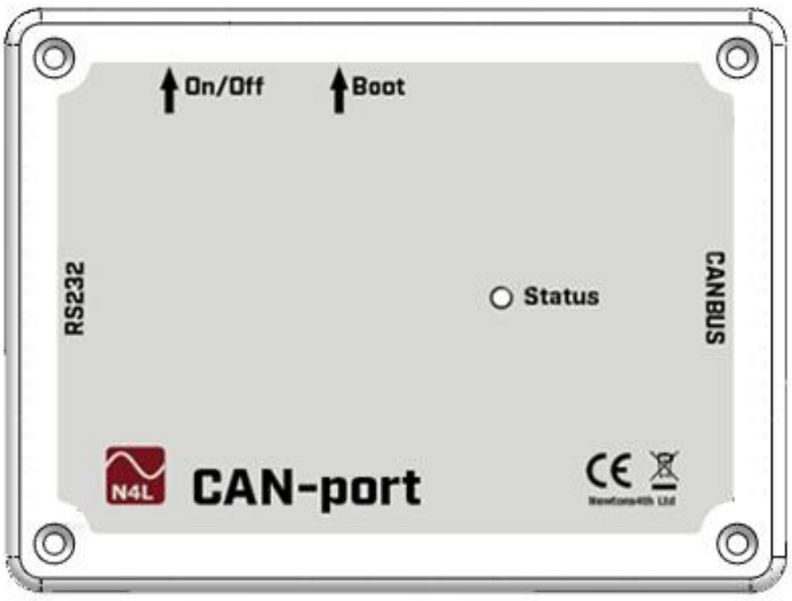
Newtons4th Ltd



CAN-port

CAN to RS232 CONVERTER

USER MANUAL

Version 2.0 May 2019

Contents

About

This user manual was written for **CAN-port** firmware version 1.0.0 and describes the general features, usage, specifications of the **CAN-port** unit, including detailed descriptions of the communications commands used by the unit.

Introduction

The **CAN-port** unit is designed as an interface between a CAN network and a Newtons4th PPA series Power Analyzer. The unit receives commands from the CAN network or via serial (RS232) and then processes these commands and controls and instructs the PPA; responses from the PPA are reformatted and split into multiple messages and placed onto the CAN network. The **CAN-port** unit can be used to control the PPA remotely by using CAN messages to send instructions to the PPA, additionally the **CAN-port** unit can be instructed to send commands as soon as it is powered on that automatically sets the PPA up for logging.

Technical Specification

Microcontroller	NXP LPC2194/01
Memory	32kByte EEPROM Atmel AT24C32B (via I2C)
CAN	High-Seed CAN ISO 11898-2 Transceiver NXP TJA1040T Bit rates 40kbit/s – 1Mbit/s No termination
RS232	RxD and TxD serial connections with DTC and CTS shorted
Status Indication	Duo LED
Supply Voltage	8-30V DC
Current Consumption	Max 70mA at 12V
Operating Temperature	-40 to +85 °C (-40 to +185 °F)
Relative Humidity	15-90%, not condensing
Size	130 x 82 x 44 (W x D x H)
Weight	150g
EMC	EN 61326-1:2013-07 EC Directive 2004/108/EG
Ingress Protection (IEC 60529)	IP20

Pin Out

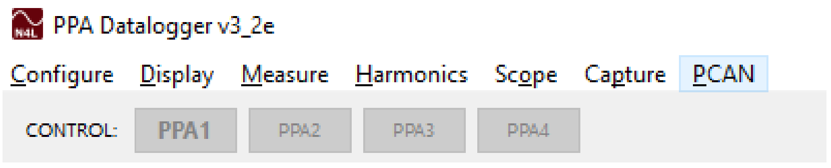
CAN Connector		RS232 Connector	
Pin	Function	Pin	Function
1	Not Used	1	Not Used
2	CAN L	2	Rx Data
3	GND	3	Tx Data
4	Not Used	4	Not Used
5	Not Used	5	GND
6	GND	6	Not Used
7	CAN H	7	RTS
8	Not Used	8	CTS
9	Not Used	9	Not Used

Quick Start

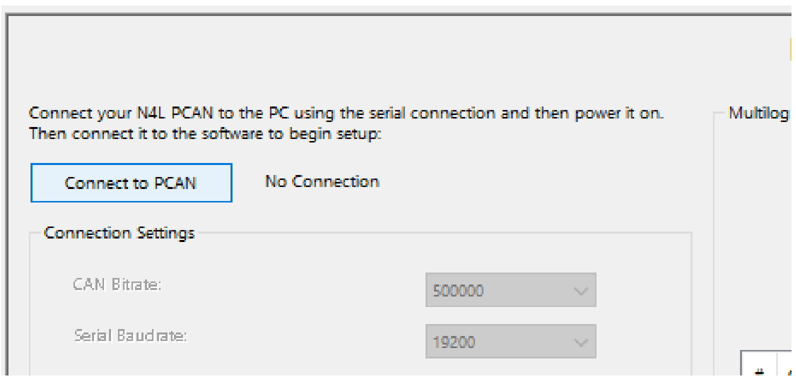
This section of the manual describes how to setup and begin using the **CAN-port** unit and PPA in a CAN network.

Connecting to PPA Datalogger

Ensure the **CAN-port** unit is powered on; the status light should be blinking green. Once the **CAN-port** unit is on, connect an RS232 cable to the serial port and connect it to your PC. Open the PPA Datalogger software and navigate to the PCAN menu. **(You require a special version of Datalogger, please contact N4L sales department to obtain this version)**



Press the Connect button to bring up the connection window where you will set up your serial connection to the **CAN-port** unit.

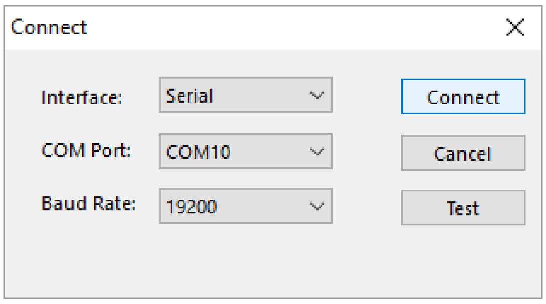


Connect your N4L PCAN to the PC using the serial connection and then power it on. Then connect it to the software to begin setup:

CAN Bitrate: 500000

Serial Baudrate: 19200

Select serial, the correct com port and the correct baud rate (default: 19200). Use the Test button to check if the software is able to form a connection and communicate with the **CAN-port** unit; once the settings are correct press Connect.

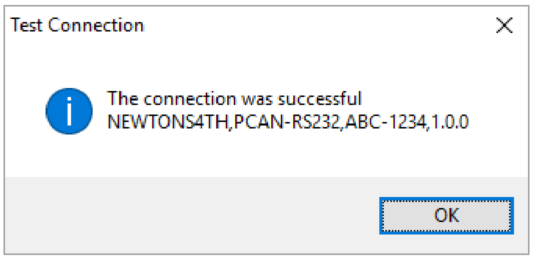


Interface: Serial

COM Port: COM10

Baud Rate: 19200

If the connection was a success you will see a message telling you that the connection was successful and the details of the unit you connected to.

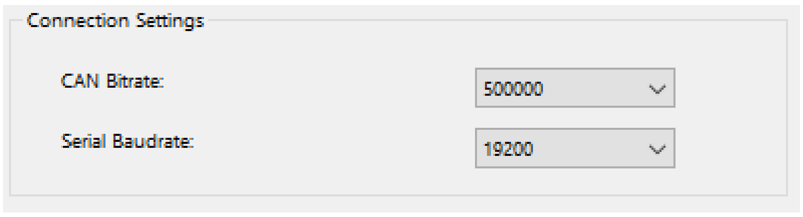

Test Connection

The connection was successful
NEWTONS4TH,PCAN-RS232,ABC-1234,1.0.0

Adjusting the CAN-port's settings using PPA Datalogger

The PPA Datalogger software now allows you to edit the settings that your **CAN-port** unit will remember when it powers on. Some of these settings will help set up the **CAN-port** and others tell the **CAN-port** to set up the PPA.

The **Connection Settings** allows you to set the CAN Bitrate and Serial Baudrate

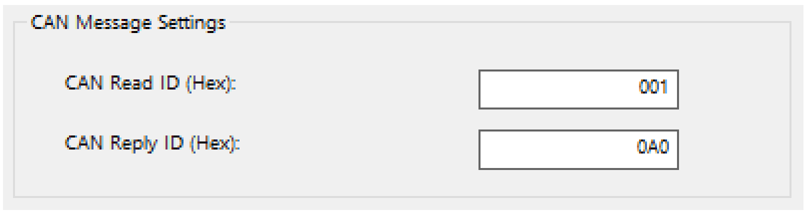

Connection Settings

CAN Bitrate: 500000
Serial Baudrate: 19200

The **CAN Message Settings** allow you to edit how the **CAN-port** unit interacts with the rest of the CAN network.

CAN Read ID is the CAN message ID (in hex) that the **CAN-port** unit will accept CAN messages from.

CAN Reply ID is the CAN message ID (in hex) that the **CAN-port** unit will send its CAN messages to.


