PoKeys protocol specification

Copyright PoLabs 2008-2025
All rights reserved

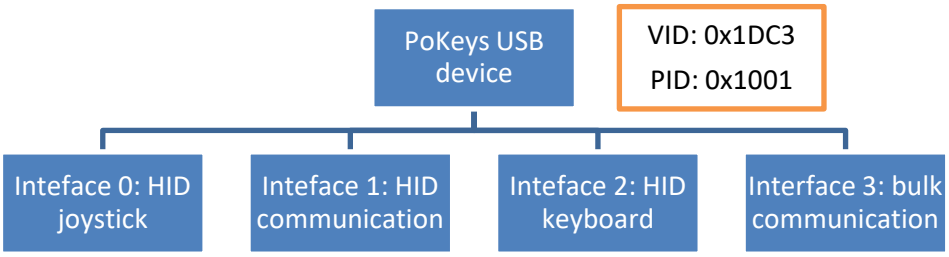
Version: 14.3.2025

Compatible PoKeys firmware versions:
PoKeys57: **4.7.15**

Brief protocol description

USB PoKeys devices

PoKeys USB devices are a composite USB device with multiple standard USB HID interfaces (for USB keyboard and joystick emulation), custom USB HID interface for communication and optional bulk communication interface. USB HID interfaces use system USB drivers by default and require no additional driver installation. The bulk communication interface can be used with WinUSB driver on Windows (GUID of 2EA10865-4FFD-4BF3-8EF3-161549BFA270) provided in the PoKeys setup package or standard libusb library on Linux and OS X.



The device encapsulates three interfaces, first (index 0) being standard USB HID joystick, second (index 1) being PoKeys communication interface and the third (index 2) standard USB HID keyboard.
The PoKeys API command interface is implemented on interfaces 1 and 3 (HID and bulk communication), both using 64-byte sized packets.
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 API interface of the PoKeys device. If more than one PoKeys device is connected to the same host, differentiation at this level is impossible, so user ID byte or serial number must be read from the PoKeys device. USB serial number descriptor 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).
The suggested sequence to API commands is as follows:

1. send report with a unique request ID (can be increased by 1 for each new request),
2. read report,
3. check request ID value - if it does not match the one sent in step 1, repeat step 2,
4. check packet checksum, if it does not match, repeat with step 1.

Ethernet PoKeys devices

Ethernet PoKeys devices use a combination of UDP and TCP packets to communicate with the host. Both use a port number 20055.
To ensure a highest possible compatibility with USB PoKeys devices, ethernet PoKeys devices share the request data structures with USB devices. Packets are transfered as standard TCP or UDP packets.
PoKeys devices support device discovery procedure that enables the network devices to be discovered in the network (i.e. determine their IP address).

Device discovery

PoKeys devices are discovered via broadcast UDP packets. A host sends out a UDP packet on a broadcasting address of the network interface. All PoKeys devices respond with a UDP packet that contains the device's identification (User ID, serial number, version) and it's IP address.
The discovery procedure operates in local networks that do not have DHCP server available - if device is configured to use DHCP server and no DHCP server provided a valid IP address yet, on reception of the discovery packet, the PoKeys device will use the temporary address of x.x.x.250, where x.x.x is the subnet address of the host that the request was sent from (with 255.255.255.0 subnet mask). This allows the host to establish the communication with the device in order to reconfigure it.
The automatic IP/subnet configuration on discovery packet reception can be disabled in the device settings, but is enabled by default.

Discovery packet format

Packet data is formated the following way:
Host -> device:

- Destination address: broadcast (e.g. 255.255.255.255, 192.168.x.255, etc.)
- Target port: 20055
- Contents: empty packet

Device -> host

- Sent back to originating port on the host
- Contents:
    - [1 B] User ID number
    - [1 B] Reserved (0)
    - [1 B] Reserved (0)
    - [2 B] FW version (major, minor)
    - [4 B] Device's IP address
    - [1 B] DHCP enabled
    - [4 B] Host's (originating) IP
    - [4 B] Device's serial number
    - [1 B] Hardware ID

Communication

All further communication with the device is acomplished with TCP or UDP connection on port 20055. Packets are 64-bytes long and their structure is shared with USB PoKeys devices.

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

Request and response packets are 64 bytes long and have the following headers

	Byte 1	Byte 2	Bytes 3-6	Byte 7	Byte 8	Bytes 9-64
Host->Device (request)	0xBB	Operation ID	Optional parameters	Request ID	Checksum (sum mod 256)	Optional parameters
Device->Host (response)	0xAA	Operation ID (copy of request)	Optional parameters	Request ID (copy of request)	Checksum (sum mod 256)	Optional parameters

Reserved bytes should be set to 0.

PoKeys InterCom protocol

The PoKeys InterCom protocol is used for communication between PoKeys devices in local network. It uses Ethernet frame type of 0xA057 and has the following structure.

- Standard Ethernet Type 2 header
- PoKeys InterCom header
- PoKeys InterCom data
- TBD…

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)
57	42	PoKeys57Utest 2	0 (default)
57	43	LiniTester programmer	0 (default)
57	44	LiniTester calibrator	0 (default)
57	45	PoKeys57Industrial1	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)
57	100	OEM	0 (default)
57	101	OEM SerialReader	0 (default)
57	102	OEM X15-02-24	0 (default)
57	103	OEM ZB-Bridge	0 (default)
57	104	OEM SerialReaderRGB	0 (default)
57	105	PoLED gateway	0 (default)
57	106	SmartLight gateway v1.0	0 (default)
57	107	PoKeys Antenna Rotator	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
PoKeys57CNCpro4x25	33	
PoKeys57U OEM	35	TBD
PoKeys57E OEM	36	TBD
PoPLC57NG	37	TBD
PoKeys57CNCdb25	38	TBD
PoKeys57Utest	39	TBD
PoKeys57Utest	42	TBD
LiniTester programmer	43	TBD
LiniTester calibrator	44	TBD
PoKeys57Industrial1	45	TBD
OEM	100	TBD
LiniTester	101	TBD
OEM X15	102	TBD
OEM ZB-Bridge	103	TBD
PoLED control node	105	TBD
OEM - CAN node	200	TBD

Series 58

Series 58 contains PoPLC v1.0 and PoKeys58EU

TBD

Table of contents

Supported commands

NFU = not for future use (with indicated alternate command)

