Kramer Electronics, Ltd.

Kramer Control Systems
Virtual Device
Build Guidelines
Revision 1

Intended for Kramer Technical Personnel or external System Integrators.

To check that you have the latest version, go to the DOWNLOADS section of our Web site at:

http://www.kramerelectronics.com/support/downloads.asp

Contents

Figures

1 Introduction

Kramer Electronics offers a wide variety of control products that meet numerous different AV requirements. These solutions make up the Kramer control system and offer seamless compatibility with Kramer signal management and scaling products, as well as interfacing with display devices and other third party products for complete room control solutions.

The control action lists (macros), associated in the K-Config configuration application with various physical (keypad controllers) and software triggers, can also be initiated by sending the Kramer Protocol 3000 commands over IP.

This document serves as a guideline for building an application (referred to as the "Virtual Device" in K-Config and in this document) that will serve as a User Interface device in a Kramer based room control system.

The application can run on any platform (mobile, PC, and so on) and under any OS (Windows®, Linux®, Android™, iOS®, and so on) - according to the competence of the developer, and end-user preference.

The developer can use "Virtual Device" development guidelines to create the application for the Virtual Device without any limitation of platform, design, and so on.

This document is intended for competent developers under the relevant OS and for the relevant devices.

1.1 Definitions (within the context of this document)

K-NET™ setup – A Kramer control setup, made from at least two units (one of them, the User Interface device, can be a Virtual Device). The K-NET units communicate between them using the Kramer Protocol 3000. Each unit on the K-NET setup has its own unique K-NET ID number.

Master Room Controller (Master RC) – A Kramer room control device; used as the Master in a K-NET room control installation.
The Master RC will have a K-NET ID of 1 and will run the relevant configuration file created in the K-Config application.
As of February 2011, Kramer's applicable Master RCs for Virtual Device setups are – **SV-551**, **SV-552**, **SL-1**, **SL-10**, **SL-12** and **SL-14RC**.

K-NET auxiliary device – A Kramer K-NET device, which is part of a Kramer room control setup and is connected with a K-NET cable to the Master RC.
The auxiliary device K-NET ID will always be greater than 1.

Virtual Device – An application simulating a User Interface device that communicates over IP, with a single Kramer Master RC as part of the K-NET setup.
The Virtual Device will use one of two reserved K-NET ID numbers – 11 or 12.

K-Config Virtual Device triggers layer – A functional duplication of the Virtual Device active buttons and text fields in the K-Config application. Used to link the Virtual Device screens to the control action lists (macros) written under K-Config.

2 The Communication Layer

The Virtual Device uses Ethernet connection (wired or wireless) and Protocol 3000 (see section 4), for communication with the associated Master RC.

3 The Virtual Device – an Application Example

The following sections define the general guidelines for building a Virtual Device.

3.1 System Overview

Sources	• Satellite TV set top box (video and audio on an HDMI output) • DVD player (video on a CV output with analog stereo audio) • PC or laptop (PC Graphics on a 15-pin HD output with analog stereo audio)
A Display Device	Flat panel LCD
A Master RC	**SV-552** *SummitView™ Processor Switcher*

All the sources transmit video and audio to the **SV-552** (using Kramer **SV-30X** wall plates) which outputs the video signals to the flat panel LCD and the amplified audio to a pair of ceiling speakers.

In this setup example, an iPhone® or an Android running cell phone is used as a user interface that will trigger switching. The audio and room control function is configured in the **SV-552** (see Figure 1).

