KORTEX QUATTRO IOT-2 Relay Programming Manual V1.3

1 Protocol: Kpsi Relais

Support TCP client, TCP server, UDP, CAN/RS485

1.1 Query status command/KeepAlive

Command code	00(2 character)	Return character (0: OFF, 1: ON) Format [relay1] [relay2]: [input1] [input2]: [channel count] example (2 channel): 00:11:2

Remarks 1 The command code is a text string and does not need to be followed by a return.

1.2 Control command

X	R	C	P
1(ON) 2(OFF)	1~32 relay X for all relay	null (ON/OFF) :(Delay) *(Jogging)	C parameter

1.2.1 ON/OFF Example

11 #relay 1 ON
21 #relay 1 OFF

12 #relay 2 ON
22 #relay 2 OFF

1X #relay all ON 2X #relay all OFF

1.2.2 Delay example

Delay ON/OFF some time and then OFF/ON
Delay second range 1-65535

11:30 #relay 1 ON, delay 30second OFF
21:30 #relay 1 OFF,delay 30second ON

12:30 #relay 2 ON, delay 30second OFF
22:30 #relay 2 OFF,delay 30second ON

1X:30 #relay all ON, delay 30second OFF
2X:30 #relay all OFF, delay 30second ON

1.2.3 Jogging example

Jogging ON/OFF little time(ms) and then OFF/ON Jogging time is 500ms (can change from web page),

11* #relay 1 ON, Jogging 500ms OFF

12* #relay 2 ON, Jogging 500ms OFF

1X* #all relay ON, Jogging 500ms OFF

2 Protocol: Kpsi binary

Support Different network segment communication (Multicast only support UDP)

Mulitcast addr: 224.0.2.11

Support password

2.1 default setting

IP	192.168.1.100
Netmask	255.255.255.0
Gateway	192.168.1.1
UDP Port	60000
Multicast addr	224.0.2.11

2.2 command

data bytes >=2byte store format is LSB

example:0x1234, store format is 0x34,0x12 format

Filed	Bytes	Comment
command	1	0xFF: set relay 0x07: multicast set relay
result (xor 0xAA)	1	pc->device: 0 xor 0xAA device->pc: result xor 0xAA result=0 success result=other fail
session	1	0~255 device reply the same
relay command	1	0: read relay status 1: write relay 2: write relay with delay 3: write relay with jogging 4: relay keep alive
password	2	0~9999 0:no password Password incurrent device no reply
command data	x	

2.2.1 read relay status

Pc send

Filed	Bytes	Comment
command	1	0xFF
result (xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	0: read relay status
password	2	0~9999 0:no password

Device reply

Filed	Bytes	Comment
command	1	0xFF
result (xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	0: read relay status
Relay status	1(2CH)	Bit0~31 map to relay relay1~2 Bit=1 relay on Bit=0 relay off

Example:
pc send:
FF AA 00 00 34 12 # password 0x1234 device reply:
FF AA 00 00 01 # relay 1 on

2.2.2 write relay

Filed	Bytes	Comment
command	1	0xFF
result (xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	1: write relay
password	2	0~9999 0:no password

relay mask	1(2CH)	Bit0~31 map to relay relay1~2 Bit=1, relay need update
relay set	1(2CH)	Bit0~31 map to relay relay1~2 Bit=1, relay on Bit=0, relay off

device reply

filed	bytes	comment
command	1	0xFF
result (xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	1: write relay

Example: pc send:

FF AA 00 01 34 12 05 01 # relay 1 on, rely 3 off device reply: FF AA 00 01

2.2.3 write relay with delay

filed	bytes	comment
command	1	0xFF
result (xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	2: write relay with delay
password	2	0~9999 0:no password
relay index and relay on/off	1	Bit0=1 relay on Bit0=0 relay off Bit1~bit7=relay index
Relay delay second	2	1~65535 second

device reply

filed	bytes	comment
command	1	0xFF
result(xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change

relay command	1	2: write relay with delay

Example: pc send:
FF AA 00 02 34 12 03 05 # relay 1 on, delay 5 second off device reply: FF AA 00 02

2.2.4 write relay with jogging

filed	bytes	comment
command	1	0xFF
result (xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	3: write relay with jogging
password	2	0~9999 0:no password
relay index and relay on/off	1	Bit0=1 relay on Bit0=0 relay off Bit1~bit7=relay index

device reply

filed	bytes	comment
command	1	0xFF
result (xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	3: write relay with jogging

Example: pc send:
FF AA 00 03 34 12 05 05 # relay 2 on, jogging device reply: FF AA 00 03

2.2.5 relay keep alive

Device send

filed	bytes	comment
command	1	0xFF
result(xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 pc not change
relay command	1	4: relay keep alive
device MAC	6	device MAC address
Relay status	1	Bit0~7 map to relay relay1~2 Bit=1 relay on Bit=0 relay off

Pc reply

filed	bytes	comment
command	1	0xFF
result (xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 pc not change
relay command	1	4: relay keep alive

Example:
device send:
FF AA 00 04 BC 34 88 12 34 56 00 # MAC BC:34:88:12:34:56 00:all relay off
pc reply: FF AA 00 00

3 Protocol: HTTP GET CGI

Relay board as HTTP server, accept HTTP GET CGI request.
Support CGI relay on/off
Support CGI relay jogging
Support CGI relay delay
Support CGI password verification

Support CGI Session check (details: 3.4 Session check)

3.1 load relay status

HTTP GET request

parameter	filed	data	comment

1	CGI API	relay_cgi_load.cgi	cgi changeable suffix relay_cgi_load.cgi, relay_cgi_load.php, relay_cgi_load.cs is work ok

HTTP GET respond

parameter	filed	data	comment
1	result	0	0: ok other fail
2	relay count	2	
3	relay 1 status	0/1	0: off 1: on
4	relay 2 status	0/1	0: off 1: on

example (4 channel relay):

