KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

instrukcja jest dystrybuowana bezpłatnie

ROBOTICS
Laboratory

INSTRUKCJA
DO LABORATORIUM

Innowacyjna
dydaktyka
bez ograniczeń
- zintegrowany rozwój Politechniki Łódzkiej
- zarządzanie Uczelnią, nowoczesna oferta edukacyjna
i wzmacniania zdolności do zatrudniania,
także osób niepełnosprawnych

Politechnika Łódzka

Laboratory location:
Robotics Laboratory belongs to Institute of Automatic Control, Department of Robot Control and is located in room E-100, second floor, building A12 (Faculty of Electrical, Electronic, Computer and Control Engineering).

Laboratory consists of 5 exercises; four of them are based on real robots and one on simulation software. Therefore, Students must observe safety precautions related to working with moving machines and high voltages.
Due to safety reasons, Students must use cloak room before entering laboratory.
No food and beverages are allowed in laboratory.

Exercises A and B – robots L1 and L2

These two exercises show the structure, principles of operation, and programming rules of simple industrial/educational robots L1, L2, shown in Fig. 1.1. Exercise A is related to robot L1 and exercise B deals with robot L2. In both cases, the main objective for students performing the experiments is to learn a robot programming language and to apply this language to control the system. Students are expected to write an appropriate robot control program. The program should demonstrate manipulating capabilities of the robot under consideration. This should be done using both step-mode and non-stop operating mode. The laboratory set-up is prepared to present the possibility of co-operation between two robots (e.g. robot L1 grips Styrofoam object and transport it to the box carried by robot L2, then L2 releases object to specified location). Therefore, collaboration between two student groups executing exercises A & B is advised.

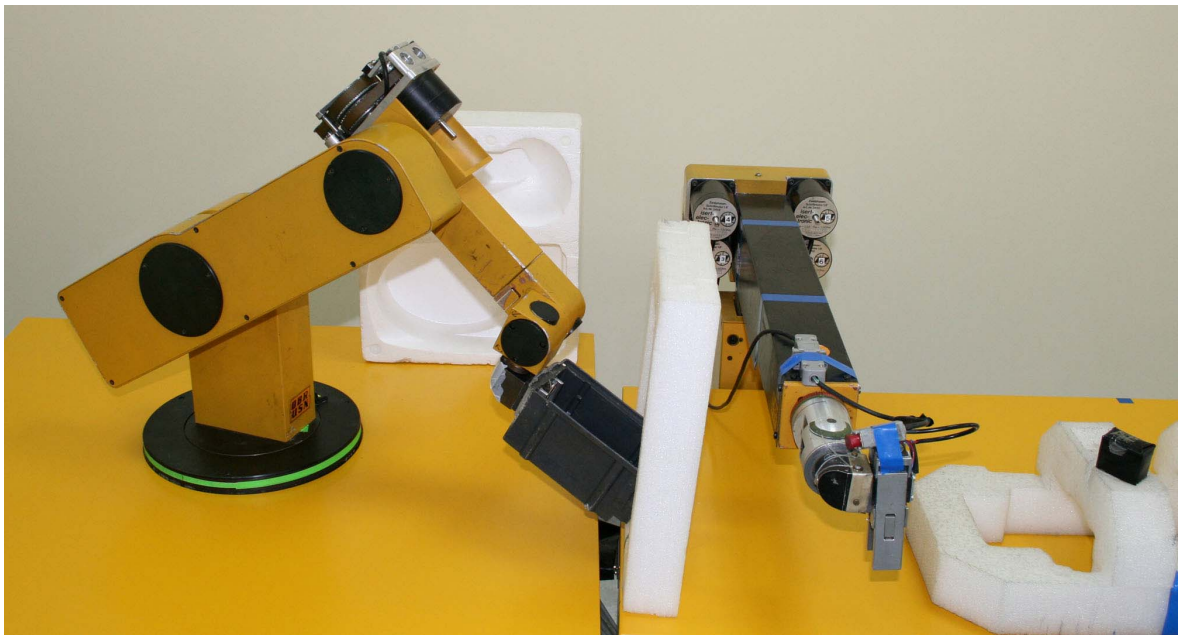

Fig. 1.1 Robots L1 and L2 during co-operation

Introduction

The industrial/educational robots L1 and L2 are the precision facilities which thanks to their un-complicated and reliable construction are useful in didactic laboratories as well as in industrial automation. Precision, simplicity and mobility of the mechanical part give them wide potential of application. These robots can be applied for a mechanical assembly as well as for more complex operations like welding, drilling, painting and short distance transportation. Each robot consists of the manipulator, power driver for stepper motors, and a PC used as the main controller. Such construction gives a possibility to place manipulation and control parts separately, depending on industrial/educational environment.

Manipulator

Robot L1 (the object of exercise A) is a typical Cartesian manipulator with three mutually perpendicular linear joints and a spherical wrist. It has 6 degrees of freedom (DOF) and an electrical gripper. A general view of robot L1 is shown in Fig. 1.2, and its technical data is given in Table 1.1.

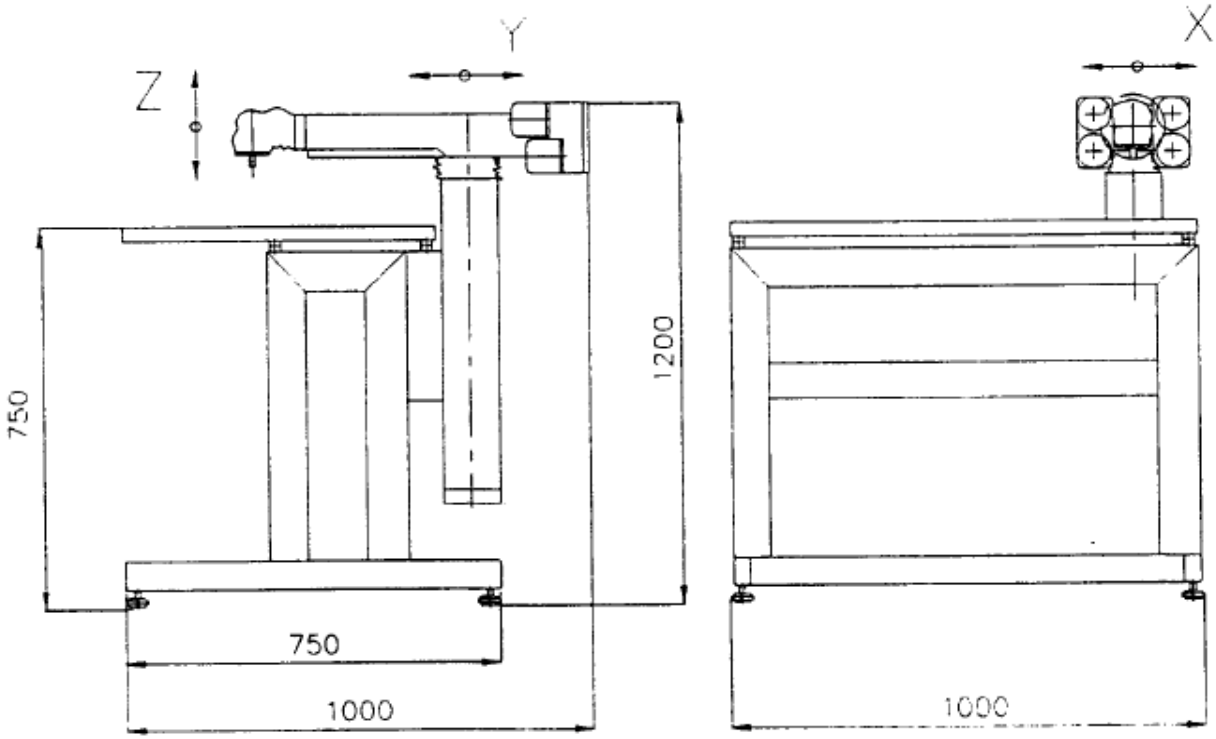

Fig. 1.2 General view of robot L1

Table 1.1 Absolute maximum ratings for robot L1

Maximum load with gripper weight	3.2 kg
Range of movement for 1st Joint	400 mm
Range of movement for 2nd Joint	300 mm
Range of movement for 3rd Joint	160 mm
Range of rotation for 4th Joint (wrist)	n×360 deg
Range of rotation for 5th Joint (wrist)	180 deg
Range of rotation for 6th Joint (wrist)	n×360 deg
Maximum velocity for linear joints	4 m/min
Maximum velocity for rotary joints	0.5 rad/s
Maximum error in Base-Test	± 0.02mm

Robot L2 has a Puma-like kinematic chain with 5DOF. It is equipped with a box for carrying objects. General view of robot L2 is shown in Fig. 1.3 and its technical data is given in Table 1.2.

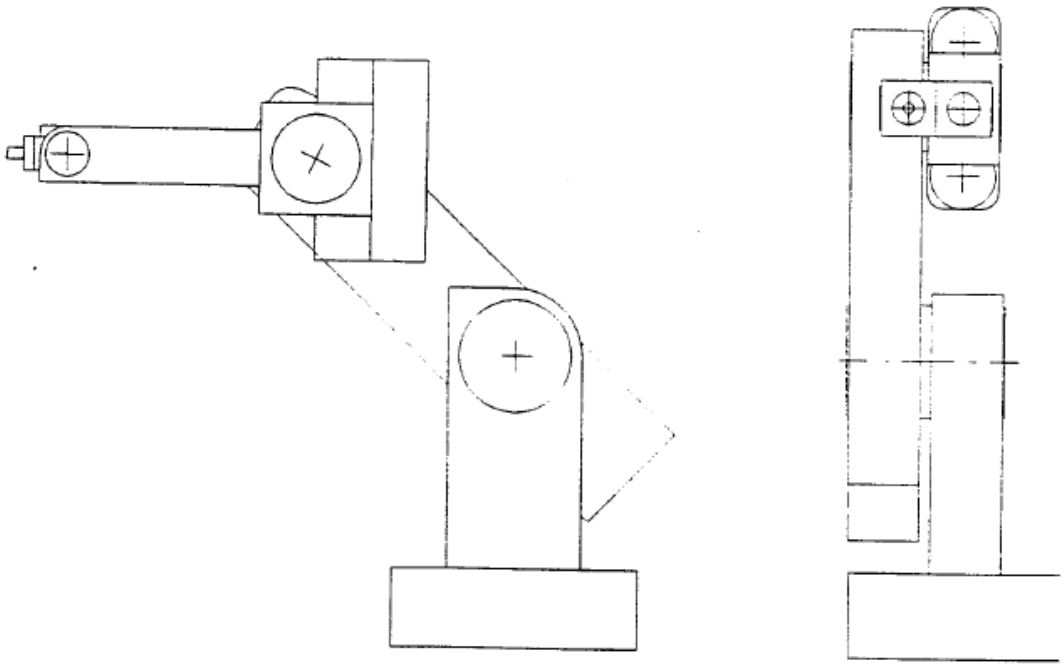

Fig. 1.3 General view of robot L2

Table 1.2 Absolute maximum ratings for robot L1

Maximum load with gripper weight	3.2 kg
Range of rotation for 1st Joint	320 deg
Range of rotation for 2nd Joint	100 deg
Range of rotation for 3rd Joint	270 deg
Range of rotation for 4th Joint (wrist)	170 deg
Range of rotation for 5th Joint (wrist)	340 deg
Maximum velocity for 1st Joint	0.3 rad/s
Maximum velocity for 2nd Joint	0.5 rad/s
Maximum velocity for 3rd Joint	0.5 rad/s
Maximum velocity for 4th Joint	12 rad/s
Maximum velocity for 5th Joint	24 rad/s
Maximum error in Base-Test	± 0.02mm
Precision of positioning (global movements)	± 0.05mm
Precision of positioning (end of wrist)	± 0.15mm
Total weight of manipulating part	18 kg

Control and drive system

Both robots have identical electrical structure. Control system of each axis contains a power controller, a stepper motor and end switches. It works in an open loop. The main part of a control system is a PC equipped with specialized IO cards, a keyboard, DOS operating system and dedicated software. Details of the control system are presented schematically in Fig. 1.4.

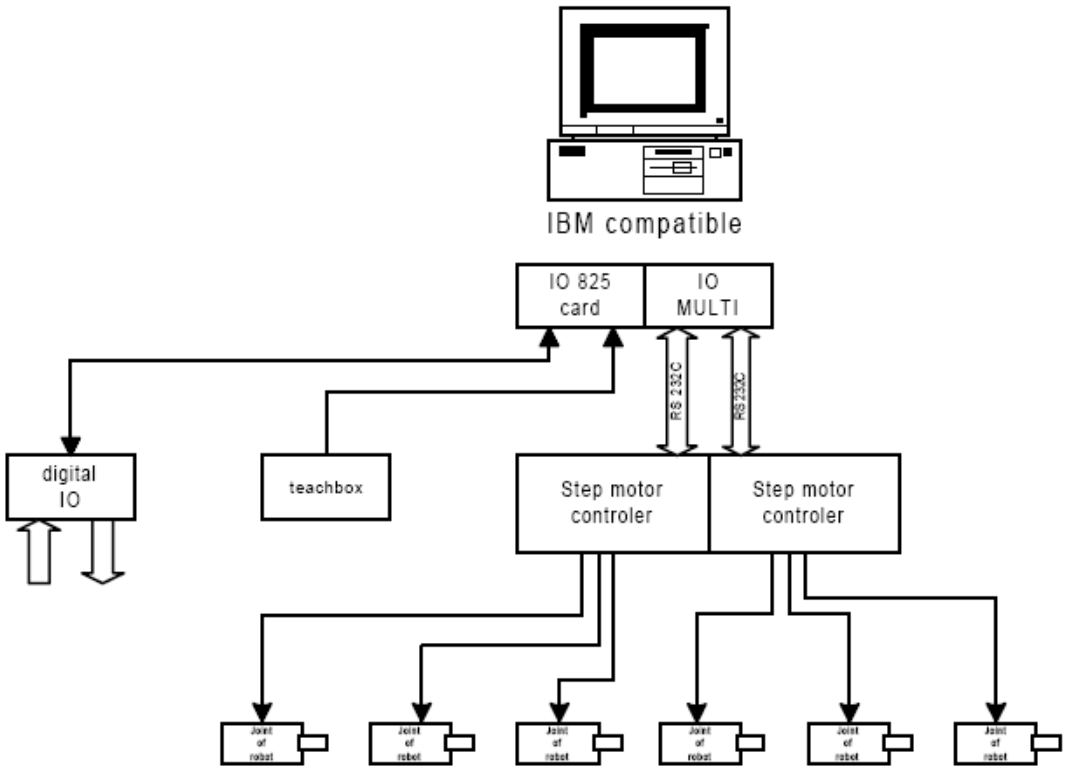

Fig. 1.4 Control system of industrial-didactic robot

Robots working in an open loop are very sensitive to overloads, high accelerations and decelerations, and collisions, therefore, operator should avoid such situations and whenever they happened, the special base-test procedure has to be executed. During base-test procedure each robot's axis is run to home position and associated counters are initialized. For regional movements in robot L1 (linear axes) and for all DOF of robot L2 base procedure is executed automatically under appropriate command of teach panel. In case of local movements of robot L1 (wrist rotations) there is no automatic procedure and therefore operator has to switch these drives off and manually position wrist in the convenient configuration (the best and repeatable configuration will be advised by supervisor). After repositioning, drives should be switched on and teaching or executing procedure could be continued.
Robots L1 and L2 are connected by digital IO lines in such way that outputs No 4, 5, 6, 7 of one robot are connected to inputs No 4, 5, 6, 7 of another robot and vice versa.
Output 0 in robot L1 is controlling gripper.

