

MPU series AN001 rev AA

Application Note – Interfacing with CAN bus

Table of Contents

1 Int	troduction	2
2 Re	eferences and Required Tools	3
2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9	Documents CAN interface Communication development environment GUI environment Monitoring software Network configuration and CAN wiring CAN wiring CAN node ID Messages description	
3 Co	ommunication tests	5
3.1 3.2 3.3	Communication using Python script CAN messages monitoring Critical fault world interpretation	5 9 13
4 Er	ndianness	14



Product application note. The given values are susceptible to change without prior notice.



Document Reference: MPU series AN001 rev AA

1 Introduction

This document describes CANopen protocol structure and configuration for MPU series chargers. The main purpose is to explain how to build necessary CAN messages to control Modular Power Unit without employing any specific CANopen stack.

MPU series chargers are 25 kW power units where the parallelization of several chargers allows to modulate the delivered power. Hence, CANopen protocol is suitable for fast communication with several power units to be controlled through a CANopen master.

MPU series are dotted with a graphical user interface that can play either the role of a master or a CAN bus sniffer. Both functionalities could be used to debug communication issues if another master than Watt&Well GUI is used. In this document, test examples are given for MPU 25 kW unidirectional charger where the master is run from Python script.

Common terms

MPU G2V GUI CAN CANopen	Modular Power Unit Grid-to-Vehicle Graphical User Interface Control Area Network Communication protocol to open and communicate with the Control Area Network
EDS	Electronic Data Sheet
Index	4-digit hexadecimal code used to identify an object: 16-bits
Sub-index	Decimal code to further identify object's parameters: 8-bits
OD	Object Dictionary
Object	Communication message
PDO	Process Data Object
RPDO	Receive PDO
TPDO	Transmit PDO
SDO	Service Data Object
NMT	Network Management



2 References and Required Tools

2.1 Documents

Reference	Document Title
MPU-R3-500-63-FD User Guide	Modular Power Unit - User guide
MPU-R3-500-63-FD Datasheet	MPU-25 Datasheet

2.2 CAN interface

For PC/CAN interface, it is recommended to use one of the listed below transceivers:

- National instruments: NI USB-8473 or NI USB-9861
- IXXAT: USB-to-CAN V2 compact
- **Kvaser:** Leaf Light V2

2.3 Communication development environment

In this document Python 2.7 is used to set up communication examples. It can be downloaded from : <u>https://www.python.org/downloads/release/python-2717/</u>.

The library *canopen 1.0.0* is required. Installation steps are listed below:

- Install pip following instructions from: <u>https://pip.pypa.io/en/stable/installing/</u>
- Install *canopen* library by running the following in command-line tool
 \$ pip install canopen

2.4 GUI environment

The GUI is compatible with National instruments interface and Windows 10/7/Vista/XP/200. NI-CAN drivers must be installed. They can be downloaded from : <u>http://www.ni.com/download/ni-can-18.5/8074/en/</u>

User is referred to GUI user guide documents *Modular Power Unit - User guide* for further information.

2.5 Monitoring software

To spy the CAN bus for test or debug purposes, it is recommended to install the following software:

- **National instruments:** NI MAX (Measurement and Automation Explorer) is included in NI-drivers package.
- **IXXAT:** canAnalyser (<u>https://www.ixxat.com/support/file-and-documents-download/demo-software-tools</u>)



2.6 Network configuration and CAN wiring

Physical CAN network must be equipped with 2 termination resistors of 120 Ω each. The smallest CAN network is composed of 2 nodes; the GUI node (master) and the BMPU node:



Figure 1 Network nodes

2.7 CAN wiring

For CAN wiring, refer to MPU datasheet *MPU-25 Specification* on Interfaces section.

2.8 CAN node ID

For all devices a unique node ID must be selected. MPU-25 takes its CAN **address** at boot based on an addressing connector on the front panel.

Address	CAN ID
1 (001)	80
2 (010)	81
3 (011)	82
4 (100)	83
5 (101)	84
6 (110)	85
Default (000 or 111)	86

Values 0 and 1 on address refer respectively to 0V and 24V on the corresponding connector pin.