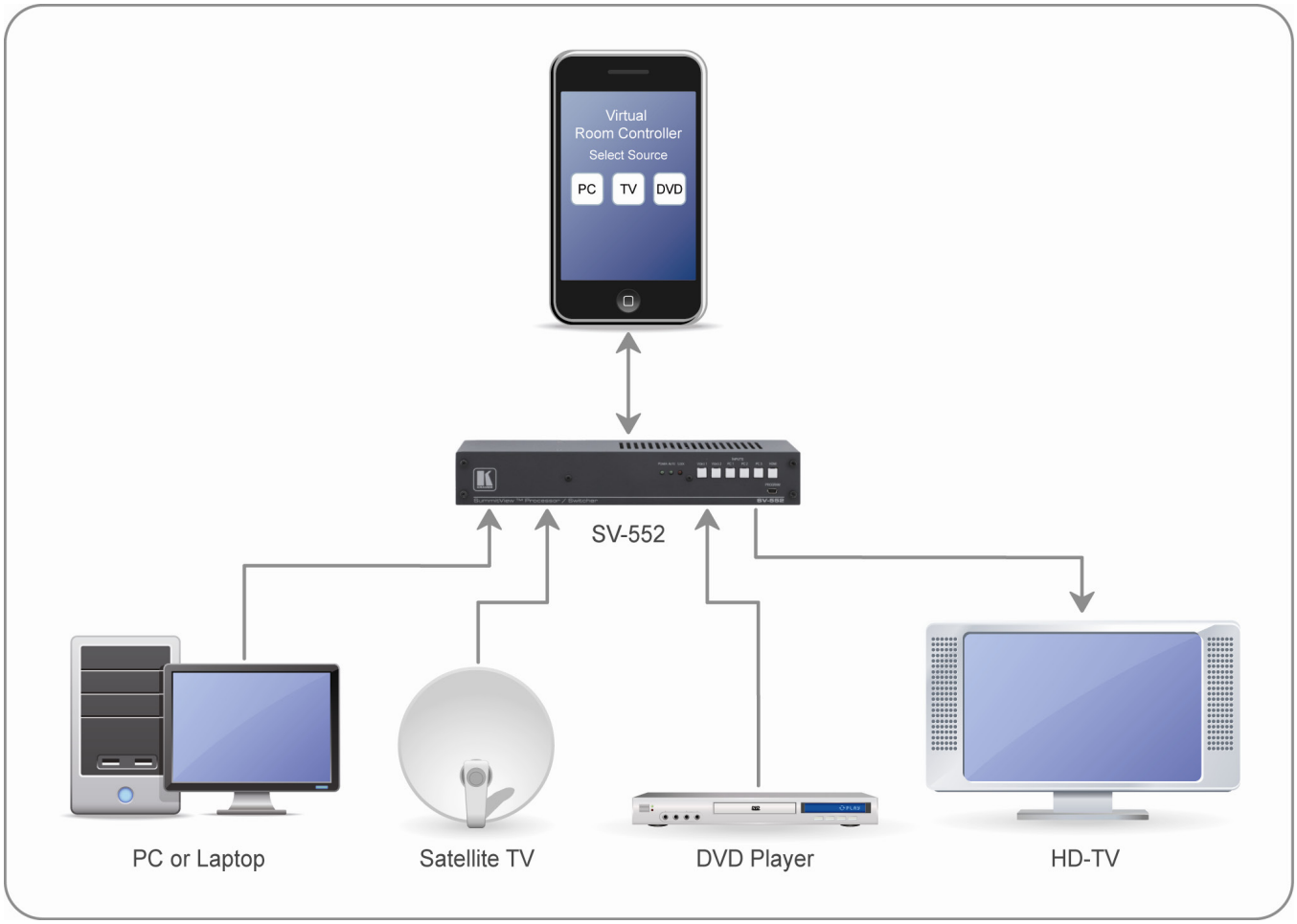

Figure 1: An Application Example

3.2 Hardware Setup

UDP[1] connection is used through port 50000 for connecting between the Mobile Device (iPhone/iPad®/Android) and the Kramer Master RC (**SV-552** in this example).

Both the Master RC and the Mobile Device should use the same net/sub-net range. The Master RC (**SV-552** in this example) will use wired connection to the relevant IT network while the mobile device will be connected to the same network wirelessly, so a wireless router or access point will probably be used (see Figure 2).

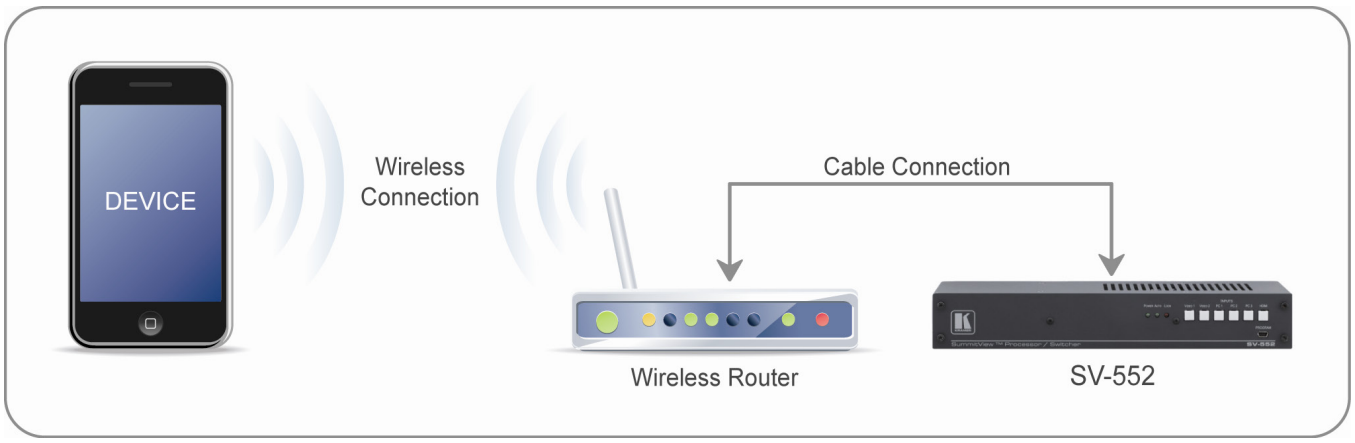

Figure 2: The Physical Connection

3.3 Implementation

The Kramer Master Room Controllers use **Protocol 3000** to interact with several K-NET auxiliary devices, including Virtual Devices (see section 4). The application should use this protocol to communicate with the Kramer Master RC and the Master RC will communicate (usually via RS-232, IR or other common control protocols) with the AV products used in the room.

