



# VxWorks 5.4.x Driver (PPC)

*with Embedded RAINlink Technology (v2.3.9)*

## Installation and User's Guide

### Overview

The VxWorks 5.4.x driver for the ZNYX NETWORKS NetBlaster™ PCI, PMC, and CompactPCI network interface includes embedded RAINlink technology. RAINlink provides critical link services such as link aggregation, link failover, and dynamic load balancing over multiple network ports. These ports can be a group of single-channel or multi-channel ZNYX NETWORKS adapter ports, up to a maximum of 32 ports.

The VxWorks 5.4.x driver consists of an object file and configuration source code and header files. The appropriate driver can be obtained from the ZNYX NETWORKS web site: <http://www.znyx.com>. The distribution is available as a self-extracting archive. The installation consists of placing files in the correct build directories, and tailoring the build environment to include the ZNYX NETWORKS driver in the VxWorks 5.4.x image.

These instructions install the ZNYX NETWORKS network driver to bring up individual ports without RAINlink features enabled. Examples are included to bring up the most common RAINlink configurations for four ports: A system-to-system trunk, a Fast EtherChannel system-to-switch trunk, and a failover group. All the instructions are duplicated in the driver distribution readme.txt.

### Systems Requirements

<b>Platforms:</b>	Motorola MCP 750-based CompactPCI system Motorola MCP N750-based CompactPCI system Motorola MCP N765-based Compact PCI system
<b>Operating Systems:</b>	VxWorks 5.4 or 5.4.2 (Tornado II) Motorola MCP750 PPC BSP 1.2/3 Motorola MCPN750 PPC BSP 1.2/0 Motorola MCPN765 PPC BSP BETA

*Note: MCP750 BSP users must upgrade to 1.2/3 or newer.*

## Supported ZNYX NETWORKS NetBlaster Adapters

Model	Bus	Number of Ports
212	PMC	2
214	PMC	4
222 Fiber	PMC	2
244 rear i/o	PMC	4
345, 345Q	PCI	1
346, 346Q	PCI	4
348, 348Q	PCI	2
372	PCI	2
374	PCI	4
412	cPCI	2
414	cPCI	4
422	cPCI	2
424	cPCI	4
442 hot-swap	cPCI	2
444 hot-swap	cPCI	4
474 rear i/o & hot-swap	cPCI	4
478 rear i/o & hot-swap	cPCI	8

### Retrieving the ZNYX NETWORKS Drivers

1. Go to the ZNYX NETWORKS web site at <http://www.znyx.com> and navigate under the driver downloads area to get the VxWorks 5.4.x driver for your adapter. Be sure to choose the driver for PPC platform. There is one package that supports the MCP750, MCPN750, and the MCPN765. The driver is available as a self-extracting archive.

### Installing the ZNYX NETWORKS Driver for VxWorks 5.4.x

1. Extract or copy the following files into the `$(WIND_BASE)/target/config/$(BSP)` directory

```
if_zxe.obj          /* driver obj file */
zxe_apps.obj       /* Applications obj file */

if_zxePci.c        /* Znyx PCI Library */
if_zxePci.c.n765  /* ZNYX PCI Library for N765 */
if_zxePci.h

if_zxeRlk.c        /* Znyx RAINLink configuration */
if_zxeRlk.h

configZxe.c        /* Znyx startup routines(optional) */
configZxe.h
```

2. If you are building for the MCPN765, copy `if_zxePci.c.mcpn765` over `if_zxePci.c`:

```
cp if_zxePci.c.mcpn765 if_zxePci.c
```

3. Change to the `$(WIND_BASE)/target/config/$(BSP)` directory. Edit the Makefile. Add "if\_zxe.obj", "configZxe.o", "if\_zxeRlk.o" and "if\_zxePci.o" to the `BOOT_EXTRA` list and/or `MACH_EXTRA` depending on your boot configuration. Add

"zxe\_apps.obj" to the MACH\_EXTRA list. The list(s) should look similar to the following when you are finished:

```
BOOT_EXTRA = sysSpeed.o  if_zxe.obj  configZxe.o  if_zxeRlk.o \
             if_zxePci.o

MACH_EXTRA = sysSpeed.o  if_zxe.obj  configZxe.o  if_zxeRlk.o \
             if_zxePci.o  zxe_apps.obj
```

4. Edit \$(WIND\_BASE)/target/config/\$(BSP)/config.h in the following way:

a. After #define INCLUDE\_END, add:

```
#define IP_MAX_UNITS 16  /* modify for your configuration */
#define STANDALONE_NET
```

(the following are optional)

```
#define INCLUDE_NET_SHOW
#define INCLUDE_PING
#define INCLUDE_TELNET
#define INCLUDE_SHOW_ROUTINES
```

b. Change all PCI, cPCI and PMC interrupt priorities that are 0 to at least 1. It is recommended to change all of them to 7.

