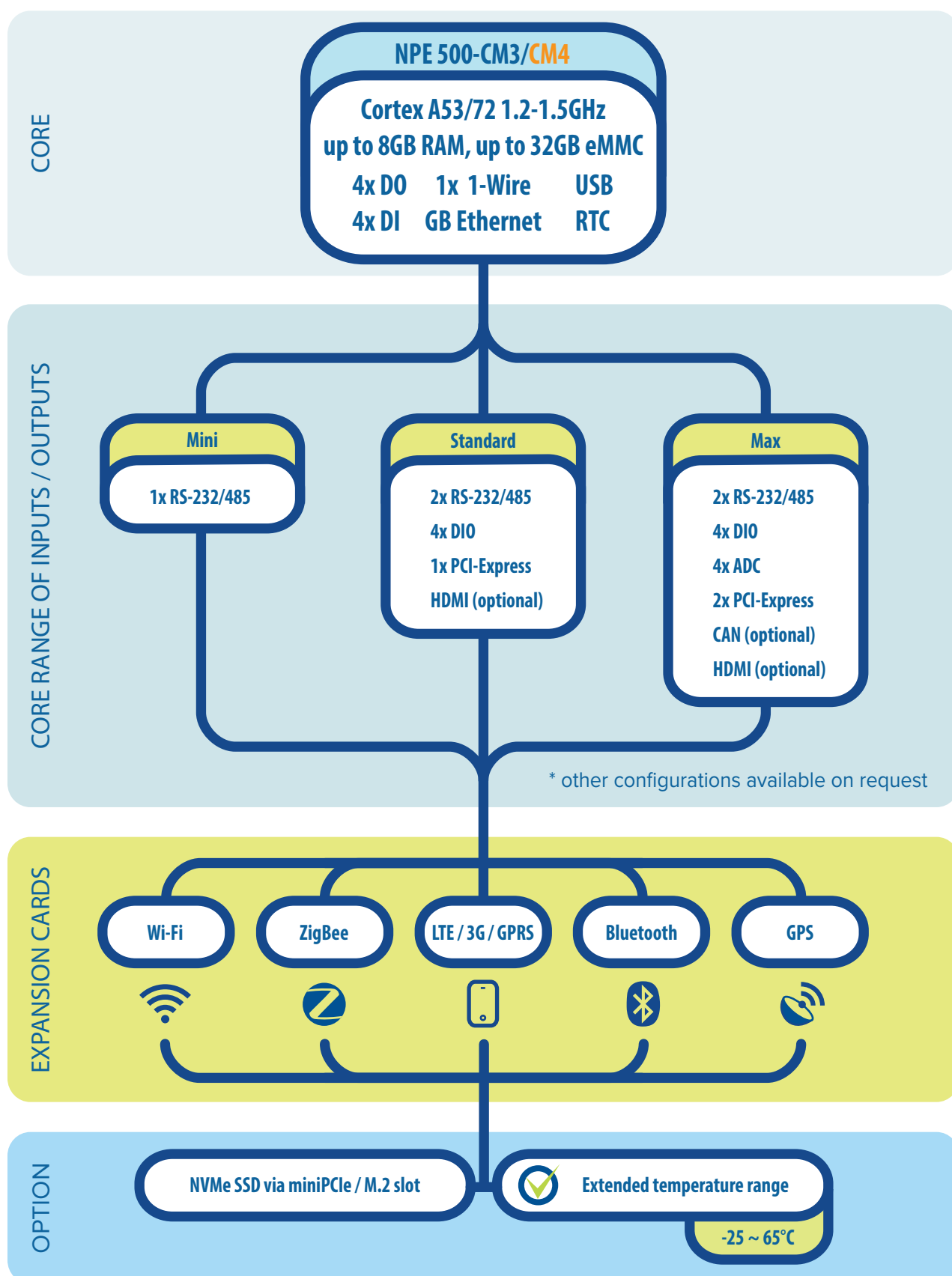


NPEX500 series

USER MANUAL

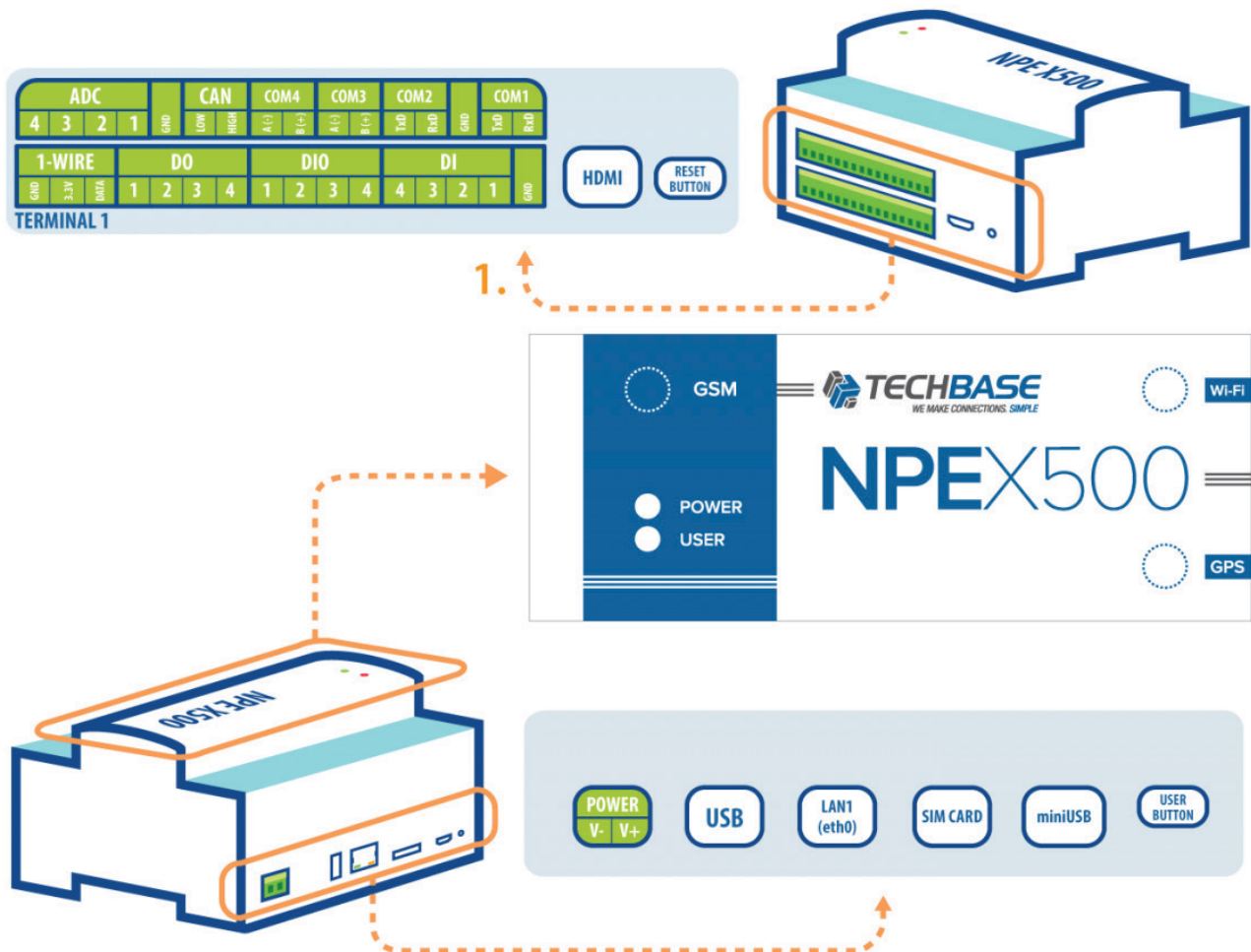
Index

1. NPE X500 Versions	3
2. NPE X500 Quickstart	5
3. Detecting device	10
4. Useful commands	14
5. Building and deploying kernels	15
5.1. Linux version support	16
5.2. Install raspbian system on the X500/9500	18
5.3. Kernel building	20
6. Troubleshooting	22
6.1. apt-get upgrade - system stuck on boot	22
7. How to install TECHBASE addons	24
8. Rescue mechanism	25
8.1. X500 CM3 restore with builtin rescue system	26
8.2. X500 CM3 restore with flash drive	27
8.3. Backup using mini USB port	30
8.4. Backup using additional system from flash drive	33
8.5. Backup main system using built-in rescue system	35
9. Communication via the Ethernet port	37
10. WiFi connection	40
11. WMBUS module	42
12. GSM Connection	43
12.1. ZTE ME3630-U1A	49
12.2. SIM-7500JE	51
12.3. SIM-7000 GPS	53
13. Processor board programming using Raspberry Development Kit	55
14. Processor board programming using X500 device	59
15. Software update	62
16. Date and time settings	65
17. Hardware	70
17.1. List of hardware components	73
17.2. Mapping GPIO ports	74
17.3. I2C interface	77
17.4. Inputs/outputs	79
17.4.1. GPIO LIB for C	87
17.4.2. Detect event on the GPIO by RPi.GPIO	95
17.5. Serial ports	96
17.5.1. Service port	101
17.6. LEDS/BUZZER	102
17.7. CAN bus	104
17.8. TPM 2.0	106
17.9. OneWire	107
17.10. Analog input	112
17.11. Hardware Watchdog	116
17.12. Fixing DO/DI ports handling	119
17.13. OLED Display	120
17.14. SPI - outside	122
18. Autostart - Preparing application to run in autostart	123
19. Ramdisk package	126
20. Extended security package	128
21. iModCloud package	130
22. CODESYS	132



NPE X500 QuickStart

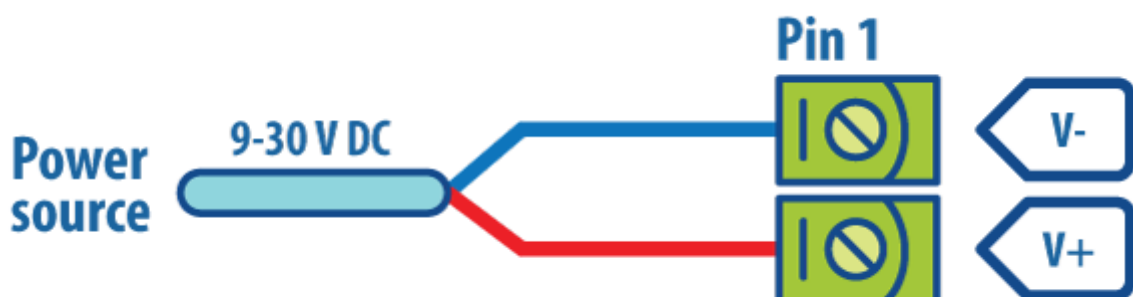
Hardware startup



1. HDMI, DI/DO, 1-Wire, RS, CAN, AI Connectors + RESET button
2. GSM antenna connector
3. LED indicators
4. Power connector, USB, Ethernet ports, SIM Card slot, Audio output port, USER button

Powering up

The first step is to connect the power supply.



LED indicators

- **POWER** - Continuously lit when device is turned on
- **USER** - turned off by default - configurable by user

LEDs support RGB color configuration

Connect to the device

Check IP address

There are 2 ways to connect with device using ETH interface

- directly connected to the PC:
 - The default IP Address is **192.168.0.101** - make sure that IP address of your PC is in the same subnet.
- connected to the LAN network:
 - Device will get IP Address from DHCP server. Use SearchNPE to detect NPE X500 devices on the network and to determine IP address. Find the application at http://www.a2s.pl/products/npe/SearchNPE_PZA2S.zip.

Legend:



Network connection via Ethernet port required



Network connection via Ethernet port not required

Login to the device

If the device is connected to the network and you know IP address, you can connect using TCP port 22



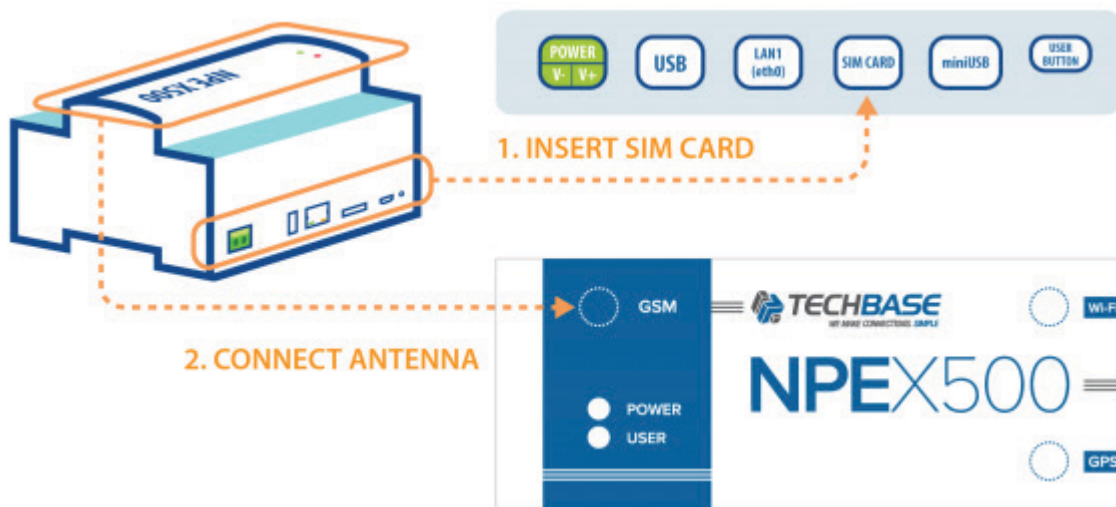
(default in SSH) e.g., using PuTTY in Windows or `ssh` command in Linux.

You can find PuTTY at following address: <http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe>

You can use following credentials to login.

*	Login	Password	Permissions
1.	user	user	User permissions
2.	root	techbase	Root permissions

3G Connection



To establish connection using gprs package, do the following:

- Insert SIM Card into the slot on the side of the device
- Connect antenna to connector on the top of the device
- Connect via ssh to Modberry
- Use `sudo su`
- Run this command: `softmgr update gprs`
- after installation, execute this command:

```
gprs connect
```

- Connection should be established automatically

gprs abort should be used if anything goes wrong *gprs status* return the actual status of the connection *gprs disconnect* disconnects the connection

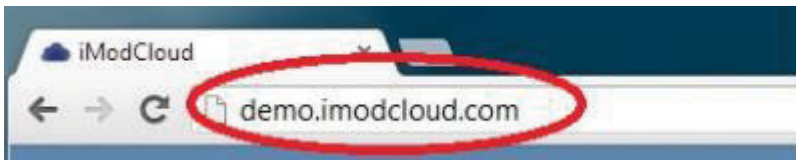
Software Startup

iModCloud

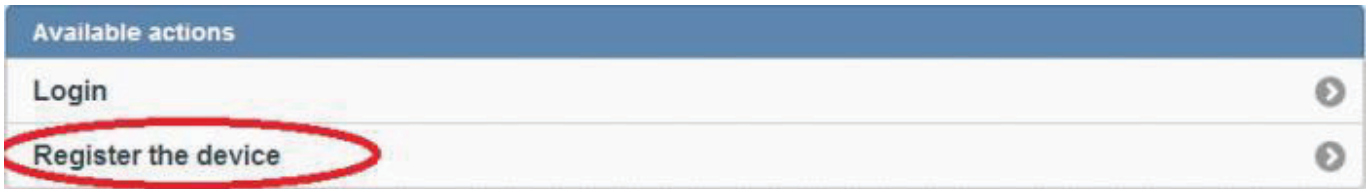
In order to register iMod/NPE X500 in iModCloud make sure your device is connected to the Internet via Ethernet port described in section 1.



- Type **demo.imodcloud.com** in your web browser

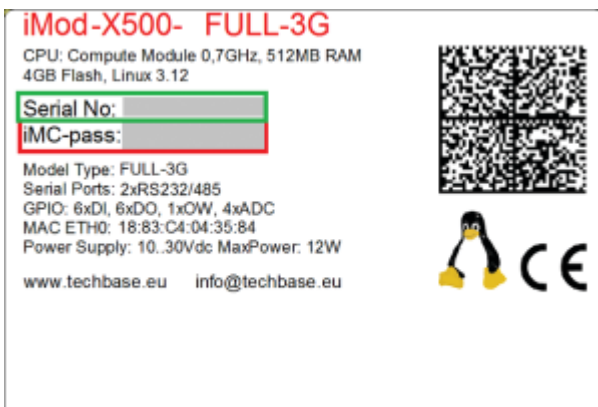


- Click **Register your device**

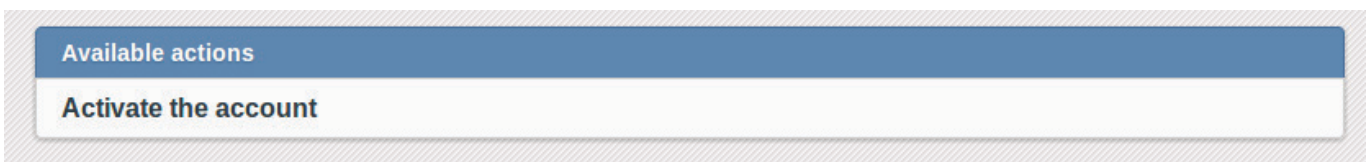


- Enter the register data:


- You will find register data on the label:



- **Activate your account:** activate your iModCloud account by clicking the button and filling in required information for your account:



- After registering the device and activating the account you will see a similar view:

status	beep	ping	Odśwież parametry
2014-05-21 10:41			
Device is online			
Nazwa	Demo Unit #1		
Device ID	30000058		
Serial No	1883C40422C0		
Model	iMod-9400RW-3G		
VPN IP	10.8.55.3		
ETH IP	192.168.7.134		
GPRS IP	46.215.77.195		
STATUS			
NX Dynamics	Open		

- You can check if your device is properly connected by going to: Devices → *YourDevice* and clicking Ping or BEEP button at the top



If you have any problems with logging to iModCloud, don't hesitate to contact us.

Managing software packages

Please do not run *apt-get upgrade* command.
It can disturb running of the unit including complete loss of connection to it.
This system is compiled by TECHBASE and uses u-boot to start the unit.
apt-get upgrade overrides this configuration.

Following commands are to be used in terminal.

How to check what packages are installed

```
softmgr list
```

How to check for updates


```
softmgr checkall
```

How to update all packages

```
softmgr update all
```

How to update iMod package to the newest version


```
softmgr update imod_tiger -b x500-beta
```


Rescue mode

Instructions for rescue mode can be found [here](#)

Contact

- For further information on NPE and related products visit: <http://www.techbase.eu/en>
- The A2S.pl online catalog offers hardware and software products for purchase: <http://www.a2s.pl/en>
- For more information on iModCloud visit: <http://www.imodcloud.com/en/>

-  info@a2s.pl

-  +48 58 345 39 22

Detecting device

1. Introduction

This document describes how to find and connect to the device

2. Connect via RS terminal

Connect RS-232 port/converter to COM1 port.
Then connect to it using following parameters:

- baudrate: 115200
- databits: 8
- parity: none
- stopbits: 1

Start the device, you should see login prompt after a while:

```
picocom -b 115200 /dev/ttyUSB0
picocom v1.7

port is          : /dev/ttyUSB0
flowcontrol      : none
baudrate is      : 115200
parity is        : none
databits are     : 8
escape is        : C-a
local echo is    : no
noinit is        : no
noreset is       : no
nolock is        : no
send_cmd is      : sz -vv
receive_cmd is   : rz -vv
imap is          :
omap is          :
emap is          : crcrlf,delbs,

Terminal ready

raspberrypi login:
```

You may use any other serial port client.
In Windows it can be for example putty.

Service port needs to be enabled for this. Refer this document: [Service port](#)

3. Connect via SSH

To connect to device via ssh use some SSH client:

- *ssh* command on Linux
- putty on Windows

Default login and passwords (not all may be available on some configurations):

- pi/raspberry
- user/user
- root/techbase

You need to know IP address of the device.

4. Detecting IP address

There are several ways to find IP address of the device.

Note that not all methods may be available (for example not all devices have HDMI port).

1. using serial terminal

Connect to the device via serial, login and execute *ifconfig*

2. using HDMI

Connect monitor and keyboard, login, start terminal and execute *ifconfig*

3. using SearchNPE

It won't work with devices with bare Raspbian system.

- Install SearchNPE utility (check: [here](#))
- Start tool
- Choose network range ('*' character means 'any')
- Press start
- Tool will display all found devices

Network range 192.168.0.*

If some information is not displayed ("---") it means that LIBNPE on detected NPE does not support this. Please update LIBNPE. NPE listed in red color indicates that they are loaded from database.

