CanOpen Configurator

IDE user manual

Rev. 1.0.26 © Promax srl

1 Introduction

The CanOpen Configurator software, is a tool that is included in the VTB IDE package and allow to describe the CanOpen net that a Promax CNC will use.

With this software, the developer can easily describe the number of slave nodes and the various PDOs that the net will exchange.

2 Note on the devices using

This software can work with Promax CanOpen slaves (Old CanIo and CanAx or newer NG series) and also with third companies slaves.
The only needed is to have a standard EDS file of the device, to understand which are the properties or the objects that can be used.

For third company EDS files, ask to the producer to provide it.

3 Note on the CanOpen configuration meaning

CanOpen configuration initialize every slave node inside the net, with the needed synchronous PDOs (Process Data Object).

It also informs the CanOpen master on which data it will exchange with the slaves that are in the net, using the PDO communication.

At the end of the procedure, it also makes the slaves in the “operational” state, with a NMT command, as defined in the DS301 draft.

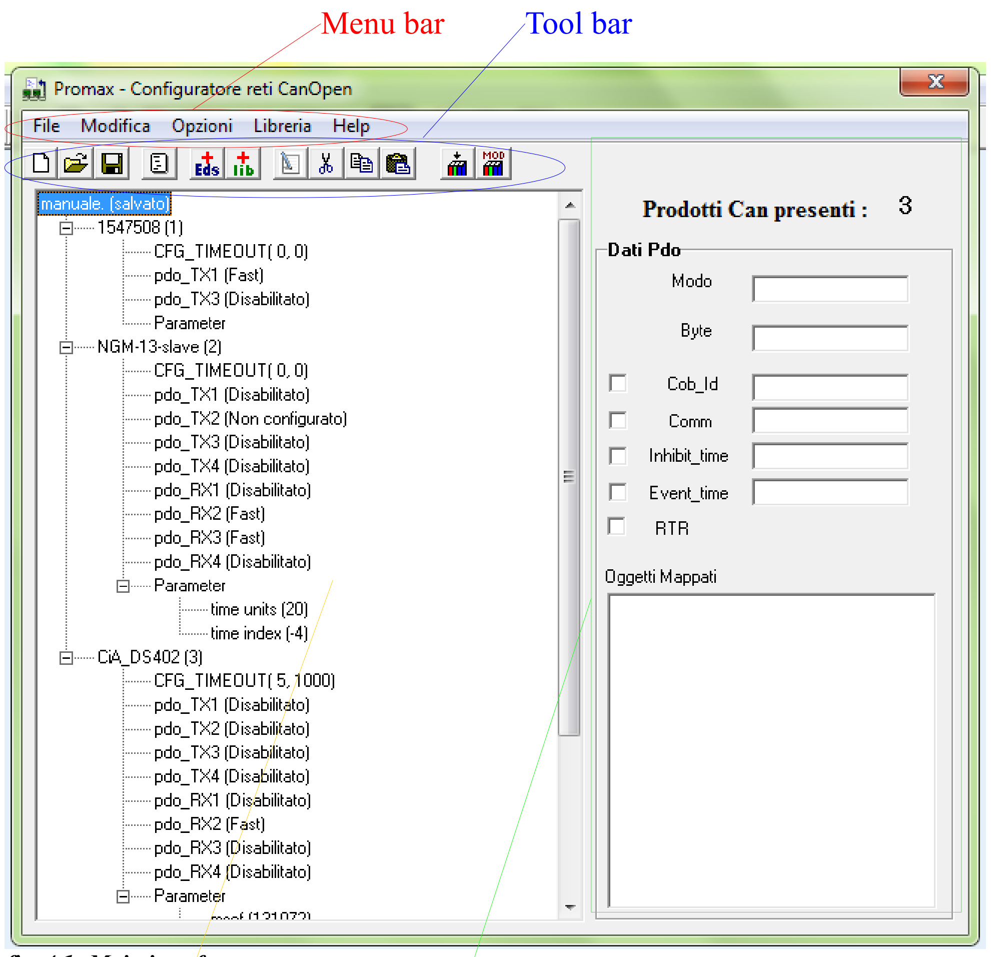
Data and the structures that can be configured inside every slave, depends on the device firmware, not from the CanOpen configurator software and are described in the EDS file of the device.

The net Configuration isn't linked to the master that will use it, but only on the slave devices present in the net.

The net configuration, can be also empty, without any PDO configured. It means that the master will not exchange any synchronous data with the slave. Perhaps, everything will be managed only with asynchronous SDO. The only thing that will be always made, is the NTM command.

An unknown slave can be also simulated with EDS file of a similar device, not supplied from the manufacturer and already present in the library, but without any guaranties of success.

4 Main interface



```
Promax - Configuratore reti CanOpen

manuale. (salvato)
⊟ 1547508 (1)
      CFG_TIMEOUT( 0, 0)
      pdo_TX1 (Fast)
      pdo_TX3 (Disabilitato)
      Parameter
⊟ NGM-13-slave (2)
      CFG_TIMEOUT( 0, 0)
      pdo_TX1 (Disabilitato)
      pdo_TX2 (Non configurato)
      pdo_TX3 (Disabilitato)
      pdo_TX4 (Disabilitato)
      pdo_RX1 (Disabilitato)
      pdo_RX2 (Fast)
      pdo_RX3 (Fast)
      pdo_RX4 (Disabilitato)
    ⊟ Parameter
          time units (20)
          time index (-4)
⊟ CiA_DS402 (3)
      CFG_TIMEOUT( 5, 1000)
      pdo_TX1 (Disabilitato)
      pdo_TX2 (Disabilitato)
      pdo_TX3 (Disabilitato)
      pdo_TX4 (Disabilitato)
      pdo_RX1 (Disabilitato)
      pdo_RX2 (Fast)
      pdo_RX3 (Disabilitato)
      pdo_RX4 (Disabilitato)
    ⊟ Parameter
          mcof (131072)

Prodotti Can presenti :   3

Dati Pdo
  Modo
  Byte
☐ Cob_Id
☐ Comm
☐ Inhibit_time
☐ Event_time
☐ RTR
Oggetti Mappati
```

fig. 4.1: Main interface

Can Open Configurator interface, appear as a standard Windows style application, with classics menu and tool bar.
Then there is a configuration panel, that shows the actual configuration with the name of the file and the various nodes included.
The other section can show a summary of the characteristics of the selected PDO.