HTTP GET request

http://192.168.1.100/relay_cgi_load.cgi # request relay board HTTP CGI API

HTTP GET respond

&0&4&1&0&1&0& # ok,4 relay,relay 1 on,relay 2 off,relay 3 on, relay 4 off

3.2 set relay

HTTP GET request

parameter	filed	data	comment
1	CGI API	relay_cgi.cgi	cgi suffix variable relay_cgi.cgi, relay_cgi.php, relay_cgi.cs is work ok
2	type	0/1/2	0: relay on/off 1: relay jogging 2: relay delay
3	relay	0~31	
4	on	0/1	0: off 1: on
5	time	0 1~255 1~65535	0: type 0: time 1: type 1~255: time(1=100ms) 2: type 1~65535: time(second)
6	pwd	0~9999	0~9999 Password incurrent device no respond

HTTP GET respond

parameter	filed	data	comment
1	result	0	0: ok other fail
2	type	0/1/2	0: relay on/off 1: relay jogging 2: relay delay
3	relay	0~31	0: relay 1 1: relay 2 ... 31: relay 32
4	on	0/1	0: off 1: on

5	time	0 1~255 1~65535	0: type 0: time
			1: type 1~255: time(1=100ms) 2: type 1~65535: time(second)

example 1(relay on):
HTTP GET request (request relay board HTTP CGI API, set relay 0 on, time 0, password 0)
http://192.168.1.100/relay_cgi.cgi?type=0&relay=0&on=1&time=0&pwd=0&
HTTP GET respond
&0&0&0&1&0& # ok, type 0 on/off,relay 0 on,time 0

example 2(relay off):
HTTP GET request (request relay board HTTP CGI API, set relay 0 off, time 0, password 0)
http://192.168.1.100/relay_cgi.cgi?type=0&relay=0&on=0&time=0&pwd=0&
HTTP GET respond
&0&0&0&0&0& # ok, type 0 on/off,relay 0 off,time 0

example 3(relay 1 jogging on):
HTTP GET request (request relay board HTTP CGI API, set relay 1 jogging on, time 500ms, password 4660)
http://192.168.1.100/relay_cgi.cgi?type=1&relay=1&on=1&time=5&pwd=4660&
HTTP GET respond
&0&1&1&1&5& # ok, type 1 jogging,relay 1 on,time 5(500ms)

example 4(relay 1 jogging off):
HTTP GET request (request relay board HTTP CGI API, set relay 1 jogging off,time 500ms,password 4660)
http://192.168.1.100/relay_cgi.cgi?type=1&relay=1&on=0&time=5&pwd=4660&
HTTP GET respond
&0&1&1&0&5& # ok, type 1 jogging,relay 1 off,time 5(500ms)

example 5(relay 1 on delay 10 second off):
HTTP GET request (request relay board HTTP CGI API, set relay 1 on delay 10 second off, time 5 second,password 4660)
http://192.168.1.100/relay_cgi.cgi?type=2&relay=1&on=1&time=10&pwd=4660&
HTTP GET respond
&0&2&1&1&10& # ok, type 2 delay,relay 1 on,time 10 second

example 6(relay 1 off delay 10 second on):

HTTP GET request (request relay board HTTP CGI API, set relay 1 off delay 10 second on, time 5 second,password 4660)

http://192.168.1.100/relay_cgi.cgi?type=2&relay=1&on=0&time=10&pwd=4660&

HTTP GET respond

&0&2&1&0&10& # ok, type 2 delay,relay 1 off,time 10 second

3.3 load input status

HTTP GET request

parameter	filed	data	comment
1	CGI API	input.cgi	cgi changeable suffix input.cgi, input.php, input.cs is work ok

HTTP GET respond

parameter	filed	data	comment
1	result	0	0: ok other fail
2	Input start postion	0	default: 0
3	input count	2/4/8/16/24/32	
4	relay 1 status	0/1	0: low 1: high
5	relay 2 status	0/1	0: low 1: high

example (4 channel relay):

HTTP GET request

http://192.168.1.100/input.cgi # request relay board HTTP CGI API

HTTP GET respond

&0&0&4&1&0&1&0& # ok, reserve,4 input, input 1 high, relay 2 low, relay 3 high, relay 4 low

3.4 Session check

The HTTP CGI session check is implemented by adding a “Cookie” header

“Cookie” header example:

Cookie: session=12345678

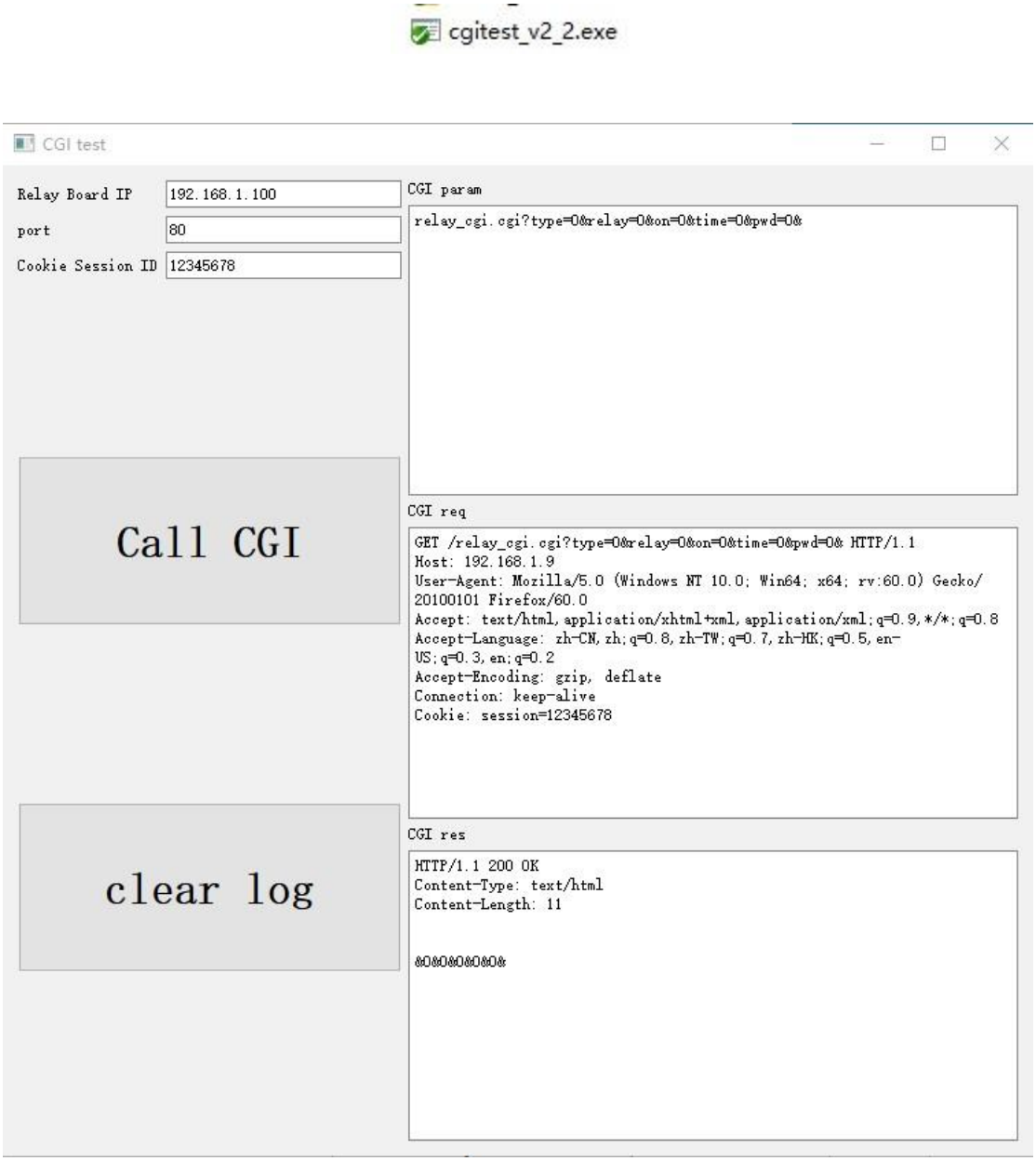
3.4.1 web config to Enable HTTP Session check

Paramètres réseaux

Version matérielle	V3.1F
Version logicielle	V3.2.2A
Date de construction	03-02-2023
Modèle	KPSI R002 LOG GDB
Numéro de série	6276
Date-Heure	22/02/2023 17:20:57 Sync Time
NTP Serveur	time.google.com
Nom d'hôte	KPSI-IOT2
Nom d'hôte+Suffixe	KPSI-IOT2 + NULL
HTTP ou HTTPS	HTTP
HTTP Port	80
HTTPS Port	443
HTTP Session	Non
HTTP ID (Session magique)	0
ETH DHCP	Non
ETH IP	192.168.1.100
ETH Netmask	255.255.255.0
ETH Gateway	192.168.1.33
ETH DNS	192.168.1.33 192.168.1.33
ETH MAC	be:34:88:00:18:84
-	
STA Activé	Non
STA Auth	WPA2 PSK
STA DHCP	Oui
STA IP	0.0.0.0
STA Netmask	0.0.0.0
STA Gateway	0.0.0.0
STA DNS	192.168.1.33 192.168.8.1
STA MAC	bc:34:88:00:18:84
STA WiFi SSID	
STA WiFi mot de passe	

Sauver

3.4.2 HTTP CGI test tool



3.4.3 full example(success):

HTTP GET request:

GET /relay_cgi.cgi?type=0&relay=0&on=1&time=0&pwd=0& HTTP/1.1

Host: 192.168.1.9

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:60.0) Gecko/20100101 Firefox/60.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2

Accept-Encoding: gzip, deflate

Connection: keep-alive

Cookie: session=12345678

HTTP GET response:

HTTP/1.1 200 OK

Content-Type: text/html

Content-Length: 11

&0&0&0&1&0&

3.4.4 full example(fail)

HTTP GET request:

GET /relay_cgi.cgi?type=0&relay=0&on=0&time=0&pwd=0& HTTP/1.1

Host: 192.168.1.9

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:60.0) Gecko/20100101 Firefox/60.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2

Accept-Encoding: gzip, deflate

Connection: keep-alive

Cookie: session=1234567

HTTP GET response:

HTTP/1.1 200 OK

Content-Type: text/html

Content-Length: 7

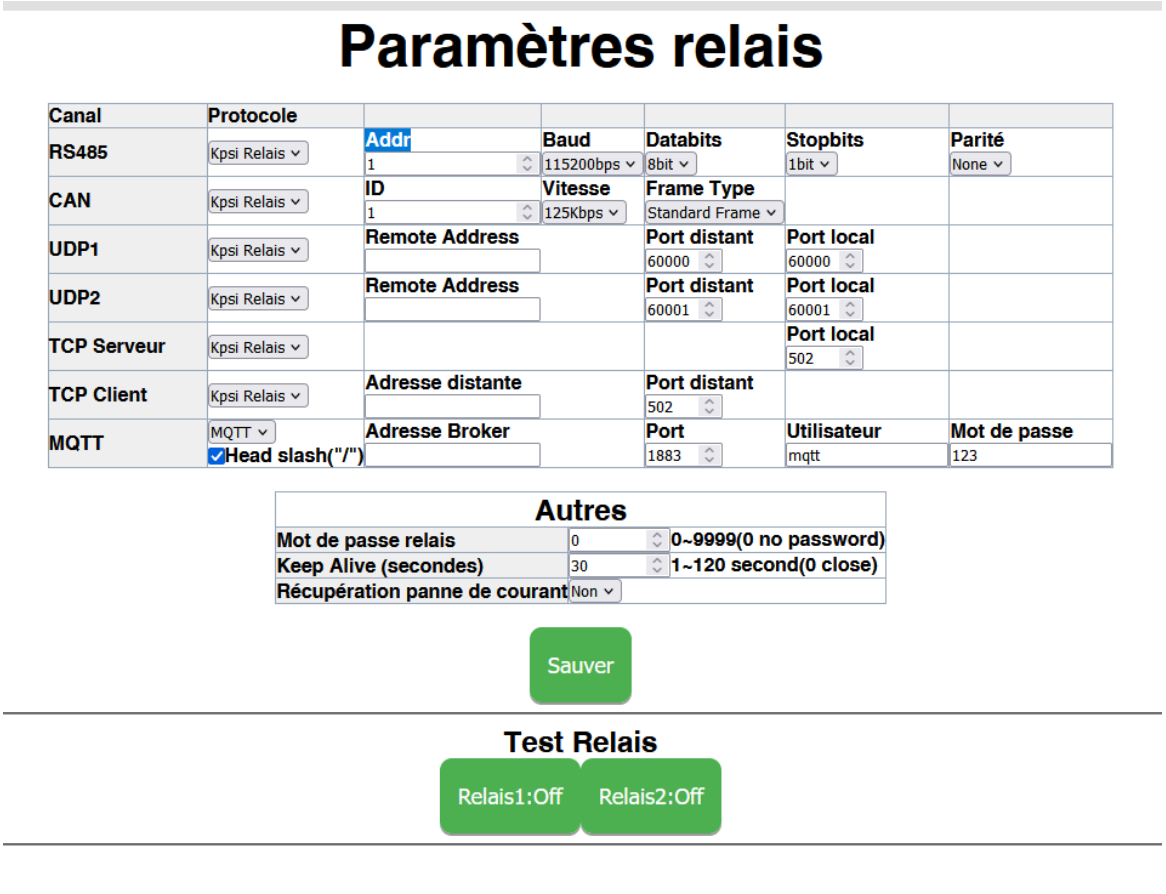
&302&/&

4 Protocol:Modbus-RTU/TCP/ASCII

Support Modbus:
Modbus-RTU
Modbus-TCP/UDP
Modbus-ASCII
Modbus-RTU Over TCP/UDP
Modbus-ASCII Over TCP/UDP

Support Modbus Function:
0x03read holding register
0x06Write Single register
0x10Wirte Multile register (CAN bus not support)

Notice: Modbus-RTU Over UDP/TCP,Modbus-ASCII Over UDP/TCP use RS485 addr



4.1 Registers

Register	Name	0x03/0x06/0x10	Value
0x0000	Relay Count	0x03	2
0x0001	Relay Status	0x03	bit0~7 map to relay1~2
0x0002	Write Relay	0x06	bit0~7 new status of relay1~2 (bit=1 ON,bit=0 OFF) bit8~15 map to relay1~2 need update(bit=1 Update)
0x0003	Advance Write Type	0x10	Bit0~5: 1: Write ON/OFF 2: Write with delay 3: Write with Jogging bit6~15:(only for Type: Write ON/OFF(1)) relay group:0~3 r1~8:G0
0x0004	Advance Write Password	0x10	Password 0~65535 when password in current do nothing
0x0005	Advance Write Relay	0x10	Type:Write ON/OFF(1) bit0~7 new status of relay1~2(bit=1 ON,bit=0 OFF) bit8~15 map to relay1~2 need update(bit=1 Update) Type:Write with delay(2) bit0: bit=1 ON,bit=0 OFF bit1~7:relay index 0~31 Type:Write with Jogging(3) bit0: bit=1 ON,bit=0 OFF bit1~7:relay index 0~31
0x0006	Advance Write Time	0x10	Type:Write ON/OFF(1) 0 Type:Write with delay(2) Number of Second need delay Type:Write with Jogging(3) Number of 100ms need jogging(1=100ms)
0x0007	Expand Write Status Group	0x10	relay1~2: G0
0x0008	Expand Write Relay Mask	0x10	bit0~15 map to relay G0:R1~2 need update (bit=1 Update)
0x0009	Expand Write Relay	0x10	bit0~15 map to relay G0:R1~2
0x000A	Expand Input Status 1~2	0x03	input1~2
0x000E	Expand Relay Status 1~2	0x03	relay1~2

0x0012	Expand Write Relay Mask 1~2	0x10	relay1~2mask bits; bit=1 need change,bit=0 no change
0x0014	Expand Write Relay bits 1~2	0x10	relay1~2 relay bits; bit=1(ON), bit=0(OFF)

Notice:

1、0x0003~6/0x0007~9/0x0012~15 is block，must written at the same time.

4.2 Modbus-RTU + Modbus-RTU Over TCP/UDP

4.2.1 0x03: Read holding register

Read all Relay Status
Send:
01 03 0000 0002 C40B Recv:
01 03 04 0004 0000 BBF2

Read all Input Status (2/4/8 channel)
Send:
01 03 000A 0001 A408 Recv:
01 03 02 FFFF B9F4

Read all Relay Status (2/4/8 channel)
Send:
01 03 000E 0001 E5C9 Recv:
01 03 02 0000 B844

4.2.2 0x06: Write Single Register

4 Relay All ON Send:
01 06 0002 0f0f 6DFE
Recv:
01 06 0002 0f0f 6DFE

4 Relay All OFF Send:
01 06 0002 0f00
2DFA Recv:
01 06 0002 0f00 2DFA

Relay 1,4 ON; Relay 2,3 stay the
same Send: 01 06 0002 0909 EE5C
Recv:
01 06 0002 0909 EE5C

4.2.3 0x10: Write Multiple Register

1、ON/OFF

4 Relay All ON Send:
01 10 0003 0004 08 0001 0000 0f0f 0000 91A9 Recv:
01 10 0003 0004 31 CA

4 Relay All OFF Send:
01 10 0003 0004 08 0001 0000 0f00 0000 A1AA Recv:
01 10 0003 0004 31 CA

Relay 2,3 ON; Relay 1,4 stay the same Send:
01 10 0003 0004 08 0001 0000 0606 0000 4237
Recv: 01 10 0003 0004 31 CA

2、Delay

Relay 1 OFF Delay 5 Second ON Send:
01 10 0003 0004 08 0002 0000 0000 0005 51BD Recv:
01 10 0003 0004 31 CA