	IP Address	MAC	NPE ID	Kernel	NxDynamics	iMod	Beeper
1	192.168.0.101	00:00:00:69:69:72	test_1_1	4.4.45-v7+ #19	NxDynamics	iMod TRM	Beep
2	192.168.0.133	00:00:00:00:00:fe	test_1_1	4.4.45-v7+ #19	NxDynamics	iMod TRM	Beep
3	192.168.0.149		test_1_1	4.19.75-v7+ #127	NxDynamics	iMod TRM	Beep
4	192.168.0.148	00:00:00:44:44:44	test_1_1	4.4.45-v7+ #19	NxDynamics	iMod TRM	Beep
5	192.168.0.124			3.19.3+ #35	NxDynamics	iMod TRM	Beep
6	192.168.0.225	00:00:00:00:00:21	TESTER_5	4.4.45-v7+ #19	NxDynamics	iMod TRM	Beep

Search

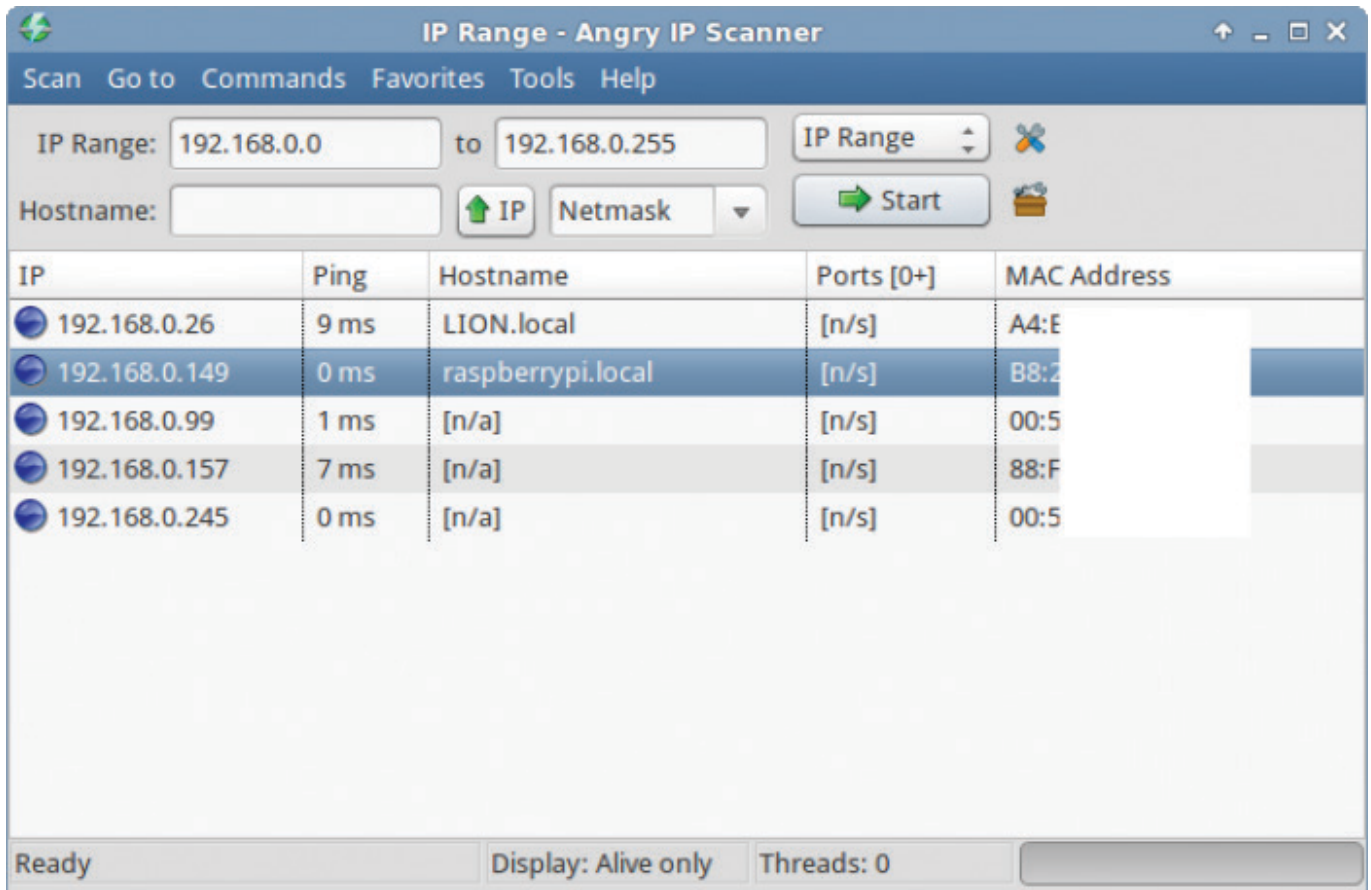
Load Save Clear Database Clear List

Legend: Green color - device is saved and was detected Gray color - device is saved but not detected

Visit TechBase website www.a2s.pl

4. using some IP scanner

- Find some IP scanner (for example Angry IP scanner <https://angryip.org/>).
- Install it and run.
- Then choose IP range and press *Start*.
- Program will display all found devices
- You can find your device by MAC
- Hostname for device would be *raspberrypi.local*



5. using router

- login to administration page of your router
- go to page with all available devices
- you can find device by MAC or hostname (*raspberrypi.local*)

Useful commands

1. Introduction

This document present some usefull commands for x500 devices.

2. Commands

Command	Description
<code>ifconfig</code>	check IP and MAC addresses of all enabled interfaces, more about network here: Communication via the Ethernet port
<code>lsusb</code>	List available USB devices
<code>lsblk</code>	List available block devices
<code>df -h</code>	Check free spaces of mounted file systems in readable format
<code>free -m</code>	Check amount of free memory in megabytes
<code>ls -la /sys/class/tty/*/device/driver awk '{print \$9}' awk -F '/' '{print \$5}'</code>	List available RS ports
<code>less /var/log/syslog</code>	Read syslog file
<code>dmesg</code>	Read kernel log
<code>service_port_ctrl status</code>	Check status of service port COM1 (ttyAMA0)
<code>service_port_ctrl on ; reboot</code>	Enable service port COM1 (ttyAMA0)
<code>service_port_ctrl off ; reboot</code>	Disable service port COM1 (ttyAMA0)
<code>service <service_name> status</code>	Get status of <service_name> service
<code>service <service_name> start</code>	Start <service_name> service
<code>service <service_name> stop</code>	Stop <service_name> service

Building and deploying kernels

1. Introduction

This documents describe:

- how to install clean Raspbian on X500/9500,
- linux version support for X500/9500,
- how to compile own kernel and driver.

If you don't see some documents or you don't have a permission to see it, please contact with our sales team.

2. X500/9500-M3 (compute module 3)

- [Linux version support](#)
- [Install raspbian system on the X500/9500](#)
- [Kernel building](#)

3. X500/9500 (compute module 1)

- [Building and deploying kernels](#)

Linux version support

1. Introduction

This documents describe the linux versions with information what is working in standard Raspbian system and our own compilation based on the standard Raspberry Pi repository. We have also listed a required files to set up, to have possibility to build your own distribution, like a:

- device tree
- dt-bloob.bin
- config.txt
- zImage/kernel7.img

1.1. Device tree (DT)

The [device tree](#) is a data structure for describing hardware, which originated from Open Firmware. The data structure can hold any kind of data as internally it is a tree of named nodes and properties. Nodes contain properties and child nodes, while properties are name-value pairs.

We had to compile own device tree to have a out of the box working RTC, ADC, CAN and SC0 serial port. In below table you can download ready file to use it in your own distribution, please remember that TECHBASE not warranty that file will be working properly in the system which is not described below. The compilation can be made by dtc which is described on the [raspberry page](#)

1.2. dt-bloob.bin

Dt-bloob.bin is a file to setup the startup state and function for all GPIO which is provided by CPU. If port is not setup then have a default behaviors. The compilation can be made by dtc which is described on the [raspberry page](#)

1.3. config.txt

In the default, the raspberry do not have turn on the i2c interface, also the default is i2c1, not i2c0 which is used in X500/9500. To setup this behavior please add below line for config.txt file

```
core_freq=250
dtparam=i2c_arm=on
dtparam=i2c0=on
i2c0_baudrate=400000
```

1.4. zImage/kernel7.img

The [kernel](#) is a computer program that is the core of a computer's operating system, with complete control over everything in the system.

We are providing this file in two version, the difference between them is described below:

- zImage: a compressed version of the Linux kernel image that is self-extracting.
- kernel7.img: a non compressed version of the Linux kernel image

2. Known issues with standard Raspbian

2.1. Serial ports

Please follow the document [Serial port](#) to know how serial ports are organized and what functions they perform in X500/9500 device. Up to this time the standard raspbian do not have a support for RS-485 for AMA0 port. You have to add your own modification to the driver and recompile the kernel to have working this port in two modes. The driver is located in [/driver/tty/serial/amba-pl011.c](#). This is

main issue why we need to release own kernel system. If you want to see our modification please contact with your sales team and click on this link [Linux Serial port - AMA0](#)

2.2. Drivers for expansion cards

We have to use some additional driver which is not provided in the default by raspbian e.g. ADS1115, MCP23017, MCP23008. This files can be provide like a modules or build-in in the kernel. This is not a big issue because we can just copy the driver to new system, in many times it will be working good, if not then we need to recompile the modules.

2.3. SC16IS7XX.C on Linux 4.9

We found a bug in this driver which caused that RTS line was always on, in this way the 232 mode was can't work.

3. Supported Linux version

Linux version	Release date	Last update	Debian version	Debian name	COM1(RS232)/COM4(RS485) - ttyAMA	COM2(RS232)/COM3(RS485) - ttySC0	GPIO	RTC	ADC	Ethernet	Kernel	DT	dt-blob.bin	config.txt
4.4.45-v7+ (own compilation)	2017-02-10	2017-12-06	8	Jessie	OK	OK	OK	OK	OK	OK	linux4.4_kernel.zip	linux4.4_dt.zip	dtblob.zip	config.zip
4.4 (from Raspberry)	2017-01-10	2017-12-06	8	Jessie	only COM1	OK	OK	OK	OK	OK	from Raspberry	from Raspberry	dtblob.zip	from Raspberry
4.9.65-v7+ (own compilation)	2017-11-27	2017-12-06	9	Stretch	OK	OK (with fixed bug in sc16is7XX driver) sc16is7xx.zip	OK	OK	OK	OK	linux4.9_kernel.zip	linux4.9_dt.zip	dtblob.zip	config.zip
4.9 (from Raspberry)	2017-09-07	2017-12-06	9	Stretch	only COM1	OK (with fixed bug in sc16is7XX driver) sc16is7xx.zip	OK	OK	OK	OK	from Raspberry	linux4.9_dt.zip	dtblob.zip	from Raspberry
4.14 (from Raspberry)	2018-11-13	2018-11-13	9.4	Stretch	only COM1	-	OK	OK	OK	OK	from Raspberry	device-tree_4.14	dtblobs_4.14	from Raspberry
4.14.98-v7+ (own compilation)	2019-02-27	2019-02-27	9.4	Stretch	only COM1	OK	OK	OK	OK	OK	kernel_4.14	device-tree_4.14	dtblobs_4.14	config_4.14
4.19.64-v7l+ (own compilation for M500 RPI4)	2019-08-12	2019-08-12	10	Buster	OK	OK	OK	OK	OK	OK	kernel_4.19	device-tree_4.19	dtblobs_4.19	config_4.19
4.19.75-v7l+ (own compilation for X500/9500)	2019-04-01	2019-04-01	10	Buster	OK	OK	OK	OK	OK	OK	kernel_4.19	device-tree_4.19	dtblobs_4.19	config_4.19

Install raspbian system on the X500/9500

1. Introduction

This documents describe how install raspbian operation system on the X500 and 9500 device. In this case the different between the platform is only USB connector which is used for flash the CPU.

- In X500 you can find this connector on the same side like a power supply, it's called "SRV".
- In 9500 usb connector to flash is located at the front where you can see the leds, it's called "service cable".

2. Flash CPU

2.1. Download image

Firstly, please download the clean Raspberry OS from raspberry page e.g. <http://downloads.raspberrypi.org/raspbian/images/raspbian-2017-01-10/>. You need to be careful because in old images the size is too big and you need to mount this image on some other machine and make a re-size. The maximum size is 4GB, this is limit for eMMC which is installed on the CPU.

2.2. Connecting the device to the PC

Please connect the USB connector from the device to the PC, after this please power on the power supply.

2.3. Preparing the environment

Next step is install additional software which is allow us to boot and flash the CPU. Please follow this documents <https://www.raspberrypi.org/documentation/hardware/computemodule/cm-emmc-flashing.md>

If CPU is not bootable please re-power the power supply.

3. TECHBASE file

The last step is copy below files from old device to the new image, without this the device will not support all features e.g. RS-485 on AMA0(COM4), CAN, ADC, RTC, RS232/485 on SC0(COM2/COM3).

If you want to use our config.txt, you need to remember that probably you need to change the line with kernel variable to kernel=kernel7.img. We set up to run u-boot first, but our u-boot version is not compatibility with the newest raspbian, so your system will be stuck at the boot stage. You will see nothing on serial console or "colorful screen" on hdmi port.

- device tree
- dt-blob.bin
- config.txt, please read description above, this is very important.

- zImage/kernel7.img
- /lib/modules

You can skip this step if you want and just build your own kernel, device tree, config and dr-boob.

Kernel building

1. Introduction

This documents describe how to prepare the environment and compile the Kernel for the X500/9500 device. The document was prepared based on the official information from [Raspberry Pi](#).

2. Prepare the environment

First, you will need a suitable Linux cross-compilation host. We tend to use Ubuntu; since Raspbian is also a Debian distribution, it means many aspects are similar, such as the command lines. Next step is download TOOLCHAIN, get the sources of the Linux and prepare default configuration, which will take some time:

2.1. INSTALL TOOLCHAIN

Use the following command to download the toolchain to the home folder:

```
git clone https://github.com/raspberrypi/tools ~/tools
```

Updating the \$PATH environment variable makes the system aware of file locations needed for cross-compilation. On a 32-bit host system you can update and reload it using:

```
echo PATH=\$PATH:~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabi-hf-raspbian/bin >> ~/.bashrc  
source ~/.bashrc
```

If you are on a 64-bit host system, you should use:

```
echo PATH=\$PATH:~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabi-hf-raspbian-x64/bin >> ~/.bashrc  
source ~/.bashrc
```

2.2. GET SOURCES

To get the sources, refer to the original GitHub repository for the various branches. \$ git clone -depth=1 <https://github.com/raspberrypi/linux>

2.3. Configuration

Run the following commands, depending on your platform version.

X500/9500:

```
cd linux  
KERNEL=kernel  
make bcmrpi_defconfig
```

X500-M3/9500-M3:

```
cd linux  
KERNEL=kernel7  
make bcm2709_defconfig
```

2.3.1. Configuration

After executing `make ..._defconfig` `.config` file will appear.

This file contains configuration for kernel.

To enable `/dev/ttyS0` serial port set following keys to:

```
CONFIG_SERIAL_8250_NR_UARTS=4
CONFIG_SERIAL_8250_RUNTIME_UARTS=1
```

3. Compilation

`make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- zImage modules dtbs`

To speed up compilation on multiprocessor systems, and get some improvement on single processor ones, use `-j n`, where `n` is the number of processors * 1.5. Alternatively, feel free to experiment and see what works!

After the compilation you can copy the needed files by command

```
$base_path=/home/LINUX_FILE
cp linux/arch/arm/boot/zImage $base_path
cp linux/arch/arm/boot/dts/bcm2710-rpi-cm3.dtb $base_path
cp linux/arch/arm/boot/dts/bcm2710-rpi-cm3.dts $base_path
```

To create the `kernel.img` file from `zImage` please do follow command:

```
sudo linux/scripts/mkkn1img $base_path/zImage $base_path/kernel.img
```

To install modules please do follow command:

```
sudo make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-
INSTALL_MOD_PATH=$base_path/modules modules_install
```

apt-get upgrade - system stuck on boot

1 Troubleshooting

If your system is stuck on boot after upgrade function, probably you have one or all problems listed below:

- u-boot is not compatible with new kernel
- firmware is not compatible with the processor hardware, this is because Raspberry created a new version of the Compute Module (currently CM3 and CM3+) e.g. Linux 4.9 doesn't have a proper firmware file for CM3+.
- device-tree can be incompatible with the **u-boot**

2 Theory

TECHBASE Group added special bootloader called **u-boot** which override default boot process. In our devices, the **u-boot** run the kernel, but in order to do it, it need a **zImage** file instead of a **kernel.img**. **zImage** is a compressed version of the Linux kernel image, which is self-extracting. Function **apt-get upgrade** update kernel files, but doesn't create new **zImage**, so if you did an upgrade, you will be provided with a new kernel file, new firmware and all the files needed, e.g. device tree, but not **zImage**. The problem is that according to the time you purchased the devices, you could receive a new CM3+ but still with the system version 4.4. In this situation follow step by step instruction:

- if you reboot, the system will get stuck, because you have new device-tree and new firmware, but both is not compatible with the **u-boot**, **zImage** and CM3+
- if you switch to run by **kernel.img**, instead of **u-boot** then system again will get stuck because it has firmware version incompatible with the CM3+
- if you add the latest firmware, switch to **kernel.img** to run system successfully.
- if you still want to use **u-boot**, then you need to replace the firmware, **u-boot** and create new **zImage** file from your new kernel.img.

3 Repair

Please remember, that if you decide to change boot process and don't want to use **u-boot**, then you don't have a possibility to access the memory, if your device will be crash and you don't have an access via USB OTG!

First, you need to determine what you need to do:

- if you don't want to use **u-boot.img** then you need to download latest firmware below and also change the /boot/config.txt, change **kernel=u-boot.bin** to **kernel=kernel7.img**
- if you want to use **u-boot** but with the system, which you upgraded, then download latest firmware, **u-boot** and create **zImage** file
- if you want to get back to the default settings
 - but you have other working device then:
 - you can backup the image following this instruction [Backup using additional system from pendrive](#),

- restore on not working device following this instruction [x500 CM3 restore with pendrive manual](#) but use the image which you backup,
- but you don't have other working device, then you need to restore it following this instruction [x500 CM3 restore with pendrive manual](#) but using the image, which you can find in this instruction
- after this, please contact with TECHBASE, because we need to setup other settings.

3.1 Files to download

- [firmware files - 2020-04-07](#)
- [u-boot - 2020-04-07](#)

3.2 Preparing zImage

- download mkkimage from official repository, e.g. for 4.9 system is "wget <https://raw.githubusercontent.com/raspberrypi/linux/rpi-4.9.y-stable/scripts/mkknlimg>", it is always in directory scripts
- run command `sudo mkknlimg /boot/kernel7.img /boot/zImage`

The quickest way to access all files is via [usb otg](#), if it is not possible, then you can try to boot the system from flash drive by [u-boot \(up to point "Restore system with command"\)](#). Please remember, that if you decide to change boot process and don't want to use u-boot, then you can't repeat it in the future, if you don't have an access via USB OTG!

How to install TECHBASE addons to raspbian 4.19 image

1. Introduction

This document describe how to install TECHBASE addons to raspbian stock system in 4.19 version. Please be careful, because after this installation the system will be working different. Below is short description what will be change:

- added uboot, system will be start by uboot, not by kernel7.img directly
- config.txt will be not used anymore
- the kernel will be in version 4.19.75-v7l+
- it will be added new dt-blob
- device-tree will be changed
- it will be added new modules which must by compatybility with the kernel version
- the pi user will be deleted
- the overlay will be move to handling by uboot

2. Files

- core <https://drive.google.com/file/d/1uPcOI6VclU2EYBFkXc8jg2gNpzKQu2S7/view?usp=sharing>

3. Instalation

X500 Rescue Mechanisms

Recovery system works only on x500/9500 platform and X500M3/9500M3 newer than 10/2017.

To backup/recovery system on x500 CM3 platform refer: [Rescue options in NPE M3 Version](#).

NPE X500/9500 MMC contains two copies of Raspbian - recovery system and main system - which share no files. The recovery system can be used to repair or reinstall the main system placed on bigger partition. There is no difference between the two systems other than the size of their root partitions therefore all software and configurations can be easily copied and reused.

MMC memory is divided into four partitions. First one contains firmware, u-boot, rescue kernel and device tree. Second one is the rescue file system which is a very cut-down Raspbian. Third partition is the *official* /boot partition visible on normal (non-rescue) system and it contains kernel and device tree for normal system. Fourth biggest partition contains main file system.

MMC memory:

p1	p2	p3	p4
F i A r T m 3 w 2 a r e	R E x s F T o x r F p b i a n	F / A b E o x r F p b i a n	R a s p b i a n

To enter rescue mode:

- press and hold *User Button* (or *SW* for NPE-9500)
- boot the system;
- release the button after 5s;
- system should now boot into rescue mode.

Use following login credentials:

```
login: root
password: techbase
```

Note: service port (serial console on COM1) is always enabled in rescue mode.

To restore factory settings please use this instruction: [X500 Rescue main system](#)

For create backup of main system: [X500 Backup main system](#)

x500 CM3 restore with builtin rescue system

Step by step instructions

1. Prepare pendrive with system image

- Prepare min. 4GB pendrive and connect it to the PC
- Download latest **Full system images** from [here](#)
- Copy downloaded image to the pendrive

2. Run device in Rescue Mode

Before proceeding this step you need to find USER Led and USER button - you need this to enter to the rescue system

- Power your device and look at USER LED (green)
- When USER LED start blinking, please press and hold USER Button until USER LED stops blinking
- Log in to the system:

```
login: root
password: techbase
```

- Restore system with command:

```
restore_main_system --auto-mode
```

This command will flash internal mmc memory with image located on pendrive

If you want more options execute:

- After flashing shutdown system with command:

```
halt
```

- Unplug pendrive and press reset button or unplug and plug it again

x500 CM3 restore with pendrive manual

Step by step instructions

- Prepare pendrive with at least 8GB of capacity
- Connect pendrive to computer
- Get pendrive dev path. You can use for example lsblk utility for this

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
sda	8:0	0	111,8G	0	disk	
├sda1	8:1	0	102,5G	0	part	/
├sda2	8:2	0	1K	0	part	
└sda5	8:5	0	9,4G	0	part	[SWAP]
sdb	8:128	1	14,5G	0	disk	
└sdb1	8:129	1	14,5G	0	part	

In this example our pendrive has path: `/dev/sdb`

You can recognize your pendrive with its size or by running this command twice: before plugging and after.

