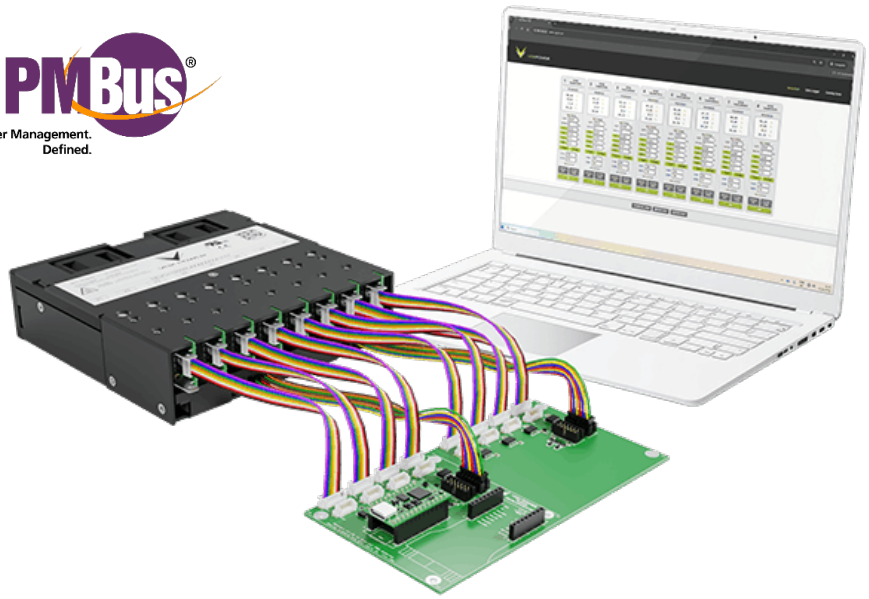
NEVO+ Series

USER MANUAL

Digital Development Kit (DevKit)



SCPI
Web Control
PMBus
and more ...

Digital control made simple.

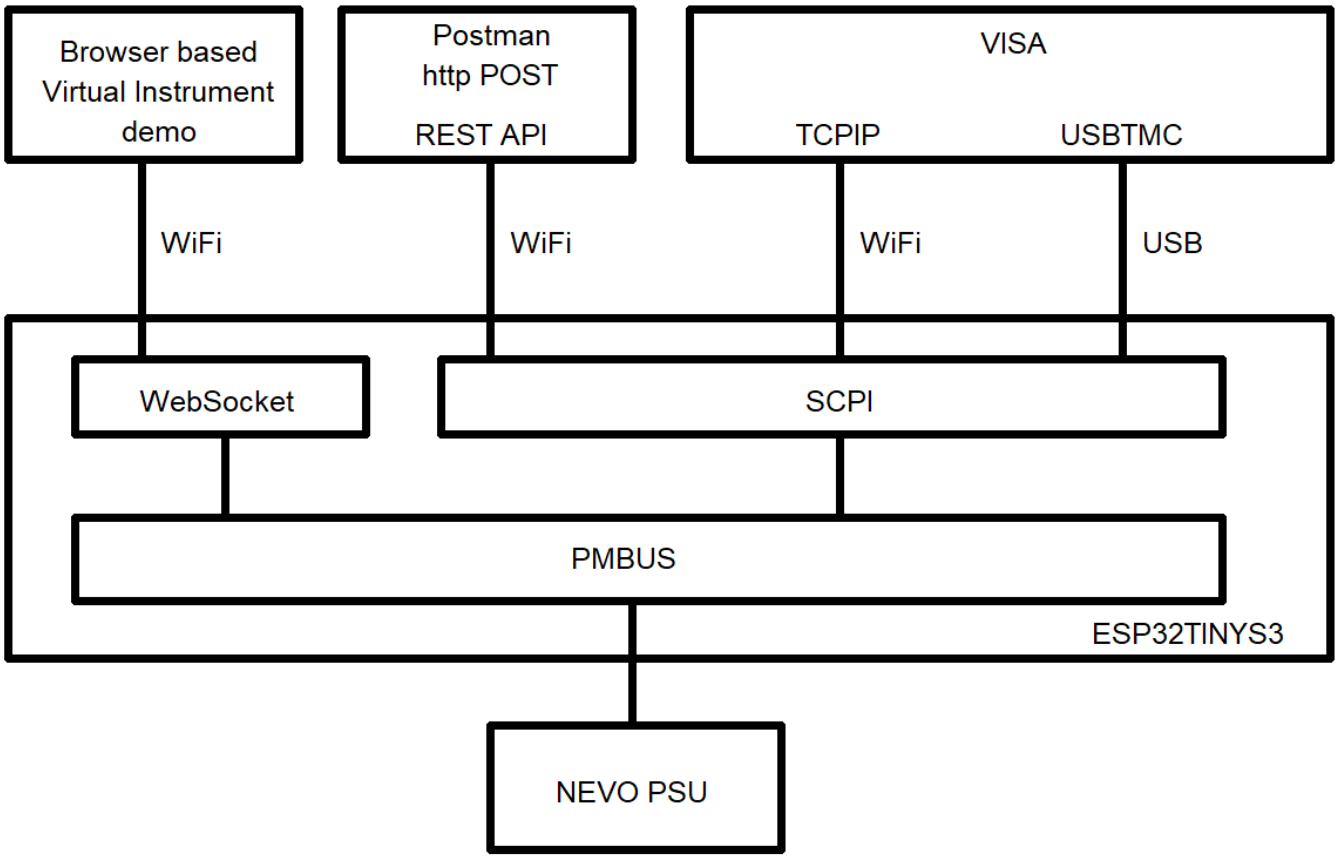
Open-source SCPI/Web control for NEVO+ Digital Output Modules (PMBus).

The NEVO+ series user manual has been prepared by the Vox Power design team to assist qualified engineers in correctly implementing the product and to achieve the best reliability and performance possible.

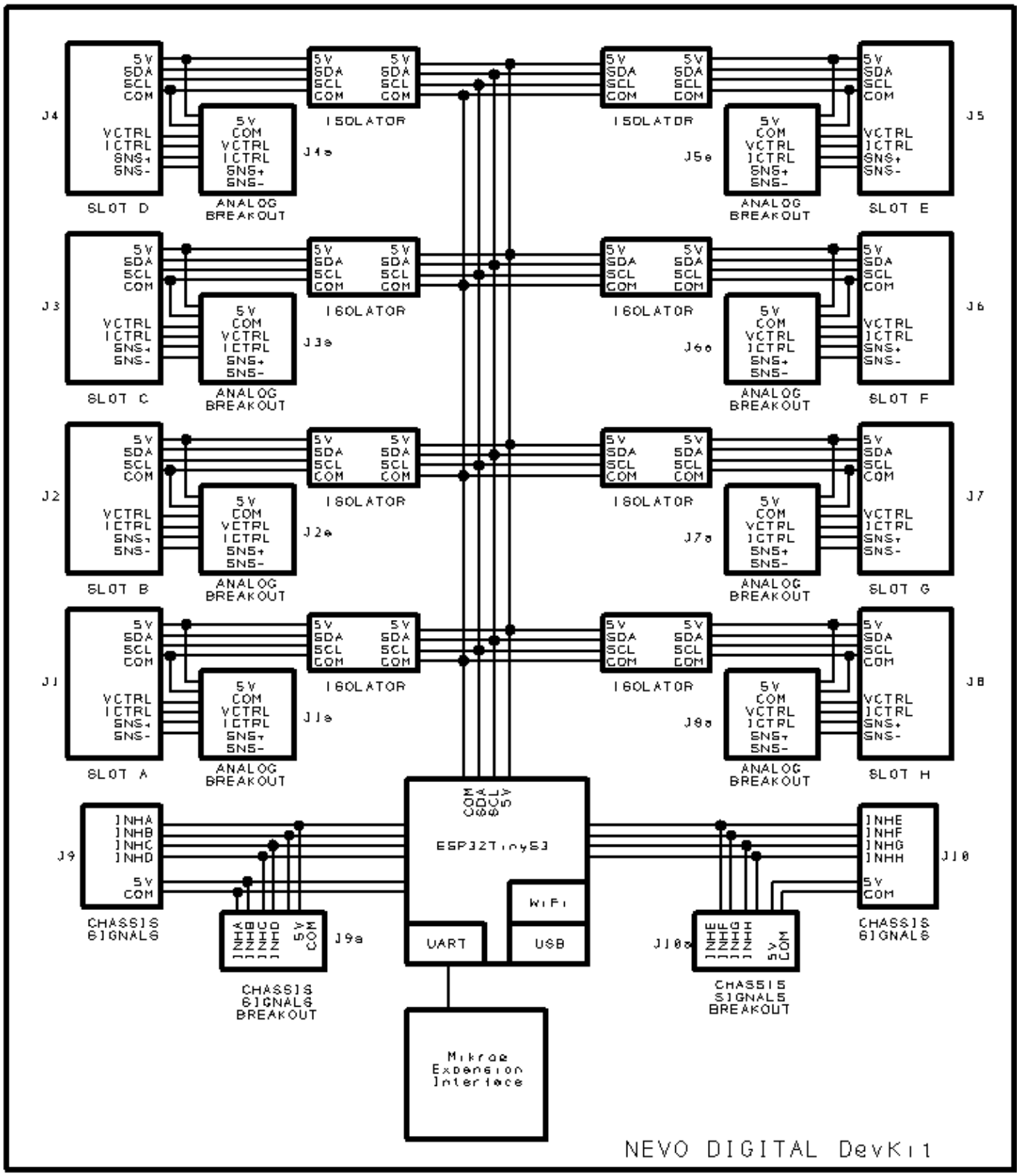
NEVO Digital Development Kit (DevKit) Overview

