PoKeys Pulse engine v2 documentation

Version: 6/3/2016

SAFETY INFORMATION

This product is intended for integration by the user into a computer numerical control (CNC) machine. It is the user's responsibility to assess the overall system design and address all safety considerations that affect the users and equipment. The user assumes all responsibility for system design, including compliance with regulatory standards and codes issued by the applicable entities. PoLabs do not make any claims as to the suitability of this equipment for the user's application. Serious personal injury or equipment damage can occur from the improper integration, installation or operation of this product.

This product is not guaranteed to be fail-safe. The system that this equipment is used with shall be fitted with a separate means of fail-safe protection, emergency-stop capability and/or system power removal. This equipment may be connected to dangerous power sources, including electrical power sources. Dangerous voltage levels may be present at this equipment or at connected devices. Measures must be taken to prevent persons from contacting voltage sources which may be present. Equipment should be housed inside an enclosure suitable for the intended environment. Safety interlocks should be provided to prevent any and all dangers to personnel.

CNC machine tools are inherently dangerous, and can cause injury to operators and maintenance personnel. Operators and maintenance personnel shall be properly trained in the safe use, operation and maintenance of such machines. Automated machines that this equipment may be used with can move at any time. All persons exposed to such machines must understand the dangers that are present.

Description

PoKeys Pulse engine v2 (upgrade of the original PoKeys Pulse engine) is available on PoKeys56U, PoKeys56E and PoKeys57E devices and enables a direct control of a positioning systems that accepts step/direction signals (stepper motors, servo systems, etc.). PoKeys Pulse engine can operate in standalone (rapid positioning and speed control mode, homing, probing) or in slave mode (under the commands of the PC application). Each axis supports two end (limit) switch inputs, one home (reference position) switch input, which can be either a dedicated one or shared with limit switch inputs. All inputs can are fully configurable (input polarity, placement) using the PoKeys protocol (used by PoKeys application or a third party application for motion control).

In addition, operations in each axis can be limited using the soft limit function, where minimum and maximum permissible axis position is specified - once enabled, the axis is put to a stop when exceeding the allowed range, but is free to move in other direction.

Each axis can be assigned a different encoder with a customizable multiplication factor for the operation with MPG (manual pulse generator).

PoKeys Pulse engine also supports a dedicated emergency switch input, which stops the pulse generation.

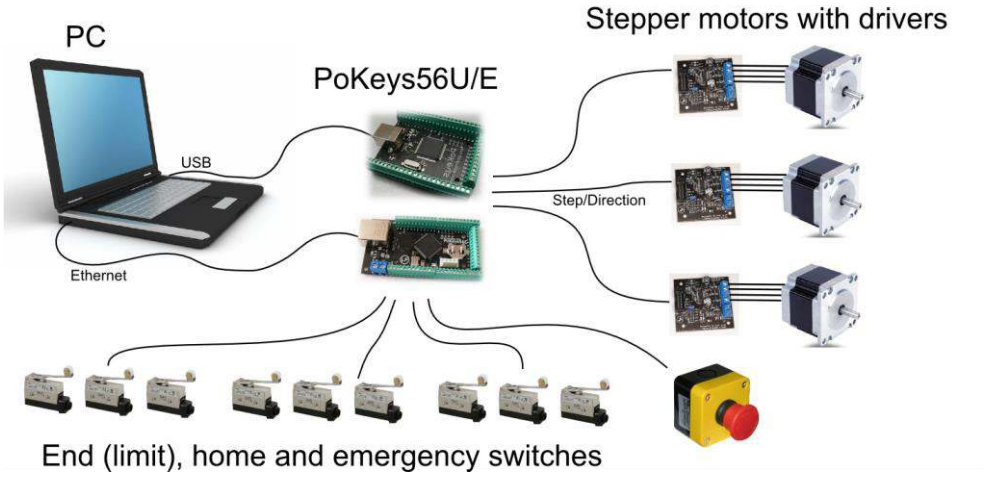


Figure 1: PoKeys Pulse engine connections for use with the integrated pulse generator

Table of contents

Pulse generator selection

Pulse engine supports different pulse generators:

- Integrated pulse generator (for up to 3 stepper motors with step frequencies up to 25 kHz)

- External pulse generator without dedicated IO capabilities (for up to 8 stepper motors with step frequencies up to 125 kHz) using PoExtensionOC16 or third-party custom board.

- External pulse generator with dedicated IO capabilities (for up to 8 stepper motors with step frequencies up to 125 kHz, dedicated limit+, limit-, home/ref, axis error inputs, 3 relay outputs, 4 open-collector outputs, 0-10 V output) using PoKeysCNCaddon (pictured below). PoKeysCNCaddon inputs and outputs are galvanically isolated from PoKeys board.

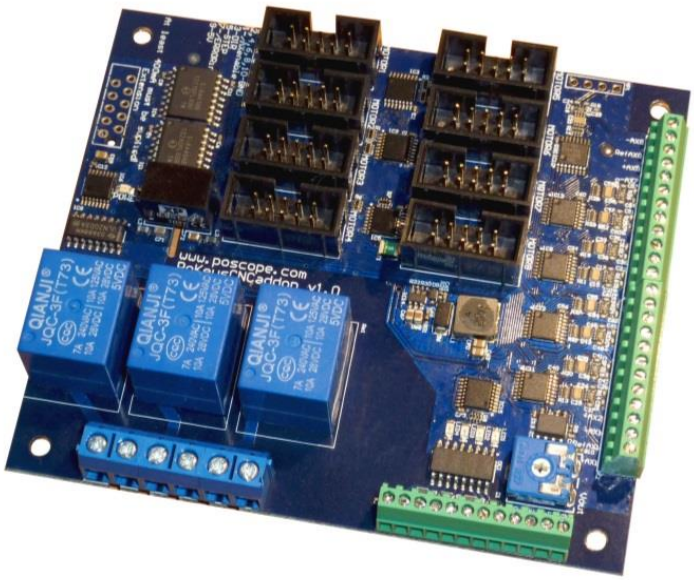


Figure 2: PoKeysCNCaddon v1.0

PoKeys device pins in use

Integrated pulse generator - up to 3 axes at 25 kHz step frequency

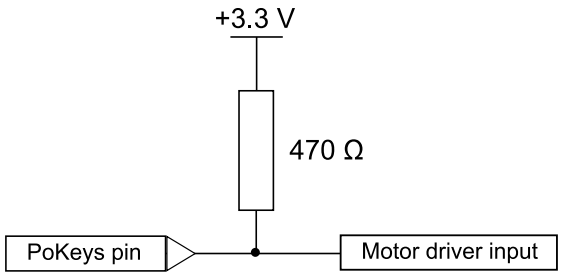
Pin	Function
38	Direction output – x
39	Direction output – y
40	Direction output – z
46 !	Step output – x
48 !	Step output – y **(external 470 Ω pull-up resistor required)**
49 !	Step output – z **(external 470 Ω pull-up resistor needed)**
52	Emergency switch input
53	Safety charge pump 5 kHz output

Inputs for limit, home and probing switches can be freely connected to any PoKeys pin and configured in software.

Remarks:

- **Watch for pin 47! It is not used for step output!**
- **All switch inputs expect normally closed (NC) switches and must be connected between specified PoKeys input pin and ground.**
- **We advise adding an additional 1 kΩ pull-up resistor on pins with an external switch**
- **Emergency switch must be connected in such way so that it cuts the power supply to the motors when the switch is activated.**

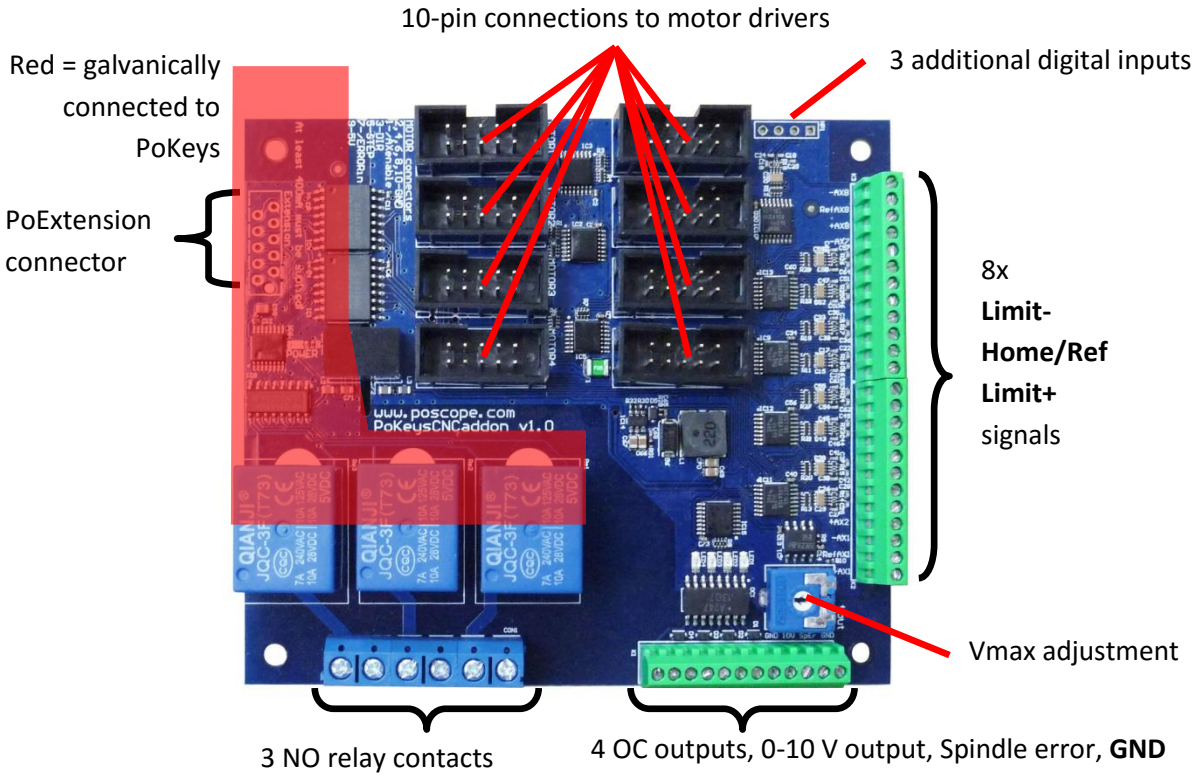
External pull-up resistor wiring for pins 48 and 49



External pulse generator with dedicated IO capability - up to 8 axes at 125 kHz step frequency[1]

Please pay attention to connecting the PoKeysCNCaddon to PoKeys device. PoKeysCNCaddon connects to PoKeys using the Expansion port flat cable, attached to the board. The Expansion port signals should be connected to PoKeys as follows:

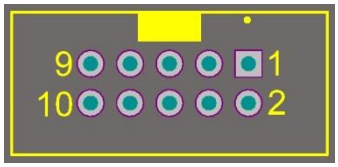
Pin	Description	PoKeys56U pin	PoKeys56E/57E pin
1 (red)	5 V power supply to the PoKeysCNCaddon - **must supply at least 400 mA for correct operation**	5 V	5 V
2	PoKeys ground ***(not to be used for PoKeysCNCaddon IO)***	GND	GND
3	PWM signal for 0-10 V output	17-22	17-22
4	Signal for IO capabilities (output)	38	38
5	Signal for IO capabilities (output)	37	37
6	Signal for IO capabilities (output)	36	36
7	Signal for IO capabilities (input)	35	35
8	Signal for pulse generation	**23**	**9**
9	Signal for pulse generation	**25**	**11**
10	Signal for pulse generation	**26**	**51**



[1] Note that PoKeys Mach3 plugin supports only 4 axes

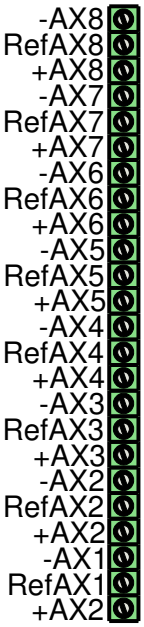
10-pin motor driver connector pinout

Pin	Function
1	Axis enable output
3	Direction output
5	Step output
7	Error input
2, 4, 6, 8, 10	GND
9	Not connected



Dedicated axis switch inputs

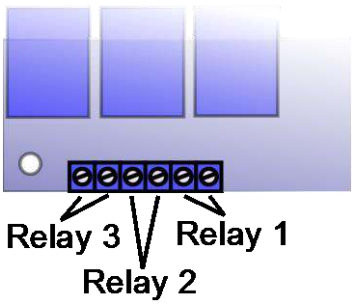
All inputs have built-in pull-up resistor - switches must be connected between **GND (on the PoKeysCNCaddon)** and the correspoding input. Select 'Dedicated pin' in the axis settings.



Pin (from top to bottom)	Function
-AX8	Limit- for axis 8
RefAX8	Ref/home for axis 8
+AX8	Limit+ for axis 8
-AX7	Limit- for axis 7
RefAX7	Ref/home for axis 7
+AX7	Limit+ for axis 7
-AX6	Limit- for axis 6
RefAX6	Ref/home for axis 6
+AX6	Limit+ for axis 6
-AX5	Limit- for axis 5
RefAX5	Ref/home for axis 5
+AX5	Limit+ for axis 5

Pin (continued)	Function
-AX4	Limit- for axis 4
RefAX4	Ref/home for axis 4
+AX4	Limit+ for axis 4
-AX3	Limit- for axis 3
RefAX3	Ref/home for axis 3
+AX3	Limit+ for axis 3
-AX2	Limit- for axis 2
RefAX2	Ref/home for axis 2
+AX2	Limit+ for axis 2
-AX1	Limit- for axis 1
RefAX1	Ref/home for axis 1
+AX1	Limit+ for axis 1

Relay outputs



PoKeysCNCaddon board features 3 relay outputs with normally-open contacts.

Rating:

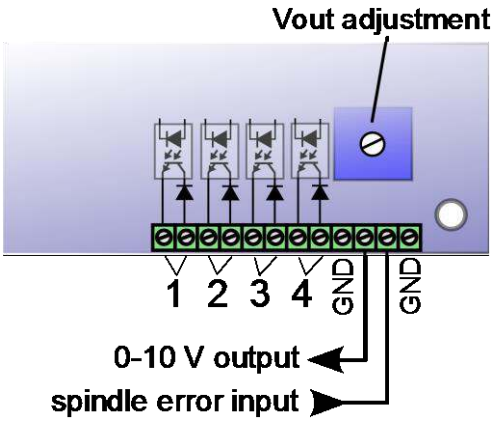
- max. 7A/240VAC, max. 10A/125VAC or max. 10A/28VDC.

Open-collector outputs

PoKeysCNCaddon board features 4 open-collector outputs with LEDs for signaling the output state.

Rating:

- Maximum applied voltage: 80 V
- Maximum DC current: up to 50 mA



0-10 V voltage output

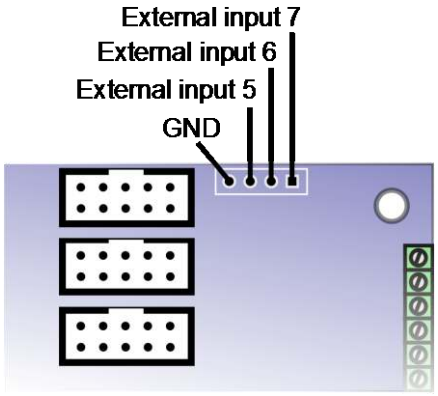
PWM signal is used to create the 0-10 V voltage output. PWM signal with 0% duty cycle produces 0 V on output, while 100% duty cycle produces Vmax on output. Vmax can be adjusted using the potentiometer 'Vout'.

In order to convert PWM signal to an analog output, a low-pass filter with the time constant of 1 ms is applied to the source signal. In order to avoid ripples in the analog output, use PWM frequency of 10 kHz or more.

Setup/calibration: either set the duty cycle to 100% or connect the PWM signal input to PoKeysCNCaddon board (pin 3) to +3.3V. Use the multimeter to measure voltage between GND adn 0-10 V output. Use the Vout adjustment potentiometer to adjust the voltage to 10 V (

Additional digital inputs

There are 4 additional digital inputs: spindle error input and 3 general purpose external digital inputs, available on the top right corner of the PoKeysCNCaddon board.



Pulse engine limitations:

- Minimum/maximum position:
  - Internal motion controller: -/+ ~16.8 million ticks
  - External (buffered) mode: -/+ ~2100 million ticks

PoKeys Mach3 plugin

Installing plugin

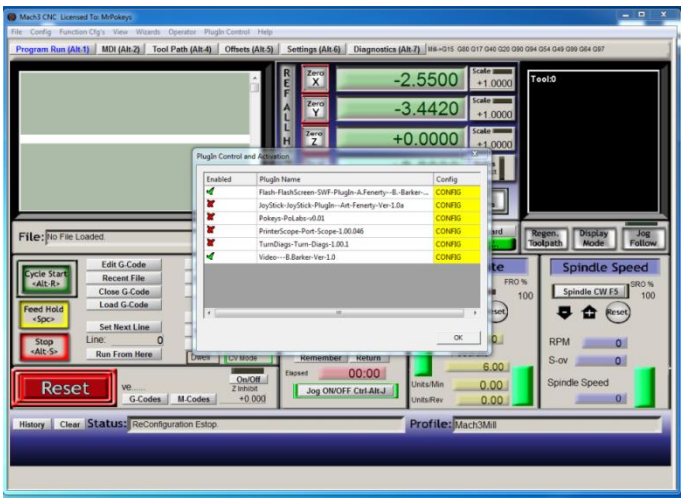
In order to install PoKeys Mach3 plugin, simply copy the Pokeys.dll to the Mach3 plugin folder (by default C:\Mach3\Plugins\). Also, install the latest PoKeys software using the provided setup file.

PoKeys Mach3 plugin existing functionality

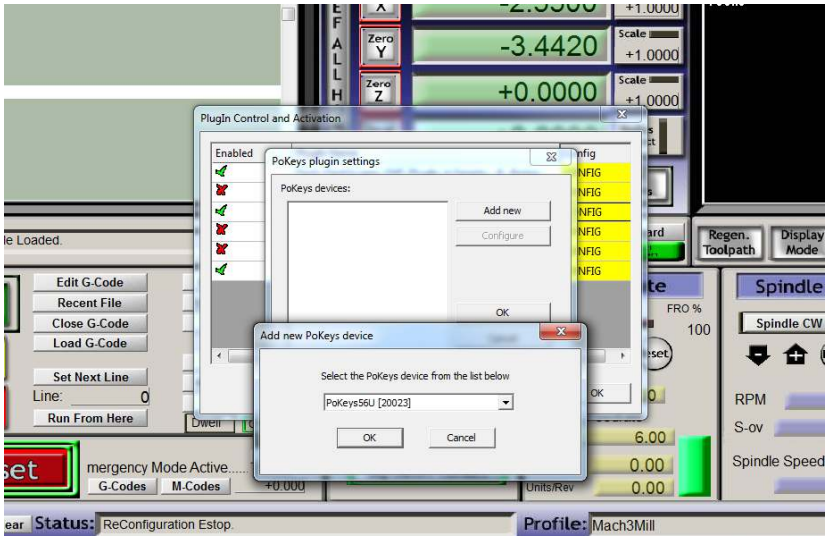
- 3-/4-axis CNC machine support
- Support for PoKeys55, PoKeys56U, PoKeys56E and PoKeys57E devices
- Mapping of PoKeys digital inputs to Mach3 OEM LEDs and OEM buttons
- Mapping of Mach3 OEM LEDs to PoKeys digital outputs
- Mapping of PoKeys encoders to Mach3 DROs
- Support for matrix keyboard
- Support for kbd48CNC keyboard on I2C address 1
- Support for PWM outputs
- Support for alphanumeric LCD display
- Support for analog inputs (analog joystick, analog to DRO mapping, offsets and gains adjustment, automatic calibration)
- Support for IO mapping (Mach3 native input-output pins, additional 100 Mach3 IO device pins - device name PoKeys_{serial number})
- Dedicated menu for each device
- Support for pendant with the activation switch
- Usage of PoKeys Pulse engine (available on PoKeys56U and Pokeys56E devices) as external motion controller for Mach3
- Safety charge pump output on pin 53
- External motion controller homing support
- Soft-limits and limit-override support
- Probing support

Configuring plugin for the first time

Open Mach3 and go to Config -> Config plugins.. The following dialog will appear.

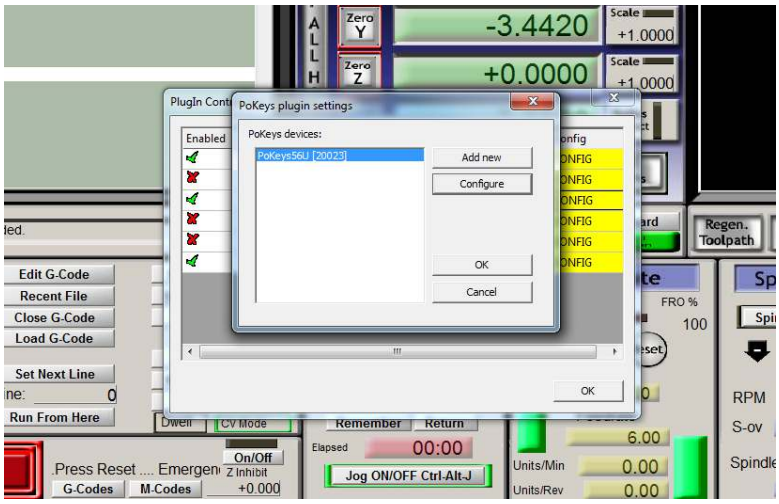


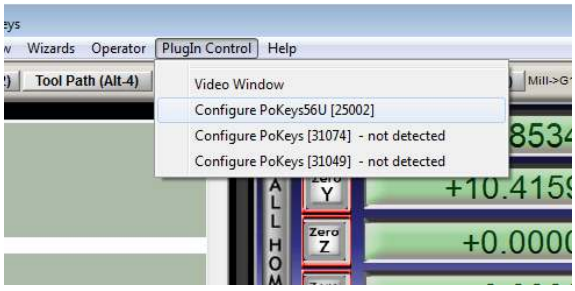
Enable the Pokeys-Polabs plugin and click CONFIG to start configuring the plugin. PoKeys plugin support multiple PoKeys devices (PoKeys55, PoKeys56U and PoKeys56E). To add a new device configuration, click the 'Add new' button and select the PoKeys device (as illustrated in the image below).



After new device configuration is added**, Mach3 MUST BE RESTARTED!**

After restart, the option 'Configure' is enabled. This opens the device configuration dialog where user can configure the device. The same can be achieved using a dedicated device menu entry in the Mach3 Plugin Control menu.





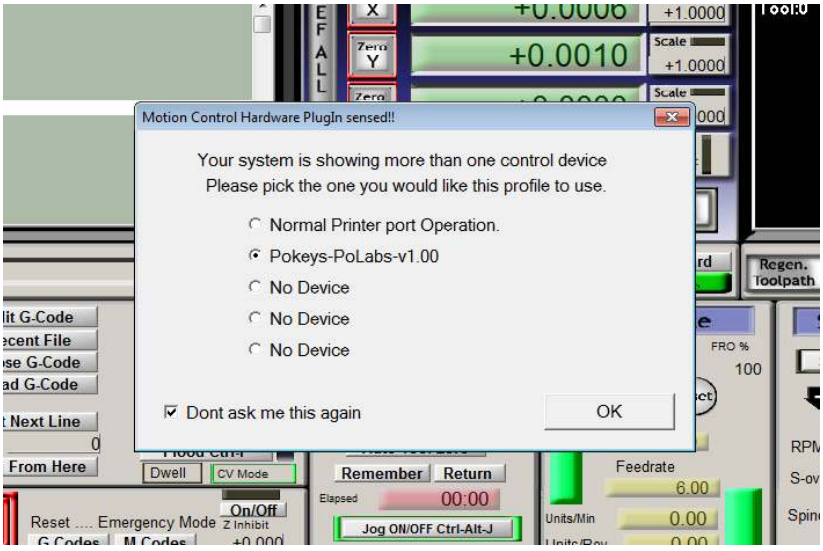
Enabling Pulse engine

In order to use Pulse engine support, go to device configuration, switch to 'Pulse engine' tab and enable one of the following options:

- Integrated 3ch: use the integrated pulse engine support in PoKeys56U and PoKeys56E. This option supports step frequencies up to 25 kHz
- External 4ch without IO: use the pulse engine with conjunction with a simple external pulse engine adapter. This option supports step frequencies up to 125 kHz.
- External 4ch with IO: use the pulse engine with PoKeysCNCaddon external boards. This option supports step frequencies up to 125 kHz.

After selecting one of the options above, click OK and restart Mach3 in order to allow Mach3 recognize an external motion controller.

On the next Mach3 startup, the following dialog will appear, notifying you that the motion control hardware plugin was detected. Select PoKeys-Polabs and click OK.

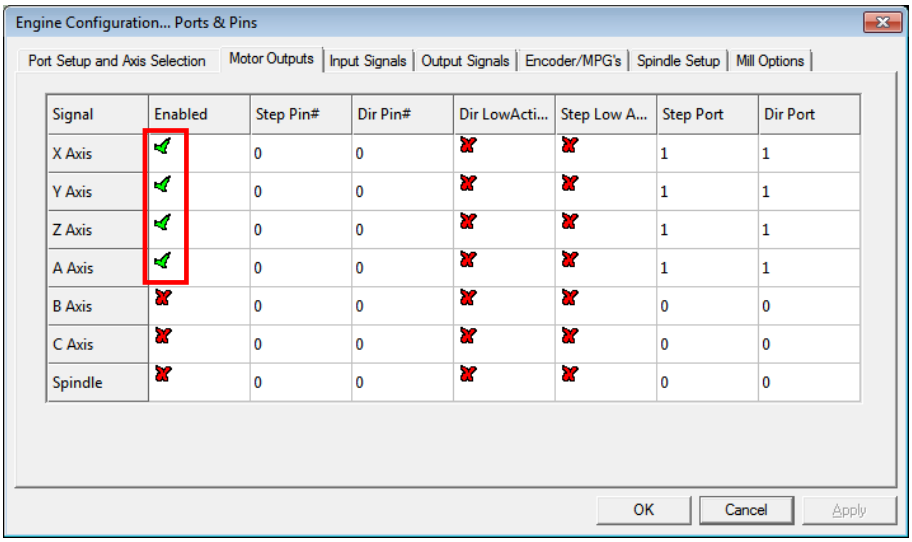


To enable Pulse engine, the emergency switch input must be connected between pin 52 and ground. The switch must be NC (normally closed) type.

At this step, configure the axes as normally through Config -> Motor tuning. See below for details.

Motors/axis setup

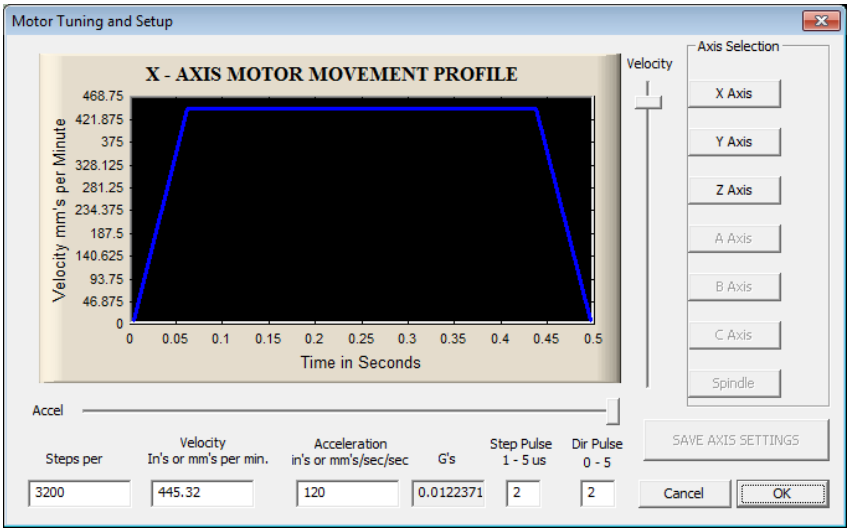
Open Config > Ports & Pins. The following dialog will appear.



Signal	Enabled	Step Pin#	Dir Pin#	Dir LowActi...	Step Low A...	Step Port	Dir Port
X Axis	✓	0	0	✗	✗	1	1
Y Axis	✓	0	0	✗	✗	1	1
Z Axis	✓	0	0	✗	✗	1	1
A Axis	✓	0	0	✗	✗	1	1
B Axis	✗	0	0	✗	✗	0	0
C Axis	✗	0	0	✗	✗	0	0
Spindle	✗	0	0	✗	✗	0	0

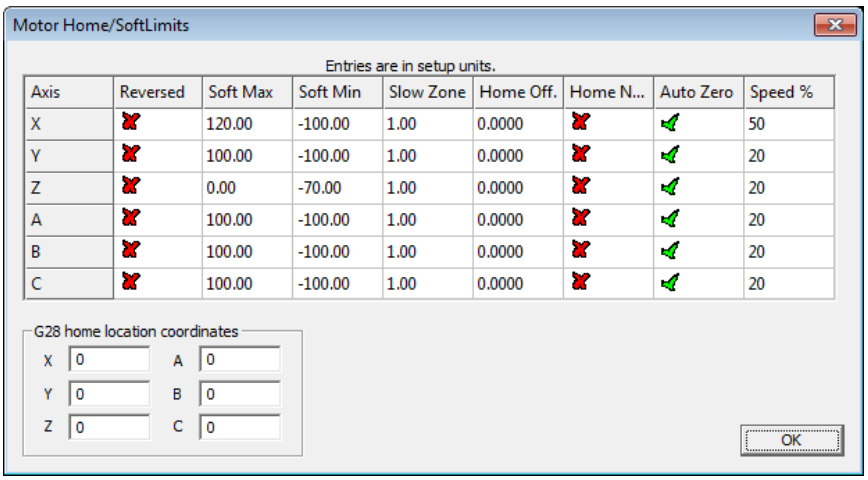
Please check that the X, Y and Z axis are enabled (enable A Axis if external pulse generator is used). Other settings are **ignored**.

After enabling the axes, open the Motor tuning dialog (Config > Motor tuning).



Follow the Mach3 motor tuning procedure to setup the appropriate values for ‘Steps per’, ‘Velocity’ and ‘Acceleration’ for each axis. The ‘Step pulse’ and ‘Dir pulse’ options are IGNORED.

To setup Home/Soft Limits, go to the menu Config > Homing/Limits. In this dialog, software limits and homing speeds can be setup. Use the 'Reversed' and 'Home neg' options to setup the axes directions. Please note that 'Slow Zone' is not supported.


