KOREBOT II

user manual

version 1.6
february 2013

Documentation Author

Julien Tharin
K-Team S.A.
Z.I. Plans-Praz 28
1337 Vallorbe
Switzerland

Email: info@k-team.com
Url : www.k-team.com

Documentation version

Version	Date	Author	Description
1.6	18.02.2013	J.Tharin	Minor corrections, Teraterm doc
1.5	04.04.2012	J.Tharin	Corrected USB, added JavaVM, C++ and Eclipse instructions
1.4	01.12.2011	J.Tharin	Added debug and virtual machine; updated likorebot instructions
1.3	23.06.2011	J. Tharin	Corrected errata of 1.2
1.2	25.09.2009	J. Tharin	Corrected cable color / added detailed instructions about cross-compiler install
1.1	24.07.2009	J. Tharin	Corrected path for template program
1.0	16.07.2009	J. Tharin	First draft

Trademark Acknowledgements:

LEGAL NOTICE:

TABLE OF CONTENTS

1. INTRODUCTION

Thank you for buying the KoreBot II!

With this card, you will be able to create many new embedded systems by interfacing standard devices or enhance the Khepera III robot by expanding its computing power.

1.1 How to use this handbook

This handbook introduces the KoreBot II and its various operating modes. For a quick start, jump to chapter 4 ”***Usage***”.

If this handbook does not answer one of the problems you wish to solve, please consult the K-Team web site (http://www.k-team.com) and especially the Forum and the FAQs.

- **Unpacking and Inspection** **:** KoreBot II package description and first use
- **Description** **:** Korebot II description
- **Usage** **:** KoreBot II usage descriptions.
- **Annexes** **:** Detailed descriptions of several helpful tools and commands are explained.

1.2 Safety precautions

Here are some recommendations on how to correctly use the KoreBot II:

- **Keep the board away from wet area**. Contact with water could cause malfunction and/or breakdown.
- **Store your board in a stable position**. This will avoid the risks of falling, which could break it or cause damage to a person.
- **Do not plug any connectors while the board is powered on**. To avoid any damage, make all connections when the board power is off.
- **Never leave the KoreBot II powered when it is unused**. When you have finished working with KoreBot II, turn it off. It will save the battery life.

1.3 Recycling

Think about the end of life of your product! Parts of the board can be recycled and it is important to do so. By recycling you can help to create a cleaner and safer environment for generations to come. For those reasons please take care to the recycling of your product at the end of its life cycle, for instance sending back the product to the manufacturer or to your local dealer.

Thanks for your contribution to a cleaner environment!

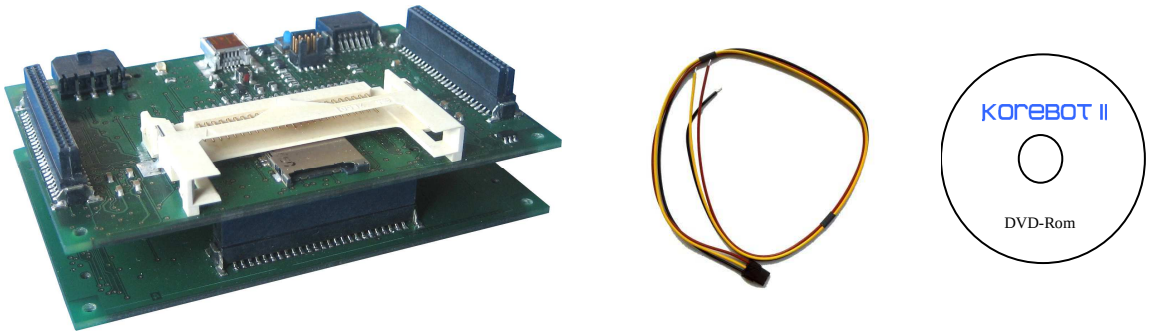
1.4 Specifications

The main specifications of the KoreBot II card are listed below:

- Processor : Marvell PXA270 with XScale @ 600MHz
- Memory : 128MB RAM
  32MB Flash
- Features :
  - USB host signals
  - Compact Flash Connector
  - Micro SD Connector
  - Mini USB connector (for device, Camera, USB key, …)
  - Compatible with KoreBot extensions (I2C)
  - Serial port
- OS : Linux OS, Angström distribution (OpenEmbedded tools), kernel 2.6.24
- Power consumption : 100 mA @ 5 V (without any "Kore" extension)
- Size : 85 width x 57 depth x 33 height [mm]
- Mass : 52 g

2. UNPACKING AND INSPECTION

2.1 Package Contents



(1) (2) (3)

Figure 2-1: Contents of the KoreBot Pack

Your package should contain the following items:

1. Korebot II card
2. Power supply cable
3. DVD-Rom with software

2.2 Inspection

The KoreBot board basic functions should be tested after unpacking. A Complete Linux system is installed in the KoreBot flash memory. The system can be started as a standalone Linux box, with the initial console displayed on the serial line. No application is started except the initial shell.

The power supply and the serial connection only are necessary for the system check. The power supply must be connected as described in section 4.2 "***Connections***". The serial link should be connected from a host computer serial port to the KoreBot board, either using a KoreConnect extension or using a custom cable (please refer to section 4.2 "***Connections***" for connection details).

The serial port should be linked to a terminal, such as ***minicom***, on the host computer. The basic configuration for the serial line should be:

- 115200 Bps
- 8 data bits
- No parity
- 2 stop bits

Then when switching power on, the Linux boot messages should be displayed on the terminal (see chapter 4.3: "***How to use this handbook for details***"). If no character is received, the serial line and terminal settings should be checked. If the boot is interrupted at some point, especially during kernel uncompress step, then the system is probably corrupted. Chapter 4.4.2.2: "***Uploading the kernel and the file system: kt-boot usage***" describes how to upload a new system to the board. If the boot process is completed to the login prompt, then the boot messages should be checked for errors, please refer to chapter 3.3: "***Korebot Software***" for further information on the KoreBot Linux system.

3. Description

3.1 Overview

An overview of the KoreBot hardware is depicted in the Figure 3-1 and Figure 3-2. The locations of various key elements are indicated for later references. The compact flash slot can receive a Type I card.

The KB-250 extension bus connectors are used to plug in KoreBot extensions. Please visit http://www.k-team.com for a list of available extensions. All the other connectors are described in the subsequent part of this chapter.

The USB Host connector is a Mini-AB USB On-The-Go connector. USB peripherals can be connected to KoreBot with this connector.

Warnings:

The KoreBot power supply system is not designed to support Compact Flash card hot plug. Inserting or removing a card when the power is switched on can cause critical damage to the Compact Flash electronic interface.

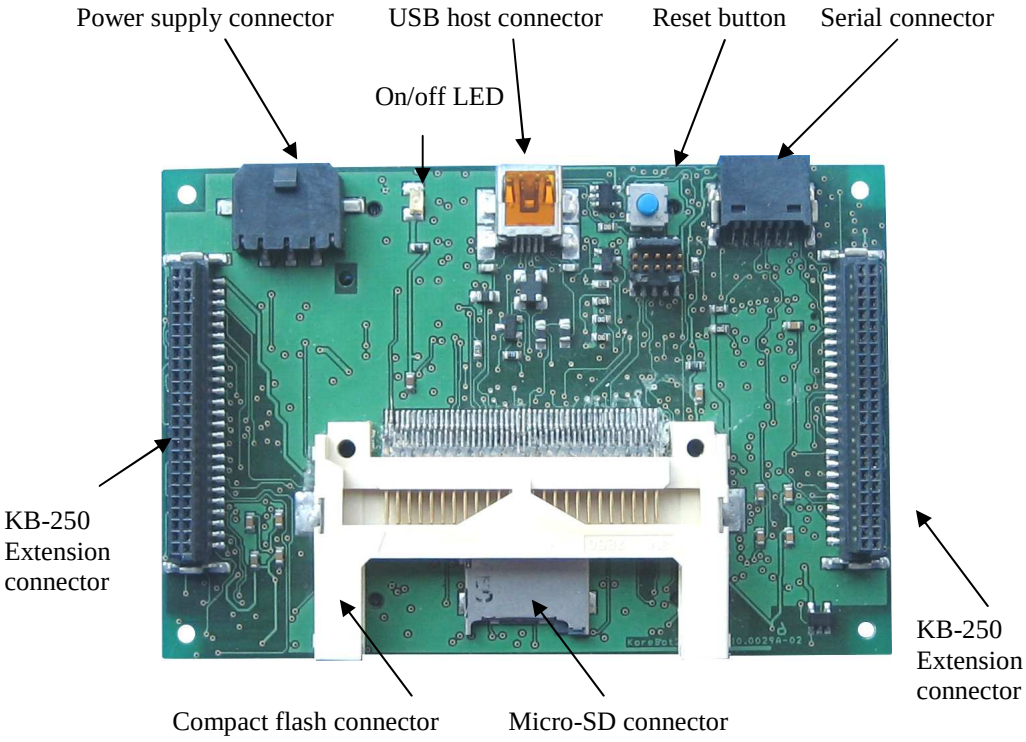


Figure 3-1: Korebot II overview – bottom view

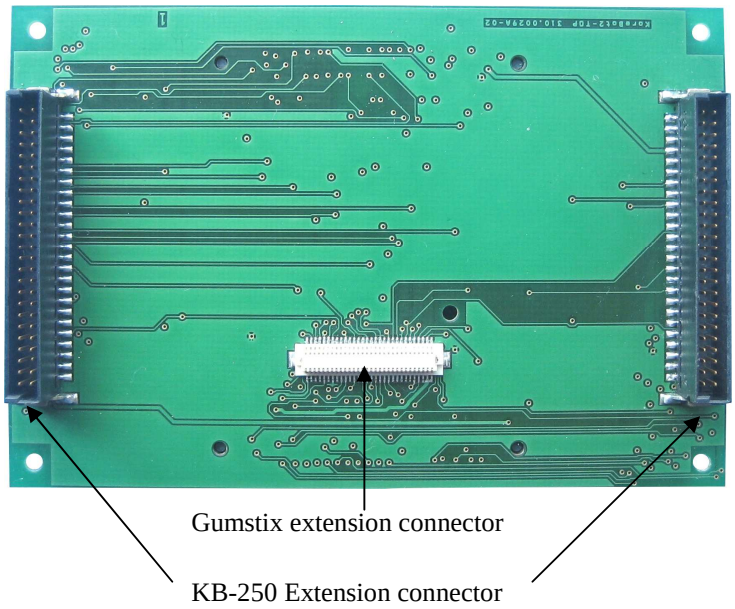


Figure 3-2: Korebot II overview – top view

3.2 Korebot Hardware

The hardware of the Korebot II is described in the sub-chapters thereafter.

3.2.1 Power Regulation

The KoreBot is designed so that unregulated power sources, such as batteries, can be directly connected to supply the board without any external regulation required. One embedded regulators is provided to enable any voltage source between 3.3V and 5.5V. Please notice that Wrong power connections can cause critical damage to the board.

A ground reference is common for the entire board. The power source must be connected to the correct input to supply the board. When using the high voltage input, the first regulation stage provides regulated 5V and supplies the second regulation stage. The second regulation stage, which cannot handle input voltage over 5.5V, provides regulated 3.3V and the processor core supply. When using the low voltage input, it is directly connected to the second regulation stage. The board will be fully functional as long as the input is over 3.3V. The CPU core can still run with an even lower input, but the 3.3V supply will fall and the board peripherals will stop functioning.

A low voltage supply connection is displayed on Figure 3-3. Only two of the three power connections are used at the same time. The power cable included with the KoreBot package has three connections be careful to connect the board power properly to avoid damage.