The NEVO digital DevKit is designed to simplify control and monitoring of up to 8 channels of fully isolated NEVO digitally controlled output modules. By utilising the DevKit, customers can significantly reduce development time of custom digital power supply solutions. The DevKit includes WiFi and USBTMC communication interfaces, a SCPI language interpreter and direct PMBUS interface that operate with the Virtual Instrument Software Architecture (VISA) or a REST API. There is also a web based virtual instrument demonstration with websocket communications. The DevKit hardware and software is open sourced and can be modified to suit any application. The hardware can be expanded using Mikroe plug in boards to implement several other communication protocols such as CANBUS, MODBUS, DALI, DMX and many more.

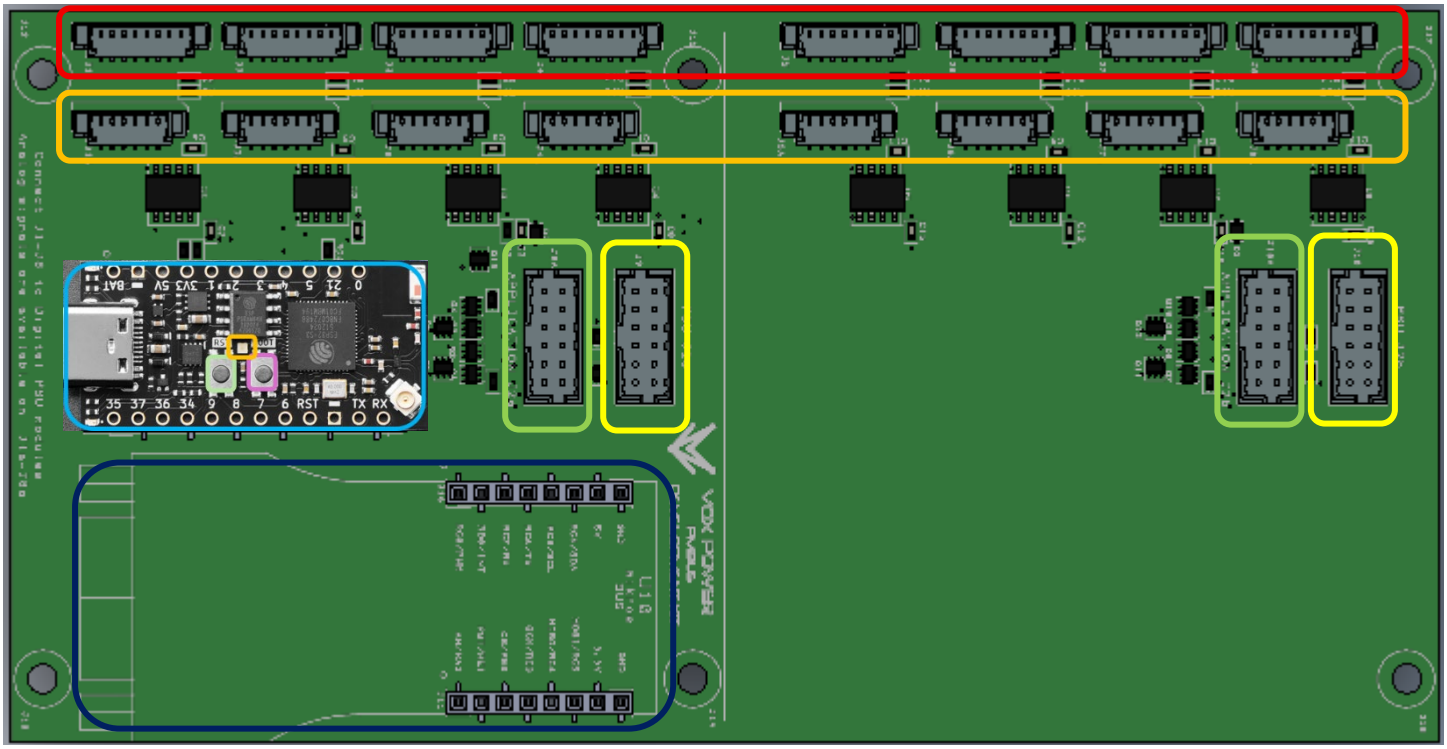
1.1 DevKit software architecture



1.2 DevKit hardware architecture



1.3 DevKit Physical layout



- J1 – J8: 8-way signals connectors to NEVO digital OP module signals
- J1a – J8a: 6-way breakout connector for analog signals & bias power.
- J9 – J10: 12-way connectors to NEVO chassis signals.
- J9a – J10a: 12-way breakout connectors for chassis signals.
- ESP32TinyS3 socket.
- LED indicator.
- BOOT button.
- RESET button.
- Mikroe expansion socket.

2.1 Connecting a NEVO PSU to the DevKit

With the power off, connect all digital OP modules to the DevKit digital interface. Connect the NEVO chassis signals to the DevKit. Once all connections are made the power can be switched on.
At powerup the DevKit will scan all NEVO slots to identify the modules. The LED will be red during this process and output modules may turn off momentarily.
If modules have identical SMBus addresses the duplicate addresses are automatically changed to ensure all modules have unique addresses.
Note the new SMBus addresses are not automatically stored in Non-Volatile Memory (NVM) and a power cycle will reset the addresses to their original values.
After the scan has completed, the PC communications systems (USB, WiFi, etc) are started, the LED will turn green, and you can communicate with the PSU.

2.2 Connecting the DevKit to a PC

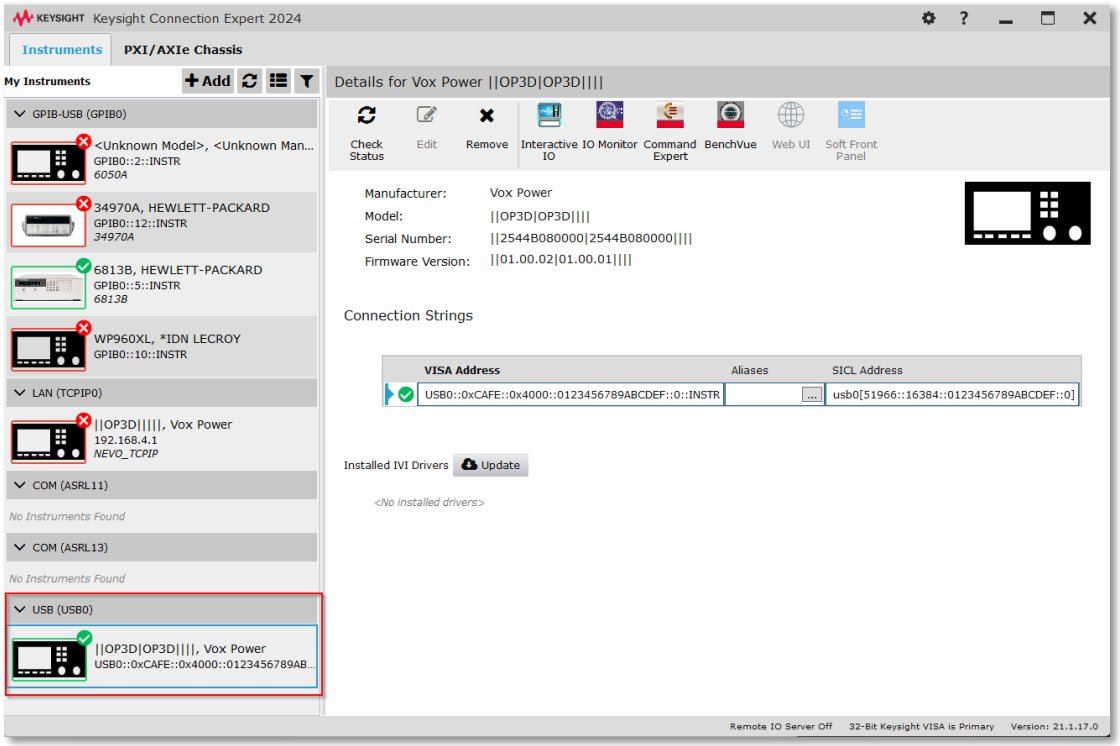
You can connect a PC to the DevKit via USB or WiFi. Note that the WiFi and USB interfaces are disabled until the NEVO PSU scan has been completed and the LED turns green.

