jDinamoCC version 3 documentation

English
October 2015

jDinamoCC3 is the third version of an application that has these main functions:

- Editing configuration of the Dinamo MCC decoder (CV's)
- Manually controlling cars
- Reading detection switches
- Sending manual commands to Ox32.

In this documentation you will find a description how to use the tool, some tips and tricks about the decoder configuration and the technical details about the application.

The differences between this version and the 2.x versions are:

- Improved stability of the serial communication.
- Included native drivers for serial ports (no more copying of dll's to system directories)
- Removed support for the native ftdi driver.
- Support for Win32 and Win64 (MacOSX in development)
- Multi-language interface: English, Dutch and German.
- Keyboard support for the car control window.
- A slightly more modern look and feel.
- Supports the Dinamo 3.1 protocol.
- Added a new but experimental way of automagically mapping your layout.
- Various bugfixes.

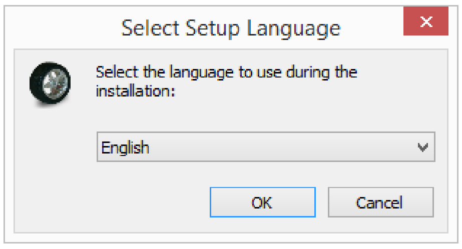
And most important:

- No more support for the obsolete version 1 decoder.
- Supports the Dinamo MCC versions 2 and 3 decoders.

This version has a slightly modified license agreement. It is still freely available for any hobby users. You can use it to demonstrate your cars at home or any show, trade show or club. However, if you apply this version professionally you will need a paid license. Please contact the author for a license based on a mutual reasonable agreement. Your fee will help to sustain the development of the application for everybody.

1 Installing the application

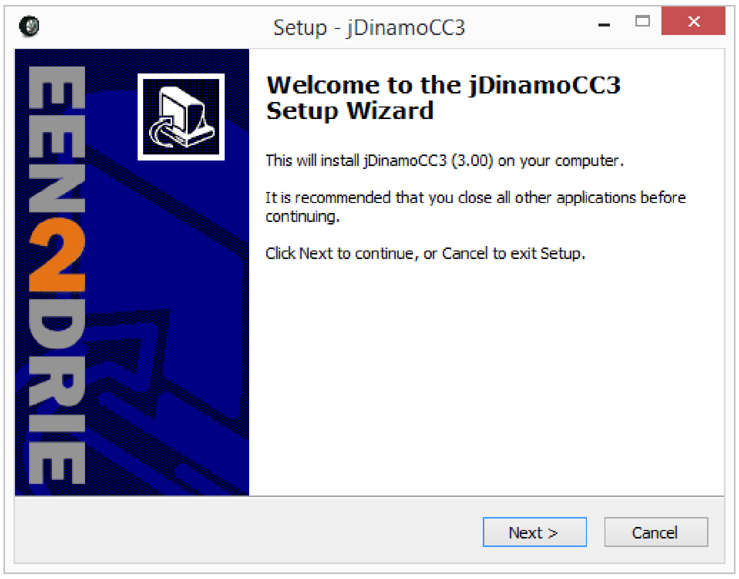
On the Windows platform the installation is just like any other application with an installer. Allow the installer to make changes and follow the steps.


Select Setup Language

Select the language to use during the installation:

English

This does not change the language of the application, it is only during installation.


Setup - jDinamoCC3

Welcome to the jDinamoCC3 Setup Wizard

This will install jDinamoCC3 (3.00) on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

etc…

2 Using the application

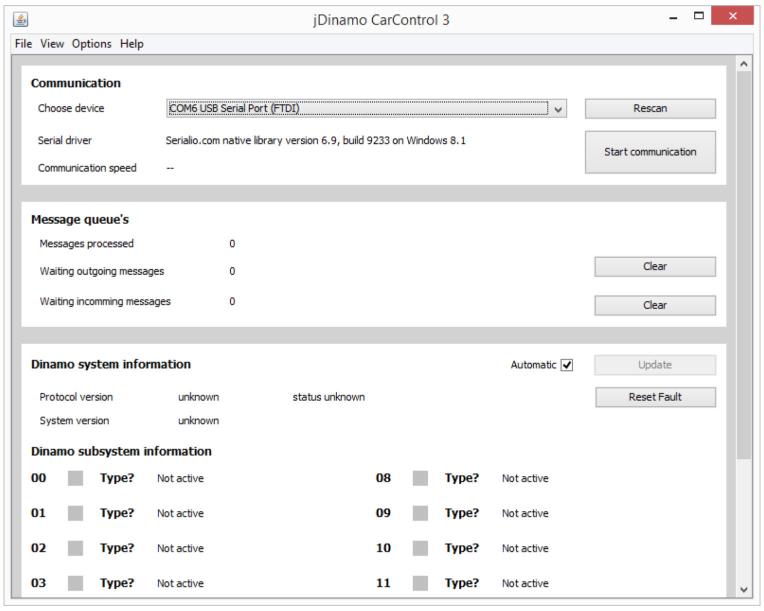
After the application has started you are welcomed by the first screen that allows you to select one of the detected serial communication ports.