CMD	Description	NFU
0x00	Read device data	
0x01	Reserved	
0x02	Set user ID	
0x03	Read User ID and lock setting	x (0x00)
0x04	Read build date	x (0x00)
0x05	Get system load status	
0x06	Read/set device name	
0x07	Configure USB interface	
0x08	Delayed startup configuration	
0x09	PPM decoder configuration and status	
0x0A	Session settings	
0x0B	Misc USB device configuration	
0x0C	Reserved	
0x0D	Reserved	
0x0E	Reserved	
0x0F	Reserved	
0x10	Set input/output settings	x (0xC0)
0x11	Set encoder settings	
0x12	Set encoder key mapping for direction A	
0x13	Set encoder key mapping for direction B	
0x14	Reserved	
0x15	Get pin function	x (0xC0)
0x16	Get encoder settings	x (0xC4, 0xC5)
0x17	Get encoder key mapping for direction A	x (0xC6)
0x18	Get encoder key mapping for direction B	x (0xC7)
0x19	Get encoder RAW value	x (0xCD)
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	x (0xC0)
0x1F	Get pin capabilities	
0x20	Set key association	x (0xC1, 0xC2, 0xC3)
0x21	Pin typematic delay	
0x22	Pin repeat rate	
0x23	Reserved	
0x24	Reserved	
0x25	Get key association	x (0xC1, 0xC2, 0xC3)
0x26	Reserved	
0x27	Reserved	
0x28	Reserved	
0x29	Reserved	
0x2A	Reserved	

0x2B	Reserved	
0x2C	Reserved	
0x2D	Reserved	
0x2E	Reserved	
0x2F	Reserved	
0x30	Get input	
0x31	Block get input I	x
0x32	Block get input II	x
0x33	Reserved	
0x34	Reserved	
0x35	Get analog input	
0x36	Block get analog (4x 8bit)	x (0x3A)
0x37	Block get analog (3x 10bit)	x (0x3A)
0x38	Get analog RC filter value	
0x39	Set analog RC filter value	
0x3A	Get all analog inputs (7x 12bit on PoKeys56) / extended mode	
0x3B	Reserved	
0x3C	Reserved	
0x3D	Reserved	
0x3E	Reserved	
0x3F	Full I/O (device specific)	
0x40	Set output	
0x41	Set analog output	
0x42	Block set output I	x
0x43	Block set output II	x
0x44	Reserved	
0x45	Reserved	
0x46	Reserved	
0x47	Reserved	
0x48	Reserved	
0x49	Reserved	
0x4A	MCS system	
0x4B	WS2812 commands	
0x4C	Simple pulse generator	
0x4D	Reserved	
0x4E	Reserved	
0x4F	Reserved	
0x50	Save configuration	
0x51	Save and lock configuration	
0x52	Disable lock and reset configuration	
0x53	Reserved	
0x54	Reserved	
0x55	Reserved	
0x56	Reserved	
0x57	Reserved	
0x58	Reserved	
0x59	Reserved	
0x5A	Reserved	
0x5B	Reserved	
0x5C	Reserved	
0x5D	Reserved	
0x5E	Reserved	

0x5F	Reserved	
0x60	Get joystick configuration	
0x61	Get joystick up Event buttons configuration	
0x62	Reserved	
0x63	Reserved	
0x64	Reserved	
0x65	Set joystick configuration	
0x66	Set joystick up Event buttons configuration	
0x67	Reserved	
0x68	Reserved	
0x69	Reserved	
0x6A	Set joystick analog to digital mapping	
0x6B	Reserved	
0x6C	Reserved	
0x6D	Reserved	
0x6E	Reserved	
0x6F	Reserved	
0x70	Setup sensors (PoKeys series 56)	x (0x76)
0x71	Setup dashboard items	x (0x78)
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)	
0x79	Simple sensor configuration (PoKeys series 57)	
0x7A	Reserved	
0x7B	Reserved	
0x7C	Reserved	
0x7D	Reserved	
0x7E	Reserved	
0x7F	Reserved	
0x80	Pulse engine commands (deprecated since firmware 3.1.0)	x (0x85)
0x81	Failsafe settings	
0x82	PoIL commands	
0x83	RTC settings	
0x84	System log operations	
0x85	Pulse engine commands v2	
0x86	CAN operations	
0x87	Reserved	
0x88	Reserved	
0x89	Reserved	
0x8A	Reserved	
0x8B	Reserved	
0x8C	Reserved	
0x8D	Reserved	
0x8E	Reserved	
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	
0x9B	Reserved	
0x9C	Reserved	
0x9D	Reserved	
0x9E	Reserved	
0x9F	Reserved	
0xA0	Setup RTmode	x (0x3F)
0xA1	RTmode packet – in/out	x (0x3F)
0xA2	Configure UDP binary realtime mode	
0xA3	Reserved	
0xA4	Reserved	
0xA5	Reserved	
0xA6	Reserved	
0xA7	Reserved	
0xA8	Reserved	
0xA9	Reserved	
0xAA	Reserved	
0xAB	Reserved	
0xAC	Reserved	
0xAD	Reserved	
0xAE	Reserved	
0xAF	Reserved	
0xB0	Multipart packet	
0xB_	Reserved	
0xC0	Pin configuration	
0xC1	Pin key mapping	
0xC2	Pin key codes	
0xC3	Pin key modifiers	
0xC4	Encoder option	
0xC5	Encoder channel A + B	
0xC6	Encoder channel A key code + modifier	
0xC7	Encoder channel B key code + modifier	
0xC8	Reserved	
0xC9	Reserved	
0xCA	Reserved	
0xCB	Reserved	
0xCC	Full device status	
0xCD	Get encoder long RAW values	
0xCE	Enable fast encoders	
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 32bit RAW encoder data	

0xCE	Enable/disable Fast encoders
0xCF	Get tick counter
0xD0	LCD configuration
0xD1	LCD operation
0xD2	Reserved
0xD3	Reserved
0xD4	Reserved
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
0xDF	Reserved
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
0xE6	CAN node commands
0xE7	Reserved
0xE8	Reserved
0xE9	Reserved
0xEA	Reserved
0xEB	Reserved
0xEC	Reserved
0xED	Reserved
0xEE	Reserved
0xEF	Get/set Cosm settings
0xF3	Reboot system

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
0xF7 - reserved
0xFA - custom command 1
0xFB - OEM command
0xFD - OEM parameters

List of commands

The following section lists all commands supported by the device with description of the parameters.

Note: 1-based byte numbering is used in the API documentation (first byte is designated as byte 1).

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 43-57: 15 bytes of device ID (applicable to application firmware)
- 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 (chksm)	CHECKSUM

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 (chksm)	CHECKSUM

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 (chksm)	CHECKSUM
9-18	session data (if byte 3 is set to 0x10)
19-63	Reserved
64 (chksm 2)	Checksum

Session settings - Response

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

19-63	Reserved
64 (chksm 2)	Checksum

Reboot system

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xF3**
3-6	reserved
7	Request ID
8 (chksm)	Checksum

Reboot system - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xF3**
3-6	Reserved
7	Request ID
8 (chksm)	Checksum

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 (chksm)	Checksum
9	Interrupt communication interval in ms
10	USB boot support
11-14	USB failsafe no communication timeout (in ms)
15	USB failsafe options: - 0: disabled - 1: reset USB on communication timeout - 2: reset device on communication timeout
16-63	Reserved
64 (chksm 2)	Checksum

Session settings - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x0B**
3-6	Reserved

7	Request ID
8 (chksm)	Checksum
9	Interrupt communication interval in ms
10	USB boot support
11-14	USB failsafe no communication timeout
15	USB failsafe options
16-63	Reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

Response

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

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 and PoKeys57 series 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

Simple pulse generator

Implemented in FW v4.4.20.