If no connector is connected to the charger, the default node ID is 86 (0x56).

Message frame IDs are defined in as: frame ID = offset ID + node ID

Example: TPDO0 offset is **0x180** and node ID is **0x56** then TPDO0 ID will be **0x1D6** =0x180+0x56

2.9 Messages description

For messages description, user is referred to MPU25 specification CAN communication section. In the same section, status word, fault word and data definition are presented. Taking note of CAN message description is required to understand the following section.

The user may also refer to CANopen literature for detailed description of CANopen specific messages as PDO (Processes Data Objects), SDO (Service Data Objects) ... etc.



3 Communication tests

The master node ID is 1 and charger node ID is 80 (address set to 001).

3.1 Communication using Python script

Follow the instructions step by step to set up a correct CAN communication using Python script

Step 1	Set the testbench :
Test setup	
	Supply 24V to MPU-25
	Connect Emergency shutdown (24V)
	• Connect the addressing connector in such way to obtain the address 80 (001)
	Connect CAN transceiver to MPU-25 CAN RJ45 connector and to a PC
	 Connection to AC power is not required and not recommended for
	communication tests. However, without AC power the charger will go to fault
	state when Charging mode is requested.
Step 2	Start new Python script :
Network	
configuration	• MPU-25 electronic datasheet (pu.eds) must be in the same folder as Python
g	script. Otherwise, file path must be specified in the script when EDS is needed.
	Import required libraries
	import canopen
	import sys
	import traceback
	import time
	Configure network and connect
	try:
	# Start with creating a network representing one CAN bus
	network = canopen.Network()
	# Specifiv CAN tranceiver type, CAN channel and haud rate
	network.connect(bustype='ixxat', channel=0, bitrate=500000)
	# network.connect(bustype='nican', channel='CAN0', bitrate=500000)
	# Specifiy node address and the corresponding Electronic Datasheet
	network.add_node(80, ./pu.eds,upioad_eds=Faise) node – network[80]
	# Check network
	network.check()
	 Send Master Heartbeat message (ID 701) for bootup (MasterStatus=0)
	# Master boot up message (MasterStatus =0)
	network.send_message(0x701, [0x0])
	Read PDO configuration from node
	# Read PDO configuration from node
	node.tpdo.read()
	noue.ipuo.iedu()
	Change master state de operational (MasterStatus=5)
	# Change master state to operational (NM I start)



Document Reference: MPU series AN001 rev AA

	network.send_message(0x701, [0x5])						
Step 3 Sync message	Check slave heartbeat # Check slave heartbeat node.nmt.wait_for_heartbeat() assert node.nmt.state == 'OPERATIONAL' Send sync message with a selected period # Transmit SYNC every 100 ms network.sync.start(0,1)						
Step 4 Sending RPDOs	Configure RPDO0 message data to be sent. # RPDO[1] message definition node.rpdo[1]['setPoints.state_Request'].raw = 5; node.rpdo[1]['setPoints.dcdc_currentOutSP'].raw = 6300; node.rpdo[1]['setPoints.dcdc_voltageOutSP'].raw = 4000; node.rpdo[1]['setPoints.pfc_iGridMaxSP'].raw = 4500;						
	 Be careful with setpoint unities (refer to 3.2.8). In the following example, dcdc_currentOutSP (DC output current) is set to 6300 which corresponds to 6300 of 10 mA → 6300 x 0.01A=63 A, then the requested value is 63 A. state_Request is set to 5 for start charging (refer to 3.2.7). Start sending RPDO0 periodically (every 0.1 s in this example) 						
	node.rpdo[1].start(0.1) print 'RPDO1 is transmitted', '\n'						
Step 5 Reading TPDOs	 Reading TPDO0s. The following example is given for TPDO0 and the apporach can be reiterated for other TPDOs. 						
	<pre># Read values from TPDO[1] node.tpdo[1].wait_for_reception() print 'Receiving TPDO1', '\n' StatusWord = node.tpdo[1]['measurements.state_Current'].raw print 'Status Word :', StatusWord FaultWord = node.tpdo[1]['measurements.faultWord'].raw print 'Fault Word :', FaultWord, '\n' • TPDOs are configured to be transmitted after every nth sync message. The numberof sync message to be received before transmission of each TPDO is defined by the Transmission Type parameter.</pre>						
	Transmission type for each TPDO are defined in Table 1.						