In the main menu you can choose

Menu	Keyboard	
File -> Exit	ALT-F4	Exit the application. No real need to stop the communication before exiting. All your communication to your cars is stopped anyway.
View -> Serial Communication	ALT-1	Serial communication, message queue's and Dinamo system information.
View -> Car Control	ALT-2	Add your cars to a handy list, control the speed and functions of a number of cars. Change address wizard. Program the decoders.
View -> Switch detection	ALT-3	View detection switches being triggered by any magnet or your running cars. This includes the experimental function to automagically map your layout.
View -> Output commands	ALT-4	Manually send commands to OM32 or OC32 modules. Some knowledge of the protocol and functions is needed to use this view.
Options -> Set GUI style		Selects one of the available GUI styles. A modern Windows look and feel or the classic Java or Unix styles. The application is most extensively tested in the Windows style.
Options -> Select Language		Change the interface language to English, Dutch of German. For this to be effective you must restart the application.
Help -> About		The usual comments about the author, license and application version.

In the next chapters the four view windows are explained.

2.1 Main view Communication



Communication

Choose device	COM6 USB Serial Port (FTDI)
Serial driver	Serialio.com native library version 6.9, build 9233 on Windows 8.1
Communication speed	--

Message queue's

Messages processed	0
Waiting outgoing messages	0
Waiting incomming messages	0

Dinamo system information Automatic ✓

Protocol version	unknown	status unknown
System version	unknown	

Dinamo subsystem information

00	▨	Type?	Not active		08	▨	Type?	Not active
01	▨	Type?	Not active		09	▨	Type?	Not active
02	▨	Type?	Not active		10	▨	Type?	Not active
03	▨	Type?	Not active		11	▨	Type?	Not active

The first window block (white) is **Communication**. It shows the native driver version and allows for selecting one of the serial ports. If you are unsure which port to choose, the (FTDI) is a reasonable clue this is your UCCI or RM-U.

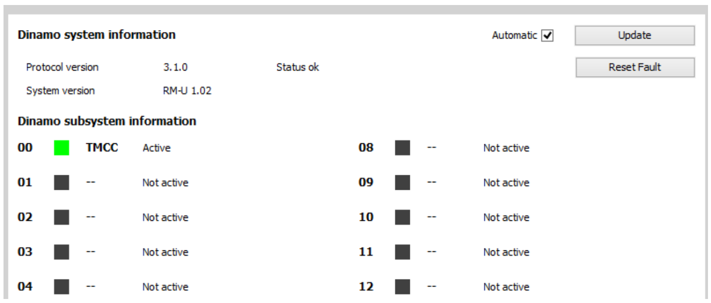
If you plug in your UCCI or RM-U after starting the application you can [Rescan] for serial devices.

Then you can hit [start communication]. After a few 100 messages the block will show the average communication speed. It should be around 100 messages per second.

The Message queue's block shows three counters. The number of processed messages during a serial session (between a start and stop communication). It shows the number of messages waiting to be send to the Dinamo hardware. If the communication is active this should be almost always zero. When communicating to the hardware with 100 messages per second any waiting messages are gone very soon.

Waiting incoming messages should be empty all the time. Otherwise there are unhandled or unknown responses from the Dinamo modules. You can clear the waiting messages with the appropriate buttons.

The third block is about **Dinamo System Information**. This shows the details about your protocol and firmware versions. It is repeatedly refreshed if the checkbox Automatic [ ] is checked.



Dinamo system information Automatic ✓

Protocol version	3.1.0	Status ok
System version	RM-U 1.02	

Dinamo subsystem information

00	🟩	TMCC	Active	08	■	--	Not active
01	■	--	Not active	09	■	--	Not active
02	■	--	Not active	10	■	--	Not active
03	■	--	Not active	11	■	--	Not active
04	■	--	Not active	12	■	--	Not active

When the communication is started the information is retrieved at least once. This picture shows my test setup versions and status.

With [Update] you can refresh the system information without having it done automatically.