CAN Message Settings

CAN Read ID (Hex): 001
CAN Reply ID (Hex): 0A0

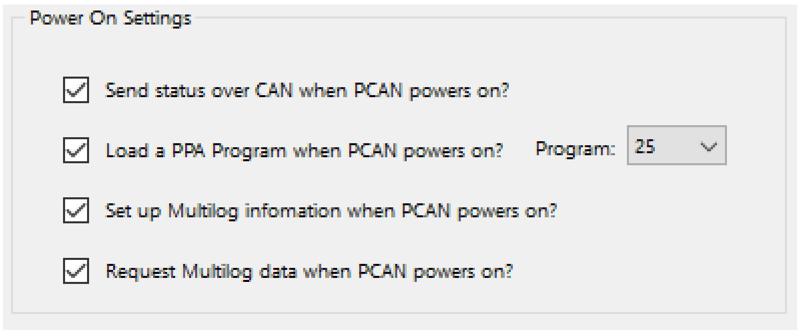
The **Power On Settings** allows you to configure what the **CAN-port** unit does when it first powers on.

Send status when PCAN powers on? causes the **CAN-port** unit to send a status message (the equivalent of the STATUS? command) over the CAN network as soon as it powers on.

Load a PPA Program when PCAN powers on? allows you to send a command over the serial connection to the PPA that gets it to load one of its stored programs, allowing the **CAN-port** to send a command when it powers on to set up all the PPA's settings.

Set up Multilog information when PCAN powers on? allows the **CAN-port** to send commands when it powers on to set up the PPA's Multilog list, ready for logging data.

Request Multilog data when PCAN powers on? causes the **CAN-port** to request an initial set of data from the PPA as soon as the multilog information has been set into the PPA upon power up. If the **CAN-port** is set to repeatedly request data from the PPA, this will start the loop of data being requested.


Power On Settings

- [✓] Send status over CAN when PCAN powers on?
- [✓] Load a PPA Program when PCAN powers on? Program: 25
- [✓] Set up Multilog infomation when PCAN powers on?
- [✓] Request Multilog data when PCAN powers on?

The **Multilog Settings** allows you to configure how the **CAN-port** unit handles multilogs, including which multilogs to set, and how to format them

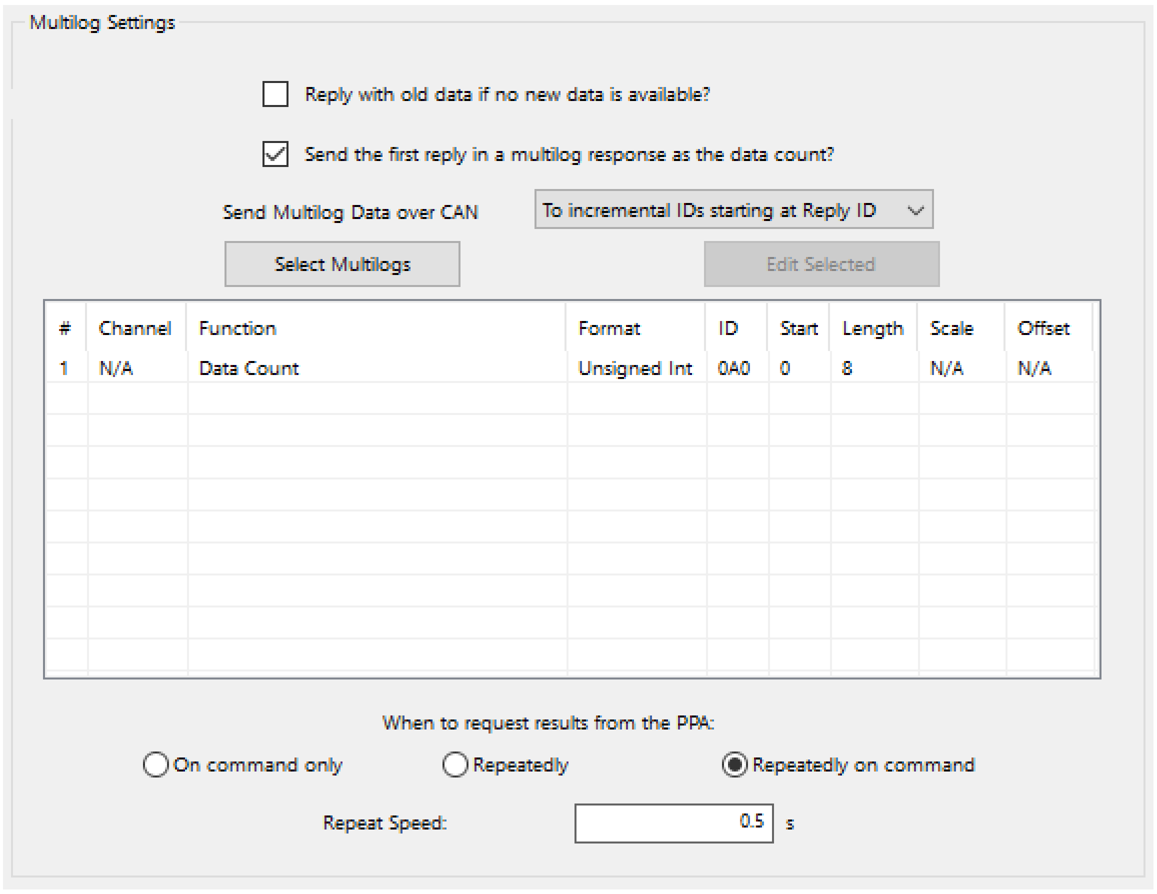
Reply with old data if no new data is available? ensures the **CAN-port** unit sends a reply when multilog data is requested, even if no new data is available. The previous set of data is sent in case a new set of data is not available.

Send the first reply in a multilog response as the data count? causes the **CAN-port** unit to send a counter as the first CAN message in a formatted set of multilog responses; the counter increases each time new data is received from the PPA.

Send multilog data over CAN… allows you to change where the multilog data will be sent, per multilog value. You can either send all responses to the CAN Reply ID, to custom IDs set per multilog value, or to incremental IDs per message, starting at the CAN Reply ID.

When to request result from the PPA… determines when the **CAN-port** would request multilog data for you. You can have multilog data return only when requested, repeatedly at a set interval or repeatedly (on command) when told to using the MLSTART/MLSTOP commands at a set interval.

Repeat Speed is the speed at which the **CAN-port** unit sends multilog data if its available.



Multilog Settings

[ ] Reply with old data if no new data is available?

[✓] Send the first reply in a multilog response as the data count?

Send Multilog Data over CAN: To incremental IDs starting at Reply ID

#	Channel	Function	Format	ID	Start	Length	Scale	Offset
1	N/A	Data Count	Unsigned Int	0A0	0	8	N/A	N/A

When to request results from the PPA:

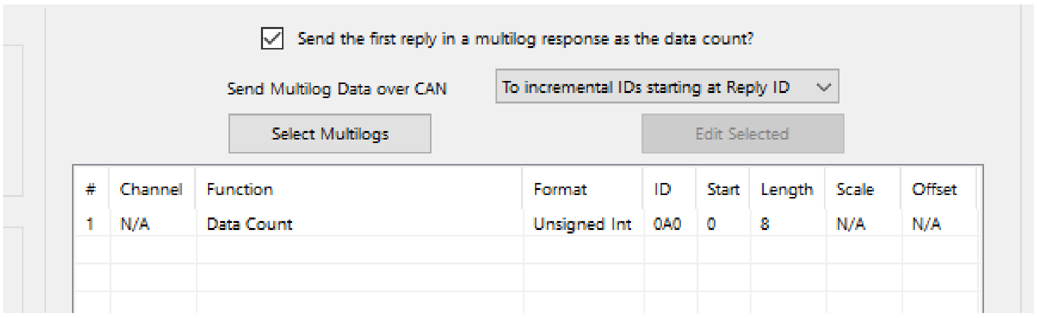
On command only   Repeatedly   [✓ Repeatedly on command]

Repeat Speed: 0.5 s

Setting up Multilogs for the CAN-port in PPA Datalogger

Using PPA Datalogger you can select up to 60 Multilog parameters that can be set into the **CAN-port** unit which it will, when powered on, send the corresponding commands to the PPA to set it up to log those parameters.

