„STYPE-HF“ TRACKING PROTOCOL, document rev 1.1

CONNECTIVITY OPTIONS:

OPTION 1 - SERIAL CONNECTION

Serial connection – RS – 422 interface

Baud rate: 38.400 bps

Parameters: 1 start, 1 stop bit, even parity

OPTION 2 – ETHERNET SOCKET CONNECTION

UDP over IP packages.

IP addresses and destination port are configurable on the Stype kit console.

Default source IP address:	10.10.10.10
Default destination IP address:	10.10.10.11
Default port:	6301
Source port (fixed):	0
Destination MAC address (fixed):	ff:ff:ff:ff:ff:ff (broadcast)

Data sending is synchronized with corresponding video blackburst signal, so one package is being sent in each video field (or frame for progressive formats).

This is a binary protocol. Constant length of each packet equals 67 bytes.

PACKAGE FORMAT

Pos	Name	Size (bytes)	Value Format	Description
0	Header	1	Const	Fixed value: 0x0F
1	* Commands	1	Unsigned char	Commands to the PC
2	* Timecode	3	Unsigned int24	Timecode data
5	Packet_no	1	Unsigned char	Packet ordinary number, starts at 0
6	X	4	Float, little endian	Position: + Right (meters)
10	Y	4	Float, little endian	Position: + Up (meters)
14	Z	4	Float, little endian	Position: - Look (meters)
18	Pan	4	Float, little endian	Orientation: + Pan to the right (degrees)
22	Tilt	4	Float, little endian	Orientation: + Tilt up (degrees)
26	Roll	4	Float, little endian	Orientation: + Roll Clockwise (degrees)
30	FovX	4	Float, little endian	Horizontal field of view (degrees)
34	Aspect_Ratio	4	Float, little endian	Aspect Ratio (ratio)
38	Focus	4	Float, little endian	0-close, 1-far (infinite)
42	Zoom	4	Float, little endian	0-wide, 1-tele
46	k1	4	Float, little endian	First radial distortion harmonic (mm^{-2}),
50	k2	4	Float, little endian	Second radial distortion harmonic (mm^{-4}),
54	Center_X	4	Float, little endian	Horizontal center zoom shift (in **mm**)
58	Center_Y	4	Float, little endian	Vertical center zoom shift (in **mm**)
62	PA_width	4	Float, little endian	Projection area width (in **mm**)
66	Checksum	1	Unsigned char	Unsigned sum of all preceeding bytes

* Can be disregarded for rael time rendering. Usually only needed for post production.

CAMERA MODEL

In order to approximate the real camera being tracked, Stype kit uses a camera model which is common in Comupter Graphics (CG) – a pinhole camera.

Pinhole camera is projecting a **point** in 3D space onto a 2D plane along the line passing through the **point** and the **center of projection**. In our pinhole camera the **projection plane** lies between the point in 3D space and the pinhole camera - eyepoint.

CAMERA COORDINATE SYSTEM

Our camera coordinate system is defined as follows:

- x – right direction (+ is right)
- y – up direction (+ is up)
- -z – look direction of a camera (- is look direction)

PROJECTION PLANE

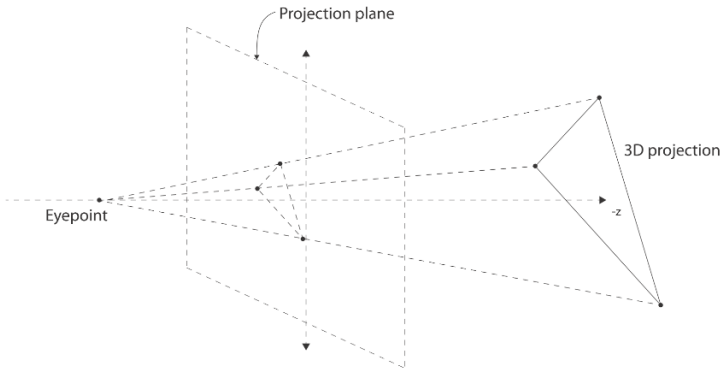

Figure 1 - Projection plane

The projection plane is parallel to the xy plane and is perpendicular to the z (look) axis. It intersects the z-axis at the coordinate$(0,0,d)$, where d is distance of an eyepoint (center of projection) to a projection plane.