4.1 Tool bar

🗋	**Nuova Configurazione** - From menu ***File*** → ***Nuova*** It creates a new configuration file. The previous one is closed requesting a confirm for saving.
📂	**Apri Configurazione** - From menu ***File*** → ***Apri*** It opens an existing configuration.
💾	**Salva Configurazione** - From menu ***File*** → ***Salva*** It saves the current project. Also create the VTB - linkable file and makes an analysis on the number of PDOs and the transmission time needed to manage them.
📄	**Guarda file di Configurazione –** From menu **Opzioni** → **Mostra file finale** Allow to see the real configuration file that will be linked in the VTB project

Aggiungi nodo da EDS – From menu **Modifica** → **Aggiungi nodo**
Adds a CanOpen node, taking the basic configuration from an EDS file.

Aggiungi nodo da libreria – From menu **Libreria** → **Prendi nodo da libreria**
Adds a CanOpen node, using a ready made configuration, saved in the library.

Modifica nodo – From menu **Modifica** → **Modifica nodo**
Allow to edit/change the node id.

Taglia nodo – From menu **Modifica** → **Rimuovi nodo**
Cut out the selected node from the configuration.

Copia nodo – From menu **Modifica** → **Copia nodo**
Copy the selected node to the clipboard, with all its characteristics, ready to be pasted as a new node

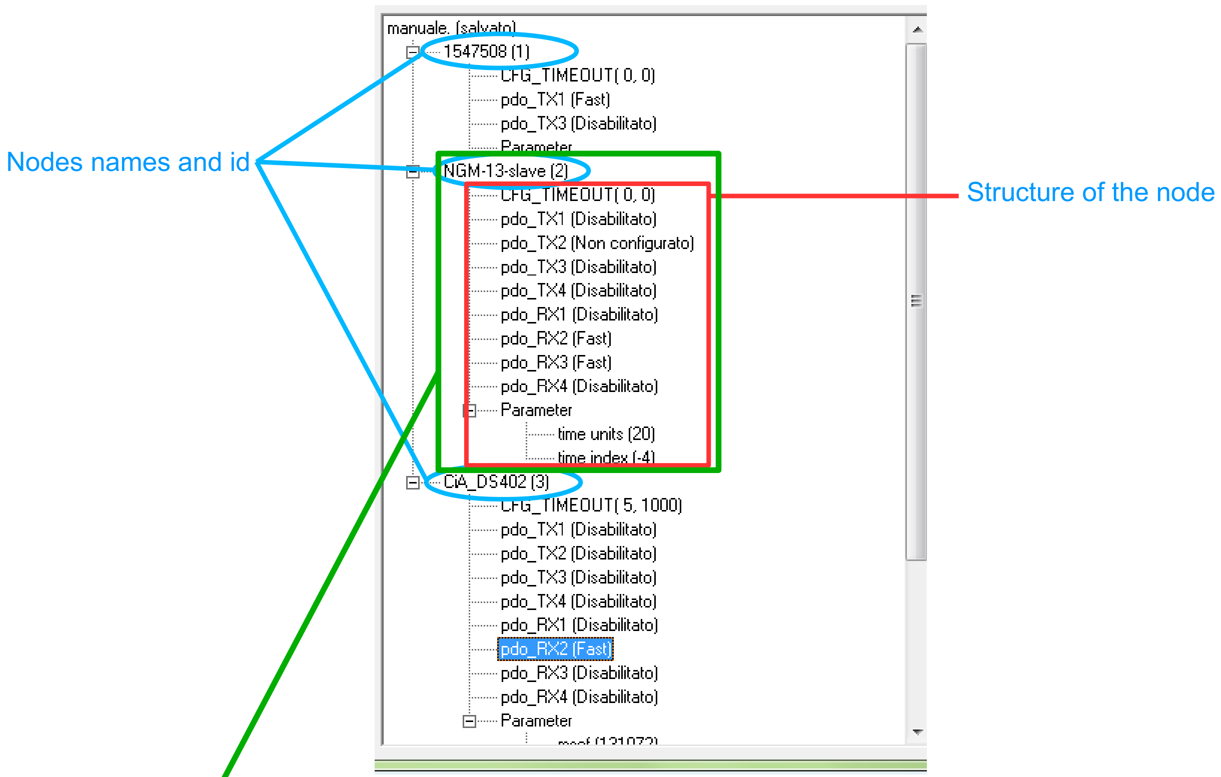
Incolla nodo – From menu **Modifica** → **Incolla nodo**
Add the selected and copied node into the configuration, allowing to set a new node id.

Esporta in libreria – From menu Libreria → **Salva nodo in libreria**
Exports the selected node to the library, making it usable for other similar projects.

Modifica libreria – From menu Libreria → **Modifica nodo in libreria**
Opens the library in edit mode, for modify some of the saved node.

4.2 Configuration panel

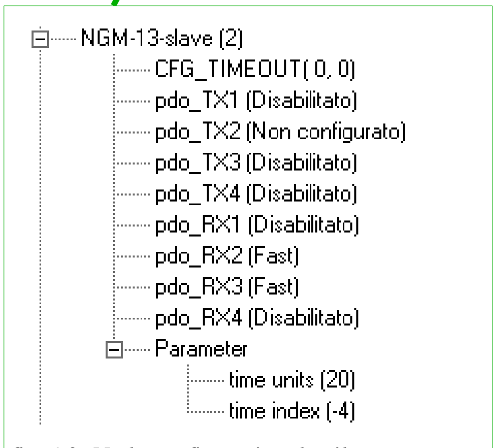
The Configuration panel it's the real work area of the application. It shows the actual net configuration and makes possible to modify it. Every work that we must do in our file, can be made here, selecting various informations and using the tool bar or menu command.



```
manuale. (salvato)
⊟ 1547508 (1)
      CFG_TIMEOUT( 0, 0)
      pdo_TX1 (Fast)
      pdo_TX3 (Disabilitato)
      Parameter
⊟ NGM-13-slave (2)
      CFG_TIMEOUT( 0, 0)
      pdo_TX1 (Disabilitato)
      pdo_TX2 (Non configurato)
      pdo_TX3 (Disabilitato)
      pdo_TX4 (Disabilitato)
      pdo_RX1 (Disabilitato)
      pdo_RX2 (Fast)
      pdo_RX3 (Fast)
      pdo_RX4 (Disabilitato)
    ⊟ Parameter
          time units (20)
          time index (-4)
⊟ CiA_DS402 (3)
      CFG_TIMEOUT( 5, 1000)
      pdo_TX1 (Disabilitato)
      pdo_TX2 (Disabilitato)
      pdo_TX3 (Disabilitato)
      pdo_TX4 (Disabilitato)
      pdo_RX1 (Disabilitato)
      pdo_RX2 (Fast)
      pdo_RX3 (Disabilitato)
      pdo_RX4 (Disabilitato)
    ⊟ Parameter
          moof (131072)
```

fig. 4.2: Configuration panel

The tree structure displays the composition of the configured CanOpen net.
The root it's the project name and every branch represents a node.



```
⊟ NGM-13-slave (2)
      CFG_TIMEOUT( 0, 0)
      pdo_TX1 (Disabilitato)
      pdo_TX2 (Non configurato)
      pdo_TX3 (Disabilitato)
      pdo_TX4 (Disabilitato)
      pdo_RX1 (Disabilitato)
      pdo_RX2 (Fast)
      pdo_RX3 (Fast)
      pdo_RX4 (Disabilitato)
    ⊟ Parameter
          time units (20)
          time index (-4)
```

fig. 4.3: Node configuration detail

In example, taken from a real configuration, we have:

