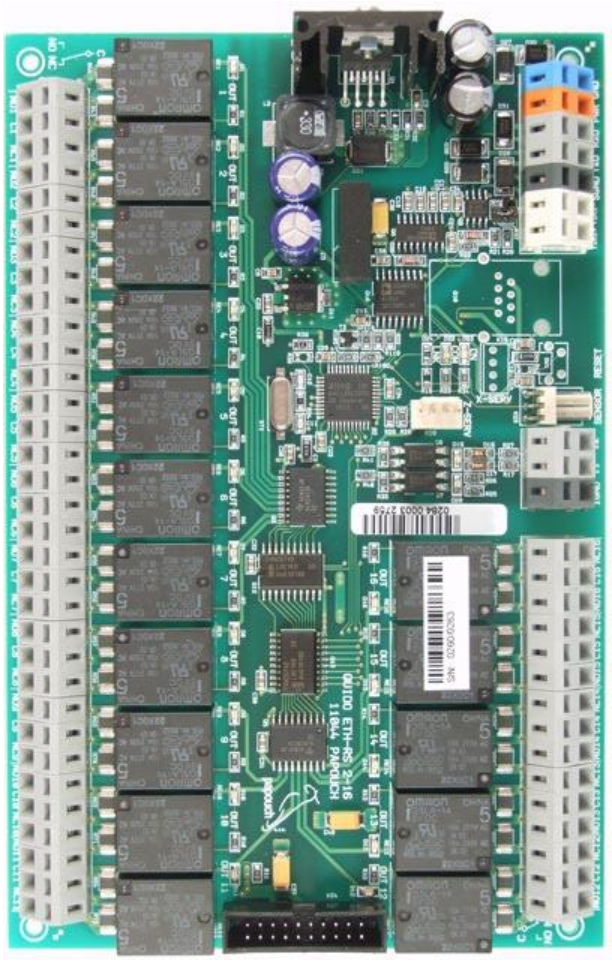
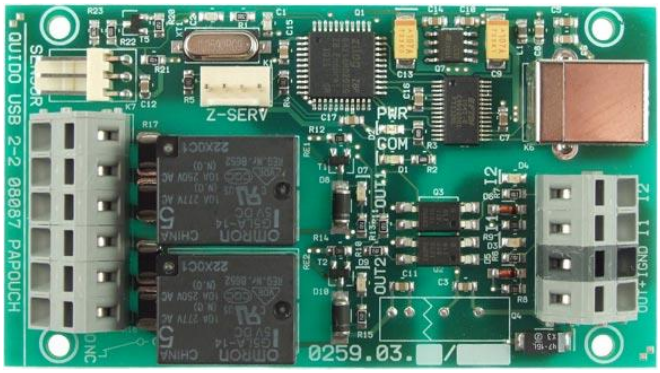
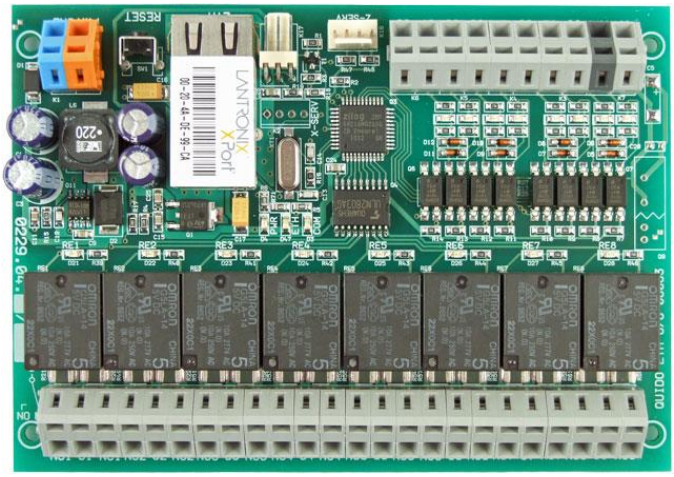
Quido Spinel

Complete description of the Quido I/O modules communication protocol







Quido Spinel

Datashet

Created: 23.11.2005

Last update: 31.5.2018 9:20

No. of pages: 52

Papouch s.r.o.

Address:

Strasnicka 3164/1a
102 00 Praha 10

Telephone:

+420 267 314 267

Fax:

+420 267 314 269

Internet:

www.papouch.com

E-mail:

papouch@papouch.com

TABLE OF CONTENTS

DOCUMENT CHANGELOG

Version 3.4

Instruction *Subtract from counter* modified so that all counters can be erased at once.

Version 3.3

New instructions: *Set automation* and *Read automation settings*, which allow for an automated output actions based on input states.

Version 3.0

- New instructions: *Measure temperature, Set temperature units, Read temperature units, Set temperature watching, Read temperature watching settings, Set sampling, Read sampling, Set binary format* timeout*, Read binary format timeout, Reset to* default.
- Instruction *Read name and version* modified to be used to search for a device on the line.
- Added option to disable ASCII format 66 and only binary protocol 97 is available. (*Switch the communication* protocol).
- Quido sends communication parameters by shorting contacts on the PCB (see page 5).

Version 2.8

- Mask parameter added to instructions Setting autonomous sending and Reading autonomous sending settings. This means you can chose which inputs are monitored. This is useful if for example some Quido inputs are used as counters (with fast changes) and other inputs are used as door contacts.

Version 2.1

- Fixed error in instruction 33H (Reading output switching for a period). When inputting 00H parameter (out -reading all outputs setting at once) a correct response was sent, but with ACK 03H.
- Fixed an FDH address error. An FDH address was incorrectly considered a universal address.

Version 2.0

- “Set outputs for a certain period“ instruction modified. Apart from “OT”, “OST” can be sent as an instruction code when using Spinel 66 protocol.
- Added – instructions to read and save user data (as names and such) for each input and output.
- New option to save length and polarity of each pulse. A different instruction is used to execute the pulse.
- Added instruction to get the output mode more quickly.
- Instructions were rearranged and re-numbered based on types.

QUIDO MODUES COMMUNICATION PARAMETERS

RS232 and RS485 interfaces

Communication speed.....selectable 300 Bd to 230400 Bd

Default communication speed.....9600 Bd

Data bits 8

Parity .....no parity

Stop bits.....1

USB interface

Communication speed .....115200 Bd (not changeable)

Data bits 8

Parity .....no parity

Stop bits.....1

Ethernet interface

Communication speed .....115200 Bd (not changeable)

Data bits 8

Parity .....no parity

Stop bits.....1

How do I find out current communication parameters?

Short the two marked pins on the Z-SERVICE connector and Quido will send the current communication parameters to the serial line. This information is always sent via Spinel protocol. RS version of Quido sends these on 9600 Bd, USB and ETH versions send these on 115,2 kBd.

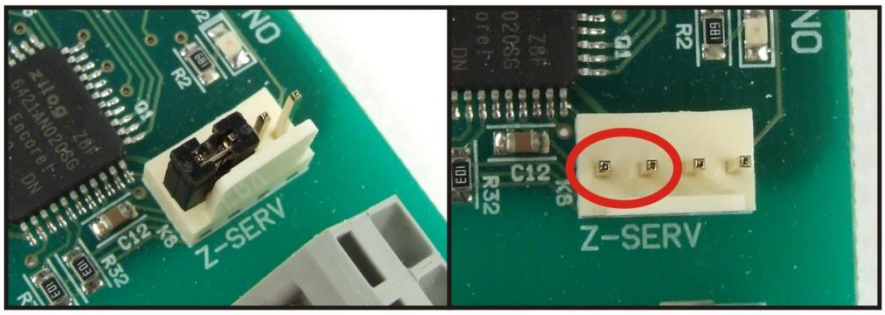


fig. 1 – shorten the two left pins on the Z-SERV connector

The device will send a response to the *Read name and version* instruction and then a packet containing Address, speed and protocol in ASCII format. Example:

**a?"4N?Address:34 Speed:6 Protocol:1ü?*

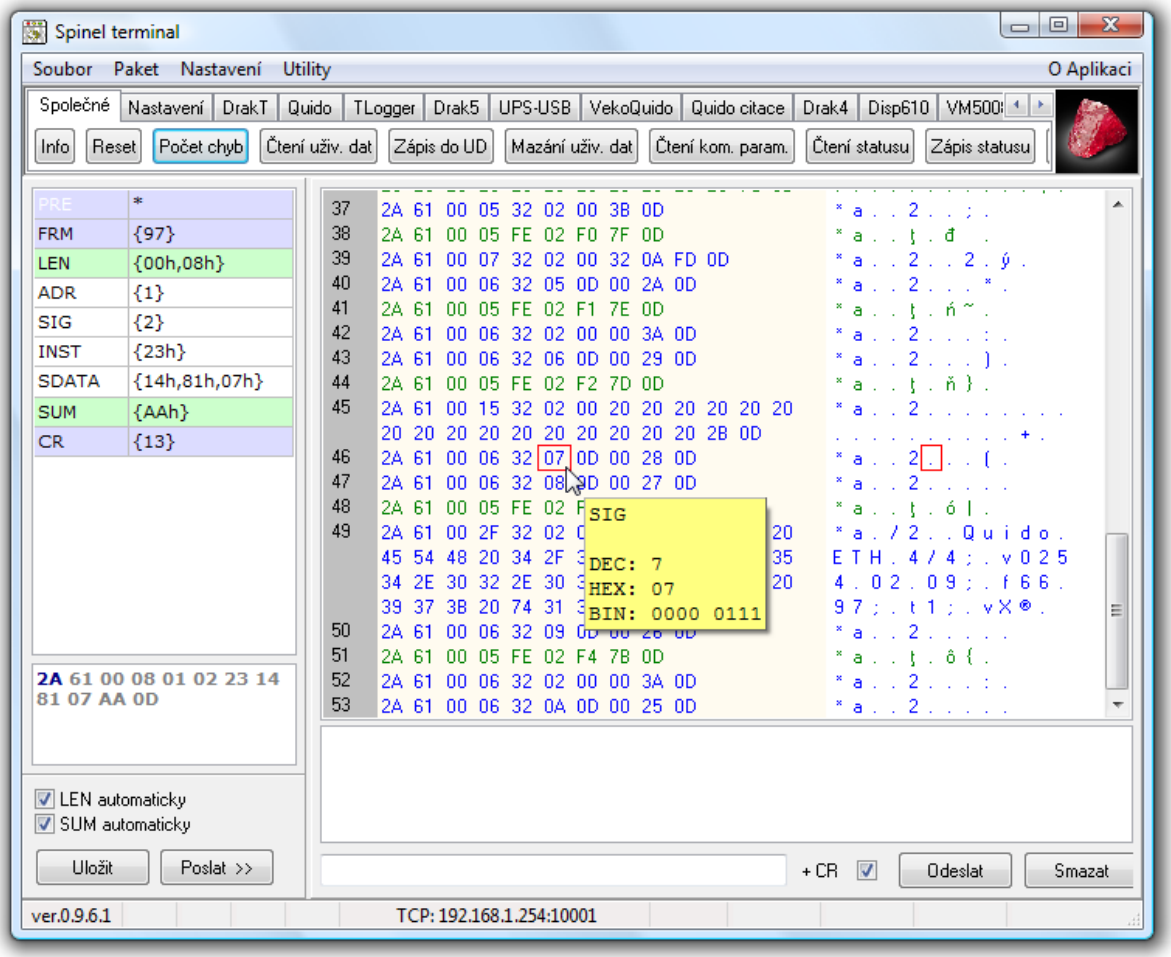
Address is in hexadecimal format, speed is a number according to *Set communication parameters* instruction and protocol is a number according to the *Switch the communication protocol* instruction.