Motor Home/SoftLimits
Entries are in setup units.
Axis
Reversed
Soft Max
Soft Min
Slow Zone
Home Off.
Home N...
Auto Zero
Speed %
X 120.00 -100.00 1.00 0.0000 50
Y 100.00 -100.00 1.00 0.0000 20
Z 0.00 -70.00 1.00 0.0000 20
A 100.00 -100.00 1.00 0.0000 20
B 100.00 -100.00 1.00 0.0000 20
C 100.00 -100.00 1.00 0.0000 20
G28 home location coordinates
X 0
Y 0
Z 0
A 0
B 0
C 0
OK

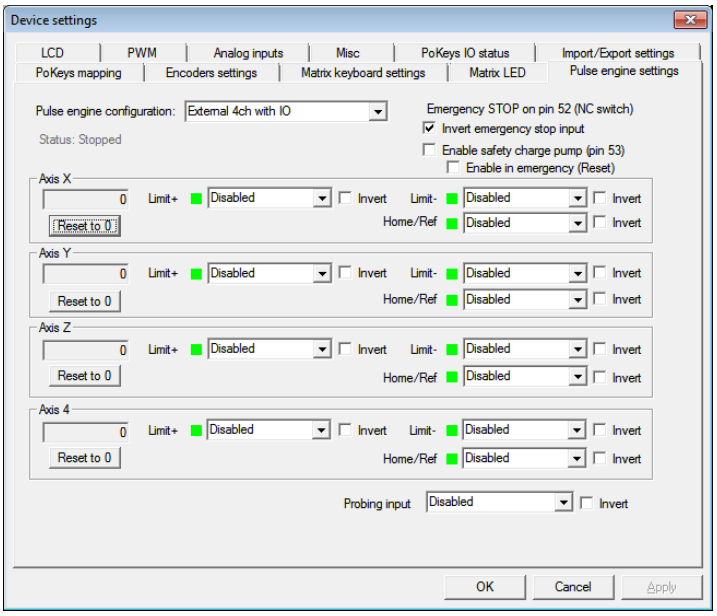
Axis switches configuration

In order to configure the axis switches, open Device configuration (either via Config Plugins or via a dedicated device configuration menu in Plugin Control main menu in Mach3) and switch to Pulse engine settings tab. The following dialog appears.

There is a separate drop-down menu for each available switch. If external pulse engine with IO functionality is selected, external dedicated option can be selected in the menu for each switch or a standard PoKeys digital input pin (the latest is the only option to use when using integrated pulse engine or external pulse engine without IO functionality).

Home/ref switch has some additional options:

- ***Shared with Limit-***: Limit- switch functions both as Limit- and as home position switch. During homing, Limit- functionality is temporarily disabled
- ***Shared with Limit+***: same as above, but with Limit+ switch



All switches can be inverted - the green/red blocks on the left of the switch selection options display the current switch status, with green indicating a free (non-tripped) switch and red indicating tripped switch. Use the 'Invert' option to switch between the states if necessary.

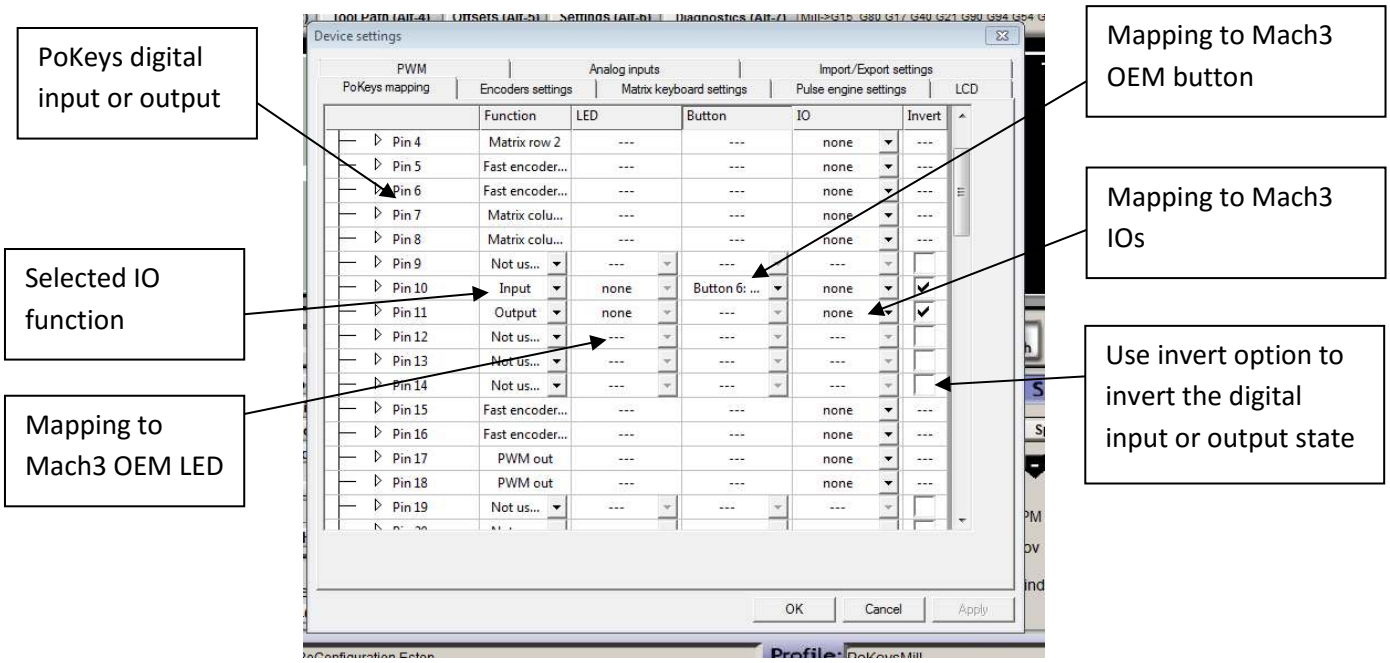
If limit switches are enabled, PoKeys Pulse engine will enter emergency mode if any limit switch gets triggered.

The probing input option is available at the bottom of the dialog and offers mapping the probing input to either external digital inputs or PoKeys digital input pins.

Setting up digital inputs and outputs mapping

To access the digital inputs and output mapping, open the menu PlugIn control > Configure PoKeys {your serial number}.

The following dialog will appear



First column displays a list of all inputs or outputs, available on your PoKeys device. Use the tree structure to navigate between different peripherals and their IO pins.

The second column displays the pin function. If the pin is assigned a special function, a description of this function will be displayed. If multiple special functions are assigned, a red 'Conflict' warning will be displayed.

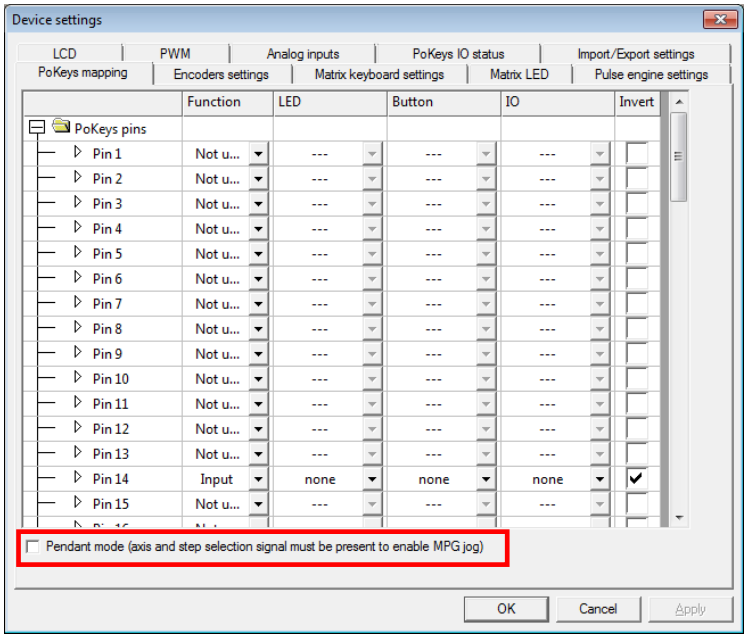
Third column (available for digital inputs and outputs) enables selection of mapping to Mach3 OEM LEDs. If the pin function is set to 'Input', this mapping will enable setting of Mach3 OEM LED state based on PoKeys IO pin state. If the pin function is set to 'Output', Mach3 OEM LED state will be reflected to PoKeys IO pin state.

Fourth column (available only for digital inputs) enables selection of IO mapping to Mach3 OEM buttons. When PoKeys IO pin is triggered, the selected Mach3 OEM button will be triggered also.

Fifth column (available for digital inputs and outputs) enables selection of mapping to/from Mach3 IOs (outputs, such as spindle relay, vacuum, … used internally by Mach to control different external devices and inputs, such as limit, home switches, … used internally by Mach to detect the status of the machine) and Mach3 IODevice inputs and outputs (accessible via VBScript).

Pendant mode

Plugin supports the usage of pendant with activation switch. If such pendant is connected to PoKeys, 'Pendant mode' should be enabled (checkbox at the bottom of the 'PoKeys mapping' dialog). In this mode, jog action will be deactivated when the activation switch is released and will be automatically activated when there is a signal detected for both the axis and step selection.



PoPendant configuration

Use the following table to configure PoPendant with Mach3. Activate 'Pendant mode' in Mapping page in order to activate the MPG jogging activation/deactivation using the 'Control switch' on the side of the PoPendant.

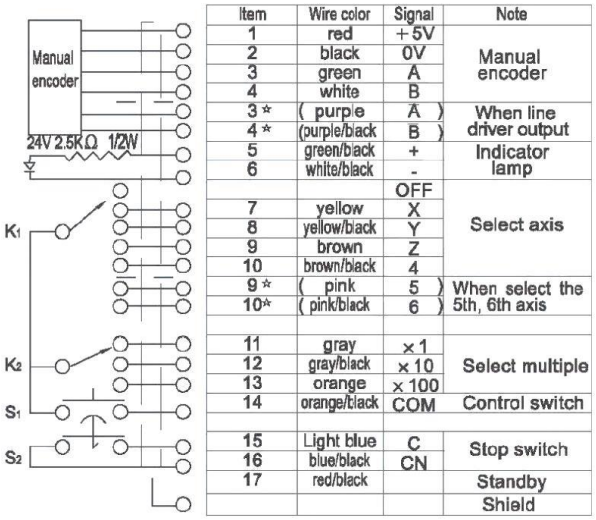
The configuration can also be downloaded (see PoPendant homepage) and imported into Mach3 (go to Import/Export tab in plugin configuration and select 'PoKeys pin mapping' and 'Encoder settings and mapping', then click on 'Import' and select the PoPendant configuration file).

Note: the following table only gives an example on how to connect the PoPendant to PoKeys device. To ease the setup process, the configuration file for this example is provided on PoPendant homepage. Wiring can be rearranged by the user, but the plugin configuration must be adjusted accordingly.

If PoKeys Pulse engine is used, 'Let PoKeys handle MPG jogging' must be checked in encoder configuration page.

Wire	PoPendant "wire colour"	Function	PoKeys pin number	Mach3 Mapping
1	red	MPG +5V	5V	/
2	black	MPG GND	GND	/
3	green	MPG A	1	MPG1 B
4	white	MPG B	2	MPG1 A
3*	purple	N.C.	*N.C.*	/
4*	purple/black	N.C.	*N.C.*	/
5	green/black	Lamp +	+3.3V	/
6	white/black	Lamp -	14	DO LED 57
7	yellow	X axis	19	Button 185
8	yellow/black	Y axis	20	Button 186
9	brown	Z axis	21	Button 187
10	brown/black	A axis	22	Button 188
9*	pink*	B axis	24	Button 189
10*	pink/black*	C axis	27	Button 190
11	gray	x1	3	Button 191
12	gray/black	x10	4	Button 192
13	orange	x100	7	Button 193
14	orange/black	Ctrl Switch	GND	/
15	Light blue	Estop	52	IO Estop**
16	blue/black	Estop GND	GND	/
17	red/black	N.C	*N.C.*	/
	shield	shield	GND	/

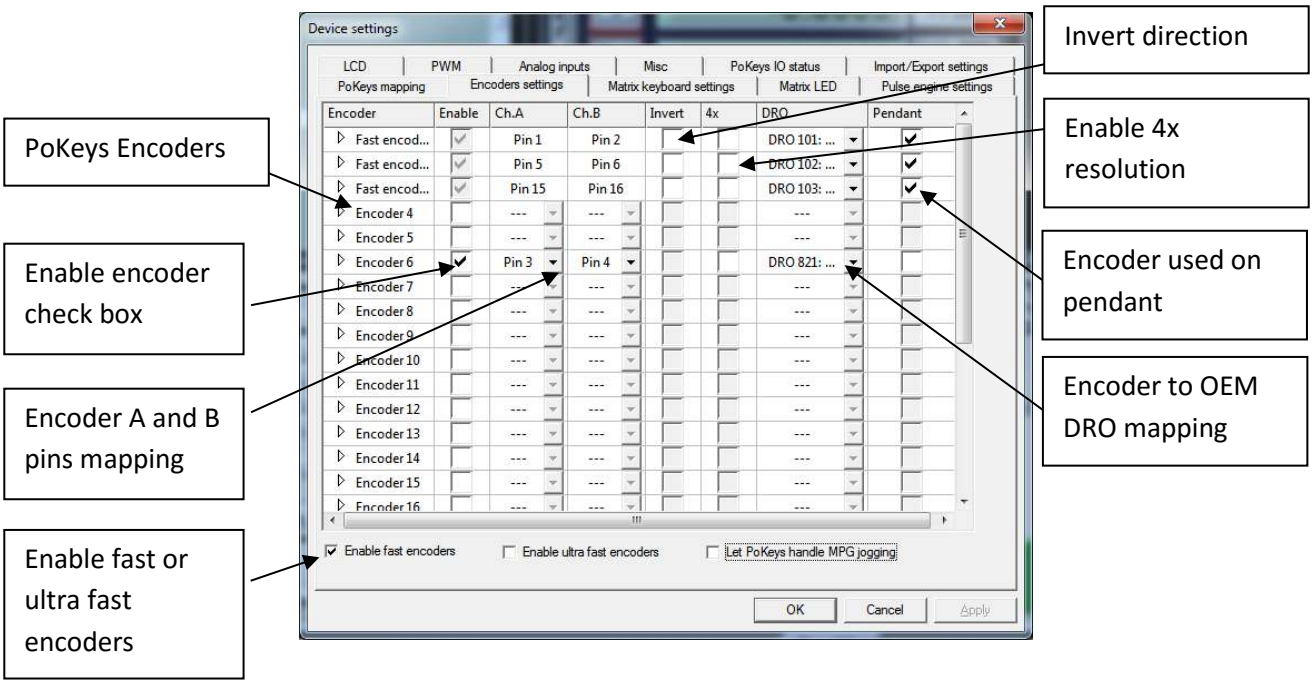
*** Not available on all units**
**** Don't configure Estop mapping if pulse engine is enabled**
N.C.= not connected
DO = digital output
DI = digital input



Item	Wire color	Signal	Note
1	red	+5V	Manual encoder
2	black	0V	
3	green	A	
4	white	B	
3 *	(purple	A)	When line driver output
4 *	(purple/black	B)	
5	green/black	+	Indicator lamp
6	white/black	-	
		OFF	Select axis
7	yellow	X	
8	yellow/black	Y	
9	brown	Z	
10	brown/black	4	
9 *	(pink	5)	When select the 5th, 6th axis
10*	(pink/black	6)	
11	gray	x 1	Select multiple
12	gray/black	x 10	
13	orange	x 100	
14	orange/black	COM	Control switch
15	Light blue	C	Stop switch
16	blue/black	CN	
17	red/black		Standby
			Shield

Figure 3: PoPendant internal wiring

Encoder (MPG) settings



The encoder page lists all supported encoders with their current settings. Fast encoders (replacing encoders 1-3) can be enabled by selecting the 'Enable fast encoders' option in the bottom part of the dialog. Ultra fast encoders (available on PoKeys56 devices) appear as encoder 26 and can also be enabled using the check box at the bottom of the window.

For other ('normal') encoders channel A and B signals must be setup. Select the appropriate PoKeys pin in each list. The selected pin will be automatically set as digital input.

To invert the encoder direction (instead of switching A and B signal connections physically) use the 'Invert' option. The check box in the sixth column enables 4x greater encoder resolution. The second to last column enables selection of encoder to Mach3 OEM DRO mapping.

The option in the last column 'Pendant' tells PoKeys plugin which encoder is used as MPG on the pendant. In case the 'Pendant mode' is enabled and there is an invalid signal from connected pendant, changes of encoders marked with 'Pendant' will have no effect on Mach3 or motion.

MPG (manual pulse generator) setup

PoKeys plugin ties itself directly into Mach3 core and does not represent a device a an LPT port-based extension. **Therefore Mach3's Ports and Pins configuration should not be used to setup the MPGs. If you configure MPG in Mach3's Ports and Pins dialog, these MPGs won't work with PoKeys.**

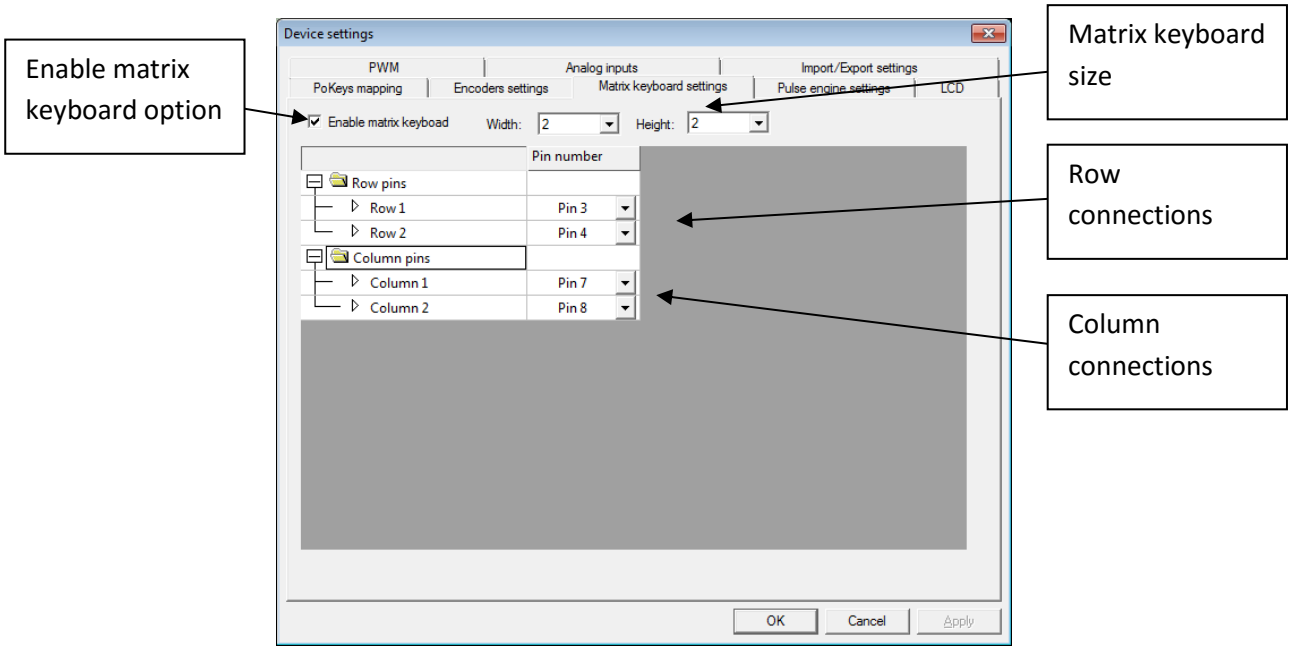
In order to setup MPG, follow the instructions above for an encoder, but select 'DRO 101 (MPG1)' (for MPG1), 'DRO 102' (for MPG2) or 'DRO 103' (for MPG3) in DRO field for that encoder. Also note, that the corresponding MPG must be only enabled in Mach3's Ports and pins under MPG tab.

If PoKeys Pulse engine is used, 'Let PoKeys handle MPG jogging' must be checked.

Matrix keyboard setup

The matrix keyboard setup gives the options to activate matrix keyboard, select its width and height and assign PoKeys pins to matrix keyboard row and column connections.

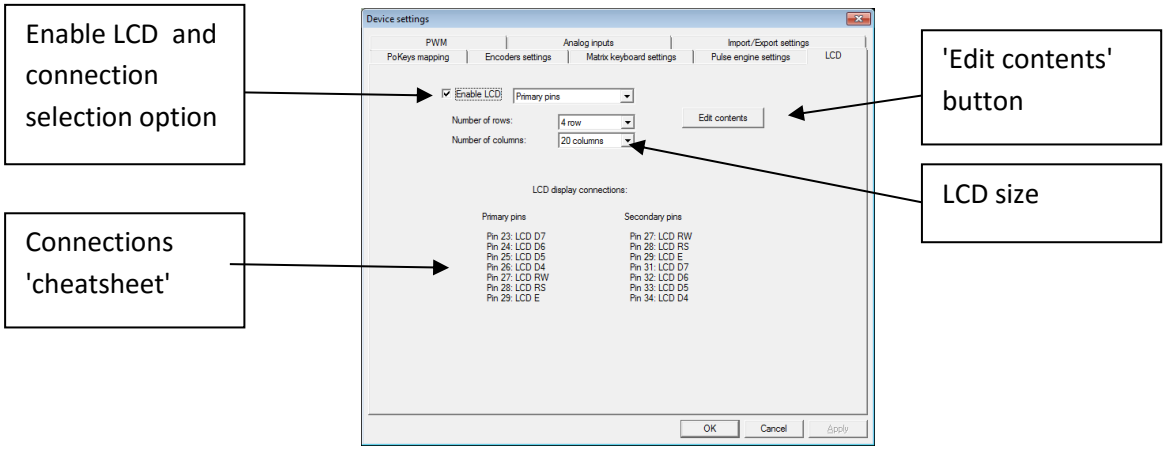
The selected pins are automatically setup as digital inputs and outputs.



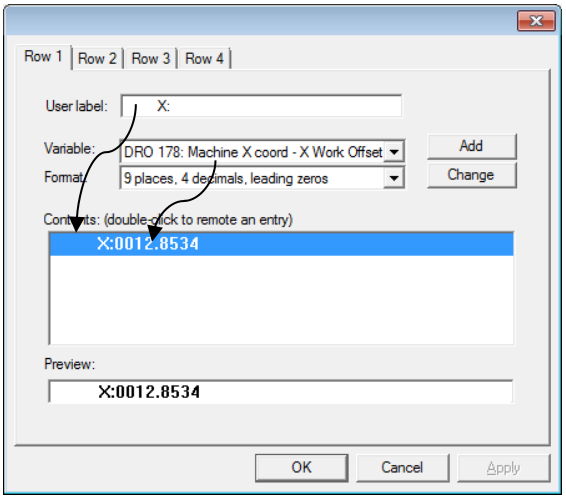
To setup mapping of matrix keyboard keys to OEM LEDs and buttons, go back to 'PoKeys mapping' tab and select appropriate functions for the matrix keyboard entries in the list of available IOs.

LCD setup

The LCD configuration dialog can be used to enable LCD, select connection option (primary or secondary pins, as defined in the PoKeys manual), select LCD size and edit contents of the LCD.



To edit the LCD contents, setup the LCD first, then click on 'Edit contents' button. The following dialog will appear

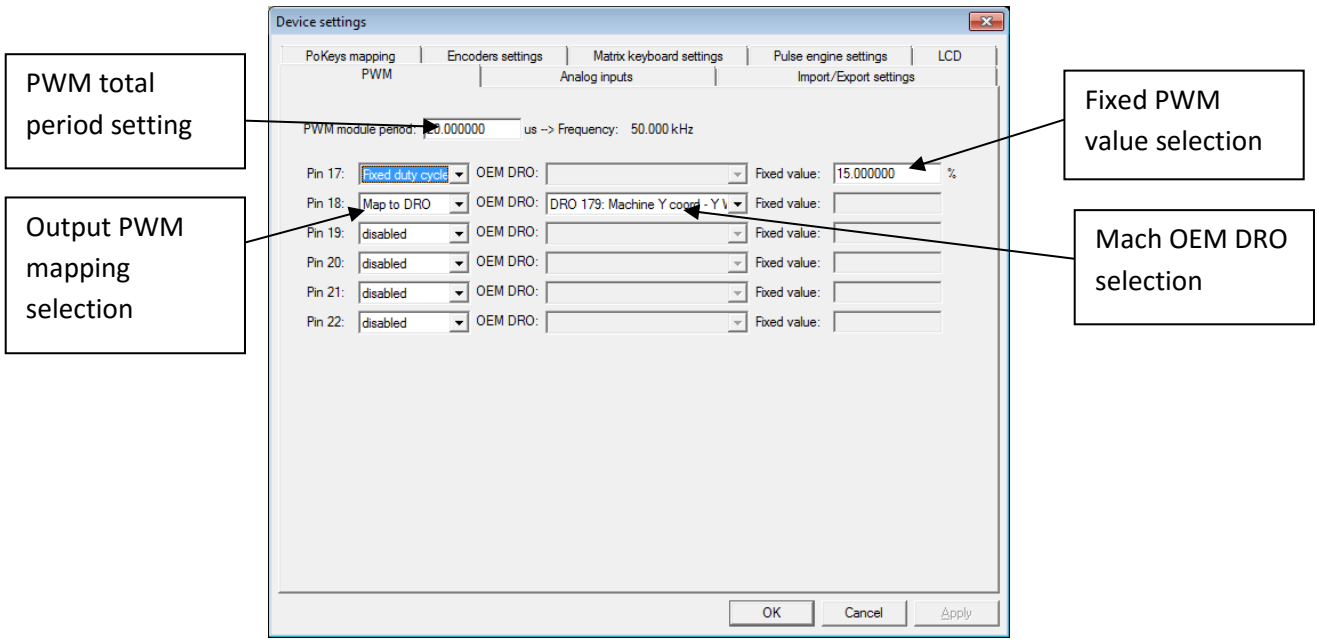


The dialog holds as many tabs as there are configured LCD lines in the previous step. The 'Row 1' dialog is used to setup only the line 1 of the LCD display ('Row 2', … are used to setup the other lines of the display).

Each row can hold multiple entries – either label only, either holding a numeric display of one of the available variables. To add a new entry, enter the 'User label' (optional), select a variable you would like to display and its display format. Then click 'Add' button. The contents list will be updated with the new entry. To remove the entry, double-click on it. Although entries are displayed in the list in the vertical manner, they are combined on the LCD horizontally as is displayed in the 'Preview' field at the bottom.

PWM (Pulse-width modulated) outputs

PoKeys devices support up to 6 pulse-width modulated digital outputs. All outputs share the same PWM total period (specified in microseconds) and have separetely configurable duty cycles. Duty cycles can be specified either in 0-100% or as raw PWM duty cycle period in microseconds.



Each PWM output can be deactivated, mapped to Mach3 OEM DRO (PWM period in microseconds or in %) or assigned a fixed value (PWM period in microseconds or in %).

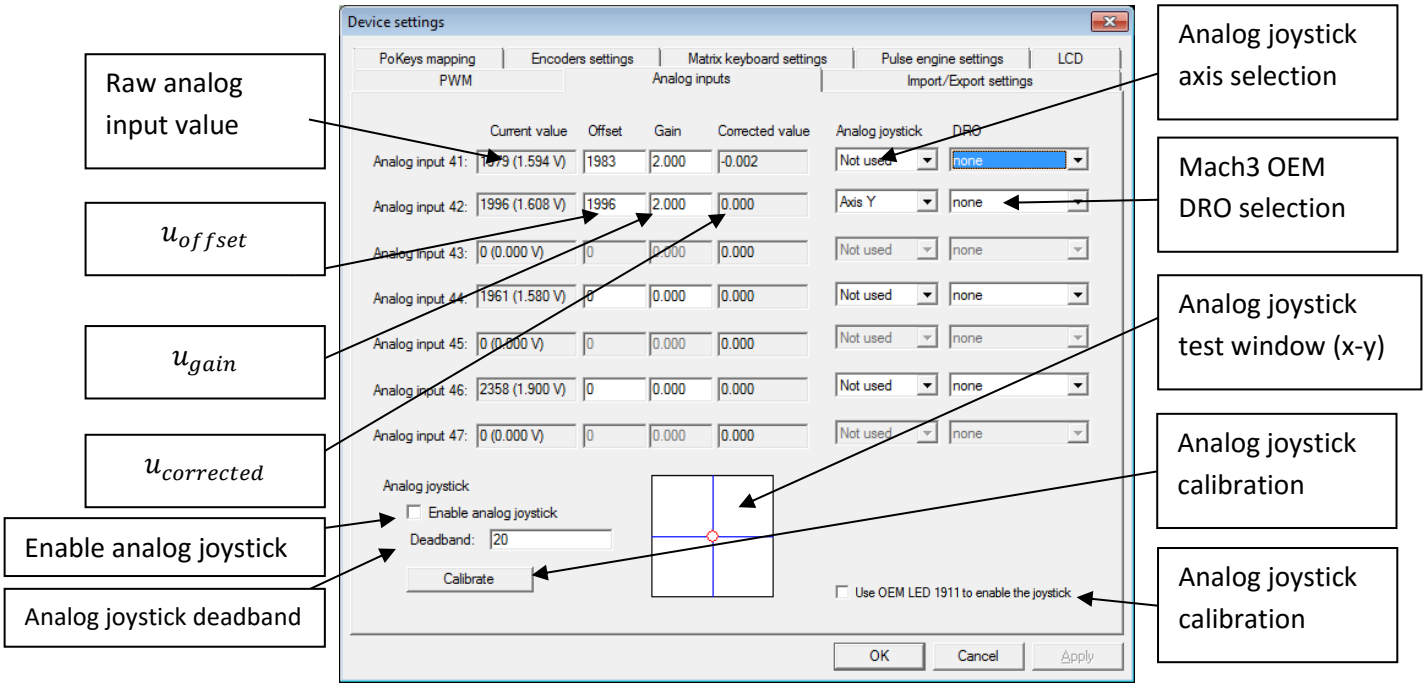
Analog inputs

Available analog inputs on the PoKeys device (pins 43-47 on PoKeys55 and pins 41-47 on PoKeys56) can be either mapped to Mach3 OEM DRO register or used as an analog joystick axis, used for jogging.

Analog inputs are displayed as 12-bit value (10-bit analog values on PoKeys55 devices are up-scaled to 12-bit) and can be configured with user specific offset and gain value using the following formula:

$$u_{corrected} = \frac{u_{AD} - u_{offset}}{4096} * u_{gain}$$

where $u_{corrected}$ is the corrected value of analog to digital readout u_{AD} using the offset u_{offset} and gain u_{gain}. The offset and gain values can be adjusted for each analog input separately. Mapping to DRO or analog joystick is done using the corrected value $u_{corrected}$.



Analog joystick functionality enables convenient jogging option. Based on the analog voltage, present on the selected pin, the selected axis can be jogged progressively. To enable analog joystick, assign the axes in the 'Analog joystick' column and click 'Calibrate' button at the bottom of the dialog. A simple wizard will walk you through the process and enable you to calibrate (automatically set the gain and offset values) based on your input. After the successfull calibration, enable analog joystick functionality by checking the box 'Enable analog joystick'.

