



BMBF Lead Project
Anthropomorphe Assistenzsysteme
(MORPHA)

Style Guide
for Icon-based Programming

compiled by



KUKA Roboter GmbH
Blücherstr. 144
D-86165 Augsburg



Reis GmbH & Co Maschinenfabrik
Industriegebiet an der B426
D-63785 Obernburg

Contact:

Dr.-Ing. Arif Kazi

Phone: ++49 - 821 - 797 - 2639

Email: arifkazi@kuka-roboter.de

Contact:

Dipl.-Ing. (FH) Markus Seyfarth

Phone: ++49 - 6022 - 503 - 565

Email: m.seyfarth@reisrobotics.de

Contents

1	Introduction.....	3
1.1	Motivation	3
1.2	Intention of this document.....	4
1.3	Document structure	5
1.4	Conventions.....	5
2	Icon editor functionality	7
2.1	Main Screens.....	7
2.1.1	Overview of main screens	7
2.1.2	Navigation between main screens.....	8
2.2	Program administration screen.....	9
2.2.1	Windows.....	9
2.2.2	Selection and Navigation.....	9
2.2.3	Functionality	11
2.3	Program editing screen	12
2.3.1	Windows.....	12
2.3.2	Selection and Navigation.....	13
2.3.3	Functionality	14
2.4	Program test screen	16
2.4.1	Windows.....	16
2.4.2	Selection and Navigation.....	16
2.4.3	Functionality	17
3	General Design Principles	19
3.1	Design rules for touch screens	19
3.1.1	General information.....	19
3.1.2	Using finger and/or pen for input	19
3.1.3	Arrangement, size and spacing of the buttons	20
3.1.4	Labeling and colouration of buttons.....	21
3.1.5	Displaying buttons on the screen	21
3.1.6	Feedback.....	21
3.1.7	Function triggering.....	22
3.2	Guidelines for Speech I/O.....	22
3.2.1	General information.....	22

3.2.2	Suitability of speech-operated interactions	23
3.2.3	Speech dialog elements	24
4	Interface objects	28
4.1	Action objects	28
4.1.1	Push Button	29
4.1.2	Jog Button	29
4.2	Selection Objects.....	30
4.2.1	Toggle Button (Check Box)	30
4.2.2	MultiState Button	31
4.2.3	Radio Button.....	32
4.2.4	ListView Control	34
4.2.5	TreeView Control.....	35
4.3	Manipulation Objects	36
4.3.1	Edit Box	36
4.3.2	On-screen Keyboard	38
4.3.3	Spin Button.....	38
4.3.4	Analog Value Bar	39
4.3.5	Slider	40
4.4	Supporting Objects	41
4.4.1	Menu	41
4.4.2	Tab Control.....	42
4.4.3	Grabber	43
4.4.4	Scroll Bar.....	44
5	Graphical symbols and icons	46
6	References	54

1 Introduction

1.1 Motivation

The user interfaces of today's industrial robots are highly manufacturer-specific. This, with the complexity of the functionality provided by state-of-the-art robots, dramatically increases the training effort for the end customer, particularly when using robots of different manufacturers. From this point of view, a minimum degree of conformity between the different user interfaces appears highly desirable. In the late 1990s, the European research project AMIRA (Advanced Man Machine Interfaces for Robot System Applications, ESPRIT Project 22646) addressed this issue by developing a style guide for the design of robotic user interfaces. In the AMIRA style guide [AMIRA], four different user interface areas were identified (installing and configuring, programming, system operation and production monitoring, training and support) and covered with respect to user interface design.

Since the end of the AMIRA project, particularly the innovative approach of *icon-based programming* (see fig. 1-1) has received considerable interest [Born]. Its main advantages over conventional text-based programming are superior overview over the program structure and a higher level of intuitiveness. In addition, no typing is required any more, which automatically leads to an elimination of syntax errors.

The AMIRA style guide was developed for teach pendants with a graphical screen surrounded by soft keys, as it is standard for most industrial robot controllers today. Recently, touch screen user interfaces have become more and more popular for stationary interfaces, and they may be also well suited for mobile robot teach pendants. Speech control is another technology which is becoming mature, and may in specific instances be convenient as a means of interaction with the robot controller. Most interestingly, an icon-based programming user interface appears to be well-suited for incorporating both touch screen and speech control.

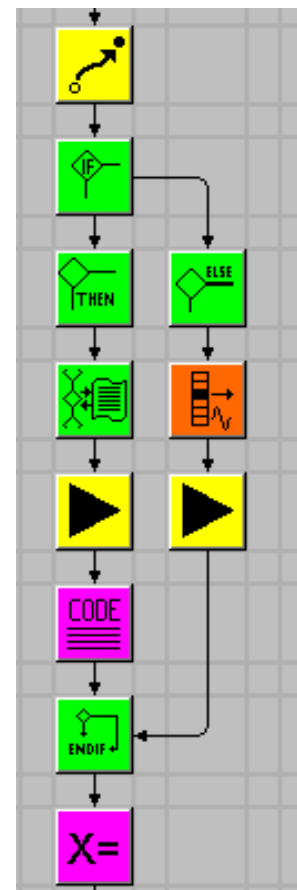


Fig. 1-1: Icon-coded program sequence

The German BMBF lead project MORPHA focuses on the interaction between the human and the robots of the future. In this context, two well-known German manufacturers of industrial robots, namely

KUKA Roboter, Augsburg and

REIS Robotics, Obernburg,

have joined forces to investigate touch screen operation and speech control for icon-based programming. They document their experience in this document, the MORPHA style guide for icon-based programming.

1.2 Intention of this document

The MORPHA style guide for icon-based programming follows largely the lines of the AMIRA style guide. It does not intend to repeat the wealth of general guidelines provided in the latter, but focuses specifically on an implementation using touch screen and speech control. Its scope is restricted to the programming user interface area. At the same time, this document describes not only the interface objects for the user interface design, but also the functionality that needs to be provided. It addresses specifically the designers of robot systems using graphical user interfaces, although much of the material presented may be of interest to anyone concerned with robots and/or user interface design.

The main benefits of this document are twofold: the user of robotic systems will benefit from reduced training effort and improved operability and ergonomics of user interfaces designed according to this style guide. For the designer, the reduced implementation effort and cost may be one of the main benefits, as he can draw on the experience incorporated in this document without having to find the corresponding solutions himself.

The guidelines and recommendations given in this document have been tested practically in prototype implementations within the MORPHA project. Please note that, however, no liability for any damages resulting from the application of this document can be assumed by the authors or their institutions.

1.3 Document structure

This document is organised in the following manner:

- Chapter 2 describes the general functionality to be provided with an icon-based programming interface, grouped according to three main screens defined in this context. This description is independent of the specific user interface implementation using touch screen or other means of interaction.
- In Chapter 3, general guidelines and recommendations for user interface design for touch screen and speech control are given.
- Chapter 4 focuses on the description of the interface objects, i.e. their properties and their behaviour when operated by touch screen or by speech input.
- Chapter 5 concludes this document with a list of graphical symbols and icons.

1.4 Conventions

In this document, two sets of conventions are used to highlight important topics and enhance the clarity for the reader.


On the level of specification of functionality, “mandatory” and “optional” functions are distinguished. Mandatory functions have to be provided in the user interface, as they are necessary for the basic operation. They are indicated by the symbol “!”, e.g.


- ! The NEW function is used to create a new folder or program. If a folder is selected, a new folder is created within the selected one. Either a default name is assigned or the user is prompted for one.


Optional functions may be provided on the user interface. They are not necessary for the basic operation, yet increase the level of comfort for the user. They are indicated by the symbol “❖”, e.g.


- ❖ The PRINT function will print out the selected programs if a printer is available.

Furthermore, two levels of requirement conventions are distinguished: “guidelines” and “recommendations”. Guidelines are the stronger form of requirement convention.

The symbol “” is used to indicate a guideline. Typically, the term “must” is used, for example

-  The *program manipulation* window must always remain visible as long as the *program editing* screen is selected.

Recommendations are the weaker form of requirement convention. The symbol “” is used to indicate a recommendation. Typically, the term “should” is used, for example

-  The magnification factor of the *program manipulation* window should remain fixed at all times.

A user interface that implements all mandatory functions and all guidelines in this document may be labelled

“Conform to the MORPHA style guide for icon-based programming”.

