specified name and type.

Valid object types are:

```
OTText (text object)
OTCounter (counter object)
OTBarcode (barcode)
OTDateTime (date/time object)
OTLogo (logo) ( not able to create at the moment )
OTField (field)
OTMail (mail field)
OTLine (line)
OTRectangle (rectangle)
OTEllipse (ellipse)
```

Delete object

```
OBJECT:<object name>;delete#
```

Works on: Both versions

Deletes the specified object and all sub-objects it might contain.

Rename object

```
OBJECT:<object name>;rename;<new object name>#
```

Works on: Both versions

Renames the object.

Monitoring

```
OBJECT:<object name>;monitor;<+/->#
```

Works on: Both versions

Adds ("+") or removes ("-") an object from the monitor list.

Transparency

```
OBJECT:<object name>;trans;<+/->#
```

Works on: Both versions

Turns object transparency on ("+") or off ("-").

Invert

```
OBJECT:<object name>;invert;<+/->#
```

Works on: Both versions

Inverts the object ("+") or returns the object to normal ("-").

Color

```
OBJECT:<object name>;COL;<color 1>[<color 2>]#
```

Works on: Both versions. The object must be either a rectangle, line or ellipse

Changes color on the specified object. If the object is a line there can be no <color 2>. Formats for colors are "w", "W", or "-" for white and "b", "B", "+" for black.

Font

```
OBJECT:<object name>;FON;<font name>[;<font size>;<font style>]#
```

Works on: Both versions. All objects containing a font (fields, schedules, barcodes, texts, counters, and dates)

Changes the font for the specified object. Font size and font style does not need to be present. Font style values are: 1=bold, 2=italic, 4=underline, 8=strikeout (cumulative).

Position

```
OBJECT:<object name>;POS;<x>;<y>#
```

Works on: Both versions. All objects except lines

Changes the position of the object. Positions are measured as the upper left corner.

Rotation

```
OBJECT:<object name>;ROT;<rotation>#
```

Works on: Both versions. All objects except lines

Rotates the object. Valid values for <rotation> are 0, 90, 180, or 270 (degrees).

Size

```
OBJECT:<object name>;SIZ;<width>;<height>#
```

Works on: Both versions. All objects except lines

Changes the size of the object. Any font will be automatically resized to fit in the new size.

Line position

```
OBJECT:<object name>;X-1;<start x>#
OBJECT:<object name>;X-2;<end x>#
OBJECT:<object name>;Y-1;<start y>#
OBJECT:<object name>;Y-2;<end y>#
```

Works on: Both versions, only line objects

Changes the position/size of the line. The start point of the line is (<start x>, <start y>) and the end point is (<end x>, <end y>).

Line width

```
OBJECT:<object name>;WID;<width>#
```

Works on: Both versions, only graphic objects (lines, rectangles and ellipses) Changes the line width.

Text

```
OBJECT:<object name>;TEX;<new text>#
```

Works on: Both versions, text objects only

Writes a new text in the object. The object will be automatically resized to fit the size of the new text.

Counter values

```
OBJECT:<object name>;MIN;<minimum value>#
OBJECT:<object name>;CUR;<current value>#
OBJECT:<object name>;MAX;<maximum value>#
OBJECT:<object name>;DIG;<number of digits>#
OBJECT:<object name>;DIR;<+/->#
OBJECT:<object name>;LDN;<lead in>#
OBJECT:<object name>;REP;<repeat number>#
```

Works on: Both versions, counters only

Changes the minimum, maximum, or current (displayed) values of the counter.

DIG sets the number of digits in the counter.

DIR sets the direction of the counter, "+" for counting up, "-" for counting down.

<lead in> is either space (" ", "space"), zero ("0", "zero"), or none ("none").

REP sets the repeat number of the counter.

Expiry date/set date

```
OBJECT:<object name>;EXP;<expiry date>#
OBJECT:<object name>;DAT;<date>#
```

Works on: Both versions, date/time objects only

Sets the date of a date/time object. If sending a date it has to be of the format year/month/day (2003/3/19). Note that the DAT command is different from the DAT command used with fields.