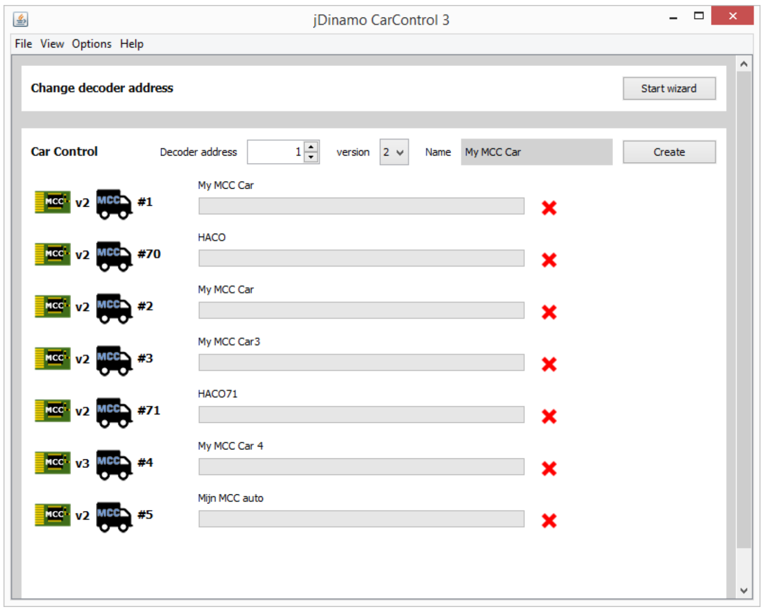
Sometimes the RM-U or UCCI has a fault status. Most of the time this is due to an unexpected cut off of the communication. With the [reset fault] button this state is reset.

If the fault status is repeatedly set during normal operations there is probably something wrong with your hardware or setup.

You can enlarge or scroll the window. There are 16 possible subsystems available.

2.2 Main view Car Control

The second main view (ALT-2) is to control and configure your cars.


jDinamo CarControl 3

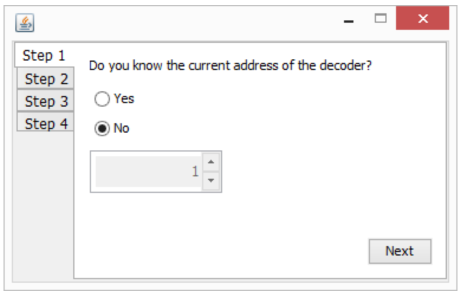
File View Options Help

Change decoder address [Start wizard]

Car Control Decoder address 1 version 2 Name My MCC Car [Create]

v2	#1	My MCC Car	✗
v2	#70	HACO	✗
v2	#2	My MCC Car	✗
v2	#3	My MCC Car3	✗
v2	#71	HACO71	✗
v3	#4	My MCC Car 4	✗
v2	#5	Mijn MCC auto	✗

The **Change decoder address** block has one simple [Start wizard] button to help you change the address of your decoder with four simple steps.

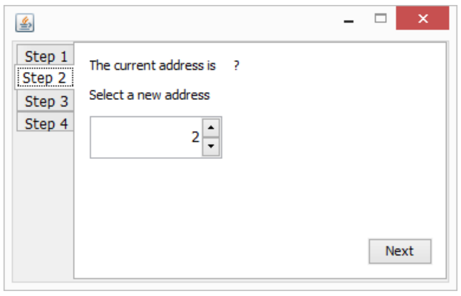

Step 1 / Step 2 / Step 3 / Step 4

Do you know the current address of the decoder?

Yes [✓ No]

1

[Next]

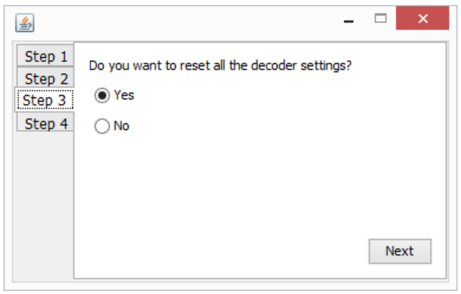

Step 1 / Step 2 / Step 3 / Step 4

The current address is ?

Select a new address

2

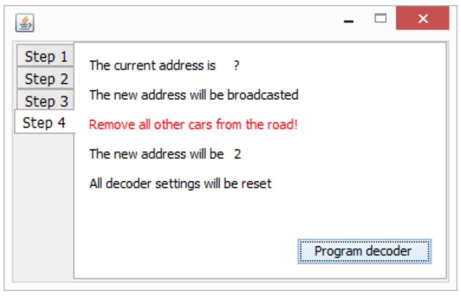
[Next]


Step 1 / Step 2 / Step 3 / Step 4

Do you want to reset all the decoder settings?

[✓ Yes] No

[Next]


Step 1 / Step 2 / Step 3 / Step 4

The current address is ?

The new address will be broadcasted

Remove all other cars from the road!