2 Icon editor functionality

This section describes the functionality of an icon-based programming user interface. Following a task-oriented approach, three main graphical screens are defined. The purpose of these main screens is described in section 2.1. The subsequent three sections detail the functionality to be provided on each main screen.

On graphical user interfaces designed for touch screen, screen space is scarce due to the relatively large minimum size of the interface objects (see section 3.1.2). In this chapter, a minimum set of mandatory functions is therefore defined that allows for basic operation. Additional optional functions will increase the level of comfort for the user. For each of the functions described in the following sections, it is defined whether the function is considered mandatory or optional. For greater clarity to the user, it is advised not only to implement the functionality described in this chapter, but also to draw on the function names used in this document. (Of course, the functionality set described in this document can be extended by manufacturer-specific functions).

Generally, all functions described should be accessible via touch screen. In addition, speech input can be used as an additional input modality to operate certain functions. In cases where it is not advisable to use speech input for safety or other reasons, appropriate annotations are given in the text. It is also mentioned if a function appears to be particularly well suited for operation by speech control.

2.1 Main Screens

2.1.1 Overview of main screens

In the “programming” user interface area, three main screens are defined which are governed by the sequence of activities of the user:

- ! The *program administration* screen offers the functionality which is found in common desktop file administration tools (e.g. the Windows® Explorer).
- ! The *program editing* screen offers the functionality needed for creating and changing the program itself (e.g. inserting and deleting commands).
- ! The *program test* screen offers the functions for testing existing programs.

- ❖ The screens for *program editing* and *program test* may be combined, if the space available for the different controls is sufficient.

2.1.2 Navigation between main screens

The possible paths of navigation between the three main screens are shown in the following diagram (fig. 2-1):

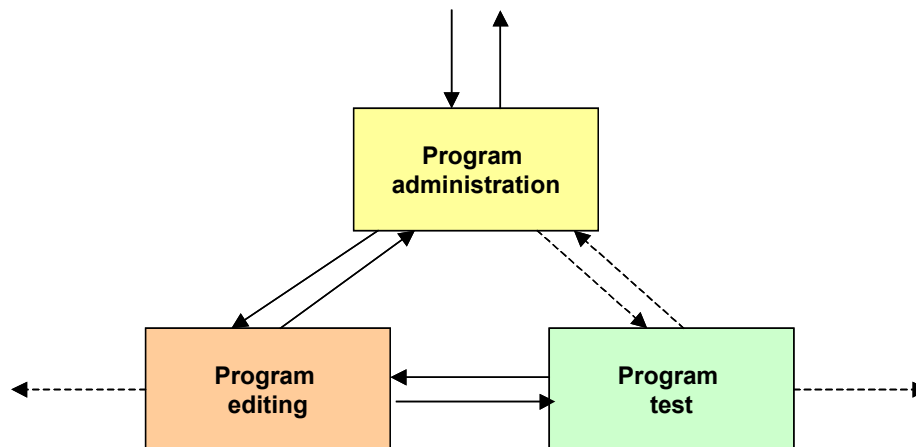


Fig. 2-1: Navigation paths between the main screens.

Program administration screen:

- ! The *program administration* screen is the entry point to the *programming* user interface area. From this screen, it will also be possible to exit.
- ! Means to access the *program editing* screen is provided.
- ❖ Access to the *program test* screen may also be provided.

Program editing screen:

- ! From the *program editing* screen, the *program test* screen can be accessed.
- ! The return to the *program administration* screen is also possible.
- ❖ Means may also be provided to exit the *programming* user interface area.

Program test screen:

- ! From the *program test* screen, it is possible to return to the *program editing* screen.
- ❖ Access to the *program administration* screen may also be made possible as well as exiting of the *programming* user interface area.

2.2 Program administration screen

2.2.1 Windows

The program administration screen contains two graphical windows with the following functionality:

- ! The left window provides means to display a tree view of the folder structure of the robot system hard disk as well as additional restoring devices (e.g. floppy disk, CD-ROM, ...).
- ! The right window provides means to display the list of programs contained in a selected folder. The folder may be selected in the left window.
- ❖ As an option, additional means may also be provided in the left window to display the contents of a selected folder (i.e. programs and subfolders). In this case, information about the path of the selected folder has to be available to the user.
- ❖ As an option, means may also be provided to display a tree view of a folder structure in the right window.

Guidelines and recommendations:

- 👍 In any case, it is recommended that the default configuration at start-up should show the tree view in the left window and the folder content in the right window.

2.2.2 Selection and Navigation

Selecting a folder or a program in one of the graphical windows will subsequently allow to perform operations to it as defined in section 2.2.3. The selection of an object is shown graphically on the display (e.g. colour change of the object or its background, etc.). Object selection functions are:

- ! the simple SELECT function: a program or a folder are selected simply by pressing the symbol on the touch screen or by speaking its name when under speech control.
- ❖ the MULTI-SELECT function, which allows for selection of more than one object at a time. If it is enabled, each newly selected object will be added to the selection list. If the object had already been selected, it will be removed from the selection list.

- ❖ the SELECT ALL function, which is used to select all programs displayed in the window (i.e. in the selected folder).

Opening and closing folder in the tree view in one of the graphical windows works similar to the well-known “Explorer” tool for office desktops.

- ! A closed folder is opened by pressing on the “+” symbol adjacent to the folder on the touch screen.
- ! An open folder is closed by pressing on the “-“ symbol adjacent to the folder on the touch screen.

Guidelines and recommendations:

- 👍 Opening and closing folders via speech input would be cumbersome and is not recommended (see also section 4.2.4 on TreeView Controls).
- 👍 If there is no ambiguity, the user may call a program contained somewhere in the tree directly. On the display, the corresponding folders should be opened automatically. In this case, speech control can be useful (see also section 4.2.4 on TreeView Controls).

In both graphical windows, adjustment of the view is provided by

- ❖ the SCROLL UP/DOWN function, which becomes available when a window contains more objects than can be displayed at once.
- ❖ the GRABBER function, which also allows the user to adjust the visible part of the window by directly touching a spot in the window area and “dragging” it to a new position.

Navigation to the other main screens is performed by

- ! the EDIT function, which will open the selected program on the *program editing* screen.
- ❖ the TEST function, which will open the selected program on the *program test* screen (only available if a direct link exists – see section 2.1.2).

2.2.3 Functionality

The functions provided in the *program administration* screen can be grouped into the two categories

- specific functions for object manipulation and
- general functions for object manipulation

The specific object manipulation functions will comprise

- ! the NEW function is used to create a new folder or program. If a folder is selected, a new folder is created within the selected one. Either a default name is assigned or the user is prompted for one.
- ! the RENAME function can be called to alter the name (and, if available, comments) of the selected folder or program. The command is also used to change the attributes (e.g. Read Only, Hidden, Archive, System). Usually, an on-screen keyboard will be used to type in the new name.
- ! the COPY function is used to copy selected programs and/or folders to the clipboard. If a folder is copied, all contents (i.e. all subfolders and programs) are copied.
- ! the CUT function is used to copy programs or folders with all contents to the clipboard and at the same time delete the selected programs or folders.
- ! the PASTE function is only available after the first CUT or COPY call. It will insert the programs or folders in the clipboard into the selected folder.
- ❖ the DELETE function erases the selected programs or folders. A message box will occur requesting the user to confirm this action. In the case of deleting a folder, all sub folders will also be deleted (optional, this function is already contained in CUT).
If the floppy disk is selected, the DELETE function can turn into the FORMAT function.
- ❖ the PRINT function will print out the selected programs, if a printer is available.

The general object manipulation functions will comprise

- ❖ the SEARCH function allows the user to search specific programs or folders by selecting criteria in a filter. The filter should contain criteria like e.g. name (including regular expressions), date/time, etc.

2.3 Program editing screen

2.3.1 Windows

The *program editing* screen contains at least one graphical window. This *program manipulation* window displays the structure of the selected program in a flow diagram-like manner. It allows to select commands and perform manipulations on them.

Guidelines and recommendations:

- 👉 The program manipulation window must always remain visible as long as the program editing screen is selected.
- 👍 The magnification factor of the program manipulation window should remain fixed at all times. The icons representing the individual commands should be displayed in a size that the type of command (e.g. PTP motion, etc.) is clearly visible and the user can select an icon by means of touch screen input.