- Download **rescue pendrive image** from [here](#)
- Unpack zip downloaded zip

Zip file contains image which has 5GB size. If you couldn't unpack it with your zip unpacker, try doing that with 7-zip

- Unmount all partitions from pendrive if necessary
- Flash pendrive with dd command using unpacked image

```
dd if=image.img of=/dev/<pendrive_name> bs=4MiB
```

- After flashing unplug pendrive and plug it
- Download x500 CM3 system image from [here](#)
- Mount third partition with 5.4GB of size
- Copy downloaded image to this partition
- Unmount pendrive
- Connect monitor, keyboard and prepared pendrive to the x500 device
- Run the device
- Press any key when message "Hit any key to stop autoboot: " will appear on the screen
- Execute following commands:

```
setenv boot_targets usb0 mmc0 pxe dhcp
saveenv
reset
```

These commands set uboot to boot from pendrive first instead of internal mmc memory

- Wait for rescue system to boot
- Log in to the system:

```
login: root
password: techbase
```

- Restore system with command:

```
restore_system --auto-mode
```

This command will flash internal mmc memory with image located in third partition of pendrive
If you want more options execute:

```
restore_system --help
```

- After flashing shutdown system with command:

```
halt
```

- Unplug pendrive and press reset button or unplug and plug it again

[Troubleshooting](#)

Rescue system doesn't start from pendrive

Synoptics

- Booting stops on uboot and with trying to read pendrive
- U-Boot displays 0 pendrive storages during start procedure

Solution

- Use another pendrive

Restore process breaks with message device busy

Synoptics

- Boot partition restore process breaks with message similar to:

```
Starting backup of boot files
mount: /dev/mmcblk0p1 is already mounted or /mnt/boot_4v0akF busy
       /dev/mmcblk0p1 is already mounted on /boot
Failed to mount BOOT partition
```

Solution

- Umount /boot directory manually and try again:

```
umount /boot  
restore_system --auto-mode
```

Restore process breaks with message error copying files

Synoptics

- Boot partition restore process breaks with message similar to:

```
Copying files to /mnt/boot_SjRkaj  
cp: cannot create regular file '/mnt/boot_SjRkaj/tmp_boot.cmd_servic_on':  
Read-only file system  
cp: cannot create regular file '/mnt/boot_SjRkaj/tmp_boot.cmd_servic_off':  
Read-only file system  
Error copying files
```

Solution

- Umount /boot directory manually and try again:

```
umount /boot  
restore_system --auto-mode
```

Backup using mini USB port

1. Introduction

Whole MMC memory can be backedup by connecting device to PC and executing appropriate commands.

Backup file created with this method can be then restored using one of following methods:

- [x500 CM3 restore with pendrive manual](#)
- [Processor board programming using x500 device](#)

Just replace clean system image from Techbase with file created accordingly to this manual.

2. What do we need?

- PC with **Ubuntu Linux** operating system
- x500/x500 CM3 device

3. Preparing the environment

3.1 Instalation of required software on your PC

Upload **rpiboot** application and unpack it on the device (for example to */root* directory).
Versions for Ubuntu:

Version	Link
32 bit	rpiboot_ubuntu_x32.tar.gz
64 bit	rpiboot_ubuntu_x64.tar.gz

If you have other Linux, you can download the source code and compile it. More informations are [here](#) in *BUILDING RPIBOOT ON YOUR HOST SYSTEM (CYGWIN/LINUX)* section.

3.2 Connecting to a PC

After the installation of required software, you must connect **x500 device** to your PC. Get some miniUSB cable and connect it to miniUSB port in the x500 device. Other end plug to free USB port of a

PC. Then you will power the device (refer next paragraph).

4. Mounting device's memory

1. Run **rpiboot** program:

```
sudo ./rpiboot
```

2. Plug power cord to the x500 device

After that steps similar view should appear:

```
root@jakub-ThinkPad-T410:/root# sudo ./rpiboot
Waiting for BCM2835/6/7
Sending bootcode.bin
Successful read 4 bytes
Waiting for BCM2835/6/7
Second stage boot server
File read: start.elf
Second stage boot server done
```

If there is only *Waiting for BCM2835/6/7* message after connecting device, you can try to replace order of 1. and 2. steps: first connect device and then run *rpiboot* program.

3. Locate compute module memory: do it with *lsblk* command. Compute module has about 4GB (3.7G) storage size. There is an example of *lsblk* command's output:

```
NAME      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda        8:0    0 111.8G  0 disk
├─sda1     8:1    0 102.5G  0 part /
├─sda2     8:2    0    1K    0 part
└─sda5     8:5    0   9.4G  0 part [SWAP]
sdb        8:96    1   3.7G  0 disk
├─sdb1     8:97    1   56M   0 part
├─sdb2     8:98    1  800M   0 part
├─sdb3     8:99    1   34M   0 part
└─sdb4     8:100   1   2.8G   0 part
sr0       11:0    1 1024M  0 rom
```

4. Umount all partitions from compute module, if they have been mounted by system:

```
sudo umount MOUNTPOINT
```

MOUNTPOINT is a directory path, which can be found in *MOUNTPOINT* column in *lsblk* command's output.

5. Creating image of device's system

1. Use disk copy tool:

```
sudo dd bs=4MiB if=/dev/sdX of=backup_image
```

sdX is a device's name, which can be found in *lsblk* command's output.

Copy progress can be checked by following command executed in another terminal:

```
sudo killall -USR1 dd
```

After the processor is programmed correctly, the screen should display similar lines:

```
root@jakub-ThinkPad-T410:/home/jakub# sudo dd bs=4MiB if=/dev/sdb  
of=/home/jakub/images/firmware1508261108.img  
932+0 records in  
932+0 records out  
3909091328 bytes (3,9 GB) copied, 652,743 s, 6,0 MB/s
```

6. Finishing work

Unplug device from power and from PC.

Backup using additional system from pendrive

This method requires booting device from pendrive with special system.

Step by step instructions

- Prepare pendrive with at least 8GB of capacity
- Connect pendrive to computer
- Get pendrive dev path. You can use for example lsblk utility for this

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
sda	8:0	0	111,8G	0	disk	
└sda1	8:1	0	102,5G	0	part	/
└sda2	8:2	0	1K	0	part	
└sda5	8:5	0	9,4G	0	part	[SWAP]
sdb	8:128	1	14,5G	0	disk	
└sdb1	8:129	1	14,5G	0	part	

In this example our pendrive has path: `/dev/sdb`

You can recognize your pendrive with its size or by running this command twice: before plugging and after.

- Download **pendrive image** from [here](#)
- Unpack zip downloaded zip

Zip file contains image which has 5GB size. If you couldn't unpack it with your zip unpacker, try doing that with 7-zip

- Unmount all partitions from pendrive if necessary
- Flash pendrive with dd command using unpacked image

```
dd if=image.img of=/dev/<pendrive_name> bs=4MiB
```

- Connect monitor, keyboard and prepared pendrive to the x500 device
- Run the device
- Press any key when message "Hit any key to stop autoboot: " will appear on the screen
- Execute following commands:

```
setenv boot_targets usb0 mmc0 pxe dhcp
saveenv
reset
```

These commands set uboot to boot from pendrive first instead of internal mmc memory

- Wait for rescue system to boot
- Log in to the system:

```
login: root  
password: techbase
```

- Backup system with command:

```
backup_system --auto-mode
```

This command will copy main system's partitions from mmc memory to third partition of pendrive
If you want more options execute:

```
backup_system --help
```

- After copying, shutdown system with command:

```
halt
```

- Unplug pendrive and press reset button or unplug and plug it again

Troubleshooting

System doesn't start from pendrive

Synoptics

- Booting stops on uboot and with trying to read pendrive
- U-Boot displays 0 pendrive storages during start procedure

Solution

- Use another pendrive

Backup main system using built-in rescue system

To backup main system perform following steps:

1. Enter backup mode. (see: [X500 Rescue Mechanisms](#))

Not all x500 CM3 devices have built-in rescue system. If you aren't able to boot rescue system, check other backup methods [here](#).

2. Acquire root access:

```
sudo su
```

3. Connect external storage to USB port

External storage should be formatted in FAT32 format and have a minimum of 4GB of free space.

4. If your device have a **core package** version **>=1710021554** (you can check this using **getenv | grep CORE**) skip to step 5. If version is less, you can update your core package:

Platform	Command
x500CM3	softmgr update core -b x500cm3-test
9500CM3	softmgr update core -b 9500cm3-test

and then skip to step 5 or go ahead and perform next steps.

- 4.1. Download backup_main_system script and copy into device. Script can be found [here](#).

- 4.2. Unzip file and upload script to the device via FTP.

```
protocol: sftp
port: 22
login: root
pass: techbase
```

- 4.3. Give execution privileges to the script:

```
chmod +x backup_main_system
```

5. Run script

Script can backup both (boot and system) partitions of the main system.

If script was uploaded by FTP:

- go to directory with it
- replace *backup_main_system* with *./backup_main_system* in next steps

Usage:

```
backup_main_system --auto-mode
```

This command will mount pendrive's first partition, copy main system's partitions from mmc memory to pendrive. If you want more options execute:

```
backup_system --help
```

6. Shutdown system after copying

```
halt
```

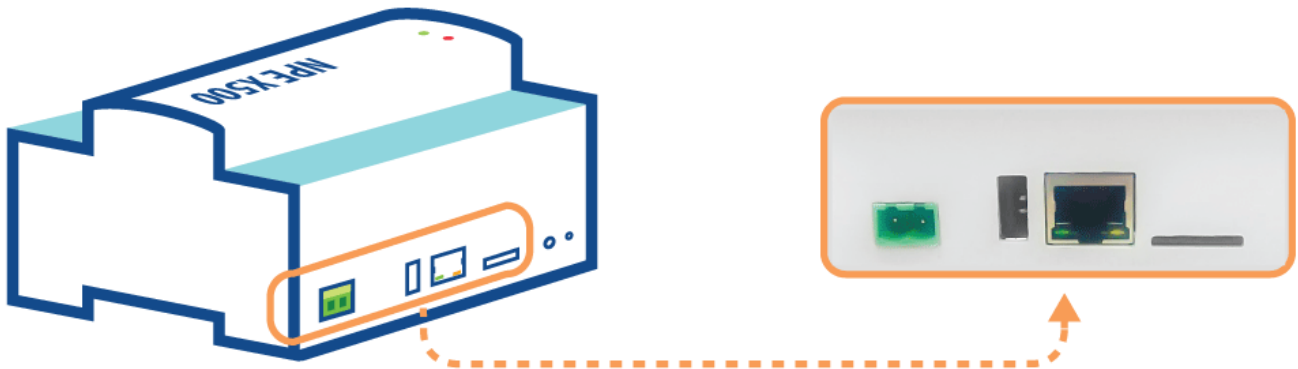
Communication via the Ethernet port

1. Introduction

This document describes how to connect the NPE X500 to the network.

2. Connection

X500 industrial computer has network interface marked on the casing as a **eth0**. Connect the network cable according to the following steps:



There are 2 ways to connect with device using ETH interface

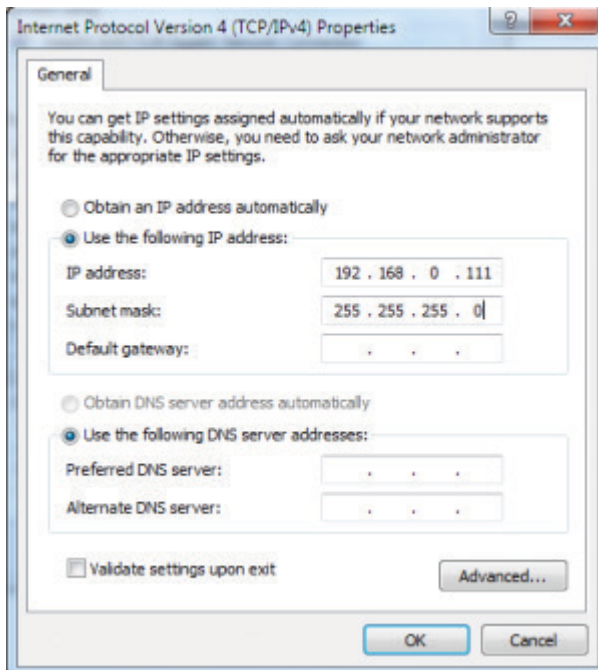
- directly connected to the PC
- connected to the LAN network

2.1 Device directly connected to the PC

You can connect your device directly to the PC, but first you need to setup static IP address on your PC in network settings. The IP address you set on PC must be in subnet "0", eg.

IP Address: 192.168.0.111

Subnet mask: 255.255.255.0



Connection parameters:

Default IP address: 192.168.0.101

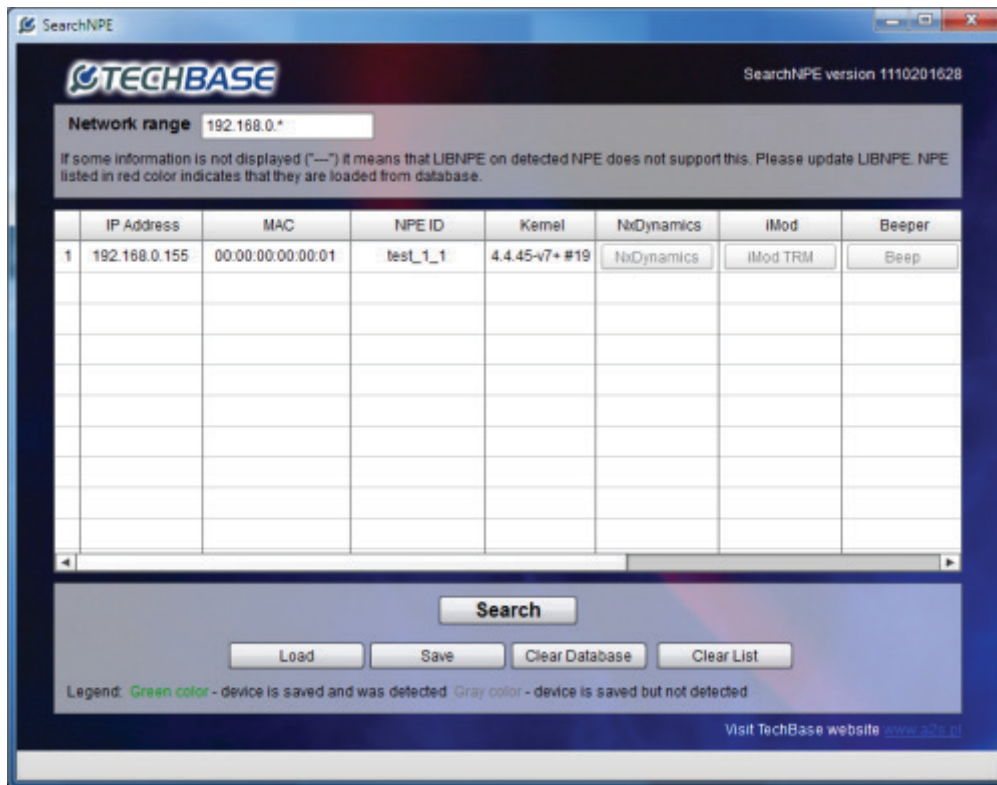
Port: 22

Login: root

Password: techbase

2.2 Device connected to LAN network

You can connect gateway device to the same LAN network as your PC and device will get IP Address from DHCP server. You can find the device in the network using [SearchNPE](#). Please use SSH terminal to connect to the device e.g. using PuTTY in Windows or `ssh` command in Linux.



If you know IP address of the device, you can connect by SSH terminal, eg Putty. You can find PuTTY at following address: <http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe>

Port: 22

Login: root

Password: techbase

3. Change network settings

3.1 Setting fixed IP

By default, the device is set to a dynamic IP address but if DHCP server is not available then device will setup static IP address which is in the default 192.168.0.101/24

To change this settings, you need to edit the file in the location: /etc/dhcpd.conf
e.g. put on the console

```
nano /etc/dhcpd.conf
```

Please go to the end of the file and change below code

```
#interface eth0
static ip_address=192.168.0.1/24
static routers=192.168.0.99
static domain_name_servers=8.8.8.8
```

WiFi connection

1. Introduction

This document describes how to run the Wifi connection modem.

2. Scan network

To scan for find networks, please type command:

```
iwlist wlan0 scan
```

3. Configuration and run

Before connection you need to setup two configuration files:

- **/etc/network/interfaces**
- **/etc/wpa_supplicant/wpa_supplicant.conf**

```
sudo nano /etc/network/interfaces
```

At the bottom of the file please add following lines

```
allow-hotplug wlan0
iface wlan0 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Then please type command:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Please add following lines to the file

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=PL

network={
ssid="Name_of_network"
psk="WiFi_PASS"
key_mgmt=WPA-PSK
}
```

Device should automatically connect to WiFi network after reboot.

```
reboot
```

4. Check connection configuration

You can check connection configuration (such as IP Address or MAC Address) using:

```
ifconfig
```


Trouble shooting

1. Connection doesn't work

Please check that the interface is present.

```
ifconfig wlan0
```

If you get some errors please try to enable interface:

```
ifconfig wlan0 up
```

(for disable interface you can do *ifconfig wlan0 down*)

2. Interface isn't present or can't enable it

Please contact us via <http://support.techbase.eu>, and attach result of following command:

```
lsusb
```

WMBUS module

1. Introduction

This document describes how you can check and manage the WMBUS module

2. Hardware version

ID	Name	Supplier	Description on the label	Documentation
1	EMB-WMB868	EMbit	WMBUS	Lora/Wmbus Embit
2	EMB-WMB169PA	EMbit	WMBUS169	Lora/Wmbus Embit
3	RC1180-MBUS3	Radiocraft	WMBUS	WMBUS RC11XX-MBUS3
4	RC1190-MBUS3	Radiocraft	WMBUS	WMBUS RC11XX-MBUS3

GPRS connection

1. Introduction

This document describes how to manage the modem communication on platforms [NPE-X500](#), [NPE-9500](#), [NPE-G1](#).

2. Available tools and supported modems

- **gprs** script - is our own script that manages the modem connection e.g. check status, enable autoreconnect, it can show the connection status using LED.
 - Supported modems:
 - Huawei MU-609
 - Huawei MU-709
 - ZTE ME3630-U1A
- **WvDial** - is a utility that helps in making modem-based connections to the Internet that is included in some Linux distributions.
 - Modems using in our devices:
 - [Huawei ME909U-521](#)
 - [SIM-7500JE](#)
 - [ZTE ME3630-U1A](#)

3. gprs script

3.1 Connection setup

File **/home/core/syscfg** includes GPRS connection configuration. To change connection parameters edit below lines in **/home/core/syscfg** file, the most important are:

- GPRS_PIN,
- GPRS_APN_NAME,
- GPRS_RECONNECT,
- GPRS_AUTOSTART,
- GPRS_LOGIN,
- GPRS_PASSWORD.

3.2 Connection

The script maintaining connection checks every minute if it is connected to the network.

In order to establish GPRS connection use **gprs connect** command. Remember first to edit properly the configuration file (type login and password, if it's necessary type PIN code and check if APN is correct).

```

root@raspberrypi:/home/pi# gprs connect
NPE GPRS Connection Utility [Version 1508270006] - command: connect
[MUX_INIT] /dev/ttyUSB0 exists
[REGISTERING] PIN number not required
[REGISTERING] Waiting for network registration. Please wait...
[CONNECTING] Starting PPP connection. Please wait...
[CONNECTING] Call center not specified. Using: *99***1#
[CONNECTING] APN not specified. Using default empty APN:
[CONNECTING] Setting GPRS login: ppp
[CONNECTING] Setting GPRS password: ppp
[CONNECTING] Connecting...
[CONNECTING] Connection script succeeded...
[CONNECTING] PPP interface established...
[CONNECTED] Successfully connected

```

During the connection **USER_LED** signicator flashes at a high frequency with green light. After establishing the connection LED is lit continuously.

3.3 Connection management

3.3.1. Checking status

To check connection status type **gprs status** command

```

root@raspberrypi:/home/pi# gprs
NPE GPRS Connection Utility [Version 1508270006] - command: status
Usage: gprs {connect|abort|status|operation|disconnect|restart|register|unregister|ready|reconnect[on|off|monitor]|version}
STATUS: CONNECTED
time: 0h:16m:13s outgoing:38830B incoming:34790B
PPP IP: 95.40.38.34
CONNECTION TYPE: NONE
RECONNECT: OFF
MODEM_INFO=MU609_new
root@raspberrypi:/home/pi# █

```

After running **gprs connect** command the **ppp0** interface will be created. It is visible after running the **ifconfig** command.

```

ppp0    Link encap:Point-to-Point Protocol
        inet addr:95.40.38.34 P-t-P:10.64.64.64 Mask:255.255.255.255
        UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
        RX packets:285 errors:0 dropped:0 overruns:0 frame:0
        TX packets:349 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:3
        RX bytes:30014 (29.3 KiB) TX bytes:33525 (32.7 KiB)

```

3.3.2 auto-reconnect

You can enable **auto-reconnect** function which allows to reconnect after losing the connection.

Auto-reconnect option is disabled by default! In order to enable it type **gprs reconnect on** command.