The new address will be 2

All decoder settings will be reset

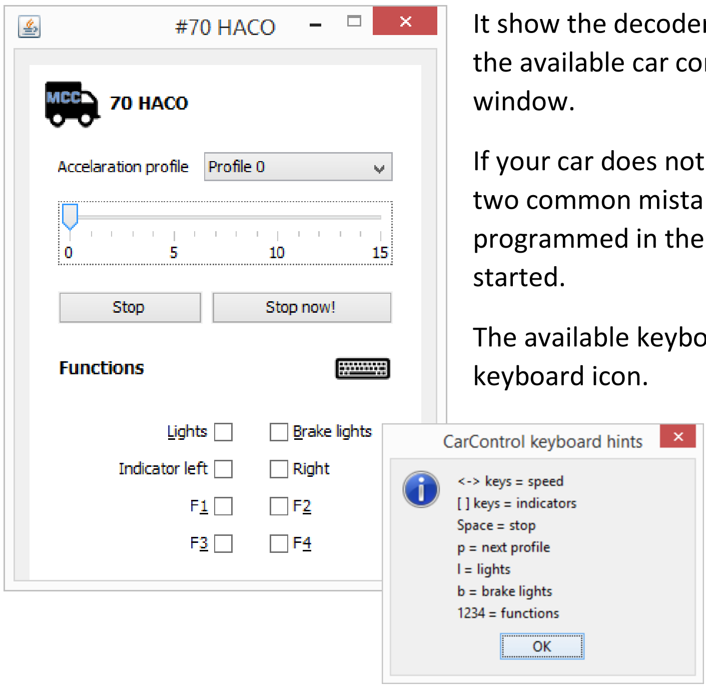
[Program decoder]

If you don’t know the current address of your car the programming commands will be broadcast to any car on the road. This means that all listening cars will have a new (and the same) address. The wizard warns for this situation. Before any sending programming messages, the car is stopped.

The second block **Car Control** allows you to create a list of your cars. Except for starting the following functions and windows the list has no configuration meaning. It’s just a handy list that is automatically saved and presented again when you restart the application. The list save function is limited to 16 cars. You can add more but they are not remembered.

Select the desired decoder address (between 1 and 4095), select the decoder version and add your car to the list. If you click on the red cross the car is stopped and removed from the list.

Click on the car symbol, the name or the speed indicator and you will open a control window.



<-> keys = speed
[ ] keys = indicators
Space = stop
p = next profile
l = lights
b = brake lights
1234 = functions

It show the decoder address and the name from the list. All the available car control functions are available from this window.

If your car does not respond it can be many things. There are two common mistakes: the selected address is not programmed in the decoder or the communication is not started.

The available keyboard keys are available if you click on the keyboard icon.

With your cursor keys you control the speed. [] Brackets for the blinkers or indicators. The P cycles through the acceleration Profiles. Etc.

If you use a mouse all functions will set the keyboard focus back on the speed slider.

Functions are send immediately to the car. [Stop] includes the acceleration profile, [Stop now!] ignores this setting.

Closing this window will stop the car and remove its address from the communication wire in the track. If needed you can open several windows to juggle the control of several cars.

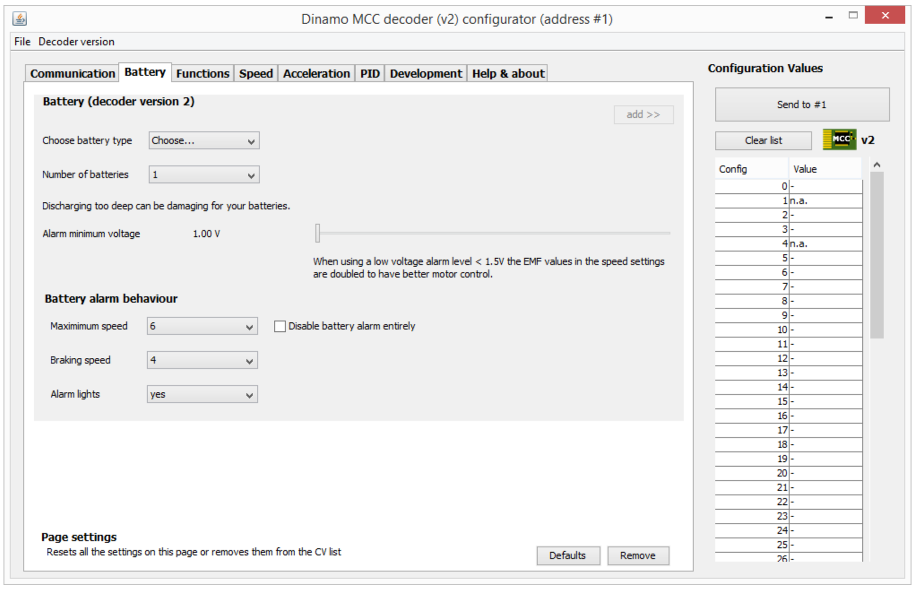
2.3 Programming a decoder

In the list you created of your cars, click on a decoder icon. A new window will appear that helps you to program your decoder. This application version only supports the version 2 and 3 of the decoder. The first version is sold in very limited numbers, early adopters will be able to use the previous application. The configuration window version (2 or 3) is created based on the version of the decoder when you added it to your list.