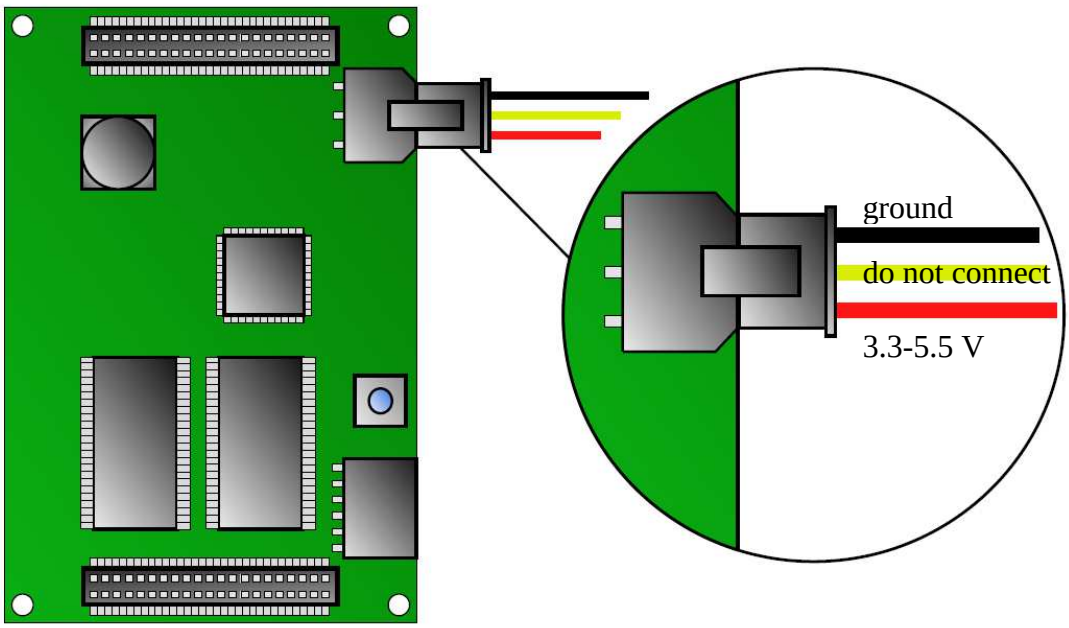


Figure 3-3: KoreBot voltage power connection

3.2.2 KoreBot Serial and USB Slave Connector

The KoreBot serial connector provides signals for several interfaces. The board itself is too small to include standard serial or USB connectors, but it can be easily extended using a KoreConnect extension or with any setup suitable for a custom application.

The serial connector provides two serial interfaces, one USB slave and a regulated 3.3V power connection. Figure 3-4 shows details for the pins function.

The regulated 3.3V power pin can supply external electronic devices such as a RS2332 transceiver. KoreBot regular extensions should use the KB-250power pins as a supply source. Please refer to the Electrical Specification for further details.

The third serial port (Bluetooth UART) is only available on the KB-250 extension bus. Additional signals, such as RS232 control signals are also provided on this bus. Special applications may need to use the KB-250 connection for special requirements, please refer to the "***Intel PXA270 Developer Manual***" for a detailed description of the various serial ports features and capabilities.

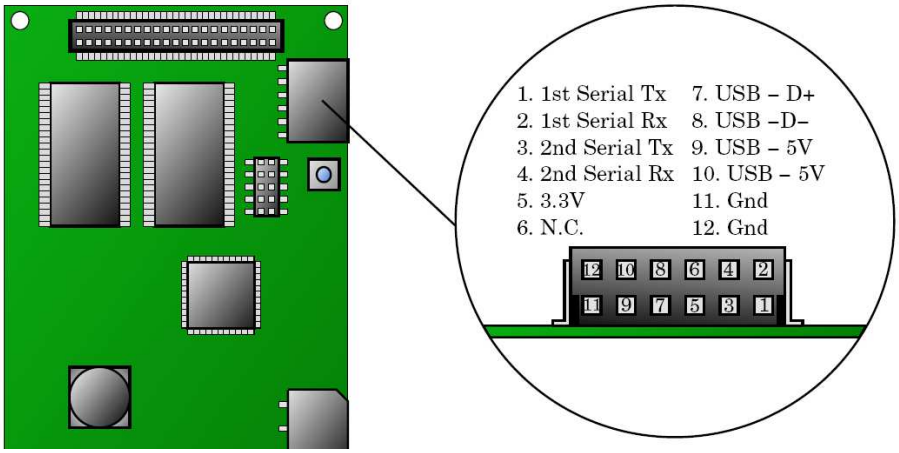


Figure 3-4: KoreBot serial connector

When the KoreBot is connected without using a KoreConnect extension, a flat cable is provided to create a custom cable connection. The flat cable pin functions are described on Figure **3-5**. Special care should be taken for USB connections, as these signals are quite sensitive.

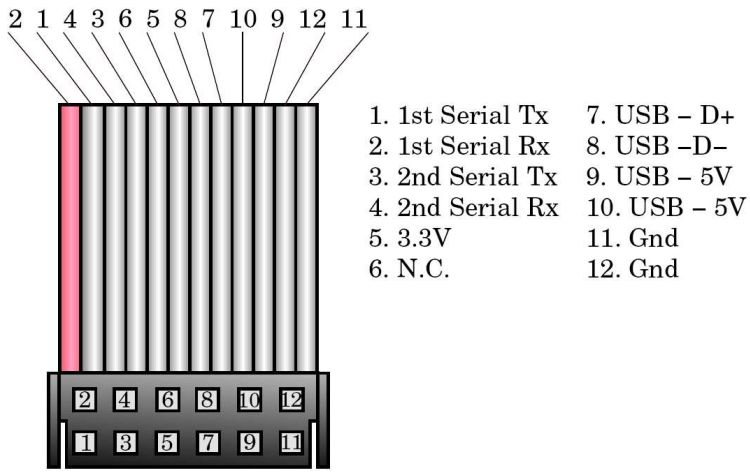


Figure 3-5: Korebot flat cable for serial connection

3.2.2.1 Serial connection

Signals from the main serial interface (Full UART) are RS232 compatible signals, a serial cable from a PC can be directly connected without any other requirement. On the other hand, signals from the second serial interface (IrdDa UART) are TTL signals and an external RS232 transceiver is required for a communication with a host computer. When using a KoreConnect extension, the transceiver is provided within the extension, so that both serial connectors are RS232 compatible.

Figure 3-6 shows how to connect a DB9 connector to the KoreBot serial connector. This connection to the first serial line will allow an easy link to a Personal Computer and a virtual terminal. This is by default the channel used for the Linux initial console and it is required for the fisrt interactions with the system.

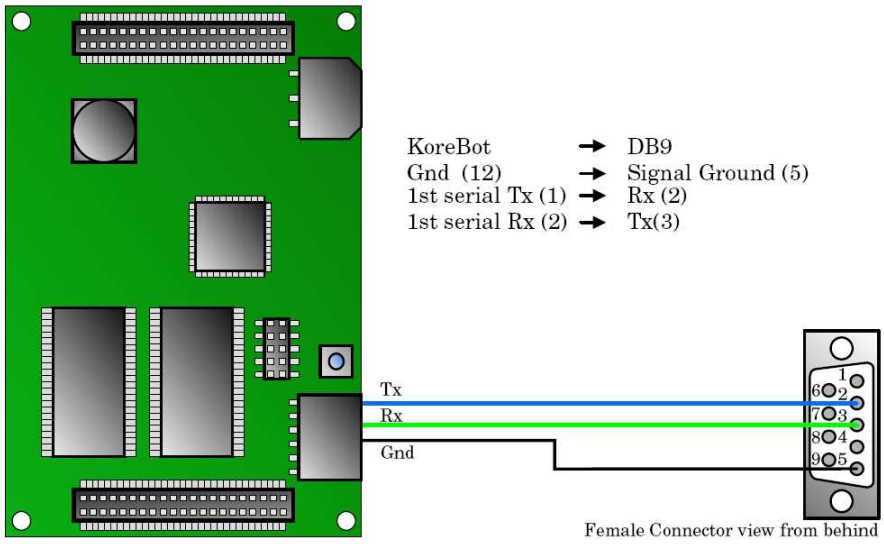


Figure 3-6: DB9 connection

3.2.2.2 USB Slave Connection

Up to now, the PXA270 CPU doesn't support USB slave correctly. Then it is not implemented here.

3.2.2.3 KoreConnect Extension

The KoreConnect extension provides a simple interface to standard cables and connectors from a personal computer to the KoreBot. Two standard DB9 serial connections and a standard USB type B connection are provided to interface the first two KoreBot serial ports and the KoreBot USB slave port. Furthermore, a RS232 transceiver will convert TTL signals from the second serial port to high voltage signal required for a direct connection to a PC.

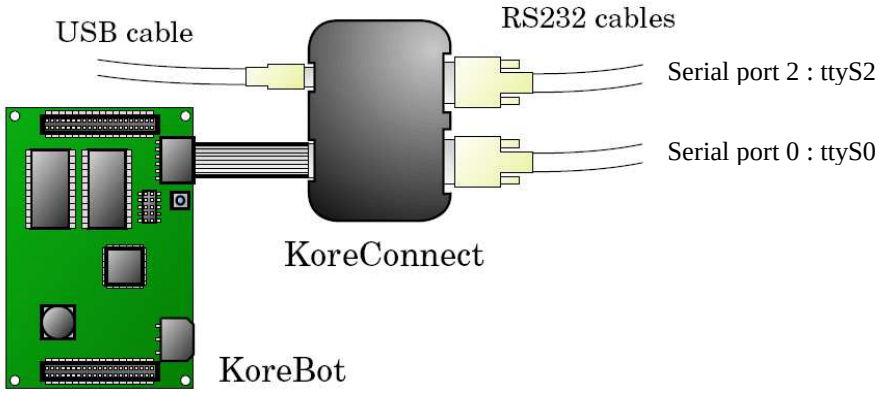


Figure 3-7: KoreConnect extension

KoreConnect is not included with the Korebot standard package and needs to be purchased separately. Figure **3-8** displays how to connect the KoreBot to a PC to get the initial boot console from the serial port 0.

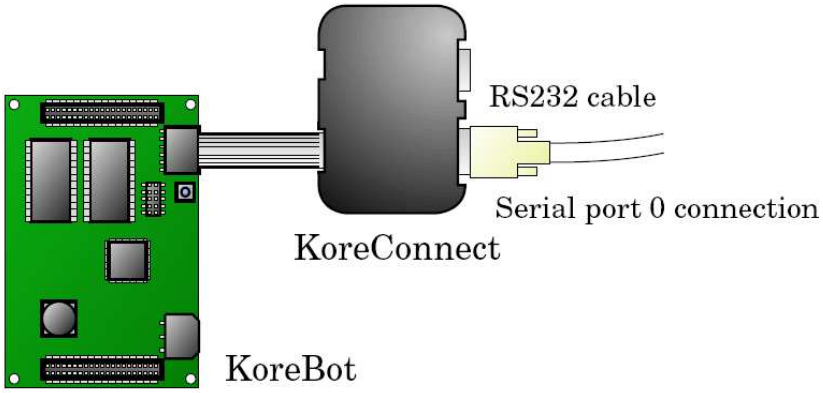


Figure 3-8: Serial Port 0 Connection

3.2.3 KoreBot Reset

The KoreBot reset chain can be triggered by pressing the reset button (see Figure **3-1**, Figure 3-9).

When pressing the reset button, the board core supply and 3.3V regulator is actually shut down and the Pxa-Reset pin is held low, ensuring a proper restart from safe state. The Pxa-Reset-Out pin is also held low by the CPU, so that all the other devices on the board are reset. On button release, the regulator itself manages to wait until the power regulation is stable before releasing the Pxa-Reset pin high. Then the Pxa complete its reset sequence before releasing the Pxa-Reset-Out pin that is connected to all the other resettable devices on the board.

When driving the Pxa-Reset pin low from an external device, the processor is reset, and it drives the Pxa-Reset-Out pin low to reset other devices. The Pxa-Reset-Out pin is not released until the processor reset sequence is completed. The only difference between the two methods is that the regulator is not shut down.

3.2.4 KoreBot JTAG Connection

The JTAG connector is a direct connection to the Pxa270 JTAG debug port. This interface enables online debugging and memory access from a host computer using the KoreJTAG extension or any compatible JTAG hardware. Figure 3-9 is a pin description for the JTAG connector.

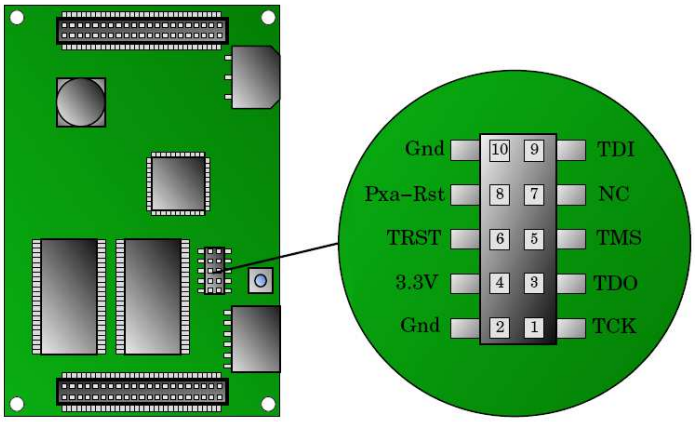


Figure 3-9: JTAG connector

3.2.5 KoreBot USB Host

The KoreBot provides one USB host. It is available from the mini-USB connector (see fig 3-9). The USB Host interface provides a way to connect USB devices, such as webcams or external disks, to the KoreBot. Plug the device when the Korebot is on and it will be detected and mounted for storage devices. See with ***mount*** command.