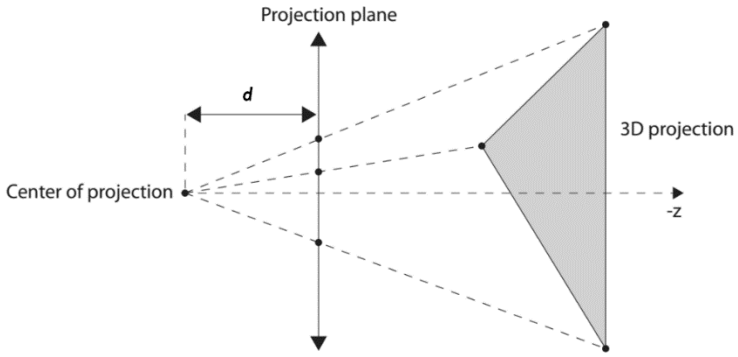

Figure 2 distance of an eyepoint to a projection plane

PROJECTION AREA

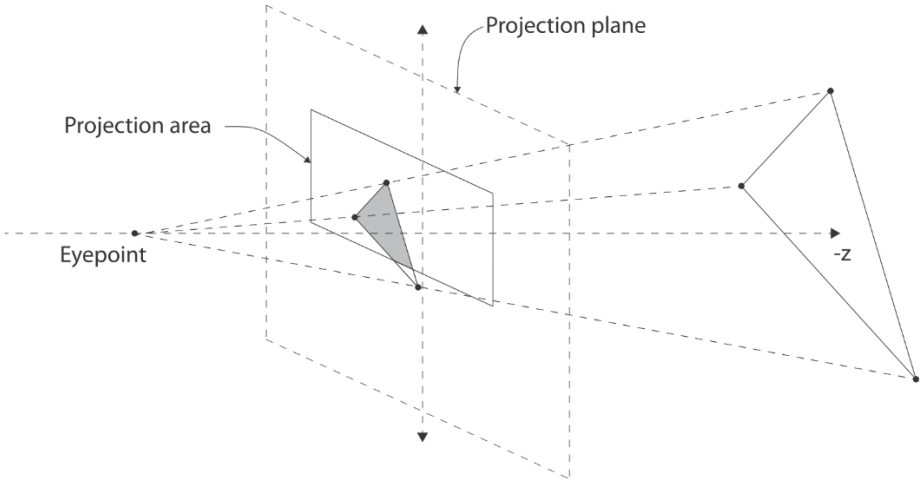

Figure 3 – Projection area

The projection area on the 2D projection plane is the rectangular area in which we project 3D points that make the camera image as shown on Figure 3. Projection area represents the CCD chip in a real camera.

In order to calculate the height of our projection area we would perform this calculation:

$$PA\_height = \frac{PA\_width}{Aspect\_Ratio}$$

Industry standard chip size of 2/3“ which produces an HD image has a PA_width of 9.59 mm. With an aspect ratio of 16:9, using the described formula, it would give us the PA_height value of 5.39 mm.

The standard 2/3“ chip size in 4:3 aspect ratio is 8.8 mm x 6.6 mm in size, and in 16:9 aspect ratio it is 9.59 mm x 5.39 mm. Both of them have a diagonal of 11 mm.

