MODULAR CONTROL SYSTEM FOR THE FAMILY OF EDUCATIONAL ROBOTS "ROBCO"

N. Chivarov*, P. Kopacek** and N. Shivarov*

*Central Laboratory of Mechatronics and Instrumentation,
Bloc II Acad. Bonchev str., Sofia 1113 Bulgaria
e-mails: nshivarov@code.bg and nedko@bgcict.acad.bg
** Department of Intelligent Handling and Robotics
Vienna University of Technology
e-mail: kopacek@ ihrt.tuwien.ac.at

Abstract: Our family of educational robots ROBCO includes (ROBCO 01 Articulated robot, ROBCO Phoenix Articulated minirobot, ROBCO SCARA robot, ROBCO Cylindrical robot and ROBCO Spherical robot). The mechanical systems of ROBCO educational robots consist of a base, modules for relative translation, modules for relative rotation and a gripper.
The easiness in assembling our module together and the simple connection to the control system make possible creation of different type of robots that possess similar structure of joints for performing variety of tasks. The purpose of this paper is to present a Control System for easy control and programming of all those types and classes of robots, allowing for both manual and automated user control and configuration.

Keywords: *Educational robot, Articulated robot; SCARA robot, Cylindrical robot, Spherical robot.*

1. Introduction

In the late 80's educational robot ROBCO 01 was developed [1].

It's mechanical system was simple and strings driven using stepper motors.

ROBKO 01 is six degree of freedom articulated minirobot consisting of base, three links and a gripper driven with stepper motors, gears and rollers (see Fig.1).

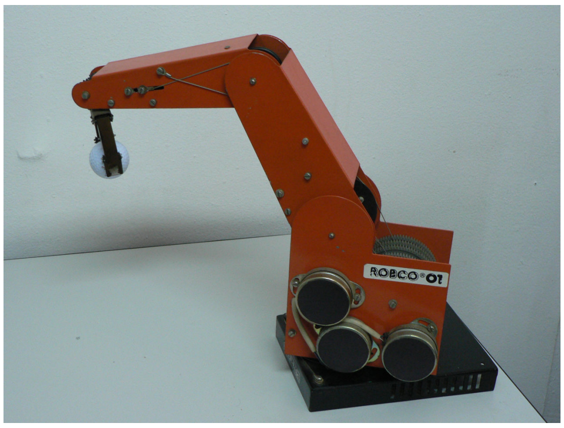

Fig. 1 Articulated Educational Robot – ROBKO 01

It was produced in a large series for domestic use and export in all former socialist countries and some west ones (small series for USA, Argentina, France etc.). They have been produced over 18000 pieces. This robot had a great impact on introducing robotics to the young generation of that time.

Having our ROBCO 01 as a good example tool for learning about robotics and interaction between robots and humans we have develop a family of Educational Robots for a variety of applications (ROBCO 01 Articulated robot, ROBCO Phoenix Articulated minirobot, ROBCO SCARA robot, ROBCO Cylindrical robot and ROBCO Spherical robot).

Educational Robotics is multidisciplinary scientific field [2] which includes mechanics, electronic hardware, software, artificial intelligences, sensor and sensory systems etc. and is a good test-bed for educating students, young specialist, researchers and is necessary for all technical schools, colleagues, laboratories and Universities.

2. Control System Overview

The common block diagram of the robot is shown on Figure 2. All electronic control modules are interconnected by a system bus. That gives an extreme flexibility and scalability to the whole architecture for performing the largest possible range of tasks of the Robot [3].

All commands to the Robot are passed and processed by the control module Robot Controller. The distribution of different queries and commands to different modules, self-test and detection of system's configuration, as well as the whole robot intelligence at high level is performed by the embedded software of the controller using the system bus Robot System Bus. The immediate control of the stepper motors is implemented by separate intelligent electronic modules. Those modules care on one hand for communication with the Robot Controller using specially designed communications protocol and on the other hand for the immediate physical control of stepper motors. On Fig. 2 an example is shown for a few stepper motors controlled by intelligent modules.

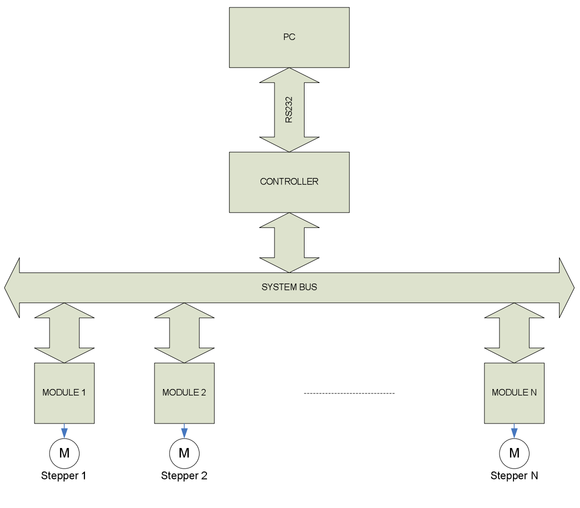

Fig. 2 Control System of Educational Robots ROBCO

The position of actuator mechanism can be monitored by counting the steps from its starting point (the position in which the power is turned on), controlled by the motor and on query – transmitted to the Robot Controller [4]. Modules also implement basic functions for electrical overload protection and other low-level functions.