Software

The control program L1/L2 allows you to create, edit and run programs of robot control. It is organized into hierarchical menu as shown in Fig. 1.5, and described in Table 1.3.

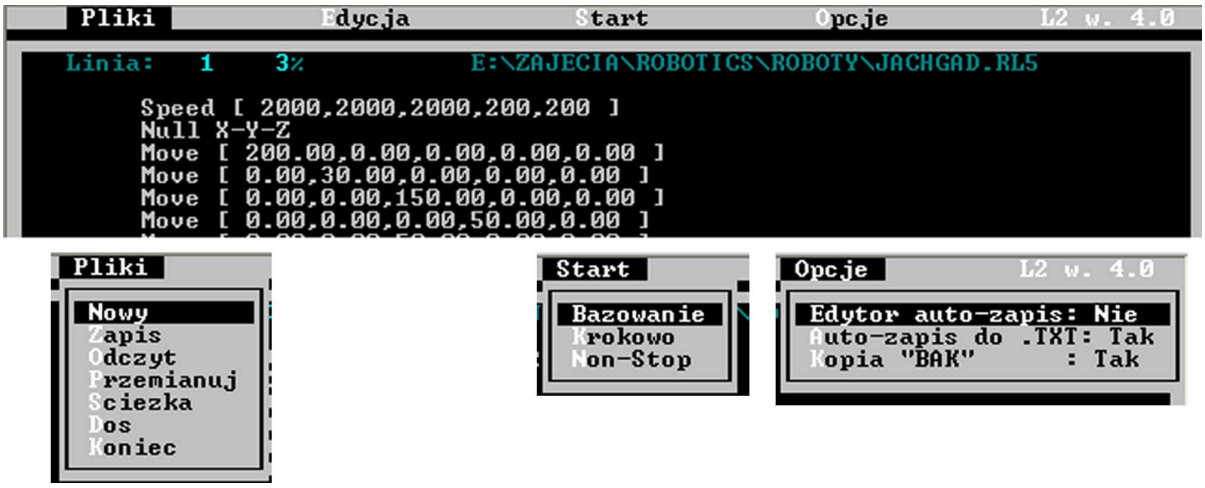

Fig. 1.5 Menu of control program L2

Table 1.3 Functions of the main menu

Menu	Function
Pliki	Disc and file operation
Edycja	Enter the edit window
Start	Base-Test of the robot and user program execution options (step and non-stop).
Opcje	Additional options

If you choose <Edycja> from the main menu you will enter the edit window given in Fig. 1.6.

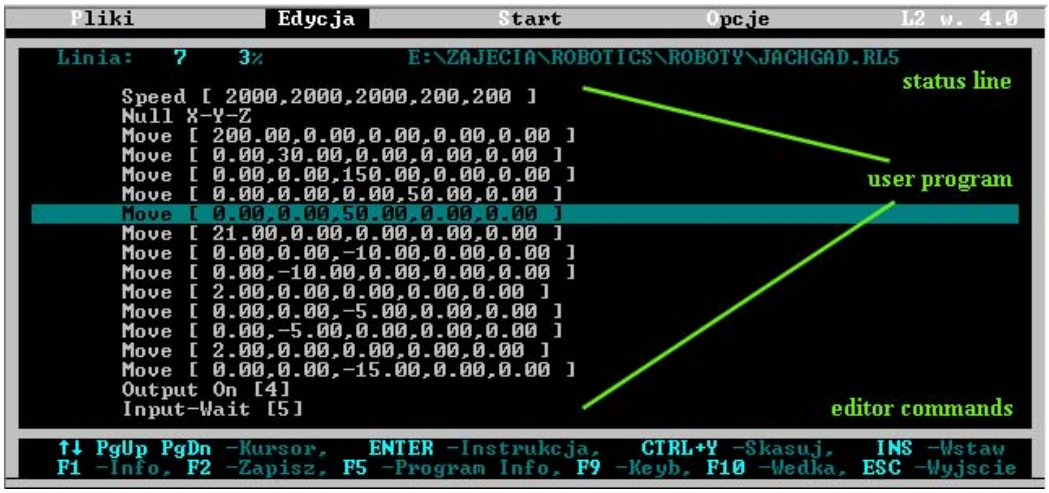

Fig. 1.6 The edit window

In the main part of editor window there is user program currently being edited. Additionally, status line shows the number of highlighted line, size of program, and name of currently edited program. Editor commands can be accessed by keyboard shortcuts:

PgUp, PgDn, ←↑→↓	moving in edit window
Enter	call instruction menu
Ctrl-Y	delete current (highlighted) line
Ins	insert line before
F1	program information
F2	save current program
F5	information of user program
F9	open a teachbox window to manually control the robot
ESC	exit editor

The user control program consists of instructions and parameters inserted from Instruction menu (activated by ENTER button). Instruction menu is shown in Fig. 1.7. The functions are explained in Table 1.4.

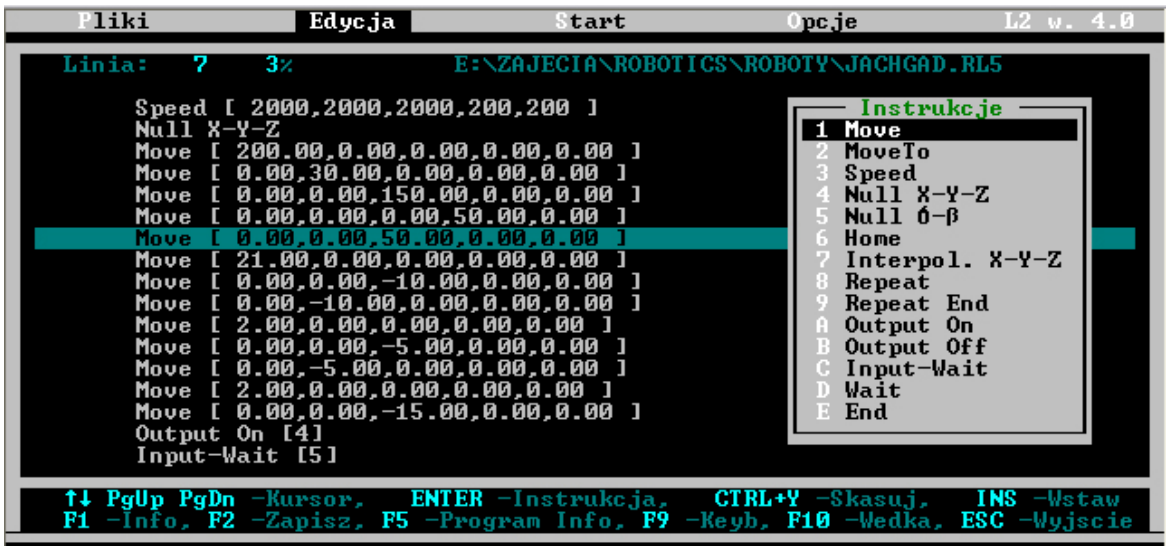

Fig. 1.7 The Instruction menu window

Table 1.4

Mnemonic	Parameters	Function
MOVE	dx, dy, dz [mm or deg] dα, dβ, dγ [deg]	moving robot from current position by a vector [defined in parameters]
MOVE TO	as above	moving robot to position described by a vector given in parameters. Zero coordinates are set by instruction NULL

SPEED	sx, sy, sz, sα, sβ, sγ {21..4000}	set current velocity of joints
NULL XYZ	no parameters	set zero coordinates of axes 1st, 2nd, 3rd
NULL αβγ	no parameters	set zero coordinates of wrist axes
INTERPOL XYZ	i {0,1,2} 0 - XY, 1 - XZ, 2 - YZ	set the couple of axis which are moving first during complex movement of robot
INTERPOL αβγ	i {0,1,2} 0 -αβ, 1 -βγ, 2 -αγ	set the couple of wrist axis which are moving first during complex movement of the robot
REPEAT	n	open the loop with setting loop counter to n
REPEAT END	no parameters	close the loop
OUTPUT ON	nr	set output number nr
OUTPUT OFF	nr	reset output number nr
INPUT-WAIT	nr	stop program until input nr is set
WAIT	t [sec]	stops executing of user program for t seconds
END	no parameters	end of user program

If you choose <F9> in the edit window you will open a teachbox window, shown in Fig. 1.8, to manually control robot. You can use keyboard to move each axis of robot, make a base-test, load current position of the robot to editor, and change speed or interpolation. Specific keys associated with each function are shown in lower-right corner of the sign indicating this function. For instance: key <a> is used to move axis X to negative direction, key <q> to move the same axis X to positive direction, similarly keys <s>, <w> are associated with Y axis, <d>, <e> with Z axis and so on. Pressing <v> increases and <b> decreases speed or robot. Key <p> is for base-test procedure; <enter> adds current position of robot to user program.

To stop the robot you have to press <spacebar> key.

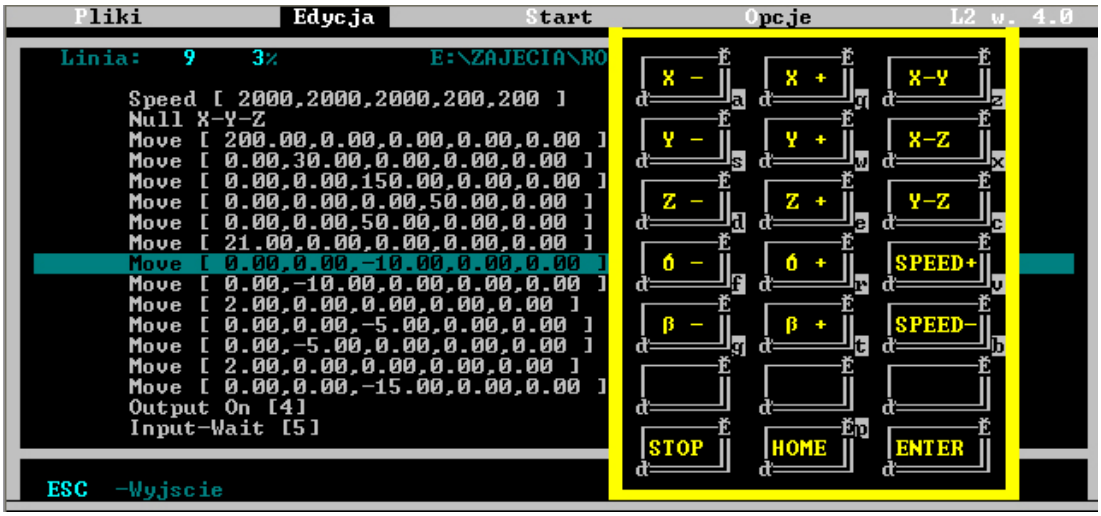

Fig. 1.8 Teachbox window

Exercise agenda

Read carefully users manual, observe operation of the robot, be ready to stop it in case of emergency. If something is not clear – ask. Always check if your user program contains correct data. All malfunctions of hardware or software have to be reported to supervisor.

To begin the exercise:

a) **Switch on the PC.**
b) **Switch on stepper motor controller (under the computer).**
c) **Run L1.exe (or L2.exe) program (in directory C:\ROBOTY\L1.ENG or C:\ROBOTY\L2.ENG).**
d) **Choose option <Edit> from menu and press <F9> to open the teach window. Use keyboard to move the robot into the working space. Do not use arrows on the keyboard, use appropriate letters instead.**
e) **Warning: Before base-testing the robot please make sure that it is inside it's working area and that none of the joint limit switches is activated (this is shown by HOME diode on the front panel of stepper motor controllers). If necessary, reposition the robot.**
f) **Make the base-test – press letter <p> in teach window. All the time observe the robot and be ready to stop it in case of emergency. To immediately stop robot's drives use stop buttons on the front panel of stepper motors driver. In case of local movements of robot L1 (wrist rotations) there is no automatic base-test procedure and therefore operator has to switch these drives off and manually position wrist in the convenient configuration (the best and repeatable configuration will be advised by supervisor).**
g) **Robot is ready for operation: you can use teach window to control the robot – move axes using appropriate keys, save current position in the editor window using <enter> key. You can also use instruction menu to insert correct commands.**
h) **Warning: user program has to begin with <Speed> command, suggested values are as follows: robot L1 – Speed(2000, 2000, 2000, 200, 200, 200), robot L2 – Speed(3000, 2000, 2000, 200, 200). User program is finished with <End> command**
i) **Check if all commands in editor contains correct data**
j) **Warning: before executing user program you have to move robot to the same position the teaching procedure has begun (it is advised that teaching and executing starts from base position)**
k) **You can execute user program in step-mode, when correct test it in non-stop. GRADE 4**
l) **Synchronize tasks of robots L1 and L2 using commands <output on/off> and <input-wait>. Warning: <input-wait> command stops program until appropriate input is being switched to 1. You can use inputs and outputs nr: 4-7.**
m) **Execute program i step – and non-stop modes. GRADE 5**

The task to be performed by robots L1 and L2 is: robot L1 picks up and carries an object into the working area common for both robots. Then L1 puts the object into the L2 container, and finally L2 takes the detail to another box located near its home position.

Exercise C – EASY-ROB 3D Robot Simulation Tool

EASY-ROB is a powerful simulation and planning software for robot cells and to create technical processes in a virtual world. Robot motions are programmed and visualized in a 3D scene immediately, as shown in Fig. 2.1.
Advantages of off-line programming:
• Assistance by the evaluation of new ideas
• Feasibility studies at reasonable price
• Increasing planning security
• Decreasing start-up time

EASY-ROB can be used to simulate production line on condition that a model is calibrated to real robot and environment. This task is difficult and sometimes impossible to perform with sufficient accuracy.