Each trigger button or icon used in the Virtual Device should be functionally duplicated in the K-Config Virtual Device triggers layer.

The layout and size of buttons and text fields on the triggers layer, as seen on the graphical layout shown on K-Config, has no real importance. The only thing to note is the ID number of each functional button – enabling the developer to address each one of them in the Virtual Device application and by that trigger the required control action list.

From the above it should be clear that the Virtual Device application design, look-and-feel, style or any other property are not limited in any way and only depend on the user needs and capabilities.

1 User Datagram Protocol

3.3.1 A Setup Example

The example illustrated in Figure 3, shows an application that includes three screens:

- A MAIN MENU screen
- A SETTINGS screen
- A VIRTUAL DEVICE (source switching) screen

The first two screens are used to set and enter the VIRTUAL DEVICE screen (the third screen). The VIRTUAL DEVICE screen is duplicated in the K-Config Virtual Device triggers layer.

Figure 3: Application Screen Examples

The MAIN MENU Screen

The MAIN MENU screen includes three buttons:

- SETTINGS to enter the Settings screen
- VIRTUAL DEVICE to enter the virtual device screen
- EXIT to exit the application

All three buttons are related to internal functions or procedures in the Virtual Device application; therefore they are **not duplicated** in the K-Config Virtual Device triggers layer.

The SETTINGS Screen

The SETTINGS screen lets you configure the communication setting with the Master RC and includes three buttons as well:

- IP (as an option) to manually configure the IP Address of the Master RC
- K-ID to enter the K-NET ID of the Virtual Device application
- CONNECT to connect to the Master RC

This manual connection approach is useful if the same Virtual Device application is to run on the same physical device as a user interface for several K-NET rooms (several Master RCs).

In most cases, the end-user should not be able to easily change the IP settings and especially the K-NET ID of the Virtual Devices, since changing these settings will result in loss of communication with the Master RC.

This screen is also not duplicated on the K-Config Virtual Device triggers layer.

The VIRTUAL DEVICE Screen

The Virtual Device screen includes three buttons that trigger the switching of a switcher, scaler or a similar device in the room[1] as well as a "Select Source" text label, see Figure 4.

This screen is duplicated as the Virtual Device in the K-Config Virtual Device triggers layer.

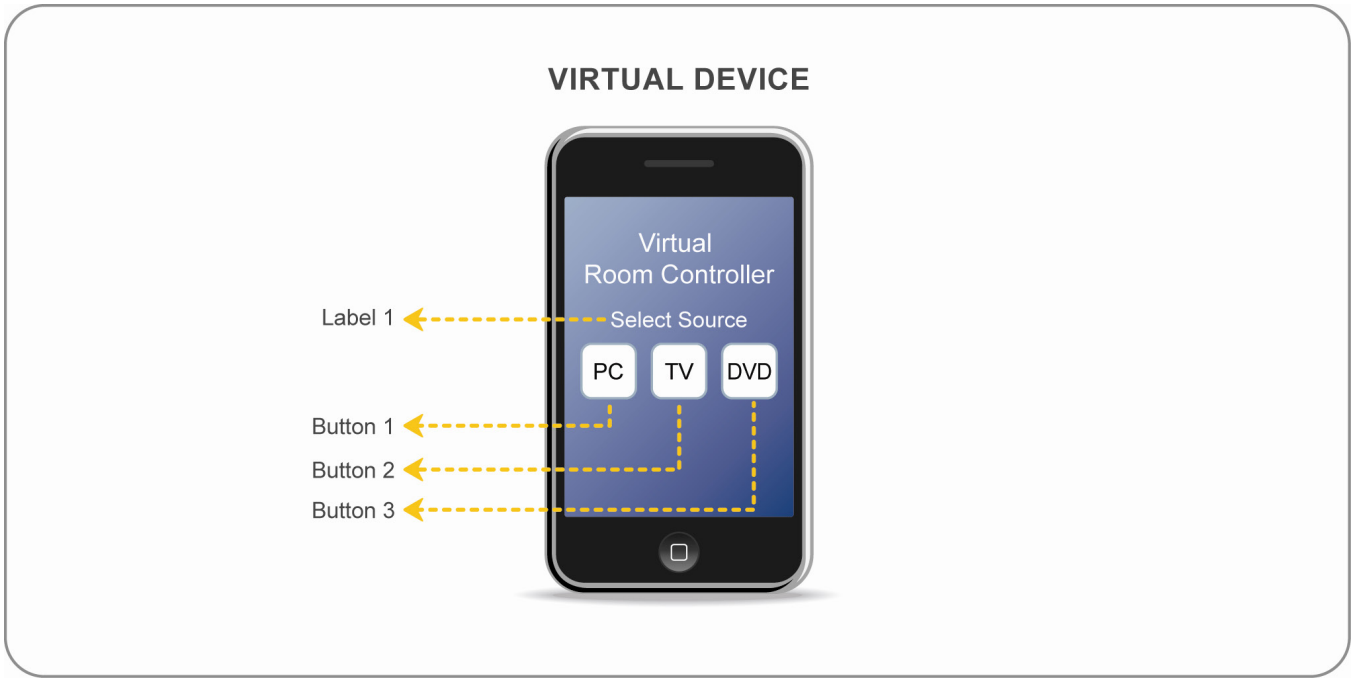

Figure 4: Virtual Device Buttons and Label

K-Config software is used to write the control action list to the K-Config Virtual Device triggers layer; and once the end user presses a button on the Virtual Device, this list of actions is executed.