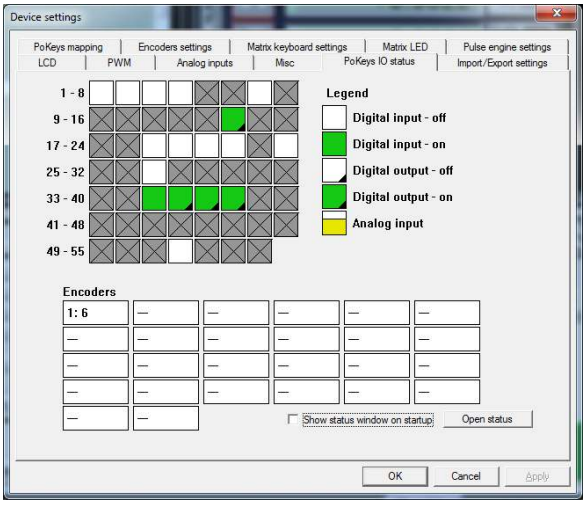
To disable unwanted jogging in zero position, adjust the parameter 'Deadband' based on the noise of your analog input (enter value in analog value ticks – 0 to 2048). If the $|u_{AD} - u_{offset}| < u_{deadband}$ the axis will not be jogged.

If 'Use OEM LED 1911 to enable the joystick' option is checked, analog joystick can be enabled and disabled using the OEM LED 1911 signal.

PoKeys IO status

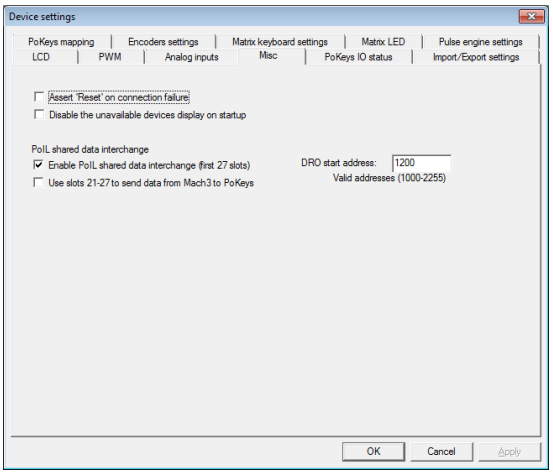
This tab gives the user an overview of the PoKeys inputs and outputs. PoKeys pins are represented as a grid of colored squares, each resembling a single PoKeys pin and encoder values are listed at the bottom of the dialog.

By clicking the 'Open status', a floating dialog is diplayed, giving the user an overview of PoKeys inputs and outputs, encoder values and PoKeys Pulse engine states even when configuration dialog is closed.



Other (miscellaneous) settings

Misc tab contains additional miscallaneous settings.



Assert 'Reset' on connection failure: if checked, Mach3 will be but into 'Reset' mode when the connection with the PoKeys device is dropped.

Disable the unavailable devices display on startup: if checked, PoKeys plugin won't display the 'Unavailable devices' window on Mach3 startup if the current device is not available

PoIL shared data interchange

This option enables interchange between PoIL shared slots (see PoBlocks manual) and Mach3 DRO registers. Once enabled, first 27 shared data slots are copied from PoKeys PoIL core to Mach3 DROs, starting by the Mach3 DRO number, specified in the field on the right (values between 1000 and 2255 are valid).

If 'Use slots 21-27 to send data from Mach3 to PoKeys' option is enabled, slots 21 to 27 are read from specified Mach3 DRO registers and sent to PoKeys PoIL core.

Custom operations can be performed on data from various PoKeys peripherals and result forwarded to Mach3 (e.g. spindle speed calculation, product counting, PID control with reference set by Mach3, …).

Reading and writing of IO from VB script

PoKeys Mach3 plugin exposes each PoKeys device (named PoKeys_{serial}, where serial is the serial number of the PoKeys device) as 100 virtual IO pins that can be accessed from Mach3 VB script with the following functions:

GetIODevName(DevID As Short) Return String

DevID - Device ID's start at zero and go up.

Return - Returns the name of the Divice as a String. If the device ID is out of range the return will be "NoDevice"

GetIODevInput(DevID As Short , IONumber As Short) Return Double

DevID - Device ID's start at zero and go up.

IONumber - The number of the IO Starting at zero (Pin number -1)

Return - Returns the value of an input OR the value that an output is set to . If the device is not found a return of 999 will be sent back.

SetIODevOutput(DevID As Short , IONumber As Short, Value As Double)

Return Short

DevID - Device ID's start at zero and go up.

IONumber - The number of the IO Starting at zero (Pin number -1)

Value - Any value to set the output to. for digital outputs 0 and 1 are used as on and off

Return - Return of 0 if there are no faults, 1 is returned if he pin is not found , 2 is returned if the pin is an output pin.

Example script (finds the PoKeys device with the serial number 25000, then toggles the IO 1 on and off at a rate of 1 Hz):

```
Sub Main()
DevName = "PoKeys_25000"
Outputnumber = 1

DevID = -1

Do
DevID = DevID+1
SearchName = GetIODevName(DevID)'Search for the Device
If(SearchName = "NoDevice") Then
  MsgBox ("Error Finding Device")
  Exit Sub
End If
Loop While (DevName <> SearchName)

For d=0 To 60'Loop 60 times to toggle the output on and off for one min
 r = SetIODevOutput(DevID,Outputnumber,1)'Activate the output
 If(r<>0) Then
   MsgBox("Output#" & Outputnumber & " " & GetIODevIOName(DevID,Outputnumber) & " is Not
an
output" )
   Exit Sub
 End If
 Sleep(500) 'Wait .5 sec
 SetIODevOutput(DevID,Outputnumber,0)'Turn the output off
 Sleep(500)'Wait .5 sec

Next d

End Sub

Main
```

Additional OEM buttons

OEM button	Function
1900	Plugin Jog Toggle
1901	Plugin Jog +
1902	Plugin Jog -
1903	Plugin Rapid Jog Toggle
1904	Plugin Goto 0's
1910	Plugin Spindle CW
1911	Plugin Spindle Stop
1912	Plugin Spindle CCW

Additional OEM LEDs

OEM LED	Function
1900	Plugin Jog X axis LED
1901	Plugin Jog Y axis LED
1902	Plugin Jog Z axis LED
1903	Plugin Jog A axis LED
1904	Plugin Jog B axis LED
1905	Plugin Jog C axis LED
1906	Plugin Jog Select 0.001 increment LED
1907	Plugin Jog Select 0.01 increment LED
1908	Plugin Jog Select 0.1 increment LED
1909	Plugin Jog Select 1 increment LED

Pulse engine v2 operating principles

PoKeys Pulse engine v2 (upgrade of the original PoKeys Pulse engine) is available on PoKeys56U and PoKeys56E devices and enables a direct control of a positioning systems that accepts step/direction signals (stepper motors, servo systems, etc.).

PoKeys Pulse engine divides the operations into 1 millisecond time slots, during which the pulse frequency is held constant, and supports the generation of up to 25 pulses per 1 millisecond time slot using integrated pulse generator or up to 125 pulses per 1 millisecond time slot using external pulse generator circuit (which equates to 25/125 kHz maximum pulse frequency supported).

At each time slot beginning, the selected limit and home switches are read and evaluated. If emergency switch or any activated limit switch (enabled in the configuration) is tripped, the pulse engine is put into Error mode and no more pulses are generated (with hard-stop mechanism). Limit switches can be disabled using the 'Limit override' function. In addition, PoKeys pulse engine also supports 'Soft-limit' function, which limits the machine motion using the virtual limit switches.

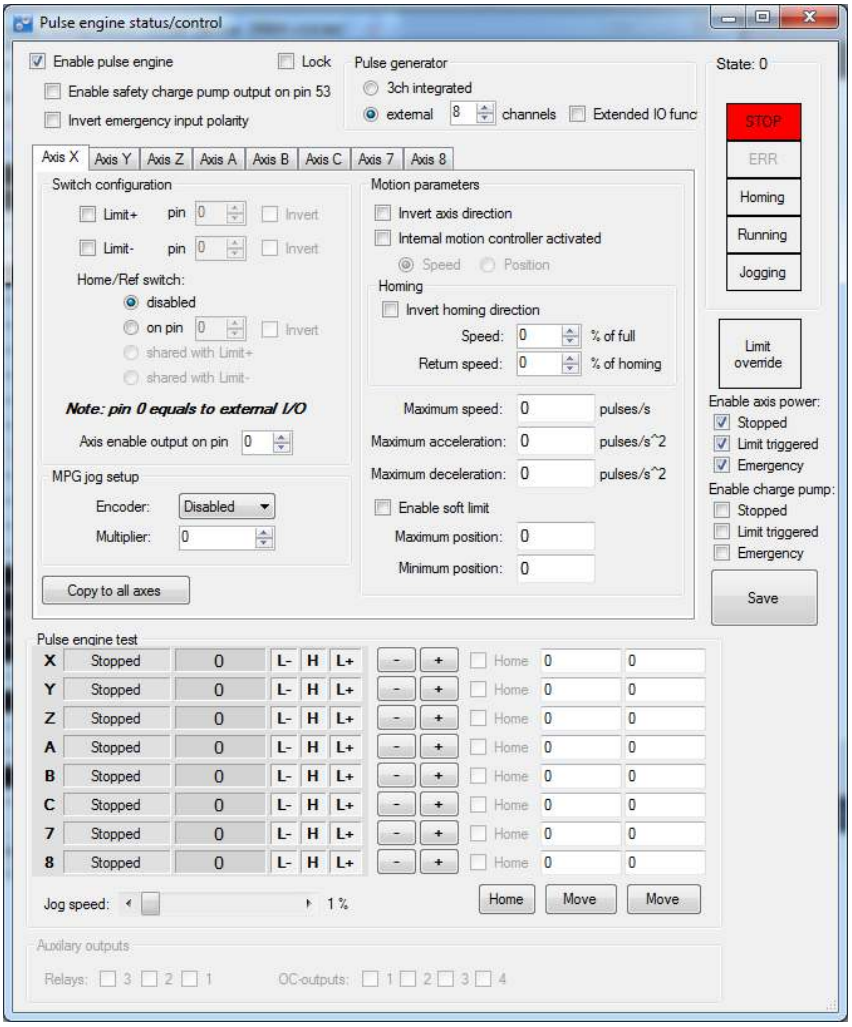


Figure 4: Pulse engine configuration window in PoKeys configuration software

PoKeys Pulse engine operates in different modes with additional modes selectable per each axis.

Modes of operation

- **Stopped**: the pulse engine does not generate any pulses.
- **Error**: the pulse engine encountered an error (e.g. limit switch was activated).
- **Homing**: homing mode is activated. In this mode, one or more axes can be homed. The selected axis (or axes) moves in negative direction at predefined fraction of the maximum speed until the home switch is tripped. Then, the direction is changed to positive and speed decreased to half the previous speed. When the switch is tripped, the internal position counter is reset and the axis is commanded to stop. This operation does not include moving back to position 0. The state of homing procedure is reflected in axes states.
- **Probing**: during probing, selected axes are actuated by PoKeys device until a probe signal has changed to a predefined state. The position of the axes is saved and the motion is stopped.
- **(MPG) Jogging**: if axis has an MPG assigned, the MPG jog is done by PoKeys device itself using the MPG multiplier value.
- **Running**: normal operation mode. In this mode, each axis can be put into either the 'buffer' mode (the internal controller is disabled and the slots are fed direclty from a slot buffer, which must be constantly filled by the external application) or into the 'internal controller mode':
    - internal position control: moves the axis to the desired position, following the limitations set by the axis parameters,
    - internal speed control: moves the axis at the desired speed, following the limitations set by the axis parameters.

Internal controller modes and buffer mode utilize separate internal buffers for operation. Hence, changing between internal or external (buffered) mode does not require clearing the motion buffers. Moreover, internal controller can be used on selected axes in parallel to the external (buffered) mode on other axes (new to Pulse engine v2).

In buffer (slave) mode, the generated motion is transferred and temporarily stored in the timeslot buffer - a 128-slots deep buffer that holds pulse frequencies for each axis, giving a 128 millisecond buffered motion period. Each slot entry in the buffer holds 16 bytes (2 byte per axis) and each axis entry uses 15-bits for pulse frequency and MSB bit for the direction signal (if MSB bit is set, the direction output is activated). Although buffer holds 16 bytes per time slot, only [number of activated axes] bytes are transferred for each slot using default 'Fill motion buffer' command.

Fill buffer command is used to transfer the data to the slot buffer. Application that uses the fill buffer command should send as much time slot data as possible. PoKeys Pulse engine will then return the number of accepted time slots. This omits the need of additional query on the buffer free space in PoKeys Pulse engine buffer. Application should only then increase the 'read pointer' based on the number of accepted slots. Additionally, fill buffer command returns a number of parameters of the pulse engine (position, engine state, state of limit and home switches and states of each axis).

In Mach3 plugin, PoKeys Pulse engine operates in buffer (slave) mode during job execution and executes the motion, generated by Mach3 motion planner. During jogging, MPG jogging, homing, probing operations, the selected axis is switched to internal motion controller, enabling real-time

responses and high accuracy of positioning. MPG jogging uses encoder (MPG) values directly to feed the internal motion controller with the up-to-date information on MPG position. This results in fast responses of the machine to the MPG input.

Safety charge-pump output

When activated, 5 kHz square safety charge-pump signal is present on pin 53 if the Pulse engine is in normal operating mode. The safety charge-pump in other operating mode can be enabled by the user by selecting the 'Enable charge-pump' check boxes.

In Mach3 plugin, the user can select whether charge pump output is active during emergency.

Motor driver enable outputs

Motor driver enable outputs are available only with conjunction with PoKeysCNCaddon. User can select modes in which the motor drivers are enabled by selecting the 'Enable axis power' check boxes.

In Mach3 plugin, the motor driver enable outputs settings are joined with safety charge-pump settings.

Axis parameters

Internal mode uses the following axis parameters of motion:

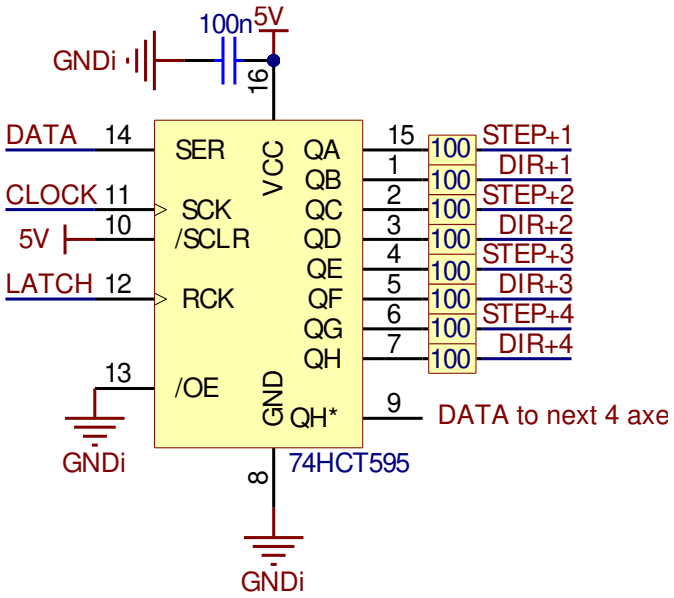
- Maximum speed: maximum frequency of pulses (in pulses/s)
- Acceleration: maximum acceleration (in pulses/s^2)
- Deceleration: maximum deceleration (in pulses/s^2)
- Limit and home switches configuration
- Direction change configuration: direction can be changed separately for each of the axes
- Homing direction configuration: direction of homing can be changed separately for each of the axes

Custom external pulse generator without IO functionality

External pulse generator is a simple circuit for deserializing step and direction data, coming from PoKeys device. The circuit uses 74HCT595 IC that is connected to PoKeys board as shown in the table below.

Two 74HCT595 can be cascaded in order to support 8 axes, but single one can be used for up to 4 axes. When cascading, CLOCK and LATCH signals are shared between both ICs, while DATA out (pin 9) of the first IC is used as DATA signal of the second IC.

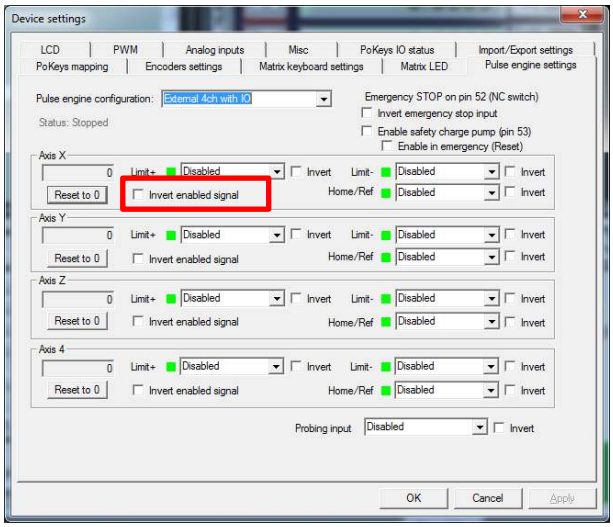
Pin	Description	PoKeys56U pin	PoKeys56E/57E pin
5V	5 V power supply	5 V	5 V
GND	PoKeys ground	GND	GND
DATA	Signal for pulse generation	**23**	**9**
CLOCK	Signal for pulse generation	**25**	**11**
LATCH	Signal for pulse generation	**26**	**51**


100n
5V
GNDi
16
DATA
14
SER
CLOCK
11
SCK
5V
10
/SCLR
LATCH
12
RCK
13
/OE
GNDi
VCC
GND
QA
QB
QC
QD
QE
QF
QG
QH
QH*
15
1
2
3
4
5
6
7
9
100
STEP+1
DIR+1
STEP+2
DIR+2
STEP+3
DIR+3
STEP+4
DIR+4
DATA to next 4 axe
74HCT595
8
GNDi

Frequently asked questions

Whenever Mach "reset" is blinking the stepper motor power is engaged, and you cannot turn the motor shaft with your fingers. When "reset" is released, the motors loose power.

There is an option 'Invert enabled signal' in the 'Pulse engine' tab of the PoKeys Mach3 plugin. There are two types of stepper drivers - ones expect active low signal for enabling the outputs, the others expect active high signal. Since PoKeys is not a LPT port extension, the Mach3's Ports & Pins is not functional and all configuration is done via PoKeys plugin configuration dialogs.



I really do not know how to set up the spindle control. Normally I would use for example the S60 (for spindle speed of 60 to produce an analog output to control the spindle). I do not know how to tell the pokeys CNC addon board how to pick up this S command.

This is achieved using the PWM output of the PoKeys board, connected to the pin 3 of the PoKeysCNCaddon board connection. Go to PoKeys Mach3 plugin configuration, tab 'PWM', select 'Map to DRO in %' and select DRO202. Then set the appropriate multiplier that the DRO value is multiplied with before sending it to the output (the output goes from 0 to 100 %, so in case you have values between 0 and 500 in this DRO, set the multiplier to 0.2). If you want to lower the output range, lower the multiplier value (digitally scale the output voltage instead of using the pot).

Similarly, spindle relay outputs can be configured in the 'PoKeys mapping' tab for one of the pins (for example, configuring one pin as Digital output and mapping it to LED 11 (Spindle ON LED).

Please read the following notes

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice.
2. PoLabs does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of PoLabs products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of PoLabs or others. PoLabs claims the copyright of, and retains the rights to, all material (software, documents, etc.) contained in this release. You may copy and distribute the entire release in its original state, but must not copy individual items within the release other than for backup purposes.
3. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of the products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. PoLabs assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
4. PoLabs has used reasonable care in preparing the information included in this document, but PoLabs does not warrant that such information is error free. PoLabs assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
5. PoLabs devices may be used in equipment that does not impose a threat to human life in case of the malfunctioning, such as: computer interfaces, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment, and industrial robots.
6. Measures such as fail-safe function and redundant design should be taken to ensure reliability and safety when PoLabs devices are used for or in connection with equipment that requires higher reliability, for example: traffic control systems, anti-disaster systems, anticrime systems, safety equipment, medical equipment not specifically designed for life support, and other similar applications.
7. PoLabs devices shall not be used for or in connection with equipment that requires an extremely high level of reliability and safety, as for example: aircraft systems, aerospace equipment, nuclear reactor control systems, medical equipment or systems for life support (e.g. artificial life support devices or systems), and any other applications or purposes that pose a direct threat to human life.
8. You should use the PoLabs products described in this document within the range specified by PoLabs, especially with respect to the maximum rating, operating supply voltage range and other product characteristics. PoLabs shall have no liability for malfunctions or damages arising out of the use of PoLabs products beyond such specified ranges.
9. Although PoLabs endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, PoLabs products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a PoLabs product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures.
10. Usage: the software in this release is for use only with PoLabs products or with data collected using PoLabs products.
11. Fitness for purpose: no two applications are the same, so PoLabs cannot guarantee that its equipment or software is suitable for a given application. It is therefore the user's responsibility to ensure that the product is suitable for the user's application.
12. Viruses: this software was continuously monitored for viruses during production, however the user is responsible for virus checking the software once it is installed.
13. Upgrades: we provide upgrades, free of charge, from our web site at www.poscope.com. We reserve the right to charge for updates or replacements sent out on physical media.
14. Please contact a PoLabs support for details as to environmental matters such as the environmental compatibility of each PoLabs product. Please use PoLabs products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. PoLabs assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
15. Please contact a PoLabs support at support@poscope.com if you have any questions regarding the information contained in this document or PoLabs products, or if you have any other inquiries.
16. The licensee agrees to allow access to this software only to persons who have been informed of and agree to abide by these conditions.
17. Trademarks: Windows is a registered trademark of Microsoft Corporation. PoKeys, PoKeys55, PoKeys56U, PoKeys56E, PoScope, PoLabs and others are internationally registered trademarks.

PoKeys protocol specification

Version: 21.12.2016

Compatible PoKeys firmware versions:
PoKeys57: **4.2.19**

Brief protocol description

USB

PoKeys USB devices are USB HID devices that use OS's integrated drivers to communicate with software. No additional drivers are neccessary to communicate with devices.

All PoKeys USB devices use these Vendor and Product IDs:

idVendor: 0x1DC3

idProduct: 0x1001

The device encapsulates three interfaces, first (index 0) being standard USB HID keyboard, second (index 1) being PoKeys communication interface and the third (index 2) standard USB HID Joystick.

Configuration is set or read using the second interface. On Windows host the PoKeys device is found by searching among connected HID devices and looking their PathNames. If the PathName contains `hid#vid_1dc3&pid_1001&mi_01`, this is the correct interface to PoKeys device. If more than one PoKeys device is connected to the same host, differentiation at this level is impossible, so user ID byte must be read from the PoKeys device. USB serial numbers can be used to differentiate between devices. PoKeys device reports serial number in the format of xxxxx.2 (where xxxxx is the serial number of the PoKeys device).

In PoKeys communication DLL, request are handled this way:

1. send report with a unique request ID (simply ever-increasing value)
2. read report
3. check Requst ID, if it does not match the one in (1), sleep for 1 ms and then go to (2) (try this 5 times, then terminate)
4. check packet checksum, if it does not match go to (1) (try this 2 times, then terminate)

PoKeys57 series firmware introduces bulk communication interface in addition to the HID interfaces mentioned above - it uses WinUSB driver on Windows (GUID of 2EA10865-4FFD-4BF3-8EF3-161549BFA270) and libusb on Linux and OS X. Optionally, HID interfaces can be disabled.

Communication

To ensure a highest possible compatibility with USB PoKeys devices, ethernet PoKeys devices use Extended packet mode as described below. Packets are transfered with TCP protocol.

Network edition

Ethernet PoKeys devices use a combination of UDP and TCP packets to communicate with the host. Both use a port number 20055.

Device discovery

PoKeys devices are discovered via broadcast UDP packets. A host sends out a UDP packet with a broadcasting address. All PoKeys devices respond with another UDP packet that contains the device's identification (User ID, serial number, version) and it's IP address. At the same time, if device is configured to use DHCP server and no DHCP server responded to the request, PoKeys will use the temporary address of x.x.x.250, where x.x.x is the subnet address of the computer that the request was sent from (with 255.255.255.0 subnet mask).

Packet data is formated the following way:

Host -> device: empty packet (ignored)

Device -> host

- byte 1: User ID number
- bytes 2,3: Serial number
- bytes 4,5: Version
- bytes 6,7,8,9: IP address

Communication

All further communication with the device is acomplished with TCP or UDP connection on port 20055. Packet structure is 64-bytes long and is described below as Extended packet mode.

Security

Ethernet PoKeys devices support additional security option that requires the user to enter the password before the access to the device is granted. The password can contain any character and can be up to 32 characters long.

Connection timeout

After 3 seconds of inactivity (or otherwise specified in the configuration), ethernet PoKeys device terminates the TCP connection with the host.

Packet formatting

Incoming and outgoing packets are 64 bytes long. Basic packets use only first 8 bytes. Extended packets use the whole packet.

Host>Device
- byte 1: control 0xBB
- byte 2: operation
- byte 3-6: operation parameters
- byte 7: request ID
- byte 8: control byte (sum after mod 0x100)

Device>Host
- byte 1: control 0xAA
- byte 2: operation
- byte 3-6: operation parameters
- byte 7: request ID
- byte 8: control byte (sum after mod 0x100)

Extended packet mode (supported since version 1.8)
Packet size is increased to 64 bytes. First 8 bytes remain the same, additional 56 bytes are used for extended mode. Reserved bytes should be set to 0.

PoKeys devices

PoKeys devices are described using three descriptors:

- PoKeys device series
- PoKeys device hardware version
- PoKeys device firmware type

Overview

Series ID	Hardware ID	Hardware version	Firmware type IDs supported
0	0	unsupported device type or old firmware	0 (default)
55	1,2,3	PoKeys55	0 (default)
56	10	PoKeys56U	0 (default) 1 (PoTLog27U)
56	11	PoKeys56E	0 (default) 1 (PoTLog27E)
57	28	PoKeys57U	0 (default)
57	29	PoKeys57E	0 (default)
57	30	PoKeys57Uv1.1	0 (default)
57	31	PoKeys57Ev1.1	0 (default)
57	32	PoKeys57CNC	0 (default)
57	35	PoKeys57U OEM	0 (default)
57	36	PoKeys57E OEM	0 (default)
57	37	PoPLC57NG	0 (default)
57	38	PoKeys57CNCdb25	0 (default)
57	39	PoKeys57Utest	0 (default)
58	40	PoKeys58EU	0 (default)
58	41	PoBootload (series 58)	0 (default)
58	50	PoPLC v1.0	0 (default)
16	60	PoKeys16	0 (default)

Device series

Series 55

Series 55 contains PoKeys55 device only. The support for this series is limited.