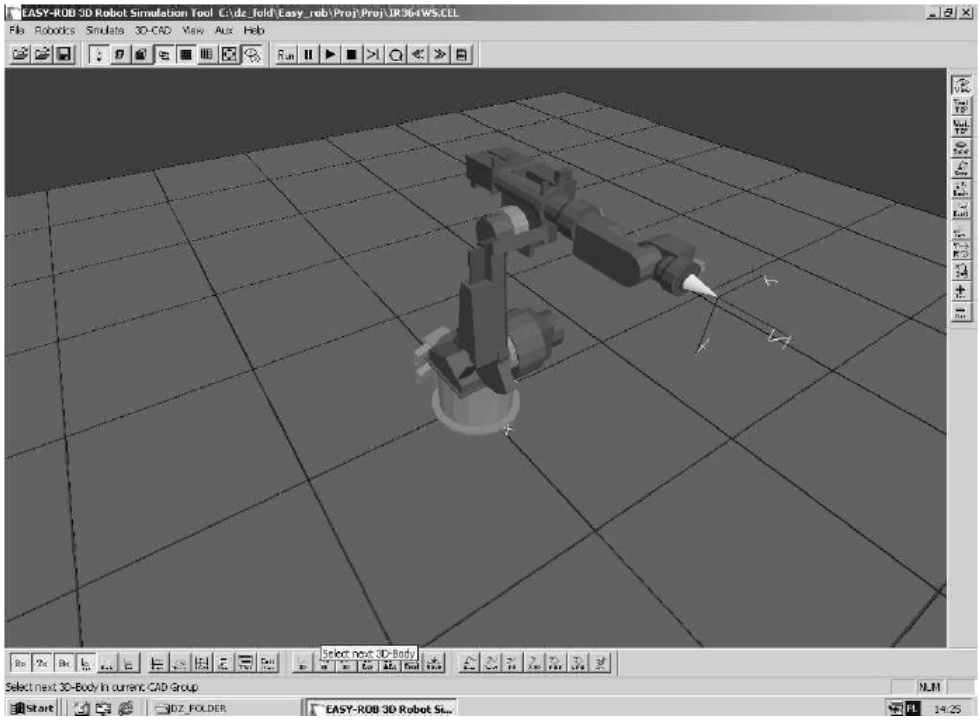

Fig. 2.1 EASY-ROB start window.

Introduction

Each work session is divided into three steps:

1. **Loading a robot,**
2. **Editing a program**
3. **Running a program to monitor and register selected variables**

Description of program Menu

A complex menu provides wide range of flexible tools to implement new functions and models. This part describes icon menu.

Top menu bar

Top menu consists of:

1. File menu

- **Load a cell file (*.cell)**
- **Load a cell, robot, tool, etc. from the library**
- **Save / Save as of the cell file**

2. View menu

- **3D scene displayed without light source**
- **3D-geometries displayed as wire frame**
- **Backface culling**
- **Switch rounding**
- **Switch off visibility of the floor**
- **Display the floor as wire frame**
- **Reset view**
- **Switch between simulation and preview**

3. Simulation menu

- **Run the actual program**
- **Continue / pause program execution**
- **Stop program execution**
- **Switch on/off step simulation**
- **Automatic program repeat**
- **Decrease simulation step size**
- **Increase simulation step size**
- **Opens the first four output windows (desired Joint Values, desired Cart. Pose, current Motion Data, actual Joint Values)**

Bottom menu bar

Bottom menu consists of:

1. Visualization menu

- **Show robot bodies visible or invisible**
- **Show tools visible or invisible**
- **Environment bodies visible or invisible**
- **Toggle robot TCP (*Tool Center Point)* coordinate system**
- **Toggle robot coordsys**
- **Show all tag coordsys**

2. Trajectory menu

- **Toogle dynamics**
- **Show robot TCP trace**
- **Activation /deactivation limit switch**
- **Show program window**
- **Open program teach window**
- **Toogle collision**

3. Position menu

- **Toogle 3D cad coorsys**
- **Select next 3D body**
- **Select previous 3Dbody**
- **Enter position for selected body**
- **Enter relative position for selected body**
- **Reset position for selected body**
- **Save position for selected body**

4. Motion menu

- **Move the robot to his home position**

- **Move the robot to joint position**
- **Move the robot to cartesian position**
- **Move the robot relative to cartesian position**
- **Move circular to cartesian position**
- **Move circular relative to cartesian position**
- **Move the robot along cPath (load set of positions from *.tag)**

Flank menu bar

This menu usually provides more than one or two functions. To activate required functionality, press constantly left<L>/ right<R> mouse button on the selected icon.

1. **Change the scene view**

   <L> Rotate scene

   <R> Zoom

   <L+R> Move scene
2. **Jog robots TCP in tool coordinates**
3. **Jog robots TCP in global coordinates**
4. **Move robot in joint coordinates (double click allow to toggle between axes group 1-3 and 3-6, additional functions: activation /deactivation limit switch, collision detection, motion in TCP or global coordsys)**
5. **Move cRobot base**
6. **Translation <L>/rotation <R> selected 3D body**
7. **Translation <L>/rotation <R> all 3D bodies in current selected group**
8. **Translate or rotate current selected tag point**
9. **Toggle translate/rotate**
10. **Toggle TCP jog On or OFF**
11. **Increases teach step**
12. **Decreases teach step**

Drop-down menu

This chapter describes selected elements from drop-down menu, that are necessary to accomplish simulation which is in the schedule of this exercise.

Sequence of actions during modeling and programming the robot:
1. Loading a robot, Fig. 2.2, Fig. 2.3,
2. Editing a program,
3. Running a program to monitor and register selected variables.

Loading
First task, after launching EASY-ROB, is to load the robot with its environment (*.cell file) and assign the program. These elements can be loaded separately (robot – file *.rob, program – file *.prg etc.)

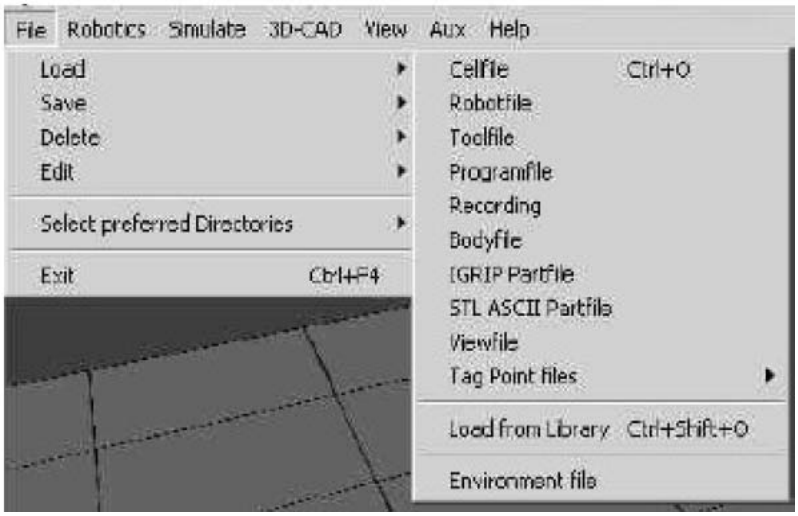

Fig. 2.2 Loading procedure

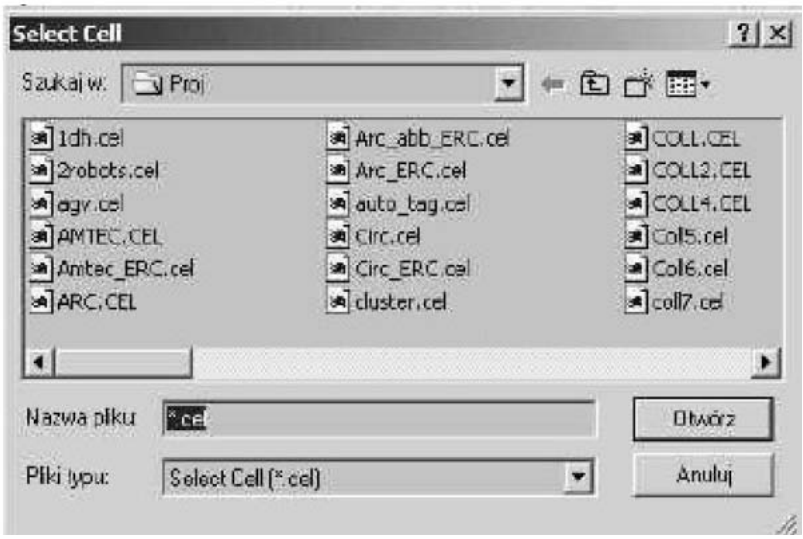

Fig. 2.3 Cell files

After loading cellfile, you have to determine robot structure, working space and location in global coordinate system. Usually, movements are programmed for working tip, more strictly for Tool Center Point (TCP). This value can be identified by rotation or translation in selected axes (select Joint position in flank menu). By default you can use left <L> mouse button for moving column and buttons <P> and <P+L> to get movement in axes 2 and 3, see Fig. 2.4.

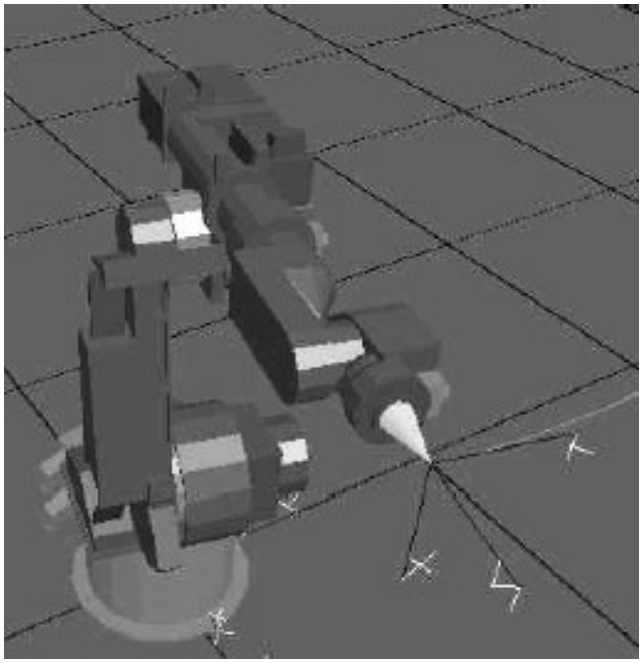

Fig. 2.4 Moving the robot on the scene

Double click opens the jog menu, shown in Fig. 2.5. This menu allows you to switch motion control to axes 4-6 (Fig. 2.6) and set parameters.

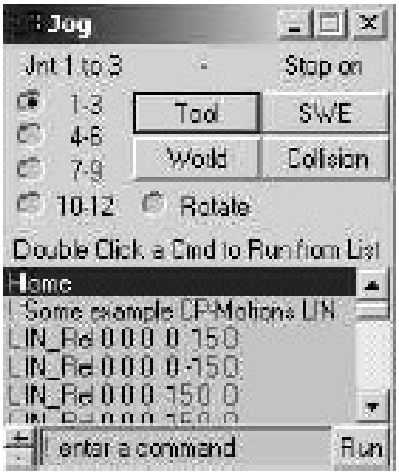

Fig. 2.5 Jog control window

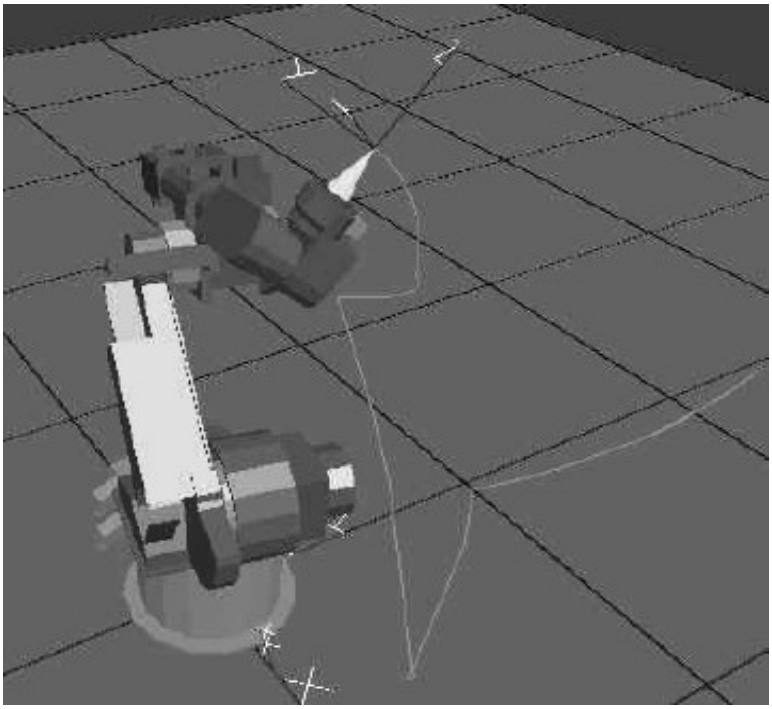

Fig. 2.6 Motion control for axes 4-6

Robot motion can be alternatively controlled by bottom menu and drop-down menu „Simulate\Moveto" (Fig 2.7).

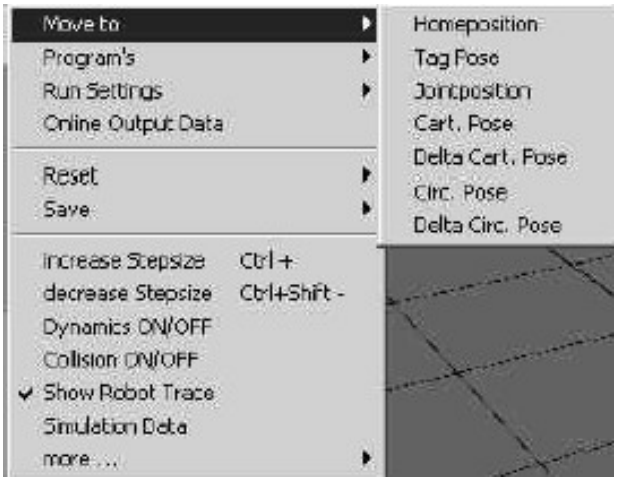

Fig. 2.7 Drop-down menu „Simulate \Moveto"

Motion can be also generated by loading and running program (Robotics\Robot Program\ LoadProgram), as shown in Fig. 2.8.

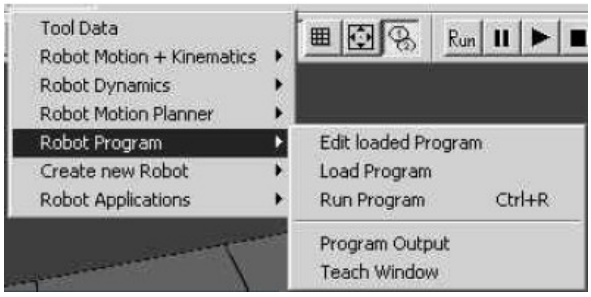

Fig. 2.8 Load program menu

This menu also allows you to call standard text editor in order to edit robot control program. <Edit loaded Program>. When editing program you have to pay attention to the syntax to avoid syntactic errors.