DESCRIPTION

This document describes Quido I/O Modules communication protocol. Inputs and output counts, communication speeds and interface description is described in a separate document of a particular Quido I/O module. Any exceptions are described at each particular instruction.

Spinel Terminal

Spinel terminal can be used for an easy debugging. It is available for free download at http://spinel.papouch.com. It can use Ethernet and serial ports using Binary protocol Spinel Format 97.



HOW TO EASILY CONTROL QUIDO - EXAMPLES

The following examples will work with a module with default settings. Send the Request string using your control program. (The individual characters must be less than 5s apart). Once everything is OK, the module will respond as shown in the Response column.

(Examples are written in Spinel 66 format for easy reading. This format is suitable for getting to know the module, debugging and the terminal communications. Format Spine 97 is more suited to control the module via your program or script. Description of format 97 starts on page 10.)

Switching a relay ON

The following example will switch ON relay No.2 on a module using address 1.

Request	Response		Description
***B1OS2H↵**		*B	Prefix
		1	Address You can also use **$** character. This character acts as a universal address and only works when a single device is on the line.
		OS	Change output state instruction code
		2	Output number
		H	ON code (High)
		↵	End character (enter)
	***B10↵**	*B	Prefix
		1	Module address
		0	Acknowledge
		↵	Change output state instruction code

Switching a relay OFF

The following example will switch ON relay No.4 on a module using address D.

Request	Response		Description
***BDOS4L↵**		*B	Prefix
		D	Address You can also use **$** character. This character acts as a universal address and only works when a single device is on the line.
		OS	Change output state instruction code
		4	Output number
		L	OFF code (Low)
		↵	End character (enter)
	***BD0↵**	*B	Prefix
		D	Module address
		0	Acknowledge
		↵	End character (enter)

Read input state

An example of reading input No. 3 on a single module connected to the line using the universal address.

Request	Response		Description
***B$IR3↵**		*B	Prefix
		$	Universal address
		IR	Read input state instruction code
		3	Input number
		↵	End character (enter)
	***B10H↵**	*B	Prefix
		1	Module address
		0	Acknowledge
		H	Input is active
		↵	End character (enter)

Changing the address

This instruction changes the modules address from **f** to **5**.

Request	Response		Description
First we have to allow configuration using a special instruction. This instruction allows configuration for a single subsequent instruction. Configuration is then disallowed again.			
***BfE↵**		*B	Prefix
		F	Address
		E	Allow configuration instruction code
		↵	End character (enter)
	***Bf0↵**	*B	Prefix
		F	Module address
		0	Acknowledge
		↵	End character (enter)
Now that the configuration is allowed, we can send the instruction to change the address.			
***BfAS5↵**		*B	Prefix
		F	Old address
		AS	Change address instruction code
		5	New address
		↵	End character (enter)
	***Bf0↵**	*B	Prefix
		f	Old address
		0	Acknowledge
		↵	End character (enter)

BASIC INSTRUCTION LIST

Description	Code [Request] [Response]	Example (Address is always 1)
Read input	*B[address]IR[input][1] ↲ *B[address]0[state][2]↲	*B1IR2↲ *B10H↲
Read output	*B[address]OR[output] [3]↲ *B[address]0[state][4]↲	*B1OR4↲ *B10L↲
Set output	*B[address]OS[output][3][state][4]↲ *B[address]0↲	*B1OS3H↲ *B10↲
Set output timing	*B[address]OT[output][3][state][4][time][5]↲ *B[address]0↲	*B1OT1H20↲[6] *B10↲
Request device name and type	*B[address]?↲ *B[address]0Quido [line][7] [I/O][8]; v[VC][9]; F66 97↲	
Allow configuration[10]	*B[address]E↲ *B[address]0↲	*B1E↲ *B10↲
Set address[11]	*B[old Address]AS[new Address]↲ *B[old Address]0↲	*B1AS5↲ *B10↲

Notes:

[address] … Instead of [address] a $ character can be used. This represents a universal address you can use to find your modules address. Universal address can only be used when a single device is on the line and you do not need to address it.

[address] … % character can also be used. That is a broadcast address meaning all modules on the line will perform the requested action will send nothing in response to avoid collisions on the line.

Comm. speed Bd	Code
110	0
300	1
600	2
1200	3
2400	4
4800	5
9600	6
19200	7
38400	8
57600	9
115200	A
230400	B

[1] Numbers 0 to 4. Once 0 is entered, all input states will be sent at once.

[2] L – input inactive (OFF) ; H – input active (ON)

[3] Numbers 1 to 4.

[4] L – relay is off; H – relay is on

[5] Selected output ON / OFF time. Enter number from 1 to 255. Unit is 0,5 sec. You can effectively set time from 0,5 sec. to 127,5 sec.

[6] Turns output 1 ON for 10 sec. (10 sec. = 20 * 0,5).

[7] Communication interface specification (USB, ETH or RS).

[8] Number of inputs / outputs (for example 10/1 for version with 10 inputs and 1 output).

[9] Device production number (for example 0227.02.02).

[10] This instruction can't be used with universal address $.

[11] This instruction must be preceded by Allow configuration instruction.

COMPLETE DESCRIPTION OF THE COMMUNICATION PROTOCOL

All Quido modules have a standardized protocol Spinel[12] implemented in them, specifically formats 66 (ASCII) and 97 (binary).

Format 97

Format 97 uses 8bit characters to communicate (ranging from 0 to 255 in decimal). To debug your communication easily, use Spinel Terminal software. Instructions are divided to request and response:

Structure

Request:

```
PRE FRM NUM NUM ADR SIG INST DATA… SUMA CR
```

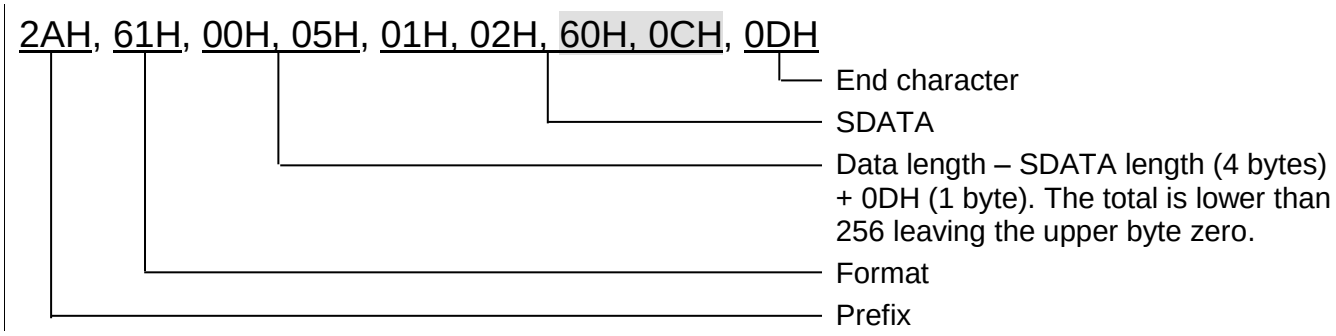
Response:

```
PRE FRM NUM NUM ADR SIG ACK  DATA… SUMA CR
```

PRE	1 Byte	Prefix, 2AH (character "*").
FRM	1 Byte	Format number 97 (61H).
NUM	2 Bytes	Number of instruction bytes from the following byte till the end of the frame.
ADR	1 Byte	Address of the module receiving request or sending response.
SIG	1 Byte	Message signature – any number from 00H to FFH. The same number sent in request will be returned in response determining to which request the response was received.
INST[13]	1 Byte	Instruction code
ACK	1 Byte	Request acknowledge. Determines if and how the request was executed. ACK ranges from 00H to 0FH.
DATA[13]	n Byte	Data.
SUMA	1 Byte	Check sum.
CR	1 Byte	End character (0DH).

Glossary

Example



[12] Detailed information about spinel protocol can also be found at: spinel.papouch.com .

[13] Instructions and data are highlighted like this in all following examples for easier read.

Data length (NUM)

16-bit value determining the number of bytes before the end of the instruction; number of all bytes following NUM, until CR (inclusive). Reaches values from 5 to 65535. If it is lower than 5, the instruction is considered invalid and the response will come with ACK "incorrect data" (provided the request is meant for the given device).

NUM creation process:

Add up the number of bytes following both NUM bytes (meaning number of bytes SDATA + 1 byte CR). Form the result in a 16-bit number. Divide it to an upper and lower byte. First NUM byte is the upper byte and second NUM byte is the lower one. (If the number of bytes is lower than 256, the first NUM byte will be 00H.)

Address (ADR)

Address FFH is reserved for broadcasts. If the request contains FFH address, the device considers it its own address. No responses are sent to requests with this address.

Address FEH is a universal address. If the request contains FFH address, the device considers it its own address. The response will contain actual address of the given device. Universal address can only be used when a single device is connected to the line.

Request Acknowledge (ACK)

ACK informs the controlling device how the received instruction was processed. Acknowledge codes:

00H ..... ALL IS OK
Instruction was received OK and properly executed.

01H ..... OTHER ERROR
Unspecified device error.

02H ..... INVALID INSTRUCTION CODE
Received instruction code is unknown.

03H ..... INVALID DATA
Data is of incorrect length or value.

04H ..... WRITE NOW ALLOWED / ACCESS DENIED
- Request was not executed because some criteria were not met.
- Attempt to write data to inaccessible memory block.
- Attempt to activate a function requiring other settings (e.g. higher comm. speed).
- Changing configuration without "*allow configuration*" instruction preceding the denied instruction.
- Access to a memory block protected by a password.

05H ..... DEVICE MALFUNCTION
- Malfunction requiring a service.
- Internal memory error of settings memory error.
- Internal peripheral device malfunction (run or initialization error).
- Any other error making the device inoperable.

06H ..... NO DATA AVAILABLE

0DH..... AUTOMATED INSTRUCTION – DIGITAL INPUT STATE CHANGE

0EH..... AUTOMATED INSTRUCTION – CONTINUOUS MEASUREMENT
- Periodic sending of measured values.

Checksum (SUMA)

1 Byte. The sum of all bytes in the instruction (all the data are added up except for the CR) subtracted from 255. Calculation: SUMA = 255 – (PRE + FRM + NUM + ADR + SIG + ACK (INST) + DATA)

No response will be sent to a request with incorrect checksum. (CR is expected even if the checksum is incorrect.)

