User's Guide

BaseWare OS 2.1

Article number: 3HAC 7677-1
Issue: M2000/Rev.1

ABB Robotics Products AB
S-721 68 Västerås
Sweden

MAIN MENU

WELCOME to the Internal manual

Use the tool field in the Acrobat Reader to manoeuvre through the on-line manual.

The buttons **Go back** and **Go forward** take you step by step through the document, view to view in the order you have seen them.

The buttons **Previous/Next page** move forward or backwards in the document one page at the time. You can also use the buttons **PageUp**/**PageDown** on the key board.

The buttons **First/Last page** move forward or backwards to the last or first page of the document.

To mark text in the document. For use in another document the text can be copied with the command **Copy** in the menu **Edit**.

To mark graphics in the document (in the menu **Tools**). For use in another document the graphics can be copied in the same way as text, see above.

It is also possible to print the manual, use the function **Print....** in the menu **File**.

The cursor, in shape of a hand, changes to a pointing finger when it moves over a linked area. To jump to a linked view, click with the mouse button.

For more information, see the Acrobat Reader on-line manual under the menu **Help**.

Click on the Main menu button to continue to the User's Guide on-line Manual.

Main menu

CONTENTS

Page

Page

Page

Page

INDEX

T

U

V

W

CONTENTS

Page

Introduction

This manual will help you whenever you use the robot. It provides step-by-step instructions on how to perform various tasks, such as how to move the robot manually, how to program, or how to start a program when running production.

1 New Features in this Version of the Robot

New functionality and other interesting information can be read from the file *readme* on the Set-up diskette. This file is continuously updated with the latest information, which is why two robots of the same version may contain different information in their *readme* files.

This file can be read from a normal PC, using any word processing program. It can also be loaded into the robot program memory and then read on the teach pendant. For information on how to load programs from diskette, see *Chapter 8: Programming and Testing*.

2 Other Manuals

Before using the robot for the first time, you should read *Basic Operation*. This will provide you with the basics of operating and programming the robot.
Basic Operation is included in this manual, see *Chapter 4*.

The *Product Manual* describes how to install the robot, as well as maintenance procedures and troubleshooting. This manual also contains a *Product Specification* which provides an overview of the characteristics and performance of the robot.

The *RAPID Reference Manual* contains a detailed explanation of the programming language as well as all *data types, instructions* and *functions*. They are described in alphabetical order for your convenience. If you are programming off-line, the *RAPID Reference Manual* will be particularly useful in this respect.

3 How to Read this Manual

Before you start reading through this manual, it is essential that you read *Chapter 3: Safety*. This tells you what you should or should not do to avoid injuring yourself or someone else.

Chapter 4: Basic Operation is an introduction to the basic operation and programming of the robot. It is recommended to be used as a tutorial, together with a robot or the PC software *QuickTeach*™.

You will find a general description of the robot, such as what happens on start-up or what the teach pendant does and looks like, in *Chapter 5: Starting up*.

Generally speaking, the robot is operated by means of different windows:

- Manual movement, see *Chapter 6: Jogging*.

- Manual operation of inputs and outputs, see *Chapter 7: Inputs and Outputs*.
- Programming and testing, see *Chapter 8: Programming and Testing*.
  The programming language is clearly described in *Chapter 9: The programming language RAPID*.
  For a more detailed description, see *RAPID Reference Manual*.
- Running production, see *Chapter 11: Production Running*.
- Setting system parameters, see *Chapter 12: System Parameters*.
- Copying programs, etc., see *Chapter 13: File Manager*.
- Service tools, see *Chapter 14: Service*.

Calibrating the robot, TCP and other coordinate systems, see *Chapter 10: Calibration.*

Chapter 15: Programming ProcessWare and *Chapter 16: System Parameters ProcessWare* describe how to program and set the system parameters for specific applications like arc welding (ArcWare) and spot welding (SpotWare).

In *Chapter 17: Program Examples*, a number of programs are built up, step by step. Here you can learn a little about how to program, and also see the instructions in their correct context.

If you want to find out what a particular menu command does, you should refer to *Chapter 18: Quick Reference*. This chapter can also be used as a pocket guide when you are working with the robot.

If the robot is delivered or upgraded with some extra functionality this is described in *Chapter 19: Special Functionality in this Robot*.

To make things easier to locate and understand, *Chapter 20* contains an *index* and a *glossary*.

3.1 Typographic conventions

The commands located under any of the five menu keys at the top of the teach pendant display are written in the form of **Menu: Command**. For example, to activate the Print command in the File menu, you choose **File: Print**.

The names on the function keys and in the entry fields are specified in bold italic typeface, e.g. ***Modpos***.

Words belonging to the actual programming language, such as instruction names, are written in italics, e.g. *MoveL*.

Examples of programs are always displayed in the same way as they are output to diskette or a printer. This differs from what is displayed on the teach pendant in the following ways:

- Certain control words that are masked in the teach pendant display are printed, e.g. words indicating the start and end of a routine.
- Data and routine declarations are printed in the formal form, e.g. *VAR num reg1;*.

4 Reader's Comments

You can use the next page to send us your comments about the manual. In this way, you will help us to improve the manual and make it easier for yourself to follow in the future. Thank you kindly for your cooperation.

CONTENTS

Safety

1 General

This information on safety covers functions that have to do with the operation of the industrial robot.

The information does not cover how to design, install and operate a complete system, nor does it cover all peripheral equipment, which can influence the safety of the total system.

To protect personnel, the complete system has to be designed and installed in accordance with the safety requirements set forth in the standards and regulations of the country where the robot is installed.

The users of ABB industrial robots are responsible for ensuring that the applicable safety laws and regulations in the country concerned are observed and that the safety devices necessary to protect people working with the robot system have been designed and installed correctly.

People who work with robots must be familiar with the operation and handling of the industrial robot, described in applicable documents, e.g. Users's Guide and Product Manual.

The diskettes which contain the robot's control programs must not be changed in any way because this could lead to the deactivation of safety functions, such as reduced speed.

1.1 Introduction

Apart from the built-in safety functions, the robot is also supplied with an interface for the connection of external safety devices.

Via this interface, an external safety function can interact with other machines and peripheral equipment. This means that control signals can act on safety signals received from the peripheral equipment as well as from the robot.

In the Product Manual/*Installation*, instructions are provided for connecting safety devices between the robot and the peripheral equipment.

2 Applicable Safety Standards

The robot is designed in accordance with the requirements of ISO10218, Jan. 1992, Industrial Robot Safety. The robot also fulfils the ANSI/RIA 15.06-1992 stipulations.

3 Fire-Extinguishing

Use a CARBON DIOXIDE extinguisher in the event of a fire in the robot (manipulator or controller).

4 Definitions of Safety Functions

Emergency stop – IEC 204-1,10.7

A condition which overrides all other robot controls, removes drive power from robot axis actuators, stops all moving parts and removes power from other dangerous functions controlled by the robot.

Enabling device – ISO 11161, 3.4

A manually operated device which, when continuously activated in one position only, allows hazardous functions but does not initiate them. In any other position, hazardous functions can be stopped safely.

Safety stop – ISO 10218 (EN 775), 6.4.3

When a safety stop circuit is provided, each robot must be delivered with the necessary connections for the safeguards and interlocks associated with this circuit. It is necessary to reset the power to the machine actuators before any robot motion can be initiated. However, if only the power to the machine actuators is reset, this should not suffice to initiate any operation.

Reduced speed – ISO 10218 (EN 775), 3.2.17

A single, selectable velocity provided by the robot supplier which automatically restricts the robot velocity to that specified in order to allow sufficient time for people either to withdraw from the hazardous area or to stop the robot.

Interlock (for safeguarding) – ISO 10218 (EN 775), 3.2.8

A function that interconnects a guard(s) or a device(s) and the robot controller and/or power system of the robot and its associated equipment.

Hold-to-run control – ISO 10218 (EN 775), 3.2.7

A control which only allows movements during its manual actuation and which causes these movements to stop as soon as it is released.

5 Safe Working Procedures

Safe working procedures must be used to prevent injury. No safety device or circuit may be modified, bypassed or changed in any way, at any time.

5.1 Normal operations

All normal operations in automatic mode must be executed from outside the safeguarded space.

6 Programming, Testing and Servicing

The robot is extremely heavy and powerful, even at low speed. When entering into the robot's safeguarded space, the applicable safety regulations of the country concerned must be observed.

Operators must be aware of the fact that the robot can make unexpected movements. A pause (stop) in a pattern of movements may be followed by a movement at high speed. Operators must also be aware of the fact that external signals can affect robot programs in such a way that a certain pattern of movement changes without warning.

If work must be carried out within the robot's work envelope, the following points must be observed:

- The key-operated switch on the controller must be in the manual mode position to render the enabling device operative and to block operation from a computer link or remote control panel.
- The robot's speed is limited to max. 250 mm/s (10 inches/s) when the key-operated switch is in position < 250 mm/s. This should be the normal position when entering the working space. The position 100% – full speed – may only be used by trained personnel who are aware of the risks that this entails.

Do not change "Transm gear ratio" or other kinematic parameters from the teach pendant or a PC. This will affect the safety function *Reduced speed 250 mm/s*.

- During programming and testing, the enabling device must be released as soon as there is no need for the robot to move.

The enabling device must never be rendered inoperative in any way.

- The programmer must always take the teach pendant with him/her when entering through the safety gate to the robot's working space so that no-one else can take over control of the robot without his/her knowledge.

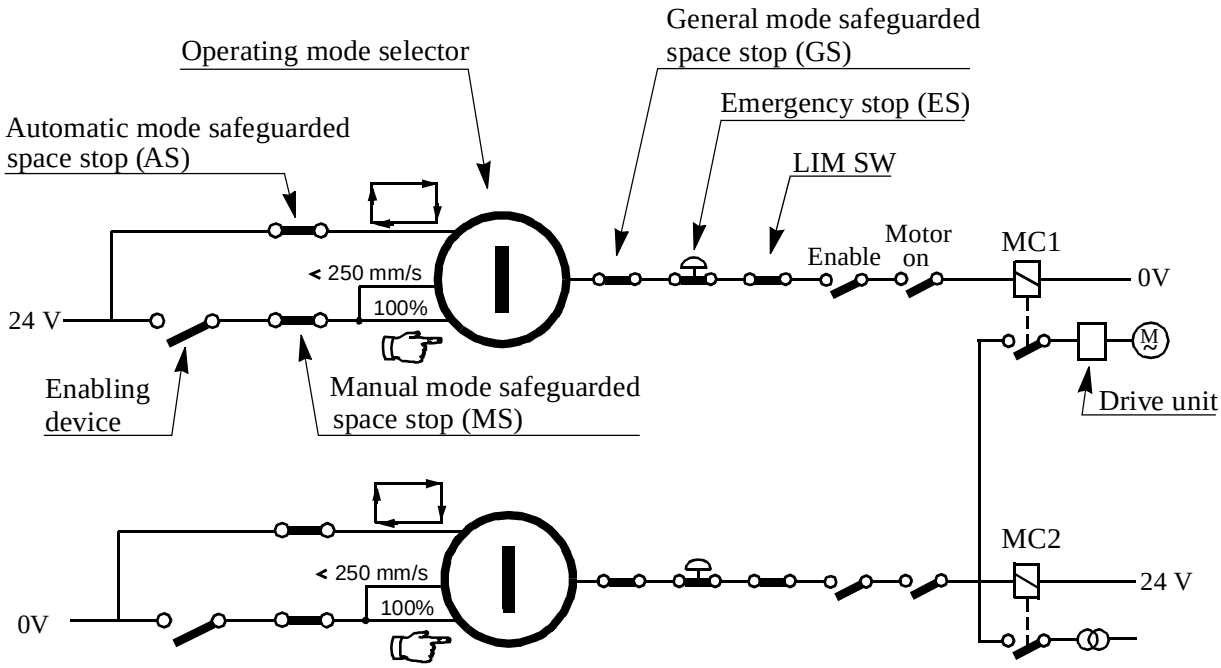
7 Safety Functions

7.1 The safety control chain of operation

The safety control chain of operation is based on dual electrical safety chains which interact with the robot computer and enable the MOTORS ON mode.

The electrical safety chains consist of several switches connected in series, in such a way that all of them must be closed before the robot can be set to MOTORS ON mode. MOTORS ON mode means that drive power is supplied to the motors.

The electrical safety chains are continuously monitored and the robot reverts to the MOTORS OFF mode when a fault is detected by the computer. MOTORS OFF mode means that drive power is removed from the robot's motors and the brakes are applied.



The positions of the switches are indicated by the LEDs on the front of the system board in the control cabinet.

If any contact in the safety chain of operation is open, the robot always reverts to MOTORS OFF mode.

After a stop, the switch must be reset at the unit which caused the stop before the robot can be ordered to start again.

The safety chains must never be bypassed, modified or changed in any other way.

7.2 Emergency stops

An emergency stop should be activated if there is a danger to people or equipment. Built-in emergency stop buttons are located on the operator's panel of the robot controller and on the teach pendant.

External emergency stop devices (buttons, etc.) can be connected to the safety chain by the user (see Product Manual/*Installation*). They must be connected in accordance with the applicable standards for emergency stop circuits.

Before commissioning the robot, all emergency stop buttons or other safety equipment must be checked by the user to ensure their proper operation.

Before switching to MOTORS ON mode again, establish the reason for the stop and rectify the fault.

7.3 Mode selection using the key-switch

The applicable safety requirements for using robots, laid down in accordance with ISO/DIS 10218, are characterised by different modes, selected by means of control devices and with clear-cut positions.

One automatic and two manual modes are available:

Manual mode:
< 250 mm/s - max. speed is 250mm/s
100% - full speed

Automatic mode: The robot can be operated via a remote control device

The manual mode, < 250 mm/s or 100%, must be selected whenever anyone enters the robot's safeguarded space. The robot must be operated using the teach pendant and, if 100% is selected, using Hold-to-run control.

In automatic mode, the key-switch is switched to  , and all safety arrangements, such as doors, gates, light curtains, light beams and sensitive mats, etc., are active. No-one may enter the robot's safeguarded space. All controls, such as emergency stops, the control panel and control cabinet, must be easily accessible from outside the safeguarded space.

Programming and testing at reduced speed

Robot movements at reduced speed can be carried out as follows:

- Set the operating mode selector to >250 mm/s
- Programs can only be started using the teach pendant with the enabling device activated.

The automatic mode safeguarded space stop (AS) function is not active in this mode.

Testing at full speed

Robot movements at programmed speed can be carried out as follows:

- Set the operating mode selector to 100%
- Programs can only be started using the teach pendant with the enabling device activated.

For “Hold-to-run control”, the program start key must be activated. Releasing the key stops program execution.

The 100% mode may only be used by trained personnel. The applicable laws and regulations of the countries where the robot is used must always be observed.

Automatic operation

Automatic operation may start when the following conditions are fulfilled:

- The key-switch is set to 
- The MOTORS ON mode is selected

Either the teach pendant can be used to start the program or a connected remote control device. These functions should be wired and interlocked in accordance with the applicable safety instructions and the operator must always be outside the safeguarded space.

7.4 Enabling device

When the operating mode selector is in the MANUAL or MANUAL FULL SPEED position, the robot can be set to the MOTORS ON mode by depressing the enabling device on the teach pendant.

Should the robot revert to the MOTORS OFF mode for any reason while the enabling device is depressed, the latter must be released before the robot can be returned to the MOTORS ON mode again. This is a safety function designed to prevent the enabling device from being rendered inactive.

When the enabling device is released, the drive power to the motors is switched off, the brakes are applied and the robot reverts to the MOTORS OFF mode.

If the enabling device is reactivated, the robot changes to the MOTORS ON mode.

7.5 Hold-to-run control

This function is active when the operating mode selector is in the MANUAL FULL SPEED position.

When Hold-to-run control is active, the enabling device and the start key on the teach pendant must be depressed in order to execute a program. When the key is released, the axis (axes) movements stop and the robot remains in the MOTORS ON mode. When the key is pressed in again, program execution continues.

7.6 General Mode Safeguarded Stop (GS) connection

The GS connection is provided for interlocking external safety devices, such as light curtains, light beams or sensitive mats. The GS is active regardless of the position of the operating mode selector.

When this connection is open the robot changes to the MOTORS OFF mode. To reset to MOTORS ON mode, the device that initiated the safety stop must be interlocked in accordance with applicable safety regulations. This is not normally done by resetting the device itself.

7.7 Automatic Mode Safeguarded Stop (AS) connection

The AS connection is provided for interlocking external safety devices, such as light curtains, light beams or sensitive mats used externally by the system builder. The AS is especially intended for use in automatic mode, during normal program execution.

The AS is disconnected when the operating mode selector is in the MANUAL or MANUAL FULL SPEED position.

7.8 Manual Mode Safeguarded Stop (MS) Connection

The MS connection is provided for interlocking external safety devices, such as light curtains, light beams or sensitive mats used externally by the system builder. The MS is especially intended for use with additional enabling devices.

7.9 Limiting the working space

For certain applications, movement about the robot's main axes must be limited in order to create a sufficiently large safety zone. This will reduce the risk of damage to the robot if it collides with external safety arrangements, such as barriers, etc.

Movement about axes 1, 2 and 3 can be limited with adjustable mechanical stops or by means of electrical limit switches. If the working space is limited by means of stops or switches, the corresponding software limitation parameters must also be changed. If necessary, movement about the three wrist axes can also be limited by the computer software. Limitation of movement about the axes must be carried out by the user.

7.10 Supplementary functions

Functions via specific digital inputs:

• A stop can be activated via a connection with a digital input. Digital inputs can be used to stop programs if, for example, a fault occurs in the peripheral equipment.

Functions via specific digital outputs:

• ERROR – indicates a fault in the robot system.

• CYCLE ON – indicates that the robot is executing a program.

• MOTORS ON – indicates that the robot is in MOTORS ON mode.

8 Safety Risks Related to End Effectors

8.1 Gripper

If a gripper is used to hold a workpiece, inadvertent loosening of the workpiece must be prevented.

8.2 Tools/workpieces

It must be possible to turn off tools, such as milling cutters, etc., safely. Make sure that guards remain closed until the cutters stop rotating.

Grippers must be designed so that they retain workpieces in the event of a power failure or a disturbance of the controller. It should be possible to release parts by manual operation (valves).

8.3 Pneumatic/hydraulic systems

Special safety regulations apply to pneumatic and hydraulic systems.

Residual energy may be present in these systems so, after shutdown, particular care must be taken.

The pressure in pneumatic and hydraulic systems must be released before starting to repair them. Gravity may cause any parts or objects held by these systems to drop. Dump valves should be used in case of emergency. Shot bolts should be used to prevent tools, etc., from falling due to gravity.

9 Risks during Operation Disturbances

If the working process is interrupted, extra care must be taken due to risks other than those associated with regular operation. Such an interruption may have to be rectified manually.

Remedial action must only ever be carried out by trained personnel who are familiar with the entire installation as well as the special risks associated with its different parts.

The industrial robot is a flexible tool which can be used in many different industrial applications. All work must be carried out professionally and in accordance with applicable safety regulations. Care must be taken at all times.

10 Risks during Installation and Service

To prevent injuries and damage during the installation of the robot system, the regulations applicable in the country concerned and the instructions of ABB Robotics must be complied with. Special attention must be paid to the following points:

- The supplier of the complete system must ensure that all circuits used in the safety function are interlocked in accordance with the applicable standards for that function.
- The instructions in the Product Manual/*Installation* must always be followed.
- The mains supply to the robot must be connected in such a way that it can be turned off outside the robot's working space.
- The supplier of the complete system must ensure that all circuits used in the emergency stop function are interlocked in a safe manner, in accordance with the applicable standards for the emergency stop function.
- Emergency stop buttons must be positioned in easily accessible places so that the robot can be stopped quickly.
- Safety zones, which have to be crossed before admittance, must be set up in front of the robot's working space. Light beams or sensitive mats are suitable devices.
- Turntables or the like should be used to keep the operator away from the robot's working space.
- Those in charge of operations must make sure that safety instructions are available for the installation in question.
- Those who install the robot must have the appropriate training for the robot system in question and in any safety matters associated with it.

Although troubleshooting may, on occasion, have to be carried out while the power supply is turned on, the robot must be turned off (by setting the mains switch to OFF) when repairing faults, disconnecting electric leads and disconnecting or connecting units.

Even if the power supply for the robot is turned off, you can still injure yourself.

- The axes are affected by the force of gravity when the brakes are released. In addition to the risk of being hit by moving robot parts, you run the risk of being crushed by the tie rod.
- Energy, stored in the robot for the purpose of counterbalancing certain axes, may be released if the robot, or parts thereof, is dismantled.
- When dismantling/assembling mechanical units, watch out for falling objects.

11 Risks Associated with Live Electric Parts

Controller

A danger of high voltage is associated with the following parts:

- The mains supply/mains switch
- The power unit
- The power supply unit for the computer system (220 V AC)
- The rectifier unit (240 V AC and 340 V DC. NB: Capacitors!)
- The drive unit (340 V DC)
- The service outlets (110/220 VAC)
- The power supply unit for tools, or special power supply units for the machining process
- The external voltage connected to the control cabinet remains live even when the robot is disconnected from the mains.
- Additional connections

Manipulator

A danger of high voltage is associated with the manipulator in:

- The power supply for the motors (up to 340 V DC)
- The user connections for tools or other parts of the installation (see *Installation*, max. 220 V AC)

Tools, material handling devices, etc.

Tools, material handling devices, etc., may be live even if the robot system is in the OFF position. Power supply cables which are in motion during the working process may be damaged.

12 Limitation of Liability

The above information regarding safety must not be construed as a warranty by ABB Robotics that the industrial robot will not cause injury or damage even if all safety instructions have been complied with.

13 Related Information

	Described in:
Installation of safety devices	Product Manual - *Installation and Commissioning*
Changing robot modes	User’s Guide - *Starting up*
Limiting the working space	Product Manual - *Installation and Commissioning*

CONTENTS

Page

CONTENTS

Page

1 Introduction

This manual explains the basics of handling and operating an ABB robot. You do not need any previous experience of robots to understand its contents.

The manual is divided into chapters, each of which describes a particular work task and how to go about performing it. The chapters complement one another and should, therefore, be read in the order they appear in the book.

It is an advantage if you have access to a robot (or the PC-program Quick Teach) when you use this manual, but just reading it should help you understand the basic operation of a robot.

The manual is written to suit a standard installation. Differences can therefore occur, depending on the configuration of the system.

There are two versions of the controller: a small one and a large one. The large version is shown in this manual. The cabinet of the small controller has the same operator's panel as the large one, but is located in another position.

Please note that this manual describes only one method of carrying out any of the normal work tasks and, if you are an experienced user, there may be other methods. For other methods and more detailed information, see the following manuals.

User's Guide provides a description of all robot functions and describes the programming language in detail. This manual is primarily intended as a reference manual for operators and programmers.

Product Manual provides a description of the installation, how to locate errors in the robot; etc.

If you just wish to be able to start programs, run the robot with the joystick, load programs from diskette, etc., it is not necessary to read Chapters 14-16.

2 Safety

Operational procedures, during training or at any other time, must be carried out safely.

Entering the safeguarded space around the robot may cause severe injury and should be avoided whenever possible. However, if this is necessary, then only authorised personnel may enter the area. The existing safety regulations must always be taken into consideration.

The safety regulations are specified in the chapters on safety in the User's Guide and in relevant plant documentation (if any).

3 System Overview

3.1 General

A robot is made up of two principal parts:

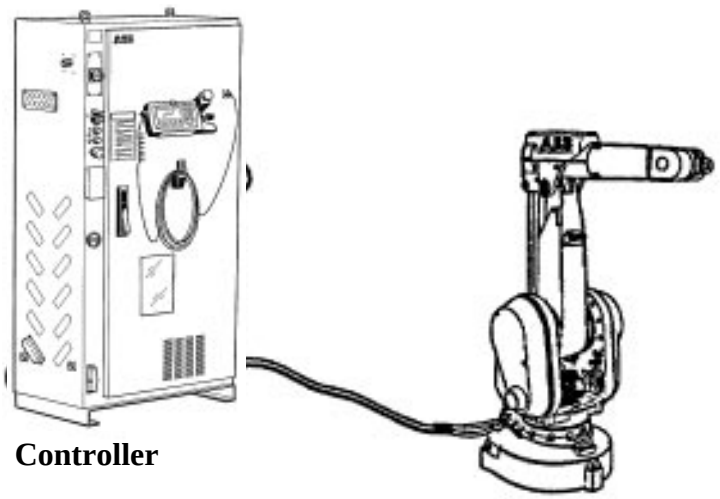


Figure 1 The controller and manipulator are connected by two cables.

You can communicate with the robot using a teach pendant and/or an operator's panel, located on the controller (see Figure 2).

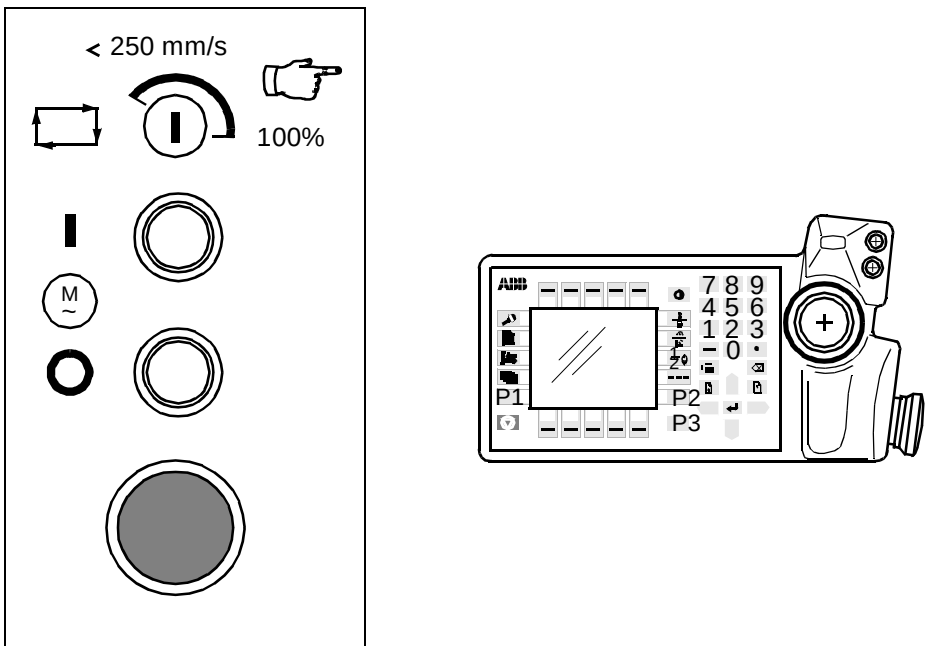


Figure 2 Operator's panel and teach pendant.

3.2 The manipulator

Figure 3 shows the directions in which the various axes of the manipulator can move and what these are called.

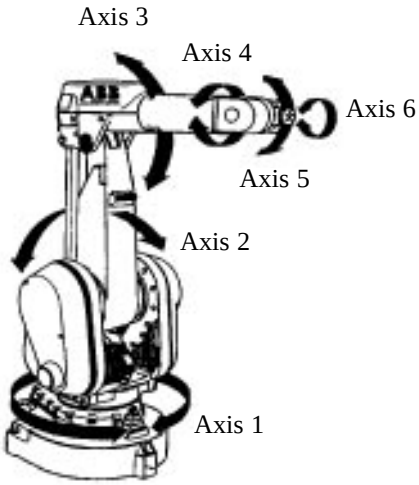


Figure 3 Manipulator, IRB 2400.

3.3 The controller

Figure 4 illustrates the principal parts of the controller.

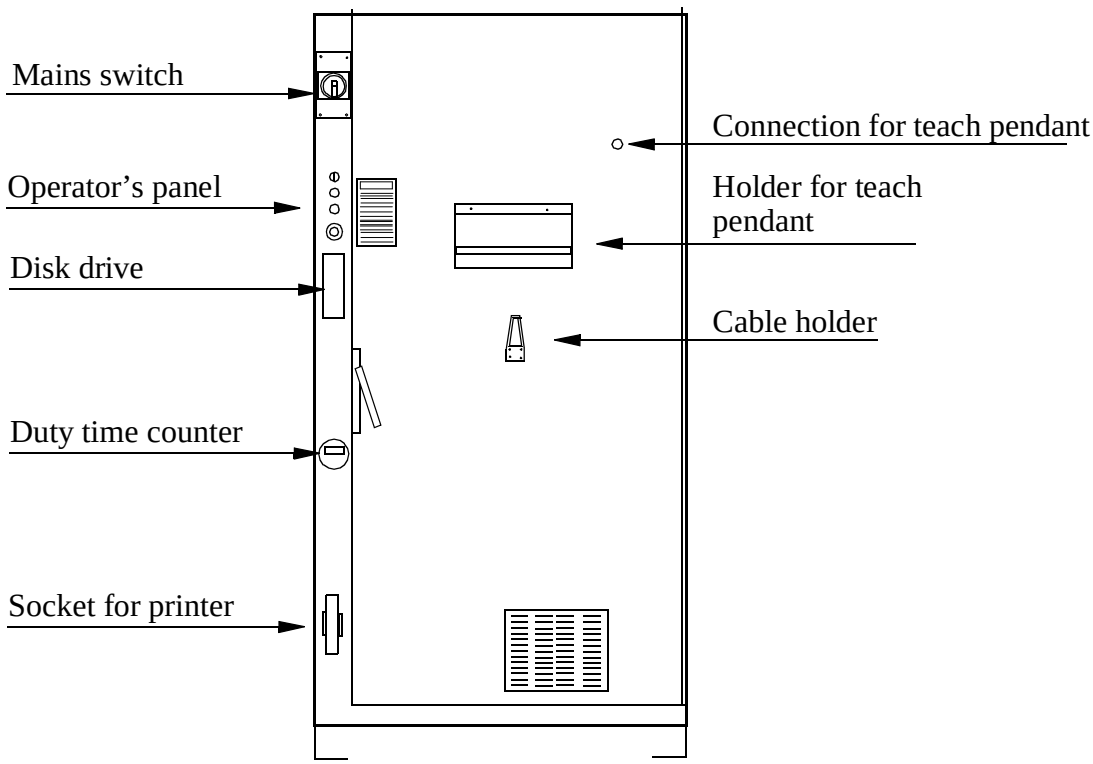
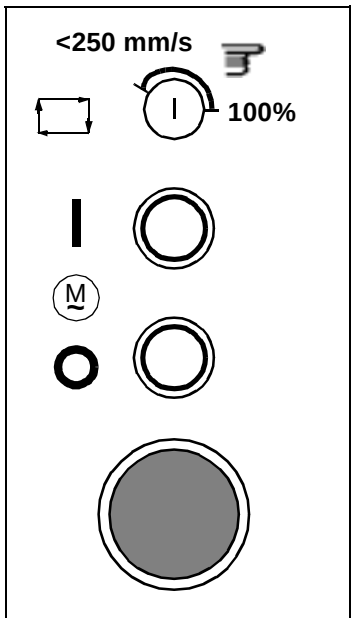


Figure 4 The S4 control system.

3.4 Operator's panel

Figure 5 below shows a close-up of the operator's panel. A short explanation of the push buttons follows.



Operating mode selector
AUTO ( )
MANUAL REDUCED SPEED(< **250 mm/s**)
MANUAL FULL SPEED (**100%**)

MOTORS ON

MOTORS OFF

Emergency stop

Figure 5 The operator's panel.

Operating mode AUTO (Production mode)

Used when running ready-made programs in production. It is not possible to move the manipulator with the joystick in this mode. The **MOTORS OFF** button is continuously lit.

Operating mode MANUAL REDUCED SPEED (Programming mode)

Used when working inside the robot's working area and when programming the robot.

Operating mode MANUAL FULL SPEED (Testing mode)

Used to test run the robot program at full programming speed.

MOTORS ON

In the **MOTORS ON** state, the motors of the robot are activated and the **MOTORS ON** button is continuously lit.

MOTORS OFF

In the **MOTORS OFF** state, the motors are switched off and brakes are applied. When in manual mode, the button flashes.

Emergency stop

The robot stops – regardless of which state or mode the system is in – immediately the emergency stop button is pressed. The button remains pressed in and, to turn to **MOTORS ON** again, must be returned to its original position.

3.5 Teach pendant

The teach pendant is described briefly below; see Figure 6 and Figure 7.

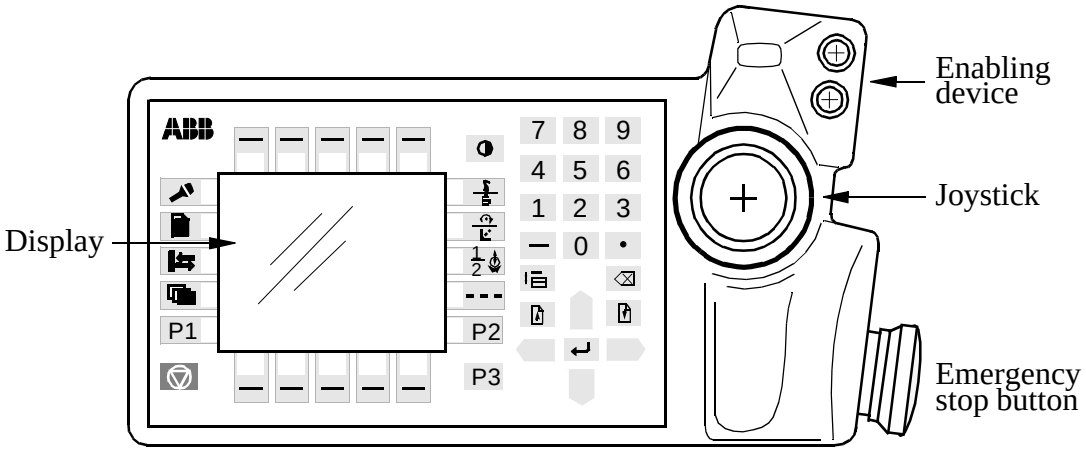


Figure 6 The teach pendant.

Emergency stop

The robot stops – regardless of which state or mode the system is in – immediately the emergency stop button is pressed. The button remains pressed in and, to turn to **MOTORS ON** again, must be returned to its original position.

Enabling device (for safe operation)

A push button on the teach pendant which, when pressed halfway in, takes the system to **MOTORS ON** (if the operating mode selector is switched to one of the two manual modes). When the enabling device is released or pushed all the way in, the robot is taken to the **MOTORS OFF** state.

If the enabling device is released and pressed in halfway again within half a second, the robot will not return to the **MOTORS ON** state.
If this happens, the enabling device must first be released and then pushed halfway in again.

The enabling device should only be activated when the robot is to be moved – either with the joystick or during program execution.

Joystick

The joystick is used to jog (move) the robot manually; e.g. when programming the robot.

Display

Used to display all information during programming, to change programs, etc. It can accommodate 16 lines; each line can accommodate 40 characters.

Figure 7 shows the names of the various keys on the teach pendant.

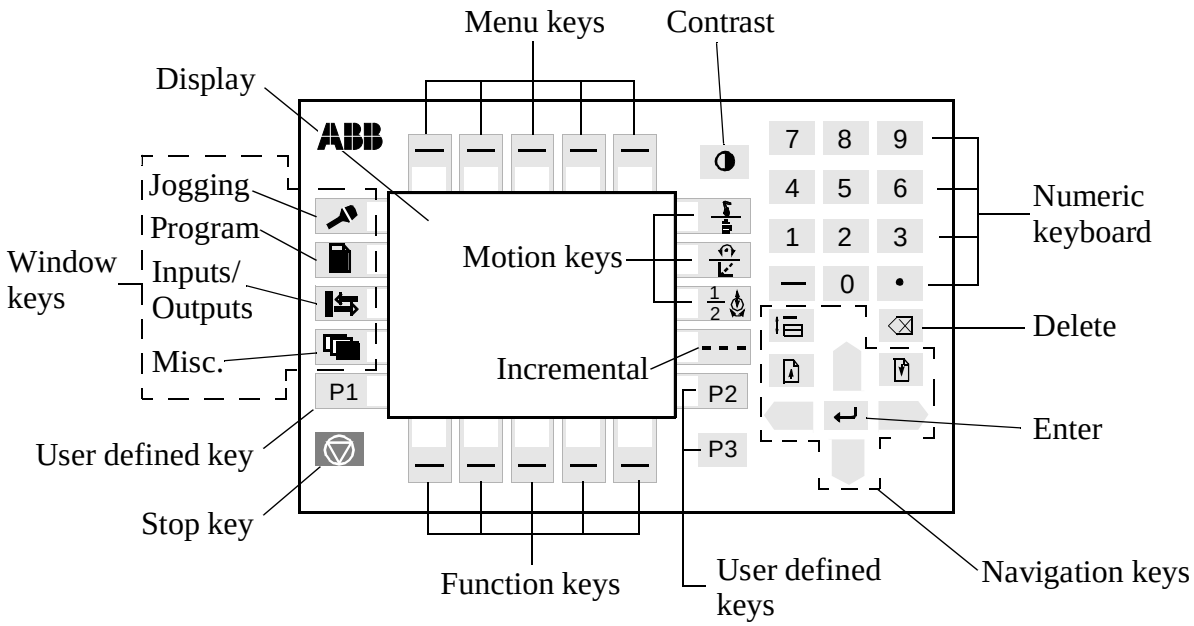


Figure 7 An overview of the various keys on the teach pendant, Version 2.

Window keys (to select a window to work with on the display):

Jogging: Used to jog the robot.

Program: Used to program and test.

Inputs/Outputs: Used to manually operate the input and output signals connected to the robot.

Misc.: Miscellaneous; other windows, i.e. the System Parameters, Service, Production and File Manager windows.

Navigation keys (to move the cursor within a window on the display):

List: Press to move the cursor from one part of the window to another (normally separated by a double line).

Previous/Next page: Press to see the next/previous page.

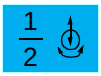
Up and Down arrows: Press to move the cursor up or down.

Left and Right arrows: Press to move the cursor to the left or right.

Motion keys: (to select how the robot or other peripheral equipment should move when using the joystick – during manual operation):

Motion Unit: Press to jog the robot or other mechanical units.

Motion Type: Press to select how the robot should be jogged, reorientation or linear.



Motion Type: Axis by axis movement. 1 = axis 1-3, 2 = axis 4-6

Incremental: Incremental jogging on/off

Other keys:

Stop: Stops program execution.

Contrast: Adjusts contrast of the display

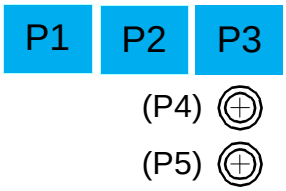
Menu keys: Press to display menus containing various commands.

Function keys: Press to select the various commands directly.

Delete: Deletes the data selected on the display.

Enter: Press to input data.

Programmable keys:



Functions to be defined by the user.

4 Starting the System

You are now going to turn the system on, i.e. get it ready for programming, running programs, etc.

Before you switch the system on, make sure that no-one is inside the safeguarded space around the robot.

1. Switch the mains switch on (see Figure 8). The robot is then automatically checked.

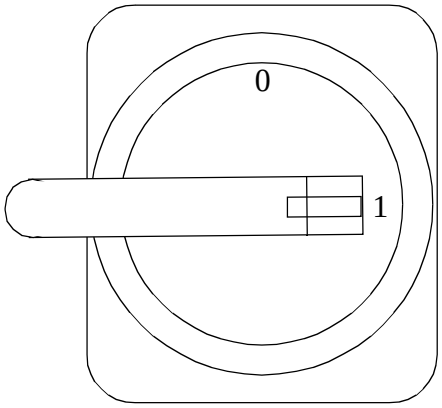


Figure 8 Mains switch

After the system has been checked and no errors are located, the following message (see Figure 9 below) appears on the display.

Figure 9 The "welcome" window may vary slightly depending on the type and version of your robot.

5 Working with Windows

In this chapter, you will find out about the basics of working with windows. The following example shows the window for **Inputs/Outputs** (manual handling of in- and outputs).

1. Press the **Inputs/Outputs** window key (see Figure 10.)

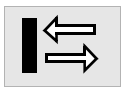


Figure 10 The Inputs/Outputs application key, both versions.

2. The window for manual I/O is now shown on the display, as in Figure 11. The appearance of the I/O list may vary depending on how the signals have been defined and how many I/O boards there are in the system.

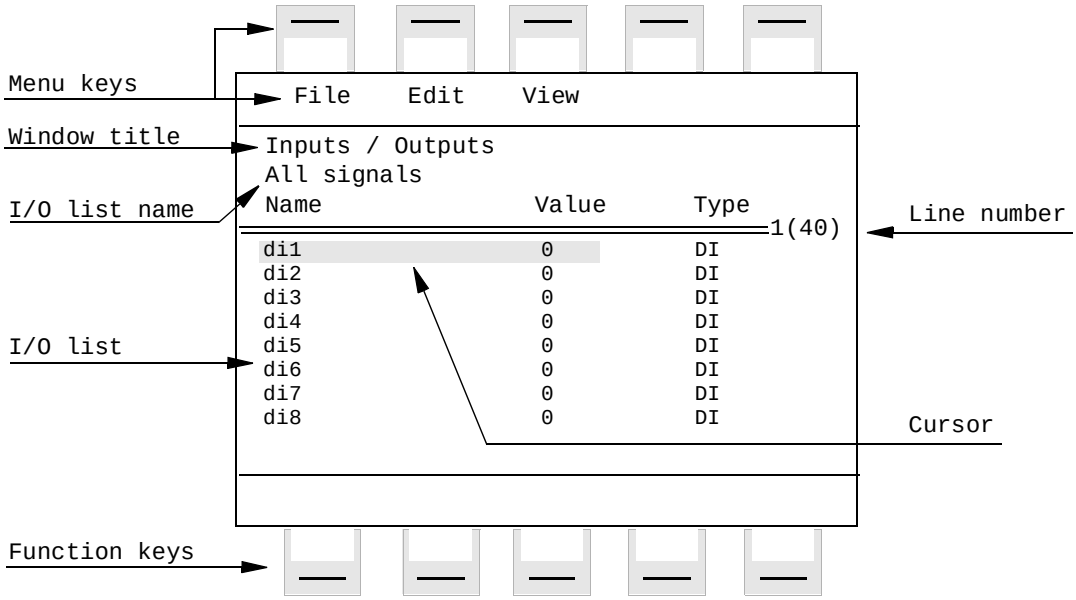


Figure 11 Window for manual I/O handling.

When a digital output is selected, its status can be changed using the function keys. If you press a function key with an arrow to the right of its name, e.g. ***Test***->, this will call up a new window to the display.

3. You can select a signal in the list (move the cursor) in several ways:

Movement	**Select**
One line up	ArrowUp
One line down	ArrowDown
To the first line in the list	**Goto top** from the **Edit** menu
To the last line in the list	**Goto bottom** from the **Edit** menu
To the next page	NextPage
To the previous page	PreviousPage
To select a specific line in the list	**Goto** from the **Edit** menu; enter the desired line number and press ***OK***

4. Windows are sometimes divided in two by a double line (see Figure 12).

5. Press the NextPage key until an output is shown on the first line, two function keys will be displayed (see Figure 12).

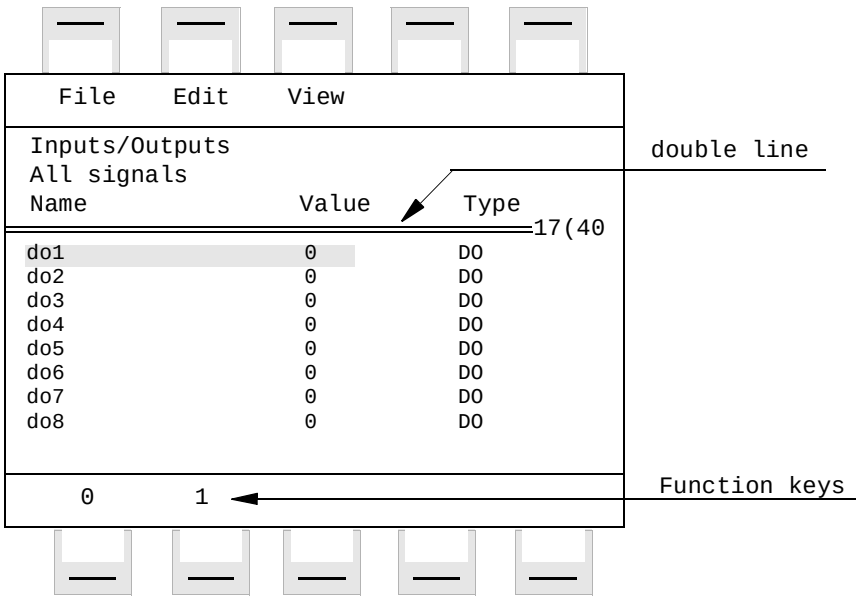


Figure 12 A window with two parts.

In some windows, you can move the cursor between the different parts of the window.

In these windows, move the cursor using the List key .

6. There are four window keys on the teach pendant (see Figure 13 below and Chapter 3).

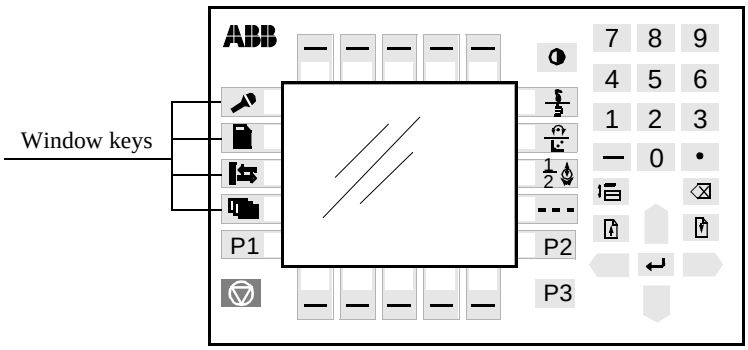


Figure 13 The four window keys.

When you press a window key, the active window will be hidden under the new one. Each time you select a window, it will look the same as it did the last time you worked with it.

6 Jogging the Robot Using the Joystick

You can move (jog) the robot using the joystick on the teach pendant. This chapter describes how to jog the robot linearly (i.e. in a straight line) and step by step, to make it easier to position the robot exactly (known as incremental jogging).

6.1 Linear jogging

1. Make sure that the operating mode selector is in the < 250 mm/s position, as shown in Figure 14.

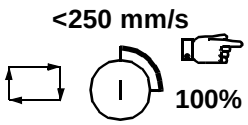


Figure 14 The maximum speed during manual operation is 250 mm/s.

2. Check that the **Robot** motion unit and the **Linear** motion type are selected (see Figure 15).

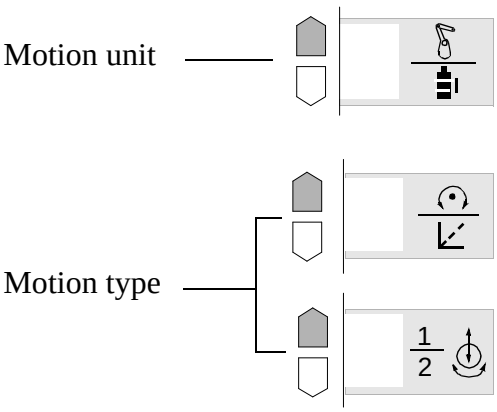


Figure 15 Motion keys, LEDs shows the current settings.

With the **Motion unit** key, you can choose between operating the robot, or some other unit connected to the controller, using the joystick. Select the robot for this exercise.

With the **Motion type** key, you can choose the way you want the robot to move when you use the joystick during manual operation.

You can choose:

- linear movement
- reorientation of a particular end-effector
- axis-by-axis movement (group 1: axes 1-3; group 2: axes 4-6)

We will use linear motion for the purposes of this exercise.

When linear type motion is selected, the robot will move as shown in Figure 16.

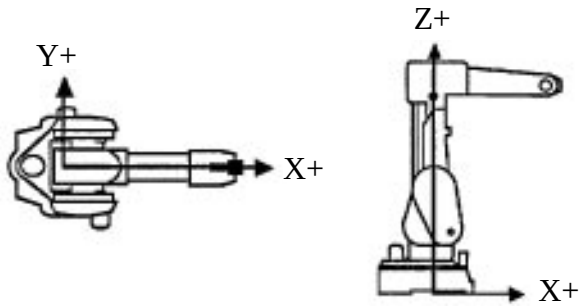


Figure 16 X, Y, Z form the robot's coordinate system.

The point that will move linearly, along the axes of the coordinate system above, is called the **Tool Centre Point (TCP) 0**. It is located at the front of the upper arm, in the centre of the robot's face plate (see Figure 17).

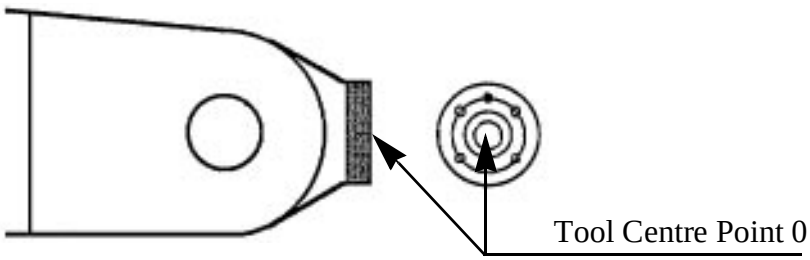


Figure 17 The centre of the face plate is called TCP 0.

3. Push the enabling device halfway in to switch the **MOTORS ON**.

4. Now, jog the robot using the joystick.
   Standing in front of the robot, the TCP 0 will, depending on how you move the joystick, move linearly along the X-, Y- and Z-axes (see Figure 18).

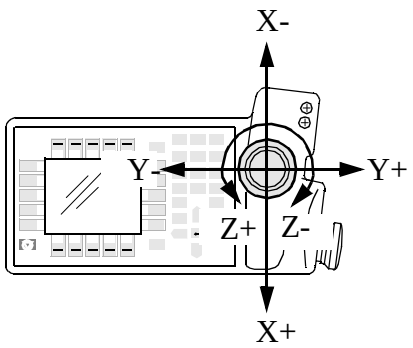


Figure 18 Robot movements with different joystick deflections.

Try jogging the robot in the directions corresponding to X, Y and Z above.
You can also combine the various movements of the joystick and move in several directions simultaneously. Note that the speed of the robot depends on how much you move the joystick. The larger deflection, the faster the robot moves.

6.2 Fine positioning

1. Press the **Jogging** window key (see Figure 19).

Figure 19 The Jogging window key.

A window like the one in Figure 20 will appear.

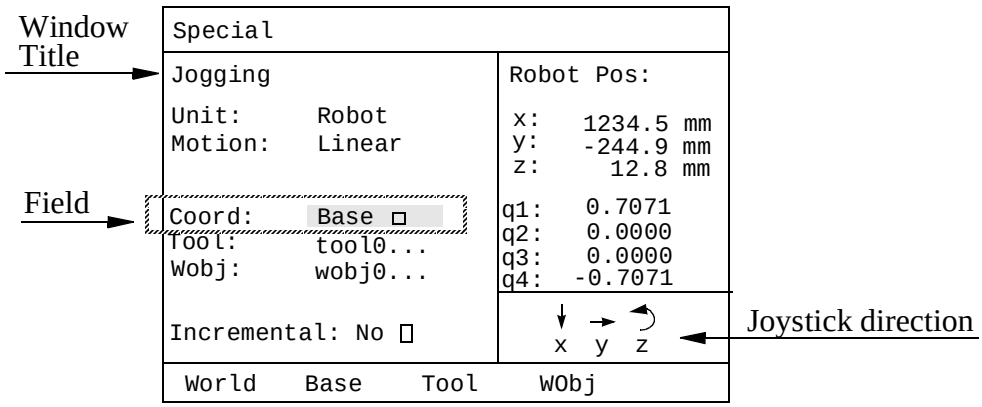


Figure 20 The Jogging window.

The appearance of the window changes depending on the type of window selected (i.e. depending on what you want to do).

The **Menu keys** perform different commands. The list of commands available is displayed in a pull-down menu when you press any of the menu keys.

The area enclosed by a dashed line is called a **Field**. The highlighted (shaded, grey) area is known as an input field and can be changed by selecting a different function using one of the **Function keys** (or, in some cases, using the **Motion keys** on the teach pendant).

The highlighted input field in Figure 20 is marked with a " □ " after it which means that:

- Selection is done using a **Function key**

2. Move the cursor (the shaded field) to the **Incremental** field using the **Down arrow** key on the teach pendant (see Figure 21).

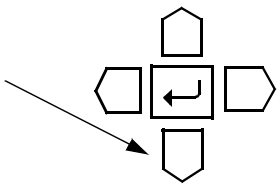


Figure 21 The Down arrow navigation key.

3. If you move the cursor to the **Incremental** field, as in Figure 22, you can choose incremental jogging by pressing one the function keys.

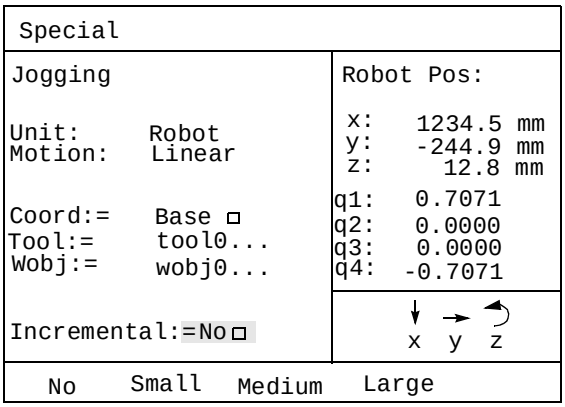


Figure 22 Selection of incremental jogging.

If you press the ***Small, Medium*** or ***Large*** function key, the No in the Incremental field will be immediately replaced.

The robot will then move one step at a time each time you move the joystick; the size of the steps (Small, Medium or Large) will depend on your choice.

You can also use the key --- to turn incremental movement on and off.

Try operating the robot using the joystick and note how the robot moves.

More information on manual operation, the various coordinate systems, etc., can be found in the chapter on jogging in the User's Guide.

If you do not wish to continue this exercise, switch off the system as explained in Chapter 12 "Switching the System off".

7 Selecting a Program

This chapter explains how to open (choose) a program. A **program** is usually made up of three different parts, one **main routine** (always present), a number of **subroutines** and **program data**. Only one main routine is permitted per program (see Figure 23).

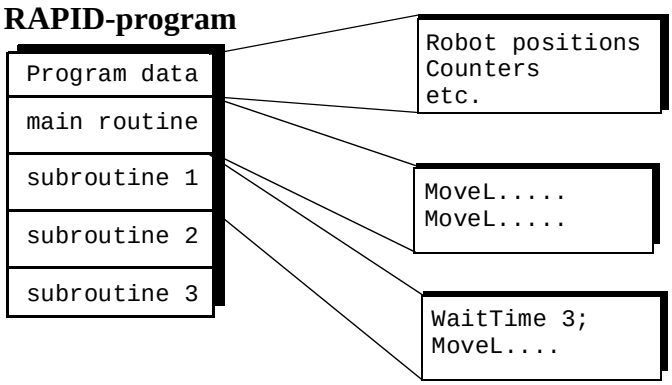


Figure 23 Program structure.

If you open a program, that program replaces the program in the robot's memory. When a program is opened, the main routine will be shown on the display with the first instruction in the main routine selected (highlighted).

7.1 Using the training program

The training program is stored on the IRB 2 system diskette, under the directory DEMO, and is called "EXERCISE".

1. Turn the operating mode selector on the operator's panel to < 250 mm/s.
2. Press the **Program** window key (see Figure 24).

Figure 24 The Program window key.

If there is no program in the robot's memory, the following window will appear (Figure 25), otherwise you will see the program that is stored in the memory of the robot.

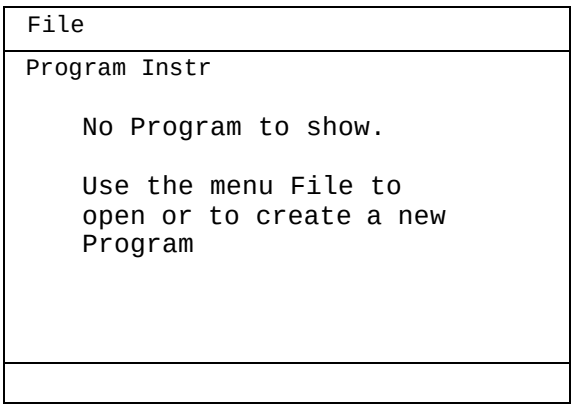


Figure 25 The Program window.

3. Insert the Setup diskette into the disk drive at the front of the cabinet. The diskette should be inserted as in Figure 26.

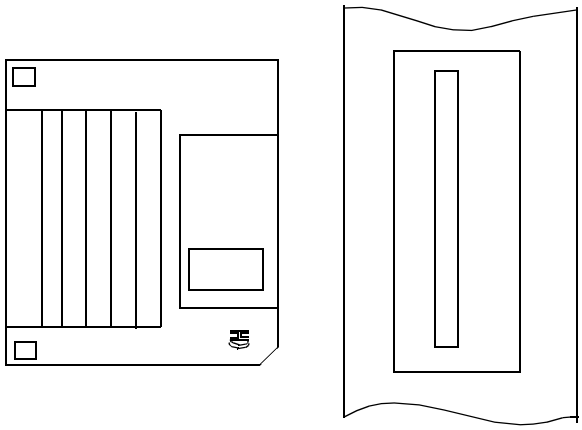


Figure 26 Inserting the diskette.

4. Press the **File** menu key (see Figure 25 above). The window in Figure 27 will appear.

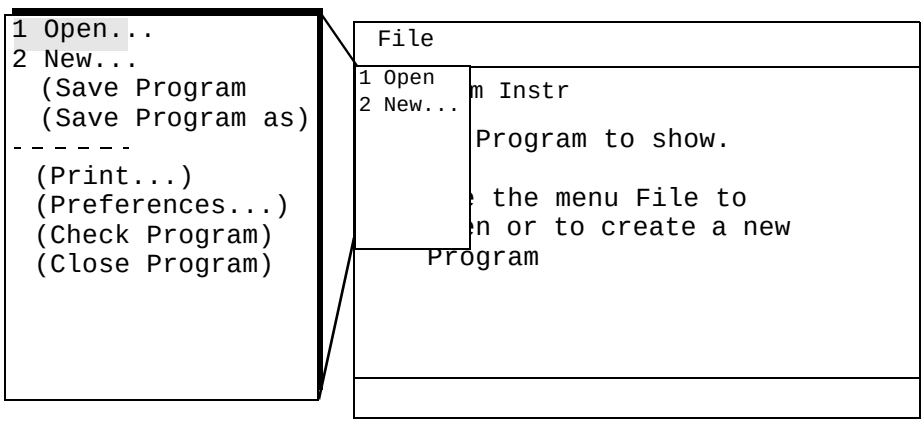


Figure 27 The File menu when there is no program in the robot's internal memory.

The menu that now appears on your display is called a “pull-down menu”. All commands that can be chosen from the **File** menu are listed here (commands that cannot be chosen are indicated by parentheses). The other menu keys work in the same way.

From now on we will write **File: Open...**, **File: Save**, etc. The name on the left of the colon is the name of the menu and the name on the right stands for the command you should choose.

The first function in the pull-down menu is always highlighted when you press the menu key. You can move the cursor within the menu using the arrow keys on the teach pendant (see Figure 24). When you have selected the command you want to choose, press **Enter**.

You can also use the numeric keyboard to choose a command; to do this enter the number in front of the command.

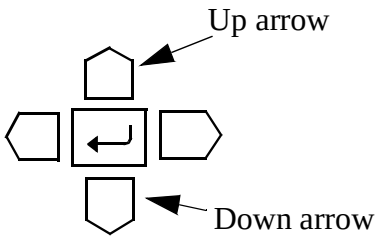


Figure 28 Navigation keys: Up arrow and Down arrow.

Three dots “...” following the command means that a dialog box will appear when that command is chosen.

To remove a pull-down menu, press the menu key with which you opened it.

5. After selecting **Open...**, press **Enter** (see Figure 29).
   This means that the “Open...” command will be carried out. However, as it has three dots “...” after it, the command will not be performed directly since more information is required. You must now, in this case, choose the particular program you wish to open.

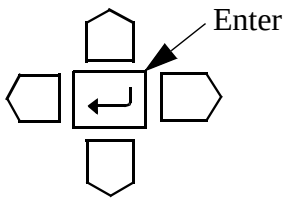


Figure 29 The Enter key on the teach pendant.

Using the ***Unit*** function key, you can switch between the robot's internal memory (ram1disk), the diskette unit (flp1:) or some other type of mass storage device.

6. Press ***Unit***. Check that "flp1:" appears after Massmemory unit:=. A dialog box will appear and the contents of the diskette will be shown, as in Figure 30.

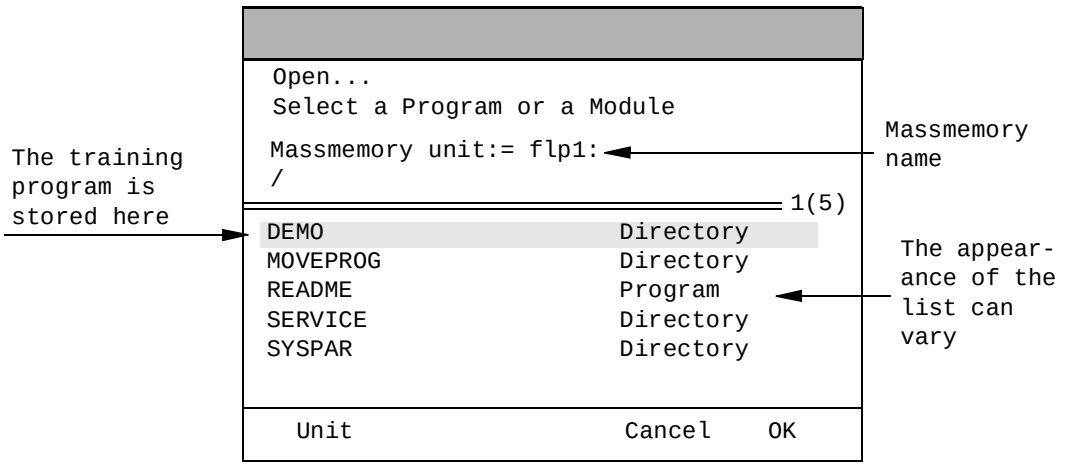


Figure 30 The Open dialog box.

If a dialog box (does not have any menus) is closed by pressing ***Cancel***, the command requested will not be carried out. If you press ***OK***, the command requested will be carried out and the dialog box will close.

7. Select **DEMO**. Move the cursor with the Up and Down arrow keys.

8. Press **Enter**.

9. Select **EXERCISE**.

10. Press ***OK*** and a window like the one in Figure 31 will appear.

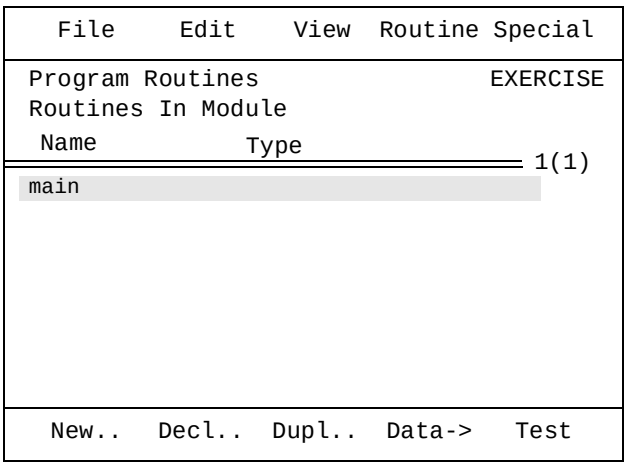


Figure 31 Opening the main file of the training program.

11. Then press Enter  . The window in Figure 32 appears.

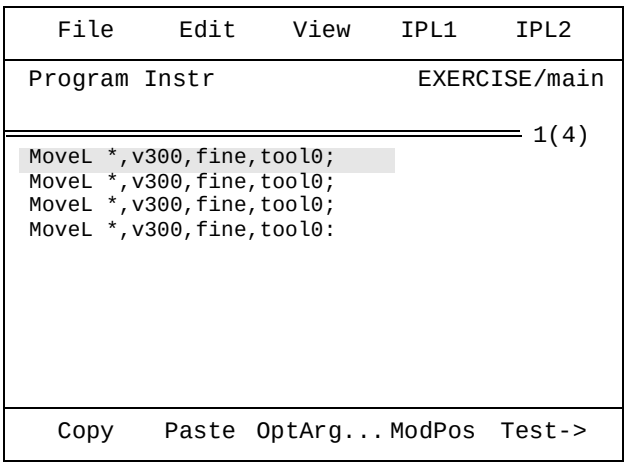


Figure 32 The training program appears on the display.

This is the main routine in the training program. It consists of four “move” instructions.

The routines consist of different types of instructions, such as move instructions, wait instructions, etc. Each instruction is followed by different arguments. Arguments can, depending on their type, be changed or omitted altogether. Figure 33 indicates an example of an instruction.

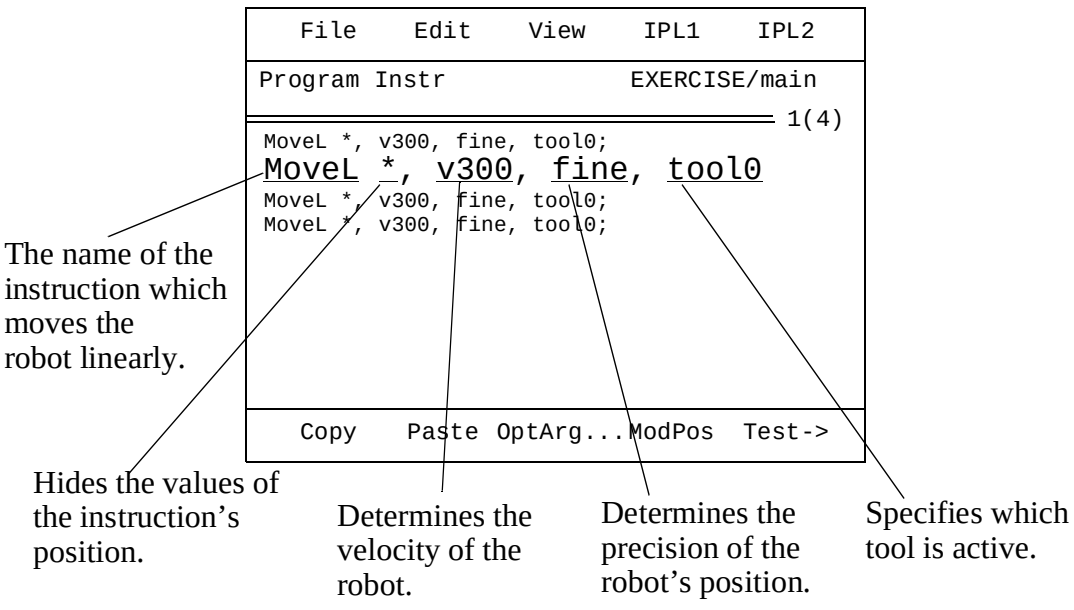


Figure 33 Example of a motion instruction.

8 Starting the Program

You are now going to start the program you just opened. It should first be run step by step using reduced velocity, then continuously.

The program consists of four motion instructions and includes positions near the robot's "calibration position" (see Figure 34).

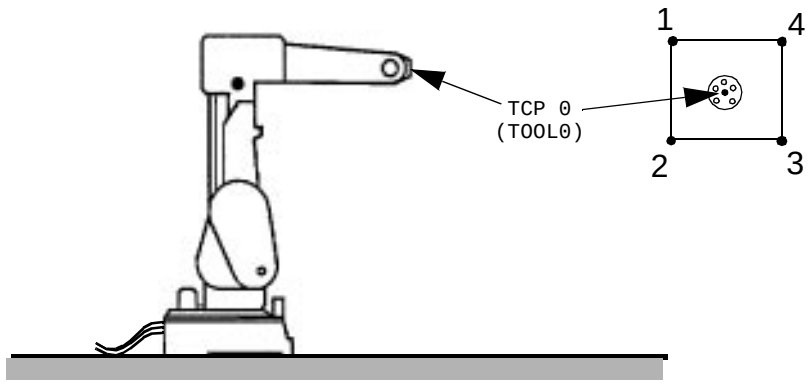


Figure 34 The robot's calibration position

Before starting the program move axis 5, manually with the joystick, downwards about 45°. (For information on the various robot axes, see Chapter 3.2 on page 8.)

1. Press the **Program**  window key and a window, like the one in Figure 35, will appear (you have already reached this stage if you have come directly from Chapter 7).

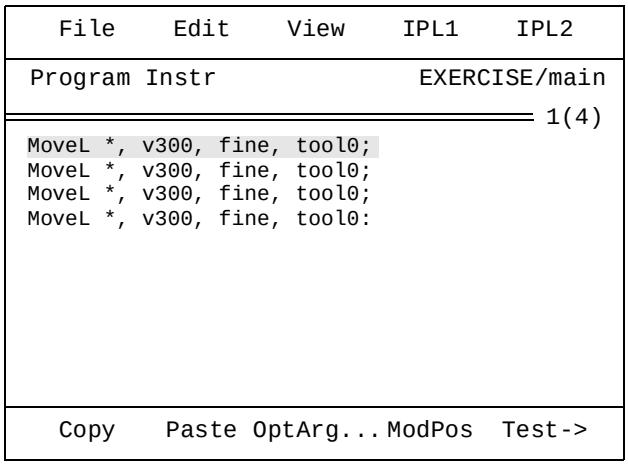


Figure 35 The Program window.

2. Press the ***Test*** function key. The window in Figure 36 appears.

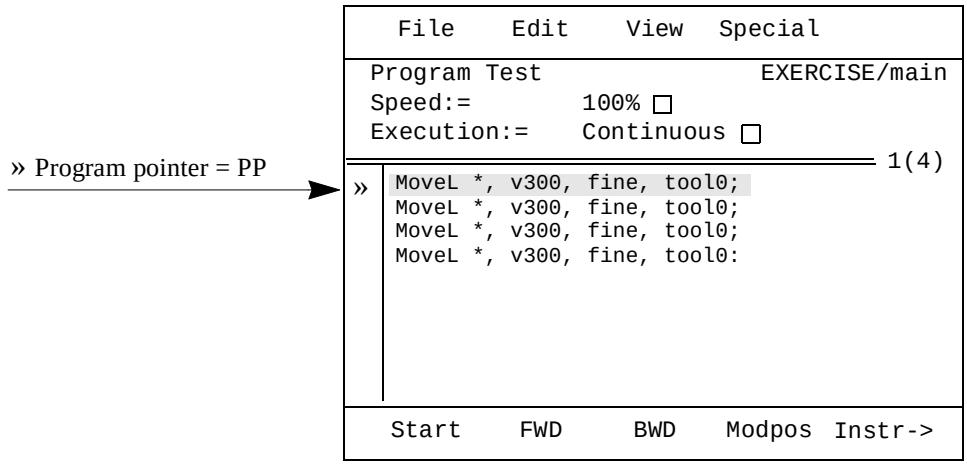
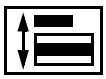


Figure 36 The Program Test window.

Function keys displayed:

- **Start**: continuous running of the program.
- **FWD**: one instruction forward.
- **BWD**: one instruction backward.
- **Instr->**: select the Program instruction window again.

The program point (PP) indicates the instruction with which the program will start when you press one of the options ***Start***, ***FWD*** or ***BWD***.

3. Select the upper part of the window by pressing the List  key**.**

4. Reduce the velocity to 75% by pressing the ***-%*** function key (see Figure 37). Correction is carried out in increments of 5%.

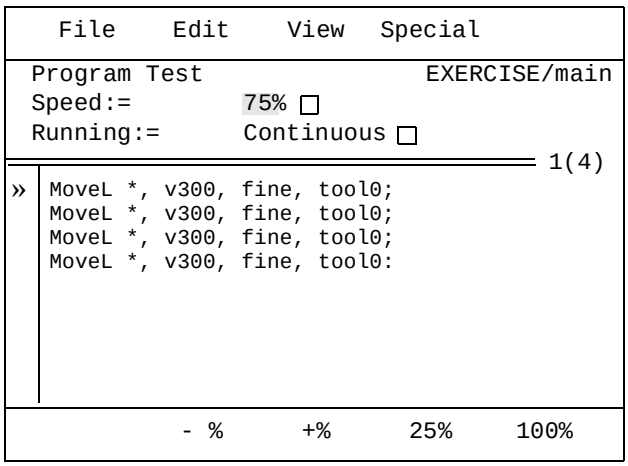


Figure 37 Correction of velocity.

5. Move (using the same key as in point 3) the cursor back to the first line of the program (see Figure 38).

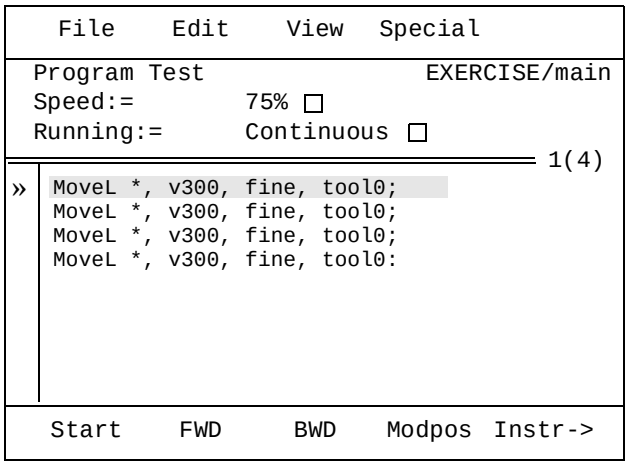


Figure 38 Window for starting the program.

The program can now be started. Make sure that no-one is inside the safeguarded space around the robot.

6. Start the program by pushing in the enabling device and pressing the ***FWD*** function key (see Figure 38).

   When the program has started, the robot will carry out one instruction, then it will stop. Press ***FWD*** to initiate the next instruction, press again for the next one, and so on.

The window in Figure 39 is displayed during the execution of the program.

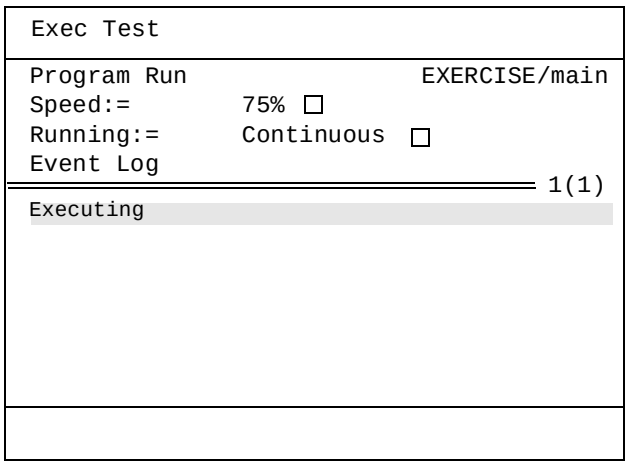


Figure 39 Window during program execution.

7. Go through all the program instructions step by step. Press ***FWD*** repeatedly after the robot is in position.

8. If you press ***FWD*** when the program comes to the final instruction, the program will start from the beginning again.

9. Let the robot move to position number 4 (see Figure 34).

10. Move, in the same way as before, the cursor to the **Running** field and change to **Cycle** execution.

11. Move the cursor back to the program.

12. Start the program by pressing ***Start***.

    When **Cycle** is selected the program will be executed once, and then will stop in position 4 (one cycle).

13. Select **Continuous** execution again.

9 Stopping the Program

Stop the program by pressing the **Stop** key on the teach pendant (see Figure 40).

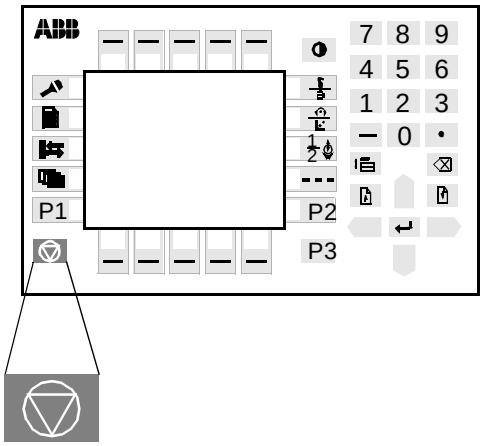


Figure 40 Stop key on the teach pendant.

10 Automatic Mode

Automatic mode is used to execute ready-made programs.

1. Turn the operating mode selector on the operator's panel to the **Auto** position. The window in Figure 41 appears.

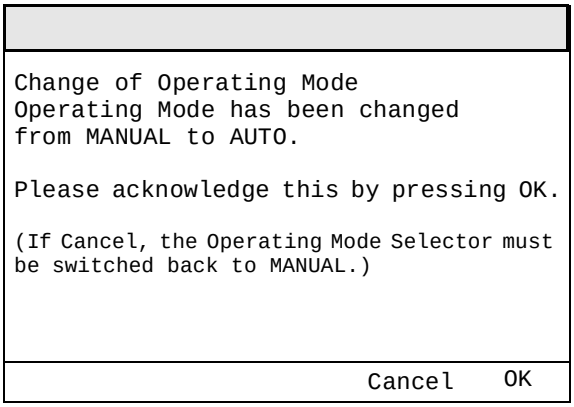


Figure 41 Dialog box used to confirm a change from manual to automatic mode.

2. Press ***OK***. You have now changed to automatic mode and the **Production** window appears on the display (see Figure 42).

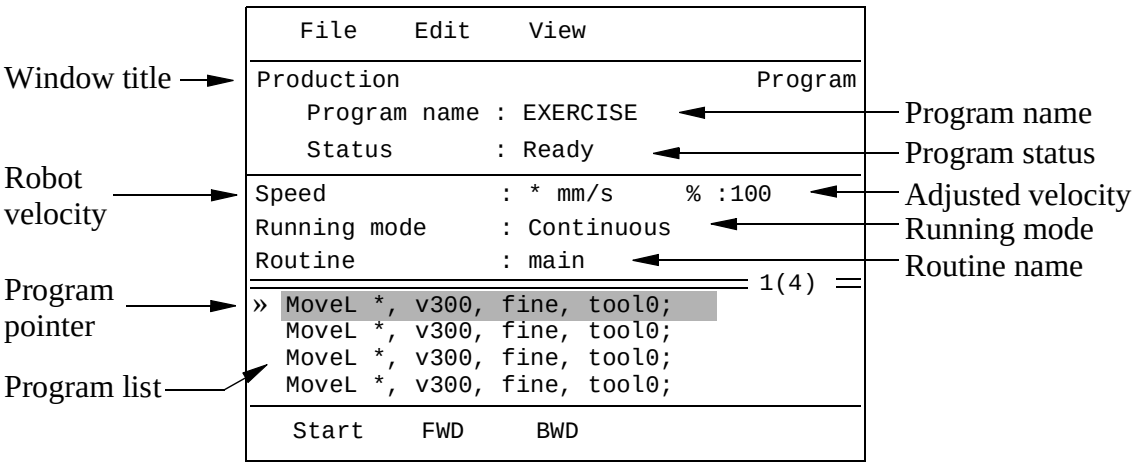


Figure 42 The Production window in auto mode.

3. Press the **MOTORS ON** button on the operators panel.
4. Start the program with the ***Start*** function key.
5. Stop the program with the **STOP** button on the teach pendant.
6. Then press **MOTORS OFF** on the operator's panel.
7. Switch back to < 250 mm/s.

For further information, see Chapter 8, Running Production in the User's Guide.

11 Errors

A window displaying an error message appears whenever there is any type of error (see Figure 43).

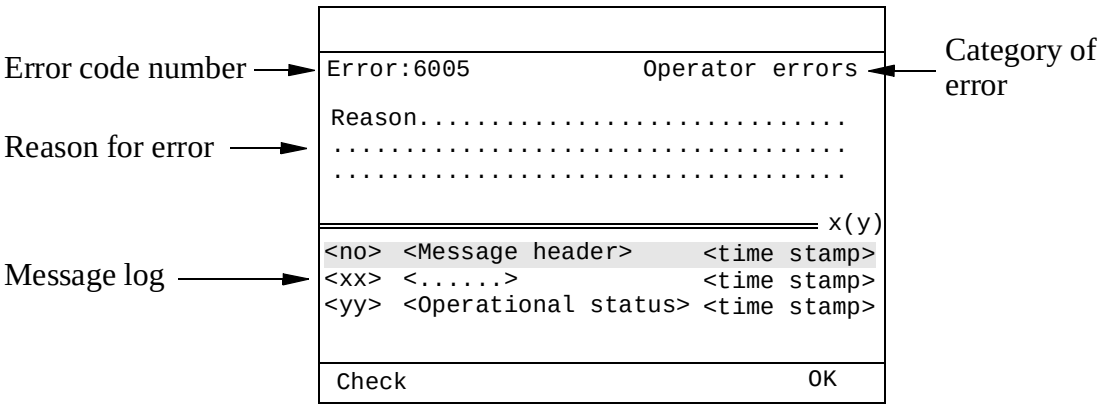


Figure 43 Example of a window displaying an error message.

Error code number

A number unique to each error.

Category of error

Assigns errors to groups relating to the type of error. Each category has its own code number series; e.g. Operator errors (6001-6999).

Reason

Describes the reason for the error in plain language. For more information regarding hardware faults, see the Product Manual.

Message log

Indicates the most recent errors. The error indicated on the first line is displayed in the window. The log shows the error code number, a brief description of the error, and the time the error was registered. If you highlight one of the errors in the log, the window will then be updated with the appropriate error code number, reason and category.

The ***Check*** function key can be used to get help on how handle a specific error.

If you press ***OK***, the error-message window will disappear.

Using the joystick, try to manually operate the robot outside its operating area. You will then see an example of an error message.

12 Switching the System off

If you are going to continue with the rest of the exercises, you can skip this chapter.

All output signals will be set to zero when the robot is switched off. This may affect the gripper and peripheral equipment.
So, before switching the system off, check first that the equipment, and any people in the area, will not come to any harm.

1. If the program is running, stop it by pressing the **Stop** push button on the teach pendant.

2. Then press the **MOTORS OFF** push button on the operator's panel.

3. After you have done this, switch off the mains switch.

   The robot's memory is battery-backed and is thus not affected when the system is switched off.

13 Changing a Program

The following chapters are intended to be read by people who will create programs, edit programs, etc.

This chapter explains some of the ways in which you can change the program you opened and started in the preceding chapters. You will:

- run the program step by step until you get to the position you want to modify
- modify this position
- change an argument in an instruction
- enter a new instruction (position)
- program a time delay (WaitTime)

13.1 Modifying positions

1. If you have exited the previous exercises, choose the **Program** window (see Figure 44).

Figure 44 The Program window key.

The window in Figure 45 appears on the display.

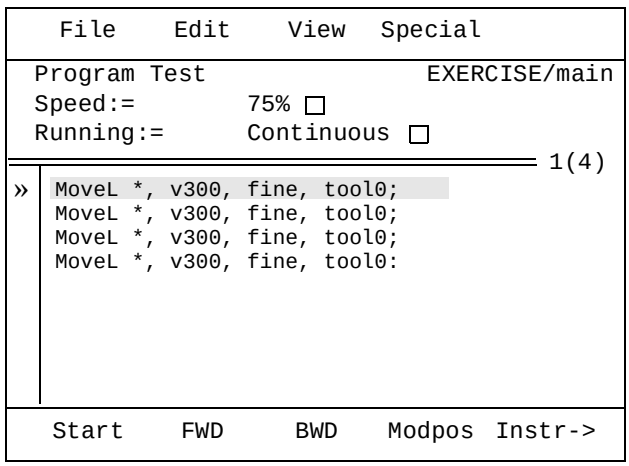


Figure 45 The Program Instr window.

2. Push in the enabling device and press ***FWD***. Move the robot to the first position in the program (the first instruction should be highlighted).

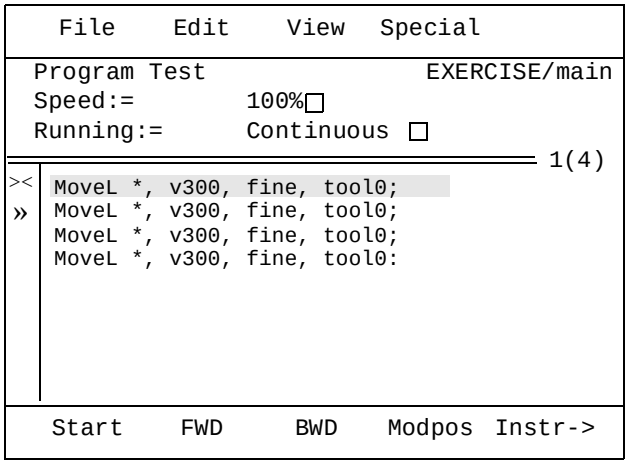


Figure 46 The first instruction is selected.

3. Then move the robot to a new position with the joystick.

4. Press the ***ModPos*** function key. The original position specified will now change to the current position of the robot.

5. Activate the enabling device and press ***FWD*** again to move the robot to the next position.

   Repeat points 3 to 5 and go through all the positions in the training program.

6. Test run the program step by step **without** reducing the velocity.
   Stop the program in any position and press the ***Instr*** function key (to terminate the Program Test). The window in Figure 47 will then appear on the display.

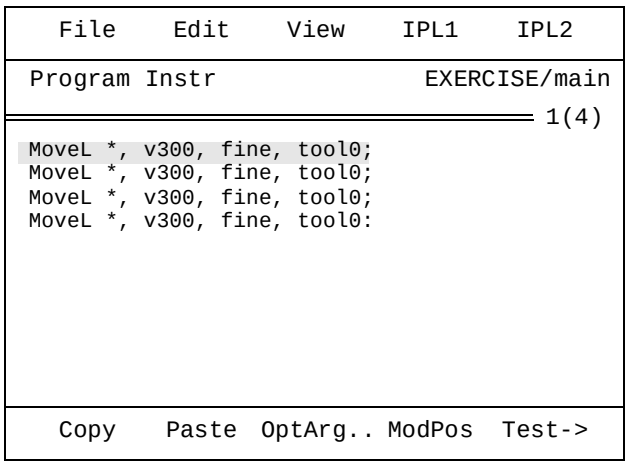
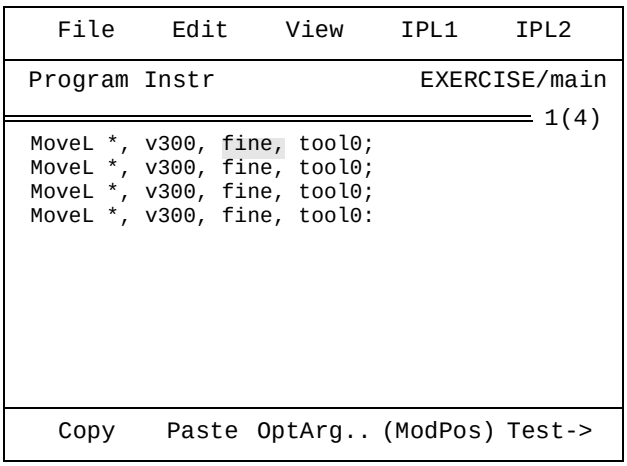


Figure 47 The Program Instr window.

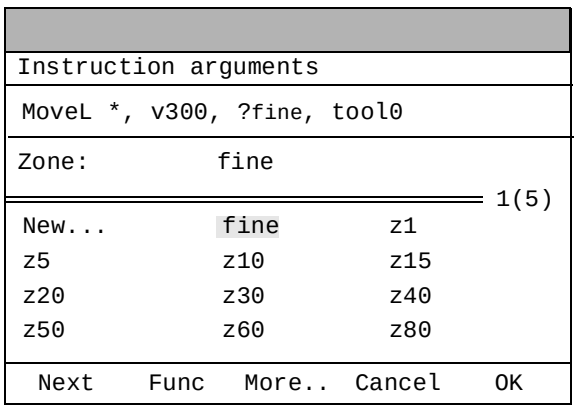
13.2 Changing arguments

You are now going to change one of the arguments of the first move instruction (MoveL), which should be highlighted. You are going to change the precision of the position.

1. Select the "**fine**" argument (see Figure 48). Move the cursor using the right arrow key.



2. Press **Enter**. The window in Figure 49 appears.



3. Move the cursor to **z10**.
4. Press Enter and the fine argument will change to z10.
5. Then press ***OK***. The instruction has now changed to z10.
6. Move the cursor so as to select the complete instruction (see Figure 50).

13.3 Adding instructions

You are now going to add a movement instruction to the program after the first instruction. The **Program Instr** window in Figure 50 should now appear on the display.

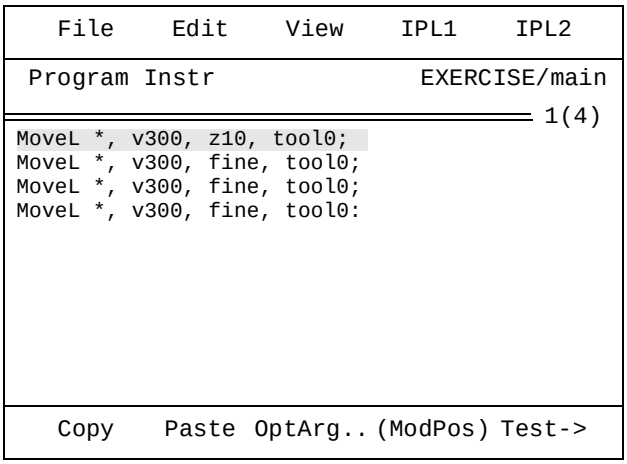


Figure 50 Program window.

1. Press the ***Copy*** function key to copy the first instruction (highlighted) in Figure 50.
2. Then press ***Paste***. The window in Figure 51 appears. As it is the first instruction in the program that is highlighted, you will be asked where you want the new instruction to be inserted.

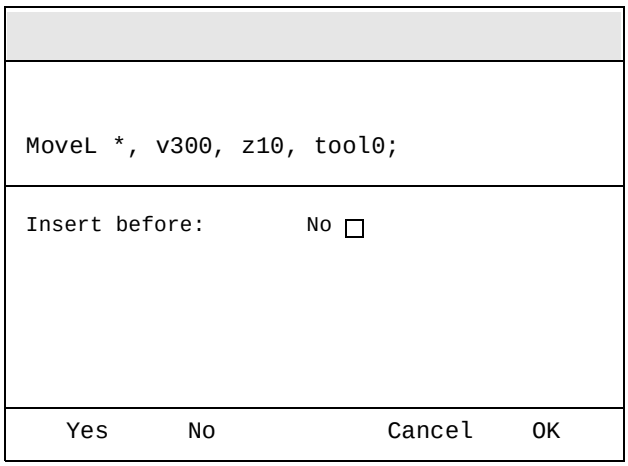


Figure 51 Dialog box used to insert new instructions when the first instruction is highlighted.