To choose the multilog parameters press the Select Multilogs



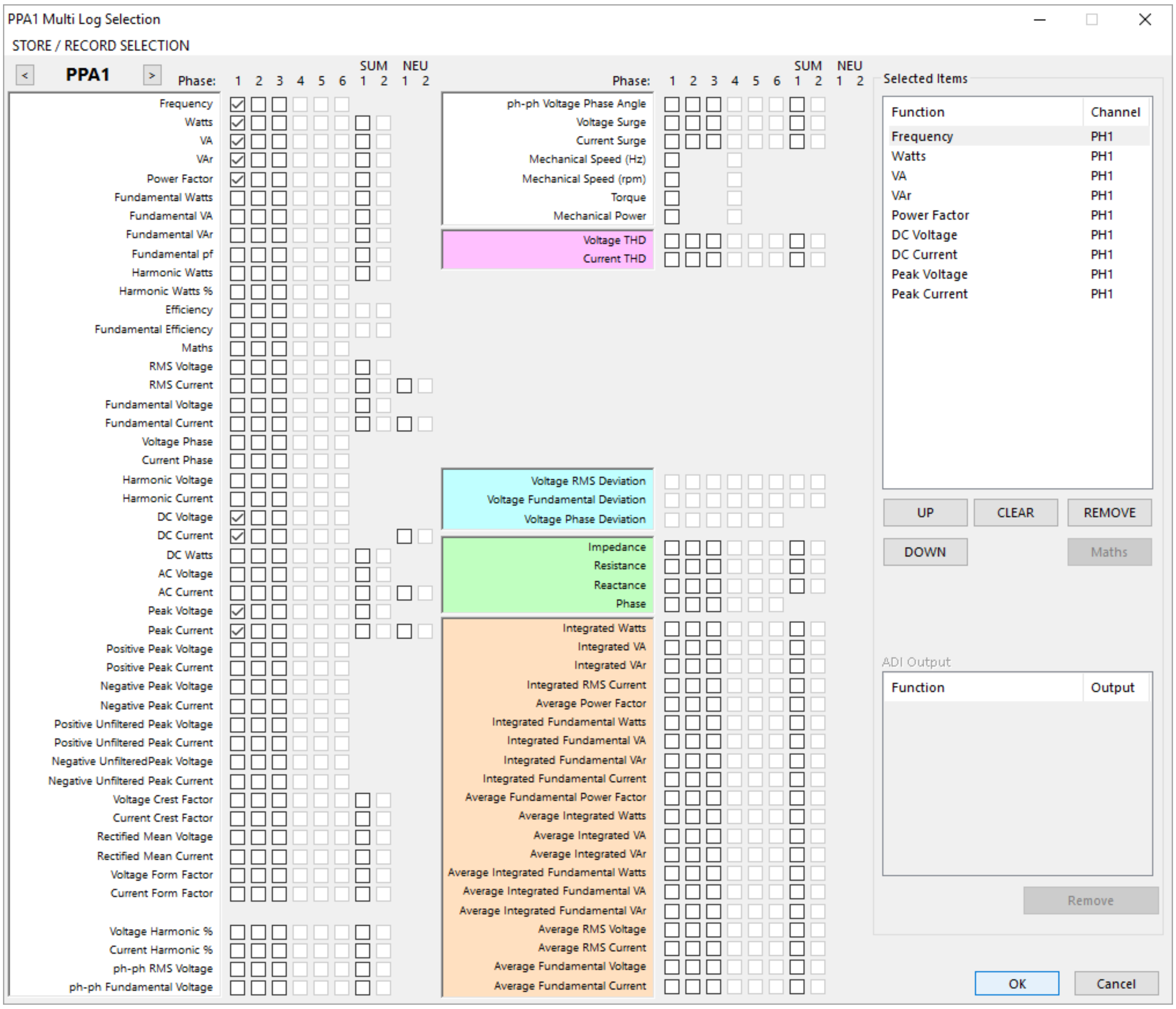
✓ Send the first reply in a multilog response as the data count?

Send Multilog Data over CAN: To incremental IDs starting at Reply ID

Select Multilogs | Edit Selected

#	Channel	Function	Format	ID	Start	Length	Scale	Offset
1	N/A	Data Count	Unsigned Int	0A0	0	8	N/A	N/A

PPA Datalogger will display a wide selection of Multilog parameters, check the values you want to log and then press OK.



PPA1 Multi Log Selection

STORE / RECORD SELECTION

PPA1 — Phase: 1 2 3 4 5 6 SUM 1 2 NEU 1 2

✓ Frequency, ✓ Watts, ✓ VA, ✓ VAr, ✓ Power Factor, Fundamental Watts, Fundamental VA, Fundamental VAr, Fundamental pf, Harmonic Watts, Harmonic Watts %, Efficiency, Fundamental Efficiency, Maths, RMS Voltage, RMS Current, Fundamental Voltage, Fundamental Current, Voltage Phase, Current Phase, Harmonic Voltage, Harmonic Current, ✓ DC Voltage, ✓ DC Current, DC Watts, AC Voltage, AC Current, ✓ Peak Voltage, ✓ Peak Current, Positive Peak Voltage, Positive Peak Current, Negative Peak Voltage, Negative Peak Current, Positive Unfiltered Peak Voltage, Positive Unfiltered Peak Current, Negative UnfilteredPeak Voltage, Negative Unfiltered Peak Current, Voltage Crest Factor, Current Crest Factor, Rectified Mean Voltage, Rectified Mean Current, Voltage Form Factor, Current Form Factor, Voltage Harmonic %, Current Harmonic %, ph-ph RMS Voltage, ph-ph Fundamental Voltage

ph-ph Voltage Phase Angle, Voltage Surge, Current Surge, Mechanical Speed (Hz), Mechanical Speed (rpm), Torque, Mechanical Power, Voltage THD, Current THD, Voltage RMS Deviation, Voltage Fundamental Deviation, Voltage Phase Deviation, Impedance, Resistance, Reactance, Phase, Integrated Watts, Integrated VA, Integrated VAr, Integrated RMS Current, Average Power Factor, Integrated Fundamental Watts, Integrated Fundamental VA, Integrated Fundamental VAr, Integrated Fundamental Current, Average Fundamental Power Factor, Average Integrated Watts, Average Integrated VA, Average Integrated VAr, Average Integrated Fundamental Watts, Average Integrated Fundamental VA, Average Integrated Fundamental VAr, Average RMS Voltage, Average RMS Current, Average Fundamental Voltage, Average Fundamental Current

Selected Items

Function	Channel
Frequency	PH1
Watts	PH1
VA	PH1
VAr	PH1
Power Factor	PH1
DC Voltage	PH1
DC Current	PH1
Peak Voltage	PH1
Peak Current	PH1

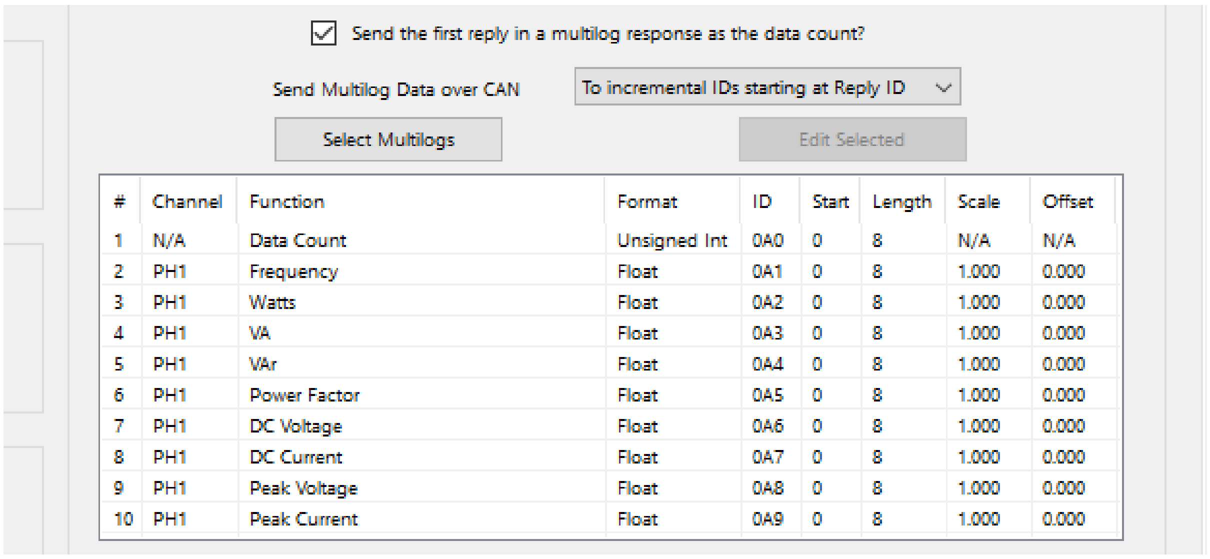
UP | CLEAR | REMOVE | DOWN | Maths

ADI Output

Function	Output

Remove | OK | Cancel

The selected values will now appear in the list on the PCAN Setup page; to change the selected multilog parameters, pres Select Multilogs again and change your selection.

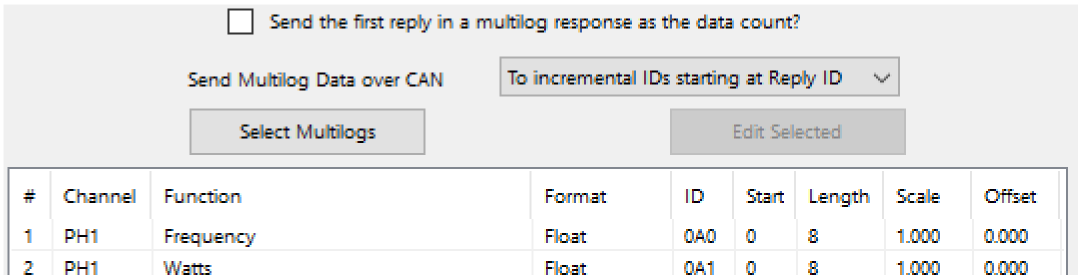


☑ Send the first reply in a multilog response as the data count?

Send Multilog Data over CAN: To incremental IDs starting at Reply ID

#	Channel	Function	Format	ID	Start	Length	Scale	Offset
1	N/A	Data Count	Unsigned Int	0A0	0	8	N/A	N/A
2	PH1	Frequency	Float	0A1	0	8	1.000	0.000
3	PH1	Watts	Float	0A2	0	8	1.000	0.000
4	PH1	VA	Float	0A3	0	8	1.000	0.000
5	PH1	VAr	Float	0A4	0	8	1.000	0.000
6	PH1	Power Factor	Float	0A5	0	8	1.000	0.000
7	PH1	DC Voltage	Float	0A6	0	8	1.000	0.000
8	PH1	DC Current	Float	0A7	0	8	1.000	0.000
9	PH1	Peak Voltage	Float	0A8	0	8	1.000	0.000
10	PH1	Peak Current	Float	0A9	0	8	1.000	0.000

The first value is Data Count, because we have "Send the first reply in a multilog response as the data count?" selected. Un-ticking that removes the data count.

