Software installation tips for the Cycab platform

Soraya Arias, Christophe Braillon
Software installation tips for the Cycab platform

Soraya Arias, Christophe Braillon
Project-Team SED


Abstract: Software installation tips for the Cycab robot

Key-words: CAN, Cycab, GCC cross-compiler for MPC555, bootloader
Software installation tips for the Cycab platform

Résumé : Software installation tips for the Cycab robot

Mots-clés : CAN, Cycab, GCC cross-compilateur pour MPC555, bootloader
# Contents

1. **Introduction** ................................................. 4

2. **CAN Network Controller Linux driver** ................. 5  
   2.1 Adlink PCI-7841 ........................................... 5  
   2.2 Peak PCAN-PCI and PCAN-USB dongle ..................... 5  
   2.3 Testing the driver ........................................ 6  
      2.3.1 CAN plug device ..................................... 6  
      2.3.2 For a Linux driver using SOCKET-CAN interface 7

3. **MPC555 programming** ..................................... 8  
   3.1 Required tools .......................................... 8  
   3.2 Build a cross-compiler for MPC555 targets .......... 8

4. **Boot-loading procedure** ................................ 10  
   4.1 Boot-loader binary format .............................. 10  
   4.2 Sending protocol ....................................... 11

5. **CycabHDK package** ....................................... 11  
   5.1 Installation and Compilation ........................... 11

A. **Obsolete** .................................................. 13  
   A.1 Making a disk image of Robosoft installation ....... 13  
   A.2 Installing a new hard drive disk ...................... 13  
   A.3 Installing Ubuntu Linux ................................ 14
1 Introduction

We bought the Cycab in 2000 to Robosoft. This robotic platform integrated an embedded PC and a tactile screen, installed by Robosoft.

The PC provided a Red Hat 9.0 Linux distribution, Linux kernel version 2.4.22-adeos, plus the softwares to control the Cycab including Syndex tools.

As for the tactile screen, its size and resolution were not suited for our needs and we got troubles with the related drivers.

As it soon became difficult to maintain such a configuration, and because we wanted to use our own tools to command the Cycab on more up-to-date Linux distribution, GCC compiler, Linux kernel version, etc, we decided to replace the whole Robosoft setup (embedded PC + tactile screen) by a laptop.

With such a setup, we do not need the softwares installed by Robosoft anymore. We need a laptop providing an up-to-date Linux distribution, in which a CAN driver can be installed to enable the communication (through a CAN dongle) with the Cycab nodes and thus control the robotic platform. Thus the tactile screen becomes useless.

To pilot and control the Cycab with the laptop, you need to:

1. Install an up-to-date Linux distribution: we are using Ubuntu 12.04 LTS;
2. Build and install a CAN Linux driver;
3. Build and install a cross-compiler for MPC555 target;
4. Retrieve the Cycab Hardware toolkit (CycabHDK) we develop:
   
   svn checkout svn+ssh://yourlogin@scm.gforge.inria.fr/svnroot/cycabhdk/trunk

5. Build the tools and binaries from CycabHDK: the binaries to download on the Cycab nodes, the down-loader tool to send the binaries through CAN bus for each Cycab node and the applications to control the Cycab from the laptop.

Then you will be able to:

- Download the binary file for each MPC555 node of the Cycab (one for the front node, another for the rear node);
- Once the Cycab nodes have been successfully downloaded, you can drive the Cycab using the joystick (default behavior) or you can launch the applications we provide in the CycabHDK package to control the Cycab.

This document is organized as follows. The section 2 presents how to build and install a CAN Linux driver. In section 3 we explain how to build and install a GCC cross-compiler for MPC555 target. In section 4 the boot-loading process for the Cycab MPC555 nodes is explained and information about the CycabHDK package is provided in section 5.

At the end of the document you can find a section containing information for those who still uses the obsolete embedded PC.
2 CAN Network Controller Linux driver

Original embedded PC from Robosoft integrates an Adlink PCI-7841 card. So we add information about the driver for this card.

But on our laptop configuration we choose a PEAK Systems CAN-USB dongle, because PEAK Systems provide up-to-date Linux drivers for their devices.

2.1 Adlink PCI-7841

For the PCI-7841, you can find Linux drivers compliant with the following Linux kernel versions (last upload 12/07/2012): 2.4.18, 2.6.11, 2.6.21, 2.6.25.

You can download the sources from (you need to be logged in to retrieve this page):


These sources are licensed under GPL.

For compiling and installing the driver, follow the documentation given by Adlink.