- NGM13 it's the slave/node type;
- node id it's 2;

After, there is a very useful function: configuration time-out (see 4.2.1).
Then we can see the defined structure of the device we are using, taken from its EDS file.
Here we have a device that have four TX (transmit) and four RX (receive) PDO.
The direction of the communication is define from the slave point of view, means that TX goes from slave to master while RX from master to slave.
Every PDO is showed with its active mode (see 4.2.2).
The last information we can see, are parameters that the function will set after making the slave configuration.

4.2.1 Configuration Time-out

In normal situations, when the machine starts, it's usual to have the master CNC that makes initialization functions in a very short time, like the NG35 for ex., while one or more

slaves starts in a longer time, like drives.
Promax system makes the net configuration as the first initialization operation, then in this case could be possible that a device will be not configured, cause it's not ready to accept master instructions. Therefore the device will not be able to exchange PDO informations, like interpolated target quotas for ex.
How can we avoid this situation?
With the Configuration time-out. Using this strategy, when the master try to set the slave and the slave doesn't answer, it will try once again after a little while, then can try again and again...
How many times the master must try to reach the slave and how many time have to wait from one and another try?
These are the data that are closed inside the round parenthesis: making a double-click, will open a form, where can be setted in the right way.

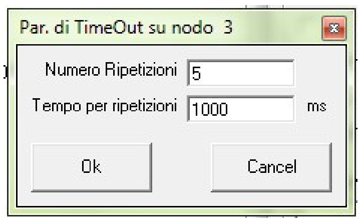

Par. di TimeOut su nodo 3
Numero Ripetizioni: 5
Tempo per ripetizioni: 1000 ms

fig. 4.4: Time out details

"Numero Ripetizioni" Repeat Number, are the times that the master will try to reach the slave.
"Tempo per ripetizioni" Time for repeat, expressed in ms, is the time between two repetitions.
In the example showed, the master will try to reach the slave (node 3) 5 times, wait 1000ms (1sec) every time. It means that the slave must have a start time, less then 5 sec.

4.2.2 Change PDO configuration

Now, let's see how we can change or add a PDO configuration in a slave device.

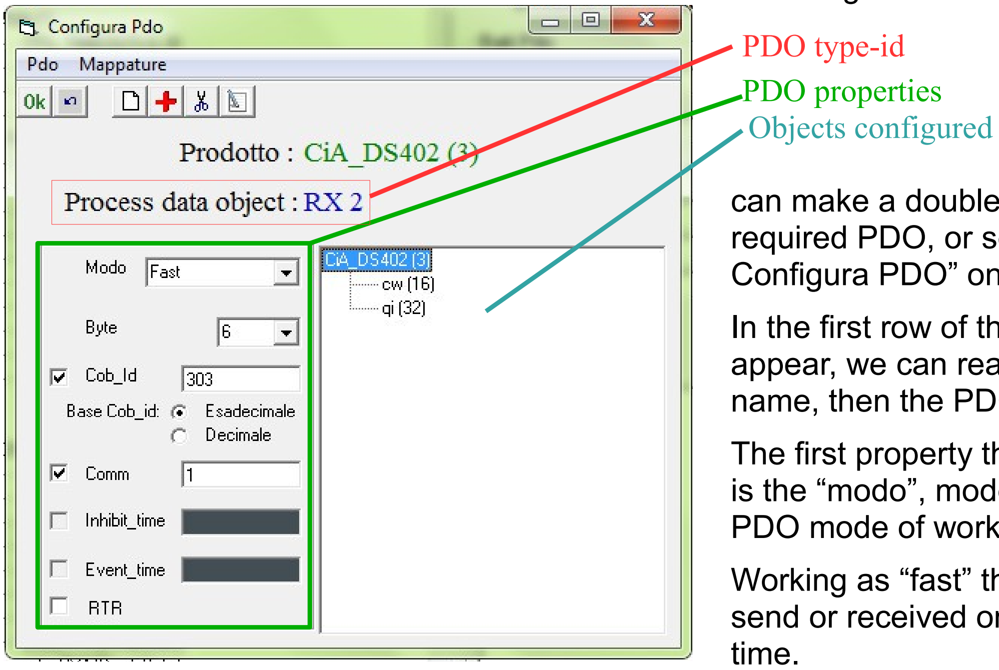

Configura Pdo
Prodotto : CiA_DS402 (3)
Process data object : RX 2
Modo: Fast
Byte: 6
Cob_Id [✓]: 303
Base Cob_id: [✓ Esadecimale] Decimale
Comm [✓]: 1
Inhibit_time [ ]
Event_time [ ]
RTR [ ]

fig. 4.5: PDO config. details

To change the PDO configuration, we can make a double-click on the required PDO, or select "Modifica → Configura PDO" on the menu.

In the first row of the form that will appear, we can read the device name, then the PDO type-id.

The first property that can be setted, is the "modo", mode, that define the PDO mode of work.

Working as "fast" the PDO will be send or received on every sample time.

Working as "slow", can be useful to discharge the net traffic. When there is more then one slow PDO, on every sample time will be send, from master to slave or vice versa, only one of this. Means that if we have 3 slow PDO, we'll have all data updated only after 3 sample times. Clearly, this mode can be used only with low rate changing data.

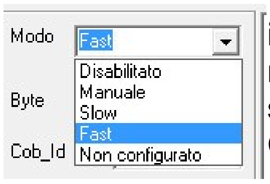


fig. 4.6: PDOs mode

"Disabilitato", disabled, means that the PDO will be disabled. It is useful to cut out some ready configured PDO in a device, but the possibility to do that, must be verified on the device manual.