Reconnect ON

Reconnect OFF

After running **gprs status** command:

```
root@raspberrypi:/home/pi# gprs reconnect on
NPE GPRS Connection Utility [Version 150827000]
root@raspberrypi:/home/pi# gprs status
NPE GPRS Connection Utility [Version 150827000]
Checking the reconnect servers.
Ping primary address: (208.67.222.222) Succeeded
STATUS: CONNECTED
time: 0h:23m:24s outgoing:51845B incoming:45B
PPP IP: 5.174.161.54
CONNECTION TYPE: NONE
RECONNECT: ON
MODEM_INFO=MU609_new
root@raspberrypi:/home/pi#
```

```
root@raspberrypi:/home/pi# gprs reconnect off
NPE GPRS Connection Utility [Version 150827000]
root@raspberrypi:/home/pi# gprs status
NPE GPRS Connection Utility [Version 150827000]
Checking the reconnect servers.
Ping primary address: (208.67.222.222) Succeeded
STATUS: CONNECTED
time: 0h:26m:31s outgoing:57360B incoming:5024B
PPP IP: 5.174.161.54
CONNECTION TYPE: NONE
RECONNECT: OFF
MODEM_INFO=MU609_new
root@raspberrypi:/home/pi#
```

File
/home/core/syscfg
:

```
GPRS_PIN=
GPRS_REG_RETRY=10
GPRS_MODE=PPP
GSM_MODE=AUTO
GPRS_MUX=Y
GPRS_AUTOSTART=Y
DEFAULT_ROUTE=GPRS
GPRS_APN_NAME=
GPRS_RECONNECT=Y
GPRS_PING_IP_1=208.67.222.222
GPRS_PING_IP_2=208.67.220.220
GPRS_AUTO_DNS=Y
GPRS_DNS_1=8.8.8.8
GPRS_DNS_2=208.67.222.222
GPRS_DNS_3=8.8.4.4
GPRS_LOGIN=ppp
GPRS_PASSWORD=ppp
GPRS_DYNDNS=N
GATEWAY_ETH=192.168.0.99
GSM_BAUD=230400
GSM_PORT=/dev/ttyUSB0
root@raspberrypi:/home/pi#
```

```
GPRS_PIN=
GPRS_REG_RETRY=10
GPRS_MODE=PPP
GSM_MODE=AUTO
GPRS_MUX=Y
GPRS_AUTOSTART=Y
DEFAULT_ROUTE=GPRS
GPRS_APN_NAME=
GPRS_RECONNECT=N
GPRS_PING_IP_1=208.67.222.222
GPRS_PING_IP_2=208.67.220.220
GPRS_AUTO_DNS=Y
GPRS_DNS_1=8.8.8.8
GPRS_DNS_2=208.67.222.222
GPRS_DNS_3=8.8.4.4
GPRS_LOGIN=ppp
GPRS_PASSWORD=ppp
GPRS_DYNDNS=N
GATEWAY_ETH=192.168.0.99
GSM_BAUD=230400
GSM_PORT=/dev/ttyUSB0
root@raspberrypi:/home/pi#
```

Do not attempt to run **gprs connect** command if the reconnect option is enabled.

3.3.3. Disconnecting

There are 2 ways to stop the connection.

1. **gprs abort** - connection will be stopped and auto-reconnect function won't start the connection back
2. **gprs disconnect** - connection will be stopped, but if the auto-reconnect option is enabled, the connection will be resumed.

```
root@raspberrypi:/home/pi# gprs abort
NPE GPRS Connection Utility [Version 1508270006] - command: abort
[ABORTING] Aborting
[ABORTING] Killing gprs script
```

1.

```
root@raspberrypi:/mnt/mtd/iMod# gprs disconnect
NPE GPRS Connection Utility [Version 1508270006] - command: disconnect
Disconnected
STATUS: DISCONNECTED
CONNECTION TYPE: NONE
RECONNECT: ON
MODEM_INFO=MU609_new
```

2.

3.4. Logs

Log files are located in 2 paths:

```
/mnt/.hidtmp/gprs.log - dedicated log file of gprs application
/var/log/messages - general log file
```

Log files are usually extensive, in order to view them use commands below:

Viewing last 50 lines of log:

```
tail -n 2500 /var/log/messages | grep chat | head -n 50 - show logs related
do communication between modem and APN-em (chat)

tail -n 2500 /var/log/messages | grep gprs | head -n 50 - show logs related
do **gprs** application

tail -n 50 /mnt/.hidtmp/gprs.log
```

4. WvDial

WvDial should be preinstalled on the device. If you don't have wvdial you can install by the command:

```
apt-get install wvdial
```

4.1 Connection setup

The configuration you can find in the file **/etc/wvdial.conf**, to edit please do:

```
nano /etc/wvdial.conf
```

4.1.1. Typical configuration

This configuration should be working with most modems.

```
[Dialer play]
Modem = /dev/ttyUSB0
Baud = 230400
Init1 = ATH
Init2 = ATE1
Init3 = AT+CGDCONT=1,"IP","internet"
Dial Command = ATD
```

```
Phone = *99#  
Stupid mode = yes  
Username = "blank"  
Password = "blank"
```

4.2 Start connection

For start connection you need to run

```
wvdial "configuration name" &  
e.g. wvdial play &
```

5. Troubleshooting

Huawei MU-609/Huawei MU-709/ZTE ME3630-U1A

If modem doesn't work properly follow these steps:

- check if modem logs correctly on USB bus - use **lsusb** command.
- next run **modem_info** command, in order to reset **USB** bus - it checks if modem is connected correctly, sets proper **baudrate** and **port** where modem is located.

Correct result after the **lsusb** command:

```
root@raspberrypi:/home/pi# lsusb  
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.  
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub  
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.  
Bus 001 Device 013: ID 12d1:1573 Huawei Technologies Co., Ltd.  
root@raspberrypi:/home/pi#
```

Correct result after the **modem_info** command:

```
root@raspberrypi:/home/pi# modem_info  
Please wait - resetting the PCI bus  
Checking 3G modem  
DETECTED MODEM: MU609_new  
Setting baudrate to: 230400  
Baudrate stored in syscfg and env variable  
Setting port to: /dev/ttyUSB0  
Port stored in syscfg  
root@raspberrypi:/home/pi#
```

Reset modems, handled by wvdial

If modem doesn't work properly (for example respond *ERROR* to each AT command) follow these steps:

- check if modem logs correctly on USB bus - use **lsusb** command.
- next reset it (use soft reset first, then hard reset)

Commands for Soft reset

These commands makes modem to perform its internal software reset:

```
echo 30 > /sys/class/gpio/export
```

```
echo out > /sys/class/gpio/gpio30/direction
echo 1 > /sys/class/gpio/gpio30/value
sleep 1
echo 0 > /sys/class/gpio/gpio30/value
sleep 1
echo in > /sys/class/gpio/gpio30/direction
```

Commands for Hard reset

These commands turns off modem and then connects power to it once again:

```
echo 31 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio31/direction
echo 0 > /sys/class/gpio/gpio31/value
sleep 15
echo 1 > /sys/class/gpio/gpio31/value
sleep 10
```


ZTE ME3630-U1A

1. Introduction

This document describes how to run the ME3630 modem which wvdial utility. It is also supported by GPRS script (refer [this](#) document).

2. Configuration

Firstly please check that your modem create the /dev/tty* directory properly, you should receive

```
ttyUSB0  
ttyUSB1  
ttyUSB2
```

If you didn't see this please run

```
modprobe option  
echo 0x19d2 0x1476 > /sys/bus/usb-serial/drivers/option1/new_id
```

After this the drive should start and you should see the ports in the /dev/ like we wrote above.

If you have gprs package 1901181015 or newer, you should have following pathes:

```
/dev/ME3630-00  
/dev/ME3630-01  
/dev/ME3630-02
```

They are pathes to appropriate /dev/ttyUSBx files. Use /dev/ME3630-02 instead of /dev/ttyUSB2.

3. Start the modem

Now you can run wvdial application and check the connection, if you don't have wvdial you can install by the command:

```
apt-get install wvdial -y
```

The configuration you can find in the

```
/etc/wvdial.conf
```

For start application you need to run

```
wvdial "configuration name" &  
e.g. wvdial play &
```

4. Typical configuration

```
[Dialer play]  
Modem = /dev/ttyUSB2  
Baud = 230400  
Init1 = ATH  
Init2 = ATE1  
Init3 = AT+CGDCONT=1,"IP","internet"  
Dial Command = ATD  
Phone = *99#  
Stupid mode = yes  
Username = "blank"  
Password = "blank"
```

5. Troubleshooting

If modem doesn't work properly (for example respond *ERROR* to each AT command) follow these steps:

- check if modem logs correctly on USB bus - use **lsusb** command.
- next reset it (use soft reset first, then hard reset)

Commands for Soft reset

These commands makes modem to perform its internal software reset:

```
echo 30 > /sys/class/gpio/export  
echo out > /sys/class/gpio/gpio30/direction  
echo 1 > /sys/class/gpio/gpio30/value  
sleep 1  
echo 0 > /sys/class/gpio/gpio30/value  
sleep 1  
echo in > /sys/class/gpio/gpio30/direction
```

Commands for Hard reset

These commands turns off modem and then connects power to it once again:

```
echo 31 > /sys/class/gpio/export  
echo out > /sys/class/gpio/gpio31/direction  
echo 0 > /sys/class/gpio/gpio31/value  
sleep 15  
echo 1 > /sys/class/gpio/gpio31/value  
sleep 10
```

[SIM-7500JE](#)

1. Introduction

This document describes how to run the 7500JE modem which is not supported by GPRS script

2. Checking if the modem is working

Firstly please check that your modem create the /dev/tty* directory properly, you should receive

```
ttyUSB0
ttyUSB1
ttyUSB2
ttyUSB3
ttyUSB4
```

3. Start the modem

Now you can run wvdial application and check the connection, if you don't have wvdial you can install by the command:

```
apt-get install wvdial
```

The configuration you can find in the

```
/etc/wvdial.conf
```

For start connection you need to run

```
wvdial "configuration name" &
e.g. wvdial play &
```

4. Typical configuration

```
[Dialer play]
Modem = /dev/ttyUSB2
Baud = 230400
Init1 = ATH
Init2 = ATE1
Init3 = AT+CGDCONT=1,"IP","internet"
Dial Command = ATD
Phone = *99#
Stupid mode = yes
Username = "blank"
Password = "blank"
```

5. Troubleshooting

If modem doesn't work properly (for example respond *ERROR* to each AT command) follow these

steps:

- check if modem logs correctly on USB bus - use **lsusb** command.
- next reset it (use soft reset first, then hard reset)

Commands for Soft reset

These commands makes modem to perform its internal software reset:

```
echo 30 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio30/direction
echo 1 > /sys/class/gpio/gpio30/value
sleep 1
echo 0 > /sys/class/gpio/gpio30/value
sleep 1
echo in > /sys/class/gpio/gpio30/direction
```

Commands for Hard reset

These commands turns off modem and then connects power to it once again:

```
echo 31 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio31/direction
echo 0 > /sys/class/gpio/gpio31/value
sleep 15
echo 1 > /sys/class/gpio/gpio31/value
sleep 10
```

SIM-7000 GPS

1. Introduction

This document describes how to run the SIM-7000 modem and get position from GPS system.

2. Configuration

Firstly please check that your modem create the /dev/tty* directory properly, you should receive:

```
ttyUSB0  
ttyUSB1  
ttyUSB2  
ttyUSB3  
ttyUSB4
```

3. Quick start

Following commands will configure GPS modem to report localization in NMEA format.

```
AT+CGNSPWR=1  
AT+CGNSINF
```

Localization data will appear after each execution of *AT+CGNSINF* command.

4. Advanced

4.1. AT commands

There is list and description of AT commands supported by this GPS modem.

Choose localization data output

```
AT+CGNSCFG=<mode>
```

where mode is:

- 0 - turn off sending localization data
- 1 - turn on sending localization data

Choose localization provider

Execute it before *AT+CGNSPWR*.

```
AT+CGNSMOD=<gps_mode,glo_mode,bd_mode,gal_mode>
```

where:

- *gps_mode* - enable/disable GPS data receiving
- *gl_mode* - enable/disable GLONASS data receiving
- *bd_mode* - enable/disable BEIDOU data receiving

- gal_mode - enable/disable GALLIEO data receiving

Replace each *_mode with:

- 0 for disable
- 1 for enable

Power up/down GPS localization services

```
AT+CGNSPWR=<mode>
```

where mode is:

- 0 - turn off
- 1 - turn on

Get localization data (NMEA format)

This command gets current localization data in NMEA format

```
AT+CGNSINF
```

Enable autoreport of localization data

This command enables/disables autoreporting.

If enabled, you don't need to send *AT+CGNSINF* command to get data.

```
AT+CGNSURC=<mode>
```

where mode is:

- 0 - turn off autoreporting
- 1 - turn on autoreporting on every GNSS FIX
- 2 - turn on autoreporting on every 2 GNSS FIX
- ...
- 255 - turn on autoreporting on every 255 GNSS FIX

Processor board programming using Raspberry Development Kit

1. Introduction

NPE X500/9500 device include Raspberry Compute Module processor board. It is the hearth of NPE X500 platform. This manual describes how to program the Raspberry Compute Module step by step.

2. What do we need?

- PC with **Linux** operating system
- Raspberry Development Kit
- Processor board (Raspberry Compute Module)

3. Preparing the environment

3.1 Instalation of required software on your PC

To upload image to processor, install the module memory on PC. Use application **rpiboot**, which can be cloned together with a package from:

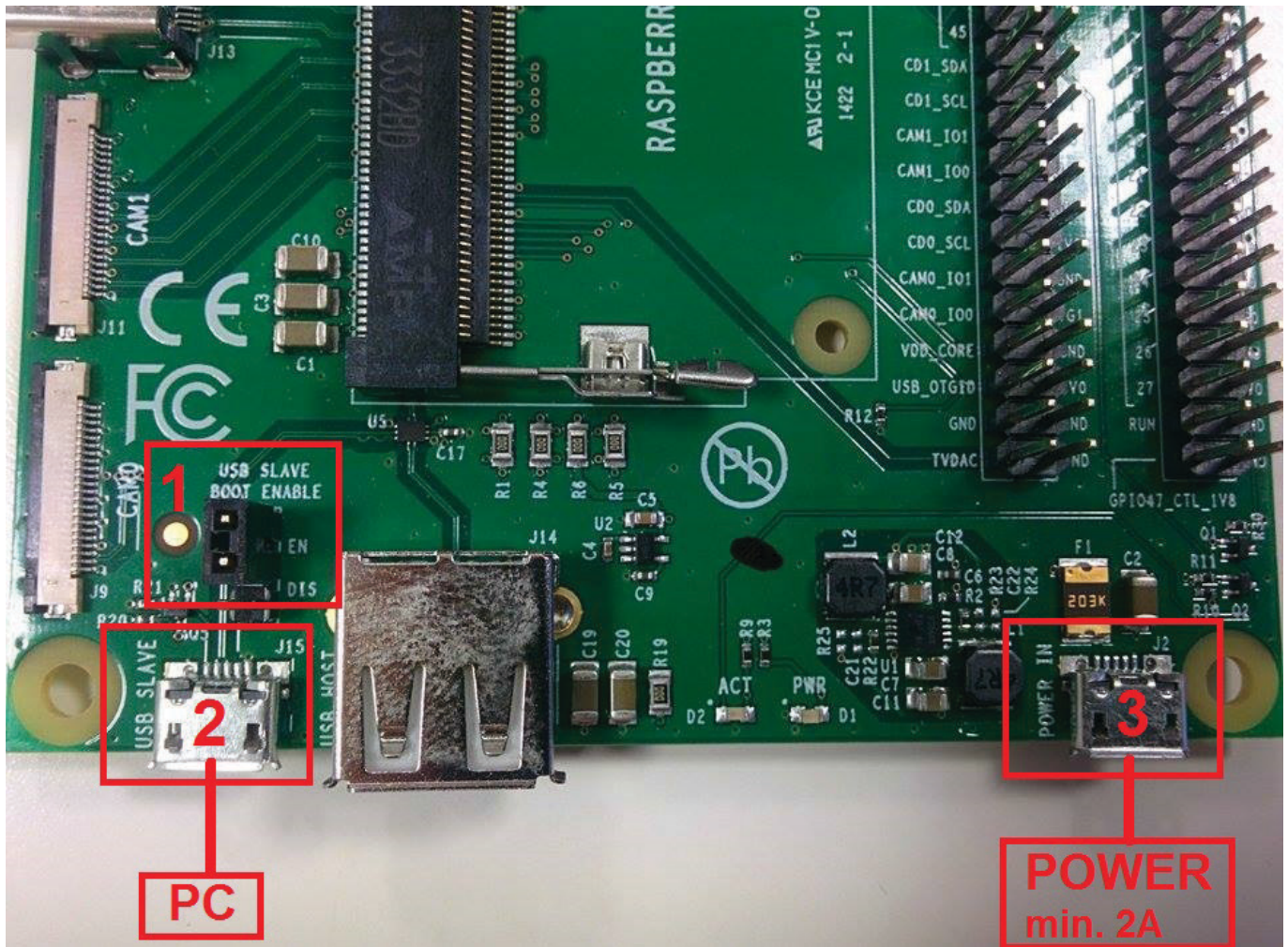
```
sudo git clone --depth=1 https://github.com/raspberrypi/tools
```

and installed with:

```
cd tools/usbboot  
sudo apt-get install libusb-1.0-0-dev  
make  
sudo make install
```

3.2 Connecting to a PC

After the installation of required software, you must connect **Development Kit** to your PC. First step is to setup **J4(1) jumper** to **USB SLAVE** position, connect DevKit **USB SLAVE(2)** via USB cable to the PC USB port, next connect power to **USB POWER IN(3)** (it has to be external power supply **2A** minimum).



4. Downloading system image

System image can be downloaded from [here](#)

5. System image upload

1. Search for file **rpiboot**:

```
find / -name rpiboot
```

2. Go to directory where the **rpiboot** file is and run it:

```
sudo ./rpiboot
```

After running similar view should appear:

```
root@jakub-ThinkPad-T410:/home/jakub# sudo /usr/bin/rpiboot
Waiting for BCM2835 ...
Found serial = 0: writing file /usr/share/rpiboot/usbbootcode.bin
```


Waiting for BCM2835 ...

3. Locate Development Kit:

```
df -h
```

Result of **df -h** command before connecting DevKit:

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sda6	55G	37G	16G	71%	/
none	4,0K	0	4,0K	0%	/sys/fs/cgroup
udev	1,9G	4,0K	1,9G	1%	/dev
tmpfs	384M	1,4M	382M	1%	/run
none	5,0M	4,0K	5,0M	1%	/run/lock
none	1,9G	80K	1,9G	1%	/run/shm
none	100M	28K	100M	1%	/run/user

Result of **df -h** command after connecting DevKit:

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sda6	55G	37G	16G	71%	/
none	4,0K	0	4,0K	0%	/sys/fs/cgroup
udev	1,9G	4,0K	1,9G	1%	/dev
tmpfs	384M	1,4M	382M	1%	/run
none	5,0M	0	5,0M	0%	/run/lock
none	1,9G	80K	1,9G	1%	/run/shm
none	100M	28K	100M	1%	/run/user
/dev/sdb3	34M	3,9M	30M	12%	/media/jakub/BE40-C9C3
/dev/sdb1	56M	24M	33M	42%	/media/jakub/boot
/dev/sdb4	2,7G	1,5G	1,1G	57%	/media/jakub/0c8cb719-f655-4c47-bcd3-bec60ead5d28
/dev/sdb2	722M	473M	194M	71%	/media/jakub/13d368bf-6dbf-4751-8ba1-88bed06bef77

As you can see in comparison of both results of **df -h** command, DevKit is located at **/dev/sdb**, has ~3.5GB file size and is divided into **4** partitions.

Alternative method is to locate DevKit with **lsblk** command.

4. Now use the processor programming command:

```
sudo dd bs=4MiB if=source_to_image of=/dev/device_name
```

After the processor is programmed correctly, the screen should display similar lines:

```
root@jakub-ThinkPad-T410:/home/jakub# sudo dd bs=4MiB
if=/home/jakub/images/firmware1508261108.img of=/dev/sdb
```

```
932+0 records in
932+0 records out
3909091328 bytes (3,9 GB) copied, 652,743 s, 6,0 MB/s
```

Processor board programming using x500 device

1. Introduction

NPE X500/9500 device includes Raspberry Compute Module processor board. It is the heart of NPE X500 platform. This manual describes how to program this module using x500 device.

2. What do we need?

- PC with **Ubuntu Linux** operating system
- x500 device with Raspberry Compute Module in it

3. Preparing the environment

3.1 Instalation of required software on your PC

Upload **rpiboot** application and unpack it on the device (for example to */root* directory).
Versions for Ubuntu:

Version	Link
32 bit	rpiboot_ubuntu_x32.tar.gz
64 bit	rpiboot_ubuntu_x64.tar.gz

If you have other Linux, you can download the source code and compile it. More informations are [here](#) in *BUILDING RPIBOOT ON YOUR HOST SYSTEM (CYGWIN/LINUX)* section.

3.2 Connecting to a PC

After the installation of required software, you must connect **x500 device** to your PC. Get some miniUSB cable and connect it to miniUSB port in the x500 device. Other end plug to free USB port of a PC. Then you will power the device (refer 5. paragraph).

4. Downloading system image

System image can be downloaded from [there](#)

You can also restore system with image created according to this document: [Backup using mini USB port](#).

5. System image upload

1. Run **rpiboot** program:

```
sudo ./rpiboot
```

2. Plug power cord to the x500 device

After that steps similar view should appear:

```
root@jakub-ThinkPad-T410:/root# sudo ./rpiboot
Waiting for BCM2835/6/7
Sending bootcode.bin
Successful read 4 bytes
Waiting for BCM2835/6/7
Second stage boot server
File read: start.elf
Second stage boot server done
```

If there is only *Waiting for BCM2835/6/7* message after connecting device, you can try to replace order of 1. and 2. steps: first connect device and then run *rpiboot* program.

3. Locate compute module memory: do it with *lsblk* command. Compute module has about 4GB (3.7G) storage size. There is an example of *lsblk* command's output:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
sda	8:0	0	111.8G	0	disk	
├─sda1	8:1	0	102.5G	0	part	/
├─sda2	8:2	0	1K	0	part	
└─sda5	8:5	0	9.4G	0	part	[SWAP]
sdb	8:96	1	3.7G	0	disk	
├─sdb1	8:97	1	56M	0	part	
├─sdb2	8:98	1	800M	0	part	
├─sdb3	8:99	1	34M	0	part	
└─sdb4	8:100	1	2.8G	0	part	
sr0	11:0	1	1024M	0	rom	

4. Umount all partitions from compute module, if they have been mounted by system:

```
sudo umount MOUNTPOINT
```

MOUNTPOINT is a directory path, which can be found in *MOUNTPOINT* column in *lsblk* command's output.