PoKeys57 series devices can generate pulses on any PoKeys digital output pin. The generator is configured with:

- pin ID for the PoKeys digital output
- pulse period (in steps of 0.1 ms)
- pulse count
- pulse duty cycle (width) in 0-255 (0-100%)

Device can generate pulses on up to 16 digital outputs simultaneously.

Start generator

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x4C**
3	Command (0x00) - start generator
4	Pin ID (1-based)
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-10	Pulse period in 0.1 ms steps, unsigned 16-bit integer
11-12	Pulse count, unsigned 16-bit integer
13	Pulse width 0-255 (0-100%)
14-64	Reserved

Start generator - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x4C**
3	0x00
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-62	Reserved

Stop generator

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x4C**
3	Command (0x01) - stop generator
4	Pin ID (1-based)
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-64	Reserved

Stop generator - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x4C**
3	0x01
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-62	Reserved

Get generator status

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x4C**
3	Command (0x10) - get status
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-64	Reserved

Get generator status- Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x4C**
3	0x10
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-63	Status for each pin indicating whether the generator is active or not

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)

Returned packet:

- byte 2: 0x42 (or 0x43)
- byte 3-6: reserved
- byte 7: request ID

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: 0xCC
- bytes 3-6: reserved
- 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)

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

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)

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)

Full I/O command (device specific)

Full I/O extends 0xCC command with support for different target devices.

PoKeys57Industrial

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x3F**
3	Reserved (set to 0)
4	Reserved (set to 0)
5	Digital outputs enable (bit-mapped)
6	Digital output re-enable (bit-mapped, 1 to re-enable the output - only set bits after the user acknowledges the fault)
7	Request ID
8 (Checksum)	Checksum
9	Digital outputs (bit-mapped)
10-16	Reserved
17-24	Analog outputs (4x 16-bit)
25-64	Reserved

Read parameters - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x3F**
3	Reserved
4	Reserved
5	Digital outputs error statuses (bit-mapped) - bit status 1 indicates the output has been shut down to prevent overcurrent situation
6	Analog outputs error statuses (bit-mapped) - bit status 1 indicates the output current has been limited to 20 mA
7	Request ID
8 (Checksum)	Checksum
9	Digital inputs (bit-mapped)
10-16	Reserved
17-32	Analog inputs (8x 16-bit)
33-44	Encoder values (3x 32-bit)
45-64	Reserved

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

Option = 50:

- byte 9: matrix keyboard scanning decimation (0-50)

Returned packet:

- byte 2: 0xCA

[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 3: keyboard ID
- byte 4-6: reserved
- 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)
- byte 52: matrix keyboard scanning decimation (0-50)
- bytes 53-63: 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

[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.

If option > 0

- 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 (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
- Send custom command to LCD controller – operation code 0x61

  - Byte 4: command byte
- Send custom data to LCD controller – operation code 0x62
  - Byte 4: data byte
- 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

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

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 - some PoKeys devices (PoKeys56U/E and PoKeys57U/E) allow the PoExtBus functionality to be moved from the dedicated PoExtBus connector to following pins (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

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xDA**
3	PoExtBus option: - 0: disable PoExtBus - 1: enable PoExtBus - 2: read the state of PoExtBus - 10: disable auto-refresh of PoExtBus outputs - 11: re-enable auto-refresh
4	Connector selection: - 0: default PoExtBus connector - 1: PoKeys pins (selected PoKeys devices only)
5	set to >0 to disable CRC check value on 11th device position data
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xDA**
3	PoExtBus status
4	Connector selection
5	CRC is used on 11th data position if this is 0
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-18	Data bytes
19-63	reserved
64 (chksm 2)	unused

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, PoKeys57U 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	PoKeys5xU pin ID	PoKeys5xE 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), no function in PoKeys57 series
- 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 and activation status
- 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 3-6: reserved or defined below
- byte 7: request ID

1-wire operations

- 0x00 - Deactivate 1-wire
  - byte 3: 1-wire operation
  - byte 4: 1-wire operation result (1 if successfull, 0 unsuccessfull, 0x10 – operation still executing)
- 0x01 - Activate 1-wire
  - byte 3: 1-wire operation
  - byte 4: 1-wire operation result (1 if successfull, 0 unsuccessfull, 0x10 – operation still executing)
- 0x10 - Start operation
  - byte 3: 1-wire operation
  - byte 4: 1-wire operation result (1 if successfull, 0 unsuccessfull, 0x10 – operation still executing)
- 0x11 - Get result of read process
  - Byte 4: activation status / result
  - 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 **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.
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-12	Baudrate
13-63	Reserved
64 (chksm 2)	unused

Response

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

UDP binary interface

UDP binary interface is a simple interface designed for PoKeys57 Ethernet devices to be used as general purpose Ethernet I/O interfaces (e.g. for use with Matlab Simulink Desktop Real-Time toolbox via Packet Input and Packet Output blocks). It uses a fixed UDP port of 20065 on the side of the device.
It uses data slots of fixed 32-bit width (32-bit signed integers) that are directly mapped to PoIL memory via 16-bit addresses. The value of 0xFFFF disables the slot (reading/writing to pin 1 is therefore not supported).

0xA2/0x01 Setup input and output mapping

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xA2**
3	Command (0x01) - setup input/output mapping
4	Reserved
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-24	Outputs mapping addresses (PoIL memory 16-bit addresses for each slot, 0 disables the slot)
25-40	Inputs mapping addresses (PoIL memory 16-bit addresses for each slot, 0 disables the slot)
41-64	Reserved

Setup input/output mapping - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xA2**
3	0x01
4	Reserved
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-24	Outputs mapping addresses (PoIL memory 16-bit addresses for each slot, 0 disables the slot)
25-40	Inputs mapping addresses (PoIL memory 16-bit addresses for each slot, 0 disables the slot)
41-64	Reserved

0xA2/0x02Read input and output mapping

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xA2**
3	Command (0x02) - read input/output mapping
4	Reserved
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-64	Reserved

Read input and output mapping - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xA2**
3	0x02

4	Reserved
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-24	Outputs mapping addresses (PoIL memory 16-bit addresses for each slot, 0 disables the slot)
25-40	Inputs mapping addresses (PoIL memory 16-bit addresses for each slot, 0 disables the slot)
41-64	Reserved

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 (chksm)	Checksum
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 (chksm)	Checksum
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 (chksm)	Checksum
9-16	Unit simple name (8 characters)
17-48	Unit HTML code (32 characters)
49-63	unused / reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-16	Unit simple name (8 characters)
17-48	Unit HTML code (32 characters)
49-63	unused / reserved
64 (chksm 2)	Checksum

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	
Generic SimpleSensor	SimpleSensor	0x80	sensor measurement ID	
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}			

Simple 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 ID		Sensor serial number				unused	unused

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 (chksm)	Checksum
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 (chksm 2)	Checksum

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 (chksm)	Checksum
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 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	unused / reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
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 (chksm 2)	Checksum

SimpleSensors protocol

Selected PoKeys57 series devices support SimpleSensors protocol that enables connection of simple external serial measurement devices.