Relay 1 ON Delay 5 Second OFF Send:
01 10 0003 0004 08 0002 0000 0001 0005 007D Recv:
01 10 0003 0004 31 CA

Relay 2 ON Delay 5 Second OFF Send:
01 10 0003 0004 08 0002 0000 0003 0005 A1BD Recv:
01 10 0003 0004 31 CA

3、Jogging

Relay 4 ON Joging 500ms OFF,Password 0x1234 Send:
01 10 0003 0004 08 0003 1234 0007 0005 420A Recv:
01 10 0003 0004 31 CA

Relay 1 OFF Joging 500ms ON Send:
01 10 0003 0004 08 0003 0000 0000 0005 417D Recv:
01 10 0003 0004 31 CA

Relay 1 ON Joging 500ms OFF Send:
01 10 0003 0004 08 0003 0000 0001 0005 10BD Recv:
01 10 0003 0004 31 CA

Relay 2 ON Joging 500ms OFF Send:
01 10 0003 0004 08 0003 0000 0003 0005 B17D Recv:

01 10 0003 0004 31 CA

4.3 Modbus-TCP/UDP

4.3.1 0x03: Read holding register

Read all Relay Status
Send:
0000 0000 0006 FF 03 0000 0002 Recv:
0000 0000 0007 FF 03 04 0004 000F

Read all Relay Status (Extend support 16/24/32 channel relay board) Send:
0000 0000 0006 FF 03 0000 0002 Recv:
0000 0000 0007 FF 03 04 0004 000F

Read all Input Status (2/4/8 channel)
Send:
0000 0000 0006 FF 03 000A 0001 Recv:
0000 0000 0005 FF 03 02 FFFF

Read all Relay Status (2/4/8 channel)
Send:
0000 0000 0006 FF 03 000E 0001 Recv:
0000 0000 0005 FF 03 02 0000

4.3.2 0x06: Write Single Register

4 Relay All ON Send:
0000 0000 0006 FF 06 0002 0f0f Recv:
0000 0000 0006 FF 06 0002 0f0f

4 Relay All OFF Send:
0000 0000 0006 FF 06 0002 0f00 Recv:
01 06 0002 0f00 2DFA

Relay 1,4 ON; Relay 2,3 stay the same Send:
0000 0000 0006 FF 06 0002 0909 Recv:
0000 0000 0006 FF 06 0002 0909

4.3.3 0x10: Write Multiple Register

1 ON/OFF

4 Relay All ON Send:
0001 0000 000F FF 10 0003 0004 08 0001 0000 0f0f 0000 Recv:
0001 0000 0006 FF 10 0003 0004

4 Relay All OFF Send:
0001 0000 000F FF 10 0003 0004 08 0001 0000 0f00 0000 Recv:
0001 0000 0006 FF 10 0003 0004

Relay 2,3 ON; Relay 1,4 stay the same Send:
0001 0000 000F FF 10 0003 0004 08 0001 0000 0606 0000 Recv:
0001 0000 0006 FF 10 0003 0004

2 Delay

Relay 1 OFF Delay 5 Second ON Send:
0001 0000 000F FF 10 0003 0004 08 0002 0000 0000 0005 Recv:
0001 0000 0006 FF 10 0003 0004

Relay 1 ON Delay 5 Second OFF Send:
0001 0000 000F FF 10 0003 0004 08 0002 0000 0001 0005 Recv:
0001 0000 0006 FF 10 0003 0004

Relay 2 ON Delay 5 Second OFF Send:
0001 0000 000F FF 10 0003 0004 08 0002 0000 0003 0005 Recv:
0001 0000 0006 FF 10 0003 0004

3 Jogging

Relay 4 ON Joging 500ms OFF,Password 0x1234 Send:
0001 0000 000F FF 10 0003 0004 08 0003 1234 0007 0005 Recv:
0001 0000 0006 FF 10 0003 0004

Relay 1 OFF Joging 500ms ON Send:
0001 0000 000F FF 10 0003 0004 08 0003 0000 0000 0005 Recv:
0001 0000 0006 FF 10 0003 0004

Relay 1 ON Joging 500ms OFF Send:
0001 0000 000F FF 10 0003 0004 08 0003 0000 0001 0005 Recv:
0001 0000 0006 FF 10 0003 0004

Relay 2 ON Joging 500ms OFF Send:
0001 0000 000F FF 10 0003 0004 08 0003 0000 0003 0005 Recv:
0001 0000 0006 FF 10 0003 0004

4.4 Modbus-ASCII + Modbus-ASCII Over TCP/UDP

4.4.1 0x03: Read holding register

Read all Relay Status
Send:
ASCII : 01 03 0000 0002 BA \r\n
HEX 3A 3031 3033 30303030 30303032 4241 0D0A
Recv:
ASCII : 01 03 04 0004 0000 54 \r\n
HEX 3A 3031 3033 3034 30303034 30303030 3534 0D0A

Read all Input Status (2/4/8 channel)
Send:
ASCII : 01 03 000A 0001 AA \r\n
HEX 3A 3031 3033 30303041 30303031 4141 0D0A
Recv:
ASCII : 01 03 02 FFFF C2 \r\n
HEX 3A 3031 3033 3032 46464646 4332 0D0A

Read all Input Status (support 16/24/32 channel)
Send:
ASCII : 01 03 000A 0002 A9 \r\n
HEX 3A 3031 3033 30303041 30303032 4139 0D0A
Recv:
ASCII : 01 03 04 FFFF 00FF D4 \r\n
HEX 3A 3031 3033 3034 46464646 30304646 4434 0D0A

Read all Relay Status (2/4/8 channel)
Send:
ASCII : 01 03 000E 0001 A6 \r\n
HEX 3A 3031 3033 30303045 30303031 4136 0D0A
Recv:
ASCII : 01 03 02 0000 1A \r\n
HEX 3A 3031 3033 3032 30303030 3141 0D0A