Device	HW ID	Device features
PoKeys55	0	USB keyboard/joystick, 55x I/O, 5x 10-bit analog input, 1x 10-bit analog output, 6x PWM, 25x encoders, 3x fast encoders, 2x Matrix LED, 16x8 Matrix keyboard, LCD, PoExtBus

Series 56

Series 56 contains PoKeys56U and PoKeys56E devices

Device	HW ID	Device features
PoKeys56U	10	USB keyboard/joystick, 55x I/O, 24x digital counter, 7x 12-bit analog input, 6x PWM, 25x encoders, 3x fast encoders, 1x ultra-fast encoder, 2x Matrix LED, 16x8 Matrix keyboard, LCD, PoExtBus , PoNET, I^2C and 1-wire buses support, PoIL core (4k program memory, 1k data memory)
PoKeys56E	11	Ethernet connectivity, web interface, Modbus TCP, server reports, 55x I/O, 24x digital counter, 7x 12-bit analog input, 6x PWM, 25x encoders, 3x fast encoders, 1x ultra-fast encoder, 2x Matrix LED, 16x8 Matrix keyboard, LCD, PoExtBus , PoNET, I^2C and 1-wire buses support, PoIL core (4k program memory, 1k data memory)

Series 57

Series 57 contains multiple PoKeys devices

Device	HW ID	Device features
PoKeys57E	31	Ethernet connectivity, web interface, Modbus TCP, server reports, 55x I/O, 24x digital counter, 7x 12-bit analog input, 6x PWM, 25x encoders, 3x fast encoders, 1x ultra-fast encoder, 2x Matrix LED, 16x8 Matrix keyboard, LCD, PoExtBus , PoNET, I^2C and 1-wire buses support, PoIL core (32k program memory, 4k data memory)
PoKeys57U	28	USB connectivity, USB keyboard/joystick, 55x I/O, 24x digital counter, 7x 12-bit analog input, 6x PWM, 25x encoders, 3x fast encoders, 1x ultra-fast encoder, 2x Matrix LED, 16x8 Matrix keyboard, LCD, PoExtBus , PoNET, I^2C and 1-wire buses support, PoIL core (32k program memory, 4k data memory)
PoKeys57Uv1.1	30	Ethernet connectivity, web interface, Modbus TCP, server reports, 55x I/O, 24x digital counter, 7x 12-bit analog input, 6x PWM, 25x encoders, 3x fast encoders, 1x ultra-fast encoder, 2x Matrix LED, 16x8 Matrix keyboard, LCD, PoExtBus , PoNET, I^2C and 1-wire buses support, PoIL core (32k program memory, 4k data memory)
PoKeys57Ev1.1	31	USB connectivity, USB keyboard/joystick, 55x I/O, 24x digital counter, 7x 12-bit analog input, 6x PWM, 25x encoders, 3x fast encoders, 1x ultra-fast encoder, 2x Matrix LED, 16x8 Matrix keyboard, LCD, PoExtBus , PoNET, I^2C and 1-wire buses support, PoIL core (32k program memory, 4k data memory)
PoKeys57CNC	32	TBD
PoKeys57U OEM	35	TBD
PoKeys57E OEM	36	TBD
PoPLC57NG	37	TBD
PoKeys57CNCdb25	38	TBD
PoKeys57Utest	39	TBD

Series 58

Series 58 contains PoPLC v1.0 and PoKeys58EU

TBD

Supported commands:

Supported functions

0x00 - Read serial number, version
0x02 - Set User ID
0x03 - Read User ID and lock setting
0x04 - Read build date
0x05 - Get system load status
0x06 - Read device name
0x07 - Enable/disable fast USB interface
0x08 - Delayed startup setup
0x09 - PPM decoder configuration and status
0x0A - Session settings
0x0B - Misc USB device configuration
0x10 - Set pin function
0x11 - Set encoder settings
0x12 - Set encoder key mapping for direction A
0x13 - Set encoder key mapping for direction B
0x15 - Get pin function
0x16 - Get encoder settings
0x17 - Get encoder key mapping for direction A
0x18 - Get encoder key mapping for direction B
0x19 - Get encoder RAW value
0x1A - Reset encoder RAW value
0x1B - Get/Set Connection signal pin status
0x1C - Enable/Disable Ultra fast encoder input
0x1D - Reset digital counters values
0x1E - Additional pin settings
0x1F - Get pin capabilities
0x20 - Set key association
0x25 - Get key association
0x30 - Get input
0x31 - Block get input I
0x32 - Block get input II
0x35 - Get analog input
0x36 - Block get analog (4x 8bit)
0x37 - Block get analog (3x 10bit)
0x38 - Get analog RC filter value
0x39 - Set analog RC filter value
0x3A - Get all analog inputs (7x 12bit on PoKeys56) – extended mode
0x40 - Set output
0x41 - Set analog output
0x42 - Block set output I
0x43 - Block set output II
0x4A - MCS system
0x50 - Save configuration
0x51 - Save and lock configuration
0x52 - Disable lock and reset configuration
0x60 - Get joystick configuration
0x61 - Get joystick up Event buttons configuration
0x65 - Set joystick configuration

0x66 - Set joystick up Event buttons configuration
0x6A - Set joystick analog to digital mapping

0x70 - Setup sensors (PoKeys series 56)
0x71 - Setup dashboard items
0x72 - Setup web users
0x73 - Setup web settings
0x74 - Read all sensors
0x75 - Reserved
0x76 - Setup sensors (PoKeys series 57)
0x77 - Read sensor values (PoKeys series 57)
0x78 - Setup dashboard items (PoKeys series 57)
0x80 - Pulse engine commands - deprecated since 3.1.0
0x81 - Failsafe settings
0x82 - PoIL commands
0x83 - RTC settings
0x84 - System log operations
0x85 - Pulse engine commands - v2
0x86 - CAN operations
0x8F - Unlock option

0x90 - Create macro
0x91 - Modify macro
0x92 - Delete macro
0x93 - Save macros to flash
0x94 - Rename macro
0x95 - Transfer macro
0x96 - Get macro length
0x97 - Get macro name
0x98 - Get macro keys
0x99 - Get free space
0x9A - Get active macros

0xB0 - Multi-part packet

0xC0 - Pin configuration
0xC1 - Pin key mapping
0xC2 - Pin key codes
0xC3 - Pin key modifiers
0x21 - Pin typematic delay
0x22 - Pin repeat rate
0xC4 - Encoder option
0xC5 - Encoder channel A + B
0xC6 - Encoder channel A key code + modifier
0xC7 - Encoder channel B key code + modifier
0xCD - Get encoder long RAW values
0xCE – Enable fast encoders for pins 1-6
0xC8 - Get macro name and length
0xC9 - Get macro keys

0xCA - Matrix keyboard configuration
0xCB - PWM configuration

0xCC - Get device status (IO, analog, encoders)
0xCD - Get 32-bit RAW encoder data
0xCE - Enable/disable Fast encoders
0xCF – Get tick counter
0xD0 – LCD configuration
0xD1 – LCD operation
0xD5 – Matrix LED configuration
0xD6 – Matrix LED update
0xD7 - Triggered input settings
0xD8 - Digital counters values
0xD9 – (Set/Get) Digital counters direction pins
0xDA – set auxilary bus settings
0xDB – I^2C settings and communication
0xDC – 1-wire settings and communication
0xDD – PoI2C communication
0xDE - UART communication
0xE0 – Get/set network settings
0xE1 – Get security setting status (and password hash seed)
0xE2 – Authorise user
0xE3 – Set user password
0xE4 – Get/Set Modbus settings
0xE5 - SPI communication
0xEF – Get/set Cosm settings

Real-time mode
0xA0 – Setup RTmode
0xA1 – RTmode packet – in/out

Bootloader operations

0xF0 – clear application memory
0xF1 – block transfer options
0xF2 – transfer block part
0xF3 – start application
0xF5 – calculate and save CRC
0xF6 – clear user settings

General

Read device data

- byte 2: 0x00
- byte 3-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x00
- byte 3: serial MSB
- byte 4: serial LSB
- byte 5: software version (v(1+[4-7]).([0-3]))
- byte 6: revision number
- byte 7: request ID

Additional info (newer devices):

- bytes 9-12: string 'PK58' or 'PKEx' (preferred)
- bytes 13-16: 32-bit serial number
- byte 17: firmware version
- byte 18: firmware revision version
- byte 19: HW ID (see table in PoKeys devices chapter)
- byte 20: user ID
- bytes 21-31: build date string
- bytes 32-41: device name string
- byte 42: firmware type ID (0: default firmware)
- bytes 43-44: firmware version of the application (applicable to bootloader)
- bytes 45-57: reserved
- byte 58: product ID offset
- byte 59: configuration lock status
- byte 60: configuration firmware version
- byte 61: configuration firmware revision number
- byte 62: bootloader flag (1 indicates bootloader)

Set user ID

- byte 2: 0x02
- byte 3: ID
- byte 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x02
- byte 3: confirmed ID
- byte 4-6: 0
- byte 7: request ID

Read user ID (deprecated, use »Read device data« instead)

- byte 2: 0x03
- byte 3-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x03
- byte 3: userID
- byte 4: device lock status (if 1, device configuration is locked)
- byte 5-6: 0
- byte 7: request ID

Read/set device name

- byte 2: 0x06
- byte 3:
  - 0 for reading device name
  - Bit 0: for writing device name
  - Bit 1: for reading joystick device name
  - Bit 2: read product ID offset
  - Bit 3: set product ID offset
- byte 4: use long device name (1)
- byte 5-6: 0
- byte 7: request ID
- bytes 9-18: device name string
- bytes 19-34: joystick device name string
- byte 35: product ID offset
- bytes 36-55: long device name string
- bytes 56-63: reserved

Returned packet:

- byte 2: 0x0x6
- byte 3-6: reserved
- byte 7: request ID
- bytes 9-18: device name string
- bytes 19-34: new joystick device name string
- byte 35: product ID offset
- bytes 36-55: long device name string
- bytes 56-63: reserved

Read build date (deprecated, use »Read device data« instead)

- byte 2: 0x04
- byte 3: part (0-2), extended info (10)
- byte 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x04
- byte 3-6: char 1-4, 5-8, 9-11
- byte 7: request ID

Extended info
Returned packet:

- byte 2: 0x04
- byte 3: hardware version
- byte 4: hardware revision
- byte 5: hardware architecture
- byte 6: reserved
- byte 7: request ID

Configuration saving

- byte 2: 0x50
- byte 3: 0xAA
- byte 4: 0x55
- byte 5-6: 0
- byte 7: request ID

Configuration saving and lock

- byte 2: 0x51
- byte 3: 0xAA
- byte 4: 0x55
- byte 5-6: 0
- byte 7: request ID

Configuration reset

- byte 2: 0x52
- byte 3: 0xAA
- byte 4: 0x55
- byte 5-6: 0
- byte 7: request ID

Get system load status

- byte 2: 0x05
- byte 3-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x05
- byte 3: system load (in %)
- byte 4-6: reserved
- byte 7: request ID

Get tick counter

- byte 2: 0xCF
- byte 3-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0xCF
- byte 3-6: 32-bit tick counter with ms resolution (LSB first)
- byte 7: request ID

Read/activate option

- byte 2: 0x8F
- byte 3: 1 for writing, 0 for reading (set to 0xFF to clear all options)
- byte 4-6: 0
- byte 7: request ID
- byte 9-16: option activation code
- bytes 17-63: reserved

Returned packet:

- byte 2: 0x8F
- byte 3-6: reserved
- byte 7: request ID
- byte 9: activations OK (bit mapped)
- bytes 10: reserved

Configure USB interface

- byte 2: 0x07
- byte 3: bit-mapped USB interface configuration (set to 10 to read the status only)
    * bit 0: fast USB interface enable status
    * bit 5: HID communication interface disable
    * bit 6: keyboard interface disable
    * bit 7: joystick interface disable
- byte 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x07
- byte 3: USB interface configuration - see bits above
- byte 4-6: reserved
- byte 7: request ID

Delayed startup configuration

In order to address the issue with selected systems where PoKeys device stops device from booting-up, delayed startup of the PoKeys device can be configured.

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x08**
3	Command - 0 for reading the configuration, 1 for writing
4	Startup delay in x100 ms (0 to 25.4 seconds delay available), a value of 0 or 0xFF results in no delay at startup
5-6	reserved
7	Request ID
8 (CRC)	CRC

Delayed startup configuration - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x08**
3	Command
4	Startup delay
5-6	Reserved
7	Request ID
8 (CRC)	CRC

Session settings

These commands are used for managing sessions with PoKeys device

0x0A Session settings

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x0A**
3	0x00 – read session data, 0x10 - set session data
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-18	session data (if byte 3 is set to 0x10)
19-63	Reserved
64 (CRC 2)	CRC

Session settings - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x0A**
3-6	Reserved
7	Request ID
8 (CRC)	CRC
9-18	session data
19-63	Reserved
64 (CRC 2)	CRC

Misc USB device configuration

0x0B Misc USB device configuration

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x0B**
3	0x00 – read configuration, 0x10 - set configuration
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Interrupt communication interval in ms
10-63	Reserved
64 (CRC 2)	CRC

Session settings - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x0B**
3-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Interrupt communication interval in ms
10-63	Reserved
64 (CRC 2)	CRC

General pin settings

PoKeys55, PoKeys56, PoKeys57 limitations

1. ***Pin codes used in PoKeys55 device are 0-based, e.g. pin 1 has pin code of 0, pin 55 has pin code of 54.***
2. ***Analog input capable pins 43 to 47 have pin codes of 42 to 46.***
3. ***Analog output capable pin 43 has pin code of 42.***
4. ***PWM (pulse-width modulation) capable pins 17 to 22 have pin codes of 16-21 (PWM module outputs are in reversed order, e.g. pin 17 (pin coded as 16) is connected to PWM6 output – see specifications below).***

Set input/output settings

- byte 2: 0x10
- byte 3: pin ID (0-54)
- byte 4: pin settings
  - bit 0: special pin function
  - bit 1: digital input
  - bit 2: digital output
  - bit 3: analog input
  - bit 4: analog output
  - bit 5: triggered input
  - bit 6: digital counter input
  - bit 7: invert state
- byte 5: other / digital counter options (PoKeys56E)
  - bit 0: count rising edges (fast counter)
  - bit 1: count falling edges (fast counter)
  - bit 2: disable 10k pull-up resistor (PoPLC)
  - bit 3: enable external analog conversion (PoPLC)
  - bits 4- 5: analog conversion resolution: 0 = 12-bit, 1 = 14-bit, 2 = 16-bit, 3 = 18-bit (PoPLC)
  - bits 6-7: analog pin mode: 0 = normal analog input, 1 = Pt1000 analog input, 2 = 0-10 V analog input, 3 = 4-20 mA analog input (PoPLC)
- byte 6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0x10
- byte 3: 0 - OK, 1 – Pin number out of range or configuration locked
- byte 4-6: 0
- byte 7: request ID

Read input/output settings

- byte 2: 0x15
- byte 3: pin ID (0-54)
- byte 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x15
- byte 3: pin ID (0-54) (or 0xFF if pin number out of range or configuration locked)
- byte 4: pin settings
  bit 0: reserved
  bit 1: digital input
  bit 2: digital output
  bit 3: analog input
  bit 4: analog output
  bit 5: triggered input
  bit 6: digital counter input
  bit 7: invert state
- byte 5: reserved / digital counter options - see packet description 0x10 above
- byte 6: extended pin function
- byte 7: request ID

Input/output settings - extended mode

	option 1	option 2
Read all pin functions	0	0
Set all pin functions	1	0
Read all additional settings	0	1
Set all additional settings	0	2

- byte 2: 0xC0
- byte 3: option 1 (see table above)
- byte 4: option 2 (see table above)
- byte 5-6: 0
- byte 7: request ID

if (option 1 > 0)

- byte 9-63: pin settings set (see above for value descriptions)

if (option 1 == 0 and option 2 == 2)

- byte 9-63: pin digital counter options (if set/read additional settings enabled)

Returned packet:

- byte 2: 0xC0
- byte 3: additional settings
- byte 4-6: reserved
- byte 7: request ID

- byte 9-63: pin settings (option 1 == 0 and option 2 == 0)
- byte 9-63: pin digital counter options (option 1 == 0 and option 2 == 1)

Additional settings

- byte 2: 0x1E
- byte 3: 0 for configuration read, 1 for write
- byte 4: auto-initialize outputs on device startup
- byte 5: set digital outputs startup values (1)
- byte 6: reserved (0)
- byte 7: request ID

- byte 9-15: digital outputs startup values

Returned packet:

- byte 2: 0x1E
- byte 3: auto-initialize outputs status (1 if enabled)
- byte 4: 1 if digital outputs startup values are provided in bytes 9-15
- byte 5-6: reserved
- byte 7: request ID

- bytes 9-15: digital outputs startup values

Get pin capabilities

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x1F**
3	0x00 - select pin range (13 pins per read)
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	CRC

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x1F**
3	Reserved
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-62	Pin capabilites (bit-masked) - up to 13 pins per range
63	Reserved
64 (CRC 2)	CRC

Pin basic and extended capabilites bit masks

```
typedef enum
{
```

```
        PK_PinCap_digitalInput      = 2,        // Digital input
        PK_PinCap_digitalOutput     = 4,        // Digital output
        PK_PinCap_analogInput       = 8,        // Analog input (only on selected pins)
        PK_PinCap_analogOutput      = 16,       // Analog output (only on selected pins)
        PK_PinCap_triggeredInput    = 32,       // Triggered input
        PK_PinCap_digitalCounter    = 64        // Digital counter (only on selected pins)
} ePinCapabilities;

typedef enum
{
        PK_PinExtCap_1wire          = (1<<8),
        PK_PinExtCap_Pt1000         = (1<<9),
        PK_PinExtCap_0_10V          = (1<<10),
        PK_PinExtCap_4_20mA         = (1<<11),
        PK_PinExtCap_FastEnc        = (1<<12),
        PK_PinExtCap_PWM            = (1<<13)
} ePinExtendedCapabilities;
```

Key association setting (for pins which are set to 1, 2 or 32)

- byte 2: 0x20
- byte 3: pin ID (0-54)
- byte 4: key modifier (Ctrl/Alt/Shift)
    - bit 0: ctrl
    - bit 1: shift
    - bit 2: Alt
    - bit 3: Windows
    - bit 4: reserved
    - bit 5: reserved
    - bit 6: Alt Gr
    - bit 7: reserved
- byte 5: key KeyCode
- byte 6: option
    - bit 0: enable key mapping
    - bit 1: direct key mapping
    - bit 2: key mapped to macro (KeyCode is macro ID)
    - bit 3: key mapped to continous macro (same as above, but macro is refiring if input is still active)
    - bit 4: key repeating (after a delay, the key is being repeatedly fired with a given rate)
    - bit 5:
    - bit 6:
    - bit 7:
- byte 7: request ID

Returned packet:

- byte 2: 0x20
- byte 3: 0 - OK, 1 if pin number out of range or configuration locked, 2 if pin not set as digital input
- byte 4-6: 0
- byte 7: request ID

Reading of key associations

- byte 2: 0x25
- byte 3: pin ID (0-54)

- byte 4-6: 0
- byte 7: request ID

Returned packet:
- byte 2: 0x25
- byte 3: key modifier
- byte 4: key code
- byte 5: option (see command 0x20)
- byte 6: 0
- byte 7: request ID

Typematic delay setup

- byte 2: 0x21
- byte 3: option
- byte 4-6: 0
- byte 7: request ID

if (option > 0)

- byte 9-63: pin typematic delay set (set in steps of 5 ms – 0 to 1275 ms possible)

Returned packet:
- byte 2: 0x21
- byte 3-6: reserved
- byte 7: request ID

- byte 9-63: pin typematic delay get (see above for value descriptions)

Key repeat rate setup

- byte 2: 0x22
- byte 3: option
- byte 4-6: 0
- byte 7: request ID

if (option > 0)

- byte 9-63: pin key repeat rate set (repeat period in 5 ms cycles (plus 1 cycle) – set to 9 (9x 5ms + 1x 5ms = 50ms) to get 20 key presses per second, set to 199 to get 1 key press per second)

Returned packet:
- byte 2: 0x22
- byte 3-6: reserved
- byte 7: request ID

- byte 9-63: pin key repeat rate get (see above for value descriptions)

Key mappings

- byte 2: 0xC1
- byte 3: option
- byte 4-6: 0
- byte 7: request ID

if (option > 0)

- byte 9-63: pin key mapping set (see above 'option' for value descriptions)

Returned packet:

- byte 2: 0xC1
- byte 3-6: reserved
- byte 7: request ID

- byte 9-63: pin key mapping get (see above 'option' for value descriptions)

Key codes

- byte 2: 0xC2
- byte 3: option
- byte 4-6: 0
- byte 7: request ID

if (option > 0)

- byte 9-63: pin USB key code set

Returned packet:

- byte 2: 0xC2
- byte 3-6: reserved
- byte 7: request ID

- byte 9-63: pin key code get

Key modifiers

- byte 2: 0xC3
- byte 3: option
- byte 4-6: 0
- byte 7: request ID

if (option > 0)

- byte 9-63: pin key modifiers set (see above for value descriptions)

Returned packet:

- byte 2: 0xC3
- byte 3-6: reserved
- byte 7: request ID

- byte 9-63: pin key modifiers get (see above for value descriptions)

Triggered input key mappings

- byte 2: 0xD7
- byte 3: option
- byte 4-6: 0
- byte 7: request ID

Option = 1
- byte 9-63: down key codes for each pin
Option = 2
- byte 9-63: down key modifiers for each pin
Option = 3
- byte 9-63: up key codes for each pin
Option = 4
- byte 9-63: up key modifiers for each pin
Option = 11..14
- byte 9-63: reserved
Option = 20
- byte 9: triggered key length write
Option = 21: read triggered key length

Returned packet:

- byte 2: 0xD7
- byte 3-6: reserved
- byte 7: request ID

Option = 11
- byte 9-63: down key codes for each pin
Option = 12
- byte 9-63: down key modifiers for each pin
Option = 13
- byte 9-63: up key codes for each pin
Option = 14
- byte 9-63: up key modifiers for each pin
Option = 21
- byte 9: triggered key length

Get/Set Connection signal pin status

Connection signal pin status can be set for pins 48 to 55. When USB connection with PC is established, pin for which ' Connection signal pin status' is set to 1, will go to into high state (if pin is not inverted). After connection with PC is lost and power through USB is still available, pin will go into low state (if pin is not inverted).
To set the value, set option byte to 1. To read the value, set the option byte to 0.
- byte 2: 0x1B
- byte 3: option
- byte 4: bit mapped Connection signal pin status
- byte 5-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x1B
- byte 3: reserved
- byte 4: bit mapped Connection signal pin status
- byte 5-6: reserved
- byte 7: request ID

Encoder settings

PoKeys supports up to 25 'normal' encoders, 3 fast encoders (mapped to encoders 0, 1 and 2).
An ultra fast encoder input (mapped to index 25) is supported on PoKeys56E.
Encoder settings

- byte 2: 0x11
- byte 3: encoder ID (0-25)
- byte 4: option
    - bit 0: enable encoder
    - bit 1: 4x sampling
    - bit 2: 2x sampling
    - bit 3: reserved
    - bit 4: direct key mapping for direction A
    - bit 5: mapped to macro for direction A
    - bit 6: direct key mapping for direction B
    - bit 7: mapped to macro for direction B
- byte 5: channel A input
- byte 6: channel B input
- byte 7: request ID

Returned packet:

- byte 2: 0x11
- byte 3: 0 - OK, 1 if encoder ID out of range or configuration locked
- byte 4-6: 0
- byte 7: request ID

Encoder key mapping for direction A

- byte 2: 0x12
- byte 3: encoder ID (0-25)
- byte 4: reserved
- byte 5: key code or macro ID
- byte 6: key modifier
- byte 7: request ID

Returned packet:

- byte 2: 0x12
- byte 3: 0 - OK, 1 if encoder ID out of range or configuration locked
- byte 4-6: 0
- byte 7: request ID

Encoder key mapping for direction B

- byte 2: 0x13
- byte 3: encoder ID (0-25)
- byte 4: reserved

- byte 5: key code or macro ID
- byte 6: key modifier
- byte 7: request ID

Returned packet:

- byte 2: 0x13
- byte 3: 0 - OK, 1 if encoder ID out of range or configuration locked
- byte 4-6: 0
- byte 7: request ID

Read encoder settings

- byte 2: 0x16
- byte 3: encoder (0-25)
- byte 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x16
- byte 3: encoder (0-25)
- byte 4: option
- byte 5: channel A pin
- byte 6: channel B pin
- byte 7: request ID

Read encoder key mapping for direction A

- byte 2: 0x17
- byte 3: encoder (0-25)
- byte 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x17
- byte 3: encoder (0-25)
- byte 4: reserved
- byte 5: key code or macro ID
- byte 6: key modifier
- byte 7: request ID

Read encoder key mapping for direction B

- byte 2: 0x18
- byte 3: encoder (0-25)
- byte 4-6: 0

- byte 7: request ID

Returned packet:

- byte 2: 0x16
- byte 3: encoder (0-25)
- byte 4: reserved
- byte 5: key code or macro ID
- byte 6: key modifier
- byte 7: request ID

Read encoder RAW value

- byte 2: 0x19
- byte 3: encoder ID
- byte 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x19
- byte 3: encoder (0-25)
- byte 4: RAW value
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

Reset encoder RAW value

- byte 2: 0x1A
- byte 3: encoder ID
- byte 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x1A
- byte 3: encoder (0-25)
- byte 4: reserved
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

Encoder option

- byte 2: 0xC4
- byte 3: option
- byte 4-6: 0

- byte 7: request ID

if (option > 0)

- bytes 9-33: encoder option set (see above for 'option' values)

Returned packet:

- byte 2: 0xC4
- byte 3-6: reserved
- byte 7: request ID

- bytes 9-33: encoder option get (see above for 'option' values)

Encoder channel A and B pin

- byte 2: 0xC5
- byte 3: option
- byte 4-6: 0
- byte 7: request ID

if (option > 0)

- bytes 9-33: encoder channel A pin set
- bytes 34-58: encoder channel B pin set

Returned packet:

- byte 2: 0xC5
- byte 3-6: reserved
- byte 7: request ID

- bytes 9-33: encoder channel A pin get
- bytes 34-58: encoder channel B pin get

Encoder channel A key code and modifier

- byte 2: 0xC6
- byte 3: option
- byte 4-6: 0
- byte 7: request ID

if (option > 0)

- bytes 9-33: encoder channel A key codes set
- bytes 34-58: encoder channel A key modifiers set

Returned packet:

- byte 2: 0xC6
- byte 3-6: reserved
- byte 7: request ID

- bytes 9-33: encoder channel A key codes get
- bytes 34-58: encoder channel A key modifiers get

Encoder channel B key code and modifier

- byte 2: 0xC7
- byte 3: option
- byte 4-6: 0
- byte 7: request ID

if (option > 0)

- bytes 9-33: encoder channel B key codes set
- bytes 34-58: encoder channel B key modifiers set

Returned packet:

- byte 2: 0xC7
- byte 3-6: reserved
- byte 7: request ID

- bytes 9-33: encoder channel B key codes get
- bytes 34-58: encoder channel B key modifiers get

Get encoder long RAW values

- byte 2: 0xCD
- byte 3: option
- byte 4-6: 0
- byte 7: request ID
- option byte:
  - 0 – get encoder RAW values for encoders 1-13
  - 1 – get encoder RAW values for encoders 14-26
  - 10 – set encoder RAW values for encoders 1-13
  - 11 – set encoder RAW values for encoders 14-26

If option == 10

- bytes 9-12: encoder 1 RAW value (LSB first)
- bytes 13-16: encoder 2 RAW value (LSB first)
- bytes 17-20: encoder 3 RAW value (LSB first)
- bytes 21-24: encoder 4 RAW value (LSB first)
- bytes 25-28: encoder 5 RAW value (LSB first)

- bytes 29-32: encoder 6 RAW value (LSB first)
- bytes 33-36: encoder 7 RAW value (LSB first)
- bytes 37-40: encoder 8 RAW value (LSB first)
- bytes 41-44: encoder 9 RAW value (LSB first)
- bytes 45-48: encoder 10 RAW value (LSB first)
- bytes 49-52: encoder 11 RAW value (LSB first)
- bytes 53-56: encoder 12 RAW value (LSB first)
- bytes 57-60: encoder 13 RAW value (LSB first)

- bytes 61-63: reserved

If option == 11

- bytes 9-12: encoder 14 RAW value (LSB first)
- bytes 13-16: encoder 15 RAW value (LSB first)
- bytes 17-20: encoder 16 RAW value (LSB first)
- bytes 21-24: encoder 17 RAW value (LSB first)
- bytes 25-28: encoder 18 RAW value (LSB first)
- bytes 29-32: encoder 19 RAW value (LSB first)
- bytes 33-36: encoder 20 RAW value (LSB first)
- bytes 37-40: encoder 21 RAW value (LSB first)
- bytes 41-44: encoder 22 RAW value (LSB first)
- bytes 45-48: encoder 23 RAW value (LSB first)
- bytes 49-52: encoder 24 RAW value (LSB first)
- bytes 53-56: encoder 25 RAW value (LSB first)
- bytes 57-60: Ultra fast encoder RAW value (LSB first)

- bytes 61-63: reserved

Returned packet:

- byte 2: 0xCD
- byte 3-6: reserved
- byte 7: request ID

If option == 0

- bytes 9-12: encoder 1 RAW value (LSB first)
- bytes 13-16: encoder 2 RAW value (LSB first)
- bytes 17-20: encoder 3 RAW value (LSB first)
- bytes 21-24: encoder 4 RAW value (LSB first)
- bytes 25-28: encoder 5 RAW value (LSB first)
- bytes 29-32: encoder 6 RAW value (LSB first)
- bytes 33-36: encoder 7 RAW value (LSB first)
- bytes 37-40: encoder 8 RAW value (LSB first)
- bytes 41-44: encoder 9 RAW value (LSB first)
- bytes 45-48: encoder 10 RAW value (LSB first)
- bytes 49-52: encoder 11 RAW value (LSB first)

- bytes 53-56: encoder 12 RAW value (LSB first)
- bytes 57-60: encoder 13 RAW value (LSB first)

- bytes 61-63: reserved

If option == 1

- bytes 9-12: encoder 14 RAW value (LSB first)
- bytes 13-16: encoder 15 RAW value (LSB first)
- bytes 17-20: encoder 16 RAW value (LSB first)
- bytes 21-24: encoder 17 RAW value (LSB first)
- bytes 25-28: encoder 18 RAW value (LSB first)
- bytes 29-32: encoder 19 RAW value (LSB first)
- bytes 33-36: encoder 20 RAW value (LSB first)
- bytes 37-40: encoder 21 RAW value (LSB first)
- bytes 41-44: encoder 22 RAW value (LSB first)
- bytes 45-48: encoder 23 RAW value (LSB first)
- bytes 49-52: encoder 24 RAW value (LSB first)
- bytes 53-56: encoder 25 RAW value (LSB first)
- bytes 57-60: Ultra fast encoder RAW value (LSB first)

- bytes 61-63: reserved

Enable/disable fast encoders on pins 1-2, 3-4 / 5-6 and 15-16

There are two different fast encoders configurations. On newer PoKeys56 devices, only second configuration can be selected.
Configuration 1: pins 1-2 as encoder 1, pins 3-4 as encoder 2, pins 15-16 as encoder 3
Configuration 2: pins 1-2 as encoder 1, **pins 5-6 as encoder 2**, pins 15-16 as encoder 3

- byte 2: 0xCE
- byte 3: option

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Encoder 3 invert	Encoder 2 invert	Encoder 1 invert	Disable 4x sampling	configuration[1]			

- byte 4: (bit 0): Enable index signal[2]
- bytes 5-6: reserved
- byte 7: request ID

Returned packet:
- byte 2: 0xCD
- byte 3: status

[1] Set to 1 to enable fast encoders with configuration 1, set to 10 to enable configuration 2, set to 2 to read setup
[2] When set to 1, the pins 9, 11 and 27 function as index signal inputs. If a positive front on these pins is detected, the encoder counter is reset accordingly.

- byte 4: enable index signal
- bytes 5-6: reserved
- byte 7: request ID
- bytes 9-63: reserved

Enable/disable ultra fast encoder

Ultra fast encoder input is supported only on PoKeys56E.
Pins used are:
Pin 8: Phase A input
Pin 12: Phase B input
Pin 13: Index input

- byte 2: 0x1C
- byte 3: enable ultra fast encoder (set to 1 to enable, set to 0 to disable, set to 0xFF to read configuration)
- byte 4: additional options

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
reserved					Enable 4x sampling	Signal mode	Invert direction

Signal mode: when = 0, A and B function as quadrature encoder inputs. When = 1, A functions as the direction signal and B functions as the clock signal.
Enable 4x sampling: when = 0, only A edges are counted (2X). When = 1, BOTH A and B edges are counted (4X), increasing resolution but decreasing range.