2.2.1 USB connection

To connect via USB simply plug a USBC cable from the DevKit to a PC and power on the PSU.
Your PC device manager should automatically show a USBTMC device called "USB test and Measurement Device (IVI)"

2.2.1.1 Sending SCPI commands (VISA:USBTMC)

To communicate to the device, you must install VISA software. The IOLibrariesSuite from Keysight is recommended.
Once VISA is installed, open "Connection Expert" and the device should be automatically visible under the USB interface.



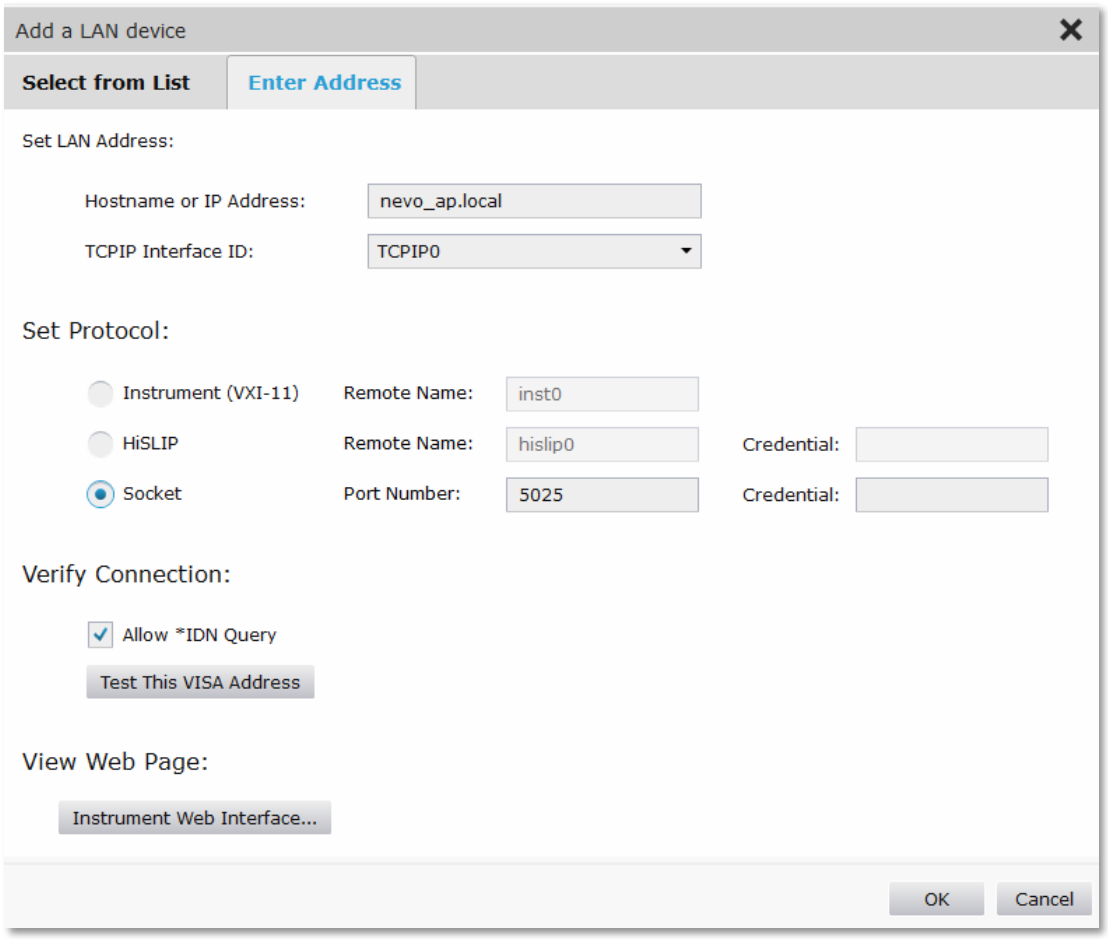
To send SCPI commands, right click the device and select "Send commands to this instrument".

2.2.2 WiFi connection

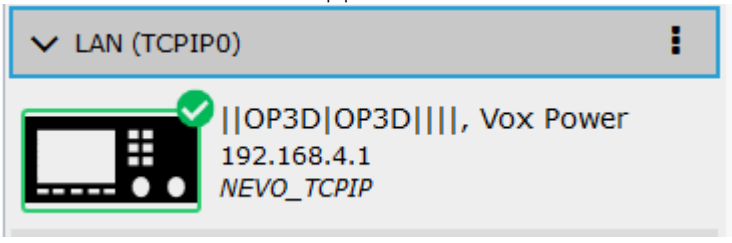
Once the NEVO PSU is powered the DevKit will start a WiFi access point called VOXPOWER_SOFTAP.
Scan for this with your PC WiFi adapter and connect to the access point. The DevKit can now be accessed for web demo, TCPIP VISA interface or REST API on the address NEVO_AP.local (or typical IP address 192.168.4.1).
Note that NEVO_AP.local will only work if mDNS is enabled on the PC.

2.2.2.1 Sending SCPI commands (VISA:TCPIP)

To setup TCPIP communications you must install VISA software. The IOLibrariesSuite from Keysight is recommended.
Once VISA is installed, open "Connection Expert", right click the LAN (TCPIP0) interface and select "Add instrument". Select the "Enter address" tab, setup as shown below and click OK.



The device should now appear under the LAN interface as shown below.

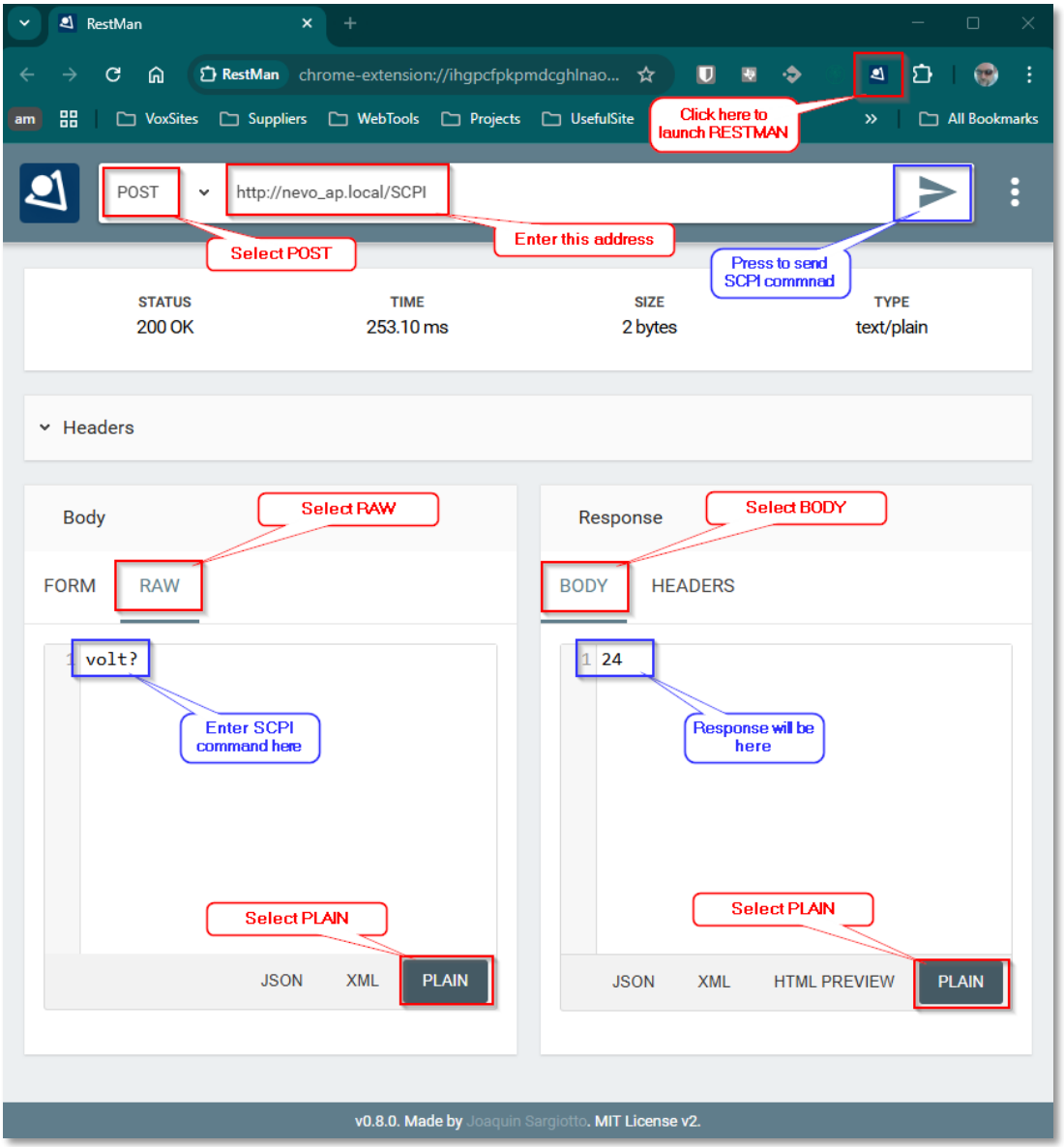


To send SCPI commands, right click the device and select "Send commands to this instrument".