Format 66

Format 66 uses only decimal variables or characters writeable on a regular keyboard. This format is ideal for debugging applications with spinel. Space between two characters cannot exceed 5 seconds. Instruction are divided to request and response:

Structure

Request:

```
PRE FRM ADR INST  DATA… CR
```

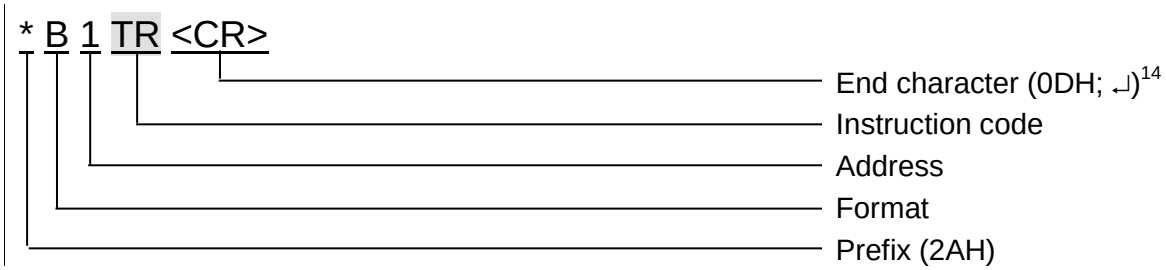
Response:

```
PRE FRM ADR ACK   DATA… CR
```

PRE	Prefix, 2AH (character "*").
FRM	Format number 66 (character "B").
ADR	Address of the module, receiving request or sending a response.
INST[13]	Instruction code - instruction codes of the given device. These are ASCII codes of letters "A" to "Z" and "a" to "z" and numbers "0" to "9".
ACK	Request acknowledge. Determines if and how the request was executed. ACK ranges from 00H to 0FH.
DATA[13]	Data. ASCII representation of the transmitted variables. It is recommended to transmit data in the usual format and units. It cannot contain prefix and CR.
CR	End character (0DH).

Glossary

Example – one time measurement



Address (ADR)

Address is one character that determines a specific device among the other devices on the communication line. That device always uses this address in response to all requests from the controlling system. Following ASCII characters can be address: numbers "0" to "9", lowercase letters "a" to "z" and uppercase letters "A" to "Z". Address cannot be identical to the prefix or CR.

Address „%" is reserved for broadcast. If the request contains "%" address, the device considers it its own address. No responses are sent to requests with this address. Address "$" is a universal address. If the request contains "$" address, the device considers it its own address. The response will contain devices actual address. Universal address can only be used when a single device is connected to the line.

[14] **End character <CR> is not shown** in examples from the chapter **Chyba! Nenalezen zdroj odkazů.** ! (It is ubstituted by character ↵ .)

Instruction code (INST)

Instruction code of the given device.

Once a valid instruction is received (ADR is correct) and contains a correct prefix, the device must respond to the instruction.

Request acknowledge (ACK)

ACK informs the controlling device how the received instruction was processed. Acknowledge codes:

0.....ALL IS OK
Instruction was received OK and properly executed.

1.....OTHER ERROR
Unspecified device error.

2.....INVALID INSTRUCTION CODE
Received instruction code is unknown.

3.....INVALID DATA
Data is of incorrect length or value.

4.....WRITE NOW ALLOWED / ACCESS DENIED
- Request was not executed because some criteria were not met.
- Attempt to write data to inaccessible memory block.
- Attempt to activate a function requiring other settings (e.g. higher comm. speed).
- Changing configuration without "*allow configuration*" instruction preceding the denied instruction.
- Access to a memory block protected by a password.

5.....DEVICE MALFUNCTION
- Malfunction requiring a service.
- Internal memory error of settings memory error.
- Internal peripheral device malfunction (run or initialization error).
- Any other error making the device inoperable.

6.....NO DATA AVAILABLE

D .....AUTOMATED INSTRUCTION – DIGITAL INPUT STATE CHANGE

E .....AUTOMATED INSTRUCTION – CONTINUOUS MEASUREMENT
- Periodic sending of measured values.

Data (DATA)

Data instruction.

COMPLETE INSTRUCTIONS OVERVIEW

Supplemental

To make the instructions more legible, only Instruction (INST) portion is shown. Acknowledge (ACK), Address (ADR), signature (SIG) and CheckSum (SUMA) are described elsewhere.

Indexes [97] or [66] before selected paragraphs show the protocol it describes. Paragraphs with no index shown are applicable for both 97 and 66 protocols. (See also footnote 14 On page 12.)

Inputs

Important: If there is no input on the module, the response will be ACK 02H (invalid instruction).

Read inputs state

Description: Instruction reads input states.

[97]Request: 31H

[97]Response: (ACK 00H)(inputs state)

[97]Legend (1 to 8) – Quidos with one to eight inputs:

(Inputs state) 1 byte in format: 87654321, where bits 1 to 8 indicates input number. Individual bits value determines the logical values of inputs. (Bits with inputs absent on a specific model will always be 0 value.)

[97]Legend (9 to 16) – Quidos with nine to sixteen inputs:

(Inputs state) 2 bytes in format: [$^{16}_{15}$ $^{14}_{13}$ $^{12}_{11}$ $^{10}_{9}$][$^{8}_{7}$ $^{6}_{5}$ $^{4}_{3}$ $^{2}_{1}$], where bits 1 to 16 indicate the number of input. Individual bits value determines the logical values of inputs. (Bits with inputs absent on a specific model will always be 0 value.)

[97]Legend (17 to 32) – Quidos with seventeen to thirty two inputs:

(Inputs state) 4 bytes in format: [$^{32}_{31}$ $^{30}_{29}$ $^{28}_{27}$ $^{26}_{25}$][$^{24}_{23}$ $^{22}_{21}$ $^{20}_{19}$ $^{18}_{17}$][$^{16}_{15}$ $^{14}_{13}$ $^{12}_{11}$ $^{10}_{9}$][$^{8}_{7}$ $^{6}_{5}$ $^{4}_{3}$ $^{2}_{1}$], where bits 1 to 32 indicates the number of input. Individual bits value determines the logical values of inputs. (Bits with inputs absent on a specific model will always be 0 value.)

[97]Legend (33 to 100) – Quidos with thirty three to a hundred inputs:

(Inputs state) 13 bytes in format: [$^{104}_{103}$ $^{102}_{101}$ $^{100}_{99}$ $^{98}_{97}$][12B][11B][10B][9B][8B][7B][6B][5B] [4B][3B][2B][$^{8}_{7}$ $^{6}_{5}$ $^{4}_{3}$ $^{2}_{1}$], where bits 1 to 104 indicates the number of input. Individual bits value determines the logical values of inputs. (Bits with inputs absent on a specific model will always be 0 value.)

[97]*Example: Read inputs state, address 01H, signature 02H*

2AH,61H,00H,05H,01H,02H,31H,3BH,0DH

Response – inputs 2, 7 and 8 are in v log. 1, other are in log. 0

2AH,61H,00H,06H,01H,02H,00H,C2H,A9H,0DH

---

[66]Request: „IR“(input) (Input Read)

[66]Response: (ACK „0“)(state)

[66]Legend: (input) Number of input – for example character „1“ (for input 1), „8“ (for 8), „15“, „32“, etc.

(state) Input is on („H“) or off („L“).

[66]*Example: Request – input 29*

**B1IR29↵*

Response – input 29 off

**B10L↵*

Set automated sending

Description: Enables or disables automated message sending upon input state change. This instruction allows informing the controlling system automatically about in input state change. It is not necessary to request a specific input state repeatedly from the device. (Default setting for this function is off.)[15]

Mask parameter can filter certain inputs that need to be monitored. (This function is only in the 97 format; default setting for is on for all inputs.)

[97]Request: 10H(state)(mask)

[97]Response: (ACK 00H)

[97]Automated response: (ACK 0DH)(inputs state)

[97]Legend: (state) 1 byte; 00H = automated sending disabled, 01H = enabled

(mask) Bit mask (for each input one bit), determining whether the given input should be monitored (bit 1) or not (bit 0). Mask parameter is not mandatory. It can be completely left out if not needed. Mask settings are written in the internal memory. Once mask settings are saved, Quido will remember it indefinitely and use it even if it is not repeatedly entered.

(input state) Once the automated sending is enabled, each logical input state change will trigger a message to the controlling system with input states. Message is in following format: *(ACK 0DH)(inputs state)* where (ACK 0DH) is a sign of automated response and (inputs state) is current state of all inputs. 01H is sent as a signature. Automated message is then sent in the same format that is defined in instruction „Set automated sending“. We recommend enabling automated sending only in case there is a single device on the line (with RS485 devices). Default setting is off.

[97]*Example: Enable automated sending for inputs 1 and 2; address 01H, signature 02H:*

```
2AH,61H,00H,07H,31H,02H,10H,01H,03H,26H,0DH
```

Response:

```
2AH,61H,00H,05H,31H,02H,00H,3CH,0DH
```

Automated response (input 1 has been activated):

```
2AH,61H,00H,06H,31H,02H,0DH,01H,2DH,0DH
```

[66]Request: „IS“(state)

[66]Response: (ACK „0“)

[66]Legend: (state) Enable („1“) or disable („0“) automated sending.

[66]*Example: Request – enable automated sending:*

```
*B1IS1↵
```

Response:

```
*B10↵
```

Automated response – example of Quido with eight inputs: Input number 7 is on. Inputs are divided into groups of five for better legibility.

```
*B1D LLLLL LHL↵
```

[15] Quido ETH 3/0B always sends info about all three inputs when this function is enabled.

Read automated sending settings

Description: Reads automated sending settings.

[97]Request: 11H

[97]Response: (ACK 00H)(state)(mask)

[97]Legend: (state) 1 byte; 00H = automated sending disabled; 66D (42H) = enabled via format 66; 97D (61H) = enabled via format 97

(mask) Bit mask (for each input one bit), determining whether the given input should be monitored (bit 1) or not (bit 0).

[97]*Example: Enable automated sending; address 31H, signature 02H*

2AH,61H,00H,05H,31H,02H,11H,2BH,0DH

Response – automated sending is enabled via format 97 (61H) and inputs 1 and 2 are monitored:

2AH,61H,00H,07H,31H,02H,00H,61H,03H,D6H,0DH

[66]Request: „IX“

[66]Response: (ACK „0“)(state)

[66]Legend: (state) „0“ – automated sending disabled; „B“ – automated sending is enabled via format 66; „a“ – automated sending is enabled via format 97

[66]*Example: Request*

**B1IX↵*

Response – automated sending is enabled via format 66

**B10B↵*

Set sampling

Description: Device samples its inputs with an interval of 1 ms. Once the set number of subsequent samples is consistent, it is considered a state change of the given input.

Default value is 20. It is possible to set values from 1 to 255 ms.

Quido ETH 3/0B does not support this instruction.

[97]Request: 62H(ms-count)

[97]Response: (ACK 00H)

[97]Legend: (ms-count) 1 byte; number of consistent consequent samples that is to be considered a valid state change.

[97]*Example: Request – set 10 samples:*

2AH,61H,00H,06H,B1H,02H,62H,0AH,4FH,0DH

Response:

2AH,61H,00H,05H,B1H,02H,00H,BCH,0DH

Read sampling

Description: Reads set number of samples for input sampling.

Quido ETH 3/0B does not support this instruction.

[97]Request: 63H

[97]Response: (ACK 00H)(ms-count)

[97]Legend: (ms-count) 1 byte; number of consistent consequent samples that is to be considered a valid state change.

[97]*Example: Request:*

2AH,61H,00H,05H,B1H,02H,63H,59H,0DH

Response:

2AH,61H,00H,06H,B1H,02H,00H,0AH,B1H,0DH

Read counters

Description: Instruction reads the state of input counters. Counter allows for recording the number of changes on input. Each input has its own counter. One is added to the counter value each time it changes from 0 to 1 or vice versa.[16]

[97]Request: 60H(parametr1)…(parameterN)

[97]Response: (ACK 00H)(bit-count)(counter-state1)…(counter-stateN)

[97]Legend:

(parameter)	1 byte in format Cxnnnnnn; If the bit C=1, the counter will be reset after a read-out. If the C=0, counter will only be read and its state left without changes. If the n>0, only counter with the stated number will be read. If the n=0, all counters will be read. (Once n=0, only one parameter must be sent (parameter). Number n is 60 max.)
(bit-count)	1 byte; information about the resolution of counters (binary number 8, 16, 24 or 32 bits)
(inputs-state)	one or more bytes containing the value of counter/counters, according to information from (bit-count)

[97]*Example: Read counters*

2AH,61H,00H,06H,31H,02H,60H,00H,DBH,0DH

Response (example from module Quido ETH 10/1)

2AH,61H,00H,1AH,31H,02H,00H,10H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,17H,0DH

---

[66]Request: „CR“(parameter)(counter) (Counter Read)

[66]Response: (ACK „0“)(state)

[66]Legend:

(parameter)	If it equals “1”, the counter will be reset after a read-out. If it is ”0”, counter will only be read and its state left without changes.
(counter)	Counter number – for example „1“, „5“, „12“.
(state)	Value of counter/counters.