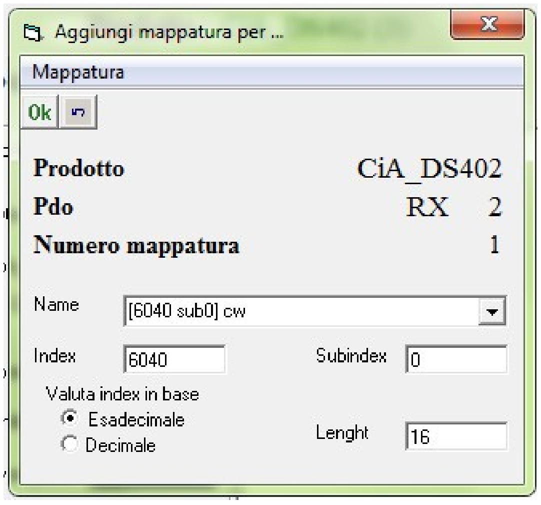
“Non configurato”, not configured, leave the standard device PDO configuration active. No change on this PDO, will be made by configuration function.

Last possibility is “Manuale” manual, means that the PDO declaration will be from another node, don't create a new PDO. In this case, the Cob-id of the PDO, will be the same as the node that have the right configuration. This type of work, will be explained better later (see 7)

Then, there is the possibility to set the number of byte that will be exchanged in this PDO, with a maximum of 8 as standard, the Cob-id that is the index of the PDO in the net and will be created automatically by the software. As mentioned talking about the mode, there are some cases where we have to set manually the Cob-id, but we will turn back later on it.

In the right side of the form, we can see the data that are configured to be exchanged in the actual PDO and the dimension in bits.
Here we have an object named cw 16 bits long and another object named qi 32 bits long.

With a double-click on one of it, will open a form where we can see and set the object that we want to manage. It shows the name of the object, that we can change to easy remember it, the Can Open object dictionary index and sub-index and the length, expressed as bit.



Prodotto CiA_DS402
Pdo RX 2
Numero mappatura 1

Name: [6040 sub0] cw
Index: 6040
Subindex: 0
Valuta index in base: [✓ Esadecimale] Decimale
Lenght: 16

fig. 4.7: Object details

Changing the index, always expressed as hexadecimal number, the software will search the relative object inside the EDS file of the device, with its standard name.
Also, it's possible to browse inside the list, to search what we need.

Clicking the “OK” button all changes will be confirmed, while the “↺” button will discard any of it.

Pushing the “+” button on the previous form, a new object will be inserted inside the PDO, and opening it in the same way, we can configure the object to be exchanged.

About the object configuration inside a PDO we have to always remember two things: the first, as define in CanOpen specifications, the max length of a PDO will be 8 bytes, 64 bits, we can't exceed this length. We can use less of it, but not more than this. Second, activating a PDO, normally the application will show a configuration taken directly on the EDS file.
We can try to change it, but we have to verify if the device we are setting, have freely configurable PDOs or not. Otherwise, our operation will the ignored, in the best case, or generate an error during the operation, in the worst one.

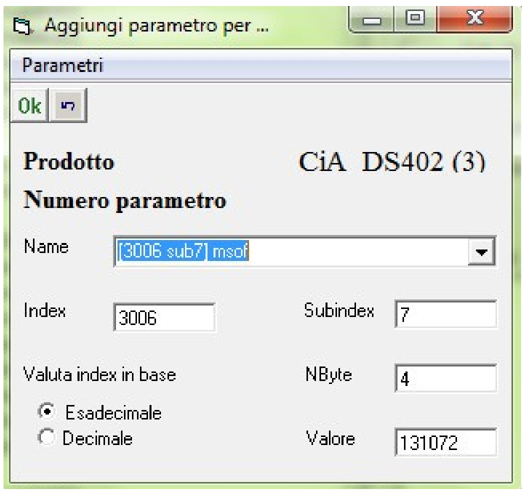
4.2.3 Parameters

In the lower part of a node-branch, there are parameters.

Parameters are device-objects that we want to write during the node initialization.
It can be useful, for ex, to set the output state when the device loose the communication with the master, or to initialize some particular function inside the slave device.

Making a double-click on the parameters branch, it's possible to add a new parameter in a form similar to the PDO configuration while making the same on a parameter, it's possible to open and change its characteristics.

The only main difference between a PDO and a parameter property, is that we have to set a value that will written into the device.


Aggiungi parametro per ...

Parametri

Ok

Prodotto CiA DS402 (3)

Numero parametro

Name	[3006 sub7] msof		
Index	3006	Subindex	7
Valuta index in base		NByte	4
[✓ Esadecimale] Decimale		Valore	131072

fig. 4.8: Parameter details

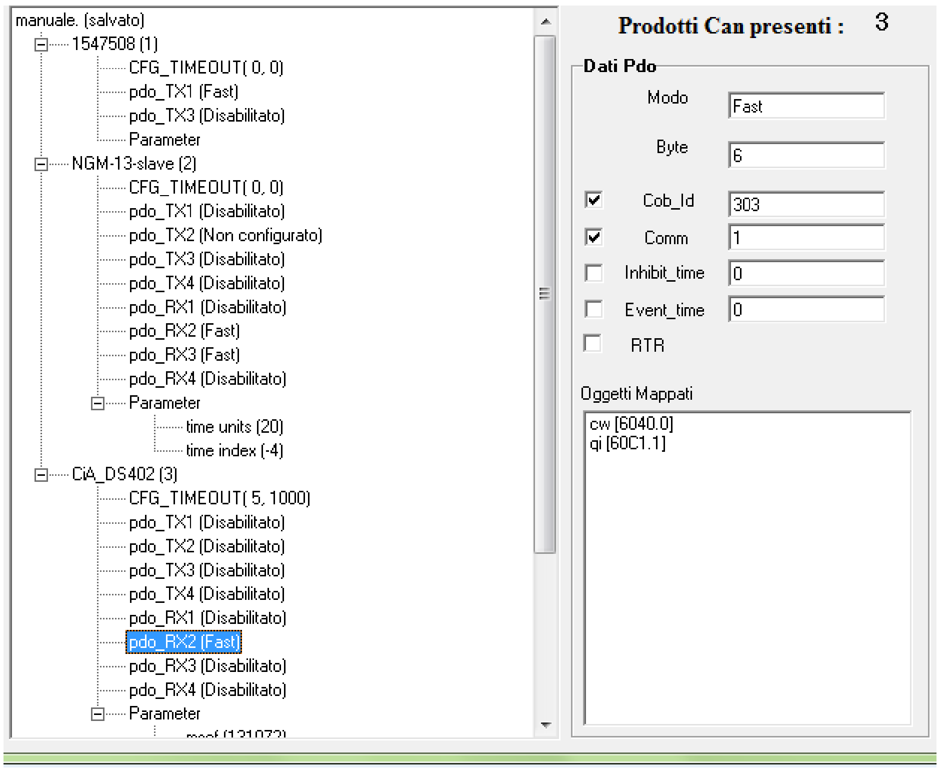
4.3 PDO characteristics panel

This panel make a summary of PDO properties.