3. Select ***No***. Press ***OK***.

4. The new instruction will be inserted directly **under** the instruction that was highlighted, and will be highlighted itself.

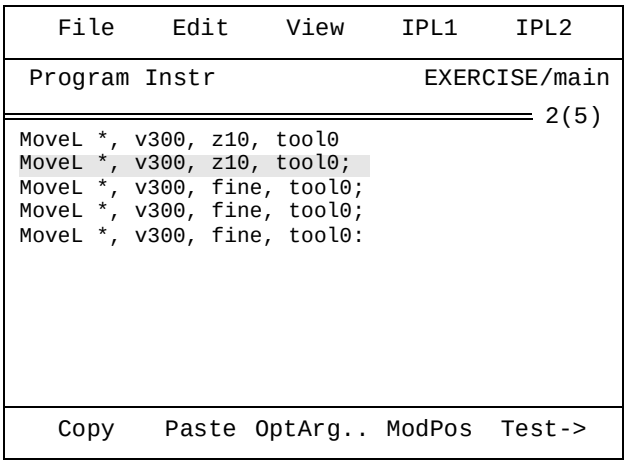


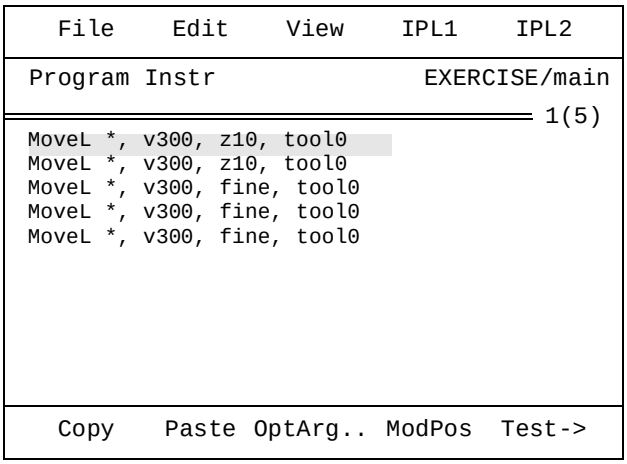
Figure 52 An extra position (the same as the one copied) is added to the program.

5. Using the joystick, move the robot to the position to which you want it moved.
6. Press ***Modpos*** (see Figure 52).
7. Test run the program using continuous execution.
8. Select Test->.
9. Push the enabling device.
10. Press Start.

13.4 Programming a delay

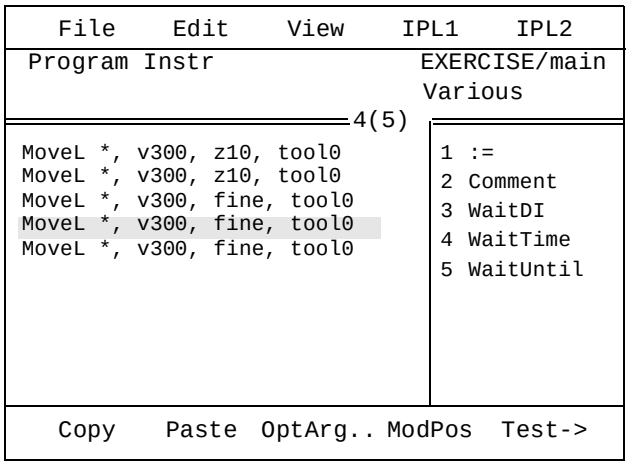
You are now going to program a delay, i.e. make the robot wait a specified amount of time. The new instruction will be inserted after the fourth instruction.

1. When the program test-run is completed, press the ***Instr*** function key.
   The window in Figure 53 appears.



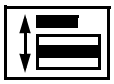
2. Using the arrow keys (up and down), move the cursor to the fourth instruction in the program. The new instruction will be inserted under the highlighted one.

3. Select **IPL1: Various** (see Figure 54). The window in Figure 54 appears.



4. Select the desired instruction from the pick list, in one of the following ways:

   - Using the numeric keyboard, enter the number (4) shown in front of WaitTime (see Figure 54). (The numeric keyboard is illustrated in chapter 3.)

   - Select the pick list by pressing the **List** key  . Then select the desired instruction and press **Enter**. See Figure 55.

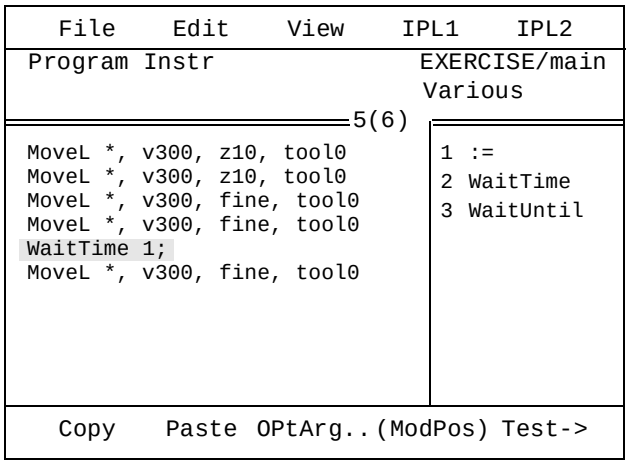


Figure 55 The window after selection of the WaitTime instruction.

5. A window like the one in Figure 56 appears.

Figure 56 Dialog box for entering arguments.

6. Type 3 on the numeric keyboard to get a wait time of 3 seconds.

7. Press ***OK***. The window like the one in Figure 57 appears.

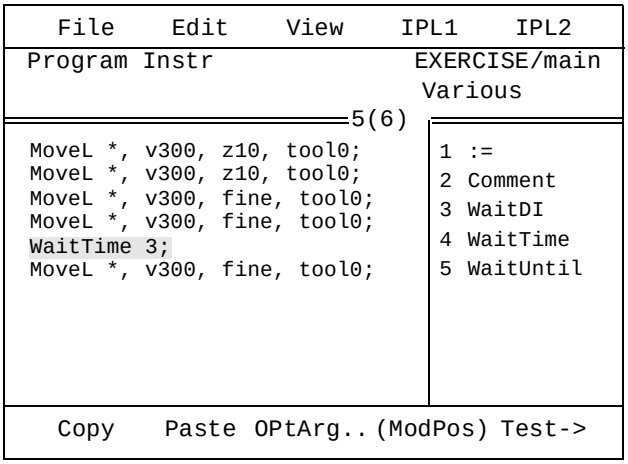


Figure 57 The Program Instr appears on the display.

8. Press **Edit: IPlhide** to remove the pick list.

9. Now test run the program again using the Program Test window. Choose continuous execution.

INFORMATION

The **Edit** menu includes a number of functions which can be used to edit the program (see Figure 58).

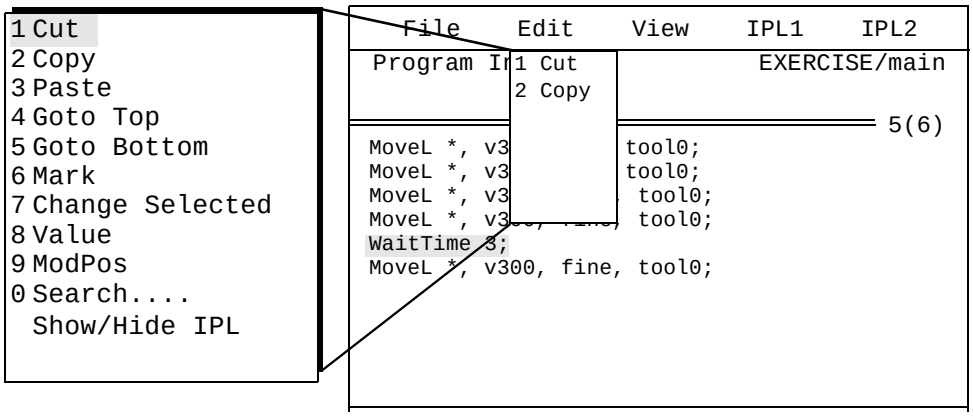


Figure 58 The Edit pull-down menu.

Press the **Edit** menu key again to remove the menu.

See the User's Guide for more detailed information.

14 Storing the Program on Diskette

You are now going to copy the program to a diskette.
Use 3.5” HD (High Density) diskettes.

Do not use the Setup diskette to store the exercise program.

If your diskette is new, you will have to format it first. This is done under the **File Manager**, which you can access using the **Miscellaneous** key.

14.1 Formatting a diskette

Note: If you format a diskette which contains information, all information will be deleted.

1. Insert the diskette into the disk drive on the front of the controller cabinet.
2. Choose the **Miscellaneous** window (see Figure 59).

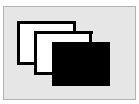


Figure 59 The Miscellaneous window key.

A list of available windows will appear on the display (see Figure 60).

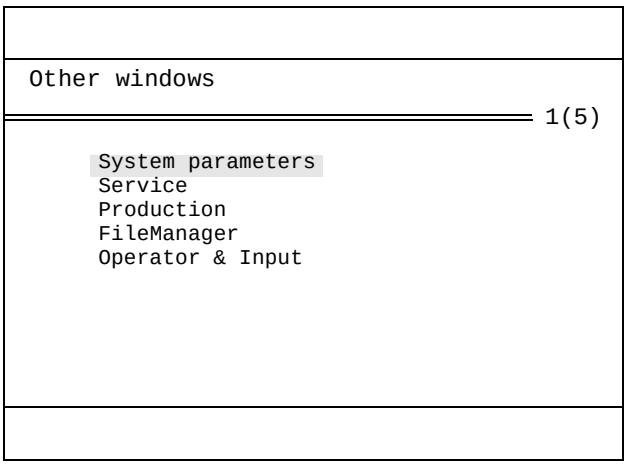


Figure 60 Available windows under Miscellaneous.

3. Select **FileManager** from the list.
4. Press Enter. The window in Figure 61 appears.

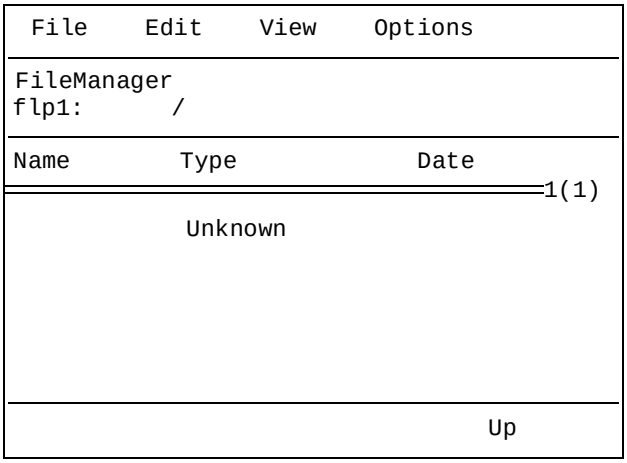


Figure 61 The window displays a list of files if there are any on the diskette.

5. Select **Options: Format.** The dialog box in Figure 62 appears.

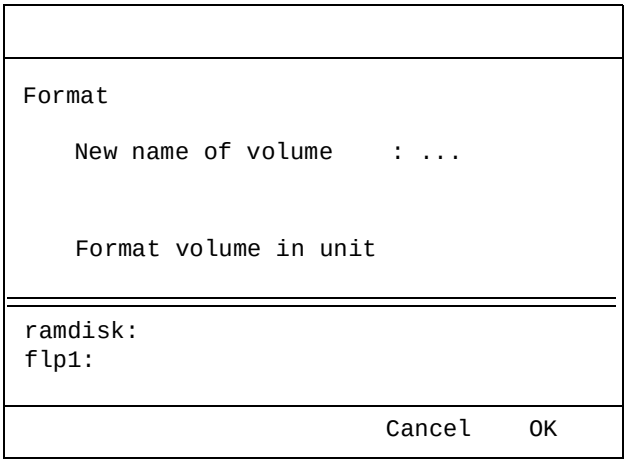


Figure 62 Dialog box for confirming the formatting of the diskette.

6. If you want to name the file, move the cursor to the upper part of the window.
7. Press Enter [↵].
8. Write the name. See section 14.2, Storing on diskette.
9. Move the cursor back to the lower part of the window.
10. Select flp1.
11. Start formatting the diskette by pressing ***OK***.
12. Wait until the dialog box disappears from the teach pendant.

14.2 Storing on diskette

1. Choose the **Program Instr** application (see Figure 63), if you are not already in it.

Figure 63 The Program window appears on the display.

2. Press the **File** menu key. The window in Figure 64 appears.

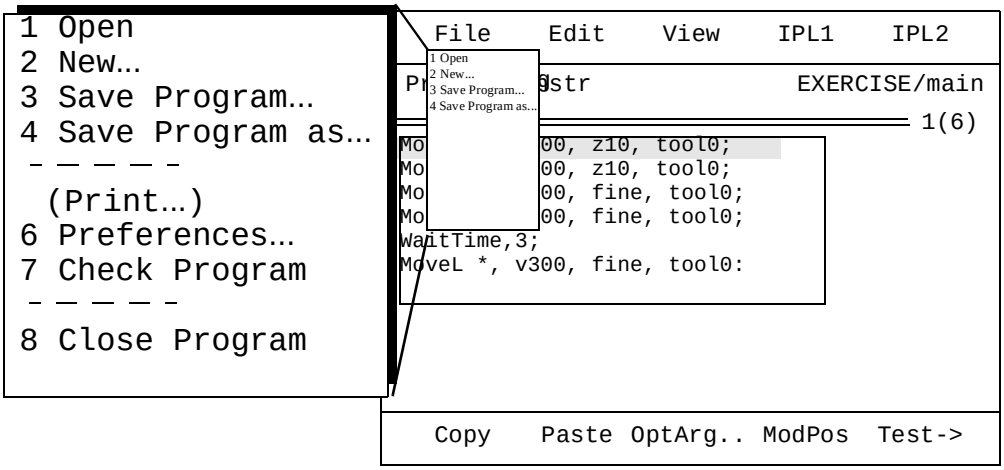


Figure 64 Commands in the File menu.

3. Select **File: Save Program as** and press **Enter**. (You could, alternatively, use the numeric keyboard to enter the number shown in front of the function name.) The dialog box in Figure 65 appears on the display.

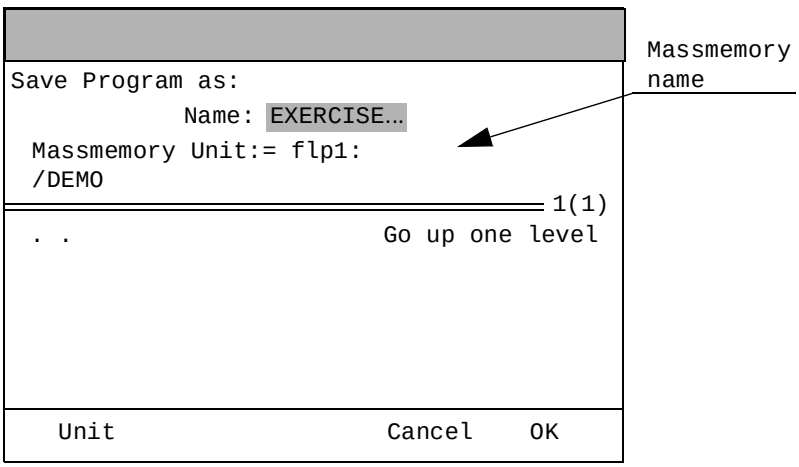


Figure 65 Dialog box for Save Program as.

4. Press ***Unit*** to choose the type of mass storage, if it is not already chosen; "flp1" should appear on the third line of the window (see Figure 65).

5. Press the Enter key to enter an optional name. The text-input dialog box in Figure 66 then appears.

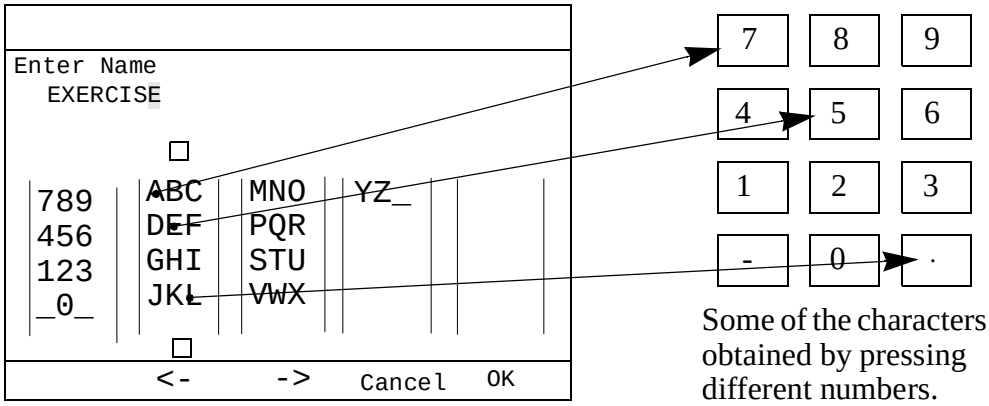


Figure 66 Window for entering text.

You can now see five groups of characters. Each group is represented on the numeric keyboard: the layout of the keyboard corresponds to the layout of the characters. You can move between the various groups using the <- and -> function keys (the selected group is indicated with squares both above and below it, see Figure 66).

Use the **Delete**  key to delete the name that is displayed or any errors you may type.

6. Now give the program a new name. When you have entered this, press ***OK.***
The window in Figure 67 appears.

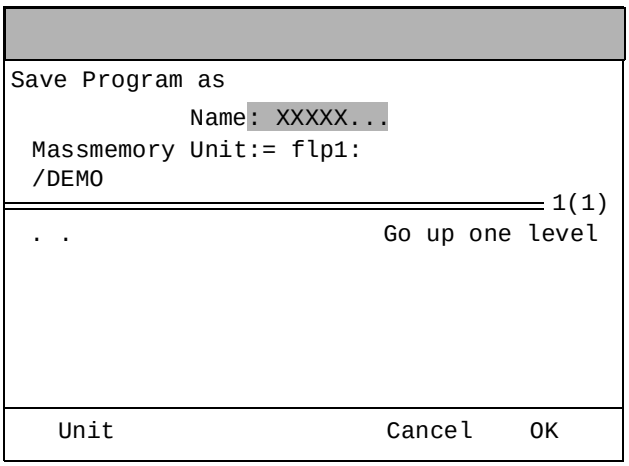


Figure 67 The Save Program as window.

7. Press ***OK***. A window like the one in Figure 68 appears.

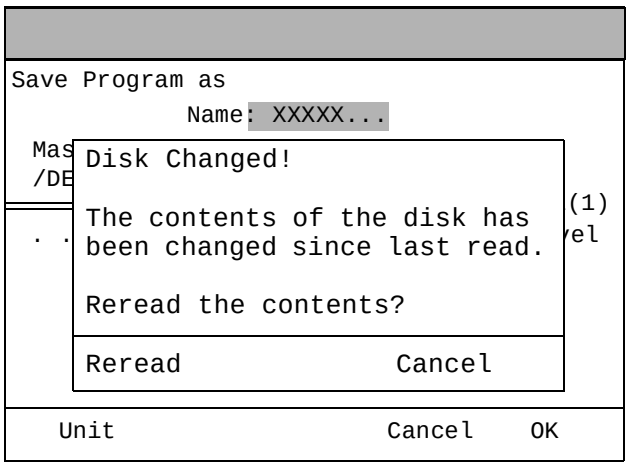


Figure 68 An alert box.

8. Press Enter. The window in Figure 69 appears.

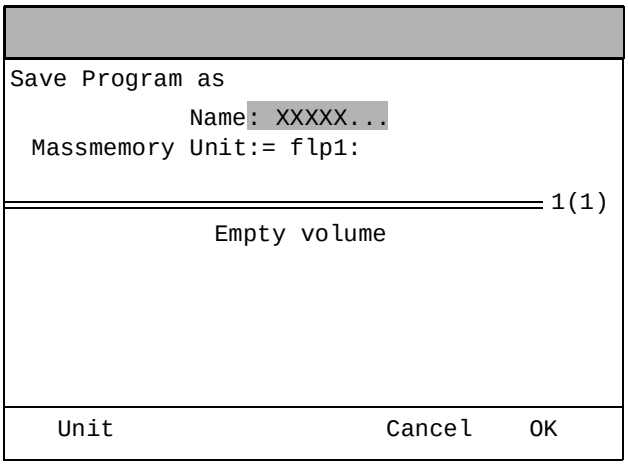


Figure 69 Window for storing the program on diskette.

9. The program is saved onto diskette when you press ***OK*** (see Figure 69).

10. Close the dialog box.

15 Printing Programs

15.1 Using a PC

It is also possible to print programs from a PC. Almost all word-processing programs can be used. The only requirement is that the computer can handle DOS-formatted diskettes.

1. Store the program on diskette.
2. Enter the program into the PC.
3. Print.

16 I/O Signals

This chapter describes how you can program an instruction which activates a digital output signal. After you have test run the program, you will manually open the I/O list and look at the signal in question.

16.1 Programming an I/O instruction

1. Select the **Program** window (see Figure 70).

Figure 70 The Program window key.

The window in Figure 71 appears on the display.

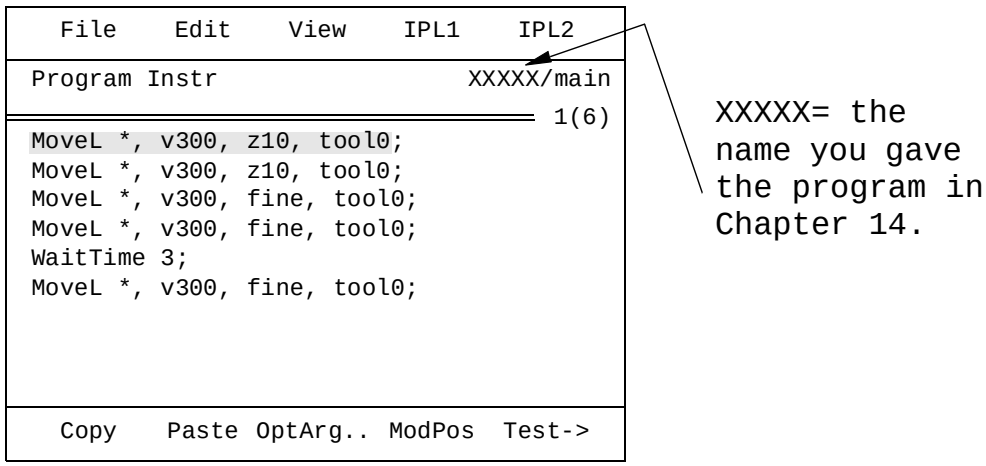


Figure 71 The Program Instruction window.

The new instruction (set an output) will be entered directly **under** the highlighted instruction. Select the third instruction in the program.

2. Select **IPL1: I/O**. The window in Figure 72 appears.

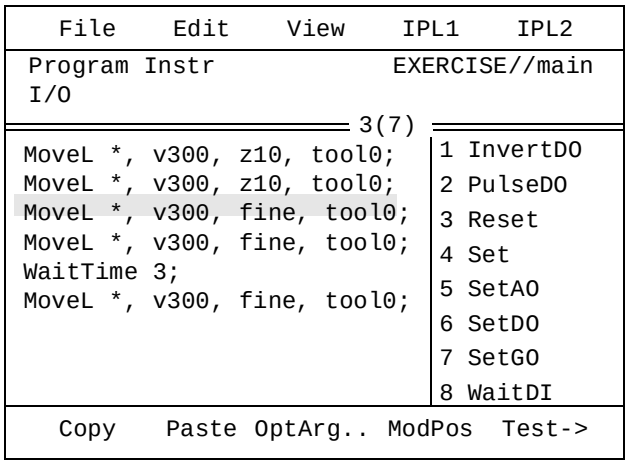


Figure 72 The pick list including the set instruction.

Select the function **Set** in the same way as you selected the WaitTime instruction in Chapter 13.

3. After you have selected the function **Set**, the dialog box in Figure 73 appears.

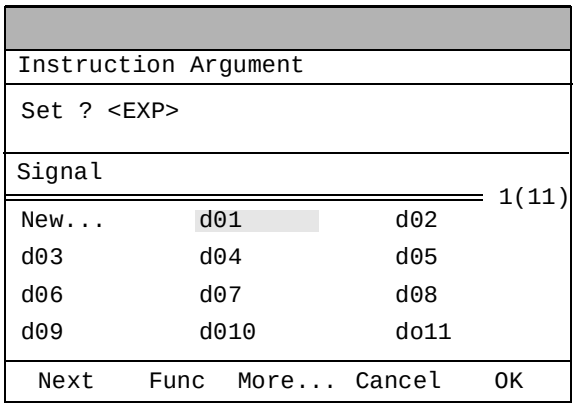


Figure 73 Dialog box for selecting a digital output signal.

4. Select “do4” in the list. Press ***OK***. The window in Figure 74 appears.

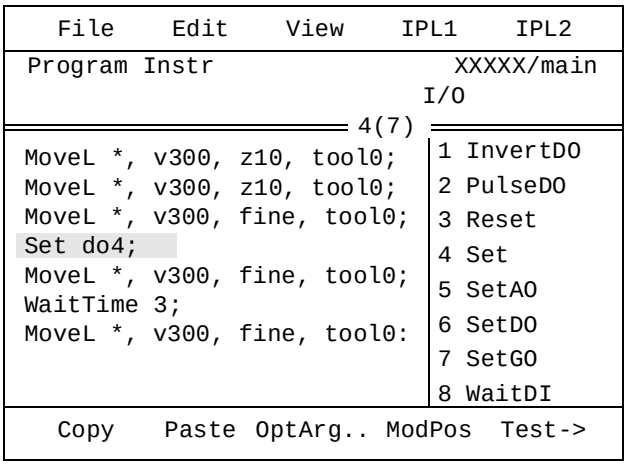


Figure 74 The “Set do4” instruction has been entered into the program.

5. Remove the pick list of instructions by pressing **Edit: IPLhide**.
   The window in Figure 75 will then appear.

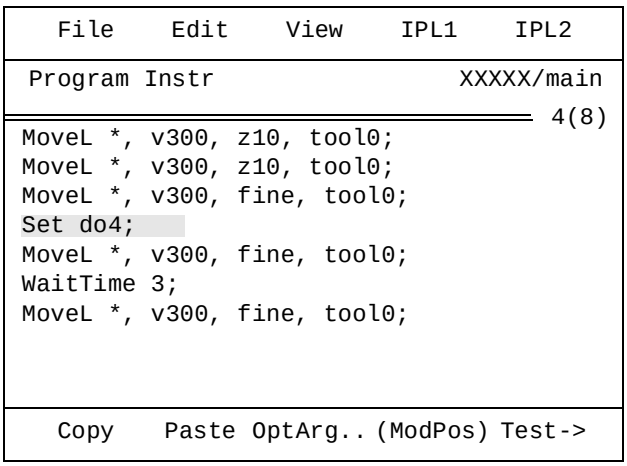
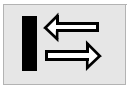


Figure 75 The Program Instruction window.

6. Test run the program using the **Test** function, one instruction at a time, so that the “Set do4” instruction can be read by the program.

7. You are now going to manually check the state of the signal.

8. Press the **Inputs/Outputs** window key (see Figure 76).



Inputs/Outputs window key

Previous page

Next page

Figure 76 The Manual I/O application.

Browse through the list displayed in the window with the keys as shown in Figure 76.

9. Find (using the up and down arrow keys) the “do4” signal in the IO list and highlight it.

10. Check its value.

    You can change the value using the function keys (***0*** / ***1***) that appear on the display each time an output signal is highlighted.

11. Change the value of the signal and then press the Prog application key.

12. Test run the program once more (Test) and check the value of the signal again.

INFORMATION

When you use lists from the **View** menu in the Manual I/O window, you can choose to have only input signals, only output signals, etc., shown on the display.

CONTENTS

Page

Starting up

1 Switching on the Power Supply

Before switching on the power supply, check that no-one is in the safeguarded space around the robot.

• Switch on the mains switch 

The robot hardware is then automatically checked. When the check is complete and if no errors have been detected, a message (see Figure 1) will be displayed on the teach pendant.

Figure 1 The welcome message after start-up.

In automatic mode, the Production window will appear after a few seconds.

The robot is started up with the same status as when the power was switched off. The program pointer remains unchanged and all digital outputs are set to the value before power off, or to the value specified in the system parameters. When the program is restarted, this is considered to be a normal stop - start:

- The robot moves back slowly to the programmed path (if there is a deviation) and then continues on the programmed path.
- Motion settings and data are automatically set to the same values as before power off.
- The robot will continue to react on interrupts.
- The mechanical units that was active before power off will automatically be activated at program start.
- The arc welding and spot welding processes are automatically restarted. But if a change of weld data has just been executed, this new data will be activated too early on the seam.

Limitations:

- All files and serial channels are closed (this can be handled by the user program).

- All analogue outputs are set to 0 and the Soft servo/Tune servo is set to default values (can be handled by the user program).
- WeldGuide cannot be restarted.
- Independent axes cannot be restarted.
- If the power failure occurs during a movement in an interrupt routine or error handler, the restart of the path is not possible.
- If the program execution is in a part when the CPU is very busy, there is a small chance that there is not enough time to make a proper close down at power failure. The robot will in this case tell the user that a restart is not possible.

1.1 Errors on start-up

During the entire start-up sequence, the robot functions are checked extensively. If an error occurs, it is reported as a message in plain text on the teach pendant, and recorded in the robot's event log. For more information on troubleshooting, see the Product Manual.

2 The Operator's Panel

The functions of the operator's panel are described in Figure 2.

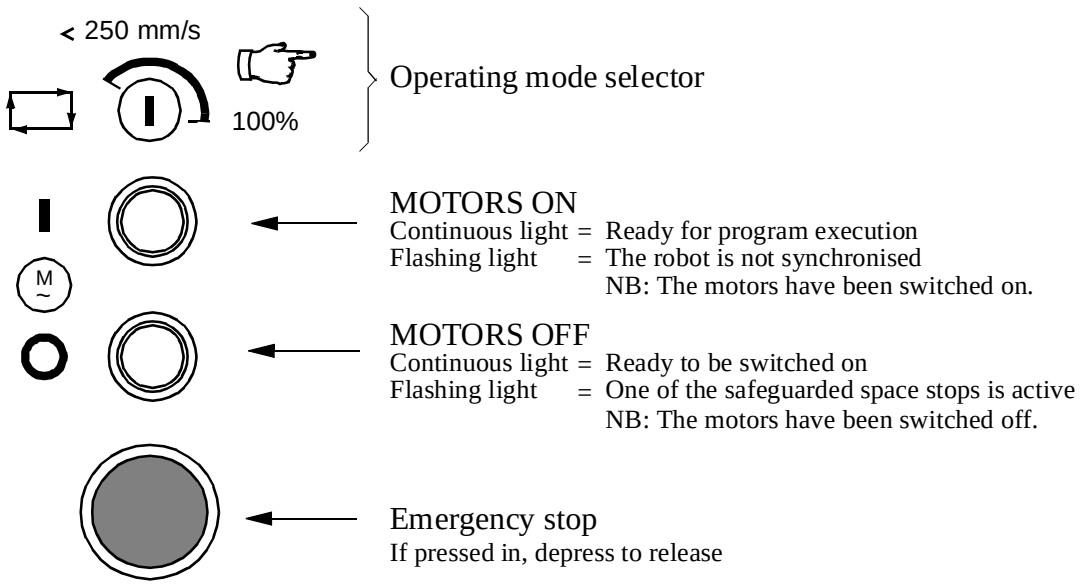


Figure 2 The operator's panel is located on the front of the cabinet.

3 Selecting the Operating Mode

The operating mode is selected using the operating mode selector.

3.1 Automatic mode (production mode)

When the robot is in the automatic operating mode, it is essential that nobody enters the safeguarded space around it. Carelessness may cause personal injury.

• Turn the key to .

Automatic mode is used when running complete programs in production operation. In this mode, the enabling device on the teach pendant is disconnected and the functions used to edit programs are locked.

3.2 Manual mode with reduced speed (programming mode)

• Turn the operating mode selector to < 250 mm/s .

If the hold-to-run function is active (the function is available by means of a system parameter), program execution will stop as soon as you release the Start key on the teach pendant Version 1, and the Hold-to-run key on the teach pendant Version 2. The two versions of the teach pendant are described in *The Teach Pendant* on page 7.

Manual mode with reduced speed is used when programming and when working in the robot working space. In this mode, external units cannot be remotely controlled.

3.3 Manual mode with full speed (testing mode)

In Manual 100% mode, the robot moves at full speed. This operating mode may only be used by trained personnel. Carelessness may cause personal injury.

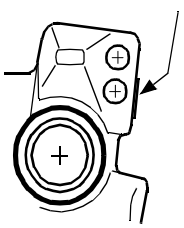
• Turn the operating mode selector to 100% .

The hold-to-run function is now active, i.e. program execution will stop as soon as you release the Start key on the teach pendant.

Manual mode with full speed is only used when testing the robot program at full speed. In this mode, external units cannot be remotely controlled.

4 Switching the Power Supply to the Motors On/Off

• In automatic mode, press the Motors On/Motors Off button on the operator's panel.



• In manual mode, turn to MOTORS ON mode by pressing the enabling device on the teach pendant halfway in.

If the enabling device is released and pressed again within half a second, the robot will not return to the MOTORS ON state. If this happens, first release the enabling device, then push it halfway in again.

5 Synchronizing External Axes

When the power to the motors is switched on and an external axis is not synchronized, a dialog box will appear, which you can use to synchronize that axis (see Figure 3).

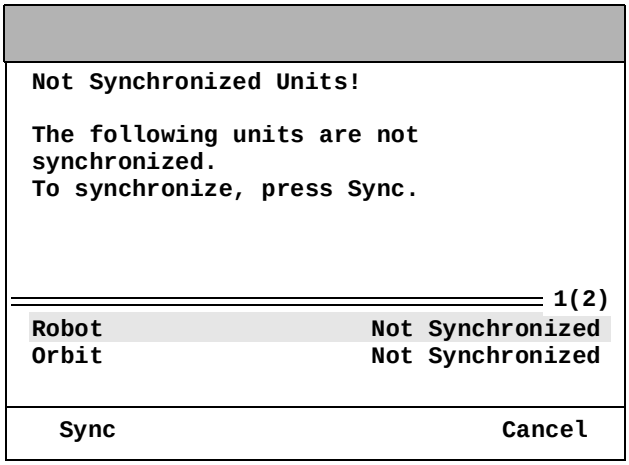


Figure 3 Unsynchronized mechanical units must be synchronized before starting the program.

• Select the required unit, press the enabling device and press the function key ***Sync***.

When the unit has been synchronized the robot automatically goes to the MOTORS OFF state. If there are several units to be synchronized, the robot must first be turned to MOTORS ON. In manual mode, this is done by releasing the enabling device and pressing it in again.

6 Emergency Stops

6.1 Activating the emergency stop button

Emergency stop buttons are located on the operator's panel and on the teach pendant. There are often other ways of activating an emergency stop, but these depend on the robot installation.

When the emergency stop button is activated, the power supply to the motors is shut off and program execution stops.

6.2 Resetting after an emergency stop

• Fix the problem that caused the emergency stop.

• Reset the emergency stop button by pressing the MOTORS OFF button.

7 The Teach Pendant

There are two versions of the teach pendant. Version 2 is outlined in Figure 4 and Version 1 in Figure 5.

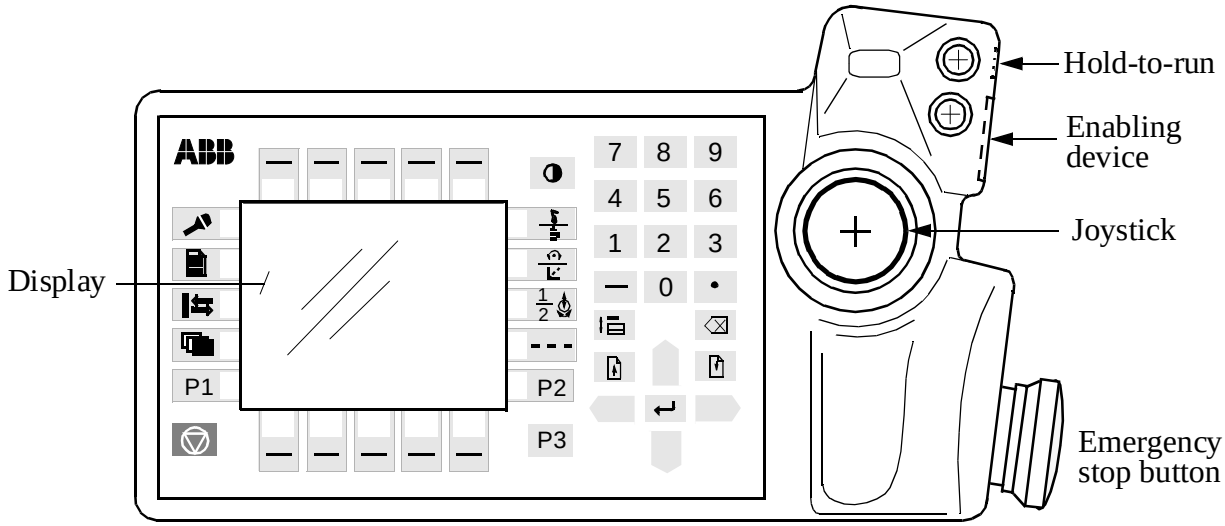


Figure 4 The teach pendant is used to operate the robot (Teach pendant Version 2).

Jogging: Used to jog the robot.

Program: Used to program and test.

Inputs/Outputs: Used to manually operate the input and output signals connected to the robot.

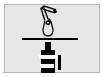
Misc.: Miscellaneous; other windows, i.e. the System Parameters, Service, Production and File Manager windows.

Stop: Stops program execution.

Contrast: Adjusts the contrast of the display.

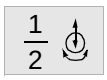
Menu keys: Press to display menus containing various commands.

Function keys: Press to select the various commands directly.



Motion Unit: Press to jog the robot or other mechanical units.

Motion Type: Press to select how the robot should be jogged, reorientation or linear.



Motion Type: Axis by axis movement. 1 = axis 1-3, 2 = axis 3-6

Incremental: Incremental jogging on/off

List: Press to move the cursor from one part of the window to another (normally separated by a double line).

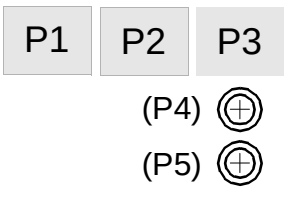
Previous/Next page: Press to see the next/previous page.

Delete: Deletes the data selected on the display.

Enter: Press to input data.

Up and Down arrows: Press to move the cursor up or down.

Left and Right arrows: Press to move the cursor to the left or right.



User defined keys: How to define these, see Chapter 12, *System Parameters*
On some teach pendants these keys are called **F1**, **F2** etc.

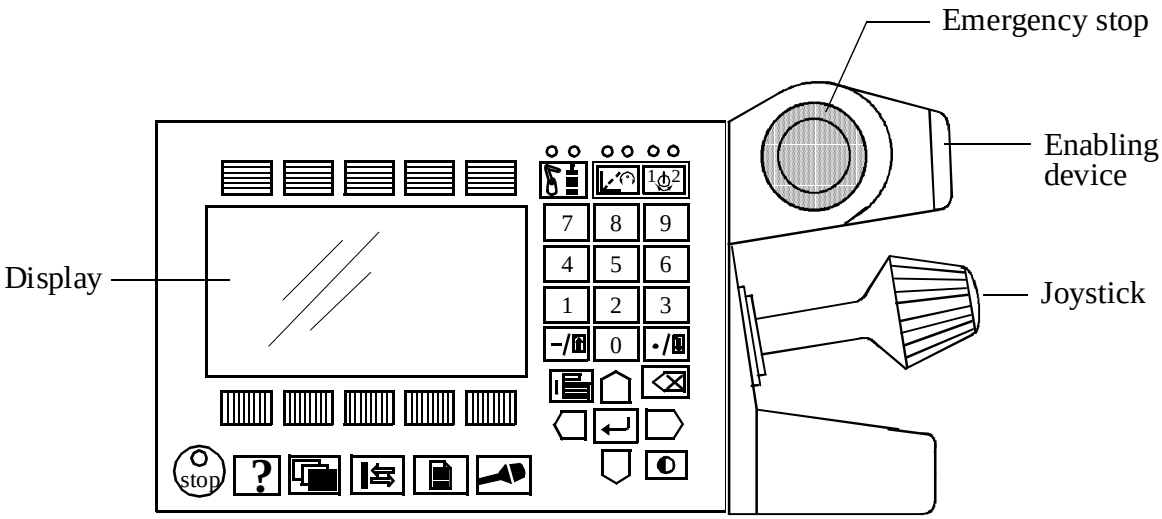
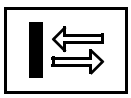


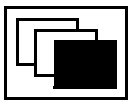
Figure 5 The teach pendant is used to operate the robot (Teach pendant Version 1).

Jogging: Used to jog the robot.

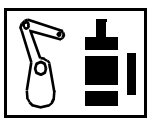
Program: Used to program and test.



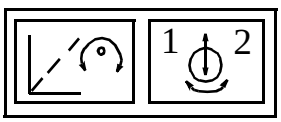
Inputs/Outputs: Used to manually operate the input and output signals connected to the robot.



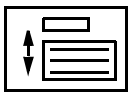
Misc.: Miscellaneous; other windows, i.e. the System Parameters, Service, Production and File Manager windows.



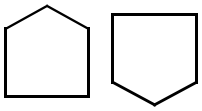
Motion Unit: Press to jog the robot or other mechanical units.



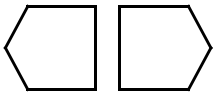
Motion Type: Press to select how the robot should be jogged.



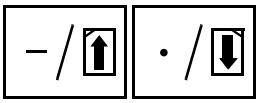
List: Press to move the cursor from one part of the window to another (normally separated by a double line).



Up and Down arrows: Press to move the cursor up or down.

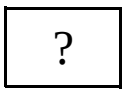


Right and Left arrows: Press to move the cursor to the right or left.

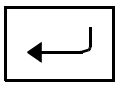


Previous/Next page: Press to see the next/previous page.

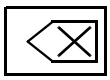
Stop: Stops program execution.



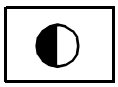
Help: Press to call up a help menu directly to the display (no information in this version of the robot).



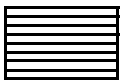
Enter: Press to input data.



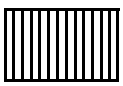
Delete: Deletes the data selected on the display.



Contrast: Adjusts the contrast of the display.



Menu keys: Press to display menus containing various commands.



Function keys: Press to select the various commands directly.

7.1 Entering text using the teach pendant

When naming files, routines, data, etc., text can be entered using the teach pendant. As there is no character keyboard available, the numeric keyboard is used in a special way (see Figure 6).

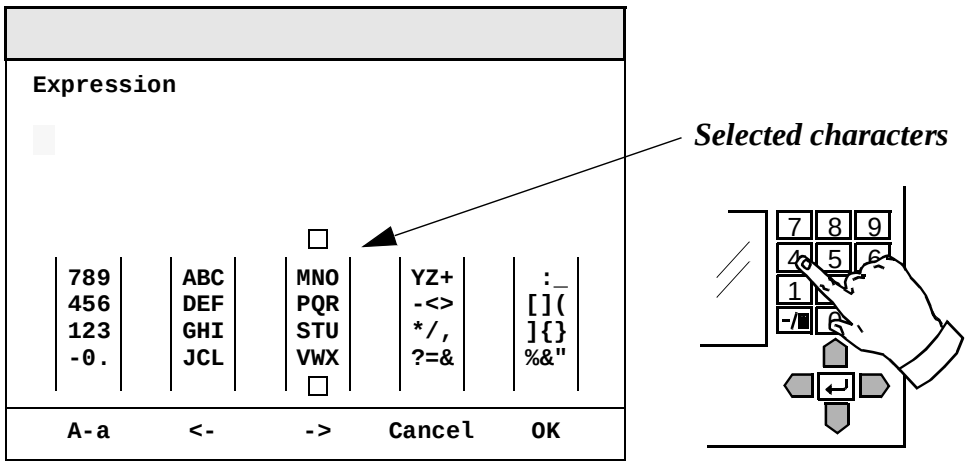


Figure 6 The dialog box used for entering text.

The keys on the numeric keyboard correspond to the selected characters on the display.

- Select a group of characters by pressing the function key -> or <-.
- Press the corresponding key on the numeric keyboard. If the third group is selected (as shown in Figure 6), 7 corresponds to M, 8 to N, 9 to O, etc.
- Move the cursor to the right or left using ArrowLeft ⟨ or ArrowRight ⟩.
- Delete the character in front of the cursor by pressing Delete ⌫.
- Switch between upper and lower case letters by pressing ***A-a***.
- When you have finished entering text, press ***OK***.

8 Error Management

If an error occurs, an error message will be displayed in plain language on the teach pendant (see Figure 7). If several errors occur simultaneously, the error that occurred first will be selected.

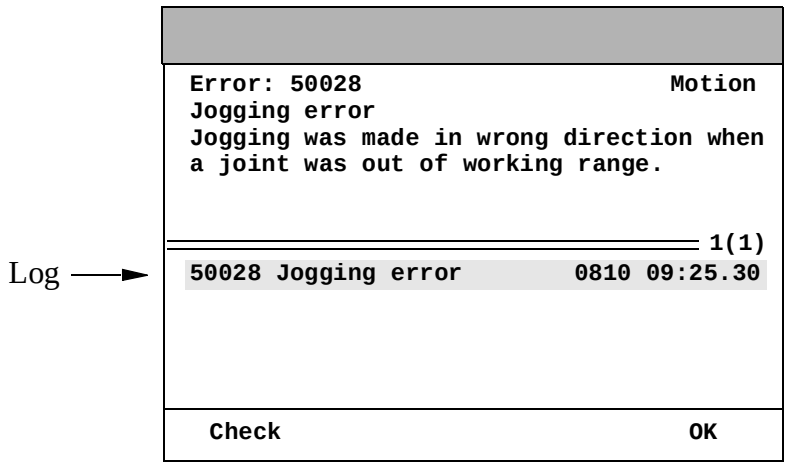


Figure 7 An error message is displayed in plain language as soon as an error occurs.

All errors and status changes are also registered and time-stamped in a log. For more detailed information on these logs, see Service in Chapter 14 of this manual.

8.1 Confirming an error message

• Press ***OK***.

The window displayed before the error occurred will be displayed once more. If you want to view an error message later on, you can find it in the log (see Service in Chapter 14 of this manual).

8.2 Calling up suggestions on how to correct an error

• Press ***Check***.

Information about possible corrective measures is displayed, along with the reason for the error (see Figure 8).

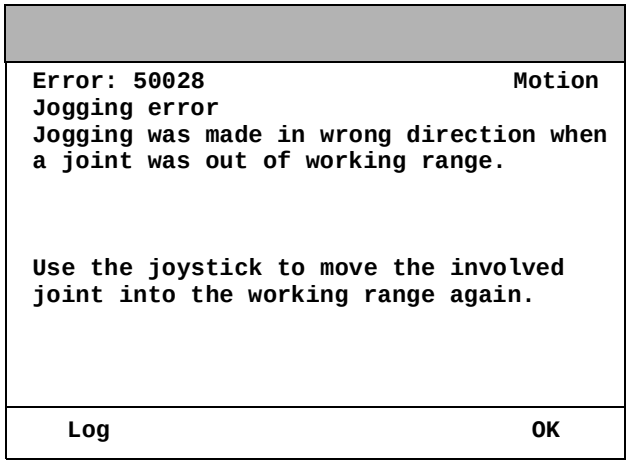


Figure 8 Suggestions on how to correct an error.

• Press ***Log*** to display the log instead of the check list.

8.3 Acknowledging warning messages

Sometimes, a warning or information message will be displayed. This message is displayed in the form of a minimised alert box that conceals only part of the previous window.

• Acknowledge the message by pressing Enter ↵.

CONTENTS

Jogging

Jogging

1 General

A joystick is used to jog the robot. It has three degrees of freedom, which means that you can move the robot in three different directions simultaneously. The robot speed is proportional to the joystick deflection, the greater the joystick deflection, the higher the speed (but not faster than 250 mm/s).

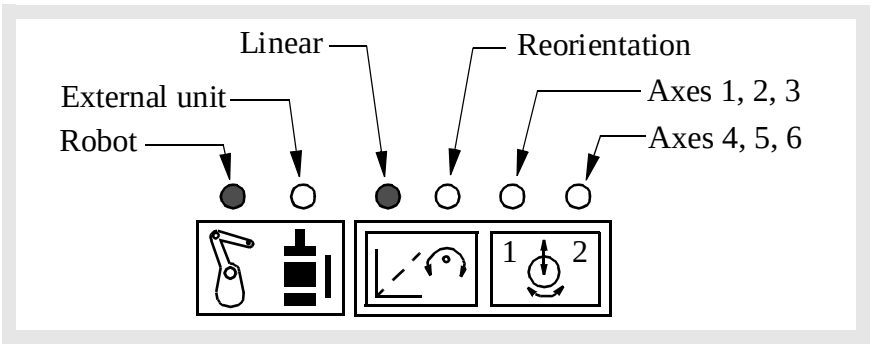
The joystick can be used irrespective of which window is open at the time. You cannot, however, jog the robot in the following situations:

- When the robot is in automatic mode .
- When the robot is in the MOTORS OFF state.
- When program execution is in progress.

If any axis is outside its working range, it can only be jogged back into its working range.

The function of the joystick can be read from and changed in the Jogging window. Some of the settings can also be changed directly using the motion keys on the teach pendant. (The different versions are illustrated in Figure 1).

Teach pendant Version 1



Teach pendant Version 2

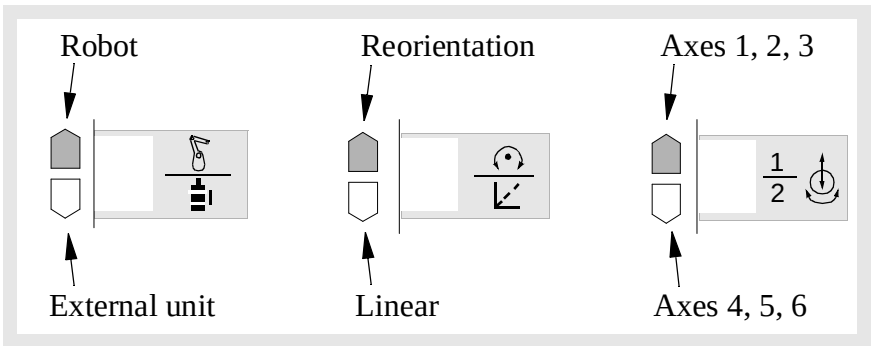


Figure 1 The LEDs above the motion keys show the current settings.

The robot or external unit will start to move immediately you move the joystick. Make sure that no one is standing in the safeguarded space around the robot and also that the motion settings for jogging are correctly set. Carelessness can injure someone or damage the robot or other equipment.

1.1 The Jogging window

• Press the Jogging key  to open the window.

The window displays the current motion settings for jogging and the current position of the robot. See the example in Figure 2.

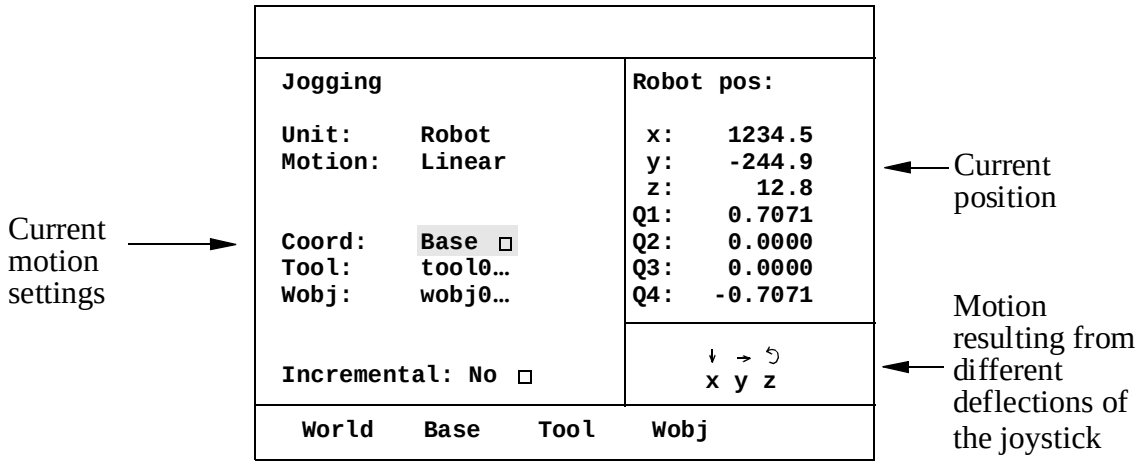


Figure 2 Define the various jogging settings in the Jogging window.

1.2 Reading the current position

The current position of the robot is displayed in the Jogging window (see Figure 2).

In *Linear* or *Reorientation* motion types, the position and orientation of the tool in relation to the coordinate system of the chosen work object is displayed (regardless of the type of coordinate system used).

In *Axis-by-Axis* motion type with *Robot* as the unit, the positions of the robot axes are displayed in degrees related to the calibration position of the respective axis.

When an external unit is moved, the position of the axes is displayed. In the case of linear axes, the position is displayed in mm related to the calibration position. For rotating axes, the position is displayed in degrees related to the calibration position.

When a unit is unsynchronised, no position is displayed.

1.3 How moving the joystick affects movements

The field that indicates the various deflections of the joystick displays how the principal joystick directions are linked to axes or coordinate directions. See the example in Figure 3.

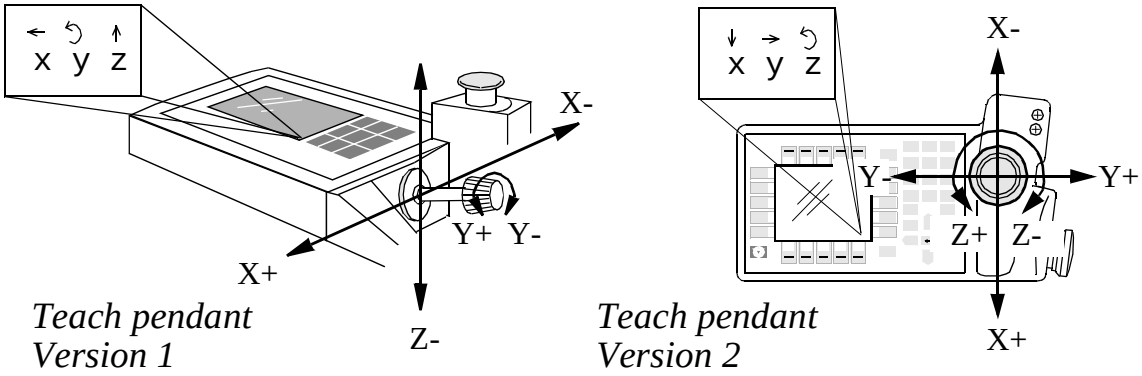
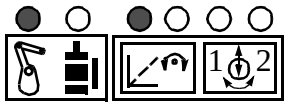
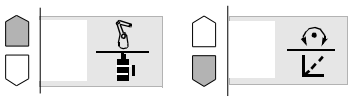


Figure 3 The direction of movements associated with each joystick deflection is displayed in the Jogging window.

2 Jogging the Robot

2.1 Jogging the robot along one of the base coordinate axes

- Set the keys  or  respectively, to jog the robot in a straight line.
- Select the field ***Coord*** (see Figure 4).
- Press the function key ***Base***.

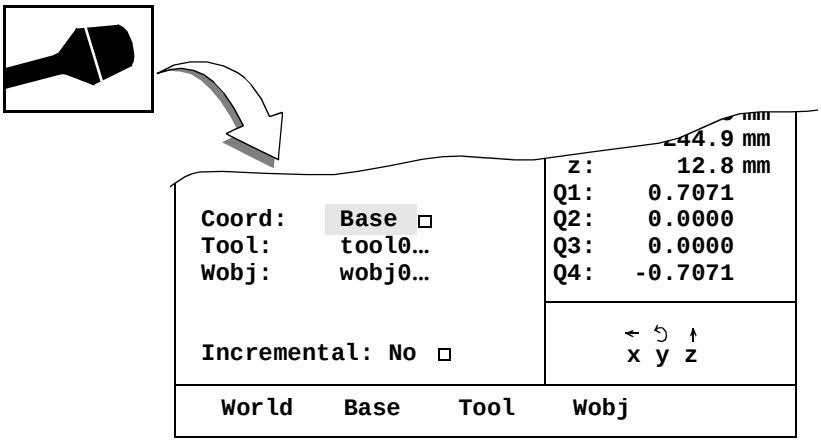


Figure 4 Specify the coordinate system in the Jogging window.

The robot will move the TCP along the base coordinate axes (see Figure 5).

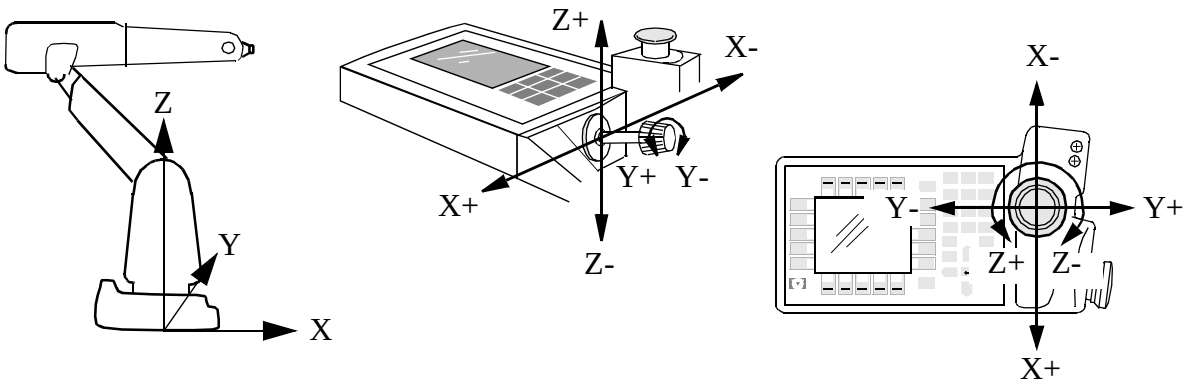
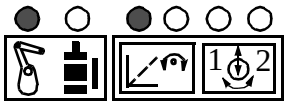
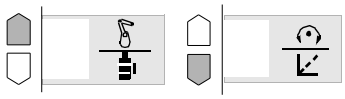


Figure 5 Linear movement in the base coordinate system.

2.2 Jogging the robot in the direction of the tool

• Set the keys  or 

respectively, to jog the robot in a straight line.

• Select the field ***Coord*** (see Figure 6).

• Press the function key ***Tool***.

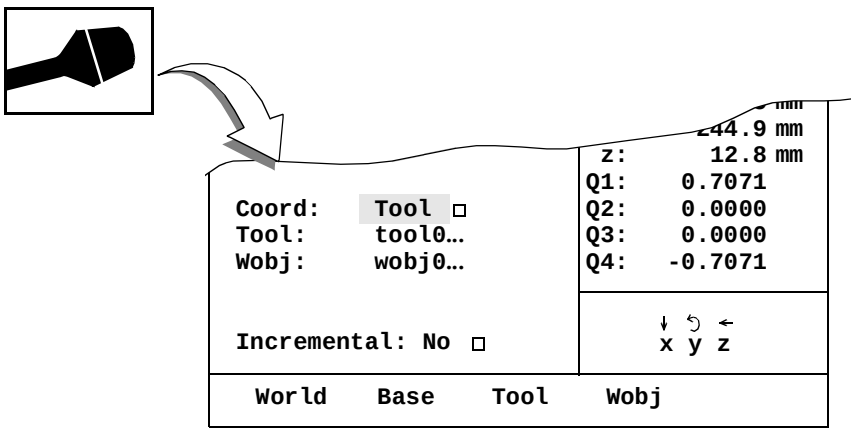


Figure 6 Specify the coordinate system in the Jogging window.

The tool that was last used when jogging the robot or last used for program execution is automatically chosen (see Figure 7).

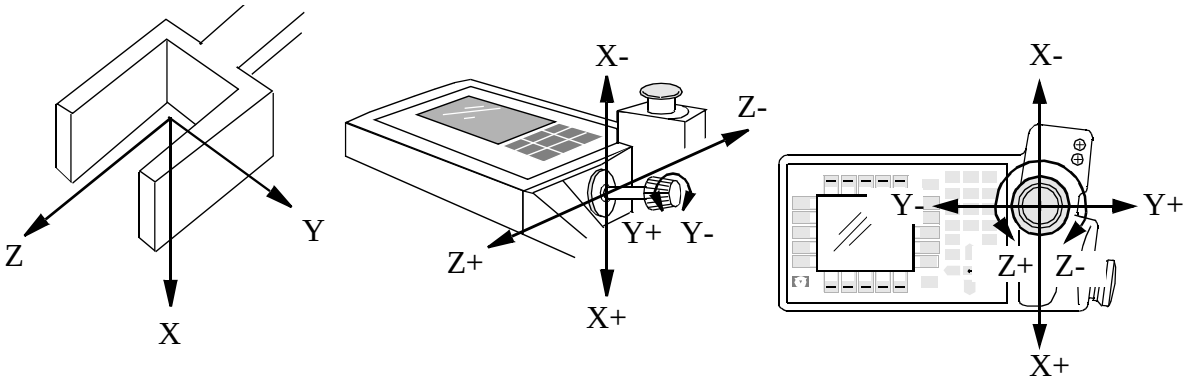


Figure 7 Linear movement in the tool coordinate system.

If you want to change the tool:

• Select the field ***Tool*** (see Figure 8).

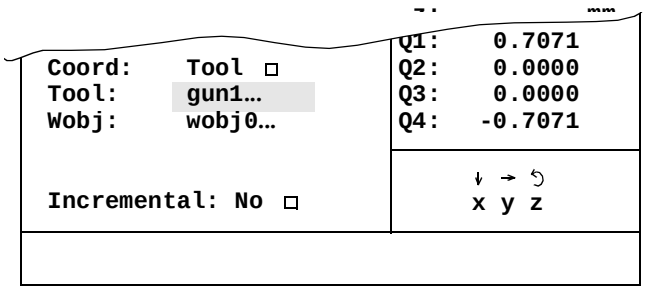


*Figure 8 Choose a tool by selecting the field **Tool**.*

• Press Enter ↵ .

• Select the desired tool from the dialog box which subsequently appears on the display. (*Tool0* in the dialog box corresponds to the centre of the mounting flange.)

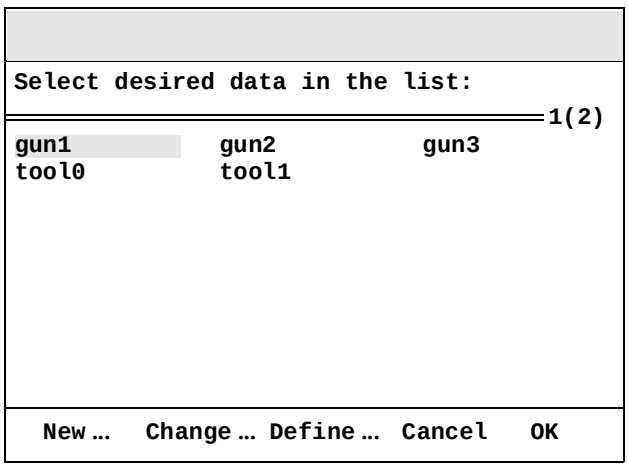


Figure 9 Changing or adding a tool.

You can create a new tool as follows:

• Press ***New.***

You can change the values of a tool as follows:

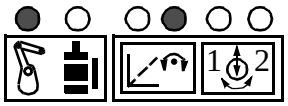
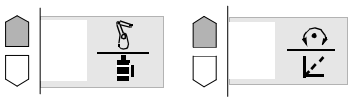
• Press

- ***Change*** to input the value manually
- ***Define*** to use the robot to measure up the tool coordinate system.

For more information see Chapter 10 Calibration.

• Press ***OK*** to terminate the dialog.

2.3 Reorienting the tool

• Set the keys  or 

respectively, to reorient the tool.

The tool is reoriented about the axes of the coordinate system that was chosen. The TCP of the chosen tool will not move (see Figure 10).

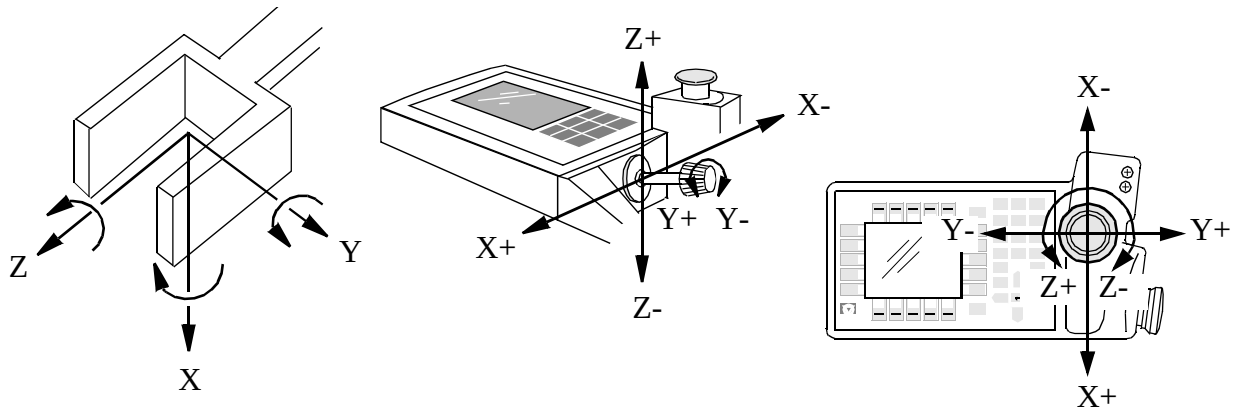


Figure 10 Reorientation about the tool coordinate system's axes.

2.4 Aligning a tool along a coordinate axis

The Z-direction of a tool can be aligned along a coordinate axis of a chosen coordinate system. The angle between the tool's Z-direction and the coordinate axes determines which coordinate axis the tool should be aligned along; the axis closest to the tool's Z-direction will be used (see Figure 11).

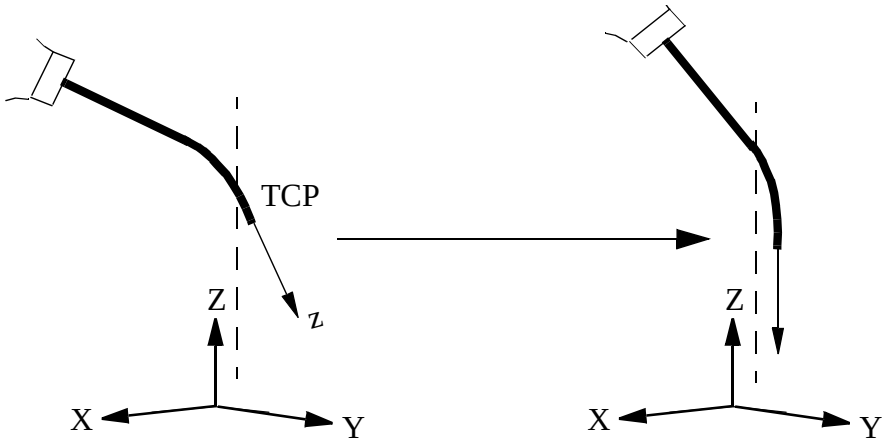


Figure 11 The tool is aligned along the Z-axis.

Adjust the direction of the tool manually so that it is close to the desired direction.

• Choose **Special: Align**

A dialog box appears displaying the coordinate system used for alignment (see Figure 12).

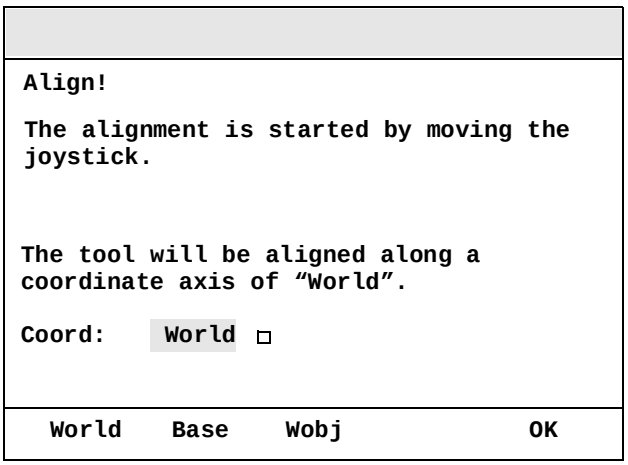


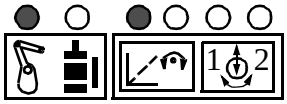
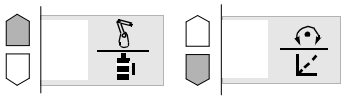
Figure 12 The dialog box for aligning the tool.

If you want to change the coordinate system, press any of the function keys ***World, Base*** or ***Wobj.***

• To start the alignment, press the enabling device and move the joystick. The joystick is used to adjust the speed. The robot will automatically stop as soon as it reaches the desired position.

• Press ***OK*** to confirm.

2.5 Jogging the robot in the direction of the work object

• Set the keys  or 

respectively, to jog the robot in a straight line.

• Select the field ***Coord*** (see Figure 13).

• Press the function key ***Wobj***.

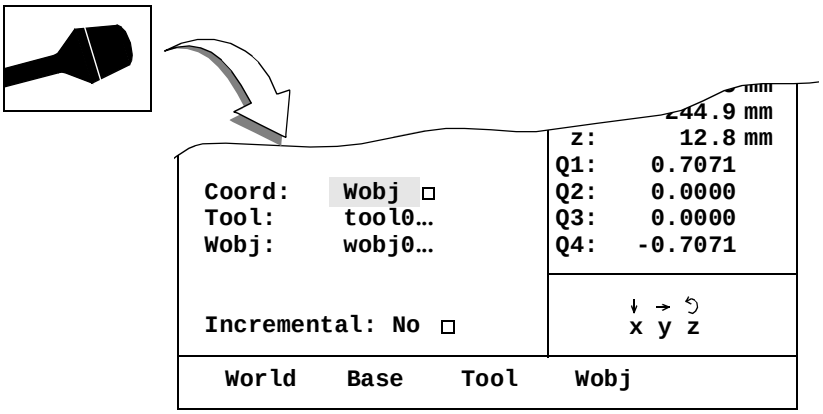


Figure 13 Specify the coordinate system in the Jogging window.

The work object that was last used when jogging the robot or last used for program execution is automatically chosen.

If you want to change the work object:

• Select the field ***Wobj*** (see Figure 14).

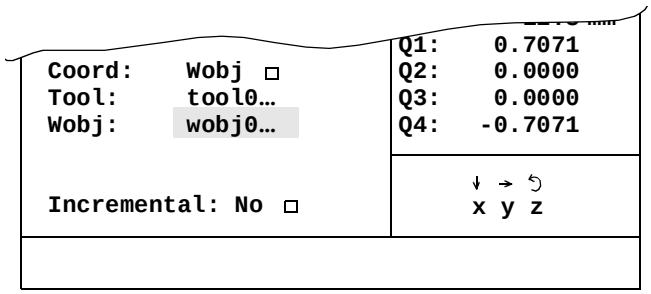


*Figure 14 Choose a work object by selecting the field **Wobj**.*

• Press Enter ↵ .

• Select the desired work object from the dialog box which subsequently appears on the display. (*Wobj0* in the dialog box corresponds to the world coordinate system.)

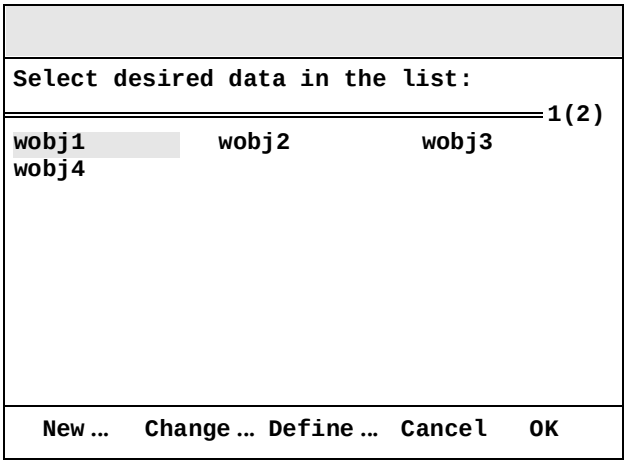


Figure 15 Changing or adding a work object.

You can create a new work object as follows:

• Press ***New.***

You can change the values of a work object as follows:

• Press
- ***Change*** to input the value manually
- ***Define*** to use the robot to measure up the object coordinate system.

For more information see chapter 10 Calibration.

• Press ***OK*** to terminate the dialog.

The robot will move along the axes of the object coordinate system (see Figure 16).

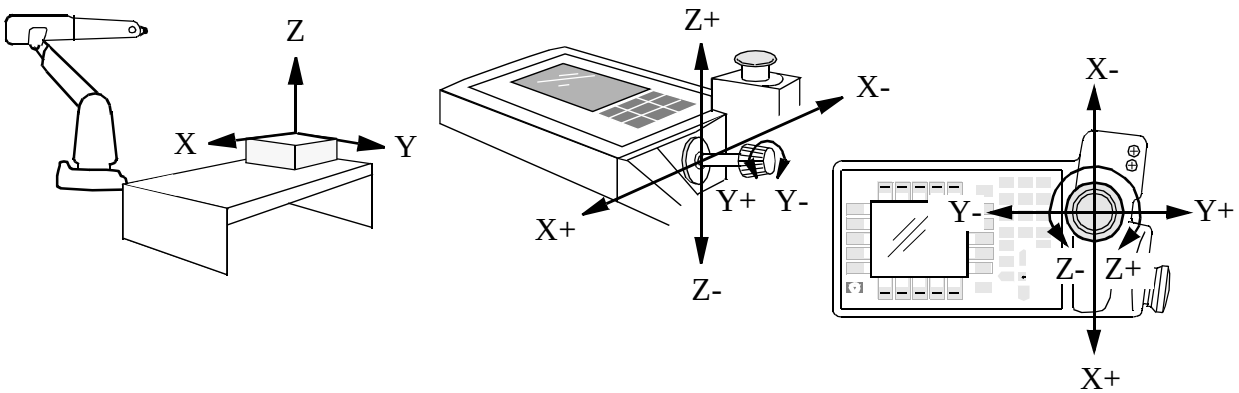
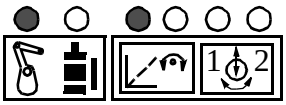
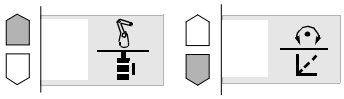


Figure 16 Linear movement in the object coordinate system.

2.6 Jogging the robot along one of the world coordinate axes

• Set the keys  or  respectively, to jog the robot in a straight line.

• Select the field ***Coord*** (see Figure 17).

• Press the function key ***World***.

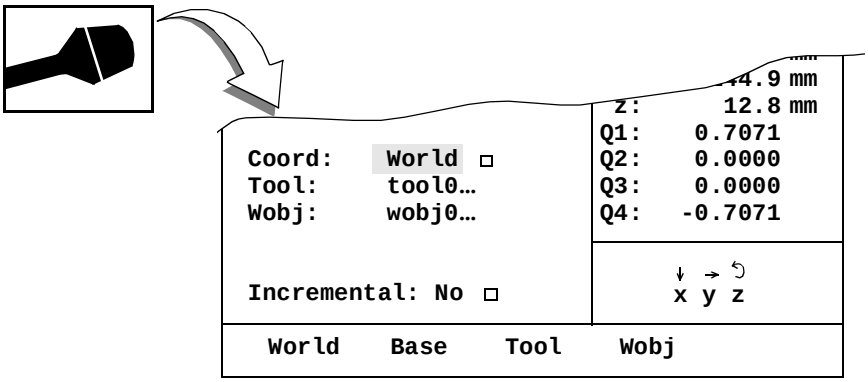


Figure 17 Specify the coordinate system in the Jogging window.

The robot will move the TCP along the world coordinate axes (see Figure 18).

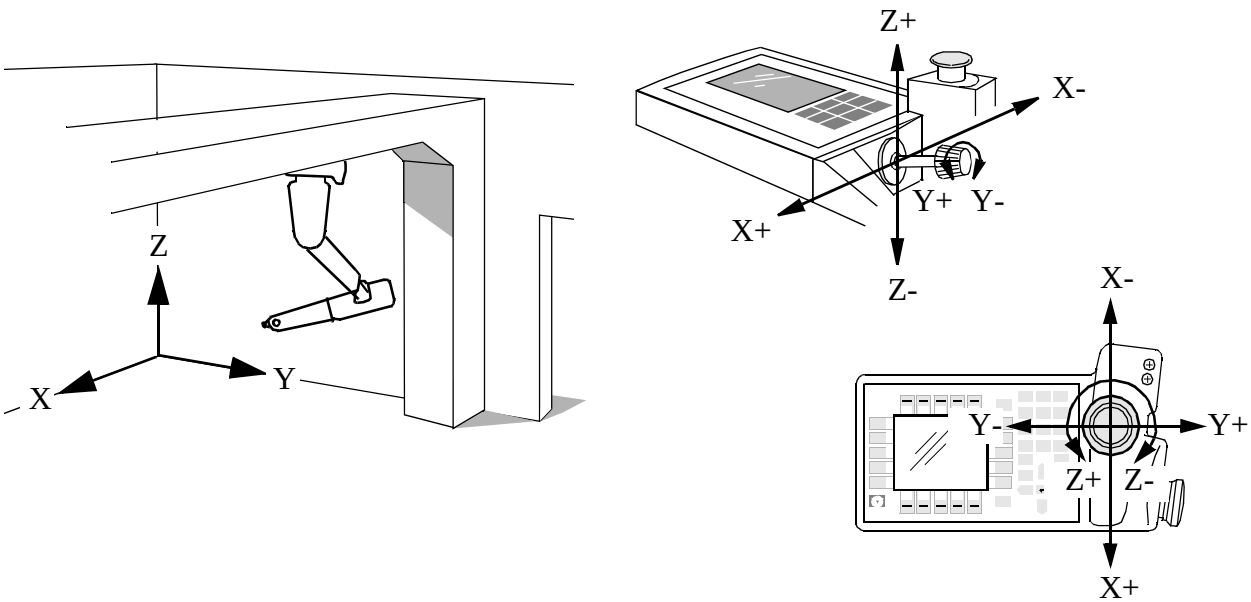


Figure 18 TCP movement is independent of the robot mounting.

2.7 Using a stationary tool

If a stationary TCP is active, the work object will move in accordance with the chosen coordinate system.

2.8 Jogging the robot axis-by-axis

• Choose axis-by-axis movement by setting the motion keys (see Figure 19).

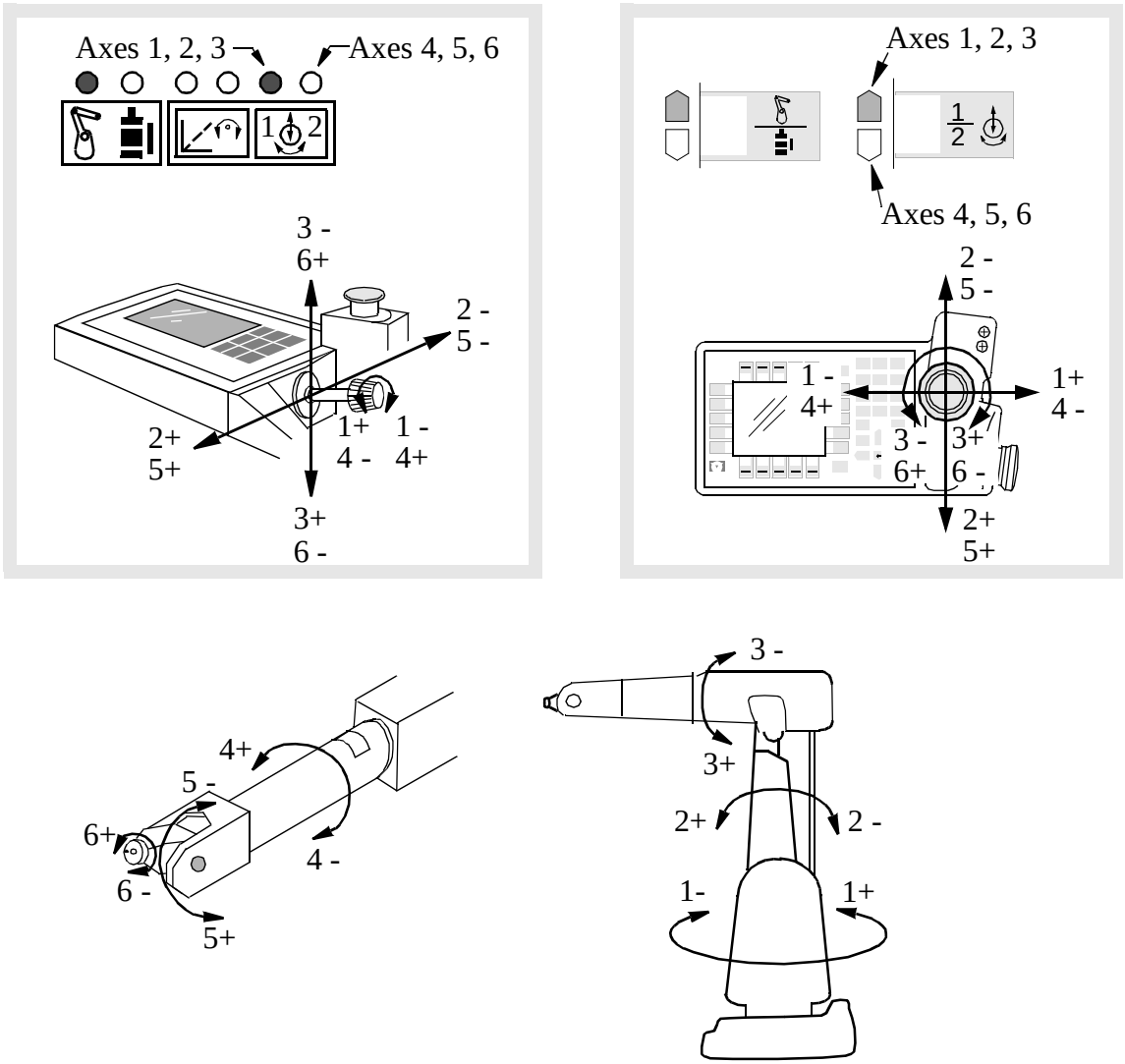


Figure 19 Specify the axes you want to move by setting the keys as above.

Only the axis affected by the joystick deflection moves, which means that the TCP does not move linearly.

2.9 Incremental movement

Incremental movement is used to adjust the position of the robot exactly. This means that each time the joystick is moved, the robot moves one step (increment). If the joystick is deflected for one or more seconds, a sequence of steps, at a rate of 10 steps per second, will be generated as long as the joystick is deflected.

• Select the field ***Incremental*** (see Figure 20).

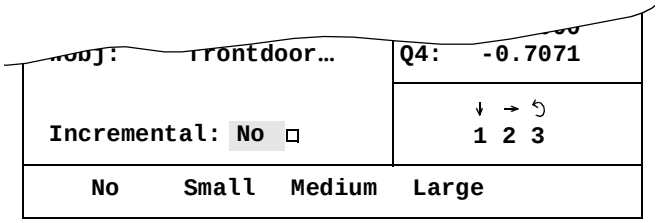


Figure 20 Specify the incremental step size in the field Incremental.

• Specify the size of the steps using the function keys.

- ***No:*** Normal (continuous) movement
- ***Small:*** Approx. 0.05 mm or 0.005 degrees per joystick deflection
- ***Medium:*** Approx. 1 mm or 0.02 degrees per joystick deflection
- ***Large:*** Approx. 5 mm or 0.2 degrees per joystick deflection

With the teach pendant Version 2 you can use the key --- to turn incremental movement on and off.

2.10 Jogging an unsynchronised axis

If the robot or an external unit is unsynchronised, it can only move using one motor at a time.

The working range is not checked, which means that the robot can be moved until it is stopped mechanically.

3 Jogging External Axes

3.1 Choosing external units

If you wish to use more than one external unit, those units must be chosen from the Jogging window.

• Set the motion key  or  on the external key.

• Select the field ***Unit*** (see Figure 21).

• Using the function keys, choose a unit.

If you are using more than 5 external units and you cannot find the one you want in the function key dialog, press Enter ↵ and select the desired unit from the new dialog.

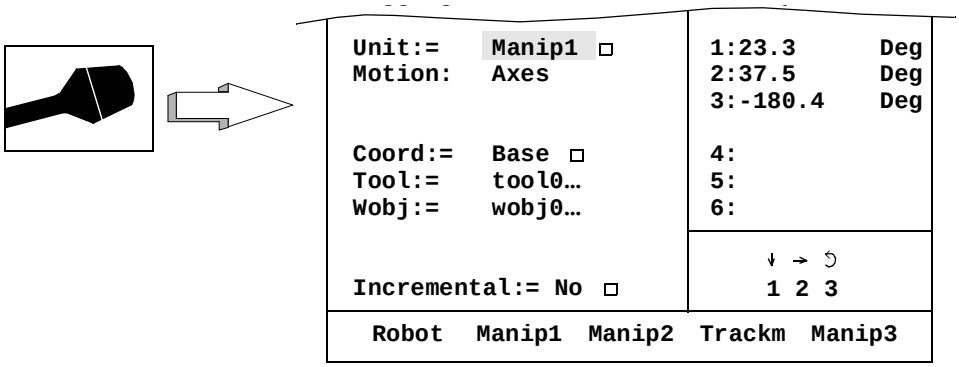
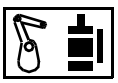


*Figure 21 Specify the unit to be jogged in the **Unit** field.*

From this stage onwards, the key  or  respectively, can be used to toggle between the external unit that was last chosen and the robot.

3.2 Jogging external units axis-by-axis

- Choose the desired axis group using the motion keys (see Figure 22). If more than one external unit is used, see *3.1 Choosing external unit*s.

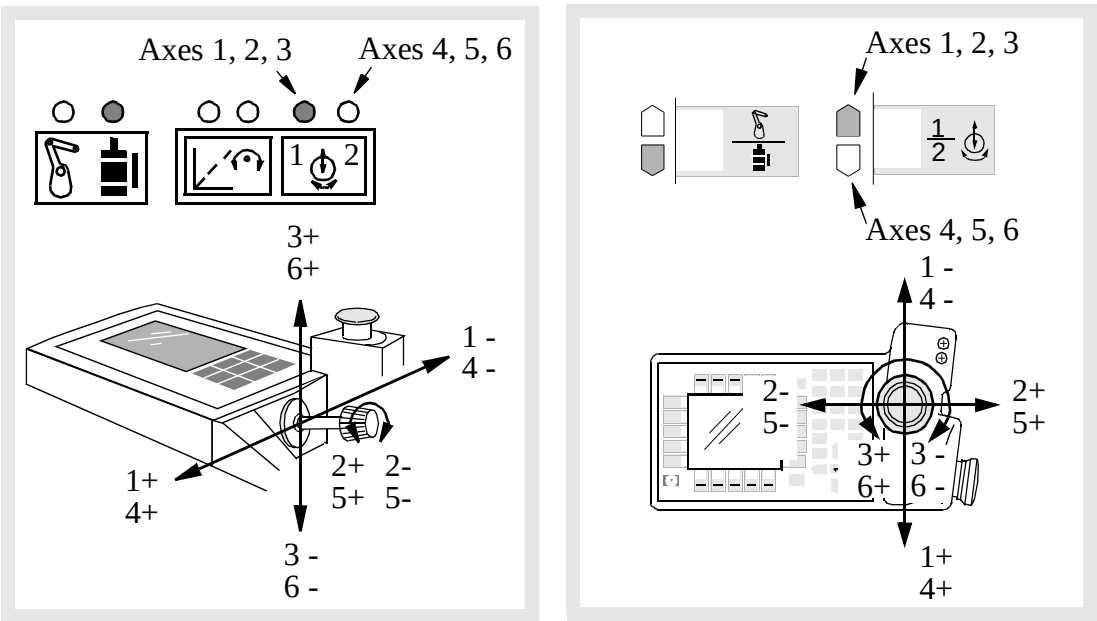


Figure 22 Specify the external axes you want to move by setting the keys as above.

3.3 Jogging external units coordinated

If an axis is coordinated with the robot (defined by the chosen work object), the robot also moves when it is jogged. The TCP, however, will not move in relation to the work object.

If you want to jog the unit uncoordinated, choose a work object which is not connected to a coordinated unit, e.g. *wobj0*, in the field ***Wobj***.

Jogging

CONTENTS

Page

Inputs and Outputs

1 General

The robot can be equipped with both digital signals (max. 96 inputs/96 outputs) and analog signals (max. 4 inputs/4 outputs). The signals are named and configured in the system parameters. They can also be assigned various system actions, e.g. program start.

In addition to this the robot has four serial channels - three RS 232 channels and one RS 485 channel - which can be used to communicate with printers and computers, for example.

1.1 The Inputs/Outputs window

• Press the Inputs/Outputs key  to open the window.

The window displays a list of appropriate signals or boards. It also provides information on the values of the signals. See the example in Figure 1.

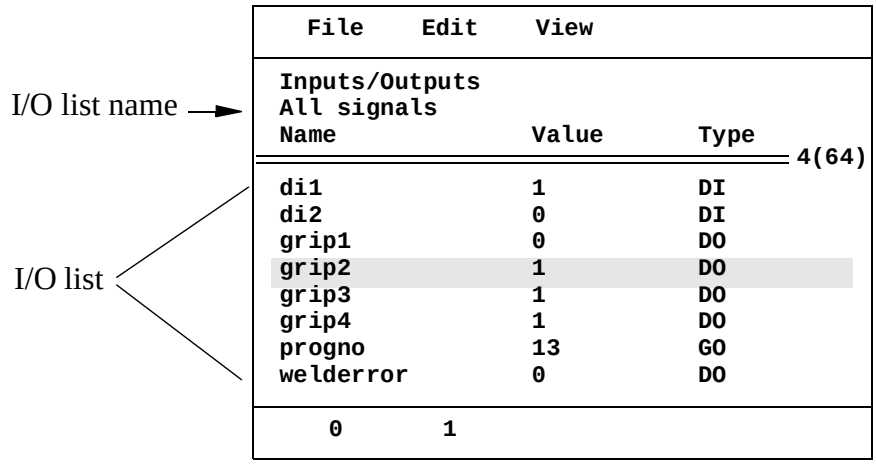


Figure 1 The Inputs/Outputs window displays a list of selected signals or I/O boards.