Prepared program can be executed by using top menu or drop-down menu: "\Simulate\Program's\Run Program" (Fig. 2.9).

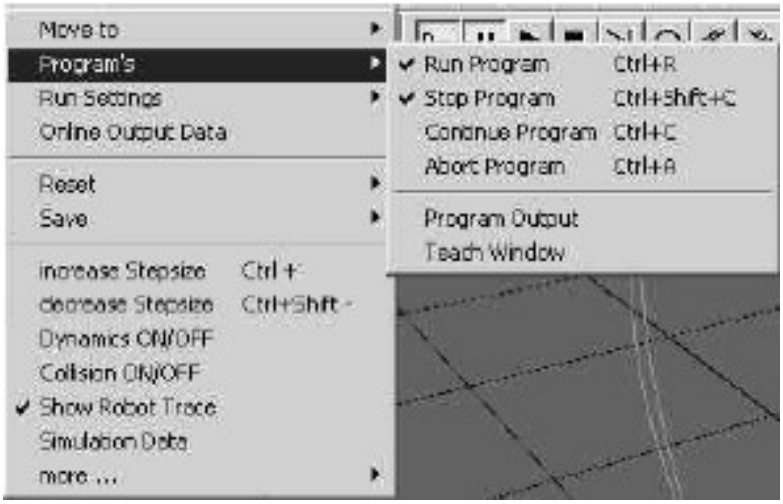

Fig. 2.9 Run program menu

This menu also allows you to open the Teach Window (Fig. 2.10). Teach Window does provide tools to edit program. It is advised to use side buttons to call appropriate functions as that helps to avoid syntax errors.

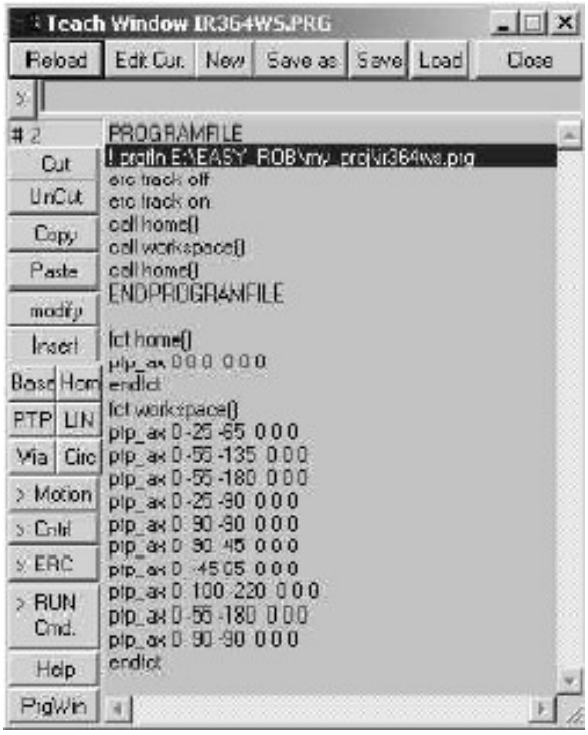

Fig. 2.10 Teach window

Run settings can be accessed from the drop-down menu: Simulate\Run Settings\" (Fig. 2.11) or in top menu bar.

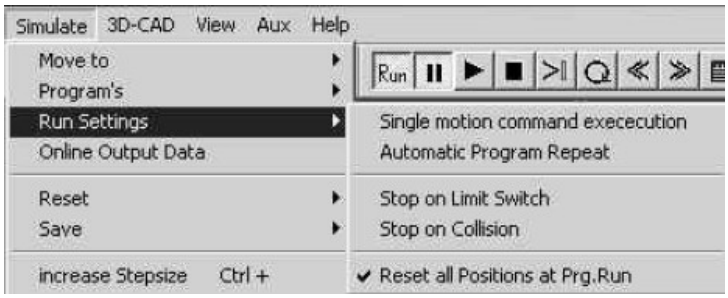

Fig. 2.11 Run settings menu

There are several ways to trace a program execution:

- **in command execution layer: „Simulate\Program's\Program Output" (Fig. 2.12)**

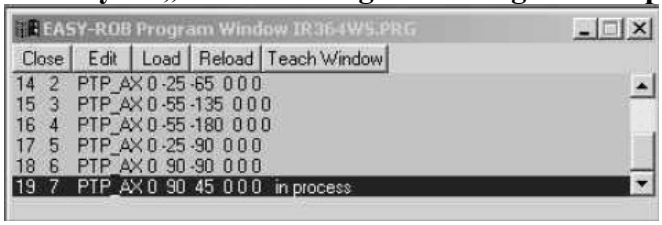

Fig. 2.12 Program window to trace execution

- **in monitoring physical values layer, using IO Output Window: „Simulate\Online output Data". Four selected IO Output windows are shown in Fig. 2.13.**

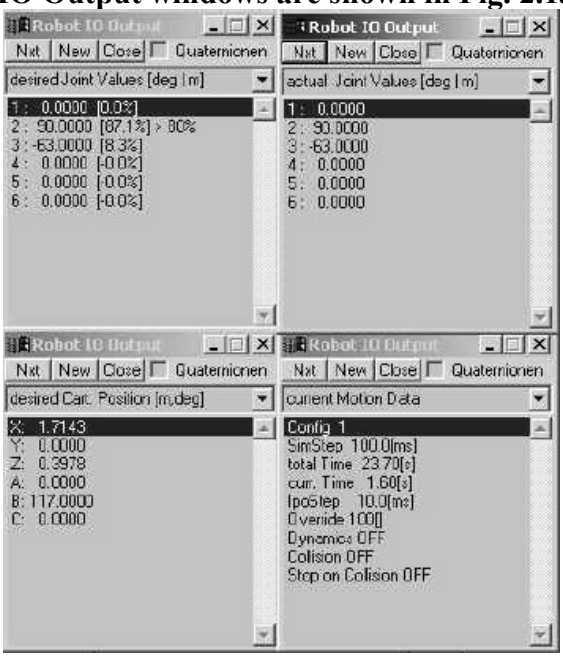

Fig. 2.13 Windows of IO output to trace program execution

- **in a view layer e.g. to observe collisions between robot and environment.**

To adjust the view to your requirements can be achieved by: „View\Set 3D view", e.g. Top View (Fig. 2.14)

Fig. 2.14 Setting the view style

Many useful functions are collected in 3D-CAD menu: e.g. to make robot structure more transparent: „3D-CAD\select group" and, than „3D-CAD\select Body from Group", e.g. to change color ...\Color (Fig. 2.15).

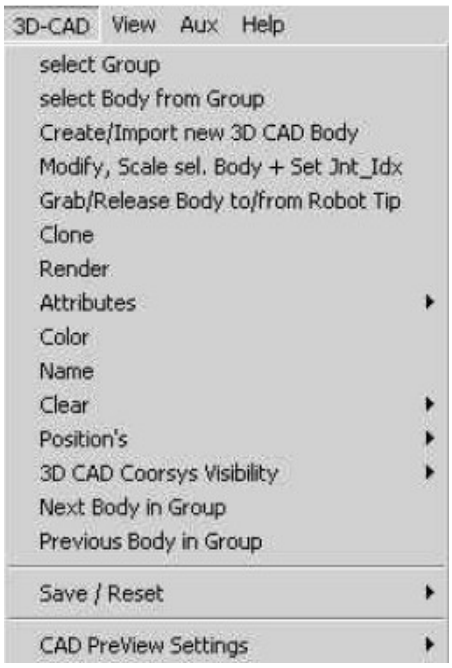

Fig. 2.15 3D-CAD menu

This menu also contains function: „\Grab\Release Body to/from Robot Tip" to grab/release object by robot. Grab function is implemented in mathematical way; it means that object is "inserted" into coordinate system associated with tool. Therefore „grab" can be achieved even without physical contact between tool and object. It means that the programmer is entirely responsible for preparing correct „grab" situation to make it more realistic.

To edit program you can use standard text editor: File\Edit\Loaded Program (Fig. 2.16, Fig. 2.17), but recommended solution is using Teach Window.

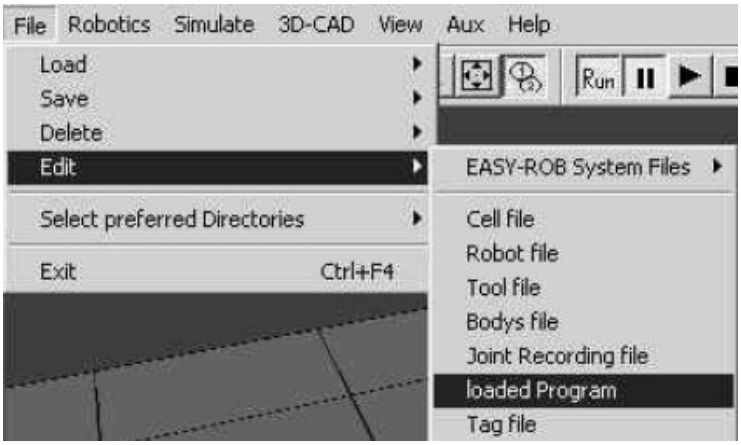

Fig. 2.16 Calling standard text editor

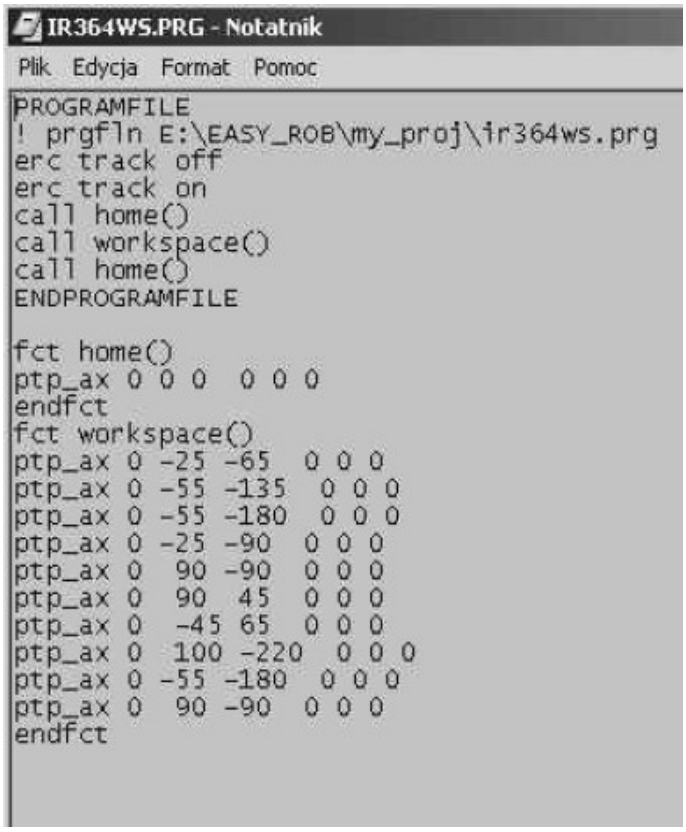

Fig. 2.17 Robot control program in Notepad editor

Easy Robot Programming Language

ERPL – Easy Robot Programming Language – is a collection of basic commands to determine robot motions. The list below gives an overview about the EASY-ROB program structure and available robot motion commands.

Principle syntax definition:

- **Units (length) are meter [m], degree [deg] or percentage [%]**
- **Speed units are in length unit per seconds e.g. [m/s]**
- **A Cartesian pose consists of a position with X, Y and Z values and an orientation with A, B and C angles. The Orientation definition for ABC angles is as follows: Rot (A,B,C) = Rot(X,A) * Rot(Y,B) * Rot(Z,C)**
- **A tagname is for example 'T_1'. To use a tagname with a motion command, the Tag must exist in the workcell.**

General Program Structure

PROGRAMFILE – Begin of Program. Executing this command as single step command will reset some status data, such as BASE, BASE_PRG, etc.

ENDPROGRAMFILE or END – End of Program, Program executing stops

CALL fct_name – Internal Function Call, fct_name() -name of function
CALL FILE filename – External Function Call, filename -name of program file
FCT fct_name() – Begin function definition, fct_name() -name of function
ENDFCT – end of function

Robot Motion Commands
OV_PRO x [%] – Programmable Override x -percentage value
SPEED_CP dx dxe [m/s] – Speed for continuous path motion dx -cartesian speed [dxe] -cart. speed at target
SPEED_PTP v ve [m, deg] – Speed for PTP motion v -joint speed [ve] -joint speed at target
CONFIG n []– Robot Configuration n -config number
TOOL X Y Z A B C [m,deg] – Tool data from tip to TCP XYZ -Position ABC -Orientation TOOL tagname -Tool data from tip to TCP tagname -name of Tag
EXT_TCP X Y Z A B C [m, deg] – Extern TCP XYZ -Position ABC -Orientation
EXT_TCP tagname – Extern TCP tagname -name of Tag
BASE X Y Z A B C [m, deg] – Shift Targets by BASE frame The goal of all BASE commands is to shift program commands. The BASE command is always with respect to the robots base. (see also ERC BASE ...) i.e. All following motion commands are transformed by the current base frame. XYZ -Position ABC -Orientation Note: see also ERC BASE BODY bodyname ERC BASE TCP
BASE tagname – Program BASE tagname -name of Tag
BASE_REL dX dY dZ dA dB dC [m, deg] – Relative Program BASE. Shift the current base frame by the relative location, dXdYdZ -delte Position, dAdBdC -delta Orientation
BASE_PRG X Y Z A B C [m, deg] – The BASE_PRG command operates with respect to the current BASE frame. The final reference for all motion and position commands with respect to the robots base is calculated as: T_base_final = T_base * T_base_prg (T -homogeneous 4x4 matrix) XYZ -Position ABC -Orientation
BASE_PRG_REL dX dY dZ dA dB dC [m, deg] – Relative Program BASE_PRG, dXdYdZ -delte Position, dAdBdC -delta Orientation
HOME n []– Home position n -number of homepos.
PTP X Y Z A B C [m, deg] – Synchro PTP, XYZ – Position, ABC – Orientation
PTP_REL dX dY dZ dA dB dC [m, deg] – Relative Synchro PTP, dXdYdZ -delte Position, dAdBdC -delta Orientation
PTP tagname – Synchro PTP tagname -name of Tag
PTP_AX q1 .. qn [m, deg] – Joint specific Synchro PTP, q1..qn -target Joint/Axis
PTP_AX_REL dq1 .. dqn [m, deg] – Relative joint specific Synchro PTP, dq1..dqn -delta Joint/Axis
LIN X Y Z A B C [m,deg] – Linear CP motion XYZ -Position ABC -Orientation
LIN_REL dX dY dZ dA dB dC [m, deg] – Relative Linear CP motion dXdYdZ -delta Position, dAdBdC -delta Orientation
LIN TagName – Linear CP motion tagname -name of Tag
LIN_ORI ori_type – Orientation Interpolation type for Linear CP motion ori_type -VARIABLE, FIX, TANGENTIAL, AUX, VARIABLE2, QUATERNION