5. Now use the processor programming command:

```
sudo dd bs=4MiB if=source_to_image of=/dev/sdX
```

sdX is a device's name, which can be found in `lsblk` command's output.

Copy progress can be checked by following command executed in another terminal:

```
sudo killall -USR1 dd
```

After the processor is programmed correctly, the screen should display similar lines:

```
root@jakub-ThinkPad-T410:/home/jakub# sudo dd bs=4MiB  
if=/home/jakub/images/firmware1508261108.img of=/dev/sdb  
932+0 records in  
932+0 records out  
3909091328 bytes (3,9 GB) copied, 652,743 s, 6,0 MB/s
```

Software update

1. Introduction

This document describes how to update or install software on NPE X500/9500 platform and describes the operation of the application **Software Manager - softmgr** .

2. Softmgr

You can update your device using **Softmgr**. **Softmgr** is an application that allows you to install and update software on your NPE/iMod device. Below the result of **softmgr -help**:

```
[root@techbase /]# softmgr
Software Manager [Version 1503021356] - TechBase Update Utility.
All rights reserved.
Wrong arguments
Usage: /bin/softmgr [-f yes/all] [-s server_addres] [-b branch_name] [-s
Y/N] [update/list/check/checkall] package_name

Options:
    -b          selection of branch name
    -f          enforces the installation of package.
Modes:
    yes - enforce the installation of a package
    all - enforce the installation of a package and all
dependant packages
    -s          selection of alternative server

Arguments:
    update      updating package
    update all  updating available packages
    check       listing available updates
    checkall    listing available updates and packages available to
install
    list        listing available packages with versions

Example:
    softmgr -f yes update sms
```

3. Software update

Remember!

Restart is required after update!

3.1 Branch list

Device	Branch name
NPE-X500	x500-stable
	x500-beta
NPE-X500-M3	x500cm3-stable
	x500cm3-beta
NPE-9500	9500-stable
	9500-beta
NPE-9500-M3	9500cm3-stable
	9500cm3-beta
NPE-BUSTER	buster-stable
	buster-beta

3.2 Check update

For check available update for installed packages please type

```
softmgr check
```

```
root@techbase:~# softmgr check
Software Manager [Version 1704281521] - TechBase Update Utility.
All rights reserved.
Trying to connect 46.248.164.54:1234
Package Name    Status
No updates available
```

3.3 Check available packages

For check all available packages please type

```
softmgr list
```

```
root@techbase:~# softmgr list
Software Manager [Version 1704281521] - TechBase Update Utility.
All rights reserved.
Trying to connect 46.248.164.54:1234
1      core 1704281521
2      firmware 1704281330
3      gprs 1509010953
4      imod_tiger 1604140000
5      imodcloud 1704101536
6      npe_tuneup 1609281346
```

7 owserver 1607221049

3.4 Update packages

Before updating to **beta** version, please contact technical support via [TechBase Support Portal - TSP](#)

For example below is command for installation/update package gprs from branch x500cm3-stable

```
softmgr -b x500cm3-stable update gprs
```


Date and time settings

This document contains informations about date and time settings and synchronization on the x500/x500cm3 devices.

Most sections refer to both devices. Descriptions for specific device are marked.

Setting timezone

To set timezone execute following command and choose your location:

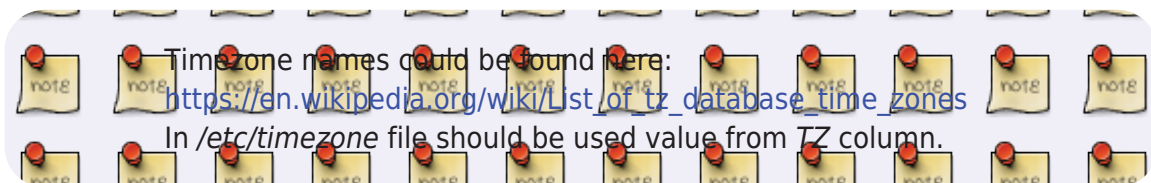
```
dpkg-reconfigure tzdata
```

You can also do it manually by setting timezone name in */etc/timezone* file and executing following command which will update system time.

```
dpkg-reconfigure -f noninteractive tzdata
```

For x500CM3 device with 4.9 system you need to execute following command instead of editing */etc/timezone* file:

```
timedatectl set-timezone <timezone>
```



After changing timezone, new time value should be written to the RTC clock:

- on x500: `hwclock -wu -f /dev/rtc0`
- on x500cm3: `hwclock -w -f /dev/rtc0`

NTP synchronization service

NTP service is responsible to synchronize device's date and time with Internet.

Check status

```
/etc/init.d/ntp status
```

Start service

```
/etc/init.d/ntp start
```

Stop service

```
/etc/init.d/ntp stop
```

Manual synchronization

To perform manual date and time synchronization from the Internet execute following commands. They stop NTP service, perform synchronization and start it again.

```
/etc/init.d/ntp stop  
ntpd -qg  
/etc/init.d/ntp start
```

After manual time synchronization, its value should be written to the RTC clock (refer next paragraph):

- on x500: `hwclock -wu -f /dev/rtc0`
- on x500cm3: `hwclock -w -f /dev/rtc0`

Configuration

NTP service can be configured in `/etc/ntp.conf` file.
After changing this file, restart of service will be needed:

```
/etc/init.d/ntp restart
```

Description of options from this file can be found on the Internet.
They are available for example [here](#).

RTC synchronization service

RTC synchronization service sets local time from built-in RTC clock and writes actual value to it.
RTC synchronization service performs following actions:

- setting the system time from the RTC clock - after device start
- setting the RTC clock from the current system time - every 10 minutes if there is connection to the NTP server

This service starts with system and works in background.
Log file is located in: `/var/log/rtc_sync.log`

Note that log is written only if `/var/log/rtc_sync.log` file is present in the filesystem.
To enable logging, create this file:

```
touch /var/log/rtc_sync.log
```

Check status

- on x500 `/etc/init.d/hwclock.sh status`
- on x500cm3 `systemctl status hwclock-sync`

If device has older core package (before 1801171539), status can be checked in following way:

```
ps aux | grep "rtc_sync" | grep -v grep
```

If there is line like following, means that service is working:

```
root      571  0.0  0.5   2752   2280 ?        S
01:14    0:00 /bin/bash /bin/rtc_sync
```

Start service

- on x500 /etc/init.d/hwclock.sh start
- on x500cm3 systemctl start hwclock-sync

Stop service

- on x500 /etc/init.d/hwclock.sh stop
- on x500cm3 systemctl stop hwclock-sync

Configuration

Configuration of the service is in /home/core/rtc/rtc_sync.conf file. It contains following options:

Option	Description	Default value
DELAY	Defines how often RTC clock will be set (if there is NTP server is available). Value is given in seconds	600
DEVICE	Defines RTC device	/dev/rtc0
LOCATION_LOGFILE	Defines path to log file	/var/log/rtc_sync.log
LOCATION_FILE	Defines location of directory with rtc_sync script. Don't change this option	/home/core/rtc

Read/set on demand

RTC clock can read or set on demand.

On x500 device RTC clock is handled with UTC time (-u argument).

In x500cm3 device RTC clock is handled with LOCAL time (no -u argument).

Don't use UTC time on x500cm3 and LOCAL time on x500 while reading/writing RTC clock. This will cause time settings problems.

Reading RTC clock value

```
hwclock -f /dev/rtc0
```

Setting the system time from RTC clock

- on x500 hwclock -su -f /dev/rtc0
- on x500cm3 hwclock -s -f /dev/rtc0

Setting the RTC clock from the current system time

- on x500 hwclock -wu -f /dev/rtc0
- on x500cm3 hwclock -w -f /dev/rtc0

Diagnostic script

[Here](#) is a diagnostic script for date and time functionality.

It checks statuses of services connected to date and time functionality and writes result to a file.

- Upload downloaded archive to the device
- Unpack it: unzip test_time_settings.sh.zip
- Execute it: ./test_time_settings.sh
- File with results will be written to aa file in the same directory as script.
- [Here](#) is an example of results from device where time and date services work properly

Hardware - X500

1. Introduction

NPE X500 is a series of industrial computers which you can easily adapt to your needs by choosing from the available options. This document includes description of connectors used in X500 platform and manuals.

2. Technical properties

Base version depending on compute module version:

- M3
 - Processor Cortex-A53 1.2MHz
 - 1GB RAM and 4GB NAND FLASH memory
- M38
 - Processor Cortex-A53 1.2MHz
 - 1GB RAM and 8GB NAND FLASH memory
- M316
 - Processor Cortex-A53 1.2MHz
 - 1GB RAM and 16GB NAND FLASH memory
- M332
 - Processor Cortex-A53 1.2MHz
 - 1GB RAM and 32GB NAND FLASH memory

Hardware Add-on Options:

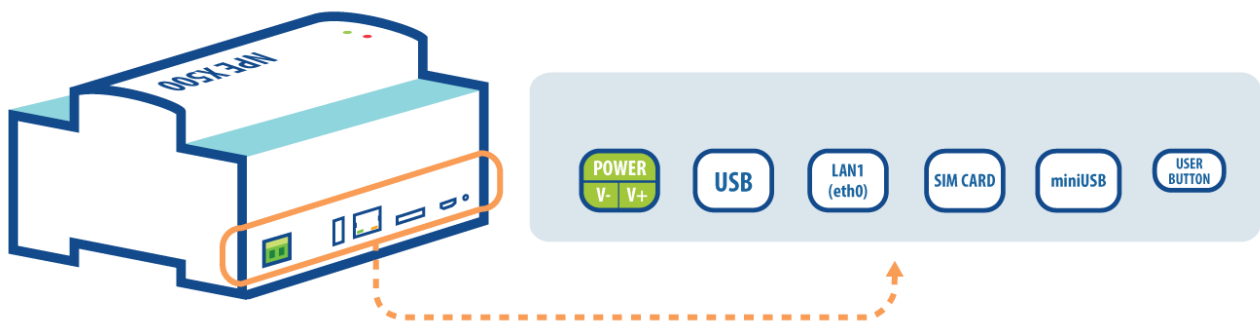
- 2x RS-232/485
- 4x Configurable Digital In/Out
- 4x Digital In
- 4x Digital Out
- 4x Analog In
- CAN bus
- HDMI

Optional internal expansion cards:

- LTE/3G/GPRS
- GPS Module
- WiFi
- Bluetooth
- ZigBee
- 4R
- 12DIO
- 4RS
- ETH
- 2ETH
- 8AI
- 4/8/12AO

3. Connector description

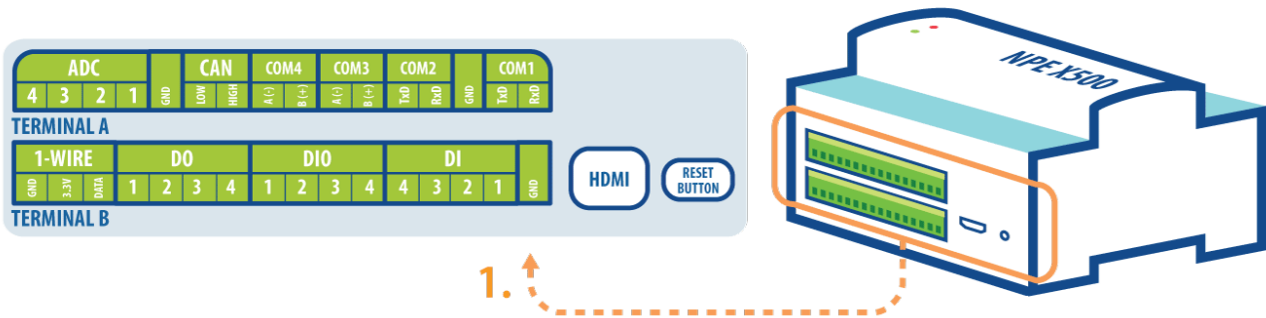
3.1 Side A



3.1.1 Detailed connector description

Name		Description
Power	V-	Ground
	V+	VDC 9-30VDC
USB		USB port
LAN1		Ethernet port
SIM CARD		SIM Card slot
minuUSB		miniUSB port for flashing
USER BUTON		Programmable user button

3.2 Side B



3.2.1 Detailed connector description

TERMINAL A			
Name		Description	
ADC1-4		0-10 VDC Analog Input	
GND		Ground	
CAN	High	CAN-High (CAN+)	
	Low	CAN-Low (CAN-)	

RS485_2	B(+)	RS-485 Serial port - Data + (COM2)
	A(+)	RS-485 Serial port - Data - (COM2)
RS485_1	B(+)	RS-485 Serial port - Data + (COM1)
	A(+)	RS-485 Serial port - Data - (COM1)
COM2	RxD	RS-232 Serial port - Received Data (COM2)
	TxD	RS-232 Serial port - Transmitted Data (COM2)
GND		Ground
COM1	RxD	RS-232 Serial port - Received Data (COM1)
	TxD	RS-232 Serial port - Transmitted Data (COM1)

TERMINAL B		
Name		Description
1-wire	DATA	1-wire data
	3.3V	1-wire power (+) 3.3V
	GND	1-wire Ground (-)
DO1-4		4x Digital Output Open Collector
DIO1-4		4x Digital Input/Output
DI1-4		4x Digital Input
GND		Ground

4. User manual

- [Inputs/outputs](#)
 - [GPIO LIB for C](#)
- [Serial ports](#)
 - [Service port](#)
- [CAN bus](#)
- [OneWire](#)
- [Mapping GPIO ports](#)
- [Analog inputs - read/setup](#)
- [Hardware Watchdog](#)

List of hardware components

1. Introduction

This document list all hardware components together with the configuration on the platforms NPE-X500 and NPE-9500. The most devices working via [I2C](#) and [SPI](#) intrface.

2. I2C devices

Device name	Chip name	Address	IRQ Line	Reset Line
RTC	DS1338Z-33+T	0x68	-	-
RS232/485(ttySC0)	SC16IS740	0x4d	GPIO45	GPIO26
ADC	MCP3424	0x6c	-	-
1-wire	DS2482S-100+	0x18	-	-

3. SPI devices

Device name	Chip name	Chip select	IRQ Line
CAN	MCP2515	CE0 - GPIO8	GPIO29

4. Ethernet and USB ports

We used one chip to have [hub](#) port [USB](#) and [Ethernet](#) port.

Device name	Chip name
Ethernet	LAN9514I-JZX
USB port	LAN9514I-JZX

5. Serial port ttyAMA0

Device name	Chip name	TX	RX	RTS
RS232/485(ttyAMA0)	Compute module 3	GPIO32	GPIO33	GPIO17

6. GPIO (Digital input/output)

To check description about mapping [GPIO](#) port please follow the article [Mapping GPIO ports](#)

Mapping GPIO ports

1. Introduction

This document describes the mapping GPIO ports on the platforms NPE-X500 and NPE-9500. GPIO (General Purpose Input / Output) interface is used for communication between components of a computer system, such as a microprocessor and various peripheral. Leads such a device (pins) may be used either as inputs and outputs and is usually configurable property. GPIO pins are often grouped in ports.

x500 devices are shipped with two types of main boards: VERSION 1 and VERSION 2.
You can also have board with some minor improvements (for example *_PUD* for configurable pull up/down mode for DI ports).
GPIO numbers described here matches major versions (1 and 2) and are the same for all minor subversions.

You can check what version do you have by executing following command:

```
getenv | grep MB_VER
```

If this command doesn't work, download [this](#) script, upload it to the device and execute.

2. Port mapping

2.1 Digital Inputs

	1	2	3	4
GPIO	18	19	20	21
Port	DI1	DI2	DI3	DI4

Example for reading a digital input **DI1 (gpio18)**:

```
echo 18 > /sys/class/gpio/export  
cat /sys/class/gpio/gpio18/value
```

2.2 Digital Outputs

For Board Version2 (newer version of x500cm3): DO state can be

changed only if GPIO4 has 0 value.
In other way, DO states will remain the same regardless values written to their gpio files.

	12	11	10	9
GPIO mainboard version 1	22	23	24	25
GPIO mainboard version 2	40	41	24	25
Port	DO1	DO2	DO3	DO4

To effectively change the status of the digital output, you should reset **latch pin**.
For mainboard version 1 it is **3** and for version 2.x is **5**.

Example of change in the digital output DO 1:

- Enter the following command:

```
echo 22 > /sys/class/gpio/export
```

- Change direction to "out", without this we will not be able to change the value.

```
echo out > /sys/class/gpio/gpio22/direction
```

- Change the value of gpio22(DO1) to 1:

```
echo 1 > /sys/class/gpio/gpio22/value
```

- Execute following command:

```
echo 3 > /sys/class/gpio/export
```

- Reset **latch gpio3** (or latch 5 for new mainboard):

```
echo 1 > /sys/class/gpio/gpio3/value
echo 0 > /sys/class/gpio/gpio3/value
```

- Read the status of the output DO 1:

```
cat /sys/class/gpio/gpio22/value
```

2.3 Configurable digital I/O

	5	6	7	8
GPIO	38	37	13	12
Port	DIO3	DIO4	DIO2	DIO1
Changing mode pin mainboard version 1	5		4	
Changing mode pin mainboard version 2	23		22	

Example for setup the mode for DIO1 and DIO2 to DO

```
echo 4 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio4/direction
echo 1 > /sys/class/gpio/gpio4/value
```

Example for setup the mode for DIO1 and DIO2 to DI

```
echo 4 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio4/direction
echo 0 > /sys/class/gpio/gpio4/value
```

Please remember that is not possible to export these GPIO because it is using by the system and npe_service application, these gpio should be exported by device tree or npe_service during boot device. If you want to use it you can use our npe/npe_service application or just open /sys/class/leds/ where it should be exported.

2.4 LEDS / BUZZER

Function	LED1	LED2	LED3	BUZZER	USER BUTTON
GPIO mainboard version X500	43	35	NC	27	39
GPIO mainboard version 9500	43	35	6	27	39

I2C interface

1. Introduction

This document describes how to use I2C interface in general. This is the most often used interface to communication between CPU and other device in Modberry/NPE/iMod platform. In this page you will not find any information about working with particular device.

2. Preparing the tools

To have easier possibility to working with I2C interface please download below package. This software allow use to read, write the value to connected devices and scan I2C interface.

```
sudo apt-get install i2c-tools
```

3.Setup I2C device

3.1 Scan I2C intarface

Firstly, please scan a I2C interface, you should see something like this:

```
root@techbase:~# i2cdetect -y 0
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  18  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  UU  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  UU  --  --
60:  --  --  --  --  --  --  --  --  UU  --  --  --  UU  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
root@techbase:~# ^C
```

In M1000 platform you need to use i2cdetect -y -r 0

Position with UU means that device on such position use a Linux driver, and this is busy device, you can't read by some other own software except native driver. Devices with normal number are a free (withouth enable driver) devices, you can normal use or try to load the Linux native driver if you want and it is exist in the system.

3.2 Prepare the device

Next, depends on your request, you can

- enable the linux driver, if is exist
- disable the linux driver, if you want to use your own software.

Below is the set of commands to this job:

At many times the device name and driver can be different. It is because the driver can be working for the group of the devices.

3.2.1 Disable the driver

```
rmmod "driver name"  
e.g. rmmod mcp3422
```

3.2.2 Check loaded drivers in the system

```
lsmod
```

3.2.3 Check all aliases for the drivers

```
cat /lib/modules/4.4.45-v7+/modules.alias
```

The directory to modules.alias file can be different depending on the linux version.