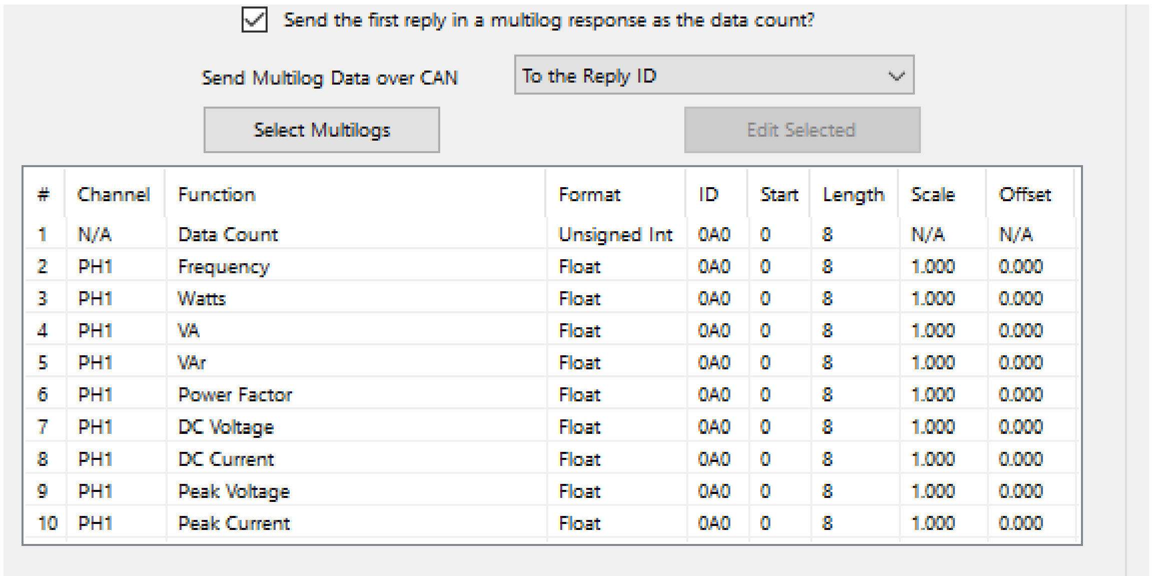


☐ Send the first reply in a multilog response as the data count?

Send Multilog Data over CAN: To incremental IDs starting at Reply ID

#	Channel	Function	Format	ID	Start	Length	Scale	Offset
1	PH1	Frequency	Float	0A0	0	8	1.000	0.000
2	PH1	Watts	Float	0A1	0	8	1.000	0.000

Additionally the ID starts at the CAN Reply ID, and increases each message that will need to be sent across the CAN Network, as "Send Multilog Data over CAN" setting is set to "To incremental IDs starting at Reply ID". Selecting "To the Reply ID" changes the ID value to the CAN Reply ID:

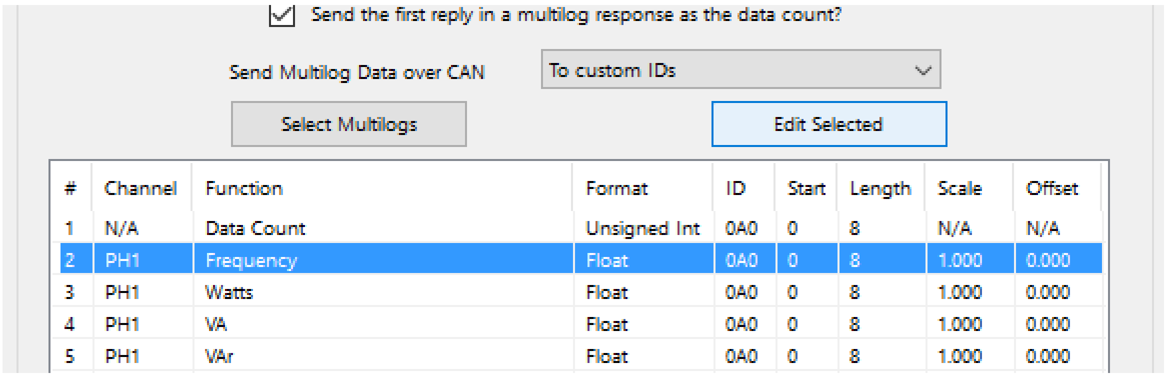


☑ Send the first reply in a multilog response as the data count?

Send Multilog Data over CAN: To the Reply ID

#	Channel	Function	Format	ID	Start	Length	Scale	Offset
1	N/A	Data Count	Unsigned Int	0A0	0	8	N/A	N/A
2	PH1	Frequency	Float	0A0	0	8	1.000	0.000
3	PH1	Watts	Float	0A0	0	8	1.000	0.000
4	PH1	VA	Float	0A0	0	8	1.000	0.000
5	PH1	VAr	Float	0A0	0	8	1.000	0.000
6	PH1	Power Factor	Float	0A0	0	8	1.000	0.000
7	PH1	DC Voltage	Float	0A0	0	8	1.000	0.000
8	PH1	DC Current	Float	0A0	0	8	1.000	0.000
9	PH1	Peak Voltage	Float	0A0	0	8	1.000	0.000
10	PH1	Peak Current	Float	0A0	0	8	1.000	0.000

Selecting Custom allows the ID to be changed using the "Edit Selected" button. To use this button, select a multilog by clicking it (other than Data

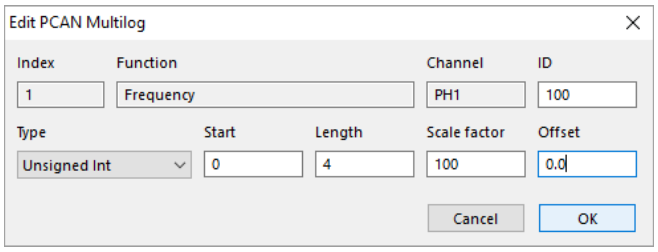

✓ Send the first reply in a multilog response as the data count?

Send Multilog Data over CAN: To custom IDs

Select Multilogs | Edit Selected

#	Channel	Function	Format	ID	Start	Length	Scale	Offset
1	N/A	Data Count	Unsigned Int	0A0	0	8	N/A	N/A
2	PH1	Frequency	Float	0A0	0	8	1.000	0.000
3	PH1	Watts	Float	0A0	0	8	1.000	0.000
4	PH1	VA	Float	0A0	0	8	1.000	0.000
5	PH1	VAr	Float	0A0	0	8	1.000	0.000

Count) and press the Edit Selected Button.

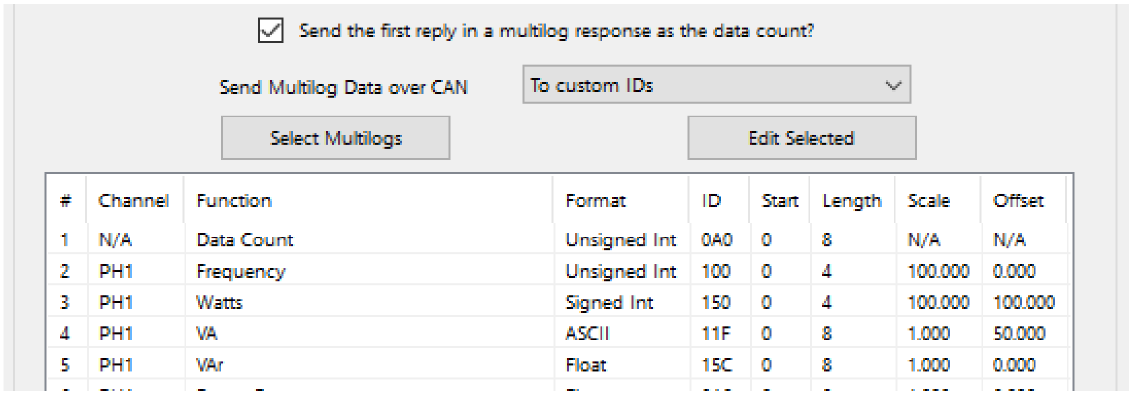

Edit PCAN Multilog

Index	Function	Channel	ID
1	Frequency	PH1	100

Type	Start	Length	Scale factor	Offset
Unsigned Int	0	4	100	0.0

Cancel | OK

Editing a multilog allows you to change the format type, start location in the message, the length of the message this value will use up and a scale factor and offset. Additionally, if "Send Multilog Data over CAN" is set to "To custom IDs" you can also edit the ID the message will be sent to.