3.4 Functionalities

The following sections describe how to:

- Establish a connection to the Master RC (see section 3.4.1)
- Send/receive data to/from Master RC (see section 3.4.2)

3.4.1 Establishing a Connection to the Master RC

The Virtual Device can be connected to the Master RC in various ways.

In the example described in section 3.3.1, the SETTINGS screen is used to enter the IP and the K-NET ID, after which the CONNECT button is pressed to establish a connection with the Master RC.

Alternatively, this handshaking procedure can be executed automatically by the application upon startup or within any other developer-defined event.

In any case, the connect procedure opens the UDP connection and once a connection is established the Virtual Device sends the **Handshaking** command (see section 4):

```
#\r
```

1 Many room setups will also need to switch the input channel of the display device in the room

The Virtual Device application then waits for a response from the Master RC for a predefined time period, as illustrated in Figure 5:

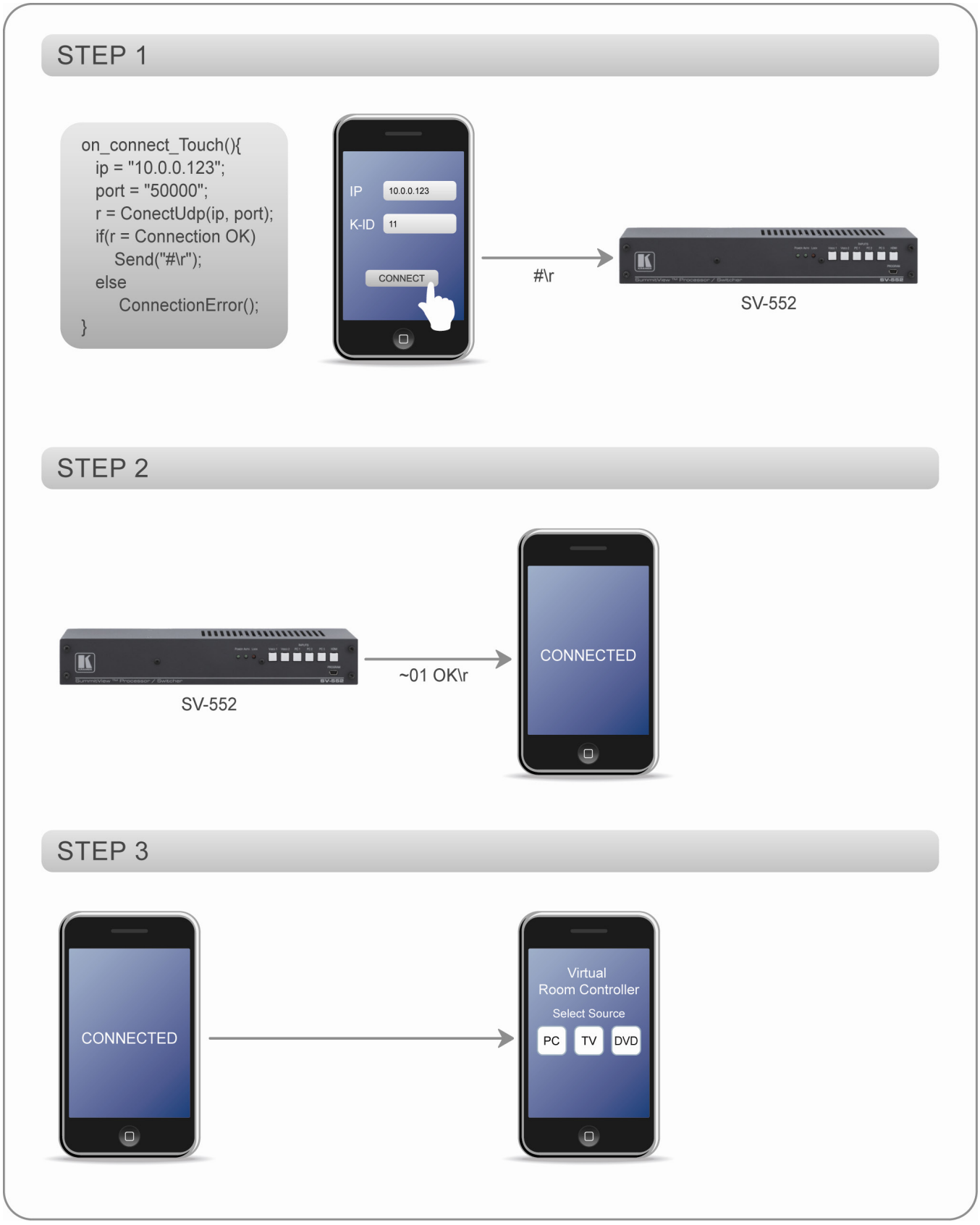

Figure 5: Connecting the Virtual Device to the Room Controller

Once an "OK" response is received from the Master RC, the connection is established, see Figure 6