On the upper edge, it shows the number of devices that the project contains.

Then, selecting a PDO of one device, it will show all the information about it, like Mode, Cob-id, length and so on.

It's only a visualization, it isn't possible to change any property on it.



```
manuale. (salvato)
⊟ 1547508 (1)
      CFG_TIMEOUT( 0, 0)
      pdo_TX1 (Fast)
      pdo_TX3 (Disabilitato)
      Parameter
⊟ NGM-13-slave (2)
      CFG_TIMEOUT( 0, 0)
      pdo_TX1 (Disabilitato)
      pdo_TX2 (Non configurato)
      pdo_TX3 (Disabilitato)
      pdo_TX4 (Disabilitato)
      pdo_RX1 (Disabilitato)
      pdo_RX2 (Fast)
      pdo_RX3 (Fast)
      pdo_RX4 (Disabilitato)
    ⊟ Parameter
          time units (20)
          time index (-4)
⊟ CiA_DS402 (3)
      CFG_TIMEOUT( 5, 1000)
      pdo_TX1 (Disabilitato)
      pdo_TX2 (Disabilitato)
      pdo_TX3 (Disabilitato)
      pdo_TX4 (Disabilitato)
      pdo_RX1 (Disabilitato)
      pdo_RX2 (Fast)
      pdo_RX3 (Disabilitato)
      pdo_RX4 (Disabilitato)
    ⊟ Parameter
```

Prodotti Can presenti : 3

Dati Pdo

	Modo	Fast
	Byte	6
✓	Cob_Id	303
✓	Comm	1
☐	Inhibit_time	0
☐	Event_time	0
☐	RTR	

Oggetti Mappati

```
cw [6040.0]
qi [60C1.1]
```

fig. 4.9: PDO properties summary

5 Project management

5.1 How to add a node

To add a node in a project, we have two main possibility:

- add a node using a EDS file,“+EDS” button;
- add a node from library;

About the using the library, we can talk about later (see 6).

Using the EDS file, we can see a list of the devices that are present in the system, where it's possible to choose the required one. To add a new device, third parties or a new Promax one as the same, just add the EDS file in the *\eds* directory inside the installation directory of Configuratore. After, the new device will appear in the list as the filename.

After selected the device, set the node of it and confirm the operation. If there is another device on the same node, the system will show an error message.

When the new node appears in the configuration panel, it can be possible to make any type of changes, allowed from the EDS file.

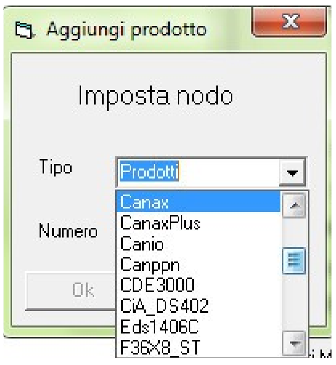
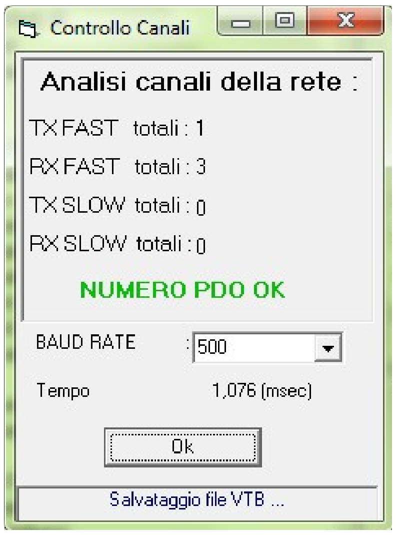


fig. 5.1: EDS file list

5.2 Saving and verify a project

Pushing the save button, the system will save the project, giving the same name as the VTB project name, only using a specific file type.
Moreover, after saving, it'll show a form where can be found some information about the project.


Controllo Canali

Analisi canali della rete :

TX FAST totali : 1

RX FAST totali : 3

TX SLOW totali : 0

RX SLOW totali : 0

NUMERO PDO OK

BAUD RATE : 500

Tempo 1,076 (msec)

Ok

Salvataggio file VTB ...

fig. 5.2: Project analysis

It's very useful, because it shows the total number of TX and RX PDO configured in fast mode, the slow mode ones and the total time that will be taken by the communication management, depending on the baud rate.

Here we have to make a little note: in Promax system CNC, the max PDO number that can be managed by a master is 10, mixed RX and TX. Only NG35 can manage a higher number of PDO in particular cases. Refer to the VTB and NG35 to know how.

Therefore, if we have setted more then 10 PDO, the system will show a warning message and we'll must make some optimization in the project.

Also, selecting the baud rate that we have to use in the VTB application, it'll show the total time taken by the communication management, that can be compared with the sample time of the application.
If the total time exceed the sample time, the application will not be able to manage any other instruction.

To ensure that the communication can't cause some problem on other sides of the VTB project, the communication time must not exceed the 60% of the sample time.

Choosing “salva con nome, the only difference is that the system gives the possibility to set a different name to save it.

5.3 Opening a project

Starting the Configuratore inside VTB IDE, it will automatically open the configuration related to the VTB project.

Also there is the possibility to open a different project, using the commands described above (see 4.1).

6 Library using

Configuratore gives the possibility to save a ready made and tested configuration, that could be used inside any other project that have to use the same particular device.

It will be created a list of ready made configurations, named library, that can be modified, selected and so on.

6.1 Save a configuration

Using the “Esporta in libreria”, export to library, button (see 4.1), it is possible to save a device configuration.