Step 6 Emergency	Reading emergency codes and critical fault world					
codes reading	# Read emergency codes print 'Emergency codes reading', '\n'					
	error_code = [emcy.code for emcy in node.emcy.active] print 'Error code :', error_code					
	error_register = [emcy.register for emcy in node.emcy.active] print 'Error register :', error_register					
	error_data = [emcy.data for emcy in node.emcy.active] print 'Critical Fault Word :', error_data, '\n'					
Step 7 SDO reading	 To read any object in the OD (parameter, signal, measurementetc), an expedited SDO is used. The target object can be either stated by its variable name or index. 					
	 If the object is a part of a category, its variable and category names must be stated or its index and sub-index (see below <i>pfc_VintMin</i> example). 					
	<pre># Read variables using SDO software_version = node.sdo['Manufacturer software version'].raw print 'Software version:', software_version, '\n'</pre>					
	VintMinRef =node.sdo['commandSaturation']['pfc_VintMin'].raw print 'DC bus voltage min reference:', VintMinRef, '\n'					
Step 8 Testing	 Run Python script Execution result is shown below 					



Document Reference: MPU series AN001 rev AA





Document Reference: MPU series AN001 rev AA

3.2 CAN messages monitoring

In order to check Python script test functioning, MPU-25 GUI could be used in spy mode. The connection of a second CAN (NI USB-8473) transceiver is necessary to communicate with the GUI.



Figure 2 Transceiver configuration for CAN monitoring

Set-point autosend period and Slow Meas period must be set to 0 ms to cease transmission from GUI to charger (to be in spy mode) and slave address must be set to 80.

Settings	?	×
Communication configuration		
CAN Port :		
CANO		•
Baud Rate:		
500 kBit/s		•
Slave CAN address		
80		•
Monitor CAN address		Ŧ
Set-point autosend period	0	ms
Slow Meas Period	0	🔹 ms
✓ Update parameters at the beginning of the	communic	ation
	Ap	oply
	Ap	oply

Figure 3 GUI settings

After Python script execution:



- Blue led of Charging mode must be lighted up on the charger and on the GUI.
- Green LEDs of PWM On DC/DC and PWM On PFC must be lighted up.
- The software version must be the same returned by Python script.

IPU Monitor							- 0
m Interface Mode Module Help							
🙀 🚳 🔟 Қ 🟮 Auto Tests							
in Advanced Accept							
ode information	Status		_				
Watt&Well QCNG-POWRx08 NodeID : 80	StandBy Stopping	Passive Pre	charge Active Precharge Char	ging P	PWM On DC/DC PWM On PFC	Thermal limit Load Z limit Power limit Safe C	TimeOut Fault
SW rev : 2.20r Build Nb : 15799	Measurements						
FPGA rev : 0.3	Live values				Current and Voltage (Output	t) V Current and Voltage (Grid) V Current vs Voltage	e (Output)
-	Grid:		Temperature:			Output Current & Output Voltage	
Firmware validity : 🤤 Refresh	Grid Voltage RMS : 1.	1 V	Temp PEC 1 -	0.0 90	E 08		E 600
utput DC command			interproce.	0.0 C	75		- 550
StandBy Charge	Grid Current RMS : 0.	/ A	Temp PFC IMS :	19.2 °C	70		
Pre-Charge	Grid Power : 1.	0 W	Temp DCDC2 L :	18.8 °C	65		- 500
	Output:		Temp Extra :	0.0 °C	60		450
CDC Available Out Current : 20.0 A	Output Voltage : 0.0	v	Temp DCDC1 L +	10.1 90	55		- 400
C Current [A] 20.0 @			ionp beder e r		50		
2.00 🗣	Output Current : 0.0	A	Temp DCDC2 XF0 :	18.7 °C	≤] 45 T		- 350 년
100.00 300.0	Output Power : 0.0	w	Temp DCDC1 XFO :	18.7 °C	40 1		300 5
And Input Grid Current [A]			Temp Ambient :	28.2 °C	10 35 T		age [
45.00 \$ 45.0	Intermediate:				30		230 5
Output limited by system gain :	PFC +400V : 3.1	v	DC/DC		25		- 200
igh level log	PEC 400V 1 2.8		Switching Frequency :	115.0 KHz	20		- 150
	FFG -100V : 2.8	v	Phase :	180 deg	15		
	DCDC +400V : 2.8	v	Ctrl PI Out (beta) :	1.0	10		- 100
2	DCDC -400V : 2.2	v	Vint Ref :	750.0 V	5		50
					E o		E o
						- Output Current [A] - Output Voltage [V]	