Example: mounted on directory ***/media/hdd*** for USB key. Unmount it with command ***umount /media/hdd*** before unplugging the storage device.

3.2.6 Korebot Compact flash connector

You can connect any compact flash, Type I card to this connector.

WARNING : The system does not support hot-plug. Switch off the Korebot before adding or removing any Compact Flash card!

The µ-SD and the Compact flash are on the same bus. Then you cannot use both together.

For using a Compact flash memory card (shipped in that mode by default):

1) Comment the ***mmc_block*** and ***pxamci*** modules
2) Uncomment by adding # at the beginning of the line or add if not present the ***pcmcia*** and ***pxa2xx_cs*** modules
3) Switch off the Korebot; insert the Compact flash card; and switch on.
4) Typing ***mount*** shows the mounted card location:

/dev/hda on /media/hda type vfat (rw,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-1)

=> You can copy files to and from the card at the location ***/media/hda***

See chapter 5.2.2: "***Using a Wireless compact flash card***" for wireless compact flash settings.

3.2.7 Korebot µ-SD connector

With the µ-SD connector and a µ-SD card, you can easily transfer files to and from the Korebot.

WARNING: The µ-SD and the Compact flash are on the same bus. Then you cannot use both together.

By default the Compact flash is activated. To change the activated card, you have to edit the ***/etc/modules*** file as follow:

1) Comment the ***pcmcia*** and ***pxa2xx_cs*** module
2) Uncomment by adding # at the beginning of the line or add if not present the ***mmc_block*** and ***pxamci***
3) Typing ***mount*** shows the mounted card location:

/dev/mmcblk0p1 on /media/card type vfat (rw,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-1)

=> You can copy files to and from the card at the location ***/media/card***

3.3 Korebot Software

A KoreBot embedded system is based on two main software components. One is the Linux Operating System kernel, and the second is a set of software packages that is called a distribution.

The Korebot II uses the Linux distribution called "Angström OpenEmbedded".

The Linux kernel is based on standard Linux kernel sources, with adaptation made by Gumstix. Further, exhaustive, information about Linux kernel is available from many sources, especially on the web, starting from http://www.kernel.org. The installed distribution is based on the handhelds familiar distribution, which is specially designed for embedded systems. Please visit the "Angström" website http://www.angstrom-distribution.org, for further information about this project. The main components within this distribution are described in the following sections.

Using the KoreBot Linux system should be pretty straightforward for users with a Linux or Unix background considering that all components are standard.

In the next chapter 4 "***Usage***
", the installation and usage of the KoreBot software is described. And in the Annexes, tools for customizing, configuring and using specific tools are detailed.

4. Usage

4.1 Required hardware / software

The required hardware and software to use the board and develop programs are described below.

4.1.1 Required hardware:

- Computer with serial (or USB) port (not included)
- Serial cable or serial-to-usb adapter cable (not included)
- KoreConnect (not included), KoreConnect cable (not included)
- 5V DC power supply (not included)
- Optional extensions:
  - KoreBase : base board
  - KoreWifi : wireless module
  - KoreUSBCam : camera module
  - Khepera 3 : robot

4.1.2 Required software:

Required free space:

- 150 MB on ***/usr/local*** *for light toolchain
- 16 MB on user account (~/)

Required files:

- Linux OS (kernel 2.6.x) on the computer with the following packages installed:
  - ***gcc*** : GNU C compiler
  - ***minicom*** : terminal emulation
  - ***lrzsz*** : communication package
  - ***picocom*** : minimal dumb-terminal emulation program
  - ***expect*** : interactive scripts running program

- Included in the DVD-ROM of the package:
  - Cross-compiler light : ***korebot2-oetools-light-1.0-kb1.tar.bz2***
  - Development folder : ***development_k2_v1.0.tar.bz2***
  - Board library sources : ***libkorebot-1.19-kb1.tar.bz2***
  - Script for uploading Kernel and file system : ***ktboot-2.0_20090416.tar.bz2*** [1]

Remark: you may find updated version of these software at http://ftp.k-team.com

[1] 20090416 means the version of this software.

4.2 Connections

The basic connections of the board are depicted in Figure **4-1**. The power supply is connected to the board with the power cable. And the serial cable is connected to the computer through the KoreConnect adapter. If there is no serial port, you may use an usb to serial adapter.

A low voltage supply connection is displayed on. Only two of the three power connections are used at the same time. The power cable included with the KoreBot package has three wires with bare ends:

- Red : DC 3.3-5.5 V
- Black : ground
- Yellow : not used

You should be careful when connecting the board power properly to avoid damage.

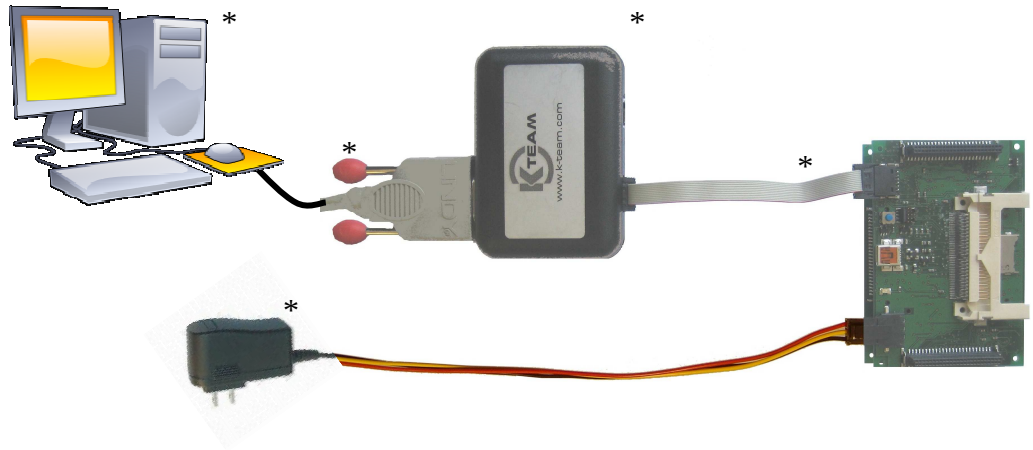


Figure 4-1: Basic connections

* Serial cable, KoreConnect, KoreConnect cable, power supply and computer are obviously not included.

4.3 Power-up

Follow the instructions below to connect to the Korebot with the serial port.

1) Install the Linux package **lrzsz** containing communications programs. If your Linux distribution is Ubuntu:

   sudo apt-get install lrzsz

2) On the Linux computer, run the emulation terminal Minicom:

   minicom

If Minicom is not installed you have to install this package. For Linux Distribution "Ubuntu", the command is:

sudo apt-get install minicom

3) Set its parameters with the sub-menu "Serial port" setup of the menu [configuration] (keys Ctrl-a + o) as described in Figure **4-2.**

You may modify the serial port device name depending where you plugged your serial port (see chapter 5.2.1 "***Using serial port and Minicom***").

If you use a serial to USB adapter, the serial device may be ***/dev/ttyUSB0***.

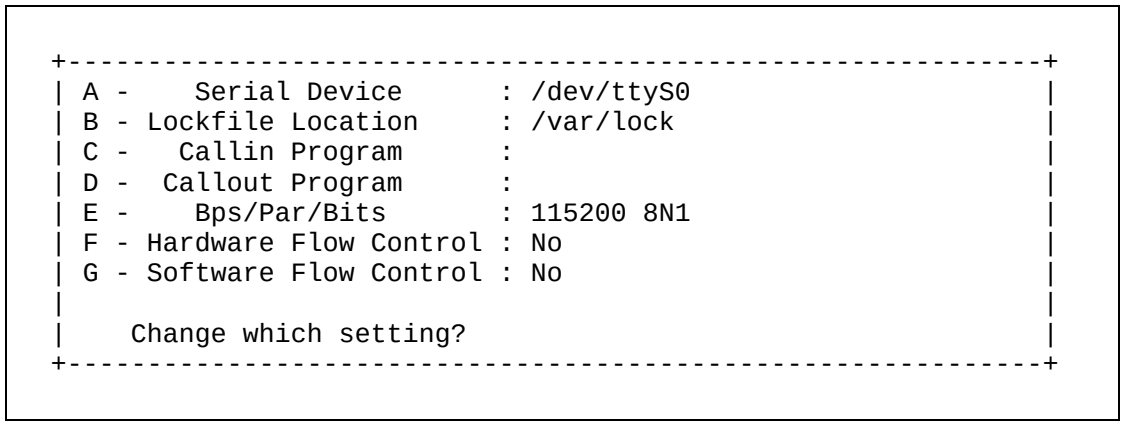


Figure 4-2: Minicom serial parameters

Save the settings with the command "***Save setup as dfl***" of the menu [configuration] (cf Figure 4-3).

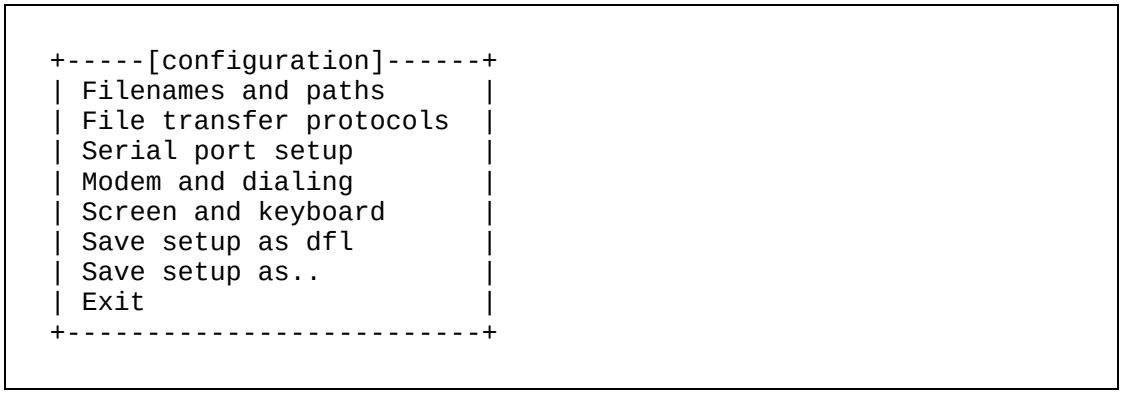


Figure 4-3: Minicom configuration menu

4) Connect the Korebot as described in chapter 4.2. And connect the power supply to the Korebot and switch it on. The Korebot will boot and the prompt will be available: see Figure 4-4.

```
U-Boot 1.2.0 (Dec 21 2007 - 13:37:16) - PXA270@600 MHz - 1578M
*** Welcome to Gumstix ***
DRAM:  128 MB
Flash: 32 MB
Using default environment

Hit any key to stop autoboot:  0
Instruction Cache is ON
Copying   kernel   to   0xa2000000   from   0x01f00000   (length
0x00100000)...done
## Booting image at a2000000 ...
   Image Name:   Angstrom/2.6.24/gumstix-custom-v
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    1000672 Bytes = 977.2 kB
   Load Address: a0008000
   Entry Point:  a0008000
OK

Starting kernel ...

Linux version 2.6.24 (jtharin@KHEPERA04) (gcc version 4.1.2) #1
Wed Jan 28 00:11:22 CET 2009
CPU: XScale-PXA270 [69054117] revision 7 (ARMv5TE), cr=0000397f

Machine: The Gumstix Platform

.
.
.


OpenEmbedded Linux korebot2 ttyS0

Angstrom 2007.9-test-20090127 korebot2 ttyS0

korebot2 login:
```

Figure 4-4: part of the Korebot Boot log

5) Login to the Korebot with the following parameters:

Login: ***root***

Password: ***(none, press “Return” key)***

=> You are at the prompt of the Linux console of the Korebot (see Figure 4-5).

```
Angstrom 2007.9-test-20090127 korebot2 ttyS0

korebot2 login: root
Password:

Welcome to Korebot2!

root@korebot2:~$
```

Figure 4-5: Korebot prompt

4.4 Software

The following sub-chapters explain the software installation and the application development with the board.

Two development packages are available:

- Light toolchain
- Full toolchain and sources: for advanced users; kernel modification; packages creation/addition

In the subsequent paragraphs, only the light tool chain is explained. The full toolchain is described in a following chapter.

4.4.1 Installation of light toolchain

The installation of the software required to use the board and the development tool is described in the next sub-chapters.