There may also be a second window in the *program editing* screen. This is the *program overview* window, which shows the program graph in a scaled-down way in order to give an enhanced overview of larger programs.

Guidelines and recommendations:

- 👍 The *program overview* window should be located to the right of the *program manipulation* window
- 👉 The magnification factor of the *program overview* window must be adjustable by the user. The maximum scaling factor should be limited to the fixed factor of the *program manipulation* window. At the minimum scaling factor, the full program should be visible within the window.

2.3.2 Selection and Navigation

Very much like in the program administration screen, selection of an icon allows to perform the operations described in section 2.3.3 on it. Icons in the *program manipulation* window may be selected by

- ! simple selection via touch screen or speech input
- ❖ the MARK ON/OFF function in cases where more than one icon needs to be selected (e.g. when creating a macro, etc.). When the MARK mode is ON, the icon selected previously serves as a reference point for the selection. After activating the mode, the selection of an icon will select all icons between this icon and the reference icon.

Guidelines and recommendations:

- 👍 In the *program overview* window, a selection of icons should not be possible.

The functions for view adjustment comprise

- ❖ the SCROLL function for both *program manipulation* and *program overview* windows, which allows the user to move a specific part of the program into the visible area.
- ❖ the GRABBER function, which also allows the user to adjust the part program visible in the two windows, but by directly touching a spot in the program area and “dragging” it to a new position.
- ❖ the INFO function, which provides additional information about each command in the *program manipulation* window (e.g. the name of the end point of a motion command).
- ❖ the ZOOM/UNZOOM function for the *program overview* window.

Guidelines and recommendations:

- 👍 In order to avoid unintentional selections of icons when working with the GRABBER function, it may be suitable to switch off the means for selection in a dedicated “grabber” mode.

Navigation to the other main screens is performed by

- ! the TEST function, which will open the program being edited in the *program test* screen.
- ! the CLOSE function, which will close (and, after a user dialog, also save) the program being edited and return to the *program administration* screen.

Guidelines and recommendations:

- ☺ For a better orientation of the operator, the contents of the *program manipulation* window of the *program editing* screen should be displayed in the *program state* window after switching to the *program test* screen.

2.3.3 Functionality

The functions supplied in the *program editing* screen can be grouped into the following categories:

- Icon tree
- Specific functions for command manipulation
- General functions for command manipulation
- Functions for program organisation

The ICON TREE is a functionality associated to the *program manipulation* window. It offers the user all available commands in a hierarchical (“menu-type”) fashion. The command icons are members of the following five command groups (see [ISO 15187])

- program structure (e.g. FOR, WHILE, IF),
- robot movement (e.g. PTP, LIN, CIRC),
- I/O control and
- application control (e.g. gripper, welding)

Guidelines and recommendations:

- 👍 It is recommended to use different colours to distinguish between command icons belonging to different command groups.

The specific functions for command manipulation comprise

- ! the INSERT ICON functionality, by which the command currently selected in the icon tree will be inserted into the program after the command currently selected in the *program manipulation* window.
- ! The COPY ICON function is used to copy the icon to the clipboard. All command parameters are copied, too.
- ! The CUT ICON function will copy to the clipboard and at the same time remove it from the program.
- ! The PASTE ICON function inserts the icon in the clipboard after the icon currently selected in the program.
- ❖ The DELETE ICON function removes the selected icon from the program.
- ! The EDIT PARAMETER function opens a dialog for the icon currently selected in the program. In the dialog, the specific command parameters can be edited.
- ! The TOUCH UP function appears within the parameter dialog of the robot movement icons. It stores the actual position of the robot (used e.g. as destination position for the PTP function).

The general functions for command manipulation comprise

- ❖ the UNDO function, which allows the user to undo the last editing command.
- ❖ the FIND & REPLACE function, which allows the user to search for a string in a program. A dialog box should allow to pre-set further options like upper/lower case distinction, using regular expressions etc.
- ❖ the PRINT function (only if a printer is available), which will print the current program. If several commands are selected in a program, the print-out may contain only these commands.

The functions for program organisation comprise

- ❖ the MACRO functionality set, which will summarise a group of commands into one icon. The user can CREATE new or REMOVE, OPEN and CLOSE existing macros.
- ❖ The SUB functionality set, which inserts a subroutine into the program. The user has the possibility to get a NEW subroutine which is empty or CREATE a subroutine from the icons currently marked in the program.

2.4 Program test screen

2.4.1 Windows

The program test screen contains at least one window. This *program state* window is equivalent to the *program manipulation* window on the *program editing* screen.

Additional windows may be provided for viewing the

- state of system variables,
- state of I/Os,
- robot position,
- etc.

These windows may also allow for the manipulation of the values displayed, wherever appropriate. The implementation is considered manufacturer-specific, therefore no further guidelines or recommendations are given in this document.

2.4.2 Selection and Navigation

The selection of an icon in the *program state* window is identical to the selection in the *program editing* screen, i.e. by

- ! simple selection via touch screen or speech input

The view in the program state window can be adjusted by

- ❖ the SCROLL function, which allows the user to move a specific part of the program into the visible area.
- ❖ the GRABBER function, which also allows the user to adjust the part program visible in the two windows, but by directly touching a spot in the program area and “dragging” it to a new position.

Guidelines and recommendations:

- 👉 In order to avoid unintentional selections of icons when working with the GRABBER function, it may be suitable to switch the means for selection off in a dedicated “grabber” mode.

Navigation to the other main screens is performed by

- ! the EDIT function, which will open the program being tested in the *program editing* screen. For an enhanced orientation of the operator, the contents of the *program state* window of the *program test* screen should be displayed in the *program manipulation* window after switching to the *program editing* screen.
- ❖ the CLOSE function, which will close (and, after a user dialog, also save) the program being tested and return to the *program administration* screen (only available if a direct link exists – see section 2.1.2).

2.4.3 Functionality

The program test screen allows the operator to test programs, especially to watch the state of the control in selected program steps. Therefore, the program test screen has to offer different functions:

- ! The RESET PROGRAM function moves the program pointer to the start icon of the program.
- ! The SET POINTER function moves the program pointer to the selected icon.
- ! The POV function is used to alter the program override.
- ❖ The SET/REMOVE BREAKPOINT function sets or removes the breakpoint for the selected icon.

- ❖ The REMOVE ALL BREAKPOINTS function will remove all breakpoints in the program.
- ! The EXECUTION MODE function changes the type of test motion. The types that can be provided are
 - ! GO, which will run the program until the next breakpoint or until the end of the program.
 - ! SINGLE STEP, which will execute the commands of the program one by one.
 - ❖ PROCEDURE STEP, which will execute macros and sub-programs
 - ❖ MOTION STEP, which will execute all commands until the next motion command.
 - ❖ BACKWARD, which will make the robot run through the program in a reverse direction to the last position.

3 General Design Principles

This chapter lists some important design rules independent of icon-based programming application. In section 3.1, design rules for touch screen applications are given. Section 3.2 lists design rules for speech control. Sources of information that have been used as a basis for this compilation are [VDI 3850-3], [Krauß], [Elo], [MUSIST], [Becker], [Blattner]

3.1 Design rules for touch screens

The use of a touch screen as an operator control element requires a fundamentally different user interface structure from a conventional Windows® application. The buttons in a Windows® application are generally too small to be operated with a finger. Furthermore, the mouse is no longer available as the input element, i.e. the functionality of the right mouse button is lost and actions such as the double click, drag&drop, etc., must be implemented in a different way.

This section introduces a number of general guidelines which are helpful when creating a new user interface.

3.1.1 General information

An application designed for touch screen operation should only be run in full screen mode, i.e. it should fill the entire screen. The Windows® application title bar and menu bar should be deactivated, as these are only significant for mouse operation.

The mouse pointer should be switched off for the same reason. The operator now sees the entire screen and no longer subconsciously grapples with the problem of how to move the mouse pointer about the screen. Thus, user actions become more effective.

3.1.2 Using finger and/or pen for input

Traditionally, operators use their index finger to interact with a touch screen user interface. However, modern small size mobile devices like PDAs, WebPads, PenPCs etc., are often equipped with a pen as an input device. As interface objects require less space when designed for pen operation, a greater number of them can be placed on a given small screen size.