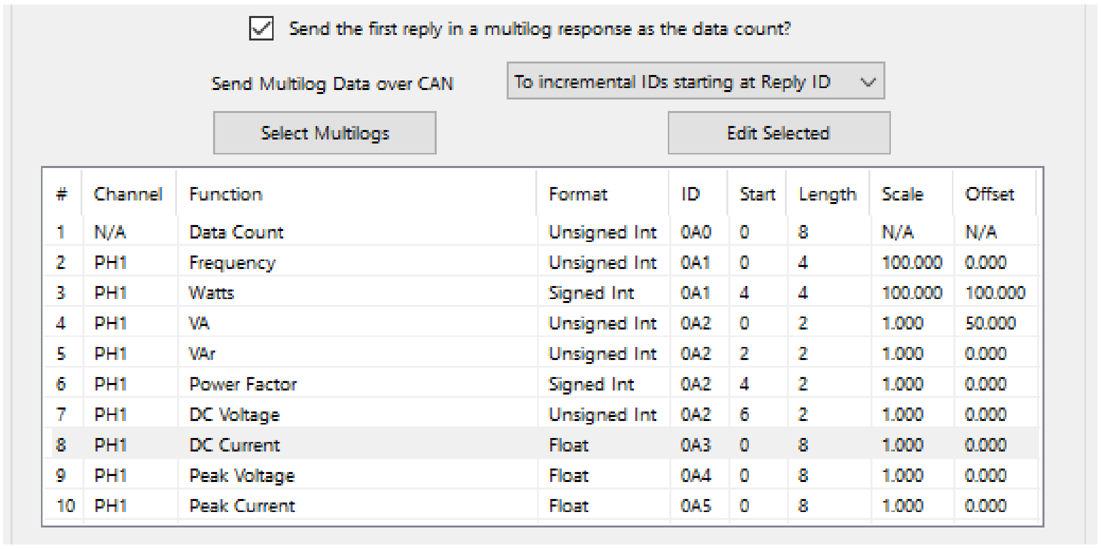

✓ Send the first reply in a multilog response as the data count?

Send Multilog Data over CAN: To custom IDs

Select Multilogs | Edit Selected

#	Channel	Function	Format	ID	Start	Length	Scale	Offset
1	N/A	Data Count	Unsigned Int	0A0	0	8	N/A	N/A
2	PH1	Frequency	Unsigned Int	100	0	4	100.000	0.000
3	PH1	Watts	Signed Int	150	0	4	100.000	100.000
4	PH1	VA	ASCII	11F	0	8	1.000	50.000
5	PH1	VAr	Float	15C	0	8	1.000	0.000

Multiple values can be share a CAN message by ensuring that they do not overlap and fit in 8 bytes. If the "Send Multilog Data over CAN" is set to "To Custom IDs" they also need to share IDs.



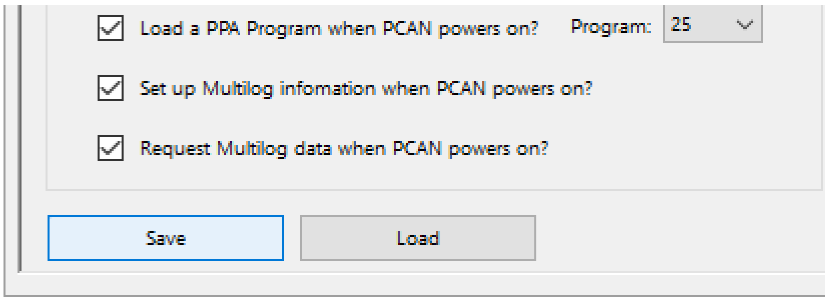
☑ Send the first reply in a multilog response as the data count?

Send Multilog Data over CAN: To incremental IDs starting at Reply ID

#	Channel	Function	Format	ID	Start	Length	Scale	Offset
1	N/A	Data Count	Unsigned Int	0A0	0	8	N/A	N/A
2	PH1	Frequency	Unsigned Int	0A1	0	4	100.000	0.000
3	PH1	Watts	Signed Int	0A1	4	4	100.000	100.000
4	PH1	VA	Unsigned Int	0A2	0	2	1.000	50.000
5	PH1	VAr	Unsigned Int	0A2	2	2	1.000	0.000
6	PH1	Power Factor	Signed Int	0A2	4	2	1.000	0.000
7	PH1	DC Voltage	Unsigned Int	0A2	6	2	1.000	0.000
8	PH1	DC Current	Float	0A3	0	8	1.000	0.000
9	PH1	Peak Voltage	Float	0A4	0	8	1.000	0.000
10	PH1	Peak Current	Float	0A5	0	8	1.000	0.000

Saving and Loading PCAN Setup Settings in PPA Datalogger

To save the setup, press the Save button in the bottom left corner of the PCAN Setup window

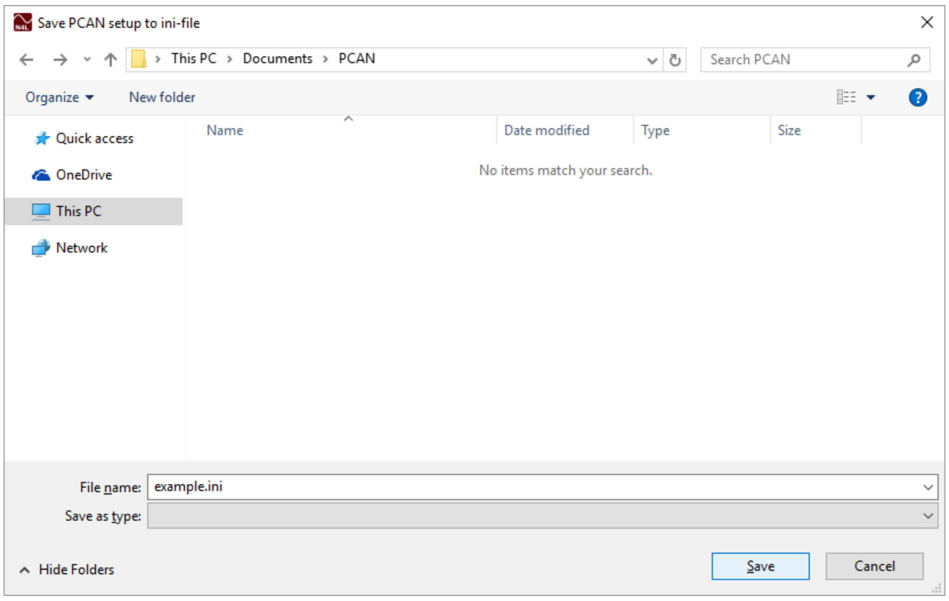


☑ Load a PPA Program when PCAN powers on? Program: 25

☑ Set up Multilog infomation when PCAN powers on?

☑ Request Multilog data when PCAN powers on?

This allows you to save the current setup as a .ini file

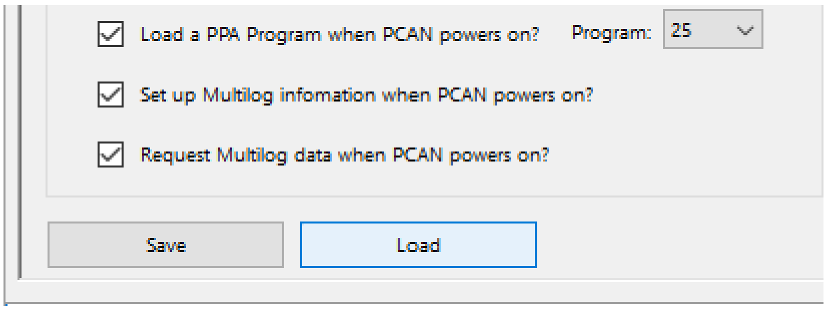


The .ini file is fully editable, and a good way to edit multilogs and other settings

```
[ConnectionSettings]
CANBitrate=500000
SERBaudrate=19200
[CANMessageSettings]
CANReadID=001
CANReplyID=0A0
[PowerOnSettings]
StatusRequest=1
LoadProgram=25
SetupMultilog=1
RequestMultilogData=1
[MultilogSettings]
ReplyWithOldData=0
ReplyDataCount=1
MultilogIDSetting=1
RepeatMultilogRequest=2
RepeatSpeed=0.5
[Multilogs:index=ch,func,type,id,start,len,scale,off]
1=1,1,3,0A1,0,8,1.000000,0.000000
2=1,2,3,0A2,0,8,1.000000,0.000000
3=1,3,3,0A3,0,8,1.000000,0.000000
4=1,4,3,0A4,0,8,1.000000,0.000000
5=1,5,3,0A5,0,8,1.000000,0.000000
6=1,58,3,0A6,0,8,1.000000,0.000000
7=1,59,3,0A7,0,8,1.000000,0.000000
8=1,62,3,0A8,0,8,1.000000,0.000000
9=1,63,3,0A9,0,8,1.000000,0.000000
```

```
[ConnectionSettings]
CANBitrate=500000
SERBaudrate=19200
[CANMessageSettings]
CANReadID=001
CANReplyID=0A0
[PowerOnSettings]
StatusRequest=1
LoadProgram=99
SetupMultilog=1
RequestMultilogData=1
[MultilogSettings]
ReplyWithOldData=0
ReplyDataCount=1
MultilogIDSetting=2
RepeatMultilogRequest=2
RepeatSpeed=0.5
[Multilogs:index=ch,func,type,id,start,len,scale,off]
1=1,1,3,0A1,0,8,1.000000,0.000000
2=1,2,2,0A2,0,4,1.000000,0.000000
3=1,3,1,0A3,4,4,1.000000,0.000000
4=2,4,3,0A4,0,8,1.000000,0.000000
5=2,5,3,0A5,0,8,1.000000,0.000000
6=1,58,0,0A6,0,8,1.000000,0.000000
7=1,59,3,100,0,8,1.000000,0.000000
8=1,62,3,101,0,8,1.000000,0.000000
9=1,63,3,102,0,8,1E4,-5.2E3
```

