VPEB Van Perlo Elektronica & Besturingstechniek

# OC32 3.0

## Configuration Manual

Smart electronics for your miniature world

Author: Leon J.A. van Perlo
Version: 3.01
Date: December 15th, 2017

# Release management

This manual applies to

- Print
  - OC32 Rev 00
  - OC32 Rev 01
  - OC32 Rev 02
  - OC32 Rev 03
  - OC32 Rev 04
  - OC32/NG Rev10

- Firmware
  - OC32 Rel 3.01

- Software
  - OC32 Config Rel 3.01

With the introduction of the OC32/NG, the "hardware" (mounting and connecting the modules) and "software" (configuration) are described in separate manuals. The reason is that firmware and software are universal for all above mentioned OC32 modules and evolves regularly. For a complete overview of all functionality you therefore should consult both this configuration manual and the hardware manual of the respective product.

The Dutch version of this OC32 Configuration Manual also applies to the use of the OC32 in conjunction with the OM32. Since the OM32 has not been sold outside the Netherlands and Belgium this topic is not covered in this manual.

History

- Manual 3.0.1 (2017-12-15)
  - Extracted from manual OC32 3.0
  - Update to OC32 Rel 3.01
  - Replaced illustrations
  - Synchronised with Dutch version 3.01

# Preface / Reading Guide

The OC32 is a product with many possibilities. These extensive capabilities make the module very attractive: in fact you can use the OC32 to control (almost) any type of accessory on your miniature world (so basically everything except the trains and cars themselves).
Without the need to buy other specific electronics, the OC32 can do it all.

This versatility has a downside: Beginners, electronically less savvy users, face the risk of losing the overview at first. Therefore, this guide attempts to structure information with the above in mind. As a reading-aid you find a colored bar in the margin, and the black&white spectators will note that the bars have a different width:

| | |
|---|---|
| Green | Novice: With these sections you should be able to get the basic functions working. It offers no extensive choices, clever savings or complex combinations. |
| Blue | Advanced level; Requires basic knowledge of electronics, some user-level experience with PC software, some logic thinking or a combination hereof. It requires you to make some choices and therefore you should be able to judge the benefits and drawbacks in your specific situation. In principle everyone should be able to practice this, however it may not be wise for everyone to start with this immediately. |
| Orange | Expert level: Requires reasonable to good knowledje of electronics, logic thinking capabilities, some programming skills or a combination of these. What is described in these sections can lead to damage to the electronics or other devices if it is not done correctly. So practice only if you fully understand what you are doing. |

Should you consider yourself a “novice” and electronically limited skilled, or just looking for the easiest start, skip the blue and orange marked sections at first. If the basics work you can always start the more advanced levels later.

Some more complex subjects are not handled in this manual because of size and readability, but are covered in additional manuals. Please consult these additional manuals as well if you miss anything.

The OC32 is supported through the Dinamo Users Portal. You find the portal at
http://www.dinamousers.net
The portal contains a “wiki” with quite some additional information, such as:

- Answers to Frequently Asked Questions
- Software and firmware updates
- A forum you can use for advice and to get your questions answered.

We urgently request you to use our support channels in the above order before personally contacting the VPEB partners or VPEB.
Or course the latter does not apply to matters of more individual nature, such as warranty and orders.

**Enjoy!**

# Contents

# 1 Introduction

## 1.1 Hardware, firmware, software and configuration

A computer needs software to function. Depending on what you want to do at a specific moment with your computer, you start the software selected for the specific task you want to perform.

The OC32 is a microcomputer optimised for one specific task: to control electr(on)ic accessories in your Miniature World. Just like a computer, the OC32 cannot work without software. However, the OC32 has only one program, that is more or less permanently glued in the OC32. For that reason we call it "firmware" instead of software. The firmware in the OC32 contains all possible functions you may reasonably need to control your devices connected to the OC32. What functions you use for which device is determined by you by means of configuration

The OC32 firmware is regularly updated and upgraded by VPEB, e.g. to add new functionality or to make sure that the OC32 stays aligned with other, connected products when those evolve. When new firmware for the OC32 is released by VPEB you can install this yourself in your OC32('s). This is done though the "bootloader".

Although there is only one program in your OC32, the variety in which you can use the module is almost infinite. Every miniature world is different. How you use the OC32 is determined by configuration. The configuration of the OC32 is stored in a separate memory block in the OC32 (the configuration memory).

The configuration of your OC32 is made with the help of a configuration program running on your PC. The standard program made available by VPEB is OC32Config. Besides OC32Config others may develop and publish software that allows you to configured, tested and controlled. The choice of software is up to you as a user, only this manual only describes the use of OC32Config.

The illustration below shows how the above mentioned components fit together.

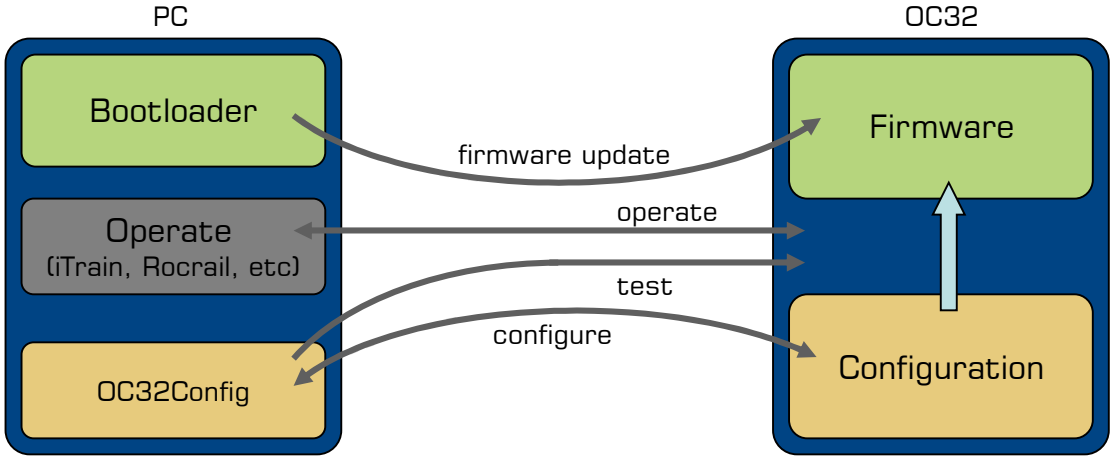


Fig. 1: Interoperation functions PC and OC32

## 1.2 Versions

The firmware operating your OC32 has a version number. That version number, or "release" determines which set of features is offered by your OC32. In order to make the configuration for your OC32, the OC32Config configuration program needs to be aware of the maximum functionality your OC32 offers. If a new version OC32 firmware is released, often new features are added and therefore at the same time a new version OC32Config is released.

If you run a newer version OC32 firmware than your OC32Config software, your OC32 offers functions of which OC32Config is not aware yet. Therefore you won't be able to configure, thus not use these new functions. If, on the other hand, you run newer OC32Config software than firmware in your OC32, OC32Config may "think" your OC32 has new features and may allow you to configure these, however, they won't work for the simple reason that your OC32 can't perform them yet

Therefore it makes a lot of sense to keep the version numbers of OC32Config and your OC32 firmware identical. If you install new OC32 firmware, do that in all your OC32's and subsequently install/update OC32Config.

## 1.3 Connection PC - OC32

To connect your PC to the OC32, preferably use the RS485. If that is not possible, use the RS232 interface. In the latter case it will not be possible to read back nor verify your OC32 configuration.
A direct connection between your PC and OC32 via RS485 can be realized by using the U485 interface. Details are described in the OC32 hardware manuals.
All functions can also be performed when your OC32's are connected 'behind' a Dinamo RM-C, RM-U, RM-U P&P or a Dinamo/MCC UCCI controller.

## 1.4 Release Notes

As written above, the OC32 firmware will be regularly upgraded or adopted to new developments elsewhere. We will not update this manual after every change.
At the DinamoUsers portal (http://www.dinamousers.net). you will find "release notes" that describe what the changes are from any version to the next. Also you may find points of attention to bear in mind when you update to a newer version. In rare cases functionality can be removed, e.q. when nobody uses the specific features or because the function is superseded already for quite some time by something that works far better.

Consider the release notes as an addendum to this manual and consult them.

# 2 Configuring the OC32 with OC32Config

## 2.1 Installation of OC32Config and Device Definitions

The OC32 configuration tool consists of two parts:

- The OC32 configuration program
- The OC32 device definitions

The OC32 configuration program needs to be installed on a PC that can connect to the OC32 module(s) via RS485 (preferred) or RS232. The connection can also be made via a Dinamo RM-C, RM-U, RM-U P&P or UCCI(/E) controller. The software distribution consists of a number of files. By running the Setup.exe file and following the instructions OC32Config is installed on your PC. You can then run OC32Config via Start → Programs(x86) → OC32 → OC32Config.

In addition to the OC32Config configuration program, you need a set of Device Definitions, that is, assuming you don't want to configure everything manually. A basic Device Definition Set is installed along with OC32Config, however, this is just a **limited basic set**. Additional definitions have to be downloaded separately, depending on your needs and theme. Device Definitions can be updated regularly and this way (updates of) Device Definitions and the OC32Config program can be kept almost completely independent.