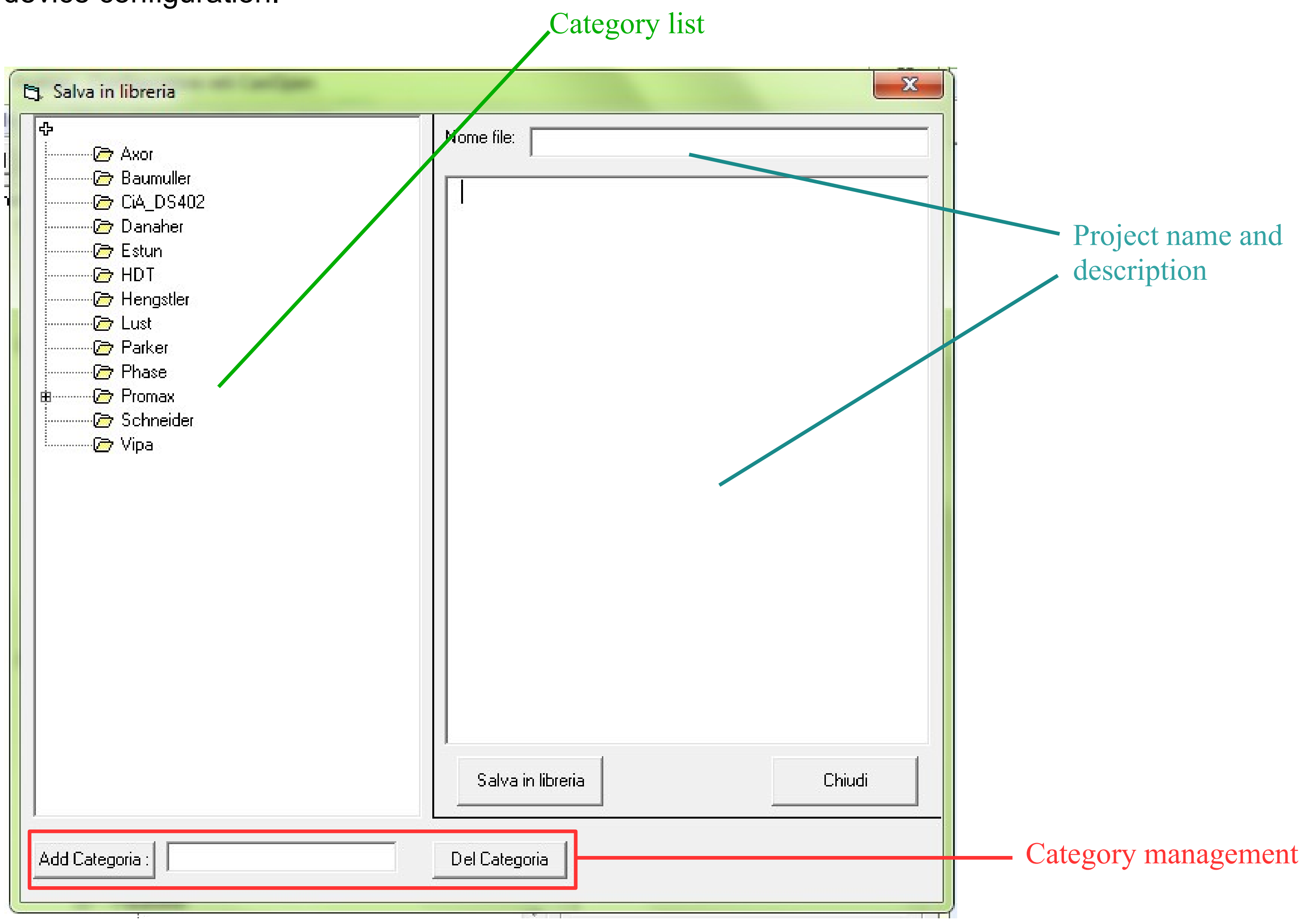


fig. 6.1: Save in library form

The library as organized with a categories list, normally the producer name, and also sub categories, normally the device name, inside every category there are some ready made configurations.

Using the “Add categoria”, add category, button and writing a new name in the text box, it's possible to create new categories.

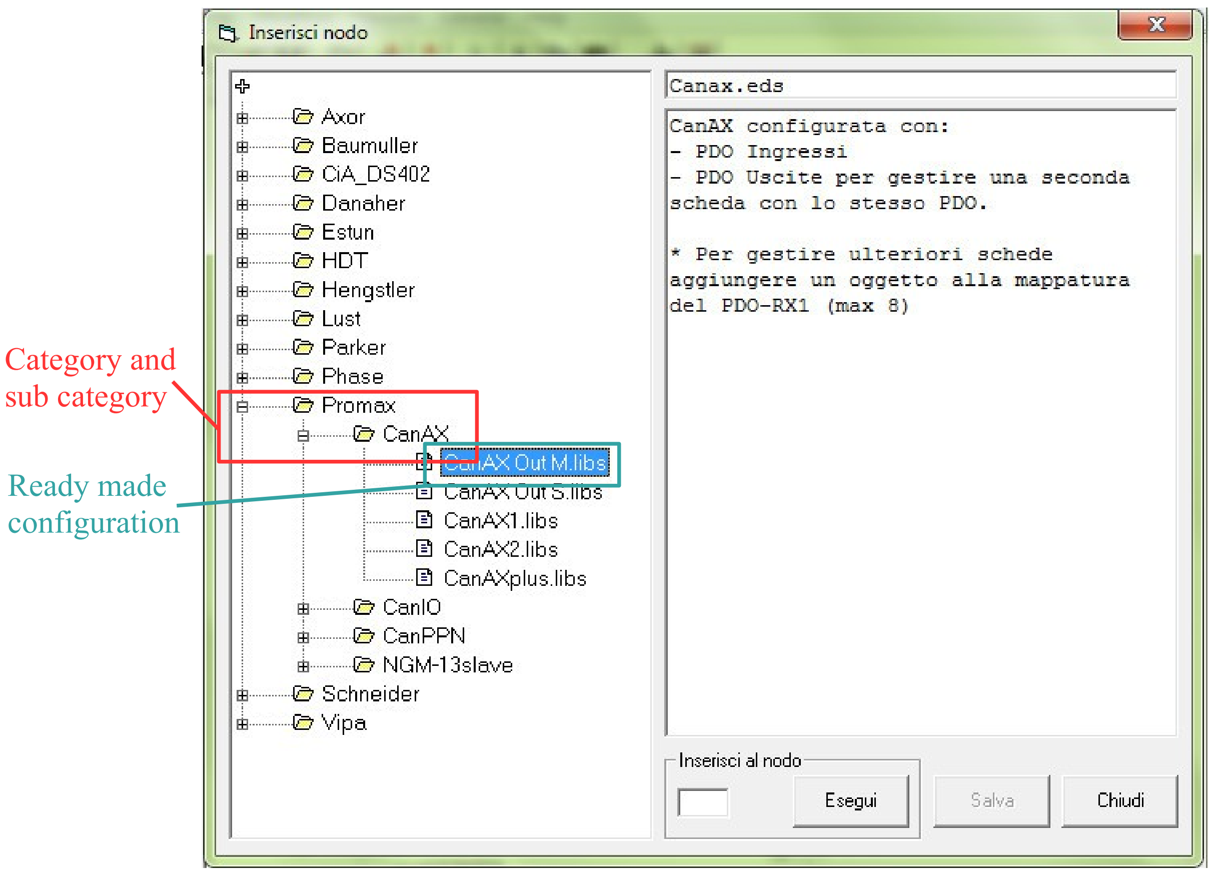
Then the configuration can be saved with a name and a description that can be inserted in the appropriated space. “Salva in libreria”, save in library, button complete the saving operation.