The information displayed in the window is automatically updated every other second.

1.2 Choosing an I/O list

• You can decide which signals you want to look at by choosing any of the lists from the **View** menu:

List name	Information in the list
Most Common	The value of the most important (most used) signals. This list can be customised to suit any robot installation.
All signals	The value of all signals.
Digital In	The value of all digital input signals.
Digital Out	The value of all digital output signals.
Analog	The value of all analog input and output signals.
Groups	The value of all groups of digital signals.
Safety	The value of all safety signals.
Boards	The type and address of all I/O boards.
I/O Board: *name*	The value and position of all signals of a board. To look at this list: • Choose **View: Boards**. • Select the desired board and press Enter ↵.
Group: *name*	The value and position of all signals in a signal group. To look at this list: • Choose **View: Groups**. • Select the desired board and press Enter ↵

1.3 Defining the *Most Common* I/O list

You can obtain an easy-to-access list of your most frequently-used signals by specifying the contents of the *Most Common* list.

• Choose **File: Preferences**.

All signals will be displayed. Those included in the *Most Common* list will be marked with an *x* to the left of their names (see Figure 2).

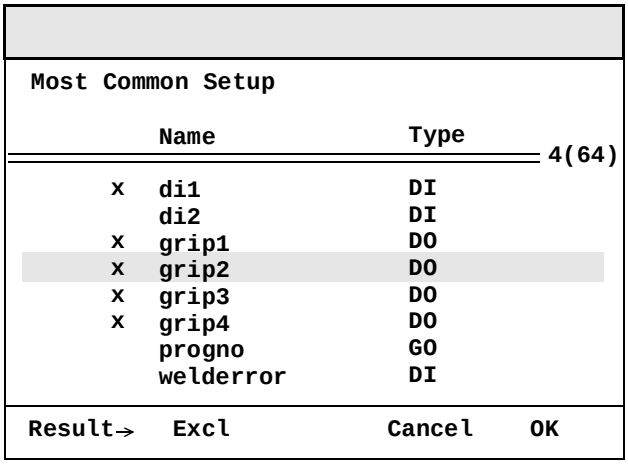


Figure 2 You specify the signals to be included in the list in the Most Common Setup dialog box.

- To add a signal, select an appropriate signal and press ***Incl.***
  This signal will then be marked with an *x* to the left of its name.
- To remove a signal, select an appropriate signal and press ***Excl.***
  This signal will remain in the window, but the *x* to the left of the signal name will disappear.
- Press ***Result***.

The signals included in the *Most Common* list will then be displayed (see Figure 3).

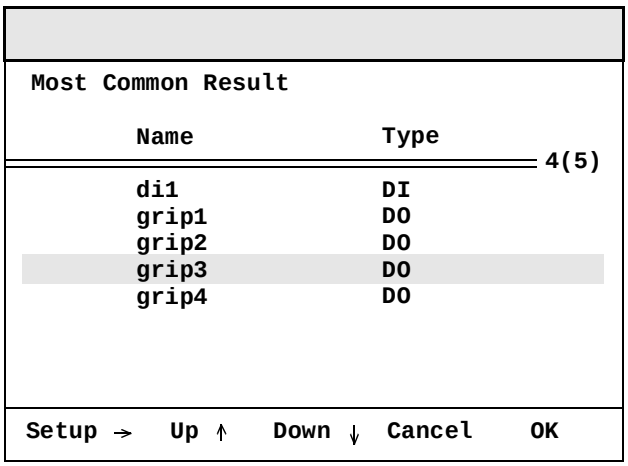


Figure 3 The order of the signals in the list can be specified in the Most Common Result dialog box.

- You can change the order of the signals using ***Up*** and ***Down***. The selected signal moves one step at a time.
- Define the signal and press ***OK***; if you want to return to the *Most Common Setup* dialog box press ***Setup*** first.

2 Changing Signal Values

Robot equipment may be affected (e.g. start to move or fall off) if you change the value of a signal.
Before you do so, make sure that no-one is in the safeguarded space around the robot. Incorrect operation can injure someone, or damage the robot or other equipment.

2.1 Changing the value of a digital output

- Select the digital output.
- Choose the desired value using the function keys (see Figure 4).

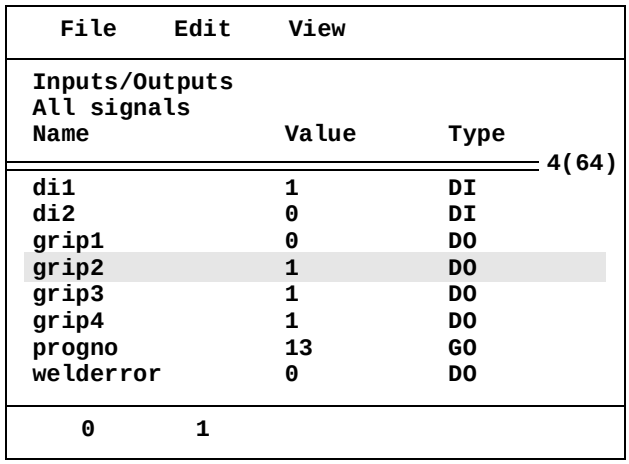


Figure 4 You can change the value of a digital output directly using the function keys.

2.2 Changing the value of an analog output signal or a group of output signals

- Select the signal and press ***Change*** (see Figure 5).

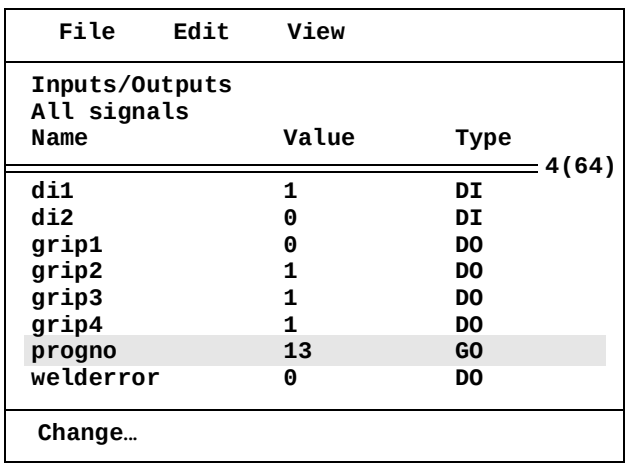


Figure 5 You can change a group of outputs or an analog output signal by choosing Change and entering a value using the numeric keyboard.

A dialog box will appear, which you can use to enter an arbitrary value.

• Specify the desired value using the numeric keyboard and press ***OK***.

You can also change the value of a group of output signals, signal by signal, by pressing Enter [↵] and changing the signals one by one.

3 Displaying Information

3.1 To display information on a given signal

• Select the desired signal and press Enter [↵].

The following information will be displayed:

- the signal name
- the signal type
- the value of the signal
- the physical connection
- cross-connections (if any)
- etc.

• Press ***OK*** when ready.

3.2 To display a chart of all digital signals of a board

• Call up the *Boards* list by choosing **View: Boards**.

• Select the desired board and press the ***State*** function key.

The values of all digital signals related to the selected I/O board will appear on the display (see Figure 6). The values of the signals are indicated by 1 or 0, where, for example, 1 is equivalent to +24 V and 0 is equivalent to 0 V.
An “x” means that the signal is not mapped (cannot be used in a program).
A “?” means that the signal value cannot be read.

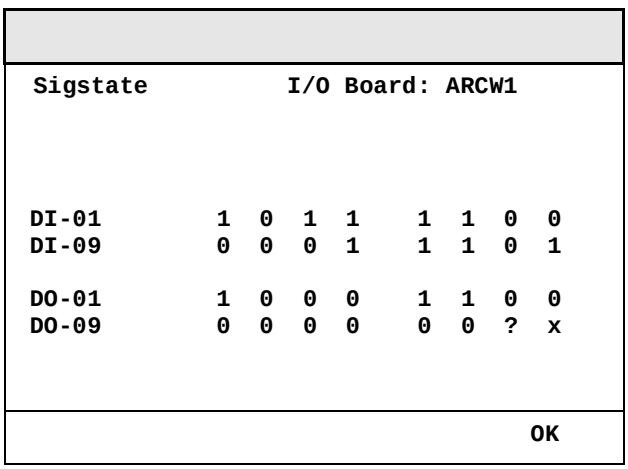


Figure 6 The value of all the digital values of an I/O board are displayed together on a signal chart.

• Leave the signal chart by pressing ***OK***.

3.3 To print an I/O list

- Select the desired I/O list from the **View** menu.
- Choose **File: Print**.

A dialog box will be displayed (see Figure 7).

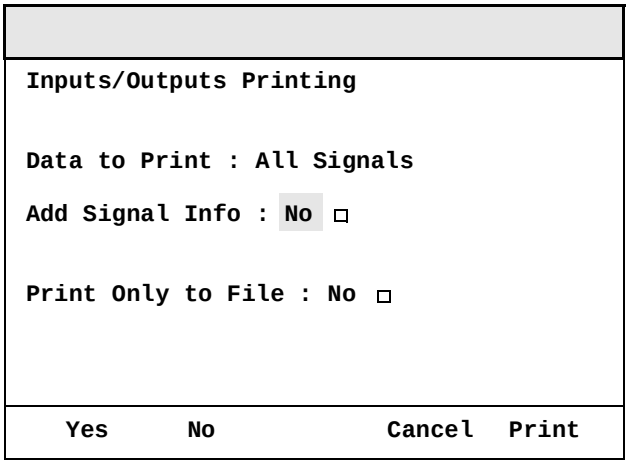


Figure 7 You can specify the extent of information and the destination.

- In the field, ***Add Signal Info***, specify how much you want to print:
  - Press ***No*** to print the list.
  - Press ***Yes*** to print other information about the signals, such as their configuration.
- Select the destination in the field, ***Print Only to File***:
  - Press ***No*** to output to the printer connected to the robot.
  - Press ***Yes*** to save the list in a file. An additional line with the filename will be displayed. If you want to change the filename, select it and press Enter ↵.
- Start the print-out by pressing ***Print***.
- Press ***OK*** to confirm.

CONTENTS

Programming and Testing

1 Creating a New Program

1.1 What is a program?

A program consists of instructions and data, programmed in the RAPID programming language, which control the robot and peripheral equipment in a specified way.

The program is usually made up of three different parts:

- a main routine
- several subroutines
- program data.

In addition to this, the program memory contains system modules (see Figure 1).

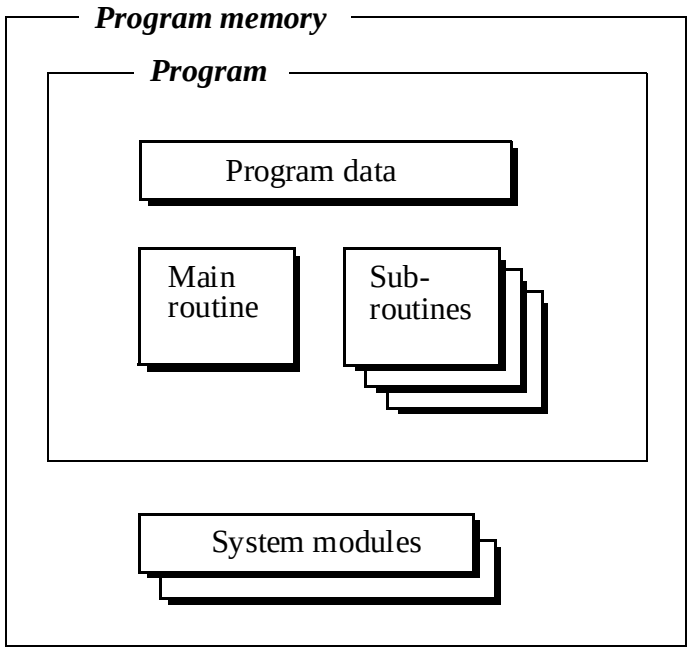


Figure 1 The program instructions control the robot and robot peripherals.

The main routine is the routine from which program execution starts.

Subroutines are used to divide the program up into smaller parts in order to obtain a modular program that is easy to read. They are called from the main routine or from some other routine. When a routine has been fully executed, program execution resumes at the next instruction in the calling routine.

Data is used to define positions, numeric values (registers, counters) and coordinate systems, etc. Data can be changed manually, but it can also be changed by the program; for example, to redefine a position, or to update a counter.

An *instruction* defines a specific action that is to take place when the instruction is executed; for instance, moving the robot, setting an output, changing data or jumping within the program. During program execution, the instructions are executed one at a time, in the order in which they were programmed.

System modules are programs that are always present in the memory. Routines and data related to the installation rather than the program, such as tools and service routines, are stored in system modules.

1.2 The Program window

All program and testing is performed using the Program window.

• Press the Program key  to open the window.

The Program window is actually made up of a number of different windows. These can be chosen from the **View** menu.

Window title	Used to:
Program Instr	Program and change program instructions
Program Routines	Choose or create new routines
Program Data	Create or change data
Program Data Types	Choose data of a specific type
Program Test	Test programs
Program Modules	Choose or create new modules

1.3 Creating a new program

• Choose **File: New**.

If the robot is already loaded with a program which has not been saved, a dialog box appears and you will be asked whether you want to save it or not. Then choose **File: New** again.

• Specify the new name of the program in the dialog box that appears.
(See Chapter 5, Starting up - *Entering text using the teach pendant*, in this manual for how to handle the text editor.)

• Choose ***OK*** to confirm.

A program with only one empty main routine is created.

1.4 Loading an existing program

• Choose **File: Open**.

A dialog box appears, displaying all programs in the current directory (see Figure 2).

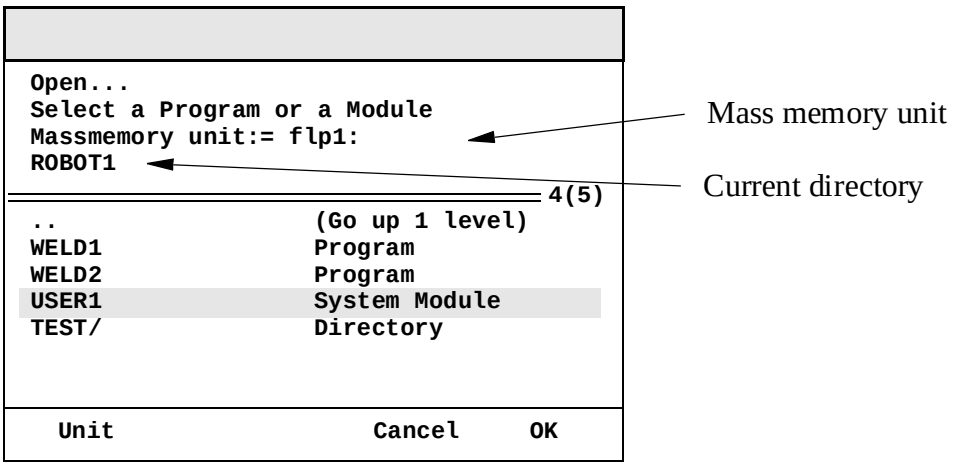


Figure 2 The dialog box used to read programs.

• If necessary, change the mass memory unit by pressing ***Unit*** until the correct unit is displayed.

• Select the desired program. Move up or down in the directory by using either '. .' (up), or the desired directory (down) and press Enter ↵.

• Choose ***OK*** to confirm.

When a program is already loaded into the system, but has not been saved, and you wish to open another program, a dialog box appears and you will be asked whether you want to save the old program or not.

Tip If there is an error in the program, this error will be displayed if you choose **File: Check Program**.

Note When a program is loaded into the robot, it requires about three times as much memory compared with the size of the file on diskette.

2 Defining Tools and Work Object

Before starting any programming work, it is essential that you define the tools, work objects, and other coordinate systems that you intend to use. The more accurately you do this, the better the results you will obtain.

See Chapter 10, Calibration.

3 Creating New Routines

3.1 What is a routine?

Before you start to program, you should think out the structure of your program:

- The program should be divided into several subroutines to get a more readable program.
- Instruction sequences that recur frequently in the program, such as gripper handling, form their own routines.

Figure 3 illustrates an example of a simple program; the robot takes parts to and from a machine. Figure 4 illustrates the structure of this program.

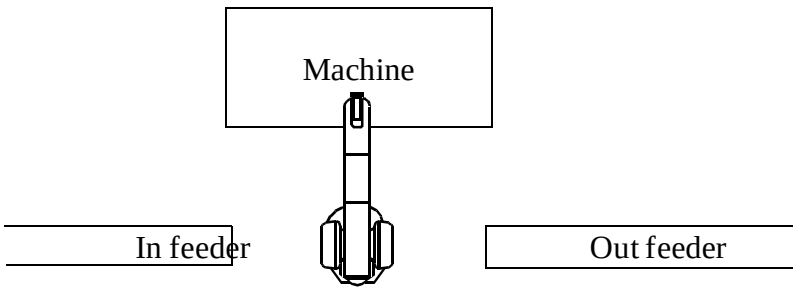


Figure 3 The robot gives a part to a machine which then processes it.

First, the robot fetches a part from the In feeder and places it in the machine where the part is processed. Then, when this has been done, the robot takes the part and places it on the Out feeder.

The main routine is built up of a number of routine calls which reflect the robot work cycle (see Figure 4).

As the gripper grips and releases parts several times during the program run, it is best to set up separate routines for this, which can be called from different places in the program.

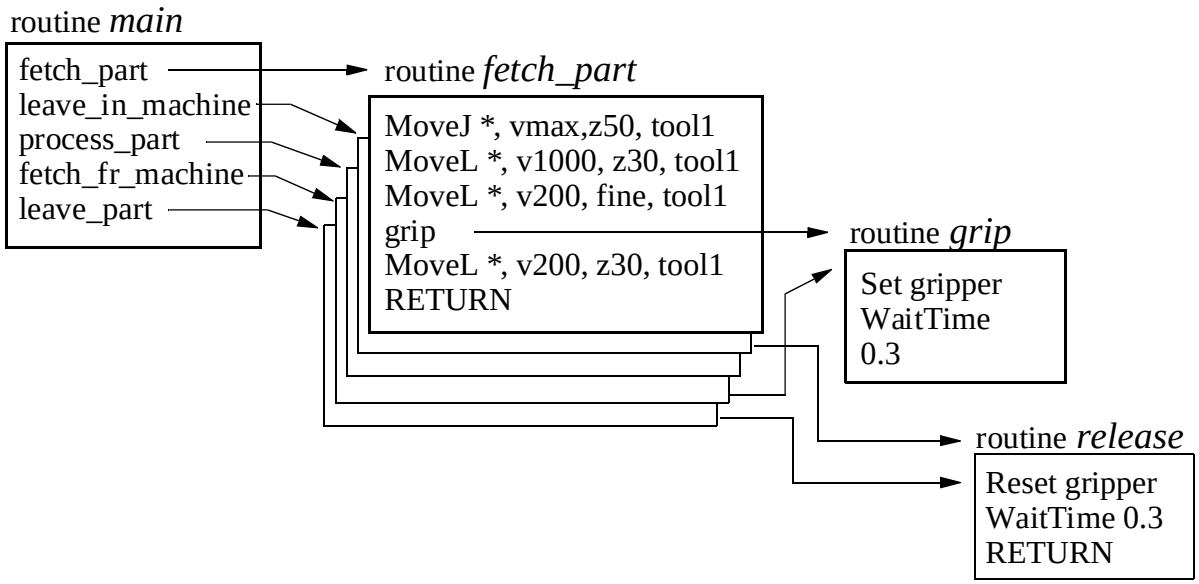


Figure 4 For more information about this example, see Chapter 17, Program Examples.

There are three types of routines: procedures, functions and trap routines.

A *procedure* could be described as a number of instructions that perform a specific task, such as welding a part or changing a tool.

A *function* returns a value and, for example, is used to displace a position or read an input.

A *trap routine* is used to deal with interrupts.

A routine comprises four parts: declarations, data, instructions and an error handler (see Figure 5).

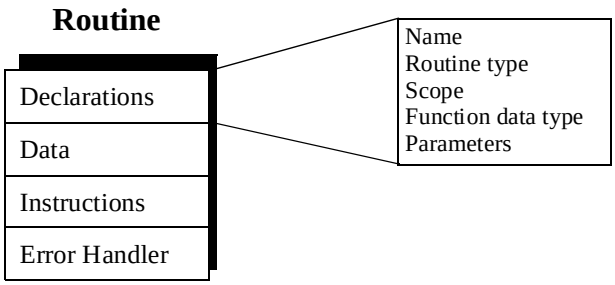


Figure 5 A routine comprises declarations, routine data, instructions and an error handler.

The *declaration* specifies routine parameters, among other things. These are used to make the routine more generally applicable. A routine that, for example, moves the robot a given distance in the direction of the tool can have that distance as a parameter. This routine can then be called using different distances and thus can be used to move the robot different distances.

The *error handler* takes care of automatic error handling (see *Error Handling* on page 48).

3.2 The *Program Routines* window

• Choose **View: Routines** to open the window.

The window displays routines and, if there is a function present, also the type of data returned for that function (see Figure 6).

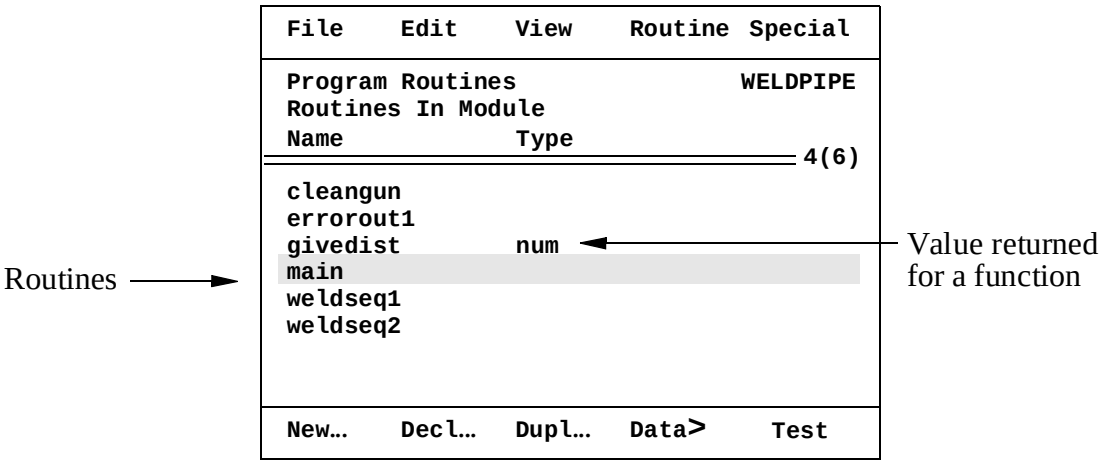


Figure 6 The Program Routines window displays all routines in the program.

3.3 Creating a new routine

• Open the *Program Routines* window by choosing **View: Routines**.

• Press the function key ***New***.

A dialog box appears, displaying the name of the routine (see Figure 7). The name is set to *routineN*, where *N* is a number incremented each time a routine is created.

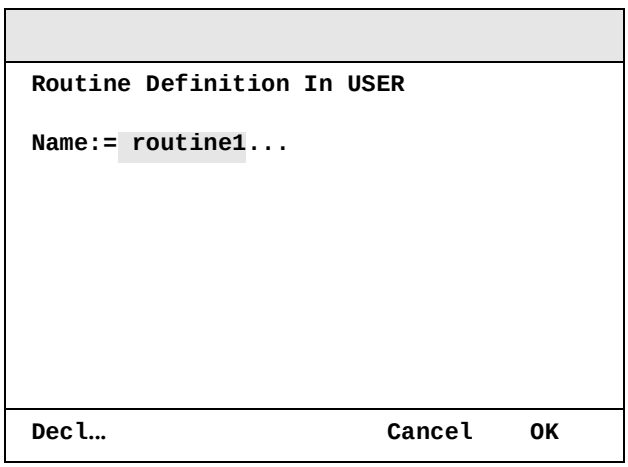


Figure 7 A new routine is created.

• Change the name by pressing Enter ↵ and specify a new name.

If you want a normal subprogram (procedure), without parameters, you should finish here by pressing ***OK***. In other cases, the characteristics of the routine must be defined.

• Press the function key ***Decl***.

• Change the characteristics of the routine by selecting the appropriate field, then:

- Press Enter ↵ and specify the desired alternative in the dialog box that appears on the display (fields marked with ...).
- Choose an alternative using the function keys (fields marked with □).

Field	Description
Name	The name of the routine (a maximum of 16 characters)
In Module	The module in which the new routine will be used
Type	Specifies whether the routine is to be a procedure (***Proc***), a function (***Func***) or a trap routine (***Trap***)
Data type	The return value for the data type (only specified for functions)

If the routine is not to include any parameters, you can terminate the definition by pressing ***OK.*** In other cases, the parameters must also be defined.

• Select the parameter list by pressing the List key.

• Add a parameter by pressing the ***New*** function key.

New parameters are placed after the selected parameter in the list. You can, however, move the parameter using ***Move*** ↑ (up one step) och ***Move*** ↓ (down); see Figure 8.

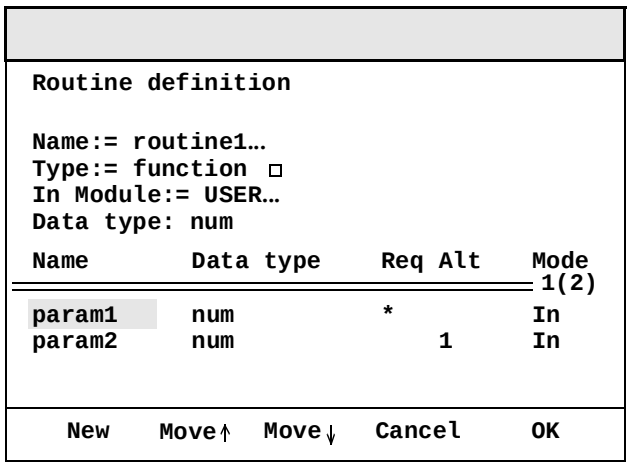


Figure 8 The dialog box used to define parameters.

• Change the name and characteristics of the parameter by selecting the appropriate field, then:

- Press Enter ↵ and specify the desired alternative in the dialog box that appears.
- Choose an alternative using the function keys.

Field	Description
Name	The name of the parameter (max. 16 characters).
Data type	The data type of the parameter.
Required	Specifies whether the parameter is compulsory (***Yes***) or can be omitted (***No***) at a call – marked with * in the list.
Alt	Non-compulsory parameters can be mutually exclusive, i.e. they cannot be used simultaneously in the instruction. To input the first of these parameters, press the function key ***First*** and to input the last one, press ***Tail***.
Mode	Specifies whether the parameter can only be read (***IN***) or whether it can be read and changed in the routine (***INOUT***).

- Add any additional parameters. To remove a parameter, select it and then press Delete ⌫.
- Choose ***OK*** to confirm.

Tip It is sometimes easier to create a new routine by duplicating and changing an existing one.

3.4 Duplicating a routine

- Choose **View: Routines**.
- Select the routine to be duplicated.
- Press the function key ***Dupl***.
- Specify the new name of the routine in the dialog box that appears.
- Choose ***OK*** to confirm the duplication.

This creates a new routine that contains the same data and instructions as the original routine.

4 Starting to Program

4.1 Choosing a routine

- Choose **View: Routines**.
- Select the routine to be programmed and press Enter ↵.

To call up the main routine

- Choose **View: Main Routine**.

To call up a routine that can be selected from the list of instructions

• Select the routine that you want to look at.

• Choose **View: Selected Routine**.

4.2 The *Program Instr* window

• Choose **View: Instr** to open the window.

If you are in the *Program Test* or *Program Data* window, you can press the function key ***Instr*** instead.

The instructions for the current routine are displayed in the window (see Figure 9).

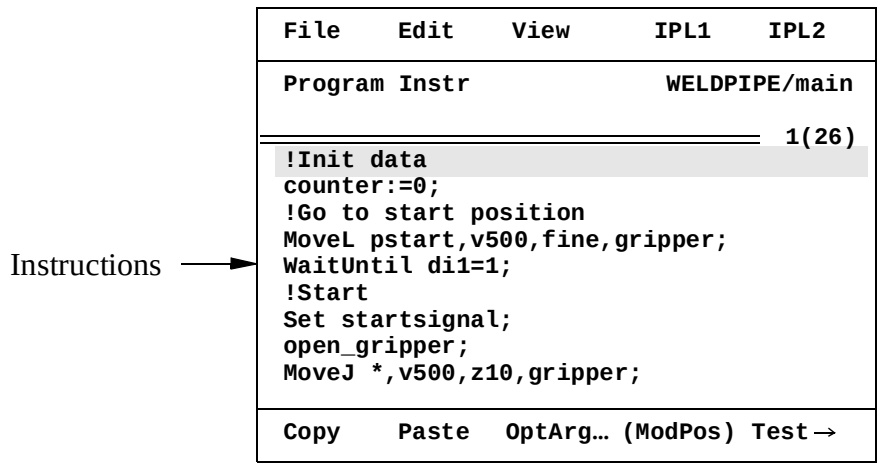


Figure 9 The Program Instr window is used for programming.

An instruction that does not fit into one line is only displayed in part. Arguments that lie outside the visible area are moved successively inwards from the right when the various arguments are selected.

4.3 What is an instruction?

An instruction defines a specific task that is to be carried out when the instruction is executed, for example:

- Moving the robot
- Setting an output
- Changing data
- Jumping within the program.

Instructions comprise an instruction name and a number of arguments. The name specifies the principal task of the instruction and the arguments specify the characteristics.

An argument may be either compulsory (required) or optional. Optional arguments

may be omitted, and are specified by the name of the argument and its value, if it has one. For example:

Instruction	Meaning
MoveL p1,v500,fine,tool1	Moves the TCP linearly to the position *p1*. The arguments, *v500*, *fine* and *tool1*, specify the current speed, position accuracy and tool.
SetDO do2,1	Sets the output *do2* to *1*.
SetDO \SDelay:=0.5,do2,1	Sets the output *do2* to *1* with a delay of *0.5* seconds. *\SDelay* is an optional argument, *do2* and *1* are compulsory.

An argument that does not have a specified value is marked with <...>.
Programs that contain such instructions (i.e. incomplete instructions) can be executed, but program execution stops when that type of instruction occurs.

Arguments can be specified as:

- numeric values, e.g. *1*
- string values, e.g. *"Waiting for machine"*
- data, e.g. *reg2*
- function calls, e.g. *Abs(reg2)*
- expressions, e.g. *reg2 + reg3 / 5*.

4.4 Getting more information about an instruction

• Select the desired instruction and press Enter [↵].

The dialog box shows the names of the arguments (see Figure 10).

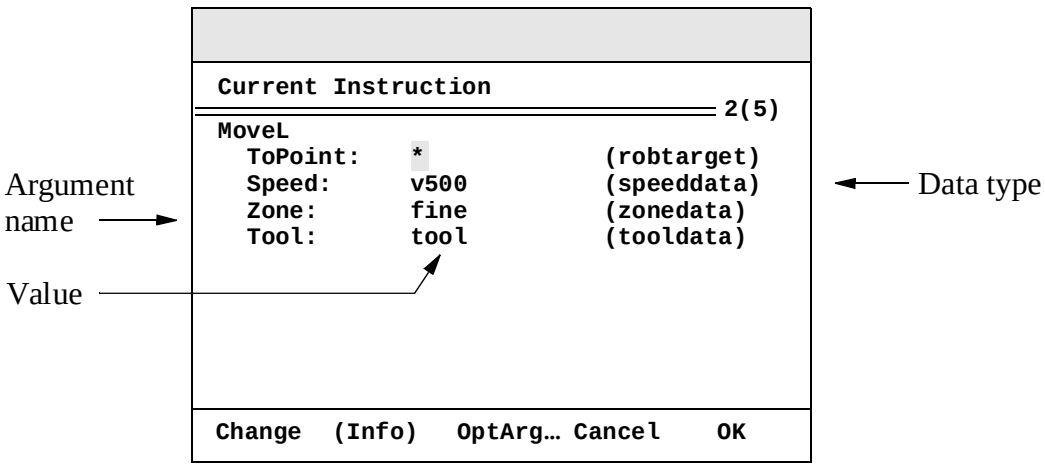


Figure 10 The name, value and data type of each argument is displayed.

• If you wish to change an argument, choose ***Change*** or press Enter [↵]. See *Changing an argument* on page 30 for more information.

- If you wish to add or remove an optional argument, choose ***OptArg***. See *Adding optional arguments* on page 32 for more information.
- Choose ***OK*** to exit the dialog.

5 Programming

In this chapter you will find descriptions for general handling of the various instructions in a program - moving, copying or adding. For details about programming the most common instructions please see the next chapter in this manual, *9 The programming language RAPID*.

For other instructions see the RAPID Reference Manual.

5.1 Choosing from the instruction pick list

You can choose instructions by selecting an appropriate instruction in an instruction pick list (IPL). Although most of these pick lists are fixed, some can be user-defined. This means that you can place the instructions you use most in the same pick list (see *Defining the Most Common instruction pick list* on page 56).

The following pick lists are available:

From the IPL1 menu

Name	Contains
Common	Some of the most-commonly used instructions
Prog. Flow	Instructions that control the program flow
Various	E.g.':=' and wait
Motion Settings	Instructions that affect movements
Motion & Process	Motion instructions
I/O	I/O instructions
Communicate	Communication instructions
Interrupts	Instructions that handle interrupts
Error Recovery	Instructions that handle errors
System & Time	Date and time instructions
Mathematics	Arithmetic instructions

From the IPL2 menu

Most Common 1	User-defined
Most Common 2	User-defined
Most Common 3	User-defined
Motion Set Adv.	Advanced Motion setting instructions

Motion Adv.	Advanced Motion instructions
Ext. Computer	Communication Ware instructions
Service	Service instructions

- Call up one of the instruction pick lists in the **IPL1** or **IPL2** menu.
- Call up the instruction pick list that was used most recently by pressing **Edit: Show IPL**. If the pick list contains more than 9 instructions you can scroll up/down in the list using 9 on the numeric keyboard.
- Change to the previous or next pick list by selecting the pick list ( ) and pressing PreviousPage or NextPage . You can also choose 0 to go directly to the next page.
- Remove the instruction pick list by choosing **Edit: Hide IPL**.

5.2 Adding an instruction

If a new instruction is added, it is placed after the instruction that is selected.

If the selected instruction is first in a routine, or in a compound instruction (IF, FOR, WHILE or TEST), you can choose whether you want the new instruction to be placed before or after the instruction (by means of a question). However, if there is only one instruction in the routine, or in the compound instruction, new instructions will always be added after the selected one.

- Select the place where you want the new instruction to be added.
- Call up one of the instruction pick lists by choosing the appropriate pick list from the **IPL1** or **IPL2** menu. If you want to call up the instruction pick list that was used most recently, choose **Edit: Show IPL**.

The pick list will be displayed on the right hand side of the window (see Figure 11).

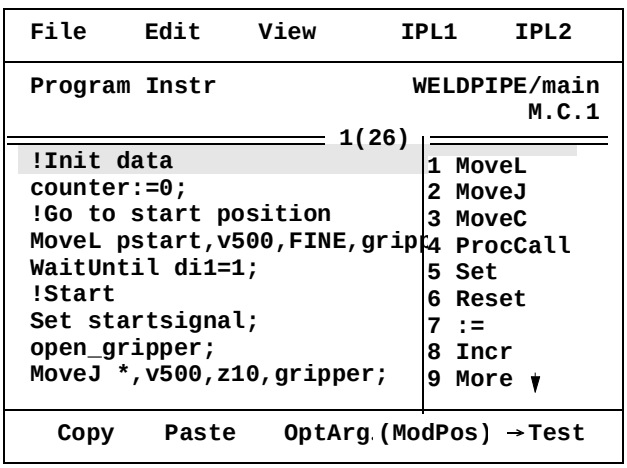


Figure 11 The instructions are chosen from an instruction pick list.

• Choose the desired instruction using one of the following alternatives:

- Using the numeric keyboard, press the number displayed in front of the appropriate instruction in the pick list.
- Select the pick list by pressing the List key. Then, select the desired instruction and press Enter .
- Use 0 on the numeric keyboard to scroll down to the lower part of the pick list or to the next pick list.

If the instruction has no arguments, or if these are automatically set, the instruction is ready for use right away.

If the instruction has arguments that cannot be automatically set, a dialog box will appear in which you can specify the value of the instruction arguments. The argument is marked with a ”?” in front of it (see Figure 12).

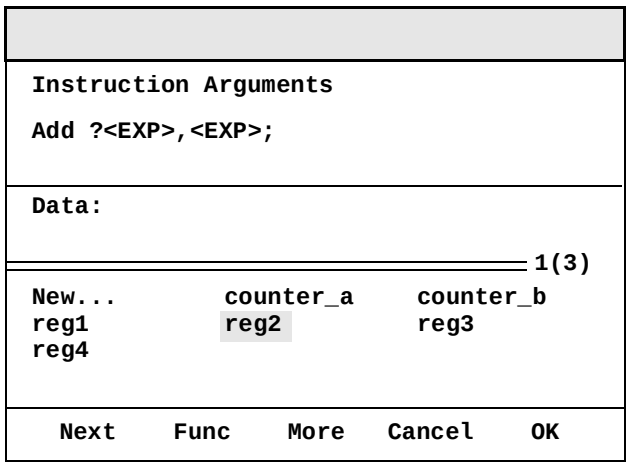


Figure 12 The dialog box used to define arguments. In this example, the instruction Add is programmed.

The argument can now be defined in four different ways:

- by entering a numeric value directly using the numeric keyboard
- by choosing data in the lower part of the dialog box

  New, the first alternative in the list, is used when you want to create new data and refer to it. If you choose ***New***, you define new data as described in *Creating Data* on page 42.

- by choosing a function; press the function key ***Func*** and select the desired alternative from the list

  A new dialog box that can be used to program arguments appears, like the one in Figure 12. Specify the function argument in the same way as you specified the instruction argument. Use the function key ***Skip*** to delete optional arguments that are not to be included.

- by entering an expression by pressing ***More***.

  For more information, see *Programming an expression* on page 17.

• Choose ***Next*** to change the next argument.

• Choose ***OK*** to confirm.

Optional arguments that are not included at the start can be inserted, see *Adding optional arguments* on page 32.

The structure of an IF, FOR or TEST instruction can be changed, see *Changing the structure of an IF, FOR or TEST instruction* on page 32.

5.3 Expressions

What is an expression?

An expression is used as an argument of an instruction and can have an arbitrary number of components.

There are three different types of expressions:

- logical expressions;
  these have the value true/false and are used together with tests, e.g.
  IF reg1=5 AND reg2 >10 .....
  IF di1 = 1 .....
- arithmetic expressions;
  these have a numeric value and are used together with calculations, e.g.
  reg1 = reg2 + 3 * reg5
  reg1 = reg2 + 1
- strings, e.g.;
  TPWrite "Producing"

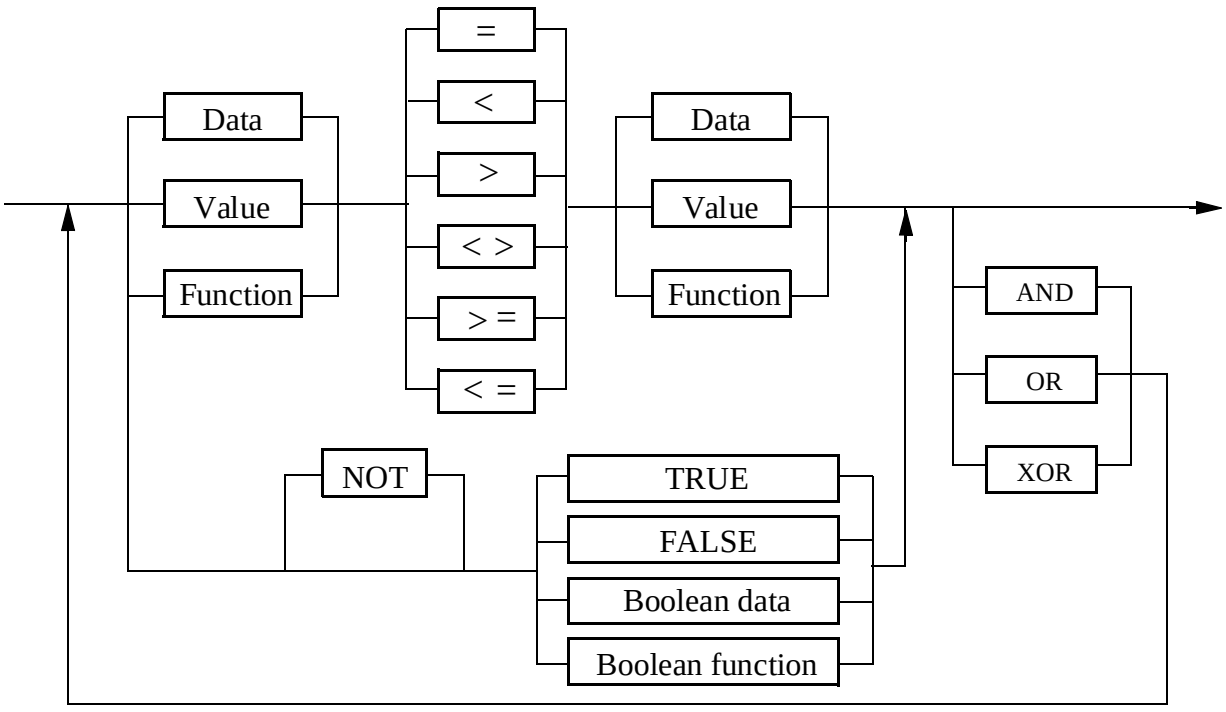


Figure 13 Logical expression.

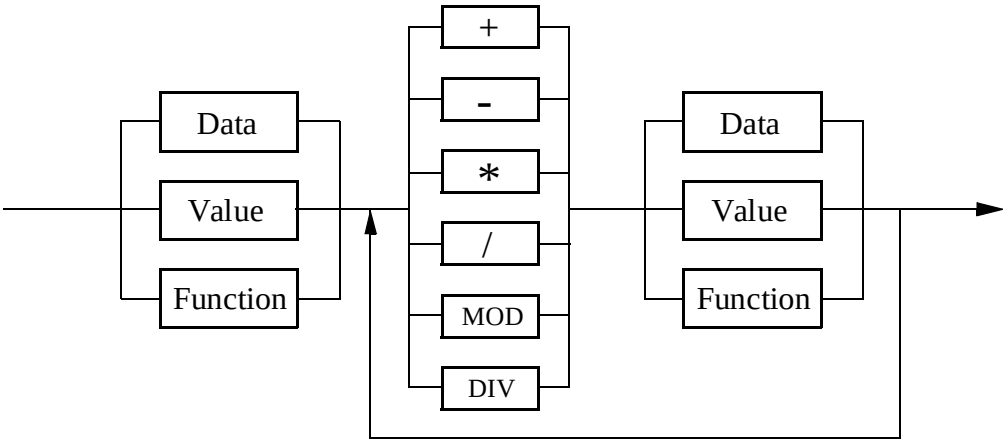


Figure 14 Arithmetic expression.

Programming an expression

Expressions are programmed by pressing the function key ***More*** in the instruction argument dialog box (see Figure 12).

Expressions can be entered or changed directly in the upper part of the dialog box (see Figure 15) by doing any of the following:

- move the cursor to the left or right using ArrowLeft ⟨ or ArrowRight ⟩;
- delete what is marked by the cursor by pressing Delete ⌫;
- add digits in front of the cursor using the numeric keyboard.

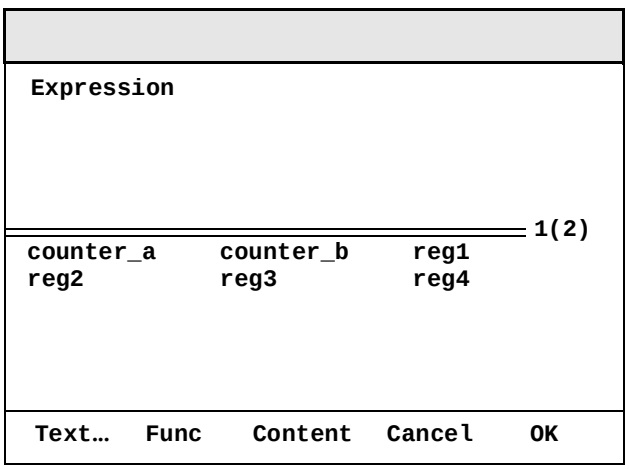


Figure 15 The Expression dialog box.

Data, functions and operators can be selected in the lower part of the dialog box. Press ☰, if needed, select the desired alternative and press Enter ↵.

Enter text by pressing ***Text***. A dialog box appears in which, using the function keys and the numeric keyboard, you can enter text.

If the desired information is not in the lower part, press one of the function keys ***Data***, ***Func*** or ***Content*** first.

- ***Data*** gives a list of all user-defined data of the selected data type
- ***Func*** gives a list of all functions of the selected data type
- ***Content*** gives an intermediate dialog where data of a new data type can be chosen in the same way as the IF instruction, for example. You can also choose to view user-defined or system-defined data, or both. √ denotes the current choice (see Figure 16).

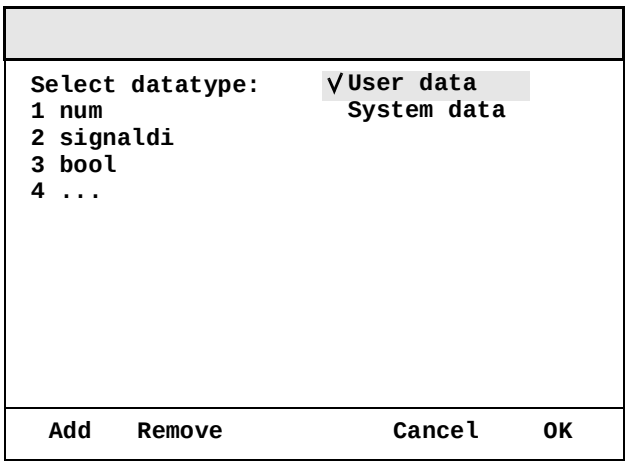


Figure 16 Dialog box for choosing data types.

Editing an expression

Move the cursor using the arrow keys. The content of the list will change so that it corresponds to that selected. The function key ***Content*** changes to ***Insert*** (see Figure 17).

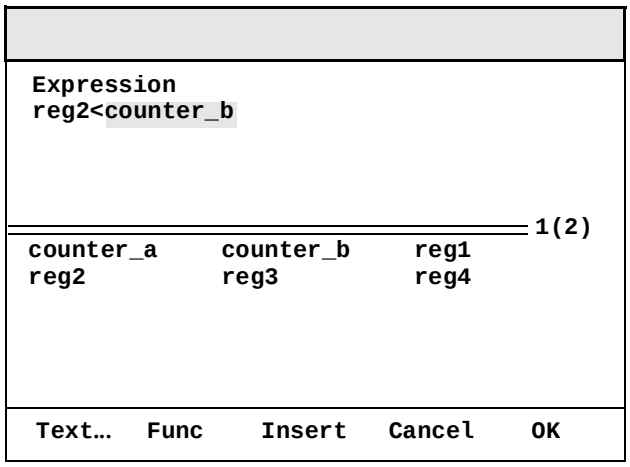


Figure 17 The dialog box for editing an expression.

Replace what has been selected by selecting the desired choice in the lower part of the box and pressing Enter ↵.

You can make an addition to an expression by pressing the function key ***Insert***. An underscored "blank" _ will then be inserted before the cursor and the function key ***Insert*** will change to ***Content*** (see Figure 18).

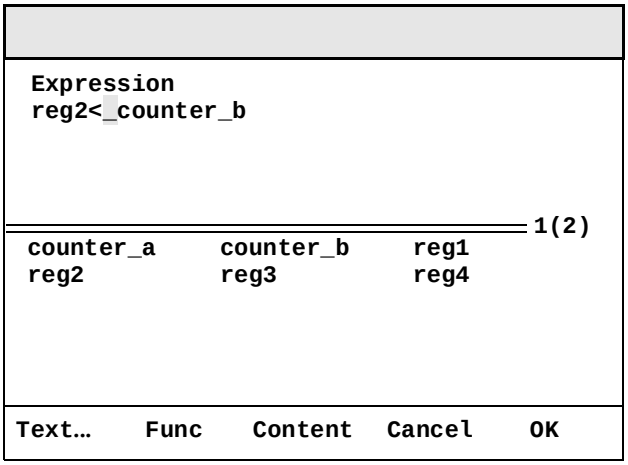


Figure 18 New data can be inserted into an expression.

5.4 Moving and copying instructions

- Select the instruction you wish to move or copy. To select several instructions, choose **Edit: Mark**.
- Choose **Edit: Cut** (move) or **Edit:Copy**.
- Indicate where you wish to add the new instructions.
- Choose **Edit: Paste**.

In the *Program Instr* window, copy and paste can also be selected using a function key.

6 Running Programs

6.1 Program execution

A program can be executed regardless of whether or not it is complete. Nonetheless, if program execution comes to an incomplete instruction, the program is stopped.

When the program is started, the robot checks that all references to data and routines are correct. If they are not, the fault is indicated and the program is not started. This check can also be performed by choosing **File: Check Program**. The first fault in the program is then indicated.

The program is usually started from the first instruction in the main routine, but can also be started from an arbitrary routine (procedure) with no parameters. A program that has been stopped is, unless otherwise specified, always started from the instruction last executed in the program.

6.2 The *Program Test* window

• Choose **View: Test**. When you are in the *Program Instr* or *Program Data* window, you can also press the function key ***Test***.

The section of the program that will be executed when you start the program is displayed in the window.

A program pointer keeps up with the program execution. This pointer is shown with » in the program list. Program execution normally continues from this point. However, if the cursor is moved to another instruction when the program is stopped, execution can be started from the position of the cursor (see Figure 19).

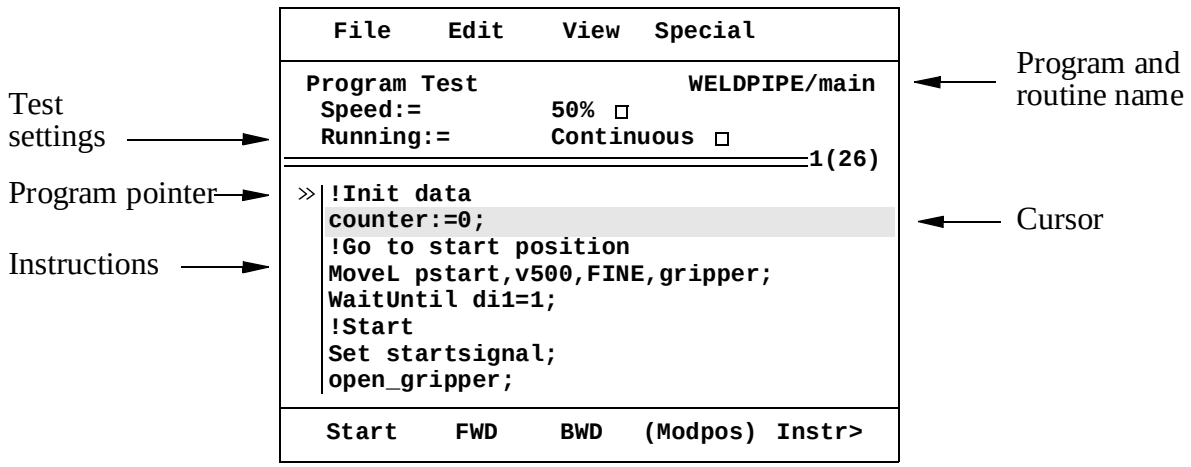


Figure 19 The Program Test window is used to execute a program.

If the robot is equipped with an arc-welding function, an extra field with the blocking status will be shown.

6.3 Choosing the speed correction

When the program is being tested for the first time, it is advisable to reduce the speed. A 50% speed correction means that the speed will be reduced to 50% of the programmed speed. On the other hand, when the robot is in manual mode with reduced speed, the speed is never more than 250 mm/s.

It is also possible to change the speed correction while the program is executing.

• Select the upper part of the window by pressing the List key (if it is not already selected).

• Select the field ***Speed*** (see Figure 20).

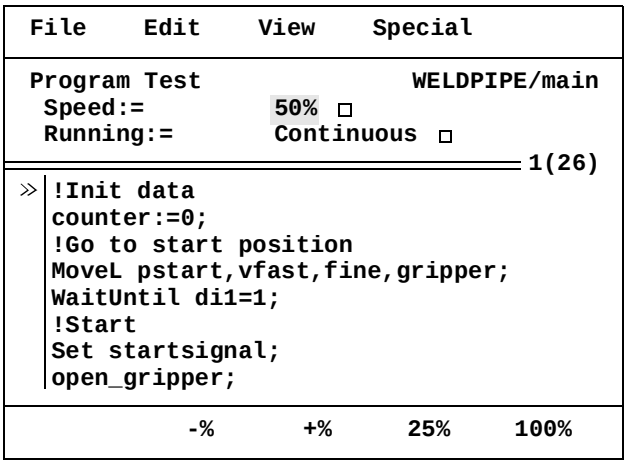


Figure 20 The speed can be changed (0 - 100%).

• Increase or decrease the speed by pressing the function keys ***-%*** or ***+%***. Correction then takes place in steps of 5%.
Set the speed to 25% or 100% by pressing the function key ***25%*** or ***100%***.

6.4 Choosing the program execution mode

The program can be run in three different modes;

- continuous
- cycle (one cycle at a time)
- step-by-step (forwards or backwards, one instruction at a time).

Choose continuous or cyclic running as follows:

• Select the upper part of the window by pressing the List key (if it is not already selected).

• Select the field ***Running***.

• Choose the program execution mode using the function key ***Cont*** or ***Cycle***.

Use the key to select the lower part of the window.

Use the function key ***Start*** to start program execution in the mode that you chose above. To step the program forwards/backwards, use the function keys ***FWD*** and ***BWD*** (see Figure 21).

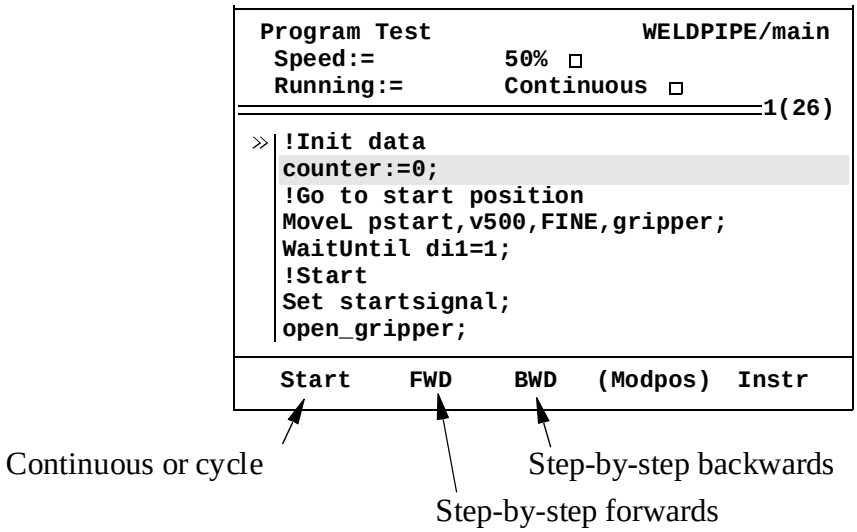


Figure 21 A program can be run in several different program execution modes.

Instructions act differently during step-by-step execution than during continuous execution. The principal differences are as follows:

- Positioning instructions are executed in the normal way, but the robot gets into position even when a fly-by point is programmed.
- Other instructions execute in the normal way when executing forwards and are skipped when executing backwards.

6.5 Starting program execution

• Choose the speed correction as above.

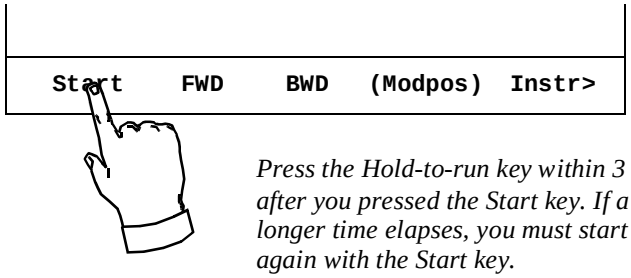
• Select the lower part of the window by pressing the List key (if it is not already selected).

When you start program execution, the robot will start to move. Peripheral equipment may also be started.
Make sure that everything is ready for program execution to begin and that nobody is in the safeguarded area around the robot. Starting the program incorrectly can injure someone, or damage the robot or other equipment.

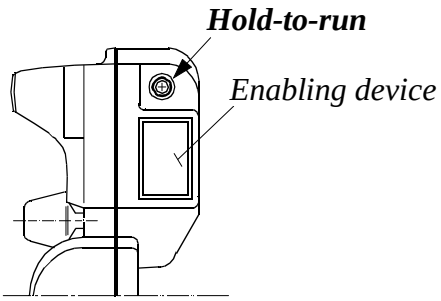
• Press the function key ***Start*** for continuous or cycle execution mode.
If you want to execute step-by-step, press the function key ***FWD*** or ***BWD*** instead.

When “Hold-to-run” is activated, the following is applicable:

- for Teach pendant Version 1; press the ***Start*** key and keep it depressed while the program is running, otherwise the program will stop.
- for Teach pendant Version 2; press the ***Start*** key, release it, and press the Hold-to-run key. Keep this key depressed while the program is running, otherwise the program will stop (see Figure 22).



Press the Hold-to-run key within 3 s after you pressed the Start key. If a longer time elapses, you must start again with the Start key.



Teach pendant Version 2

Figure 22 The Hold-to-run key is located on the side of the teach pendant.

6.6 Stopping program execution

When Hold-to-run control is enabled

• Release the start key for Version 1 or the Hold-to-run key for Version 2.

When Hold-to-run control is NOT enabled

• Press the Stop key on the teach pendant.

If the program execution mode is changed from continuous to step-by-step or cycle, the robot will stop automatically after it has completed the instruction or the cycle.

6.7 Where will the program start?

How do you recognise the program pointer?

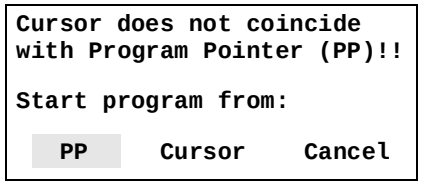
The program pointer shows how far the program has run and is marked with » in front of the instruction.

An instruction that has been fully executed is marked with an ×, but is only shown during instruction-by-instruction execution. If the cursor is positioned at this instruction, the program starts from the program pointer ». See the example below. (In all other cases, the cursor will define the instruction that will be executed when you press ***Start***.)

Example:

```
IF reg1=5 THEN
× reg2:=5;                    The last instruction executed.
ELSE
   reg2:=8;
ENDIF
» Set do1                     The next instruction to be executed
```

If the cursor is not located on the last instruction executed, then when you press ***Start***, an alert box will be displayed (because the program flow has been changed).
Select whether you wish to start from the program pointer (PP) or the cursor using the arrow keys:



• Press Enter ↵.

To move the cursor to the program pointer

• Choose **Special: Move cursor to PP.**

To move the program pointer to the cursor

• Choose **Special: Move PP to cursor.**

Note If the program pointer is moved into a FOR statement the program will run the rest of the FOR statement to the end, and then continue with the next statement.

To start the program from the beginning

• Choose **Special: Move PP to Main.**

The program pointer and the cursor are set to the first instruction in the main routine.

To start the program from a routine

The program pointer and cursor can be moved to any routine (procedure) with no parameters. If it is moved, the call hierarchy at that time will no longer be valid, which means that program execution continues from the start of the routine after the routine has been fully executed.

• Choose **Special: Move PP to Routine.**

A dialog box appears, displaying all possible routines.

• Select the desired routine and press ***OK***.

To go to a position without moving the program pointer

Place the cursor on the position argument in the instruction. You can also select a position (robtarget) in the *Program Data* window.

• Choose **Special: Go to selected position.**

A dialog box appears, see Figure 23.

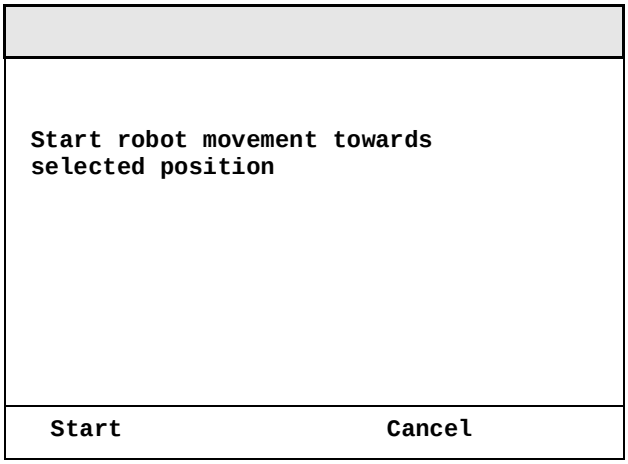


Figure 23 The Go to selected position dialog box.

• Press the function key ***Start*** to start the movement.

6.8 Simulating wait conditions

When the robot is stationary in a wait instruction, e.g. *WaitDI di1* or *WaitTime 3*, a dialog box is automatically displayed.

• To continue in the program without fulfilling the condition or time, press Enter ↵.

The dialog box will disappear automatically when the condition has been fulfilled.

7 Saving and Printing Programs

7.1 Saving the program on diskette or some other type of mass memory

To save a program that has been stored previously

• Choose **File: Save Program**.

The program is duplicated to mass memory and replaces the version that was last saved. If the file name or module name is not the same, the dialog **Save Program As** will be displayed automatically.

To save under a new name

• Choose **File: Save Program As**.

A dialog box appears, displaying all programs in the current directory (see Figure 24).

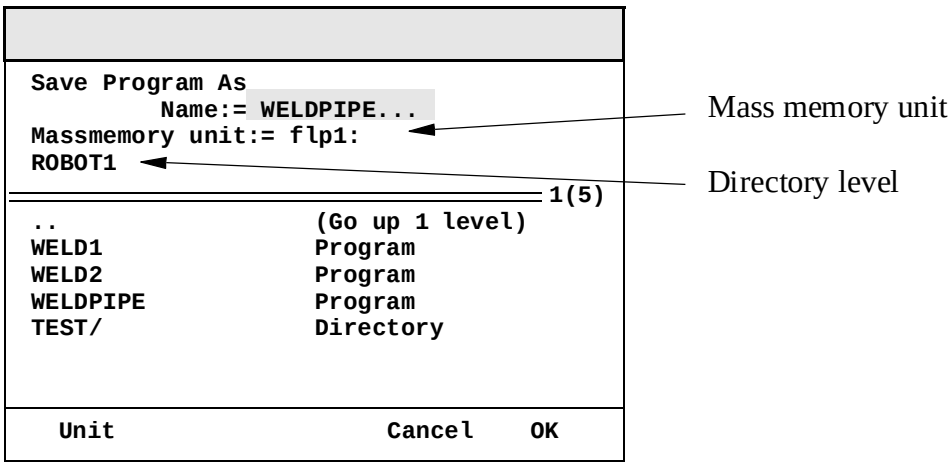


Figure 24 The dialog box used to store programs.

- If necessary, change the mass memory unit by pressing ***Unit*** until the correct unit is displayed.

If the program is to be saved in another directory:

- Select the lower part of the window by pressing the List key.
- Choose the directory in which the program is to be saved. Move up or down in the directory by choosing either '. .' (up), or the desired directory (down) and press Enter.
- Select the upper part of the window by pressing the List key.

- Press Enter when the field ***Name*** is selected.
- Specify the new name in the dialog box that appears. When you have finished entering text, press ***OK***.
- Choose ***OK*** to confirm the save.

Note If a file with the same name already exists, a warning will be given and you can choose to finish or continue.

Note If you have made a change in a system module, you will be requested to save this alteration.

7.2 Printing a program from the robot

Print a whole program

- Save the program on a diskette or the ramdisk, and print out from the File Manager. See Chapter 13 in this manual - *File Manager*.

Print a module

- Choose **File: Print**. The current module will be printed directly or saved to a file.

To be able to print, a printer must be connected to the robot controller.

7.3 Printing a program using a PC

A program can be printed using a personal computer. Most word-processing programs can be used, the only requirement being that the PC can support diskettes in DOS format.

- Store the program on a diskette.
- Load the program into the PC.
- Print the program.

If you do not wish to print out the position values of a position instruction, save the program using the command **File: Print** in the *Program* window and choose **Save to file**. Only the current module will be saved.

8 Changing the Program

Programs can be protected against alteration by making the appropriate settings in the system parameters. A password must then be used to make any changes. See chapter 12, System parameters *Topic: Teach Pendant*.

8.1 Selecting an instruction or an argument

A complete instruction or a single argument can be selected before a command is given to change the program. If you wish to change a single argument, it is often easiest to select the argument first. If you wish to change a complete instruction, you select the complete instruction. Often, e.g. when adding a totally new instruction, it does not make any difference whether the complete instruction, or an individual argument, is selected.

To select a complete instruction

Movement	Choose
Up one instruction	ArrowUp 
Down one instruction	ArrowDown 
To first instruction	**Edit: Goto Top**
To last instruction	**Edit: Goto Bottom**
To next page	NextPage 
To previous page	PreviousPage 

If the cursor is moved to the first line in a compound instruction (IF, FOR, WHILE or TEST), all instructions, including the last line (e.g. ENDIF), will be selected. If ArrowDown is then pressed, the instructions in the compound instruction will be selected, one after the other. Terminators (e.g. ENDIF, ELSE) cannot, however, be selected separately.

When the cursor is moved upwards to a compound instruction, ArrowUp can be used to select the instructions within that instruction, and ArrowLeft can be used to select the complete compound instruction.

To select a number of instructions

You can select a group of instructions that are in sequence.

- Select the first or the last instruction in the group.
- Choose **Edit: Mark**.
- Select the other instructions using ArrowUp or ArrowDown .

The selection will automatically be deactivated when **Edit: Cut** or **Edit: Copy** is chosen. You can also make the selection inactive by choosing **Edit: Unmark**.

To select an argument

- Use ArrowRight to move the cursor one argument to the right, or ArrowLeft to move the cursor one argument to the left.

8.2 Modifying the position in a positioning instruction

- Move the robot to the desired position.
- Select the instruction that is to be changed. For instructions containing more than one position, e.g. *MoveC*, select the position argument to be changed.
- Press the function key ***ModPos*** or choose **Edit: ModPos**.

The maximum movement/reorientation of a position can be limited in the system parameters. If this has already been implemented, the system parameters must be changed in order to allow any greater changes of position. See chapter 12, System parameters *Topic: Teach Pendant.*

The old position is replaced by the current position of the robot. This position is related to the current tool and work object.

Note If a named position data is modified, all other instructions which refer to that position data will also be changed.

8.3 Tuning position during program execution

The tuning command makes it possible to tune the x, y and z coordinates of a robtarget during program execution. The function is valid only for named robtargets of the datatypes constant and persistent. It is not valid for positions represented by ‘*’ and robtargets of the datatype variable. The change is also valid in stopped mode.

- Start with the *Program Test* window.
- Press ***Start***.

The window *Program Run Info* appears (see Figure 25).

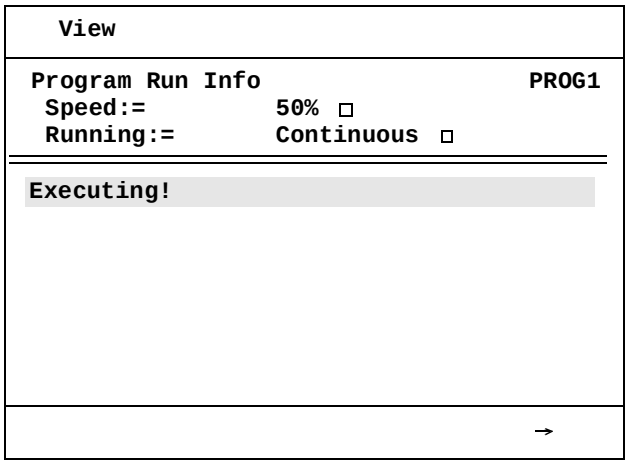


Figure 25 The window Program Run Info.

• Select **View: Position**.

The window *Program Run Position* appears (see Figure 26).

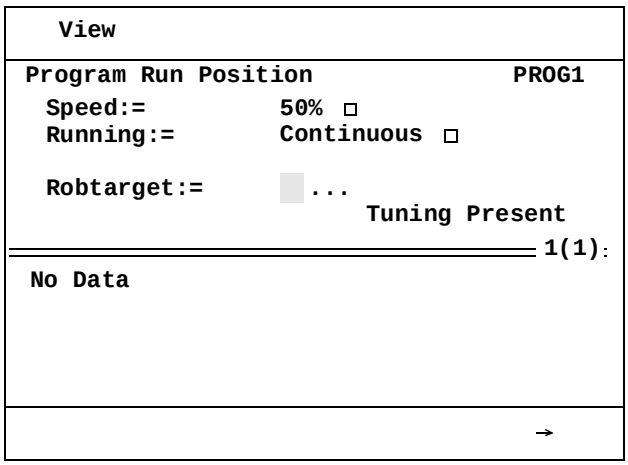


Figure 26 Window for tuning position during execution.

• Select the field ***Robtarget*** and press Enter [↵].

• Select the robtarget to be tuned.

• Choose ***OK*** or press Enter [↵] to confirm the choice.

The x, y and z values of the chosen position are displayed (see Figure 27).

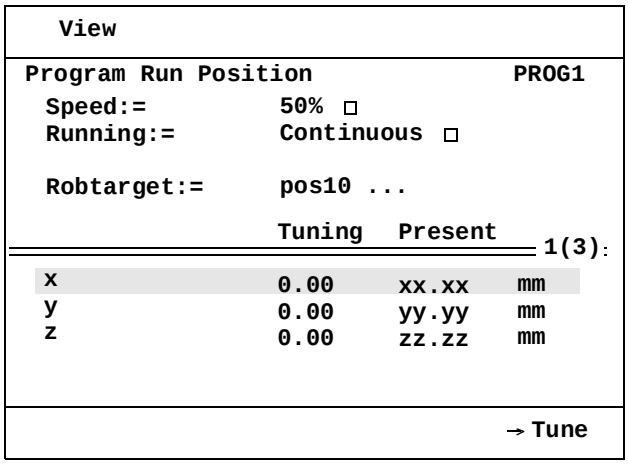


Figure 27 The Program Run Position window with a robtarget selected.

• Choose the x, y or z coordinate in the list.

• Press ***Tune***.

A dialog box will appear where you can tune the position.

• Enter the desired tuning value and press Enter ↵.

- No change = 0
- Maximum change in one step = ±10 mm.

Several steps can be entered. The position data is changed immediately after each step but will not affect the robot path until the next instruction using this position data is executed. The values in the ***Present*** column will be used in this instruction.

The total tuning will be displayed in the ***Tuning*** column.

Note If a named position data is modified, all instructions which refer to that position data will be affected. Unnamed positions (marked as * in the instruction) cannot be tuned.

8.4 Changing an argument

• Select the argument that is to be changed.

• Press Enter ↵.

The dialog box used to program instruction arguments appears and the selected argument is marked with ”?” in front of it (see Figure 28).

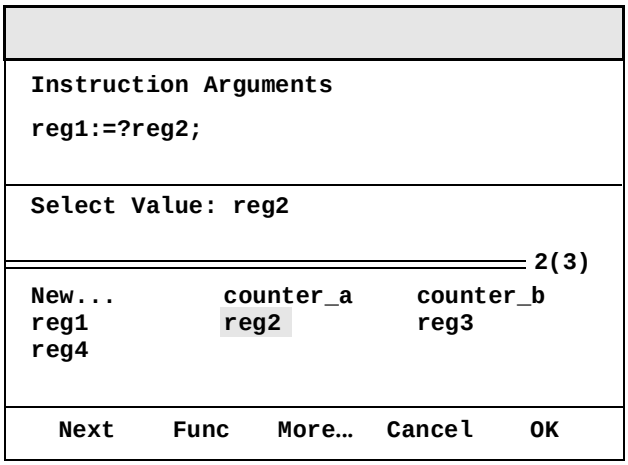


Figure 28 The dialog box used to change arguments. In this example, the argument reg2 will be changed.

The argument can now be changed in four different ways:

- by changing a numeric value; this alternative is used when a numeric value is to be specified, e.g. *5*, or when an argument is to be changed, e.g. from *reg2* to *reg3*;
  - Select the middle part of the dialog box and alternately do one of the following:
    - move the cursor to the left or right using ArrowLeft or ArrowRight;
    - delete the character in front of the cursor by pressing Delete;
    - enter digits at the cursor using the numeric keyboard.
- by choosing data in the lower part of the of the dialog box; this alternative is used when the argument is to constitute a reference to data, e.g. *reg2;*
- by choosing a function; press the function key ***Func*** and select the desired alternative from the list. This alternative is used when an argument is to constitute a function call, e.g. *Offs(p1,5,0,0);*

  A new dialog box that can be used to program function arguments appears. Use the function key ***Skip*** to delete optional arguments that are not to be included.

- by entering an optional expression, press the function key ***More***; this alternative is used when the argument is to constitute an expression with several components, e.g. *reg1+reg2* or *reg1>5*, or a string value, e.g. *"Producing part A"*.

  For more information, see *Expressions* on page 16.

• If desired, choose ***Next*** to change the next argument.

• Choose ***OK*** to confirm.

Note You can also use ***Copy*** and ***Paste*** to change arguments.

Note Any changes in an active position instruction (except for ***ModPos***) will be valid first for the next execution of the instruction. To get an immediate result, choose **Special: Move PP to cursor.**

8.5 Adding optional arguments

Optional arguments of an instruction are not normally included when programming an instruction, but have to be added afterwards.

- Select the instruction that is to be modified.
- Press the function key ***OptArg***.
  If you are in the *Program Test* window, you must first select the whole instruction, then press Enter [↵] and then ***OptArg***.

A dialog box appears, displaying all arguments that the current instruction can possibly have. The arguments not included in the instruction are enclosed within square brackets (see Figure 29).

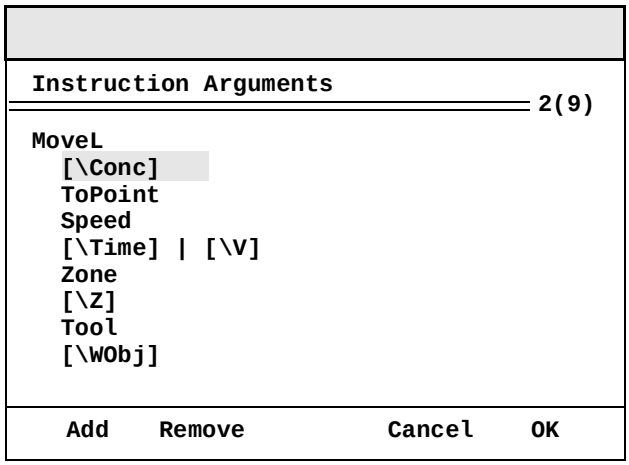


Figure 29 The dialog box used to add optional arguments.

- Add an optional argument by selecting the desired argument and pressing ***Add***.

Some arguments (displayed on the same line) cannot exist simultaneously in an instruction. When such an argument is added, the corresponding mutually exclusive argument is automatically removed.

An optional argument can also be removed by selecting the desired argument and pressing the function key ***Remove***.

- Choose ***OK*** to confirm the change.

8.6 Changing the structure of an IF, FOR or TEST instruction

- Select the complete instruction that is to be changed.
- Press Enter [↵].

A dialog box appears, displaying the structure that the instruction can have. Structure parts not included in the instruction are enclosed within square brackets (see Figure 30).

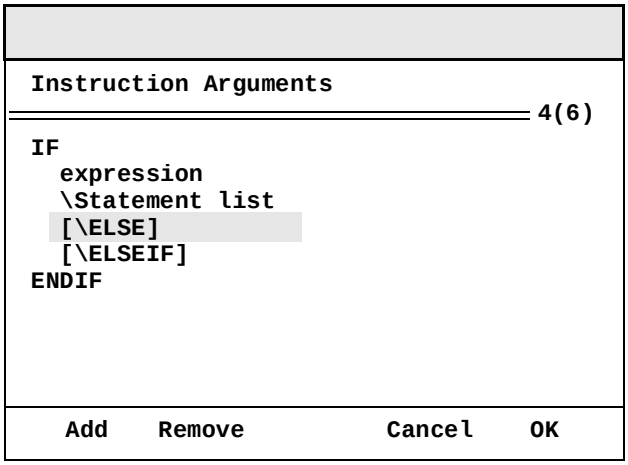


Figure 30 The dialog box used to change the structure of an IF instruction.

• Add part of the structure by selecting the desired part and pressing ***Add***.

• Remove a part of the structure by selecting the desired part and pressing ***Remove***.

• Choose ***OK*** to confirm the change.

Note If you want to add more than one ELSEIF or CASE, these can be added in the Program window using ***Copy*** and ***Paste***. Different CASE statements, such as CASE 1, 2, 3, can also be added using ***Copy*** and ***Paste***.

8.7 Changing the name or declaration of a routine

• Choose **View: Routine**.

• Select the desired routine.

• Press the function key ***Decl***.

A dialog box appears, displaying the routine declaration.

• Make whatever changes you wish to make (see *Creating a new routine* on page 8).

• Choose ***OK*** to confirm the change(s).

8.8 Deleting an instruction or an argument

• Select the instruction or the argument you wish to delete.

• Press Delete ⌫ .

Note An answer must be given to the password check and confirmation dialog if they have been set to active in the configuration. The correct password must be entered at the password prompt. The confirmation dialog is confirmed by Yes or No.

If an argument is compulsory (required), it will be replaced by <...>.

8.9 Deleting a routine

- Choose **View: Routines**.
- Select the desired routine.
- Press Delete ⌫.

Note An answer must be given to the password check and confirmation dialog if they have been set to active in the configuration. The correct password must be entered at the password prompt. The confirmation dialog is confirmed by Yes or No.

9 Special Editing Functions

9.1 Search & replace

The search and replace function makes it possible to search for, and replace data names in the program. It is also possible to search for/replace procedure/function calls. Instruction names can also be changed, e.g. from *MoveL* to *MoveJ*.

Choose **Edit: Search & Replace** in the *Program Test* or *Program Instr* window.

A dialog box appears (see Figure 31).

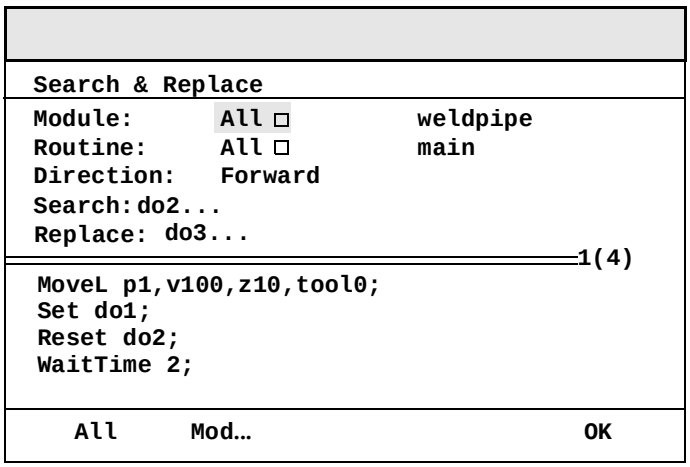


Figure 31 The search and replace dialog.

• Define how the search is to be carried out by defining the following fields.

Field	Description
Module	If "All modules" is selected, the current module name is shown to the right of the field. If ***Mod...*** or Enter ↵ are pressed, a list of available modules is shown, and a specific module can be selected.
Routine	Searching/replacing is performed in all available routines in the module selected from the module field. If ***Rout...*** or Enter ↵ are pressed, a list of available routines is shown, and a specific routine can be selected.
Direction	Directions for searching.
Search	Selection field where a list of available names to search for is shown when you press Enter ↵.
Replace	Selection field where a list of available names to replace is shown when you press Enter ↵.

To start a search

• Move the cursor to the lower part of the window using the list key (see Figure 32).

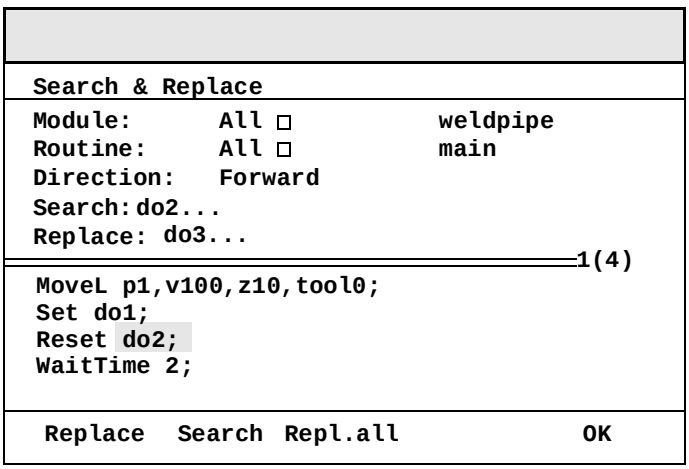


Figure 32 Search & Replace dialog when the program is selected.

• Press the function key ***Search*** to start the search.

The first match will be selected in the lower part of the window.

• Press ***Replace*** to replace the selected text or press ***Repl.all*** to replace all matches without having to confirm.

• Continue the search with ***Search***.

• Press ***OK*** to end the search.

9.2 Mirroring

The mirror function can be applied to any routine in a program.

Mirroring a routine means that a copy of the routine is created with all positions mirrored in a specific mirror plane.

The new, mirrored routine will be given a new name (a default name is proposed). All stored data of type robtarget, used in the routine, will be mirrored and stored with a new name (the old name ending with _m). All immediate robtarget data, shown with an "*", in movement instructions will also be mirrored.

What does mirrored mean?

In general, all data of the type robtarget, used in the routine, will be mirrored. It makes no difference whether the robtarget data is declared as a constant (which it should be), as a persistent or as an ordinary variable. Any other data, e.g. of type pos, pose, orient, etc., will not be mirrored. Mirroring data only affects the initialization value, i.e. any current value will be ignored. This means that if a robtarget variable has been defined without an init value, this variable will not be mirrored.

The mirroring works as follows:

- The new routine is scanned for any local robtarget data, declared inside the routine with an init value. All such data's init values are mirrored.
- Then the new routine is scanned for any statement with one or more arguments of type robtarget.
- When such a statement is found, the following actions will take place:
  - If the argument is programmed with a reference to a local variable or a constant, this argument will be ignored, since it has already been mirrored as described above.
  - If the argument is programmed with an immediate robtarget data, shown with an asterisk "*", then this value will be mirrored directly.
  - If the argument is programmed with a reference to a global variable, persistent or a constant, defined outside the routine with an init value, then a duplicate is created and stored in the module with a new name (the old name ending with _m). The init value of this new data is mirrored, and after that the argument in the statement is changed to the new name. This means that the module data list will expand with a number of new mirrored robtarget data.

Error handlers or backward handlers, if any, in the routine, are not mirrored.

Mirror plane

The mirror function will mirror all positions, mentioned above, in the mirror plane, i.e. the mirrored position will be located symmetrically on the other side of the plane, relative to the original position. The mirror plane is always the xy-plane of an object frame, used for mirroring. This object frame is defined by a work object data, e.g. with the name MIRROR_FRAME. The work object MIRROR_FRAME uses, as all work objects, two frames for defining the object frame: the user frame and object frame.

The object frame is defined relative to the user frame, and the user frame is defined relative to the world frame. Usually, the user frame is set equal to the unity frame and, in such a case, the object frame is defined relative to the world frame (see Figure 33).

The mirror frame must be stated in the mirror dialogue.

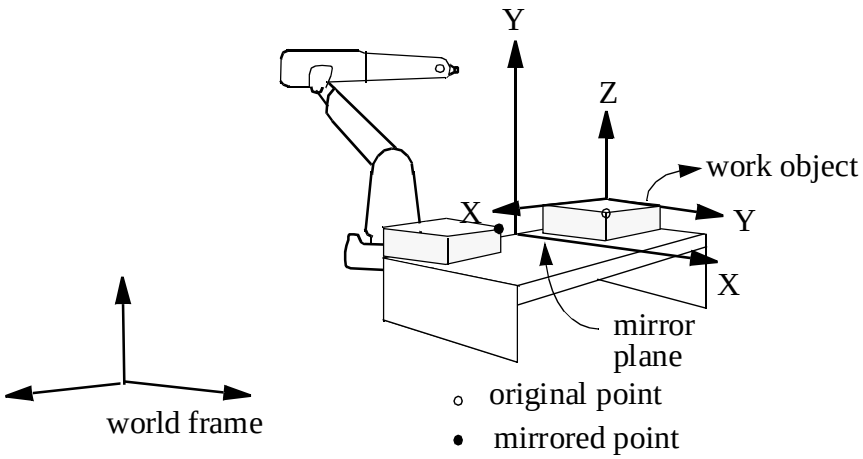


Figure 33 The mirror plane.

Work object

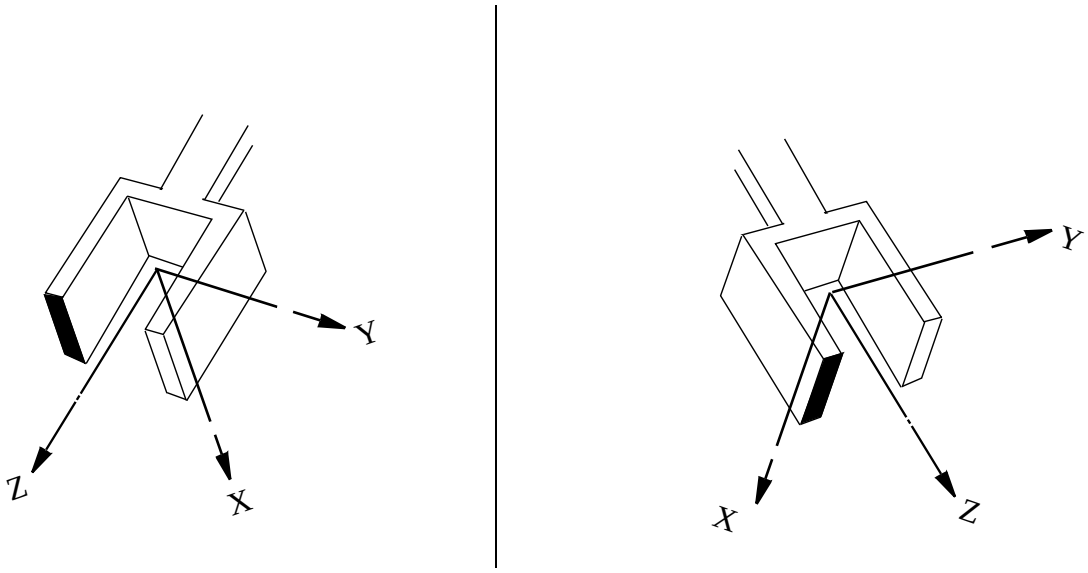
All positions which are to be mirrored are related to a specific work object frame. This means that the coordinates of the robtarget data are expressed relative to this work object frame (see the figure above). Furthermore, the mirrored position will be related to the same work object frame.

In the dialogue, before mirroring, this specific work object must be stated. This work object will be used as the reference frame for all variables that are to be mirrored.

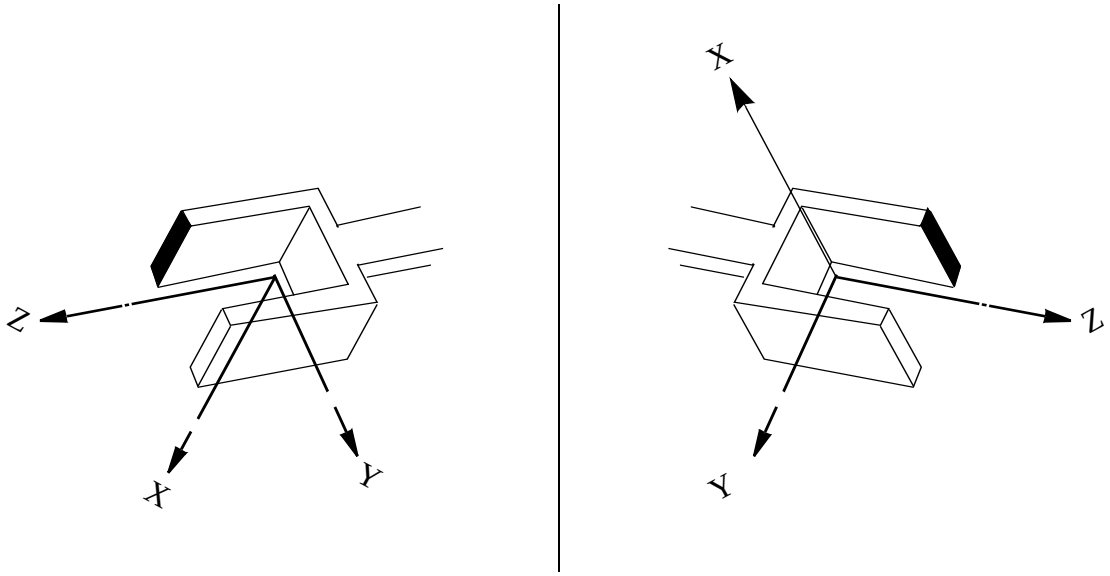
IMPORTANT:
Be sure to state the same work object as was originally used when defining the robtarget data, and which was used as a parameter in the movement instructions. If no work object was used, the wobj0 should be stated.

Mirroring of orientation

The orientation of the robtarget position is also mirrored. This mirroring of the orientation can be done in two different ways, where either the x and z axes are mirrored or the y and z axes (see Figure 34). The method used, x or y axis (the z axis is always mirrored), is dependent on the tool used and how the tool coordinate system is defined. In the mirror dialogue, the method must be stated.



Mirroring of x- and z-axes



Mirroring of y- and z-axes

Figure 34 Two different ways of mirroring.

