



JetBox 95xx/ 93xx

# Linux Auto Run Function

---

## User Manual

[www.korenix.com](http://www.korenix.com)

## **Copyright Notice**

Copyright© 2009 Korenix Technology Co., Ltd.

All rights reserved.

Reproduction without permission is prohibited.

Information provided in this manual is intended to be accurate and reliable. However, the original manufacturer assumes no responsibility for its use, or for any infringements upon the rights of third parties that may result from its use. The material in this document is for product information only and is subject to change without notice. While reasonable efforts have been made in the preparation of this document to assure its accuracy, Korenix assumes no liabilities resulting from errors or omissions in this document, or from the use of the information contained herein.

Korenix reserves the right to make changes in the product design without notice to its users.

## **Acknowledgments**

Korenix is a registered trademark of Korenix Technology Co., Ltd.

All other trademarks or registered marks in the manual belong to their respective manufacturers.

# Table of Contents

Copyright Notice .....	2
Acknowledgments.....	2
Table of Contents .....	3
Chapter 1      Overview .....	4
1-1      Applied Korenix Model.....	4
Chapter 2      Functional Description .....	5
2-1      Function Architecture .....	5
2-2      Autorun.sh Update.....	6
2-3      Example I—Running a Marquee Light.....	7
2-4      Example II—WLAN Setting.....	9
Chapter 3      Applications Supported through Auto Run Function.....	10
3-1      Process Monitor .....	10
3-2      Modbus Gateway (Optional).....	11
Chapter 4      Appendix .....	12
4-1      Pictures & Notices Index .....	12
4-2      Linux Commands.....	13
4-3      Customer Service .....	14

# Chapter 1 Overview

The auto-run function is an advanced feature provided in the SW v1.3 release of JetBox 9300/9310. This function allows customers to run specific configuration or run specific applications in the JetBox 9300/9310 automatically. The auto-run configuration or application can then be stored onto a SD card.

There are two ways to apply the auto-run function:

**Configured by customers (autorun.sh file)**

**Software service provided by Korenix (CUST file)**

## Configured by customers

Insert the SD card into the JetBox and reboot the JetBox to run the configuration or the applications.


**autorun.sh** file


1. The content of **autorun.sh** must be written in Linux shell command
2. The content must be executable in Linux

## Software service provided by Korenix

**CUST** file

1. The content is written by Korenix according to customers' requirement.
2. The content is encrypted by Korenix.


 **Notice 1:** Each SD card has its own **CUST** file and **CANNOT** be interchanged with other 9300 Series JetBox when written by Korenix.

 **Notice 2:** If both **autorun.sh** and **CUST** files were inside one SD card, the system applies **autorun.sh** file first and then **CUST** file. The system keeps the **CUST** setting if there were any conflicts between **autorun.sh** file and **CUST** file.

## 1-1 Applied Korenix Model

JetBox auto-run function is applied in the following models:

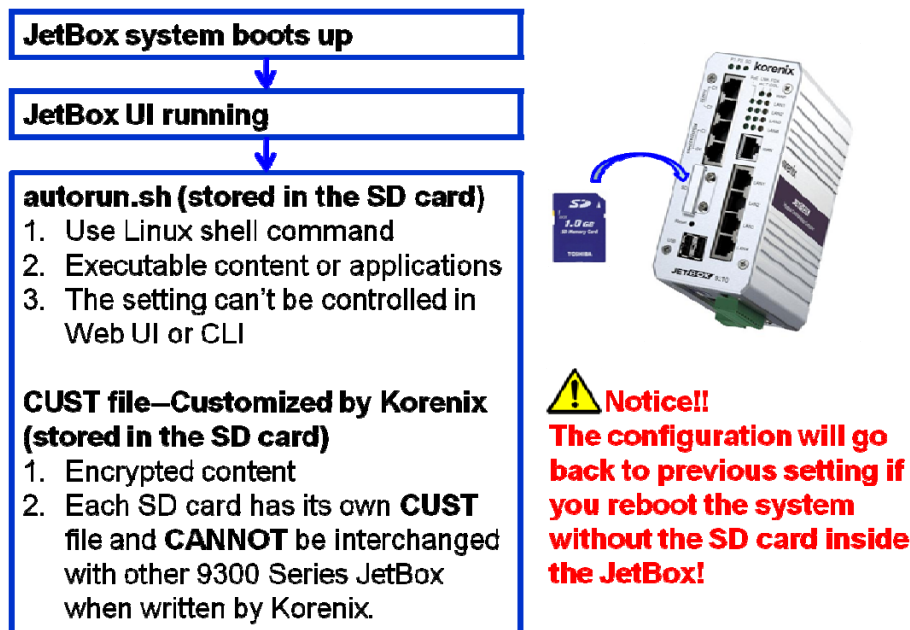
1. JetBox 9300 Industrial Networking Computer (SW v1.3 or later version)
2. JetBox 9310 Industrial PoE Networking Computer (SW v1.3 or later version)
3. All JetBox 95xx models