Below are listed the files needed to install the light toolchain. These files are included in the DVD-ROM of the package or from the Internet url: http://ftp.k-team.com/KorebotII/software/

From ***light_toolchain/*** folder:

- Cross-compiler light : ***korebot2-oetools-light-1.0-kb1.2.tar.bz2***

From ***common_files/*** folder:

- Development folder : ***development_k2_v1.0.tar.bz2***
- Board library sources : ***libkorebot-1.19-kb1.tar.bz2***
- Library package : ***libkorebot_1.19-r0_armv5te.ipk***
- Script for uploading kernel and file system : ***ktboot-2.1_20111025.tar.bz2***

4.4.1.1 Installation of the development directory

The development directory will be the base folder for your development. It contains links and scripts to easily use the cross-compiler to make your programs.

1) Extract the file **development_k2_v1.0.tar.bz2** in your home directory and enter in the directory ***development_k2_v1.0*** created just before, which will be your development directory. You can use the following commands, assuming you are in a console, which current directory contains the file:

   tar -xjf development_k2_v1.0.tar.bz2 -C ~/

   cd ~/development_k2_v1.0

2) Modify the **KTEAM_HOME** variable in the file **env.sh** to point to your development directory: replace **YOUR_USERNAME** by your Linux account name. You can use ***vi*** text editor (see chapter 5.2.7 "***Using vi text file editor***"):

   KTEAM_HOME=/home/YOUR_USERNAME/development_k2_v1.0

4.4.1.2 Installation of the cross-compiler (light toolchain)

1) Extract the cross compiler **korebot2-oetools-light-1.0-kb1.2.tar.bz2** in /usr/local with the command:

   sudo tar -xjf korebot2-oetools-light-1.0-kb1.2.tar.bz2 -C /usr/local

Remark: you must be root or use ***sudo***

2) And create a symlink using this command in your development directory:

   ln -s /usr/local/korebot2-oetools-1.0/tmp/cross ~/development_k2_v1.0/cross

3) You can check if the installation is correct by running the cross-compiler. Firstly make the environment variables available then check the version of the cross-compiler:

   source env.sh

   arm-angstrom-linux-gnueabi-gcc --version

=> The last command should return:

arm-angstrom-linux-gnueabi-gcc (GCC) 4.1.2

…

4.4.1.3 Installation of the board library libkorebot

The library is already installed on the Korebot. See chapter 4.4.1.5 for a reinstall or update. To install the library on your development system, follow the following instructions:

1) Extract the library **libkorebot-1.19-kb1.tar.bz2** in your development folder, where ***1.19-kb1*** is the current version:

   tar -xjf libkorebot-1.19-kb1.tar.bz2 -C ~/development_k2_v1.0

2) Check in the file env.sh of your development folder the variable ***LIBKOREBOT_ROOT*** to update the version of the libkorebot if it is different of the one installed above.

3) You can recompile the whole library by running the following commands in the **libkorebot-1.19-kb1** folder:

   source ../env.sh

   make clean

   make

On the Korebot II, there are 3 ways to check the version of the installed libkorebot (only the last one works if the library version is <1.17):

- Command: ***ipkg list_installed | grep libkorebot***
- Command: l***s –l /usr/local/libkorebot****
- Transfer to the Korebot II the ***template*** program from the ***libkorebot/template*** (chapter 4.4.2.1) and execute it with the parameters: ***./template --kb-version 0***

  For libkorebot version >= 1.17: ***./template --version*** or ***./template -v***

You can find an updated version of the ***libkorebot*** from the following ftp site:

http://ftp.k-team.com/KorebotII/software/common/libkorebot/

4.4.1.4 Installation of the scripts for uploading the kernel and the file system: ktboot

With these scripts, the Korebot can be reinitialised to its default settings.

1) Extract the file **ktboot-2.1_20111025.tar.bz2** in your development folder:

   tar -xjf ktboot-2.1_20111025.tar.bz2 -C ~/development_k2_v1.0

=> For usage, see chapter 4.4.2.2: ”***Uploading the kernel and the file system: kt-boot usage***”.

4.4.1.5 Installation/ update of the libkorebot on the Korebot

The library is already installed on the Korebot. For an update, download the new version at the site below and transfer it to the Korebot:

http://ftp.k-team.com/KorebotII/software/common/libkorebot/

Then run: ***ipk install libkorebot_A.B-r0_armv5te.ipk***

where ***A.B*** is the libkorebot version.

If you customized the libkorebot, you can install it manually:

- Transfer your new ***libkorebot_A.B.so*** that is from your ***dev_folder/libkorebotA_B/build-korebot-2.6/lib*** to the directory ***/usr/lib*** of the Korebot II.
- Delete the link and recreate it:

  rm /usr/lib/libkorebot.so

  ln -s /usr/lib/libkorebot_A.B.so /usr/lib/libkorebot.so

4.4.2 Programming and light toolchain usage

4.4.2.1 C Programming: application development

A template program **prog-template.c** is available in the board library **libkorebot** in the folder **libkorebot-1.19-kb1/template**.

You can start your code into the template program and use the following commands to build it. The first one makes the environment variables (path to the cross-compiler, libraries) available to the system and the second run the **Makefile** script to compile and build the executable program. Enter in the **libkorebot-1.19-kb1/template** folder and type in a console to build the template program:

source ../../env.sh *******

make

=> The "**template**" file is the executable output file.

You can transfer the program to the Korebot by serial (see chapter 5.2.1 "***Using serial port and Minicom***") or wireless connection (see chapters 5.2.2 and 5.2.3).

Then execute it by running:

./template

The Application Programming Interface (API) documentation of the library is available in the ***libkorebot*** directory under ***doc/html/index.html*** or there:

http://ftp.k-team.com/KorebotII/software/common/libkorebot/api/html/index.html

You can find programming examples in the sub-directory ***src/tests*** of the ***libkorebot*** library.

You can debug your program with the instructions in the chapter 5.2.8.

For C++ programming, see sub-chapter 4.4.2.4.

For programming using a GUI such as Eclipse, see sub-chapter 4.4.2.5.

Remarks:

If you modify the program name, you will have to modify its occurrences in the **Makefile** file.

*** You need to source the env.sh file every time you open a new console** when you want to program, to give access to the cross-compiler.

4.4.2.2 Uploading the kernel and the file system: kt-boot usage

The kt-boot allows to upload to the Korebot the kernel and file system if it was erased/corrupted or if a new is available or compiled with the full toolchain (see chapter 4.5, "***Full toolchain and sources***").

WARNING: All the data on the Korebot will be lost after running the following instructions!

Remarks: If the **full toolchain** is installed, the kernel and file system will be taken from the toolchain directory by default and no more from the **kt-boot** directory! See the ***Makefile*** file in the ***kt-boot*** directory for details.

Instructions for uploading the kernel and the file system:

1) Install the ***picocom*** terminal and the scripts program ***expect***

   sudo apt-get install picocom expect

2) Edit the script ***picocom*** in your development directory to match your serial port connection (***/dev/ttyS0*** by default) and run it:

   ./picocom

3) Reboot the KoreBot, then press a key when U-Boot asks for autoboot (see Figure 4-6).

```
U-Boot 1.2.0 (Dec 21 2007 - 13:37:16) - PXA270@600 MHz - 1578M

*** Welcome to Gumstix ***

DRAM:  128 MB
Flash: 32 MB
Using default environment

Hit any key to stop autoboot:  0
GUM>
```

Figure 4-6: U-Boot

4) Type Ctrl-y Ctrl-x to exit **picocom**.

5) Then change directory to **ktboot**, edit the Makefile (with ***vi*** editor (see chapter 5.2.7 "***Using vi text file editor***") and modify the variable to match your serial port:

   SDEV = /dev/ttyS0

6) type to flash the kernel (~2min):

   make flashk

7) flash the image file system (~30min), type :

   make flashd

8) configure the kernel boot parameters (~10s), type:

   make bootargs

9) reboot the Korebot.

4.4.2.3 Wifi usage

See chapter 5.2.2 : "***Using a Wireless compact flash card***" for the description and usage.

4.4.2.4 C++ programming

The C++ cross-compiler ***arm-angstrom-linux-gnueabi-g++*** is automatically installed with the light toolchain (chapter 4.4.1). If you have already the Makefile of your program, replace inside ***g++*** by ***arm-angstrom-linux-gnueabi-g++***.

By default, only the standard C library (libc) is installed on the Korebot II. If you want use C++ compiled program, you will have to install the C++ library (libstdc++). It is available on the DVD or here as a package:

http://ftp.k-team.com/KorebotII/software/common/libstdc++6_4.1.2-r10_armv5te.ipk

Transfer this file to the Korebot and install it with the command:

ipkg install libstdc++6_4.1.2-r10_armv5te.ipk

4.4.2.5 C/C++ Programming: using GUI source code editor Eclipse

You can use also Eclipse as source code editor. See instructions below how to install and use it.

1) Install the Java Runtime Environment (JRE) if not already installed.

   Find the version corresponding to your OS here:
   http://www.eclipse.org/downloads/moreinfo/jre.php

   and the JRE file there: http://java.com/en/download/index.jsp

2) Download the linux version of "Eclipse IDE for C/C++ Developers (includes Incubating components)" from (for Ubuntu users, don't install with apt-get because, the apt-get version is older):

   http://www.eclipse.org/cdt/

3) Extract the Eclipse program file:

   mkdir ~/bin

   tar -xzf eclipse-cpp-indigo-SR2-incubation-linux-gtk.tar.gz –C ~/bin

   You can also create a link to the start file: ***sudo ln -s ~/eclipse/eclipse /usr/bin/eclipse***

4) You should have installed the latest version of the Korebot 2 toolchain (light or full). Copy the debugger program file:
http://ftp.k-team.com/KorebotII/software/common/debug/arm-angstrom-linux-gnueabi-gdb

   to ***/usr/local/korebot2-oetools-1.0/tmp/cross/bin*** on your computer with the command (one line):

   wget http://ftp.k-team.com/KorebotII/software/common/debug/arm-angstrom-linux-gnueabi-gdb -P /usr/local/korebot2-oetools-1.0/tmp/cross/bin

   You can check by opening a terminal and sourcing the Korebot 2 environement with "source env.sh" (see Korebot " User Manual", chapter 4.4.2.1, command

   arm-angstrom-linux-gnueabi-gdb --version

   which should return: ***GNU gdb (GDB) 7.0***

5) Run eclipse and at the "Workspace launcher" window, choose where you would like to put your project.

6) Go to file menu “File => C Project” (or C++; for running a C++ on the Korebot 2, you must install the standard C++ library:

   http://ftp.k-team.com/KorebotII/software/common/libstdc++6_4.1.2-r10_armv5te.ipk
   with the command ***ipkg install libstdc++6_4.1.2-r10_armv5te.ipk*** on the Korebot 2) and choose a Project Name (ex: test).

7) In the next window "C Project", push the "Advanced Settings" button and on the “C/C++ Build => Discovery Options”,

for c: On 'GCC C Compiler' change '***Compiler invocation command***' to '***/usr/local/korebot2-oetools-1.0/tmp/cross/bin/arm-angstrom-linux-gnueabi-gcc'***

or (depending if your project is C or C++)

for c++: On 'GCC C++ Compiler' change '***Compiler invocation command***' to '***/usr/local/korebot2-oetools-1.0/tmp/cross/bin/arm-angstrom-linux-gnueabi-g++***'

8) On C/C++ “Build => Settings”, “Cross GCC Compiler” change '***Command***' to '***/usr/local/korebot2-oetools-1.0/tmp/cross/bin/arm-angstrom-linux-gnueabi***-gcc'

9) On “Cross GCC Compiler” Includes => Include paths

add ***/usr/local/korebot2-oetools-1.0/tmp/staging/arm-angstrom-linux-gnueabi/include***

10) On Miscellaneous, replace "Other flags" with "***-c -march=armv5te -mtune=xscale -Wa,-mcpu=xscale***"

11) On the Cross GCC Linker, change '***Command***' to '***/usr/local/korebot2-oetools-1.0/tmp/cross/bin/arm-angstrom-linux-gnueabi-gcc'***

12) On “Cross GCC Linker => Libraries” at "Libraries (-l), add ***korebot*** with the + button on the upper right.

13) At "Libraries search path", add '***/usr/local/korebot2-oetools-1.0/tmp/staging/arm-angstrom-linux-gnueabi/lib***'

14) On "Cross GCC Assembler", in Command field, modify to : ***/usr/local/korebot2-oetools-1.0/tmp/cross/bin/arm-angstrom-linux-gnueabi-as***

15) Choose “C/C++” Build menu at the left, “Discovery options” repeat the steps above from 7) for the Release configuration.

16) push “Finish” button.

17) Go to menu “File => New => Source file” choose ***test.c*** as filename

18) Close the Welcome window with its cross at the upper left.