The configuration window allows the followings settings to be made:

- Communication
  factory defaults, communication timeout,
- Battery
  Different screens and options for version 2 or 3.
- Functions
  Lights and blinkers

- Speed
  a table of settings for each speed step.
- Acceleration
  8 different profiles can be given to brake or accelerate.
- PID
  Motor control for the cruise control
- Development
  Special page to test unreleased functions.



Dinamo MCC decoder (v2) configurator (address #1)

File Decoder version

Communication | Battery | Functions | Speed | Acceleration | PID | Development | Help & about

Battery (decoder version 2)

Choose battery type: Choose...

Number of batteries: 1

Discharging too deep can be damaging for your batteries.

Alarm minimum voltage 1.00 V

When using a low voltage alarm level < 1.5V the EMF values in the speed settings are doubled to have better motor control.

Battery alarm behaviour

Maximimum speed: 6 ☐ Disable battery alarm entirely

Braking speed: 4

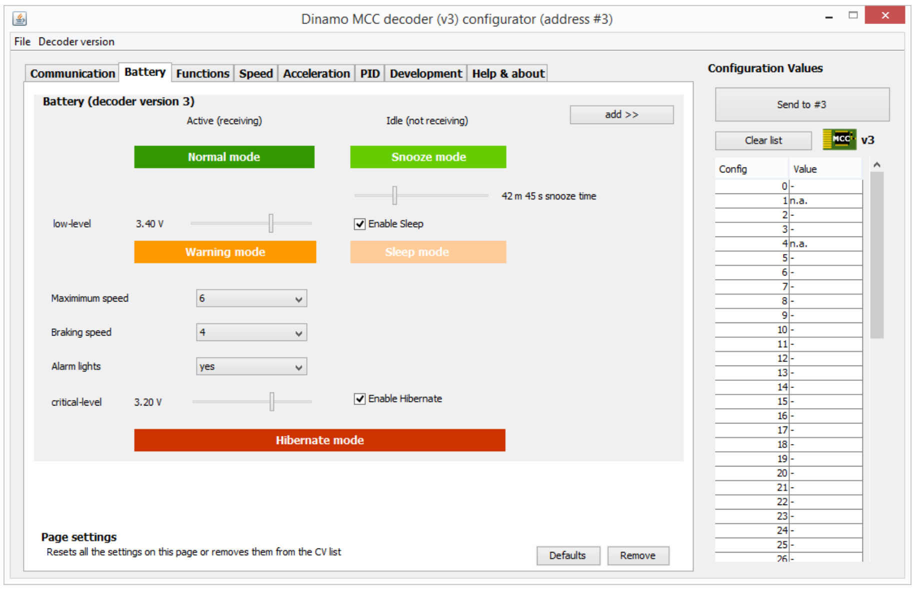
Alarm lights: yes

Page settings
Resets all the settings on this page or removes them from the CV list

Configuration Values

Config	Value
0	-
1	n.a.
2	-
3	-
4	n.a.
5	-
6	-
7	-
8	-
9	-
10	-
11	-
12	-
13	-
14	-
15	-
16	-
17	-
18	-
19	-
20	-
21	-
22	-
23	-
24	-
25	-
26	-

version 2 battery tab



Dinamo MCC decoder (v3) configurator (address #3)

File Decoder version

Communication | Battery | Functions | Speed | Acceleration | PID | Development | Help & about

Battery (decoder version 3)

Active (receiving) Idle (not receiving)

Normal mode Snooze mode

42 m 45 s snooze time

low-level 3.40 V ☑ Enable Sleep

Warning mode Sleep mode

Maximimum speed: 6

Braking speed: 4

Alarm lights: yes

critical-level 3.20 V ☑ Enable Hibernate

Hibernate mode

Page settings
Resets all the settings on this page or removes them from the CV list

Configuration Values

Config	Value
0	-
1	n.a.
2	-
3	-
4	n.a.
5	-
6	-
7	-
8	-
9	-
10	-
11	-
12	-
13	-
14	-
15	-
16	-
17	-
18	-
19	-
20	-
21	-
22	-
23	-
24	-
25	-
26	-

version 3 battery tab

On all the different screens you can make changes, add them to a CV list and send them to the car. Of course you must start the serial communication first, otherwise the commands are buffered.

All the settings you make can be saved to a file. There is no way to read settings from a car, this is one-way communication. If you read the settings from a file you still have to add them to the list and send them to the car. *[[In the first release of this version this save function is disabled]]*

You can always reset the decoder to the factory defaults. Only limitation on experimenting is determined by the hardware of the processor. It can “only” save settings about 100.000 times.

A factory default reset is done before setting the other values and can be send in one action. This command is automatically added to the CV list. The current decoder address is send after resetting so the decoder will keep its current address.

A change between the previous versions of the application is that the changing of an address is changed to the main car control view. You will not find it here anymore.