2.2.2.2 Sending SCPI commands (REST-API)

To communicate via the REST API you must install a REST browser extension such as RESTMAN or other REST software. RESTMAN is available for most browsers in the relevant browser extensions websites. This example will use chrome browser.

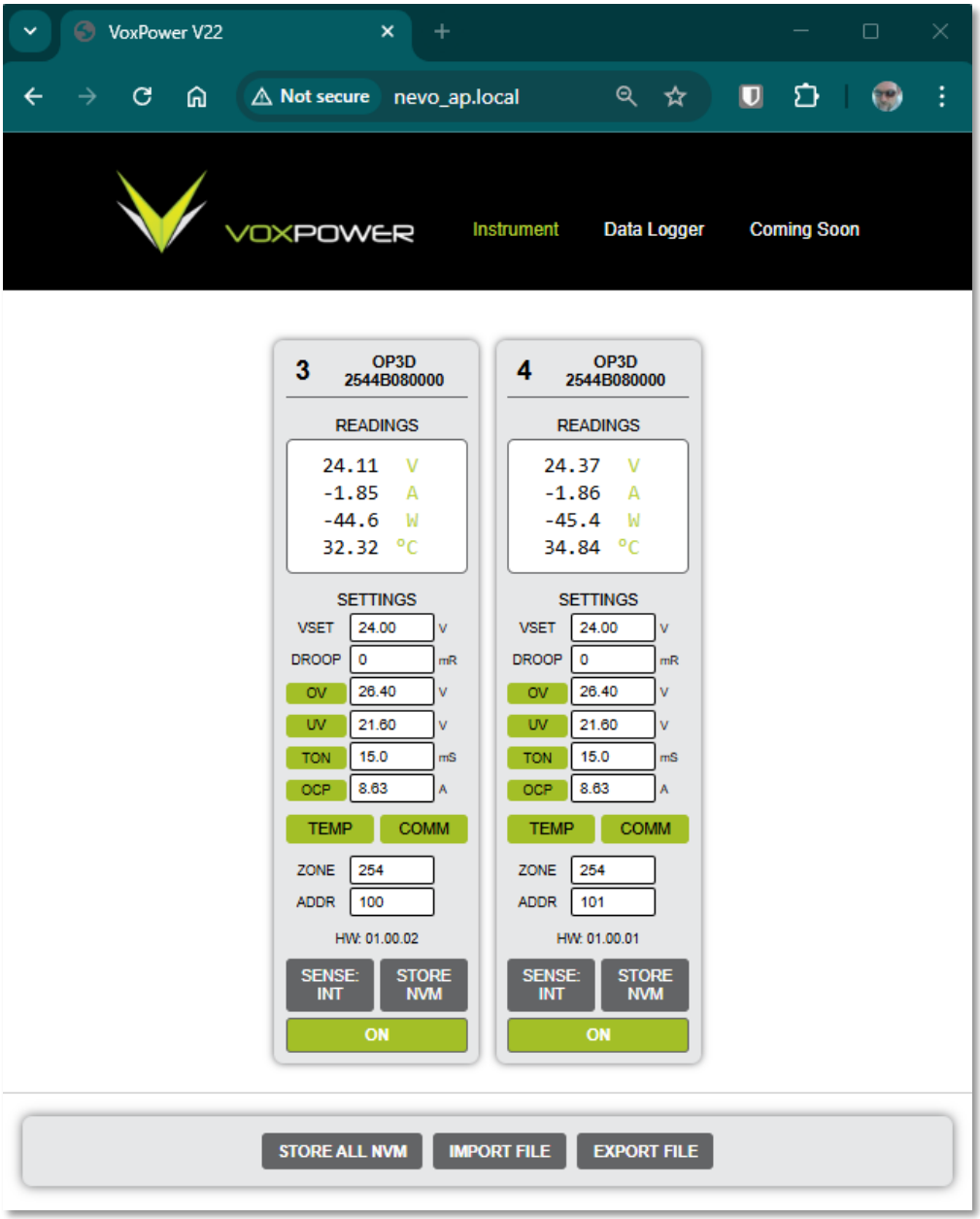
Once the extension is installed, activate the extension, setup and use as shown below.



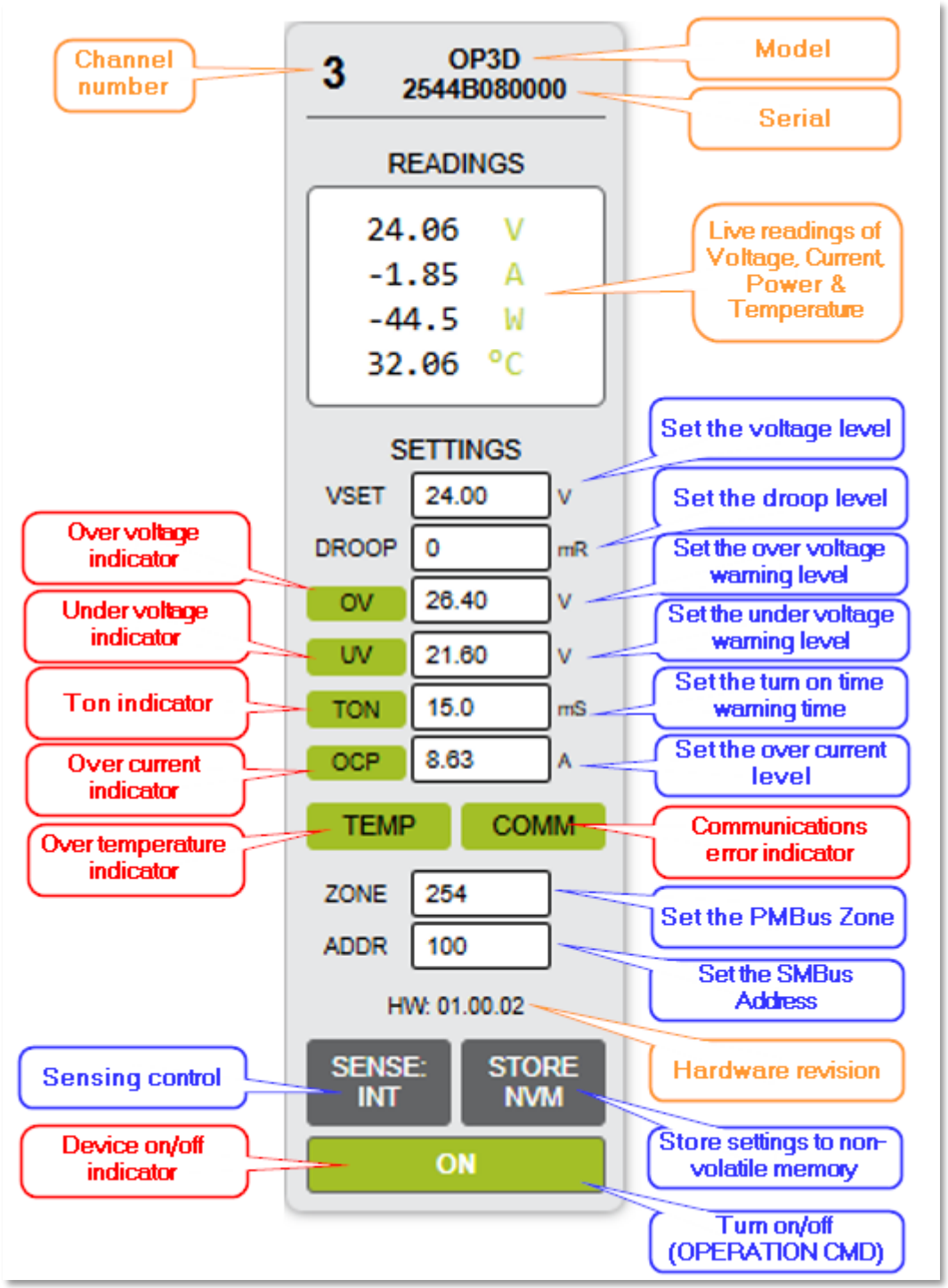
2.2.2.3 Web instrument demo

The web instrument demo can be accessed when connected to the DevKit via WiFi. To access the demo, open a browser and enter NEVO_AP.local

The demo gives a virtual instrument and datalogger for each channel connected to the DevKit.