- byte 5: if 1, encoder will be reset on next index signal (new in firmware 4.2.19)
- byte 6: 0
- byte 7: request ID
- bytes 9-12: digital filter sampling delay used in ultra fast encoder

Returned packet:

- byte 2: 0x1C
- byte 3: status
- byte 4: additional options as above
- bytes 5-6: reserved
- byte 7: request ID
- bytes 9-12: digital filter sampling delay used in ultra fast encoder
- byte 13: enable encoders status (copy of byte 3)
- byte 14: encoders options (copy of byte 4)
- bytes 15-63: reserved

Digital counters

Get digital counters values

- byte 2: 0xD8
- byte 3: reserved
- byte 4-6: 0
- byte 7: request ID

- bytes 9-21: pin IDs for which the value will be returned

Returned packet:

- byte 2: 0xD8
- byte 3-6: reserved
- byte 7: request ID

- bytes 9-12: counter value 1 (LSB first)
- bytes 13-16: counter value 2 (LSB first)
- bytes 17-20: counter value 3 (LSB first)
- bytes 21-24: counter value 4 (LSB first)
- bytes 25-28: counter value 5 (LSB first)
- bytes 29-32: counter value 6 (LSB first)
- bytes 33-36: counter value 7 (LSB first)
- bytes 37-40: counter value 8 (LSB first)
- bytes 41-44: counter value 9 (LSB first)
- bytes 45-48: counter value 10 (LSB first)
- bytes 49-52: counter value 11 (LSB first)
- bytes 53-56: counter value 12 (LSB first)
- bytes 57-60: counter value 13 (LSB first)

- bytes 61-63: reserved

Get/Set digital counter direction pins

- byte 2: 0xD9
- byte 3: 0 for reading, 1 for writing
- byte 4-6: 0
- byte 7: request ID

- bytes 9-63: direction pin ID for each counter. If pin ID is set to zero, no direction input pin will be used and counter's value will always increase

Reset digital counters values

- byte 2: 0x1D
- byte 3-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x1D

- byte 3-6: 0
- byte 7: request ID

I/O operations

Reading of inputs

- byte 2: 0x30
- byte 3: pin ID (0-54)
- byte 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x30
- byte 3: 0 - OK, 1+ error ID
- byte 4: input value
- byte 5-6: 0
- byte 7: request ID

Block inputs reading

- byte 2: 0x31
- byte 3-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x31
- byte 3: pins state 1-8
- byte 4: pins state 9-16
- byte 5: pins state 17-24
- byte 6: pins state 25-32
- byte 7: request ID

Block inputs reading - part 2

- byte 2: 0x32
- byte 3-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x32
- byte 3: pins state 33-40
- byte 4: pins state 41-48
- byte 5: pins state 49-55
- byte 6: 0
- byte 7: request ID

Analog inputs

Analog inputs reading:

- byte 2: 0x35
- byte 3: pin ID
- byte 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x35
- byte 3: 0 - OK, 1+ error ID
- byte 4: input value (8-bit)
- byte 5: MSB (2-bit) (4-bit on PoKeys56/PoKeys57 devices)
- byte 6: LSB (8-bit)
- byte 7: request ID

Analog inputs block reading - 4x 8bit

- byte 2: 0x36
- byte 3: pin for input 1
- byte 4: pin for input 2
- byte 5: pin for input 3
- byte 6: pin for input 4
- byte 7: request ID

Returned packet:

- byte 2: 0x36
- byte 3: input 1
- byte 4: input 2
- byte 5: input 3
- byte 6: input 4
- byte 7: request ID

Analog inputs block reading - 3x 10bit

- byte 2: 0x37
- byte 3: pin for input 1
- byte 4: pin for input 2
- byte 5: pin for input 3
- byte 6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x37
- byte 3: MSB 1
- byte 4: MSB 2

- byte 5: MSB 3
- byte 6: LSB 1 LSB 2 LSB 3
- byte 7: request ID

Analog inputs reading – all analog inputs in one command

- byte 2: 0x3A
- byte 3: first pin of the series (0 - default)
- byte 4: number of analog input values (0 - default)
- byte 5-6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0x3A
- byte 3-6: reserved
- byte 7: request ID

- bytes 9-x: analog inputs values – 2 bytes per input (1 byte for MSB, 1 byte for LSB)

By default, first value is analog input value on pin 41 (on PoKeys56E, value of 0 on PoKeys55), 42 (also 0 on PoKeys55) and 43-47.

Get analog RC filter value

PoKeys uses a discrete low-pass filter with the following equation:

$$y(k) = y(k-1) * \frac{RC}{RC + 1} + u(k) * \frac{1}{RC + 1}$$

- byte 2: 0x38
- byte 3-6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0x38
- byte 3-6: RC constant (LSB first)
- byte 7: request ID

Set analog RC filter value

- byte 2: 0x39
- byte 3-6: RC constant (LSB first) – 0 turns filtering off
- byte 7: request ID

Returned packet:

- byte 2: 0x39
- byte 3-6: RC constant (LSB first)
- byte 7: request ID

Outputs setting

Early design decision led to digital output values to be inverted. Writing a state 0 to an uninverted output pin results in pin outputting 3.3 V and writing a 1 results in 0 V.

- byte 2: 0x40
- byte 3: pin ID (0-54)
- byte 4: value (0-1)
- byte 5-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x40
- byte 3: 0 - OK, 1+ error ID
- byte 4-6: 0
- byte 7: request ID

Block outputs writing

code 0x42: set block of outputs 1: byte 3-6: output data (1-32)
code 0x43: set block of outputs 2: byte 3-5: output data (33-55)

Analog ouputs settings

- byte 2: 0x41
- byte 3: pin ID (42)
- byte 4: MSB value (0-255)
- byte 5: LSB value (upper 2 bits)
- byte 6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x41
- byte 3: 0 - OK, 1+ error ID
- byte 4-6: 0
- byte 7: request ID

Get device status (extended mode - IO, analog, encoders)

- byte 2: 0xCC
- byte 3: option (0 - short packet, 1 - output data is provided)
- byte 4: reserved
- byte 5: reserved
- byte 6: reserved

- byte 7: request ID

if (option > 0)

- bytes 9-12: output data (1-32) - bit-mapped, bit 0 of byte 9 = pin 1, bit 7 of byte 9 = pin 8, bit 0 of byte 10 = pin 9, etc.
- bytes 13-15: output data (33-55) - bit-mapped output statuses continued
- byte 16: analog output MSB - should be set to 0 for non-PoKeys55 devices
- byte 17: analog output LSB - should be set to 0 for non-PoKeys55 devices
- bytes 18-20: reserved
- bytes 21-27: ignore outputs mask (bit-mapped as digital outputs above), if bit is set, specific pin is not updated with the command call
- bytes 28-63: reserved (0)

Returned packet:

- byte 2: 0x41
- byte 3: 0 - OK, 1+ error ID
- byte 4-6: 0
- byte 7: request ID

- bytes 9-12: input status (1-32)
- bytes 13-15: input status (33-55)
- bytes 16-25: analog 1-5 (MSB+LSB for each input)
- bytes 26-50: 25x 8-bit encoder RAW values
- bytes 51-58: matrix keyboard status (each byte is bit-mapped to a matrix keyboard row)[3]
- bytes 59-62: ultra fast encoder RAW value
- byte 63: reserved (0)

Get custom device status

- byte 2: 0xCC
- byte 3: option (0 - short packet, 1 - output data is provided)
- byte 4: reserved
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

if (option > 0)

- bytes 9-12: output data (1-32) - bit-mapped, bit 0 of byte 9 = pin 1, bit 7 of byte 9 = pin 8, bit 0 of byte 10 = pin 9, etc.
- bytes 13-15: output data (33-55) - bit-mapped output statuses continued
- byte 16: analog output MSB - should be set to 0 for non-PoKeys55 devices
- byte 17: analog output LSB - should be set to 0 for non-PoKeys55 devices
- bytes 18-63: reserved (0)

[3] This status only retrieves first part of the matrix keyboard keys (upper 8x8)

Returned packet:

- byte 2: 0x41
- byte 3: 0 - OK, 1+ error ID
- byte 4-6: 0
- byte 7: request ID

- bytes 9-12: input status (1-32)
- bytes 13-15: input status (33-55)
- bytes 16-25: analog 1-5 (MSB+LSB for each input)
- bytes 26-50: 25x 8-bit encoder RAW values
- bytes 51-58: matrix keyboard status (each byte is bit-mapped to a matrix keyboard row)[4]
- bytes 59-62: ultra fast encoder RAW value
- byte 63: reserved (0)

[4] This status only retrieves first part of the matrix keyboard keys (upper 8x8)

Joystick settings

Joystick feature supports mapping of analog inputs to joystick analog axes and digital inputs to joystick buttons. In the PoKeys57 series, the support for matrix keyboard keys to joystick buttons mapping was included, supporting first 64 keys of the matrix keyboard (8 rows of 8 buttons). The PoKeys pins therefore are referenced as pins 1 to 55, while matrix keyboard keys are referenced as pins 64-127.

Read joystick configuration

- byte 2: 0x60
- byte 3-6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0x60
- bytes 3-6: reserved
- byte 7: request ID
- bytes 9-14: joystick axis mapping
- bytes 15-46: joystick buttons mapping
- bytes 47-50: joystick hat buttons mapping

Set joystick configuration

- byte 2: 0x65
- byte 3-6: reserved
- byte 7: request ID
- bytes 9-14: joystick axis mapping[5]
- bytes 15-46: joystick buttons mapping (**1-based pin codes**, 0 disables the button, if bit 7 is set, this sets down Event pin)
- bytes 47-50: joystick hat buttons mapping (1-based pin codes)

Returned packet:

- byte 2: 0x65
- byte 3: 0 - OK, 1+ error ID
- byte 4-6: 0
- byte 7: request ID

Get joystick up Event buttons configuration

- byte 2: 0x61
- byte 3-6: reserved
- byte 7: request ID

Returned packet:

[5] Set this value to the 1-based pin code (analog inputs have pin codes from 43 to 47), axes have the following order: rotation x, rotation y, x, y, z and throttle

- byte 2: 0x60
- bytes 3-6: reserved
- byte 7: request ID
- bytes 9-14: reserved
- bytes 15-46: joystick buttons mapping for up event

Set joystick up Event buttons configuration

- byte 2: 0x66
- byte 3-6: reserved
- byte 7: request ID
- bytes 9-14: reserved
- bytes 15-46: joystick buttons up Event mapping (**1-based pin codes**, 0 disables the up Event button)

Returned packet:
- byte 2: 0x61
- byte 3: 0 - OK, 1+ error ID
- byte 4-6: 0
- byte 7: request ID

Set/Get joystick analog to digital key mapping options

- byte 2: 0x6A
- byte 3: option
- byte 4-6: reserved
- byte 7: request ID

if (option = 0, 1 or 2)

- bytes 9-63: reserved (0)

if (option = 10) - setup for a lower part of the values

- bytes 9-14: mapping type
  - bit 0: enable key mapping
  - bit 1: direct key mapping
  - bit 2: key mapped to macro (KeyCode is macro ID)
  - bit 3: key mapped to continous macro (same as above, but macro is refiring if input is still active)
  - bit 4: key repeating (after a delay, the key is being repeatedly fired with a given rate)
  - bit 5: reserved
  - bit 6: reserved
  - bit 7: reserved
- bytes 15-20: key code (or macro ID if mapped to macro)
- bytes 21-26: key modifier
- bytes 27-32: Typematic delay (set in steps of 5 ms – 0 to 1275 ms possible)
- bytes 33-38: pin key repeat rate set (repeat period in 5 ms cycles (plus 1 cycle) – set to 9 (9x 5ms + 1x 5ms = 50ms) to get 20 key presses per second, set to 199 to get 1 key press per second)
- bytes 39-44: pin max key repeat rate set
- bytes 45-63: reserved

if (option = 11) - setup for a upper part of the values

- bytes 9-14: mapping type
    bit 0: enable key mapping
    bit 1: direct key mapping
    bit 2: key mapped to macro (KeyCode is macro ID)
    bit 3: key mapped to continous macro (same as above, but macro is refiring if input is still active)
    bit 4: key repeating (after a delay, the key is being repeatedly fired with a given rate)
    bit 5: reserved
    bit 6: reserved
    bit 7: reserved
- bytes 15-20: key code (or macro ID if mapped to macro)
- bytes 21-26: key modifier
- bytes 27-32: Typematic delay (set in steps of 5 ms – 0 to 1275 ms possible)
- bytes 33-38: pin key repeat rate set (repeat period in 5 ms cycles (plus 1 cycle) – set to 9 (9x 5ms + 1x 5ms = 50ms) to get 20 key presses per second, set to 199 to get 1 key press per second)
- bytes 39-44: pin max key repeat rate set
- bytes 45-63: reserved

if (option = 2) - band margins setup

- bytes 9-14: lowest value
- bytes 15-20: lower band deadband value
- bytes 21-26: upper band deadband value
- bytes 27-32: highest value

Returned packet:
- byte 2: 0x6A
- byte 3-6: reserved
- byte 7: request ID

if (option = 0) - setup for a lower part of the values
- bytes 9-14: mapping type
    bit 0: enable key mapping
    bit 1: direct key mapping
    bit 2: key mapped to macro (KeyCode is macro ID)
    bit 3: key mapped to continous macro (same as above, but macro is refiring if input is still active)
    bit 4: key repeating (after a delay, the key is being repeatedly fired with a given rate)
    bit 5: reserved
    bit 6: reserved
    bit 7: reserved
- bytes 15-20: key code (or macro ID if mapped to macro)
- bytes 21-26: key modifier
- bytes 27-32: Typematic delay (set in steps of 5 ms – 0 to 1275 ms possible)
- bytes 33-38: pin key repeat rate set (repeat period in 5 ms cycles (plus 1 cycle) – set to 9 (9x 5ms + 1x 5ms = 50ms) to get 20 key presses per second, set to 199 to get 1 key press per second)
- bytes 39-44: pin max key repeat rate set
- bytes 45-63: reserved

if (option = 1) - setup for a upper part of the values

- bytes 9-14: mapping type
  - bit 0: enable key mapping
  - bit 1: direct key mapping
  - bit 2: key mapped to macro (KeyCode is macro ID)
  - bit 3: key mapped to continous macro (same as above, but macro is refiring if input is still active)
  - bit 4: key repeating (after a delay, the key is being repeatedly fired with a given rate)
  - bit 5: reserved
  - bit 6: reserved
  - bit 7: reserved
- bytes 15-20: key code (or macro ID if mapped to macro)
- bytes 21-26: key modifier
- bytes 27-32: Typematic delay (set in steps of 5 ms – 0 to 1275 ms possible)
- bytes 33-38: pin key repeat rate set (repeat period in 5 ms cycles (plus 1 cycle) – set to 9 (9x 5ms + 1x 5ms = 50ms) to get 20 key presses per second, set to 199 to get 1 key press per second)
- bytes 39-44: pin max key repeat rate set
- bytes 45-63: reserved

if (option = 2) - band margins setup

- bytes 9-14: lowest value
- bytes 15-20: lower band deadband value
- bytes 21-26: upper band deadband value
- bytes 27-32: highest value

Macros

Create macro

- byte 2: 0x90
- byte 3: reserved
- byte 4: macro length
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0x90
- byte 3: macro ID
- byte 4: macro length
- byte 5: 0 - OK, 1+ error ID
- byte 6: reserved
- byte 7: request ID

Modify macro

- byte 2: 0x91
- byte 3: macro ID
- byte 4: new length
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0x91
- byte 3: macro ID
- byte 4: new length
- byte 5: 0 - OK, 1+ error ID
- byte 6: reserved
- byte 7: request ID

Delete macro

- byte 2: 0x92
- byte 3: macro ID
- byte 4: reserved
- byte 5: reserved

- byte 6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0x92
- byte 3: macro ID
- byte 4: reserved
- byte 5: 0 - ok, 1+ error ID
- byte 6: reserved
- byte 7: request ID

Save macros to flash

- byte 2: 0x93
- byte 3: reserved
- byte 4: reserved
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

Returned packet:

- byte 2: reserved
- byte 3: reserved
- byte 4: reserved
- byte 5: 0 - ok, 1+ error ID
- byte 6: reserved
- byte 7: request ID

Rename macro

- byte 2: 0x94
- byte 3: macro ID
- byte 4: index [0..3]
- byte 5: char 1
- byte 6: char 2
- byte 7: request ID

Returned packet:

- byte 2: 0x94
- byte 3: macro ID
- byte 4: reserved
- byte 5: 0 - ok, 1+ error ID
- byte 6: reserved
- byte 7: request ID

Transfer macro

- byte 2: 0x95
- byte 3: macro ID
- byte 4: index
- byte 5: key code
- byte 6: key modifier
- byte 7: request ID

Returned packet:

- byte 2: 0x95
- byte 3: macro ID
- byte 4: reserved
- byte 5: 0 - ok, 1+ error ID
- byte 6: reserved
- byte 7: request ID

Get macro length

- byte 2: 0x96
- byte 3: macro ID
- byte 4: reserved
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0x96
- byte 3: macro ID
- byte 4: macro length
- byte 5: 0 - ok, 1+ error ID
- byte 6: reserved
- byte 7: request ID

Get macro name

- byte 2: 0x97
- byte 3: macro ID
- byte 4: index [0..3]
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0x97
- byte 3: macro ID
- byte 4: index
- byte 5: char 1
- byte 6: char 2
- byte 7: request ID

Get macro keys

- byte 2: 0x98
- byte 3: macro ID
- byte 4: index [0..255]
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0x98
- byte 3: macro ID
- byte 4: index [0..255]
- byte 5: key code
- byte 6: key modifier
- byte 7: request ID

Get free space

- byte 2: 0x99
- byte 3: reserved
- byte 4: reserved
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0x99
- byte 3: free space MSB
- byte 4: free space LSB
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

Get active macros

- byte 2: 0x9A
- byte 3: page [0..1]
- byte 4: reserved
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0x9A
- byte 3: bit masked macro enabled MSB
- byte 4: bit masked macro enabled
- byte 5: bit masked macro enabled
- byte 6: bit masked macro enabled LSB
- byte 7: request ID

Set/Get macro name and length

- byte 2: 0xC8
- byte 3: option
- byte 4: macro ID
- byte 5-6: 0
- byte 7: request ID

if (option > 0)

- bytes 9-15: new macro name

Returned packet:

- byte 2: 0xC8
- byte 3-6: reserved
- byte 7: request ID

- bytes 9-15: macro name
- byte 16: macro length

Set/Get macro keys

- byte 2: 0xC9
- byte 3: option
- byte 4: macro ID
- byte 5: page (25 keys per page)
- byte 6: length
- byte 7: request ID

if (option > 0)

- bytes 9-58: key+modifier pairs
- bytes 59-63: reserved (0)

Returned packet:

- byte 2: 0xC9
- byte 3-6: reserved
- byte 7: request ID

- bytes 9-58: key+modifier pairs
- bytes 59-63: reserved (0)

Matrix keyboard

Get/Set matrix keyboard configuration

- byte 2: 0xCA
- byte 3: option
- byte 4: keyboard ID
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

If option == 1[6]

- byte 9: new configuration
  - bit 0: enable matrix keyboard
  - bit 1-7: reserved
- byte 10: size of matrix keyboard[7]
  - bit 0-3: height-1
  - bit 4-7: width-1
- bytes 11-18: row pins[8]
- bytes 19-26: column pins
- bytes 27-42: bit mapped direct/macro (1 for macro), 27.0 for key 0
- bytes 43-50: row pins (if height set to 8 or greater)
- byte 51: pin alternate function enable – pinID (must be input)+1 (value read only if option == 1)
- bytes 52-63: reserved

If option == 2...9[9]

- bytes 9-24: key codes (keys (option-2)*16 – (option-1)*16-1)
- bytes 25-40: key modifiers
- bytes 41-42: triggered mode (bit mapped for above keys) – ignored if alternate function is enabled
- bytes 43-63: reserved

If option == 22...29[10]

- bytes 9-24: key codes (keys (option-2)*16 – (option-1)*16-1) for up key event (if triggering enabled)
- bytes 25-40: key modifiers for above keys
- bytes 41-63: reserved

Returned packet:

- byte 2: 0xCA
- byte 3: keyboard ID
- byte 4-6: reserved

---

[6] Use option 1 to setup of the matrix keyboard

[7] This size defines the size of the matrix keyboard in use. For example, if width is set to 4, only columns A-D are used and thus only first four column pins are checked, others are ignored.

[8] Set pin codes appropriately (pins codes are 0-based, so pin 1 has the pin code 0). If width is set to 4, only first four pin codes are read. For unused pins use any value, 0 or 255 is recommended.

[9] Use options 2-5 to setup key codes mapping of the pins. First row of matrix keyboard has keys with indexes from 0 to 7, second row from 8 to 15... Even if matrix keyboard is setup as having only four columns, first row still has keys with indexes 0 to 3, second row from 8 to 11... (values of keys 4-7, 12-15 are not refreshed). Option 2 sets key codes for keys 0-15, Option 3 for keys 16-31...

[10] Use options 2-5 to setup key codes mapping of the pins. First row of matrix keyboard has keys with indexes from 0 to 7, second row from 8 to 15... Even if matrix keyboard is setup as having only four columns, first row still has keys with indexes 0 to 3, second row from 8 to 11... (values of keys 4-7, 12-15 are not refreshed). Option 2 sets key codes for keys 0-15, Option 3 for keys 16-31...

- byte 7: request ID

If option < 12[11]

- byte 9: new configuration
  bit 0: enable matrix keyboard
  bit 1-7: reserved
- byte 10: size of matrix keyboard
  bit 0-3: height-1
  bit 4-7: width-1
- bytes 11-18: row pins
- bytes 19-26: column pins
- bytes 27-42: bit mapped direct/macro (1 for macro), 27.0 for key 0
- bytes 43-50: row pins (if height set to 8 or greater)
- byte 51: pin alternate function enable – pinID (must be input)+1 (value read only if option == 1)
- bytes 52-63: reserved

If option == 12..19[12]

- bytes 9-24: key codes (keys (option-12)*16 – (option-11)*16-1)
- bytes 25-40: key modifiers
- bytes 41-42: triggered mode
- bytes 43-63: reserved

If option == 32..39[13]

- bytes 9-24: key codes (keys (option-12)*16 – (option-11)*16-1) for up key event
- bytes 25-40: key modifiers
- bytes 41-63: reserved

If option == 20

- bytes 9-24: matrix keyboard status (whole 16x8 matrix keyboard)

PWM channels

Get/set PWM configuration

- byte 2: 0xCB
- byte 3: option
- byte 4: option 2 – set to 1 to update only PWM duty values, else set to 0
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

If option > 0

- byte 9: bit-mapped PWM enabled

[11] Use option less than 12 to retrieve the configuration of the matrix keyboard. If option is used, that is not previously defined for setup of matrix keyboard, settings are only read, none are set.

[12] Retrieve key codes for keys. Look above for description for options 2-5. This options do not change the setup of matrix keyboard.

[13] Retrieve key codes for keys. Look above for description for options 2-5. This options do not change the setup of matrix keyboard.

bit 0: enable PWM1 (pin 22)
bit 1: enable PWM2 (pin 21)
bit 2: enable PWM3 (pin 20)
bit 3: enable PWM4 (pin 19)
bit 4: enable PWM5 (pin 18)
bit 5: enable PWM6 (pin 17)
- bytes 10-13: PWM1 value (LSB first)
- bytes 14-17: PWM2 value
- bytes 18-21: PWM3 value
- bytes 22-25: PWM4 value
- bytes 26-29: PWM5 value
- bytes 30-33: PWM6 value

- bytes 34-37: PWM period
- bytes 38-63: reserved

Returned packet:

- byte 2: 0xCB
- bytes 3-6: reserved
- byte 7: request ID

- byte 9: bit-mapped PWM enabled
bit 0: enable PWM1 (pin 22)
bit 1: enable PWM2 (pin 21)
bit 2: enable PWM3 (pin 20)
bit 3: enable PWM4 (pin 19)
bit 4: enable PWM5 (pin 18)
bit 5: enable PWM6 (pin 17)
- bytes 10-13: PWM1 value
- bytes 14-17: PWM2 value
- bytes 18-21: PWM3 value
- bytes 22-25: PWM4 value
- bytes 26-29: PWM5 value
- bytes 30-33: PWM6 value

- bytes 34-37: PWM period
- bytes 38-63: reserved

LCD displays

Primary pin assignment	Secondary pin assignment
- DB4 = Pin 26 - DB5 = Pin 25 - DB6 = Pin 24 - DB7 = Pin 23 - E = Pin 30 - RW = Pin 28 - RS = Pin 29	- DB4 = Pin 34 - DB5 = Pin 33 - DB6 = Pin 32 - DB7 = Pin 31 - E = Pin 30 - RW = Pin 28 - RS = Pin 29

Set LCD configuration

- byte 2: 0xD0
- byte 3: 0 for writing, 1 for reading only
- byte 4: LCD enabled
  0 – LCD disabled
  1 – LCD enabled on primary pins (23-26)
  2 – LCD enabled on secondary pins (31-34)
- byte 5: number of rows
- byte 6: number of columns
- byte 7: request ID

Returned packet:

- byte 2: 0xD0
- byte 3: reserved
- byte 4: LCD enabled
- byte 5: number of rows
- byte 6: number of columns
- byte 7: request ID

LCD operation

- byte 2: 0xD1
- byte 3: LCD operation
- byte 4-6: reserved
- byte 7: request ID

LCD operations

- Init LCD – operation code 0
  - No additional parameters
- Clear LCD – operation code 0x10
  - No additional parameters
- Move cursor – operation code 0x20
  - Byte 4: x position (column) (x=1 for the first column)
  - Byte 5: y position (row) (y=1 for the first row)
- Print to LCD – operation code 0x30
  - Bytes 9-29: string to be printed on screen (up to 20 characters, \0 terminated)
- Put character to LCD – operation code 0x31
  - Byte 9: character code
- Define custom character – operation code 0x40
  - Byte 9: character code
  - Bytes 10-17: character data
- Entry mode set – operation code 0x50
  - Byte 9: cursor move direction (1 – increment, 0 – decrement)
  - Byte 10: display shift on/off
- Display on/off control – operation code 0x60
  - Byte 9: display on/off
  - Byte 10: cursor on/off
  - Byte 11: cursor blinking on/off
- Set LCD mode – operation code 0x80
  - Byte 4: 0 - Direct (default) mode / 1 - Buffered mode
- Buffered mode write – operation code 0x85

- Byte 4: row
- Bytes 9-33: 24 bytes

Returned packet:

- byte 2: 0xD1
- byte 3-6: reserved
- byte 7: request ID

Matrix LED display operations

Matrix LED display pins are fixed due to hardware design. On PoKeys55 device (pin numbers for PoKeys prototype design are given in parenthesis), pins used are

Display 1:

- Pin **9** (10): serial data
- Pin **10** (11): output register clock
- Pin **11** (12): serial clock

Display 2:

- Pin **23** (23): serial data
- Pin **24** (24): output register clock
- Pin **25** (25): serial clock

Get/set Matrix LED display configuration

- byte 2: 0xD5
- byte 3: option[14]
- byte 4: matrix LED enabled
  - bit 0: enable display 1
  - bit 1: enable display 2
- byte 5: display 1 size
  - bits 0-3: number of rows (1...8)
  - bits 4-7: number of columns (1...8)
- byte 6: display 2 size
  - bits 0-3: number of rows (1...8)
  - bits 4-7: number of columns (1...8)
- byte 7: request ID

Returned packet:

- byte 2: 0xD5
- byte 3: reserved
- byte 4: matrix LED enabled
- byte 5: display 1 size
- byte 6: display 2 size
- byte 7: request ID

Update matrix LED display

- byte 2: 0xD6
- byte 3: action
  - 1 - update whole display 1 (ignoring row and column bytes)
  - 5 - set pixel at row,column on display 1 (ignoring row data bytes 9-16)
  - 6 - clear pixel at row,column on display 1
  - 11 - update whole display 2 (ignoring row and column bytes)
  - 15 - set pixel at row,column on display 2 (ignoring row data bytes 9-16)
  - 16 - clear pixel at row,column on display 2

[14] To set the configuration, set option byte to 0, else only reading operation will commence

- byte 4: row[15]
- byte 5: column[16]
- byte 6: reserved
- byte 7: request ID

- byte 9-16: row data (LSB bit of each byte is assigned to a pixel on left of a row)
- bytes 17-63: reserved

Returned packet:

- byte 2: 0xD6
- bytes 3-6: reserved
- byte 7: request ID

PoExtBus functionality

PoExtBus enables to extend number of PoKeys device outputs for 80. This is acomplished using up to 10 daisy-chained 8-bit shift registers with latches. Bit 0 of byte 0 is sent first, followed by bits 1-7 then bits 0-7 of byte 1... If shorter chains of shift registers are used, use only the highest bytes (in case only one shift register is used, only use byte 9 to send data).
Connector option 0 (only on PoKeys56E and PoKeys56U): PoExtBus has a dedicated connector on PoKeys56 board. This option enables this connector.
Pins used are (PoKeys55 and PoKeys56) – connector option 1

- Pin code **34** (35): serial clock
- Pin code **35** (36): serial data
- Pin code **36** (37): output register clock

Set PoExtBus settings

- byte 2: 0xDA
- byte 3: PoExtBus option (set to 1 to enable PoExtBus, set to 0 to disable and set to 2 to read the state)
- byte 4: connector selection (only on PoKeys56E and PoKeys56U)
- bytes 5-6: reserved
- byte 7: request ID
- bytes 9-18: data bytes

Returned packet:

- byte 2: 0xDA
- byte 3: PoExtBus status
- byte 4: Connector selection
- byte 5-6: reserved
- byte 7: request ID

- bytes 9-18: data bytes

[15] Row and column indexes are 0-based.
[16] Row and column indexes are 0-based.