[66]*Example: Request – counter 5 – counter fill be reset*

**B1CR15↵*

Response – counter has value 230 (decimal)

[16] Counter states are not preserved during power off or reset.

**B10230↵*

Subtract from counter

Description: Instruction subtracts given value from current counter state.[16]

This function prevents data loss during the instruction “Read counters” along with deleting the counter state. If there were a change during the read process, it would be deleted right after reading the value. (If the change occurs right between reading and deleting the counter state)

This can be prevented by using the instruction “Subtract from counter“. Use “Read counters” instruction first and then subtract the value you have read from the counter. Its state can be increased during this process and no change is lost.

Higher than current counter state value cannot be subtracted.

One instruction can manage up to 12 inputs at once.

If only parameter counter=0 and at the same time value=0, all counters will be deleted at once.

[97]Request: 61H(counter1)(value1)… (counterN)(valueN)

[97]Response: (ACK 00H)

[97]Legend: (counter) 1 byte; determines counter number (1 to 60)

(value) 2 bytes – value to be subtracted from counter state

[97]Example: Request – counter 2, subtract value 1:

2AH,61H,00H,08H,31H,02H,61H,02H,00H,01H,D5H,0DH

Response:

2AH,61H,00H,05H,31H,02H,00H,3CH,0DH

[66]Request: „CD“(counter)(value)

[66]Response: (ACK „0“)

[66]Legend: (counter) Counter number – for example „01“, „05“, „12“. (Number is always two digits.)

(value) Value to be subtracted from counter state.

[66]Example: Request - counter 2, subtract value 1:

**B1CD021↵*

Response

**B10↵*

Set counters

Description: Instruction sets the parameters of the counters changes on inputs. [16]

Counter allows user to count changes of input state. Change of state is considered a logic change from “0” to “1” or “1” to “0”. Each input has its counter with distinct parameters. Counter can be increased upon change from “0” to “1”, change from “1” to “0” or both.

[97]Request: 6AH(parameter1)…(parameterN)

[97]Response: (ACK 00H)

[97]Legend: (parameter) 1 byte in format CCnnnnnn; nnnnnn determines counter number (0 to 63). If it is je 0, mode set by CC bits applies to all counters.

CC determines counter mode:

CC=00 … counter is off.

CC=10 … counter increases its value by one upon every leading edge on a given input.

CC=01 … counter increases its value by one upon every falling edge on a given input.

CC=11 … counter increases its value by one upon any edge on a given input (both leading and falling).

[97]*Example: Set counters*

```
2AH,61H,00H,06H,31H,02H,6AH,80H,51H,0DH
```

Response – ok

```
2AH,61H,00H,05H,31H,02H,00H,3CH,0DH
```

[66]Request: „CO“(parameter)(counter) (Counter Options)

[66]Response: (ACK „0“)

[66]Legend: (counter) Counter number – for example „1“, „5“, „12“. Number „0“ allows user to set all counters.

(parameter)„0“ … counter is off.

„1“ … counter increases its value by one upon every leading edge on a given input.

„2“ … counter increases its value by one upon every falling edge on a given input.

„3“ … counter increases its value by one upon any edge on a given input (both leading and falling).

[66]*Example: Request – counter 5, counting upon leading edge*

```
*B1CO15↵
```

Response

```
*B10↵
```

Read counters settings

Description: Instruction reads input counters settings. [16]

[97]Request: 6BH(parameter1)…(parameterN)

[97]Response: (ACK 00H)(setting1)… (settingN)

[97]Legend: (parameter) 1 byte; determines counter number (0 to 63). If it is 0, all counters are read.

(setting) 1 byte in format CCnnnnnn; nnnnnn is counter number (1 to 60).

CC determines counter mode:

CC=00 … counter is off.

CC=10 … counter increases its value by one upon every leading edge on a given input.

CC=01 … counter increases its value by one upon every falling edge on a given input.

CC=11 … counter increases its value by one upon any edge on a given input (both leading and falling).

[97]Example: Read settings – counters 1, 5, 7 and 9:

`2AH,61H,00H,09H,31H,02H,6BH,01H,05H,07H,09H,B7H,0DH`

Response – counter 1: leading edge, counter 5: both edges, counters 7 and 9: falling edge:

`2AH,61H,00H,09H,31H,02H,00H,81H,C5H,47H,49H,62H,0DH`

[66]Request: „CX“(counter)

[66]Response: (ACK „0“)(state)

[66]Legend: (counter) Counter number – for example „1“, „5“, „12“.

(state) „0“ … counter is off.

„1“ … counter increases its value by one upon every leading edge on a given input.

„2“ … counter increases its value by one upon every falling edge on a given input.

„3“ … counter increases its value by one upon any edge on a given input (both leading and falling).

[66]Example: Request – counter 3

`*B1CX3↵`

Response – counter on input 3 counts leading edges

`*B11↵`

Outputs

Important: If there is no output on the module, the response will be ACK 02H (invalid instruction).

Read outputs

Description: Instruction reads current outputs states (relays).

[97]Request: 30H

[97]Response: (ACK 00H)(state OUT)

[97]Legend (1 to 8) – Quidos with one to eight outputs:

(state OUT) 1 byte; byte in format: 87654321, where bits 1 to 8 determine the number of output. Outputs with bit values 1 are turned on. (Bits with outputs that are not used on a given module are always 0.)

[97]Legend (9 to 16) – Quidos with eight to sixteen outputs:

(state OUT) 2 bytes; bytes in format: [$^{16}_{15}\ ^{14}_{13}\ ^{12}_{11}\ ^{10}_{9}$][$^{8}_{7}\ ^{6}_{5}\ ^{4}_{3}\ ^{2}_{1}$], where bits 1 to 16 determine the number of output. Outputs with bit values 1 are turned on. (Bits with outputs that are not used on a given module are always 0.)

[97]Legend (17 to 32) – Quidos with seventeen to thirty two outputs:

(state OUT) 4 bytes; bytes in format: [$^{32}_{31}\ ^{30}_{29}\ ^{28}_{27}\ ^{26}_{25}$][$^{24}_{23}\ ^{22}_{21}\ ^{20}_{19}\ ^{18}_{17}$][$^{16}_{15}\ ^{14}_{13}\ ^{12}_{11}\ ^{10}_{9}$][$^{8}_{7}\ ^{6}_{5}\ ^{4}_{3}\ ^{2}_{1}$], where bits 1 to 32 determine the number of output. Outputs with bit values 1 are turned on. (Bits with outputs that are not used on a given module are always 0.)

[97]*Example: Read relay state, address 01H, signature 02H*

```
2AH,61H,00H,05H,01H,02H,30H,3CH,0DH
```

Response – relays 1 and 5 on

```
2AH,61H,00H,06H,01H,02H,00H,11H,5AH,0DH
```

[66]Request: „OR“(output) (Output Read)

[66]Response: (ACK „0“)(state)

[66]Legend: (output) Output number – for example character „1“ (for output 1), „8“ (for 8), „15“, „32“, etc.

(state) Selected output is on („H“) or off („L“).

[66]*Example: Request*

```
*B1OR14↵
```

Response – relay 14 turned on

```
*B10H↵
```

Set outputs

Description: Basic instruction to control inputs – immediate on or off switching.

[97]Request: 20H (OUTx)…(OUTy)

[97]Response: (ACK 00H)

[97]Legend (OUTx) 1 byte; byte in format: SOOOOOOO, where "S" is the state to which the output is to be set (1 = on; 0 = off) and "O" is a number of the output (binary representation of number 1 to 127). Instruction can contain more of these bytes regardless of their order.

[97]Example: Set output 2, address 01H, signature 02H

2AH,61H,00H,06H,01H,02H,20H,82H,C9H,0DH

Response

2AH,61H,00H,05H,01H,02H,00H,6CH,0DH

[66]Request: „OS"(output)(state) (Output Set)

[66]Response: (ACK „0")

[66]Legend: (output) Output number – for example character „1" (for output 1), „8" (for 8), „15", „32", etc.

(state) Switch on („H") or switch off („L") the chosen output.

[66]Example: Request – switches relay 25 on

**B1OS25H↵*

Response

**B10↵*

Set outputs for a certain period

Description: Instruction activates chosen outputs for a certain period of time – runs a pulse of given polarity for a set time. The pulse will start right after receiving this instruction. Recurring pulse initiation is possible, even if the previous pulse did not end on time.

(Length of the pulse set by this instruction will not affect he length of pulse set by instruction "Set pulse length on an outputSet pulse length on an output" on page 26.)

[97]Request: 23H(time)(OUTx)…(OUTy)

[97]Response: (ACK 00H)

[97]Legend: (time) 1 byte; period of time in which all the following outputs should be switched on. Ranges from 1 to 255, 1 unit equals 0.5 sec.

(OUTx) 1 byte; byte in format: SOOOOOOO, where "S" is the state to which the output is to be set (1 = on; 0 = off) and "O" is a number of the output (binary representation of number 1 to 127). In case the relay is already on before the instruction, it will stay on and switch off after the given time. The same applies for the off state. Instruction can contain maximum of 12 outputs or as many outputs as there are on your device regardless of their order.

[97]Example: Switch relay 1 and 4 for 2 seconds, address 35H, signature 02H

2AH,61H,00H,08H,35H,02H,23H,04H,81H,84H,09H,0DH

Response

2AH,61H,00H,05H,35H,02H,00H,38H,0DH

[66]Request: „OT“(output)(state)(time) (Output Timing)

„OST“ (output)(state)(time) (Output Set Timing)[17]

[66]Response: (ACK „0“)

[66]Legend: (output) Output number – for example character „1“ (for output 1), „8“ (for 8), „15“, „32“, etc.

(state) Switch on („H“) or switch off („L“).

(time) Number 1 to 255. One unit equals 0.5 sec. Therefore, the period can be 0.5 to 127.5 seconds.

[66]*Example: Request – switch the output 5 for 10 sec*

**B1OT5H20↵*

Response

**B10↵*

Read settings of set outputs for a given time

Description: Instruction reads current settings applicable for “Set outputs for a certain period” instruction. This instruction allows you to find out which outputs are currently activated by period switching and how many impulses until they are switched back.

[97]Request: 33H(out)

[97]Response: (ACK 00H)(OUT)(time)

[97]Legend: (out) n bytes; numbers of outputs to be read – one byte for each number; in a 0 value is sent, all outputs settings are sent in response; Instruction can contain as many outputs as there are on your device regardless of their order.

(time) 1 byte; time period remaining of a given output pulse. Range 1 to 255, One unit equals 0.5 sec. Outputs with no period settings will always show 0 value.

(OUT) 1 byte; byte in format: SOOOOOOO, where “S” is the state to which the output is to be set (1 = on; 0 = off) and “O” is a number of the output (binary representation of number 1 to 127).

