V24KOME.BES Some useful information about low level communication

Author: S. Klauser (P&P Elektronik)
Date: 22.03.1990
14.09.1990 some corrections

Some useful information about the low level communication between ROB3 / ROB3i using the RS232C interface.

1. Some general information about the communication between robot and host computer (normally a PC) using the RS232C interface:

1.1 The communication works without hardware handshake, that means only 3 lines of the interface cable are used:

- TxD PIN 3 robot
- RxD PIN 2 robot
- Signal Ground PIN 1 robot

1.2 Up till now there is no standardized protocoll used for the communication. But by most of the commands the robot returns a message (for example: position reached or the actual axis values) to the host computer.

1.3 Every command invoked by the host computer must be closed by the ETX (03 hex) char.

1.4 In a return messsages of the robot to the host, the robot normally echos the command byte (but not at all commands).

2. Start up of the communication and baud-rate-fixing:

2.1 The interface of the robot detects after a RESET of the robot automatically the baudrate and adjustes the baud rate automatically.

The host computer must transmitt first a SPACE (20 hex) to the robot.

2.2 The answers of the robot to the SPACE - byte during start up of the communication:

15 hex	Init Okay, Analog (electrical) gripper
16 hex	Init Okay, Digital (pneumatic) gripper (only ROB3i)
F1 Hex	Multiple Init of the robot
F2 Hex	Multiple Init of the robot
F3 Hex	Multiple Init of the robot

Other responses of the robot to the SPACE byte are not allowed. That means the communication does not work correctly.

1. 3
13
3 33

The following chapters describe the commands to be invoked by the host and the responses of the robot:

The numbers in the commands are all written in HEX or binary format.

1. activation commands:

1.1 axis position commands (00 - 0F hex):

1.1.1 single axis position command:

command byte data
0 0 0 0 R .a.., 1 byte position (binary), ETX

a: axis number (0 - 5)
R: 1 (the robot has to return a "position reached" message)
R: 0 (the robot returns nothing)
1st position byte: eight most significant bits of position
2nd position byte: last significant bit of position (0 or 1)

example:

drive axis 2 to position 300D (= 100101100B = 012CH) with return when position is reached:

command byte, data, ETX
0 0 0 0 1 0 1 0, 96H, ~~00H~~, ETX

1.1.2 multiple axis position command:

command byte data
0 0 0 0 R 1 1 1, 6 byte position (binary), ETX

R = 1 (the robot has to return a "position reached" message)
R = 0 (the robot returns nothing)
first 6 bytes of position data: eight most significant bits of position data
7th byte of position data: 6 least significant bits of position data (with bit 0 = axis 0, bit 1 = axis 1 etc.)

Returnmessage from robot:

command byte, ETX (that means robot has reached pos.)

example:

drive all axis to position 300D (= 100101100B = 012CH) with return when position is reached:

command byte, data, ETX
0 0 0 0 1 1 1 1, 96H, 96H, 96H, 96H, 96H, 96H, ~~00H~~, ETX

1.2 digital output commands:

```
command byte      data
0 0 0 1 0 0 a o,  7 6 5 4 3 2 1 0, ETX
                  <------bits------>
```

=> 3 bytes

a	o	
0	0	put the mask direct to the digital outputs
0	1	logical OR with the actual port values
1	0	logical AND with the actual port values
1	1	logical XOR with the actual port values

example:

set following bit mask direct at the digital output port:

DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
1	1	0	0	1	0	1	0

(DO = digital output)

```
command byte      data, ETX
0 0 0 1 0 0 0 0, CAH,  ETX
```

1.3 servo control commands:

command byte

60, ETX = turn off servo control of DC motors
61, ETX = turn on servo control of DC motors
62, ETX = turn off position control

1.4 check of the internal serial number of the robot:

command byte

63, ETX = read out serial number of robot

return message from the robot:

63, S0, S1, S2, ETX

S0, S1, S3 = serial number of the robot

1 34 07
100
3400

10000
3400
9
13409

3. axis position commands with speed:

After recognizing the command the previous position will be incremented or decremented by 1 after running out of the time T until the endposition is reached.

The time T is to calculated as follows:

T = "time" * 10 ms

If the "time" in the command has the value 0, then the desired endposition will be reached with maximum speed (see also point 1.1 of this manual).

3.1 single axis positon command with speed:

command byte data
0 1 1 1 R .a., 1 byte position (binary), 1 byte "time" ,ETX

a = axis number (0 - 5)
R = 1; the returns a "position reached" message
R = 0; the robot returns nothing

Returnmessage:

command byte, ETX

(the position data is coded the same way as described under point 1.1.1 of this manual)

3.2 multiple axis position command with speed:

command byte data
0 1 1 1 R 1 1 1, 6 byte position (binary), 6 byte "time", ETX

R = 1; the returns a "position reached" message
R = 0; the robot returns nothing

Returnmessage:

command byte, ETX

(the position data is coded the same way as described under point 1.1.2 of this manual)

Descrição dos comandos a enviar ao robô 2

Por lapso as folhas em anexo não incluem os comandos para leitura do valor dos eixos que têm o seguinte formato:

Ler um eixo:

- Command Byte, ETX (01000a, 3) em que "a" são os três bits que indicam o valor do eixo cuja posição se pretende conhecer.

- O controlador devolve Command Byte, valor do eixo, ETX.

Ler todos os eixos:

- Command Byte, ETX (01000111, 3).

- O controlador devolve Command Byte, valor e1, valor e2, valor e3, valor e4, valor e5, valor e6, ETX.

Nota: Para expandir as expressões no Mathematica retirem o ";" do final da linha que querem avaliar e façam Kernel - Evaluation - Evaluate NoteBook.