I²C communication bus

PoKeys56E devices support communication with I²C slave devices, connected to the PoExtBus connector. Both I²C and PoExtBus use the same connector and those two protocols share the connector on a priority-based rule. By default, both PoExtBus and I²C are enabled, but PoExtBus update has higher priority than I²C. When PoExtBus outputs must be refreshed, I²C is temporarily disabled, PoExtBus is refreshed, then I²C is enabled again (all this is transparently done by PoKeys device itself). Up to 30 bytes can be transferred in one I²C transaction.
Marking the pin closer to the bottom of the board (the oposite side of either Ethernet or USB connector) as pin 1, the I²C devices should be connected as follows:

Pin 1	**Power supply 3.3V**
Pin 2	Ground
Pin 3	Serial data
Pin 4	
Pin 5	Serial clock

I²C settings and communication

- byte 2: 0xDB
- byte 3: I²C operation
- byte 4-6: reserved (defined below)
- byte 7: request ID

I²C operations

- ~~0x00 - Deactivate I²C~~ **- deprecated command, I²C bus is always activated**
  - No additional parameters
- ~~0x01 - Activate I²C~~ **- deprecated command, I²C bus is always activated**
  - No additional parameters
- ~~0x02 – Get activation status~~ **- deprecated command, I²C bus is always activated**
  - No additional parameters (returns successfull if I2C turned on)
- 0x10 - Write to I²C - start
  - Byte 4: address of the device
  - Byte 5: length of data packet
  - Byte 6: number of bytes to read after write
  - Bytes 9-40: data bytes
- 0x11 - Write to I²C – get result
- 0x20 - Read from I²C - start
  - Byte 4: address of the device
  - Byte 5: length of data packet
- 0x21 - Read from I²C – get result
- 0x30 - Scan I²C – start
- 0x31 - Scan I²C – get result

Returned packet:

- byte 2: 0xDB
- byte 3: I²C operation
- byte 4: I²C operation result (1 if successfull, 0 unsuccessfull, 0x10 – operation still executing)
- byte 5-6: reserved
- byte 7: request ID

I²C operations

- 0x21 - Read from I²C – get result
  - Byte 9: operation result (copied from byte 4)
  - Byte 10: data length

    - Bytes 11-42: data bytes
- 0x31 - Scan I^2C – get result
    - Byte 9: operation result (copied from byte 4)
    - Bytes 10-25: bit encoded result (if bit 0 of byte 10 is set, I2C device with the address of 0x00 was detected)

PoNET – »PoI2C« commands

PoNET settings and communication

- byte 2: 0xDD
- byte 3: PoI2C operation
- byte 4-6: reserved (defined below)
- byte 7: request ID

PoNET operations

- 0x00 – Get PoNET status
- 0x10 – Get PoNET module settings (i2c address, type, size, options)
  - Byte 4: Module ID
- 0x11 – Get PoNET firmware version
  - Byte 4: Module ID
- 0x15 – Set PoNET module settings (mapping options)
  - Byte 4: Module ID
  - Byte 5: Mapping options
- 0x20 – Clear PoNET module settings
  - Byte 4: Module ID
- 0x21 – Reinitialize PoNET
- 0x25 – Reinitialize PoNET and clear settings
- 0x30 – New device discovery
  - Byte 4: Activate (0x10) / deactivate (0x20) /get status (0x30)
- 0x40 – Check for devices
  - Byte 4: Start (0x10) / Get status (0x30)
- 0x50 – Get PoNET module data
  - Byte 4: Start (0x10) / Get data and status (0x30)
  - Byte 5: Module ID
- 0x55 – Set PoNET module data
  - Byte 4: Module ID
  - Bytes 9-24: data
- 0x60 – Get Light sensor value
  - Byte 4: Start (0x10) / Get data and status (0x30)
  - Byte 5: Module ID
- 0x70 – Set PWM value
  - Byte 4: Module ID
  - Byte 5: PWM value
- 0xF0 – Start bootloader
  - Byte 4: Execute (0x10) / get status (0x30)
  - Byte 5: Module ID
- 0xF1 – Start programming
  - Byte 4: Execute (0x10) / get status (0x30)
- 0xF2 – Transfer firmware part
  - Byte 4: Execute (0x10) / get status (0x30)
  - Bytes 9-16: Firmware data (8 bytes)
- 0xF3 – Finish firmware transfer and restart
- 0xF4 – Exit bootloader mode
  - Byte 4: Execute (0x10) / get status (0x30)
- 0xF5 – Activate bootloader
  - Byte 4: Execute (0x10) / get status (0x30)

Returned packet:

- byte 2: 0xDD
- byte 3: PoNET operation
- byte 4: PoNET operation result (1 if successfull, 0 unsuccessfull, 0x10 – operation still executing)
- byte 5-6: reserved

- byte 7: request ID

PoNET operations:

- 0x00 – Get PoNET status
  - Byte 9: PoNET state

    PoNET_inactive = 0,
    PoNET_initializing = 1,
    PoNET_scanningI2C = 2,
    PoNET_initialized = 10,

    PoNET_newDevice = 20,
    PoNET_newDeviceCheck = 21,

    PoNET_retrievingConfigurationCommand = 30,
    PoNET_retrievingConfigurationResponse = 35,

    PoNET_readingStatusCommand = 50,
    PoNET_readingStatusResponse = 55,

    PoNET_writingData = 60,

    PoNET_readingLightCommand = 70,
    PoNET_readingLightResponse = 71,

    PoNET_settingPWMcommand = 80,

    PoNET_bootloader = 100,

    PoNET_PoExtBusScanning = 200,

    PoNET_error = 255
- 0x10 – Get PoNET module settings (i2c address, type, size, options)
  - Byte 9: assigned i2c address
  - Byte 10: module type
    - 0x10 = PoEBkb v1
  - Byte 11: module size
  - Byte 12: module options
    - bit 0 - has inputs
    - bit 1 - has outputs
    - bit 2 - x-y rotated
    - bit 3 - light sensor present
    - bit 4 - device mapped to PoKeys peripheral
    - bit 5 - configuration retrieved
    - bit 6 - device configured
    - bit 7 - device present
- 0x11 – Get PoNET firmware version
  - Byte 9: firmware version
- 0x30 – New device discovery
  - Byte 9 (get status): 1 if newly added device was configured
- 0x40 – Check for devices
  - Byte 9 (get status): 1 if unconfigured device detected
- 0x50 – Get PoNET module data – get data ans status (0x30)
  - Byte 9: Status (0xFF if still reading)
  - Bytes 10-25: PoNET module data
- 0x60 – Get Light sensor value - get data ans status (0x30)

- Byte 9: status (0xFF if still reading)
- Byte 10: sensor value

- 0xF0 – Start bootloader
  - Byte 9: bootloader started flag
- 0xF1 – Start programming
  - Byte 9: programming started flag
- 0xF2 – Transfer firmware part
  - Byte 9: transfer complete
- 0xF5 – Activate bootloader
  - Byte 9: 1 if bootloader detected

kbd48CNC specifics:

Left the upper left key be designated with coordinates (0,0), x-axis going to the right of the keyboard and the y-axis going to the bottom. Then the key statuses can be decoded as

```
    private Point DecodeButtonPosition(int byteIndex, int bitIndex)
{
    Point t = new Point();

    t.X = (byteIndex ^ 1) * 2;
    if (bitIndex < 4)
    {
        t.X++;
        t.Y = bitIndex;
    }
    else
    {
        t.Y = 7 - bitIndex;
    }

    return t;
}

private void SetLED(int x, int y, bool status)
{
    int byteindex = 1 ^ (int)(x / 2);
    int bitindex = y;

    if ((x % 2) == 0)
    {
        bitindex = 7 - y;
    }

    if (status)
    {
        LEDstatus[byteindex] |= (byte)(1 << bitindex);
    }
    else
    {
        LEDstatus[byteindex] &= (byte)~(1 << bitindex);
    }
}
```

SPI communication bus

PoKeys56U, PoKeys56E and PoKeys57E devices support communication with SPI slave devices. Chip select pins can be freely selected. SPI configuration is not saved in the device.

SPI pin	PoKeys56U pin ID	PoKeys56E/PoKeys57E pin ID
MOSI	9	23
MISO	10	28
SCK	11	25

SPI settings and communication

- byte 2: 0xE5
- byte 3: SPI operation
- byte 4-6: reserved (defined below)
- byte 7: request ID

SPI operations

- 0x00 - Disable SPI (not implemented)
  - No additional parameters
- 0x01 - Enable SPI
  - Byte 4: Prescaler value (2-254, even only)
  - Byte 5: SPI format (bit 0 = CPOL, bit 1 = CPHA)
- 0x10 - Write to SPI - start
  - Byte 4: length of data packet (1 to 16, 1 to 55 with 3.1.46)
  - Byte 5: pin ID to use as CS
  - Bytes 9-63: data bytes
- 0x20 - Read SPI
  - Byte 4: data length

Returned packet:

- byte 2: 0xE5
- byte 3: SPI operation
- byte 4: SPI operation result (1 if successfull, 0 unsuccessfull, 0x10 – operation still executing)
- byte 5-6: reserved
- byte 7: request ID

SPI operations

- 0x20 - Read SPI – get result
  - Bytes 9-63: data bytes

1-wire communication bus

1-wire protocol uses Pin 55 and needs external 5kΩ pull-up resistor. On PoKeys57 series devices, any pin can be used for 1-wire communication.

1-wire settings and communication

- byte 2: 0xDC
- byte 3: 1-wire operation
- byte 4: PoKeys pin (only on PoKeys57)
- byte 5-6: reserved (defined below)
- byte 7: request ID

1-wire operations

- 0x00 - Deactivate 1-wire
  - No additional parameters
- 0x01 - Activate 1-wire
  - No additional parameters
- 0x02 – Get activation status (returns successfull if 1-wire turned on)
- 0x10 – Start Reset, Write and Read process
  - Byte 4: number of bytes to write (up to 16)
  - Byte 5: number of bytes to read (up to 16)
  - Byte 6: pin ID (PoKeys57)
  - Bytes 9-24: data bytes
- 0x11 – Get result of read process
- 0x20 - Start bus scan (PoKeys57)
  - Byte 4: pin ID
- 0x21 - Get bus scan status (PoKeys57)
- 0x22 - Continue bus scan (PoKeys57)
- 0x23 - Stop bus scan (PoKeys57)

Returned packet:

- byte 2: 0xDC
- byte 3: 1-wire operation
- byte 4: 1-wire operation result (1 if successfull, 0 unsuccessfull, 0x10 – operation still executing)
- byte 5-6: reserved
- byte 7: request ID

1-wire operations

- 0x11 - Get result of read process
  - Byte 9: operation result (copied from byte 4)
  - Byte 10: data length
  - Bytes 11-26: data bytes
- 0x21 - Get bus scan status
  - Byte 9: operation result (copied from byte 4)
  - Byte 10: scan result (bit 0: scan stage complete, bit 1: all devices scanned)
  - Bytes 11-18: device ROM

UART communication

0xDE/0x10 Setup UART communication

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xDE**
3	0x10 - Setup UART communication
4	Interface (1, 2 or 3)
5	Frame format
6	Reserved
7	Request ID
8 (CRC)	CRC
9-12	Baudrate
13-63	Reserved
64 (CRC 2)	unused

Bit	Symbol	Value	Description
1:0	Word Length Select	00	5-bit character length
		01	6-bit character length
		10	7-bit character length
		11	8-bit character length
2	Stop Bit Select	0	1 stop bit.
		1	2 stop bits (1.5 if UnLCR[1:0]=00).
3	Parity Enable	0	Disable parity generation and checking.
		1	Enable parity generation and checking.
5:4	Parity Select	00	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.
		01	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.
		10	Forced "1" stick parity.
		11	Forced "0" stick parity.

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xDE**
3	0x10 - Setup UART communication
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

0xDE/0x20 Send data

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xDE**
3	0x20 - Send data
4	Interface (1, 2 or 3)

5	Byte count
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Data bytes
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xDE**
3	0x20 - Send data
4	Data bytes queued
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

0xDE/0x30 Read data

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xDE**
3	0x30 - Read data
4	Interface (1, 2 or 3)
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xDE**
3	0x30 - Read data
4	Data bytes available
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Data bytes
64 (CRC 2)	unused

Real-Time mode (experimental, not implemented in current release)

Real-Time mode enables application to increase the speed of capturing data from PoKeys device on the account of reliabilty (similar to the relation between TCP and UDP protocols).
In RTmode, PoKeys device will constantly respond with the packet 0xA1. RTmode is automatically disabled when any other packet (other than 0xA1) is received.

RTmode – setup

- byte 2: 0xA0
- byte 3: enable RTmode (set to 1 to enable, set to 0 to disable)
- byte 4: reserved
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0xA0
- byte 3: 0 - OK, 1+ error ID
- byte 4-6: 0
- byte 7: request ID

RTmode – set/get data

- byte 2: 0xA1
- byte 3: option (1 - output data is provided)
- byte 4: reserved
- byte 5: reserved
- byte 6: reserved
- byte 7: request ID

if (option > 0)

- bytes 9-12: output data (1-32) (LSB first)
- bytes 13-15: output data (33-55) (LSB first)
- bytes 16-39: 6x 32-bit PWM outputs duty cycles (MSB first)
- bytes 40-63: reserved (0)

Returned packet:

- byte 2: 0xA1
- byte 3: 0 - OK, 1+ error ID
- byte 4-6: 0
- byte 7: request ID

- bytes 9-12: input status (1-32) (LSB first)
- bytes 13-15: input status (33-55) (LSB first)
- bytes 16-29: 12-bit analog inputs 1-7 (MSB+LSB for each input)
- bytes 30-45: 4x 32-bit fast encoder RAW values (MSB first)

- bytes 46-49: tick counter (MSB first)
- bytes 50-63: reserved (0)

Network settings

PoKeys56E device is a network device. It uses a combination of TCP/IP and UDP/IP to communicate. It supports either fixed IP or IP assigned by DHCP server.
Due to exposed nature of the network device, a simple authentication mechanism was implemented. When the device is locked, user must enter the password (or an application instead of user directly) at the beginning of each new connection.
First, the 'Get security setting status' must be called, which returns the current level of security (0 is fully unlocked, 0xFF is fully locked) and the 32-bytes of hash seed. When user enters the password, the ASCII values of the password is XOR-ed with the hash seed to produce 32-bytes of data that is fed into the SHA-1 algorithm to calculate 20-bytes of SHA-1 hash value. This value is then sent as an authentication password in the 'Autorise user' command. This approach provides some degree of protection against unathorized use of the device.

Get/set network configuration

- byte 2: 0xE0
- byte 3: option[17]
- bytes 4-6: reserved
- byte 7: request ID

If (option == 10)[18]

- byte 9: IP setup – 0 for fixed IP, 1 for DHCP server assigned one
- bytes 10-13: fixed IP
- bytes 14-17: reserved
- bytes 18-19: TCP connection timeout (default **30** x100 ms = 3s)
- bytes 20-23: Gateway IP
- bytes 24-27: Subnet mask
- byte 28: 1 if gateway and subnet are also set
- byte 29: additional options - bits 7:4 are 0xA, lower are the following:
    - bit 3: reserved
    - bit 2: disable IP configuration via UDP broadcast
    - bit 1: disable automatic device IP configuration during discovery
    - bit 0: disable automatic device discovery mechanism

Returned packet:

- byte 2: 0xE0
- byte 3-6: reserved
- byte 7: request ID
- byte 9: IP setup
- bytes 10-13: fixed IP
- bytes 14-17: current IP (if assigned by DHCP server)
- bytes 18-19: TCP connection timeout
- bytes 20-23: Gateway IP
- bytes 24-27: Subnet mask
- byte 28: additional options as described in the packet above

[17] To set the configuration, set option byte to 10, else only reading operation will commence
[18] This command also saves the configuration

Get security setting status

- byte 2: 0xE1
- bytes 3-6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0xE1
- byte 3-6: reserved
- byte 7: request ID
- byte 9: current security level
- bytes 10-41: seed for calculating the password hash

Authorise user

- byte 2: 0xE2
- byte 3: security level to unlock to (0 means full unlock)
- bytes 4-6: reserved
- byte 7: request ID
- byte 9-28: password hash code

Returned packet:

- byte 2: 0xE2
- byte 3-6: reserved
- byte 7: request ID
- byte 9: unlock status (0 means error, 0xAA means succeded)

Set user password

- byte 2: 0xE3
- byte 3: Default security setting
- bytes 4-6: reserved
- byte 7: request ID
- byte 9-40: password in plain text

Returned packet:

- byte 2: 0xE3
- byte 3-6: reserved
- byte 7: request ID

Modbus settings

For Modbus in PoKeys56E these options can be specified:

- **Modbus port number**: default port number for Modbus TCP protocol is 502. User can however change the port number to other values
- **Modbus connection timeout**: When the connection is not in use for more than (Timeout x 100ms), the connection is dropped and new connections can be established
- **Modbus read/write access settings**: User can define, which peripherals are accessible through Modbus protocol (access settings are bit encoded):

```
#define  access_IO                  (1<<0)
#define  access_analogIn            (1<<1)
#define  access_PWMOut              (1<<2)
#define  access_MatrixKB            (1<<3)
#define  access_I2CMatrixKB         (1<<4)
#define  access_I2CMatrixKBLED      (1<<5)
#define  access_LEDMatrix           (1<<6)
#define  access_PoExtBus            (1<<7)
#define  access_Encoders            (1<<8)
#define  access_Counters            (1<<9)
#define  access_LCD                 (1<<10)
#define  access_SensorList          (1<<11)
#define  access_PoIL                (1<<12)
#define  access_PoILcore            (1<<13)
```

Modbus registers:

Address (0-based)	Access (R – Read, W – Write)	Description
10-16	R	Analog inputs
20-45	RW	Encoder counter values (lower 16-bit)
100-154	RW	Digital counter values
200-213	RW	PWM
200,201		PWM period (MSB first)
202,203		PWM duty1 (MSB first) – pin 22
...		
212,213		PWM duty6 (MSB first) – pin 17
300-304	RW	PoExtBus
400-453	R	Sensors (32-bit values, LSB first)
500-579	RW	LCD buffer
590	W	LCD configuration (0=disabled, 1=primary or 2=secondary) - writing to this register will re-init and clear the LCD
591	W	Number of rows (lower byte) and number of columns (upper byte) of the LCD module
592	W	Not used
593	W	Clear LCD (both bytes = 0xAA)
600	R	Tick counter (lower 16-bit)
610	RW	RTC seconds
611	RW	RTC minutes
612	RW	RTC hours
613	RW	RTC day
614	RW	RTC day of week
615	RW	RTC month
616	RW	RTC year

620	RW	PoIL core state Set to: 0 - stop core 1 - reset core 10 - set core into run mode
700-751	R[W]	Digital encoder values (32-bit values, LSB first) - any write to these registers causes the reset of the encoder value to 0
800-909	RW	Digital counter values (32-bit values, LSB first)
1000-1127	RW	PoIL shared data slots (32-bit values/2 registers per slot, LSB first!)

Get/set modbus settings

- byte 2: 0xE4
- byte 3: option[19]
- bytes 4-6: reserved
- byte 7: request ID

If (option == 10)

- bytes 9-10: Modbus port number (LSB first)
- bytes 11-12: Modbus connection timeout (in x100 ms)
- bytes 13-16: Modbus read access settings
- bytes 17-20: Modbus write access settings

Returned packet:

- byte 2: 0xE4
- byte 3-6: reserved
- byte 7: request ID
- bytes 9-10: Modbus port number
- bytes 11-12: Modbus connection timeout (in x100 ms)
- bytes 13-16: Modbus read access settings
- bytes 17-20: Modbus write access settings

[19] To set the configuration, set option byte to 10, else only reading operation will commence

Web interface

PoKeys56E acts like a HTTP server and serves simple web pages that reflect certain information on the state of the device. The web interface has the following options:

- Web interface can be disabled – PoKeys56E won't react to http requests on port 80
- Anonymous access to dashboard and IO status can be enabled
- Outputs can be allowed to be toggled via web interface

For security measures, PoKeys56E supports one administrator account and three additional user accounts.

Setup web interface settings

- byte 2: 0x73
- byte 3: if setting the configuration, set bit 7 to 1
- byte 4-6: reserved
- byte 7: request ID
- bytes 9: disable web interface (disabled if 1)
- bytes 10: allow anonymous access to dashboard and IO status page (enabled if 1)
- bytes 11: allow anonymous access to dashboard and IO status page (enabled if 1)

Returned packet:

- byte 2: 0x73
- byte 3-6: reserved
- byte 7: request ID

- bytes 9: disable web interface (disabled if 1)
- bytes 10: allow anonymous access to dashboard and IO status page (enabled if 1)
- bytes 11: allow anonymous access to dashboard and IO status page (enabled if 1)

User accounts

Four user accounts exist. First is for the administrator and its username can not be changed. It defaults to Admin.

Setup user account

- byte 2: 0x72
- byte 3: user account ID (0 to 3, set bit 7 to set the configuration)
- byte 4-6: reserved
- byte 7: request ID
- bytes 9-16: user name (8 characters)
- bytes 17-24: password (8 characters)

Returned packet:

- byte 2: 0x72
- byte 3-6: reserved
- byte 7: request ID

- bytes 9-16: user name (8 characters)
- bytes 17-24: password (8 characters)

Dashboard items

PoKeys56E supports up to 16 dashboard items. Each item is linked to one of the sensors from the sensor list (I2C, 1-wire and analog sensors) and additional digital inputs and outputs.

Dashboard item types (with supported display types)

- 0: unused (inactive)
- 1: digital input (only display type 0)
- 2: digital output (only display types 0 and 1)
- 3: analog input
- ~~*4: PoExtBus output*~~ (only display types 0 and 1) *will be supported in the next release*
- 5: sensor input
- 6: digital counters
- 7: PoIL shared data

Display types

All analog displays have the resolution of 0.01 and the range of -327,68 to +327.67.

Display type	0	+1
Digital displays		
0	ON/OFF text value display	+ on/off buttons
Analog displays - various		
2	generic - no unit	+ bar graph
4	per second (s^{-1})	+ bar graph
6	per minute (min^{-1})	+ bar graph
8	rpm	+ bar graph
10	voltage in V	+ bar graph
12	current in mA	+ bar graph
14	current in A	+ bar graph
16	voltage in mV	+ bar graph
Temperature		
20	temperature in degrees C	+ bar graph
22	temperature in degrees F	+ bar graph
24	temperature in K	+ bar graph
30	relative humidity in % RH	+ bar graph
Power		
32	power in W	+ bar graph
34	power in kW	+ bar graph
36	power cons. In kWh	+ bar graph
Volume		
38	volume in cubic m	+ bar graph
40	volume in cubic feet	+ bar graph
42	volume in liters	+ bar graph
44	volume in gallon	+ bar graph
Weight		
50	weight in g	+ bar graph
52	weight in kg	+ bar graph
54	weight in t	+ bar graph
56	weight in oz	+ bar graph
58	weight in pound	+ bar graph
Time		
100	seconds	+ bar graph
102	minutes	+ bar graph
104	hours	+ bar graph
106	days	+ bar graph
Pressure		
120	pressure in kPa	+ bar graph
122	pressure in bar	+ bar graph
124	pressure in atm	+ bar graph

126	pressure in psi	+ bar graph
128	pressure in mm H_20	+ bar graph
130	pressure in Pa	+ bar graph

Setup dashboard items

- byte 2: 0x71
- byte 3: dashboard item ID (set bit 7 to set the configuration)
- byte 4-6: reserved
- byte 7: request ID

If bit 7 of byte 3 is set, the following values must be also set:

- byte 9-16: dashboard item label
- byte 17: dashboard item type (see above)
- byte 18: sensor ID (or pin ID for digital input/output or output index for PoExtBus)
- byte 19: dashboard item display type (see the second list above)
- byte 20: access rights (bit mapped user access list – see web interface users chapter)
- byte 21-22: minimal value for progress bar display (MSB first)
- byte 23-24: maximal value for progress bar display

Returned packet:

- byte 2: 0x71
- byte 3-6: reserved
- byte 7: request ID
- byte 9-16: dashboard item label
- byte 17: dashboard item type (see above)
- byte 18: sensor ID (or pin ID for digital input/output or output index for PoExtBus)
- byte 19: dashboard item display type (see the second list above)
- byte 20: access rights (bit mapped user access list – see web interface users chapter)
- byte 21-22: minimal value for progress bar display
- byte 23-24: maximal value for progress bar display

Dashboard items

PoKeys57 series devices support up to 100 dashboard items. Each item can be linked to one of the following supported data sources:

- 0: unused (inactive)
- 1: digital input
- 2: digital output
- ~~3: analog input~~
- 4: PoExtBus output
- 5: sensor input
- 6: digital counters
- 7: PoIL shared data

Display types

Display type	ID	Parameter 1	Parameter 2
Value display	0	Bit 0: show bar graph Bit 1: show slider Bit 2: show text entry	
On/Off display	10	Bit 0: show on/off Bit 1: show toggle button	
Gauge display	20		

All analog displays have the resolution of 0.01 and the range of -327,68 to +327.67.

Supported units

Unit ID	Unit display	Unit ID	Unit display
0-2	generic - no unit	10	voltage in V
4	per second (s^{-1})	12	current in mA
5	per second (1/s)		
6	per minute (min^{-1})	14	current in A
7	per minute (1/min)		
8	rpm	16	voltage in mV
Temperature			
20	temperature in degrees C	24	temperature in K
22	temperature in degrees F	30	relative humidity in %RH
		31	relative humidity in %
Power			
32	power in W	35	power cons. in Wh
34	power in kW	36	power cons. in kWh
Volume			
38	volume in cubic m	40	volume in cubic feet
39	volume flow in cubic m per second	41	volume flow in cubic feet per second
42	volume in liters	44	volume in gallon
43	volume flow in liters per second		
Weight			
50	weight in g	56	weight in oz
52	weight in kg	58	weight in pound
54	weight in t		
Time			
100	seconds	104	hours
102	minutes	106	days
Pressure			
120	pressure in kPa	126	pressure in psi

122	pressure in bar	124	pressure in atm
130	pressure in Pa	128	pressure in mm H_20
Custom units			
200-219	custom unit 0 - 19		

0x78/0x00 Dashboard item operation

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x78**
3	0x00 - dashboard item operation
4	Operation code: 0x00 - Read item 0x10 - Write item
5	Item index
6	Items num (1 or 2)
7	Request ID
8 (CRC)	CRC
9	Data source
10	Data source parameter
11	Display type
12	Unit
13	Display parameter 1
14	Display parameter 2
15	Access rights (bit mapped user access list – see web interface users chapter)
16-28	Label (13 bytes)
29-32	Minimum value (32-bit)
33-36	Maximum value (32-bit)
37-64	Repeated bytes 9-36 for second item if byte 6 set to 2

Dashboard item operation - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x78**
3	0x00 - dashboard item operation
4	Operation code: 0x00 - Read item 0x10 - Write item
5	Item index
6	Items num (1 or 2)
7	Request ID
8 (CRC)	CRC
9	Data source
10	Data source parameter
11	Display type
12	Unit
13	Display parameter 1
14	Display parameter 2
15	Access rights
16-28	Label (13 bytes)

29-32	Minimum value (32-bit)
33-36	Maximum value (32-bit)
37-64	Repeated bytes 9-36 for second item if byte 6 set to 2

0x78/0x80 Custom unit operation

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x78**
3	0x80 - custom unit operation
4	Operation code: 0x00 - Read unit 0x10 - Write unit
5	Unit index
6	Reserved
7	Request ID
8 (CRC)	CRC
9-16	Unit simple name (8 characters)
17-48	Unit HTML code (32 characters)
49-63	unused / reserved
64 (CRC 2)	CRC

Custom unit operation - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x78**
3	0x80 - custom unit operation
4	Operation code
5	Unit index
6	Reserved
7	Request ID
8 (CRC)	CRC
9-16	Unit simple name (8 characters)
17-48	Unit HTML code (32 characters)
49-63	unused / reserved
64 (CRC 2)	CRC

PoKeys EasySensors

Subsystem of PoKeys series 57 devices that automatically reads various sensors on I2C, 1-wire buses and analog inputs. Up to 100 sensors can be setup.

Supported sensor types

Sensor	Sensor bus	Sensor type	Reading IDs	Sensor resolution
LM75	I^2C	0x10	0 - temperature in °C	0.5 °C
SHT21	I^2C	0x11	0 - temperature in °C	0.01 °C
			1 - humidity in %RH	0.04 %RH
BME280	I^2C	0x12	0 - temperature in °C	0.01 °C
			1 - absolute pressure in Pa	0.01 Pa
			2 - humidity in %RH	0.01 Pa
MCP9600	I^2C	0x13	0 - hot-end abs. temperature in °C	0.0625 °C
			1 - cold-junction temperature in °C	0.0625 °C
			2 - temp. difference in °C	0.0625 °C
iAQ-Core C	I^2C	0x14	0 - Prediction of CO_2 level	1 ppm
			1 - Prediction of TVOC (volatile organic compound)	1 ppb
			2 - sensor status	/
DS18S20	(Dallas) 1-wire	0x18	0 - temperature in °C	0.0625 °C
DS18B20	(Dallas) 1-wire	0x19	0 - temperature in °C	0.0625 °C
DS2413	(Dallas) 1-wire	0x1A	0 - digital input PIOA	
			2 - digital input PIOB	
DS2450	(Dallas) 1-wire	0x1B	0 - A/D input A in µV	78 µV
			1 - A/D input B in µV	78 µV
			2 - A/D input C in µV	78 µV
			3 - A/D input D in µV	78 µV
DHT11	(DHTxx) 1-wire	0x20	0 - temperature in °C	1 °C
			1 - humidity in %RH	1 %RH
DHT21	(DHTxx) 1-wire	0x21	0 - temperature in °C	0.1 °C
			1 - humidity in %RH	0.1 %RH
DHT22	(DHTxx) 1-wire	0x22	0 - temperature in °C	0.1 °C
			1 - humidity in %RH	0.1 %RH
Si7020	I^2C	0x23	0 - temperature in °C	0.01 °C
			1 - humidity in %RH	0.04 %RH
BH1750	I^2C	0x40	0 - light intensity in lx	1 lx
Si1141	I^2C	0x41	0 - light intensity (visible) - indoor	
			1 - light intensity (visible) - outdoor	
			2 - light intensity (IR) - indoor	
			3 - light intensity (IR) - outdoor	
			10 - light reflection	
MCP3425	I^2C	0x50	0 - A/D input (1x gain) in µV	62.50 µV
			1 - A/D input (2x gain) in µV	31.25 µV
			2 - A/D input (4x gain) in µV	15.63 µV
			3 - A/D input (8x gain) in µV	7.81 µV
ADS1115	I^2C	0x51	bits 7-5: PGA setting (see ADS1115)	
			bist 4-3: reserved	
			bits 2-0: MUX setting (see ADS1115)	
MMA7660	I^2C	0x60	0 - acceleration in axis x	

			1 - acceleration in axis y	
			2 - acceleration in axis z	
LIS2DH12	I²C	0x61	0 - acceleration in axis x	
			1 - acceleration in axis y	
			2 - acceleration in axis z	
Analog	Analog input	0xF0	41 - analog input on pin 41	~0.8 mV
			42 - analog input on pin 42	~0.8 mV
			43 - analog input on pin 43	~0.8 mV
			44 - analog input on pin 44	~0.8 mV
			45 - analog input on pin 45	~0.8 mV
			46 - analog input on pin 46	~0.8 mV
			47 - analog input on pin 47	~0.8 mV

Analog sensors

Various analog sensors can be attached to PoKeys (to one of the analog input pins). The measured value is then transformed according to the following formula:

$$u = AD_{val} * \frac{A_{gain}}{4096} + A_{offset}$$

Where AD_val is a measurement of the analog-to-digital converter (a value between 0 and 4095), A_{gain} is gain (32-bit integer number) and A_{offset} is result offset (32-bit integer number).

Sensor failure (failsafe configuration)

If sensor is detected as being offline (the sensors has either failed or is disconnected), the various default values can be specified:

- Bit 7: set the value to 0x7FFFFFFF
- Bit 6: set the value to 0
- Bits 0-5: sensor timeout value (in s)

Sensor item data structure

Each sensor entry is described by 4 fields - sensor type, sensor refresh period, failsafe configuration, sensor reading ID and sensor ID. Sensor type field contains the 'Sensor type' value from the table above, refresh period is specified in 0.1 second intervals, failsafe configuration byte is described above, sensor reading ID is also given in the table above. Sensor ID depends on the sensor's bus type and is structured as follows.