For the purposes of easy user control over the Robot, a set of commands is defined and supported by its Robot Controller. We created an interface that provides the means for user access to the Controller and its command set – the Command Line Interface (CLI) which is a simple text console for inputting commands and displaying the results.

3. Command Line Interface

Command Line Interface (CLI) is an interface to *Robot's Controller*, allowing the user to control the robot using simple text input and output through RS-232 serial interface using common terminal programs (Hyper Terminal, PuTTY, etc.) or custom software that communicates through PC's serial (COM) port (hardware or emulated through USB, etc.).

This interface allows for custom robot control using user software written in any language (i.e. C++) that utilizes PC's COM port (real or virtual) to send text commands to CLI.

3.1 Common notes

3.1.1 *Terms and symbols*

- Triangular brackets <> denote a parameter with name given between them. The command is therefore entered without those brackets, using the value of the parameter which name they surround.
- Rectangular brackets [] denote an optional parameter (s) that can be omitted. Note: a subsequent optional parameter cannot be provided without providing the previous parameters first. It means that the exact order of inputting the values of parameters is essential and when omitting an optional parameter, the subsequent parameters (if any) must be omitted too.
- A symbol | in parameter definitions delimits possible values of the parameter.
- "Cold reset" means powering the device down, waiting for 1 minute and powering up again.

3.1.2 Communication settings

RS-232 communication has the following settings:

- **Cable:** DB9 or DB25 (depends on PC side, usually DB9) Female to DB9 Male

Straight Serial Cable (only Rx, Tx and GND pins are used);

- **Baud rate:** 9600
- **Data bits:** 8
- **Parity:** None
- **Stop bits:** 1
- **Flow control:** None
- **Note:** The interface is compatible with common USB-to-RS232 (COM) port adapters. A connection to modern desktop PCs and laptops (without COM ports) requires such an adapter.

3.1.3 *Other notes*

- Besides the described command outputs below, CLI transmits back an echo for each input symbol, as well as "carriage return" followed by "line feed" (0x0d, 0x0a) characters at the end of each line. At the beginning of each command line CLI also inserts a prompt (the symbol ">" that can be preceded by a text i.e. "Settings>" or just ">"). This information is important when an own custom (user) control software is being designed.
- Each CLI command keyword (command keywords only, not parameters!) can be abbreviated to the least possible string that is not a beginning of another command, i.e. "**stop**" can become "**sto**", "**st**" but not "**s**" because there is a command keyword **show**.

3.2 Command descriptions

3.2.1 Show Joint Position

- **Description:** displays the current position (a signed integer number) of a joint.
- **Format**:

jp *<jnum>*

<jnum> - a number indicating the joint address (0 – (N-1)), N- number of joints

- **Output:** an integer value indicating the current position of the given joint, always in half steps from the initial position.
- **Example:**

>jp 0

-148

>

3.2.2 *Show Joint Status*

- **Description:** displays the current status of a joint. Usually a joint has status READY (displayed value 0) when not moving, and BUSY (displayed value 1) while executing a command. ERR status (displayed result value 2) is displayed when a joint module is not present (too big *jnum*) or on another abnormal condition. The latter is most often fixed by "cold resetting" the robot.
- **Format**:

jstatus *<jnum>*

or

js *<jnum>*

<jnum> - a number indicating the joint address (0 – (N-1)), N- number of joints

- **Output:** an integer value indicating the current status of the given joint: 0 for READY, 1 for BUSY, 2 for ERROR.
- **Example:**

>js 1

0

>

3.2.3 Move Joint Forward

- **Description:** Rotates a joint in the positive direction.
- **Format**:

fw *<jnum> [<step_delay> [<steps> [<fh> [-s]]]]*

jnum - a number indicating the joint address (0 – (N-1)), N- number of joints;

step_delay – a positive integer giving the delay (in milliseconds) between 2 steps. Can take values between 1 and 255;

steps – a positive integer or 0 – the number of full or half (see *fh* parameter below) steps to be performed; a value of 0 for this parameter forces the joint to rotate until stopped by another command or until it reaches its mechanical limitations;

fh – sets the step mode: 1 for full stepping, 0 for half stepping;

-s – when provided, this parameter causes CLI to synchronize with command completion, i.e. the output is displayed after the command

execution is complete. *Note: if steps=0 providing –s parameter results in locking out of CLI and the robot must be cold-reset in order to regain control!*

- **Output:** "ok" on success, or "err *n*" on failure; *n* is the error code.
- **Example:**