CIRC X Y Z A B C [X2 Y2 Z2] – Circular CP motion [m, deg] XYZ – Position ABC -Orientation [X2 Y2 Z2] -Via Point Pose
CIRC_REL dX dY dZ dA dB dC [dX2 dY2 dZ2] [m, deg] – Relative Circular CP motion, dXdYdZ delte Position, dAdBdC – delta, Orientation [dX2 dY2 dZ2] -delta Via Point Pose
CIRC TagName [TagName2] – Circular CP motion TagName -name of target Tag [TagName2] -name of via point Tag pose
CIRC_ORI ori_type – Orientation Interpolation type for Circular CP motion ori_type - VARIABLE, FIX, TANGENTIAL, AUX, VARIABLE2, QUATERNION
VIA_POS X Y Z A B C[m, deg] – Via Position for Circular CP motion, XYZ -Position ABC - Orientation
VIA_POS_REL dX dY dZ dA dB dC [m, deg] – Relative Via Position for Circular CP motion dXdYdZ -delte Position dAdBdC -delta Orientation
VIA_POS TagName – Via Position for Circular CP motion TagName -name of via point Tag pose
WAIT x [sec] – Wait Statement x -time in seconds

ERCL – EASY- ROB Command Language
ERCL is an extension of ERPL, to automate all user interaction
ERC SET_DEFAULTS – Set default values:Enables the robot, tool and environment bodies. Disbales the robot joint coorsys.
ERC SIM_STEP x [sec] – Set Simulation step size, x -sim. step size
ERC CNTRL_STEP x [sec] – Set Controller sample rate, x -controller sample rate ERC
SYSTEM_STEP x [sec] – Set robot model sample rate, x -modell sample rate
ERC IPO_STEP x [sec] – Set Interpolation sample rate, x -ipo sample rate
ERC IPO_LEAD_TIME x [sec] – Set IPO Lead time, the motion will start after this time. x -ipo lead time
ERC IPO_LAG_TIME x [sec] – Set IPO Lag time, at the end of motion, the robot will rest in that pose for this time before moving to the next target pose. x -ipo lag time

ERCL - ON / OFF Commands
ERC TRACK ON,OFF – Enables / Disables the TCP trace
ERC DYNAMICS ON,OFF – Enables / Disables Dynamics
ERC STOP_SWE ON,OFF – Enables / Disables monitoring of software endswitches
ERC STOP_SPEED ON,OFF – Enables / Disables monitoring of joint/axis speed
ERC STOP_ACCEL ON,OFF – Enables / Disables monitoring of joint/axis acceleration
ERC COLLISION ON,OFF – Enables / Disables monitoring of collision
ERC STOP_COLLISION ON,OFF – Enables / Disables the STOP motion on collision
ERC ROBOTJOINTS ON,OFF; ERC ROBOTPOSITIONS ON,OFF -Enables / Disables the Online Robot IO Output
ERC FLOOR ON,OFF – Enables / Disables the Floor
ERC FLOOR_RENDER ON,OFF – Enables / Disables the flat shaded floor (flat or wire)
ERC EXT_TCP ON,OFF – Enables / Disables the brown colored external TCP Coorsys
ERC ORTHOGRAFIC ON,OFF – Enables / Disables Orthografic view

ERC DISPLAY_ROBOT ON,OFF – Enables / Disables the visualization of Robot
ERC DISPLAY_ROBOT_COORSYS ON,OFF – Enables / Disables the visualization of yellow/green colored Robot Joint/Axis Coorsys
ERC DISPLAY_TOOL ON,OFF – Enables / Disables the visualization of Tool
ERC DISPLAY_BODYS ON,OFF – Enables / Disables the visualization of Bodies /Environment
ERC TCP_COORSYS ON,OFF – Enables / Disables the visualization of blue colored Tool/TCP Coorsys
ERC IPO_COORSYS ON,OFF – Enables / Disables the visualization of red colored IPO Coorsys
ERC BASE_COORSYS ON/OFF – Enables / Disables the visualization of green colored Base Coorsys
ERC CREATE_TARGET_TAGS ON/OFF – Enables / Disables creating Tag at target location
ERC RESET_ALL_POSITIONS_JOINTS ON/OFF – Enables / Disables restting all posiitons and robot joints
ERC NO_DECEL ON/OFF – Enables / Disables the Speed deceleration at target pose
ERC GRAFIC_UPDATE ON/OFF – Enables / Disables the visualization of the Render Scene during program Execution. This command is useful to hide background command.
ERC DISPLAY_TAGS ON/OFF – Enables / Disables the visualization of Tag poses

ERCL - Render Commands
ERC RENDER FLAT – Sets the complete Render Scene FLAT shaded
ERC RENDERWIRE – Sets the complete Render Scene in WIRE frame
ERC RENDER POINT – Sets the complete Render Scene as POINTS
ERC RENDER BBOX ON,OFF – Sets the complete Render Scene as Bbox (bounded boxes)
ERC RENDER group bodyname render -Sets/modifies the render of an unique body group - BODY, ROBOT, TOOL name -name of body render WIRE, FLAT, BBOXWIRE, BBOXFLAT, INVISIBLE, POINT ERC RENDER group render -Sets/modifies the render of all parts in the group group BODY_GRP, ROBOT_GRP, TOOL_GRP; render WIRE, FLAT, BBOXWIRE, BBOXFLAT, INVISIBLE, POINT
ERC COLOR group bodyname color – Sets/modifies the color of an unique body group -BODY, ROBOT, TOOL name -name of body color predef. Color
ERC COLOR group color – Sets/modifies the color of all parts in the group Group BODY_GRP, ROBOT_GRP, TOOL_GRP color predef. Color
ERC COLOR track color – Sets/modifies the color the robotsTCP trace track TRACK, TRACK_DYN, color predef. Color
ERC COLOR TAG color – Sets the predefined TAG color, color predef. color

ERCL - Reset and Save Commands
ERC RESET JOINTPOSITION – Reset the joint/axis position
ERC SAVE JOINTPOSITION – Saves the joint/axis position

ERCL - Load Commands
This feature gives the user the ability to load a file, e.g. a robot tool or a view file during program execution.
ERC LOAD TOOL filename – Loads a Tool file (*.tol)
ERC LOAD VIEW filename – Loads a View file (*.vie)
ERC VIEW – steps n
ERC LOAD ROBOT filename – Loads a Robot file (*.rob)
ERC LOAD BODY filename – Loads a Body file (*.bod)
ERC LOAD TAGS filename – Loads a Tag file (*.tag)
ERC LOAD ENVIRONMENT [filename] – Loads an Environment file (*.env)

ERCL - Move Commands
ERC MOVE BODY bodyname XYZ ABC [m,deg] – Moves a unique part from the BODY group to a new location. Bodyname – name of body, XYZ – Position, ABC -Orientation
ERC MOVE BODY bodyname TagName – Moves a unique part from the BODY group to a new tag point pose. Bodyname – name of body, TagName -name of Tag
ERC MOVE TOOL bodyname XYZ ABC [m,deg] – Moves a unique part from the TOOL group to a new location. Bodyname -name of body, XYZ – Position, ABC – Orientation
ERC MOVE TOOL bodyname TagName – Moves a unique part from the TOOL group to a new tag point pose. Bodyname – name of body TagName -name of Tag
ERC MOVE ROBOT bodyname XYZ ABC [m,deg] – Moves a unique part from the ROBOT group to a new location. Bodyname -name of body XYZ – Position ABC -Orientation
ERC MOVE ROBOT bodyname TagName – Moves a unique part from the ROBOT group to a new tag point pose. Bodyname -name of body TagName -name of Tag
ERC MOVE_REL BODY bodyname dXdYdZ dAdBdC [m,deg] – Relative movement of a unique part from the BODY group. bodyname -name of body, dXdYdZ -delta Position. dAdBdC -delta Orientation
ERC MOVE_REL TOOL bodyname dXdYdZ dAdBdC [m,deg] – Relative movement of a unique part from the TOOL group, bodyname -name of body, dXdYdZ -delta Position, dAdBdC -delta Orientation
ERC MOVE_REL ROBOT bodyname dXdYdZ dAdBdC [m,deg] – Relative movement of a unique part from the ROBOT group. bodyname -name of body, dXdYdZ -delta Position, dAdBdC -delta Orientation
ERC MOVE_REL BODY_GRP bodyname dXdYdZ dAdBdC [m,deg] – Relative movement of the complete BODY goup, with respect to the reference body, bodyname -name of reference body. dXdYdZ – delta Position, dAdBdC -delta Orientation

ERCL - Grab and Release Commands
ERC GRAB BODY 'bodyname' – Grab a body Bodyname -name of body
ERC GRAB BODY_GRP – Grab all parts in the BODY_GRP
ERC RELEASE BODY 'bodyname' – Release a body Bodyname -name of body
ERC RELEASE BODY_GRP – Release all parts in the BODY_GRP

ERCL - View Commands
ERC VIEW steps n – Sets the number of view steps, important when loadinf a (*.vie) file
ERC VIEW hither x – Sets value for the hither plane
ERC VIEW yonder x – Sets value for the yonder Plane
ERC VIEW zoom x – Zoom the render scene by the value x
ERC VIEW zoom_in x – Zoom In the render scene by the value x
ERC VIEW zoom_out x – Zoom Out the render scene by the value x
ERC VIEW tcp_rot_tcp ABC – Rotate the render scene about the current robots TCP with respect to the TCP orientation, ABC -relative Rotations
ERC VIEW tcp_rot_world ABC – Rotate the render scene about the current robots TCP with respect to theWorld frame orientation, ABC -relative Rotations
ERC VIEW world_rot_base ABC – Rotate the render scene about the world coorsys frame with respect to the robot base frame orientation, ABC -relative Rotations
ERC VIEW world_rot_world ABC – Rotate the render scene aboutthe world coorsys frame with respect to the world frame orientation ABC -relative Rotations
ERC LOAD VIEW – filename Loads a View file (*.vie)

Exercise Agenda

All files should be created in Proj\Proj_Usr directory. File names should start with a group name (e.g. RG3_dha.prg). Any modifications of source files are forbidden!!!

During this exercise students are required to be aware of the EASY-ROB functionality and than perform the following tasks:

1. **Load the simplest robot model (1dh.cell);**
2. **Select the top view;**
3. **Determine the workspace for loaded model;**

Use different methods to generate motion and programming:

4. **Create motion along a straight line and back using TeachWindow. Change the configuration and repeat the motion.**
5. **For the same start and end points make PTP move, and compare with the previous.**
6. **Write a program following the square path and then circular path written inside square, as shown in Fig. 2.18.**
7. **Check how limits affect the workspace (Jog Window). GRADE 3**

Load the 6-DOF robot IR364WS.cel and repeat steps from 1 to 5.

8. **Repeat Step 5 for different configurations (1-8)**
9. **Write a control program that for the robot's tool-tip follows a 3D path along Cube's edges, as shown in Fig. 2.19. GRADE 4**

Load the paleta.cel file and repeat steps from 1 to 5.

10. Write a control program that for the robot moves boxes from one storage space to another and sets them into the form of pyramide, as shown in Fig. 2.20. Use GRAB and RELEASE functions. GRADE 5

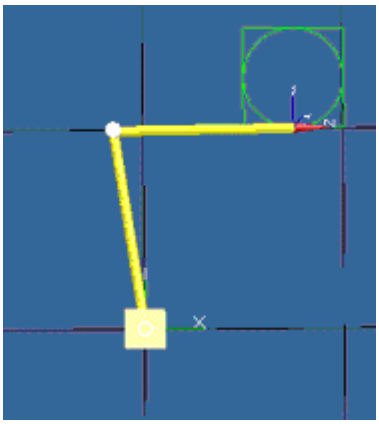

Fig. 2.18 Square and circular path

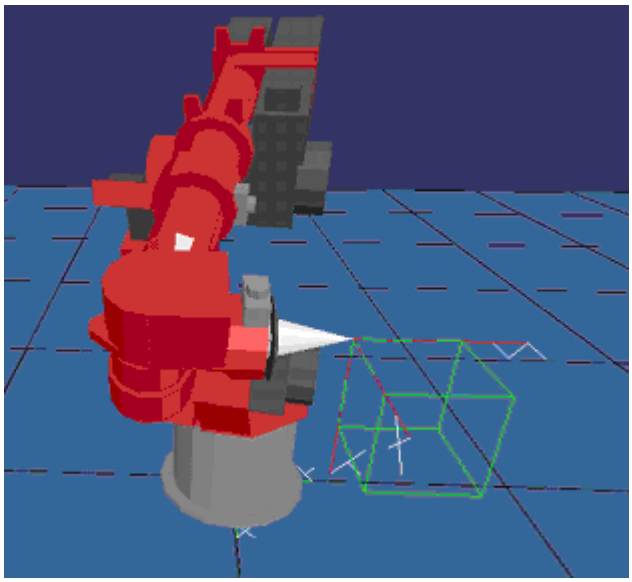

Fig. 2.19 6DOF robot drawing a cube

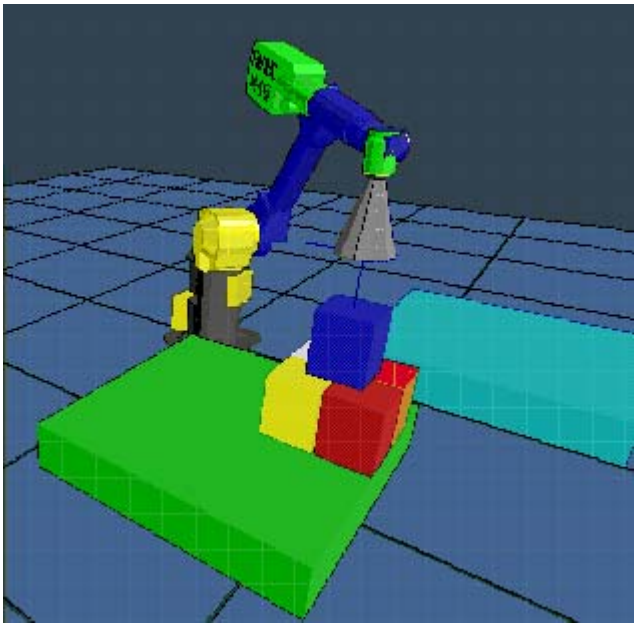

Fig. 2.20 Robot during paletization

Exercise D – Industrial robot IRp-6

This exercise presents a typical industrial robot (IRp-6) equipped with pneumatic gripper and an assembly stand consisting of a rotational table and a belt conveyer, as shown in Fig. 3.1. The task to be performed is to start-up, synchronize, and then to teach and control the robot. The main part of the exercise is to write and execute a control program which will perform an assembly task specified by supervisor.