3.2.4 Load the driver

```
echo "Driver name" "address device" > /sys/bus/i2c/devices/i2c-  
0/new_device  
e.g. echo MCP23017 0x27 > /sys/bus/i2c/devices/i2c-0/new_device
```

3.3 Check if device is working

Below is a examples to read or write the data to I2C device based on i2c-tools

3.3.1 Read data

```
i2cget -y 0 "number device" "register number" -> e.g. i2cget -y 0 0x6c  
0x00
```

3.3.2 Write data

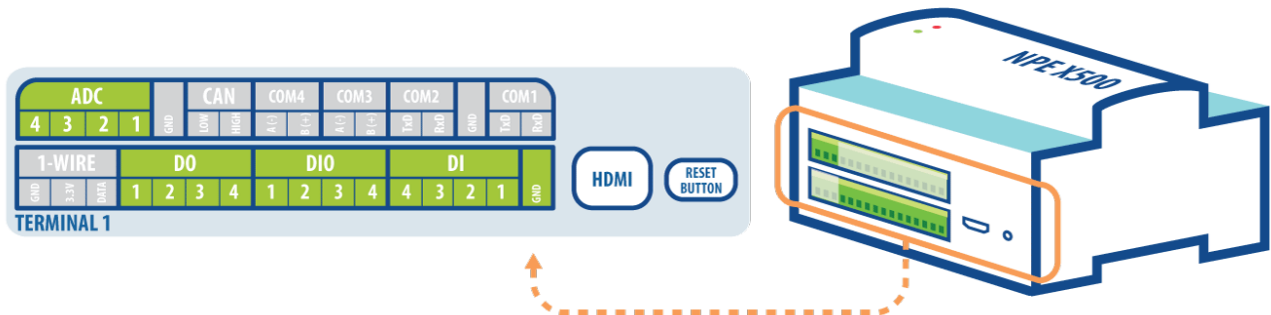
```
i2cset -y 0 "number device" "register number" "value to set" -> e.g.  
i2cset -y 0 0x6c 0x00 0xff
```

Inputs/outputs

1. Introduction

NPE X500 platform is equipped with analog inputs, digital outputs, configurable digital I/O and digital inputs:

- 4x AI
- 4x DO
- 4x DI/DO (configurable)
- 4x DI



2. Technical parameters

Analog Input:

- 4xAI max 18-bit
- max sampling 240sps for 12-bit mode
- 0-10 VDC
- option: 0-20mA
- overvoltage protection transil unidirectional 12V DC
- Peak min 600W"

Digital Output:

- 4x DO Open Collector
- max voltage 30VDC
- max. current load: 500 mA
- overvoltage protection transil unidirectional 30V DC
- Peak min 600W

Digital Input/Output:

- 4x DIO
- DI:
 - 4x DI
 - VIL 0 - 0.5 VDC
 - VIH 1.5 - 30 VDC
- DO:
 - 4x Open Collector
 - max output voltage 30VDC
 - max. current load: 500 mA
- overvoltage protection transil unidirectional 30V DC

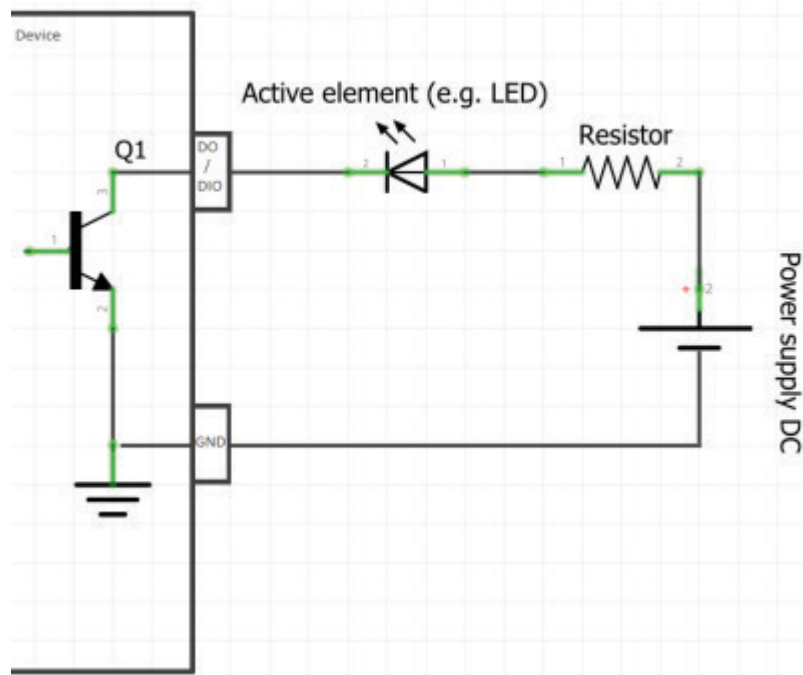
- Peak min 600W

Digital Input:

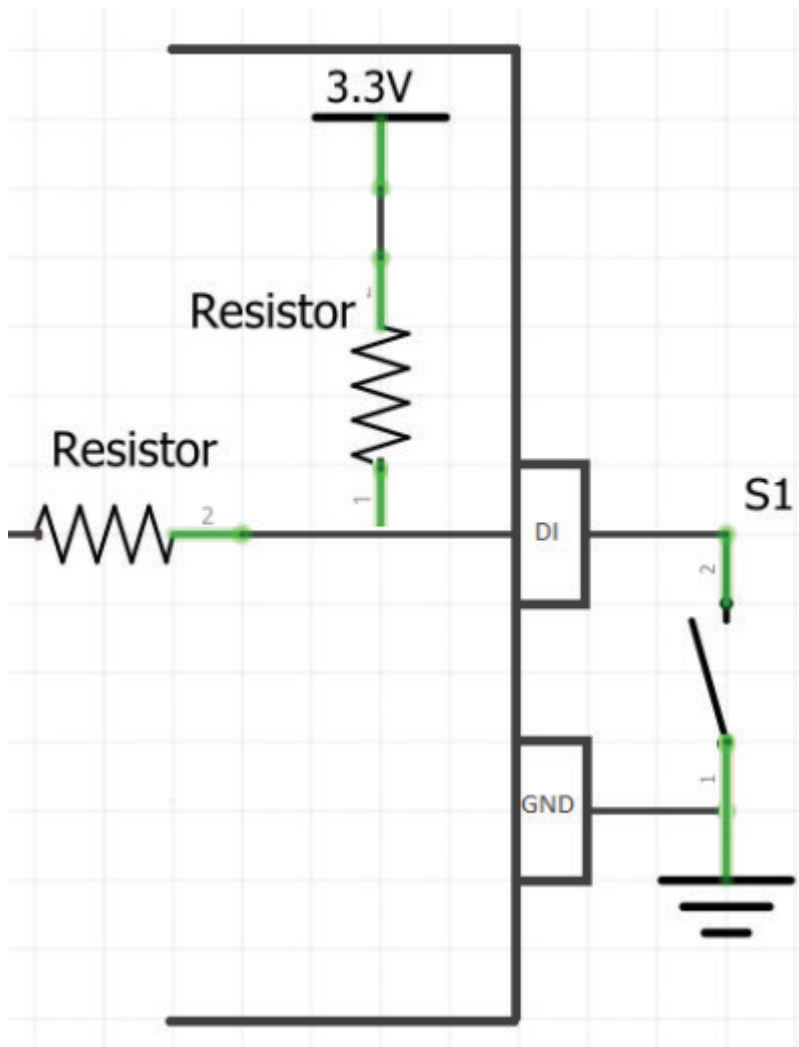
- 4x DI
- VIL 0 - 0.5 VDC
- VIH 1.5 - 30 VDC
- PULL UP 3.3 VDC
- overvoltage protection transil unidirectional 30V DC
- Peak min 600W

3. Schematic diagram

3.1. Digital Output (open collector)



3.2. Digital Input



4. Service

Hardware channel is manageable with **npe** application.

4.1 Npe application

The basic tool for changing the output state is the **npe** application.

Syntax:

```
npe  [+|-|?|~][D0|DIO][NUMBER]
np. npe +D02
```



+	change state to 1
-	change state to 0

?	state check
~	change state to opposite

To change **DO1** input state, use command:

```
npe ~DO1
```

next to verify **DO1** state, use command:

```
npe ?DO1
```

Full **npe** application manual is available after running **npe** command in device's terminal.

4.2 Modmas application

Modmas is an application for changing **modbus** parameters. When the device is properly configured to manage **hardware channel** (see section 4. [Cooperation with iMod application](#)), we can use **modmas** application to change output values.

In order to change output state use **modmas** application as follows: Syntax:

```
modmas write:id:value
```

4.3 Configurable I/Os

NPE X500 platform has 4 configurable inputs/outputs (DIO1-4), which can work as a digital input or output. Operating mode can be changed using **npe** application, after running it enter as a parameter one of virtual ports: **DIOconf1** or **DIOconf2**. For operating mode of **DIO1** and **DIO2** is responsible virtual port: **DIOconf1**, for **DIO3** and **DIO4** - **DIOconf2**.

```
DIOconf1 -> DI01, DI02  
DIOconf2 -> DI03, DI04
```

4.3.1 Changing operating mode (IN/OUT)

To change operating mode (IN/OUT) use **npe** application.

DIO1, DIO2 to OUT mode

```
npe 0DIOconf1
```

DIO1, DIO2 to IN mode

```
npe IDIOconf1
```

Operating mode for DIO3, DIO4 is changed similarly using **DIOconf2** virtual port.

5. Cooperation with iMod application

5.1 Detailed description of config file

Definition of access-channel **Modbus_S1** (modbus channel)

```
<access-channel name="Modbus_S1">
  <protocol name="MODBUS"/>
  <port>"ET-502-TCP"</port>
</access-channel>
```

Definition of source-channel **NPE_io** (hardware channel)

```
<source-channel name="NPE_io">
  <protocol name="HARDWARE"/>
</source-channel>
```

Parameter definition:

```
<parameter type="int16">
  <id>"DI1_101"</id>
  <description><![CDATA["DI1"]]></description>
  <!-- Definition of source-channel in parameter -->
  <source-channel channel-name="NPE_io" parameter-id="DI1"/>
  <!-- Definition of access-channel in parameter -->
  <access-channel channel-name="Modbus_S1" parameter-id="101"/>
</parameter>
```

5.2 Configuration service

Configuration can be managed with **modmas** application (see section [3.2 Modmas application](#)).

Example: In order to change state of DO1 output to value 1, use command:

```
modmas write:201:1
```

Ports and assigned modmas parameters.

Name	Modmas parameter
DO1-DO4	201-204
DIOconf1-2	601-602

DIO1-DIO4	501-504
AI1-AI4	301-304

5.3 Configuration file

Bellow you will find configuration file [\[DOWNLOAD\]](#) :

```
<imod version="1.0.0" parameter-db="true">
  <group name="Channel definition">
    <access-channel name="Modbus_S1">
      <protocol name="MODBUS"/>
      <port>"ET-502-TCP"</port>
    </access-channel>
    <source-channel name="NPE_io">
      <protocol name="HARDWARE"/>
    </source-channel>
    <!-- DI1-4 -->
    <parameter type="int16">
      <id>"DI1_101"</id>
      <description><![CDATA["DI1"]]></description>
      <source-channel channel-name="NPE_io" parameter-id="DI1"/>
      <access-channel channel-name="Modbus_S1" parameter-id="101"/>
    </parameter>
    <parameter type="int16">
      <id>"DI2_102"</id>
      <description><![CDATA["DI2"]]></description>
      <source-channel channel-name="NPE_io" parameter-id="DI2"/>
      <access-channel channel-name="Modbus_S1" parameter-id="102"/>
    </parameter>
    <parameter type="int16">
      <id>"DI3_103"</id>
      <description><![CDATA["DI3"]]></description>
      <source-channel channel-name="NPE_io" parameter-id="DI3"/>
      <access-channel channel-name="Modbus_S1" parameter-id="103"/>
    </parameter>
    <parameter type="int16">
      <id>"DI4_104"</id>
      <description><![CDATA["DI4"]]></description>
      <source-channel channel-name="NPE_io" parameter-id="DI4"/>
      <access-channel channel-name="Modbus_S1" parameter-id="104"/>
    </parameter>
    <!-- D01-4 -->
    <parameter type="int16">
      <id>"D01_201"</id>
      <description><![CDATA["D01"]]></description>
      <source-channel channel-name="NPE_io" parameter-id="D01"/>
      <access-channel channel-name="Modbus_S1" parameter-id="201"/>
    </parameter>
    <parameter type="int16">
```

```

        <id>"D02_202"</id>
        <description><![CDATA["D02"]]></description>
        <source-channel channel-name="NPE_io" parameter-id="D02"/>
        <access-channel channel-name="Modbus_S1" parameter-id="202"/>
    </parameter>
    <parameter type="int16">
        <id>"D03_203"</id>
        <description><![CDATA["D03"]]></description>
        <source-channel channel-name="NPE_io" parameter-id="D03"/>
        <access-channel channel-name="Modbus_S1" parameter-id="203"/>
    </parameter>
    <parameter type="int16">
        <id>"D04_204"</id>
        <description><![CDATA["D04"]]></description>
        <source-channel channel-name="NPE_io" parameter-id="D04"/>
        <access-channel channel-name="Modbus_S1" parameter-id="204"/>
    </parameter>
    <!-- DIOconf1 -->
    <parameter type="int16">
        <id>"DIOconf1_601"</id>
        <description><![CDATA["DIOconf1"]]></description>
        <source-channel channel-name="NPE_io" parameter-id="DIOconf1"/>
        <access-channel channel-name="Modbus_S1" parameter-id="601"/>
    </parameter>
    <parameter type="int16">
        <id>"DIO1_501"</id>
        <description><![CDATA["DIO1"]]></description>
        <source-channel channel-name="NPE_io" parameter-id="DIO1"/>
        <access-channel channel-name="Modbus_S1" parameter-id="501"/>
    </parameter>
    <parameter type="int16">
        <id>"DIO2_502"</id>
        <description><![CDATA["DIO2"]]></description>
        <source-channel channel-name="NPE_io" parameter-id="DIO2"/>
        <access-channel channel-name="Modbus_S1" parameter-id="502"/>
    </parameter>
    <!-- DIOconf2 -->
    <parameter type="int16">
        <id>"DIOconf2_602"</id>
        <description><![CDATA["DIOconf2"]]></description>
        <source-channel channel-name="NPE_io" parameter-id="DIOconf2"/>
        <access-channel channel-name="Modbus_S1" parameter-id="602"/>
    </parameter>
    <parameter type="int16">
        <id>"DIO3_503"</id>
        <description><![CDATA["DIO3"]]></description>
        <source-channel channel-name="NPE_io" parameter-id="DIO3"/>
        <access-channel channel-name="Modbus_S1" parameter-id="503"/>
    </parameter>

```

```
<parameter type="int16">
  <id>"DI04_504"</id>
  <description><![CDATA["DI04"]]></description>
  <source-channel channel-name="NPE_io" parameter-id="DI04"/>
  <access-channel channel-name="Modbus_S1" parameter-id="504"/>
</parameter>
<!-- AI1-4 -->
<parameter type="int16">
  <id>"AI1_301"</id>
  <description><![CDATA["AI1"]]></description>
  <source-channel channel-name="NPE_io" parameter-id="AI1"/>
  <access-channel channel-name="Modbus_S1" parameter-id="301"/>
</parameter>
<parameter type="int16">
  <id>"AI2_302"</id>
  <description><![CDATA["AI2"]]></description>
  <source-channel channel-name="NPE_io" parameter-id="AI2"/>
  <access-channel channel-name="Modbus_S1" parameter-id="302"/>
</parameter>
<parameter type="int16">
  <id>"AI3_303"</id>
  <description><![CDATA["AI3"]]></description>
  <source-channel channel-name="NPE_io" parameter-id="AI3"/>
  <access-channel channel-name="Modbus_S1" parameter-id="303"/>
</parameter>
<parameter type="int16">
  <id>"AI4_304"</id>
  <description><![CDATA["AI4"]]></description>
  <source-channel channel-name="NPE_io" parameter-id="AI4"/>
  <access-channel channel-name="Modbus_S1" parameter-id="304"/>
</parameter>
</group>
</imod>
```

GPIO LIB for C

NPE GPIO library, allows usage of general purpose ports (GPIO).

Although all GPIOs can be used directly, via the kernel API, this library implies it and allows some additional functions, such as saving the state in file.

To use this library, first call the `new_gpio_dev()` function, which initializes the library and returns a structure, which contains pointers to all the other functions.

Then you need a port, which is returned by one of the `open_` functions. Although they take arguments of type `int`

it is strongly suggested that you use one of the enums in `gpio.h`

For all ports other than DIO getting or setting their values is fairly simple - simply call one of the `set_*` or `get_*` functions.

`set_*` and `get_*` functions will fail if port direction is not appropriate.

For DIO you need the port to be set in appropriate direction - either input or output, that is done using the `set_dinout_dir(gpio_port_t* port, int val)` function. Be warned though, DIOs are grouped in fours for x1000

(1-2, 3-4 and so on) or in pairs for x500 device (1-2 and 3-4) and setting direction for one sets it for the whole group.

GPIOlib can use a buffer file for DIOs, (including direction), DOs and relays. This allows checking output and direction

states between programs. DIO directions are ALWAYS buffered and checked against the buffer whenever necessary.

`set_value_buff(gpio_port_t* port, int value)` writes the set value to buffer in `/tmp/iobuf`

`get_value_buff(gpio_port_t* port)` gets the state written in the buffer

Functions descriptions

open_dinout

gpio_port_t open_dinout (int id, int dir)*

opens given dinout

Parameters

- **id** dinout pin id as defined in `gpio_dinout_t`
- **dir** direction, as defined in `gpio_direction_t`

Remarks

- arguments should be typed to their enums, but compatibility with old API requires they be `int`

Returns

`gpio_port_t` pointer or `NULL` on failure

open_dinout_noconf

gpio_port_t open_dinout_noconf(int id)*

opens given dinout, the direction is read from buffer

Parameters

- **id** dinout pin id as defined in `gpio_dinout_t`

Remarks

- arguments should be typed to their enums, but compatibility with old API requires they be int

Return

`gpio_port_t` pointer or NULL on failure

open_optoin

gpio_port_t open_optoin(int id)*

open optoin

Parameters

- **id** optoin id in range 0 to `__DI_MAX`

Return

pointer to created `gpio_port_t` or NULL on failure

open_optoout

gpio_port_t open_optoout(int id)*

open optout

Parameters

- **id** optoout id in range 0 to `__DO_MAX`

Return

pointer to created `gpio_port_t` or NULL on failure

open_relay

gpio_port_t open_relay(int id)*

open relay

Parameters

- **id** relay id in range 0 to __RELAY_MAX

Returns

pointer to created gpio_port_t or NULL on failure

get_analog

int get_analog(gpio_port_t port)*

gets value of given ADC in

Parameters

- **port** ADC in, port comes from open_analog(int id) function in GPIOLib

Returns

port value on success (0, 1 or ADC reading < 4096) or negative error code

open_LED

gpio_port_t open_LED(int id)*

opens LED

Parameters

- **id** LED id from gpio_led_t

Returns

pointer to created gpio_port_t or NULL on failure

open_buzzer

gpio_port_t open_buzzer()*

open buzzer

Returns

pointer to created gpio_port_t or NULL on failure

open_button

gpio_port_t open_button()*

opens user button

Returns

pointer to created `gpio_port_t` or NULL on failure

set_value

int set_value(gpio_port_t port, int value)*

sets value for given GPIO port

Parameters

- **port** GPIO port
- **value** value to set, 0 or 1

Preconditions

- port comes from one of the `open_*` functions in GPIOlib
- GPIO set to output

Returns

0 on success or negative error code

set_value_buf

int set_value_buf(gpio_port_t port, int value)*

sets value for given GPIO port and writes it to buffer in `/tmp/iobuff`

Parameters

- **port** GPIO port
- **value** value to set, 0 or 1

Preconditions

- port comes from one of the `open_*` functions in GPIOlib
- GPIO set to output

Returns

0 on success or negative error code

set_LED

int set_LED(gpio_port_t port, int value)*

sets LEDs

Parameters

- **port** port description
- **value** value to set, 0 or 1

Preconditions

- port comes from open_LED or open_buzzer

Returns

0 or a negative error code

get_LED

int get_LED(gpio_port_t port)*

gets value of given LED port

Parameters

- **port** GPIO port

Preconditions

- port comes from one of the open_* functions in GPIOlib
- GPIO set to input

Returns

port value on success (0 or 1) or negative error code

set_buzzer

int set_buzzer(gpio_port_t port, int value)*

sets buzzer's value

Parameters

- **port** port description
- **value** value to set, 0 or 1

Preconditions

- port comes from open_LED or open_buzzer

Returns

0 or a negative error code

get_buzzer

int get_buzzer(gpio_port_t port)*

gets value of given BUZZER port

Parameters

- **port** GPIO port

Preconditions

- port comes from one of the open_* functions in GPIOlib
- GPIO set to input

Returns

port value on success (0 or 1) or negative error code

get_value

int get_value(gpio_port_t port)*

gets value of given GPIO port other than ADC

Parameters

- **port** GPIO port

Preconditions

- port comes from one of the open_* functions in GPIOlib
- GPIO set to input

Returns

port value on success (0 or 1) or negative error code

get_value_buf

int get_value_buf(gpio_port_t port)*

gets value of given buffered GPIO port as last saved in buffer

Parameters

- **port** GPIO port

Preconditions

- port comes from one of the open_* functions in GPIOlib

Returns

port value on success (0 or 1) or negative error code

int get_analog_value(gpio_port_t port)*

gets value of given ADC in

Parameters

- **port** ADC in

Preconditions

- port comes from open_analog(int id) function in GPIOlib

Returns

port value on success (0, 1 or ADC reading < 4096) or negative error code

set_dinout_dir

int set_dinout_dir(gpio_port_t port, int dir)*

sets direction for given dinout

Parameters

- **port** port description
- **dir** direction, as defined in gpio_direction_t

Preconditions

- port comes from open_dinout()

Remarks

- id should be of type gpio_direction_t, but compatibility with old API requires it is int

Returns

0 on success or negative error code

get_dinout_dir

```
int get_dinout_dir(gpio_port_t* port)
```

reads gpio direction from the buffer

Parameters

- **port** port which to check

Returns

0 input, 1 output or negative error code

close

```
void close_port(gpio_port_t* port)
```

close given GPIO port

Parameters

- **port** port which to be closed

Building

On x500/x500cm3/m1000 the best way would be build program on the device.

gpio.h file is located in */usr/local/include/gpio.h* on these devices.

Upload code to device and build it.

Example command for C:

```
gcc -o program program.c -lx1000gpio
```

Examples

Examples can be downladed [here](#).

Commands to build examples:

```
gcc -o adc_example adc_example.c -lx1000gpio
gcc -o buffer_example buffer_example.c -lx1000gpio
gcc -o DI_D0_example DI_D0_example.c -lx1000gpio
gcc -o dinout_example dinout_example.c -lx1000gpio
gcc -o lb_example lb_example.c -lx1000gpio
```

Detect event on the GPIO by RPi.GPIO

Below document describe and contains an example to have a possibility to catch events like a RISING, FALLING or BOTH on the corresponding GPIO. This function realize by method `add_event_detect` by RPi.GPIO library. Good description you can find on the [link](#).