6.2 Open a configuration

The “+lib” button (see 4.1), gives the access to the library, where a particular configuration can be selected.
Every configuration will be displayed with its description, allowing to see the already

configured objects or properties, for ex.

After the configuration as been inserted in the project, it can be modified as required in the normal manner.



```
Canax.eds
```

```
CanAX configurata con:
- PDO Ingressi
- PDO Uscite per gestire una seconda
scheda con lo stesso PDO.

* Per gestire ulteriori schede
aggiungere un oggetto alla mappatura
del PDO-RX1 (max 8)
```

fig. 6.2: Insert from library form

7 Note on “packed” PDOs

A particular case of PDOs are “packed” PDO.

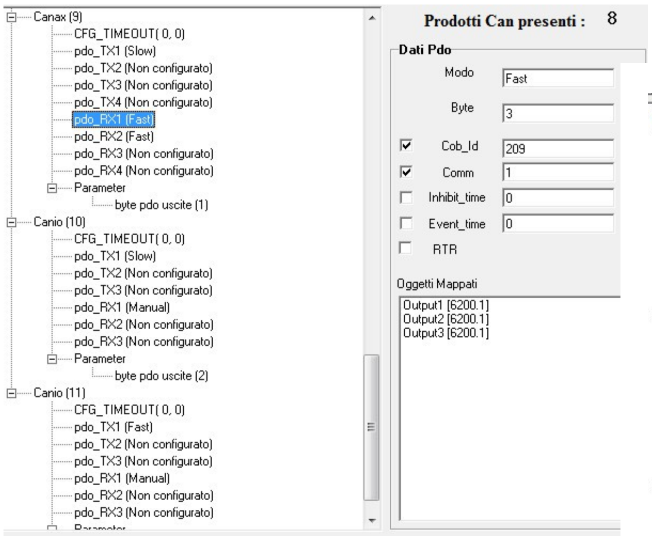
This a useful manner to manage data useful for more than one node, interpolated quotas or output state for ex, as a single PDO. It can be used only with RX PDOs (from master to slave).

As we know, the length of the PDOs data is 8 byte, but the various objects that can be exchanged inside it, have less length, for ex interpolated quotas have 4 byte length, or Promax slaves output state is 1-byte long.

How can we fully use the entire PDO length, without any “hole” in it?

Using the packed PDO.

What does it means? Consider the three images below,



```
Canax (9)
    CFG_TIMEOUT( 0, 0)
    pdo_TX1 (Slow)
    pdo_TX2 (Non configurato)
    pdo_TX3 (Non configurato)
    pdo_TX4 (Non configurato)
    pdo_RX1 (Fast)
    pdo_RX2 (Fast)
    pdo_RX3 (Non configurato)
    pdo_RX4 (Non configurato)
    Parameter
        byte pdo uscite (1)
Canio (10)
    CFG_TIMEOUT( 0, 0)
    pdo_TX1 (Slow)
    pdo_TX2 (Non configurato)
    pdo_TX3 (Non configurato)
    pdo_RX1 (Manual)
    pdo_RX2 (Non configurato)
    pdo_RX3 (Non configurato)
    Parameter
        byte pdo uscite (2)
Canio (11)
    CFG_TIMEOUT( 0, 0)
    pdo_TX1 (Fast)
    pdo_TX2 (Non configurato)
    pdo_TX3 (Non configurato)
    pdo_RX1 (Manual)
    pdo_RX2 (Non configurato)
    pdo_RX3 (Non configurato)
```

Prodotti Can presenti : 8

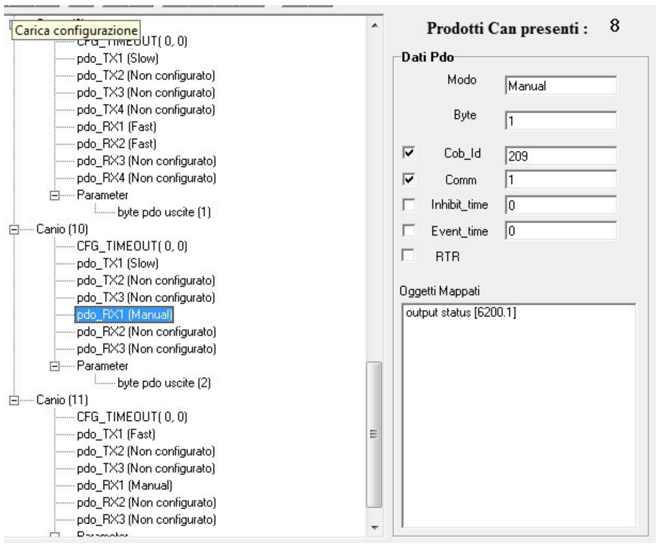
Dati Pdo

Modo	Fast
Byte	3
☑ Cob_Id	209
☑ Comm	1
☐ Inhibit_time	0
☐ Event_time	0
☐ RTR	

Oggetti Mappati

```
Output1 [6200.1]
Output2 [6200.1]
Output3 [6200.1]
```

fig. 7.1: First node Pdo declaration



```
    pdo_TX1 (Slow)
    pdo_TX2 (Non configurato)
    pdo_TX3 (Non configurato)
    pdo_TX4 (Non configurato)
    pdo_RX1 (Fast)
    pdo_RX2 (Fast)
    pdo_RX3 (Non configurato)
    pdo_RX4 (Non configurato)
    Parameter
        byte pdo uscite (1)
Canio (10)
    CFG_TIMEOUT( 0, 0)
    pdo_TX1 (Slow)
    pdo_TX2 (Non configurato)
    pdo_TX3 (Non configurato)
    pdo_RX1 (Manual)
    pdo_RX2 (Non configurato)
    pdo_RX3 (Non configurato)
    Parameter
        byte pdo uscite (2)
Canio (11)
    CFG_TIMEOUT( 0, 0)
    pdo_TX1 (Fast)
    pdo_TX2 (Non configurato)
    pdo_TX3 (Non configurato)
    pdo_RX1 (Manual)
    pdo_RX2 (Non configurato)
    pdo_RX3 (Non configurato)
```