2.2 Peak PCAN-PCI and PCAN-USB dongle

You can download the sources from:

http://www.peak-system.com/fileadmin/media/linux/index.htm#download

These sources are licensed under GPL.

On a Ubuntu 12.04 LTS distribution release (Linux kernel version 3.2.0), we use the driver release version 7.7 or 7.8 (last upload 31.01.2013).

For compiling and installing the driver, documentation is given by PEAK Systems.

Make sure the libpopt-dev library (used for parsing command line parameters) is installed in your system.

By default, this driver sets CAN baud rate to 500kBaud and supports the network NETDEV_SUPPORT interface, i.e uses SocketCAN interface to handle the CAN bus.

Once you build and install the PEAK driver, you should set up the driver module:

/sbin/modprobe pcan

You can check that the driver has been successfully installed with the command cat /proc/pcan, you should get on standard output the following information:

*------------ PEAK-Systems CAN interfaces (http://www.peak-system.com) ------------
*------------------- Release_20120726_n (7.7.0) -------------------
*------------------- [mod] [isa] [pci] [dng] [par] [usb] [pcc] [net] -------------------
*------------------- 1 interfaces @ major 249 found -------------------
* n -type- ndev --base-- irq --btr- --read-- --write-- --irqs-- --errors-- status
32 usb can0 ffffffff 255 0x001c 00000000 00000000 00000000 0x0000

On this settings, you can check that the CAN driver is set such as:

- it uses the /dev/pcan32 device, as n field is set to 32;
<table>
<thead>
<tr>
<th>Hexadecimal value</th>
<th>CAN Baud Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0014</td>
<td>1MBit/s</td>
</tr>
<tr>
<td>0x0016</td>
<td>800 kBit/s</td>
</tr>
<tr>
<td>0x001C</td>
<td>500 kBit/s</td>
</tr>
<tr>
<td>0x011C</td>
<td>250 kBit/s</td>
</tr>
<tr>
<td>0x031C</td>
<td>125 kBit/s</td>
</tr>
<tr>
<td>0x432F</td>
<td>100 kBit/s</td>
</tr>
<tr>
<td>0x472F</td>
<td>50 kBit/s</td>
</tr>
<tr>
<td>0x532F</td>
<td>20 kBit/s</td>
</tr>
<tr>
<td>0x672F</td>
<td>10 kBit/s</td>
</tr>
<tr>
<td>0x7F7F</td>
<td>5 kBit/s</td>
</tr>
</tbody>
</table>

Table 1: CAN Baud rate values

- it supports netdev (i.e socketcan) interface, as [net] field is listed and ndev field is set to can0;
- the CAN baud rate is set to 500kBaud, as btr field is set to 0x001c.

If you need to change the baud rate, once you install the driver, you can:
- use the ip command to bring up the CAN device and change its bit rate; for instance to change the baud rate to 250kBaud, use:

  ip link set can0 up type can bitrate 250000

- (recommended) modify the file /etc/modprobe.d/pcan.conf such as:

  install pcan /sbin/modprobe --ignore-install pcan bitrate=0x011c

Once the CAN driver is well set up (via /sbin/modprobe or insmod), you can bring the CAN device up using:

  sudo ifconfig can0 up

You can retrieve socketCAN statistics using:

  cat /proc/net/can/stats

As a reminder, the table provides the hexadecimal values related to CAN baud rate values.

2.3 Testing the driver

2.3.1 CAN plug device

The CAN DB9 plug pin out is given on table.

To test the driver, we will need to build a small cable (let’s say less than 10cm) to connect CAN0 and CAN1 interfaces to each other. This cable is composed of two female DB9 plugs where pin 2, 3 and 7 are linked. The schematic is given on figure 1.
Software installation tips for the Cycab platform

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CAN_L</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CAN Ground</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Shield</td>
<td>✓</td>
</tr>
<tr>
<td>6</td>
<td>Ground</td>
<td>✓</td>
</tr>
<tr>
<td>7</td>
<td>CAN_H</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Power supply (V+)</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 2: CAN DB9 plug pin out

![CAN DB9 plug pin out](image)

Legend:
- CAN_L
- CAN_GND
- CAN_H

Figure 1: Schematic of the CAN test cable

2.3.2 For a Linux driver using SOCKET-CAN interface

We recommend to install and use the SocketCAN Tool package, can-utils, from gitorious (previously hosted on Berlios project).