0x79/0x00 Get SimpleSensors status

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x79**
3	Command (0x00) - get status
4-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-64	Reserved

Get SimpleSensors status - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x79**
3	0x00
4	Number of sensors detected
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-62	Detected sensors list (up to 9 sensors), each entry containing 6 bytes per sensor - sensor serial (4 bytes) - sensor ID (2 bytes)
63-64	Reserved

0x79/0x10 Set index for full sensor packet

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x79**
3	Command (0x10) - set index
4	Sensor index
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-64	Reserved

Set index for full sensor packet - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x79**
3	0x10
4	Sensor index
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
63-64	Reserved

0x79/0x10 Get full sensor packet contents

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x79**
3	Command (0x11) - get full sensor packet
4-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-64	Reserved

Get full sensor packet contents - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x79**
3	0x11
4	Sensor index
5	Sensor data length (0 if data is not yet available)
6	Sensor data age (in seconds)
7	Request ID
8 (chksm)	Checksum
9-64	Sensor data package

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 (chksm)	Checksum
Operation 10	
9-10	Update rate
11-14	Server IP
15	Request type bit 1:0: protocol selection Legacy protocols (see description for the other bits below) 0 - specified HTTP header 1 - RAW data New protocols: 2 - New protocol description - description of the protocol in bytes 18-* 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 - deprecated support in FW 4.3.9 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	Protocol description - only valid if request type bits set to 2 Byte 15: communication protocol selection - bits 1:0: constant value of 2 - bits 4:2: o 0: custom HTTP request o 1: UDP data mode - bits 7:5: reserved Byte 20: value formatting - bits 4:0: format o 0: standard: {name},{value} o 10: value only: {value} - bits 6:5: reserved - bit 7: ignore sensor OK value Byte 21: value delimiter - between name and value - 0: comma: {name},{value} - 1: equal sign =: {name}={value} - 10: | as a delimiter: {name}|{value} Byte 22: line delimiter - between values - 0: standard: new line

	- 1: only \n - 2: only \r - 10: comma - 11: semicolon - 12: &
Operations 11-15	
9-58	Request header – 50 bytes (operation number selects page index)
59-63	Reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
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 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
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 (chksm 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 (chksm)	Checksum
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 (chksm 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 (chksm)	Checksum
9-63	reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
9-63	reserved
64 (chksm 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 (chksm)	Checksum
9-40	Axis positions (32-bit LSB first)
41-63	Reserved
64 (chksm 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 (chksm)	unused
9-63	reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
9-63	reserved
64 (chksm 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	PoKeys57CNC	PoKeys57CNCpro4x25
0	SSR 2	FAN control
1	Relay 2	Relay 1
2	Relay 1	Relay 2
3	OC 1	OC 1

	4	OC 2	OC 2	
	5	OC 3	OC 3	
	6	OC 4	OC 4	
	7	SSR 1	Plasma relay	
6	If set to 1, the above output values are not used - read function			
7	Request ID			
8 (chksm)	Checksum			
9-63	Reserved			
64 (chksm 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 (chksm)	Checksum
9-63	reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x05 - Reboot pulse engine
4-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-63	reserved
64 (chksm 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 (chksm)	Checksum
9	Emergency switch input pin (0 - disabled, 1 - default, 10+ pin ID)
10-63	Reserved
64 (chksm 2)	unused

Response

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

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x08 - Status message 2

4	
5	
6	
7	Request ID
8 (chksm)	Checksum
9	Status of dedicated limit- inputs
10	Status of dedicated limit+ inputs
11	Status of dedicated home inputs
64 (chksm 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 (chksm)	Checksum
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 (chksm)	Checksum
9-63	reserved
64 (chksm	unused

2)

0x85/0x0B Setup synced digital outputs

Synched digital outputs enable specified outputs to be controlled directly via the motion buffer queue. First, pin IDs are configured for each of the 8 channels, then, motion data is interprted bit-wise to control the selected channels.

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x0B - setup synced digital outputs
4	enabled flag (set to 1 to enable synched digital outputs functionality, set to 2 to read the settings)
5	source axis ID (0 to 7)
6	reserved
7	Request ID
8 (chksm)	Checksum
9-16	Output pin IDs for each of 8 synched digital outputs 0 - unused 1 - x: pin ID (1-based) 92-99: on-board dedicated digital outputs
17-64	

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x0B - setup synced digital outputs
4	enabled flag
5	source axis ID (0 to 7)
6	reserved
7	Request ID
8 (chksm)	Checksum
9-16	Output pin IDs for each of 8 synched digital outputs
17-64	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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
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)
46-49	Home back-off distance (32-bit integer)
50-51	MPG jogging - encoder divider value
52	Additional misc options ao_INVERT_STEP = (1<<0), ao_INVERT_DIRECTION = (1<<1) (is XOR-ed with aoINVERTED option!) ao_INDEX_FALLING_EDGE = (1<<2)
53	Probe input filter setting (0-254)
54-63	reserved
64 (chksm 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 (chksm)	Checksum
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, removed in 4.2.32)
46-49	Home back-off distance (32-bit integer)
50-51	MPG jogging - encoder divider value
52	Additional misc options ao_INVERT_STEP = (1<<0), ao_INVERT_DIRECTION = (1<<1) (is XOR-ed with aoINVERTED option!) ao_INDEX_FALLING_EDGE = (1<<2)
53	Probe input filter setting (0-254)
54-63	reserved (set to 0)
64 (chksm 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 (chksm)	Checksum
9-63	reserved
64 (chksm	unused

2)	

0x85/0x18 Get internal motor drivers configuration

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x18 - Get axis drivers configuration
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x18 - Get axis drivers configuration
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
9	Axis 1 - step setting [0 - 8]: 0: 1/1, 1: ½ non-circular, 2: ½, 3: ¼, 4: 1/8, 5: 1/16, 6: 1/32, 7: 1/128, 8: 1/256
10	Axis 1 - current setting 0-2.5 A [0-255]
11-12	Axis 2 (as above for axis 1)
13-14	Axis 3 (as above for axis 1)
15-16	Axis 4 (as above for axis 1)
17-63	reserved
64 (chksm 2)	unused