After OC32Config is started you have the option to select the Device Definition Set of your choice, depending on your requirements at that time.

The default name of the device definition file is "OC32Devices.def". The default location is the location where the program OC32Config.exe is installed, usually "C:\Program Files \OC32Config" (on a 64 bit PC it will usually be "C:\Program Files (x86)\OC32Config"). If the definition file has the default name and is placed in the default location, it will be loaded automatically when OC32Config is started.

If you want another Device Definition Set to be loaded at OC32COnfig startup, you can simply replace the standard "OC32Devices.def" by the file of your choice, give it the default name "OC32Devices.def" and make sure it's in the default location. You ca also choose not to load a default definition file at all. By removing the standard definition file from it's default location, OC32Config displays a warning message and prompts you to select the definition file of your choice. If you make no selection you'll receive a warning that no definition file is loaded.

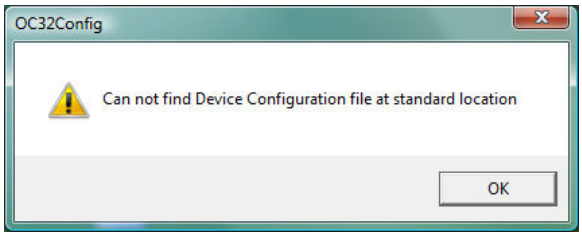

Fig 2: Warning when loading OC32Devices fails

## 2.2 Some principles and terminology of OC32 Config

After you have started OC32Config it is loaded in your PC's "working memory". All settings you change in OC32Config are also in "working memory" until you explicitly instruct the program to do something with it. Working memory is temporair. When you close the program without saving your settings, your changes are lost forever. If you forget to save before you close this may lead to loss of data. On the other hand, it can ease your mind: If you completely mess-up things in OC32Config, just close the program and restart and no harm will be done.

The usual intent is that your settings end up in an OC32 module. What you have stored in an OC32 is (semi) permanent: It is remembered by the module until you change it, even when the module is switched off. Transferring settings from OC32Config's working memory to the

OC32 is done by buttons called "Write". Transferring settings from the OC32 into OC32Config's working memory is done by buttons called "Read".
You'll also find buttons "Verify". "Verify" never changes anything, but compares settings in the OC32 working memory with the settings stored in the OC32 module and reports if they are equal or different.
**"Read" and "Write" move information between OC32Config and the OC32.**
**"Verify" compares information in OC32Config with informatie in the OC32.**

Also it is possible to store settings to your PC's harddisk. "Harddisk" refers to everything in or around your PC that can store data and keep it when your PC is turned off, so includes network-locations, SD cards, USB sticks, etc. Storing settings from OC32Config's working memory to your PCs harddisk is done by buttons called "Save". Retrieving information from harrdisk to OC32Config is done by buttons called "Load".
**"Load" and "Save" move information between OC32Config and your PC's harddisk.**

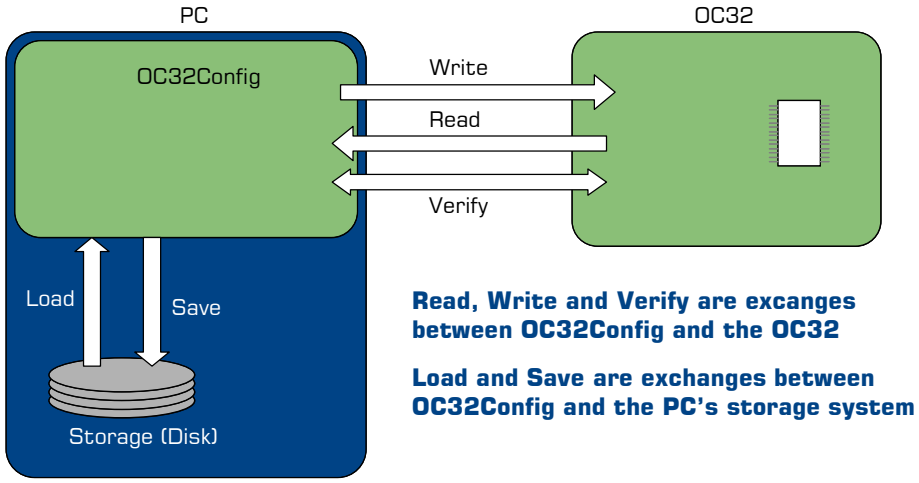


Fig 3: Load, Save, Read, Write and Verify

OC32Config always operates on one single OC32 at the time. So if you have 5 OC32's, you will need to configure those 5 OC32's individually. The settings you store to disk with "Save" and retrieve from disk with "Load" are stored separately for each OC32.

A "Device" in OC32 terminology is a functional part in your miniature world that you connect to and control from your OC32. Every Device has one or more electrical connections with certain characteristics and control rules. In OC32Config a "Device" is a set of "Pins" with electrical properties and control rules for one functional part in your miniature world.
You can define the electrical properties and control rules all by yourself, but fortunately there are predefined "Device Definitions". In many cases, the only thing you have to do is select the right "Device" and write the settings to your OC32 to ensure correct operation.
Predefined Device Definitions can always be adapted (by you) to your specific requirements if these are different from the standard requirements. For some Device Definitions it is always necessary to make some adjustments (finetuning), since it is virtually impossible to make a separate definition fo each variant.
Device Definitions are accompanied by documentation that describes how it can be used and how the device shall be connected for a correct operation.

Device Definitions have "Aspects". An "Aspect" is in OC32 terminology a "state"in which a device can be put. A railway crossing could for instance have Aspects "Open" and "Closed" and a signal could have Aspects "Red", "Green" and "Yellow".

## 2.3 Starting and General Functions

When OC32Config is started you see something like the window below. If you use a different version than 3.0.1 both the visual and functionality can be different. In that case, please consult the Release Notes of that specific version.

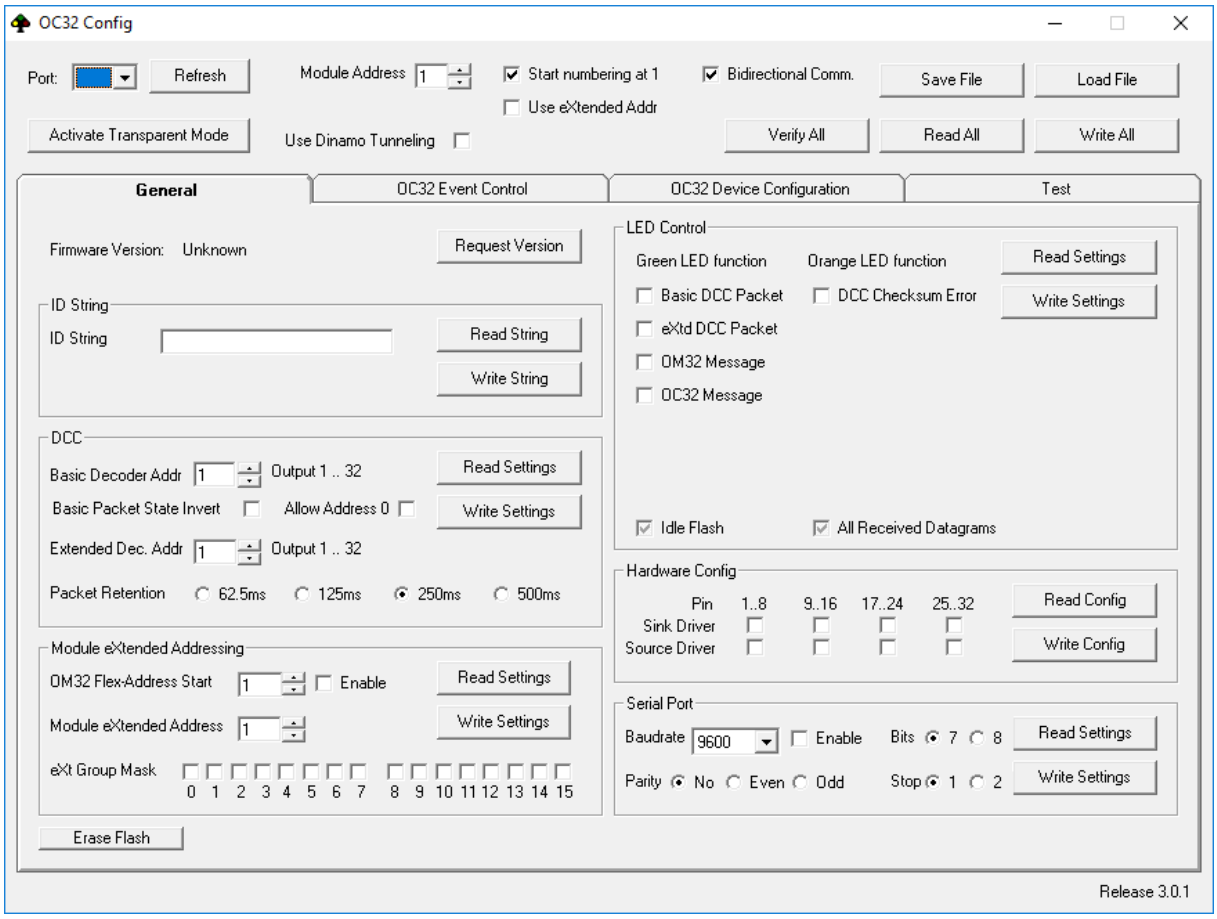


Fig 4: OC32Config

In the lower-right corner you find the "Release" of your running OC32Config. Note this is the version of your configuration software running on your PC. It is **NOT** the version of your OC32 firmware!