5. Edit \$(WIND\_BASE)/target/config/\$(BSP)/configNet.h. At the end of the file, add:

```
#include "configZxe.h"
```

6. Rebuild the vxWorks.st file. In the \$(WIND\_BASE)/target/config/\$(BSP) directory execute:

```
make vxWorks.st
```

This command is case sensitive.

7. A routine called zxeUp() is included in configZxe.c to bring up the ZNYX NETWORKS network ports. You can use zxeUp() directly to bring up individual ports or RAINlink configurations, or use the routine as an example for bringing up the ports for your specific configuration. Documentation on zxeUp() is included towards the end of this document. As an example, to bring up the first two individual ports, you would enter:

```
zxeUp "znb0","10.0.0.1",0xff000000,0,0
zxeUp "znb1","11.0.0.1",0xff000000,0,1
```

The first parameter specifies the device, the second is the IP address, the third is the netmask, the fourth is the media type (0=autonegotiate), and the last is the physical port. To bring up a RAINlink configuration of ports, the device name changes from "znb" to "zrl", and the physical port parameter is used as a index into a table of pre-defined RAINlink configurations in if\_zxeRlk.c. Details on RAINlink follow in the next section. As an example, to configure a 2-port system-to-system trunk (index=3), enter:

```
zxeUp "zrl0","10.0.0.1",0xff000000,0,3
```

If you would like to have the ports come up when vxWorks loads, one way would be to call `zxeUp()` in the `usrRoot()` task in `$(WIND_BASE)/target/config/all/usrConfig.c`.

## Configuring RAINlink for VxWorks 5.4.x

RAINlink features are statically added into your VxWorks image at compile time. The RAIN management configuration table in `if_zxeRlk.c` contains the available RAINlink configurations. ZNYX NETWORKS provides six standard configurations for basic two-port and four-port trunking and failover groups. The table below shows the provided RAINlink configurations, the name of the table entry, and the corresponding index number:

Configuration	Name	Index Number
System-to-System trunking between 4 ports	<code>zxe_rlss4</code>	0
Fast EtherChannel trunking between 4 ports	<code>zxe_rlfe4</code>	1
Fast Failover between 4 ports	<code>zxe_rlff4</code>	2
System-to-System trunking between 2 ports	<code>zxe_rlss2</code>	3
Fast EtherChannel trunking between 2 ports	<code>zxe_rlfe2</code>	4
Fast Failover between 2 ports	<code>zxe_rlff2</code>	5

Standard single port ZNYX NETWORKS devices have the name `znb`. RAINlink-configured devices have the name `zrl`. To use one of the provided configurations, create an END table entry for the RAINlink configuration you wish to load, and then load the driver. Examine `configZxe.c` and implement the method that best fits your applications needs to bring up the network interfaces. Reference `usrNetwork.c`, and `bootConfig.c` for routines to bring up the interface. A typical entry looks like:

```
{ 0, ZRL_LOAD_FUNC, "0:0:0:0:0:0:0", ZXE_BUFF_LOAN, NULL, FALSE },
```

The number at the beginning of the line specifies the unit number for the device. The unit number in the line above is 0, therefore the device name it loads is `zrl0`. `ZRL_LOAD_FUNC` is the name of the load function for a RAINlink device. Load functions for `znb` and `zrl` devices are:

```
#define ZNB_LOAD_FUNC      znbEndLoad /*driver external interface*/
#define ZRL_LOAD_FUNC      zrlEndLoad /*driver external interface*/
```

The next element is the init string, which contains all the device-specific parameters. For `zrl` devices, the first field of the init string is the RAINlink configuration table index (`zxe_rlk_config_tbl[]` in `if_zxeRlk.c`). For `znb` devices, the first field refers to the `ppa` of that device. The init string format consists of:

```
ppa | index:memBase:mediaType:mtu:recvBufs:xmitBufs:staticBufs:flags
```

where:

Name	Description
<code>&lt;ppa   index&gt;</code>	specifies the <code>ppa</code> or configuration record to use
<code>&lt;memBase&gt;</code>	defines main memory base as seen from PCI bus
<code>&lt;mediaType&gt;</code>	defines the media type of the device
<code>&lt;recvBufs&gt;</code>	defines the number of receive buffers per <code>ppa</code>

<xmitBufs>	defines the number of transmit buffers per ppa
<staticBufs>	defines the number of static buffers per ppa
<flags>	defines various interface flags

The next element in the table entry defines buffer loaning:

```
#define ZXE_BUFF_LOAN 1 /* enable buffer loaning */
```

The last two elements are a NULL reserved for the BSP, and a TRUE/FALSE indicator of whether or not the load request for that device has been processed.

For example, an END table entry for a zrl device such as a 4-port Fast EtherChannel, Layer 3 trunk, would be the following:

```
{ 0, ZRL_LOAD_FUNC, "1:0:0:0:0:0:0", ZXE_BUFF_LOAN, NULL, FALSE },
```

If you wish to customize configuration for trunks for other than two or four ports, or disable the dynamic load balancing, go to the "Advanced Trunking" section. If you wish to configure failover groups for other than two or four ports, or set timeout parameters, then go to the "Advanced Failover" section. You can also configure failover groups of trunks themselves. Always configure trunks before failover groups.

### Advanced Trunking

To add a new trunk configuration, edit the file `if_zxeRlk.c` and add an entry for the trunk. Each entry corresponds to one device and contains the `RLK_TRUNK` type, the desired mode, a 0 for the timeout field, and -1 terminated list of ports to be included in that trunk. Acceptable modes for trunks are:

Mode:	Appropriate for:
<code>IP_TRUNKING_MODE</code>	System-to-system trunking
<code>LAYER_2_MODE</code>	System-to-switch trunking
<code>LAYER_3_MODE</code>	System-to-switch trunking
<code>NO_BALANCE_MODE</code>	System-to-switch trunking. Use a logical OR in the mode field to disable dynamic load balancing. (Layer 2 or Layer 3 only)

For system-to-system trunking, use `IP_TRUNKING_MODE`. Both systems should be setup identically. In `IP_TRUNKING_MODE`, packets are sent to the driver as large datagrams and fragmented across the active links.

For system-to-switch trunking, the default method of load balancing scheme uses Layer 3 Protocols. Layer 3 mode balances traffic across the available ports based on destination IP addresses, as opposed to Layer 2, which is based on destination MAC addresses. If you wish to use only Layer 2 protocols for load balancing, change the second element to `LAYER_2_MODE`.

For system-to-switch trunking, you can disable dynamic load balancing, which is on by default. Dynamic load balancing periodically tries to balance the load across the available members of the trunk, and is normally left enabled. Disable it by performing a logical OR of `NO_BALANCE_MODE` and either `LAYER_2_MODE` or `LAYER_3_MODE` in the mode field.

You can use any available ports in any order. Port entries need to be in order, and should not be duplicated in trunks. A trunk must consist of at least two ports, and cannot contain other trunks. You can have more than one trunk. Each trunk will have its own entry in the device table. For example, to configure two system-to-switch trunks of four ports each, use layer 3 mode, and disable dynamic load balancing, you would add two table element definitions in `if_zxeRlk.c`:

```

/*
 * Two 4-Port Fast EtherChannel Trunks, layer 3 mode with
 * balance mode disabled.
 */
zxe_rlk_ppa_list_t zxe_rlfe4a[] = {
    { RLK_TRUNK, LAYER_3_MODE | NO_BALANCE_MODE, 0,
      { 0, 1, 2, 3, -1 } },
    { 0, }
};
zxe_rlk_ppa_list_t zxe_rlfe4b[] = {
    { RLK_TRUNK, LAYER_3_MODE | NO_BALANCE_MODE, 0,
      { 4, 5, 6, 7, -1 } },
    { 0, }
};

```

You would also need to add the entries to the `zxe_rlk_config_tbl[]` in `if_zxeRlk.c`:

```

zxe_rlk_ppa_list_t *zxe_rlk_config_tbl[] = {
    zxe_rlss4,          /* 0 */
    zxe_rlfe4,         /* 1 */
    zxe_rlff4,         /* 2 */
    zxe_rlss2,         /* 3 */
    zxe_rlfe2,         /* 4 */
    zxe_rlff2,         /* 5 */
    zxe_rlfe4a,        /* 6 new first trunk */
    zxe_rlfe4b,        /* 7 new second trunk */
    0
};

```

Add two END table entries for `zrl0` and `zrl1`, and bring up the network interfaces.