 **Notice 3:** After the JetBox 9300/9310 SW version1.5, the system will transfer the autorun.sh file format to UNIX automatically before executing it.

## Chapter 2 Functional Description

### 2-1 Function Architecture

Following is the architecture for the auto-run function.




Picture 1: Function architecture of auto-run in the JetBox


After the JetBox 9300/9310 boots up, the system runs the UI and a system check for an SD card.

If the system check locates an SD card, the **autorun.sh** file or **CUST** file will execute automatically.

 **Notice 4:** The content of **autorun.sh** file must be written in Linux shell command

and is executable.

 **Notice 5:** The autorun.sh file must be **UNIX text file** format. You can use the text editor (For example: **metapad**) associated in the JetBox 9300/9310 CD-ROM to write the content of autorun.sh.

 **Notice 6:** Each SD card has its own **CUST** file and **CANNOT** be interchanged with other 9300 Series JetBox when written by Korenix.


## 2-2 Autorun.sh Update

Although users make their settings ready in the beginning, sometimes, users still need to modify some parameters depending on the occasions. The Autorun.sh Update function provides the flexibility to let users update the autorun.sh file through TFTP server under the CLI of the JetBox.

Users telnet the JetBox LAN IP address to enter the CLI of the JetBox, specify the TFTP server and swap the original autorun.sh in the SD card of the JetBox with the new one in the TFTP server.

Step by step to update the autorun.sh file.

1. Telnet <the JetBox LAN IP address> to enter the CLI
2. Select (9) Backup and Restore, then (3) Autorun Update
3. Enter TFTP server IP address
4. The system will update the autorun.sh directly
5. Reboot the JetBox to apply the new setting of autorun.sh

 **Notice 7:** The new autorun.sh file must exist in TFTP server; otherwise, the update will not succeed and a warning message will pop up.

## 2-3 Example I—Running a Marquee Light

Below is an example application for the JetBox, utilizing the **autorun.sh** file. It will use the digital outputs to control the attached four indicators to light up in sequence

The source code of the example application is as follows—**demobox**.

```
/*
**  boxdemo -- JetBox Demo Program
**
**  This program is a demo tool that can set
**  DO to high and low state. This program can also setup PoE
**  state.
**
**  Copyright (C) 2007-2008 Korenix Ltd.
**
**
=====
**
*  boxdemo.c -- program source
*
*  Copyright (C) 2007-2008 Korenix Ltd.
*
*  Digital I/O Controller (DIO) - System peripherals.
*  Based on JetBox9300 series datasheet.
*
*/
#include <termios.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <sys/signal.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include "at91_dio.h"
```

```

#include <syslog.h>

#define KNX_IOFILE    "/dev/dio"

int main( int argc, char *argv[] )
{
    int i,fd;
    struct krx_iocmd iocmd;

    fd = open( KNX_IOFILE, O_RDWR | O_NOCTTY | O_NONBLOCK);
    if (fd <0)
        {perror(KNX_IOFILE);  goto exit; }

    //set diocfg
    iocmd.sec_idx = DO_SECTION;

    while(1)
    {
        for (i=0;i<4;i++)
        {
            //set port ON
            iocmd.dio_no = i;
            ioctl (fd, SET_DO, &iocmd);
            sleep(1);
        }

        for (i=0;i<4;i++)
        {
            //set port OFF
            iocmd.dio_no = i;
            ioctl (fd, CLR_DO, &iocmd);
            sleep(1);
        }

    }


exit:
    close(fd);

```



```
    return 0;
}
```

Compile this source code into the executable file for the JetBox.