Read all Relay Status (support 16/24/32 channel)
Send:
ASCII : 01 03 000E 0002 A5 \r\n
HEX 3A 3031 3033 30303045 30303032 4135 0D0A
Recv:
ASCII : 01 03 04 0000 0080 FB93 \r\n
HEX 3A 3031 3033 3034 30303030 30303830 3530 0D0A

4.4.2 0x06: Write Single Register

4 Relay All ON Send:

ASCII : 01 06 0002 0F0F 8B \r\n HEX 3A 3031 3036 30303032 30463046 3842 0D0A Recv:

ASCII : 01 06 0002 0F0F 8B \r\n

HEX 3A 3031 3036 30303032 30463046 3842 0D0A

4 Relay All OFF Send:

ASCII : 01 06 0002 0F00 A1 \r\n HEX 3A 3031 3036 30303032 30463030 4131 0D0A Recv:

ASCII : 01 06 0002 0F00 A1 \r\n

HEX 3A 3031 3036 30303032 30463030 4131 0D0A

4.4.3 0x10: Write Multiple Register

1 ON/OFF

4 Relay All ON Send:

ASCII :01 10 0003 0004 08 0001 0000 0F0F 0000 22 \r\n

HEX 3A 3031 3130 30303033 30303034 3038 30303031 30303030 30463046 30303030 3232 0D0A

Recv:

ASCII :01 10 0003 0004 B7 \r\n

HEX 3A 3031 3130 30303033 30303034 4237 0D0A

4 Relay All OFF Send:

ASCII :01 10 0003 0004 08 0001 0000 0F00 0000 38 \r\n

HEX 3A 3031 3130 30303033 30303034 3038 30303031 30303030 30463030 30303030 3338 0D0A

Recv:

ASCII :01 10 0003 0004 B7 \r\n

HEX 3A 3031 3130 30303033 30303034 4237 0D0A

Relay 2,3 ON; Relay 1,4 stay the same Send:

ASCII :01 10 0003 0004 08 0001 0000 0606 0000 42 \r\n

HEX 3A 3031 3130 30303033 30303034 3038 30303031 30303030 30363036 30303030 3432 0D0A

Recv:

ASCII :01 10 0003 0004 B7 \r\n

HEX 3A 3031 3130 30303033 30303034 4237 0D0A

2 Delay

Relay 1 ON Delay 5 Second OFF Send:

ASCII :01 10 0003 0004 08 0002 0000 0001 0005 47 \r\n

HEX 3A 3031 3130 30303033 30303034 3038 30303032 30303030 30303031 30303035 3437 0D0A

Recv:

ASCII :01 10 0003 0004 B7 \r\n

HEX 3A 3031 3130 30303033 30303034 4237 0D0A

Relay 4 ON Delay 5 Second OFF Send:

ASCII :01 10 0003 0004 08 0002 0000 0007 0005 41 \r\n

HEX 3A 3031 3130 30303033 30303034 3038 30303032 30303030 30303037 30303035 3431 0D0A

Recv:

ASCII :01 10 0003 0004 B7 \r\n

HEX 3A 3031 3130 30303033 30303034 4237 0D0A **3**

Jogging

Relay 4 ON Joging 500ms OFF,Password 0x1234 Send:

ASCII :01 10 0003 0004 08 0003 1234 0007 0005 36 \r\n

HEX 3A 3031 3130 30303033 30303034 3038 30303033 31323334 30303037 30303035 3336 0D0A

Recv:

ASCII :01 10 0003 0004 B7 \r\n

HEX 3A 3031 3130 30303033 30303034 4237 0D0A

Relay 1 ON Joging 500ms OFF Send:

ASCII :01 10 0003 0004 08 0003 0000 0001 0005 46 \r\n

HEX 3A 3031 3130 30303033 30303034 3038 30303033 30303030 30303031 30303035 3436 0D0A

Recv:

ASCII :01 10 0003 0004 B7 \r\n

HEX 3A 3031 3130 30303033 30303034 4237 0D0A

Relay 4 ON Joging 500ms OFF Send:

ASCII :01 10 0003 0004 08 0003 0000 0007 0005 40 \r\n

HEX 3A 3031 3130 30303033 30303034 3038 30303033 30303030 30303037 30303035 3430 0D0A

Recv:

ASCII :01 10 0003 0004 B7 \r\n

HEX 3A 3031 3130 30303033 30303034 4237 0D0A

5 Protocol: MQTT

MQTT version 3.1.1

Relay board as MQTT client, communication with broker.

Support relay on/off

Support relay jogging

Support relay delay

Support password verification

Support LWT (details: 5.5 MQTT LWT topic)

Paramètres réseaux

Version matériel	**V3.6F**
Version logiciel	**V3.2.1A**
Date de construction	**30-12-2022**
Modèle	**KPSI R002 LOG GDB**
Numéro de série	**6276**
Date-Heure	**03/01/2023 11:16:23** Sync Time
NTP Serveur	pool.ntp.org
Nom d'hôte	**KPSI-IOT2**
Nom d'hôte+Suffixe	KPSI-IOT2 + NULL
HTTP ou HTTPS	HTTP
HTTP Port	80
HTTPS Port	443
HTTP Session	Non
HTTP ID (Session magique)	0
ETH DHCP	Non
ETH IP	192.168.1.100
ETH Netmask	255.255.255.0
ETH Gateway	192.168.1.1
ETH DNS	**192.168.1.1** 192.168.1.1
ETH MAC	ba:34:88:00:18:84
-	
STA Activé	Non
STA Auth	WPA2 PSK
STA DHCP	Oui
STA IP	0.0.0.0
STA Netmask	0.0.0.0
STA Gateway	0.0.0.0
STA DNS	**192.168.1.1** 192.168.8.1
STA MAC	bc:34:88:00:18:84
STA WiFi SSID	Airbox-1394
STA WiFi mot de passe	MbkKm7dv3RG9

Relay board Ethernet MQTT Client Id kortex-relay+SN Relay board WiFi MQTT Client Id kortex-wrelay+SN