0x85/0x19 Set internal motor drivers configuration

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x19 - Set axis drivers configuration
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
9	Axis 1 - step setting [0 - 8]: 0: 1/1, 1: ½ non-circular, 2: ½, 3: ¼, 4: 1/8, 5: 1/16, 6: 1/32, 7: 1/128, 8: 1/256
10	Axis 1 - current setting 0-2.5 A [0-255]

11-12	Axis 2 (as above for axis 1)
13-14	Axis 3 (as above for axis 1)
15-16	Axis 4 (as above for axis 1)
17-63	Reserved
64 (chksm 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x19 - Set axis drivers configuration
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
9	Axis 1 - step setting [0 - 8]: 0: 1/1, 1: ½ non-circular, 2: ½, 3: ¼, 4: 1/8, 5: 1/16, 6: 1/32, 7: 1/128, 8: 1/256
10	Axis 1 - current setting 0-2.5 A [0-255]
11-12	Axis 2 (as above for axis 1)
13-14	Axis 3 (as above for axis 1)
15-16	Axis 4 (as above for axis 1)
17-63	reserved
64 (chksm 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 (chksm)	Checksum
9-40	Reference position or speed (8x 32-bit integer, LSB first)
41-63	reserved
64 (chksm 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 (chksm)	Checksum
9-63	reserved
64 (chksm 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 (chksm)	Checksum
9-40	Home position offsets (32-bit LSB first per axis)
41-63	Reserved
64 (chksm 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x21 - Start homing
4-6	Reserved

7	Request ID
8 (chksm)	Checksum
9-63	reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	unused

Response

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

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x23 - Start probing
4-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-63	reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	unused

Response

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

0x85/0x25 Move PV (Set reference position and speed)

Request

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x85**
3	0x25 - move PV
4	Axis mask - bits 7..0 configure which axes are commanded. Set to 0xFF for all axes.
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-40	Reference position (8x 32-bit integer, LSB first)
41-56	Move velocity (0 - 0xFFFF => 0 - 100 %)
57-63	reserved
64 (chksm 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x25 - move PV
4	Reserved
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-63	reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	unused

Response

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

7	Request ID
8 (chksm)	Checksum
9-63	reserved
64 (chksm 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 (chksm)	Checksum
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 (chksm)	Checksum
9-63	See 'Get status' command report
64 (chksm 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** Reserved **Bit 6** Reserved **Bit 5** Reserved **Bit 4** Stop bit **Bit 3** Start bit **Bit 2 Bit 1 Bit 0** 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 (chksm)	Checksum
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 (chksm)	Checksum
9-63	See 'Get status' command report
64 (chksm 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 (chksm)	Checksum
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 (chksm)	Checksum
9-64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
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 (chksm 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 (chksm)	Checksum
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 (chksm 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 (chksm)	Checksum
9-63	reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
9-63	reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm	unused

2)

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x95 - Read smart pulse generator status
4-6	Reserved
7	Request ID
8 (chksm)	Checksum
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 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
9-40	Current encoder positions
41-63	reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
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)
25-28	Spindle index counter
29-63	reserved
64 (chksm 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 (chksm)	Checksum
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 Bit 6: enable index counter
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 (chksm 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 (chksm)	Checksum
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
26-29	Spindle index counter
30-63	reserved
64 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 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 (chksm)	Checksum
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 (chksm 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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	unused

Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x85**
3	0x40 - Get backlash compensation settings
4	Backlash compensation enabled
5	Maximum backlash speed
6	Reserved
7	Request ID
8 (chksm)	Checksum
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 (chksm 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
5	Maximum backlash speed
6	Reserved
7	Request ID
8 (chksm)	Checksum
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 (chksm 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 (chksm)	Checksum
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 (chksm 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 (chksm)	Checksum
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 (chksm 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 (chksm)	Checksum
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 (chksm 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 (chksm)	Checksum
9-63	reserved
64 (chksm 2)	unused

Response

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

2)

Response

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

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 (chksm)	Checksum
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 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

PoIL core status - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x82**
3	0x00 – get PoIL core status
4	Current PoIL core ID
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
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
33-52	Exception information (location, parameters)
53-59	reserved
60	PoIL master enable status
61	Number of PoIL tasks
63	Number of external PoIL cores supported
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

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

0x82/0x06 Select PoIL core

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x82**
3	0x06 – select PoIL core
4	PoIL core ID
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

PoIL core status - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x82**
3	0x06 – select PoIL core
4	PoIL core ID
5	Reserved
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9	Memory chunk size

10-63	Reserved
64 (chksm 2)	Checksum

PoIL memory read - Response

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

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 (chksm)	Checksum
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 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	Memory contents by chunks of data
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9	Memory chunk size
10-63	Memory contents
64 (chksm 2)	Checksum

PoIL memory write - Response

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

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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

PoIL memory erase - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x82**
3	0x16 – erase memory
4-6	Reserved
7	Request ID
8 (chksm)	Checksum

9-63	reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
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 (chksm)	Checksum
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 (chksm 2)	Checksum

Get custom device status - Response

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

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 (chksm)	Checksum
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 (chksm 2)	Checksum

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 (chksm)	Checksum
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 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-62	27 Log entries (LSB first)
63	Reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
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 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	MCS chanels statuses
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	MCS chanels statuses
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	MCS chanels statuses
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
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 (chksm 2)	Checksum

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 (chksm)	Checksum
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 (chksm 2)	Checksum

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 (chksm)	Checksum
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 (chksm 2)	Checksum

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 (chksm)	Checksum

9-63	Reserved
64 (chksm 2)	Checksum

Read MCS rough positions - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x4A**
3	0x20
4-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-63	MCS chanels positions (0-255 in the specified minimum to maximum range)
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-63	MCS chanels positions (0-255 in the specified minimum to maximum range)
64 (chksm 2)	Checksum

Set MCS rough positions - Response

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

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 (chksm)	Checksum
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 (chksm 2)	Checksum

Configure CAN - Response

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

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 (chksm)	Checksum
9-12	CAN ID
13-16	
17-18	
19-20	
21-22	
23	
24-63	reserved
64 (chksm 2)	Checksum

Register CAN filter - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x86**
3	0x10
4	
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-12	
13-16	
17-18	
19-20	
21-22	
23	
24-63	reserved
64 (chksm 2)	Checksum

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 (chksm)	Checksum
9-12	CAN ID
13-20	data byte
21	number of bytes
22	format
23	frame type
24-63	reserved
64 (chksm 2)	Checksum

Send CAN message - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x86**
3	0x20

4	
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-12	
13-16	
17-18	
19-20	
21-22	
23	
24-63	reserved
64 (chksm 2)	Checksum