To do so, follow these steps:

1. Retrieve the sources:

```bash
git clone git://gitorious.org/linux-can/can-utils.git
```

RT n° ????
2. Build and install these tools:
   ```
cd can-utils
make
sudo make install
   ```
3. Set the network up:
   ```
sudo ifconfig can0 up
   ```
4. Test the communication with `candump` to dump traffic from the CAN bus and `cansend` to send CAN-frames via CAN_RAW sockets; open a console and type:
   ```
candump can0
   ```
   On another console, type:
   ```
cansend can0 5A1#11.2233.44556677.88
   ```
   You should see on the first console, these lines:
   ```
can0 5A1 [8] 11 22 33 44 55 66 77 88
   ```
   Where 5A1 is the CAN id of the raw message, [8] the size in bytes of the message and the rest 11.2233.44556677.88 is the content of the message.

`can-utils` package provides tools like: `canbusload`, `can-calc-bit-timing`, `candump`, `canfdtest`, `cangen`, `cangw`, `canlogserver`, `canplayer`, `cansend`, `cansniffer`.

### 3 MPC555 programming

In this section we present some guidelines on how to build a cross-compiler based on GCC for MPC555 target. Important, this Howto has been tested successfully with `gcc v4.4.4`.

#### 3.1 Required tools

1. `powerpc-linux-gcc`: GCC Cross-compiler to produce MPC555 executives (ELF format).
2. `elf2bin`: Tool to convert ELF file format into loadable binary file (see subsection 3.2) via CAN bus.

#### 3.2 Build a cross-compiler for MPC555 targets

Make sure the `mpc555.1d` file you use sets the following variables:

```
SDA2_BASE_ = .;
   just before *(.sdata2) section and
SDA_BASE_ = .;
   just before *(.data) section
```

Make sure these tools are installed on your system: `bison`, `flex`, `m4`, `makeinfo` (texinfo package), `mpfr` (libmpfr-dev package).
Software installation tips for the Cycab platform

References about building a cross-compiler
Follow these links to get more information about how to build a cross-compiler:

- [http://www.kegel.com/crosstool/](http://www.kegel.com/crosstool/)
- [http://ecos.sourceware.org/build-toolchain.htm](http://ecos.sourceware.org/build-toolchain.htm)

Steps to build a GCC cross-compiler for MPC555 target

1. Packages to retrieve
   - binutils-2.20.1
   - gcc-4.4.4
   - newlib-1.18.0

2. Prepare the generation
   - Check that on your host a GCC compiler is installed
   - Set sh as shell (important to build GCC)
   - Set these variables:
     - Work directory CROSS_PRJ:
       ```
       export CROSS_PRJ=$HOME/MyDirectoryForMPC555
       ```
     - Directory where cross-compiler tools will be installed PREFIX:
       ```
       export PREFIX=$CROSS_PRJ/tools
       ```
     - Set PATH variable to include cross-compiler tools directory:
       ```
       export PATH=${PREFIX}/bin:${PATH}
       ```

3. Build binutils Package

   ```
   cd $CROSS_PRJ
   tar xvfz /DIRECTORYXX/binutils-2.20.1.tar.gz
   mkdir build-binutils
   cd build-binutils
   CC=gcc
   ../binutils-2.20.1/configure --target=powerpc-linux
   --prefix=$PREFIX
   --with-newlib
   --disable-werror
   make all
   make install
   ```

4. Build cross-compiler GCC Package

   ```
   cd $CROSS_PRJ
   tar xvfz /DIRECTORYXXX/gcc-4.4.4.tar.gz
   mkdir build-gcc
   cd build-gcc
   ../gcc-4.4.4/configure --target=powerpc-linux
   --prefix=$PREFIX
   make all
   make install
   ```

RT n° ????
5. Build newlib Package

    cd $CROSS_PRJ<br>tar xvfz /DIRECTORYXXX/newlib-1.18.0.tar.gz
    mkdir build-newlib
    cd build-newlib
    ../newlib-1.18.0/configure --target=powerpc-linux
       --prefix=$PREFIX
    make all
    make install

    May be you will get an error because of makeinfo command.
To correct that, check where is your makeinfo command, edit Makefile and change line containing MAKEINFO = path-to-build-newlib/missing makeinfo

Test the cross-compiler A way to check the powerpc-linux-gcc cross-compiler is to launch:

    powerpc-linux-gcc --version

And check that the result is : 4.4.4.

4 Boot-loading procedure

When you switch on the Cycab, both MPC555 nodes (front and rear) wait to be boot-loaded.
The files to be boot-loaded require a special format defined by Robosoft, and need to be sent
through the CAN bus following a particular boot-loader protocol, also defined by Robosoft.