Configuration

The configuration will not be mirrored, which means that, after mirroring, it has to be carefully checked by executing the path in test mode. If the configuration has to be changed, this must be done manually and the position corrected with a modpos command.

Mirror example 1, one robot

A programmed routine, org, is stored in the robot's memory. A mirrored copy of this routine is to be created and stored with the name mir in memory. All positions are related to the work object, wobj3. The mirror plane is known from three positions in the plane, p1, p2 and p3.

An original position in org, pos, is mirrored to pos_m (See Figure 35).

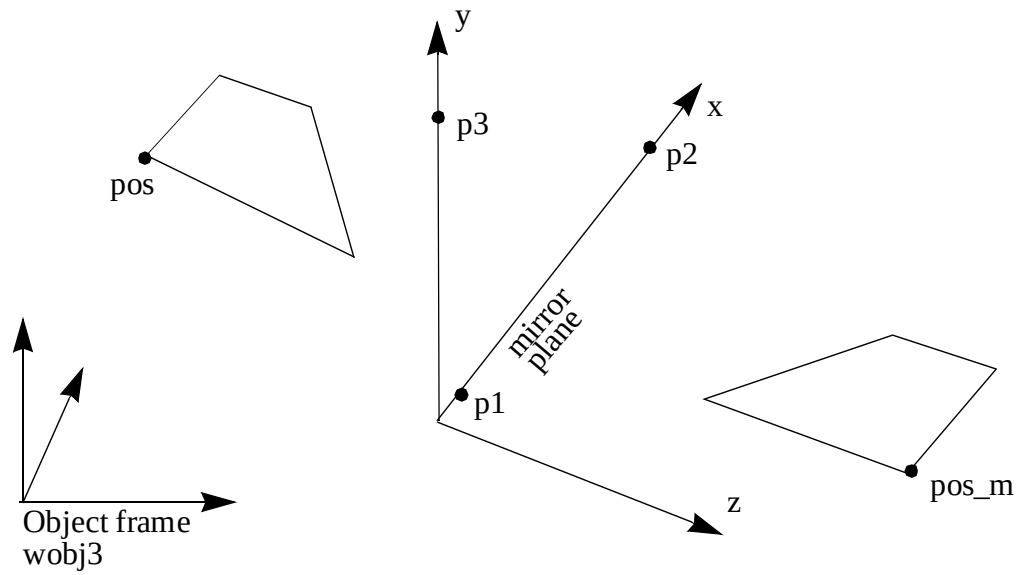


Figure 35 Mirroring of a routine, using one robot.

To perform this mirroring, the mirror frame must first be defined. To do this, start off by creating a new work object and name it mirror or whatever. Then, use the three points, p1 to p3, to define the object coordinate system with the help of the robot (see Chapter 10, Calibration).

After this, the routine, org, can be mirrored using wobj3 and mirror as input data.

Mirror example 2, two robots

In this case, a routine, org, created on one robot, is to be mirrored and used on another robot. Suppose that a spot welding robot, robot 1, is used for the left side of a car body. When the program for the left side is done, it should be mirrored and used again for the right side by robot 2.

The original program, org, is programmed relative to a work object, wobj1, which is defined with the help of three points, A, B and C on the left side of the car body, using the "3-point" method, (see Chapter 10, Calibration). The mirrored program, mir, is to be related to a corresponding work object, wobj1, defined by the corresponding points D, E and F on the right side of the car body. Wobj1 for robot2 is defined with robot2, using the same "3-point" method. Note that since the points D, E, F are reflected images of points A, B and C, the wobj1 for robot2 will also be mirrored. One of the consequences of this is that the z-axis will point downwards.

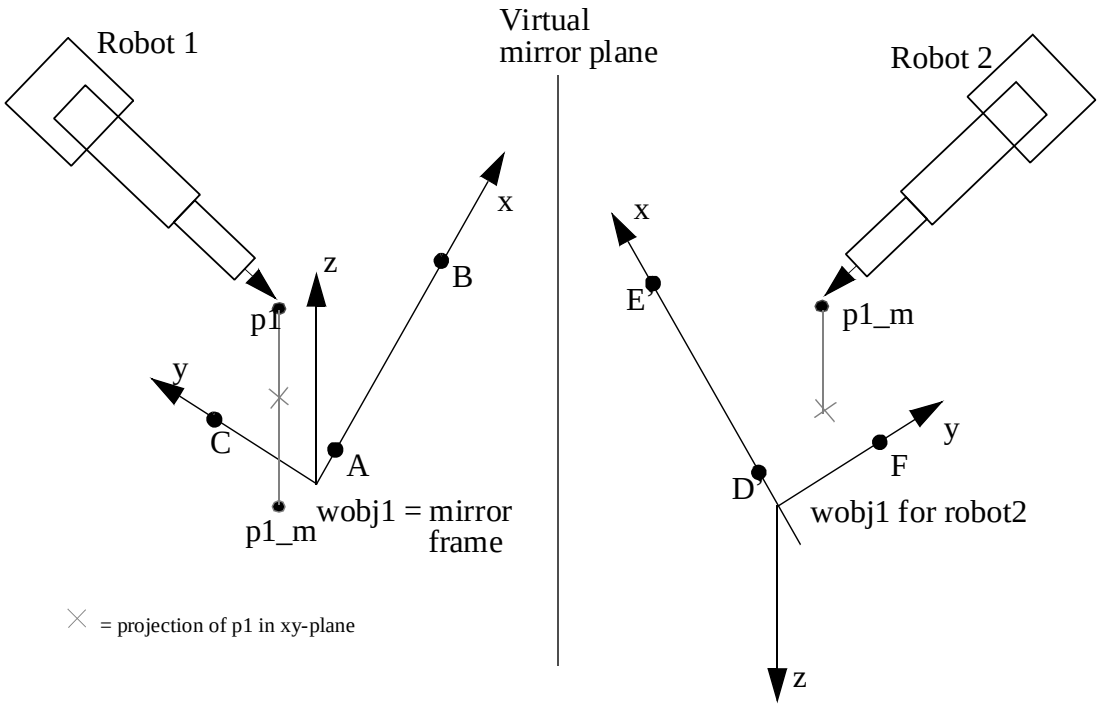


Figure 36 Mirroring of a routine, using two robots.

After the work object, wobj1, has been defined, all programming is done in this frame. Then the program is mirrored using the same wobj1 frame as the mirroring frame. A position, p1, will be mirrored to the new position p1_m.

After this, the mirrored program is moved to robot 2, using the work object wobj1, as described above. This means that the mirrored position, p1_m, will be “turned up” as if it were mirrored in a “virtual” mirror plane between the two robots (see Figure 36).

Mirror function dialogue

• Choose **View: Routines**

• Select the routine to be mirrored.

• Choose **Special: Mirror**

A dialog box appears (see Figure 37).

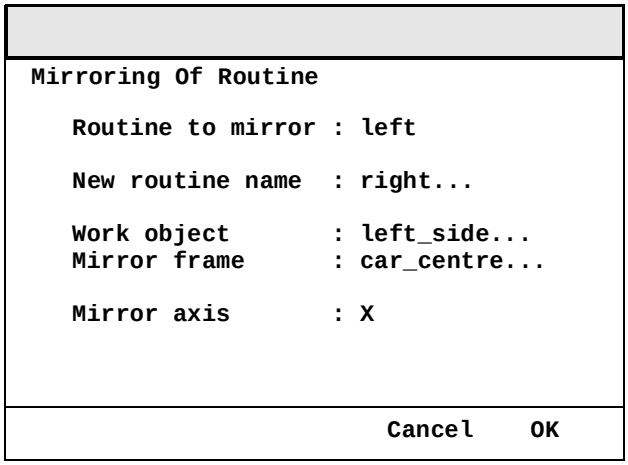


Figure 37 Mirror function dialogue.

• Define how the mirroring is to be performed in the fields below.

Field	Description
Routine to mirror	The name of the routine that will be mirrored.
New routine name	The mirrored routine will be given this name. If the Enter ↵ key is pressed when this field is selected, a text input dialog will be displayed.
Work object	The work object to be used when applying the mirror function on robtarget variables. If the Enter ↵ key is pressed, the work object selection dialogue will be displayed.
Mirror frame	The frame to be used as the mirror plane. The frame is of the type wobjdata. If the Enter ↵ key is pressed, a mirror frame selection dialogue will be displayed.
Mirror axis	Specifies the mirroring of orientation. When this field is selected, the function key bar shows the alternatives X and Y. The mirroring of orientation is then selected by pressing the corresponding function key.

• Start the mirroring with ***OK***.

10 Creating Data

10.1 What is data?

Data is used to store values that can be used at a later stage in the program. Data is grouped into different data types that describe its contents and its field of application.

Data type	Used for:
num	Numeric values (registers, counters)
bool	Logical values (true or false)
robtarget	Position data
tooldata	Tool data (see Chapter 10, Calibration)
wobjdata	Work objects (see Chapter 10, Calibration)
pose	Program displacement frames (see Chapter 10, Calibration)

For more detailed information on data and its contents, see the appropriate data type in Chapter 14 in this Manual - *Data Types*.

Data must be defined (declared) before it can be used. However, depending on the configuration of the robot, there is usually a number of predefined data.

Data can be defined as constants, variables or persistents:

- The value of a *constant* can only be changed manually.
- A *variable* can also be changed by the program, but its initialisation value is automatically set when:
  - the program is read from diskette or the like,
  - the program is started from the beginning, i.e. from the first instruction in the main routine,
  - the program pointer is moved to the beginning of a routine by choosing **Test: Move PP To Routine**.
- A *persistent* can be described as a variable whose initialisation value is constantly updated so that it corresponds to the current value. Thus, its value is not changed when the program is started from the beginning. If the program is output to a diskette, the new initialisation value is stored.

10.2 The *Program Data* window (used to manage data)

• Choose **View: Data** to open the Program Data window.

The window displays all data of the type last selected. The current values are also displayed (see Figure 38).

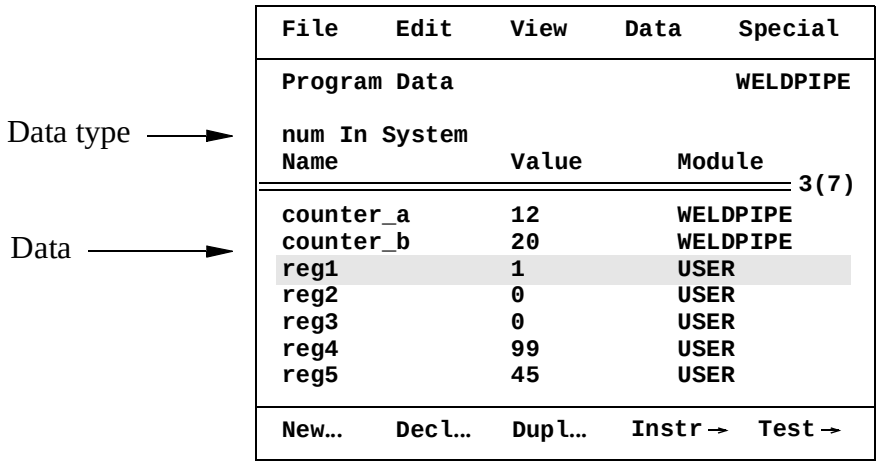


Figure 38 All data of a given type are displayed in the Program Data window.

To choose a new data type in the Program Data window

• Open the window *Program Data Types* by choosing **View: Data Types**.

The *Program Data Types* window opens and displays all data types that have at least one declared data (see Figure 39).

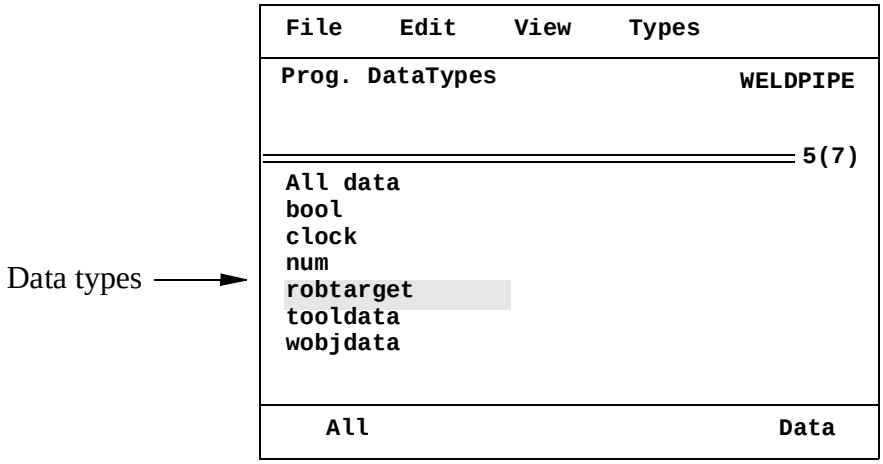


Figure 39 The Program DataTypes window is used to change the data type.

• Select the desired data type and press Enter ↵. If the desired type is not displayed in the window, you can call up all data types by pressing ***All type*** or choosing **Types: All Types**.

All data can be chosen by selecting ***All data***.

Data for a selected type can be chosen by pressing ***Data*** or Enter ↵.

10.3 Creating new data

• Open the *Program Data* window by choosing **View: Data ...**

The *Program Data* window is opened and displays all data of the type last selected.

If you wish to create data of a type other than that displayed, choose **View: Data Types**, select the desired data type and press Enter ↵.

• Press the function key ***New***.

A dialog box appears, displaying the name of the data (see Figure 40). The name of the data is set to *xxxN* where *xxx* describes the data type and *N* is a number incremented each time this type of data is created. The first data of the type clock is named *clock1*, the second, *clock2*, etc. Some data types are abbreviated, e.g.:

Data type	Predefined name	Data type	Predefined name
num	reg*N*	loaddata	load*N*
robtarget	p*N*	tooldata	tool*N*
bool	flag*N*	speeddata	speed*N*

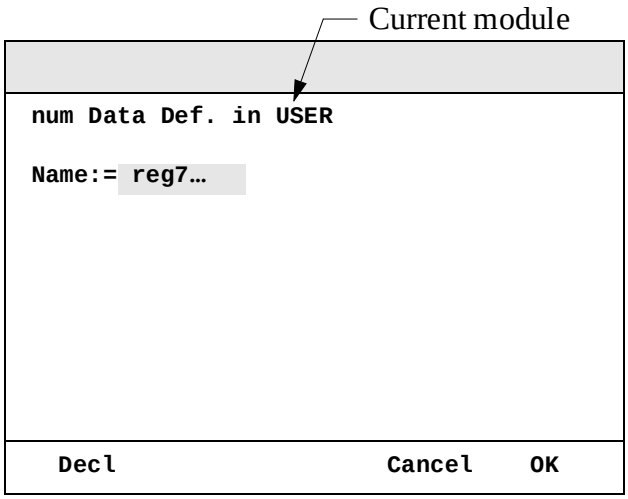


Figure 40 New data is created.

• Change the name by pressing Enter ↵ and specify a new name.

The data will automatically be given characteristics that are best suited to the current type, but these can be changed when necessary.

Normally, data is stored as a part of the program. However, when data is to be present in the memory, irrespective of which program is loaded, it is stored in the system module *User*. Examples of this type of data are:

- tools and work objects; changing this data will affect all programs.
- registers and other data that are not to be initialised when a program changes.

When you wish to save in the current module and with standard characteristics, you can finish by pressing ***OK***. In other cases the characteristic must be defined.

• Press the function key ***Decl***.

A dialog box appears, displaying the basic data declaration (see Figure 41).

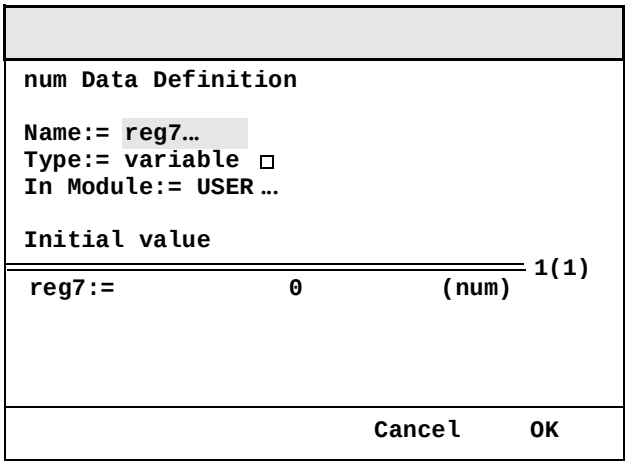


Figure 41 A data declaration includes the name and characteristics of the data.

• Select the appropriate field and specify the desired characteristics by:

- pressing Enter ↵ and specifying the desired alternative in the dialog box that appears (fields marked with ...)
- choosing an alternative using the function keys (fields marked with □)
- specifying the value directly using the numeric keyboard (numeric initial value).

Field	Description
Name	The name of the data (a maximum of 16 characters).
Type	Specifies whether the data is to be a constant (***Const***), variable (***Var***) or persistent variable (***Pers***).
In Module	The module in which the new routine will be used.
Initial value	A value assigned to the data when, e.g. reading from a diskette. Change the value by pressing and enter the new initial value.

• Choose ***OK*** to terminate the definition.

Tip It is sometimes easier to create new data by duplicating and changing existing data.

10.4 Duplicating data

• Open the window *Program Data* by choosing **View: Data**.

• Select the data to be duplicated.

• Press the function key ***Dupl***.

• Specify the new name in the dialog box that appears.

• Choose ***OK*** to confirm the duplication.

10.5 Storing position data using the robot

- Open the *Jogging* window and specify the tool and work object on which the position is to be based.
- Jog the robot to the desired position.
- Create new data as described in *Creating Data* on page 42. Specify the data type *robtarget*.

The current position of the robot will be automatically stored as an initial value.

10.6 Routine data

Normally, data – *program data* – can be accessed from anywhere in the program. Data can also be linked to a specific routine – *routine data* – and, in this case, exists locally within the routine.

- Open the *Program Data* window by choosing **View: Data**.
- Choose **Data: In Routine ...**

The window will then display the routine data for the current routine. The window is identical to the window shown in Figure 38, except that it displays the routine name after the program name.

Now you can create and change routine data in the same way as for program data.

11 Changing Data

11.1 Viewing and possibly changing the current value

- Select the desired data in an instruction.
- Choose **Edit: Value**.

A dialog box will appear, displaying the current value (see the example in Figure 42). For more detailed information on the meaning of the various components, see the appropriate data type in the RAPID Reference Manual.

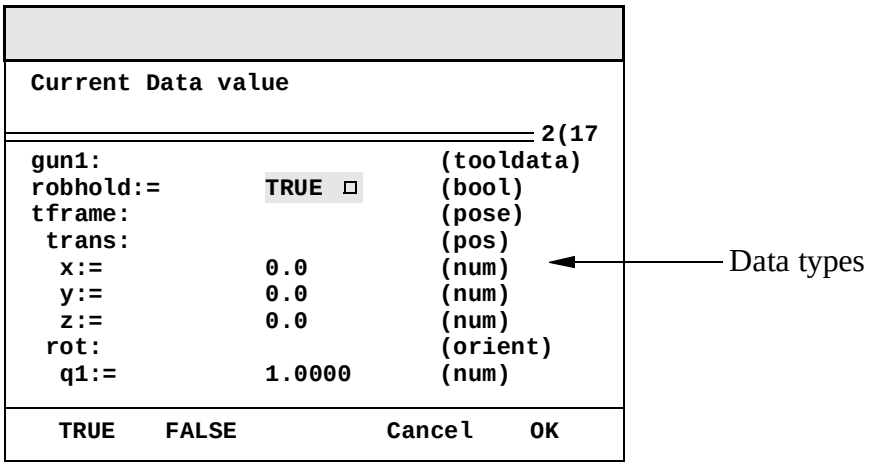


Figure 42 The dialog box used to change values.

- Change the value by selecting the desired field, then:
  - Choose an alternative using the function keys.
  - Specify the value directly using the numeric keyboard.
- Choose ***OK*** to confirm the change.

You can also open the dialog box as follows:

- Choose **View: Data**.
- Select the desired data. If you wish to view data of a type other than that displayed, choose **Data: Datatypes** and select the desired data type.
- Press Enter ↵ or choose **Data: Value**.

Continue as above when the dialog box appears.

11.2 Changing data names or declarations

- Choose **View: Data**.
- Select the desired data. If you wish to view data of a type other than that displayed, choose **Data: Datatypes** and select the desired data type.
- Press the function key ***Decl***.

A dialog box appears, displaying the data declaration.

- Change the name and declaration as described in *Creating Data* on page 42.
- Choose **OK** to confirm the change.

11.3 Deleting data

- Choose **View: Data**.
- Select the desired data.
- Press Delete ⌫.
- Press ***OK*** to confirm the deletion.

12 Error Handling

Each routine has an error handler that can be programmed to deal with any errors that occur during program execution. In this way, some errors (listed below) can be dealt with automatically by the program:

- when no search stop is obtained during a search,
- when a file cannot be opened,
- when there is division by 0.
- Other errors are listed under *Data Types - errnum - Predefined Data* (RAPID Reference Manual).

The error handler is programmed in the normal way using RAPID instructions. When an error occurs, a jump is made to the error handler in the routine in which the error occurred.

If there is no error handler, a jump is made instead to the error handler in the routine that called the routine in question. A general error handler for the whole program can therefore be created in the main routine. If there is no error handler when an error occurs, program execution will stop and an error message will be displayed.

The error can then be remedied in the error handler and the program can be automatically restarted as in the example in Figure 43.

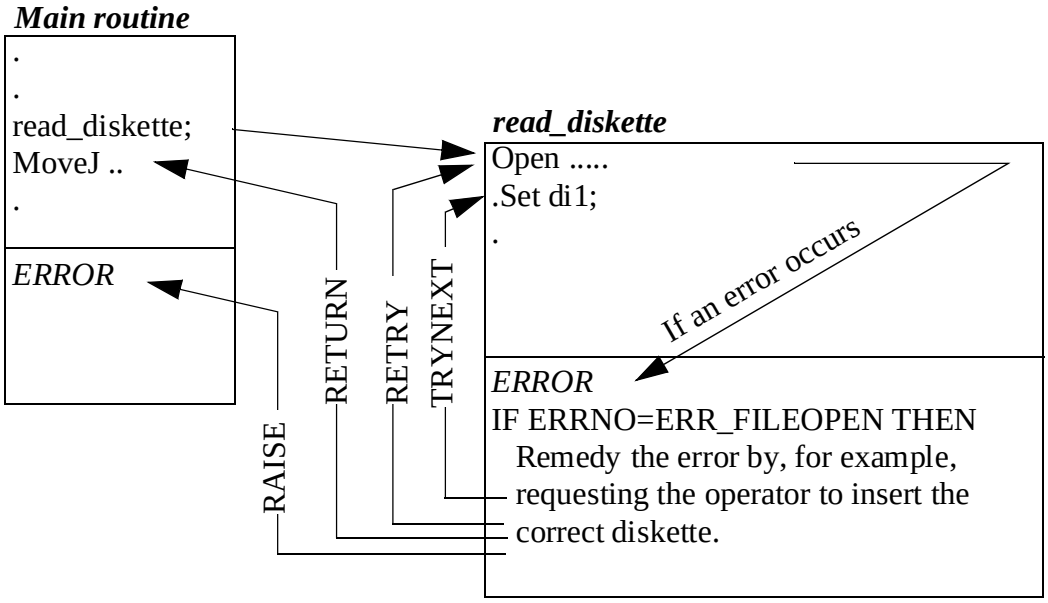


Figure 43 The program can be restarted from the error handler in various ways.

If the program cannot read a diskette, a jump is made to the error handler of the routine, where the error is remedied. The program can then be restarted by re-executing (RETRY) the instruction that caused the error, executing the next instruction (TRYNEXT) or by returning (RETURN) to the calling routine. The error can also be remedied in the error handler of the main routine (RAISE).

To create an error handler

- Choose **View: Routines**.
- Choose the routine to which the error handler is to belong.
- Choose **Routine: Add Error Handler**.

To program the error handler

- Choose the routine to which the error handler is to belong.
- In the *Routine* window: Choose **Routine: Error Handler.**
  In other windows: Choose **View: Error Handler**.
- Program the error handler in the usual way.
- Return to the main part of the routine by choosing **View: Instr**.

To remove an error handler

- Choose **View: Routines**.
- Choose the routine to which the error handler is to belong.
- Choose **Routine: Remove Error Handler**.

13 Using Modules

13.1 What is a module?

The robot program can be subdivided into *program modules*, each module containing a group of routines and data. In addition to this, system modules which are always present in the memory, can be used (see Figure 44).

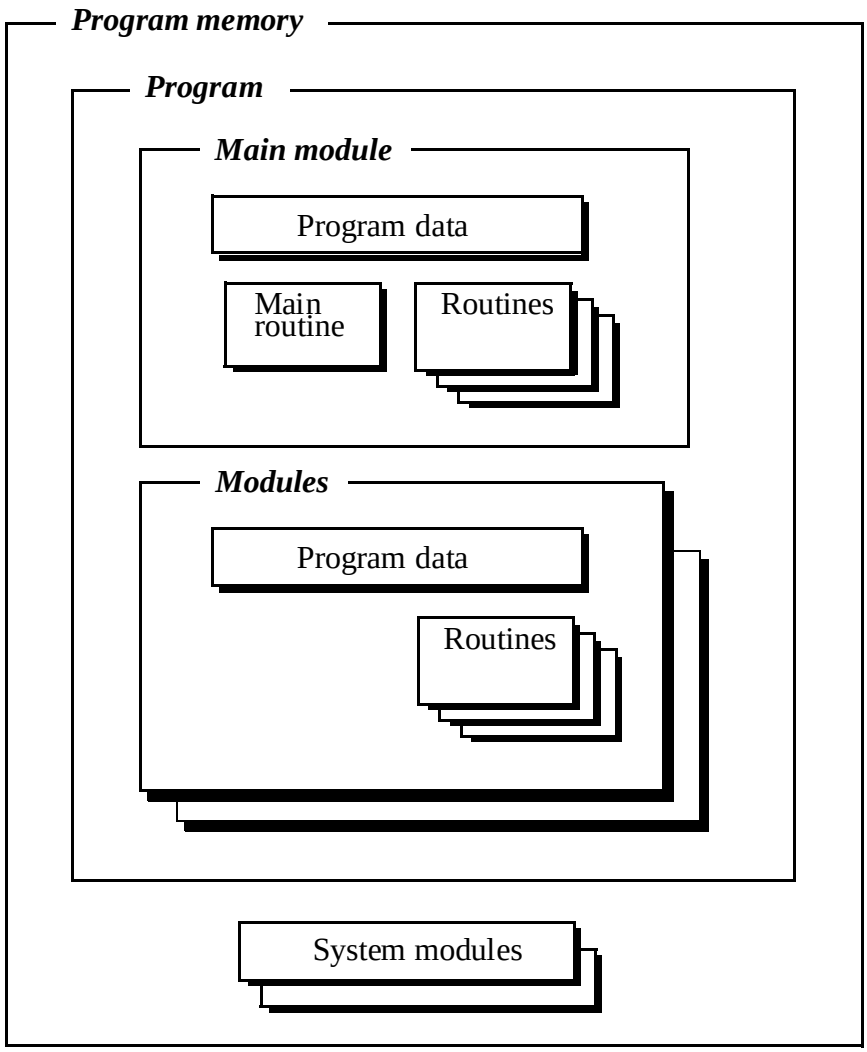


Figure 44 Routines and data can be grouped together to form modules.

The entire program or separate modules can be stored on diskette or some other type of mass memory. System modules are automatically loaded when the system is cold-started.

A module can include, for example:

- general routines for many different installations,
- positions generated via CAD,
- routines for a certain type of external equipment, such as a workpiece manipulator.

System modules can, for example, include general data (e.g. tool data) for all programs used in the same robot.

The main routine of the program is located in one of the modules (the module with the same name as the program).

Both program and system modules work in the same way once they have been loaded into the memory. All modules can normally be edited using the teach pendant, but, as system modules are often write-protected, the write protection must first be removed.

13.2 Choosing modules

• Choose **View: Modules**.

The window *Program Modules* displays all modules present in the program memory (see Figure 45).

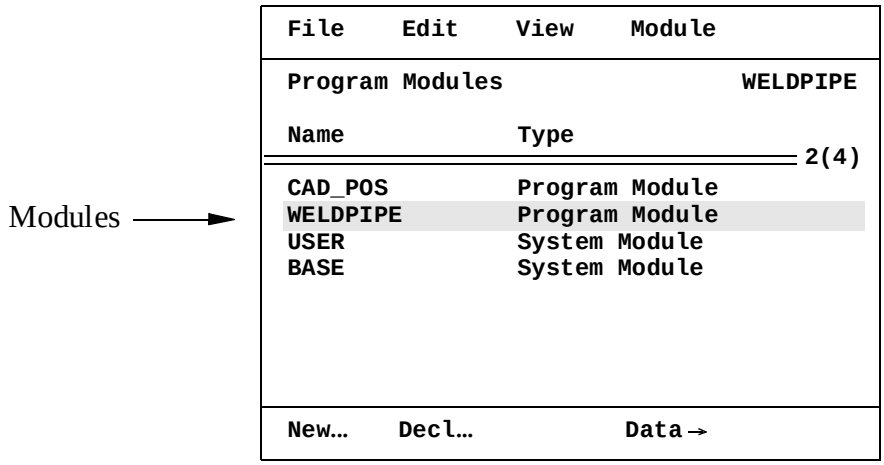


Figure 45 The Program Modules window displays all modules in the task program.

• Select the desired module.

• Press Enter [↵].

The *Program Routines* window, in which you can choose the desired routine, opens.

13.3 Creating a new module

• Open the window *Program Modules* by choosing **View: Modules**.

• Press the function key ***New***.

A dialog box appears, displaying the basic module declaration (see Figure 46). The name of the routine is set to *moduleN*, where *N* is a number incremented each time a routine is created.

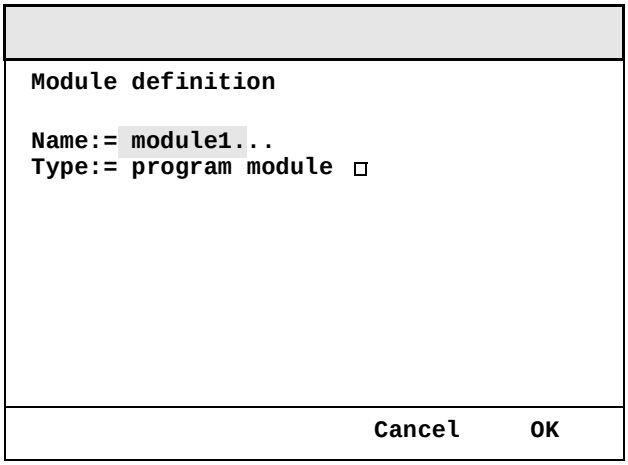


Figure 46 A module declaration specifies the name and characteristics of a module.

• Change the name and characteristics of the module by selecting the appropriate field, then:

- Press Enter ↵ and specify the desired alternative in the dialog box that appears on the display (fields marked with...).
- Choose an alternative using the function keys (fields marked with □).

Field	Description
Name	The name of the module (a maximum of 16 characters).
Type	Specify whether the module is to be a program or system module.

• Press ***OK*** to end the module declaration.

13.4 Changing the name or declaration of a module

• Choose **View: Module**.

• Select the desired module.

• Press the function key ***Decl***.

A dialog box appears, displaying the module declaration.

• Make whatever changes you wish to make (see *Creating a new module* on page 51).

• Choose ***OK*** to confirm the change(s).

13.5 Reading a program module from diskette or some other type of mass memory

• Choose **File: Open**.

A dialog box appears, displaying all modules and programs in the current directory (see Figure 47).

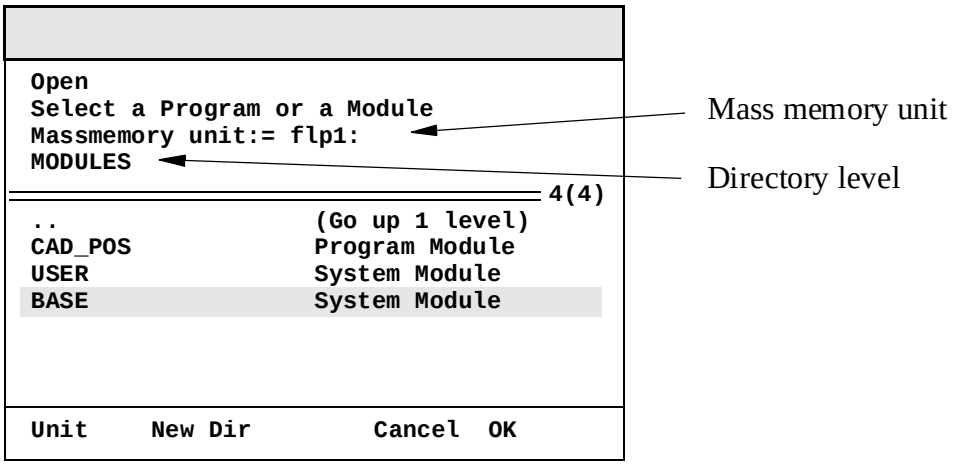


Figure 47 The dialog box used to read modules.

• If necessary, change the mass memory unit by pressing ***Unit*** until the correct unit is displayed.

• Select the desired program module. Move up or down in the directory by choosing either '. .' (up), or the desired directory (down) and press Enter ↵.

• Choose ***OK*** to confirm.

The specified module will then be read to the robot memory and added to the rest of the program.

13.6 Deleting program modules from the program

• Open the window *Program Modules* by choosing **View: Modules**.

• Select the desired module.

• Press Delete ⌫.

Note An answer must be given to the password check and confirmation dialog if they have been set to active in the configuration. The correct password must be entered at the password prompt. The confirmation dialog is confirmed by Yes or No.

13.7 Listing all routines in all modules

Usually only the routines contained in the current module are displayed in the *Program Routines* window. You can, however, change this so that all routines in all modules are displayed.

• Open the window *Program Routines* by choosing **View: Routines**.

• Choose **Routine: In System**.

To list only the routines in the current module again, choose **Routine: In Module**.

13.8 Duplicating a routine from one module to another

• Choose the module in which the new routine is to be included.

• List all routines by choosing **Routine: In System** in the *Program Routines* window.

• Select the routine to be duplicated.

• Continue in the normal way, as described in *Duplicating a routine* on page 10.

13.9 Listing all data in the current module

Usually the data contained in all modules is displayed in the *Program Data* window. You can, however, change this to display only the data in the current module.

• Open the window *Program Data* by choosing **View: Data**.

• Choose **Data: In Module**.

To list all program data in all modules again, choose **Data: In System**.

13.10 Duplicating data from one module to another

Data can be duplicated from one module to another. Routine data cannot, however, be duplicated.

• Choose the module in which the new data is to be included.

• Select the data to be duplicated in the *Program Data* window.

• Continue in the normal way, as described in *Duplicating data* on page 45.

13.11 Saving modules on diskette or some other type of mass memory

To save a module that has been stored previously

• Open the window *Program Modules* by choosing **View: Modules**.

• Select the module to be saved.

• Choose **File: Save Module**.

The module is duplicated to mass memory and replaces the last version saved.

To save under a new name

• Open the window *Program Modules* by choosing **View: Modules**.

• Select the module to be saved.

• Choose **File: Save Module As**.

A dialog box appears, displaying all modules and programs in the current directory (see Figure 48).

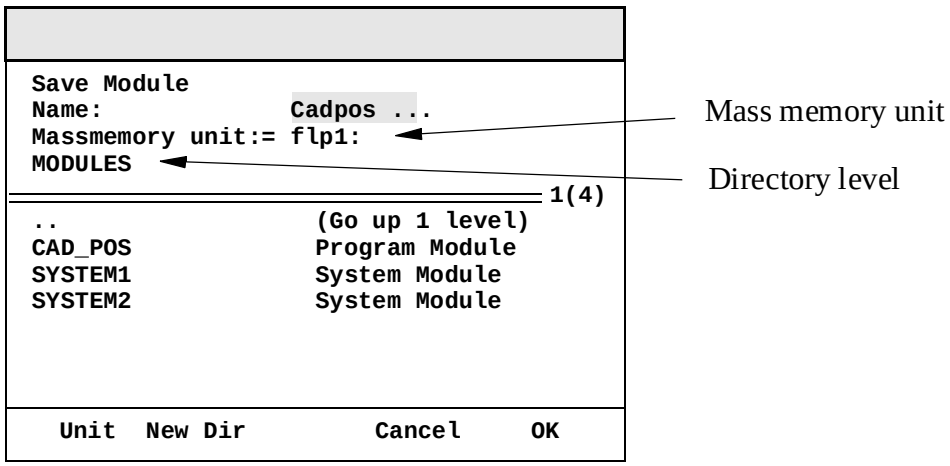


Figure 48 The dialog box used to store modules.

• If necessary, change the mass memory unit by pressing ***Unit*** until the correct unit is displayed.

• Choose the directory in which the module is to be saved. Move up or down in the directory by choosing either '. .' (up), or the desired directory (down) and press Enter [↵]. Create a new directory by pressing ***New Dir***.

• Press Enter [↵] when the field ***Name*** is selected.

• Specify the new name (using the numeric keyboard) in the dialog box that appears. Press ***OK*** when you have finished entering the new name.

• Choose ***OK*** to confirm the save.

13.12 Calling up the complete module list

• Choose **View: Modules**.

• Select the desired module.

• Choose **Module: Module List**.

The complete module is displayed, including its data declarations and routines. It cannot, however, be changed.

• Exit the module list by pressing ***OK***.

14 Preferences

14.1 Defining the *Most Common* instruction pick list

You can define the contents of the *Most Common* instruction pick list to obtain a pick list of the instructions you use most.

- Choose **File: Preferences**.
- Select, for example, ***Most Common SetUp 1***.
- Press Enter ↵.

All instructions and procedures are displayed. Those included in the pick list are marked with an *x* to the left of their names (see Figure 49).

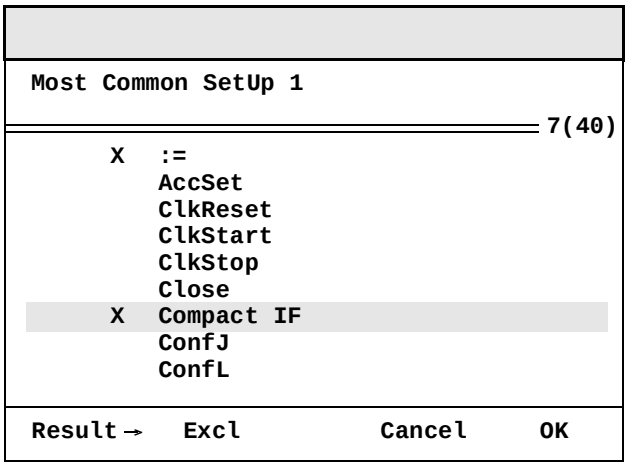


Figure 49 You specify the instructions to be included in the list in the Most Common Setup dialog box.

- Add an instruction by selecting the appropriate instruction and pressing ***Incl.*** That instruction will then be marked with an *x* to its left.
- Remove an instruction by selecting the appropriate instruction and pressing ***Excl.*** The instruction will still be displayed in the window but the *x* to its left will disappear.
- Press ***Result***.

The instructions included in the pick list are displayed (see Figure 50).

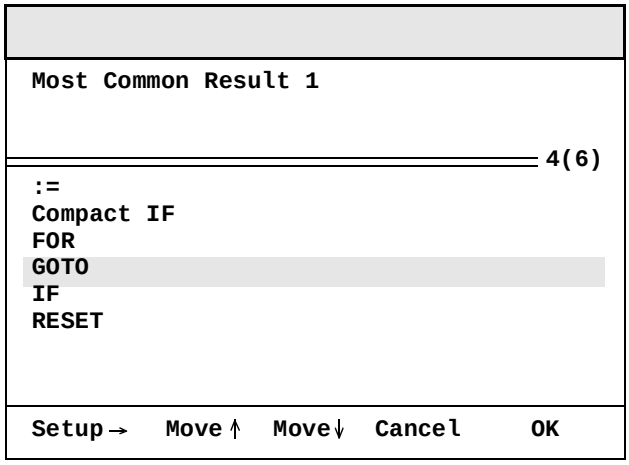


Figure 50 You specify the order of the instructions in the list in the Most Common Result dialog box.

- Change the order of the instructions using ***Move*** ↑ and ***Move*** ↓. ***Move*** ↑ moves the selected instruction up one step and ***Move*** ↓ moves it down one step.
- When the definition is ready, press ***OK***.
  To return to the *Most Common Setup* dialog box, press ***Setup*** instead.

The current Most Common list is automatically chosen as the active pick list. The various Most Common lists can be chosen from the IPL2 menu in the Program Instr window.

Note This definition is stored in the system parameters (topic Teach Pendant) which should be saved from the *System Parameters* window.

CONTENTS

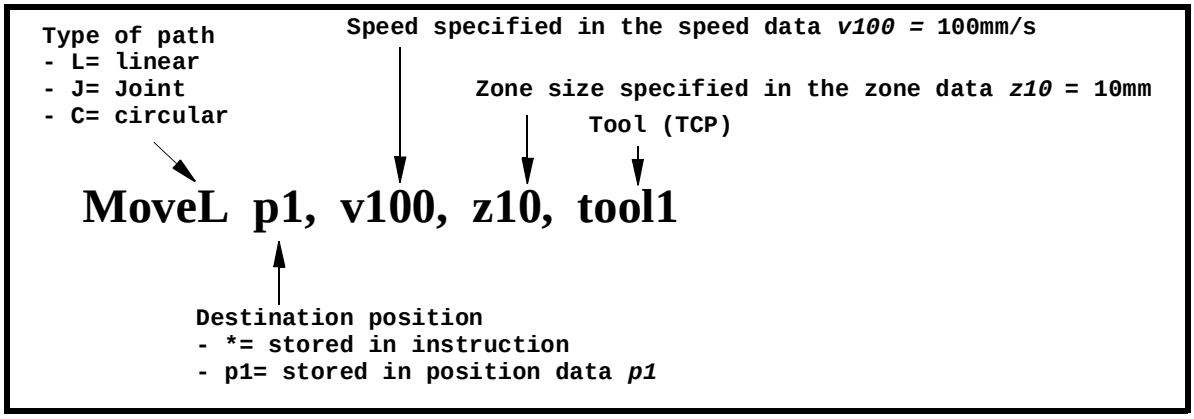
Page

1 Programming a Position

1.1 Positioning instructions

A positioning instruction contains the following information:

- Type of path (e.g. linear, joint motion)
- The destination position to which the robot is to move
- Speed
- Zone size (accuracy), i.e. how close the robot must be to the destination position before it can start to move towards the next position. If *fine* is chosen, the robot moves to the position.
- Current tool (TCP).



The speed and zone size refer to different data, which includes the desired speed, zone size in mm/s, etc. You can create and name this data yourself, but, the most commonly used values are already available.

You specify the tool – its dimensions and weight – in the tool data (see Chapter 10, Calibration). The TCP of the tool is moved to the specified destination position when the instruction is executed (see Figure 1).

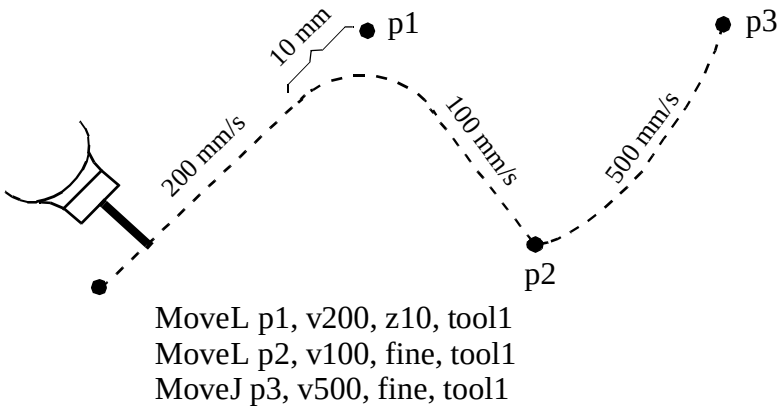


Figure 1 Positioning the robot.

Apart from these arguments, a positioning instruction may contain optional arguments, e.g. arguments used to specify the positioning time. See the appropriate instruction in RAPID Reference Manual for more details.

• Jog the robot to the desired destination position.

• Call up the instruction pick list by choosing **IPL1: Motion&Process**.

The program and specified pick list will then appear in the window (see Figure 2).

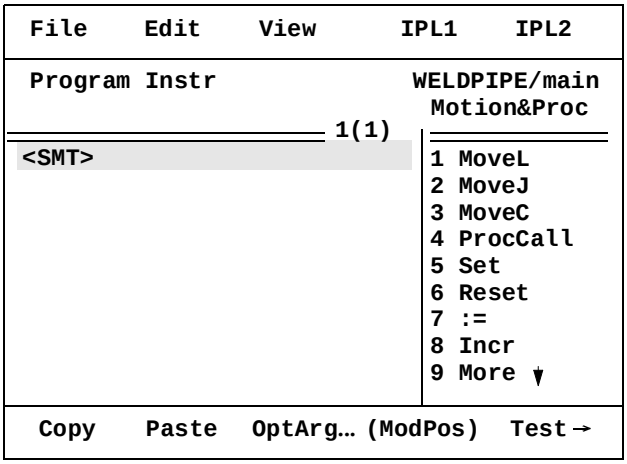


Figure 2 The dialog box used to program positioning instructions.

• Choose the desired instruction by pressing the appropriate numeric key.

The instruction will be added directly to the program, as illustrated in Figure 3. The arguments are set automatically.

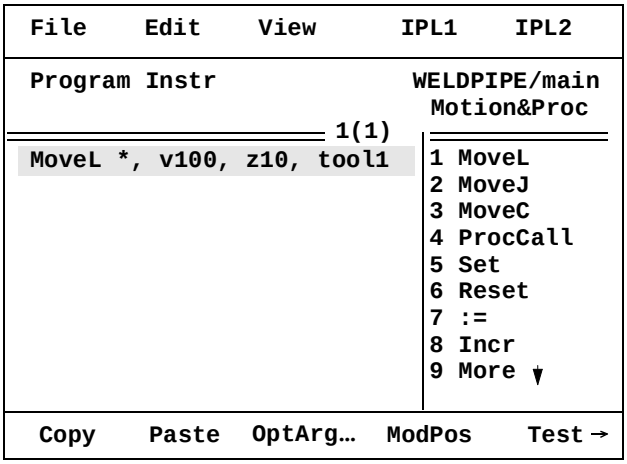


Figure 3 A positioning instruction is added directly to the program.

If the correct argument was chosen, the instruction is now ready for use. However, we will continue and change the speed and zone size.

• Select the argument you wish to change (*v100* in this example).

• Press Enter [↵].

The dialog box, used to program instruction arguments, appears. The selected argument is marked with a ? in front of it (see Figure 4). The lower part of the box displays all available speed data that can be selected.

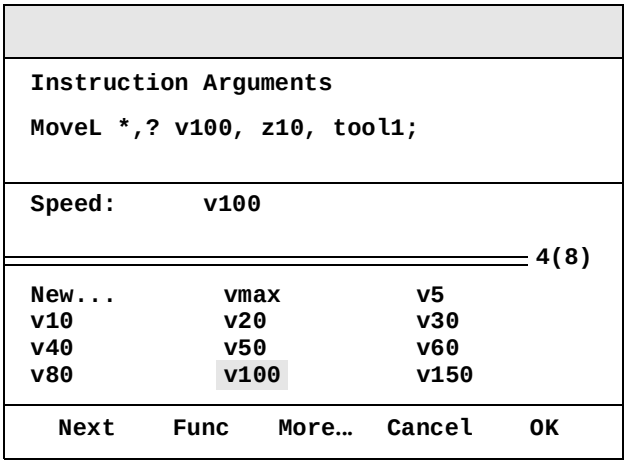


Figure 4 The dialog box used to change the speed.

• Select the desired speed.

• Go to the next argument (zone data) by pressing ***Next***.

All available zone data will be displayed (see Figure 5).

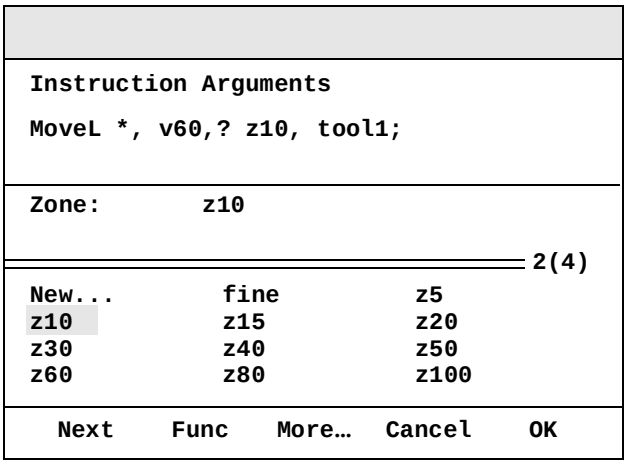


Figure 5 The dialog box used to change zone data.

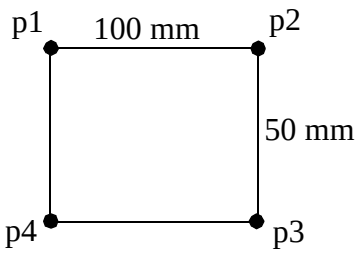
• Select the desired zone size.

• Choose ***OK*** to confirm the change.

The instruction is now ready for use.

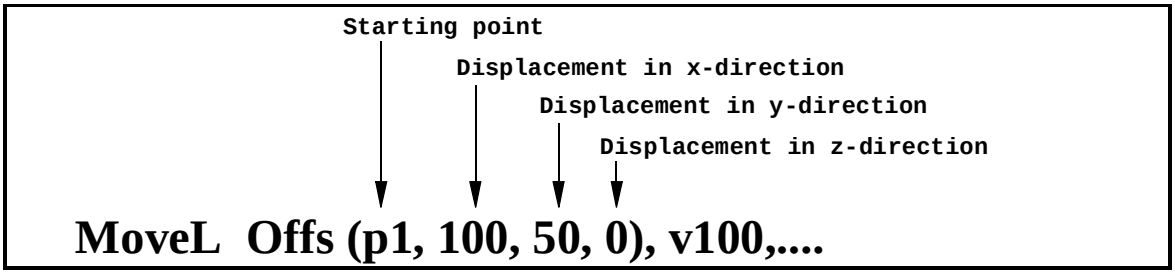
1.2 Programming an offset

Sometimes it is easier to define a position as an offset from a given position. If, for example, you know the exact dimensions of a work object, it will only be necessary to jog to one position (see Figure 6).



MoveL p1, ...	MoveL p1, ...
MoveL p2, ...	MoveL Offs (p1, 100, 0, 0), ...
MoveL p3, ...	MoveL Offs (p1, 100, 50, 0), ...
MoveL p4, ...	MoveL Offs (p1, 0, 50, 0), ...
MoveL p1, ...	MoveL p1, ...

Figure 6 Two different ways of programming a movement.



• Program a positioning instruction as described in *Programming a Position* on page 3.

• Select the position argument and press Enter ↵ .

• Press ***Func***.

• Select the function ***Offs*** and press Enter ↵ .

A dialog box appears in which you can enter the arguments of the function (See Figure 7).

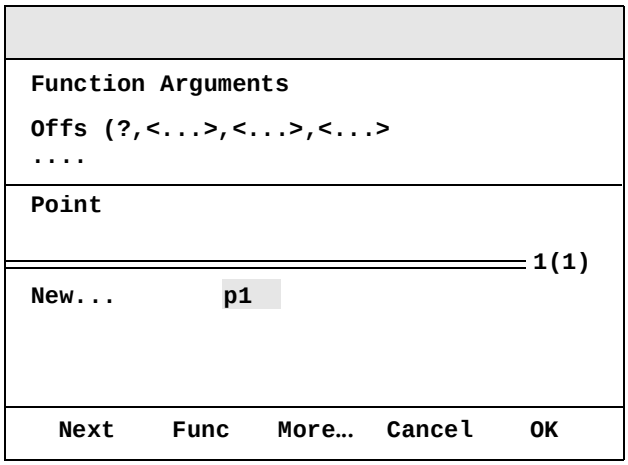


Figure 7 The dialog box used to set an offset.

• Select the starting point.

• Press ***Next***.

• Enter the offset (the offset value) in the x-direction using the numeric keyboard.

• Press ***Next***.

• Enter the offset in the y-direction using the numeric keyboard.

• Press ***Next***.

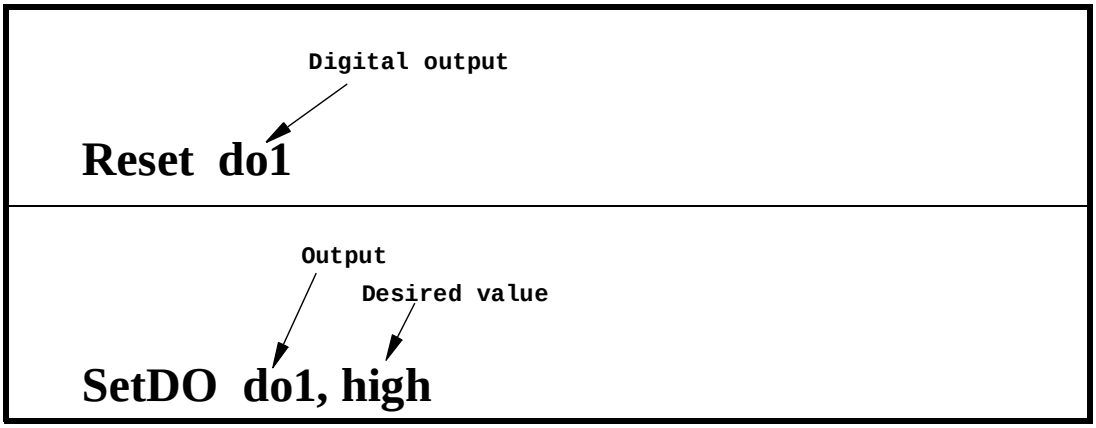
• Enter the offset in the z-direction using the numeric keyboard.

• Press ***OK***.

2 Changing the Value of an Output

An output instruction contains the following information:

- information on the output to be changed,
- information on the desired value.



• Call up the instruction pick list for I/O instructions by choosing **IPL1: I/O**.

• Choose the desired instruction by pressing the appropriate numeric key.

You must now specify the output to be changed. All the different robot outputs are displayed for this purpose (see Figure 8).

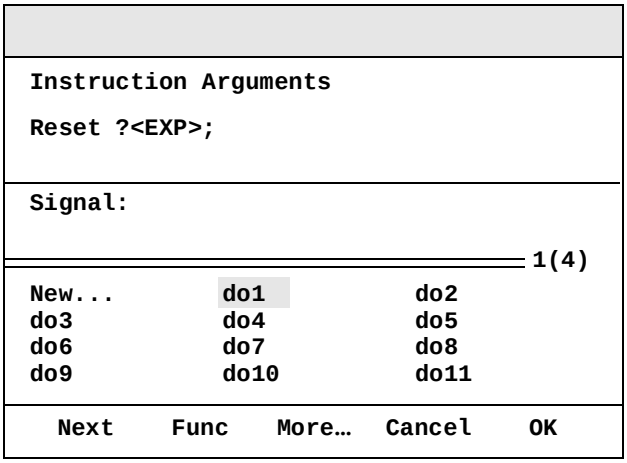


Figure 8 The dialog box used to define an output.

• Select the desired output.

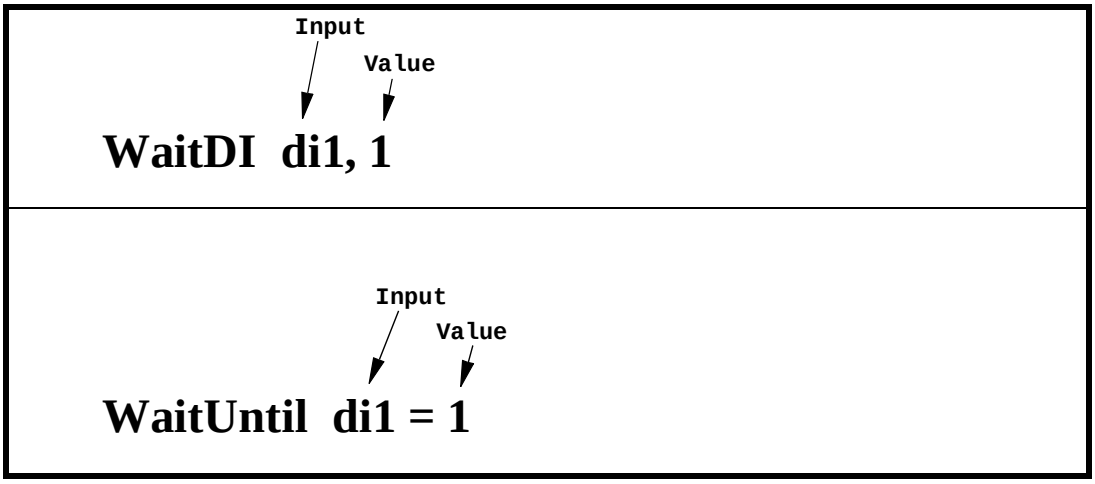
• Choose ***OK*** to confirm.

3 Waiting

3.1 Wating for an input

A wait-until-input instruction contains the following information:

- the name of the input,
- the input value necessary for program execution to continue.



The *WaitUntil* instruction can also be used to wait for several inputs.

• Choose **IPL1: Various**.

• Select the instruction *WaitDI*.

You must now specify the condition that must be satisfied if program execution is to continue. You do this using the dialog box illustrated in Figure 9.

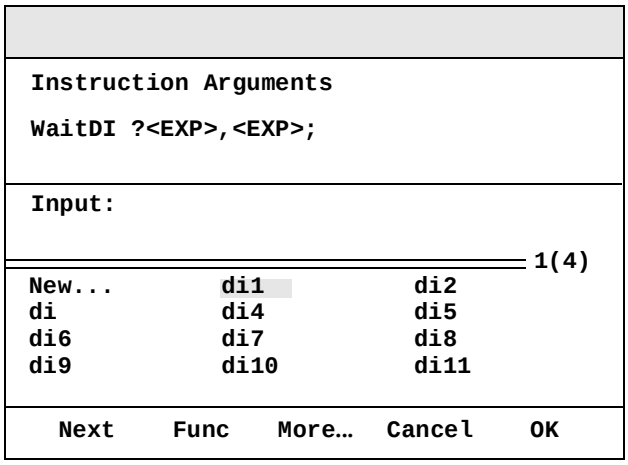


Figure 9 The dialog box used to define an input.

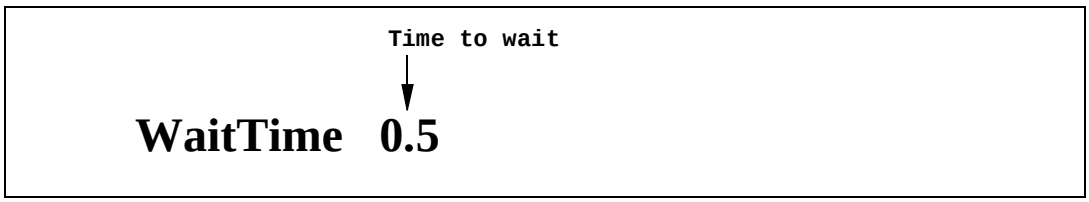
• Select the desired input.

• Choose ***Next*** to define the next argument, i.e. the value of the input.

• Enter the input value using the numeric keyboard.

• Press ***OK*** to confirm.

3.2 Waiting a specific amount of time



- Choose **IPL1:Various**.
- Select the instruction *WaitTime.*

A dialog box appears in which you can enter the time (see Figure 10).

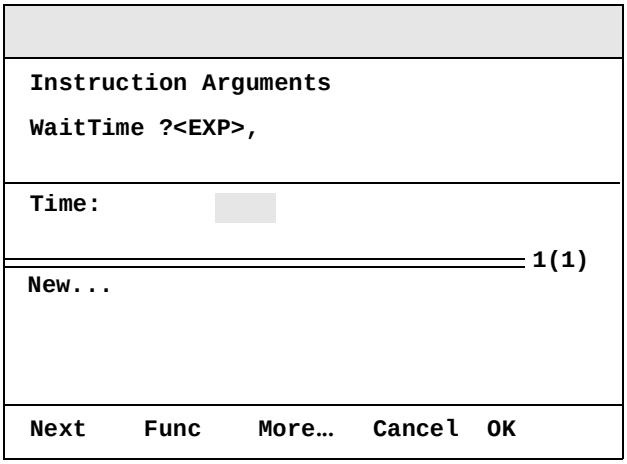


Figure 10 The dialog box used to define WaitTime.

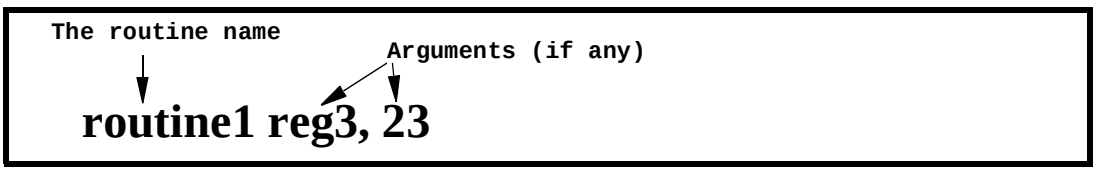
- Enter the time using the numeric keyboard.
- Press ***OK*** to confirm.

4 Controlling the Program Flow

4.1 Calling a subroutine

A call instruction contains the following information:

- information on the routine to be called,
- information on any arguments.



When this instruction is executed, the called routine will be executed. Following this, execution will return to the calling routine (see Figure 11).

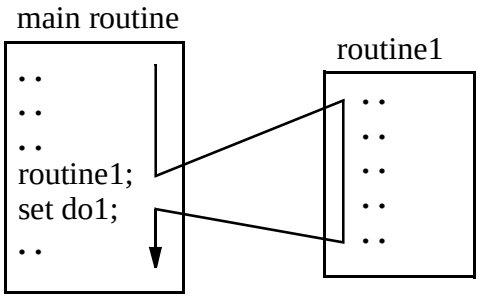


Figure 11 A routine can call another routine.

• Call up the instruction pick list for the program flow by choosing **IPL1: Prog. Flow**.

• Choose the instruction *ProcCall* by pressing the appropriate numeric key.

You must now specify the routine that is to be called. All routines are displayed for this purpose (see Figure 12).

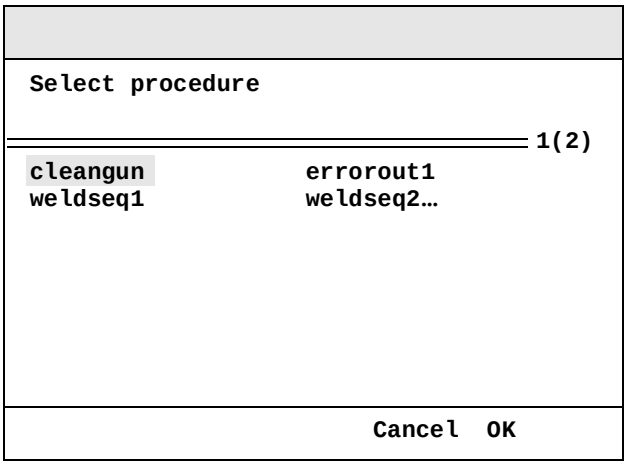


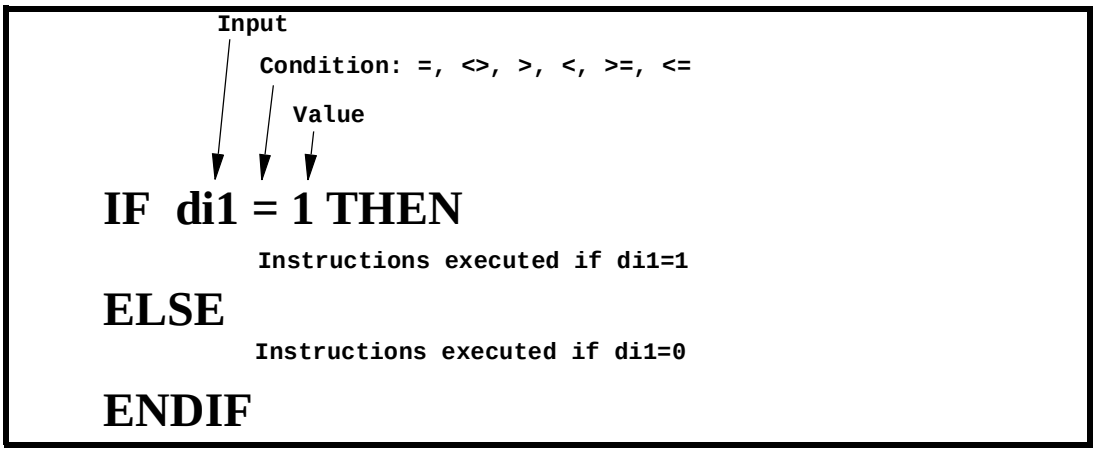
Figure 12 The dialog box used to select procedures.

• Select the desired routine and press ***OK***.

If the routine has no parameters, the instruction is ready for use; if it has parameters (indicated by ...), a dialog box will appear in which you specify the parameters of the routine in the same way as you specify an instruction argument.

4.2 Program control within a routine

The *IF* instruction is used when different instructions are to be executed depending on whether a condition is satisfied or not, e.g. depending on whether an input is set or not.



An *IF* instruction without ELSE is used when certain instructions are to be executed only if a condition is satisfied. If only one instruction is to be executed, a *Compact IF* instruction can be used.

Compact IF:

Any condition — Any instruction

IF reg1 = reg2 set do1

IF:

Any condition

IF di1 = 1 OR di2 = 1 THEN

Instructions executed if di1 or di2=1

ENDIF

To program an IF instruction in order to test an input

- Call up the correct instruction pick list by choosing **IPL1: Prog. Flow**.
- Choose the instruction *IF* (or *Compact IF*) by pressing the appropriate numeric key.

You now specify the condition that must be satisfied if program execution is to continue.

- When the condition <EXP> is selected press Enter [↵].

A dialog box will appear in which you specify the required data type for the condition (see Figure 13).

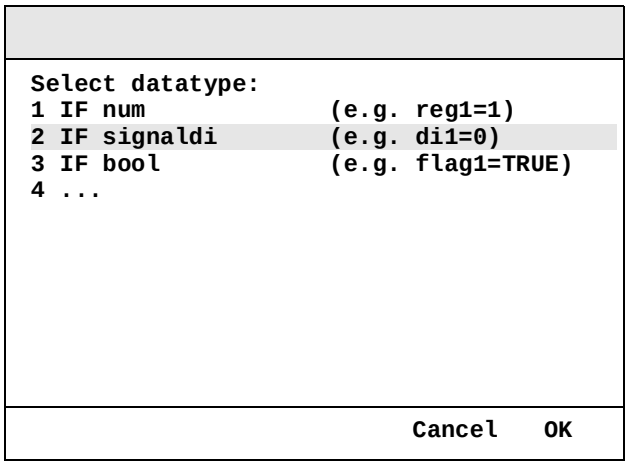


Figure 13 The dialog box used to select data type.

• Select ***IF signaldi*** and press Enter [↵] . Alternatively, you can use the numeric keyboard to select the figure in front of the desired data type.

A dialog box will appear in which you can specify the desired input (see Figure 14).

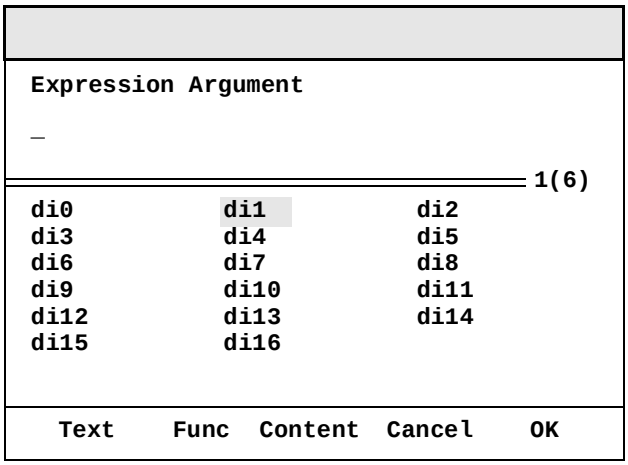


Figure 14 The dialog box used to define expression arguments.

• Select the desired input and press Enter [↵] .

• Choose ***OK*** to confirm.

The dialog box used to program expressions will be called up again. All operators are now displayed in the lower part of the box.

• Select the operator = and press Enter [↵] .

• Enter 0 or 1 directly using the numeric keyboard.

• Choose ***OK*** to confirm the change.

• Add instructions between THEN and ELSE and between the ELSE and ENDIF by selecting the empty instruction <SMT> and choosing the desired instructions from the pick list.

If you want to remove the ELSE part of the instruction:

• Select the complete IF instruction and press Enter [↵] .

A dialog box will appear, displaying the possible structure of the instruction. Structure parts not included in the instruction are enclosed within square brackets (see Figure 15).

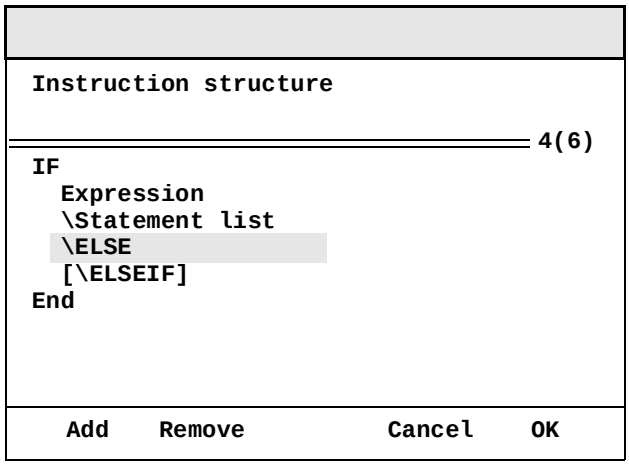


Figure 15 The dialog box to change the structure of an IF instruction.

• Select \ELSE and press ***Remove***.

• Choose ***OK*** to confirm the change.

5 Assigning a Value to Data (Registers)

An assignment instruction contains the following information:

- information on the data to be changed
- information on the desired value, which may be a complete expression, e.g. *reg1+5*reg2*.

The following instructions can be used to perform simple calculations on register variables.

Clear	reg1	clears a register
Incr	reg1	increments by 1
Decr	reg1	decrements by 1
Add	reg1, 5	adds a value (5) to a register

To program an assignment instruction

• Call up the correct instruction pick list by choosing **IPL1: Various** or **Mathematics**.

• Choose the instruction **:=** by pressing the appropriate numeric key.

You must now specify the data to be changed. All the various data are displayed for this purpose (see Figure 16).

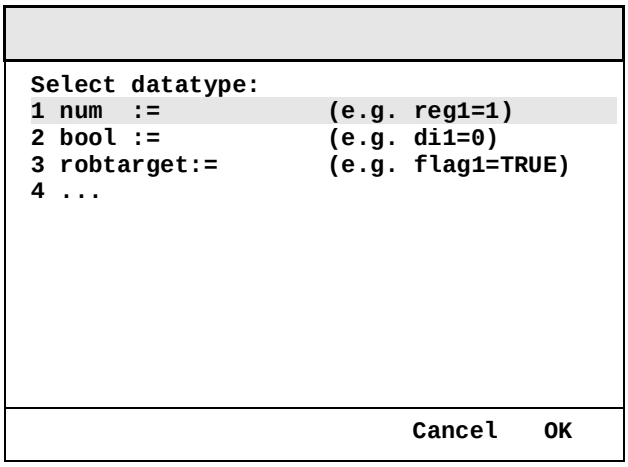


Figure 16 The dialog box used to select data type.

• Select the desired data type and press Enter ↵. Alternatively, you can use the numeric keyboard to select the figure in front of the desired data type.

If the desired data type is not found among the three predefined types, choose alternative 4 for more types. The data types that have already been used in the program will now be listed in the lower half of the box (see Figure 17).

To view all the data types, press the function key ***All Type***.

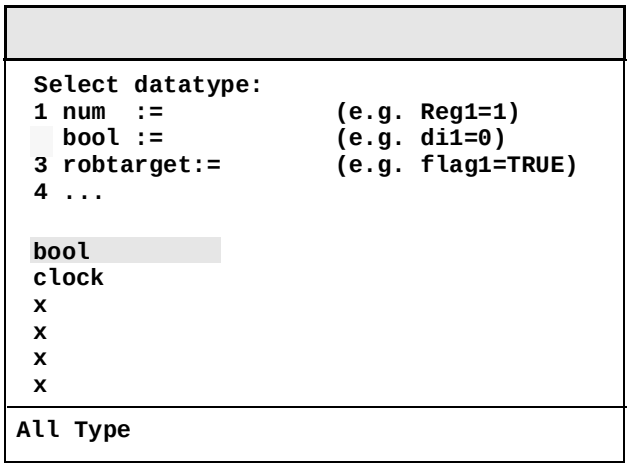


Figure 17 The dialog box shows data types used in the program.

• Choose the desired data type and press Enter ↵

A dialog box will appear in which you can define data that is to be changed (see Figure 18).

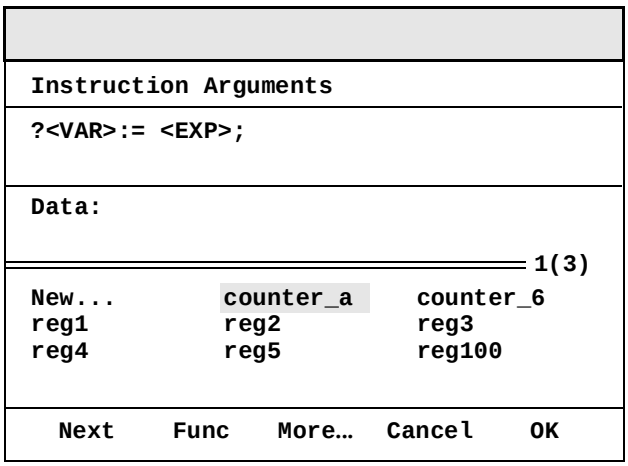


Figure 18 The dialog box used to define data that is to be changed. Only num data is shown in the list.

• Select the desired data.

• Select the next argument by pressing ***Next***.

You must now specify the new value for the data. For the purposes of this exercise, we have chosen a constant value, e.g. reg1:=5.
Use (list) to select a data instead of a numeric value.

• Using the numeric keyboard, enter the value directly.

• Choose ***OK*** to confirm the input of the instruction.

The instruction is now ready for use.

CONTENTS

Page

Calibration

1 Coordinate systems

All robot positions in a robot program, are stored in rectangular coordinates (e.g. xyz values for position), related to a defined coordinate system (or frame). This coordinate system may in turn be related to another coordinate system etc. in a chain. Some of these coordinate systems are embedded in the configuration of the robot system, and are not visible to the user, while others may be programmed by the user. The table below provides an overview of the various coordinate systems (or frames) used in the robot system:

Coordinate system	Defined where	Related to
Base Frame of robot	Service/View:BaseFrame. Base frame definition of robot gives relation between world and base frame.	World Frame
World Frame	No definition needed	Nothing
User Frame, fixed in room. (Tool mounted on robot)	Program/View: Data Types - wobjdata In any work object data	World Frame
User Frame, fixed on robot mounting plate. (Tool fixed in room)	Program/View: Data Types - wobjdata In any work object data	Wrist Frame
User Frame, coordinated to an external axis	Service/View:/BaseFrame. In the base frame definition of an external mechanical unit	World Frame
Object Frame	Program/View: Data Types - wobjdata In any work object data	User Frame
Program Displacement Frame	In the system variable C_PROGDISP, set up by instructions PDispSet or PDispOn etc.	Object Frame
Robtarget frame (Programmed position)	When a position is programmed.	Program Displacement Frame
Base Frame of a mechanical unit (only for internal system use)	Service/View:BaseFrame. In the base frame definition of an external mechanical unit or as configuration parameter.	World Frame
Wrist Frame	Implicit in the kinematic model of robot	Base Frame of the robot.
Tool Frame (Tool mounted on robot)	Program/View: Data Types - tooldata In any tool data	Wrist Frame

Now any programmed position, e.g. p1, will be related to the World Coordinate system through the chain:

world frame ’ user frame ’ object frame ’ program displacement frame ’ p1

The current position of the robot, i.e. the location of the tool, is related to the World Coordinate system through the chain:

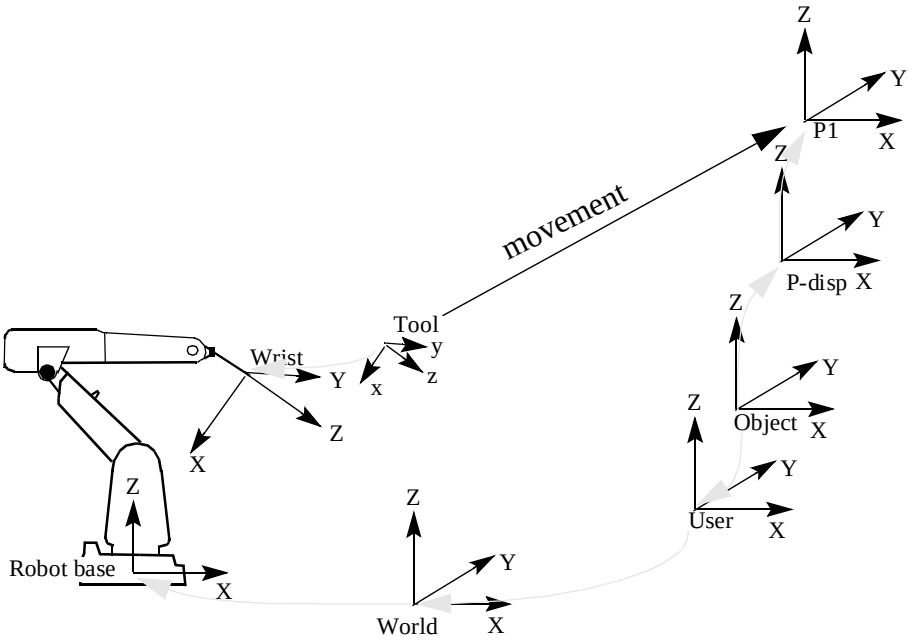
world frame ’ base frame ’ kinematic model ’ wrist centre frame ’ tool frame

When the robot is moved in automatic mode to a programmed position, the aim is to bring the tool (tool frame) to coincide with the programmed position, i.e. to close the chain:

user frame ’ object frame ’ program displacement frame ’ p1

world frame

base frame ’ kinematic model ’ wrist centre frame ’ tool frame



The accuracy of the robot, i.e. how well the tool frame will coincide with the programmed position, is normally independent of the accuracy of the various coordinate systems. This is true, however, only if the same coordinate systems are used as when programming the robot, pointing out all positions with the robot (repetition accuracy). If the coordinate systems are changed, making it possible to displace the program, then the accuracy is dependent on every single link in the chain. This means that the accuracy is directly dependent on the calibration accuracy of the various frames. This is even more important for off-line programming.

In the following chapters, an overview will be given of the steps to be taken to calibrate and define the robot and the different coordinate systems mentioned above.

2 Coordinated axes

2.1 External axes, general

All external axes are handled in mechanical units. This means that before an external axis may be moved, the mechanical unit to which it belongs, must be activated. Within a mechanical unit, the different axes will be given a logical name, from *a* to *f*. In the system parameters, these logical axes will be connected to the external axes joints. For each joint a motor and a drive unit is defined. Different joints may share the same motor and drive unit.

Two or more mechanical units may be activated at the same time, as long as they do not have the same logical axes defined in their set of external axes. However, two or more mechanical units may have the same logical axes, if they are not activated simultaneously.

Two or more mechanical units may not be activated at the same time, if they share one or more drive units, even if they use separate logical axes. This means that two logical axes, each belonging to different mechanical units, may control the same drive unit, but not at the same time.

2.2 Coordination

A mechanical unit may be coordinated or not coordinated with the robot movements.

If it is not coordinated, each axis will be moved independent of the robot movements, e.g. when jogging, only the separate axis will move. However during program execution, the external axes will be synchronized to the robot movement, in such a way that both movements will be completed in the same time.

If the mechanical unit is coordinated, it is guaranteed that the robot TCP movements, as seen in the object or user coordinate system, will be the same irrespective of the movements of the external axes.

Two types of coordination categories exist. The first category of coordination is when the robot itself is moved, e.g. the coordination to a gantry or track movement. This means that the robot is mounted on a gantry or a track, and may be moved along these axes. The world and user/object coordinate systems, however, will be fixed in the room, and the robot movements in these coordinate systems will be independent of simultaneous gantry or track movements. This coordination is automatically active, if the mechanical unit with the track motion is active.

The second coordination category, is when the robot movements are coordinated to the movements of a user frame connected to a mechanical unit. E.g. a user frame may be placed on a turntable and connected to its movements. An ordinary work object may be used for this purpose, if it is marked with the name of the mechanical unit to be connected to, and that it should be moveable. The coordination will be active if the mechanical unit is active, and the “coordinated” work object is active. When such a “coordinated” work object is used, in jogging or in a move instruction, the data in the “uframe” component will be ignored and the location of the user coordinate system will

only depend on the movements of the mechanical unit. However the “oframe” component will still work giving an object frame related to the user frame and also the displacement frame may be used.

3 Calibration

3.1 What is calibration?

Calibration involves setting the calibration positions (zero positions) of the axes and is used as the basis for their positioning. If the robot or external axes are not correctly calibrated, this will result in incorrect positioning and will have a negative effect on the agility of the robot. The robot is calibrated on delivery.

The position of the robot axes is determined using a resolver and a counter that counts the number of resolver revolutions. If the robot is correctly calibrated, it is automatically able to calculate the current position on start-up.

Calibration is carried out in two stages:

- Calibration of resolvers (fine calibration): the axes are placed in their specific calibration positions and the current resolver values are stored. For information on how to do this, see the chapter on *Repairs* in the Product Manual.
- Update of revolution counters: the correct motor revolution for the calibration is defined; the axes are placed close to their calibration positions and the revolution counters are updated.

The position of an external axis is determined using sync. switches. The same method used for the robot can be used.

3.2 Viewing the calibration status

• Press the Miscellaneous key  and select the Service window.

• Choose **View: Calibration.**

This window displays an overview of the status of all the mechanical units in the robot system (see Figure 1).

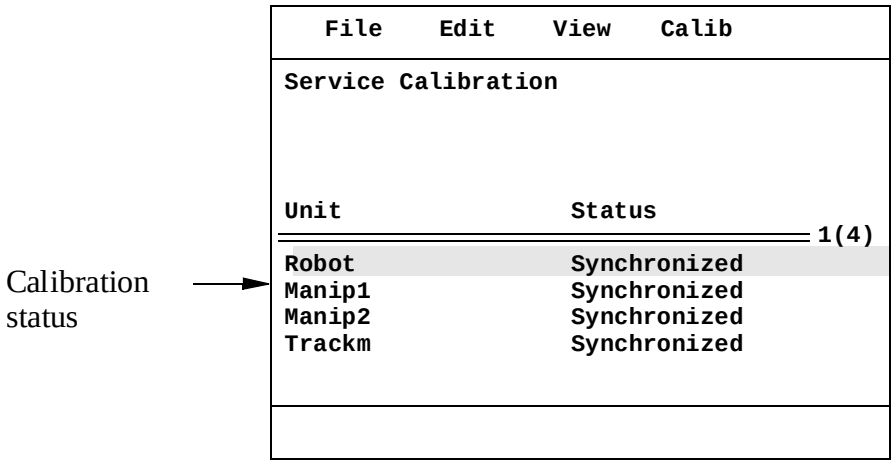


Figure 1 The Service Calibrate window shows whether or not the robot system is calibrated.

The calibration status can be any of the following:

- **Synchronized**
  All axes are calibrated and their positions are known. The unit is ready for use.
- **Not updated Rev. Counter**
  All axes are fine-calibrated but one (or more) of the axes has a revolution counter that is NOT updated. This or these must thus be updated.
- **Not calibrated**
  One (or more) of the axes is NOT fine-calibrated. This or these must thus be fine-calibrated.
- **Unsynchronized**
  At least one of the axes has a position that is NOT known. An external axis with a sync. switch must thus be synchronized. See Section 5, *Starting up*, in this manual.

3.3 Checking the calibration

If a revolution counter is incorrectly updated, it will cause incorrect positioning. Thus, check the calibration very carefully after each update.
An incorrect update can damage the robot system or injure someone.

- Run the calibration program under the /SERVICE/CALIBRAT/ directory on the system diskette, *Set up*. An alternative method is to jog the robot axis-by-axis until the axis angles in the Jogging window equal zero.
- Check each axis to see if the marks are positioned exactly opposite one another. If they are not, the calibration must be redone.

The marks may be scribed lines, vernier scales or the like. Their location is described in the chapter on *Installation and Commissioning* in the Product Manual.

3.4 Updating revolution counters

- Open the Service window.
- Choose **View: Calibration**.
- Select the desired unit.
- Move the robot or the chosen unit close to (half a motor revolution at the furthest) the calibration pose. The latter is usually indicated by a scribed line or a vernier scale. The calibration pose of the robot is described in the chapter on *Installation and Commissioning* in the Product Manual.
- Choose **Calib: Rev.Counter Update.**

A dialog box will appear, in which you can choose the axis you want to update (see Figure 2).

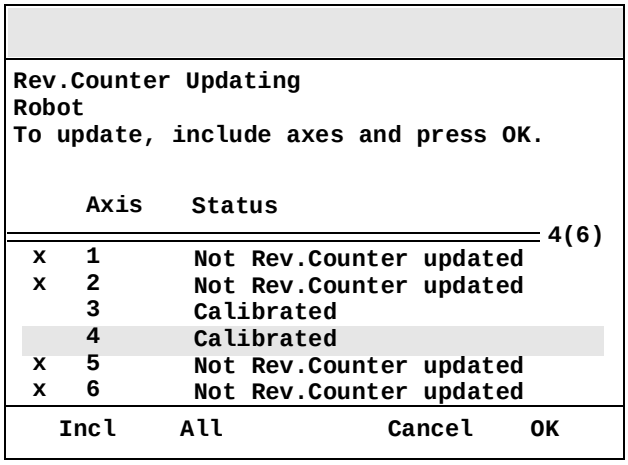


Figure 2 The dialog box used to select axes when updating the revolution counter.

- Select the axis to be updated and press the ***Incl*** function key.
  An **x** to the left indicates that the axis is to be updated.
- Use the same procedure on the remaining axes or press the function key ***All*** which selects all axes. A selected axis can be deselected by pressing the ***Excl*** function key.
- Confirm the choice of axes by pressing ***OK***.
- Start updating by pressing ***OK*** in the confirmation dialog box.

4 Base Frame for the Robot

4.1 Defining the Base Frame for the Robot

The following methods are used to define the location of the robot's base frame in relation to the world coordinate system.

In order to define a robot base frame you need a world fixed tip within the robot's working range, and optionally an elongator attached to the tip. If the robot is mounted on a track or similar, the track should be in its calibration position. The calibration procedure consists of a number of positionings for the robot's TCP to a reference point. The reference point´s coordinates in the world coordinate system, must be known. The coordinates must be stated before the calibration can be done.

The following positions on the world fixed tip device are involved in the calibration:

- the tip itself (with known coordinates in world), used when defining the base frame translation
- one point on the elongator defining the positive z direction for the world coordinate system
- one point on the elongator defining the positive x direction for the world coordinate system.

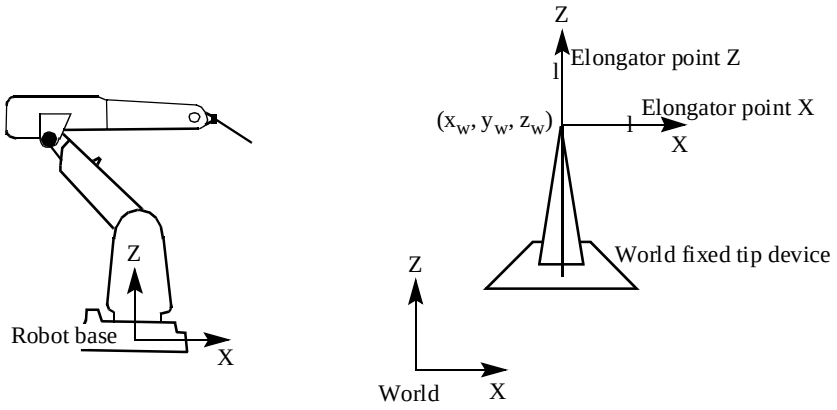


Figure 3 Robot base frame definition points.

When the necessary conditions are fulfilled the definition of the robot base frame can be performed. Please observe, that in the case of a track mounted robot, the track must be in the calibration position before the base frame of the robot may be defined.

• Press the Miscellaneous key and select the Service window.

• Choose **View:BaseFrame**.

A dialog containing all synchronized mechanical units is shown.

• Select the robot and press Enter or ***Def***.

A dialog like the one in Figure 4 will appear.

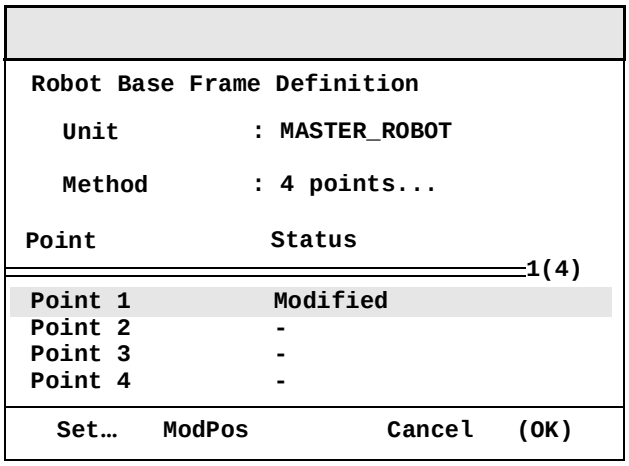


Figure 4 Robot base frame definition dialog.

To choose a definition method

Before you start modifying any positions, make sure the desired method is displayed.

- Select the field ***Method*** and press Enter ↵.
- Choose method for definition and press ***OK***.

There are two methods available in the list. The ***4 points*** method is used when the orientation shall be the same as for the world coordinate system. The ***4 points XZ*** method requires an elongator attached to the world fixed tip.

Input of world coordinates of the reference point

- Press ***Set***.
- Input the x, y and z values.
- Verify that the input is correct and press ***OK***.