I²C bus sensors

ID byte 1	ID byte 2	ID byte 3	ID byte 4	ID byte 5	ID byte 6	ID byte 7	ID byte 8
sensor I²C address	unused	unused	unused	unused	unused	unused	unused

(Dallas) 1-wire bus sensors

ID byte 1	ID byte 2	ID byte 3	ID byte 4	ID byte 5	ID byte 6	ID byte 7	ID byte 8
PoKeys pin	56 bits of 64-bit Dallas 1-wire sensor ROM ID (without family ID)						

(DHTxx) 1-wire bus sensors

ID byte 1	ID byte 2	ID byte 3	ID byte 4	ID byte 5	ID byte 6	ID byte 7	ID byte 8
PoKeys pin	unused	unused	unused	unused	unused	unused	unused

Analog sensors

ID byte 1	ID byte 2	ID byte 3	ID byte 4	ID byte 5	ID byte 6	ID byte 7	ID byte 8
A_{gain}				A_{offset}			

0x76 Read / write sensors setup (series 57)

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x76**
3	Sensor index (0 - 99)
4	Sensor count (1 - 4)
5	Read (0) or write (1)
6	Reserved
7	Request ID
8 (CRC)	CRC
9	Sensor type
10	Sensor reading ID
11	Refresh period
12	Failsafe configuration
13-20	Sensor ID
…	repeat bytes 9 to 20 for each additional sensor
x-63	unused / reserved (depends on sensor count, x between 21 and 57)
64 (CRC 2)	CRC

Read / write sensors setup - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x76**
3	Sensor index (0 - 99)
4	Sensor count (1 - 4)
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Sensor type
10	Sensor reading ID
11	Refresh period
12	Failsafe configuration
13-20	Sensor ID
…	repeat bytes 9 to 20 for each additional sensor
x-63	unused / reserved (depends on sensor count, x between 21 and 57)
64 (CRC 2)	CRC

0x77 Read sensors values (series 57)

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x77**
3	Sensor index (0 - 99)
4	Sensor count (1 - 13)
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	unused / reserved
64 (CRC 2)	CRC

Read / write sensors setup - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x77**
3	Sensor index (0 - 99)
4	Sensor count (1 - 13)
5-6	Bit-mapped sensor status information (bit 0 of byte 5 - first sensor)
7	Request ID
8 (CRC)	CRC
9	Sensor value (LSB first)
10	repeat bytes 9 to 12 for each additional sensor
11	unused / reserved (depends on sensor count, x between 13 and 61)
64 (CRC 2)	CRC

Server reports settings

In order to upload data to various web services, user must specify request header, server IP and update rate. The request header is constructed of two parts – HTML header and data header, divided by double new line character (\n). Example header (Cosm.com web service):

PUT /v2/feeds/62592.csv HTTP/1.1 Host: api.cosm.com X-ApiKey: CAb3XX634daSAKxZc3M0M3I1NWVZVT0g User-Agent: PoKeys56E Content-Type: text/csv Content-Length: 010 Connection: close Test1234,-15000.00	The HTTP header without 'Connection' and 'Content-length' tags must be provided by the user The 'Connection' and 'Content-length' tags are automatically inserted by PoKeys56E device Data is inserted at the end of the packet

The above example is saved as

```
PUT /v2/feeds/62592.csv HTTP/1.1
Host: api.cosm.com
X-ApiKey: CAb3XX634daSAKxZc3M0M3I1NWVZVT0g
User-Agent: PoKeys56E
Content-Type: text/csv
\n
```

The extra new line at the end is essential.

More customized header:

POST /myScript.php HTTP/1.1 Host: api.cosm.com User-Agent: MyCustomDeviceAgent Content-Type: text/csv Content-Length: 010 Connection: close MyData: Test1234,-15000.00	The HTTP header without 'Connection' and 'Content-length' tags must be provided by the user The 'Connection' and 'Content-length' tags are automatically inserted by PoKeys56E device

The above example is saved as

```
POST /myScript.php HTTP/1.1
Host: api.cosm.com
User-Agent: MyCustomDeviceAgent
Content-Type: text/csv
\n
MyData:
```

Again, an extra new line character in fifth line is essential.

Total length of final header and data is limited to 350 bytes.

Server reports settings

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xEF**
3	**Operation code**: 0 – read server IP, update rate, request type 1–5 – read request header (50 bytes per request)

	10 – set server IP, update rate, request type 11-15 – set request header (50 bytes per request)
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
Operation 10	
9-10	Update rate
11-14	Server IP
15	Request type bit 7:5: internal type reference (set to 0 if setting the fields manually) 0 - Cosm/Xively, 1 - POST, 2 - PUT, 3 - Custom, 4 - UDP mode bit 4: source selection 0 - sensor's values 1 - UART buffer bit 3:2: value type 0 - CSV, each variable in its own line, variable name and variable value delimited with comma in each line 1 - CSV, all variables in one line in format {variableName},{variable} 2 - same as 1 but with | as delimiter 3 - JSON format bit 1:0: protocol selection 0 - specified HTTP header 1 - RAW data
16-17	Server port number
18-63	Reserved
Operations 11-15	
9-58	Request header – 50 bytes (operation number selects page index)
59-63	Reserved
64 (CRC 2)	CRC

Server reports settings – Response

Byte	**Function**
1 (header)	Header (0xAA)
2 (CMD)	**0xEF**
3	Operation code
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
Operation 0	
9-10	Update rate
11-14	Server IP
15	Request type
16-17	Last status code
18-19	Server port number
20-63	Reserved
Operations 1-5	

9-58	Request header – 50 bytes (operation number selects page index)
59-63	Reserved
64 (CRC 2)	CRC

Pulse engine commands v2

Pulse engine v2 supports two different pulse generator modules:

- Internal: similar to basic Pulse engine, limited to 25 kHz pulse frequency at 3 channels, uses built-in circuitry and pins
- External: new in v2, limited to 125 kHz pulse frequency at 8 channels, requires external circuitry to deserialize the data to pulses
- Smart external

Fast outputs

- Axes with 'Fast output' option activated, have no motion capability, but can be used to drive an auxilary load quickly (laser cutter)

Constants used

Pulse engine states

```
peSTOPPED           = 0,
peINTERNAL          = 1,
peBUFFER            = 2,
peRUNNING           = 3,

peJOGGING           = 10,
peSTOPPING          = 11,

peHOME              = 20,
peHOMING            = 21,

pePROBECOMPLETE     = 30,
pePROBING           = 31,
pePROBEERROR        = 32,

peSTOP_LIMIT        = 100,
peSTOP_EMERGENCY    = 101
```

Axis states

```
axSTOPPED           = 0,
axREADY             = 1,
axRUNNING           = 2,
axBUFFER            = 5,

axHOME              = 10,
axHOMINGSTART       = 11,
axHOMINGSEARCH      = 12,
axHOMINGBACK        = 13,

axPROBED            = 14,
axPROBESTART        = 15,
axPROBESEARCH       = 16,

axERROR             = 20,
axLIMIT             = 30,
```

0x85/0x00 Get status (position, limits, home, …)

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x00 - Get status
4	Check byte
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x00 - Status message
4	Soft limit status (bit-mapped, one bit per axis)
5	States with enabled power
6	Limit override status
7	Request ID
8 (CRC)	CRC
9	Pulse engine enabled - number of enabled axes
10	Pulse engine initialized (activated) status
11	Pulse engine state
12	Safety charge pump enabled
13	Limit+ switch status (bit mapped, one bit per axis)
14	Limit- switch status (bit mapped, one bit per axis)
15	Home switch status (bit mapped, one bit per axis)
16	Pulse generator type bits 0-3: generator type (0 - 8ch external, 1 - 3ch internal, 2 - 8ch smart external) bits 4-5: reserved bit 6: swap step / dir signals bit 7: use external extended IO features (only for external pulse generator)
17-24	Axes status (8x 8-bit, values listed above)
25-56	Axes positions (8x 32-bit, LSB first)
57	Maximum number of axes supported
58	Maximum pulse frequency in kHz
59	Motion buffer size (number of available time slots)
60	Slot timing (in 100us steps)
61	Emergency switch polarity
62	External error inputs status (bit-mapped to axes)
63	External misc inputs status (bit 4 = input 5, bit 5 = input 6, bit 6 = input 7, bit 7 = spindle error input)
64 (CRC 2)	0x5A + check byte from request

0x85/0x01 Setup pulse engine

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x01 - Setup pulse engine
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9	Pulse engine enabled - number of enabled axes
10	Safety charge pump enabled (0 - disabled, 1 - default pin, otherwise 10-based pinID)
11	Pulse generator configuration bits 0-3: Pulse generator type (0 - 8ch external, 1 - 3ch internal) bits 4-5: reserved bit 6: swap step / dir signals bit 7: is used to enable or disable external inputs&outputs
12	Motion buffer size (0 - default value)
13	Emergency switch inverted polarity
14	States with enabled power (bit-mapped): bit 0: peSTOPPED bit 1: peSTOP_LIMIT bit 2: peSTOP_EMERGENCY States with enabled charge pump (bit-mapped) bit 4: peSTOPPED bit 5: peSTOP_LIMIT bit 6: peSTOP_EMERGENCY
15-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x01 - Setup pulse engine
4	Command result (0 - OK)
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

0x85/0x02 Set state

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x02 - Set state
4	Pulse engine state
5	Limit override option
6	Output enable mask (bit-mapped, 1 = enable axis output enable control), must be enabled in axis settings aoENABLED_MASKED
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x02 - Set state
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

0x85/0x03 Set axis position

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x03 - Set axis position
4	Axis position setup selection (bit-mapped)
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-40	Axis positions (32-bit LSB first)
41-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x03 - Set axis position
4	Reserved
5	Reserved
6	Reserved
7	Request ID

8 (CRC)	unused
9-63	reserved
64 (CRC 2)	CRC

0x85/0x04 Set outputs (using PoKeysCNCaddon)

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x04 - Set outputs
4	Relay outputs (bit 0 = relay 1, bit 1 = relay 2, bit 2 = relay 3)
5	OC outputs (bit 3 = OC 1, bit 4 = OC 2, bit 5 = OC 3, bit 6 = OC 4)
6	If set to 1, the above output values are not used - read function
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x04 - Set outputs
4	Relay outputs
5	OC outputs
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

0x85/0x04 Set outputs (using PoKeys57CNC)

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x04 - Set outputs
4	Reserved
5	bit 0: SSR 2 bit 1: Relay 2 bit 2: Relay 1 bit 3: OC 1 bit 4: OC 2 bit 5: OC 3 bit 6: OC 4 bit 7: SSR 1
6	If set to 1, the above output values are not used - read function
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x04 - Set outputs
4	Reserved
5	Outputs statuses as specified in the request
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

0x85/0x05 Reboot pulse engine

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x05 - Reboot pulse engine
4-6	reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x05 - Reboot pulse engine
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

0x85/0x06 Configure other parameters

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x06 - Configure other parameters
4	Write (1) or read (0) parameters
5-6	reserved
7	Request ID
8 (CRC)	CRC
9	Emergency switch input pin (0 - disabled, 1 - default, 10+ pin ID)
10-63	Reserved

64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x06 - Configure other parameters
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Emergency switch input pin (0 - disabled, 1 - default, 10+ pin ID)
10-63	reserved
64 (CRC 2)	unused

0x85/0x08 Get status 2

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x08 - Get status 2
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x08 - Status message 2
4	
5	
6	
7	Request ID
8 (CRC)	CRC
9	Status of dedicated limit- inputs
10	Status of dedicated limit+ inputs
11	Status of dedicated home inputs

64 (CRC 2)	0x5A + check byte from request

0x85/0x0A Setup synced PWM output

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x0A - setup synced PWM output
4	enabled flag (set to 1 to enable synched PWM output functionality, set to 2 to read the settings)
5	source axis ID (0 to 7)
6	destination PWM channel ID (0 to 5)
7	Request ID
8 (CRC)	CRC
9-64	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x0A - setup synced PWM output
4	enabled flag
5	source axis ID (0 to 7)
6	destination PWM channel ID (0 to 5)
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

0x85/0x10 Get axis configuration

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x10 - Get axis configuration
4	Axis ID
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x10 - Get axis configuration
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9	Axis options aoENABLED = (1<<0), aoINVERTED = (1<<1), aoINTERNAL_PLANNER = (1<<2), aoPOSITION_MODE = (1<<3), aoINVERTED_HOME = (1<<4), aoSOFT_LIMIT_ENABLED = (1<<5), aoFAST_OUTPUT = (1<<6), aoENABLED_MASKED = (1<<7)
10	Axis switch options aoSWITCH_LIMIT_N = (1<<0), aoSWITCH_LIMIT_P = (1<<1), aoSWITCH_HOME = (1<<2), aoSWITCH_COMBINED_LN_H = (1<<3), aoSWITCH_COMBINED_LP_H = (1<<4), aoSWITCH_INVERT_LIMIT_N = (1<<5), aoSWITCH_INVERT_LIMIT_P = (1<<6), aoSWITCH_INVERT_HOME = (1<<7)
11	Home switch input pin (0 for external)
12	Limit- switch input pin (0 for external)
13	Limit+ switch input pin (0 for external)
14	Homing speed (in % of the maximum speed)
15	Homing return (fine positioning) speed (in % of the homing speed)
16	MPG jog encoder ID
17-20	Axis maximum speed in timeslot units (32-bit float)
21-24	Axis maximum acceleration in timeslot units (32-bit float)
25-28	Axis maximum decelleration (braking) in timeslot units (32-bit float)
29-32	Soft-limit minimum position
33-36	Soft-limit maximum position

37-38	MPG jog multiplier
39	Axis enable output pin (0 for external)
40	Invert axis enable signal
41	Limit- switch filter setting (0-254)
42	Limit+ switch filter setting (0-254)
43	Home switch filter setting (0-254)
44	Home algorithm selection: Algorithm is based on handling two distinct events: - home switch activation (bits 0-3) - home switch release (bits 4-7) On each event, the following actions can be taken: 0000 do nothing 0001 slow down 0010 reverse 0011 reverse, slow-down 0100 arm encoder index 0101 slow down, arm encoder index 0110 reverse, arm encoder index 0111 reverse, slow-down, arm encoder index 1000 reset position, stop The values of 0 or 0xFF are invalid and default to 0x83 (reverse and slow-down on home switch activation, reset counter on home switch release)
45	MPG jog uses 1x resolution (added in 4.2.18)
46-63	reserved
64 (CRC 2)	unused

0x85/0x11 Set axis configuration

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x11 - Set axis configuration
4	Axis ID
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9	Axis options aoENABLED = (1<<0), aoINVERTED = (1<<1), aoINTERNAL_PLANNER = (1<<2), aoPOSITION_MODE = (1<<3), aoINVERTED_HOME = (1<<4), aoSOFT_LIMIT_ENABLED = (1<<5), aoENABLED_MASKED = (1<<7)
10	Axis switch options aoSWITCH_LIMIT_N = (1<<0), aoSWITCH_LIMIT_P = (1<<1), aoSWITCH_HOME = (1<<2), aoSWITCH_COMBINED_LN_H = (1<<3), aoSWITCH_COMBINED_LP_H = (1<<4), aoSWITCH_INVERT_LIMIT_N = (1<<5), aoSWITCH_INVERT_LIMIT_P = (1<<6),

	aoSWITCH_INVERT_HOME = (1<<7)
11	Home switch input pin (0 for external)
12	Limit- switch input pin (0 for external)
13	Limit+ switch input pin (0 for external)
14	Homing speed (in % of the maximum speed)
15	Homing return (fine positioning) speed (in % of the homing speed)
16	MPG jog encoder ID
17-20	Axis maximum speed in timeslot units (32-bit float)
21-24	Axis maximum acceleration in timeslot units (32-bit float)
25-28	Axis maximum decelleration (braking) in timeslot units (32-bit float)
29-32	Soft-limit minimum position
33-36	Soft-limit maximum position
37-38	MPG jog multiplier
39	Axis enable output pin (0 for external)
40	Invert axis enable signal
41	Limit- switch filter setting (0-254)
42	Limit+ switch filter setting (0-254)
43	Home switch filter setting (0-254)
44	Home algorithm selection
45	MPG jog uses 1x resolution (added in 4.2.18)
46-63	reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x11 - Set axis configuration
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

0x85/0x20 Move (Set reference position or speed)

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x20 - set reference position or speed
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-40	Reference position or speed (8x 32-bit integer, LSB first)
41-63	reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x20 - set reference position or speed
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

0x85/0x21 Start homing

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x21 - Start homing
4	Bit-mapped axis selection
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-40	Home position offsets (32-bit LSB first per axis)
41-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x21 - Start homing
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved

64 (CRC 2)	unused

0x85/0x22 Finish homing

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x22 - Finish homing
4	Pulse engine mode to transition into
5	Skip the pulse engine mode transition if this is set to > 0
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x22 - Finish homing
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

0x85/0x23 Start probing

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x23 - Start probing
4	Bit-mapped axis selection
5	Enable hybrid probing (ignore bytes 4 and 9-44)
6	Reserved
7	Request ID
8 (CRC)	CRC
9-40	Probe maximum position (32-bit LSB first per axis)
41-44	Probe speed (ration of the maximum speed - 0.01 equals to 1% of max. speed) - 32-bit float
45	Probe input (0 - disabled, 1-8 external inputs, 9+ PoKeys pin ID + 9)
46	Probe input polarity
47-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x23 - Start probing

4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

0x85/0x24 Finish probing

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x24 - Finish probing
4	Action (0 - normal finish, 1 - just clear probing state)
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x24 - Finish probing
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-40	Probe position
41	Probe status (probe completion bit-mapped status)
41-63	reserved
64 (CRC 2)	unused

0x85/0xF0 Clear motion buffer

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0xF0 - Clear buffer
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0xF0 - Clear buffer
4-6	Reserved

7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

0x85/0xFF Fill motion buffer - 8-bit mode (max. 127 steps per slot)

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0xFF - fill buffer: 8-bit mode
4	Number of new slots
5	Number of axes in buffer mode
6	Reserved
7	Request ID
8 (CRC)	CRC
9-64	Data (56 bytes) 8 axes = max. 7 slots 7 axes = max. 8 slots 6 axes = max. 9 slots 5 axes = max. 11 slots 4 axes = max. 14 slots 3 axes = max. 18 slots 2 axes = max. 28 slots 1 axis = max. 56 slots

Response (same as get status report, except for byte 3)

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	Number of accepted slots
4	Soft limit status (bit-mapped, one bit per axis)
5	States with enabled power
6	Limit override status
7	Request ID
8 (CRC)	CRC
9-63	See 'Get status' command report
64 (CRC 2)	unused

0xB0/0x85 Fill motion buffer via multi-part packet

Request - each packet in series of 8 packets

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xB0**
3	Multi-part index, flags **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** Reserved | Reserved | Reserved | Stop bit | Start bit | Packet index (0-7)
4 (only first packet)	0xFF - fill buffer: 8-bit mode 0xFE - fill buffer: 16-bit mode
5 (only first packet)	Number of new slots
6 (only first packet)	Number of axes in buffer mode
7	Request ID
8 (CRC)	CRC
9-64	Data (56 bytes) from each packet is combined into final buffer, which is then inserted into motion buffer

Response (same as get status report, except for byte 3)

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xB0**
3	Number of accepted slots
4	Soft limit status (bit-mapped, one bit per axis)
5	States with enabled power
6	Limit override status
7	Request ID
8 (CRC)	CRC
9-63	See 'Get status' command report
64 (CRC 2)	unused

0x85/0xE0 Transfer RAW data

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	Number of bytes in payload
4	Last part of data (1)
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-64	Payload (up to 56 bytes)

Response (same as get status report, except for byte 3)

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	Status
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-64 (CRC 2)	unused

0x85/0x90 Read smart pulse generator configuration

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x90 - Read smart pulse generator configuration
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x90 - Configure smart pulse generator
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-16	Pulse widths
17-20	Spindle PWM period
21	Encoder configurations
22	Encoder count mode
23-24	Step PWM period

25	Pulse engine enabled - number of enabled axes
26	Probe input configuration
27	Encoder debounce filter setting
28	Probe debounce filter setting
29	Step as GPIO
30	Dir as GPIO
31	Step as PWM
32-39	Step pulse multiplication factors
40-47	Encoder pulse multiplication factors
48-63	Maximum dismatch value
64 (CRC 2)	unused

0x85/0x91 Configure smart pulse generator

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x91 - Configure smart pulse generator
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-16	Pulse widths
17-20	Spindle PWM period
21	Encoder configurations
22	Encoder count mode
23-24	Step PWM period
25	Pulse engine enabled - number of enabled axes
26	Probe input configuration
27	Encoder debounce filter setting
28	Probe debounce filter setting
29	Step as GPIO
30	Dir as GPIO
31	Step as PWM
32-39	Step pulse multiplication factors
40-47	Encoder pulse multiplication factors
48-63	Maximum dismatch value
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x91 - Configure smart pulse generator
4	Command result (0 - OK)
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved

64 (CRC 2)	unused

0x85/0x92 Reset smart pulse generator counters

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x92 - Reset smart pulse generator counters
4	bit mapped axes registers to reset
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x92 - Reset smart pulse generator counters
4	Command result (0 - OK)
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

0x85/0x95 Read smart pulse generator status

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x95 - Read smart pulse generator status
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x95 - Read smart pulse generator status

4-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Axis position dismatch statuses
10-11	Spindle rotation period
12	Input statuses
13-28	Current dismatch values
29-60	Current position values
61-63	reserved
64 (CRC 2)	unused

0x85/0x96 Read smart pulse generator encoder positions

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x96 - Read smart pulse generator encoder positions
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x96 - Read smart pulse generator encoder positions
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-40	Current encoder positions
41-63	reserved
64 (CRC 2)	unused

0x85/0x30 Prepare trigger for threading

This commands tells PoKeys device to empty the motion buffer, switch to spindle-synchronized motion and pause.

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x30 - Prepare trigger
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

0x85/0x31 Force trigger prepare for threading

This commands tells PoKeys device to switch to spindle-synchronized motion and pause immediately.

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x31 - Force trigger prepare
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

0x85/0x32 Arm the trigger

PoKeys will resume motion on the next index impulse of the ultra fast encoder in synchronization with the spindle.

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x32 - Arm the trigger
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

0x85/0x33 Release the trigger

This commands tells PoKeys device to disable spindle synchronization. PoKeys device uses smooth transition back to normal operating speed.

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x33 - Release the trigger
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

0x85/0x34 Cancel threading mode

Cancel the threading mode - go back into normal operating mode, restart motion if still waiting for the trigger.

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x34 - Cancel threading mode
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

0x85/0x35 Get threading mode status

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x35 - Get threading mode status
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x35 - Get threading mode status
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Trigger preparring flag
10	Trigger prepared flag
11	Trigger waiting flag

12	Threading mode active
13-16	Estimated spindle velocity
17-20	Current rotor position error
21-24	Spindle RPM (sensor modes 2 and 3)
21-63	reserved
64 (CRC 2)	unused

0x85/0x36 Set threading mode parameters

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x36 - Set threading mode parameters
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Spindle sensor mode (0 - index signal, 1 - encoder+index signal, 2 - high-speed spindle index signal (milling), 3 - high-speed encoder (milling)) Test mode: bit 7 set to 1
13-14	Encoder ticks per rotation
15-16	Target spindle speed in rpm (revolutions per minute)
17-18	Spindle filter - speed error gain
19-20	Spindle filter - position error gain
21	Mask for axes ignoring the synchronisation
22-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x36 - Set threading mode parameters
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Trigger preparring flag
10	Trigger prepared flag
11	Trigger waiting flag
12	Threading mode active
13-16	Estimated spindle velocity
17-20	Current rotor position error
21-24	Spindle RPM (sensor modes 2 and 3)
25	Mask for axes ignoring the synchronisation
26-63	reserved
64 (CRC 2)	unused

0x85/0x37 Get encoder test mode results

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x37 - Get encoder test mode results
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x37 - Get encoder test mode results
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-12	Current encoder position
13-16	Current encoder index count
17-20	Last encoder rotation tick count
21-24	Current encoder speed
25-63	reserved
64 (CRC 2)	unused

Backlash compensation

PoKeys Pulse engine v2 (PoKeys57 firmware 4.1.26 onwards) supports backlash compensation. For each direction change, 2*W (W is half of backlash width in pulses) of pulses is additionally generated using the additional motion profile, that is configured using the acceleration setting only.
PoKeys device holds an additional register for backlash compensation that is transparent to the application using the Pulse engine (the value of the register is available in 0x85/0x40 command).
The acceleration value (integer) specifies the motion profile acceleration in pulses per ms^2 (multiply this value by 10^6 to obtain the acceleration in pulses per s^2).

0x85/0x40 Get backlash compensation settings

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x40 - Get backlash compensation settings
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x40 - Get backlash compensation settings
4	Backlash compensation enabled
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-10	Half of backlash width (16-bit, LSB first) - axis 1
11	Backlash acceleration (8-bit) - axis 1
12	Reserved - axis 1
13-40	Backlash settings for other axes (same structure as above)
41-56	State of the backlash compensation for each axis (signed 16-bit per axis)
57-63	reserved
64 (CRC 2)	unused

0x85/0x41 Set backlash compensation settings

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x41 - Set backlash compensation settings
4	Backlash compensation enabled
4-6	Reserved
7	Request ID

8 (CRC)	CRC
9-10	Half of backlash width (16-bit, LSB first) - axis 1
11	Backlash acceleration (8-bit) - axis 1
12	Reserved - axis 1
13-40	Backlash settings for other axes (same structure as above)
41-63	reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x41 - Set backlash compensation settings
4	Backlash compensation enabled
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-10	Half of backlash width (16-bit, LSB first) - axis 1
11	Backlash acceleration (8-bit) - axis 1
12	Reserved - axis 1
13-40	Backlash settings for other axes (same structure as above)
41-56	State of the backlash compensation for each axis (signed 16-bit per axis)
41-63	reserved
64 (CRC 2)	unused

PoStep driver interface

PoKeys Pulse engine v2 (PoKeys57 firmware 4.1.42 onwards, limited to PoKeys57CNC) can talk directly to selected PoStep drivers (PoStep60-256) to bring the live driver status and abiltiy to configure the driver directly via PoKeys device.
Drivers and PoKeys device have to be connected via a PoExtension cable (using 10-pin red connector). Multiple PoStep devices can be connected to PoKeys57CNC device by a means of 'hopping' connector (PoExtension connectors are pressed onto multiple positions along the 10-pin flat cable - all connected in parallel).
Each PoStep driver must initially be assigned with a unique I2C address using the PoStep configuration application.
The following status is available for each PoStep driver:

- Supply voltage
- Driver temperature
- Inputs statuses (bit-mapped)
    - Bit 0: Bootloader override
    - Bit 1: Sleep
    - Bit 2: Step / AIN1
    - Bit 3: Direction / AIN2
    - Bit 4: BIN1
    - Bit 5: BIN2
    - Bit 6: End switch
    - Bit 7: reserved
- Driver status (status value as below)
    - 1: Driver in sleep mode
    - 2: Driver active
    - 3: Driver in idle mode
    - 4: Driver overheated
    - 5: Driver in DC motor control mode
- Faults status (bit-mapped)
    - Bit 0: OTS - Device has entered over temperature shutdown. OTS bit will clear once temperature has fallen to safe levels
    - Bit 1: AOCP - Channel A overcurrent shutdown. Check wiring or possible short circuit.
    - Bit 2: BOCP - Channel B overcurrent shutdown. Check wiring or possible short circuit.
    - Bit 3: APDF - Channel A predriver fault. Check driver settings.
    - Bit 4: BPDF - Channel B predriver fault. Check driver settings.
    - Bit 5: UVLO - Power supply voltage too low. Bit clears after voltage rises above lower limit.
    - Bit 6: STD - Stall detected.
    - Bit 7: STDLAT - Latched stall detect.

The following configuration can be changed:

- Driver mode:
    - 1 Default mode – external control
    - 2 Step control
    - 3 DC motor control
    - 4 Position control
    - 5 BINx buttons
- Step mode:
    - 0 Full step
    - 1 Half step
    - 2 1/4 step
    - 3 1/8 step
    - 4 1/16 step
    - 5 1/32 step
    - 6 1/64 step
    - 7 1/128 step
    - 8 1/256 step

- Temperature limit
- Full-scale current
- Idle current
- Overheat current

The PoKeys device communicates with PoStep devices asynchronously with PC - PoKeys device communication:

1. A instruction is given to PoKeys device to fetch or configure selected parameter
2. PoKeys executed the instruction when possible
3. Retrieved parameter's value is available in PoKeys device

Driver update configuration

PoKeys device can be configured to fetch selected driver status information automatically.

0x85/0x50 Setup driver communication

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x50 - Setup driver communication
4	Command (0 - read configuration, 0x10 - set configuration)
5	Enable driver communication
6	Reserved
7	Request ID
8 (CRC)	CRC
9	Axis 1 driver type
10	Axis 1 driver I2C address
11	Axis 1 driver update configuration Bits 6-7: operation mode (0 - normal, 2 - write) Bit 5: Retrieve all parameters mode Bits 0-4: Parameters status bits each is bit-mapped to indicate which parameter should be retrieved (normal mode) bit 0: supply voltage bit 1: driver's temperature bit 2: input pin status bit 3: driver's status bit 4: faults status Bits 0-4 in write mode: - 6: setup step mode - 7: setup temperature limit - 8: setup full-scale current - 9: setup idle current - 10: setup overheat current - 11: Save settings to EEPROM
12-32	Axes 2-8 driver types, I2C addresses and update configurations
33-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function

1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x50 - Setup driver communication
4	Command
5	Enable driver communication status
6	Reserved
7	Request ID
8 (CRC)	CRC
9	Axis 1 driver type
10	Axis 1 driver I2C address
11	Axis 1 driver update configuration
12-32	Axes 2-8 driver types, I2C addresses and update configurations
33-63	Reserved
64 (CRC 2)	unused

0x85/0x51 Get drivers' statuses

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x51 - Get drivers' statuses
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x51 - Get drivers' statuses
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Axis 1: Supply voltage (in 0.25 V / unit)
10	Axis 1: Temperature (degrees C)
11	Axis 1: Input status
12	Axis 1: Driver status
13	Axis 1: Fault status
14	Axis 1: Update counter
15-56	Statuses from bytes 9-14 repeated for axes 2 - 8
57-63	reserved
64 (CRC 2)	unused

0x85/0x52 Get/Set driver's current parameters

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x52 - Get/Set driver's parameters
4	Command (0 - read configuration, 0x10 - set configuration)
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-10	Full-scale current setting - axis 1 driver
11-12	Idle current setting - axis 1 driver
13-14	Overheat current setting - axis 1 driver
15-56	Settings for axes 2-8, with the structure as above
57-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x52 - Get/Set driver's parameters
4	Command
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-10	Full-scale current setting - axis 1 driver
11-12	Idle current setting - axis 1 driver
13-14	Overheat current setting - axis 1 driver
15-56	Settings for axes 2-8, with the structure as above
57-63	Reserved
64 (CRC 2)	unused

0x85/0x53 Get/Set driver's mode parameters and temperature limit

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x53 - Get/Set driver's mode parameters
4	Command (0 - read configuration, 0x10 - set configuration)
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Driver mode setting - axis 1 driver
10	Driver step mode setting - axis 1 driver
11	Driver temperature limit setting - axis 1 driver
12-14	reserved
15-56	Settings for axes 2-8, with the structure as above
57-63	Reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x53 - Get/Set driver's mode parameters
4	Command
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Driver mode setting - axis 1 driver
10	Driver step mode setting - axis 1 driver
11	Driver temperature limit setting - axis 1 driver
12-14	reserved
15-56	Settings for axes 2-8, with the structure as above
57-63	Reserved
64 (CRC 2)	unused

0x85/0x54 Get drivers' HW and FW versions

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x54 - Get drivers' versions
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x54 - Get drivers' versions
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Axis 1: HW version major
10	Axis 1: HW version minor
11	Axis 1: FW version major
12	Axis 1: FW version minor
13	reserved
14	reserved
15-56	Versions from bytes 9-14 repeated for axes 2 - 8
57-63	reserved
64 (CRC 2)	unused

0x85/0xFF Control output enable

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0xFF - Control output enable
4	Disable (0), enable (1)
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	unused

Failsafe mode

This settings give an option to set the outputs of the device to a predefined state after a preset communication timeout occurs.