The version of your OC32 firmware can be retrieved from the module by a command (to be covered later in this manual). Note that version 3.x has quite a number of new functions compared to older versions. Therefore the compatibility between OC32Config 3.x and OC32 Firmware 0.0.2.x (or vice versa) is bad. So don't mix these versions. We recommend you to upgrade all OC32 software and firmware to the same, most recent version.

At the top of the window a number of buttons and settings is found:

- Port: Select the de com-port by which your OC32 is connected to your PC. If there is an RM-C, RM-U or UCCI in between, select te port of your RM-C, RM-U or UCCI.
- Refresh: When OC32Config starts, it checks which ports are available for use. Available means "present and not-in-use by something else". So, if there is another control program (iTrain, Rocrail) active and connected to that port when you start OC32Config, OC32Config won't show the port. The "Refresh" button renews the list without the need to close and restart OC32Config.
- Use Dinamo Tunnelling: If your OC32('s) is/are connected via a Dinamo RM-C or RM-U controller with firmware 1.3 or later, you can configure the OC32's through the Dinamo

system by activating this option. If no Dinamo system is encountered or your Dinamo system does not support this method, you'll receive a warning message and the function is turned off.

- Activate Transparant Mode: If your OC32('s) is/are connected via a Dinamo or Dinamo/MCC controller that does not support OC32 Dinamo Tunneling, you can use "Activate Transparant Mode" to switch the Dinamo controller in a special mode that allows configuration by OC32Config. When you are finished configuring your OC32's the Dinamo controller has te be reset or rebooted to switch back to normal operation. Note that using "Dinamo Tunneling" is the preferred option when your system supports it. If your system does not, consider upgrading your system firmware if there is a new release of your system that supports tunneling.
- Module address: This is the address of the OC32 module you want to configure. OC32Config operates per module.
  This means that when you have more than one module you have to configure them one by one. If you want to save the configuration(s) then you must store them in a separate file for each module.
- Start numbering at 1: This gives you the choice wether you number modules, Pins and addresses from 0 or from 1. Dinamo uses numbering from 0 (internally), many programs number starting from 1. You can switch this option on the fly as you like. The only thing that changes is the way data is presented to you on screen.. In the background everything stays the same.
- Bidirectional Comm.: Ensures that for every message sent by OC32Config a reply from the OC32 is requested. If a command is sent that requests real information from the OC32, this option is automatically turned on. If you use an RS232 medium, the OC32 cannot answer and this option shall be switched off. If the option is switched on and OC32Config received no answer, the option is switched off automatically after it has presented you an error warning.
- Use eXtended Addr: By this option you choose to address the OC32 by normal addresses (1 of 16) or eXtended addresses. When you switch this option on, first you receive a warning that this is "difficult stuff". If you confirm that you accept the consequences, the Module Address changes in a Channel Number and an additional address field eXtended Address appears. With Channel Number + eXtended Address you can address in theory 16 * 96 = 1536 modules. The advice is to use this only when you have more than 16 OC32 modules or expect to reach this amount on short term.
- Save File: This button saves the current configuration settings from OC32 Working Memory to a file. Configuration Settings is everything in the 3 Tabs below (General, Event Configuration, Device Configuration)
- Load File: This button loads a previously saved configuration into working memory.
- Write All: This button writes the **complete** configuration from OC32Config working memory into the addressed OC32. The entire OC32Configuration in the module is rewritten.
- Read All: This button reads **all** configuration settings from the addressed OC32 end thereby overwrites all configuration content of the OC32Config working memory.
- Verify All: Compares the configuration in OC32Config working memory with the configuration in de addressed OC32. Differences are reported. No changes are made.

Note: Any settings mentioned directly above are operational settings only. They determine how OC32Config interoperates with the OC32 at this specific moment. These settings are never saved in a file ore retrieved from any configuration file.

Below the general settings and functions you will find four tabs. The rightmost one (Test) is for testing several experimental functions of the OC32. This tab is not relevant to configuration and it is disabled by default. We will not discuss that tab in this guide.

The other three tabs are used for the configuration of the OC32. These can be divided into three main categories:

- General settings (General)
  These are settings that apply to the module as a whole or groups of I/O Pins;
- Event Configuration (OC32 Event Control)
  Here you can specify how the OC32 must respond to the Event Inputs for 'external events';
- Device Definition (OC32 Device Configuration)
  The OC32 can generate almost all electrical signals that your miniature world needs. You just have to configure the OC32 in such a way that for each connected device (for instance a signal, turnout or railway crossing) the proper signal patterns are generated.

## 2.4 General Settings

Click the "General" tab. You get the window as shown in figure 5.
The tab is subdivided is several "Frames". In most frames you find "Read" and "Write" buttons. These read/write settings within that frame from/to the selected OC32.

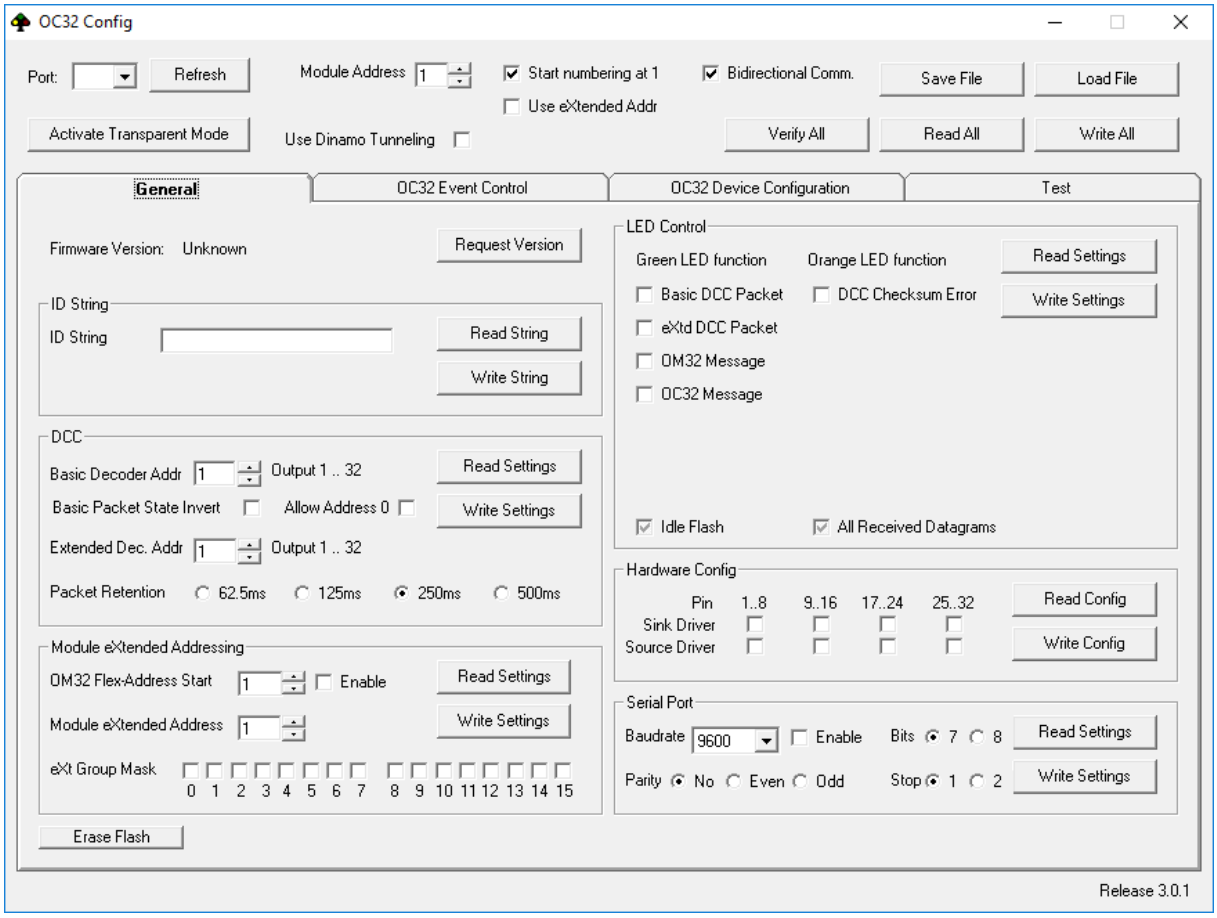


Fig 5: OC32Config: General Settings

You can read and write the following items:

- Firmware Version: This retrieves the firmware version which is actually running in the addressed OC32.
  Click "Request Version" to read the firmware version of the OC32. The OC32 reports the version of the firmware (the software in the OC32). This works for firmware release 0.0.1.0 and up. In case you have an older firmware version in your OC32, you'll get **no answer** to this request. Even if the OC32Config version on your PC is newer, you still won't be able to read an old firmware version.
  In case the Request Version gives no response, the firmware version can also be retrieved using the boot loader (see chapter 4). That procedure works independent of

the firmware version in your OC32, so also with versions prior to 0.0.1.0.
Version 3.x has many new functions compared to previous versions. The compatibility between 3.x and older (0.0.2.x) versions is bad. OC32 versions 0.0.3.x are temporary versions and no longer supported. If you run OC32Config 3.x And your OC32 reports a version prior to 3.0.0.0 update your OC32 before continuing with any further configuration.