19) Insert in the ***test.c*** file the following C source code:

```
#include <korebot/korebot.h>

     int main(int argc, char *argv[]) {
     int rc;

     /* Set the libkorebot debug level - Highly recommended for
development. */
     kb_set_debug_level(2);
     printf("LibKorebot Template Program\r\n");

     /* Init the korebot library */
     if((rc = kb_init( argc , argv )) < 0 )
           return 1;

     /* ADD YOUR CODE HERE */

     return 0;
     }
```

20) Then cross-compile the project with menu “Menu Project => Build-All”

=> The output file will be in the subdirectory Debug or Release of the project.

21) Transfer the file "test" to your Korebot 2 (see chapter 5.2.3).

22) And execute the program file with command: ***./test***

4.5 Full toolchain and sources

The full toolchain is for advanced users who would like to modify the kernel, rebuild the image system or develop new packages for the Korebot.

Remarks: Prior knowledge of Linux, its kernel and Open Embedded tools is highly recommended.

4.5.1 Required software:

Required free space:

- 5.5 GB on ***/usr/local***
  (10 GB for the installation process, including temporary files)
- 16 MB on user account (~/)

Required files:

- Linux packages
  - ***g++*** : GNU C++ compiler
  - ***patch*** : patch software
  - ***help2man*** : manual converter
  - ***diffstat*** : reads the output of ***diff*** and displays a histogram
  - ***texi2html*** : convert to html
  - ***makeinfo*** : produce doc (***texinfo*** on ***Ubuntu***)
  - ***ncurses-dev*** : library allowing the programmer to write (***libncurses5-dev*** on ***Ubuntu***)
  - ***cvs*** : revision control system
  - ***gawk*** : programming language designed for processing text-based data
  - ***python-dev*** : dynamic object-oriented programming language
  - ***python-pysqlite2*** : Python sql interface
  - ***subversion*** : version control system

On ***Ubuntu*** Linux distribution, you can use the following command to install all the above packages in one time:

sudo apt-get install g++ patch help2man diffstat texi2html texinfo libncurses5-dev cvs gawk python-dev python-pysqlite2 subversion

- Included in the DVD-ROM of the package:
  - Cross-compiler and "Open Embedded" tools sources : ***korebot2-oetools-1.0-kb1.2.tar.bz2***
  - Development folder : ***development_k2_v1.0.tar.bz2***
  - Board library sources : ***libkorebot-1.19-kb1.tar.bz2***
  - Script for uploading kernel and file system : ***ktboot-2.1_20111025.tar.bz2***

4.5.2 Installation

1) Install the development directory as explained in chapter 4.4.1.1: "***Installation of the development directory***" if not already done.

2) Extract the cross compiler sources ***korebot2-oetools-1.0-kb1.2.tar.bz2*** in ***/usr/local*** with the command:

   sudo tar -xjf korebot2-oetools-1.0-kb1.2.tar.bz2 -C /usr/local

3) And create a symlink using this command in your development directory:

   ln -s /usr/local/korebot2-oetools-1.0/tmp/cross ~/development_k2_v1.0/cross

4) You can check if the installation is correct by running the cross-compiler. Firstly make the environment variables available then check the version of the cross-compiler:

   source env.sh

   arm-angstrom-linux-gnueabi-gcc -version

=> The last command should return:

arm-angstrom-linux-gnueabi-gcc (GCC) 4.1.2

5) Install the board library ***Libkorebot*** as explained in chapter 4.4.1.3: ***"Installation of the board library libkorebot"***.

6) Install the scripts for upload the file system and the kernel; see chapter 4.4.1.4: ***"Installation of the scripts for uploading the kernel and the file system: ktboot"***.

4.5.3 Full toolchain usage

4.5.3.1 Rebuilding the whole system

To rebuild the toolchain system, the image file and the kernel, execute the following instructions:

1) In a console in the directory ***/usr/local/korebot2-oetools-1.0***, source the following file to have access to the environment setup:

   source extras/profile

2) With the following commands you can build/rebuild the whole system:
   - Cleaning : ***bitbake -c clean korebot2-image***
   - Build : ***bitbake -c build korebot2-image***
   - Rebuild : ***bitbake -c rebuild korebot2-image***

=> The output files will be stored in:

/usr/local/korebot2-oetools-1.0/tmp/deploy/glibc/images/gumstix-custom-verdex/

Three main files will then be built:

- the kernel : ***uImage-2.6.24-r1-gumstix-custom-verdex.bin***
- its modules : ***modules-2.6.24-r1-gumstix-custom-verdex.tgz***
- the file system : ***Angstrom-korebot2-image-glibc-ipk-2007.9-test-20090520-gumstix-custom-verdex.rootfs.jffs2***

Remarks:

You may find updated version of these software at http://ftp.k-team.com

With the ***kt-boot*** (see chapter 4.4.2.2: "***Uploading the kernel and the file system: kt-boot usage***"), you can upload these files to the Korebot.

You can find more information about bitbake, the tool for executing task and managing metadata here: http://bitbake.berlios.de/manual/

And information about the OEtools is available here:

- Cross-compiler and tools (OpenEmbedded): http://www.openembedded.org/wiki/Main_Page
- Linux distribution of the Korebot: http://www.angstrom-distribution.org
- Gumstix documentation:
  - http://www.gumstix.org/
  - http://wiki.gumstix.org/index.php?title=Main_Page

4.5.3.2 Kernel modification

You can modify the kernel here by accessing to its menu with these commands:

source extras/profile

bitbake -c menuconfig gumstix-kernel

Then you rebuild it and the file system by executing these commands at the root of the *korebot oetools (**/usr/local/korebot2-oetools-1.0**):*

source extras/profile

and before rebuilding the kernel, you have to copy the config file:

***cp ***
***/usr/local/korebot2-oetools-1.0/tmp/work/gumstix-custom-verdex-angstrom-linux-gnueabi/gumstix-kernel-2.6.24-r1/linux-2.6.24/.config ***
/usr/local/korebot2-oetools-1.0/user.collection/packages/linux/gumstix-kernel-2.6.24/gumstix-custom-verdex/defconfig

Rebuild the kernel and filesystem:

bitbake -c rebuild gumstix-kernel

bitbake -c rebuild task-base-gumstix

bitbake korebot2-image

You can now upload the files to the Korebot with ***kt-boot*** (see chapter 4.4.2.2: "***Uploading the kernel and the file system: kt-boot usage***").

4.5.4 Packages installations

With OpenEmbedded, you can easily cross-compile existing packages or add your own.

4.5.4.1 Existing packages:

Many packages are available for Open-Embedded. Here below are the instructions to add an existing package:

1) Check if the package is already in ***/usr/local/korebot2-oetools-1.0/org.openembedded.dev/packages***

2) If it is present, you can compile it by running in the ***/usr/local/korebot2-oetools-1.0 directory***:

   source extras/profile

   bitbake PACKAGE_NAME

3) The package will be created in one of the folders ***/usr/local/korebot2-oetools-1.0/build/tmp/deploy/glibc/ipk***

4) Transfer the package to the Korebot (with Minicom or ssh; see chapter 5.2.1.2 "***To send a file to the Korebot (upload)***").

5) Then install it:

   ipkg install PACKAGE_NAME.ipk

You can list installed packages on the Korebot II and find your PACKAGE with the command:

ipkg list_installed | grep PACKAGE

4.5.4.2 Creating new package:

You can create new packages for the Korebot, following the instructions there:

http://www.gumstix.org/software-development/open-embedded/160-bitbake-tutorial.html

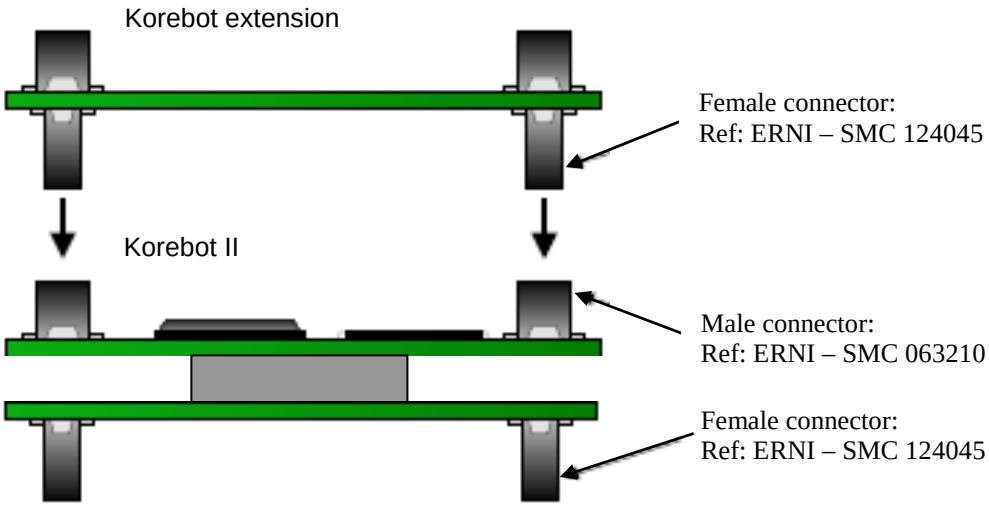
The recipes of the packages should be put in:

/usr/local/korebot2-oetools-1.0/user.collection/packages

5. Annexes

5.1 KB-250 Extension bus

Here after are described the extension specifications for developing a new extension board.



Overview

Figure 5-1: Korebot II - extension Bus

5.1.1 Mechanical specifications

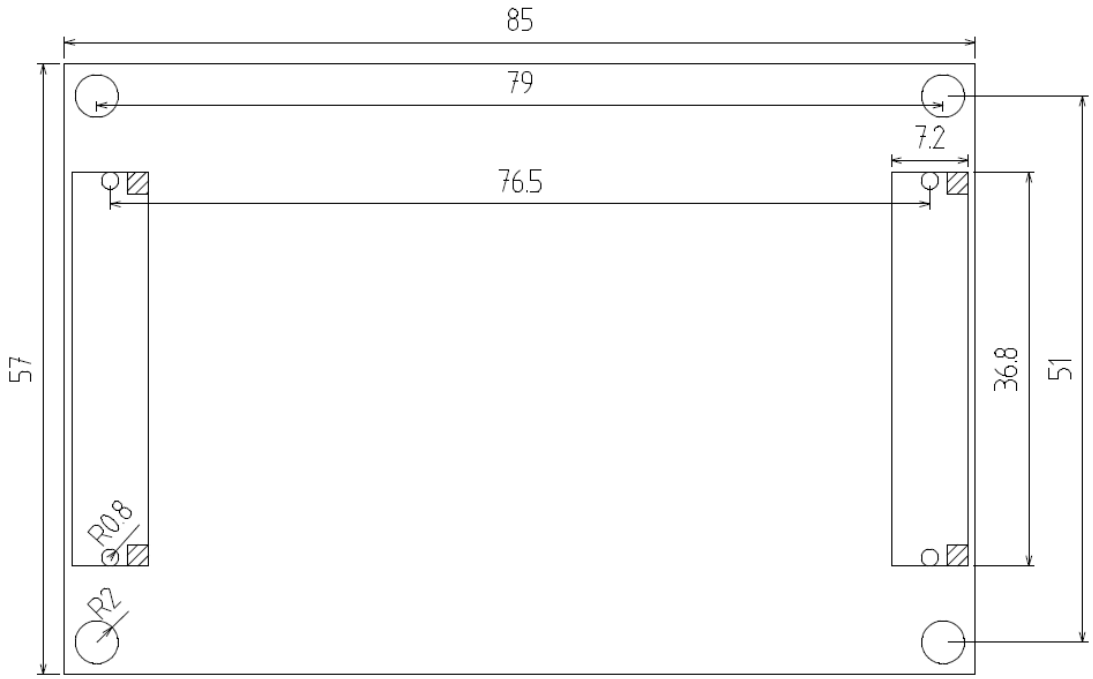


Figure 5-2: Board mechanical dimensions

5.1.2 Electrical specifications

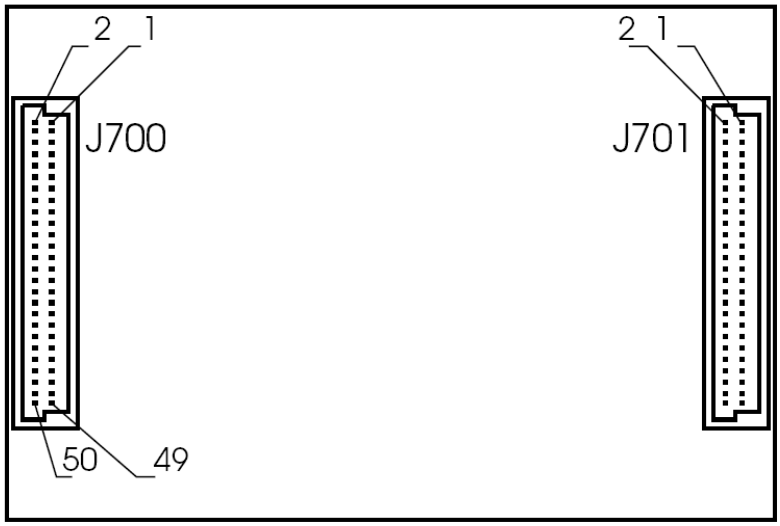


Figure 5-3: Connectors pinning (top view)