There is as many sequences (OUT)(time) as there were sent in the request or as many as there are outputs on your device if 0 was sent in the request.

[97]*Example: Read all outputs, address 35H, signature 02H*

2AH,61H,00H,06H,31H,02H,33H,00H,08H,0DH

Response – output 1 will remain switched on for another 13.5 sec, output 2 will remain switched off for another 13.5 sec, output 3 will remain off for another 4.5 sec.

2AH,61H,00H,0BH,31H,02H,00H,81H,1BH,02H,1BH,83H,09H,F1H,0DH

[66]Request: „ORT“(output) (Output Read Timing)

[66]Response: (ACK „0“)(state)(time)

[66]Legend: (output) Output number – for example character „1“ (for output 1), „8“ (for 8), „15“, „32“, etc.

(state) Switched on („H“) or switched off („L“).

(time) Number 1 to 255. One unit equals 0.5 sec. Outputs with no period settings have 0 as this parameter.

[66]*Example: Request output 3 state*

[17] Both variants can be used.

**B1ORT3↵*

Response – output will remain switched for another 4.5 sec

**B10H9↵*

Set pulse length on an output

Description: This instruction allows the user to set an interval and polarity of a pulse in advance for each output. Then this pulse can be initiated using “Initiate pulse on an output” (page 27) without setting the time.

[97]Request: 26H(output)(status)(time)

[97]Response: (ACK 00H)

[97]Legend: (output) 1 byte; output number (OUT1 = 01H)

(status) 1 byte; can have following values:

00H = no pulse settings

02H = positive pulse mode

03H = negative pulse mode

(time) 1 byte; time period of the pulse. Number 1 to 255. One unit equals 0.5 sec.

There can be up to 12 sequences (output)(status)(time) in a single request. This allows for multiple outputs setting.

[97]*Example: Set 2 second pulse for relay 4.*

2AH,61H,00H,08H,31H,02H,26H,04H,02H,04H,09H,0DH

Response

2AH,61H,00H,05H,31H,02H,00H,3CH,0DH

Read pulse length on an output

Description: Reads set pulse lengths and their polarity from the memory.

[97]Request: 36H(output)

[97]Response: (ACK 00H)(status)(time)

[97]Legend: (output) x byte; output number or numbers, 0 for all outputs at once

(status) 1 byte; can have following values:

00H = no pulse settings

02H = positive pulse mode

03H = negative pulse mode

(time) 1 byte; time period of the pulse. Number 1 to 255. One unit equals 0.5 sec.

[97]Note: There is as many sequences (status)(time) as there were sent in the request or as many as there are outputs on your device if 0 was sent in the request.

[97]*Example: Read all outputs pulse settings (example of Quido ETH 4/4):*

2AH,61H,00H,06H,31H,02H,36H,00H,05H,0DH

Response – OUT1: negative pulse 10 sec; OUT2: positive pulse 10 sec; OUT3: on/off mode; OUT4: kpositive pulse 2 sec.

2AH,61H,00H,0DH,31H,02H,00H,03H,14H,02H,14H,00H,00H,02H,04H,01H,0DH

Initiate pulse on an output

Description: Runs a pulse on an output according to settings done by instruction “Set pulse length on an output” on page 26. Recurring pulse initiation is possible, even if the previous pulse did not end on time. When the pulse is running, the remaining time can be read using the instruction “Read settings of set outputs for a given time” (page 25).

Pulse on the output will only be initiated if the function for temperature watching (thermostat mode) is inactive on it. If the function is active, the pulse can only be initiated when the temperature is between TEMPx and TEMPy. Otherwise the Quido will respond ACK 03H.

[97]Request: 25H(output)

[97]Response: (ACK 00H)

[97]Legend: (output) x byte; output number or numbers, a pulse will be initiated on these outputs

[97]*Example: Initiate pulse on outputs 2 and 4:*

`2AH,61H,00H,07H,31H,02H,25H,02H,04H,0FH,0DH`

Response:

`2AH,61H,00H,05H,31H,02H,00H,3CH,0DH`

Read output mode

Description: This instruction reads the set output mode.

[97]Request: 38H(output)

[97]Response: (ACK 00H)(status)

[97]Legend: (output) x byte; output number or numbers, 0 for all outputs at once

(status) 1 bit oriented byte (bit 7 = MSB); individual bits meaning is following:

7 A	**6** M1	**5** M0	**4**	**3** S3	**2** S2	**1** S1	**0** S0	**Bit** Bit name **Meaning**
1	0	1	0	0	*S*	*S*	*K*	Output is in automated thermostat mode. Bits SSK match SSK bits from z “Set thermostat” instruction on page 32.
0	0	0	0	0	0	1	0	Output is in manual mode and a positive pulse length is set.
0	0	0	0	0	0	1	1	Output is in manual mode and a negative pulse length is set.
0	0	0	0	0	0	0	0	Output is in manual mode and a NO pulse is set.

There are as many (status) bytes in the response as there are outputs in the request or as many as there are outputs on your device if 0 was sent in the request.

[97]*Example: Read all output modes (example of Quido ETH 4/4):*

`2AH,61H,00H,06H,31H,02H,38H,00H,03H,0DH`

Response – OUT1: static mode; OUT2: positive pulse; OUT3: negative pulse; OUT4: thermostat mode (bits SSK = 0):

`2AH,61H,00H,09H,31H,02H,00H,A0H,02H,03H,A0H,F3H,0DH`

Set automation

Description: This instruction allows for a few simple actions of outputs in relation to input states. For example, activating an input can activate an output etc.

(This function is only available with Quidos 2/2, 4/4 and 8/8.)

[97]Request: 40H (IN)(OUT)(function)(Hin)(Hout)

[97]Response: (ACK 00H)

[97]Legend: (IN) 1 byte; input number to which the function is related; one input can only have one function related to it.

(OUT) 1 byte; output number to which the function is related

(function) 1 byte; contains function number:

00H.... None: Automation is off.

01H.... Copy: Input state is copied to output state.

02H.... Run pulse: An edge on the input runs pulse on the output.[18]

03H.... Switch on or off by edge: Works as a pulse relay; first edge witches output on and the second edge switches it off.

(Hin) 1 byte; *Only for functions 2 and 3:*
React to leading (00H) or falling (01H) edge on the input.

(Hout) 1 byte; *For function 1:* Allows for output state negation (01H). Once the output is negated, switching the input on will switch it off and vice versa.
For function 2: If it is 00H a recurring edge does not affect the already running pulse. If it is 01H a recurring edge will prolong the output pulse.

[97]*Example:* *Request – each leading edge on input 2 will invert the state of output 2:*

`2AH,61H,00H,0AH,31H,02H,40H,02H,02H,03H,00H,00H,F0H,0DH`

Response:

`2AH,61H,00H,05H,31H,02H,00H,3CH,0DH`

Read automation settings

Description: Allows the user to read input-output automation that was set using the previous instruction.

[97]Request: 41H (IN)

[97]Response: (ACK 00H) (IN)(OUT)(function)(Hin)(Hout)

[97]Legend: *(Identical with previous instruction)*

[97]*Example:* *Request:*

`2AH,61H,00H,05H,31H,02H,41H,FBH,0DH`

Response – inverted state of input 3 copies on output 4:

`2AH,61H,00H,0AH,31H,02H,00H,03H,04H,01H,00H,01H,2EH,0DH`

[18] To set output pulse parameter use instruction *Set pulse length on an output* on page 26.

Measure and watch temperature

Important: If the module has no temperature sensor input, the response to the following instructions will be ACK 02H (invalid instruction).

Temperature measuring

Description: Returns temperature measured using the connected thermometer. Temperature is sent in previously set units.

[97]Request: 51H(thermometer)

[97]Response: (ACK 00H)(thermometer)(value)

[97]Legend: (thermometer) number of the thermometer to be read as a binary value (first thermometer as 01H); thermometers number that are not connected are now allowed; If it is 0, all thermometer values will be sent. Request can contain multiple subsequent numbers of thermometers (if those are connected). The response will then contain only those thermometers values.

(value) temperature in format signed INT (16 bit)

temperature = value / 10; The outcome has a resolution 1/10 of the set temperature unit.[19]

[97]*Example:* *Request – read thermometer 1:*

`2AH,61H,00H,06H,31H,02H,51H,01H,E9H,0DH`

Response – thermometer 1 - value 246, so the temperature is 24,6° :

`2AH,61H,00H,08H,31H,02H,00H,01H,00H,F6H,42H,0DH`

[97]Note: If the thermometer is out of range or the value cannot be read, the response is ACK 05H (device malfunction).

Instruction can contain multiple sequences (thermometer)(value) according to the number of connected thermometers.

---

[66]Request: „TR“(thermometer) (Temperature Read)

[66]Response: (ACK „0“)(value)

[66]Legend: (thermometer) number of the thermometer to be read as a binary value; thermometers number that are not connected are now allowed.

(value) temperature as an ASCII string (always 7 characters aligned to the right). Unused characters are filled with zeros (30H). Dot is used as a decimal point divider (2EH).

[66]*Example:* *Request: Read temperature from thermometer 1*

`*B1TR1↵`

Response: 29,1°

`*B10+029.1C↵`

[66]Note: If the thermometer is out of range or the value cannot be read, the response is ACK 5 (device malfunction).

[19] The real sensor accuracy is stated in the give Quido module datasheet.

Measure temperature - formatted

Description: Reads temperature from the connected thermometer and sends it as a (1) whole number multiplied by ten, (2) as a float decimal number and (3) as an ASCII string.

Quido ETH 3/0B does not support this instruction.

[97]Request: 58H(id)

[97]Response: (ACK 00H)[(id)(status)(int)(float)(string)]

[97]Legend: (id) number of the thermometer to be read as a binary value (first thermometer as 01H); thermometers number that are not connected are now allowed; If it is 0, all thermometer values will be sent. Request can contain multiple subsequent numbers of thermometers (if those are connected). The response will then contain only those thermometers values.

(status) 80H = temperature is valid; 00H temperature is invalid

(int) temperature in signed int format (16 bit): *temperature = int / 10*

(float) temperature in float format (IEEE 754)

(string) temperature as an ASCII string

[97]*Example: Request – read thermometers:*

`2AH,61H,00H,06H,B1H,02H,58H,00H,63H,0DH`

Response – thermometer 1 measured temperature 27,2°C :

`2AH,61H,00H,17H,B1H,02H,00H,01H,80H,01H,10H,41H,DAH,00H,00H,20H,20H,20H,20H,20H,20H,32H,37H,2EH,32H,74H,0DH`

[97]Note: If the thermometer is out of range or the value cannot be read, the response is ACK 05H (device malfunction).

Instruction can contain multiple sequences [(id)…] in the response based on the number of connected thermometers.

Set temperature units

Description: This instruction sets temperature unit that is later used for all read temperatures.

Quido ETH 3/0B does not support this instruction. Temperature unit is always °C.

[97]Request: 1CH(id)(unit)

[97]Response: (ACK 00H)

[97]Legend: (id) always 00H

(unit) 00H = Celsius; 01H = Fahrenheit; 02H = Kelvin

[97]*Example: Request – set Fahrenheit:*

`2AH,61H,00H,07H,B1H,02H,1CH,00H,01H,9DH,0DH`

Response:

`2AH,61H,00H,05H,B1H,02H,00H,BCH,0DH`

Read temperature units

Description: This instruction reads the currently set temperature units.