0x86/0x30 Receive CAN messages

```
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 messages
4	
5	Confirmation of last message queue ID (ignored if 0xFF or greater than CAN queue pointer)
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-12	
13-20	
21	
22	
23	
24-63	reserved
64 (chksm 2)	Checksum

Read CAN message - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x86**
3	0x30
4	status -
5	last message queue ID
6	Reserved
7	Request ID

8 (chksm)	Checksum
9-12	CAN ID
13-20	data byte
21	number of bytes
22	format
23	frame type
24-63	up to 2 additional CAN messages
64 (chksm 2)	Checksum

0x86/0x31 Receive single 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	0x31 – Read single CAN message
4-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-63	reserved
64 (chksm 2)	Checksum

Read CAN message - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x86**
3	0x31
4	status - 1 if message available, 0 otherwise
5	
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-12	CAN ID
13-20	data byte
21	number of bytes
22	format
23	frame type
24-63	reserved
64 (chksm 2)	Checksum

0x86/0x32 Flush CAN Rx queue

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x86**
3	0x32 – Flush CAN Rx queue
4-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-63	reserved
64 (chksm 2)	Checksum

Read CAN message - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x86**
3	0x32
4-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-63	reserved
64 (chksm 2)	Checksum

WS2812 control

Selected PoKeys57 series devices support generating control signals for WS2812 RGB LED driver. Single channel is available on pin 23 (PoKeys57U) and up to 1023 WS2812 can be connected in series. When WS2812 support is activated, keyboard macro functionality is disabled and should not be used.
Color of each LED is represented with three 8-bit values (red, green, blue channels) with an additional (currently-unused) lumma byte, totaling to 4 bytes per LED. Commands are listed below that can be used to alter the LEDs data memory directly. The PoKeys device refreshes all (1023) WS2812 in less than 45 ms.

0x4B/0x00 Control WS2812

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x4B**
3	Command (0x00)
4-5	LED count
6	Refresh command (1 to start single refresh)
7	Request ID
8 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

Control WS2812 - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0x4B**
3	0x00
4	Status (1 - OK)
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-63	Reserved
64 (chksm 2)	Checksum

0x4B/0x10 Update memory - no lumma

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0x4B**
3	Update memory command (0x10)
4-5	First LED index (i)
6	LED data count N (bytes 9-63 contain 3 times more bytes)
7	Request ID
8 (chksm)	Checksum
9-11	LED i data (R, G, B values)
12-14	LED (i+1) data
	…
60-62	LED (i+17) data (if N = 18)
63	reserved
64 (chksm 2)	Checksum

No response is generated, command can be sent in bursts over USB bulk endpoint.

OEM parameters

PoKeys57 series device can store up to 62 32-bit parameters that can be used as non-volatile storage of device-specific parameters.

0xFD/0x00 Read parameters

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xFD**
3	Command (0x00) - read parameters
4	Parameters index (0-61)
5	Parameters count (1-8)
6	Reserved
7	Request ID
8 (chksm)	Checksum
9-64	Reserved

Read parameters - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xFD**
3	0x00
4	Parameters index
5	Parameters count
6	Parameters status (bit-mapped, bit indicates that the parameter is set, bit 0 corresponds to first parameter value returned)
7	Request ID
8 (chksm)	Checksum
9-62	List of parameter values

0xFD/0x01 Set parameter

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xFD**
3	Command (0x01) - set parameter
4	Parameter index
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-12	Parameter value
13-64	Reserved

Set parameter - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xFD**
3	0x01
4	Parameter index
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-12	Parameter value
13-64	Reserved

0xFD/0x02 Clear parameter

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xFD**
3	Command (0x02) - clear parameter
4	Parameter index