To record world fixed reference points

- Select the first point ***Point 1***.
- Jog the robot as close as possible to the world fixed tip.
- Modify the position by pressing the function key ***ModPos***.
- Repeat the above for the points ***Point 2*** to ***Point n***.

To record the elongator X point

- Select the elongator point ***Point X***.

• Jog the robot as close as possible to the elongator point on the positive X axis.
• Modify the position by pressing the function key ***ModPos***.

To record the elongator Z point

• Select the elongator point ***Point Z***.
• Jog the robot as close as possible to the elongator point on the positive Z axis.
• Modify the position by pressing the function key ***ModPos***.

To calculate the robot base frame

• Press ***OK*** to calculate the robot base frame for the selected mechanical unit.

When the calculation is finished, a dialog like the one in Figure 5 will appear.

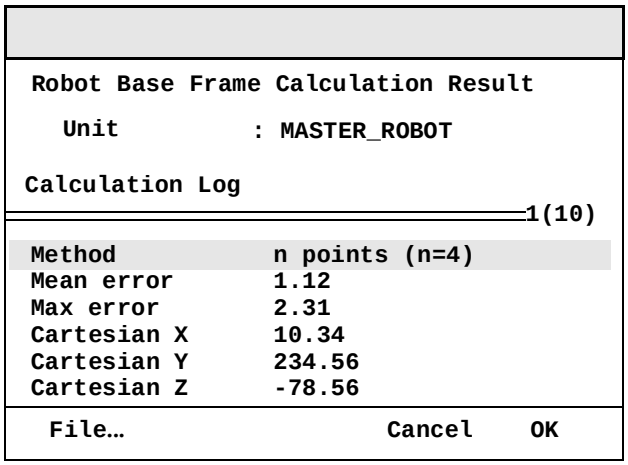


Figure 5 The result of a robot base frame calculation.

Field	Description
Unit	The name of the mechanical unit for which the definition of robot base frame is to be done.

List contents	Description
Method	Displays the selected calibration method.
Mean error	The accuracy of the robot positioning against the tip.
Max error	The maximum error for one positioning.
Cartesian X	The x coordinate for the base frame.
Cartesian Y	The y coordinate for the base frame.
Cartesian Z	The z coordinate for the base frame.
Quaternion 1-4	Orientation components for the base frame.

The result of the calculation is expressed in the world coordinate system.

The calculation result can be saved in a separate file for later use in a PC:

- Press the function key ***File***.
- Specify a name and a location where to save the result.
- Choose ***OK*** to confirm the save.

If the estimated error is

- acceptable, press ***OK*** to confirm the new robot base frame.
- not acceptable, redefine by pressing ***Cancel***.

• Choose **File: Restart** in the Service window to activate the base frame.

The definition is now complete, but before proceeding with other tasks, verify it by jogging the robot in the world coordinate system.

5 Coordinated track motion

5.1 How to get started with a coordinated track motion

In the checklist below, the steps required to coordinate track motion are described. In each step, there may be a reference to another chapter in this manual, where more details of the specific actions to be taken will be found.

• Define the system parameters for the track motion, see chapter 12 in this manual System Parameters/*Defining a track motion with coordinated motion*. Find out the name of this mechanical unit, and the corresponding logical axis.

• Calibrate the robot and the track motion, i.e. the zero position of the measuring system for both robot and track must be carefully determined. See *Calibration* on page 6.

• Define the base frame of the robot, see *Defining the Base Frame for the Robot* on page 9. Please observe that the track must be in its calibration position when the robot base frame is defined.

• Define the base frame of the track, see *Defining the Base Frame for a track motion* on page 13.

• Store all these definitions on a diskette by giving the command **File: Save All as** in the System parameter window. See chapter 12 in this manual.

• Activate the track unit in the jogging window and check that the coordination is working satisfactorily. This may be done by choosing ***World*** or ***Wobj*** in the field ***Coord*** and then jogging the track axis. The robot TCP should not move, but be fixed relative to the object coordinate system.

5.2 Defining the Base Frame for a track motion

To make coordinated track motion possible it is necessary to define the base frame of the track. This frame is located in the calibration position of the track, see Figure 6.

For the definition of a track base frame you need a world fixed tip within the robot's working range. The calibration procedure consists of a number of positionings of the TCP to the reference point. Please note that before the base frame of the track may be defined, the base frame of the robot must be defined with the track in the calibration position.

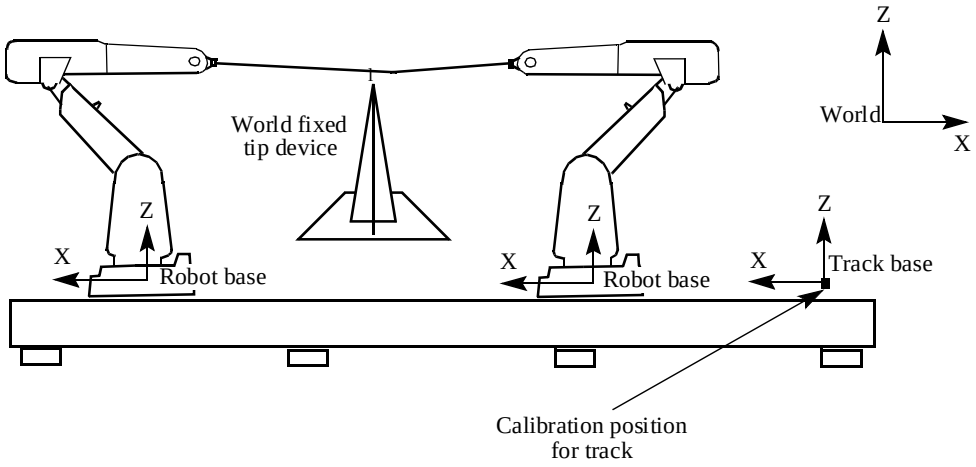


Figure 6 Track base frame definition procedure.

The track's base coordinate system has its origin in the robot's base when the track is in its calibration position. The x direction is pointing along the linear track path and the z axis of the track's coordinate system is parallel with the z axis of the robot's base coordinate system.

Figure 7 shows an example of how the base systems are oriented for a specific robot mounting. In this case the robot is mounted on the track at an angle of 45 degrees.

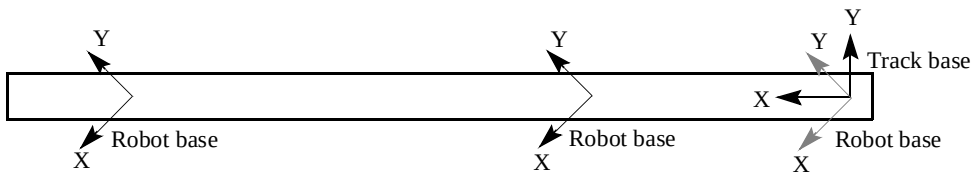


Figure 7 Track and robot base coordinate systems seen from above.

• Press the Miscellaneous key  and select the Service window.

• Choose **View:BaseFrame**.

A dialog containing all synchronized mechanical units is shown.

• Select the track unit and press Enter ↵ or ***Def***.

A dialog like the one in Figure 8 will appear.

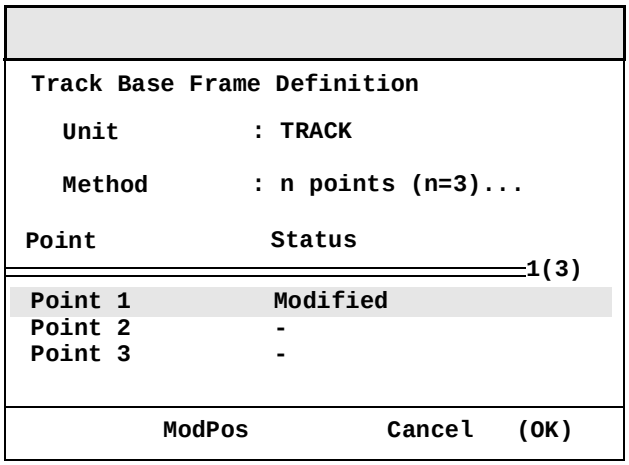


Figure 8 Track base frame definition dialog.

To choose definition methods

Before you start modifying any positions, make sure the desired method is displayed. The method defines the number of track positions from where the robot TCP will be moved to the reference point.

• Select the field ***Method*** and press Enter ↵.

• Choose the number of points to be used for definition and press ***OK***. (Currently only the three point method is implemented.)

To record world fixed reference points

Activate the track unit and run it to the calibration position, i.e. zero position should be displayed on the teach pendant.

• Select the first point ***Point 1***.

• Jog the robot as close as possible to the world fixed tip.

• Modify the position by pressing the function key ***ModPos***.

• Move the robot along the track and repeat the steps above for the points ***Point 2*** and ***Point 3***.

To calculate the track base frame

• Press ***OK*** to calculate the track base frame for the selected mechanical unit.

When the calculation is finished, a dialog like the one in Figure 9 will appear.

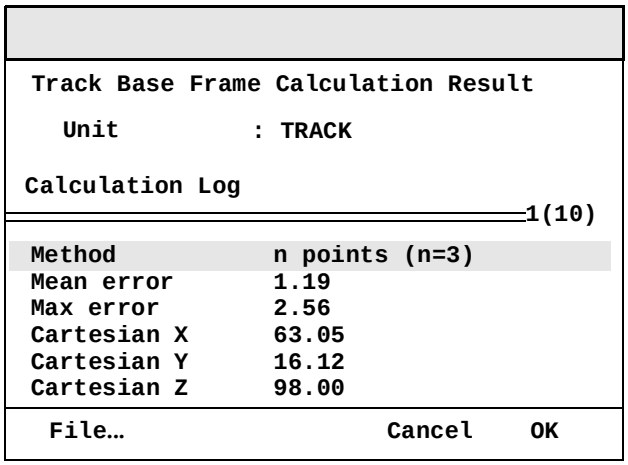


Figure 9 The result of a track base frame calculation.

The result of the calculation is expressed in the world coordinate system.

Field	Description
Unit	The name of the mechanical unit for which the definition of base frame is to be done.

List contents	Description
Method	Displays the selected track definition method.
Mean error	The accuracy of the robot positioning against the tip.
Max error	The maximum error for one positioning.
Cartesian X	The x coordinate for the base frame. (x, y, z is the same as for the robot base frame).
Cartesian Y	The y coordinate for the base frame.
Cartesian Z	The z coordinate for the base frame.
Quaternion 1-4	Orientation components for the base frame.

The calculation result can be saved in a separate file for later use in a PC:

- Press the function key ***File***.
- Specify a name and a location where to save the result.
- Choose ***OK*** to confirm the save.

If the estimated error is

- acceptable, press ***OK*** to confirm the new track base frame.
- not acceptable, redefine by pressing ***Cancel***.

• Choose **File: Restart** in the Service window to activate the track base frame.

The definition is now complete but before proceeding with other tasks, verify it by doing the following:

• Point out with the robot, in coordinated mode, the world fixed reference point with the track in different positions, and print out the position in world coordinates. Jog the track in coordinated mode.

6 Coordinated external axes

6.1 How to get started with a coordinated (moveable) user coordinate system

In the checklist below, the steps required to coordinate a user coordinate system are described. In each step, there may be a reference to another chapter in this manual, where more details of the specific actions to be taken will be found.

- Define the system parameters for the external mechanical unit, see chapter 12 in this manual System Parameters/*Defining an external mechanical robot coordinated with the robot*. Find out the name of this mechanical unit, and the corresponding logical axis.
- Calibrate the robot and the mechanical unit, i.e. the zero position of the measuring system for both robot and mechanical unit must be carefully determined. See *Calibration* on page 6.
- Define the base frame of the robot, see *Defining the Base Frame for the Robot* on page 9.
- Define the user frame of the mechanical unit, see *Defining the User Frame for a rotational axis (single)* on page 17 or *Defining the User Frame for a two-axes mechanical unit, Method 1* on page 20 or *Defining the User Frame for a two-axes mechanical unit, Method 2* on page 23.
- Store all these definitions on a diskette, by giving the command **File: Save All as** in the System parameter window. See chapter 12 in this manual.
- Create a new work object data and give it a name, e.g. *turntable*. In this work object, change the component *ufprog* to FALSE, indicating that the user object should be connected to a moveable mechanical unit. Also change the component *ufmec* to the name of the mechanical unit *turntable* (must be written in text mode).
- If you want the object frame to be displaced relative to the user frame, you may write the displacement in the x, y, z values of the “oframe” component of the work object. For other methods see *Defining a moveable object frame* on page 40.
- Activate the mechanical unit in the jogging window and check that the coordination is working satisfactorily. This may be done by choosing ***Wobj*** in the field ***Coord***, and the work object, e.g. *turntable*, in the field ***Wobj***, and then jogging one of the mechanical unit axes. The robot TCP should also move, following the moveable object coordinate system.
- When programming, it is important to have the coordinated work object, in this case *turntable*, programmed as an argument in each move instruction. This will be automatically added to the move instruction, if the work object is activated in the jogging window before starting the programming.

6.2 Defining the User Frame for a rotational axis (single)

This method will define the location of the user coordinate system of a rotational single axis type mechanical unit, relative to the world coordinate system. This user coordinate system should be used when a coordinated work object is used.

The definition of a user frame for a rotational external axis requires that the turntable on the external axis has a marked reference point. The calibration procedure consists of a number of positionings for the robot's TCP on the reference point when the turntable is rotated to different angles. See Figure 10.

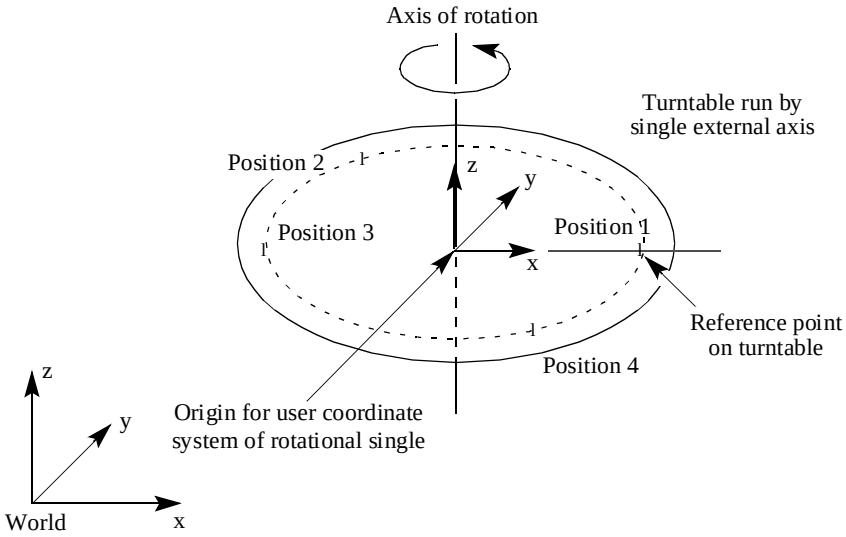


Figure 10 Definition points for a rotational axis.

The user coordinate system for the rotational axis has its origin in the centre of the turntable. The z direction coincides with the axis of rotation and the x axis goes through the reference point. Figure 11 shows the user coordinate system for two different positionings of the turntable (turntable seen from above).

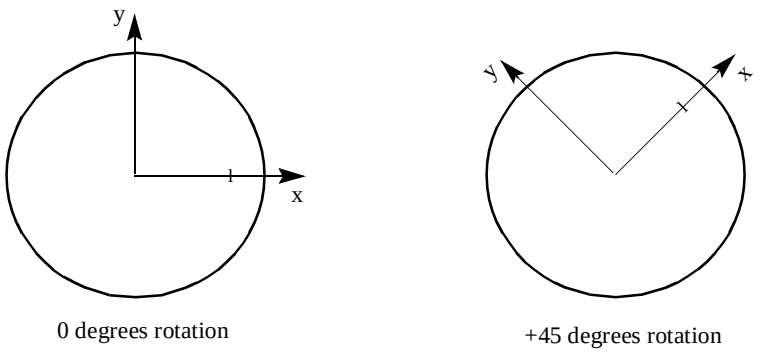


Figure 11 The user coordinate system at various angles of rotation.

• Press the Miscellaneous key  and select the Service window.

• Choose **View:BaseFrame**.

A dialog containing all synchronized mechanical units is shown.

• Select the mechanical unit and press Enter ↵ or ***Def***.

A dialog like the one in Figure 12 will appear.

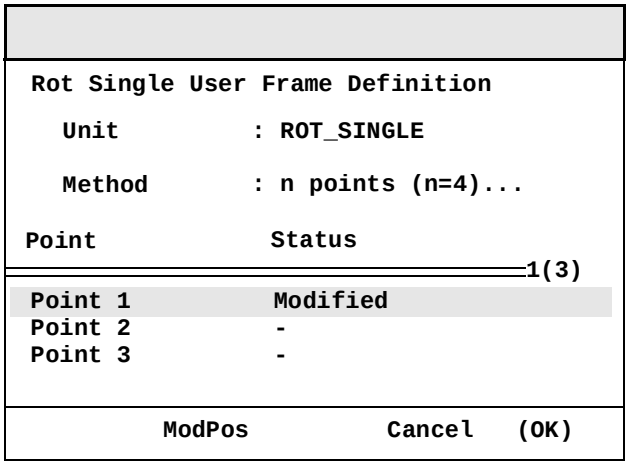


Figure 12 Dialog for definition of user frame for a rotational axis.

To choose a definition method

Before you start modifying any positions, make sure the desired method is displayed.

• Select the field ***Method*** and press Enter ↵.

• Choose number of points to use for definition and press ***OK***. (Currently only the four point method is implemented.)

To record turntable reference points

Activate the mechanical unit and run it to its calibration position, i.e. zero position should be displayed on the teach pendant.

• Select the first point ***Point 1***.

• Point out the reference point on the turntable with the robot's TCP.

• Modify the position by pressing the function key ***ModPos***.

• Rotate the turntable in the positive direction and repeat the above for the points ***Point 2*** and ***Point 3***.

To calculate the user frame

• Press ***OK*** to calculate the user frame for the selected mechanical unit.

When the calculation is finished a dialog like the one in Figure 13 will appear.

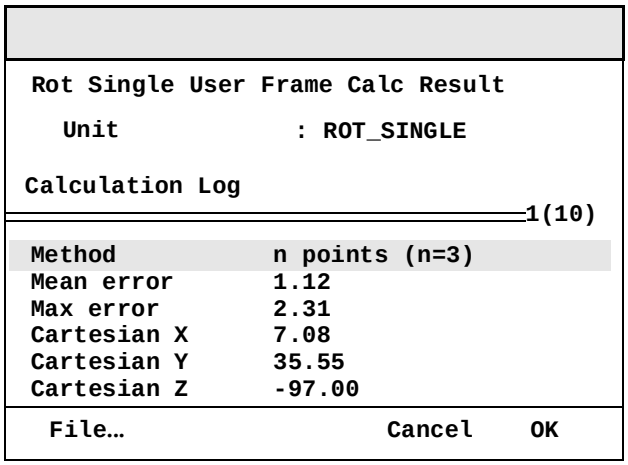


Figure 13 The result of a user frame calculation for a rotating single.

The calculation log shows the user frame expressed in the world coordinate system when the mechanical unit is in its calibration position.

Field	Description
Unit	The name of the mechanical unit for which the definition of user frame is to be done.
List contents	Description
Method	Displays the selected calibration method.
Mean error	The accuracy of the robot positioning against the reference point.
Max error	The maximum error for one positioning.
Cartesian X	The x coordinate for the user frame.
Cartesian Y	The y coordinate for the user frame.
Cartesian Z	The z coordinate for the user frame.
Quaternion 1-4	Orientation components for the user frame.

The calculation result can be saved in a separate file for later use in a PC:

- Press the function key ***File***.
- Specify a name and a location where to save the result.
- Choose ***OK*** to confirm the save.

If the estimated error is

- acceptable, press ***OK*** to confirm the new user frame.
- not acceptable, redefine by pressing ***Cancel***.

• Choose **File: Restart** in the Service window to activate the user frame.

The definition is now complete, but before proceeding with other tasks, verify it by jogging the mechanical unit in coordinated mode.

Note The user frame is stored in the system parameters as the base frame of the external mechanical unit. The user frame in the corresponding work object is therefore not used.

6.3 Defining the User Frame for a two-axes mechanical unit, Method 1

This method will define the location of the user coordinate system of an “Orbit” type mechanical unit, relative to the world coordinate system. This user coordinate system should be used when a coordinated work object is used.

It should be noted that this method requires that the kinematics (relationship between two axes) of the mechanical unit are defined in the robot system configuration. Therefore, this method can only be used for workpiece manipulators supplied by ABB, where a ready-made configuration was included in the delivery. For other types of workpiece manipulator see *Defining the User Frame for a two-axes mechanical unit, Method 2* on page 23.

The definition of this user coordinate system requires that the orbit turntable is marked with a coordinate system as shown in Figure 14. The coordinate system must have the x axis in the plane of the two turning axes of the Orbit station, when the turn table is in its calibration position.

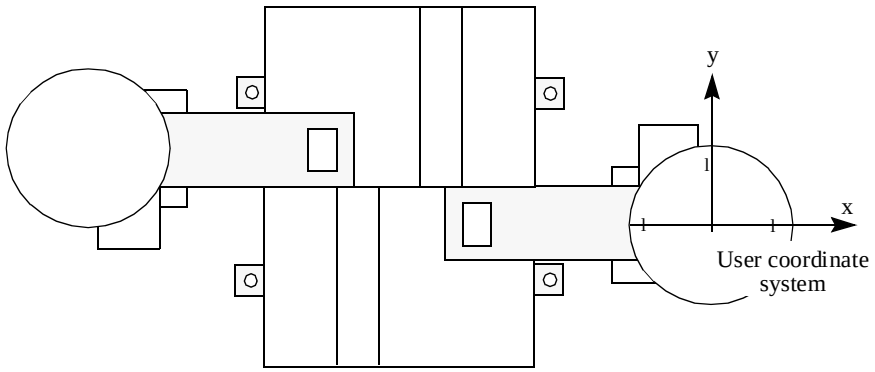


Figure 14 Orbit user coordinate system.

The coordinate system of the orbit station has its xy plane in the surface of the turntable, and the origin is located in the centre of the turntable, i.e. the z axis will coincide with the second axis.

• Press the Miscellaneous key  and select the Service window.

• Choose **View:BaseFrame**.

A dialog containing all synchronized mechanical units is shown.

• Select the mechanical unit and press Enter ↵ or ***Def***.

A dialog like the one in Figure 15 will appear.

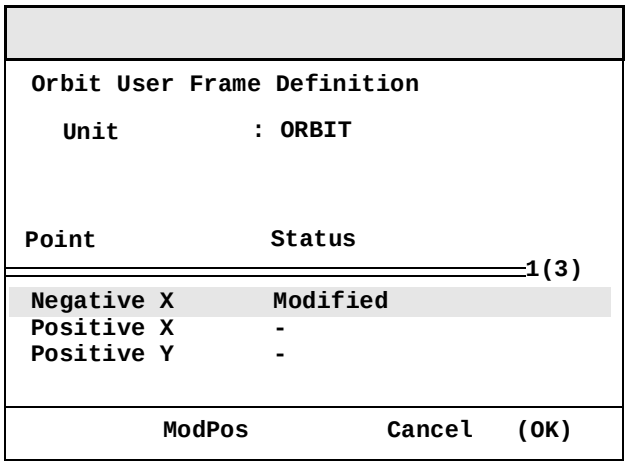


Figure 15 Dialog for definition of user frame for orbit station.

To record reference points

• Activate the mechanical unit and run it to its calibration position, i.e. zero position should be displayed on the teach pendant.

• Select the first point ***Negative X***.

• Point out the reference point on the negative x axis with the robot's TCP (it is not necessary that the position is on the negative side of the origin, but it must be on the negative side relative to the next point "Positive X").

• Modify the position by pressing the function key ***ModPos***.

• Select the point ***Positive X***.

• Point out the reference point on the positive x axis with the robot's TCP.

• Modify the position by pressing the function key ***ModPos***.

• Select the point ***Positive Y***.

• Point out the reference point on the positive y axis with the robot's TCP.

• Modify the position by pressing the function key ***ModPos***.

To calculate the user frame

• Press ***OK*** to calculate the user frame for the selected mechanical unit.

When the calculation is finished, a dialog like the one in Figure 16 will appear.

The calculation log shows the user frame expressed in the world coordinate system when the mechanical unit is in its calibration position.

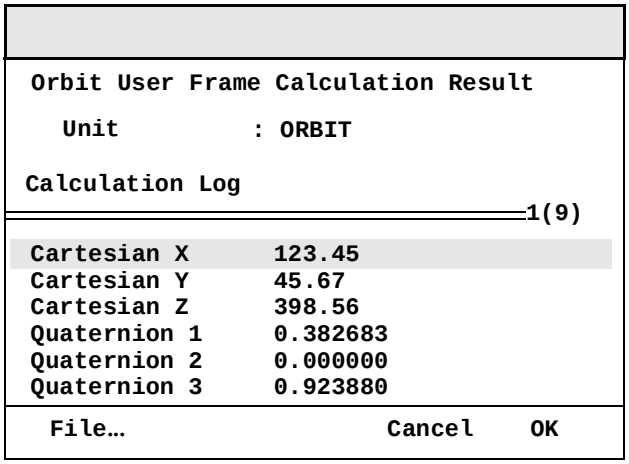


Figure 16 The result of a linear moving base frame calculation.

Field	Description
Unit	The name of the mechanical unit for which the definition of user frame is to be done.
List contents	Description
Cartesian X	The x coordinate for the user frame.
Cartesian Y	The y coordinate for the user frame.
Cartesian Z	The z coordinate for the user frame.
Quaternion 1-4	Orientation components for the user frame.

The calculation result can be saved in a separate file for later use in a PC:

- Press the function key ***File***.
- Specify a name and a location where to save the result.
- Choose ***OK*** to confirm the save.

If the estimated error is

- acceptable, press ***OK*** to confirm the new user frame.
- not acceptable, redefine by pressing ***Cancel***.

The definition is now complete, but before proceeding with other tasks, verify it by jogging the mechanical unit in coordinated mode.

Note The user frame is stored in the system parameters as the base frame of the external mechanical unit. The user frame in the corresponding work object is therefore not used.

6.4 Defining the User Frame for a two-axes mechanical unit, Method 2

This method will define the location of the user coordinate system of an “Orbit” type mechanical unit, relative to the world coordinate system. This user coordinate system should be used when a coordinated work object is used.

It should be noted that this method does not require that the kinematics (relationship between two axes) of the mechanical unit are defined in the robot system configuration. If this is a known factor, another method can be used. See *Defining the User Frame for a two-axes mechanical unit, Method 1* on page 20.

Figure 17 shows an orbit station with two rotational axes and a turntable mounted on the second axis.

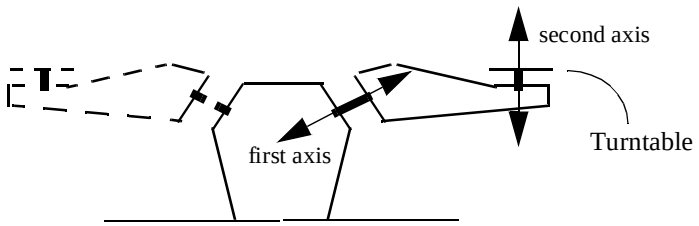


Figure 17 Geometric structure of an orbit station.

The definition of the user frame requires that the turntable has a marked reference point. The origin of the user frame is located in the centre of the turntable with the z axis coinciding with the second axis of rotation. The x axis goes through the reference point (see figure below).

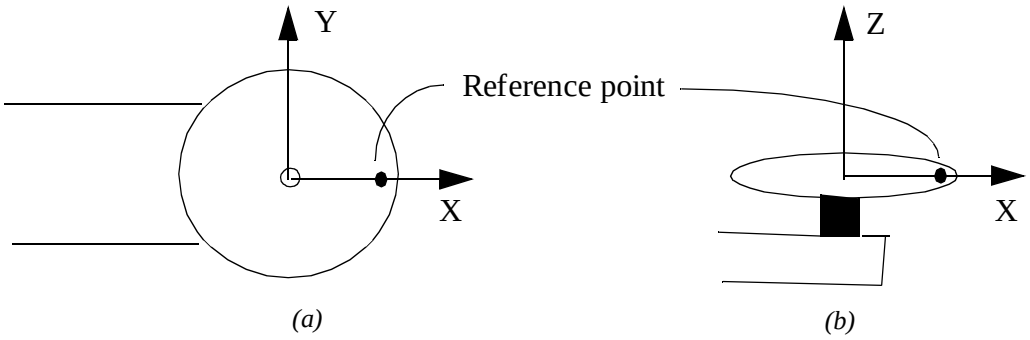


Figure 18 The turntable seen from above (a) and side (b).

The user frame is determined by two definition procedures. One procedure for the first axis and another similar procedure for the second axis. These two procedures are performed separately but both are necessary to complete the user frame definition.

• Press the Miscellaneous key and select the Service window.

• Choose **View: Two Axes Definition**

A dialog containing all synchronized mechanical units is shown.

• Select the mechanical unit and press Enter or ***Def***.

A dialog like the one in Figure 19 will appear.

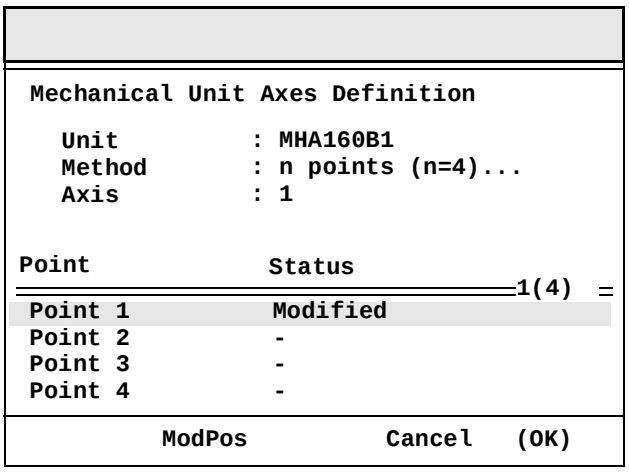


Figure 19 Dialog for definition of axes.

Defining the first axis

Before defining the first axis, both axes must be run to their calibration positions. The procedure to define the first axis consists of a number of positionings for the robot's TCP on the reference point when the first axis is rotated to different angles. Position 1 is the position of the reference point when both axes are fixed to their calibration positions. The following positions, position 2, 3, 4 etc., are the positions of the reference point when the first axis is rotated to greater angles in successive steps. See Figure 20.

.

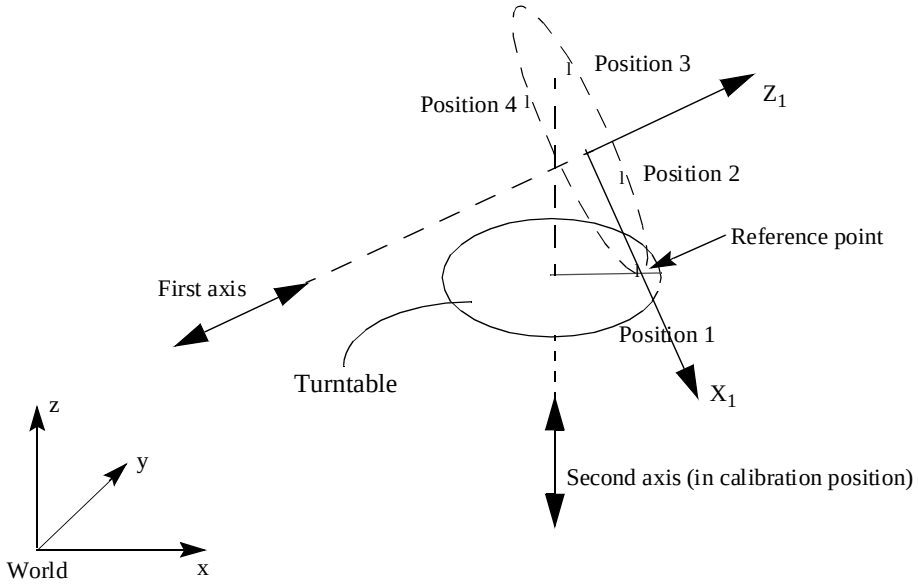


Figure 20 Definition of the first axis. Four positionings of the robot's TCP on the reference point are performed with the first axis rotated to different angles.

Defining the second axis

Before defining the second axis, both axes must be run to their calibration positions. The procedure to define the second axis consists of a number of positionings for the robot's TCP on the reference point when the second axis is rotated to different angles. Position 1 is the position of the reference point when both axes are fixed to their calibration positions. The following positions, position 2, 3, 4 etc., are the positions of the reference point when the second axis is rotated to greater angles in successive steps. See Figure 21.

.

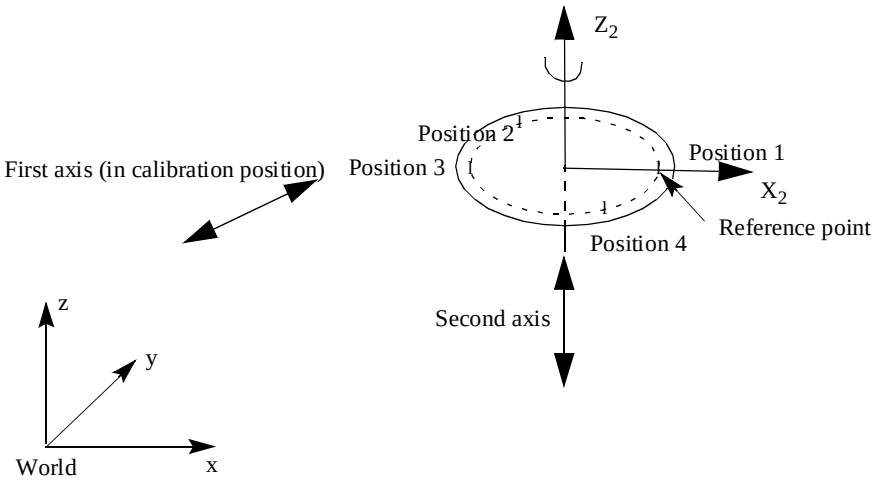


Figure 21 Definition of the second axis. Four positionings of the robot's TCP on the reference point are performed with the second axis rotated to different angles.

This frame coincides with the user frame when both axes are fixed to their calibration positions.

To choose a definition method

Before you start modifying any positions, make sure the desired method is displayed and the mechanical unit is activated.

- Select the field ***Method*** and press Enter ↵.
- Choose the number of points to use for the axis definition and press **OK**.

To choose axis

You can choose which one of the axes you want to define. Remember that both axes must be defined to complete the user frame definition. It is possible to redefine both axes or just one of them.

- Select the field ***Axis*** and press Enter ↵ to switch axis.

To record reference points for the first axis definition

Make sure Axis 1 is chosen. Run the mechanical unit to its calibration position.

- Select the first point, ***Point 1***.
- Point out the reference point on the turntable with the robot's TCP.
- Modify the position by pressing the function key ***ModPos***.
- Rotate the first axis to a greater angle and repeat the above for the points ***Point 2*** to ***Point n***.
- Press ***OK*** to calculate the frame of the first axis.

To record reference points for the second axis definition

Make sure Axis 2 is chosen. Run the mechanical unit to its calibration position.

- Select the first point ***Point 1***.
- Point out the reference point on the turntable with the robot's TCP.
- Modify the position by pressing the function key ***ModPos***.
- Rotate the second axis to a greater angle and repeat the above for the points ***Point 2*** to ***Point n***.
- Press ***OK*** to calculate the frame of the second axis.

To confirm/cancel the new axis definition

When ***OK*** is pressed after the points have been modified for an axis, a dialog like the one in Figure 22 will appear.

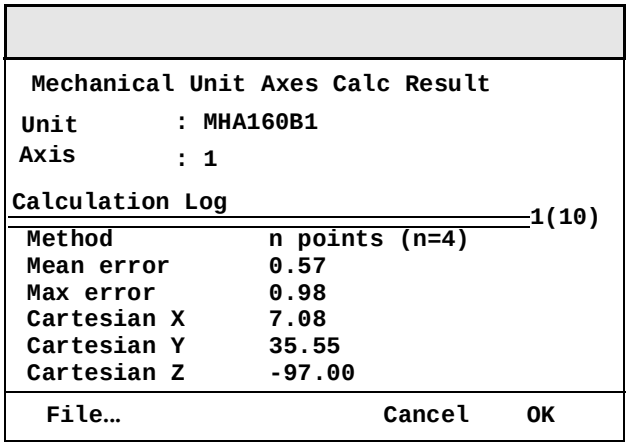


Figure 22 The result of the first axis definition.

The calculation log shows the calculated frame expressed in the world coordinate system.

Field	Description
Unit	The name of the mechanical unit for which the definition of the axis is to be done.
Axis	The chosen axis.

List contents	Description
Method	Displays the selected method.
Mean error	The accuracy of the robot positioning relative to the reference point.
Max error	The maximum error for one positioning.
Cartesian X	The x coordinate for the frame.
Cartesian Y	The y coordinate for the frame.
Cartesian Z	The z coordinate for the frame.
Quaternion 1-4	Orientation components for the frame.

The calculation result can be saved in a separate file for later use in a PC:

- Press the function key ***File***.
- Specify a name and a location where to save the result.
- Choose ***OK*** to confirm the save.

If the estimated error is

- acceptable, press ***OK*** to confirm the new axis definition. Now the next axis can be defined if necessary.
- not acceptable, redefine by pressing ***Cancel***.

The user frame definition is now completed, but before proceeding with other tasks, verify it by jogging the mechanical unit in coordinated mode.

Note The user frame is stored in the system parameters as the base frame of the external mechanical unit. The user frame in the corresponding work object is therefore not used.

7 Defining Tools

The position of the robot and its movements are always related to its tool coordinate system, i.e. the TCP and tool orientation (see Figure 23). To get the best performance, it is important to define the tool coordinate system as correctly as possible. For more information, see the RAPID Reference Manual/ Motion and I/O Principles.

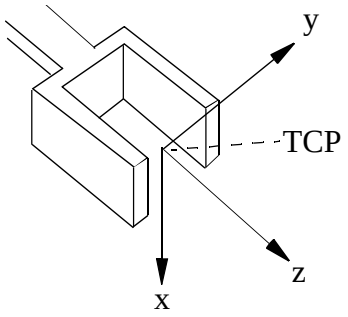


Figure 23 The tool coordinate system for a gripper.

A tool coordinate system can either be defined manually or the robot can be used as the measuring tool. Manual definitions can be used if accurate data for the dimensions of the tool is available or if minor corrections are to be done.

7.1 Creating a new tool

A tool should normally be placed in the system module, *User*. In that way, it will be common to all programs, which means that if a TCP is modified, all programs will automatically be affected. The tool can then also be used for jogging when there is no program in the program memory.

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *tooldata* and press Enter [↵].
- Create the new tool using one of the following alternatives:
  - **alt 1**. Press the function key ***New***.
    The tool's TCP and orientation will then be the same as the robot's mounting flange.
  - **alt 2**. Select an existing tool and press the function key ***Dupl***.
    The tool's TCP and orientation will then be the same as the one duplicated.

A window appears, displaying the name of the data.

- If you want to change the name, press Enter [↵] and specify a new name.
- Press the function key ***Decl***.

A dialog box appears, displaying the basic tooldata declaration.

- If you want to save the data in another module, select the field ***In Module*** and press Enter [↵]. Specify the name of the module in which the data is to be saved.
- Press ***OK*** to confirm.

Note: Do not change the type of the tool. This must always be of the persistent type.

7.2 Manually updating the TCP and weight of a tool

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *tooldata* and press Enter ↵ .
- Select the tool to be changed and press Enter ↵ .
- Select the TCP component (x, y, z) that you wish to change.
- Change the value using the numeric keyboard. To enter a decimal point (.) or minus sign (-), use the function keys.
- Select the *mass* component.
- Change the weight using the numeric keyboard.
- If the tool is stationary, i.e. not mounted on the robot, change the component *robhold* to FALSE. For more information about stationary tools see *Stationary tool* on page 33.
- Choose ***OK*** to confirm the change.

Note: Only the mass of the tool should be specified. A payload handled by a gripper is specified by the instruction GripLoad.

7.3 Methods of defining the tool coordinate system

To define the TCP of a tool, you need a world fixed tip within the robot's working space. You then jog to (at least) four robot positions with different orientations, as close as possible to the world fixed tip (see Figure 24). These positions are called *approach points*.

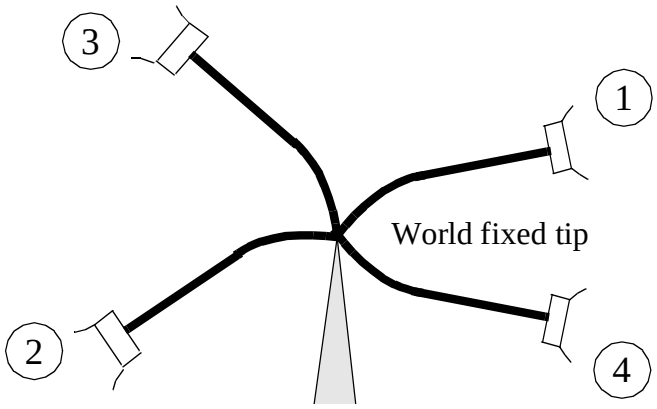


Figure 24 Approach points for a tool's TCP.

To define a complete orientation of a tool, you move any position on the desired z axis and any position on the desired x axis to the world fixed tip. These positions are called *elongator points* (see Figure 25). This can be done by fitting an elongator to the tool to define the z and x directions or by aligning the tool according to the world coordinate system and then jogging the robot in these directions.

Note The elongator points must be defined with the same orientation as the last approach point used.

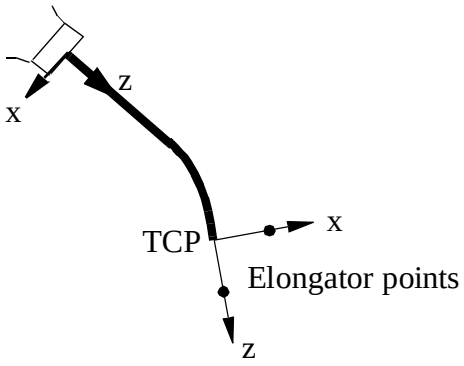


Figure 25 Elongator points for a tool's orientation.

If you only want to define the TCP, only the world fixed tip is needed. If you only need a definition of the orientation in the z direction, the elongator will only point to z.

The following methods are supported:

- ***4-point TCP***
  Four approach points are used to define the TCP. The orientation will be set according to the wrist coordinate system (see Figure 26).

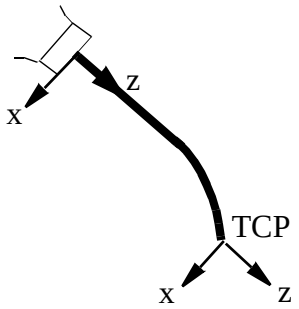


Figure 26 Using the 4-point method, only the TCP is defined. The tool direction will correspond to the wrist coordinate system.

- ***4-p TCP ORIENT NOT SET***
  The same as 4-point TCP but the orientation will not be changed.

- ***5-point TCP&Z***
  Four approach points are used to define the TCP and one elongator point is used to define the z direction of the tool. The x and y directions will be as close as possible to the corresponding axes in the wrist coordinate system (see Figure 27).

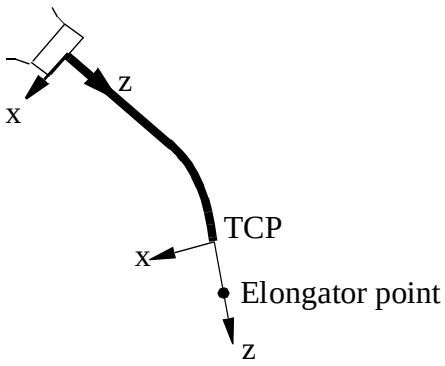


Figure 27 Using the 5-point method, the TCP and the tool's z direction are defined. The x and y directions are set automatically by the robot.

- ***6-point TCP&ZX***
  Four approach points are used to define the TCP, one elongator point is used to define the z direction and one elongator point is used to define the x direction of the tool (see Figure 28).

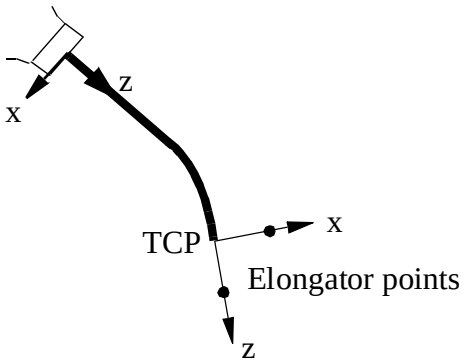


Figure 28 Using the 6-point method, the TCP and all the tool's directions are defined.

7.4 Using the robot to change the TCP and orientation of a tool

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *tooldata* and press Enter [↵].
- Select a tool (or create a new tool, see ***Creating a new tool*** on page 28).
- Choose **Special: Define Coord**.

A dialog box appears, displaying the points defined by whichever method was used (see Figure 29).

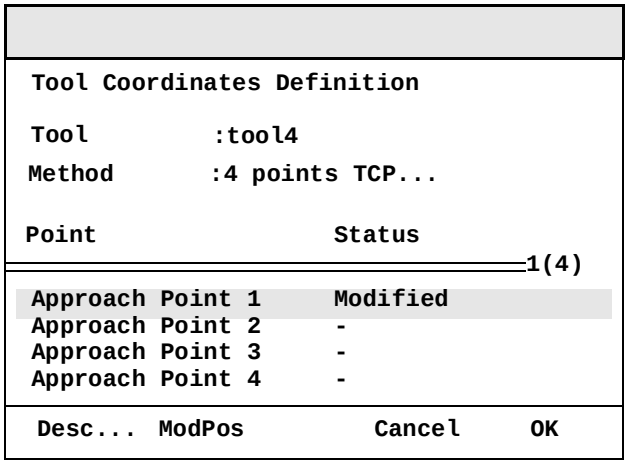


Figure 29 The robot can be used to define the tool coordinate system.

The status can be defined as follows:

Status	Meaning
-	No position defined
Modified	Position modified

To choose a definition method

Before you start modifying any positions, make sure the desired method is displayed. See ***Methods of defining the tool coordinate system*** on page 29.

- Select the field ***Method*** and press Enter [↵].
- Choose a method and press ***OK***.

To record Approach Points

- Select the first point ***Approach Point 1***.
- Jog the robot as close as possible to the world fixed tip.
- Modify the position by pressing the function key ***ModPos***.
- Repeat the above for the points ***Approach Point 2-4***.

To record Elongator Point Z (if the 4-point TCP method is not used)

- Select ***Elongator z Point***.
- Jog - without changing the orientation from the last approach point - any point on the desired positive z axis to the world fixed tip. An extension should be fitted to obtain better accuracy.
- Modify the position by pressing the function key ***ModPos***.

To record Elongator Point X (only if the 6-Point TCP&XZ method is used)

- Select ***Elongator x Point***.
- Jog - without changing the orientation from the last approach point - any point on the desired positive x axis to the world fixed tip.
- Modify the position by pressing the function key ***ModPos***.

To calculate the tool coordinate system

- Press ***OK*** to calculate the tool coordinate system.

When the calculation is finished, a dialog like the one in Figure 30 will appear.

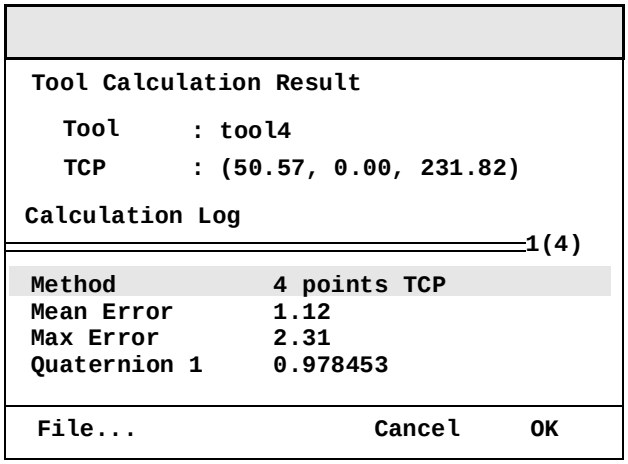


Figure 30 The result of a tool calculation.

Field	Description
TCP	The values of the calculated TCP.
Mean Error	The average distance that the approach points are from the calculated TCP, i.e. how accurately the robot was positioned relative to the tip.
Max Error	The maximum error for one approach point.

The calculation result can be saved in a separate file for later use in a PC. However, this file cannot be read by the robot:

- Press the function key ***File***.
- Specify a name and a place to save the result.
- Choose ***OK*** to confirm the save.

• If the estimated error is

- acceptable, press ***OK*** to confirm the new tool coordinate system;
- not acceptable, redefine by pressing ***Cancel***.

The definition is now complete, but before proceeding with other tasks, verify it by linearly jogging in the tool coordinate system and by reorienting the TCP.

If the tool has been stored in a system module, save this module.

7.5 Stationary tool

When using a stationary tool, the robot is holding the work piece and the tool is stationary in the room. In this case the TCP coordinates are related to the world coordinate system, and the work object (i.e. the user coordinate system) is related to the wrist coordinate system.

Creating a new tool.

• The tool is created as described in previous chapters.

• The component *robhold* is changed to FALSE.

Creating a corresponding work object

When using a stationary tool, it is also necessary to use a work object held by the robot.

• The work object is created as described in *Creating a new work object* on page 36.

• The component *robhold* is changed to TRUE.

Methods for defining the tool coordinate system

The methods are the same as for a TCP mounted on the robot. However in this case, the reference tip is mounted on the robot and the robot is moved, so as to bring the tip to the stationary tool TCP. The tip must be defined and activated as a tool before the definition of the stationary tool may be done.

• Define and activate the tool, which should be used as a pointing tip, and which is mounted on the robot.

• Now the same methods for defining the stationary tool may be used, as described in *Manually updating the TCP and weight of a tool* on page 29 and *Using the robot to change the TCP and orientation of a tool* on page 31. Use the robot mounted tip to point out the stationary TCP with four approach points, and if needed, the z and x directions of the axes. It is possible to use the same positioning for all four TCP approach points to perform a faster frame definition. However, it is recommended to point out the stationary TCP with different orientations to obtain a reliable statistical result. The point that is used to approach the stationary TCP must be the active TCP (hold by the robot).

8.1 General

All programmed positions are related to a program displacement frame, which in turn is related to the object frame, related to the user frame, related to the world frame. Both object and user frames are included in a work object, which may be added to each move instruction. See Figure 31.

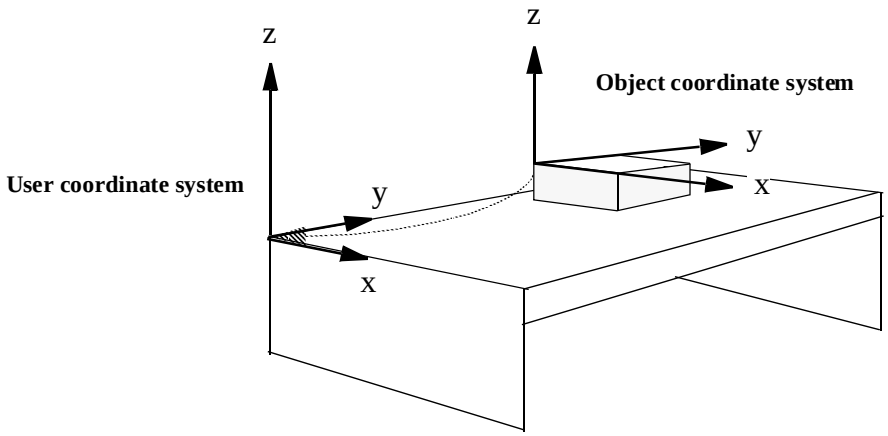


Figure 31 A user and an object coordinate system describe the position of a work object.

The intention is to use the work object to define both the position of a table (user frame) and the position of the object to work on (object frame). When the table or the object is moved, the program may still work if the corresponding work object is updated. These coordinate systems are very well suited to off-line programming since the positions specified can usually be taken directly from a drawing of the work object.

The program displacement coordinate system is used for small temporary displacements, e.g. as the result of a search operation. This displacement is modal, i.e. it is activated in a separate instruction and then it remains active until it is deactivated in another separate instruction. See Figure 32.

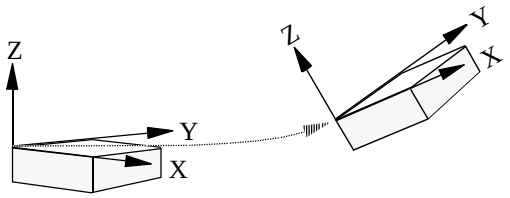


Figure 32 Using a displacement frame, all positions in the program can be displaced.

All such program displacements include both robot displacements and external axes displacements.

Please note the difference between work object and program displacement. The work object used must be added to each move instruction and it must be active when programming the move instruction. It should be included from the beginning because it is a little tricky to add it afterwards. A program displacement, however, which is activated in a separate instruction, is very easy to add afterwards.

8.2 Using work objects

In the checklist below, the steps required to define and use a work object are described. In each step, there may be a reference to another chapter in this manual, where more details of the specific actions to be taken will be found.

- Before starting to program, the work objects to be used must be defined. First create a new work object and give it a name, e.g. “wobj1”, see *Creating a new work object* on page 36.
- Define the work object by using the robot to point out three points on the user frame and the object frame respectively. See *Using the robot to change the work object* on page 38. Please note that if the same positions are used both for the user frame and for the object frame, then all the locations will go into the user frame and the object frame will still be zero. It should also be noted that it is possible to update the values of the work object manually. See *Manually updating the user and object coordinate system of the work object* on page 37.
- Now check that the definition of the work object is correct by jogging the robot in the object coordinate system. This may be done by choosing the ***Wobj*** in the field ***Coord*** in the jogging window, and the work object, e.g *wobj1*, in the field ***Wobj***, and then jogging the robot.
- When programming it is important to have the work object, in this case *wobj1*, programmed as an argument in each move instruction. This will be automatically added to the move instruction, if the work object is activated in the jogging window before starting the programming.

8.3 Creating a new work object

A work object should normally be placed in the system module, *User*. In this way it will be common to all programs, which means that if a work object is modified, all programs will also automatically be modified. The work object can also be used for jogging when there is no program in the program memory.

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *wobjdata* and press Enter [↵].
- Create the new work object using one of the following alternatives:
  - **alt 1**. Press the function key ***New***.
    The user and object coordinate systems will then coincide with the world coordinate system.
  - **alt 2**. Select an existing work object and press the function key ***Dupl***.
    The coordinate systems will then be the same as those duplicated.

A window appears, displaying the name of the data.

- If you want to change the name, press Enter [↵] and specify a new name.
- Press the function key ***Decl***.

A dialog box appears, displaying the basic wobjdata declaration.

- If you want to save the data in another module, select the field ***In Module*** and press Enter ↵ . State the name of the module where the data is to be sent.
- Press ***OK*** to confirm.

Note: Do not change the work object type. This must always be of the persistent type.

8.4 Manually updating the user and object coordinate system of the work object

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *wobjdata* and press Enter ↵ .
- Select the work object to be changed and press Enter ↵ .
- Select the component (x, y, z, q1-q4) that you wish to change.
- Change the value using the numeric keyboard. To enter a decimal point (.) and minus sign (-), use the function keys.
- Choose ***OK*** to confirm.

Note If the work object is defined using a movable user coordinate system, only the object coordinate system need be defined. The user coordinate system is defined in the Service window. See *Coordinated external axes* on page 16.

8.5 Methods of defining a work object

The methods used to define the user and object coordinate system are called:

- ***No change***
  No changes to the definition of the user or object coordinate system will be made, i.e. the definition of the user or object frame will be left as it is.
- ***3-point***
  Three points are used: two points on the x axis and one point on the y axis (see Figure 33). A tool with a known TCP is required.

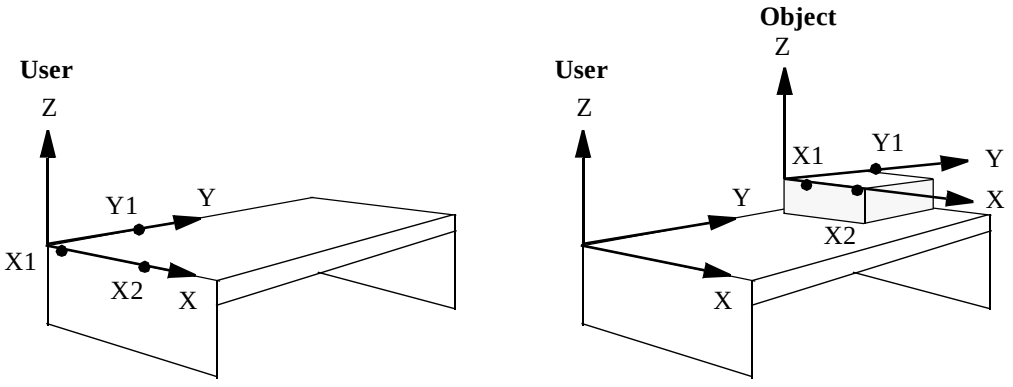


Figure 33 Measuring points for defining a work object.

- Choose **View: Data Types**.
- Select the type *wobjdata* and press Enter ↵.
- Select the work object to be defined (or create a new one, see *Creating a new work object* on page 36).
- Choose **Special: Define Coord**.

A dialog box appears, displaying the points defined by the method that was used (see Figure 34).

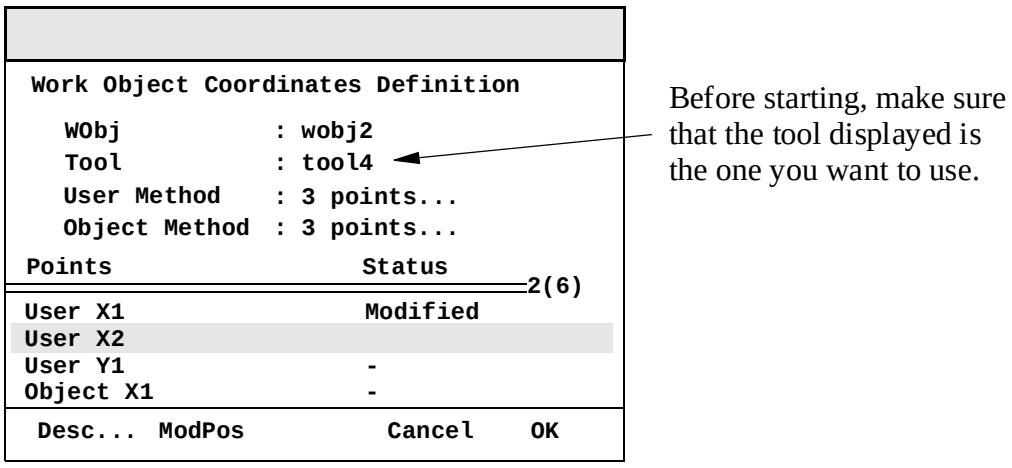


Figure 34 The robot can be used to define the position of the work object.

The status can be defined as follows:

Status	Meaning
-	No position defined
Modified	Position modified

To record Measuring Points for the user coordinate system

Note If the work object is defined using a movable user coordinate system, the user coordinate system is defined in the Service window. See *Coordinated external axes* on page 16.

- Select the first measuring point ***User X1***.
- Jog the robot as close as possible to a point on the x axis.
- Modify the position by pressing the function key ***ModPos***.
- Select the measuring point ***User X2***.
- Jog the robot as close as possible to a point on the x axis defining the positive x direction.
- Modify the position by pressing the function key ***ModPos***.

• Select the measuring point ***User Y1***.

• Jog the robot as close as possible to a point on the positive y axis.

• Modify the position by pressing the function key ***ModPos***.

To record measuring Points for the object coordinate system

• Select the first measuring point ***Object X1***.

• Jog the robot as close as possible to a point on the x axis.

• Modify the position by pressing the function key ***ModPos***.

• Select the measuring point ***Object X2***.

• Jog the robot as close as possible to a point on the x axis defining the positive x direction.

• Modify the position by pressing the function key ***ModPos***.

• Select the measuring point ***Object Y1***.

• Jog the robot as close as possible to a point on the positive y axis.

• Modify the position by pressing the function key ***ModPos***.

To calculate the user and object coordinate system

• Press ***OK*** to calculate the coordinate systems.

When the calculation is finished, a dialog like the one shown in Figure 35 will appear.

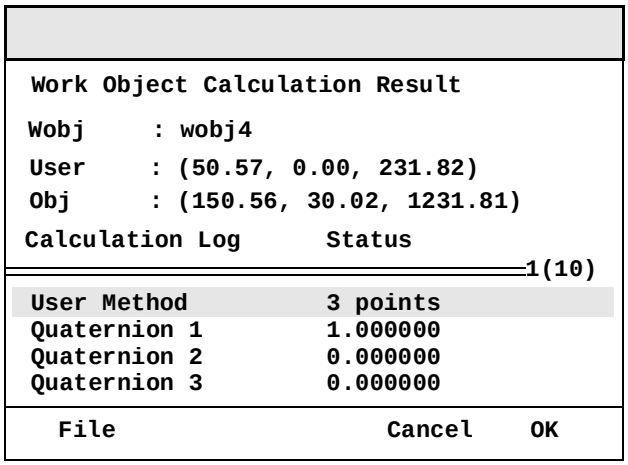


Figure 35 The result of a work object calculation.

Field	Description
User	The origin of the user coordinate system.
Obj	The origin of the object coordinate system.

The calculation result can be saved in a separate file for later use in a PC.

Note, however, that this file cannot be read by the robot:

- Press the function key ***File***.
- Specify a name and a place to save the result.
- Choose ***OK*** to confirm the save.

The definition is now complete, press ***OK*** to confirm the new work object, but before proceeding with other tasks, verify it by jogging linearly in the work object's coordinate system.

If the work object was stored in a system module, save this module.

8.7 Defining a moveable object frame

Method 1

- Use the method for defining a work object. See *Using the robot to change the work object* on page 38. When using this method, please observe that the coordination flag, i.e. the component *ufprog* in the work object data must be temporarily set to TRUE. You must point out three positions for the user system (which must be placed as the coordinated one) and three positions for the object system.

If the user system is not possible to reach, use method 2 or 3 below.

Method 2

- Activate the coordinated work object and jog the robot to the point where you want to place the origin of the object frame.
- Read the coordinates, x, y, z for this position in the jogging window.
- Write these values in the *o_frame* component of the work object data.

This will shift the object frame to the new position, with the same orientation as the user frame. If you want another orientation, use method 3.

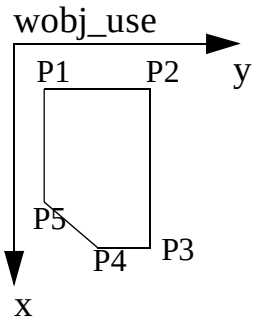
Method 3

- Activate the coordinated work object (suppose it is named *co_wobj*), create three positions, e.g. *p1*, *p2* and *p3*. *p1* should be located at the origin of the shifted object frame, *p2* on the x axis and *p3* in the x-y plane.
- Program and execute the instruction
  co_wobj.oframe: = DefFrame(p1, p2, p3);

8.8 How to use different work objects to get different displacements

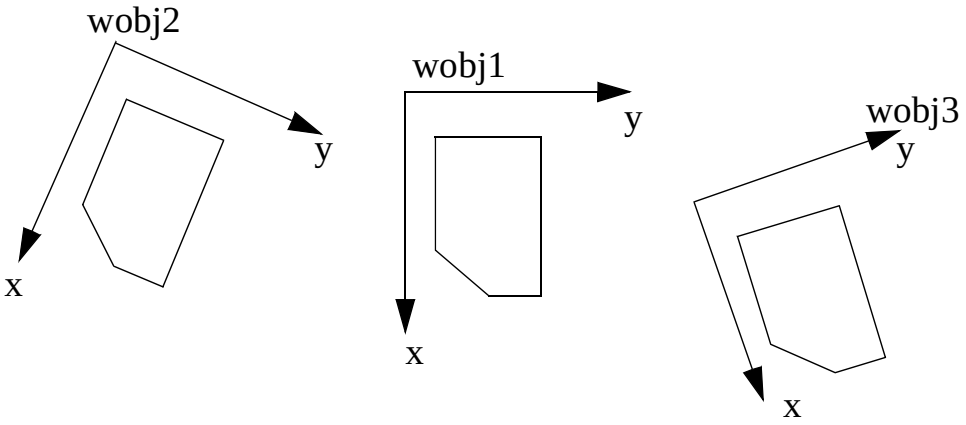
Suppose you have used the work object *wobj_use* when creating a procedure, *draw_fig*, as below.

```
MoveL p1, v200, z1, tool1\WObj:=wobj_use;
MoveL p2, v200, z1, tool1\WObj:=wobj_use;
MoveL p3, v200, z1, tool1\WObj:=wobj_use;
MoveL p4, v200, z1, tool1\WObj:=wobj_use;
MoveL p5, v200, z1, tool1\WObj:=wobj_use;
```



Now you want it to be performed displaced, corresponding to *wobj1*, *wobj2* or *wobj3*, see below.

:



Suppose that the value of *reg1* is used to control which work object should be used.

If *reg1 = 1*, *wobj1* should be used; if *reg1 = 2*, *wobj2* should be used; and if *reg1 = 3*, *wobj3* should be used.

The program below will set wobj_use = wobj1 if reg1 = 1, then call the *draw_fig* procedure, etc.

```
IF reg1=1 THEN
                    wobj_use:=wobj1;
                    draw_fig;
ENDIF
IF reg1=2 THEN
                    wobj_use:=wobj2;
                    draw_fig;
ENDIF
IF reg1=31 THEN
                    wobj_use:=wobj3;
                    draw_fig;
ENDIF
```

8.9 How to adjust the program vertically using the object frame

When running your program in the location defined by *wobj2*, suppose you find it is positioned a little too high. The vertical position can be adjusted by moving the object coordinate system a small amount vertically, relative to the user coordinate system, i.e. the z coordinate for object is changed. E.g. if the robot is to work a little lower, then the z value should be decreased.

8.10 Using program displacement

A program displacement is set with a *pose* data, using a *PDispSet* instruction. This will store the program displacement in a system variable, *C_PROGDISP*, holding also displacement values for external axes. The current value in *C_PROGDISP* is used in all movement instructions and added to the programmed positions. The program displacement is cleared, when a *PDispOff* instruction is executed, resulting in no further displacement.

A *PDispOn* instruction will both calculate a new program displacement, from the difference between two positions, and store this displacement in the *C_PROGDISP* variable. When this instruction has been executed a new program displacement will become active.

The following example will illustrate how to use a *PDispOn* instruction in combination with a *SearchL* instruction, to make a movement on different locations, depending on the search point.

The program should do the following:

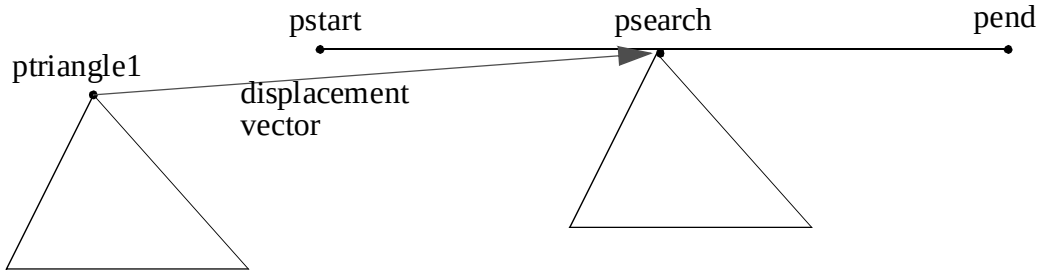
- Go to a start point, *pstart*, for searching.
- Make a linear search from the start position to an end position, *pend*. When a digital input *di1* is set, the robot should stop the movement and draw a figure, *triangle*, the position of which will depend on the search point, *psearch*.

The figure, *triangle*, is programmed with no displacement active and with the first position in *ptriangle1*.

The program may look like:

```
MoveL pstart, v200, fine, tool1;
SearchL \Stop, di1, psearch, pend, v100, tool1;
PDispOn \ExeP: = psearch, ptriangle1, tool1;
triangle;
PDispOff
etc.
```

.



8.11 Creating a new displacement frame

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *pose* and press Enter ↵.
- Create the new displacement frame using one of the following alternatives:
  - **alt 1**. Press the function key ***New***.
    The displacement frame will then have no translation or rotation.
  - **alt 2**. Select an existing displacement frame and press the function key ***Dupl***.
    The displacement frame will then be the same as the one duplicated.

A window appears, displaying the name of the data.

- If you want to change the name, press Enter ↵ and specify a new name.
- Press ***OK*** to confirm.

8.12 Manually updating a displacement frame

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *pose* and press Enter ↵.
- Select the displacement to be changed and press Enter ↵.
- Select the frame component (x, y, z, q1-q4) that you wish to change.
- Change the value using the numeric keyboard. To enter a decimal point (.) and minus (-), use the function keys.
- Choose ***OK*** to confirm the change.

8.13 Methods for defining a displacement frame

The following method is supported:

- ***n-point***
  At least three well-defined points on an object at its initial position and the same points when the object is in its new position (see Figure 36) are used to define the displacement frame.

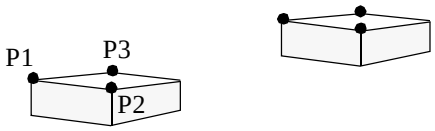


Figure 36 A displacement frame can be defined by moving the robot to a number of points.

8.14 Using the robot to change a displacement frame

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *pose* and press Enter ↵.
- Select the displacement frame to be defined (or create a new one, see ***Creating a new displacement frame*** on page 43).
- Choose **Special: Define Coord**.

A dialog box appears, displaying the points defined by the method that was used (see Figure 37).

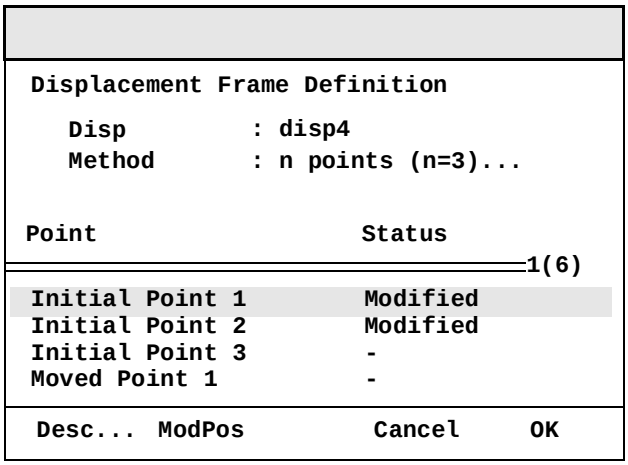


Figure 37 Displacement frame definition dialog

The status can be defined as follows:

Status	Meaning
-	No position defined
Modified	Position modified

To choose the definition method

Before you start modifying any positions, make sure the ***n-point*** method, together with the number of points that you want to use, is displayed:

- Select the field ***Method*** and press Enter [↵].
- Enter the desired number of points and press ***OK***.

To record the Initial Points

- Select the first definition point ***Initial Point 1***.
- Jog the robot as close as possible to a well-defined position on the object.
- Modify the position by pressing the function key ***ModPos***.
- Repeat the above for the points ***Initial Point 2***, ***Initial Point 3***, etc.

To record Moved Points

- Move the object to its new position.
- Select the first definition point ***Moved Point 1***.
- Jog the robot as close as possible to the same position on the object as for ***Initial Point 1***.
- Modify the position by pressing the function key ***ModPos***.
- Repeat the above for the points ***Moved Point 2***, ***Moved Point 3,*** etc.

To calculate the displacement frame

- Press ***OK*** to calculate the displacement frame.

When the calculation is finished, a dialog like the one shown in Figure 38 will appear.

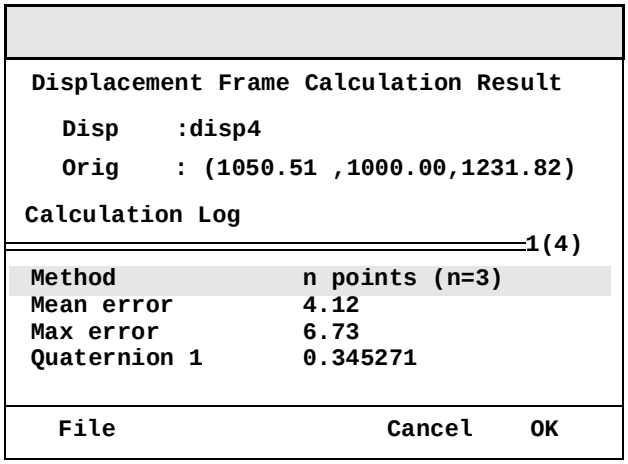


Figure 38 The result after a displacement frame calculation.

Field	Description
Orig	The origin of the displacement frame.
Mean Error	The average distance that the points are from the original points, i.e. how accurately the robot was positioned.
Max Error	The maximum error for one point.

The calculation result can be saved in a separate file for later use in a PC. Note, however, that this file cannot be read by the robot:

- Press the function key ***File***.
- Specify a name and a place to save the result.
- Choose ***OK*** to confirm the save.

• If the estimated error is

- acceptable, press ***OK*** to confirm the new displacement frame;
- not acceptable, redefine by pressing ***Cancel***.

CONTENTS

Page

Production Running

The Production window appears automatically on the teach pendant display when the power is switched on, and the operating mode selector is in the Auto position. You can also call it up by pressing  and choosing ***Production***.

1 The Production Window

The Production window is used to start and stop program execution (see Figure 1). There are two different views in the Production Window. Production Info is described below and Production Position is described in 7 *Tuning position* on page 7.

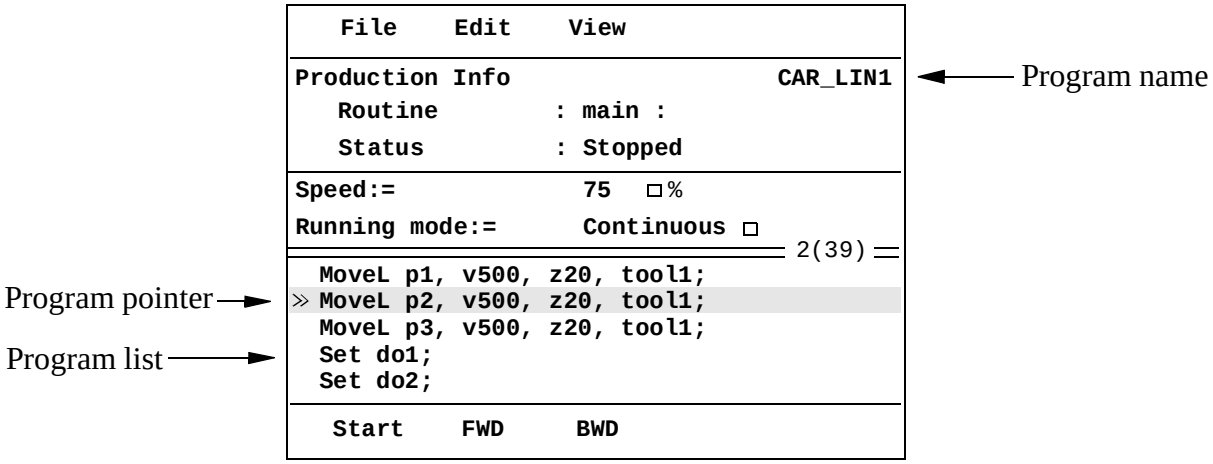


Figure 1 All production runs are controlled from the Production window.

Before starting the program, check the program name to see that it is the correct program. The Program name is displayed in the right hand upper corner of the window.

• Choose **View:Info** to open the window *Production Info*.

To start the program, see *5 Starting the Program* on page 6.

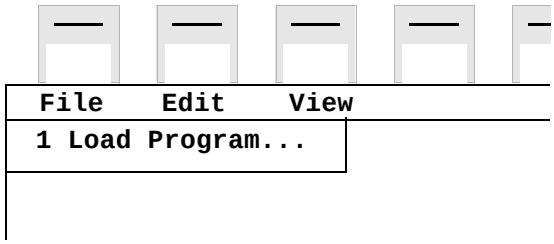
If the Status field indicates NOT LOADED, then you must load a program (see *2 Reading a Program* on page 4).

Field:	Indicates:		
Routine	The subprogram that is being run		
Status	NOT LOADED	=	no program is loaded
	STOPPED	=	a program is loaded and it can be executed (PP is set)
	RUNNING	=	program execution is in progress
	NOT EXECUTABLE	=	a program is loaded but cannot be executed
Speed	The chosen speed correction as a percentage		
Running mode	Continuous	=	continuous execution
	Cycle	=	the program is executed once
Program list	The part of the program that is being run		
Program pointer	The instruction to be executed when ***Start*** is pressed.		

2 Reading a Program

A program can be read from a diskette or from the robot's mass memory. To open a program, do as follows:

• Choose **File: Load Program**.



The following dialog box will appear (see Figure 2).

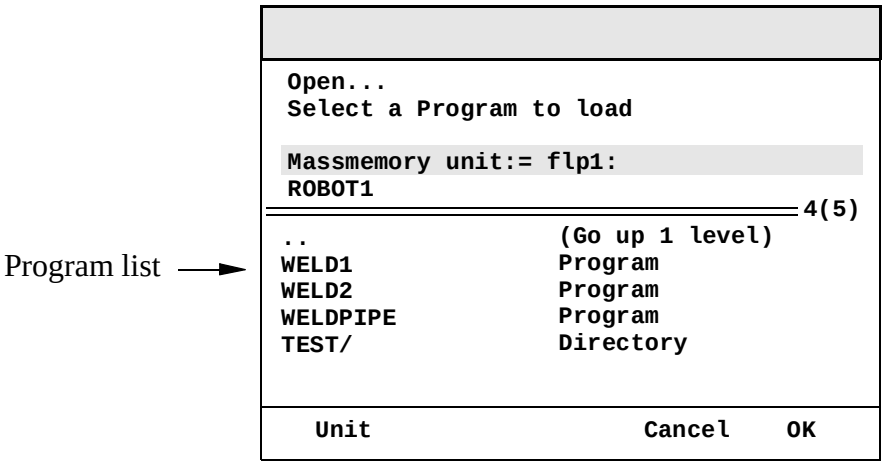


Figure 2 The dialog box displays a list of all available programs.

The Mass memory unit field indicates:

- **flp1** to denote a diskette
- **ram1disk** to denote the robot's internal memory (the RAM disk)

• Press ***Unit*** until the desired unit is displayed.

• Choose the desired program – use ArrowUp or ArrowDown to scroll through the list: select . . to go up one level and press to go down one level.

• Press ***OK***.

3 Changing the Override Speed

The speed of the robot can be adjusted while running production. The function keys indicate how the speed can be decreased or increased.

–%	Decreases the value by 5% (or 1% if <5%)
+%	Increases the value by 5% (or 1% if <5%)
25%	Sets the value to 25%
100%	Sets the value to 100%

To override the speed, do as follows:

• Select the middle part of the display by pressing  .

• Using one of the arrow keys, select the field for the corrected speed (see Figure 3).

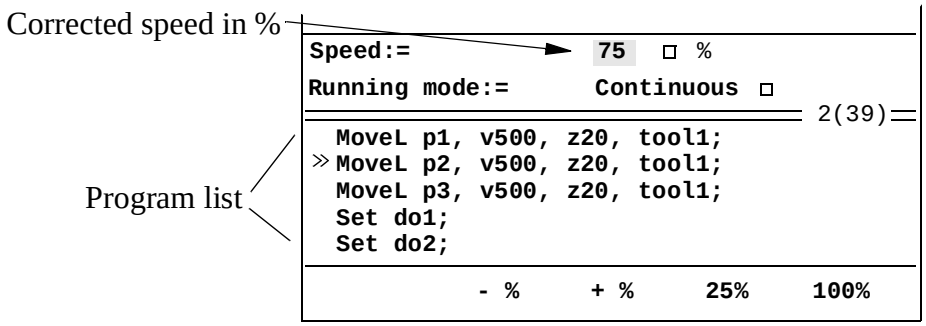


Figure 3 The function keys can be used to both increase and decrease the programmed speed.

• Press the desired alternative.

• To return to the program list, use  .

4 Changing the Program Running Mode

A program can be run in either of the following two ways:

- ***Cont*** – continuous execution
- ***Cycle*** – the program is executed once.

You can change the program running mode in the ***Running mode*** field:

• Select the middle part of the display by pressing  .

• Select ***Running mode*** (see Figure 4).

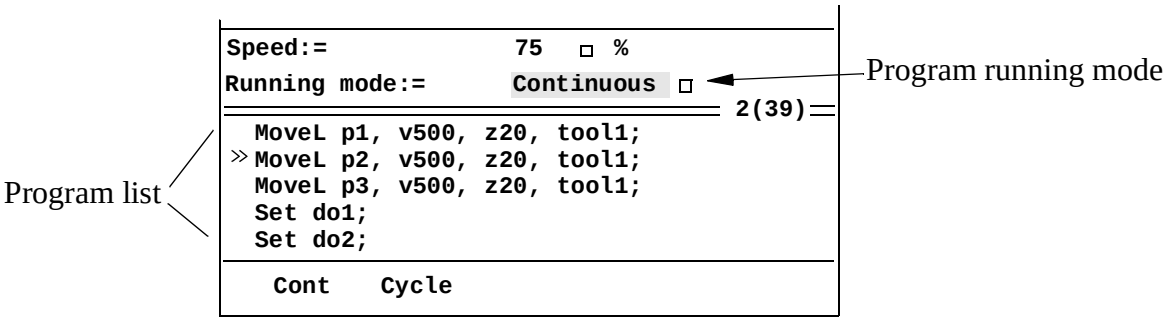


Figure 4 The function keys are used to select the different program running modes.

- Press the desired function key ***Cont*** or ***Cycle***.
- To return to the program list part, use  .

5 Starting the Program

When the program is started, the robot and any peripheral equipment will start to move. Check that all preparations have been made for program execution. Make sure that the workcell is clear of all personnel before starting the robot.

If NOT LOADED is displayed on the program status line, then a program must be loaded (see *2 Reading a Program* on page 4).

If a program is loaded and is executable, STOPPED will be displayed on the program status line and the program can be started:

- Press the function key ***Start***.

5.1 Restarting after a stop

If you wish to restart program execution from where it was interrupted:

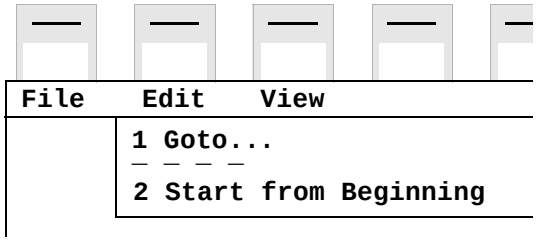
- Press ***Start***.

The program can also be restarted from the beginning. This is described below.

5.2 Starting a program from the beginning

To start again from the beginning, proceed as follows:

• Choose **Edit: Start from Beginning**.



• Press ***OK*** to confirm.

The program pointer ≫ will then move to the first instruction in the program.

• Press ***Start***.

6 Stopping the Program

Program execution can be stopped by pressing the stop button on the teach pendant (see Figure 5).

In case of an emergency, press one of the red stop buttons instead. This will cut off the power supply to the robot motors and engage all brakes.

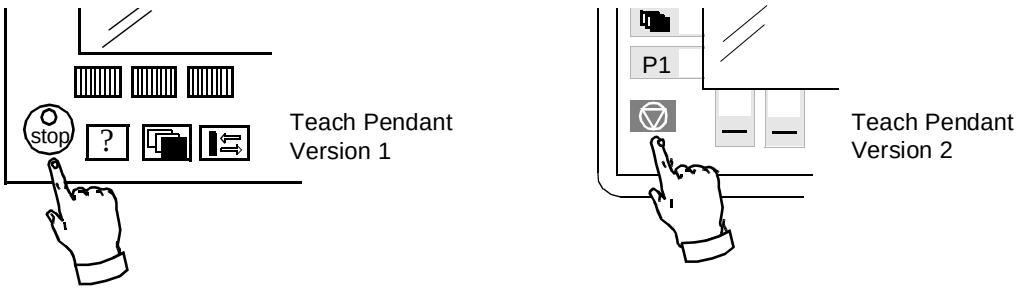
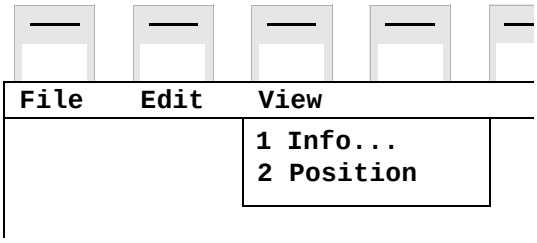


Figure 5 This stop button is used to stop the program.

7 Tuning position

• Choose **View: Position**



The tuning function in the Production window makes it possible to tune the x, y and z coordinates of a robot position (see Figure 6). The tuning can be performed either when status is Stopped or when status is Running.

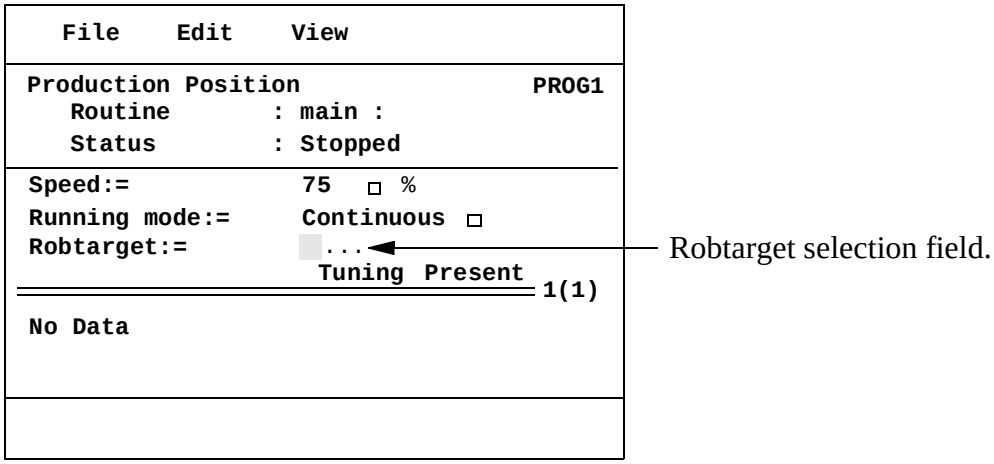


Figure 6 The Production Info view. No robtarget selected.

• Select the field ***Robtarget*** and press Enter ↵.

• Choose the position to be tuned in the list that will appear.

• Press ***OK*** or Enter ↵ to confirm the choice.

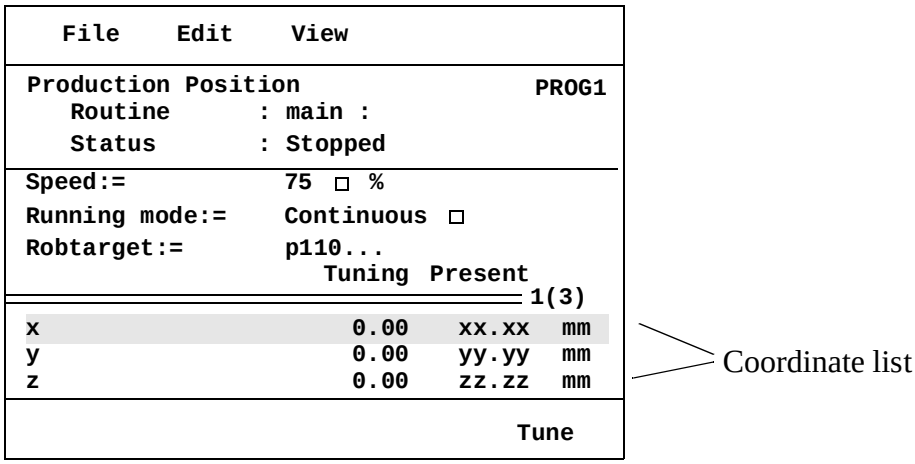


Figure 7 The Production Position view with a robtarget selected.

• Choose the x, y or z coordinate in the coordinate list (see Figure 7).

• Press: **Tune**

A dialog will appear where you can tune the position.

• Enter the desired tuning value and press Enter ↵.

- No change = 0.
- Max. change in one step = ±10 mm

Several steps can be entered. The position data is changed immediately after each step but will not affect the robot path until the next instruction using this position data is executed. The values in the ***Present*** column will be used in this instruction.

The total tuning will be displayed in the ***Tuning*** column.

Note If a named position data is modified, all instructions which refer to that position data will be affected. Unnamed positions (marked as * in the instruction) cannot be tuned.

See also Chapter 8 Programming and Testing - *Tuning position during program execution.*

The tuning function can be disabled in automatic mode. See chapter 12 System Parameters - *Topic:Teach Pendant.*

8 Operator Dialogs

Special instructions can be created in the program and used as a form of communication between the program and the operator (see Figure 8).

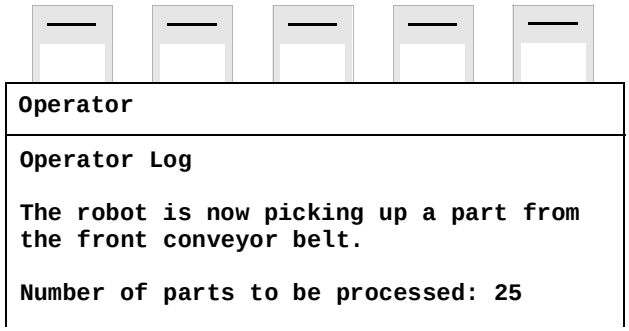


Figure 8 Example of a message sent to the operator.

• You can return to the *Production* window by pressing  and choosing ***Production***.

Sometimes, the operator must respond before program execution can continue (see the example in Figure 9).