The image below describes the features of a virtual instrument channel,



2.2.2.3.1 Zones

Multiple channels can be coupled to the same zone to receive commands simultaneously. This is recommended if channels are to be parallel or series connected.
If a channel is coupled to a valid zone, certain control commands will be sent to that zone instead of directly to the channel. The allowed zones are 0 to 254. Zone 254 is "No Zone". Channels set to zone 254 are not coupled to a valid zone and will receive their control commands directly.
Only the voltage, current and output subsystem control commands are affected see SCPI interface section for details.

2.2.2.3.2 Addresses

Any address from 1 to 127 can be selected and addresses must be unique. At startup the DevKit scans all channel addresses and changes addresses where necessary to ensure unique addresses are used. The changed addresses are not automatically stored to NVM.
After the power up sequence, if the user sets multiple channels to have the same address, communications will be corrupted and the system must be reset by power cycling or ESP32 reset button.

2.2.2.3.3 Global controls

There are three global control buttons at the bottom of the instrument webpage.

Store ALL NVM
This button stores all current data for all channels to non-volatile memory then recalls all stored settings to ensure they have been stored correctly. Settings should be reviewed after pressing this button to ensure they are correct.

Export File
This button exports all current settings to a JSON file. This file can be provided to the factory for pre-programming or used to simplify programming of multiple systems.

Import File
This button loads all settings from a previously exported JSON file.

2.2.3 Other connections

The hardware design is laid out to accept Mikroe expansion boards for various other communications protocols such as CANBUS, MODBUS, DALI etc.
See https://www.mikroe.com/click/interface for available interfaces.
However, no software implementation has been developed at this stage.

3.1 Numerical Data Formats

3.1.1 <NR1>

- **Format**: Signed integer.
- **Description**: Represents a whole number, positive or negative, without any decimal places.
- **Example**: 10, -5, 12345

3.1.2 <NR2>

- **Format**: Floating-point number with a fixed number of decimal places.
- **Description**: Represents a number that includes a decimal point, with a specific number of digits after it.
- **Example**: 3.14, -1.23, 0.50.

3.1.3 <crd>

- **Format**: Character Response Data
- **Description**: Alphanumeric strings that represent specific settings or states
- **Example**: CH1, ON, OFF, LOW, HIGH

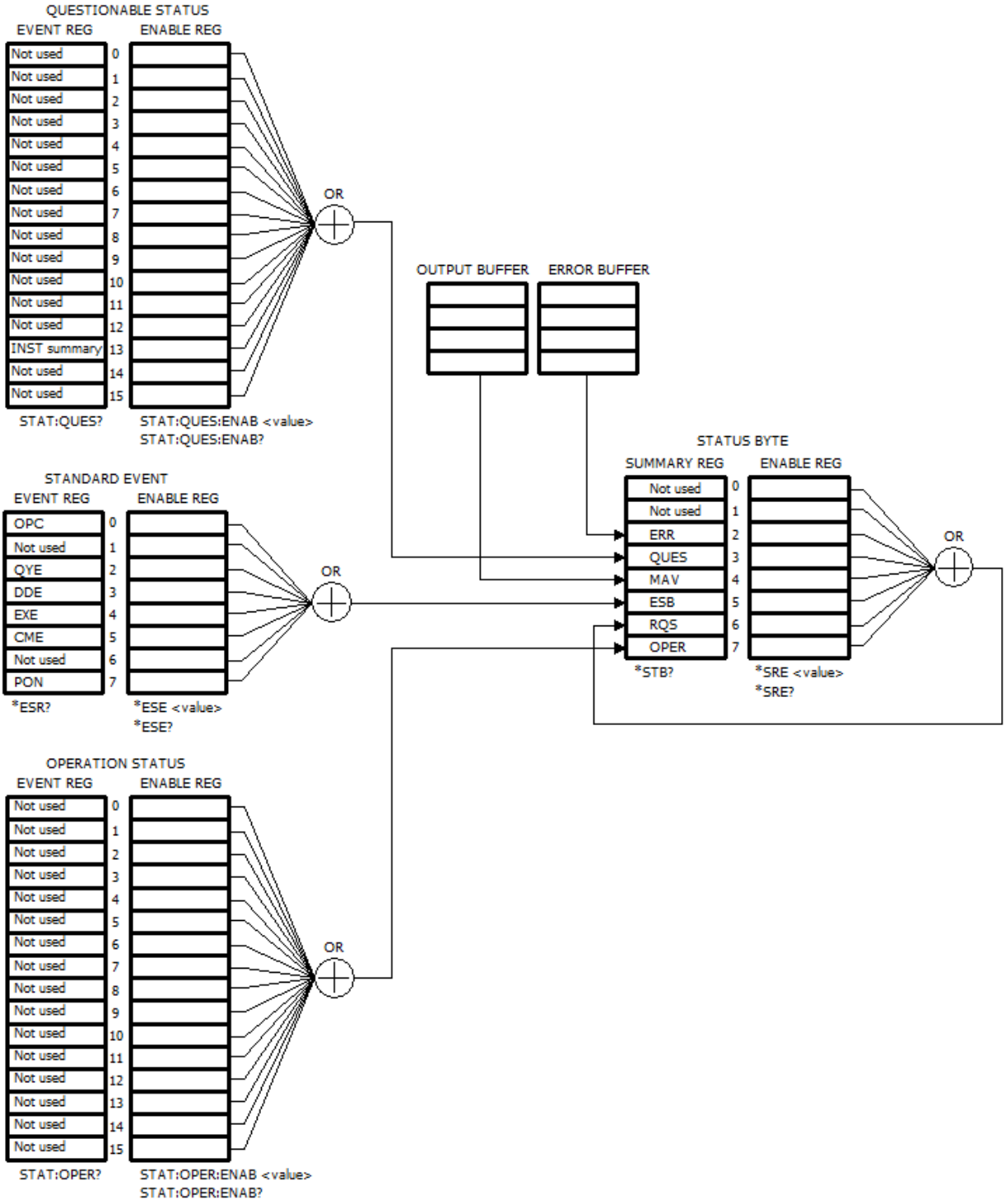
3.1.4 [Suffix]

- **Format**: Text strings
- **Description**: Can be appended to numeric parameters to specify the intended unit of measure and scale
- **Example**: V for Volts, mV for millivolts, A for amps, etc