Pens may also be suitable for interacting with touch screen robot teach pendants, if it is ensured that a pen is always available when required. Pen input may even be advantageous as it avoids the display getting covered with dirt, oil or grease on the operators' fingers or gloves. However, it has to be taken into account that during programming, robot operators are also confronted with tasks that require operation of other input elements ('hard' keys, 6D mouse). Repeatedly taking a pen into the hand and putting it away again will prove to be annoying. In many cases, a mixed design featuring finger operation for certain functions and pen input for others will be a sensible solution. However, such a design needs to be based on a careful analysis of the work flow of the user.

3.1.3 Arrangement, size and spacing of the buttons

Between operation movements, the user adopts a "default posture" (or "rest position"). In the case of right-handed users, the hand will come to rest at the bottom right-hand corner of the screen. As buttons in this area require only small movements, they provide the shortest operator response times, i.e. frequently used functions are best positioned along the bottom or right-hand edge of the screen.

The size and spacing of the buttons are largely dependent on the particular technology being used. For this reason, the values given in the following table are only to be taken as a rough guideline. If in doubt, the use of larger buttons is recommended. Smaller values are only acceptable when it is clear that the use of a pen as input device is acceptable (see section 3.1.2).

		Minimum dimensions [mm]
Touch screen button size	Width	20
	Height	10
Touch screen button spacing	Horizontal	10
	Vertical	5

Table 1: Size and spacing of buttons

With larger buttons (>25 mm), users tend to position their finger at the edge of the button, below the center. For this reason, there should be an inactive area around each button. This area should be at least 5 mm wide in the case of first contact activation, but may be reduced in the case of last contact activation (see below).

3.1.4 Labeling and colouration of buttons

Labels should be positioned at the (upper) edges of the buttons. In this way, users can touch a button without obscuring the label.

Positional coding is to be given priority over colour coding. Buttons should first be arranged according to positional criteria and then assigned colours. Recommendations regarding colours may be found, for example, in DIN and VDI guidelines.

As a rule, light colours should be used for the background in the application. These help to conceal fingerprints on the surface of the screen and minimize reflection in the case of bright lighting. Backgrounds with a pattern or gridlines additionally help the eyes to concentrate on the screen instead of on reflections.

3.1.5 Displaying buttons on the screen

The buttons should appear automatically (without the need to press another button) when an input is required from the operator. Only those buttons should appear which are actually required for the input (letters or numbers). The type of keyboard, i.e. the arrangement of the buttons (MF2, QWERTZ or alphabetical from A-Z), is dependent on the scope of the input and the experience of the user with such layouts and should be investigated separately. For a user unacquainted with a “typewriter” layout and only a small quantity of data to be entered, an alphabetical A-Z layout is more sensible. Bear in mind that it is not possible to activate two buttons simultaneously on a touch screen. For this reason, the shift key, for example, must be implemented as a shift lock key.

3.1.6 Feedback

The nature and speed of feedback from a touch screen has significant influence on the response and performance of the user. This results primarily from the absence of the tactile feedback with which the user is most familiar. A combination of visual and audio feedback appears to be most suitable. The audio feedback informs the user that the system is executing a function or that an entry has been detected. The visual feedback provides information on the touch screen about the function or button which has been detected by the system.

3.1.7 Function triggering

There are two alternatives for triggering functions on the touch screen: the function may be triggered when the display is touched (first contact) or when the button is released (last contact). The error rate is influenced by the way in which functions are triggered. Function triggering on releasing the touch screen allows the operator to correct positioning before removing his finger from the display, thereby reducing the chances of triggering incorrect or undesired functions.

3.2 Guidelines for Speech I/O

Pronouncing speech controls is, in addition to the use of the graphical interface, one further possibility to interact with a multimodal user interface. Since speech is a privileged mode of communication for Human Computer Interaction, it should be designed as intuitive as possible.

Normally, we are used to communicate with machines via graphical interfaces. Speech control is currently only an “add-on” that allows to operate the graphical interface objects by voice. When designing new user interfaces, some specifics of speech should be taken into account. This section helps in designing speech I/O interfaces.

3.2.1 General information

Ideally, the user should be allowed to speak commands or to press alternatively the corresponding graphical button. To reach an optimal advantage of the speech interface, it should be complementary to the graphical one. Which way of interaction fits best to the desired task may vary from case to case and should be left to the user.

General Recommendations

- 👍 A graphical feed-back should be integrated to inform the user about the perception, e.g. a *Push Button* that is activated by speech should be animated in the same manner as being activated by pressing it.
- 👍 In addition to the graphical one, an auditory feedback should be implemented. In many cases, the use of *Earcons* (see section below) is suitable for this purpose.
- 👍 Too short words, esp. words containing only one syllable, should be avoided in order to avoid ambiguities.
- 👍 Vocal Recognition is considered positive by the user only when the performance is high enough: Rate of recognition should be min. 90%, time span until feed-back should be less than 1 sec.
- 👍 An “UNDO” function should be available by one action to revoke the last editing action or to cancel e.g. a menu selection.
- 👍 A Help function should be displayed on the screen and should be accessible by one action.
- 👍 The vocal mode should tolerate several key words to reach one function.
- 👍 Functions available on vocal mode should be shown on the screen when asked (only one action to reach these information).
- 👍 To help the user memorising the speech controls, a list of the available controls should be directly accessible by voice command.
- 👍 For the input of numbers, the speaking the digits one after the other should be allowed.

3.2.2 Suitability of speech-operated interactions

As mentioned above, speech operation is often only another way to operate the graphical interface objects. This section describes suitability of general interaction types. For details, please refer to sect. 4 where the interface objects and their specific suitability for speech operation is noted.

Interactions that are suitable for speech control:

- giving particular commands to the control, e. g. "STOP", "CLOSE".
- operating the interaction objects (described in sect. 4) by calling their labels.
- giving commands or calling objects that are not visible on the screens, e.g. calling directly actions or items of actually closed menus.
- calling objects of a tree structure or a menu (e.g. folder tree) is suitable only when it is possible to call the direct item. Ambiguities must be omitted in this case.
- input of boolean parameter values
- input of numeric parameter values

Interactions that are not recommended for speech control

- navigating in a tree structure by calling the tree nodes to open a child node (e.g. the sub folder of a folder tree)
- navigating through tables
- giving incremental commands for a continuous process like “less” or “more”. In this case, the interaction object (e.g. the *Analog Value Bar*) should be used to enter a concrete parameter value.
- giving spatial commands

3.2.3 Speech dialog elements

For operation of industrial robots, sophisticated dialogs via speech I/O are currently not playing a very important role. Normally, the communication consists of giving special commands to the control and feedback of the machine. In future, speech operation will become more and more important and speech dialogs will transcend simple one-word commands. In this section, the elements of more or less sophisticated dialogs are briefly described.

A dialog based speech UI consists of several basic elements that describe the kind of the individual inputs and outputs. In dialogs, Basic elements are combined to structure elements.

Basic elements

Basic elements are distinguished by following aspects:

- Who initiates the dialog: the user or the machine
- Is it a question / a statement / a command

Basic elements are:

- Output of status information: This is the plainest way of dialog. All values that are given out have to be identified by an unequivocal name: e. g. "The motor speed of axis 5 is 700 rpm."
- Speech commands: Two possibilities have to be distinguished for speech commands:
 - Commands that are defined by words or phrases: The machine recognises exactly pre-defined commands. In this case, no complex dialogs are possible. Disadvantageous is hereby that the user has to know the words exactly.
 - Keywords that are recognised from whole sentences: The speech recognition searches specific keywords in the sentence. The realisation effort of this method is very high, but it offers a comfortable and intuitive operation.
- Request for data: In this element, the same possibilities are given like in the speech commands.
- Dictation of text: Due to the relatively high error rate, the recognition of continuous spoken text is not playing an important role in operation of devices or machines.