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-64	Reserved

Clear parameter - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xFD**
3	0x02
4	Parameter index
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-64	Reserved

PoCAN commands

Commands specific to PoCAN functionality of the firmware. PoCAN supports automatic remapping of specific PoLabs devices (PoPendant and kbd48CNC) to existing functionality of the standard PoKeys device.

0xE6/0x00 PoCAN status

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xE6**
3	Command (0x00) - status
4	Reserved
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-64	Reserved

PoCAN status - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xE6**
3	0x00
4	Enable/disable value
5	PoCAN available (if 1)
6	Reserved
7	Request ID
8 (chksm)	Checksum
9	PoPendant on CAN detected
10	kbd48CNC on CAN detected
11-64	Reserved

0xE6/0x01 Enable/disable PoCAN

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xE6**
3	Command (0x01) - enable/disable
4	Enable PoCAN operation (0x00 or 0xFF disables the functionality)
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-64	Reserved

Enable/disable PoCAN - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xE6**
3	0x01
4	Enable/disable value
5-6	Reserved
7	Request ID
8 (chksm)	Checksum
9-64	Reserved

0xE6/0x02 Configure PoCAN functions

Byte	Function
1 (header)	Header (0xBB)
2 (CMD)	**0xE6**
3	Command (0x02) - configure PoCAN functions
4-6	Reserved
7	Request ID
8 (chksm)	Checksum
9	Require PoPendant on CAN for operation
10	Require kbd48CNC on CAN for operation
9-64	Reserved

Configure PoCAN functions - Response

Byte	Function
1 (header)	Header (0xAA)
2 (CMD)	**0xE6**
3	0x02
4-6	Reserved
7	Request ID
8 (chksm)	Checksum
9	Require PoPendant on CAN for operation
10	Require kbd48CNC on CAN for operation
9-64	Reserved

Support for OSC (Open Sound Control)

PoKeys57 series Ethernet devices have a built-in support for OSC (Open Sound Control) protocol over UDP.

Receiving data

Data can be received via OSC Path /poil/<Shared slot ID>, where <Shared slot ID> is a two character ID of the target shared data slot (e.g. '/poil/00'). Either integer or float numbers are expected - in case of a float number, the value is multiplied by 100 before written to shared data slot. This data is then accessible from within PoBlocks environment.
UDP port 20066 is used for listening to OSC commands.

Sending data

Data can be sent from within PoBlocks using the PoIL system commands (or the dedicated block that wraps the system commands).

Peripheral interfaces

PoKeys devices support different peripheral interfaces, such as:

- I2C bus
- SPI bus
- 1-wire bus
- PoExtBus
- PoExtBus Smart
- PoNET
- CAN
- PoCAN

This section will provide specifications for the supported buses and the PoLabs peripherals connected to them. Description of basic I2C, SPI, 1-wire and CAN buses is omitted.

PoExtBus

PoExtBus is a five pin extension bus which is used to connect various peripheral devices to some PoLabs products. It is used to transfer signals (and optionally power) to the connected peripheral.
PoExtBus enables to extend PoKeys device outputs number for 80 additional outputs. This is acomplished using up to 10 daisy-chained 8-bit shift registers with latches (HCT595 compatible) - 80 bits of data is sent serially and latched in the external registers.
In case of PoRelay8, the original PoExtBus data frame of 80 bits is extended to 88 bits, including an 8-bit CRC value as the first byte that is sent from the PoKeys device (the support for additional CRC value is implemented in PoKeys57 series devices starting with firmware release 4.2.35). Implemented CRC uses 0x8C polynomial with seed value of 0 - 1's complement of the calculated CRC value over 10 data bytes is used as the first byte sent from PoKeys device (ignored by older PoExtBus devices that accept only the last 80 bits on the bus).

Pin 1	**Power supply 5V**
Pin 2	Ground
Pin 3	Data (+ I2C SDA)
Pin 4	Latch
Pin 5	Clock (+I2C SCL)

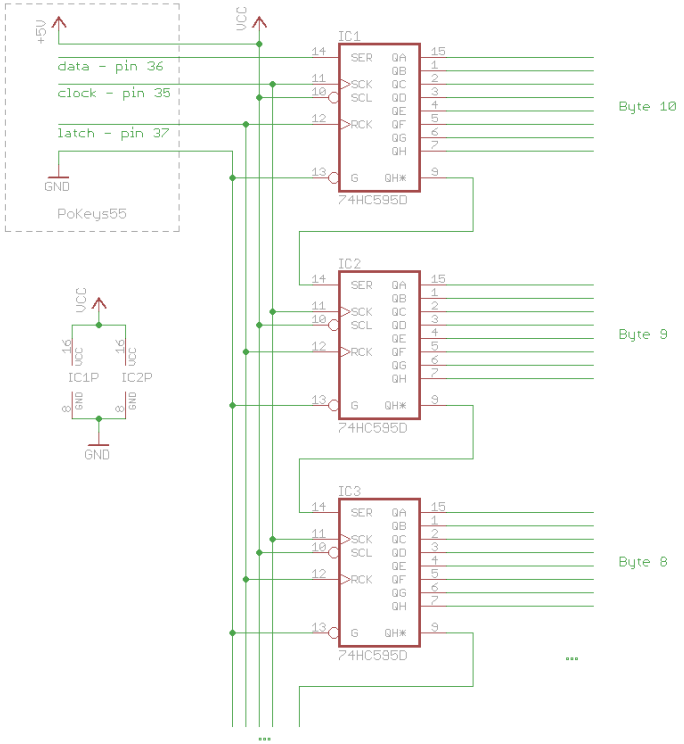


Figure 1: PoExtBus deserialization circuit

Example code for the CRC checksum function:

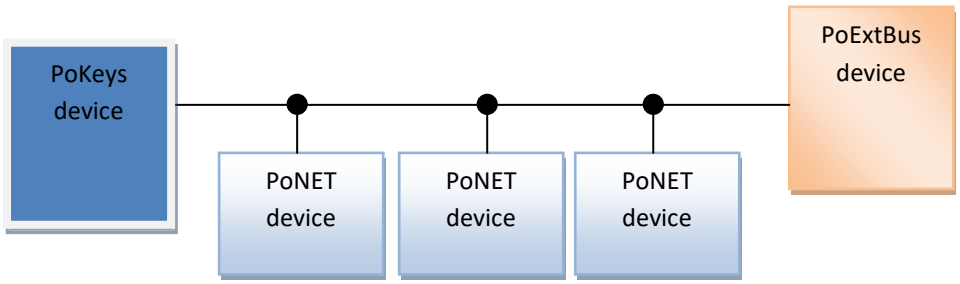
```
#define CRC_polynomial    0x8c;

