Accessing robot data through the MODBUS server

(Important new updates to be seen at the end of page)

Purpose:

Give read and write access to data in the robot controller for other devices

How it works:

The robot controller acts as a Modbus TCP server (port 502), clients can establish connections to it and send standard MODBUS requests to it. The server is available at the IP address of the controller which can be found and modified in PolyScope (SETUP Robot → Setup NETWORK) For more information about MODBUS, see www.modbus.org specifications of MODBUS messages can be found at http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

Functionality:

- Several clients can connect to the server at the same time
- The server can respond to the following function codes:
  - 0x01: READ_COILS (read output bits)
  - 0x02: READ_DISCRETE_INPUTS (read input bits)
  - 0x03: READ_HOLDING_REGISTERS (read output registers)
  - 0x04: READ_INPUT_REGISTERS (read input registers)
  - 0x05: WRITE_SINGLE_COIL (write output bit)
  - 0x06: WRITE_SINGLE_REGISTER(write output register)
  - 0x0F: WRITE_MULTIPLE_COILS (write multiple output bits)
  - 0x10: WRITE_MULTIPLE_REGISTERS (write multiple output registers)
- The server will send a response to all requests
- The server responds error messages with exception codes:
  - 0x01: ILLEGAL_FUNCTION_CODE
  - 0x02: ILLEGAL_DATA_ACCESS (if the request address is illegal)
  - 0x03: ILLEGAL_DATA_VALUE (if the request data is invalid)
- The 16 bit ports can be read using the script function `read_port_register(<address>)`, many of the ports, including 128 general purpose registers (address 128-256) (see the port map) can also be written to using `write_port_register(<address>,<value>)`. You can use the script function `integer_to_binary_list(<value>)` to get the register value as a list of booleans (e.g.[True,False,.....]), or `binary_list_to_integer(<list>)` to go back, see the script manual ManualsForURCustomers for details.
- Coils (digital ports)can be accessed using `read_port_bit(<address>)` and `write_port_bit(<address>,<value>)`.
- Note that all values are unsigned, if you want to convert to signed integers, program "if (val > 32768): val = val - 65535".
- The MODBUS Server has 0-based addressing. Be aware that some other devices are 1-based (e.g. Anybus X-gateways), then just add one to the address on that device. (e.g address 3 on the robot will be address 4 on the Anybus X-gateway)

Setup:

Setup the controller IP and your MODBUS client with static IP addresses in the same subnet, see ModbusSetupGuide

Port map 16bit register addresses:

- Use function codes 0x03: READ_HOLDING_REGISTERS, 0x04: READ_INPUT_REGISTERS, 0x06: WRITE_SINGLE_REGISTER, and 0x10: WRITE_MULTIPLE_REGISTERS to access the following addresses.
- Use the script functions `read_port_register(<address>)` and `write_port_register(<address>, <value>)` to read and write to those addresses on your robot from the running program.

```
 R=Read, W=Write

  Address   R W

     0      x   Inputs, bits 0-15 [xxxxxxTTBBBBBBBB] x=undef, T=tool, B=box
     1      x x Outputs, bits 0-15 [xxxxxxTTBBBBBBBB] x=undef, T=tool, B=box
     2        x SetOutputsBitsMask 0-15 [xxxxxxTTBBBBBBBB] x=undef, T=tool, B=box
     3        x ClearOutputsBitsMask 0-15 [xxxxxxTTBBBBBBBB] x=undef, T=tool, B=box
     4      x   Analog input 0  (0-65535)
     5      x x Analog input 0 domain e {0=current[mA], 1=voltage[mV]}
     6      x   Analog input 1  (0-65535)
     7      x x Analog input 1 domain e {0=current[mA], 1=voltage[mV]}
     8      x   Analog input 2 (tool) (0-65535)
     9      x x Analog input 2 (tool) domain e {0=current[mA], 1=voltage[mV]}
    10      x   Analog input 3 (tool) (0-65535)
    11      x x Analog input 3 (tool) range e {0=current[mA], 1=voltage[mV]}
    16      x x Analog output 0 output (0-65535)
    17      x x Analog output 0 output domain e {0=current[mA], 1=voltage[mV]}
    18      x x Analog output 1 output (0-65535)
    19      x x Analog output 1 output domain {0=current[mA], 1=voltage[mV]}
    20      x x Tool output voltage (V) e {0V, 12V, 24V}
    24      x   Euromap67 input bits (0-15)
    25      x   Euromap67 input bits (16-32)
    26      x   Euromap67 output bits (0-15)  (read only!)
    27      x   Euromap67 output bits (16-32) (read only!)
    28      x   Euromap 24V voltage
    29      x   Euromap 24V current

    30-127      Spare (undef)
    128-255 x x General purpose 16 bit registers
    256-    x   Robot state
    512-    x   Program state
    768-    x   Tool states
    1024-   x   GUI state
    2048-   x   RT Machnie control

Robot state

256  Controller version high nuber
257  Controller version low number
258  Robot mode (see http://support.universal-robots.com/Technical/PolyScopeProgramServer )
260  isPowerOnRobot
261  isSecurityStopped
262  isEmergencyStopped
263  isTeachButtonPressed
264  isPowerPuttonPressed
265  isSafetySignalSuchThatWeShouldStop


270  Base joint angle (in mrad)
271  Shoulder joint angle (in mrad)
272  Elbow joint angle (in mrad)
273  Wrist1 joint angle (in mrad)
274  Wrist2 joint angle (in mrad)
275  Wrist3 joint angle (in mrad)

280  Base joint angle velocity (in mrad/s)
281  Shoulder joint angle velocity (in mrad/s)
282  Elbow joint angle velocity (in mrad/s)
283  Wrist1 joint angle velocity (in mrad/s)
```

```
284  Wrist2 joint angle velocity (in mrad/s)

285  Wrist3 joint angle velocity (in mrad/s)


290  Base joint current (in mA)

291  Shoulder joint current (in mA)

292  Elbow joint current (in mA)

293  Wrist1 joint current (in mA)

294  Wrist2 joint current (in mA)

295  Wrist3 joint current (in mA)


300  Base joint temperature (in C)

301  Shoulder joint temperature (in C)

302  Elbow joint temperature (in C)

303  Wrist1 joint temperature (in C)

304  Wrist2 joint temperature (in C)

305  Wrist3 joint temperature (in C)


Joint modes (see http://support.universal-robots.com/Technical/ListOfJointModes) From version 1.7

310  Base joint mode

311  Shoulder joint mode

312  Elbow joint mode

313  Wrist1 joint mode

314  Wrist2 joint mode

315  Wrist3 joint mode


TCP

400  TCP-x in tenth of mm (in base frame)

401  TCP-y in tenth of mm (in base frame)

402  TCP-z in tenth of mm (in base frame)

403  TCP-rx in mrad (in base frame)

404  TCP-ry in mrad (in base frame)

405  TCP-rz in mrad (in base frame)


410  TCP-x speed in mm/s (in base frame)

411  TCP-y speed in mm/s (in base frame)

412  TCP-z speed in mm/s (in base frame)

413  TCP-rx speed in mrad/s (in base frame)

414  TCP-ry speed in mrad/s (in base frame)

415  TCP-rz speed in mrad/s (in base frame)


420  TCP-x offset in mm (in tool frame)

421  TCP-y offset in mm (in tool frame)

422  TCP-z offset in mm (in tool frame)

423  TCP-rx offset in mrad (in tool frame)

424  TCP-ry offset in mrad (in tool frame)

425  TCP-rz offset in mrad (in tool frame)


450  Robot current (in mA)

451  I/O current (in mA)


768  Tool state

769  Tool temperature (in C)

770  Tool current (in mA)


GUI state


1024  GUI state not implemented


RTMachine


 Address    R W
```

```
2048        x Split time (latches the actual time to the registers 2049-2053) calculates the time from last time the controller was restarted
2049      x   Milliseconds
2050      x   Seconds
2051      x   Minutes
2052      x   Hours
2053      x   Days
```

Port map bit (Digital) addresses

- Use function codes 0x01: READ_COILS, 0x02: READ_DISCRETE_INPUTS, 0x05: WRITE_SINGLE_COIL, 0x0F: WRITE_MULTIPLE_COILS to access the following addresses.
- Use the script functions `read_port_bit(<address>)` and `write_port_bit(<address>, <value>)` to read and write to those addresses on your robot from the running program.

```
Address  R W

 0-15    x   Inputs, bits 0-15 [xxxxxxTTBBBBBBBB] x=undef, T=tool, B=box
 16-31   x x Outputs, bits 0-15 [xxxxxxTTBBBBBBBB] x=undef, T=tool, B=box
 32-47     x SetOutputsBitsMask 0-15 [xxxxxxTTBBBBBBBB] x=undef, T=tool, B=box
 48-63     x ClearOutputsBitsMask 0-15 [xxxxxxTTBBBBBBBB] x=undef, T=tool, B=box
 64-79   x   Euromap67 input bits (0-15)
 80-95   x   Euromap67 input bits (16-32)
 96-111  x   Euromap67 output bits (0-15)  (read only!)
 112-127 x   Euromap67 output bits (16-32) (read only!)

 260     x   isPowerOnRobot
 261     x   isSecurityStopped
 262     x   isEmergencyStopped
 263     x   isTeachButtonPressed
 264     x   isPowerPuttonPressed
 265     x   isSafetySignalSuchThatWeShouldStop
```

Important info for Server applications with Siemens clients(masters)

From UR Software version 1.7 and on:

Some MODBUS units use designated addresses for each function code, i.e. Siemens use: coils (generally addressed as 0xxxx), contacts (1xxxx), input registers (3xxxx), holding registers (4xxxx). To solve this issue, address "x" is reflected at 10000+"x", 20000+"x", 30000+"x" and 40000+"x" Address 0 is also reflected at address 9999, as some units have an addresses starting at 1

The value "Unit Identifier"/"Slave ID" is ignored in the MODBUS Server

Topic revision: r16 - 25 Jul 2013 - 06:05:34 - OlufNielsen