Structure elements

- The task of Structure elements is the combination of several basic elements to a complete dialog. It is hereby important to take the users needs into account, e. g. interposed questions.
- Structure elements are:
 - Incremental prompts: This is a method to adept to the experience state of the user. To avoid annoying experienced users, the first prompt of the machine is very short, e. g. “What can I do for you?”. If the user is not familiar to the system, he will not be able to give an adequate answer to the question. In this case, the verbosity level will be incremented and the machine is asking more detailed by offering some keywords.
 - Verbosity Level: This is a system-wide control level for incremental prompts. It allows the speech dialog system to choose extensive or short texts for questions, help or information. It is incremented or decremented by the success of prompts.
 - Structure in context: Due to human specifics, design of the dialogs has to be open, e. g. allowing interposed questions. This means implementation of a memory that latches and recalls information in order to be more flexible in the dialog.

An example for a simple dialog is the setting of a parameter value:

Human: “Set speed to 50%”

Machine: “Speed is set to 50%”

Earcons

Earcons are abstract, musical tones that can be used in structured combinations to create auditory messages. Auditory messages have always been used to provide information, e. g. post-horns or military bugle calls. Since *Earcons* are short, distinctive audio patterns, they can be used for auditory feedback of the user's actions, e. g. in case of recognition of a spoken command or to indicate an error.

Windows® users are familiar to some *Earcons* like the "ding" tone e.g. when trying to activate a disabled object

The use of *Earcons* is reasonable in cases where the feedback may be one of several particular possibilities, e. g.

- Recognized command is executed
- Command not recognized
- An Error occurred
- Starting a robot program

4 Interface objects

In the literature, generally only little information is available on suitable interface objects for the design of touch screen or speech control applications. Often, this information focuses on simple *Push Buttons* and *Edit Boxes*. However, for the design of complex user interfaces (like one for icon-based programming), a larger set of interface objects with differing characteristics is needed. In this chapter, a set of useful interface objects for touch screen and speech interaction is defined. Several of these objects are similar to common controls in standard Microsoft Windows[®] applications, but are somewhat distinct in their appearance.

Interface objects typically belong to one of the following groups:

- Action objects will cause an action immediately after actuation.
- Selection objects will offer or request the user to make a choice.
- Manipulation objects will allow to manipulate values of parameters.
- Supporting objects are objects that help the user in interacting with the dialogs.

The classification of the objects is given for the standard use. Some objects offer to change characteristics by altering properties of the control, e. g. a *Menu* may also contain items that behave like *Check Boxes*.

Since some interface objects have the same shape as others (e. g. different types of buttons), the associated behaviour should be indicated clearly to the user, e.g. by a small graphical symbol. Of course, symbols should be selected so that they are easy to distinguish. In the following paragraphs, symbols are suggested for the different types of buttons.

4.1 Action objects

Operation of an action object will cause an immediate action. Two different types of interface objects are distinguished

- *Push Button and*
- *Jog Button*

These two different interface objects are described in the following sections.

4.1.1 Push Button

Task: Initiation of an action (one shot).

A *Push Button* is a control of mostly rectangular shape. A short text and/or an icon can describe its functionality. Pressing (and releasing) the *Push Button* will trigger the associated function.

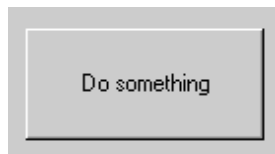


Fig. 4-1: *Push Button*

Usage via touch screen:

- The action will occur after releasing the button (“last contact” (see section 3.1.7), pressing the button does not cause an action yet).

Usage via speech control:

- Calling the caption of the button will trigger the function.
- Guidelines and recommendations:
 - 👍 A *Push Button* is the most common type of a button-shaped interface element. It is suggested not to identify *Push Buttons* by a special symbol, but only the other more specialised button-shaped elements.

4.1.2 Jog Button

Task: Initiation of an on-going action (over a period of time).

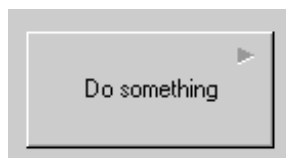


Fig. 4-2: *Jog Button* for inching operations

Usage via touch screen:

- The action sets in after pressing the *Push Button* and will stop in the moment of releasing.

Usage via speech control:

- Speech control is not considered suitable for the operation of jog buttons.
- Guidelines and recommendations:
 - 👉 It is suggested to identify *Jog Buttons* by an triangular symbol “▶” (similar to the “play” symbol found on many consumer electronic devices)

4.2 Selection Objects

Selection objects offer or request the user to make a choice. Five different interface objects of this type are described in this section:

- *Toggle Button (Check Box)*,
- *Multistate Button*,
- *Radio Button*,
- *ListView Control* and
- *TreeView Control*.

4.2.1 Toggle Button (Check Box)

Task: Selection of exactly one of two options (binary selection, yes/no)

Toggle Buttons are a special form of buttons designed to select from several offered options, e. g. properties. They do not initiate actions. *Check Boxes* have the same functionality as *Toggle Buttons*, but a somewhat different graphical appearance.

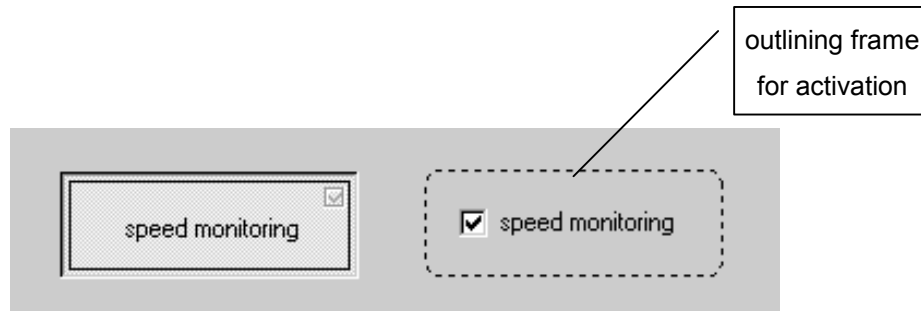


Fig. 4-3: a) left: *ToggleButton*, b) right: *Check Box*

Usage via touch screen:

- *Toggle Buttons*: Pushing the button will toggle the state on/off
- *Check Boxes*: By pressing the check box within the outlining frame, the state of the *Check Box* is toggled (there is no need hit the small box itself very precisely).

Usage via speech control:

- Calling the label of the *ToggleButton* or the *Check Box*, it will be behave as being pushed.

Guidelines and recommendations:

- 👍 Since *Toggle Buttons* are more intuitive to operate, they should be used instead of *Check Boxes*.
- 👍 It is suggested to identify *Toggle Buttons* by a symbol resembling a Windows® check box “ ”.

4.2.2 MultiState Button

Task: Selection of exactly one of several options

A *MultiState Button* is an extended version of a *ToggleButton*. It is used to switch between several options by pressing the button several times. Every time it is pressed, the state is altered to the next defined value. After the last defined value has been reached, the initial state is assumed.

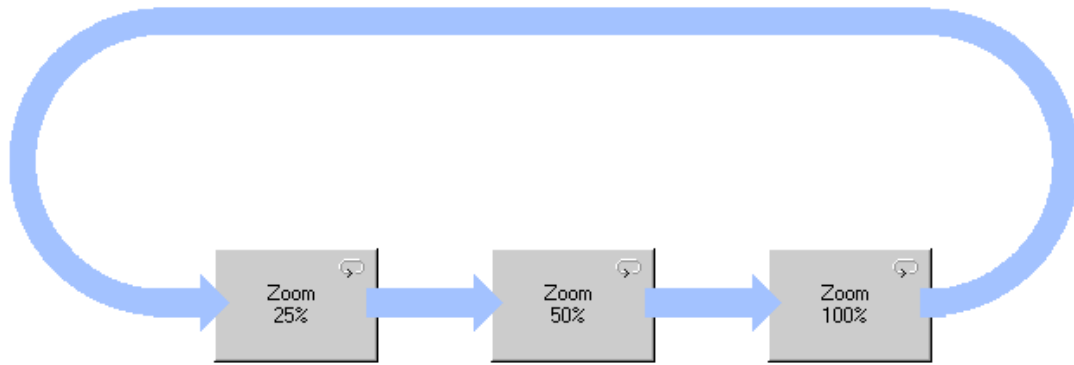


Fig. 4-4: *MultiState Button* with 3 states

Usage via touch screen:

- Each time a *MultiState Button* is pressed, its state is shifted.

Usage via speech control:

- Calling the label of the button, it will behave as being pushed.