CENTER SHIFT

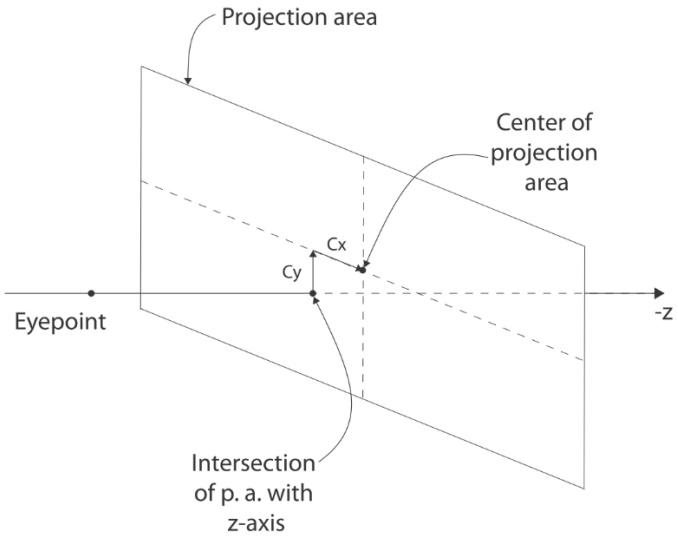

Figure 4 - Center shift

Center of projection area usually lies close to the intersection of the projection plane with the z-axis at coordinate (c_x, c_y, z). The displacement of the center of the projection area from the z-axis is due to the fact that in the real world the line passing through the center of the lens will not intersect with a CCD plane exactly on center of a CCD chip. This happens because of a mounting tolerances of a lens to the camera body. Decribed displacement is called „center shift“ and is described with a 2D vector (c_x, c_y)

REPRESENTATION OF A CAMERA IMAGE POSITIONS

Camera image positions are expressed as a 2D vectors within a **projection plane**, with the origin at the center of a **projection area** where $\boldsymbol{x_p}$ and $\boldsymbol{y_p}$ are describing the projected 3D points onto the 2D projection plane.

Example:
A 3D point with coordinates (x, y, z) in the camera coordinate system is projected onto a point (x_p, y_p) using the following equations:

$$\boldsymbol{x_p} = d * \left(\frac{x}{z}\right) - c_x \text{ and}$$

$$\boldsymbol{y_p} = d * \left(\frac{y}{z}\right) - c_y.$$

Where $\boldsymbol{x_p}$ and $\boldsymbol{y_p}$ are within the following boundaries:

$$-\frac{x_w}{2} \leq x_p \leq \frac{x_w}{2}$$

$$-\frac{y_w}{2} \leq y_p \leq \frac{y_w}{2}$$

$\boldsymbol{x_w}$ is width of a projection area and $\boldsymbol{y_w}$ is height of a projection area.

FIELD OF VIEW

In Stype HF procotol visible portion of a 3D space is specified by a **Field of View** angle and aspect ratio. Stype kit is sending the horizontal viewing angle ***FovX***, and the aspect ratio ***Aspect Ratio***.
If needed, vertical Field of View can be calculated as follows:

$$\boldsymbol{FovY} = 2 * tan^{-1}\left(\frac{\tan\left(\frac{FovX}{2}\right)}{aspect\ ratio}\right)$$

LENS DISTORTION

In order to render the radial distortions which are present because of imperfections on camera lenses, two coefficients called *k1* and *k2* are used to describe the distortion. The following equations explains how the position (x_{dis}, y_{dis}) of a distorted point on the real CCD chip can be used to find the position (x_p, y_p) which this point would have if there was no lens distortion:

$$x_p = s * (x_{dis} + c_x) - c_x$$
$$y_p = s * (y_{dis} + c_y) - c_y$$

With

$$s = 1 + r^2 * k_1 + r^4 * k_2$$

Where

$$r = \sqrt{(x_{dis} + c_x)^2 + (y_{dis} + c_y)^2}$$

r is a distance of a distorted point on a CCD chip to the **optical axis**.