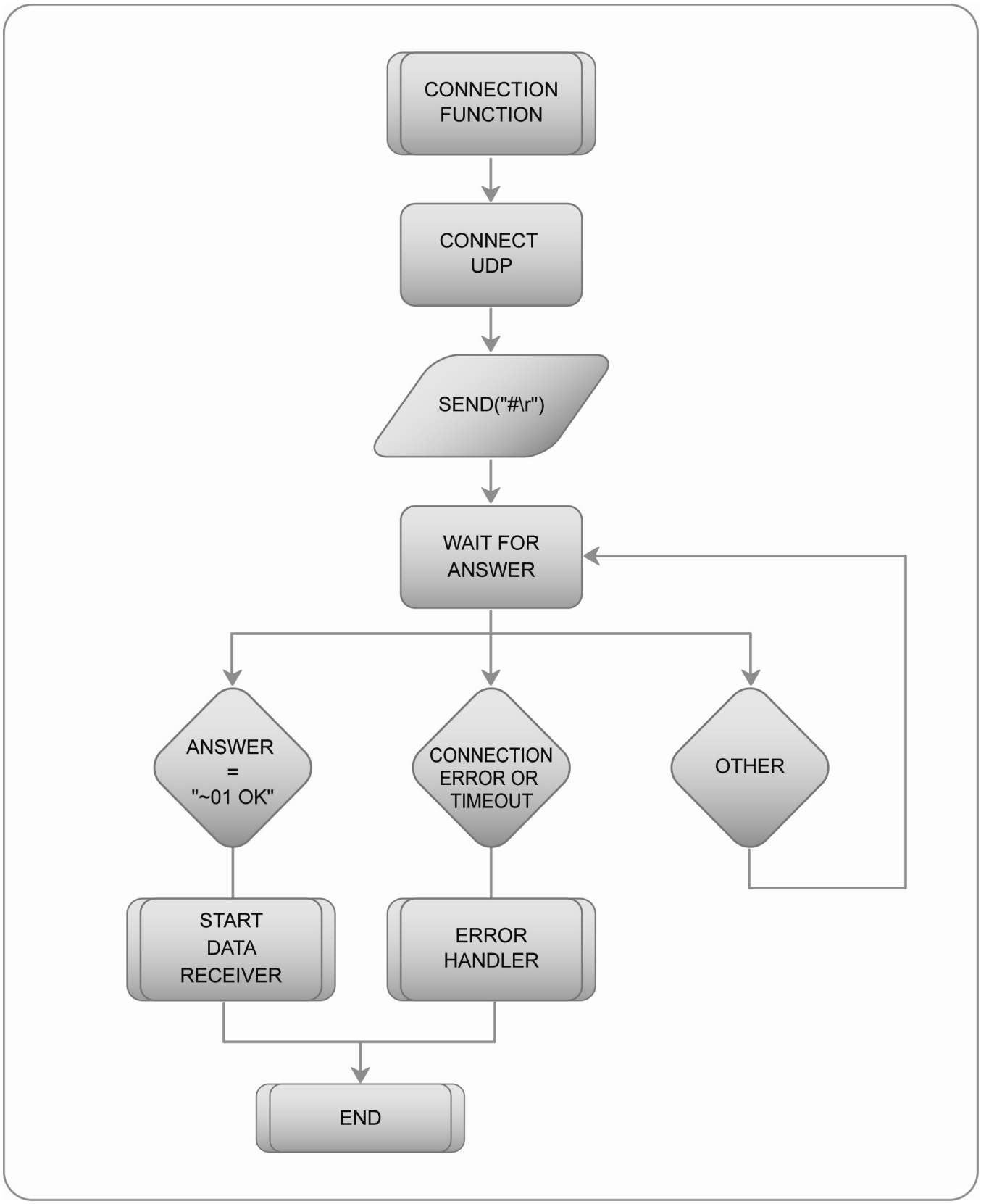

Figure 6: Connecting Flow Chart

3.4.2 Sending and receiving Data to/from Master RC

The Master RC broadcasts its outgoing Protocol 3000 messages on the K-NET bus. All its data messages are sent to all the K-NET devices that are connected to it. Each K-NET Auxiliary Device should execute only commands that match its K-NET ID. All the Other commands must be ignored, see Figure 8.

Depending on the Master RC in use, up to two Virtual Devices with reserved K-NET ID of 11 and 12 can be used.

The K-Net ID of a Virtual Device is defined in K-Config as a property of the K-Config Virtual Device triggers layer.