Guidelines and recommendations:

- 👍 The actual state of a *MultiState Button* should be indicated visually, e. g. by the text displayed on the button.
- 👍 *MultiState Buttons* with more than four states should be avoided in order not to confuse or annoy the user.
- 👍 *MultiState Buttons* are less commonly used than *Radio Buttons* and should be used in a self-explaining context.
- 👍 It is suggested to identify *MultiState Buttons* by an arrow symbol shaped as an oval “↻”.

4.2.3 Radio Button

Task: Selection of exactly one of several options

A *Radio Button* is a special form of a *Toggle Button*. Like the latter, *Radio Buttons* have a checked state and an unchecked state. However, *Radio Buttons* are always used in groups of two or more. In each group, always exactly one *Radio Button* is checked and the others are unchecked, corresponding to the current state of the selection. Each *Radio Button* is of a button style or may be surrounded by an (“active”) outlining frame, which is visualised by a colour different from the background.

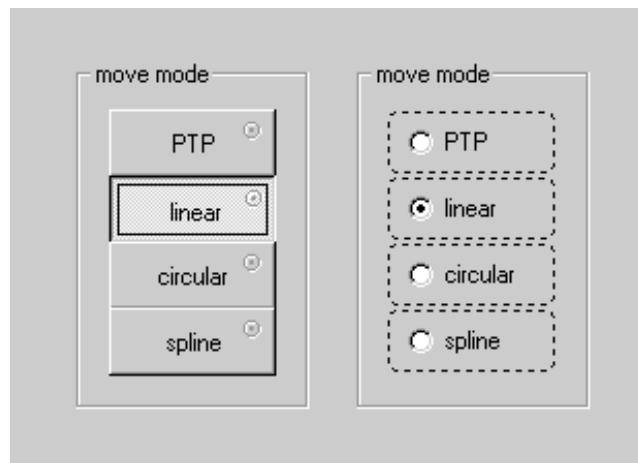


Fig. 4-5: Group of *Radio Buttons*: a) left: button style; b) right: standard

Usage via touch screen:

- By pressing an unchecked element of the *Radio Button* (or within the outlining frame), it becomes checked. The previously checked element of the *Radio Button* becomes unchecked.
- Pressing a checked element of the *Radio Button* does not change anything.

Usage via speech control:

- When calling the label of the *Radio Button* element, it will behave as being pressed.

Guidelines and recommendations:

- 👍 If there is lack of space on the screen and the number of options is small, *MultiState Buttons* may be used instead of *Radio Buttons*.
- 👍 In order to save space on the screen, *Radio Buttons* may also be used inside a *Menu*.

- ☺ It is suggested to identify button style *Radio Buttons* by the Windows® radio button symbol “●”

4.2.4 ListView Control

Task: Selection of one or several items from a list

The *ListView Control* is a rectangular shaped control which contains several items. Each item consists of an icon with an attached describing text. The actually selected item is highlighted.

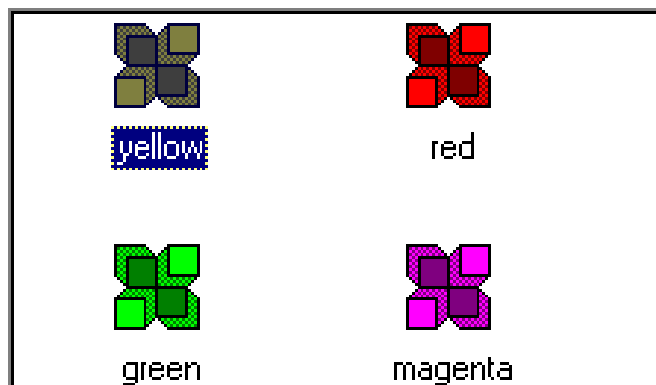


Fig. 4-6: *ListView Control*

Usage via touch screen:

- By pressing within the *ListView Control*, items can be selected or the selected item can be changed. It is also possible to change the preferences so that more than one item can be selected (multi-select function, see section 2.2.2).
- To leave the *ListView Control* without changing selections, the user has to press the touch screen outside the selection area.

Usage via speech control:

- Calling the label of an item, it will behave as being selected.
- Guidelines and recommendations:
 - ☺ Following the general design principles for touch screens, the items should be large and have sufficient space in between (see section 3.1.2).
 - ☺ *ListView Controls* may be intuitive for selecting from a list of items which may vary (e.g. a list of robot programs).

- ☺ If it is intended to display a great number of items (more than the window allows for), the use of a pen as input device can be helpful.
- ☺ Generally, *ListView Control* can be cascaded in a hierarchical manner. However, *TreeView Controls* may be more suited for this purpose.

4.2.5 TreeView Control

Task: Selection of one or several items from a hierarchical list

The *TreeView Control* is a rectangular shaped control which contains several items in a structured, hierarchic manner. Each item is described by an icon and/or an attached describing text. The items are grouped below nodes (“folders” in a file administration environment). The nodes are structured in a hierarchical manner.

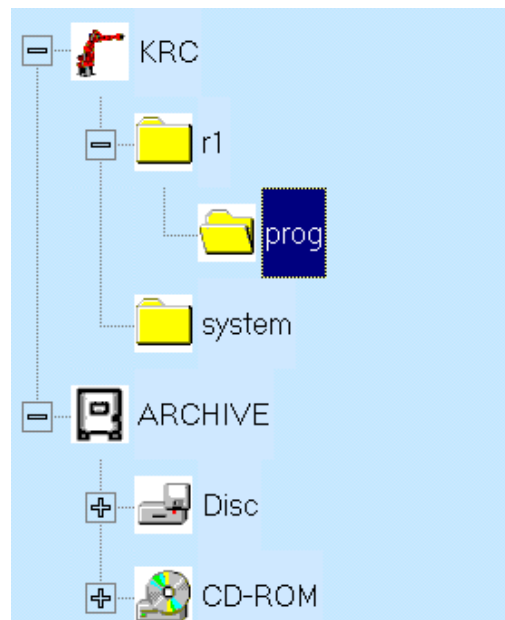


Fig. 4-7: *TreeView Control*

Usage via touch screen:

- An item is selected by pressing on a symbol or text. The item is highlighted.
- By pressing the plus/minus buttons on the left side of the icons, the specific node can be expanded or collapsed to show/hide the child items
- To leave the *TreeView Control* without changing selections, the user has to press the touch screen outside the selection area.

Usage via speech control:

- Calling the label of the item, it will behave as being pressed.
- Navigating through trees by speech control is considered cumbersome and is therefore generally not recommended.

Guidelines and recommendations:

- 👍 Following the general design principles for touch screens, the items should be large and have sufficient space in between (see section 3.1.2).
- 👍 If it is intended to display a great number of items (more than the window allows for), the use of a pen as input device can be helpful.
- 👍 If there is no ambiguity, the user may call a (hidden) child item somewhere in the tree directly via speech input. On the display, the corresponding nodes should automatically be opened.
- 👍 The whole path of the node currently selected should be visible to the user.

4.3 Manipulation Objects

Manipulation objects allow to manipulate values of parameters. In this section, five different interface objects are explained:

- *Edit Box*,
- *On-screen Keyboard*,
- *Spin Button*,
- *Analog Value Bar* and
- *Slider*.

4.3.1 Edit Box

Task: Allow textual or numerical input

The *Edit Box* contains text which can be entered by the user. In most cases, it will be of rectangular shape. The text is displayed inside the *Edit Box*. Entered text will replace any selected text.

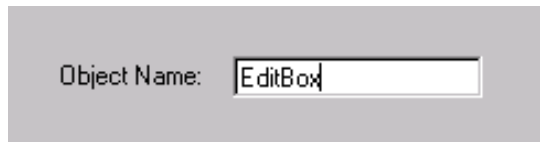


Fig. 4-8 *Edit Box*

Usage via touch screen:

- For entering text, the *Edit Box* object is selected. A cursor will appear in the *Edit Box* at the position where characters will be entered, and the prior text will appear as a selection. At the same time, an *On-screen Keyboard* will appear.
- Pressing the <RETURN> key on the keyboard will accept the text that has been entered. The *On-screen Keyboard* will disappear, and the new text is stored in the *Edit Box*.
- Pressing the <ESC> key on the keyboard will abort the process. The original text is restored in the *Edit Box*.