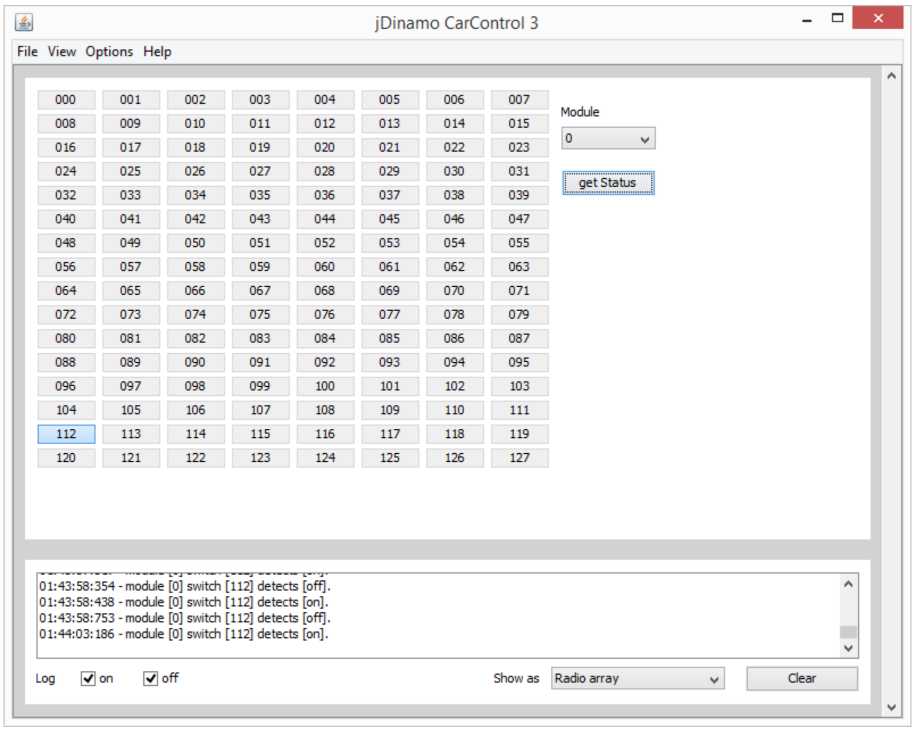
A car receiving new configuration settings stops immediately and confirms each setting with a short blinking indicator. The more commands you send, the longer the indicators light up. Look at your car to verify that it is receiving commands.

2.4 Main view Switch detection

In this view you can test the detection switches. You can trigger them manually with a magnet or you can drive your car and let the car trigger the detection.

Here you see a single switch in detection state. On the bottom of the screen you can tweak the logging by enabling on or off events. You can also clear the log. After 10.000+ lines it will clear itself.

On the top of the window you can choose different modules and retrieve the status of all the shown switches. This sends 128 messages to your controller. If the chosen module does not exists it will take some time to handle this.

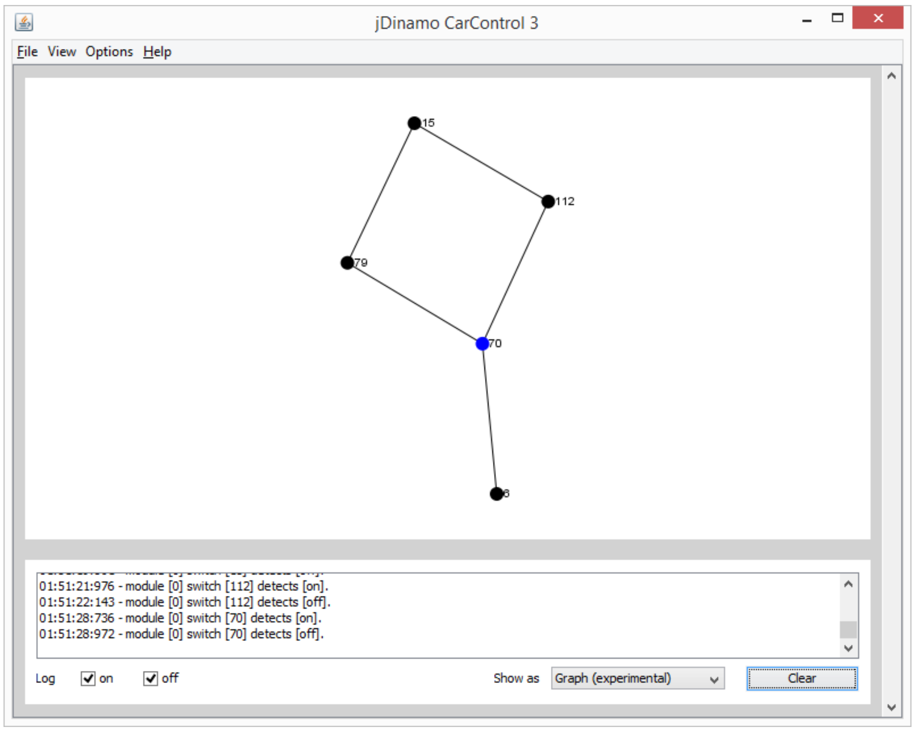


Module: 0

```
01:43:58:354 - module [0] switch [112] detects [off].
01:43:58:438 - module [0] switch [112] detects [on].
01:43:58:753 - module [0] switch [112] detects [off].
01:44:03:186 - module [0] switch [112] detects [on].
```

Log ☑ on ☑ off Show as: Radio array

An experimental screen is shown by choosing Graph instead of the Radio array.