Quido ETH 3/0B does not support this instruction. Temperature unit is always °C.

[97]Request: 1DH

[97]Response: (ACK 00H)(id)(unit)

[97]Legend: (id) always 01H

(unit) 00H = Celsius; 01H = Fahrenheit; 02H = Kelvin

[97]*Example: Request:*

2AH,61H,00H,05H,B1H,02H,1DH,9FH,0DH

Response –Fahrenheit is set in the unit:

2AH,61H,00H,07H,B1H,02H,00H,01H,01H,B8H,0DH

Set temperature watching

Description: This instruction sets temperature limits. Once the temperature in outside these limits, the device sends an automated message.

[97]Request: 13H(idt)(on)(upper-int)(lower-int)(interval)(upper-str)(lower-str)

[97]Response: (ACK 00H)

[97]Legend: (idt) 1 Byte; thermometer id ranging from 1 to 8

Al of the following parameters have to be sent at once. It has to be defined as a (parameter-id)(parameter) for the device to recognize it reliably.

(on) id:01H; 1 Byte; witch on (01H) or switch off (00H) temperature watching

(upper-int) id:02H; 2 Bytes; upper limit as a whole number (signed int). It is temperature multiplied by ten. Example: Value 24.6 is entered as a number 246.

(lower-int) id:03H; 2 Bytes; lower limit as a whole number (signed int).

(interval) id:04H; 2 Bytes; when the temperature gets outside the limits and a notification is being sent repeatedly, enter an interval in seconds between individual notifications.

(upper-str) id:05H; 2 Bytes; upper limit as a string with one decimal number. Example: Value 24.6 enter as a number „24,6“.

(lower-str) id:06H; 2 Bytes; lower limit as a string with one decimal number. Example: Value -12.3 enter as a number „-12,3“.

[97]Example: Request – thermometer 1:

2AH,61H,00H,11H,31H,02H,13H,01H,01H,01H,02H,01H,36H,03H,00H,FAH,04H,00H,01H,DFH,0DH

Response:

2AH,61H,00H,05H,31H,02H,00H,3CH,0DH

Automated message – format is identical with instruction 58H[20] on page 30:

2AH,61H,00H,1CH,31H,01H,0FH,01H,30H,02H,01H,03H,82H,04H,01H,3CH,41H,FDH,80H,00H,20H,20H,20H,20H,20H,20H,33H,31H,2EH,37H,D6H,0DH

[20] *Quido ETH 3/0B always sends the float value empty.*

Read temperature watching settings

Description: This instruction reads set temperature limits.

[97]Request: 14H(idt)

[97]Response: (ACK 00H)(idt)(on)(upper-int)(lower-int)(interval)(upper-str)(lower-str)

[97]Legend: Parameters as the previous instruction (13H).

[97]*Example: Request:*

2AH,61H,00H,06H,31H,02H,14H,01H,26H,0DH

Response:

2AH,61H,00H,27H,31H,02H,00H,01H,01H,01H,02H,01H,36H,03H,00H,FAH,04H,00H,01H,05H,20H,20H,20H,20H,20H,20H,33H,31H,2EH,30H,06H,20H,20H,20H,20H,20H,20H,32H,35H,2EH,30H,CAH,0DH

Set thermostat

Description: Sets temperature limits at which the output state should be changed. [21]

[97]Request: 1AH (OUT)(FLAG)(TEMPx)(TEMPy)(TIME)(ERR)…

[97]Response: (ACK 00H)

[97]Legend: (OUT) 1 byte; output number to which a temperature watching (thermostat) function will be assigned (binary number; maximum of 255 outputs in theory; 0 value is not allowed); The request can contain up to 12 sequences: (OUT)(FLAG)(TEMPx)(TEMPy)(TIME)(ERR) at once.

(FLAG) 1 byte; byte in format: FSSKTTTT; bit meaning:

„F“ – temperature watching (thermostat) switched on / switched off for a given output (OUTx); (1 = on; 0 = off)

„SS“ – Action to perform once the temperature is reached

00 = switch the output on

01 = switch the output off

10 = switch the output on for a set interval (“positive pulse”)

11 = switch the output off for a set interval (“negative pulse”)

„K“ – Critical temperature tendency – only works with positive pulse:

0 – temperature rise

1 – temperature drop

„TTTT“ – Binary number of the thermometer providing the readout(TEMP.)

(TEMPx) 2 Bytes (Hbyte:Lbyte); value in format signed int[22]; higher temperature

(TEMPy) 2 Bytes (Hbyte:Lbyte); value in format signed int[22]; lower temperature

(TIME) 1byte; Time to switch relay on in seconds, if the pulse mode is active.

(ERR) 1byte; Determines what action should follow once the sensor cable is disconnected.

0 – leave the relay state as it is

1 – switch the relay off

[21] Description of the thermostat mode is in Appendix 1 on page 45 of this document.

[22] Value is a whole number. The real temperature is the value divided by ten: *temperature = (TEMP) / 10*

2 – switch the relay on

[97]*Example: Request – set thermostat on output 1 – switch on and off at 27.0°C*

2AH,61H,00H,0DH,31H,02H,1AH,01H,81H,01H,0EH,01H,0EH,05H,00H,75H,0DH

Response

2AH,61H,00H,05H,31H,02H,00H,3CH,0DH

Read thermostat settings

Description: Reads temperature limits on all outputs.[23]

[97]Request: 1BH (OUTs)

[97]Response: (ACK 00H)(OUT)(FLAG)(TEMPx)(TEMPy)(TIME)(ERR)…

[97]Legend: (OUTs) list of thermometer numbers to be shown (maximum of 12 thermometers at once, 0 value is not allowed).

(OUT) 1 byte; output number

(FLAG) 1 byte; byte in format: FSSKTTTT; bit meaning:

(FLAG) 1 byte; byte in format: FSSKTTTT; bit meaning:

„F“ – temperature watching (thermostat) switched on / switched off for a given output (OUTx); (1 = on; 0 = off)

„SS“ – Action to perform once the temperature is reached

00 = switch the output on

01 = switch the output off

10 = switch the output on for a set interval (“positive pulse”)

11 = switch the output off for a set interval (“negative pulse”)

„K“ – Critical temperature tendency – only works with positive pulse:

0 – temperature rise

1 – temperature drop

„TTTT“ – Binary number of the thermometer providing the readout(TEMP.)

(TEMPx) 2 Bytes (Hbyte:Lbyte); value in format signed int[24]; higher temperature

(TEMPy) 2 Bytes (Hbyte:Lbyte); value in format signed int[24]; higher temperature

(TIME) 1byte; Time to switch relay on in seconds, if the pulse mode is active.

(ERR) 1byte; Determines what action should follow once the sensor cable is disconnected.

0 – leave the relay state as it is

1 – switch the relay off

2 – switch the relay on

[97]*Example: Request*

2AH,61H,00H,05H,31H,02H,1BH,21H,0D

Response – Output 1 switches on and off at 27,0°C, output 2 has the thermostat mode off

2AH,61H,00H,15H,31H,02H,00H,01H,81H,01H,0EH,01H,0EH,05H,00H,02H,00H,27H,0FH,D8H,F1H,00H,00H,86H,0DH

[23] Description of the thermostat mode is in Appendix 1 on page 45 of this document.

[24] Value is a whole number. The real temperature is the value divided by ten: *temperature = (TEMP) / 10*

Response can contain multiple sequences ((OUT)(FLAG)(TEMPx)(TEMPy)(TIME)(ERR)). The number of sequences in response matches the number in request.

Communication line and address settings

Allow configuration

Description: This instruction allows the user to make configuration changes. It has to precede some instruction to set communication parameters. It is invalidated after a following instruction (even if said instruction is invalid) and configuration is automatically disabled. (Universal address cannot be used with this instruction.)

[97]Request: E4H

[97]Response: (ACK 00H)

[97]Example: Allow configuration

2AH,61H,00H,05H,01H,02H,E4H,88H,0DH

Response

2AH,61H,00H,05H,01H,02H,00H,6CH,0DH

[66]Request: „E“ *(Enable)*

[66]Response: (ACK „0“)

[66]Example: Request

**B1E↵*

Response

**B10↵*

Set communication parameters

Description: Sets address and communication speed (baudrate). (Universal address cannot be used with this instruction.[25]) [26]

[97]Request: E0H(address)(baudrate)

[97]Response: (ACK 00H)

[97]Legend: (address) 1 byte; Can range from 00H to FDH. When the 66 format is used, only addresses viewable as an ASCII character can be used (see Address section on page 12).

(baudrate) 1 byte; baudrate code according to tab. 1.[27]

[97]*Example: Set address 02H and communication speed (baudrate) 115200Bd; old address 01H*

2AH,61H,00H,07H,01H,02H,E0H,02H,0AH,7EH,0D

Response

2AH,61H,00H,05H,01H,02H,00H,6CH,0DH

Notes: New address and baudrate is set after sending a response. Allow configuration (page 34) instruction must precede this instruction. After setting the new communication parameters, configuration is automatically disabled.

Other parameters of the communication line are described in the datasheet of a given Quido module with the given interface.[27]

[66]Request: „AS“(Address)[28] (Address Set)

[66]Response: (ACK „0“)

[66]Legend: (Address) See Address section on page 12.

[66]*Example: Request: Address 4*

**B1AS4↵*

Response

**B10↵*

[66]Request: „SS“(code)[28] (Speed Set)

[66]Response: (ACK „0“)

[66]Legend: (code) Baudrate code according to tab. 1 (column 66)[27]

[66]*Example: Request: Communication speed 19200Bd (code 7)*

**B1SS7↵*

Response

**B10↵*

Baudrate Bd	Code 97	Code 66
110	00H	0
300	01H	1
600	02H	2
1200	03H	3
2400	04H	4
4800	05H	5
9600	06H	6
19200	07H	7
38400	08H	8
57600	09H	9
115200	0AH	A
230400	0BH	B

tab. 1 – comm. speed codes

[25] In case you the address is unknown and there is a single device on the line, address can be found using the instruction “Read communication parameters”. (Use universal address FEH.) If that is not possible (there are several devices on the communication line), new address can be set using the “Set address using serial number” (page 37).

[26] Changing these parameters will reset the device and their counters as well.

[27] **Ethernet and USB models have pre-set communication speed 115 200 Bd. Module responds with [97]ACK 03H (invalid data) or [66]ACK 4 (access denied) when a different baudrate code is sent in request.**

[28] Two different instructions are used to set comm. speed and address with 66 format. (Format 97 uses one instruction for both.)

Read communication parameters

Description: This instruction reads address and baudrate.

[97]Request: F0H

[97]Response: (ACK 00H)(address)(baudrate)

[97]Legend: (address) 1 byte; Devices address

(baudrate) 1 byte; Comm. speed codes are described in tab. 1.

[97]*Example: Read communication parameters; universal address FEH, signature 02H*

`2AH,61H,00H,05H,FEH,02H,F0H,7FH,0DH`

Response - Address 04H, comm. speed 9600Bd

`2AH,61H,00H,07H,04H,02H,00H,04H,06H,5DH,0DH`

[97]Poznámky: Use this instruction once the address is unknown. Request is sent to the universal address FEH. If the communication speed is unknown as well, all communication speeds must be tried. This only works in a single device is connected to the line.

[66]Request: „CP“ (Comm. parameter)

[66]Response: (ACK „0“)(address)(baudrate)

[66]Legend: (address) See Address section on page 12.