KB-250 Extension Interface (J700)					
	Signal	Function		Signal	Function
1	3.3V	Power	26	LCD_D14 (GPIO 72)	LCD controller
2	3.3V		27	LCD_D13 (GPIO 71)	
3	Gnd		28	LCD_D12 (GPIO 70)	
4	Gnd		29	LCD_D11 (GPIO 69)	
5	BT-Txd	BlueTooth UART ttyS1	30	LCD_D10 (GPIO 68)	
6	BT-Rxd		31	LCD_D9 (GPIO 67)	
7	BT-Cts		32	LCD_D8 (GPIO 66)	
8	BT-Rts		33	LCD_D7 (GPIO 65)	
9	IrDa-Txd	IrDa UART ttyS2	34	LCD_D6 (GPIO 64)	
10	IrDa-Rxd		35	LCD_D5 (GPIO 63)	
11	GPIO101	I/O	36	LCD_D4 (GPIO 62)	
12	NRESET		37	LCD_D3 (GPIO 61)	
13	PXA_NRESET		38	LCD_D2 (GPIO 60)	
14	PXA_NRESET_OUT		39	LCD_D1 (GPIO 59)	
15	X_USB+	USB Bus	40	LCD_D0 (GPIO 58)	
16	X_USB-		41	-	
17	SYS_EN		42	-	
18	-		43	5V	Power
19	LCD_D17 (GPIO 87)	LCD controller	44	5V	
20	LCD_D16 (GPIO 86)		45	-	
21	LCD_BIAS		46	-	
22	LCD_PCLK		47	GND	Power
23	LCD_LCLK		48	GND	
24	LCD_FCLK		49	GND	
25	LCD_D15		50	GND	

Figure 5-4: Interface J700

KB-250 Extension Interface (J701)					
	Signal	Function		Signal	Function
1	3.3V	Power	26	SSP-Rxd	SSP/SPI Bus
2	3.3V		27	SSP-Txd	
3	5V		28	SSP-Clk	
4	5V		29	SSP-Frm	
5	GND		30	CF BVD1	MMC Bus
6	GND		31	MMC Clk	
7	RS-Txd	Full RS232 UART ttyS0	32	MMC Cmd	
8	RS-Rts		33	PWM1	PWM
9	-		34	PWM2	
10	RS-Rxd		35	NAC Reset	Koresound bus
11	-		36	SYNC	
12	-		37	SDATA OUT	
13	-		38	SDATA IN0	
14	RS-Cts		39	SDATA IN1	
15	I2C SCL	I2C Bus	40	X BIT CLK	
16	I2C SCA		41	-	
17	USB OC	USB Bus	42	-	
18	USB DMOUT		43	-	
19	USB DPOUT		44	-	
20	MMC CF RESET	MMC Bus	45	-	
21	MMC CF READY		46	-	
22	USB -	USB Bus host master	47	-	
23	USB +		48	-	
24	USB ID		49	GND	Power
25	MMC DAT0	MMC Bus	50	GND	

Figure 5-5: Interface J701

5.2 Tools and commands

In this part, the detailed descriptions of several tools and helpful commands are explained.

You can use Minicom program on Linux and Teraterm on PC for connecting to the Korebot and also for sending and receiving files.

The instructions below are for Minicom, but can be adapted for Teraterm as some hints are given. Teraterm can be downloaded from there: http://en.sourceforge.jp/projects/ttssh2/releases/

5.2.1 Using serial port and Minicom

1) Connect and power up the Korebot as explained in chapter 4.3.

2) Install the Linux package **lrzsz** containing communications programs. If your Linux distribution is *Ubuntu*:

   sudo apt-get install lrzsz

5.2.1.1 Establish the serial connection

1) Set its parameters with the sub-menu "**Serial port**" setup of the menu **[configuration]** (Figure **5-7**) (keys "**Ctrl-a + o**") as described in Figure 5-6. You may modify the serial port device name depending where you plugged your serial port. If you use a serial to USB adapter, the serial device may be ***/dev/ttyUSB0***.

   On **Teraterm**, go to menu and Serial Port to set the settings.

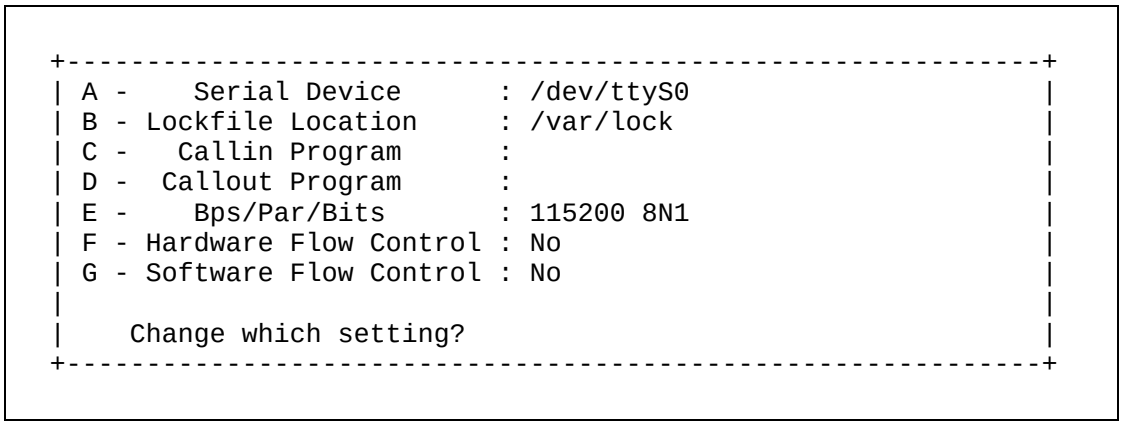


Figure 5-6: Minicom serial parameters

Save the settings with the sub-menu "***Save setup as dfl***" (Figure 5-7).

```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
+--------------------------+
```

Figure 5-7: Minicom configuration menu

5.2.1.2 To send a file to the Korebot (upload)

1) In the Minicom console, hold the keys "**Ctrl + a**" and press "**s**" and select "Z-Modem".

2) Select the file you would like to upload to the Korebot (navigate with the arrows keys, 2x "**spacebar**" to change directory and "**spacebar**" to select the file).

   Select [Okay] to send it.

For **Teraterm**, choose menu File, then Transfer, Z-Modem and Send.

5.2.1.3 To send a file to the computer (download)

1) In the Minicom console at the prompt of the Korebot, type the following command, where FILENAME is the file you would like to send.

   lsz FILENAME

⇨ The file ***FILENAME*** is sent to the last directory Minicom used (or if not changed, where it started).

For **Teraterm**, run the command ***lsz FILENAME*** then choose menu File, then Transfer, Z-Modem and Receive. The file will be in the directory given by menu File, "Change directory".

5.2.2 Using a Wireless compact flash card

Two wireless compact flash models are supported. The card name and its driver are listed below:

A) Ambicom WL1100C-CF with hostap_cs driver module

B) Ambicom WL5400G-CF with libertas_cs driver module

Remark:

The following instructions are for the wireless compact flash ***B) WL5400G-CF***. With the model ***A) Ambicom WL1100C-CF***, you have to instruct the driver to ignore Vcc differences by setting module parameter ***ignore_cis_vcc=1***

- either in ***/etc/module*** : put ***hostap_cs ignore_cis_vcc=1*** before the line: ***### update-modules: start processing /etc/modutils/pcmcia***
- or for testing, by manually loading the module with ***'modprobe hostap_cs ignore_cis_vcc=1'*** before inserting the card.

Then you have to replace the wireless port name ***eth0*** by ***wlan0*** in the following instructions.

1) Insert a Wireless compact flash card in the Korebot

2) Load the module by typing:

modprobe pxa2xx_cs

You may load the Wifi module automatically by adding ***pxa2xx_cs*** in the file

/etc/modules

You can use the following command echo to add the module name to the file:

echo pxa2xx_cs >>/etc/modules

3) Configure the wireless network:

i) Without any encryption for security:

Modify the file ***/etc/network/interfaces*** with your wireless network settings with ***vi*** editor (see chapter 5.2.7 "***Using vi text file editor***"):

```
/****************** /etc/network/interfaces **********/
# The loopback interface
#
auto lo
iface lo inet loopback

#
# Wireless interfaces
#
auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
```

wireless_mode managed
wireless_essid YOUR_SSID_OF_NETWORK
address YOUR_IP_ADDRESS
netmask YOUR_NETMASK
gateway YOUR_GATEWAY_IP

/*******/***

ii) WEP encryption support

a) for configuring the wifi connection, type:

iwconfig eth0 essid YOUR_SSID_OF_NETWORK

b) if the network is secured, enter the key by typing :

iwconfig eth0 key YOUR_KEY

c) then set an ip address to the korebot:

ifconfig eth0 YOUR_IP_ADDRESS

d) configure the gateway by entering the gateway ip:

route add default gw YOUR_GATEWAY_IP eth0

e) insert the local domain name in ***/etc/resolv.conf***

echo search YOUR_LOCAL_DOMAIN_NAME >>/etc/resolv.conf

f) and the dns server

echo nameserver YOUR_DNS_SERVER_IP_ADDRESS >>/etc/resolv.conf

You can also create a file in ***/etc/network/if-pre-up.d*** named wireless to have these settings saved.

Put the following into it:

#!/bin/sh
ifconfig eth0 up
iwconfig eth0 essid YOUR_SSID_OF_NETWORK
iwconfig eth0 key s:YOUR_KEY
ifconfig eth0 YOUR_IP_ADDRESS
route add default gw YOUR_GATEWAY_IP eth0

And make it executeable with the command:

chmod +x /etc/network/if-pre-up.d/wireless

And the following in a file named ***/etc/resolv.conf***:

search YOUR_LOCAL_DOMAIN_NAME

nameserver YOUR_DNS_SERVER_IP_ADDRESS

iii) WEP, WPA and other encryptions:

a) Create a file named ***/etc/wpa_supplicant/wpa_supplicant.conf*** and insert your selected wireless encryption:

WEP:

```
#Shared WEP key connection (no WPA):
network={
        ssid="YOUR_SSID"
        key_mgmt=NONE
        wep_key0="YOUR_WEP_KEY"
        auth_alg=SHARED
        wep_tx_keyidx=0
        priority=5
}
```

WPA-TKIP:

```
#/etc/wpa_supplicant/wpa_supplicant.conf
#with WPA-PSK TKIT:
network={
        ssid="YOUR_SSID"
        psk="YOUR_PASS_KEY"
        key_mgmt=WPA-PSK
        group=TKIP
        pairwise=TKIP
        proto=WPA
        priority=5
}
```

You can check the following link for other encryptions:
http://hostap.epitest.fi/wpa_supplicant/

b) run the daemon controlling the wireless connection with the following command:

wpa_supplicant -c/etc/wpa_supplicant/wpa_supplicant.conf -ieth0 -Dwext -B

c) In ***/etc/network/if-pre-up.d*** named wireless, add the following commands:

```
#!/bin/sh
ifconfig eth0 up
ifconfig eth0 YOUR_IP_ADDRESS
route add default gw YOUR_GATEWAY eth0
pre-up wpa_supplicant -c/etc/wpa_supplicant/wpa_supplicant.conf -ieth0 -Dwext –B
post-down killall -q wpa_supplicant
```

d) reboot the system or restart the network with the following command:

/etc/init.d/networking restart

5.2.3 Transferring files using scp (ssh)

1) Establish a network connection between the computer and the Korebot (for using wireless, see chapter 5.2.2 ***"Using a Wireless compact flash card"***).

2) Execute the following command:

scp FILE root@KOREBOT_IP:/home/root

where

FILE : is the file to transfer,

KOREBOT_IP : is the Korebot ip address.

5.2.4 Using the KoreUSBCam module

You can connect the KoreUSBCam and use it as described in the user manual available at the link below.

You don't need to install the driver, as it is already included for the Korebot II.