Examine `configZxe.c` and implement the method that best fits your applications needs to bring up the network interfaces.

### **Advanced Failover**

You can create link failover groups, assign ports and trunks to these groups and select a modes of failover. Failover groups are configured in the same manner as trunks: Edit the `if_zxeRlk.c` file and rebuild the kernel.

Acceptable modes for failover groups are:

Mode:	Appropriate for:
-------	------------------

timeout_mode <time>	Failover mode; if no traffic is received on the ACTIVE port within the specified time, failover. Time is in milliseconds
hub_mode	Only receives on the ACTIVE port. Use when connecting failover ports in the same collision domain, like to a hub. Do not use when connecting system to system, or system-to-switch.
snap_bcast_mode	Send one SNAP packet out ACTIVE port upon failover. Use this option to help the connection on the other end recognize the change and update routing information.

Failover groups can contain individual ports, trunks, or both ports and trunks. Always configure trunks before failover groups. Add an entry for each failover group containing the list of ports and trunks to be included in that group. The default mode of failover is fast failover. In this mode, RAINlink moves the traffic over to a redundant stand-by link in case of a link failure in as little as 500 milliseconds.

In addition to fast failover, timeout mode can be enabled for a failover group. By enabling timeout mode, if no traffic is received within the specified time interval, the active port is automatically switched to another available link. You can choose different failover modes for different groups. You can also specify different timeout intervals. To build a failover group of two trunks, first build the trunks, then the failover group and set the timeout mode on with a value of 30 seconds:

```

/*
 * 2-Port Failover of 2 2-Port Trunks, timeout=30 seconds
 */
zxe_rlk_ppa_list_t zxe_rlff2ss2[] = {
    { RLK_TRUNK, IP_TRUNKING_MODE, 0, { 0, 1, -1 } },
    { RLK_TRUNK, IP_TRUNKING_MODE, 0, { 2, 3, -1 } },
    { RLK_FAILOVER, TIMEOUT_MODE, 30000, { 0, 2, -1 } },
    { 0, }
};

```

This results in one RAINlink device. You would also need to add the entries to the `zxe_rlk_config_tbl[]` in `if_zxeRlk.c`:

```

zxe_rlk_ppa_list_t *zxe_rlk_config_tbl[] = {
    zxe_rlss4,          /* 0 */
    zxe_rlfe4,          /* 1 */
    zxe_rlff4,          /* 2 */
    zxe_rlss2,          /* 3 */
    zxe_rlfe2,          /* 4 */
    zxe_rlff2,          /* 5 */
    zxe_rlff2ss2,      /* 6 new failover */
    0
};

```

Add an END table entry for `zrl0`, and bring up the network interface.

Examine `configZxe.c` and implement the method that best fits your applications needs to bring up the network interfaces.



## ZNYX NETWORKS VxWorks Utilities and Sample Routines

RAINlink includes the following files and utilities to aid in configuration and monitoring:

Name	Description
zxeUp	Initializes the driver and the device
znbEndLoad	Initialize the NetBlaster znb driver and device
zrlEndLoad	Initialize the RAINlink zrl driver and device
zxeVersion	Displays the driver version
rlConfigShow	Displays the current RAINlink configuration
zxePortShow	Displays the current port status/statistics
zxeDown	Brings down the driver
zxeAllMultiSet	Set Promiscuous Multicast Mode
ZxeAllMultiClear	Clears Promiscuous Multicast Mode

This section includes Unix-style man pages to be used as reference. The file `zxeConfig.c` contains the sample routine, `zxeUp()`, that loads and starts the `zxe` driver for a specified device using the default configuration and a sample routine, `zxeDown()`, that stops and unloads the `zxe` driver for that device. To include in the kernel, copy the file to the `$(WIND_BASE)/target/config/$(BSP)` directory and include it in `sysLib.c`