Figure 4 MPU-25 GUI in spy mode

• To check that RPDO0 message has been correctly received, it is possible to verify the received values of RPDO0 objects in the CANopen Device Manger (see Figure 5).



Document Reference: MPU series AN001 rev AA

谢 CANopen Device Manager	-	_		×	
<u>F</u> ile					
Name				^	
0x23 dcdc_vOutRampDown			200.0	000	
0x24 safeC_RestartDelay			0.03	200	
0x25 vintChargedMarginVsInpu	t		1.1	500	
✓ 0x4600 setPoints					
Ox00 max sub-index				5	
0x01 state_Request				5	
0x02 dcdc_currentOutSP		6300			
0x03 dcdc_voltageOutSP		4000			
0x04 dcdc_powerOutSP			25	000	
0x05 pfc_iGridMaxSP			4	500	
Device Profile Area				~	
Index Sub-Index Type Mem	ory	Туре	Data		
	4	* *			
Store parameter Download log Read					
Restore factory settings Erase log		Updat	e all value	es 🔵	
				100%	
	_				

Figure 5 CANopen Device Manager

- Another way to spy on CAN messages is the utilization of NI Measurement and Automation Explorer (NI MAX). This software is included in NI-drivers package and installed automatically with the drivers. Configuration steps to monitor the CAN bus are as follows:
 - Start NI MAX
 - Go to Devices and Interfaces → NI CAN → USB-xxxx→ CAN0 (or select the corresponding interface)

CAN0 - Measurement & Automation Explo	rer			_		×
File Edit View Tools Help						
V 📮 My System	🛅 Bus Monitor	Properties		_	💦 Hide	Help
 Data Neighborhood Devices and Interfaces 	Attribute	Value	Description	Back		. ~
√ ■ NI-CAN	Interface Name	CAN0	Interface name assigned by the user	NI-CA		+ ^
✓ ■ USB-8473	Location	Port 1 High-Speed	Interface location on the board			-
	Transceiver Nar	ne Philips TJA1	Transceiver brand name	What d	lo you	
> 💭 Software	Baud Rate	125.000 kB	Baud rate selected by the user	want to	o do?	
> 😫 Remote Systems	Sample Point	87.5%	Sample point within the bit transmission			
	BIR0 BIR1	0x03 0x1C	Bit Timing Register 0 value Bit Timing Register 1 value	♦ <u>Run</u> Monitor	the Bus	
				+ Cha	ange Port	~
< >>	Port Properties			Propert	ies	

• In Properties, set the baud rate to 500.



Document Reference: MPU series AN001 rev AA

Port Properties		×
Settings		
Interface	CAN0 ~	
Baud Rate	500.0 ~	kBaud
	Advanced >>	
ОК	Cancel	Help

 Click on Bus Monitor to start CAN bus monitoring. All frames generated by Python script must be visible.