The information and main programs can be found here:

- User manuals:
  Current KoreUSBCam:
  http://ftp.k-team.com/korebot/koreusbcam/KoreUSBCam_QuickUserManual_v1.0.pdf

  Old KoreUSBCam:
  http://ftp.k-team.com/korebot/koreusbcam/KoreUSBCam_UserManual_v1.2.pdf

- Main folder with user manual, applications programs:
  http://ftp.k-team.com/korebot/koreusbcam/

You can use the following programs compiled for the Korebot II:

- Ktgrab: video acquisition server:

http://ftp.k-team.com/korebot/koreusbcam/application/KTGrab_java_kore2_20090602.tar.bz2

- Ktgrab: video acquisition client:

  http://ftp.k-team.com/korebot/koreusbcam/application/libfg-0.3.1-kb1_kore2_20090602.tar.bz2

- Driver source code:

  http://ftp.k-team.com/korebot/koreusbcam/driver/source/gspcav1-20071224-kb2_kore2_20090602.tar.bz2

Remark: the camera module focus is manual only. No autofocus mode is available.

5.2.5 nfs configuration

The first service to set up should be transparent file sharing using NFS. Most Linux distributions include NFS support by default, and the KoreBot system is ready to be connected. The directory to be shared between the computer and the board must be declared to the NFS service in the ***/etc/exports*** configuration file. Please refer to NFS documentation, or man exports, for a detailed syntax description. Basically, the following line should be added to the file ***/etc/exports*** on the computer:

/mnt/nfsarm KOREBOT_IP/255.255.255.0(rw,no_root_squash,sync)

And this directory should be created on the computer with: ***mkdir /mnt/nfsarm***

Then the export should be refreshed with the command: ***exportfs –av***

The next step is to mount the shared directory to the KoreBot file system. Mounting a local hard drive partition or a network directory is exactly the same from the user point of view, the mount commands should be on the Korebot, where ***COMPUTER_IP***, is the IP address of the computer which the Korebot will be connected:

mkdir /mnt/nfs

mount -t nfs -o nolock COMPUTER_IP:/mnt/nfsarm /mnt/nfs

If the NFS service is not started on the PC, the mount command will issue the following message:

mount: RPC: Unable to receive; errno = Connection refused

NFS: mount program did not respond!

mount: nfsmount failed: Bad file descriptor

The NFS service is usually started using a startup script for which location and name depend on you distribution (for example ***/etc/init.d/nfs***).
Documentation for the distribution should detail the method to start and stop services.

Caution: For the nfs service to work properly, the portmap service should be started as well and if a firewall is active on the host machine, it should be configured to allow the nfs port access from the KoreBot.

Once the directory is successfully mounted, it can be accessed from the board exactly as if it was a local directory. Files on the PC can be read or written, new files can be created, and programs can be executed, as long as they are ARM executables. If required, the shared directory can be unmounted using the command:

umount myMountPoint

5.2.6 Korebot II console by WIFI and ssh

You can connect to the KorebotII also by wireless. Execute the following command in a Linux terminal or use your favorite ssh program (Tera Term on Windows):

ssh root@KOREBOT_IP

where ***KOREBOT_IP*** is the Korebot ip address.

At the password prompt, just press the RETURN key (no password). You will have a terminal that is the Korebot II console.

5.2.7 Using vi text file editor

You can use the installed ***vi*** text file editor to modify files directly on the Korebot.

Launch it with: ***vi FILENAME***
where ***FILENAME*** is the filename with path you would like to edit.

Here below are the basic commands:

i	enters in write mode for adding text. You will see the I indicating this modeat the last line of the file: **I /tmp/essai 1/1 100%**
a	enters in write mode for adding text. You will see the I indicating this modeat the last line of the file: **I /tmp/essai 1/1 100%**
ESC	to go back in command mode, push ESC key
x	delete character on cursor
d d	delete current line
ESC w q !	to save and quit
ESC q !	to quit without saving
arrow keys	move around text

You will find more commands here: http://www.tutorialspoint.com/unix/unix-vi-editor.htm

5.2.8 Debugging

You can debug a program you made directly in console on the Korebot II or remotely.
The files can be found here:

http://ftp.k-team.com/KorebotII/software/common/debug/

5.2.8.1 Debugging on the Korebot II

- You need firstly to install the following packages with the commands in that order:

  ipkg install ncurses_5.4-r9_armv5te.ipk

  ipkg install libcidn1_2.9-r37.4.6_armv5te.ipk

  ipkg install libc6_2.9-r37.4.6_armv5te.ipk

  ipkg install libgcc1_4.3.3-r24.2.6_armv5te.ipk

  ipkg install libpython2.6-1.0_2.6.6-ml12.2.6_armv5te.ipk

  ipkg install gdb_7.0-r0.5_armv5te.ipk

- Then your program must be cross-compiled with the option ***–g*** . Edit your Makefile and replace the flag ***-O2*** by ***-g*** .
- Copy to the Korebot II the compiled program and its source file.
- And run for debugging it:

  gdb YOUR_PROGRAM

  The basic commands are:

r	:	run
s	:	next: one step in the program; enter in subroutines
n	:	one step in the program
b line/function	:	break at line/function
delete break	:	delete breakpoint number
until line	:	continue until line
c	:	continue
l	:	list code
q	:	quit gdb
h	:	help; you can have all the commands here

  A common sequence of debugging can be:

  - to list the code with ***l***
  - set a breakpoint at the beginning of main with ***b main***
  - run the program with ***r***

- execute a step with ***n***
- display a variable with ***p VAR_NAME***
- execute next step with ***n***
- continue to the end with ***c***
- Quit with ***q***.

5.2.8.2 Remote debugging

You need firstly to install the following packages with the commands:

ipkg install libthread-db1_2.9-r37.4.6_armv5te.ipk

ipkg install gdbserver_7.0-r2.0.5_armv5te.ipk

- Then run the gdbserver with your cross-compiled program, the computer host ip and a unused port number as argument:

  gdbserver --multi HOST_IP:1234 YOUR_PROGRAM

- Source your env.sh (see chapter 4.4.1.1) and run:

  source ~/development_k2_v1.0/env.sh

- Go to the folder where you cross-compiled your program

- Copy the file ***arm-angstrom-linux-gnueabi-gdb*** from the link below to :

  /usr/local/korebot2-oetools-1.0/tmp/cross/bin

  http://ftp.k-team.com/KorebotII/software/common/debug/

The libexpat need to bes installed on the computer:

- Create the directory on the computer:

  sudo mkdir -p /usr/local/korebot2-oetools-1.0/tmp/staging/i686-linux/lib

- Unpack and copy the libexpat files to that directory (with sudo) from the link below:

  http://ftp.k-team.com/KorebotII/software/common/debug/

- Create the links with the following command (in one line each):

  sudo ln -s /usr/local/korebot2-oetools-1.0/tmp/staging/i686-linux/lib/libexpat.so.0.5.0 /usr/local/korebot2-oetools-1.0/tmp/staging/i686-linux/lib/libexpat.so.0

sudo ln -s /usr/local/korebot2-oetools-1.0/tmp/staging/i686-linux/lib/libexpat.so.0.5.0 /usr/local/korebot2-oetools-1.0/tmp/staging/i686-linux/lib/libexpat.so

You can use the debugger in command line or with ***ddd*** GUI:

Command line:

- Execute for running the debugger in the computer:
  arm-angstrom-linux-gnueabi-gdb YOUR_PROGRAM
- In this debugger, set parameter and connect to the remote gdb with the 2 commands:
  set sysroot /usr/local/korebot2-oetools-1.0/tmp/staging/arm-angstrom-linux-gnueabi
  target extended-remote YOUR_KOREBOT_IP_ADDRESS:1234

  ⇨ The commands (r,…,h) are the same as for ***gdb*** in chapter 5.2.8.1.

With GUI:

- On your computer, install ddd the debugger GUI with :
  sudo apt-get install ddd
- Launch ddd with:
  ddd --debugger arm-angstrom-linux-gnueabi-gdb YOUR_PROGRAM
- In the window at the bottom where the (gdb) prompt is run these 2 commands:
  set sysroot /usr/local/korebot2-oetools-1.0/tmp/staging/arm-angstrom-linux-gnueabi
  target extended-remote YOUR_ KOREBOT_ADDRESS:1234

The sequence of debugging and also the commands are the same as described above in chapter 5.2.8.1.

You can even send your program to the remote (Korebot II) with the gdb commands:

remote put *hostfile targetfile*
set remote exec-file *targetfile*

where *hostfile* and *targetfile* is the same name of your new program to be debugged.

For quitting the debugger and remote, execute commands ***monitor exit***, then ***disconnect*** and finally ***quit*** .

Debugging libkoret functions:

If you would like to debug the libkorebot functions, you will have to recompile the libkorebot in debug mode with the commands to be executed in the root directory libkorebot source files:

cd ~/development_k2_v1.0/ libkorebot-1.19-kb1

make clean

make DEBUG=1

Then transfer the new complied file

~/development_k2_v1.0/libkorebot-1.19-kb1build-korebot-2.6/lib/libkorebot.so.1.19

to the ***/usr/lib*** directory of the Korebot.

Tips for using the advantage of the GUI:

- For setting a breakpoint, double-click on the line you want to put it, just before the line number or right click at the same place or choose “Set Breakpoint”.
- With the floating menu, you have the different commands for debugging.
- You can add a watch on a variable with clicking with the right mouse button on the variable and choose “Display VARIABLE_NAME”.

5.2.8.3 Remote debugging with Eclipse

You can also debug remotely your program with Eclipse (see chapter 4.4.2.5 for a proper installation). Follow instructions below:

1) Install ***gdbserver*** on your Korebot as explained in chapter 5.2.8.2 above.

2) From http://ftp.k-team.com/KorebotII/software/common/openssh ,

download to the Korebot the following openssh packages:

openssh-scp_4.6p1-r3_armv5te.ipk

openssh-ssh_4.6p1-r3_armv5te.ipk

openssh-sshd_4.6p1-r3_armv5te.ipk

openssh_4.6p1-r3_armv5te.ipk

openssh-sftp-server_4.6p1-r3_armv5te.ipk

openssh-sftp_4.6p1-r3_armv5te.ipk

3) Remove ***dropbear*** program:

ipkg remove dropbear

4) Connection with SSH connection will be dropped in the middle then connect with serial port to the Korebot 2.

5) Install these openssh packages in that order. The command for each is

ipkg install PACKAGE.ipk

openssh-scp_4.6p1-r3_armv5te.ipk

openssh-ssh_4.6p1-r3_armv5te.ipk

openssh-sshd_4.6p1-r3_armv5te.ipk

openssh_4.6p1-r3_armv5te.ipk

openssh-sftp-server_4.6p1-r3_armv5te.ipk

openssh-sftp_4.6p1-r3_armv5te.ipk

6) With this new openssh server, a password is needed: run this command and create a new password: ***passwd***

7) Reboot the korebot

8) Remove on your computer the known host (as the ssh server changed the security key changed): ***rm ~/.ssh/known_hosts***

9) On Eclipse menu 'Run' => Debug Configurations' , double click on 'C++ Remote Application'

=> test Debug is created.

10) In the 'Main' tab => “C/C++ application”, enter: Debug/test

11) On “Connection”: push “New” button and choose “SSH”, in the next window, ***Hostname***: ip address of your korebot 2 and push “Finish” button.

12) On "Remote Absolute File Path for C/C++ Application" modify to ***/home/root/test***

13) In the Debugger tab, choose

'***/usr/local/korebot2-oetools-1.0/tmp/cross/bin/arm-angstrom-linux-gnueabi-gdb***'

14) On 'Main' tab change 'GDB Debugger' to '***/usr/local/korebot2-oetools-1.0/tmp/cross/bin/arm-angstrom-linux-gnueabi-gdb***'

15) On 'Shared Libraries' tab add '***/usr/local/korebot2-oetools-1.0/tmp/staging/arm-angstrom-linux-gnueabi/lib***'

16) Then Push Debug, enter ***root*** as "User ID" and use the password created above at point 6)

17) Debug step by step.

You can go again into the settings, and select "Release" configuration when your program has been fully debugged.

You can run the program from the korebot2 (with a remote terminal) or with Eclipse GUI.
After having set the Debug configuration, the parameters are also available for the run.

Just go the menu Run, then choose “Run”.

In the lower part of the Eclipse window, you will see the output of your program in the Console tab.