(rychlost) Comm. speed code according to tab. 1 (column 66)

[66]*Example: Request with universal address*

`*B$1CP↵`

Response – Address B, baudrate 9600Bd (code 6)

`*B10B6↵`

Set address using serial number

Description: Instruction allows the user to set an address based on a unique serial number of the device. This instruction is used when the controlling system or the user loses connected devices' addresses.

Serial number is written on the device on a label in format *[dev-number].[hardware-version].[software-version]/[serial-nubmer]* for example: *0227.00.03/0001*[29]

[97]Request: EBH(new-address)(device-number)(serial-number)

[97]Response: (ACK 00H)

[97]Legend: (new-address) 1 byte; new address of the module.

(dev-number) 2 Bytes; device number.

(serial-number) 2 Bytes; serial number is written on the device on a label after the device number. This number can also be found using instruction “Read factory data” (see page 38).

[97]*Example: Request – new address 32H, device number 199 (= 00C7H), serial number 101 (= 0065H)*

`2AH,61H,00H,0AH,FEH,02H,EBH,32H,00H,C7H,00H,65H,21H,0DH`

Response – device already responds with new address

`2AH,61H,00H,05H,32H,02H,00H,3BH,0DH`

[29] Information about finding out your device number is on page 5.

Supplemental

Read name and version

Description: Reads device name, internal software version and a list of possible communication formats. These settings are factory set.

Parameters (device-number) and (serial-number) do not need to be in the request. If they are sent in the request, they both have to be sent. If these parameters are sent and the instruction is received with broadcast address FFH, the device responds. This is the only case the device responds to broadcast address. This function allows for searching the device without knowing its address.

Example of the label on an actual device is on the right side. Number is in format (device-number)/(serial-number).

[97]Request: F3H*(device-number)(serial-number)*

[97]Response: (ACK 00H)(string)

[97]Legend: (device-number) 2 Bytes; device number

(serial-number) 2 Bytes; serial number

(string) Text in format: „*Quido [interface] [number-of-inputs]/[number-of-outputs]; **v**[device-number].[hardware-version].[software-version]; **f**66 97; **t**[no-of-thermometers]“ Actual example: „Quido ETH 4/4; v0254.02.07; f66 97; t1“.

[97]*Example: Request*

```
2AH,61H,00H,05H,FEH,02H,F3H,7CH,0DH
```

Example of Quido ETH 4/4 response:

```
2AH,61H,00H,2BH,31H,02H,00H,51H,75H,69H,64H,6FH,20H,45H,54H,48H,20H
,34H,2FH,34H,3BH,20H,76H,30H,32H,35H,34H,2EH,30H,32H,2EH,30H,37H,3B
H,20H,66H,36H,36H,20H,39H,37H,3BH,20H,74H,31H,DEH,0DH
```

[97]Note: Instruction can also contain other information in sections divided by semicolon, brake and a lowercase letter defining the type of information to follow.

(Example: *Quido USB 8/8; v0227.00.03; f66 97; t1; s358; dDG21*)

[66]Request: „?“

[66]Response: (ACK „0“)

[66]*Example: Request*

```
*B1?↵
```

Response –Quido ETH 4/4 response example:

```
*B10Quido ETH 4/4; v0254.02.07; f66 97; t1↵
```

[97]Note: Instruction can also contain other information in sections divided by semicolon, brake and a lowercase letter defining the type of information to follow.

(Example: *Quido ETH 4/4; v0254.02.07; f66 97; t1; s358; dDG21*)

Read factory data

Description: Instruction reads saved factory data from device.

[97]Request: FAH

[97]Response: (ACK 00H)(device-number)(serial-number)(factory-data)

[97]Legend: (device-number) 2 Bytes; device number.

(serial-number) 2 Bytes; serial number

(factory-data) 4 Bytes

[97]*Example: Request*

```
2AH,61H,00H,05H,FEH,02H,FAH,75H,0DH
```

Response – device number 199 (=00C7H), serial number 101 (=0065H)

```
2AH,61H,00H,0DH,35H,02H,00H,00H,C7H,00H,65H,20H,05H,09H,23H,B3H,0DH
```

Save user data

Description: Instruction saves user data. There is a section of the memory that user can use to save any data that will stay in the device even after power down or reset. This memory can be used for example to name the location.

[97]Request: E2H(position)(data)

[97]Response: (ACK 00H)

[97]Legend: (position) 1 byte; address of the memory where the data will be saved. Number ranging from 00H to 0FH.

(data) 1 to 16 bytes; any user data.

[97]*Example: Save words "Basement 1" to memory address 00H; Address 01H, signature 02H*

```
2AH,61H,00H,0FH,01H,02H,E2H,00H,"BASEMENT 1",61H,0DH
```

Response

```
2AH,61H,00H,05H,01H,02H,00H,6CH,0DH
```

Notes: Memory for user data is 16 bytes. In case data is written for example to 0CH, maximum of 4bytes can be saved.

---

[66]Request: „DW"(position)(data) *(Data Write)*

[66]Response: (ACK „0")

[66]Legend: (position) Address of the position in the memory where the data will be saved. It ranges from 0-9 or A-F.

(data) 1 to 16 bytes; Any user data ranging from 0 to 9 or from A to F.

[66]*Example: Request*

```
*B1DW0BASEMENT 1↵
```

Response

```
*B10↵
```

Read saved user data

Description: Instruction reads the saved user data. There is a section of the memory that user can use to save any data that will stay in the device even after power down or reset. This memory can be used for example to name the location.

[97]Request: F2H

[97]Response: (ACK 00H)(data)

[97]Legend: (data) 16 bytes; saved user data.

[97]*Example: Read user data; Address 01H, signature 02H*

2AH,61H,00H,05H,01H,02H,F2H,7AH,0DH

Response - "Basement 1 "

2AH,61H,00H,15H,01H,02H,00H,"BASEMENT 1 ",5DH,0DH

[66]Request: „DR“ *(Data Read)*

[66]Response: (ACK „0“)(data)

[66]Legend: (data) 1 to 16 bytes; User data.

[66]*Example: Request*

**B1DR↵*

Response

**B10BASEMENT 1↵*

Set input name

Description: This instruction allows the user to name each input with a string.

The free software for Quido uses this memory space. It is also used for Ethernet Quidos to save input and output names shown on the WEB interface. We do not recommend modifying this memory when using the WEB interface or standard software.

[97]Request: 2BH(input)(data)

[97]Response: (ACK 00H)

[97]Legend: (input) 1 byte; input number (IN1 = 01h)

(data) 21 bytes; Any user data.

[97]*Example: Save name "0Basement" for input 1 (unused bytes are filled with zeros)*

2AH,61H,00H,1BH,31H,02H,2BH,01H,30H,4BH,6FH,74H,65H,6CH,6EH,61H,00H ,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,FCH,0DH

Response

2AH,61H,00H,05H,31H,02H,00H,3CH,0DH

If there are no inputs on the device, it responds ACK 02H (invalid instruction).

Read input name

Description: This instruction reads set input name.

[97]Request: 3BH(input)

[97]Response: (ACK 00H)(data)

[97]Legend: (input) 1 byte; input number (IN1 = 01h)

(data) 21 bytes; any user data.

[97]*Example: Reads input 1 name*

```
2AH,61H,00H,06H,31H,02H,3BH,01H,FFH,0DH
```

Response – name is "0Basement" (unused bytes are filled with zeros)

```
2AH,61H,00H,1AH,31H,02H,00H,30H,4BH,6FH,74H,65H,6CH,6EH,61H,00H,00H
,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,29H,0DH
```

If there are no inputs on the device, it responds ACK 02H (invalid instruction).

Set output name

Description: This instruction allows the user to name each input with a string.

The free software for Quido uses this memory space. It is also used for Ethernet Quidos to save input and output names shown on the WEB interface. We do not recommend modifying this memory when using the WEB interface or standard software.

[97]Request: 2AH(output)(data)

[97]Response: (ACK 00H)

[97]Legend: (output) 1 byte; output number (IN1 = 01h)

(data) 21 bytes; any user data.

[97]*Example: Save name "0Siren" to output 4 (unused bytes are filled with zeros)*

```
2AH,61H,00H,1BH,31H,02H,2AH,04H,30H,53H,69H,72H,65H,6EH,61H,00H,00H
,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,66H,0DH
```

Response

```
2AH,61H,00H,05H,31H,02H,00H,3CH,0DH
```

If there are no outputs on the device, it responds ACK 02H (invalid instruction).

Read output name

Description: This instruction reads set output name.

[97]Request: 3AH(output)

[97]Response: (ACK 00H)(data)

[97]Legend: (output) 1 byte; output number (IN1 = 01h)

(data) 21 bytes; any user data.

[97]*Example: Read output 4 name*

2AH,61H,00H,06H,31H,02H,3AH,04H,FDH,0DH

Response – name is "0Siren" (unused bytes are filled with zeros)

2AH,61H,00H,1AH,31H,02H,00H,30H,53H,69H,72H,65H,6EH,61H,00H,00H,00H
,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,95H,0DH

If there are no outputs on the device, it responds ACK 02H (invalid instruction).

Set status

Description: Sets device status. This is user-defined byte used to get the device state. User can freely set this byte at any time. It is a free memory space allocated for device status. (This byte is reset after a power down or reset of the device.)

[97]Request: E1H (status)

[97]Response: (ACK 00H)

[97]Legend: (status) 1 byte; device status. After a power cycle (and even software reset) byte is set to 00H. If it was set using the instruction Set status before, it is then easy to identify the device state.

[97]*Example: Set status 12H; address 01H, signature 02H*

2AH,61H,00H,06H,01H,02H,E1H,12H,78H,0DH

Response

2AH,61H,00H,05H,01H,02H,00H,6CH,0DH

[66]Request: „SW“(status) (Status Write)

[66]Response: (ACK „0“)

[66]Legend: (status) character from “space to” “~” (32 – 126)

[66]*Example: Request – character A*

**B1SWA↵*

Response

**B10*

Read status

Description: Reads user defined status byte.

[97]Request: F1H

[97]Response: (ACK 00H)(status)

[97]Legend: (status) 1 byte; device status, meaning – see “Set status” instruction.

[97]*Example: Read status; address 01H, signature 02H*

2AH,61H,00H,05H,01H,02H,F1H,7BH,0DH

Response - status 12H

2AH,61H,00H,06H,01H,02H,00H,12H,59H,0DH

[66]Request: „SR“ (Status Read)

[66]Response: (ACK „0“)(character)

[66]Legend: (character) character from “space” to “~” (32 – 126)

[66]*Example: Request*

**B1SR↵*

Response

**B10A↵*

Read communication errors

Description: Instruction reads the number of communication errors that occurred since the device was powered up or since the last use of instruction Read communication errors.

[97]Request: F4H

[97]Response: (ACK 00H) (errors)

[97]Legend: (errors) 1 byte; Number of errors of communication since the device power up or last read. Communication errors can be following events:

Another byte comes instead of prefix

Checksum (SUMA) does not match

Incomplete message

[97]*Example: Read communication errors; address 01H, signature 02H*

2AH,61H,00H,05H,01H,02H,F4H,78H,0DH

Response - 5 errors

2AH,61H,00H,06H,01H,02H,00H,05H,66H,0DH

Allow checksum

Description: Allows the user to switch off checksum. This instruction is useful when debugging your application. Checksum (second to last byte) is not required when entering instructions by hand using a terminal. We do not recommend switching checksum off in regular communications. It is the only way of securing the data on the communication line. Checksum is switched on as a default.