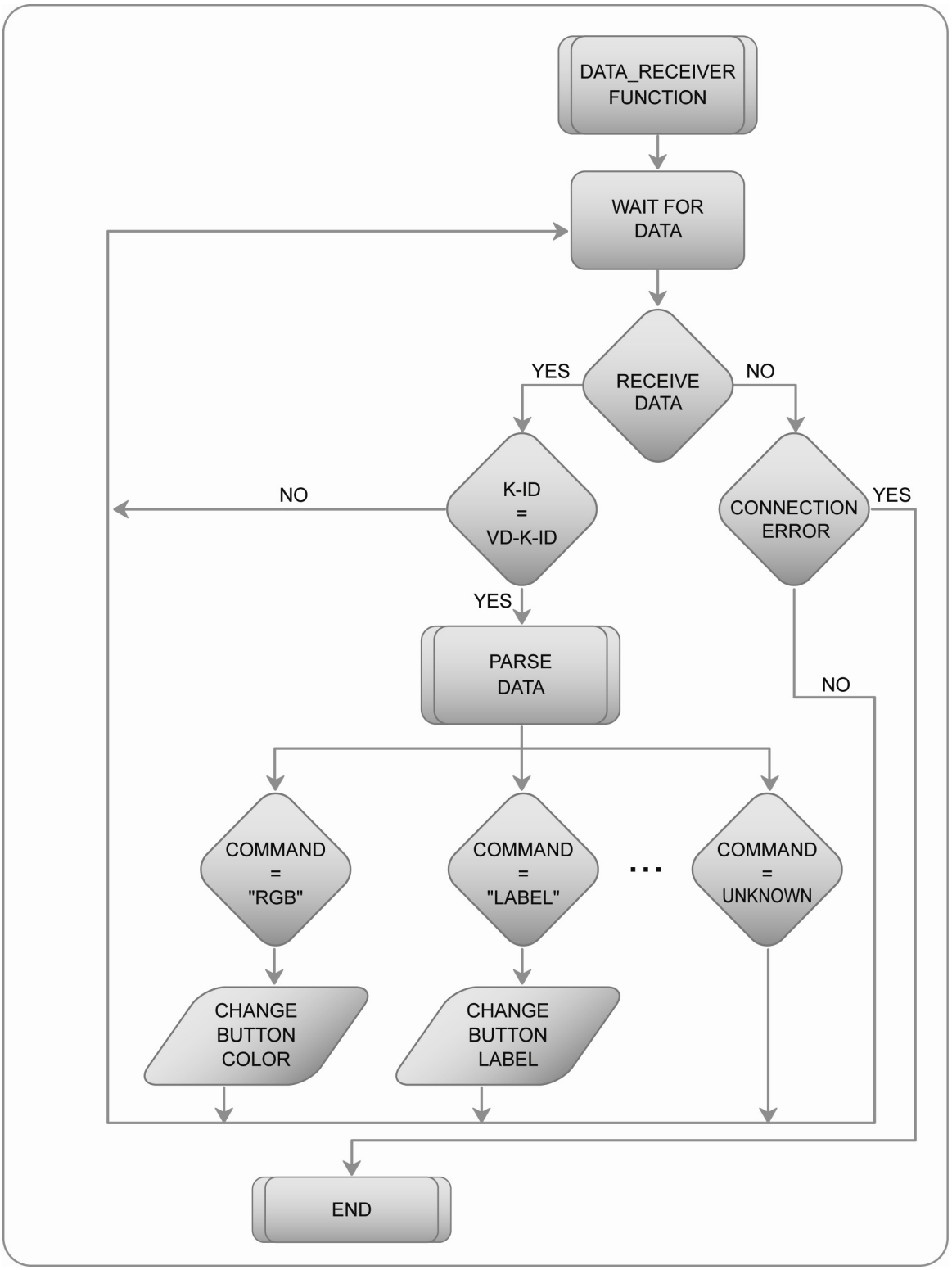

Figure 7: Receiving data from the Room Controller Flow Chart

The number of buttons on the Virtual Device and their functionalities are defined using the K-Config software in K-Config Virtual Device triggers layer.

Pressing a "button" on a Virtual Device application will send a corresponding command to the Master RC.

In the following example, the Virtual Device has K-NET ID = 11 and pressing the first button (#1 in the defined K-Config Virtual Device trigger layer) of the Virtual Device executes the following actions:

- The button color changes to red
- The text label (a defined text field) changes to PC SELECTED
- The **SV-552** switches to the VGA input

The command sent from the virtual device is:

```
#BTN 1,11,R\r          // Sent command from Virtual  Device application
```

The answer from the Master RC to this event, as defined for Button 1 of the Virtual Device with K-NET ID=11 in K-Config, is:

```
#11@RGB 1,255,0,0,1              // Set RED Color on Virtual Device
#11@LABEL 1,"ON"                 // Set Label "PC SELECTED" on Virtual Device
#... (some others commands)
~01@BTN 1,11,R OK      // ACK for command  from Master RC
```

The application catches the answer and changes the button color to "RED" and the label to "PC SELECTED", as illustrated in Figure 8:

Figure 8: Sending and Receiving Commands

Important – The Master RC can send mostly button status/lighting and text field commands to its K-Net Aux devices. However, the competent developer can easily use a combination of virtual buttons defined as a property of the Virtual Device in K-Config and use these messages as global variables and other powerful options to enable application flexibility.