Usage via speech control:

- Calling the label of the *Edit Box* will activate it.
- Speaking a text will have the same effect as entering it via the *On-screen Keyboard*. While the *Edit Box* is activated, no speech input for other commands is available.
- To close the *Edit Box* and accept all entries, the user has to speak a key word (e.g. “close”).
- To abort, the user has to speak a different key word (e.g. “exit”).

Guidelines and recommendations:

- 👍 Texts that have to be entered by the keyboard should be as short as possible.
- 👍 Upon activating an *Edit Box*, the behavior of the cursor should be determined by the following pattern:
 - Entering the ← cursor key will move the cursor to the beginning of the text and deselect it.
 - Entering the → cursor key will move it to the end of the text and deselect the text.

- ☺ If the text is a numeric value, a combination with a *Spin Button* (see section 4.3.3) or a *Analog Value Bar* (sect. 4.3.4) could be a good alternative.

4.3.2 On-screen Keyboard

Task: Enter text into an *Edit Box*

The *On-screen Keyboard* is a combination of several *Push* and *Toggle Buttons* and is used to enter text and numbers via touch screen.



Fig. 4-9 *On-screen Keyboard*

Usage via touch screen:

- Equivalent to a common (“hard”) keyboard.

Usage via speech input:

- Not applicable.

Guidelines and recommendations:

As pressing several keys in parallel is not possible on a touch screen, “Shift”, “Ctrl” and “Alt” keys have to be implemented as *Toggle Buttons*.

4.3.3 Spin Button

Task: Adjust the numeric value of a parameter

The *Spin Button* is a combined control consisting of two *Jog Buttons*. Minimum and maximum values may be defined between which the *Spin Button* control will operate. In most cases, it is attached to an *Edit Box* to display and/or alter its value. In any cases, a visual feedback of the value that is manipulated should be provided.



Fig. 4-10 *Spin Button*, combined with an *Edit Box*

Usage via touch screen:

- As long as one of the two buttons is pressed, the current value will continue to increase/decrease.

Usage via speech control:

- Operation of the *Spin Button* via speech control would be cumbersome and is therefore not recommended.

4.3.4 Analog Value Bar

Task: Adjust the numeric value of a parameter

The *Analog Value Bar* consists of a triangular object which is divided into several sections. The actual value is displayed by colouring the "active" sections of the triangle. An *Edit Box* may be combined with this object. It displays the current value in numerical format and allows inputs from the *On-screen Keyboard*. The combination is advantageous, as it allows for fast graphical input of approximate values and offers the possibility to adjust the value precisely by means of the *Edit Box* if necessary.

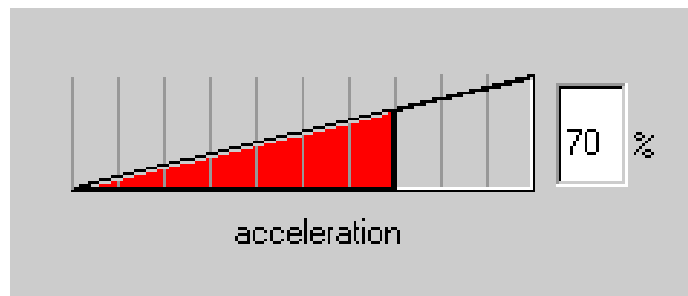


Fig. 4-11: Analog Value Bar combined with an Edit Box.

Usage via touch screen:

- Pressing a section will adjust the parameter to the corresponding value, e.g. tipping into the middle will set the value to the middle of the value range. At the same time, the value in the Edit Box will be adjusted.
- Selecting the *Edit Box* will allow to enter precise numerical values by means of the *On-screen Keyboard*

Usage via speech control:

- The *Analog Value Bar* can be operated via speech control by calling the label (normally the name of the parameter to be adjusted) to activate the *Edit Box* and then speaking the value.

4.3.5 Slider

Task: Adjust the numeric value of a parameter

A *Slider* consists of a bar with an attached grip which are aligned horizontally or vertically. While moving the grip between the extremes, the variable controlled by the slider will change correspondingly. For example, in the leftmost position the controlled variable will be 0, in the rightmost position it will be 100 and between the extreme positions, the controlled variable increases linearly from left to right.

In most cases, a short text describing the controlled variable will be added to the *Slider*. In many cases, information about the current value of the variable will also be displayed.



Fig. 4-12 Horizontal *Slider*

Usage via touch screen:

- The grip can be directly moved in both directions by dragging it with a pen or the fingertip.
- It is also possible to move the grip in larger incremental steps by pressing the bar at the side of the grip it is supposed to move.

Usage via speech control:

- Usage of the *Slider* via speech control would be cumbersome and is therefore not recommended.

Guidelines and recommendations

- 👍 The size of grip and bar needs to be chosen large enough for touch screen operation (see section 3.1.2).

- 👍 Due to the large minimum size of a *Slider*, it appears to be more suitable for pen-based touch screen applications.
- 👍 Since “drag & drop” is considered problematic with touch screens, an *Analog Value Bar* may be the better choice in many cases.

4.4 Supporting Objects

Supporting objects are objects that help the user in interacting and navigating with the dialogs. Four types of interface objects belong to this group:

- *Menu*,
- *Tab Control*,
- *Grabber* and
- *Scroll Bar*.

4.4.1 Menu

Task: Selection of one (of several) actions or options.

A *Menu* is a temporarily visible control which contains a list of actions or options to be chosen from. In most cases, each action or option will be described by a short text and/or an icon.

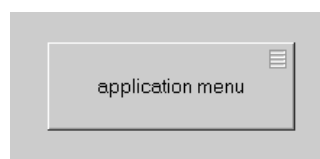


Fig. 4-13: *Menu Button* indicating a *Menu* underneath

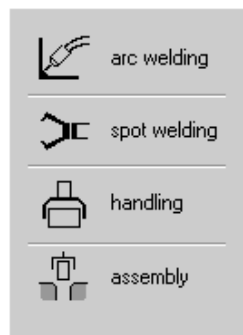


Fig. 4-14: *Menu* after opening

Usage via touch screen:

- Pressing the *Menu Button* will open the assigned *Menu*.
- After the user has selected one of the available actions or options, the *Menu* will close automatically and the action or option will be executed.
- To close the *Menu* without selecting a command, the user has to press the touch screen outside the menu area.

Usage via speech control:

- Calling the caption of the *Menu Button* will have the same effect as selecting the *Menu Button* on the touch screen.
- Calling the label of a *Menu* entry will execute the associated action or option.
- To close the *Menu* without selecting a command, the user has to speak a key word (e.g. “exit”).

Guidelines and recommendations:

- 👍 It is suggested to identify menu buttons by a ladder-like symbol “☰” (resembling the list of entries in a vertical menu).
- 👍 For touch screen operation, the selection area of the objects has to be large enough for operation. Please refer to section 3.1.2 for appropriate sizing.
- 👍 For speech control, it should be possible to call menu entry directly without opening the *Menu* itself.
- 👍 If a *Menu* is structured in a hierarchical (tree-like) manner and leads to sub-menus, the whole path should be made visible to the user.

4.4.2 Tab Control

Task: Select exactly one of several windows

A *Tab Control* is a combination of several buttons, in which each button is associated to an own window. Each tab has short text and/or an icon describing the associated window. The *Tab Controls* can be positioned on each side of the window area.

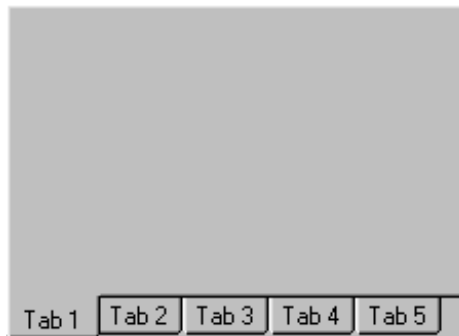


Fig. 4-15 *Tab Control*

Usage via touch screen:

- When pressing a tab, the window is replaced by the window associated to the pressed *Tab Control*.

Usage via speech control:

- Calling the label of a tab via speech will have the same effect as pressing it.

Guidelines and recommendations:

- 👍 Following the general design principles for touch screens, the tabs should be large enough (see section 3.1.2).

4.4.3 Grabber

Task: Adjust the viewing area of a window

The *Grabber* is a hand-shaped figure that allows to move the view of a window which is not big enough to show its whole content (e.g. a *ListView* or *TreeView Control* with many items, or a large text document).