 **Notice 8:** You will need the JetBox 9300/9310 SDK to compile the application source code.

Content of the **autorun.sh** file to execute the **demobox** application

```
/mnt/card/boxdemo &
```

Store the **demobox** application and **autorun.sh** file onto a SD card, then insert the card into the JetBox, and reboot. When the system is ready, it will execute the **demobox** application automatically.

## 2-4 Example II—WLAN Setting

Below is an example for WLAN setting through **autorun.sh** file.

It will enable the attached Ralink IEEE802.11b/g USB dongle and assigns the static IP 192.168.1.84 and the gateway 192.168.1.1 to the JetBox to link to the Korenix3 wireless AP.

Typically, you would enter the JetBox Linux environment and set up the WLAN through Linux commands, however, you can now write these WLAN setting commands in the **autorun.sh** file so it may execute automatically after the JetBox boots up.

Content of **autorun.sh** for the WLAN setting

```
ifconfig rausb0 up
iwconfig rausb0 key xxxxxxxxxx
iwconfig rausb0 essid korenix3
sleep 2
ifconfig rausb0 192.168.1.84
route add default gw 192.168.1.1
```

Store the **autorun.sh** file onto a SD card, insert the SD card into the JetBox, and reboot the JetBox. The setting will automatically activate after the JetBox boots up.

## Chapter 3 Applications Supported through

### Auto Run Function

#### 3-1 Process Monitor

The monitoring function is an advanced feature provided in the SW v1.3 release of JetBox 9300/9310. This function allows customers to monitor the status of the specific processes defined in the autorun.sh and restart the processes or send out a warning message once the processes stop.

The content of autorun.sh for process monitor:

```
procmon -c '<command>' [-n <seconds>] [-a 0..3] [-b '<command>']
```

-c 'cmdline' - Monitor process name.


-b 'backup cmdline' - Backup action process name


-a 0:do nothing | 1:restart | 2:backup program | 3: backup & restart – action type (default 1:restart)

-n # -- period checking in # seconds (default 5 second)

Example: **procmon -c snmpd**

Monitor if snmpd keeps alive.

 **Notice 9:** If the command of process monitor is incorrect or the monitored processes are improper, it might keep the process monitor checking in an unlimited loop.

 **Notice 10:** In SW1.3, the process monitor could only support one add-on parameter after the process name. In later SW version, all add-on parameters can be

supported.

## 3-2 Modbus Gateway (Optional)

The Modbus Gateway is optional value-added software provided by Korenix. The major function of the Modbus Gateway enables serial Modbus RTU (or Modbus ASCII) devices to communicate with Modbus TCP devices.

You can use `autorun.sh` to initiate the Modbus gateway, too.

The example content of `autorun.sh` for Modbus gateway:

```
cd/mnt/card
modbusgw -parity even &
cd/
```

The parameters of Modbus gateway are as follows:

```
modbusgw -port <port> [-protocol {rtu | ascii}] [-baud <baud>]
          [-parity {none | even | odd}] [-bits {7 | 8}] [-stop {1 | 2}] [-timeout <t>]
          [-srate <t>] [-tcp <port>]
```

or

```
modbusgw [-f <file_name>]
```

-port : the name of the serial port. Ex:ttyS1, ttyS2..

-protocol : RTU or ASCII. The default is rtu.

-baud : Baud rate. The default is 9600. Up to 460800.

-parity : odd | even | none. The default is none

-bits : Data length. RTU is 8 bits. ASCII is 7 bits.

-stop : Stop bit. The default is 1.

-timeout : Response timeout. The default is 5 seconds.

-tcp : Tcp port. The default is 502.

-srate : Scan rate. The default is 200ms.

-f : the Modbus gateway configuration file. The default file name is `./modbusgw.conf`.