[97]Request: EEH (state)

[97]Response: (ACK 00H)

[97]Legend: (state) 1 byte; 01H to switch checksum on; 00H to switch it off

[97]*Example: Request*

`2AH,61H,00H,06H,01H,02H,EEH,01H,7CH,0DH`

Response

`2AH,61H,00H,05H,01H,02H,00H,6CH,0DH`

Read checksum settings

Description: This instruction reads the current checksum setting. (See description of "Allow checksum" instruction.)

[97]Request: FEH

[97]Response: (ACK 00H) (state)

[97]Legend: (state) 1 byte; 01H to switch checksum on; 00H to switch it off

[97]*Example: Request setting state*

`2AH,61H,00H,05H,01H,02H,FEH,6EH,0DH`

Response – checksum on

`2AH,61H,00H,06H,01H,02H,00H,01H,6AH,0DH`

Set binary format timeout

Description: Sets timeout for Spinel 97 format (binary) communication.

Quido ETH 3/0B does not support this instruction.

[97]Request: E5H (time)

[97]Response: (ACK 00H)

[97]Legend: (time) 1 byte; time in tens of milliseconds – it sis possible to set 10ms to 2.55 sec.

[97]*Example: Request*

`2AH,61H,00H,06H,B1H,02H,E5H,20H,B6H,0DH`

Response

`2AH,61H,00H,05H,01H,02H,00H,6CH,0DH`

Read binary format timeout

Description: Reads binary format timeout.

Quido ETH 3/0B does not support this instruction.

[97]Request: F5H

[97]Response: (ACK 00H)(time)

[97]Legend: (time) 1 byte; time in tens of milliseconds

[97]*Example: Request*

2AH,61H,00H,05H,B1H,02H,F5H,C7H,0DH

Response

2AH,61H,00H,06H,B1H,02H,00H,20H,9BH,0DH

Reset

Description: This instruction resets the device. It is returned to the exact same state it was after a power on.

[97]Request: E3H

[97]Response: (ACK 00H)

[97]*Example: Reset; address 01H, signature 02H*

2AH,61H,00H,05H,01H,02H,E3H,89H,0DH

Response

2AH,61H,00H,05H,01H,02H,00H,6CH,0DH

Note: The device resets after a response is sent back.

[66]Request: „RE“ *(Reset)*

[66]Response: (ACK „0“)

[66]*Example: Request*

**B1RE↵*

Response

**B10↵*

Note: The device resets after a response is sent back.

Reset to default settings

Description: Sets all settings and parameters to the default. This instruction must be preceded by instruction “Allow configuration” on page 34.)

Quido ETH 3/0B does not support this instruction.

[97]Request: 8FH

[97]Response: (ACK 00H)

[97]*Example: Request*

2AH,61H,00H,05H,B1H,02H,8FH,2DH,0DH

Response

2AH,61H,00H,05H,B1H,02H,00H,BCH,0DH

Switch the communication protocol

Description: This instruction switches the communication protocol. (This instruction must be preceded by instruction “Allow configuration” on page 34.)

Modbus Configurator available on www.papouch.com can also be used to switch protocols.

Quido ETH 3/0B does not support this instruction.

[97]Request: EDH (id)

[97]Response: (ACK 00H)

[97]Legend: (id) 1 byte; protocol ID number:
01H – Spinel protocol, format 97 (binary) and 66 (ASCII)
02H – MODBUS RTU protocol (only Quido RS and Quido USB variants)
0AH – Spinel protocol, only format 97 (binary)

[97]*Example:* *Request*

```
2AH,61H,00H,06H,31H,02H,EDH,FFH,4FH,0DH
```

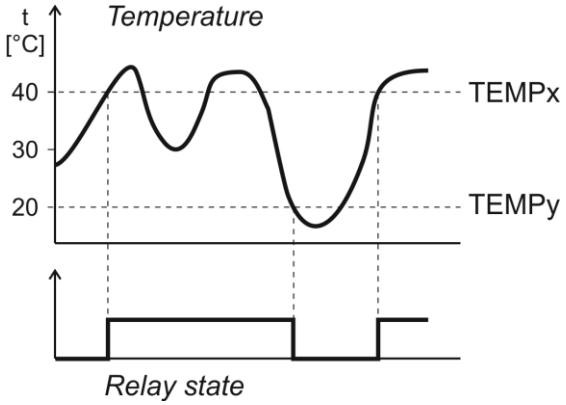
Response

```
2AH,61H,00H,05H,31H,02H,00H,3CH,0DH
```

APPENDIX 1: THERMOSTAT MODE

The following images show different options to setup temperature watching in Quido modules.[30] Temperatures and variables description is consistent with "Set temperature watching" instruction.

Mode 1



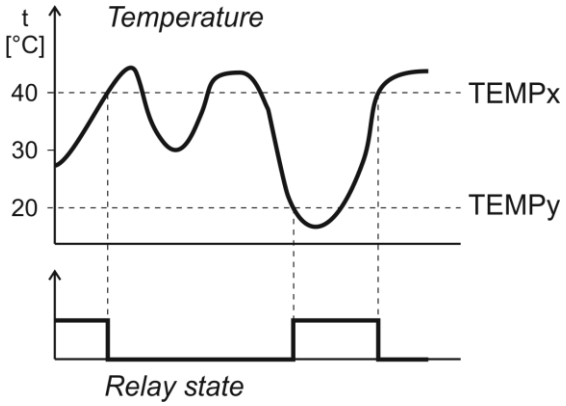
In this mode, the assigned relay is **switched ON** when set limits are exceeded. The relay switches on after TEMPx is exceeded and switches off when it gets back above TEMPy. This is called hysteresis. (Both temperatures can be set to the exact same value and disable the hysteresis.)

The relay is off when idle.

Settings for this mode – important bits in parameter FLAG:

Bity „SS": 00 – switch the output on

Mode 2



In this mode, the assigned relay is **switched OFF** when set limits are exceeded. The relay switches on after TEMPx is exceeded and switches off when it gets back above TEMPy. This is called hysteresis. (Both temperatures can be set to the same value to disable the hysteresis.)

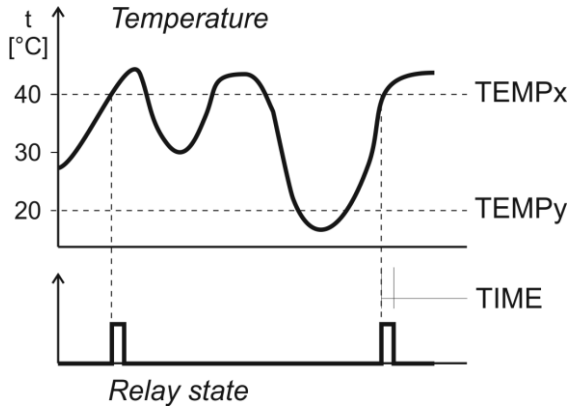
The relay is when idle.

Settings for this mode – important bits in parameter FLAG:

Bity „SS": 01 – switch the output off

[30] Thermostat mode is only available on modules with at least one thermometer and one output.

Mode 3



In this mode, the assigned relay **switches ON for a set period** (impulse mode) when the temperature rises above set limit. It can make another impulse only after the temperature drops below TEMPy and then rises above TEMPx again. (Both temperatures can be set to the same value and disable the oscillation protection.)

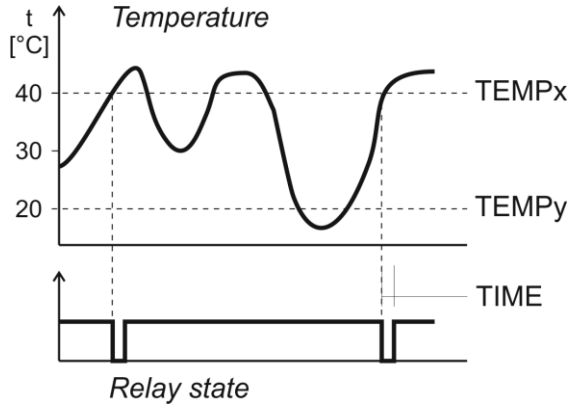
The relay is off when idle.

Settings for this mode – important bits in parameter FLAG:

Bity „SS“: 10 – switch output on for a set period

Bit „K“: 0 – temperature rise

Mode 4



In this mode, the assigned relay **switches OFF for a set period** (impulse mode) when the temperature rises above the set limit. It can make another impulse only after the temperature drops below TEMPy and then rises above TEMPx again. (Both temperatures can be set to the same value and disable the oscillation protection.)

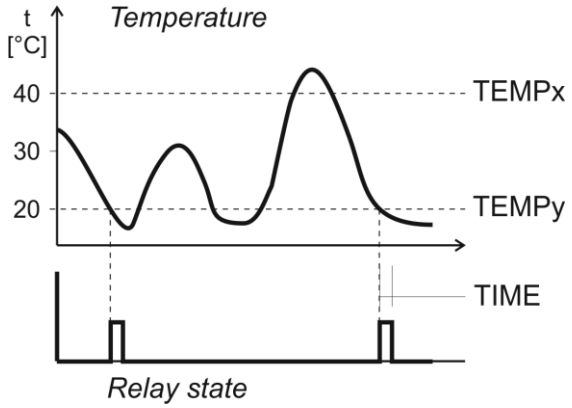
The relay is on when idle.

Settings for this mode – important bits in parameter FLAG:

Bity „SS“: 11 – switch output off for a set period

Bit „K“: 0 – temperature rise

Mode 5



In this mode, the relay **switches ON** for a set period (impulse mode) when the temperature drops below set limit. It can make another impulse only after the temperature rises above TEMPx and drops below TEMPy again. (Both temperatures can be set to the same value and disable the oscillation protection.)

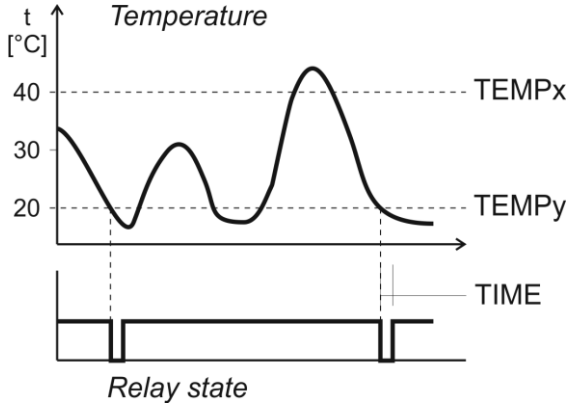
The relay is off when idle.

Settings for this mode – important bits in parameter FLAG:

Bity „SS“: 10 – switch output on for a set period

Bit „K“: 1 – temperature drop

Mode 6



In this mode, the relay **switches OFF** for a set period (impulse mode) when the temperature drops below set limit. It can make another impulse only after the temperature rises above TEMPx and drops below TEMPy again. (Both temperatures can be set to the same value and disable the oscillation protection.)

The relay is on when idle.

Settings for this mode – important bits in parameter FLAG:

Bity „SS“: 11 – switch output off for a set period

Bit „K“: 0 – temperature drop

Papouch s.r.o.

Data transmission in industry, line and protocol conversions, RS232/485/422/USB/Ethernet/GPRS/ WiFi, measurement modules, intelligent temperature sensors, I/O modules, and custom-made electronic applications.

Address:

Strasnicka 3164/1a
102 00 Praha 10

Telephone:

+420 267 314 267

Fax:

+420 267 314 269

Internet:

www.papouch.com

E-mail:

papouch@papouch.com