3.2 Status registers

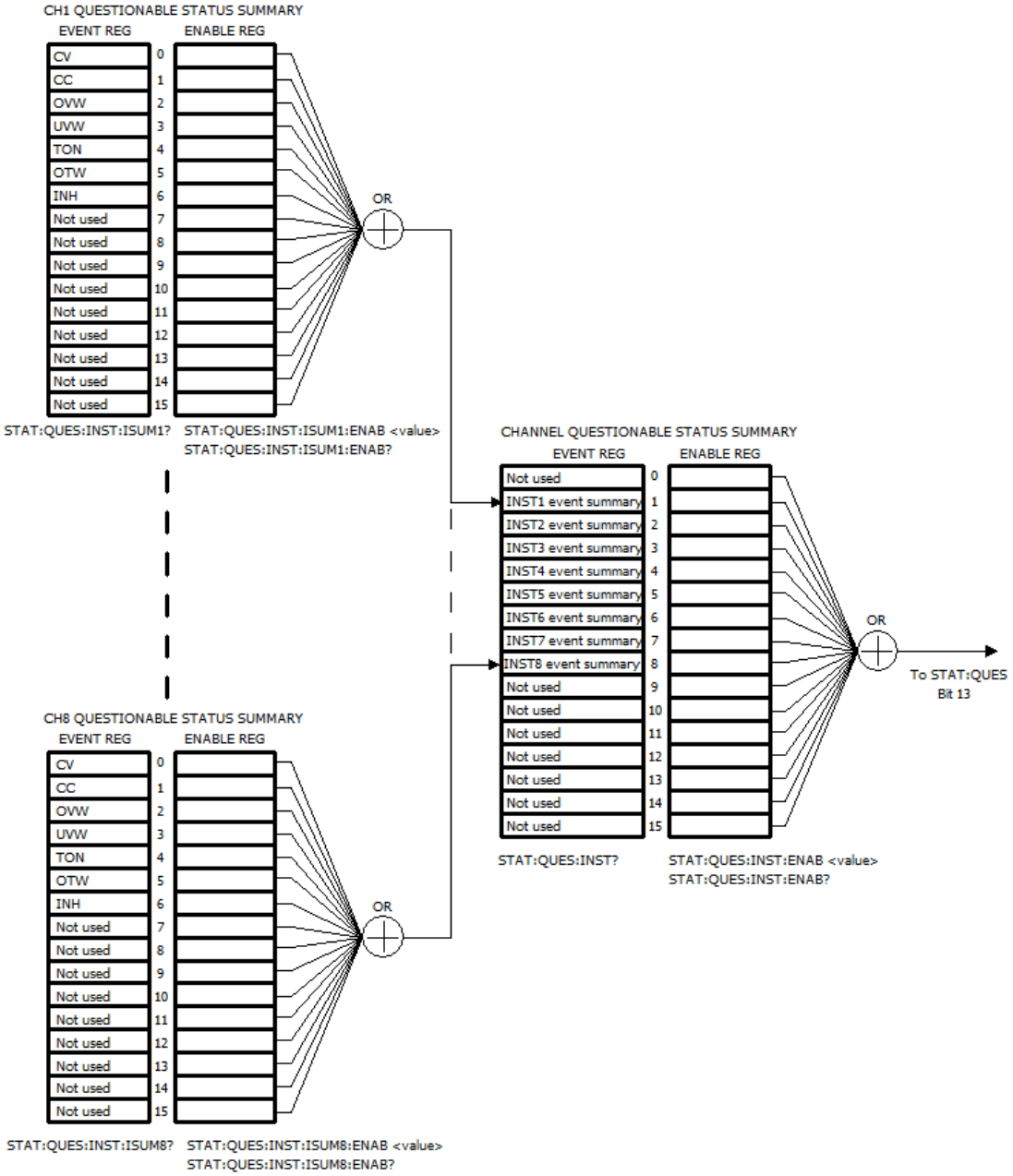
3.2.1 Overview

All SCPI instruments implement status registers in the same way. The status system records various instrument conditions in three register groups: the Status Byte register, the Standard Event register, and the Questionable Status register groups. The Status Byte register records high-level summary information reported in the other register groups. The figure below illustrates the SCPI status system implemented on the DevKit.



3.2.2 Questionable Instrument Summary Register

The eight logical outputs of the power supply include an INSTrument summary status register and an individual instrument ISUMmary register for each logical output. The ISUMmary registers report to the INSTrument register, which in turn reports to bit 13 of the Questionable status register, as shown in the diagram below. This status register configuration allows a status event to be cross-referenced by output and type of event. The INSTrument register indicates which output(s) have generated an event.



SUMMARY REGISTER BIT DESCRIPTION		
Bit	Name	Description
0	VOLTage	Output is in constant current mode. Voltage may be unregulated.
1	CURRent	Output is in constant voltage mode. Current may be unregulated.
2	OVW	Over voltage warning.
3	UVW	Under voltage warning.
4	TON	Turn on time warning.
5	OTW	Over temperature warning.
6	INH	Unit is off. (All other bits set to 0)

3.3 IEEE-488 Subsystem

3.3.1 Command summary

Command	Description
*CLS	Clears the error queue and all event registers except status byte.
*ESE <value>	Sets bits in the Standard Event Enable register.
*ESE?	Returns the binary weighted sum of the bits in the Standard Event Enable register.
*ESR?	Returns the binary weighted sum of the bits in the Standard Event register.
*IDN?	Returns the instruments identification string.
*OPC	Sets the "Operation Complete" bit of the Standard Event register when the instrument has completed all pending operations sent before *OPC.
*OPC?	Returns 1 to the output buffer when all pending operations complete.
*RST	Nothing implemented for this command.
*SRE <value>	Sets bits in the Status Byte Enable register.
*SRE?	Returns the binary weighted sum of the bits in the Status Byte Enable register.
*STB?	Returns the binary weighted sum of the bits in the Status Byte register.
*TST?	Nothing implemented for this command.
*WAI	Nothing implemented for this command.

3.3.2 Instruments identification string (*IDN?)

The query returns the instrument's identification string. An example is shown below.

Vox Power,
OP1D|OP1D|OP1D|OP1D||||,
2432D010000|2432D010001|2432D010002|2432D010003|||||, 01.00.00|01.00.00|01.00.00|01.00.00|||||

The four comma-separated fields are the manufacturer's name, the model number, the serial number, and the revision code. Up to 8 channels are listed, separated by |.

3.3.3 Standard Event Register (*ESR?)

STANDARD EVENT REGISTER BIT DESCRIPTION		
Bit	Name	Description
0	OPC	Operation Complete.
1	Not Used	Always 0
2	QYE	Query error.
3	DDE	Device error.
4	EXE	Execution error.
5	CME	Command error.
6	Not used	Always 0
7	PON	Power On. Set when power is turned on.

3.3.4 Status Byte (*STB?)

STATUS BYTE BIT DESCRIPTION		
Bit	Name	Description
0-1	Not used	Always 0
2	ERR	Error message available.
3	QUES	Questionable status register flag
4	MAV	Output message available
5	ESB	Standard event register flag
6	RQS	Request for service
7	OPER	Operation status register flag

3.4 INSTrument Subsystem

3.4.1 INSTrument:SELect

Description: The command selects the default channel to be programmed. The query returns the channel currently selected by INSTrument [:SELect].	
Command Syntax: INSTrument:SELect <ch>	**Command parameters:** <ch> CH1 – CH8
Query Syntax: INSTrument:SELect?	**Returned Parameters:** <crd> Character response data. E.g. CH1
Examples: INST:SEL CH3 INST:SEL?	**Default:** Lowest channel present

3.4.2 INSTrument:COUPle

Description: The command couples the selected channel to a zone. Multiple channels can be coupled to the same zone to receive commands simultaneously. If a channel is coupled to a valid zone, certain control commands will be sent to that zone instead of directly to the channel. The allowed zones are 0 to 254. Zone 254 is "No Zone". Channels set to zone 254 are not coupled to a valid zone and will receive their control commands directly. Only the voltage, current and output subsystem control commands are affected. Queries are always performed directly on each channel. If the optional <ch> parameter is omitted the channel selected with the INSTrument:SELect command is used. The query returns the zone currently assigned to the selected channel.	
Command Syntax: INSTrument:COUPle <[ch>],<zone>	**Command parameters:** <ch> CH1 – CH8 <zone> Zone address 0-254 254 = No zone.
Query Syntax: INSTrument:COUPle? [<ch>],[<param>]	**Query Parameters:** <ch> CH1 – CH8 <param> MIN | MAX | DEF **Returned Parameters:** <NR1> Integer Digits E.g. 100
Examples: INST:COUP CH3,100 INST:COUP?	**Default:** 254

3.5 MEASure Subsystem

3.5.1 MEASure:VOLTage?

Description: The query returns the voltage measured at the output of the specified channel. If the optional <ch> parameter is omitted the channel selected with the INSTrument:SELect command is used.	
Command Syntax: None	**Command parameters:** None
Query Syntax: MEASure:VOLTage? [<ch>]	**Query Parameters:** <ch> CH1 – CH8 **Returned Parameters:** <NR2> Digits with a decimal point. E.g. 10.12
Examples: MEAS:VOLT? MEAS:VOLT? CH1	**Default:** Not applicable

3.5.2 MEASure:CURRent?

Description: The query returns the output current of the specified channel. If the optional <ch> parameter is omitted the channel selected with the INSTrument:SELect command is used.	
Command Syntax: None	**Command parameters:** None
Query Syntax: MEASure:CURRent? [<ch>]	**Query Parameters:** <ch> CH1 – CH8 **Returned Parameters:** <NR2> Digits with a decimal point. E.g. 10.12
Examples: MEAS:CURR? MEAS:CURR? CH1	**Default:** Not applicable

3.5.3 MEASure:POWer?

Description: The query returns the output power measured from the specified channel. If the optional <ch> parameter is omitted the channel selected with the INSTrument:SELect command is used.	
Command Syntax: None	**Command parameters:** None
Query Syntax: MEASure:POWer? [<ch>]	**Query Parameters:** <ch> CH1 – CH8 **Returned Parameters:** <NR2> Digits with a decimal point. E.g. 10.12
Examples: MEAS:POW?	**Default:** Not applicable

3.5.4 MEASure:TEMPerature?

Description: The query returns the output temperature in degrees Celsius measured from the specified channel. If the optional <ch> parameter is omitted the channel selected with the INSTrument:SELect command is used. If the channel is specified but is not initialised an ILLEGAL_PARAMETER error will be generated.	
Command Syntax: None	**Command parameters:** None
Query Syntax: MEASure:TEMPerature? [<ch>]	**Query Parameters:** <ch> CH1 – CH8 **Returned Parameters:** <NR2> Digits with a decimal point. E.g. 10.12
Examples: MEAS:TEMP?	**Default:** Not applicable