Example:

below relay board “Serial Number” is 1868 so

ETH MQTT client id is:kortex-relay1868 so

WiFI MQTT client id is:kortex-wrelay1868

5.1 MQTT Topic Fast View

firmware version <V2.15.869

/kortex/relay/in/control /kortex/relay/out/relayX

firmware version >=V2.15.869

/kortex/relaySN/in/control /kortex/relaySN/out/relayX

firmware version >= V2.17.xx

ETH

/kortex/relaySN/in/control

/kortex/relaySN/in/rX

/kortex/relaySN/out/rX

/kortex/relaySN/out/iX

/kortex/relaySN/out/relayX

/kortex/relaySN/out/inputX

/kortex/relaySN/out/ip

/kortex/relaySN/out/sn

/kortex/relaySN/out/mac

/kortex/relaySN/out/input_cnt

/kortex/relaySN/out/relay_cnt

WiFi

/kortex/wrelaySN/in/control

/kortex/wrelaySN/in/rX

/kortex/wrelaySN/out/rX

/kortex/wrelaySN/out/iX

/kortex/wrelaySN/out/relayX

/kortex/wrelaySN/out/inputX

/kortex/wrelaySN/out/ip

/kortex/wrelaySN/out/sn

/kortex/wrelaySN/out/mac

/kortex/wrelaySN/out/input_cnt

/kortex/wrelaySN/out/relay_cnt

5.2 MQTT Topic (firmware version < V2.15.869)

topic	type	value		
/kortex/relay/in/control	subscribe	parameter	filed	data
		type	command type	ON/OFF DELAY JOGGING
		idx	relay index	1~2
		status	relay status	ON, OFF
		time	time for type	ON/OFF:0 DELAY:1~65535second JOGGING:1~255*100ms
		pass	password	0~9999
		example: {"type":"ON/OFF",'idx':'1',"status":"ON","time":"0","pass":"0"} {"type":"DELAY",'idx':'2',"status":"ON","time":"5","pass":"0"} {"type":"JOGGING",'idx':'1',"status":"ON","time":"5","pass":"0"} {"type":"ON/OFF",'idx':'2',"status":"OFF","time":"0","pass":"0"}		
/kortex/relay/out/relayX example: /kortex/relay/out/relay1 /kortex/relay/out/relay2	publish	parameter	filed	data
		idx	relay index	1~2
		status	relay status	ON, OFF
		example: {“idx”:”1”,”status”:”OFF”}		

5.3 MQTT Topic (firmware version >= V2.15.869)

topic	type	value		
/kortex/relaySN/in/control example: /kortex/relay1868/in/control	subscribe	parameter	filed	data
		type	command type	ON/OFF DELAY JOGGING
		idx	relay index	1~2
		status	relay status	ON,OFF
		time	time for type	ON/OFF:0 DELAY:1~65535second JOGGING:1~255*100ms
		pass	password	0~9999

<table>
<tr><td></td><td></td><td colspan="3">example:
{"type":"ON/OFF",'idx':'1',"status":"ON","time":"0","pass":"0"}
{"type":"DELAY",'idx':'2',"status":"ON","time":"5","pass":"0"}
{"type":"JOGGING",'idx':'1',"status":"ON","time":"5","pass":"0"
}
{"type":"ON/OFF",'idx':'2',"status":"OFF","time":"0","pass":"0"}</td></tr>
<tr><td rowspan="5">/kortex/relaySN/out/relayX
example:
/kortex/relay1868/out/ relay1</td><td rowspan="5">publish</td><td>parameter</td><td>filed</td><td>data</td></tr>
<tr><td>idx</td><td>relay index</td><td>1~2</td></tr>
<tr><td>status</td><td>relay status</td><td>ON,OFF</td></tr>
<tr><td colspan="3">example:
{“idx”:”1”,”status”:”OFF”}</td></tr>
</table>

5.4 MQTT Topic (firmware version >= V2.17.xx)

ETH: firmware version >= V2.17.xx

WiFi: firmware version >= V1.0.xx

<table>
<tr><td>topic</td><td>type</td><td colspan="3">value</td></tr>
<tr><td rowspan="8">ETH
/kortex/relaySN/in/control
WiFi
/kortex/wrelaySN/in/control

example:
/kortex/relay1868/in/control
/kortex/wrelay1868/in/control</td><td rowspan="8">subscribe</td><td>parameter</td><td>filed</td><td>data</td></tr>
<tr><td>type</td><td>command type</td><td>ON/OFF
DELAY
JOGGING</td></tr>
<tr><td>idx</td><td>relay index</td><td>1~2</td></tr>
<tr><td>status</td><td>relay status</td><td>ON, OFF</td></tr>
<tr><td>time</td><td>time for type</td><td>ON/OFF:0
DELAY:1~65535second
JOGGING:1~255*100m
s</td></tr>
<tr><td>pass</td><td>password</td><td>0~9999</td></tr>
<tr><td colspan="3">example:
{"type":"ON/OFF",'idx':'1',"status":"ON","time":"0","pass":"0"}
{"type":"DELAY",'idx':'2',"status":"ON","time":"5","pass":"0"}
{"type":"JOGGING",'idx':'1',"status":"ON","time":"5","pass":"0"}
{"type":"ON/OFF",'idx':'2',"status":"OFF","time":"0","pass":"0"}</td></tr>
</table>

ETH /kortex/relaySN/in/rX WiFi /kortex/wrelaySN/in/rX example: /kortex/relay1868/in/r1 /kortex/relay1868/in/r2 /kortex/wrelay1868/in/r1 /kortex/wrelay1868/in/r2	subscribe	X:1~2 value: ON,OFF
ETH /kortex/relaySN/out/rX WiFi /kortex/wrelaySN/out/rX example: /kortex/relay1868/out/r1 /kortex/relay1868/out/r2 /kortex/wrelay1868/out/r1 /kortex/wrelay1868/out/r2	publish	X:1~2 value: ON,OFF

ETH /kortex/relaySN/out/iX WiFi /kortex/wrelaySN/out/iX example: /kortex/relay1868/out/i1 /kortex/relay1868/out/i2 /kortex/wrelay1868/out/i1 /kortex/wrelay1868/out/i2	publish	X:1~2 value: ON,OFF
ETH /kortex/relaySN/out/relayX WiFi	publish	parameter / filed / data idx / relay index / 1~2 status / relay status / ON, OFF