- ID string: You can save an identification string in the OC32. This is a string (text), intended to give the OC32 a "name" to identify it. The string can have a maximum of 12 characters and all characters may be used. ID string has no other function than identifying the OC32, you are completely free in its use. You can for instance choose the filename under which you save the OC32 settings on your PC..
- DCC: The OC32 can be equipped with a DCC interface. In this frame several DCC specific settings can be made.
  The OC32 can be controlled using Basic DCC Accessory Packets and by Extended DCC Accessory Packets, if required intermingled. The amount of basic and extended DCC addresses assigned to the module is configured under 'OC32 Device Configuration'.
  - Basic Decoder Addr: Here you define the DCC Basic Accessory address range by which the OC32 is controlled. A 'standard' DCC Accessory Decoder contains 4 'ports'. The number of DCC ports used by the OC32 is variable and configured under OC32 Device Definition. Usually, the OC32 will occupy multiple consecutive DCC Accessory Decoder addresses. The first decoder address can be configured by the field 'Basic Decoder Addr'. To the right then appears the range of DCC portnumbers by which the OC32 van be reached from the digital DCC system.
  - Basic Packet State Invert: Every Basic DCC Accessory Decoder 'port' can be put in two 'states': 0 and 1, 'straight' and 'thrown' or 'green' and 'red'. In the OC32 DCC state 0 will trigger Aspect 0 and DCC state 1 trigers Aspect 1. Some DCC central stations interpret the DCC states 0 and 1 the other way around. If that is the case, you can correct this by activating the BPSI option. Note this works for the entire module.
  - Allow Address 0: According to the DCC specifications, decoder number 1 is the first addressable decoder. Only some central stations start numbering from address 0 onwards. The first 4 DCC portnumbers then can not be addressed by the cetral station. With the 'Allow Address 0'option, the OC32 numbers de DCC decoders starting from addres 0 (note: this then counts for both basic and extended DCC decoder numbers)
  - Extended Dec.Addr: Here you specify the first DCC Extended Decoder address on which the OC32 should respond. For Extended Addresses counts: 1 address per decoder (but of course, multiple (max32) addresses per OC32).
  - Packet Retention: It is undefined in DCC how often a DCC Accessory Packet may or shall be transmitted (in contrary to locomotive packets where it is clearly defined). When DCC Accessory Packets are repeated (reliability) by the central station, the OC32 will act on every reception of each DCC packet. To avoid this, the OC32 remembers the last received DCC packet and when exactly the same packets is received within the 'Retention Period' the corresponding action is only executed at the first packet reception.

- Module eXtended Addressing: In this frame several additional addressing options can be configured.
  - Module eXtended Address: This is the eXtended Address assigned to the OC32 module. Mind that every OC32 module always has an eXtended Address. Let op: Elke module heeft altijd een eXtended Address. This setting does not define wheter eXtended Addressing is actually used, that is determined "at runtime" by the controlling station in the protocol used.

  - eXtended Group Mask: Except an eXtended Address there is also an eXtended Group Mask. In a later stage it will be possible to address certain functions of the OC32 by addressing a group of OC32 module within one 'Channel' Every OC32 can be member of none, one or multiple groups. 16 Groups are defined.
  - OM32 Flex Address: Some systems cannot addres the OC32 by eXtended Addressing, but are capable of addressing OM32 modules. OM32 Flex Addressing is a trick to address more than 16 OC32's by a system that ca only address OM32 modules. This method requires very careful configuration of all your OC32 modules and is not discussed any further in this manual.

- LED Configuration: In this frame you can modify the way the diagnostic LED's operate, to facilitate troubleshooting of specific situations. Also if the continuous blinking of the green LED annoys you, you can switch that function off here.

- Hardware configuration: You can (should) set which output drivers are installed on your OC32. Depending on the selected driver(s) control options are switched on or off in the OC32.
  Each group of 8 outputs has 4 possible settings:
  - No sink-driver and no source-driver. This is the choice if you've placed a resistor array.
  - Sink driver, if there is a ULN2803 in the sink driver socket.
  - Source driver, if there is a UDN2981 in the source driver socket. Choose this configuration also when you insert a Decoder63 driver (also a source driver). When a source driver is used the outputs are electrically pair-wise interchanged. If you indicate that you have placed a source driver the software corrects this interchange. If you do place a source driver and do not set the hardware configuration the pin assignment on the 37 pin connector is incorrect.
  - Sink driver and source driver. If both types of driver are inserted, each pair of outputs becomes an H-bridge. In this case it is **very important** that the sink and source driver of the same half of the H-bridge are never active at the same time. This would lead to a short-circuit and blow-out the drivers. By correctly setting the hardware configuration, the OC32 protects this from ever happening, even if the control software instructs otherwise.

- Serial Port: In this frame you can choose if you wat to activate the Serial Accessory Por, and if so, what settings this port shall have. Note: When you activate the SAP, the RS232/TTL interface on your OC32 may no longer work reliable! More information on the SAP can be found in the hardware manual.
- In the bottom-left corner you'll find an additional button: "Erase Flash". This button will reset the configuration memory of your OC32 to factory settings. This procedure will not erase your firmware nor will it reset the working memory of OC32Config.

## 2.5 Configuration using Device definitions

### 2.5.1 Introduction

An electrical Device, controlled by the OC32, is connected to the OC32 by one or more wires. The number of wires needed depends on the Device. A flashing light uses one I/O Pin, a block signal with 2 lamps or LEDs uses two Pins. A German outbound signal with distant signal requires 9 Pins.

The OC32 must correctly control the I/O Pin(s) to which the Device is connected. You can configure this manually, that is the complicated way, which unfortunately sometimes is necessary. In most cases, however, there is a more convenient method by using pre-defined "Device Definitions".

Device Definitions are stored in 'Device Definitions' files. When OC32Config starts, a default Device Definition file is loaded. During operation of OC32Config you can switch to another Device Definition file or you can load multiple sets if required.

Each Device Definition file is accompanied by a description/manual. This manual describes for each Device what it's function is, how many I/O Pins it needs, in which order the wires from the Device need to be connected to achieve correct control by the OC32 and, if applicable, finetuning options for the Device.

Logically, a Device is always referred to by the first I/O Pin to which the Device is connected. This first Pin is therefore called 'Device Control Pin'. Every Device can be put in 4 or 12 'states' (0..3 or 0..11), depending on the Device Definition. In the OC32 such state is called an 'Aspect'. Every Aspect makes that the OC32 controls the Device Pins in such a way that the Device does what the corresponding state requires.

### 2.5.2 Selecting a Device Definition

Selecting a Device is done from the tab 'OC32 Device Configuration'.
On this tab you find, amongst many other items, a selection box 'Show Details'. **We strongly recommend you NOT to select this option fort he time being. This will keep things simpler until you start to understand how the process works.**

At the top-right corner of the tab, above the dotted line, you'll find the name of the Device Definition File that is currently loaded. In the example of figure 6 this is 'Generic 2017/02/19'. What definitions are included in that set can be discovered by clicking on the triangle, left from the button "Load Device". This will open a drop-down list of Devices you can choose from.

The device-names of the Generic set are in the English language. To keep the authentic look & feel, we stick to the convention that names of devices, aspects, etc are written in the language of the country the Device Definition Set represents. The Device Definition file for the German signalling system therefore is in German, the one for the NS signalling system is in Dutch and the one for the SNCF signalling system is in French.