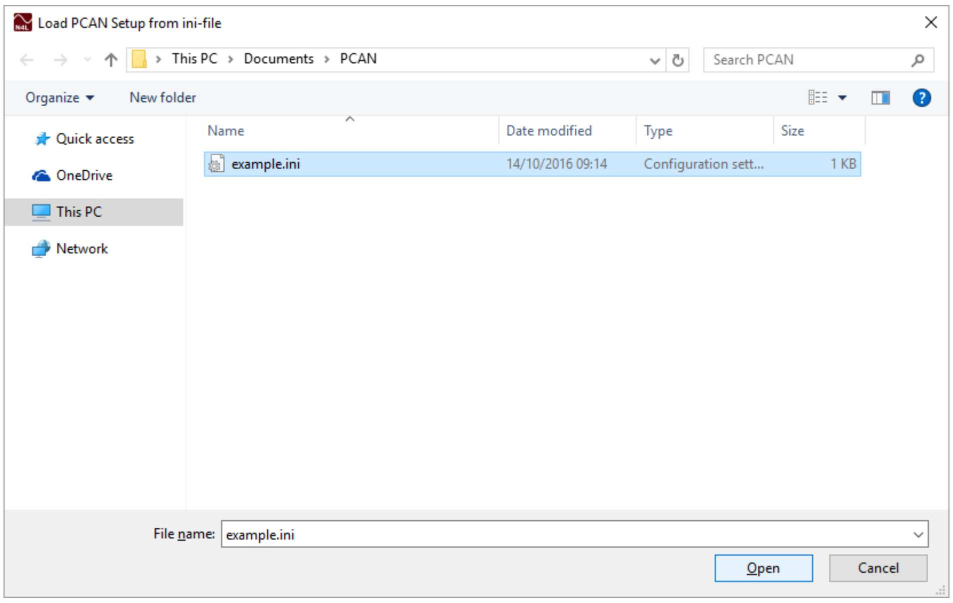
Load an .ini file using the Load button in the bottom left corner



✓ Load a PPA Program when PCAN powers on? Program: 25

✓ Set up Multilog infomation when PCAN powers on?

✓ Request Multilog data when PCAN powers on?



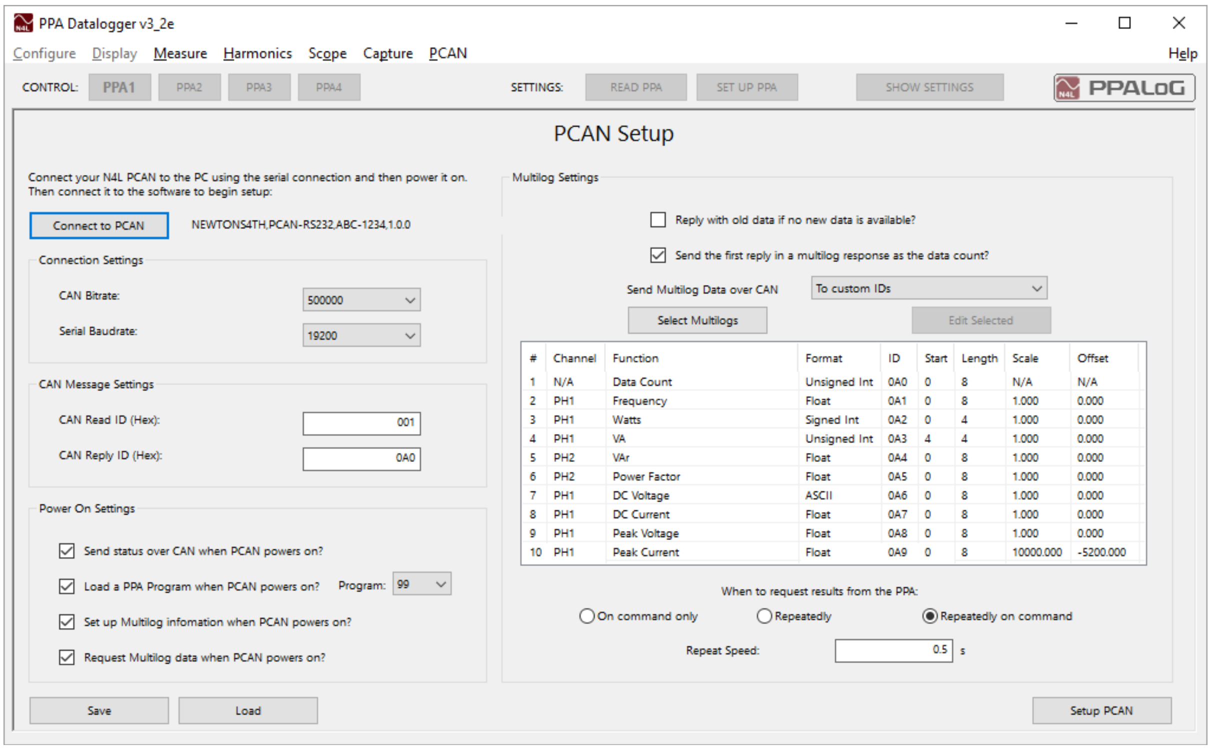
Load PCAN Setup from ini-file

This PC › Documents › PCAN

Name	Date modified	Type	Size
example.ini	14/10/2016 09:14	Configuration sett...	1 KB

File name: example.ini

And the settings will match the values in the .ini file


PPA Datalogger v3_2e
Configure Display Measure Harmonics Scope Capture PCAN Help
CONTROL: PPA1 PPA2 PPA3 PPA4
SETTINGS: READ PPA SET UP PPA SHOW SETTINGS
PPALOG

PCAN Setup

Connect your N4L PCAN to the PC using the serial connection and then power it on. Then connect it to the software to begin setup:

Connect to PCAN — NEWTONS4TH,PCAN-RS232,ABC-1234,1.0.0

Connection Settings

CAN Bitrate:	500000
Serial Baudrate:	19200

CAN Message Settings

CAN Read ID (Hex):	001
CAN Reply ID (Hex):	0A0

Power On Settings

- [✓] Send status over CAN when PCAN powers on?
- [✓] Load a PPA Program when PCAN powers on? Program: 99
- [✓] Set up Multilog infomation when PCAN powers on?
- [✓] Request Multilog data when PCAN powers on?

Multilog Settings

- [ ] Reply with old data if no new data is available?
- [✓] Send the first reply in a multilog response as the data count?

Send Multilog Data over CAN: To custom IDs

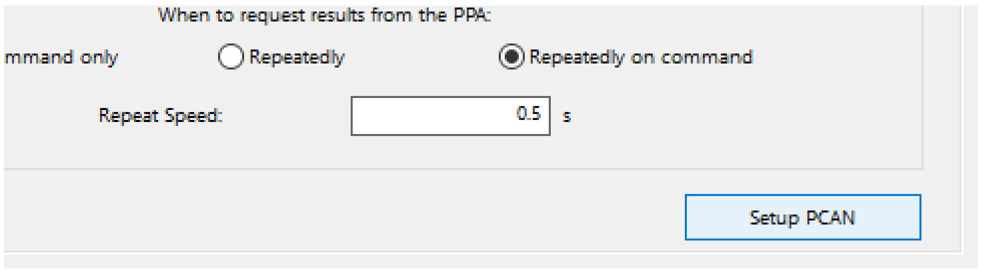
#	Channel	Function	Format	ID	Start	Length	Scale	Offset
1	N/A	Data Count	Unsigned Int	0A0	0	8	N/A	N/A
2	PH1	Frequency	Float	0A1	0	8	1.000	0.000
3	PH1	Watts	Signed Int	0A2	0	4	1.000	0.000
4	PH1	VA	Unsigned Int	0A3	4	4	1.000	0.000
5	PH2	VAr	Float	0A4	0	8	1.000	0.000
6	PH2	Power Factor	Float	0A5	0	8	1.000	0.000
7	PH1	DC Voltage	ASCII	0A6	0	8	1.000	0.000
8	PH1	DC Current	Float	0A7	0	8	1.000	0.000
9	PH1	Peak Voltage	Float	0A8	0	8	1.000	0.000
10	PH1	Peak Current	Float	0A9	0	8	10000.000	-5200.000

When to request results from the PPA: On command only / Repeatedly / [✓ Repeatedly on command]

Repeat Speed: 0.5 s

Finalising the setup with PPA Datalogger

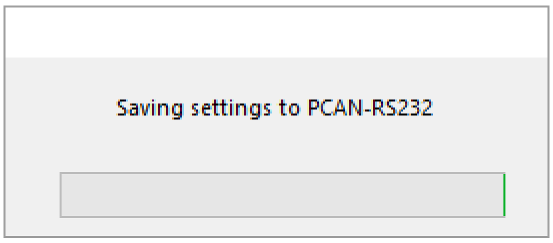
Once all the settings are correct press the Setup PCAN button to transfer the settings to the **CAN-port** unit.