Date format

```
OBJECT:<object name>;FOR;<format>#
```

Works on: Both versions, date/time objects only

Sets the format for the date objects. Available formats are all Windows formats as well as all special Inkdraw date format features.

Load logo

```
OBJECT:<object name>;PAT;<path to new logo>#
```

Works on: Both versions, logos only

Loads a new logo specified by the name and path given.

Barcode contents

```
OBJECT:<object name>;CON;<contents>#
```

Works on: Both versions, barcode only

Changes the contents of a barcode. This will only work if no objects have been inserted into the barcode.

Barcode type

```
OBJECT:<object name>;TYP;<barcode type>#
```

Works on: Both versions, barcodes only

Sets the type of the barcode (EAN13, Codabar, etc). The type must be equal to the type given in Obj INKDraw (EAN13 will work, EAN-13 will not).

Barcode module

```
OBJECT:<object name>;MOD;<module>#
```

Works on: Both versions, barcodes only

Selects between the original Obj INKDraw barcodes and the expanded (Tec-It) barcodes. Allowed values for <module> are "Tec-It", "expanded" or "1" for the expanded module (everything else will set the original module). It is important to resend the barcode type after changing modules.

Number of field lines

```
OBJECT:<object name>;LIN;<amount of lines>#
```

Works on: Both versions, empty fields only (in Obj INKdraw 2.02 this command also works on mail objects)

Sets the number of lines in the field object. The command will not work unless the field is empty.

Alignment

```
OBJECT:<object name>;ALN;<alignment>#
```

Works on: Both versions, field objects only (in Obj INKdraw 2.02 this command also works on mail objects)

Changes the alignment of the field object. Allowed values are "left", "center", or "right".

Field data

```
OBJECT:<object name>;DAT;<line 1>;[<line 2>;<line 3>;...;]#
```

Works on: Both versions, field objects only (in Obj INKdraw 2.02 this command also works on mail objects)

The command enters data to multiple (text only) lines in a field object. Note that the DAT command is different from the DAT command used with date/times.

Parameters

PAR will be accepted instead of **PARAMETER**

Start distance

```
PARAMETER:start;<start distance>#
```

Works on: Both versions

The command changes the "start mm" parameter. The given value must be in mms. If using the 20_XJxxx EPROMs or later it will change during printmode, else it will change only in edit mode.

Set edge

```
PARAMETER:edge;<edge>#
```

Works on: Both versions, edit mode only

The command will change the sensor trigger to either positive ("pos", "positive", or "+") edge, or negative ("neg", "negative", "-") edge. The command will be accepted in print mode, but the change will not work before exiting and re-entering print mode.

Print signal

```
PARAMETER:signal;<type>#
```

Works on: Both versions

The command will set the signal type to either print signal ("print", or "+") or message signal ("message", or "-").

Endless

```
PARAMETER:endless;+#  
PARAMETER:endless;-#
```

Works on: Both versions, edit mode only

The command will turn on/off endless mode. It will be accepted in print mode, but will not change before exiting printmode.

Print mode

```
PARAMETER;mode;<print mode>#
```

Works on: Both versions, edit mode only

Changes between position encoder ("pos", "position", or "P"), modular encoder ("mod", "modular", or "M"), and velocity ("vel", "velocity", "V") modes. Changes will be accepted in print mode, but will not take effect before exiting print mode.

Print speed

```
PARAMETER:encoder;<value>#  
PARAMETER:velocity;<value>#
```

Works on: Both versions

Changes the encoder/velocity parameter. With EPROMs version 20_XJxxx and on this change will take immediate effect in print mode. With older EPROMs exiting printmode is needed.

Return data from OBJ INKdraw

All objects

```
DATA:sub;<true/false>#
DATA:rotation;<rotation>#
DATA:transparent;<+/->#
DATA:invert;<+/->#
DATA:monitor;<+/->#
```

Result from: Both versions when requesting data on any object

This is always the first line sent from Inkdraw when requesting object data. If sub sends "true" the object is stored inside another object (barcode/field) and will have less properties,

<rotation> is the rotation of the object (in degrees). Other data describe if the object is transparent, inverted and monitored during print.