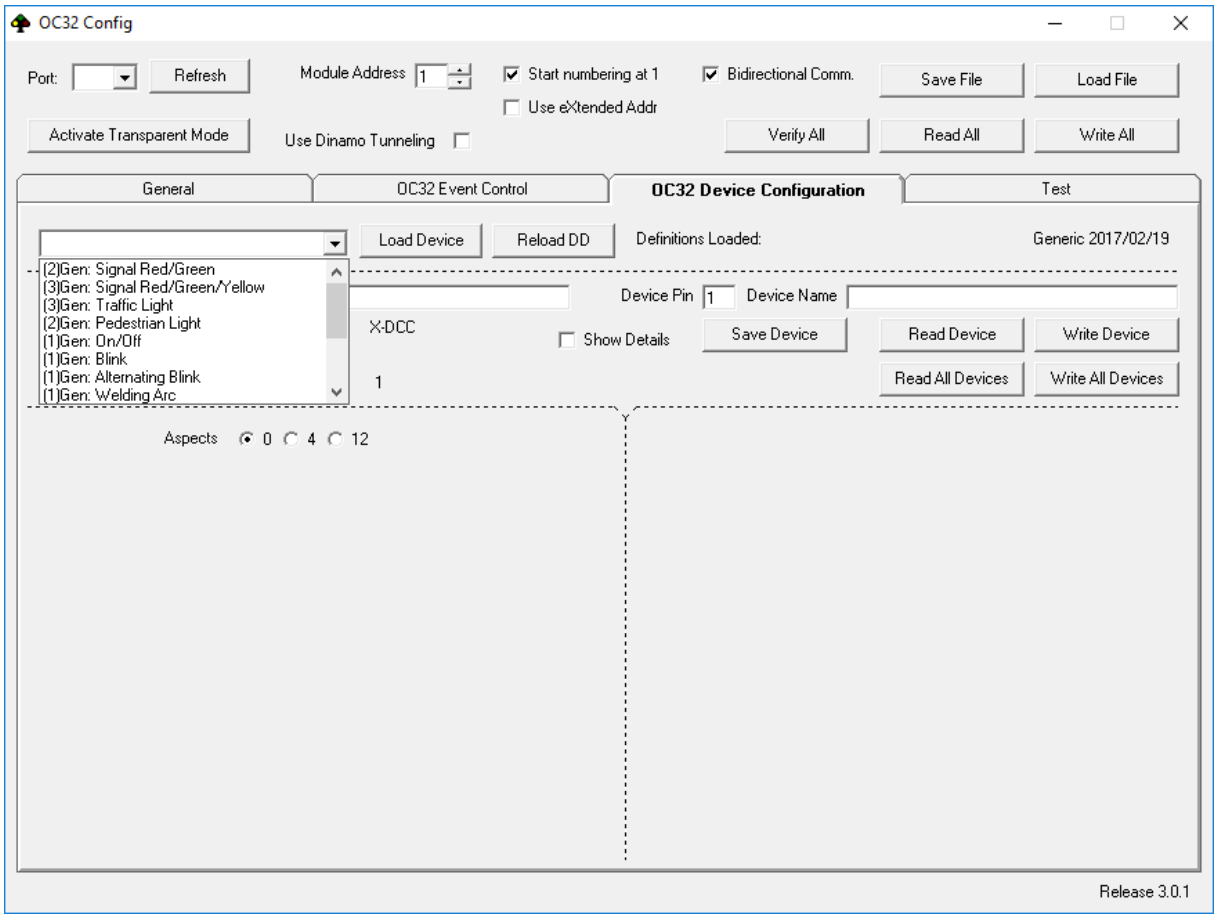


Fig. 6: OC32Config Device Configuration

The Device names always start with a number in brackets, followed by a number of characters and a colon. The number in brackets indicates the number of I/O Pins the Device needs. If this is a single number, it means the number of consecutive Pins. If the number says (1+1) it means 2 Pins that are not necessarily consecutive. The characters before the colon indicate to which definition set the Device belongs, e.g. DE for Germany, CH for Switzerland and Gen for Generic. After the colon a hopefully sensible description follows.

At any time you can choose which definition file you want to use. We assume you have already downloaded additional Device Definition Files and remember where on your PC you saved them. To select another definition file, click the button 'Reload DD'. A window opens by which you can browse to the location where you have stored your definition files. Select the required file and confirm your choice. The name in the top-right corner now has changed and in the drop-down list you now will find different Devices to choose from.
In stead of selecting another definition file, you can also add a definition file tot he already loaded set. That can be achieved by pressing the 'Shift' key and simultaneously clicking 'Reload DD'. The above process now leads to the result that the selected file is added to the current file.

How things work can best be explained by an example.
We want to connect a Dutch NS 3-color signal with number to Pins 4, 5, 6 and 7.
The first step is selecting the right definition file. Click 'Reload DD' and find and select the file 'OC32Devices NL 20130305'.
Left to the button 'Load Device' you now can select the Device '(4)NL: 3kleuren+cijfer'.

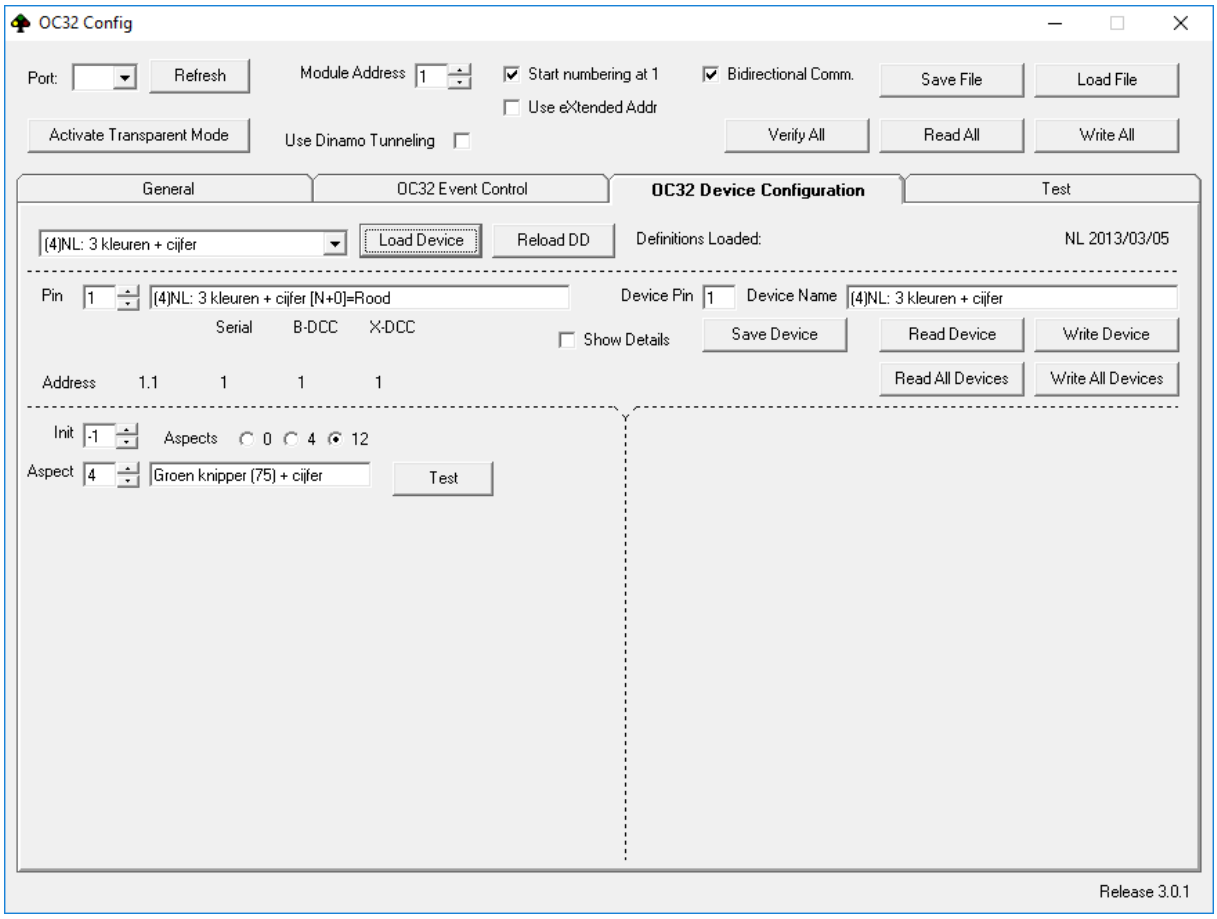


Fig 7: Configuring an NS signal with OC32Config

Now, below the dotted line on the left, select Pin 4, the first Pin we want to use for this Device. Use the arrows besides the 'Pin' field or type the number '4' in the box. Finally click the button 'Load Device'. You should see something like figure 7 now.

The Device Definition '(4)NL: 3kleuren+cijfer' now is loaded in OC32Config on Pins 4 to 7. Next to the box with the Pin number you now find the description of the function of the corresponding Pin. Scroll through Pins 4 to 7 and note that every Pin has the description of the lamp that should be connected to it. Also note that with every Pin in this range the box called "Device Pin" shows number 4 to indicate that all these Pins belong to the Device controlled by Pin number 4 and therefore these Pins belong together. Right to Device Pin you'll see the 'name' of the Device..

Select Pin 4 again. In the dotted area bottom-left also something has changed. There you'll find now Aspect 0 and behind that the description 'Rood'. This means that activating state ('Aspect 0') of this Pin will result in showing 'Red'. The 4 Pins assigned to the Device Definition to the Device will be switched in such a pattern that the signal will show 'Red'. Scroll through the different 'Apects' of the signal to see the possible states. In this case you'll find that 'Aspects' 0 to 7 have a meaning. States 8 to 11 are not in use.

On top of the 'Aspect' number you see the box 'Init'. Here you can select in which state the OC32 shall put the Device when the OC32 is powered up. This will be done independent of any external control signal. '-1' means no initialisation by an Aspect definition takes place. '0'to '11' activates the corresponding Aspect.

Behind Aspect you see a 'Test' button. This botton won't work yet. The reason is nothing is configured in your OC32 yet. The Device Definition is loaded only in OC32Config working memory. (see paragraph 2.2).

### 2.5.3 Testing the Device Definition

Before you can actually test the Device, of course the Device needs to be electrically connected to your OC32. Furthermore the Device Definition needs to be transferred to the OC32. Before you can send your settings to the OC32, you need to select the correct com-port and the correct OC32 module address at the top of the screen. If your OC32 is connected via Dinamo, select 'Dinamo Tunneling' if your system supports this, or otherwise put your Dinamo system in transparent mode by clicking the corresponding button.
Clik the button 'Write Device' while one of the Pins belonging to your Device is selected. Which Pin exactly is not important. All settings corresponding to the Device, controlled by the 'Device Pin' are now transferred to the OC32. During transfer the text 'Working … Please WAIT' appears. Please do as the text indicates.