Basic settings

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x81**
3	Read (0) / Write (1) failsafe configuration
4	Failsafe timeout (100 ms resolution) – 0 disables failsafe
5	Enabled failsafe peripherals **Bit 7**: Reserved **Bit 6**: Reserved **Bit 5**: Reserved **Bit 4**: Reserved **Bit 3**: Pulse engine **Bit 2**: PWM **Bit 1**: PoExtBus **Bit 0**: Digital IO
6	Reserved
7	Request ID
8 (CRC)	CRC
9-15	Bit-mapped digital outputs state
16-25	Bit-mapped PoExtBus outputs state
26-31	PWM outputs duties (0-100 %)
32-63	Reserved
64 (CRC 2)	CRC

Basic settings - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x81**
3	Read (0) / Write (1) failsafe configuration request value
4	Failsafe timeout (100 ms resolution) – 0 disables failsafe
5	Enabled failsafe peripherals **Bit 7**: Reserved **Bit 6**: Reserved **Bit 5**: Reserved **Bit 4**: Reserved **Bit 3**: Pulse engine **Bit 2**: PWM **Bit 1**: PoExtBus **Bit 0**: Digital IO
6	Reserved
7	Request ID
8 (CRC)	CRC
9-15	Bit-mapped digital outputs state
16-25	Bit-mapped PoExtBus outputs state
26-31	PWM outputs duties (0-100 %)
32-63	Reserved
64 (CRC 2)	CRC

PoIL commands

See PoIL.pdf for description of the PoIL core.

0x82/0x00 PoIL core status

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x82**
3	0x00 – get PoIL core status
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	CRC

PoIL core status - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x82**
3	0x00 – get PoIL core status
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-10	Code memory size
11-12	Data memory size
13	Version
14-15	Current PoIL core state
16	PoIL core debug mode
17	Current task
18-19	PoIL core debug breakpoint
20-21	Current PC value
22	Current STATUS register value
23-26	Current W register value
27-28	Exception PC
29-30	Current function stack pointer
31-32	Current data stack pointer
60	PoIL master enable status
61	Number of PoIL tasks
33-63	reserved
64 (CRC 2)	CRC

0x82/0x01 Set PoIL core state

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x82**
3	0x01 – set PoIL core state
4-5	PoIL core state
6	Reserved
7	Request ID

8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	CRC

Response contents is similar to »PoIL core status – Response«.

0x82/0x02 Reset PoIL processor

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x82**
3	0x02 – Reset processor
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	CRC

Response contents is similar to »PoIL core status – Response«.

0x82/0x03 Set PoIL master enable status

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x82**
3	0x03 – set PoIL master enable
4	PoIL master enable value
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	CRC

Response contents is similar to »PoIL core status – Response«.

0x82/0x05 Set PoIL debug mode

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x82**
3	0x05 – set PoIL core debug mode
4	PoIL core debug mode 0 - debug STOP mode 10 - running to breakpoint 11 - run to breakpoint 20 - one instruction
5-6	Breakpoint address
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	CRC

Response contents is similar to »PoIL core status – Response«.

0x82/0x10 PoIL memory read

This command reads a chunk of up to 55 bytes from the selected memory

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x82**
3	0x10 – PoIL memory read
4	Memory selection (0 – code, 1 – data, 2 – code stack, 3 – data stack, 4 - shared data, 5 - internal addressing used)
5-6	Memory address (or shared data index)
7	Request ID
8 (CRC)	CRC
9	Memory chunk size
10-63	Reserved
64 (CRC 2)	CRC

PoIL memory read - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x82**
3	0x10 – read memory
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Memory contents
64 (CRC 2)	CRC

0x82/0x11 PoIL memory read – monitor mode

This command reads multiple chunks of up to 55 bytes from the selected memory (only data memory allowed)

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x82**
3	0x11 – PoIL memory read (monitor)
4	Memory selection (1 – data, 5 - internal addresssing, other sources not allowed)
5-6	reserved
7	Request ID
8 (CRC)	CRC
9-10	Chunk 1 address
11	Chunk 1 size
12-13	Chunk 2 address (set to FFFF to stop reading
14	Chunk 2 size
…	
60-61	Chunk 18 address
62	Chunk 18 size
63	Reserved
64 (CRC 2)	CRC

PoIL memory read - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x82**
3	0x11 – PoIL memory read (monitor)
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Memory contents by chunks of data
64 (CRC 2)	CRC

0x82/0x15 PoIL memory write

This command writes a chunk of up to 54 bytes to the selected memory. If code memory is selected, data memory locations 0-255 are used as data source and must be filled with correct data prior to this call. Code memory write always writes 256 bytes at a time.

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x82**
3	0x15 – PoIL memory write
4	Memory selection (0 – code, 1 – data, 4 - shared data, 5 - internal addressing used)
5-6	Memory address (or shared data index)
7	Request ID
8 (CRC)	CRC
9	Memory chunk size
10-63	Memory contents
64 (CRC 2)	CRC

PoIL memory write - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x82**
3	0x15 – write memory
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	CRC

0x82/0x16 PoIL memory erase

This command erases the selected PoIL memory

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x82**
3	0x16 – PoIL memory erase
4	Memory selection (0 – code, 1 – data)
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	CRC

PoIL memory erase - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x82**
3	0x16 – erase memory
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	CRC

0x82/0x20 PoIL task status read

This command retrieves the PoIL task statuses

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x82**
3	0x20 – Read PoIL task status
4	First task index (t)
5	Number of tasks to incldude (max. 7, depends on system)
6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	CRC

Read PoIL task status - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x82**
3	0x20 – Read PoIL task status
4	System inactive load
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9+t*8	Task status
10+t*8	Task load
11-12 +t*8	Configured task period
13-14 +t*8	Real task period - current
15-16 +t*8	Real task period - filtered

0x82/0x30 Get custom device status

This function is used to combine different data in one customizable request. The request contains a list of commands and parameters that tell the device to either set specific register value or retrieve one. It uses the addressing space of the PoIL core and supports bit, byte, word and double word access to these registers.
The supported commands are **set** and **get register**.
The command code is packaged into the top 2 bits of the 16-bit PoIL address space, with the next two bits selecting the access width.

Command header structure

Bits 15-14	Bits 13-12	Bits 11-0
Command code: - 00 set - 01 get - 10 reserved - 11 end of list	Data type: - 00 bit - 01 byte - 10 word (16-bit) - 11 dword (32-bit)	Peripheral address (see PoIL specifications)

After the header, command data follow. Command data is either 2 bytes (bit, byte and word types) or 4 bytes (double-word). Data is always aligned to LSb. Reserved commands are ignored, 2 additional bytes are skipped (total of 4).

Set register command

The command contains the header (2 bytes) and the data value (2 or 4 bytes, depending on data type). The provided value is written to the specified peripheral address when the packet is received and parsed. No data is returned.

Get register command

The command contains only the header (2 bytes), but retrieves the value of the register at the specified peripheral address.

End of list command

When this command is encountered, the device stops parsing the list of commands.

Get custom device status - Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x82**
3	0x30 – get custom device status
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-62	A list of command headers and data Header 1 + Data 1 (set register) Header 2 (get register) Header 3 + Data 3 (set register) Header 4 (get register) Header 5 (get register) Header 6 + Data 6 (set register) …
64 (CRC 2)	CRC

Get custom device status - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x82**
3	0x30
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	A list of requested values Data 2 Data 4 Data 6 …
64 (CRC 2)	CRC

RTC (Real Time Clock) setup

These commands are used to read or set the RTC clock, running in the device.

0x83 Read/Set current time

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x83**
3	0x00 – read time, 0x10 – set time, 0x11 - set time and daylight saving time options
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Second (0-59)
10	Minute (0-59)
11	Hour (0-23)
12	Day of week (0-6)
13	Day of month (1-31)
14	Month (1-12)
15-16	Day of year (1 – 365)
17-18	Year
19	Daylight saving time rule: 0: disabled 1: manually enabled 10: EU (from 01:00 UTC of last Sunday in March to 01:00 UTC of last Sunday in October) 20: US (from 02:00 AM of second Sunday in March to 02:00 AM of first Sunday in November) 30: AU (from 02:00 AM of first Sunday in November to 02:00 AM of first Sunday in April)
20	UTC offset for EU DST (signed 8-bit number) - supported values between -1 and 2
21-63	Reserved
64 (CRC 2)	CRC

Read/set time - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x83**
3	Same as byte 3 in request
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9	Second (0-59)
10	Minute (0-59)
11	Hour (0-23)
12	Day of week (0-6)
13	Day of month (1-31)
14	Month (1-12)
15-16	Day of year (1 – 365)
17-18	Year
19	Daytime saving time rule: 0: disabled 1: manually enabled 10: EU (from 01:00 UTC of last Sunday in March to 01:00 UTC of last Sunday in October) 20: US (from 02:00 AM of second Sunday in March to 02:00 AM of first Sunday in November) 30: AU (from 02:00 AM of first Sunday in November to 02:00 AM of first Sunday in April)

20	UTC offset for EU DST (signed 8-bit number) - supported values between -1 and 2
21-63	Reserved
64 (CRC 2)	CRC

System event logging

0x84 Read/clear system log

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x84**
3	0x00 – read log, 0x10 – clear log
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	CRC

Read/set time - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x84**
3	Total number of log entries
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-62	27 Log entries (LSB first)
63	Reserved
64 (CRC 2)	CRC

PPM decoder configuration and status

This command configures the PPM decoding option in PoKeys devices. PPM signal consist of series of pulses that encode the position information of up to 8 R/C servos. PoKeys detects the synchronisation pulse and extracts the commanded positions of each servo. This feature is available on PoKeys57U devices on pin 3 and on PoKeys57E devices on pin 24.

Unsupported devices: PoKeys55, PoKeys56 series

0x09 PPM decoder configuration and status

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x09**
3	0x00 – read configuration and status, 0x10 – set configuration
4	Configuration: bit 0: enable PPM decoding; bit 1: positive edge decoding
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	CRC

Read/set time - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x09**
3	Configuration - see request header above
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-10	Decoded PPM data age in 0.1 ms
11-26	Decoded PPM data (8 channels, 2 bytes each, LSB first)
63	Reserved
64 (CRC 2)	CRC

MCS (Multi Channel Servo) system configuration and control

PoKeys57 series devices support MCS system to generate PWM signals for R/C servo motors. The number of supported channels depends on the device type, but is typically limited by the number of digital output pins.

Each MCS channel can be configured with maximum acceleration and maximum velocity values, that are used for generating the changes of the PWM duty cycle values (if maximum velocity is set to 0, no motion planning is executed). The duty cycle of the PWM outputs is specified in microseconds, while the system operates at the resolution of 2 us.

MCS generates PWM signals in pulse batches of at least 10 outputs at a time (the exact number depends on the values of the PWM signals), so the system needs up to 6 pulse batches (each taking 2.5 ms to execute) in order to update all outputs. This results in minimum update period of 15 ms for all 55 outputs. Each PWM channel supports duty cycles between 100 and 2300 us.

Supported devices: PoKeys57E, PoKeys57U

0x4A/0x00 Read MCS status

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x4A**
3	0x00 – read status
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	CRC

Read MCS status - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x4A**
3	0x00
4	MCS global enable
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	MCS chanels statuses
64 (CRC 2)	CRC

0x4A/0x01 Set MCS status

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x4A**
3	0x01 – set status
4	MCS global enable
5-6	Reserved
7	Request ID

8 (CRC)	CRC
9-63	MCS chanels statuses
64 (CRC 2)	CRC

Set MCS status - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x4A**
3	0x00
4	MCS global enable
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	MCS chanels statuses
64 (CRC 2)	CRC

0x4A/0x10 Get MCS channel parameters

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x4A**
3	0x10 – get MCS channel parameters
4	Channel ID
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	reserved
64 (CRC 2)	CRC

Get MCS channel parameters - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x4A**
3	0x10
4	Channel ID
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-12	Maximum velocity (pulse us / period)
13-16	Maximum acceleration (pulse us / period ^{2})
17-18	Minimum position (pulse us)
19-20	Maximum position (pulse us)
21-22	Default position (pulse us)
23	Failsafe / startup configuration (0 - Off, 1 - On, 2 - Position)
24-63	reserved
64 (CRC 2)	CRC

0x4A/0x11 Set MCS channel parameters

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x4A**
3	0x11 – set MCS channel parameters
4	Channel ID
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-12	Maximum velocity (pulse us / period)
13-16	Maximum acceleration (pulse us / period ^{2})
17-18	Minimum position (pulse us)
19-20	Maximum position (pulse us)
21-22	Default position (pulse us)
23	Failsafe / startup configuration (0 - Off, 1 - On, 2 - Position)
24-63	reserved
64 (CRC 2)	CRC

Set MCS channel parameters - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x4A**
3	0x11
4	Channel ID
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-12	Maximum velocity (pulse us / period)
13-16	Maximum acceleration (pulse us / period ^{2})
17-18	Minimum position (pulse us)
19-20	Maximum position (pulse us)
21-22	Default position (pulse us)
23	Failsafe / startup configuration (0 - Off, 1 - On, 2 - Position)
24-63	reserved
64 (CRC 2)	CRC

0x4A/0x20 Read MCS rough positions

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x4A**
3	0x20 – read rough positions
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	CRC

Read MCS rough positions - Response

Byte	Function
1 (header)	Header (0xAA)

2 (CMD)	**0x4A**
3	0x20
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	MCS chanels positions (0-255 in the specified minimum to maximum range)
64 (CRC 2)	CRC

0x4A/0x21 Set MCS rough positions

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x4A**
3	0x21 – set rough positions
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	MCS chanels positions (0-255 in the specified minimum to maximum range)
64 (CRC 2)	CRC

Set MCS rough positions - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x4A**
3	0x21
4-6	Reserved
7	Request ID
8 (CRC)	CRC
9-63	Reserved
64 (CRC 2)	CRC

CAN operations

0x86/0x01 Configure CAN

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x86**
3	0x01 – configure CAN
4	
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-12	Bitrate (32-bit integer) - if set to 0, CAN is disabled
13-16	
17-18	
19-20	
21-22	
23	
24-63	reserved
64 (CRC 2)	CRC

Configure CAN - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x86**
3	0x01
4	
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-12	
13-16	
17-18	
19-20	
21-22	
23	
24-63	reserved
64 (CRC 2)	CRC

0x86/0x10 Register CAN filter

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x86**
3	0x10 – register CAN filter
4	CAN message format
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-12	CAN ID
13-16	
17-18	
19-20	
21-22	
23	
24-63	reserved
64 (CRC 2)	CRC

Register CAN filter - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x86**
3	0x10
4	
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-12	
13-16	
17-18	
19-20	
21-22	
23	

24-63	reserved
64 (CRC 2)	CRC

0x86/0x20 Send CAN message

```
unsigned int   id;              /* 29 bit identifier */
unsigned char  data[8];           /* Data field */
unsigned char  len;             /* Length of data field in bytes */
unsigned char  format;            /* 0 - STANDARD, 1- EXTENDED IDENTIFIER */
unsigned char  type;            /* 0 - DATA FRAME, 1 - REMOTE FRAME */
```

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x86**
3	0x20 – Send CAN message
4	
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-12	CAN ID
13-20	data byte
21	number of bytes
22	format
23	frame type
24-63	reserved
64 (CRC 2)	CRC

Send CAN message - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x86**
3	0x20
4	
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-12	
13-16	
17-18	
19-20	
21-22	
23	
24-63	reserved
64 (CRC 2)	CRC

0x86/0x30 Receive CAN message

```
unsigned int   id;              /* 29 bit identifier */
unsigned char  data[8];           /* Data field */
unsigned char  len;             /* Length of data field in bytes */
unsigned char  format;            /* 0 - STANDARD, 1- EXTENDED IDENTIFIER */
unsigned char  type;            /* 0 - DATA FRAME, 1 - REMOTE FRAME */
```

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x86**
3	0x30 – Read CAN message
4	
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-12	
13-20	
21	
22	
23	
24-63	reserved
64 (CRC 2)	CRC

Read CAN message - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x86**
3	0x30
4	status -
5-6	Reserved
7	Request ID
8 (CRC)	CRC
9-12	CAN ID
13-20	data byte
21	number of bytes
22	format
23	frame type
24-63	reserved
64 (CRC 2)	CRC

Old commands

Sensor list (only for PoKeys56E/PoKeys56U)

Subsystem of PoKeys56E that automatically reads various sensors on I2C and 1-wire buses. It supports up to 10 I2C (IDs from 0 to 9) and up to 10 1-wire sensors (IDs from 10 to 19) and up to 7 analog sensors (IDs from 20 to 26).

Supported sensor types (I2C):

- 0x10: LM75 temperature sensor
- 0x20: SHT21 temperature
- 0x21: SHT32 humidity
- 0x30: MCP3425 with PGA = 1
- 0x31: MCP3425 with PGA = 2
- 0x32: MCP3425 with PGA = 4
- 0x33: MCP3425 with PGA = 8

Supported sensor types (1-wire):

- 0x28: DS18B20

Analog sensors

Analog sensors can be attached to PoKeys to a properly configured analog input pin. The measured value is then transfored according to the following formula:

$$u = AD_{val} * \frac{A_{gain}}{4096} + A_{offset}$$

Where AD_val is a measurement of the analog-to-digital converter (a value between 0 and 4095), A_{gain} is gain (16-bit integer number) and A_{offset} is result offset (16-bit integer number).

Handling off-line sensors:

If sensor is detected as being offline, the various default values can be specified:

- Bit 7: set the value to 0xFFFFFFFF
- Bit 6: set the value to 0
- Bits 0-5: sensor timeout value (in s)

Setup sensors

- byte 2: 0x70
- byte 3: sensor ID (set bit 7 to set the configuration)
- byte 4: offline sensor configuration byte (see above for description)
- byte 5-6: reserved
- byte 7: request ID

For each sensor connection type:

Sensor IDs 0 to 9 (I2C sensor):

- byte 9: Sensor type (see above list of supported I2C sensors)
- byte 10: I2C address of the sensor
- byte 11: Refresh period in 100 ms

Sensor IDs 10 to 19 (1-wire sensor):

- bytes 9-16: 64-bit 1-wire identifier
- byte 17: Refresh period in 100 ms

Sensor IDs 20 to 26 (analog sensor):

- byte 9: analog channel (pin ID 40-46)
- bytes 10-13: gain (MSB first) * remark 25.11.2012: changed to 32-bit signed integers
- bytes 14-17: offset (MSB first) * remark 25.11.2012: changed to 32-bit signed integers

Returned packet:

- byte 2: 0x70
- byte 3: reserved
- byte 4: offline sensor configuration byte (see above for description)
- byte 5-6: reserved
- byte 7: request ID

For each sensor connection type:

Sensor connection type 0 (I2C sensor):

- byte 9: Sensor type (see above list of supported I2C sensors)
- byte 10: I2C address of the sensor
- byte 11: Refresh period in 100 ms
- bytes 20-23: Sensor value (MSB first)

Sensor connection type 1 (1-wire sensor):

- bytes 9-16: 64-bit 1-wire identifier
- byte 17: Refresh period in 100 ms
- bytes 20-23: Sensor value (MSB first)

Sensor connection type 2 (analog sensor):

- byte 9: analog channel (pin ID 40-46)
- bytes 10-13: gain (MSB first)
- bytes 14-17: offset (MSB first)
- bytes 20-23: Sensor value (MSB first)

Read all sensors

- byte 2: 0x74
- byte 3: page index (0-2), sensor index (100-103)
- byte 4-6: reserved
- byte 7: request ID

Returned packet:

- byte 2: 0x74
- byte 3-6: reserved
- byte 7: request ID

Page indexes 0-1:

- bytes 9-12: sensor (index * 14 + 1) value (MSB first)
- bytes 13-16: sensor (index * 14 + 2) value (MSB first)

...

- bytes 57-60: sensor (index * 14 + 13) value (MSB first)
- bytes 61-64: sensor (index * 14 + 14) value (MSB first)

Sensor indexes 100-103:

- bytes 9-16: sensor ((index-100) * 7 + 1) 64-bit ID

...

Cosm support settings (deprecated in 3.0.39 firmware).

In order to upload data to Cosm web service, user must specify Cosm API key and Feed ID strings. Also, Cosm server IP and update rate can be selected.

Cosm settings

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xEF**
3	**Operation code**: 0 – read general Cosm configuration 1 – read first 50 bytes of Cosm API key 2 – read second 50 bytes of Cosm API key 10 – set general Cosm configuration 11 – set first 50 bytes of Cosm API key 12 – set second 50 bytes of Cosm API key
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
Set general Cosm configuration	
9	Cosm update rate
10-13	Cosm server IP
14-23	Cosm Feed ID
24-63	Reserved
Set first 50 bytes of Cosm API key	
9-58	Cosm API key – first 50 bytes
59-63	Reserved
Set second 50 bytes of Cosm API key	
9-58	Cosm API key – second 50 bytes
59-63	Reserved
64 (CRC 2)	CRC

Basic settings – Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xEF**
3	Operation code
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (CRC)	CRC
Read general Cosm configuration	
9	Cosm update rate
10-13	Cosm server IP
14-23	Cosm Feed ID
24	Last status code
25-63	Reserved
Read first 50 bytes of Cosm API key	

9-58	Cosm API key – first 50 bytes
59-63	Reserved
Read second 50 bytes of Cosm API key	
9-58	Cosm API key – second 50 bytes
59-63	Reserved
64 (CRC 2)	CRC

Pulse engine commands **(not supported since FW version 3.0.66)**

(this module must be activated before first use with the activation code supplied by the manufacturer)

Constants used

Pulse engine states

peSTOPPED	= 0,
peINTERNAL	= 1,
peBUFFER	= 2,
peJOGGING	= 10,
peSTOPPING	= 11,
peHOME	= 20,
peHOMING	= 21,
peSTOP_LIMIT	= 100,
peSTOP_EMERGENCY	= 101

Axis states

axSTOPPED	= 0,
axREADY	= 1,
axRUNNING	= 2,
axHOME	= 10,
axHOMINGSTART	= 11,
axHOMINGSEARCH	= 12,
axHOMINGBACK	= 13,
axERROR	= 20,
axLIMIT	= 30,

Get status (position, limits, home, ...)

- byte 2: 0x80
- byte 3: 0x00
- byte 4: 0
- byte 5: 0
- byte 6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x80
- byte 3: 0x00
- byte 4-6: 0
- byte 7: request ID

- byte 9-12: x position
- byte 13-16: y position

- byte 17-20: z position
- byte 21: limits+ status (bit mapped, x+=bit 0; y+=bit 1; z+=bit 2;
- byte 22: limits- status (bit mapped, x-=bit 4; y-=bit 5; z-=bit 6)
- byte 23: home status (bit mapped, x0=bit 0; 0+=bit 1; z0=bit 2)
- byte 24: pulse engine state
- byte 25: x axis state
- byte 26: y axis state
- byte 27: z axis state
- byte 28: pulse engine enabled
- byte 29: safety charge pump enabled
- bytes 30-49: axes 4-8 position
- bytes 50-54: axes 4-8 state

Set current position

- byte 2: 0x80
- byte 3: 0x01
- byte 4: 0
- byte 5: 0
- byte 6: 0
- byte 7: request ID

- byte 9-12: x position
- byte 13-16: y position
- byte 17-20: z position
- byte 21-40: axes 4-8 position

Returned packet:

- byte 2: 0x80
- byte 3: 0x01
- byte 4-6: 0
- byte 7: request ID

Set current pulse engine state

- byte 2: 0x80
- byte 3: 0x02
- byte 4: 0
- byte 5: 0
- byte 6: 0
- byte 7: request ID

- byte 9: mode

Returned packet:

- byte 2: 0x80
- byte 3: 0x02
- byte 4-6: 0

- byte 7: request ID

Set MPG joggging options

- byte 2: 0x80
- byte 3: 0x03
- byte 4: 0 to read the configuration, 10 to write it
- byte 5: 0
- byte 6: 0
- byte 7: request ID

- byte 9: MPG jog activated
- bytes 10-13: MPG jog multiplier X
- bytes 14-17: MPG jog multiplier Y
- bytes 18-21: MPG jog multiplier Z
- byte 22: Axis X encoder ID+1 (0 disables the MPG jogging for the axis)
- byte 23: Axis Y encoder ID+1 (0 disables the MPG jogging for the axis)
- byte 24: Axis Z encoder ID+1 (0 disables the MPG jogging for the axis)
- bytes 25-44: MPG jog multipliers for axes 4-8
- bytes 45-49: Axes 4-8 encoder ID+1 (0 disables the MPG joggine for the axis)

Returned packet:

- byte 2: 0x80
- byte 3: 0x03
- byte 4-6: 0
- byte 7: request ID
- byte 9: MPG jog activated
- bytes 10-13: MPG jog multiplier X
- bytes 14-17: MPG jog multiplier Y
- bytes 18-21: MPG jog multiplier Z
- byte 22: Axis X encoder ID+1 (0 disables the MPG jogging for the axis)
- byte 23: Axis Y encoder ID+1 (0 disables the MPG jogging for the axis)
- byte 24: Axis Z encoder ID+1 (0 disables the MPG jogging for the axis)
- bytes 25-44: MPG jog multipliers for axes 4-8
- bytes 45-49: Axes 4-8 encoder ID+1 (0 disables the MPG joggine for the axis)

Set reference position

- byte 2: 0x80
- byte 3: 0x05
- byte 4: mode (0 for position reference, 10 for speed reference)
- byte 5: 0
- byte 6: 0
- byte 7: request ID

- byte 9-12: reference x position
- byte 13-16: reference y position
- byte 17-20: reference z position
- bytes 21-40: reference position for axes 4-8

Returned packet:

- byte 2: 0x80
- byte 3: 0x05
- byte 4-6: 0
- byte 7: request ID

Enable/disable pulse engine (read with command 0x80, 0x00)

- byte 2: 0x80
- byte 3: 0x06
- byte 4: enabled (1) / disabled (0)
- byte 5: activate safety charge pump output on pin 53
- bytes 6: 0
- byte 7: request ID

Get parameters

- byte 2: 0x80
- byte 3: 0x10
- byte 4: 0 for parameters for axes 1-3, 1 for axes 4-6, 2 for axes 7-8
- byte 5: 0
- byte 6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x80
- byte 3: 0x10
- byte 4-6: 0
- byte 7: request ID

- byte 9-20: max speed for x,y,z (32-bit float for each axis) - speed / 1ms
- byte 21-32: max acceleration for x,y,z (32-bit float for each axis) - acc / 1ms * 1ms
- byte 33-44: max decceleration for x,y,z (32-bit float for each axis) - acc / 1ms * 1ms
- byte 45: Homing speed in % of maximum speed
- byte 46: Homing return speed in % of maximum speed
- byte 47: Emergency switch polarity selection:
  - 0 – HIGH state activates the emergency
  - 1 – LOW state activates the emergency

Set parameters

- byte 2: 0x80
- byte 3: 0x11
- byte 4: 0
- byte 5: 0
- byte 6: 0
- byte 7: request ID

- byte 9-20: max speed for x,y,z (32-bit float for each axis) - speed / 1ms
- byte 21-32: max acceleration for x,y,z (32-bit float for each axis) - acc / 1ms * 1ms
- byte 33-44: max decceleration for x,y,z (32-bit float for each axis) - acc / 1ms * 1ms
- byte 45: Homing speed in % of maximum speed
- byte 46: Homing return speed in % of maximum speed
- byte 47: Emergency switch polarity selection
    0 – HIGH state activates the emergency
    1 – LOW state activates the emergency

Returned packet:

- byte 2: 0x80
- byte 3: 0x11
- byte 4-6: 0
- byte 7: request ID

Get axes parameters

- byte 2: 0x80
- byte 3: 0x12
- bytes 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x80
- byte 3: 0x12
- byte 4-6: 0
- byte 7: request ID

- byte 9: limit+ switches configuration (bits 0,1,2 - axes x,y,z)
- byte 10: limit- switches configuration (bits 0,1,2 - axes x,y,z)
- byte 11: home switches configuration (bits 0,1,2 - axes x,y,z)
- byte 12: direction change (bits 0,1,2 - axes x,y,z)
- byte 13: homing direction change (bits 0,1,2 - axes x,y,z)

Set axes parameters

- byte 2: 0x80
- byte 3: 0x13
- bytes 4-6: 0
- byte 7: request ID

- byte 9: limit+ switches configuration (bits 0,1,2 - axes x,y,z)
- byte 10: limit- switches configuration (bits 0,1,2 - axes x,y,z)
- byte 11: home switches configuration (bits 0,1,2 - axes x,y,z)
- byte 12: direction change (bits 0,1,2 - axes x,y,z)
- byte 13: homing direction change (bits 0,1,2 - axes x,y,z)

Get controller setup

- byte 2: 0x80
- byte 3: 0x14
- bytes 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x80
- byte 3: 0x14
- byte 4-6: 0
- byte 7: request ID

- byte 9: CNC keyboard enabled status - 1 if keyboard is enabled if detected

Set CNC keyboard setup

- byte 2: 0x80
- byte 3: 0x15
- bytes 4-6: 0
- byte 7: request ID

- byte 9: CNC keyboard enabled status - 1 if keyboard is enabled if detected

Execute homing

- byte 2: 0x80
- byte 3: 0x20
- byte 4: 0
- byte 5: 0
- byte 6: 0
- byte 7: request ID

- byte 9-11: x,y,z homing status (set to 1 to home the axis)

Returned packet:

- byte 2: 0x80
- byte 3: 0x20
- byte 4-6: 0
- byte 7: request ID

Get engine info

- byte 2: 0x80
- byte 3: 0x30
- byte 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x80
- byte 3: 0x30
- byte 4-6: 0
- byte 7: request ID

- byte 9: number of axes supported
- byte 10: maximum pulse frequency in kHz
- byte 11: buffer size
- byte 12: slot timing (in x100 us)

Fill buffer

- byte 2: 0x80
- byte 3: 0x35
- byte 4: number of new entries
- byte 5-6: 0
- byte 7: request ID

- bytes 9-64: buffer entries (one byte per axis) – 0xFF (-127) for axis x is special command

Returned packet:

- byte 2: 0x80
- byte 3: 0x35
- byte 4-6: 0
- byte 7: request ID

- byte 9: number of entries accepted
- byte 10-13: x position
- byte 14-17: y position
- byte 18-21: z position
- byte 22: limits+ status (bit mapped, x+=bit 0; y+=bit 1; z+=bit 2;
- byte 23: limits- status (bit mapped, x-=bit 4; y-=bit 5; z-=bit 6)
- byte 24: home status (bit mapped, x0=bit 0; 0+=bit 1; z0=bit 2)
- byte 25: pulse engine state
- byte 26: x axis state
- byte 27: y axis state
- byte 28: z axis state

Clear buffer

- byte 2: 0x80
- byte 3: 0x36
- byte 4-6: 0
- byte 7: request ID

Get buffer size

- byte 2: 0x80
- byte 3: 0x37
- byte 4-6: 0
- byte 7: request ID

Returned packet:

- byte 2: 0x80
- byte 3: 0x37
- byte 4-6: 0
- byte 7: request ID

- byte 9: buffer size