On the next page you will find one very simple pseudocode example on how to simple render a distorted pixels to match the distorted camera image when rendering a HD resolution.

```
/*Pseudo code example for 1920x1080 output image (HD) without anti-aliasing*/
float X,Y,Z;           //VALUES RECEIVED FROM STYPE KIT - eyepoint X,Y,Z position
float Pan, Tilt, Roll; //VALUES RECEIVED FROM STYPE KIT - camera pan, tilt, roll euler angles
float FovX, AR;        //VALUES RECEIVED FROM STYPE KIT - Field of view and aspect ratio
float K1, K2;          //VALUES RECEIVED FROM STYPE KIT - distortion values
float PA_w;            //VALUES RECEIVED FROM STYPE KIT - projection area width
float CSX, CSY;        //VALUES RECEIVED FROM STYPE KIT - center shift in mm

float Xu, Yu;          //image coordinates in undistorted image
float Xd, Yd;          //image coordinates in distorted image
float r2;              //squared distance of a distorted point to a center of projection
float fake_FovX;       //Fake Field of View

struct px {int red, green, blue;} //struct representing one pixel
px image_u[1920*1.25, 1080*1.25]; /*Undistorted image matrix. Note that 25% more pixels are added
on both width and height in order to fill in the 'gaps' that would appear on the edges of a
distorted image as well as a shift of projection area (center shift).*/

px image_d[1920,1080];         //Distorted image (matching the size of an output image)

PA_h = PA_w / AR;              //Calculate projection area height

Pan  *= Pi / 180.0f; //Convert angle to radians
Tilt *= Pi / 180.0f; //Convert angle to radians
Roll *= Pi / 180.0f; //Convert angle to radians
FovX *= Pi / 180.0f; //Convert angle to radians

d = PA_w/ (2 * tan(FovX/2));  //calculate distance of an eyepoint to the projection area/plane

Fake_fovX = 2*atan((PA_w*1.25/2)/d);/* Calculate bigger horizontal field of view that corresponds
to the field of view that occurs if the projection area is enlarged for 25%, but the distance
from an eyepoint to to projection plane remains the same. This way, our end result field of view
will remain the same.*/

image_u = Render_u_image(1920*1.25, 1080*1.25, Fake_fovX, aspect_ratio,X,Y,Z,Pan,Tilt,Roll);
/*we are rendering 25% bigger image to provide extra pixels for filling the gaps. This percentage
can be bigger or smaller. If we put a bigger number like 1.5, unless we use extremely distorted
lens, the output result will be the same, but the performance of the system will suffer.*/

For (int yi=0; yi++; yi<=(1080-1))           // scans each line of a distorted image
    For (int xi=0; xi++; xi<=(1920-1))       // scans each pixel of a distorted image
    {
       Xd= (xi-1919.0f/2) / 1919 * PA_w ;
       Yd= (yi-1079.0f/2) / 1079 * PA_h;

       r2 = (Xd+CSX)^2 + (Yd+CSY)^2;
       f  = 1 + r2 * k1 + r2*r2 * k2;

       Xp = f * (Xd + CSX);
       Yp = f * (Yd + CSY);

       Xpi = round((1920*1.25/2 - 0.5f) + Xp*(1919/PA_w));
       Ypi = round((1080*1.25/2 - 0.5f) + Yp*(1079/PA_h));

       image_d[xi,yi] = image_u[Xpi, Ypi];
       /*takes the pixel from a calculated position in undistorted image and puts it on the
       correct place in distorted image*/
    };
/* Additional rendering performance improvement can be made if projection area is already shifted
when rendering undistorted image. This way we would need to render slightly smaller projection
area, as we would need to provide additional pixels only for distortions and not for center
shift. For the sake of simplicity, this approach is chosen in the document.*/
```

EXPLANATION OF FIELD VALUES:

Header – Fixed value 0x0F. Used for syncronisation in serial data transfer and for protocol recognition.

Commands – 1 byte (8 bits of data). b0 is least significant bit, and b7 is most significant bit.