When the transfer is finished, you can click the button 'Test'. If everything is fine your Device will now go into the desired state.

### 2.5.4 Changing the Descriptions

After loading the Device Definition, the descriptions in the fields behind Pin number, Device Pin and Aspect can be changed as desired. For example you can change the text:
'(4)NL: 3 kleuren + cijfer [N+0]=Rood'
into
'(4)NL: 3 color + number [N+0]=Red'
or:
'(4)NL: Main Signal + number Station north side [N+0]=Red'
The text itself has no function other than documentation purposes for you.
The same applies to the description behind the various 'Aspects' and 'Device Pin'

The tekst after 'Device Pin' can only be modified if the 'Device Pin' is selected as Pin.
The descriptions of 'Pin' and those of 'Device Pin' seem partly redundant . The cause is that before version 3.0 there was no 'Device Pin'. Gradually the Device Definitions will be modified so that 'Pin' description just mentions the actual function if that I/O Pin and no longer te complete Device name.

**Note that the descriptions themselves are not stored in the OC32 module.** The memory capacity in the OC32 module itself is not sufficient for this. Descriptions are saved in a configuration file you can save by using the 'Save File' button. You must do so if you have modified any descriptions, otherwise all your work is for nothing

### 2.5.5 Control of subsequent pins

The aspect definitions associated with the first pin of a device also control the subsequent pins. The second, third and any other subsequent pins of the device are not directly addressed.

Nevertheless you will see that for aspect 0 and 1 of the higher pin numbers (that is, those that have [N+1], [N+2], etc. in the description behind Pin) often something is defined. The reason for this is control by DCC.
Many digital systems are only able to switch the outputs of accessory decoders in two positions: "straight" and "thrown", or "green" and "red". When you use such a system to control an OC32 via DCC, this would mean that the aspects 2 through 11 can not be reached. The solution is in the Aspects 0 and 1 of the subsequent pin numbers. These Aspects redirect to the Aspects 2..11 that are defined for the first pin. In our example:

Pin [N+0]
Aspect 0 = Red
Aspect 1 = Green
Aspect 2 = Yellow + number
Aspect 3 = Yellow
Aspect 4 = Green blink (75) + number
Aspect 5 = Green blink (75)
Aspect 6 = Yellow + blinking number (75)
Aspect 7 = yellow blink (75)

These Aspects work if you can control at least 8 aspects on 1 output. This is possible when you define the OC32 as OM32 from the control program Koploper, as "OC32 aspect" in control program iTrain or if your digital system has the capability to address "Extended DCC Accessory Decoders".

Simultaneously the following definitions apply for the same device:

Pin [N+0]
Aspect 0 = Red
Aspect 1 = Green
Pin [N+1]
Aspect 0 = (R) Yellow
Aspect 1 = (R) Yellow blink
Pin [N+2]
Aspect 0 = (R) Green blink
Aspect 1 = (R) Yellow + number
Pin [N+3]
Aspect 0 = (R) Green blink + number
Aspect 1 = (R) Yellow + blinking number

So the Aspects 2 through 7 of the pin [N+0] are also accessible in a different way, by the Aspects 0 and 1 of the pins [N+1] through [N+3]. Your digital (DCC) system can use this mechanism to activate all possible Aspects of a device by setting the relevant addresses in positions 0 (straight) or 1 (thrown).
The (R) in the definition signifies "Redirect". This means that this Aspects refers to a different Aspect of another pin.

The Aspects associated with a pin in a Device Definition are described in the documentation of the corresponding Device Definition.

### 2.5.6 Device Addresses

A Device, configured in the OC32, is controlled by Aspects, stored under one or more Pins. Every Pin has one or more addresses. In principle there are three types of addresses per Pin:

- A serial address (Serial). This is the address by which the Pin is Device is addressed via one of the serial interfaces, so RS485 or RS232/TTL.
- A Basic DCC address (B-DCC). This is the address, or DCC portnumber, by which the Pin is addressed using Basic DCC Accessory packets.
- An Extended DCC address (X-DCC) This is the address by which the Pin is addressed using Extended DCC Accessory packets.

The Pin-address can be calculated, but OC32Config also shows the addresses below te Pin number. There you'll find the tekst 'Address' followed by 4 numbers. In order these numbers mean:

- The serial address, expressed by the module number followed by the serial address within the module.
- The serial address, expressed in linear format, assuming 32 addresses per module.
- The Basic DCC address
- The Extended DCC address

### 2.5.7 Fine tuning within device definitions

Most devices are straightforward to use, as described in paragraph 2.5.2. Simply select the right device for the pin to which you have connected it and you have little else to worry about. Unfortunately this does not apply to all devices. Some devices need some further adjustment. Of course we could make separate device definitions for every variation, but that would result in quite a large number of devices. For devices that require fine-tuning you will have to make some adjustments yourself.

In the documentation of the Device Definitions you find a description of the fine tuning process you can follow for a device. So check the specific documentation that supports the device definition.

What is described below is how changes can be made. As an example we use the device definition "(1) Misc: Servo Turnout", the control of a turnout by a servomotor. We load the device definition to pin 1 as described in paragraph 2.5.2. Then activate the checkbox "Show Details". The screen will change and look similar to the one in figure 8.

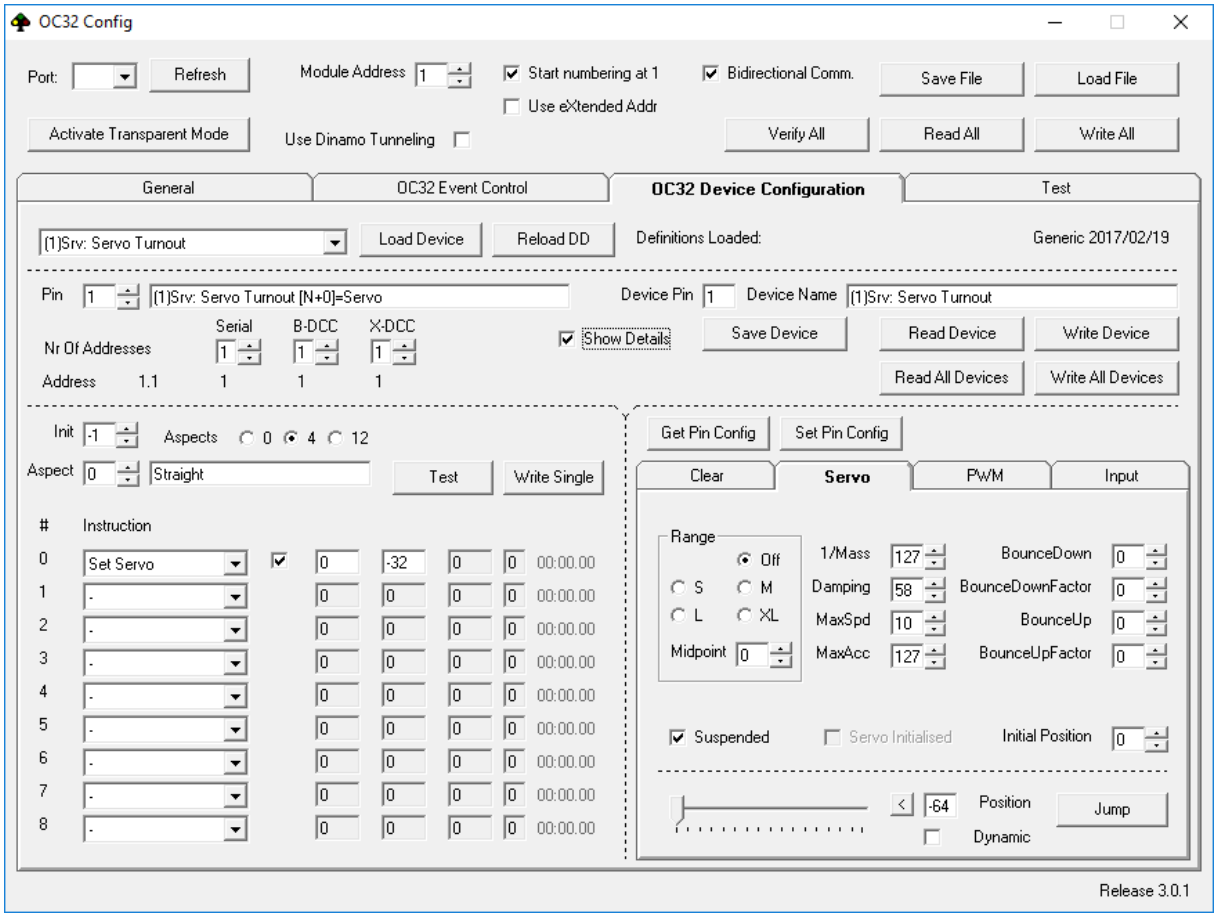


Fig 8: OC32Config: Finetuning a Device Definition

In the lower part of the screen you see two ‘frames’. In these frames two main properties can be configured.