3.6 OUTPut Subsystem

3.6.1 OUTPut:STATe

Description: The command turns the specified channel on or off using the PMBus OPERATION command. The query returns the current setting for the specified channel. If the optional <ch> parameter is omitted the channel selected with the INSTrument:SELect command is used. MAX, MIN and DEF can be used to program or query the maximum, minimum or default settings.	
Command Syntax: OUTPut:STATe [<ch>],<param>	**Command parameters:** <ch> CH1 – CH8 <param> ON (1) | OFF (0)
Query Syntax: OUTPut:STATe? [<ch>],[<param>]	**Query Parameters:** <ch> CH1 – CH8 <param> MIN | MAX | DEF **Returned Parameters:** <crd> Character response data. E.g. ON
Examples: OUTP:STAT ON, OUTP:STAT?	**Default:** ON

3.6.2 OUTPut:STATe:TONMax

Description: The command programs the maximum startup time of the selected channel in milliseconds using the PMBus TON_MAX_FAULT_LIMIT command. The query returns the current setting for the specified channel in milliseconds. If the optional <ch> parameter is omitted the channel selected with the INSTrument:SELect command is used. MAX, MIN and DEF can be used to program or query the maximum, minimum or default settings.	
Command Syntax: OUTPut:STATe:TONMax [<ch>],<param>	**Command parameters:** <ch> CH1 – CH8 <param> MINimum | DEFault | MAXimum | <NR2>[suffix]
Query Syntax: OUTPut:STATe:TONMax? [<ch>],[<param>]	**Query Parameters:** <ch> CH1 – CH8 <param> MIN | MAX | DEF **Returned Parameters:** <NR2> Digits with a decimal point. E.g. 10.12
Examples: OUTP:STAT:TONM 15 OUTP:STAT:TONM?	**Default:** See OP module datasheet

3.7 VOLTage Subsystem

3.7.1 VOLTage

Description: The command programs the output voltage level of the specified channel in volts. The query returns the output voltage level setting for the specified channel in volts. If the optional <ch> parameter is omitted the channel selected with the INSTrument:SELect command is used. MAX, MIN and DEF can be used to program or query the maximum, minimum or default settings.	
Command Syntax: VOLTage [<ch>],<param>	**Command parameters:** <ch> CH1 – CH8 <param> MINimum | DEFault | MAXimum | <NR2>[suffix]
Query Syntax: VOLTage? [<ch>],[<param>]	**Query Parameters:** <ch> CH1 – CH8 <param> MIN | MAX | DEF **Returned Parameters:** <NR2> Digits with a decimal point. E.g. 10.12
Examples: VOLT 5.55 VOLT?	**Default:** See OP module datasheet

3.7.2 VOLTage:DROop

Description: The command programs the output voltage droop level of the specified channel in mΩ. The query returns the output voltage droop level setting for the specified channel in mΩ. If the optional <ch> parameter is omitted the channel selected with the INSTrument:SELect command is used. MAX, MIN and DEF can be used to program or query the maximum, minimum or default settings.	
Command Syntax: VOLTage:DROop [<ch>],<param>	**Command parameters:** <ch> CH1 – CH8 <param> MINimum | DEFault | MAXimum | <NR2>[suffix]
Query Syntax: VOLTage:DROop? [<ch>],[<param>]	**Query Parameters:** <ch> CH1 – CH8 <param> MIN | MAX | DEF **Returned Parameters:** <NR2> Digits with a decimal point. E.g. 10.12
Examples: VOLT:DRO 100 VOLT:DRO?	**Default:** 0

3.7.3 VOLTage:LIMit:LOW

Description: The command programs the output under-voltage level of the specified channel in volts. The query returns the output under-voltage level setting for the specified channel in volts. If the optional <ch> parameter is omitted the channel selected with the INSTrument:SELect command is used. MAX, MIN and DEF can be used to program or query the maximum, minimum or default settings.	
Command Syntax: VOLTage:LIMit:LOW [<ch>],<param>	**Command parameters:** <ch> CH1 – CH8 <param> MINimum | DEFault | MAXimum | <NR2>[suffix]
Query Syntax: VOLTage:LIMit:LOW? [<ch>],[<param>]	**Query Parameters:** <ch> CH1 – CH8 <param> MIN | MAX | DEF **Returned Parameters:** <NR2> Digits with a decimal point. E.g. 10.12
Examples: VOLT:LIM:LOW 8.54 VOLT:LIM:LOW?	**Default:** See OP module datasheet

3.7.4 VOLTage:LIMit:HIGH

Description: The command programs the output over-voltage level of the specified channel in volts. The query returns the output over-voltage level setting for the specified channel in volts. If the optional <ch> parameter is omitted the channel selected with the INSTrument:SELect command is used. MAX, MIN and DEF can be used to program or query the maximum, minimum or default settings.	
Command Syntax: VOLTage:LIMit:HIGH [<ch>],<param>	**Command parameters:** <ch> CH1 – CH8 <param> MINimum | DEFault | MAXimum | <NR2>[suffix]
Query Syntax: VOLTage:LIMit:HIGH? [<ch>],[<param>]	**Query Parameters:** <ch> CH1 – CH8 <param> MIN | MAX | DEF **Returned Parameters:** <NR2> Digits with a decimal point. E.g. 10.12
Examples: VOLT:LIM:HIGH 10.54 VOLT:LIM:HIGH?	**Default:** See OP module datasheet

3.7.5 VOLTage:SENSe[:SOURce]

Description: The command configures where the specified channel senses voltage. Internal sensing (INT) monitors the directly at the output terminals while external sensing (EXT) senses voltage at the sense terminals. The query returns the sense setting for the specified channel. If the optional <ch> parameter is omitted the channel selected with the INSTrument:SELect command is used. MAX, MIN and DEF can be used to program or query the maximum, minimum or default settings.	
Command Syntax: VOLTage:SENSe[:SOURce] [<ch>],<param>	**Command parameters:** <ch> CH1 – CH8 <param> INT (1) | EXT (0)
Query Syntax: VOLTage:SENSe[:SOURce]? [<ch>],[<param>]	**Query Parameters:** <ch> CH1 – CH8 <param> MIN | MAX | DEF **Returned Parameters:** <crd> Character response data. E.g. INT
Examples: VOLT:SENS EXT VOLT:SENS?	**Default:** INT

3.8 CURRent Subsystem

3.8.1 CURRent[:PROTection]

Description: The command programs the current limit level of the specified channel in amperes. The query returns the current limit level setting for the specified channel in amperes. If the optional <ch> parameter is omitted the channel selected with the INSTrument:SELect command is used. MAX, MIN and DEF can be used to program or query the maximum, minimum or default settings.	
Command Syntax: CURRent[:PROTection] [<ch>],<param>	**Command parameters:** <ch> CH1 – CH8 <param> MINimum | DEFault | MAXimum | <NR2>[suffix]
Query Syntax: CURRent[:PROTection]?	**Returned Parameters:** <NR2> Digits with a decimal point. E.g. 10.12
Examples: CURR 10 CURRENT:PROTECTION 15 CURR:PROT MAX	**Default:** Maximum

3.9 STATus Subsystem

The status subsystem consists of 4 register groups as described below. Each register group has 3 registers. A condition register which shows the current state, an event register which latches events from the condition register and an enable register which enables bits from the event register to set bits in following registers. See chapter 4.2, status register for register organisation.
The condition register is queried by appending :CONDition? to the main register header.
The event register is queried by appending [:EVENt]? to the main register header.
The enable register is queried by appending :ENABle? to the main register header.
The enable register can be set by appending :ENABle <value> to the main register header. <value> is binary weighted sum of the bits to be set.

3.9.1 STATus:OPERation

This register tracks the instrument's operational conditions.

3.9.2 STATus:QUEStionable:INSTrument:ISUMmary<n>

This register tracks events affecting measurement quality for each channel, like overloads or range issues, indicating *why* a reading might be inaccurate. <n> defines the channel and can be any number from 1 to 8. Events from these registers can be route to the STATus:Questionable:INSTrument register using the enable register.

3.9.3 STATus:QUEStionable:INSTrument

This register summarises events from the STATus:QUEStionable:INSTrument:ISUMmary<n> registers. Events from this register can be routed to the STATus:QUEStionable register using the enable register.

3.9.4 STATus:QUEStionable

This register summarises events from the STATus:QUEStionable:INSTrument register. Events from this register can be routed to the Status Byte register using the enable register.

3.9.5 STATus:PRESet

The command clears all the Operation and Questionable enable registers.

3.10 SYSTem Subsystem

3.10.1 SYSTem:ERRor[:NEXT]?

The query returns the next error message on the device's error queue of up to 17 errors. See SCPI Error Messages.

3.10.2 SYSTem:ERRor:COUNt?

This command gets the number of errors in the error queue. If the queue is empty zero will be returned.

3.10.3 SYSTem:VERSion?

The query returns the present SCPI version of the power supply. The returned value is a string in the form of YYYY.V where "YYYY" represent the year of the version, and the "V" represents the current version number of the SCPI.