For example, Button #15 in Virtual Device #12 is not seen on the Virtual Device application. Instead, its color status is changed as part of the K-Config action lists and it is used a global variable. The Virtual Device application will check it when needed (If BTN15=BLUE then…., If BTN15=RED then…).

4 Communication Protocol 3000

CR = Carriage return (ASCII 13 = 0x0D)

LF = Line feed (ASCII 10 = 0x0A)

SP = Space (ASCII 32 = 0x20)

4.1 Host message format

start	Address (optional)	body	delimiter
#	*Destination_id@*	**message**	CR

4.1.1 Simple command

(Commands string with only one command without addressing):

start	body	delimiter
#	Command SP *Parameter_1,Parameter_2,...*	CR

Device message format:

start	Address (optional)	body	delimiter
~	*Sender_id@*	**message**	CR LF

4.1.2 Device long response (Echoing command)

start	Address (optional)	body	delimiter
~	*Sender_id@*	**command** SP **[***param1 ,param2 ...***]result**	CR LF

4.2 Command parts details

Command:

Sequence of ASCII letters ('A'-'Z', 'a'-'z' and '-').

Command will be separated from parameters with at least single space.

Parameters:

Sequence of Alfa-Numeric ASCII chars ('0'-'9','A'-'Z','a'-'z' and some special chars for specific commands), parameters will be separated by commas.

Message string:

Every command must be entered as part of a message string that begins with **message starting char** and ends with **message closing char**. Note that string can contain more than one command separated by pipe ("|") char.

Message starting char:

'#' for host command\query.

'~' for machine response.

Device address:

K_Net Device ID followed by '@' char.

Query sign = '?', will follow after some commands to define query request.

Message closing char =

Host messages - Carriage Return (ASCII 13), will be referred to by CR in this document.

Machine messages - Carriage Return (ASCII 13) + Line-Feed (ASCII 10), will be referred to by CR LF

Spaces between parameters or command parts will be ignored.

4.3 Commands Entering

If a terminal software is used to connect over serial \ ethernet \ USB port, it is possible to directly enter all commands characters (CR will be entered by Enter key, that key sends also LF, but this char will be ignored by commands parser).

Sending commands from 3rd party controllers requires the coding of some characters in a special form (such as \X##). With most 3rd party units there is a way to enter all ASCII characters so it is possible to send all the commands also from the controller.

4.4 Commands Forms

Some commands have a short name syntax beside the full name to allow faster typing, response is always in long syntax.

4.5 Input String Max Length

64 characters.

General Commands		
Command	**Syntax**	**Response**
Protocol Handshaking	#DEV_ID@CR	~DEV_ID@ OKCR LF
Parameters: **DEV_ID** – Each device has its own K-Net ID that matches to a specific device in K-Config configuration tree. K-Net Auxiliary Device must react only to commands which are preceded by its own ID. Example: WebAccess: **#03@**CR Device Application: **~03@ OK**CR LF		
Note: Every device (or application that emulates a device) should reply to this request if preceded by its K-Net ID.		

Room controllers commands		
Emulate button action	**#01@BTN** *BTN*,[*KEYPAD_ID*,[*ACTION*]]**CR**	**~01@BTN** *BTN*,*KEYPAD_ID*,*ACTION* *RESULT***CRLF**
Device needs to send this command to the master device when it wants to emulate button action *BTN* – Button ID according to the configuration in K-Config *KEYPAD_ID* – The KNET ID of the device that the emulated button belongs to according to the configuration in K-Config *ACTION* - P=Button Pressing / H= Button Holding / R=Button Release. Valid button operations sequence will be P->H(->H…)->R or P->R or only R. Pressing and Holding will activate "Activate while pressed" trigger if defined in K-Config, Releasing will activate "Activate on Release" and "Toggle" triggers. *RESULT* – OK User can supply 1, 2 or 3 arguments. The default arguments are : *KEYPAD_ID* - the first ID on the list (in the file portdef.ini) *ACTION* – P=Button Pressing / R=Button Release.		
Example (device application emulate releasing of button 1 in keypad 3): Device Application: `#01@BTN 1,3,R`**CR** Master response: `~01@BTN 1,3,R OK`**CRLF** Note: Master response may be followed by other commands or replies according to the Action list for the button trigger from configuration in K-Config.		
Config RGB button leds	#**DEV_ID@RGB** *BTN*, *RED*, *GREEN*, *BLUE*, *STATE*CR	~**DEV_ID@RGB** *BTN*, *RED*, *GREEN*, *BLUE*, *STATE* *RESULT***CRLF**
Device that implements lit buttons should react to this command and set its button. Device can support full color buttons or just plain on/off lights.		

BTN - Button ID according to the configuration in K-Config
RED, *GREEN*, *BLUE*- Color intensity for each base color. Values are from 0 (No color) to 255 (Full color). Plain buttons that support only on/off light should ignore values of these parameters (except if all set to zero – same as turning light off).
STATE :
"0" or "Off" (Color parameters can be ignored).
"1" or "On"
"2" or "Slow" for slow blink
"3" or "Fast" for fast blink
RESULT – OK if command has been executed.
"**#DEV_ID@**RGB ERR 003" should be returned if any parameter exceeds its limits (for example – Button ID does not exist).