When to request results from the PPA: mmand only / Repeatedly / [✓ Repeatedly on command]

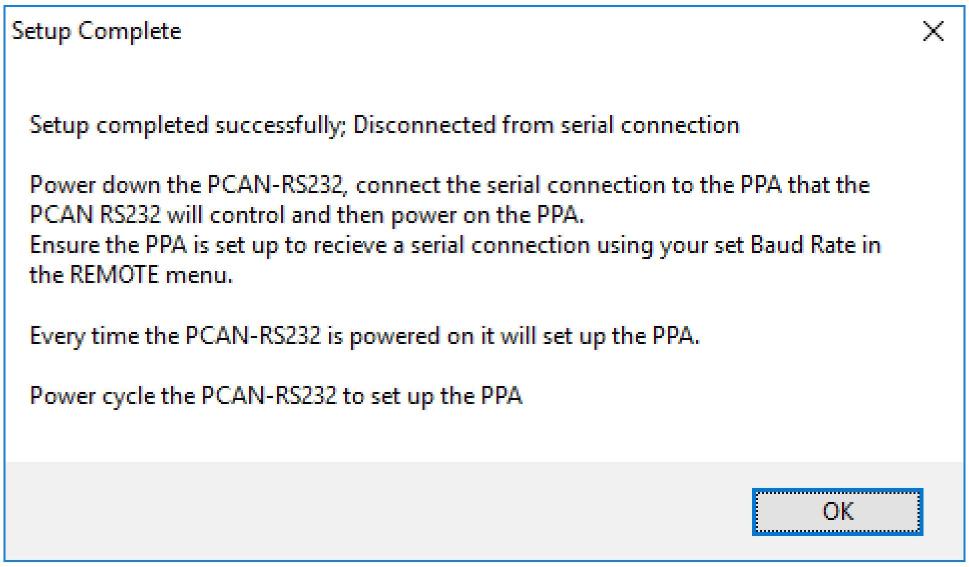
Repeat Speed: 0.5 s

Once PPA Datalogger has sent all the settings to the **CAN-port**



Saving settings to PCAN-RS232

PPA Datalogger disconnects from the **CAN-port** unit. At this point, the **CAN-port** unit should be turned off, and the serial cable removed from the PC. Attach the serial cable to the PPA and ensure the PPA is turned on and its interface is set to RS232 (and with the correct baud rate) in the REMOTE menu.
From then onwards, each time you turn the **CAN-port** unit on, it will then set up the PPA as per the settings from PPA Datalogger.



Setup completed successfully; Disconnected from serial connection

Power down the PCAN-RS232, connect the serial connection to the PPA that the PCAN RS232 will control and then power on the PPA.
Ensure the PPA is set up to recieve a serial connection using your set Baud Rate in the REMOTE menu.

Every time the PCAN-RS232 is powered on it will set up the PPA.

Power cycle the PCAN-RS232 to set up the PPA

Communications Usage

All commands can be sent over SERIAL or CAN connection, all responses from the PCAN RS232 will be sent back along the connection you sent the command from, unless otherwise noted. Commands sent via SERIAL must end in either a Semi Colon "**;",** a Newline character or a Line Feed Character. Everything received over SERIAL that isn't a command is stored until terminated by a Semi Colon, Newline or Linefeed character, and then it is processed and sent over CAN. There is a 1 second timeout on all serial commands, if the command is not terminated before the timeout, it is discarded.
The **CAN-port** will only respond to messages where the message's CAN ID matches the set read ID. CAN commands that require more than 1 CAN Message, or CAN commands to be sent to the PPA will be stored until terminated by either a Semi Colon, Newline or Linefeed character, then it is processed. There is a 1 second timeout for multiline CAN commands; if the command is not terminated before the timeout it is discarded.

Sending Commands to the CAN-port unit

Commands can be sent to the **CAN-port** unit either by sending a CAN message with the CAN Message's ID set to the ID that the **CAN-port** unit is set to read over the CAN network, or by connecting to the **CAN-port** unit with a serial (RS232) connection and using Newtons4th's CommView2 program.

Direct Commands

Commands can be sent to the PPA via the **CAN-port** unit by sending the full command for the instrument followed by a semi-colon, line feed or newline character (eg. "SPEED,WINDOW,0.002;"). To see a full list of commands that can be sent to the PPA please refer to your PPA's Comms Manual.

Requesting Multilog Data

To request Multilog data from the PPA via the **CAN-port**, simply send a direct command to the PPA using the "MULTIL?;", "MULTI#?;" or "MULTIL,*lines*?;" query over CAN, as described in your PPA's Comms Manual.

The "MULTI#?;" command will be sent for you automatically if you have set the **CAN-port** to repeatedly request data from the PPA, and on power up if POWML is set (either via command or by using PPA Datalogger's "Request Multilog Data when PCAN powers on?" option)

Loading new firmware into the CAN-port

To load new firmware into the **CAN-port** unit, ensure it is powered off. Connect the CAN terminal to a PC using a CAN converter. To place the **CAN-port** unit into Boot Mode, hold the boot switch on and then power on the unit. The light on the **CAN-port** unit should be flashing orange, indicating it has successfully entered boot mode.
Once in boot mode, load up PCAN-Flash and navigate to Application->Option. Set the Hardware Profile to PCAN-RS-232, and then select the Filename's browse button, and navigate to the firmware's .bin file, and select it. Leave all other settings as default and press "OK".
Next navigate to PCAN->Connect and select the **CAN-port** device which should appear in a list in the connect window and press OK.
Finally, Navigate to Module->Detect to have PCAN-Flash detect the **CAN-port's** firmware, select Module 15 which will appear in the main window of PCAN-Flash and then select Module->Programme.
The status bar at the bottom shows the progress of the firmware install and will announce when the firmware upload has finished. Close the program and power cycle the **CAN-port** unit and it will now use the new firmware

Manual Setup of the CAN-port

The **CAN-port** unit can be set up manually by sending commands to the **CAN-port** unit via a serial connection. Ensure the **CAN-port** unit is powered on; the status light should be blinking green. Once the **CAN-port** unit is on, connect an RS232 cable to the serial port and connect it to your PC. Open Newton4th CommView2 program and connect to the COM Port that is connected to the **CAN-port** unit.
Once connected on CommView2 commands can be sent to the **CAN-port** unit by typing them into the command bar at the bottom of the program and pressing the Enter key. Any responses from the **CAN-port** unit will be displayed in CommView2.

Using PPA Datalogger to set up the CAN-port

The easiest way to set up the **CAN-port** unit is to use Newtons4th's PPA Datalogger program to choose which settings the **CAN-port** unit should be set up with when it powers on, and allow PPA Datalogger to send all the commands to set up the unit

This is a simple one-time setup that can be performed once, and the **CAN-port** will remember the settings each time it powers on and will attempt to set up the connected PPA.

Command List

A list of all commands the **CAN-port** uses
BAUD
BAUD?
BITR
BITR?
ID
ID?
IDN?/*IDN?
MLCOUNT
MLCOUNT?
MLNOOLD
MLNOOLD?
MLREP
MLREP?
MLREPLY
MLREPLY?
MLSTART
MLSTOP
MS
MS?
MULTILOG
POWLOAD
POWLOAD?
POWML
POWML?
POWSTAT
POWSTAT?
POWSET
POWSET?
PROG
PROG?
REPLY
REPLY?
STATUS?

BAUD

Description:

Sets the baudrate of the serial output

Parameters:

Single integer value, representing baudrate

Values:

0 = 1200
1 = 2400
2 = 4800
3 = 9600
4 = 19200
5 = 38400
6 = 57600
7 = 115200
Example: "BAUD3"
Sets the baudrate to 9600

BAUD?

Description:

Replies with an integer value representing the baudrate

Parameters:	None

Values:

0 =	1200
1 =	2400
2 =	4800
3 =	9600
4 =	19200
5 =	38400
6 =	57600
7 =	115200

Example:

```
"BAUD?"
> "3"
```

Replies with current baudrate (9600)

BITR

Description:

Sets the CAN Bitrate

Parameters:

Single hex value, representing bitrate

Values:

0	=	10000
1	=	20000
2	=	33300
3	=	47600
4	=	50000
5	=	83300
6	=	95200
7	=	100000
8	=	125000
9	=	200000
A	=	250000
B	=	500000
C	=	800000
D	=	1000000

Example:

"BITRC"
Sets the bitrate to 800000

BITR?

Description:

Replies with a hex value representing the CAN Bitrate

Parameters: None

Values:

0 = 10000
1 = 20000
2 = 33300
3 = 47600
4 = 50000
5 = 83300
6 = 95200
7 = 100000
8 = 125000
9 = 200000
A = 250000
B = 500000
C = 800000
D = 1000000

Example:

```
"BITR?"
 > "C"
```

Replies with current bitrate (800000)

ID

Description:

Sets the CAN ID the **CAN-port** responds to

Parameters:

ID In Hex (3CH)

Values:

Between "001" and "7FF"

Example:

"ID07A"
Sets CAN ID to (07A [122])

ID?

Description:

Replies with the CAN ID that the **CAN-port** responds to

Parameters:

None

Values:

Between "001" and "7FF"

Example:

```
"ID?"
 > "07Ah"
```

Replies with the CAN ID in hex

IDN? / *IDN?

Description:

Returns a standard format identification string

Parameters:

None

Values:

None

Example:

```
"IDN?"
> NEWTONS4TH,CAN-port,01234,1.00
 (manufacturer, model, serial no, version)
```

MLCOUNT

Description:

If set ON, the **CAN-port** sends a counter as the first message in a multilog reply. The counter increments each time new data is sent over CAN. The counter will reset at a count of 65535

Parameters:

A single integer value, representing on/off

Values:

0 = Off
1 = On

Example:

"MLCOUNT1"
Sets the **CAN-port** to send the counter as the first message in a multilog response.

MLCOUNT?

Description:

Replies with the status of MLCOUNT that the **CAN-port** is set to.

Parameters:

None

Values:

0 = Off
1 = On

Example:

"MLCOUNT?"
> "1"
Indicates that the **CAN-port** will send the counter as the first message in a multilog response.

MLNOOLD

Description:

Set to signify if the **CAN-port** should send the last multilog response if no new data has been received

Parameters:

A single integer value, representing whether repeat data can be sent

Values:

0 = Send repeat data.
1 = Only send new data.

Example:

"MLNOOLD1"
Sets the **CAN-port** to only send new data when requesting multilog data.

MLNOOLD?