1. Code description

First, we define all GPIO's which we want to use it and setup the correct mode, in this case it will be input. We need also to setup if this input will be pull-up or down, like a default state. In our example code we have "PUD_UP" which is pull-up.

```
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
```

Then, we can connect `add_event_detect` to corresponding input port. In this case, we need to define :

- what edge this event should be react (RISING, FALLING or BOTH),
- callback function
- is also possibility to define bouncetime which allow us to ignoring further edges.

```
#Add event both(RISING,FALLING) for GPIO36 and define callback function
GPIO.add_event_detect(self.hardware['GPIO36'],GPIO.BOTH, callback= lambda
x: self.diPressed(), bouncetime=10)
```

2. Full code

Please download the file [gpio_event_detect.tar.gz](#) and upload to the device e.g. by sftp to the directory `/home/user`. Then please login to the device by ssh and run below command:

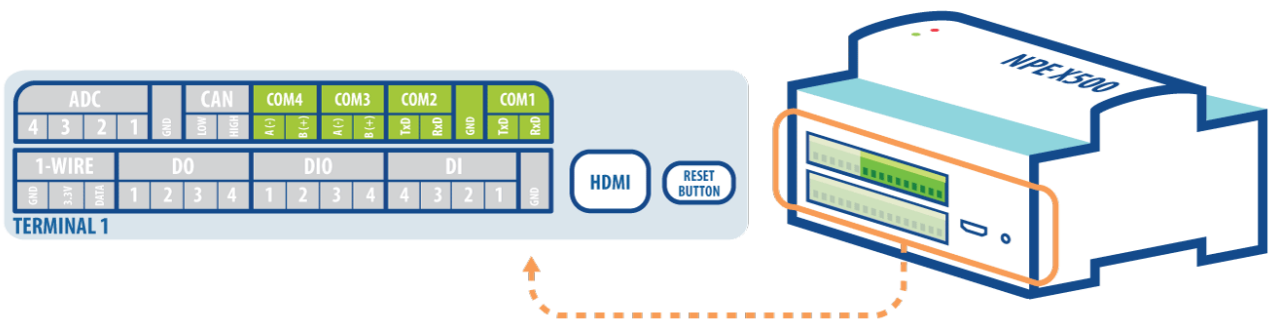
```
sudo su
cd /home/user/
tar -xvf gpio_event_detect.tar.gz
cd gpio_event_detect
python gpioChange.py
```

Serial ports - X500

1. Introduction

Serial port is computer port, which transfers data in string of bits. NPE X500 platform has 4 serial ports - 2x RS-232(3-pin) and 2x RS-485(2-pin). But simultaneously can operate only two of them: **COM1** or **COM4** and **COM2** or **COM3**.

- COM1 - RS-232
- COM2 - RS-232
- COM3 - RS-485
- COM4 - RS-485



2. Technical parameters

- 2xRS-232 / 2xRS-485
- 3pin / 2pin
- overvoltage protection transil bidirectional 24V DC
- Peak 600W

3. Port mapping in Linux operating system

```
COM1/COM4 <=> /dev/ttyAMA0
COM2/COM3 <=> /dev/ttySC0
```

4. Operating mode

ttysC0 and **ttysMA0** ports can work in 2 modes - **RS-485** or **RS-232**. By default **ttysMA0**(COM1 - RS232) port works as a service port. Before the first connecting an external device to the serial ports, you must set the operating mode (RS-232/RS-485).

4.1 ttysMA0 as service port

In order to check if **ttysMA0** port works as a service port, run following command:

```
/root/scripts/service_port_ctrl status
```

If so, you can disable it with command

```
/root/scripts/service_port_ctrl off
```


To enable it as a service port, use command:

```
/root/scripts/service_port_ctrl on
```

4.2 Changing operating mode

To change operating mode of port (RS-232/RS-485) use **comctrl** command:

```
comctrl 1 RS-232 2 RS-232 - both ports work as a RS-232  
comctrl 1 RS-485 2 RS-485 - both ports work as a RS-485
```

To effectively change the operating mode of ports, a restart is required.

5. Cooperation with iMod application

At the bottom of this page you will find sample configurations, which can be used to check if the serial ports are running properly. First step should be uploading a configuration to the device. Next perform a loop - connect Następnie wykonać pętle, connect to each other corresponding ports (bellow you will find connection scheme). This test allows checking status of connection.

Check if **ttyAMA0** port is running as a service port

```
/root/scripts/service_port_ctrl status
```

If so, it should be disabled with following command

```
/root/scripts/service_port_ctrl off
```

1. Connect corresponding ports with each other (perform a loop).

```
COM1 <--> COM2 (232)  
COM3 <--> COM4 (485)
```

To change operating more of port (RS-232/RS-485) use **comctrl** command:

```
comctrl 1 RS-232 2 RS-232
comctrl 1 RS-485 2 RS-485
```

2. Test

Use **modmas** command to perform a test:

```
modmas write:parameter_id:value
np. modmas write:101:12
```

or

```
modmas -p baudrate,8N1 -a RTU:/dev/ttyXX write:parameter_id:value
np. modmas -p 115200,8N1 -a RTU:/dev/ttySC0 write:1:1
```

3. Configuration.

For the purpose of the test there were prepared 2 configurations, but there are 4 cases (COM1 and COM2 for RS-232 or RS-485). Therefore, to test all cases, during the operation of configuration, change the ports' operating mode.

1. Configuration where COM1/COM4 (/dev/ttyAMA0) is Source Channel, and COM2/COM3 (/dev/ttySC0) is Access Channel.
2. Configuration where COM2/COM3 (/dev/ttySC0) is Source Channel, and COM1/COM4 (/dev/ttyAMA0) is Access Channel.

1. [rs_x500_sc_ttyama0_ac_ttysc0.xml](#)
2. [rs_x500_sc_ttysc0_ac_ttyama0.xml](#)

1.

```
<imod version="1.2.45">
  <group name="Definitions 1">
    <access-channel name="ttySC0">
      <protocol name="MODBUS">
        <property name="type" value="RTU"/>
      </protocol>
      <port>"/dev/ttySC0-115200-8N1"</port>
      <property name="device-id" value="1"/>
      <property name="varspace" value="INPUT"/>
    </access-channel>
    <source-channel name="ttyAMA0">
      <protocol name="MODBUS">
        <property name="type" value="RTU"/>
      </protocol>
      <port>"/dev/ttyAMA0-115200-8N1"</port>
      <cycle>"2s"</cycle>
    </source-channel>
    <access-channel name="MODBUS_TCP_Access_1">
      <protocol name="MODBUS">
```

```

        <property name="type" value="TCP"/>
    </protocol>
    <port>"ET-502-TCP"</port>
    <property name="device-id" value="1"/>
    <property name="varspace" value="INPUT"/>
</access-channel>
<parameter type="int16">
    <id>"101_ttySC0_AC"</id>
    <access-channel channel-name="ttySC0" parameter-id="101"/>
    <init-value>"1"</init-value>
    <access-channel channel-name="MODBUS_TCP_Access_1"
parameter-id="101"/>
</parameter>
<parameter type="int16">
    <id>"100_ttyAMA0_SC"</id>
    <source-channel channel-name="ttyAMA0" parameter-id="101"/>
    <access-channel channel-name="MODBUS_TCP_Access_1"
parameter-id="100"/>
</parameter>
</group>
</imod>

```

2.

```

<imod version="1.2.45">
    <group name="Definitions 1">
        <access-channel name="ttyAMA0">
            <protocol name="MODBUS">
                <property name="type" value="RTU"/>
            </protocol>
            <port>"/dev/ttyAMA0-115200-8N1"</port>
            <property name="device-id" value="1"/>
            <property name="varspace" value="INPUT"/>
        </access-channel>
        <source-channel name="ttySC0">
            <protocol name="MODBUS">
                <property name="type" value="RTU"/>
            </protocol>
            <port>"/dev/ttySC0-115200-8N1"</port>
            <cycle>"2s"</cycle>
        </source-channel>
        <access-channel name="MODBUS_TCP_Access_1">
            <protocol name="MODBUS">
                <property name="type" value="TCP"/>
            </protocol>
            <port>"ET-502-TCP"</port>
            <property name="device-id" value="1"/>
            <property name="varspace" value="INPUT"/>
        </access-channel>
    </group>
</imod>

```

```
<parameter type="int16">
  <id>"101_ttyAMA0_AC"</id>
  <access-channel channel-name="ttyAMA0" parameter-id="101"/>
  <init-value>"1"</init-value>
  <access-channel channel-name="MODBUS_TCP_Access_1"
parameter-id="101"/>
</parameter>
<parameter type="int16">
  <id>"100_ttySC0_SC"</id>
  <source-channel channel-name="ttySC0" parameter-id="101"/>
  <access-channel channel-name="MODBUS_TCP_Access_1"
parameter-id="100"/>
</parameter>
</group>
</imod>
```

Service port

Serial port COM1 (RS-232) acts as a **service port** by default.

In order to check if **ttyAMA0** works as a service port, run following command:

```
service_port_ctrl status
```

If service port is disabled, you can enable it with command:

```
service_port_ctrl on
```

Connecting to device via service port

1. Connect service cable into COM1 port of a device.
2. Using **Putty** open computer port, to which you plugged your device
 - **Baudrate:** 115200
3. Login data:
 - **login:** root
 - **password:** techbase

LEDS/BUZZER

1. Introduction

This document describes the LEDs and BUZZER function in the NPE platform. Both function are working using GPIO. The mapping you can find on the link [leds_buzzer](#). Please remember that in the default, you can't setup the state of the LEDs and BUZZER by GPIO because the pins are busy by system. To remove this, you should change the device tree and remove part about leds and buzzer. We are provider two possibility to manage LEDs and BUZZER, npe and /sys/class.

2. NPE application

The basic tool for changing the LED/BUZZER state is the **npe** application.

Syntax:

```
npe  [+ | - | ? | ~] [LED1 | LED2 | BUZZER]  
np.  npe +LED1
```

+	change state to on
-	change state to off
?	state check
~	change state to opposite

To change **LED1** state, use command:

```
npe ~LED1
```

next to verify **LED1** state, use command:

```
npe ?LED1
```

Full **npe** application manual is available after running **npe** command in device's terminal.

3. System driver

If you want to manipulate the LEDs or BUZZER by system driver you can use `/sys/class/leds`.

```
lrwxrwxrwx 1 root root 0 Sep 11 13:51 BUZZER ->
../../devices/platform/soc/soc:leds/leds/BUZZER
lrwxrwxrwx 1 root root 0 Sep 11 13:51 LED1 ->
../../devices/platform/soc/soc:leds/leds/LED1
lrwxrwxrwx 1 root root 0 Sep 11 13:51 LED2 ->
../../devices/platform/soc/soc:leds/leds/LED2
```

To change **LED1** state to on, use command:

```
cd /sys/class/leds/LED1
echo 255 > brightness
```

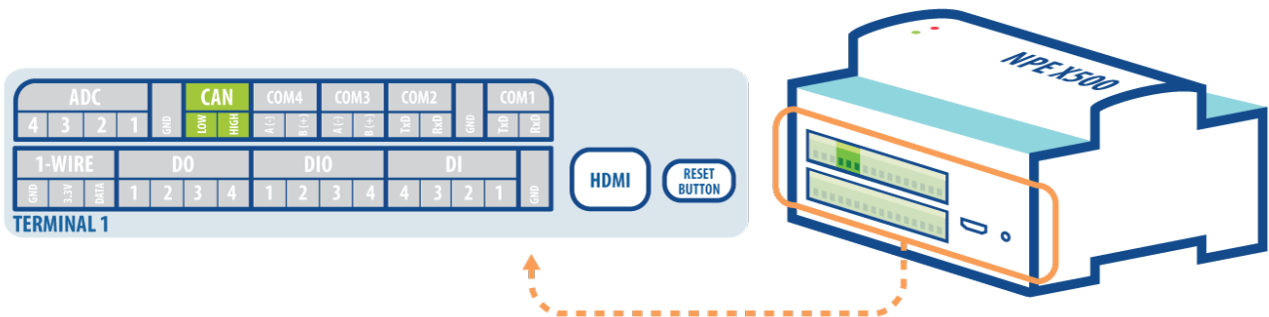
next to verify **LED1** state, use command:

```
cd /sys/class/leds/LED1
cat brightness
```

CAN bus in NPE-X500

1. Introduction

CAN (Controller Area Network) bus is a serial bus originally designed for automotive industry. During the communication there is no specified master device. Communication is broadcast-typed, all connected devices receive exactly the same message.



2. Technical parameters

CAN:

- 1x CAN
- overvoltage protection transil bidirectional 24V DC
- Peak 600W

3. Package installation description

CAN bus support for new devices is preinstalled on system. In situation when there's a need to reinstall the package (for example when user deletes files essential for running the program) you can instal CAN package using following command:

- For x500softmgr update can -b x500-test
- For x500cm3softmgr update can -b x500cm3-test

If the package has been installed properly, there will be available following applications: canbusload, canfdtest, cangw, canplayer, cansniffer, can-calc-bit-timing, candump, cangen, canlogserver, cansend.

Main commands:

candump: shows CAN network traffic
cansend: sends frame
cangen: generates CAN frames (useful in tests)
canplayer: sends CAN frames from file

In order to run CAN interface, run following commands:

- `ip link set can0 up type can bitrate 10000`
- `ifconfig can0 up`

4. Examples of application

To check CAN bus status you can connect two X500 devices and send a frame between them.

Connection:

- Low-to-Low
- High-to-High

5. Bus connection using can-utils.

Device A:

```
cansend can0 5A1#11.2233.44556677.88
```

Device B:

```
candump can0
```

TPM 2.0 (trusted platform module)

1. Introduction

TPM 2.0 is an international standard for a secure cryptoprocessor, a dedicated microcontroller designed to secure hardware through integrated cryptographic keys. We are using [TPM SLx 9670 TPM2.0](#) with SPI interface from Infineon.

2. How to test

To make a test we are using eltt2 software, below is the list of command to download, compile and run this application.

```
git clone https://github.com/Infineon/eltt2
loc=$( pwd ); cd $loc/eltt2; gcc -o eltt2 eltt2.c
./eltt2 -gc
```

Description how is this software working you can read on the page <https://github.com/Infineon/eltt2> on chapter "2. Usage of ELTT2"

3. How to configure on clean system

- First, please check that you have linux 4.19 (uname -a), if not, you can flash your device with this instruction http://documents.techbase.eu/doku.php?id=en:npex500:cm_flashing_with_x500
- Next step, add information for overlays that we have tpm

```
exist=$( cat /root/scripts/u-boot/overlays_list )
overlays="letstrust-tpm "$exist
echo "$overlays" > /root/scripts/u-boot/overlays_list
actual_mac=$( getenv HOST_MAC | tr "=" " " | awk {'print $2'} )
/root/scripts/set_mac_in_uboot.sh $actual_mac
```

- Now, please do reboot

OneWire

1. Introduction

Onewire is an interface and communication protocol. It uses only one data line (and the zero line) for communication. Receiver can be powered directly from the data line using parasitic power, which is an advantage of this interface.

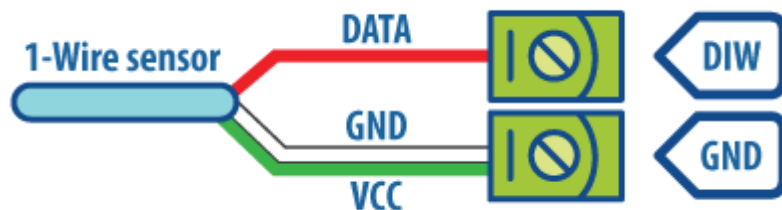
2. Package installation

To instal the OneWire bus service package, you need do run the following command:

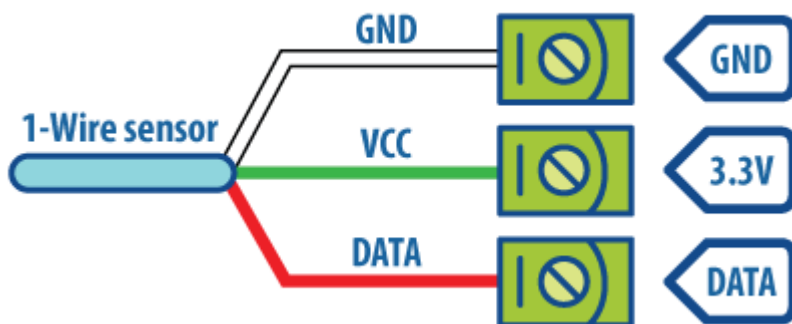
```
softmgr update owserver
```

3. Connection

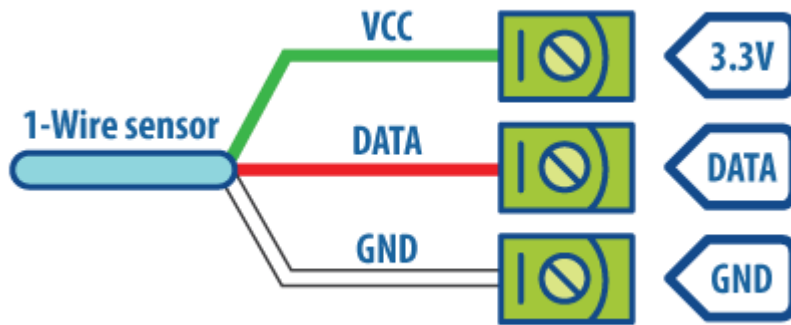
NPE 9000 platform:



NPE X500 platform:



NPE 9500 platform:



4. Reading data from 1-Wire sensor

Using `owlist info` command, the **npe** device will list all the connected sensors and read values from them.

```
[root@techbase123 ~]# owlist info
id: 28D66173060000 chip: DS18B20 desc: THERMOMETER value: 24.3125
id: 285E8273060000 chip: DS18B20 desc: THERMOMETER value: 24.3125
id: 28DEE198050000 chip: DS18B20 desc: THERMOMETER value: 24.5
```

5. Integration with iMod application

iMod application allows quick integration of connected 1-Wire sensors with our iMod device's configuration. After running a command:

```
imod scan onewire
```

OneWire bus will be scanned for connected sensors and will automatically generate configuration to read data from the sensors.

OneWire configuration path:

```
/mnt/mtd/iMod/config/ONEWIREScan.xml
```

Sample file generated after running above command:

```
<?xml version="1.0" encoding="UTF-8" ?>
<imod version="1.2.112" parameter_db="false">
  <group name="ONEWIRE channels group">
    <access-channel name="Modbus">
      <protocol name="MODBUS"/>
      <port>"ET-1502-TCP"</port>
    </access-channel>

    <source-channel name="OneWire">
      <protocol name="ONEWIRE"/>
      <port>"ET-0.0.0.0"</port>
```

```

        <gap>"0"</gap>
        <cycle>"10"</cycle>
    </source-channel>

</group>

<group name="THERMOMETER_DS18B20 parameters group">
    <parameter type="real32">
        <id>"THERM171_341"</id>
        <description>"THERMOMETER_DS18B20"</description>
        <source-channel channel-name="OneWire"
parameter-id="28CF4FD8040000:temperature"/>
        <access-channel channel-name="Modbus"
parameter-id="11340"/>
    </parameter>

    <parameter type="real32">
        <id>"THERM172_343"</id>
        <description>"THERMOMETER_DS18B20"</description>
        <source-channel channel-name="OneWire"
parameter-id="28CF4FD8040000:temperature9"/>
        <access-channel channel-name="Modbus"
parameter-id="11342"/>
    </parameter>

    <parameter type="real32">
        <id>"THERM173_345"</id>
        <description>"THERMOMETER_DS18B20"</description>
        <source-channel channel-name="OneWire"
parameter-id="28CF4FD8040000:temperature10"/>
        <access-channel channel-name="Modbus"
parameter-id="11344"/>
    </parameter>

    <parameter type="real32">
        <id>"THERM174_347"</id>
        <description>"THERMOMETER_DS18B20"</description>
        <source-channel channel-name="OneWire"
parameter-id="28CF4FD8040000:temperature11"/>
        <access-channel channel-name="Modbus"
parameter-id="11346"/>
    </parameter>

    <parameter type="real32">
        <id>"THERM175_349"</id>
        <description>"THERMOMETER_DS18B20"</description>
        <source-channel channel-name="OneWire"
parameter-id="28CF4FD8040000:temperature12"/>
        <access-channel channel-name="Modbus"

```

```
parameter-id="11348"/>
    </parameter>

    <parameter type="real32">
        <id>"THERM166_331"</id>
        <description>"THERMOMETER_DS18B20"</description>
        <source-channel channel-name="OneWire"
parameter-id="2823E773060000:temperature"/>
        <access-channel channel-name="Modbus"
parameter-id="11330"/>
    </parameter>

    <parameter type="real32">
        <id>"THERM167_333"</id>
        <description>"THERMOMETER_DS18B20"</description>
        <source-channel channel-name="OneWire"
parameter-id="2823E773060000:temperature9"/>
        <access-channel channel-name="Modbus"
parameter-id="11332"/>
    </parameter>

    <parameter type="real32">
        <id>"THERM168_335"</id>
        <description>"THERMOMETER_DS18B20"</description>
        <source-channel channel-name="OneWire"
parameter-id="2823E773060000:temperature10"/>
        <access-channel channel-name="Modbus"
parameter-id="11334"/>
    </parameter>

    <parameter type="real32">
        <id>"THERM169_337"</id>
        <description>"THERMOMETER_DS18B20"</description>
        <source-channel channel-name="OneWire"
parameter-id="2823E773060000:temperature11"/>
        <access-channel channel-name="Modbus"
parameter-id="11336"/>
    </parameter>

    <parameter type="real32">
        <id>"THERM170_339"</id>
        <description>"THERMOMETER_DS18B20"</description>
        <source-channel channel-name="OneWire"
parameter-id="2823E773060000:temperature12"/>
        <access-channel channel-name="Modbus"
parameter-id="11338"/>
    </parameter>

</group>
```

```
</imod>
```

To run configuration that was generated by the iMod application, edit the second line of the configuration file:

```
<imod version="1.2.112" parameter_db="false"> ==> <imod  
version="1.2.112" parameter_db="true">
```

Then run ***imod start*** command.