Example:
Master Device: `#03@RGB 1,0,255,0,1` **CR**
Result: Device Application needs to light on its button 1 and set color to Full green.
Device Application response: `~03@RGB 1,0,255,0,1 OK` **CRLF**

Get RGB button leds configuration	**#DEV_ID@RGB?** *BTN* **CR**	**~DEV_ID@RGB** *BTN*,*RED*,*GREEN*,*BLUE*,*STATE* **CRLF**

Device that implements lit buttons should reply this request and give its button color and state. Device can support full color buttons or just plain on/off lights.

BTN - Button ID according to the configuration in K-Config
RED, *GREEN*, *BLUE*- Color intensity for each base color. Values are from 0 (No color) to 255 (Full color). Plain buttons that support only on/off light can set these parameters to any value (except if all are set to zero – same as turning light off).
STATE =
"0" or "Off" (Values of color parameters doesn't matter).
"1" or "On"
"2" or "Slow" for slow blink
"3" or "Fast" for fast blink

Example:
Master Device: `#03@RGB? 1` **CR**
Device Application response: `~03@RGB 1,0,255,0,1` **CRLF**

Write text to LCD	**#DEV_ID@LCD** *LCD_ID*,"*TEXT*" **CR**	**~DEV_ID@LCD** *LCD_ID*,"*TEXT*" *RESULT* **CRLF**

Device that implements label (LCD) should react to this command and set its "LCD" (or text label).

LCD_ID – Label (LCD) ID according to the configuration in K-Config
TEXT – The text to display in the label (This parameter should be enclosed by quotation marks, character 34 in Ascii)

RESULT – OK if command has been executed.
"LCD ERR 003" should return if any parameter exceeds its limits (for example – LCD with this ID doesn't exist).

Read text from LCD	**#DEV_ID@LCD?** *LCD_ID* **CR**	**~DEV_ID@LCD** *LCD_ID*,"*TEXT*" **CRLF**

Device that implements label (LCD) should react to this command and set its "LCD" (or text label).

LCD_ID – Label (LCD) ID according to the configuration in K-Config
TEXT – The text to display in the label (This parameter should be enclosed by quotation marks, character 34 in Ascii)

Restore User Interface	**#01@UI** *DEVICE_ID* **CR**	**~01@UI** *DEVICE_ID* **OK CRLF**

After answering OK, the master will send to the device commands to refresh user interface elements like RGB buttons and LCD labels.
For example:
#DEV_ID@RGB 1, *RED*, *GREEN*, *BLUE*, *STATE* **CR**
#DEV_ID@RGB 2, *RED*, *GREEN*, *BLUE*, *STATE* **CR**
…(All RGB)

#DEV_ID@LCD 1,"*TEXT***" CR**
#DEV_ID@LCD 2,"*TEXT*" **CR**
…(All LCD)

Use this command to get the current user interface state on program start or refreshing the GUI.

Result codes (errors)	
	Syntax
No error. Command running succeeded	**~DEV_ID@**COMMAND PARAMETERS **OKCRLF**
Protocol Errors	
Syntax Error (For example: Not enough parameters)	**~DEV_ID@**COMMAND **ERR001CRLF**
Command not available for this device	**~DEV_ID@ ERR002CRLF**
Parameter is out of range	**~DEV_ID@**COMMAND **ERR003CRLF**
Unauthorized access (running command without the matching login).	**~DEV_ID@**COMMAND **ERR004CRLF**

Security commands		
Command	**Syntax**	**Response**
Read Security flag	**SECUR?**	**SECUR** *ON/OFF*
Login	**LOGIN** *AUTH* *PASSWORD*	**LOGIN** *AUTH* *RESULT*
Get Login	**LOGIN?**	**LOGIN** *AUTH*
Logout	**LOGOUT**	**LOGOUT** *RESULT*

AUTH = USER \ ADMIN.

ON/OFF = "0" or "OFF"
"1" or "ON"

Login flow:

(Password and SECUR mode can be set from K-Config program).

1. **Send SECUR? Command**
2. **If answer is SECUR OFF – no login needed.**
3. **If answer is SECUR ON – continue to login (step 4)**
4. **Send LOGIN USER, *PASSWORD***
5. **Device will start to work normally.**
6. **If communication is not active for more than 5 minutes – UDP socket will be closed and device will logout automatically so you will need to LOGIN again.**
7. **To eliminate automatic logout – you can send any command (for example Handshake) from time to time.**
8. **On program exit send LOGOUT (If not – device will automatically logout after timeout).**

Password sessions:
Login with user or admin password will open a match session, so it is possible to continue entering commands without re-login until connection will terminate.
For Ethernet connection, sessions will be opened while socket connection is live.
For Serial or USB connection, session will live until time-out will reach.

* To reset admin password you need to factory reset all the device data.