Description:

Replies with the status of the **CAN-port** which signifies if it replies with repeat data if no new data is available from the PPA or only sends new data.

Parameters:

None

Values:

0 = Send repeat data.
1 = Only send new data.

Example:

```
"MLNOOLD?"
 > "1"
```

Indicates that the **CAN-port** will not send repeat data.

MLREP

Description:

Sets the **CAN-port** to repeatedly request multilog data from the PPA

Parameters:

A single integer value, representing how often the **CAN-port** should re-request data from the PPA

Values:

0 = Don’t Repeat
1 = Repeat
2 = Repeat On Command

Example:

"MLREP1"
Sets the **CAN-port** to repeatedly request multilog data by sending “MULTI#?” to the PPA over the serial link.

MLREP?

Description:

Replies with the status of the **CAN-port,** signifying if it is repeatedly requesting multilog data

Parameters:

None

Values:

0 = Don’t Repeat
1 = Repeat
2 = Repeat On Command

Example:

```
"MLREP?"
 > "1"
```

Indicates that the **CAN-port** is repeatedly sending “MULTIL#?” commands to the PPA over serial.

MLREPLY

Description:

Sets how the **CAN-port** should send multilog responses

Parameters:

A single integer value, representing how the multilog response should be sent

Values:

0 = Static – All responses sent to the reply ID
1 = Incremental – Starting at the reply ID, each message in a single response is sent to incremental IDs
2 = Custom – Responses are sent to custom IDs

Example:

"MLREPLY1"
Sets the **CAN-port** to send each message in a multilog response to incremental IDs starting from the reply ID.

MLREPLY?

Description:

Replies with the status of how the **CAN-port** will parse multilog responses into CAN messages

Parameters:

None

Values:

0 = Static – All responses sent to the reply ID
1 = Incremental – Starting at the reply ID, each message in a single response is sent to incremental IDs
2 = Custom – Responses are sent to custom IDs

Example:

```
"MLREPLY?"
 > "1"
```

Indicates that the **CAN-port** will send each message in a multilog response to incremental IDs starting from the reply ID.

MLSTART

Description:

Starts the **CAN-port** to repeatedly request multilog data when MLREP is set to Repeat On Command (2)

Parameters:

None

Values:

None

Example:

“MLREP2”
"MLSTART"
…
“MLSTOP”

MLSTOP

Description:

Stops the **CAN-port** from repeatedly request multilog data when MLREP is set to Repeat On Command (2)

Parameters:

None

Values:

None

Example:

“MLREP2”
"MLSTART"
…
“MLSTOP”

MS

Description:

Sets the speed (in seconds) that the **CAN-port** repeatedly asks the PPA for multilog results, if MLREP is set to a value of 1 or 2

Parameters:

A single floating point value, representing the speed (in seconds) the **CAN-port** requests multilog results

Values:

Between 0.025 (2.5E-2) and 100.0 (1.0E3) seconds

Example:

"MS7.5E-1"
Sets the **CAN-port** to request multilog data every 750 milliseconds.

MS?

Description:

Replies with how frequently the **CAN-port** will parse multilog responses into CAN messages

Parameters:

None

Values:

Between 0.025 (2.5E-2) and 100.0 (1.0E3) seconds

Example:

```
"MS?"
 > "0.75"
```

Replies with the speed the **CAN-port** will be requesting multilog results.

MULTILOG

Description:

Sets the **CAN-port**’s internal multilog settings and sends the appropriate setup command to the PPA over serial.

Parameters:

MULTILOG,*Index*,*Phase*,*Func*
or
MULTILOG,*Index*,*Phase*,*Func,CustomID*
or
MULTILOG,*Index*,*Phase*,*Func,Form,Scale,Off*
or
MULTILOG,*Index*,*Phase*,*Func,Form,CustomID,*
Scale,Off
or
MULTILOG,*Index*,*Phase*,*Func,Form,Start,Len,Scale*
,Off
or
MULTILOG,*Index*,*Phase*,*Func,Form,CustomID,*
Start,Len,Scale,Off

Index is the Multilog index as an integer (see PPA Comms Manual)
Phase is the Multilog phase as an integer (see PPA Comms Manual)
Func is the Multilog function as an integer (see PPA Comms Manual)
CustomID is the CAN ID this value is sent to if sending to custom IDs
Form is the format the reply takes as an integer:
0 = ascii
1 = unsigned integer (as hex)
2 = signed integer (as hex)
3 = IEEE Floating Point (as hex)
4 = IEEE Double precision floating point (as hex)
Start is the start byte for this data in the message as an integer
Len is the length of this value in the message in bytes as an integer
Scale is a floating point value that you will scale this value by
Off is a floating point value that you will offset this value by

Values:

Index: between 1 and 64 (See PPA Comms Manual)
Phase between 1 and 11 (See PPA Comms Manual)
Function between 0 and 255 (See PPA Comms Manual)
CustomID between 001 and 7FF
Form between 0 and 4
Start between 0 and 7
Len between 1 and 8
Scale between 1.0E-6 and 1.0E6
Offset between 1.0E-6 and 1.0E6

Example:

"MULTILOG,1,3,51,1,01F,0,8,100.0,-5.0E3;”
Sets a multilog in index 1, for Current RMS on phase 3, it will be sent to ID 01F if MLREPLY is set to custom, it will be placed in location 0 in the message and will be 8 bytes long. The value will be divided by 100x and have 500 added to it.

POWLOAD

Description:

Set to decide if the **CAN-port** loads the last settings it used on power up

Parameters:

Single integer value, representing on/off

Values:

0 = Off
1 = On

Example:

"POWLOAD1"
Sets the **CAN-port** to load settings on power up

POWLOAD?

Description:

Replies with a value that signifies if the **CAN-port** will load settings on power up

Parameters:

None

Values:

0 = Off
1 = On

Example:

"POWLOAD?"
> "1"
Replies with current POWLOAD setting (On)

POWML

Description:

Sets to decide if the **CAN-port** sends multilog commands for a PPA when it next powers up

Parameters:

Single integer value, representing if the multilog commands should be sent

Values:

0 = Off
1 = On

Example:

"POWML1"
Sets POWSET to on, indicating that when the **CAN-port** next powers on it will send multilog commands over serial

POWML?

Description:

Replies with a value that signifies if the **CAN-port** will send multilog commands over serial next time it powers up

Parameters:

None

Values:

0 = Off
1 = On

Example:

```
"POWML?"
 > "1"
```

Indicates that the **CAN-port** will send multilog commands over serial next time it powers up

POWSTAT

Description:

Set to decide if the STATUS? Command should be performed on power up

Parameters:

Single integer value, representing if the STATUS? should be sent

Values:

0	= Off
1	= On

Example:

"POWSTAT1"
Sets POWSTAT to on, indicating that when the **CAN-port** next powers on the STATUS? Command will be performed, sending the STATUS over CAN

POWSTAT?

Description:

Replies with a value that designates if the STATUS? Command will be performed on power up

Parameters:

None

Values:

0 = Off
1 = On

Example:

```
"POWSTAT?"
 > "1"
```

Indicates that the STATUS? Command will be performed next time the **CAN-port** powers up

POWSET

Description:

Sets to signify if the **CAN-port** sends setup commands for a PPA when it next powers up

Parameters:

Single integer value, representing if the setup commands should be sent

Values:

0 = Off
1 = On

Example:

"POWSET1"
Sets POWSET to on, indicating that when the **CAN-port** next powers on it will send setup commands over serial

POWSET?

Description:

Replies with a value that signifies if the **CAN-port** will send setup commands over serial next time it powers up

Parameters:

None

Values:

0 = Off
1 = On

Example:

"POWSET?"
 > "1"
Indicates that the **CAN-port** will send setup commands over serial next time it powers up

PROG

Description:

Sets which program from the PPA, if any, the **CAN-port** should load the next time the **CAN-port** powers on

Parameters:

The program to load, in hex (3 chars).

Values:

0 = Don’t Load a program
1 = NOT USED
2+ = Load the program of the given value

Example:

"PROG000"
Tells the **CAN-port** to not load a program when it next powers on

“PROG032”
Tells the **CAN-port** to load program (032 [50]) when it next powers on.

PROG?

Description:

Replies with which program on the PA, if any, the **CAN-port** will load in when its next powered on

Parameters:

None

Values:

0 = Off
2+ = The program that will be loaded in, in hex.

Example:

"PROG?"
> "032"
Indicates that the **CAN-port** will load the program (032 [50]) into the PPA the next time it powers on

REPLY

Description:

Sets the CAN ID to reply to

Parameters:

ID in hex (3CHAR)

Values:

Between "001" and "7FF"

Example:

"REPLY1AB"
Sets the CAN ID to which the **CAN-port** sends replies to 1AB (427)

REPLY?

Description:

Replies with the CAN ID the **CAN-port** has been set to reply to

Parameters:

None

Values:

Between "001" and "7FF"

Example:

"REPLY?"
> "1AB"
Replies with CAN ID to which the **CAN-port** sends replies (1AB [427])

STATUS?

Description:

Replies with the current status of the **CAN-port**
SENT VIA CAN

Parameters:

None

Values:

4 ascii chars, representing setups were performed OK

CHAR 1: PowerOn OK
CHAR 2: LoadFromMemory OK
CHAR 3: SerialSetup OK
CHAR 4: CANSetup OK

Example:

"STATUS?"
> "1011"
Indicates that the **CAN-port** was able to perform Power On setup, Serial setup and CAN Setup OK, but didn’t Load From Memory.