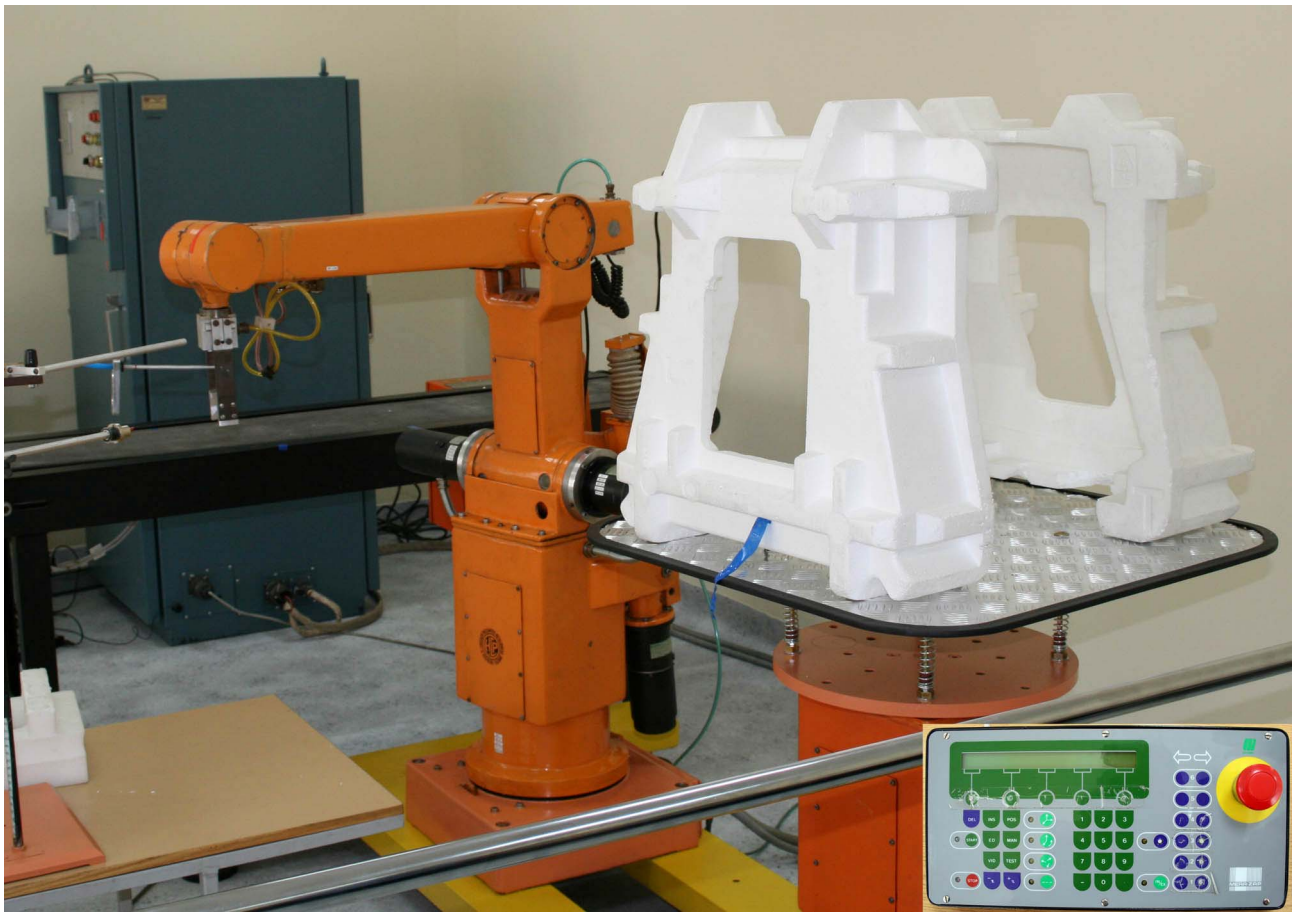

Fig. 3.1 Assembly stand with robot IRp-6

Start up and synchronization procedure

Robot control system contains main control unit (big grey box) and teachbox, shown in corner of Fig. 3.1. The initial operations (start, stop, etc.) have can be executed using the control panel mounted at the front of the control unit. Functionality of this panel is shown in Fig. 3.2.

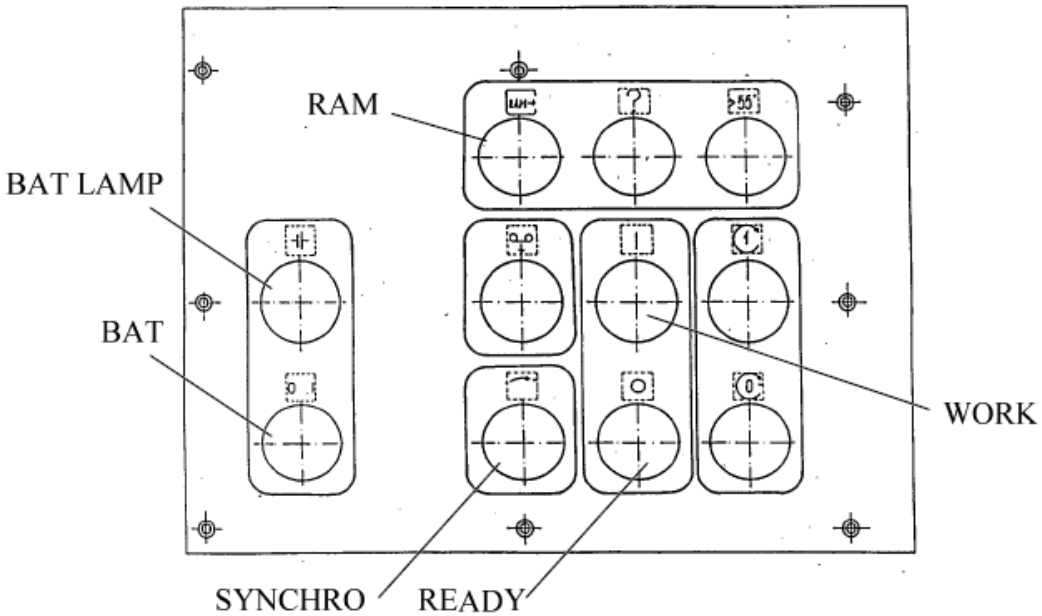

Fig. 3.2 Control panel

Starting the robot:

1. **Turn right the key located under the control panel (two lamps placed next to the key should light on).**
2. **Push the button/lamp READY (at this stage READY and RAM lamps should light on and SYNCHRO lamp should flash).**
3. **After about 5 seconds press READY again (now, one of the lamps located next to the key should switch off).**
3. **Push the button/lamp WORK (this lamp should light on and READY - turn off).**
4. **Using teachbox move the robot to the position described below:**
    - **column turned left about 30 deg,**
    - **lower arm turned forwards about 15 deg,**
    - **upper arm turned down about 30 deg,**
    - **wrist in line with the upper arm,**
    - **tool tip in up position.**
5. **Push flashing button/lamp SYNCHRO (synchronization starts automatically). After the end of the synchronization process the lamps SYNCHRO and RAM should be off. Adequate message should appear on the teachbox.**

Switching off the robot:

1. **If you want to save your program turn BAT switch to the right**
2. **Then turn the key under the control panel to the left (if the previous step has been made then BAT LAMP should be on)**

Teachbox

The robot is equipped with a teachbox shown in Fig. 3.3. It is used both for manual control of the robot as well as for programming. Function of all components located on the teach box is described in 5 groups. Additionally, there is red safety button located in the right part of the teachbox.

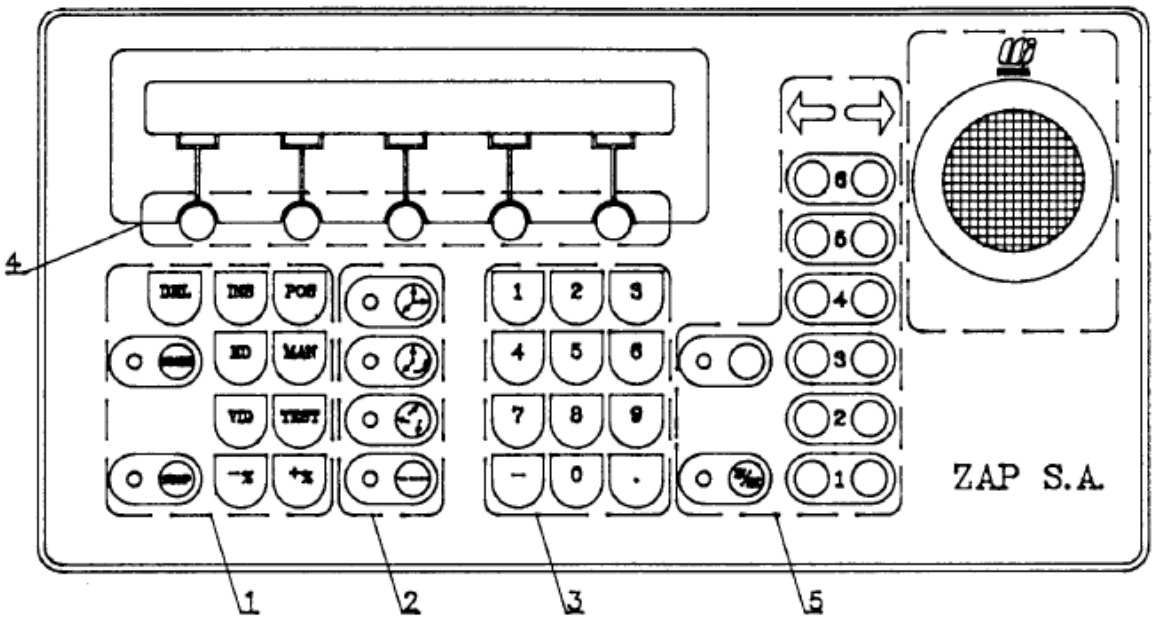

Fig. 3.3 Teachbox

Group 1 Menu keys

DEL	- remove an alphanumeric sign (letter or number) on the LCD display or delete an error message
INS	- used to program instructions other than positioning
POS	- positioning instruction
START	- begin program execution
ED	- program editing
MAN	- manual mode (used for reading, declaring and modifying parameters)
VID	- not used
TEST	- keyboard and display panel test
STOP	- terminates program execution or cassette read/write function -
-%	- a) in automatic program execution mode – velocity decrease - b) in programming mode - move the top text line right
+%	- a) in automatic program execution mode --velocity increase - b) in programming mode -- move the top text line left

Group 2 Movement mode selection keys
These keys are used to select the actual co-ordinate frame used during teaching the robot. The following co-ordinate frames are available:

- **Cartesian frame - the top key of group 2,**
- **cylindrical - the second from the top key within group 2**
- **internal, each joint of the robot is controlled separately - the third key in the same group**
- **The bottom key within this group can be used to reduce robot's velocity.**

Group 3
An alphanumeric keyboard used to define parameters of the robot control program.

Group 4
LCD display and function keys. Specific, interactive menu is shown on LCD.

Group 5 Manual control keys
This group contains six keys, numbered from one to six and located in the right-hand side of the group, and two other keys located in the left part of the group (ACTIVATE - upper one - not marked at the panel, and IN/EXT key). Numbered keys allow you to manually operate each of the robot's joint. Manual operation of the robot is possible only if the manual mode has been activated beforehand. For that purpose the ACTIVATE key has to be pressed and diode on the side of this key should turn on. IN/EXT key is used to determine which set of axis, i.e. internal or external is to be controlled. If external axes are currently controlled a diode is on. Note: the rotation table provided in the laboratory set-up is controlled as the external axis no.1. Consequently, in order to control the table it is necessary to use both ACTIVATE and IN/EXT keys.

Editing user program

There is a built-in editor provided in teachbox which allows many basic operations on program and functions. The following edition steps are available: scroll program by the line forward and backward, jump to first or last instruction, insert specified line number, delete instruction, delete block of instructions, insert new instruction, and change values of function's parameters. This is schematically shown in Fig. 3.4. You can choose editor function by pressing appropriate function keys under LCD display or you can scroll edit menu by pressing keys located under arrow signs.

Program execution
There are three modes of program execution available:
STPOCZ running program from the first instruction,
START begin from the current instruction,
KROK step working mode.
PREDK change velocity on-line during continuous program execution.
It is strongly recommended to use step (KROK) mode for testing new programs. Only fully verified programs should be run using a continuous execution mode.

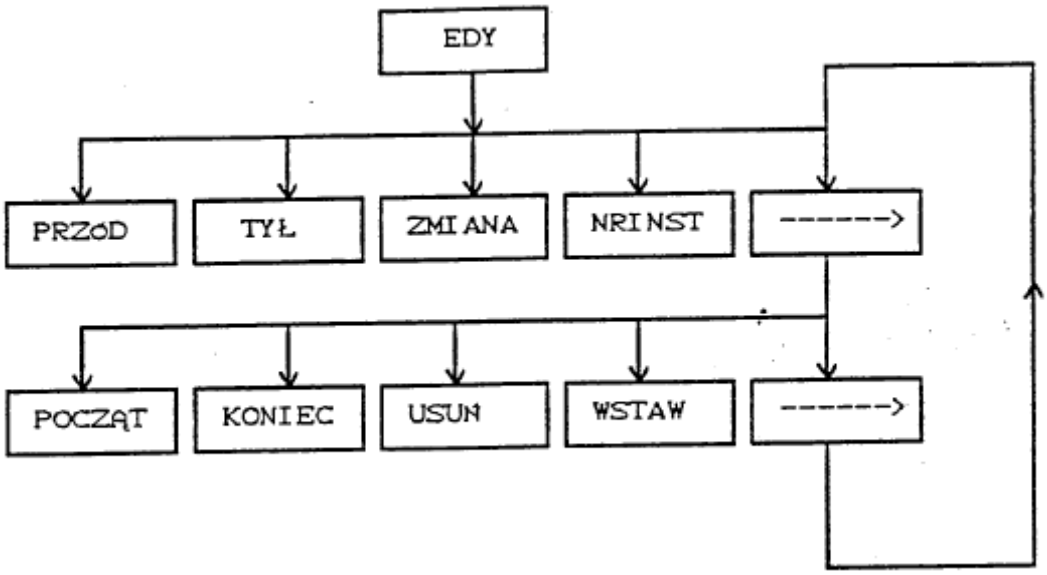

Fig. 3.4 Edit commands

Robot programming

Two groups of instructions are needed to construct any control program. These are positioning instruction menu and additional functions; the structures of these groups are shown in Fig. 3.5 and Fig. 3.6, respectively.