>fw 0 20 100 0 -s

ok

>

3.2.4 Move Joint Backward

- **Description:** Rotates a joint in the negative direction.
- **Format**:

bw *<jnum> [<step_delay> [<steps> [<fh> [-s]]]]*

jnum - a number indicating the joint address (0 – (N-1)), N- number of joints;

step_delay – a positive integer giving the delay (in milliseconds) between 2 steps. Can take values between 1 and 255;

steps – a positive integer or 0 – the number of full or half (see *fh* parameter below) steps to be performed; a value of 0 for this parameter forces the joint to rotate until stopped by another command or until it reaches its mechanical limitations;

fh – sets the step mode: 1 for full stepping, 0 for half stepping;

-s – when provided, this parameter causes CLI to synchronize with command completion, i.e. the output is displayed after the command execution is complete. *Note: if steps=0 providing –s parameter results in locking out of CLI and the robot must be cold-resetted in order to regain control!*

- **Output:** "ok" on success, or "err *n*" on failure; *n* is the error code.
- **Example:**

>bw 0 20 100 0 -s

ok

>

3.2.5 Stop a Joint

- **Description:** Stops joint rotation.
- **Format**:

stop *<jnum> [-b | -r]*

jnum - a number indicating the joint address (0 – (N-1)), N- number of joints; a value of ***all*** can be provided instead of a number; then *stop* command is issued to all joints available;

-b or -s – when provided, this parameter determines the ***stop mode*** of a joint. The ***–b*** value means "brake stop" (joint is difficult to be moved manually), and the ***–r*** mode means "release stop" (joint's motor is disconnected and if possible, it is easier to be moved manually). This mode remains valid until another stop command explicitly changes it, or until power down. *Note: "brake stop"* mode consumes more power than *"release stop"* and should be applied only when necessary! *Note 2:* The stop mode defines the stop behavior of all other commands: i.e. after completion of *"fw", "bw", "jg"* commands the joint stops on the stop mode, defined by the last *"stop"* command that provided this parameter. On power on, the default is *"release stop". This means that on "release stop" mode, a joint may "roll back" driven by some external cables, gravity etc. In such cases, the joint must be given "brake stop" mode once and it will operate normally afterwards. This can be done from the example Robot Arm Control software, provided with the robot and operating under Microsoft Windows.*

- **Output:** "ok" on success, or "err *n*" on failure; *n* is the error code, and it can be absent. Note: If *"all"* value is provided, the ***stop*** command does not produce output!
- **Example 1:**

>stop 1

ok

>

- **Example 2:**

>stop all

ok

>

- **Example 3:**

>stop 3 –b

ok

>

3.2.6 Move Joint to a Position

- **Description:** Rotates a joint to a given position.
- **Format**:

jg *<jnum> [<step_delay> [<pos> [<fh> [-s]]]]*

jnum - a number indicating the joint address (0 – (N-1)), N- number of joints;

step_delay – a positive integer giving the delay (in milliseconds) between 2 steps. Can take values between 1 and 255;

pos – a signed integer – the position to be reached. This value is provided always in half steps, and if *fh* parameter is 1 (see below), it will be reached twice as fast;

fh – sets the step mode: 1 for full stepping, 0 for half stepping;

-s – when provided, this parameter causes CLI to synchronize with command completion, i.e. the output is displayed after the command execution is complete.

- **Output:** "ok" on success, or "err *n*" on failure; *n* is the error code.
- **Example:**

```
>jg 3 20 100 0 -s
ok
>
```

4. Conclusion

As it was described in the paper, we have produced family of educational robots ROBCO which can be easy assembled from our modules and simple connect to the developed control system for performing variety of tasks

The System Controller that we have developed possess several key features that allow extreme flexibility, easy control and programmability and future development of Robots covering wide range of applications not only from educational field but in Service and Industry tasks.

References:

[1] Shivarov, N.; *Educational robots and flexible manufacturing systems for use in training;* United Nations, Economic Commission for Europe; Symposium on Management Training Programs and Methods: Implications of New Technologies. Geneva, 17-19 November 1987.

[2] Kopacek, P.; *"Playing with Robots - Robots for Entertainment, Leisure and Hobby - Robotsoccer"*; Talk: eAROB, TU Wien, Wien, Österreich; July 13-14-2007; in: "Proceedings of the First European Workshop on Artificial Life and Robots - eAROB", (2007), 13 - 19.

[3] Chivarov, N., Shivarov, N. and Kopacek, P.; *Educational Articulated Robot - ROBKO PHOENIX,* Robotics and Mechatronics 2008, September 17-21, Varna, Bulgaria; p.207-p.211, ISSN1310-3946.

[4] Chivarov, N., Shivarov, N. and Kopacek, P.; *Mechatronics Educational Robots ROBKO Phoenix;* "International Journal Automation Austria - IJAA" 2009, unpublished, accepted for publication, in press.