The lower left section contains details of the Aspect definitions. The instructions are the actions carried out by the activation of the selected aspect. For every pin there are 4 or 12 aspects, depending on the settings. In most cases while finetuning, there is no need to change a lot, at most a single parameter. Refer to the description of the corresponding device definition.

The lower right section contains the characteristics how each pin is controlled. There are actually three possibilities: driving as PWM output (Pulse Width Modulation), as a servo output or use the Pin as inout.

In our example, in which we use a servomotor to drive a turnout, the right-hand section is of course already set to "Servo". You'll need to adjust the parameters to the characteristics of the servo you're using and this partially depends on the way you have mounted it in your miniature world. A servo can have many parameters. You will find full details on setting servo behaviour in a separate document OC32 3.0 Extended Configuration. Here we limit ourselves to the parameters relevant to the device that drives a turnout.

A servo usually has a maximum rotation of about 180 degrees. To control your turnout you'll probably never fully use this maximum angle. The part of the maximum angle that you actually use can be set below "Servo Range". "S" means the smallest angle, "XL" the biggest. An exact number of degrees cannot be given in this manual, because it depends on the actual brand and type of servo that you are using. You need to try and see. If you have adjusted a setting you can temporarily write it to the OC32 by using button "Set Pin Config". The settings then become active so you can test it. Testing can be done with the slider at the bottom.

The Range can also be influenced by the "Midpoint", which is the centre position that belongs to the "0" position of the slider. In the range "XL" an adaptation of the Midpoint has no effect. In that range the maximum angle of the servo is already in use. If you limit the range you can use the midpoint to determine what portion of the total angle you want to use. Note that when you change Midpoint, you must press ‘Set Pin Config’ after the change to activate the new setting.

Set Range and Midpoint in such a way that the angle of the servo is large enough to change between the two extreme positions of the turnout. The speed by which the turnout is moved between positions, can be set by parameter MaxSpd (1..127). To test this you need to check "Dynamic" next to the slider.

When you are satisfied with the basic settings Range and Midpoint, you need to set the two extremes between which the servo actually moves. The most convenient way is by first establishing these extremes with the slider. Write down the numbers in which the turnout is correctly in the straight and thrown position.

Now in the left-hand area select "Aspect 0 (Straight)" and in field "Param" (which is now -32) type the servo position that corresponds to the "straight" position you found for your turnout. Next select "Aspect 1 (Thrown)" and in field "Param" (which is now 32) type the servo position that corresponds to the "thrown" position of the turnout.

Note that the checkbox “Suspended” is ticked in the lower right frame. This makes that the servo is not activated (suspended) until the first command (Aspect setting) for the Device is received. If you want the servo to take an initial position when the OC32 starts, you can either choose an Initial Aspect in the left-hand frame, or you can untick the “Suspended” checkbox. If you do this however, you also need to set the “Initial Position”. The initial position can be set by either typing the desired number in the box “Initial Position” or by finding the desired position using the slider and then doubleclick the “Initial Position” box.

Now actually configure the Device in the OC32 by clicking 'Write Device'. Everything related to this Device now is uploaded in the OC32 as configuration. By the button 'Test' you now can actually test positions (Aspects) 0 and 1 of the corresponding turnout and verify that they are correct.

The above is just one example describing the "logical process" that can be used to make adjustments. Modifications in other device definitions may vary. As stated earlier, for this you must consult the specific description of the device definition.

## 2.6 Event Inputs

If your OC32 is equipped with Event Inputs, you can configure these functions via the 'OC32 Event Control' tab (see figure 9)

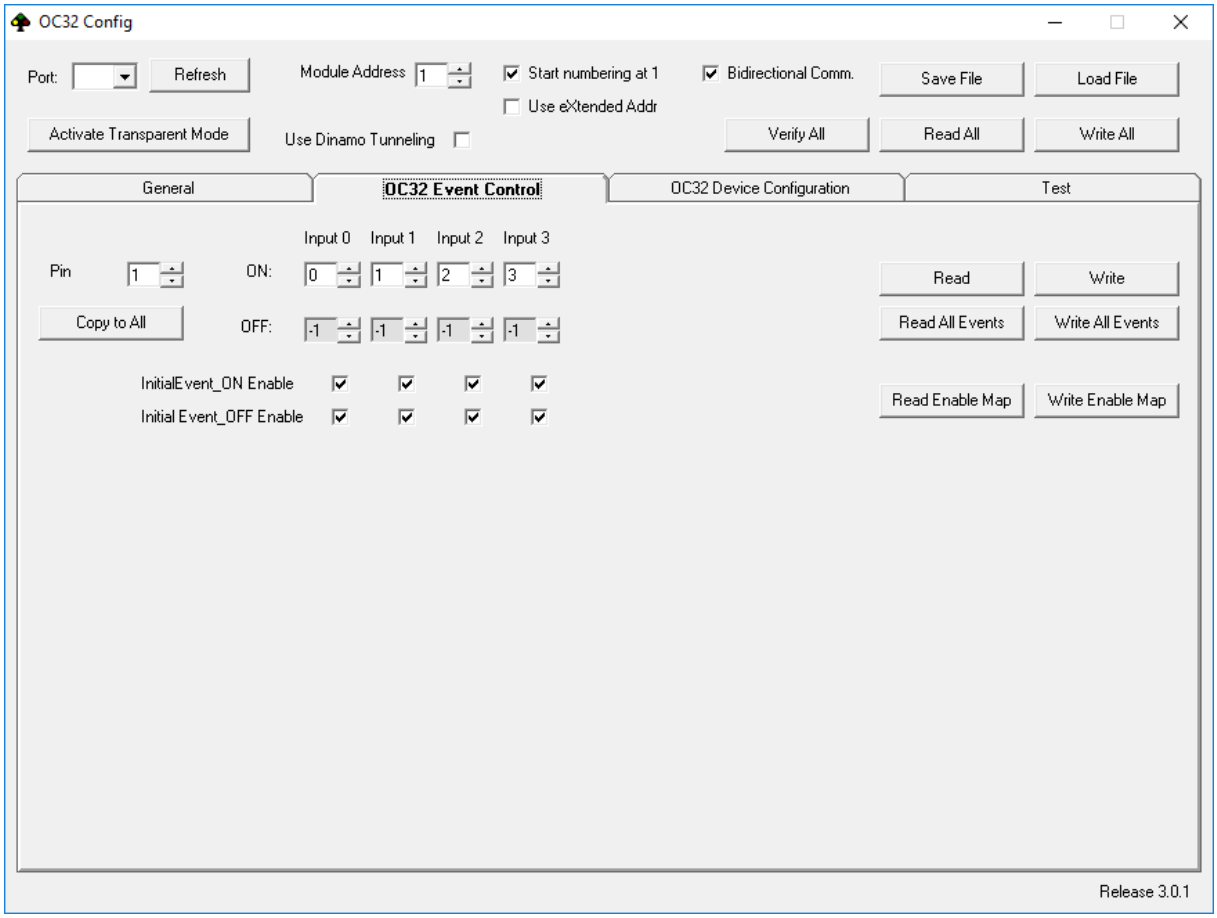


Fig 9 "OC32 Event Control"

There are 4 Event Inputs: Input0 to Input3. Every Event Input has 2 possible events:

- ON-Event: When the input becomes active
- OFF-Event: When the input becomes inactive

For every individual Pin (0..31) you can define what shall happen when any of the 8 possible events occurs, specifically which Aspect (0..11) in any of these cases shall be triggered. A value of '-1' means 'Do Nothing'.

The 8 external events also have an 'enable mask'. This mask defines for which Events the triggering results in the per-Pin defined actions. When the event is 'disabled', the OC32 will not react on the external Event.

The 'Initial Event Enable Mask' configures which Events shall be enabled when the OC32 is started. The active mask can be modified during operation of the OC32 via the 'Event Input' instruction (see the separate document OC32 3.0 Extended Configuration

The buttons "Read" and "Write" receive and send the event setting from/tot he OC32 for the selected Pin only. The buttons 'Read All Events' and 'Write All Events' do the same for all Pins.

The buttons 'Read Enable Map' and 'Write Enable Map' received/sends the Initial Event Enable Map from/to the OC32

Filling all Pins with the desired setting scan be quite some work. The button 'Copy to All' fills all Pins with the active settings.

# 3 Controlling a configured OC32 from a digital system

## 3.1 Controlling the OC32 connected via Dinamo or directly from your PC

The OC32 can be controlled either directly or via a Dinamo or DInamo/MCC system by iTrain, Rocrail or Koploper. For details how to configure your software to achieve this, consult the manual of your control software.

## 3.2 Controlling from a DCC system

You can control the OC32 from a DCC compatible digital system. To do this, you need an OC32 with a DCC interface. If this interface is not already mounted on the module, you can add it yourself or have it added later on.

How you connect the DCC interface to your digital system is, amongst others, described in the hardware manual of your module..

Before the OC32 can act on DCC commands, you must configure the OC32 as described in Chapter 2. In most cases configuration with default device definitions will be adequate. Just remember that you set the proper (preferred) DCC address with the generic settings (see 2.4).