Lines

```
DATA:x1;<start x>#
DATA:x2;<end x>#
DATA:y1;<start y>#
DATA:y2;<end y>#
DATA;size;<line width>#
DATA;color;<color>#
```

Result from: Both versions when requesting data from a line

The format for <color> is "+" (black) or "-" (white).

Rectangles/ellipses

```
DATA:x;<x position>#
DATA:y;<y position>#
DATA:width;<width>#
DATA:height;<height>#
DATA:size;<line width>#
DATA;color;<color 1>;<color 2>#
```

Result from: Both versions when requesting data from a rectangle or an ellipse

The format for <color 1> and <color 2> is "+" (black) or "-" (white). Color 1 is the lineout and color 2 is the body of the object.

Text objects

```
DATA:text;<text>#
[DATA:x;<x position>#]
[DATA:y;<y position>#]
[DATA:width;<width>#]
[DATA:height;<height>#]
DATA;font;<font name>#
```

Result from: Both versions when requesting data from a text object

<text> is the contents of the object. Font, position and size are not send with texts inside barcodes or fields.

Logos

```
DATA:path;<path to logo>#
```

```
DATA:x;<x position>#
DATA:y;<y position>#
DATA:width;<width>#
DATA:height;<height>#
```

Result from: Both versions when requesting data from a logo

<path to logo> will be the full path, except if the logo is stored in the (inkdraw)\logos folder, then it will be .<filename> (notice the dot).

Counters

```
DATA:minimum;<minimum value>#
DATA:current;<current value>#
DATA:maximum;<maximum value>#
[DATA:x;<x position>#]
[DATA:y;<y position>#]
[DATA:width;<width>#]
[DATA:height;<height>#]
DATA:direction;<+/->#
DATA:leadin;<lead in>#
```

Result from: Both versions when requesting data of a counter.

Note that the current value might change quite fast during print. Position and size are not send with counters inside barcodes or fields.

<lead in> is either " " (space), "0" (zero), or "-" (none).

Barcodes

```
DATA:contents;<contents>#
DATA:x;<x position>#
DATA:y;<y position>#
DATA:width;<width>#
DATA:height;<height>#
DATA:module;<module>#
DATA:type;<barcode type>#
```

Result from: Both versions when requesting data from a barcode

<module> will be either "HSA" (original module) or "Tec-It" (expanded module).

Fields

```
DATA:x;<x position>#
DATA:y;<y position>#
DATA:width;<width>#
DATA:height;<height>#
DATA:data;<line 1>#
[DATA:data;<line 2>#]
...
```

Result from: Both versions when requesting data of a field object

The number of data lines will equal the number of text objects in the field.

Schedules

```
DATA:x;<x position>#
DATA:y;<y position>#
DATA:width;<width>#
DATA:height;<height>#
```

Result from: Both versions when requesting data of a schedule.

Dates

```
DATA:date;<expiry date>#  
DATA:format;<date format>#  
[DATA:x;<x position>#]  
[DATA:y;<y position>#]  
[DATA:width;<width>#]  
[DATA:height;<height>#]
```

Result from: Both versions when requesting data from a date/time.

Note that it sends the expiry date – not the current date. Position and size are not send with dates inside barcodes or fields.

Parameters

```
DATA:start;<start distance>#  
DATA:edge;<+/->#  
DATA:signal;<+/->#  
DATA:endless;<+/->#  
DATA:mode;<print mode>
```

Result from: both versions when requesting parameters

<print mode> is "M" (modular), "P" (position), or "V" (velocity).

Status

```
DATA:printmode;<+/->#  
DATA:printing;<+/->#  
DATA:status;<status>#
```

Result from: Both versions when requesting printer status

<status> will be a text describing errors on the system, for example "[5V fuse burned, low ink]".

All replies

```
RESULT:<error code>
```

Returns from: Both versions after all commands send

This will always be the last command send from Inkdraw and indicates the end of transmission.