MI-CA	N BusMo	onitor	500.000 l/Dl		Upd	ate Rate —		Busload	-	
Interface		Baud Rate :	SUU.UUU KDaud		100	u v ms		0 %		
Arb.ID	Len	Data	Time St F	Rate	dt Min	dt Max	# (tot	al)		Help
0×80	0		2614.18 1	0.00	0.100	0.100	23			
0xD0	8	01 FF 10 00 00 00 00 10	1.7691e		-	-	1			Options
0x1D0	6	07 00 00 00 00 10	2613.98 1	.43	0.700	0.700	3			
0x250	7	05 CC 10 B8 0B 94 11	2614.01 1	0.00	0.100	0.100	20			Bus Statistics
0x2D0	8	01 00 12 00 00 00 CF 07	2613.88 2	2.00	0.500	0.500	4			Error Doteilo
0x3D0	8	00 00 02 00 00 00 94 11	2614.18 1	0.00	9.975e	0.100	23			Lifor Details
0x4D0	8	00 00 7C 15 D0 07 A8 61	2613.68 0).91	1.100	1.100	2			
0x5D0	8	60 00 45 11 00 00 00 00	2612.11 3	319.49	1.264e	0.681	60			
0×650	8	23 00 45 11 00 00 20 C1	2612.11 3	342.11	1.665e	0.681	60			
0×701	1	05	2611.10 8	3474	1.180e	1.180e	2			
0x750	1	05	2627.41 1	.23	3.640e	0.839	3245			
										Listen Only
										Stop
										Decet
										Heset
										Close

Figure 6 NI-CAN BusMonitor

If Sync message is sent every 0.1s, then TPDO1 with transmission type of 5 will be transmitted every 0.5 s (the value is comprised between *dt Min* and *dt Max*) which gives a rate of 2 (the message is transmitted twice per second).



3.3 Critical fault world interpretation

When critical fault occurs, Emergency message will return the Critical Fault Word as 5 bytes data (byte 3 to byte 7, keep in mind that byte 3 is not used).

To interpret the received data and determine which fault has occurred, bits must be compared to the fault word defined in 3.2.7.

The Critical Fault Word in the example below is decoded to illustrate how critical faults are determined. To create fault condition, AC power is disconnected

```
Emergency codes reading...
```

```
Error code : [65281]
Error register : [16]
Critical Fault Word : ['\x00\xc0\x01\x00\x00']
```

	Byte 7							Byte 6						Byte 5							Byte 4											
	CO							01						00							00											
Bits	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	23	22	21	20	19	18	17	16	31	30	29	28	27	26	25	24
	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Order of bytes and bits is due to CANopen endianness. Refer to section 4 for more details.

Then, occurred faults are determined by bits 6, 7 and 8 which corresponds to faults UV_PhaseAVoltageRMS, UV_PhaseBVoltageRMS and UV_PhaseCVoltageRMS. It corresponds to the created fault where there is no voltage at charger AC input.



4 Endianness

All numerical data types consisting of multiple bytes are transferred in CANopen (whether in SDO or PDO) in the Little-Endian format. Bytes are ordered by significance and the lower significant bytes come first. It means that last byte of binary representation of the multibyte datatype is stored first.

For example, the 32-bit hexadecimal number « 0xCDE11C0A» will be transmitted in CAN bus as follows

		0.	A	1C	E1	CD]							
					/									
NI-CAN	NI-CAN BusMonitor - I													
MAX Port Settings Update Rate Busload														
Interface :	CAN	0 Baud Rate : 5	500.000 kBaud		1000 ~ ms	0%								
Arb.ID	Len	Data	Time St	Rate	dt Min	dt Max	# (total)	Help						
0x80	0		14.212	10.00	0.100	0.100	23							
0x1D0	6	05 18 00 00 00 40	13 8724	1.43	0.700	0.700	3	Options						
0x250	7	05 CC 10 B8 0B 94 11	14.1601	10.00	0.100	0.100	21	Due Statistice						
0x2D0	8	09 00 44 00 01 00 CF 07	14.2724	2.00	0.500	0.500	5	Dus Statistics						
0x3D0	8	00 00 04 00 00 00 94 11	14.2727	9.98	9.975e-002	0.100	23	Error Details						
0x4D0	8	00 00 7C 15 D0 07 A8 61	13.8728	0.91	1.100	1.100	2	Endi Detalis						
0x5D0	8	60 00 45 11 00 00 00 00	12.1548	400.00	1.857e-003	5.312e-002	59							
0×650	8	23 00 45 10 0A 1C 1E CD	12.1539	408.66	1.696e-003	5.320e-002	59							
0x701	1	05	11.7518	8474	1.180e-004	1.180e-004	2							
0x750	1	05	17.9482	11.51	1.160e-004	0.138	217							
								Listen Only						
								Start						
								Reset						
								Close						