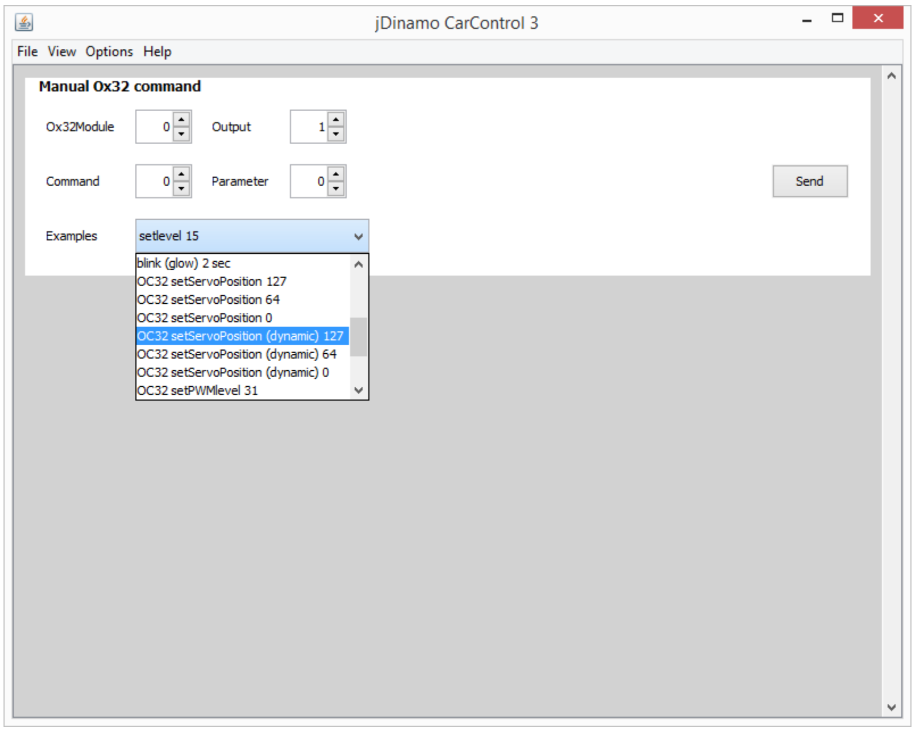


```
01:51:21:976 - module [0] switch [112] detects [on].
01:51:22:143 - module [0] switch [112] detects [off].
01:51:28:736 - module [0] switch [70] detects [on].
01:51:28:972 - module [0] switch [70] detects [off].
```

This is the incomplete result of a car driving on my test track. The software remembers which previous switch was detected and tries to create a visualization of your layout. With your mouse pointer you can jiggle the points around to improve the layout. Again, this is experimental and is known to fail after a while.

2.5 Main view Output commands

This window allows you to send command to an OM32 or OC32 module. You sort of have to know what you are doing. The details of the commands can be found in the manuals.



Ox32Module 0 Output 1

Command 0 Parameter 0

Examples setlevel 15

The examples help you to set commonly used commands and parameters. After changing the preferred parameter you will have to send it used the button. Of course you will have to start the serial communication first, otherwise the commands are buffered until you do start the communication.

3 Starting the application

3.1 Installation

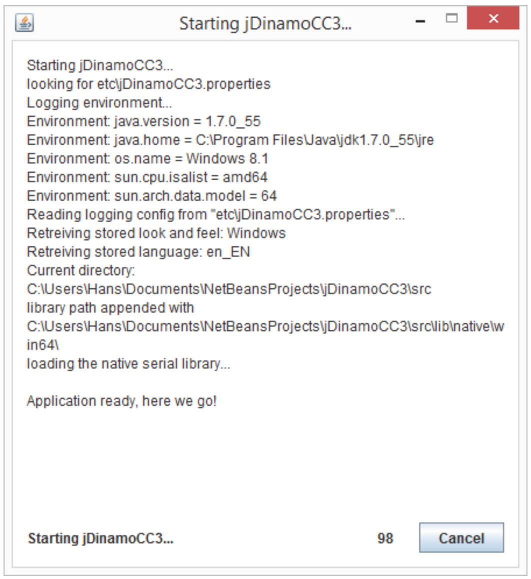
There are two distributions made for the Windows platform. They are quite different in size. The smaller one is about 2.5 Mb, the larger one, including the Java is about 34 Mb. If in doubt, choose the larger one. It is self-supporting and independent of any other Java installations.