5.2.9 Development on a virtual machine

Files needed:

- VirtualBox installed (from http://www.virtualbox.org/)

And the images files from the DVD or from there:

http://ftp.k-team.com/KorebotII/software/common/Ubuntu_VirtualBox_image/

- Image file: {94dd2975-4410-4e00-9a4c-b6186322a9c3}.vmdk
- Image file checksum: Ubuntu-KorebotII_VirtualBox.mf
- Image file settings: Ubuntu-KorebotII_VirtualBox.ovf

Hardware needed:

- computer with at least 2GB of RAM and 4 GB of free disk space

5.2.9.1 Installation of the virtual machine

Follow these instructions for installing the virtual machine:

- Install VirtualBox from http://www.virtualbox.org/
- Go to File/Import Appliance and import the Ubuntu-KorebotII_VirtualBox.ovf file settings.

  This will take some time.

- You may configure the serial port for using the KoreConnect to connect to the Korebot II.
- Select the "Ubuntu-KorebotII" machine and press "Settings" button.
- Go to "Serial Ports" settings
- On the tab "Port 1", enable the checkbox "Enable Serial Port"
- Choose : Port Number: COM1 (=> this will be the Linux serial port /dev/ttyS0)

Port Mode:	Host Device
Port/File Path:	COM1, if you have a standard serial port and using COM1 or change to adapt to your computer serial port.

If you have the KheperaIII and the Korebot II, you can even use the Bluetooth com port instead of the KoreConnect. See the Khepera III user manual for the Bluetooth connection.

The network should be detected and installed autonomously.
You may want to share folders between your virtual machine and your computer.

You can share folders like this:

- Select the "Ubuntu-KorebotII" machine and press "Settings" button.
- Go to "Shared Folders" settings
- Push the Insert keyboard key or the icon "Add Shared Folder"
- In the new dialog box, for Folder path choose a existing (or create one) folder on your computer. This folder will be available from the virtual machine (ex : "C:\virtual_machine_shared").
  The folder name should be autonomously created. You can change it. Try to not add any space in the name (ex : "virtual_machine_shared")..

After starting the virtual machine and having installed the VirtualBox drivers as described in the next points, the shared folder will be mounted in ***/mnt/windows/*** .

5.2.9.2 Usage of the virtual machine

- Start the virtual machine "Ubuntu-KorebotII". Its login is :
  user: kteam
  password: kteamroot

- You may need to add the VirtualBox drivers, depending of the version of your computer and your VirtualBox. These drivers provide access to network, shared folder, mouse integration, ...
  In the "Device menu" (top bar of the window) of your virtual machine, Install guest additions.

The Ubuntu OS should be updaded. Use the Update Manager from the menu System, Administration.

Korebot II development:

You will have the cross-compiler (full toolchain and sources) and development folder already installed (see the Korebot II user manual) in the virtual machine. You can start developing as described in chapter "4.4.2.1 Application development".

You should update the toolchain of the virtual machine to the latest available. Follow instructions in chapter 4.5.2 with the new version from:

http://ftp.k-team.com/KorebotII/software/full_toolchain/

Also the libkorebot should be updated. See instructions at chapters 4.4.1.3 and 4.4.1.5.

Stopping the virtual machine:

When going to the menu "Machine" and "Close, you will have 3 choices:

- "Save the machine state": saves the current state, which will be reloaded next time you restart it.
- "Send the shutdown signal", or press the shutdown button into the machine:
  power off the machine and save only the work done.

- "Power off the machine": WARNING: it doesn't save anything; any work done will be discarded!

Serial port sharing:

You may configure your virtual machine to share the serial port to connect to the Korebot II using Bluetooth …).

- Power off your virtual machine.
- Select your machine image and press "Settings" button.
- Go to "Serial Ports" settings;
- On the tab "Port 1", enable the checkbox "Enable Serial Port" and choose:

  Port Number: COM1 (=> this will be the Linux serial port /dev/ttyS0)

  Port Mode: Host Device

  Port/File Path: COM1 if you have a standard serial port and using COM1. Or change to adapt to your computer serial port or Bluetooth emulated serial port.

5.2.10 Java virtual machine

The Java virtual machine is not installed by default on the Korebot II.

5.2.10.1 Instructions for Installing the java virtual machine on the Korebot II

You will find from the link below the packages files to install for having the Java virtual machine on the KorebotII of your KheperaIII.

http://ftp.k-team.com/KorebotII/software/common/java_vm/

classpath-common_0.96.1-r2_armv5te.ipk

classpath-dev_0.96.1-r2_armv5te.ipk

classpath_0.96.1-r2_armv5te.ipk

jamvm_1.5.0-r0_armv5te.ipk

- Download and transfer them to the robot and install them with the command:

ipkg install PACKAGE_NAME

for each, where PACKAGE_NAME, is any of the file finishing
by .ipk extension inside the directory of the link above.
Install the package javavm_ at the end.

- Install also the Khepera3 interface library file by copying ***libKhepera3.so*** file from http://ftp.k-team.com/KorebotII/software/common/java_vm/Khepera3_java_20120120.zip into the directory ***/usr/lib*** of the Korebot2.

5.2.10.2 Testing and start developping

You can test and start developping in Java with the instructions below:

- install the java compiler openjdk-6-jdk (command on Ubuntu):
sudo apt-get install openjdk-6-jdk

a) "Hello World" example
- Create a file Hello.java with the following lines on the computer:

```
/* Hello.java */
public class Hello
{
      public static void main(String[] args)
      {
            System.out.println("Hello world!");
      }
}
```

and compile it with the command: ***javac Hello.java***
Or use your favorite java compiler.

=> This will create a file named ***Hello.class***

- transfer the file ***Hello.class*** on the Korebot II and run it with:

java Hello

=> This should return: Hello world!

b) start from the Khepere3.java example:

- extract the sources from
http://ftp.k-team.com/KorebotII/software/common/java_vm/Khepera3_java_20120120.zip

- modify and add your code inside the part
// ***main program public static void main(String[] args)***

- compile the program with: ***javac Khepera3.java***

- on the Korebot2, execute this for adding the library path for finding the ***libKhepera3.so***:
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib

- transfer ***Khepera3.class*** to the Korebot2 and run it with: ***java Khepera3***

5.2.10.3 Optional: instructions how to (re-)build the java virtual machine

The machine was build with the commands below. Execute it only if you want to rebuild or patch the java virtual machine.

- You need the Korebot II full toolchain (See chapter 4.5).

Then with the commands below, build the java virtual machine and its library:

- Open a terminal and go to the directory ***/usr/local/korebot2-oetools-1.0/***

- Execute this command to have access to the variables: ***source extra/profile***

- then install the java compiler on your computer (on Ubuntu):
sudo apt-get install openjdk-6-jdk

- build the the classpath packages:
bitbake classpath

- and the virtual machine:
bitbake jamvm

=> These 2 last steps will produce the packages there:
/usr/local/korebot2-oetools-1.0/tmp/deploy/glibc/ipk/armv5te

classpath-common_0.96.1-r2_armv5te.ipk
classpath-dev_0.96.1-r2_armv5te.ipk
classpath_0.96.1-r2_armv5te.ipk
jamvm_1.5.0-r0_armv5te.ipk

These Instructions were adapted from:
http://wiki.gumstix.org/index.php?title=Category:How_to_-_JAVA

5.2.10.4 Optional: instructions how to (re-)build the Khepera3 interface library

You can rebuild the Khepera3 interface library for adding new functions
or modify parameters.

- extract the sources from
  http://ftp.k-team.com/KorebotII/software/common/java_vm/Khepera3_java_20120120.zip

- If you would like to add some new function:
  - add the declaration of the new function in ***Khepera3.java***, in the beginning
    (under ***// methods available*** comment).

  - update the ***Khepera3.h*** header with the command: ***javah Khepera3***

  - add the new declaration of ***Khepera3.h*** in ***libKhepera3.c*** and transform this function
    declaration into the definition by adding code inside.

- cross-compile the interface library ***libKhepera3.c***:
  - firstly, source the env.sh, as mentionned in the chapter 4.4.2.

      - cross-compile with the command (one line):
        arm-angstrom-linux-gnueabi-gcc -shared libKhepera3.c -o libKhepera3.so
        -I /usr/lib/jvm/java-6-openjdk/include -I $INCPATH -L $LIBPATH -lkorebot

      => This will create the library ***libKhepera3.so***.

6. WARRANTY

K-TEAM warrants that the Korebot II is free from defects in materials and workmanship and in conformity with the respective specifications of the product for the minimal legal duration, respectively one year from the date of delivery.

Upon discovery of a defect in materials, workmanship or failure to meet the specifications in the Product during the afore mentioned period, Customer must request help on K-Team Internet forum on http://www.k-team.com/forum/ by detailing:

- the type of Korebot II used (version)
- the kernel version of the Korebot II
- the programming environment of the Korebot/robot (standard, version, OS)
- the standard use of Product before the appearance of the problem
- the description of the problem.

If no answers have been received within two working days, Customer can contact K-TEAM support by phone or by electronic mail with the full reference of its order and Korebot II serial number.

K-TEAM shall then, at K-TEAM's sole discretion, either repair such Product or replace it with the equivalent product without charging any technical labor fee and repair parts cost to Customer, on the condition that Customer brings such Product to K-TEAM within the period mentioned before. In case of repair or replacement, K-TEAM may own all the parts removed from the defective Product. K-TEAM may use new and/or reconditioned parts made by various manufacturers in performing warranty repairs and replacement of the Product. Even if K-TEAM repairs or replaces the Product, its original warranty term is not extended.

This limited warranty is invalid if the factory-applied serial number has been altered or removed from the Product.

This limited warranty covers only the hardware and software components contained in the Product. It does not cover technical assistance for hardware or software usage and it does not cover any software products contained in the Product. K-TEAM excludes all warranties expressed or implied in respect of any additional software provided with Product and any such software is provided "AS IS" unless expressly provided for in any enclosed software limited warranty. Please refer to the End User License Agreements included with the Product for your rights with regard to the licensor or supplier of the software parts of the Product and the parties' respective obligations with respect to the software.

This limited warranty is non-transferable.

It is likely that the contents of Customer's flash memory will be lost or reformatted in the course of the service and K-TEAM will not be responsible for any damage to or loss of any programs, data or other information stored on any media or any part of the Product serviced hereunder or damage or loss arising from the Product not being available for use before, during or after the period of service provided or any indirect or consequential damages resulting therefore.

IF DURING THE REPAIR OF THE PRODUCT THE CONTENTS OF THE FLASH MEMORY ARE ALTERED, DELETED, OR IN ANY WAY MODIFIED, K-TEAM IS NOT RESPONSIBLE WHATEVER. CUSTOMER'S PRODUCT WILL BE RETURNED TO CUSTOMER CONFIGURED AS ORIGINALLY PURCHASED (SUBJECT TO AVAILABILITY OF SOFTWARE).

Be sure to remove all third parties' hardware, software, features, parts, options, alterations, and attachments not warranted by K-TEAM prior to Product service. K-TEAM is not responsible for any loss or damage to these items.

This warranty is limited as set out herein and does not cover, any consumable items (such as batteries) supplied with the Product; any accessory products which is not contained in the Product; cosmetic damages; damage or loss to any software programs, data, or removable storage media; or damage due to (1) acts of God, accident, misuse, abuse, negligence, commercial use or modifications of the Product; (2) improper operation or maintenance of the Product; (3) connection to improper voltage supply; or (4) attempted repair by any party other than a K-TEAM authorized module service facility.

This limited warranty does not apply when the malfunction results from the use of the Product in conjunction with any accessories, products or ancillary or peripheral equipment, or where it is determined by K-Team that there is no fault with the Product itself.

K-TEAM EXPRESSLY DISCLAIMS ALL OTHER WARRANTIES THAN STATED HEREINBEFORE, EXPRESSED OR IMPLIED, INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE TO THE FULLEST EXTENT PERMITTED BY LAW.

Limitation of Liability: IN NO EVENT SHALL EITHER PARTY BE LIABLE TO THE OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM PERFORMANCE OR FAILURE TO PERFORM UNDER THE CONTRACT, OR FROM THE FURNISHING, PERFORMANCE OR USE OF ANY GOODS OR SERVICE SOLD OR PROVIDED PURSUANT HERETO, WHETHER DUE TO A BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, OR OTHERWISE. SAVE THAT NOTHING HEREIN SHALL LIMIT EITHER PARTY'S LIABILITY FOR DEATH OR PERSONAL INJURY ARISING FROM ITS NEGLIGENCE, NEITHER PARTY SHALL HAVE ANY LIABILITY TO THE OTHER FOR INDIRECT OR PUNITIVE DAMAGES OR FOR ANY CLAIM BY ANY THIRD PARTY EXCEPT AS EXPRESSLY PROVIDED HEREIN.

K-TEAM S.A.
Z.I. PLANS-PRAZ 28
1337 VALLORBE
SWITZERLAND