<table>
<tr><td>/kortex/wrelaySN/out/relayX

example:
/kortex/relay1868/out/relay1
/kortex/relay1868/out/relay2
/kortex/wrelay1868/out/relay1
/kortex/wrelay1868/out/relay2</td><td></td><td colspan="3">example:
{“idx”:”1”,”status”:”OFF”}</td></tr>
<tr><td>ETH
/kortex/relaySN/out/inputX
WiFi
/kortex/wrelaySN/out/inputX

example:
/kortex/relay1868/out/input1
/kortex/wrelay1868/out/input1</td><td>publish</td><td>parameter</td><td>filed</td><td>data</td></tr>
<tr><td></td><td></td><td>idx</td><td>relay index</td><td>1~2</td></tr>
<tr><td></td><td></td><td>status</td><td>relay status</td><td>HIGH, LOW</td></tr>
<tr><td></td><td></td><td colspan="3">example:
{"idx":"1","status":"HIGH"}
{"idx":"1","status":"LOW"}</td></tr>
<tr><td>ETH
/kortex/relaySN/out/ip
WiFi
/kortex/wrelaySN/out/ip

example:
/kortex/relay1868/out/ip
/kortex/wrelay1868/out/ip</td><td>publish</td><td colspan="3">example:
192.168.1.100</td></tr>
<tr><td>ETH
/kortex/relaySN/out/sn
WiFi
/kortex/wrelaySN/out/sn

example:
/kortex/relay1868/out/sn</td><td>publish</td><td colspan="3">example:
1868</td></tr>
</table>

<table>
<tr><td>/kortex/relay1868/out/sn</td><td></td><td></td></tr>
<tr><td>ETH
/kortex/relaySN/out/mac
WiFi
/kortex/wrelaySN/out/mac

example:
/kortex/relay1868/out/mac
/kortex/relay1868/out/mac</td><td>publish</td><td>example:
bc:34:88:00:00:00</td></tr>
</table>

ETH /kortex/relaySN/out/input_cnt WiFi /kortex/wrelaySN/out/input_cnt example: /kortex/relay1868/out/input_cnt /kortex/wrelay1868/out/ input_cnt	publish	2
ETH /kortex/relaySN/out/relay_cnt WiFi /kortex/relaySN/out/relay_cnt example: /kortex/relay1868/out/relay_cnt /kortex/wrelay1868/out/relay_cnt	publish	2

5.5 MQTT LWT topic

ETH: firmware version >= V2.17.188

WiFi: firmware version >= V1.0.449

topic	type	value
ETH /kortex/relaySN/out/lwt_availability WiFi /kortex/wrelaySN/out/lwt_availability example /kortex/relay1868/out/lwt_availability /kortex/wrelay1868/out/ lwt_availability	publish	online,offline

6 Protocol:CoAP

Relay board as CoAP server, accept CoAP Client request.

Support relay on/off

Support relay jogging

Support relay delay Support password verification

you need linux system to compile libcoap

6.1 Compile libcoap

git clone --recurse-submodules https://github.com/obgm/libcoap

./autogen.sh

./configure --disable-manpages --enable-examples --enable-tests

make

6.2 Get relay status

Relay Status(1:ON, 0:OFF)

./coap-client -m get coap://192.168.1.100/kortex/r1

./coap-client -m get coap://192.168.1.100/kortex/r2

Input Status(1:High, 0:Low)

./coap-client -m get coap://192.168.1.100/kortex/i1

./coap-client -m get coap://192.168.1.100/kortex/i2

6.3 Control relay(simple)

```
./coap-client -e "1" -m put coap://192.168.1.100/kortex/r1      # relay1 ON
./coap-client -e "0" -m put coap://192.168.1.100/kortex/r1      # relay1 OFF
./coap-client -e "1" -m put coap://192.168.1.100/kortex/r2      # relay2 ON
./coap-client -e "0" -m put coap://192.168.1.100/kortex/r2      # relay2 OFF
```

6.4 Control relay

format:

status:type:time:password

parameter	filed	data	comment
status	relay status	0,1	
type	ON/OFF DELAY JOGGING		
time	time for type	ON/OFF:0 DELAY:1~65535second JOGGING:1~255*100ms	
password	password	0~9999	

example:

1:ON/OFF:0:4660

status:1

type:ON/OFF

time:0

password:4660

ON/OFF example:

./coap-client -e "1:ON/OFF:0:4660" -m put coap://192.168.1.100/kortex/r1

./coap-client -e "1:ON/OFF:0:4660" -m put coap://192.168.1.100/kortex/r2

DELAY example:

./coap-client -e "1:DELAY:5:4660"-m put coap://192.168.1.100/kortex/r1

./coap-client -e "1:DELAY:5:4660" -m put coap://192.168.1.100/kortex/r2

JOGGING example:

./coap-client -e "1:JOGGING:5:4660" -m put coap://192.168.1.100/kortex/r1

./coap-client -e "1:JOGGING:5:4660" -m put coap://192.168.1.100/kortex/r2

./coap-client -e "0:JOGGING:5:4660" -m put coap://192.168.1.100/kortex/r1

./coap-client -e "0:JOGGING:5:4660" -m put coap://192.168.1.100/kortex/r2

CONTACTS

Address

KORTEX PSI
8, rue Auber
75009 Paris
Tel: +33(0)1 34 04 37 60

Sales Support

Sales support is available at the following email address: contact@kortex-psi.fr.

Technical Support

0 820 6 9 0 0 4 2 Service 0,12 € / min + prix appel

Technical support can be accessed at the following email address: technique@kortex-psi.fr.

Documentation

KORTEX PSI strives to constantly improve the understanding and proper use of its product documentation. Constructive feedback from users is significant for KORTEX PSI. Please send your comments and suggestions regarding the documentation to: technique@kortex-psi.fr.