- **b0:** When data is recorded to PC, this bit indicates wheather the recording is in ON (bit set to 1) or OFF state (bit set to 0). It can be controlled from a Stype kit console by a long press of a Start-Stop button.
- **b1:** When set to 1, Focus field is used to send the distance of the sharpest point from the eyepoint (in m), otherswise Focus field is sending the Focus percentage in range [0,1], where 0 is 0%, and 1 is 100%.
- **b2-b7:** Reserved for future use.

Package no –unsigned char representing Packet ordinary number. It starts at 0 when the unit is turned on, and increases for each following packet that is sent. After reaching the value of 255 it starts again from zero (overflows). It can be used to make sure that there are no missing packages in the packet transmission, and also to check that packets are arriving in the correct order.

Timecode – Timecode is sent in 24-bit format (3 bytes). Bits 0-6 are representing frames, Bits 7-12 are representing seconds, Bits 13-18 are for minutes, and bits 19-23 are representing hours. Timecode can be disregarded for live broadcast, as it is needed mostly for post-production purposes where it is needed to sync the picture, audio, and the positional data using the same timecode data. If no timecode generator is connected to the Stype kit, this field will send all zeroes.

X – Camera eyepoint position in meters along the X axis. Description of axis orientation is described later in the document on illustration.

Y – Camera eyepoint position in meters along the Y axis.

Z – Camera eyepoint position in meters along the Z axis.

Pan – Camera Pan movement. Represented in degrees <-180, 180]. Description of Pan, tilt and roll orientation is described later in the document.

Tilt – Camera Tilt movement. Represented in degrees <-180, 180].

Roll – Camera Roll movement. Represented in degrees <-180, 180].

FovX – Horizontal field of view. Represented in degrees <-180, 180].

A.R. – Aspect ratio of a projection area.

Focus – Position of a lens focus point. 0-closest, 1-infinite. This value can be used for generating picture with depth of field effect.

Zoom – Position of a lens focus point. 0-wide, 1-tele. Can be used for various purposes.

K_1 – Lens radial distortion first harmonic parameter. Represented in (mm^{-2}).

K_2 – Second radial lens distortion harmonic. Written in (mm^{-4}).

CenterX – Horizontal mounting shift of the lens optical axis from the center of the CCD chip. Represented in mm.

CenterY – Vertical mounting shift of the lens optical axis from the center of the CCD chip. Represented in mm.

PA_width – Projection area width. Written in mm.

Checksum – Unsigned sum of all previous bytes. C++ code example of a generating checksum from a received package is given on the next page.

CHECKSUM C++ CODE EXAMPLE

We calculate checksum by adding the unsigned char representation of all previous bytes together.

```
union {
      unsigned char element [66];   //used for checksum calculation

      struct StypePacket {
            unsigned char header;
            unsigned char command;
            unsigned char timecode_1;
            unsigned char timecode_2;
            unsigned char timecode_3;
            unsigned char Package_no;
            float X;
            float Y;
            float Z;
            float Pan;
            float Tilt;
            float Roll;
            float FovX;
            float AR;
            float Focus;
            float Zoom;
            float K1;
            float K2;
            float CSX;
            float CSY;
            float PA_width;
            unsigned char checksum;
      } StypeARPacket;
};

unsigned char check = 0;

for (char x=0; x<66; x++) check += StypeARPacket.element[x];
if (check == StypeARPacket.checksum) printf("CheckSum is OK!");
```

Note 1: Some compilers will not interpret this code correctly, as they will assign four bytes to a single byte variables (instead of one) so this pseudo-code example is only given to provide an idea of how the checksum calculation works.

Note 2: Although Ethernet packets already have multiple headers and checksums built in by the IP and Ethernet protocol, we are still using our own header and checksum calculation for situations where data is sent over the serial connection.

AXIS ORIENTATION

Pictures show the direction of each axis, and direction of rotation. The zero point of a studio can be set anywhere in space easily through our interface, but the default value is show on picture.