3.11 PMBUS Subsystem

3.11.1 PMBUS

Description:
The command only allows for PMBus commands to be sent directly to devices on the PMBus. Commands may be sent to channels using CH1 – CH8, all devices using GC (General call), to a PMBus zone using ZONE or to a specific SMBus address using DIR (Direct).
The query only allows for PMBus queries to be sent directly to devices on the PMBus. Return types depend on the query command. Queries may be sent to channels using CH1 – CH8 or to a specific SMBus address using DIR (Direct).
If no channel is specified, the device selected with the INSTrument command is used.
If the query is sent without any parameters, it will return the SMBus address of the device selected with the INSTrument command.

Command Syntax:	**Command Parameters:**
PMBUS [<ch>],[<add>],<cmd>,[<type>],[<data>]	<ch> CH1 – CH8 | GC | ZONE | DIR See special functions for GC, ZONE and DIR <cmd> PMBUS command code 0-255 <type> send_byte, write_byte, write_word, write_block <data> Data NR1
Query Syntax:	**Query Parameters:**
PMBUS? [<ch>],<cmd>,[<type>],[<bytes>]	<ch> CH1 – CH8 | GC | DIR See special functions for GC and DIR <cmd> PMBUS command code 0-255 <type> read_byte, read_word, read_block, read_string <bytes> Number of bytes to read 0 – 12 **Query returned Parameters:** Various depending on CMD

Examples:

PMBUS 126,255	– Write 255 to STATUS_CML on INST:SEL channel
PMBUS CH3,3	-CLEAR_FAULTS on CH3
PMBUS?	– Return SMBus address for INST:SEL channel
PMBUS? CH2	– Return SMBus address for CH2

Special functions:
Special functions can be invoked by setting <ch> to GC, ZONE or DIR for commands or DIR for queries.
Parameters for special function commands may include one or more of the following,

Parameter	Description	<data>
<add>	SMBus address	Integer 0-127
<zadd>	SMBus zone address	Integer 0-253
<cmd>	PMBus command	Integer 0-255
<type>	Transaction type	See below
<data>	Transaction data	See below
<bytes>	Number of bytes to read	See below

For special function commands the required <data> parameter depends on the <type> parameter. The list of valid <type> and <data> parameters are shown below.

Parameter	Description	<data>
<type>	send_byte	Not required
<type>	write_byte	Integer 0-255
<type>	write_word	Integer 0-65536
<type>	write_block	List of integers 0-255. Maximum 22 bytes

For special function queries the required <bytes> parameter depends on the <type> parameter. The list of valid <type> and <bytes> parameters are shown below.

Parameter	Description	<bytes>
<type>	read_byte	Not required
<type>	read_word	Not required
<type>	read_block	Number of bytes to read. Maximum 12 bytes
<type>	read_string	Number of characters to read. Maximum 12 characters

For special function commands or queries, <type> may be omitted for commands listed in the OP module datasheet and <data> may be omitted for commands that do not require data.

GC – General call

Send commands or queries to all devices on general call address. Returned parameters from queries are likely to be corrupted if multiple devices respond.

Parameters: <cmd>,[<type>],[<data>]

Examples:

PMBUS GC,33,write_word,500	-Writes 500 to register 33 (VOUT_COMMAND) on all devices.
PMBUS GC,33,500	-Same as above with <type> omitted.
PMBUS GC,3	-Sends CLEAR_FAULTS command to all devices.
PMBUS? GC,208	-Reads from register 208 (MFR_SMBUS_ADDRESS) on all devices.

ZONE – Zone write

Send commands to all devices configured to the specified zone address.

Parameters: <zadd>,<cmd>,[<type>],[<data>]

Examples:

PMBUS ZONE,1,33,write_word,500	-Writes 500 to register 33 (VOUT_COMMAND) on all devices configured for zone address 1.
PMBUS ZONE,53,3	-Sends CLEAR_FAULTS command to all devices configured for zone address 53.

DIR – Direct (commands)

Sends commands to devices at any SMBus address using any valid <type> format.

Parameters: <add>,<cmd>,<type>,[<data>]

Examples:

PMBUS DIR,100,33,write_word,500	-Writes 500 to register 33 (VOUT_COMMAND) on the device with SMBus address 100.
PMBUS DIR,100,158,write_block,23,53,44,56	-Writes the sequence 23,53,44,56 to register 158 on the device with SMBus address 100.

DIR – Direct (queries)

Sends queries to devices at any SMBus address using any valid <type> format.

Parameters: <add>,<cmd>,[<type>],[<bytes>]

Examples:

PMBUS? DIR,100,33	-Reads the value stored in register 33 (VOUT_COMMAND) on the device with SMBus address 100.
PMBUS? DIR,100,158,read_block,8	-Reads 8 bytes from register 158 (MFR_SERIAL) on the device with SMBus address 100. Bytes are returned as a sequence of integers.
PMBUS? DIR,100,158,read_string,8	-Reads 8 bytes from register 158 (MFR_SERIAL) on the device with SMBus address 100. Bytes are returned as a string.

3.12 GPIO Subsystem

3.12.1 GPIO

Description: The command turns the specified channel on or off using the NEVO inhibit signals. The query returns the current setting for the device selected with the INSTrument command.	
Command Syntax: GPIO <ch>,<state>	**Parameters:** <ch> CH1-CH8 | ALL <state> HIGH (1) | LOW (0)
Query Syntax: GPIO?	**Returned Parameters:** <crd> Character response data. E.g. ON
Examples:	
GPIO ALL,HIGH	-Inhibits all channels.
GPIO CH1,LOW	-Enables channel 1.
Default: LOW	**Related commands:**

3.13 SCPI Error Messages

The error queue is 17 messages long and error retrieval is first-in-first-out (FIFO). Errors are cleared as you read them. If more than 17 errors have occurred, the last error stored in the queue (the most recent error) is replaced with - 350,"Queue overflow". No additional errors are stored until you remove errors from the queue. If no errors have occurred when you read the error queue, the instrument responds with +0,"No error". Sending SYSTem:ERRor? reads the most recent error. Each error is in the format: -104,"Data type error". The error queue is cleared by power cycles and *CLS, but not *RST. The instrument's error codes are listed below:

Code	Text	Description
0	SCPI_ERROR_NO_ERROR	No error This is the response to the ERR? query when there are no errors
-101	SCPI_ERROR_INVALID_CHARACTER	An invalid character was found in the command string
-103	SCPI_ERROR_INVALID_SEPARATOR	An invalid separator was found in the command string
-104	SCPI_ERROR_DATA_TYPE_ERROR	The wrong parameter type was found in the command string
-108	SCPI_ERROR_PARAMETER_NOT_ALLOWED	More parameters were received than expected for the command
-109	SCPI_ERROR_MISSING_PARAMETER	Fewer parameters were received than expected for the command
-113	SCPI_ERROR_UNDEFINED_HEADER	A command was received that is not valid for this device
-114	SCPI_ERROR_HEADER_SUFFIX_OUTOFRANGE	The numeric suffix attached to a command header is not one of the allowable values
-131	SCPI_ERROR_INVALID_SUFFIX	A suffix was incorrectly specified for a numeric parameter.
-138	SCPI_ERROR_SUFFIX_NOT_ALLOWED	A suffix was received following a numeric parameter which does not accept a suffix.
-151	SCPI_ERROR_INVALID_STRING_DATA	An invalid character string was received.
-170	SCPI_ERROR_EXPRESSION_PARSING_ERROR	A generic expression error.
-200	SCPI_ERROR_EXECUTION_ERROR	Unable to execute command.
-222	SCPI_ERROR_DATA_OUT_OF_RANGE	A numeric parameter value is outside the valid range for the command.
-224	SCPI_ERROR_ILLEGAL_PARAMETER_VALUE	A discrete parameter was received which was not a valid choice for the command.
-241	SCPI_ERROR_HARDWARE_MISSING	The command could not be executed because of missing hardware. If a channel is specified but is not initialised a HARDWARE_MISSING error will be generated.
-310	SCPI_ERROR_SYSTEM_ERROR	System error.
-350	SCPI_ERROR_QUEUE_OVERFLOW	Error buffer overflow
-360	SCPI_ERROR_COMMUNICATION_ERROR	PMBus communication error
-363	SCPI_ERROR_INPUT_BUFFER_OVERRUN	Input buffer overrun

Modifying the DevKit code

1. Clone the repository.
2. Modify the code.
3. Unit must be powered or ESP32TinyS3 board removed from socket.
4. Put the ESP32 in program mode: press RESET & BOOT buttons together. Release RESET button first, then release BOOT button.
5. Program then press RESET button to restart with updated code.

Note: Mikroe USB UART 5 click is recommended for debug console output.