4.1 Boot-loader binary format

It is not possible to send directly the binary files generated using the cross-compiler powerpc-gcc-linux
tool. The files to be downloaded to the nodes through CAN bus need to be re-written in a special
format; the elf2bin tool is used to convert a MPC555 binary executable into this format :

- 4 prefix bytes: containing the size of this binary file minus the 4 bytes of this heading. This
  value is written in BIG ENDIAN.
- Sequence encoding ONE complete EXECUTIVE:
  - One or more section blocks formatted as follow:
    - 4 bytes: containing the address of the section in memory. This value is written in BIG ENDIAN.
    - 4 bytes: containing the size of the section. This value is written in BIG ENDIAN.
    - The section code.
    - 4 bytes: containing the START address of the program. This value is written in BIG ENDIAN.
    - 4 bytes: containing only zeros aiming to indicate the end of the binary.
4.2 Sending protocol

Here is the boot-loader protocol defined by Robosoft to send a binary file converted with `elf2bin` to a MPC555 node.

1. Each node waits for a CAN message containing:
   - CAN ID : 0x02A8
   - CAN message : 4 bytes for MPC555 node ID, e.g. 0x4000 + 4 bytes for the size of the binary file to be downloaded on the node (see subsection 4.1)

2. Once this message has been received by one node, this node sends back a message containing only the CAN ID: 0x02A8

3. Complete executive file can then be sent frame by frame, each frame being 8 bytes long (or less) and being sent as content of a CAN message which ID is 0x02A8.

5 CycabHDK package

The Cycab Hardware toolkit (CycabHDK) provides tools and source code to control Cycab platform and to handle the downloading of MPC555 nodes:

- the `elf2bin` tool in the `Utils` directory;
- the `downloader` program in the `Downloader` directory;
- the source code to be compiled to get binary files for the MPC555 nodes in the `MPC555Ctrl` directory. This directory provides also a shell script, `downloaderMPC555.sh`, to make easy the downloading phase on both front and rear nodes given the nodes ID and the converted binary files for each node; it calls the `downloader` program;
- the source code for the Cycab communication handling through CAN into `CANCtrl` directory.

These sources are licensed under GPL.

5.1 Installation and Compilation

CycabHDK package can be retrieved via:

`svn checkout svn+ssh://yourlogin@scm.gforge.inria.fr/svnroot/cycabhdk/trunk`

This package depends on:

- Peak or Adlink driver development interface (see section 2 on how to install the CAN driver)
- `powerpc-linux-gcc` cross-compiler to generate binaries from C source to MPC555 nodes (see section 3 on how to generate a cross-compiler);
- `libelf-dev` and `libsigc++-dev` libraries.
The directories of interests are:

**Utils** to generate **elf2bin** tool. **Makefile** and **CMakeLists.txt** are provided.

**MPC555Ctrl** to generate the MPC555 binaries for front and rear nodes. **Makefile** and **CMakeLists.txt** are provided.

**CANCtrl** to generate the library to ease the communication through CAN bus. **Makefile**, **CMakeLists.txt** and **README** files are provided.

**Downloader** to generate the tool to transfer the MPC555 binaries to the front and rear nodes through CAN bus. **Makefile**, **CMakeLists.txt** and **README** files are provided.

**OpenCycab** to get documentation on the package.

In the root directory, a **README** and global **CMakeLists.txt** files are provided.

Once checked out, you can compile the whole package via **cmake** tool, on root directory you can type:

```
mkdir build
cd build; cmake ..
```

By default, the **cmake** compilation process is configured such as:

**CANCtrl** PEAK driver supporting net_dev (ie socket can) is used and **CAN_RATE** is set to 500kbps

**Downloader** PEAK driver supporting net_dev (ie socket can) is used and **CAN_RATE** is set to 500kbps

**MPC555Ctrl** INRIA_RA_GRIS configuration is set, in particular **CAN_RATE** is set to 500kbps, rear node address is set to 0x4000; front node address is set to 0x4001.

You can change this configuration. See **Makefile** or **CMakeLists.txt** set on each directory.
Appendices

A Obsolete

This section provides information for those who still want to use the original embedded PC without losing Robosoft original settings.

But beware: this PC is now obsolete with regards to its configuration.

A.1 Making a disk image of Robosoft installation

In order to prevent the installation from scratch of the embedded PC after a hard drive crash, we decide to keep an image of the hard drive, containing the Linux 2.4.22-edeos kernel, a Red Hat 9.0 Linux distribution and the Robosoft related softwares (aka Syndex v6.0, m4 macros, RTAI 24.1.13, etc).