On the Windows platform during installation the application and support files are places in a folder called `C:\Users\{username}\AppData\Roaming\jDinamoCC3`. Later in this documentation we refer to this as `{jDinamoCC3}`

The installer also creates some icons to start the application, a link to the website and one to start the uninstaller. This uninstaller removes these files except for any config or logfiles the application creates.

3.2 Starting

As a user simply click on the icon that is created by the installer



```
Starting jDinamoCC3...
looking for etc\jDinamoCC3.properties
Logging environment...
Environment: java.version = 1.7.0_55
Environment: java.home = C:\Program Files\Java\jdk1.7.0_55\jre
Environment: os.name = Windows 8.1
Environment: sun.cpu.isalist = amd64
Environment: sun.arch.data.model = 64
Reading logging config from "etc\jDinamoCC3.properties"...
Retreiving stored look and feel: Windows
Retreiving stored language: en_EN
Current directory:
C:\Users\Hans\Documents\NetBeansProjects\jDinamoCC3\src
library path appended with
C:\Users\Hans\Documents\NetBeansProjects\jDinamoCC3\src\lib\native\w
in64\
loading the native serial library...

Application ready, here we go!
```

The starting of the application has been technically the most challenging part. This version 3 tries to find out your OS type, 32 or 64 bit, your Java version and home etc. All details needed to kick start the application. During the start as much as possible information is shown. This helps us to find out what is happening if the application somehow fails to start.

3.3 Logfiles

During the running of the application logfiles are created. Like the startup screen they potentially help us to find any issues concerning the application not starting or improperly functioning. Log files can be found in a folder called `{jDinamoCC3}\log`
If the log files or folder do not exist, they are created during the startup. The default setting allows for three log files of maximum 1 Mb. If they become larger they are automatically removed. You don't have to worry about this.

The log contains lines like this

```
31      [INFO ] Splash                    main          - Reading logging config from "etc\jDinamoCC3.properties"...
33      [INFO ] Splash                    main          - Retreiving stored look and feel: Windows
33      [INFO ] Splash                    main          - Retreiving stored language: de_DE
41      [INFO ] Splash                    main          - Current directory: C:\Users\Hans\AppData\Roaming\jDinamoCC3\app
51      [INFO ] Splash                    main          - library path appended with
51      [INFO ] Splash                    main          - C:\Users\Hans\AppData\Roaming\jDinamoCC3\app\lib\native\win64\
52      [INFO ] Splash                    main          - loading the native serial library...
84      [INFO ] Splash                    main          - Application ready, here we go!
862     [INFO ] dinamoRMH                 AWT-EventQueue-0  - RMH inits dinamo RM-H
1020    [INFO ] MainComm                  AWT-EventQueue-0  - Available serial port: COM5 Standard Serial (FTDI)
1404    [INFO ] JDinamoCC3                AWT-EventQueue-0  - Stored look and feel: Windows
```

In case of some issues you may be asked to send the log files that are created. Except for your user name and Windows version information they contain no privacy information.

In some cases we may ask you to increase the amount of information in the log files. This is called debug logging and is explained in the next section.

3.4 Application configuration

For normal use you do not have to edit the configuration file. This is only in case of advanced trouble shooting. If the configuration file or folder do not exist, a default one is created during the startup.

The configuration file is called `{jDinamoCC3}\etc\jDinamoCC3.properties`

The application allows for selecting a display language, a look-and-feel and saves a list of cars. These settings are saved in the configuration file.

Next to this the configuration file contains settings for the logger. Unfortunately the lines can and will be not sorted like this, they are randomly placed in the file.

```
#jDinamoCC3 properties
#Sat Sep 26 01:20:33 CEST 2015
jDinamoCC3.lang=de_DE
jDinamoCC3.lookandfeel=Windows
jDinamoCC3.CarControlItem.1=2//2//My second MCC Car
jDinamoCC3.CarControlItem.0=1//2//My MCC Car

log4j.rootLogger=info, stdout, R

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.R.File=log/jdinamocc3.log
log4j.appender.R.MaxFileSize=1MB
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.MaxBackupIndex=2
log4j.appender.R.layout.ConversionPattern=%-6r [%-5p] %-25c %-12t %x - %m%n
log4j.appender.stdout.layout.ConversionPattern=%-6r [%-5p] %-25c %-12t %x - %m%n
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.R=org.apache.log4j.RollingFileAppender
```

If asked, debug logging is enabled by changing this line:

```
log4j.rootLogger=info, stdout, R
```

to

```
log4j.rootLogger=debug, stdout, R
```

You will have to restart the application until it stops working unexpectedly again, quit and send the log files.