uint8_t CRC8(uint8_t * p, uint8_t data_size) {
        uint8_t cd, i, fb, result = 0;

        while (data_size--) {
                cd = *p++;

                for (i = 0; i < 8; i++) {
                        fb = (result & 0x01);
                        result >>= 1;
                        if (fb ^ (cd & 0x01)) result ^= CRC_polynomial;
                        cd >>= 1;
                }
        }
        return ~result;
}
```

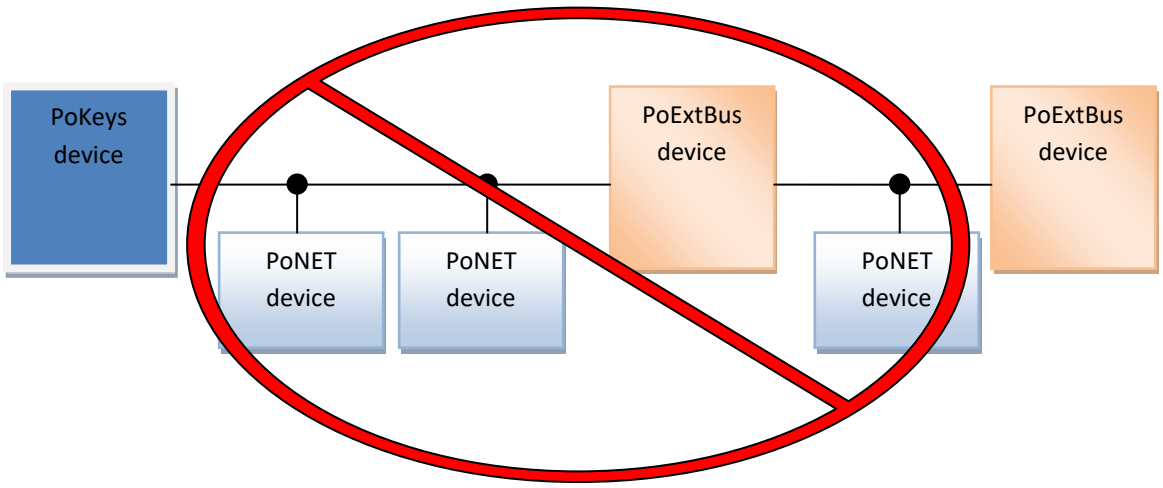
PoExtBus Smart

PoExtBus Smart devices feature backwards-compatibility with PoExtBus devices and add additional features to improve reliability and offer more configuration options. Additional communication uses I2C bus that is already available on the same PoExtBus connector. PoExtBus Smart devices allow configuration and firmware updates via the additional communication interface.
PoExtBus Smart devices can be automatically detected and identified. Unlike PoNET, PoExtBus Smart features are used by the configuration applications only, while PoNET is implemented in PoKeys devices directly.

PoNET

PoNET bus also shares the PoExtBus connector as it is based on I2C bus. Multiple PoNET and PoExtBus devices can be linked together. Since the PoNET devices and PoExtBus devices share the same ExtBus connector, user should pay attention in connecting devices of both types together (see the schematics below).





In general, all PoNET devices must be connected in parallel to each other and directly to the PoKeys device, while the PoExtBus devices should be connected in series after the PoNET devices.
The following table provides information on how different devices should be connected to PoExtBus connector of the PoKeys device.

	PoExtBus	**PoExtBusSmart**	**PoNET**
PoExtBus	daisy-chain	parallel	parallel
PoExtBusSmart	parallel	use CAN for chaining	parallel
PoNET	parallel	parallel	parallel

There are three wiring possibilities:

- **daisy-chain**: uses the pairs of PoExtBus connectors (or PoExtension connectors on newer devices) on PoExtBus devices to daisy-chain devices one after the other with the provided cables. Devices that only have a pair of white PoExtBus connectors must be positioned first in the chain (connected to PoKeys devices on one end)
- **parallel**: both devices must be connected in parallel to the PoExtBus connector on the PoKeys device. A 1-to-2 cable splice is needed (is not currently provided by PoLabs).
- **daisy-chain** over CAN: PoExtBusSmart devices are virtually chained using the CAN bus - all PoExtBusSmart devices are connected in parallel to CAN bus connector as described in this manual.

Pay attention to the following:

- PoNET device can be connected only to PoKeys device or other PoNET device
- PoExtBus device can be connected to either PoKeys board, PoNET or PoExtBus device.
- There should not be any PoExtBus device connected between PoNET devices and the PoKeys device

PoNET protocol

PoNET devices use a combination of broadcasted (general call) and directed I2C commands. During identification step, each module is assigned a unique 4-bit address that is then used when the master wants to address a specific device.
Each command is a 8-bit data, containing both the instruction (upper 4 bits) and parameter/address (lower 4 bits). Commands can contain additional parameter bytes (depending on the command type).
Devices operate in three distinct modes:

- identification mode: no address is assigned to the module, status LED blinking
- startup mode: module address was assigned before, but no I2C address was received yet
- operational mode: module has I2C address assigned and is fully operational, status LED stops blinking

Identification step

Identification step is started each time the master device boots up. The following sequence is executed:

1. Master sends command IDENT1, which instructs all connected devices to release the temporary assigned I2C address
2. Master scans the I2C bus for free addresses
3. Master assigns an I2C address to each device using SETADDR command

IDENT0 command (0xA0)

GENERALCALL+W[ACK]=>IDENT0[ACK/NACK]

IDENT0 command instructs all devices to restore all their settings to factory defaults (including stored addresses). Devices are expected to enter identification mode.

IDENT1 command (0xA1)

GENERALCALL+W[ACK]=>IDENT1[ACK/NACK]

IDENT1 command instructs all devices to enter startup mode and wait for addresses to be assigned. Devices release previously assigned I2C addresses.

IDENT15 command (0xAF)

GENERAL CALL +W[ACK]=>IDENT15[ACK/NACK] => DUMMY BYTE

IDENT15 command is used to detect modules that are still in identification mode with no address assigned. Only such devices respond with ACK after the dummy byte.

SETADDR command (0xB0)

GENERAL CALL+W[ACK]=>SETADDRx[ACK/NACK]=>I2C address[ACK/NACK]

Master sends out SETADDR command for each of the addresses x = 0…15, where the module was previously assigned. The command is acknowledged (and provided I2C address accepted) by the following devices:

- if a device is present that was assigned with the address x before
- if a device is present that hasn't been assigned with the internal address before, but the user has pressed a button on the device. Such device remembers address x as its internal address

GETPARAM command (0x20)

ADDRESS+W[ACK]=>GETPARAM[ACK]=>STOP=>ADDRESS+R[ACK]=>DATA0[ACK]=>DATA1[ACK]=>internal address[ACK]

Addressed device responds with 4 data bytes:

- Byte 0: device type (0x10 for kbd48CNC – ver1.0)
- Byte 1: bits 7…4 - number of LED columns of 8 diodes (0 if no LEDs are present, max. 16 for 16x8 LED matrix), bits 3..0 - number of keys.
- Byte 2: additional options:
    - Bit 7: light sensor present
    - Bit 6: LEDs under the keys
    - Bits 5…0: reserved
- Byte 3: internal device address

GETSTAT command (0x30)

ADDRESS+W[ACK]=>GETSTAT[ACK]=>STOP=>ADDRESS+R[ACK]=>DATA0[ACK]=>DATA1[ACK]=>...

Device returns state of the keys (bit-mapped to bytes). Number of bytes depend on the device's layout (as reported in the GETPARAM command).

SETLED command (0xC0)

ADDRESS+W[ACK]=>SETLED[ACK]=>DATA0[***]=>DATA1[***]=>...

Master sends desired state of LEDs in bit-mapped bytes.

SETPWM command (0xD0)

ADDRESS+W[ACK]=>SETPWM[ACK]=>PWM_VALUE[***]

Command controls the brightness of LEDs in the module. 8-bit PWM value is sent in the command parameter (0 = LEDs off, 255 = LEDs full on).

READLS command (0x10)

ADDRESS+W[ACK]=>READLS[ACK]=>STOP=>ADDRESS+R[ACK]=>data transfer from slave to master

Target device responds with the current value of the light level, measured by the light sensor on the module (if present).

PoCAN

PoCAN uses standard CAN messages with 11-bit IDs and 1-8 byte length (at 250 kbit/s). While device identification message is shared with multiple PoCAN devices, the exact protocol depends on the device type.

Messages with CAN id of 0x108 are interpreted as command requests and the following commands are common to all PoCAN devices.

Device identification (0x10)

Device identification returns information on the PoCAN device, including device type, firmware version and 32-bit device identifier. This identified is unique device identification used for accessing PoCAN devices.

Request

ID 0x108	Command 0x10

Response

ID 0x108	Command 0x10	TYPE_2 (1)	FW_ver_1	FW_ver_2	Device ID (32-bit)

Device types

Type reported	Device type
0x01	PoRelay8 - relay card with 8 relay outputs and 4 digital inputs
0x02	PoCAN15i - TBD
0x03	PoCAN-LIN - TBD
0x04	PoCAN15triac - TBD
0x05	kbd48CNC - 48-key keyboard for use with CNC machines
0x06	PoCAN4triac - TBD

Configuration read (0x11)

This command is used to access the settings of the PoCAN device. Only lower 16 bits of the parameter values can be accessed via CAN.

Request

ID 0x108	Command 0x11	Device ID (32-bit)	Index

Response

ID 0x108	Command 0x11	Device ID (32-bit)	Index	Value (16-bit)

See Parameter indexes table on page **Napaka! Zaznamek ni definiran.**.

Configuration write (0x12)

This command is used to access the settings of the PoCAN device. Only lower 16 bits of the parameter values can be accessed via CAN.

Request

ID 0x108	Command 0x12	Device ID (32-bit)	Index	Value (16-bit)

Configuration save (0x13)

This command is used to save the settings of the PoCAN device to non-volatile memory.

Request

ID 0x108	Command 0x13	0xA5