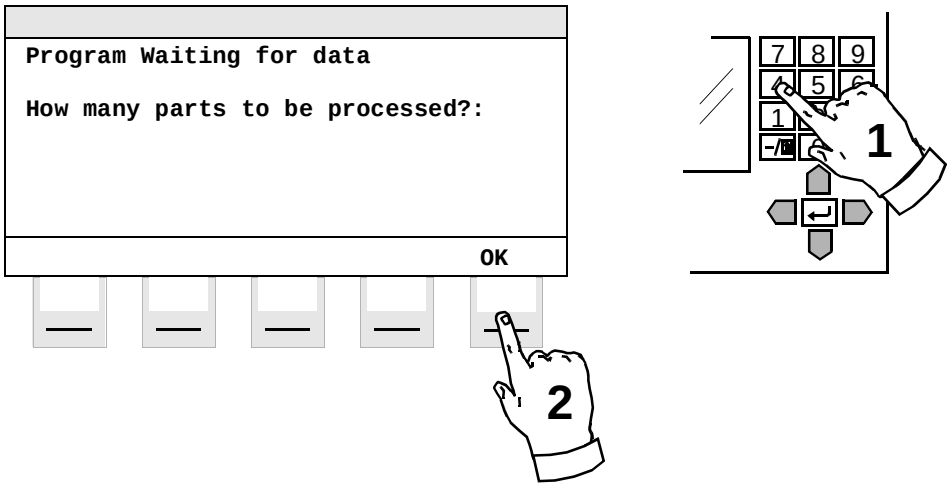


Figure 9 Use the numeric keyboard to answer questions from the program.

• Use the numeric keys when the reply is a numeric value.

• Press ***OK***.

• If text is displayed above the function keys, you can give your answer by pressing the desired alternative (see Figure 10).

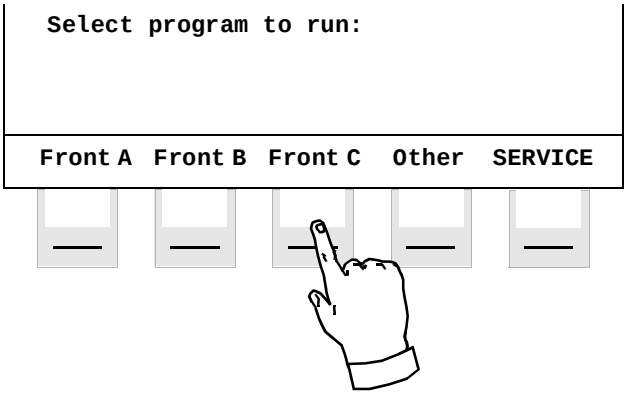


Figure 10 Operator dialogs can be tailor-made to suit any robot installation.

The dialog window shown in Fig. 10 can only be exited in one of two ways, either by answering the question or by stopping program execution.

CONTENTS

Page

Page

System Parameters

The *system parameters* describe the equipment and area of application of the robot system, e.g. I/O names and the characteristics of the external axes.

1 Changing a Parameter

1.1 Subdivision of parameters

The available parameters are grouped together in a number of different *topics*. These topics are, in turn, divided up into different *type*s.

Topic	Parameters that affect	File name
Controller	Event routines, etc.	SYS.CFG
Communication	Serial channels	SIO.CFG.
IO Signals	I/O boards and signals	EIO.CFG
Manipulator	The robot and external axes	MOC.CFG
TeachPendant	Displaying data and access on the teach pendant	MMC.CFG
Arc Welding	Arc welding	PROC.CFG

• To view all parameters, choose **Topics: All Topics** in the System Parameters window.

All relevant topics in the robot system will then be displayed. Choose the desired topic by selecting it and pressing Enter [↵].

1.2 Changing a parameter

• Press the Miscellaneous key  to open the System Parameters window.

• Select ***System Parameters*** from the dialog box that appears.

• Press ***OK*** or Enter [↵].

• Call up the parameter type that contains the parameter to be changed, by choosing a topic from the **Topics** menu and a type from the **Types** menu.

All parameters of that type will be displayed, as illustrated in Figure 1. To be able to display some parameter types, however, you must first choose the current unit, such as an I/O board or a signal, by selecting it and pressing Enter [↵].

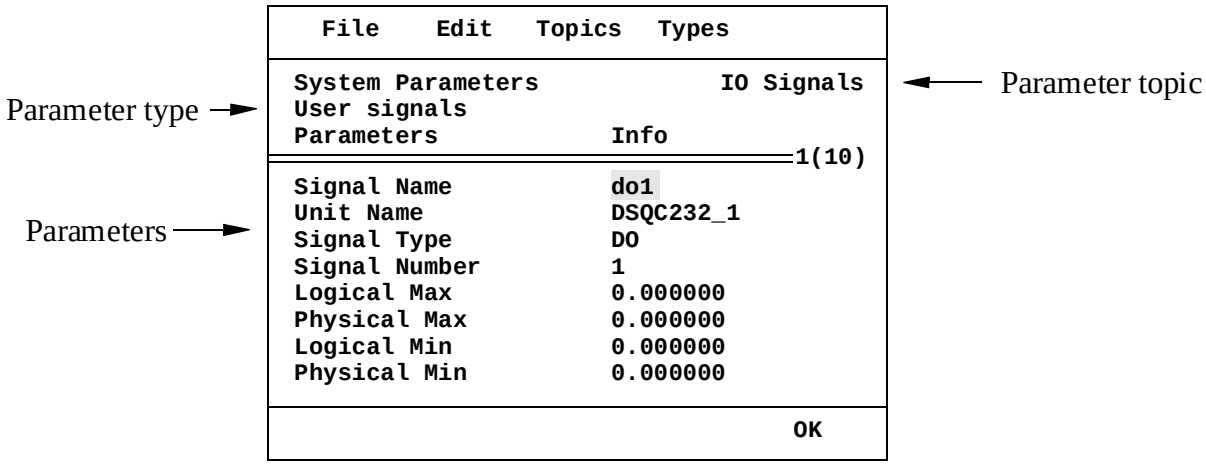


Figure 1 All parameters of a given type are displayed in the window at the same time.

- Select the parameter to be changed.
- Change the value of the parameter by
  - pressing Enter [↵] and specifying the desired alternative in the dialog box that appears,
  - choosing an alternative from the function keys (fields marked with ▫).

All parameters, together with possible values, are described in the following sections under the appropriate topic.

Note. You will have to restart the robot in order for the change to have an effect on some parameters. You will be informed of this the first time you change such a parameter and when you exit the system parameters, i.e. change window.

1.3 Deleting a parameter

- Select the parameter to delete
- Press ***Delete*** [⌫] .
- Press ***OK*** to confirm the delete.

1.4 Generating a restart

You have to restart the robot in order for a change to have an effect on some of the parameters. If you exit the system parameters without generating a restart, the parameter values will not be the same as those used in the robot. Nevertheless, if you generate a restart at a later stage, then the changes will take effect.

- Choose **File: Restart** and press ***OK*** or turn the mains switch off and then on again.

An error message will be displayed when there is an error in the parameters. However, this can be due to a sequential error. The origin of an error can be found by looking at the robot's error logs. See chapter 14, Service: *Logs*.

1.5 Viewing the last changes that were made

• Choose **Edit: Show Change Log**.

A dialog box appears, displaying the changes that were last made (see Figure 2).

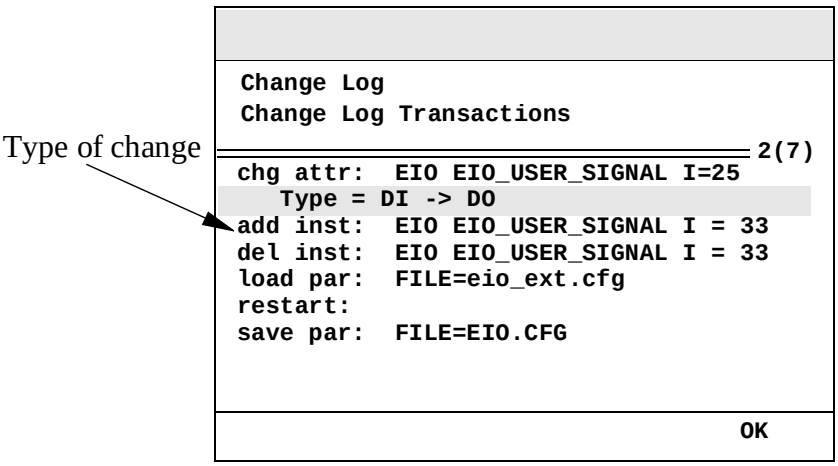


Figure 2 The Change Log dialog box.

The following identification tags are used:

- chg attr: Shows the parameter that has been changed and how it was changed.
- add inst: Shows that a new parameter has been added.
- del inst: Shows that a parameter has been deleted.
- load par: Shows that new parameters have been loaded.
- save par: Shows that parameters have been saved.
- restart: Shows that the robot has been restarted.

1.6 Checking Parameters

When you have changed a parameter, it is a sensible idea to check the change before restarting, in order to avoid problems when restarting. In the current version the Manipulator area can be checked.

• Select the area to be checked in **Topics** (only Manipulator can currently be checked)

• Select **File: Check Parameters** and the check will start.

When the check is finished, a report will be made showing that either there was an error or the change of the parameter was done correctly. The error will be reported via the usual error log. See chapter 14, Service.

2 Saving and Loading Parameters

2.1 Saving parameters to diskette or some other mass storage device

The system parameters can be stored in their entirety or stored as individual parameter topics, for example, on a diskette.

To save all parameters

The parameters are always saved to a directory in the form of a separate file for each topic.

• Choose **File: Save All As**.

A dialog box appears, displaying all files in the current directory (see Figure 3).

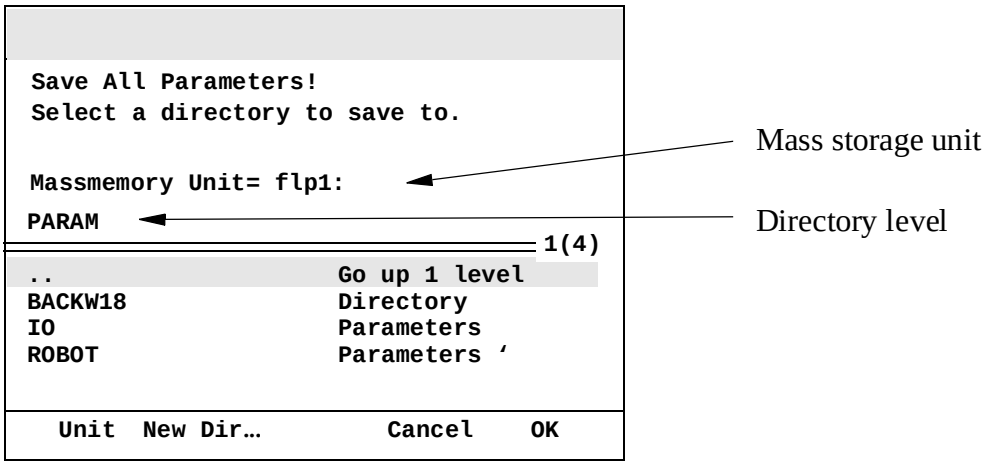


Figure 3 The dialog box used to store parameters.

• If necessary, change the mass storage unit by pressing the ***Unit*** function key until the correct unit is displayed. To store on a diskette, choose ***flp1:***.

• Select the directory to which the parameters are to be saved. You can move to the next directory level by selecting the desired directory or ‘. .’ (upwards) and pressing Enter ↵.

Create a new directory by pressing the ***New Dir*** function key. Specify the new directory name in the dialog box that appears. Choose ***OK*** to confirm.

• Choose ***OK*** to confirm the save.

To save an individual parameter topic

• Choose **File: Save As**.

A dialog box appears, displaying all the previously saved parameters in the current directory.

- If necessary, change the mass storage unit by pressing the ***Unit*** function key until the correct unit is displayed. To store on a diskette, choose ***flp1:***.
- Specify the file name by selecting the field ***Name*** and pressing Enter ↵. Enter the desired name and press ***OK*** to confirm.
- Select the directory to which the parameters are to be saved. You can move to the next directory level by selecting the desired directory or '. .' (upwards) and pressing Enter ↵.
- Choose ***OK*** to confirm the save.

2.2 Loading parameters from a diskette or some other mass storage device

Parameters can be loaded in their entirety or loaded as individual parameter topics. If several parameters are to be loaded, the parameters must be placed in a directory.

Parameters previously saved (using **Save All As** or **Save As**) can be loaded back with **File: Load Saved Parameters,** and new parameters can be loaded with **File: Add New Parameters**.

- Choose **File: Load... (Saved Parameters or New Parameters)**

A dialog box appears, displaying all parameters in the current directory (see Figure 4).

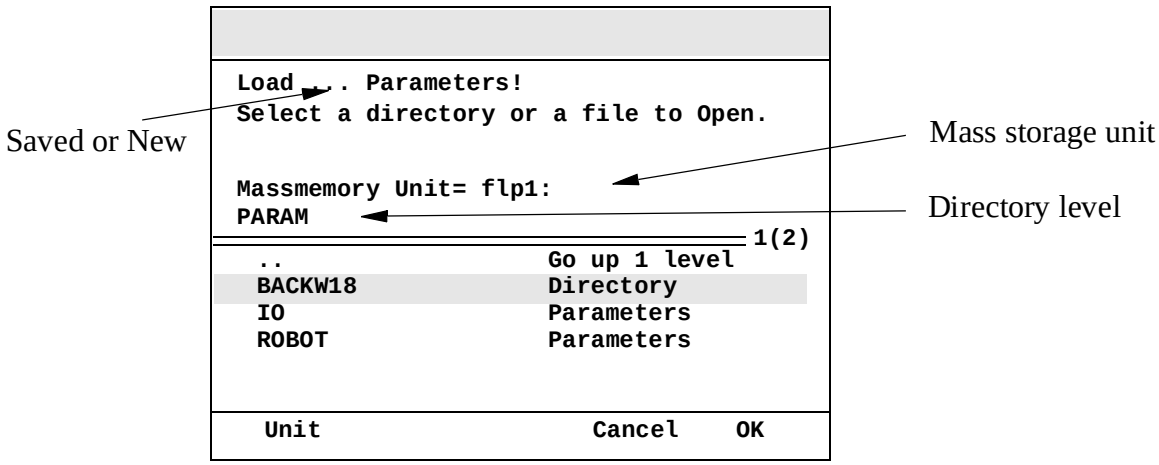


Figure 4 The dialog box used to load parameters.

- If necessary, change the mass storage unit by pressing the ***Unit*** function key until the correct unit is displayed. To load parameters from a diskette, choose ***flp1:***.
- Select the directory from which the parameters are to be loaded. You can move to the next directory level by selecting the desired directory or '. .' (upwards) and pressing Enter ↵.
- Choose ***OK*** to confirm the load.

An alert box will be displayed during reading. After this the robot must be restarted (see *Generating a restart* on page 4).

3 Topic: IO Signals

The following parameters are found under the topic IO Signals:

- Specification of all I/O boards
- Name and characteristics of input and output signals
- Groups of digital signals
- System signals
- Cross connections

• Choose **Topics: IO Signals**.

3.1 Defining I/O boards

• Choose **Topics: IO Signals**

• Choose **Types: IO Boards**.

All defined boards will be displayed, as shown in Figure 5.

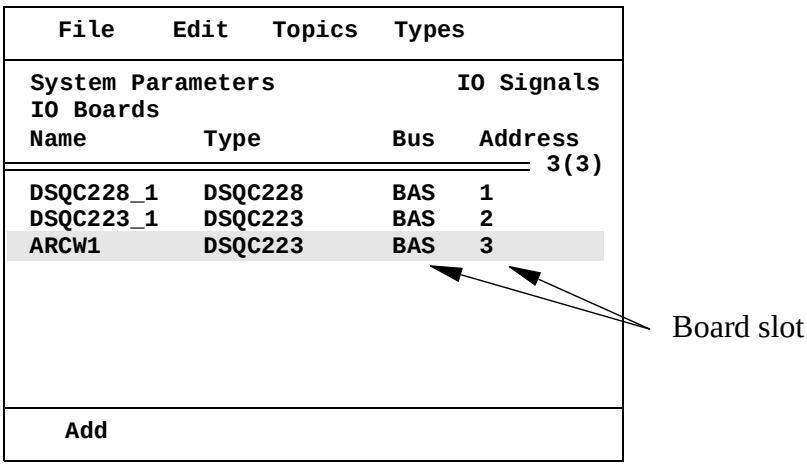


Figure 5 System parameters of the type IO Boards.

• Select the appropriate I/O board to be changed and press Enter [↵] , or add a new one by pressing ***Add***.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Unit Name	The name of the board (max. 16 characters).
Unit Type	Board type. Compare with the designation on the front of the board. Unit type *eip000* and *eip001* are simulated digital and analog units, *eiobus* and *eiolog* are internal boards and are not to be selected.
Unit Bus	The bus on which the board is located, normally BAS, except for simulated units which are located on the SIM bus.
Address	The address (slot) of the board on the bus.
Digital Outputs	The number of digital output signals.
Digital Inputs	The number of digital input signals.
Analog Inputs	The number of analog input signals.
Analog Outputs	The number of analog output signals.
RIO Node Add (octal)	The node address for the Allen Bradley PLC. The range is (1->77), and the address is input in octal form.
RIO Baud Rate	The baud rate on the Allen Bradley RIO link. Choice (57,6kB, 115kB, 230kB).
RIO Start Group	The starting group of the Allen Bradley RIO Link. Choice (0, 2, 4, 6).
RIO Rack size	The rack size of the Allen Bradley RIO board. Choice (1/4, 1/2, 3/4, FULL)
RIO Last rack	Set to Yes if the RIO board is the last Rack in the PLC polling chain.
Slot Addr	Not used
IBS Rack size	The rack size of the Interbus-S D260 board. Choice: 1/4 = 16 inputs, 16 outputs 1/2 = 32 inputs, 32 outputs 3/4 = 48 inputs, 48 outputs Full = 64 inputs, 64 outputs

To delete a board

• Select the appropriate board.

• Press ⌫.

All the signals on this board will remain defined. These must be deleted separately.

3.2 Defining input and output signals

• Choose **Topics: IO Signals**.

• Choose **Types: User Signals**.

All named signals will be displayed (see Figure 6).

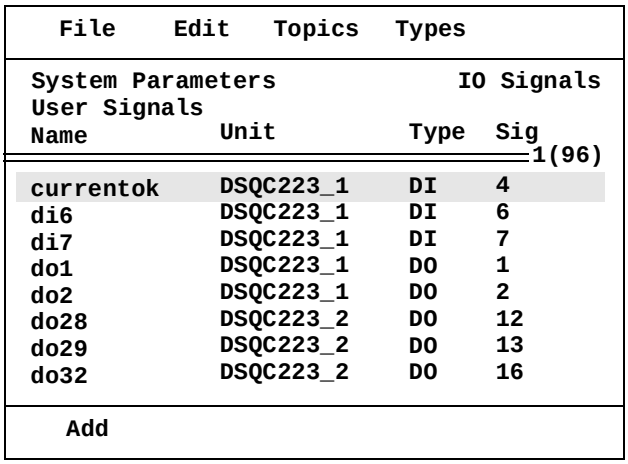


Figure 6 System parameters of the type User Signals.

Note that several signals can be connected to the same physical channel. The same signals cannot, however, be connected to different physical channels.

• Select the signal to be changed and Press Enter ↵, or add a new one by pressing ***Add***.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Signal Name	The name of the signal (max. 16 characters).
Unit Name	The board to which the signal is connected.
Signal Type	The type of signal: ***DI***= Digital Input, ***DO***= Digital Output, ***AI***= Analog Input, ***AO***= Analog Output.
Signal Number	The physical channel to which the signal is connected. The numbering restarts from 1 for every board.
Logical Max ***Physical Max*** ***Logical Min*** ***Physical Min***	The scaling between the programmed and physical value of an analog signal (see Figure 7). Must be set to 0 for digital signals.

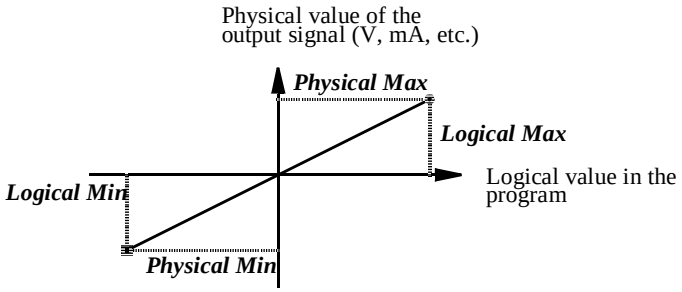


Figure 7 Diagram to show how analog signal values are scaled

Logical max/min. is the maximum/minimum value that an analog input or output can be set to, from a RAPID program or from the teach pendant. The units are user define (e.g. meter/ minute).

Physical max/min. is the maximum/minimum physical value that can be set on the output or input. To obtain the physical limit for a specified board, see the Product Manual.

If both physical and logical max/min. are set to 0, the default values for the board are picked up, which are the physical maximum and minimum limits. The logical and physical is set to the same value, which gives an amplification factor of 1.
If any of the values is set by the user, all four must be defined. Therefore, make sure that:
physical maximum is > physical minimum
logical maximum is > logical minimum.

Example: An analog board is controlling a current source with an amplification of 50 A/V and a max current of 500A. The following settings could then be applicable.

Physical Max = 10 V
Physical Min = 0 V
Logical Max = 500 A/V
Logical Min = 0 A/V

Filter Passive The time (in millisecs) that a digital input signal must be zero, before the robot acts on the signal change (100 ms to 32 s).

Filter Active The time (in millisecs) that a digital input signal must be 1, before the robot acts on the signal change (100 ms to 32 s). Filter resolution 10 ms.
A maximum of 5 different filter times can be used. If two signals are connected to the same physical channel, the filter time for these signals must be the same. The filter parameters must be set to 0 for analog signals and digital outputs.

Inverted Set to YES, if the digital signal shall be inverted, i.e. if logical ”1” should be set on the output as ”0”. This must be ”NO” for analog signals.

Store If set to YES, the digital outputs will be stored at a power failure, and restored when the system is powered up again. It should be noted that the value is connected to a logical signal. If more than one logical signal is connected to the same physical signal, an unexpected value may be obtained. In such cases this parameter should be set to NO.

Maximum number of user defined signals including group signals = 256.

3.3 Defining signal groups

Digital signals can be grouped together and handled as if they were one signal. The value of such a signal will thus be a positive integer that is binary coded using the individual digital signals as a basis.

• Choose **Topics: IO Signals**.

• Choose **Types: Groups**.

All defined signal groups will be displayed (see Figure 8).

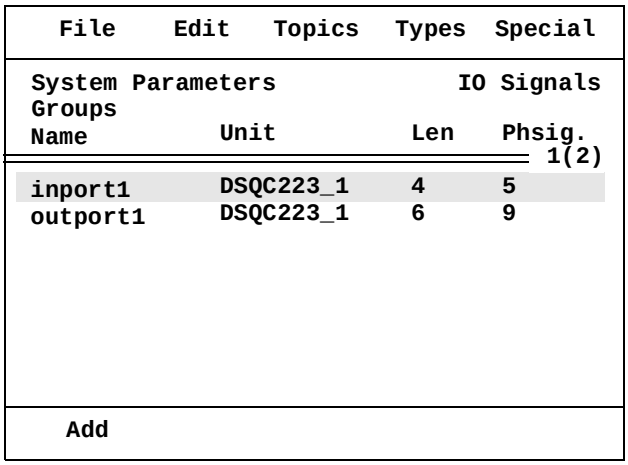


Figure 8 System parameters of the type Groups.

To view all signals contained in a given group, select the group in question and press Enter [↵].

• Select the signal group to be changed, and Press Enter [↵], or add a new one by pressing ***Add***.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Signal Name	The name of the signal (max. 16 characters).
Unit Name	The board to which the signal is connected.
Signal Type	The type of signal: ***GI***= Group of input signals, ***GO***= Group of output signals.
Group length	The number of digital signals in the group. The length must be set so that the group is within one board. The maximum value for length is 16.
Start signal	The physical channel of the board to which the first signal (the least significant) is connected. The remaining signals are automatically connected to the next channels on the same board.
Inverted	Set to Yes if all signals in the group shall be inverted.

Store If set to YES, the digital outputs will be stored at a power failure, and restored when the system is powered up again. It should be noted that the value is connected to a logical signal. If more than one logical signal is connected to the same physical signal, an unexpected value may be obtained. In such cases this parameter should be set to NO.

3.4 Defining cross connections

A digital input or output signal can be logically connected to one or several digital input or output signals. This means that a cross-connected signal will automatically be changed when the “activation” signal is changed. For more information, see RAPID Reference Manual - *Motion and I/O Principles*.

- If the signal has not already been defined, then define its name in the normal way. See *Defining input and output signals* on page 11.
- Choose **Types: Cross Connections**.

All the defined cross connections will be displayed (see Figure 9).

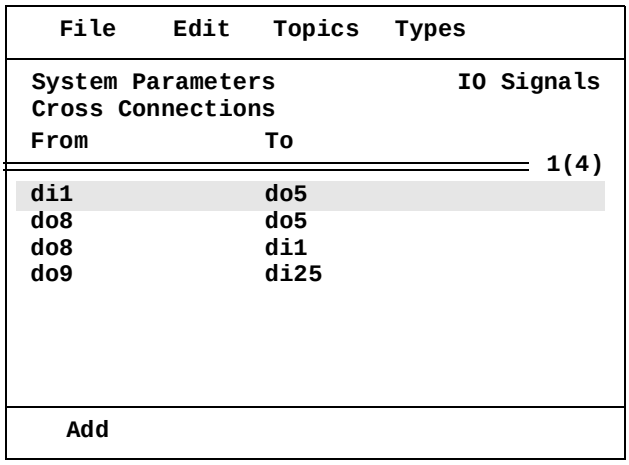


Figure 9 An output signal can be logically connected to an input signal.

- Select the cross connection to be changed and press Enter ↵, or add a new one by pressing ***Add***.
- Define the ”activation (From)” signal and the corresponding ”cross connected (To)” signal.
- Press ***OK*** to confirm.

A maximum of 40 signals can be cross connected. Make sure that the same signal is not connected on both the ”From” and ”To” sides, in the same chain.

3.5 Cross connections with logical conditions

The digital I/O signals can have the logical conditions AND or OR, to set up a condition for a cross connection. These conditions cannot be entered from the teach pendant. They are instead set up in the configuration file EIO.SYS in the cross-connection section (starting with the line “EIO_CROSS:”) using a standard PC. The same rules apply to the logical condition connections for the result signals as for the normal cross-connected result signals. The actors in the cross-connection section have the logical condition operators.

The logical condition operators are:

- AND, syntax in configuration file = “&”
- OR, syntax in configuration file = “!”

For each logical condition connection there can only be one kind of logical operator. Each logical condition connection can be seen as a logical operator box.

The AND operator has the following function:

- If **all** in signals (actor signals) to the AND box are high, the result signals will be high.

The OR operator has the following function:

- If **any** in signals (actor signals) to the OR box are high, the result signals will be high.

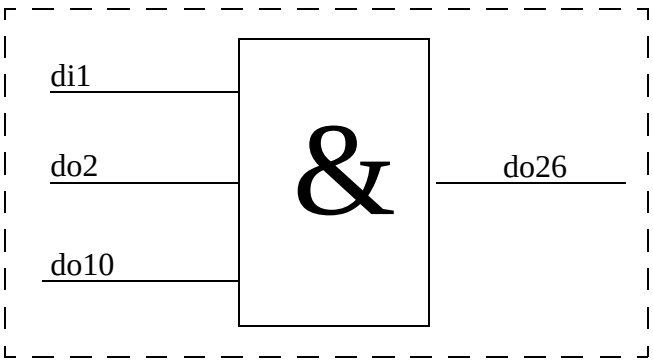
There is one help operator:

- INV, syntax in configuration file = “*” , inverted.

The INV help operator can be connected before an in signal (actor signal) to an AND or OR box which means that the signal is inverted before being checked in the operator box.

Examples

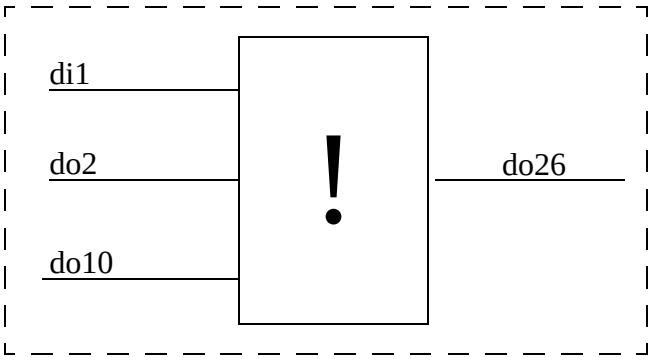
Logical Box AND



In Configuration file:

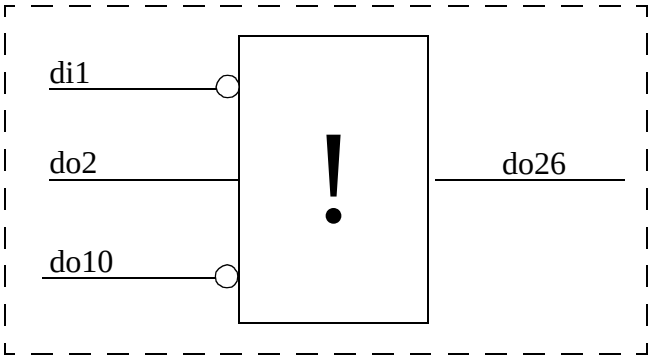
EIO_CROSS:
-Lres do26 -Lact di1 & do2 & do10

Logical Box OR



In Configuration file:

EIO_CROSS:
-Lres do26 -Lact di1 ! do2 ! do10

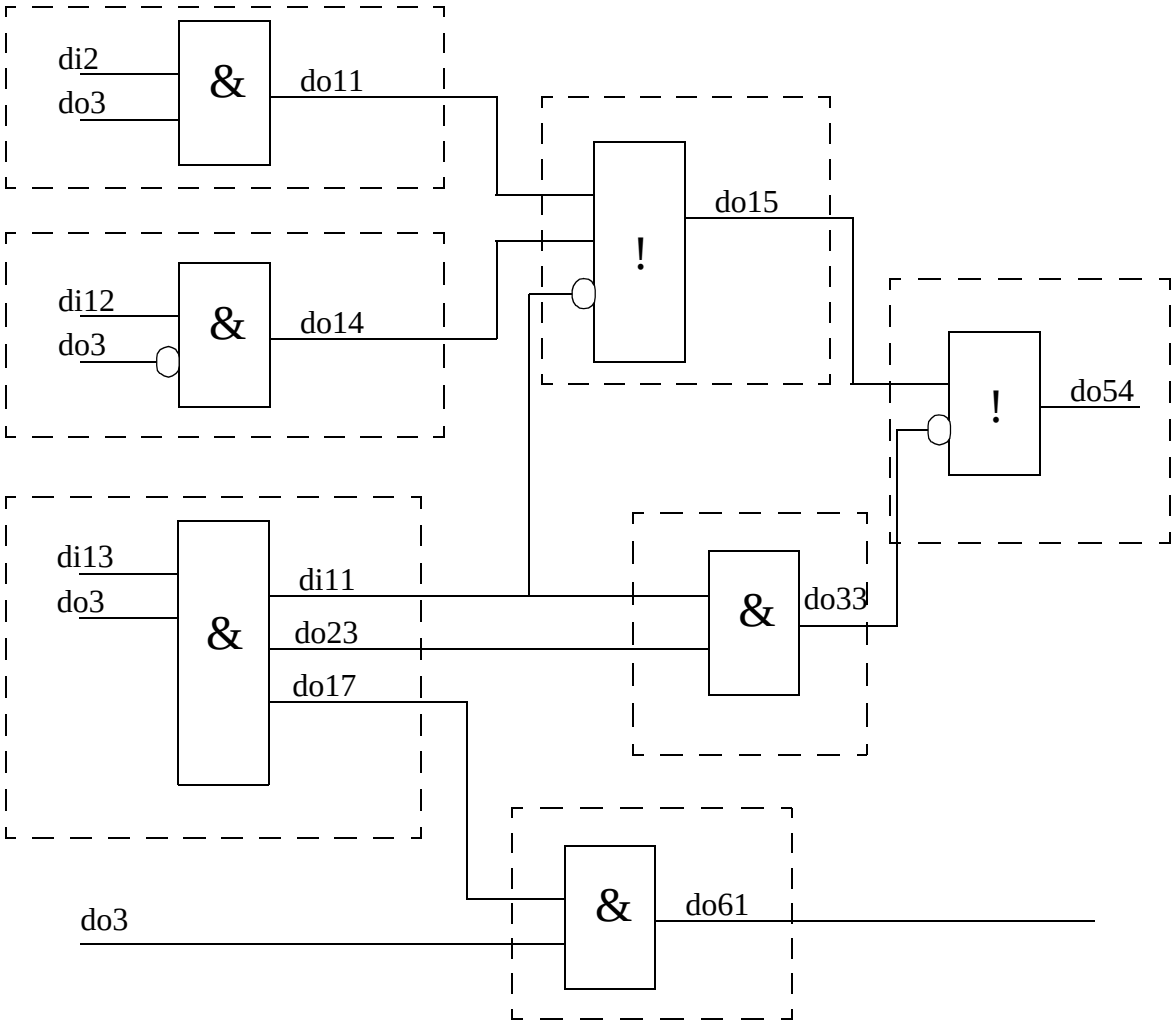


○ = INV = Invert signal

In Configuration file

EIO_CROSS:
-Lres do26 -Lact *di1 ! do2 ! *do10

The actor signals can be both digital In and Out signals. There can be 5 actor signals in each condition but there can be several conditions. The cross-connected signals cannot use Delay or Pulse or any parameters, only “clean” set and reset of digital in and out signals. The example below describes a configuration file that has several logical condition connections



In Configuration file

EIO_CROSS:
-Lres do11 -Lact di2 & do3
-Lres do14 -Lact di12 & *do3
-Lres di11 + do23 + do17 -Lact di13 & do3
-Lres do15 -Lact do11 ! do14 ! *di11
-Lres do33 -Lact di11 & do23
-Lres do61 -Lact do17 & do3
-Lres do54 -Lact do15 ! *do33

3.6 Defining system inputs

The input signals can be assigned specific system actions. In this case, they will automatically be handled by the robot. See also the Product Manual - Installation and Commissioning - *PLC Communication*. They are normally allowed in automatic mode only.

- If the signal has not already been defined, define its name in the normal way. See *Defining input and output signals* on page 11.
- Choose **Types: System Inputs**.

All defined system inputs will be displayed (see Figure 10).

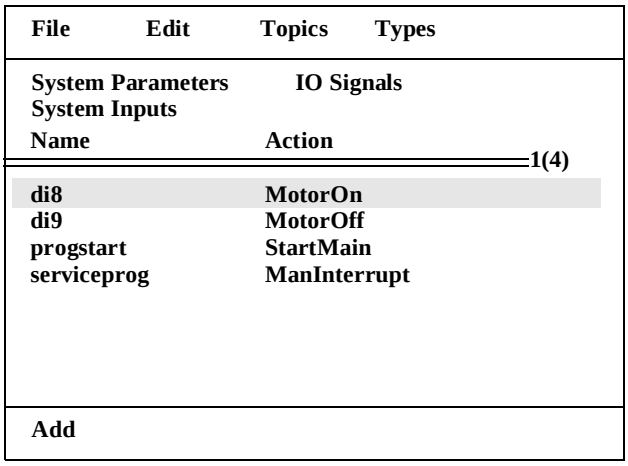


Figure 10 Input signals can be assigned specific system actions.

- Select the system input to be changed and press Enter ↵, or add a new one by pressing ***Add***.
- Define the name of the signal and the system action that is assigned to it. To add the system actions, *Interrupt*, *LoadStart* and *Sync ExtAx*, define their arguments as well.
- Press ***OK*** to confirm.

The following system actions are available:

System action	Description
MotorOn	Sets the robot to the MOTORS ON state.
MotOnStart	Sets the robot to the MOTORS ON state and starts the program (continuous execution) from the current instruction, i.e. from the program pointer.
MotorOff[1]	Sets the robot to the MOTORS OFF state. If a program is executing, it will be stopped before changing state. The robot cannot be set to the MOTOR ON state when this signal is high.
Start	Starts the program (continuous execution) from the current instruction, i.e. from the program pointer.
StartMain	Starts the program (continuous execution) from the beginning. Not valid during program execution.
Stop[1]	Stops program execution. A program cannot be started when this signal is high.
StopInstr[1]	Stops program execution after the current instruction has been finished. A program cannot be started when this signal is high.
StopCycle[1]	Stops program execution when the complete program has been executed, i.e. when the last instruction in the main routine has been executed. A program cannot be started when this signal is high.
SysReset	Performs a system reset (restart), i.e. similar to power off/on.
Interrupt	Executes a routine (procedure) without changing the start pointer. Not valid during program execution. When the routine has been executed, the normal program can be restarted. The name of the routine to be executed is also defined in this dialog, e.g. routine1. This signal, for example, can be used to go to a service position. When the normal program is started again, the robot will move to the point where it was stopped and continue from there.
ResetError	Resets the system output signal Error.
SyncExtAx	Synchronizes an external mechanical unit. The unit to be synchronized is also defined in this dialog e.g. orbit1. One signal is required for each unit.
LoadStart	Loads a program from diskette or other mass storage device. The program is then started from the beginning. The program file name (including mass memory unit) to be loaded is also defined in this dialog, e.g. flp1: PROGRAM1.PRG. Not valid during program execution.
ResetEstop	Resets the emergency stop. The robot can then be set to the MOTORS ON state.

Several signals can be assigned the same system actions.

1. Allowed in both manual and automatic mode.

To delete a system action

- Select the signal to be deleted.
- Press ⌫

The system action assigned to this signal is then deleted, but the signal itself remains defined.

3.7 Defining system outputs

The output signals can be assigned a specific system status. In this case, they will automatically be handled by the robot.

- If the signal has not already been defined, define its name in the normal way. See *Defining input and output signals* on page 11.
- Choose **Types: System Outputs**.

All defined system outputs will be displayed (see Figure 11).

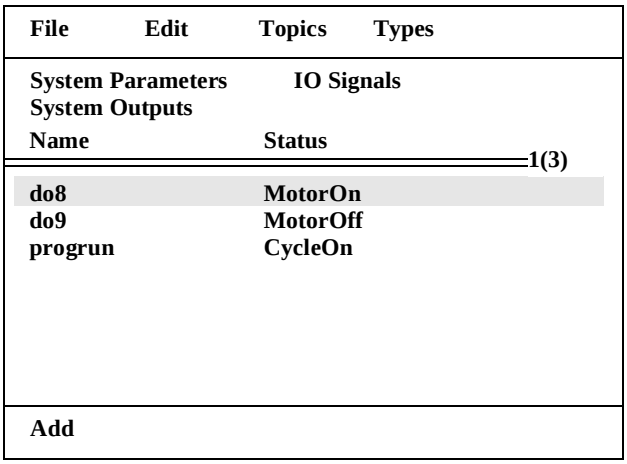


Figure 11 Output signals can be assigned specific types of system status

- Select the system output to be changed and press Enter ↵, or add a new one by pressing ***Add***.
- Define the name of the signal and the system action assigned to it.
- Press ***OK*** to confirm.

The following types of system status are available:

System status	Description
MotorOn	The robot is in the MOTORS ON state. If the robot system is not synchronized, the output will start flashing.
MotOnState	The robot is in the MOTORS ON state. The output is stable, i.e. no flashing.
MotorOff	The robot is in the MOTORS OFF state. If the safety chain is broken, the output will start flashing.
MotOffState	The robot is in the MOTORS OFF state. The output is stable, i.e. no flashing.
CycleOn	A program is executing.
EmStop	The robot is in the Emergency Stop state. The emergency stop must be reset before the robot can be set to the MOTORS ON state.
AutoOn	The robot is in automatic mode.
RunchOk	The run chain is not broken.
TCPSpeed	An analog signal that describes the speed of the TCP. The logical value of the signal is specified in m/s, e.g. a speed of 2000 mm/s corresponds to the logical value 2. The scaling factor for the physical value is specified in the system parameters of the corresponding signal.
Error	The robot execution has been stopped due to an error. (If an error occurs when a program is not executing, this output will not be set.)
PFError	A power failure error has occurred. The program would not restart after this type of error. The program can usually be started, but it will start from the beginning.

Several signals can be assigned the same system status.

To delete a system status

• Select the signal to be deleted.

• Press ⌫.

The system status assigned to this signal is then deleted but the signal itself remains defined.

4 Topic: Communication

The following parameters are found under the Communication topic:

- Specification of physical channels.
- Transmission protocols.
- Application protocols.

• Choose **Topics: Communication**

4.1 Defining physical channels

• Choose **Topics: Communication**

• Choose **Types: Physical Channels**

All defined physical channels will be displayed, as shown in Figure 12.

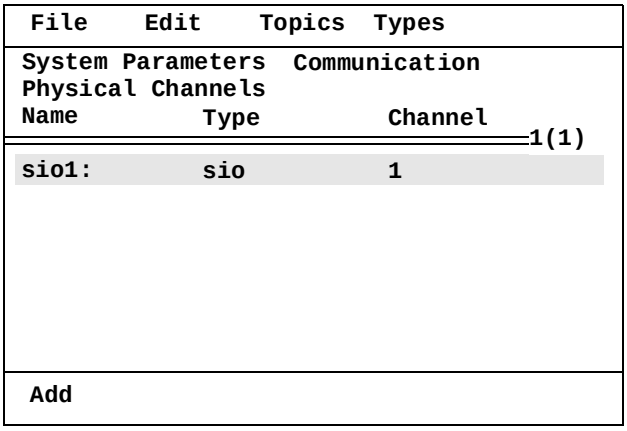


Figure 12 System parameters of the type Serial Channels.

• Select the physical channel to be changed and press Enter ↵, or add a new one by pressing ***Add***.

• Select the desired parameter and change its value. Press ***OK*** to confirm.

Parameters	Description
Name	Name of physical channel (max. 16 characters)
Type	Type of physical channel (sio)
Channel	Channel number (1 - 4).
Baud rate	Baud rate for the serial channel (300 - 19200).
Parity	Type of parity for serial channel. (Odd, Even, None).
No / of bits	Number of data bits (7,8).
No / of stops bits	Number of stop bits (1, 2).
RTS / CTS Control	RTS / CTS flow control when sending from the robot (ON/ OFF). If channel 1 is set to ON, DCD and DTR must be strapped. RTS / CTS must be set to OFF for channel 4.

4.2 Defining Transmission Protocol

- Choose **Topics: Communication**
- Choose **Types: Transmission Protocol**

All defined transmission protocols will be displayed, as shown in Figure 13.

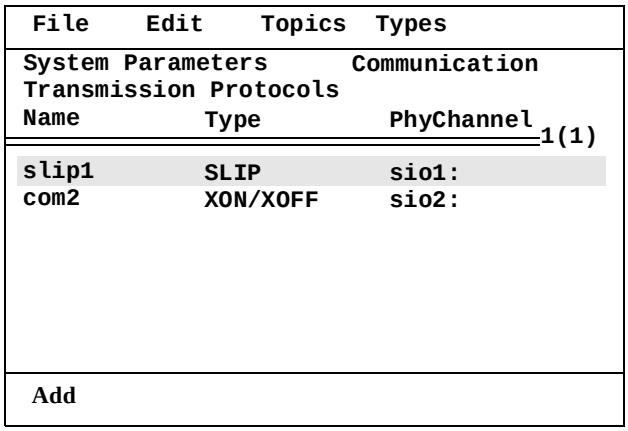


Figure 13 System parameters of the type Transmission protocol.

- Select the transmission protocol to be changed and press Enter ↵, or add a new one by pressing ***Add***.
- Select the desired parameter and change its value. Press ***OK*** to confirm.

Parameters	Description
Name	Name of the transmission protocol (max. 16 characters). The name must be unique and must not be used anywhere else.
Type	Type of transmission protocol (None, XON/XOFF, SLIP).
PhyChannel	Name of the physical channel the protocol should use.
For Slip only:	
Local Address (SLIP)	Local address of the SLIP connection.
Remote Address (SLIP)	Remote address of the SLIP connection.
PortNo	The TCP protocol port number of the remote computer.

4.3 Defining Application Protocol

- Choose **Topics: Communication**
- Choose **Types: Application Protocol**

All defined application protocols will be displayed, as shown in Figure 14.

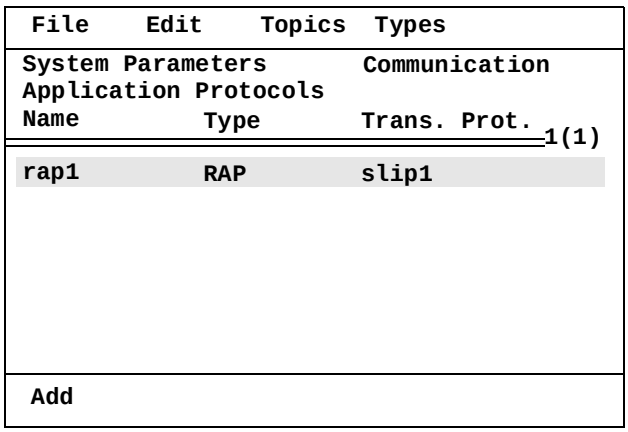


Figure 14 System parameters of the type Application protocol.

- Select the application protocol to be changed and press Enter ↵, or add a new one by pressing ***Add***.
- Select the desired parameter and change its value. Press ***OK*** to confirm.

Parameters	Description
Name	Name of the application protocol (max. 16 characters). The name must be unique and must not be used anywhere else.
Type	Type of application protocol (RAP).
Trans Prot.	Name of the transmission protocol the protocol should use.
For RAP only:	
Enable SysEvent	See RAP Service Specification, Start-up Log Event. (TRUE/FALSE)

5 Topic: Controller

The following parameters are found under the Controller topic:

- Activation of Hold-To-Run Control
- Event routines
- Maximum regain distances
- System miscellaneous
- Automatic loading of modules and programs
- Tasks (option Multitasking is required)

• Choose **Topics: Controller**.

5.1 Activate Hold-To-Run Control

When using the Hold-To-Run control, the program start key must be held down all the time, in order to execute a program.
This function is always activated in the manual operating mode at full speed, but can also be activated at reduced speed.

• Choose **Types: Safety**.

• Select the safety function to be changed and press Enter ↵, or add a new one by pressing ***Add***.

• Define the function and whether or not it shall be active ***(True)***.

• Press ***OK*** to confirm.

5.2 Defining event routines

Special system events, such as program stop, can be connected together with an ordinary routine. When the event occurs, the corresponding routine is executed automatically.

• Choose **Types: Event Routines.**

All defined event routines will be displayed (see Figure 15).

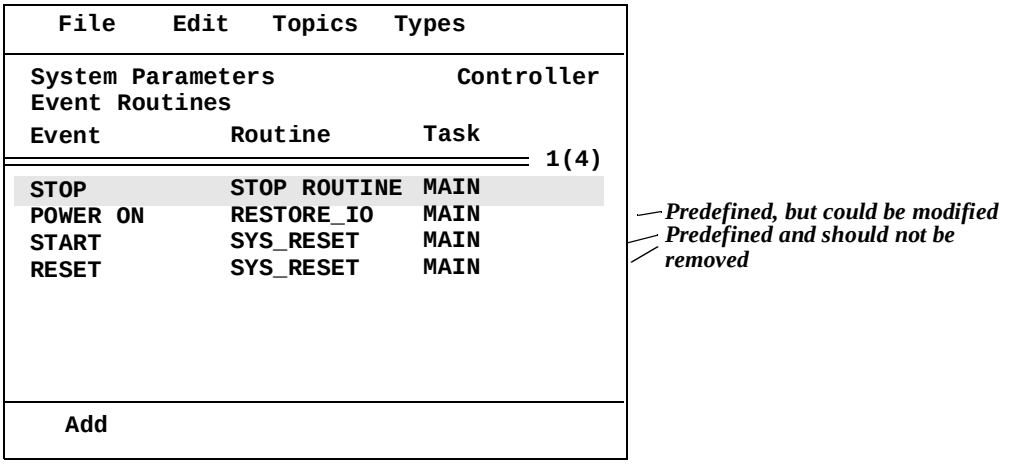


Figure 15 Certain events can start routines automatically.

- Select the event routine to be changed and press Enter ↵, or add a new one by pressing ***Add***.
- Define the system event and the routine assigned to it, also select which task the definition is for.
- Press ***OK*** to confirm.

The following types of system events are available:

System event	Description
POWER ON	The robot is started.
START	Execution is started from the beginning of the program.
RESTART	Execution is started from the position where is was stopped.
STOP	The program was stopped. **Note:** A delayed stop after current cycle will not execute the routines connected to this state.
QSTOP	The robot was quick stopped (emergency stop).
RESET	The old program was erased.

The specified routine must be a procedure without any parameters. The routines should be in a system module, at least the routines for the RESET event.

If the robot cannot find the specified routine, no error message will be given.

Avoid motion instructions in the routines. For STOP/QSTOP, a motion instruction in the corresponding event routine will result in an error. It is advisable to keep the routines short and quick.

A maximum of four routines may be specified for each system event and each task (multitasking). The same routine can be used in more than one event.

The task(s) available are dependent on the type **Tasks**.

5.3 Specifying regain distances

Maximum distance for a regain movement (the distance from the current robot position to the last executed path). This can be set both for start in manual mode and for start in automatic mode.

A regain movement will begin when program start is ordered and before the program continues with the instruction that was interrupted due to a stop request. If the regain distance exceeds the specified max. distance, an error message will occur.

• Choose **Types: Regain distances.**

The operating modes will be listed, (see Figure 16).

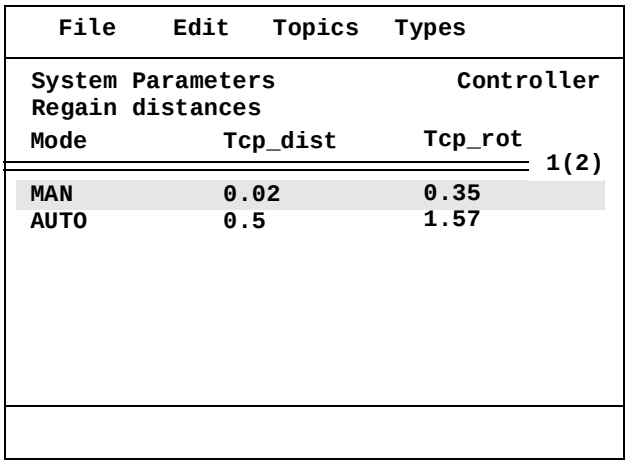


Figure 16 Maximum regain distances.

• Select the operating mode to be changed and press Enter ↵ .

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Mode	AUTO or MAN.
Tcp_dist	The maximum TCP distance (m).
Tcp_rot	The maximum TCP rotation (rad).
Ext_dist	The maximum distance for external axes (m).
Ext_rot	The maximum rotation for external axes (rad).

5.4 System miscellaneous

Changes to any item in this menu will force the system to restart the program handling part of the system at the next warm start. All user programs will be erased and all task modules specified in the configuration will be reloaded.

• Choose **Types: System misc.**

All functions already added will be listed, (see Figure 17).

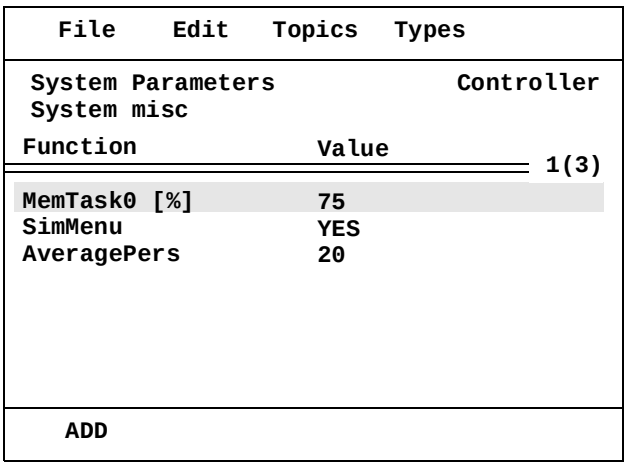


Figure 17 System miscellaneous.

• Mark the function to be changed and change it, or add a new one.

• Press ***OK*** to confirm.

Function	Description
MemTask0	Only valid if the robot is equipped with option Multitasking. If the parameter **size** (**Types: Tasks**) not is set for the main task (program 0), this function specifies the amount of the total memory that the main task should occupy. The rest will be shared by the other tasks. If it is a single task system the amount will always be 100%**.**
SimMenu	The WaitTime, WaitUntil and WaitDI instructions will generate an alert box in manual mode to make it possible for the user to simulate the instruction and continue to execute the next instruction. If this is set to **NO**, no menu will be generated. **YES** is the default behaviour.
AveragePers	Average size in bytes of one PERSISTENT variable. This setting will affect the maximum number of persistents in the system.

5.5 Automatic loading of modules and programs

System modules and/or normal RAPID programs can be loaded automatically when the robot is powered on (restarted).

Changes to any item in this menu will force the system to restart the program handling part of the system at the next warm start. All user programs will be erased, and all task modules specified in the configuration will be reloaded.

• Choose **Types: Task modules.**

A list of the files which will be preloaded will be shown, (see Figure 18).

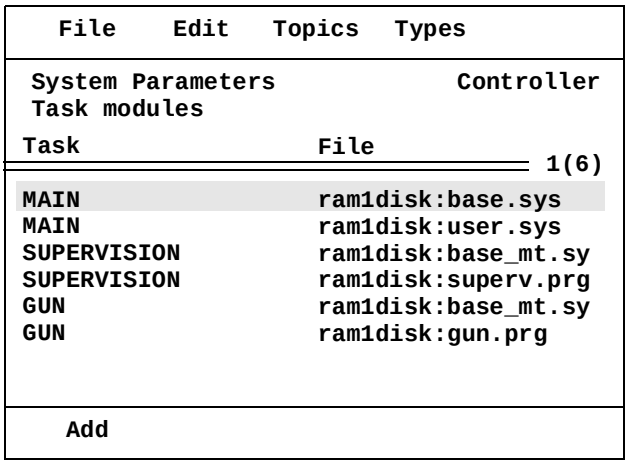


Figure 18 Programs loaded into the system in the warm start sequence.

• Select the item to be changed and press Enter ↵, or add a new one by pressing ***Add***.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
File	A path to the program file. (**Note:** The file must be reachable in every warm start, e.g. ram1disk:base.sys)
Task	The symbolic name of the task to which the module should be loaded. The available task(s) are those under the type **Tasks**. (See *Defining multitasking* on page 31).
Storage	**Built in** or **loaded**. A built in module is not visible, it will not occur in the list of modules and cannot be removed from the program window. Loaded is the default behaviour.
TextResource	If Storage is set to **Built in** it is possible to use a national language for routine names, for example. This parameter should be 0 as English is used for the RAPID language.

The files “ram1disk:base.sys” and “ram1disk:user.sys” are predefined and should not be removed, but the contents of “user.sys” may be modified.

The file “ram1disk:base_mt.sys” should always be defined for any additional tasks.

5.6 Defining multitasking

Available when the option *Multitasking* is installed. The various tasks are defined with name, priority and execution behaviour.

Changes to any item in this menu will force the system to restart the program handling part of the system at the next warm start. All user programs will be erased and all task modules specified in the configuration will be reloaded.

• Choose **Types: Tasks.**

All specified tasks will be listed, (see Figure 19).

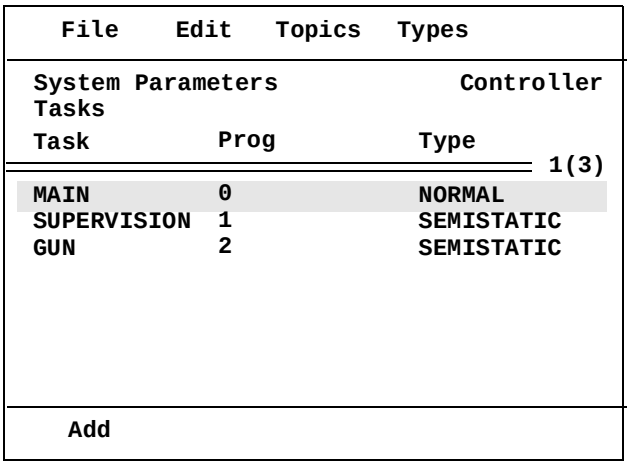


Figure 19 All available tasks.

• Select the task to be changed and press Enter ↵, or add a new one by pressing ***Add***.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Task	The name of a task. (max 16 characters)
Prog	The program number. Program no. 0 is reserved for the normal robot program which is the only one that may include motion instructions.
Type	Controls the start/stop and system restart behaviour. **NORMAL** - The task will react on START/STOP requests given from the teach pendant or other sources. **STATIC** - The task will be started from the beginning at the first warm start after a cold start, and automatically restarted at the current position at all other warm starts. **SEMISTATIC**- The task will be restarted from the beginning

at all warm starts.
Program no. 0 must be of type NORMAL. The other tasks should be of type STATIC or SEMISTATIC.

Task in foreground The name (or program number) of a task that should run in foreground the of this. If “-1” or an empty string ““ is set for this parameter, it will run at the highest level with no other task that could suspend its execution.

Main entry The name of the start routine. It should be a RAPID routine without any parameters and reachable in this task (only valid for STATIC and SEMISTATIC tasks).

BindRef This parameter should be set to **NO** if the system is to accept unsolved references in the program while linking a module, or otherwise set to **YES** (default value is **YES**). The parameter must be set to **NO** if the instructions **Load/Erase** are to be used.

There will be a runtime error on execution of an unresolved reference.

Size If specified, the system will allocate the specified amount of memory for this task. If not specified (the default setting) the system will give this task a memory with a size depending on how many other tasks there are in the system. Do not specify a size less than 120 000 bytes unless a reduced set of instructions is used in this task. The only tasks with a reduced set of instructions are those included in an application package delivered from ABB. In these cases, this size is already preset.

If a task is specified as a STATIC or SEMISTATIC type, all modules must be preloaded. See *Automatic loading of modules and programs* on page 30.

6 Topic: TeachPendant

The following parameters are found under the TeachPendant topic:

- Optional packages
- Defining customised file extensions
- Authorising and confirmation of user commands, changing Pass Codes.
- Activation of limited modpos function.

• Choose **Topics: Teach Pendant**.

The Most Common instruction pick lists and I/O list are also stored when saving this topic.

6.1 Defining Optional Packages

If several process packages (ArcWare, SpotWare etc.) have been added to the system, it is possible to define which package is to be used for the Program window and the Production window.

• Choose **Topics: TeachPendant**.

• Choose **Types: Optional Package.**

• Press Enter [↵].

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Use for Program	The name of the process package to be used for Program, or NONE if not used.
Use for Production	The name of the process package to be used for Production, or NONE if not used. (No process package available in this version).

6.2 Defining File Extension

It is possible to add file extensions for RAPID created files, so that they are recognised by any file dialogue.

• Choose **Topics: TeachPendant**.

• Choose **Types: File Extensions.**

• Select the File extension to be changed and press Enter ↵ or add a new one by pressing ***Add***.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Name	The name of the extension (max. 3 characters)
Description	Explains the type of file

6.3 Defining authorisation and confirmation

It is possible to limit the access to certain commands, by using user levels and associated pass codes. This means that a function will not be executed unless you have the correct user level. It is also possible to define that a command will not be performed until it is confirmed.

In the robot there are four (4) user levels:

Operator for functions accessible to all users.
No pass code needed.

Service for functions associated with service.
Pass code needed.

Programmer for functions related to programming and testing.
Pass code needed.

Service & ***Programmer***
For functions needed for both programming and service.
Pass code needed for either Service or Programmer.

A pass code can be up to 8 digits long.

If you try to perform a command and you do not have the correct user level, a dialogue will appear, as shown in Figure 20:

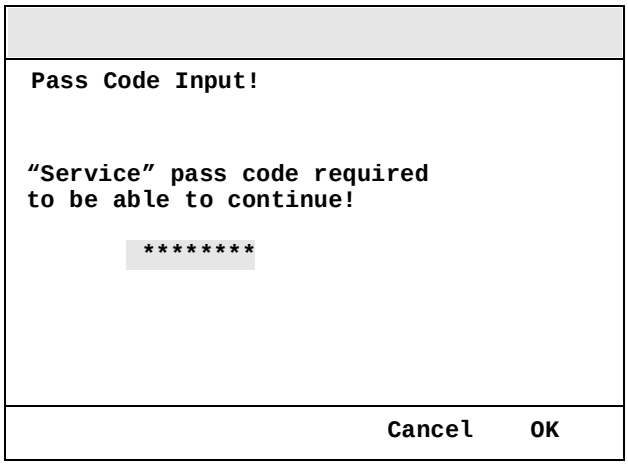


Figure 20 Pass Code Input Dialogue.

• Input your pass code for the correct user level.

• Press ***OK*** to confirm the pass code input.

If the pass code is still not correct, press ***Cancel*** and ask your system administrator for the correct one.

Defining new pass codes

• Choose **Topics: TeachPendant**.

• Choose **Edit: Change Pass Codes**.

• Read the warning message and press ***OK***.

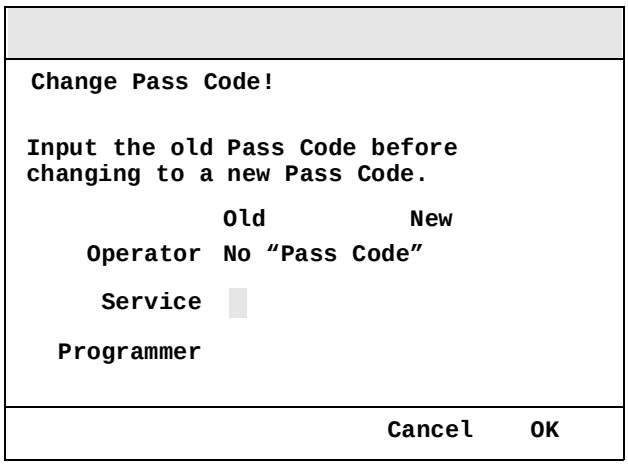


Figure 21 Pass CodeChange Dialogue

• Select the old pass code of the user level to be changed (use the arrow keys Up or Down)

• Input the old pass code (the pass code will not be visible). After installation of the control program, the pass code is *007*.

• Select the new pass code of the user level to be changed.

• Input the new pass code, (the pass code will be visible).

• Press ***OK*** to acknowledge the change of pass code.

• Press Enter ↵ to confirm the updating.

Defining authorisation

To authorise a function:

• Choose **Topics: TeachPendant**.

• Choose from the **Types** menu, the window you want to authorise (names start with **Authorise...**).

All functions that can be authorised will be displayed (e.g. as shown in Figure 22).

• Select the function to change and press Enter ↵.

Parameter	Description
Function	The name of the function to be authorised (cannot be changed).
User Level	Required user level to be able to execute the function, (can be Operator, Service and Programmer).
Confirm	Should the function be confirmed before it is executed? Yes or No.

• To change

- User Level, select parameter ***User Level*** and press Enter ↵.
- Confirm, select parameter ***Confirm*** and press Enter ↵.

• Choose appropriate value and press ***OK***.

• When finished, press ***OK*** to confirm the change.

Authorise SystemParameters

• Choose **Topics: TeachPendant**.

• Choose **Types: Authorise SystemParameters**.

All possible functions will be displayed, as shown in Figure 22.

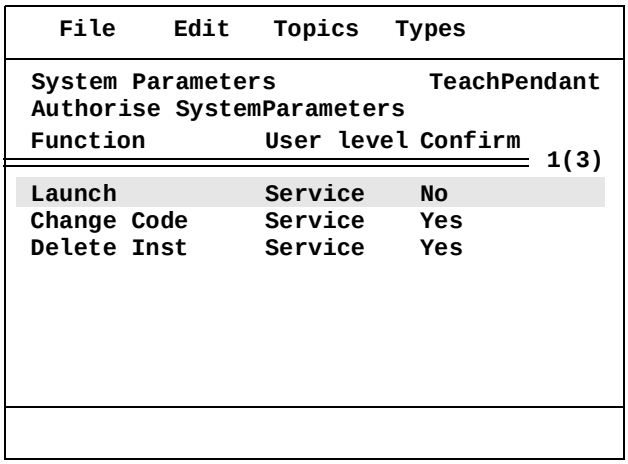


Figure 22 Authorise System Parameters.

Function	Description
Launch	To authorise the opening of the window.
Change Code	To authorise the change of pass codes.
Delete Inst	To authorise the deletion of a parameter.

• To change user level and/or confirm, see *Defining authorisation* on page 38.

Authorise Program

• Choose **Topics: TeachPendant**.

• Choose **Types: Authorise Program.**

All possible functions will be displayed, as shown in Figure 23.

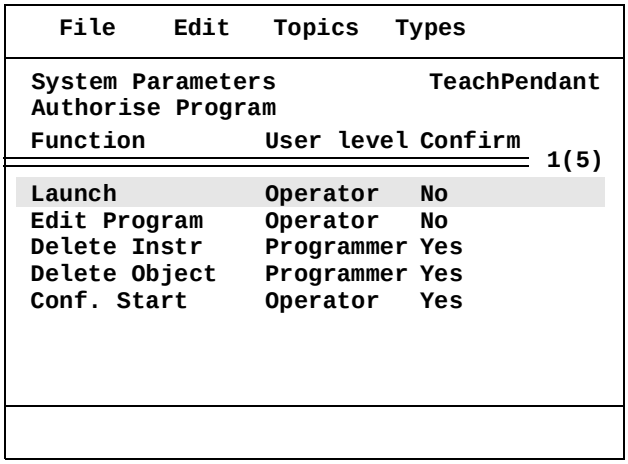


Figure 23 Authorise Program.

Function	Description
Launch	To authorise the opening of the window.
Edit Program	To authorise changing of the program.
Delete Instr	To authorise the deletion of any instruction in a RAPID routine.
Delete Object	To authorise the deletion of any RAPID objects (e.g. routines, modules or data).
Conf. Start	Only confirmation. If set to ***No*** the program execution will always start from the program pointer (PP).

• To change user level and/or confirm, see *Defining authorisation* on page 38.

6.4 Activation of Limited ModPos Function

If the Limit ModPos function is active, only a limited deviation from the original position is allowed, when the ***ModPos*** key is pressed to modify a position. The limited deviation concerns both the linear distance and the orientation.

• Choose **Topics: TeachPendant**.

• Choose **Types: Modify Position**.

Now the current type of ModPos function will be displayed, ***ModPos*** or ***LModPos***.

• Press Enter [↵].

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Type	The current type of modpos. *ModPos* means that Limit ModPos is deactivated, i.e. any change is accepted. *LModPos* means that Limit ModPos is activated, i.e. the change must be within a limited area.
Tuning In Auto	Tuning Off or On in auto. *On* = tuning functionality enabled in automatic mode. *Off* = tuning functionality disabled in automatic mode.
Mode	The current mode of limited modpos. *Abs* = The limited area is around a fixed original point. *Rel* = The limited area is around the current point and will be moved when you modify the point.
Max Trans	The maximum allowed deviation in mm from the current or original position.
Max Rot	The maximum allowed reorientation in degrees from the current or original position.
Max External Trans	The maximum allowed deviation in mm from the current or original position concerning external linear axes.

Max External Rot	The maximum allowed deviation in degrees from the current or original position concerning external rotational axes.
If Auto	Parameter for automatic activation of Limit ModPos when the operator's key is switched to Auto Mode. *LModPos* means that Limit ModPos is activated when the operator's key is switched to Auto Mode. *As Is* means that ModPos is not changed.

6.5 Programmable keys

On Version 2 of the Teach pendant, see Figure 24, you can define five keys for the purpose of setting outputs and generating signal events.

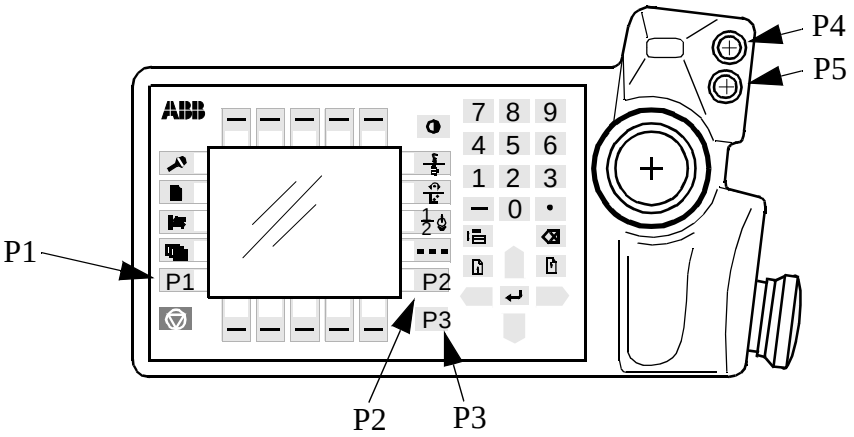


Figure 24 There are five programmable keys on the teach pendant.

• Choose**Topics: Teach Pendant**.

• Choose**Types: Programmable Keys**.

Now the definition of the keys will be displayed.

• Select the key to be defined and press Enter ↵.

Parameter	Description
Key	The designation of the key P1-P5
Type	Type of key: **Input**, **Output** or **None** (not activated)
Connection	Name of signal to be chosen.

When Type is selected as Output, the following are also available:

Key Pressed	Defines how the output should be set. Toggle: if the signal value is high (1), it will become low (0) and vice versa. Pulse: a positive pulse is generated. Set1/Set0: either to high (1) or to low (0). Press/Release: the signal will be high (1) as long as the key is depressed. When the key is released, the signal will change to low (0).

When Type is selected as Input, all events will wait for this input to be activated. These events can be:

- System inputs. The input must then be associated with a system activity, see *Defining input and output signals* on page 11.
- Interruptions are defined by the instruction ISignalDI (see RAPID Reference Manual).
- Waiting for input via the instruction WaitDI (but not WaitUntil). (See RAPID Reference Manual).

A physical signal can be connected to the output. However, its value will be affected.

7 Topic: Manipulator

The Manipulator topic contains parameters associated with motion control in the robot and external axes, e.g.:

- The commutation offset
- The calibration offset
- The working space limits.

• Choose **Topics: Manipulator**.

Do not change "Transm gear ratio" or other kinematic parameters from the teach pendant or a PC. This will affect the safety function *Reduced speed 250 mm/s*.

7.1 Defining the commutation offset and calibration offset of the motors

These values are generally updated from the Service window. If, however, they are known, they can be specified in the System Parameters window.

• Choose **Topics: Manipulator**.

• Choose **Types: Motor**.

• Select the desired motor and press Enter ↵.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Name	The name of the motor, e.g. motor_1 (max. 16 characters).
Cal offset	The position of the motor (resolver) when it is in the calibration position (in radians).
Cal offset valid	Specifies if the calibration offset is defined.
Com offset	The position of the motor (resolver) when the rotor is in the electrical zero position relative to the stator (in radians).
Com offset valid	Specifies if the commutation offset is defined.

7.2 Defining the range of movement and calibration position of each axis

• Choose **Topics: Manipulator**.

• Choose **Types: Arm**.

• Select the desired arm (axis) and press Enter ↵.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Name	The name of the arm, e.g. arm_1 (max. 16 characters).
Upper joint bound	The upper joint limit, e.g. +3.139 for axis 1 (in radians). 180^{o} = 3.1416 radians.
Lower joint bound	The lower joint limit, e.g. -3.139 for axis 1 (in radians).
Use check point	The name of a check point (if any). See *Defining arm check point* on page 46.
Use customer arm load	The name of an arm load (if any). See *Defining arm load* on page 45.
Cal pos	The position of the axis when it was calibrated. If this value is changed, the robot must subsequently be fine-calibrated in the Service window. See the Product Manual.

7.3 Defining supervision level

It is possible to change the default supervision levels if a system needs to be more or less tolerant to external disturbances.

Increasing the tune factors can reduce the lifetime of the robot.

- Choose **Topics: Manipulator**.
- Choose **Types: Arm**
- Select the desired arm (axis) and press Enter ↵.
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
Jam supervision trim factor	Tune factor for jam supervision. The factor influences maximum allowed time at zero speed with maximum torque.
Load supervision trim factor	Tune factor for load supervision. The factor influences maximum allowed time at non-zero speed with maximum torque.
Speed supervision trim factor	Tune factor for speed supervision. The factor influences the maximum allowed speed error.
Position supervision trim factor	Tune factor for position supervisions. The factor influences the maximum allowed position error.

7.4 Defining sync speed

If the default values for sync speed are not satisfactory they can be changed.

- Choose **Topics: Manipulator**.
- Choose **Types: Arm**.
- Select the desired arm (axis) and press Enter ↵ .
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
Sync speed high	High sync speed. Use (rad/s) or (m/s) depending on whether it is a rotating or linear motion.
Sync speed low	Low sync speed. Use (rad/s) or (m/s) depending on whether it is a rotating or linear motion. This speed must be sufficiently low so that the motor stops within one revolution after leaving the sync switch.

7.5 Defining teach mode speed

When there is a requirement to monitor manual mode with reduced speed lower than 250 mm/s, this can be achieved by changing the teach mode maximum speed.

- Choose **Topics: Manipulator**.
- Choose **Types: Motion system**
- Select the desired system and press Enter ↵ .
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
Teach mode max speed	Maximum allowed speed in manual mode with reduced speed.

7.6 Defining arm load

The performance of the robot will be negatively affected if the arm load is not defined. When more than one load is mounted on one and the same arm, the total weight and the position of the centre of gravity must be calculated.

All loads mounted on the upper arm are related to axis 3, including loads on the rotating part.

• Choose **Topics: Manipulator**.

• Choose **Types: Arm load**.

Parameter	Description
Name	The name of the arm load, e.g. armload_1 (max. 16 characters).
Mass	The mass of the arm load (in kg).
Mass cent x ***Mass cent y*** ***Mass cent z***	The mass centre, specified based on the coordinate system of the arm. See the example in Figure 25.

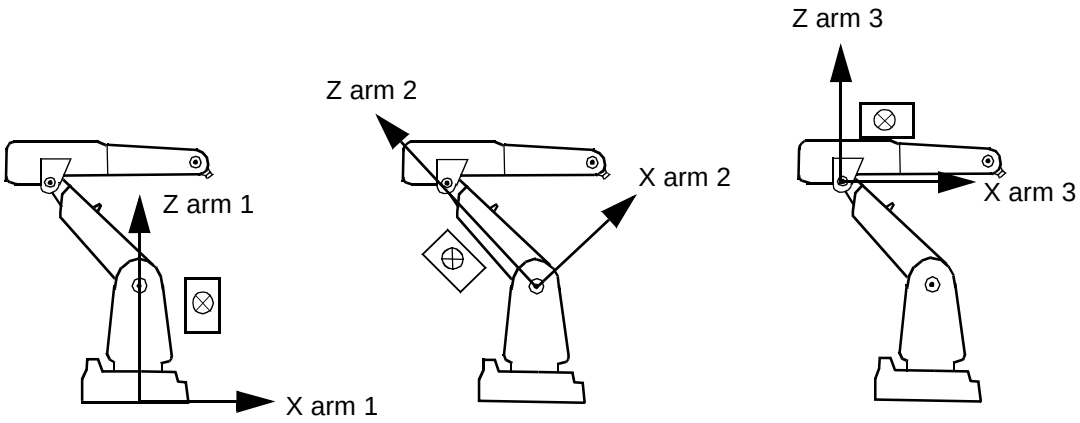


Figure 25 The arm coordinate system for axes 1, 2 and 3.

Now, this arm load must be connected to the current arm (axis):

• Choose **Types: Arm**.

• Select the desired arm and press Enter ↵.

• Select the parameter ***Use customer arm load*** and specify the name of the arm load previously defined.

• Press ***OK*** to confirm.

7.7 Defining arm check point

If an extra load, such as a transformer or a welding-bar roller, is attached to an arm, then a point on this equipment can be defined. In this case, the robot monitors the speed of that point so that it does not exceed 250 mm/s in the manual operating mode (reduced speed).

• Choose **Topics: Manipulator**.

• Choose **Types: Arm check point**.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Name	The name of the check point, e.g. chk_pnt_1 (max. 16 characters).
Pos x ***Pos y*** ***Pos z***	The position of the check point, specified on the basis of the current coordinate system of the arm (in meters). See the example in Figure 26.

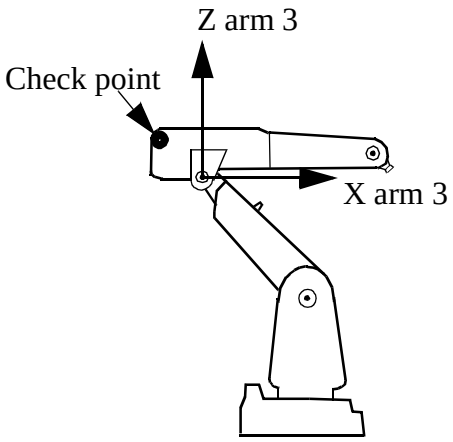


Figure 26 Definition of the check point for arm 3.

Now, this check point must be connected to the current arm (axis):

• Choose **Types: Arm**.

• Select the desired axis and press Enter ↵.

• Select the parameter ***Use check point*** and specify the name of the arm check point previously defined.

• Press ***OK*** to confirm.

7.8 Defining the base coordinate system

Normally the base coordinate system of the robot coincides with the global coordinate system. However, the base coordinate system can be moved relative to the global coordinate system. Please note that the programmed positions are always related to the global coordinate system, and all positions will therefore also be moved, as seen from the robot. Normally this would be defined from the Service window, but if the values are known they can be input from the system parameters.

• Choose **Topics: Manipulator**.

• Choose **Types: Robot**.

• Select the manipulator whose base coordinate system is to be changed.

• Press Enter ↵.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Name	The name of the robot, e.g. robot_1 (max. 16 characters).
Type	Robot type (not to be changed).
Base frame x	The X-coordinate of the base coordinate system's position in relation to the world coordinate system (in metres).
Base frame y	The Y-coordinate of the base coordinate system (in metres).
Base frame z	The Z-coordinate of the base coordinate system (in metres).
Base frame u0-u3	The orientation of the base coordinate system in relation to the world coordinate system (expressed in quaternions q1-q4). Figure 27 illustrates some examples of different values.
Base frame moved by	Specifies if the robot is to be operated in coordination with a track. See *Defining a track motion with coordinated motion* on page 49.

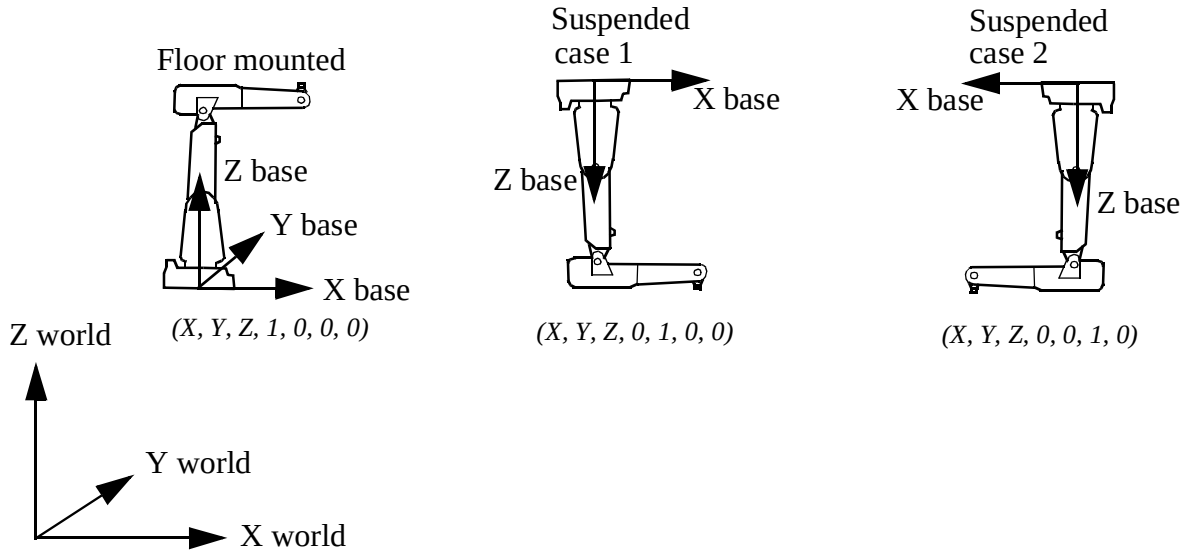


Figure 27 Some examples of definitions of the base coordinate system.

7.9 Defining external manipulators with more than one axis

To achieve the best possible performance from an external manipulator, a set of data, describing its kinematic and dynamic properties (among other things), must be defined. This data cannot be defined in the system parameters, but must be read from a diskette. If no diskette was supplied with the manipulator, the manipulator cannot be coordinated with the robot. It can, however, be defined as a number of separate external axes. See *Defining external axes with external drive units* on page 49 or *Defining external axes with internal drive units* on page 53.

- Read the files that define the manipulator. See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add New parameters**.
- Define the calibration offset, name of the mechanical unit etc. See *Defining external axes with external drive units* on page 49 or *Defining external axes with internal drive units* on page 53.

• Define the base coordinate system as described in *Defining an external mechanical unit coordinated with the robot* on page 49.

7.10 Defining a track motion with coordinated motion

• Define the axis in the usual way. See *Defining external axes with external drive units* on page 49 or *Defining external axes with internal drive units* on page 53. Define the base coordinate system of the track motion, from the Service window, for example. See Chapter 10, Calibration.

• Choose **Types: Robot**.

• Select the robot and press Enter ↵.

• Set the parameter ***Base frame moved by*** to the name of the axis (single) that is used by the defined track.

7.11 Defining an external mechanical unit coordinated with the robot

• Define the axis in the usual way. See *Defining external axes with external drive units* on page 49 or *Defining external axes with internal drive units* on page 53. Define the base coordinate system of the unit from the Service window, for example. See Chapter 10, Calibration.

• Choose **Types: Mechanical unit**.

• Select the mechanical unit to be coordinated with the robot and press Enter ↵.

• Set the parameter ***User frame moved by*** to the name of the axis (*Single*) that rotates the work object.

7.12 Defining external axes with external drive units

The system diskette, *Set-up*, supplied with the robot, contains a number of predefined combinations of external axes. These can be found in the EXTAXIS directory and must be used for the installation.

• When defining external axes, *Service motion parameters* must be used. To activate these, the control program must be installed once again. Select ***Service*** at the question *Service/ Standard motion parameters?*. See Product Manual - Installation and Commissioning.

• Read the files corresponding to the current installation. Use one file for each axis according to the physical channel connection. See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add New parameters**.

File name		Content
/EXTLIN/	ECLIN7	Linear external axis 7 with external drive unit
	.	
	ECLIN12	Rotating external axis 12 with external drive unit
/EXTROT/	ECROT7	Rotating external axis 7 with external drive unit
	.	
	ECROT12	Rotating external axis 12 with external drive unit

A mechanical unit is now created for each axis. The name of this unit is the same as the name of the file, e.g. ECROT7.

- Restart the robot, **File: Restart**.
- Choose **Topics: Manipulator**.
- Choose **Types: Mechanical Unit** and specify the characteristics of the mechanical unit.

Parameter	Description
Name	The name of the unit (max. 7 characters) This name will subsequently be used in the Jogging window and from the program, e.g. when a unit is to be activated.
Stand-by state	The unit waits until the first move instruction or jogging command, before releasing the brakes and controlling the axis.
Activate at start up	The unit is to be activated automatically at start up.
Do not allow deact.	The unit cannot be deactivated.

- Choose **Types: Joint** and specify, under the parameter ***Logical axis***, the logical number of the axis from the RAPID program. For example, ECROT10 is by default defined as logical axis 10. This would correspond to the *eax_d* field of a *robtarget* data.
- Choose **Types: Axc Channel** and verify the measurement type for each channel (axis).
- Choose **Types: Motor** and define the calibration offset as in *Defining the commutation offset and calibration offset of the motors* on page 43. Do not forget to set the parameter ***Cal offset valid*** to *Yes*. The calibration offset can be suitably updated by putting the axis in its calibration position and then fine-calibrating in the Service window. When the synch-switch is in use, the synchronisation should be tested a number of times from different directions. If an erroneous result is sometimes obtained, the synch-switch should be moved just a very short distance.
  It is not necessary to specify the commutation offset for an axis with an external drive unit.
- Restart the robot, **File: Restart**.

Specify whether common drive units are used

When one or more axes use a common drive unit, this fact must be specified.

- Choose **Topics: Manipulator**.
- Choose **Types: Drive System** and specify under the parameter ***Use drive unit***, the name of the drive unit used.

If axis 7 is read, for example, then axis 7 will have a drive unit with the name ECROT7 after that the file ECROT7 has been read. Axis 10 will have a drive unit with the name ECROT10. If these axes share a common drive unit, the name of the drive unit for axis 10, for example, should be changed to ECROT7.

Add activation relays (if connected)

The external drive unit can be activated via signals from the robot. If the unit is activated, for example, by selecting the unit in the Jogging window, the corresponding signal is set automatically. A check is then made that the corresponding output signal from the relay is set.

• Define the input and output signals of the relays. See *Defining input and output signals* on page 11. Restart the robot, **File: Restart**, and check that the external axes can be activated via the Inputs/Outputs window.

• Read a relay configuration file, EXTAXIS/UTIL/RELAY on the Set-up diskette. See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add New parameters**. This will create a new relay with the name RELAY_x.

• Choose **Topics: Manipulator**.

• Choose **Types: Relay** and specify the name of the relay and its signal connections.

Parameter	Description
Name	The name of the relay, e.g. *relay_track* (max. 16 characters). This must be changed if more than one relay is used.
Delay time	The time delay between the activation signal and the electrical relay function, e.g. 0.2 s.
Inverted input	Specifies if an inverted input is to be used in the relay.
In signal	Specifies the logical name of the input signal to the relay. The name must be identical (including upper and lower case letters) to the signal name specified in the I/O configuration.
Out signal	Specifies the logical name of the output signal to the relay. The name must be identical (including upper and lower case letters) to the signal name specified in the I/O configuration.

• Choose **Types: Mechanical Unit** and specify, under the parameter ***Use act relay***, the name of the activating relay.

• When more than one activating relay is used, read in a new relay (file RELAY) and repeat the above procedure.

Add brake relays (if connected)

If the external unit is equipped with a brake, it will be activated automatically when the unit is deactivated or when the robot system assumes the MOTORS OFF state. It will also be activated when the robot has been stationary for a specific time in the MOTORS ON state.

Define the input and output signals of the relays. See *Defining input and output signals* on page 11. Restart the robot, **File: Restart**, and check that the brakes can be activated from the Inputs/Outputs window.

• Read a relay configuration file, EXTAXIS/UTIL/RELAY on the Set-up diskette. See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add New parameters**. This will create a new relay with the name RELAY_x.

• Choose **Topics: Manipulator**.

• Choose **Types: Relay** and specify the name of the relay and its signal connections. See *Add activation relays (if connected)* on page 50.

• Choose **Types: Mechanical Unit** and specify, under the parameter ***Use brake relay***, the name of the brake relay.

• The point in time, at which the regulator shall stop braking the axis and allow the mechanical brakes to work on their own, can be changed by choosing **Types: Brake**.

Parameter	Description
Control off delay time	Time delay (s) before the regulator is turned off, if the speed is < *Control off speed*.
Control off speed	The speed (as a percentage of the max. speed) at which the time delay starts.

• If more than one brake relay is used, read a new relay into the system (file RELAY) and repeat the above procedure.

Define the other parameters for an external axis

• Choose **Types: Arm** and specify the geometrical characteristics of the axis.

Parameter	Description
Transm gear ratio	The transmission ratio of the motor to the axis. Example: For a linear axis, 21.43 means that the axis moves 1 m when the motor rotates 21.43 radians.
Upper joint bound	The upper joint limit (in radians or meters).
Lower joint bound	The lower joint limit (in radians or meters).
Cal pos	The position of the axis when it was calibrated. If this value is changed, the robot must accordingly be fine-calibrated in the Service window.
Rotating move	Specifies if the axis is rotating (***Yes***) or linear (***No***).
wc_acc	The acceleration of the axis in radians/s^2. If the values entered are too high, the motor will run at its torque limit.
wc_dec	The deceleration of the axis in radians/s^2. If the values entered are too high, the motor will run at its torque limit. If these numbers are not available or if friction is very large, choose a reasonable number in the range 1.5 to 2.0 depending upon the inertia and motor.

• Choose **Types: Stress Duty Cycle** and specify the characteristics of the drive system at maximum utilisation.

Parameter	Description
Speed abs. max	The max. allowed motor speed (in radians/s).

• Choose **Types: Drive Unit** and specify the type of drive unit.

Parameter	Description
Speed constant	Conversion factor from voltage (Volt) to speed (radians/s).

Trimming the axes

The servo control method can be adjusted to achieve the best possible performance.

• Choose **Types: Lag control master 0** and specify the values that give good performance.

Parameter	Description
Kp	The amplification of the position control, e.g. 10. A high value will give a stiff axis that quickly assumes its new position. The value should be large without inducing overshoot in the position or oscillations of the axis.

The simplest way to tune is to increase ***Kp*** successively, until the axis starts to oscillate or show signs of overshooting. The other parameters need not be specified.

• Choose **Types: Uncal control master 0** and specify the tuning values that give good performance when jogging, and when the axis is run uncalibrated.

7.13 Defining external axes with internal drive units

The system diskette, *Set-up*, supplied with the robot, contains a number of predefined combinations of external axes. These can be found in the EXTAXIS directory and must be used to make the installation.

- When defining external axes, *Service motion parameters* must be used. To activate these, the control program must be installed once again. Select ***Service*** at the question *Service/ Standard motion parameters?* See Product Manual - Installation and Commissioning.
- Read the drive unit configuration file, EXTAXIS/UTIL/IC236C from the Set-up diskette. (Depending on the type of drive unit physically installed, read an additional IC236T or IC 236G configuration file.)
  See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add New parameters**.
  This file must be read only once, even if there are several external axes in use. The drive unit will then be defined automatically, as the common drive unit for all external axes with internal drive units.
- Read a motor configuration file, EXTAXIS/UTIL/ICMOTORS from the Set-up diskette.
  See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add New parameters**.
  This file must be read only once, even if there are several external axes in use.
- Read the files corresponding to the current installation, one file for each axis to be used.
  See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add New parameters**.

File name		Content
/EXTLIN/	ICLIN7	Linear external axis 7 with internal drive unit
	.	
	ICLIN12	Linear external axis 12 with internal drive unit

/EXTROT/	ICROT7	Rotating external axis 7 with internal drive unit
	.	
	ICROT12	Rotating external axis 12 with internal drive unit

A mechanical unit is now created for each axis. The name of this unit is the same as the name of the file, e.g. ICROT7.