Fig. 4-16 *Grabber*

Usage via touch screen:

- Pressing into a free space in the window and dragging the finger into any direction will scroll the content by the same amount.

- Dragging the finger to an edge of the window and leaving the finger pressed in this position will start to scroll the content automatically towards the direction of the edge.

Usage via speech control:

- The *Grabber* is a spatial operation tool especially for graphical control interfaces. It is not suitable for speech control.

Guidelines and recommendations:

- 👍 In the case that items in the window could be selected accidentally during the use of the *Grabber*, the capability should be provided to switch the selection in the window off through a dedicated *Grabber* mode.
- 👍 Window contents that are much too large to fit into the window should be avoided.

4.4.4 Scroll Bar

Task: Adjust a viewing area.

The *Scroll Bar* is a combination of a *Spin Button* and a *Slider*. The buttons are positioned at each end of the *Slider*. *Scroll Bars* may be arranged horizontally or vertically. They are most commonly used to adjust the viewable area in a window that does not fit into the screen.

The position of the movable grip indicates the value relatively to its definition range.



Fig. 4-17 Scroll Bar

Usage via touch screen:

- The grip can be directly moved in both directions by dragging it with a pen or the fingertip.
- As long as one of the *Spin Buttons* is pressed, the grip is slowly moving into the corresponding direction.
- It is also possible to move the grip in larger increments by pressing the bar at the side of the grip it is supposed to move.

Usage via speech control:












- Usage of the *Scroll Bar* via speech control would be cumbersome and is therefore not recommended.

Guidelines and recommendations:

- ☞ The size of the *Scroll Bar* needs to be chosen large enough for touch screen operation (see section 3.1.2).
- ☞ Due to the large minimum size of a *Scroll Bar*, it appears to be more suitable for pen-based touch screen applications.
- ☞ In many cases, the more intuitive way to move the contents of a window is to use the *Grabber* function (see sect. 4.4.2)















5 Graphical symbols and icons















The icons listed below are derived from the AMIRA styleguide [AMIRA] and should be used as icons for menus and program commands.















Icon Description	Main Menu	Sub Menu	Program Icons
Program Start			
Program End			
Macro (Closed)			
Macro Begin			
Macro End			
Program Control			
Program Flow			
If			
Then			
Else			
Endif			









Icon Description	Main Menu	Sub Menu	Program Icons
Switch			
Case			
Default			
Endswitch			
For			
Endfor			
Loop			
Endloop			
While			
Endwhile			
Repeat (Do)			
Until (While)			
Exit Loop (Break)			
Goto			

Icon Description	Main Menu	Sub Menu	Program Icons
Subroutine Call			
Multitasking			
Wait			
Delay			
Wait for Digital Input			
Wait for Digital Output			
Wait for Condition			
Interrupt Handling			
Enable Interrupt			
Disable Interrupt			
Interrupt On			
Interrupt Off			
Return from Interrupt Service Routine			
Brake during Interrupt Service Routine			



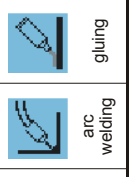
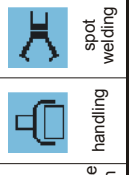

















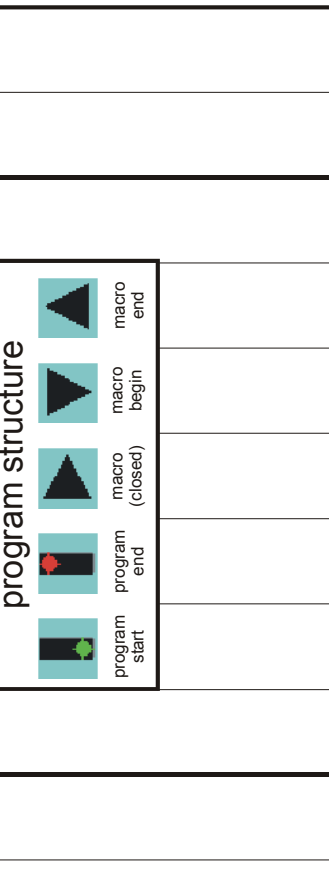
Icon Description	Main Menu	Sub Menu	Program Icons
Resume from Interrupt Service Routine			
Error Handling			
Program Stop			
Robot Motion			
Motion Commands			
Point-To-Point Motion			
Linear Motion			
Circular Motion			
Motion Settings			
Set Speed / Acceleration			
Set Orientation			
Set Base / Tool			
Sensor Motion			
Trigger			

Icon Description	Main Menu	Sub Menu	Program Icons
Input / Output			
Input			
Set / Reset Digital Input			
Set / Reset Analog Input			
Output			
Set / Reset Digital Output			
Set / Reset Analog Output			
Set / Reset Pulse Output			
Reset All Outputs			
Computer Communication			
Application			
Mutual Exclusion Of Workspace			
Mutual Exclusion Of Workspace Enter			
Mutual Exclusion Of Workspace Exit			

Icon Description	Main Menu	Sub Menu	Program Icons
Handling			
Open Gripper			
Close Gripper			
Check Gripper			
Spot Welding			
Spot Welding Open			
Spot Welding Close			
Arc Welding			
Arc On			
Arc Off			
Arc Switch			
Gluing			
Painting			
Assembly / Disassembly			

Icon Description	Main Menu	Sub Menu	Program Icons
Others			
Specific Code			
Data Handling: Variable Assignment			
Data Handling: Geometric Assignment			
User Specific Code			
User Defined Comment			
Mathematics			
Functions			

Overview of Icons and Symbols for Menus and Program Commands

program control	robot motion	input / output	application	others
<p>program flow</p> 	<p>program flow</p> 	<p>wait</p> 	<p>interrupt handling</p> 	<p>error handling</p> 
<p>motion cmds.</p> 	<p>motion settings</p> 	<p>sensor motion</p> 	<p>workspace exclusion</p> 	<p>computer comm.</p> 
<p>input</p> 	<p>output</p> 	<p>handling</p> 	<p>spot welding</p> 	<p>arc welding</p> 
<p>gluing</p> 	<p>painting</p> 	<p>(dis-) assembly</p> 	<p>specific code</p> 	<p>mathematics</p> 
<p>program structure</p> 				<p>others</p> 

6 References

- [AMIRA] AMIRA consortium: *User Interface Style Guide for Robot System Applications*. Final report of the ESPRIT Project 22646 “Advanced Man Machine Interfaces for Robot System Applications” AMIRA. Distributed on behalf of the project consortium by Fraunhofer IPK, Berlin, June 1999.
- [Becker] Becker, B.: *Entwicklung eines Bedienkonzeptes für Spracheingaben*. Web pages of MMI-Interaktiv, Technische Universität Berlin, Zentrum Mensch-Maschine-Systeme. See http://www.mmi-interaktiv.de/ausgaben/07_01/becker.pdf
- [Elo] *Touchscreen application tips*. Web pages of Elo TouchSystems Inc., Fremont, CA, USA. See <http://www.elotouch.com/Support/10tips.asp>
- [Blattner] Blattner, M.M., Sumikawa, D.A. and Greenberg, R.M.: *Earcons and Icons: Their Structure and Common Design Principles*, Human-Computer Interaction 4(1), 1989, p. 11-44
- [Born] Born, H.: *Graphische online Programmierung von Industrierobotern*, Elektronik 15. März 2000, WEKA Fachzeitschriften-Verlag, Poing
- [ISO 15187] ISO 15187:2000. *Manipulating industrial robots - Graphical user interfaces for programming and operation of robots (GUI-R)*. ISO standard currently under development.
- [Krauß] Krauß, L.: *Einsatz von Touchscreens auf Basis von Flachbildschirmen*. Universität Kaiserslautern, Lehrstuhl für Produktionsautomatisierung (pak), 1998.
- [MUSIST] *Final Online Styleguide*. Deliverable D20 of the ACTS Project AC010 „Multimedia User Interfaces For Interactive Systems and TV“, GSM GmbH, Stuttgart, June 1998. See <http://www.gsm.de/musist>
- [VDI 3850-3] *Dialoggestaltung für Touchscreens - Nutzergerechte Gestaltung von Bediensystemen für Maschinen*. VDI/VDE 3850 Blatt 3 (in German language). Internal draft version, May 2001.