Prodotti Can presenti : 8

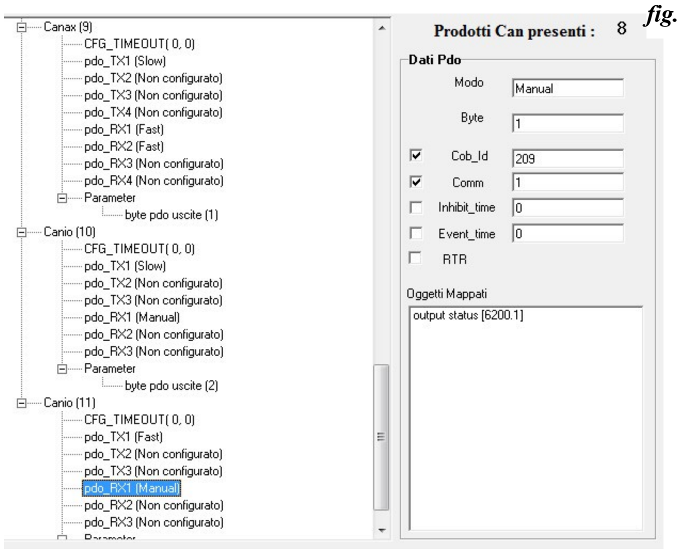
Dati Pdo

Modo	Manual
Byte	1
☑ Cob_Id	209
☑ Comm	1
☐ Inhibit_time	0
☐ Event_time	0
☐ RTR	

Oggetti Mappati

```
output status [6200.1]
```

fig. 7.2: Second node PDO declaration



```
Canax (9)
    CFG_TIMEOUT( 0, 0)
    pdo_TX1 (Slow)
    pdo_TX2 (Non configurato)
    pdo_TX3 (Non configurato)
    pdo_TX4 (Non configurato)
    pdo_RX1 (Fast)
    pdo_RX2 (Fast)
    pdo_RX3 (Non configurato)
    pdo_RX4 (Non configurato)
    Parameter
        byte pdo uscite (1)
Canio (10)
    CFG_TIMEOUT( 0, 0)
    pdo_TX1 (Slow)
    pdo_TX2 (Non configurato)
    pdo_TX3 (Non configurato)
    pdo_RX1 (Manual)
    pdo_RX2 (Non configurato)
    pdo_RX3 (Non configurato)
    Parameter
        byte pdo uscite (2)
Canio (11)
    CFG_TIMEOUT( 0, 0)
    pdo_TX1 (Fast)
    pdo_TX2 (Non configurato)
    pdo_TX3 (Non configurato)
    pdo_RX1 (Manual)
    pdo_RX2 (Non configurato)
    pdo_RX3 (Non configurato)
```

Prodotti Can presenti : 8

Dati Pdo

Modo	Manual
Byte	1
☑ Cob_Id	209
☑ Comm	1
☐ Inhibit_time	0
☐ Event_time	0
☐ RTR	

Oggetti Mappati

```
output status [6200.1]
```

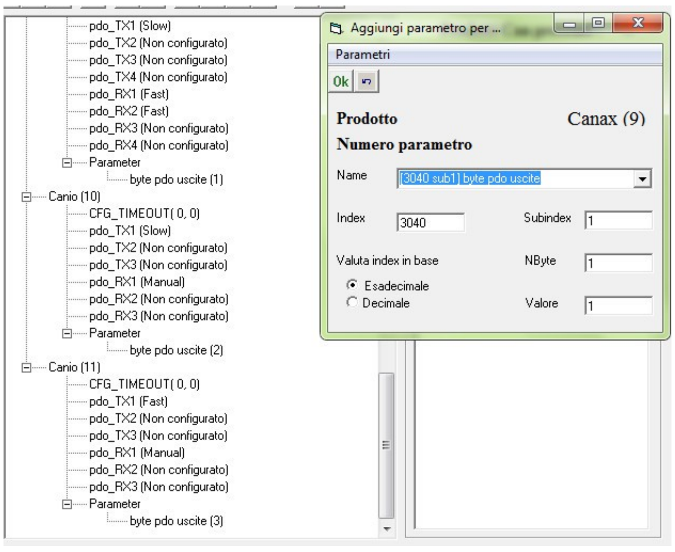
fig. 7.3: Third node PDO declaration

There are displayed three I/O slaves, old Promax devices named CanIo and CanAx, that have the RX1 PDO configured. As we can see the first slave (CanAx, node 9) have the PDO setted to “Fast”, while the two others have the same one as “Manual”.
But have a look on the summary on the right of each image. All three PDOs have the same Cob-id, but the first have three objects configured, while the others have only an objects.

The first one is the “packed” PDO, this PDO will be read from all three slaves for the output state. But how can they split its own data from the others?
On every slave there is a parameter that specify this information.

As it is displayed in the figures above and showed in fig. 7.4, on every slave there is a parameter (“byte PDO uscite”) that specify to the devices, where in the PDO data, starts the byte that it have to use. As showed, the first slave, will have his own data starting from the first byte, the second slave starting from the second byte and so on.

Using this PDO management, we can have only a PDO for 8-byte output block, also on 8 different slaves and not 8 different PDOs, one for each one. It means low net charging.
The real PDO is defined in the first slave, where is the “Fast” mode. In the others, the “Manual” mode, tells to the device that the PDO where will be the data, is already defined somewhere, no new PDO have to be defined. The Cob-id have to be inserted manually as the same as “fast” PDO. The parameter tells to the device, what are its data within the PDO.
Therefore the master will send all data in one PDO, that every slaves will read it and select his own data.



```
    pdo_TX1 (Slow)
    pdo_TX2 (Non configurato)
    pdo_TX3 (Non configurato)
    pdo_TX4 (Non configurato)
    pdo_RX1 (Fast)
    pdo_RX2 (Fast)
    pdo_RX3 (Non configurato)
    pdo_RX4 (Non configurato)
  ⊟ Parameter
      byte pdo uscite (1)
⊟ Canio (10)
    CFG_TIMEOUT( 0, 0)
    pdo_TX1 (Slow)
    pdo_TX2 (Non configurato)
    pdo_TX3 (Non configurato)
    pdo_RX1 (Manual)
    pdo_RX2 (Non configurato)
    pdo_RX3 (Non configurato)
  ⊟ Parameter
      byte pdo uscite (2)
⊟ Canio (11)
    CFG_TIMEOUT( 0, 0)
    pdo_TX1 (Fast)
    pdo_TX2 (Non configurato)
    pdo_TX3 (Non configurato)
    pdo_RX1 (Manual)
    pdo_RX2 (Non configurato)
    pdo_RX3 (Non configurato)
  ⊟ Parameter
      byte pdo uscite (3)
```

Aggiungi parametro per ...

Parametri

Ok

Prodotto Canax (9)

Numero parametro

Name	[3040 sub1] byte pdo uscite		
Index	3040	Subindex	1
Valuta index in base		NByte	1
[✓ Esadecimale] Decimale		Valore	1

fig. 7.4: Output byte selection parameter

The same thing can be made with the interpolated target quotas. This data is 4-bytes long, then we can have one PDO with at least 2-devices data inside.

Not all devices supports “packed” PDOs, refer to the devices user manual to verify it.

Main index