### zxeUp()

<b>NAME</b>	<code>zxeUp()</code> – Initializes the driver and the device.
<b>SYNOPSIS</b>	<pre> int zxeUp(     char *devString,          /* device to initialize, e.g. */                              /* "znb0", "zrl1" */     char *pAddr,             /* IP address of the interface */     ULONG iMask,             /* IP netmask of the interface */     int mediaType,           /* media type, 0-7 */     int ppa   index          /* Specifies the ppa or configuration */                              /* index to use for RAINlink */ ) </pre>
<b>DESCRIPTION</b>	<p>This routine initializes the driver and the device to the operational state.</p> <p>The following device strings are acceptable:</p> <pre> "znb<i>n</i>"          /* NetBlaster device <i>n</i>=0-31*/ "zrl<i>n</i>"          /* RAINlink device <i>n</i>=0-31*/ </pre> <p>The following media types are defined:</p> <pre> MEDIA_AUTO      0    /* auto negotiation */ MEDIA_TP        1    /* 10 Mbit - half duplex */ MEDIA_BNC       2    /* not supported */ MEDIA_AUI       3    /* not supported */ MEDIA_TPFD     4    /* 10 Mbit - full duplex */ MEDIA_TX        5    /* 100 Mbit - half duplex */ MEDIA_TXFD     6    /* 100 Mbit - full duplex */ MEDIA_T4       7    /* not supported */ </pre>
<b>RETURNS</b>	OK or ERROR

## **znbEndLoad() zrlEndLoad()**

**NAME** **znbEndLoad, zrlEndLoad** - initialize the driver and device

**SYNOPSIS**

```
END_OBJ * znbEndLoad (  
  char *initString, /* device specific parameters */  
  void *pBsp /* not used by the driver */  
)  
END_OBJ * zrlEndLoad (  
  char *initString, /* device specific parameters */  
  void *pBsp /* not used by the driver */  
)
```

**DESCRIPTION** Initializes the driver and the device to the operational state. All of the device specific parameters are passed in the *initString*. The void pointer parameter is currently not used by the driver.

*initstring* format:

*unit:ppa|index:memBase:mediaType:recvBufs:xmitBufs:staticBufs:flags*

<i>unit</i>	- The device unit to initialize
<i>ppa index</i>	- The <i>ppa</i> or configuration record to use.
<i>memBase</i>	- Main memory base as seen from PCI bus.
<i>mediaType</i>	- Media type of the device.
<i>recvBufs</i>	- The number of receive buffers per <i>ppa</i>
<i>xmitBufs</i>	- The number of transmit buffers per <i>ppa</i>
<i>staticBufs</i>	- The number of static buffers per <i>ppa</i>
<i>flags</i>	- Defines various interface flags.

The following *mediaTypes* are defined:

<b>MEDIA_AUTO</b>	<b>0</b>	<b>/* auto negotiation */</b>
<b>MEDIA_TP</b>	<b>1</b>	<b>/* 10 Mbit - half duplex */</b>
<b>MEDIA_BNC</b>	<b>2</b>	<b>/* not supported */</b>
<b>MEDIA_AUI</b>	<b>3</b>	<b>/* not supported */</b>
<b>MEDIA_TPFD</b>	<b>4</b>	<b>/* 10 Mbit - full duplex */</b>
<b>MEDIA_TX</b>	<b>5</b>	<b>/* 100 Mbit - half duplex */</b>
<b>MEDIA_TXFD</b>	<b>6</b>	<b>/* 100 Mbit - full duplex */</b>
<b>MEDIA_T4</b>	<b>7</b>	<b>/* not supported */</b>

The following *flags* are defined:

**END\_NOCOPY\_ON\_TRANSMIT**    **0x100**

**RETURNS** An END object pointer or NULL on error.

### **zxeVersion()**

---

<b>NAME</b>	<i><b>zxeVersion()</b></i> – returns the version of the znb driver
<b>SYNOPSIS</b>	<b>int zxeVersion ()</b>
<b>DESCRIPTION</b>	Returns the version of the driver
<b>RETURNS</b>	OK or ERROR