- Restart the robot, **File: Restart**.
- Choose **Types: Drive unit** and specify, under the parameter ***Use Drive Unit Type***, the type of drive unit in use (type number).
- Choose **Topics: Manipulator**.
- Choose **Types: Motor** and specify, under the parameter ***Use Motor Type***, the type of the motor in use (article number).

If the article number is unknown, or not found in the list of available motor types, then use one of the general purpose types: *IC_motor_1*, *IC_motor2*, or *IC_motor_3*. Then choose **Types: Motor Type** and specify the following information.

Parameter	Description
Pole pairs	The number of pole pairs for the motor, typically 2 or 3.
ke (V/rad/S)	Nominal voltage constant, induced voltage phase to phase.
Max Current (A rms)	Maximum current without irreversible demagnetisation.

- Choose **Types: Mechanical Unit** and specify, under the parameter ***Name***, the characteristics of the mechanical axis.

Parameter	Description
Name	The name of the unit (max. 7 characters). This name will subsequently be used in the Jogging window and from the program, e.g. when an unit is to be activated.
Stand by state	The unit waits until the first move instruction or jogging command, before releasing the brakes and controlling the axis.
Activate at start up	The unit is to be activated automatically at start up.
Do not allow deact.	The unit cannot be deactivated.

- Choose **Types: Joint** and specify the logical number of the axis from the RAPID program. For example, ICROT10 is by default defined as logical axis 10. This would correspond to the *eax_d* field of a *robtarget* data.
- Choose **Types: Motor** and define the calibration and commutation offset as in *Defining the commutation offset and calibration offset of the motors* on page 43. Do not forget to set the parameters ***Cal offset valid*** and ***Com offset valid*** to *Yes*.
- Restart the robot, **File: Restart**.

The calibration offset is suitably updated by moving the axis to its calibration position and then fine-calibrating from the Service window.

The commutation offset can be calculated by running the program /SERVICE/ MOTOR on the Set-up diskette:

1. Turn the operating mode selector to Manual 250 mm/s.
2. Press in the enabling device to the intermediate position.
3. Press the emergency stop (with the enabling device still pressed in).
4. Reset the emergency stop, by acknowledging the error message and by pressing the MOTORS OFF button (still with the enabling device pressed in).
5. Open the Program Test window and select **Test: Simulate**. Specify ***Add*** in the dialog box that follows.
6. Start the program and follow the instructions shown on the display. When the brakes are released manually, the specified motor moves to the commutation angle.
7. Release the enabling device and commutate the axis in the Service window.

Add activation relays (if connected)

The external unit can be activated via signals from the robot. When the unit is activated, e.g. by selecting the unit in the Jogging window, the corresponding signal is set automatically. A check is subsequently made that the corresponding output signal from the relay is set.

- Define the input and output signals of the relays. See *Defining input and output signals* on page 11. Restart the robot, **File: Restart**, and check that the external axes can be activated via the Inputs/Outputs window.
- Read a relay configuration file, EXTAXIS/UTIL/RELAY from the Set-up diskette. See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add New parameters**. This will create a new relay with the name RELAY_x.
- Choose **Topics: Manipulator**.
- Choose **Types: Relay** and specify the name of the relay and its signal connections.

Parameter	Description
Name	The name of the relay, e.g. relay_track (max. 16 characters). This must be changed if more than one relay is used.
Delay time	The time delay between the activation signal and the electrical relay function, e.g. 0.2 s.
Inverted input	Specifies if an inverted input is to be used in the relay.
In signal	Specifies the logical name of the input signal to the relay. The name must be identical (including upper and lower case letters) to the signal name specified in the I/O configuration.
Out signal	Specifies the logical name of the output signal to the relay. The name must be identical (including upper and lower case letters) to the signal name specified in the I/O configuration.

- Choose **Types: Mechanical Unit** and specify, under the parameter ***Use act relay***, the name of the activating relay.
- When more than one activating relay is used, read a new relay (file RELAY) and repeat the above procedure.

Add brake relays (if connected)

If the external unit is equipped with a brake, it will be activated automatically when the unit is deactivated or when the robot system assumes the MOTORS OFF state. It will also be activated when the robot has been stationary for a specific time, in the MOTORS ON state.

- Define the input and output signals of the relays. See *Defining input and output signals* on page 11. Restart the robot, **File: Restart**, and check that the brakes can be activated via the Inputs/Outputs window.
- Read a relay configuration file, EXTAXIS/UTIL/RELAY on the Set-up diskette. See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add New parameters**. This will create a new relay with the name RELAY_x.
- Choose **Topics: Manipulator**.
- Choose **Types: Relay** and specify the relay name and its signal connections. See *Add activation relays (if connected)* on page 50.
- Choose **Types: Mechanical Unit** and specify, under the parameter ***Use brake relay***, the name of the brake relay.
- The point in time at which the regulator shall stop braking the axis and allow the mechanical brakes to work on their own, can be modified by choosing **Types: Brake**.

Parameter	Description
Control off delay time	Time delay (s) before the regulator is turned off, if the speed is < *Control off speed*.
Control off speed	The speed (as a percentage of the max. speed) at which the time delay starts.

- If more than one brake relay is used, read a new relay into the system (file RELAY) and repeat the above procedure.

Define other parameters for an external axis

- Choose **Types: Arm** and specify the geometrical characteristics of the axis.

Parameter	Description
Transm gear ratio	The transmission ratio of the motor to the axis. Example: For a linear axis, 21.43 means that the axis moves 1 m when the motor rotates 21.43 radians.
Upper joint bound	The upper joint limit (in radians or meters).
Lower joint bound	The lower joint limit (in radians or meters).
Cal pos	The position of the axis when it was calibrated. If this value is changed, the robot must accordingly be fine-calibrated in the Service window.
Rotating move	Specifies if the axis is rotating (***Yes***) or linear (***No***).

wc_acc	The acceleration of the axis in radians/s^2. If the values entered are too high, the motor will run at its torque limit.
wc_dec	The deceleration of the axis in radians/s^2. If the values entered are too high, the motor will run at its torque limit. If the value is too low then overshoot will be high and the axis will oscillate.

• Choose **Types: Stress Duty Cycle** and specify the characteristics of the drive system at maximum utilisation.

Parameter	Description
Torque abs. max	The max. allowed motor torque (in Nm).
Speed abs. max	The max. allowed motor speed (in radians/s).

Tuning the axes

The servo control method can be adjusted to achieve the best possible performance.

• Choose **Types: Lag control master 0** and specify the values that give good performance.

Parameter	Description
Kp	The amplification of the position control, e.g. 15. A high value will give a stiff axis that quickly assumes its new position. The value should be large without inducing overshoot in the position or oscillations of the axis.
Kv	The amplification of the velocity control, e.g. 2. A high value gives better high frequency stiffness, better response speed and low overshoot. If the value is too high the axis will vibrate. *Kv* controls the amount of damping for the axis and is the most limiting of the parameters. A poor value of *Kv* will limit *Kp* and *Ti*, and the axis will not be fully utilised.
Ti	The integration interval constant of the velocity control, e.g. 0.2. A low value gives low steady state error and better path following.

• Choose **Types: Uncal control master 0** and specify the tuning values that give good performance when jogging, and when the axis is run uncalibrated.

Simple trimming

• Set *Kp* to 5.

• Select *Ti* based on the mass moment of inertia of the external axis. *Ti* should be in the range from 0.1 for very light axes (J = 0.3 kgm^2) to 0.5 for the heaviest axes (J = 12 kgm^2).
A typical value is 0.3.

• Increase *Kv* to its highest value until the axis starts to vibrate/oscillate or a clear vibration can be heard from the axis, either during motion or when stationary. The axis velocity supervision may also indicate speed failure. When you reach the unstable point, reduce (divide) *Kv* by a factor of 2.

• Increase *Kp* in increments of 0.5 for the fastest response time, until the first signs of overshooting are observed. Then reduce (subtract) *Kp* by 1. If you observe overshooting at a later time, reduce *Kp* to an even lower value.

Final trimming

• Connect the printer channels to the test outputs in the cabinet. The outputs are marked 1 and 2 with a common zero point ground. Voltage level ±10V. (Required: a two channel printer, chart recorder, 25-135 mm/s, e.g. Brush 220.) Inputs:

- X41 0 V
- X42 TESTOUT1
- X43 TESTOUT2.

• Make sure that the external axis is commutated and calibrated. Any position may be defined as the calibration position.

• Tune the axis so it may be jogged without stopping due to speed or torque supervision. Keep the same values of *Kp* and *Kv* as above but multiply *Ti* by a factor of 2. *Ti* will be retuned last in the procedure.

• Program a back-and-forth motion of the external axes with test signals enabled. The following program may be used as an example:

```
PROC main()
TestSign 1,speed STN1,1,1,0;
TestSign 2,torque_ref,STN1,1,8,0;
ActUnit STN1;
FOR i FROM 1 TO 10 DO
      MoveJ t1,v_tune,fine,too10
      MoveJ t2,v_tune,fine,too10;
ENDFOR
ENDPROC
```

The velocity data and test positions can be modified depending to the value that is to be tuned.

TestSign *Output, SignalId, MechUnit, Axis, Scale, Stime*

Output Data type: *num*
Selection of test output, acceptable values are 1 and 2.

SignalId Data type: *testsignal*
Name of the test signal.

MechUnit Data type: *mecunit*
Mechanical unit for which test signal is required.

Axis Data type: *num*
Axis number.

Scale Data type: *num*
Scaling factor. Acceptable values are 1, 2, 4, 8 , 16, etc.

Stime Data type: *num*
Sampling time in seconds. The test output is updated with a new value at each sampling.
The value 1 denotes an updating every second.
The value 0 denotes an updating as often as possible.
The value 0.01 denotes 100 updatings per second.

Nominal acceleration and deceleration

If an axis has a variable moment of inertia, **Nominal acceleration** and **Nominal deceleration** should be tuned with the maximum inertia.

If gravity has an influence on the axis, then **Nominal acceleration** should be tuned with a motion accelerating upwards against gravity. **Nominal deceleration** should be tuned with a stopping motion (deceleration) while moving downwards with gravity.

Selecting Positions and velocity:

Program two test points for acceleration and two test points for deceleration with the following requirements:

- Velocity: choose a velocity that is approximately **50%** of the maximum speed of the external axis, i.e. speed test signal of approximately **2.5 V**.
- Distance: the distance should be chosen to ensure that the axis stabilises at the programmed velocity before deceleration.

1. Using the chart recorder, record the values of *speed* and *torque_ref* for the external axis.

2. Use test positions for **Nominal acceleration**. Run the motion and check the value of *torque_ref* for the torque limit. Adjust the value of **Nominal acceleration** upwards or downwards in increments of **0.5** until the *torque_ref* signal shows that the axis does not reach the torque limit. Reduce the final value by **10%** to allow for variations of the mechanical system over time.

3. Use test positions for **Nominal deceleration**. Run the motion and check the value of *torque_ref* for the torque limit. Adjust the value of **Nominal deceleration** upwards or downwards in increments of **0.5** until the *torque_ref* signal shows that the axis does not reach the torque limit. Reduce the final value by **10%** to allow for variations of the mechanical system over time.

Position and speed control, internally controlled axes

If the axis has a variable moment of inertia, the *Kp* and *Kv* gains should be tuned with the maximum value for the moment of inertia. The integral gain, *Ti*, should also be tuned with the maximum moment of inertia.

Selecting positions and velocity:

Program two test points with the following requirements:

Velocity: choose a velocity that is approximately **5%** of the maximum speed of the external axis. The speed must be low enough to guarantee that the axis does not encounter the current limit but high enough to prevent friction from affecting the result.

Distance: approximately **10** revolutions of the resolver. The distance should be chosen to ensure that the axis stabilizes at the programmed velocity before deceleration starts.

1. Using the chart recorder, record the values of *speed* and *torque_ref* for the external axis.

2. Adjust the value of *Kv* for the **Lag ctrl master 0** for the external axis. *Kv* should be adjusted to be as high as possible on the basis of the noise and vibration of the external axis, as in simple tuning above, but verified using the chart recorder on the *speed* and *torque_ref.*

Divide *Kv* by a factor of **2** to provide a robustness margin.

7.14 Activate notch filter for an external axis

This is used only in arc welding applications when a variation in external axis speed affects the welding process. This problem sometimes occurs when both coordinated interpolation and weaving are used. The frequency of the speed variation is typically 2 times the weaving frequency. The notch filter will prevent the external axis from oscillating at the weave motion frequency.

• Choose **Types: Lag control master 0**.

Parameter	Description
Notch filter activated	Yes (if activated), No (otherwise)
Notch filter frequency	Frequency of speed variation Typical value: $\frac{\text{2 x Weld speed}}{\text{Weave length}}$
Notch filter width	Width of notch filter. A higher value increases the width but can also have a negative effect on the performance (response) of the external axis. Recommended value: 0.2.

CONTENTS

Page

File Manager

The File Manager is used to

- copy or transfer files,
- change the name of a file,
- create directories on diskettes or other mass storage devices,
- print files,
- format diskettes.

1 Program/Data Storage

Programs and data are stored as normal PC text files. These can be saved and restored to/from a diskette or an internal RAM disk.

The *diskette* is a standard 3.5", High Density, 1.44 Mbytes, DOS formatted diskette.

Note. Before saving programs and data, the diskette should be formatted in the robot or in a PC. Pre-formatted DOS diskettes will not always operate satisfactorily.

Note. The diskettes must never be stored inside the cabinet as the information on them can be destroyed due to heat and magnetic fields.

The internal *RAM disk* is a special part of the robot's memory, and can be used in the same way as a diskette.

A *file* can be a program, data created by the program or system parameters and the like, stored in some sort of mass storage.

Directories are used to group files together to achieve a memory unit that is more structured. For example, test programs in one directory and production programs in another (see Figure 1).

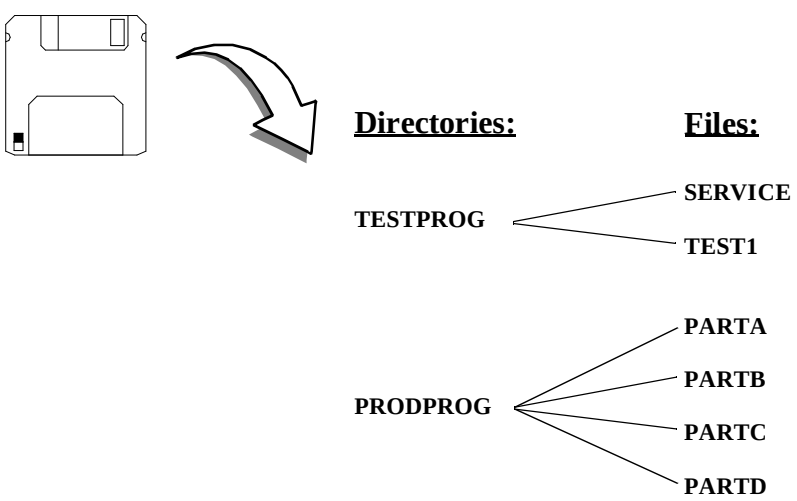


Figure 1 The files can be stored in directories on a diskette.

2 The FileManager Window

• Press the Miscellaneous key  .

• Select ***FileManager*** in the dialog box that appears.

• Press Enter [↵] .

The FileManager window will be displayed (see Figure 2)

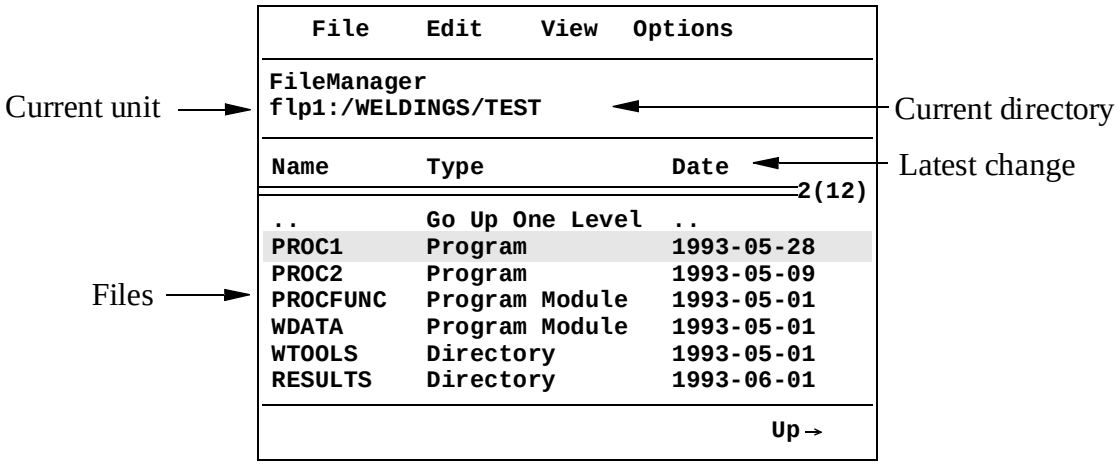


Figure 2 The FileManager window displays all files in a directory.

• Choose the desired unit from the **View** menu:

- Diskette **View: [flp1:]**
- RAM disk **View: [ram1disk:]**

2.1 Choosing a directory

• Select the desired directory.

• Press Enter [↵] .

The directories and files located in the chosen directory will be displayed. The next directory above this can be selected by moving to the top line in the list (..) and then pressing Enter [↵] , or by using the ***Up*** function key.

2.2 Viewing file information

• Select a file in the list and press Enter [↵] .

The following information will be specified:

- the name and type of the file,
- the size of the file in bytes,
- the date and time when the file was last changed.

• Choose ***OK*** to terminate the dialog.

3 Creating or Moving Files and Directories

3.1 Creating a new directory

• Choose **File: New Directory.**

A dialog will be displayed, as in Figure 3.

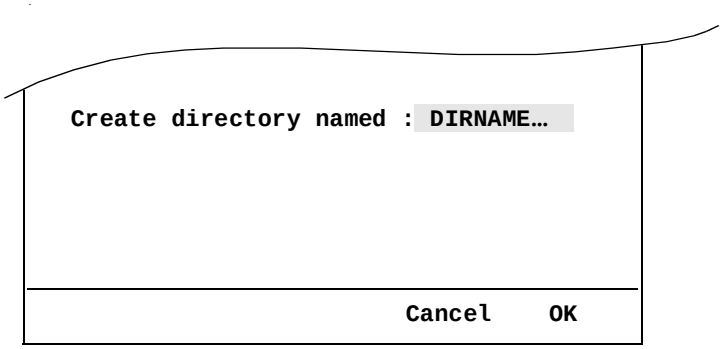


Figure 3 The New Directory dialog.

• Press Enter ↵.

• Enter the new name and press ***OK***.

Confirm by pressing ***OK***. The directory will be created under the current directory.

3.2 Renaming a file or a directory

• Choose **File: Rename**.

A dialog will be displayed, as in Figure 4.

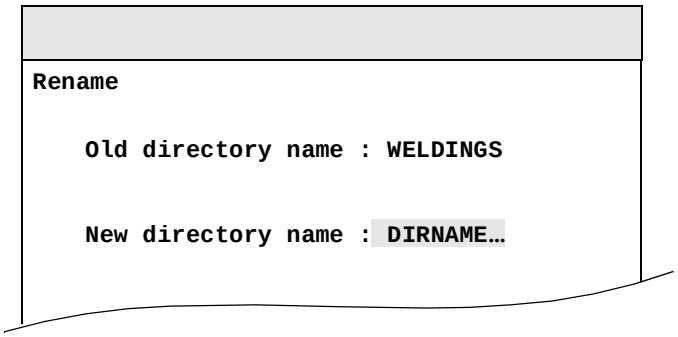


Figure 4 The Rename dialog for a directory.

• Press Enter ↵.

• Enter the new name (max. 8 characters) and press ***OK***.

• Confirm by pressing ***OK***.

3.3 Deleting a file or directory

- Select the desired file or directory.
- Press Delete [⌫] .
- Choose ***OK*** to confirm the deletion.

You can only delete a directory if it is empty.

3.4 Copying files and directories

- Select the file or directory to be copied. If you select a directory, all subordinate directories and files will also be copied.
- Choose **File: Copy.**

A dialog will be displayed, as in Figure 5.

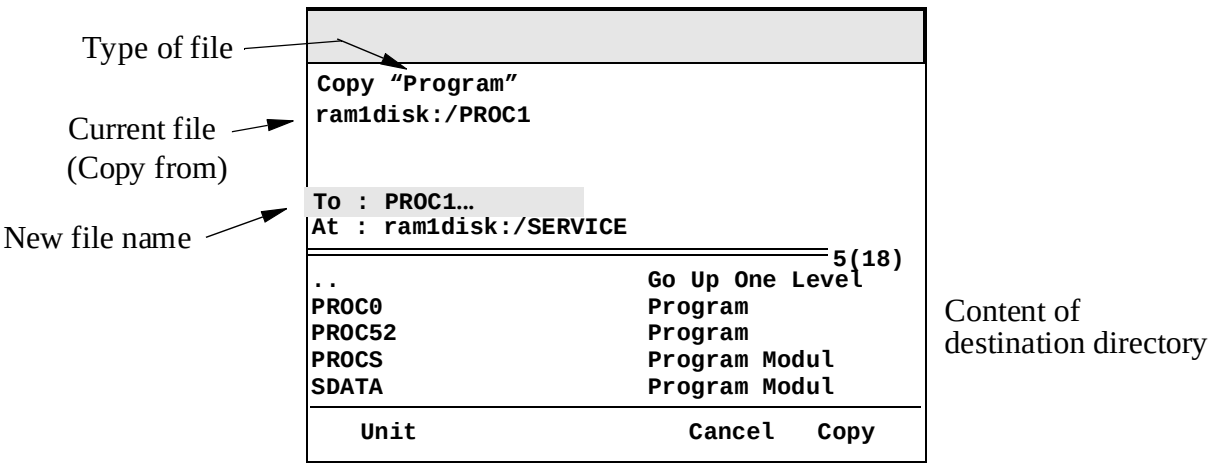


Figure 5 The dialog for copying files or catalogues.

- Specify the name of the new file by selecting the field ***To***, and press Enter [↵]. If you do not specify a name, the copied file/directory will be given the same as the original.
- Specify the destination unit (first part of **At** field) by pressing the Unit function key. If you do not specify a unit, the same unit that was used originally will be used.
- Specify the destination directory (latter part of **At** field) by selecting the lower part of the window. Select the desired directory and press Enter [↵]. If you do not specify a directory, the same directory that was used originally will be used.
- Choose ***Copy*** to start copying.

3.5 Moving files and Directories

- Select the file or directory that is to be moved.
- Choose **File: Move.**

A dialog will be displayed, as in Figure 6.

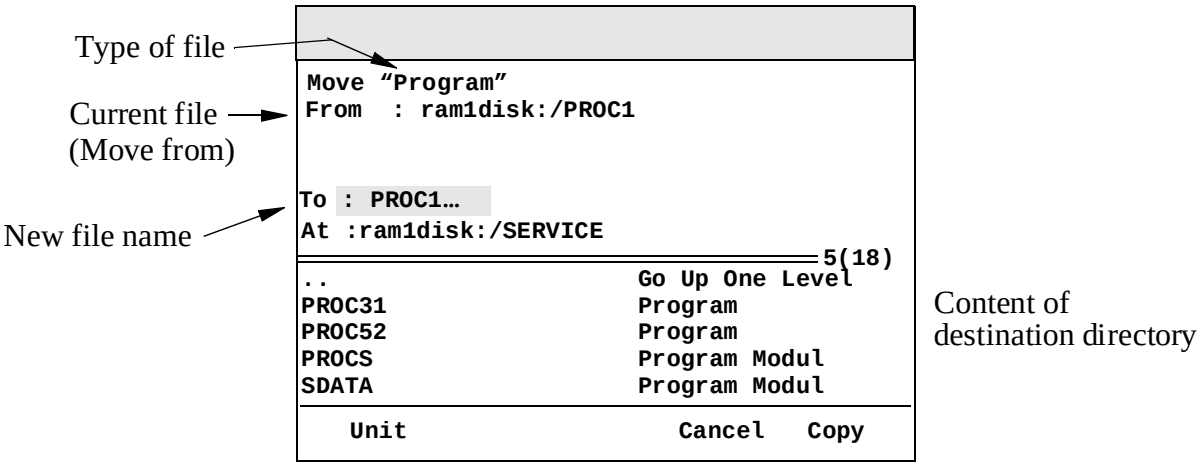


Figure 6 The dialog for moving of files and catalogues.

- Give the file to be moved a new name by selecting ***To***, and press Enter ↵. If you do not specify a new name, the file/directory that is moved will retain the same name.
- Specify the destination unit (first part of **At** field) by pressing the Unit function key. If you do not specify a unit, the same unit that was used originally will be used.
- Specify the destination directory (latter part of **At** field) by selecting the lower part of the window. Select the desired directory and press Enter ↵.
- Choose ***Move*** to start moving.

3.6 Printing files

- Select the file to be printed.
- Choose **File: Print File**
- Choose ***OK*** to start printing.

4 Formatting a Diskette

NB: The previous contents on the diskette will be erased when formatting.

- Choose **Option: Format**.

A confirmation dialog will be displayed.

- If desired, rename the diskette and press Enter ↵.
- Choose ***OK*** to start formatting.

CONTENTS

Page

Service

The Service window is used to

- obtain information on the robot system
- view and change logs (e.g. error log)
- calibrate the measuring system for the robot and external axes
- commutate the motors for the robot and external axes
- set the date and time.

For more detailed information on service, maintenance and troubleshooting, see the Product Manual.

1 The Service Window

• Press the Miscellaneous key  to open the Service window.

• Select ***Service*** in the dialog box that appears.

• Press Enter ↵ .

The service window comprises a number of different windows:

Window title	Used to:
Service Date & Time	Change the current date and time
Service Logs	View logs.
Service Calibrate	Test/Calibrate the measuring system for the robot or external axes.
Service Commutate	Test/Commutate the motors for the robot or external axes.
Frame Definition	Define base or user coordinate system.
Two Axes Definition	Define user frame for a two axes rotational mechanical unit
System Info	Obtain information about storage capacity, task states, system and product ID etc.

2 Changing the Current Date and Time

• Choose **View: Date & Time**.

A dialog box will be called up displaying the current date and time (see Figure 1).

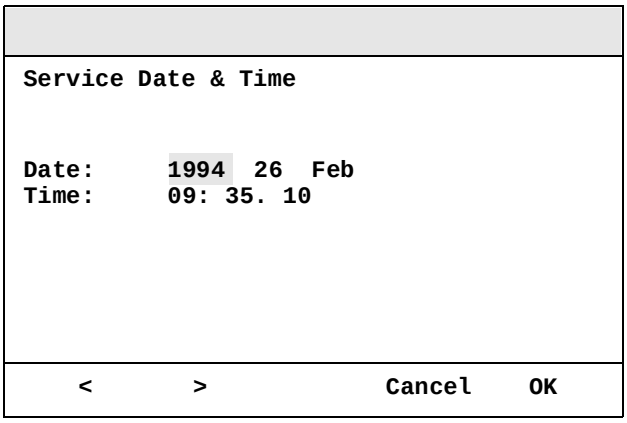


Figure 1 The dialog box used to set the date and time.

- Select that which you wish to change using the arrow keys.
- Using the function keys, < (decreases) and > (increases), change the date or time.
- Choose ***OK*** to confirm.

3 Logs

3.1 What is a log?

All messages reported, such as error messages and changes in the status, are stored in a log. Each message stored is timestamped and it is thus possible to determine the order of events from a log.

When the maximum number of messages in a log is attained, a new message will replace the oldest one.

3.2 What types of logs are there?

The following logs exist:

Name	Max. limit	Used to show
Common	40	All messages
Operational	20	Changes in the status, e.g. a change of operating mode
System	20	The messages related to the control program
Hardware	20	The messages related to defective hardware components
Motion	20	Any messages that appear when moving the robot or other mechanical units
Program	20	Any messages displayed during program execution
Operator	20	Any messages that appear when using the teach pendant
I/O & Communication	20	The messages related to I/O and communication
User	20	User defined messages (by using the instruction ErrWrite)
Arc Welding	20	The messages related to the arc welding process
Spot Welding	20	The messages related to the spot welding process
Internal	20	Internal errors – does not usually contain any messages

3.3 Viewing all logs

• Choose **View: Log**.

The window will display information on all logs in the robot (see Figure 2).

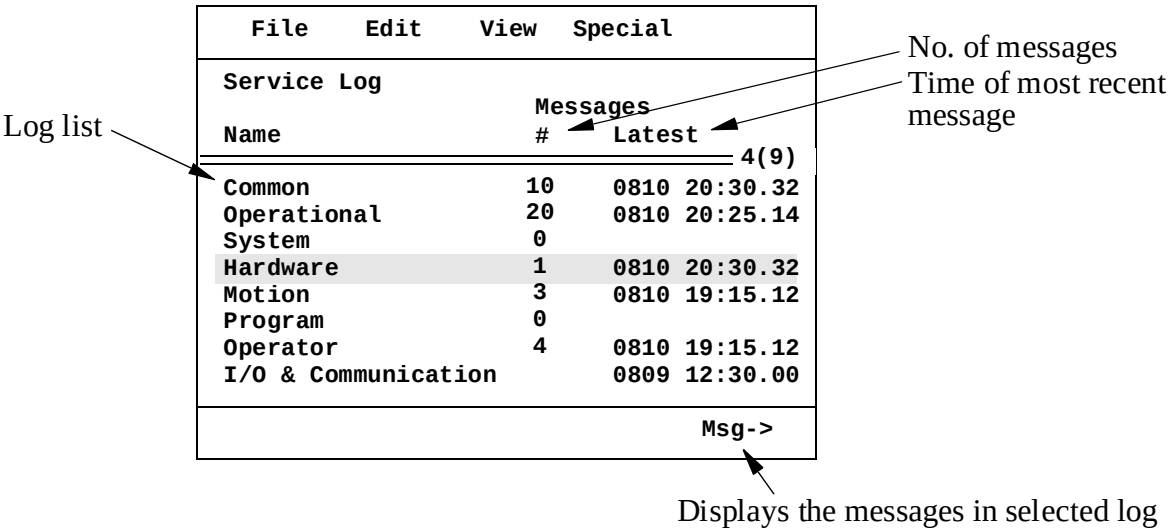


Figure 2 The Service Log window displays all existing logs.

3.4 Viewing a message in a log

- Open the Log window by choosing **View: Log**.
- Choose the log you wish to look at by selecting that log from the list and pressing the ***Msg*** function key, or press Enter ↵.

The window will display all messages for the log that you choose (see Figure 3).

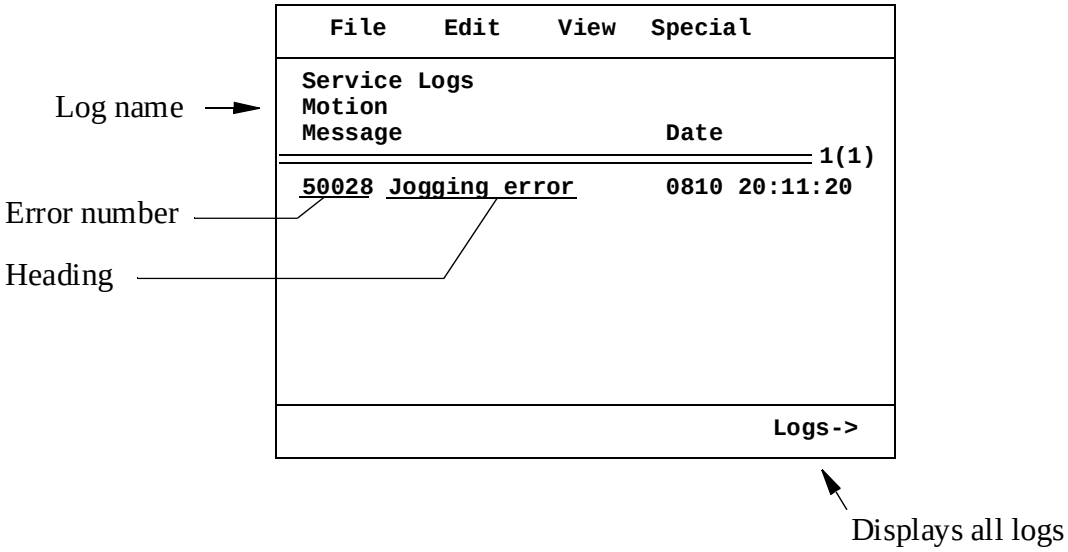


Figure 3 The Service Log Messages window displays all messages in the log.

- You can obtain more information on a specific message by selecting the message and pressing Enter ↵, or by choosing **Edit: Info**.

3.5 Erasing the contents of a log

- Open the Log window by choosing **View: Log**.
- Select the log to be erased.
- Choose **Special: Erase Log**.
- Choose ***OK*** to confirm.

3.6 Erasing the contents of all logs

- Open the Log window by choosing **View: Log**.
- If there are log messages displayed, press the function key ***logs***.
- Choose **Special: Erase All Logs**.
- Choose ***OK*** to confirm.

3.7 Updating the contents of a log automatically or by means of a command

When you view a log message and a new message appears, you have two choices: you can either update the log

- automatically when the message appears; or
- update the log using the function key ***Update***.
  (The ***Update*** function key is only visible if there are more messages.)

To update automatically:

• Choose **Special: Update log on Event.**

To update on command:

• Choose **Special: Update log on Command.**

3.8 Avoiding normal error reports

When trying to isolate faults in different hardware components, you may not wish to be shown error alert boxes. To prevent these appearing:

• Open the Log window by choosing **View: Log**.

• Choose the Common log by selecting it and pressing the ***Msg*** function key, or press Enter ↵.

Now, error alert boxes will not be shown. Error messages will be either displayed directly (if you chose **Special: Update log on Event**) or following a command in the log (if you chose **Special: Update log on Command**).

3.9 Saving log messages on diskette or some other mass storage device

• Open the Log window by choosing **View: Log**.

• Select the log that you wish to save and choose **File: Save As**.

A dialog box will appear, displaying the contents of the current directory (see Figure 4).

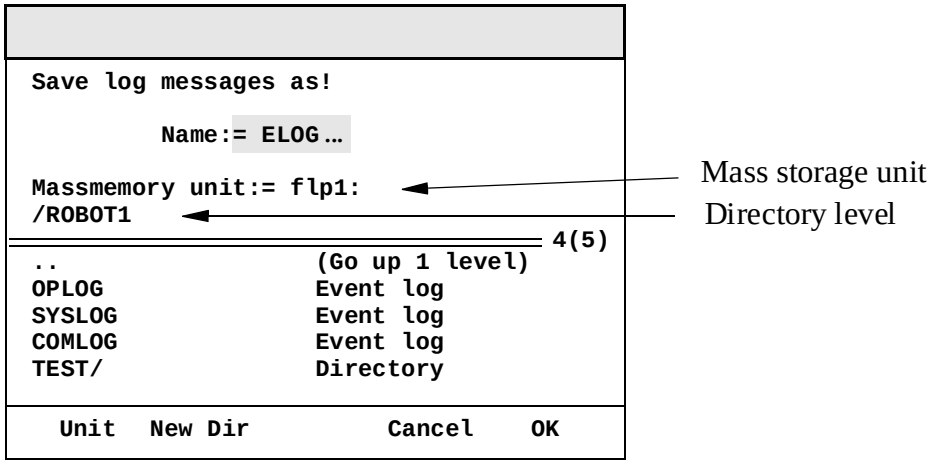


Figure 4 Dialog box for storing logs.

- If necessary, change the mass storage unit by pressing the ***Unit*** function key until the correct unit is displayed. To store on a diskette, choose ***flp1:***.
- Select the ***Name*** field, press Enter ↵ and enter the new name in the dialog box that appears. Choose ***OK*** to confirm.
- Select the directory to which the log is to be saved. You can move to the next directory level by selecting the desired directory or '. .' (upwards) and pressing Enter ↵.

  Create a new directory by pressing the ***New Dir*** function key. Specify the new directory name in the dialog box that appears. Choose ***OK*** to confirm.

- Choose ***OK*** to confirm the save.

4 Calibration

4.1 What is calibration?

Calibration involves setting the calibration positions of the axes and is used as the basis for their positioning. If the robot or external axes are not correctly calibrated, this will result in incorrect positioning and will have a negative effect on the agility of the robot. The robot is calibrated on delivery.

For more information see *Calibrating the robot* in Chapter 10 Calibration in this manual.

5 Commutation

5.1 What is commutation?

Each motor must be commutated in order to be able to utilise it to its full capacity. Commutation involves reading the resolver value when the motor is in a given pose. The robot motors are commutated on delivery.

For information on how to do this, see the section on *Repairs* in the Product Manual.

6 Frame Definition

See *Frames* in Chapter 10 Calibration in this manual.

7 Two Axes Definition

See *Frames* in Chapter 10 Calibration in this manual.

8 Obtaining information on the robot system

• Choose **View: System Info.**

A list of topics is shown in the dialog box (see Figure 5).

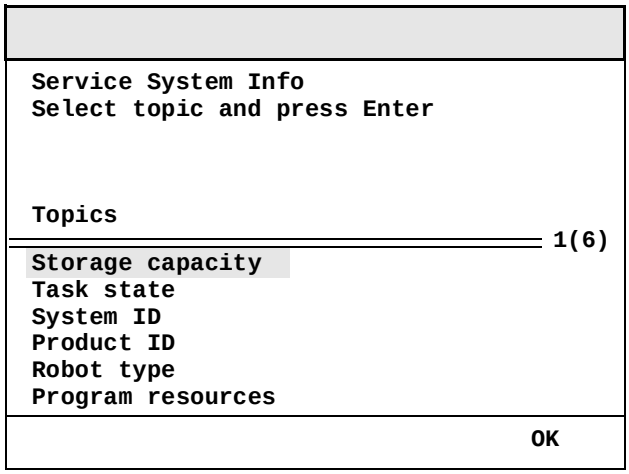


Figure 5 The system information window.

• Select a topic using the arrow keys and press Enter ↵. Information on the selected topic will be displayed.

Topic	Description
Storage capacity	All available storage devices are shown in a list containing the device name, free space and total size.
Task state	All tasks are shown in a list containing task name, free memory, the total assigned memory and the current task state. Possible task states are uninitialised, ready, executing and stopped.
System ID	The unique system identification code is shown.
Product ID	The identification code for all installed products is shown.
Robot type	Shows the robot type specification.
Program resources	Shows the total program memory before task configuration and the maximum number of persistents.

• Press ***Update*** to update the information.

CONTENTS

Page

Programming ArcWare

Before you start to program arc welding instructions, you must configure the arc welding system and any external axes. This is described in Chapter 12 of this manual, System Parameters.

1 Programming Arc Welding

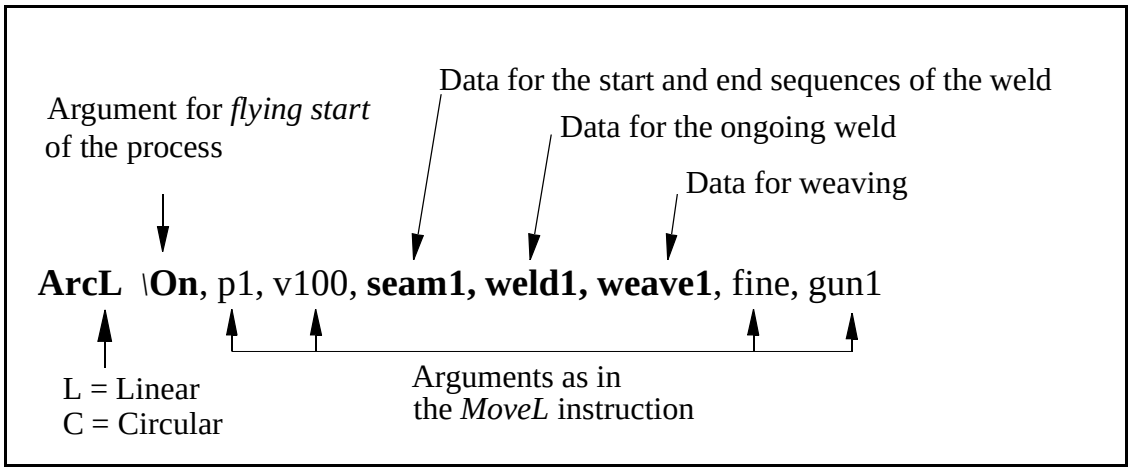
1.1 Program structure

When there are several seams to be welded on an object, the welding sequence may be of critical importance for the quality of the object. The risk of deformation due to thermal stress can be reduced by choosing the seam welding sequence. It is often best to make a specific routine for this, *object routin*e, with all the seams specified in the correct order. When the object is placed in a positioner, its orientation can also be specified in the object routine.

The object routine can call a welding routine for each seam to be welded.

1.2 Arc welding instructions

An arc welding instruction basically contains the same types of information as a positioning instruction. However, each arc welding instruction includes a further three arguments, *seam*, *weld* and *weave*, that serve as data for the arc welding process (data types: *seamdata*, *welddata* and *weavedata*).



The speed argument, *v100*, in the instruction is only valid when executing the program instruction-by-instruction (forwards or backwards). The *process speed* in different phases of the process is included as components of seam and weld data.

For more information on programming this type of instruction, see *Programming arc welding instructions* on page 5.

1.3 Defining arc welding data

Before starting to program arc welding instructions, you should define the arc welding data that is to be used. This data is divided into three types:

- *seamdata*; describes how the weld is to be started and ended,
- *welddata*; describes the actual welding phase,
- *weavedata*; describes how any weaving is to be carried out.

The exact components of the above data depend on the configuration of the robot at the time.

Normally, data is stored as a part of the program. However, when data is to remain in the memory regardless of which program is loaded, it is stored in a system module.

• Open the *Program Data Types* window by choosing **View: Data Types**.

• Select the type *seamdata*, *welddata* or *weavedata* and press Enter ↵.

• Press the function key ***New***.

A window appears, displaying the name of the data (see Figure 1).

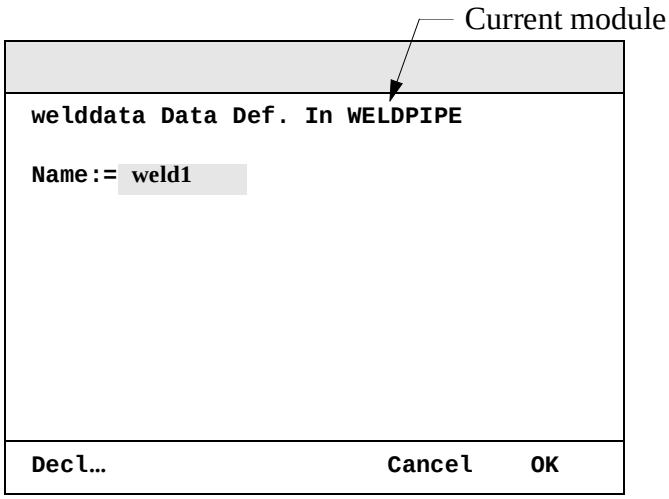


Figure 1 Definition of welddata

• Change the name by pressing Enter ↵ and specify a new name.

• Press the function key ***Decl***.

A dialog box appears, displaying the basic welddata declaration (see Figure 2).

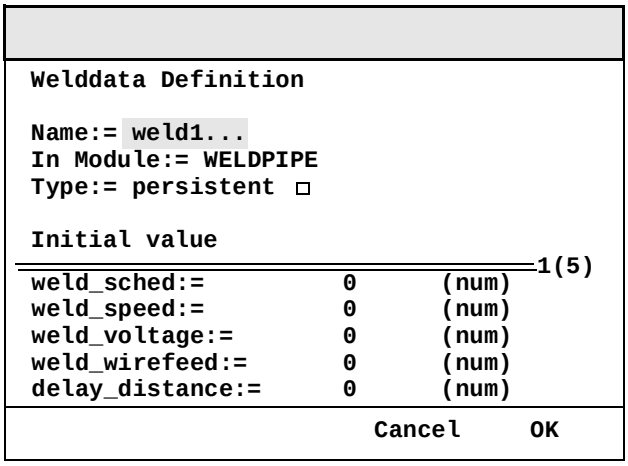


Figure 2 Definition of welddata.

- If you wish to save the data in another module, select the field *In Module* and press Enter ↵ . Choose the module in which the data is to be saved.
- Select the lower part of the window by pressing the List key.
- Select the appropriate component in the data and specify the desired value. More information on the individual components can be found in RAPID Reference Manual.
- Choose ***OK*** to terminate the definition.

Tip It is sometimes easier to create new data by copying and modifying existing data.

1.4 Programming arc welding instructions

- Jog the robot to the desired destination position.
- Call up the instruction pick list by choosing **IPL1: Motion & Process**.
- Select the instruction *ArcL* or *ArcC*.

The instruction will be added directly to the program, as illustrated in Figure 3. The arguments are set in relation to the last arc welding instruction that was programmed.

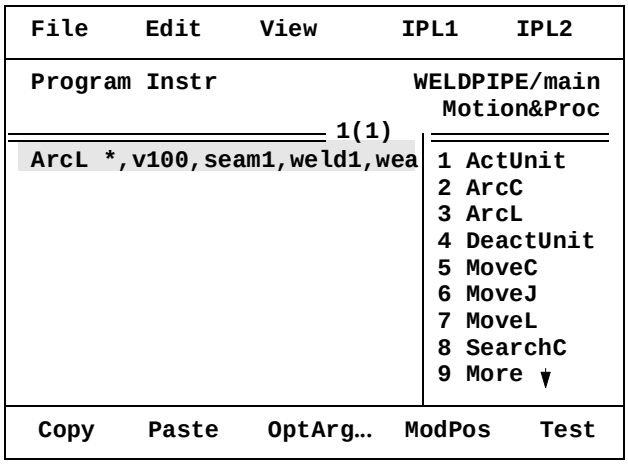


Figure 3 An arc welding instruction is added directly to the program.

If the correct argument was chosen, the instruction is now ready for use. However, for the purposes of this exercise, we will also change the seam, weld and weave arguments.

• Select the argument you wish to change (*seam1* in this example).

• Press Enter [↵].

The window used to program instruction arguments appears. The selected argument is marked with a ? in front of it (see Figure 4). The lower part of the box displays all available seam data that can be selected.

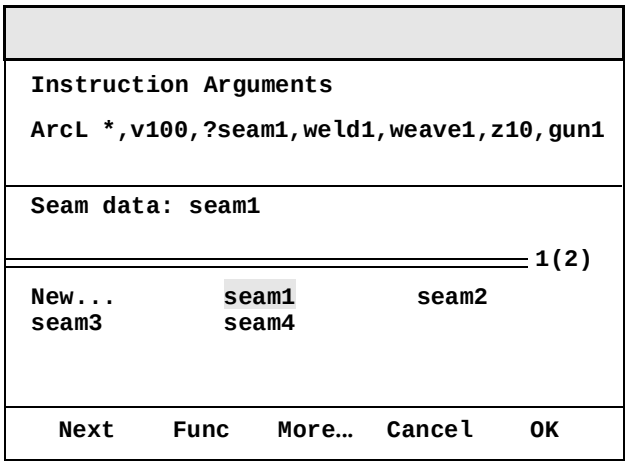


Figure 4 The dialog box used to change seamdata.

• Select the desired seam data.

• Move to the next argument (weld data) by pressing ***Next***.

All available weld data will be displayed (see Figure 5).

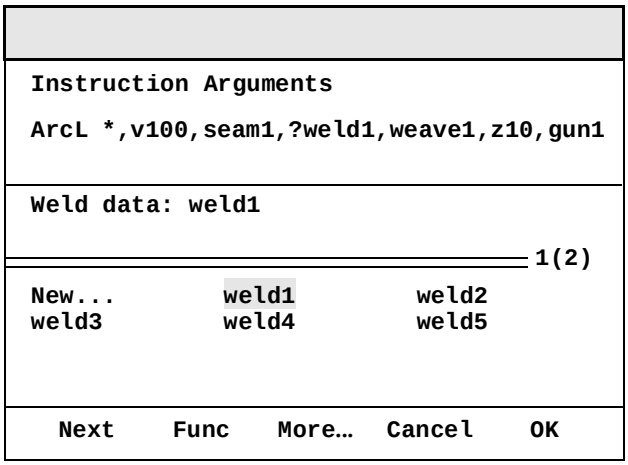


Figure 5 The dialog box used to change welddata.

• Select the desired weld data.

• Move to the next argument (weave data) by pressing ***Next***.

All available weave data will be displayed (see Figure 6).

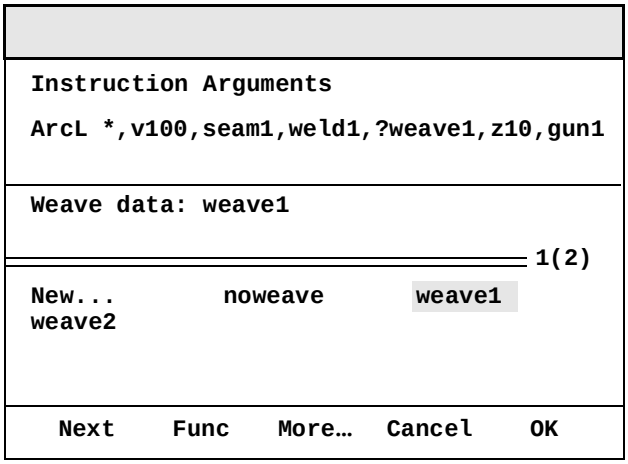


Figure 6 The dialog box used to change weavedata.

• Select the desired weave data.

• Choose ***OK*** to confirm the change.

The instruction is now ready for use.

1.5 Example of an arc welding instruction

The seam illustrated in Figure 7 is to be welded. The seam line is represented by the thick line in the figure.

The xxxxx characters between points p10 and p20 designate a *flying start*. In other words, preparations (e.g. gas preflowing) for welding are carried out on the way to the starting-point, p20. The weld is terminated at point p80.

The weld data, *weld1*, applies until position a is reached, where a transition to *weld2* takes place.

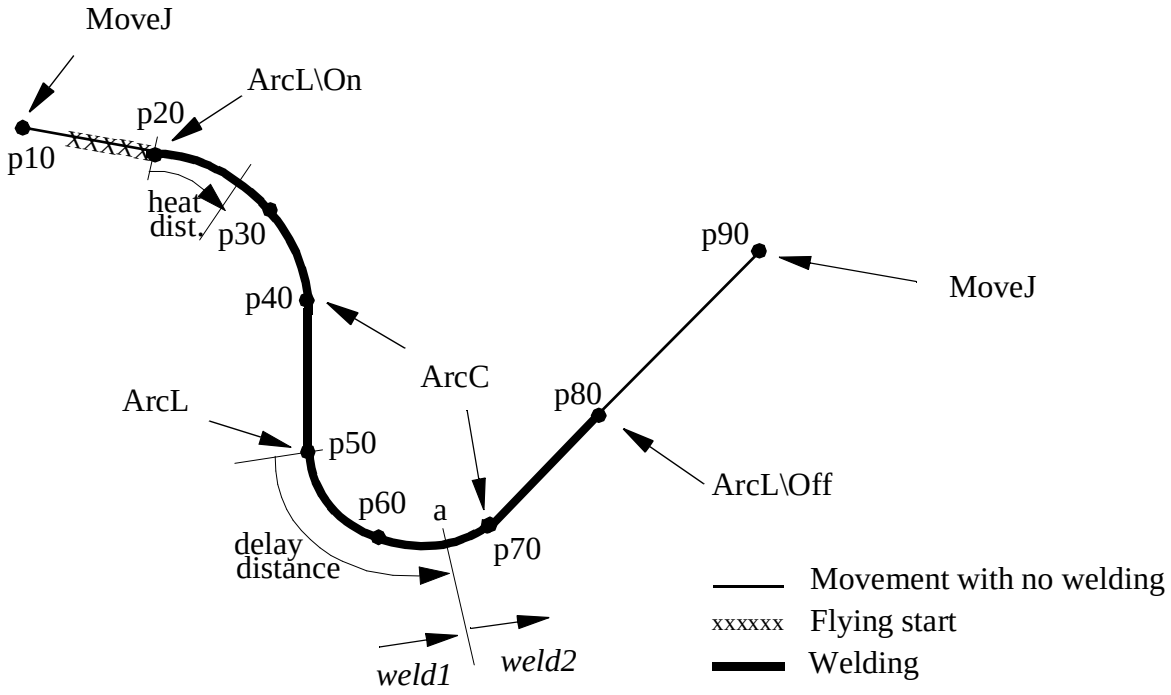


Figure 7 An example of an arc welding instruction.

The programming sequence for this weld could be written as follows:

```
MoveJ p10, v100, z10, gun1;
ArcL  \On, p20, v100, seam1, weld1, weave1, fine, gun1;
ArcC  p30, p40, v100, seam1, weld1, weave1, z10, gun1;
ArcL  p50, v100, seam1, weld1, weave1, z10, gun1;
ArcC  p60, p70, v100, seam1, weld2, weave1, z10, gun1;
ArcL  \Off, p80, v100, seam1, weld2, weave1, fine, gun1;
MoveJ p90, v100, z10, gun1;
```

If the weld is to be coordinated with an external axis, an argument of the type *workobject* must be included in all arc welding instructions except for the start instruction. For more information, see RAPID Reference Manual.

The following values apply to the *welddata* components in the above example:

Component	weld1	weld2	
weld_speed	15 mm/s	15 mm/s	
weld_voltage	45 V	50 V	
weld_wirefeed	20 m/min.	20 m/min.	
delay_distance	25 mm	0 mm	(weld2 is delayed 25 mm)

The schedule number, voltage adjustment and current adjustment of the components are not active in this example.

2 Manual functions for arcwelding

More than one arcwelding system can exist configured in the robot.

The dialogs for manual functions all start with the window *Program Test*;

• Select **View: Test**

2.1 Choice of arcwelding system

• Select **Arcweld: Select syst**.

The following dialog box is displayed (see Figure 8):

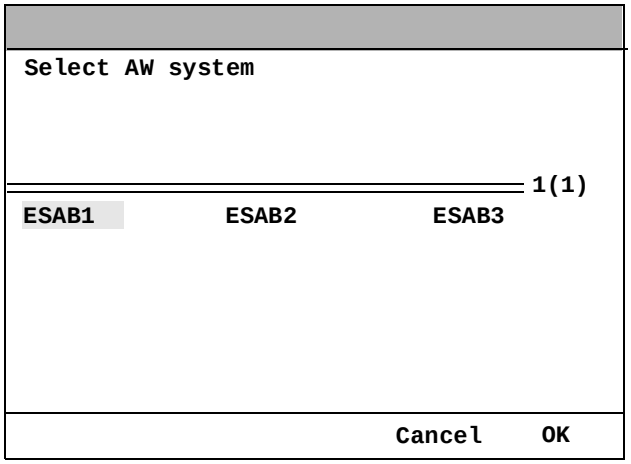


Figure 8 Dialog box for selection of arcwelding system.

• Select the desired system using the arrow keys.

• Choose ***OK*** to confirm.
If 'Cancel' is chosen, the previous AW system is retained as the current system.

2.2 Blocking certain parts of the process

• Select **Arcweld: Blocking**.

The following dialog box is displayed (see Figure 9):

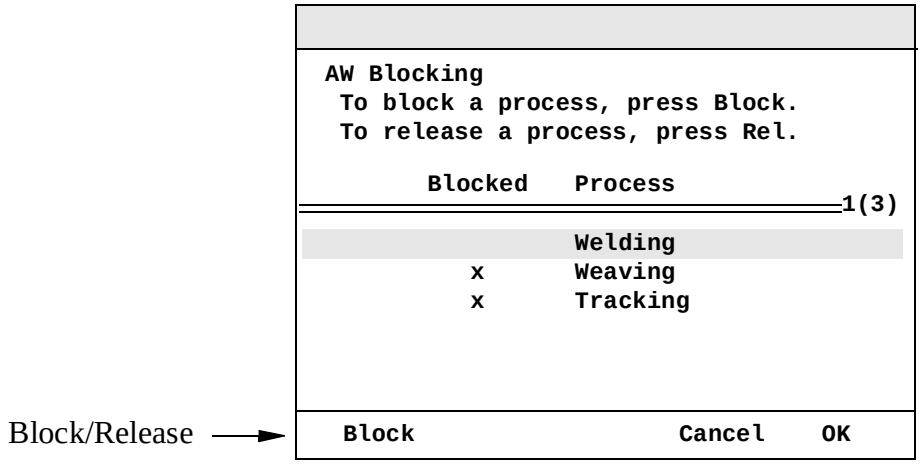


Figure 9 Dialog box for blocking.

• Select the desired process using the arrow keys.

• Select ***Block*** or ***Release***.

• Choose ***OK*** to confirm.

Blocking can also be implemented by activating the digital inputs.

The parts of the process that have been blocked will be shown in the *Program Test* and *Program Run* windows.

Blocking that is implemented in the above dialog, is active only in the Manual operating mode.

2.3 Manual wire feed

• Select **Arcweld: Man wiref**.

The following dialog box will be displayed (see Figure 10):

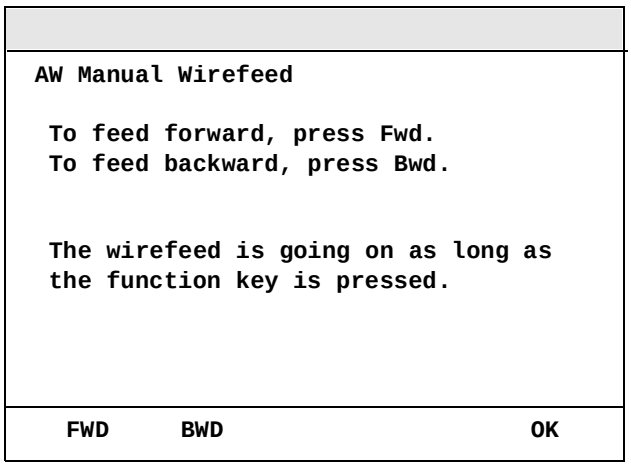


Figure 10 Dialog box for manual wire feed.

• Select ***FWD*** or ***BWD***.

The wire will be fed forwards or backwards at the preset speed, as long as the function button is pressed in.

• Choose ***OK*** to terminate the dialog.

2.4 Manual gas on/off

• Select **Arcweld: Gas on/off**.

The following dialog box will be displayed (see Figure 11):

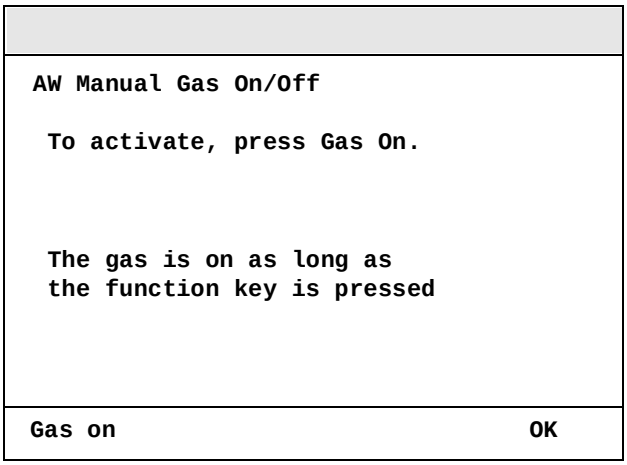


Figure 11 Dialog box for gas on/off.

• Select ***Gas on.***

The output GAS will remain activated as long as the function button is pressed in.

• Choose ***OK*** to terminate the dialog.

3 Tuning arcwelding data

Certain arcwelding data can be tuned using the Tuning function.

There are two stored values for all tunable arcwelding data, namely, the current value and the so called ”original value”. This allows you to be able to see how much the value was changed and to be able to revert to the original value.

When tuning, it is always the current value that is changed.

The original value can also be updated, i.e. it can be set to the same value as the current value. This updating can be done by the operator using a dialog.

The same changes can also be made in the *Program Data* window.

3.1 Tuning welding data when program execution has been stopped

• Select **Arcweld: Weld Tuning**.

A dialog box is shown where the current values and tuning of the latest selected welding data is displayed (see Figure 12).

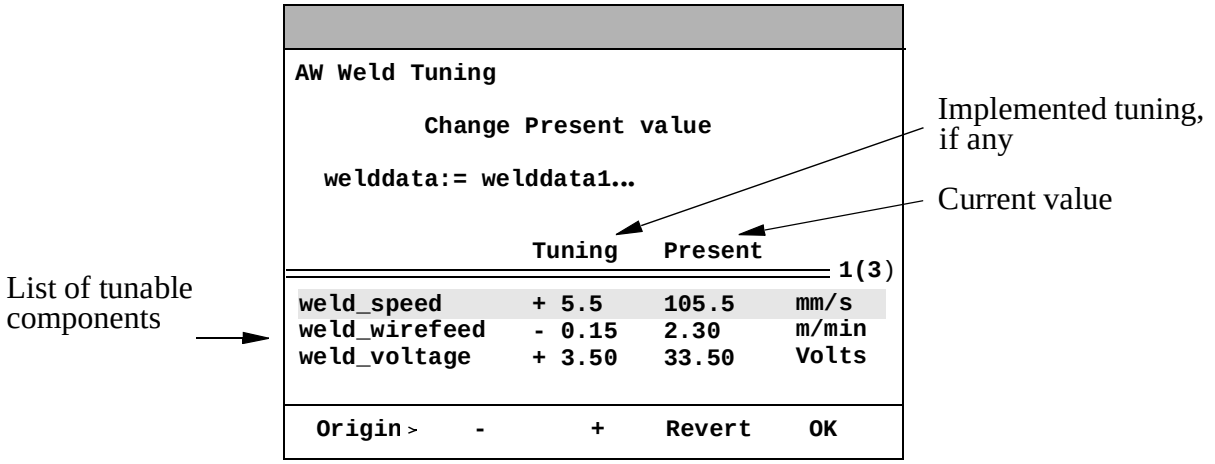


Figure 12 Dialog box for tuning current welding data.

Select welding data to be tuned

• Select the field *welddata* and press Enter [↵].

• Select the desired welding data in the dialog box displayed, and press ***OK***.

Tuning current values

Select the values to be tuned, in the lower part of the box, and press the function keys "-" or "+". Each time these keys are pressed, the value will decrease/increase in increments. The tuning increment can be preset. For adjustment see *Changing the tuning increments* on page 16.

Resetting the current tuning value

• Press ***Revert*** to reset the tuning value.
The current value will be reset to the original value.

Updating the original value to the current value

• Press ***Origin***

A dialog box will be shown where the current value and the original value are displayed (see Figure 13).

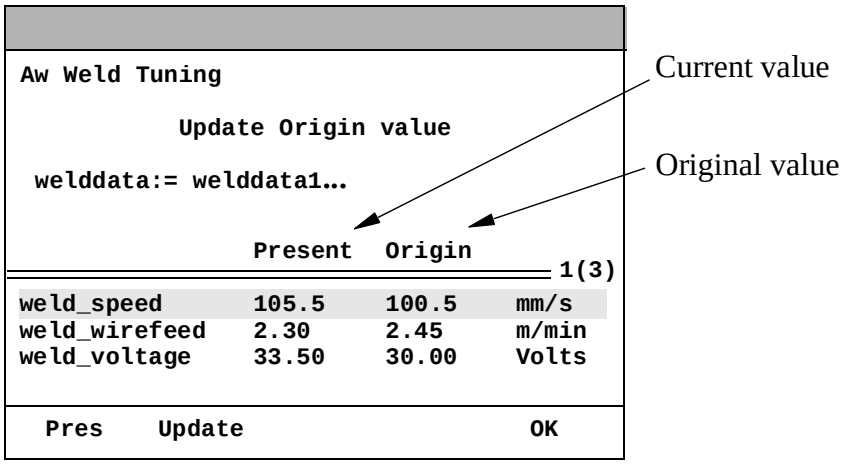


Figure 13 Dialog box for updating the original value.

- Press ***Update***.
  The original value will be reset to the current value.
- Press ***Pres*** to revert to the first dialog box .
- Press ***OK*** to terminate the dialog.

3.2 Tuning weaving data when program execution has been stopped.

- Select **Arcweld: Weave Tuning**.

A dialog box is shown where the current values and tuning values, if any, for the latest selected weaving data are displayed (see Figure 14).

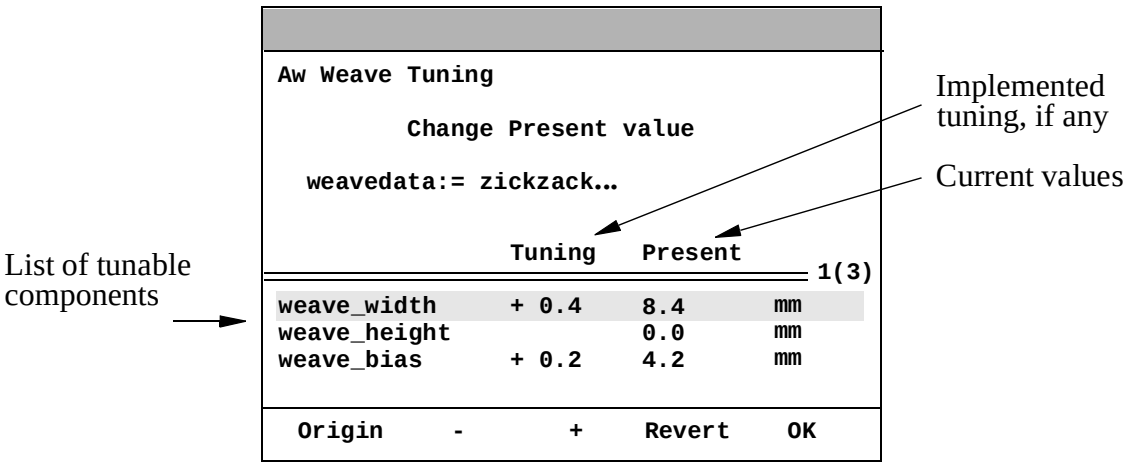


Figure 14 Dialog box for tuning current weaving data.

Selecting weaving data to be tuned

- Select the field *weavedata* and press Enter [↵] .
- Select the desired weaving data in the dialog box displayed and press ***OK***.

Tuning current

Select the values to be tuned, in the lower part of the box, and press the function keys "-" or "+". Each time these keys are pressed, the value will decrease/increase by a fixed value (defined in the system parameters).

Resetting the current tuning value

• Press ***Revert*** to reset the tuning value.
The current value will be reset to the original value.

Updating the original value to the current value

• Press ***Origin***

A dialog box is shown where the current and original values are displayed (see Figure 15).

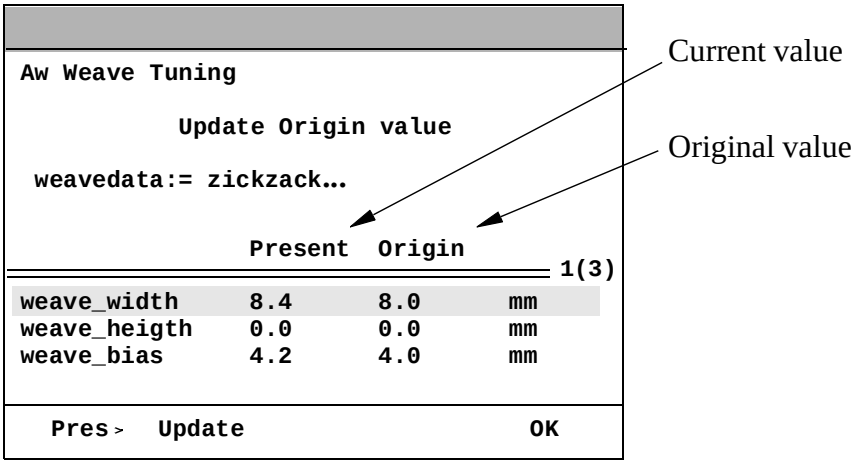


Figure 15 Dialog box for updating original value of weaving data.

• Press ***Update***
The original value will be reset to the current value.

• Press ***Pres*** to revert to the first dialog box .

• Press ***OK*** to terminate the dialog.

3.3 Tuning data while the program is executing

Certain data can be tuned while it is active, when the program is executing. However, only the current values can be tuned. The original values can be tuned only when program execution has been terminated.

Tuning while the program is executing is not possible when Hold-to-run is active.

• Start with the *Program Test* window.

• Press ***Start***.

The window *Program Run* will be displayed, during program execution, with a list of tunable data of the type that was latest selected (see Figure 16).

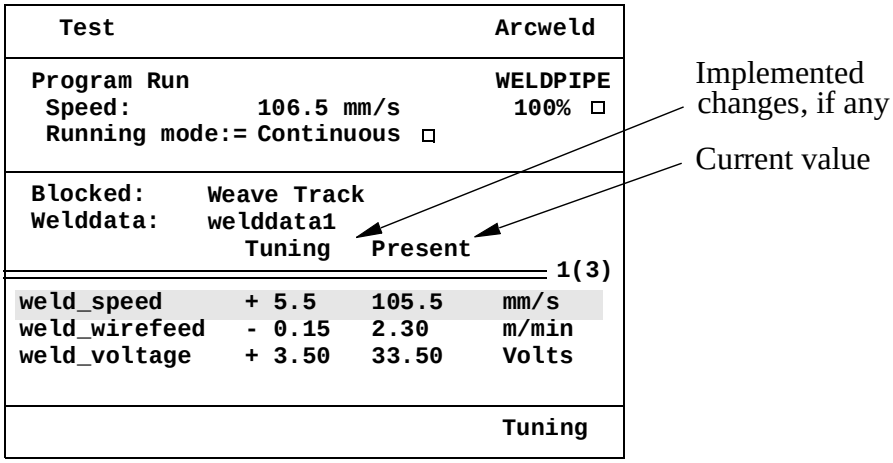


Figure 16 Window for tuning welding data, during program execution.

Selecting data type to be tuned

• Select **Arcweld** and enter the desired type using the number keys.

Selection of the type of data, of the type chosen, to be displayed is done automatically during program execution as follows:

- When welding is in progress, the data in use is displayed.
- When welding is not in progress, the latest data used is displayed.

Tuning data type

• Press ***Tuning.***

The following window is shown (see Figure 17).

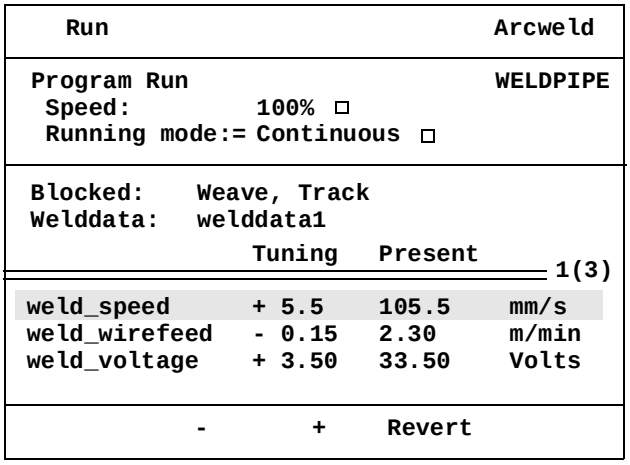


Figure 17 Window for tuning data type.

• Select what you want to tune, in the lower part of the box, and press the function keys "**-**" or "**+**" to reduce/increase the value.

Resetting the current tuning value

• Press ***Revert*** to reset the tuning value.
The current value is reset to the original value.

When the welding data displayed is no longer active, the ***Tuning*** window will disappear automatically. This prevents the unintentional modification of data when changing data between two instructions. ***Tuning*** is interlocked, which is indicated by parenthesis ( ) around the function button text.

If any part of the process has been blocked, it is not possible to tune the parameters that affect the part of the process that is blocked.

3.4 Changing the tuning increments

• Start with the *Program Test* window.

• Select **Arcweld: Increments**.

The following window is shown (see Figure 18):

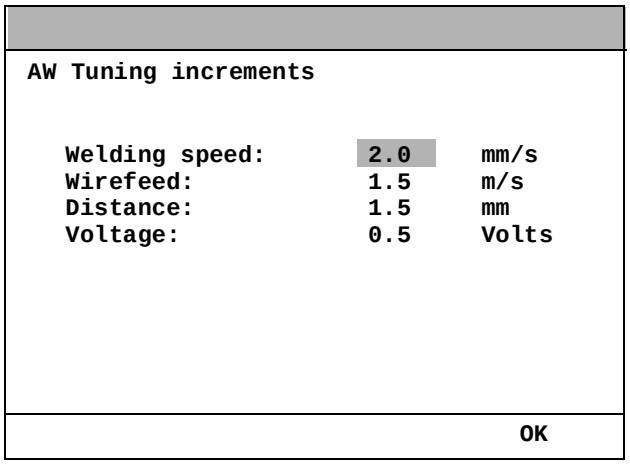


Figure 18 Dialog box for changing tuning increments.

• Move to the desired field using the arrow keys.

• Change the value using the number keys.

• Choose ***OK*** to terminate the dialog.

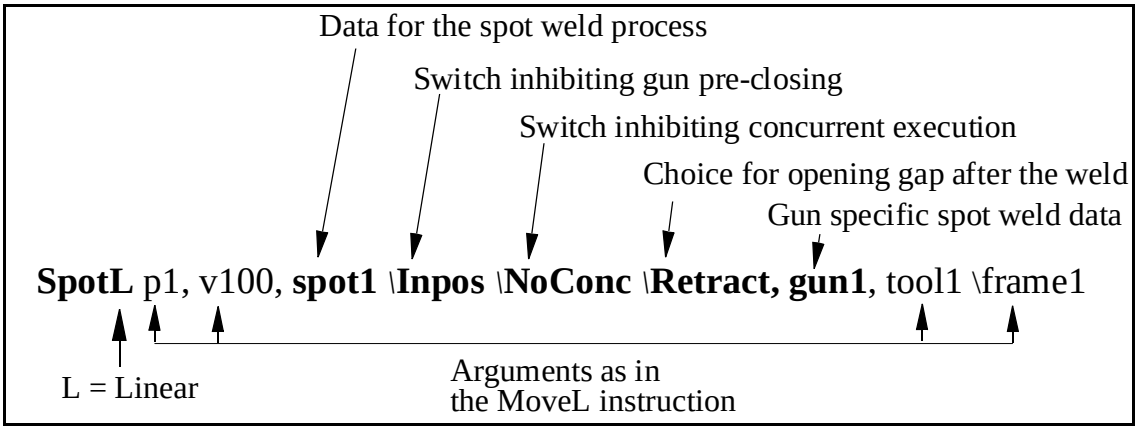
Programming SpotWare

The SpotWare package is added to the base system and is ready to use. However, in order to be able to adapt the package to special needs, the I/O configuration and application specific program data and routines can be modified. See *Customizing the spot weld instruction* on page 20.

4 Programming Spot Welding

4.1 Spot weld instructions

A spot weld instruction comprises both motion and process. It contains basically the same types of information as a positioning instruction. However, each spot weld instruction includes further arguments, *spot* and *gun*, that serve as data for the spot weld process (data types: *spotdata* and *gundata*).



Each *SpotL* instruction includes a movement to the weld position and one weld.

Since the spot weld is always carried out at a stop point the zone argument is omitted.

Note. gun1 and tool1 contain data for the same tool, i.e. the spot weld gun, where tool1 contains the motion-specific data known from the MoveL instruction.

4.2 Defining spot weld data

Before starting to program the instructions, you should define the spot weld data that is to be used. This data is divided into two types:

- *spotdata*; describes the spot weld process-specific data
- *gundata;* describes spot weld gun setup.

There is already data defined in the spot weld system module SWUSER. For SpotWare Plus the corresponding system module is SWURSF. If the new data that is being defined belongs to the spot weld application, it should also be stored in SWUSER/ SWUSRF (SpotWare Plus).

4.3 Manual gun control

The gun control is done by digital outputs.

To ensure fast access to the gun, design your own I/O preference list with the gun control on top.

SpotWare

close_tip1 / *close_tip2* for closing and opening the gun

work_select for choosing the opening gap (work=small=1/retract=big=0)

SpotWare Plus

The gun is controlled by actions executing the user defined routines *sw_close_gun / sw_open_gun*. These routines use the internal gundata of the last SpotL instructions. Use *man_gun* and *man_retr* in the system module SWUSRC to change this internal gundata.

SwOpenGun / SwCloseGun for closing and opening the gun.

SwNewData for transferring the contents of man_gun to the internal gundata.

4.4 Manual weld (SpotWare Plus)

A complete weld, including supervision and error recovery, may be run independently of the positioning. It is activated by a digital output triggering the attached action. The same principle as for manual gun manipulation is used, i.e. use *man_spot* for transferring data.

SwRunProc for executing a spot weld including gun open/close.

4.5 Programming spot weld instructions

- Jog the robot to the desired destination position.
- Call up the instruction pick list by choosing **IPL1: Motion&Process**.
- Select the instruction *SpotL*.

The instruction will be added directly to the program. The arguments are set in relation to the last spot weld instruction. (See Figure 19).

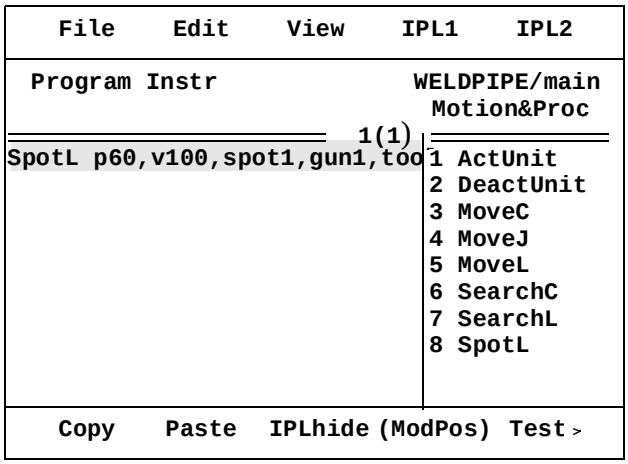


Figure 19 A spot weld instruction is added directly to the program.

• Change the arguments if necessary.

4.6 Testing a spot weld instruction

To prevent the spot weld process executing during programming, it is possible to run *SpotL* in simulation mode.

This can be done by setting *sw_inhib_weld* TRUE. This will set the output *current_enable* low and the simulation will be carried out by the weld timer.

If such a signal is not connected the spot weld can be internally inhibited by setting *sw_sim_weld* TRUE. The simulated weld time is then stored in *sw_sim_time*. In this simulation mode the start signal is never sent to the welding timer.

Note. The gun will be opened and closed as normal.

5 Defining gun closing times

The *SpotL* instruction has a built-in pre-closing of the weld gun, i.e. when approaching the position the gun will start to close in advance, in order to save time. The *pre_closing* relates to closing the gun from the work stroke. Closing the retract stroke should be handled in the user program, by a fly-by position.

The gun closing times are defined in *gundata*. It is possible to define a closing time for each gun pressure (max. 4).

The pressure to use in the actual *SpotL* instruction is stored in *spotdata*. It will also pick the accurate closing time.

Note. The pre-closing can be disabled by choosing the \InPos argument in the instruction or by setting *sw_inpos* TRUE in SWUSER / SWURSF (SpotWare Plus) (global disabling).

Defining the closing time manually

- First, define a rough guess in the actual gundata
- Look up the pressure in spotdata
- Move to the test window
- Choose execution mode *Instruction*
- Run *SpotL* forward and check the closing. The gun should be just about closed when the robot stops in the position.
- If the result is not OK, execute *SpotL* backwards. The gun will open automatically before it moves.
- Adjust the corresponding closing time.
- Repeat the test until the result is OK.
- Repeat the whole sequence for a different pressure and different stroke until all closing times are defined.

Defining the closing time automatically

It is possible to measure and store the closing times using a RAPID program.

6 Customizing the spot weld instruction

Customizing can be done at different levels:

- SWUSER / SWUSRF, SWUSRC: Modules containing routines and data used by *SpotL*
- I/O configuration.

SWUSER

Global data, service and supervision routines affecting the entire program.
(See Predefined Data and Programs).

SWUSRF (SpotWare Plus)

Global data, service and equipment routines affecting the entire program.
(See Predefined Data and Programs).

SWUSRC (SpotWare Plus)

Global data, service and equipment routines affecting the entire program.
(See Predefined Data and Programs).

I/O configuration

Mirroring the connected equipment.
(See System Parameters).

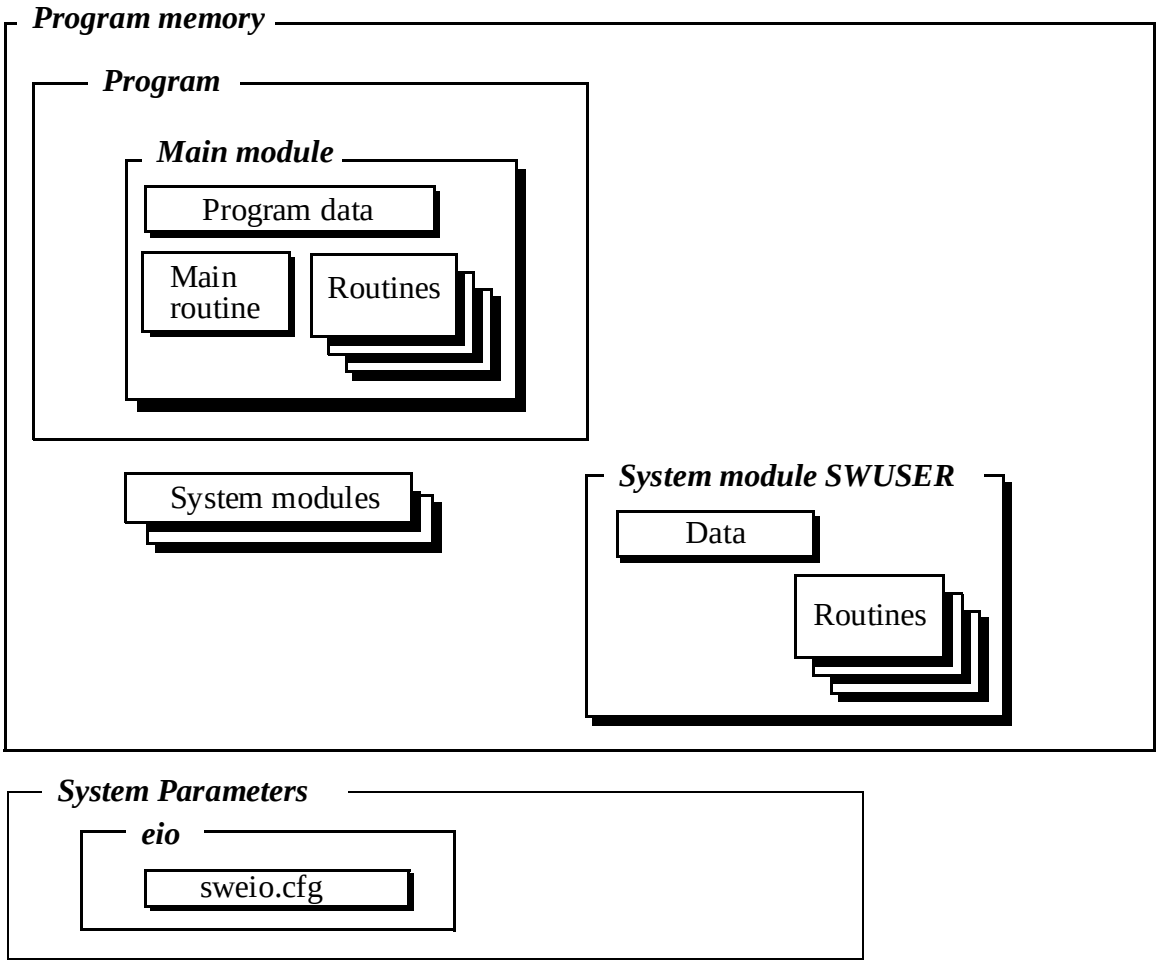


Figure 20 The SpotWare environment.

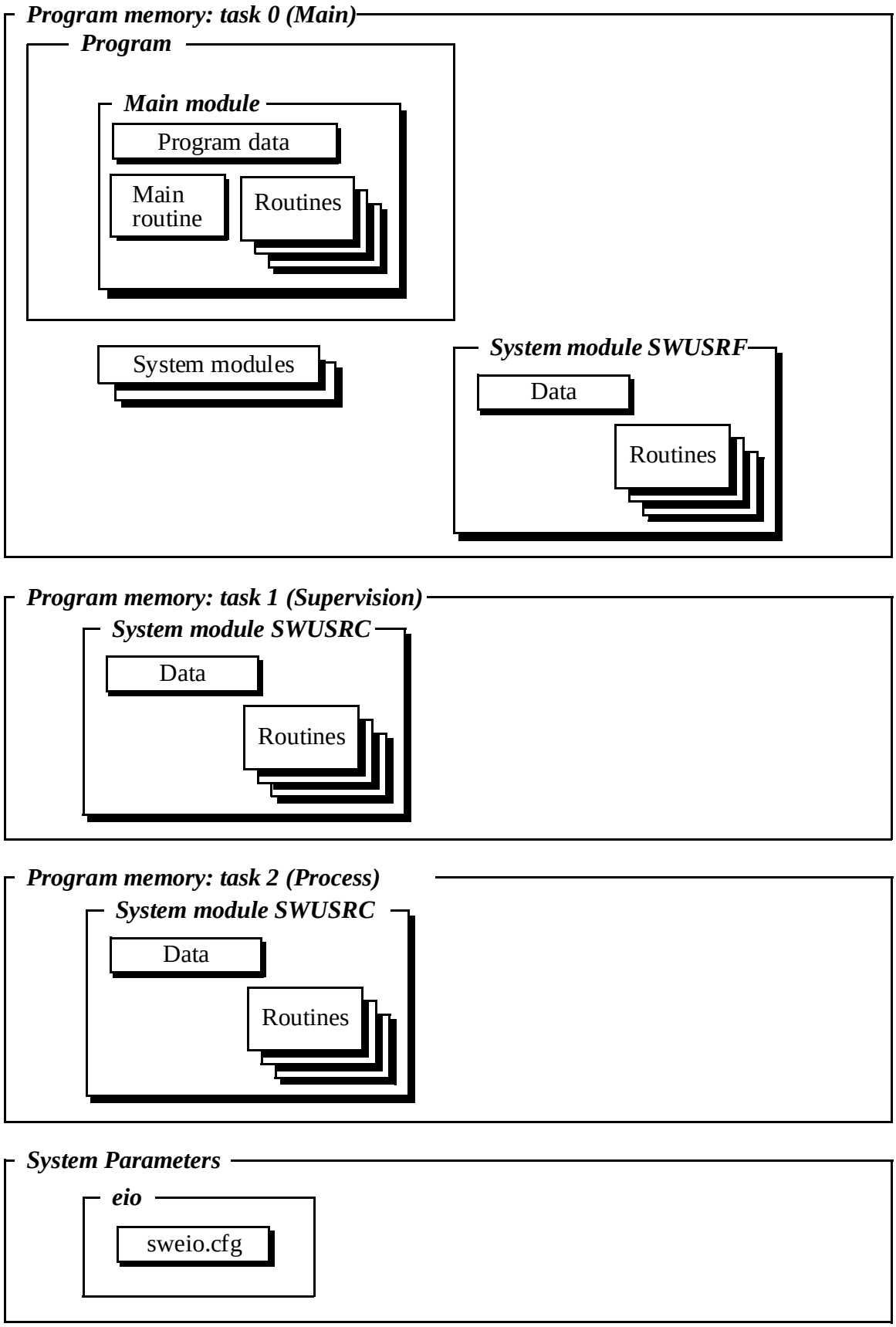


Figure 21 The SpotWare Plus environment.

7 Running spot weld instructions

In continuous mode the spot weld instructions are by default running in concurrent execution, i.e. the program execution is continuing whilst welding is in progress, and it will not stop until the next motion instruction.

Therefore if the running is stopped during welding, the program pointer is normally moved to next *SpotL* or motion instruction. This fact is important to remember when *SpotL* positions are modified with ***ModPos***.

To avoid problems it is recommended to execute the program in step-by-step mode when positions are modified. In this mode the program pointer is in agreement with the robot position. If a strictly sequential execution is desired also in continuous mode the *NoConc* flag must be set in the instruction.

8 Tip-dressing

The gundata contains counters and limits for each pair of tips. The counters will be incremented for each spot.

A tip-dress supervision can however be programmed in the main program or in the *SpotL* supervision routines.

In this way, an automatic tip-dressing can be performed.

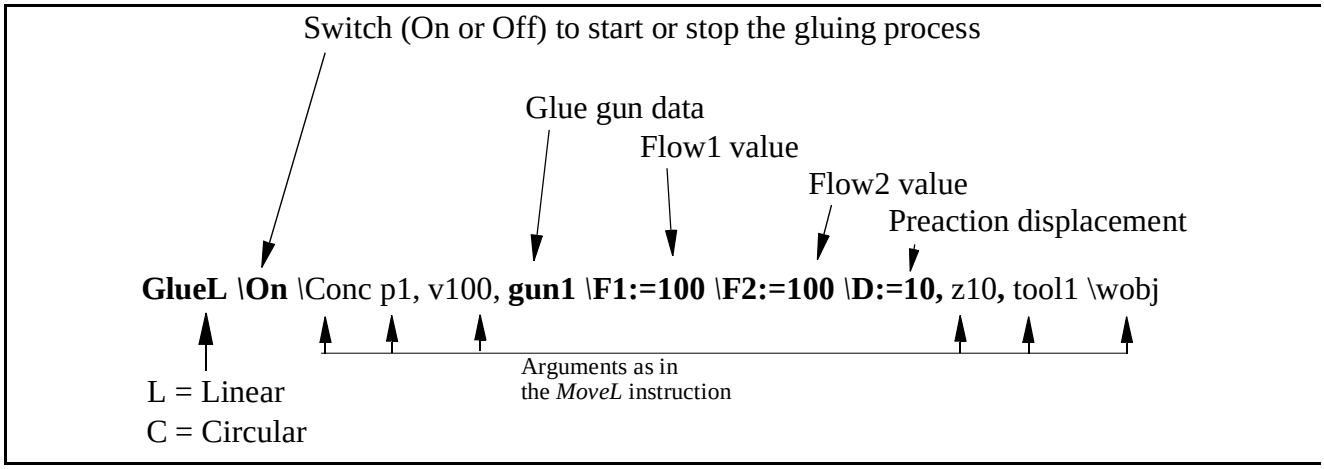
Programming GlueWare

The GlueWare package is added to the base system and is ready to use. However, in order to adapt the package to special needs, the I/O configuration and application specific program data and routines can be modified. See *Customizing the glue instruction* on page 27.