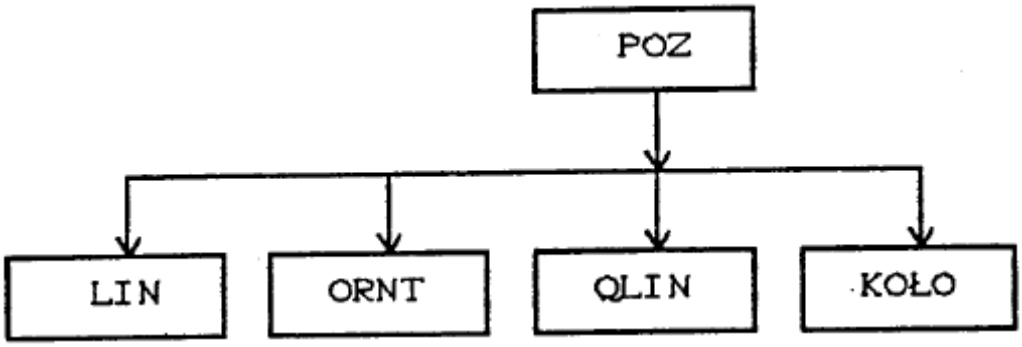

Fig. 3.5 The Positioning commands

The most important instructions are positioning commands. Control system allows you to use four types of positioning:

- **LIN linear transition between two points,**
- **ORNT change tool orientation without changing tool operating point,**
- **QLIN quasi-linear transition with constant joint velocities,**
- **KOLO circle transition via three points.**

In order to execute positioning command it is necessary to set the following parameters:

- **velocity or time angle of rotation (KOLO only)**

- **precision of reaching the destination point**
- **type of reference for co-ordinate determination (relative or absolute)**

Exercise 1, GRADE 3:

Choose two points inside the robot workspace. Try to generate movements between these points using different types of positioning with different settings, compare results. Repeat this exercise with two points having larger distance one form another (at least 2.5m).

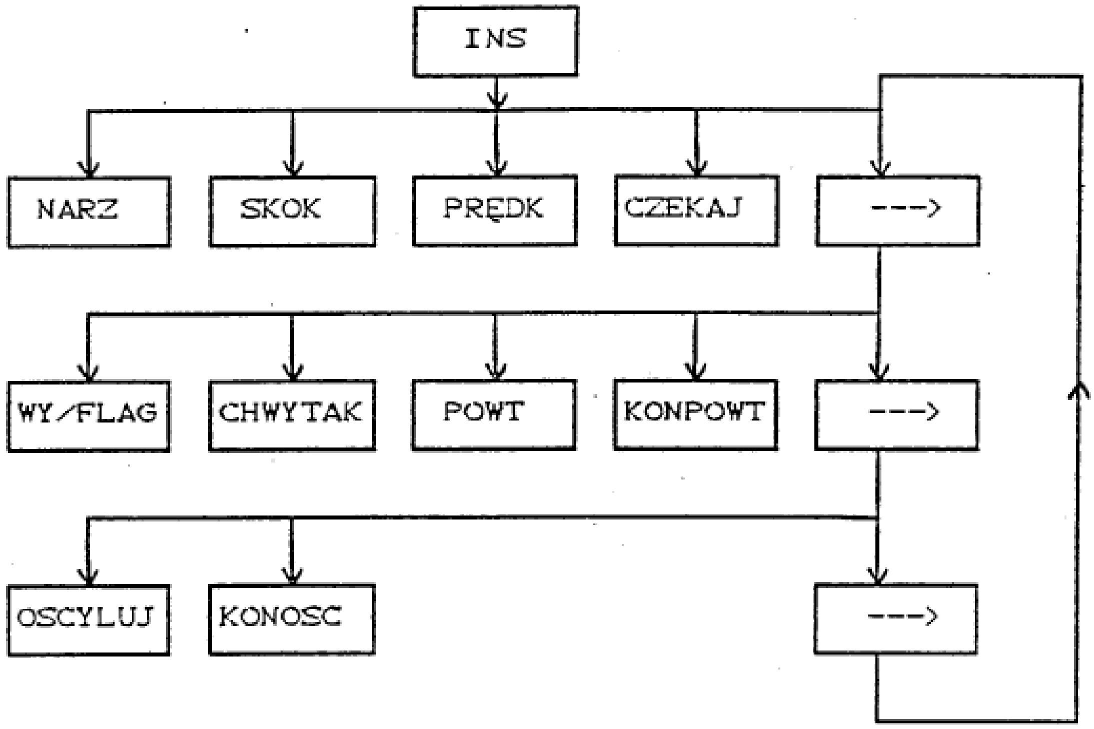

Fig. 3.6 Additional commands

There are 10 additional commands available in the Irp-6 system:

NARZ	tool declaration
SKOK	jump to specified instruction No
PREDK	velocity setting
CZEKAJ	wait
WY/FLAG	output/flag setting
CHWYTAK	gripper operating
POWT	loop organizer
KONPOWT	loop closing
OSCYLUJ	add oscillations to the linear movement
KONOSC	end of oscillations

Before mentioned instructions require some parameters which are set interactively during programming. Every program has to begin with the instruction of tool declaration and velocity setting as shown below:

10 NARZEDZIE 1
20 PREDKOSC = 200 MM/S, PREDKOSC MAX = 500MM/S

Tool number 1 is predefined and can be used without definition, but any other tool requires definition which will be described in the following chapters.

Exercise 2, GRADE 4:

1. **Write a program for the robot to carry a small detail along a square path located in one of Cartesian co-ordinate planes. Velocities on every side of the square should be different. Next, modify the program making the robot draw a triangle 3 times and then go back to drawing the square. To operate gripper use the following commands:**

CHWYTAK 2 ZWOLNIJ **CHWYTAK 1 CHWYC** **CZEKAJ 1 SEK**	**Open a gripper,**
CHWYTAK 2 CHWYC **CHWYTAK 1 ZWOLNIJ** **CZEKAJ 1 SEK**	**Close a gripper,**

2. **Try to use the belt conveyer which is activated as a binary output by the following commands, use time delay (CZEKAJ) to control distance traveled:**

WYJSCIE 1=1 **WYJSCIE 2=0;**	**move right,**
WYJSCIE 1=0 **WYJSCIE 2=1;**	**move left,**
WYJSCIE 1=0 **WYJSCIE 2=0;**	**stop.**

Manual operation instructions

The Irp-6 robot has some additional instructions for low level system controlling and modifying. Scheme of those instructions is shown in Fig. 3.7, and they are described in the following table.

NARZ	tool definition
INKREM	incremental movement of joints
Z PK	loading user program from external memory
DO PK	saving current program from the editor to the external memory
POZWEW	current position of the robot in internal co-ordinates
PAMIEC	available memory

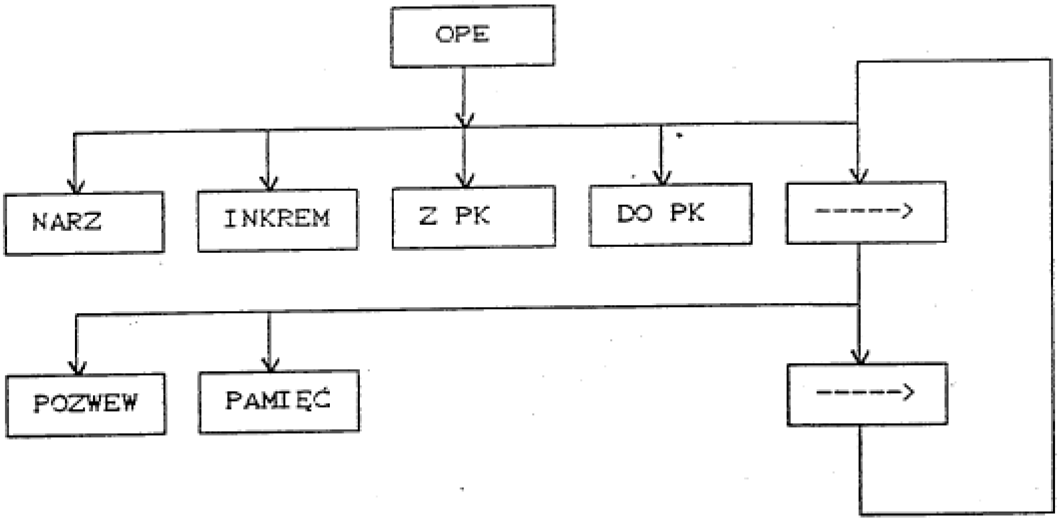

Fig. 3.7 Manual control commands

To define a user specified tool it is required to set position of the working point and orientation of the tool axis. Co-ordinates of the tool should be given in the frame shown in Fig. 3.9 and described with k index (end of forearm). Orientation of the tool axis is described by two angles precession and nutation. These angles are shown in Fig. 3.9 (right-hand part of the figure).

Exercise 3, GRADE 5:

1. **Define new tool No 2 with tool tip at the end of metal pin mounted on the gripper and tool axis along this pin, as shown in Fig. 3.8.**
2. **Write a robot control program which uses new tool and present QLIN positioning function for the toll tip.**

Fig. 3.8 New user tool

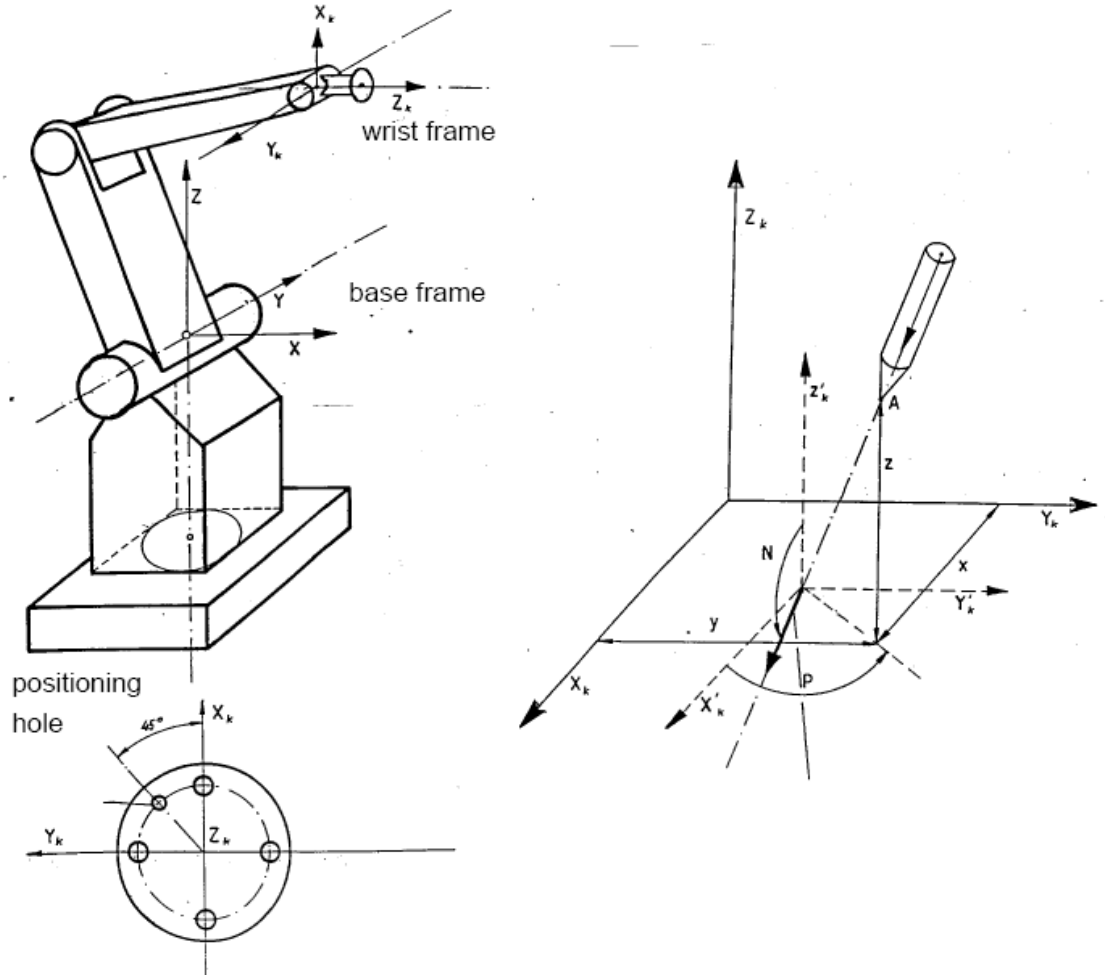

Fig. 3.9 User defined tool

Exercise E – FESTO manipulator

Introduction

This exercise presents a laboratory stand with a pneumatic-electric manipulator and an industrial PLC controller, as shown in Fig. 4.1. The manipulator has a Cartesian kinematic chain with one joint driven by DC motor and two other joint driven by pneumatic actuators. It is additionally equipped with a suction gripper. All facilities are made by Festo Company.