And the default Modbus gateway setting is as follows:

```
[modbusgw]
```

port=ttyS1  
protocol=rtu  
baud=115200  
parity=even  
bits=8  
stop=1  
timeout=5  
srate=200  
tcp=502

## Chapter 4 Appendix

### 4-1 Pictures & Notices Index

#### Pictures

Picture 1: Function architecture of auto-run in the JetBox .....5

#### Notices

**Notice 1:** Each SD card has its own **CUST** file and **CANNOT** be interchanged with other 9300 Series JetBox when written by Korenix. ....4

**Notice 2:** If both **autorun.sh** and **CUST** files were inside one SD card, the system applies **autorun.sh** file first and then **CUST** file. The system keeps the **CUST** setting if there were any conflicts between **autorun.sh** file and **CUST** file. ....4

**Notice 3:** After the JetBox 9300/9310 SW version1.5, the system will transfer the autorun.sh file format to UNIX automatically before executing it. ....5

**Notice 4:** The content of **autorun.sh** file must be written in Linux shell command and is executable. ....5

**Notice 5:** The autorun.sh file must be **UNIX text file** format. You can use the text editor (For example: **metapad**) associated in the JetBox 9300/9310 CD-ROM to write the content of autorun.sh. ....6

**Notice 6:** Each SD card has its own **CUST** file and **CANNOT** be interchanged with other 9300 Series JetBox when written by Korenix. ....6

**Notice 7:** The new autorun.sh file must exist in TFTP server; otherwise, the

update will not succeed and a warning message will pop up. ....	6
<b>Notice 8:</b> You will need the JetBox 9300/9310 SDK to compile the application source code. ....	9
<b>Notice 9:</b> If the command of process monitor is incorrect or the monitored processes are improper, it might keep the process monitor checking in an unlimited loop. ....	10
<b>Notice 10:</b> In SW1.3, the process monitor could only support one add-on parameter after the process name. In later SW version, all add-on parameters can be supported. ....	10

## 4-2 Linux Commands

### WLAN related Linux commands:

Linux commands	Description
<b>iwconfig</b>	<p>Iwconfig is similar to ifconfig, but is dedicated to the wireless interfaces. It is used to set the parameters of the network interface which are specific to the wireless operation (for example: the frequency). Iwconfig may also be used to display those parameters, and the wireless statistics (extracted from /proc/net/wireless).</p> <p>All these parameters and statistics are device dependent. Each driver will provide only some of them depending on hardware support, and the range of values may change.</p>
<b>iwlist</b>	<p>Iwlist is used to display some additional information from a wireless network interface that is not displayed by iwconfig. The main argument is used to select a category of information, iwlist displays in detailed form all information related to this category, including information already shown by iwconfig.</p>
<b>iwspy</b>	<p>Iwspy is used to set a list of addresses to monitor in a wireless network interface and to read back quality of link information for each of those. This information is the same as the one available in /proc/net/wireless: quality of the link, signal strength and noise level.</p> <p>This information is updated each time a new packet is received, so each address of the list adds some overhead in the driver.</p>

Linux commands	Description
	Note that this functionality works only for nodes part of the current wireless cell, you cannot monitor Access Points you are not associated with (you can use Scanning for that) and nodes in other cells. In Managed mode, in most case packets are relayed by the Access Point, in this case you will get the signal strength of the Access Point. For those reasons this functionality is mostly useful in Ad-Hoc and Master mode.
<b>iwpriv</b>	<p>Iwpriv is the companion tool to iwconfig. Iwpriv deals with parameters and setting specific to each driver (as opposed to iwconfig which deals with generic ones).</p> <p>Without any argument, iwpriv list the available private commands available on each interface, and the parameters that they require. Using this information, the user may apply those interface specific commands on the specified interface.</p> <p>In theory, the documentation of each device driver should indicate how to use those interface specific commands and their effect.</p>

Above Linux command descriptions refer to the website <http://linux.die.net/man/>. You can surf in the Internet for more information.

## 4-3 Customer Service



Korenix Technologies Co., Ltd.

Business service: [sales@korenix.com](mailto:sales@korenix.com)

Customer service: [koreCARE@korenix.com](mailto:koreCARE@korenix.com)