9 Programming Glueing

9.1 Glue instructions

A glue instruction includes both motion and process. It basically contains the same types of information as a positioning instruction. However, each glue instruction also includes further arguments for the glue process.



NB: *gun1* and *tool1* are containing data for the same tool i.e. the glue gun where *tool1* contains the motion specific data known from the *MoveL* instruction and *gun1* contains process specific data.

9.2 Glue data

Before starting to program the instructions, you should define the *ggundata* (glue gun data) that is to be used.

Ggundata has the following structure:

- Which gun is to be used (1 or 2).
- Time for preopening and preclosing the gun.
- Type of flow1 and flow2.
- Times for presetting of flow1 and flow2 on.
- Times for presetting flow1 and flow2 changes.

- Times for presetting of flow1 and flow2 off.
- Times for lag in glue gun for speed dips.
- Value for the speed, at which the logical maximum value for the analog outputs shall be set.

There are already data defined in the glue system module GLUSER. If new data shall that is being defined belongs to the glue application, it should also be stored in GLUSER.

9.3 Programming glue instructions

• Jog the robot to the desired destination position.

• Call up the instruction pick list by choosing **IPL1: Motion & Process**.

• Select the desired instruction *GlueL, GlueL/On, GlueL/Off or GlueC*.

The instruction will be added directly to the program. The arguments are set in relation to the last programmed glue instruction.

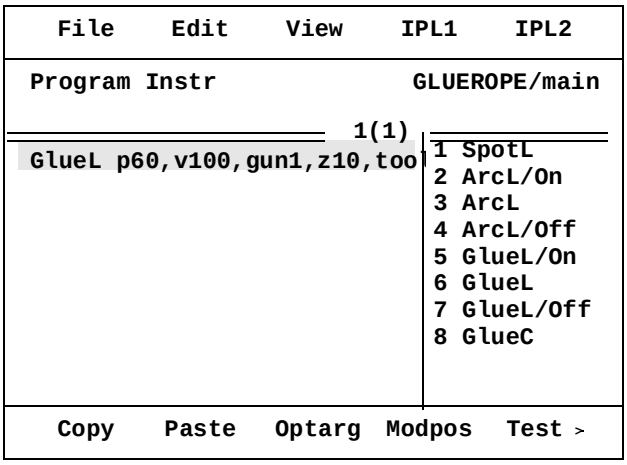


Figure 22 A glue instruction is added directly to the program.

• Change the arguments if necessary.

10 Testing glue instructions without gluing.

There are two possibilities to run glue instructions without gluing.

• Run the program stepwise forward or backward.

• Run the program in simulation mode. This can be done by setting *gl_sim_glue* to TRUE. Choose **View: DataTypes-bool**.

11 Manual gun control

The gun control is done by digital and analog outputs.

To have a fast access to the gun make sure to design your most common I/O list with the gun control on top.

DO_Ggun1 for closing and opening the gun1.

AO_G1Flow1 for setting the flow1 values.

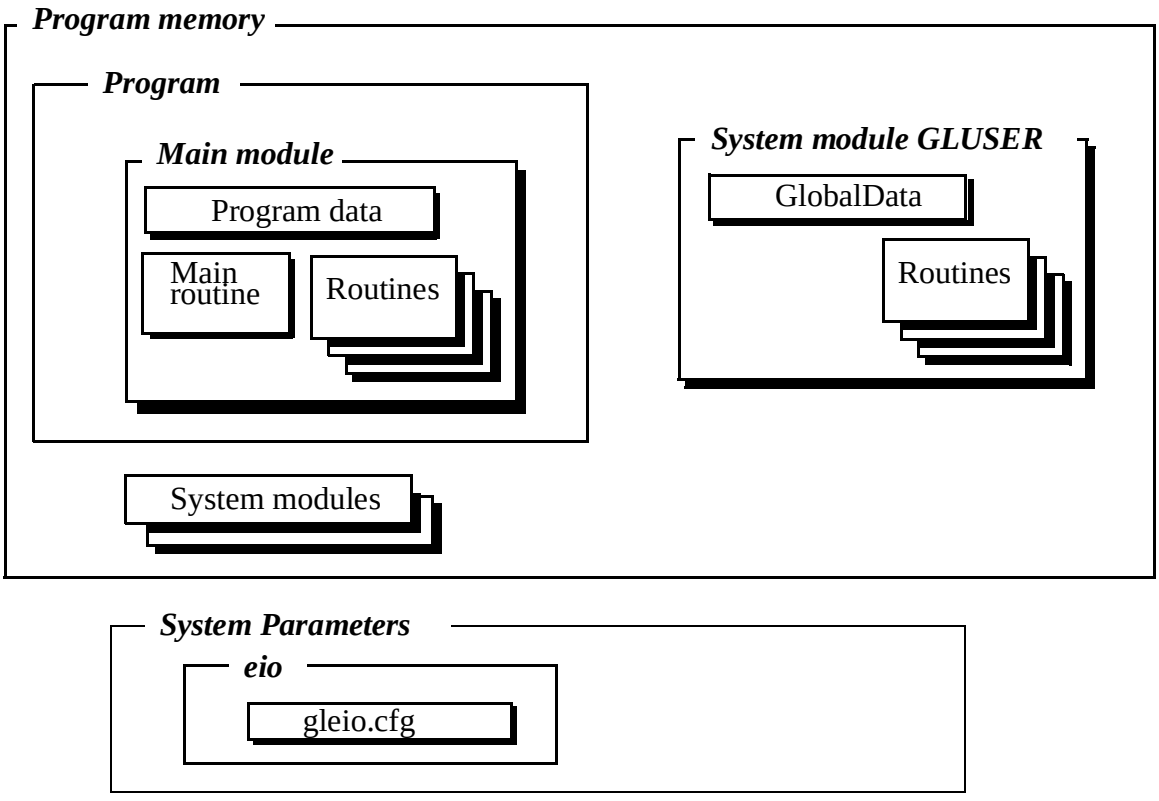
AO_G1Flow2 for setting the flow2 values.

If two guns are used, define also desired signals for gun two in the list.

12 Customizing the glue instruction

The customizing can be done on different levels.

- Instruction parameters
- GLUSER: Module containing routines and data used by the glue instructions.
- I/O configuration



Instruction parameters

Data and switches are affecting the actual instruction as well as the following ones, if these are not using new data.

GLUSER

Global data and routines affecting the entire program (see RAPID Reference Manual, *Predefined Programs and Data*.

I/O configuration

As default all Glue outputs are defined but connected to a simulated board. Used glue outputs have to be connected to physical signals. Not used signals shall remain on the standard simulated board (*see System Parameters*).

CONTENTS

Page

1 Arc Welding

The arc welding topic contains parameters that define the arc welding functions:

- The units used when the parameters are entered
- The process functions used
- The equipment used
- The weldguide sensor used

• Choose **Topics: Arc Weld**.

1.1 Activating arc welding parameters

Note When an arc welding parameter is activated, the program memory will be erased. So, make sure that you have saved all programs on diskette.

Make sure that all input and output signals referenced by the Arc Weld configuration are defined in the topic *IO Signals*, before any parameters are activated.

Arc welding data (*seamdata*, *welddata* and *weavedata*) is always adapted to the current configuration. If, for instance, an analog output is defined to control the weld voltage, the *Voltage* component will be included in the *welddata*. Changing these parameters will thus also affect the RAPID program as a whole, which may mean that any programs created with a different set of parameters cannot be run. Activating this change requires a special procedure.

It is not always necessary to convert the AW data from one configuration to another. When opening a program, mismatching AW data in the program being opened is always pointed out.

The RAPID converter is a tool to prepare a RAPID program in the current configuration for later use, and later also to make it usable again in any configuration.

In order to be able to run old programs, they must be stored in a special format:

• Open the FileManager window.

• Select the program to be converted.

• Start conversion by using the command **Options: RAPID Converters**.

• The program will be stored in the new format with the extension XRG (instead of PRG). This XRG file is to be used in any later configuration to re-shape a PRG file, i.e. a working RAPID program.

• Change the parameters as desired.

• Perform a normal warm start by choosing **File: Restart**.

• Choose **File: Restart** in the *Service window*.

• When the confirmation dialog appears, enter the digits 2, 5 and 8.

• The text above the fifth function key will then change to ***P-start***. Activate the parameters by pressing this key.

• Convert the program back to the normal format again, using the command **Options: RAPID Converters**, in the FileManager as described above.

1.2 Defining arc welding systems

Up to 5 arc welding systems can be activated simultaneously in the same robot installation. This may be required when, for example:

- more than one piece of process equipment is connected;
- two different electrode dimensions are used (different feeding systems must be used for this to happen);
- more than one process is used, e.g. TIG and MIG/MAG.

If more than one arc welding system is defined, a new set of instructions and data types is activated for each system. The first additional system (which is defined by the sequence of defined system in a configuration file) is connected to instructions and data types with the suffix 1, and the second to the suffix 2. In other words, *ArcL1* and *ArcC1* would be connected to *seamdata1*, *welddata1*, *weavedata1*.

• Choose **Topic: Arc Weld**.

• Choose **Types: Arc Weld System**.

All defined arc welding systems will be displayed, as shown in Figure 1.

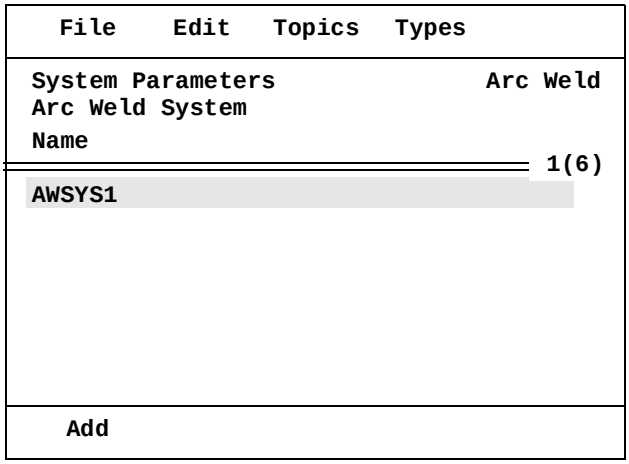


Figure 1 All defined arc welding systems are displayed.

• Select the arc welding system to be changed and press Enter ↵, or add a new one by pressing ***Add***.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
name[1]	The name of the system (max. 16 characters).
units[1]	The name of the group of measurements to be used. These groups are defined in *Defining measurements* on page 5.
functions[1]	The name of the group of functions to be used. These groups are defined in *Defining arc welding functions* on page 6.
equipment[1]	The name of the equipment to be used. This is defined in *Defining arc welding equipment* on page 9.
weldguide	The name of the weldguide definition to be used. This is defined in *Defining weldguide arc welding sensor* on page 12.

1.3 Defining measurements

The welding speed, dimensions and the like can be specified in different units. These are grouped together under a common name and coupled to the appropriate arc welding system.

- Choose **Topic: Arc Weld**.
- Choose **Types: Arc Weld Unit**.

All defined groups of measurements (max. 5) will be displayed, as shown in Figure 2.

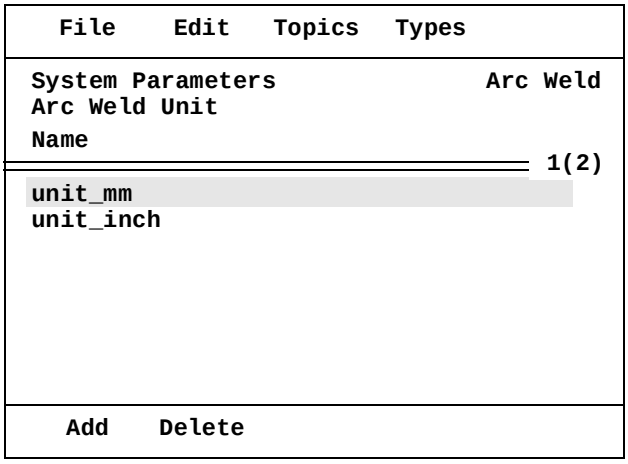


Figure 2 All groups of measurements are displayed.

- Select the group of units to be changed and press Enter ↵, or add a new one by pressing ***Add***.
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

1. These parameters must always be defined.

Parameter	Description
name[1]	The name of the group (max. 16 characters).
speed[1]	The desired unit for speed of welding (m/min, mm/s, ipm or cm/min).
length[1]	The desired unit for distance and dimensions (mm or inch).
wire feed[1]	The desired unit for wire feed (m/min, mm/s or ipm).
	Note: Scaling between a logical and a physical value on an analog output signal, is always expressed in m/s.
	Example: If a wire feed of 1m/min. shall be equivalent to a value of 10 V on the analog output signal, the scaling shall be expressed as Logical Max= 60 (m/s) Physical Max= 10 (V).
	See Chapter 12, System Parameters.

1.4 Defining arc welding functions

To obtain the desired arc welding functionality, such as the restart and crater fill functions, the appropriate functions must be activated in the system parameters. The functions are grouped together under a common name and then connected to the appropriate arc welding system.

• Choose **Topic: Arc Weld**.

• Choose **Types: Arc Weld Function**.

All defined function groups (max. 5) will be displayed, as shown in Figure 3.

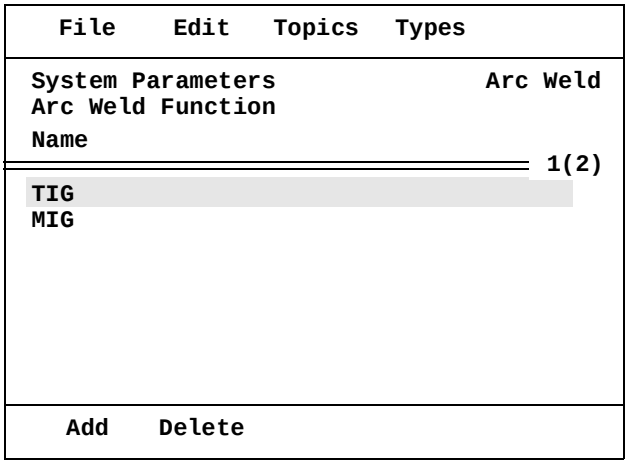


Figure 3 All groups of functions are displayed.

1. These parameters must always be defined

• Select the group of functions to be changed and press Enter ↵, or add a new one by pressing ***Add***.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
name[1]	The name of the function group (max. 16 characters).
restart	Specifies whether the weld is to be restarted in the event of a welding defect. This restart can be done in three different ways: - Automatically (the number of times specified in the parameter ***no of retries***) - Program controlled (using the routine's error handler) - Manually (when the error has been remedied, the program can be started in the normal way). If a restart is chosen (yes), the robot automatically reverses to a position before the position where it was interrupted (as specified in the parameter ***restart distance***). The weld is then restarted and program execution continues.
restart distance	The distance that the robot must reverse (relative to the position where it was interrupted) before a restart will take place (0–150 mm).
regain speed	The speed at which the robot moves to the position where it was interrupted.
no of retries	The number of automatic restart attempts.
scrape	Specifies if the robot is to weave at the actual weld start (scrape start). This weaving is automatically interrupted when the arc is ignited.
opt ignition scrape	Specifies scrape type at weld start. This function will not affect the scrape type at restart. The scrape types are specified in *seamdata*.
scrape width	The width of the weave pattern for a scrape start.
scrape direction	The angle of direction of the weave for a scrape start. It is specified in degrees, where 0^o implies a weave that is carried out at right angles to the direction of the weld.
scrape cycle time	The time (in seconds) it takes for a complete weave cycle for a scrape start.
ignition data	Specifies if ignition data is to be used at the start of the weld phase. At the start it is often beneficial to define higher weld data values for a better ignition. The values to be used are specified in *seamdata*. If the ignition data parameter is changed, the contents of *seamdata* will also change and therefore it must be activated as in *Activating arc welding parameters* on page 3.

1. These parameters must always be defined

preheating Specifies whether preheating is to be used at the start of the weld phase.

When the arc is ignited, the seam will generally not have reached the correct temperature. Preheating can thus be used at the start of the weld to define higher weld data values. The values to be used are specified in *seamdata*.

If the preheating parameter is changed, the contents of *seamdata* will also change and therefore it must be activated as in *Activating arc welding parameters* on page 3.

weave Specifies whether a weave pattern is to be added to the basic path. In this case, the weave starts when preheating is started at the beginning of the weld phase. The way the weave is to be carried out is described in *weavedata*.
If coordinated interpolation is used, a notch filter for the external axis may have to be used. See Chapter 12, System Parameters.

weldguide Specifies whether Weldguide is to be used.

weave sync Specifies whether synchronisation pulses are to be transmitted at the end positions of the weave.

track Specifies that a tracker is connected to the senor interface.

crater fill Specifies whether a crater fill is to be used in the final phase. This means that the end crater that can form in the completed weld will be filled in with extra filler material. Exactly how the crater fill is to be carried out is described in *seamdata*.

If the Crater fill parameter is changed, the contents of *seamdata* will also change and therefore it must be activated as in *Activating arc welding parameters* on page 3.

burnback Specifies whether burnback is to be used in the final phase. It is used in MIG/MAG welding and means that the power supply switches on for a short while after the electrode feed has been turned off. The end of the weld electrode is then melted and transferred to the molten metal in the weld deposit. In this way, the electrode will separate from the molten metal and not stick to it when it starts to harden. Exactly how the burnback is to be carried out is described in *seamdata*.

If the Burnback parameter is changed, the contents of *seamdata* will also change and therefore it must be activated as in *Activating arc welding parameters* on page 3.

rollback Specifies whether rollback is to be used in the final phase. It is used in TIG welding and means that the cold wire is reversed before the molten metal hardens, to prevent the wire sticking. Exactly how the rollback is to be carried out is described in *seamdata*.

If the Rollback parameter is changed, the contents of *seamdata* will also change and therefore it must be activated as in *Activating arc welding parameters* on page 3.

override Specifies whether override is to used.

precond	Specifies whether preconditions is to be used. If precond is on, the gas supervision and water supervision signals are verified before welding is started.
auto inhib	Specifies whether inhibition will be allowed in AUTO-mode or not.

1.5 Defining arc welding equipment

The equipment used in arc welding must be defined. In addition, the signals must be connected to the desired arc welding functions. These are grouped together under a common name and then connected to the appropriate arc welding system.

- First, define the names of the signals to be used for communication between the robot and welding equipment. See Chapter 12, System Parameters.
- Choose **Topic: Arc Weld**.
- Choose **Types: Arc Weld Equipment**.

All defined equipment (max. 5) will be displayed, as shown in Figure 4.

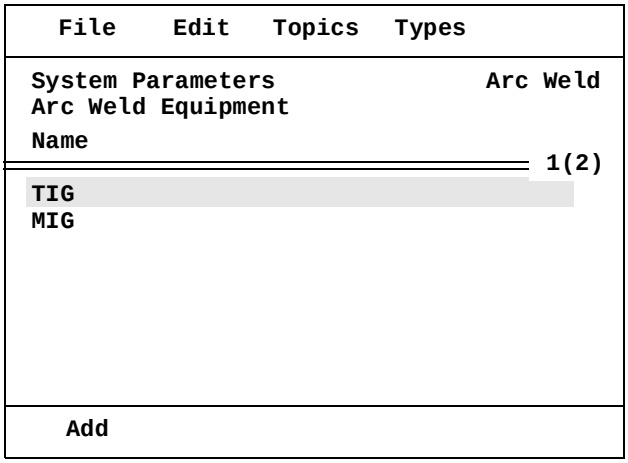


Figure 4 All defined equipment is displayed.

- Select the equipment to be changed and press Enter ↵, or add new equipment by pressing ***Add***.
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
name[1]	The name of the equipment (max. 16 characters).
manual wirefeed	Digital input signal for manual wire feed (active *High*).

1. These parameters must always be defined

weld inhibit	Digital input signal for program execution without welding (active *High*).
weave inhibit	Digital input signal for program execution without weaving (active *High*).
superv inhibit	Digital input signal for program execution without any supervision. (active *High*).
stop process	Digital input signal for stopping program execution. This signal only affects arc welding instructions. A high signal means that program execution will stop as soon as an arc welding instruction is executed.
ign timeout	The maximum time (in seconds) permitted for igniting the welding arc.
arc ok delay	The time it takes the welding arc to stabilise at the start of a weld. The arc is only considered ignited after the ***arc supervision*** signal has been high for the specified number of seconds.
arc supervision[1]	Digital input signal for supervision of the welding arc. A high signal means that the welding arc is ignited.
volt supervision	Digital input signal for supervision of the voltage. A high signal means that the voltage is OK.
current supervision	Digital input signal for supervision of the current. A high signal means that the current is OK.
water supervision	Digital input signal for supervision of the cooling water. A high signal means that the cooling water is OK.
wire supervision	Digital input signal for supervision of the wire feed. A high signal means that the wire feed is OK.
gas supervision	Digital input signal for supervision of the protective gas. A high signal means that the protective gas is OK.
torch supervision	Digital input signal for supervision of the torch. A high signal means that the torch is OK.
wirestick status	Digital input signal for supervision of the wire stick status. A high signal means that an error has occurred.
arc weld status	Digital output signal for indication of welding defects. A high signal means that an error has occurred. If a normal program stop occurs in the middle of a weld, no high signal will be generated.
gas control	Digital output signal for control of the gas flow (active *High*).
power control[1]	Digital output signal for control of the weld voltage (active *High*).
arc preset time	Delays the ***power control*** signal. This allows the analog reference signals enough time to stabilise before the weld is started.

1. These parameters must always be defined

wirefeed	Digital output signal for activation of the wire feed (active *High*).
wirefeed backward	Digital output signal for backward activation of the wire feed (active *High*).
weldschedule port	Group of digital output signals used to transfer program numbers to the welding equipment.
schedule strobe	Digital output signal used for handshaking if, when program numbers are transferred to the welding equipment, ***schedule port type*** has been defined as ***Pulse*** (active *High*).
schedule port type	Type of port used to transfer program numbers to the welding equipment: ***Binary*** = Binary-coded group of digital output signals ***Pulse*** = Program numbers are sent in the form of a number of pulses on the Weldschedule port signal which should then comprise two digital signals. They are pulsed in tens on one of the outputs and in ones on the other. ***CAN*** = Weldschedule number written on the CAN bus (dedicated for the Arcitec system).
weld voltage	Analog output signal for analog voltage reference.
voltage adjust	Analog output signal for relative adjustment of the analog voltage reference.
wire feed	Analog output signal for analog wire feed reference.
current adjust	Analog output signal for relative adjustment of the analog current reference.
process stopped	Digital output signal used to indicate that the weld has been interrupted. A high signal means that the weld has been interrupted either because of a welding defect or because of a normal program stop.
supervision arc	Digital output signal for indication of welding arc errors. A high signal means that an error has occurred.
supervision voltage	Digital output signal for indication of voltage errors. A high signal means that an error has occurred.
supervision current	Digital output signal for indication of current errors. A high signal means that an error has occurred.
supervision water	Digital output signal for indication of cooling water errors. A high signal means that an error has occurred.
supervision gas	Digital output signal for indication of protective gas errors. A high signal means that an error has occurred.
supervision torch	Digital output signal for indication of torch errors. A high signal means that an error has occurred.
supervision wirefeed	Digital output signal for indication of wire feed errors. A high signal means that an error has occurred.

1.6 Defining weldguide arc welding sensor

The weldguide used in arc welding must be defined. In addition, the signals must be coupled to the desired arc welding functions. These are grouped together under a common name and then coupled to the appropriate arc welding system.

- First, define the names of the signals to be used for communication between the robot and Weldguide. See Chapter 12, System Parameters.
- Define in which functions weldguide is to be used (parameter weldguide, weave sync). See *Defining arc welding functions* on page 6.
- Choose **Topic: Arc Weld.**
- Choose **Types: Arc Weld Weldguide.**

All defined weldguides (max. 5) will be displayed, as shown in Figure 5.

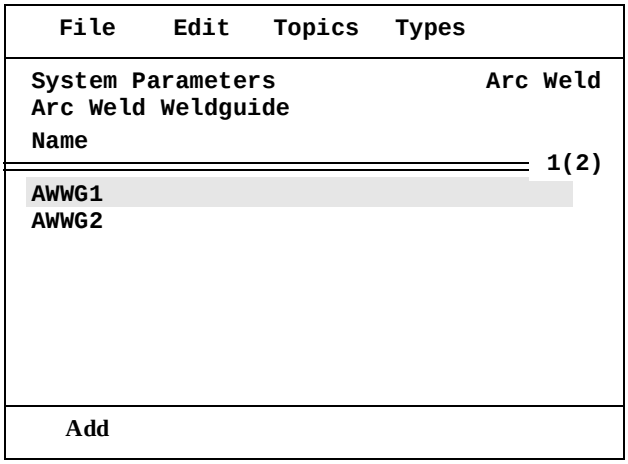


Figure 5 All defined weldguides are displayed.

- Select the weldguide to be changed and press Enter ↵, or add new weldguide by pressing ***Add***.
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
name[1]	The name of the weldguide sensor (max. 16 characters).
wg ready	Digital input signal indicating that the weldguide tracker is ready (active High).
wg inhib	Digital output signal for stopping the weldguide tracker (active High).
wg track	Digital output signal for starting the weldguide tracker (active High).

1. These parameters must always be defined.

wg left sync	Digital output signal indicating that the welding torch is at the left hand joint side of the weave pattern (active High).
wg right sync	Digital output signal indicating that the welding torch is at the right hand joint side of the weave pattern (active High).
wg height corr	Group of digital input signals for height corrections.
wg side corr	Group of digital input signals for side corrections.
wg data valid	Digital input signal indicating that weldguide is giving height and side corrections (active High).
wg data ack	Digital output signals for acknowledgement of data valid. The acknowledgement output pulse must be least 10 ms wide (active High).

2 Spot Welding

The spot welding system parameters are defined by input and output signals.

• Choose **Topics: IO Signals**

• Choose **Types: User Signals**

See Chapter 12, System Parameters.

2.1 IO configuration

The SpotWare package can be configured for different equipment setups. This chapter describes the signals used by SpotWare and their dependency of the equipment.

The physical connections can be freely changed. To save physical signals, signals not in use can be connected to a virtual board (type eip000).

If different signal names are required, new logical signals can be added and connected to the corresponding physical signal. The predefined logical name must remain in the configuration as it is used internally by the SpotL instruction.

2.2 Basic setup

There are two predefined boards:

-one physical, DSQC 223 board, named SW_BOARD, at address 1.

-one virtual, named SIM-BOARD.

The basic setup signals are connected to the SW_BOARD. All others are connected to the SIM_BOARD which makes them easy to activate.

Start and monitoring of the weld

start1	output 7	Start signal to the weld timer (Tip1 if a dual gun is used)

Weld program number

prog_no_group	output 11-16	Weld program number

N.B. Signals in the same group must be connected to physical signals in sequence on the same board.

Gun control

close_tip1	output 5	Close gun signal. (Tip 1, if a dual gun is used).
work_select	output 6	Work stroke select. Default functionality: The signal is set to 1 after the welding, if work stroke (small gap) is desired, and is set to 0 if retract stroke (large gap) is desired. When the SpotL instruction orders the gun to close, the work_select is also set. The signal is also set according to the Retract argument in the instruction.

Pressure setting and supervision

pressure_group	output 2-4	p2_req-p4_req

The SpotWare can set and supervise up to four discrete pressure levels. The pressure values are selected in spotdata.The number of possible pressure levels is defined for each gun by *nof_plevels* in gundata. The corresponding output signals *p2_req* to *p4_req* must be defined as consecutive physical outputs. When only one pressure level exists, no pressure request signals need to be connected.

nof_plevels	output	output group
1	none	pressure group (simulated outputs)
2	p2_req	pressure group (p2_req)
3	p3_req	pressure group (p2_req, p3_req)
4	p4_req	pressure group (p2_req, p3_req, p4_req)

When supervision of the ordered pressure value is required before welding is allowed to start (*close_request* = TRUE in gundata), this is done on one of the input signals *p1_ok* to *p4_ok* depending on the selected pressure value in spotdata.

Gun opening supervision

If *open_request* = TRUE in gundata, a check is made whether or not the gun is open before a motion is released.

Timer reset

reset_fault	output 1	Reset signal. Can be used to reset the welding controller after a weld error. The signal is pulsed with a user defined pulse length. (sw_reset_time1) before manual or automatic rewelding.

Process error

process_error	output 9	Operator request is set when an error situation occurs.

Current signal

current_enable	output 8	Current enable signal. Used for the weld inhibit function. See Predefined Data and Programs (sw_inhib_weld).

Equipment supervision

By default, the following input signals are tested in every welding position before the weld start. If desired, other user defined signals can be tested, before or after the weld-

ing process. (See also Predefined Data)

timer_ready	input 8	The timer is ready to weld.
flow_ok	input 9	Signal indicating problems with the water supply.
temp_ok	input 10	Signal indicating over-temperature.
current_ok	input 11	Signal indicating that the weld current exceeds the permissible tolerances.

2.3 Double gun

If a double gun is used it has to be indicated in gundata by *nof_tips* = 2. Tip_no in spotdata chooses the tip (tip_no = 12 or 21 closes both tips together). Close_tip1, work_select and close_tip2 must be defined as consecutive physical signals.

close_tip2	sim output	Close gun request, tip 2

2.4 Equipment control (SpotWare Plus only)

By default, the following input signals are set depending on the motor on state and the output signals current_enable and process_error. (See also Predefined Data)

weld_power	sim output	Possible to use for a weld power unit contactor
water_start	sim output	Possible to use to activate water cooling

Table 1 Default functionality of water_start and weld_power

Activators	motors on	0	1	0	1	0	1	0	1
	process_error	0	0	1	1	0	0	1	1
	current_enable	0	0	0	0	1	1	1	1
Result	water_start	0	0	0	0	0	pulse	0	0
	weld_power	0	0	0	0	0	1	0	0

2.5 Manual Actions (SpotWare Plus only)

The manual action signals are internally triggering RAPID routines to execute. The sig-

nal are defined on the SIM_BOARD.

The manual actions operate on local data. This data may be updated in three ways:

- Running SpotL - local data contains the contents of the SpotL-parameters.
- Set the output *SwNew Data* - local data contains the contents of the manual data *man_spot, man_gun* and *man_retr*.
- Using the utility routines *SwSetCurrSpot*, etc. (See also “Predefined data”.)

SwNewData	sim output	Transfers manual data to the manual actions
SwCloseGun	sim output	Close the gun, runs *sw_close_gun*
SwOpenGun	sim output	Open the gun, runs *sw_open gun*
SwRunProc	sim output	Runs complete spot weld including opening/ closing of the gun. This action may also be used to run a reweld when the process has stopped in a weld ready timeout situation.
SwSkipProc	sim output	Skip the ongoing action and be ready for a new action. This signal is also used by SpotWare for internal process abortion.

2.6 Process state (SpotWare Plus only)

The following signals give the information about the state of the SpotWare process:

proc_run	sim output	Process is executing
inhib_move	sim output	SpotL movement is blocked and waiting to be released.
weld_error	sim output	Set when a weld ready timeout has occurred. Reset when re-welding was successful or after skipping the current weld.

3 Gluing

The glueware parameters are defined by output signals:

• Choose Topics: **I/O Signals**

• Choose Types: **User Signals**

See *System Parameters*.

3.1 I/O Configuration

The GlueWare package can be configured to suit different types of equipment. This chapter describes the signals used by GlueWare and their dependency on the equipment.

The physical connections can be freely changed. To save physical signals, signals not in use can be connected to a virtual board (type eip000).

If different signal names are required, new logical signals can be added and connected to the corresponding physical signal. The predefined logical name must remain in the configuration as it is used internally by the GlueL instruction.

3.2 Basic setup

There is one predefined board:

- a simulated board named GL_SIMBOARD.

The basic setup signals are connected to GL_SIMBOARD. The signals in use must be connected to the desired physical outputs.

Output signals controlling the glue gun

DO_Ggun1	digital output 1	Signal for the gluing equipment to open gun 1 when the signal is set, or to close the gun when the signal is reset.
DO_Ggun2	digital output 2	Signal for the gluing equipment to open gun 2 when the signal is set, or to close the gun when the signal is reset.
AO_G1Flow1	analog output 1	Signal for setting the value of flow1 for glue gun 1.
AO_G1Flow2	analog output 2	Signal for setting the value of flow2 for glue gun 1.
AO_G2Flow1	analog output 3	Signal for setting the value of flow1 for glue gun 2.
AO_G2Flow2	analog output 4	Signal for setting the value of flow2 for glue gun 2

Error signals

DO_GL_Err	digital output 3	A high signal indicates any glue specific error when gluing is active.
DO_GL_OvrSpd	digital output 4	A high signal indicates that the calculated value for one analog output signal exceeds the logical maximum value.

Other signals

DO_GL_Active	digital output 5	A high signal indicates that the glue process is active. Used internally by the *GlueL* and *GlueC* instructions.

3.3 Analog output scaling

The analog output signals above are defined with the following default values:

Logical Min:	0
Logical Max:	1000
Physical Min:	0 V
Physical Max:	10 V

Normally, the user only needs to adapt the Physical Min and Max to the equipment in use. See *Calculation of the glue flow values* in RAPID Reference Manual, *Instructions - GlueL*

CONTENTS

1 Simple Material Handling

1.1 What the robot does

The robot takes parts to and from a machine, as in Figure 1.

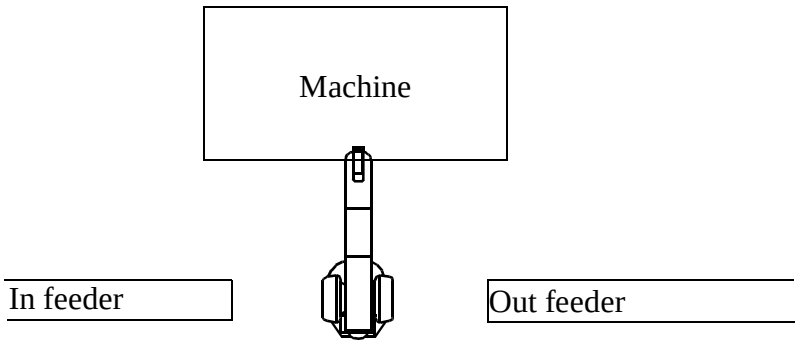


Figure 1 The robot gives a part to a machine which then processes it.

First, the robot fetches a part from the In feeder and places it in the machine where the part is processed. Then, when this has been done, the robot takes the part and places it on the Out feeder.

This work cycle is repeated until the operator stops production.

1.2 The main routine

The main routine is built up of a number of routine calls which reflect the robot work cycle.

Routine *main*	Comments
fetch_part;	Fetch part from In feeder.
leave_machine;	Leave the part in the machine.
process_part;	Start the actual processing.
fetch_machine;	Fetch the part.
leave_part;	Leave the part on the Out feeder.

1.3 Operating the gripper

The robot is equipped with a gripper that handles parts. A tool, *tool1*, and its associated tool centre point (TCP), is defined for this.

The tool is controlled by a digital output signal defined in the system parameters with the name *gripper*. A high signal indicates that the gripper is holding the part, and a low signal indicates that the part has been released.

In addition, a load data, *load1*, is defined which describes the load held by the gripper. The best possible motion performance is achieved if the correct load is always specified.

As the gripper grips and releases parts several times during the course of the program, it is best to set up separate routines for this which can be called by the program.

Routine *grip*	Comments
Set gripper;	Grip the part.
WaitTime 0.3;	Wait 0.3 s.
GripLoad load1;	Specify that there is a load in the gripper.

Routine *release*	Comments
Reset gripper;	Release the part.
WaitTime 0.3;	Wait 0.3 s.
GripLoad LOAD0;	Specify that there is no load in the gripper

1.4 Fetching a part from the In feeder

A part is fetched from the In feeder. As the robot cannot go straight from the previous position (Out feeder), it performs a joint movement to the first position. Then, it uses linear movement to achieve good path accuracy.

Routine *fetch_part*	Comments
MoveJ *, vmax, z50, tool1;	Go quickly to position near In feeder.
MoveL *, v1000, z30, tool1;	Go to position above part.
MoveL *, v200, fine, tool1;	Go slowly to grip position.
grip;	Grip part.
MoveL *, v200, z30, tool1;	Go to position above part.

1.5 Leaving the part in the machine

The robot leaves the part in the machine and then leaves that area so that the machine can be started.

Routine *leave_machine*	Comments
MoveJ *, vmax, z50, tool1;	Go quickly to position outside machine.
MoveL *, v500, z10, tool1;	Go to machine.
MoveL *, v200, fine, tool1;	Go to leave position.
release;	Release part.
MoveL *, v200, z30, tool1;	Go to position above part.
MoveL *, v500, z30, tool1;	Go to position above machine.

1.6 Starting to process

Processing starts when the robot pulses an output, *do1*. Then, using the input *di1*, the machine informs the robot that the part has been processed and can be fetched.

Routine *process_part*	Comments
PulseDO do1;	Pulse output to start machine.
WaitDI di1, 1;	Wait for the ready signal.

1.7 Fetching the part from the machine

The robot fetches the part from the machine.

Routine *fetch_machine*	Comments
MoveL *, v500, z10, tool1;	Go to machine.
MoveL *, v200, fine, tool1;	Go to fetch position.
grip;	Grip part.
MoveL *, v200, z30, tool1;	Go to position above part.
MoveL *, v500, z30, tool1;	Go to position outside machine.

1.8 Leaving the part on the Out feeder

The robot leaves the part on the Out feeder.

Routine *leave_part*	Comments
MoveJ *, vmax, z30, tool1;	Go quickly to position near Out feeder.
MoveL *, v500, z30, tool1;	Go to position above part.
MoveL *, v200, fine, tool1;	Go slowly to leave position.
release;	Release part.
MoveL *, v200, z30, tool1;	Go to position above part.

2 Material Handling

2.1 What the robot does

The robot takes parts to and from a machine, as in Figure 2.

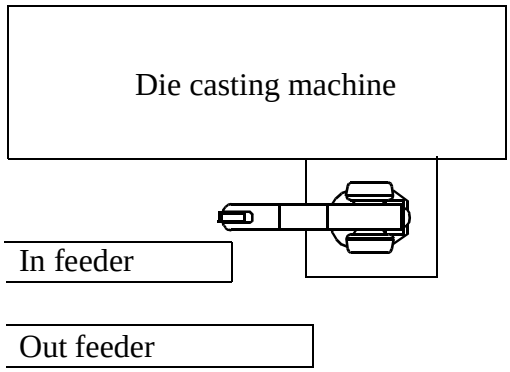


Figure 2 The robot serves a machine.

First, the robot fetches a part from the In feeder and places it in the machine. When the machine is ready, the robot grips the part and places it on the Out feeder.

The work cycle is repeated until the operator presses a push-button “Stop production”. The robot then completes the cycle, but does not fetch any new part from the In feeder.

The robot keeps a record of production statistics; it displays the number of parts produced during the day on the teach pendant and also, at the end of the work day, stores this information on a diskette that can be read using a PC.

2.2 The main routine

The main routine is built up of a number of routine calls which reflect the robot work cycle.

A digital input signal, *prodstop*, defined in the system parameters, is used to find out if the button “Stop production” is depressed. The button remains depressed until someone presses it again.

Routine *main*	Comments
start_production;	Initialise production for the day.
WHILE Dinput(prodstop) = 0 DO	Repeat the cycle until the button is pressed.
fetch_part;	Fetch the part from In feeder.
leave_machine;	Leave part in the machine.
process_part;	Start the actual processing.
fetch_machine;	Fetch the part.
leave_part;	Leave the part on Out feeder.
update_cycle;	Update operating statistics.
ENDWHILE	
stop_production;	Stop production for the day.

The routines *process_part*, *fetch_machine* and *leave_part* are not included in this example.

2.3 Operating the gripper

A tool, *gripper1*, defines the TCP and the weight of the gripper. This tool data is defined in the system module USER. In this way, the tool is always present in memory irrespective of which program is loaded.

The gripper is controlled by electric, bistable air valves, which means that there is one signal that controls the grip action and another that controls the release. The names of the signals are defined in the system parameters as *grip1* and *release1*. There is also a signal, *gripok*, that is high if a part is held by the gripper. This signal is used to check if the gripper has gripped a part correctly.

A load data, *payload*, is defined which describes the load held by the gripper. The best possible motion performance is achieved if the correct load is always specified.

As the gripper grips and releases parts several times during the course of the program, it is best to set up separate routines for this which can be called by the program. For example:

Routine *grip_part*	Comments
Reset release1;	
Set grip1;	Grip the part.
WaitTime 0.5;	Wait 0.5 s.
IF DInput(gripok)=0 THEN	If error (no part in the gripper) ...
TPWrite "ERROR: No part in the gripper";	Write error message on teach pendant.
EXIT;	Exit program execution.
ENDIF	
GripLoad payload;	Specify that there is a load.

The routine *release_part* is not included in this example.

2.4 Starting production

Before the actual production is started, the counter (*reg1*), which counts the number of parts that are produced during the day, is set to zero. The robot also goes into a home position.

Routine *start_production*	Comments
reg1 := 0;	Reset the counter.
MoveJ home, v500, fine, gripper1;	Go to home position.

In this example, all positions (e.g. *home* or *p1*) are named. They are stored as separate position data and can thus be reused in subsequent instructions. However, it is often just as easy to store the positions directly in the instructions (indicated by * in the instruction).

2.5 Fetching the part from the In feeder

Before fetching a part, the robot must check if there is any part to fetch. It does this by means of a photocell (via the *feeder* signal). This informs the robot if there is a part in position or not. If there is no part, the operator is sent a message and must first correct the error before starting program execution again.

Routine *fetch_part*	Comments
WHILE DInput(feeder) = 0 DO	Check if there is any part to fetch.
TPErase;	If not: Clear the teach pendant and
TPWrite "ERROR: No part on feeder";	write error message. Then wait until
TPWrite "";	the start signal is given by the opera-
tor.	
TPReadFK reg2, "Put part on feeder and press start", "Start", "", "","","",";	
ENDWHILE	
MoveJ p1, vmax, z50, gripper1;	Go quickly to position above part.
MoveL p2, v100, fine, gripper1;	Go to grip position.
grip_part;	Grip part.
MoveL p1, v200, z30, gripper1;	Go to position above part.

2.6 Leaving the part in the machine

The robot leaves the part in the machine and then leaves that area so that the machine can be started. Often, the robot and the machine communicate with one another to check such things as whether the machine is open. This check is not included in the following example.

Routine *leave_machine*	Comments
MoveJ p3, vmax, z50, gripper1;	Go quickly to position outside machine.
MoveL p4, v500, z10, gripper1;	Go in to machine.
MoveL p5, v100, fine, gripper1;	Go to release position.
release_part;	Release part.
MoveL p4, v200, z30, gripper1;	Go to position above part.
MoveL p3, v500, z50, gripper1;	Go to position outside machine.

2.7 Updating operating statistics

The number of parts produced during the day is written on the teach pendant display.

Routine *update_cycle*	Comments
reg1 := reg1 +1;	Increment produced parts.
TPErase;	Clear the display.
TPWrite "";	A few blank lines.
TPWrite "";	
TPWrite "No of produced parts = " \Num:=reg1;	The number of parts.

2.8 Stopping production for the day

If the operator presses “Stop production” and the robot has completed a work cycle, the robot goes to home position. In addition, the production figures for the day (the day’s date followed by the number of parts produced) are written on diskette.

Routine *stop_production*	Comments
MoveJ home, v500, fine, gripper1;	Go to home position.
Open "flp1:" \File:="logfile.doc", file\Append;	Open the file for writing.
Write file, CDate() \Num:=reg1;	Write to the file.
Close file;	Close the file.
Stop;	Stop program execution.

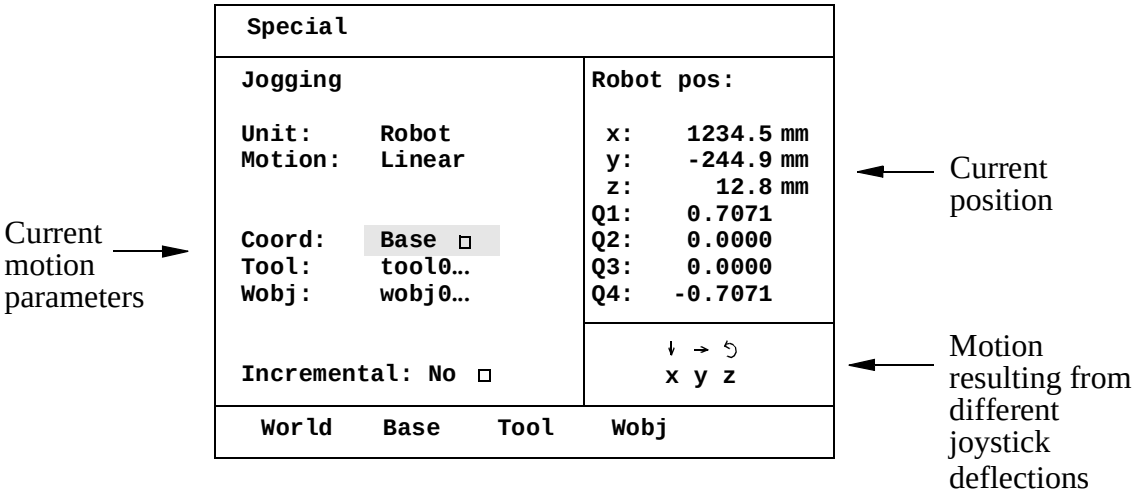
Before a file can be opened, the data, *file*, must be created by the type *iodev*. The real name of the file is *logfile.doc*.

CONTENTS

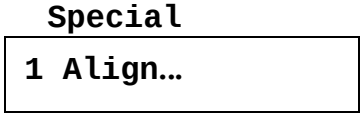
Page

1 The Jogging Window

1.1 Window: Jogging



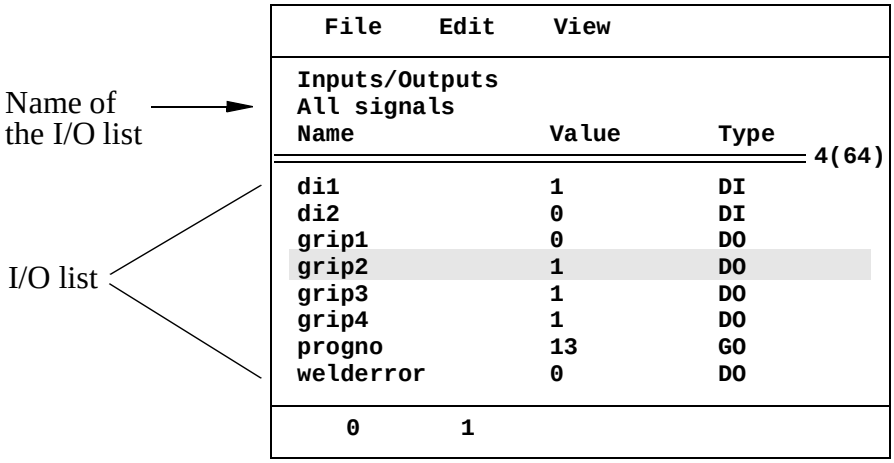
1.1.1 Menu: Special



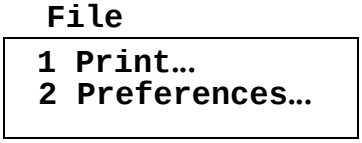
Command	Used to:
Align	Align the tool (see page 6-8)

2 The Inputs/Outputs Window

2.1 Window: Inputs/Outputs

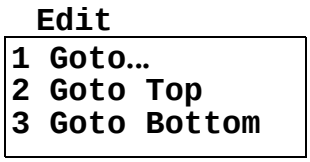


2.1.1 Menu: File



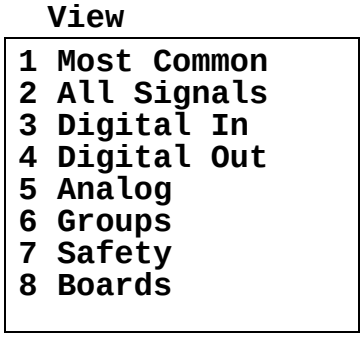
Command	Used to:
Print	print the current I/O list (see page 7-8)
Preferences	make preferences in the Inputs/Outputs window (see page 7-4)

2.1.2 Menu: Edit



Command:	Used to:
Goto	go to a specific line in the list
Goto Top	go to the first line in the list
Goto Bottom	go to the last line in the list

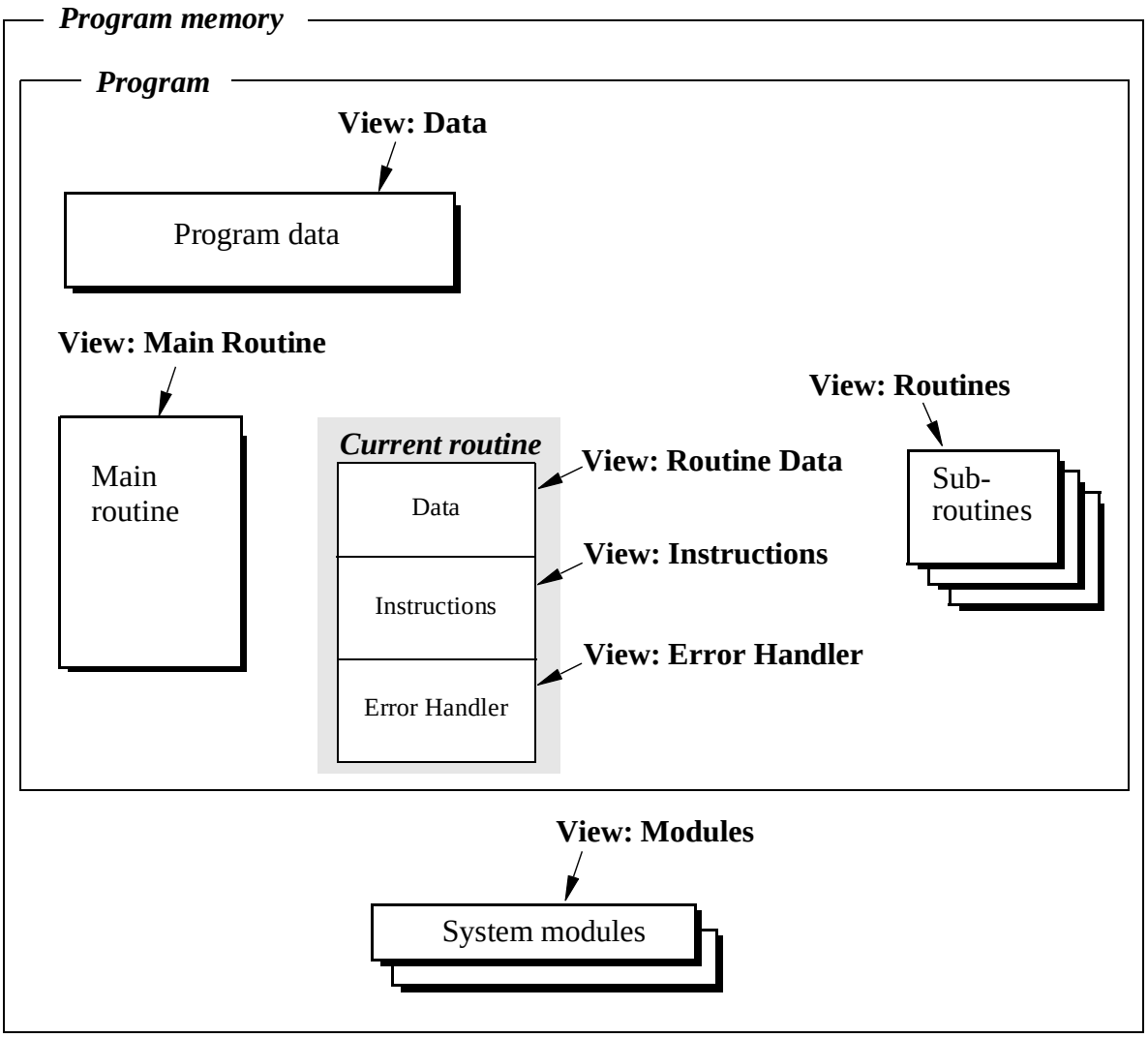
2.1.3 Menu: View



Command:	Used to view: (see page 7-4)
Most Common	the most common list
All Signals	all user signals
Digital In	all digital inputs
Digital Out	all digital outputs
Analog	all analog signals
Groups	all groups of digital signals
Safety	all safety signals
Boards	all I/O boards

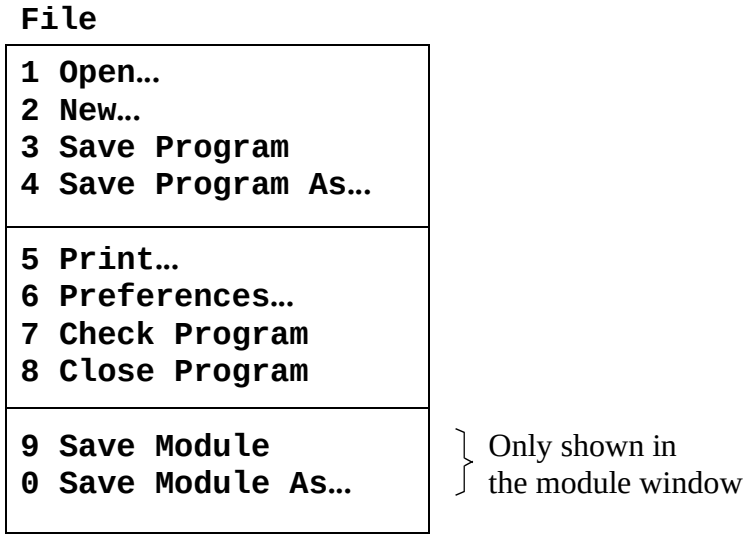
3 The Program Window

3.1 Moving between different parts of the program



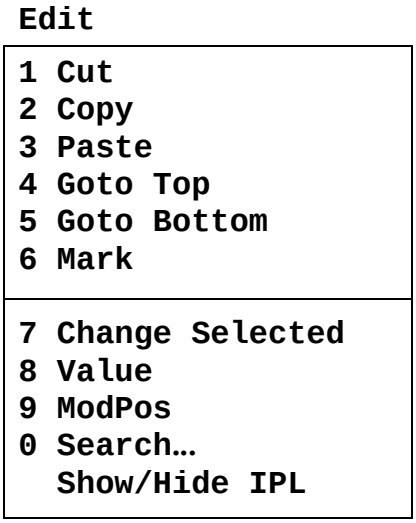
3.2 General menus

3.2.1 Menu: File



Command:	Used to:
Open	read programs from mass storage (see page 8-5)
New	create new programs (see page 8-4)
Save Program	save programs on mass storage (see page 8-25)
Save Program As	save programs on mass storage with new names (see page 8-25)
Print	print the program (see page 8-26)
Preferences	make preferences in the Program window (see page 8-56)
Check Program	check that the program is correct (see page 8-19)
Close Program	erase the program from the program memory
Save Module	save a module on mass storage (see page 8-54)
Save Module As	save a module on mass storage with a new name (see page 8-54)

3.2.2 Menu: Edit



Command	Used to:
Cut	cut selected lines to the clipboard buffer (see page 8-19)
Copy	copy selected lines to the clipboard buffer (see page 8-19)
Paste	paste the contents of the clipboard buffer into a program (see page 8-19)
Goto Top	go to the first line (see page 8-27)
Goto Bottom	go to the last line (see page 8-27)
Mark	select several lines (see page 8-28)
Change Selected	change an instruction argument (see page 8-30)
Value	show the current value (for the selected argument) (see page 8-46)
ModPos	modify a position (see page 8-28)
Search	search for/replace a specific argument (see page 8-34)
Show/Hide IPL	show/hide an instruction pick list (see page 8-13)

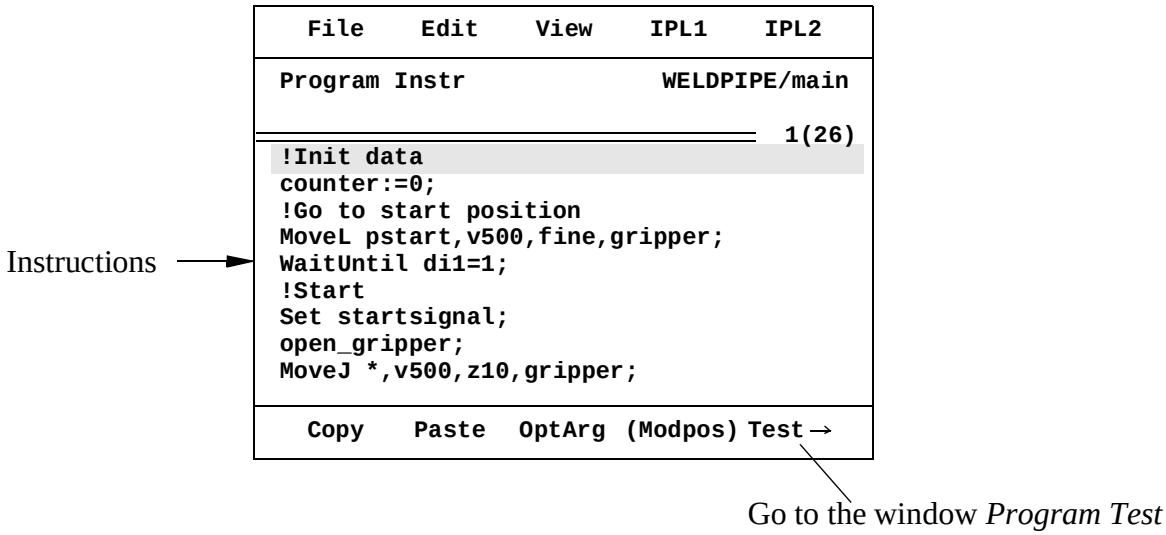
3.2.3 Menu: View

View

1 Instr.”<latest routine>” **2 Routines** **3 Data ”<latest type>”** **4 Data Types** **5 Test** **6 Modules**
7 Main Routine **8 Selected Routine**
9 Error Handler

Command	Used to view:
Instr.	instructions for the current routine – *Program Instruction* window – (see page 8-11)
Routines	all routines – *Program Routines* window – (see page 8-8)
Data	program data – *Program Data* window – (see page 8-42)
Data Types	all data types – *Program Data Types* window – (see page 8-42)
Test	the *Program Test* window (see page 8-20)
Modules	all modules – *Program Modules* window – (see page 8-51)
Main Routine	instructions for the main routine (see page 8-10)
Selected Routine	instructions for the selected routine (see page 8-11)
Error Handler	error handler of the current routine (see page 8-48)

3.3 Window: Program Instr



3.3.1 Menu: IPL1 (shows different instruction pick lists)

IPL1

```
1 Common
2 Prog. Flow
3 Various
4 Motion Settings
5 Motion&Process
6 IO
7 Communicate
8 Interrupts
9 Error Recovery
0 System&Time
  Mathematics
```

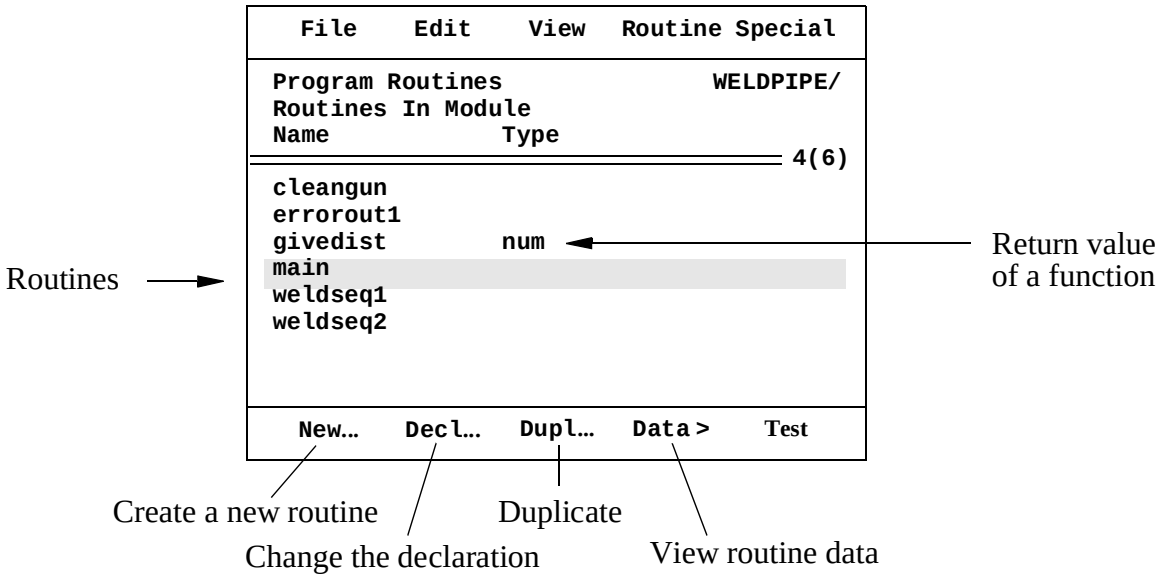
3.3.2 Menu: IPL2 (shows different instruction pick lists)

IPL2

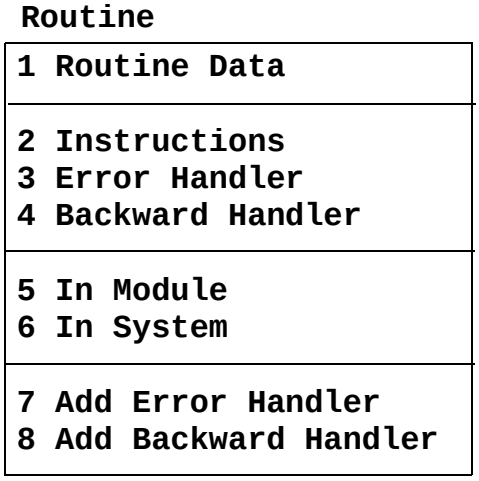
```
1 Most Common 1
2 Most Common 2
3 Most Common 3
```

```
4 Motion Set Adv
5 Motion Adv
6 Ext. Computer
7 Multi Tasking
8 RAPID Support
9 Service
```

3.4 Window: Program Routines

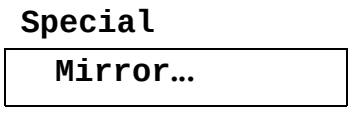


3.4.1 Menu: Routine



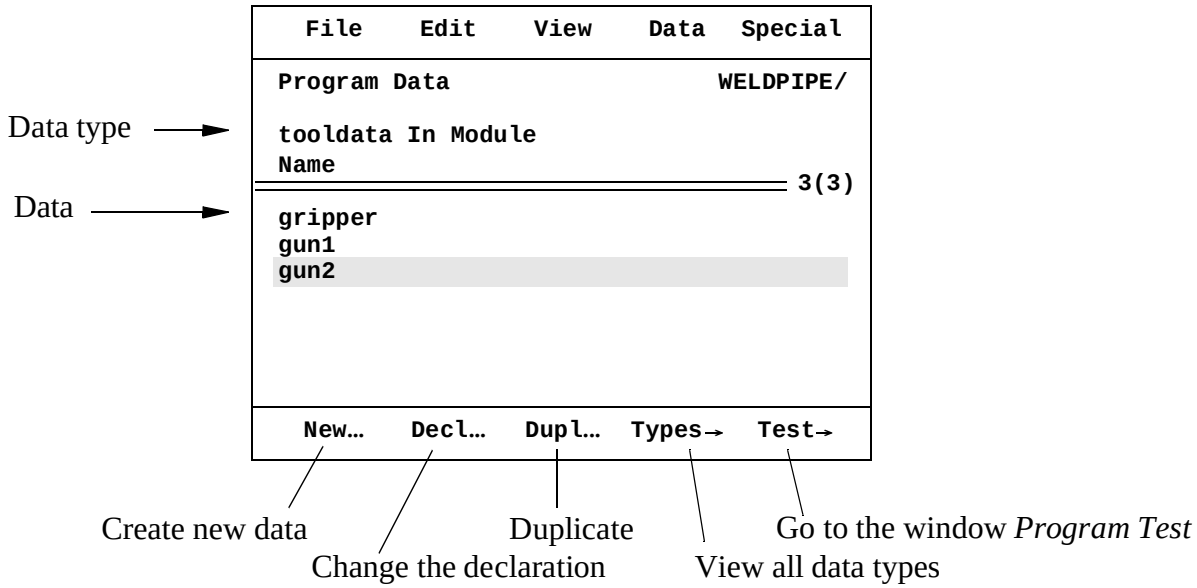
Command:	Used to:
Routine Data	create a new routine (see page 8-8)
Instructions	view the instructions of the selected routine
Error Handler	view the error handler of the selected routine
Backward Handler	view the backward handler of the selected routine
In Module	view only the routines in the current module (see page 8-53)
In System	view all routines in all modules (see page 8-53)
Add/Remove Error Handler	add/remove an error handler to the selected routine (see page 8-49)
Add/Remove Backward Handler	add/remove a backward handler to the selected routine (see RAPID Reference Manual - *Programming off-line*)

3.4.2 Menu: Special



Command:	Used to:
Mirror	mirror a routine or a module (see page 8-36)

3.5 Window: Program Data



3.5.1 Menu: Data

Data

1 Value **2 Types**
3 In Module **4 In System** **5 In Routine “routine name”**

Command:	Used to:
Value	read or change the current value of selected data (see page 8-46)
Types	call up the list with all data types (see page 8-42)
In Module	call up only the data in the current module (see page 8-54)
In System	create new data (see page 8-43)
In Routine	call up all routine data

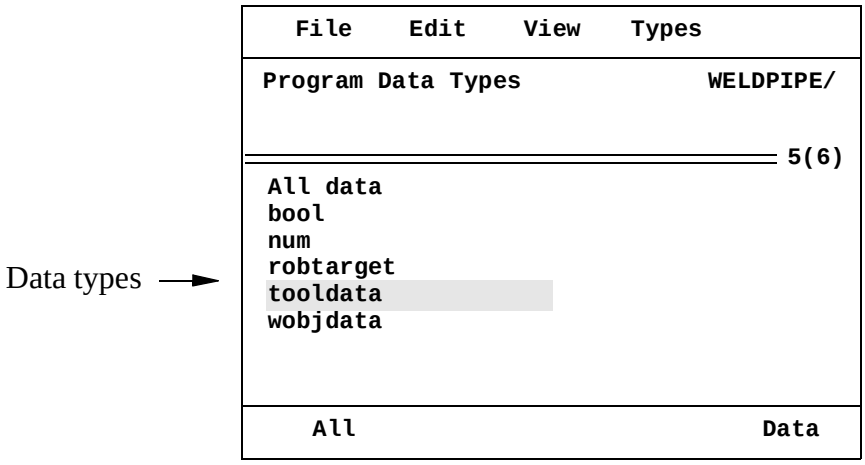
3.5.2 Menu: Special

Special

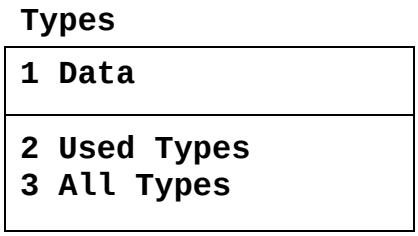
1 Define Coord…
2 Go to selected position

Command:	Used to:
Define Coord	define a tool, work object or program displacement (see Chapter 10, Calibration)
Go to selected position	go to a selected position

3.6 Window: Program Data Types

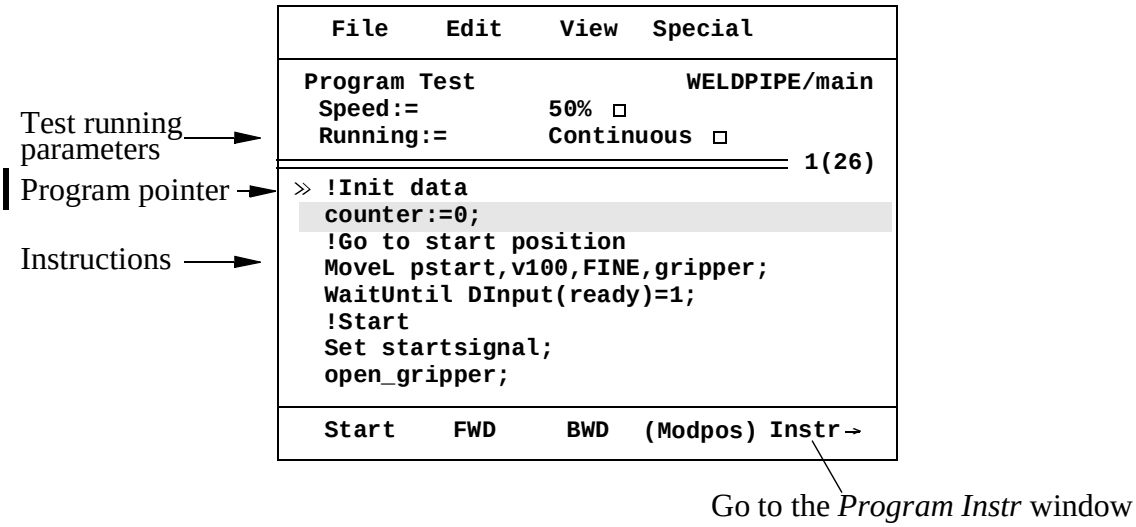


3.6.1 Menu: Types

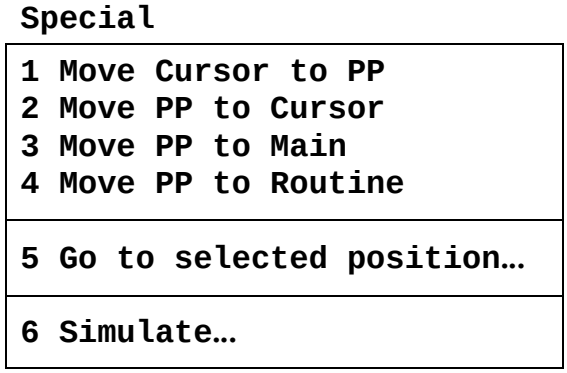


Command:	Used to:
Data	call up all data of a selected type
Used Types	call up only those data types that are used
All Types	call up all data types (see page 8-42)

3.7 Window: Program Test

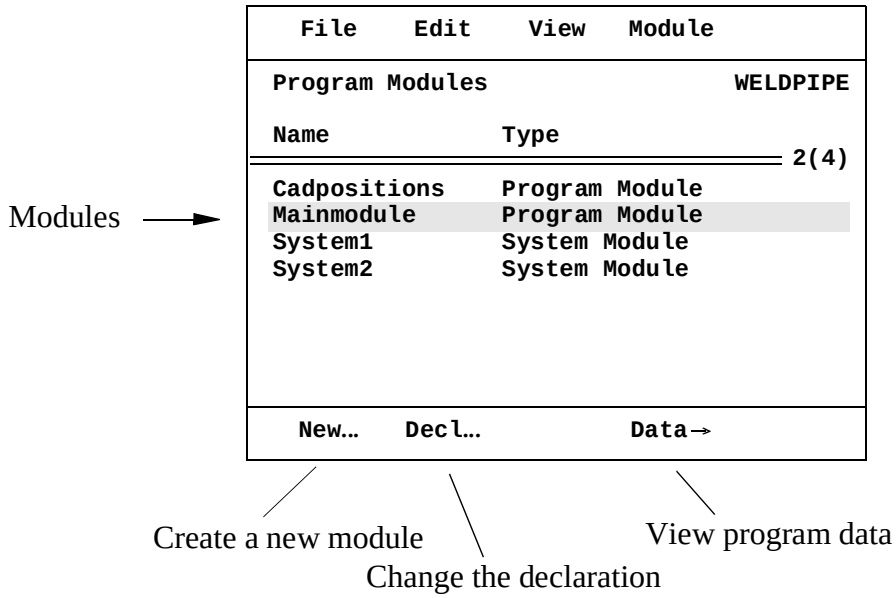


3.7.1 Menu: Test

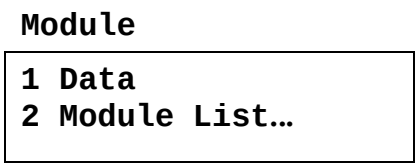


Command:	Used to:
Move Cursor to PP	start from the latest stopped instruction (see page 8-23)
Move PP to Cursor	start from the selected instruction (see page 8-23)
Move PP to Main	start from the main routine (see page 8-23)
Move PP to Routine	start from any routine (see page 8-23)
Go to selected position	go to a selected position
Simulate	allow program execution in MOTORS OFF mode

3.8 Window: Program Modules



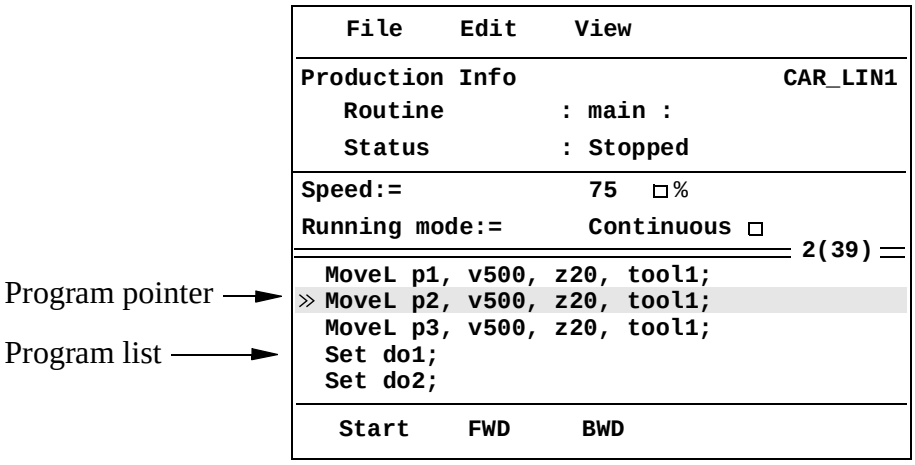
3.8.1 Menu: Module



Command:	Used to:
Data	view program data
Module List	view the complete module in a list (see page 8-55)

4 The Production Window

4.1 Window: Production



4.1.1 Menu: File

File

```
1 Load Program…
```

Command	Used to:
Load Program	load a program (see page 11-4)

4.1.2 Menu: Edit

Edit

```
1 Goto…
2 Start from Beginning
```

Command	Used to:
Goto	go to a specific instruction
Start from Beginning	go to the first instruction in the program (see page 11-6)

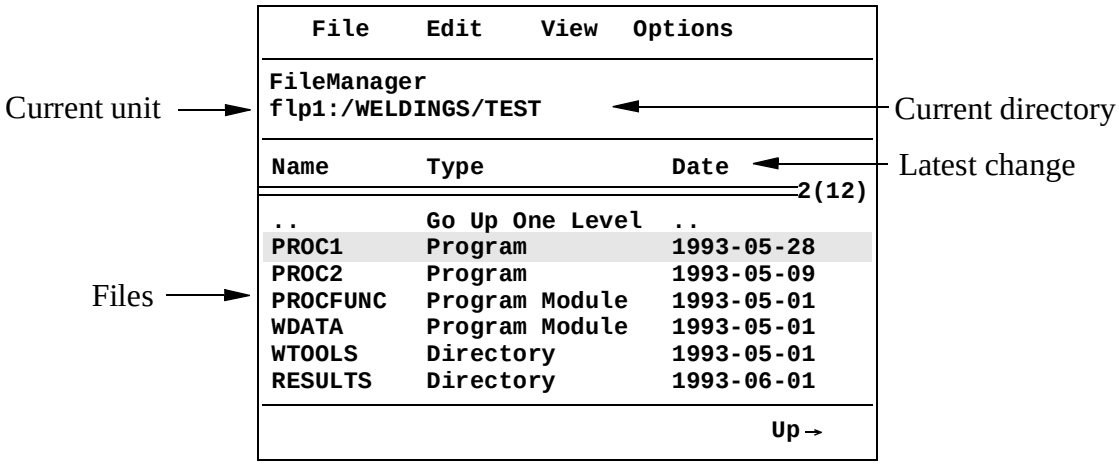
4.1.3 Menu: View

```
View
1 Info
2 Position
```

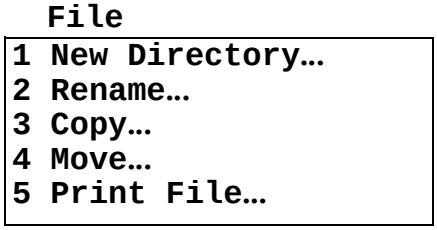
Command	Used to:
Info	display the program in the lower part of the window
Position	tune a position (see page 11-7)

5 The FileManager

5.1 Window: FileManager

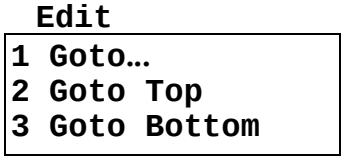


5.1.1 Menu: File



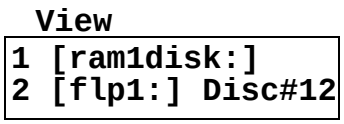
Command:	Used to:
New Directory	create a new directory (see page 13-5)
Rename	change the name of a selected file (see page 13-5)
Copy	copy a selected file or directory to another mass memory or directory (see page 13-6)
Move	move a selected file or directory to another mass memory or directory (see page 13-7)
Print File	print a file on a printer

5.1.2 Menu: Edit



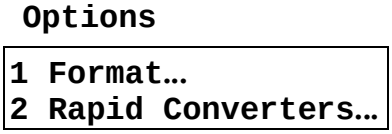
Command:	Used to:
Goto	go to a specific line in a list
Goto Top	go to the first file in a list
Goto Bottom	go to the last file in a list

5.1.3 Menu: View



Command:	Used to view:
ram1disk:	the files on the RAM disk (see page 13-4)
flp1:	the files on the diskette (see page 13-4)

5.1.4 Menu: Options

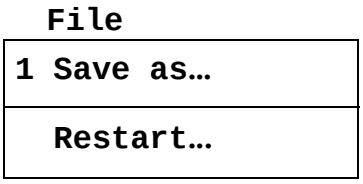


Command:	Used to:
Format	format a diskette (see page 13-7)
Rapid Converters	convert old program versions

6 The Service Window

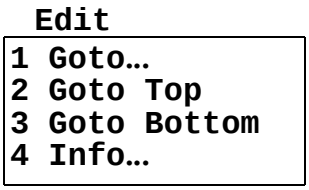
6.1 General menus

6.1.1 Menu: File



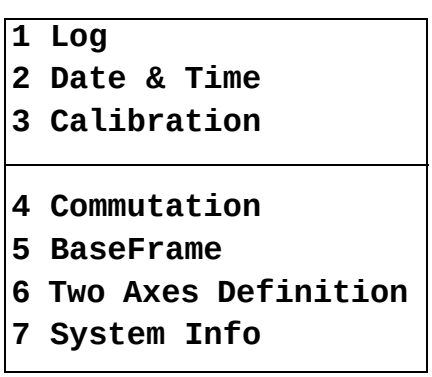
Command	Used to:
Save as	save logs on a diskette or other mass memory (see page 14-7)
Restart	restart the robot

6.1.2 Menu: Edit



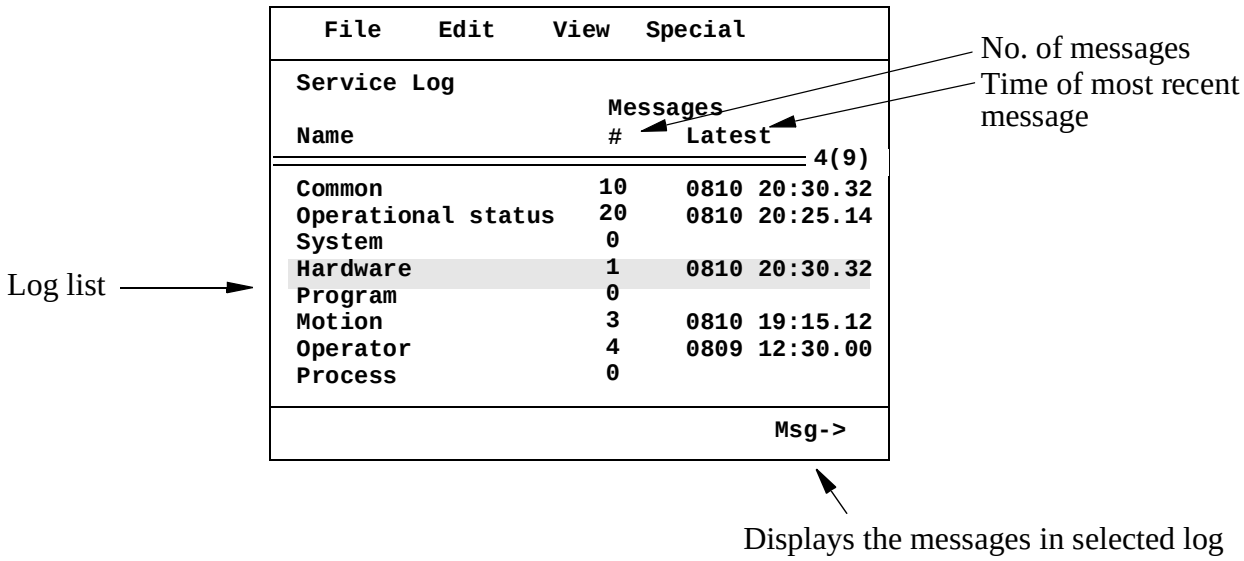
Command	Used to:
Goto	go to a specific line in a list
Goto Top	go to the first line in a list
Goto Bottom	go to the last line in a list
Info	view information about selected log messages (see page 14-6)

6.1.3 Menu: View



Command	Used to:
Log	display the different logs (see page 14-5)
Date & Time	set the date and time (see page 14-3)
Calibration	calibrate the robot (see page 14-8)
Commutation	commutate the motors (see The Product Manual/ *Repairs*)
BaseFrame	calibrate the base coordinate system (see Chapter 10, Calibration)
Two Axes Definition	calibrate the base coordinate system for a two axes manipulator (see Chapter 10, Calibration)
System Info	display system information (see page 14-9)

6.2 Window Service Log



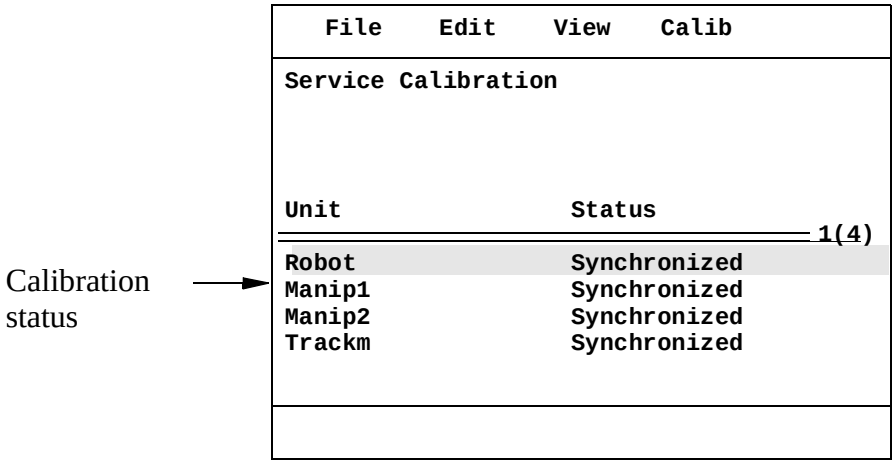
6.2.1 Menu: Special

Special

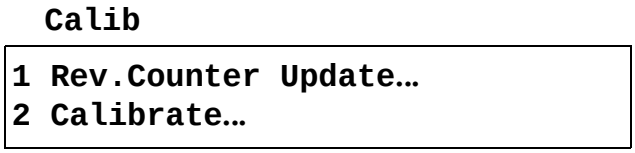
1 Erase Log **2 Erase All Logs**
3 Update log on Event

Command:	Used to:
Erase Log	erase contents in selected log (see page 14-6)
Erase All Logs	erase contents in all logs (see page 14-6)
Update log on Event	update the log directly when a message is sent – the command is changed to **"Update log on Command"** when selected, which means that the log is not updated until the function key **Update** is pressed (see page 14-7)

6.3 Window Service Calibration

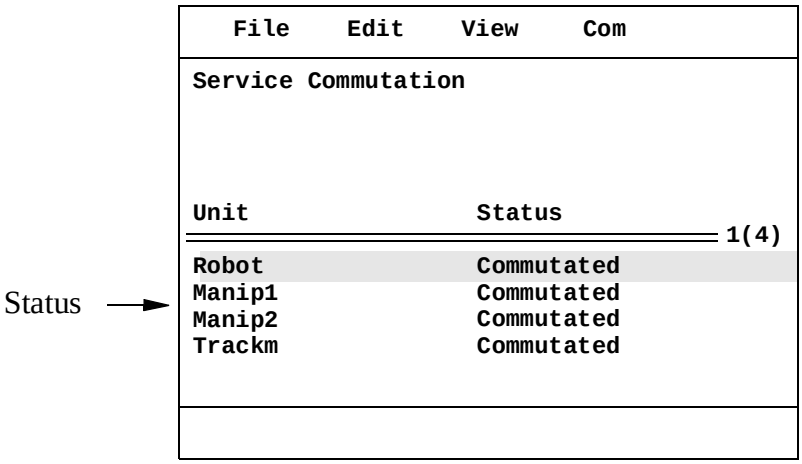


6.3.1 Menu: Calib

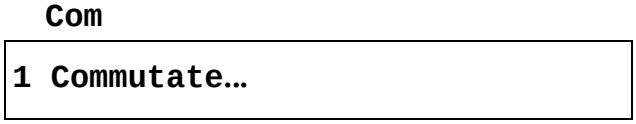


Command:	Used to:
Rev.Counter Update	update the counter (see Chapter 10, Calibration)
Calibrate	calibrate using the measurement system (see the Product Manual/*Repairs*)

6.4 Window Service Commutation



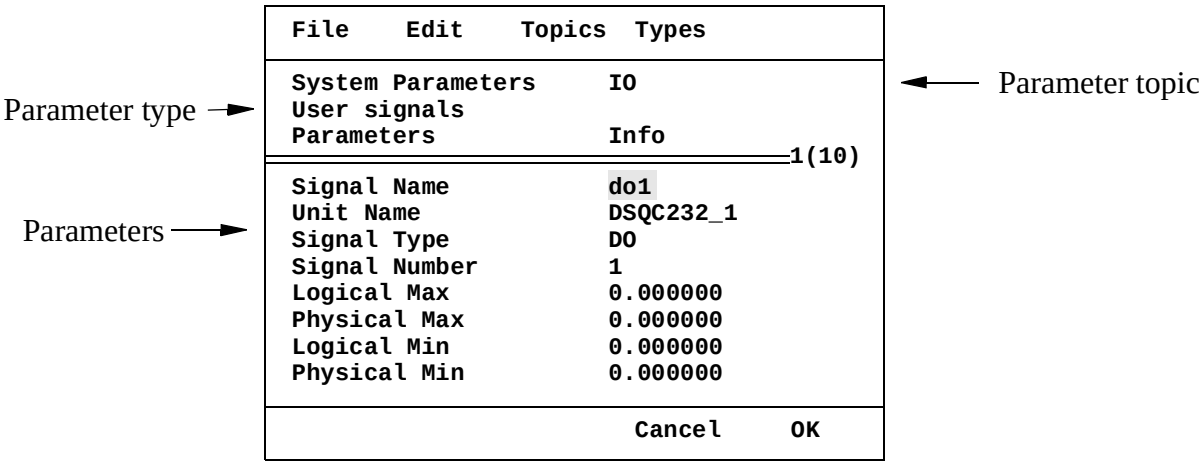
6.4.1 Menu: Com



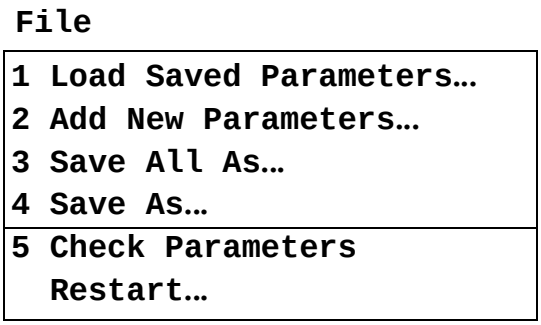
Command:	Used to:
Commutate	commutate using the measurement system (see the Product Manual/*Repairs*)

7 The System Parameters

7.1 Window: System Parameters

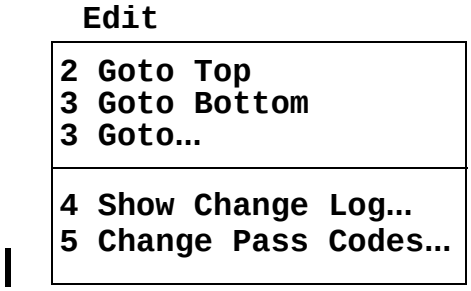


7.1.1 Menu: File



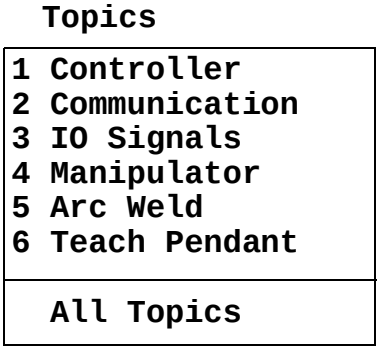
Command:	Used to:
Load Saved Parameters	load parameters from mass storage (see page 12-7)
Add New Parameters	add parameters from mass storage (see page 12-7)
Save All As	save all parameters on mass storage (see page 12-6)
Save As	save parameters on mass storage (see page 12-6)
Check Parameters	check parameters before restart (see page 12-5)
Restart	restart the robot (see page 12-4)

7.1.2 Menu: Edit



Command:	Used to:
Goto Top	go to the first line in a list
Goto Bottom	go to the last line in a list
Goto	go to a specific line in a list
Show Change Log	view information about the latest modifications made (see page 12-5)
Change Pass Codes	change pass codes (see page 12-36)

7.1.3 Menu: Topics



Command:	Used to view:
Controller	the parameter of the Controller topic (see page 12-27)
Communication	the parameter of the Communication topic (see page 12-14)
IO Signals	the parameters of the IO topic (see page 12-9)
Manipulator	the parameters of the Manipulator topic (see page 12-43)
Arc Weld	the parameters of the Arc Weld topic (see Chapter 16, System Parameters ProcessWare
Teach Pendant	the parameters of the Teach Pendant topic (see page 12-35)
All Topics	all topics (see page 12-3)

The contents under the chapter

“Special Functionality in this Robot”

is depending on how the robot is equipped,

and is therefore not included in the On-line Manual.

Click on the Main menu button below to continue to the front page.

Main menu