If you have a DCC system that can generate "extended DCC accessory packets" you can set each decoder output to 32 positions. The OC32 knows only 12 of these, but usually that will be enough to control all of the "Aspects". Which "Aspect" does what can be found in the description of the OC32 Device Definitions.

If your DCC system can only generate "basic DCC accessory packets", you can only address of each output the Aspects 0 and 1 ("straight" and "thrown"). That's not a problem because the Aspects 0 and 1 of the successive pins, that are used by a device, normally have a reference to an Aspect > 1 of the first pin (the "base-pin" of the device).
In this way, usually all necessary ‘states’ for a Device can be addressed, although indirectly. The matrix, that shows which Aspect of which pin activates which ‘state’ you will find in the description of the OC32 Device Definitions.

# 4 OC32 Firmware Update

For the OC32, software updates and upgrades are published regularly, which offer additional functionality. You can install new firmware yourself, provided that you have a PC with RS485 interface, like the U485 or you control your OC32's via a Dinamo or Dinamo/MCC system.

New firmware for the OC32 can be found on the DinamoUsers portal (http://www.dinamousers.net). The requirement for obtaining this software is to register on this portal and that you have the OC32 Customer Status. Registration is free and possible for everybody who accepts the Terms of Use, The OC32 Customer Status you get for free or can be requested when you have purchased the OC32.

To be able to perform an update/upgrade, the following is required:

- A PC with the Windows operating system
- An RS485 interface on your PC, for example in the form of a U485, or a Dinamo RM-C, RM-U or UCCI(/E) controller.
- VPEB Bootloader software.
- The OC32 firmware of your choice.

The last 2 items can be downloaded from http://www.dinamousers.net

Perform the following steps:

1. Install the VPEB bootloader software on your PC. This can be done easily by unpacking the .zip file in a map of your choice. It is convenient to do that somewhere under "Program Files". You might want to create a shortcut to the unpacked AVRootloader.exe. This step needs to be executed only once.
2. Download the firmware version that you want to install. Unpack the .zip file. The file you need has extension *.acy. Store that somewhere on your PC where you can retrace it.
3. Make sure the OC32('s) you want to update/upgrade is (are) connected by RS485 either directly to your PC (e.g. U485) or indirectly via an RM-C, RM-U or UCCI(/E).
   Switch on the power for the OC32('s)
4. Only if your OC32's are connected indirectly, before you continue with step 5, you need to switch your RM-C, RM-U or UCCI into Bootloader Transparent Mode:
   - Start DinamoConfig (fig.12) and select the COM-port of your Dinamo system,
   - On tab RM-U / UCCI, select options"Transp.M" AND "Boot.TM",
   - Click "RM-U-Options"
   - Ignore the fault message and close DinamoConfig.
   NOTE: The above **can not be done with OC32Config**!.
5. Start AVRootloader.exe. You will see a screen looking like the one in figure 10.
   The "baud rate" is set to 38400 and "Sign" is set by default to "VPEBbootloader".
   **Don't change this**, otherwise it won't work!
   It may be convenient to check "Open protocol-window after processing".
6. Set "Port" to the COM-port to which your RS485 connection is connected. **Be aware that** the choice "AUTO" doesn't work with the type bootloader that the OC32 contains. You do have to select the correct port.
7. Choose in the box to the right of the "FLASH" button, the *.acy file that you have stored under step 2. You can do that by clicking on the button "…" to the right of corresponding box and then select the correct file. Remind to indicate that you are looking for an *.acy file, otherwise you won't see any.
8. Place a RESET-jumper on the OC32 that you want to update/upgrade. You do that by connecting the 2 pins of the 6-pin connector (actually 8-pin with 2 pins missing) the furthest away from the edge of the printed circuit board. The easiest way is to borrow the jumper of JP2 and place it on the mentioned 2 pins (see figure 11)
9. Click in the AVRootloader program window on button "Connect to device". At the top of the window you will see: "Connecting…, please press RESET on the Device"

10. Now pull the RESET-jumper "with a smooth move" off the OC32. If everything has gone well, both LED's on the OC32 will light (and remain lighted). At the top of the AVRootloader window you will now see "connected". In the tab "Device Information" you can find some additional information about the type of processor and the current software. These details are of no importance in this procedure.
11. Now click (in the tab "Programming") on the button "Program". If you have checked "Open protocol-window after processing" in step 4, then after about 6 seconds the result appears on the "Protocol" tab. Your latest software is now installed in the OC32. Note: it might be that this duration will increase when the OS32 software grows in size by adding new features.
12. Click on the "Programming" tab on button "Disconnect device". The OC32 will now start up normal with the new firmware.
13. If you want to update more OC32's, repeat from step 8.
14. Don't forget to restore the JP2 to its original position. Restore any connections that you have changed before performing the update.
15. If applicable: Reset your RM-C, RM-U or UCCI(/E) to leave Bootloader Transparent Mode.


Fig 10: AVRootloader

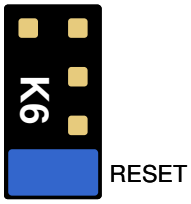


Fig 11: OC32 RESET jumper

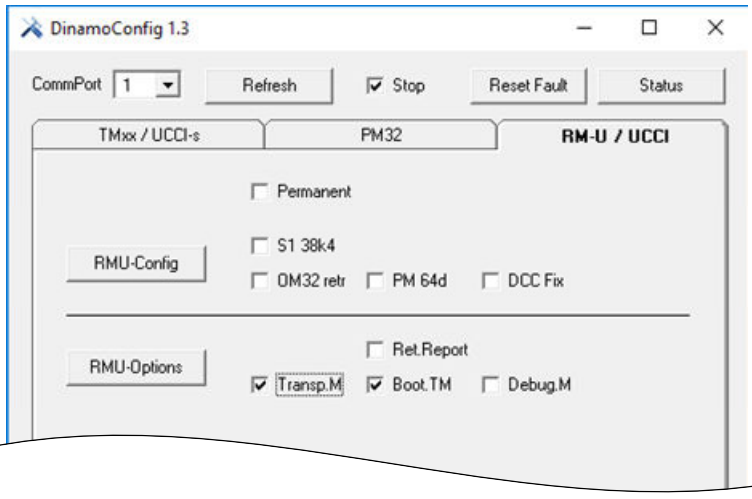


Fig 12: Start Bootloader Transparant Mode

# 5 Solving known issues

## 5.1 OC32 won't start after reconfiguration

This description applies to the situation where your OC32 won't start anymore **after you have reconfigured the module** using a self-made configuration. 'Self-made' means that you have not just selected a standard Device Definition from one of the libraries, but either have modified a standard Device Definition to your requirements or have 'programmed' a self-invented Device Definition.
This description never applies when you connect a new OC32 for the first time and have not configured the module yet.

It can occur that an infinite loop without delay has been programmed in an Aspect Definition When the Aspect, containing this error, is activated, the OC32 will stall. If this is the case you can, while nothing has activated the erroneous Aspect yet, connect to the OC32 by OC32Config and fix the error by rewriting the Aspect with a correct or empty Aspect Definition.

However, if the Aspect containing the endless loop has been set as "Initial Aspect", the OC32 will execute the Aspect Definition as soon as the OC32 is powered-on. The result is that there is no way to prevent the OC32 from stalling. When this is the case:

- The orange LED flashes briefly once when power is applied
- The green LED won't flash after start-up[1]
- You can't connect to the OC32 with OC32Config
- You **can** connect to the OC32 with the bootloader

Since you can connect to the OC32 with the bootloader, you can install new firmware in your OC32. This is no solution to your problem, since re-installing the firmware does not clear the configuration memory. However it gives you the opportunity to install firmware 3.0.0.0 or later in your module in case you still work with an older version, that does not support the procedure below.

If your OC32 won't start after reconfiguration because of an endless loop in an initial Aspect, you can achieve that the OC32 skips starting the initial Aspects. To be sure that nothing else can start the faulty Aspect Definition you best be sure that:

- The connector(s) for the I/O Pins and Event Inputs are disconnected
- The DCC interface (if present) is disconnected
- No control software (Koploper, iTrain, etc) is active that could issue commands

Execute the following procedure:

1. De-power the OC32
2. Place 2 jumpers on the 6-pin jumper-block (K6) as indicated in figure 45. The lower jumper is the Reset-jumper you know from the bootloader procedure, the jumper to the right is meant to instruct the OC32 to skip executing the initial Aspects. If you don't have any spare jumpers, temporarily use the ones from the RS232 selection and the RS485 terminator jumper.
3. Power-up the OC32

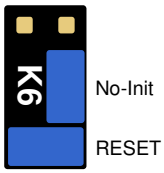

Fig 13: No-Init

[1] Of course the green LED won't flash either if you have configured it that way.

4. Pull the Reset-jumper 'in a fluent motion' from the jumper block. Make sure the 'No-Init' jumper stays in position. You should now see:
   - The orange and green LEDs both light-up shortly (bootloader)
   - the orange LED goes off, the green one remains lit for about 4 seconds (indication that initialization is skipped)
5. Now remove the No-Init jumper from K6.
6. Put both jumpers back to where they were

You now should be able to connect to the OC32 using OC32Config. Find and fix the configuration error, reconnect the OC32 the normal way (should you have changed anything) and reboot the OC32.

(This page is intentionally left blank)