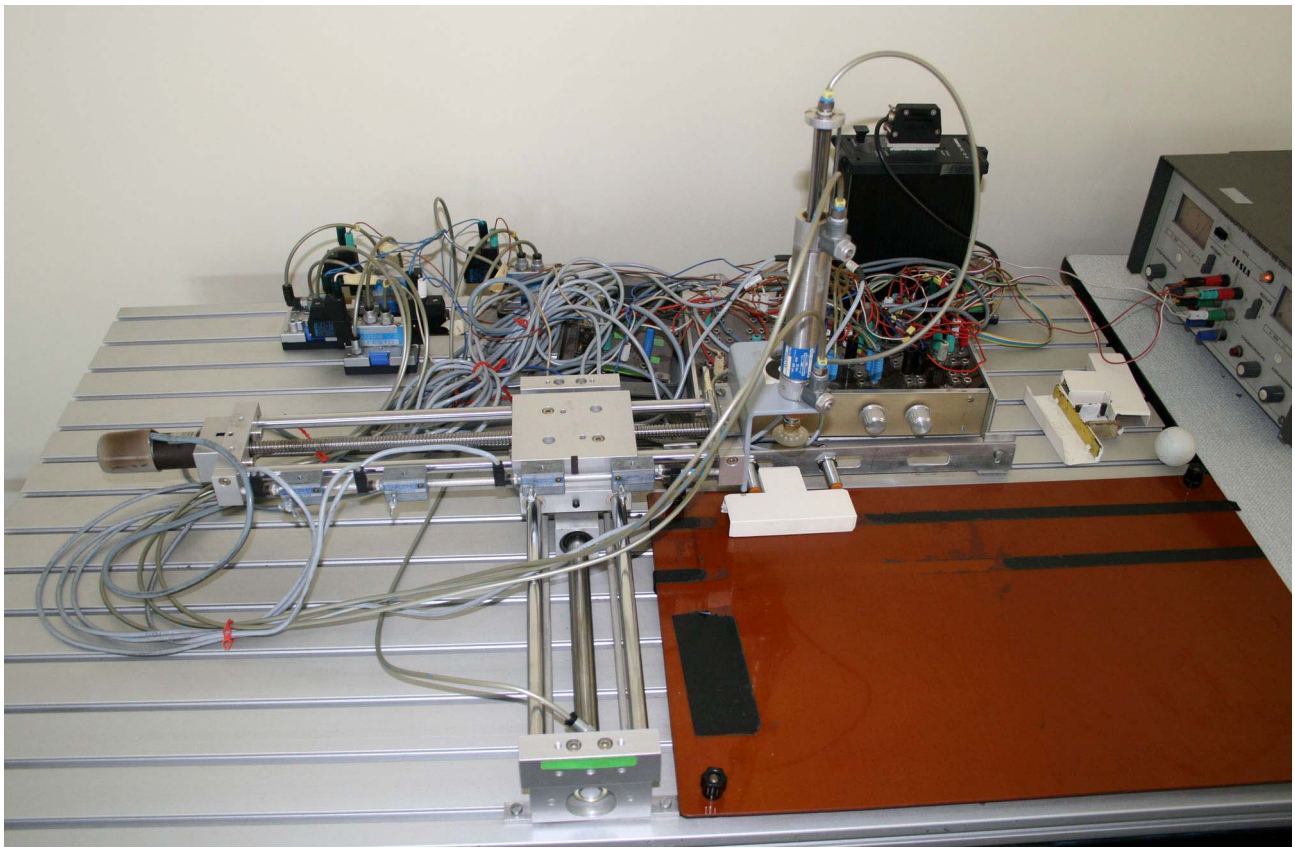

Fig. 4.1 Festo manipulator

Exercise agenda:

1. **Learn AWL language structure and its possibilities, get acquainted with system properties,**
2. **Switch on the computer and run fst101.exe program,**
3. **Switch on the power supplier and pneumatic connection,**
4. **Create/Open your own project in FST101 system,**
5. **Prepare a procedure for delay generation,**
6. **Design program for manipulator control – imitation of the assembly task, segregation of detail depending on orientation, and storage system management,**
7. **Download program to PLC and execute it by pressing red button.**

Festo-INTERPRETER language

Festo-INTERPRETER is a high level language for programming industrial processes. The fundamental element of its syntax is the following conditional sentence:

IF <condition> THEN <execution 1>
OTHRW <execution 2>

In the condition part it is checked if real state of system variables matches the programmed one. If it is true, execution 1 is performed, otherwise execution 2 is done.

Example:
IF i1.0 'if input bit i1.0=1
THEN SET o1.0 'then set output bit o1.0=1
OTHRW RESET o1.0 'other case set bit o1.0=0

Any complex logical function can be inserted into condition part as well as many other orders can be inserted into execution parts. Total number of instructions cannot exceed 100 per one sentence. Syntax of the Festo-INTERPRETER allows to use

- **sentences without condition part:**
  THEN <execution 1>
- **without execution 2 part:**
  IF <condition> THEN <execution 1>
- **full condition sentences like in an example given above,**
- **it is possible to use NOP (no operation) function in any execution part.**

More complex element is STEP. It is a collection of condition sentences beginning with keyword STEP <number>. Number of steps cannot exceed 255. They are executed sequentially. A skip from one step to another depends on a structure of logical inner sentences.

Example:
STEP 1
THEN <exec. 1>

STEP 2
IF <con.>
THEN <exec. 1>

STEP 3
IF <con.>
THEN <exec. 1>
OTHRW <exec. 2>

STEP 4

IF <con.>
THEN <exec. 1>
OTHRW <exec. 2> ***'sentence 4.1***
IF <con.>
THEN <exec. 1>
OTHRW <exec. 2> ***'sentence 4.2***
IF <con.>
THEN <exec. 1> ***'sentence 4.3***

STEP 5

THEN <exec. 1>

Step 1 will be executed unconditionally and all commands in <exec. 1> part will be realized. Step 2 will be executed only if <condition> part is fulfilled (has logic value 1). Step 3 will be realized always but reaching part <exec. 1> or <exec. 2> depending on the condition (if logical value is 1 then program passes through <exec. 1>). All sentences in step 4 will be realized continuously until the condition given in sentence 4.3 is fulfilled (has logic value 1).

Table 4.1 System variables

symbol	number of bits	explaining	possible operation
i<w>.<b>	1	one bit input <w> input word [0..2] <b> bit [0..7]	AND, OR, EXOR, N
iw<n>	8	input word <n> word number [0..2]	LOAD, mathematical relations
o<w>.<b>	1	one bit output <w> output word [0..1] <b> bit [0..7]	AND, OR, EXOR, N, SET, RESET
ow<n>	8	output word <n> word number [0..1]	LOAD, mathematical relations
f<w>.<b>	1	one bit of memory <w> memory word [0..15] <b> bit [0..7]	AND, OR, EXOR, N, SET, RESET
fw<n>	16	memory word <n> word number [0..15]	LOAD, mathematical relations
t<n>	1	timer flag <n> number of timer [0..31]	AND, OR, EXOR, N, SET, RESET
tp<n>	16	timer set value <n> number of timer [0..31]	LOAD
tw<n>	16	timer current value <n> number of timer [0..31]	LOAD, mathematical relations

c<n>	1	counter flag <n> number of counter [0..15]	AND, OR, EXOR, N, SET, RESET
cp<n>	16	counter set value <n> number of timer [0..15]	LOAD
cw<n>	16	counter current value <n> number of timer [0..15]	LOAD, mathematical relations
r<n>	16	register	LOAD, mathematical
		<n> number of register [0..63]	relations
v		direct argument	LOAD

It is possible to use your own characteristic name for each variable. To do this you should edit an allocation table.

Timer/counter operations
There are 32 independent timers and 16 forward/back counters in FPC 101 system. They give possibilities to measure time with resolution 0.01 sec. and to count program repetitions. Both of these elements contain some special registers: set value register for initializing, current value register and flag informing about system state.

Examples:
Measure 1 sec. delay:

```
STEP 10
THEN      LOAD V100      'load the value 100
          TO TP12        'to set value register of 12th timer
          SET T12        'start timer 12
STEP 20
IF        N T12          'if T12=0 timer counted down
THEN
```

Three times repeated loop (counting down)

```
THEN LOAD V3 'load the value 3
        TO CW3 'to current value register of 3rd counter
..... 'body of the loop
THEN DEC CW3  'decrement value
IF N C3 'if counter finished
THEN
```

Three times repeated loop (counting forward):

```
THEN LOAD V3 'load the value 3
            TO CP3 'to set value register of 3rd counter
        LOAD V0 'load the value 0
```

```
                TO CW3 'to current value register
…….'body of loop
THEN INC CW3 'increment value
IF N C3 'if counter finished
THEN
```

User defined procedures

System allows defining up to 8 procedures with up to 16 input arguments each. To define new procedure you have to enter editor with new program settings (program/module - B, program number - 0..7). Every module has independent step numeration and can use up to 16 input parameters stored in local variables FU32 -FU47. For instance the procedure number 0 realizing time delay with length passed by the variable looks like follows:

```
STEP 10
THEN      LOAD FU32      'load first parameter value
          TO TP0         'to set value register of 0 timer
          SET T0         'start timer 0
STEP 20
IF        N T0           'if T0=0 timer counted down
THEN      NOP
```

Calling this procedure for measuring 1 sec. delay should be as follows:

```
THEN      CMP 0 'call user defined module 0
          WITH V100 'with first argument 100
```

FPC 101 system does not allow to call the procedure from the body of another one or itself.

Table 4.2 List of instructions

mnemonic	variable	part of sentence	function
STEP IF, THEN, OTHRW	number		fundamental instructions of AWL language syntax
AND OR N EXOR ( )	i, o, f, t, c, r	condition	logical and logical or logical not logical exclusive or
>, <, >=, <=, =, <>	r, cw, cp, tw, tp, iw, ow, fw, v	condition	mathematical relations
SET RESET	o, f, t, c	execution	set bit reset bit

LOAD TO	r, v, cp, cw, tp, tw, fw, iw, ow	execution	load multibit value to destination variable
+, -, *, /, (,)	r, cw, cp, v, tw, tp, fw, iw, ow	execution	mathematical operations
DEC INC ROR ROL SHR SHL BID DEB SWAP CPL INV	iw, ow, fw, tw, tp, cw, cp, r	execution	decrement increment rotation right rotation left shift right shift left transcoding binary to BCD trancoding BCD to binary byte changing x = not x + 1 inversion
CMP WITH	no of procedure parameters	execution	jump to user procedure [0..7] with parameters [p0..p15]
NOP		condition/exec.	no operating
JMP TO	step no	execution	jump to program step

Festo manipulator system

Laboratory stand contains PC computer, programmable logic controller (PLC), mechanical parts of manipulator, driving system and the valves system. Description of connections is schematically shown in Fig. 4.2 and Table 4.3 below.

Table 4.3

Element	Logical address	Function
DC motor, X axis, P1 relay	**o0.0**	**Motor on/off**
DC motor, X axis, P1 relay	**o0.1**	**Motor direction**
Vertical cylinder, upper valve	**o0.2**	**SET pressurizes chamber**
Vertical cylinder, lower valve	**o0.3**	**RESET exhausts chamber**
Horizontal cylinder, Y axis, valve	**o0.4**	**SET/RESET –go to either end**
Suction cup, valve	**o0.5**	**SET/RESET – suction on/off**
LED diodes (red, green)	**o1.0 – o1.3**	**SET/RESET – on/off**
magnetic sensor, X axis, right	**i1.0**	**could be used to position axis X in different spots in the temporary storage area, when active returns value 1**
mag. sensor, X axis, middle-right	**i1.1**	
mag. sensor, X axis, middle-left	**i1.2**	
magnetic sensor, X axis, left	**i1.3**	
magnetic sensor, Y axis	**i1.4**	**when active returns value 1**
inductive sensor (left, right)	**i0.6, i0.7**	**orientation detection**
buttons (lower, middle, upper)	**i1.5, i1.6, i1.7**	**selection**

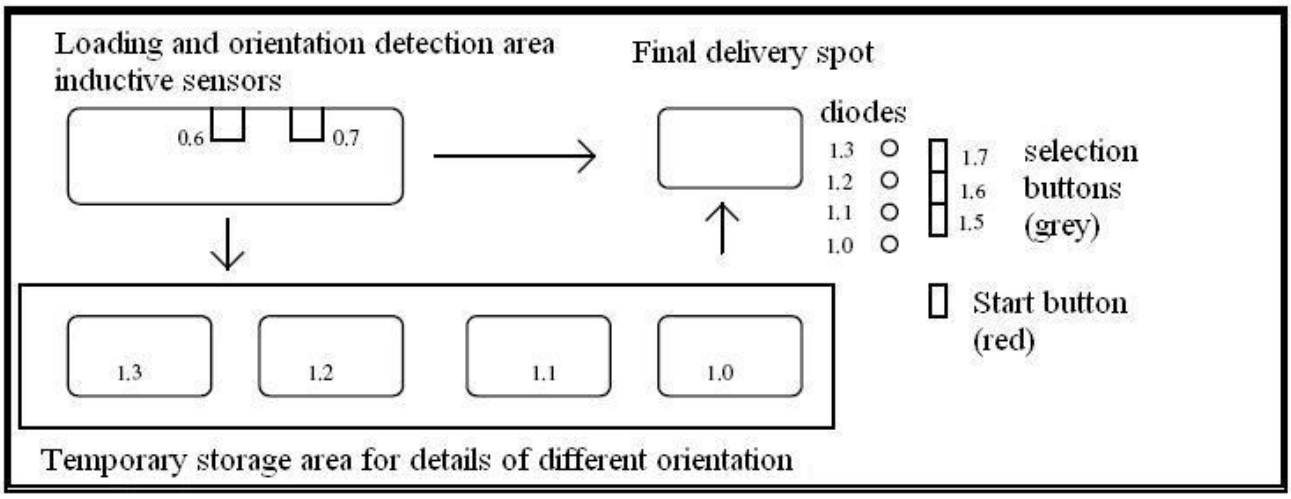

Fig. 4.2 Description of components used in Exercise E

Detailed scheme of DC motor driving system is shown below in Fig. 4.3. Motor is control by two magnetic relays – one determining motor rotation and one setting motor on/off.

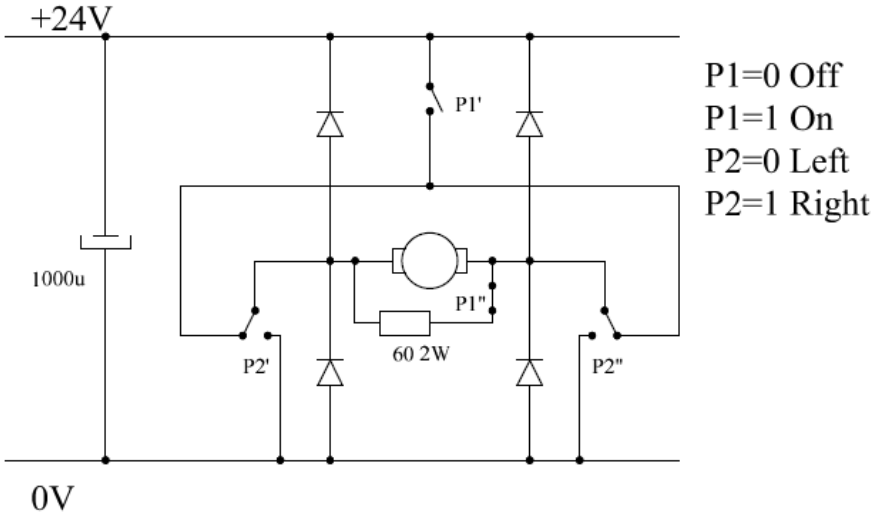

Fig. 4.3 DC motor drive circuit, P1, P2 – relays, P1 – motor on/off, P2 motor direction

Although scenario of the manipulation tasks may be very different, it is suggested that control program contains:

- **automatic detection of the orientation of detail inserted to the loading area,**
- **indication of the choice by switching on one of the green diodes, GRADE 3,**
- **segregation of details with different orientation into different spots in the storage area, GRADE 4,**
- **appropriate reaction on selection buttons – when pressed, detail with specific orientation is delivered to the final delivery spot,**
- **any collision in the algorithm is indicated by flashing red diode, GRADE 5.**