6. Troubleshooting

Useful commands:

1-Wire Server	Service for accessing the one-wire bus sensors.
owlist info	listing all connected 1-Wire sensors with their values
owrun restart	owserver restart
owtest	testing one sensor at specified time

Analog input

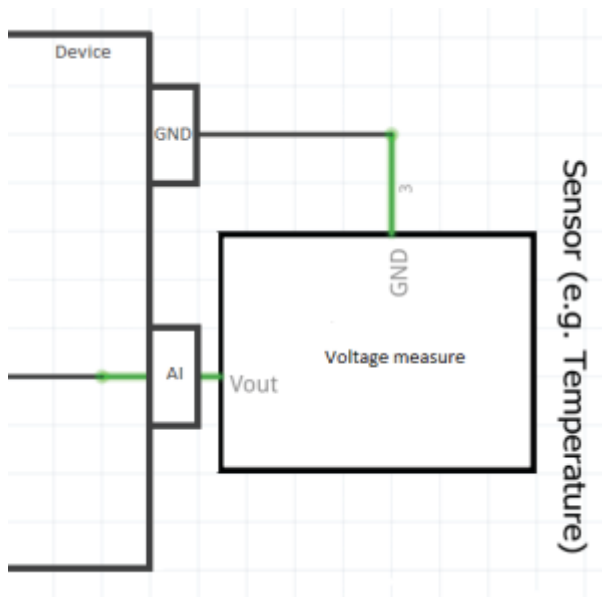
1. Introduction

Analog input is a special interface which use [ADC](#) to possibility measure the analog data like a temperature, humidity and convert this to digital value. This document describes how to use analog input, especially how to setup the scale, frequency and read the value.

This documents describe the analog inputs located on the main board, described on the box like a ANALOG IN 1-4.

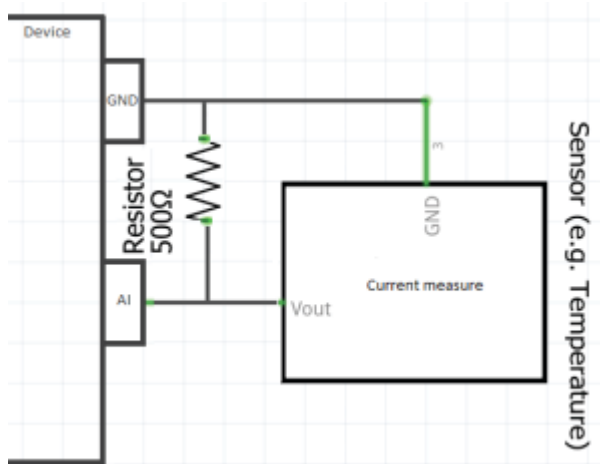
2. Schematic diagram

2.1 Analog input - Voltage measure



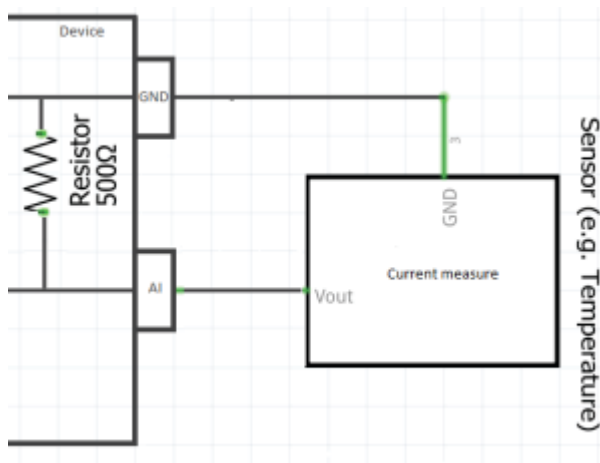
2.2 Analog input - Current measure with additional resistor

If you have a version with voltage measure (in default we are produce such version) but you want to measure the current you need to add extra resistor (500 ohm) between analog input channel and the ground. This situation is showing below



2.3 Analog input - Current measure (on special request)

If you want to measure current you can always send us a special request to do this during the production.



3. Using analog inputs in system which is not delivery by TECHBASE

Below information is helped in situation when you need to use your own compilation or standard raspbian operating system. Firstly, please note it that we are using [MCP3424](#) chip to have analog input, this device should be visible on position 0x6C in the I2C interface or in some time it can be also 0x6b, 0x6d, 0xe, 0x6f. For setup this device in your system please follow [I2C interface](#).

4. Using analog inputs in standard system which is delivery by TECHBASE

In default we are using standard native Linux driver which is loaded automatically by device tree. In the typical application this is the best solution. If you want to disable this driver, please check the document about [I2C interface - prepare the device](#)

4.1 Checking the driver functionality

Go to the location `/sys/bus/i2c/devices/0-006c/iio:device0` and using the command **ls** you'll get these files:

Please remember, if the device is on other address than 6c the path for the driver will be other e.g `/sys/bus/i2c/devices/0-006b/iio:device1`

```
root@raspberrypi:/sys/bus/i2c/devices/0-006c/iio:device0# ls
dev                in_voltage2_raw    in_voltage_scale_available
in_voltage0_raw    in_voltage2_scale  name
in_voltage0_scale  in_voltage3_raw    sampling_frequency_available
in_voltage1_raw    in_voltage3_scale  subsystem
in_voltage1_scale  in_voltage_sampling_frequency uevent
root@raspberrypi:/sys/bus/i2c/devices/0-006c/iio:device0#
```

- Files **...raw** - for reading analog input values
- Files **...scale** - scale of reading
- File **in_voltage_scale_available** - all available scales
- File **sampling_frequency_available** - all available sampling frequency

4.2 Reading analog input values

For reading analog input value, you should read the appropriate file **...raw**. e.g. If you want to read AI1 value, you should read **in_voltage0_raw** file.

```
cat in_voltage0_raw
```

The same function can be performed using the command **npe ?AI1**.

4.3 Changing the scale reading

To change the reading scale of analog inputs, change the value stored in file **...scale**. Before the change, read the file **in_voltage_scale_available** to know all the possible values.

```
root@raspberrypi:/sys/bus/i2c/devices/0-006c/iio:device0# cat in_voltage_scale_available
0.000015625 0.000007812 0.000003906 0.000001953
```

Next save proper value to one of the files **...scale**.

e.g To change the scale reading for AI1:

```
echo 0.000015625 > /sys/bus/i2c/devices/0-006c/iio:device0/in_voltage0_scale
```

4.4 Changing the sampling frequency/reading accuracy of analog inputs

You can change the sampling frequency using **in_voltage_sampling_frequency**. Before the change, read the file **sampling_frequency_available** to know all the possible values. The values stored in this file translate into resolution (precision) of measurement in bits:

```
root@raspberrypi:/sys/bus/i2c/devices/0-006c/iio:device0# cat sampling_frequency_available
240 60 15 3
root@raspberrypi:/sys/bus/i2c/devices/0-006c/iio:device0#
```

- 240 = 11-bit
- 60 = 13-bit
- 15 = 15-bit
- 3 = 17-bit

Use echo to write value to file **in_voltage_sampling_frequency**.

```
echo 3 > /sys/bus/i2c/devices/0-006c/iio:device0/in_voltage_sampling_frequency
```

4.5 Calculating voltage from raw data

ADC input based on MCP3424 can measure the value up to 11,6363V, this is due to the fact that we have voltage divide on input. The value of this divide is 0,1760%. The maximal value which can be measure by MC3424 is 2,048V, so $0,1760\% * x = 2,048V$ then $X = 11,6363V$.

Regarding to above explanation, to calculate volage value from raw data you need to do simple mathematic formula, for 11,6363V we will receive maximal value (2048 for 12 bits), so for each point is $11,6363/2048$ (0,0056817~). The maximal value for 12 bits is 2048 but not 4096 because we are measure one way (from 0 to +10V) this is why for each scale is -1 e.g. 14bit is 8192 but not 16384.

Hardware Watchdog

1. Introduction

This document describes how to use watchdog on the NPE-X500 and NPE-9500

In default watchdog is turn on in both versions

2. Configure the watchdog in CM1

Firstly we need to add the watchdog module to load automatically

```
sudo modprobe bcm2708_wdog  
sudo sh -c "echo 'bcm2708_wdog' » /etc/modules"
```

Next, we need to install watchdog service

```
sudo apt-get install watchdog chkconfig
```

Last stage is configure the watchdog and enabled it. Please open the file `sudo nano /etc/watchdog.conf` and uncomment two lines

```
watchdog-device = /dev/watchdog  
max-load-1 = 24
```

Now you can define the heartbeat, enabled the service and reboot the unit

```
echo "options bcm2708_wdog nowayout=1 heartbeat=13" | sudo tee  
/etc/modprobe.d/watchdog.conf  
sudo chkconfig watchdog on  
sudo service watchdog start  
sudo reboot
```

3. Configure the watchdog in CM3

Firstly we need to add the watchdog module to start automatically

```
sudo modprobe bcm2835_wdt  
echo bcm2835_wdt | sudo tee -a /etc/modules
```

Next, we need to install watchdog service

```
sudo apt-get install watchdog chkconfig
```

Last stage is configure the watchdog and enabled it. Please open the file `sudo nano`

/etc/watchdog.conf, uncomment two lines

```
watchdog-device = /dev/watchdog  
max-load-1 = 24
```

and add one extra

```
watchdog-timeout = 10
```

Check if file */lib/systemd/system/watchdog.service* contains following lines:

```
[Install]  
WantedBy=runlevel2.target runlevel3.target runlevel4.target  
runlevel5.target
```

if not - add them.

Now you can enable the service and reboot the unit

```
sudo chkconfig watchdog on  
sudo service watchdog start  
sudo reboot
```

4. Check if is working

To test the functionality, you can use the simple python script

```
#!/bin/python  
#fork_bomb.py  
  
import os  
  
while True:  
    os.fork()
```

5. Autostart

To run the service at boot time we need to open the file

```
/lib/systemd/system/watchdog.service
```

and add two lines at top

```
[Install]  
WantedBy=multi-user.target
```

After this we can run

```
systemctl enable watchdog.service
```

Fixing DO/DI ports handling

Described here functionality is available for devices with core package version **>=1806121000**.

If DI and DO ports don't work in your device, you can fix this issue with following command:

```
fix_core
```

It reinstalls services responsible for DI/O ports.

After that you have to restart system:

```
reboot
```

Script `fix_core` can be executed only once.

After first execution files necessary for fixing are removed (they shouldn't be needed any more).

If you want to run this script again, you have to (re)install core package.

If you compiled your own program linking to static version of `gpiolib`, you will have to compile it once again after fixing process.

OLED Display

1. Introduction

OLED display module have a 0.96 inch and resolution 128×64. It can communicate by I2C bus and allow to display simple message.

2. How it's working

The sample code was creating in python. We created special service which allow us to send message to the oled by mqtt protocol. The oled side was created with PIL and SSD1306 library. All messages which was send to the OLEDService is storing in queue and display on the display one by one. The software have rotation function so if the oled can't show more then will delete the oldest and add new.

We have 2 topics, one to add new message and one to change the header :

- "OLED/info" - add new message
- "OLED/header" - change header

We need below software to run it:

- apt-get install mosquitto -y
- apt-get install python-pip -y
- apt-get install python-pil -y
- apt-get install mosquitto-clients -y
- pip install paho-mqtt

3. Download and run

Please download [OLEDService](#) application, unzip and run it by command "python OLEDService.py", on other terminal you can make a test and send below message:

- mosquitto_pub -t 'OLED/header' -m 'Test'
- mosquitto_pub -t 'OLED/info' -m 'Hello World'

You should see this message on the OLED.

4. Autostart / Check if is working

4.1 Autostart

To run service automatically you need to create [autostart](#) script, below is the example

```
#!/bin/bash

### BEGIN INIT INFO
# Provides:          script
# Required-Start:    $all
# Required-Stop:     $network
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: short description
```



```
# Description:      description
### END INIT INFO

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
start(){
    python /home/user/OLED/OLEDService.py &
}
stop(){
    killall python
}
case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        ;;
    *)
        echo "Usage: $0 {start|stop|restart}"
esac

exit 0
```

4.2 Check if is working

To know if the application is working you can run simple command:

```
ps aux | grep OLEDService
```

SPI - outside

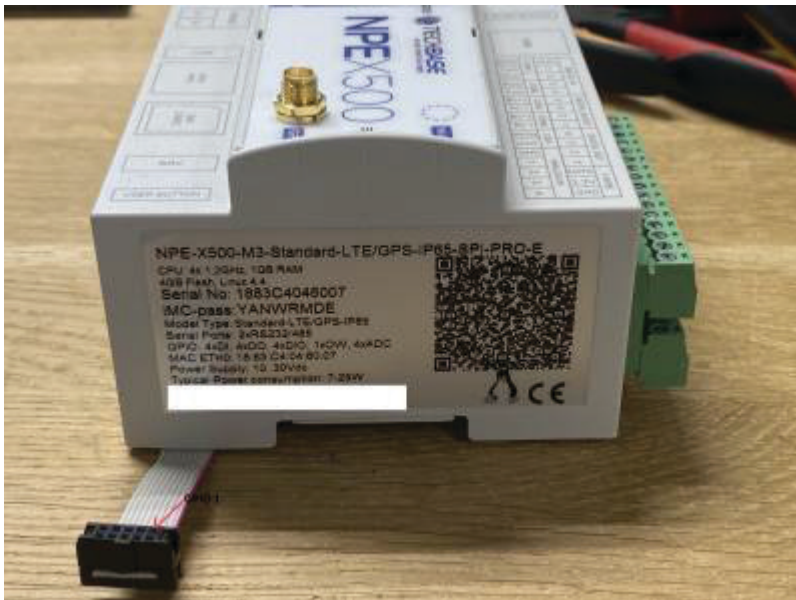
1. Introduction

This documents describe SPI connector which is mounted external, this is special version which can be done on demand.

2. GPIO mapping

When we look at the device at ethernet port side then the first pin is on the left top side of the connector. Below is the picture where this is marked

We strong recommended to manually detect what side is where, it can by done by measure the voltage and find 5VDC (PIN 3) and 3.3VDC (PIN 10)



GPIO	Function	GPIO	Function
1	CS0	2	NC
3	5VDC	4	MISO
5	NC	6	MOSI
7	GPIO34	8	CLK
9	GND	10	3.3VDC

Autostart - Preparing application to run in autostart

Introduction

Responsible for script running at boot in Linux operating system is **init** program (initialize). **init** is runned first in unix systems (e.g. Linux) by the kernel during operating system boot. Next, based on configuration files or start scripts, **init** runs other system processes, for which it is the parent process.

Preparing and adding your own scripts to autostart

In order to run your own application/script in autostart mode you need to prepare proper file (start script) and place it in following catalog:

```
/etc/init.d
```

Add a header in your start script, right after `#!/bin/bash` :

```
#!/bin/bash

### BEGIN INIT INFO
# Provides:          script
# Required-Start:    look_below_for_sample_input
# Required-Stop:     look_below_for_sample_input
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: proba
# Description:       Enable service provided by daemon.
### END INIT INFO
```

If we add following phrase in Required section :

```
$all - will be added at the end
$local_fs - mounting file systems
$remote_fs - mounting the remote file system
$syslog - login operations
$network - booting the network interface
```

The safest way is to enter **\$all** as the script will be starting after all services. The \$ at the beginning means that these groups of services with the same name are running in specific order are included in the configuration file **/etc/insserv.conf**. Thus, for example **\$network** includes a group of services containing networking service. Each service has a starting script included in **/etc/init.d** and these scripts start a service.

Leaving this section empty, our script gets „number 01” - it is added at the beggining of particular starting level (runlevel).

It is important that scripts, which imposes our script, run first. The safest way to do that is typing this in section Required:

```
Required-Start: $all
```

```
Required-Stop: $all
```

Make sure if scripts start with lower number than your script - if higher, assign higher "starting number". Responsible for each run level are catalogs **/etc/rc0.d** , **/etc/rc1.d** , **/etc/rc2.d** , **/etc/rc3.d** , **/etc/rc4.d** , **/etc/rc5.d** , **/etc/rc6.d**. Catalog **/etc/rc0.d** i **/etc/rc6.d** is responsible for stopping and reloading the system. They correspond to runlevel:

```
Runlevel 0 - Stop
Runlevel 1 - Running system as a single-user.
Runlevels 2-5 - Running multi-user.
Runlevel 6 - Reloading the system.
```

If your correctly prepared script is already located in **/etc/init.d/** catalog, run following command first:

To run following command, go to **/etc/init.d/** path

```
update-rc.d moj_skrypt.sh defaults
```

To delete the script run command:

```
update-rc.d -f moj_skrypt.sh remove
```

Example of starting script

A good practice is to equip your starting script with **start**, **stop** and **restart** functions.

```
#!/bin/bash

### BEGIN INIT INFO
# Provides:          script
# Required-Start:    $all
# Required-Stop:     $network
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: short description
# Description:       description
### END INIT INFO

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
start(){
    :::
    :::
}
```

```
}
stop(){
    :::
    :::
}
case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        ;;
    *)
        echo "Usage: $0 {start|stop|restart}"
esac

exit 0
```

Ramdisk package

Ramdisk package installs script which enables ramdisk on /mnt/ramdisk directory. It can be configured by options from /home/core/syscfg.

Installation

To install package, execute one of the following command.

```
softmgr update ramdisk
```

If you want the newest version of package

```
softmgr update ramdisk -b x500cm3-beta
```

Replace *x500cm3-beta* with name of branch for your device.
Refer [Software update](#) for names of branches.

This functionality may be unavailable for your platform.

Restart device after installation.
After that ramdisk will be enabled.

Configuration

Following table shows available options of configuration.
Default option is set during installation.

Option	Explanation	Default value
RAMDISK_ENABLE	Enables or disables creating ramdisk during boot. Available values: Y -for enable, N - for disable	Y
RAMDISK_SIZE	Size of ramdisk. Contains value and unit. For example: 100K, 12M	64M

Starting/stopping ramdisk

Ramdisk is created during boot of the system.
You can stop it during system work with:

```
/etc/init.d/ramdisk_conf.sh stop
```

Then you can start it with:

```
/etc/init.d/ramdisk_conf.sh start
```

Starting ramdisk with command doesn't work if RAMDISK_ENABLE is set to N.

Extended security package

1. Introduction

This document describe extended security package form x500CM3 device.
It adds following settings to device:

- sets ssh port to 320
- blocks root login on ssh
- sets max auth tries to 3
- installs fail2ban package
- configures that every ip address with 3 failed login attempts will be blocked for 24 hours

To achieve some of above functionalities, fail2ban packackage is installed.

2. Installation

To install package, execute following command and reboot the device:

- stable version `softmgr update extended_security`
- the newest, test version `softmgr update extended_security -b x500cm3-test`

3. Configuration

To change default configuration, you need to edit configuration of two servivces:

- SSHD - `/etc/ssh/sshd_config`
- SSHD jail configuration (fail2ban) - `/etc/fail2ban/jail.d/sshd_jail.conf`

If you reinstall package, configuration will be reset to default.
SSHD jail file will be overwritten.

File	Key	Possible values	Description
/etc/ssh/sshd_config	Port	Valid port number	Port for SSHD service
	PermitRootLogin	yes/no	Allows/Disallows root user to login through SSHD
	MaxAuthTries	number	Maximum count of authentication tries

/etc/fail2ban/jail.d/sshd_jail.conf	enabled	true/false	Enables/disables jail configuration
	port	Port of sshd service	Port of service, copy sshd port here
	filter	Service name	Service name for filter. For sshd, put <i>sshd</i>
	logpath	Valid file path	Path to log file
	maxretry	number	Maximum count of authentication tries
	findtime	time in seconds	Time for user to login after first failed login attempt
	bantime	time in seconds	Bantime for client which failed to login in given tries count and time. Service bans client by its IP address

Both services support more parameters.
Refer appropriate documentation to learn more.

[iModCloud package](#)

Introduction

NPE-X500 - as each of our devices can be part of the ecosystem **iModCloud**. X500 equipped with imodcloud package, can be registered in our cloud service **iModCloud**.

iModCloud is a software-service allowing full **iMod** devices control. Together they form a standalone solution – **iModCloud Ecosystem**. In other words - it is a combination of service in cloud with web user interface and industrial devices, fully managable remotely.

Package installation/update

To install or update **imodcloud** package use Software Manager application:

```
softmgr update imodcloud
```

Description

imodcloud package allows to register devices in service **iModCloud**, however to do that, the device must be connected to **VPN** network.

Watchdog VPN

In **imodcloud** package there is watchdog included, allowing **VPN** network connection and monitoring if the connection is correct.

Connecting using VPN

In order to establish connection with VPN network, run following command:

```
/home/imodcloud/scripts/vpn_watchdog --check
```

This action can be also performed to check the connection!

Automatic validation of connection

After running following command, watchdog will be started automatically every minute to check if **VPN** is connected properly, so after any failure of network or loss of certificates, VPN will be automatically renewed.

```
/home/imodcloud/scripts/./vpn_watchdog --on
```

Status check:

```
/home/imodcloud/scripts/./vpn_watchdog --status
```

If **vpn_watchdog** is running, after typing **crontab -l** command in console, you will see following entry in **crona** table:

```
* * * * * /home/imodcloud/scripts/vpn_watchdog --check
```

Shutting down watchdog:

```
/home/imodcloud/scripts/./vpn_watchdog --off
```

Registering devices in iModCloud

Manual of device's registration in iModCloud system:

[Device registration in iModCloud](#)

Troubleshooting

If any problem with connection occurs and the connection won't be re-established after 5 minutes, check **internet connection** first. If the internet connection is not the problem, simply run following command:

```
imodcloud initialize
```

CODESYS

Codesys is a development environment for programming controller applications according to the international industrial standard IEC 61131-3. The main product of the software suite is the CODESYS Development System, an IEC 61131-3 tool.

1. Example configuration

All below examples was tested on CODESYS V3.5 SP16 patch 2. We had to prepare also special Linux version (5.4) because older versions is not working and this is a bug which CODESYS know it and answered that it should be repair.

<https://forge.codesys.com/forge/talk/Runtime/thread/4d43247a3a/#5dae>

- Check stat of the DI and based on it change DO
- Change state of the DO and check DI status
- Modbus RTU - read/write