### **zxePortShow()**

---

<b>NAME</b>	<i><b>zxePortShow()</b></i> – Displays current port statistics
<b>SYNOPSIS</b>	<b>int zxePortShow (</b> <b>int port</b> /* <b>port == device unit</b> */ <b>)</b>
<b>DESCRIPTION</b>	This routine displays the current status and statistics for the designated <i>port</i> . <i>port</i> refers to the specific hardware port, not to a zrl znb unit.
<b>RETURNS</b>	OK or ERROR

### **rlConfigShow ()**

---

<b>NAME</b>	<i><b>rlConfigShow()</b></i> – Displays current RAINlink configuration
<b>SYNOPSIS</b>	<b>int rlConfigShow()</b>
<b>DESCRIPTION</b>	This routine displays the current RAINlink configuration
<b>RETURNS</b>	OK or ERROR

### **zxeDown()**

---

<b>NAME</b>	<i><b>zxeDown()</b></i> - Stop and unload an instance of the driver.
<b>SYNOPSIS</b>	<b>int zxeDown(</b> <b>int unit</b> /* <b>device unit, 0-31</b> */ <b>)</b>
<b>DESCRIPTION</b>	This routine brings down an interface and stops and unloads the driver and device for that interface.
<b>RETURNS</b>	OK or ERROR

### **zxeAllMultiSet()**

---

<b>NAME</b>	<i><b>zxeAllMultiSet()</b></i> - Set Promiscuous Multicast on selected unit
<b>SYNOPSIS</b>	<b>int zxeAllMultiSet (</b> <b>char *name</b> /* <b>device name</b> */ <b>)</b>
<b>DESCRIPTION</b>	This routine sets promiscuous multicast on the selected unit. String format is "znb(x)" or "zrl(x)" e.g. "znb1", "zrl12", etc. Strict string checking is performed against the name parameter.
<b>RETURNS</b>	OK or ERROR

## **zxeAllMultiClear ( )**

---

<b>NAME</b>	<b><i>zxeAllMultiClear ( )</i></b> - Clears Promiscuous Multicast on selected unit
<b>SYNOPSIS</b>	<b>int zxeAllMultiClear( char *name /* device name */ )</b>
<b>DESCRIPTION</b>	This routine clears promiscuous multicast on the selected unit. String format is "znb(x)" or "zrl(x)" e.g. "znb1", "zrl12", etc. Strict string checking is performed against the name parameter.
<b>RETURNS</b>	OK or ERROR

### ***Changes Since Last Release***

- Added support for newer revisions of ZNYX adapters that will use the Intel 21152-BB, Intel 21154-BE, and 21555-AA bridge chips.
- Added new Failover mode: SNAP\_BCAST\_MODE. Enable to send a SNAP packet out the ACTIVE port upon failover.
- Added support for Tornado AE PPC.

### ***Discrepancies***

- SNAP\_BCAST\_MODE requires that HUB\_MODE be enabled.
- RAINlink trunks with invalid ports are not handled correctly. Properly configured systems should not be affected.

## ***Where to go for Technical Support***

<b>Resource</b>	<b>Address</b>
Telephone	(510) 249-0800
Toll-Free	(800) 724-0911 (USA Only)
FAX	(510) 656-2460
Website	<a href="http://www.znyx.com">www.znyx.com</a>
E-mail	<a href="mailto:support@znyx.com">support@znyx.com</a>



ZNYX Networks  
48421 Milmont Drive  
Fremont, CA 94538 USA

Tel.: (510) 249-0800  
Toll-Free: (800) 724-0911  
Fax: (510) 656-2460  
E-mail: [sales@znyx.com](mailto:sales@znyx.com)  
Website: [www.znyx.com](http://www.znyx.com)

**Document # DC0116-08**  
**Version: March 26, 2002**

© 2002 ZNYX NETWORKS Corporation. All rights reserved worldwide. Information in this document is subject to change without prior notice. ZNYX, RAIN, RAINlink, RAINlink for VxWorks, and NetBlaster are trademarks or registered trademarks of ZNYX NETWORKS Corporation in the United States and/or other countries. All other marks, trademarks or service marks are the property of their respective owners.

ZNYX NETWORKS may have patents, pending patent applications, trademarks, copyrights or other intellectual property rights covered in the subject matter of this document. By furnishing this document, ZNYX NETWORKS does not license nor waive its license to those intellectual property rights except as expressly provided in a written license agreement from ZNYX NETWORKS. Information in this document is subject to change without prior notice.