Note: on the Cycab embedded PC, the hard drive requires an IDE interface.

The image has been created using the `dd` command:

```
dd if=/dev/hda | ssh <login>@host "gzip -9 > $PATH/20060724-backup-rcycab-hda.dd.gz"
```

Beware, the hard drive size is 40Go.

The resulting image can be downloaded into a new hard drive, providing that the new hard drive geometry (cf head, cylinder number) is the same as the primary hard drive.

In our case, the configuration of the hard drive is such as:

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Toshiba</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>MK4032GAX</td>
</tr>
<tr>
<td>Size</td>
<td>40 Go</td>
</tr>
<tr>
<td>Geometry</td>
<td>Cyl 16383 - H 16 - S 63</td>
</tr>
</tbody>
</table>

Table 3: Hard drive configuration

A.2 Installing a new hard drive disk

Get a new hard drive which geometry is the same as the primary one and which size is large enough.

We test the installation using a Hitachi Deskstar 7K80 IDE.

Once plugged on the right IDE socket on the motherboard, we switch on the PC and we check in which device (`/dev/hdx`) the new IDE drive has been detected. To do so, use `dmesg` and get the right `/dev/hdx` value.

To install the image in the new hard device, use `dd` command:

```
gunzip -c $PATH/20060724-backup-rcycab-hda.dd.gz | dd of=/dev/hdb
```

Note: the command below assumes that the image is available on the PC you install the new hard drive.

To check your new hard drive and its contains, try to mount its file partitions:

`RT n° ???`
mount /dev/hdb1 /mnt/fs1 -> contains /boot
mount /dev/hdb3 /mnt/fs3 -> contains /home
mount /dev/hdb5 /mnt/fs5 -> contains /usr/local
mount /dev/hdb6 /mnt/fs6 -> contains /

Note : /dev/hdb2 should contain the swap partition and can not be mounted.

A.3 Installing Ubuntu Linux

We install this Linux distribution on the same hard drive disk containing the image of the dedicated Cycab Robosoft installation. As the hard drive disk size is 80 Go, and the related Robosoft image is 40 Go, there are 40 Go left, enough to put up another Linux distribution.

In order to install Ubuntu on a hard drive, but keeping existing file systems, we need to create new partitions and to update the existing partition table. To do so, we use the command `fdisk`.

First we retrieve the primary table using `fdisk -l` and we get the above output :

```
Disk /dev/hdb: 82.3 GB, 82348277760 bytes
255 heads, 63 sectors/track, 10011 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot Start End Blocks Id System
/dev/hdb1 * 1 63 506016 83 Linux
/dev/hdb2 64 126 506047+ 82 Linux swap / Solaris
/dev/hdb3 127 1343 9775552+ 83 Linux
/dev/hdb4 1344 10012 69630561+ 5 Extended
/dev/hdb5 1344 2560 9775521 83 Linux
/dev/hdb6 2561 4864 18506848+ 83 Linux
```

The idea is to rearrange the partitions to take into account the 40Go left and to modify the table. For that we need to modify the extended partition hdb4.

Steps to rearrange the partitions on an existing disk, using `fdisk` :

1. Keep the Start and End values for each partition.
2. Delete the partitions /dev/hdbi which $i \geq 4$. With `fdisk`, the partitions are deleted in descending order, i.e from /dev/hdb6 to /dev/hdb4.
3. Create a new partition such as configuration is :
   - Extended
   - Start value : the start block value of the previous partition /dev/hdb4, i.e 1344
   - End value : default value given by `fdisk`. This value should be the correct one , i.e value corresponding for the end block of a 80 Go disk.
4. Create a new partition (/dev/hdb5) such as :
   - Start value : the start block value of the previous partition /dev/hdb5, i.e 1344.
   - End value : the end block value of the previous partition /dev/hdb5, i.e 2560.
5. Create a new partition (/dev/hdb6), such as :
   - Start value : the start block value of the previous partition /dev/hdb6, i.e 2561.
• End value: the end block value of the previous partition /dev/hdb6, i.e. 4864.

Once this has been performed, the 40 Go left are now available and can be used to put up a Linux distribution.

At the time we made the changes (2006), we chose Ubuntu 6.06 LTS Linux distribution (Dapper Drake), which kernel Linux version is 2.6.15. Its installation was straightforward and detected previously installed distributions on hard disk. You can try with up to date Linux distribution.